Ubiquity, an ACM publication
May 2014

Ubiquity Symposium

The Multicore Transformation

Opening Statement

by Walter Tichy

Editor's Introduction

Chips with multiple processors, called multicore chips, have caused a resurgence of interest in parallel computing. Multicores are now available in servers, PCs, laptops, embedded systems, and mobile devices. Because multiprocessors could be mass-produced for the same cost as uniprocessors, parallel programming is no longer reserved for a small elite of programmers such as operating system developers, database system designers, and supercomputer users. Thanks to multicore chips, everyone's computer is a parallel machine. Parallel computing has become ubiquitous. In this symposium, seven authors examine what it means for computing to enter the parallel age.

Ubiquity Symposium

The Multicore Transformation

Opening Statement

by Walter Tichy

Chips with multiple processors, called multicore chips, have caused a resurgence of interest in parallel computing. Multicores are now available in servers, PCs, laptops, embedded systems, and mobile devices. PCs with a single core are hard to find, while top-of-the-line smartphones are already powered by quadcores and octacores. Parallel computing is not new, but has been confined to a few areas such as high performance computing, databases, and operating systems. Until about 2005, the average computer user simply could not afford a parallel machine, while sequential computers kept getting faster with every chip generation. However, the latter trend stopped at about the same time that the integration of multiple processors, caches, and interconnects onto a single chip became feasible. This integration caused costs to drop so dramatically that truly parallel computers are now available for the price of sequential ones. Parallelism has entered the mainstream.

As a result, we're witnessing a revolution-in-the-making: The parallel computing paradigm is displacing the old, sequential paradigm, relegating it to a special case. As with so many revolutions, this change has come upon computer users rather stealthily. It began in 1999 with SUN's MAJC, the first commercial dual core chip. Now, chips with dozens of full-fledged processor cores are routine, and graphical processing units (GPUs) have literally thousands of processors on a chip.

Computer users have barely noticed a dramatic shift is taking place under the hoods of PCs, phones, and tablet computers. To understand what is going on and what still needs to be done, it is helpful to first take a look at the past. There is a rich history of research and experience with parallel computing. While considered esoteric in the past, this work is now coming to the forefront. It is being used as the basis of an accelerating development in both parallel hardware and software.

A Brief History of Parallel Computing

Many people do not realize it, but parallelism has been part of computing from the start. The Atanasoff-Berry Computer of 1942 already had 30 electronic add-subtract units running in parallel. However, parallelism did not play a major role in the design of other first generation digital computers. In 1945, John von Neumann's report on the EDVAC defined the stored-program, sequential computer. The von Neumann architecture is sequential at the programming level, although the arithmetic and logic circuits may exploit some internal parallelism. In the late 1950s and the 1960s, first steps in parallel operation were taken: operating systems allowed I/O devices such as disks to run in parallel with an application. Advanced timesharing operating systems of that era supported multi-programming: Several applications were loaded in memory, and the CPU switched rapidly between them to give the appearance of parallel operation. In 1962, Burroughs produced a military computer named D825, which was probably the first symmetric multiprocessor. It had up to four identical processors sharing memory. The Burroughs B6500 was a commercial version that followed in 1969. In 1966 Michael Flynn of Stanford University described an architectural taxonomy that bears his name. He distinguished between SIMD (single instruction stream, multiple data stream) and MIMD (multiple instruction stream, multiple data stream) computers. An SIMD machine consists of a single control unit that broadcasts an instruction stream to several processors. These processors simultaneously execute each instruction on different data. The MIMD computer, on the other hand, has multiple instruction streams, meaning that each processor can follow a different program or a different path through the same program. The Burroughs D825 and the B6500 were MIMD computers; the ILLIAC-IV and the Connection Machine (see below) were SIMD.

Important concepts for parallel programming were also developed at that time. In 1962, C. A. Petri at the University of Bonn described Petri Nets, a theoretical model for specifying and analyzing concurrent systems. In 1965, General Electric, MIT, and Bell Labs started work on the famous Multics multiprocessor timesharing system. At the Technische Hogeschool Eindhoven Edsgar Dijkstra organized the THE operating system in 1968 as a set of cooperating sequential processes and provided semaphores for signaling processes and protecting critical regions. The same year, Dijkstra introduced the Dining Philosopher's problem, which became a standard example for concurrency. The Burroughs 6500 provided separate stacks for each process, making all code reentrant (meaning that multiple processors could execute the same code simultaneously while working on different tasks). Also that year, Duane Adams at Stanford introduced dataflow in his doctoral dissertation. IBM began working on VECTRAN, an extension of FORTRAN with array-valued operators. Honeywell delivered the first Multics system with up to eight processors.

The 1970s brought forth a number of parallel machines and important, fundamental concepts for programming them. The CRAY-1 was a supercomputer that provided vector operations, multiple arithmetic units, and pipelining among them. The ILLIAC-IV at University of Illinois was even faster than the Cray-1, with 64 processors running in SIMD mode, but only one machine was ever built. At Carnegie-Mellon University, researchers built a full-fledged MIMD computer, the C.mmp, out of 16 minicomputers connected by a crossbar switch. Jack Dennis and David Misunas at MIT published the first description of a dataflow computer and followed with the dataflow language VAL. Leslie Lamport's paper "Parallel Execution of Do-Loops" laid the foundations for later work on vectorization and shared-memory parallelization. He also invented an algorithm, called Lamport timestamps, for ordering events in distributed computer systems. Carl Hewitt at MIT invented the actors model, which is a theoretical basis for understanding parallel computation. Tony Hoare and Per Brinch Hansen independently introduced the concepts of conditional critical regions; later, Hoare defined monitors, a structured mutual-exclusion mechanism, and Brinch Hansen described remote procedure calls. Hoare also synthesized the Communicating Sequential Processes (CSP) model, the basis for several programming languages. The LINPACK benchmark for linear algebra was established and used to compare 23 different computers, including the CRAY-1.

In the 1980s, parallel computers blossomed. Intel introduced the iPSC in 1985, the same year that nCUBE offered its new parallel computer. Both were MIMD machines with hypercube interconnects. The Connection Machine with a then-astounding 65,536 processors demonstrated that programming a massively parallel SIMD machine in data-parallel mode was actually fairly easy. The machine came with parallel extensions of Lisp and C, called *Lisp and C*; FORTRAN extensions came later. Sequent Computer Corporation delivered shared-memory multiprocessors running Unix with up to 32 processors. In Europe, the Transputer chip was designed to connect with four neighbors. Up to 1024 Transputers were hooked up in a grid, using CSP-based Occam as programming language. Additional parallel machines such as the MasPar, the KSR, and others came and went.

The 1990s saw the rise of the clusters. Instead of building specialized, parallel computers, the idea was to connect inexpensive, off-the-shelf PCs on solve large problems on them. In 1994, T. Sterling and D. Becker at NASA built Beowulf, the first cluster consisting entirely of commodity-grade hardware. Using Internet, Clusters could span large geographic areas. An example application (1999) was SETI@home, which used idle cycles on personal computers around the world. SETI central sent small tasks to hundreds of thousands of PCs and compiled their responses into analyses of extra terrestrial signals. The same principle was used to factor large non-prime numbers such as those used in RSA cryptosystems. Portable message passing libraries PVM and MPI were standardized. The cluster approach turned out to scale so well that the world's fastest computers, documented in the TOP 500 list (www.top500.org), are all clusters. In November 2010, the Chinese Tianhe-1A took first place with 186,368 processor cores, while Oak Ridge's Jaguar with 224,162 cores placed second. Half a year later, Japan took the top spot. 2012 belonged to US supercomputers, while a new Chinese cluster, the Tianhe-2 with over three million cores, beat the competition in 2013. The benchmark used in this race is LINPACK (see above).

In addition to the work on parallel architectures, languages to program them, and compilers to extract parallelism from program code, there was considerable theoretical research to understand the limitations of parallelism. In 1969, Richard Karp and Ray Miller published a famous paper "Parallel Program Schemata" in which they studied a mathematical model of parallel sequences and examined structural constraints that would result in the entire system having desirable properties, such as determinacy, in which the system's output depends only on input values but not on the internal timing of tasks in the system. In 1971, Steve Cook created the theory of NP-completeness while trying to understand when parallelism makes problem solving easier. In 1966, Jack Dennis and Earl Van Horn published "Programming Semantics for Multiprogrammed Computations" in which they laid out a strategy to deal with parallelism, determinacy, and protection in multiprocess operating systems. Nearly all their prescriptions are parts of modern operating system infrastructure. In 1973, Ed Coffman and

Peter Denning published *Operating Systems Theory*, in which they demonstrated a theoretical basis for concurrency control in analyzing and designing large multiprocess systems.

A detailed timeline of parallel computing can be found here.

Enter Multicores

While supercomputer users were preoccupied with the growth of clusters, something else happened. Transistors had shrunk enough to make it possible to place more than one processor on a chip. The first of these chips appeared in the late 1990s when Sun Microsystems announced the MAJC 2500, a single chip that contained two processors. Suddenly a new door was opened: Parallel computers were not longer huge behemoths in air-conditioned centers, but could be built into a PC. The lessons learnt with the MAJC, in particular how multiple threads can hide memory stalls, were the basis for Sun's UltraSparc T1 with eight processors on a single chip, which appeared in 2005. Intel followed with a dual core chip in 2006. Presently, all major chip manufacturers produce multicore chips, and the number of processor cores on these chips has been climbing steadily. Oracle (which bought Sun) offers a chip with 32 processors. Tilera integrates 64 to 100 cores on a single die. But the record holder (as of early 2014) is NVIDIA's GTX 780 Ti with 2880 cores. These are SIMD cores without an instruction counter, so comparison with general-purpose CPUs is perhaps unfair, but because of their power and efficiency, more and more data-intensive applications are being ported to GPUs.

While the above designs replicate the same processor, Intel is producing heterogeneous chips: Sandy Bridge of 2011 contains four general-purpose processors and twelve graphics execution units, doing away with separate graphics boards for all but the most demanding gaming applications. Haswell, the 2013 successor, has eight general-purpose CPUs and up to 40 graphics execution units. With the extra-slim 3D transistor introduced by Intel in 2011, ever more processors can be placed on a chip.

It appears multicore has crossed a critical threshold. When it became possible to place several full-fledged processors, caches, and interconnects on a single chip, a tremendous reduction in cost ensued: Multiprocessors could be mass-produced for the same cost as uniprocessors.

Before crossing this threshold, parallel programming was reserved for a small elite of programmers such as operating system developers, database system designers, and supercomputer users. Parallel computers were affordable only to large companies or wealthy scientific institutions. With multicore chips, this situation changed dramatically: everyone's computer became a parallel machine. Anyone with a mobile phone may actually carry around a multiprocessor. Parallel computing has become ubiquitous.

Two other developments are pushing the transition to parallel computing. First, hardware has reached a thermal limit; it is no longer possible to increase the clock speed of silicon chips significantly, because the chips would simply get too hot and fail. Thus, if applications require more performance, waiting for chips with faster clocks is futile. Performance increases now come from parallelism.

Second, instruction level parallelism has also reached a limit. Instruction level parallelism means to execute multiple instructions from a sequential sequence at once, provided they have no data dependencies among them. However, there is a practical limit to how much parallelism can be extracted from an instruction sequence. It now falls to the programmer to figure out how to divide up a computation into larger parts that can be executed in parallel.

What Does It All Mean?

Although parallelism is as old as computing, it has remained relatively hidden and the methods of parallel programming are not yet widely know. What is different now?

Chip makers have moved to multicore chips because they could not speed up clocks and because instruction level parallelism does not scale. As a result, application programmers can no longer hide from parallelism. They need to deal with concurrency to improve performance and reliability. But the hardware architectures, software methods, and languages for parallel programming are not nearly as mature as their sequential counterparts. Nor is there an accepted machine model to emulate or program to. Many new dimensions of software design need to be considered: methods for thread management and synchronization, parallel design patterns, parallel algorithms, new error classes, testing and verification, performance tuning, heterogeneous systems, parallel programming languages, and more. But parallelism is already ubiquitous and here to stay. There is a major shift in computational thinking under way. This symposium addresses the major challenges that come with the multicore transformation. In this symposium, seven authors examine what it means for computing to enter the parallel age.

About the Author

Walter Tichy (walter.tichy@kit.edu) is professor of software engineering at Karlsruhe Institute of Technology (formerly University of Karlsruhe) and a director of the Forschungszentrum Informatik, a technology transfer institute. He is both a Distinguished Scientist and a Fellow of the ACM, and an associate editor of ACM Ubiquity and IEEE Transactions on Software Engineering. He earned M.S. and Ph.D. degrees from Carnegie Mellon University. He received the Intel Award for the Advancement of Parallel Computing, the ACM Sigsoft Impact Paper Award, and the IEEE Most Influential Paper Award, among others.

DOI: 10.1145/2618393