Savanna

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 codar.savanna Namespace Reference

Namespaces

- consumer
- exc
- machines
- main
- model
- producer
- scheduler
- status

4.1.1 Detailed Description

```
Classes for running pipelines of MPI tasks based on a specified
total process limit. The system is designed to use two + N threads:

1. consumer thread: get pipelines from queue and execute them when process
   slots become available. Stops when a None pipeline is received.

2. producer thread: add pipelines to queue. Can be from file or from network
   service.

3. monitor threads: each process spawned by the consumer thread has a monitor
   thread that blocks on the processes completing with a timeout, and kills the
   process if it's not done after the timeout is reached.
```

4.2 codar.savanna.consumer Namespace Reference

Classes

- class PipelineRunner

4.2.1 Detailed Description

```
Classes for 'consuming' pipelines - running groups of MPI tasks based on a
specified total process limit.
```

4.3 codar.savanna.exc Namespace Reference

Classes

- class MachineNotFound
- class SavannaException

4.3.1 Detailed Description

```
Exceptions.
```

4.4 codar.savanna.machines Namespace Reference

Classes

- class Machine
- class MachineNode
- class SummitNode

Functions

- def **get_by_name** (name)

Variables

- **SCHEDULER_OPTIONS** = set(["project", "queue", "constraint", "license"])
- **local** = Machine('local', "local", "mpiexec", MachineNode, processes_per_node=1)
- **titan**
- **cori**
- **theta**
- **summit**

4.4.1 Detailed Description

```
Configuration for machines supported by Codar.
```

4.4.2 Variable Documentation

4.4.2.1 cori

codar.savanna.machines.cori

Initial value:

```
1 =  Machine('cori', "slurm", "srun", MachineNode,
2                processes_per_node=32, node_exclusive=True,
3                dataspaces_servers_per_node=4,
4                scheduler_options=dict(project="",
5                                       queue="debug",
6                                       constraint="haswell",
7                                       license="SCRATCH,project"))
```

4.4.2.2 summit

codar.savanna.machines.summit

Initial value:

```
1 =  Machine('summit', "ibm_lsf", "jsrun", SummitNode,
2                  processes_per_node=42, node_exclusive=True,
3                  scheduler_options=dict(project=""))
```

4.4.2.3 theta

codar.savanna.machines.theta

Initial value:

```
1 =  Machine('theta', "cobalt", "aprun", MachineNode,
2                processes_per_node=64, node_exclusive=True,
3                dataspaces_servers_per_node=8,
4                scheduler_options=dict(project="",
5                                       queue="debug-flat-quad"))
```

4.4.2.4 titan

codar.savanna.machines.titan

Initial value:

```
1 =  Machine('titan', "pbs", "aprun", MachineNode,
2                processes_per_node=16, node_exclusive=True,
3                scheduler_options=dict(project="", queue="debug"),
4                dataspaces_servers_per_node=4)
```

4.5 codar.savanna.main Namespace Reference

Functions

- def **parse_args** ()
- def **main** ()
- def **get_job_id** ()

Variables

- **consumer** = None

4.5.1 Detailed Description

```
Main program for executing workflow script with different producers and
runners.
```

4.6 codar.savanna.model Namespace Reference

Classes

- class NodeConfig
- class Pipeline
- class Run

Variables

- string **STDOUT_NAME** = 'codar.workflow.stdout'
- string **STDERR_NAME** = 'codar.workflow.stderr'
- string **RETURN_NAME** = 'codar.workflow.return'
- string **WALLTIME_NAME** = 'codar.workflow.walltime'
- int **KILL_WAIT** = 30
- int **WAIT_DELAY_KILL** = 30
- int **WAIT_DELAY_GIVE_UP** = 120

4.6.1 Detailed Description

```
Classes for tracking pipelines and the runs within each pipeline in separate
monitor threads that synchronize state.

Note that there is state tracked in these classes which is not available just
by looking at the return code. In particular, a run my be killed for several
different reasons: external signal, run timeout reached, other run in pipeline
failed (when kill on partial fail is set), or if the entire workflow is killed.

The goal here is to provide as much information as possible about why a
pipeline failed, to make an informed decision about whether it is worth
running again when the workflow is restarted, or if it's failure was more
permanent and not subject to outside forces like the job walltime expiring.
```

4.7 codar.savanna.producer Namespace Reference

Classes

- class JSONFilePipelineReader

4.7.1 Detailed Description

```
Classes for producing pipelines.
```

4.8 codar.savanna.scheduler Namespace Reference

Classes

- class JobList

4.8.1 Detailed Description

```
Classes related to finding a job that can run on available resources. Does
not assume any knowledge of how long each job will take. Designed for greedy
search of a job that will fit whenever resources are freed.

In the context of Cheetah workflows, it's unlikely that there will be more than
a few hundred jobs, so it's not worth optimizing the python search code very
much. It is however worth making sure that a job is run when resources are
available, since super computer resources are expensive. Basically it's worth
doing some work in python to make sure we start a big unit of work on compute
nodes.
```

4.9 codar.savanna.status Namespace Reference

Classes

- class PipelineState
- class WorkflowStatus

Variables

- string **NOT_STARTED** = 'not_started'
- string **RUNNING** = 'running'
- string **DONE** = 'done'
- string **KILLED** = 'killed'
- string **REASON_TIMEOUT** = 'timeout'
- string **REASON_FAILED** = 'failed'
- string **REASON_SUCCEEDED** = 'succeeded'
- string **REASON_EXCEPTION** = 'exception'
- string **REASON_NOFIT** = 'nofit'

4.9.1 Detailed Description

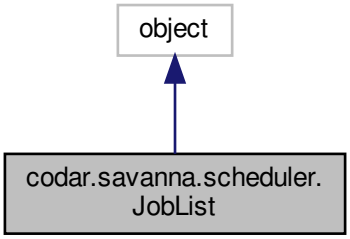
```
Class for maintaining state of all FOB runs that the workflow consumer is
managing. State is saved in a JSON file, overwritten on each state change.
```

Chapter 5

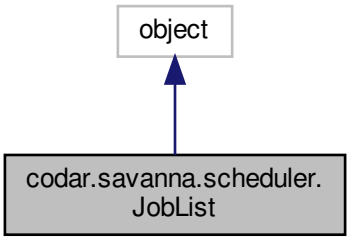
Class Documentation

5.1 codar.savanna.scheduler.JobList Class Reference

Inheritance diagram for codar.savanna.scheduler.JobList:



Collaboration diagram for codar.savanna.scheduler.JobList:



Public Member Functions

- def **__init__** (self, costfn, initial_jobs=None)
- def **add_job** (self, job)
- def pop_job (self, max_cost)
- def **__len__** (self)

5.1.1 Detailed Description

```
Manage a job list that can find and remove the highest cost job that
doesn't exceed max_cost and insert new jobs.

The job objects can be any type, but a key function must be provided
that takes an instance of a job and returns it's cost.

Uses a coordinated pair of sort list for costs and jobs, along with
the bisect module. A linked list might be more efficient, since the
list copy on insert and delete may dominate the time to do a linear
search of a small list, but it's likely fine either way for the
sizes we will encounter.
```

5.1.2 Member Function Documentation

5.1.2.1 pop_job()

```
def codar.savanna.scheduler.JobList.pop_job (
            self,
            max_cost )
```

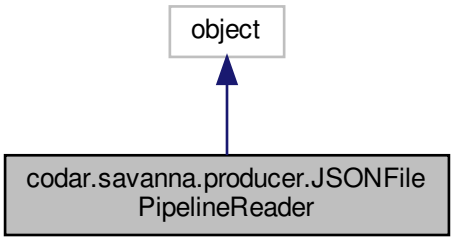
```
Get the highest cost job that doesn't exceed max_cost, and remove
it from the job list. Raises IndexError if the job list is empty,
returns None if no suitable jobs exist in the list.
```

The documentation for this class was generated from the following file:

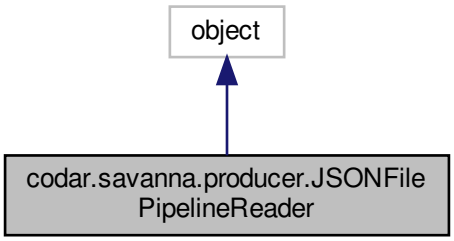
- scheduler.py

5.2 codar.savanna.producer.JSONFilePipelineReader Class Reference

Inheritance diagram for codar.savanna.producer.JSONFilePipelineReader:



Collaboration diagram for codar.savanna.producer.JSONFilePipelineReader:



Public Member Functions

- def **__init__** (self, file_path)
- def **read_pipelines** (self)

Public Attributes

- **file_path**

5.2.1 Detailed Description

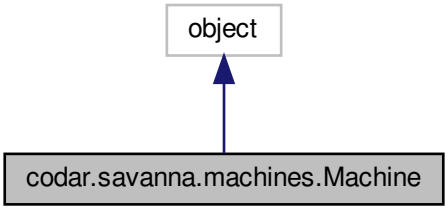
```
Load pipelines from a file formatted as a new line separated list of
JSON documents. Each JSON document must be a list containing dictionaries,
each dictionary discribing a code to run as part of the pipeline.
```

The documentation for this class was generated from the following file:

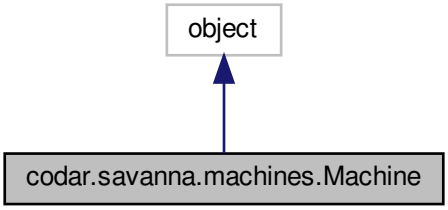
- producer.py

5.3 codar.savanna.machines.Machine Class Reference

Inheritance diagram for codar.savanna.machines.Machine:



Collaboration diagram for codar.savanna.machines.Machine:



Public Member Functions

- def **__init__** (self, name, scheduler_name, runner_name, node_class, processes_per_node=None, node_exclusive=False, scheduler_options=None, dataspaces_servers_per_node=1)
- def get_scheduler_options (self, options)
- def **get_nodes_reqd** (self)

Public Attributes

- **name**
- **scheduler_name**
- **runner_name**
- **node_class**
- **processes_per_node**
- **node_exclusive**
- **scheduler_options**
- **dataspaces_servers_per_node**

5.3.1 Detailed Description

```
Class to represent configuration of a specific Supercomputer or
workstation, including the scheduler and runner used by the machine.
This can be used to map an experiment to run on the machine without
having to define machine specific parameter for every experiment
separately.
```

5.3.2 Member Function Documentation

5.3.2.1 get_scheduler_options()

```
def codar.savanna.machines.Machine.get_scheduler_options (
              self,
              options )
```

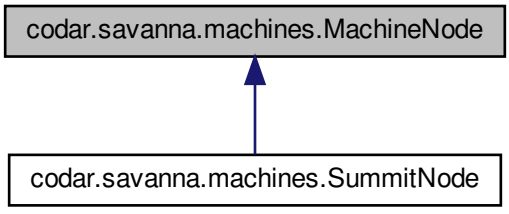
```
Validate supplied options and add default values where missing.
Returns a new dictionary.
```

The documentation for this class was generated from the following file:

- machines.py

5.4 codar.savanna.machines.MachineNode Class Reference

Inheritance diagram for codar.savanna.machines.MachineNode:



Public Member Functions

- def **__init__** (self, num_cpus, num_gpus)
- def **validate_layout** (self)
- def **to_json** (self)

Public Attributes

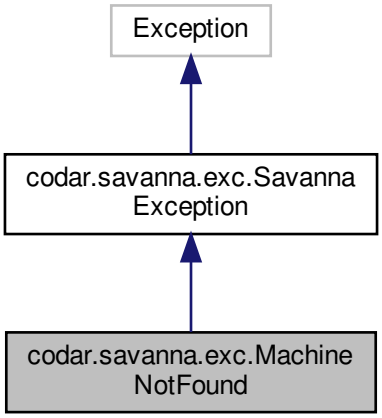
- **cpu**
- **gpu**

The documentation for this class was generated from the following file:

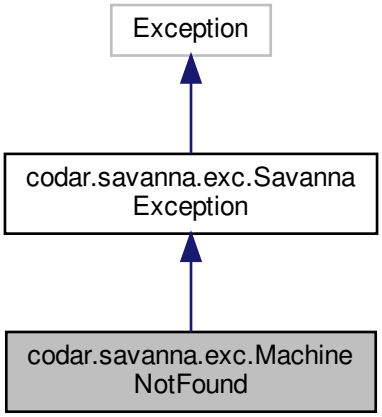
- machines.py

5.5 codar.savanna.exc.MachineNotFound Class Reference

Inheritance diagram for codar.savanna.exc.MachineNotFound:



Collaboration diagram for codar.savanna.exc.MachineNotFound:



Public Member Functions

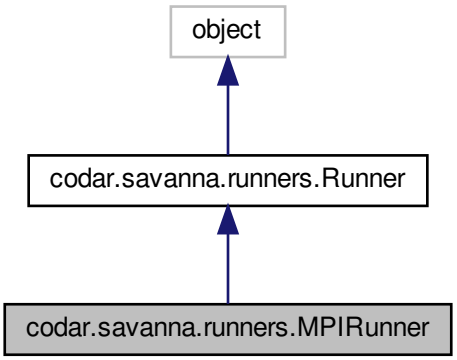
- def **__init__** (self, machine_name)

The documentation for this class was generated from the following file:

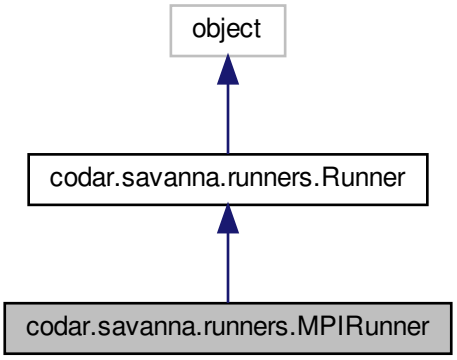
- exc.py

5.6 codar.savanna.runners.MPIRunner Class Reference

Inheritance diagram for codar.savanna.runners.MPIRunner:



Collaboration diagram for codar.savanna.runners.MPIRunner:



Public Member Functions

- def **__init__** (self, exe, nprocs_arg, nodes_arg=None, tasks_per_node_arg=None, hostfile=None)
- def **wrap** (self, run, sched_args, find_in_path=True)

Public Attributes

- **exe**
- **nprocs_arg**
- **nodes_arg**
- **tasks_per_node_arg**
- **hostfile**

The documentation for this class was generated from the following file:

- runners.py

5.7 codar.savanna.model.NodeConfig Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- **num_ranks_per_node**
- **cpu**
- **gpu**

5.7.1 Constructor & Destructor Documentation

5.7.1.1 __init__()

```
def codar.savanna.model.NodeConfig.__init__ (
              self )
```

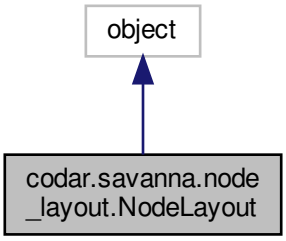
```
Intended to look like
cpu = [ 0=[], 1=[], 2=[], 3=[] ]
gpu = [ 0=[], 1=[], 2=[], 3=[] ]
```

The documentation for this class was generated from the following file:

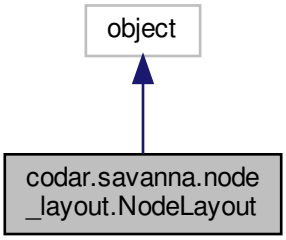
- model.py

5.8 codar.savanna.node_layout.NodeLayout Class Reference

Inheritance diagram for codar.savanna.node_layout.NodeLayout:



Collaboration diagram for codar.savanna.node_layout.NodeLayout:



Public Member Functions

- def **__init__** (self, layout_list)
- def add_node (self, node_dict)
- def get_node_containing_code (self, code)
- def **codes_per_node** (self)
- def **shared_nodes** (self)
- def **ppn** (self)
- def validate (self, ppn, codes_per_node, shared_nodes)
- def **as_data_list** (self)
- def serialize_to_dict (self)
- def **copy** (self)
- def group_codes_by_node (self)
- def **populate_remaining** (self, rc_names, ppn)
- def default_no_share_layout (cls, ppn, code_names)

Public Attributes

- **layout_list**
- **layout_map**

5.8.1 Detailed Description

```
Class representing options on how to organize a multi-exe task across
many nodes. It is the scheduler model's job to take this and produce the
correct scheduler and runner options to make this happen, or raise an error
if it's not possible. Note that this will generally be different for each
machine unless it is very simple and suppored uniformly by all desired
machines.

A layout is represented as a list of dictionaries, where each dictionary
described codes to be run together on a single node. The keys are
the names of the codes, and the values are the number of processes to
assign to each.
```

5.8.2 Member Function Documentation

5.8.2.1 add_node()

```
def codar.savanna.node_layout.NodeLayout.add_node (
              self,
              node_dict )
```

```
Add a node to an existing layout, e.g. add sosflow.
```

5.8.2.2 default_no_share_layout()

```
def codar.savanna.node_layout.NodeLayout.default_no_share_layout (
              cls,
              ppn,
              code_names )
```

```
Create a layout object for the specified codes and ppn, where each
code uses max procs on it's own node.
```

5.8.2.3 get_node_containing_code()

```
def codar.savanna.node_layout.NodeLayout.get_node_containing_code (
              self,
              code )
```

```
Get node dict containing the specified code. Raises KeyError if
not found.
```

5.8.2.4 group_codes_by_node()

```
def codar.savanna.node_layout.NodeLayout.group_codes_by_node (
              self )
```

```
Return a list of dicts, where each list represents codes on a
node, and a dict key for ppn
Example: [ {sim,analysis1}, {analysis2}, {viz} ].
Must take Summit NodeConfigs into account
```

5.8.2.5 serialize_to_dict()

```
def codar.savanna.node_layout.NodeLayout.serialize_to_dict (
              self )
```

```
Get a copy of the data list passed to the constructor,
suitable for JSON serialization.
```

5.8.2.6 validate()

```
def codar.savanna.node_layout.NodeLayout.validate (
              self,
              ppn,
              codes_per_node,
              shared_nodes )
```

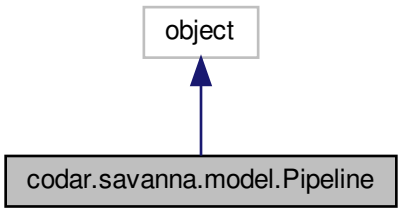
```
Given a machine ppn and max numer of codes (e.g. 4 on cori),
raise a ValueError if the specified layout won't fit.
Dont modify this yet, this is being used by the tests
```

The documentation for this class was generated from the following file:

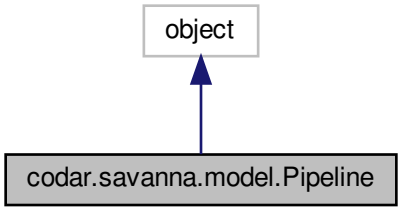
- node_layout.py

5.9 codar.savanna.model.Pipeline Class Reference

Inheritance diagram for codar.savanna.model.Pipeline:



Collaboration diagram for codar.savanna.model.Pipeline:



Public Member Functions

- def **__init__** (self, pipe_id, runs, working_dir, total_nodes, machine_name, kill_on_partial_failure=False, post_process_script=None, post_process_args=None, post_process_stop_on_failure=False, node_↩ layout=None, launch_mode=None)
- def from_data (cls, data)
- def **start** (self, consumer, nodes_assigned, runner=None)
- def **run_finished** (self, run)
- def **run_post_process_script** (self)
- def **add_done_callback** (self, fn)
- def **remove_done_callback** (self, fn)
- def **add_fatal_callback** (self, fn)
- def **remove_fatal_callback** (self, fn)
- def **get_nodes_used** (self)
- def set_ppn (self, ppn)
- def set_total_nodes (self)
- def **get_state** (self)
- def **get_pids** (self)
- def force_kill_all (self)
- def **join_all** (self)

Public Attributes

- **id**
- **runs**
- **working_dir**
- **kill_on_partial_failure**
- **post_process_script**
- **post_process_args**
- **post_process_stop_on_failure**
- **node_layout**
- **machine_name**
- **done_callbacks**
- **fatal_callbacks**
- **total_procs**
- **log_prefix**
- **total_nodes**
- **launch_mode**
- **nodes_assigned**

5.9.1 Member Function Documentation

5.9.1.1 force_kill_all()

```
def codar.savanna.model.Pipeline.force_kill_all (
              self )
```

```
Kill all runs and don't run post processing. Note that this call may
block waiting for all runs to be started, to avoid confusing races.
If the pipeline is already done, this does nothing. If one or more
runs are still active, or have not yet been marked as finished, then
it will mark the entire pipeline as killed so it can be re-run from
scratch on a restart if desired.
```

5.9.1.2 from_data()

```
def codar.savanna.model.Pipeline.from_data (
              cls,
              data )
```

```
Create Pipeline instance from dictionary data structure, containing
at least "id" and "runs" keys. The "runs" key must have a list of dict,
and each dict is parsed using Run.from_data.
Raises KeyError if a required key is missing.
```

5.9.1.3 set_ppn()

```
def codar.savanna.model.Pipeline.set_ppn (
              self,
              ppn )
```

```
Determine number of nodes needed to run pipeline with the specified
node layout or full occupancy layout with ppn. Also updates runs
to set node and task per node counts.
TODO: This should be set by Cheetah in fobs.json
```

5.9.1.4 set_total_nodes()

```
def codar.savanna.model.Pipeline.set_total_nodes (
              self )
```

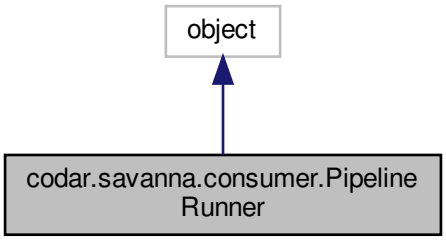
```
To be deprecated
```

The documentation for this class was generated from the following file:

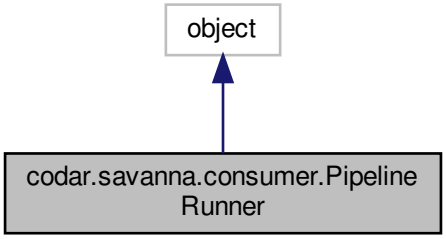
- model.py

5.10 codar.savanna.consumer.PipelineRunner Class Reference

Inheritance diagram for codar.savanna.consumer.PipelineRunner:



Collaboration diagram for codar.savanna.consumer.PipelineRunner:



Public Member Functions

- def **__init__** (self, runner, max_nodes, machine_name, processes_per_node, status_file=None)
- def **add_pipeline** (self, p)
- def stop (self)
- def kill_all (self)
- def run_finished (self, run)
- def pipeline_finished (self, pipeline)
- def **pipeline_fatal** (self, pipeline)
- def run_pipelines (self)

Public Attributes

- **max_nodes**
- **machine_name**
- **ppn**
- **runner**
- **job_list_cv**
- **job_list**
- **free_cv**
- **free_nodes**
- **pipelines_lock**
- **pipelines**
- **allocated_nodes**

5.10.1 Detailed Description

```
Runner that assumes a homogonous set of nodes. Now only support only
node based limiting (although process limiting can be emulated by setting
process_per_node=1 and max_nodes=max_procs).

Threading model: assumes there could be multiple producer threads calling
add_pipeline, e.g. if using a dynamic job submission model based on
results of previous jobs. Pipelines and each Run in a pipeline are all
executed in separate threads, so their notification callbacks execute in
separate threads, and their threads must be joined before exiting. The
stop and kill_all methods could be called from any of the producer,
Pipeline or Run threads.
```

5.10.2 Member Function Documentation

5.10.2.1 kill_all()

```
def codar.savanna.consumer.PipelineRunner.kill_all (
              self )
```

Kill all running processes spawned by this consumer and don't
start any new processes.

5.10.2.2 pipeline_finished()

```
def codar.savanna.consumer.PipelineRunner.pipeline_finished (
              self,
              pipeline )
```

Monitor thread(s) should call this as pipelines complete.

5.10.2.3 run_finished()

```
def codar.savanna.consumer.PipelineRunner.run_finished (
              self,
              run )
```

TO BE DEPRECATED.
Monitor thread(s) should call this as runs
complete. To be deprecated, as the functionality fails when
node_layout is set to node-sharing.

This means that for node_exclusive, resources held by a run are not
released when the run terminates. For kill_on_partial_failure=False,
this could lead to unused resources, which is ok.

5.10.2.4 run_pipelines()

```
def codar.savanna.consumer.PipelineRunner.run_pipelines (
              self )
```

Main loop of consumer thread. Does not return until all child
threads are complete.

5.10.2.5 stop()

```
def codar.savanna.consumer.PipelineRunner.stop (
              self )
```

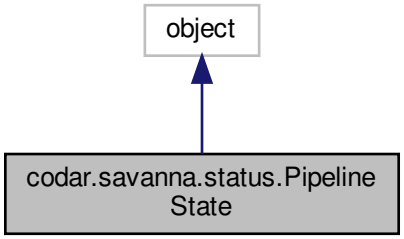
```
Signal to stop when all pipelines are finished. Don't allow adding
new pipelines.
```

The documentation for this class was generated from the following file:

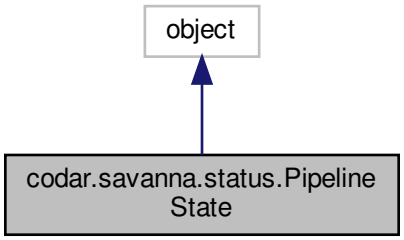
- consumer.py

5.11 codar.savanna.status.PipelineState Class Reference

Inheritance diagram for codar.savanna.status.PipelineState:



Collaboration diagram for codar.savanna.status.PipelineState:



Public Member Functions

- def **__init__** (self, pipeline_id, state, reason=None, return_codes=None)
- def **as_data** (self)

Public Attributes

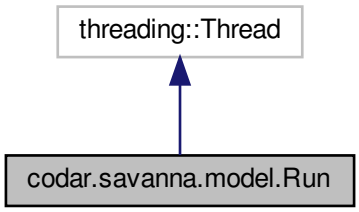
- **id**
- **state**
- **reason**
- **return_codes**

The documentation for this class was generated from the following file:

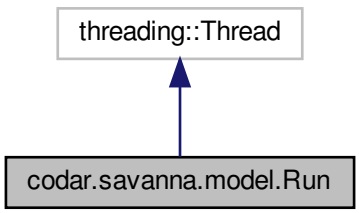
- status.py

5.12 codar.savanna.model.Run Class Reference

Inheritance diagram for codar.savanna.model.Run:



Collaboration diagram for codar.savanna.model.Run:



Public Member Functions

- def **__init__** (self, name, exe, args, sched_args, env, working_dir, timeout=None, nprocs=1, res_set=None, stdout_path=None, stderr_path=None, return_path=None, walltime_path=None, log_prefix=None, sleep_↩ after=None, depends_on_runs=None, hostfile=None, runner_override=False)
- def from_data (cls, data)
- def **mpmd_run** (cls, runs)
- def **set_runner** (self, runner)
- def timed_out (self)
- def killed (self)
- def exception (self)
- def succeeded (self)
- def add_callback (self, fn)
- def **remove_callback** (self, fn)
- def **run** (self)
- def kill (self)
- def **get_returncode** (self)
- def **get_pid** (self)
- def **close** (self)
- def **join** (self)
- def get_nodes_used (self)
- def **create_node_config** (self)

Public Attributes

- **name**
- **exe**
- **args**
- **sched_args**
- **env**
- **working_dir**
- **timeout**
- **nprocs**
- **res_set**
- **stdout_path**
- **stderr_path**
- **return_path**
- **walltime_path**
- **sleep_after**
- **log_prefix**
- **runner**
- **callbacks**
- **nodes**
- **tasks_per_node**
- **depends_on_runs**
- **hostfile**
- **machine**
- **nodes_assigned**
- **node_config**
- **erf_file**
- **runner_override**

5.12.1 Detailed Description

Manage running a single executable within a pipeline. When start is called, it will launch the process with Popen and call wait in the new thread with a timeout, killing if the process does not finish in time.

5.12.2 Member Function Documentation

5.12.2.1 add_callback()

```
def codar.savanna.model.Run.add_callback (
              self,
              fn )
```

Function takes single argument which is this run instance, and is called when the process is complete (either normally or killed by timeout). Callbacks must not block.

5.12.2.2 exception()

```
def codar.savanna.model.Run.exception (
              self )
```

True if there was a python exception in the run method. When this is the case, the state of the underlying process is unknown - it may have been started or not.

5.12.2.3 from_data()

```
def codar.savanna.model.Run.from_data (
              cls,
              data )
```

Create Run instance from nested dictionary data structure, e.g. parsed from JSON. The keys 'name', 'exe', 'args' are required, all the other keys are optional and have the same names as the constructor args. Raises KeyError if a required key is missing.

5.12.2.4 get_nodes_used()

```
def codar.savanna.model.Run.get_nodes_used (
              self )
```

Get number of nodes needed to run this app. Requires that the
pipeline set_ppn method has been called to set this and tasks_per_node
on each run.

5.12.2.5 kill()

```
def codar.savanna.model.Run.kill (
              self )
```

Kill process and cause run thread to complete after the wait
returns. If the run is already done, does nothing. If the process is
killed, it will mark the state as killed so it can be re-run on
workflow restart. Thread safe.

5.12.2.6 killed()

```
def codar.savanna.model.Run.killed (
              self )
```

True if the run is done and the kill method was called. Note that
this will _NOT_ be true if an external kill signal caused the process
to exit. Raises ValueError if the run is not complete.

5.12.2.7 succeeded()

```
def codar.savanna.model.Run.succeeded (
              self )
```

True if the run is done, finished normally, and had 0 return value.
Raises ValueError if the run is not complete.

5.12.2.8 timed_out()

```
def codar.savanna.model.Run.timed_out (
              self )
```

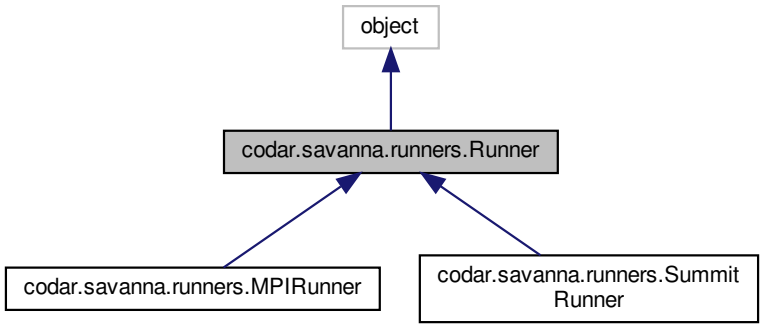
```
True if the run is done and was killed because it exceeded the
specified run timeout. Raises ValueError if the run is not complete.
```

The documentation for this class was generated from the following file:

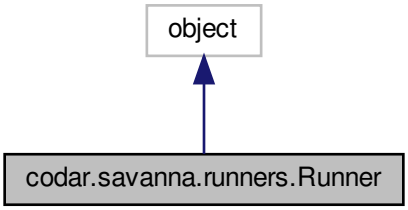
- model.py

5.13 codar.savanna.runners.Runner Class Reference

Inheritance diagram for codar.savanna.runners.Runner:



Collaboration diagram for codar.savanna.runners.Runner:



Public Member Functions

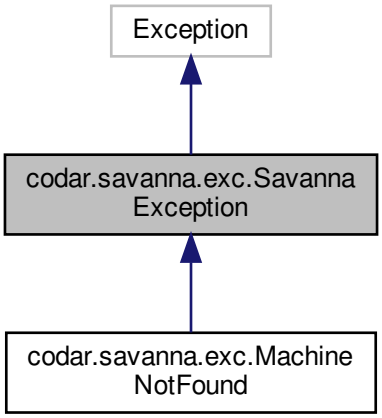
- def **wrap** (self, run, sched_args)

The documentation for this class was generated from the following file:

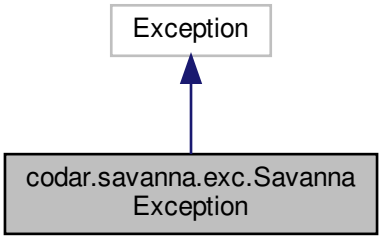
- runners.py

5.14 codar.savanna.exc.SavannaException Class Reference

Inheritance diagram for codar.savanna.exc.SavannaException:



Collaboration diagram for codar.savanna.exc.SavannaException:

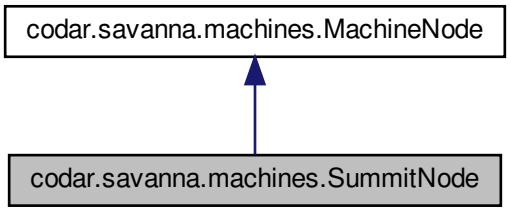


The documentation for this class was generated from the following file:

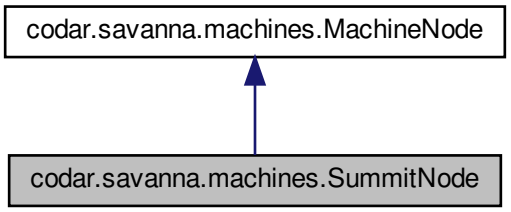
- exc.py

5.15 codar.savanna.machines.SummitNode Class Reference

Inheritance diagram for codar.savanna.machines.SummitNode:



Collaboration diagram for codar.savanna.machines.SummitNode:



Public Member Functions

- def **__init__** (self)
- def validate_layout (self)
- def **to_json** (self)

Additional Inherited Members

5.15.1 Member Function Documentation

5.15.1.1 validate_layout()

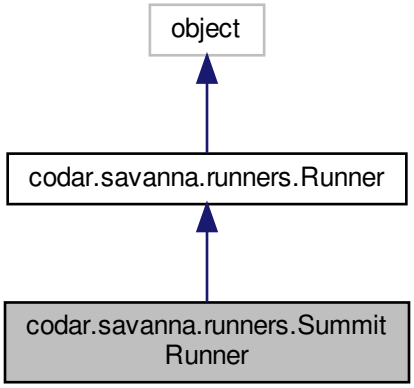
```
def codar.savanna.machines.SummitNode.validate_layout (
              self )

Check that 1) the same rank of the same code is not repeated,
2) a gpu is not mapped to multiple executables.
```

The documentation for this class was generated from the following file:

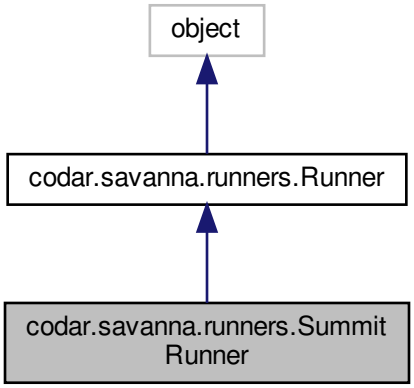
- machines.py

5.16 codar.savanna.runners.SummitRunner Class Reference

Inheritance diagram for codar.savanna.runners.SummitRunner:



Collaboration diagram for codar.savanna.runners.SummitRunner:



Public Member Functions

- def **__init__** (self)
- def **wrap** (self, run, sched_args)
- def wrap_deprecated (self, run, jsrun_opts, find_in_path=True)

Public Attributes

- **exe**
- **nrs_arg**
- **tasks_per_rs_arg**
- **cpus_per_rs_arg**
- **gpus_per_rs_arg**
- **rs_per_host_arg**
- **launch_distribution_arg**
- **bind_arg**
- **machine**

5.16.1 Member Function Documentation

5.16.1.1 wrap_deprecated()

```
def codar.savanna.runners.SummitRunner.wrap_deprecated (
              self,
              run,
              jsrun_opts,
              find_in_path = True )
```

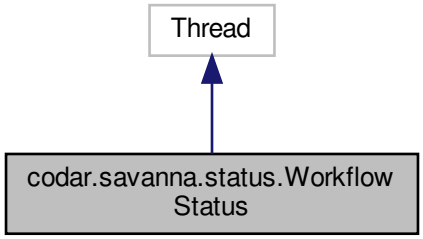
```
This function is deprecated in favor of the above that uses erf
files
```

The documentation for this class was generated from the following file:

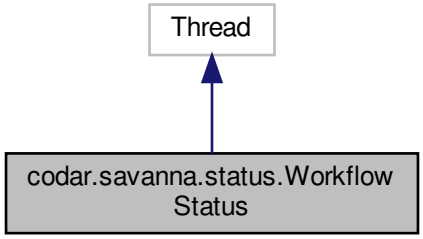
- runners.py

5.17 codar.savanna.status.WorkflowStatus Class Reference

Inheritance diagram for codar.savanna.status.WorkflowStatus:



Collaboration diagram for codar.savanna.status.WorkflowStatus:



Public Member Functions

- def **__init__** (self, file_path)
- def **set_state** (self, pipeline_state)

Public Attributes

- **file_path**

The documentation for this class was generated from the following file:

- status.py

Index