FirecREST: RESTful API on Cray XC systems

Felipe Cruz
Swiss National Supercomputing Centre, ETH Zurich, Lugano, Switzerland
felipe.cruz@cscs.ch
FirecREST, what is it and why it matters?

- What is FirecREST? a RESTful Services Gateway to HPC resources based on a microservices architecture.
  - RESTful interfaces. Architecture abstraction that allows for applications to be simple, lightweight, and fast.
  - Microservice architecture. Decouple functionalities in traditional monolithic servers architecture.
  - HPC resources. Integrates with infrastructure available at HPC centers:
    - Identity Access Management
    - Workload management
    - Data mover
FirecREST, what is it and why it matters?

- Why FirecREST? Allows web applications to securely access user-owned HPC resources
  - enforce that all API requests are authenticated
  - applications never manipulate user credentials
  - only allow requests from registered applications
  - user-managed access permissions per application
  - stateless security model by use of tokens
  - enables managing execution of workloads
  - enables external transfer of data from/to HPC filesystems
FirecREST motivation / use-cases
FirecREST architecture
FirecREST architecture components

Services
- Kong API gateway
- Compute
- Storage
- Utilities
- Support
  - Tasks
  - Status
  - Delegation

CSCS core services dependencies
- Workload Manager node
  - Compute
  - Xfer
- Utility node, filesystem utilities
- SWIFT, data transfer staging area
- Keycloak, authentication and authorization
FirecREST microservices

- **Kong**: Open-Source microservice API Gateway. Implements and **enforces** Authentication, Authorization, Traffic control, Analytics, Logging.
- **Compute**: Non-blocking calls to workload manager to submit and query jobs. The service responds with a reference to the temporary task resource that tracks the state of the request.
- **Storage**: Non-blocking calls to high-performance storage services. The service responds with a reference to the temporary task resource that tracks the state of the request.
FirecREST microservices

- **Utilities**: Fundamental filesystem utilities. All calls are blocking operations, maximum operation duration is limited by a timeout.
- **Support**
  - *Tasks*: state of tasks. Tracking of non-blocking compute and storage operations.
  - *Delegation*: Delegation with conversion from OIDC token to SSH user-certificate.
  - *Status*: information on services and infrastructure.
FirecREST core workflows
FirecREST workflow: Authentication
FirecREST workflow: Job submission

Client

HTTPS GET -H <token> 
firecrest/jobs/jobid

Return: task_id

:Compute

HTTPS PUT -H <token> 
firecrest/tasks

Return: task_id

:Tasks

HTTPS GET -H <token> 
firecrest/tasks

Return: public_key, private_key, SSH Certificate

Asynchronous call: scp sbatch_file /remote/path/sbatch_file 
sbatch /remote/path/sbatch_file

Return: job_id

:Delegation

HTTPS POST -H <token> 
firecrest/tasks/task_id

Return: task_status

<system>
FirecREST workflow: Download data

Client

HTTPS GET -H <token> firecrest/xfer-external/upload

Return: task_id

:Storage

HTTPS PUT -H <token> firecrest/tasks

Return: task_id

:Tasks

HTTPS GET -H <token> firecrest/certificate -d system

Return: public_key, private_key, SSH Certificate

Python swiftclient upload to staging area

Upload file

Return: upload completed

:Delegation

Return: file, container, account

Python swiftclient upload to staging area

Return: file, container, account

<system>

HTTPS POST -H <token> firecrest/tasks/task_id

Return: task_finished
FirecREST workflow: Upload data

- **Client**
  - HTTPS GET -H `<token>`
    - firecrest/xfer-external/upload
  - Return: task_id

- **:Storage**
  - HTTPS POST -H `<token>`
    - firecrest/tasks
  - Return: task_id
  - HTTPS POST -H `<token>`
    - firecrest/tasks/task_id
  - Return: task_status
  - HTTPS GET -H `<token>`
    - firecrest/tasks/task_id
  - Return: upload_temp_url information

- **:Tasks**
  - HTTP POST: Upload object to staging area
  - Return: completed upload
  - HTTPS GET -H `<token>`
    - firecrest/xfer-external/upload-completed

- **Swift**
  - HTTPS GET -H `<token>`
    - firecrest/certificate -d system
  - Return: public_key, private_key, SSH Certificate
  - Web download from temporal staging area
  - Return: download completed
  - HTTPS POST -H `<token>`
    - firecrest/tasks/task_id
  - Return: task_finished

- **:Delegation**

- **<system>**
FirecREST API overview
Non-blocking calls to workload manager to submit and query jobs. The service responds with a reference to the temporary task resource that tracks the state of the request.

**POST** /Jobs/{machine} Submit Job

**GET** /Jobs/{machine} Retrieves information from all jobs

**GET** /Jobs/{machine}/{jobid} Retrieves information from a job

**DELETE** /Jobs/{machine}/{jobid} Delete Job

**GET** /Jobs/{machine}/sacct Job account information
Storage

Non-blocking calls to high-performance storage services. The service responds with a reference to the temporary task resource that tracks the state of the request.

GET /Storage/xfer-internal/rsync rsync

GET /Storage/xfer-internal/mv move (rename) files

GET /Storage/xfer-internal/cp copy files and directories

GET /Storage/xfer-internal/rm remove files or directories

GET /Storage/xfer-external/upload Upload a file

GET /Storage/xfer-external/download Download a file
Queue  Access status and response of compute and storage tasks.

GET  /Queue/tasks  returns all tasks

PUT  /Queue/tasks  Creates a task

GET  /Queue/tasks/{id}  task status information

POST  /Queue/tasks/{id}  Updates a task

DELETE  /Queue/tasks/{id}  Delete task
**Utilities**

Basic system utilities. All calls are blocking and low-latency operations, maximum operation duration is limited by a timeout.

<table>
<thead>
<tr>
<th>Method</th>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/utilities/{machinename}/ls</td>
<td>List directory contents</td>
</tr>
<tr>
<td>POST</td>
<td>/utilities/{machinename}/mkdir</td>
<td>Creates a directory</td>
</tr>
<tr>
<td>POST</td>
<td>/utilities/{machinename}/rename</td>
<td>Rename/move a file, directory, or symlink</td>
</tr>
<tr>
<td>POST</td>
<td>/utilities/{machinename}/chmod</td>
<td>Change file mode bits</td>
</tr>
<tr>
<td>POST</td>
<td>/utilities/{machinename}/chown</td>
<td>Change file owner and group</td>
</tr>
<tr>
<td>POST</td>
<td>/utilities/{machinename}/file</td>
<td>Determine file type</td>
</tr>
<tr>
<td>POST</td>
<td>/utilities/{machinename}/symlink</td>
<td>Create a symlink</td>
</tr>
</tbody>
</table>
Thank you