Abstract

Overview :

In the past, particle physics has relied upon improvements in processing speed of single computing cores in order to improve data acquisition rates. This feature will be critical to the proposed "High Luminosity Large Hadron Collider" (HLLHC). Unfortunately, since the mid-2000's, this improvement in single-core processing speed has hit a limit, and improvements in processing time must now come from concurrent processing. However, the current reconstruction techniques are not enabled for concurrent processing. In order to continue improving the processing capabilities of particle physics experiments in the HLLHC era, it will be necessary to explore cutting-edge techniques in parallel processing.

There are several general classes of problems in particle physics event reconstruction that could be modified in order to achieve concurrent processing. One such opportunity that has not yet been explored is in "jet clustering," a nearest-neighbor type of algorithm used to cluster hadronically-fragmented jets into a single object.

Intellectual Merit :

This proposal focuses on parallelizing the existing jet clustering algorithms in use at the LHC experiments. The proposed improvements will be to use this as a test case for deployment of cutting-edge parallelization techniques such as lightweight concurrency extraction, speculative computing, and smarter distribution. Some recent experience shows that this nearest-neighbor type of algorithm used by the jet clustering is amenable to such improvements.

Broader Impacts :

The benefits of this proposal are twofold : firstly, there will be an immediate improvement of the jet clustering algorithms themselves that will lead to higher data acquisition rates at the LHC. Secondly, the computing techniques developed could be used in other applications, inside of particle physics and elsewhere. Since nearest-neighbor algorithms are ubiquitious in scientific computing, it is expected that techniques developed to parallelize this particular problem will be applicable to a wide variety of others in academia and industry.

In addition, these core developments can train students in the newest computing techniques, giving them cutting-edge experience that is highly relevant in academia and private industry.

Clustering Jets at the Exascale

Mark Adams, Steven Ko, Salvatore Rappoccio, Lukasz Ziarek

November 7, 2013

1 Project Overview

With the discovery of the Higgs boson by the Large Hadron Collider (LHC) experiments ATLAS and CMS [6, 7], the standard model (SM) of particle physics is now complete. This model unifies the electromagnetic force (carried by the *photon*) with the weak force, responsible for radioactive decay (carried by the *W and Z bosons*). At long last, physicists now understand that via interactions with the Higgs field, the W and Z bosons acquire a mass, but the photon does not. This is referred to as "electroweak symmetry breaking".

A new phase of particle physics has therefore begun. The questions have shifted from the cause of electroweak symmetry breaking, to the study of the Higgs boson and its interactions in detail. To understand the larger picture of the fundamental forces in nature, the past excellence of the LHC experiments must therefore continue unabated in the face of new technical challenges.

One of the major technical challenges that lies ahead is the continuation of the scaling of computational power year by year, known colloquially as "Moore's Law". To set the scale, at the CMS experiment with the LHC collision flux ("luminosity") reaching $7 \times 10^{33}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$, the processing time to reconstruct each collision event by CMS was approximately 20 seconds per event. However, as the luminosity is increased, the computational time currently scales quadratically. As the upgraded LHC is expected to deliver $> 12 \times 10^{33}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$ in the upcoming run, the processing time per event is expected to reach several minutes per event as shown in Figure 1. Furthermore, in future runs of the LHC in the next 15 years, the luminosity is expected to reach as high as $> 1 \times 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$, which would correspond (naively)

to several hours of computational time per event! Clearly, it is necessary for the computing power to scale in order to compensate for this dramatic increase in CPU time with increasing luminosity.

A critical element in the LHC physics program is the clustering of final-state particles produced in collisions into groups called "jets". Jets are produced via quantum chromodynamic (QCD) interactions of quarks and gluons that hadronize.

the hadronization of outgoing final-state particles from the core interaction of the collision. Due to the asymptotic freedom of the QCD interaction, the strong force is effectively very weak at higher energies, but grows quickly at lower energies, due to the process of "anti-screening" of the color charge by gluon fields. This results in the phenomenon that the final states of colored particles tend to spray in an extended region of space, while the centroid of that spray closely approximates the direction of the progenitor gluon or quark. Thus, the goal of any "jet clustering" algorithm is to accumulate the final-state particles

These "jet clustering" algorithms group the particles This algorithm can become very computationally expensive as the number of particles that are produced in a collision grows, scaling as $N^2 \log N$.

However, with the expected end of the historic scaling of single-core processing capability [8], it is imperative to utilize a parallel processing strategy in order to maintain the levels of computational speed that are required.

Oftentimes, the codes used by CMS (and experimental HEP in general) tend to lack clear numerical "kernels" where optimization efforts can be focused. Given these characteristics they are generally more properly classified as "high throughput computing" (HTC) rather than "high performance computing" (HPC). In terms of their detailed behavior on the CPU many of these codes resemble more general enterprise or "cloud" applications [9, 10].

1.1 Jet Clustering

The energetic deposits of particles in detectors need to be clustered to obtain the complete response. This is because the process inherently involves a shower of particles called a "jet". This "jet clustering" is a well-established technique employed at many different particle physics experiments worldwide, and is implemented in a common software framework called `fastjet` [11]. The mathematical problem is analo-

gous to the "K-nearest neighbors algorithm" [12] (kNN). The single-core optimization of jet clustering is outlined in Ref. [13]. In a single core, the computational time scales as $O(N^2)$ or $O(N \ln N)$, where N is the number of inputs to the algorithm, which scales linearly with luminosity.

There is existing work and literature on the topic of the parallelization of the kNN algorithm, for instance, in Refs. [14, 15, 16], where improvements $O(100)$ in CPU performance are observed over standard algorithms. Since the proposed use case is very similar to the kNN algorithm, similar improvements to the processing time by parallelization strategies are expected.

We now discuss specific strategies that can be developed to optimize performance in this algorithm.

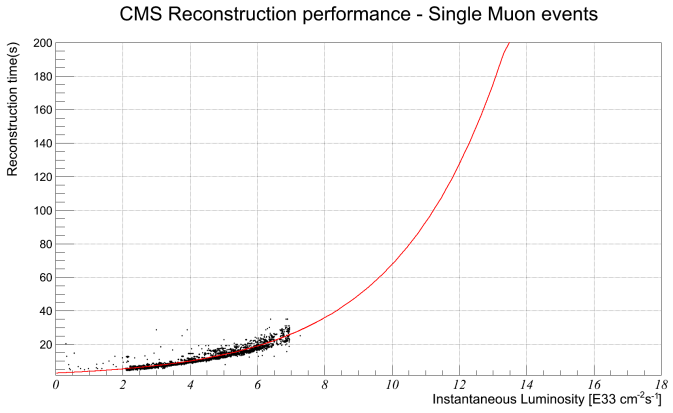

Figure 1: Event processing time versus instantaneous luminosity.

1.2 Full Stack Parallelization

To achieve the necessary improvements in performance required for scalability of jet clustering, we propose to examine parallelization opportunities across the entire software stack, including three specific areas : (1) the use of lightweight concurrency extraction to mask high-latency computations or I/O actions, (2) extraction of parallelization from the computation itself in the form of optimistic speculation and specialized transform, and (3) new methods for distributing the computation to maximize parallelization on each node.

Lightweight Concurrency for Latency Masking

Many mathematical kernels contain opportunities for extracting "micro parallelism," usually on the order of tens of instructions, from their computational components. Unfortunately, it is very difficult to parallelize this computation profitably as the overhead of thread creation, scheduling, synchronization, and migration outweigh the gains in parallelism. Instead of extracting explicit parallelism from such computations, we propose to explore methods of lightweight asynchrony to allow for computation to proceed while waiting on high latency I/O operations to complete or the results of other computations. Since the creation of threads and associated schedule and synchronization costs are typically prohibitive, we will explore new threading models that allow for logically-distinct computations to execute within a given construct. The PIs previous research has indicated that such schemes can profitably boost overall performance in the context of ML code [17, 18].

Speculative Computation

In addition to exploring explicit parallelization of the numeric kernels in jet clustering, we propose to explore extraction of parallelism via speculative computation. At its core, speculative computation breaks apart sequential or parallel tasks into smaller tasks to be run in parallel. Once the speculation has completed, the runtime system validates the computation. If the computation is incorrect (*i.e.* a "datarace" is detected, the computation cannot be serialized, *etc.*), the incorrect computation is re-executed in a non-speculative manner. If the rate of mis-speculation is low, such techniques can be leveraged to extract additional parallelism. The PIs have extensive experience with transactional memory [19], lightweight rollback methods [20], leveraging memoization to reduce re-computation costs [21, 22], and deterministic speculation [23]. We propose to explore a specialized speculation framework leveraging different speculation strategies, including speculation extracted by the programmer via programming language primitives, library level speculation, and compiler extracted speculation.

Smart Distribution

In order to increase parallelism, we will explore the use of the MapReduce execution framework [24, 25]. MapReduce is a runtime

system recently developed for large-scale parallel data processing. It enables programmers to easily deploy their applications on a cluster of machines. Programmers only need to write two functions, Map and Reduce, and submit these two functions as a job to the system. Then the MapReduce framework takes care of all the aspects of the execution of the job. For example, the framework packages and distributes the two functions over the cluster so that the whole cluster can be utilized to execute the job; it also takes care of fault-tolerance by monitoring the cluster during the execution of the job and redistributes the job if some machine fails.

Due to this simplicity and power, it is quickly gaining popularity in industry for large-scale data processing. Many applications in scientific computing have not yet explored the use of MapReduce in depth, however previous research has explored implementing kNN with MapReduce [26, 27]. We intend to explore this question in the context of jet clustering for the LHC.

2 Prior Work

The principle investigators (PIs) of this proposal have a widely-varied and applicable skill set to accomplish the goals of extending LHC computing to the exascale.

- Salvatore Rappoccio has 15 years of experience programming in a high-energy physics environment, as well as other numerical software design for the private sector. He is an expert in jet reconstruction at CMS, maintaining and developing this software for the last five years.
- Lukasz Ziarek has 9 years of experience in language, compiler, and runtime design targeted at improving multicore performance. He has worked on 5 compilers and 3 Java VMs. He is an expert at speculative and transactional computation focusing on the extraction of parallelism and lightweight concurrency.
- Steven Ko has 10 years of experience in distributed systems. His recent focus has been large-scale data processing in the cloud using MapReduce and other technologies built on top of it. He also has 5 years of experience in large-scale storage and data management in data centers.

Need more discussion here.

3 Outreach and Education

Any ideas for outreach?

3.1 Summary

In summary, the problem of expanding LHC computing to the exascale is a difficult, but tractable one. This proposal investigates the possibility of applying cutting-edge parallelization techniques such as lightweight concurrency extraction, speculative computation, and smarter distribution, to the real-world application of LHC data processing. The overall goal is to reduce the computational time for k-nearest-neighbor-like numerical kernels used for jet clustering. The investigators of this proposal have extensive experience in the various aspects of the problem, and the synergistic application of this experience is expected to attain considerable improvements in this area, which are absolutely critical to the success of the future LHC physics program.

References

[1] US Department of Energy Office of Science. The Opportunities and Challenges of Exascale Computing. `http://science.energy.gov/~/media/ascr/ascac/pdf/reports/Exascale_subcommittee_report.pdf`, 2010.

[2] US Department of Energy Office of Science. Congressional Budget for ASCR. `http://science.energy.gov/~/media/budget/pdf/sc-budget-request-to-congress/fy-2013/Cong_Budget_2013_ASCR.pdf`, 2013.

[3] National Science Foundation. Core Techniques and Technologies for Advancing Big Data Science & Engineering (BIGDATA). `http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=504767`, 2013.

[4] National Science Foundation. High Performance System Acquisition: Building a More Inclusive Computing Environment for Science and Engineering. `http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503148`, 2013.

[5] Google. Faculty Research Awards. `http://research.google.com/university/relations/research_awards.html`, 2013.

[6] Serguei Chatrchyan et al. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Phys. Lett. B*, 2012.

[7] Georges Aad et al. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Phys. Lett. B*, 2012.

[8] Samuel H. Fuller and Lynette I. Millett. *The Future of Computing Performance: Game Over or Next Level?* The National Academies Press, 2011. Committee on Sustaining Growth in Computing Performance; National Research Council.

[9] Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. In *Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '12, 2012.

[10] P. Calafiura, S. Eranian, D. Levinthal, S. Kama, and R.A. Vitillo. GOoDA: The generic optimization data analyzer. *J.Phys.Conf.Ser.*, 396:052072, 2012.

[11] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. FastJet User Manual. *Eur.Phys.J.*, C72:1896, 2012.

[12] T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.

[13] Matteo Cacciari and Gavin P. Salam. Dispelling the N^3 myth for the k_t jet-finder. *Phys.Lett.*, B641:57–61, 2006.

[14] Vincent Garcia, Eric Debreuve, Frank Nielsen, and Michel Barlaud. k-nearest neighbor search: fast GPU-based implementations and application to high-dimensional feature matching. In *IEEE International Conference on Image Processing (ICIP)*, Hong Kong, China, September 2010.

[15] V. Garcia, E. Debreuve, and M. Barlaud. Fast k nearest neighbor search using gpu. In *CVPR Workshop on Computer Vision on GPU*, Anchorage, Alaska, USA, June 2008.

[16] Vincent Garcia. *Suivi d'objets d'intrt dans une squence d'images : des points saillants aux mesures statistiques.* PhD thesis, Universit de Nice - Sophia Antipolis, Sophia Antipolis, France, December 2008.

[17] Lukasz Ziarek, KC Sivaramakrishnan, and Suresh Jagannathan. Composable asynchronous events. In *ACM SIGPLAN Notices*, volume 46, pages 628–639. ACM, 2011.

[18] KC Sivaramakrishnan, Lukasz Ziarek, Raghavendra Prasad, and Suresh Jagannathan. Lightweight asynchrony using parasitic threads. In *Proceedings of the 5th ACM SIGPLAN workshop on Declarative aspects of multicore programming*, pages 63–72. ACM, 2010.

[19] Lukasz Ziarek, Adam Welc, Ali-Reza Adl-Tabatabai, Vijay Menon, Tatiana Shpeisman, and Suresh Jagannathan. A uniform transactional execution environment for java. *ECOOP 2008–Object-Oriented Programming*, pages 129–154, 2008.

[20] Lukasz Ziarek and Suresh Jagannathan. Lightweight checkpointing for concurrent ml. *Journal of Functional Programming*, 20(02):137–173, 2010.

[21] Lukasz Ziarek and Suresh Jagannathan. Memoizing multi-threaded transactions. *Workshop on Declarative Aspects of Multicore Programming*, 2008.

[22] Lukasz Ziarek, KC Sivaramakrishnan, and Suresh Jagannathan. Partial memoization of concurrency and communication. *ACM Sigplan Notices*, 44(9):161–172, 2009.

[23] Lukasz Ziarek, Siddharth Tiwary, and Suresh Jagannathan. Isolating determinism in multi-threaded programs. *Runtime Verification*, pages 63–77, 2012.

[24] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the 6th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2004.

[25] Hadoop MapReduce. `http://hadoop.apache.org/mapreduce`.

[26] Wei Lu, Yanyan Shen, Su Chen, and Beng Chin Ooi. Efficient processing of k nearest neighbor joins using mapreduce. *Proc. VLDB Endow.*, 5(10):1016–1027, June 2012.

[27] Chi Zhang, Feifei Li, and Jeffrey Jestes. Efficient parallel knn joins for large data in mapreduce. In *Proceedings of the 15th International Conference on Extending Database Technology*, EDBT '12, pages 38–49, New York, NY, USA, 2012. ACM.

BUDGET JUSTIFICATION

Institution : The State University of New York at Buffalo (UB)
PI : Salvatore Rappoccio

Personnel

The requested funds of $XXXXXX USD (for 3 years starting in 2014) would cover two (2) months of summer salary for Profs. Ko, Rappoccio, and Ziarek per year for three (3) years ($XXXXXX), the full salary for one (1) computer-science-based postdoctoral fellow for three (3) years ($XXXXX), and salary plus tuition for two (2) graduate students, one physics-based and the other computer-science-based, totaling $XXXXXXX in salary and $XXXXXX in tuition.

Travel

This research will require moderate travel to CERN for collaboration with the `fastjet` authors occasionally. Hence, the proposal requests $15,000 in travel funds for the three years of activity.

Facilities and Administration Indirect Costs

The above figures carry an indirect cost percentage of 56%, totaling $XXXXXX.

Facilities, Equipment and Other Resources

Laboratory Space

The UB group has two 900 sqft labs, VME/NIM crates, several PC's, a Tektronix Logic Analyzer, several scintillator-based muon detectors with photo-multiplier tubes, and a machine shop. In addition, the UB group has extensive access to the Silicon Detector Laboratory Facility ("SiDet") at FNAL.

Center For Computational Research

UB has a large computational research center, CCR, which is a Linux-based cluster on the Open Science Grid, and also has a large GPU cluster for possible parallel processing developments.

In addition, there are **BLA BLA BLA BLA BLA** personal computing resources, funded by the startup funds of the PI's.

Office

The faculty, postdoctoral fellows and graduate students will all have office space at UB.

Postdoctoral Fellow Mentoring

One postdoctoral fellow will be funded on this project. There are extensive postdoctoral fellowship mentoring activities at UB, as well as at the LHC Experiments at CERN. These include guidance in career paths, work/life balance discussions, and technical skill development such as writing grant proposals, etc. Specific elements are highlighted below.

- **University at Buffalo (UB)**
  - The UB Office of Postdoctoral Scholars offers diverse services for postdoctoral fellows, including the *"Postdoc Survival Skills Workshops"*, targeted seminars and symposia for postdoctoral fellows, social functions, and logistical assistance.
  - The UB Physics Department offers several services to our postdoctoral fellows, including a biweekly Journal Club for particle physics and cosmology, weekly seminars and colloquia, and weekly social functions inside the department.
- **CMS Experiment and CERN**
  - As at FNAL, the opportunities for a postdoctoral fellow at CERN and at CMS are extensive. There are also a plethora of workshops, seminars, conferences, etc, at CERN. There are also smaller weekly avenues for networking possibilities, as well as seminars for postdoctoral fellows to gain visibility for their work.
  - It is also worth pointing out that, because of the world-class nature of CERN, it often attracts very high-level members of the particle physics community on a regular basis. Such opportunities for visibility among the top-tier scientists in the world (including Nobel and Milner Prize winners, etc) are hard to understate.

In all, the postdoctoral fellow that will be supported by this proposal will have ample opportunities for professional advancement and development, as well as a myriad of opportunities for a community of peers in both professional and social settings.

Data Management Plan : Research Data

The LHC experiments are dedicated to timely dissemination of their data and procedures, in addition to documentation of results in publications and journal articles. The LHC experiments (including CMS) are world leaders in grid computing and cloud-like data storage, and the solutions that have been developed have robustly handled the many petabytes of data that have been collected. There are several "tiers" of data, which are designated by how widely they are deployed throughout the Open Science Grid (OSG) and the other LHC sites.

In addition to these "tiers" of the actual data collected, there are also software and documentation schema for the LHC data and analysis. The software that will be derived via the activities of this proposal will be stored in the `fastjet` central repository `http://fastjet.fr`, which is open-source, freely-accessible, and well-maintained for the benefit of the entire HEP community at large.