MANCHESTER 1824

The University of Manchester

Proceedings of the

EMerging Technology (EMiT) Conference 2015

30 June – 1 July 2015

University of Manchester, U.K.

Edited by B.D. Rogers, D. Topping, M.K. Bane

NVIDIA OCF POLYHEDRON SOFTWARE hp EAO intel

Harwell Imaging Partnership
Science & Technology Facilities Council

http://emit.manchester.ac.uk

MANCHESTER 1824

The University of Manchester

June 2015

Foreword to the 2015 Emerging Technology (EMiT) Conference

Dear Delegate,

The University of Manchester is delighted to welcome you to the 2015 Emerging Technology (EMiT) Conference.

EMiT 2015 is a ground-breaking conference on how emerging computational technology is helping to accelerate today's research and tomorrow's applied computing. Hardware for computing and how it is used by researchers and industry are going through a period of radical change. Energy efficiency and future proofing are becoming key issues for researchers, software developers and industrial users of applications. Questions of how to take advantage of new technology and plan strategically for the future are now of paramount importance.

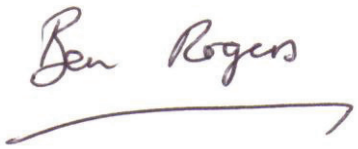
The aim of EMiT is to bring together leading key figures in the computing communities, the end users of new software & hardware in industry and university research along with the vendors from across the international arena to address the following objectives:

- identify latest trends in development for novel computing;
- share how best to exploit Emerging Technology for application;
- focus on new techniques, their development and transfer to new areas.

We have a very exciting programme of talks from different disciplines and we hope you enjoy the conference.

This event could not have taken place without the support of the University of Manchester and the Science and Technology Facilities Council (STFC), and the hard work of all the members of the Organising Committee.

Thank you for your participation.



Benedict Rogers
Chair of the Organising Committee of 2015 Emerging Technology (EMiT) Conference
University of Manchester

Committee

Organising Committee

• Dr Benedict D. Rogers	Chair, University of Manchester, School of Mechanical, Aerospace & Civil Engineering (MACE)
• Dr David Topping	University of Manchester, School of Earth, Atmospheric & Environmental Sciences
• Prof. John Keane	University of Manchester, School of Computer Science
• Dr Stephen Longshaw	Science & Technology Facilities Council, STFC
• Dr Stefan Güttel	School of Mathematics, University of Manchester
• Prof. David Emerson	Science & Technology Facilities Council, STFC
• Dr Michael K. Bane	University of Manchester, IT Services
• Irfan Alibay	University of Manchester, School of Pharmacy
• Dr Robin Pinning	University of Manchester, IT Services
• Julie Samson	University of Manchester, School of Earth, Atmospheric & Environmental Sciences
• Richard Mervin	University of Cambridge
• Dalinder Sall	University of Manchester, IT Services
• Cerian Lindsey	University of Manchester, IT Services
• Naheed Akhtar	University of Manchester, School of Mechanical, Aerospace & Civil Engineering (MACE),

Acknowledgements

The Committee would like to express its sincere gratitude to the following whose support was essential to make this conference happen:

School of Mechanical, Aerospace and Civil Engineering (MACE)
School of Earth, Atmospheric & Environmental Sciences (SEAES)
University of Manchester, IT Services

Printing

"The Organising Committee of the 2015 EMiT Conference" shall not be held responsible for any statement or opinion advanced in papers or printed in this volume. The authors' papers have been prepared for final reproduction and printing without any reduction, correction, etc. The authors are fully responsible for all the information contained in their papers.

ISBN 978-0-9933426-0-8

CONTENTS

SpiNNaker and the Human Brain Project

Steve Furber

School of Computer Science
The University of Manchester
Manchester, UK
steve.furber@manchester.ac.uk

***Abstract*—The SpiNNaker (Spiking Neural Network Architecture) project aims to produce a massively-parallel computer capable of modelling large-scale neural networks in biological real time. The machine has been 15 years in conception and ten years in construction, and has far delivered a 100,000-core machine in a single 19-inch rack, which is now being expanded towards the million-core full system. Even with a million processor cores the machine will be capable of modelling only perhaps 1% of the network complexity of the human brain, underlining the scale of the challenge facing the EU Human Brain Project, which has an objective of developing technologies capable of supporting full brain models.**

I. INTRODUCTION

The human brain remains as one of the frontiers of science, and its information processing principles remain a mystery despite its central role in human existence. A great deal is known about the details of the neuron – the fundamental brain cell unit from which the brain is constructed – and imaging machines tell us a lot about the movement of activity within the brain at a coarse level, but the only instrument that gives us access to the intermediate scales at which information is stored, processed and communicated is the computer model. Unfortunately, conventional computers are poorly matched to the challenges of brain modelling as they cannot efficiently capture the very fine-grain communication that takes place between neurons in the brain.

SpiNNaker [1] is a massively-parallel computer based upon small ARM processor cores (Fig. 1) with an interconnect architecture inspired by the challenge of matching the brain's fine-grain communication capabilities. Its primary intended application domain is in brain modelling, though as in all other respects it looks like a conventional parallel computer (with some restrictions, such as no floating-point hardware) it is interesting to consider whether there might be other application domains for which it is also suited.

The Human Brain Project (HBP) is one of two successful proposals to the EU ICT Flagship programme. With a core budget of €500M (which various complementary funding schemes are expected to double) the HBP has ambitious goals under "Future Neuroscience, Future Medicine and Future Computing" top-level headings, among which are the creation of several ICT "platforms" to support neuroinformatics, (brain-related) medical informatics, high-performance computing, neurorobotics, and neuromorphic computing – novel models of computation based on neuroscience [2]. The Neuromorphic Computing Platform will incorporate two pre-existing technologies, each of which brings a decade or more of prior development to the HBP party. One is a wafer-scale analogue neuromorphic system from Heidelberg University, and the second is the many-core digital system – SpiNNaker – from the University of Manchester.

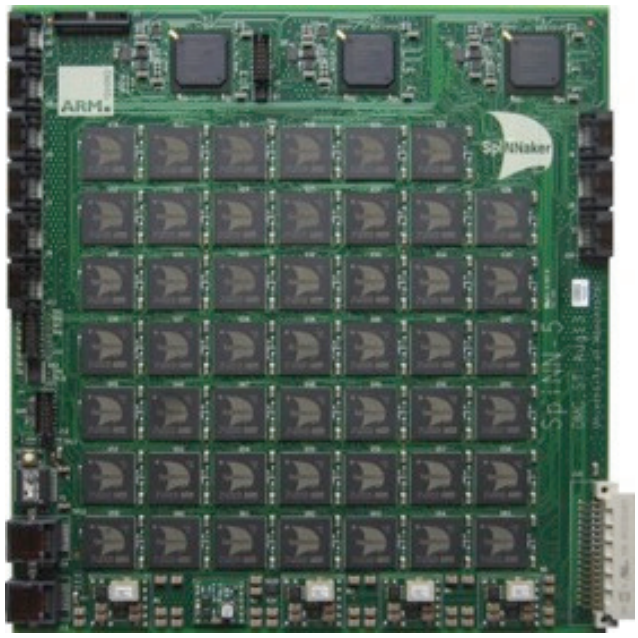


Figure 1. A 48-node SpiNNaker circuit board incorporating 864 ARM processor cores

II. COMMUNICATING WITH SPIKES

Neurons – brain cells – can be viewed as multiple input, single output devices. They accumulate inputs from a very large number – typically of the order of 10,000 – of sources, apply some sort of non-linear process to evaluate those inputs, and signal their response to those inputs by generating an electrical impulse – a spike – along their axon, a "wire" that

Exploring Deep, Hierarchical Pipelines and High-level Synthesis on FPGAs for Accelerating the Emanuel Convection Scheme

Kristian Thorin Hentschel, Syed Waqar Nabi, Wim Vanderbauwhede

School of Computing Science,
University of Glasgow, Glasgow, UK

***Abstract*— The potential of FPGAs for High-Performance Computing is increasingly being recognised, but most work focuses on acceleration of small, isolated kernels. We present a parallel FPGA implementation of a scientific legacy algorithm, the seminal scheme for cumulus convection in large-scale models developed by Emanuel [1]. Our design makes use of pipelines both at the arithmetic and at the logical stage level, keeping the entire algorithm on the FPGA. We assert that modern FPGAs have the resources to support this type of large algorithms. Through a practical and theoretical evaluation of our design we show how such an FPGA implementation compares to a multicore CPU implementation using OpenMP.**

I. INTRODUCTION

Many scientific fields, such as weather modelling, have started to make use of modern high-performance computing (HPC) architectures such as GPUs many-core systems and FPGAs, yet there still exists a large body of legacy code. These original algorithms need to be tackled for parallelisation to continue providing performance and efficiency gains. When comparing HPC architectures and performance figures, most benchmarking approaches use small, well-isolated algorithms that operate uniformly on a large amount of data. These kernels are carefully selected and extracted to explicitly target execution-time hotspots in a larger application.

By contrast, in this work we report the porting of a widely used convection algorithm [1] from single-threaded Fortran code to a pipelined FPGA architecture using a high-level synthesis (HLS) tool. Making use of a streaming paradigm and targeting fine-and coarse-grained pipeline parallelism, we analyse and port a complete convection algorithm. We argue that modern FPGAs have the capacity and performance required for dealing with such considerably more complex floating point algorithms, and explore options for optimization of our design. The convection kernel is part of the FLEXPART lagrangian particle dispersion simulator [2] used a.o. to predict the spread of the ash cloud from the Icelandic volcano in 2010 and the radio-active fall-out from the Fukushima reactor incident in 2011. We have ported the rest of the simulator to OpenCL but the express aim of this work is to explore the issues in porting a real-life, large and complex kernel to FPGAs.

II. PRIOR WORK

Recent FPGA, GPU, and CPU performance surveys such as that by Vestias et al [3] show that FPGAs are competitive for certain workloads, such as some map-reduce applications and custom state-machines. More importantly, the flexible architecture allows for custom pipeline depths and has more efficient support for branches and loops when implemented as a high-throughput streaming design. In analysing performance of these architectures, or when demonstrating HLS tools, most work focuses on relatively small examples, which can be efficiently mapped as custom state-machines or data-parallel designs: Hussain describes a k-means ranking algorithm [4] implemented as a parametrized and showing the FPGA's capability's for saving space and energy through custom number representations. Lienhart show an implementation of the n-body simulation [5], aiming for high throughput. Our work, in contrast, while not yet claiming optimal performance, shows that complete, significantly larger (in terms of lines of code) and more complex (in terms of control flow) floating point algorithms can also be implemented within the resource limits of modern FPGAs. Our approach, based on pipelining at a coarse level (streaming multiple data sets), and a ne-grained arithmetic level (streaming individual layers through a loop), faces similar considerations to those described by Gordon [6]. They apply coarse software pipelining to a multi-core CPU system and develop optimization strategies for determining the best mapping of a set of streaming computations to data-parallel or task-parallel sections.

III. APPROACH

We analysed the original Fortran code for data flow dependencies between blocks, restructured it following the original top-level do-loops, and informally expressed the parallelism in each block as a set of map and reduce operations. Some of the nested loops were merged or split into multiple kernels. For an initial exploration, we dealt with the set of loops towards the end of the algorithm, representing about 60% of execution time and over 30% of

Inexact hardware for numerical simulations of weather and climate: What is possible, why it is useful and how to do it

Peter D. Düben, Stephen Jeffress, T. N. Palmer

AOPP, Department of Physics
University of Oxford
Oxford, UK
dueben@atm.ox.ac.uk

Abstract— **We have studied several approaches to trade numerical precision against performance in numerical models that are used in weather and climate research. This paper provides an overview of what has been tested so far, the potential and limits for the different approaches, how to integrate the use of inexact hardware in model development and the reasons why the use of inexact hardware in simulations of atmosphere and ocean is promising.**

I. Introduction

Numerical models of the atmosphere and ocean are essential to provide meaningful forecasts of future weather and climate. Reliable forecasts of weather and climate are of enormous importance for the preservation and generation of prosperity in society.

The quality of global atmosphere and ocean models depends on the used resolution which is the minimal length scale that can be resolved in model simulations. In modelling, resolution is commonly measured as the typical length of the edge of grid cells in horizontal direction. If a model is running with a higher resolution, more small-scale features can be resolved and more accurate forecasts are possible. This is of particular importance since both the atmosphere and ocean are chaotic, turbulent systems for which physical phenomena at very small scales (often discussed as the flap of the wing of a butterfly) will eventually influence large scale dynamics.

However, the resolution of atmosphere and ocean models is limited by the computational performance of today's supercomputing centres, and weather and climate models are already running on the fastest supercomputers of the world.

During the last couple of decades, model developers could trust in an exponential growth of computational power which allowed the use of ever-increasing resolution. However, Moore's law appears to be under threat due to limits in the size of transistors that can be used in silicon technology. It seems that even an excessive use of parallelism will not keep up the exponential growth in computing power in the near term future. Also a strong increase in the numbers of processors that are used in parallel for model simulations will cause a strong increase in power consumption, which is already a significant cost factor for today's high-performance-computing centres.

In this paper, we explore the use of inexact hardware in numerical weather and climate models. Inexact hardware is promising a reduction of computational cost and power consumption of supercomputers, and could be a shortcut to exascale supercomputing and higher resolution forecasts. Simulations with inexact hardware will introduce numerical errors, such as rounding errors or bit flips. However, if model simulations are not influenced significantly by these errors, the possible increase in resolution would improve forecast quality and bring us closer to global model simulations at a resolution of 1 km. Simulations at such a resolution would allow an explicit representation of important cloud formations and deep convection, which would provide a significant improvement in weather and climate forecasts.

II. Why should/can inexact hardware be used in weather and climate models

It is a very difficult challenge to make accurate predictions of future weather and climate. This is mostly due to the overwhelming complexity of the earth-system which includes the dynamics of the atmosphere, oceans and sea-ice as well as atmospheric chemistry, hydrology, ice on land, biogeochemistry and more. Furthermore, the chaotic nature of the atmosphere and ocean guarantees that numerical weather simulations will diverge from the real state of the system with increasing forecast lead time. Even a perfect model could not prevent this, since we cannot avoid errors in model initialisations which are based on the measurements available. For climate predictions we will never be able to forecast a particular state of the atmosphere (the weather) in the far-away future. It is only possible to predict the mean state of future climates. Naturally, the analysis of such a mean state will be limited by the amount of data which is available for statistics.

In a model simulation, it is impossible to represent every detail of all components of the earth-system. Parts of the model

FPGA processing for High Performance Computing

Prabu Thiagaraj[1], Benjamin Stappers[2], Jayanta Roy[3], Michael Keith[4], Mitchell Mickaliger[5]

Jodrell Bank Centre for Astrophysics, School of Physics and Astronomy
University of Manchester
Manchester, UK
[1] prabu.thiagaraj.manchester.ac.uk [2] Ben.Stappers@manchester.ac.uk [3] jayanta.roy@manchester.ac.uk
[4] mkeith@pulsarastronomy.net [5] mitch.mickaliger@gmail.com

***Abstract*—Field Programmable Gate Arrays (FPGAs) are fine-grained, massively parallel, digital logic arrays with architecture suitable to execute computations in parallel. Although FPGAs have been in existence for more than two decades and known for their inherent ability to perform fine grain parallel processing tasks very efficiently, it is only in the last couple of years we could see the realization of their potential in the high-performance computing world. This transformation is mostly due to the recent and radical progress in the FPGA development tools and in the hardware technology. This paper outlines this evolution, touches upon the specific tools and vendor technologies that made this transformation possible and attractive. This paper outlines the program development cycle for an example of FPGA-based processing. The second half of the paper will present how this new technology appears attractive especially due to the power efficiency for a signal processing application in radio astronomy, namely for the Square Kilometre Array (SKA) that we are involved in developing at the Jodrell Bank Centre for Astrophysics, School of Physics and Astronomy, University of Manchester.**

I. INTRODUCTION

Field Programmable Gate Arrays (FPGAs) are designed using fine-grained, massively parallel arrays of digital logic with architecture suitable to execute computations in parallel. While FPGAs of early days offered handy solutions for simple digital circuitry, the present generation of FPGAs promise energy efficient solutions for computation intensive problems. Although high-capability FPGAs have been in existence for more than a decade, only in the last couple of years we could recognize their roles in the high-performance computing world. The reasons for this includes, availability of well matured FPGA development tools, electronic design automation processes, and a radical approach in adapting to a host- device concept for using the FPGAs [1].

In the wake of this new era, we are considering to evaluate an FPGA-based solution for a computation intensive application in radio astronomy, namely to search for pulsars using the upcoming Square Kilometre Array (SKA) telescope [2]. This paper outlines the specific tools and vendor technologies that transformed the FPGAs into an accelerator for computers, and show how it is appealing to a digital signal processing problem in radio astronomy.

II. A NEW ACCELERATOR FOR HPC

The introduction of FPGAs by Bill Carter, Xilinx ® in 1985 modified the traditional approaches to circuit design [3]. The changes introduced can be compared to the impacts after the programmable loom of Jacquard [early 1800s] and after the universal machine of Turing [1937]. Use of an FPGA involved programming it to represent a circuit of interest. A suitable mechanism was required to describe large number of serial and parallel (concurrent) operations of the circuit to be implemented, and a provision to make analysis and synthesis of the circuit operation. At the introduction of the first FPGA, the electronic design automation mechanisms and languages to support such automation were already well established. In other words, FPGA program development tools could be based on: a) existing hardware description languages (HDL) that conceptually evolved from of the data transfer commands of the first processors, b) state control sequencer familiar in programmable logic controllers (PLC), c) design descriptions and analysis methods used in application specific integrated circuits, and d) fuse map generation schemes of programmable logic devices (PLD). FPGA tools that evolved with this background, gave easy implementations for complex circuits. However, operating these tools demanded accurate knowledge about the underlying hardware architecture of the FPGA, and incurred long design cycle to even reach a first-level implementation for complex circuits. FPGA designers were also faced with difficulties to exploit the resources of large FPGAs in reasonable time. At this juncture, a relief came to the developers in the form of automated HDL generators. These are based on code generation scripts from Matlab® like environments. HDL generators could play significant role in generating large designs and helped in exploited the capabilities of large FPGAs. While these design approaches worked well and appeared sufficient for dedicated and single FPGAs platforms, circuits that required use of multiple FPGAs, and for developing FPGA circuits where FPGAs operated alongside of other computing hardware such as central processing units (CPU), graphics processing units (GPU), digital signal processors (DSP) and etc., faced difficulty. In such heterogeneous platforms, where FPGAs need to interact between each other or with a conventional

Communication avoiding algorithms

Laura Grigori

Laboratoire J.L. Lions, UMR 7598
Universite Pierre et Marie Curie
Paris, France

Abstract—The increased communication cost is one of the main challenges we need to address in high performance computing today. There is an exponentially increasing gap between the time required to perform arithmetics and the time required to transfer data between different levels of the memory hierarchy or between different processors. In a quest to address this problem, communication avoiding algorithms aim at reducing drastically, or even minimize when possible, the volume of communication and the number of messages exchanged during the execution of an algorithm. This talk will review first communication avoiding algorithms for dense linear algebra, and then will focus on Krylov subspace methods and associated preconditioners.

Energy Aware Scheduling and Power Measurement on BlueWonder

Mr. Neil Morgan (STFC), Dr. Vadim Elisseev (IBM), Dr. Manish Modani (IBM), Prof. Terry Hewitt (STFC), Dr. Luigi Brochard (Lenovo), Dr. Francois Thomas (IBM).

Science and Technology Facilities Council, The Hartree Centre, STFC Daresbury Laboratory,
Sci-Tech Daresbury, Warrington WA4 4AD, UK.
neil.morgan@stfc.ac.uk

***Abstract*—This paper describes some initial experiences in setting up and deploying the energy aware features of the Platform LSF 9.1.2 resource management system on an IBM NextScale Cluster 'Blue Wonder'.**

I. INTRODUCTION *(MOTIVATION)*

Currently the power consumption of the world's leading supercomputers is of the order of tens of MegaWatts (MW). It is commonly accepted that the power consumption of sustainable Exascale computing needs to stay in a 20 MW range. Therefore, increasing the energy efficiency of HPC systems is one of the main goals of the HPC community. This paper presents our experiences of reducing the energy consumption of supercomputers with the use of energy aware workload management software that relies on thorough power monitoring hardware and detailed application performance monitoring. The immediate goals of our research are three-fold: to validate Energy Aware Scheduling (EAS) implementation by the IBM Platform LSF workload management software, to reduce power bills at Hartree Centre, UK, to change users behavior towards being more "energy conscious".

Every modern CPU now has the ability to vary processor clock frequency and supply voltage, according to the actual demand [2], [3]. This technology is commonly referred to as Dynamic Voltage and Frequency Scaling (DVFS) and is the topic of this paper: the energy aware scheduling features of IBM Platform LSF. It has been demonstrated that it is possible to achieve an acceptable trade-off between energy savings and performance [4,6] by running applications at different processor frequencies. IBM Platform LSF allows to take advantage of such trade-offs at scale by automatically selecting optimum CPU frequency for an application using various EAS policies and performance and energy consumption prediction model.

We describe the model used by the IBM Platform LSF to predict the power and performance of applications running at different processor frequencies on a cluster of standard x86 servers. The model is similar to the one used by the IBM LoadLeveler [5]. We also present how we apply the prediction model to reduce the energy consumption of the BlueWonder. The supercomputing resource which will be used throughout is the recently installed IBM NextScale Cluster, BlueWonder, which is located at the Daresbury Laboratory, UK. It consists of 365 nodes connected via FDR14 InfiniBand and provides 8760 Intel processor cores as well as 24 TBytes of main memory to its users. Each compute node is diskless and features two Xeon E5-2697 v2 CPUs with 12 cores each running at up to 2.7 GHz and 64GB of main memory configured in 8 x 8GB 1866 MHz DIMMs. In addition to the Infiniband network, the nodes are connected via Gigabit Ethernet for booting and remote management..

II. LSF PREDICTION MODEL

We concentrate on modelling and optimizing the power consumed by BlueWonder's compute nodes consisting of standard x86 server hardware. Where measuring node power is necessary, we use NeXtScale paddle cards. The paddle cards measure the consumed energy of each node at the power supply unit. By providing energy values, measuring with paddle cards causes only little interference with the applications running on the system since the energy counter needs only be sampled when applications start and end. The mean power consumption can then be derived from dividing the runtime into the consumed energy.

To predict the power and performance of applications running at different frequencies, our model takes into account the compute nodes and the application. For simplicity, we analyze application behavior as a whole, focusing on the mean power and performance during the runtime. Looking first at power consumption, we know that the power consumed by an application when running on a server varies with the node configuration and the type of application. In earlier experiments [5], we found that node power consumption consists of two parts: a dynamic part which relates to the activity of the processor, cache, memory, etc. and a static part which is unrelated to any activity and originates mostly from the south bridge chip. This static part is dominating the power consumption when the node is idle. We also found that the processor power consumption varied according to the average number of Cycles Per Instruction (*CPI*) required during the application run and the memory power consumption varied according to the total read and write memory bandwidth (*GBS*) of the application. This is equivalent to the commonly accepted characterization of application behavior under frequency scaling depending on memory or compute boundedness.

Based on these findings, the power consumption of an application on a particular compute node at frequency f_n can be predicted from a measurement performed at a reference frequency f_0:

$$PWR(f_n)=A_n*GIPS(f_0)+B_n*GBS(f_0)+C_n \quad (1)$$

Real-time Social Analytics in GPU-based Multi-Dimensional Databases

Peter Strohm
Research & Development
Jedox AG
Freiburg im Breisgau, Germany
Peter.Strohm@jedox.com

Alexander Haberstroh
Research & Development
Jedox AG
Freiburg im Breisgau, Germany
Alexander.Haberstroh@jedox.com

Abstract— **Analytical queries in databases often involve calculations of extremely large areas of aggregated values as input for further processing like conditional calculating (if-then-else) or top-k evaluation and therefore often run into memory problems when calculated on Big Data sets like Social Media entries. We present the design of optimized condition-based processors in large data sets combined with a floating frame approach to stream through these data areas. Conditional calculations are especially useful to split large value sets into clusters for further analysing or aggregating. With our multi-dimensional GPU-based database approach we are able to accelerate these analytical queries extensively on real world social media data including localized Twitter trends and Wikipedia page hits.**

I. Introduction

Social Media Networks in the World Wide Web provide more and more data through to extensive usage of mobile apps and increasing user numbers. These data is already stored and analysed in many ways but for the massive amount of the data it can't be stored or used for long time. Also real-time analysis and connecting historical data to newest web entries isn't possible without an immense consumption of calculation power and therefore extensive hardware costs. So most of the data remains unused and information is lost. Automated intelligent analysis of these Big Data packages could help to improve the quality of the collected data and reveal new ways of Social Analytics by connecting historical data to latest events in real-time. With the GPU accelerating approach for our multi-dimensional OLAP Server we now provide a tool to make analysing, prediction and visualizing Big Data from Social Media possible without the need of intense hardware wastage.

II. GPU-based OLAP Server

A. Multi-dimensional OLAP Server

Multi-dimensional databases allow users to analyze data from different perspectives by applying OLAP operations such as roll up, drill down, or slice and dice. Handling very large dimensions with hundreds of thousands or even millions of elements requires filter methods in order to show users only those data that are relevant to their analytical needs. Apart from simple filters based on characteristics of the elements themselves (e.g. element names, hierarchy levels, etc.), more advanced methods can filter elements based on conditions regarding the values of the (numeric) measures associated with the elements and stored in the OLAP cube.

An example from the sales analytics domain would be to include only those products in a calculation of which at least k units were sold in any store (or, alternatively: all stores) during a certain time. Filters involving such conditions can be expressed as IF-THEN-ELSE rules on whole data sets. Other filters might require that an aggregated value such as SUM, MIN, MAX, or AVG fulfils a condition; those are called *aggregation filters*. There can also be the need of only returning elements (e.g. stores) where a certain number of any (all) product were sold. These filters are called *data filters* with ANY/ALL condition.

The calculation of all such filters involves the processing of very large amounts of data. Essentially, for each of the input elements to be filtered, an (n-1)-dimensional slice of an n-dimensional cube must be scanned with respect to the given condition(s). The slices may consist of aggregated values, which have to be computed first (cf. Fig. 1).

The Potential for Real-time Computational Fluid Dynamics via GPU acceleration

Alistair Revell
School of Mechanical, Aerospace & Civil Engineering
The University of Manchester
Alistair.revell@manchester.ac.uk

***Abstract*— With the abundance of affordable energy efficient hardware such as Graphical Processing Units (GPUs), the potential for industrial exploitation of high performance computing is drastically increased. In the past decade, emerging technology has motivated the development of novel numerical simulation methods for Computational Fluid Dynamics (CFD); designed to leverage maximum performance in parallel computation. As a result, impressive accelerations of CFD software have been reported, and the margin is increasing further with each new technology release. Furthermore, the rise in related scientific research in this area has demonstrated that the prevalence and accessibility to efficient CFD simulations on GPU is ever increasing, with an associated spread of applications outside the realms of pure computer science. In this paper we discuss the background for fast-fluid simulation tools and the potential for Interactive or 'Real-time CFD', including a review the main differences it encompasses and the various forms it has taken to date. We assess the potential that it could offer in the near future to applications across society, industry and research.**

I. INTRODUCTION

The challenges posed for high performance computing increase relentlessly as fidelity and complexity requirements of the scientific community are pushed ever higher. The past decade has witnessed a sustained and diverse development of Computational Fluid Dynamics (CFD) software for novel and emerging hardware across a range of disciplines. CFD algorithms are particularly well-suited to acceleration on GPU, or multi-stream architectures, on account of the inherent data parallelism necessitated by large numbers of iterations via the solution of huge arrays of simplified, *discretized*, equations. Computing technology has evolved specifically to address these requirements, and the application programming interfaces have been refined to improve accessibility.

Early fluid simulations on GPU architectures were developed for computer gaming and animations, based on simple but visually convincing approximations. Before long researchers had developed a range of solvers based on increasingly accurate methods such as the inviscid Euler equations (e.g. Hagen et al 2006) and Navier-Stokes formulations (e.g. Elsen et al 2008). Subsequently there was a rapid growth in the exploitation of manycore hardware for engineering software applications and attempts were made to implement a range of different CFD methods. The numerical simulation of fluid flow encompasses a plethora of numerical approaches, some more popular than others, with each generally intended for a specific purpose and application. Those approaches most suited to GPU computing have two general characteristics; explicit rather than implicit solution schemes and structured data arrangements rather than unstructured (Vanka et al. 2011). Manual software optimisations making use of shared memory, asynchronous memory transfers and coalesced read/write requests can deliver order of magnitude speedups versus naïve implementations. However such manual optimisations can take time, and in some cases OpenACC directives can achieve almost the same performance with much less implementation effort, particularly for larger problem sizes, (Niemeyer et al., 2014).

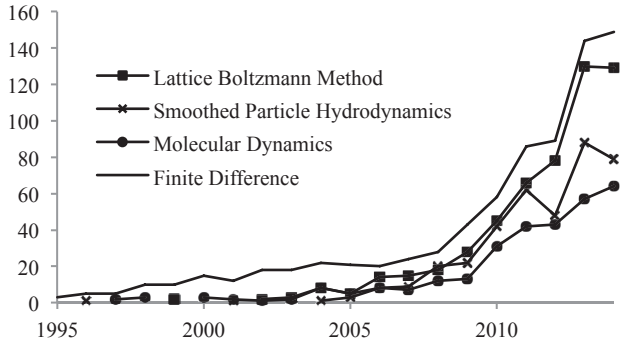


Figure 1: The number of scientific publications reporting both CFD methods and GPU related terms over recent years.

Figure 1 provides some insight into the evolution of GPU activity in CFD over the past two decades, where plotted data is taken from Scopus using a systematic search for CFD and GPU[1] combined in turn with each of four of the most prominent numerical methods. Early Finite difference publications are relating to work on gaming and video animations, while after 2005 there is a notable growth in activity corresponding to the uptake by the scientific community. First generation CFD experiences on GPU were hampered by a particularly challenging backdrop of changing hardware releases, as well as early ambitions to port large CPU-based CFD codes to GPU. Figure 1 indicates a minor pause in the growth rate at around 2012, which might be representative of a concerted effort to restart the programming cycle to overcome some of these problems.

[1] Scopus search: ALL("fluid dynamics" OR "CFD" OR "fluid simulation" OR "flow simulation") AND ALL("GPU" OR "GPGPU" OR "shared memory") AND <method>.

Kppa: A High Performance Source Code Generator for Chemical Kinetics

John C. Linford, Sameer Shende and Allen Malony
ParaTools, Inc.
Eugene, OR, USA
{jlinford, sameer, malony}@paratools.com

***Abstract*—Fast, accurate numerical simulations of chemical kinetics are critical to aerospace, manufacturing, materials research, earth systems research, energy research, climate and weather prediction, and air quality prediction. Although these codes address different problems, chemical kinetics simulation often accounts for 60-95% of their runtime. Kppa is a software tool that generates simulation code from a high-level description of a chemical mechanism to enable high performance simulation on low-cost computers with multi-core CPUs and computational accelerators like the Intel Xeon Phi and NVIDIA GPUs. This enables domain experts to maximize the advantage of parallel and accelerated computers without learning advanced programming concepts. To demonstrate this capability, we used Kppa to generate the chemical operator of the NASA GEOS-5 with Chemistry (GEOS-Chem) geophysical model. The model's runtime was reduced by approximately 20% without any loss of precision or stability and on the same hardware by replacing a hand-tuned time stepping integrator with an automatically-generated integrator. The Kppa-generated code is $1.7 - 2.5\times$ faster than the hand-optimized parallel code and $22 - 30\times$ faster than code generated by competing tools.**

I. Introduction

Numerical simulations of chemical kinetics are a critical research component in aerospace, manufacturing, materials, earth systems, energy, climate and weather prediction, and air quality prediction. These simulations enable a better understanding of the evolution of chemical species over time in domains as diverse as climate and weather prediction (GEOS-Chem [1] and WRF-Chem [2]), combustion simulation (S3D [3]), and air quality prediction (CMAQ [4]).

Although these application codes address different problems, the majority of their computational effort is spent solving the stiff[1], coupled equations that arise from widely varying reaction rates in their chemical mechanisms, which may involve millions of variables [5]. Chemical kinetics simulation accounts for 60-95% of the wall clock runtime of many application codes in these domains [6]. One example is GEOS-Chem, a global 3-D model of atmospheric composition driven by assimilated meteorological observations from the Goddard Earth Observing System (GEOS) of the NASA Global Modeling and Assimilation Office (GMAO). As shown in Figure 1, GEOS-Chem spends 65% of its runtime in chemical kinetic simulation. Similarly, in a WRF-Chem simulation of RADM2SORG on a 40×40 grid with 20 horizontal layers, the meteorological part of the simulation (the WRF weather model itself) is only 160×10^6 floating point operations per time step, about 2.5% the cost of the full WRF-Chem with both chemical kinetics and aerosols [7]. The remaining 97.5% is spent simulating chemical kinetics.

[1]For chemistry, *stiff* means that the system comprises widely varying reaction rates and cannot be solved using explicit numerical methods.

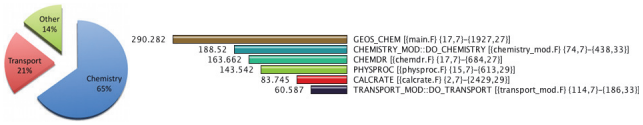


Fig. 1: Runtime performance of GEOS-Chem showing exclusive time in seconds of major program components as measured by the TAU Performance System [8].

The time-to-solution of numerical simulations of chemical kinetics can be improved by up to $30\times$ when optimized for computational accelerators like the Intel Xeon Phi, Graphical Processing Units (GPUs), or the Cell Broadband Engine [6], [7]. The trend for the foreseeable future is towards HPC systems with computational accelerators: the two fastest supercomputers in the world and 15% of the November 2014 Top500 Supercomputers use computational accelerators, up from 10.6% in 2013 and 3.4% in 2010. Yet domain experts lacking extensive computer programming expertise often find it difficult or impossible to program computational accelerators, and the state-of-the-art tools for numerical simulation of chemical kinetics (e.g. KPP [5], CHEMKIN [9], and SMVGEAR [10]) do not support accelerators. Without tool support for parallel architectures, HPC users must generate a serial program code and then rewrite it to introduce parallelism. This impractical "generate and port" approach is time consuming and error prone. The tools contain a wealth of chemical and numerical research in their input files, but re-engineering them to target the latest computer architectures is a prohibitively expensive short-term solution. Instead, the input files from these tools should be used as input for a new tool that targets the latest multi-core CPUs and accelerators directly and, by design, will target new computer architectures and languages as they emerge.

We present *Kppa* (pronounced "kappa"), a software tool that translates a high-level description of a chemical reaction network into numerical simulation code that takes full

High-Performance Computing with the Tyche Framework

Mark Sinclair-McGarvie

CTO, Marriott Sinclair
Chesterford Research Park, UK, CB10 1XL
mark@marriottsinclair.com

***Abstract*—Tyche is a highly parallel Monte Carlo simulation engine. A principal aim of Tyche is to expose a simple, high-level, interpreted model-construction language that is intuitive and easy to learn, but which at the same time allows direct vectorisation of all operations. The intent of this interpreted language is to impose little additional overhead at runtime over that which would be experienced from hand-coding all relevant low-level SIMD code, but to reduce the time taken for model construction by several orders of magnitude.**

I. Introduction

The past decade has seen the advent of new high-performance general-purpose GPU computing languages such as CUDA, OpenCL and AMP ([1], [2], [3]). These languages allow users to leverage the SIMD (single instruction multiple data) capabilities of GPUs. Additionally, SIMD may also be leveraged on CPU vector processors via instructions such as SSE and AVX, and at a higher level by use of compilers such as Intel Composer.

However, implementing codes in these languages so as to achieve optimal performance is non-trivial. Depending on the specific architecture employed, attention must be given to the time taken for data transfer between host and device, memory availability on the device itself, and proper composition of operations so as to achieve the desired degree of parallelism. These tasks can require a high degree of skill on the part of the system implementors ([4]).

The Tyche architecture is an attempt to introduce a high-level framework that, as much as possible, manages this complexity internally and presents a simple, intuitive, model construction environment to the end-user.

II. Architecture

Tyche's architecture has been designed so as to abstract away the complexity of SIMD calculations, and achieve a clear separation between these underlying "low-level" operations and the "high-level" interface presented to the end-user. The main modules in this architecture are as follows:

- **Tyche GUI**. This is the interaction environment presented to the end-user. The user constructs models in the GUI using a declarative language called "T#", which is specialised to vectorised model construction.
- **Tyche Engine**. The Tyche Engine takes the model definition specified via Tyche GUI, converting it into a Directed Acyclic Graph (DAG) of elementary operations. These operations are then executed in a determined sequence.
- **SIMD Engine**. On execution of a given node in the DAG, one or more elementary operations may be sent to the SIMD Engine for execution. The SIMD Engine in turn passes these operations to the appropriate low-level implementation.

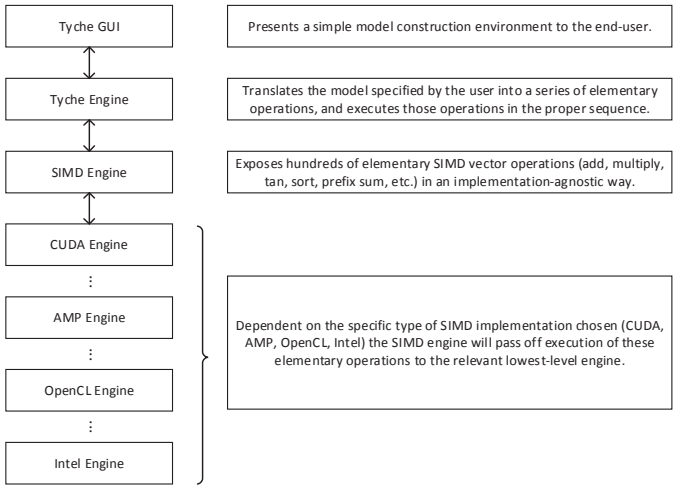


Figure 1. Overview of Tyche architecture.

III. User Interface

Tyche's user interface is designed so as to present a simple and intuitive modelling interface to the user, with no direct exposure of the complexity of the underlying SIMD operations. Model construction is based around flowcharts,

Towards Performance Portability with GungHo and GOcean

Rupert W. Ford, Andrew R. Porter, Mike Ashworth

Scientific Computing Department,
STFC Daresbury,
Warrington, U.K.
rupert.ford@stfc.ac.uk

Chris Maynard, Thomas Melvin

The Met Office,
Exeter, U.K.

***Abstract*—This paper presents a domain specific approach to performance portability and code complexity reduction in finite element and finite difference codes. This approach has been developed for the Met Office's next generation atmospheric model which uses finite elements on a quasi-unform grid, and has also been prototyped on two finite difference Ocean model benchmarks, one of which is based on the NEMO ocean model. The approach is outlined and the API's for the atmosphere and ocean models are discussed.**

I. Introduction

The Met Office's numerical weather prediction and climate model code, the Unified Model (UM), is almost 25 years old. Up to the present day the UM has been able to be run efficiently on many of the worlds most powerful computers, helping to keep the Met Office at the forefront of climate prediction and weather forecasting.

However, with performance increases from each new generation of computers now being primarily provided by an increase in the amount of parallelism rather than an increase in the clock-speed of the processors themselves, running higher resolutions of the UM now faces the double challenge of code scalability and numerical accuracy.

The UM's atmospheric dynamical core makes use of a finite-difference scheme on a regular latitude-longitude grid. The regular latitude-longitude grid results in an increasingly disparate grid resolution as the resolution increases, due to lines of longitude converging at the poles. For example, a 10km resolution at mid-latitudes would result in a 12m resolution at the poles. The difference in resolution leads to increased communication at the poles and load balance issues which are known to impair scalability; it also leads to issues with numerical accuracy and smaller time-steps due to the difference in scale.

To address this problem the Met Office, NERC and STFC initiated the GungHo project. The primary aim of this project is to deliver a scalable, numerically accurate dynamical core. This dynamical core is scheduled to become operational around the year 2022. The project is currently investigating the use of quasi-uniform meshes, such as triangular, icosahedral and cubed-sphere meshes, using finite element methods.

The associated GungHo software infrastructure is being developed to support multiple meshes and element types thus allowing for future model development. GungHo is also proposing a novel separation of concerns for Dynamo - the software implementation of the dynamical core. This approach distinguishes between three layers: the Algorithm layer, the Kernel layer and the Parallelisation System (PSy) layer. Together this separation is termed PSyKAl (pronounced as 'cycle').

Rather than writing the PSy layer manually, the GungHo project is developing a code generation system called PSyclone which can generate correct code and help a user to optimise the code for a particular architecture (by providing optimisations such as blocking, loop merging, inlining etc), or alternatively, generate the PSy layer automatically.

In the GOcean project the PSyKAl approach and the PSyclone system have been extended for use with finite difference models and applied to two Ocean benchmarks, a shallow water model and a cut down version of the NEMO ocean model.

In the following sections the PSyKAl approach is discussed in more detail, the PSyclone system is introduced and the Dynamo and GOcean API's are outlined.

II. PSyKAl

The PSyKAl approach separates code into three layers, the Algorithm layer, the PSy layer and the Kernel layer. Whilst this approach is general we have applied it to Atmosphere and Ocean models written in Fortran where domain decomposition is typically performed in the latitude-longitude direction, leaving columns of elements on each domain-decomposed partition.

Use of Parallel Monte Carlo to Validate Particle Radiotherapy Calculations

Hywel L Owen, Andrew F Green
School of Physics and Astronomy, University of Manchester, Manchester M13 9PL
Cockcroft Institute, Daresbury Science and Innovation Campus, Daresbury, Warrington WA4 4AD
(Corresponding author: hywel.owen@manchester.ac.uk)

Adam H Aitkenhead, Ranald I Mackay
The University of Manchester, Manchester Academic Health Science Centre, The Christie NHS Foundation Trust, Wilmslow Road, Manchester M20 4BX

***Abstract*—The treatment plans used to administer radiotherapy are today generally calculated using semi-analytic approximations that are fast enough to enable optimisation of the dose delivery. However, proton therapy has a specific enough dose distribution that these approximations may give rise to calculation errors. Monte Carlo particle tracking simulations that include all the transport physics can validate whether the approximations are good enough, but require the following of a large number of incident particle histories - often more than 10 million within a simulation - to obtain a suitable statistical uncertainty in the estimated dose. Such calculations may be carried out using a number of parallel computing architectures but in all cases one must preserve the validity of the models whilst optimising the code for targeting to a given platform. This is most conveniently done within a common code base such as the widely-used Geant4 framework. We have recently optimised the multi-threaded version 10 of Geant4 to give fast calculation on two notable platforms: the Intel Xeon Phi MIC architecture and the Google Cloud Compute Engine. Each platform required particular algorithmic and memory optimisation, and in this paper we present methods and results for both platforms. We demonstrate that a single Xeon Phi card may carry out a clinically-relevant treatment plan validation in around two hours, whilst the use of 1600 cores in the Google Compute Engine allows a validation calculation to be carried out at low cost in just a few minutes. The latter high-throughput approach in particular brings real-time validation within reach for clinical use at lower cost than present cluster-based solutions, and we foresee that similar approaches may in the future enable the optimisation to be directly carried out with Monte Carlo.**

I. Introduction

The accurate administration of any radiotherapy to a patient demands very good knowledge of where the dose is going to be delivered; this in turns demands both an accurate image representation of the patient composition, and a method of determining the dose given that image data that is sufficiently accurate. Proton therapy is a radiotherapy modality that offers increased dose specificity because of the inherent sharp Bragg peak of dose at the end of the proton range, but hand-in-hand with that is an increased accuracy requirement over the more widespread X-ray treatment method. If the imaging or dose calculation is inaccurate there can be a discrepancy between the estimated and real Bragg peak position of several millimetres leading to significant under- or over-dosing either of the Planning Treatment Volume (PTV) or of the surrounding tissues; any of these dose errors can have significant patient consequences.

Leaving aside the problem of obtaining adequate imaging information – which is itself a significant current challenge in proton therapy – we give attention here to the problem of accurate calculation. Most so-called treatment plans use human-assisted optimisation software to determine the best possible dose distribution that treats the PTV to the required dose whilst sparing the surrounding tissues and Organs at Risk. To achieve this optimisation in a clinical timeframe (i.e. less than a few hours) the dose calculation method – which follows each incident delivered proton spot to determine where its dose went – typically uses a semi-analytic approach that dispenses with some physical accuracy in order to obtain speed [1]. However, such calculations are prone to error particularly in highly inhomogeneous patient regions such as near metallic implants or at air/tissue boundaries such as next to nasal cavities. The accepted method of addressing the dose calculation shortcoming is to utilise Monte Carlo simulation of a sufficient number of incident proton primaries that statistically represent the intended treatment plan [2]. To obtain a 1% statistical error in the patient around 10 to 100 million primaries must be simulated, which is computationally burdensome. A very widely used Monte Carlo code is Geant4, and a serial calculation of this many primaries can take more than 24 hours on a typical desktop. This is not fast enough to allow Monte Carlo validation of the treatment plan to be used in a clinical workflow.

One response to the speed limitations of Monte Carlo methods is to simplify either the physics or the patient

Optimizing Performance of the HPC electron collisions R-matrix code PFARM on the Intel Xeon Phi

Andrew Sunderland, Martin Plummer, Greg Corbett
Scientific Computing Department
STFC Daresbury Laboratory
Warrington, UK
Andrew.sunderland@stfc.ac.uk

Michael Lysaght
Irish Centre for High End Computing (ICHEC)
Trinity Technology & Enterprise Campus
Dublin 2, Ireland

***Abstract*—The PFARM code is part of a suite of R-matrix programs that calculate electron-molecule, electron-atom and electron-ion collisions data. The code has been designed to exploit the performance of both serial and parallel highly optimized dense linear algebra routines on a wide range of high-performance computing architectures. This paper presents performance results from a new implementation of the code designed to exploit the high parallelism and performance of the Intel Xeon Phi.**

I. INTRODUCTION

PFARM is part of a suite of programs based on the 'R-matrix' ab-initio approach to the variational solution of the many-electron Schrödinger equation for electron-molecule, electron-atom, and electron-ion scattering [1] [2]. The package has been used to calculate electron collision data for astrophysical applications (such as: the interstellar medium, planetary atmospheres) with, for example, various ions of Fe and Ni and neutral O, plus other applications such as plasma modelling and fusion reactor impurities [3]. The code has been adapted for electron (positron) molecule collisions in these and other application areas [2] [4], and also to study RNA/DNA base molecule resonances relevant to biological radiation damage [5].

Recently PFARM developers have focused on optimizing the code for new and emerging architectures. The PFARM code is developed in a scalable, modular style and was designed from the start to exploit a range of highly optimized numerical library routines for the computational cores of the overall calculations [6] [7]. The resulting code therefore has the potential to exploit the computational power of new many-core architectures and computational accelerator hardware via appropriate highly optimized numerical libraries such as Intel MKL and MAGMA MIC.

II. TECHNICAL OVERVIEW & RESULTS

A. Background Testing

For efficiency, the PARM external region calculation takes place in two distinct stages, named EXDIG and EXAS, with intermediate files linking the two. EXDIG uses a parallel domain decomposition approach where large-scale parallel symmetric eigensolver computations predominate. EXAS uses a combined functional/domain decomposition approach where good load-balancing is essential to maintain efficient parallel performance [1] [7].

Initial work involved testing the performance of several computationally intensive routines (DGEMM, DGETRF, DSYEVD, DGESVD [8]) used by the EXDIG and EXAS codes on the Intel Xeon Phi. Test runs were undertaken using the numerical libraries MKL v11.2 [9] and MAGMA MIC v1.1.0 [10] for a wide range of representative data sets on the 'Knights Corner' SE10/7210 platform with 61 cores (with 4 logical threads per core) and an Intel Xeon E5 Sandy Bridge host with 2x8 cores (with 2 logical threads per core). Calls to MKL routines optimized for the Xeon Phi can either be initiated from the host Xeon processor (automatic or compiler assisted offload) or natively from code running natively on the Xeon Phi itself.

MAGMA MIC uses a hybridization approach, where the computation is split into tasks of varying granularity and their execution scheduled over both the hardware components [11]. Efficient use of the hardware is obtained by scheduling small non-parallelizable tasks on the CPU, whilst larger, more parallelizable tasks, such as Level 3 BLAS, are scheduled on the Intel Xeon Phi.

A detailed performance analysis for all the above routines involving a wide range of representative data sizes from

Optimising DL_MESO For Intel Xeon Phi

Luke Mason

Application Performance Engineering
IPCC@Hartree, STFC
Warrington, UK
luke.mason@stfc.ac.uk

Peter Mayes

High Performance Computing
Lenovo Technology UK

Michael Seaton

Computational Chemistry
IPCC@Hartree, STFC
Warrington, UK
michael.seaton@stfc.ac.uk

DL MESO contains two mesoscale simulation methods: Lattice Boltzmann Equation (LBE) and Dissipative Particle Dynamics (DPD). The Intel Parallel Computing Center based at the Hatree Centre is currently working to port DL MESO to the Xeon Phi architecture. Previously the main decomposition method employed by DL MESO has been MPI but the large number of cores and threads available on a Xeon Phi in a single shared memory enviroment have prompted refinement and expansion of the existing OpenMP threading. It was found that the original OpenMP threading provided good performance when the number of threads was low, but had a detrimental effect when higher thread counts on the Xeon Phi were used. Our fix for this poor performance is described. The Xeon Phi is also equipped with wide vector processing units and so the extent to which algorithms can be vectorised has a significant impact on code performance. The vecotrisation reporter packaged with the Intel compiliers and its use for improving performance is explained. The advantages and disadvantages of the offload computation scheme are discussed and techniques such as asynchronous transfers and data persistance that can reduce the communication overheads are outlined.

I. Introduction

DL_MESO offers a comprehensive mesoscale modelling package capable of bridging the gap between atomistic and continuum methods, the current package offers both DPD and LBE programs. DL_MESO is supplied with a simple graphical user interface (GUI) that offers a convenient way of using the package, although the GUI is not an essential tool for those who prefer command line operation. The DL_MESO package is available free of charge to academic scientists pursuing research of a non-commercial nature and for a small fee for industrial application. It can be applied to a broad range of condensed phase scientific studies: highlights include studies of transport phenomena in catalyst layers of membrane fuel cells [1] and drug loading and release in amphiphilic copolymers [2].

Until recently the performance of HPC systems has increased following Moore's law. Increases in clock speed have had a direct effect on the performance of scientific codes without need for extensive optimisation. CPU clock speeds have stalled due to limitations of power usage and densities, however. Future performance gains will be obtained from increased levels of parallelism and the exploitation of new architectures. Intel's Many Integrated Core (MIC) is such an architecture, offering high degrees of parallelism via high CPU densities, large vector units and increased threading support. The current generation of the architecture is the Xeon Phi coprocessor providing up to 61 cores, 244 threads and 1.2 TFLOPs. Future generations of Xeon Phi will be available as a coprocessor or as a standard host CPU. With integration as a standard compute node the potential for the Xeon Phi to increase the performance of scientific codes will be further increased, making the porting and optimisation of codes for the current generation even more valuable. In order to avoid some confusion that can arise from the names "Xeon Phi" and "Xeon" the Xeon Phi co-processor will be refered to as knights corner (KnC) which is the current generation of the hardware.

II. Scalability on KnC

A. Baseline Performance

DL_MESO was orignally designed with only MPI decomposition and the OpenMP was added much later around

Mont-Blanc: High Performance Computing Based on Mobile Embedded Processors

Filippo Mantovani, Luna Backes Drault

Barcelona Supercomputing Center
Barcelona - Spain
filippo.mantovani,luna.backes @bsc.es

***Abstract*—In the late 1990s, mostly economic reasons led to the adoption of commodity processors in high-performance computing. This transformation has been so effective that in 2015 the TOP500 list is still dominated by x86-based supercomputers. In 2015, the largest commodity market is the one of smartphones and tablets, most of which are built with ARM-based SoCs. This leads to the suggestion that once mobile SoCs deliver sufficient performance, mobile SoCs can help reduce the cost of HPC. In view of the experiences within the Mont-Blanc project, this contribute describes an overview of outcomes and challenges raised when developing HPC platforms from mobile embedded technology.**

I. Introduction

Over the last decades, the design of high-performance computing (HPC) systems has been conditioned by waves of technology generated in many cases by the commodity markets. Those waves had more or less impulse: the one with most impact has certainly been the one that happened in the late nineties, when the commodity x86 microprocessors entered the list of the most powerful supercomputers in the world (TOP500), rapidly growing till dominating it. Other waves have been generated by technology coming from the video-games market: both the IBM-Cell processor and the general purpose graphic processing units (GP-GPUs) are derivative technology of the commodity game market that affected -and still affects- HPC. The most active commodity market nowadays, by far, is the one of smartphones and tablets, devices carrying technology capable of performing HPC and data-center workloads. The target of the Mont-Blanc project, since 2011, is to exploit the cost effectiveness of mobile technology for scientific computation [1].

In this contribution is given an overview of the project and a more detailed presentation of the major scientific and technological enhancements obtained during the first funded period 2011-2015 ending at the end of June 2015. In the first section the hardware prototype platforms are presented, while in the second section a panoramic view of the system software ecosystem developed for ARM-based platforms is described. The third section is dedicated to HPC and data-center workloads tested on Mont-Blanc prototypes. The contribution ends with conclusions, remarks and future plans and projections.

II. Prototypes

In order to have platforms to perform experiments, develop software, port applications and take measurements of performance, power and efficiency, BSC led since 2011 the development line of various ARM-based prototypes, first under the partnership for advanced computing in Europe (PRACE) and most recently within the Mont-Blanc project. The gained experience brought to the achievement of the first objective of the Mont-Blanc project, i.e. to build a full scale prototype supercomputer based on mobile embedded technology.

A. ARM-based mini-clusters

Tibidabo - Has been the first ARM-based supercomputer, based on NVIDIA Tegra 2 dual Cortex-A9 system-on-chips (SoCs) mounted on Seco development kit and manually integrated in two racks. On this platform, a complete software stack for HPC has been ported to ARM [2].

CARMA - Based on the same carrier board composing Tibidabo, each node of this mini-cluster included a quad-core Cortex-A9 Tegra 3 SoC plus a discrete GPU Quadro 1000M, both by NVIDIA. This platform had been used for developing CUDA support for ARM, consequently included in the following CUDA releases [3].

Pedraforca - The idea of having a network of high performance GP-GPUs (NVIDIA K20) was behind the development of Pedraforca. The NVIDIA Tegra 3 quad-core Cortex-A9 allows handling of GP-GPU with minimal power consumption overhead, while the PCIe lanes plus the Infiniband technology have been used for implementing a high speed network of GP-GPUs. This platform has been used for early scalability test of HPC applications and to develop ARM support for Infiniband, however the combination of high-end GP-GPU tightly coupled with mobile SoCs showed serious limitations.

Arndale - This has been the first mini-cluster offering the possibility of using an embedded GPU in combination with a dual Cortex-A15 in the same SoC. On this platform the first OpenCL support for ARM Mali GPUs has been developed.

Odroid and Odroid XU3 - In both these mini-clusters octa-cores SoCs are used, featuring 4x Cortex-A15 and 4x Cortex-A7. These platforms have been used for exploring big.LITTLE architectures in an HPC environment, studying workload balancing, execution models and application partition [4].

PyFR: Next-Generation Computational Fluid Dynamics

P. E. Vincent, F. D. Witherden, A. M. Farrington, G. Ntemos, B. C. Vermeire, J. S. Park, A. S. Iyer, N. Loppi

Department of Aeronautics
Imperial College London
London, United Kingdom

***Abstract* — High-order numerical methods for unstructured grids combine the superior accuracy of high-order spectral or finite difference methods with the geometric flexibility of low-order finite volume or finite element schemes. The Flux Reconstruction (FR) approach unifies various high-order schemes for unstructured grids within a single framework. Additionally, the FR approach exhibits a significant degree of element locality, and is thus able to run efficiently on modern many-core hardware platforms, such as Graphical Processing Units (GPUs). The aforementioned properties of FR mean it offers a promising route to performing affordable, and hence industrially relevant, scale-resolving simulations of hitherto intractable unsteady flows within the vicinity of real-world engineering geometries. In this talk we present PyFR (www.pyfr.org), an open-source Python based framework for solving advection-diffusion type problems using the FR approach. The framework is designed to solve a range of governing systems on mixed unstructured grids containing various element types. It is also designed to target a range of hardware platforms via use of our own Mako-derived domain specific language. The current release of PyFR is able to solve the compressible Euler and Navier-Stokes equations on mixed grids of quadrilateral and triangular elements in 2D, and hexahedral, tetrahedral, prismatic, and pyramidal elements in 3D, targeting clusters of multi-core CPUs, Nvidia GPUs (K20, K40 etc.), AMD GPUs (S10000, W9100 etc.), and heterogeneous mixtures thereof. Performance and accuracy results will be presented for various benchmark and `real-world' flow problems.**

I. Introduction

There is an increasing desire amongst industrial practitioners of computational fluid dynamics (CFD) to undertake high-fidelity scale-resolving simulations of transient compressible flows within the vicinity of complex geometries. For example, to improve the design of next generation unmanned aerial vehicles (UAVs), there exists a need to perform simulations – at Reynolds numbers 10^4-10^7 and Mach numbers 0.1-1.0 – of highly separated flow over deployed spoilers/air-brakes; separated flow within serpentine intake ducts; acoustic loading in weapons bays; and flow over entire UAV configurations at off-design conditions. Unfortunately, current generation industry-standard CFD software based on first- or second-order accurate Reynolds Averaged Navier-Stokes (RANS) approaches is not well suited to performing such simulations. Henceforth, there has been significant interest in the potential of high-order accurate methods for unstructured mixed grids, and whether they can offer an efficient route to performing scale-resolving simulations within the vicinity of complex geometries. Popular examples of high-order schemes for unstructured mixed grids include the discontinuous Galerkin (DG) method, first introduced by Reed et al. [1], and the spectral difference (SD) methods originally proposed under the moniker 'staggered-gird Chebyshev multi-domain methods' by Kopriva et al. in 1996 [2] and later popularised by Sun et al. [3]. In 2007 Huynh proposed the flux reconstruction (FR) approach [4]; a unifying framework for high-order schemes for unstructured grids that incorporates a nodal DG scheme and, at least for a linear flux function, any SD scheme. In addition to offering high-order accuracy on unstructured mixed grids, FR schemes are also compact in space, and thus when combined with explicit time marching offer a significant degree of element locality. As such, explicit high-order FR schemes are characterised by a large degree of structured computation, even on unstructured grids. Over the past two decades improvements in the arithmetic capabilities of processors have significantly outpaced advances in random access memory. Algorithms which have traditionally been compute bound – such as dense matrix-vector products – are now limited instead by the bandwidth to/from memory. This is epitomised in Fig. 1. Whereas the processors of two decades ago had FLOPS-per-byte of ~0.2 more recent chips have ratios upwards of ~4. This disparity is not limited to just conventional CPUs. Massively parallel accelerators and co-processors such as the Nvidia K20X and Intel Xeon Phi 5110P have ratios of 5.24 and 3.16, respectively.

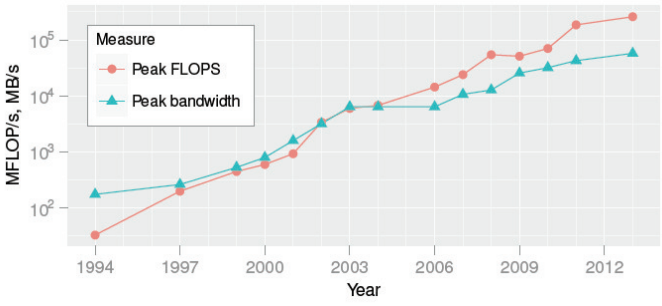


Figure 1. Trends in the peak floating point performance (double precision) and memory bandwidth of sever-class Intel processors from 1994-2013. The quotient of these two measures yields the FLOPS-per-byte of a processor. Data courtesy of Jan Treibig. Reproduced from [5] with permission.

Complex Systems Simulation with CUDA

Paul Richmond

Department of Computer Science
University of Sheffield
Sheffield, UK
p.richmond@sheffield.ac.uk

***Abstract*—Simulation of complex systems provides a computational challenge due to the amount of computational performance required to simulate all individuals within a large population. The Graphics Processing Unit (GPU) presents a potential solution. Using the CUDA programming language GPUs can be programmed for general purpose use. Unfortunately the translation of a complex systems model to CUDA code is a non-trivial task requiring specialist knowledge of the architecture to obtain good performance. This paper reviews FLAME GPU a framework which provides transparent mapping and simulation of complex systems models to a CUDA enabled GPU. The framework has considerable performance benefits over its CPU counterpart and provides real-time visualisation for interactive steering of simulations in domains as diverse as computational biology and pedestrian dynamics.**

I. Introduction *and Background*

A complex system is a system consisting of a large number of entities which interact to produce behaviour which demonstrates intelligence or self-organisation beyond that of the individuals within the system (emergent behaviour). Within biology complex systems are prevalent at many biological scales; from molecular systems to, cellular systems through to ecological systems of entire populations. Beyond the biological domain, complex systems represent phenomenon such as economic markets where the term 'complexity economics' has been derived to describe the application of complexity science to the field.

In order to understand a complex system, simulation can be applied to gain insight into the processes which result in system level behaviour. Simulation allows the behaviours of the entities comprising a complex system to be described as a set of rules, or *agents*, and their interaction to be simulated in order to observe and predict. This form of *agent based* simulation is advantageous as it can be used to explore the impact of changing an agent's behaviour or its environment. For example the impact of a drug on a particular chemical pathway or a building design with respect to evacuation safety.

The challenge of simulating large systems as interacting entities presents a computational challenge. For example, when compared to alternative methods of simulating population dynamics (such as top down equation based modelling), larger demands are placed both on computational performance and memory for storage of the systems state. Parallel computing presents a possible solution to providing increased levels of computing performance, in particular the Graphics Processing Unit (GPU) is commodity hardware boasting very high theoretical performance within a small power window. Applications which have been ported to the GPU often report speedups of orders of magnitude beyond their serial counterparts [1]. The transfer of a complex systems model to the GPU is however non-trivial. Whilst it can be observed that agents within a complex system 'may' be simulated independently, the simulation does not constitute an embarrassingly parallel problem (a problem where the task can be separated into tasks with no interdependencies). Within a complex systems simulation, communication (and hence synchronisation) between individuals must be carefully managed, agent creation and deletion (life and death) must be handled defiantly and code divergence must be minimised (see section III). In order to solve the challenge of mapping a complex systems model to the GPU therefore requires knowledge of the underlying architecture. This is particularly important on GPUs where much of the intelligence offered in traditional computing architectures (such as large data caches, out of order execution and long pipelines) is sacrificed for additional throughput.

II. FLAME GPU FRAMEWORK

The Flexible Large Scale Agent Modelling Environment for the GPU (FLAME GPU) is a framework which aims to vastly improve the performance of complex systems simulations whilst simultaneously providing a level of abstraction which hides the underlying complexities of the architecture. Ultimately the objective of the software is to provide a high level interface for describing the behaviour and interactions of agents (the model) and then automatically translating this to high performance parallel GPU code [2].

The basis of a FLAME GPU model is an XML description of the individual agents and the information (in the form of messages) shared between them. Agents are described using a high level representation based on a communicating stream X-Machine, a form of state machine. During the transition between two states a user defined *agent function* is defined to capture the translation of internal memory into a new state. During the agent function a message (the only form of communication between agents) may be input or output (but never both) via a global list. This indirect message communication allows functional dependencies between agents to be observed, allowing increased parallelism within the model and minimising synchronisation points between agent functions.

Enabling High-Performance Database Applications on Supercomputing Architectures: Accelerating the Oasis Loss Modelling Framework

M.A. Johnston, K.E. Jordan

Data-Centric Systems,
IBM Research

M. Modani, D. Moss

Hartree Centre,
Daresbury, UK

***Abstract*—The use of supercomputers in many industry segments is rare, and a key question is how High Performance Computing (HPC) could be harnessed by commercial companies. The potential benefits are a large decrease in time-to-solution, which can be a critical parameter in many industry contexts, and a large increase in the amount of data that can be processed, which can provide greater insight to decision making. One challenge in this area is modifying existing industry applications to exploit supercomputers as they may be written in programming models not familiar to the HPC community and thus there is a lack of knowledge on how to deploy them. In this article we present the results of deploying such an application, the SQL based Oasis Loss Modelling Framework, on a supercomputer, and discuss the key issues we encountered in this process. Our prototype clustered database solution achieves a 41x and 49x speed-up respectively for the two core Oasis calculations and thus is a concrete example of how HPC can accelerate such applications.**

I. INTRODUCTION

Supercomputers have played a central role in computational science since its inception and a wide array of scientific applications have been developed by the academic community that harness these machines. In comparison, the use of supercomputers in many industry segments is rare, and a key question is how the benefits of High Performance Computing (HPC) could be harnessed by commercial companies. One area of investigation is how existing industry applications could be modified to exploit supercomputers, which are characterised by 1000s of compute nodes connected by a high-performance network and accessing a parallel file-systm. The potential benefits are a large decrease in time-to-solution, which can be a critical parameter in many industry contexts, and a large increase in the amount of data that could be processed, which can provide greater insight to decision making. However these applications may be written in programming models e.g. SQL not familiar to the scientific community making porting them to traditional HPC architectures challenging.

Catastrophe modelling is the process of estimating the losses caused by catastrophic events like floods and earthquakes due to the insurance policies held on properties damaged by the event [1]. It plays a key role in the insurance and reinsurance industries in analysing risk, allowing them to provide more accurate pricing and to asses the adequacy of their capital. The Oasis Loss Modelling Framework (LMF) is an open-source software application for catastrophe modelling. It is a SQL database application - all data is stored in database tables and the loss calculation is carried out as SQL operations over these tables.

The aim of this work was to examine if supercomputing architectures could be used to accelerate the Oasis LMF. Distributed or clustered databases have been a central focus of research into how SQL applications can harness multiple nodes [2]. Here we present a prototype clustered database solution for Oasis which achieves up-to a 49x speed-up over a base MySQL server implementation using supercomputing resources. We describe some of the main issues encountered in deploying this prototype on a supercomputer, including inter-process synchronization and file-system interactions, and show how they were overcome. We identify issues in scaling this solution to larger node counts and discuss some future directions for Oasis and SQL applications in general on HPC architectures.

A. *Oasis Kernel Calculation*

Due to space limitations the Oasis Kernel calculation can only be briefly sketched here. Readers are referred to [3] for more detail.

The aim of loss-modelling is to estimate the loss to an insurer due to the property insurance policies it has sold (exposures) if a specific catastrophic event occurs in a certain area. This essentially reduces to determining the likelihood of a property being damaged to a certain degree by the event and then assigning a cost to that damage. The basic calculation unit in Oasis is the combination of an event, a geographic area and a property type – termed a

INDEX

http://emit.manchester.ac.uk

University of Manchester, Manchester, U.K.
ISBN 978-0-9933426-0-8