Big Data Interactive Visualization Hits Bottlenecks Everywhere – *Benchmarks needed*
David Nadeau, Ph.D.
San Diego Supercomputer Center / University of California, San Diego

1.  **Introduction**

*Interactive* visualization establishes a firm end-to-end performance requirement to deliver a visual response to each user action in $1/10^{th}$ of a second or less. Meeting this requirement for big data runs into performance limits at every level in CPU and cluster architectures, from CPU caches to networks. Here are just a few examples.

2.  **Memory**

Many visualization algorithms don’t do much math – they mainly move data from place to place for slicing, subsetting, filtering, value conversion, and polygon generation. Their performance isn’t limited so much by CPU speed as the memory architecture, including L1/L2/L3 cache sizes and latency, RAM latency, memory bus bandwidth, the number of cores sharing all of these, and the effectiveness of the CPU’s prefetcher.

Let’s take the prefetcher as an example. To the right, a benchmark sweeps through memory with varying stride. Data size is on the horizontal axis, time is vertical, and lines are colored by stride. The plot’s stair steps show L1, L2, L3, and memory access times as data overflows from one to the next. The lines below the stair steps occur when the CPU’s prefetcher can anticipate the algorithm and bring data to the L1 cache ahead of need. This has a big performance benefit, so designing algorithms to be prefetcher-friendly is worth doing. *While common benchmarks report cache and memory performance, benchmarks are needed showing prefetcher performance.*

The previous benchmark is abstract. Here’s a real-world one. To the right, a benchmark sweeps an axis-aligned slice through a 3D volume (such as a CT or MRI scan). Data size is on the horizontal axis, time is vertical, and the lines are slice orientations. The plot shows that performance for some orientations is 20x worse than others due to cache misses. But for all of them the access pattern is a regular stride that a prefetcher should anticipate. The fact that it does not is important when designing these algorithms. *Benchmarks are needed to reveal these prefetcher limits/quirks.*

The industry’s fascination with multi-core processors is also an issue. More cores doesn’t necessarily give better performance for memory-intensive tasks like visualization. Algorithms running on multiple cores can fight over shared caches and the memory bus, yielding worse performance. Predicting this core-fighting is important when designing multi-threaded code. *Benchmarks are needed to characterize core-fighting.*

Other experiments illustrate bottlenecks throughout the memory architecture for common access patterns in visualization, simulation, etc. Basic memory speed benchmarks are not enough; their simple sequential and random access patterns don’t match many real applications and they don’t test advanced processor features, like the prefetcher and resources shared by multiple cores. *Benchmarks are needed for the full application-visible CPU and memory architecture and a variety of real-world access patterns.*

3.  **Storage**

When data is too large to fit in memory, it can be streamed from disk or SSD. Physical memory becomes a big disk cache and its performance is important.

Let’s look at the OS’s disk block prefetcher. To the right, a benchmark sweeps a slice through a multi-dimensional data set on disk in 120 different orientations. Data size is on the horizontal axis, time is vertical, and the lines are different slice orientations. While all slice orientations use a regular stride, only those that sweep through data sequentially benefit from the OS
prefetcher. For regular strides, but non-sequential accesses, the OS prefetcher fails badly and performance is exponentially worse. Most OSes support application prefetcher hints to overcome some of these problems, but a broader impact solution would be to improve the OS prefetcher. Benchmarks are needed for common access patterns, not just sequential access, and to test OS prefetcher and disk cache characteristics.

Legacy block-based I/O is also problem. When an algorithm needs a full block, reading the full block is efficient. But many slicing, subsetting, and feature detection algorithms inherently use only a small portion of a block as they hop about through the data. Reading a full block wastes time getting unused data. With bigger storage capacities, block sizes are increasing and block use is even more sparse. When OS prefetchers fail, as they often do, this wasted I/O time stalls the application and reduces performance. Benchmarks are needed that show I/O system performance for sparse block use and the impact of block size and block-centric I/O.

Other experiments characterize issues with the disk bus and the OS’s shared disk cache. Like a CPU’s shared memory caches for multiple cores, a shared OS disk cache for multiple processes can cause fighting. Benchmarks for the full application-visible I/O system are needed, including characterizing disk cache fighting.

4. Network
Instead of paging big data from disk on a single host, data can be resident on a cluster of hosts. Algorithms coordinate those hosts to divide and conquer big data. As cluster nodes finish their jobs, data is gathered, assembled, and returned to the user. The cluster network’s performance becomes important.

To the right, a benchmark ping-pongs a message between nodes. Message size is on the horizontal axis, effective bandwidth is vertical, and the colored lines are for different numbers of nodes. The network is QDR InfiniBand. For small messages under a few Kbytes, network latency dominates and the effective bandwidth is very low. To get full QDR bandwidth, messages must be ½ Gbyte or larger. But for visualization algorithms, messages are often small, such as a few dozen bytes sending a 3D viewpoint, or a few Mbytes for an image. The network industry’s focus on increasing bandwidth does little to help this cluster visualization, or others cluster task with messages smaller than ½ Gbyte. For these tasks, network latency is more important than bandwidth. Benchmarks are needed that measure latency and bandwidth for a variety of common network access patterns and message sizes.

The above benchmark was on an unloaded cluster. To the right is the same benchmark on a cluster with CPU and memory intensive jobs. The performance skew on the right appears to come from memory bus fighting as the network interface attempts to read/write memory buffers while other jobs are accessing data in memory. Beyond illustrating that repeatable benchmarks should be run on unloaded systems, the plot shows that resource contention within nodes can strongly affect network performance between nodes.

Also note the high point of both plots. This is InfiniBand QDR with a theoretical bandwidth of 4 Gbytes/sec at the top of the plot. Achieving 85% of this peak is possible on a Westmere cluster, but on this Magny-Cours cluster the best bandwidth is just 70%. The slower memory bus for Magny-Cours appears to be the problem. Benchmarks are needed that measure application-visible network performance, taking into account system architecture and OS issues along with raw network abilities.

5. Conclusions
And so on. For interactive visualization, the response time to the user is fixed even though data is getting bigger and bigger. The performance of all aspects of the OS, CPU, memory, and network architecture is critical. Benchmarks are needed that look at end-to-end performance on full systems, not just isolated hardware components. And benchmarks are needed to look at many common access patterns, not just sequential and random accesses. Big data problems push system limits everywhere, making it much more important that we understand those limits, and then design systems to reduce them.