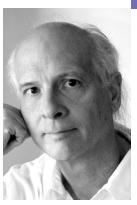
RIK FARROW

musings



Rik is the Editor of *;login:*.

rik@usenix.org

I'VE BEEN HAVING A DIFFICULT TIME keeping my head out of the clouds. Not that I've been flying, or even daydreaming much. It's just that some interesting clouds popped into the foreground recently, and I am finding it hard not to pay attention.

Intel announced its Single-chip Cloud Computer [1] back on December 2, right about the time I was working on my previous column. Unlike Intel's earlier Teraflops project [2], the SCC seemed like something I had once dreamed about, as well as a practical experiment that researchers might actually want to work with.

The Teraflops project was a proof-of-concept: 80 floating-point processors tiled on a chip. While this was cool, it wasn't particularly useful and seemed more like a publicity stunt. But the Teraflops Chip did prove to Intel that it was possible to put many cores on a single chip and have them work.

The SCC also sounded like some PR at first, but that is probably because it has the word "cloud" in its name. It seems as though everything must include "cloud" for marketing purposes, even AV software [3], so ignoring yet another cloud announcement makes perfect sense. One of the OS researchers I contacted about the SCC just blew it off at first for that reason. Yet the SCC represents a likely future design for manycore CPUs.

Distributed Systems

Using a network of processors goes back to the dawn of computing. Even the tube-based IBM 709s had channel I/O processors [4], programmable processors subordinate to the main CPU that handled I/O tasks. Using channel I/O makes a lot of sense, as I/O is slow, and potentially a lot of work could be done if the main CPU wasn't waiting on I/O or, worse, copying data between I/O and memory.

Channel I/O even appeared, briefly, in microprocessor-based systems in the early '80s. Morrow Designs had a hard-disk controller that worked just like a channel controller, complete with the ability to execute programs stored in main memory and copy data between memory and hard drives. At the time, I thought that distributed processing would take off (1984), but Morrow was far ahead of the curve.

Distributed systems got popular with the development of various clusters, starting as early as 1970, and really taking off with the Parallel Virtual

Machine [5] software in the late '80s and Beowulf clusters in the '90s. The ability to use groups of heterogeneous systems as if they were a single supercomputer changed how scientific computing was done. These days, MapReduce and Hadoop are the most used systems for building large-scale clusters, sometimes composed of thousands of systems networked together.

Not Quite a Cloud

Although Intel PR conflates the SCC with cloud computing, that's just abusing the current hot buzzword with their distributed computing design. The SCC consists of 24 dual-core x86 CPUs, each core having its own level 2 cache. The 24 dual-core CPUs each has both memory and hardware dedicated to message passing, with all the CPUs connected in a mesh network. Memory controllers sit at the edges of this network, implying the ability to have four independent memory transfers simultaneously.

Each dual-core CPU, or "tile" in Intel-speak, can run at a different frequency, and groups of four tiles can be run at reduced voltage levels, giving the chip a thermal envelope from 25 to 125 watts. Perhaps this is why Intel styles this chip a "cloud," since, like a cloud, its computing resources can be varied on demand.

The SCC only vaguely resembles today's clusters/clouds, which are composed of networked but complete systems. So each member of a Hadoop cluster, for example, has its own disk, memory, and network. In the SCC, memory, disk, and network get shared among all the cores on the chip.

Even with four memory controllers, the use of the mesh network implies that reading or writing to memory will involve the routers along the path to the proper memory controller. And that suggests to me that a lot of the issues with memory bandwidth contention will still exist in the SCC. OS developers will have to take the latency, based on position in the mesh network and the physical address of memory, into account when they design or modify their operating systems to use the SCC.

But it is the message passing that most intrigues me. Details are vague, but the message itself is not. Current multicore chips have cache-coherent memory, meaning that they also include hardware that keeps track of cache lines that are shared between cores. If data in one core's copy of a cache line changes, then all other copies of that cache line must be invalidated and eventually updated with the current data. The cache-coherency mechanisms share the memory bus, as well as interfering with memory accesses, and this in itself is a limiting factor to how many cores can be used effectively in one chip.

Intel has announced a six-core chip (Gulftown) that still uses cache-coherent hardware, and the SCC has only been released in very limited quantities to researchers. Although the size of these chips is similar, as is the total transistor count—about 1.3 billion—the number of processors and how they maintain memory consistency are very different. I believe the issue here is software, as current AMD and Intel multicore chips are supported by a variety of operating systems.

Intel has demonstrated real systems running Linux on the SCC, so software capable of using these systems does exist. But the SCC takes the concept of multicore into the realm of manycore made with standard cores (Pentium-light CPUs with no out-of-order execution capability) into reality. What are lacking are operating systems and software that can take advantage of the amount of potential parallelism in the SCC.

Multikernels appear best posed as a new model for manycore operating systems. Barrelfish [6] is the best example around today. It not only already relies on message passing instead of cache coherency, but can also run on heterogeneous hardware. Not that the SCC provides this, as it is all x86, but if you imagine a system with intelligent NICs or cores dedicated to simple instruction pipelines (like GPUs), then Barrelfish is well suited to do this.

There are other forms of mildly distributed systems popping up. The Apple iPad uses its own CPU design. Based on an ARM processor, the A4 is a System-on-Chip (SoC), which means it incorporates many of the functions found in separate chips on motherboards in a single chip: the GPU, NIC, I/O bus, and memory controller. SoC designs using the ARM have been around for years, but it will be interesting to see just how well Apple's A4, running at 1GHz, will work in practice. That is, will the A4 be able to render Web pages quickly enough for impatient users, while not sucking dry its battery in a matter of a few hours?

Again, details about the A4 and its host, the iPad, are vague at this time, but iPads should be in the hands of users by the time you read this. Then we will see if the A4 is just another way Apple can lock in control, or if it is really an innovative processor design that saves energy while appearing as zippy as its more energy-intensive relatives, such as the Atom.

Lineup

We lead off this issue with another article about Hadoop. Konstantin Shvachko, one of the developers of the Hadoop File System (HFS), discusses the implications of having a single namespace server and how that might limit performance in very large Hadoop installations. Along the way, you will learn more about how the open source, distributed, cluster, but not cloud, HFS works and what it is capable of in terms of performance.

Next, I had the opportunity of exchanging email with Timothy Roscoe. Roscoe is one of the participants in the development of Barrelfish, the world's first multikernel OS. Mothy was kind enough to correct the many mistaken impressions I retained after reading the SOSP paper several times, and I found myself more enthused than ever about the direction taken by the Barrelfish researchers.

We also have several sysadmin researchers sharing their views about the future of sysadmin. Mark Burgess and Carolyn Rowland discuss the results of past LISA workshops on the Business Directions of IT Management (BDIM). The authors offer advice for sysadmins on how they might better align themselves with business goals and thus become a more integral part of their organization.

Alva Couch takes issue with describing system administrators in terms of the tasks they perform. Instead of tasks, Alva suggests using the notion of social contracts, as sysadmins do more than manage a mail server, for example. Sysadmins have tacitly agreed to provide a reliable mail service to their customers, which is an agreement that goes beyond the mere task of a configuring and maintaining a mail server.

We have two articles on file systems. The first, by William Josephson and his co-authors, is based on their FAST '10 paper about the Direct File System for virtualized flash devices. Josephson explains that key features of current file systems, the buffer cache and block allocation strategies, can actually hinder performance when used with a flash device that handles these features at the device-driver level. This technique places intimate knowledge

about the flash device at a point in the stack where much more is known about the way the device operates. You will also learn more about how current flash drives (solid state drives) work and how flash devices that have interfaces like hard drives compare with the product used in this research.

Jake Wires and Andrew Warfield give us their perspective on file systems. Both Jake and Andy work with the Xen VMM, and this gives them a much different way of looking at how file systems should ideally work. Current VMMs hook into file systems at the block layer, and that obscures a lot of information that would make storage for VMs much more efficient or allow better methods of system updating.

Elizabeth Zwicky provides us with some advice about passwords. Using yet another massive exposure of passwords as her starting point, Elizabeth points out several strategies for the use of passwords, an old technology that just won't go away.

David Blank-Edelman expresses his admiration for regular expressions in Perl. As is usual for David, he provides useful modules that make regular expressions easier to use, something I would not have considered possible until I read his column.

Peter Galvin exposes us to deduplication in ZFS. Peter explains that deduplication is currently not supported by Sun/Oracle, but you can start using it now with the latest OpenSolaris build. Peter also provides examples of what deduplication does and does not do.

Dave Josephsen takes a look at how to get Nagios to scale further. The DNX event broker distributes events to worker nodes, so they can execute plugins and share load with the Nagios server. His second topic is the op5 Merlin module, an event broker that can synchronize events in the database of your choice, as well as perform load balancing and failover of Nagios.

Robert Ferrell examines network protocols that, while interesting to consider, failed for various reasons.

We conclude with book reviews by Elizabeth Zwicky and Brandon Ching.

There are no summaries in this issue, as there were no conferences or workshops over the Christmas holidays, for some reason.

The cloud is more than marketing talk, but also much more specific than marketers would have us believe. What I find much more interesting is how distributed systems, from smart phones and tablets, through manycore chips, right up to massive clusters, appear to be the future of computing.

REFERENCES

[1] Single Chip Cloud (SCC): http://www.theregister.co.uk/2009/12/02/intel_scc/.

[2] Teraflop chip: http://techresearch.intel.com/articles/Tera-Scale/1449.htm.

[3] McAfee Cloud (really SaaS): http://www.mcafee.com/us/enterprise/products/hosted_security/index.html.

[4] Channel I/O: http://en.wikipedia.org/wiki/Channel_I/O.

[5] Parallel Virtual Machine: http://www.csm.ornl.gov/pvm/pvm_home.html.

[6] Barrelfish: http://www.barrelfish.org/.

[7] Apple's A4: http://www.pcworld.com/businesscenter/article/188146/apple_inside_the_significance_of_the_ipads_a4_chip.html.