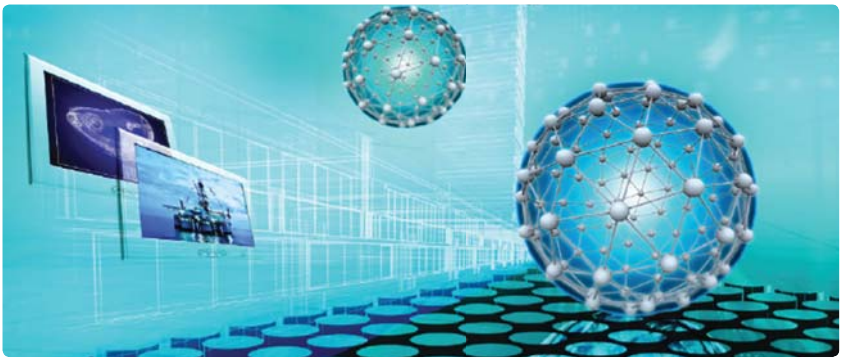
Adaptive COMPUTING

Scheduling for High-Throughput Computing



Nitro

High-Throughput Computing Convergence with High-Performance Computing Systems

As use of clusters/compute grids becomes more diversified, the industry is witnessing a convergence of high-throughput computing (HTC) with high-performance computing (HPC). While HPC workloads are compute/data intensive and can take from hours to months to complete on multiple nodes, HTC jobs tend to have short runtimes, often in the millisecond to minutes range on a single node or core. Nearly all HTC jobs can be classified as "embarrassingly parallel," meaning the workload can be divided up into multiple, autonomous serial pieces, each of which can be independently executed. Other workloads include regression tests, job arrays, or other tasks which create large queues. While not covered in this document, SOA workloads have many similar needs (see Nitro for SOA).

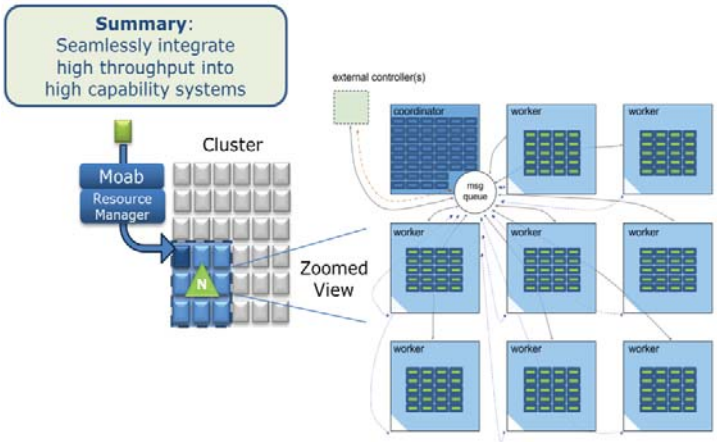
The Bottleneck in running HTC on HPC

To leverage HPC platforms, administrators need a scheduling tool which can meet HTC performance requirements on HPC systems. The typical HPC scheduler's decision engine excels at automating complex decisions while taking into account many factors and policies. However, in the case of HTC workloads, a complex decision process adds excess overhead and is unnecessary for such embarrassingly parallel jobs. Navigating around this decision process becomes exponentially more difficult and time consuming when HTC jobs number in the hundreds of thousands or millions.

Introducing Nitro an HTC-on-HPC solution

A scheduler is required to ensure a highly utilized HPC system, but it cannot efficiently make individual decisions on each job in an HTC scenario. An effective HTC scheduler needs to enforce policies on a set of HTC jobs once and then place those jobs on the optimal resources at the optimal time.

Enter Nitro - a high-throughput scheduling solution for traditional HPC systems. Nitro is a highly powerful, yet simple task launching solution that operates as an independent product but also integrates seamlessly with Adaptive's Moab HPC suite and other traditional HPC schedulers. Instead of requiring individual job scheduling, Nitro enables high-speed throughput on short computing jobs by allowing the scheduler to incur the scheduling overhead only once for a large set of jobs, effectively creating a Nitro "session." Nitro then quickly and efficiently executes the HTC jobs as "tasks" within this session without any additional HPC scheduling overhead. When pushing a high quantity of HTC jobs through the system, Nitro works in unison with the main HPC scheduler and parallel jobs running alongside the short-duration HTC jobs.



Nitro works with existing schedulers

- Nitro coexists with other HPC schedulers such as Platform LSF, PBS Pro, Moab/TORQUE, UGE, and SLURM.
- Because of Nitro's staged coordinator-worker architecture, Nitro throughput increases with the number of cores allocated to the session.
- Simple syntax to use: one line = one task and a line contains an executable command.
- If current HTC job syntax requires conversion, a tasks-filter can be created to automatically reformat into "task" definition syntax.

Nitro

Nitro Features

- Supports millions to billions of tasks
- Provides simple user job submission (API-based submission and monitoring is available in Nitro for SOA)
- Works on existing clusters with no application modifications
- Runs with any HPC scheduler or as a stand alone service
- Tasks can be named, given labels and are indexed for easy parsing and task identification
- Tasks on failed workers are moved to functioning workers
- Checkpoint tasks to enable restart in case of task interruption
- Dynamically add/remove nodes to/from a Nitro session to provide load balancing and meet Service Level commitments
- Linger Mode allows ongoing submission to an open session for near real-time workloads
- Track Nitro session status in real time (tasks completed, pending, failed, worker health, etc) via CLI or Moab Viewpoint
- Reports on task details (resources utilized, task run time, standard error, etc)

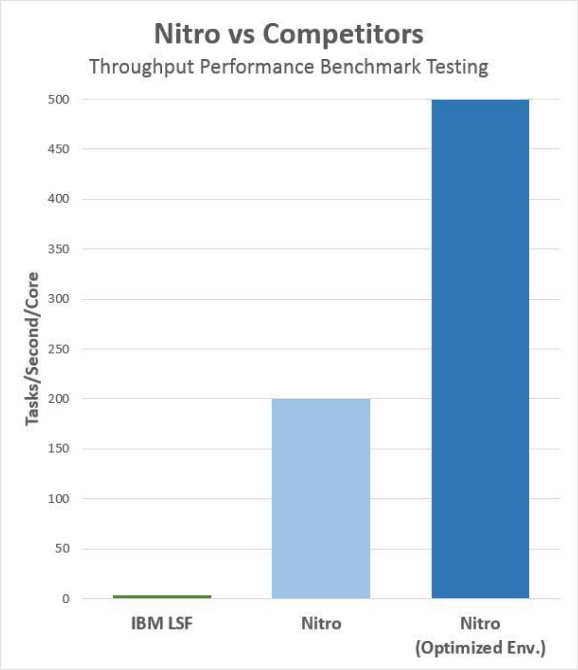
Optimal Use

Nitro is highly valuable to administrators who want drastically improved throughput on serial jobs or single node parallel jobs that run for milleseconds to minutes, though use cases of longer run times can see a benefit with Nitro as well. There are multiple use cases where if even a fraction of the workload matches the high-throughput use cases covered by Nitro, there is still significant overall system efficiency gained by running Nitro.

How Nitro Works

A user submits a Nitro job to the job scheduler. Nitro operates by using its scheduler-allocated nodes for a session where it executes tasks (short-running HTC jobs). Nitro starts a coordinator on the first node in the session. It reads a set of task definitions from a task file and delegates task assignments to each worker running on an allocated node in its session over a high-speed message bus. Each worker executes the assigned tasks, reports back to the coordinator for a new assignment, and so on until all tasks are completed.

Each Nitro job submission can build, activate, and tear down a new Nitro session. This is seamless and requires no management by administrators or users. Individual Nitro sessions can scale up to a few hundred nodes (enough to launch hundreds of thousands of tasks per second); a given cluster can simultaneously host as many or as few Nitro sessions as need dictates.



Nitro Benchmarks

In benchmarked testing against IBM Platform LSF, Nitro displayed a completion rate of 200 tasks per second per core, outperforming LSF by 54x. In an optimized environment it ran 500 tasks per second per core, putting it over 100x faster than LSF. In another performance test, Nitro was able to run over a billion tasks on a single server over a weekend. Clearly this tool puts high throughput performance into your computing center and is ready to handle any volume you can throw at it.

Hardware Requirements

Nitro is designed to work in a wide variety of environments and systems. The hardware required for Nitro is 500 MB of RAM and two hardware threads. With additional hardware Nitro can launch jobs even faster. There are no limits on the task file size, and increasing the task file size does not increase hardware requirements.

For more information contact: Lars Stenstedt
North America, Latin America +1 (510) 299.2151
Asia, Pacific, Japan, India +1 (510) 299.2151
Europe, Middle East, Africa +44 (0) 1483 243578
Email: lars@adaptivecomputing.com
www.adaptivecomputing.com

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

Adaptive COMPUTING