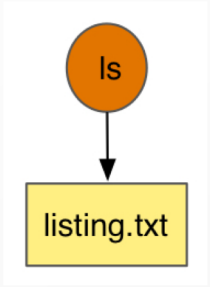
Pegasus : Introducing Integrity to Scientific Workflows

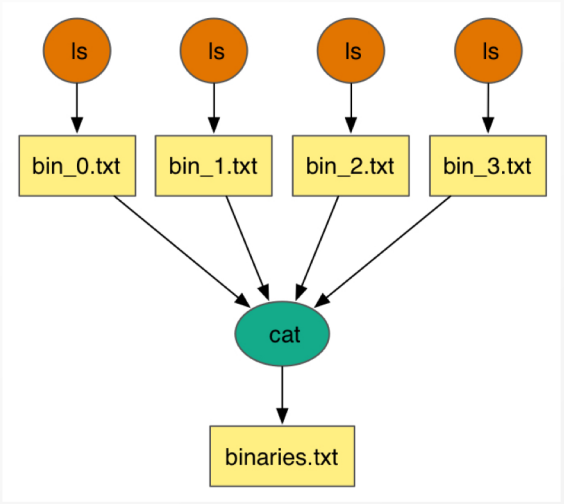
Karan Vahi
vahi@isi.edu

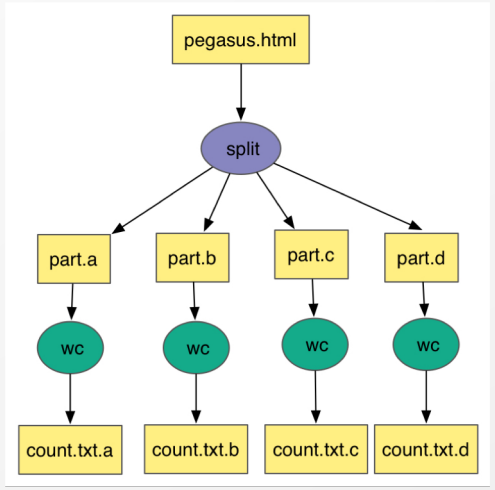
USC Viterbi
School of Engineering
Information Sciences Institute

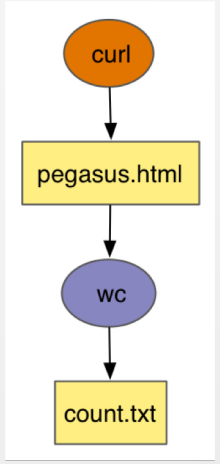
https://pegasus.isi.edu

Compute Pipelines – Building Blocks









HTCondor DAGMan

- DAGMan is a reliable and a scalable workflow executor
  - Sits on top of HTCondor Schedd
  - Can handle very large workflows
- Has useful reliability features in-built
  - Automatic job retries and rescue DAG's (recover from where you left off in case of failures)
- Throttling for jobs in a workflow

However, it is still up-to user to figure out

- Data Management
  - How do you ship in the small/large amounts data required by your pipeline and protocols to use?
- How best to leverage different infrastructure setups
  - OSG has no shared filesystem while XSEDE and your local campus cluster has one!
- Debug and Monitor Computations.
  - Correlate data across lots of log files.
  - Need to know what host a job ran on and how it was invoked
- Restructure Workflows for Improved Performance
  - Short running tasks?
  - Data placement

Why Pegasus?

Automates complex, multi-stage processing pipelines

Enables parallel, distributed computations

Portable: Describe once; execute multiple times

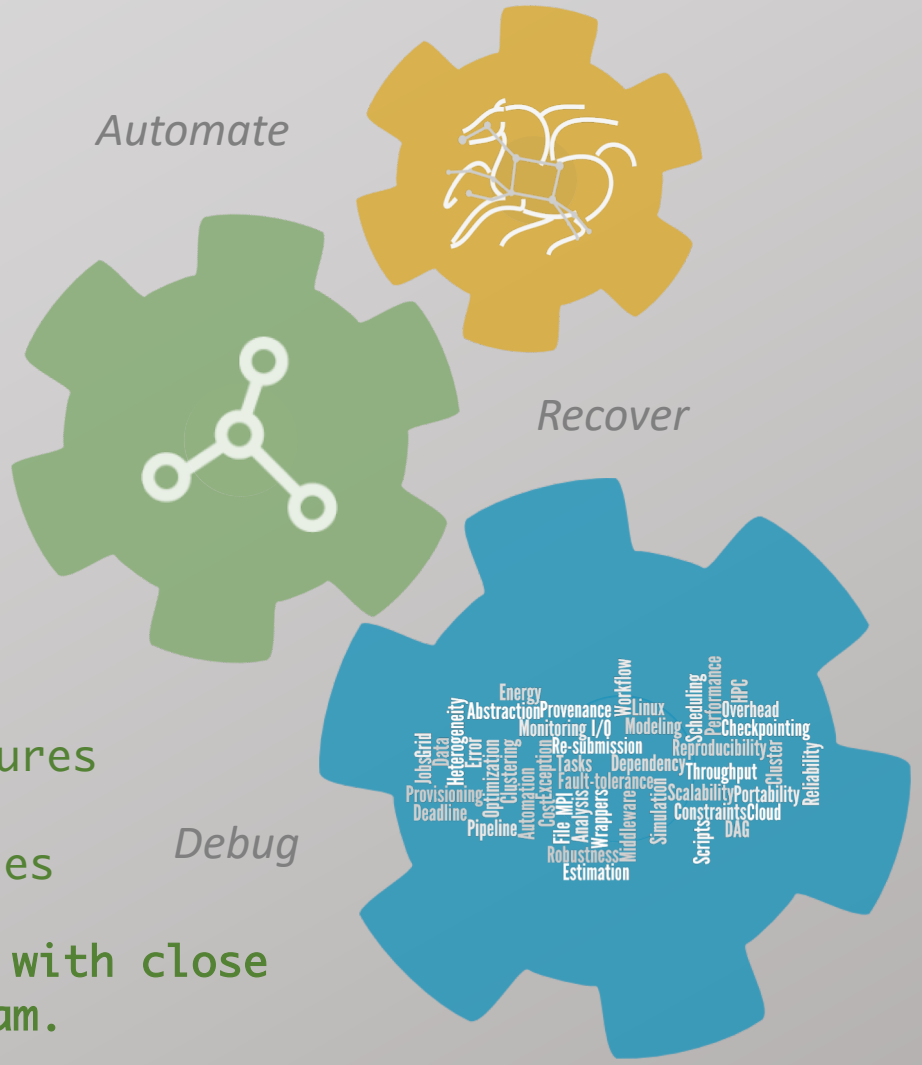
Automatically executes data transfers

Reusable, aids reproducibility

Records how data was produced (provenance)

Provides to tools to handle and debug failures

Keeps track of data and files



HTCondor
High Throughput Computing

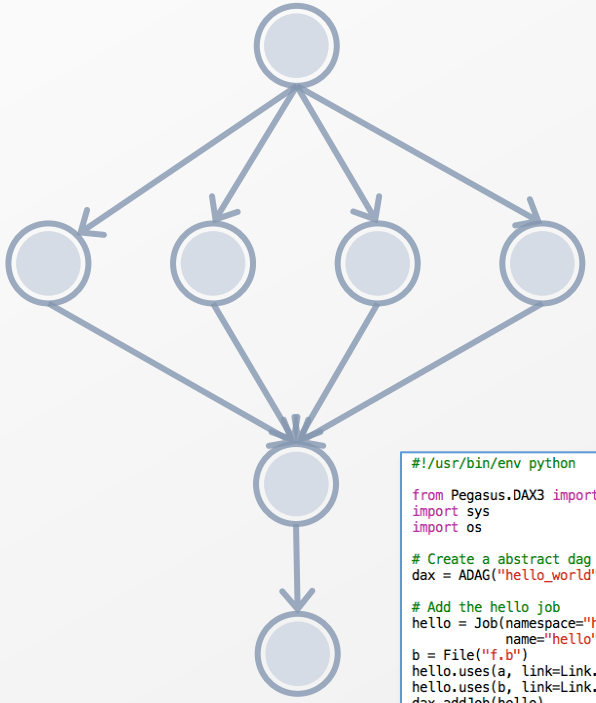
NSF funded project since 2001, with close Collaboration with HTCondor team.

DAX

DAG in XML

Portable Description

Users don't worry about low level execution details



```
#!/usr/bin/env python

from Pegasus.DAX3 import *
import sys
import os

# Create a abstract dag
dax = ADAG("hello_world")

# Add the hello job
hello = Job(namespace="hello_world",
            name="hello", version="1.0")
b = File("f.b")
hello.uses(a, link=Link.INPUT)
hello.uses(b, link=Link.OUTPUT)
dax.addJob(hello)

# Add the world job (depends on the hello job)
world = Job(namespace="hello_world",
            name="world", version="1.0")
c = File("f.c")
world.uses(b, link=Link.INPUT)
world.uses(c, link=Link.OUTPUT)
dax.addJob(world)

# Add control-flow dependencies
dax.addDependency(Dependency(parent=hello,
                             child=world))

# Write the DAX to stdout
dax.writeXML(sys.stdout)
```

DAG

directed-acyclic graphs

stage-in job

Transfers the workflow input data

clustered job

Groups small jobs together to improve performance

cleanup job

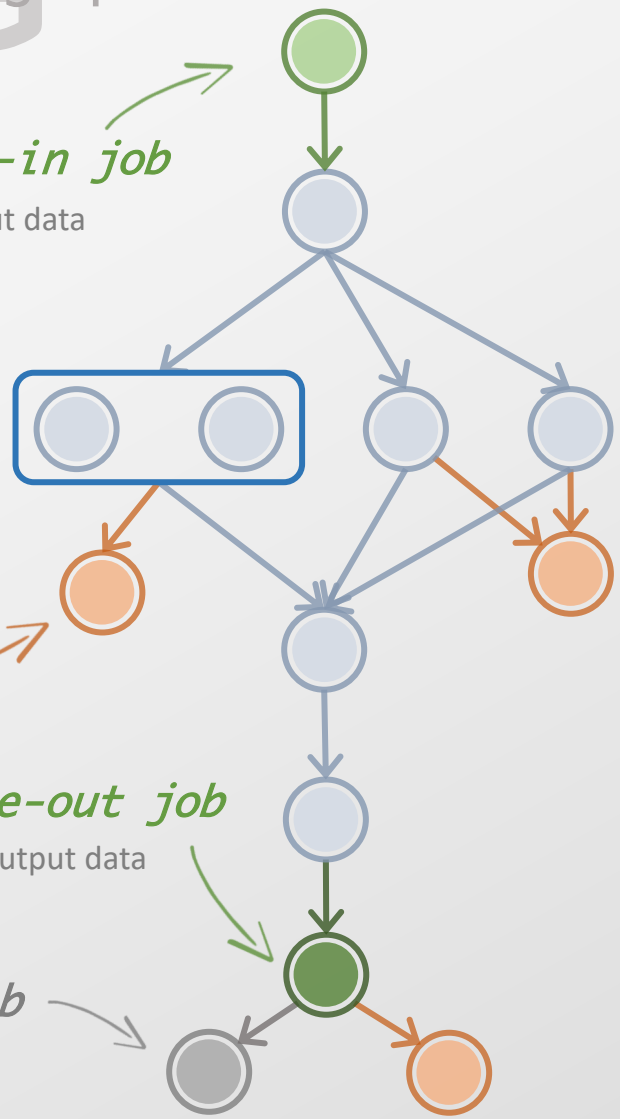
Removes unused data

stage-out job

Transfers the workflow output data

registration job

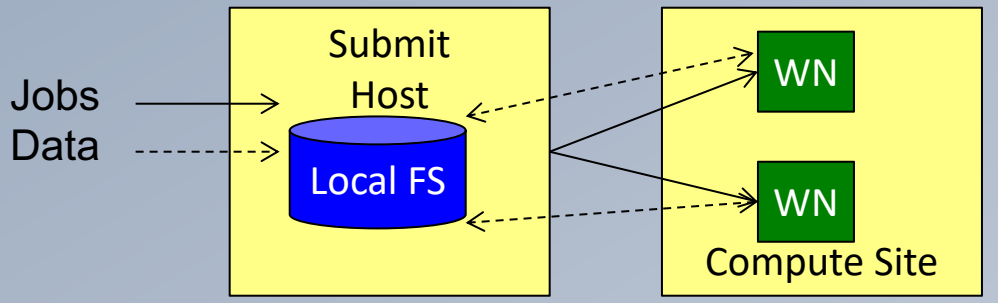
Registers the workflow output data



Data Staging Configurations

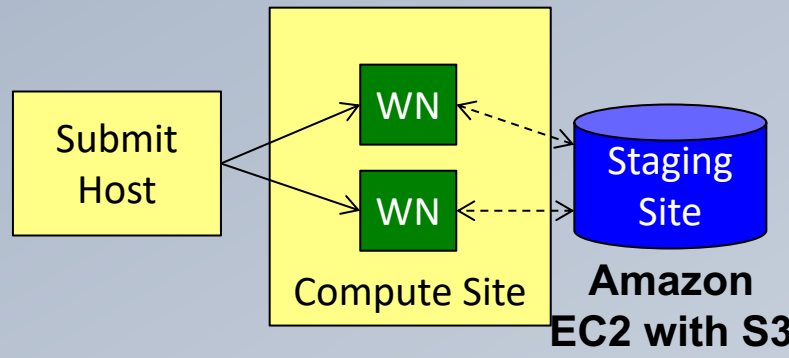
Condor I/O (HTCondor pools, OSG, …)

- 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, …)

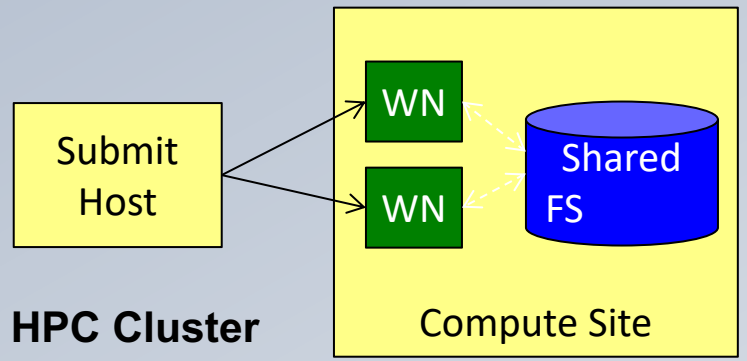
- Worker nodes do not share a file system
- Data is pulled / pushed from a staging site, possibly not co-located with the computation



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.

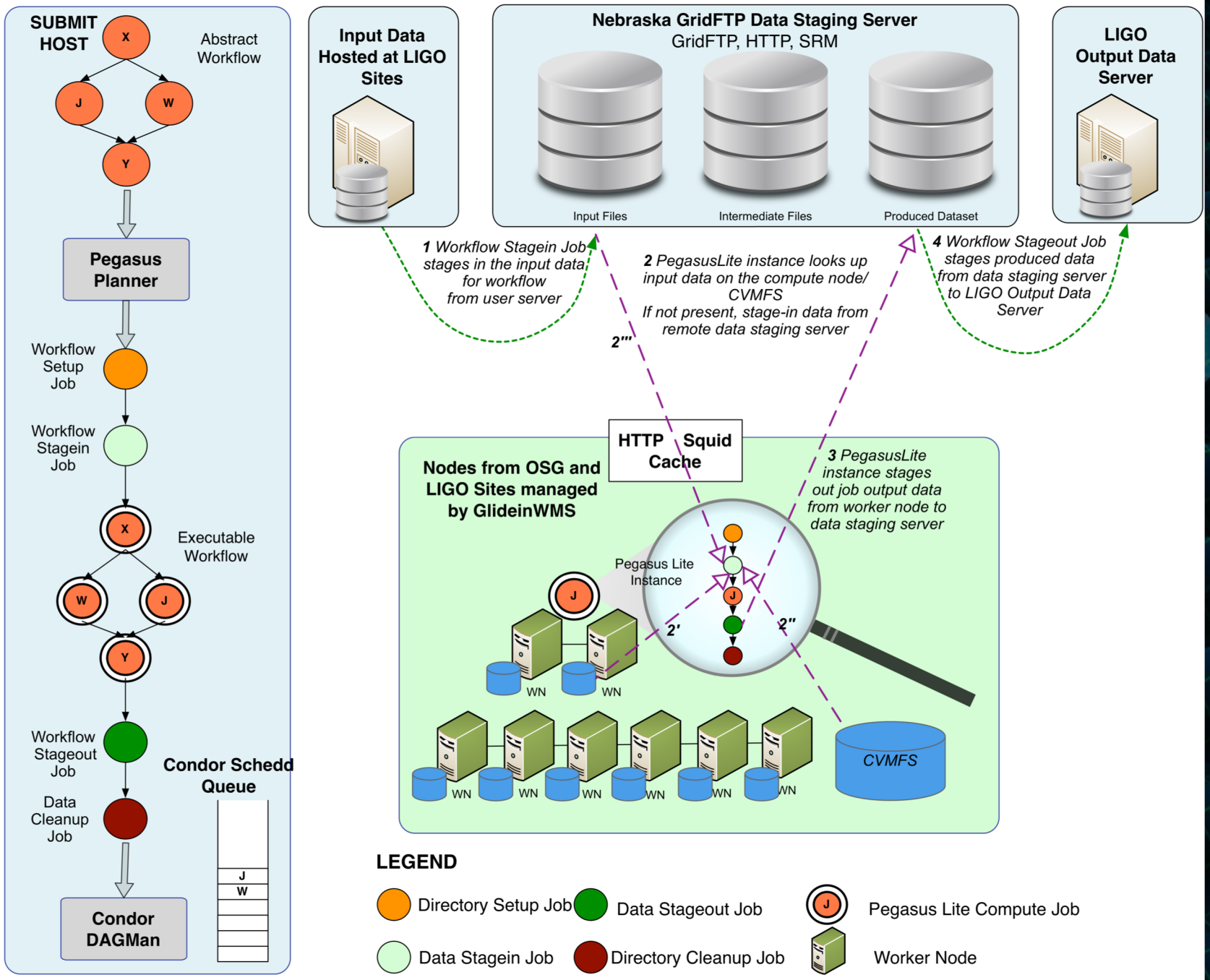


pegasus-transfer

- Pegasus' internal data transfer tool with support for a number of different protocols
- Directory creation, file removal
  - If protocol supports, used for cleanup
- Two stage transfers
  - e.g. GridFTP to S3 = GridFTP to local file, local file to S3
- Parallel transfers
- Automatic retries
- Credential management
  - Uses the appropriate credential for each site and each protocol (even 3rd party transfers)

HTTP
SCP
GridFTP
Globus Online
iRods
Amazon S3
Google Storage
SRM
FDT
stashcp
cp
ln -s

Data Flow for LIGO Pegasus Workflows in OSG



Advanced LIGO – Laser Interferometer Gravitational Wave Observatory

Scientific Workflow Integrity with Pegasus

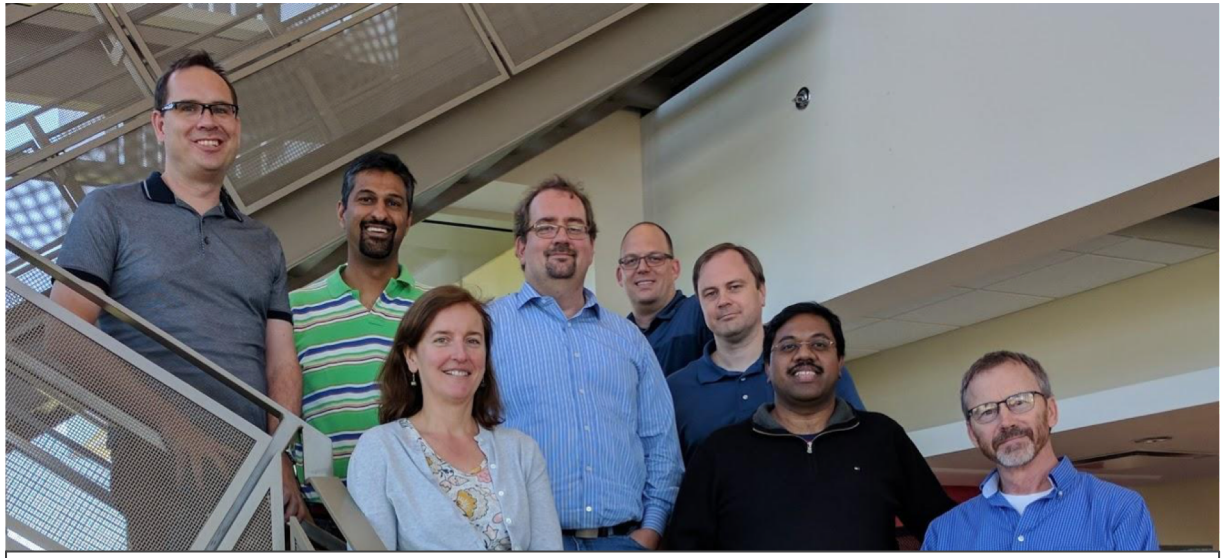
NSF CICI Awards 1642070, 1642053, and 1642090

GOALS

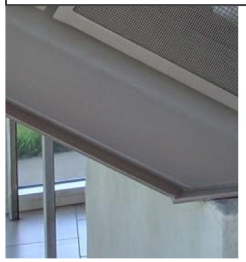
Provide additional assurances that a scientific workflow is not accidentally or maliciously tampered with during its execution

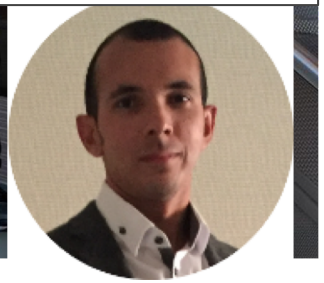
Allow for detection of modification to its data or executables at later dates to facilitate reproducibility.

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



SWIP Scientific Workflow Integrity with Pegasus | cacr.iu.edu/projects/swip/

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.

Data integrity

Bernd Panzer-Steindel, CERN/IT
Draft 1.3 8. April 2007

Executive Summary

We have established that low level data corruptions exist and that they have several origins. The error rates are at the 10^{-7} level, but with complicated patterns. To cope with the problem one has to implement a variety of measures on the IT part and also on the experiment side. Checksum mechanisms have to implemented and deployed everywhere. This will lead to additional operational work and the need for more hardware.

Introduction

During January and February 2007 we have done a systematic analysis of data corruption cases in the CERN computer center. The major work in the implementation of probes and automatic running schemes were done by Tim Bell, Olof barring and Peter Kelemen from the IT/FIO group. There have been similar problems reported in Fermilab and Desy and information exchange with them was done.
The following paper will provide results from this analysis, a judgment of the situation and a catalogue of measures needed to get the problem under control.
It is also to be seen as a starting point for further discussions with IT, the experiments and the T1 sites.

https://indico.cern.ch/event/13797/contributions/1362288/attachments/115080/163419/Data_integrity_v3.pdf

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.

BROCADE

TECHNICAL SUPPORT BULLETIN

June 28, 2013

TSB 2013-162-A SEVERITY: Critical- Service Impact

PRODUCTS AFFECTED:
Brocade NetIron XMR/MLX 100G module (BR-MLX-100Gx2-X and BR-MLX-100Gx1-X).

CORRECTED IN RELEASE:
The fix will be in patch releases of NI 5.3.00eb, 5.4.00d and 5.5.00c and later releases.
This issue is not applicable to software release NI 5.2.00 and previous releases.

BULLETIN OVERVIEW

When transferring data through 100G modules, a portion of the packet may get corrupted. Corruption is typically seen when transferring jumbo frames.

Brocade TSB 2013-162-A

XSEDE
Extreme Science and Engineering Discovery Environment

HOME ABOUT USER SERVICES EDUCATION & OUTREACH RESOURCES GATEWAY

News

XSEDE Network Status

Posted by Bob Garza on 07/25/2013 18:27 UTC

On March 1, 2013 XSEDENet, the network between XSEDE Service Providers, moved to Internet2's Advanced Layer 2 Service (AL2S) national network to take advantage of new features and performance capabilities.

XSEDE was notified recently by Internet2 that an error was discovered on the devices that Internet2 uses on its AL2S network that could possibly lead to data corruption. This error could have affected approximately 0.001% of the data that traversed **each** AL2S device and was undetectable by the standard TCP packet checksum. These errors would have primarily affected data transfers using protocols that did not employ data integrity capabilities (application compression, encryption or checksums). XSEDE users who used secure copy (scp) to transfer files were not affected due to its application layer checksums. Data transfers initiated with the Globus Online web interface also were not affected as Globus Online implemented default checksums in December 2012. Other data transfers including manual gridftp or other protocols without data integrity checking could have been affected by this error.

By July 17, 2013 Internet2, in cooperation with the device vendor, upgraded all the affected devices with a new version of software that corrected the error. XSEDE recommends that users who transferred files using data transfer protocols that do not incorporate data integrity capabilities check the integrity of their file transfers that occurred between March 1, 2013 and July 17, 2013. Please refer to the XSEDE documentation on data integrity and validation of data transfers for details about data integrity checks.

Please submit any questions you may have by sending email to help@xsede.org or by submitting your questions through the XSEDE User Portal @ https://portal.xsede.org/help-desk.

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 their 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

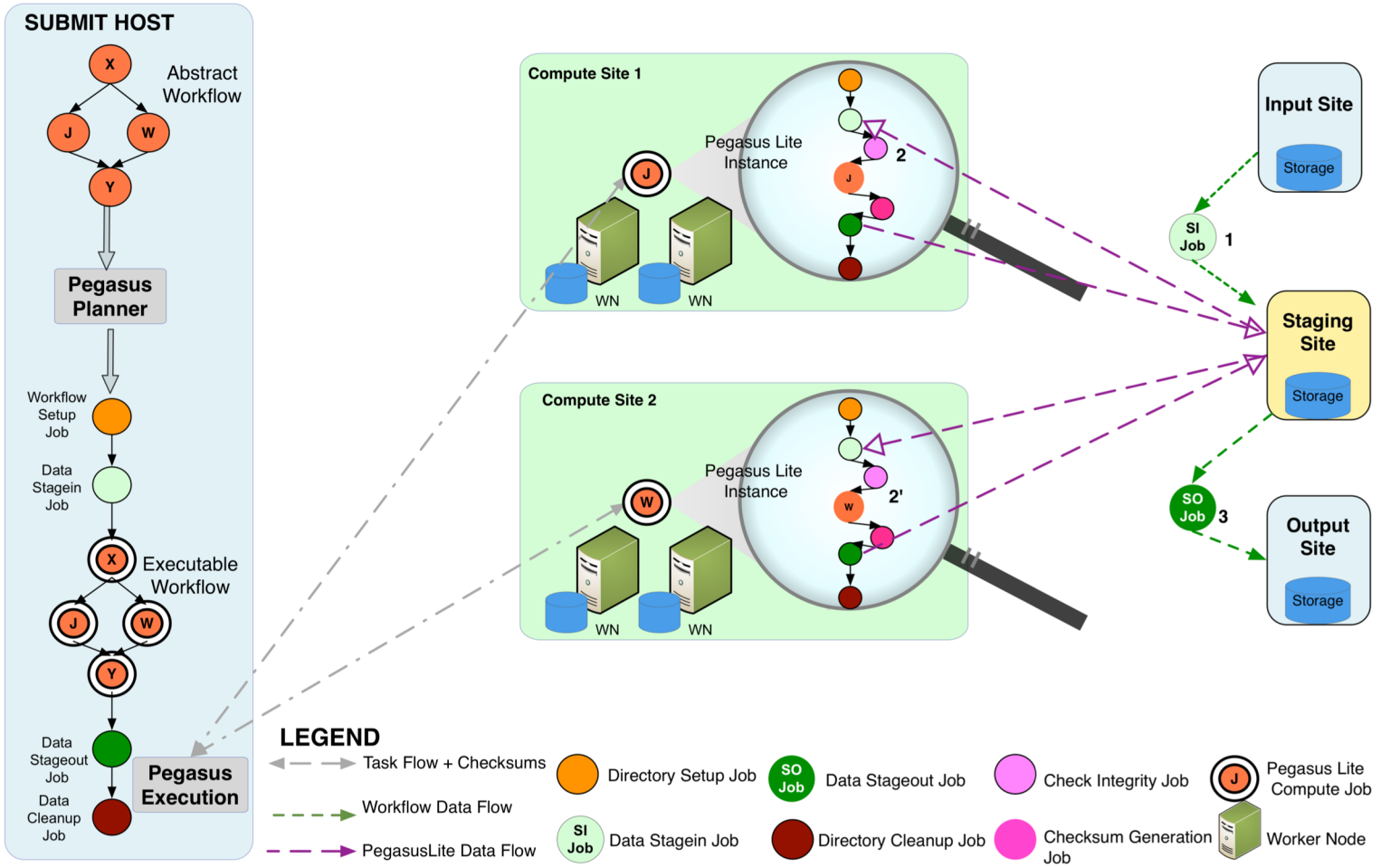
Automatic Integrity Checking

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

Introduced in Pegasus 4.9



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.

Checksum Overheads

- We have instrumented overheads and are available to end users via *pegasus-statistics.*

```
Type           Succeeded  Failed   Incomplete   Total      Retries    Total+Retries
Jobs           1606       0        0            1606       31         1637

Workflow wall time                                  : 7 hrs, 59 mins
Cumulative job wall time                            : 17 days, 23 hrs
# Integrity Metrics
3944 files checksums compared with total duration of 9 mins, 18 secs
1947 files checksums generated with total duration of 4 mins, 37 secs
# Integrity Errors
Failures: 0 jobs encountered integrity errors
```

1000 Node OSG Kinc Workflow

Overhead of 0.054 % incurred

- Other sample overheads on real world workflows
  - Ariella Gladstein's population modeling workflow
    - A 5,000 job workflow used up 167 days and 16 hours of core hours, while spending 2 hours and 42 minutes doing checksum verification, with an overhead of 0.068%.
  - A smaller example is the Dark Energy Survey Weak Lensing Pipeline with 131 jobs.
    - It used up 2 hours and 19 minutes of cumulative core hours, and 8 minutes and 43 seconds of checksum verification. The overhead was 0.062%.

Pegasus est. 2001

Automate, recover, and debug scientific computations.

Get Started

Pegasus Website
https://pegasus.isi.edu

Users Mailing List
pegasus-users@isi.edu

Support
pegasus-support@isi.edu

Pegasus Online Office Hours
https://pegasus.isi.edu/blog/online-pegasus-office-hours/

Bi-monthly basis on second Friday of the month, where we address user questions and also apprise the community of new developments

Pegasus est. 2001

Automate, recover, and debug scientific computations.

Thank You

Questions?

Karan Vahi
vahi@isi.edu

USC Viterbi
School of Engineering
Information Sciences Institute

Meet our team


Ewa Deelman


Karan Vahi


Mats Rynge


Rajiv Mayani


Rafael Ferreira da Silva

NSF U.S. DEPARTMENT OF ENERGY NIH