© 2014 IJIRT | Volume 1 Issue 5 | ISSN : 2349-6002

THE LANDSCAPE OF PARALLEL COMPUTING RESEARCH

Vivek Gusain, Varsha Chauhan
Department of Information technology
Dronacharya College of Engineering, Gurgaon, India

Abstract- **The recent switch to parallel microprocessors is a milestone in the history of computing. Industry has laid out a roadmap for multicore designs that preserves the programming paradigm of the past via binary compatibility and cache coherence. Conventional wisdom is now to double the number of cores on a chip with each silicon generation. multidisciplinary group of Berkeley researchers met nearly two years to discuss this change. Our view is that this evolutionary approach to parallel hardware and software may work from 2 or 8 processor systems, but is likely to face diminishing returns as 16 and 32 processor systems are realized, just as returns fell with greater instruction-level parallelism.**

I. INTRODUCTION

The computing industry changed course in 2005 when Intel followed the lead of IBM's Power 4 and Sun Microsystems' Niagara processor in announcing that its high performance microprocessors would henceforth rely on multiple processors or cores. The new industry buzzword "*multicore*" captures the plan of doubling the number of standard cores per die with every semiconductor process generation starting with a single processor. Multicore will obviously help multiprogrammed workloads, which contain a mix of independent sequential tasks, but how will individual tasks become faster? Switching from sequential to modestly parallel computing will make programming much more difficult without rewarding this greater effort with a dramatic improvement in power-performance. Hence, multicore is unlikely to be the ideal answer.

II. MOTIVATION

The promise of parallelism has fascinated researchers for at least three decades. In the past, parallel computing efforts have shown promise and gathered investment, but in the end, uniprocessor computing always prevailed. Nevertheless, we argue general-purpose computing is taking an irreversible step toward parallel architectures. What's different this time? This shift toward increasing parallelism is not a triumphant stride forward based on breakthroughs in novel software and architectures for parallelism; instead, this plunge into parallelism is actually a retreat from even greater challenges that thwart efficient silicon implementation of traditional uniprocessor architectures.

III. APPLICATIONS AND DWARFS

We decided to mine the parallelism experience of the high-performance computing community to see if there are lessons we can learn for a broader view of parallel computing. The hypothesis is *not* that traditional scientific computing is the future of parallel computing; it is that the body of knowledge created in building programs that run well on massively parallel computers may prove useful in parallelizing future applications. Furthermore, many of the authors from other areas, such as embedded computing, were surprised at how well future applications in their domain mappedclosely to problems in scientific computing.
Our goal is to delineate application requirements in a manner that is not overly specific to individual applications or the optimizations used for certain hardware platforms, so that we can draw broader conclusions about hardware requirements. Our approach, described below, is to define a number of "dwarfs", which each capture a pattern of computation and communication common to a class of important applications.

IV. HARDWARE

4.1 Processors: Small is Beautiful

In the development of many modern technologies, such as steel manufacturing, we can observe that there were prolonged periods during which bigger equated to better. These periods of development are easy to identify: The demonstration of one *tour de force* of engineering is only superseded by an even greater one. Due to diminishing economies of scale or other economic factors, the development of these technologies inevitably hit an inflection point that forever changed the course of development. We believe that the development of general-purpose microprocessors is hitting just such an inflection point.

4.2 Memory Unbound

The DRAM industry has dramatically lowered the price per gigabyte over the decades, to $100 per gigabyte today from $10,000,000 per gigabyte in 1980 [Hennessy and Patterson 2007]. The good news is that if we look inside a DRAM chip, we see many independent, wide memory blocks. [Patterson et al 1997] For example, a 512 Mbit DRAM is composed of hundreds of banks, each thousands of bits wide. Clearly, there is potentially tremendous bandwidth inside a DRAM chip waiting to be tapped, and the memory latency inside a DRAM chip is obviously much better than from separate chips across an interconnect.

4.3 Interconnection networks

At the level of the physical hardware interconnect, multicores have initially employed buses or crossbar switches between the cores and cache banks, but such solutions are not scalable to 1000s of cores. We need on-chip topologies that scale close to linearly with system size to prevent the complexity of the interconnect from dominating cost of manycore systems. Scalable on-chip communication networks will borrow ideas from larger-scale packet-switched networks [Dally and Towles 2001]. Already chip implementations such as the IBM Cell employ multiple ring networks to interconnect the nine processors on the chip and use software-managed memory to communicate between the cores rather than conventional cache-coherency protocols.

4.4 Communication Primitives

Initially, applications are likely to treat multicore and manycore chips simply as conventional symmetric multiprocessors (SMPs). However, chip-scale multiprocessors (CMPs) offer unique capabilities that are fundamentally different from SMPs, and which present significant new opportunities:
□□The inter-core bandwidth on a CMP can be many times greater than is typical for an SMP, to the point where it should cease to be a performance bottleneck.
□□Inter-core latencies are far less than are typical for an SMP system (by at least an order of magnitude).
□□CMPs could offer new lightweight coherency and synchronization primitives that only operate between cores on the same chip. The semantics of these fences are very different from what we are used to on SMPs, and will operate with much lower latency.

V. Programming Models

A *programming model* is a bridge between a system developer's natural model of an application and an implementation of that application on available hardware. A programming model must allow the programmer to balance the competing goals of *productivity* and *implementation efficiency.* Implementation efficiency is always an important goal when parallelizing an application, as programs with limited performance needs can always be run sequentially. We believe that the keys to achieving this balance are two conflicting goals:
□□*Opacity* abstracts the underlying architecture. Abstraction obviates the need for the programmer to learn the architecture's intricate details and increases programmer productivity.
□□*Visibility* makes the key elements of the underlying hardware visible to the programmer. It allows the programmer to realize the performance constraints of an application by exploring design parameters such as thread boundaries, data locality, and the implementation of elements of the application.

VI. Systems Software

In addition to programming models, compilers and operating systems help span the gap between applications and hardware towers. In our view, both of these vital programs have grown so large over the decades that it is hard to do the innovation that may need as we switch to parallelism. Hence, instead of completely re-engineering compilers for parallelism, we recommend relying more on autotuners that search to yield efficient parallel code. Instead of relying on the conventional large, monolithic operating systems, we recommend relying more on virtual machines and system libraries to include only those functions needed by the application.

VII. CONCLUSION

Virtually any change can be justified—new programming languages, new instruction set architectures, new interconnection protocols, and so on—if it can deliver on the goal of making it easy to write programs that execute efficiently on manycore computing systems.

This opportunity inspired a group of us from many backgrounds to spend nearly two years discussing the issues, leading to the seven questions of and the following unconventional perspectives:

Regarding multicore versus manycore: We believe that manycore is the future of computing. Furthermore, it is unwise to presume that multicore architectures and programming models suitable for 2 to 32 processors can incrementally evolve to serve manycore systems of 1000s of processors.

Regarding the application tower: We believe a promising approach is to use 13 Dwarfs as stand-ins for future parallel applications since applications are rapidly changing and because we need to investigate parallel programming models as well as architectures.

Regarding the programming models that bridge the two towers: To improve productivity, programming models must be more human-centric and engage the full range of issues associated with developing a parallel application on manycore hardware.

REFERENCES

[1] [ABAQUS 2006] ABAQUS finite element analysis home page. http://www.hks.com

[2] [Adiletta et al 2002] M. Adiletta, M. Rosenbluth, D. Bernstein, G. Wolrich, and H. Wilkinson, “The Next Generation of the Intel IXP Network Processors,” *Intel Technology Journal*, vol. 6, no. 3, pp. 6–18, Aug. 15, 2002.

[3] [Allen et al 2006] E. Allen, V. Luchango, J.-W. Maessen, S. Ryu, G. Steele, and S. Tobin-Hochstadt, *The Fortress Language Specification*, 2006. Available at http://research.sun.com/projects/plrg/

[4] [Altschul et al 1990] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman, “Basic local alignment search tool,” *Journal Of Molecular Biol*ogy, vol. 215, no. 3, 1990, pp. 403–410.

[5] [Alverson et al 1990] R. Alverson, D. Cllahan, D. Cummings, B. Koblenz, A. Porterfield, and B. Smith, “The Tera Computer System,” in *Proceedings of the 1990 ACM International Conference on Supercomputing (SC’90),* pp. 1–6, Jun. 1990.

[6] [Alverson et al 1999] G.A. Alverson, C.D. Callahan, II, S.H. Kahan, B.D. Koblenz, A. Porterfield, B.J. Smith, “Synchronization Techniques in a Multithreaded Environment,” US patent 6862635.

[7] [Arnold 2005] J. Arnold, “S5: the architecture and development flow of a software configurable processor,” in *Proceedings of the IEEE International Conference on Field-Programmable Technology*, Dec. 2005, pp. 121–128.