BlackSwan: The Case for a Practical Reproducibility Platform

RESCUE-HPC’18 (SC’18)

Ivo Jimenez and Carlos Maltzahn (UC Santa Cruz)
[ivo.jimenez|carlosm]@ucsc.edu
twitter: @ivotron
Problem of *Practical* Reproducibility in Computer, Computational and Data Science

Replicate:
- What compiler was used?
- Which compilation flags?
- How was subsystem X configured?

Reuse:
- What parameters can be modified?
- What if I use input dataset Y?
- And if I run on platform Z?
How is a Manuscript Prepared?
A Deadline-friendly Approach

Typical

DevOps

• Automation
  • Bash, Python, etc.

• Version-control
  • Git, Mercurial, etc.

• Portability
  • Containers, VMs, Spack, etc.

myscript.sh

$ bash myscript.sh
The Popper Experimentation Protocol

1. Pick one or more tools from the DevOps toolkit.
2. Write *portable* scripts for a pipeline.
3. Put all scripts in a version control repository.

BLIS vs. other BLAS implementations

BLIS is a portable software framework for instantiating high-performance BLAS-like dense linear algebra libraries, experiment corresponds to the one presented in the first BLIS paper. A subsequent report documents how to repeat this experiment. This pipeline corresponds to sections 2.1-2.3 of the replicability report and consists of three stages:

- **build-docker-image**: A Docker image prepares all the binaries for BLIS, OpenBLAS and Atlas precompiled.
- **run**: Executes the experiment that compares BLIS against other BLAS implementations, generating output to results/ folder.
- **analyze**: Runs the analysis of the output of the experiment, corresponding to figures 13-15 of the original paper. To visualize results locally, one can execute the following:

```
docker run -d 
  -v /path/to/your/experiment:/code/experiment 
  -v /path/to/your/JupyterNotebook:/JupyterNotebookApp.sock 
  -e "PATH=$PATH:/usr/local/bin" 
  --name=scruffy 
  $PYTHON
```

After the above executes, open Browser and point it to http://localhost:8888. To see an example of how the notebook looks click here.

---


Popper Compliant Pipelines

On every execution:
1. Checkout code and data dependencies.
2. Build, install and deploy system under test.
3. Allocate resources (statically or dynamically).
4. Expose pipeline parameters in a plain text file.
5. Capture environment info (hardware, software stacks).
6. Validate results.

Community Building, Outreach and Teaching

• $$$ for teaching Popper
• Organize one-day hands-on tutorials
• Attend hackathons.
• Engage with OpenScience community.
• Improved experimentation practices.
• Increased confidence on results.
• Efficient use of time, both at personal and collaboration levels.
• Steep learning curve; radical paradigm shift.

• Submission deadline pressure is too big.

• Lack of incentives for taking the leap; no group-wide adoption.
### DevOps Technology

<table>
<thead>
<tr>
<th>Collaborate</th>
<th>Build</th>
<th>Test</th>
<th>Deploy</th>
<th>Run</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Lifecycle Mgmt.,</td>
<td>SCM/VCS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JIRA, GitLab, Git</td>
<td>Github, Bitbucket, Gitbucket</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication &amp; ChatOps</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slack, HipChat, irc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JUnit, PyUnit, MTrig</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Knowledge Sharing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GitHub Pages, Confluence</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Build</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jenkins, Travis CI</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MySQL, MongoDB, H2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Community**

- [BlackSwan](#): Practical Reproducibility Platform
- [open source](#)
Pipeline as the Central Concept

Search:
spark

Results:
spark-crawl
spark-tensorflow
spark-tpch

Pipepline:

Stage: cluster-setup

Description: This pipeline benchmarks tensorflow embedded in Spark workers.

Runs on: CloudLab

Status: blackwan passing

Required variables:
- CLOUDLAB_USER_NAME
- CLOUDLAB_PASSWORD
- CLOUDLAB_CERT_PATH

Dependencies:
- Docker 17.03
- Python 2.7
Automated Compliance Validation

Self-verifiable Experimentation Pipeline (SEP) criteria:
1. Checkout code and data dependencies.
2. Build, install and deploy system under test.
3. Expose pipeline parameters in a plain text file.
4. Allocate resources (statically or dynamically).
5. Capture environment info (hardware, software stacks).
6. Validate results.

```yaml
pipeline-folder/parameters.yml
/software.yml
/infra-manifest.json
```

```
expect linear(num_nodes, throughput)
when code not net_saturated
expect throughput data (raw_bw * 0.9)
expect MAPE <= 20
```

stdout:
```
[true] system scales linearly
[false] MAPE not within expected CI
```
# Validation Dashboard

<table>
<thead>
<tr>
<th>RUN</th>
<th>MESSAGE</th>
<th>DURATION</th>
<th>COMPLETED</th>
</tr>
</thead>
<tbody>
<tr>
<td>232</td>
<td>Synchronised bandwidth-monitored framework</td>
<td>3h 21s</td>
<td></td>
</tr>
<tr>
<td>231</td>
<td>Rete monitorata incrementale</td>
<td>2h 5m 58s</td>
<td>22 secs ago</td>
</tr>
<tr>
<td>230</td>
<td>Elus accusamus qui omnis.</td>
<td>29 sec</td>
<td>1 mins ago</td>
</tr>
<tr>
<td>229</td>
<td>Phased zereadministratim middleware</td>
<td>1h 6m 22 sec</td>
<td>18 mins ago</td>
</tr>
<tr>
<td>228</td>
<td>Innovative 24/7 encryption. Fundamental dynamic matrix. Moratoria...</td>
<td>22 min 46 sec</td>
<td>36 mins ago</td>
</tr>
<tr>
<td>227</td>
<td>Public-key logistical function</td>
<td>20 min 59 sec</td>
<td>43 mins ago</td>
</tr>
<tr>
<td>226</td>
<td>Asperiores omnis voluptatem autem voluptatem.</td>
<td>1h 0m 51s</td>
<td>1 hour ago</td>
</tr>
<tr>
<td>225</td>
<td>Focused national analyzer</td>
<td>7m 24s</td>
<td>3 hours ago</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>3m 21s</td>
<td>16 hours ago</td>
</tr>
<tr>
<td>3</td>
<td>Exclusive national success. Up-sized high-level matrix. Definizione...</td>
<td>37s</td>
<td>22 hours ago</td>
</tr>
<tr>
<td>10</td>
<td>Reduced high-level application</td>
<td>5m 27s</td>
<td>1 day ago</td>
</tr>
<tr>
<td>1</td>
<td>Sfida switchabile massimizzata</td>
<td>1m 46s</td>
<td>6 days ago</td>
</tr>
<tr>
<td>4</td>
<td>Future-proofed clear-thinking groupware</td>
<td>13m 48s</td>
<td>1 week ago</td>
</tr>
</tbody>
</table>
Differential Analysis

Semantic (environment) `diff`:

- Hardware.
- Software dependencies.
- Parameters.
- Input datasets.
- etc..
Black Swan – OSS Community-driven Platform

• Success depends on having multiple domain-specific communities involved in the creation, maintenance and curation of pipelines.

• Killer feature: UX – Make it *ridiculously easy* to implement and re-run an experimentation pipeline.
Use Cases

• Research curation.
  • Institutional libraries (universities and government).
  • Synergies with existing efforts (UC3, DOE code, etc.).

• Technology transfer in R&D scenarios.
  • University / Industrial / Government labs.
  • DOE labs scenario.

• Reproducibility Evaluation for Conferences and Journals.
  • SEP as complement to AD/AE.
  • Paper reproduced by SC’s Student Cluster Competition.
Thrilled to announce that our "Black Swan" @cross_ucsc incubator proposal has been accepted!! This means that, after I graduate, I get to keep working on making @getpopper better, and collaborating with more communities to bring DevOps practices to #opencience #opendata #oss

- Incubation of BlackSwan OSS project.
  - Starting Spring 2019.
- Looking for collaborators!
  - Beta-test community aspects of the platform.
Thanks!