EURO-PAR CONFERENCE 2019

ABSTRACTS

Support Tools and Environments

ONLINE FAULT CLASSIFICATION IN HPC SYSTEMS THROUGH MACHINE LEARNING

Alessio Netti, Zeynep Kiziltan, Ozalp Babaoglu, Alina Sîrbu, Andrea Bartolini, Andrea Borghesi

As High-Performance Computing (HPC) systems strive towards the exascale goal, studies suggest that they will experience excessive failure rates. For this reason, detecting and classifying faults in HPC systems as they occur and initiating corrective actions before they can transform into failures will be essential for continued operation. In this paper, we propose a fault classification method for HPC systems based on machine learning that has been designed specifically to operate with live streamed data. We cast the problem and its solution within realistic operating constraints of online use. Our results show that almost perfect classification accuracy can be reached for different fault types with low computational overhead and minimal delay. We have based our study on a local dataset, which we make publicly available, that was acquired by injecting faults to an in-house experimental HPC system.

ACCELERATING DATA-DEPENDENCE PROFILING WITH STATIC HINTS

Mohammad Norouzi, Qamar Ilias, Ali Jannesari, Felix Wolf

Data-dependence profiling is a program-analysis technique to discover potential parallelism in sequential programs. Contrary to purely static dependence analysis, profiling has the advantage that it captures only those dependences that actually occur during execution. Lacking critical runtime information such as the value of pointers and array indices, purely static analysis may overestimate the amount of dependences. On the downside, dependence profiling significantly slows down the program, not seldom prolonging execution by a factor of 100. In this paper, we propose a hybrid approach that substantially reduces this overhead. First, we statically identify persistent data dependences that will appear in any execution. We then exclude the affected source-code locations from instrumentation, allowing the profiler to skip them at runtime and avoiding the associated overhead. At the end, we merge static and dynamic dependences. We evaluated our approach with 38 benchmarks from two benchmark suites and obtained a median reduction of the profiling time by 62% across all the benchmarks.

MULTI-VALUED EXPRESSION ANALYSIS FOR COLLECTIVE CHECKING

Pierre Huchant, Emmanuelle Saillard, Denis Barthou, Patrick Carribault

Determining if a parallel program behaves as expected on any execution is challenging due to non-deterministic executions. Static analyses help to detect all execution paths that can be executed concurrently by identifying multi-valued expressions, i.e. expressions evaluated differently among processes. This can be used to find collective errors in parallel programs. In this paper, we propose a new method that combines a control-flow analysis with a multi-valued expressions detection to find such errors. We implemented our method in the PARCOACH framework and successfully analyzed parallel applications using MPI, OpenMP, UPC and CUDA.

Performance and Power Modeling, Prediction and Evaluation

TOWARDS PORTABLE ONLINE PREDICTION OF NETWORK UTILIZATION USING MPI-LEVEL MONITORING

Shu-Mei Tseng, Bogdan Nicolae, George Bosilca, Emmanuel Jeannot. Aparna Chandramow. Iishwaran, Franck Cappello

Stealing network bandwidth helps a variety of HPC runtimes and services to run additional operations in the background without negatively affecting the applications. A key ingredient to make this possible is an accurate prediction of the future network utilization, enabling the runtime to plan the background operations in advance, such as to avoid competing with the application for network bandwidth. In this paper, we propose a portable deep learning predictor that only uses the information available through MPI introspection to construct a recurrent sequence-to-sequence neural network capable of forecasting network utilization. We leverage the fact that most HPC applications exhibit periodic behaviors to enable predictions far into the future (at least the length of a period). Our online approach does not have an initial training phase, it continuously improves itself during application execution without incurring significant computational overhead. Experimental results show better accuracy and lower computational overhead compared with the state-of-the-art on two representative applications.

A COMPARISON OF RANDOM TASK GRAPH GENERATION METHODS FOR SCHEDULING PROBLEMS

Louis-Claude Canon, Mohamad El Sayah, Pierre-Cyrille Héam

How to generate instances with relevant properties and without bias remains an open problem of critical importance to compare heuristics fairly. When scheduling with precedence constraints, the instance is a task graph that determines a partial order on task executions. To avoid selecting instances among a set populated mainly with trivial ones, we rely on properties such as the mass, which measures how much a task graph can be decomposed into smaller ones. This property and an in-depth analysis of existing random instance generators establish the sub-exponential generic time complexity of the studied problem.

HARDWARE COUNTERS' SPACE REDUCTION FOR CODE REGION CHARACTERIZATION

Jordi Alcaraz, Anna Sikora, Eduardo César

This work proposes that parallel code regions in an OpenMP application can be characterized using a signature composed by the values of a set of hardware performance counters. Our proposal is aimed towards dynamic tuning and, consequently, the metrics must be collected at execution time, which limits the number of metrics that can be measured. Therefore, our main contribution is the definition of a methodology to determine a reduced set of hardware performance counters that can be measured at application's execution time and that still contains enough information to characterize a parallel region. The proposed methodology is based on principal component analysis and linear correlation analysis. Preliminary results show that it can be used to successfully reduce the number of hardware counters needed to characterize a parallel region, and that this set of counters can be measured at run time with high accuracy and low overhead using counter multiplexing.

COMBINING CHECKPOINTING AND DATA COMPRESSION TO ACCELERATE ADJOINT-BASED OPTIMIZATION PROBLEMS

Navjot Kukreja, Jan Hückelheim, Mathias Louboutin, Paul Hovland, Gerard Gorman

Seismic inversion and imaging are adjoint-based optimization problems that process up to terabytes of data, regularly exceeding the memory capacity of available computers. Data compression is an effective strategy to reduce this memory requirement by a certain factor, particularly if some loss in accuracy is acceptable. A popular alternative is checkpointing, where data is stored at selected points in time, and values at other times are recomputed as needed from the last stored state. This allows arbitrarily large adjoint computations with limited memory, at the cost of additional recomputations.
In this paper, we combine compression and checkpointing for the first time to compute a realistic seismic inversion. The combination of checkpointing and compression allows larger adjoint computations compared to using only compression, and reduces the recomputation overhead significantly compared to using only checkpointing.

Scheduling and Load Balancing

LINEAR TIME ALGORITHMS FOR MULTIPLE CLUSTER SCHEDULING AND MULTIPLE STRIP PACKING

Klaus Jansen, Malin Rau

We study the Multiple Cluster Scheduling problem and the Multiple Strip Packing problem. For both problems, there is no algorithm with approximation ratio better than 2 unless P=NP. In this paper, we present an algorithm with approximation ratio 2 and running time $O(n)$ for both problems for $N>2$ (and running time $O(n\log(n)^2)$ for $N=2$). While a 2 approximation was known before, the running time of the algorithm is at least $\Omega(n^{256})$ in the worst case. Therefore, an $O(n)$ algorithm is surprising and the best possible. While the above result is strong from a theoretical point of view, it might not be very practical due to a large hidden constant caused by calling an AEPTAS with a constant $\varepsilon \geq 1/8$ as subroutine. Nevertheless, we point out that the general approach of finding first a schedule on one cluster and then distributing it onto the other clusters might come in handy in practical approaches. We demonstrate this by presenting a practical algorithm with running time $O(n\log(n))$, without hidden constants, that is an approximation algorithm with ratio 9/4 if the number N of clusters is dividable by 3 and bounded by 9/4+3/4N otherwise.

SCHEDULING ON TWO UNBOUNDED RESOURCES WITH COMMUNICATION COSTS

Massinissa Ait Aba, Alix Munier Kordon, Guillaume Pallez (Aupy)

Heterogeneous computing systems are popular and powerful platforms, containing several heterogeneous computing elements (e.g. CPU+GPU). In this work, we consider a platform with two types of machines, each containing an unbounded number of elements. We want to execute an application represented as a Directed Acyclic Graph (DAG) on this platform. Each task of the application has two possible execution times, depending on the type of machine it is executed on. In addition we consider a cost to transfer data from one platform to the other between successive tasks. We aim at minimizing the execution time of the DAG (also called makespan). We show that the problem is NP-complete for graphs of depth at least three but polynomial for graphs of depth at most two. In addition, we provide polynomial-time algorithms for some usual classes of graphs (trees, series-parallel graphs).

IMPROVING FAIRNESS IN A LARGE SCALE HTC SYSTEM THROUGH WORKLOAD ANALYSIS AND SIMULATION

Frédéric Azevedo, Dalibor Klusáček, Frédéric Suter

Monitoring and analyzing the execution of a workload is at the core of the operation of data centers. It allows operators to verify that the operational objectives are satisfied or detect and react to any unexpected and unwanted behavior. However, the scale and complexity of large workloads composed of millions of jobs executed each month on several thousands of cores, often limit the depth of such an analysis. This may lead to overlook some phenomena that, while not harmful at a global scale, can be detrimental to a specific class of users.

In this paper, we illustrate such a situation by analyzing a large High Throughput Computing (HTC) workload trace coming from one of the largest academic computing centers in France. The Fair-Share algorithm at the core of the batch scheduler ensures that all user groups are fairly provided with an amount of computing resources commensurate to their expressed needs. However, a deeper analysis of the produced schedule, especially of the job waiting times, shows a certain degree of unfairness between user groups. We identify the configuration of the quotas and scheduling queues as the main root causes of this unfairness. We thus propose a drastic reconfiguration of the system that aims at being more suited to the characteristics of the workload and at better balancing the waiting time among user groups. We evaluate the impact of this reconfiguration through detailed simulations. The obtained results show that it still satisfies the main operational objectives while significantly improving the quality of service experienced by formerly unfavored users.

TOGGLE: CONTENTION-AWARE TASK SCHEDULER FOR CONCURRENT HIERARCHICAL OPERATIONS

Saurabh Kalikar, Rupesh Nasre

Rooted hierarchies are efficiently operated on using hierarchical tasks. Effective synchronization for hierarchies therefore demands hierarchical locks. State-of-the-art approaches for hierarchical locking are unaware of how tasks are scheduled. We propose a lock-contention aware task scheduler which considers the locking request while assigning tasks to threads. We present the design and implementation of Toggle, which exploits nested intervals and work-stealing to maximize throughput. Using widely used STMBench7 benchmark, a real-world XML hierarchy, and a state-of-the-art hierarchical locking protocol, we illustrate that Toggle considerably improves the overall application throughput.

LOAD-BALANCING FOR PARALLEL DELAUNAY TRIANGULATIONS

Daniel Funke, Peter Sanders, Vincent Winkler

Computing the Delaunay triangulation (DT) of a given point set in $\mathbb{R}^D$ is one of the fundamental operations in computational geometry. Recently, Funke and Sanders [Funke, D., Sanders, P.: Parallel d-d Delaunay triangulations in shared and distributed memory. In: ALENEX, pp. 207–217, SIAM (2017)] presented a divide-and-conquer DT algorithm that merges two partial triangulations by re-triangulating a small subset of their vertices – the border vertices – and combining the three triangulations efficiently via parallel hash table lookups. The input point division should therefore yield roughly equal-sized partitions for good load-balancing and also result in a small number of border vertices for fast merging. In this paper, we present a novel divide-step based on partitioning the triangulation of a small sample of the input points. In experiments on synthetic and real-world data sets, we achieve nearly perfectly balanced partitions and small border triangulations. This almost cuts running time in half compared to non-data-sensitive division schemes on inputs exhibiting an exploitable underlying structure.

DESIGN-SPACE EXPLORATION WITH MULTI-OBJECTIVE RESOURCE-AWARE MODULO SCHEDULING

Julian Oppermann, Patrick Sittel, Martin Kumm, Melanie Reuter-Oppermann, Andreas Koch, Oliver Sinnen

Many of today's applications in parallel and concurrent computing are deployed using reconfigurable hardware, in particular field-programmable gate arrays (FPGAs). Due to the complexity of modern applications and the wide spectsrum of possible implementations, manual design of modern custom hardware is not feasible. Computer-aided design tools enable the automated transformation of high-level descriptions into hardware. However, the efficient identification of Pareto-optimal solutions to trade-off between resource utilisation and throughput is still an open research topic. Combining resource allocation and modulo scheduling, we propose a new approach for design-space exploration of custom hardware implementations. Using problem-specific rules, we are able to exclude obviously dominated solutions from the design space before scheduling and synthesis. Compared to a standard, multi-criteria optimisation method, we show the benefits of our approach regarding runtime at the design level.

IMPLEMENTING YEWPAR: A FRAMEWORK FOR PARALLEL TREE SEARCH

Blair Archibald, Patrick Maier, Robert Stewart, Phil Trinder

Combinatorial search is central to many applications yet hard to parallelise. We argue for improving the reuse of parallel searches, and present the design and implementation of a new parallel search framework. YewPar generalises search by abstracting search tree generation, and by providing algorithmic skeletons that support three search types, together with a set of search coordination strategies. The evaluation shows that the cost of YewPar generality is low (6.1%); global knowledge is inexpensively shared between workers; irregular tasks are effectively distributed; and YewPar delivers good runtimes, speedups and efficiency with up to 255 workers on 17 localities.

PLB-HAC: DYNAMIC LOAD-BALANCING FOR HETEROGENEOUS ACCELERATOR CLUSTERS

Luis Sant'Ana, Daniel Cordeiro, Raphael Y. de Camargo

Efficient usage of Heterogeneous clusters containing combinations of CPUs and accelerators, such as GPUs and Xeon Phi boards requires balancing the computational load among them. Their relative processing speed for each target application is not available in advance and must be computed at runtime. Also, dynamic changes in the environment may cause these processing speeds to change during execution. We propose a Profile-based Load-Balancing algorithm for Heterogeneous Accelerator Clusters (PLB-HAC), which constructs a performance curve model for each resource at runtime and continuously adapt it to changing conditions. It dispatches execution blocks asynchronously, preventing synchronization overheads and other idleness periods due to imbalances. We evaluated the algorithm using data clustering, matrix multiplication, and bioinformatics applications and compared with existing load-balancing algorithms. PLB-HAC obtained the highest performance gains with more heterogeneous clusters and larger problems sizes, where a more refined load-distribution is required.

Data Management, Analytics and Deep Learning

* BEST PAPER 2019 ENHANCING THE PROGRAMMABILITY AND PERFORMANCE PORTABILITY OF GPU TENSOR OPERATIONS

Arya Mazaheri, Johannes Schulte, Matthew W. Moskewicz, Felix WolfAli Jannesari

Deep-learning models with convolutional networks are widely used for many artificial-intelligence tasks, thanks to the increasing adoption of high-throughput GPUs, even in mobile phones. CUDA and OpenCL are the two largely used programming interfaces for accessing the computing power of GPUs. However, attaining code portability has always been a challenge, until the introduction of the Vulkan API. Still, performance portability is not necessarily provided. In this paper, we investigate the unique characteristics of CUDA, OpenCL, and Vulkan kernels and propose a method for abstracting away syntactic differences. Such abstraction creates a single-source kernel which we use for generating code for each GPU programming interface. In addition, we expose auto-tuning parameters to further enhance performance portability. We implemented a selection of convolution operations, covering the core operations needed for deploying three common image-processing neural networks, and tuned them for NVIDIA, AMD, and ARM Mali GPUs. Our experiments show that we can generate deep-learning kernels with minimal effort for new platforms and achieve reasonable performance. Specifically, our Vulkan backend is able to provide competitive performance compared to vendor deep-learning libraries.

UNIFIED AND SCALABLE INCREMENTAL RECOMMENDERS WITH CONSUMED ITEM PACKS

Rachid Guerraoui, Erwan Le Merrer, Rhicheek Patra, Jean-Ronan Vigouroux

Recommenders personalize the web content using collaborative filtering to relate users (or items). This work proposes to unify user-based, item-based and neural word embeddings types of recommenders under a single abstraction for their input, we name Consumed Item Packs (CIPs). In addition to genericity, we show this abstraction to be compatible with incremental processing, which is at the core of low latency recommendation to users. We propose three such algorithms using CIPs, analyze them, and describe their implementation and scalability for the Spark platform. We demonstrate that all three provide a recommendation quality that is competitive with three algorithms from the state-of-the-art.

DECLARATIVE BIG DATA ANALYSIS FOR HIGH-ENERGY PHYSICS: TOTEM USE CASE

Valentina Avati, Milosz Blaszkiewicz, Enrico Bocchi, Luca Canali, Diogo Castro, Javier Cervantes, Leszek Grzanka, Enrico Guiraud, Jan Kaspar, Prasanth Kothuri, Massimo Lamanna, Maciej Malawski, Aleksandra Mnich, Jakub Moscicki, Shravan Murali, Danilo Piparo, Enric Tejedor

The High-Energy Physics community faces new data processing challenges caused by the expected growth of data resulting from the upgrade of LHC accelerator. These challenges drive the demand for exploring new approaches for data analysis. In this paper, we present a new declarative programming model extending the popular ROOT data analysis framework, and its distributed processing capability based on Apache Spark. The developed framework enables high-level operations on the data, known from other big data toolkits, while preserving compatibility with existing HEP data files and software. In our experiments with a real analysis of TOTEM experiment data, we evaluate the scalability of this approach and its prospects for interactive processing of such large data sets. Moreover, we show that the analysis code developed with the new model is portable between a production cluster at CERN and an external cluster hosted in the Helix Nebula Science Cloud thanks to the bundle of services of Science Box.

CLUSTERING AS APPROXIMATION METHOD TO OPTIMIZE HYDROLOGICAL SIMULATIONS

Elnaz Azmi, Uwe Ehret, Jörg Meyer, Rik van Pruijssen, Achim Streit, Marcus Strobl

Accurate water-related predictions and decision-making require a simulation of hydrological systems in high spatio-temporal resolution. However, the simulation of such a large-scale dynamical system is compute-intensive. One approach to circumvent this issue, is to use landscape properties to reduce model redundancies and computation complexities. In this paper, we extend this approach by applying machine learning methods to cluster functionally similar model units and by running the model only on a small yet representative subset of each cluster. Our proposed approach consists of several steps, in particular the reduction of dimensionality of the hydrological time series, application of clustering methods, choice of a cluster representative, and study of the balance between the uncertainty of the simulation output of the representative model unit and the computational effort. For this purpose, three different clustering methods namely, K-Means, K-Medoids and DBSCAN are applied to the data set. For our test application, the K-means clustering achieved the best trade-off between decreasing computation time and increasing simulation uncertainty.

Cluster and Cloud Computing

YOLO: SPEEDING UP VM AND DOCKER BOOT TIME BY REDUCING I/O OPERATIONS

Thuy Linh Nguyen, Ramon Nou, Adrien Lebre

Although this comes as a surprise, the time to boot a Docker-based container can last as long as a virtual machine in high consolidated cloud scenarios. Because this time is critical as boot duration defines how an application can react w.r.t. demands' fluctuations (horizontal elasticity), we present in this paper the YOLO mechanism (You Only Load Once). YOLO reduces the number of I/O operations generated during a boot process by relying on a boot image abstraction, a subset of the VM/container image that contains data blocks necessary to complete the boot operation. Whenever a VM or a container is booted, YOLO intercepts all read accesses and serves them directly from the boot image, which has been locally stored on fast access storage devices (e.g., memory, SSD, etc.). In addition to YOLO, we show that another mechanism is required to ensure that files related to VM/container management systems remain in the cache of the host OS. Our results show that the use of these two techniques can speed up the boot duration 2–13 times for VMs and 2 times for containers. The benefit on containers is limited due to internal choices of the docker design. We underline that our proposal can be easily applied to other types of virtualization (e.g., Xen) and containerization because it does not require intrusive modifications on the virtualization/container management system nor the base image structure.

Parallel and Distributed Programming, Interfaces, and Languages

CELERITY: HIGH-LEVEL C++ FOR ACCELERATOR CLUSTERS

Peter Thoman, Philip Salzmann, Biagio Cosenza, Thomas Fahringer

In the face of ever-slowing single-thread performance growth for CPUs, the scientific and engineering communities increasingly turn to accelerator parallelization to tackle growing application workloads. Existing means of targeting distributed memory accelerator clusters impose severe programmability barriers and maintenance burdens.

The Celerity programming environment seeks to enable developers to scale C++ applications to accelerator clusters with relative ease, while leveraging and extending the SYCL domain-specific embedded language. By having users provide minimal information about how data is accessed within compute kernels, Celerity automatically distributes work and data.

We introduce the Celerity C++ API as well as a prototype implementation, demonstrating that existing SYCL code can be brought to distributed memory clusters with only a small set of changes that follow established idioms. The Celerity prototype runtime implementation is shown to have comparable performance to more traditional approaches to distributed memory accelerator programming, such as MPI+OpenCL, with significantly lower implementation complexity.

DATAFLOW EXECUTION OF HIERARCHICALLY TILED ARRAYS

Chih-Chieh Yang, Juan C. Pichel, David A. Padua

As the parallelism in high-performance supercomputers continues to grow, new programming models become necessary to maintain programmer productivity at today's levels. Dataflow is a promising execution model because it can represent parallelism at different granularity levels and to dynamically adapt for efficient execution. The downside is the low-level programming interface inherent to dataflow. We present a strategy to translate programs written in Hierarchically Tiled Arrays (HTA) to the dataflow API of Open Community Runtime (OCR) system. The goal is to enable program development in a convenient notation and at the same time take advantage of the benefits of a dataflow runtime system. Using HTA produces more comprehensive codes than those written using the dataflow runtime programming interface. Moreover, the experiments show that, for applications with high asynchrony and sparse data dependences, our implementation delivers superior performance than OpenMP using parallel for loops.

SCALABLE FIFO CHANNELS FOR PROGRAMMING VIA COMMUNICATING SEQUENTIAL PROCESSES

Nikita Koval, Dan Alistarh, Roman Elizarov

Traditional concurrent programming involves manipulating shared mutable state. Alternatives to this programming style are communicating sequential processes (CSP) and actor models, which share data via explicit communication. These models have been known for almost half a century, and have recently had started to gain significant traction among modern programming languages. The common abstraction for communication between several processes is the channel. Although channels are similar to producer-consumer data structures, they have different semantics and support additional operations, such as the select expression. Despite their growing popularity, most known implementations of channels use lock-based data structures and can be rather inefficient.

In this paper, we present the first efficient lock-free algorithm for implementing a communication channel for CSP programming. We provide implementations and experimental results in the Kotlin and Go programming languages. Our new algorithm outperforms existing implementations on many workloads, while providing non-blocking progress guarantee. Our design can serve as an example of how to construct general communication data structures for CSP and actor models.

TWA – TICKET LOCKS AUGMENTED WITH A WAITING ARRAY

Dave Dice, Alex Kogan

The classic ticket lock is simple and compact, consisting of ticket and grant fields. Arriving threads atomically fetch-and-increment ticket to obtain an assigned ticket value, and then wait for grant to become equal to that value, at which point the thread holds the lock. The corresponding unlock operation simply increments grant. This simple design has short code paths and fast handover (transfer of ownership) under light contention, but may suffer degraded scalability under high contention when multiple threads busy wait on the grant field – so-called global spinning.

We propose a variation on ticket locks where long-term waiting threads – those with an assigned ticket value far larger than grant – wait on locations in a waiting array instead of busy waiting on the grant field. The single waiting array is shared among all locks. Short-term waiting is accomplished in the usual manner on the grant field. The resulting algorithm, TWA, improves on ticket locks by limiting the number of threads spinning on the grant field at any given time, reducing the number of remote caches requiring invalidation from the store that releases the lock. In turn, this accelerates handover, and since the lock is held throughout the handover operation, scalability improves. Under light or no contention, TWA yields performance comparable to the classic ticket lock. Under high contention, TWA is substantially more scalable than the classic ticket lock, and provides performance on par or beyond that of scalable queue-based locks such as MCS by avoiding the complexity and additional accesses incurred by the MCS handover operation while also obviating the need for maintaining queue elements.

We provide an empirical evaluation, comparing TWA against ticket locks and MCS for various user-space applications, and within the Linux kernel.

ENABLING RESILIENCE IN ASYNCHRONOUS MANY-TASK PROGRAMMING MODELS

Sri Raj Paul, Akihiro Hayashi, Nicole Slattengren, Hemanth Kolla, Matthew Whitlock, Seonmyeong Bak, Keita Teranishi, Jackson Mayo, Vivek Sarkar

Resilience is an imminent issue for next-generation platforms due to projected increases in soft/transient failures as part of the inherent trade-offs among performance, energy, and costs in system design. In this paper, we introduce a comprehensive approach to enabling application-level resilience in Asynchronous Many-Task (AMT) programming models with a focus on remedying Silent Data Corruption (SDC) that can often go undetected by the hardware and OS. Our approach makes it possible for the application programmer to declaratively express resilience attributes with minimal code changes, and to delegate the complexity of efficiently supporting resilience to our runtime system. We have created a prototype implementation of our approach as an extension to the Habanero C/C++ library (HClib), where different resilience techniques including task replay, task replication, algorithm-based fault tolerance (ABFT), and checkpointing are available. Our experimental results show that task replay incurs lower overhead than task replication when an appropriate error checking function is provided. Further, task replay matches the low overhead of ABFT. Our results also demonstrate the ability to combine different resilience schemes. To evaluate the effectiveness of our resilience mechanisms in the presence of errors, we injected synthetic errors at different error rates (1.0%, and 10.0%) and found modest increase in execution times. In summary, the results show that our approach supports efficient and scalable recovery, and that our approach can be used to influence the design of future AMT programming models and runtime systems that aim to integrate first-class support for user-level resilience.

Multicore and Manycore Parallelism

AVOIDING SCALABILITY COLLAPSE BY RESTRICTING CONCURRENCY

Dave Dice, Alex Kogan

Saturated locks often degrade the performance of a multithreaded application, leading to a so-called scalability collapse problem. This problem arises when a growing number of threads circulating through a saturated lock causes the overall application performance to fade or even drop abruptly. This problem is particularly (but not solely) acute on oversubscribed systems (systems with more threads than available hardware cores).

In this paper, we introduce GCR (generic concurrency restriction), a mechanism that aims to avoid the scalability collapse. GCR, designed as a generic, lock-agnostic wrapper, intercepts lock acquisition calls, and decides when threads would be allowed to proceed with the acquisition of the underlying lock. Furthermore, we present GCR-NUMA, a non-uniform memory access (NUMA)-aware extension of GCR, that strives to ensure that threads allowed to acquire the lock are those that run on the same socket.

The extensive evaluation that includes more than two dozen locks, three machines and three benchmarks shows that GCR brings substantial speedup (in many cases, up to three orders of magnitude) in case of contention and growing thread counts, while introducing nearly negligible slowdown when the underlying lock is not contended. GCR-NUMA brings even larger performance gains starting at even lighter lock contention.

GRAPH COLORING USING GPUS

Meghana Aparna Sistla, V. Krishna Nandivada

Graph coloring is a widely studied problem that is used in a variety of applications, such as task scheduling, register allocation, eigenvalue computations, social network analysis, and so on. Many of the modern day applications deal with large graphs (with millions of vertices and edges) and researchers have exploited the parallelism provided by multi-core systems to efficiently color such large graphs. GPUs provide a promising parallel infrastructure to run large applications. In this paper, we present new schemes to efficiently color large graphs on GPUs.

We extend the algorithm of Rokos et al. [Rokos, G., Gorman, G., Kelly, P.H.J.: A fast and scalable graph coloring algorithm for multi-core and many-core architectures. In: Träff, J.L., Hunold, S., Versaci, F. (eds.) Euro-Par 2015. LNCS, vol. 9233, pp. 414–425. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48096-0_32] to efficiently color graphs using GPUs. Their approach has to continually resolve conflicts for color assignment. We present a data driven variation of their algorithm and use an improved scheme for conflict resolution. We also propose two optimizations for our algorithm to reduce both the execution time and memory requirements. We have evaluated our scheme (called SIRG) against the NVIDIA cuSPARSE library and the work of Chen et al. [Li, P., et al.: High performance parallel graph coloring on GPGPUs. In: IPDPS Workshops, pp. 845–854. IEEE (2016)], and show that SIRG runs significantly faster: geomean 3.42X and 1.76X, respectively. We have also compared SIRG against the scheme of Rokos et al. for CPUs and show that SIRG performs faster on most input graphs: geomean 10.37X.

FEATHERLIGHT SPECULATIVE TASK PARALLELISM

Vivek Kumar

Speculative task parallelism is a widely used technique for solving search based irregular computations such as graph algorithms. Here, tasks are created speculatively to traverse different search spaces in parallel. Only a few of these tasks succeed in finding the solution, after which the remaining tasks are canceled. For ensuring timely cancellation of tasks, existing frameworks either require programmer introduced cancellation checks inside every method in the call chain, thereby hurting the productivity, or provide limited parallel performance.

In this paper we propose Featherlight, a new programming model for speculative task parallelism that satisfies the serial elision property and doesn't require any task cancellation checks. We show that Featherlight improves productivity through a classroom-based study. Further, to support Featherlight, we present the design and implementation of a task cancellation technique that exploits runtime mechanisms already available within managed runtimes and achieves a geometric mean speedup of 1.6X over the popular Java ForkJoin framework on a 20 core machine.

ONE TABLE TO COUNT THEM ALL: PARALLEL FREQUENCY ESTIMATION ON SINGLE-BOARD COMPUTERS

Fatih Taşyaran, Kerem Yıldırır, Mustafa Kemal Taş, Kamer Kaya

Sketches are probabilistic data structures that can provide approximate results within mathematically proven error bounds while using orders of magnitude less memory than traditional approaches. They are tailored for streaming data analysis on architectures even with limited memory such as single-board computers that are widely exploited for IoT and edge computing. Since these devices offer multiple cores, with efficient parallel sketching schemes, they are able to manage high volumes of data streams. However, since their caches are relatively small, a careful parallelization is required.

In this work, we focus on the frequency estimation problem and evaluate the performance of a high-end server, a 4-core Raspberry Pi and an 8-core Odroid. As a sketch, we employed the widely used Count-Min Sketch. To hash the stream in parallel and in a cache-friendly way, we applied a novel tabulation approach and rearranged the auxiliary tables into a single one. To parallelize the process with performance, we modified the workflow and applied a form of buffering between hash computations and sketch updates.

Today, many single-board computers have heterogeneous processors in which slow and fast cores are equipped together. To utilize all these cores to their full potential, we proposed a dynamic load-balancing mechanism which significantly increased the performance of frequency estimation.

FINE-GRAINED MPI+OPENMP PLASMA SIMULATIONS: COMMUNICATION OVERLAP WITH DEPENDENT TASKS

Jérôme Richard, Guillaume Latu, Julien Bigot, Thierry Gautier

This paper demonstrates how OpenMP 4.5 tasks can be used to efficiently overlap computations and MPI communications based on a case-study conducted on multi-core and many-core architectures. It focuses on task granularity, dependencies and priorities, and also identifies some limitations of OpenMP. Results on 64 Skylake nodes show that while 64% of the wall-clock time is spent in MPI communications, 60% of the cores are busy in computations, which is a good result. Indeed, the chosen dataset is small enough to be a challenging case in terms of overlap and thus useful to assess worst-case scenarios in future simulations.

Two key features were identified: by using task priority we improved the performance by 5.7% (mainly due to an improved overlap), and with recursive tasks we shortened the execution time by 9.7%. We also illustrate the need to have access to tools for task tracing and task visualization. These tools allowed a fine understanding and a performance increase for this task-based OpenMP+MPI code.

PARALLEL ADAPTIVE SAMPLING WITH ALMOST NO SYNCHRONIZATION

Alexander van der Grinten, Eugenio Angriman, Henning Meyerhenke

Approximation via sampling is a widespread technique whenever exact solutions are too expensive. In this paper, we present techniques for an efficient parallelization of adaptive (a.k.a. progressive) sampling algorithms on multi-threaded shared-memory machines. Our basic algorithmic technique requires no synchronization except for atomic load-acquire and store-release operations. It does, however, require $O(n)$ memory per thread, where n is the size of the sampling state. We present variants of the algorithm that either reduce this memory consumption to $O(11)$or ensure that deterministic results are obtained.

Using the KADABRA algorithm for betweenness centrality (a popular measure in network analysis) approximation as a case study, we demonstrate the empirical performance of our techniques. In particular, on a 32-core machine, our best algorithm is 2.9X faster than what we could achieve using a straightforward OpenMP-based parallelization and 65.3X faster than the existing implementation of KADABRA.

Theory and Algorithms for Parallel Computation and Networking

PARALLEL STREAMING RANDOM SAMPLING

Kanat Tangwongsan, Srikanta Tirthapura

This paper investigates parallel random sampling from a potentially-unending data stream whose elements are revealed in a series of element sequences (minibatches). While sampling from a stream was extensively studied sequentially, not much has been explored in the parallel context, with prior parallel random-sampling algorithms focusing on the static batch model. We present parallel algorithms for minibatch-stream sampling in two settings: (1) sliding window, which draws samples from a prespecified number of most-recently observed elements, and (2) infinite window, which draws samples from all the elements received. Our algorithms are computationally and memory efficient: their work matches the fastest sequential counterpart, their parallel depth is small (polylogarithmic), and their memory usage matches the best known.

Parallel Numerical Methods and Applications

CHOLESKY AND GRAM-SCHMIDT ORTHOGONALIZATION FOR TALL-AND-SKINNY QR FACTORIZATIONS ON GRAPHICS PROCESSORS

Andrés E. Tomás, Enrique S. Quintana-Ortí

We present a method for the QR factorization of large tall-and-skinny matrices that combines block Gram-Schmidt and the Cholesky decomposition to factorize the input matrix column panels, overcoming the sequential nature of this operation. This method uses re-orthogonalization to obtain a satisfactory level of orthogonality both in the Gram-Schmidt process and the Cholesky QR.

Our approach has the additional benefit of enabling the introduction of a static look-ahead technique for computing the Cholesky decomposition on the CPU while the remaining operations (all Level-3 BLAS) are performed on the GPU.

In contrast with other specific factorizations for tall-skinny matrices, the novel method has the key advantage of not requiring any custom GPU kernels. This simplifies the implementation and favours portability to future GPU architectures.

Our experiments show that, for tall-skinny matrices, the new approach outperforms the code in MAGMA by a large margin, while it is very competitive for square matrices when the memory transfers and CPU computations are the bottleneck of Householder QR.

AUTOMATIC EXPLORATION OF REDUCED FLOATING-POINT REPRESENTATIONS IN ITERATIVE METHODS

Yohan Chatelain, Eric Petit, Pablo de Oliveira Castro, Ghislain Lartigue, David Defour

With the ever-increasing need for computation of scientific applications, new application domains, and major energy constraints, the landscape of floating-point computation is changing. New floating-point representation formats are emerging and there is a need for tools to simulate their impact in legacy codes. In this paper, we propose an automatic tool to evaluate the effect of adapting the floating point precision for each operation over time, which is particularly useful in iterative schemes. We present a backend to emulate any IEEE-754 floating-point operation in lower precision. We tested the numerical errors resilience of our solutions thanks to Monte Carlo Arithmetic and demonstrated the effectiveness of this methodology on YALES2, a large Combustion-CFD HPC code, by achieving 28% to 67% reduction in communication volume by lowering precision.

LINEAR SYSTEMS SOLVERS FOR DISTRIBUTED-MEMORY MACHINES WITH GPU ACCELERATORS

Jakub Kurzak, Mark Gates, Ali Charara, Asim Yar, KhanIchitaro Yamazaki, Jack Dongarra

This work presents two implementations of linear solvers for distributed-memory machines with GPU accelerators—one based on the Cholesky factorization and one based on the LU factorization with partial pivoting. The routines are developed as part of the Software for Linear Algebra Targeting Exascale (SLATE) package, which represents a sharp departure from the traditional conventions established by legacy packages, such as LAPACK and ScaLAPACK. The article lays out the principles of the new approach, discusses the implementation details, and presents the performance results.

Accelerator Computing

* BEST PAPER 2019 RADIO-ASTRONOMICAL IMAGING: FPGAS VS GPUS

Bram Veenboer, John W. Romein

FPGAs excel in performing simple operations on high-speed streaming data, at high (energy) efficiency. However, so far, their difficult programming model and poor floating-point support prevented a wide adoption for typical HPC applications. This is changing, due to recent FPGA technology developments: support for the high-level OpenCL programming language, hard floating-point units, and tight integration with CPU cores. Combined, these are game changers: they dramatically reduce development times and allow using FPGAs for applications that were previously deemed too complex.

In this paper, we show how we implemented and optimized a radio-astronomical imaging application on an Arria 10 FPGA. We compare architectures, programming models, optimizations, performance, energy efficiency, and programming effort to highly optimized GPU and CPU implementations. We show that we can efficiently optimize for FPGA resource usage, but also that optimizing for a high clock speed is difficult. All together, we demonstrate that OpenCL support for FPGAs is a leap forward in programmability and it enabled us to use an FPGA as a viable accelerator platform for a complex HPC application.

CONFERENCE PROGRAMME OVERVIEW

PLEASE NOTE

The conference venue is - if not explicitly specified
Conference Centre Alte Mensa, Wilhelmsplatz 3, 37073 Göttingen ▶ Map

Conference Programme - Details

Wednesday, 28.08.2019 *** Thursday, 29.08.2019 *** Friday, 30.08. 2019