The Role of Multicore Processors in the Evolution of General-Purpose Computing

John McCalpin
Chuck Moore
Phil Hester
Advanced Micro Devices, Inc.

Introduction

The title of this issue, "The Promise and Perils of the Coming Multicore Revolution and Its Impact", can be seen as a request for an interpretation of what multicore processors mean now, or as an invitation to try to understand the role of multicore processors from a variety of different directions. We choose the latter approach, arguing that, over time, multicore processors will play several key roles in an ongoing, but ultimately fundamental, transformation of the computing industry.

This article is organized as an extended discussion of the meaning of the terms "balance" and "optimization" in the context of microprocessor design. After an initial review of terminology (Section 1), the remainder of the article is arranged chronologically, showing the increasing complexity of "balance" and "optimization" over time. Section 2 is a retrospective, "why did we start making multiprocessors?" Section 3 discusses ongoing developments in system and software design related to multicore processors. Section 4 extrapolates current trends to forecast future complexity, and the article concludes with some speculations on the future potential of a maturing multicore processor technology.[1]

[1] Colwell, B. "The Art of the Possible", *Computer*, vol. 37, no. 8, August 2004, pp.8-10.

1. Background

Key Concepts

Before attempting to describe the evolution of multicore computing, it is important to develop a shared understanding of nomenclature and goals. Surprisingly, many commonly used words in this area are used with quite different meanings across the spectrum of producers, purchasers, and users of computing products. Investigation of these different meanings sheds a great deal of light on the complexities and subtleties of the important trade-offs that are so fundamental to the engineering design process.

Balance

Webster's online dictionary lists twelve definitions for "balance."[2] The two that are most relevant here are:

[2] Merriam-Webster Online: http://www.m-w.com/dictionary/balance

- **5 a :** stability produced by even distribution of weight on each side of the vertical axis **b :** equipoise between contrasting, opposing, or interacting elements **c :** equality between the totals of the two sides of an account
- **6 a :** an aesthetically pleasing integration of elements

Both of these conceptualizations appear to play a role in people's thinking about "balance" in computer systems. The former is quantitative and analytical, but attempts to apply this definition quickly run into complications. What attributes of a computer system should display an "even distribution"? Depending on the context, the attributes of importance might include cost, price, power consumption, physical size, reliability, or any of a host of (relatively) independent performance attributes. There is no obvious "right answer" in choosing which of these attributes to "balance" or how to combine multiple comparisons into a single "balanced" metric.

The aesthetic definition of "balance" strikes an emotional chord when considering the multidimensional design/configuration space of computer systems, but dropping the quantitative element eliminates the utility of the entire "balance" concept when applied to the engineering and business space.

The failure of these standard definitions points to a degree of subtlety that cannot be ignored. We will happily continue to use the word "balance" in this article, but with the reminder that the word only makes quantitative sense in the context of a clearly defined quantitative definition of the relevant metrics, including how they are composed into a single optimization problem.

Optimization

Webster's online dictionary lists only a single definition for "optimization":[3]

- **:** an act, process, or methodology of making something (as a design, system, or decision) as fully perfect, functional, or effective as possible; *specifically* : the mathematical procedures (as finding the maximum of a function) involved in this

[3] Merriam-Webster Online: http://www.m-w.com/dictionary/optimization

For this term, the most common usage error among computer users draws from the misleadingly named "optimizers" associated with compilers. As a consequence of this association, the word "optimization" is often used as a synonym for "improvement." While the terms are similar, optimization (in its mathematical sense) refers to minimizing or maximizing a particular objective function in the presence of constraints (perhaps only those provided by the objective function itself). In contrast, "improvement" means to "make something better" and does not convey the sense of trade-off that is fundamental to the concept of an "optimization" problem. Optimization of a computer system design means choosing parameters that minimize or maximize a particular objective function, while typically delivering a "sub-optimal" solution for other objective functions. Many parameters act in mutually opposing directions for commonly used objective functions, e.g., low cost vs high performance, low power vs high performance, etc. Whenever there are design parameters that act in opposing directions, the "optimum" design point will depend on the specific quantitative formulation of the objective function – just exactly how much is that extra 10% in performance (or 10% reduction in power consumption, etc.) worth in terms of the other design goals?

Methodology

For the purposes of this article, the performance impacts of various configuration options will be estimated using an analytical model with coefficients "tuned" to provide the best fit for a large subset of the SPECfp2000 and SPECfp_rate2000 benchmarks collected in March 2006.[4] The analysis included 508 SPECfp2000 results and 730 SPECfp_rate2000 results. The remaining 233 results were excluded from the analysis because either advanced compiler optimizations or unusual hardware configurations made the results inappropriate for comparison with the bulk of the results.

[4] All data is freely available at: http://www.spec.org/cpu2000/

The performance model has been described previously[5] [6] but has been extended here to include a much more complete set of data and has been applied to each of the 14 SPECfp2000 benchmarks as well as to the geometric mean values. Although the model does not capture some of the details of the performance characteristics of these benchmarks, using a least-squares fit to a large number of results provides a large reduction in the random "noise" associated with the individual results and provides a significant degree of platform-independence.

[5] McCalpin, J. "Composite Metrics for System Throughput in HPC", unpublished work presented at SuperComputing2003, http://www.cs.virginia.edu/~mccalpin/SimpleCompositeMetrics2003-12-08.pdf

[6] McCalpin, J. "Composite Metrics for System Throughput in HPC", presented at the HPC Challenge Benchmark panel discussion at SuperComputing2003, http://www.cs.virginia.edu/~mccalpin/CompositeMetricsPanel2003-11-20b.pdf

In brief, the model assumes that the execution time of each benchmark is the sum of a "CPU time" and a "Memory Time," where the amount of "work" done by the memory subsystem is a simple function of the cache size – linearly decreasing from a maximum value with no cache to a minimum value with a "large" cache (where "large" is also a parameter of the model), and a constant amount of memory work for caches larger than the large size. The rate at which CPU work is completed is assumed to be proportional to the peak floating-point performance of the chip for 64-bit IEEE arithmetic, while the rate at which memory work is completed is assumed to be proportional to the performance of the system on the 171.swim (base) benchmark. Previous studies have shown a strong correlation between the performance of the 171.swim benchmark and direct measurement of sustained memory performance using the STREAM benchmark.[7]

[7] McCalpin, J. "Using Timings for the SWIM Benchmark from the SPEC CPU Suites to Estimate Sustainable Memory Bandwidth", revised to 2002-12-02, http://www.cs.virginia.edu/stream/SWIM-BW.html

The model results are strongly correlated with the measured results, with 75% of the measurements falling within 15% of the projection. This suggests that the underlying model assumptions are reasonably consistent with the actual performance characteristics of these systems on these benchmarks. Although there are some indications of systematic errors in the model, not all of the differences between the model and the observations are due to oversimplification of the hardware assumptions – much of the variance also appears to be due to differences in compilers, compiler options, operating systems, and benchmark configurations. Overall, the model seems appropriately robust to use as a basis for illustrations of performance and price/performance sensitivities in microprocessor-based systems.

Assumptions and Modeling

For the performance and price/performance analysis, we will assume

- The bare, two-socket system (with disks, memory, and network interfaces, but without CPUs) costs $1,500.
- The base CPU configuration is a single-core processor at 2.4 GHz with a 1 MB L2 cache, costing $300.
- The die is assumed to be about ½ CPU core and about ½ L2 cache, with the other on-die functionality is limited to a small fraction of the total area.
- The "smaller chip" configuration is a single-core processor at 2.8 GHz with a 1 MB L2 cache, costing $150.
- The "lots of cache" configuration is a single-core processor at 2.8 GHz with a 3 MB L2 cache, costing $300.
- The "more cores" configuration is a dual-core processor at 2.0 GHz with 1 MB L2 cache per core, costing $300.

2. Initial Development of Multicore Chips

The initial development of multicore processors owed much to the continued shrinking of CMOS lithography. It has long been known that increasing the size/width of a CPU core quickly reaches the realm of diminishing returns.[8] So as the size of a state-of-the-art core shrinks to a small fraction of an economically viable die size, the options are:

[8] Farkas, K. I., Jouppi, N. P., Chow, P.. "Register file design considerations in dynamically scheduled processors", 1996. *Proceedings: Second International Symposium on High-Performance Computer Architecture*, pp.40-51, 3-7 Feb 1996.

- Produce a smaller chip
- Add lots of cache
- Add more cores

The option of adding more memory bandwidth typically adds significant cost outside of the processor chip, including revised motherboards (perhaps requiring more layers), additional DIMMs, etc. Because of these additional costs and the burden of socket incompatibility, the option of adding more memory bandwidth will be considered separately from options that involve only processor die size.

Figures 1 and 2 show some of the relative performance and performance/price metrics for these three options, assuming a 30% lithography shrink (i.e., 50% area shrink) which also allows a 17% frequency increase for the single core, but requires a 17% frequency decrease for the dual-core (to remain in the same power envelope). Note that the SPECfp_rate2000 benchmark consists of 14 individual tests that are modeled independently. These are summarized as a *minimum speedup*, *median speedup*, *geometric mean speedup*, and *maximum speedup*. For the dual-core processor option, both the single-core (uni) and dual-core (mp) speedups are estimated.

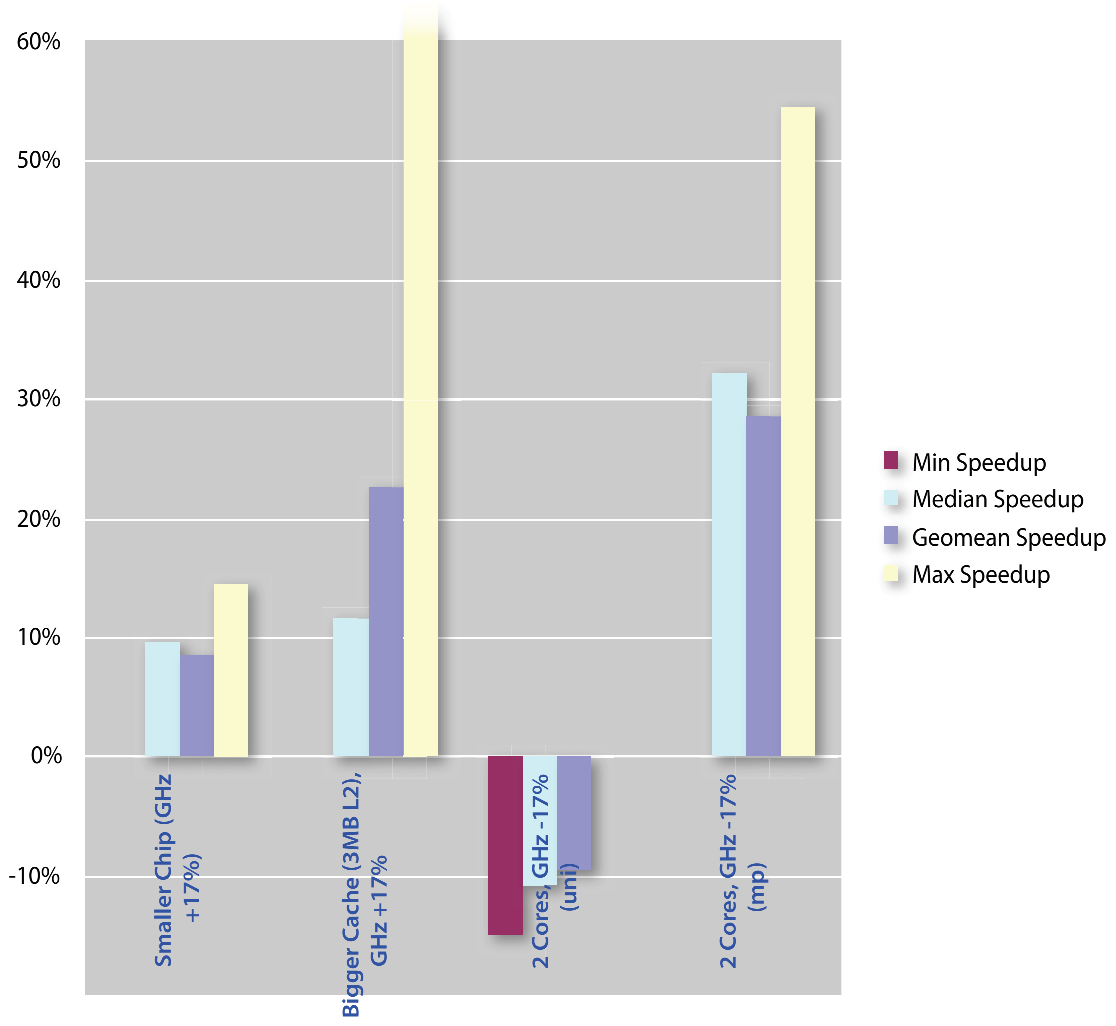

Figure 1: Estimated Relative SPECfp_rate2000 performance for three system configurations (based on the analytical model described in Section 1), assuming a 30% lithography shrink (50% area reduction) with a corresponding 17% frequency boost for the single-core chip and a 17% frequency reduction for the dual-core chip. The *Smaller Chip* is ½ the size of the reference chip, while the *Bigger Cache* and *2 Core* versions are scaled to be the same size and have the same power requirements as the reference chip. Note that the Max Speedup for the *Bigger Cache* case is +156%, but is truncated by the scale here.

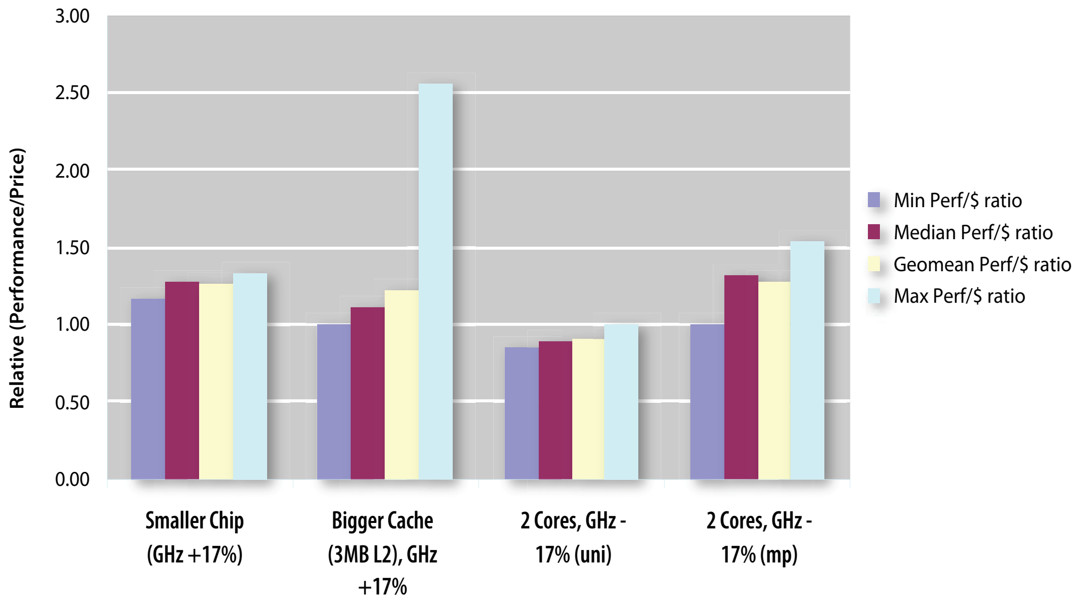

Figure 2: Estimated Relative SPECfp_rate2000/price for three system configurations (based on the analytical model described in Section 1), assuming a 30% lithography shrink (50% area shrink) with a corresponding 17% frequency boost for the single-core chip and a 17% frequency reduction for the dual-core chip.

Looking at these three options in turn:

Produce Smaller Chips

Shipping smaller/cheaper chips with modest frequency increases is of modest value in improving performance and performance/price for customers. In this example, halving the price of the processor reduces the cost of the overall system by 14% ($1,800 vs $2,100) while the 17% frequency boost provides from 0% to 14% improvement in performance, with median and geometric mean improvements of 8%-9% (Figure 1). The combination of these two factors provides performance/price improvements of 17% to 33% with median and geometric mean performance/price improvements of 27%-28%.

Add Lots of On-Chip Cache

Adding lots of cache provides more variable improvement across workloads than most other options. In this example, tripling the L2 cache from 1 MB to 3 MB provides performance improvements of 0% to 127%, with a median improvement of 0% and an improvement in the geometric mean of 11.8%.

The 17% CPU frequency improvement associated with the cache size improvement provides additional benefits; with the combined performance improvement ranging from 0% to 156%, a median improvement of 11.5%, and an improvement in the geometric mean of 22.5%.

In this case, the assumed cost of the chips is the same as the reference system, so the performance/price shows the same ratios as the raw performance.

Add CPU Cores

Adding cores improves the throughput across a broad range of workloads, at the cost of a modest (17%) reduction in frequency in order to meet power/cooling restrictions. Here we assume that the 50% area reduction allows us to include two CPU cores, each with the same 1 MB L2 cache as the reference chip and at the same cost. When running a single process, the performance varies from 15% slower than the reference platform to 0% slower, with median and geometric mean changes of -10% to -11%.

If we can use the second core to run a second copy of the code, the throughput of the system increases from a minimum of 0% to a maximum of 54%, with median and geometric mean speedups of 29% to 32%.

In this case, the assumed cost of the chips is the same as in the reference system, so the performance/price shows the same ratios as the raw performance.

Discussion

The three examples above provided a disturbingly large number of independent performance and performance/price metrics – 70 relative values. Reducing the 14 performance values per benchmark to three (minimum, geometric mean, maximum) still leaves us with nine performance values and 12 performance/price values (of which nine are identical to the performance values). Combining these into a metric that can be used to make rational design decisions is not a trivial task.

Each of the three options has significant benefits and significant disadvantages:

Design Option	Major Benefits	Major Drawbacks
Small Chip	Price reduction	Weakest best-case improvements
Big Cache	Huge performance boost on a few codes	Weakest median and geometric mean performance/price improvements
Dual-Core	Strongest median and geometric mean throughput improvements	Decreased single-processor performance

It is relatively straightforward to find customers for whom any of the six *Major Benefits* or *Major Drawbacks* constitutes critical decision factors. It is much more complex to determine how to take that information, generalize it, and use it in support of the company's business model.

Engineering decisions must, of course, support the business model of the company making the investments. Even the seemingly simple goal of "making money" becomes quite fuzzy on detailed inspection. Business models can be designed to optimize near-term revenue, near-term profit, long-term revenue, long-term profit, market "buzz" and/or "goodwill", or they can be designed to attain specific goals of market share percentage or to maximize financial pressure on a competitor. Real business models are a complex combination of these goals, and unfortunately for the "purity" of the engineering optimization process, different business goals can change the relative importance of the various performance and performance/price metrics.

Caveat

In each of these scenarios the performance changes depend on the ratio of memory performance to CPU performance on the baseline system. As the amount of available bandwidth increases, the benefits of larger caches decrease and the benefit of having more CPU cores increases. Conversely, relatively low memory bandwidth makes large caches critical but significantly reduces the throughput improvements obtainable with additional CPU cores.

For the more cache-friendly SPECint_rate2000 benchmark, results on the IBM e326 servers running at 2.2 GHz show that doubling the number of cores per chip at the same frequency results in a throughput improvement of 65% to 100% on SPECint_rate2000 (geometric mean improvement = 95%).[9]

[9] Results published at http://www.spec.org/ with IBM e326 2-socket servers running single-core and dual-core parts at 2.2 GHz, all running Suse Linux 9.0 and version 2.1 of the PathScale EKOPath compiler suite.

3. Current and Near-Term Issues

The "Power Problem"

Like performance, power is more multidimensional than one might initially assume. In the context of high-performance microprocessor-based systems, the "power problem" might mean any of:

- Bulk current delivery to the chip through a large number of very tiny pins/pads. (Note that as voltages go down, currents go up and resistive heating in the pins/pads increases even at the same bulk power level.)
- Bulk heat removal to prevent die temperature from exceeding threshold values associated with significantly shortened product lifetime.
- "Hot Spot" power density in small areas of the chip that can cause localized failures. (Note that halving the size of the processor core on the die while increasing frequency to maintain the same bulk power envelope causes a doubling of power density within the core.)
- Bulk power deliver to the facility housing the service – including cost of electricity plus the cost of electrical upgrades.
- Bulk heat removal from the facility housing the servers – including the cost of electricity plus the cost of site cooling system upgrades.
- Bulk heat removal from the processor chips that preheats air flowing over other thermally sensitive components (memory, disk drives, etc.).

So the power problem actually refers to at least a half-dozen inter-related, but distinct, technical and economic issues.

Throughput vs Power/Core vs Number of Cores

Restricting our attention to the issue of delivering increased performance at a constant power envelope, we can discuss the issues associated with using more and more cores to deliver increased throughput.

Depending on exactly what parameters are held constant, power consumption typically rises as the square or cube of the CPU frequency, while performance increases less than linearly with frequency. For workloads that can take advantage of multiple threads, multiple cores provide significantly better throughput per watt. As we saw earlier however, this increased throughput comes at the cost of reduced single-thread performance.

In addition to waiting for lithography to shrink so that we can fit more of our current cores on a chip, the opportunity also exists to create smaller CPU cores that are intrinsically smaller and more energy efficient. This has not been common in the recent past (with the exception of the Sun T1 processor chips) because of the assumption that single-thread performance was too important to sacrifice. We will come back to that issue in the section on *Longer-Term Extrapolations*, but a direct application of the performance model above shows that as long as the power consumption of the CPU cores drops faster than the peak throughput, the *optimum* throughput will always be obtained by an infinite number of infinitesimally fast cores. Obviously, such a system would suffer from infinitesimal single-thread performance, which points to a shortcoming of optimizing to a single figure of merit. In response to this conundrum, one line of reasoning would define a *minimum acceptable single-thread performance* and then optimize the chip to include as many of those cores as possible, subject to area and bulk power constraints.

There are other reasons to limit the number of cores to a modest number. One factor, which is missing in the simple throughput models used here, is communication and synchronization. If one desires to use more than one core in a single parallel/threaded application, some type of communication/synchronization will be required, and it is essentially guaranteed that for a fixed workload, the overhead of that communication/synchronization will be a monotonically increasing function of the number of CPU cores employed in the job.

This leads to a simple modification of Amdahl's Law:

$$T = T_s + \frac{T_p}{N} + T_o \times N$$

Here ***T*** is the total time to solution, T_s is the time required to do serial (non-overlapped) work, T_P is the time required to do all of the parallelizable work, ***N*** is the number of processors applied to the parallel work, and T_o is the overhead per processor associated with the communication & synchronization required by the implementation of the application. This last term representing the growth in overhead as more processors are added is the addition to the traditional Amdahl's Law formulation.

In the standard model (no overhead), the total time to solution is a monotonically decreasing function of ***N***, asymptotically approaching T_s. In the modified formation, it is easy to see that as ***N*** increases there will come a point at which the total time to solution will begin to increase due to the communication overheads. In the simple example above, a completely parallel application (T_s=0) will have an optimal number of processors defined by:

$$N_{opt} = \sqrt{\frac{T_p}{T_o}}$$

So, for example, if T_o is 1% of T_p, then maximum performance will be obtained using 10 processors. This may or may not be an interesting design point, depending on the relative importance of other performance and performance/price metrics.

Market Issues

There are intriguing similarities between multi-core chips of the near future and the RISC SMPs that allowed the development of the RISC server market in the 1990's -- a market responsible for over $240B in hardware revenue in the last 10 years.[10]

Like the RISC SMPs of the mid-1990's, these multi-core processors will provide an easy-to-use, cache-coherent, shared-memory model, now shrunk to a single chip.

[10] All revenue data here is based on an AMD analysis of the IDC 2Q2006 world-wide server tracker report. http://www.idc.com/

In ~1995, the SGI POWER Challenge was the best selling HPC server in the mid-range market – one of us (McCalpin) bought an 8-cpu system that year for about $400,000. CPU frequency in 1996-1997 was 90 MHz (11 ns) with a main memory latency close to 1000 ns, or 90 clock periods. In 2007, a quad-core AMD processor will have a frequency of over 2 GHz (0.5 ns) with a main memory latency of ~55 ns, or 110 clock periods. These are remarkably similar ratios.

Delivering adequate memory bandwidth was a challenge (sorry for the pun) on the RISC SMPs. An 8-cpu SGI POWER Challenge had a peak floating-point performance of 2.88 GFLOPS with a peak memory bandwidth of 1.2 GB/s, about 0.42 Bytes/FLOP. A quad-core AMD processor will launch with peak floating-point performance of about 32 GFLOPS and a peak memory bandwidth of about 12.8 GB/s, also giving about 0.4 Bytes/FLOP.

By 1996, the UNIX server market was generating over $22B in hardware revenue, increasing to almost $33B in 2000. The market has been in decline since then, dropping to about $18B in 2006.

Three factors have combined to lead to this decline:

- Increasing difficulty in maintaining the system balances that initially made the servers successful,
- Inability for larger RISC SMPs to follow the smaller RISC SMPs to lower price per processor, and
- Introduction of even less expensive servers based on the IA32 architecture, accelerating with the introduction of products based on the AMD64 architecture in 2003.

It is interesting to look at these three factors in more detail.

Shifting System Balance

As noted above, the initial RISC SMPs had main memory latency in the range of 100 CPU clocks and bandwidth in the 0.4 Bytes/FLOP range. The latency was largely independent of CPU count, while the bandwidth per processor could be adjusted by configuring different numbers of processors.

There has been a clear systematic correlation between application area and bandwidth per processor, with "cache-friendly" application areas loading up the SMPs with a full load of processors, and "high-bandwidth" areas configuring fewer processors or sticking with uniprocessor systems.

By the year 2000, main memory latencies in RISC SMPs had decreased by a factor of about three, while CPU frequencies had increased by factors of three to six. Bandwidth per processor became more complex as monolithic system buses shifted to a variety of NUMA implementations.

Price Trends

During the second half of the 1990's, server vendors went to great lengths to maintain the desirable system balance properties of their very successful systems from the early to middle 1990's. Although largely successful, this effort had a cost -- a financial cost. The two major contributors to this cost were off-chip SRAM caches and snooping system buses for cache coherency. The large, off-chip SRAM caches were critical for these systems to tolerate the relatively high memory latencies and to reduce the bandwidth demand on the shared address and data buses. When Intel quit using standard, off-chip SRAM caches, the market stalled and price/performance of SRAM failed to follow the downward trend of other electronic components. By the year 2000, a large, off-chip SRAM cache could cost several times what the processor cost.

For small SMPs, however, the reduced sharing of the memory and address bus meant lower latency and higher bandwidth per processor. These, in turn, allowed the use of smaller off-chip SRAM caches. The gap between price/processor of small RISC SMPs and large RISC SMPs widened, and customers increasingly turned to clusters of small SMPs instead of large SMPs.

Killer Micros

By the early 2000's, servers based on commodity, high-volume architectures had come within striking distance of the absolute performance of servers based on proprietary RISC architectures, with the high-volume servers delivering superior price/performance. The trend toward small RISC SMPs made the transition to small commodity SMPs much easier. This trend was given a large boost in 2003 with the introduction of processors based on the AMD64 architecture, providing even better performance and native 64-bit addressing and integer arithmetic. Intel followed with the EM64T architecture, leading to a remarkably non-disruptive transition of the majority of the x86 server business from 32-bit to 64-bit hardware in just a few years.

These trends should not be read as indicating a lack of customer interest in SMPs. They do, however, provide an indication of the price sensitivity of various customers to the capabilities provided by the larger SMP systems. Make the price difference too large, and the market will figure out how to use the cheaper hardware.

Just as the RISC SMP market resulted in the parallelization of a large number of ISV codes (in both enterprise and technical computing), the multicore processor trend is likely to generate the impetus to parallelization of the much larger software base associated with the dramatically lower price points of today's small servers.

Unlike the RISC SMP market of the 1990's, the multicore processors of today do not rely on off-chip SRAM caches and can be configured to avoid expensive chip-to-chip coherence traffic (either through snoop filters or by simply using single-chip servers, e.g., Sun's T1/Niagara). There is no obvious general-purpose competitor overtaking x86 performance from lower price points, except perhaps in the case of mobile/low-power devices.

4. Longer-Term Extrapolations

SMP on a chip

The RISC SMP market of the mid to late 1990's was dominated by 4, 8, and 16-way SMPs. This "sweet spot" in the market provided enough extra CPU power to justify efforts at parallelizing applications, without incurring unacceptable price overheads or providing too many processors for applications to use effectively. Current trends suggest that similar SMP sizes will be available on a single chip within a few years, leading one to speculate in several directions:

- Will multicore chips create a new market for parallel codes in the same way that RISC SMPs created the UNIX server market in the 1990's?
- Will effective exploitation of multicore processor chips require fundamental changes in architecture or programming models, or will growth in parallel applications happen on its own – perhaps supported by more modest architectural enhancements (e.g., transactional memory)?
- Will the increasing number of cores/threads available on a single chip obviate the need for larger SMP systems for the majority of users?

Even at the scale of one socket and two socket systems, the increasing number of cores per chip will likely lead to users running combinations of multithreaded and single-threaded jobs (none using all the CPU cores) – more like the large SMP servers of the last decade than like traditional usage models. Increasing numbers of cores is also likely to lead to widespread adoption of virtualization even in these small systems, with multiple guest operating systems having dedicated cores, but competing for memory space, memory bandwidth, shared caches, and other shared resources.

Design Space Explosion

The simple examples at the beginning of this article demonstrated the complexity of defining appropriate figures of merit for microprocessor chips with fairly limited degrees of freedom (e.g., one or two cores plus small or large cache). Dual-core processors first shipped in volume from 90 nm fabrication processes, with quad-core processors expected in quantity in 2007 based on 65 nm process technology. Going to 45 nm provides the possibility for another doubling (8 cores), 32 nm provides the possibility for another doubling (16 cores), and 22 nm technology providing yet another doubling (32 cores) is considered feasible.

Recent studies[11] have shown that the CMP design space is multidimensional both from the engineering and application performance perspectives. The parameter space associated with all the design possibilities inherent in the flexibility of having so many independent "modules" on chip is huge, but the real problem is that the definition of performance and performance/price metrics grows to have almost as many degrees of freedom. If every application of interest has different optimal design points for single-threaded performance, multi-threaded performance design point, single-threaded performance/price design point, and multi-threaded performance/price, then making design decisions is going to become even more difficult, and the struggle between commodity volumes and a proliferation of independent "optimal" design points will become one of the key challenges facing the industry.

[11] Li, Y., et al. "CMP Design Space Exploration Subject to Physical Constraints", *The Twelfth International Symposium on High-Performance Computer Architecture*, 2006, 11-15 Feb, 2006, pp 17-28. http://www.cs.virginia.edu/papers/hpca06.pdf

Heterogeneous Architectures

It is straightforward to demonstrate that, assuming different resources have different costs, a homogeneous multicore chip cannot be optimal for a heterogeneous workload. Expansion of the design space to include heterogeneous processor cores adds many new degrees of freedom. Parameters that might vary across cores include:

- Base ISA
- ISA extensions
- Cache size(s)
- Frequency
- Issue width
- Out of Order capability

The myriad possibilities created by multiplying the homogeneous multicore design space by this extra set of degrees of freedom are both exciting and daunting.

Of course multicore processors will not be limited to containing CPUs. Given the widespread incorporation of three-dimensional graphics into mobile, client, and workstation systems, integration of graphics processing units (or portions of graphics processing units) onto the processor chip seems a natural evolution, as in AMD's announced "Fusion" initiative. Other types of non-CPU heterogeneous architectures may become logical choices in the future, but today none of the contenders appear to have the critical mass required to justify inclusion in high-volume processor products.

Too Many Cores

While the short-term expectation of 4-8 CPU cores per chip is exciting, the longer-term prospect of 32, 64, 128, 256 cores per chip raises additional challenging issues. During the peak of the RISC SMP market in the late 1990's, large (8p – 64p) systems were expensive and almost always shared. Individual users seldom had to worry about finding enough work to do to keep the CPUs busy. In contrast, the multicore processor chips of the near future will be inexpensive commodity products. An individual will be able to easily afford far more CPU cores than can easily be occupied using traditional "task parallelism" (occupying the CPUs running independent single-threaded jobs). In 2004, for example, a heavily-configured 2-socket server based on AMD or Intel single-core processors typically sold for \$5,000 to \$6,000, allowing a scientist/engineer with a \$50,000 budget to purchase approximately eight servers (16 cores) plus adequate storage and networking equipment. Since the explosion in popularity of such systems beginning around the year 2000, many users have found that these small clusters could be fully occupied running independent serial jobs, or jobs parallelized only within a single server (using OpenMP or explicit threads). Assuming that vendors deliver at approximately the same price points, 16-core processor chips will provide 256 cores for the same budget. Few scientific/engineering users have such large numbers of independent jobs (typically generated by parameter surveys, sensitivity analyses, or statistical ensembles) and do not consider such throughput improvements a substitute for improving single-job performance. Clearly, the prospect of a 128-core chip providing 2048 threads for a \$50,000 budget calls for a fundamental change in the way most users think about how to program and use computers. There is therefore a huge burden on the developers of multicore processors to make it much easier to exploit the multiple cores to accelerate single jobs as well as a huge opportunity for computing users to gain competitive advantage if they are able to exploit this parallelism ahead of their competition.

What about Bandwidth?

The flexibility allowed by multicore implementations, along with the quadratic or cubic reduction of power consumption with frequency, will allow the development of processor chips with dramatically more compute capability than current chips. Sustainable memory bandwidth, on the other hand, is improving at a significantly slower rate. DRAM technology has improved its performance primarily through increased pipelining and this approach is reaching its practical limits. Power consumption associated with driving DRAM commands and data, consuming data from the DRAMs, and transmitting/receiving data and probe/snoop traffic to/from other chips are growing to be non-trivial components of the bulk power usage. As noted above, historical data suggests that systems supporting less than about 0.5 GB/s main memory bandwidth per GFLOP/s of peak floating point capability are significantly less successful in the marketplace. Looking forward to, for example, eight cores with four floating point operations per cycle per core at 3 GHz, it is clear that 100 GFLOPS (peak) chips are not a distant dream but a logical medium-term extrapolation. On the other hand, sustaining 50 GB/s of memory bandwidth per processor chip looks extremely expensive. Even if DDR2/3 DRAM technology offers data transfer rates of 1600 MHz (12.8 GB/s per 64-bit channel), actually sustaining the desired levels of memory bandwidth will require many channels (probably eight channels for 102.4 GB/s peak bandwidth), meaning at least eight DIMMs (which can easily be more memory than desired), and will require on the order of 40 outstanding cache misses to reach 50% utilization. (Assuming 50 ns memory latency, the latency-bandwidth product for 102.4 GB/s is 5120 Bytes, or 80 cache lines at 64 Bytes each. So it will require approximately 40 concurrent cache line transfers to reach the target 50 GB/s sustained bandwidth.)

Summary & Conclusions

The examples and illustrations in this article suggest that we are just beginning to grapple with the many opportunities and challenges associated with multicore processors. Our initial steps have been limited by technology to modest changes in the overall balance of systems, but technology trends make it clear that the flexibility provided by future process technologies will present us with an abundance of opportunities to design microprocessor-based systems with radically different power, performance, and cost characteristics. The tension between maintaining high volumes by selling standardized products and increasing performance, performance/watt and performance/price by creating multiple differentiated products will be a major challenge for the computing industry. Even if we try to maintain a modest number of "fast" cores, process technology will enable us to provide more cores than users are currently capable of using effectively. This will challenge industry, academia, and computer users to work together to develop new methods to enable the use of multiple cores for "typical" applications, exploiting the physical locality of on-chip communications to enable more tightly coupled parallelism than has previously been widely adopted.