To calculate a single number to use to compare the power efficiency of systems, SPECpower uses

$$\text{Overall ssj\_ops/watt} = \frac{\sum \text{ssj\_ops}}{\sum \text{power}}$$

The overall ssj_ops/watt of the three servers is 10,802 for the R730, 11,157 for the R630, and 10,062 for the cluster of 16 R630s. Therefore the single node R630 has the best power-performance. Dividing by the price of the servers, the ssj_ops/watt/$1,000 is 879 for the R730, 899 for the R630, and 789 (per node) for the 16-node cluster of R630s. Thus, after adding power, the single-node R630 is still in first place in performance/price, but now the single-node R730 is significantly more efficient than the 16-node cluster.

1.11 Fallacies and Pitfalls

The purpose of this section, which will be found in every chapter, is to explain some commonly held misbeliefs or misconceptions that you should avoid. We call such misbeliefs *fallacies.* When discussing a fallacy, we try to give a counterexample. We also discuss *pitfalls*—easily made mistakes. Often pitfalls are generalizations of principles that are true in a limited context. The purpose of these sections is to help you avoid making these errors in computers that you design.

Pitfall *All exponential laws must come to an end.*

The first to go was Dennard scaling. Dennard's 1974 observation was that power density was constant as transistors got smaller. If a transistor's linear region shrank by a factor 2, then both the current and voltage were also reduced by a factor of 2, and so the power it used fell by 4. Thus chips could be designed to operate faster and still use less power. Dennard scaling ended 30 years after it was observed, not because transistors didn't continue to get smaller but because integrated circuit dependability limited how far current and voltage could drop. The threshold voltage was driven so low that static power became a significant fraction of overall power.

The next deceleration was hard disk drives. Although there was no law for disks, in the past 30 years the maximum areal density of hard drives—which determines disk capacity—improved by 30%–100% per year. In more recent years, it has been less than 5% per year. Increasing density per drive has come primarily from adding more platters to a hard disk drive.

Next up was the venerable Moore's Law. It's been a while since the number of transistors per chip doubled every one to two years. For example, the DRAM chip introduced in 2014 contained 8B transistors, and we won't have a 16B transistor DRAM chip in mass production until 2019, but Moore's Law predicts a 64B transistor DRAM chip.

Moreover, the actual end of scaling of the planar logic transistor was even predicted to end by 2021. Figure 1.22 shows the predictions of the physical gate length

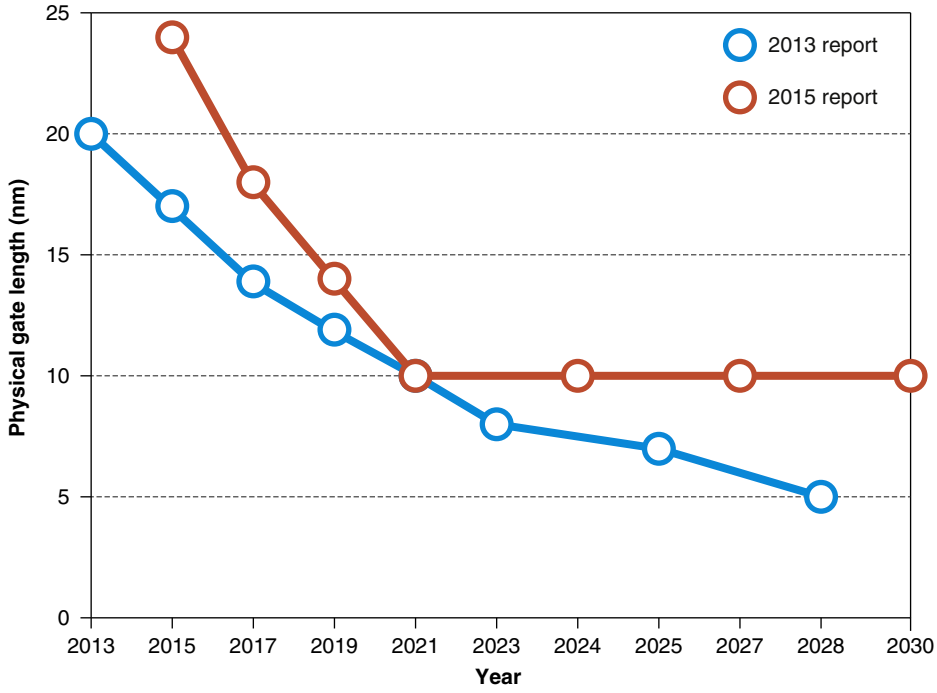

Figure 1.22 Predictions of logic transistor dimensions from two editions of the ITRS report. These reports started in 2001, but 2015 will be the last edition, as the group has disbanded because of waning interest. The only companies that can produce state-of-the-art logic chips today are GlobalFoundaries, Intel, Samsung, and TSMC, whereas there were 19 when the first ITRS report was released. With only four companies left, sharing of plans was too hard to sustain. From IEEE Spectrum, July 2016, "Transistors will stop shrinking in 2021, Moore's Law Roadmap Predicts," by Rachel Courtland.

of the logic transistor from two editions of the International Technology Roadmap for Semiconductors (ITRS). Unlike the 2013 report that projected gate lengths to reach 5 nm by 2028, the 2015 report projects the length stopping at 10 nm by 2021. Density improvements thereafter would have to come from ways other than shrinking the dimensions of transistors. It's not as dire as the ITRS suggests, as companies like Intel and TSMC have plans to shrink to 3 nm gate lengths, but the rate of change is decreasing.

Figure 1.23 shows the changes in increases in bandwidth over time for microprocessors and DRAM—which are affected by the end of Dennard scaling and Moore's Law—as well as for disks. The slowing of technology improvements is apparent in the dropping curves. The continued networking improvement is due to advances in fiber optics and a planned change in pulse amplitude modulation (PAM-4) allowing two-bit encoding so as to transmit information at 400 Gbit/s.

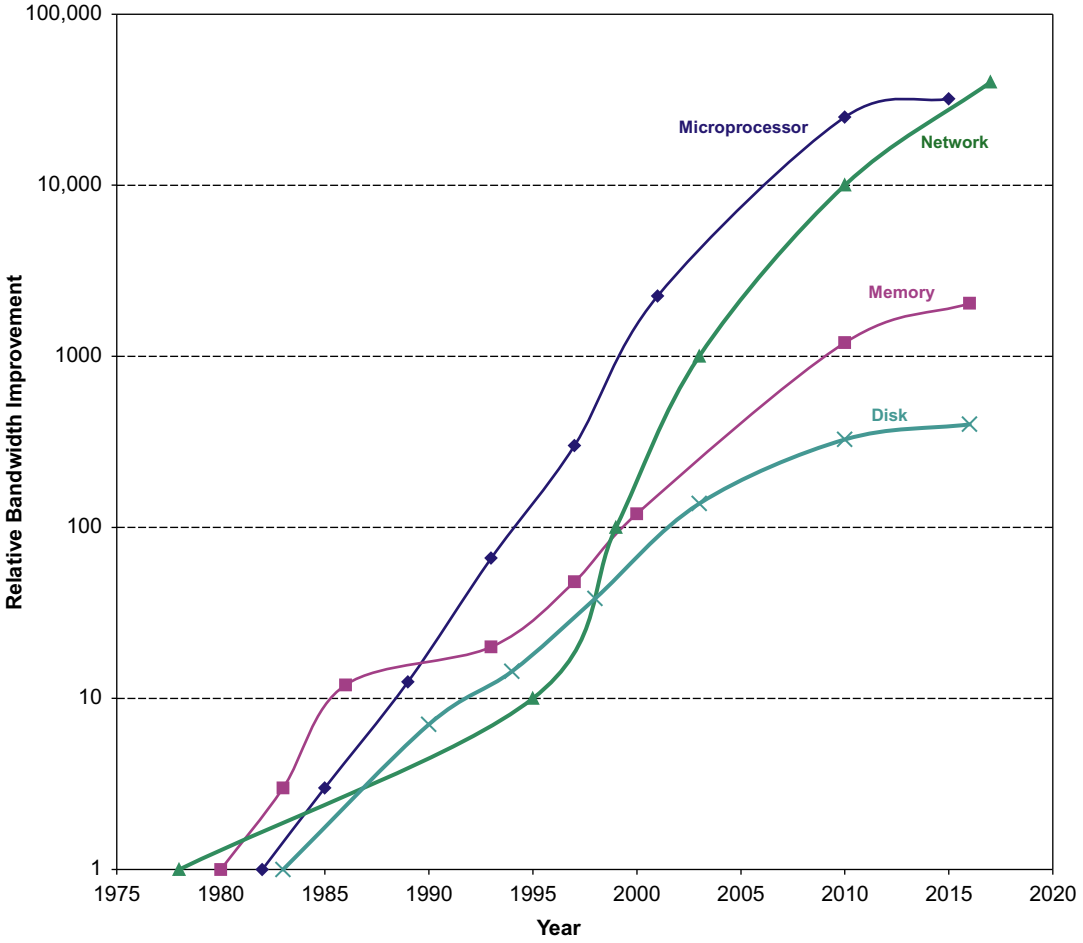

Figure 1.23 Relative bandwidth for microprocessors, networks, memory, and disks over time, based on data in Figure 1.10.

Fallacy *Multiprocessors are a silver bullet.*

The switch to multiple processors per chip around 2005 did not come from some breakthrough that dramatically simplified parallel programming or made it easy to build multicore computers. The change occurred because there was no other option due to the ILP walls and power walls. Multiple processors per chip do not guarantee lower power; it's certainly feasible to design a multicore chip that uses more power. The potential is just that it's possible to continue to improve performance by replacing a high-clock-rate, inefficient core with several lower-clock-rate, efficient cores. As technology to shrink transistors improves, it can shrink both capacitance and the supply voltage a bit so that we can get a modest increase in the

number of cores per generation. For example, for the past few years, Intel has been adding two cores per generation in their higher-end chips.

As we will see in Chapters 4 and 5, performance is now a programmer's burden. The programmers' La-Z-Boy era of relying on a hardware designer to make their programs go faster without lifting a finger is officially over. If programmers want their programs to go faster with each generation, they must make their programs more parallel.

The popular version of Moore's law—increasing performance with each generation of technology—is now up to programmers.

Pitfall *Falling prey to Amdahl's heartbreaking law.*

Virtually every practicing computer architect knows Amdahl's Law. Despite this, we almost all occasionally expend tremendous effort optimizing some feature before we measure its usage. Only when the overall speedup is disappointing do we recall that we should have measured first before we spent so much effort enhancing it!

Pitfall *A single point of failure.*

The calculations of reliability improvement using Amdahl's Law on page 53 show that dependability is no stronger than the weakest link in a chain. No matter how much more dependable we make the power supplies, as we did in our example, the single fan will limit the reliability of the disk subsystem. This Amdahl's Law observation led to a rule of thumb for fault-tolerant systems to make sure that every component was redundant so that no single component failure could bring down the whole system. Chapter 6 shows how a software layer avoids single points of failure inside WSCs.

Fallacy *Hardware enhancements that increase performance also improve energy efficiency, or are at worst energy neutral.*

Esmaeilzadeh et al. (2011) measured SPEC2006 on just one core of a 2.67 GHz Intel Core i7 using Turbo mode (Section 1.5). Performance increased by a factor of 1.07 when the clock rate increased to 2.94 GHz (or a factor of 1.10), but the i7 used a factor of 1.37 more joules and a factor of 1.47 more watt hours!

Fallacy *Benchmarks remain valid indefinitely.*

Several factors influence the usefulness of a benchmark as a predictor of real performance, and some change over time. A big factor influencing the usefulness of a benchmark is its ability to resist "benchmark engineering" or "benchmarketing." Once a benchmark becomes standardized and popular, there is tremendous pressure to improve performance by targeted optimizations or by aggressive interpretation of the rules for running the benchmark. Short kernels or programs that spend their time in a small amount of code are particularly vulnerable.

For example, despite the best intentions, the initial SPEC89 benchmark suite included a small kernel, called matrix300, which consisted of eight different 300×300 matrix multiplications. In this kernel, 99% of the execution time was in a single line (see SPEC, 1989). When an IBM compiler optimized this inner loop

(using a good idea called *blocking*, discussed in Chapters 2 and 4), performance improved by a factor of 9 over a prior version of the compiler! This benchmark tested compiler tuning and was not, of course, a good indication of overall performance, nor of the typical value of this particular optimization.

Figure 1.19 shows that if we ignore history, we may be forced to repeat it. SPEC Cint2006 had not been updated for a decade, giving compiler writers substantial time to hone their optimizers to this suite. Note that the SPEC ratios of all benchmarks but libquantum fall within the range of 16–52 for the AMD computer and from 22 to 78 for Intel. Libquantum runs about 250 times faster on AMD and 7300 times faster on Intel! This "miracle" is a result of optimizations by the Intel compiler that automatically parallelizes the code across 22 cores and optimizes memory by using bit packing, which packs together multiple narrow-range integers to save memory space and thus memory bandwidth. If we drop this benchmark and recalculate the geometric means, AMD SPEC Cint2006 falls from 31.9 to 26.5 and Intel from 63.7 to 41.4. The Intel computer is now about 1.5 times as fast as the AMD computer instead of 2.0 if we include libquantum, which is surely closer to their real relative performances. SPECCPU2017 dropped libquantum.

To illustrate the short lives of benchmarks, Figure 1.17 on page 43 lists the status of all 82 benchmarks from the various SPEC releases; Gcc is the lone survivor from SPEC89. Amazingly, about 70% of all programs from SPEC2000 or earlier were dropped from the next release.

Fallacy *The rated mean time to failure of disks is 1,200,000 hours or almost 140 years, so disks practically never fail.*

The current marketing practices of disk manufacturers can mislead users. How is such an MTTF calculated? Early in the process, manufacturers will put thousands of disks in a room, run them for a few months, and count the number that fail. They compute MTTF as the total number of hours that the disks worked cumulatively divided by the number that failed.

One problem is that this number far exceeds the lifetime of a disk, which is commonly assumed to be five years or 43,800 hours. For this large MTTF to make some sense, disk manufacturers argue that the model corresponds to a user who buys a disk and then keeps replacing the disk every 5 years—the planned lifetime of the disk. The claim is that if many customers (and their great-grandchildren) did this for the next century, on average they would replace a disk 27 times before a failure, or about 140 years.

A more useful measure is the percentage of disks that fail, which is called the *annual failure rate*. Assume 1000 disks with a 1,000,000-hour MTTF and that the disks are used 24 hours a day. If you replaced failed disks with a new one having the same reliability characteristics, the number that would fail in a year (8760 hours) is

$$\text{Failed disks} = \frac{\text{Number of disks} \times \text{Time period}}{\text{MTTF}} = \frac{1000 \text{ disks} \times 8760 \text{ hours/drive}}{1{,}000{,}000 \text{ hours/failure}} = 9$$

Stated alternatively, 0.9% would fail per year, or 4.4% over a 5-year lifetime.

Moreover, those high numbers are quoted assuming limited ranges of temperature and vibration; if they are exceeded, then all bets are off. A survey of disk drives in real environments (Gray and van Ingen, 2005) found that 3%–7% of drives failed per year, for an MTTF of about 125,000–300,000 hours. An even larger study found annual disk failure rates of 2%–10% (Pinheiro et al., 2007). Therefore the real-world MTTF is about 2–10 times worse than the manufacturer's MTTF.

Fallacy *Peak performance tracks observed performance.*

The only universally true definition of peak performance is "the performance level a computer is guaranteed not to exceed." Figure 1.24 shows the percentage of peak performance for four programs on four multiprocessors. It varies from 5% to 58%. Since the gap is so large and can vary significantly by benchmark, peak performance is not generally useful in predicting observed performance.

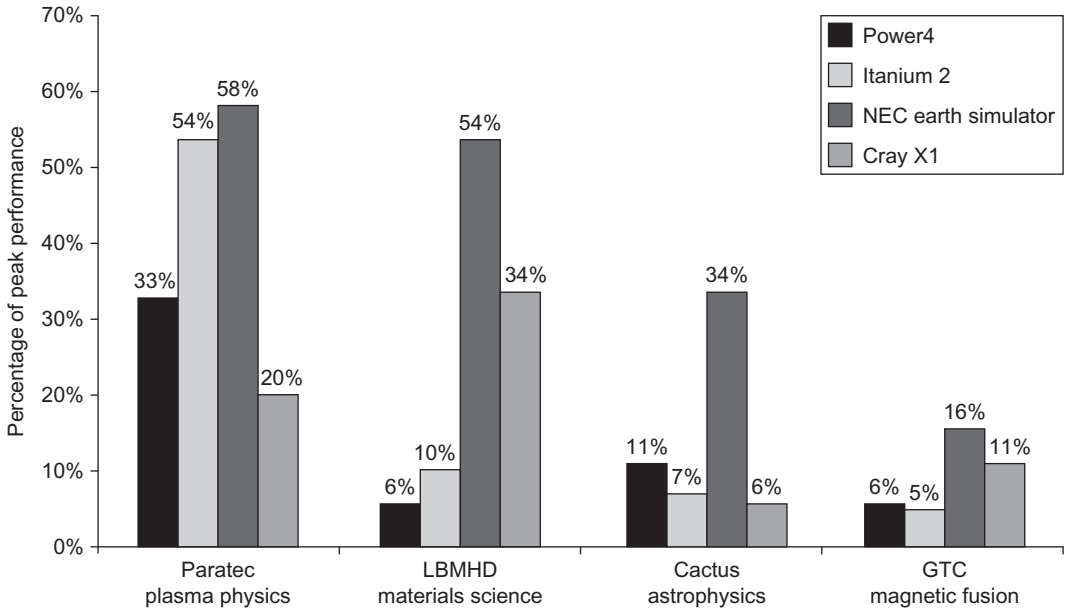

Figure 1.24 Percentage of peak performance for four programs on four multiprocessors scaled to 64 processors. The Earth Simulator and X1 are vector processors (see Chapter 4 and Appendix G). Not only did they deliver a higher fraction of peak performance, but they also had the highest peak performance and the lowest clock rates. Except for the Paratec program, the Power 4 and Itanium 2 systems delivered between 5% and 10% of their peak. From Oliker, L., Canning, A., Carter, J., Shalf, J., Ethier, S., 2004. Scientific computations on modern parallel vector systems. In: Proc. ACM/IEEE Conf. on Supercomputing, November 6–12, 2004, Pittsburgh, Penn., p. 10.

Pitfall *Fault detection can lower availability.*

This apparently ironic pitfall is because computer hardware has a fair amount of state that may not always be critical to proper operation. For example, it is not fatal if an error occurs in a branch predictor, because only performance may suffer.

In processors that try to exploit ILP aggressively, not all the operations are needed for correct execution of the program. Mukherjee et al. (2003) found that less than 30% of the operations were potentially on the critical path for the SPEC2000 benchmarks.

The same observation is true about programs. If a register is "dead" in a program—that is, the program will write the register before it is read again—then errors do not matter. If you were to crash the program upon detection of a transient fault in a dead register, it would lower availability unnecessarily.

The Sun Microsystems Division of Oracle lived this pitfall in 2000 with an L2 cache that included parity, but not error correction, in its Sun E3000 to Sun E10000 systems. The SRAMs they used to build the caches had intermittent faults, which parity detected. If the data in the cache were not modified, the processor would simply reread the data from the cache. Because the designers did not protect the cache with ECC (error-correcting code), the operating system had no choice but to report an error to dirty data and crash the program. Field engineers found no problems on inspection in more than 90% of the cases.

To reduce the frequency of such errors, Sun modified the Solaris operating system to "scrub" the cache by having a process that proactively wrote dirty data to memory. Because the processor chips did not have enough pins to add ECC, the only hardware option for dirty data was to duplicate the external cache, using the copy without the parity error to correct the error.

The pitfall is in detecting faults without providing a mechanism to correct them. These engineers are unlikely to design another computer without ECC on external caches.

1.12 Concluding Remarks

This chapter has introduced a number of concepts and provided a quantitative framework that we will expand on throughout the book. Starting with the last edition, energy efficiency is the constant companion to performance.

In Chapter 2, we start with the all-important area of memory system design. We will examine a wide range of techniques that conspire to make memory look infinitely large while still being as fast as possible. (Appendix B provides introductory material on caches for readers without much experience and background with them.) As in later chapters, we will see that hardware-software cooperation has become a key to high-performance memory systems, just as it has to high-performance pipelines. This chapter also covers virtual machines, an increasingly important technique for protection.

In Chapter 3, we look at ILP, of which pipelining is the simplest and most common form. Exploiting ILP is one of the most important techniques for building