

# Moab 6.0 Release Notes

19 Jan 2011

The table below describes the Moab 6.0 new features/enhancements and the product suite(s) to which they apply.

- MCS = Moab Cluster Suite
- HPC = Moab Adaptive HPC Suite
- MAC = Moab Adaptive Computing Suite

Feature Name	Description	MCS	HPC	MAC
<b>Moab Requirements</b>	<ul style="list-style-type: none"> <li>• <b>Minimum System Requirements</b> <ul style="list-style-type: none"> <li>– Quad-core processor</li> <li>– 8 GB RAM</li> <li>– 80 GB disk space</li> <li>– Optional ODBC-compliant database</li> </ul> </li> <li>• <b>Component Compatibility</b> <ul style="list-style-type: none"> <li>– Viewpoint 2.0</li> <li>– MSM 6.0</li> </ul> </li> <li>• <b>Backward Compatibility</b></li> </ul> <p>When running Moab 6.0 in a grid configuration, administrators must upgrade all grid instances to Moab 6.0. Moab 6.0 is incompatible with previous versions.</p>	•	•	•
<b>General Enhancements</b>	<ul style="list-style-type: none"> <li>• <b>New Default Directories</b></li> </ul> <p>The default Moab programs (binaries) installation directory has changed from /usr/local to /opt/moab. The default Moab configuration file directory has changed from /opt/moab to /opt/moab/etc.</p> <p><i>Note: Moab first searches in the /opt/moab directory for its configuration files (moab.cfg and moab-private.cfg) and second in the /opt/moab/etc directory in order to maintain backward compatibility.</i></p> <ul style="list-style-type: none"> <li>• <b>Parameterized Generic Metrics</b></li> </ul> <p>Moab now supports a configurable quantity of generic metrics (GMETRIC) specified by the new MAXGMETRIC parameter in the moab.cfg file, illustrated below.</p> <pre>MAXGMETRIC 100</pre> <p>This feature permits administrators to set a limit on the quantity of different generic metric objects Moab internally allocates based on the needs of the cluster/cloud.</p> <p><i>Note the quantity needed is relative to different <u>types</u> of generic metrics defined (different GMETRIC names), not multiple <u>instances</u> of the same GMETRIC name assigned to different objects such as nodes, etc.</i></p> <p>If Moab reaches the MAXGMETRIC limit, an administrator simply raises the limit in the Moab configuration file and then restarts Moab to implement the new limit, which is much better than the old method of compiling the limit into the Moab product.</p> <ul style="list-style-type: none"> <li>• <b>Parameterized Generic Resources</b></li> </ul> <p>Moab now supports a configurable quantity of generic resources (GRES) specified by the new MAXGRES parameter in the moab.cfg file, illustrated below.</p> <pre>MAXGRES 150</pre> <p>This feature permits administrators to set a limit on the quantity of generic resource objects Moab internally allocates when it starts up based on the needs of the cluster/cloud.</p> <p>The value of this parameter affects the amount of memory Moab uses; the larger the limit, the greater the quantity of memory used. Administrators should try to keep this limit as close as possible to the actual GRES quantity defined. If not specified, the default value is 127.</p> <p>Administrators should keep in mind that Moab defines ten (10) generic resources for its own internal purposes. This means that if a Moab site has 100 different GRES definitions, this parameter must have a value of at least 110.</p>	•	•	•

Feature Name	Description	MCS	HPC	MAC
	<p><i>Note: The quantity needed is relative to different <u>types</u> of generic resources defined (different GRES names), not multiple <u>instances</u> of the same GRES name assigned to different objects such as nodes, etc.</i></p> <p><i>However, there is an exception with "typed" generic resources. Each typed GRES definition counts toward the MAXGRES limit. For example, in the following moab.cfg file, the "gpu" generic resource definitions (GRES=gpu:2) for the two nodes count against the MAXGRES limit only once, while each of the four typed GRES definitions (e.g. gpu0_0 in GRES=gpu:2,gpu0_0,gpu0_1) count against the limit, yielding a total of five (gpu + four typed gpuX_N).</i></p> <pre style="background-color: #f0f0f0; padding: 5px;"> NODECFG [node0]      GRES=gpu:2,gpu0_0,gpu0_1 NODECFG [node1]      GRES=gpu:2,gpu1_0,gpu1_1 GRESCFG [gpu0_0]     TYPE=gpu GRESCFG [gpu0_1]     TYPE=gpu GRESCFG [gpu1_0]     TYPE=gpu GRESCFG [gpu1_1]     TYPE=gpu                     </pre> <p>If Moab reaches the MAXGRES limit, an administrator simply raises the limit in the Moab configuration file and then restarts Moab to implement the new limit, which is much better than the old method of compiling the limit into the Moab product.</p> <ul style="list-style-type: none"> <li>• <b>Unlimited Triggers</b></li> </ul> <p>Moab now supports an unlimited trigger quantity, which is subject, of course, to the limitation of total memory on the Moab head node.</p> <ul style="list-style-type: none"> <li>• <b>Unlimited Users, Groups and Accounts</b></li> </ul> <p>Moab now supports an unlimited quantity of users, groups, and accounts, which is subject, of course, to the limitation of total memory on the Moab head node.</p> <ul style="list-style-type: none"> <li>• <b>ODBC-compliant Database Support</b></li> </ul> <p>Moab now supports the option to store node and job information in an ODBC-compliant DBMS (e.g., MySQL), which permits real-time access to the information by external applications. This feature requires the installation of an ODBC-enabled build of Moab 6.0 in the head node. To enable this feature, add the following line to the moab.cfg file.</p> <pre style="background-color: #f0f0f0; padding: 5px;"> USEDATABASE          ODBC                     </pre> <p>Note the ODBC-compliant database is for use by external applications; Moab does not use it but only writes information to it.</p> <p><i>Note: If upgrading from an earlier version of Moab, read the README.database file. In addition, the contrib/sql/moab-db.sql file contains SQL statements that create the job and node information tables, which need to match the syntax used by the ODBC-compliant database. The file uses MySQL syntax.</i></p> <p><i>Note: Be aware that some databases may truncate some values due to internal string size limitations, which may happen for information from large clusters with many nodes and/or jobs using many resources.</i></p> <ul style="list-style-type: none"> <li>• <b>Moab "man" Pages</b></li> </ul> <p>All Moab client or "command line" commands have updated "man" pages that display the same information as the online Moab 6.0 documentation.</p>			
<b>Scalability Enhancements</b>	<ul style="list-style-type: none"> <li>• <b>Multi-tiered, Multi-threaded Job / Node Information Storage</b></li> </ul> <p>Moab has increased its scalability by using a new two-layer caching model. The first layer is an internal cache and the optional second layer is multi-threaded node and job information storage in an ODBC-compliant database. Testing has shown that larger clusters noticeably benefit from this feature.</p> <p>Prior to Moab 6.0, Moab reported job and node information on demand by gathering the information from internal data structures and then reporting it back to the requester. This gathering and reporting happened on every request and required the Moab core intelligence engine to stop its scheduler processing during the time it spent gathering and reporting the information. If many users were requesting information, which happened more often on larger clusters, the</p>	●	●	●

Feature Name	Description	MCS	HPC	MAC
	<p>reporting bottlenecked Moab's scheduling, often with undesirable results (e.g. client timeouts, late triggers, etc).</p> <p>After a scheduling iteration, Moab 6.0 saves job and node information to an internal cache and then optionally queues the information for writing to an ODBC-compliant database using a dedicated writing thread. Moab uses the internal cache for reporting job and node information, while external applications use the ODBC-compliant database for real-time access of the same information.</p> <p><i>Note: This new internal caching and information queuing (for database writing) consumes memory and is one of the main reasons why the minimum system requirements for Moab 6.0 specify eight gigabytes of memory.</i></p> <p><i>Note: Testing has shown head nodes with faster processors are able to process the database write queue faster and therefore actually use less memory than head nodes with slower processors.</i></p> <p><b>• Multi-threaded Job / Node Information Reporting</b></p> <p>Moab has increased its scalability, including in a grid configuration, with multi-threaded node and job information reporting using an internal cache.</p> <p>Prior to Moab 6.0, Moab client commands directly communicated with the core intelligence engine, which meant that if Moab was busy communicating node and job information to users, this became a bottleneck for workload scheduling. This included communicating node and job information between Moabs within a grid.</p> <p>Moab now has a dedicated execution thread that responds to job and node information requests by client commands. This thread retrieves the requested information from the new Moab job/node information cache to fulfill the user requests for job and node information.</p> <p>The following specific Moab client commands and options below use this new multi-threaded implementation to obtain cached node and job information to report to the user.</p> <pre style="background-color: #f0f0f0; padding: 5px;"> mdiag -j mdiag -n showq </pre> <p>Users or scripts issuing these particular client commands and options will not notice any functional difference in their execution, but will notice quicker response times, especially in larger clusters. In addition, the Moab intelligence engine no longer stops its scheduler processing to respond to the commands, which means Moab processes scheduling activities (e.g., triggers) in a timelier manner, especially for larger clusters/clouds.</p> <p>Testing has shown consistent reporting response times around 0.1 seconds regardless of node and job information quantity. In addition, testing has shown consistent response times between 1-2 seconds when retrieving node and job information from the ODBC-compliant DBMS, which external applications can expect when accessing the database.</p> <p>To disable use of this multi-threaded, non-blocking, cache-based reporting feature by the Moab client commands listed above, add the following line to the moab . cfg file. The commands will use the former blocking method used in pre-Moab 6.0 versions. The default is to use the multi-threaded, non-blocking, cache-based reporting method.</p> <pre style="background-color: #f0f0f0; padding: 5px;"> DISPLAYFLAGS USEBLOCKING </pre> <p><b>• Thread Pooling</b></p> <p>If users or Moabs in a grid heavily use the client commands listed above, Moab can increase the throughput for these commands by using additional execution threads. This enhancement gives administrators the ability to specify the size of the thread pool used for responding to non-blocked client commands. To configure the thread pool size, set the THREADPOOLSIZ parameter in the moab . cfg file. The default size of the thread pool is 3.</p> <pre style="background-color: #f0f0f0; padding: 5px;"> THREADPOOLSIZ 4 </pre>			

Feature Name	Description	MCS	HPC	MAC
	<ul style="list-style-type: none"> <li>• <b>Grid Communication</b> Moab has increased its grid scalability by improving the grid communications performance. Moab communication between clusters within a grid configuration now use a multi-threaded implementation to allow the core intelligence engine to perform its scheduling processing without the need to stop in order to communicate with a cluster. In addition, it uses non-blocking communications between Moab-managed clusters.</li> <li>• <b>Database-backed Checkpointing</b> Moab now supports saving checkpoint information in a database. For larger clusters, tests have shown checkpointing to a database is faster than the old method of checkpointing to a flat file. To enable this feature, add the following line to the moab.cfg file. <pre>CHECKPOINTWITHDATABASE TRUE</pre> <i>Note: For new installations requiring checkpointing, administrators should turn on and use this feature.</i> <i>Note: If upgrading an existing Moab installation, enabling this feature while running active jobs will destroy existing checkpoint files. To safely enable this feature, administrators must first drain the cluster/cloud of all existing jobs, enable the feature, and then allow job execution to resume.</i></li> </ul>			
<b>GPU Support Enhancements</b>	<ul style="list-style-type: none"> <li>• <b>Native GPU Support</b> Moab has been scheduling GPU systems for many years using "typed" generic resources (example shown under Parameterized Generic Resources above). Moab 6.0 has integrated support for TORQUE's new GPU resource reporting and requesting. Moab now recognizes "GPU" as a node resource keyword and automatically populates internal node resources, including GPUs, as reported by TORQUE. Jobs / workloads / applications can now request GPUs as they can CPUs or memory. If a Moab site uses the "typed GRES" method for GPU scheduling and installs Moab 6.0 and TORQUE 2.5.4, the administrator may remove the GPU-related typed generic resource definitions from the moab.cfg file and use the native TORQUE GPU support, which conveniently tracks and schedules individual GPU devices within a node. Moab then simply tracks the GPU quantities scheduled and allows TORQUE to schedule and track the individual GPU devices and to indicate the scheduled GPU devices to the job/workload using its usual method of allocated resource notification.</li> <li>• <b>TORQUE-based GPU Scheduling Support</b> TORQUE version 2.5.4 supports native GPU resource reporting and scheduling. Since Moab recognizes "GPU" as a node resource, it automatically populates its internal node resources, including GPUs, as reported by TORQUE. Administrators must install TORQUE version 2.5.4 or later in order to use the native GPU support with Moab 6.0. <i>Note: TORQUE 2.5.4 does not perform GPU auto-detection, although TORQUE 2.5.5 is scheduled to do so and then only for NVIDIA GPUs. If a node is running TORQUE 2.5.4 or is running TORQUE 2.5.5 with a non-NVIDIA GPU, the administrator must specify the GPUs to TORQUE in the manner it requires for resources it does not automatically detect.</i></li> </ul>	●	●	●
<b>Virtualization Enhancements</b>	<ul style="list-style-type: none"> <li>• <b>HV/VM Feature Availability</b> The following virtual machine (VM) and hypervisor (HV) management features are now in general availability. These include the following. <ul style="list-style-type: none"> <li>– Hybrid hypervisors (ESX and KVM in same cluster/cloud)</li> <li>– Auto-migration of VMs due to Overutilization (see below)</li> <li>– Over-provision Multipliers use Utilization Target Percentages</li> <li>– Hypervisor Overcommit (more 1-processor VMs than server cores)</li> </ul> </li> </ul>			●

Feature Name	Description	MCS	HPC	MAC
	<p>• <b>Hypervisor and VM Auto-migration Exclusion</b></p> <p>Moab now has a way for administrators to prevent it from automatically or manually migrating VMs or hypervisors on specific nodes. Administrators can specify the new node configuration flag <code>novmmigrations</code> for nodes on which Moab should not perform VM and hypervisor auto-migrations or administrators should not manually perform migrations, as illustrated below.</p> <pre>NODECFG[node1]      FLAGS=novmmigrations</pre> <p>• <b>New Trigger Internal Event Types</b></p> <p>Moab has two new VM migration-related internal event types for use with triggers. Administrators can use these events to trigger migration of VMs due to VM over-commitment or "green" policy at specific times of the day.</p> <p>The new internal event type names are <code>overcommit</code> and <code>green</code> and appear in the <code>action</code> option of a TRIGGER definition in the <code>moab.cfg</code> file.</p> <p>To illustrate the overcommitted VM-based VM migration event type, assume VMs experience heavy use during the working day. A VM migration based on VM over-commitment policy could spread VMs among more hypervisors for better response times. The example below illustrates triggering VM migrations based on VM over-commitment policy at 8:00 (8 AM) every day. This policy moves VMs from overcommitted hypervisors to underutilized hypervisors before the VMs become busy with heavy day use.</p> <pre>SCHEDCFG[Moab]      TRIGGER=Atype=internal, \                     Name=morning_action, Etype=standing, \                     action=sched:-:vmmigrate:overcommit, \                     Period=day, Offset=8:00:00</pre> <p>To illustrate the green-based VM migration event type, assume VMs experience light use during evenings and nights. A VM migration based on green policy could consolidate (pack) VMs onto fewer hypervisors, allowing Moab to power down physical servers to conserve energy. The example below illustrates triggering VM migrations based on green policy at 18:00 (6 PM) every day.</p> <pre>SCHEDCFG[Moab]      TRIGGER=Atype=internal, \                     Name=evening_action, Etype=standing, \                     action=sched:-:vmmigrate:green, \                     Period=day, Offset=18:00:00</pre>			
<b>Event Enhancements</b>	<p>• <b>Alternative Job and Reservation Event Information Format</b></p> <p>Moab can now output job and reservation event information in the so-called Moab "native wiki"<sup>2</sup> format, which uses the "<code>&lt;attribute&gt;=&lt;value&gt;</code>" syntax instead of the older whitespace-delimited, positional format. This permits greater format flexibility for handling different types of data and values, and facilitates easier parsing of the information by software. The older format simply delimited values with whitespace and one required knowledge of the position of values to know which value represented what information. With the "native wiki" format, one can now easily identify the values.</p> <p>Add the following line to the <code>moab.cfg</code> file to enable the output of job and reservation event information in the "native wiki" format. The default, output job and reservation events in whitespace-delimited, positional format, maintains backward compatibility.</p> <pre>WIKIEVENTS          TRUE</pre>	•	•	•

<sup>1</sup> Note that a "\ (backslash) at the end of a `moab.cfg` file line in this document means line continuation. The backslash should not actually appear in the file nor should the line break as it does in this document, meaning the entire continued line should be a single line in the `moab.cfg` file with no backslashes.

<sup>2</sup> The term "wiki" is the Hawaiian word for "quick" or "fast" and in Moab has nothing to do with Wikipedia or any wiki-style markup language. Since the Moab Workload Manager descended from the open-source Maui workload manager, a Hawaiian word seemed appropriate at the time as a name for the fast "native" format.

Feature Name	Description	MCS	HPC	MAC
<b>Job Submission Enhancements</b>	<ul style="list-style-type: none"> <li>• <b>Cray Syntax Support</b></li> </ul> <p>The msub client command now recognizes the Cray ppn= (processors per node) resource attribute name.</p> <pre style="background-color: #f0f0f0; padding: 2px;">msub -l nodes=5:ppn=4</pre>	●	●	●
<b>Job Management Enhancements</b>	<ul style="list-style-type: none"> <li>• <b>Forced Job Removal</b></li> </ul> <p>When submitting a VM job that had VM storage requirements, Moab started a job to create the VM and another job to create the VM storage. If the VM storage job failed to start, both the VM creation and VM storage jobs remained idle. Administrators could remove the VM creation job but the VM storage job became orphaned.</p> <p>The Moab client command mjobctl (Moab Job Control) has a new option -F (force job removal) that forces the removal of a job regardless of its state.</p>	●	●	●
<b>Deprecated Features</b>	<ul style="list-style-type: none"> <li>• <b>Resource Managers</b></li> </ul> <p>Moab no longer supports the following deprecated resource managers.</p> <ul style="list-style-type: none"> <li>- rms</li> <li>- sql</li> <li>- sss</li> </ul>	●	●	●