Cloud scaling, Part 2: **Tour high-performance cloud system design advances**

Learn how to leverage co-processing, nonvolatile memory, interconnection, and storage

Sam B. Siewert  
Assistant Professor  
University of Alaska Anchorage  

21 May 2013

Breakthrough device technology requires the system designer to re-think operating and application software design in order to realize the potential benefits of closing the access gap or pushing processing into the I/O path with coprocessors. Explore and consider how the latest memory, compute, and interconnection devices and subsystems can affect your scalable, data-centric, high-performance cloud computing system design. Breakthroughs in device technology can be leveraged for transition between compute-centric and the more balanced data-centric compute architectures.

The author examines storage-class memory and demonstrates how to fill the long-standing performance gap between RAM and spinning disk storage; details the use of I/O bus coprocessors (for processing closer to data); explains how to employ InfiniBand to build low-cost, high performance interconnection networks; and discusses scalable storage for unstructured data.

View more content in this series

Computing systems engineering has historically been dominated by scaling processors and dynamic RAM (DRAM) interfaces to working memory, leaving a huge gap between data-driven and computational algorithms (see Resources). Interest in data-centric computing is growing rapidly, along with novel system design software and hardware devices to support data transformation with large data sets.

The data focus in software is no surprise given applications of interest today, such as video analytics, sensor networks, social networking, computer vision and augmented reality, intelligent transportation, machine-to-machine systems, and big data initiatives like IBM’s Smarter Planet and Smarter Cities.
The current wave of excitement is about collecting, processing, transforming, and mining the big data sets:

- The data focus is leading toward new device-level breakthroughs in nonvolatile memory (storage-class memory, SCM) which brings big data closer to processing.
- At the same time, input/output coprocessors are bringing processing closer to the data.
- Finally, low-latency, high-bandwidth off-the-shelf interconnections like InfiniBand are allowing researchers to quickly build 3D torus and fat-tree clusters that used to be limited to the most exotic and expensive custom high-performance computing (HPC) designs.

Yet, the systems software and even system design often remain influenced by out-of-date bottlenecks and thinking. For example, consider threading and multiprogramming. The whole idea came about because of slow disk drive access; what else can a program do when waiting on data but run another one. Sure, we have redundant array of independent disks (RAID) scaling and NAND flash solid-state disks (SSDs), but as noted by IBM Almaden Research, the time scale differences of the access time gap are massive in human terms.

The access time gap between a CPU, RAM, and storage can be measured in terms of typical performance for each device, but perhaps the gap is more readily understood when put into human terms (as IBM Almaden has done for illustrative purposes).

If a typical CPU operation is similar to what a human can do in seconds, then RAM access at 100 times more latency is much like taking a few minutes to access information. However, by the same comparison, disk access at 100,000 times more latency compared to RAM is on the order of months (100 days). (See Figure 1.)

**Figure 1. The data access gap**

![Figure 1. The data access gap](image)

Many experienced computer engineers have not really thought hard about the 100 to 200 random I/O operations per second (IOPS) — it is the mechanical boundary for a disk drive. (Sure, sequential access is as high as hundreds of megabytes per second, but random access remains what it was more than 50 years ago, with the same 15K RPM seek and rotate access latency.)

Finally, as Almaden notes, tape is therefore glacially slow. So, why do we bother? For the capacity, of course. But how can we get processing to the data or data to the processing more efficiently?
Look again at Figure 1. Improvements to NAND flash memory for use in mobile devices and more recently SSD has helped to close the gap; however, it is widely believed that NAND flash device technology will be pushed to its limits fairly quickly, as noted by numerous system researchers (see Resources). The transistor floating gate technology used is already at scaling limits and pushing it farther is leading to lower reliability, so although it has been a stop-gap for data-centric computing, it is likely not the solution.

Instead, several new nonvolatile RAM (NVRAM) device technologies are likely solutions, including:

- **Phase change RAM (PCRAM):** This memory uses a heating element to turn a class of materials known as chalcogenides into either a crystallized or amorphous glass state, thereby storing two states that can be programmed and read, with state retained even when no power is applied. PCRAM appears to show the most promise in the near term for M-type synchronous nonvolatile memory (NVM).

- **Resistive RAM (RRAM):** Most often described as a circuit that is unlike a capacitor, inductor, or resistor, RRAM provides a unique relationship between current and voltage unlike other well-known devices that store charge or magnetic energy or provide linear resistance to current flow. Materials with properties called memristors have been tested for many decades but engineers usually avoid them because of their nonlinear properties and the lack of application for them. IEEE fellow Leon Chua describes them in "Memristor: The Missing Circuit Element." A memristor's behavior can be summarized as follows: Current flow in one direction causes electrical resistance to increase and in the opposite direction resistance decreases, but the memristor retains the last resistance it had when flow is re-started. As such, it can store a nonvolatile state, be programmed, and the state read. For details and even some controversy on what is and is not a memristor, see Resources.

- **Spin transfer torque RAM (STT-RAM):** A current passed through a magnetic layer can produce a spin-polarized current that, when directed into a magnetic layer, can change its orientation via angular momentum. This behavior can be used to excite oscillations and flip the orientation of nanometer-scale magnetic devices. The main drawback is the high current needed to flip the orientation.

Consult the many excellent entries in Resources for more in-depth information on each device technology.

From a systems perspective, as these devices evolve, where they can be used and how well each might fill the access gap depends on the device's:

- Cost
- Scalability (device integration size must be smaller than a transistor to beat flash; less than 20 nanometers)
- Latency to program and read
- Device reliability
- Perhaps most importantly, durability (how often it can be programmed and erased before it becomes unreliable).

Based on these device performance considerations, IBM has divided SCM into two main classes:
• **S-type**: Asynchronous access via an I/O controller. Threading or multiprogramming is used to hide the I/O latency to the device.

• **M-type**: Synchronous access via a memory controller. Think about this as wait-states for RAM access in which a CPU core stalls.

Further, NAND SSD would be considered fast storage, accessed via a block-oriented storage controller (much higher I/O rates but similar bandwidth to a spinning disk drive).

It may seem like the elimination of asynchronous I/O for data processing (except, of course, for archive access or cluster scaling) might be a cure-all for data-centric processing. In some sense it is, but systems designers and software developers will have to change habits. The need for I/O latency hiding will largely go away on each node in a system, but it won't go away completely. Clusters built from InfiniBand deal with node-to-node data-transfer latency with Message Passing Interface or MapReduce schemes and enjoy similar performance to this envisioned SCM node except when booting or when node data exceeds node working RAM size.

So, for scaling purposes, cluster interconnection and I/O latency hiding among nodes in the cluster is still required.

**Moving processing closer to data with coprocessors**

Faster access to big data is ideal and looks promising, but some applications will always benefit from the alternative approach of moving processing closer to data interfaces. Many examples exist, such as graphics (graphics processing units, GPUs), network processors, protocol-offload engines like the TCP/IP Offload Engine, RAID on chip, encryption coprocessors, and more recently, the idea of computer vision coprocessors. My research involves computer vision and graphics coprocessors, both at scale in clusters and embedded. I am working on what I call a computer vision processing unit, comparing several coprocessors that became more widely pursued with the 2012 announcement of OpenVX by Khronos (see Resources).

In the embedded world, such a method might be described as an intelligent sensor or smart camera, methods in which preliminary processing of raw data is provided by the sensor interface and an embedded logic device or microprocessor, perhaps even a multicore system on a chip (SoC).

In the scalable world, this most often involves use of a coprocessor bus or channel adapter (like PCI Express, PCIe, and Ethernet or InfiniBand); it provides data processing between the data source (network side) and the node I/O controller (host side).

Whether processing should be done or is more efficient when done in the I/O path or on a CPU core has always been a topic of hot debate, but based on an existence proof (GPUs and network processors), clearly they can be useful, waxing and waning in popularity based on coprocessor technology compared to processor. So, let's take a quick look at some of the methods:

**Vector processing for single program, multiple data**

Provided today by GPUs, general-purpose GPUs (GP-GPUs), and application processing units (APUs), the idea is that data can be transformed on its way to an output device like a
display or sent to a GP-GPU/APU and transformed on a round trip from the host. "General purpose" implies more sophisticated features like double-precision arithmetic compared to single precision only for graphics-specific processing.

**Many core**

Traditional many-core coprocessor cards (see Resources) are available from various vendors. The idea is to lower cost and power consumption by using simpler, yet numerous cores on the I/O bus, with round-trip offloading of processing to the cards for a more capable but power-hungry and costly full-scale multicore host. Typically, the many-core coprocessor might have an order of magnitude more cores than the host and often includes gigabit or 10G Ethernet and other types of network interfaces.

**I/O bus field-programmable gate arrays (FPGAs)**

FPGA cards, most often used to prototype a new coprocessor in the early stages of development, can perhaps used as a solution for low-volume coprocessors as well.

**Embedded SoCs**

A multicore solution can be used in an I/O device to create an intelligent device like a stereo ranging or time-of-flight camera.

**Interface FPGA/configurable programmable logic devices**

A digital logic state machine can provide buffering and continuous transformation of I/O data, such as digital video encoding.

Let's look at an example based on offload and I/O path. Data transformation has obvious value for applications like the decoding of MPEG4 digital video, consisting of a GPU coprocessor in the path between the player and a display as shown in Figure 2 for the Linux® MPlayer video decoder and presentation acceleration unit (VDPAU) software interface to NVIDIA MPEG decoding on the GPU.
Likewise, any data processing or transformation that can be done in-bound or out-bound from a CPU host may have value, especially if the coprocessor can provide processing at a lower cost with great efficiency or with lower power consumption based on purpose-built processors compared to general-purpose CPUs.

To start to understand a GP-GPU compared to a multicore coprocessor approach, try downloading the two examples of a point spread function to sharpen the edges on an image (threaded transform example) compared with the GPU transform example. Both provide the same 320x240-pixel transformation, but in one case, the Compute Unified Device Architecture (CUDA) c code provided requires a GPU or GP-GPU coprocessor and, in the other case, either a multicore host or a many-core (for example, MICA) coprocessor.

So which is better?

Neither approach is clearly better, mostly because the NVRAM solutions have not yet been made widely available (except as expensive battery-backed DRAM or as S-type SCM from IBM Texas Memory Systems Division) and moving processing into the I/O data path has traditionally involved less friendly programming. Both are changing, though: Coprocessors are adopting higher-level languages like the Open Compute Language (OpenCL) in which code written for multicore hosts runs equally well on Intel MICA or Altera Startix IV/V architectures.
Likewise, all of the major computer systems companies are working feverishly to release SCM products, with PCRAM the most likely to be available first. My advice is to assume that both will be with us for some time and operating systems and applications must be able to deal with both. The memristor, or RRAM, includes a vision that resembles Isaac Asimov's fictional positronic brain in which memory and processing are fully integrated as they are in a human neural system but with metallic materials. The concept of fully integrated NVM and processing is generally referred to as processing in memory (PIM) or neuromorphic processing (see Resources). Scalable NVM integrated processing holds extreme promise for biologically inspired intelligent systems similar to the human visual cortex, for example. Pushing toward the goal of integrated NVM, with PIM from both sides, is probably a good approach, so I plan to keep up with and keep working on systems that employ both methods—coprocessors and NVM. Nature has clearly favored direct, low-level, full integration of PIM at scale for intelligent systems.

Scaling nodes with Infiniband interconnection

System designers always have to consider the trade-off between scaling up each node in a system and scaling out a solution that uses networking or more richly interconnected clustering to scale processing, I/O, and data storage. At some point, scaling the memory, processing, and storage a single node can integrate hits a practical limit in terms of cost, power efficiency, and size. It is also often more convenient from a reliability, availability, and servicing perspective to spread capability over multiple nodes so that if one needs repair or upgrade, others can continue to provide service with load sharing.

Figure 3 shows a typical InfiniBand 3D torus interconnection.

**Figure 3. Example of InfiniBand 4x4x4 3D torus with 1152 nodes (SDSC Gordon)**

In Figure 3, the 4x4x4 shown is for the San Diego Supercomputing Center (SDSC) Gordon supercomputer, as documented by Mellanox, which uses a 36-port InfiniBand switch to connect nodes to each other and to storage I/O.
InfiniBand, Converged Enhanced Ethernet iSCSI (CEE), or Fibre Channel is the most often used scalable storage interface for access to big data. This storage area network (SAN) scaling for RAID arrays is used to host distributed, scalable file systems like Ceph, Lustre, Apache Hadoop, or the IBM General Parallel File System (GPFS). Use of CEE and InfiniBand for storage access using the Open Fabric Alliance SCSI Remote Direct Memory Access (RDMA) Protocol and iSCSI Extensions for RDMA is a natural fit for SAN storage integrated with an InfiniBand cluster. Storage is viewed more as a distributed archive of unstructured data that is searched or mined and loaded into node NVRAM for cluster processing. Higher-level data-centric cluster processing methods like Hadoop MapReduce can also be used to bring code (software) to the data at each node. These topics are big-data-related topics that I describe more in the last part of this four-part series.

The future of data-centric scaling

This article makes an argument for systems design and architecture that move processors closer to data-generating and -consuming devices, as well as simplification of memory hierarchy to include fewer levels, leveraging lower-latency, scalable NVM devices. This defines a data-centric node design that can be further scaled with low-latency off-the-shelf interconnection networks like InfiniBand. The main challenge with data-centric computing is not instructions-per-second or floating-point-operations-per-second only, but rather IOPS and the overall power efficiency of data processing.

In Part 1 of this series, I uncovered methods and tools to build a compute node and small cluster application that can scale with on-demand HPC by leveraging the cloud. In this article I detailed such high-performance system design advances as co-processing, nonvolatile memory, interconnection, and storage.

In Part 3 in this series I provide more in-depth coverage of a specific data-centric computing application — video analytics. Video analytics includes applications such as facial recognition for security and computer forensics, use of cameras for intelligent transportation monitoring, retail and marketing that involves integration of video (for example, visualizing yourself in a suit you’re considering from a web-based catalog), as well as a wide range of computer vision and augmented reality applications that are being invented daily. Although many of these applications involve embedded computer vision, most also require digital video analysis, transformation, and generation in cloud-based scalable servers. Algorithms like Sobel transformation can be run on typical servers, but algorithms like the generalized Hough transform, facial recognition, image registration, and stereo (point cloud) mapping, for example, require the NVM and coprocessor approaches this article discussed for scaling.

In the last part of the series, I deal with big data issues.
## Downloads

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU accelerated image transform</td>
<td>sharpenCUDA.zip</td>
<td>644KB</td>
</tr>
<tr>
<td>Grid threaded comparison</td>
<td>hpc_dm_cloud_grid.zip</td>
<td>1.08MB</td>
</tr>
<tr>
<td>Simple image for transform benchmark</td>
<td>Cactus-320x240-pixel.ppm.zip</td>
<td>206KB</td>
</tr>
</tbody>
</table>
Resources

- IBM Almaden has developed systems theory for storage-class memory to define uses and scaling for new NVM devices like racetrack memory, STT-RAM, and PCRAM, developed by IBM Zurich, as well as methods to integrate and interface new NVRAM devices at the TJ Watson research center with mixed ionic-electronic conduction materials for PCRAM.

- Competitive NVM work on memristor devices, described by Leon Chua as the fourth fundamental circuit device (along with resistors, capacitors, and inductors) by HP and Hynix has been delayed, but Hynix is working with several research partners to define fabrication methods for many variants of NVM.

- There has been some controversy on what exactly constitutes a memristor device, but regardless of controversy, progress is being made on numerous two-terminal NVM devices, including ReRAM or RRAM. Regardless of which device makes it to market first, it's clear that access to NVM much faster than NAND flash will revolutionize data-centric computing.

- The memristor has created excitement about direct integration of processing in an NVM, much like a human neural system. One can imagine more of a neuromorphic cortex, like the Intel Neuromorphic Research Project, and numerous FPGA and embedded architecture research projects such as those featured at the Institute of Neuromorphic Engineering. The elusive aspect of neuromorphic engineering has always been that silicon scaling is nowhere near that of the biological scaling of neurons, but perhaps some of the memristor combined compute-memory concepts can take a step closer.

- In the mean time, NAND flash S-type SCM, such as IBM® FlashSystem™ 710 and FlashSystem 810 with InfiniBand interfaces and FlashSystem 720 and FlashSystem 820 with Fibre Channel interfaces, provide an interim solution for high-IOPS workloads and fill in the hierarchy between DRAM and spinning disk storage with S-type asynchronous SCM solutions.

- If you are new to HPC, you'll find the developerWorks article "High-performance Linux clustering, Part 1: Clustering fundamentals" by Aditya Narayan (September 2005) a great place to start. Likewise, if you have never set up a Linux cluster (Beowulf or OSCAR), the developerWorks article "High-performance Linux clustering, Part 2: Build a working cluster" (October 2005) should help you get going.

- Rather than closing the data access gap, coprocessors in the I/O path to S-type NVM, cameras and sensors, displays, or network interfaces show promise for data-centric computing, including the new Intel MIC Xeon Phi™ coprocessor used with IBM iDataPlex® and the Intel MIC development tools.

- Whether you decide to leverage S-type SCM today, plan for M-type as an early adopter, or use I/O path coprocessors, at some point, a single node may not provide the cloud scaling needed for data-centric applications with high-throughput data processing requirements like video analytics. If this is your scenario, then InfiniBand 3D torus clustering, described on YouTube and used in the SDSC Gordon cluster as well as the Red Sky Supercomputer, is a viable interconnection scaling solution off the shelf.

- In the developerWorks cloud developer resources, discover and share knowledge and experience of application and services developers building their projects for cloud deployment.

- Follow developerWorks on Twitter.
• Watch developerWorks demos ranging from product installation and setup demos for beginners to advanced functionality for experienced developers.
• Competing PCIe coprocessor technology from Cavium with OCTEON and Tilera can also be found to scale up processing in the I/O channel. These many-core coprocessors compete with NVIDIA Fermi GP-GPU and GPU vector coprocessors programmed with CUDA5 or OpenCL 1.2 and AMD APU vector coprocessors that also use OpenCL.
• Specialized coprocessors for computer vision are likely to emerge as a result of the Khronos OpenVX standard, such as that being prototyped for computer vision research by the author using the Altera Startix IV/V PCIe FPGA coprocessors, which can also be programmed with Altera-supported OpenCL or traditional hardware design languages.
• Each node in a scalable cluster is likely to need access to lots of unstructured data, also known as big data, and will likely benefit from design for a scalable file system such as GPFS, Ceph, Hadoop HDFS and MapReduce, pNFS, or Lustre.
• Join a developerWorks community cloud computing group.
• Read all the great cloud blogs on developerWorks.
• Join the developerWorks community, a professional network and unified set of community tools for connecting, sharing, and collaborating.
About the author

Sam B. Siewert

Dr. Sam Siewert is an assistant professor in the Computer Science and Engineering department at the University of Alaska Anchorage. He is also an adjunct assistant professor at the University of Colorado at Boulder and teaches several summer courses in the Electrical, Computer, and Energy Engineering department. As a computer system design engineer, Dr. Siewert has worked in the aerospace, telecommunications, and storage industries since 1988. Ongoing interests as a researcher and consultant include scalable systems, computer and machine vision, hybrid reconfigurable architecture, and operating systems. Related research interests include real-time theory, digital media, and fundamental computer architecture.

© Copyright IBM Corporation 2013
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)