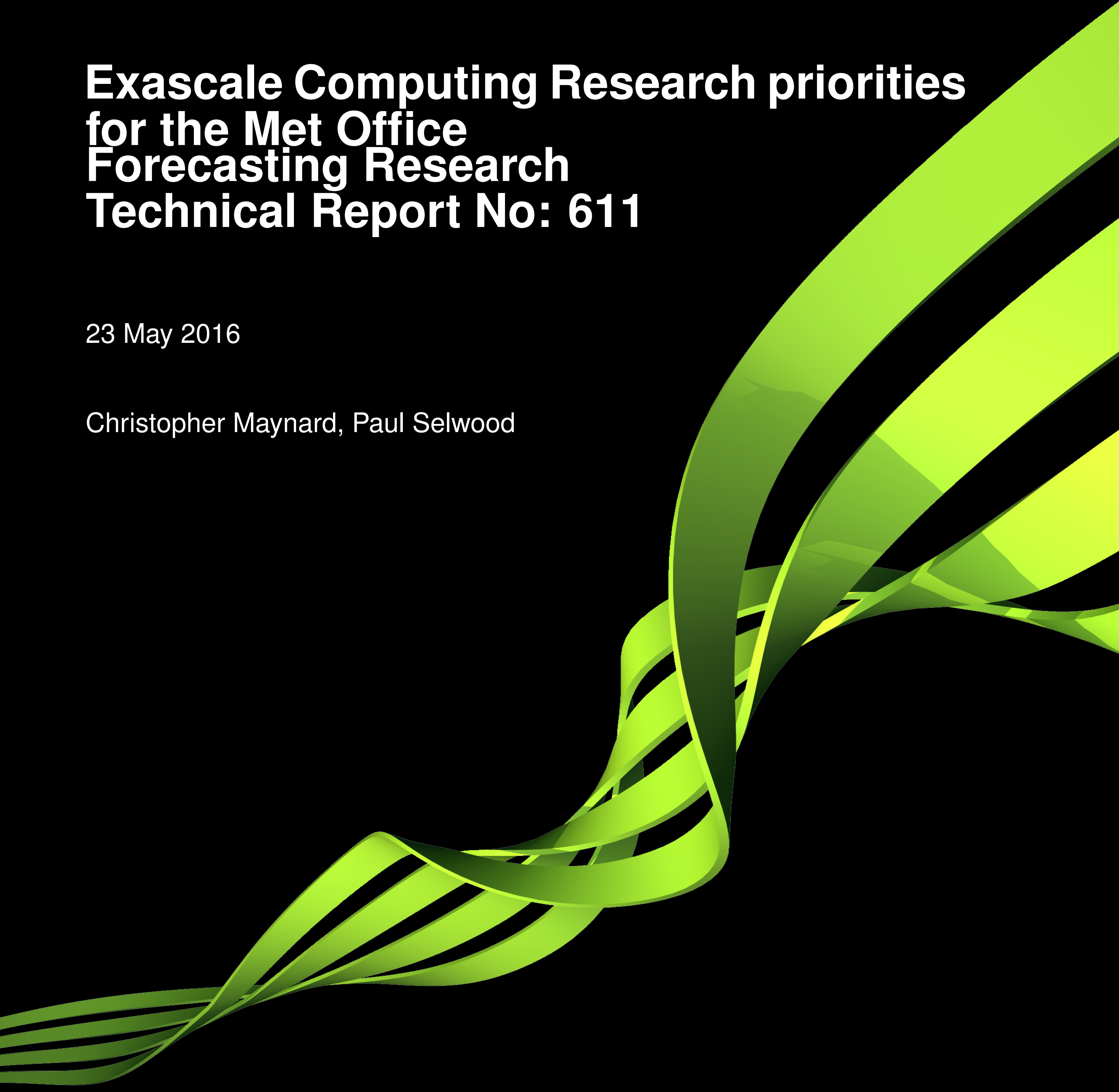
Exascale Computing Research priorities for the Met Office
Forecasting Research Technical Report No: 611

23 May 2016

Christopher Maynard, Paul Selwood



1 Introduction

The paradigm of ever increasing NWP and climate model accuracy is in part predicated on increases in computer processor speed. Whilst Moore's law for transistor size has continued, so that the number of transistors per unit area of silicon has increased with each processor generation, the lack of Dennard scaling has resulted in no increase in single thread performance since 2005. This has led to a massive increase in the number of individual cores or threads in super computers built from commodity processors and a consummate increase in the total electrical power required to run such machines[1]. Moreover, increases in the speed of memory access for DRAM technology has been modest in comparison leading to a decrease in the memory bandwidth available per thread.

To prevent power requirements rising above what is actually available[2] even before the cost of such huge electricity demand is considered it is likely that new processor architectures such as GPUs, Intel Xeon Phi, or as yet to emerge processors based on ARM will be required. See for example the US "pre-Exascale" machines: *Cori* a Xeon Phi generation Knights Landing (KNL) processor machine at NERSC[3], the DOE CORAL initiative [1] which will fund 2 GPU machines (Power9 + NVIDIA Volta GPUs) at Oak Ridge National Laboratory (*Summit*[4]) and Lawrence Livermore National Lab (*Sierra*[5]), and a Knights Hill (KNH) generation Xeon Phi processor machine , *Aurora* [6] at Argonne National Lab. See also the Japanese "post-K" computer[7]. These new architectures exhibit deep memory hierarchies, high levels of shared memory parallelism and large SIMD or Vector units. The programming model for such devices is still evolving and current compiler technology is unable to support vectorising code automatically. Moreover, many more architectural and run-time features are exposed to the application and this breaks down the abstraction of a high level language. Consequently, such new technologies can be considered as *disruptive* technologies. Without extensive re-engineering and optimisation by expert computational scientists, large, legacy science codes will fail to exploit the performance potential of such processors or worse, be unable to run on them at all.

The term *Exascale Computing* has been used to describe a computer with *Exaflops* computational capability. The word *Exascale* is used to emphasise that many aspects of such a machine have to be scaled from current HPC machines, not just the number of floating point operations per second. In this document, Exascale Computing refers to any software or hardware technology which could be used to enable an Exascale computer.

[1] http://www.top500.org/
[2] IT hall 3 has approximately 6MW, with an upgrade to 10-20MW possible
[3] https://www.nersc.gov/users/computational-systems/cori/
[4] https://www.olcf.ornl.gov/summit/
[5] https://asc.llnl.gov/coral-info
[6] http://aurora.alcf.anl.gov/
[7] http://www.aics.riken.jp/en/postk/project

2 Exascale Computing Research

To solve the Exascale computing challenge and deliver software capable of exploiting complex machines for Met Office operations and science research several computational research themes must be pursued. These include:

1. Programming models
2. Novel Architectures
3. Model Coupling: multi-scale and, multi-timescale
4. I/O and Workflow

2.1 Programming Models

HPC codes traditionally are a mix of a high level language, typically Fortran for weather and climate codes and the MPI library. This straightforward programming model is unlikely to be able to exploit sufficient parallelism, particularly node-level and instruction level parallelism (ILP). Indeed, much of the work of the MPI forum to propose an MPI 4 standard[8] is to make the plus of "MPI + X", where *X* is some other programming model, work better so that different programming models fit together. The Met Office doesn't have sufficient effort available to join the MPI forum, however, the weather and climate community could be better represented from a HPC research perspective. Moreover, MPI 3.0 developments have yet to be fully tested in implementations. For example, the memory management for one-sided communications should allow for much improved performance. This asynchronous communication pattern could allow new algorithms to exploit a greater degree of parallelism.

Examples of the *X* programming model include OpenMP and OpenACC which employ directives to annotate source code to control concurrent threading across shared memory. Whilst in general, either could be used to target any multi-core or accelerator architecture, compiler support for the directives divides along processor vendor lines, *viz.* OpenMP for Intel Processors and OpenACC for NVIDIA[9] GPUs. Moreover, the standards and implementations for exploiting ILP parallelism (*e.g.* Single Instruction Multiple Data (SIMD)) and managing non-coherent memory spaces, are still immature. Other X candidates include the so called Partitioned Global Address Space languages. The most well established of these are the language extensions Unified Parallel C (UPC) and Co-Array Fortran (CAF) and the library OpenSHMEM. These aim to bridge the divide between shared and distributed memory programming. Again, compiler and implementation support for these can be immature. Other models include new languages such as OpenCL, CUDA and Chapel, however these tend to be very vendor and thus architecture specific. A further possibility is to couple MPI to itself *i.e.* MPI + MPI, an example of which is the IO server pattern.

[8] http://meetings.mpi-forum.org/MPI_4.0_main_page.php

[9] AMD also claim to support OpenACC.

Whilst research and development into the most promising MPI+X candidate(s) is necessary, a higher level approach based on the Domain Specific Language (DSL) abstraction is also important. The Gung Ho/LFRic project is taking this approach, by employing an abstraction of the Finite Element Method and implementing a layered software architecture. The parallelism is expressed in the middle layer and thus hidden from the science code behind an interface. Generated parallel code can then use different programming models to target different architectures. Hiding the parallel complexity behind interfaces is probably a good idea but if the abstraction can be generalised then other applications besides the dynamical core can use it. Moreover, this abstraction could be used for more general development. Firstly, by feeding back into the programming model standards development. Secondly by looking at how this approach can merge directly into compiler technologies such as LLVM[10].

So called Big Data/Data analytics computations have advanced rapidly in recent years. Whilst some of these ideas are not directly applicable to HPC, some of the emerging programming models are highly parallel. What can traditional HPC learn from these techniques? How could numerical modelling approached with, for example *Spark*[11]. Can machine learning techniques be deployed to improve parameterisation development? What other techniques and methods can cross-over or merge with HPC?

2.2 Novel Architectures

Total power requirements suggest that CPUs will not be suitable commodity processor for supercomputers in the future [2]. Intel Xeon Phi and NVIDIA GPUs, are the CPU alternatives which are the most developed. Others, such as AMD GPUs or a 64-bit ARM scientific processor are in existence or being developed. For these processors the research is the programming model and developing applications and is described above. However, other processors and architectures may be developed that may be of interest.

Field Programmable Gate Arrays (FPGAs) are a well established technology but difficult for applications developed using high level languages to use [3]. Two potential research avenues include firstly developing a software stack to transform high level language code and transform it into hardware logic for FPGAs. There has been research in this area, typically using C as the high level language but with limited success. Secondly, developing FPGA logic for solving common computational patterns in scientific codes, for example matrix-vector operations common in linear solvers. Thus only using power for circuits in silicon which are heavily used.

Other novel architectures include the D-WAVE Quantum processor[12], which can in principle compute the entire space of a minimisation problem, albeit with some non-trivial restrictions on how that minimisation problem is defined. Many scientific computations can be reformulated as

[10] http://llvm.org/
[11] For an implementation see http://spark.apache.org/
[12] http://www.dwavesys.com/

minimisation, two notable computations are Data Assimilation and linear solvers. How can such a non-traditional processor be used?

Quantum Optical computing deceives are beginning to become a reality, for example in silicon photonics [4]. Research into how such devices might be exploited [5] to perform simple computations, how can they be coupled to existing software stacks and what new algorithms might be possible.

2.3 Model coupling: Multi-scale and Multi-timescale

In NWP and Climate models the Atmospheric model and Ocean Model are two separate and simultaneously executed models. They are coupled together using a third executable called a *coupler*. At certain pre-defined times they exchange information via the coupler which translates the data (*re-gridding etc*) from one to the other. These simulations can run at different resolutions and thus different scales. This concurrent task parallelism is currently used for science reasons but exploiting greater parallelism could be used for parallel performance. To enable this, research into both the science of how to combine processes with different length and time scales and computer science of how to manage, move and transform the data asynchronously is required. In particular, how to achieve some kind of work load balance whilst minimising data movement and managing an update scheme with super and sub time-stepping.

Other science communities attempt to solve multi-scale problems with multiple coupled simulations *e.g.* the Virtual Physiological Human Project[13]. Collaborations with other communities might be possible.

The CFL condition, which imposes restrictions on the size of the time step, will ultimately limit the weak scaling of numerical models. Novel algorithms such as those which exploit parallelism in the time-domain [6] may allow this limit to be circumvented. Besides research into new algorithms, computer science research into how to manage and couple a temporal ensemble of models is required.

2.4 I/O and Workflow

For an exascale computing system the energy cost of moving data will be high. Reading and writing data from distributed file systems[14] to (differently) distributed memory systems will be expensive (in terms of energy) and time consuming. The check-point/restart model of fault-tolerance may not be feasible if some of the worse case predictions for reliability of machines made from extremely large numbers of components and I/O rates are to be believed. New research into I/O methods that are parallel and asynchronous including how check-point/restart can be revised for such an environment is required.

[13] http://www.vph-institute.org/

[14] such as LUSTRE

Non-Volatile Memory (NVM) is a new storage/memory technology which is becoming available under various proprietary brand names. Solid state storage technology has been used for storage instead of disk for some time. NVM is similar, but rather than storage it is addressable memory whose state persists after the process has exited. This can be viewed as deepening the storage hierarchy from processor registers, multiple levels of cache, addressable “fast” memory, slow (main) memory, NVM, virtualised, parallel file system, multiple disk storage components and tape. What are the algorithms and what should the I/O software stack look like to exploit such a hierarchy?

A workflow analysis of data flow through Met Office processes reveals the same data is read and written - possibly multiple times. Effectively, different processes which run sequentially are coupled together through the file system. This likely to be more inefficient in the future than it is at the moment. For example, Satellite data arrives from external feeds and is written to disk. It is read from disk and processed this is then written to disk. It is then read from disk and combined with NWP model output (data assimilation (DA)) and written to disk. The NWP model then reads this from disk, and a model run performed. The result is written to disk. This model output is read by DA (see above) and various post-processing models which produce forecast products. How can NVM be used in a work flow to reduce or avoid the IO in the current workflow? In effect coupling sequential processes through NVM.

3 Tools and Training

Software tools for exploiting Exascale computers will become increasingly important, as automation will be critical to use extremely complex machines. For example, it is already necessary to *measure* the raw performance characteristics of the Cray XC40, so dynamic is the run-time environment. See also [7]. The Met Office already uses profilers, debuggers and workflow management tools (*e.g.* FCM and Rose) but modifying/adapting existing tools and developing new tools will be necessary. Combined together, these tool chains will require scientist to be trained in how to use them. Moreover, training in new HPC techniques developed from the research will need to be delivered to Met Office scientists and software engineers.

4 Summary

Exascale Computing represents a disruptive change to technology. The “free lunch” of increasing processor speed that the Met Office (and science in general) has relied upon to deliver more science capability is over. Performance gains will now only come through parallelism. This both difficult to achieve and requires HPC/computer science research into methodology. Thereafter, such solutions have to implemented for Met Office software systems to be enabled for exascale computing.

References

[1] DOE Office of Science, NNSA: Fact sheet: Collaboration of Oak Ridge, Argonne and Livermore. http://energy.gov/sites/prod/files/2014/12/f19/CORAL%20Fact%20Sheet __FINAL%20AS%20ISSUED_UPDATED.pdf (2014)

[2] Kogge, P.: Updating the energy model for future exascale systems. In Kunkel, J.M., Ludwig, T., eds.: High Performance Computing. Volume 9137 of Lecture Notes in Computer Science. Springer International Publishing (2015) 323–339

[3] Baxter, R., Booth, S., Bull, M., Cawood, G., Perry, J., Parsons, M., Simpson, A., Trew, A., McCormick, A., Smart, G., Smart, R., Cantle, A., Chamberlain, R., Genest, G.: Maxwell - a 64 FPGA Supercomputer. In: 2nd NASA/ESA Conference on Adaptive Hardware and Systems, Edinburgh, IEEE (August 2007) 287–294

[4] Masada, G., Miyata, K., Politi, A., Hashimoto, T., O'Brien, J.L., Furusawa, A.: Continuous-variable entanglement on a chip. Nat Photon **9**(5) (May 2015) 316–319 Letter.

[5] Maynard, C., Pius, E.: A quantum multiply-accumulator. Quantum Information Processing **13**(5) (2014) 1127–1138

[6] Haut, T., Wingate, B.: An asymptotic parallel-in-time method for highly oscillatory pdes. SIAM Journal on Scientific Computing **36**(2) (2014) A693–A713

[7] Beckman, P.: Software for Exascale. http://www.easc2015.ed.ac.uk/program-archive/slides/s3Beckman.pdf (2015) Keynote presentation, EASC2015.

Met Office
FitzRoy Road, Exeter
Devon, EX1 3PB
UK

Tel: 0370 900 0100
Fax: 0370 900 5050
enquiries@metoffice.gov.uk
www.metoffice.gov.uk

Produced with the Met Office LaTeX template v2.0.0