DOI: 10.14529/jsfi180104

The High-Q Club: Experience with Extreme-scaling Application Codes

***Dirk Brömmel*[1], *Wolfgang Frings*[1], *Brian J. N. Wylie*[1], *Bernd Mohr*[1], *Paul Gibbon*[1], *Thomas Lippert*[1]**

Jülich Supercomputing Centre (JSC) started running (extreme) scaling workshops with its first IBM Blue Gene supercomputer, finally spanning three generations each seeing an increase in the number of cores and available threads. Over the years, this workshop series attracted numerous international code teams and resulted in many applications capable of running on all available cores of each system.

This article reviews some of the knowledge gained with running and tuning highly-scalable applications, focussing on *JUQUEEN*, the IBM Blue Gene/Q at JSC. The ability to execute successfully on all 458,752 cores with up to 1.8 million processes or threads may qualify codes for the High-Q Club, which serves as a showcase for diverse codes scaling to the entire 28 racks, effectively defining a collection of the highest scaling codes on *JUQUEEN*. The intention was to encourage other developers to invest in tuning and scaling their codes while identifying the necessary key aspects for that goal.

As this era closes, it is timely to compare the characteristics of the 32 High-Q Club member codes, considering their strong and/or weak scaling, exploitation of hardware threading, and whether/how intra-node multi-threading is employed combined with message-passing. We also identify the obstacles for scaling such as inefficient use of limited compute node memory and file I/O as key governing factors. Overall, the analysis provides guidance as to how applications may (need to) be designed in the future to exploit expected exascale computer systems.

Keywords: JUQUEEN, IBM Blue Gene/Q, extreme scaling, application codes, High-Q Club.

Introduction

Jülich Supercomputing Centre (JSC) has more than a decade of experience with the range of IBM Blue Gene systems and scaling HPC applications to use their considerable capabilities effectively. Applications that demonstrate scalability to exploit the entire computational resources of the *JUQUEEN* system qualify for recognition in the High-Q Club. With the decommissioning of *JUQUEEN* in spring 2018, it is timely to analyse the characteristics of these extremely scalable applications for valuable insight into how applications may (need to) look in the future to exploit forthcoming exascale computer systems. Furthermore, supporting development tools and libraries also need to have commensurate scalability to address current and future application needs on these massively-parallel systems.

We start by reviewing application scaling activities at JSC in Section 1, focussing on its leadership computer systems, and in Section 2 the *JUQUEEN* hardware environment, followed with an overview of the associated High-Q Club in Section 3. Member applications and their basic characteristics are summarised in Section 4, as adapted to the BG/Q software environment, while Section 5 is a detailed comparison of the demonstrated scalability of these codes on *JUQUEEN*. Specific tools which have proven effective for use with them at (very) large scale are then reviewed in Section 6. Finally we present our conclusions from this study regarding High-Q Club member application codes and readiness for exascale.

[1]Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Jülich, Germany

1. Background

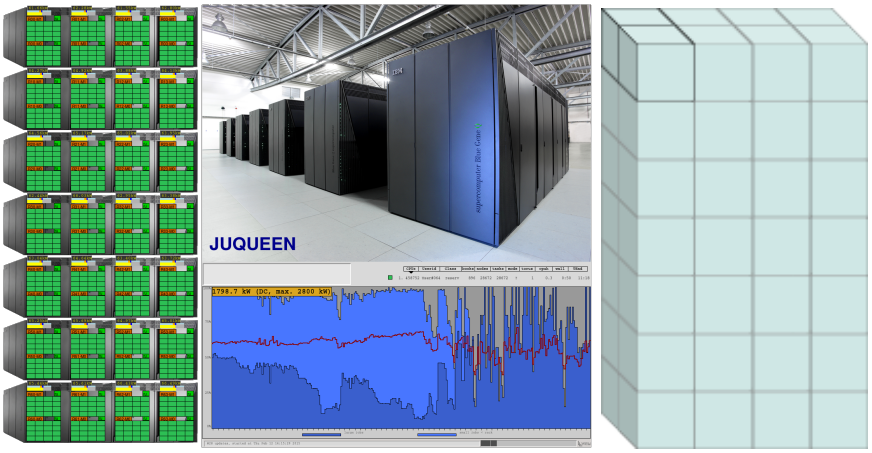
The first of the IBM Blue Gene series installed in 2005 by Jülich Supercomputing Centre was the *JUBL* Blue Gene/L, succeeded by the *JUGENE* Blue Gene/P [2] in 2007, and ultimately the *JUQUEEN* Blue Gene/Q [34] in 2012 which remained in operation to 2018. Key characteristics of these supercomputers are summarised in Tab. 1. Part of each of these systems was normally reserved for small/short application development executions, complemented by a variety of larger partitions of different sizes for longer jobs. While batch jobs could also be submitted to queues requiring the full system, they would only be run immediately following maintenance sessions or other predefined times to avoid undue interference with general usage. "Big Blue Gene Weeks" with seven or more days restricted exclusively to large-scale jobs were introduced first in 2015 and readily were exploited by numerous application teams, both for scaling tests and production. Before then, scaling-up applications could be a protracted process as a series of larger jobs was prepared, eventually executed, and adjusted. Dedicated scaling, and subsequently extreme scaling, workshops were held to facilitate this.

A "Blue Gene/L Scaling Workshop" [10] was held in 2006, became a "Blue Gene/P Porting, Tuning & Scaling Workshop" in 2008 [25], followed by dedicated "Extreme Scaling Workshops" in 2009 [26], 2010 [27] and 2011 [28]. These latter three workshops attracted 28 teams selected from around the world to investigate scalability on the most massively-parallel supercomputer at the time with its 294,912 cores. 26 of their codes were successfully executed at that scale, three became ACM Gordon Bell prize finalists, and one participant was awarded an ACM/IEEE-CS George Michael Memorial HPC fellowship.

"Extreme Scaling Workshops" for Blue Gene/Q continued in 2015 [4], 2016 [6] and 2017 [7] with a similar format. Based on their demonstrated and planned use of BG/Q, a total of 19

Table 1. Blue Gene series of computers installed at Jülich Supercomputing Centre

Name	*JUBL*	*JUGENE*	*JUQUEEN*
Installation year	2005	2007	2012
Series	BG/L	BG/P	BG/Q
Racks	8	72	28
Compute nodes	8,192	73,728	28,672
Processor	PowerPC 440	PowerPC 450	PowerPC A2
Frequency (MHz)	700	850	1,600
Compute cores	2	4	16
Hardware threads/core	1	1	4
Total cores	16,384	294,912	458,752
Total threads	16,384	294,912	1,835,008
Main memory (TiB)	4.1	144	448
Memory per core (MiB)	256	512	1,024
I/O Nodes	288	600	248
Network	3D torus	3D torus	5D torus
Peak performance (TFlop/s)	45.9	1,002.7	5,872.0
Footprint (m^2)	15	130	82
Power consumption (kW)	179	2,268	2,301



JUQUEEN

Partition	Cores
$8 = 4\times1\times2$	65,536
$16 = 4\times2\times2$	131,072
$24 = 4\times3\times2$	196,608
$32 = 4\times4\times2$	262,144
$40 = 4\times5\times2$	327,680
$48 = 4\times6\times2$	393,216
$14 = 1\times7\times2$	114,688
$28 = 2\times7\times2$	229,376
$42 = 3\times7\times2$	344,064
$56 = 4\times7\times2$	458,752

Figure 1. *JUQUEEN* Blue Gene/Q as presented by the *LLview* system monitor (28 racks with two midplanes each with 16 nodeboards of 32 processors), schematic of the 56 BG/Q rack midplanes arranged $4\times7\times2$ and list of partitions with corresponding number of cores available when scaling rows or columns of racks to the entire configuration

international teams were selected and hosted on-site by JSC for two or three days with dedicated access to *JUQUEEN* for up to 50 hours in each event. Many of the teams' codes had thematic overlap with JSC Simulation Laboratories (SimLabs), which provided assistance during the workshops along with Cross-Sectional Teams available to do performance analyses and suggest optimisation opportunities. The first of these workshops had seven applications all running successfully within 24 hours on all 28 racks (using 458,752 cores), however, in the following two editions some participants encountered difficulties which they were unable to resolve, and there were only eleven additional successes. File I/O bottlenecks were a frequent constraint for some codes, while large-scale in situ interactive visualization using 458,752 MPI processes running on 28 racks coupled via *JUSITU* to VisIt was successfully demonstrated as a possible alternative [16].

These workshops motivated Leibniz Supercomputing Centre (LRZ), JSC's partner with HLRS in the German national Gauss Centre for Supercomputing (GCS), to adopt a similar format for workshops to scale applications on the *SuperMUC* IBM iDataPlex system [18], and similar opportunities are expected to be offered on future HPC systems at JSC.

2. *JUQUEEN* Blue Gene/Q

JUQUEEN[2] is an IBM Blue Gene/Q system consisting of 28 racks [34] (Fig. 1), each with two midplanes comprising 512 compute nodes with 1.6 GHz PowerPC A2 processors and 16 GiB RAM, connected via a custom five-dimensional torus network. Compute node processors provide 16 cores to applications, each with a 256-bit SIMD/vector unit and capable of running four hardware threads, therefore *JUQUEEN* offers a total of 458,752 cores and can concurrently run 1,835,008 processes or threads. An additional 248 I/O nodes connect via Cisco network switches to the *JUST* GPFS filesystem.

IBM Blue Gene/Q systems [1] (like their predecessors) have demonstrated their merits for large-scale HPC regarding energy-efficiency, reliability and scalability to extremely large con-

[2] `http://www.fz-juelich.de/ias/jsc/juqueen`

figurations, making them highly-productive workhorses. *JUQUEEN* was ranked number five for energy efficiency as well as fifth for HPL performance in the November 2012 Green500 and Top500 lists of supercomputers, and coarse-grain energy usage is monitored during operation (though no control is available for applications to adjust their usage). System warnings and errors are also closely monitored, leading to pre-emptive hardware replacement at the next scheduled maintenance, such that their is rarely impact on production (and particularly full-system jobs). Application-level checkpointing is still recommended for long-running jobs.

By providing dedicated system partitions, applications receive isolated resources for computation and communication, the latter based on a proprietary 5-D torus network. With an extra 17th processor core for system services and lightweight compute-node kernel (CNK), applications are further isolated from "system noise" that can otherwise severely impact performance of collective operations. And with uniform memory access, process/thread placement and bindings to address NUMA issues are unnecessary. On the other hand, compute-node memory is limited (typically 16 GiB) and the relatively weak individual processor cores require the effective use of large numbers of them by applications, which is further exacerbated by the in-order processor architecture [20]. Multiple application threads for the four hardware threads per processor core are therefore generally considered to be the best bet for increased instruction throughput and latency hiding.

Figure 1 shows a topological schematic of the 28 BG/Q racks of *JUQUEEN* with its 56 midplanes arranged $4 \times 7 \times 2$. When performing scaling tests using rows or columns of racks, increments of 8 or 14 midplanes are available. (Square numbers of 1, 4, 9 and 16 racks can also be used, however, $25 = 5 \times 5$ racks cannot be configured, which severely limits scaling.) Note that physical adjacency of racks is not essential to exploit mesh or torus topologies, however, even-sized dimensions generally offer superior performance since odd-sized dimensions do not support torus connectivity. 16 racks (32 midplanes) therefore constitute the preferred configuration for compact torus communication, and this partition is available and commonly used in general operation of *JUQUEEN*. Partitions larger than this are typically only made available after maintenance periods and for specially scheduled sessions. In particular, full-system executions requiring all 28 racks occupy the rack that is otherwise reserved for small/short development jobs. Large jobs using 24 racks are therefore often a convenient compromise.

Bearing these considerations in mind, executions on partitions with 24 racks (48 midplanes, 86% of entire system) and perhaps even only 20/21 racks (40/42 midplanes, 71/75%) could be covered by the definition of "large-scale" with respect to *JUQUEEN*.

3. High-Q Club

The High-Q Club is a collection of the highest scaling codes on *JUQUEEN*, and membership requires codes to run successfully using all 28 racks. Codes also have to demonstrate that they profit from each additional rack of *JUQUEEN*, either with reduced time to solution when strong scaling a fixed problem size or a tolerable increase in runtime when weak scaling progressively larger problems. Furthermore, application configurations should be beyond toy examples, and use of all available hardware threads is encouraged which is often best achieved via mixed-mode programming. Each code is then individually evaluated based on its weak or strong scaling results with no strict limit on efficiency.

The full description of the High-Q Club codes and a summary of their scaling performance along with developer and contact information is maintained on-line [22]. Further detail is avail-

able in participants' reports from Extreme Scaling Workshops [4, 6, 7]. 32 codes are listed (with those from Extreme Scaling Workshops marked with an asterix*):

1D-NEGF *1D Non-Equilibrium Green's Functions framework for transport phenomena*
JSC SimLab Quantum Materials

*CIAO *multiphysics, multiscale NS solver for turbulent reacting flows in complex geometries*
RWTH Aachen University Institute for Combustion Technology [16]

*Code_Saturne *multiphysics simulation of the Navier-Stokes equations*
EDF & STFC Daresbury Laboratory[3]

*CoreNeuron *electrical activity of neuronal networks with morphologically-detailed neurons*
EPFL Blue Brain Project and Yale University [29]

dynQCD *lattice quantum chromodynamics with dynamical fermions*
JSC SimLab Nuclear and Particle Physics & Bergische Universität Wuppertal

*FE2TI *scale-bridging approach incorporating micro-mechanics in macroscopic simulations of multi-phase steels*
Universities of Cologne, Freiberg, Duisburg-Essen, Dresden and Erlangen-Nürnberg [23]

*FEMPAR *massively-parallel finite-element simulation of multi-physics PDE problems*
Universitat Politècnica de Catalunya CIMNE

Gysela *gyrokinetic semi-Lagrangian code for plasma turbulence simulations*
CEA-IRFM Cadarache

hp-fRG *hierarchically parallelised functional renormalisation group calculations*
JSC [31]

*ICON *solver for fully compressible non-hydrostatic motion at very high horizontal resolution*
Deutsches Klimarechenzentrum (DKRZ) & JSC SimLab Climate Science

IMD *classical molecular dynamics simulations*
Ruhr-Universität Bochum & JSC SimLab Molecular Systems

JURASSIC *infrared radiative transfer in the Earth's atmosphere*
JSC SimLab Climate Science

JuSPIC *fully relativistic particle-in-cell code for plasma physics and laser-plasma interaction*
JSC SimLab Plasma Physics

*KKRnano *Korringa-Kohn-Rostoker Green function code for quantum description of nano-materials in all-electron density-functional calculations*
Forschungszentrum Jülich IAS

LAMMPS-DCM *molecular dynamics simulation with dynamic cutoff method for arbitrarily large interfacial systems*
RWTH Aachen University AICES [33]

MP2C *massively-parallel multi-particle collision dynamics for soft matter physics and mesoscopic hydrodynamics*
JSC SimLab Molecular Systems [9]

*MPAS-A *multi-scale non-hydrostatic atmospheric model for global, convection-resolving climate simulations*
Karlsruhe Institute of Technology & National Center for Atmospheric Research [19]

[3] `http://www.code-saturne.org/`

$\mu\phi$ (muPhi) *algebraic multi-grid solver for simulation of water flow and solute transport in porous media*
Universität Heidelberg

Musubi *multi-component Lattice Boltzmann solver for flow simulations*
Universität Siegen [30]

NEST *large-scale simulations of biological neuronal networks*
Forschungszentrum Jülich INM-6, IAS-6 & JSC SimLab Neuroscience [21]

OpenTBL *direct numerical simulation of turbulent flows*
Universidad Politécnica de Madrid

*ParFlow+p4est *high resolution parallel simulation of variably saturated flow*
Forschungszentrum Jülich IBG-3, Colorado School of Mines, LLNL & Universität Bonn [8]

*pe *physics engine framework for simulations of rigid bodies with arbitrary shapes*
Universität Erlangen-Nürnberg[4]

PEPC *tree code for N-body simulations, beam-plasma interaction, vortex dynamics, gravitational interaction, molecular dynamics simulations*
JSC SimLab Plasma Physics

PMG+PFASST *space-time parallel solver for systems of ODEs with linear stiff terms*
LBNL, Universität Wuppertal, Università della Svizzera italiana & JSC

PP-Code *simulator for relativistic and non-relativistic astrophysical plasmas*
University of Copenhagen

*psOpen *direct numerical simulation of fine-scale turbulence*
Jülich-Aachen Research Alliance & TU Freiberg [17]

*Seven-League Hydro (SLH) *all Mach number fluid dynamics in astrophysics*
Heidelberg Institute for Theoretical Studies[5]

*SHOCK *structured high-order finite-difference computational kernel for numerical simulation of compressible flow*
RWTH Aachen University Shock Wave Laboratory [14]

TERRA-NEO *simulation of Earth mantle dynamics*
Universität Erlangen-Nürnberg, LMU & TUM

waLBerla *Lattice-Boltzmann method for simulation of fluid scenarios*
Universität Erlangen-Nürnberg

ZFS *multiphysics framework for compressible/incompressible flow, aero-acoustics & combustion*
RWTH Aachen Uni. Inst. of Aerodynamics & JSC SimLab Fluids and Solids Engineering

Half of the member codes involved institutions from the local region, ten were from the rest of Germany, and six from other European nations. International collaborations included four institutions in the USA.

4. Parallel Program & Execution Configuration Characteristics

Characteristics of these application codes (as contributed by the respective code teams submitting their data) are summarised in Tab. 2 and discussed in this section, with scaling performance compared in the following section. Five codes were accepted to the High-Q Club when

[4] https://www10.informatik.uni-erlangen.de/Research/Projects/pe/
[5] https://slh-code.org/

Table 2. High-Q Club member application code characteristics

Compiler and main programming languages (excluding external libraries), parallelisation including maximal process/thread concurrency (per compute node and overall) and strong and/or weak scaling type, and file I/O implementation. (Supported capabilities unused for scaling runs on *JUQUEEN* in parenthesis)

		Programming				Parallelisation							
Code	Accepted	Compiler / Languages				Tasking		Threading		Concurrency	Type		File I/O
1D-NEGF	2018/02	XL:	C			MPI	1	OpenMP	64	64: 1,835,008	S		N/A
*CIAO	2015/08	XL:			Ftn	MPI	16			16: 458,752	S		MPI-IO,HDF5
*Code_Saturne	2016/03	XL:	C		Ftn	MPI	16	OpenMP	4	64: 1,835,008	S		MPI-IO
*CoreNeuron	2015/02	XL:	C	C++		MPI	1	OpenMP	64	64: 1,835,008	S	W	MPI-IO
dynQCD	2013/06	XL:	C			SPI	1	pthreads	64	64: 1,835,008	S		*unspecified*
*FE2TI	2015/02	XL:	C	C++		MPI	16	OpenMP	4	64: 1,835,008	S	W	N/A
*FEMPAR	2014/12	XL:			F08	MPI	64	(OpenMP)		64: 1,756,001		W	N/A
Gysela	2013/06	XL:	C		F90	MPI		OMP+pthrd		64: 1,835,008		W	(HDF5)
hp-fRG	2016/10	XL:	C	C++		MPI	1	OpenMP	32	32: 917,504	S		N/A
*ICON	2015/02	XL:	C		Ftn	MPI	1	OpenMP	64	64: 1,835,008	S		(netCDF)
IMD	2014/10	XL:	C			MPI	64	(OpenMP)		64: 1,835,008		W	*unspecified*
JURASSIC	2014/05	XL:	C			MPI	32	OpenMP	2	64: 1,835,008		W	netCDF
JuSPIC	2013/10	GCC:			F90	MPI	4	OpenMP	16	64: 1,835,008	S		MPI-IO, POSIX
*KKRnano	2014/10	XL:			F03	MPI	4	OpenMP	16	64: 1,835,008	S		SIONlib
LAMMPS-DCM	2015/04	XL:		C++		MPI	4	OpenMP	16	64: 1,835,008	S	W	N/A
*MPAS-A	2017/03	XL:	C		Ftn	MPI	16	(OpenMP)		16: 458,752	S		SIONlib,PIO,pNetCDF
MP2C	2013/11	XL:			Ftn	MPI	32			32: 917,504		W	SIONlib
muPhi ($\mu\phi$)	2013/11	XL:		C++		MPI	32			32: 917,504		W	SIONlib
Musubi	2014/12	XL:			F03	MPI		OpenMP		32: 917,504	?	?	N/A
NEST	2013/11	XL:		C++		MPI	1	OpenMP	64	64: 1,835,008		W	(SIONlib)
OpenTBL	2014/05	XL:			F03	MPI		OpenMP		64: 1,835,008		W	pHDF5
*ParFlow+p4est	2017/03	XL:	C		Ftn	MPI	16			16: 458,752	S		MPI-IO
*pe	2017/03	GCC:		C++		MPI	64			64: 1,835,008		W	N/A
PEPC	2013/06	GCC:	C		F03	MPI	1	pthreads	61	61: 1,744,992		W	(SIONlib,MPI-IO,vtk)
PMG+PFASST	2013/08	XL:	C		F03	MPI	16	(pthreads)		16: 458,752	S		N/A
PP-Code	2014/08	XL:			F90	MPI		OpenMP		64: 1,835,008	S	W	*unspecified*
*psOpen	2015/02	XL:			F90	MPI	32	OpenMP	2	64: 1,835,008	S		pHDF5
*SHOCK	2015/02	XL:	C			MPI	64			64: 1,835,008	S	W	(cgns/HDF5)
*SLH	2016/02	XL:	C		F95	MPI	16	OpenMP	4	64: 1,835,008	S		MPI-IO
TERRA-NEO	2013/06	XL:		C++	Ftn	MPI		OpenMP		64: 1,835,008		W	*unspecified*
waLBerla	2013/06	XL:		C++		MPI		OpenMP		64: 1,835,008	?	?	N/A
ZFS	2015/03	Clang:		C++		MPI	16	OpenMP	2	32: 917,504	S		(pNetCDF)

it opened in June 2013, and another group of five codes added as a result of the Extreme Scaling Workshop in February 2015, otherwise membership submission requests and acceptance were more sporadic and generally diminishing towards 2018 when *JUQUEEN* is due to be decommissioned. Throughout this period no particular trends seem discernable in the characteristics of the accepted codes.

Member codes are typically able to run in a variety of configurations, which may trade-off different capabilities and indeed evolve as the code continues to be developed, however, this analysis provides a snapshot at the point of qualification for membership.

Programming Languages. IBM provide their XL suite of optimising compilers for C, C++, and Fortran which feature support for automatic SIMD vectorisation and QPX vector intrinsics, as well as transactional memory (TM) and speculative execution (SE), though relatively few codes have investigated or currently exploit these capabilities. GCC and LLVM/Clang compilers are also available, often providing support for newer language standards (e.g., C++11) and non-standard extensions which can ease porting, or in some cases delivering better performance: these have been exploited by several High-Q member codes (JuSPIC, pe, PEPC & ZFS).

Since Blue Gene/Q offers lower-level function calls for some hardware-specific features that are sometimes not available for all programming languages, a starting point is looking at the languages used. Figure 2 (left) shows a Venn set diagram of the main programming language(s) used, i.e. languages used by the parallel application itself and not its auxiliary libraries or pre/post-processing components. It indicates that all three major programming languages are roughly equally popular (without considering lines of code). Seven combine Fortran with C, three used C++ and C, one Fortran and C++, and the remainder exclusively used a single language. Notably, Python and similar languages relying on dynamic linking have not been used

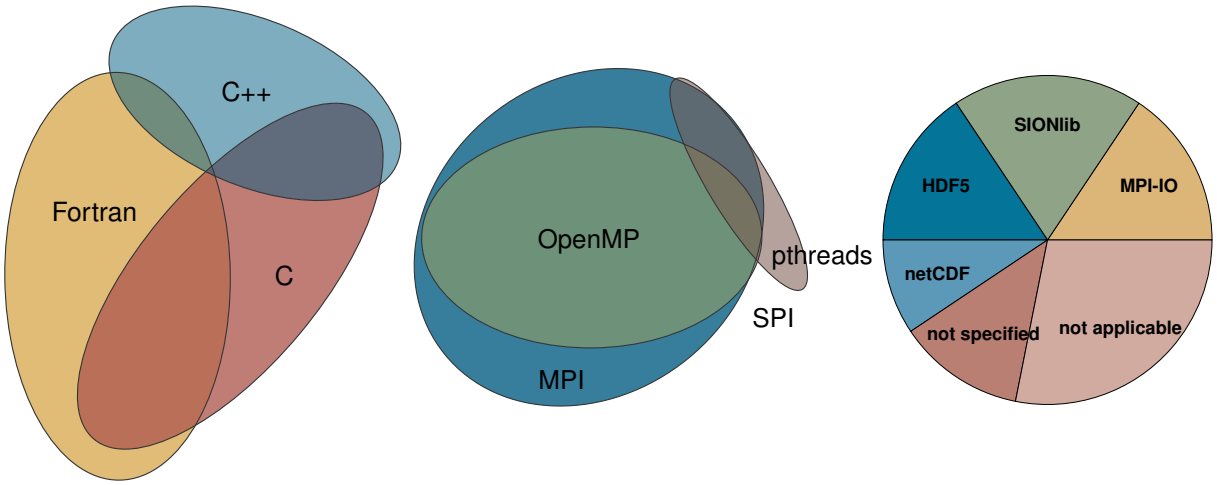


Figure 2. Left: Venn set diagram of main programming languages used by High-Q Club member codes. Middle: Venn set diagram of parallelisation modes of High-Q Club member codes run on *JUQUEEN*. Right: Pie-chart showing file I/O as available in High-Q Club member codes

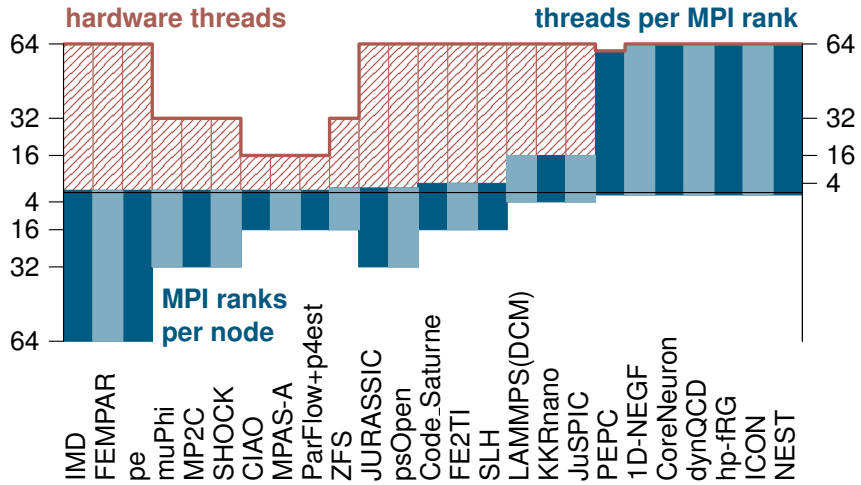


Figure 3. Bar chart of MPI ranks per node and threads per MPI rank, determining the number of hardware threads exploited, for executions of High-Q Club member codes on *JUQUEEN*

by any High-Q Club member code, even though tools like *Spindle*[6] were developed to address performance issues with loading shared libraries [12].

Optimised libraries are also provided by IBM for BG/Q, that are often both more convenient and performant than self-written versions. Only in a few special cases High-Q codes used their own libraries, e.g., psOpen for non-blocking 3D Fast Fourier Transforms [17]. Issues which initially inhibited scaling of ParFlow were addressed by employing the p4est library for mesh partitioning and an updated version of the HYPRE library of linear solvers [8].

Parallelisation Modes. The four hardware threads per core of the Blue Gene/Q chip in conjunction with the limited amount of memory recommend to make use of multi-threaded programming. It is therefore interesting to see whether this is indeed the preferred programming model and whether available memory is an issue. Figure 2 (middle) shows a Venn set diagram of the parallelisation modes used, revealing that mixed-mode programming does indeed dominate.

[6] https://computation.llnl.gov/project/spindle/

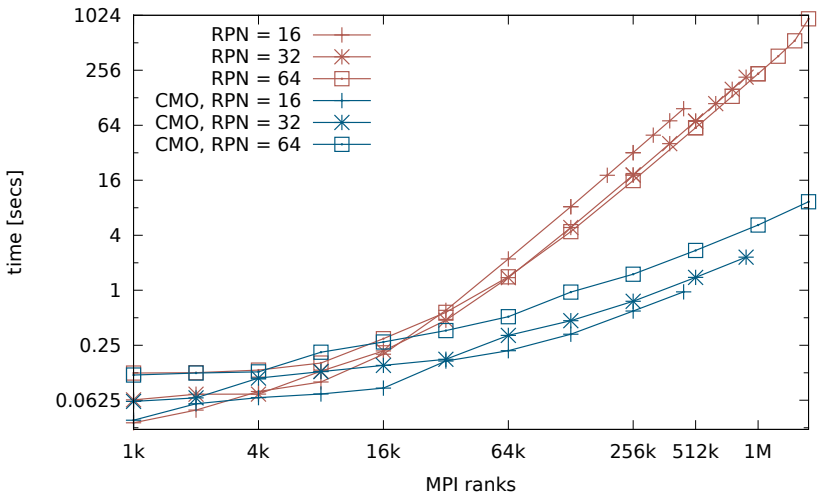


Figure 4. Scaling of wallclock execution time for `MPI_Comm_split` sub-communicator creation on *JUQUEEN* Blue Gene/Q: default and less memory-optimised alternative (CMO)

IBM provide an optimised implementation of MPI, which is almost ubiquitous for portable distributed-memory parallelisation, such that only one High-Q code (dynQCD) found it worthwhile to program an alternate version that directly used the underlying machine-specific SPI primitives. While a number of High-Q codes have demonstrated that they can scale when using only MPI ("MPI everywhere"), this requires adapting to very limited per-rank memory (i.e., less than 256 MiB/rank with 64 ranks per node) and switching to MPI communicator management routines that are optimised to reduce execution time rather than memory utilisation[7], further reducing memory available to the application itself. Figure 4 compares the execution time of both versions.

Ten codes exclusively used MPI for their scaling runs, both between and within compute nodes, accommodating to the restricted per-process memory and even trading higher memory requirements for faster MPI communicator management: this allowed FEMPAR to reduce sub-communicator creation time (`MPI_Comm_split`) from 15 minutes to under 10 seconds.

Convenient and portable multi-threading within compute nodes is supported via OpenMP, though some newer directives are not always optimised (e.g., for tasking). Only a few High-Q codes have pursued using POSIX threads (pthreads) for additional multi-threading control (dynQCD, Gysela, PEPC & PMG+PFASST). Almost all High-Q codes use MPI only from a single thread (`MPI_THREAD_FUNNELED`), with at least Gysela, ICON, and PEPC requiring `MPI_THREAD_MULTIPLE` which internally needs to use locks for synchronisation and precludes use of hardware support in the torus network for collective operations.

The majority of High-Q Club codes successfully employ OpenMP multi-threading to exploit compute node shared memory in conjunction with MPI. A memory fragmentation issue in a third-party library inhibited the use of OpenMP by FEMPAR, problems with nested parallel regions blocked MPAS-A, and an earlier investigation with the SHOCK code found this not to be beneficial. CoreNeuron has an ongoing effort investigating use of OpenMP-3 tasking and new MPI-3 capabilities (e.g. non-blocking collectives) are under consideration, so these are generally expected to become increasingly important.

[7] via setting the `PAMID_COLLECTIVES_MEMORY_OPTIMIZED` environment variable

File I/O Libraries. Figure 2 (right) shows a pie-chart breakdown of the I/O libraries used by High-Q Club codes, although in most cases writing output and in some cases reading input files was disabled for their large-scale executions, and synthesised or replicated data was used instead. Some of the (early) submissions for the High-Q Club unfortunately did not specify their file I/O usage. One quarter of the High-Q Club codes can use either (p)HDF5 or (p)NetCDF, despite their often disappointing performance, whereas one-sixth can use MPI file I/O directly. 20% of High-Q Club codes have migrated to using *SIONlib* for effective parallel I/O (see Section 6.1).

Compute Node Memory. For CoreNeuron available memory was the limiting factor for larger simulations, with the current limit being 155 million neurons using 15.9 GiB of RAM. The other neuroscience code NEST was similarly constrained by the amount of compute node memory available to store simulation data, however, ultimately managed to simulate 645 million neurons on *JUQUEEN* when using all available cores. MPAS-A required 1 GiB of memory on each process for its regular 3 km mesh simulation (over 65 million grid cells with 41 vertical levels), and could therefore only use a single hardware thread per core, limiting its effective performance. Using all four hardware threads of each processor code, FEMPAR was able to increase its efficiency and scalability to 1.75 million processes using $27\frac{1}{2}$ racks of *JUQUEEN* when employing an additional (fourth-)level of domain decomposition.

Concurrency. Figure 3 shows the relation between the number of MPI ranks and threads per node. On either side of this diagram are the two extremes of using all 64 hardware threads on each CPU by either 64 MPI ranks or 64 OpenMP/POSIX threads. Whereas multiples of two matching the available hardware were generally employed, PEPC delivered its best performance with the rather unusual number of 61 POSIX threads. Hatching shows the resulting number of hardware threads used by the codes, i.e. the concurrency. Clearly, codes benefit from using more hardware threads than physical cores and favour this configuration.

5. Comparison of Code Scalability

An overview of application code execution time scaling on *JUQUEEN* entails comparison of achievements in *strong* (fixed total problem size), and *weak* (fixed problem size per process or thread) scaling. This section reviews execution performance of submissions that qualified for membership. Various member codes continued to improve their performance and scalability (or that of additional functionality) beyond that of their qualifying submission.

A significant spread in execution results, and diverse scaling characteristics of the codes are visible in Fig. 5. A single BG/Q rack was chosen to provide a convenient baseline for scaling, however, a (half-rack) mid-plane would also have offered an isolated dedicated resource for this purpose. For strong scaling a minimum of three measurements spanning a factor of four in size up to the full configuration of 28 racks was mandated. Note that in many cases timings do not have a common baseline of one rack since datasets sometimes did not fit available memory, or no data was provided for 1,024 compute nodes: for strong scaling an execution with a minimum of seven racks (one quarter of *JUQUEEN*) is accepted for a baseline, and perfect scaling assumed from a single rack to the baseline. While full-system runs have a dedicated allocation of all 28 racks, other measurements were generally done when *JUQUEEN* is operational with a full and varying workload (which particularly impacts parallel I/O to the shared GPFS filesystem).

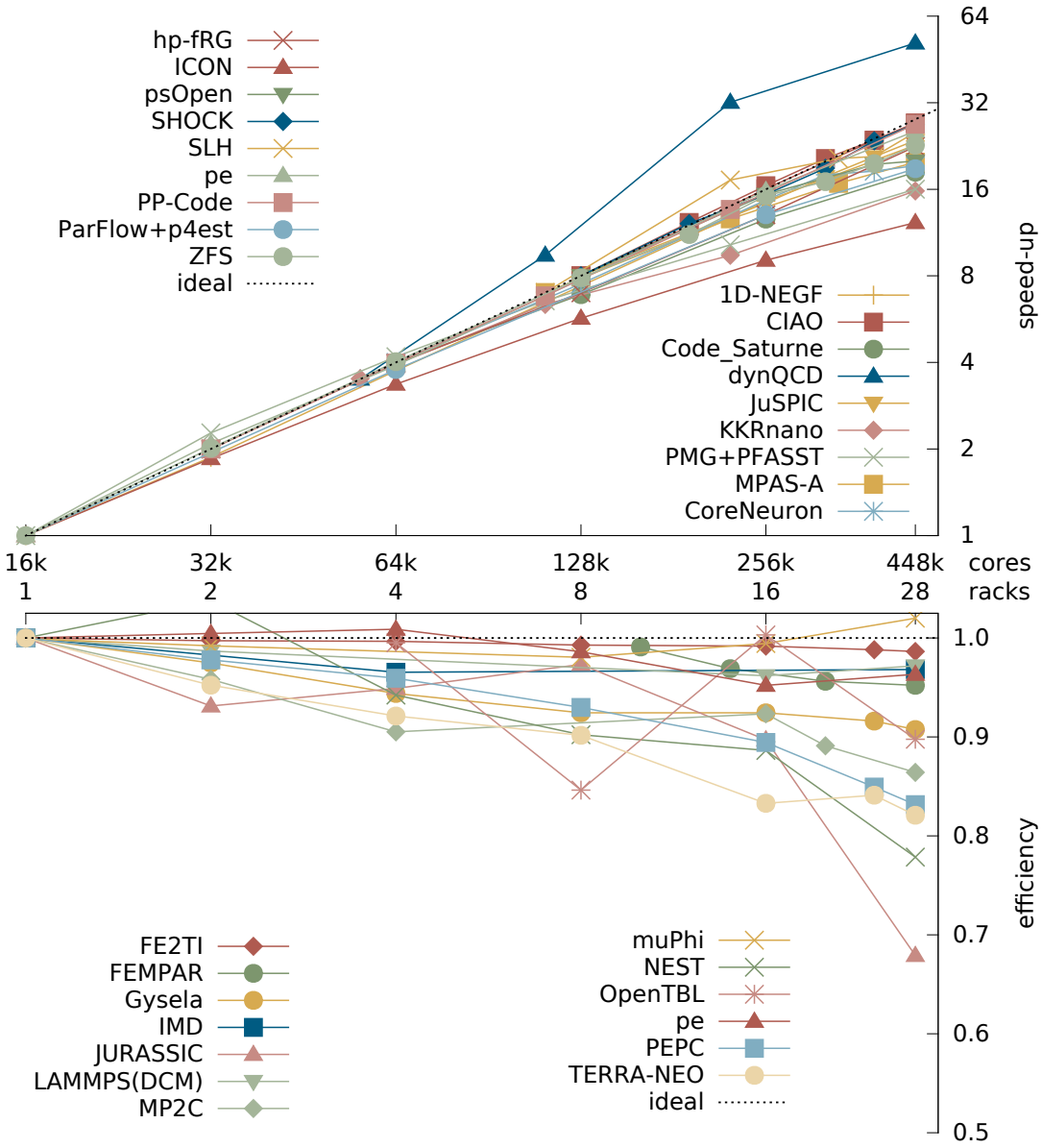


Figure 5. Strong and weak scaling of High-Q Club member application codes on *JUQUEEN*. Compared to a single rack, ideal strong scaling has a speed-up of 28× on 28 racks and weak scaling has 100% efficiency. Of the 18 codes showing strong scalability, ten maintained scaling efficiency above 80%, and only three were below 65% efficiency

Many High-Q Club member codes demonstrated very good strong-scaling speed-up close to 28×, with dynQCD standing out with superlinear speed-up of 52× due to its exceptional ability to exploit caches as problem size per thread decreases. ICON only achieved a modest 12× speed-up, and while this is less than 50% of the ideal, clear reduction in overall time to completion was shown.

Size of dataset was often critical for successful strong scaling to 28 racks, as diminishing per-rank computation can be overwhelmed by growing communication costs. Scaling of MPAS-A with a dataset of 65 million grid cells was only demonstrated to 24 racks (with worse performance for 28 racks), however, simulations using a 2 km global mesh with more than 147 million grid

cells scaled to the full 28 racks. Several codes switched to lower-precision (32-bit instead of 64-bit) datatypes to allow them to fit larger simulations in available memory.

Weak scaling is generally easier, as shown by the High-Q Club member codes maintaining over 80% efficiency from a single to 28 racks. JURASSIC only managed 68% efficiency, due to excessive I/O for the reduced-size test case, which was the lowest accepted for club membership, whereas muPhi was able to achieve 102% efficiency on 28 racks.

Various codes show erratic scaling performance, most likely due to topological effects. SHOCK is characterised by particularly poor configurations with an odd number of racks in one dimension (i.e. 4×3, 4×5 and 4×7). Similarly, OpenTBL shows marked efficiency drops for non-square numbers of racks (8 and 28).

Most optimisations employed by the codes are not specific to Blue Gene (or BG/Q) systems, but can also be exploited on other highly-parallel systems. High-Q Club codes have also run at scale on various Cray supercomputers, K computer, *MareNostrum-III*, *SuperMUC* and other x86-based computers, as well as on systems with GPGPUs [3].

6. Supporting Tools and Libraries

A variety of tools and libraries were invaluable during application tuning and scaling on *JUQUEEN*. Particularly during workshops, *LLview*[8] (shown in the left part of Fig. 1) facilitated monitoring the current system usage and additionally showing job energy consumption and file I/O performance. Custom mappings of MPI process ranks to *JUQUEEN* compute nodes generated by the *Rubik*[9] tool were investigated with psOpen and found to deliver some benefits, however, for the largest machine partitions these did not provide the expected reduction in communication times yet suffered from greatly increased application launch/initialisation time.

Efficient parallel file I/O libraries and performance analysis tools both delivered significant benefits for many applications, specifically when addressing extreme scalability.

6.1. Managing Parallel File I/O

A critical point attracting increasing attention is performance of file I/O, which is often a scalability constraint for codes which need to read and write huge datasets or open a large number of files. Large-scale executions of various High-Q member codes using the popular HDF5 and pNetCDF libraries needed to disable file I/O and synthesise initialisation data, e.g., CoreNeuron replicated a small dataset to fill memory to 15.9 GiB.

Initial full-scale runs of MPAS-A needed 20 minutes to load its initial condition data of 1.2 TiB using PIO/NetCDF and simulation output was disabled for large-scale tests to avoid similar writing inefficiency. This deficiency was subsequently addressed by adopting SIONlib to improve file I/O, allowing MPAS-A to load 1.4 TiB of finer resolution mesh data and output 4 TiB of model data to disk at three stages of the model run. Benefits were also observed on other systems, i.e. MPAS-A reported a 10x speed-up from PIO/NetCDF on 1,024 nodes or more on SuperMUC [7].

The *SIONlib*[10] library for parallel task-local file I/O, was specifically developed to address such file I/O scalability limitations [11, 13]. It has been used effectively by four High-Q codes

[8] http://www.fz-juelich.de/jsc/llview
[9] https://computation.llnl.gov/project/performance-analysis-through-visualization/software.php
[10] http://www.fz-juelich.de/jsc/sionlib/

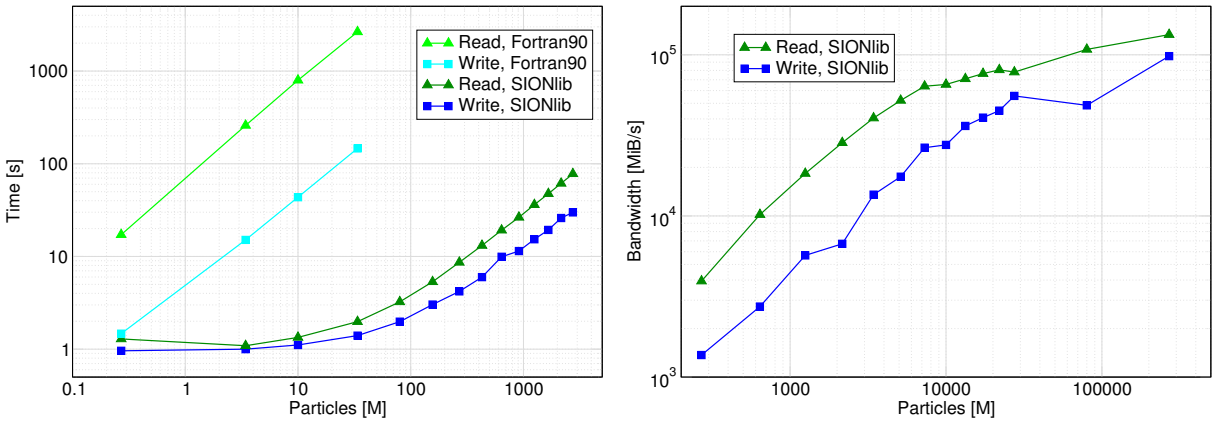


Figure 6. Left: MP2C file I/O time on one *JUQUEEN* midplane (8,192 MPI ranks) using traditional F90 I/O compared to improved I/O with *SIONlib*. Right: MP2C file I/O bandwidth for checkpointing executions with 1.8 million MPI ranks on 28 racks of *JUQUEEN* using *SIONlib* for reading and writing particle data

(KKRnano, MPAS-A, MP2C and muPhi) and several other applications are investigating migrating to adopt it (e.g. NEST [32]).

The scalability limitations of naïve parallel file management is demonstrated by I/O optimization results from the integration of *SIONlib* into MP2C [9]. MP2C couples multiple-particle collision dynamics with molecular dynamics to implement mesoscale simulation of hydrodynamic media. MP2C uses a large number of particles in its calculation, which have to be read from files in the initialisation phase and stored in files at the end of a simulation run in restart files. Furthermore, MP2C regularly does I/O to checkpoint files to be able to restart after system or program failure.

Originally, MP2C implemented I/O to a single file storing all particle data, which prevents bottlenecks which arise when using a large number of task-local files. However, data output was implemented with standard Fortran90 I/O calls for writing in two steps by first collecting the data on one task and then writing the data in a second step from that task to file. Data input is done individually by each MP2C MPI rank, reading the whole input file and selecting those particles that will be needed in the local domain. As both approaches limit the scalability of data input and output, I/O had to be improved for using it at large scale on *JUQUEEN*. Figure 6 (*Fortran90 write and read*) shows the results of initial benchmark runs on one BG/Q midplane with 8,192 MPI ranks using the traditional I/O approach. The measurements at the small scale of one midplane show the increasing time for I/O operations which hinders MP2C from using more than 50 million particles. However, the algorithm and its memory requirements allow orders of magnitude larger numbers of particles.

SIONlib supports efficient parallel file I/O for applications which use task-local I/O to individual files for each MPI rank. It was easily integrated into MP2C by replacing Fortran90 file open and close calls by corresponding *SIONlib* collective calls and write and read operations of local particle data with *SIONlib* I/O operations. After changing approximately 50 lines of code, we could reduce MP2C I/O time by several orders of magnitude. Furthermore, we could scale MP2C's I/O on one midplane to more than 4 billion particles, which is 125 times more than in the traditional approach (see *SIONlib write and read* in Fig. 6).

With integration of *SIONlib* into MP2C, it now runs with enabled I/O at the full scale of *JUQUEEN*. Figure 6 shows the I/O bandwidth for reading and writing different numbers of

particles. Data was stored on the GPFS scratch filesystem which provides a theoretical I/O bandwidth of 200 GiB/s. MP2C could achieve at the largest scale about 100 GiB/s for writing and more than 130 GiB/s for reading, which is 50–66 % of the peak bandwidth. For the largest number of particles (270 trillion), the restart file of MP2C was 14.4 TiB which could be written in 147 s (and read in 108 s).

6.2. Parallel Performance Analysis

During their development and specifically when preparing for large-scale runs on *JUQUEEN*, the execution performance of various High-Q Club applications was investigated with open-source tools. While most analyses are adequate at modest scales (e.g., a single BG/Q rack with 64k processes/threads), occassionally it is necessary to investigate performance issues that only manifest at larger scales.

The *Darshan*[11] tool was typically most convenient for low-overhead measurement and analysis of file I/O, distinguishing MPI file I/O from underlying POSIX file I/O per file and detailed breakdown of operation counts and I/O sizes. This was complemented by the *Scalasca*[12] toolset for scalable performance analysis of large-scale parallel applications [15] which is widely deployed on some of the largest HPC systems and clusters, supporting runtime summarization and event trace analyses of MPI, and OpenMP primarily focus on locating and quantifying communication and synchronization inefficiencies in C/C++/Fortran applications. Scalasca uses its own parallel trace tools with the community-developed *Score-P*[13] instrumentation and measurement infrastructure [24], itself based on OTF2 event trace and CUBE profile libraries. *SIONlib* is employed for efficient large-scale parallel file I/O when writing and reading OTF2 trace files (e.g., handling one container file per BG/Q IONode).

Scalasca measurements of applications running with 1.8 million threads on *JUQUEEN* have been done where 64 OpenMP threads are used for a single MPI process on each processor. In such an execution configuration the 16 GiB of processor memory is adequate to store profile and execution trace data collected during measurement for unification and collation during finalisation. This is not the case when 32 or 64 MPI processes split the available processor memory, along with memory required by the MPI library and application itself, restricting measurements to smaller configurations.

Parallel execution profiles are by default based on full compiler instrumentation of application user-level source routines, combined with OPARI2 instrumentation of OpenMP constructs and PMPI interposition on MPI library routines. Where these initial profile measurements manifest notable execution time dilation, filtering during measurement or selective instrumentation can be employed, directed by scoring which assesses event frequencies and associated measurement overheads. Iterative refinement of instrumentation and measurement of profiles is essential prior to collecting event traces where overheads are more significant, particularly with respect to in-memory buffering during measurement and final trace sizes.

Event traces are analysed by *Scalasca* trace tools with a parallel replay following measurement within the allocated partition, using the same configuration of MPI processes and OpenMP threads, to determine the fraction of OpenMP and MPI time due to waiting, the origins of delays, and critical execution path. Since traces are loaded entirely in memory for forward and

[11] `http://www.mcs.anl.gov/darshan`
[12] `http://www.scalasca.org/`
[13] `http://www.score-p.org/`

backward event replays, which require additional data-structures and pointers, the number of recorded events similarly governs the size of trace that can be analysed. (While analysis of smaller execution configurations may employ larger partitions with more memory, this option is not possible for event traces from the full system.)

Finally, the minimal set of metrics provided in *Score-P* profiles and *Scalasca* trace analysis reports are subsequently post-processed by a remapper which derives a large number of additional metrics and hierarchies. Since the CUBE remapper and GUI are serial processes requiring large amounts of RAM to process metrics in memory (and to minimise expensive paging to disk), generally these are best done on dedicated visualisation nodes, such as those of the JSC general-purpose Linux cluster *JURECA* with 1 TiB shared RAM.

During an Extreme Scaling Workshop, *Scalasca* helped identify a critical performance issue that manifest at large scale with a version of the NEST application when it was importing 1.9 TiB of neuron and synapse data with HDF5 configured to use collective MPI file I/O. To avoid IBM XL C++ compiler instrumentation overhead, manual annotation of the relevant code regions was used to augment the instrumentation of OpenMP and MPI. Measurement of an execution with 16 OpenMP threads for 28,672 MPI ranks (458,752 threads in total) revealed a large imbalance in MPI File I/O which was mirrored in the following OpenMP parallel region. Instead of the expected MPI collective file I/O, much less efficient individual file I/O was found which originated from a mismatch between the import module's data structure and the HDF5 file objects [5]. After suitably modifying the import data structure to match the HDF5 file object, the imbalance was eliminated, and performance greatly improved [32].

The cost of MPI collective file writing for final simulation output of the CIAO application on *JUQUEEN* with 458,752 MPI ranks was also identified by *Scalasca* as a key performance limitation and motivation for *JUSITU* coupling in situ visualization to VisIt [16] as previously incorporated in psOpen and ZFS.

Exponentially growing memory requirements of ParFlow were located to originate from version of the HYPRE library used by its by preconditioner using memory allocation profiling [8]. Examination of *Scalasca* execution traces was key to determining optimal load-balancing of MPI and OpenMP computations, and associated workload distribution and loop scheduling strategies, to allow hp-fRG to scale effectively to use all of the *JUQUEEN* compute nodes with 1.8 million threads [31].

Conclusions

As the highly productive operation of the *JUQUEEN* Blue Gene/Q by Jülich Supercomputing Centre draws to a close in spring 2018, the High-Q Club documents 32 codes from a wide range of HPC application fields that demonstrated effective extreme-scale execution using its entire 458,752 cores (and often 1.8 million threads). Standard programming languages and MPI combined with multi-threading was sufficient, and provided a straightforward migration path for application developers which has also delivered performance and scalability benefits on diverse HPC computer systems (including K computer, Cray supercomputer systems and other clusters). Similar ease-of-use and reliability of well-established homogeneous Blue Gene/Q systems probably cannot be expected to be representative of the current and future generations of heterogeneous HPC systems, however, we believe it is a worthwhile goal.

Each of the High-Q Club member codes is quite distinct, encountering and resolving a variety of often unique impediments in scaling to the full *JUQUEEN* configuration. Code teams

themselves ultimately determine whether and how to address the considerable challenges, with the High-Q Club promoting successes. Engagement of experts from JSC involved close long-term collaborations in some cases to very little in others. Extreme scaling workshops provided a brief opportunity for particularly intense interaction and experimentation, which benefits many code teams.

Extreme scaling on *JUQUEEN* generally required adapting to the limited compute node memory, either via employing alternate communicator management optimised for large numbers of MPI ranks or effective exploitation of OpenMP multi-threading in a mixed-mode configuration. Often file I/O is not done in a scalable fashion, requiring many codes to forfeit I/O (and use synthetic or replicated simulation data) for their large-scale runs. Application codes which are leaner and less restrictive in their memory and I/O usage can be desirable as they can exploit more affordable systems, however, each code has its own requirements and constraints. High-Q Club member codes are those which were able to adapt, but many important codes may not be so fortunate. Despite these limitations, notable extreme-scale simulation capabilities were demonstrated and led to subsequent Big Blue Gene Weeks with prioritised production executions.

The High-Q Club was entirely neutral as to how application codes achieve qualifying scalability. JSC training and consultancy introduce and consider a variety of technologies and techniques (including novel programming models and languages) from which application developers themselves decide which to pursue based on their individual cost-benefit determination. Our assessment of the High-Q Club member code characteristics shows that incremental changes were convenient for a wide variety of codes. The absence of disruptive approaches can perhaps be explained by the additional effort required and associated technology immaturity at this time.

While focussing on scalability to the entire 28 racks of the *JUQUEEN* BG/Q installed at JSC was a natural choice, it is also rather arbitrary. Loosening the High-Q Club qualification criteria to accept scaling to 85% (or perhaps even 70%) of the entire *JUQUEEN* system could also have been justifiable and still offer value in distinguishing extreme-scaling codes. Additional credit may also have been appropriate for executions that are representative of production configurations including associated file I/O, and possibly other desirable aspects (such as node-level optimisation via vectorisation and core over-subscription). Greater differentiation between the High-Q Club member codes is also desirable, particularly for better insight into readiness for capability-mode production on current leadership systems and expected future exascale systems. These aspects will be re-considered for the successor to the High-Q Club.

Acknowledgements

We would like to thank the numerous application code-teams who participated in Extreme Scaling Workshops at JSC and contributors to the High-Q Club for generously sharing their experience, identifying performance and scalability inhibitors and effective solutions. We also recognise the invaluable assistance provided by the JSC Simulation Laboratories, Cross-Sectional Teams, system administrators, and *JUQUEEN* support staff.

References

1. Allock, W., Bacon, C., Bailey, A., Bair, R., et. al.: Blue Gene/Q: *Sequoia* and *Mira*. In: Vetter, J. (ed.) Contemporary High Performance Computing: From Petascale toward Exascale. pp. 225–282. Chapman & Hall/CRC (2013)

2. Attig, N., Docter, J., Frings, W., Grotendorst, J., Gutheil, I., Janetzko, F., Mextorf, O., Mohr, B., Stephan, M., Wolkersdorfer, K., Wollschläger, L., Krieg, S., Lippert, T.: Blue Gene/P: *JUGENE*. In: Vetter, J. (ed.) Contemporary High Performance Computing: From Petascale toward Exascale. pp. 153–188. Chapman & Hall/CRC (2013)

3. Brömmel, D., Frings, W., Wylie, B.: MAXI – Multi-system Application Extreme-scaling Imperative. In: Joubert, G.R., Leather, H., Parsons, M., Peters, F., Sawyer, M. (eds.) Parallel Computing: On the Road to Exascale. vol. 27, pp. 765–766. IOS Press (2016), DOI: 10.3233/978-1-61499-621-7-765

4. Brömmel, D., Frings, W., Wylie, B.J.N.: JUQUEEN Extreme Scaling Workshop 2015. Tech. Rep. FZJ-JSC-IB-2015-01 (2015), `http://juser.fz-juelich.de/record/188191`, accessed: 2018-03-21

5. Brömmel, D., Frings, W., Wylie, B.J.N.: Extreme-scaling Applications En Route to Exascale. In: Proceedings of the Exascale Applications and Software Conference 2016. pp. 1:1–1:10. EASC '16, ACM, New York, NY, USA (2016), DOI: 10.1145/2938615.2938616

6. Brömmel, D., Frings, W., Wylie, B.J.N.: JUQUEEN Extreme Scaling Workshop 2016. Tech. Rep. FZJ-JSC-IB-2016-01 (2016), `http://juser.fz-juelich.de/record/283461`, accessed: 2018-03-21

7. Brömmel, D., Frings, W., Wylie, B.J.N.: JUQUEEN Extreme Scaling Workshop 2017. Tech. Rep. FZJ-JSC-IB-2017-01 (2017), `http://juser.fz-juelich.de/record/828084`, accessed: 2018-03-21

8. Burstedde, C., Fonseca, J.A., Kollet, S.: Enhancing speed and scalability of the ParFlow simulation code. Computational Geosciences 22(1), 347–361 (2018), DOI: 10.1007/s10596-017-9696-2

9. Freche, J., Frings, W., Sutmann, G.: High-Throughput Parallel-I/O using *SIONlib* for Mesoscopic Particle Dynamics Simulations on Massively Parallel Computers. In: Chapman, B., Desprez, F., Joubert, G.R., Lichnewsky, A., Peters, F., Priol, T. (eds.) Parallel Computing: From Multicores and GPU's to Petascale. vol. 19, pp. 371–378. IOS Press (2010), DOI: 10.3233/978-1-60750-530-3-371

10. Frings, W., Mohr, B., Orth, B.: Report on the Jülich Blue Gene/L Scaling Workshop 2006. Tech. Rep. FZJ-ZAM-IB-2007-02, Jülich (2007), `http://juser.fz-juelich.de/record/55967`, accessed: 2018-03-21

11. Frings, W.: Efficient Task-Local I/O Operations of Massively Parallel Applications. Ph.D. thesis, RWTH Aachen University, Jülich (2016), `http://juser.fz-juelich.de/record/811621`, accessed: 2018-03-21

12. Frings, W., Ahn, D.H., LeGendre, M., Gamblin, T., de Supinski, B.R., Wolf, F.: Massively Parallel Loading. In: Proceedings of the 27th International ACM Conference on International Conference on Supercomputing. pp. 389–398. ICS '13, ACM, New York, NY, USA (2013), DOI: 10.1145/2464996.2465020

13. Frings, W., Wolf, F., Petkov, V.: Scalable Massively Parallel I/O to Task-local Files. In: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. pp. 17:1–17:11. SC '09, ACM, New York, NY, USA (2009), DOI: 10.1145/1654059.1654077

14. Gageik, M., Klioutchnikov, I., Olivier, H.: Mesh study for a direct numerical simulation of the transonic flow at $\mathrm{Re}_c = 500,000$ around a NACA 0012 airfoil. Computers & Fluids 122, 153–164 (2015), DOI: 10.1016/j.compfluid.2015.08.030

15. Geimer, M., Wolf, F., Wylie, B.J.N., Ábrahám, E., Becker, D., Mohr, B.: The Scalasca Performance Toolset Architecture. Concurrency and Computation: Practice and Experience 22(6), 702–719 (2010), DOI: 10.1002/cpe.1556

16. Göbbert, J.H., Bode, M., Wylie, B.J.N.: Extreme-Scale In Situ Visualization of Turbulent Flows on IBM Blue Gene/Q JUQUEEN. In: Taufer, M., Mohr, B., Kunkel, J.M. (eds.) High Performance Computing. pp. 45–55. Springer International Publishing, Cham (2016), DOI: 10.1007/978-3-319-46079-6_4

17. Göbbert, J., Gauding, M., Ansorge, C., Hentschel, B., Kuhlen, T., Pitsch, H.: Direct numerical simulation of fluid turbulence at extreme scale with psOpen. In: Gerhard, R., Leather, H., Parsons, M., Peters, F., Sawyer, M. (eds.) Parallel Computing: On the Road to Exascale. vol. 27, pp. 777–785. IOS Press (2016), DOI: 10.3233/978-1-61499-621-7-777

18. Hammer, N., Jamitzky, F., Satzger, H., et al.: Extreme scale-out SuperMUC phase 2 – lessons learned. In: Gerhard, R., Leather, H., Parsons, M., Peters, F., Sawyer, M. (eds.) Parallel Computing: On the Road to Exascale. vol. 27, pp. 827–836. IOS Press (2016), DOI: 10.3233/978-1-61499-621-7-827

19. Heinzeller, D., Duda, M.G., Kunstmann, H.: Towards convection-resolving, global atmospheric simulations with the Model for Prediction Across Scales (MPAS) v3.1: an extreme scaling experiment. Geoscientific Model Development 9(1), 77–110 (2016), DOI: 10.5194/gmd-9-77-2016

20. IBM Corporation: IBM System Blue Gene Solution Blue Gene/Q application development. `http://www.redbooks.ibm.com/`, accessed: 2018-03-21

21. Jordan, J., Ippen, T., Helias, M., Kitayama, I., Sato, M., Igarashi, J., Diesmann, M., Kunkel, S.: Extremely Scalable Spiking Neuronal Network Simulation Code: From Laptops to Exascale Computers. Frontiers in Neuroinformatics 12 (2018), DOI: 10.3389/fninf.2018.00002

22. Jülich Supercomputing Centre: The High-Q Club. `http://www.fz-juelich.de/ias/jsc/high-q-club`, accessed: 2018-03-21

23. Klawonn, A., Lanser, M., Rheinbach, O.: $\mathrm{FE}^2\mathrm{TI}$: Computational scale bridging for dual-phase steels. In: Joubert, G.R., Leather, H., Parsons, M., Peters, F., Sawyer, M. (eds.)

Parallel Computing: On the Road to Exascale. vol. 27, pp. 797–806. IOS Press (2016), DOI: 10.3233/978-1-61499-621-7-797

24. Knüpfer, A., Rössel, C., an Mey, D., Biersdorff, S., Diethelm, K., Eschweiler, D., Geimer, M., Gerndt, M., Lorenz, D., Malony, A., Nagel, W., Oleynik, Y., Philippen, P., Saviankou, P., Schmidl, D., Shende, S., Tschüter, R., Wagner, M., Wesarg, B., Wolf, F.: Score-P: A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir. In: Tools for High Performance Computing 2011. pp. 79–91. Springer, Berlin, Heidelberg (2012), DOI: 10.1007/978-3-642-31476-6_7

25. Mohr, B., Frings, W. (eds.): Jülich Blue Gene/P Porting, Tuning & Scaling Workshop 2008, Innovatives Supercomputing in Deutschland (InSiDE), vol. 6 (2008), http://inside.hlrs.de/_old/htm/Edition_02_08/article_28.html, accessed: 2018-03-21

26. Mohr, B., Frings, W.: Jülich Blue Gene/P Extreme Scaling Workshop 2009. Tech. Rep. FZJ-JSC-IB-2010-02, Jülich (2010), http://juser.fz-juelich.de/record/8924, accessed: 2018-03-21

27. Mohr, B., Frings, W.: Jülich Blue Gene/P Extreme Scaling Workshop 2010. Tech. Rep. FZJ-JSC-IB-2010-03, Jülich (2010), http://juser.fz-juelich.de/record/9600, accessed: 2018-03-21

28. Mohr, B., Frings, W.: Jülich Blue Gene/P Extreme Scaling Workshop 2011. Tech. Rep. FZJ-JSC-IB-2011-02, Jülich (2011), http://juser.fz-juelich.de/record/15866, accessed: 2018-03-21

29. Ovcharenko, A., Kumbhar, P., Hines, M., Cremonesi, F., Ewart, T., Yates, S., Schürmann, F., Delalondre, F.: Simulating morphologically detailed neuronal networks at extreme scale. In: Joubert, G.R., Leather, H., Parsons, M., Peters, F., Sawyer, M. (eds.) Parallel Computing: On the Road to Exascale. vol. 27, pp. 787–796. IOS Press (2016), DOI: 10.3233/978-1-61499-621-7-787

30. Qi, J., Jain, K., Klimach, H., Roller, S., Schürmann, F., Delalondre, F.: Performance evaluation of the LBM solver Musubi on various HPC architectures. In: Joubert, G.R., Leather, H., Parsons, M., Peters, F., Sawyer, M. (eds.) Parallel Computing: On the Road to Exascale. vol. 27, pp. 807–816. IOS Press (2016), DOI: 10.3233/978-1-61499-621-7-807

31. Rohe, D.: Hierarchical Parallelisation of Functional Renormalisation Group Calculations – hp-fRG. Computer Physics Communications 207, 160–169 (2015), DOI: 10.1016/j.cpc.2016.05.024

32. Schumann, T., Frings, W., Peyser, A., Schenck, W., Thust, K., Eppler, J.M.: Modeling the I/O behavior of the NEST simulator using a proxy. In: Conference Proceedings of the YIC GACM 2015 / ed.: Stefanie Elgeti; Jaan-Willem Simon. 3rd ECCOMAS Young Investigators Conference, Aachen (Germany), 20–23 July 2015, RWTH Aachen University (2015), http://juser.fz-juelich.de/record/202952, accessed: 2018-03-21

33. Springer, P., Ismail, A.E., Bientinesi, P.: A Scalable, Linear-Time Dynamic Cutoff Algorithm for Molecular Dynamics. In: Kunkel, J.M., Ludwig, T. (eds.) High Performance Computing. pp. 155–170. Springer International Publishing, Cham (2015), DOI: 10.1007/978-3-319-20119-1_12

34. Stephan, M., Doctor, J.: JUQUEEN: IBM Blue Gene/Q supercomputer system at the Jülich Supercomputing Centre. Journal of Large-Scale Research Facilities 1, 1–5 (2015), DOI: 10.17815/jlsrf-1-18