IRMaSim Development Notes

Esteban Stafford

November 11, 2022

Simulator

- energy : int
- job_limits : dict
- job_queue : JobQueue
- logger : RootLogger, NoneType, Logger
- platform
- simulation_time : int
- workload_manager

build_platform()
build_workload_manager()
generate_workload()
get_next_step(): float
get_resources(resource_type: type)
get_resources_ids()
log_state()
reap(tasks: list)
schedule(tasks: list)
start_simulation(): None

WorkloadManager

- simulator : str

on_end_simulation()
on_end_step()
on_job_completion(jobs: list)
on_job_submission(jobs: list)
schedule_next_job()

Policy

- agent
- environment : Environment
- flow_flags : dict
- last_reward : bool
- last_time : int
- load_agent : bool
- optimizer : Adam
- pending_jobs : list
- resources
- reward

apply_policy(action: int)
create_agent()
on_end_simulation()
on_end_step()
on_job_completion(jobs: list)
on_job_submission(jobs: list)

JobQueue

- finished_jobs : list
- future_jobs : list
- submitted_jobs : list

add_job(job: Job)
add_jobs(jobs: list)
finish_jobs()
get_job_counts()
get_next_jobs(now: float)
get_next_step()

Minimal

- idle_resources
- pending_jobs : list
- resources
- running_jobs : list

allocate(task: Task)
deallocate(task: Task)
on_end_simulation()
on_end_step()
on_job_completion(jobs: list)
on_job_submission(jobs: list)
schedule_next_job()

NodeWM

- idle_resources
- pending_jobs : list
- resources
- running_jobs : list

allocate(task: Task)
deallocate(task: Task)
on_end_simulation()
on_end_step()
on_job_completion(jobs: list)
on_job_submission(jobs: list)
schedule_next_job()

Resource

- children : list
- config : dict
- id : str
- parent : NoneType

add_child(child: 'Resource')
count_resources()
enumerate_ids(parent_id: list)
enumerate_resources(resource_type)
find_child(id: str)
full_id()
get_parent()
pstr(indent: str, current: str)

EnergyConsumer

get_joules(delta_time: float)

Environment

- action_size
- action_space : Discrete
- actions : list
- core_selections : OrderedDict
- env_options
- job_selections : OrderedDict
- last_job_queue_length : int, NoneType
- observation : partial
- observation_size
- observation_space : Box
- queue_sensitivity
- reward
- simulator : Simulator
- with_void : bool
- workload_manager : str

edp_reward(): float
energy_consumption_reward(): float
makespan_reward(): float

Agent

- rewards : list

alter(action: int, environment): None
decide(observation: np.ndarray): int
observe(environment): np.ndarray
rewarded(environment): None

Job

- finish_time : float
- id : int
- memory
- memory_vol
- name : str
- opc
- ops
- profile : dict
- req_time
- resources : int
- start_time : float
- submit_time : float
- tasks : list, NoneType
- type : str

generate_tasks()
is_job_finished()
set_start_time(time: float)

TaskRunner

advance(delta_time: float)
get_joules(delta_time: float)
get_next_step()
reap(task: Task, resource_id: list)
schedule(task: Task, resource_id: list)

ClassicAgent

- actionId : int

decide(observation: np.ndarray): int

LearningAgent

- options : dict

forward(observation: np.ndarray): typing.Any
loss(): float
transform_rewards(): list

PolicyLearningAgent

- log_probs : list
- probs : NoneType

decide(observation: np.ndarray): int
forward(observation: np.ndarray): torch.Tensor
forward_policy(observation: np.ndarray): torch.Tensor
get_action(probabilities: torch.Tensor): int
loss(): float
policy_loss(rews_or_advs: list): list
save_log_prob(log_prob: torch.Tensor): None

ValueLearningAgent

- values : list

forward(observation: np.ndarray): torch.Tensor
forward_value(observation: np.ndarray): torch.Tensor
loss(): float
save_value(value: torch.Tensor): None
value_loss(rews: list): list

ActorCritic

- actor_hidden_0 : Linear
- actor_hidden_1 : Linear
- actor_hidden_2 : Linear
- actor_output : Linear
- critic_hidden_0 : Linear
- critic_hidden_1 : Linear
- critic_hidden_2 : Linear
- critic_output : Linear
- input : Linear
- probs

calculate_advantages(trewards: list): list
decide(observation: np.ndarray): int
forward(observation: np.ndarray): tuple
forward_policy(observation: np.ndarray): torch.Tensor
forward_value(observation: np.ndarray): torch.Tensor
loss(): float

BasicNode

BasicCluster

BasicProcessor

BasicCore

Node (models.modelV1)

- aaa
- aab
- aba
- abb
- baa
- bab
- bba
- bbb
- caa
- cab
- cba
- cbb
- cca
- ccb
- daa
- dab
- dba
- dbb
- gops
- p00
- p01
- p02
- p10
- p11
- p20
- pidle
- processes : list
- requested_memory_bandwidth : int

advance(delta_time: float)
get_joules(delta_time: float)
get_next_step()
reap(task: Task, resource_id: list)
schedule(task: Task, resource_id: list)
update_speedup()

Processor

- mops_per_core
- requested_memory_bandwidth : float

advance(delta_time: float)
get_joules(delta_time: float)
reap(task: Task, resource_id: list)
schedule(task: Task, resource_id: list)
update_speedup()

Core

- b
- c
- da
- db
- dc
- dd
- dynamic_power
- min_power
- mops
- requested_memory_bandwidth : int, float
- speedup : float
- state : dict
- static_power
- task : NoneType, Task

advance(delta_time: float)
get_next_step()
reap(task: Task, resource_id: list)
schedule(task: Task, resource_id: list)

Node

Cluster

Process (models.modelV2)

- node : str
- requested_memory_bandwidth
- speedup : int
- task : Task

advance(delta_time: float)
get_next_step()
update_speedup()

Cluster

Task

- job : str
- memory : float
- memory_volume : float
- opc : float
- ops : float
- resource : NoneType, list

allocate(resource: list)

JobMapper

Options

get()

Statistics

- edp : list
- energy : list
- latest_edp : list
- latest_energy : list
- options

calculate_energy_and_edp(core_pool: list, diff_time: float, all_jobs_scheduler)
write_results(time: float, finished_jobs: list)