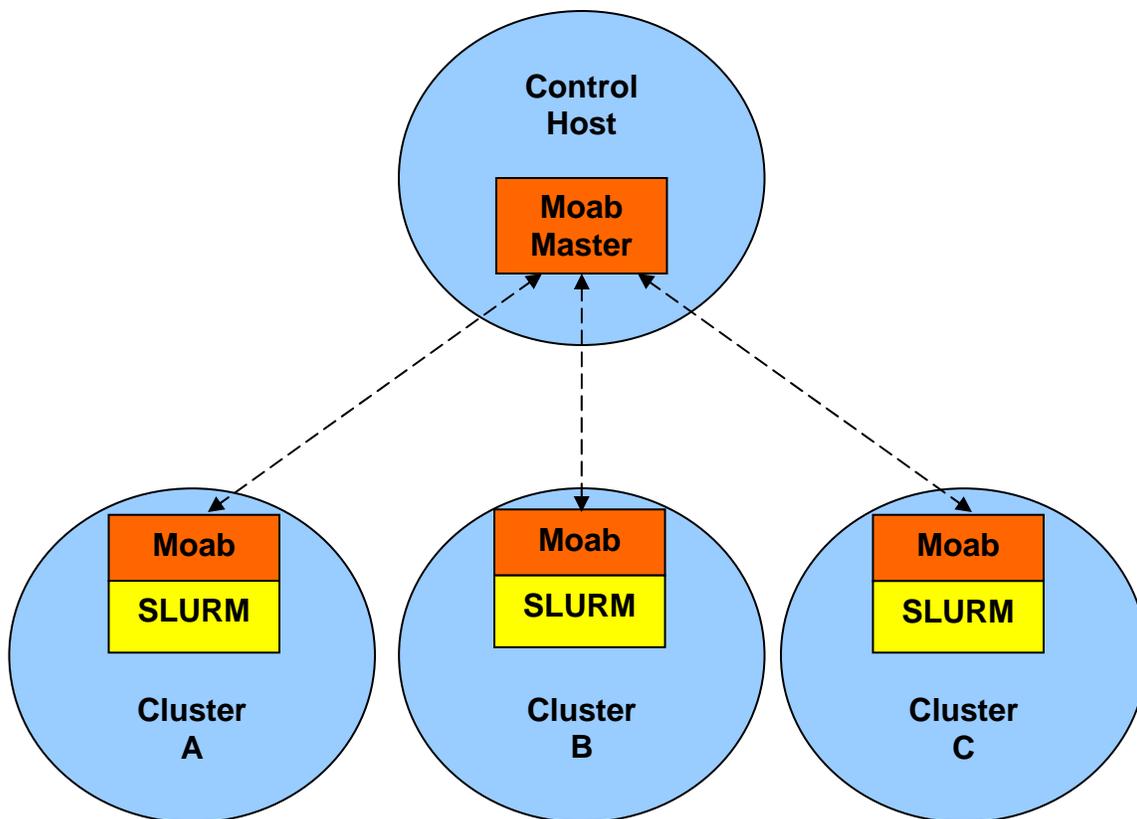


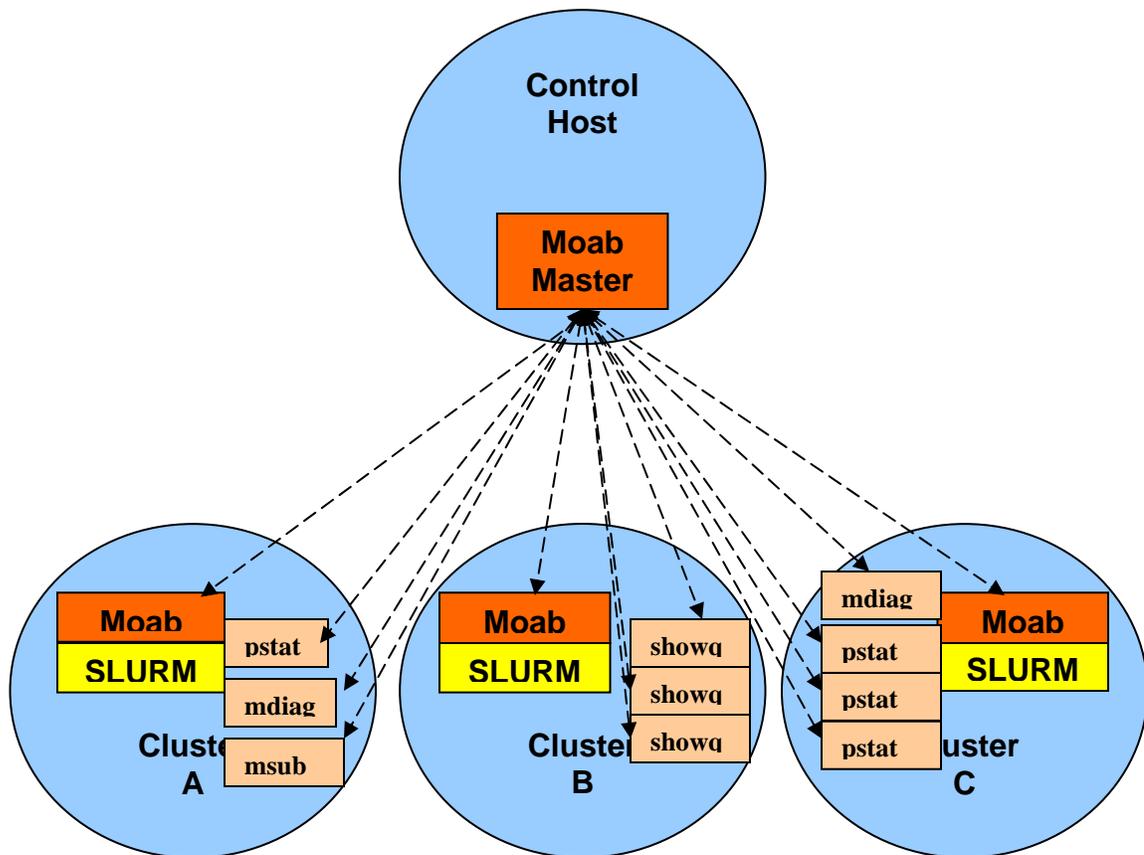
Performance Issues with the Moab Workload Manger

The production clusters scheduled by the Moab Workload Manager at LC are typically configured into a grid. There is one Moab master responsible for communicating with slaves stationed on the management nodes of each cluster. The Moab master sends the slaves the jobs to run, and the slaves return job and node status to the master, as illustrated in the figure below.



All of the client commands (`msub`, `showq`, `pstat`, etc.) invoked by users and jobs on any of the nodes of the cluster also communicate directly to the Moab grid master. This design puts a significant load on the grid master under normal conditions. Performance problems arise when there are a large simultaneous number of client requests to the grid master. Not surprisingly, the Moab grid master can get severely bogged down under such circumstances.

The messaging traffic problem is exacerbated by jobs that invoke the status commands (like `showq` or the `pstat` wrapper) once per second on every node of a parallel job, as illustrated in the figure below.



There are alternate architectures that would relieve the messaging traffic on a single daemon. Moab’s vendor, Cluster Resources, is currently exploring alternative designs that will alleviate this vulnerability.

In the meantime, users are asked to adopt the following “good neighbor” practices to minimize the burden on the Moab master daemon. Users should adopt these practices whether they are invoking Moab client/LCRM wrapper commands from a cron job, from a batch job, or manually from the command line.

1. Do not invoke Moab client status commands (`showq`, `checkjob`, `mdiag`, or the `pstat` wrapper) more than once per minute per job.
2. If you are interested in only your jobs, invoke `showq` with the `-u` option to limit the output to just your jobs, e.g., `showq -u <user_name>`.
3. If you are interested in jobs in a certain state (running or idle or blocked) invoke `showq` with the `-r`, `-i`, or `-b` option to see jobs of that state, e.g., `showq -r` to see running jobs.
4. If you are interested in jobs on just one cluster, invoke `showq` with the `-p` option to see jobs from just that cluster (or see item 7 below), e.g., `showq -p <cluster_name>`.

5. Combine the last three items. For example, to see just your running jobs on a given cluster: `showq -r -u <user_name> -p <cluster_name>`.
6. If you use the `pstat` wrapper to get job status, follow the `pstat` equivalent options to the last four items. For example,
`pstat -R -c <cluster_name> -u <user_name>`.
7. Consider using the `squeue` command. It stays local to the cluster and doesn't involve the Moab master at all. For example,
`squeue -u <user_name> -t running`

Submitting Jobs

Moab offers the ability to limit the number of jobs a user has running on a cluster. There is currently, however, no way to limit the number of jobs a user submits to the Moab queue. Consequently, a user can submit hundreds of jobs and Moab will place them all in the queue.

After hundreds of jobs have each been submitted by hundreds of users, Moab's responsiveness and scheduling performance starts to degrade. We therefore advise another "good neighbor" practice of limiting the number of jobs a user has queued across the grid to 150.

For users who need to submit a large number of jobs to the queue, we advise placing the `msub/psub` command in the job script to submit the next job at the time the first job starts to run. This mechanism avoids flooding the queue with jobs and yet perpetuates a steady stream of job submissions to the queue.

As an example, here is a Moab job script that submits a job when the first job starts to run:

```
#MSUB [...]
msub MyNextJob.cmd
MyApp
```

It is also possible to submit a dependent job (from the job script) that will only run after the first job completes.

```
#MSUB [...]
msub -l depend=$SLURM_JOBID MyNextJob.cmd
MyApp
```

`MyNextJob` will only run once the job running the above script completes.

Corrective Action

Until the Moab product is enhanced to guard against frequent status requests or limit the number of jobs a user can submit to the system, system administrators and operators will be monitoring the system and manually intervening when abuses are detected. The form of intervention will include canceling excessive jobs or canceling running jobs that bog the system with excessive status requests.