WHITEPAPER

Operation Radical Ascent

Performance Gains in HPC Speed and Scalability

Adaptive COMPUTING

Table of contents

Pakistan is known as the land of mountains. It is home to nearly 70 peaks above 7,000 meters (approximately 23,000 feet), and attracted mountaineers and climbers from all over the world seeking to ascend to the top of the country's many mountains and glaciers. Even with all the modern equipment available, however, there are still hundreds of peaks that have not been summited.

Today the world of High-Performance Computing is experiencing similar obstacles as they attempt to scale ever-mounting data workloads. There is a pressing need for scheduling software that can surmount these mountains of data. As a result of the convergence of HPC, cloud and big data, organizations are jumping from hundreds of thousands of jobs to possibly millions of jobs.

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 began the Ascent vision for the first time in a big way. However, Operation Radical Ascent is still in full swing, and the December 2014 release introduces more innovation in additional waves.

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 the June 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?

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 1980s 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.

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. The Ascent team knew several aspects of communication 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 contained 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 Schedule Iteration (SIT) (Figure 1)

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).

Figure 1

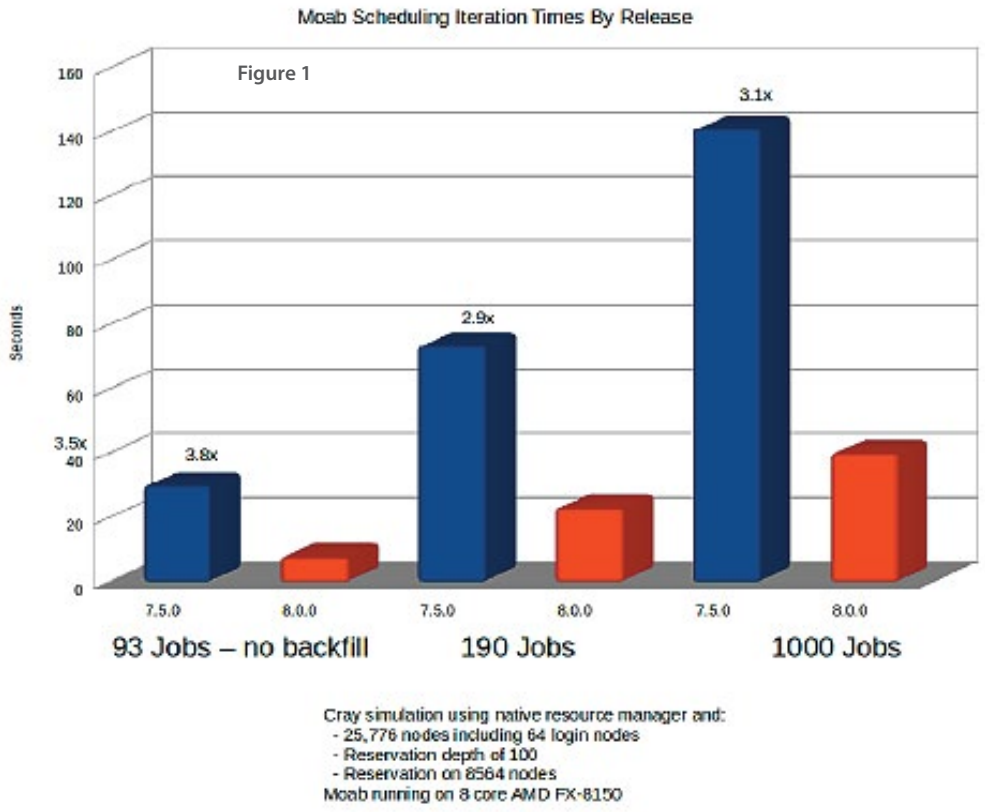
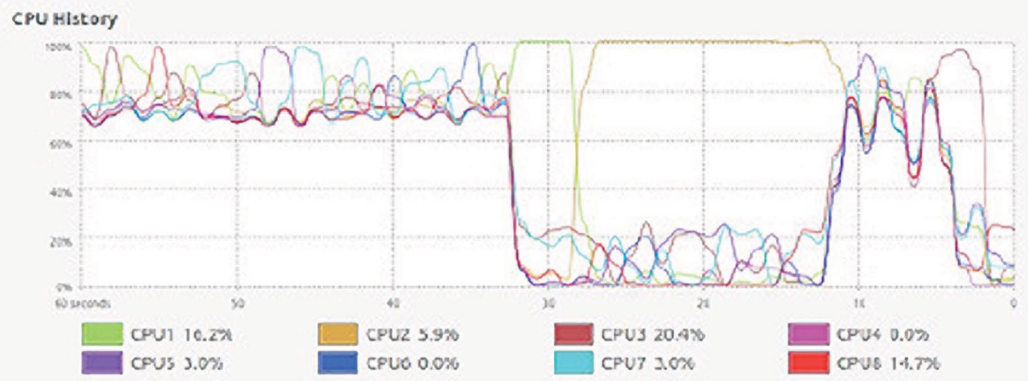


Figure 2



Huge Improvement in Proc Usage (SICU) *(Figure 2)*

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) *(Figure 3)*

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).

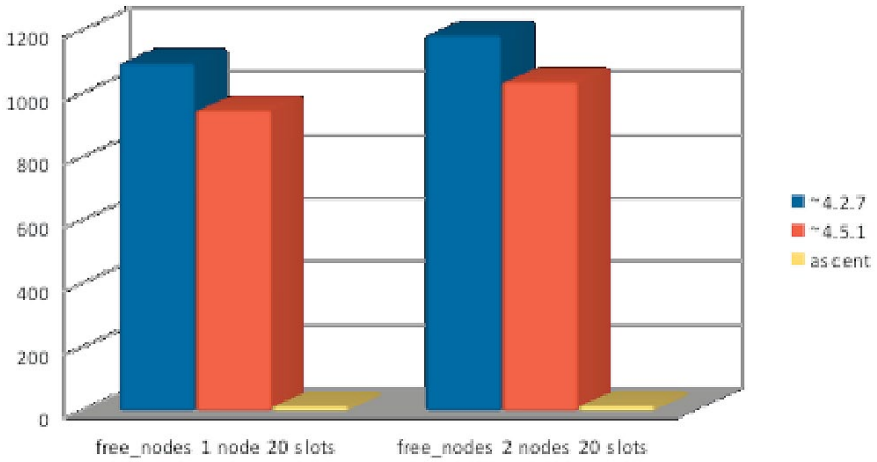
Gains on other communication tasks are similar; while mileage will vary according to the makeup of a particular cluster, testing reveals that much larger queues are now practical and much more complex jobs flow with ease.

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.

Figure 3



What's New for Ascent

The performance gains that Operation Radical Ascent delivered in the June 2014 release are just the beginning. The November 2014 release of Moab HPC Suite marks a significant advancement in Adaptive's Ascent vision and contains numerous improvements introduced by the Ascent team.

More Parallelization

In its previous version, Moab currently collected data from its resource manager(s) as a discrete step in the scheduling loop, right before it begins re-analyzing the queue. With Moab 8.1, this data ingestion work has been redesigned to overlap with other tasks, so that a slow resource manager has minimal effect on Moab's speed.

This improvement increases the decoupling between Moab and TORQUE's network communication so that Moab is less dependent on TORQUE'S responsiveness. As a result, Moab will never block users because TORQUE is being unresponsive due to a failed node or workload spike.

In a typical HPC environment, this enhancement results in 2x speed improvements and shortens the duration of the average scheduling iteration in half. These enhancements apply to TORQUE and all other resources managers. In addition, Moab is now able to poll multiple resource managers simultaneously, instead of one after the other.

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.

Moab 8.1 introduces innovations that harmonize these key structures to reduce overhead, and improve communication between the HPC scheduler and the resource manager(s).

This release introduces new capabilities for TORQUE to reduce check-ins with jobs with unchanged states by communicating with Moab in batches instead of one-off messages. This condensed status reporting reduces the network stress that was previously occurring between Moab and TORQUE during each scheduling iteration.

This condensed status reporting feature is optimized for HPC environments that are running longer scheduling iterations with jobs that run for days, weeks and even months. For certain use cases, this feature will be able to deliver 2x speed and scale improvements.

Enhanced Accounting Capabilities

Moab 8.1 introduces the ability to select one of three new alternate accounting modes that avoids the blocking lien enforcement in the default strict allocation mode. By selecting a higher throughput accounting mode such as fast-allocation, Moab can schedule job surges at the same pace as it does when not using accounting.

The accounting mode allows a site to specify what level of accounting (e.g. showback or chargeback) and usage level enforcement is needed for their notion of accounting. The accounting mode modifies the way in which Moab interacts with Moab Accounting Manager during the various stages of the job or reservation lifetime (e.g. job submission, job start, job completion, etc.). The accounting mode can be one of usage-tracking, notional-charging, fast-allocation or strict-allocation.

The following table describes the valid values for the accounting mode.

Value	Description
Strict-allocation	Use this mode if you wish to strictly enforce allocation limits. Under this mode, holds (called liens) will be placed against allocations in order to prevent multiple jobs from starting up on the same funds. Jobs and reservations may be prevented from running if the end-users do not have sufficient funds. This is the default.
Fast-allocation	Use this mode if you wish to debit allocations, but need higher throughput by eliminating the lien and quote operations of strict-allocation mode. Under this mode, jobs and reservations check a cached account balance, and may be prevented from running after the balance has become zero or negative.
Notional-charging	Use this mode if you wish to calculate and record charges for workload usage, but not keep track of fund balances or allocation limits.
Usage-tracking	Use this mode if you wish to record workload usage details, but not to calculate a charge nor keep track of fund balances or allocation limits.

When using one of the new alternative modes (usage-tracking, notional-charging or fast-allocation), scheduling overhead due to accounting can be reduced from a few tenths of a second per job, to less than a few thousands of a second per job. Depending on your existing scheduling overhead due to Moab and TORQUE policies, using one of the higher throughput modes has the potential of doubling job throughput on a sustained basis and improving peak job throughput by an order of magnitude. This is achieved by handling charges in the background in a separate thread and the elimination of the blocking lien in the strict-allocation accounting mode.

Stacking Benefits

When enabling the extra parallelization and the tighter cooperation, Adaptive Computing has found that things perform even better than simply adding the benefits for each. For some workloads Moab iterations could be up to 5X faster than using neither of the new features.

The Role of Ascent in Big Workflow

Adding Ascent design improvements to Moab represents a major step forward in the scale and performance of scheduler technology. It puts exascale scheduling within reach and strengthens the foundation to realize the potential of the next generation of HPC.

The massive performance gains made by Ascent are critical to Adaptive Computing executing its Big Workflow vision. In order to deliver accelerated insights and shorten the time to discovery, Moab must be able to streamline the workflow and schedule computing jobs across multiple platforms, environments and locations rapidly, accurately and cost-effectively. Operation Radical Ascent plays a critical role in making this innovation possible.

Adaptive Computing will continue to innovate around its Ascent Initiative to drive further improvements in speed and scaling.

The View from Here

With a decade of expertise in high performance computing, cloud, big data and data center automation, Adaptive Computing has a rich history of advancing research and accelerating insights through its Moab scheduling and optimization software. Operation Radical Ascent represents the company's latest effort to provide the HPC market with the most capable, efficient, scalable and robust scheduler available.

Talk to an Adaptive Computing sales representative about how you can leverage Ascent-enabled Moab in your environment 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