J Supercomput (2016) 72:4043–4046
DOI 10.1007/s11227-016-1822-8

Introduction to sustainable ultrascale computing systems and applications

Jesus Carretero[1] · **Javier Garcia-Blas**[1] · **Raimondas Ciegis**[2]

Published online: 25 July 2016

1 Introduction

Ultrascale systems are envisioned as large-scale complex systems joining parallel and distributed computing systems that will be two to three orders of magnitude larger than today's systems. Targeting sustainable solutions for ultrascale computing will need cross fertilization among HPC, large-scale distributed systems, and big data management, and the reformulation of many algorithms and applications used today in different areas of research. Such a reformulation has to address different challenges that arise from the different application areas, algorithms, and programs. Challenges include the scalability of the applications programs to use a large number of system resources efficiently, the usage of resilience methods to include mechanisms to enable application programs to react to system failures, and the inclusion of energy-awareness features into the application programs to be able to obtain an energy-efficient execution. As a large number of resources has to be controlled when using ultrascale systems, the availability of suitable programming models and environments also plays an important role for ultrascale applications. The programming models for ultrascale computing should provide enough abstractions, such that the application programmer does not need to deal with all low-level details of an efficient execution of the (parallel)

✉ Jesus Carretero
jcarrete@inf.uc3m.es; jesus.carretero@uc3m.es

Javier Garcia-Blas
fjblas@inf.uc3m.es

Raimondas Ciegis
raimondas.ciegis@leu.lt

1 Universidad Carlos III de Madrid, Madrid, Spain

2 Vilnius Gediminas Technical University, Vilnius, Lithuania

programs and the control of the execution resources of the platform. On the other hand, the programming models should enable the application programmer to concentrate on the algorithmic aspects and problem-specific issues of the specific application area, such that program development is supported as far as possible.

This special issue features nine papers showing new paradigms and applications that could be suitable for ultrascale systems. Several of them are extensions of the best papers selected from the Second International Workshop on Ultrascale Computing Systems held in Krakow in September 2016.

In *Water-Level scheduling for parallel tasks in compute-intensive application components*, the author presents the development of complex simulations with very high computational demands optimized using heterogeneous computing nodes. Exploiting heterogeneous computing resources for the execution of parallel tasks can be achieved by integrating dedicated scheduling methods into the complex simulation code. The paper shows several scheduling methods for distributing parallel simulation tasks among heterogeneous computing nodes. Performance results and comparisons are shown for two novel scheduling methods and several existing scheduling algorithms for parallel tasks.

Data locality is a major goal in ultrascale systems. The paper *Exploiting in-memory storage for improving workflow executions in Cloud platform* proposes the usage of the Hercules system within Data Mining Cloud Framework (DMCF) as an ad-hoc storage system for temporary data produced inside workflow-based applications. Hercules is a distributed in-memory storage system highly scalable and easy to deploy. The proposed solution takes advantage of the scalability capabilities of Hercules to avoid the bandwidth limits of the default storage. The integration of Hercules and DMCF on a real application have been evaluated comparing the performance with Azure. The results show that the I/O overhead in this real-life scenario using Hercules has been reduced by 36 % with respect to Azure storage, leading to a 13 % reduction of the total execution time.

Increasing locality is needed to reduce data movement and execution time, but also to reduce energy consumed by data movement in the storage system. In *Analyzing the Energy Consumption of the Storage Data Path*, Llopis et al. analyze the power costs of performing I/O operations and intra-node data movement, focusing on the operating system's I/O stack. Their approach combines the hardware instrumentation of a fine-grained watt-meter, software instrumentation for gathering system metrics time series, and data analysis techniques to gain insights into how different I/O patterns make use of system resources, including the electrical power. This data-driven process is synthesized into a methodology, and the results of applying this methodology on various I/O intensive workload patterns are presented in the paper, together with the identification of key system metrics that contribute to I/O-related power usage.

Network contention is another major topic in ultrascale systems. In *High-performance network traffic analysis for continuous batch intrusion detection*, the authors characterize the computational requirements of the network traffic packets for several conditions, which constitute a useful tool for generating network workload in simulated scenarios, focusing on map-intensive jobs, such as string matching-based virus and malware detection. The model is then applied to a novel architecture for a Hadoop-based network analysis solution, including a scheduler, report on using this

approach in a cluster, and scheduling performance results obtained. The scheduler considers a cloud-based Elastic MapReduce-style traffic analysis solution to overcome the local resources limitations. The results show that we are able to reduce up to 50 % the amount of the traffic to burst out and still accomplish the real-time traffic analysis.

Following the path, data movement is a major contributor to network traffic in the new data-intensive applications era. Thus, current storage trends dictate placing fast storage devices in all servers and using them as a single distributed storage system. The paper *Mitigation of NUMA and synchronization effects in high-speed network storage over raw Ethernet* proposes to use Tyche, an in-house protocol for network storage based on raw Ethernet, to examine and address the performance implications of NUMA servers on end-to-end path, and the synchronization issues arising with multiple NICs and multicore servers. NUMA and synchronization issues are evaluated on a real setup with multicore servers and six 10 GBits/s NICs on each server to find that NUMA effects have significant negative impact and can reduce throughput by almost 2x on the servers with as few as 8 cores (16 hyper-threads). Protocol extensions are presented to almost entirely eliminate NUMA effects by encapsulating all protocol structures to a 'channel' concept and then carefully mapping channels and their resources to NICs and NUMA nodes.

Not only system software and runtimes need enhancements for ultrascale systems. As said before, applications must also be rethought with the new requirements in mind. This rethinking affects all levels, from algorithms and programming facilities to use case applications. In this special issue, we show four of those applications.

The paper *Nekbone Performance on GPUs with OpenACC and CUDA Fortran Implementations* presents a GPU implementation and performance analysis of Nekbone, which represents one of the core kernels of the incompressible Navier–Stokes solver Nek5000, based on OpenACC and CUDA Fortran for local parallelization of the compute-intensive matrix-matrix multiplication part, which significantly minimizes the modification of the existing CPU code while extending the simulation capability of the code to GPU architectures. OpenACC interoperating with CUDA Fortran and the gather-scatter operations with GPUDirect communication provide performance of up to 552 Tflops on 16, 384 GPUs of the OLCF Cray XK7 Titan.

Another general use case is solving large-scale sparse linear systems, which plays a major role in most fluid mechanics, simulation and design of materials, petroleum seismic data processing, numerical weather prediction, computational electromagnetic, and numerical simulation of unclear explosions. The authors of *Performance modeling of hyper-scale custom machine for the principal steps in block Wiedemann algorithm* propose a hyper-scale custom supercomputer architecture that matches specific data features to process the key procedure of block Wiedemann algorithm and its parallel algorithm on the custom machine. The paper builds optimizations and a performance model to evaluate the execution performance and power consumption for a custom machine, resulting in a considerable speedup, even three times faster than the fastest supercomputer, TH2, while consuming less power.

Global optimization problems, where a set of feasible solutions is discrete and very large, are a major challenge as a priori estimation techniques that cannot be applied to exclude an essential part of elements from the feasible set, and full searches are

required to solve these very computational demanding problems. In *Application of distributed parallel computing for dynamic visual cryptography*, the authors present a library of C++ templates that allow its user to implement parallel master–slave algorithms for its application without any knowledge of parallel programming API. Design of the templates supports several programming models, as MPI, BOINC, adn C/C+∗, and examples of parallel and distributed computing applications are shown.

Providing access to high-performance resources to users of domain-specific languages is also very important for ultrascale systems. In *Cellular automata labeling of connected components in n-dimensional binary lattices*, a new cellular automata-based algorithm for labeling of connected components in n-dimensional binary lattices, for $n \geq 2$, is proposed. The algorithm for 3D binary images was implemented in NetLogo and MatLab programming environments. The algorithm is local and can be efficiently implemented on dataflow parallel platforms with an average asymptotic complexity of $O(L)$ on Ln binary lattices. However, some worst-case arrangements of the n-dimensional lattice cells could require O(Ln) calculation steps.

To finish, we would like to thank all the authors for contributing to this special issue. We also would like to show our recognition to the *Journal of Supercomputers* editor, Prof. Hamid Arabia, for allowing us to publish this special issue in the journal.