Optimisation of PBS Hooks on the Cray XC40

Sam Clarke

May 2019

Overview

- About the Met Office workload
- An overview of our hook architecture
- Hook performance problems
- Problem characterisation
- Implementation of optimisations
- Assessment of performance improvements
- Discoveries

About the Met Office

- A global weather forecasting site
  - Many time critical forecasts run every day
  - Start and end times should not vary
  - Even small delays can cause problems
  - PBS reservations are used to guarantee resources
- An earth sciences research centre
  - Weather model development
  - Climate research
  - Fundamental physical and mathematical modelling
  - Collaborative research with UK academics engaged in environmental research

The Rose framework

- Almost all the work is run using Rose
  - A framework for running scientific applications
  - Provides a standard web-based interface
  - Manages user job output
  - Includes a database that allows jobs to be shared
  - Supports complex job networks (suites)
    - Suites are composed of a number of tasks
      - Tasks can have complex dependencies
      - Each task is a separate batch job
    - Suites usually cycle
      - Same tasks run against different data points

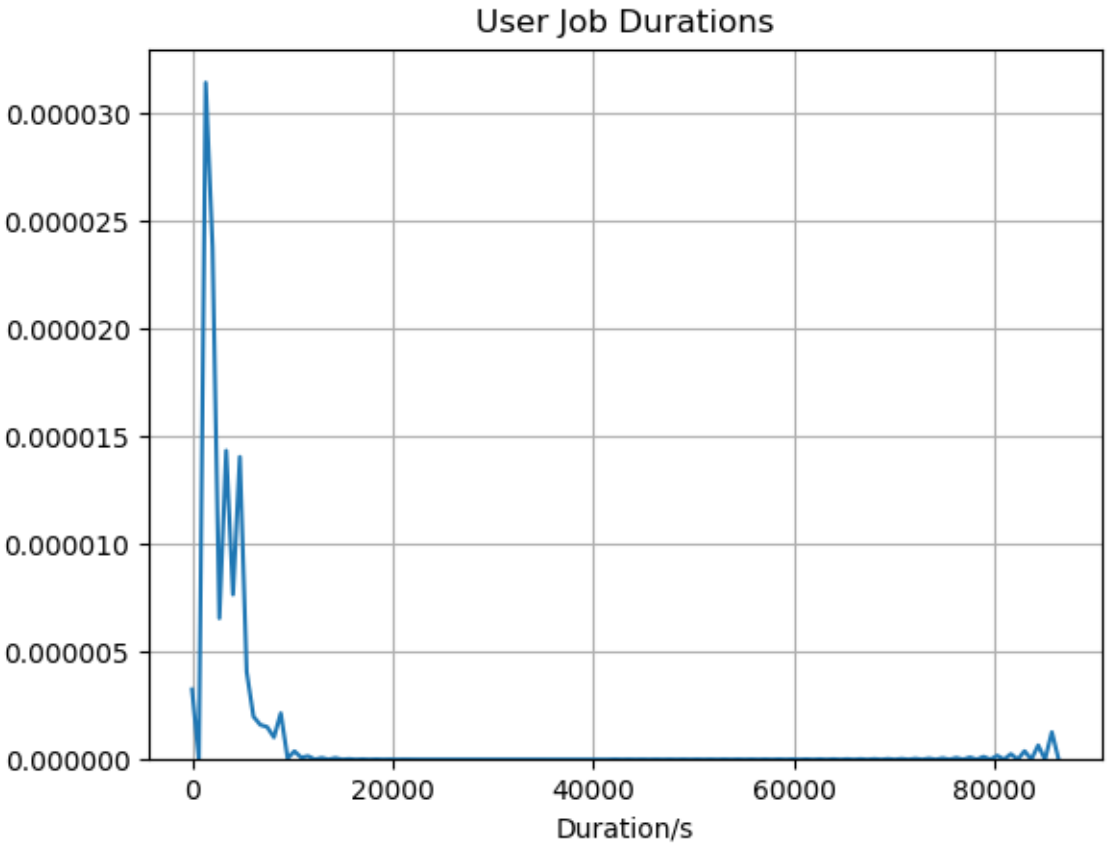
Rose Architecture

- Rose uses a client-server model
  - Tasks are spawned by the Rose server
    - Submitted to the Cray as PBS jobs
    - Qstat used to monitor the job state
  - Running jobs communicate with the server
    - Messages sent when a job start, completes, etc
- Rose runs in user space
  - Provides a standard web-based interface
  - There are no special hooks into the batch system
  - The implementation is batch software agnostic

About the Workload

- Largely homogenous due to the use of Rose
- Climate research jobs
  - Run on a smaller number of nodes
  - Composed of fewer tasks
  - Run for maximum time and resubmit
  - Timings for each job can vary by up to 10%
- Weather science jobs
  - Range of node counts – from one to a thousand
  - Complex suite of dependent tasks
  - Most jobs are very short
  - Timings are important when assessing changes for inclusion in next-generation forecasts

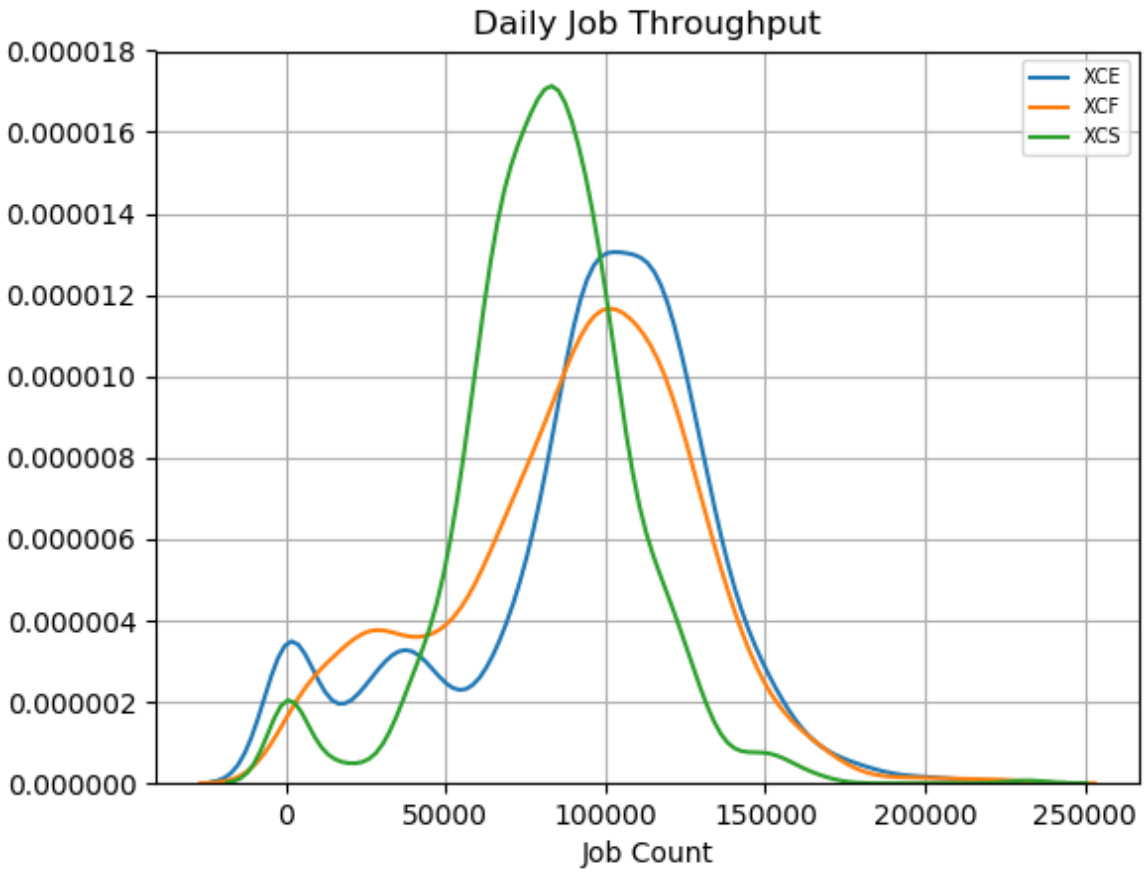
Average Job Durations



About the Jobs

- High job throughput
  - Averages are around 90-100,000 jobs per day
  - Peak throughputs as high as 230,000 jobs per day
- Short queue depths
  - Queues typically contain 300-400 jobs at any one time
- Workload is latent
  - Rose does not submit a job until the dependency has completed
  - The completion of a dependent job may result a flood of submissions
  - User job submissions can be triggered by the operational forecast suite

Daily Job Throughput



Hook Architecture: Submit hooks

- Run whenever a job arrives in the system
- Run on the PBS server
- Generally not time critical
  - Control is not returned to the user until all the hooks have completed
- We have a chain of submit hooks which:
  - Enforce queue resource rules
  - Confirm user projects
  - Enable our trustzone partitioning scheme
  - Support memory-resident temporary file systems

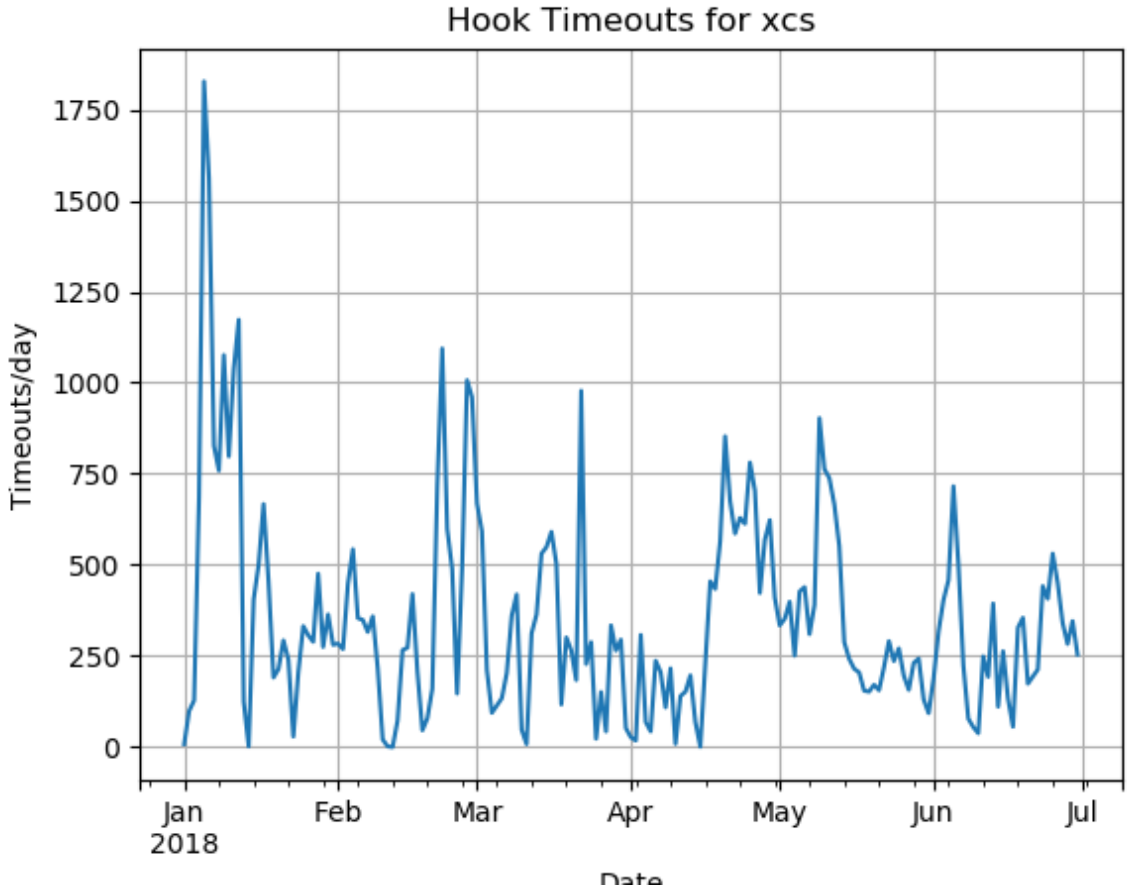
Hook Architecture: Execjob hooks

- Run at different points in the job lifecycle
- Run on the executing node
  - vntype=cray_serial for non-compute jobs
  - vntype=cray_login for compute/parallel jobs
- Time critical
  - User job will not start or end until hooks are complete
  - Time running hooks counts against the walltime
  - Compute resources are not released until job exits
- Important to system health
  - cgroup setup and removal
  - Trustzone setup and removal

Hook Timeouts

- Each hook has a timeout
  - Default is 30 seconds
  - Timeout events are logged
- We noticed a trend of increasing numbers of timeouts caused by
  - Increasing job throughput
  - Increased numbers of execjob hooks
  - Increased complexity and dependencies of hooks
    - Trustzone hook which carries out actions on the compute nodes
    - Cgroup hook which relies on OS activities
- Timeouts were clustered across nodes

Hook Timeout Trends



Hook Dependency Problems

- Each hook point runs a sequence of scripts
  - Site scripts and PBS hooks share the same sequences
- Scripts run in a defined order
  - Default is 1 but value is tuneable
  - Only one order is possible even if a hook runs from multiple points
- Timeouts are fatal to the entire sequence
  - All subsequent hooks are abandoned at the first timeout
  - Site hook timeouts can cause PBS system hooks to fail to run

User Output Problems

- Users were complaining about missing outputs
  - Job output and error streams were missing in Rose
  - Caused particular problems for users running short development suites
  - Lead to a lack of confidence in the system as a whole
- Caused by delays in job shutdown
  - Rose sends a message to the server when the user job script completes
  - Rose waits for PBS to spool the job output
  - The output takes longer than expected to stage
  - Rose times out waiting for the job output

Characterising the Problem

- A python decorator was added to every hook
  - Puts out a message when the hook starts
  - Reports the elapsed time when a hook completes
  - Logs a traceback and the contents of the current PBS event to the daemon logs when an error occurs
- Analysis of hook performance showed specific problems in scripts which accessed server data
  - Example: calling a library routine to determine whether the current queue is serial or parallel
  - Example: querying the node attributes of the current host to determine the trustzone

Characterising the Problem

- User output delays were not immediately caused by slow hooks or timeouts
  - They occurred in clusters on the same node
  - They did not appear to be caused by hook timeouts
- Problem was characterised by a stall on the executing node
  - PBS daemons logged a limited number of messages during the period
  - Process listings showed some zombie processes associated with the stuck jobs
  - All other processes continued to run as normal

Caching Optimisations

- Many slow operations were calls to a local library
  - Checking queue states and parameters was used particularly heavily
- Created routines to create and read cached copies of parameters
  - Data stored on node-specific /var file system
  - Cache files assigned a time-to-live to ensure dynamic data was current
  - Automatic conversion of multiple data values into python lists
- No top-level interface changes required
- Introduced with a restart of the PBS server

Caching Problems

- Caching of node data does not work in submit hooks
  - These execute on the SDB making the caches global
  - Cache will contain data from the last hook to run but time-to-live makes the value sticky
  - Explicit rules added to some hooks to avoid unpredictable behaviour
- Caching of dynamic data requires care
  - Problem with caching of queue data
  - Queues change rarely but reservations change often
  - Problems occur if a reservation is created and used in less than the time-to-live of the cache

Environment Optimisations

- Some server calls were for fixed job attributes
  - These were set in a submit hook and processed later
    - Size of memory resident temporary file system
    - End-of-job summary options
- Environment variables
  - Attributes were copied into environment variables
  - Variables were picked up and used by subsequent prologue and epilogue hooks
  - Server calls removed entirely
  - Variables can be changed by the job but hooks execute up-stream of any user scripts

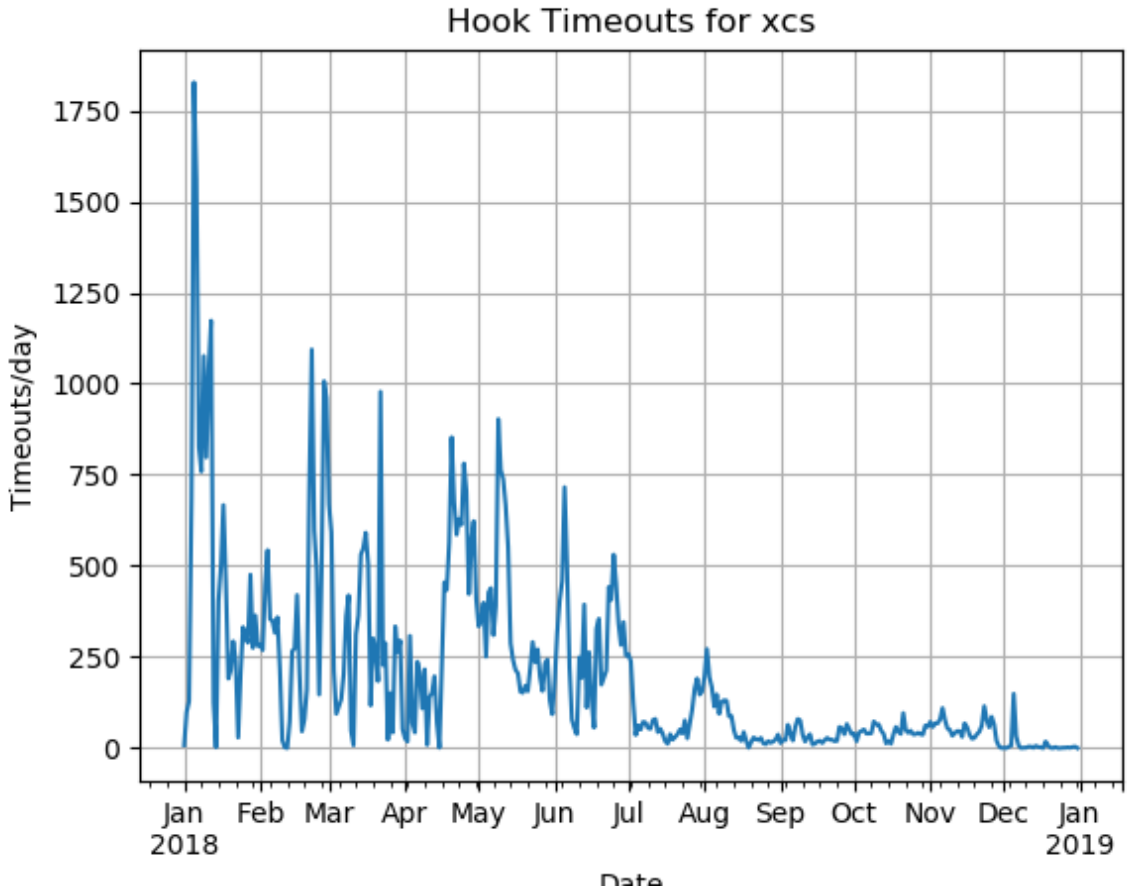
Removal of Start Time Estimator

- Separate process that estimates start times for each job
  - Puts a large strain on the server at the start of each run
  - Takes 15-20 minutes to run on our large systems
  - Typical job queue times are less than this
- Disabled on advice from Altair
- Users have not missed it

Results: Hook timeouts

- Substantial decrease in numbers of hook timeouts following changes
- Failures occurring in less critical hooks
- Remaining important hooks split and re-ordered
  - Cgroup hook was split into separate prologue and epilogue and assigned different orders
  - This works around a weekness in PBS where a single script is triggered from multiple points

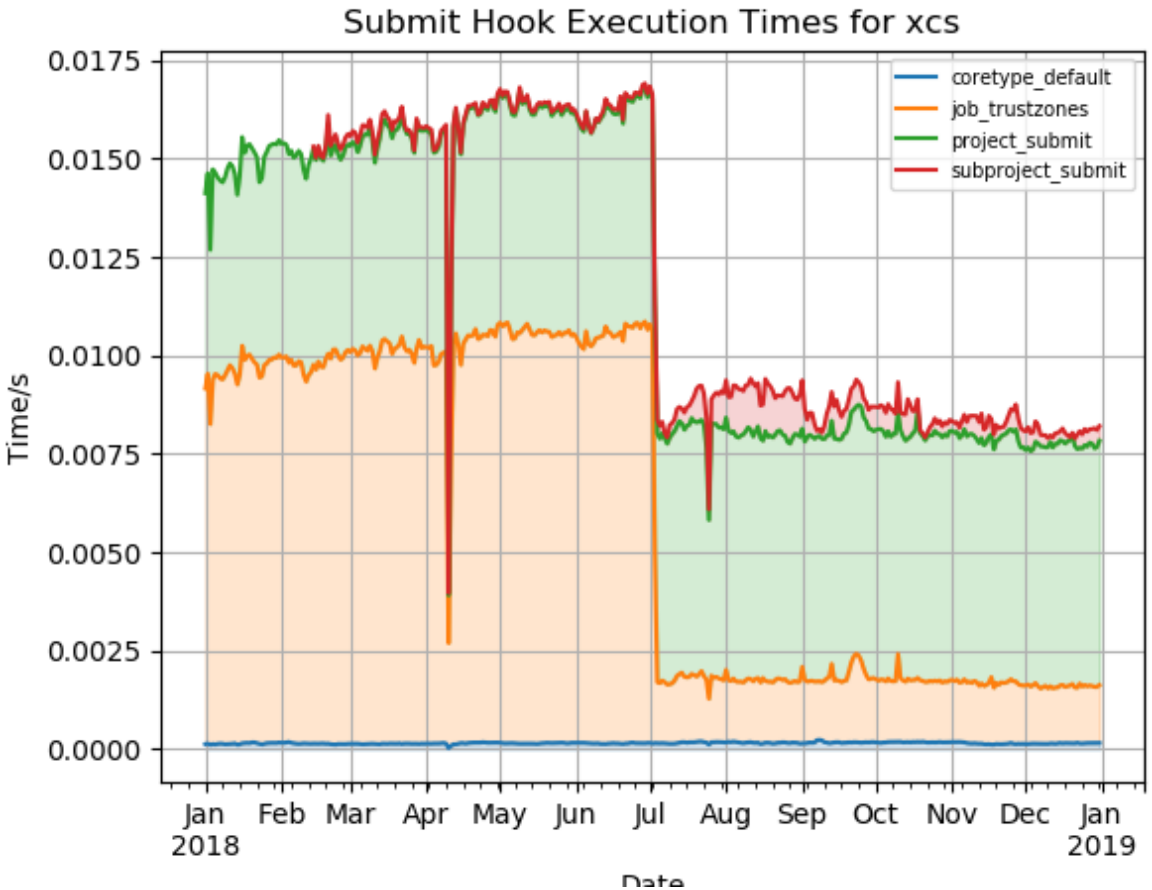
Results: Hook timeouts



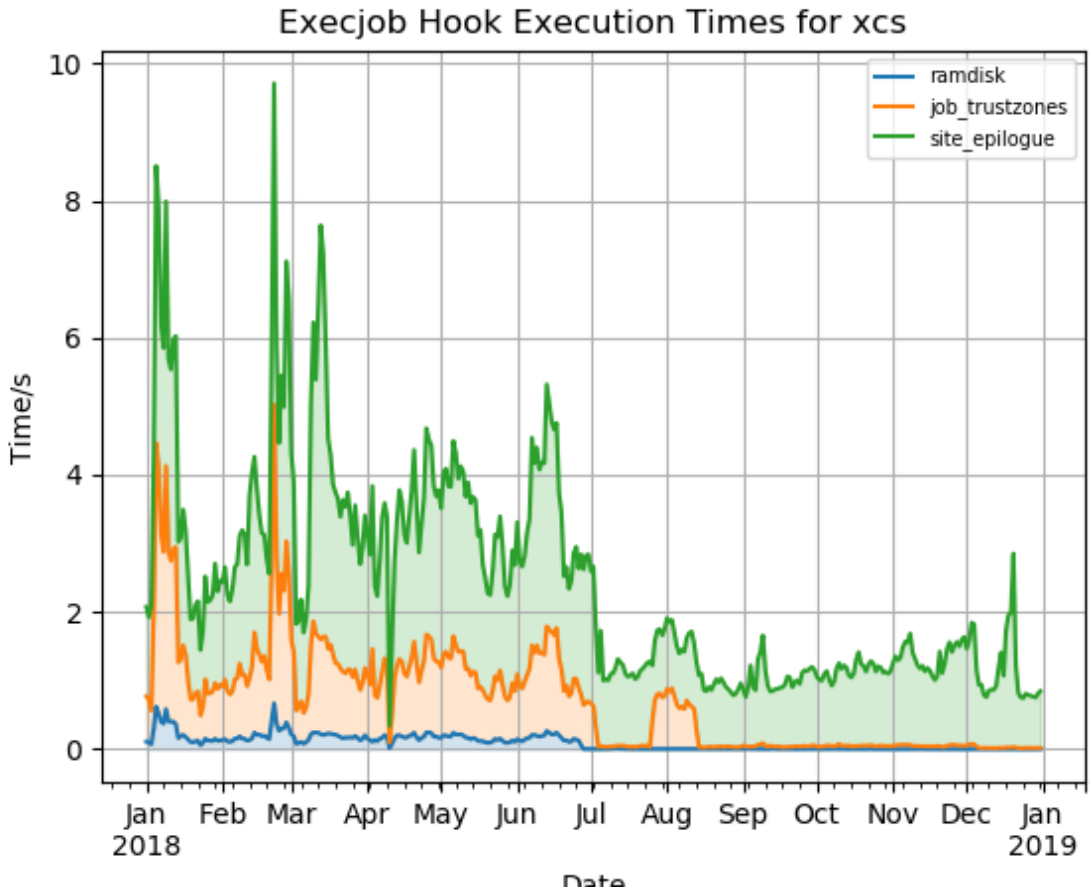
Results: Script performance

- Submit hooks much improved
  - Average times of trustzone hook are 75 per cent less
  - But: actual walltimes are very short!
- Improvements to execjob hooks more substantial
  - Trustzone hook performance also much better in execjob phases
  - Times down from multiple seconds to around 0.3 seconds
  - Some hooks down to almost nothing
    - Where a user has not requested memory resident temporary space, the hook cost has been reduced to a check on an environment variable

Submit hook improvements



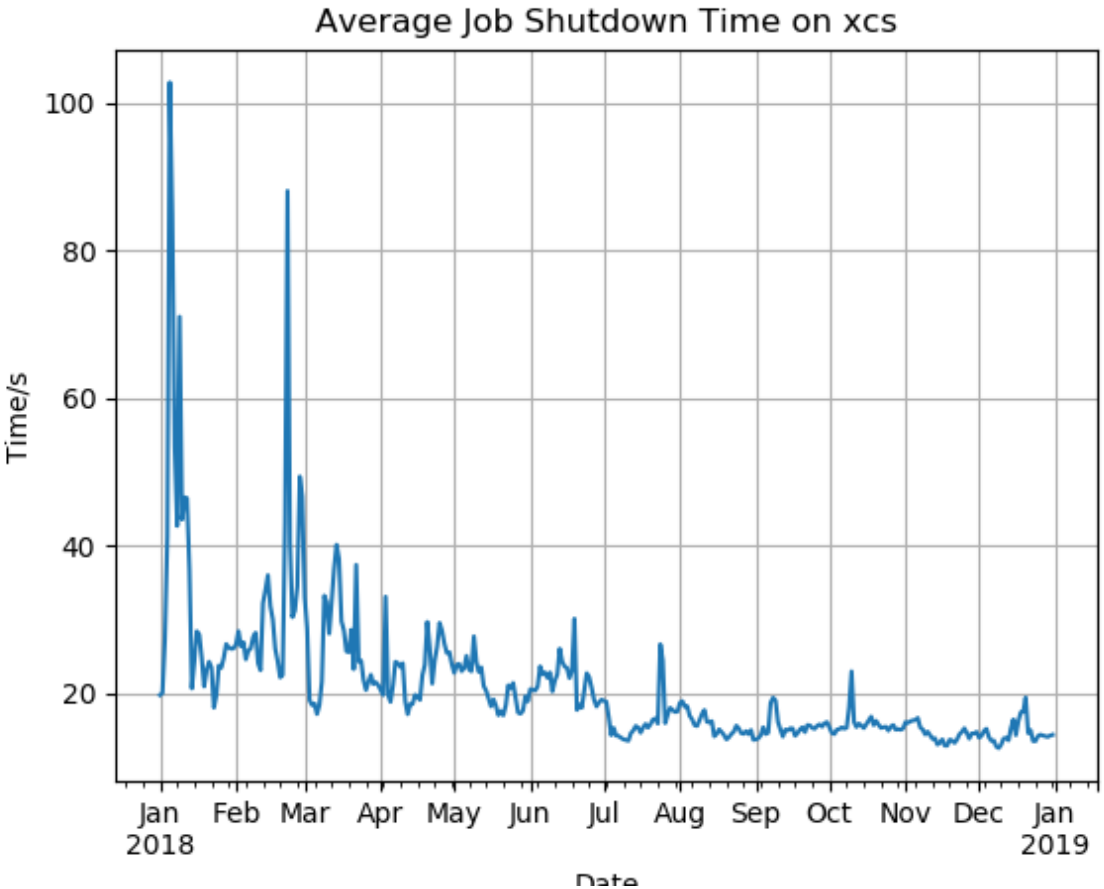
Execjob Hook Improvements



Results: Rose Output

- Slow output problems were found to be caused by the PBS daemons blocking:
  - An execjob hook would run and stall
  - Other jobs would exit while the hook was pending
    - These would block, waiting for the PBS daemon to clean up their shell processes and complete their epilogue actions
    - These delays were sufficient to cause the Rose stage-out action to time out
  - The execjob hook would timeout
  - The PBS daemon would resume normal execution
- Improving the hooks improved this problem too

Job Shutdown Performance



Conclusions

- Reducing server calls has improved hook performance across the board
  - Replacing a handful of local library routines made implementation straightforward
  - Some care required with time-to-live and file locations
- Number of alarm timeouts substantially reduced
- User problem with missing output resolved
- Useful discoveries made about the architecture and implementation of PBS hooks