WHITEPAPER

Operation Radical Ascent

Game-Changer for HPC Scale and Performance

Adaptive COMPUTING

Table of contents

By the time commercial aviation became a reality, the automobile was thoroughly entrenched in the modern psyche. Driver licenses, traffic lights and painted lanes to efficiently route bumper-to-bumper cars were in everyday use. Policemen and motorists both understood speeding tickets.

Yet air traffic control, despite some shared DNA, is quite different from the systems that manage wheeled vehicles.

Why?

The answer lies in the complexity of air traffic, the speed with which decisions must be made and communicated, and the higher stakes for suboptimal choices. Airplanes cannot pull over if they run out of fuel. They require certain sizes of runway. They are more sensitive to weather. And they must think in three dimensions, not two.

The Need in HPC

The exascale wave in today's HPC market is creating a similar inflection point, where familiar solutions are simply inadequate. Modern supercomputers now have so many internal network interconnects and coordinate so many calculations at such a rate that the painted lanes and traffic lights of traditional scheduling cannot keep up. Jobs sit idle when they should be running; policy constraints are lost in the noise of hardware failure at scale; data remains opaque and unanalyzed instead of generating insight.

The exascale challenge is intensifying. A generation ago, most HPC problems consumed modest amounts of data in a single stage; many of today's projects require Big Data processing in complex workflows that are impossible to manage manually. This "Big Workflow" phenomenon multiplies problems at scale, raises the stakes for performance, and demands groundbreaking sophistication. Schedulers used to place with only CPU and RAM as major considerations; now workflow demands smarter policy and more savvy choices based on data locality and interdependent deadlines.

In 2013, Adaptive Computing recognized the opportunity and challenge inherent in this situation and made a strategic choice to invest in new scheduling technology. Dubbed "Operation Radical Ascent," the resulting initiative aimed to marry the best thinking from earlier generations of Moab with important changes to the fundamental engine, forever changing scale and performance standards for the industry.

The June 2014 release of Moab HPC Suite delivers on the Ascent vision for the first time in a big way. However, Operation Radical Ascent is still in full swing; subsequent releases are slated to introduce more innovation in additional waves. The final result should be as game-changing to HPC and Big Workflow problems as air traffic control was to modern airports.

The Zen of Ascent

The first thing that the Ascent team at Adaptive Computing did, when they were chartered, was to agree on guiding imperatives:

- Formalized measurement and the scientific method
- Parallelized, distributed, cooperative designs
- Changes that make the biggest difference to customers in real-world problem solving
- Better manufacturing process
- Reproducible results that are checked and rechecked to prevent backsliding

Initial Targets

As a down payment on this philosophy, the Ascent team brainstormed a series of formal metrics that they could collect, that would quantify progress. The HPC industry has long used LINPACK and similar benchmarks to assess the performance of supercomputers in a formal way; why, they reasoned, should we not have analogous numbers for the technology that runs those supercomputers?

After considerable debate, the team identified a small set of metrics as their initial focus. Each metric has a formal test procedure and associated reference hardware (Appendix A). What follows is just an informal summary:

ARTEC – Average Run-Time for Expensive Commands

On a large, busy cluster, how long does it take, on average, to run a read-only command that performs significant computation? A common symptom of an underpowered scheduler is that a command like Moab's "showstart," that predicts when a job is likely to start, may appear to hang for seconds or even minutes, waiting for a chance to claim attention. Performing well on this metric means that admins and end users always have a responsive system.

ATS100K – Average Time to Submit 100,000 Jobs

Given a realistic distribution of job types and sizes, how long does it take to submit 100k jobs? Experience told Adaptive Computing that performance on this metric would be challenged both by ingestion handling and by the overall backlog/calculation load as the queue size grew. Performing well on this metric means that a scheduler can handle both usage spikes and very large queues with ease.

ATEMJS – Average Time to Exit Many Jobs Simultaneously

When a job exits, there is a brief period of intense communication. On large clusters, we knew from experience that sometimes many jobs would exit at approximately the same time and that the overhead of passing status back and forth could overwhelm the cluster for as long as a few minutes. We also knew that communication on job-exit and communication for job status were related, so improving this metric would likely slash communication overhead for many other use cases.

SIT – Schedule Iteration Time

Given a moderately complex configuration, how long does it take to re-analyze the entire queue in a large, busy cluster, making new, re-optimized decisions about job placement? In large clusters with complex policies, the industry often sees times in minutes; the Ascent team wanted something much faster.

SICU – Schedule Iteration CPU Utilization

During the period of time when a scheduler is re-analyzing its queue, how efficiently does it use available processing power to make decisions? An old-fashioned, serial scheduler running on a box with eight cores might keep only one of them busy; a scheduler that scales with hardware should show a much broader, more savvy usage pattern.

These are not the only metrics that the Ascent team came up with. In future releases, more will be described and reported. Even in this release, much time has been spent measuring and tuning some additional dimensions. But these are the heart of Ascent's first focus, and the results show that they have paid off handsomely.

Moving the Needle

Once the Ascent team had articulated its worldview and had identified specific measurements that would reduce its progress to crisp numbers, it was time to formulate a plan of attack.

The general pattern was easy to guess: take baseline measurements and look for places where code could be rewritten or designs could be altered, such that things became much faster and more scalable.

But where, exactly, should changes be made?

As with the automobile-to-air-traffic inflection, Adaptive Computing was not starting from scratch when it launched Operation Radical Ascent. Certain aspects of scheduling were well understood, and some foundational metaphors and algorithms remained relevant. However, Adaptive Computing also recognized the need to challenge our thinking in fundamental ways. In Phase 1 of Ascent, the work delivered in the June 2014 release, they were particularly interested in the following new dimensions of the problem:

Parallelism

At the time Adaptive Computing launched Ascent, the scheduling algorithms that kept massively parallel supercomputers busy were, themselves, mostly serial. Moab and its competitors had their roots in theoretical work first productized in the 1980's and 1990s. At that time, computer science mainly used parallelism for enormous matrix math problems—not for multithreading the daemons that managed that computation.

Adaptive Computing knew they could change this. Parts of the decision-making at the heart of a scheduler are friendly to parallelization. For example, identifying the subsets of a cluster that might be available during the time range required for a particular job is something that can be done for many different jobs simultaneously. So is the calculation about which nodes match a particular job's theoretical hardware requirements, before filtering through the lens of policy constraints.

If Adaptive Computing could solve many aspects of a problem simultaneously, instead of doing each step in an inalterable sequence, they knew that modern hardware would reward them with significant improvements to both scale and

performance. This is the analog to air traffic control thinking in three dimensions, while terrestrial traffic limits itself to two.

Caching

Additional improvements could be derived from remembering the results of previous calculations, instead of repeating work each time the scheduler had the same question—or from operating from one copy of data while another copy was being modified.

The Ascent team quickly identified places where caching could pay big dividends. For example, Adaptive Computing found that diagnostic commands such as "showstart" and "mdiag" could often operate from a snapshot of the cluster's state, while other threads were modifying unrelated portions of Moab's object model. They also found that some of the computations at the heart of the scheduling loop could be eliminated as redundant if they remembered more intermediate work.

Communication

A major challenge in complex systems of all kinds is making sure that information flows to the right places at the right times. In air traffic control, quantum leaps in communication were enabled by radar and radio; Adaptive Computing needed analogous improvements in HPC.

Several aspects of communication, they knew, were particularly likely to be fertile fields for study: how "pbs_server" and "pbs_mom" communicate in TORQUE, for example, or how Moab sends "pbs_server" newly submitted jobs. These sections of code were attractive redesign possibilities because communication could be streamlined while also making it parallel. Instead of looping serially over large numbers of nodes that each needed to send or receive information, performing the loop in parallel fashion could create huge gains in performance or scale.

Mutexing

When multiple threads need to access the same shared information and there is any possibility of that information changing, software engineers often use a technique called mutexing to guarantee that access is granted in an orderly fashion.

This guarantee is important, but it is also expensive because other parts of a program can be blocked while they wait for a scarce resource to become available. In addition to its speed implications, mutexing can be painful because it introduces the possibility of deadlocks and (when done wrong) seg faults.

A final emphasis of Ascent efforts, then, was to mutex with great care and precision—doing enough to correctly enable parallelism, but avoiding performance penalties and guaranteeing robustness.

Phase 1 Results

The June 2014 release of Moab HPC Suite contains numerous improvements and design changes introduced by the Ascent team. Some of the headline achievements include:

Drastic Reduction in Command Latency (ARTEC)

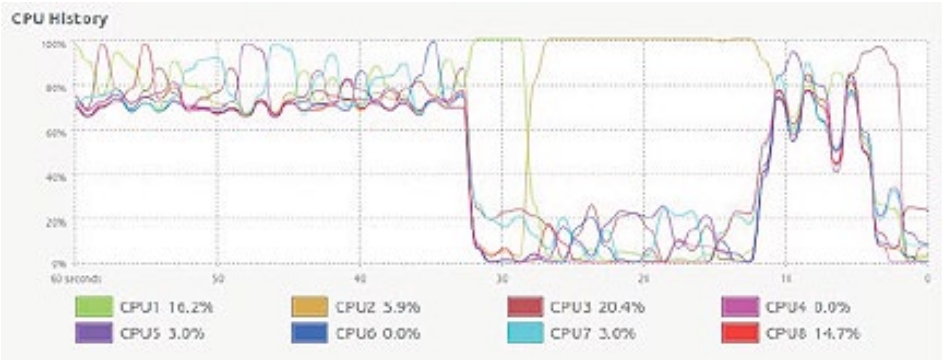
Even on the largest and heavily burdened clusters, it should now be possible to submit "expensive" read-only commands and get an answer within a few seconds, no matter whether the scheduler is busy or idle. A combination of cached data and more efficient use of background threads makes this possible.

Drastic Reduction in ScheduleIteration (SIT)

The time it takes to process a full queue of jobs, making new placement decisions, is now significantly less (between 3x and 6x faster according to benchmarks).

Huge Improvement in Proc Usage (SICU)

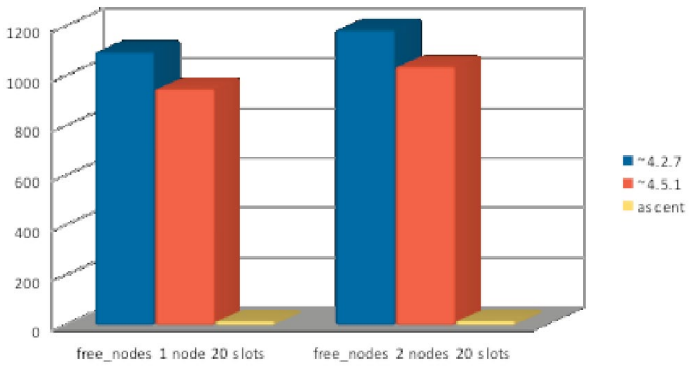
Moab now scales up with hardware—the more CPU horsepower you dedicate to the scheduler, the faster it goes. This is revealed by graphs like the following, which show Moab making full use of multiple cores during its scheduling cycle:



Importantly, this means that the hardware specifications for Moab can now be tailored to the needs of a specific cluster; a beefier server will run much faster than an underpowered one.

Dramatic Gains in TORQUE and Moab+TORQUE Communication (ATEMJS, ATS100K)

Moab now communicates newly submitted jobs to TORQUE using a more efficient API. Internally, TORQUE passes job information at start time, during subsequent status reports, and at job exit, in a way that is more robust and more efficient than ever before. The following graph shows one communication task that's been optimized (smaller is better; scale is microseconds):



Gains on other communication tasks are similar; while mileage will vary according to the makeup of a particular cluster, testing reveals that much larger queues are now practical and much more complex jobs flow with ease.

Raising the Bar

Adding Ascent design improvements to Moab represents a major step forward in the scale and performance of scheduler technology. Customers no longer need to grit their teeth at sluggishness when they put 50,000 jobs in a queue. Exascale scheduling is not just a pipe dream.

Of equal importance, Ascent establishes formal benchmarks by which Moab and its competitors can be measured. When evaluating choices, customers can ask for hard numbers from Adaptive and collect similar data points for their other options as well. Performance and scale become a science, not guesswork.

What's Next for Ascent

The performance gains that Operation Radical Ascent delivers in the June 2014 release are just the beginning. Many future changes are well underway, and some of them are multiplicative rather than additive in their effect. Here's a sneak peak:

More Parallelization

Moab currently collects data from its resource manager(s) as a discrete step in the scheduling loop, right before it begins re-analyzing the queue. This data ingestion work can be redesigned to overlap with other tasks, so that a slow resource manager has minimal effect on Moab's speed.

Smart Aggregates

Many of the nodes in a cluster are similar or identical for the purposes of certain algorithms inside Moab. Yet, today, Moab iterates over these identical nodes one at a time, computing the same answer for each one. Short-circuiting such loops with logic that recognizes that the next thousand nodes all look alike could drastically speed some computations.

Tighter Cooperation between Moab and TORQUE

Today, a significant amount of overhead in Moab-TORQUE communication derives from the fact that each of these applications has its own unique version of key structures. When Moab sends a job to TORQUE, the structure has to be serialized on one side, and de-serialized on the other. Harmonizing key structures would reduce overhead, as would communicating in batches instead of one-off messages. Where "pbs_server" and Moab are on the same machine, shared memory might be a further enhancement with huge upside.

Optimized File I/O

Independent of the Ascent work, the June 2014 release of Moab HPC Suite includes important enhancements for data staging. A future release could optimize how Moab and TORQUE use the file system in various ways: taking into account the characteristics of a parallel file system, for example, or altering how check-pointing and other persistence happens. This has the potential to drastically improve startup time, make checkpointing cheap, and add even more sophistication to data staging. That will pay off as Big Workflow thinking begins to permeate the HPC and Big Data marketplace.

Smarter Communication

In late 2013, Adaptive released the first version of its technology stack to use a high-speed message queue. The June 2014 release builds on this pioneering effort, making the message queue the backbone for the new Insight component.

Communication inside of TORQUE, and between Moab and TORQUE, could be further optimized and hardened by building on this message queue technology.

The View from Here

The HPC market needs something far more capable, more efficient, more scalable and more robust than schedulers have offered in the past. This is true because of exascale complexity and because Big Data demands it.

Fortunately, Ascent delivers. As the "radical" in its name implies, Operation Radical Ascent isn't about business as usual; it's about thinking faster, better and in more dimensions than ever before.

Talk to Adaptive sales about how you can test drive an Ascent-enabled Moab today.

Let's talk...Set up a Demonstration...and Test in your Environment

An Adaptive Computing solutions advisor can guide you to the products and services that will best meet your needs and will work with you to set up a live, online demonstration designed specifically for your organization.

Contact a solutions advisor by phone or email, or visit our website today

North America, Latin America +1 (801) 717.3700
Europe, Middle East, Africa +44 (0) 1483 243578
Asia, Pacific, Japan, India +65 6597-7053
Email: solutions@adaptivecomputing.com
www.adaptivecomputing.com

Corporate Headquarters

1712 S. East Bay Blvd.
Suite 300
Provo, Utah 84606

Adaptive COMPUTING