Introducing the graph 500

Article · January 2010

CITATIONS
171

READS
766

4 authors:

James Ang
Sandia National Laboratories
27 PUBLICATIONS **327** CITATIONS
SEE PROFILE

Brian Barrett
University of Cambridge
55 PUBLICATIONS **2,197** CITATIONS
SEE PROFILE

Kyle B. Wheeler
Amazon
22 PUBLICATIONS **510** CITATIONS
SEE PROFILE

Richard Murphy
Micron Technology, Inc.
29 PUBLICATIONS **630** CITATIONS
SEE PROFILE

Introducing the Graph 500

Richard C. Murphy Kyle B. Wheeler Brian W. Barrett
James A. Ang
Sandia National Laboratories*
, PO Box 5800, MS-1319, Albuquerque, NM 87185
{rcmurph,kbwheel,bwbarre,jaang}@sandia.gov

May 5, 2010

1 Introduction: Why Another Benchmark?

In the words of Lord Kelvin, "if you cannot measure it, you cannot improve it". One of the long-lasting successes of the Top 500 list is sustained, community-wide floating point performance improvement. Emerging large-data problems, either resulting from measured real-world phenomena or as further processing of data generated by simulation, have radically different performance characteristics and architectural requirements. As the community contemplates scaling to large-scale HPC resources to solve these problems, we are challenged by the reality that supercomputers are typically optimized for the 3D simulation of physics, not large-scale, data-driven analysis. Consequently, the community contemplating this kind of analysis requires a new yard stick for evaluating future platforms.

Since the invention of the von Neumann architecture, the physics simulation has largely driven the development and evolution of High Performance Computing. This allows scientists and engineers to test hypotheses, designs, and ask "what if" questions. Emerging informatics and analytics applications are different both in purpose and structure. While physics simulations typically are core-memory sized, floating point intensive, and well-structured, informatics applications tend to be out of core, integer oriented, and unstructured. (It could be argued that physics simulations are moving in this direction.) The graph abstraction is a powerful model in computer science and discrete math for addressing informatics problems. The purpose of informatics is not typically to ask a "what if," but to search through large datasets and discover hypotheses to be tested. Today, these data sets are collected through mechanisms including scientific experiments, simulation outputs, and social network formation. The Graph 500 list recognizes these fundamental differences and serves to focus the community. This is analogous to and inspired by the Top 500 list, which served as a focal point for the simulation of physics community.

Work at Sandia has demonstrated that the performance properties of large-scale graph problems are very different from those of physics applications. Complex graph applications exhibit very low spatial and temporal locality (or reuse of data near data already used and reuse of data over time respectively). It also exhibits a significantly larger dataset than is typical for real-world physics applications or industry benchmarks. Compared to the LINPACK benchmark that defines the Top 500 list, an example graph problem consumes 6,900 times the unique data, and exhibits 2.6% the temporal reuse and 55% the temporal reuse[8].

The Graph 500's defining benchmark and associated metrics are being formed by the Graph 500 steering committee. The list and associated benchmark will be announced at the International Supercomputing Conference in June of 2010, and the first ranking unveiled at Supercomputing 2010 in November

To succeed, this new benchmark must exhibit the following properties:

1. **Fundamental Kernel with Broad Application Reach:** rather than a single transitory point application, the benchmark must reflect a class of algorithms that impact many applica-

*Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

tions.

2. **Map to Real World Problems:** results from the benchmark should map back to actual problems, rather than exhibiting a theoretical or "pure computer science" result.

3. **Reflect Real Data Sets (that are important!):** similar to mapping to real problems, the data sets should exhibit real-world patterns. Application performance for many real-world problems is highly input-deck dependent.

Many of these properties have been observed with prior benchmark suites[2, 8, 7, 1, 5].

The proposed benchmark should push industry to build machines that benefit the new application set, which in turn needs to be economically important enough to motivate industry R&D investments. Graphs are a fundamental data structure that challenge current architectures and that appear in key large-scale data analysis problems. Many of those problems (discussed in Section 2) are large enough to support industry architecture investments – indeed, they have the potential to far surpass traditional high performance computing.

The remainder of this paper is organized as follows: Section 2 enumerates the business areas underlying the Graph 500 group's thinking. Section 3 describes the high-level kernels we propose as fundamental problems. Section 4 describes the reference implementations we will make available to the community. Section 5 describes the trends in architecture and underlying implementation technology that we hope to impact. Section 6 provides some preliminary results, and Section 7 provides the conclusions.

2 Data Sets

Table 1 summarizes the Graph 500 benchmark's proposed data sets, representing critical business and science problems. Many are easily beyond petascale today, and limited in size only by available computing resources.

3 Kernels

We propose three key classes of graph kernels with multiple possible implementations, several of which have been demonstrated to stress modern computers[8, 6].

1. **Search:** Concurrent Search, requiring traversal of the graph and labeling or identification of a result. A parallel Breadth First Search using Delta Stepping is an example of this kind of search, though the final Graph 500 kernel may have more relaxed requirements.

2. **Optimization:** Single Source Shortest Path is a core example optimization problem proposed by the Graph 500 committee. Many parallel algorithms exist.

3. **Edge Oriented:** is an independent set in a graph that is not a subset of any other independent set.

There are numerous practical requirements for creating a benchmark from these core kernels. For example, the Edge Oriented kernel lends itself to solutions from random algorithms, which if chosen correctly against known data sets could produce overly optimistic or nonreproducible results. These approaches will be eliminated in the rules for benchmark submission.

4 Reference Implementations

The Graph 500 committee will allow both reference implementation and highly optimized results to be submitted to the list, very much like the SPEC benchmark's base and peak measures.

Distributed Memory: This is the dominant model for high performance computing, and is reflected in the Parallel Boost Graph Library (PBGL)[4].

Cloud/MapReduce: More reflective of industry trends, and represents a key programming model for loosely coupled architectures.

Multithreaded Shared Memory: This reference implementation will support large-scale shared memory machines, like SMPs or the Cray XMT. Currently, this is the preferred programming model for many graph applications. This is reflected in Sandia's Multithreaded Graph Library (MTGL)

The Graph 500 committee will allow alternative implementations to enable platforms like the LexisNexis Data Analytic Supercomputer (DAS) which rely on customized programming languages and runtime environments rather than custom hardware to solve graph problems.

Table 1: Example Large-Scale Data Sets

Area	Typical Size	Description
Cybersecurity	15 Billion Log Entries/Day (for large enterprises)	Full data scan with end-to-end join required
Medical Informatics	50M patient records, 20-200 records/patient, billions of individual records	Entity Resolution Required
Data Enrichment	Petabytes of Data (or more)	Example, Maritime Domain Awareness with hundreds of millions of transponders, tens of thousands of ships, and tens of millions of pieces of bulk cargo
Social Networks	Almost Unbounded	Example, Facebook
Symbolic Networks	Petabytes	Example, human brain: 25B neurons with approximately 7K connections each

5 Architecture and Technology Trends

Graph problems are often more difficult to optimize on modern architectures than their scientific or industry counterparts[8]. They typically require "global reach" across the machine, whereas scientific applications primarily require local communication (across the six faces of a cube in a 3D decomposition). This is often expressed as a requirement for a global address space, but is more accurately thought of as a requirement for efficient global namespace management. Regardless of implementation (layer or mechanism), most platforms provide poor global reach. Graph problems often present very sparse, fine-grained data access, requiring a very high message rate from the network (regardless of whether or not the algorithm is a distributed or shared memory algorithm). This is again challenging for supercomputer platforms, especially those designed to facilitate bulk message transfers in a BSP style of computation. As a result, caches and other capabilities typical in modern processor design tend to have limited impact on these problems.

Unfortunately, underlying CMOS technology trends reinforce trends in architectures that will not support the business areas described in Section 2.

Figure 1 depicts the trend in chip packaging versus transistors. The projections show that over the next decade, every communication channel will have to support an order of magnitude more transistors than today. Technologies such as moving to serial signaling, 3D integration, and other disruptive techniques may provide a one-time gain over conventional approaches. However, regardless of the technology

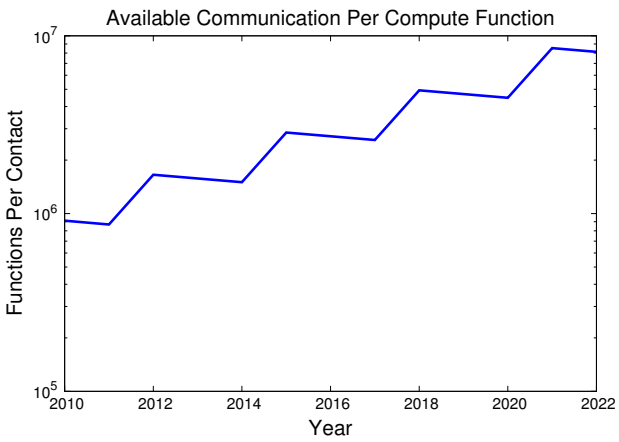

Figure 1: Number of functions (transistors) supported by each off-chip contact available for communication according to the ITRS 2008 roadmap[3].

every communication channel will have to support significantly more computation than it does today. Given the sparse, fine-grained communication patterns required by the problems supported by the Graph 500 list, this trend in technology will push architectures away from supporting the Graph 500 application base. Making the problem space relevant to computer architects is a key contribution of the Graph 500.

The compute/communication trend towards imbalance has a historical analog in the Memory Wall[9]. Originally the memory wall represented a disparity between memory access times and processor cycle times. With processor clock rates flattening, this disparity will stabilize (or possibly even improve).

The processor/memory communication model will not improve without significant investment. This can be seen in today's memory systems that may support increased bandwidth, but decreased capacity (per core) and fewer independent channels. Key architectural techniques including caching, out-of-order execution, branch prediction, and predication were designed to allow processors to cope with the memory wall. Modern architectures are exhibiting increased capabilities for structured memory access (as seen in GPUs), scratchpad memories to exploit high temporal locality operations, and well-scripted communication. These are all designed to address the disparity between compute capabilities and communication channels, but it is unclear if they will significantly impact large-scale data problems exemplified by the Graph 500 kernels.

6 Preliminary Results

Figure 2 shows some preliminary results for the concurrent search benchmark. The problem is 14GB graph generated by R-MAT with the following parameters: $a = 0.57$, $b = 0.19$, $c = 0.19$, and $d = 0.05$, which provides a steep degree distribution power-law graph. This produces a maximum degree of approximately $200,000$, with 2^{25} vertices and 2^{28} edges.

Figure 2(a) depicts the results for the XMT and (b) shows results for a Nehalem, Niagara2, and an Altix 37000 platform. Unfortunately, it is very difficult to fairly and accurately compare smaller SMP platforms with the XMT, despite the fact that XMT provides better absolute performance for this small problem. We would caution against direct comparison between the two datasets. There are two reasons for not comparing across those platform sets:

First, the problem is fundamentally unstructured and responds well to increased memory parallelism. The XMT hashes memory throughout the machine, thus without re-wiring the platform it is impossible to "restrict" the memory to a small fixed number of nodes. In effect, every memory controller on the machine is used for every problem. While this is an advantage for single benchmark runs with much of the machine idle, it is a disadvantage when multiple processes are competing for bandwidth across the machine (which would be a more realistic system utilization scenario outside benchmarking).

Second, the MTGL-based implementation of search is highly tuned for the XMT platform, and not necessarily for other platforms. An apples-to-apples comparison running the same code is consequently unfair, and similarly highly optimized implementations on other platforms are only now being generated.

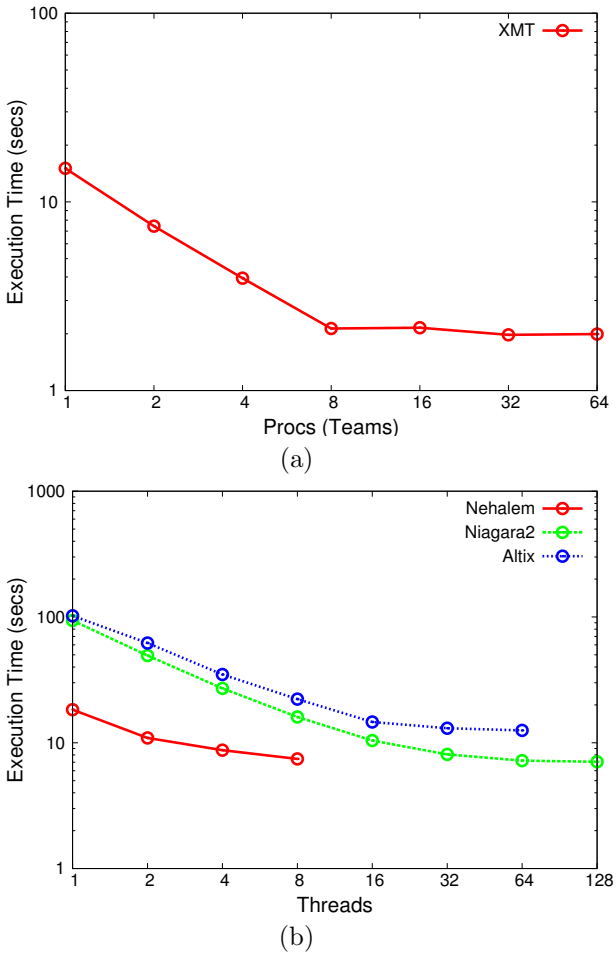

Figure 2: Concurrent Search Results on (a) XMT and (b) Nehalem, Niagara2, and Altix Platforms

The Graph 500 benchmark will address the latter problem, and may address the former by requiring full-memory sized runs with normalization of the results *a posteriori*.

All the results show limited scalability, reflecting the difficulty in addressing even simple graph problems.

7 Conclusions

This paper has discussed the need to create a Graph 500 list, how the problems represented by the pro-

posed benchmarks are different from scientific computing and commercial problems, and the implications for commercial architectures. The proposed Graph 500 data sets represent real-world applications of interest in science, engineering, and business, and the algorithms are core to many large-scale data problems. We propose three reference implementations for shared memory/multithreaded, distributed memory, and MapReduce platforms.

We have demonstrated that there is a pressing need for the Graph 500 benchmark to address the disparity between architectural capabilities enabled by the underlying technology and application requirements. These trends, combined with the existing challenges for running large-scale graph applications require an application push from an economically important market segment to drive industry towards viable solutions.

We plan to roll out the Graph 500 benchmark in multiple venues over the summer of 2010, and compile the first list in the Fall of 2010.

Acknowledgments

This paper reflects the deliberations of the Graph 500 Steering Committee, and preliminary results generated at Sandia to help motivate the problem. The authors would like to thank the other members of the Graph 500 committee for their dedication to creating a new benchmark, their time in debating the right path forward, and their considered opinions (which we hope are also reflected in this work). Errors or omissions may be blamed on the authors of this paper. The other committee members are: David Bader (Georgia Tech), Jon Berry (Sandia), Bill Brantley (AMD), Almadena Chtchelkanova (NSF), John Daly (DoD), John Feo (PNL), Michael Garland (NVIDIA), John Gilbert (UCSB), Bill Harrod (DARPA), Bruce Hendrickson (Sandia), Jure Leskovec (Stanford), Bob Lucas (USC/ISI), Andrew Lumsdaine (Indiana University), Mike Merrill (DoD), Hans Meuer (University of Mannheim), David Mizel (Cray), Shoaib Mufti (Cray), Nick Nystrom (PSC), Fabrizio Petrini (IBM), Wilf Pinfold (Intel), Steve Poole (ORNL), Arun Rodrigues (Sandia), Rob Schreiber (HP), John Simmons (LexisNexis), Marc Snir (UIUC), Thomas Sterling (LSU), Blair Sullivan (ORNL), T.C. Tuan (DoD), Jeff Vetter (ORNL), Mike Vildibill (Oracle).

References

[1] Vishal Aslot and Rudolf Eigenmann. Performance characteristics of the spec omp2001 benchmarks. *SIGARCH Comput. Archit. News*, 29(5):31–40, 2001.

[2] Daniel Citron, John Hennessy, David Patterson, and Gurindar Sohi. The use and abuse of SPEC: An ISCA panel. *IEEE Mico*, 23(4):73–77, July-August 2003.

[3] ITRS Committee. International Technology Roadmap for Semiconductors 2008 Update. `http://www.itrs.net/Links/2008ITRS/Home2008.htm`.

[4] Nick Edmonds, Alex Breuer, Douglas Gregor, and Andrew Lumsdaine. Single-source shortest paths with the Parallel Boost Graph Library. In *The Ninth DIMACS Implementation Challenge: The Shortest Path Problem*, Piscataway, NJ, November 2006.

[5] Swathi Tanjore Gurumani and Aleksandar Milenkovic. Execution characteristics of SPEC CPU2000 benchmarks: Intel C++ vs. Microsoft VC++. In *Proceedings of the 42nd annual Southeast regional conference*, pages 261–266. ACM Press, 2004.

[6] Richard C. Murphy. On the Effects of Memory Latency and Bandwidth on Supercomputer Application Performance. In *IEEE International Symposium on Workload Characterization 2007 (IISWC2007)*, September 27-29, 2007.

[7] Richard C. Murphy, Jonathan Berry, William McLendon, Bruce Hendrickson, Douglas Gregor, and Andrew Lumsdaine. DFS: A Simple to Write Yet Difficult to Execute Benchmark. In *IEEE International Symposium on Workload Characterization 2006 (IISWC06)*, October 25-27, 2006.

[8] Richard C. Murphy and Peter M. Kogge. On the Memory Access Patterns of Supercomputer Applications: Benchmark Selection and its Implications. *IEEE Transactions on Computers*, 56(7):937–945, July 2007.

[9] Wm. A. Wulf and Sally A. McKee. Hitting the memory wall: Implications of the obvious. *Computer Architecture News*, 23(1):20–24, 1995.