OBJECTIVE ANALYSIS
Semiconductor Market Research

OBJECTIVE ANALYSIS WHITE PAPER

MATCHING FLASH TO THE PROCESSOR

Why Multithreading Requires Parallelized Flash

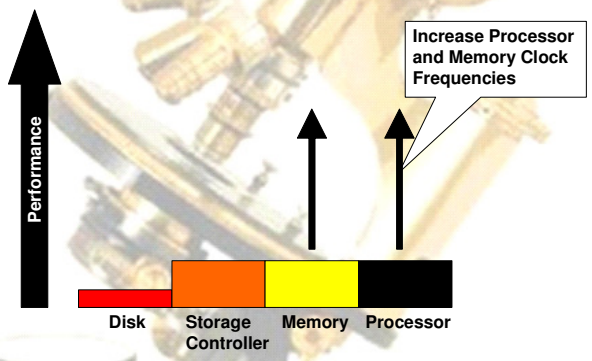
The computing community is at an important juncture: flash memory is now generally accepted as a way to significantly increase performance, but in order to take advantage of all of the throughput that flash offers, system developers must abandon certain classic concepts of storage.

Flash memory is growing in popularity as a new layer in the memory-storage hierarchy. Flash is faster than HDD but slower than DRAM; it is cheaper than DRAM but more expensive than HDD. This is the magic combination that gives flash its appeal.

One significant disadvantage, though, is that the distinction between memory and storage is very clear in conventional computing structures, with storage on one side of the storage controller and memory on the other. To date, flash has been relegated to the storage side of the storage controller, even though it is, in fact, memory.

The problem with getting from here to there is that we are looking at computer architecture from an outdated vantage point, and this makes it difficult to see the clearest way to guarantee peak performance.

Figure 1.) Classic Approach to Performance Increases



Source: Objective Analysis 2014

How Did We Get Here?

In the past microprocessors vied to outdo each other by increasing clock rates. Each generation was faster than the next. This worked because a semiconductor phenomenon called Dennard Scaling made transistor speed scale with size, allowing each new process shrink to naturally result in faster speeds.

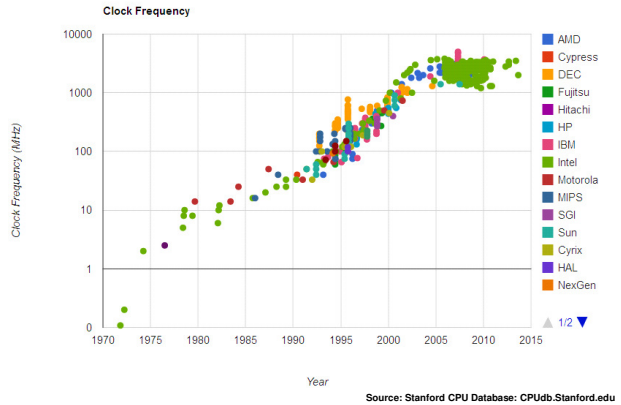
It was relatively easy to balance computer performance in those days. The system designer would choose a processor clock frequency and would match that processor's speed with a DRAM interface that had been designed to fit. Figure 1 is a loose graphic depiction of this approach. To increase performance the clock speeds of both the processor and the memory were raised. Since the boxes in the diagram remain the same height then making both boxes taller by increasing their clock speeds will ensure that the systems' performance remains balanced.

Storage usually received little attention because hard drives (HDDs) were significantly slower than the rest of the system and the storage controller always

provided significantly more bandwidth than even multiple HDDs could use.

In the 1990s it became clear that this approach would become unfeasible within the next few years. A combination of leakage currents and reisitor-capacitor (RC) time constants limited the maximum speed at which a processor could run to roughly 2GHz, after which heat dissipation would become a significant issue. Indeed, processor clock speeds have reached a limit (See Figure 2.)

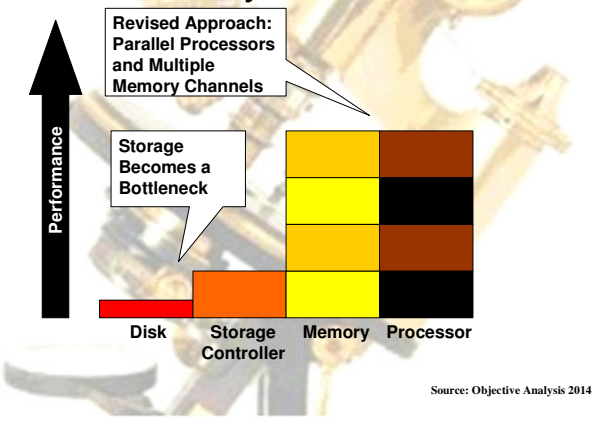
Figure 2.) Processor Clock Speeds



Still, transistors continued to shrink, so more of them could be economically added to the processor chip. The designers' dilemma was to find a way to improve processor performance with a growing number of transistors even if clock speeds had reached a stopping point.

The result was to increase the number of cores in the processor chip – rather than to use a single processor running at its maximum speed, designers harnessed the power of multiple processors running in parallel. This created a new challenge. Most software at that point had been written under the assumption that it would run on a single CPU. Intel and AMD, the two processor leaders, undertook the task of creating a computing environment around multiple processing.

Figure 3.) Using Parallel CPUs & Memory Channels



The two companies worked with compiler companies and other software toolmakers to ensure that these tools could create code to allow multiple concurrent tasks to run on separate CPU cores. Intel and AMD also trained the programmer community to think in terms of "multithreading" – how could they maximize the number of tasks that could be run concurrently? The idea was for software to support as many separate concurrent tasks as possible so that it would run at a speed dictated by the number of available cores. Ideally, a program would run twice as fast on a dual-core processor as it would on a single-core processor, three times as fast on three cores, and so on, until the number of cores exceeded the number of tasks to be performed. Naturally this motivated programmers to explore ways to divide their programs into the largest number of threads that made sense, so that their product would always perform better if more cores became available.

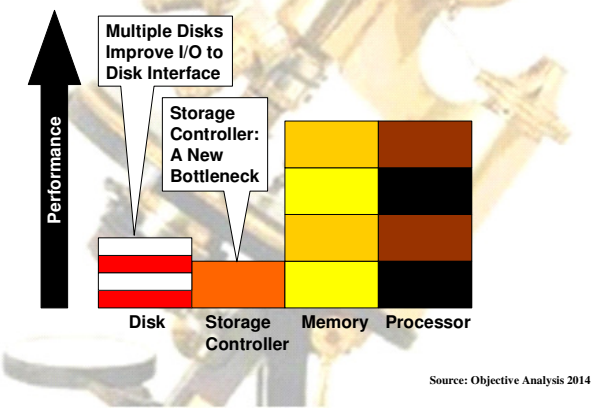
At the same time server designers defined motherboards that amplified this concept, teaming two or four processor chips into dual-socket or quad-socket designs.

Adding DRAM Channels for Bandwidth

Memory had to play a role in feeding this architecture's voracious hunger for data and instructions. Not only did DRAM interfaces migrate to SDRAM, then DDR, DDR2, and DDR3, but multiple memory were added to the board supporting as many as four separate memory buses. This is depicted as multiple memory boxes in Figure 3. A "Four-Socket" (four processor chip) motherboard with four memory channels per socket would then have sixteen memory channels, each pumping out 64 bits (8 bytes) of data at double the 1,066MHz clock rate, amounting to a total memory bandwidth of 270GB/s.

The move to four memory channels not only quadrupled memory/processor bandwidth, but it also allowed processors support memory arrays that were four times as large. Even with these larger memory arrays, today's ballooning datasets expanded beyond the server's maximum memory sizes, forcing the working data to be swapped into and out of HDD, and shifting the focus to the storage subsystem.

Figure 4.) Boosting Storage Speed



A Disconnect with Storage

Disk I/O speeds didn't undergo the same speed ramp. There was really no reason to – because of their mechanical nature HDD bandwidth is relatively limited and cannot be improved. Although spindle speeds did double with the advent of the enterprise HDD, this change significantly increased power consumption, and designers determined that even faster spindle speeds would be impractical simply due to issues of power dissipation. Since HDDs peaked at around 0.4MB/s parallel interfaces to the CPU would have made little difference.

If the disk speed wouldn't improve, there was little reason to improve the speed of the storage controller.

Disk systems were accelerated not by improving the bandwidth between the disk system and the memory-processor complex, but by increasing the data transfer rate between the disk and the storage controller, which already had significantly more bandwidth than was available from HDDs. Common approaches were to use RAID or other striping mechanisms to parallelize the disk array. This is represented by the multiple very small boxes in Figure 4.

When SSDs came along the game changed significantly. Think of it as adding taller boxes to the disk column of Figure 4. These taller boxes, representing the higher performance of the SSD were suddenly hampered by the storage controller, which could not service all of the bandwidth the SSD provided. RAID

cards had to be redesigned to handle all the bandwidth that was suddenly available in multiple-SSD systems.

Disk makers partially solved this problem by routing the SSD through the PCIe bus, which was originally conceived as a way to add coprocessors onto a system. This opened up bandwidth significantly, but still resulted in a bottleneck at the interface between the disk system and the processor-memory complex.

Even with multiple PCIe "lanes" the system still suffers from significantly lower bandwidth than does the memory bus, mainly because there is only one PCIe bus per processor chip. This has prevented the flash from providing its best performance to the processor.

Achieving Parallel Storage with Flash DIMMs

Notice that during this entire discussion nothing has been said about changing the system topology to optimize it for flash. Even when the flash has been moved to the PCIe bus, it is still on the storage side of the graphic, where the storage controller (and storage control software) slows it down.

Users who break with convention and think of flash as a new kind of memory can get past this snag. What if flash could be added to the memory bus? Rather than adding a knot of high-performance storage on the slow side of the storage controller, several smaller blades of fast flash could be added to the system, each on a different memory bus. This would allow parallel access from the processor to the flash at speeds several times that of the storage controller.

This is the way that DRAM bandwidth is made to match the needs of multiple processors, and this is exactly the way that flash should be used by anyone who wants to achieve the highest performance this technology can deliver.

Figure 5 illustrates this approach. The small boxes in this diagram represent flash memory that has been moved (arrows) from the storage side of the system to the RAM side of the system.

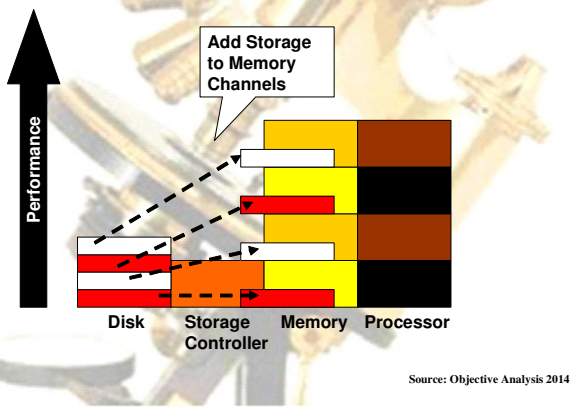
These boxes could all be RAIDed on the HDD side of the storage controller, or they could communicate with the processor via multiple PCIe channels, but the memory-processor interface between flash and the CPU chip is still significantly faster than either of these options.

Breaking Up DIMMs for Bandwidth

Another interesting point is that the designer doesn't need to add all that much flash to each memory bus. As a rule of thumb tiers in the memory hierarchy should be ten times the size of the next-faster tier. In a large server board the maximum memory size is four channels of 32GB each, or 128GB. The rule of thumb would indicate that each channel should have about 320GB of NAND flash.

More realistically, the designer might reduce the sys-

Figure 5.) Getting Past the Slow Disk Interface



tem's DRAM complement when adding flash, partly because they will be giving up a DIMM socket. Let's say the system actually used 16GB of DRAM and around 160GB of NAND flash.

Such a trade-off is well warranted: Objective Analysis has performed benchmarks that show that systems with a small DRAM and flash solidly outperform systems with a large DRAM and no flash, when the dollar spent on flash plus DRAM is the same for both systems.

Spreading smaller amounts of flash among the systems various memory buses should provide the highest throughput. Rather than use a single large 1TB SSD on a PCIe, SAS, or SATA interface, and rather than add a RAID of SSDs that communicate at a high speed through a PCIe-based RAID card, the optimum system would pepper ten times as many 200GB memory-channel storage (MCS) based flash DIMMs throughout the system right on the memory bus to allow the processor's multiple memory channels to gain access to flash memory's ultimate performance benefit. The amount of flash used would remain the same, but the speed at which it can be accessed would increase by more than an order of magnitude. In most cases the flash is added to the system in order to improve throughput at a reasonable cost, so the focus isn't on installing the largest amount of flash, but in achieving the speediest access to an affordable amount of flash.

All-flash DIMMs are available today in the form of SanDisk's ULLtraDIMM or the IBM eXFlash DIMM memory-channel storage. These devices are both available in capacities of 200GB and 400GB and are block accessible, so that they can be treated as storage by existing software, yet accessed at memory speeds through the memory bus.

Summary

Flash memory has brought exciting improvements to computing performance, but it has not been able to perform to its maximum potential because current computer architecture calls for all storage to be placed behind a storage controller.

Meanwhile, other parts of the system have been broken into a number of parallel paths in order to coax increasing performance out of the system without raising clock rates. This is how multicore processors evolved, with each supporting multiple memory channels. Storage alone has failed to keep pace with this change.

Now that flash is accepted as a storage layer designers need to further explore the way that it is put to work. Placing flash behind the storage controller slows it down. Modern flash-based DIMMs allow NAND flash to be added to the highest-speed channel in the system – the memory bus. With the adoption of a bus-based parallel flash system we can expect to see performance increase significantly over the boost already being realized in systems that use SSD-based flash storage.

Jim Handy, March 2014