Creating and Integrating A Linux Batch Processing Farm

David Boyes

Leland Lucius

2012 VM Workshop

Agenda

- Describe the Problem We Solved
- Outline the Solution We Used
- Show How the Solution Was Implemented
- Demonstration Output
- Q&A

Problem Statement

- In a lot of cases, traditional VM and z/OS workload can be augmented by Linux-based utilities
  - Examples: zipping up files for transfer, PDF conversion, etc.
  - These operations are often too CPU-expensive to perform on standard engines
- How can we set up a method for users to queue workload to a set of Linux images so that we distribute the workload fairly and consistently across the provided images?

Our Approach

- We wanted a batch queuing system that could take requests from multiple sources and assign them to a pool of virtual machines.
- Solution would need to:
  - Accept jobs from multiple sources
  - Allow managing a queue of jobs and pools of resources
  - Allocate jobs to execution queues
  - Return results to the submitter and notify them that the jobs were completed.

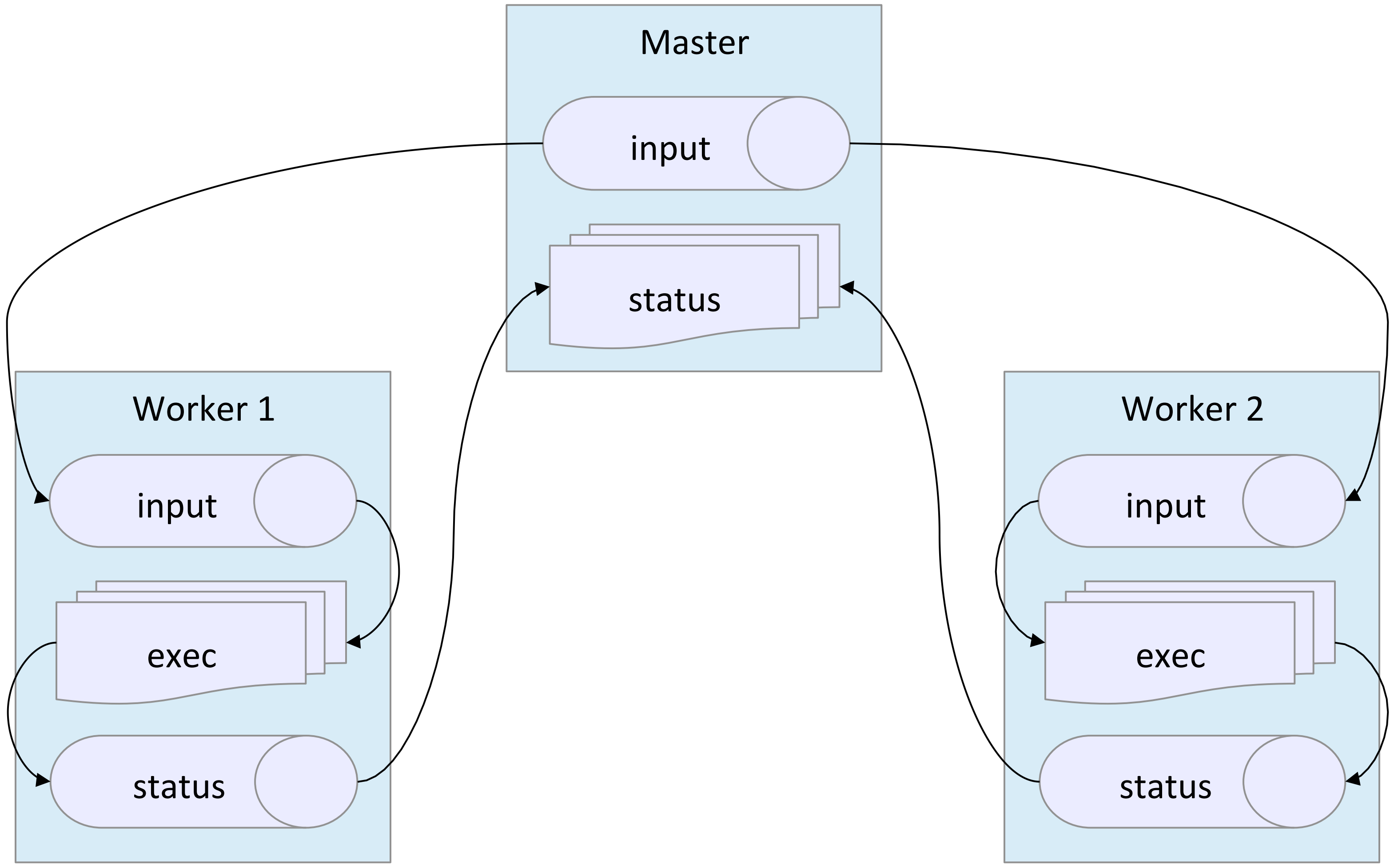
Our Solution

- Locate a pre-existing job queuing system that runs on Linux
  - We chose NQS because it was most similar to the way jobs and queues are managed in the mainframe world
  - Originally developed by Sterling Software, and widely used in the supercomputing community to get the most out of machines like Crays and Connection Machines.
  - Written in fairly portable C and shell scripts
  - Source code available to allow porting to System z without major fuss

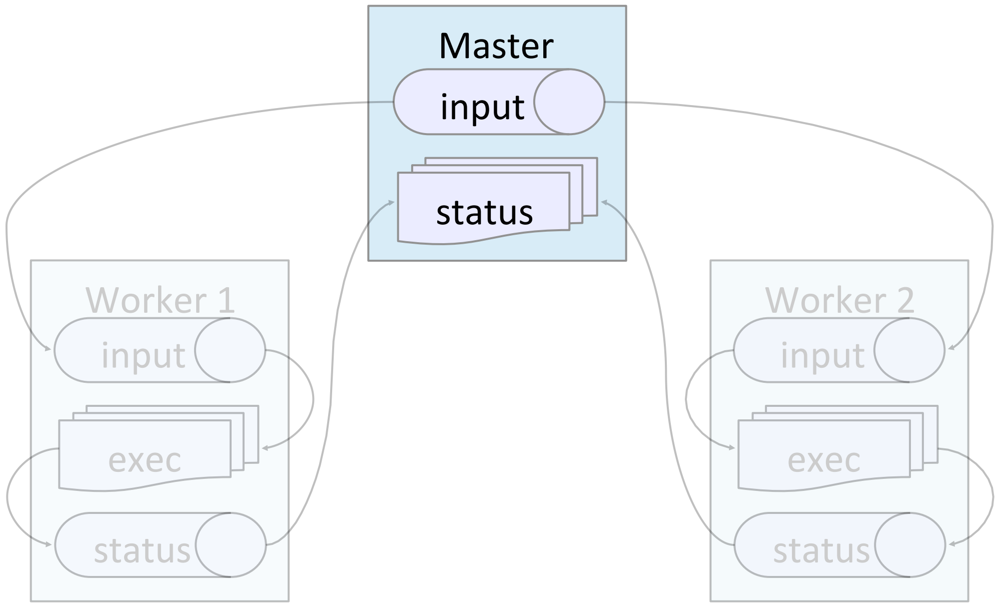
Demonstration Job Flow

- Job submission
- Job routing
- Job execution
- Output return
- Connection to NJE

Our Configuration



The Master



Where:

- Jobs are submitted
- No user jobs run
- Completion info is recorded

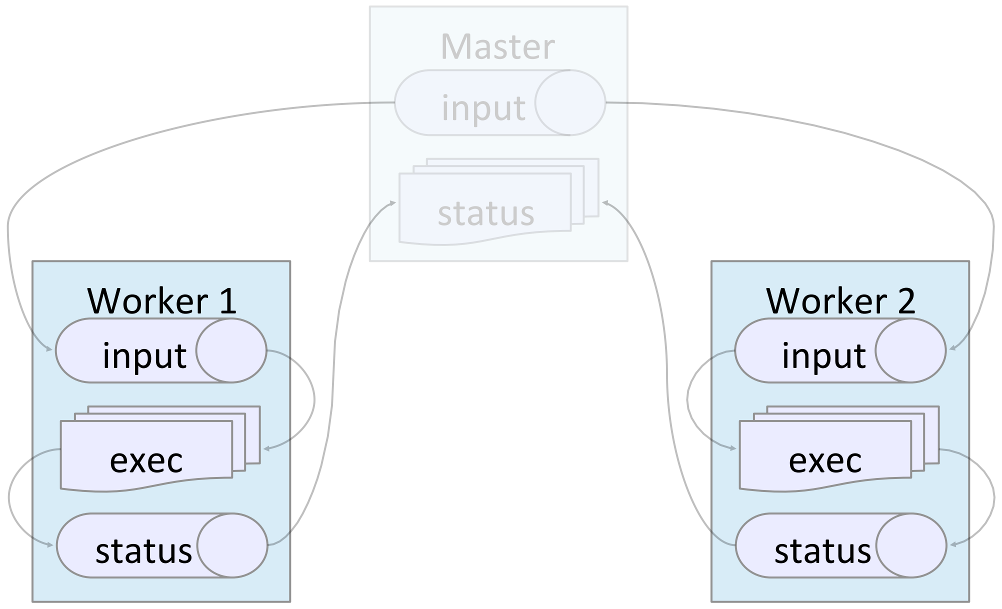
Jobs remain in the queue on Master until a Worker is available.

Jobs are forwarded to workers using a round robin scheduler. A scheduler based on resource usage is also available.

Jobs are transferred to the Worker and the Master no longer knows anything about the jobs.

As jobs end, a small status job is submitted to run on Master which records completion result and run times. These are recorded in a database for later analysis.

The Workers



Where:

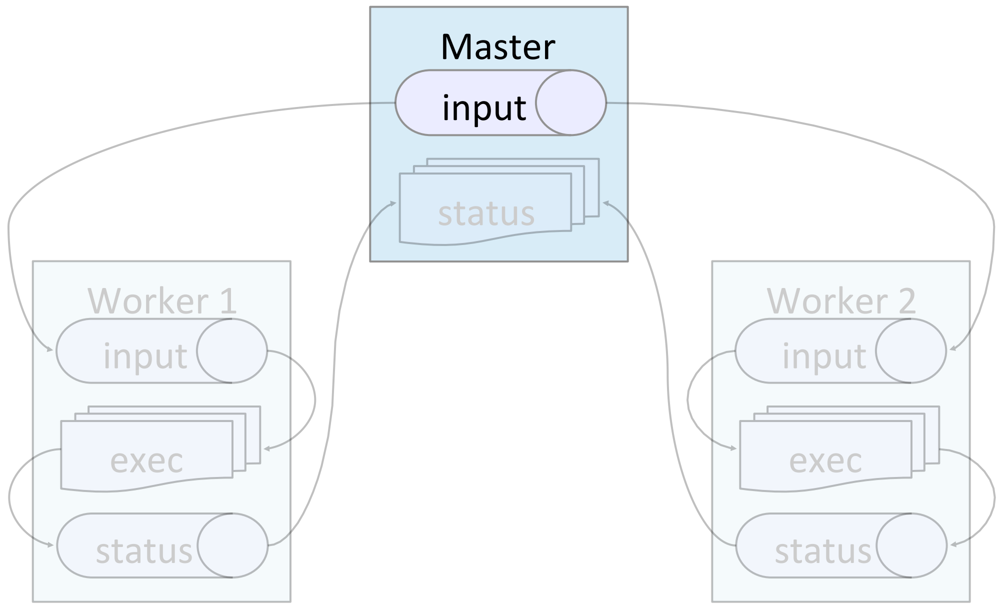
- User jobs execute
- Status reported to Master

In our configuration, the Worker input queues accept only one job at a time from Master and passes newly accepted jobs on to the exec queue.

Job output (stdout and stderr) is queued locally and returned to Master when job ends. This can be overridden if immediate output is required on Master.

In our configuration, job status is provided by the job itself. However, NQS provides user modifiable scripts to handle generic tasks like this.

Master Input



```
qmgr create pipe_queue input
qmgr set lb_out input
qmgr set dest=input@worker1 input
qmgr set dest=input@worker2 input
```

Accepts user submitted jobs.

Prioritizes queued jobs.

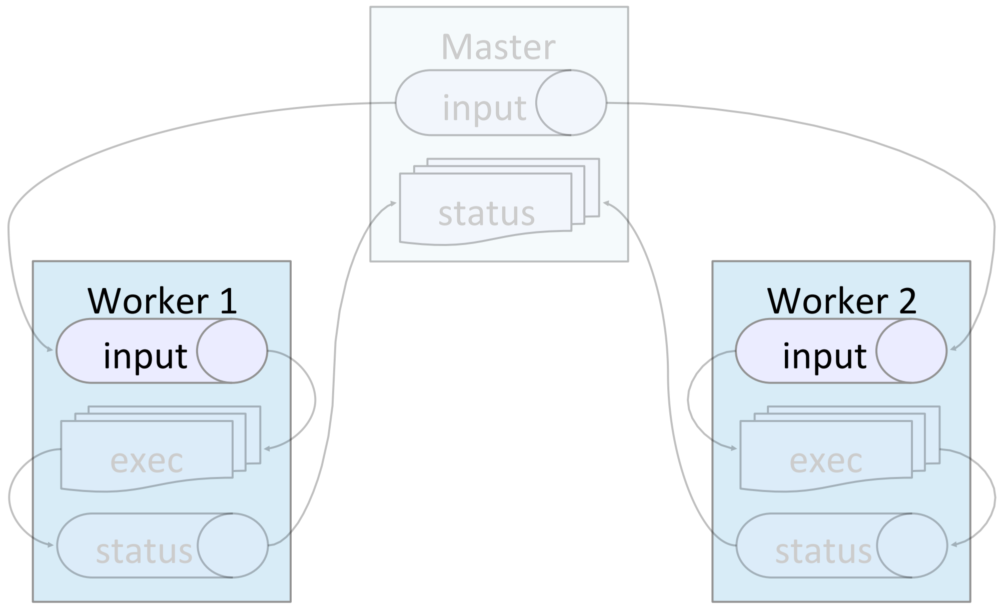
Schedules jobs on available destination queues.

Jobs are submitted to the Master input queue via:

```
qsub -q input <script name | stdin>
```

Plethora of qsub options to control job processing.

Worker Input



```
qmgr create pipe_queue input
qmgr set lb_in input
qmgr set dest=exec input
```

Accepts work from the Master input queue.

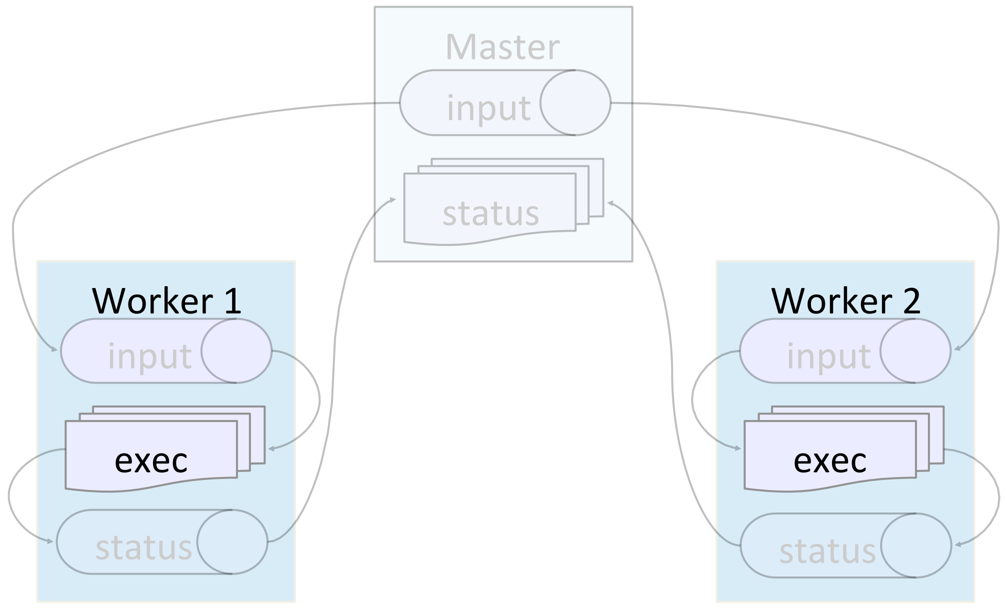
Blocks new jobs from Master if the local exec queue is busy.

Schedules jobs on local exec queue.

Jobs can be load balanced on multiple local batch queues if desired.

Jobs can come from multiple Master servers.

Worker Exec



```
qmgr create batch_queue exec
qmgr set pipeonly exec
```

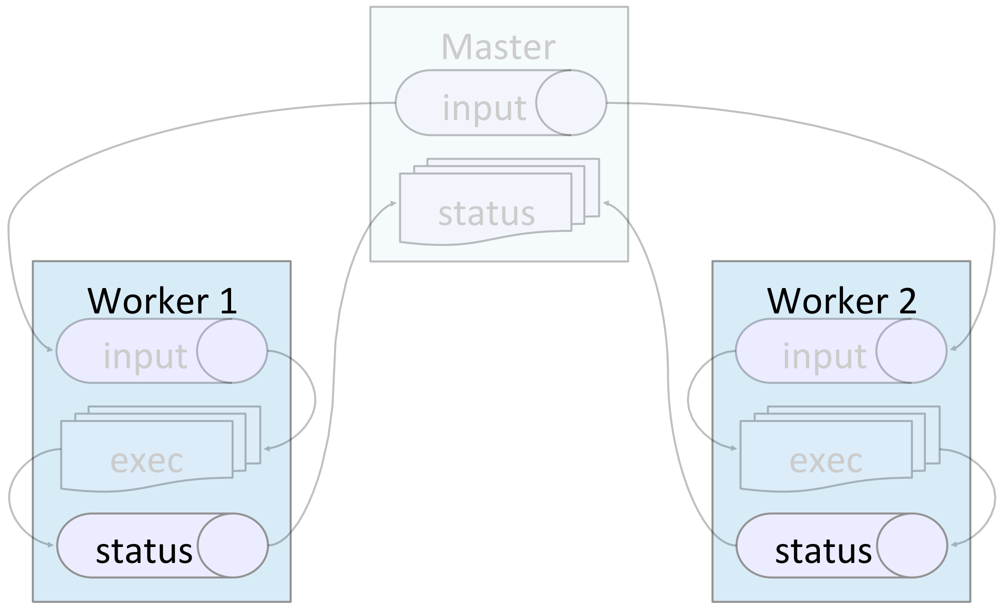
Accepts work from the local input queue only.

Only runs 1 job at a time.

The exec queue can be customized via shell scripts to capture job start and stop times, completion status, job triggering, etc.

By default, jobs are scripts and are executed as if they were run directly from the command line.

Worker Status



```
qmgr create pipe_queue status
qmgr set dest=status@master status
```

Accepts work from the local Worker exec queue.

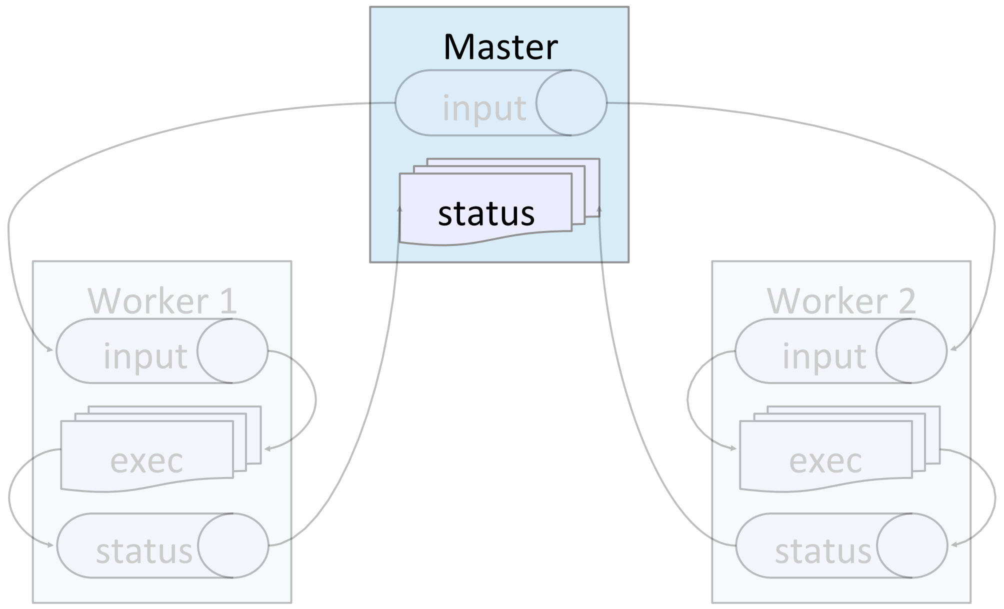
Forwards status jobs to Master status queue.

Is not load balanced like the Master input queue.

Is only used to send status jobs back to master.

Status jobs can originate from batch jobs or from the customized exec queue.

Master Status



```
qmgr create batch_queue status
qmgr set pipeonly status
qmgr set run_limit=32 status
```

Accepts work from the remote Worker status queues.

Can run multiple jobs at once.

Nothing actually makes it a “job status” queue.

Q&A