JAliEn JobAgent

Maxim Storetvedt

Department of Computing, Mathematics, and Physics

April 18, 2018

Western Norway
University of
Applied Sciences

Background

- Plans to introduce multi-layered pilot with JAliEn
- Two layers:
  - First layer for job matching
  - Second prepares job files and runs payload
- Implemented by splitting the JobAgent into two components
  - Each on their own JVM

Motivation

- Can be used for SL6 compatibility on SL7
- Improves isolation
  - The JobAgent runs with token certificates for the JobAgent role
  - The payload will be executed with another token certificate specific for the job ID
    - Limits what a job can do based on its requirements
- Can be strengthened further by incorporating containers

Singularity

- Lightweight container platform
  - Very simple
  - Mostly a namespace wrapper
- Tailor made for the HPC use-case
  - Overlapping requirements with WLCG
  - Rapidly gaining popularity
- Useful as a mechanism for isolation
  - File isolation
  - Process isolation

Singularity

- Singularity is not just "a lightweight Docker"!
- Created with HPC in mind
  - Supports numerous resource managers
  - Multiple system architectures
  - Native support for using GPU resources
  - Single-file container images

Singularity

- Very different approach to isolation and user privileges
  - Singularity maintains the same user context as outside the container
  - Must be launched as root to gain root
  - File access is determined by launch wrapper code
  - User code can only interact with its own processes
- No UID switching required
- Light namespace footprint
- No need for services in the background

Singularity

- Isolation
  - File isolation
  - Process isolation
  - ~~Kernel isolation~~
- Job execution
  - Nonprivileged users

Singularity

- Let Job Agents use Singularity to separate workloads from other processes(?)
- What about Docker?
- Singularity is
  - Lightweight
  - Has no background services
  - Works without root
  - No installation/configuration required
  - Support for directory-based images

Singularity

- Questions in need of answers:
  - How to set up / configure Singularity for this use-case?
  - How to distribute / maintain image
- Moving towards:

Singularity

- Questions in need of answers:
  - How to set up / configure Singularity for this use-case?
  - How to distribute / maintain image
- Moving towards:
  - Sites not required to provide Singularity...

Singularity

- Questions in need of answers:
  - How to set up / configure Singularity for this use-case?
  - How to distribute / maintain image
- Moving towards:
  - Sites not required to provide Singularity...
  - . . . but should support it

Singularity

- Questions in need of answers:
  - How to set up / configure Singularity for this use-case?
  - How to distribute / maintain image
- Moving towards:
  - Sites not required to provide Singularity...
  - . . . but should support it
  - CVMFS

Singularity

- Requirements for running from CVMFS:
  - CVMFS
  - Linux kernel 3.10.0-693 or above (EL 7.4)
- Will otherwise need kernel capabilities only accessable to the root user
  - Must be installed with setuid-root executables
- Intended as a preview, but used in production

Running jobs in Singularity

- A read-only Singularity for job execution:
  - Setup comparable to OSG Singularity in CVMFS
  - Slight change in configurations
    - Overlay and bind-control enabled
    - Setuid disabled
  - CVM3 from CVMFS

JobAgent Progress

JobAgent Progress

- JAliEn JobAgent class split in two:
  - JobAgent
  - JobWrapper

JobAgent

- Has mainly two tasks
  - Get matched job
  - Launch JobWrapper
    - Generate launch command
    - Pipe over data
    - Provide monitoring

JobAgent

- Has mainly two tasks
  - Get matched job
  - Launch JobWrapper
    - Generate launch command
    - **Pipe over data**
    - Provide monitoring

JobAgent: Containers

- Initial support for launching the JobWrapper within both
  - Singularity
  - Docker
- Generate platform-specific launch command using an environment variable
  - USE_CONTAINERS=SINGULARITY
  - USE_CONTAINERS=DOCKER
- How to transfer data when JobWrapper runs in a container?

Creating a "pipe" (JobAgent)

```
final Process p;

OutputStream stdin;
OutputStream stdout;

ObjectOutputStream stdinObj;
ObjectOutputStream stdoutObj;

try {

   p = pBuilder.start();

   stdin = p.getOutputStream();
   stdinObj = new ObjectOutputStream(stdin);

   stdinObj.writeObject(jdl
   ...
```

Creating a "pipe" (JobWrapper)

```
try {
        inputFromJobAgent       = new ObjectInputStream(System.
        jdl  = (JDL)   inputFromJobAgent.readObject();
...
```

JobWrapper

- Responsible for job execution
  - Receive piped data from JobAgent
  - Set up execution environment
  - Run job
  - Save output
- In other words, most of the tasks previously handled by the JobAgent class

What remains...

- Identity handling in JobWrapper...?
- Prepare for production
  - Merge with recent commits
  - Thorough testing
  - Proper launch commands for containers

End

Thanks!

[Questions, comments]?

E-mail: msto@hvl.no