MetaBatch
Preliminary Design

George Flanagin

Revision of Sunday 14th May, 2023

IMPORTANT: *MetaBatch* is not a very good name. The recent collapse of the Metaverse and generally bad feelings about the Meta corporation have made *MetaBatch* an unattractive name, even if it is descriptive. We are currently soliciting better names from the user community.

1 What does *MetaBatch* do?

MetaBatch has three functions:

1. **Optimized use of the cluster.** SLURM's scheduling algorithm for any partition is a simple search for the next available space on the next available node, rather than best fit. On clusters with 1000 nodes that are generally very busy, the algorithm works well enough. On smaller clusters, we want to use every available cycle.

   It we think of the cluster usage as a number of finite rectangles into which we wish to place LEGOs of varying sizes, the ideal is to find a place where a job will fit exactly, thereby using all the cores on a node, and all its memory.

   Alternatively, we can override SLURM's behavior to always schedule on lower numbered nodes before higher numbered ones within the same partition.

2. **Application of University of Richmond's business rules.** As an example, there is no way to tell SLURM that the owner of a condo-node is on sabbatical or that it is OK for a particular user to run jobs on a node that the user does not ordinarily use without editing the `slurm.conf` file. The `slurm.conf` file is designed (primarily) to express facts about the hardware rather than apply business rules.

3. Functional proofreading of SLURM scripts. For example, the Gaussian software is not aided by more than 12 cores, and *MetaBatch* can edit SLURM files before they are submitted to be run.

2 User experience

In general use, the users will have no experience of metabatch. As they have always done, they will type `sbatch somefile.slurm` to submit jobs for scheduling. A system wide shell function named `sbatch` will wrap itself around the “real” `sbatch` command. Any non-default operations will be concealed inside “expert mode” switches. At the moment, there does not seem to be an obvious need for many options, perhaps only these three:

--dry-run Perform a syntax check and edit of the submitted SLURM script, but do not run it, and instead tell the user what would have happened.

--no-best-fit Let SLURM send it to the next available location rather than searching for place to run that uses the node maximally.

--test Adjust the time to some short duration (named in the config file mentioned below) for the purpose of assessing resource usage.

It is worth noting that, should *MetaBatch* fail or crash, the `sbatch` submission will revert to the standard SLURM program of the same name and the operations it is configured for.

3 Design of the *MetaBatch* program

1. The primary code will be located in a resident dæmon, `metabatchd`.
2. Configuration will be in a multipart, standard configuration file in `/etc/metabatch-/config`, and additional configuration files in `/etc/metabatch/conf.d`.

   The configuration file will contain information about:

   - ✧ Business rules that involve users and condo-nodes.
   - ✧ Program limits for CPU and memory allocation.
   - ✧ Rules about load balancing across nodes.
3. Communication with the dæmon will be done by sending standard Linux signals. For example, standard use in Linux is `SIGHUP` for re-reading configuration files, `SIGTERM` to close the program in a graceful shutdown, *etc.*
4. It is unnecessary for the dæmon to be running on the headnode. The dæmon only needs to know the IP addresses of the headnode[s]. In this situation, one dæmon could conceivably carry out campus wide scheduling.

5. SLURM can provide all the information about instantaneous use of the cluster and assignment of CPU and memory on each node. This allows for stateless optimization and no need to maintain a copy of any configuration information.

6. SLURM itself is run as the privileged user, `slurm`, on our systems and most other systems. Even on SPYDUR where we lack `root` access, we have access to both running processes on behalf of the `slurm` user and the `installer` user, and the latter can impersonate ordinary ("LDAP") users.

7. *MetaBatch* will use the feature in SLURM to assign processes to specific nodes. Users who submit jobs will continue to submit the jobs to *partitions*, so no changes to user behavior will be required.

✧ ✧ End of Document ✧ ✧

George Flanagin, Provost Office
Academic Research Computing

Phone:	+1.804.287.6392
Address:	Richmond Hall, Office 104
	114 Richmond Way
	University of Richmond
	Richmond, VA 23173
Email:	gflanagin@richmond.edu
ORCID	0000-0002-2084-5831