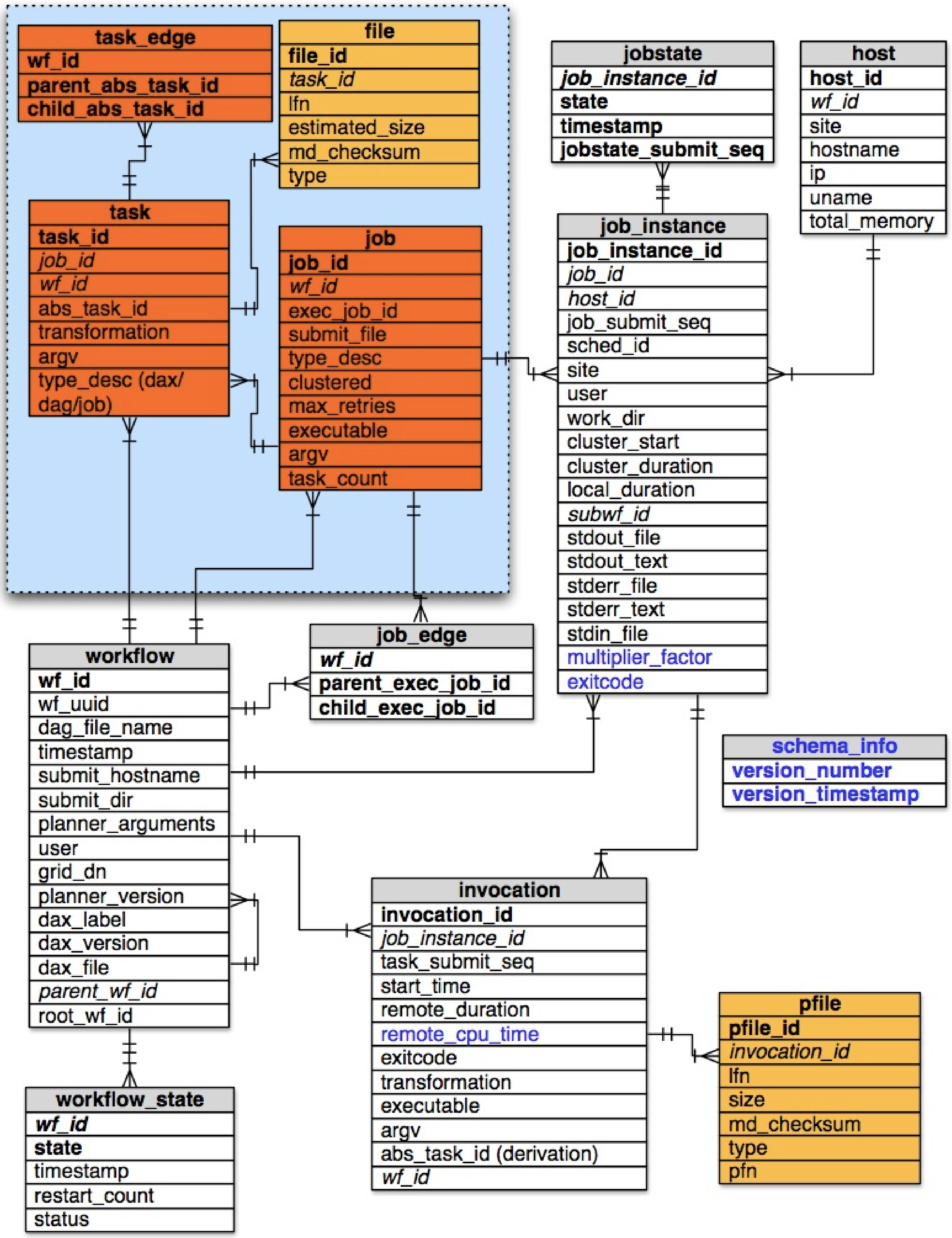
Stampede Database for 4.0

Table Of Contents:

Stampede Database Schema

Schema Picture in 3.2



The blue box with light and dark orange tables inside is for information provided by pegasus-plan.
The entries in blue are new tables or new keys in existing tables.

Download Omni Graffle File

Terms

- **Task** - A job appearing the DAX. An abstract job. Can be of type job|dax|dag
- **Job** - A job appearing in the executable workflow created by Pegasus. Can be of type compute, stagein, stageout etc
- **Job Instance** - Captures a job as run by Condor.
- **Invocation** - Information about how the job ran, on the remote host. PostScripts and Prescripts also appear invocations, associated with a Job Instance. To distinguish them, from kickstart invocations they have a negative task submit sequence number.

No Job Clustering and Kickstart Used

One Task maps to One Job.
That Job can have multiple Job Instances (dependant on DAGMan retrying the job in case of failure)
Each Job instance, will one main invocation associated with it for the corresponding dax task and additional postscript and prescript invocations (if specified)

Job Clustering and Kickstart Used

In case of job clustering with kickstart, we will have multiple tasks associated with one Job.
That Job can have multiple Job instances (dependant on DAGMan retrying the job in case of failure)
Each Job instance, will have mulitple invocations associated with it (one per task and additional postscript and prescript invocations (if specified))

No Job Clustering and Kickstart Not Used

One Task maps to One Job.
That Job can have multiple Job Instances (dependant on DAGMan retrying the job in case of failure)
Each Job instance, will one main invocation associated with it for the corresponding dax task and additional postscript and prescript invocations (if specified)
The main invocation associated with each Job will not have the detailed information that would (otherwise) be provided by Kickstart. Data comes from the submit file only.

Job Clustering and Kickstart Not Used

In case of job clustering without kickstart, we will have multiple tasks associated with one Job.
That Job can have multiple Job instances (dependant on DAGMan retrying the job in case of failure)
Each Job instance, will have only one main invocation associated with it (with the additional postscript and prescript invocations (if specified))
Although each job has multiple tasks associated with it, the main invocation for each job only has information coming from the submit file, and does not provided the high-level of detail information given when Kickstart is used.

Information Source for Each Table

task

==> Information comes from the DAX (Pegasus will generate events in the NetLogger format in a file in the submit directory)

task_id = autogenerated by the DB
job_id = from the job table
wf_id = from the workflow table
transformation = from dax
arguments = from dax
abs_task_id = from dax
tasktype (dax/dag/job) = from dax

Combination of wf_id and abstract_task_id is a candidate unique key.

task_edge

==> Information comes from the DAX file

wf_id = autogenerated from the workflow table
parent_abs_task_id = name of the parent task
child_abs_task_id = name of the child task

job

==> Information comes from Pegasus, directly in the NetLogger format in a file in the submit directory (it also appears in the Condor submit files)

job_id = autogenerated by the DB
wf_id = from the workflow table
exec_job_id = jobname (as given by Pegasus, unique within the scope of a workflow)
submit_file = a job's submit file (relative to the submit directory)
jobtype = internal Pegasus job classification
clustered = (boolean attribute) that indicates if a job consists of multiple tasks
max_retries = maximum number of times Condor retries a job that fails
executable = job's executable
arguments = job's arguements to the executable
task_count = number of tasks from the DAX mapped into this job by Pegasus

workflow

==> Information comes from braindump.txt file

wf_id = autogenerated by the database
wf_uuid = wf_uuid (generated by pegasus-plan)
dag_file_name = dag (basename of the dag file)
timestamp = pegasus_wf_time
submit_hostname = submit_hostname
submit_dir = submit_dir
planner_arguments = planner_arguments
user = user
grid_dn = grid_dn
planner_version = planner_version
dax_label = label
dax_version = dax_version
dax_index = dax_index
dax_file = dax
parent_wf_id = wf_id of parent workflow
root_wf_id = wf_id of the root workflow

workflow_state

==> Information comes from DAGMan

wf_id = from workflow table (autogenerated)
state = state of the workflow (start|stop)
restart_count = number of times the workflow has restarted (starts from 0)
timestamp = timestamp of the state change

job_instance (formerly known as the "job" table)

==> Information comes mainly from kickstart output file, but also from dagman.out file

job_instance_id = autogenerated by the database
job_id = key from the job table
host_id = key from the host table
job_submit_seq = integer <generated by pegasus-monitord, and guaranteed to be unique within a workflow>
sched_id = id in <condor id from dagman.out file>
site_name = <resource from invocation element>
remote_user = <user from invocation element>
remote_working_dir = <cwd element>
cluster_start_time = (only for clustered job, struct entry of .out file, start – from seqexec's point of view)
cluster_duration = (only for clustered job, struct entry of .out file, duration – from seqexec's point of view)
local_duration = calculated from DAGMan output, as JOB_TERMINATED - JOB_EXECUTED times
subwf_id = wf_uuid of the subworkflow started by this job (if applicable)
job_stdout = value of the output key in the condor submit file
job_stderr = value of the error key in the condor submit file
job_stdin = value of the input key in the condor submit file

jobstate

==> Same information that currently goes into jobstate.log file, obtained from dagman.out file

job_instance_id = from job_instance table (autogenerated)
state = from dagman.out file (3rd column of jobstate.log file)
timestamp = from dagman,out file (1st column of jobstate.log file)
jobstate_submit_seq = sequence of this state, generated by pegasus-monitord

host

==> Information from kickstart output file

host_id = autogenerated by the database
wf_id = root workflow id, filled in by the loader
site_name = <resource, from invocation element>
hostname = <hostname, from invocation element>
ip_address = <hostaddr, from invocation element>
uname = <combined (system, release, machine) from machine element>
total_ram = <ram_total from machine element>

==> Note that old kickstart records may not have a machine element, need to handle it gracefully

invocation

==> Information comes from kickstart output file

invocation_id = autogenerated here
job_instance_id = from job_instance table, autogenerated
task_submit_seq = generated by pegasus-monitord, counts tasks in the kickstart record
start_time = <start from mainjob element>
remote_duration = <duration, from mainjob element>
exitcode = <regular exitcode, from status element>
transformation = <transformation from invocation element>
executable = <file name in the mainjob's stat call>
arguments = <argument vector, joined by single space>
abs_task_id = <derivation, from invocation element> – this will be added by Karan
wf_id = autogenerated from the workflow table

job_edge

==> Information comes from the DAG file

wf_id = autogenerated from the workflow table
parent_exec_job_id = name of the parent job
child_exec_job_id = name of the child job

file

==> Information will come from kickstart output file

Sample NetLogger Events

As pegasus-monitord parses the various files in a workflow directory (braindump, workflow-map, dagman.out file), it will generate NetLogger events that can be used to populate a database using the Stampede schema. All events have the "stampede." prefix. Here are examples for each of these events:

stampede.workflow.plan event

ts=2010-10-12T17:43:23.000000Z event=stampede.workflow.plan level=Info parent.wf.id=None root.wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d submit_hostname=butterfly.isi.edu dax_label=diamond dax_index=0 planner_version=3.0.0cvs grid_dn=null user=prasanth submit_dir=/lfs1/prasanth/grid-setup/pegasus/3.0.0/examples/grid-blackdiamond/work/prasanth/pegasus/diamond/20101012T104323-0700 wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d planner_arguments="--dax /dax/diamond.dax --force --dir dags -s local -o local --nocleanup -v" dax_version=3.2 dax_file=/dax/diamond.dax

This event is generated when pegasus-monitord parses braindump.txt. The wf.id field is generated by Pegasus and is guaranteed to be unique. The ts field contains the timestamp the workflow was planned.

stampede.workflow.start event

ts=2011-10-12T17:43:26.000000Z event=stampede.workflow.start level=Info wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d restart_count=0

This event is generated by pegasus-monitord when it detects that DAGMan has started. The ts field contains the timestamp DAGMan started.

stampede.workflow.end event

ts=2011-10-12T18:06:57.000000Z event=stampede.workflow.end level=Info wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d restart_count=0

This event is generated by pegasus-monitord when it detects that DAGMan has finished. The ts field contains the timestamp DAGMan ended.

stampede.task event

ts=2011-10-12T17:43:26.000000Z event=stampede.task level=Info wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d transformation=diamond::preprocess:2.0 arguments="-a preprocess -T60 -i f.a -o f.b1 f.b2" abs_task.id=ID0000001 type=job

This event is generated by Pegasus during the planning phase, and is written to a file in the workflow's directory in NetLogger format. Pegasus-monitord reads this file when it enter the workflow directory and pipes its content to the loader.

stampede.task.map event

```
ts=2011-10-12T17:43:26.000000Z event=stampede.task.map level=Info wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d abs_task.id=ID0000001
exec_job.id=preprocess_ID00000001
```

This event is generated by Pegasus during the planning phase of a workflow, but will appear only after a task (or tasks) get mapped into a job.

stampede.job event

```
ts=2011-10-12T17:44:15.000000Z event=stampede.job level=Info wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d exec_job.
id=create_dir_diamond_0_ISIViz submit_file=create_dir_diamong_0_ISIViz.sub jobtype="create dir" clustered=False max_retries=3 executable=/opt
/pegasus/2.4/bin/kickstart arguments="-n pegasus::dirmanager -N pegasus::dirmanager:1.0 -R futuregrid -L diamond -T 2010-08-13T13:37:20-07:00 /opt
/pegasus/pegasus-3.0.0cvs/bin/dirmanager --create --dir /Users/voeckler/Pegasus/futuregrid/work/outputs/voeckler/pegasus/diamond/20100813T175039-
0700" task_count=0
```

This event is generated by Pegasus during the planning phase of a workflow, and contains a description of every job in the executable workflow. Jobs inserted by Pegasus, which do not have a mapped task from the DAX, will have its task_count set to 0.

stampede.task.edge

```
ts=2011-10-12T17:43:26.000000Z event=stampede.task.edge level=Info wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d parent_abs_task.id=ID0000001
child_abs_task.id=ID00000002
```

This event is generated by pegasus-monitord when it parses the file generated by Pegasus during the planning phase.

stampede.jobinstance.prescript.start event

```
ts=2010-02-20T23:25:28.000000Z event=stampede.jobinstance.prescript.start level=Info wf.id=wftest-id exec_job.id=pegasus-plan_ID000001 job.id=2
```

This event is generated by pegasus-monitord whenever it detects the start of a prescript for a new job. This event is similar to the stampede.jobinstance.mainjob.start event (see below), but it does not contain the sched_id field (as it is not yet assigned one). The ts field contains the timestamp the prescript started.

stampede.jobinstance.prescript.end event

```
ts=2010-02-20T23:14:11.000000Z event=stampede.jobinstance.prescript.finish level=Info wf.id=wftest-id exec_job.id=pegasus-plan_ID000001 job.id=2
```

This event is generated by pegasus-monitord whenever it detects the end of a prescript for a job. The ts field contains the timestamp the prescript ended.

stampede.jobinstance.mainjob.start event

```
ts=2011-10-12T17:43:40.000000Z event=stampede.jobinstance.mainjob.start level=Info wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d sched.id=388.0
job.id=1 exec_job.id=create_dir_diamond_0_ISIViz job_stdout=/workflow/run/create_dir_diamong_0_ISIViz.out job_stderr=/workflow/run
/create_dir_diamong_0_ISIViz.err job_stdin=None
```

The job.mainjob.start event is generated by pegasus-monitord every time a job is found in the dagman.out file. The job.id tag is generated by pegasus-monitord and starts in 1. The combination of wf_uuid and job.id guarantees an unique job. When a job begins, only certain information will be available. Later, when the job finishes, monitord will parse the kickstart output file and send the rest of the information in the job.mainjob.end event (see below). The ts field contains the timestamp the main job started.

stampede.jobinstance.mainjob.end event

```
ts=2011-10-12T17:44:15.000000Z event=stampede.jobinstance.mainjob.end level=Info remote_user=prasanth site_name=ISIViz exec_job.
id=create_dir_diamond_0_ISIViz remote_working_dir=/tmp job.id=1 wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d sched.id=388.0 job_stdout=/workflow
/run/create_dir_diamong_0_ISIViz.out job_stderr=/workflow/run/create_dir_diamong_0_ISIViz.err job_stdin=None cluster_start_time=None
cluster_duration=None local_duration=5.23 subwf.id=None
```

This event is generated by pegasus-monitord whenever a main job finishes. It contains all the remaining information for the job table (which comes from the kickstart output file) that was unavailable at the beginning of the job execution. The ts field contains the timestamp the main job ended.

stampede.jobinstance.postscript.start event

```
ts=2011-10-12T17:44:15.000000Z event=stampede.jobinstance.postscript.start level=Info wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d job.id=1
exec_job.id=create_dir_diamond_0_ISIViz
```

This event is generated by pegasus-monitord when it detects the start of the postscript for a given job. The ts field contains the timestamp the postscript started.

stampede.jobinstance.postscript.end event

```
ts=2011-10-12T17:44:20.000000Z event=stampede.jobinstance.postscript.end level=Info wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d job.id=1
exec_job.id=create_dir_diamond_0_ISIViz
```

This event is generated by pegasus-monitord when it detects the end of the postscript for a given job. The ts field contains the timestamp the postscript ended.

stampede.jobinstance.state event

```
ts=2011-10-12T17:44:15.000000Z event=stampede.jobinstance.state level=Info wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d state=POST_SCRIPT_STARTED job.id=1 exec_job.id=create_dir_diamond_0_ISIViz js.id=7
```

A job.state event is generated every time a job changes state (e.g. SUBMIT, then EXECUTE, then JOB_SUCCESS, ...). The ts field contains the timestamp the job state changed. The js.id field contains the state submit sequence for this particular state transition.

stampede.invocation.prescript, stampede.invocation.mainjob, stampede.invocation.postscript events

```
ts=2011-10-12T17:44:32.000000Z event=stampede.invocation.mainjob level=Info executable=/lfs1/prasanth/grid-setup/pegasus/default/bin/pegasus-transfer exec_job.id=stage_in_local_ISIViz_0 start_time=1286905467 job.id=2 remote_duration=2.008 task.id=1 arguments="" wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d transformation=pegasus::pegasus-transfer exitcode=0 abs_task.id=None
```

```
ts=2011-10-12T17:50:23.000000Z event=stampede.invocation.mainjob level=Info executable=/cluster-software/pegasus/testing/latest/bin/keg exec_job.id=preprocess_ID0000001 start_time=1286905586 job.id=3 remote_duration=60.813 task.id=1 arguments="-a preprocess -T60 -i f.a -o f.b1 f.b2" wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d transformation=diamond::preprocess:2.0 exitcode=0 abs_task.id=ID00000001
```

```
ts=2011-10-12T17:44:20.000000Z event=stampede.invocation.postscript level=Info executable=/lfs1/prasanth/grid-setup/pegasus/default/bin/pegasus-exitcode exec_job.id=create_dir_diamond_0_ISIViz start_time=1318441455 job.id=1 remote_duration=5 task.id=-2 arguments=" /lfs1/prasanth/grid-setup/pegasus/3.0.0/examples/grid-blackdiamond/work/prasanth/pegasus/diamond/20101012T104323-0700/create_dir_diamond_0_ISIViz.out" wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d transformation=dagman::post exitcode=0 abs_task.id=None
```

These three events are similar and indicate the termination of a mainjob (2 examples), or a postscript invocation. The ts field contains the timestamp the invocation ended. The task.id field contains the value -1 for prescript invocations, -2 for postscript invocations, and an integer (starting in 1) for each main job invocation. The abs_task.id is only populated for jobs that are in the dax (and not for pegasus-generated jobs, not pre and post-scripts).

host event

```
ts=2011-10-12T17:44:15.000000Z event=stampede.host level=Info job.id=1 site_name=ISIViz exec_job.id=create_dir_diamond_0_ISIViz total_ram=2124730368 hostname=viz-login.isi.edu uname=linux-2.6.18-194.3.1.el5-i686 wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d ip_address=128.9.72.178
```

This event is generated by pegasus-monitord whenever it parses a kickstart output file. In the case of clustered jobs (when there is more than 1 task in a mainjob), it is generated once per task. The ts field contains the timestamp the task associated with this host ended.

stampede.job.edge

```
ts=2011-10-12T17:43:26.000000Z event=stampede.job.edge level=Info wf.id=934cb609-ddd4-4b67-ad7a-886ae40fc94d parent_exec_job.id=stage_out_local_ISIViz_1_0 child_exec_job.id=clean_up_stage_out_local_ISIViz_1_0
```

This event is generated by pegasus-monitord when it parses the DAG file. It is used to populate the structure of the underlying dag.