USC Viterbi
School of Engineering
Information Sciences Institute

Pegasus 5.0 Preview

Online Office Hours: *August 7th, 2020*

Karan Vahi , Ryan Tanaka

USC Information Sciences Institute

- Reworked Python API to compose, submit and monitor workflows and configure catalogs
  - Developed brand new from grounds up.
  - New ***yaml*** based format to describe workflows
  - Allows for creation of each of the catalogs (site, transformation, replica and properties)
  - Allows you to plan/submit/monitor/analyze/statistics of your workflow
- Python 3 support
  - All Pegasus tools are Python 3 compliant.
  - 5.0 release will require Python 3 on workflow submit node
  - Python PIP packages for workflow composition and monitoring

Tip: *JAVA and R API's now emit workflow descriptions in YAML like the new python API.*

Pegasus 5.0

Automate, recover, and debug scientific computations

- Formats
  - Adoption of **YAML** for all file based catalogs.

- Following are now represented in YAML
  - Abstract Workflow
  - Replica Catalog
  - Transformation Catalog
  - Site Catalog
  - Kickstart Provenance Records

- Replica Catalog
  - Pegasus Python API allows for easy creation as a separate file or embedded in the workflow description itself
  - By default, Pegasus will look for a file named **"replicas.yml"** in the current working directory

```
from Pegasus.api import *

infile = File('input.txt')
rc = ReplicaCatalog()\
      .add_replica('local', infile, "http://example.com/pegasus/input/" + infile.lfn,\
                   checksum = {'sha256':'66a42b4be204c824a7533d2c677ff7cc5c44526300ecd6b450
       .write()

# the Replica Catalog will be written to the default path "./replicas.yml"
```

- Support for Regular Expressions
  - Format supports regular expressions as long as it is written out as a separate file

```
pegasus: '5.0'
replicas:
  - lfn: input.txt
    pfns:
      - {site: local, pfn: 'http://example.com/pegasus/input/input.txt'}
    checksum: {sha256: 66a42b4be204c824a7533d2c677ff7cc5c44526300ecd6b450602e06128063f9}
```

Tip: *use pegasus-rc-converter to convert your existing replica catalog.*

- Transformation Catalog
  - Pegasus Python API allows for easy creation as a separate file or embedded in the workflow description itself
  - By default, Pegasus will look for a file named ***"transformations.yml"*** in the current working directory

```python
from Pegasus.api import *

# create the TransformationCatalog object
tc = TransformationCatalog()

# create and add the transformation
keg = Transformation(
        "keg",
        namespace="example",
        version="1.0",
        site="isi",
        pfn="/path/to/keg",
        is_stageable=False,

    ).add_profiles(Namespace.ENV, APP_HOME="/tmp/myscratch", JAVA_HOME="/opt/java/1.6")

tc.add_transformations(keg)

# write the transformation catalog to the default file path "./transformations.yml"
tc.write()
```

```yaml
x-pegasus: {apiLang: python, createdBy: vahi, createdOn: '07-23-20T16:43:51Z'}
pegasus: '5.0'
transformations:
- namespace: example
  name: keg
  version: '1.0'
  sites:
  - {name: isi, pfn: /path/to/keg, type: installed}
  profiles:
    env: {APP_HOME: /tmp/myscratch, JAVA_HOME: /opt/java/1.6}
```

Tip: *use pegasus-tc-converter to convert your existing transformation catalog.*


```python
from Pegasus.api import *

# create a SiteCatalog object
sc = SiteCatalog()

# create a "local" site
local = Site("local", arch=Arch.X86_64, os_type=OS.LINUX)

# create and add a shared scratch and local storage directories to
    the site "local"
local_shared_scratch_dir = Directory(Directory.SHARED_SCRATCH,
    path="/tmp/workflows/scratch")\

                            .add_file_servers(FileServer
                            ("file:///tmp/workflows/scratch",
                            Operation.ALL))

local_local_storage_dir = Directory(Directory.LOCAL_STORAGE,
    path="/tmp/workflows/outputs")\

                            .add_file_servers(FileServer
                            ("file:///tmp/workflows/outputs",
                            Operation.ALL))

local.add_directories(local_shared_scratch_dir,
    local_local_storage_dir)


# add all the sites to the site catalog object
sc.add_sites(
    local
)

# write the site catalog to the default path "./sites.yml"
sc.write()
```

- Site Catalog
- Pegasus Python API allows for easy creation as a separate file or embedded in the workflow description itself
- By default, Pegasus will look for a file named ***"sites.yml"*** in the current working directory

```yaml
x-pegasus: {apiLang: python, createdBy: vahi, createdOn: '07-23-20T14:05:48Z'}
pegasus: '5.0'
sites:
- name: local
  arch: x86_64
  os.type: linux
  directories:
  - type: sharedScratch
    path: /tmp/workflows/scratch
    fileServers:
    - {url: 'file:///tmp/workflows/scratch', operation: all}
  - type: localStorage
    path: /tmp/workflows/outputs
    fileServers:
    - {url: 'file:///tmp/workflows/outputs', operation: all}
```

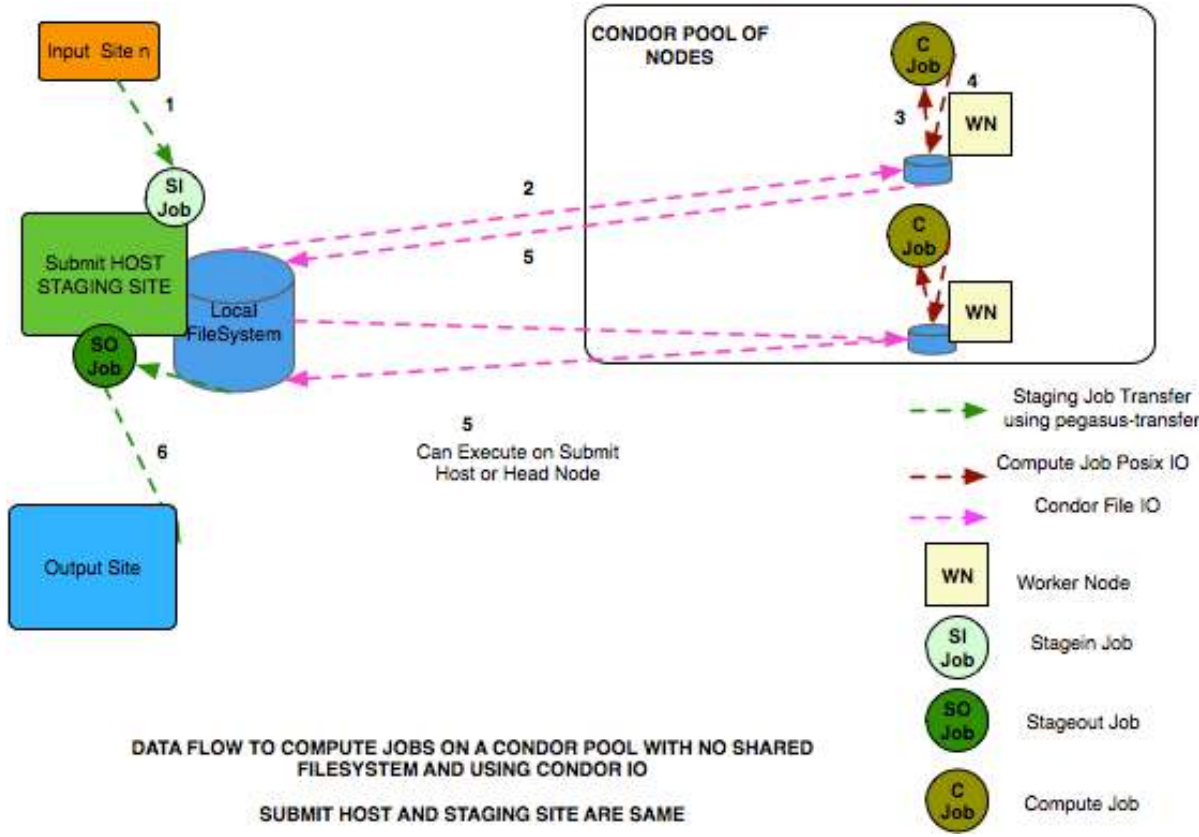

Tip: *use pegasus-sc-converter to convert your existing site catalog.*

- Zero configuration required to submit to local HTCondor pool.
  - The ***"hello world"*** example on the right will work out of the box
  - Pegasus will automatically create sensible defaults for sites
    - local
    - condorpool
  - By default, site ***"condorpool"*** is used as execution site.
  - Site ***"local"*** still designates the submit node, and is used to run Pegasus auxillary jobs.

```
#!/usr/bin/env python3
import logging
import sys

from Pegasus.api import *

# logs to be sent to stdout
logging.basicConfig(level=logging.DEBUG, stream=sys.stdout)

# --- Transformations ----------------------------------
echo = Transformation(
        "echo",
        pfn="/bin/echo",
        site="condorpool"
    )

tc = TransformationCatalog()\
        .add_transformations(echo)

# --- Workflow -----------------------------------------
Workflow("hello-world", infer_dependencies=True)\
    .add_jobs(
        Job(echo)
            .add_args("Hello World")
            .set_stdout("hello.out")
    ).add_transformation_catalog(tc)\
    .plan(submit=True)\
    .wait()
```

- Default Data Configuration
  - IS **“condorio”** from **“sharedfs”** earlier.
  - 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

- **Existing users if operating in sharedfs mode now need to set**
  - **pegasus.data.configuration = sharedfs** in their properties



Input Replica Catalog

Discover the location of input files or previously generated datasets to use for planning purposes.

- **Configuration:** use the properties prefix ***pegasus.catalog.replica***

Output Replica Catalog

- Registers outputs including file metadata such as size and checksums
- By default Pegasus will registers outputs to a JDBC based Replica Catalog (***workflow-name.replicas.db***) in the workflow submit directory.
- For hierarchical workflows only one output replica catalog db is generated in the root workflow submit directory.
- **Configuration:** use the properties prefix ***pegasus.catalog.replica.output***

Note: In 4.9.x and before, the input replica catalog was used for registration of outputs.

Hierarchical Workflow Improvements

Automatic handing of data dependencies between pegasusWorkflow (dax) jobs and compute jobs

```
x-pegasus: {apiLang: python, createdBy: bamboo, createdOn: '07-10-20 11:09:29'}
pegasus: '5.0'
name: local-hierarchy
jobs:
- type: pegasusWorkflow
  file: blackdiamond.yml
  id: ID0000001
  arguments: [--input-dir, input, --output-sites, local, -vvv, --force]
  uses:
  - {lfn: blackdiamond.yml, type: input}
  - {lfn: f.d, type: output, stageOut: true, registerReplica: true}

- type: job
  namespace: diamond
  version: '4.0'
  name: post-analyze
  id: ID0000002
  arguments: [-a, post-analyze, -T, '60', -i, f.d, -o, f.e]
  uses:
  - {lfn: f.d, type: input}
  - {lfn: f.e, type: output, stageOut: true, registerReplica: true}
jobDependencies:
- id: ID0000001
  children: [ID0000002]
```

pegasusWorkflow job (ID0000001) generates an output file “f.d”

Compute job (ID0000002) in the same workflow can use “f.d” as input

Data Management Improvements

- Ability to do bypass staging of files at a per file, executable and container level
  - Set the ***"bypass"*** flag for file/executable/container
  - Useful when you want to pull the container down only once from Docker|Singularity Hub, but do bypass for other input data
- Support for integrity checking of user executables and application containers in addition to data
- WebDAV support

New Common Credential File

- Manage simple credentials such as username / password / accesskey / secretkey / tokens in one place
- WebDAV, S3 - more to come
- ~/.pegasus/credentials.conf

Credentials Pre-flight Check

- Planner now does simple existence and permission checks for local credentials.

```
# For simple username/password protocols, such as WebDAV,
# just specify the hostname and credentials. In this
# example, the credentials would be used for URLs
# matching the section, such as
# webdav://data.cyverse.org/some/file.txt

[data.cyverse.org]

username = joe
password = secretsauce1

# For S3 access, you can create an entry for the cloud
# specific options, and then one or more user specific
# entries with a key @ matching the cloud one (for
# example, [amazon] and [joe@amazon] below)

[amazon]
endpoint = https://s3.amazonaws.com/

[joe@amazon]
access_key = 90c4143642cb097c88fe2ec66ce4ad4e
secret_key = abababababababababababababababab
```

Monitoring Improvements

- Unicode compatibility for databases. We also enforce a consistent UTF-8 environment
- We record “maxrss” the maximum physical memory used by a job during it’s execution
- average cpu utilization (utime+stime/duration)
  - Utime: CPU time spent in user code
  - Stime: CPU time spent in kernel code
  - Duration: total runtime of a job
- Pegasus-statistics will report these metrics
- **Useful for creating application profiles.**

CWL Support: pegasus-cwl-converter

- Convert a subset of CWL v1.1 to native Pegasus yaml format
- Automatic generation of Pegasus workflow, replica, and transformation catalogs into single file

COMMON WORKFLOW LANGUAGE

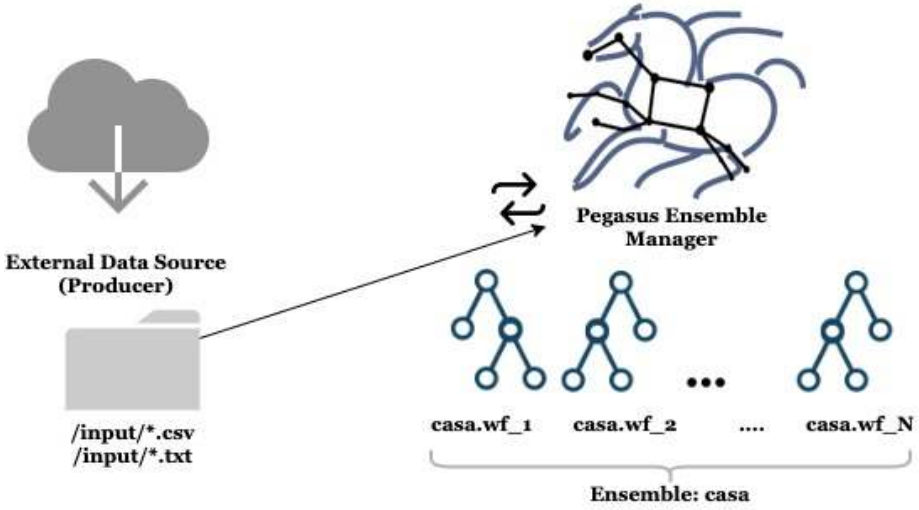
```
my_cwl_workflow
├── tar.cwl
├── compile_1.cwl
├── compile_2.cwl
├── get_file_sizes.cwl
├── get_file_sizes.sh
├── src_tarball
├── input.yml
└── workflow.cwl
```

```
apiLang: python
createdBy: ryantanaka
createdOn: 07-24-20T10:08:48Z
pegasus: "5.0"
name: compile
replicaCatalog:
  replicas: [...]
 transformationCatalog:
   transformations: [...]
jobs: [...]
jobDependencies: [...]
```

Manpage: *https://pegasus.isi.edu/docs/5.0.0dev/manpages/pegasus-cwl-converter.html*

Pegasus Ensemble Manager Improvements

- Python3 compliant
- Added ***file pattern-based triggering of workflows***
  - Multiple triggers can be started to dynamically submit workflows as new input files arrive
  - Each trigger can be given multiple file patterns to watch for
  - Each trigger operates on a given time interval
    - Files that match the given patterns, and that have been created during the current time interval are passed as inputs to a given workflow generation script



Pegasus 5.0

Automate, recover, and debug scientific computations

DEMO

https://github.com/pegasus-isi/pegasus-workflow-development-environment

Pegasus 5.0

Automate, recover, and debug scientific computations

Coming soon! Beta1 is out.

- 5.0beta1 is out
  - Grab it from https://download.pegasus.isi.edu/pegasus/5.0.0beta1/
  - Binary tarballs, RPM's and DEB packages available
- Existing Users
  - Carefully follow the migration guide
    - https://pegasus.isi.edu/docs/5.0.0dev/migration.html#migrating-from-pegasus-4-9-x-to-pegasus-5-0
- Major documentation improvements
  - Moved to restructured text format
  - https://pegasus.isi.edu/docs/5.0.0dev/index.html