Bringing the ATLAS HammerCloud setup to the next level with containerization

Benjamin Rottler, Michael Böhler, Günter Duckeck
Alexander Lory, Christoph Anton Mitterer,
Jaroslava Schovancova

on behalf of the ATLAS Computing Activity

CHEP 2023 – Norfolk, Virginia, USA
09 May 2023
HammerCloud at a glance

- framework to test, commission, benchmark WLCG resources
  - testing with **full chain** jobs
    - same environment as the “real” analysis/production jobs

- part of the ATLAS distributed computing operations (ADC Ops) automation suite
  - automatic exclusion & recovery of the resources
  - spot system-wide outages & alerting
  - spot issues with the next generation clients / environments
  - commission new resources & distributed system components
  - benchmark resources & workflows

→ **http://hammercloud.cern.ch/hc/app/atlas/**

Benjamin Rottler
CHEP 2023 – ATLAS HammerCloud Containerization 09 May 2023
Automated testing of grid sites with HammerCloud

- **Template 1**
  - Test 1
  - Site A: Job 1, Job 2, ..., Job i
  - Site B: Job 1, Job 2, ..., Job i
  - Site C: ...

- **Template 2**
  - Test 2
  - Site A: Job 1, Job 2, ..., Job j
  - Site B: Job 1, Job 2, ..., Job j

- **Template 3**
  - Test 3
  - Site A: Job 1, Job 2, ..., Job k
  - Site B: Job 1, Job 2, ..., Job k

- **Template** defines test type (sites, workload, auto-exclusion)
- **Test runs** 24 hours on all sites defined in the template
- **Jobs are continuously** sent to each site until the end of the test
HammerCloud use cases

- spots issues with infrastructure
- helps with commissioning and R&D
  - resources
  - components of distributed systems
  - new approaches to data access/management
  - GPUs
- in total 60k+ jobs per day
Architecture of HammerCloud

- Core node:
  - Certificate renewal
  - CRIC\(^1\) collector
  - Dataset cache creation
  - Queue injection
  - Auto exclusion
  - Test assignment

- Web server:
  - Django
  - Static file hosting

- Submission nodes:
  - Certificate renewal
  - Test generation
  - Test submission & monitoring

\(^1\)Computing Resource Information Catalog

- three types of nodes: core (1x), web server (1x), submission nodes (multiple)
Workflow: test submission

Core node
- Certificate renewal
- CRIC\(^1\) collector
- Dataset cache creation
- Queue injection
- Auto exclusion
- Test assignment

Database

Web server
- Django
- Static file hosting

Submission nodes
- Certificate renewal
- Test generation
- Test submission & monitoring

HC Operator

\(^1\) Computing Resource Information Catalog
\(^2\) Production and Distributed Analysis – workload management system
Workflow: test submission

1. +2. create test

1. CRIC\(^1\) collector
2. Test generation
3. assign test to submission node
4. +5. generate config files for PanDA jobs
5. submit and monitor PanDA jobs

\(^1\) Computing Resource Information Catalog
\(^2\) Production and Distributed Analysis – workload management system
Workflow: test submission

1. create test
2. assign test to submission node
3. generate config files for PanDA jobs
4. submit and monitor PanDA jobs

1. Certificate renewal
2. CRIC\(^1\) collector
3. Dataset cache creation
4. Queue injection
5. Auto exclusion
6. Test assignment

Web server
- Django
- Static file hosting

Database

Submission nodes
- Certificate renewal
- Test generation
- Test submission & monitoring

Core node

1. Certificate renewal
2. CRIC\(^1\) collector
3. Dataset cache creation
4. Queue injection
5. Auto exclusion
6. Test assignment
7. Test submission & monitoring

PanDA\(^2\)
Workflow: test submission

1.+2. create test
3. assign test to submission node
4.+5. generate config files for PanDA jobs

1. Core node
   - Certificate renewal
   - CRIC\(^1\) collector
   - Dataset cache creation
   - Queue injection
   - Auto exclusion
   - Test assignment

2. Database

3. Web server
   - Django
   - Static file hosting

4. Submission nodes
   - Certificate renewal
   - Test generation
   - Test submission & monitoring

5. HC Operator

\(^1\) Computing Resource Information Catalog
\(^2\) Production and Distributed Analysis – workload management system

Benjamin Rottler
CHEP 2023 – ATLAS HammerCloud Containerization
09 May 2023
Workflow: test submission

1. create test
2. Test submission
3. assign test to submission node
4. +5. generate config files for PanDA jobs
5. Test submission & monitoring
6. +7. submit and monitor PanDA jobs
## Containerization of HammerCloud

### Current setup
- Python 2.7 / Django 1.1
- no containerization – binaries/libraries depend on host operating system → are shared between apps/services
- HammerCloud software distributed via RPMs
- dedicated machines needed for development

### Containerized setup
- migrate codebase to Python 3.8 / Django 4.0 (almost done)
- containerization via Docker
  - services can run in their own environment
  - more agile development process
    → local development on laptop possible
    → application of CI/CD workflow
- long term goal: streamline commissioning of HPC sites
  → provide containerized HammerCloud setup
  → HPC sites can test their setup with a local HammerCloud instance
Containerization of HammerCloud

Current setup
- Python 2.7 / Django 1.1
- no containerization – binaries/libraries → depend on host operating system → are shared between apps/services
- HammerCloud software distributed via RPMs
- dedicated machines needed for development

Containerized setup
- migrate codebase to Python 3.8 / Django 4.0 (almost done)
- containerization via Docker → services can run in their own environment
- more agile development process → local development on laptop possible → application of CI/CD workflow
- long term goal: streamline commissioning of HPC sites → provide containerized HammerCloud setup → HPC sites can test their setup with a local HammerCloud instance
Containerized architecture

- Core node:
  - Certificate renewal
  - CRIC\(^1\) collector
  - Dataset cache creation
  - Auto exclusion
  - Test assignment

- Web server:
  - Django
  - Static file hosting

- Submission nodes:
  - Certificate renewal
  - Test generation
  - Test submission & monitoring

- docker philosophy: use separate container for each service

\(^1\)Computing Resource Information Catalog
Local development setup – web server

- Local container orchestration with `docker compose`
- Include config and static files via bind mounts → persistent storage on host system
- Allows one to operate the entire web server on a laptop
HammerCloud website running locally on laptop

<table>
<thead>
<tr>
<th>State</th>
<th>Id</th>
<th>Host</th>
<th>Template</th>
<th>Start (Europe/Zurich)</th>
<th>End (Europe/Zurich)</th>
<th>Sites</th>
<th>submit jobs</th>
<th>run jobs</th>
<th>comp jobs</th>
<th>fail jobs</th>
<th>fail %</th>
<th>tot jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>running</td>
<td>20252624</td>
<td>hammercloud-ai-74</td>
<td>1184: PFT mc15 Sim_tf 21.0.15 FC</td>
<td>21/Nov, 7:32</td>
<td>22/Nov, 5:50</td>
<td>AGLT2_TEST, BEIJING, GoGrid, 162 more...</td>
<td>102</td>
<td>90</td>
<td>5890</td>
<td>1273</td>
<td>17</td>
<td>7355</td>
</tr>
<tr>
<td>running</td>
<td>20252627</td>
<td>hammercloud-ai-77</td>
<td>952: AFT EventLoop 21.2.1 Analy</td>
<td>21/Nov, 8:16</td>
<td>22/Nov, 7:23</td>
<td>ANALY_ARNES_DIRECT, ANALY.SignET_DIRECT, ANALY.CERN_TO_ART, 108 more...</td>
<td>76</td>
<td>22</td>
<td>8460</td>
<td>1018</td>
<td>11</td>
<td>9576</td>
</tr>
<tr>
<td>running</td>
<td>20252638</td>
<td>hammercloud-ai-79</td>
<td>1192: PFT mc21 Sim_tf 22.0.73</td>
<td>21/Nov, 15:14</td>
<td>22/Nov, 14:13</td>
<td>CERN, UNI FREIBURG, AGLT2, 162 more...</td>
<td>52</td>
<td>86</td>
<td>1963</td>
<td>333</td>
<td>14</td>
<td>2434</td>
</tr>
<tr>
<td>running</td>
<td>20252640</td>
<td>hammercloud-ai-78</td>
<td>1013: AFT AthDerivation 21.2.33.0</td>
<td>21/Nov, 16:44</td>
<td>22/Nov, 17:38</td>
<td>ANALY_ARNES_DIRECT, ANALY.SignET_DIRECT, ANALY.CERN_TO_ART, 108 more...</td>
<td>46</td>
<td>53</td>
<td>2358</td>
<td>186</td>
<td>7</td>
<td>2643</td>
</tr>
<tr>
<td>running</td>
<td>20252645</td>
<td>hammercloud-ai-74</td>
<td>1169: GPU Container GPU Available + Vector Multiplication</td>
<td>22/Nov, 0:08</td>
<td>23/Nov, 0:58</td>
<td>ANALY_MANC_GPU, ANALY.BNL_GPU_ARC, ANALY.SLAC_GPU, 4 more...</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>41</td>
<td>17</td>
</tr>
<tr>
<td>running</td>
<td>20252646</td>
<td>hammercloud-ai-77</td>
<td>883: AFT PlottingJobOptions_ExampleCode 21.0.8</td>
<td>22/Nov, 0:46</td>
<td>23/Nov, 2:09</td>
<td>ANALY_ARNES_DIRECT, ANALY.SignET_DIRECT, ANALY.CERN_TO_ART, 108 more...</td>
<td>65</td>
<td>27</td>
<td>241</td>
<td>14</td>
<td>4</td>
<td>347</td>
</tr>
</tbody>
</table>
minimal setup to submit HammerCloud jobs from a local machine (e.g. personal laptop) to PanDA

1 **Computing Resource Information Catalog**  
2 **Production and Distributed Analysis** – workload management system  
3 **Data management tool**
Local development setup – test submission

- **CRIC**
- **Rucio**
- **Django web server**
- **NGINX**
- **Database**
- **Config files**
- **Static files**
- **Test generation**
- **Test submission & monitoring**

- **Core services**
  - **CRIC** collector
  - Dataset cache creation
  - Test assignment

- **Submission services**
  - **Django web server**
  - Test generation
  - Test submission & monitoring

- **Powered by**
  - MySQL

- **Powered by**

- **Reverse proxy**
  - Host static files

- **Local machine**

---

**minimal setup to submit HammerCloud jobs from a local machine (e.g. personal laptop) to PanDA**

1 **Computing Resource Information Catalog**

2 **Production and Distributed Analysis – workload management system**

3 **Data management tool**

---

Benjamin Rottler

CHEP 2023 – ATLAS HammerCloud Containerization

09 May 2023
Local development setup – test submission

1. + 2. create test
3. assign test to submission node
4. + 5. generate config files for PanDA jobs
6. + 7. submit and monitor PanDA jobs

minimal setup to submit HammerCloud jobs from a local machine (e.g. personal laptop) to PanDA

1. Computing Resource Information Catalog
2. Production and Distributed Analysis – workload management system
3. Data management tool

Benjamin Rottler
CHEP 2023 – ATLAS HammerCloud Containerization
09 May 2023
Conclusion & outlook

**Containerization of HammerCloud**

- more than 80% of the code base has been migrated successfully to Python 3.8 / Django 4.0
- developed a suitable container architecture for HammerCloud infrastructure
  - this allows local development
  - more agile code deployment and CI
- web server completely containerized

**Outlook**

- migrate remaining HammerCloud components to Python 3.8 / Django 4.0
- containerize remaining services
- put containerized setup into production
- develop CI/CD workflow