Considerations for Developing and Deploying Petascale Systems to Ensure Optimum Scheduling of Resources

Scott Jackson and Trev Harmon

Adaptive Computing, Provo, Utah
May 26, 2010

Abstract: As petascale systems establish themselves as the standard for high-end Top 500 machines and as the leadership sites sets their sights on exascale systems, many important considerations in the design and efficient operation of batch systems need to be resolved. In this paper, we will discuss scalability and other issues regarding the scheduling of batch workload and resources these leading sites will face in the future.

1. Background/Introduction

Helpful to a proper understanding of the properties needed in future workload schedulers is an understanding of the trends that have brought us to petascale* computing and the emerging issues that are shaping our path to the next level of computing—exascale† computing.

Through the 1970s and 1980s, the fastest supercomputers were based on vector architectures. Gigascale computing (10^9) was achieved in 1985 with the Cray 2 vector supercomputer. But the vector architecture eventually ran into scaling limits and began to be surpassed by clusters. The first terascale computer (10^{12}) appeared in 1997 and was based on a massively parallel cluster of compute nodes (ASCI Red). Massively parallel architectures using thousands of processors from commodity PCs running Unix dominated supercomputing for the next couple of decades. (Wladawsky-Berger.)

As increases due to clock speed tapered off and clusters of nodes expanded to the tens of thousands, the path from terascale to petascale was driven by the growth of multi-core processors. As a case in point, in 2005 the 26 teraflop/s Cray XT had only one CPU core on each compute node. Over the next three years, the computational power of this system doubled three times as the number of cores doubled to two, then four, and finally eight cores per node, giving the Cray XT system more than a quarter of a petaflop/s capability. In late 2008, a 1.3 petaflop/s Cray XT5 system was installed at ORNL and a year later was upgraded to faster processors to achieve its present 2.3 petaflop/s computational power using 12 cores per node. (Geist.)

Cray's strategy of employing a lightweight Linux operating system that severely reduces system "noise" as well as network chatter was an important key in being able to scale to over 100,000 processors and achieve petascale computing in 2008. IBM's BlueGene used similar strategies to bring Lawrence Livermore systems to be the first to achieve hundreds of teraflop/s. The trend of minimizing system noise and interrupts is likely to continue and result in more lightweight compute node platforms and may also instigate the increased use of virtualization and custom provisioning.

Current trends project that exascale systems will begin to appear by around 2018. Because we are no longer able to increase the performance of a single processing element by turning up the clock rate because of power and cooling issues, we now have to rely solely on increased concurrency. Since our primary path to increasing performance will be through parallelism, we anticipate exascale systems may have upwards of 100 million cores. (Wladawsky-Berger.)

Beginning today with petascale systems such as Jaguar, the roadmap shows that systems with a few tens of petaflop/s will be available in the 2011–2012 timeframe. The first example may be the National Science Foundation's Blue Waters system, which is expected to be delivered to the National Center for Scientific Applications in 2011. By 2015, systems with a few hundred petaflop/s are expected to exist. In the 2018–2020 timeframe, the first exascale system is expected to appear. (Geist.)

Just as multi-core chips were the technology driver that enabled petascale science, research suggests that the emerging heterogeneous many-core processor tech-

* A computer system capable of achieving performance of excess of one quadrillion (10^{15}) floating point operations per second.

† A computer system capable of achieving performance of one quintillion (10^{18}) floating point operations per second.

nology will replace the current multi-core trend in order to reach exaflop/s scaling. Whereas multi-core processors have a few to tens of cores, many-core processors will have hundreds of cores on each chip. These many-core processors place hundreds of specialized cores (such as GPUs) on a chip with a few general-purpose cores as controllers. Such heterogeneous many-core chips have been announced by AMD, Intel, and other chip manufacturers. Heterogeneity in future systems may also be driven by the increasingly common inclusion of accelerators such as Field Programmable Arrays (FPGA), vector accelerators, and special-purpose computers.

Heterogeneous, many-core processors additionally have the potential to overcome a major exascale challenge—power consumption—by providing two orders of magnitude more flop/s per watt than multi-core processors. (Geist.) Even with these innovations, power consumption continues to be a major challenge for exascale. Projections in the Defense Advanced Research Projects Agency's extreme scale study show that the power consumption for an exascale system in 2018 even under optimistic assumptions would be 100–200 MW. The electric bill for that much power would be in excess of $100 million per year and is tantamount to the power produced by a small power plant. (Geist.)

Interconnection network topologies will become of increasing importance. With millions of nodes, nodes will be anywhere from one to many hops from each other. Optimal inter-node connectivity becomes critical for both path redundancy and communication latency. The volume of data traffic will tend to increase significantly in future systems since a job will now be spread out over more nodes and more communication will be relayed over the network. Network chatter will have to be severely curtailed in such systems and efficient communication mechanisms utilized.

Further we have the challenge of resiliency (Bianchini), which becomes significantly more pronounced in large systems since as more nodes are added to the system, each with a relatively constant MTBF (Mean Time Between Failure), the MTBF for the overall system is significantly reduced. Assuming a node MTBF of 106 hours, we would have a system MTBF of just a few hours at best for an exascale system. (Kogge.)

These technology trends have major implications for batch schedulers, requiring major innovations in a wide range of areas. New mechanisms to address scalability will be paramount. An exascale batch system will have to be more autonomic in nature, optimizing and adapting to new conditions and failures. It will have to be fault tolerant and intelligently migrate jobs around failed or failing components. Scheduler support for virtualization may be needed to perform live process migration from failing nodes and may also be used to allow the compute node operating system to be extremely lightweight. The batch system will need to take into account the physical network topology to minimize connection distance and communication latency. Energy-aware scheduling may become more important. Policies and mechanisms for efficiently scheduling over heterogeneous many-core nodes will be essential.

Increased capacity will lead to new usage patterns and will require more adaptable scheduling mechanisms and policies. Support for reservations for failure isolation, rolling maintenances, and political policy partitioning will be key. Emerging cloud technologies will spur the need for dynamic provisioning of virtually every aspect of the environment.

2. Scalability

The increased scale of future high-performance computing systems, in terms of number of nodes, will have effects on job scheduling, job submission and job startup rates, collection of node-state and job-state information from the compute nodes, and client-command performance.

For capability machines[‡], job submission rate will be relatively unchanged since we expect a comparable number of jobs on the next-generation systems. But for capacity machines, where the increase in the number of nodes per job will not keep up with the increase of the number of nodes in the machine, job counts will be significantly higher and the resource manager will have to be able to handle a significantly increased (perhaps on the order of 100) job submission rate. Since the clock rate is not expected to be significantly increased in this timeframe, submissions may have to be handled in a more distributed manner (with more logic in the client, or communicating with one of many batch servers, though independent of the scheduling thread). Another optimization may be for resource managers and schedulers to store their object data in an enterprise database so job submissions, client requests, and resource managers, can perform their functions without impinging on scheduling or other simultaneous scheduler requests.

Start rate will be more of a concern as the launching of processes on an increased number of nodes per job will have to be coordinated. One might naively expect job start times to increase linearly with the number of nodes (~100x), but this is clearly untenable. At pe-

[‡] High-end supercomputers can be classified as being designated for either capability or capacity computing. Capability supercomputers use large portions of the resources to solve very large problems in the shortest amount of time. Capacity supercomputers, on the other hand, support large numbers of simultaneous smaller problems.

tascale and on to exascale, a resource manager will need to use a more sophisticated approach for data dissemination and aggregation than one-to-all communication. It will need to utilize some form of distributed or hierarchical approach. If we assume that we will be using a tree-based hierarchical data-dissemination model, the exascale startup time may go up merely by one third (assuming an n-ary tree of degree 10) from the petascale startup time.

Another serious challenge will be collecting the job and node-state information from the compute nodes where the jobs are running. The volume of data from a million nodes to update the resource manager with the current node and job state would completely overwhelm both the network and the resource manager unless very carefully managed by a number of enhancements. First, instead of sending all information, each node will need to send only "delta" information, that is, send only the values that have changed since the last data transfer. Next, some form of distributed or tiered approach will be needed to avoid directly sending data over the network from all compute nodes to the resource manager. The compute nodes may each update to a distributed database, or they may use a tree-based hierarchical data-aggregation model where information is aggregated in tiers from child sub-trees (in reverse of the hierarchical distribution tree) to funnel the information back in a manner that will not overwhelm the network or the resource manager. Data compression will also become a key factor, especially if using a hierarchical data-aggregation tree.

Scheduling algorithms will also be affected by next-generation realities. Scheduling algorithms are diverse and complex, but as a first-order approximation, one might roughly say that scheduling computation may roughly scale on the order of jobs times nodes. Therefore, the scheduling problem would be expected to be at least 100 times more complex on capability machines (because of the assumed hundredfold increase in nodes from petascale to exascale) and even more so (perhaps a thousandfold) for capacity machines. Some of the better optimizing scheduling routines would suffer a higher-order impact. This poses a challenge since any ordered scheduling (priority, FIFO, etc.) is not an easy thing to make multi-threaded and clock speeds are not likely to increase with system size. Undoubtedly, batch-system vendors will continue to innovate through coding optimizations (and potential use of accelerators) to keep up with the challenge at least through exascale. Some current algorithms, however, will likely have to be discarded for more efficient ones.

Batch-system client-command invocation will undoubtedly see an increase on the capacity machines. However, to the extent that these requests can be engineered to interact directly with an enterprise data repository for their information, such invocation will likely not pose a serious problem.

3. Resiliency

Future systems will have millions of nodes in them. They will have substantially more processing elements, more memory components, and more network links. The system MTBF will be reduced to the point that failures will be continuous rather than exceptional occurrences. Every layer of the batch and application stack will have to be designed to dynamically detect, adapt to, and recover from failures. The typical capability job will encounter component, node, or network failure as a matter of course in its lifetime, so new measures must be instituted to dynamically route around failures that formerly proved to be fatal for the job.

Workload manager systems need to be fault tolerant. Compute nodes will need to automatically register themselves to the resource management system and be deregistered when they fail to check in. It may be useful to establish a hierarchical or peer-to-peer relationship between the resource management subsystems on the compute nodes for communication efficiencies. These connections will need to be formed dynamically, and the connection graph will need to adapt automatically to node or network failures.

Users expect their jobs to run to completion in spite of failures that will become the rule rather than the exception. System or application-level job check-pointing could be one way to approach this issue. That way, if a failure were to occur and the job were to terminate, the job could be restarted on a set of nodes and network links to avoid the problem. However, this approach may not be viable at future scales. Although processing power is keeping up with Moore's law, network bandwidth, disk storage, and memory access are becoming increasingly limiting factors for next-generation systems. Consider the magnitude of trying to write out a full-state checkpoint (including all memory, disk usage, in-flight messages, operating system states, etc.) of a million-node job network-attached disk storage. Next consider the length of time that would be required to complete this against the frequency with which these checkpoints would need to be taken, in the face of system MTBFs measuring in hours or minutes. By the exascale timeframe, checkpoint-restart may cease to be the holy grail of HPC resiliency.

A possible alternative (or supplementary) approach could be job or task migration. Some computing vendors and leadership sites are studying ways to identify failing components before failures become catastrophic. If one were able to detect signs of component failure prior to loss, one could potentially migrate the effected

processes and tasks to other locations during the job run. One of the most promising ways to do this is with virtual machines. Virtual machines today are able to operate very close in performance to the physical machines they run on. Live process-level or job-level migration could be performed as a remedy to failing components in the case where these failures can be predicted. (Wang.)

In fact, virtualization helps address another resiliency issue. We described the trend to have more lightweight operating system images to avoid noise and interrupts. This trend also has a positive effect on resiliency, since the fewer software elements that are running and installed on the system, the less that can go wrong. Through virtualization or other means, a job submitter could dynamically provision the application environment with the entire required software stack by means of a virtual machine image booted on their allocated compute nodes. If this practice became the norm, the compute nodes could become very streamlined indeed. This approach could additionally protect the security of the system software and provide an effective means to allow for bounded sharing of the many cores per node among multiple jobs.

4. Power

Some exascale studies are finding that the single most difficult and pervasive challenge deals with energy use. (Kogge.) This involves not only the power consumed by processing components but also the increasingly expensive energy costs related to data transportation. It is generally agreed that the thousandfold increase in processing power between petascale and exascale computing must be accompanied by not more than a tenfold increase in energy consumption, or the costs for operating these systems will be prohibitively expensive. (Wladawsky-Berger.) One of the technologies that could contribute to lower overall energy costs is energy-aware scheduling.

One of the highest payoffs could be achieved by reducing power consumption for unused or underutilized resources. Intelligent batch systems, such as the Moab Workload Manager, can be used to place idle servers or resources in power-saving modes or even power nodes off completely until needed. Next-generation power-management software will be able to turn off or ratchet down unneeded resources within a node. The workload manager may be able to interact with the power management software on a node to help it identify and power down resources that are not currently needed.

Additionally, workload managers may be able to take into account component power usage and factors that contribute to high power consumption, such as temperature and activity hotspots. Cost savings could be achieved by using an intelligent scheduler to route workload around such hotspots and make placement and timing decisions to minimize overall energy costs.

5. Topology

The computational processing rate of nodes as measured in flop/s is increasing substantially faster than the rate of bandwidth increases in the communication interconnects. Thus inter-node communication is becoming an ever-increasing bottleneck for parallel-application performance. Moreover, the topology of the network interconnect will continue to become a much larger factor in system scalability and application performance. With millions of nodes in future systems, it is not feasible for all nodes to be physically or even logically close to each other. While some nodes will be single hops apart, other nodes will be many hops in distance apart.

Schedulers that can make use of network-topology information to allocate nodes to jobs that are physically or logically close to each other will be needed. The placement of nodes in close network proximity will result in communication-bound application performance being many times improved over less optimal placements.

One question to ask is how much should a workload manager know about the topology? One could relatively easily construct a map indicating pair-wise connections of all nodes that have direct connections, but this fails to model the problem, since many of the hops will be through switches and routers. A list of distances between each one of a million nodes and every other node would include about 500 billion paths, not to mention the intractability of trying to find a least-distance node set within this data. One approach to this problem is to have the scheduler make a call to an external service for this information. The request would specify a list of nodes that have been prescreened as feasible for the job and possibly even presorted according to other preference and priority factors, along with a count of how many nodes are wanted. The topology service would then return a list of nodes that have topological proximity, which the scheduler would then allocate for the job. This model is currently being used by the Moab Workload Manager on BlueGene systems.

However, another very viable approach can be derived from the fact that almost all systems are delivered and organized in cabinets or pods. These units normally have up to a few hundred compute nodes that are physically and logically close to each other and among which the network topology is approximately flat. Thus, nodes can be organized into chunks, where members of each chunk are close to one other. These chunks themselves

may then be organized into chunks of chunks that are close to each other, depending on the actual network topology. This approach vastly simplifies the problem in that a scheduler can actually know about and manage scheduling within this granularity of detail. It can quite readily compute a first-order placement decision, which will come reasonably close to optimal performance. Jobs that are smaller than these chunks would simply be placed entirely within a chunk. Jobs that are larger than the chunk size would fill up whole chunks (cabinets or pods) and be placed in chunks that are close to each other. The need to address more than two layers of chunking in the proximal future is unlikely. Schedulers will need a way to allocate nodes across the minimal number of maximally filled chunks possible. It will also be useful to be able to express the closeness of these chunks to each other in order to minimize hops in cross-chunk communication. (Bhatele.)

6. Usage Considerations

A variety of usage considerations must be taken into account for any large-scale system. In addition to the technical considerations outlined in the previous sections, a system administrator must also look at how the system is going to be used by the target user base. Large-scale systems should not only be able to serve the current needs of the current users but also offer new, exciting possibilities to the administrators. With the increased capacity, system owners can potentially tap new markets and provide their users with new offerings. This section will not only outline some of the practical considerations for today's users but also provide additional ideas and directions for potential new uses for a petascale system in the areas of dynamic provisioning and virtualization.

Dynamic Provisioning

Just as system designers must decide what mix of hardware to use in their clusters, they are also faced with many software-mix decisions. One such important decision in this area is what operating system or mix of operating systems to install on the selected hardware. Many cluster users have very specific needs in terms of what operating systems are required to execute their workloads. Users' needs for specific operating systems can potentially lead to the creation of resource silos based on the software stacks.

Fortunately, unlike hardware changes, software changes are comparably easy to automate. Already, a number of clusters in the world automatically change their operating systems or other parts of their software stacks dynamically to meet workload needs. One such HPC system that has effectively deployed this type of solution is SciNet at the University of Toronto (Adaptive Computing). With such a system, it is necessary to increase and improve what is covered by a job definition. Workload must understand its own software requirements and be able to communicate those needs to the scheduler, just as it would communicate any requirement for processor or memory. In addition, to make the most effective use of the resources possible, an intelligent scheduler or orchestrator with the following attributes is needed:

1. Capable of receiving and understanding software requirements
2. Able to effect change to the underlying infrastructure
3. Understand trade-offs caused by temporarily taking systems offline to change operating system or software stack
4. Maintains future view of resources and future reservations to avoid potential thrashing situations

Virtualization

Another closely related topic is that of virtualization. Traditionally, virtualization has not been a consideration for heavily utilized HPC clusters because of the overhead and potential unpredictability that may be potentially injected into the cluster by virtualization and hypervisor technologies. However, as these technologies have matured and continued to receive increased support from hardware vendors, some members of the HPC community have expressed renewed interest in virtualization.

Virtualization provides a number of potential gains for an owner of a large-scale cluster. While one school of thought may approach this issue and simply state that enough hardware exists within a petascale system to accomplish any task, the opposite camp may respond with the fact that virtualization technologies make possible new types of workload on the cluster. So, it is a case of adding new capabilities, as opposed to simply replacing existing use cases. In fact, if virtualization were to only replace existing functionality, the inherit drawbacks of virtualization technology would almost certainly override any potential gains.

One potential use for virtualization technology in a large-scale system is that of being able to run user-defined images. This is closely related to our earlier discussion around the desired ability for a cluster to dynamically change its software infrastructure to meet user needs. In this slightly more advanced use case, users would be able to submit their workload encapsulated within a virtual machine image. This is especially

useful for end users who have very specific needs in terms of operating system, required libraries, specialized software, etc. Often these environments may be either comparably brittle or otherwise untenable for standard deployment within the cluster. However, by treating virtual machine images as just another type of workload, just like HPC batch, environments as workload can fall under the same set of policies and control as other workload. Naturally, creating a cluster capable of handling these more advanced use cases requires additional work and more intelligent management than a traditional HPC cluster. Such an effort requires a scheduler that can understand both types of workload and route them appropriately, as improved or additional resource managers may be needed to handle this enhanced form of workload.

Another potential area of benefit is that of checkpointing workload. Almost all virtualization technologies support a form of checkpointing. In a virtualized environment, this checkpointing can be used to archive running systems. It can be used to perform a checkpoint and restart function. When combined with the migration capabilities found in many virtualization technologies, many potential benefits arise. For example, large HPC clusters often have many large, long-running jobs. Assuming the I/O and network bandwidth issues can be addressed, a virtualized system running under an intelligent scheduler would be able to automatically migrate workload from servers that are beginning to experience problems as reported by built-in or external hardware-monitoring systems. This will become increasingly important as we continue to build larger systems with shorter MTBF periods.

Sources

Adaptive Computing. “Adaptive Computing Delivers Energy Efficiency to Canada’s Largest Supercomputer.” 17 Nov. 2009. 24 May 2010 <http://www.adaptivecomputing.com/news/2009scinet.php>.

Bhatele, Abhinav Sudarshan. “Scaling Scientific Applications to Exascale.” *Parallel Programming Laboratory: Department of Computer Science, University of Illinois at Urbana Champaign*. 24 May 2010 <http://charm.cs.uiuc.edu/~bhatele/job/Bhatele_res_stmt.pdf>.

Bianchini, Ricardo, et al. “System Resilience at Extreme Scale” (DARPA Exascale Report on Resilience). *LANL Institutes Office*. E. N. (Mootaz) Elnozahy, ed. Los Alamos National Laboratory. 24 May 2010 <http://institutes.lanl.gov/resilience/docs/IBM Mootaz White Paper System Resilience.pdf>.

Geist, Al. “Paving the Roadmap to Exascale.” *SciDAC Review*. IOP Publishing in association with Argonne National Laboratory, for the US Department of Energy, Office of Science. 24 May 2010 <http://www.scidacreview.org/1001/html/hardware.html>.

Kogge, Peter, et al. “ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems.” *Georgia Tech School of Electrical and Computer Engineering*. 28 Sept. 2008. 24 May 2010 <http://users.ece.gatech.edu/mrichard/ExascaleComputingStudyReports/exascale_final_report_100208.pdf>.

Wang, Chao, et al. “Proactive Process-Level Live Migration in HPC Environments.” 24 May 2010 <http://moss.csc.ncsu.edu/~mueller/ftp/pub/mueller/papers/sc08.pdf>.

Wladawsky-Berger, Irving. “Extreme Scale Computting.” *Irving Wladawsky-Berger: A Collection of Observations, News and Resources on the Changing Nature of Innovation, Technology, Leadership, and Other Subjects*. 15 February 2010. 24 May 2010 <http://blog.irvingwb.com/blog/2010/02/extreme-scale-computing.html>.