|  |  |  |
| --- | --- | --- |
|  | Makeflow | Pegasus |
| Installation | Download a tar file, extract and ./configure to install make. Add to path.  $ cd $HOME  $ wget http://ccl.cse.nd.edu/software/files/cctools-5.1.1-source.tar.gz  $ tar xvzf cctools-5.1.1-source.tar.gz  $ cd cctools-5.1.1-source  $ ./configure  $ make  $ make install  $ cd $HOME  $ echo "PATH=/$PATH:\$HOME/cctools-5.1.1/bin" >> ~/.bashrc  $ source ~/.bashrc | $ wget <http://download.pegasus.isi.edu/pegasus/4.6.1/pegasus-binary-4.6.1-x86_64_rhel_7.tar.gz>  $ tar zxf pegasus-binary-4.6.1-x86\_64\_rhel\_7.tar.gz  $ echo "PATH=/$PATH:\$HOME/pegasus-4.6.1/bin" >> ~/.bashrc  $ source ~/.bashrc |
| Setup | * Makefile script defines all the rules. Rules consist of a target followed by a colon and any prerequisites, and commands on the indented next line. * output.txt: mysim.exe calib.data   ./mysim.exe -c calib.data -o output.txt | * Reads workflow descriptions from DAX files (Directed Acyclic Graph in XML). * First step in using Pegasus is to write a DAX generator. Can be done with Perl, Java, or Python. |
|  | Differences from Make:   * Cannot have rules that don’t actually create the output files. EG you cannot use a “clean” rule like would normally be used in Make. | DAX file:   * Initialize abstract ADAG object as dax. * Initialize files (input, output, anything in between) * Initialize each job as a Job object. Add arguments. Indicate which files the job uses for input and output. Add job to dax. Final job output is tagged with “transfer=true”. * Indicate dependencies (parent and child jobs) of dax. * Write dax as an XML to sys.stdout |
|  | To use SLURM:   * makeflow -T slurm -B "-p normal -t 1" example.makeflow * -B submits options to the batch system. SLURM requires –p to specify the queue to submit through, and –t time allocated for the job to run. More arguments can be added between the quotes. | Catalogs:   * Site catalog: describes the sites where the workflow jobs will be executed. Typically describes a remote cluster such as Slurm or HTCondor. Picked up automatically from an XML file named sites.xml in the cwd of the pegasus-plan. Must specify the allocation. * Transformation catalog: describes all of the executables (transformations) used by the workflow. Must include the site where they are located, architecture and os they are compiled for, and any other information need to transfer and run them. Automatically grabbed from a file named tc.txt. * Replica catalog: where to find each of the input files for the workflow. |
|  | Can also run through Work Queue:   * Start with makeflow -T wq -p 0 -N PROJECT-$USER --**work-queue-preferred-connection by\_hostname** example.makeflow * Next, start workers. Can be local: work\_queue\_worker -N PROJECT-$USER * Can be batch using slurm: slurm\_submit\_workers -N PROJECT-$USER -p "-t 1:00 -p normal" 2 | Configuration:   * File pegasus.properties contains configuration settings such as: where to find the catalogs, what to use to transfer data (condor etc) and the name of the app. |
|  | Other notes:   * Uses garbage collection to clean up intermediate files (not input or final output files) | pegasus-plan command plans the workflow. |
|  |  | Has a visual dashboard Pegasus-service showing current jobs |
|  |  |  |

Workflows in general:

* Workflows are a way to automate data flow to complete multi-step computational tasks with dependencies. For example: retrieve some data, reformat the data, and run an analysis.
* Dependencies are typically data file dependencies. Tasks can be anything from short tasks one after another, to large parallel tasks surrounded by a lot of small serial tasks for pre and post processing.
* The workflow keeps track of the data and executes tasks in the correct order.

Impressions:

* Makeflow: much easier to find resources on how to install
* Pegasus: better tutorials on writing scripts and getting started after installation
* Makeflow: only uses one script
* Atmosphere has some VMs set up for Pegasus

Look at tomorrow:

* Does Pegasus have a built-in Work Queue functionality?
* Pegasus more portable than Makeflow?
* Is Pegasus better enough to be worth the extra learning curve of writing the DAX and catalog files?

|  |  |  |
| --- | --- | --- |
|  | **Makeflow** | **Pegasus** |
| **Portability** | Does not reference any particular batch system or distributed computing technology, so existing workflows can be easily moved between different systems. | Neutral DAX scripts can be run on a single system or across a set of different resources. |
| **Performance** | Makeflow treats file dependencies very literally. This improves workflow performance by setting up the right environment, verifying correct execution, and managing resources. | The Pegasus mapper reorders, groups and prioritizes tasks to increase workflow performance. |
| **Scalability** | Makeflow can be run very easily on a single machine, or can use SLURM directly or with Work Queue to run on Stampede. | Pegasus can scale the size of the workflow and the resources the workflow is distributed over. For very large jobs, it allows “subworkflows” to allow compartmentalization of jobs. |
| **Reliability** | Can pass parameter –r when starting a job so Makeflow will automatically retry failed tasks. Outputs a logfile of debugging statements. | Jobs and data transfers are automatically retried when failed. Has a tool Pegasus-analyzer to debug non-recoverable failures. |
| **Error recovery** | Uses the Makeflow log file to recover after a failure and for use in logging and debugging. | Tries to recover by retrying tasks and the workflow; providing checkpoints; re-mapping the workflow; using alternative data sources. If everything still fails, provides a “rescue” workflow of only what still needs to be done. |
| **Provenance** | Does not support provenance queries. Produces a log file which records how and when every task is run. Used for logging and debugging. | Jobs are launched through a kickstart process that captures runtime provenance to help in debugging. Provenance data is stored in a database and can be summarized with Pegasus tools or through SQL queries. |
|  |  |  |
| **Installation** | Easy to find instructions. Simple installation of cctools followed by about a minute to compile and install Make. | Difficult to dig through instructions to find the correct installation process for Stampede. Simple to install once instructions are acquired.  Atmosphere has some images available with Pegasus already installed. |
| **Learning curve** | Similar to Make with some modifications. Makeflow tutorials are easy to follow but are not very detailed. The user manual mostly gives an overview of the differences between Makeflow and Make, rather than extensive instructions on actually using Makeflow. In order to learn Makeflow, the user would either have to be already familiar with Make or would have to learn Make first. | Nice clear tutorials are provided by Pegasus, available preloaded on a VirtualBox VM. Pegasus has a lot more components to learn, but the website has a very nice user guide with a table of contents and instructions on each part of the process. |
| **Ease of use** | *Makes workflow creation easier through abstraction by not having to specify low-level details.*  Easy to use for users who are familiar with Make and makefiles. Does not have as many convenient built-in tools as Pegasus. | *Makes workflow creation easier through abstraction by not having to specify low-level details.*  Pegasus requires several components for the script – DAX generator and catalogs |