Integrity Checking in Pegasus

Karan Vahi, Mats Rynge
Scientific Workflow Integrity with Pegasus
NSF CICI Awards 1642070, 1642053, and 1642090

GOALS

Provide additional assurances that a scientific workflow is not accidentally or maliciously tampered with during its execution.

Allow for detection of modification to its data or executables at later dates to facilitate reproducibility.

Integrate cryptographic support for data integrity into the Pegasus Workflow Management System.

PIs: Von Welch, Ilya Baldin, Ewa Deelman, Steve Myers
Team: Omkar Bhide, Rafael Ferrieira da Silva, Randy Heiland, Anirban Mandal, Rajiv Mayani, Mats Rynge, Karan Vahi
Challenges to Scientific Data Integrity

Modern IT systems are not perfect - errors creep in. At modern “Big Data” sizes we are starting to see checksums breaking down.

Plus there is the threat of intentional changes: malicious attackers, insider threats, etc.
Motivation: CERN Study of Disk Errors

Examined Disk, Memory, RAID 5 errors.

“The error rates are at the 10^-7 level, but with complicated patterns.” E.g. 80% of disk errors were 64k regions of corruption.

Explored many fixes and their often significant performance trade-offs.

Motivation: Network Corruption

Network router software inadvertently corrupts TCP data and checksum!

XSEDE and Internet2 example from 2013.

Second similar case in 2017 example with FreeSurfer/Fsurf project.

https://www.xsede.org/news/-/news/item/6390
Motivation: Software failure

Bug in StashCache data transfer software would occasionally cause silent failure (failed but returned zero).

Internal to the workflow this was detected when input to a stage of the workflow was detected as corrupted and retry invoked. (60k retries and an extra 2 years of cpu hours!)

However, failures in the final staging out of data were not detected because there was no workflow next stage to catch the errors.

The workflow management system, believing workflow was complete, cleaned up, so final data incomplete and all intermediary data lost. Ten CPU*years of computing came to naught.
Enter application-level checksums

Application-level checksums address these and other issues (e.g. malicious changes).

In use by many data transfer applications: scp, Globus/GridFTP, some parts of HTCondor, etc.

To include all aspects of the application workflow, requires either manual application by a researcher or integration into the application(s).
Automatic Integrity Checking - Goals

• Capture data corruption in a workflow by performing integrity checks on data

• Come up with a way to query, record and enforce checksums for different types of files
  • Raw input files – input files fetch from input data server
  • Intermediate files – files created by jobs in the workflow
  • Output files – final output files a user is actually interested in, and transferred to output site

• Modify Pegasus to perform integrity checksums at appropriate places in the workflow.

• Provide users a dial on scope of integrity checking
Data Staging Configurations

Condor I/O (HTCondor pools, OSG, ...)
- Worker nodes do not share a file system
- Data is pulled from / pushed to the submit host via HTCondor file transfers
- Staging site is the submit host

Non-shared File System (clouds, OSG, ...)
- Worker nodes do not share a file system
- Data is pulled / pushed from a staging site, possibly not co-located with the computation

Shared File System (HPC sites, XSEDE, Campus clusters, ...)
- I/O is directly against the shared file system

Pegasus Guarantee - Wherever and whenever a job runs it’s inputs will be in the directory where it is launched.
Automatic Integrity Checking

Pegasus will perform integrity checksums on input files before a job starts on the remote node.

- For raw inputs, checksums specified in the input replica catalog along with file locations. Can compute checksums while transferring if not specified.

- All intermediate and output files checksums are generated and tracked within the system.

- Support for sha256 checksums

Failure is triggered if checksums fail
Recent Developments

• **pegasus-transfer** can now checksum files during a file transfer.
  - The Planner decides what files should be check summed and indicates via a flag “`generate_checksum`”.
  - Can **involve an extra transfer** to where the stage-in job runs if third party transfer.
  - The checksum information is **populated in the kickstart record and populated in the monitoring database**, and for generation of **meta files to be used for the compute jobs**

• **pegasus-transfer** can now verify remote checksum of files after completing a file transfer
  - The Planner decides what files should be verified and indicates via a flag “`verify_remote_checksum`”.
  - Most of the times involves an extra transfer to as the file has to pulled down from the remote destination site, unless destination was a file URL
Cases Addressed

- Avoid triggering integrity checks for raw inputs if checksum not available in Replica Catalog.
- Allows us to compute checksums of input files for which the user did not provide checksums.
- Allows us to pull down output data after stage-out to output site is done, and verify the checksum.
  - Checksum information is retrieved from the meta files of the compute jobs.
- Gives a complete end to end solution for non shared fs deployments.
  - We are checking integrity of files at each step.
Initial Results with Integrity Checking on

• OSG-KINC workflow (50606 jobs) encountered **60 integrity errors** in the wild (production OSG). The problematic jobs were automatically retried and the workflow finished successfully.

• The 60 errors took place on 3 different hosts. The first one at UColorado, and group 2 and 3 at UNL hosts.

  **Error Analysis**

  • Host 2 had 3 errors, all the **same bad checksum** for the "kinc" executable with only a few seconds in between the jobs.

  • Host 3 had 56 errors, all the same bad checksum for the same data file, and over the timespan of 64 minutes. The site level cache still had a copy of this file and it was the correct file. Thus we suspect that the node level cache got corrupted.
Automatic Integrity Checking – Dials under consideration

- Allow a user to specify what files need to be checked
  1. No checking
  2. Raw inputs if checksum specified and all intermediate files and all intermediate files on the compute site
  3. All input files (compute for raw inputs if checksum not available) and all intermediate files on the compute site. No verification of staged outputs on output site
  4. All files included the staged final outputs to output site.

<table>
<thead>
<tr>
<th>Dial</th>
<th>Inputs</th>
<th>Intermediate Files</th>
<th>Final Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>2</td>
<td>Y*</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>3</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>4</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

* Full Integrity Checking will be turned ON by default for nonsharedfs and condorio deployments
Pegasus est. 2001
Automate, recover, and debug scientific computations.

Get Started

Pegasus Website
http://pegasus.isi.edu

Users Mailing List
pegasus-users@isi.edu

Support
pegasus-support@isi.edu

HipChat
Pegasus est. 2001
Automate, recover, and debug scientific computations.

Thank You

Questions?

Karan Vahi
vahi@isi.edu

Meet our team

Ewa Deelman
Karan Vahi
Mats Rynge
Rajiv Mayani
Rafael Ferreira da Silva