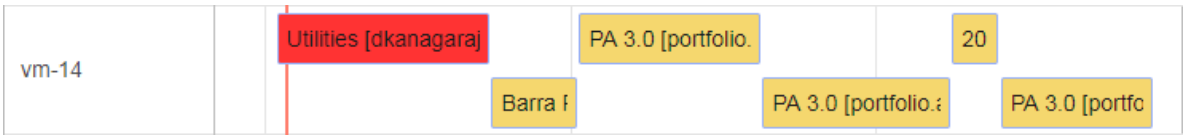
Tellus Scheduler Problem Statement

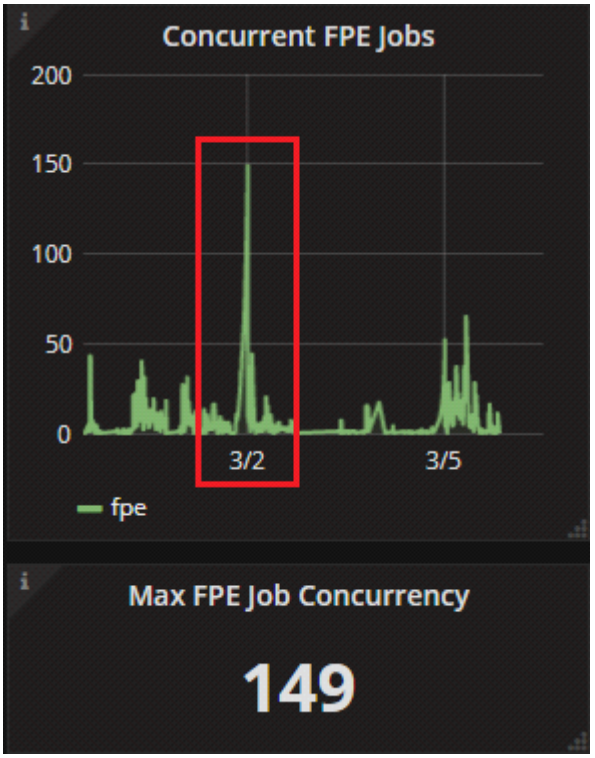
Tuesday, March 06, 2018 11:42 AM

The current Tellus Scheduler arranges job schedules according to job priority and resource availability calculations:



However, it has a design defect that it depends on *estimates* of job durations. When the actual job run time is longer than estimated, the next job would content with the current job for resources. And recently, jobs have been running much longer than estimates. Some of the reasons include: hung jobs, longer test scripts, longer NPM package installation time (more packages to install, for example), and virtual machines might be running slower because of larger number of sessions.

For example, in the following diagram, the Focal Point VM consumption reached 149, which is near the Focal Point limit, while the Tellus Scheduler was actually configured to limit it to 15:



As the algorithm of Tellus Schedule is complex, and a proper fix might involve re-implementing and moving the Scheduler into the legacy STAF Cron job scheduler, and it would be a large effort that essentially amounts to implementing another Kubernetes Scheduler, so we are probably better served to just go with Kubernetes. (Note: The creation of Tellus Scheduler was roughly contemporaneous with Kubernetes.)

In addition to queuing jobs for our custom (FocalPoint, Selenium Grid) resource availability, Kubernetes also queues up job for CPU and memory availability. That would help with the sharing of job CPU and memory resources, to avoid exhausting CPU and memory. That is a feature not available in Tellus Scheduler, and the short-term solution is a "custom load balancing" solution, as Dan suggested. It is

described below.

We also evaluated seven other job schedulers, but Kubernetes is preferred by the infrastructure team and enjoys strong industry momentum.

Short-Term Solution: Custom Load Balancing

Tuesday, March 06, 2018 1:36 PM

As Kubernetes infrastructure might be a few months away, we probably need a short-term solution. Dan Herman suggested a load balancing: round-robin among execution worker machines.

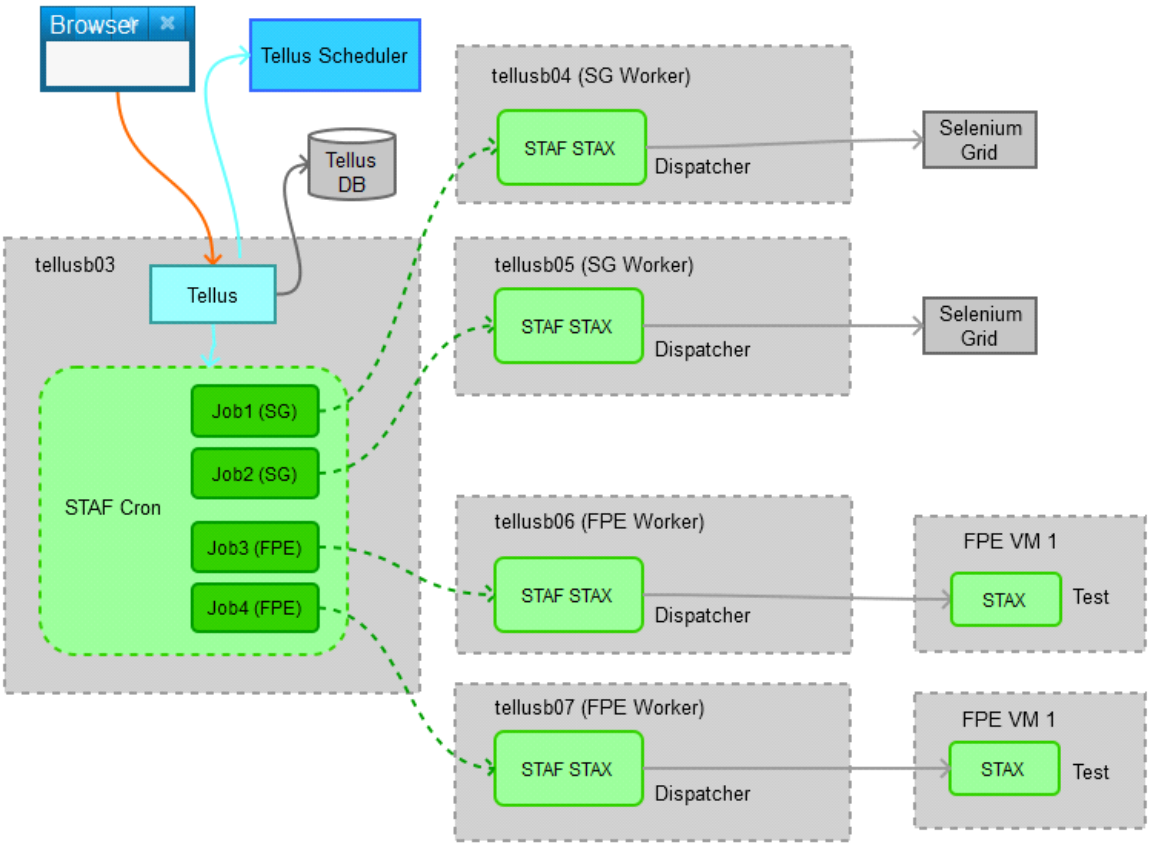


Figure. Short-term Solution: Tellus load balancing architecture with custom round-robin load balancing among multiple STAX workers. Tellusb03 only hosts STAF Cron (job scheduler). Job executions will be on STAF STAX "worker" machines (tellusb04, 5, 6, 7).

The figure above shows a "master" STAF Cron machine (tellusb03) dispatching jobs to multiple worker machines (tellusb04, tellusb05, tellusb06). The load balancing policy could be simple round robin. That would spread out the CPU and memory load.

Long-term Solution: Kubernetes

Tuesday, March 06, 2018 1:48 PM

The long-term solution would be a distributed resource-aware job scheduler like Kubernetes. The architecture is shown below.

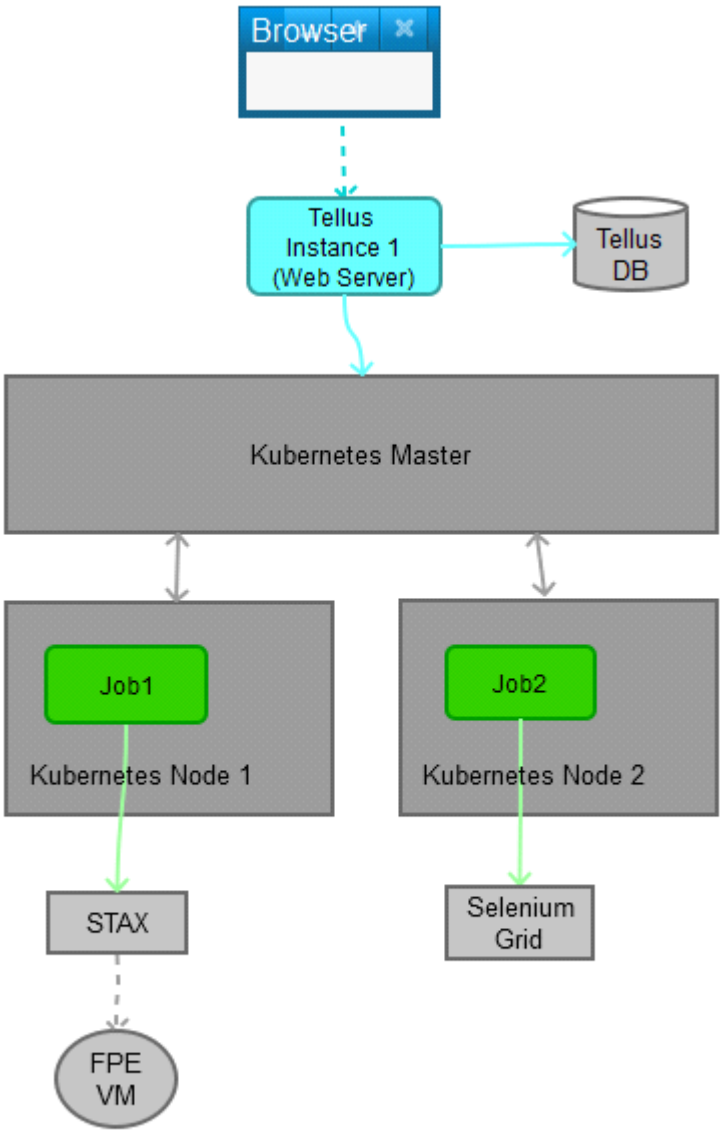


Figure. Tellus job execution architecture with Kubernetes.

It shows that jobs spread out to worker nodes. Kubernetes supports job priority, as well as sharing of CPU, memory, and custom (user-defined) resources, and it queues up jobs if resources are not available. Note: The STAX component could potentially be replaced by Dockers for Windows, when it becomes available (which completely replaces STAF).