

Supercomputing Frontiers and Innovations

2025, Vol. 12, No. 2

Scope

- Future generation supercomputer architectures
- Exascale computing
- Parallel programming models, interfaces, languages, libraries, and tools
- Supercomputer applications and algorithms
- Novel approaches to computing targeted to solve intractable problems
- Convergence of high performance computing, machine learning and big data technologies
- Distributed operating systems and virtualization for highly scalable computing
- Management, administration, and monitoring of supercomputer systems
- Mass storage systems, protocols, and allocation
- Power consumption minimization for supercomputing systems
- Resilience, reliability, and fault tolerance for future generation highly parallel computing systems
- Scientific visualization in supercomputing environments
- Education in high performance computing and computational science

Editorial Board

Editors-in-Chief

- **Jack Dongarra**, University of Tennessee, Knoxville, USA
- **Vladimir Voevodin**, Moscow State University, Russia

Editorial Director

- **Leonid Sokolinsky**, South Ural State University, Chelyabinsk, Russia

Associate Editors

- **Pete Beckman**, Argonne National Laboratory, USA
- **Arndt Bode**, Leibniz Supercomputing Centre, Germany
- **Boris Chetverushkin**, Keldysh Institute of Applied Mathematics, RAS, Russia
- **Alok Choudhary**, Northwestern University, Evanston, USA
- **Alexei Khokhlov**, Moscow State University, Russia
- **Thomas Lippert**, Jülich Supercomputing Center, Germany

- **Satoshi Matsuoka**, Tokyo Institute of Technology, Japan
- **Mark Parsons**, EPCC, United Kingdom
- **Thomas Sterling**, CREST, Indiana University, USA
- **Mateo Valero**, Barcelona Supercomputing Center, Spain

Subject Area Editors

- **Artur Andrzejak**, Heidelberg University, Germany
- **Rosa M. Badia**, Barcelona Supercomputing Center, Spain
- **Franck Cappello**, Argonne National Laboratory, USA
- **Barbara Chapman**, University of Houston, USA
- **Yuefan Deng**, Stony Brook University, USA
- **Ian Foster**, Argonne National Laboratory and University of Chicago, USA
- **Geoffrey Fox**, Indiana University, USA
- **William Gropp**, University of Illinois at Urbana-Champaign, USA
- **Erik Hagersten**, Uppsala University, Sweden
- **Michael Heroux**, Sandia National Laboratories, USA
- **Torsten Hoefler**, Swiss Federal Institute of Technology, Switzerland
- **Yutaka Ishikawa**, AICS RIKEN, Japan
- **David Keyes**, King Abdullah University of Science and Technology, Saudi Arabia
- **William Kramer**, University of Illinois at Urbana-Champaign, USA
- **Jesus Labarta**, Barcelona Supercomputing Center, Spain
- **Alexey Lastovetsky**, University College Dublin, Ireland
- **Yutong Lu**, National University of Defense Technology, China
- **Bob Lucas**, University of Southern California, USA
- **Thomas Ludwig**, German Climate Computing Center, Germany
- **Daniel Mallmann**, Jülich Supercomputing Centre, Germany
- **Bernd Mohr**, Jülich Supercomputing Centre, Germany
- **Onur Mutlu**, Carnegie Mellon University, USA
- **Wolfgang Nagel**, TU Dresden ZIH, Germany
- **Edward Seidel**, National Center for Supercomputing Applications, USA
- **John Shalf**, Lawrence Berkeley National Laboratory, USA
- **Rick Stevens**, Argonne National Laboratory, USA
- **Vladimir Sulimov**, Moscow State University, Russia
- **William Tang**, Princeton University, USA
- **Michela Taufer**, University of Delaware, USA
- **Andrei Tchernykh**, CICESE Research Center, Mexico
- **Alexander Tikhonravov**, Moscow State University, Russia
- **Eugene Tyrtysnikov**, Institute of Numerical Mathematics, RAS, Russia
- **Roman Wyrzykowski**, Czestochowa University of Technology, Poland
- **Mikhail Yakobovskiy**, Keldysh Institute of Applied Mathematics, RAS, Russia

Technical Editors

- **Andrey Gogachev**, South Ural State University, Chelyabinsk, Russia
- **Yana Kraeva**, South Ural State University, Chelyabinsk, Russia
- **Dmitry Nikitenko**, Moscow State University, Moscow, Russia
- **Mikhail Zymbler**, South Ural State University, Chelyabinsk, Russia

Editor's Introduction for Special Issue on
results obtained using computing facilities of the MSU Supercomputer Center

Moscow State University has always been a leader in the use of powerful computing systems. Over the past 15 years, the Moscow State University supercomputer complex has provided access to nearly 2,500 scientific and educational projects, carried out by approximately 6,000 users from 500 organizations.

Currently, the supercomputer complex provides services to nearly 1,500 users from Moscow State University, RAS institutes, and other organizations engaged in 500 different fundamental and applied projects. Each research group granted access to the SCC resources submits its research results annually. Each report undergoes a thorough review, and the results influence the allocation of quotas and priorities for future projects.

This issue invited participation from relevant projects from various research fields, carried out using the computing resources of the MSU Supercomputer Center, and, most importantly, from those that have received support from the Russian Science Foundation for research using scientific infrastructure facilities.





Contents

Bacterial Mini Microtubule as a Minimal Model System for Exploring Dynamic Instability Using Molecular Dynamics Simulations V.A. Fedorov, E.G. Kholina, N.B. Gudimchuk, I.B. Kovalenko	5
GPU Implementation of Zippel Method for Feynman Integral Reconstruction A.V. Smirnov, B.I. Rozhnov, Vad.V. Voevodin	17
Mastering 3D-detection of Extensive Air Showers in Cherenkov Light E.A. Bonvech, O.V. Cherkesova, D.V. Chernov, E.L. Entina, V.I. Galkin, V.A. Ivanov, T.A. Kolodkin, N.O. Ovcharenko, D.A. Podgrudkov, T.M. Roganova, M.D. Ziva	30
Prospects for Improving Computational Efficiency of Hydrodynamic Simulations on Supercomputers by Increasing the Number of GPUs per Compute Node S.S. Khrapov, E.O. Agafonnikova, A.V. Khoperskov	43
Comparison of Quantum-Chemical Programs and Methods for the Calculation of Enthalpies of Formation of High-Energy Tetracyclic Compounds V.M. Volokhov, V.V. Parakhin, E.S. Amosova, D.B. Lempert, Vl.V. Voevodin	60
Dynamic Content-Oriented Indexing and Replication for High-Performance Storage and Analysis of Big Data in the IPFS Network M.V. Shevarev, S.V. Suvorov	74



This issue is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

Bacterial Mini Microtubule as a Minimal Model System for Exploring Dynamic Instability Using Molecular Dynamics Simulations

Vladimir A. Fedorov¹ , Ekaterina G. Kholina¹ ,
Nikita B. Gudimchuk^{2,3,4} , Ilya B. Kovalenko¹ 

© The Authors 2025. This paper is published with open access at SuperFri.org

Large scale computational modeling has been fruitfully applied to explore microtubules – an essential component of the cellular skeleton – for over two decades. In this paper, we describe simulations of a yet computationally unexplored minimalistic system of the bacterial mini microtubule, using the high performance resources of Lomonosov Moscow State University. We highlight similarities between the eukaryotic and bacterial microtubules at the protofilament level, the size and stability of the entire mini microtubule system and the computational benefits of using the bacterial mini microtubule as a minimal model to understand dynamic instability. Our results are discussed in the context of a bigger picture of the evolution of molecular dynamics simulations, aiming to understand microtubules, illustrating how the sophistication and scale of the computational efforts increased over the years.

Keywords: bacterial microtubule, Lomonosov-2, computational performance, multi-scale simulations, molecular dynamics.

Introduction

Tubulins are among the most evolutionarily conserved proteins in eukaryotic cells, from yeast to mammals, which suggests that any considerable changes in their sequence may be lethal to the organism [48]. The reason is that they form essential structures, microtubules, which are critical for numerous intracellular functions, including transport and cell division [6]. Structurally, microtubules are composed of tubulin dimers arranged into linear chains called protofilaments. Typically, 13 protofilaments laterally associate to form a hollow tube (the microtubule), which can elongate by incorporating tubulin dimers at its so-called (+)-end or shorten by loss of subunits. One of the remarkable features of these polymers is their ability to alternate between phases of elongation and shortening, with abrupt transitions between them [29]. This behavior, termed “dynamic instability”, is physiologically important for the constant re-modeling of microtubule networks in cells, the continuous search-and-capture of chromosomes during mitosis, and many other essential processes [17]. Due to their dynamic instability, microtubules also act as active force generators within cells [15]. The non-equilibrium behavior of all microtubules is fueled by the energy of guanosine triphosphate (GTP) hydrolysis. GTP molecules bind to tubulin dimers in solution and provide energy for conformational changes within tubulin through the chemical reaction that cleaves a phosphate group from GTP, converting it into guanosine diphosphate (GDP). The precise mechanisms by which this biochemical reaction drives the mechanical destabilization of microtubules and how microtubules generate forces remain long-standing questions in cytoskeletal research [4, 17]. These are some of the problems that have long fascinated scientists and have driven numerous experimental, theoretical, and computational studies. Interestingly, while most bacterial species do not possess microtubules, a unique type has been

¹Department of Biology, Lomonosov Moscow State University, Moscow, Russia

²Department of Physics, Lomonosov Moscow State University, Moscow, Russia

³Center for Theoretical Problems of Physicochemical Pharmacology, Moscow, Russia

⁴Pskov State University, Pskov, Russia

identified in *Prostheco bacter* species [22]. These bacterial microtubules, composed of the tubulin homologs BtubAB, consist of only 4–5 protofilaments but display remarkably similar dynamic properties to those of eukaryotic microtubules [5]. Here, we use molecular dynamics simulations to study a short bacterial microtubule and its basic building block – a four-subunit protofilament. Our results show that this system behaves in a way similar to eukaryotic microtubules, making it a useful model for studying dynamic instability.

1. Methods

1.1. Molecular Systems

Molecular model of the four-stranded mini microtubule (model 2) formed by bacterial tubulin-like BtubAB proteins was based on 5o09 PDB structure [5]. The PDB structure also contained a regulatory BtubC subunit, which we did not include in our simulation. BtubAB proteins were placed in a cubic reaction volume with periodic boundary conditions and 31.72 nm side filled with TIP3P water and ions. The total number of atoms in the system was 3134589 where 110400 atoms belong to tubulin. Additionally, we created the molecular model (model 1) of one protofilament of bacterial mini microtubule based on the same PDB structure 5o09. This model included 544280 atoms where 27599 atoms belong to tubulin in a $16.65 \times 15.3 \times 21.73$ nm cubic reaction volume with TIP3P water and ions. In each model, we added unresolved mobile amino acid chains, using the Modeller program [45]. We used Propka [35] to calculate the degree of protonation of amino acid residues and Dowser [32] to identify and solvate cavities inside the protein. In both models every BtubA and BtubB subunit was bound to a GDP molecule. The size of the reaction volume was set in such a way that the distance from the protein surface to the nearest box boundary was not initially less than two nanometers. The ionic strength 0.1 M of the solution was achieved by adding K^+ and Cl^- ions in a way that the total charge of the system was zero.

1.2. Molecular Dynamics Simulations

Simulations were performed using hybrid computational architecture and the GROMACS 2022.4 software package [1] with the CHARMM27 force field [25, 26]. The parameters of the GDP molecule were taken from the CHARMM27 force field, and the parameters of their phosphate groups were set in accordance with [37]. The steepest descent algorithm was used to minimize energy of each system. After this, a two-step equilibration was applied: 1-ns-long simulation with constrained positions of all heavy protein atoms at constant pressure and temperature, and 5-ns-long simulation with constrained positions of protein backbone atoms, using the Berendsen barostat and thermostat. The production simulation runs were carried out in the NPT ensemble at 300 K, using the Parrinello–Rahman algorithm [36] and the V-rescale thermostat. Totally we produced three trajectories of the bacterial protofilament and one trajectory of the bacterial microtubule, each trajectory no less than 1 μs . All-bond P-LINCS constraints and mass rescaling (partial transfer of mass from heavy atoms to bound hydrogens [11]) allowed molecular dynamics simulations with 4 fs time step. Pymol (The PyMOL Molecular Graphics System, Version 2.0 Schrodinger, LLC) was used for visualization.

1.3. Analysis

Calculation of bend and twist angles of a tetrameric protofilament of bacterial mini microtubule was made using custom python scripts inside Pymol in the way it had been described in our previous work on eucaryotic tubulins [10]. To do this, a Cartesian coordinate system was associated with a bacterial microtubule structure. Then a bacterial tubulin tetramer was aligned onto the microtubule fragment by its reference subunit. So, the reference subunit became aligned along the microtubule-bound coordinate system. To determine the orientation of the next tubulin subunit relative to the reference subunit, another microtubule fragment was aligned onto the next tubulin subunit, producing another Cartesian coordinate system. After this, we calculated the Euler angles of the rotation of the second coordinate system relatively to the first one.

2. Results

2.1. Exploring the Behavior of the Bacterial Microtubule System through Molecular Dynamics Simulations

Bacterial microtubules consist of only four protofilaments and are known to display dynamic instability behavior very similar to that of eukaryotic microtubules [5]. Moreover, they offer additional advantages due to the ease of re-engineering and modification, as bacterial systems are generally easier to manipulate genetically.

To examine the behavior of this system *in silico*, we prepared two molecular models: (1) a minimal protofilament, composed of two bacterial tubulin dimers, BtubAB, where each BtubAB monomer contained a GDP molecule in the nucleotide binding site; and (2) a minimal fragment of the entire four-protofilament microtubule, based on the CryoEM structure [5] (Fig. 1). We should note that the CryoEM structure also includes a regulatory BtubC subunit. However, since our goal was to model a minimalistic system, investigating the impact of this regulatory subunit on the microtubule was beyond the scope of the present study, and we therefore excluded it from our simulations.

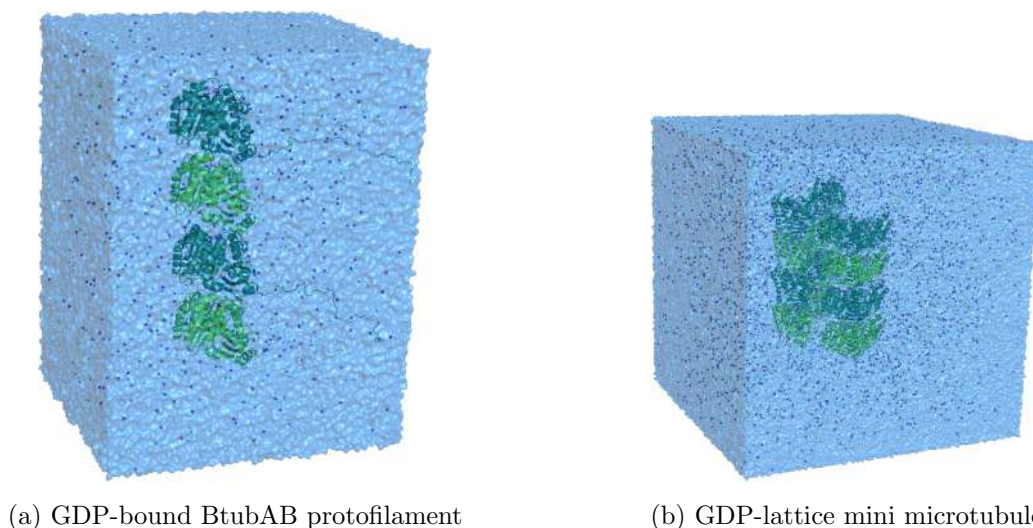


Figure 1. Molecular models of the GDP-bound BtubAB protofilament in $16.65 \times 15.3 \times 21.73 \text{ nm}^3$ molecular dynamics cell and the GDP-lattice mini microtubule in $31.72 \times 31.72 \times 31.72 \text{ nm}^3$ molecular dynamics cell filled with TIP3P water and ions. α -tubulins are shown in dark green, β -tubulins in light green

We performed three independent microsecond-long molecular dynamics simulations of molecular model (1) and one such simulation of molecular model (2). These timescales match the current state-of-the-art in computations with eukaryotic microtubules, as we describe in the Discussion section. The simulations have demonstrated that the bacterial GDP-bound protofilaments tend to relax into outwardly curved shapes, very similar to those observed in eukaryotic tubulin (Figs. 2, 3). This result is consistent with the expectation that the mechanism driving the switch of bacterial microtubules to disassembly is analogous to that of dynamic eukaryotic microtubules: they accumulate energy from GTP hydrolysis in the form of mechanical strain, which is released during the depolymerization of GDP-bound subunits.

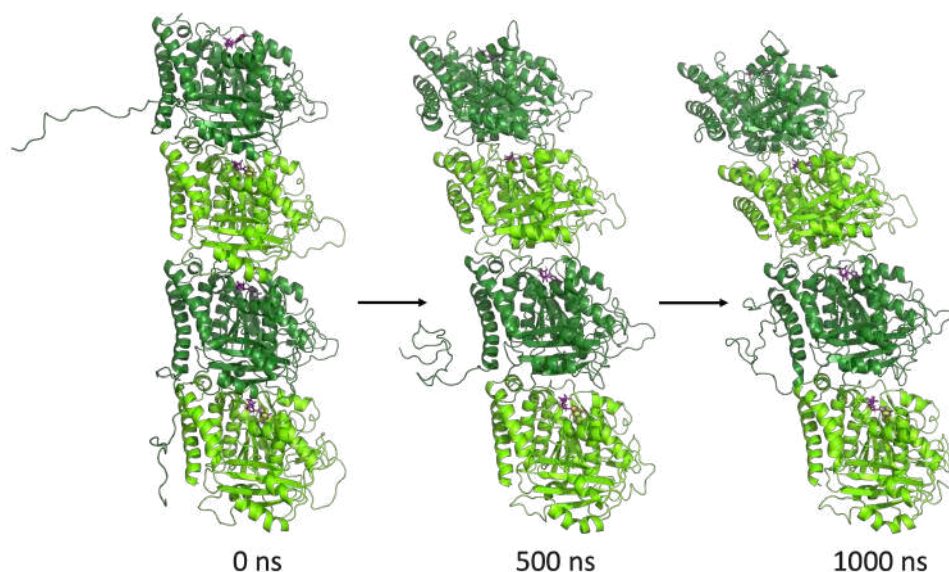


Figure 2. Results of molecular dynamics simulations of the GDP-bound BtubAB protofilament. Three snapshots of the protofilament in the onset of the simulation, upon 500 ns of the simulation and upon 1000 ns of the simulation. α -tubulins are shown in dark green, β -tubulins in light green

Interestingly, our simulations of the four-protofilament mini microtubule revealed that it remained stable throughout one-microsecond simulation despite the tendency of individual protofilaments to become curved (Figs. 4, 5). We believe this result is expected, given the much slower rate of microtubule depolymerization observed in experiments [5]. Notably, in analogous simulations of eukaryotic microtubules [21, 47], lateral bonds are lost considerably faster – within one microsecond – suggesting an overly rapid depolymerization. In our opinion, this puzzling behavior has not yet received sufficient attention in the literature.

2.2. Computational Efficiency of the Molecular Dynamics Simulations of BtubAB Microtubules

We have previously investigated the computational efficiency of molecular dynamics simulations with GROMACS on diverse computational architectures, such as the Lomonosov-2 supercomputer at Moscow State University [43] and others, using various molecular systems, including tubulins [7–9] (Fig. 6). This analysis yields a simulation efficiency of 1.2 ns/day on one node of the Lomonosov-2 supercomputer for our whole bacterial microtubule system, comprising 3.5 million atoms. Using our newest server with RTX 4090, the performance can be increased by

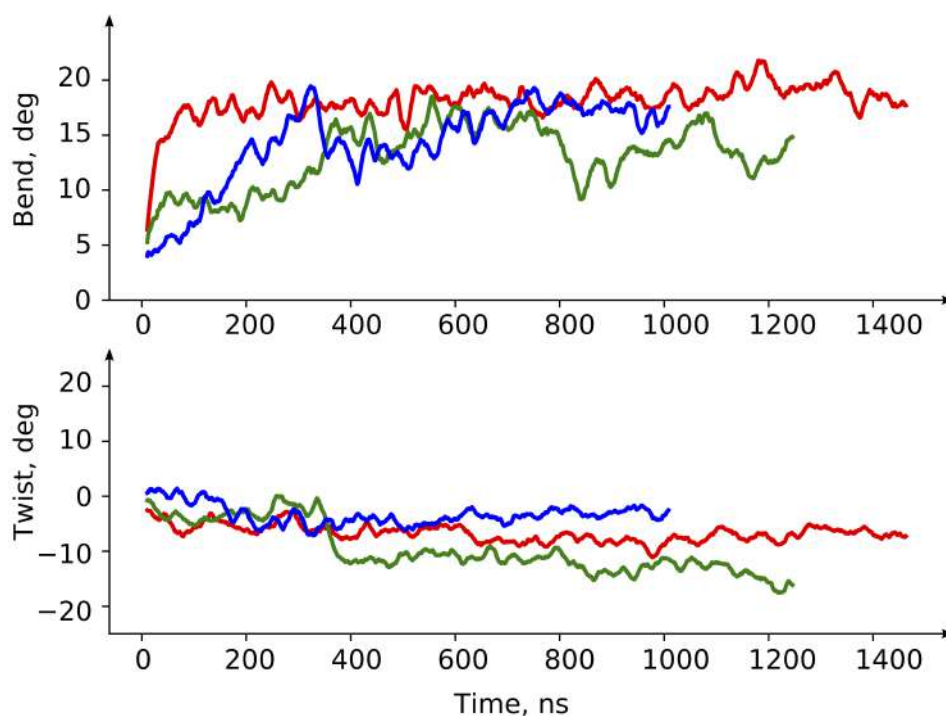


Figure 3. Averaged inter dimer bend and twist angles of the BtubAB monomers relative to each other as a function of the molecular dynamics simulation time. Colors show results from independent simulation runs

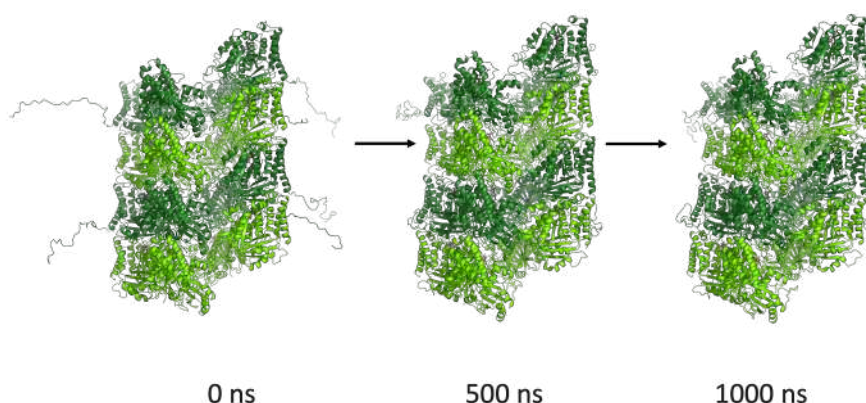


Figure 4. Results of molecular dynamics simulations of mini microtubule, containing four laterally attached GDP-bound BtubAB protofilaments. α -tubulins are shown in dark green, β -tubulins in light green

an order of magnitude, up to 12 ns/day. Importantly, the system can be further optimized by reducing its size if simulating the long unstructured tails on the B subunit of bacterial tubulin is not necessary. In that case, a simulation box with dimensions of only $18 \times 18 \times 25 \text{ nm}^3$ may be sufficient, reducing the computational cost by approximately 4-fold.

Discussion

During the last two decades, multi-scale computational modeling, including efforts from our group, has provided important insights into our understanding of microtubules and their

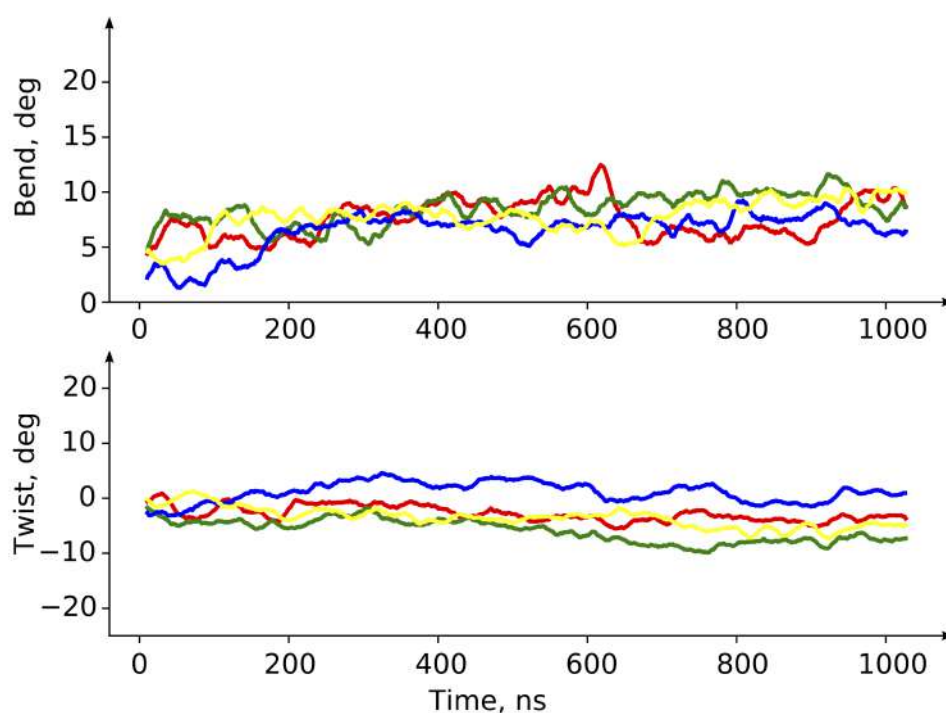


Figure 5. Averaged inter dimer bend and twist angles of the BtubAB monomers relative to each other as a function of the molecular dynamics simulation time. Colors show results for each protofilament

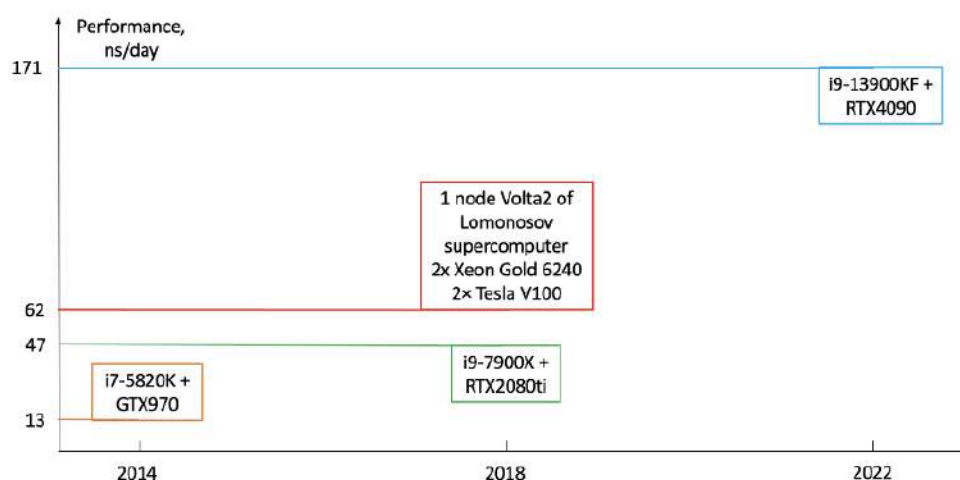


Figure 6. Timeline of performance (ns/day) evolution for molecular dynamics simulations on a single-node server for tubulin tetramer depending on various combinations of CPUs and GPUs with 2 fs timestep. The data presented is based on our previous benchmarking studies [7–9]

functioning at the molecular and cellular levels (reviewed in [3, 16, 19]). Molecular dynamics, as one of the most detailed biomolecular simulation techniques, has been particularly important in illuminating the interplay between microtubule structure and dynamics at the nanoscale. This method describes atoms as point masses interacting through a variety of interatomic potentials, collectively called a force field. The first structure of tubulin was obtained in the late 1990s [34]. Pioneering molecular dynamics simulations of tubulin were reported in 2004 by Mitra and Sept, aiming to explore the mechanisms of interaction between tubulin and microtubule targeting agents as potential chemotherapeutics [30]. In 2008, Gebremichael et al. were the first to use

molecular dynamics to address the question of the shape of individual tubulin subunits, providing crucial arguments about the similarity in curvature among the main building elements of microtubules, regardless of their bound nucleotide: GTP or GDP [12]. In the same year, Mitra and Sept developed the first model of six interacting tubulins within the microtubule lattice to gain insights into effects of the important chemotherapy drug paclitaxel on the microtubule [31]. Both studies simulated tubulin systems containing approximately 160000–180000 particles, including protein, solvent, and ion atoms, each covering no more than a few tens of nanoseconds, which was already a significant computational achievement at the time those pioneering simulations were carried out. During the 2010s, molecular dynamics simulations of tubulin already focused on examining the behavior of tubulin protofilaments and small fragments of the microtubule lattice, up to 3×6 monomers, rather than individual tubulin dimers [10, 13, 14, 20, 33, 38]. The total simulated time reached 1 μ s, typically with 3–5 repetitions to improve sampling. Remarkably, in 2010 the first model of a complete microtubule was constructed by Wells and Aksimentiev [46] to study mechanical response of microtubules to deformations. Given the impressive system size, the simulation timescales, which were achieved, were quite modest, only spanning a few nanoseconds. In 2014, the cryo-EM revolution enabled the first structures of whole microtubules to be resolved at sufficient resolution, allowing for more informed studies of tubulin interactions within the microtubule lattice under different nucleotide states [2, 27, 50, 51]. By the 2020s, advancements in computational power finally enabled the simulation of entire microtubules by several research teams. In 2020 a 16-layer-microtubule model was constructed and simulated by Tong and Voth for about 200 ns [41]. Two years later, a massive computational effort by Igaev and Grubmüller described molecular dynamics simulations of over 15 million particles for a combined time exceeding 5000 ns with each nucleotide [21]. This was sufficient to observe at least partial relaxation of the modeled microtubule tips into flared morphologies, consistent with recent cryo-EM observations. In very recent paper [47] the authors used large microtubule lattice systems comprising ~ 21 –38 million atoms, and applied their multiscale approach to leapfrog through time and nearly double the computational efficiency in realizing relaxed all-atom conformations of GDP- and GTP-complexed microtubule tips. At the time of writing this manuscript, at least one more research preprint is available on the BioRxiv server, exploring entire microtubule system with simulation times extending up to several microseconds [23]. This clear increase in the simulated timescales and the system sizes, illustrated in Fig. 7, reflects the combined progress in computational algorithms, hardware architecture, and simulation software in recent years, Fig. 6.

Despite the spectacular progress in recent years, it is clear that molecular dynamics simulations of tubulin have reached a point where further advancement becomes more challenging due to the inability to cover the timescales required to observe microtubule dynamics as seen in experiments. While the current molecular dynamics simulations of whole microtubule tips cover only a few microseconds, real microtubules lose layers of tubulin subunits during their rapid depolymerization only once per tens of milliseconds on average [5, 44]. Given the pace of research development, we are confident that this gap will eventually be closed, likely through increases in both the spatiotemporal resolution of experimental methods and the further enhancement of computational resources and algorithms. However, as is often the case with scientific progress, unforeseen breakthroughs or inventions may also help accelerate this development in yet unclear ways. Coarse-graining is one promising pathway, though it inevitably comes at the cost of some loss of information and precision [18, 23, 24, 28, 39, 40, 42, 49]. As a somewhat unconventional

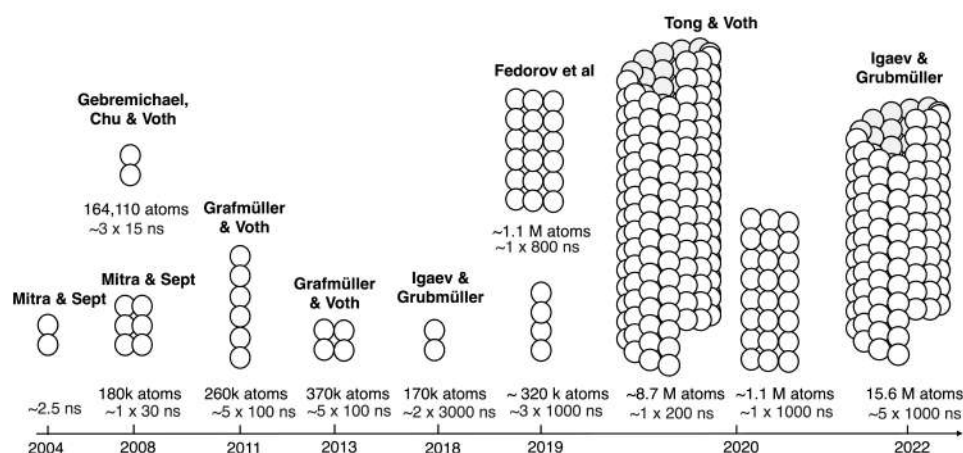


Figure 7. Timeline of the molecular dynamics studies of tubulins: from building blocks to whole microtubule

approach to facilitating this work, we have proposed here turning to bacterial microtubules as a more computationally feasible model. We demonstrate that bacterial microtubules share many features with their eukaryotic counterparts, including twist-bending relaxation of their individual building blocks. The computational model can be about an order of magnitude smaller compared to whole eukaryotic microtubule, significantly reducing computational cost and providing a good system for iterative exploration alongside experimental work, given the ease of genetic manipulation and expression of BtubAB mutant genes for in vitro characterization of the modified BtubAB polymers.

Acknowledgments

Molecular dynamics simulations and data processing of bacterial tubulins were supported by the Russian Science Foundation grant No. 24-74-00002 (<https://rscf.ru/project/24-74-00002/>). The simulations were carried out using the infrastructure of the shared research facilities of HPC computing resources at Lomonosov Moscow State University. N.G. acknowledges Scientific and Educational Mathematical Center “Sofia Kovalevskaya Northwestern Center for Mathematical Research” for financial support of the analysis of computational performance (agreement № 075-02-2025-1607, 27.02.2025).

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Abraham, M.J., Murtola, T., Schulz, R., *et al.*: GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX* 1, 19–25 (2015). <https://doi.org/10.1016/j.softx.2015.06.001>
2. Alushin, G.M., Lander, G.C., Kellogg, E.H., *et al.*: High-resolution microtubule structures

- hr/>
- reveal the structural transitions in $\alpha\beta$ -tubulin upon GTP hydrolysis. *Cell* 157(5), 1117–1129 (2014). <https://doi.org/10.1016/j.cell.2014.03.053>
3. Bowne-Anderson, H., Zanic, M., Kauer, M., Howard, J.: Microtubule dynamic instability: a new model with coupled GTP hydrolysis and multistep catastrophe. *BioEssays* 35(5), 452–461 (2013). <https://doi.org/10.1002/bies.201200131>
 4. Brouhard, G.J., Rice, L.M.: Microtubule dynamics: an interplay of biochemistry and mechanics. *Nature Reviews Molecular Cell Biology* 19(7), 451–463 (2018). <https://doi.org/10.1038/s41580-018-0009-y>
 5. Deng, X., Fink, G., Bharat, T.A., *et al.*: Four-stranded mini microtubules formed by *Prostheco bacter* BtubAB show dynamic instability. *Proceedings of the National Academy of Sciences* 114(29), E5950–E5958 (2017). <https://doi.org/10.1073/pnas.1705062114>
 6. Desai, A., Mitchison, T.J.: Microtubule polymerization dynamics. *Annual Review of Cell and Developmental Biology* 13(1), 83–117 (1997). <https://doi.org/10.1146/annurev.cellbio.13.1.83>
 7. Fedorov, V.A., Kholina, E.G., Gudimchuk, N.B., Kovalenko, I.B.: High-Performance Computing of Microtubule Protofilament Dynamics by Means of All-Atom Molecular Modeling. *Supercomputing Frontiers and Innovations* 10(4), 62–68 (2023). <https://doi.org/10.14529/jsfi230406>
 8. Fedorov, V.A., Kholina, E.G., Kovalenko, I.B., Gudimchuk, N.B.: Performance analysis of different computational architectures: Molecular dynamics in application to protein assemblies, illustrated by microtubule and electron transfer proteins. *Supercomputing Frontiers and Innovations* 5(4), 111–114 (2018). <https://doi.org/10.14529/jsfi180414>
 9. Fedorov, V.A., Kholina, E.G., Kovalenko, I.B., *et al.*: Update on performance analysis of different computational architectures: Molecular dynamics in application to protein-protein interactions. *Supercomputing Frontiers and Innovations* 7(4), 62–67 (2020). <https://doi.org/10.14529/jsfi200405>
 10. Fedorov, V.A., Orekhov, P.S., Kholina, E.G., *et al.*: Mechanical properties of tubulin intra- and inter-dimer interfaces and their implications for microtubule dynamic instability. *PLoS Computational Biology* 15(8), e1007327 (2019). <https://doi.org/10.1371/journal.pcbi.1007327>
 11. Feenstra, K.A., Hess, B., Berendsen, H.J.: Improving efficiency of large time-scale molecular dynamics simulations of hydrogen-rich systems. *Journal of Computational Chemistry* 20(8), 786–798 (1999). [https://doi.org/10.1002/\(SICI\)1096-987X\(199906\)20:8<786::AID-JCC5>3.0.CO;2-B](https://doi.org/10.1002/(SICI)1096-987X(199906)20:8<786::AID-JCC5>3.0.CO;2-B)
 12. Gebremichael, Y., Chu, J.W., Voth, G.A.: Intrinsic bending and structural rearrangement of tubulin dimer: molecular dynamics simulations and coarse-grained analysis. *Biophysical Journal* 95(5), 2487–2499 (2008). <https://doi.org/10.1529/biophysj.108.129072>
 13. Grafmüller, A., Noya, E.G., Voth, G.A.: Nucleotide-dependent lateral and longitudinal interactions in microtubules. *Journal of Molecular Biology* 425(12), 2232–2246 (2013). <https://doi.org/10.1016/j.jmb.2013.03.029>

14. Grafmüller, A., Voth, G.A.: Intrinsic bending of microtubule protofilaments. *Structure* 19(3), 409–417 (2011). <https://doi.org/10.1016/j.str.2010.12.020>
15. Gudimchuk, N.B., Alexandrova, V.V.: Measuring and modeling forces generated by microtubules. *Biophysical Reviews* 15(5), 1095–1110 (2023). <https://doi.org/10.1007/s12551-023-01161-7>
16. Gudimchuk, N.B., Alexandrova, V.V., Ulyanov, E.V., *et al.*: Modeling microtubule dynamics on Lomonosov-2 supercomputer of Moscow State University: from atomistic to cellular scale simulations. *Supercomputing Frontiers and Innovations* 11(3), 107–116 (2024). <https://doi.org/10.14529/jsfi240307>
17. Gudimchuk, N.B., McIntosh, J.R.: Regulation of microtubule dynamics, mechanics and function through the growing tip. *Nature Reviews Molecular Cell Biology* 22(12), 777–795 (2021). <https://doi.org/10.1038/s41580-021-00399-x>
18. Gudimchuk, N.B., Ulyanov, E.V., O’Toole, E., *et al.*: Mechanisms of microtubule dynamics and force generation examined with computational modeling and electron cryotomography. *Nature Communications* 11(1), 3765 (2020). <https://doi.org/10.1038/s41467-020-17553-2>
19. Hemmat, M., Castle, B.T., Odde, D.J.: Microtubule dynamics: moving toward a multi-scale approach. *Current Opinion in Cell Biology* 50, 8–13 (2018). <https://doi.org/10.1016/j.ceb.2017.12.013>
20. Igaev, M., Grubmüller, H.: Microtubule assembly governed by tubulin allosteric gain in flexibility and lattice induced fit. *eLife* 7, e34353 (2018). <https://doi.org/10.7554/eLife.34353>
21. Igaev, M., Grubmüller, H.: Bending-torsional elasticity and energetics of the plus-end microtubule tip. *Proceedings of the National Academy of Sciences* 119(12), e2115516119 (2022). <https://doi.org/10.1073/pnas.2115516119>
22. Jenkins, C., Samudrala, R., Anderson, I., *et al.*: Genes for the cytoskeletal protein tubulin in the bacterial genus *Prostheco bacter*. *Proceedings of the National Academy of Sciences* 99(26), 17049–17054 (2002). <https://doi.org/10.1073/pnas.012516899>
23. Kalutskii, M., Grubmüller, H., Volkov, V.A., Igaev, M.: Microtubule dynamics are defined by conformations and stability of clustered protofilaments. *bioRxiv* (2024). <https://doi.org/10.1101/2024.11.04.621893>
24. Kliuchnikov, E., Klyshko, E., Kelly, M.S., *et al.*: Microtubule assembly and disassembly dynamics model: Exploring dynamic instability and identifying features of microtubules growth, catastrophe, shortening, and rescue. *Computational and Structural Biotechnology Journal* 20, 953–974 (2022). <https://doi.org/10.1016/j.csbj.2022.01.028>
25. MacKerell, A.D.J., Bashford, D., Bellott, M., *et al.*: All-atom empirical potential for molecular modeling and dynamics studies of proteins. *The Journal of Physical Chemistry B* 102(18), 3586–3616 (1998). <https://doi.org/10.1021/jp973084f>

26. MacKerell Jr, A.D., Feig, M., Brooks, C.L.: Improved treatment of the protein backbone in empirical force fields. *Journal of the American Chemical Society* 126(3), 698–699 (2004). <https://doi.org/10.1021/ja036959e>
27. Manka, S.W., Moores, C.A.: The role of tubulin–tubulin lattice contacts in the mechanism of microtubule dynamic instability. *Nature Structural & Molecular Biology* 25(7), 607–615 (2018). <https://doi.org/10.1038/s41594-018-0087-8>
28. Michaels, T.C., Feng, S., Liang, H., Mahadevan, L.: Mechanics and kinetics of dynamic instability. *eLife* 9, e54077 (2020). <https://doi.org/10.7554/eLife.54077>
29. Mitchison, T., Kirschner, M.: Dynamic instability of microtubule growth. *Nature* 312(5991), 237–242 (1984). <https://doi.org/https://doi.org/10.1038/312237a0>
30. Mitra, A., Sept, D.: Localization of the antimitotic peptide and depsipeptide binding site on β -tubulin. *Biochemistry* 43(44), 13955–13962 (2004). <https://doi.org/10.1021/bi0487387>
31. Mitra, A., Sept, D.: Taxol allosterically alters the dynamics of the tubulin dimer and increases the flexibility of microtubules. *Biophysical Journal* 95(7), 3252–3258 (2008). <https://doi.org/10.1529/biophysj.108.133884>
32. Morozenko, A., Stuchebrukhov, A.: Dowser++, a new method of hydrating protein structures. *Proteins: Structure, Function, and Bioinformatics* 84(10), 1347–1357 (2016). <https://doi.org/10.1002/prot.25081>
33. Natarajan, K., Mohan, J., Senapati, S.: Relating nucleotide-dependent conformational changes in free tubulin dimers to tubulin assembly. *Biopolymers* 99(5), 282–291 (2013). <https://doi.org/10.1002/bip.22153>
34. Nogales, E., Wolf, S.G., Downing, K.H.: Erratum: Structure of the $\alpha\beta$ tubulin dimer by electron crystallography. *Nature* 393(6681), 191–191 (1998). <https://doi.org/10.1038/34465>
35. Olsson, M.H., Søndergaard, C.R., Rostkowski, M., Jensen, J.H.: PROPKA3: Consistent treatment of internal and surface residues in empirical pK_a predictions. *Journal of Chemical Theory and Computation* 7(2), 525–537 (2011). <https://doi.org/10.1021/ct100578z>
36. Parrinello, M., Rahman, A.: Polymorphic transitions in single crystals: A new molecular dynamics method. *Journal of Applied Physics* 52(12), 7182–7190 (1981). <https://doi.org/10.1063/1.328693>
37. Pavelites, J.J., Gao, J., Bash, P.A., Mackerell Jr, A.D.: A molecular mechanics force field for NAD^+ NADH, and the pyrophosphate groups of nucleotides. *Journal of Computational Chemistry* 18(2), 221–239 (1997). [https://doi.org/10.1002/\(SICI\)1096-987X\(19970130\)18:2<221::AID-JCC7>3.0.CO;2-X](https://doi.org/10.1002/(SICI)1096-987X(19970130)18:2<221::AID-JCC7>3.0.CO;2-X)
38. Peng, L.X., Hsu, M.T., Bonomi, M., *et al.*: The free energy profile of tubulin straight-bent conformational changes, with implications for microtubule assembly and drug discovery. *PLoS Computational Biology* 10(2), e1003464 (2014). <https://doi.org/10.1371/journal.pcbi.1003464>

39. Sahoo, A., Hanson, S.M.: Martini without the twist: Unveiling a mechanically correct microtubule through bottom-up coarse-graining in Martini 3. *bioRxiv* (2024). <https://doi.org/10.1101/2024.05.29.596440>
40. Stewman, S.F., Tsui, K.K., Ma, A.: Dynamic instability from non-equilibrium structural transitions on the energy landscape of microtubule. *Cell Systems* 11(6), 608–624 (2020). <https://doi.org/10.1016/j.cels.2020.09.008>
41. Tong, D., Voth, G.A.: Microtubule simulations provide insight into the molecular mechanism underlying dynamic instability. *Biophysical Journal* 118(12), 2938–2951 (2020). <https://doi.org/10.1016/j.bpj.2020.04.028>
42. Ulyanov, E.V., Vinogradov, D.S., McIntosh, J.R., Gudimchuk, N.B.: Brownian dynamics simulation of protofilament relaxation during rapid freezing. *PLoS ONE* 16(2), e0247022 (2021). <https://doi.org/10.1371/journal.pone.0247022>
43. Voevodin, V.V., Antonov, A.S., Nikitenko, D.A., *et al.*: Supercomputer Lomonosov-2: large scale, deep monitoring and fine analytics for the user community. *Supercomputing Frontiers and Innovations* 6(2), 4–11 (2019). <https://doi.org/10.14529/jsfi190201>
44. Walker, R.A., O'Brien, E.T., Pryer, N.K., *et al.*: Dynamic instability of individual microtubules analyzed by video light microscopy: rate constants and transition frequencies. *Journal of Cell Biology* 107(4), 1437–1448 (1988). <https://doi.org/10.1083/jcb.107.4.1437>
45. Webb, B., Sali, A.: Comparative protein structure modeling using MODELLER. *Current Protocols in Bioinformatics* 47(1), 5.6.1–5.6.32 (2014). <https://doi.org/10.1002/0471250953.bi0506s47>
46. Wells, D.B., Aksimentiev, A.: Mechanical properties of a complete microtubule revealed through molecular dynamics simulation. *Biophysical Journal* 99(2), 629–637 (2010). <https://doi.org/10.1016/j.bpj.2010.04.038>
47. Wu, J., Dasetty, S., Beckett, D., *et al.*: Data-driven equation-free dynamics applied to many-protein complexes: The microtubule tip relaxation. *Biophysical Journal* (2025). <https://doi.org/10.1016/j.bpj.2025.01.009>
48. Yutin, N., Koonin, E.V.: Archaeal origin of tubulin. *Biology Direct* 7(1), 10 (2012). <https://doi.org/10.1186/1745-6150-7-10>
49. Zakharov, P., Gudimchuk, N., Voevodin, V., *et al.*: Molecular and mechanical causes of microtubule catastrophe and aging. *Biophysical Journal* 109(12), 2574–2591 (2015). <https://doi.org/10.1016/j.bpj.2015.10.048>
50. Zhang, R., Alushin, G.M., Brown, A., Nogales, E.: Mechanistic origin of microtubule dynamic instability and its modulation by EB proteins. *Cell* 162(4), 849–859 (2015). <https://doi.org/10.1016/j.cell.2015.07.012>
51. Zhang, R., LaFrance, B., Nogales, E.: Separating the effects of nucleotide and eb binding on microtubule structure. *Proceedings of the National Academy of Sciences* 115(27), E6191–E6200 (2018). <https://doi.org/10.1073/pnas.1802637115>

GPU Implementation of Zippel Method for Feynman Integral Reconstruction

Alexander V. Smirnov¹ , Boris I. Rozhnov¹ , Vadim V. Voevodin¹ 

© The Authors 2025. This paper is published with open access at SuperFri.org

The Zippel algorithm performs a rational reconstruction of multivariate polynomials and aims specifically at the sparse case, where other approaches, such as iterative Newton and Thiele reconstructions, have a significantly higher complexity. It is applied in different fields of science, lately becoming an important step in Feynman integral reduction in elementary particle physics within the modular approach to reduction. For some cases with multiple variables the Zippel reconstruction might become a bottleneck for the whole evaluation so that different optimizations are required. In this paper, we describe how we ported the classical Zippel algorithm for polynomials together with its balanced version for rational functions to graphical processor units (GPUs), as well as carried out its performance evaluation on several types of GPUs. According to our information, this is the first publically available implementation of this algorithm on GPUs, and the results show speedup up to 14.5 times compared to CPU-based version.

Keywords: rational reconstruction, Zippel algorithm, Feynman integrals, GPU.

Introduction

The main motivation for this paper lies in the field of elementary particle physics, namely Feynman integral reduction [4], one of the key steps of evaluation of Feynman integrals. However, a Feynman integral reduction technically means solving a huge sparse linear system with coefficients being rational functions of multiple variables; hence, the scope of the paper and possible applications are much more broad. For example, the algorithm is used internally in computer algebra systems such as **Wolfram Mathematica**, **Maple**, **SageMath**.

The classical approach to Feynman integral reduction was to solve the system directly on a large enough server [1, 10–12, 14, 15, 17–19, 23–25], but with increasing complexity and the availability of supercomputer infrastructure, new methods were required. Therefore, an approach for a rational reconstruction of functions was proposed, which treats the unknown coefficients as black-box rational functions of multiple variables that need to be reconstructed afterwards [2, 5, 9, 13, 16, 20–22]. Within this “modular” approach the reduction is first performed multiple times with fixed values of variables in modular fields over large prime numbers (fitting into 2^{64} to utilize machine-size arithmetics), and then the functions are reconstructed. One should not confuse reconstruction and interpolation: with the values of a rational function fixed in multiple points, one surely has an unlimited number of rational functions with such properties, but the reconstruction methods aim to find the “true” function which is the most “simple” one satisfying such conditions and that *all* following probes (evaluations of the function at other points) should also match the guessed function.

The reconstruction method has a long history starting with Newton interpolation reconstructing a polynomial of one variable.

$$\begin{aligned} f_N(x) &= \text{Newton}_x[f(x), N] \\ &\equiv a_1 + (x - x_1) \left[a_2 + (x - x_2) \left[a_3 + (x - x_3) [a_4 + \dots] \right] \right]. \end{aligned} \quad (1)$$

¹Lomonosov Moscow State University, Moscow, Russian Federation

For multivariate polynomial reconstruction one can proceed variable by variable, i.e. when reconstructing from n variables to $n + 1$ variables one takes a needed number of reconstructed functions $f(x_1, x_2, \dots, x_n, x_{n+1,i})$ for different values of $x_{n+1,i}$ and runs the univariate Newton reconstruction.

For univariate rational functions there is the Thiele reconstruction

$$\begin{aligned} f_T(x) &= \text{Thiele}_x[f(x), T] \\ &\equiv b_0 + (x - x_1) \left[b_1 + (x - x_2) \left[b_2 + (x - x_3) [b_4 + \dots]^{-1} \right]^{-1} \right]^{-1}. \end{aligned} \quad (2)$$

The situation becomes more complicated with multivariate rational functions, since the recursive Thiele formula is a combination of continued fractions and is technically too complex to be evaluated in real examples. Therefore, for multivariate rational functions, other methods are used, one of those being the balanced Newton reconstruction proposed in [2] and the balanced Zippel reconstruction proposed in [26] for sparse multivariate functions.

The balanced Zippel reconstruction of rational functions is the main subject of this paper. It is based on the Zippel reconstruction of polynomials suggested initially in [28] and developed in [3, 8]. The Zippel method is an approach with the lowest complexity for the sparse polynomial reconstruction. It is already widely applied in programs for Feynman integral reduction using the modular approach followed by reconstruction. However, in some cases, the Zippel reconstruction itself might become a bottleneck for the whole approach due to its quadratic complexity by the number of terms in the skeleton polynomial (see next section for details) which might be measured in millions. Hence, any possible optimization of the Zippel algorithm is needed.

There is a number of public Zippel algorithm implementations, mostly in some open-source repositories. Such programs as **Kira** or **FIRE** use the Zippel algorithm as a step in the reduction of Feynman integrals.

The main contribution of this paper is an implementation of the Zippel algorithm on GPUs (Graphical Processing Units). According to our information, this is the first such implementation of this algorithm on GPUs. We currently publish a description and evaluation results, while the code is in a private repository related to Feynman integrals, but we intend to make it public as a part of the new **FIRE** version this year. We are also considering a possible separation of the modular GPU and reconstruction part providing it as a separate library.

The rest of the paper is organized as follows. Section 1 describes the original Zippel algorithm. Section 2 is devoted to our implementation of this algorithm on GPUs. Section 3 provides evaluation results and benchmarks.

1. Description of Zippel Method

The Zippel method is a reconstruction procedure from a polynomial with $k - 1$ variables to a polynomial with k variables. The traditional approach for dense polynomials (meaning that if one considers all possible monomials of a given degree most of the coefficients are non-zero) is to proceed with a Newton reconstruction formula. However, for sparse polynomials, this is quite inefficient because it requires a large number of probes for the reconstruction.

Let us remind the method taking a few formulas from our paper [26]. Let us suppose that we have already reconstructed a polynomial $f(x_1, \dots, x_{k-1}, c_0)$, where c_0 is some constant value of x_k . There might be more variables, but they should be fixed at this point. We aim to run the

Newton reconstruction for x_k , so we need similar reconstructed polynomials for other values of x_k .

Let us suppose we take another value c_i of x_k . We consider the existing polynomial $f(x_1, \dots, x_{k-1}, c_0, \dots)$ as a skeleton, take all its non-zero monomials and assume that the set of nonzero monomials will remain the same for the yet unknown $f(x_1, \dots, x_{k-1}, c_i, \dots)$ so that it can have the following form:

$$f(x_1, \dots, x_{k-1}, c_i, \dots) = a_1 \cdot x_1^{p_{1,1}} \dots x_{k-1}^{p_{k-1,1}} + \dots + a_t \cdot x_1^{p_{1,t}} \dots x_{k-1}^{p_{k-1,t}} \quad (3)$$

for some t , where a_i are unknown constant coefficients and p are exponents taken from the skeleton.

This is a linear system for a_i and hence knowing the values of $f(x_1, \dots, x_{k-1}, c_i, \dots)$ for t different sets of $\{x_1, \dots, x_{k-1}\}$ (sampling points or probes) we can solve the system. This, however, has a complexity $O(t^3)$, thus the Zippel algorithm for polynomials suggests a specific set of sampling points, i.e. $y_1, \dots, y_{k-1}, y_1^2, \dots, y_{k-1}^2, \dots, y_1^t, \dots, y_{k-1}^t$. In this case, the system turns into a Vandermonde system which can be solved with complexity $O(t^2)$. The Zippel method consists of solving this system leading to the knowledge of $f(x_1, \dots, x_{k-1}, c_i, \dots)$ for different i , followed by univariate Newton reconstruction in x_k to obtain $f(x_1, \dots, x_{k-1}, x_k, \dots)$.

For the tasks described one needs to reconstruct rational functions of multiple variables, but it is possible to adapt the Zippel approach, for example, as the balanced Zippel method described in [26].

It is also important for the proper reconstruction order to perform it with the use of modular arithmetic over large prime numbers. This prevents coefficient growth and decreases the number of needed sampling points, and the final reconstruction to rational numbers is performed as a final step. Thus, to proceed with the algorithms, one needs an efficient library working with modular polynomials of multiple variables. There are few such libraries, with FLINT [6, 7] being one of the most efficient.

2. Proposed GPU Implementation of the Zippel Method

Most nodes of modern supercomputers and clusters are now equipped with dedicated GPUs that enable parallel execution of mathematical operations on large datasets in a multithreaded mode. When implemented correctly, they can significantly reduce the computation time during the program execution.

Modern GPUs are capable of handling thousands of threads simultaneously, offering excellent horizontal scalability, and their architecture is optimized for the simultaneous execution of uniform operations – a critical feature for the scientific computing tasks performed by the FIRE program.

For the case of multiple variables the Zippel reconstruction step sometimes becomes a bottleneck for the whole reduction process, hence implementing it on a GPU seemed an important task.

2.1. Modular Integer Operations on GPUs

There are not many ready-to-use solutions for modular arithmetic on GPU. Only after implementing our solution we found a CUMODP (CUDA Modular Polynomial Library) library.

A further investigation showed that this library uses 32-bit modular integers so is not directly ready for our case, which requires 64-bit operations.

Thus, during the development of the GPU migration solution we tested several approaches:

- standard modular arithmetic operators provided by the `nvcc` compiler using the `uint128_t` data type;
- an algorithm leveraging properties of modular arithmetic with the `uint64_t` data type with special GPU intrinsics;
- a ported version from CUMODP to the 64-bit case;
- and GPU-porting basic functions from the FLINT library (which is used in the original FIRE application).

Let us describe those variants in detail and compare their efficiency on a multiplication modular some large prime n fitting into `uint64_t`.

Variant 1 is the most straightforward using the `uint128_t` type, here we completely rely on the compiler:

```
// return (a * b) mod n
__inline__ __device__
unsigned long long mul_mod(
    unsigned long long a,
    unsigned long long b,
    unsigned long long n)
{
    return (static_cast<__uint128_t>(a) * b) % n;
}
```

A better approach (number 2) is to avoid the `uint128_t` type stating in `uint64_t` and using special CUDA intrinsic, `__umul64hi(a, b)` which takes two 64-bit numbers and returns the high part of its multiplication as a result. This approach lets one avoid using the `uint128_t` type. The code can be implemented in the following way:

```
__inline__ __device__
unsigned long long mul_mod(
    unsigned long long a,
    unsigned long long b,
    unsigned long long n)
{
    // Calculate the most significant 64 bits
    // of the product of the two 64 unsigned bit integers
    unsigned long long hi = __umul64hi(a, b);
    unsigned long long lo = a * b;
    hi %= n;
    lo %= n;
    // pow = 2^64 mod n
    unsigned long long pow = (1ULL << 63) % n;
    pow = (pow << 1) % n;
    hi = (hi * pow) % n;

    return (hi + lo) % n;
}
```

Variant 3 is based on CUMODP and uses a fast modular multiplication algorithm based on floating-point arithmetic. It has several significant limitations, the main one being its dependence on floating-point precision – the mantissa must be large enough to hold the full range of values without rounding errors. In this example, it has been extended to support larger numbers by using quadruple-precision floating-point numbers. The code looks the following way:

```
__inline__ __device__
int64_t mul_mod1(int64_t a, int64_t b, int64_t n)
{
    long double ninv = 1.0L / (long double) n;
    int64_t q = (int64_t)((((long double)a) * ((long double)b)) * ninv);
    int64_t res = a * b - q * n;
    res += (res >> 63) & n;
    res -= n;
    res += (res >> 63) & n;

    return res;
}
```

The FLINT library is open-source, written in C, and optimized for 32/64-bit arithmetic, which allows its code to be ported to CUDA with minimal modifications. However, the implementation is large enough due to many functions that are required. The top-layer function for multiplication looks as follows:

```
__device__
mp_limb_t nmod_mul_d(mp_limb_t a, mp_limb_t b, nmod_t mod)
{
    b <= mod.norm;
    mp_limb_t res;
    do {
        mp_limb_t q0xx, q1xx, rxx, p_hixx, p_loxx;
        mp_limb_t nxx, ninvxx;
        unsigned int normxx;
        ninvxx = (mod).ninv;
        normxx = (mod).norm;
        nxx = (mod).n << normxx;
        umul_ppmm_d(p_hixx, p_loxx, (a), (b));
        umul_ppmm_d(q1xx, q0xx, ninvxx, p_hixx);
        add_ssaaaa_d(q1xx, q0xx, q1xx, q0xx, p_hixx, p_loxx);
        rxx = (p_loxx - (q1xx + 1) * nxx);
        if (rxx > q0xx) rxx += nxx;
        rxx = (rxx < nxx ? rxx : rxx - nxx) >> normxx;
        (res) = rxx;
    } while (0);

    return res;
}
```

We ran a benchmark for all variants and got the following results. The multiplication was performed modulo two numbers located in global memory. The tests were performed on Radeon RX 560X (163 GFlop/s). The test involved 1 core and 1 thread.

Table 1. Results of additional fuzzing tests
for modular multiplication (10 000 000 iterations)

Algorithm	Time (sec)
uint128_t	140.13
uint64_t	24.67
CUMODP*	9.96
FLINT	1.48

As we can see from Tab. 1, both native implementations are quite inefficient. The reason is that the GPUs themselves are 32-bit machines, so even the 64-bit operations are supported but are not natural for the cores implemented as consequent 32-bit operations. The CUMODP implementation also has problems due to a division in the code. The FLINT version is an obvious winner.

Thus, we decided to port the FLINT multiplication to GPUs using all its powerful features including the precalculated inversion letting one to avoid the division operations. The following functions were ported for modular addition, multiplication, exponentiation, inversion, and remainder operations for 128- and 192-bit composite integers: `nmod_mul`, `add_ssaaaa`, `umul_ppmm`, `NMOD_RED2`, `NMOD_RED3`, `n_gcdinv`, `nmod_pow`, `n_invmod`, `nmod_inv`. We are considering to later provide the ported code as a standalone library.

2.2. Zippel Algorithm Implementation Details

The next step was to benchmark the CPU implementation of the Zippel algorithm in order to determine the algorithm parts requiring optimization. This was performed both with profilers and by manual code analysis searching for parts with quadratic complexity by the number of terms t in the skeleton monomial.

The most resource-intensive parts of the algorithm are two functions: `ZippelMultiplePrime()`, which performs a Zippel reconstruction of several expressions of various degrees in the simple case, and `BalancedZippel()`, which prepares a balanced numerator and denominator for a subsequent call to `ZippelMultiplePrime()`.

The `ZippelMultiplePrime()` function computes values of expressions at specified points using Lagrange interpolation. The estimated theoretical complexity for a single thread is high; however, with efficient parallelization, the complexity can be reduced to $O((n^2 + nm)/k)$, where k is the number of threads. The main steps of the Zippel algorithm can be illustrated as follows:

- Polynomial values at specified points are computed using Horner's scheme.
- During reconstruction, a vector of powers is generated in parallel.
- This vector of powers is multiplied by the vector of polynomial values.
- The resulting vector is scalar-multiplied by the modular inverse of the polynomial value computed via Horner's method.
- The final result vector is written to memory.

Several optimizations were introduced during GPU porting.

First, we used aligned memory, which plays a crucial role in working with two-dimensional data arrays on GPUs. The data is arranged in memory such that the addresses are powers-of-two aligned, with the alignment degree depending on the GPU architecture. This approach minimizes memory access latency and increases bandwidth during parallel access to array elements.

Second, the power vector in this algorithm is temporary and can be replaced with an accumulated sum, which length does not exceed t . Since access to this accumulator is frequent, placing it in shared memory provides the best data access performance.

Third, the modular addition operation which normally requires a modulo reduction at each step can be replaced by a standard summation with overflow control (emulating a 192-bit integer). This allows deferring a costly modulo operation until the final data processing stage.

$$(\dots(((a_1 \cdot b_1) \bmod N + (a_2 \cdot b_2) \bmod N) \bmod N + \dots) \bmod N) \equiv ((a_1 \cdot b_1) + (a_2 \cdot b_2) + \dots) \bmod N$$

The schematic representation of summing a large number of 64-bit values followed by modulo reduction is as follows:

```

...
ulong tempAccumLow = 0, TempAccumMid = 0, TempAccumHi = 0;
...
for (...) {
    ...
    // Multiply two 64-bit integers, result being split
    // into high (p1) and low (p0) parts
    umul_ppmm(p1, p0, term, value);

    // Add p1 and p0 to the accumulator
    add_ssaaaa(tempAccumMid, tempAccumLow, tempAccumMid, tempAccumLow, p1, p0);

    // Overflow control
    if (tempAccumMid < p1) tempAccumHi++;
    ...
}
...
// Equivalent to: ((a << 128) | (b << 64) | c) % N
NMOD_RED3(res, TempAccumHi, tempAccumMid, tempAccumLow, N);
...

```

Replacing the temporary power vector in the `ZippelMultiplePrime()` function also significantly reduces the amount of memory consumed on the GPU, which is critical when processing data in a large number of parallel threads. For example, processing a polynomial of approximately 5 million terms would require an additional 0.3 GB of memory per thread to store intermediate results.

Another block ported to the GPU is the preparation of balanced coefficients. The balancing approach involves constructing expressions by simultaneously multiplying both the numerator and denominator by additional numerical factor. This allows the numerator and denominator to be independently reconstructed using the Zippel algorithm (for details see [26]).

To construct such expressions, it is necessary to raise base variable values to various powers and substitute them into the polynomial on the GPU.

For this purpose, a fast exponentiation algorithm is used. A two-dimensional array of coefficients is generated by multiplying a base value by a precomputed k -th power of the same value, where k is the width of the array. This approach allows for quickly obtaining a coefficient array with a step of k :

$$\begin{aligned}
x_1^n, x_1^{n-1}, \dots, x_1^{n-k} &= x_1^k \cdot x_1^{n-k}, x_1^{n-k-1}, \dots, x_1^{n-2k} \\
x_2^n, x_2^{n-1}, \dots, x_2^{n-k} &= x_2^k \cdot x_2^{n-k}, x_2^{n-k-1}, \dots, x_2^{n-2k} \\
&\vdots \\
x_m^n, x_m^{n-1}, \dots, x_m^{n-k} &= x_m^k \cdot x_m^{n-k}, x_m^{n-k-1}, \dots, x_m^{n-2k}
\end{aligned}$$

Each GPU thread (kernel) processes its own value independently of others. Second-level loops are parallelized inside the kernel using individual threads. Each thread computes a portion of the polynomial. The final polynomial values are then summed together.

GPUs offer flexible mechanisms for inter-thread communication within the same warp, which provides significantly faster data exchange compared to global or shared memory access. In this case, inter-thread communication is implemented using the built-in function `__shfl_down_sync()`, which allows threads within a warp to exchange data in a downward direction.

Below is an example of a `warpReduceSum()` function, which sums all values across threads within a warp without relying on global or shared memory:

```

__inline__ __device__
mp_limb_t warpReduceSum(ulong val, nmod_t flint_mod)
{
    for (int shift = warpSize/2; shift > 0; shift /= 2)
    {
        val = nmod_add_d(val,
                        __shfl_down_sync(warpSize - 1, val, shift),
                        flint_mod);
    }

    return val;
}

```

To conclude, the implemented algorithm enables offloading the most resource-intensive part – coefficient reconstruction – to the GPU, thereby providing a flexible horizontal scalability. It also means that this implementation can be easily adapted for running on multiple devices or cluster nodes since in real physical examples there are multiple coefficients which have to be reconstructed, so that they can be distributed among different GPUs and even supercomputer nodes. Additionally, this approach frees up CPU resources, which can then be used for other tasks. The resulting performance will be discussed in the next section.

3. Performance Evaluation and Analysis

After developing the proposed GPU solution, it was necessary to evaluate its performance and speedup in comparison with the original CPU version (which is available as a private version of FIRE but with possible pre-publication access by request).

3.1. Experimental Conditions

Experiments were carried out on three GPUs: NVIDIA V100, P100 and A100, and Intel Xeon Gold 6126 2.6 GHz (12 cores, 178 Gop/s peak for INT64 operations) was used for comparing with the original CPU version. The main characteristics of used GPUs are shown in Tab. 2. The main type of data used is int64, so the theoretical peak performance was calculated for this operation type. Experiments were conducted on the equipment of the Supercomputer Center of Lomonosov Moscow State University: “pascal” partition of the Lomonosov-2 supercomputer [27] was used for tests on P100; “volta1” Lomonosov-2 partition – for tests on V100 and Intel Xeon 6126; calculations on A100 were performed on a standalone server.

Table 2. Characteristics of GPUs used for performance evaluation

GPU name	P100	V100	A100
INT64 peak performance, Top/s	2.38	3.53	4.87
Number of CUDA cores	3584	5120	6912
Theoretical peak memory bandwidth, GB/s	720	900	1555
Memory size, GB	16	32	40

Four different input data sizes were considered, all related to the final reconstruction over the sixth variable (d , the space-time dimension) using the Zippel algorithm. The datasets correspond to the same physical problem and represent reductions to master integrals of varying level. As a result, they differ in the number of monomials in the skeleton polynomial. The selected examples include the following numbers of monomials: 125 thousand (“125k”), 300 thousand (“300k”), 750 thousand (“750k”), and 4.8 million (“4.8m”). The computational complexity (and therefore the calculation time) does not depend much on the structure of the datasets, only on their size, so there was no need to consider different dataset variants.

As it was mentioned earlier, the complexity of the Zippel algorithm grows quadratically with the number of monomials, which makes this dataset sufficiently representative. In particular, the largest example was barely within the time constraints when computed on a CPU on the Lomonosov-2 supercomputer.

Each test was repeated 10 times in order to collect enough statistics on statistical significance between execution time of different experiments (the only exception is that for “4.8m” size the number of launches on CPU was reduced to 5, as they require a lot of time to run). The differences in execution times between identical runs were minimal and there were no outliers, so only average values are reported below.

3.2. Evaluations Results

Table 3 shows the results of execution time comparison on four platforms and for four input data sizes as described above. The top row shows the execution time in seconds for CPU version on Intel Xeon, while the rows below show the speedup on GPUs compared to Intel Xeon result.

We can see that execution time on Intel CPU varies from less than 1 minute up to 20+ hours, showing a wide duration spectrum. On “125k” input size, GPU speedup is up to 4, and it starts to increase as the input size gets bigger, leading up to 14.5x speedup on A100 on “4.8m” input size. It is interesting to note that on “125k” input size the speedup for V100 is shown to be bigger than the speedup for A100, although the latter GPU is more modern and powerful, but

Table 3. Comparison of execution time for different platforms

Platform	Input data size			
	125k	300k	750k	4.8m
Xeon 6126, time	44 s	255 s	1550 s	74431 s
P100, speedup	2.69	3.23	3.55	3.84
V100, speedup	3.87	5.36	7.69	8.85
A100, speedup	3.81	7.46	10.76	14.57

this difference is actually statistically insignificant (confidence intervals are overlapping). The difference between all other results is statistically significant.

In Tab. 4, the same results for GPUs are presented, but the speedup on the V100 and A100 platforms relative to the P100 is shown, which allows for an easier comparison of execution time difference between three GPUs. We can see that on “125k” input size, the speedup is not so big – less than 1.5 times both for V100 and A100, but on “4.8m” input size it rises up to 2.3 for V100 and 3.8 for A100 GPU. Thus, for small task sizes, the usage of more powerful graphics accelerators does not provide a significant benefit, but as the size increases, the advantage of more powerful GPUs becomes more and more noticeable.

Table 4. Comparison of execution time between different GPUs

Platform	Input data size			
	125k	300k	750k	4.8m
Speedup of V100 compared to P100	1.44	1.66	2.17	2.31
Speedup of A100 compared to P100	1.42	2.31	3.03	3.80

A few words about the efficiency of the proposed implementation should be said. For its rough estimation, the GPU load metric was used, which shows the percent of time during which one or more kernels was active (executing on the GPU). According to preliminary information (obtained not for all types of experiments), the efficiency grows with the increase of the task size, and for the case of “4.8m” it reaches values above 90% on all three types of GPU. For the size “700k” this value is on average not lower than 80%. A more detailed analysis is planned in the future, but it can already be concluded that the efficiency of the obtained implementation is high. It is also worth noting that the utilization within the CPU version is also high – a commonly used CPU load metric (percentage of time spent in user program) shows on average a value above 95%.

Conclusion

In this paper, a GPU implementation of the Zippel method based on porting and adapting FLINT functions is proposed. According to our information, this is the first implementation of this algorithm on GPUs. It provides a significant speedup when compared to a CPU variant. We intend to make the code public this year as part of the new FIRE version and expect this implementation to be used for different reduction tasks in elementary particle physics, both as part of the FIRE package and without it, since the reconstruction utilities can be used stand-alone. It is in our plans to speed up the reconstruction even more with the use of texture memory.

Acknowledgements

The work was supported by the Ministry of Education and Science of the Russian Federation as part of the program of the Moscow Center for Fundamental and Applied Mathematics under Agreement No. 075-15-2025-345. The research was carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University [27].

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.






References

1. Anastasiou, C., Lazopoulos, A.: Automatic integral reduction for higher order perturbative calculations. *Journal of High Energy Physics* 07, 046 (2004). <https://doi.org/10.1088/1126-6708/2004/07/046>
2. Belitsky, A.V., Smirnov, A.V., Yakovlev, R.V.: Balancing act: Multivariate rational reconstruction for IBP. *Nucl. Phys. B* 993, 116253 (2023). <https://doi.org/10.1016/j.nuclphysb.2023.116253>
3. Ben-Or, M., Tiwari, P.: A deterministic algorithm for sparse multivariate polynomial interpolation. In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*. p. 301–309. STOC '88, Association for Computing Machinery, New York, NY, USA (1988). <https://doi.org/10.1145/62212.62241>
4. Chetyrkin, K.G., Tkachov, F.V.: Integration by Parts: The Algorithm to Calculate beta Functions in 4 Loops. *Nucl. Phys. B* 192, 159–204 (1981). [https://doi.org/10.1016/0550-3213\(81\)90199-1](https://doi.org/10.1016/0550-3213(81)90199-1)
5. De Laurentis, G., Page, B.: Ansätze for scattering amplitudes from p-adic numbers and algebraic geometry. *Journal of High Energy Physics* 12, 140 (2022). [https://doi.org/10.1007/JHEP12\(2022\)140](https://doi.org/10.1007/JHEP12(2022)140)
6. Hart, W.: Fast Library for Number Theory: An introduction. In: Fukuda, K., van der Hoeven, J., Joswig, M., Takayama, N. (eds.) *Mathematical Software – ICMS 2010. Lecture Notes in Computer Science*, vol. 6327, pp. 88–91. Springer (2010). https://doi.org/10.1007/978-3-642-15582-6_15
7. Hart, W., Johansson, F., Schultz, D., *et al.*: FLINT: Fast Library for Number Theory. <http://www.flintlib.org/>, version 2.9 (or your version), accessed: 2025-05-01
8. Kaltofen, E., Lee, W.s., Lobo, A.A.: Early termination in Ben-Or/Tiwari sparse interpolation and a hybrid of Zippel’s algorithm. In: *Proceedings of the 2000 International Symposium on Symbolic and Algebraic Computation*. p. 192–201. ISSAC '00, Association for Computing Machinery, New York, NY, USA (2000). <https://doi.org/10.1145/345542.345629>
9. Klappert, J., Lange, F.: Reconstructing rational functions with FireFly. *Comput. Phys. Commun.* 247, 106951 (2020). <https://doi.org/10.1016/j.cpc.2019.106951>

10. Klappert, J., Lange, F., Maierhöfer, P., *et al.*: Integral reduction with Kira 2.0 and finite field methods. *Comput. Phys. Commun.* 266, 108024 (2021). <https://doi.org/10.1016/j.cpc.2021.108024>
11. Lange, F., Usovitsch, J., Wu, Z.: Kira 3: integral reduction with efficient seeding and optimized equation selection (2025), <https://arxiv.org/abs/2505.20197>
12. Laporta, S.: High precision calculation of multiloop Feynman integrals by difference equations. *Int. J. Mod. Phys. A* 15, 5087–5159 (2000). <https://doi.org/10.1142/S0217751X00002159>
13. Laurentis, G., Maître, D.: Extracting analytical one-loop amplitudes from numerical evaluations. *Journal of High Energy Physics* 07, 123 (2019). [https://doi.org/10.1007/JHEP07\(2019\)123](https://doi.org/10.1007/JHEP07(2019)123)
14. Lee, R.N.: Presenting LiteRed: a tool for the Loop InTEgrals REDuction (12 2012)
15. Lee, R.N.: LiteRed 1.4: a powerful tool for reduction of multiloop integrals. *J. Phys. Conf. Ser.* 523, 012059 (2014). <https://doi.org/10.1088/1742-6596/523/1/012059>
16. Magerya, V.: Rational Tracer: a Tool for Faster Rational Function Reconstruction (11 2022)
17. Maierhöfer, P., Usovitsch, J.: Kira 1.2 Release Notes (12 2018)
18. Maierhöfer, P., Usovitsch, J., Uwer, P.: Kira—A Feynman integral reduction program. *Comput. Phys. Commun.* 230, 99–112 (2018). <https://doi.org/10.1016/j.cpc.2018.04.012>
19. von Manteuffel, A., Studerus, C.: Reduze 2 – Distributed Feynman Integral Reduction (2012), <https://arxiv.org/abs/1201.4330>
20. von Manteuffel, A., Schabinger, R.M.: A novel approach to integration by parts reduction. *Phys. Lett. B* 744, 101–104 (2015). <https://doi.org/10.1016/j.physletb.2015.03.029>
21. Peraro, T.: Scattering amplitudes over finite fields and multivariate functional reconstruction. *Journal of High Energy Physics* 12, 030 (2016). [https://doi.org/10.1007/JHEP12\(2016\)030](https://doi.org/10.1007/JHEP12(2016)030)
22. Peraro, T.: FiniteFlow: multivariate functional reconstruction using finite fields and dataflow graphs. *Journal of High Energy Physics* 07, 031 (2019). [https://doi.org/10.1007/JHEP07\(2019\)031](https://doi.org/10.1007/JHEP07(2019)031)
23. Smirnov, A.V., Chuharev, F.S.: FIRE6: Feynman Integral REDuction with Modular Arithmetic. *Comput. Phys. Commun.* 247, 106877 (2020). <https://doi.org/10.1016/j.cpc.2019.106877>
24. Smirnov, A.V., Smirnov, V.A.: FIRE4, LiteRed and accompanying tools to solve integration by parts relations. *Comput. Phys. Commun.* 184, 2820–2827 (2013). <https://doi.org/10.1016/j.cpc.2013.06.016>
25. Smirnov, A.V.: FIRE5: a C++ implementation of Feynman Integral REDuction. *Comput. Phys. Commun.* 189, 182–191 (2015). <https://doi.org/10.1016/j.cpc.2014.11.024>

26. Smirnov, A.V., Zeng, M.: Feynman integral reduction: balanced reconstruction of sparse rational functions and implementation on supercomputers in a co-design approach. *Numerical Methods and Programming* 25(Special issue), 30–45 (2024). <https://doi.org/10.26089/NumMet.2024s03>
27. Voevodin, V., Antonov, A., Nikitenko, D., *et al.*: Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community. *Supercomputing Frontiers and Innovations* 6(2), 4–11 (2019). <https://doi.org/10.14529/jsfi190201>
28. Zippel, R.: Probabilistic algorithms for sparse polynomials. In: Ng, E.W. (ed.) *Symbolic and Algebraic Computation*. pp. 216–226. Springer Berlin Heidelberg, Berlin, Heidelberg (1979)

Mastering 3D-detection of Extensive Air Showers in Cherenkov Light

*Elena A. Bonvech*¹ , *Olga V. Cherkesova*^{1,2}, *Dmitriy V. Chernov*¹ ,
*Elena L. Entina*¹, *Vladimir I. Galkin*^{1,3} , *Vladimir A. Ivanov*^{1,3},
Timofey A. Kolodkin^{1,3}, *Natalia O. Ovcharenko*^{1,3},
Dmitriy A. Podgrudkov^{1,3} , *Tatiana M. Roganova*¹ , *Maxim D. Ziva*^{1,4}

© The Authors 2025. This paper is published with open access at SuperFri.org

A new SPHERE series complex extensive air shower detector is under development. The main goal of its mission is to study the mass composition of cosmic ray nuclei in the 1–100 PeV energy range at a new level. The helium-filled balloon will be substituted by an unmanned aerial vehicle as a carrier. This change opens the upper hemisphere for observations. The already well-established telescope of Cherenkov light reflected from the snow-covered ice surface of Lake Baikal from an altitude of 500–1000 m will be supported by a detector of direct light pointed upward. Since the two detectors will study the same shower at different stages of its development, it could be called a 3D-detection, which is completely new for the EAS method. The development is based on an extensive MC modeling of the shower and the detection process using the Supercomputer Complex of the Lomonosov Moscow State University.

Keywords: Cherenkov light, primary cosmic rays, Lomonosov-2 supercomputer, extensive air showers, air-borne telescope.

Introduction

The history of cosmic ray (CR) study in general and the study of high-energy CR through the registration of extensive air showers (EAS), in particular, involves the continuous development of new detectors and approaches to experiment design. Since the first observations by Pierre Auger using simple Geiger counters and a coincidence scheme [3], experiments have become more complex. The size of the detector arrays has increased, with the addition of new types of detectors dedicated to muon detection, leading to the development of the hybrid EAS detection technique. This approach has allowed more precise measurements of the CR energy spectrum and the first assessments of the mass composition of CR at high energies.

In addition to the registration of the charged particles of the EAS, the detection of optical components was proposed by A.E. Chudakov (Cherenkov light [13] and fluorescent light [8]). He pioneered the use of Cherenkov light registration to find local gamma-ray sources [16], thus founding what would become gamma-ray astronomy.

The very first gamma-ray telescope (a set of four) had a single photoelectron multiplier and only registered an excess in the EAS count rate. However, the technique has evolved into modern projects [1, 12, 14] that analyze the properties of Cherenkov light images in telescope cameras [11], e.g., the angular distribution of Cherenkov photons.

In 1974, A.E. Chudakov proposed a new method of EAS registration: detection of Cherenkov light reflected from the snow surface by an airborne detector [7]. The first successful realization of this method – SPHERE-1 and -2 [2] – registered the lateral distribution of Cherenkov photons

¹Skobeltsyn Institute for Nuclear Physics, Lomonosov Moscow State University, Moscow, Russian Federation

²Space Research Department, Lomonosov Moscow State University, Moscow, Russian Federation

³Physics Department, Lomonosov Moscow State University, Moscow, Russian Federation

⁴Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, Moscow, Russian Federation

with high accuracy. This distribution is also sensitive to the properties of the EAS primary particle [6].

However, most of the above-mentioned registration and data analysis methods were based on analytical estimates or limited simulations. Later simulations became more complex but were still limited in power. Nowadays, with advances in computing power, data analysis can become more complex and more interrelated approaches to the experiment can be explored.

In the developing SPHERE-3 experiment, we take a new approach to hybrid EAS registration: both spatial and angular Cherenkov light distributions, simultaneously at ground level and at altitude, a 3D view of the shower. This task requires a wide use of available tools: parallel computing for large simulations and analysis of large data sets [5], neural networks for both data analysis [4] and detector optimization [9].

The paper incorporates four sections and a conclusion. Section 1 describes the specific features of the future experiment derived from its main physical problem. Section 2 portrays different specialized procedures for the primary parameter assessment to be included in the general procedure later. Section 3 deals with a procedure for the event classification by the primary mass based on the data of both reflected and direct Cherenkov light. Section 4 gives a draft of a general self-consistent procedure for all primary parameter estimation. Conclusion summarizes our present achievements and states the importance of direct light data, thus setting our course for the 3D detection of EAS.

1. Experiment Features Resulting from its Main Goal

In general, the SPHERE-3 detector inherits its main features and scientific goals from its predecessor, the SPHERE-2 detector. However, we are shifting the focus of our research towards the primary cosmic ray composition, which will be the primary goal of the new experiment. The new 3D approach can help us greatly in this task.

The study of the mass composition of ultrahigh-energy primary cosmic rays is a challenging problem to be addressed in an EAS registration experiment. In order to obtain data on the mass of the primary nucleus, we need to find a measurable quantity that depends on the mass but is virtually independent of hadron interaction models, which remain a major obstacle to solving the cosmic ray mass composition problem.

Our experience with EAS Cherenkov light detection allows us to formulate some general requirements for such quantities. First, the quantity should be related to the shape of the observable distribution rather than to the absolute value. Second, the quantity must be a combination of integrals over parts of the measured image, or alternatively a parameter of an approximation to the image, in order to be less sensitive to image fluctuations. These two requirements are met by any measurable EAS Cherenkov light (CL) characteristic. As a result, the mass sensitive parameter based on the EAS CL is only slightly dependent on the interaction model.

Considering the EAS simulation results, we decided to use the image from the reflected light telescope to estimate E_0 , the axis direction, the core location on the snow and the primary mass (see Fig. 1). The image from the direct light detector is used for a more accurate evaluation of the axis direction and the primary mass [10]. The size of the long dimension of the light spot is a tentative mass parameter, but a better one is being sought. However, the sensitivity of the spot length to mass is greater than that of the image steepness in the telescope, forcing us to modify the measurement strategy to maximize the fraction of events with images in both detectors. At

an altitude of 500 m, the fraction is about 30% and decreases with altitude, making 500 m the preferred altitude for SPHERE-3 flights.

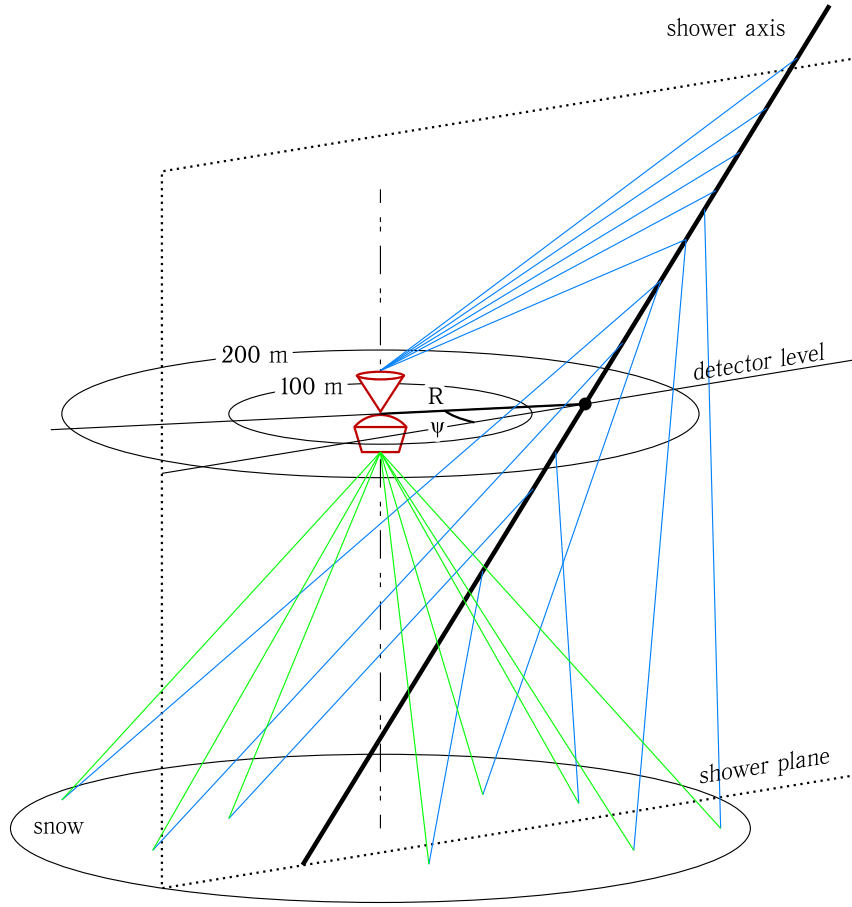


Figure 1. EAS detection at two levels. Reflected light (green) is collected by the lower detector of the SPHERE-3, direct light (blue) is accepted by the upper one

2. Building Blocks of the Experimental Data Handling Procedure

The entire shower parameters evaluation process involves a series of specialized algorithms that estimate a particular parameter based on data from a single detector. Then these algorithms are grouped according to their data source.

2.1. Based on the Reflected Light Telescope Images

2.1.1. Axis position on the snow and axis direction evaluation

The arrival direction of the primary particle is determined from the temporal characteristics of the EAS CL registered in pixels of the telescope mosaic. A Cherenkov pulse is identified in each pixel of the mosaic, and its maximum is located. The corresponding time is related to the time delay of the shower at the corresponding location on the snow, taking into account the

position of the pixel in the mosaic and the altitude. The time taken for the photon to travel from the observed snow spot to the telescope is also taken into account.

This process results in a set of time delays at different points on the snow surface forming a EAS CL temporal front. This front is approximated by a quadratic polynomial in the EAS coordinate system, creating a paraboloid of rotation around the shower axis. This method allows us to estimate the primary particle arrival direction with an accuracy of about 1–2°.

The shower axis location on the snow is determined by the spatial distribution of the CL photons on the telescope mosaic. For each pixel, the integral of the Cherenkov pulse is estimated. The pixel with the highest integral is then identified. If this maximum is close to a true maximum of the EAS CL distribution (and not caused by fluctuations, as discussed in the next paragraph), the shower axis is defined as the center of mass of two rows of pixels around the maximum pixel. If the maximum pixel is close to the edge of the mosaic, the center of mass is calculated using only one row. If the maximum pixel is at the edge of the mosaic, the event should be ignored as the shower axis cannot be reliably determined as may be outside the field of view of the detector.

Once the shower axis has been determined, its coordinates are mapped onto the snow surface and expressed in the telescope's coordinate system (relative to the telescope's vertical axis).

2.1.2. Primary energy evaluation procedure

The energy evaluation is based on the integral Q of the approximation function, which depends on the distance R from the detector axis to the shower axis on the snow. The set of dependencies $Q(R, E_0)$ is constructed by approximating the artificial image data. The dependency $E_0(R, Q)$ is constructed as an inverse interpolation of $Q(R, E_0)$. The quantities R_{exp} and Q_{exp} are obtained from the approximation of an image to be processed. By substituting R_{exp} and Q_{exp} into the formula, we obtain a primary energy estimate E_0^{est} :

$$E_0^{\text{est}} = E_0(R_{\text{exp}}, Q_{\text{exp}}). \quad (1)$$

To accurately determine the spatial distribution function (SDF), it is important that the distribution axis is within the mosaic and aligned with the image axis. Otherwise, errors may occur in the determination of Q_{exp} and R_{exp} . However, it is not always possible to discard an unsuitable image. If the SDF axis is outside the mosaic, the shape of the image may increase towards the boundary in the direction of the SDF axis. This can cause the maximum in the image to be on the mosaic boundary and the approximation to show that the SDF axis is outside the field of view. However, due to fluctuations in the number of photons comparable to the number of photons in the segment, the maximum of the image can shift from its true position. Therefore, during approximation, this local maximum can be mistaken for the SDF axis (we will refer to such maximums and axes as false ones). This leads to an incorrect distribution shape and therefore the value of Q_{exp} and consequently the energy will be underestimated.

The method for determining false maxima is developed on the basis of model data. It is based on finding the boundary of the distribution of the value characterizing the image for true and false maxima. The boundary is chosen to reject as few images as possible with a correctly defined axis, and as many as possible with a false axis. To make the selection criterion universal, relative features that are weakly dependent on the primary parameters are considered. If the criterion is still dependent on the primary energy, its relation to a measurable quantity correlated with the energy is established and used to adjust the criterion.

Several quantities have been considered. At the moment, the quantity chosen for the method is:

$$L = \left(\frac{f_{\text{surf}}}{q_1} - \frac{f}{q_2} \right) \times 100, \quad (2)$$

where f_{surf} and f are the values of the function characterizing the accuracy of the plane approximation and the axially symmetric function, q_1 and q_2 are the number of degrees of freedom. The separation boundary for this value L is linearly dependent on f .

In case of a sample of 200 events, with 100 false and 100 true maxima, the average energy error is 28%. After applying the false maximum detection method, the average error is reduced to 17%. The method eliminates 73% of the false maxima with a 3% error in eliminating true maxima.

2.1.3. Primary mass estimation procedure

It is well known that the depth of the maximum of a EAS depends directly on the mass number A of the primary particle:

$$\Delta X_{\text{max}} \approx -D \ln A, \quad (3)$$

which we simplify to $\Delta X_{\text{max}} \approx -\ln A$.

This means that the pattern of Cherenkov light on the ground (and in the detector) will differ as a function of the mass A , all else being equal. For lighter nuclei (such as protons), we expect a wider pool of light with a less pronounced peak, whereas for heavier nuclei (such as iron) we expect a narrower pool with a more pronounced peak.

Using a lateral distribution function (LDF) $F(R)$, which approximates the shape of the image in the focal plane of the telescope, we can quantify these differences and determine the primary particle type. We consider three types of nuclei: protons (p), nitrogen (N) and iron (Fe).

Given the LDF F , we introduce a one-dimensional mass sensitive criterion:

$$C = \frac{\int_0^{r_1} F \, dR}{\int_{r_1}^{r_2} F \, dR}, \quad (4)$$

where r_1 and r_2 are the radii of two concentric circles centered on the intensity peak with $r_2 \geq r_1$.

By choosing appropriate values for r_1 and r_2 for each event, we can formally measure the width and height of the intensity distribution of the shower image on the camera. In our current implementation, we scan through the ranges $r_1 \in [10, 110]$ and $r_2 \in [110, 270]$.

The optimal (r_1, r_2) pair is determined by minimizing the type II error in two binary classifications: p vs. N and N vs. Fe. Once these radii have been determined, we construct the distribution of C -values for simulated events and use it to define separation boundaries in a training sample. These boundaries are then used to classify the primary particle in each measured EAS image.

2.2. Based on the Direct Light Detector Images

2.2.1. Primary direction estimation using the angular distribution of Cherenkov light

The primary particle direction is determined from the characteristics of the Cherenkov image captured on the sensor after the Cherenkov photons have passed through a 400 cm^2 lens with a focal length of 64 cm. Currently, the primary particle arrival direction is evaluated using a key point, which is the center of mass or the location of the maximum intensity of the light spot. The distribution of intensity maxima and centers of mass of the images for a given distance R from the shower axis to the detector and a given detector azimuth ψ is shown in Fig. 2.

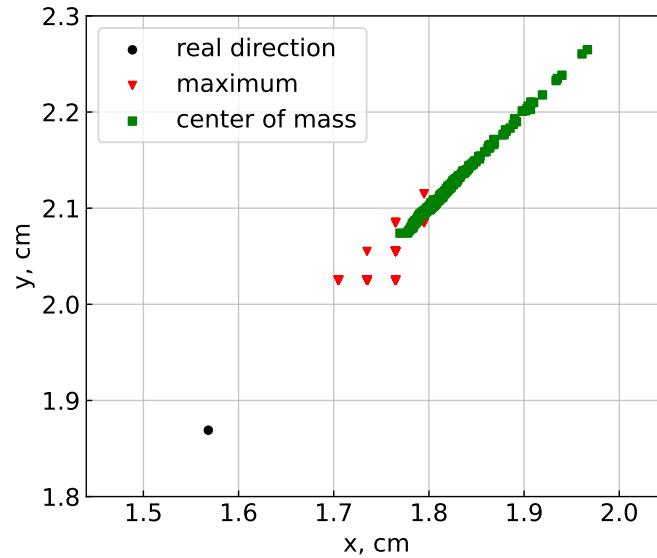


Figure 2. Distribution of maxima and centers of mass of the images for a distance $R = 100\text{ m}$ from the shower axis to the detector and a detector azimuth $\psi = 45^\circ$. Flight altitude 500 m above the snow level

In determining the primary particle arrival direction, a systematic shift occurs as a function of R . This shift can be estimated as the average difference between the actual direction and the key point direction. A comparison of the uncertainties in the primary particle arrival directions before and after eliminating the shift for different R is given in Tab. 1.

Table 1. Uncertainties of the primary particle direction estimates before and after the shift elimination

$R, \text{ m}$	By maximum		By center of mass	
	before	after	before	after
100 m	1.28	0.10	2.28	0.22
140 m	1.46	0.20	2.78	0.32

2.2.2. Primary mass evaluation by the angular distribution of Cherenkov light

One of the challenges of the SPHERE-3 project is to find an optimal criterion for classifying primary nuclei. Currently, they are divided into three groups based on the length of the image: proton, nitrogen and iron.

The classification procedure for the p–N pair implies that an event is caused by a proton if the spot length exceeds the critical value, and by a nitrogen nucleus if the opposite is true. For the N–Fe pair, an event with a spot length above the critical value is attributed to N, while an event with a shorter spot length is attributed to Fe.

For a fixed primary energy, the spot length depends on several parameters: the distance from the detector to the shower axis at flight level, the azimuthal angle of the detector with respect to the shower plane, and the absolute threshold. The latter is the number of photoelectrons per histogram bin, so that bins with contents above this threshold are taken into account. We have shown (see Tab. 2) that using a common criterion for all distances and azimuths can lead to misclassification errors exceeding 0.3. In Tab. 2, p is the probability of proton misclassification, p –N denotes the probability to take N for proton, N–Fe – probability to take N for Fe, Fe – probability of iron nuclei misclassification.

Table 2. Probabilities of misclassification of the primary particles using different criteria

Approach	p	p –N	N–Fe	Fe
common criterion	0.44	0.14	0.42	0.11
system of criteria	0.34	0.23	0.32	0.16
absolute threshold dependent on R	0.25	0.26	0.23	0.24
with double detection method	0.39	0.25	0.39	0.26

Therefore, it was decided to develop a system of criteria based on the critical Cherenkov image length, which depends on the distance R from the detector to the shower axis and the azimuth ψ of the detector. The system considers distances R in the range 100–200 m and azimuths ψ in the range 0–360°. The R -range is divided into five bins and the ψ -range into 24 bins. The number of R bins was chosen because of the uncertainty in R the flight level (see Fig. 3).

The size of the azimuthal bins is 15°, which allows the corresponding criterion to have approximately the same misclassification error as one with smaller bins.

The use of this system reduces the probability of misclassifying primary particles, as shown in Tab. 2.

Since the number of photons in the image at fixed primary particle energy E_0 depends strongly on the distance from the detector to the shower core R at the detector altitude, it is possible that using an absolute threshold that varies with R can reduce the chance of misclassifying the primary particle. We found that applying a higher threshold for $R < 150$ m and a lower threshold for $R > 150$ m actually reduced the classification errors to the levels shown in Tab. 2.

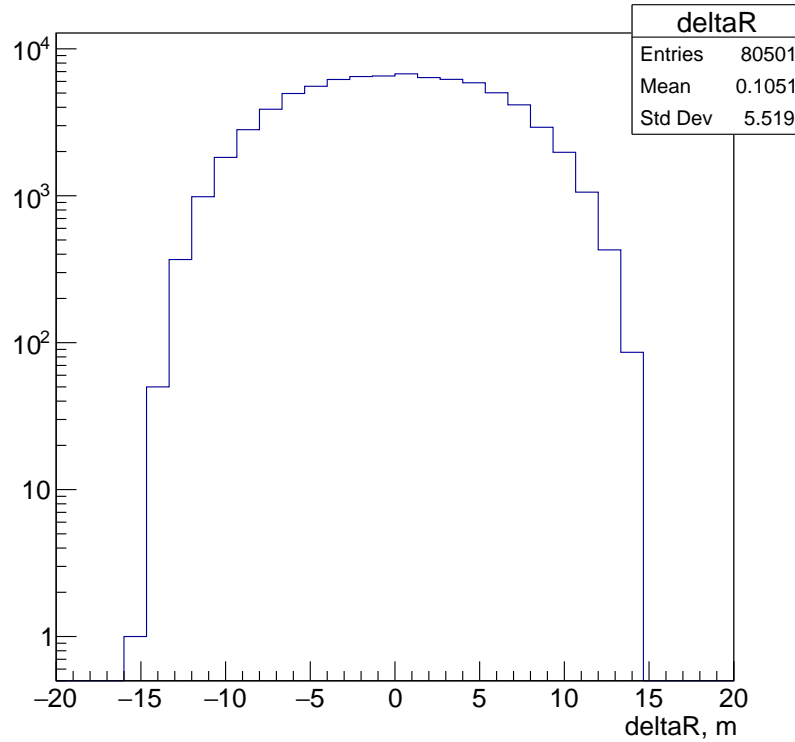


Figure 3. Distribution of the uncertainties of the distances R from the shower axis to the detector at the altitude of 500 m

3. Joint Criterion of Primary Mass

Now we come to the point where the power of the new development must be demonstrated. The ability of the reflected light telescope to distinguish between different nuclei is limited. The direct light detector can analyze the Cherenkov angular image of the EAS with better results for axis direction and mass classification. However, it is still not reasonable to ignore even a small assistance in such a sensitive matter as the primary mass study. Therefore, a joint criterion has been constructed based on the two criteria already described.

Training and testing of the criteria require the generation of appropriate samples that reproduce the situation of double detection. Specifically, a shower must be visible to both the reflected light telescope and the direct light detector. At an altitude of 500 m, this means that the shower axis must hit the snow at a maximum distance of 230 m from the detector axis. The shower axis at the altitude of 500 m must cross the ring (100 m, 200 m) around the detector. Close passes produce images that are too compact and difficult to classify. Distant showers produce faint images with fluctuating dimensions and shapes.

For each EAS event, a number of clones with different core locations on the snow are created within 230 m radius circle centered on the detector axis. If the axis of a clone intersects the ring at flight altitude, the clone is accepted and its images in both detectors are calculated and classified using the appropriate criteria. Otherwise, the clone is rejected.

For each clone, the values of both features are combined to form a vector. This creates a point in a two-dimensional feature space, which can then be used to search for the optimal linear decision function $ax + b$. The line that gives the lowest probability of misclassification is

considered optimal. The system also uses the criteria of the direct light detector with the absolute threshold depending on the distance R . This creates a system where the choice of linear decision function is determined by the R value at altitude and detector azimuth. An example of two decision functions of the system pertinent to a certain distance and azimuth ranges is presented in Fig. 4.

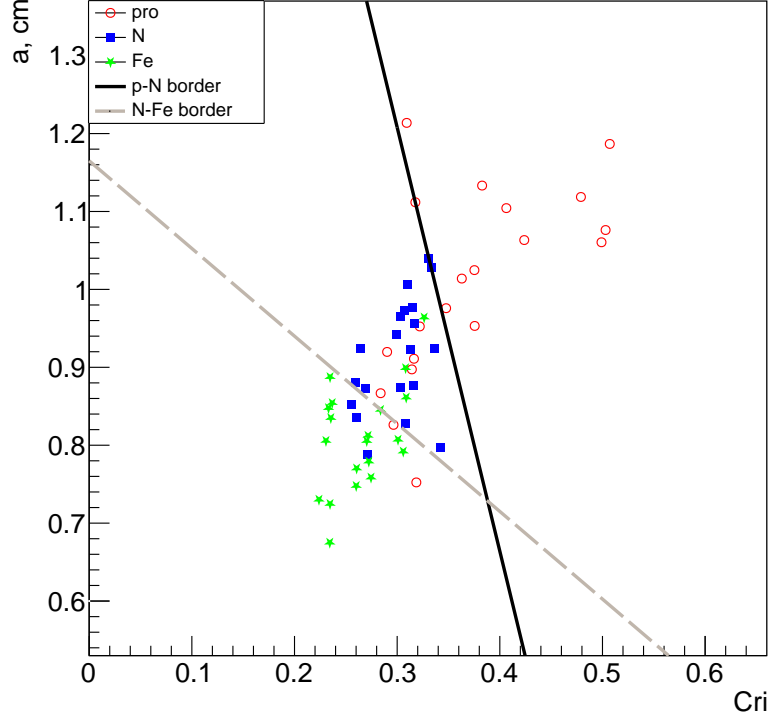


Figure 4. Straight lines separating the pairs p–N and N–Fe during dual detection. Here a is the feature made of the direct light image, Cri is the feature of the reflected light image. Detector altitude of 500 m above the snow level, distance range of 100–112.5 m, azimuthal range of 0° – 15°

The dual detection method currently produces the primary particle misclassification probabilities shown in Tab. 2. As can be seen from the values, the probability of misclassification is quite high. In order to reduce them, it is planned to modify the function to be minimized and to improve the gradient descent algorithm.

4. Self-consistent Procedure for EAS Primary Parameter Assessment

Images of an EAS in the reflected Cherenkov light telescope and the direct light detector contain a vast amount of data on the primary parameters of the shower, such as the energy E_0 , the arrival direction (θ, ϕ) and the mass A of the nucleus that initiated the shower. We have already developed specific methods to determine E_0 , (θ, ϕ) , and axis location separately using data from the telescope, without considering A . We have also developed a method to determine (θ, ϕ) from the direct light image, ignoring E_0 and A . For both the telescope and the detector,

we have developed procedures to estimate A using only a rough estimate of E_0 . Now we need to develop a comprehensive procedure that determines all the parameters consistently (see Fig. 5).

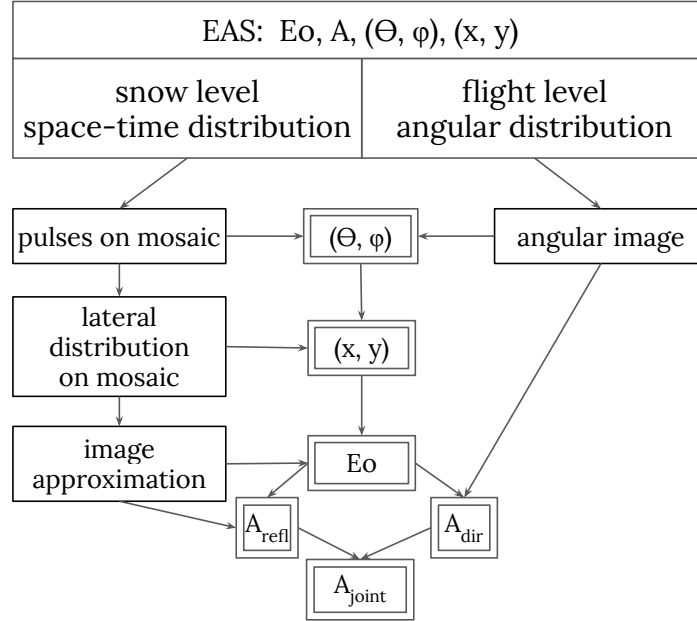


Figure 5. Scheme of the self-consistent procedure to estimate the primary parameters. Some parts of it will be looped. At the final stage one might require to iterate the whole procedure

Inevitably, one must start with estimates of E_0 , (θ, ϕ) and the axis location based on the image in the telescope, even if the image from the upper detector is available. First, the image of the reflected light must be corrected for the optical distortions in the telescope and approximated by an axially symmetric function. Typically, the image maximum corresponds to the shower core region on the snow, and so does the approximation maximum. Events with axes outside the field of view of the telescope may mimic the true maximum due to fluctuations. To eliminate these cases, certain measures are taken: 1) images with maxima at the edge of the mosaic are discarded, and 2) in addition to the approximation using aforementioned function, another approximation of the image is made using a plane, and the results of the two fits are compared, which helps to filter out false maxima. After confirming that the image maximum is real, the location of the shower core on the snow is calculated. A fit to the time delay of the light contributions to each pixel gives an estimate of the shower arrival direction with some accuracy.

A rough estimate of E_0 (ignoring the A estimate) is obtained by integrating the true approximation over a given circle. Finally, a ratio of two integrals of the approximation function is calculated on a central circle and an outer ring, which represents the steepness of the approximation and is sensitive to the primary particle mass. If the direct light image is not available, the final step in the process is an estimation loop: the direction and mass estimates are used to refine the energy estimate; the mass estimate is recalculated taking into account the new energy estimate, and so on. The axis position on the snow remains unchanged. If the direct light image is suitable for analysis, the axis of the shower is extrapolated to altitude. Its coordinates (R, ψ) are then estimated, which helps more accurately determine the shower axis direction. The shower axis position and direction are then adjusted in a recursive, self-consistent manner at the flight height.

The primary energy estimate is also important (see Section 2.2.1). Therefore, one must repeat the direction evaluation with a given energy estimate and the energy evaluation with a given direction estimate until both estimates converge. Until now, the energy estimate has not been affected by the mass estimate. To account for mass, a joint procedure for mass evaluation must be applied, based on mass sensitive parameters of both the reflected light telescope image and the direct light detector image.

The axis position is important for the primary mass estimation, as explained in the previous Section 2.2.2. The uncertainty in the axis position, ensured by the self-consistent recursive procedure, amounts to approximately 10 m for 500 m altitude, which is sufficient for the complex mass criterion system to divide the detected events into three groups by mass and presumably even for the construction of a mass regression with more sensitive parameters, which have yet to be found. The mass criterion to be used is chosen from a set of $5 \times 24 = 120$ different ones, depending on the position of the shower relative to the detector.

The next step is to combine the mass criterion with the reflected light criterion into a joint one, as described in Section 3. The mass estimate should now be used to correct the energy estimate. This correction may affect the direction and mass estimates, so the whole procedure must be repeated.

Perhaps we will eventually be able to combine all of the above criteria and relations into a single function. For now, the procedure described above seems to make some sense, especially since we have already fine-tuned some of its parts.

Conclusion

The design of the new SPHERE-3 complex detector is being optimized to meet the stated goal of studying the mass composition of primary cosmic rays in the 1–100 PeV energy range. This optimization is only possible through a large number of Monte Carlo simulations of the EAS and the processes inside the detector. We have already explored several detector designs and experimental strategies, and now present some preliminary conclusions on how to proceed.

A special feature of the SPHERE-3 detector is that the telescope for the registration of the reflected Cherenkov light will be supported by a detector of direct light pointed at the zenith and capable of analyzing the Cherenkov images in detail. The strategy behind this is that a certain fraction of the detected events will contribute images to both detectors, providing an unprecedented level of information about EAS at two different depths in the atmosphere.

We have shown the importance of the additional information provided by the direct light detector, and this has led us to choose an optical scheme and experimental design that will maximize the fraction of events seen by both detectors.

Acknowledgements

The research was carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University [15].

This work is supported by the Russian Science Foundation under Grant No. 23-72-00006, <https://rscf.ru/project/23-72-00006/>.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Acharya, B.S., Actis, M., Aghajani, T., *et al.*: Introducing the CTA concept. *Astroparticle Physics* 43, 3–18 (2013). <https://doi.org/10.1016/j.astropartphys.2013.01.007>
2. Antonov, R.A., Aulova, T.V., Bonvech, E.A., *et al.*: Detection of reflected Cherenkov light from extensive air showers in the SPHERE experiment as a method of studying superhigh energy cosmic rays. *Physics of Particles and Nuclei* 46(1), 60–93 (2015). <https://doi.org/10.1134/S1063779615010025>
3. Auger, P., Maze, R., Grivet-Meyer, T.: Grandes gerbes cosmiques atmosphériques contenant des corpuscules ultrapénétrants. *Comptes Rendus l'Académie des Sciences Paris* 206, 1721 (1938), <http://gallica.bnf.fr/ark:/12148/bpt6k3158p/f1721.image.langEN>
4. Bonvech, E.A., Azra, C.G., Cherkesova, O.V., *et al.*: Investigation of the capability of restoring information on the primary particle from Cherenkov light generated by extensive air showers using the Lomonosov-2 supercomputer. *Supercomputing Frontiers and Innovations* 11(3), 45–63 (Oct 2024). <https://doi.org/10.14529/jsfi240303>
5. Chernov, D.V., Azra, C.J., Bonvech, E.A., *et al.*: Approaches to optimization of experimental design for cosmic ray mass composition studies in the 1–1000 PeV energy range. *Physics of Atomic Nuclei* 87(2), S319–S338 (Dec 2024). <https://doi.org/10.1134/S1063778824700959>
6. Chernov, D., Podgrudkov, D., Antonov, R., *et al.*: Cosmic ray study by means of reflected EAS Cherenkov light method with the SPHERE-2 detector. *PoS ICRC2017*, 537 (2017). <https://doi.org/10.22323/1.301.0537>
7. Chudakov, A.E.: A possible method to detect EAS by the Cherenkov radiation reflected from the snowy ground surface. (in Russian). In: *Experimental methods of studying cosmic rays of superhigh energies: Proc. All-Union Symposium*. vol. 620, pp. 69–72. Siberian branch of Academy of Science of USSR (1972)
8. Chudakov, A.E., Dadykin, V.L., Nesterova, N.M., Zatsepin, V.I.: Search of high energy photons from sources of radionoise. In: *Proceedings of 5-th Inter-American Seminar on Cosmic Rays*. vol. 2, pp. 44–49 (1962)
9. Entina, E.L., Podgrudkov, D.A., Azra, C.G., *et al.*: Application of convolutional neural networks for extensive air shower separation in the SPHERE-3 experiment. *Moscow University Physics Bulletin* 79(2), S676–S683 (Dec 2024). <https://doi.org/10.3103/S0027134924702126>
10. Galkin, V.I., Azra, C.G., Bonvech, E.A., *et al.*: SPHERE-3: Tackling the problem of primary cosmic ray mass composition with a new approach. *Moscow University Physics Bulletin* 79(1), 384–391 (Dec 2024). <https://doi.org/10.3103/S002713492470111X>

11. Hillas, A.M.: Cerenkov light images of EAS produced by primary gamma rays and by nuclei. In: Jones, F.C. (ed.) 19th International Cosmic Ray Conference (ICRC19), Volume 3. International Cosmic Ray Conference, vol. 3, p. 445 (1985), <https://ui.adsabs.harvard.edu/abs/1985ICRC...3..445H/abstract>
12. Hofmann, W., HESS collaboration: The high energy stereoscopic system (HESS) project. AIP Conference Proceedings 515(1), 500–509 (2000). <https://doi.org/10.1063/1.1291416>
13. Nesterova, N.M., Chudakov, A.E.: On the observation of Cerenkov radiation accompanying broad atmospheric showers of cosmic rays. Journal of Experimental and Theoretical Physics 1(2), 388–389 (1954)
14. Perkins, J.S., Maier, G., Collaboration, T.V.: VERITAS telescope 1 relocation: Details and improvements. Proceedings of the 2009 Fermi Symposium, eConf Proceedings C091122 (2009). <https://doi.org/10.48550/arXiv.0912.3841>
15. Voevodin, V.V., Antonov, A.S., Nikitenko, D.A., *et al.*: Supercomputer Lomonosov-2: Large scale, deep monitoring and fine analytics for the user community. Supercomputing Frontiers and Innovations 6(2), 4–11 (Jun 2019). <https://doi.org/10.14529/jsfi190201>
16. Zatsepin, G.T., Chudakov, A.E.: Method of finding local sources if high-energy photons. Journal of Experimental and Theoretical Physics 14(2), 469–470 (1961)

Prospects for Improving Computational Efficiency of Hydrodynamic Simulations on Supercomputers by Increasing the Number of GPUs per Compute Node

*Sergei S. Khrapov*¹ , *Ekaterina O. Agafonnikova*¹ ,
*Alexander V. Khoperskov*¹ 

© The Authors 2025. This paper is published with open access at SuperFri.org

Hydrodynamic models for studying surface water dynamics on realistic topography place special demands on computational performance. Such simulations must cover large areas to ensure hydrological connectivity of the territory due to the influence of catchment areas. On the other hand, small topographic inhomogeneities on the scale of a meter are often the determining factors of fluid dynamics. Our analysis is based on a model of surface water and sediment dynamics for a large mountainous area of the Krasnodar region under rainfall/runoff conditions. The results of such large-scale models can be provided by parallel OpenMP-CUDA codes for computing systems with multi-GPU. We focus on different ways of transferring data between GPUs using both GPUDirect and HostCopy technologies on computing systems with one to eight GPUs. The parallel code with HostCopy is on average several times slower and less efficient compared to the GPUDirect approach. We propose to use auxiliary characteristics to analyze the efficiency of parallel implementation of a numerical algorithm. These values are calculated based on the average processing time of one computational cell and allow us to determine the optimal grid resolution in terms of performance.

Keywords: computational fluid dynamics, GPUs, efficiency, scalability, data transfer.

Introduction

Simulations of hydrological regimes for specific regions of the Earth often require outstanding computational resources, approaching the needs of climate modeling [2, 12, 21, 35]. This is due to at least three factors. First, the vastness of the simulated territory is determined by the length of the river system, the large area of drainage basins or hydrological landscape. Second, the need to use very fine grids is dictated by the presence of small-scale heterogeneities in topography and distributions of other physical characteristics that can significantly affect the results. The required spatial resolution in urban flooding conditions can be less than one meter, which implies hundreds of millions of cells in the digital elevation model (DEM) for specific urbanized areas [7]. Finally, the study of some hydrological processes requires the construction of long-term data series over a year or more ($T \sim 10^8$ sec), which already for two-dimensional models gives the number of integration time steps of the order of 10^9 in typical land surface hydrology problems. Flood forecast models are in high demand for assessing the consequences and identifying critical factors in the development of decision support systems and real-time emergency warning tools [2, 7, 34, 35]. Flood behavior is determined by the presence of physical factors that provide significant impacts at multiscale levels. Therefore, an interesting direction could be a 2D shallow water model using subgrid-scale topography for the 1D model as internal boundary conditions [33]. This provides additional savings in computing resources in multiscale problems.

The price-to-computing efficiency ratio dictates the need for a mass transfer of hydrodynamic simulation software to multi-GPU systems [9, 17, 21, 24, 37, 38]. This is facilitated by the presence of several reliable and convenient programming systems (CUDA, OpenCL and Ope-

¹Volgograd State University, Volgograd, Russian Federation

nACC). The organization of parallel simulations on several GPUs has numerous peculiarities and subtleties that require special analysis. Improving the efficiency of code performance on GPUs involves special optimization for specific hardware, which reduces the portability of the software. However, hardware-oriented algorithms can provide higher performance if we take into account both the GPU architecture from different manufacturers (NVIDIA, AMD, Intel) and the specific computational task [12, 26, 31]. The authors [9] highlight the difficulties of organizing work with the memory of several CPUs/GPUs as a critical problem of parallelization.

A significant problem of multi-GPU simulations is the deterioration of scalability due to the complex data transfer between the host and device, which depends on the features of data communication between processes [1]. For example, unstructured CFDs are provided with more sophisticated algorithms to optimize the performance of parallel computing compared to simpler structured Cartesian grids [37]. There are studies showing a strong decrease in scalability in systems with a large number of GPUs [38], which requires special studies for each specific numerical model [9]. Parallelization of implicit numerical schemes on multi-GPU requires significant efforts and special algorithms, since global memory access and global dependence of variables are required [3]. Different programming languages and compilers imply the presence of additional ways to optimize calculations and parallelize codes [24]. There is some progress in the transition to new high-performance interconnect architectures for supercomputers, which improves the capabilities of the MPI interface [29].

Specific software implementations of CFD in different approximations for GPUs demonstrate their effectiveness. The wetland flooding model using multiple GPUs on the Wetland DEM software yields a speedup of 2.39x on four GPUs compared to a single unit [21]. Unstructured CFD simulations in [37] show good performance on a single GPU (Nvidia Tesla V100), achieving an average speedup of 13.4x compared to 28 CPU cores (Intel Xeon Gold 6132). The MPI-OpenACC method for parallel simulation of Huizhou City Mountain Flood achieves a speedup of over 800 using 8 GPUs [7]. Free-surface fluid flow models with complex solid boundary surfaces based on the SPH method show great parallelism potential on GPUs with 10^9 particles [6]. An example is the C++ and CUDA-based DualSPHysics software, which is widely used for both research and engineering tasks [5].

The purpose of this work is to determine the impact of different parallel communication methods in a multi-GPU computing system on minimizing the overhead associated with data transfer. The article is organized as follows. Section 1 contains descriptions of the mathematical model and numerical algorithms. In Section 2, we discuss the features of setting up computational experiments, OpenMP-CUDA algorithm, implementing CSPH-TVD (Combined Smoothed Particle Hydrodynamics – Total Variation Diminishin) method for hydrological simulations. The following section shows the result of simulations of the consequences of storm rain for the Southern area of the Krasnodar region. Section 4 is devoted to the analysis of the efficiency of our parallel code on multi-GPU computing systems. Finally, the conclusion summarizes the study and outlines potential future research topics.

1. Mathematical Model and Numerical Implementation

The equations of surface water dynamics at high velocities must describe the self-consistent movement of water and sediment [10, 22, 23]. The standard Saint-Venant system of equations

includes changes in the topography of b with time also [15, 20, 27, 30]

$$\frac{\partial H}{\partial t} + \nabla_{\perp} (H \mathbf{u}) = q, \quad (1)$$

$$\frac{\partial(H\mathbf{u})}{\partial t} + \nabla_{\perp} (H\mathbf{u} \otimes \mathbf{u}) = -gH\nabla_{\perp}(b+H) + H\mathbf{f}, \quad (2)$$

where $H(x, y, t)$ is the water depth, $\mathbf{u} = \{u(x, y, t), v(x, y, t)\}$ is the average velocity vector of surface water flow, $\nabla_{\perp} = \{\partial/\partial x, \partial/\partial y\}$, (x, y) , $g = 9.81 \text{ m/s}^2$, $b(x, y, t)$ is the digital elevation model (DEM), which defines the topography of the computational domain, \mathbf{f} is the specific force of hydraulic resistance to the flow of water. The source function $q(x, y, t)$ determines the rate of inflow ($q > 0$) or outflow ($q < 0$) of liquid in the surface layer of water.

The temporal variability of the DEM is a result of sediment transport. A simple but effective model of sediment transport is the Exner equation [4, 10, 16, 22]

$$(1 - \psi) \frac{\partial b}{\partial t} + \nabla_{\perp} \mathbf{J}_b = c_J \nabla_{\perp} (|\mathbf{J}_b| \nabla_{\perp} b), \quad (3)$$

where \mathbf{J}_b is the horizontal sediment flux, ψ is the soil porosity, c_J is an empirical value depending on the type and condition of the soil ($c_J \simeq 1\text{--}10$ [SI system]), as an analogue of the diffusion coefficient [11]. The value of \mathbf{J}_b on a flat bottom is determined by the Grass formula [20, 22, 27]:

$$\mathbf{J}_b = \begin{cases} a_J \mathbf{u} |\mathbf{u}|^{m_J}, & |\mathbf{u}| > u^{(cr)} \\ 0, & |\mathbf{u}| \leq u^{(cr)} \end{cases}, \quad (4)$$

where $u^{(cr)}$ is the critical velocity determined by the Shamov formula [28], a_J and m_J ($-1 \leq m_J \leq 3$) are the adjustable parameters [10, 20, 27].

The choice of the hydraulic resistance model is not trivial, since it relies on subgrid physics as well as sediment transport modeling. The traditional description of hydraulic resistance only through the Manning roughness coefficient n_M [7, 10, 14, 33] requires its complex calibration. The additional influence of internal friction due to turbulence allows for a better fit of simulations with measured data [12, 15, 18], and we follow this approach of taking these factors into account here. The sum of b and H defines the water surface level $\eta(x, y, t) = b + H$, the measurements of which at gauging stations provide validation of numerical models by fitting measured and model time series $\eta(t)$. Thus, the value of \mathbf{f} consists of two components and includes resistance due to roughness $\mathbf{f}^{(M)}$ and turbulence $\mathbf{f}^{(turb)}$ [12, 18]:

$$\mathbf{f} = \mathbf{f}^{(M)} + \mathbf{f}^{(turb)} = g \frac{n_M^2 |\mathbf{u}|}{H^{4/3}} \mathbf{u} + \hat{\alpha} H^{\gamma} |\mathbf{u}|^{1-\gamma} \mathbf{u}, \quad (5)$$

where n_M , $\hat{\alpha}$, γ are the free parameters.

We use the Lagrangian-Eulerian numerical scheme Combined Smoothed Particle Hydrodynamics – Total Variation Diminishin (CSPH-TVD) [14, 16], which combines the strengths of both approaches at different steps of integration of the system of equations (1)–(3). A purely Lagrangian method SPH is complemented by the advantages of the grid algorithm for the exact calculation of the fluxes of physical quantities. Smoothed particles move within their cells under the action of hydrodynamic and external forces in the first step of the algorithm. Then, the mass and momentum fluxes are calculated through the boundaries of the cells of a fixed grid at the intermediate time $t_{n+1/2} = t_n + \Delta t_n/2$ using the modified TVD approach and an approximate

solution of the Riemann problem. The third stage involves the return of SPH particles to the cell centers, which gives the state of the system at time $t_{n+1} = t_n + \Delta t_n$.

The function $q(x, y, t)$ in (1) specifies the precipitation rate in the model. The rain front moves with the velocity $\mathbf{u}^{(r)}$ from left to right in the base model. We roughly reproduce the disaster in 2012, when a rain storm led to mountain torrents and severe flooding of Krymsk City and other settlements in the Krasnodar region [19]. We analyze different variants of rain characteristics. The base model is the velocity $\mathbf{u}^{(r)} = 30 \text{ m sec}^{-1}$ with the intensity $30\text{--}50 \text{ mm h}^{-1}$. This model is intended primarily to study the efficiency of parallelization and we do not solve the problem of exact reproduction of the flooding of this area. The uniform motion of the rain front ensures that the maximum number of cells are included in the calculations. The presence of $\mathbf{u}^{(r)}$ velocity and heterogeneity of topography lead to a complex structure of water flows and maximum computational load.

2. Parallel CUDA Implementation of the CSPH-TVD Scheme

2.1. Problem Statement

The analysis of computational efficiency depending on the amount of memory used and the method of data transfer between units is convenient when solving a problem with a computational domain in which the maximum number of grid cells contains water. Therefore, we consider the flooding of a large region with complex topography under the action of heavy rains. A good challenge is the choice of a mountainous area, which creates a complex spatial structure of surface water flows with a rapid change in the number of flooded cells due to rainfall/runoff. Examples of such extreme events are numerous and they are actively studied through numerical simulations [25, 32, 34]. We focus on a series of catastrophic events in the Krasnodar region as a geographic reference, when precipitation in the southern mountainous part created horrific flood phenomena, as in Krymsk City and other settlements in 2002 and 2012 [19].

Figure 1 shows the computational domain covering the Black Sea coast of the Krasnodar region. The basis is the digital elevation model $b_0(x_i, y_j) = b(x_i, y_j, t = 0)$ in the equations (2), (3) with a cell size of 15 meters. The flat part of the Krasnodar region is separated from the Black Sea by the Caucasus mountain ranges along the latitudinal direction. Storm rains in the mountains can create flood waves for settlements in the foothills, such as the cities of Krymsk and Abinsk, the Nizhnebakanskaya, Neberdzhavskaya, Kholmskaya, Azovskaya, Severskaya, Verkhnebakansky, Ilsky, Erivansky, Sinegorsk settlements and others. In the simulations, we do not specify the initial water in the region of the Sea of Azov, where the DEM coincides with the sea surface level. We should note the work [34], which in detail analyzes flood situations in the area of the Dyurso River, located inside our computational domain slightly west off the Novorossiysk City (Fig. 1).

2.2. Computing System Topology

The development of a parallel algorithm of the CSPH-TVD method for CPU-multi-GPUs hybrid computing systems is based on OpenMP (CPU – Host level) and CUDA (GPU – Device level) technologies. Using OpenMP at the Host level allows us to create eight parallel threads (OpenMP Threads) that provide parallel data loading and launch of CUDA Kernels of the CSPH-TVD numerical algorithm on eight GPUs (GPU 0–7). We have created two versions of

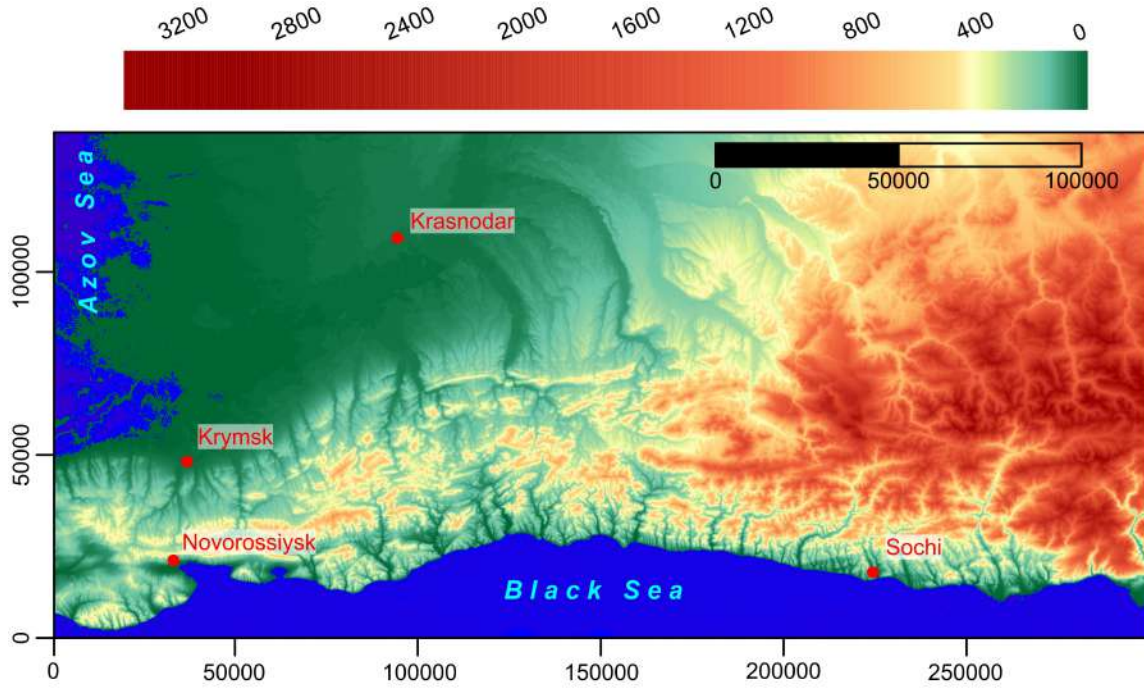


Figure 1. Topography of computational domain in the form of a digital elevation model

the parallel multi-GPU code. The first version uses the GPUDirect approach, which provides direct data exchange between GPUs via the NVLINK interface. Synchronization of calculations on different GPUs at each moment in time occurs directly due to the common address space of multi-GPUs, which is created programmatically at the Device level. This allows one GPU to directly access the memory of another and not involve the CPU (Host). The second version of our code uses the HostCopy approach instead of GPUDirect. This provides data synchronization between GPUs at each time step by copying data from the GPU to the CPU and back.

Figure 2 shows the flow diagram of our code for hydrological simulations based on the system of equations (1)–(3), which are solved by the CSPH-TVD method. The software is intended for CPU-multi-GPU computing systems. The maximum number of GPUs is eight (GPU 0, GPU 1, ..., GPU 7). The flow diagram demonstrates the principle of organizing computations on a hybrid CPU + 8 GPU system using parallel OpenMP-CUDA technologies and different approaches to data transfer between GPUs (GPUDirect and HostCopy). The initial loading of the source data in the form of arrays of hydrodynamic quantities is performed at the Host level. Arrows of different colors show the method of memory transfer between GPUs. The HostCopy approach uses low-speed PCI Express (PCIe) and QuickPath Interconnect (QPI) connections. According to NVIDIA DGX-1 documentation, each V100 GPU has a 16-channel connection to PCIe Switches, which provides a data exchange rate of ~ 16 GB/s. The processor E5-2698v4 (CPUs) in the NVIDIA DGX-1 system has two QPI links with a speed of ~ 9.6 GT/s ($T=Transfer$). One unit of one transfer has a width of $= 80$ bit. Thus, the QPI connection is characterized by a data transfer rate, which is: $2(link) \times 9.6 \text{ GT/s} \times 80 \text{ bit} = 1536 \text{ Gbit/s} = 192 \text{ GB/s}$. GPUDirect technology enables higher data transfer rates of up to 50 GB/s via the NVLink interface. The data transfer rate can vary from 25 GB/s (red arrows in Fig. 2) to 50 GB/s (magenta arrows in Fig. 2) depending on the connection sequence of GPUs on a multi-GPU platform. For example,

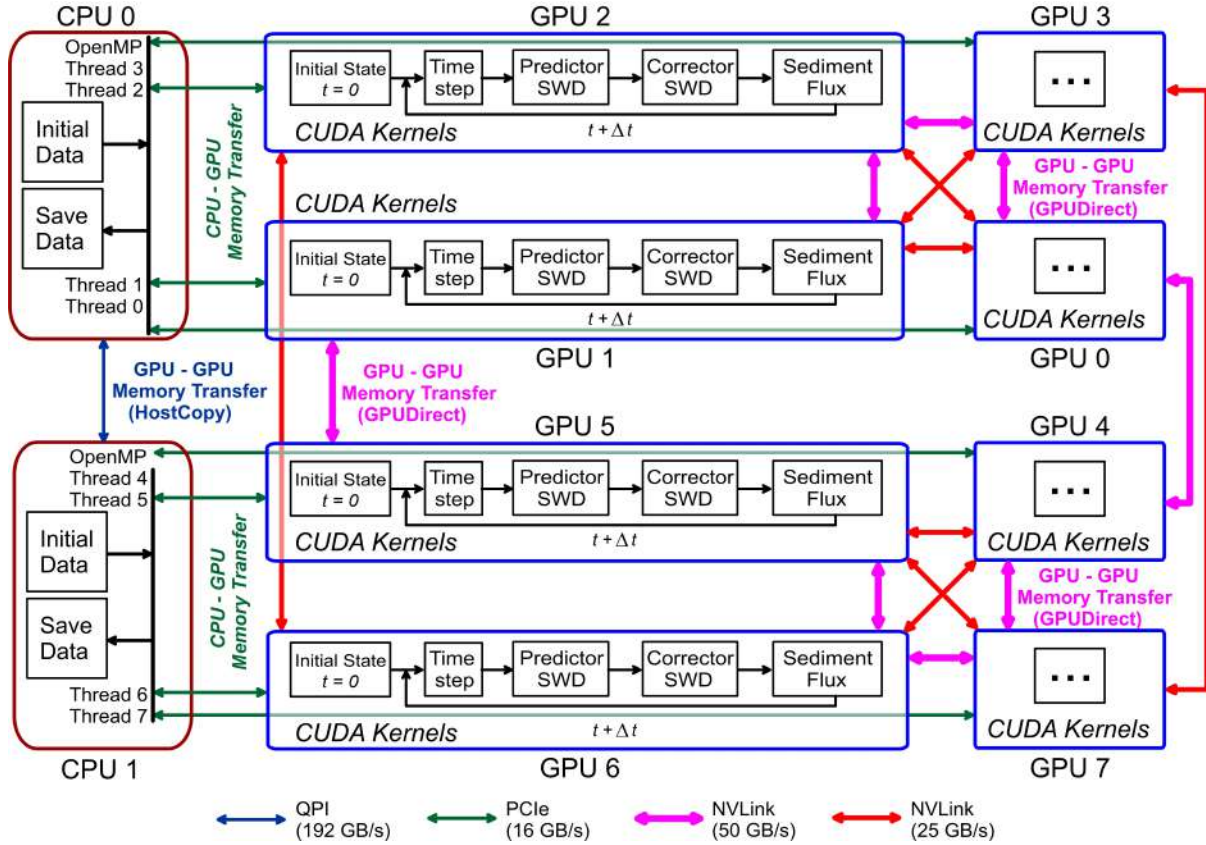


Figure 2. Flow diagram of the parallel OpenMP-CUDA algorithm of CSPH-TVD method for CPU-multi-GPU computing systems

the maximum data transfer rate (50 GB/s) for 8 GPUs is achieved on the NVIDIA DGX-1 supercomputer with the following connection method: GPU 0 – GPU 3 – GPU 2 – GPU 1 – GPU 5 – GPU 6 – GPU 7 – GPU 4.

The implementation of the numerical algorithm CSPH-TVD is based on the following main CUDA Kernels.

- 1) The initial configuration of water flow is created by the CUDA Kernel «Initial State».
- 2) The numerical stability condition of the CSPH-TVD algorithm gives the time step of integration in the CUDA Kernel «Time Step».
- 3) CUDA Kernel «Predictor SWD» ensures the execution of the predictor step when integrating the system of equations (1)–(3) at the Lagrangian stage of the CSPH-TVD method. The result are intermediate values of hydrodynamic quantities (water depth, current velocity, surface-relief height) in grid cells on the time layer $t + \Delta t/2$.
- 4) The corrector for calculating shallow water dynamics of the Lagrangian and Eulerian stages of the CSPH-TVD method is performed in «Corrector SWD». The result is an update of the final values of the hydrodynamic quantities (depth and current velocity) in all cells on the time layer $t + \Delta t$.
- 5) CUDA Kernel «Sediment Flux» is designed to calculate the sediment flux density and update the topography (function b) in all grid cells on the time layer $t + \Delta t$.

The decomposition of the computational domain is carried out for a selected number of GPUs. Figure 3 shows an example of decomposition for the conditions in Fig. 2 with eight computational subdomains for individual GPUs. The process of performing calculations on the GPU

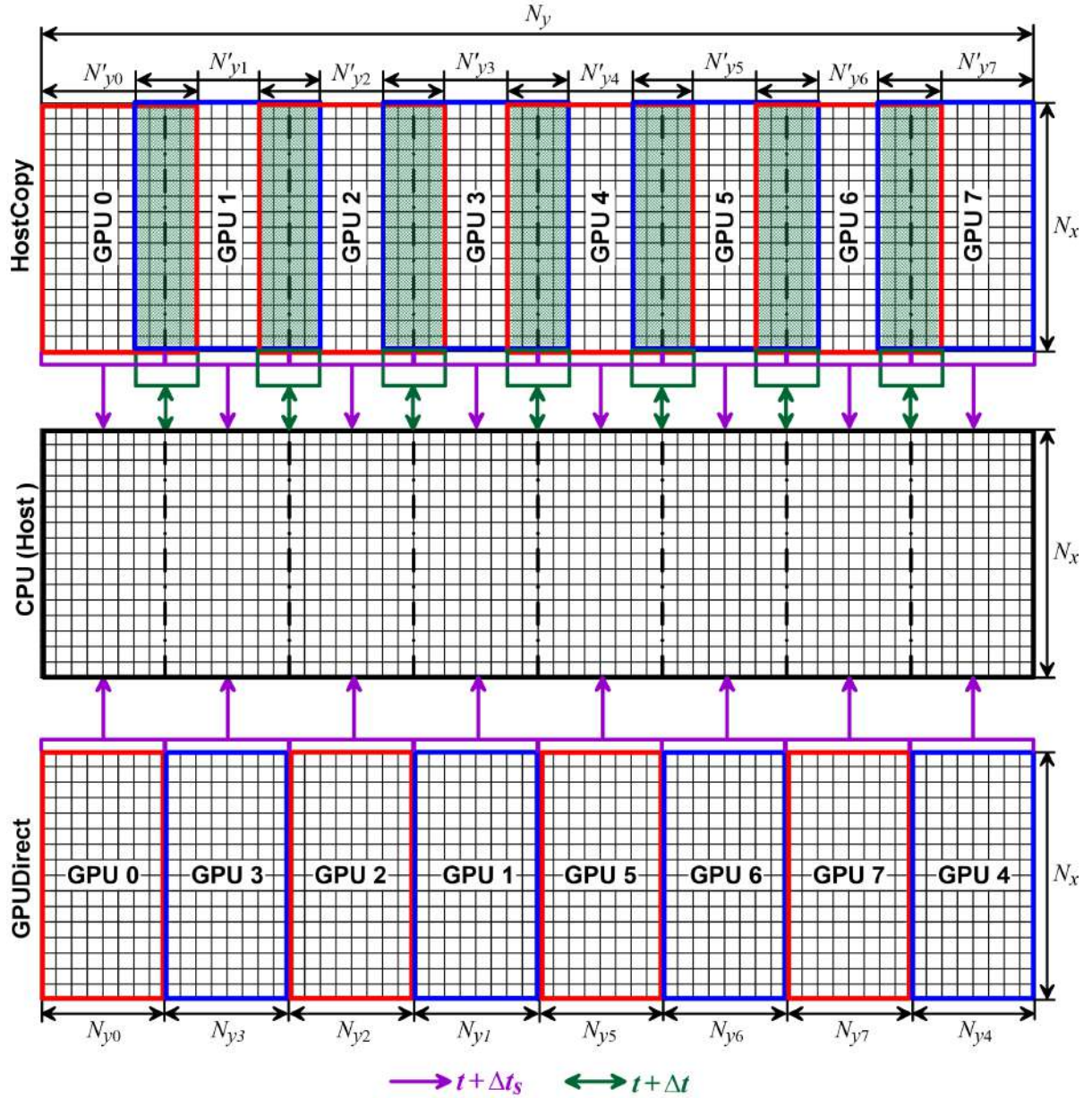


Figure 3. Decomposition of the computational domain for computing system

must be accompanied by data exchange (GPUDirect or HostCopy) to synchronize calculations in the CUDA Kernels. Unloading of newly calculated data back to the Host occurs periodically after a specified number of iterations

$$n_t = \frac{\Delta t_s}{\Delta t} \gg 1, \quad (6)$$

where Δt_s is the time interval between recording the states of the simulations, Δt is the time step of integration from the stability condition of the numerical algorithm [15].

The main differences between the GPUDirect and HostCopy approaches are shown in Fig. 3. The size of the computational subdomains ($N_x \times N'_{yk}$) in the code with HostCopy on each GPU is larger than in the code with GPUDirect support ($N_x \times N_{yk}$), and the condition $N'_{yk} > N_{yk}$ is satisfied. This is due to the synchronization of computations on different GPUs due to the use of neighboring cells located on another GPU in the CSPH-TVD numerical algorithm. Such synchronization in the GPUDirect approach is ensured by direct access to the memory of the

Table 1. A set of models with different computational grids

Model	N_x	N_y	N	Model	N_x	N_y	N
G_0	9 216	18 432	169 869 312	-	-	-	-
G_{x1}	8 192	18 432	150 994 944	G_{y1}	9 216	16 384	150 994 944
G_{x2}	7 168	18 432	132 120 576	G_{y2}	9 216	14 336	132 120 576
G_{x3}	6 144	18 432	113 246 208	G_{y3}	9 216	12 288	113 246 208
G_{x4}	5 120	18 432	94 371 840	G_{y4}	9 216	10 240	94 371 840
G_{x5}	4 096	18 432	75 497 472	G_{y5}	9 216	8 192	75 497 472
G_{x6}	3 072	18 432	56 623 104	G_{y6}	9 216	6 144	56 623 104
G_{x7}	2 048	18 432	37 748 736	G_{y7}	9 216	4 096	37 748 736
G_{x8}	1 024	18 432	18 874 368	G_{y8}	9 216	2 048	18 874 368
G_{x9}	512	18 432	9 437 184	G_{y9}	9 216	1 024	9 437 184
G_{x10}	256	18 432	4 718 592	G_{y10}	9 216	512	4 718 592
G_{x11}	128	18 432	2 359 296	G_{y11}	9 216	256	2 359 296
G_{x12}	128	9 216	1 179 648	G_{y12}	4 608	256	1 179 648
G_{x13}	128	4 608	589 824	G_{y13}	2 304	256	589 824
G_{x14}	128	2 304	294 912	G_{y14}	1 152	256	294 912

neighboring GPU (GPU $k \leftrightarrow \text{GPU } k'$). Using the HostCopy approach to synchronize computations requires first copying neighboring cells from the neighboring GPU to the CPU at each time layer (GPU $k \rightarrow \text{CPU}$ and GPU $k' \rightarrow \text{CPU}$) and then back (CPU $\rightarrow \text{GPU } k'$ and CPU $\rightarrow \text{GPU } k$). Therefore, the HostCopy case requires overlapping computational subdomains on neighboring GPUs. Since CSPH-TVD is a five-point scheme, the total number of cells copied to the Host from each boundary between neighboring GPUs is $4 \times N_x$.

The analysis of parallelization efficiency in Section 4 is based on a set of grids with different numbers of cells of up to $1.7 \cdot 10^8$, which allows us to study the effect of the data load size for different numbers of GPUs. The memory capacity of a single GPU limits the size of the computational domain to a grid with $N \leq 2.5 \cdot 10^8$ cells. Table 1 contains characteristics of the computational grids used to analyze the efficiency of the simulations. Each grid G_{xn} or G_{yn} differs in the number of cells N_x along either the x coordinate (the vertical direction in Fig. 1) or the number N_y along the y coordinate (the horizontal direction in Fig. 1). The set (G_{xn}, G_{yn}) forms a hierarchy of computational domains with different numbers of cells along the two coordinates. This allows us to consider different loading and data transfer scenarios between CPUs/GPUs when using different numbers of GPUs. Since the cell sizes are the same in these grids, the corresponding areas are different.

3. Flash Flood Simulations

Figure 4 shows the result of the simulations at $t = 3$ hours, where the blue color highlights the water distribution. In the mountainous areas, the structure of the flows in the form of numerous streams in accordance with the topography is clearly visible. The region near the Krymsk City is highlighted with a red frame. The main part of the rain mass from the mountainous area goes to the flat northern part or flows into the Black Sea.

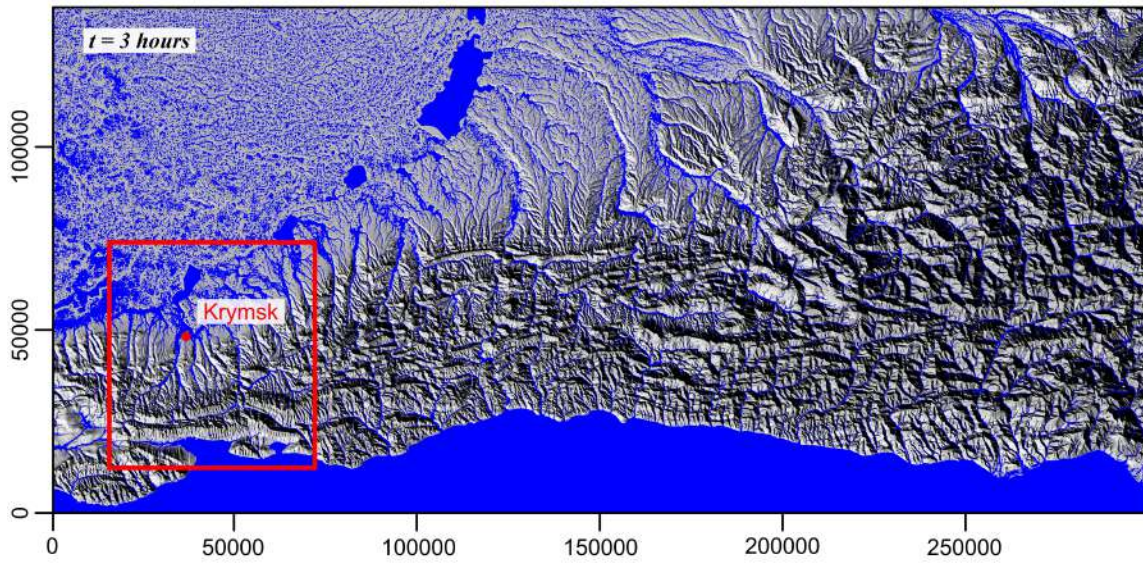


Figure 4. Water distribution 3 hours after the start of a storm rain in one of the models

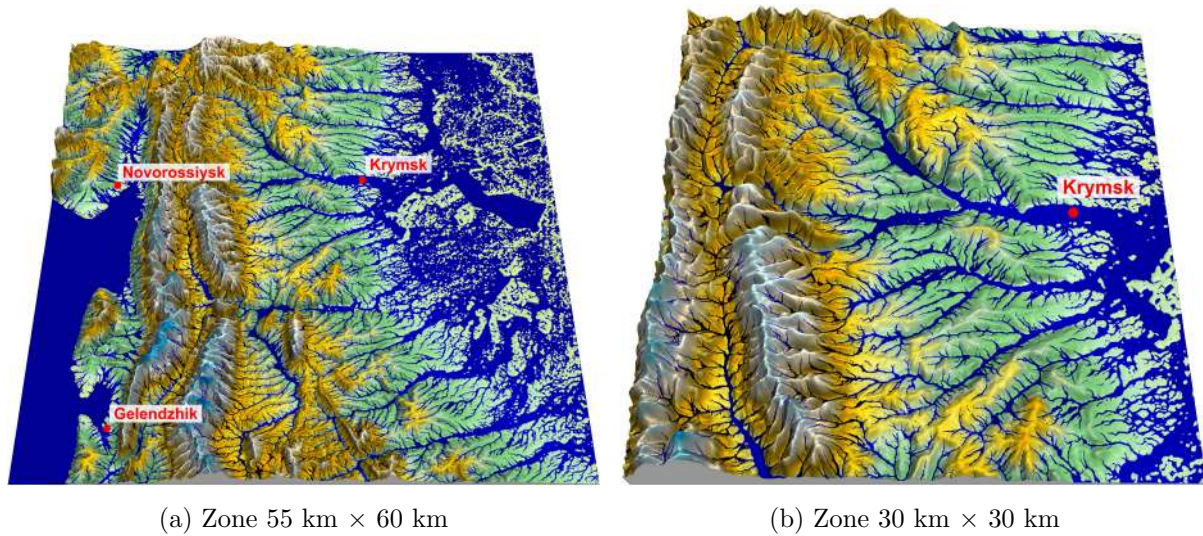


Figure 5. Flooding in the area of the Krymsk City

A more detailed picture of flooding on grid G_0 for the Krymsk catchment area is shown in Fig. 5. We use the 3D relief image as a background to highlight water flows. The specific features of the relative positions of channel structures on the relief and the Krymsk City are the cause of possible catastrophic flooding as in July 6–7, 2012 [19]. There is a bottleneck effect, when the power of several flows combines to form the main water flow in the urban area. The capacity of the channel and floodplain of the Adagum River may not ensure the passage of water under flash flood conditions. Negative factors also include infrastructural structures such as road and railway bridges of the Adagum River with the additional influence of uprooted trees and anthropogenic

debris brought by a powerful flow of water. Our model reproduces a catastrophic rise in water level and flooding of the Krymsk City. These results are preliminary and provide only a general qualitative picture, since the main task is to analyze the efficiency of parallelization in Section 3.

4. Efficiency of Parallelization on Multi-GPU

The computational performance of two implementations of our parallel OpenMP-CUDA code (GPUDirect & HostCopy) for simulating self-consistent surface water and sediment dynamics was carried out on two hybrid supercomputers with the following characteristics.

1) NVIDIA DGX-1 consists of $2 \times \text{CPU}$ (Intel Xeon E5-2698v4, DDR4 512Gb) + $8 \times \text{GPU}$ (NVIDIA TESLA V100, NVLINK, HBM2 256 Gb).

2) Lomonosov-2 supercomputer (Volta 1 and Volta 2 with GPU, Lomonosov Moscow State University) consists of computing nodes that are implemented on the platform $1 \times \text{CPU}$ (Intel Xeon Gold 6142, DDR4 96Gb) + $2 \times \text{GPU}$ (NVIDIA TESLA V100, NVLINK, HBM2 64 Gb) [36].

Figure 6 shows the scalability of the parallel OpenMP-CUDA code (GPUDirect & HostCopy) computational performance from the total number of cells N in the numerical models G_0 , G_{y1} – G_{y14} for different numbers of GPUs on the NVIDIA DGX-1 supercomputer. The simulations with the GPUDirect code were performed using the fastest NVLink connection (50 GB/s).

We propose to evaluate the efficiency of the implementation of the numerical algorithm by calculating the average processing time of one computational cell:

$$\tau_{nGPU}(N) = \frac{t_{nGPU}(N)}{N}, \quad (7)$$

where t_{nGPU} is the average execution time of one computational iteration per time step Δt . The dependences of τ_{nGPU} on the total number of computational cells N for one and several GPUs are shown in Fig. 6a. The value of τ_{nGPU} decreases rather rapidly with increasing N and reaches its minimum value at $N > 10^7$. The model for one GPU (blue line with circles) is characterized by an almost constant level in the region of $10^7 < N < 10^8$ with small deviations within about 3 percent. There is a more significant increase in the time τ_{GPU} up to 10 percent in the region of $N > 10^8$. The G_0 model includes $N \approx 1.7 \cdot 10^8$ cells. We believe that this behavior is due to the additional cost of transferring large amounts of data between the GPU's global memory and the internal memory (register, shared) of its stream multiprocessors (SMs). GPU V100 has 80 such SMs, each containing 32 scalar cores (SCs).

Using 2, 4 and 8 GPUs reduces the time τ_{nGPU} while maintaining the characteristic shape of the dependence $\tau_{nGPU}(N)$ as for 1GPU (see Fig. 6).

Based on the value τ_{GPU} , we can determine the efficiency of using the GPU computing resources, which we will call the efficiency of parallel implementation of a numerical algorithm or, for short, algorithm efficiency

$$\eta_{nGPU}(N) = \frac{\tau_{nGPU}^{(min)}}{\tau_{nGPU}(N)}, \quad (8)$$

where $\tau_{nGPU}^{(min)} = \tau_{nGPU}(N_*)$ is the minimum processing time of one cell at maximum GPU load, corresponding to $N = N_*$. The model with 1GPU yields $\tau_{nGPU}^{(min)} \approx 2.16$ ns and $N_* \approx 2 \cdot 10^7$. The value η_{nGPU} allows one to estimate the efficiency of parallel implementation of the numerical algorithm for a fixed number of GPUs depending on N . The scalability effects described above for $\tau_{nGPU}(N)$ are more pronounced in the $\eta_{nGPU}(N)$ dependencies (see Fig. 6b). GPU load

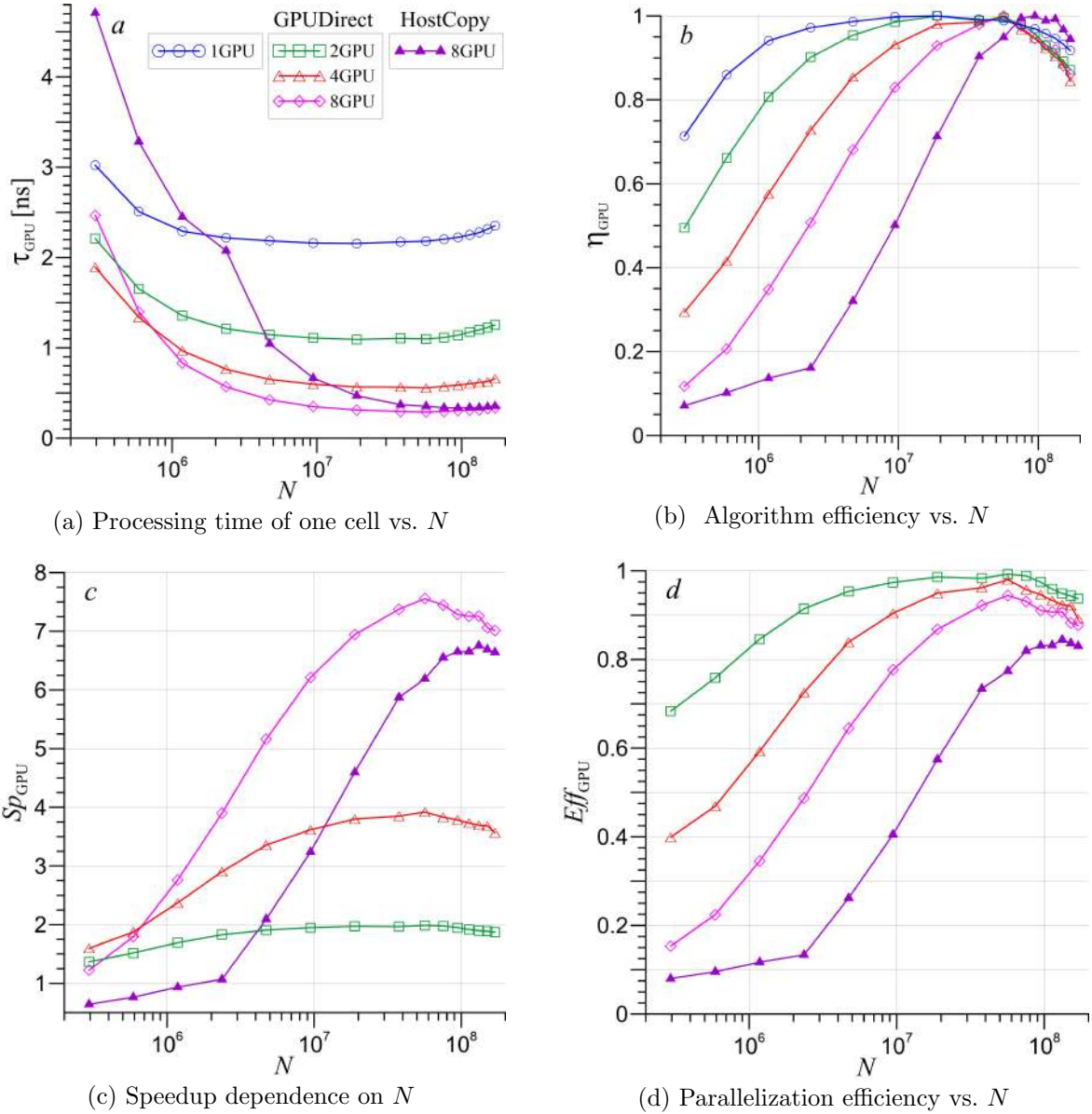


Figure 6. Comparison of the computational performance of parallel OpenMP-CUDA code (GPUDirect & HostCopy) depending on the total number of grid cells N in numerical simulations for 1, 2, 4 and 8 GPUs

efficiency or GPU computational resource utilization efficiency strongly depends on both the number of computational cells N and the number of GPUs. For example, the G_{y14} model for 1 GPU with $N \approx 3 \cdot 10^5$ yields algorithm efficiency $\eta_{nGPU} \approx 0.7$. The efficiency of the G_{y8} model with $N \approx 2 \cdot 10^7$ reaches its maximum value of $\eta_{nGPU} \approx 1$. It decreases to $\eta_{nGPU} \approx 0.92$ in the G_0 model ($N \approx 1.7 \cdot 10^8$). With an increase in the number of GPUs, the algorithm efficiency decreases several times for small values of $N < 10^6$, and its maximum shifts to the region of large N ($N_* \approx 6 \cdot 10^7$). This is due to both a decrease in the number of computational cells per GPU in multi-GPU parallelization and the cost of data transfer between GPUs.

First of all, the values τ_{nGPU} and η_{nGPU} are intended to evaluate the performance of the computational algorithm on a single GPU and allow one to determine the most optimal grid resolution for $\tau_{nGPU}(N) = \tau_{nGPU}^{(min)}$. In multi-GPU parallelization, these characteristics can also be used to find the most optimal number of computational cells in each subdomain on 1 GPU.

The positions of the extrema of the functions $\tau_{nGPU}(N)$ and $\eta_{nGPU}(N)$ determine the optimal grid resolution that ensures maximum computational performance. Figures 6a and 6b show shifts of such extrema to the right toward larger N with an increase in the number of GPUs used. The calculated dependence $\tau_{nGPU}(N)$ for one GPU (see Fig. 6a, blue circles and line) makes it possible to estimate the efficiency of multi-GPU parallelization for any number of GPUs even without performing these simulations.

The impact of data transfer between GPUs and the load on one GPU during multi-GPU parallelization can be conveniently analyzed using standard quantities, such as speedup (Sp_{nGPU}) and parallelization efficiency Eff_{nGPU} :

$$Sp_{nGPU} = \frac{t_{1GPU}}{t_{nGPU}}, \quad Eff_{nGPU} = \frac{Sp_{nGPU}}{nGPU}. \quad (9)$$

The quantity Eff_{nGPU} differs from η_{nGPU} in that it allows one to estimate the efficiency of multi-GPU parallelization of a computational algorithm depending on the number of GPUs used in the simulations for a fixed value of N . Figure 6c, d shows the $Sp_{nGPU}(N)$ and $Eff_{nGPU}(N)$ dependences for 2, 4 and 8 GPUs in the GPUDirect code. We plot the corresponding values for 8 GPUs in the HostCopy code for comparison, which on average turns out to be several times slower and less efficient than the GPUDirect code.

If the behavior of the curves $\eta_{nGPU}(N)$ and $Eff_{nGPU}(N)$ in Fig. 6b, d is compared for 2, 4 and 8 GPUs, these dependencies correlate well with each other. This allows using two approaches to analyzing the performance of calculations (algorithm efficiency and parallelization efficiency) to estimate the data transfer time between GPUs from the following equations:

$$t_{1GPU} = \frac{N \cdot \tau_{1GPU}^{(min)}}{\eta_{1GPU}(N)}, \quad t_{nGPU} = \frac{N \cdot \tau_{1GPU}}{nGPU \cdot \eta_{1GPU}(N_{GPU})} + N_B \cdot \tau_{nGPU}^{(MT)}, \quad (10)$$

where $\tau_{nGPU}^{(MT)}$ is the specific time (per cell) of memory transfer between $nGPU$, $N_{GPU} = N/nGPU$ is the number of computational cells processed on one GPU, N_B is the number of computational cells along the grid boundary between GPUs. The expressions (10) predict the efficiency of multi-GPU parallelization depending on N , $nGPU$, N_B and parallelization technologies (GPUDirect, HostCopy). It is sufficient to determine the dependence $t_{1GPU}(N)$ in simulations with one GPU for different values of N , then, based on this dependence, calculate $\tau_{1GPU}^{(min)}$ and $\eta_{1GPU}(N)$, and also determine $\tau_{nGPU}^{(MT)}$ for one fixed value of N .

An important characteristic of the computational algorithm is the memory size m_{cell} allocated to the GPU in the simulation per cell, which limits the size of the computational domain in the model. Our parallel OpenMP-CUDA algorithm of the CSPH-TVD method requires $m_{cell} \approx 128$ Bit/cell. This gives the following limit on the size of the computational domain for different numbers of GPU V100: $N \simeq nGPU \cdot 2.6 \cdot 10^8$.

We analyzed the influence of the high-speed NVLink connection type (25 or 50 GB/s) related to the sequence of GPUs used in decomposition of the computational domain. Simulations show that the performance of our GPUDirect code at $N > 10^7$ is practically independent of the choice of the NVLink connection type between GPUs. The deviations are no more than 0.5 percent, which corresponds to the level of thermal noise. Simulation variants with NVLink 50 GB/s at small values of $N \leq 10^6$ turn out to be ~ 5 –10 percent more efficient than with NVLink 25 GB/s.

Similar estimates were obtained for different decomposition variants (see models G_{yn} and G_{xn} in Tab. 1). Models G_{xn} also turn out to be $\simeq 5$ –10 percent more efficient than models with grids G_{yn} for values of $N < 10^7$.

We can compare the NVIDIA DGX-1 system and the Lomonosov-2 supercomputer with each other only for simulations on one and two GPUs [36]. Our analysis showed that using Lomonosov-2 is several percent more efficient than NVIDIA DGX-1. This may be due to the presence of a more powerful CPU and a more efficient cooling system on Lomonosov-2.

Our performance analysis of various hydrodynamic codes for CPUs shows that parallel OpenMP versions of these codes on modern CPUs with 40 cores (DGX-1) are 100–1000 times slower, than parallel OpenMP-CUDA programs, depending on the number of GPUs [8, 13].

Conclusion

We studied numerical hydrodynamic models that require large computational grids with the number of cells $N > 10^8$ using the example of simulating a flash flood over a large area of $\sim 5 \cdot 10^4 \text{ km}^2$ with a good spatial resolution $\sim 15 \text{ m}$, which is provided by a digital elevation model based on satellite data. Such numerical models require about 20–30 GB of GPU memory to store them, which limits their use on a single GPU and can lead to a decrease in computational efficiency.

Our numerical rainfall/runoff simulations for the southern mountainous part of the Krasnodar region are aimed at studying the consequences of catastrophic floods that have repeatedly occurred in the past. We have shown the formation of merging channel flows, the power of which increases in the mountainous area. A powerful flow is formed in Krymsk City as a result of the bottleneck effect, which can cause emergency situations for this and other settlements.

We compared the computational efficiency of parallel simulations on CPU-multi-GPU computing systems for two implementations of parallel OpenMP-CUDA using GPUDirect and HostCopy technologies. The maximum speedup value Sp_{nGPU} in the parallel multi-GPU code with GPUDirect for eight GPUs reaches ~ 7.6 , which provides parallelization efficiency $Eff_{nGPU} \sim 0.95$. Multi-GPU code with HostCopy requires more operations compared to GPUDirect, which leads to delays in calculations, and as a result, reduces the efficiency of this implementation several times.

We propose to use a special dimensionless coefficient to estimate the efficiency of a numerical algorithm for solving evolutionary problems using the example of computational fluid dynamics for the grid approach. This characteristic is defined through the average processing time of one computational cell τ_{nGPU} . The value τ_{nGPU} has a non-monotonic dependence on the total number of cells N and has a minimum $\tau_{GPU}^{(min)}$ for some $N = N_*$. We define the algorithm efficiency η_{nGPU} to evaluate the efficiency of using the computational resources of a fixed number of GPUs depending on N . The value $\eta_{nGPU} = 1$ means the maximum possible GPU load for a specific parallel implementation of the numerical algorithm. It is important that the calculation of the standard characteristic of computational performance Eff_{nGPU} together with the algorithm efficiency η_{nGPU} makes it possible to predict the scalability of the computational algorithm with an increase in the number of cells and the number of GPUs in numerical simulations.

Note that the value η_{nGPU} can be used to evaluate the efficiency of a numerical algorithm on other computing platforms, including CPU.

Acknowledgements

This work is supported by the Russian Science Foundation (grant no. 23-71-00016, <https://rscf.ru/project/23-71-00016/>). The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References






1. Aamodt, T.M., Fung, W.W.L., Rogers, T.G.: General-Purpose Graphics Processor Architectures. Springer Cham (2018)
2. Belikov, V., Borisova, N., Vasileva, E., *et al.*: A Numerical Hydrodynamic Model of a Long Segment of the Ural River and Its Application to Assessing the Inundation Risk of Residential Areas by Floods and Breakthrough Waves. *Water Resources* 51(5), 654–665 (2024). <https://doi.org/10.1134/S0097807824701008>
3. Bocharov, A., Evstigneev, N., Petrovskiy, V., *et al.*: Implicit method for the solution of supersonic and hypersonic 3D flow problems with Lower-Upper Symmetric-Gauss-Seidel preconditioner on multiple graphics processing units. *Journal of Computational Physics* 406, 109189 (2020). <https://doi.org/10.1016/j.jcp.2019.109189>
4. Diaz, M.C., Fernandez-Nieto, E., Ferreiro, A., *et al.*: Two-dimensional sediment transport models in shallow water equations. a second order finite volume approach on unstructured meshes. *Computer Methods in Applied Mechanics and Engineering* 198(33-36), 2520–2538 (2009). <https://doi.org/10.1016/j.cma.2009.03.001>
5. Dominguez, J.M., Fourtakas, G., Altomare, C., *et al.*: DualSPHysics: from fluid dynamics to multiphysics problems. *Computational Particle Mechanics* 9, 867–895 (2022). <https://doi.org/10.1007/s40571-021-00404-2>
6. Dominguez, J., Crespo, A., Valdez-Balderas, D., *et al.*: New multi-GPU implementation for smoothed particle hydrodynamics on heterogeneous clusters. *Computer Physics Communications* 184(8), 1848–1860 (2013)
7. Dong, B., Huang, B., Tan, C., *et al.*: Multi-GPU parallelization of shallow water modelling on unstructured meshes. *Journal of Hydrology* 657, 133105 (2025). <https://doi.org/10.1016/j.jhydro1.2025.133105>
8. Dyakonova, T., Khoperskov, A., Khrapov, S.: Numerical Model of Shallow Water: The Use of NVIDIA CUDA Graphics Processors. *Communications in Computer and Information Science* 687, 132–145 (2016). https://doi.org/10.1007/978-3-319-55669-7_11
9. Gorobets, A., Bakhvalov, P.: Heterogeneous CPU+GPU parallelization for high-accuracy scale-resolving simulations of compressible turbulent flows on hybrid supercomputers. *Computer Physics Communications* 271, 108231 (2022). <https://doi.org/10.1016/j.cpc.2021.108231>

10. Hafiyyan, Q., Harlan, D., Adityawan, M.B., *et al.*: 2D Numerical Model of Sediment Transport Under Dam-break Flow Using Finite Element. International Journal on Advanced Science Engineering and Information Technology 11(6), 2476–2481 (2021). <https://doi.org/10.18517/ijaseit.11.6.14484>
11. Jayaratne, R., Takayama, Y., Shibayama, T.: Applicability of suspended sediment concentration formulae to large-scale beach morphological changes. In: Lynett, P., McKee Smith, J.e. (eds.) Coastal Engineering Proceedings, vol. 1 (33), pp. 1–15. Coastal Engineering Research Council (2012). <https://doi.org/10.9753/icce.v33.sediment.57>
12. Khoperskov, A., Khrapov, S., Klikunova, A., *et al.*: Efficiency of Using GPUs for Reconstructing the Hydraulic Resistance in River Systems Based on Combination of High Performance Hydrodynamic Simulation and Machine Learning. Lobachevskii Journal of Mathematics 45(7), 3085–3096 (2024). <https://doi.org/10.1134/S199508022460376X>
13. Khrapov, S.: Numerical modeling of two-dimensional gas-dynamic flows in multicomponent nonequilibrium media. Mathematical Physics and Computer Simulation 28(1), 60–88 (2025)
14. Khrapov, S., Pisarev, A., Kobelev, I., *et al.*: The numerical simulation of shallow water: Estimation of the roughness coefficient on the flood stage. Advances in Mechanical Engineering 2013, 787016 (2013). <https://doi.org/10.1155/2013/787016>
15. Khrapov, S.: Numerical modeling of hydrodynamic accidents: Erosion of dams and flooding of territories. Vestnik of the St. Petersburg University: Mathematics 56(2), 261–272 (2023). <https://doi.org/10.1134/s1063454123020085>
16. Khrapov, S., Khoperskov, A.: Application of graphics processing units for self-consistent modelling of shallow water dynamics and sediment transport. Lobachevskii Journal of Mathematics 41(8), 1475–1484 (2020). <https://doi.org/10.1134/S1995080220080089>
17. Khrapov, S., Khoperskov, A.: Study of the Effectiveness of Parallel Algorithms for Modeling the Dynamics of Collisionless Galactic Systems on GPUs. Supercomputing Frontiers and Innovations 11(3), 27–44 (2024). <https://doi.org/10.14529/jsfi240302>
18. Klikunova, A., Polyakov, M., Khrapov, S., *et al.*: Problem of building high-quality predictive model of river hydrology: the combined use of hydrodynamic simulations and intelligent computing. Communications in Computer and Information Science 1909, 191–205 (2023). https://doi.org/10.1007/978-3-031-44615-3_13
19. Kotlyakov, V.M., Desinov, L.V., Dolgov, S.V., *et al.*: Flooding of July 6-7, 2012, in the town of Krymsk. Regional Research of Russia 3, 32–39 (2013). <https://doi.org/10.1134/S2079970513010061>
20. Li, W., Hu, P., Pahtz, T., *et al.*: Limitations of empirical sediment transport formulas for shallow water and their consequences for swash zone modelling. Journal of Hydraulic Research 55(1), 114–120 (2017). <https://doi.org/10.1080/00221686.2016.1212942>
21. Liu, T., Trim, S.J., Ko, S.B., *et al.*: The multi-GPU Wetland DEM Ponding Model. Computers & Geosciences 199, 105912 (2025). <https://doi.org/10.1016/j.cageo.2025.105912>

22. de Luna, T.M., Diaz, M.J.C., Madronal, C.P.: On a sediment transport model in shallow water equations with gravity effects. In: Kreiss, G., Lotstedt, P., Malqvist, A., Neytcheva, M. (eds.) *Numerical Mathematics and Advanced Applications*, pp. 655–661. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11795-4_70
23. Macca, E., Avgerinos, S., Castro-Diaz, M.J., *et al.*: A semi-implicit finite volume method for the Exner model of sediment transport. *Journal of Computational Physics* 499, 112714 (2024). <https://doi.org/10.1016/j.jcp.2023.112714>
24. McKevitt, J., Vorobyov, E.I., Kulikov, I.: Accelerating Fortran codes: A method for integrating Coarray Fortran with CUDA Fortran and OpenMP. *Journal of Parallel and Distributed Computing* 195, 104977 (2025). <https://doi.org/10.1016/j.jpdc.2024.104977>
25. Mignot, E., Paquier, A., Haider, S.: Modeling floods in a dense urban area using 2D shallow water equations. *Journal of Hydrology* 327(1-2), 186–199 (2006). <https://doi.org/10.1016/j.jhydrol.2005.11.026>
26. Narasiman, V., Shebanow, M., Lee, C.J., *et al.*: Improving GPU performance via large warps and two-level warp scheduling. In: 2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Porto Alegre, Brazil, pp. 308–317. IEEE (2011)
27. Ndengna, A.R.N., Njifenjou, A.: A well-balanced PCCU-AENO scheme for a sediment transport model. *Ocean Systems Engineering* 12(3), 359–384 (2022). <https://doi.org/10.12989/ose.2022.12.3.359>
28. Shao, X., Wang, X.: *Introduction to River Dynamics*. Tsinghua University Press Co., Beijing, China (2005)
29. Simonov, A.S., Semenov, A.S., Shcherbak, A.N., *et al.*: The High Performance Interconnect Architecture for Supercomputers. *Supercomputing Frontiers and Innovations* 10(2), 127–136 (2023). <https://doi.org/10.14529/jsfi230208>
30. Siviglia, A., Vanzo, D., Toro, E.: A splitting scheme for the coupled Saint-Venant-Exner model. *Advances in Water Resources* 159, 104062 (2022). <https://doi.org/10.1016/j.advwatres.2021.104062>
31. Sukhinov, A.I., Protsenko, E.A., Protsenko, S.V.: WAVEWATCH III Hybrid Parallelization for Azov Sea Wave Modeling. *Supercomputing Frontiers and Innovations* 11(1), 81–96 (2024). <https://doi.org/10.14529/jsfi240104>
32. Taccone, F., Antoine, G., Delestre, O., *et al.*: A new criterion for the evaluation of the velocity field for rainfall-runoff modelling using a shallow-water model. *Advances in Water Resources* 140, 103581 (2020). <https://doi.org/10.1016/j.advwatres.2020.103581>
33. Valles, P., Fernandez-Pato, J., Morales-Hernandez, M., *et al.*: A 2D shallow water flow model with 1D internal boundary condition for subgrid-scale topography. *Advances in Water Resources* 189, 104716 (2024). <https://doi.org/10.1016/j.advwatres.2024.104716>
34. Vasileva, E.S., Alekseyuk, A.I., Belyakova, P.A., *et al.*: Numerical modeling of the behavior of a destructive rain flood on a mountain river. *Water Resources* 46(1), 45–55 (2019). <https://doi.org/10.1134/S0097807819070169>

35. Vatyukova, O., Klikunova, A., Vasilchenko, A., *et al.*: The problem of effective evacuation of the population from floodplains under threat of flooding: algorithmic and software support with shortage of resources. *Computation* 11(8), 150 (2023). <https://doi.org/10.3390/computation11080150>
36. Voevodin, V., Antonov, A., Nikitenko, D., *et al.*: Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community. *Supercomputing Frontiers and Innovations* 6(2), 4–11 (2019). <https://doi.org/10.14529/jsfi190201>
37. Zhang, X., Guo, X., Weng, Y., *et al.*: Hybrid MPI and CUDA paralleled finite volume unstructured CFD simulations on a multi-GPU system. *Future Generation Computer Systems* 139, 1–16 (2023). <https://doi.org/10.1016/j.future.2022.09.005>
38. Zolfaghari, H., Becsek, B., Nestola, M.G., *et al.*: High-order accurate simulation of incompressible turbulent flows on many parallel GPUs of a hybrid-node supercomputer. *Computer Physics Communications* 244, 132–142 (2019). <https://doi.org/10.1016/j.cpc.2019.06.012>

Comparison of Quantum-Chemical Programs and Methods for the Calculation of Enthalpies of Formation of High-Energy Tetracyclic Compounds

Vadim M. Volokhov¹ , Vladimir V. Parakhin² , Elena S. Amosova¹ ,
David B. Lempert¹ , Vladimir V. Voevodin³ 

© The Authors 2025. This paper is published with open access at SuperFri.org

A comparative study was carried out on the thermochemical properties of a series of high-energy tetracyclic compounds containing amino, cyano, azido, and dinitrophenyl groups. Various quantum-chemical methods were employed to calculate the gas-phase enthalpies of formation, including the B3LYP functional with 6-311+G(2d,p) and cc-pVTZ basis sets, the composite G4MP2 method implemented in Gaussian 09, and a G4MP2-based scheme adapted for implementation in NWChem. The G4MP2 method was used as a reference for accuracy, against which the results of other approaches were evaluated. It is shown that the use of NWChem and reaction-based schemes yields enthalpy values close to those obtained by G4MP2, while significantly reducing computational costs. Structural factors affecting the enthalpy of formation are analyzed, along with differences in the IR absorption spectra. The results confirm the applicability of various theoretical levels for the thermochemical evaluation of promising high-energy materials.

Keywords: high-energy materials, tetracyclic compounds, enthalpy of formation, quantum chemical calculations, isodesmic reactions, atomization, IR spectra, high-performance computing.

Introduction

The enthalpy of formation (ΔH_f°) is one of the key characteristics determining the energy performance of high-energy-density materials (HEDMs) [1–6]. High enthalpy of formation values can be achieved through the incorporation of nitrogen-rich aromatic cores into polycyclic molecular scaffolds [7–10], the energetic properties of which have recently been investigated by our research group [11–18]. Among such structures are tetracyclic compounds – tris(1,2,4-triazolo)-1,3,5-triazines and tris(1,2,3-triazolo)benzenes – the functionalization of which with endothermic nitrogen-containing groups is expected to further increase their enthalpy of formation. To investigate the effect of such combinations, this study focuses on tetracycles bearing amino, cyano, and azido substituents (Fig. 1):

2,6,10-triamino-tris([1,2,4]triazolo)[1,5-a:1',5'-c:1'',5''-e][1,3,5]triazine (**1a**),
2,6,10-tricyano-tris([1,2,4]triazolo)[1,5-a:1',5'-c:1'',5''-e][1,3,5]triazine (**2a**),
2,6,10-triazido-tris([1,2,4]triazolo)[1,5-a:1',5'-c:1'',5''-e][1,3,5]triazine (**3a**),
2,5,8-triamino-tris([1,2,3]triazolo)[1,2-d:3,4-d':5,6-d'']benzene (**1b**),
2,5,8-tricyano-tris([1,2,3]triazolo)[1,2-d:3,4-d':5,6-d'']benzene (**2b**) and
2,5,8-triazido-tris([1,2,3]triazolo)[1,2-d:3,4-d':5,6-d'']benzene (**3b**), which currently remain hypothetical structures,

as well as 2,5,8-tri(2,4-dinitrophenyl)-tris([1,2,3]triazolo)[1,2-d:3,4-d':5,6-d'']benzene (**4**), which has previously been synthesized and characterized as a highly stable and promising, exhibiting exceptionally high thermal stability (with a decomposition onset temperature above 400 °C) and low mechanical sensitivity, with an impact energy threshold of 18 J [19, 20].

¹Federal Research Center of Problems of Chemical Physics and Medicinal Chemistry of the Russian Academy of Sciences, Chernogolovka, Moscow Region, Russian Federation

²N.D. Zelinskiy Institute of Organic Chemistry of the Russian Academy of Sciences, Moscow, Russian Federation

³Research Computing Center of Lomonosov Moscow State University, Moscow, Russian Federation

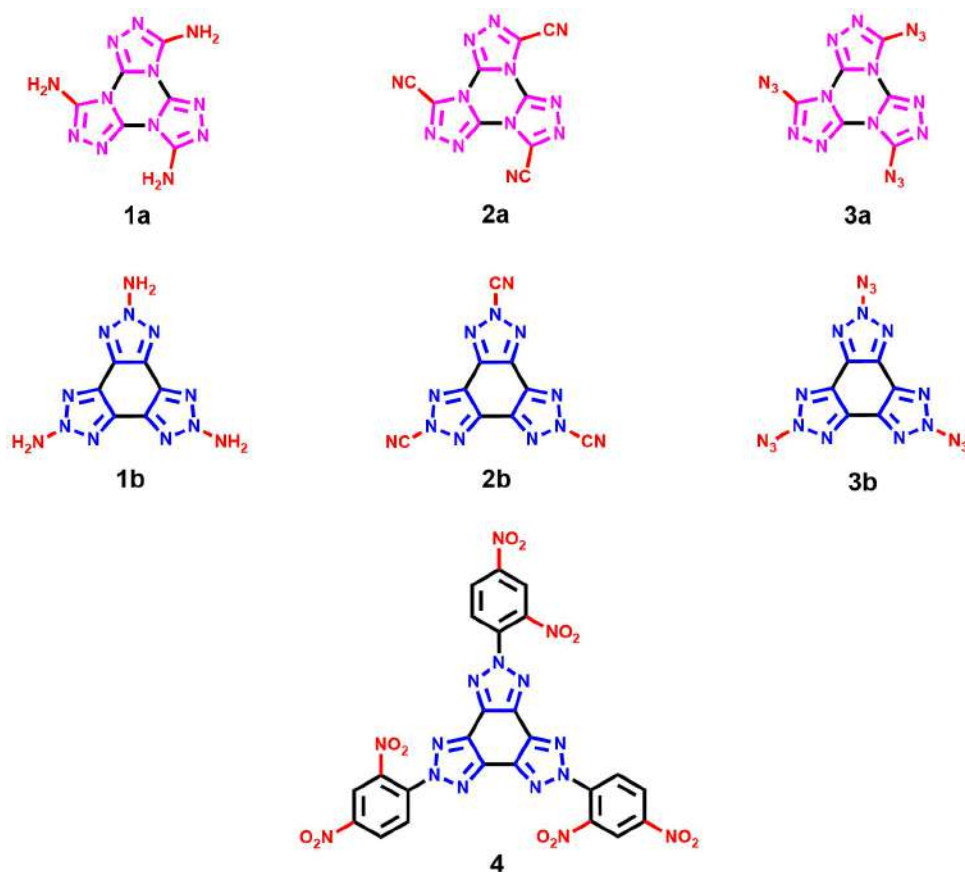


Figure 1. Compounds under investigation – a series of tetracyclic structures **1a,b**, **2a,b**, **3a,b**, and **4**

The article is organized as follows. Section 1 presents the computation methods employed in this work. In Section 2 we discuss the obtained results. Section 3 contains the analysis of method performance on different computational resources. Conclusion summarizes the study and suggests directions for future research.

1. Computation Method

Quantum-chemical calculations were carried out using the GAUSSIAN 09 [21] and NWChem [22] program packages. Geometry optimization was performed using the hybrid density functional B3LYP [23, 24] with the 6-311+G(2d,p) basis set. The stability of the optimized molecular structures was confirmed by calculating vibrational frequencies using analytical first and second derivatives, without applying anharmonic corrections (i.e., the absence of imaginary frequencies). The gas-phase enthalpy of formation of the investigated compounds was calculated using the B3LYP hybrid functional [23, 24] with 6-311+G(2d,p) and cc-pVTZ basis sets [25], as well as the composite G4MP2 [26, 27] method. For the NWChem calculations, a custom module was developed to reproduce the sequence of computational steps of the G4MP2 composite method as described in the literature [26, 28], using the corresponding published equations to obtain the total molecular energy.

$$E_0[G4(MP2)] = CCSD(FC, T)/6 - 31G(d) + \Delta E_{MP2} + \Delta E_{HF} + \Delta E(SO) + E(HLC) + E(ZPE),$$

where CCSD(FC,T)/6-31G(d) is the energy calculation at the triples-augmented coupled cluster level of theory, CCSD(T), with the 6-31G(d) basis set, using frozen core; ΔE_{MP2} and ΔE_{HF} are the energy corrections calculated by the MP2 and HF methods accordingly, $\Delta E(SO)$ is the spin-orbit correction, $E(HLC)$ is the higher-level correction, and $E(ZPE)$ is the zero-point energy.

The IR absorption spectra were calculated using the hybrid density functional B3LYP with the cc-pVTZ basis set. A scaling factor of 0.967 was applied to the computed harmonic frequencies [29].

The enthalpy of formation of the investigated compounds was calculated using two approaches: (I) based on atomization reactions, and (II) based on formation reactions.

The method based on the atomization reaction for the $C_wH_xN_yO_z$ molecule consisted of the following steps:

1. The atomization energy is calculated by subtracting the total energy of the molecule from the sum of the total energies of the atoms, where the total energies are determined by quantum chemical calculations.
2. The enthalpy of formation at 0K is calculated by subtracting the atomization energy from the sum of the enthalpies of formation of gaseous atomic components as stated in the NIST-JANAF database of thermochemical parameters [30].
3. The enthalpy of formation at 298.15K is calculated by introducing thermal corrections both for the molecule and for the atoms, obtained from the quantum-chemical calculation of the molecule or known from experiment (or calculated from experimental molecular constants).

Reaction schemes used for the calculation of enthalpy of formation are shown in Fig. 2. For the benzene-based compounds (**1–3b**, **4**), the total energies of the reactants and products were calculated using the B3LYP hybrid density functional [23, 24] with 6-311+G(2d,p) basis set. For the 1,3,5-triazine-based compounds (**1–3a**), the total energies were calculated using the ω B97XD functional [31] with cc-pVTZ basis set. The atomization enthalpies used in the calculations for all investigated compounds were obtained using the composite G4MP2 method [26, 27].

2. Results and Discussion

2.1. Enthalpy of Formation

Figure 3 presents the main geometric parameters of the optimized structures. Table 1 gathers the results of calculation of the enthalpy of formation by various methods.

The use of the B3LYP functional with the 6-311+G(2d,p) basis set for atomization-based calculations yielded the highest enthalpy of formation values for all compounds under investigation. Based on prior experience, these values appear to be overestimated, as this functional does not provide high accuracy in thermochemical predictions due to its limited treatment of electron correlation, dispersion effects, and basis set incompleteness. Upon replacing the basis set with cc-pVTZ, which had demonstrated good performance in previous studies [17, 18], the resulting values were on average 108 kJ/mol lower compared to those obtained with 6-311+G(2d,p). The root-mean-square deviation (RMSD) of these values from the results obtained using the composite G4MP2 method (excluding compound **4**) was 57 kJ/mol. The smallest deviation – 5 kJ/mol – was observed for compound **2b**. When using the NWChem package with a custom implementation analogous to the G4MP2 method, the RMSD from the G4MP2 results obtained in Gaussian 09 was less than 2 kJ/mol. The module developed for these calculations does not

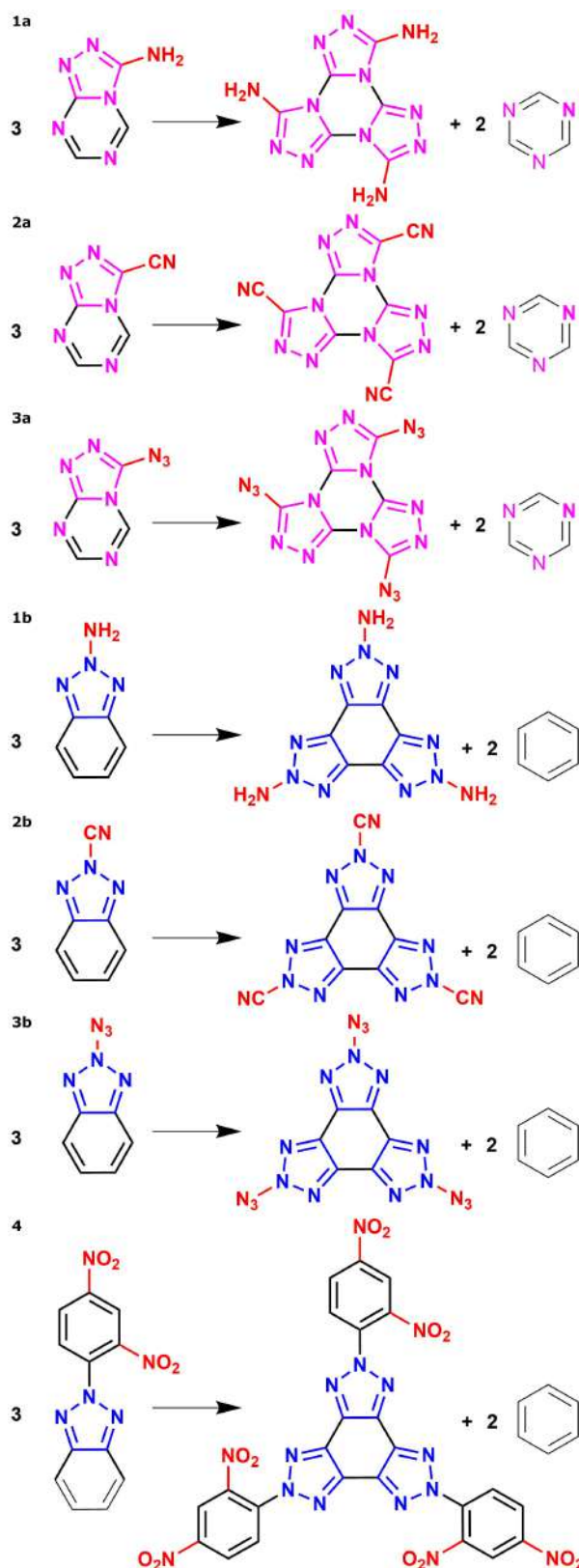


Figure 2. Isodesmic reaction schemes used to calculate the enthalpy of formation of investigated tetracycles **1a,b**, **2a,b**, **3a,b** and **4**

exactly replicate the computational sequence of the G4MP2 method as implemented in GAUSSIAN 09, but rather employs analogous basis sets and functions available in NWChem, which

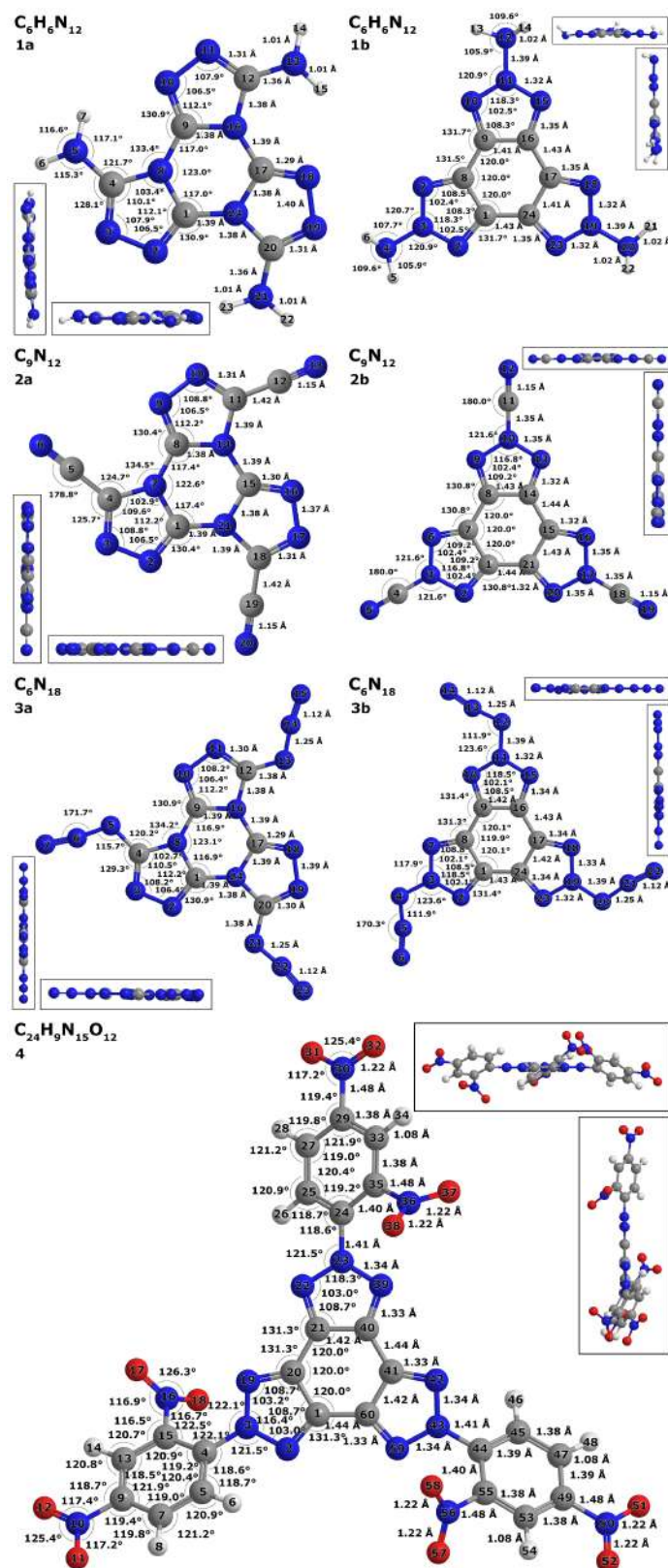


Figure 3. Structures and geometric parameters (in Å and °) of of tetracycles **1a,b**, **2a,b**, **3a,b** and **4** (calculated at B3LYP/6-311+G(2d,p) level)

may also contribute to the observed discrepancies. In the reaction scheme-based calculations, the deviation of the obtained results from the G4MP2 values was less than 2 kJ/mol for most of the

Table 1. Gas-phase enthalpies of formation of tetracycles **1a,b**, **2a,b**, **3a,b**, and **4** calculated by different methods (kJ/mol, (*kJ/kg*))

No	Formula	α	B3LYP	G4MP2	NWChem G4MP2	Reactions	B3LYP/ cc-pVTZ
1a	C ₆ H ₆ N ₁₂	0.00	827.9 (3364.5)	765.6 (3109.6)	764.4 (3104.9)	747.3 (3036.6)	749.5 (3045.7)
2a	C ₉ N ₁₂	0.00	1461.6 (5294.8)	1352.6 (4897.4)	1352.6 (4897.4)	1352.5 (4899.6)	1372.1 (4970.7)
3a	C ₆ N ₁₈	0.00	1874.9 (5785.8)	1849.9 (5706.0)	1850.2 (5707.0)	1842.1 (5684.6)	1754.0 (5412.8)
1b	C ₆ H ₆ N ₁₂	0.00	1169.0 (4750.6)	1116.5 (4535.0)	1117.5 (4539.2)	1118.2 (4543.9)	1096.5 (4455.9)
2b	C ₉ N ₁₂	0.00	1597.7 (5788.1)	1505.2 (5449.9)	1507.5 (5458.3)	1506.0 (5546.0)	1510.0 (5470.5)
3b	C ₆ N ₁₈	0.00	2200.1 (6789.2)	2176.6 (6714.0)	2179.4 (6722.7)	2177.4 (6719.2)	2081.8 (6424.1)
4	C ₂₄ H ₉ N ₁₅ O ₁₂	0.23	1528.3 (2186.3)	– –	– –	1146.1 (1639.5)	1339.0 (1915.5)

Oxygen saturation coefficient: $\alpha = \frac{2O}{4C+H}$

compounds studied. For compounds **1a** and **2a**, the values obtained using reaction schemes were lower than those from G4MP2 by 18 and 8 kJ/mol, respectively. For subsequent comparison and analysis, G4MP2-derived values were used for compounds **1–3a,b**, while for compound **4**, we used the value obtained from the reaction scheme approach.

Previously published literature reports enthalpy of formation calculations for compounds **1a**, **1b**, **2b**, and **3a**, with values very close to those obtained in the present work for the corresponding structures (differences within 1–5 kcal/mol). In particular, Yang [32] reported B3LYP/6-311G(d,p)-level enthalpy of formation values for compounds **1b** and **2b** as 1220.9 and 1602.1 kJ/mol, respectively (literature values have been rounded to one decimal place for consistency with the present results). These values are 52 and 4 kJ/mol higher than those obtained in our work using DFT, and 104 and 97 kJ/mol higher than those obtained using the G4MP2 composite method. The discrepancy between B3LYP-based results may be attributed to our use of a more extended basis set, and possibly to differences in the optimized geometries. The deviation from G4MP2 results falls within the expected error margin of DFT methods for enthalpy of formation calculations. Zeng and co-authors [33] reported a gas-phase enthalpy of formation for compound **3a** of 439.1 kcal/mol (1837.1 kJ/mol), which is approximately 13 kJ/mol lower than the value obtained in this work using the G4MP2 method, and 5 kJ/mol lower than the value calculated using isodesmic reaction schemes. For compound **1a**, the table provided by the authors appears to contain a mix-up in the rows corresponding to the gas-phase enthalpy of formation; the correct value should be 176.5 kcal/mol (738.5 kJ/mol), which is 27 kJ/mol lower than our G4MP2 result and 9 kJ/mol lower than our isodesmic reaction-based value. In their study, the authors also employed reaction schemes, but of a different composition, and the enthalpies of formation for the reactants were calculated using the G3B3 method. Qu and co-authors [34] also reported calculated solid-phase enthalpy of formation values for compounds **1a** and **3a**: 155.3 and 376.1 kcal/mol, respectively (although the article states “kJ/mol”, this appears to be a typographical error). After conversion, these values correspond to 649.8 and

1573.6 kJ/mol, which are 116–276 kJ/mol lower than the gas-phase values obtained in our work. This discrepancy is expected, as the cited literature values correspond to the solid phase and include the contribution of sublimation enthalpy, which the authors estimated using an empirical formula within the framework of the Politzer approach.

The obtained computational results clearly demonstrate the relationship between gas-phase enthalpy of formation and the molecular structure of tris(1,2,4-triazolo)-1,3,5-triazines (**1a**, **2a**, **3a**) and tris(1,2,3-triazolo)benzenes (**1b**, **2b**, **3b**). Most notably, the specific enthalpy of formation of the tetracycles **1a**, **2a**, **3a** and **1b**, **2b**, **3b** follows a consistent increasing trend with respect to the substituent: **1** (–NH₂) < **2** (–CN) < **3** (–N₃). The lowest specific $\Delta H_f^\circ(\text{g})$ values within this series are predicted for the amino derivatives **1a** and **1b** ($\Delta H_f^\circ(\text{g}) \sim 3\text{--}4.5$ MJ/kg), as their structures contain the smallest number of C–N bonds. The enthalpy of formation of the cyano derivatives **2a** and **2b** is significantly higher than that of the corresponding amino compounds due to the presence of three additional C \equiv N triple bonds. In turn, the azido derivatives **3a** and **3b** exhibit the highest enthalpy of formation values in the series ($\Delta H_f^\circ(\text{g}) \sim 6\text{--}7$ MJ/kg), as their structures incorporate the highly endothermic, energy-rich azido groups –N[–]–N⁺ \equiv N, which are saturated with nitrogen–nitrogen multiple bonds [35].

It is evident that compounds **1a**, **2a**, and **3a**, which contain the 1,2,4-triazole ring, exhibit gas-phase enthalpies of formation (both molar and specific) that are 150–350 kJ/mol (550–1450 kJ/kg) lower than those of their isomeric counterparts **1b**, **2b**, and **3b**, respectively, which incorporate 1,2,3-triazole rings with identical substituents. This difference is primarily attributed to the fact that 1,2,3-triazoles, as previously demonstrated by our group [13], provide the highest gas-phase enthalpy of formation ($\Delta H_f^\circ(\text{g})$) values among triazole isomers due to their greater number of endothermic N–N bonds. In addition, the substituents in compounds **1a**, **2a**, and **3a** are attached to the 1,2,4-triazole rings via carbon atoms, whereas in structures **1b**, **2b**, and **3b**, the functional groups are bonded to the 1,2,3-triazole rings through nitrogen atoms, which further contributes to the increase in $\Delta H_f^\circ(\text{g})$.

Tetracycle **4** exhibits a significantly lower specific $\Delta H_f^\circ(\text{g})$ compared to the compounds discussed above, due to the presence of three dinitrophenyl radicals in its structure, which markedly reduce the enthalpy of formation. This structural feature undoubtedly contributes to the compounds remarkable resistance to external thermal and mechanical impact.

2.2. IR Spectra and Frequency Analysis

The IR absorption spectra of the compounds under investigation are presented in Fig. 4.

Most of the intense bands in the IR spectra of compounds **1a** and **1b** are associated with vibrations in the amino groups. Both spectra exhibit bands corresponding to symmetric and asymmetric N–H *stretching* vibrations. In the 1,3,5-triazine-based compound **1a**, these bands are shifted to higher frequencies – 3565 cm^{–1} and 3439 cm^{–1} – compared to 3470 and 3371 cm^{–1} in the benzene-based isomer **1b**. This shift is likely due to the electronic structure of the triazine ring, which exhibits more pronounced electron-accepting properties than benzene. The N–H *scissoring* vibrations appear around 1575 cm^{–1} in the benzene-based compound and are split into two peaks (1588 cm^{–1} and 1537 cm^{–1}) in the triazine-based isomer. In the region around 1626 cm^{–1} in compound **1a**, the N–H *scissoring* vibrations are coupled with C–N *stretching* modes that connect the amino groups to the triazole fragments. *Wagging* deformations of the amino groups are observed at 1118 cm^{–1} in the benzene-based isomer and at 1087 cm^{–1} in the triazine-based analogue. Compound **1a** also exhibits a characteristic *scissoring* mode at

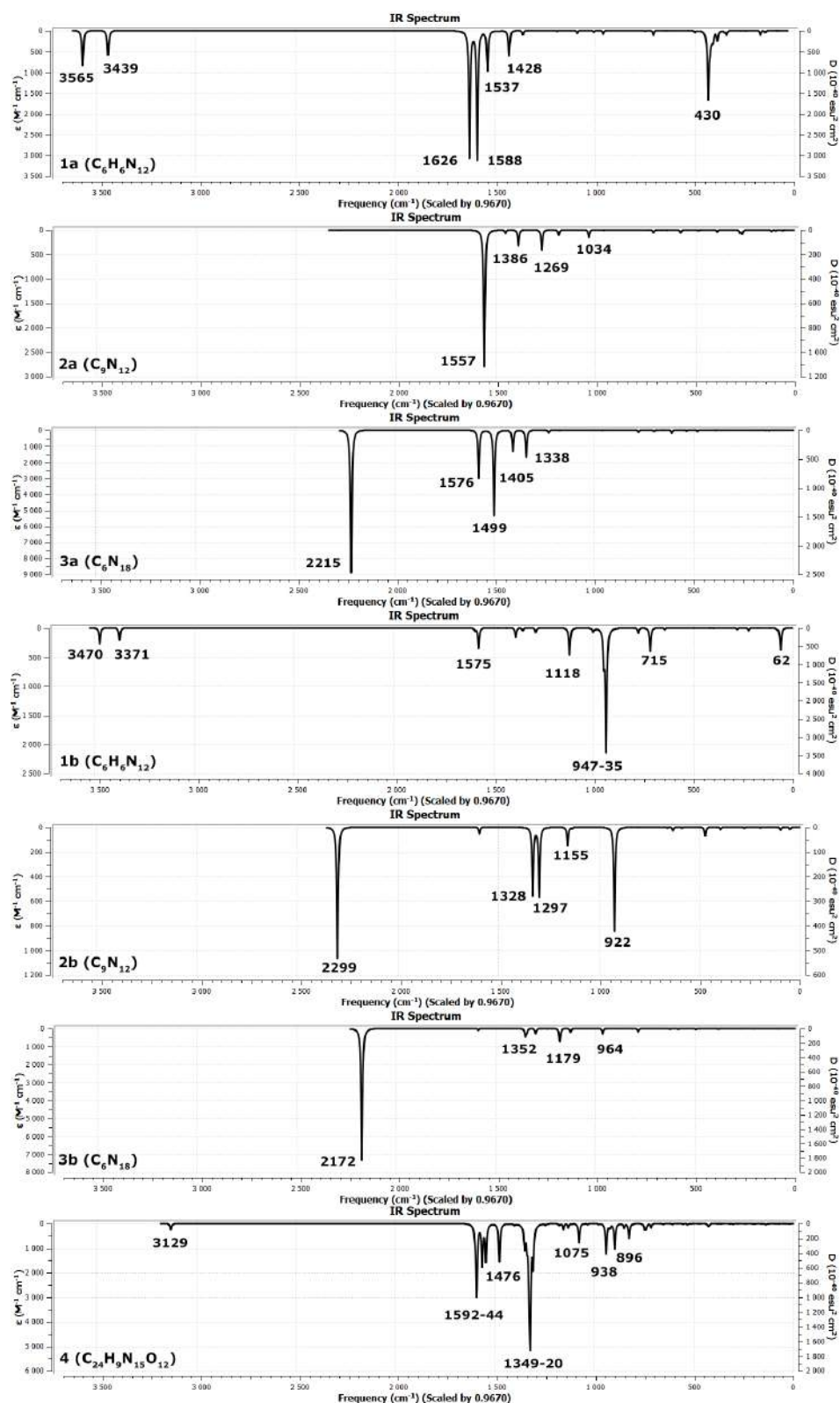


Figure 4. IR absorption spectra of tetracycles **1a,b**, **2a,b**, **3a,b**, and **4** (calculated at the B3LYP/cc-pVTZ level)

1428 cm⁻¹, in which the nitrogen atom of the amino group vibrates in a motion involving both the hydrogen atom and a carbon atom of the triazole ring. In the out-of-plane region, the triazine-based compound exhibits a distinct out-of-plane *wagging* vibration of the NH group at

430 cm^{-1} , whereas the benzene-based isomer displays more diverse out-of-plane *deformations* distributed across the spectrum at 947, 935, and 62 cm^{-1} .

In the IR spectrum of the benzene-based compound **2b**, a distinct band is observed at 2299 cm^{-1} , corresponding to the *stretching* vibrations of the $\text{C}\equiv\text{N}$ bonds in the nitrile groups. In the triazine-based isomer **2a**, similar vibrations are located around 2364 cm^{-1} ; however, their IR intensity is extremely low, resulting in the absence of a visible band in the spectrum. Additional features in the spectrum of compound **2b** include a band at 1297 cm^{-1} , corresponding to $\text{C}-\text{N}$ *stretching* between the triazole and nitrile group, and a peak at 1155 cm^{-1} , which arises from complex *deformation* vibrations involving carbon atoms of the benzene ring and nitrogen atoms of the triazole rings. In the spectrum of the triazine-based isomer **2a**, prominent bands appear at 1269 cm^{-1} , attributed to $\text{C}-\text{N}$ *stretching* within the triazole rings, and at 1034 cm^{-1} , corresponding to $\text{N}-\text{N}$ *stretching* vibrations in the triazole moiety.

Both azido-substituted isomers (**3a** and **3b**) are characterized by *stretching* vibrations of the external $\text{N}-\text{N}$ bonds in the azide groups, observed at 2172 cm^{-1} for the benzene-based compound and at 2215 cm^{-1} for the triazine-based compound. In the spectrum of compound **3b**, a band at 1179 cm^{-1} is also observed, corresponding to the internal $\text{N}-\text{N}$ bond vibrations within the azide group. The spectrum of compound **3a** exhibits the following features: a peak at 1499 cm^{-1} associated with $\text{C}-\text{N}$ *stretching* between the triazole ring and the azide group, and bands at 1405 cm^{-1} and 1338 cm^{-1} , which correspond to combined $\text{C}-\text{N}$ *stretching* vibrations in different fragments (triazine and triazole).

The IR spectrum of compound **4** exhibits several intense absorption bands associated with $\text{C}-\text{H}$ vibrations in the dinitrophenyl rings: 3129–3125 cm^{-1} corresponding to *stretching* vibrations, and 1476 cm^{-1} and 1075 cm^{-1} corresponding to *deformation* modes. The compound also shows characteristic peaks at 1562 cm^{-1} and 1544 cm^{-1} , attributed to asymmetric $\text{N}-\text{O}$ *stretching* vibrations in the nitro groups, and at 823 cm^{-1} , corresponding to *scissoring* deformations in the nitro groups. In addition, prominent intensity peaks are observed in the 1592–1589 cm^{-1} region, associated with $\text{C}-\text{C}$ *stretching* in all benzene rings; at 1349 cm^{-1} , corresponding to $\text{C}-\text{N}$ *stretching* between benzene and the nitro group; at 1337 cm^{-1} and 1326 cm^{-1} , related to $\text{C}-\text{C}$ *stretching* within benzene rings and $\text{C}-\text{N}$ *stretching* in the nitrated fragments; at 1320 cm^{-1} , attributed to combined $\text{C}-\text{N}$ *stretching* vibrations across several parts of the molecule; and at 896 cm^{-1} , corresponding to combined *deformation* vibrations in the peripheral benzene rings.

Most of the benzene-based compounds (**2b**, **3b**, **4**) exhibit characteristic vibrations of the $\text{N}-\text{N}-\text{N}$ fragment within the triazole ring: 922 cm^{-1} for compound **2b**, 964 cm^{-1} for compound **3b**, and 938 cm^{-1} for compound **4**. The spectra of compounds **2b** and **3b** also display similar peaks in the regions of 1328 cm^{-1} and 1352–1344 cm^{-1} , respectively, associated with $\text{C}-\text{C}$ *stretching* vibrations in the benzene rings. The IR spectra of the triazine-based compounds **2a** and **3a** are characterized by strong absorption bands at 1557 and 1576 cm^{-1} , corresponding to $\text{C}-\text{N}$ *stretching* vibrations within the triazine core.

Thus, the IR spectra clearly demonstrate how structural differences – namely, the nature of the substituents and the type of central ring – affect the vibrational behavior of the molecules. All observed bands correspond to vibrations characteristic of the respective functional fragments.

3. Computational Details

Quantum-chemical calculations were carried out using the computing facilities of the Lomonosov Moscow State University Supercomputing Center (project 2312) [36, 37] and the

computational resources of the Federal Research Center of Problems of Chemical Physics and Medicinal Chemistry of the Russian Academy of Sciences. Geometry optimizations for all compounds, as well as calculations using the composite G4MP2 method and its adaptation for the NWChem software package (for all compounds except compound **4**), were performed on the *volta2* partition of the “Lomonosov-2” supercomputer using Intel Xeon Gold 6240 processors (18 cores, 2.60 GHz, 1497.6 GFlop/s) with GPU acceleration (Nvidia Tesla V100, 900-2G500-0010-000, 1246 MHz, 7 TFlop/s).

Geometry optimization using the GAUSSIAN 09 software on the *volta2* partition took approximately 1 hour for compounds **1–3a,b** and 33 hours for compound **4**. G4MP2 calculations for most of the investigated molecules were completed within 11–20 hours; however, the calculation for compound **3a** was not completed due to the 48-hour time limit imposed by the supercomputer. Although the calculation for the structurally similar benzene-based compound **3b** finished successfully in 20 hours, the CCSD(T) step (used to account for electron correlation) proceeded significantly more slowly in case of the triazine derivative, causing the overall computation to exceed the allowed time limit. This may be attributed to the electronic structure of the triazine core combined with azide groups, which could result in poorer convergence or increased complexity in the correlation analysis at the CCSD(T) level. In this case, GAUSSIAN was unable to write a new checkpoint file in time, and upon restart, re-executed the CCSD(T) step from the beginning.

The NWChem software package allows for task distribution across multiple nodes via MPI. For most of the compounds studied in this work, two nodes per task were used, and the calculations took between 2.5 and 6.5 hours. For compound **4**, which has the most complex structure among those considered, eight nodes were employed. However, we encountered technical difficulties during the CCSD(T) step under MPI parallelization, as in our previous studies [18]. These issues may be related to insufficient memory or suboptimal MPI configuration.

DFT calculations using the B3LYP functional with the cc-pVTZ basis set for all compounds were performed using the GAUSSIAN 09 software package on a separate computational resource with the following specifications: Intel(R) Xeon(R) Gold 6140 CPU @2.30 GHz, 259 GB RAM, and 20 TB of disk space. The computations took between 2.5 and 5.5 hours for compounds **1–3a,b**, and 33 hours for compound **4**. The G4MP2 calculation for compound **3a** was also carried out on this system and required approximately 90 hours.

For the enthalpy of formation calculations based on the reaction schemes described above, data obtained from both computational platforms were used. The total calculation time for compounds **1–3a,b** ranged from 3 to 8 hours, and 121 hours for compound **4**. This total runtime is significantly lower than that of G4MP2 calculations on the supercomputer.

Conclusions

The conducted study enabled an assessment of the efficiency of various quantum-chemical approaches for calculating the enthalpy of formation of high-energy tetracyclic compounds. The G4MP2 method was used as a benchmark to evaluate the accuracy of less resource-intensive methods. It was shown that density functional theory (DFT) using the B3LYP functional with the cc-pVTZ basis set significantly reduces the overestimation of thermochemical parameters typical of simpler basis sets and, in some cases, yields enthalpy of formation values close to those obtained with G4MP2. Calculations based on isodesmic reaction schemes demonstrated strong agreement with G4MP2 results while requiring significantly lower computational resources. The

adapted implementation of the G4MP2 scheme in the NWChem software package also showed good agreement with the original implementation in GAUSSIAN 09, while offering greater flexibility for parallel computing on high-performance computing platforms.

At the same time, it should be noted that the G4MP2 calculation for the most complex structure under investigation could not be completed in either GAUSSIAN 09 or NWChem due to runtime limitations and the complexity of the CCSD(T) correlation step. This highlights the limitations of composite methods when applied to large molecules and underscores the relevance of using less computationally demanding approaches such as reaction schemes and DFT methods with carefully selected basis sets.

The analysis of the relationship between molecular structure and thermochemical characteristics confirmed the expected increase in enthalpy of formation along the substituent series $-\text{NH}_2 < -\text{CN} < -\text{N}_3$, as well as the advantage of tetracycles containing 1,2,3-triazole rings over their 1,2,4-triazole-based isomers. IR spectrum calculations showed that the characteristics of the central ring and the nature of functional groups significantly influence the vibrational profile of the compounds.

The obtained results confirm the feasibility of using reaction schemes and adapted high-accuracy methods in the modeling of promising energetic molecules and will serve as a foundation for further evaluation of their energy potential and possible applications.

Acknowledgements

Calculations carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University [36], were supported by the Russian Science Foundation (project No. 23-71-00005). V.M. Volokhov and E.S. Amosova performed research in accordance with the State Task, state registration No. 124013100856-9. V.V. Parakhin was engaged in the formulation of a scientific problem, literature review, analysis of the results, writing and editing the article. D.B. Lempert performed work in accordance with the State Task, state registration No. 124020100045-5. V.V. Voevodin participated in the quantum-chemical research and the analysis of the results.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Rice, B.M., Pai, S.V., Hare, J.: Predicting heats of formation of energetic materials using quantum mechanical calculations. *Combust. Flame* 118, 445–458 (1999). [https://doi.org/10.1016/S0010-2180\(99\)00008-5](https://doi.org/10.1016/S0010-2180(99)00008-5)
2. Byrd, E.F., Rice, B.M.: Improved prediction of heats of formation of energetic materials using quantum mechanical calculations. *J. Phys. Chem. A* 110, 1005–1013 (2006). <https://doi.org/10.1021/jp0536192>
3. Muthurajan, H., Sivabalan, R., Talawar, M.B., *et al.*: Prediction of heat of formation and related parameters of high energy materials. *J. Hazard. Mater.* 133, 30–45 (2006). <https://doi.org/10.1016/j.jhazmat.2005.10.009>

4. Zhong, L., Liu, D., Hu, M., *et al.*: First-principles calculations of solid-phase enthalpy of formation of energetic materials. *Commun. Chem.* 8, 1–7 (2025). <https://doi.org/10.1038/s42004-025-01544-9>
5. Larin, A.A., Muravyev, N.V., Pivkina, A.N., *et al.*: Assembly of tetrazolylfuroxan organic salts: Multipurpose green energetic materials with high enthalpies of formation and excellent detonation performance. *Chem. Eur. J.* 25, 4225–4233 (2019). <https://doi.org/10.1002/chem.201806378>
6. Luk'yanov, O.A., Parakhin, V.V., Shlykova, N.I., *et al.*: Energetic N-azidomethyl derivatives of polynitro hexaazaisowurtzitanes series: CL-20 analogues having the highest enthalpy. *New J. Chem.* 44, 8357–8365 (2020). <https://doi.org/10.1039/D0NJ01453b>
7. Gao, H., Zhang, Q., Sreeve, J.M.: Fused heterocycle-based energetic materials (2012–2019). *J. Mater. Chem. A* 8, 4193–4216 (2020). <https://doi.org/10.1039/c9ta12704f>
8. Wen, L., Wang, Y., Liu, Y.: Data-Driven Combinatorial Design of Highly Energetic Materials. *Acc. Mater. Res.* 6, 64–76 (2025). <https://doi.org/10.1021/accountsmr.4c00230>
9. Kuznetsova, A.N., Leonov, N.E., Anikin, O.V., *et al.*: Parent 1,4-dihydro-[1,2,3]triazolo[4,5-d][1,2,3]triazole and its derivatives as precursors for the design of promising high energy density materials. *New J. Chem.* 49, 311–320 (2025). <https://doi.org/10.1039/D4NJ04427D>
10. Feng, S., Li, Y., Lai, Q., *et al.*: A strategy for stabilizing of N₈ type energetic materials by introducing 4-nitro-1,2,3-triazole scaffolds. *Chem. Eng. J.* 430, 133181 (2022). <https://doi.org/10.1016/j.cej.2021.133181>
11. Volokhov, V.M., Amosova, E.S., Volokhov, A.V., *et al.*: Quantum-chemical calculations of physicochemical properties of high enthalpy 1,2,3,4- and 1,2,4,5-tetrazines annelated with polynitroderivatives of pyrrole and pyrazole. Comparison of different calculation methods. *Comput. Theor. Chem.* 1209, 113608 (2022). <https://doi.org/10.1016/j.comptc.2022.113608>
12. Volokhov, V. M., Parakhin, V. V., Amosova, E. S., *et al.*: Quantum-Chemical Study of Gas-Phase 5/6/5 Tricyclic Tetrazine Derivatives. *Supercomputing Frontiers and Innovations* 10(3), 61–72 (2023). <https://doi.org/10.14529/jsfi230306>
13. Volokhov, V.M., Parakhin, V.V., Amosova, E.S., *et al.*: Quantum-chemical calculations of the enthalpy of formation of 5/6/5 tricyclic tetrazine derivatives annelated with nitrotriazoles. *Russ. J. Phys. Chem. B*, 18(1), 28–36 (2024). <https://doi.org/10.1134/S1990793124010196>
14. Volokhov, V.M., Amosova, E.S., Parakhin, V.V., *et al.*: Quantum-Chemical Study of Some Tris(pyrrolo)benzenes and Tris(pyrrolo)-1,3,5-triazines. In: Voevodin, V., Sobolev, S., Yakobovskiy, M., Shagaliev, R. (eds) *Supercomputing. RuSCDays 2023. Lecture Notes in Computer Science*, vol. 14388, pp. 177–189. Springer, Cham. (2023) https://doi.org/10.1007/978-3-031-49432-1_14
15. Volokhov, V., Amosova, E., Parakhin, V., *et al.*: Quantum-Chemical Simulation of Some Triimidazolobenzenes and Triimidazolo-1,3,5-Triazines. In: Sokolinsky, L., Zymbler, M., Vo-

- evodin, V., Dongarra, J. (eds) Parallel Computational Technologies. PCT 2024. Communications in Computer and Information Science, vol. 2241, pp. 292–303. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-73372-7_21
16. Volokhov, V.M., Parakhin, V.V., Amosova, E.S., *et al.*: Quantum-Chemical Study of Some Trispyrazolobenzenes and Trispyrazolo-1,3,5-triazines. Supercomput. Front. Innov. 11(3), 64–73 (2024). <https://doi.org/10.14529/jsfi240304>
17. Volokhov, V., Amosova, E., Parakhin, V., *et al.*: Quantum-chemical study of unsubstituted tris(triazolo)benzenes and 1,3,5-triazines. In: Parallel Computational Technologies. PCT 2025. Communications in Computer and Information Science (in print).
18. Volokhov, V.M., Parakhin, V.V., Amosova, E.S., *et al.*: Quantum-Chemical Study of High Energy Derivatives of Tris(triazolo)benzenes. Lobachevskii J. Math. 46(8), 3883–3894 (2025). <https://doi.org/10.1134/S1995080225610100>
19. Samsonov, V.A., Volodarskii, L.B., Korolev, V.L., Khisamutdinov, G.K.: Synthesis of benzotriazole. 4-nitrobenzo[1,2-d:3,4-d']bistriazole and 4,4'-dicarboxy-5,5'-1H-1,2,3-triazole. Chem. Heterocycl. Compd. 29, 1169–1171 (1993). <https://doi.org/10.1007/BF00538063>
20. Thottempudi, V., Forohor, F., Parrish, D.A., Shreeve, J.M.: Tris(triazolo)benzene and its derivatives: high-density energetic materials. Angew. Chem. Int. Ed. 51(39), 9881–9885 (2012). <https://doi.org/10.1002/anie.201205134>
21. Frisch, M.J., Trucks, G.W., Schlegel, H.B., *et al.*: Gaussian 09, Revision B.01. Gaussian, Inc., Wallingford CT (2010).
22. Valiev, M., Bylaska, E. J., Govind, N., *et al.*: NWChem: a comprehensive and scalable open-source solution for large scale molecular simulations. Comput. Phys. Commun. 181, 1477 (2010). <https://doi.org/10.1016/j.cpc.2010.04.018>
23. Becke, A.D.: Densityfunctional thermochemistry. III. The role of exact exchange. J. Chem. Phys. 98(4), 5648–5652 (1993). <https://doi.org/10.1063/1.464913>
24. Johnson, B.J., Gill, P.M.W., Pople, J.A.: The performance of a family of density functional methods. J. Chem. Phys. 98(4), 5612–5626 (1993). <https://doi.org/10.1063/1.464906>
25. Dunning, Th.H. Jr.: Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen. J. Chem. Phys. 90(2), 1007–1023 (1989). <https://doi.org/10.1063/1.456153>
26. Curtiss, L.A., Redfern, P.C., Raghavachari, K.: Gaussian-4 theory using reduced order perturbation theory. J. Chem. Phys. 127, 124105 (2007). <https://doi.org/10.1063/1.2770701>
27. Curtiss, L.A., Redfern, P.C., Raghavachari, K.: Gn theory. Comput. Mol. Sci. 1, 810–825 (2011). <https://doi.org/10.1002/wcms.59>
28. Curtiss, L.A., Redfern, P.C., Raghavachari, K.: Gaussian-4 theory. J. Chem. Phys. 126, 084108 (2007). <https://doi.org/10.1063/1.2436888>

29. CCCBDB Vibrational Frequency Scaling Factors. <https://cccbdb.nist.gov/vsfx.asp>, accessed: 2025-04-10
30. NIST-JANAF Thermochemical Tables. <https://janaf.nist.gov/>, accessed: 2025-04-10
31. Chai, J.-D., Head-Gordon, M.: Long-range corrected hybrid density functionals with damped atom-atom dispersion corrections. *Phys. Chem. Chem. Phys.* 10, 6615–6620 (2008). <https://doi.org/10.1039/b810189b>
32. Yang, J.: Theoretical studies on the structures, densities, detonation properties and thermal stability of tris(triazolo)benzene and its derivatives. *Polycycl. Aromat. Compd.* 35(5), 387–400 (2015). <https://doi.org/10.1080/10406638.2014.918888>
33. Zeng, Q., Qu, Y., Li, J., Huang, H.: Theoretical studies on the derivatives of tris([1,2,4]triazolo)[4,3-a:4',3'-c:4'',3''-e][1,3,5]triazine as high energetic compounds. *RSC Adv.* 6, 5419 (2016). <https://doi.org/10.1039/c5ra22524h>
34. Qu, Y., Zeng, Q., Wang, J., *et al.*: Synthesis and properties for benzotriazole nitrogen oxides (BTzO) and tris[1,2,4]triazolo[1,3,5]triazine derivatives. *Int. J. Mater. Sci. Appl.* 7(2), 49–57 (2017). <https://doi.org/10.11648/j.ijmsa.20180702.13>
35. Klapötke, T.M., Krumm, B., Martin, F.A., Stierstorfer, J.: New azidotetrazoles: structurally interesting and extremely sensitive. *Chem. Asian J.* 7, 214–224 (2012). <https://doi.org/10.1002/asia.201100632>
36. Voevodin, V.I.V., Antonov, A.S., Nikitenko, D.A., *et al.*: Supercomputer Lomonosov-2: largescale, deep monitoring and fine analytics for the user community. *Supercomput. Front. Innov.* 6(2), 4–11 (2019). <https://doi.org/10.14529/jsfi190201>
37. Nikitenko, D.A., Voevodin, V.I.V., Zhumatiy, S.A.: Deep analysis of job state statistics on Lomonosov-2 supercomputer. *Supercomput. Front. Innov.* 5(2), 4–10 (2019). <https://doi.org/10.14529/jsfi180201>

Dynamic Content-Oriented Indexing and Replication for High-Performance Storage and Analysis of Big Data in the IPFS Network

Maxim V. Shevarev¹, Stanislav V. Suvorov²

© The Authors 2025. This paper is published with open access at SuperFri.org

This paper presents an architecture for dynamic, content-oriented indexing and adaptive replication that enables high-performance storage and analysis of big data on IPFS. We first outline key gaps of vanilla IPFS for analytics – no global content search, non-guaranteed persistence without coordinated pinning, static replication, and highly variable retrieval latency – and address them with two components: (1) a two-tier distributed index (per-attribute/keyword inverted lists as IPFS objects plus a lightweight catalog that maps search keys to index CIDs via DHT/IPNS or CRDT-based dissemination); and (2) an adaptive replication service that aggregates access telemetry and adjusts replica counts and placement using hysteresis thresholds and topology-aware selection. The contribution is a theoretical proposal and architectural blueprint; no prototype or experimental results are reported here. We discuss integration with analytical engines through two paths: a pragmatic FUSE mount that exposes IPFS content as a local filesystem to Spark/Flink, and prospective native connectors that parallelize block reads over the IPFS API. For tabular datasets, dataset metadata (schema, partitioning, file CIDs) is maintained in IPFS to support versioning and reproducibility. A plan for comparative evaluation versus HDFS, Ceph, and S3 (e.g., TPC-DS and subsets of Common Crawl) is outlined. Expected benefits are faster content discovery, higher throughput under skew and multi-tenant load, and improved resilience, with modest index/coordination overheads. The approach combines the openness of a decentralized P2P substrate with the manageability required by enterprise-scale analytics.

Keywords: dynamic indexing, IPFS, big data, replication, content-oriented indexing, high-performance storage.

Introduction

Modern Big Data storage solutions such as HDFS, cloud storage, and others, have a number of limitations in scalability, flexibility, and decentralization. The IPFS network offers content-addressable, decentralized storage capable of eliminating single points of failure and increasing data availability. However, IPFS in its basic implementation does not contain the means for efficient content indexing and dynamic data replication necessary for typical Big Data tasks. As a result, the problem of efficient storage and high-performance parallel access to big data in the IPFS environment has not been solved. This study aims to solve this problem by developing a new architecture that combines the capabilities of IPFS with additional indexing and replication mechanisms, which will allow IPFS to be used for Big Data tasks, providing both scientific novelty and practical value. The novelty includes the use of a content-based approach to metadata organization and adaptive replication management based on access profiles – such combinations have not previously been implemented in big data storage systems. The practical significance of the work is due to the potential reduction in the load on centralized nodes, increased speed of access to popular data and system resilience to failures due to the decentralization of storage.

The article is organized as follows. Section 1 surveys the current state of the field and limitations of existing storage systems. Section 2 reviews the architecture of IPFS and its applicability to Big Data. Section 3 presents the proposed approach, including content-oriented indexing,

¹Moscow Polytechnic University, Moscow, Russia. E-mail: shevarev.max@mail.ru

²Candidate of Economics, Head of the Department, Professor, Moscow Polytechnic University, Moscow, Russia

adaptive replication, and integration with analytics frameworks. Section 4 introduces the reference systems for comparison. Section 5 describes the evaluation methodology, metrics, and scenarios. Section 6 outlines the test datasets. The Conclusion summarizes the study and points directions for further work.

1. Analysis of the Current State of the Field

Storage and processing of big data today. Big data is characterized by volume (petabytes), velocity of accumulation, and variety of structures. The traditional platform for Big Data is the Hadoop ecosystem, where HDFS (Hadoop Distributed File System) serves as a distributed storage system, and MapReduce or Apache Spark are used for data processing. HDFS splits files into blocks and replicates each block, typically three times on different cluster nodes [4, 8], to ensure fault tolerance. This strategy provides high throughput for sequential reading and “divide and conquer” writing, close to the performance of local disks on each node. In addition to HDFS, large infrastructures use distributed object storages such as Amazon S3, Ceph, as well as HPC-level file systems (e.g., Lustre, GlusterFS). Cloud storages (S3, Google Cloud Storage) have become popular due to their elasticity and integration with analytical engines (Presto, Hive, etc.), while Ceph offers unified object, block, and file storage for private clouds.

1.1. Technologies Used and Their Limitations

Each of the existing solutions has its limitations. HDFS relies on a central metadata node (NameNode) to store information about the file structure. This means potential non-scalability of metadata and the need for High Availability mechanisms for the NameNode; otherwise, a NameNode failure paralyzes the cluster. In addition, HDFS is designed for a “write once, read many” model: after writing, a file cannot be arbitrarily modified, only append and truncate operations are supported. This simplifies integrity maintenance but makes HDFS unsuitable for applications with frequent updates or random writes; such workloads are better supported by Ceph or traditional DBMSs. Replication in HDFS is fixed: the replication factor is the same for all blocks (by default, 3), without considering the actual popularity of the data. During peak demand for certain “hot” data, three copies may be insufficient (causing overload of nodes storing these blocks), while less popular data is stored in redundant three copies unnecessarily. The lack of a dynamic replication mechanism is a known drawback that research groups are trying to address. The literature proposes approaches where the number of HDFS block replicas changes depending on file popularity [4]. For example, X. Cao et al. describe an algorithm for predicting file “hotness” using a Markov model and adapting the number of copies proportionally to the forecast, which significantly increases cluster efficiency during surges in requests for the same data [4]. However, such solutions have not yet been implemented in mainstream HDFS versions and require external management.

Other systems also have limitations. Ceph is a fully distributed storage system with no single point of failure (metadata is distributed using the CRUSH algorithm), supports replication and erasure coding, and provides more flexible space utilization [9]. However, this flexibility comes at a cost: deploying and maintaining a Ceph cluster requires high expertise, and configuration is more complex compared to HDFS.

Amazon S3 and similar cloud storages provide high reliability through replication at the data center level, but when working with big data, other bottlenecks appear: access latency

over the Internet, as well as charges for each request and data retrieval. In addition, S3 objects have eventual consistency for write operations, which complicates scenarios with frequent data updates. GlusterFS and other distributed file systems (Lustre, etc.) are often used in HPC and can provide high throughput, but their efficiency in Big Data scenarios also depend on the workload, and scaling to thousands of nodes is associated with challenges.

Thus, modern solutions are either centralized or require complex support, and do not dynamically adapt to changing workload profiles. This opens up opportunities for new architectures that combine decentralization and intelligent data management algorithms.

2. Architecture of IPFS and Its Applicability to Big Data

The InterPlanetary File System (IPFS) is a protocol and a peer-to-peer network for distributed data storage and delivery. IPFS uses a content-addressable model: each file (and its fragments) is assigned a Content Identifier (CID) – a cryptographic hash of its content. The network layer of IPFS is a distributed hash table (DHT) based on the Kademlia algorithm, where nodes store records about which CIDs they can provide [6]. To retrieve a file, a node queries the DHT to find which peers have the required CID and establishes direct connections to download fragments (blocks). A simplified P2P network with DHT is shown in Fig. 1.

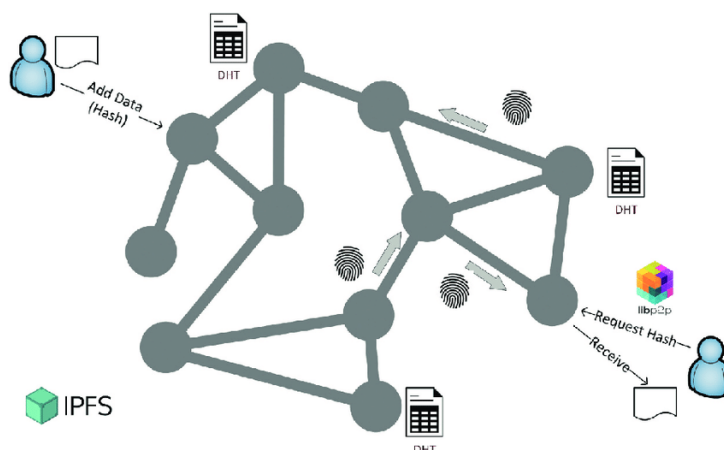


Figure 1. Simplified P2P network with DHT

Block transfer is performed using the IPFS-specific Bitswap protocol, similar to BitTorrent. Bitswap allows nodes to request missing blocks from multiple peers in parallel, theoretically scaling throughput: popular content can spread like a “swarm” in a P2P network, offloading individual nodes. IPFS also uses a Merkle DAG to organize files: each file is split into fixed-size blocks, and the hashes of the blocks form a tree structure (Merkle graph), enabling integrity verification and assembly of large files from parts.

The advantages of IPFS for big data include the absence of a central node: content is automatically duplicated on as many nodes as have downloaded it, and unified addressing by hash eliminates storage duplication – identical files are automatically the same, with no double storage. Content addressing also provides built-in data integrity verification, which is critical for large volumes where recalculating checksums is expensive. In addition, IPFS supports content versioning: a new file version receives a new CID, while the old one remains accessible by its previous hash, solving the lack of versioning typical for classic replicated storages [1]. This is

important for reproducibility in big data experiments – IPFS can store multiple versions of a dataset in parallel, guaranteeing their immutability.

However, in practice, IPFS has serious limitations that hinder its direct application to Big Data. First, there is no guaranteed storage persistence: if data is not pinned on nodes, nodes may free up space and delete content, making it unavailable over time. IPFS lacks a built-in automatic replication mechanism: the network relies on either the user to ensure pinning on multiple nodes or other participants to download the data and thus automatically become replicas. If a file is not in demand, there is a risk that the only copy will disappear (the owner node disconnects or clears the cache). This is unacceptable for most Big Data tasks, where data must be stored reliably regardless of short-term interest. The solution in the IPFS ecosystem is IPFS Cluster: an add-on that coordinates a group of IPFS nodes and automatically replicates pinnable content among them.

IPFS Cluster maintains a global pinset and distributes objects across cluster nodes according to a user-defined replication factor, using Raft consensus for consistency between nodes [7]. If one of the cluster nodes fails, the Cluster can automatically re-replicate “orphaned” content to other nodes, maintaining the required number of copies [7]. Thus, IPFS Cluster provides fault tolerance, but replication in the Cluster is static – it is set by the number of copies, which does not change dynamically with the load. In addition, the Cluster assumes trust between nodes and is usually deployed as a private network (e.g., within a single organization), not leveraging the full global potential of the IPFS network. The architecture of an IPFS Cluster is shown in Fig. 2.

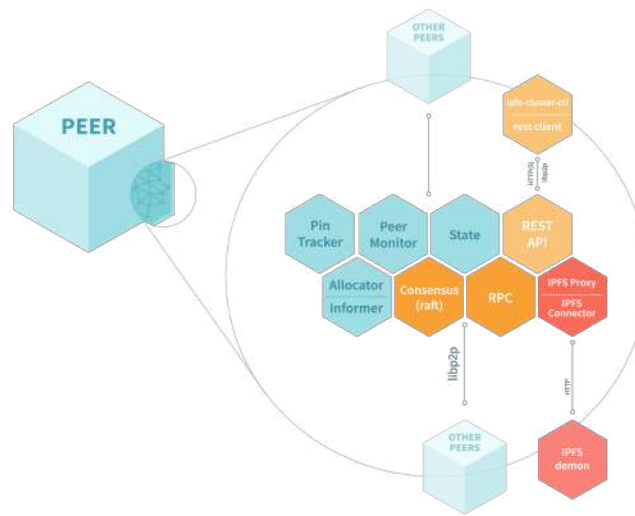


Figure 2. Architecture of an IPFS Cluster

The second problem is data search and indexing. In IPFS, a file can be efficiently retrieved only if its CID is known. Out-of-the-box search by name, keywords, or content is not supported. The global DHT provides hash-based routing but does not solve the problem of obtaining the hashes for the required data. In traditional big data systems, metadata (file names, table schemas, etc.) is managed centrally (e.g., HDFS NameNode stores the file directory, Hive Metastore stores table schemas and part locations on HDFS). In IPFS, there is no unified catalog of all content – this is a deliberate architectural decision (a network without a centralized index),

but it complicates the use of IPFS in analytical tasks where, for example, it is necessary to find all files of a dataset by name pattern or perform a selection by attribute value.

Currently, there are experiments on building decentralized search systems on top of IPFS. One approach is to use two-level indexing with distributed index updates. For example, Ling Cao and Yue Li (2022) proposed a system where each node, when adding a file to IPFS, automatically extracts keywords and updates a local inverted index – a list of words and documents [3]. For each keyword, a separate index file is created – the first level – storing a list of CIDs of files containing this word. These index files themselves are placed in IPFS, receiving their own CIDs. Then, at the second level, a distributed index is built: an association “keyword \rightarrow CID of the index file” [10]. This second-level index can be stored either in a distributed hash table or updates can be broadcast via the IPFS PubSub mechanism, using CRDTs to ensure consistency between nodes [10]. As a result, searching by keyword reduces to finding the corresponding index file via DHT and then loading the list of documents from IPFS. Experiments show that this approach can provide content search, but as the number of documents grows, the volume of index data also increases, requiring optimization and caching. Nevertheless, the work of Ling Cao demonstrates the fundamental feasibility of content-oriented indexing in IPFS and serves as a starting point for developing a custom indexing system in this study.

The third problem is data transfer performance. Despite the potential of P2P distribution, in reality, delays in IPFS can be higher than with direct access to high-speed cluster storage. Data requests in IPFS include latency for peer discovery via DHT (hundreds of milliseconds), then establishing multiple connections and exchanging blocks. If source nodes are geographically distributed or have limited bandwidth, downloading a large file may be slower than in the traditional “high-speed cluster” model. IPFS performance is highly variable: download speed can be inconsistent, especially for large files if slow or distant nodes are involved. On the other hand, with proper node organization, distributed caching can outperform centralized schemes in multi-node environments, but it requires protocol-level optimizations and careful tuning. For Big Data, it is likely necessary to combine the IPFS approach with data pre-placement on compute nodes (data locality) to reduce inter-node traffic. The next section discusses how IPFS can be integrated with frameworks like Spark to run computations close to the data or move data closer to computations.

In summary, current big data solutions suffer from centralization and static architectures, while IPFS offers the prospect of decentralized storage but requires mechanisms for indexing, reliable replication, and performance assurance. The following section formulates the proposed approach, taking these observations into account.

3. Proposed Approach: Content-Oriented Indexing and Adaptive Replication in IPFS

General architecture of the solution. The new system assumes the use of IPFS as the basic data storage layer (transport and distributed block storage), on top of which two key components operate: the first one is a content-oriented data indexing subsystem, and the second one is an adaptive replication module based on access profiles. In addition, an integration layer with analytics tools (Spark, Flink, etc.) is provided, ensuring seamless access for analytical jobs to data in IPFS via standard interfaces. These components are described below.

3.1. Content-Oriented Indexing over IPFS

The goal of this subsystem is to enable fast search and addressing of required data by content, metadata, or logical names, rather than by a hash unknown to users. The approach is based on content-centric indexing ideas. Importantly, indexing *does not imply centralization*: index artifacts are data structures (e.g., inverted lists, manifests) that are themselves stored and versioned in a distributed manner (e.g., in IPFS), and can be published/replicated without introducing a single authoritative catalog. A coordinating component may orchestrate publication for practicality, but the storage and retrieval model remains decentralized.

Each file in a large dataset, when added to the system, is accompanied by the generation of metadata for the index. Metadata may include: file name or logical identifier, size, data type/schema (e.g., CSV, Parquet – then the table schema), file hash (CID), and possibly keywords or content statistics. These metadata are registered in a distributed index. A two-level scheme is proposed, inspired by the work of Ling Cao (described above): at the first level, specialized index structures are formed, for example, for each attribute or word – a list of files containing the given attribute/word. At the second level, a global index catalog is maintained, linking the search key to the location of the corresponding index block.

A possible implementation is as follows. Let us suppose our dataset consists of many files, each described by a set of attributes (e.g., date, topic, source). For each attribute (or a combination like “field name=value”), a separate index document is created – a file containing a list of CIDs of all files matching this attribute/value. This index document itself is placed in IPFS and receives a CID_i . Then, at the second level, a mapping table “attribute $\rightarrow CID_i$ ” is formed. This table can be stored in several ways:

- as a separate index file (a large JSON or a database like IPFS-Lite) with records for all attributes;
- distributed among nodes responsible for different attributes, using DHT/IPNS;
- duplicated on all nodes via CRDT (less scalable, but simple).

The most suitable is the DHT approach: we use the existing IPFS DHT, but publish attribute hashes as keys instead of content hashes. The IPFS DHT allows storing a small data block (usually up to 1 KB) in a record. Therefore, for each attribute (e.g., a keyword), it is possible to publish in the DHT a record: key = hash(attribute name), value = CID_i of the index file. Thus, any node, knowing the attribute, can use the DHT to find the CID of the index file and then download the index itself from IPFS.

To update indexes when new data is added, provider publication is used: when a new file is added, the initiating node either updates the corresponding index file (adds the new CID) and republishes it in IPFS, or creates a new index fragment. To avoid race conditions and ensure consistency during simultaneous additions, a CRDT model is applied: for example, each node maintains a local copy of the index file for each relevant key and exchanges delta updates via PubSub. Centralized coordination via a dedicated “catalog node” is also possible, but this introduces a single point of failure, which is undesirable. In this work, a basic variant is assumed: one node (the index coordinator) collects updates and periodically publishes new versions of index files. This compromise simplifies the initial implementation, and the coordinator’s fault tolerance can be ensured by running its copies on different nodes, with leader election, for example, via Raft. This indexing component is content-oriented, as the key data identifier is not its location, but either a semantic feature (keyword, attribute) or a content hash (CID). A user

or program can request data by semantic criteria, the system will find the required CIDs, and then IPFS will deliver the content.

It is expected that this approach will eliminate one of the obstacles to using IPFS for big data – the problem of searching and organizing large numbers of files. It certainly introduces overhead (storage of index structures, search time), but these costs are justified by the scale of the data: without an index, working with thousands of files in IPFS is impossible within a reasonable time.

3.1.1. Data model and supported data types

The system targets data that is either immutable or versioned and can be addressed content-wise. The primary supported types are:

- **Structured tabular datasets** (e.g., Parquet, ORC, CSV), accompanied by a schema and optional partitioning metadata. A dataset is represented by a manifest (JSON/protobuf) that lists file CIDs and logical partitions.
- **Large binary objects** (e.g., model checkpoints, archives, images, WARC segments) that are naturally chunked by IPFS into fixed-size blocks and referenced by a root CID.
- **Directory-like collections** of homogeneous files referenced from a root directory CID.

The system does not target arbitrary in-place mutable files. Updates are performed by publishing a new version (new CID/root) while previous versions remain accessible. For many small files, ingestion tools can optionally pack them into container objects (e.g., CAR-like bundles) to reduce per-file indexing overhead and improve read parallelism.

3.2. Adaptive Replication Algorithm Based on Access Profile

The second pillar of the proposed solution is dynamic management of the number and placement of data copies in the IPFS network based on access analysis. The idea is as follows: in Big Data workloads, access patterns are often highly skewed – a small portion of the data (hot data) is requested very frequently (e.g., recent logs, active database partitions), while “cold” archival data is accessed rarely. In traditional systems (HDFS, Ceph), the replication factor is usually fixed (three copies), and in object storages – two or three copies plus occasional caching via CDN. It is proposed to adaptively increase the number of replicas for hot objects to improve throughput and fault tolerance, and decrease the number of replicas for cold data to save space. This mechanism is similar to automatic caching, but differs in that the copies are full-fledged and are considered in task planning (unlike random cache, which the system may not be aware of).

Specifically, the following algorithm is proposed. The system monitors data requests: each node, when serving a request for a specific CID (block or file), signals the replication module about the access event. The module aggregates statistics for each large object (e.g., file or logical data partition). Periodically, at intervals of Δt (e.g., one hour), an estimate of popularity $P(obj)$ is calculated for each object over the recent time window. Metrics such as request frequency, number of unique nodes requesting the object, and volume of data transferred are used. Based on $P(obj)$, a decision is made to change replication:

- a) If $P(obj)$ exceeds the upper threshold T_{hot} (the object is clearly hot) and the current number of replicas is less than some maximum R_{max} , the system initiates the creation of additional copies. How to create a copy in IPFS? In the global IPFS network, it is sufficient

for another node to download this CID and pin it locally. Our module can send a command to specific nodes to perform pinning. Node selection for replication can take topology into account – for example, aiming to place copies in different data centers for resilience, or closer to expected consumers (if a particular analytics group frequently requests the data, replicate to their site). The selection algorithm is a separate task; in the simplest case, N nodes can be randomly chosen from the participants, excluding those already storing a copy.

- b) If $P(obj)$ drops below the lower threshold T_{cold} and there are more than the minimum required R_{min} copies (e.g., one or two), some copies can be removed (unpin). Removal should be done cautiously: it may be advisable to reduce gradually and always keep at least one main copy.
- c) For objects of medium popularity (between thresholds), leave as is, without changes.

Such a hysteresis with thresholds prevents oscillation in the number of replicas due to load fluctuations. An important aspect is that the access profile may change over time (e.g., data ages and becomes less used), so the algorithm should account for popularity decay. The literature describes approaches where popularity is predicted using machine learning or time series methods [4], but, in this study, a heuristic method is sufficient: a sliding average of requests over a window, with exponential decay of past counters.

Adaptive replication requires coordination: if each node decides independently which objects to replicate, conflicts may arise. Therefore, a centralized replication coordinator is introduced, which may be combined with the indexing coordinator or be separate. This coordinator collects the global popularity picture, for example, via periodic node reports or by subscribing to PubSub events indicating “interest in CID.” It then calculates new replication levels and sends commands to nodes: “create a copy of CID X” or “it is safe to remove a copy of CID Y.” Upon receiving such a command, a node either executes `ipfs pin add <CID>` (downloading the content) or `ipfs pin rm <CID>` (allowing the garbage collector to remove the content if needed). For reliability, the IPFS Cluster’s API can be used to manage the pin-set on nodes, but this brings us closer to the IPFS Cluster architecture. The difference is that IPFS Cluster requires a fixed replication factor, while we change it dynamically. In principle, it is possible to build an additional layer on top of IPFS Cluster: let the Cluster assume that each pin has `replication_factor`= ∞ (i.e., any node that wants to can pin), and we decide which nodes actually receive the pin command.

It is expected that adaptive replication will significantly improve throughput during mass simultaneous access to the same data, as the load will be distributed across more nodes (CDN effect). In addition, resilience increases: even if several nodes fail, a hot object likely had many copies and remains available. On the other hand, the downside is increased disk space usage for popular data. However, for Big Data, CPU/Memory or network are often the bottlenecks, while disks are relatively cheap; moreover, as data cools, excess copies will be removed, freeing up space.

3.2.1. Coordination model: centralized coordinator and path to decentralization

In the prototype, coordination (index publication and replication control) is centralized for pragmatic reasons: (i) predictable convergence and simpler reasoning about consistency; (ii) lower operational complexity and clearer auditability in multi-tenant settings; (iii) ease of enforcing policy constraints (quotas, placement rules) and reproducing experiments. The coordinator runs in a highly available configuration (leader election, write-ahead log, periodic

snapshots) so that the *control plane* has no single point of failure in practice, while the *data plane* remains decentralized.

Limitations of this approach include tighter bounds on control-plane throughput and potential coupling to a trust domain. Our roadmap to decentralization foresees: (a) CRDT-backed index state with gossip-based dissemination over PubSub; (b) leaderless aggregation of access statistics using sketches and windowed counters; (c) conflict resolution via monotonic policies and attribute-level last-writer-wins; (d) eventual-consistency guarantees with bounded staleness for popularity signals. This evolution preserves the benefits of decentralized operation while maintaining verifiability and policy compliance required in enterprise environments.

3.2.2. Local block placement and device selection

To avoid I/O bottlenecks on a single physical drive, each node exposes a set of storage devices (mount points) with capacity and performance descriptors. For every block/object identified by a CID, the node selects a target device using capacity-aware rendezvous hashing (a.k.a. highest-random-weight), with per-device weights reflecting free space and service rate. This yields balanced distribution, minimal data movement when devices are added/removed, and awareness of heterogeneous media (e.g., SSD vs. HDD).

Replica placement applies an anti-affinity rule: multiple replicas of the same object are never co-located on the same device. For large sequential reads, adjacent blocks can be co-scheduled on the same device to exploit readahead, while concurrent readers are spread across devices to maximize queue depth utilization. These policies are enforced at the storage-adapter layer and do not require changes to IPFS content addressing.

3.3. Integration with Analytical Frameworks

For practical use of the proposed system, it is necessary to provide convenient access for analytical tools to data stored in IPFS. Most big data frameworks (Spark, Presto, Flink) can read data from HDFS, S3, or the local file system, but are not aware of IPFS. Two integration approaches are proposed: via FUSE mounting and via specialized connectors.

FUSE bridge: IPFS provides the ability to mount its content to the local file system (ipfs mount). For example, ipfs mount mounts /ipfs (content by CID) and /ipns (named content) into the file system tree. Knowing the CID of a dataset’s root directory, it can be mounted and presented to Spark as a local directory. Then Spark (or Hadoop) will read files without knowing they come from IPFS. However, the standard FUSE implementation of IPFS is not designed for high-performance reading of huge files: it works sequentially and may incur context-switching overhead. Nevertheless, at the prototyping stage, FUSE is the simplest path. The plan is to mount the IPFS dataset on each Spark executor node. Thanks to adaptive replication, by the time a job starts, the required data is likely already replicated on those nodes (or nearby ones) that will read it – this is similar to data locality in Hadoop (when computation moves to the node with the data). Even if some data is not present locally, IPFS will fetch the missing blocks over the network.

In the future, a connector for Spark can be developed, implementing the Hadoop InputFormat interface, which will use the CID to access the IPFS API (HTTP gateway or go-ipfs library) for reading blocks. Such a connector could more efficiently parallelize reading by requesting dif-

ferent file fragments from different IPFS nodes. This is more complex but potentially faster, bypassing FUSE. Similarly, connectors can be developed for Flink or Presto.

Integration also includes a schema registry for tabular data: if the data is a table, e.g., Parquet files, it is necessary to store the table schema and partitioning. The indexing system can be extended: in addition to search indexes, dataset metadata can be maintained, similar to Hive Metastore, storing column descriptions, types, partitioning, and a list of CIDs for all file parts. This metadata file (JSON or protobuf) itself resides in IPFS and is versioned. The analytical engine (SparkSQL, Presto) can retrieve it via an IPNS link (a human-readable name pointing to the current metadata CID), thus always working with the up-to-date schema. In this work, we limit ourselves to describing the concept; implementation of the metadata repository is beyond the scope.

The uniqueness of the proposed method lies in combining content-dependent indexing with dynamic replication in a distributed system. These ideas have been seen separately (indexing for IPFS [10], dynamic replication for HDFS [4]), but their joint application in the context of IPFS for Big Data has not been described in the literature. The novelty is also in adapting IPFS, originally not intended for analytical clusters, by developing an additional data management layer. The proposed architecture essentially forms a hybrid of a decentralized P2P network with elements of centralized control (index and replication coordinators). This hybrid approach preserves the main advantage of IPFS – the absence of a single point of failure for the data itself – while introducing manageability characteristic of enterprise storage.

The practical significance of the solution is manifested in the potential to build inter-organizational data lakes based on IPFS. For example, several organizations can place large public datasets in IPFS; our system will ensure that these data are indexed for search and replicated where there is demand. This will facilitate data sharing without burdening a single storage center, eliminate duplication (thanks to content addressing), and reduce transfer costs – data will be loaded from the nearest nodes if they already have copies. For internal use, a company can avoid deploying expensive HDFS/Ceph clusters and instead use an IPFS cluster across different sites, automatically optimizing data placement for current tasks. In addition, replication flexibility will allow more efficient use of resources: unlike the rigid three-copy rule in HDFS, the system does not store unnecessary copies of cold petabytes, reallocating space for hot fragments.

The next section describes the plan for experimental evaluation of the proposed approach compared to traditional solutions.

4. Reference Systems for Comparison

Based on prevalence and architectural diversity, the following systems have been selected:

- **Apache Hadoop HDFS + Spark:** the classic big data cluster. It serves as a benchmark for sequential distributed processing. Configuration: three data replicas on HDFS (default), Spark deployed in Standalone or YARN mode. This allows us to assess how competitive our P2P system is compared to a time-tested technology.
- **Cloud storage S3 + Presto (Trino):** as a representative of object storage with an analytical engine. Presto (now Trino) is used for SQL queries on data in S3. It is interesting to compare query speeds in our system versus S3 (considering that S3 lacks data locality and each request pulls data from the cloud). Optionally, Amazon Athena (the same engine) can be used for purity.

- **Ceph (RADOS) + Spark:** CephFS or RADOS as the storage layer, Spark as the processing engine. Ceph will be configured for three replicas or equivalent fault tolerance via erasure coding. This will show how our system compares to a more modern distributed storage.
- **Lustre (parallel file system):** a widely used HPC parallel file system with dedicated metadata servers (MDS) and object storage servers (OSS). Lustre is included to represent a high-throughput baseline for large sequential scans and parallel I/O. Depending on resource availability, it may be evaluated on a small testbed; otherwise, it is listed for completeness with qualitative discussion, since deploying and tuning Lustre requires specialized infrastructure and administrative privileges.
- **(Optional) HDFS with dynamic replication:** if it is possible to implement or emulate a dynamic replication algorithm in HDFS (e.g., via HDFS Balancer or a third-party tool), the idea of adaptive replication in a centralized environment can be compared to the proposed approach. However, such a ready-made product is unlikely to be available for testing.
- **(Optional) Basic IPFS Cluster:** to see what improvements our extensions provide, a comparison can be made with “vanilla” IPFS Cluster, where the same data is stored with a fixed number of replicas. The difference will be due to indexing (Cluster lacks content search, only direct CID access) and replication dynamics.

For fairness, all systems will be deployed on similar hardware or in equivalent cloud conditions. The cluster size (number of nodes) and total storage volume will be the same to ensure a fair comparison. For example, we will use 10 nodes, each with 16 CPUs, 64 GB RAM, and a 1 Gbit/s network (for IPFS, 10G is possible, but then HDFS should also have 10G, otherwise the conditions differ).

It is expected that the system will demonstrate better fault tolerance (due to the absence of a NameNode and overall decentralization) and potentially higher throughput under a very large number of parallel requests (IPFS can scale content distribution via P2P, whereas HDFS, when a DataNode is overloaded, hits the network limit). We also expect acceleration under skewed workloads: when a small volume of hot data is requested frequently, adaptive replication should provide an advantage over static three copies [4]. On the other hand, for sequential reading of the entire dataset, traditional HDFS will likely perform as well or better – since it has been optimized for years, while IPFS has overhead. Success can be considered if our scan time is no more than 10–20% longer than HDFS. As for S3, we expect a significant advantage for our system in speed, especially for repeated queries, due to local copies and the absence of Internet latency.

Overall, the comparative analysis plan will reveal in which scenarios the proposed architecture truly outperforms existing solutions, and where it may lag and require further improvements.

5. Plan for Comparative Efficiency Analysis

To rigorously assess the advantages and limitations of the new system, a comparative experimental analysis must be conducted. The plan includes defining metrics, test scenarios, and selecting systems for comparison.

Efficiency metrics. The following key indicators are highlighted:

- **System throughput for reading and writing data.** For Big Data, the speed of reading large volumes is critical (how many MB/s or GB/hour the cluster can process). The aggregate speed of typical jobs (e.g., scanning the entire dataset) will be measured.
- **Average and maximum data access latency.** In a distributed environment, the time from data request to receiving the first byte and to receiving the last byte is important. In IPFS, there may be additional delays for peer discovery; these will be measured and compared to HDFS/S3.
- **Scalability:** how performance changes with the number of nodes and data volume. Linear scalability is important – e.g., when doubling the number of nodes, how much faster is processing.
- **Fault tolerance:** system behavior when part of the nodes fail (e.g., 10% or 30% of nodes go down during a job). The ability to continue operation without data loss and the time to restore replication will be evaluated.
- **Overhead:** the volume of service data (indexes, metadata) and traffic (e.g., how much extra traffic is generated by our replication algorithm compared to basic data transfer). CPU/memory usage on coordinating nodes will also be measured.
- **Execution time of typical analytical queries:** e.g., a SQL query with filtering, an aggregate query, machine learning on a dataset. This is a complex metric, depending on both storage and computation, but it is important to see if the ultimate goal – faster data analysis – is achieved.

According to these metrics, the new system will be considered efficient if:

- The throughput of reading/analysis is not lower, and preferably higher, than that of the compared systems (e.g., more GB/s per node).
- Access latency is acceptable (not much higher than HDFS). Our system may be slightly worse in latency due to DHT lookup, but this should be offset by parallel delivery. If the first-byte latency is no more than twice that of HDFS/S3, this is acceptable; the full scan latency should be comparable or better.
- In case of failures, the system does not lose data (this is a mandatory criterion) and continues to deliver it with minimal delays. Ideally, a 30% node failure should slow down operation by no more than $X\%$ compared to normal.
- The volume of index data and additional storage does not exceed reasonable limits, say, less than 10% of the main data volume (i.e., our index and metadata do not create significant redundancy).
- Infrastructure cost under equal conditions is not higher: if our system requires significantly more nodes or resources, this is a drawback. We assume similar nodes are used. The IPFS system has the advantage that nodes from different organizations can participate without a single storage center.

Test scenarios and workload. To comprehensively test the system, several types of tests are planned:

1. **Sequential scan of the entire dataset** – simulates a large batch analytical query (e.g., “compute statistics for the entire dataset”). The total execution time on our system and on the comparison systems will be measured. This will show the pure throughput of the disk subsystem and network.

2. **Multiple parallel queries to different data parts** – simulates the situation of several analyst teams or services working simultaneously. This will reveal the system’s ability to share resources and parallelize without conflicts.
3. **Hot data:** a scenario where 20% of the data is requested very frequently, the remaining 80% rarely. It is checked whether replication adapts – i.e., whether access to hot data accelerates over time compared to cold data. In our system, this should be evident: the first requests to a hot object are slower (few copies), later – faster (replicas added). In HDFS/standard systems, speed is fixed regardless of access frequency. We will compare the average access time to frequently requested files at the beginning and end of the test.
4. **Fault tolerance:** a long-running job is started, and during its execution, random nodes are “shut down”. We will measure whether the job completes successfully, how many tasks (Spark tasks) were restarted, and compare with HDFS/Spark when a DataNode fails – there should also be resilience, but it is interesting to compare recovery time.
5. **Indexing overhead:** the time to add a new file to the system with index update is measured. In HDFS, this is a file creation operation via NameNode – usually milliseconds; in our system – hash calculation, index update (may be hundreds of ms). It is acceptable if our delay is slightly higher, as indexing is not on the critical path of query execution, but we must ensure that even adding thousands of files does not “overload” the index coordinator.

6. Test Datasets

To evaluate the system, a representative large public dataset is required, sufficient in volume and complexity. Selection criteria: volume of at least several terabytes (to reveal scalability issues), presence of structure or content diversity, open availability and recognition, so that experimental results are reproducible by the community. Several candidates are considered:

- **Common Crawl** [5] – an open archive of web pages (HTML, WARC files) totaling tens of petabytes accumulated over many years. For our purposes, it is too large, but a subset can be used. For example, a monthly Common Crawl dump is about 100 TB of uncompressed data; a 1–5 TB sample is sufficient. The structure is many compressed WARC files (1 GB each), with diverse content (text, HTML). This dataset is good for testing with a large number of files and content-based indexing (e.g., by words). However, it is more difficult to analyze semantically (raw web data).
- **OpenStreetMap (OSM)** – global geodata describing the world map. The full OSM dump (planet file) is about 100 GB (XML or PBF format), which is relatively small; but derived data can be generated, such as tiles, indexes, geo-queries. OSM is well-structured (has schemas), allowing attribute-based search (roads, buildings, etc.). However, the volume may be insufficient for stress tests (hundreds of GB, not terabytes).
- **Enron Email Dataset** – a corpus of 0.5 million emails (tens of GB). Interesting for content analysis (text, communication graphs), but too small in volume, processed too quickly on a modern cluster, and will not reveal scalability issues.
- **CERN Open Data** – datasets from the Large Hadron Collider (LHC), particularly from CMS and ATLAS experiments. These data weigh many terabytes and have complex structure (ROOT files, particle event sets). The plus is realistic scientific Big Data, used to test distributed analysis (CERN uses systems like XRootD, similar to HDFS). The downside is the specific format, difficult to interpret without physics knowledge; however, for infras-

tructure testing, this is not a problem, as speed can be measured without understanding the content.

- **Generated data (TPC-DS benchmark)** – an artificial dataset can be generated according to the TPC-DS standard, e.g., at 10 TB scale. It is structured (simulates retail data: sales, products, customers, etc.) and widely used for comparative testing of SQL engines. The advantage is that identical SQL queries can be run on different systems and execution times compared. This option is good for evaluating analytical performance, though it is not a “real” external dataset.

Given the goals, the best choice appears to be a limited-volume Common Crawl in combination with a generated TPC-DS dataset. Common Crawl will allow testing on unstructured data and will check content-based indexing capabilities (e.g., searching for web pages containing specific terms). TPC-DS (10 TB) will provide a structured workload for SQL queries and comparison with benchmark results (it is known how long Spark or Presto take on these benchmarks).

Both can be used: first, configure the system on Common Crawl (e.g., 1 TB of data, 1000 files of 1 GB each) and test search/replication on it; then, on the same cluster, place TPC-DS (in Parquet format, partitioned by folders) with a volume of 10 TB and run the standard set of 99 TPC-DS queries in our system vs. competitors.

Specifically, the plan for Common Crawl: take, for example, the July 2022 Crawl, selectively download 1000 WARC files (1 GB each). Upload them to IPFS, record the CIDs. Set up indexing: for example, index by domains or frequently occurring words. Check that a search for “pages containing the word COVID” finds the corresponding CIDs via our index, and then these pages can be quickly retrieved. Also, simulate multi-user access: parallel requests for the 100 most popular WARC files – see if adaptive replication manages to distribute their copies to different nodes by the time the second/third user requests the same file.

Ultimately, the main focus of testing will be on TPC-DS 10 TB as a standardized benchmark. TPC-DS consists of fact tables (sales, 60 billion records) and dimension tables (products, stores, etc.). In Parquet format, the total volume is 10 TB. It is well suited because:

- it is widely known,
- contains both several very large files (fact tables) and many small ones (e.g., date table – 365 records),
- allows for complex SQL queries, forcing the system to read data with filtering, join, and aggregate – close to real retail analytics scenarios.

This dataset is structured, and for it, the indexing mechanism will be used for “dataset meta-data”: we will register the schema and possibly build an inverted index by some attribute (e.g., by year for the sales table, to quickly find all partition files for a year).

The chosen volume (10 TB) is large enough that a scan job on a 10-node cluster will last several minutes, allowing us to measure the difference. But not so large as to be unmanageable (the data can be stored and copied). If time and resources permit, 30 TB (TPC-DS scale 30TB) can be attempted.

Justification for dataset suitability: TPC-DS is the de facto standard for big data testing, so results will be understandable to the community and easily comparable with published results for Spark, Presto, etc. Common Crawl complements it by testing functions specific to IPFS (e.g., full-text search on unstructured content, which TPC-DS does not cover). Moreover, Common Crawl has immediate practical relevance: if our system can efficiently store a

web archive, it demonstrates suitability for real scenarios where data is not neat tables but heterogeneous files.

In sum, testing on these datasets will demonstrate both the versatility (working with tables and texts) and scalability of the proposed architecture. Successful completion of the tests will confirm that content-oriented indexing and adaptive replication enable the IPFS network to evolve into a Big Data platform, combining the strengths of decentralization with the requirements of high-performance analytics.

Conclusion

This work proposes a new architecture for big data storage and analysis, combining the decentralized IPFS network with mechanisms for dynamic content-oriented indexing and adaptive replication. The analysis shows that existing approaches (HDFS, Ceph, S3), while having become industry standards, have significant limitations in flexibility and require substantial resources for scaling, whereas IPFS offers a promising foundation due to content addressing and P2P interaction, but needs extended functionality for Big Data applications. The proposed solution fills this gap: the content-oriented index provides efficient search and organization of data by content, and the adaptive replication algorithm distributes data copies optimally according to current load, improving system performance and reliability.

The scientific novelty of this work lies in the integration of indexing and search methods with the distributed hash tables of IPFS and in introducing self-* properties into storage – the system autonomously responds to access patterns, reallocating resources. The practical value is confirmed by scenarios where decentralized storage is essential: inter-organizational data exchange, open data publication, fault-tolerant storage without a single owner. The proposed architecture enables such systems to be built without strict dependence on cloud providers and with savings on redundant data copies.

Future work includes completing the prototype and conducting the planned comparative analysis. The empirical evaluation is outside the scope of this paper and will be reported separately. The knowledge gained will serve as a basis for further optimization – in particular, improving replica placement algorithms, possibly considering network topology (to minimize latency), and implementing a fully decentralized indexing variant using CRDT, eliminating even the index coordinator as a point of failure. Another direction may be integration with blockchain technologies for access auditing or storage incentives (e.g., Filecoin) [2].

In summary, dynamic content-oriented indexing and replication based on IPFS represent a step toward the next generation of Big Data systems, combining the scalability and openness of peer-to-peer networks with the requirements of enterprise storage. The proposed architecture is both scientifically novel and practically significant, and its successful implementation could greatly expand the applicability of IPFS and similar decentralized technologies in the era of big data.

Acknowledgements

The author expresses gratitude to the scientific advisor Stanislav V. Suvorov for consultations and support during this work.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Benet, J.: IPFS - content addressed, versioned, P2P file system. <https://doi.org/10.48550/arXiv.1407.3561>
2. Benet, J., Greco, N., *et al.*: Filecoin: A decentralized storage network. Protocol Labs report, 2017.
3. Cao, L., Li, Y.: IPFS keyword search based on double-layer index. In: Proceedings of the International Conference on Electronic Information Engineering and Computer Communication (EIECC 2021), vol. 12172, pp. 1217209. SPIE (2022). <https://doi.org/10.1117/12.2639406>
4. Cao, X., Wang, C., Wang, B., He, Z.: A method to calculate the number of dynamic HDFS copies based on file access popularity. *Mathematical Biosciences and Engineering* 19(12), 12212–12231 (2022). <https://doi.org/10.3934/mbe.2022583>
5. Common Crawl Foundation. Common Crawl - Open Web Data (HTML, WARC files). <https://commoncrawl.org/> (2025), accessed: 2025-05-25
6. Maymounkov, P., Mazières, D.: Kademlia: A peer-to-peer information system based on the XOR metric. In: *Peer-to-Peer Systems. IPTPS 2002. Lecture Notes in Computer Science*, vol. 2429, pp. 53–65. Springer, Berlin, Heidelberg (2002). https://doi.org/10.1007/3-540-45748-8_5
7. Estrada-Galiñanes, V., ElRouby, A., Theytaz, L.: Towards efficient data management for IPFS-based applications. <https://doi.org/10.48550/arXiv.2404.16210>
8. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The Hadoop Distributed File System. In: *IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST 2012, Lake Tahoe, Nevada, USA, May 3-7, 2010*, pp. 1–10. IEEE (2010). <https://doi.org/10.1109/MSST.2010.5496972>
9. Weil, S.A., Brandt, S.A., Miller, E.L., Maltzahn, C.: Grid resource management - CRUSH: controlled, scalable, decentralized placement of replicated data. In: *Proceedings of the ACM/IEEE SC2006 Conference on High Performance Networking and Computing*, November 11-17, 2006, Tampa, FL, USA, pp. 122. ACM (2006). <https://doi.org/10.1145/1188455.1188582>
10. Zhu, Z., Cen, F.: Research on key technologies of information search based on IPFS. In: *International Conference on Electronic Information Engineering and Computer Communication (EIECC 2021)*, pp. 1217201. SPIE (2022). <https://doi.org/10.1117/12.2640819>