Testing and Improving Deployment of ATLAS Releases to CVMFS

Tomáš Stefan
$ whoami
$ whoami

Student between bachelors and masters studies

Enthusiast in:

• computer security
• open source
• Rust
Prague, Czech Republic
Main task
Main task

Compare ATLAS software (Athena) publication to CVMFS in various conditions:

• current approach vs. simultaneous publications
• physical machines vs. virtual machines
• different backends

Why Athena? – It is big enough & real world scenario
What is the **CVMFS**?

- POSIX read-only file system in user space (a FUSE module)
- way to **deploy** software on the **worldwide** distributed computing infrastructure
- using **HTTP** allows integration of existing solutions for aggressive caching and reduces firewall-related problems
Main task - First approach

I got access to **five physical servers** and **four virtual servers**

I needed to setup everything **by hand** on **each** machine before **each** test, and then run shell script with measured publication

That was **terribly slow** process
Main task - Improvements

The setup before each test was almost always the same and very time consuming.

I decided to move to an Ansible for automated process. Now it is faster and much more reliable.

Now the initial phase lasts a bit over an hour:
- download the releases
- setup CVMFS gateway, release managers, ...

And the cycle of setup + test is completed in about six hours.
Main task - Results I

Local disk installation
Athena_22.0.1_x86_64-centos7 (master)

Time (s)

<table>
<thead>
<tr>
<th>Physical Machines</th>
<th>Virtual Machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>814</td>
<td>1253</td>
</tr>
<tr>
<td>813</td>
<td>1093</td>
</tr>
<tr>
<td>811</td>
<td>1178</td>
</tr>
<tr>
<td>809</td>
<td></td>
</tr>
</tbody>
</table>
Main task - Results II

local CVMFS installation + publication
Athena_22.0.1_x86_64-centos7 (master)

![Bar chart showing time differences between physical and virtual machines.](chart.png)
Main task - Results III

**local CVMFS publication vs ingestion**
Athena_22.0.1_x86_64-centos7 (master)

<table>
<thead>
<tr>
<th></th>
<th>publish</th>
<th></th>
<th>ingest</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cvm-perf04</td>
<td>135</td>
<td>cvm-perf07</td>
<td>153</td>
<td>cvm-perf05</td>
</tr>
<tr>
<td>cvm-perf07</td>
<td>149</td>
<td>cvm-perf08</td>
<td>137</td>
<td>cvm-perf05</td>
</tr>
<tr>
<td>cvm-perf08</td>
<td>145</td>
<td>cvm-perf08</td>
<td>149</td>
<td>cvm-perf08</td>
</tr>
</tbody>
</table>
Main task - Results IV

![Graph: simultaneous installation - physical machines](image-url)
After the simultaneous installation, we did also verify the functionality by running some jobs - everything worked as expected.

I was asked to also test more simultaneous publications than is the current number of receiver processes (4 publications to 1 worker) - slightly higher times, finished well.

During the tests we didn’t see any crash of CVMFS, not even in the new RM and GW functionality.
All the results together with the Ansible playbook can be found here.

If you have a few spare machines and some time, the tests should be easily reproducible.

Feel free to contact me at ts@stdin.cz
Side project
$ man cvmfs_server
$ man cvmfs_server <subcommand>
$ cvmfs_server help <subcommand>
$ cvmfs_server <subcommand> --help

$ cvmfs_server help check

cvmfs_server check - Checks if the repository is sane

Synopsis: cvmfs_server check [options] <fqrn>

Options:
  -c : disable data chunk existence check
  -i : check data integrity (may take some time)
  -r : repair reflog problems
  -s : path to nested catalog subtree to check
  -t : tag (check given tag instead of trunk)
The tool is written in **shell**, need to write **own solution**

Before each `cvmfs_server` subcommand are placed documentation variables

```bash
_CVMFS_DOC_CHECK_SHORT="Checks if the repository is sane"
_CVMFS_DOC_CHECK_SYNOPSIS="[options] <fqrn>"
_CVMFS_DOC_CHECK_OPTIONS="\nc:disable data chunk existence check
i:check data integrity (may take some time)
... "

cvmfs_server_check() { ... }
```
“It’s hardware that makes a machine fast. It’s software that makes a fast machine slow.”

–Craig Bruce