Extreme-scaling applications *en route* to exascale

Dirk Brömmel
Jülich Supercomputing Centre
Forschungszentrum Jülich
52425 Jülich, Germany
d.broemmel@fz-juelich.de

Wolfgang Frings
Jülich Supercomputing Centre
Forschungszentrum Jülich
52425 Jülich, Germany
w.frings@fz-juelich.de

Brian J. N. Wylie
Jülich Supercomputing Centre
Forschungszentrum Jülich
52425 Jülich, Germany
b.wylie@fz-juelich.de

ABSTRACT

Feedback from the previous year's very successful workshop motivated the organisation of a three-day workshop from 1 to 3 February 2016, during which the 28-rack *JUQUEEN* Blue Gene/Q system with 458 752 cores was reserved for over 50 hours. Eight international code teams were selected to use this opportunity to investigate and improve their application scalability, assisted by staff from JSC Simulation Laboratories and Cross-Sectional Teams.

Ultimately seven teams had codes successfully run on the full *JUQUEEN* system. Strong scalability demonstrated by Code_Saturne and Seven-League Hydro, both using 4 OpenMP threads for 16 MPI processes on each compute node for a total of 1 835 008 threads, qualify them for *High-Q Club* membership. Existing members CIAO and iFETI were able to show that they had additional solvers which also scaled acceptably. Furthermore, large-scale in-situ interactive visualisation was demonstrated with a CIAO simulation using 458 752 MPI processes running on 28 racks coupled via JUSITU to VisIt. The two adaptive mesh refinement utilities, ICI and p4est, showed that they could respectively scale to run with 458 752 and 971 504 MPI ranks, but both encountered problems loading large meshes. Parallel file I/O issues also hindered large-scale executions of PFLOTRAN. Poor performance of a NEST-import module which loaded and connected 1.9 TiB of neuron and synapse data was tracked down to an internal data-structure mismatch with the HDF5 file objects that prevented use of MPI collective file reading, which when rectified is expected to enable large-scale neuronal network simulations.

Comparative analysis is provided to the 25 codes in the *High-Q Club* at the start of 2016, which includes five codes that qualified from the previous workshop. Despite more mixed results, we learnt more about application file I/O limitations and inefficiencies which continue to be the primary inhibitor to large-scale simulations.

EASC '16, April 26–29, 2016, Stockholm, Sweden

ACM ISBN 978-1-4503-4122-6/16/04…$15.00

DOI: http://dx.doi.org/10.1145/2938615.2938616

CCS Concepts

•Computing methodologies → Massively parallel and high-performance simulations; •Software and its engineering → Ultra-large-scale systems; Software libraries and repositories;

Keywords

HPC application scalability, performance analysis, in-situ visualisation, *JUQUEEN*, IBM Blue Gene/Q, *High-Q Club*

1. INTRODUCTION

Exascale computer systems are expected to require one or two orders of magnitude more parallelism than the current leadership computer systems [1]. The current top ten computer systems each have more than 256k physical cores, and when exploiting hardware threading capabilities most of these can run over one million concurrent processes or threads. Jülich Supercomputing Centre (JSC) has almost a decade of experience with the range of IBM Blue Gene systems and in scaling HPC applications to use their considerable capabilities effectively in preparation for expected exascale systems.

From 1 to 3 February 2016, Jülich Supercomputing Centre organised the latest edition of its series of IBM Blue Gene Extreme Scaling Workshops. This series started with the 2006 "Blue Gene/L Scaling Workshop" [2] using the 8-rack (16 384 cores) *JUBL*, and then moved to *JUGENE* for the 2008 "Blue Gene/P Porting, Tuning & Scaling Workshop" [3] and dedicated "Extreme Scaling Workshops" in 2009 [4], 2010 [5] and 2011 [6]. 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.

Last year's workshop [7, 8] was the first for the *JUQUEEN* Blue Gene/Q [9, 10], and all seven participating application teams had within 24 hours successfully ran on all 28 racks (458 752 cores capable of running 1 835 008 processes or threads). With their results, five of the codes later joined the list of High-Q Club [11] codes and one existing member improved their scalability.

The *High-Q Club* is a collection of the highest scaling codes on *JUQUEEN* (Appendix A) and as such requires the codes to run on all 28 racks. Codes also have to demonstrate that they profit from each additional rack of *JUQUEEN* in reduced time to solution (*speed-up*) when strong scaling a fixed problem size or a tolerable increase in runtime when weak scaling progressively larger problems (*size-up*). Furthermore the application configurations should be beyond toy examples and we encourage use of all available hardware threads 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. Extreme-scaling workshops thus provide an opportunity for additional candidates to prove their scalability and qualify for membership, or – as was the case for some of the codes this and last year – improve on the scaling and efficiency that they had already achieved.

The *MAXI* mini-symposium [12] at the ParCo2015 conference enabled five High-Q Club members (including three of that year's workshop participants) to present and discuss their experience scaling their applications on a variety of the largest computing systems including *Hermit*, *K computer*, *Mira*, *Piz Daint*, *Sequoia*, *Stampede*, *SuperMUC*, *Tianhe-2* and *Titan*. Exchange of application extreme-scaling experience from these and other leadership HPC computer systems was also the focus of the full-day *aXXLs* workshop at the ISC-HPC conference [13].

Eight international application teams were selected for this year's three day workshop, and given dedicated use of the entire *JUQUEEN* system for a period of over 50 hours. Many of the teams' codes had thematic overlap with JSC Simulation Laboratories [1] or were part of an ongoing collaboration with one of the SimLabs for Fluids & Solids Engineering, Neuroscience, and Terrestrial Systems. While most of the application teams were experienced users of *JUQUEEN* (and other Blue Gene/Q systems), and had successfully scaled their application codes previously, additional time was scheduled and support from JSC Cross-Sectional Teams was available to do performance analyses and investigate optimisation opportunities.

The eight participating code teams were:

- CIAO *multiphysics, multiscale Navier-Stokes solver for turbulent reacting flows in complex geometries* [2] [14]
  RWTH-ITV Aachen University Inst. for Combustion Technology, Germany
- Code_Saturne *CFD based on the finite volume method to solve Navier-Stokes equations* [15] [3]
  STFC Daresbury Laboratory, UK
- ICI *simulation based on an implicit finite-element formulation including anisotropic mesh adaptation* [16] [4]
  Inst. de Calcul Intensif, École Centrale de Nantes, France
- iFETI *implicit solvers for finite-element problems in nonlinear hyperelasticity & plasticity* [17] [5]
  Universität zu Köln and Technische Universität Bergakademie Freiberg, Germany
- NEST-import *module to load neuron and synapse information into the NEST neural simulation tool* [18] [6]
  Blue Brain Project, Switzerland
- p4est *library for parallel adaptive mesh refinement and coarsening* [19] [7]
  Universität Bonn, Germany
- PFLOTRAN *subsurface flow and reactive transport* [20] [8]
  FZJ IEK6/JARA-HPC and AMPHOS^{21} Consulting S. L., Spain
- Seven-League Hydro (SLH) *astrophysical hydrodynamics with focus on stellar evolution* [21] [9]
  Heidelberger Inst. für Theoretische Studien, Germany

A summary of workshop results follows, looking at the employed programming models and languages, code scalability, tools at scale, and parallel I/O. Detailed results for each code are found in the reports provided by each of the participating teams [22]. These present and discuss more execution configurations and scaling results achieved by the application codes during the workshop.

2. SUMMARY OF RESULTS

Characteristics of the eight workshop codes are summarised in Table 1 and discussed first, then followed by comparison of scaling performance.

2.1 Parallel characteristics

Programming languages.

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. The left of Figure 1 shows a Venn set diagram of the programming language(s) used by the High-Q Club codes. It indicates that all three major programming languages are equally popular (without considering lines of code). Of the 8 workshop codes, three are exclusively written in Fortran, two only in C++, one is C, and the other two combine C with C++ or Fortran. Portability is apparently important, as hardware-specific coding extensions are generally avoided. The workshop codes all used IBM's XL compiler suite, whereas various High-Q Club application codes have preferred GCC or Clang compilers which offer support for more recent language standards. Most optimisations employed by the codes are therefore not specific to Blue Gene (or BG/Q) systems, but can also be exploited on other highly-parallel systems.

Parallelisation modes.

The four hardware threads per core of the Blue Gene/Q chip in conjunction with the limited amount of compute node memory suggest to make use of multi-threaded programming. It is therefore interesting to see whether this is indeed the preferred programming model and whether the available memory is an issue. The middle of Figure 1 shows a Venn set diagram of the programming models used by High-Q Club codes, revealing that mixed-mode programming does indeed dominate. Looking at the workshop codes in particular, all eight used MPI, which is almost ubiquitous for portable distributed-memory parallelisation. dynQCD is the only High-Q Club application employing lower-level

[1] http://www.fz-juelich.de/ias/jsc/simlabs
[2] http://www.itv.rwth-aachen.de/en/
[3] http://www.code-saturne.org/
[4] http://ici.ec-nantes.fr/
[5] part of the FE2TI suite formerly referred to as ex_nl/FE^2
[6] http://www.nest-initiative.org/
[7] http://www.p4est.org/
[8] http://www.pflotran.org/
[9] http://www.slh-code.org/

Table 1: 2016 Extreme Scaling Workshop 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.)

Code	Programming Compiler / Languages	Parallelisation Tasking	Threading	Concurrency	Scaling	File I/O
CIAO	XL: Ftn	MPI 16		16: 458 752	S	MPI-IO, HDF5
Code_Saturne	XL: C Ftn	MPI 16	OpenMP 4	64: 1 835 008	S	MPI-IO
ICI	XL: C++	MPI 16		16: 458 752	W	MPI-IO
iFETI	XL: C C++	MPI 32		32: 917 504	W	N/A
NEST-import	XL: C++	MPI 1	OpenMP 16	16: 458 752	S W	HDF5 (MPI-IO)
p4est	XL: C	MPI 32		32: 917 504	N/A	(MPI-IO)
PFLOTRAN	XL: F03	MPI 16		16: 131 072	S	HDF5 (SCORPIO)
SLH	XL: F95	MPI 16	OpenMP 4	64: 1 835 008	S	MPI-IO

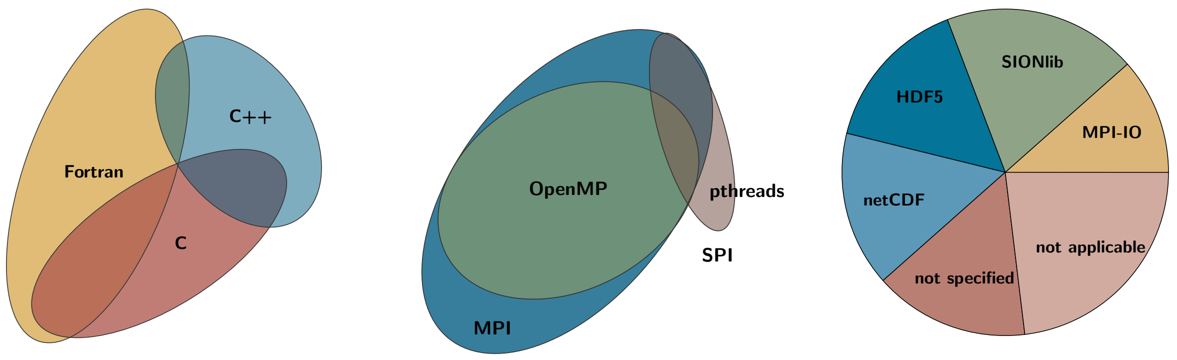


Figure 1: Venn set diagrams of programming languages (left) and parallel programming models (middle), plus a pie-chart showing file I/O (right) used by codes in the High-Q Club.

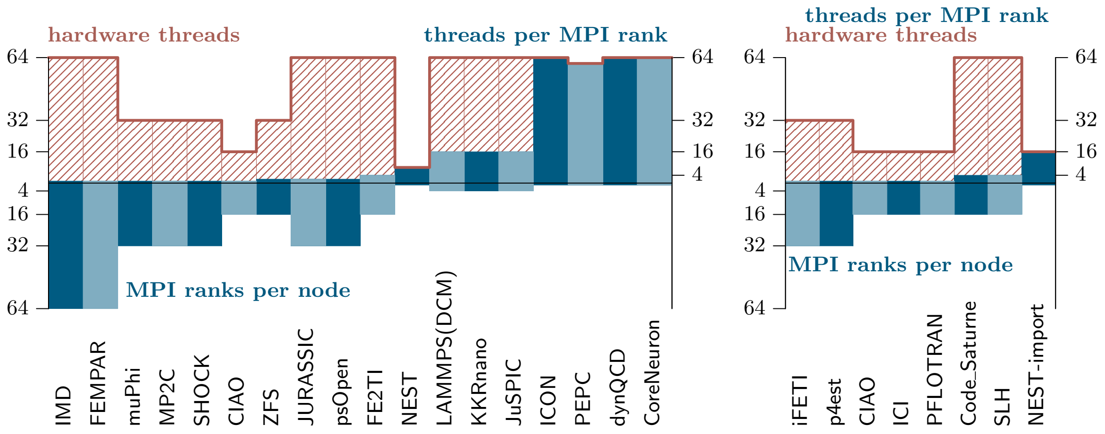


Figure 2: Chart showing the relation between the number of MPI ranks per node and threads per rank used by codes in the High-Q Club (left) and taking part in the workshop (right). The number of resulting hardware threads used on each compute node is shown in red.

machine-specific SPI for maximum performance. Five of the workshop codes exclusively used MPI for their scaling runs, both between and within compute nodes, accommodating to the restricted per-process memory. p4est has started testing the use of MPI-3 shared memory functionality, which is expected to save memory when running multiple MPI processes on each compute node. The remaining three workshop codes employ OpenMP multi-threading to exploit compute node shared memory in conjunction with MPI, as do the majority of High-Q Club applications. Instead of OpenMP, three of the High-Q Club applications prefer POSIX threading for additional control.

File I/O.

The right of Figure 1 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 used instead. Unfortunately, I/O usage by over 40% of High-Q Club was not provided with their submissions for membership indicating that file I/O has not yet received the required attention. Whereas half of the codes in the workshop can use MPI-I/O directly, only 10% of club members stated that they can do so. One quarter of the High-Q Club codes can use either (p)HDF5 or (p)NetCDF, despite their often disappointing performance as seen during the workshop. 20% of High-Q Club codes have migrated to using SIONlib [28] for effective parallel I/O, but only a couple of workshop codes have started this.

Concurrency.

Figure 2 shows the relation between the number of MPI ranks and threads per compute node where this information was available for High-Q Club (left) and workshop (right) codes. On either side of each diagram are the two extremes of exclusively using a distributed or shared memory approach within a node, so at most all 64 hardware threads on each CPU with either 64 processes or 64 threads. The charts show the number of processes on each compute node with downward bars while the associated number of threads is indicated with the bars extending upwards. Included in red hatching is the resulting number of hardware threads used by the code, i.e., the node concurrency. High-Q Club member codes often seem to benefit from using more hardware threads than physical cores and therefore favour this configuration. For others, such as NEST (and NEST-import), making full use of the available compute node memory for simulations is more important than full exploitation of processor cores and hardware threads. Using lower precision is occassionally exploited to reduce memory requirements and improve time to solution of large-scale simulations, however, larger PFLOTRAN simulations were prevented by its use of 32-bit (rather than 64-bit) integer indices. The two workshop codes Code_Saturne and SLH which qualified to join the High-Q Club similarly exploit all hardware threads by combining MPI+OpenMP, whereas none of the codes using purely MPI managed to this time.

2.2 Weak and strong scaling and performance

Here we show an overview of the scaling results achieved during the workshop. We compare *strong* (fixed total problem size) and *weak* (fixed problem size per process or thread) scaling, put in context of the scalability results from other codes in the High-Q Club.

Figures 3 and 4 show strong and weak scaling results of the workshop codes, including in grey results from a selection of High-Q Club codes. This indicates the spread in execution results and diverse scaling characteristics of the codes. The graphs show six of the workshop codes managing to run on the full *JUQUEEN* system, and most achieved good scalability. p4est scalability is not included as it had execution times of only a couple of seconds. Note that in many cases the graphs do not have a common baseline of a single rack since datasets sometimes did not fit available memory or no measurement was provided for 1024 compute nodes: for strong scaling an execution with a minimum of seven racks (one quarter of *JUQUEEN*) is therefore accepted for a baseline measurement, with perfect scaling assumed from a single rack to the baseline.

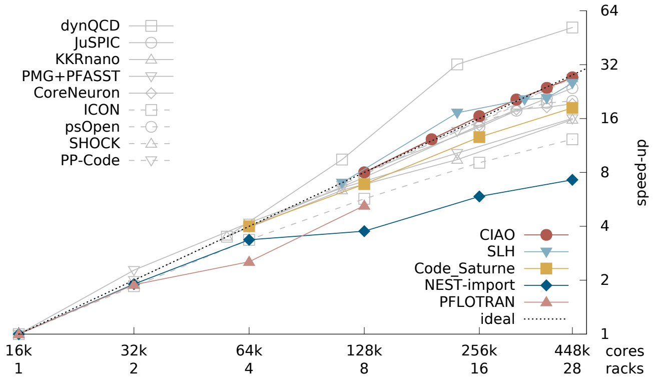


Figure 3: Strong scaling results of the workshop codes with results from existing High-Q Club members included in light grey.

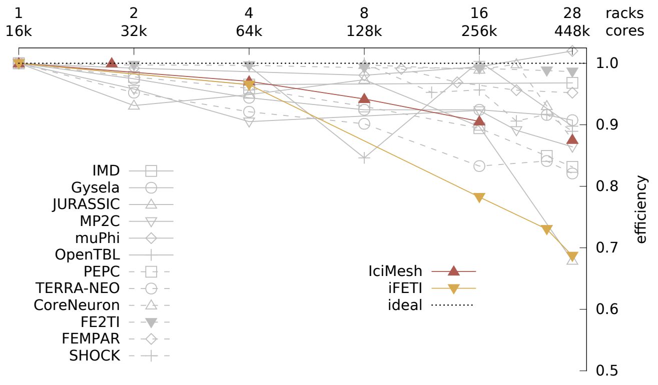
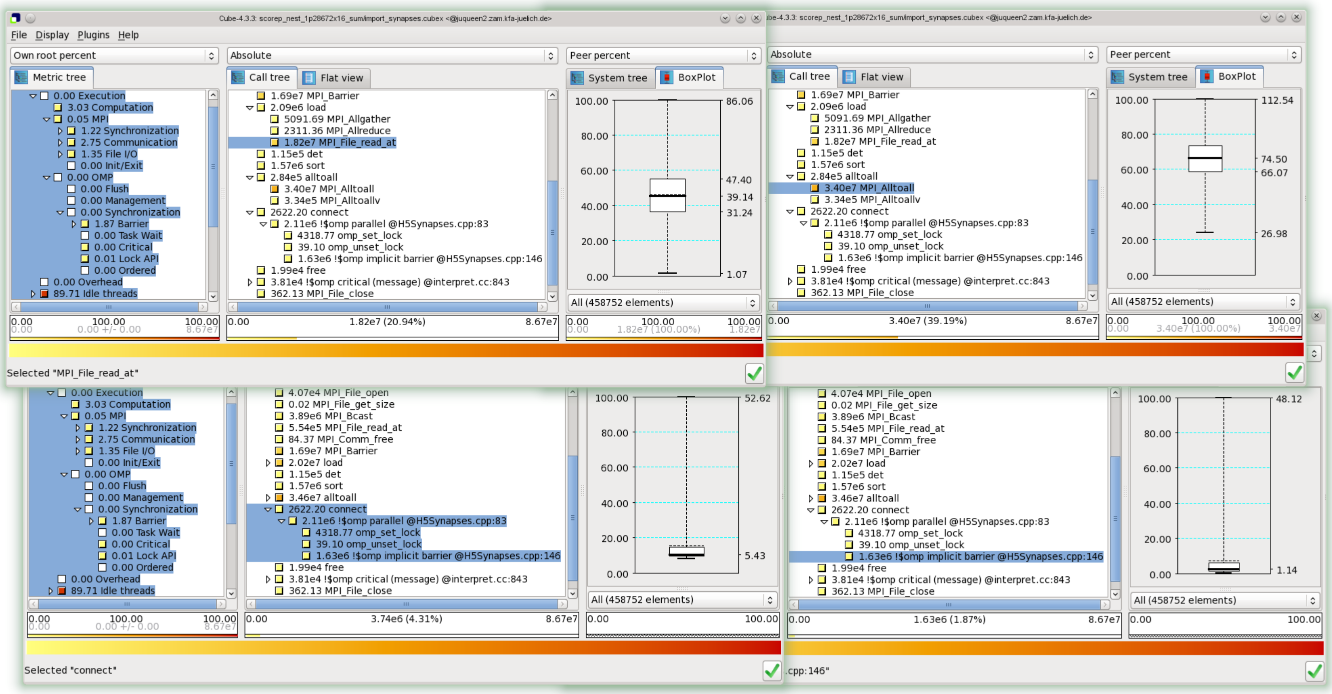


Figure 4: Weak scaling results of the workshop codes with results from existing High-Q Club members included in light grey.

In Figure 3 almost ideal strong-scaling speed-up of 27× on 28 racks is achieved by CIAO (in both configurations tested, like its previous High-Q Club entry), whereas Code_Saturne only shows a 19× speed-up. SLH speed-up is somewhere in between for the two problem sizes and run configurations measured. dynQCD stands-out with superlinear speed-up of 52×, due to its exceptional ability to exploit caches as problem size per thread decreases, whereas ICON achieved only a modest 12× speed-up.

PFLOTRAN managed to run on up to 8 racks before file I/O became prohibitive, but showed reasonable scalability of the solver for sufficiently large problem sizes. NEST-import ran successfully on all 28 racks, but only reached a scalability of 7× (probably largely due to its increasingly inefficient non-collective file reading and all-to-all redistribution).

In Figure 4, the weak scaling efficiency of IciMesh is a respectable 87% with 28 racks (though the largest measurement comes from a somewhat different problem configuration), whereas the new iFETI solver at only 69% scales considerably less well than their current High-Q Club FE2TI solver with 99%. muPhi was able to achieve 102% efficiency on 28 racks compared with a single rack, whereas JURASSIC only managed 68% efficiency due to excessive I/O for the reduced-size test case. Various codes show erratic



```
Cube-4.3.3: scorep_nest_1p28672x16_sum/import_synapses.cubex <@juqueen2.zam.kfa-juelich.de>
File  Display  Plugins  Help
Own root percent | Absolute | Peer percent
Metric tree | Call tree  Flat view | System tree  BoxPlot

Metric tree:
▽ 0.00 Execution
    3.03 Computation
  ▽ 0.05 MPI
    ▷ 1.22 Synchronization
    ▷ 2.75 Communication
    ▷ 1.35 File I/O
      0.00 Init/Exit
  ▽ 0.00 OMP
      0.00 Flush
      0.00 Management
    ▽ 0.00 Synchronization
      ▷ 1.87 Barrier
        0.00 Task Wait
        0.00 Critical
        0.01 Lock API
        0.00 Ordered
  0.00 Overhead
▷ 89.71 Idle threads
0.00   100.00   100.00
0.00   0.00 +/- 0.00   8.67e7

Call tree:
  1.69e7 MPI_Barrier
▽ 2.09e6 load
    5091.69 MPI_Allgather
    2311.36 MPI_Allreduce
    1.82e7 MPI_File_read_at
  1.15e5 det
  1.57e6 sort
▽ 2.84e5 alltoall
    3.40e7 MPI_Alltoall
    3.34e5 MPI_Alltoallv
▽ 2622.20 connect
  ▽ 2.11e6 !$omp parallel @H5Synapses.cpp:83
      4318.77 omp_set_lock
      39.10 omp_unset_lock
      1.63e6 !$omp implicit barrier @H5Synapses.cpp:146
  1.99e4 free
▷ 3.81e4 !$omp critical (message) @interpret.cc:843
  362.13 MPI_File_close
0.00   1.82e7 (20.94%)   8.67e7

BoxPlot: 86.06  47.40  39.14  31.24  1.07
All (458752 elements)
0.00   100.00   100.00
0.00   1.82e7 (100.00%)   1.82e7

Selected "MPI_File_read_at"
```

```
Cube-4.3.3: scorep_nest_1p28672x16_sum/import_synapses.cubex <@juqueen2.zam.kfa-juelich.de>
Absolute | Peer percent
Call tree  Flat view | System tree  BoxPlot

Call tree:
  1.69e7 MPI_Barrier
▽ 2.09e6 load
    5091.69 MPI_Allgather
    2311.36 MPI_Allreduce
    1.82e7 MPI_File_read_at
  1.15e5 det
  1.57e6 sort
▽ 2.84e5 alltoall
    3.40e7 MPI_Alltoall
    3.34e5 MPI_Alltoallv
▽ 2622.20 connect
  ▽ 2.11e6 !$omp parallel @H5Synapses.cpp:83
      4318.77 omp_set_lock
      39.10 omp_unset_lock
      1.63e6 !$omp implicit barrier @H5Synapses.cpp:146
  1.99e4 free
▷ 3.81e4 !$omp critical (message) @interpret.cc:843
  362.13 MPI_File_close
0.00   3.40e7 (39.19%)   8.67e7

BoxPlot: 112.54  74.50  66.07  26.98
All (458752 elements)
0.00   100.00   100.00
0.00   3.40e7 (100.00%)   3.40e7
```

```
Metric tree:
▽ 0.00 Execution
    3.03 Computation
  ▽ 0.05 MPI
    ▷ 1.22 Synchronization
    ▷ 2.75 Communication
    ▷ 1.35 File I/O
      0.00 Init/Exit
  ▽ 0.00 OMP
      0.00 Flush
      0.00 Management
    ▽ 0.00 Synchronization
      ▷ 1.87 Barrier
        0.00 Task Wait
        0.00 Critical
        0.01 Lock API
        0.00 Ordered
  0.00 Overhead
▷ 89.71 Idle threads
0.00   100.00   100.00
0.00   0.00 +/- 0.00   8.67e7

Call tree:
  4.07e4 MPI_File_open
  0.02 MPI_File_get_size
  3.89e6 MPI_Bcast
  5.54e5 MPI_File_read_at
  84.37 MPI_Comm_free
  1.69e7 MPI_Barrier
▷ 2.02e7 load
  1.15e5 det
  1.57e6 sort
▷ 3.46e7 alltoall
▽ 2622.20 connect
  ▽ 2.11e6 !$omp parallel @H5Synapses.cpp:83
      4318.77 omp_set_lock
      39.10 omp_unset_lock
      1.63e6 !$omp implicit barrier @H5Synapses.cpp:146
  1.99e4 free
▷ 3.81e4 !$omp critical (message) @interpret.cc:843
  362.13 MPI_File_close
0.00   3.74e6 (4.31%)   8.67e7

BoxPlot: 52.62  5.43
All (458752 elements)
0.00   100.00

Selected "connect"
```

```
Call tree:
  4.07e4 MPI_File_open
  0.02 MPI_File_get_size
  3.89e6 MPI_Bcast
  5.54e5 MPI_File_read_at
  84.37 MPI_Comm_free
  1.69e7 MPI_Barrier
▷ 2.02e7 load
  1.15e5 det
  1.57e6 sort
▷ 3.46e7 alltoall
▽ 2622.20 connect
  ▽ 2.11e6 !$omp parallel @H5Synapses.cpp:83
      4318.77 omp_set_lock
      39.10 omp_unset_lock
      1.63e6 !$omp implicit barrier @H5Synapses.cpp:146
  1.99e4 free
▷ 3.81e4 !$omp critical (message) @interpret.cc:843
  362.13 MPI_File_close
0.00   1.63e6 (1.87%)   8.67e7

BoxPlot: 48.12  1.14
All (458752 elements)
0.00   100.00

.cpp:146"
```

Figure 5: Scalasca analysis report explorer views of `import_synapses` extract of a Score-P measurement profile from NEST-import execution on all 28 racks of *JUQUEEN* (28 672 MPI ranks each with 16 OpenMP threads). The upper-left view shows 21% of time is in `MPI_File_read_at`, when loading 1.9 TiB of HDF5 neuron and synapse data in six parts (174 calls). Reading time by rank varies from 1.07 to 86.06 seconds, resulting in up to 85 seconds of waiting within the subsequent synchronising collective `MPI_Alltoall` (upper-right), as part of the 120 seconds each rank takes to load and redistribute the data. The lower views show time in the subsequent `connect` step ranging from 4.14 to 52.62 seconds, where some threads wait up to 48 seconds in the implicit barrier closing the OpenMP parallel region.

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

2.3 Tools at scale

The community-developed Score-P instrumentation and measurement infrastructure [25] employed by Scalasca [26] was used to profile NEST-import. Since automatic compiler instrumentation of all routines in C++ applications typically results in prohibitive measurement overheads, instead manual annotation of the relevant code regions was used to augment the instrumentation of OpenMP and MPI. An example profile from an execution on all 28-racks of *JUQUEEN* (28 672 MPI ranks each with 16 OpenMP threads, for 458 752 in total) is shown in Figure 5. Substantial time and imbalance in MPI file I/O and the main OpenMP parallel region are evident. The former was subsequently identified as originating from a mismatch between the module's data structure and the HDF5 file objects which resulted in use of individual MPI file I/O.

While Score-P does not yet support POSIX file I/O or provide measurements of the number of bytes read or written, Darshan [27] was available for this. Darshan problems with Fortran codes using MPI on *JUQUEEN* required a linking workaround, which worked for CIAO but not for PFLOTRAN, and the C++ codes ICI and iFETI also reported issues (whereas NEST-import was successful). A revised set of Darshan linking wrappers have been developed which are expected to resolve these problems.

2.4 Parallel I/O

File I/O performance is a critical scalability constraint for many large-scale parallel applications which need to read and write huge datasets or open a large number of files. Half of the workshop codes used MPI file I/O directly, whereas others (e.g. NEST-import) use it indirectly via HDF5. Additionally, p4est can use MPI file I/O but did not for the runs during the workshop.

Code_Saturne used MPI collective file I/O effectively to read 618 GiB of mesh input data, however, writing of simulation output was disabled as this was a known bottleneck. The NEST-import module read 1.9 TiB of HDF5 neuron and synapse data but only attained a fraction of the GPFS filesystem bandwidth. Internal data structures are currently being adapted to be able to exploit MPI collective file reading, which is expected to significantly out-perform the current MPI individual/independent file reading and should enable large-scale data-driven neuronal network simulations in future. SLH compared writing 264 GiB of astrophysical simulation output using MPI-IO to a single file or using C stdio to separate files for each MPI process. Writing many process-local files is impractical as it requires all of the files to be created on disk in advance, to avoid filesystem meta-data issues, and an expensive post-processing to aggregate the output into a single file for subsequent use.

While IciMesh was able to generate an adapted mesh with over 100 billion elements using 458 752 MPI ranks on all 28 racks of *JUQUEEN*, the associated solver IciSolve was limited to 65 536 MPI ranks due to problems uncovered with their use of MPI individual/independent file I/O to read their 1.7 TiB mesh files. The parallel adaptive mesh refinement code p4est demonstrated generation, refinement and partitioning, managing in-core meshes with up to 940 billion elements in 21 seconds using 917 504 MPI ranks on 28 *JUQUEEN* racks. In a test case with a larger coarse mesh, memory was the limiting factor for broadcasting data from a single MPI rank to the others. HDF5 file I/O also presented an insurmountable scalability impediment for PFLOTRAN, particularly for larger problem sizes and with more than ten thousand MPI processes.

All of the above is evidence that file I/O is critical and needs the appropriate attention by the programmer and the right methods to perform I/O. At JSC, SIONlib [28] has been developed to address file I/O scalability limitations. It has been used effectively by three High-Q codes (KKRnano, MP2C and muPhi) and several other applications are currently migrating to adopt it (e.g., NEST and SLH). SIONlib is highly optimized for task-local I/O patterns on massively parallel architectures. For example, with SIONlib on *JUQUEEN* applications are able to achieve an I/O bandwidth of more than 100 GiB/s using a virtual shared file container and techniques for accessing the container efficiently. The latter involve file system block alignment and the separation of I/O-streams by creating one physical file per I/O-bridge node of the Blue Gene/Q system [29].

Apart from specifying a GPFS filesystem type, additional hints for MPI-IO were not investigated by these applications. CIAO experimented with various MPI-IO (ROMIO) hints, but did not observe any benefit when writing single 9 TiB files with MPI file I/O. Whether the parameters have no effect due to the MPI implementation on *JUQUEEN* is still under investigation, but it is well known that reading and writing single shared files fails to exploit the available filesystem bandwidth.

2.5 In-situ visualisation

Large amounts of application file I/O can be avoided via methods for in-situ visualization of large simulations developed within JARA-HPC [10] to allow easy and fast control of large simulations and reduce the amount of data which needs to be stored [23]. More precisely, the coupling layer JUSITU [11] has been implemented and coupled to CIAO and VisIt [24]. A compressible, turbulent channel (Reynolds number 13 760) containing small droplets (described by an Eulerian indicator function) was the chosen test case for the workshop. Figure 6 shows a screenshot of the full 28-rack *JUQUEEN* run. It visualizes the simulation state after running for 9 units of simulation time. Beside the *VisIt* overview window (on the left), *Window 1* showing a histogram of the pressure, *Window 2* visualizing the turbulent kinetic energy within the channel, the *Compute engines* window giving information about the simulation on *JUQUEEN*, and the *Simulation* window allowing to give instructions to the simulation are visible.

[10]http://www.jara.org/en/research/jara-hpc/
[11]https://gitlab.version.fz-juelich.de/vis/jusitu

2.6 Miscellany

The workshop participants were all associated with active projects on *JUQUEEN*, which allowed them to prepare their codes and datasets in advance of the workshop. The most successful teams were very familiar with their codes, able to build and run them in different configurations, and had also prepared performance profiles and analysis reports for examination.

Many of the workshop participants' codes used popular libraries such as HDF5 and PETSc. This facilitated discussion and exchange of experience, despite apparent favouring of different library versions and configurations (which were often customised rather than relying on the system installed versions).

All of the workshop accounts were part of the training group sharing the provided compute-time allocation and file-system quota. When one team exceeded the 200 TiB group quota, by forgetting to remove test files at the end of their jobs, this temporarily resulted in compilation and execution failures for the other participants. Over the 50 hours of the workshop, 3 500 TiB was read and 124 TiB was written in total between applications and the I/O nodes, with the largest jobs reading 330 TiB and writing 15 TiB respectively. Maximum bandwidths recorded over one minute intervals was 700 GiB/s for reading and 18 GiB/s for writing. Despite the considerable load on the GPFS file-system from large-scale parallel jobs doing file I/O during the workshop, the only issue encountered was variable performance.

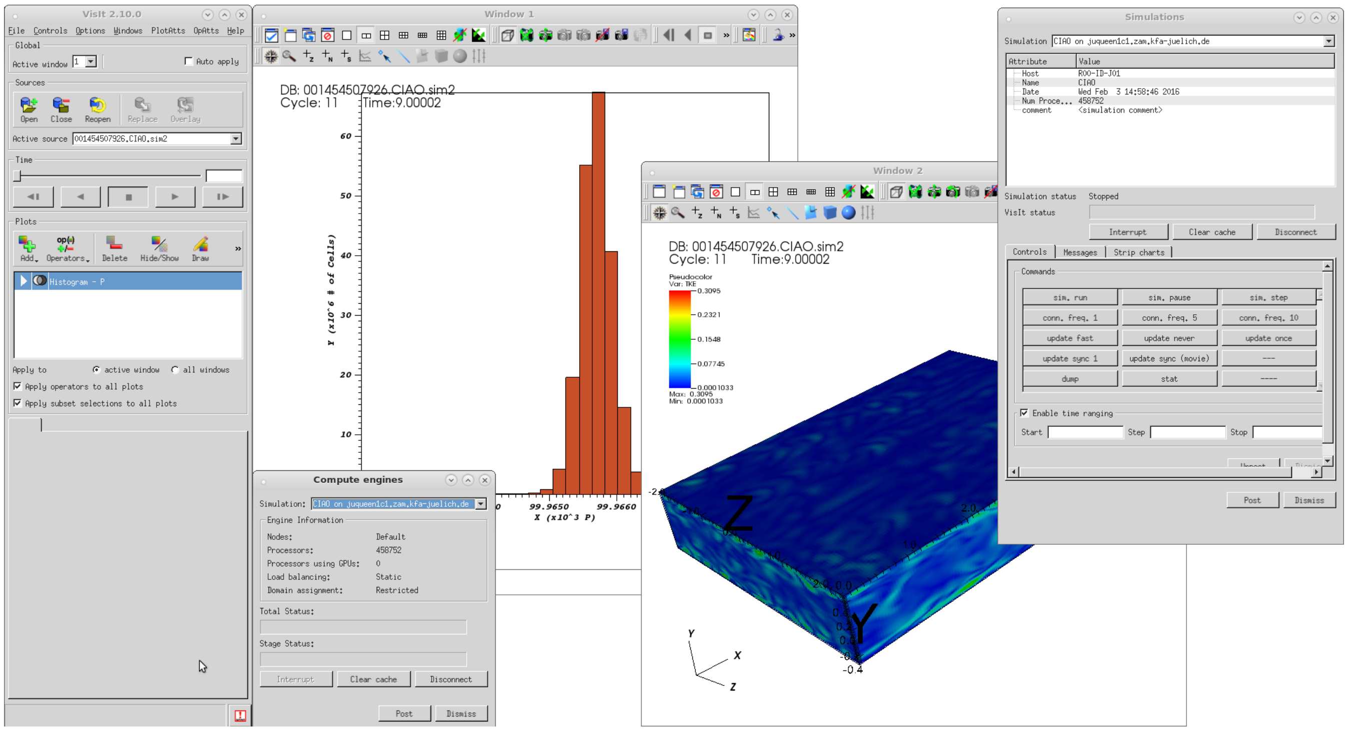
LLview was invaluable for monitoring the current use of *JUQUEEN* (Figure 7), additionally showing job energy consumption and file I/O performance, while real-time accounting of jobs during the workshop facilitated tracking of resource usage by each participant. LoadLeveler job scheduling quirks were avoided by deft intervention from sysadmins closely monitoring *JUQUEEN* during the workshop (day and night). 43 jobs were run on all 28 racks, 15 on 24 racks, 13 on 20 racks, as well as 90 jobs with 16 racks, consuming a total of 15.4 million core-hours.

At the start of the workshop, two defective nodeboards limited initial scaling tests to 24 racks for the first 24 hours, but after their replacement 28-rack jobs were quickly tested by most teams. No additional hardware failures were encountered during the workshop.

Given the demanding nature of the workshop, considerable flexibility is essential, both from participants and their test cases and regarding scheduling of breaks, sessions and system partitions. Physical presence of at least one member of each code team in the classroom for the workshop is therefore necessary for rapid communication. This proved especially the case given the mix of proven highly-scaling codes (often needing the entire compute resource) and the need for smaller-scale tests to investigate problems and verify solutions. The physical configuration of *JUQUEEN* makes the workshop particularly unsuited to small-scale tests with long execution times.

3. CONCLUSIONS

The variety of codes and participants, from different but often related subject areas as well as different institutions and countries, combined with similarly diverse support staff, contributes to the intense yet highly productive nature of JSC Extreme Scaling Workshops which have the goal of



Simulation CIAO on juqueen1c1.zam.kfa-juelich.de

Attribute	Value
Host	R00-ID-J01
Name	CIAO
Date	Wed Feb 3 14:58:46 2016
Num Proce...	458752
comment	<simulation comment>

Simulation status Stopped

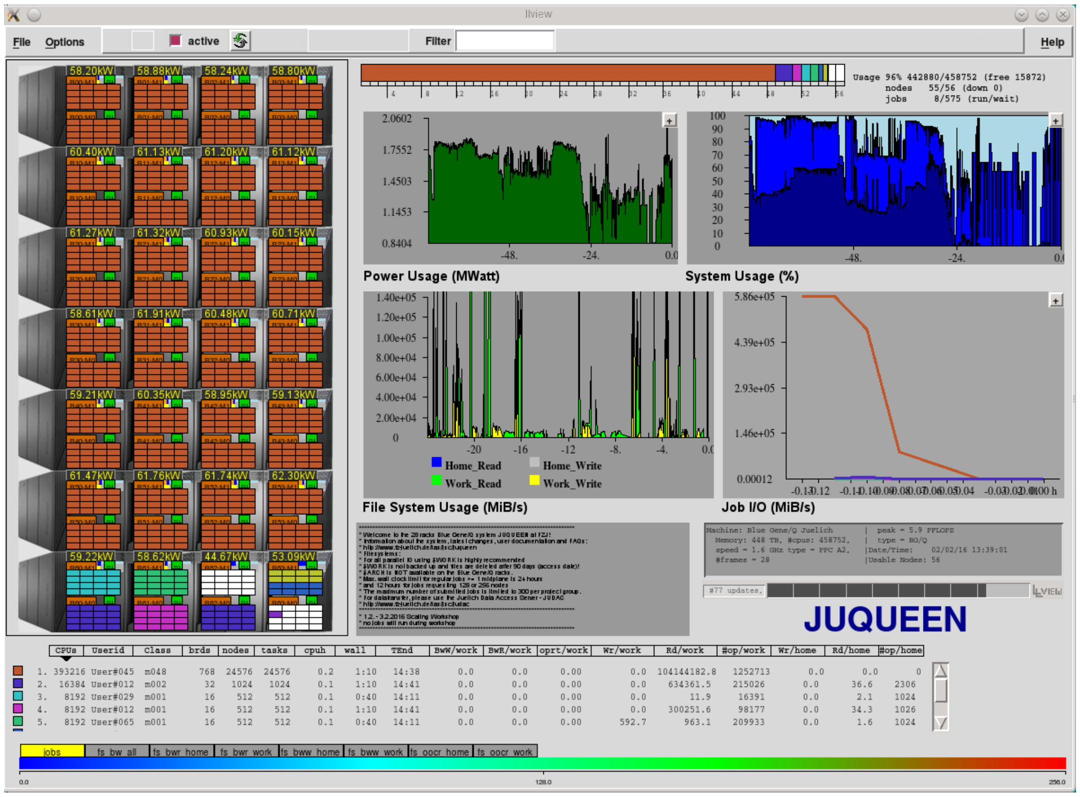
Compute engines

Simulation: CIAO on juqueen1c1.zam.kfa-juelich.de

Engine Information

Nodes:	Default
Processors:	458752
Processors using GPUs:	0
Load balancing:	Static
Domain assignment:	Restricted

Figure 6: VisIt interactive visualization client coupled via JUSITU to CIAO fluid dynamics simulation running on *JUQUEEN* with 458 752 MPI processes. (Image credit: Jens Henrik Göbbert)



```
Usage 96% 442880/458752 (free 15872)
nodes      55/56 (down 0)
jobs       8/575 (run/wait)
```

```
* Welcome to the 28 racks Blue Gene/Q system JUQUEEN at FZJ!
* Information about the system, latest changes, user documentation and FAQs:
* http://www.fz-juelich.de/ias/jsc/juqueen
* Filesystems:
* For all parallel IO using $WORK is highly recommended
* $WORK is not backed up and files are deleted after 90 days (access date)!
* $ARCH is NOT available on the Blue Gene/Q racks.
* Max. wall clock limit for regular jobs >= 1 midplane is 24 hours
* and 12 hours for jobs requesting 128 or 256 nodes
* The maximum number of submitted jobs is limited to 300 per project group.
* For datatransfer, please use the Juelich Data Access Server - JUDAC
* http://www.fz-juelich.de/ias/jsc/judac
* 1.2.-3.2.2016 Scaling Workshop
* no jobs will run during workshop
```

```
Machine: Blue Gene/Q Juelich          |  peak = 5.9 PFLOPS
  Memory: 448 TB, #cpus: 458752,      |  type = BG/Q
  speed = 1.6 GHz type = PPC A2,      |Date/Time:    02/02/16 13:39:01
  #frames = 28                        |Usable Nodes: 56
```

		CPUs	Userid	Class	brds	nodes	tasks	cpuh	wall	TEnd	BwW/work	BwR/work	oprt/work	Wr/work	Rd/work	#op/work	Wr/home	Rd/home	#op/home
🟧	1.	393216	User#045	m048	768	24576	24576	0.2	1:10	14:38	0.0	0.0	0.00	0.0	104144182.8	1252713	0.0	0.0	0
🟦	2.	16384	User#012	m002	32	1024	1024	0.1	1:10	14:41	0.0	0.0	0.00	0.0	634361.5	215026	0.0	36.6	2306
🟦	3.	8192	User#029	m001	16	512	512	0.1	0:40	14:11	0.0	0.0	0.00	0.0	11.9	16391	0.0	2.1	1024
🟪	4.	8192	User#012	m001	16	512	512	0.1	1:10	14:41	0.0	0.0	0.00	0.0	300251.6	98177	0.0	34.3	1026
🟩	5.	8192	User#065	m001	16	512	512	0.1	0:40	14:11	0.0	0.0	0.00	592.7	963.1	209933	0.0	1.6	1024

Figure 7: LLview monitoring of *JUQUEEN* during workshop during an execution of CIAO on 24 racks (and several smaller jobs), with charts of the CIAO job file I/O bandwidth (middle right), file-system usage of the previous 24 hours (centre), and three-day histories of power usage and job-size mix (centre and upper right).

proving and improving the ability of applications to exploit current and forthcoming extremely large and complex high-performance computer systems.

The High-Q Club documents codes from a wide range of HPC application fields demonstrating effective extreme-scale execution on the *JUQUEEN* Blue Gene/Q, generally with only modest tuning effort. This year's Extreme Scaling Workshop identified two additional codes which qualify for membership, Code_Saturne and Seven-League Hydro, both scaling to 1.8 million threads. Standard programming languages and MPI combined with multi-threading are sufficient, and provide a straightforward migration path for application developers which has also delivered performance and scalability benefits on diverse HPC computer systems (including K computer, Cray 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 target.

The High-Q Club reflects how users of *JUQUEEN* manage to achieve their goals, irrespective of other approaches which may be available. Several alternatives to MPI and OpenMP are actively researched at JSC and also promoted or available on *JUQUEEN*, however, the High-Q Club can only document what has been used successfully without knowing reasons for possible failures, readiness of alternatives, or any work in progress on the respective codes. Similarly, it does not allow to identify particularly efficient codes in terms of peak performance or a better time to solution for a given problem. Only simple comparative metrics like speed-up or efficiency on JUQUEEN can be assessed for several reasons: we rely on measurements that users provide, and they solve very different problems from many different scientific fields.

Whereas previously application codes have tended to avoid doing file I/O in their scaling runs, disabling outputs and using replicated or synthesised input data, this year's workshop participants were encouraged to thoroughly analyse their applications' I/O requirements and address limitations. Performance analysis of the NEST-import module for the neuronal network simulator identified individual MPI file reads resulting from an internal data-structure definition mismatch with the HDF5 file objects that prevented use of much more efficient MPI collective file reads. Also in-situ interactive visualisation of a CIAO CFD simulation with 458 752 MPI processes running on 28 racks demonstrated a viable alternative to file I/O that is expected to be an essential capability for the coming generation of simulations and expected exascale systems.

Acknowledgments

We would like to thank the workshop participants from the eight code-teams for openly sharing their knowledge and experience: Mathis Bode, Guido Deissmann, Fabien Delalondre, Abhishek Deshmukh, Hugues Digonnet, Hedieh Ebrahimi, Philip Edelmann, Johannes Holke, Alex Klawonn, Martin Lanser, Charles Moulinec, Oliver Rheinbach, and Till Schumann. We also recognise the invaluable assistance provided by JSC staff: Jutta Doctor, Jens Henrik Göbbert, Klaus Görgen, Inge Gutheil, Andreas Lintermann, Sebastian Lührs, Alex Peyser, Wendy Sharples, Michael Stephan, Kay Thust, and Ilya Zhukov.

4. REFERENCES

[1] Rich Stevens, Andrew White, et al., (eds). *Architectures and Technology for Extreme Scale Computing*, Advanced Scientific Computing Research: Scientific Grand Challenges Workshop Series. US Department of Energy Office of Science, Office of Advanced Scientific Computing Research, Dec. 2009.

[2] Wolfgang Frings, Marc-André Hermanns, Bernd Mohr & Boris Orth (eds), *Jülich Blue Gene/L Scaling Workshop 2006*, Tech. Report FZJ-ZAM-IB-2007-02, Forschungszentrum Jülich, 2007. http://juser.fz-juelich.de/record/55967

[3] Bernd Mohr & Wolfgang Frings, *Jülich Blue Gene/P Porting, Tuning & Scaling Workshop 2008*, Innovatives Supercomputing in Deutschland, inSiDE 6(2), 2008.

[4] Bernd Mohr & Wolfgang Frings (eds), *Jülich Blue Gene/P Extreme Scaling Workshop 2009*, Technical Report FZJ-JSC-IB-2010-02, Forschungszentrum Jülich, Feb. 2010. http://juser.fz-juelich.de/record/8924

[5] Bernd Mohr & Wolfgang Frings (eds), *Jülich Blue Gene/P Extreme Scaling Workshop 2010*, Technical Report FZJ-JSC-IB-2010-03, Forschungszentrum Jülich, May 2010. http://juser.fz-juelich.de/record/9600

[6] Bernd Mohr & Wolfgang Frings (eds), *Jülich Blue Gene/P Extreme Scaling Workshop 2011*, Technical Report FZJ-JSC-IB-2011-02, Forschungszentrum Jülich, Apr. 2011. http://juser.fz-juelich.de/record/15866

[7] Dirk Brömmel, Wolfgang Frings & Brian J. N. Wylie (eds), *JUQUEEN Extreme Scaling Workshop 2015*, Technical Report FZJ-JSC-IB-2015-01, Forschungszentrum Jülich, Feb. 2015. http://juser.fz-juelich.de/record/188191

[8] Dirk Brömmel, Wolfgang Frings & Brian J. N. Wylie, *Extreme-scaling applications 24/7 on JUQUEEN Blue Gene/Q*, in Proc. Int'l Conf. on Parallel Computing (ParCo, Edinburgh, Scotland), Advances in Parallel Computing vol. 27, 817–826, IOS Press, Sept. 2015. http://juser.fz-juelich.de/record/809001

[9] Michael Stephan & Jutta Doctor, *JUQUEEN: IBM Blue Gene/Q supercomputer system at the Jülich Supercomputing Centre*, Journal of Large-Scale Research Facilities **1** A1 (2015). http://dx.doi.org/10.17815/jlsrf-1-18

[10] JUQUEEN: Jülich Blue Gene/Q. http://www.fz-juelich.de/ias/jsc/juqueen

[11] The High-Q Club at JSC. http://www.fz-juelich.de/ias/jsc/high-q-club

[12] Dirk Brömmel, Wolfgang Frings & Brian J. N. Wylie, *MAXI – Multi-system Application Extreme-scaling Imperative*, Mini-symposium at Int'l Conf. on Parallel Computing (ParCo, Edinburgh, UK), Advances in Parallel Computing vol. 27, 763–846, IOS Press, Sept. 2015. http://www.fz-juelich.de/ias/jsc/MAXI

[13] Dirk Brömmel, Wolfgang Frings & Brian J. N. Wylie, *Application Extreme-scaling Experience of Leading Supercomputing Centres*, Workshop at 30th Int'l Conf. ISC High Performance (Frankfurt, Germany), July 2015. http://www.fz-juelich.de/ias/jsc/aXXLs

[14] Mathis Bode, Jens Henrik Göbbert, Heinz Pitsch, *High-fidelity multiphase simulations and in-situ visualization using CIAO*, Poster at 8th NIC Symposium, Forschungszentrum Jülich, Feb. 2016. http://juser.fz-juelich.de/record/809475

[15] Frédéric Archambeau, Namane Méchitoua & Marc Sakiz, *Code_Saturne: A finite volume code for the computation of turbulent incompressible flows — Industrial applications*, Int'l J. Finite Volumes **1**, Feb. 2004. https://hal.archives-ouvertes.fr/hal-01115371

[16] Hugues Digonnet, Luisa Silva & Thierry Coupez, *Using full Tier0 supercomputers for finite element computations with adaptive meshing*, in Proc. 4th Int'l Conf. on Parallel, Distributed, Grid and Cloud Computing for Engineering, P. Iványi & B.H.V. Topping (eds), Civil-Comp Press, Stirlingshire, UK, Paper 13, 2015.

[17] Axel Klawonn, Martin Lanser & Oliver Rheinbach, *Toward extremely scalable nonlinear domain decomposition methods for elliptic partial differential equations*, SIAM J. Scientific Computing **37**(6), C667–696, Dec. 2015. http://dx.doi.org/10.1137/140997907

[18] Hans Ekkehard Plesser, Markus Diesmann, Marc-Oliver Gewaltig & Abigail Morrison, *NEST: the Neural Simulation Tool*, Springer Encyclopedia of Computational Neuroscience, 1849–1852, March 2015. http://dx.doi.org/10.1007/978-1-4614-6675-8_258

[19] Carsten Burstedde, Lucas C. Wilcox & Omar Ghattas, *p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees*, SIAM J. Scientific Computing **33**(3):1103–1133, 2011. http://dx.doi.org/10.1137/100791634

[20] Glenn E. Hammond, Peter C. Lichtner & Richard T. Mills, *Evaluating the performance of parallel subsurface simulators: An illustrative example with PFLOTRAN*, Water Resources Research **50**, 2014. http://dx.doi.org/10.1002/2012WR013483

[21] Fabian Miczek, Friedrich K. Röpke & Philip V. F. Edelmann, *New numerical solver for flows at various Mach numbers*, Astronomy and Astrophysics **576**:A50, Feb. 2015. http://dx.doi.org/10.1051/0004-6361/201425059

[22] Dirk Brömmel, Wolfgang Frings & Brian J. N. Wylie (eds), *JUQUEEN Extreme Scaling Workshop 2016*, Tech. Report FZJ-JSC-IB-2016-01, Forschungszentrum Jülich, Feb. 2016. http://juser.fz-juelich.de/record/283461

[23] Jens Henrik Göbbert, Mathis Bode & Brian J. N. Wylie, *Extreme-scale in-situ visualization of turbulent flows on IBM Blue Gene/Q JUQUEEN*, in Proc. ISC-HPC'16 Workshop on Exascale Multi/Many-Core Computing Systems (E-MuCoCoS, 23 June 2016), Springer (to appear).

[24] Hank Childs, Eric Brugger, Brad Whitlock, Jeremy Meredith, Sean Ahern, David Pugmire, Kathleen Biagas, Mark Miller, Cyrus Harrison, Gunther H. Weber, Hari Krishnan, Thomas Fogal, Allen Sanderson, Christoph Garth, E. Wes Bethel, David Camp, Oliver Rübel, Marc Durant, Jean M. Favre & Paul Navrátil, *VisIt: An end-user tool for visualizing and analyzing very large data*, in High Performance Visualization – Enabling Extreme-scale Scientific Insight, 357–372, Oct. 2012. https://visit.llnl.gov/

[25] Andreas Knüpfer, Christian Rössel, Dieter an Mey, Scott Biersdorff, Kai Diethelm, Dominic Eschweiler, Markus Geimer, Michael Gerndt, Daniel Lorenz, Allen D. Malony, Wolfgang E. Nagel, Yury Oleynik, Peter Philippen, Pavel Saviankou, Dirk Schmidl, Sameer S. Shende, Ronny Tschüter, Michael Wagner, Bert Wesarg & Felix Wolf, *Score-P – A joint performance measurement run-time infrastructure for Periscope, Scalasca, TAU, and Vampir*, In Proc. 5th Parallel Tools Workshop (Dresden, Germany), pp. 79–91. Springer, Sept. 2012. http://www.score-p.org/

[26] Markus Geimer, Felix Wolf, Brian J. N. Wylie, Erika Ábrahám, Daniel Becker & Bernd Mohr, *The Scalasca performance toolset architecture*, Concurrency and Computation: Practice and Experience, **22**(6):702–719, Apr. 2010. http://www.scalasca.org/

[27] Philip Carns, Kevin Harms, William Allcock, Charles Bacon, Samuel Lang, Robert Latham & Robert Ross, *Understanding and improving computational science storage access through continuous characterization.* ACM Transactions on Storage, 7:8:1-8:26, Oct. 2011. http://www.mcs.anl.gov/research/projects/darshan/

[28] Wolfgang Frings, Felix Wolf & Vestsislav Petkov, *Scalable massively parallel I/O to task-local files.* In Proc. ACM/IEEE SC09 Conference (Portland, OR, USA), Nov. 2009. http://www.fz-juelich.de/jsc/sionlib

[29] Wolfgang Frings, *Efficient Task-Local I/O Operations of Massively Parallel Applications.* Ph.D. thesis, to be published in IAS Series, Forschungszentrum Jülich.

APPENDIX

A. HIGH-Q CLUB CODES

The full description of the High-Q Club codes along with developer and contact information can be found on the WWW [11]. Before this year's workshop at the start of 2016, there were 25 member codes:

CIAO *multiphysics, multiscale Navier-Stokes solver for turbulent reacting flows in complex geometries*
RWTH-ITV & Sogang University

CoreNeuron *simulation of electrical activity of neuronal networks including morphologically detailed neurons*
EPFL Blue Brain Project

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

FE2TI *scale-bridging incorporating micro-mechanics in macroscopic simulations of multi-phase steels*
Universität zu Köln & TUB Freiberg

FEMPAR *massively-parallel finite-element simulation of multiphysics governed by PDEs*
UPC-CIMNE

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

ICON *icosahedral non-hydrostatic atmospheric model*
DKRZ & JSC SimLab Climate Science

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	Compiler / Languages	Tasking	Threading	Concurrency	Scaling	File I/O
CIAO	XL: Ftn	MPI 16		16: 458 752	S	MPI-IO, HDF5
CoreNeuron	XL: C C++	MPI 1	OpenMP 64	64: 1 835 008	S W	MPI-IO
dynQCD	XL: C	SPI 1	pthreads 64	64: 1 835 008	S	*unspecified*
FE2TI	XL: C C++	MPI 16	OpenMP 4	64: 1 835 008	S W	N/A
FEMPAR	XL: F08	MPI 64	(OpenMP)	64: 1 756 001	W	N/A
Gysela	XL: C F90	MPI	OMP+pthrd	64: 1 835 008	W	(HDF5)
ICON	XL: C Ftn	MPI 1	OpenMP 64	64: 1 835 008	S	(netCDF)
IMD	XL: C	MPI 64	(OpenMP)	64: 1 835 008	W	*unspecified*
JURASSIC	XL: C	MPI 32	OpenMP 2	64: 1 835 008	W	netCDF
JuSPIC	GCC: F90	MPI 4	OpenMP 16	64: 1 835 008	S	MPI-IO, POSIX
KKRnano	XL: F03	MPI 4	OpenMP 16	64: 1 835 008	S	SIONlib
LAMMPS-DCM	XL: C++	MPI 4	OpenMP 16	64: 1 835 008	S W	N/A
MP2C	XL: Ftn	MPI 32		32: 917 504	W	SIONlib
muPhi ($\mu\phi$)	XL: C++	MPI 32		32: 917 504	W	SIONlib
Musubi	XL: F03	MPI	OpenMP	32: 917 504	S W	N/A
NEST	XL: C++	MPI 2	OpenMP 8	16: 458 752	W	(SIONlib)
OpenTBL	XL: F03	MPI	OpenMP	64: 1 835 008	W	pHDF5
PEPC	GCC: C F03	MPI 1	pthreads 61	61: 1 744 992	W	(SIONlib,MPI-IO)
PMG+PFASST	XL: C F03	MPI 16	(pthreads)	16: 458 752	S	N/A
PP-Code	XL: F90	MPI	OpenMP	64: 1 835 008	S W	*unspecified*
psOpen	XL: F90	MPI 32	OpenMP 2	64: 1 835 008	S	pHDF5
SHOCK	XL: C	MPI 64		64: 1 835 008	S W	(cgns/HDF5)
TERRA-NEO	XL: C++ Ftn	MPI	OpenMP	64: 1 835 008	W	*unspecified*
waLBerla	XL: C++	MPI	OpenMP	64: 1 835 008	W	N/A
ZFS	Clang: C++	MPI 16	OpenMP 2	32: 917 504	S	(pNetCDF)

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

JURASSIC *solver for 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*
FZJ-IAS

LAMMPS(DCM) *a Dynamic Cutoff Method for the Large-scale Atomic/Molecular Massively Parallel Simulator for classical molecular dynamics simulations*
RWTH-AICES

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

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

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

NEST *large-scale simulations of biological neuronal networks*
FZJ/INM-6 & IAS-6

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

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, e.g. from discretisations of PDEs*
LBNL, Universität Wuppertal, Università della Svizzera italiana & JSC

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

psOpen *direct numerical simulation of fine-scale turbulence*
RWTH-ITV Inst. for Combustion Technology & JARA

SHOCK *structured high-order finite-difference kernel for compressible flows*
RWTH Shock Wave Laboratory

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

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

ZFS *computational fluid dynamics & aero-acoustics, conjugate heat transfer, particulate flows*
RWTH Fluid Mechanics and Inst. of Aerodynamics & JSC SimLab Fluids and Solids Engineering