D6.5 – Peta to exascale enabled applications (Software)

WP6: Co-design via applications

Project Acronym	CRESTA
Project Title	Collaborative Research Into Exascale Systemware, Tools and Applications
Project Number	287703
Instrument	Collaborative project
Thematic Priority	ICT-2011.9.13 Exa-scale computing, software and simulation

Due date:	M39
Submission date:	31/12/2014
Project start date:	01/10/2011
Project duration:	39 months
Deliverable lead organization	CSC
Version:	1.5
Status	Final version for EC review
Author(s):	Mark Abraham (KTH), Mikko Byckling (CSC), Willem Deconinck (ECMWF), Derek Groen (UCL), Jing Gong (KTH), Mats Hamrud (ECMWF), George Mozdzynski (ECMWF), Adam Peplinski (KTH), Jan Åström (CSC), Jan Westerholm (ABO)
Reviewer(s)	Stefano Markidis (KTH), Achim Basermann (KTH), Lorna Smith (EPCC)

Dissemination level	
PU	*PU – Public*

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	29/08/2014	Template of the deliverable	Mikko Byckling (CSC), Jan Åström (CSC)
0.2	27/11/2014	Added introduction and summary	Mikko Byckling (CSC)
0.3	27/11/2014	IFS contribution	George Mozdzynski (ECMWF), Mats Hamrud (ECMWF) , Willem Deconinck (ECMWF)
0.4	27/11/2014	HemeLB contribution	Derek Groen (UCL)
0.5	28/11/2014	Nek5000 contribution	Jing Gong (KTH), Adam Peplinski (KTH)
0.6	28/11/2014	Minor improvements	Mikko Byckling (CSC)
0.7	2/12/2014	Gromacs contribution	Mark Abraham (KTH)
0.8	2/12/2014	Elmfire contribution	Jan Westerholm (ABO)
1.0	2/12/2014	First version for internal review	Mikko Byckling (CSC)
1.1	9/12/2014	Addressed reviewer comments	Mikko Byckling (CSC)
1.2	17/12/2014	Addressed reviewer comments	Mikko Byckling (CSC), Lorna Smith (EPCC), Jan Westerholm (ABO)
1.3	17/12/2014	Addressed reviewer comments	Mikko Byckling (CSC)
1.4	17/12/2014	Final version for EC review	Mikko Byckling (CSC)
1.5	18/12/2014	Final check	Catherine Inglis (UEDIN)

Table of Contents

Index of Figures

Index of Tables

1 Executive summary

This deliverable, “D6.5 Peta to exascale enabled applications (Software)”, describes the status of the CRESTA applications at the end of the project. In short, the licensing policy, availability and the performance improvements achieved during CRESTA for each of the applications are:

ELMFIRE: Proprietary license, however access available on request. Contact details to use when applying for a license are given in the Elmfire section of this document. A domain decomposition version of ELMFIRE has been developed during CRESTA, resulting in a significant reduction in memory consumption.

GROMACS: Licensed under LGPLv2 and available in a public code repository. CRESTA modifications included in the main trunk. Performance and scalability (through ensemble computations) have been significantly improved during CRESTA.

HemeLB: Licensed under GPLv3 and available in a public code repository. CRESTA modifications included in the main trunk. Both performance and scalability have been significantly improved during CRESTA.

IFS: Proprietary license, contact details for applying for a license given in the IFS section of this document. CRESTA modifications included in the main trunk. Both performance and scalability have been significantly improved during CRESTA.

NEK5000: Licensed under GPLv3 and available in a public code repository. CRESTA modifications for OpenACC included in the main trunk, AMR modifications to be included in the trunk as soon as scalable pressure preconditioner has been implemented. By using OpenACC to offload computations to GPGPUs, performance improved during CRESTA.

OpenFOAM: Licensed under GPLv3 and available in a public code repository. Development effort ceased after M24. No improvements made to the code trunk during CRESTA.

Note that this document is closely linked to CRESTA deliverable “D6.1.3 Roadmap to exascale (update 2)“. D6.1.3 contains thorough performance analysis and a roadmap to reach exascale performance. This document considers the practical issues, such as how and from where to obtain the source code and how to apply or use the modifications implemented during CRESTA with the application.

2 Introduction

This document contains a description of the updated CRESTA applications that were successfully improved during the project (ELMFIRE, GROMACS, HemeLB, IFS and Nek5000). As improving the performance of OpenFOAM was deemed to require almost a complete rewrite of the whole application, its development within the project was not continued after M24.

For each application included we give either instructions or links to instructions for building the application with modifications done during the project in place. Also included are performance and scalability metrics to assess the achieved performance improvements. For detailed analysis of application performance and scalability as well as a roadmap for future improvements, see CRESTA deliverable "D6.1.3 Roadmap to exascale (update 2)".

The functionality and research goals of the applications can be summarized as follows:

ELMFIRE: is a gyro kinetic particle-in-cell code that simulates movement and interaction between high-speed particles in a torus-shaped geometry on a three dimensional grid. The particles are held together by an external magnetic field. The objective is to simulate significant portions of large-scale fusion reactors like JET or ITER.

GROMACS: is a molecular dynamics code that is extensively used for simulation of biomolecular systems. Useful investigation of this kind of system is typically limited by computational capacity. The limitations relate to both the system sizes and in particular the time duration of the processes which are of interest. Efficient implementation of ensemble simulations is also needed in order to achieve statistical validity.

HemeLB: is intended to form part of a clinically-deployed exascale virtual physiological human. HemeLB simulates blood flow in measured blood vessel geometries. The objective is to develop a clinically useful exascale tool.

IFS: is the production weather forecasting application used at the European Centre for Medium Range Weather Forecasts (ECMWF). The objective is to develop more reliable 10-day weather forecasts that can be run in an hour or less.

NEK5000: is an open-source code for the simulation of incompressible flow in complex geometries. Simulation of turbulent flow is of one of the major objectives of NEK5000.

2.1 Glossary of Acronyms

ACML	AMD Core Math Library
AVX	Advanced Vector Extensions
BSD	Berkeley Software Distribution
CAF	Coarray Fortran
CUDA	Compute Unified Device Architecture
DLR	Deutschen Zentrums für Luft- und Raumfahrt
ECMWF	European Centre for Medium-Range Weather Forecasts
EPCC	Edinburgh Parallel Computing Centre
FFT	Fast Fourier Transform
FFTW	Fastest Fourier Transform in the West
FTP	File Transfer Protocol

GGI	General Graphics Interface
GNU	GNU's Not Unix!
GPL	GNU General Public License
GPU	Graphics Processing Unit
GSL	Gnu Scientific Library
HDF5	Hierarchical Data Format
KTH	Kungliga Tekniska Högskolan
LGPL	GNU Lesser General Public License
MKL	Math Kernel Library
MPI	Message Passing Interface
NDA	Non-disclosure agreement
OpenMP	Open Multiprocessing
PESSL	Parallel Engineering and Scientific Software Library
PETc	Portable, Extensible Toolkit for Scientific Computation
PME	Particle Mesh Ewald
POSIX	Portable Operating System Interface
PRACE	Partnership for Advanced Computing in Europe
S3L	Sun Scalable Scientific Subroutine Library
SVN	Subversion
UCL	University College London
USTUTT	University of Stuttgart
XML	Extensible Markup Language

3 Elmfire

Application	Identification (e.g. Version No)
Elmfire	version 12

Table 3.1 Elmfire identification

3.1 Licensing

The code is not licensed, but access to the code can be obtained on request. Public licensing of the code is under consideration once the remaining changes have been implemented.

3.2 Acquisition

The application is not publicly available, but can be accessed on request for appropriate scientific purposes.

Contact person

Timo Kiviniemi (timo.kiviniemi@aalto.fi), Department of Applied Physics, Aalto University School of Science, P.O.Box 14100, FI-00076 AALTO Finland

3.3 Installation

The Makefile includes options for different platforms.

3.3.1 Prerequisites

The code has been used on CRAY, BULL, IBM and INTEL/AMD architectures. The code can utilize the PETSc/PESSL and ACML/MKL/GSL/ESSL libraries. The code is purely MPI-based and has been compiled using the INTEL, Cray and PGI compilers.

3.3.2 Compilation and installation

Different options exist within the Makefile for different platforms.

3.3.3 External links

None

3.4 Application performance overview

The main challenge for this code was memory consumption and one of the key goals for CRESTA was to develop a domain decomposition version of the code. This development has been achieved and performance runs have been carried out with this code on half a billion particles on 4096 cores (see figure below). These results demonstrate that memory consumption per core is currently almost proportional to the number of particles, a significant achievement and an original objective set within CRESTA. This is described in more detail in Deliverable D6.1.3 [1].

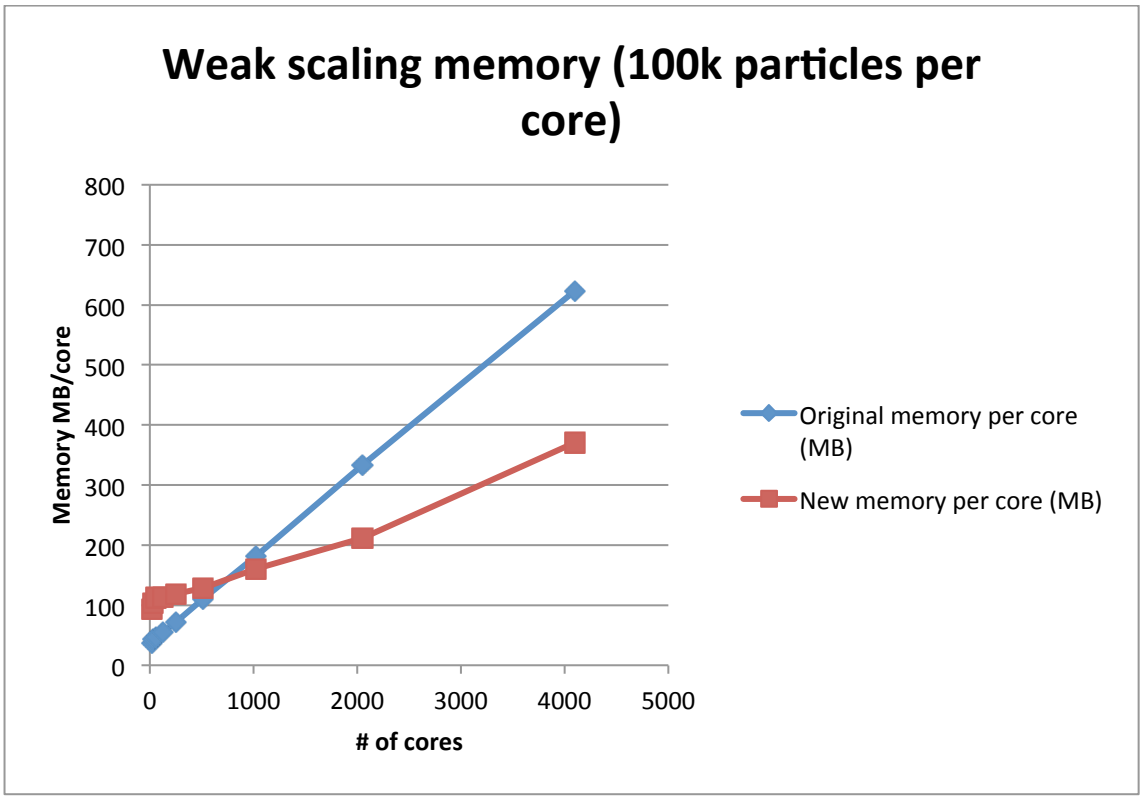

Figure 3.1 Elmfire memory scalability per core in a weak scaling test for a model problem

3.5 References

[1] Roadmap to exascale (update 2), CRESTA Deliverable D6.1.3.

4 Gromacs

Application	Identification (e.g. Version No)
Gromacs	4.6, 5.0 (Molecular Dynamics)

Table 4.1 Gromacs identification

4.1 Licensing

GROMACS is free software, distributed under the GNU Lesser General Public License (LGPL) Version 2.1.

4.2 Acquisition

Released versions of the code can be freely downloaded without registration from http://www.gromacs.org/Downloads. The source distribution can be used to download and run the tests automatically, or they can be obtained separately by following the instructions at http://www.gromacs.org/Documentation/Installation_Instructions#testing-gromacs-for-correctness.

The main development version of Gromacs is also publicly accessible as a git repository from either the Gromacs servers

```
git clone git://git.gromacs.org/gromacs.git
```

or from Github

```
git clone https://github.com/gromacs/gromacs.git
```

4.3 Installation

4.3.1 Prerequisites

Gromacs 5.0 requires

- both C99 and C++98 compilers (and is extremely compiler-portable),
- CMake version 2.8.8 or newer,
- for best performance with PME, either FFTW or MKL is required; FFTW can be downloaded and built automatically, as shown above, and
- for builds that target accelerators, you should honor the requirements of that platform, e.g. nvcc supports only specific versions of named compilers for GPU builds. (Our experience is that you can use other compilers if you manually circumvent the checks, however.)

On BlueGene/Q, either XL or bgclang compilers may be used.

4.3.2 Compilation and installation

For example, on everyday x86 machines with MPI and GPUs, where the build host is identical to the execution host, having downloaded a source distribution (or made a git clone), one can use

```
tar xfz gromacs-5.0.2.tar.gz
cd gromacs-5.0.2
mkdir build
cd build
cmake .. -DGMX_BUILD_OWN_FFTW=ON -
DREGRESSIONTEST_DOWNLOAD=ON -DGMX_GPU=on -DGMX_MPI=on
make
```

```
make check
sudo make install
source /usr/local/gromacs/bin/GMXRC
```

Any released version of any of the major compilers may be used, although the most recent version should be preferred. Best results are normally obtained with gcc.

Standard CMake behaviour is fully supported, including the use of CMAKE_INSTALL_PREFIX, CMAKE_PREFIX_PATH, BUILD_SHARED_LIBRARIES, CMAKE_C_COMPILER, CMAKE_CXX_COMPILER.

4.3.3 External links

Up-to-date Gromacs installation instructions may be found at http://www.gromacs.org/Documentation/Installation_Instructions. Detailed instructions for BlueGene/Q and Cray systems can be found there.

4.4 Application performance overview

The improvements implemented during CRESTA have greatly enhanced the simulation throughput throughput scientists can obtain with Gromacs. Strong scaling is the major design target for MD target for MD simulation software, because scientific quality is normally in direct proportion to the proportion to the amount of sampling conducted. Not only are CUDA, BlueGene/Q, and K-computer K-computer architectures now fully supported, but their implementation and supporting infrastructure are designed to be performance portable, to the extent we can anticipate hardware hardware trends. Even on x86 hardware that was available three years ago, the performance of performance of Gromacs 4.6 is around 20% faster than that of Gromacs 4.5 on the same hardware same hardware (through better use of SIMD of that era, and elimination of redundant computation computation in inner kernels), and the enhancements to strong scaling improve total performance performance by more than a factor of 2. Changes to x86 scaling performance can be seen in seen in

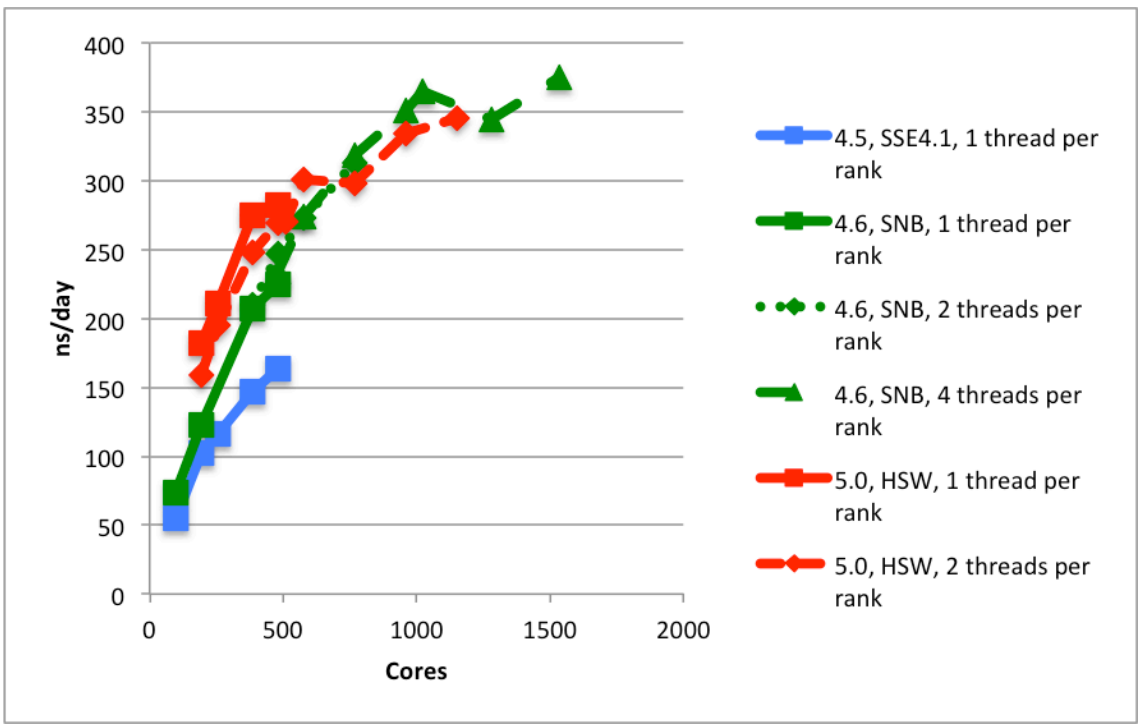

Figure 4.1. Further information is available in Deliverables D6.1.3 [3] and D6.4 [4].

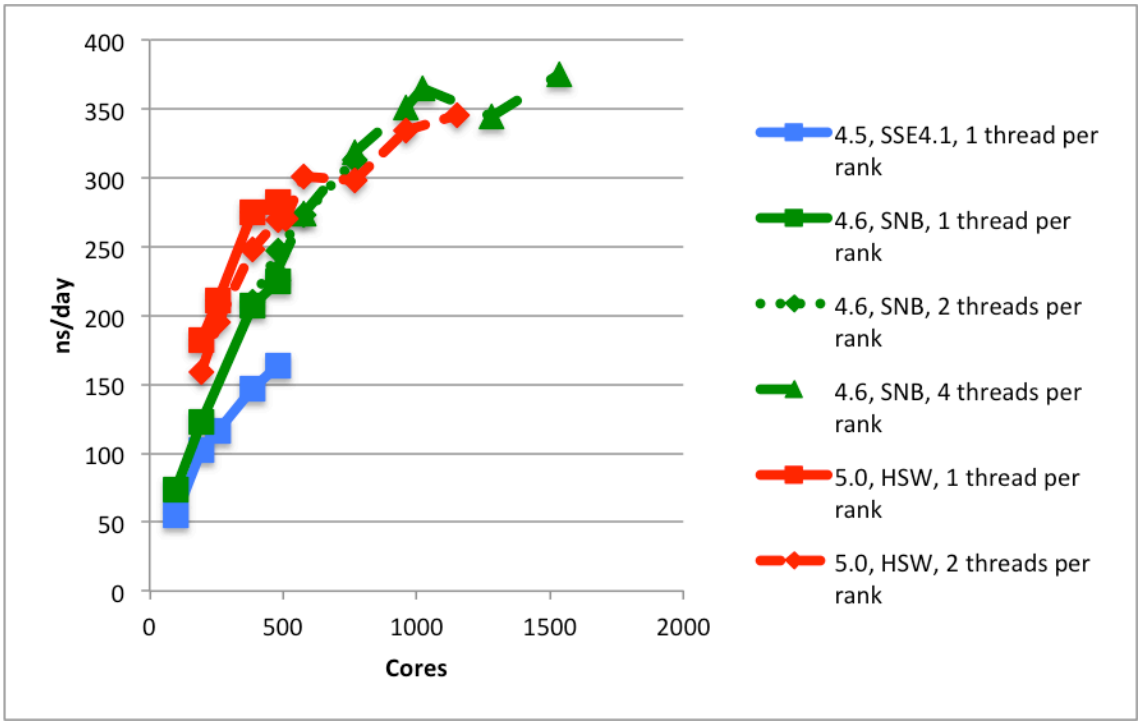

Figure 4.1 Scaling performance of Gromacs 4.5, 4.6 and 5.0 on Sandy Bridge (SNB) and Haswell (HSW) x86 platforms.

The completion of the related Copernicus project [2] makes it straightforward to execute elaborate ensemble-parallelism schemes using multiple Gromacs simulations as the computational workhorse. For the first time, it is easy for scientists to make effective use of Gromacs to implement advanced sampling algorithms at full scale on any current petascale machine.

4.5 References

[2] Pronk et al., “Copernicus: a new paradigm for parallel adaptive molecular dynamics”, SC11 High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for. IEEE, 2011.

[3] Roadmap to exascale (update 2), CRESTA Deliverable D6.1.3.

[4] Exemplar scientific simulations, CRESTA Deliverable D6.4.

5 HemeLB

HemeLB is a software package for modelling and simulation of haemodynamics in sparse geometries. To date, it has been mainly applied to the simulation of complex cerebrovascular networks and cerebrovascular malformations such as aneurysms. HemeLB includes: a) a GUI for simulation definition, b) a highly-optimised parallel lattice-Boltzmann (LB) solver including different LB collision operators, boundary conditions, and rheology models, c) a visualisation client allowing real-time visualisation and simulation steering, and d) a number of postprocessing tools. The package is currently developed at the Centre for Computational Science, UCL.

Application	Identification (e.g. Version No)
HemeLB	Commit: 4eefc9f6b55a29d8ec26f7b6de1605fff6357e60

Table 5.1 HemeLB identification

5.1 Licensing

HemeLB is available under GPL, version 3.

5.2 Acquisition

Public repository is hosted at http://ccs.chem.ucl.ac.uk/hemelb.

5.3 Installation

5.3.1 Prerequisites

HemeLB requires the following to be present on a system to run simulations:

- **C++** - a C++ compiler compatible with C++ 2003 or better
  - Supporting alignment control with __align__ or similar in order to build dependencies
  - Including C++ standard library
  - Tested with Cray, Intel, and Gnu
- **CMake** – version 2.8.6 or better is required
- **POSIX** – environment must conform to POSIX standard. Compilation has been tested with Debian linux, Mac OSX Snow Leopard, and various PRACE machines.
- **MPI** – required for parallelism. HemeLB has been tested with openmpi, mpich and cray MPI.
- **MPI_IO** – required for parallel I/O
- **Python** - Optional, needed only for validation of regression tests. Not needed for production. Has an additional NumPy-package dependency.

HemeLB will automatically download and install the following dependencies if not present:

- **Google CTemplate** – For details see http://code.google.com/p/ctemplate
- **Boost**
- **CPPUnit** - Optional, needed only for unit testing, not in production use. For details see http://sourceforge.net/projects/cppunit/
- **MPWide** - Optional, needed only for multiscale simulations. For details see http://castle.strw.leidenuniv.nl/software/mpwide.html
- **ZLib** – Data compression library, see http://www.zlib.net/
- **TinyXML** – For details see http://www.grinninglizard.com/tinyxml/
- **Parmetis** – Parallel graph partitioning package. For details see http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview

5.3.2 Compilation and installation

With your HemeLB checkout, run:

```
mkdir build
cd build
cmake -DCMAKE_INSTALL_DIR=path_to_install_to
make
```

5.3.3 External links

Detailed information on installation can be found in hemelb/README and on the HemeLB wiki at http://pauli.chem.ucl.ac.uk/trac, password restricted, access available on request by members of the CRESTA consortium.

5.4 Application performance overview

In this section, we show the development of the maximum achieved performance of HemeLB over the years in Figure 5.1, as published in papers and technical reports. For a detailed analysis of HemeLB performance, see [4].

At the start of the CRESTA project, in 2011, we were able to obtain a performance of 9.1 billion site updates per second using 12,288 cores. As a result of our collaborations in CRESTA, we now have been able to obtain a performance of 153 billion site updates per second using 49,152 cores in 2014. This constitutes a performance improvement of a factor of 16.8, whereas Moore's law would predict a performance improvement of approximately a factor of 2.8 over that period.

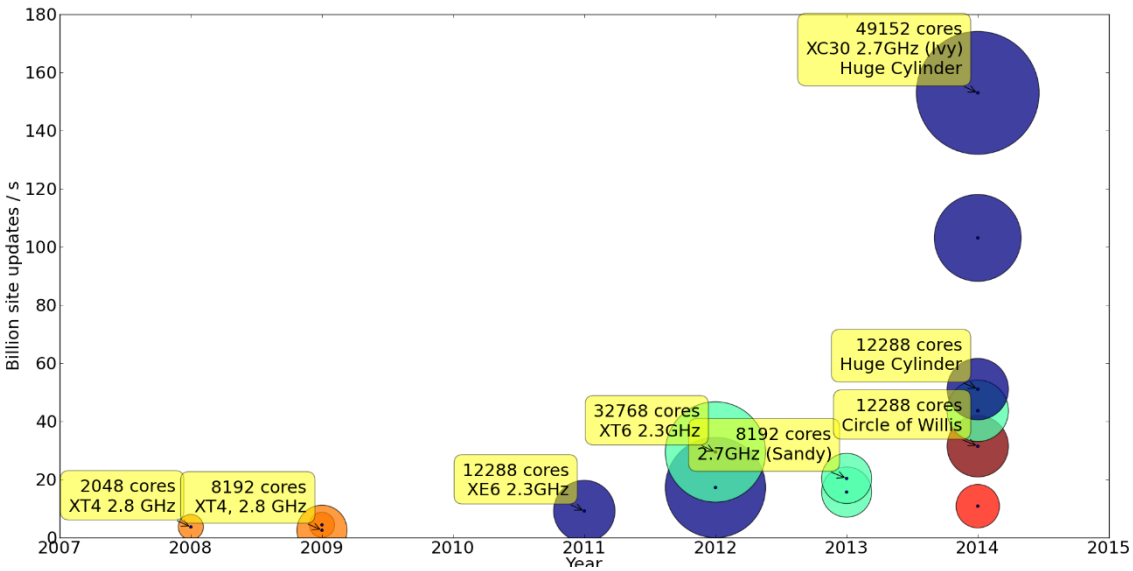

Figure 5.1 Obtained maximum performance achieved with HemeLB between 2007 and 2014. All improvements after 2011 were achieved during the CRESTA project. The sparsity of the data sets is roughly indicated by the color of the circle (very sparse is red, non-sparse cylinder data sets are blue), and the core count used by the size of the circle

Although the maximum obtained performance is a good indicator of how HemeLB is able to make more efficient use of high-end computing resources, it is only one of two important performance aspects for scientific users in the field. This is because in lattice-Boltzmann simulations the number of time steps required to reach convergence scales along with the size of the system modelled. As such, it becomes increasingly important for larger problems to measure the speed of our simulations, measured in time steps resolved per second. In Figure 5.2 we present this measure for all the runs which we previously presented in Figure 5.1. We find that our efforts within CRESTA have allowed us to simulate larger geometries than ever, and that we have managed to increase the rate of simulation to almost 2000 time steps per second. We believe these benefits are mainly results from our single-core optimizations, our improvements in domain decomposition, and the advent of newer and faster architectures such as Intel Ivy Bridge.

For very large data sets (e.g., the huge cylinder), however, we do observe a lower speed of about 600 time steps per second. Indeed, one of the major future challenges

for HemeLB will be to improve this measure for large problems, allowing us to reach convergence for these simulations within reasonable time spans.

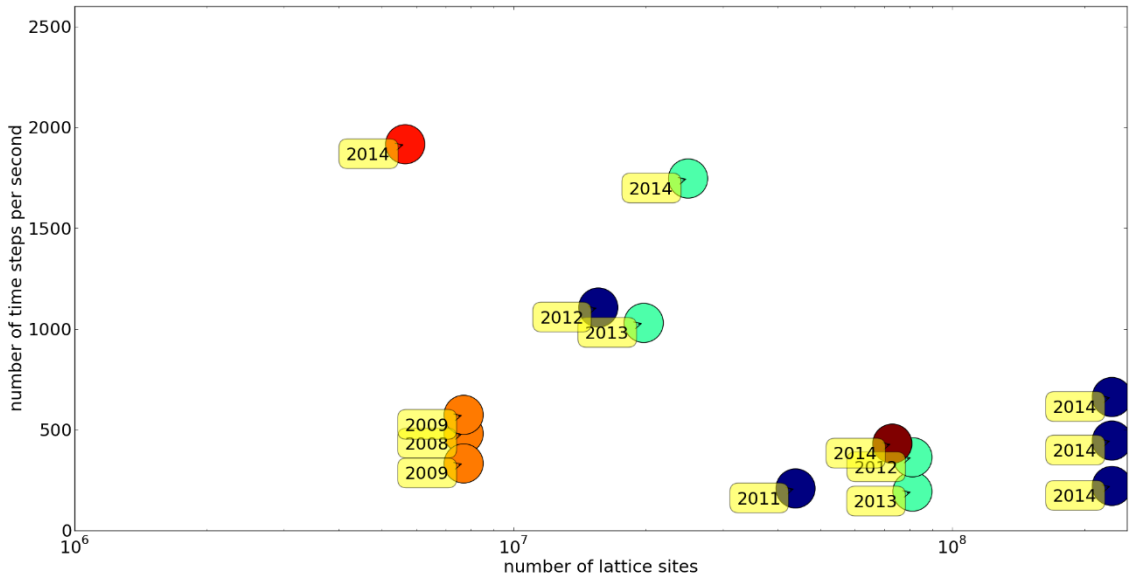

Figure 5.2 Obtained maximum number of time steps per second achieved, as a function of the problem size in the simulation (measured in number of lattice sites).

The work done in CRESTA has had major benefits for the domain-specific research activities with HemeLB, raising its profile in the scientific community. Primarily we have been able to model larger and more complicated geometries, and greatly reduce the time-to-completion for our simulations. As examples of the scientific impact using HemeLB, we have been able to publish new advances in high-profile domain journals such as J. R. Soc. Interface [1], Physics Review E [2] and Interface Focus [3].

5.5 References

[1] "Computer simulations reveal complex distribution of haemodynamic forces in a mouse retina model of angiogenesis", M.O. Bernabeu, C.A. Franco, M. Jones, J.H. Nielsen, T. Krüger, R.W. Nash, D. Groen, J. Hetherington, H. Gerhardt, P.V. Coveney, J. R. Soc. Interface (in press), arXiv preprint arXiv:1311.1640, 2013.

[2] "Choice of boundary condition for lattice-Boltzmann simulation of moderate Reynolds number flow in complex domains", R.W. Nash, H.B. Carver, M.O. Bernabeu, J. Hetherington, D. Groen, T. Krüger, P.V. Coveney, Physics Review E 89, 023033, 2014.

[3] "Impact of blood rheology on wall shear stress in a model of the middle cerebral artery", M.O. Bernabeu, R.W. Nash, D. Groen, H.B. Carver, J. Hetherington, T. Krüger, P.V. Coveney, Interface focus 3 (2), 20120094, 2013.

[4] Roadmap to exascale (update 2), CRESTA Deliverable D6.1.3.

6 IFS

The Integrated Forecasting System (IFS) is the production numerical weather forecast application at ECMWF. The IFS RAPS13 benchmark release contains a number of model cases, ranging from a small T159 model (125 km global resolution) to a large T3999 model (5 km global resolution).

The Atlas library is a framework for unstructured meshes on the sphere (including parallelization) being used in the development of an alternative dynamical core option to the spectral transform method used today in IFS. Atlas was developed within the CRESTA project. A stand-alone benchmark case is provided for a simple gradient computation based on a vertex-centered finite volume scheme, which is representative of a communicating kernel with nearest neighbor overlap regions.

Application	Identification (e.g. Version No)
IFS	RAPS13, ECMWF internal cycle 38R2
Atlas library	Pre-release version 0.3

Table 6.1 IFS and Atlas identification

6.1 Licensing

IFS software (RAPS13) is available solely for benchmarking purposes and requires a license from ECMWF. To request a license for RAPS13 IFS software please contact Isabella.Weger@ecmwf.int, Deputy Director of Computing at ECMWF. The license for IFS software prohibits distribution to a 3rd party.

The Atlas library is available under the conditions of an Apache 2.0 license.

6.2 Acquisition

Upon receipt of a signed license by ECMWF, details for downloading (ftp address/password), building and running the IFS RAPS13 benchmark will be provided by email. These instructions are contained in a benchmark release document (14 pages).

The contact person for the IFS RAPS13 benchmark is George Mozdzynski (George.Mozdzynski@ecmwf.int).

The contact person for the Atlas library is Willem Deconinck (Willem.Deconinck@ecmwf.int).

6.3 Installation

The installation process for the IFS RAPS13 benchmark is described in the benchmark release document.

6.3.1 Prerequisites

To be able to build and run the RAPS13 release the following facilities are needed:

- **C** - an ANSI standard C compiler
- **Fortran 95** - a Fortran 95 compiler that supports an auto-double (or -r8) capability
- **ksh** - Korn shell (ksh93)
- **perl** - used extensively with generic Makefile to build source libraries
- **mpi1** - required for parallel execution

Note that if your Fortran compiler supports the Fortran 2008 standard, then the use of **Fortran coarrays** will require specifying the macro option **-DCOARRAYS** at compile time. Then at runtime you can decide to use the IFS coarray optimisations by setting

the **LCOARRAYS** namelist variable in NAMPAR1 to true, noting that the default is false.
Installation of the Atlas library is described in a Readme file within the release.

6.3.2 Compilation and installation

Refer to RAPS13 benchmark release document and Atlas Readme file.

6.4 Application performance overview

The accuracy of numerical weather prediction crucially depends on the quality of the forecast model and the initial conditions. Both require the computationally efficient integration of complex physical process equations at global scale and methods optimizing memory usage, load balancing and data communication. Future improvements in predictive skill are expected from increased spatial resolution and much enhanced observational data usage, both of which impose significant requirements on code scalability and algorithmic flexibility.

Figure 6.1 shows how scalability of a 10 km IFS global model with 137 atmospheric levels has improved during the CRESTA project. The details of these improvements are presented in [1] and summarized below in Table 6.2 for runs using 45,056 AMD Interlagos cores on HECToR and TITAN. The performance measure is Forecast Days per Day (FD/D), where the operational requirement for a 10-day forecast is one hour or 240 FD/D. This model case is expected to enter operations at ECMWF in 3Q2015.

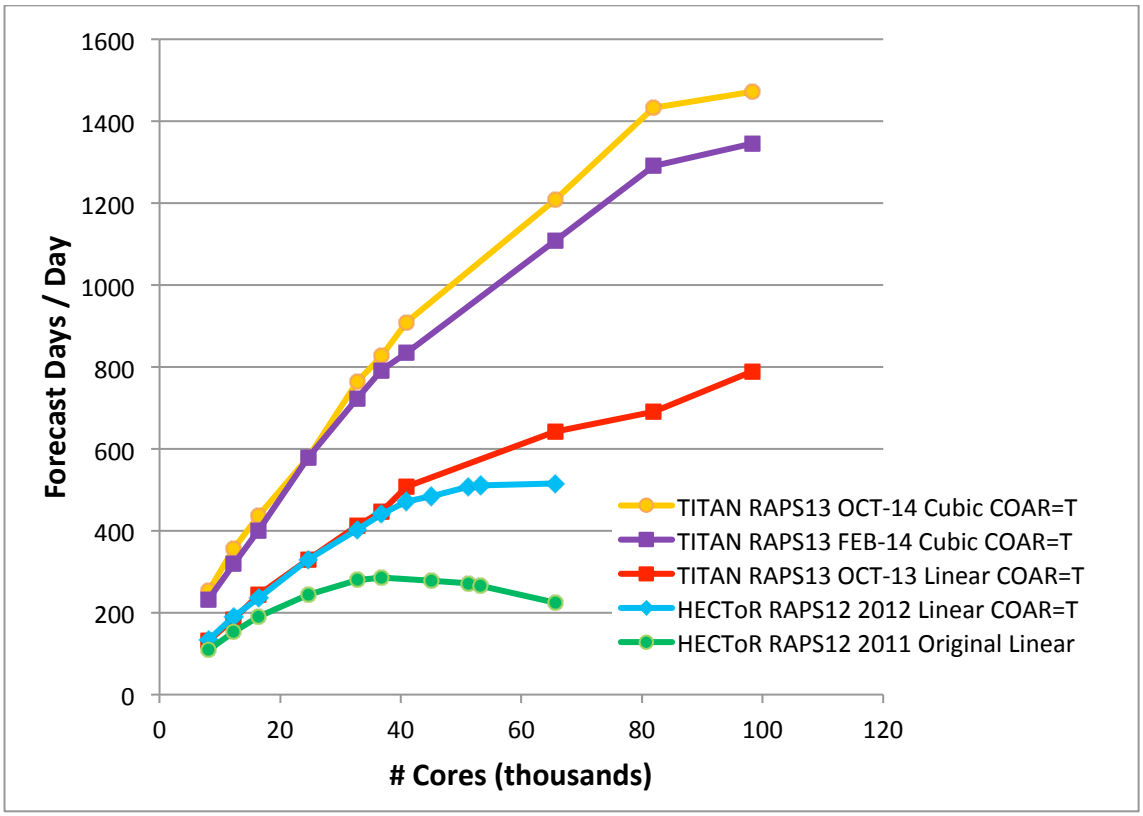

Figure 6.1 10 km / L137 global IFS forecast model performance, RAPS12 (CY37R3, on HECToR), RAPS13 (CY38R2, on TITAN)

Code version	Compiler Release/System	10 km model FD/D	Relative Performance
RAPS12 (CY37R3) base, linear grid, TSTEP=450s	8.0.3 HECToR	277	1.00
MPI optimizations to wave model	8.0.3 HECToR	356	1.29

new compiler release, improved compiler opts	8.0.6 HECToR	419	1.51
All coarray optimizations (LT, FT, SL)	8.0.6 HECToR	485	1.75
RAPS13 (CY38R2) base	8.1.5 TITAN	500 est.	1.80 est.
Using cubic grid (still 10km global grid), TSTEP=600s	8.2.2 TITAN	880 est.	3.17 est.
Final runs OCT-14 with reduced norms	8.3.0 TITAN	925	3.34

Table 6.2 Evolution of IFS 10 km L137 model performance using 45,056 cores on HECToR and TITAN

6.5 References

[1] Roadmap to exascale (update 2), CRESTA Deliverable D6.1.3.

[2] Mozdzynski, G., and J.-J. Morcrette, Reorganization of the radiation transfer calculations in the ECMWF IFS. ECMWF Technical Memorandum No.721, April 2014.http://old.ecmwf.int/publications/library/ecpublications/_pdf/tm/701-800/tm721.pdf

7 Nek5000

Application	Identification (e.g. Version No)
Nek5000_AMR	Nek5000 version with Adaptive Mesh Refinement Components: Nek5000 – revision 1020; p4est – version 0.3.4.1; ParMETIS – version 4.0.3
NekBone_ACC	NekBone version v3.1 with OpenACC derivatives
Nek5000_ACC	Nek5000 revision 1039 with OpenACC derivatives

Table 7.1 Nek5000 identification

7.1 Licensing

Main solver

- **Nek5000** is open-source software released under General Public License.
- **NekBone** is open-source software released under General Public License.

Additional libraries required by Adaptive Mesh Refinement

- **p4est** library is free software released under GNU General Public License version 2.
- **ParMETIS** is copyrighted by the Regents of the University of Minnesota. It can be freely used for educational and research purposes by non-profit institutions and US government agencies only. Other organizations are allowed to use ParMETIS only for evaluation purposes, and any further uses will require prior approval. The software may not be sold or redistributed without prior approval. One may make copies of the software for personal use provided that the copies are not sold or distributed, and are used under the same terms and conditions.

7.2 Acquisition

Main solver

- **Nek5000**: source code is maintained in a subversion (SVN) repository and can be downloaded with the SVN client checkout command:

```
svn co -r rev_number
https://svn.mcs.anl.gov/repos/nek5/ \
    ./nek5_svn
```

  where *–r* option specifies the revision number to be downloaded. The repository and downloading instruction can be found on the Nek5000 homepage: https://nek5000.mcs.anl.gov/index.php/GETNEK
- **NekBone**: source code is available from Argonne national laboratory webpage, located at: https://cesar.mcs.anl.gov/content/software/thermal_hydraulics

Additional libraries required by Adaptive Mesh Refinement

- **p4est library**: Official stable releases of the source code in the form of Unix gz.tar -file can be downloaded from the project home page [3]. The p4est source code contains two necessary libraries **sc** and **p4est**.
- **ParMetis**: ParMETIS's distribution is available as a Unix gz.tar -file. It can be downloaded from the project home page [4].

Developed tools

- **GenTree**: Nek5000 uses relatively simple description of the simulation mesh included in ###.map and ###.rea or ###.re2 files. However, the p4est library requires specific information about tree forest structure and connectivity, which is not directly available in the Nek5000 input files. To extract this information we

have developed the gentree tool, which, using Nek5000 input files for conformal mesh, generates ###.tree file in the p4est format. This file replaces ###.map file. The source code for gentree can be found at:
ftp://ftp.mech.kth.se/pub/adam/Nek5000/CRESTA/Nek5000_AMR/GenTree

Code modifications

- **Nek5000_AMR**: AMR version of Nek5000 is not yet included in the official repository at Argonne. As the code is under constant development we found the distribution of complete examples to be the most effective. In this case solver modifications are distributed together with setup files including mesh and compilation scripts.
- **NekBone_ACC**: OpenACC version of NekBone is not yet included in the official releases at Argonne. As the code is under constant development we found the distribution of complete examples to be the most effective. In this case solver modifications are distributed together with setup files including mesh and compilation scripts.
- **Nek5000_ACC**: OpenACC version of Nek5000 is not yet included in the official repository at Argonne. As the code is under constant development we found the distribution of complete examples to be the most effective. In this case solver modifications are distributed together with setup files including mesh and compilation scripts.

Setups

In all of the setups, solver modifications are distributed together with the setup files including mesh and compilation scripts.

- **Nek5000_AMR**: Unix gz.tar -files with working examples can be found under:
ftp://ftp.mech.kth.se/pub/adam/Nek5000/CRESTA/Nek5000_AMR/Setups
- **NekBone_ACC**: Unix gz.tar -files with working examples can be found under:
ftp://ftp.mech.kth.se/pub/adam/Nek5000/CRESTA/NekBone_ACC/
- **Nek5000_ACC**: Unix gz.tar -files with working examples can be found under:
ftp://ftp.mech.kth.se/pub/adam/Nek5000/CRESTA/Nek5000_ACC/

Contact people

- Adam Peplinski (adam@mech.kth.se) – Nek5000_AMR
- Jing Gong (gongjing@kth.se) – NekBone_ACC, Nek5000_ACC

7.3 Installation

7.3.1 Prerequisites

Main solver

- **Nek5000** runs under Linux or any Unix-like OS such as Mac, AIX, BG, Cray etc. It was tested with GNU, PGI, Intel and Cray compilers. The installation procedure is described in [1]. The installation process consists of building the tools and copying the scripts necessary to create input files for nek5000 simulations. Most of those tools are not necessary for performing the simulation, but they are important in the preprocessing step. Nek5000 uses GNU make for compilation and requires Fortran77, C compilers and the MPI library.
- **NekBone** runs under Linux or any Unix-like OS such as Mac, AIX, BG, Cray etc. It was tested with GNU, PGI, Intel and Cray compilers. The installation procedure is described at
https://cesar.mcs.anl.gov/content/software/thermal_hydraulics.
The NekBone benchmark uses GNU make for compilation and requires Fortran77, C compilers and the MPI library.

Additional libraries required by Adaptive Mesh Refinement

- **p4est** library runs under Linux or any Unix-like OS such as Mac, AIX, BG, Cray etc. Within CRESTA project it was tested with GNU, PGI and Intel compilers. The nstallation procedure is described in the README and INSTALL files

included in the source code tarball. p4est uses the GNU autoconf/automake/libtool build system and requires a C compiler and MPI library for compilation.

- **ParMetis** have been extensively tested on a number of different parallel computers. Within the CRESTA project it was tested with GNU, PGI and Intel compilers. The installation procedure is described in the INSTALL.txt file included in the source code tarball. ParMetis uses the GNU make tool for compilation and requires a C compiler and MPI library.

Compilers required by NekBone_ACC and Nek5000_ACC

A PGI or Cray CCE compiler supporting OpenACC derivatives is required for NekBone_ACC and Nek5000_ACC. To obtain better performance, the use of the latest releases, PGI v14.10.0 or Cray CCE v8.4.0, is recommended.

Developed tools

- **GenTree** runs under Linux or any Unix-like OS such as Mac, AIX, BG, Cray etc. It was tested with GNU, PGI and Intel compilers. This tool is used in the preprocessing step. The installation procedure is described in the README file included in the source code tarball. It uses the GNU make tool for compilation and requires Fortran77, C compilers and MPI, p4est libraries. This tool is similar to the genmap tool included in the Nek5000 repository.

Code modifications

- **Nek5000_AMR**: the AMR version of Nek5000 requires the source code of the Nek5000 solver revision 1020, compilers for Fortran77 and C, and MPI, p4est, ParMetis libraries. It was tested with GNU, PGI and Intel compilers on a Linux system and Cray XE6.
- **NekBone_ACC**: the OpenACC version of NekBone requires the source code of the NekBone solver version 3.1, compilers for Fortran77 and C. It was tested with PGI and Cray CCE compilers on a Linux system and Cray XK7.
- **Nek5000_ACC**: the OpenACC version of Nek5000 requires the source code of the NekBone solver version 3.1, compilers for Fortran77 and C. It was tested with PGI and Cray CCE compilers on a Linux system and Cray XK7.

7.3.2 Compilation and installation

Main solver

- **Nek5000:** The installation procedure is described in the Nek5000 user guide [1]. The process consists of building the tools and copying the scripts necessary to create input files for nek5000 simulations. The source code can be downloaded with

```
svn co -r rev_number
https://svn.mcs.anl.gov/repos/nek5/ \
        ./nek5_svn
```

The shell commands

```
cd nek5_svn/trunk/tools
maketools all
```

build the tools and copy them to the generated top level bin directory. In addition there are a number of scripts located under directory

```
nek5_svn/trunk/tools/scripts
```

This can be useful during different simulation stages. For more information about tools, their installation and preparation of the input files for nek5000 simulations see [2].

NekBone

- The installation procedure is described in:

Additional libraries required by Adaptive Mesh Refinement

- **p4est:** The installation procedure is described in the README and INSTALL files included in the source code tarball. Briefly, the shell commands:

```
./configure
make
make install
```

configure, build, and install the package. For more information see INSTALL file and [3].

- **ParMetis:** The installation procedure is described in the INSTALL.txt file included in the source code tarball. ParMetis uses the GNU make tool for compilation and requires a C compiler and MPI library. Briefly, the shell commands

```
make config
make
make install
```

configure, build, and install this package. For more information see the INSTALL.txt file and [4].

Developed tools

- **GenTree:** The installation procedure is described in the README file included in the source code tarball. Briefly, the shell commands

```
make clean
make all
```

clean the source tree and build this tool. Before executing make src/genconn.h and Makefile files should be edited. genconn.h contains sizes for static array allocation (max number of trees, grid dimension and number of passive scalars). These numbers have to correspond to the values used in the simulation (SIZE file). In the Makefile information about compilers, include and library paths must be updated. This tool is similar to the genmap tool included in Nek5000 repository. For more information see the README file.

Code modifications

- **NekBone_ACC**: The installation procedure is described in the README file included in the source code tarball.
- **Nek5000_ACC**: The installation procedure is described in the README file included in the source code tarball.

Setups

- **Nek5000_AMR**: Each example setup under ftp://ftp.mech.kth.se/pub/adam/Nek5000/CRESTA/Nek5000_AMR/Setups contains the complete set of files and scripts to compile and run the given examples except the Nek5000 code, which has to be downloaded separately. Although other files usually do not require changes they can be modified or set up from scratch. To compile and run example setup first static arrays sizes have to be defined (SIZE file used by Nek5000) and input files have to be generated. There are number of parameters in SIZE and we refer the reader to https://nek5000.mcs.anl.gov/index.php/SIZEu for more in-depth description.
  We mention only *ldim* (number of spatial dimensions) and *ldimt* (maximum number of T-array fields), which must be consistent with *N_DIM* and *N_NPSCL* in src/nekp4est.h (used by p4est) and genconn.h (part of GenTree source code). The input files ###.rea, ###.re2 (optionally) and ###.map can be created with native Nek5000 tools (genbox, prex, genmap, n2to3 …) and next ###.tree

should be generated with gentree. More information about grid generation and different tools can be found at
https://nek5000.mcs.anl.gov/index.php/UG
Every example tarball contains:

- Directory structure
  - bin: to store executable nek5000
  - nek: temporary storage for Nek5000 source code revision 1020 (some files overwritten by files from src)
  - obj: to store object files
  - src: nekp4est source code and modified Nek5000 files
  - test: directory containing input files and running script
- Scripts
  - compile_np4est.sh: main compilation script
  - compile.sh, clean.sh: wrappers for fast compilation/cleaning
  - makenek: Nek5000 native compilation script called by compile.sh
- Files
  - makefile_usr.inc: include file with user makefile definitions
  - README: setup description
  - SIZE: static array definitions for Nek5000

Before compiling, the following files have to be updated:

- compile_np4est.sh
  - NEK_HOME: should point to Nek5000 source code (only release 1020 is supported)
- makenek
  - F77: F77 mpi compiler,
  - CC: C mpi compiler,
  - USR_LFLAGS: path to p4est and ParMetis libraries
- makefile_usr.inc
  - P4EST_HOME: include path for p4est
  - PARMETIS_HOME: include path for ParMetis
- src/nekp4est.h has to be consistent with SIZE

To compile the code, run the compile script. Executable bin/nek5000 and compilation log compiler.out should be generated. Check log file for errors. There may be a number of warnings due to inconsistent common block structures. To run a simulation, move to the test directory, copy the compiled executable and run the execute script. The execute script may require updating as the simulation execution method is system-dependent. The number and type of input/output files depends on the given example and is described in the README file.

- **NekBone_ACC**: Each example setup under the directory
ftp://ftp.mech.kth.se/pub/adam/Nek5000/CRESTA/NekBone_ACC
contains the complete set of files and scripts to compile and run the given example except the NekBone code, which has to be downloaded separately. Although other files usually do not require changes they can be modified or set up from scratch. To compile and run example setup first static arrays sizes have to be defined (SIZE file used by NekBone) and input files have to be generated.
Every example tarball contains:
  - Directory structure
    - src: NekBone files
    - test: directory containing input files and running script
  - Script
    - makenek: NekBone native compilation script called by compile.sh
  - Files
    - README: setup description
    - SIZE: static array definitions for NekBone

Before compiling, the following files have to be updated:

- makenek
  - SOURCE_ROOT: the path to the NekBone source code F77
  - F77: the name of the F77 MPI compiler
  - CC: the name of the C MPI compiler
  - USR_LFLAGS: specify any desired flags
  - IFPGIACC: Uncomment and specify PGI OpenACC derivatives
  - IFCRAYACC: Uncomment and specify Cray OpenACC derivatives
- SIZE
  - lp: the maximum number of GPU/MPI ranks
  - lelt: the maximum number of element per GPU/MPI ranks

To compile the code, run the compile script. The executable nekbone should be generated. There may be a number of warnings due to inconsistent common block structures. To run a simulation, move to the test directory, copy the compiled executable and run the execute script. The execute script may require updating as the running method can be system-dependent. The current version of this script uses the number and type of input/output files depending on the given example and is described in the README file.

- **Nek5000_ACC**: Each example setup under ftp://ftp.mech.kth.se/pub/adam/Nek5000/CRESTA/Nek5000_ACC contains the complete set of files and scripts to compile and run the given example except the Nek5000 code, which has to be downloaded separately. Although other files usually do not require changes they can be modified or set up from scratch. To compile and run the example setup static arrays sizes have to be defined first (SIZE file used by Nek5000) and input files have to be generated. There are a number of parameters in SIZE and we refer the reader to https://nek5000.mcs.anl.gov/index.php/SIZEu for a more in-depth description.
  Every example tarball contains:
  - Directory structure
    - src: Nek5000 files
    - test: directory containing input files and running script
  - Scripts
    - makenek: Nek5000 native compilation script called by compile.sh
  - Files
    - README: setup description
    - SIZE: static array definitions for Nek5000

  Before compiling, the following files have to be updated:
  - makenek
    - SOURCE_ROOT: the path to the NekBone sourcecodeF77
    - F77: the name of the F77 MPI compiler
    - CC: the name of the C MPI compiler,
    - USR_LFLAGS: specify any desired flags
    - IFPGIACC: Uncomment and specify PGI OpenACC derivatives
    - IFCRAYACC: Uncomment and specify Cray OpenACC derivatives
  - SIZE
    - lp: the maximum number of GPU/MPI ranks
    - lelt: the maximum number of element per GPU/MPI ranks

  To compile the code, run the compile script. The executable nek5000 and compilation log compiler.out should be generated. Check log file for errors. There may be number of warnings due to inconsistent common block structures. To run a simulation, move to the test directory, copy the compiled executable and run the execute script. The execute script may require updating as the running method can be system-dependent. The current version of this

script uses the number and type of input/output files dependent on the given example and is described in the README file.

7.3.3 External links

The Nek5000 homepage is located at http://nek5000.mcs.anl.gov.

Building and using nek5000 is described at http://nek5000.mcs.anl.gov/index.php/UG

Homepages for p4est and ParMetis are located at http://www.p4est.org/ and http://glaros.dtc.umn.edu/gkhome/metis/parmetis/ respectively.

7.4 Application performance overview

In this section, we discuss the achieved performance of the CRESTA modifications of Nek5000. For a detailed analysis of Nek5000 performance, we refer the reader to deliverable D6.1.3 [5].

Nek5000_AMR

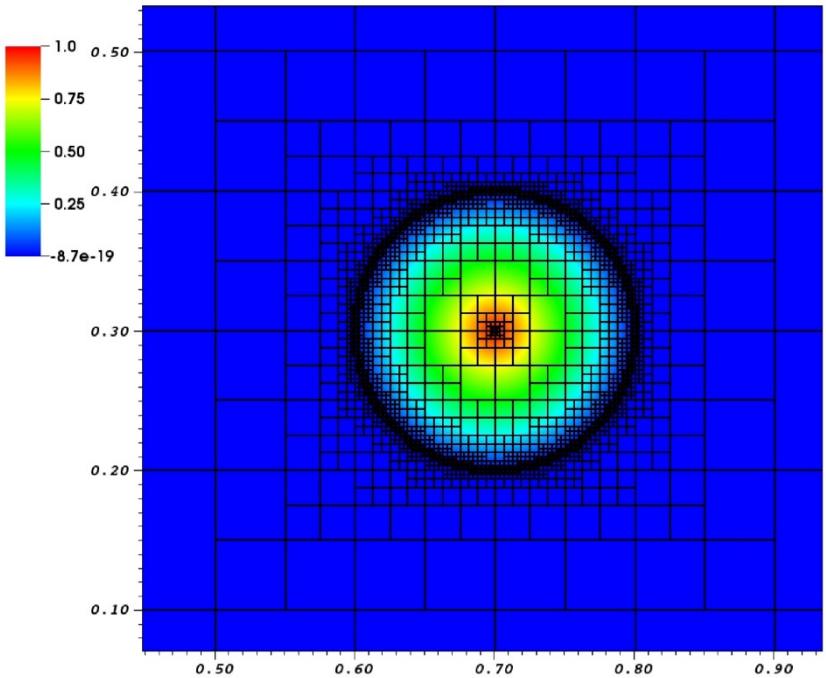

Figure 7.1 Two–dimensional cut through the domain of the convected-cone problem showing the grid structure (black squares) and the passive scalar profile (color scale). Each element (3D cube depicted by a square) corresponds to the mesh of 12x12x12 grid points.

Within CRESTA we implemented in Nek5000 all the tools necessary to dynamically modify the mesh structure during the simulation by changing the global number of elements through h-refinement, see Figure 7.1. This allows for non-conformal meshes, which add more flexibility to grid generation by removing e.g. the refinement propagation problem in the conformal meshes which lead to unnecessary elements in the far-field and to high aspect-ratio elements that are detrimental to iterative solver performance. It also allows for control of the computational error during the simulation by using a proper error estimator.

We performed a number of model simulations with the AMR version of Nek5000 based on the convected-cone example introduced by Gottlieb and Orszag, which is the passive scalar transport problem. We adopted this example to 3-dimensional simulations evolving a sphere-shape (strong scaling) or cylinder-shape (weak scaling) cone according to the energy equation in Nek5000. In all the tests performed, we

followed advected features in the flow (the cone), which requires continuous adjustment of the mesh and does not converge to any time independent grid structure. In our simulations the mesh was regenerated every 50 Nek5000 steps. All runs were performed on a Cray XE6 system with the core number being power of 2 and ranging from 2048 up to 32,768. For an example of parallel efficiency related to weak scaling for a model problem, see Figure 7.2.

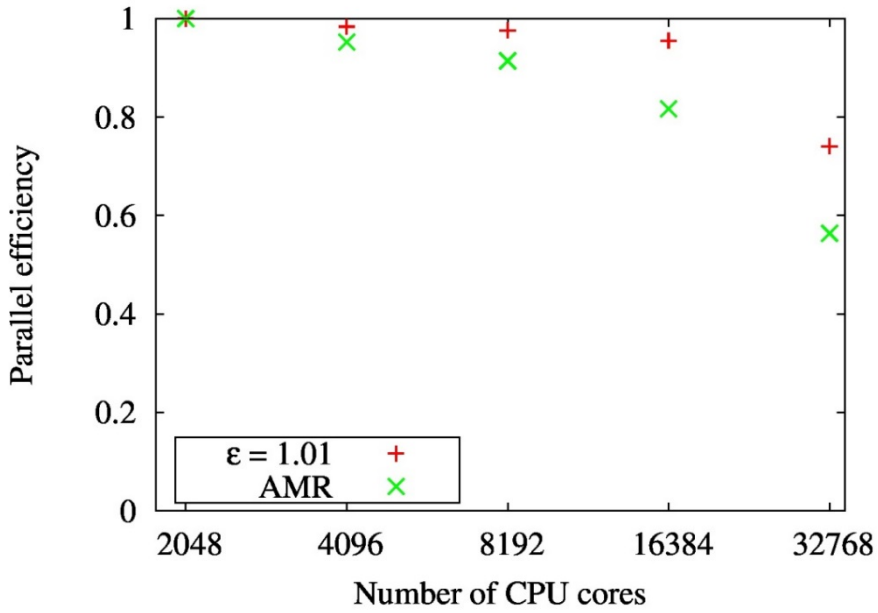

Figure 7.2 Parallel efficiency of the simulations without (ε=1.01) and with (AMR) grid adaptation for non-conformal version of Nek5000.

We found the non-conformal version of Nek5000 to be the most efficiently parallelized component of our code. The biggest constraint in the parallel scaling comes from the performance of the grid partitioner showing the partitioning from scratch strategy to be inefficient. This would not allow for exascale simulations in which advected flow features are followed, as the mesh requires continuous adjustment and does not converge to any time-independent grid structure. However, in the stability calculations, where the final mesh structure can be time-independent, costly AMR with mesh adaptivity turned on can be used as a pre-processing tool and non-conformal Nek5000 solver can be used during the main simulation.

NekBone_ACC

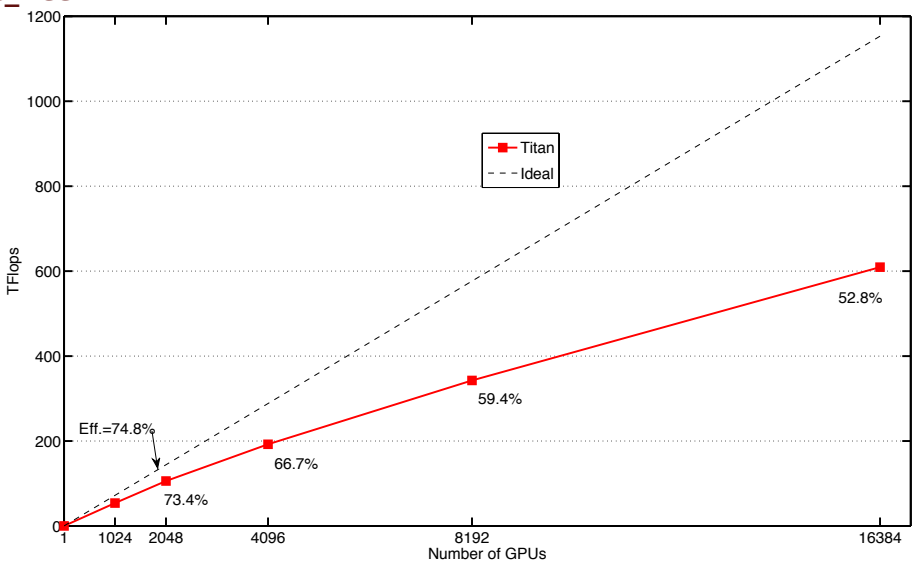

Figure 7.3 The weak scaling results on the Titan supercomputer with up to 16,384 GPUs for NekBone (red line) and in the ideal case (black dashed line).

Within the CRESTA project, NekBone has been ported to multi-GPU systems using OpenACC compiler directives. The focus of this work is on porting the most time-consuming routines of the NekBone to a GPU system, i.e. matrix-matrix multiplications. The optimized version for multi-GPU systems gives a performance of 609.8 Tflops on 16,384 GPUs on Titan, as described in Figure 7.3.

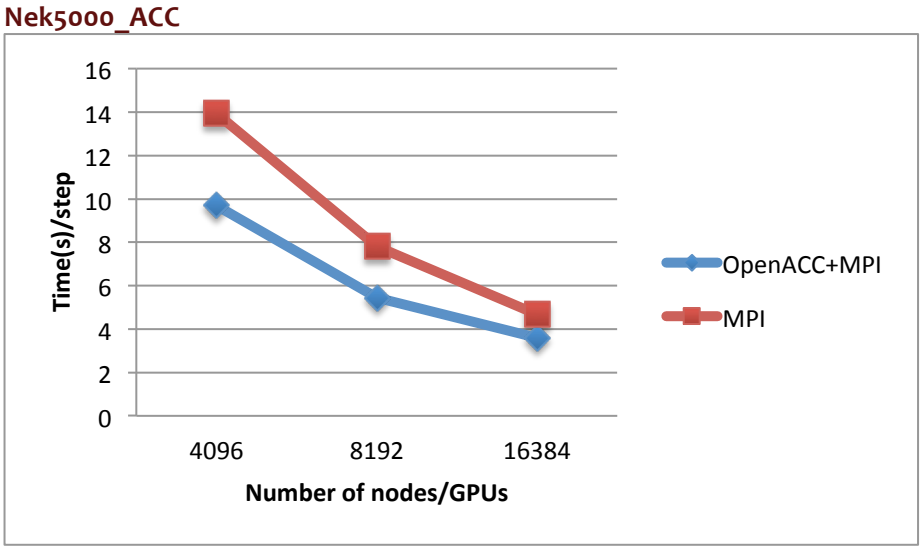

Figure 7.4 The strong scalability for Nek5000 using 16th-order polynomial. Total number of grid points is 5177 M.

Within the CRESTA project, the full Nek5000 code has been ported to a multi-GPU system using OpenACC compiler directives. The work focused on porting the most time-consuming parts of Nek5000 to the GPU system, namely the preconditioned iterative linear solver. The gather-scatter method with MPI operations has been redesigned in order to decrease the amount of data to transfer between the host and the accelerator. On 4096 nodes of the Titan supercomputer, the speed-up can be approach 1.4 times with a16th-order polynomial, see Figure 7.4. A preliminary study showed that overlapping of GPU kernels with host-accelerator memory transfers could increase the performance of the OpenACC version of Nek5000. This will be part of future research.

7.5 References

[1] Paul Fischer et. al., Nek5000 user guide, http://www.mcs.anl.gov/~fischer/new.pdf.

[2] Nek5000 wiki page, https://nek5000.mcs.anl.gov/index.php/UG

[3] p4est home page, http://www.p4est.org/

[4] ParMetis home page, http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview/

[5] Roadmap to exascale (update 2), CRESTA Deliverable D6.1.3.