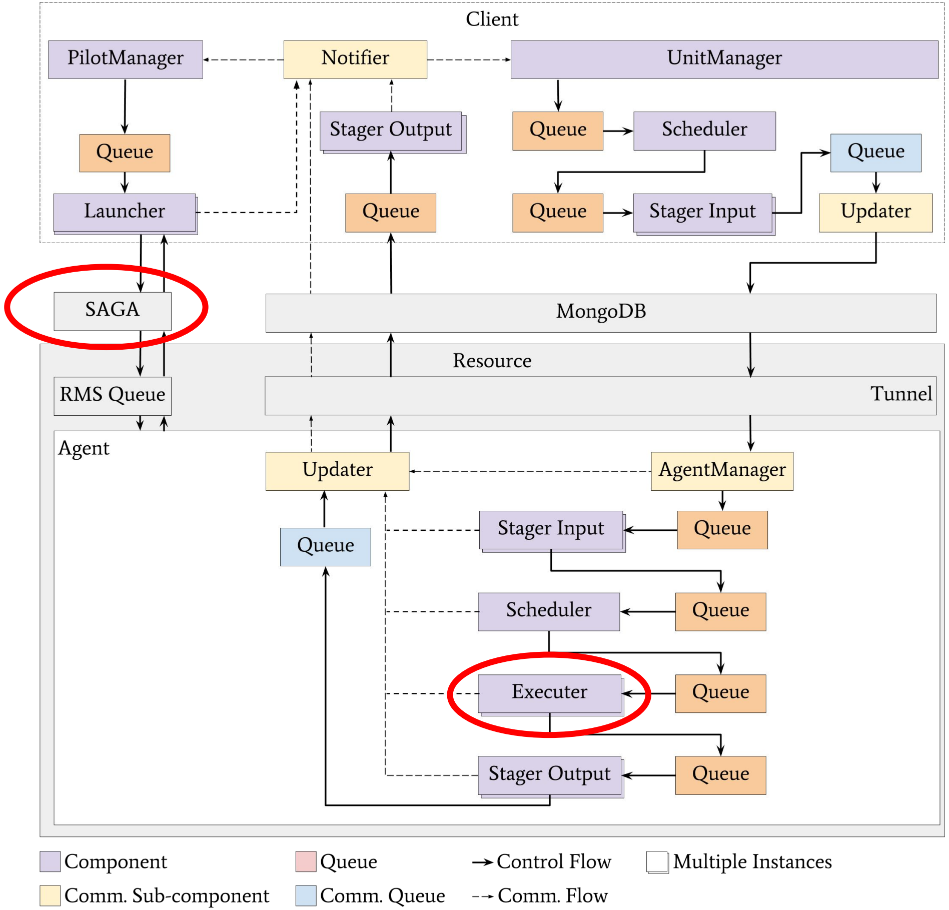
NGE on Summit and Harvester Integration

ATLAS WFM SW weekly meeting
14/03/2019

Next Generation Executor (NGE) Motivation

- Run multiple tasks **concurrently** and **consecutively** in a **SINGLE** batch job:
  - Tasks are programs, i.e., executables, not methods, functions, threads, or processes.
  - Tasks are executed within the scope of the batch job, i.e., walltime is still binding.
- Late binding:
  - Tasks are NOT packaged into the batch job before submission.
  - Tasks are scheduled and then placed within the batch job at runtime.
- Task and resource heterogeneity:
  - Scheduling, placing and running CPU/GPU/OpenMM/MPI tasks in the same batch job
  - Use single/multiple CPU/GPU for the same tasks and/or across multiple tasks.
- Use cases:
  - Molecular dynamics, HEP, and any other use case requiring multiple tasks on Summit.
- Current limitations:
  - Still a prototype: API needs to be extended depending on the integration with Harvester
  - Early adoption of Summit: reliability and scaling still under evaluation.

NGE and RADICAL-Pilot on Summit



- NGE:
  - REST API for RADICAL-Pilot
  - No code rewriting/porting needed.
- RADICAL-Pilot:
  - Writing dedicated launcher subsystems. Focussing on JSRUN first but experimenting also with PRRTE.
- RADICAL-SAGA:
  - RS to LSF. Required reviving old code in the LSF adaptor.

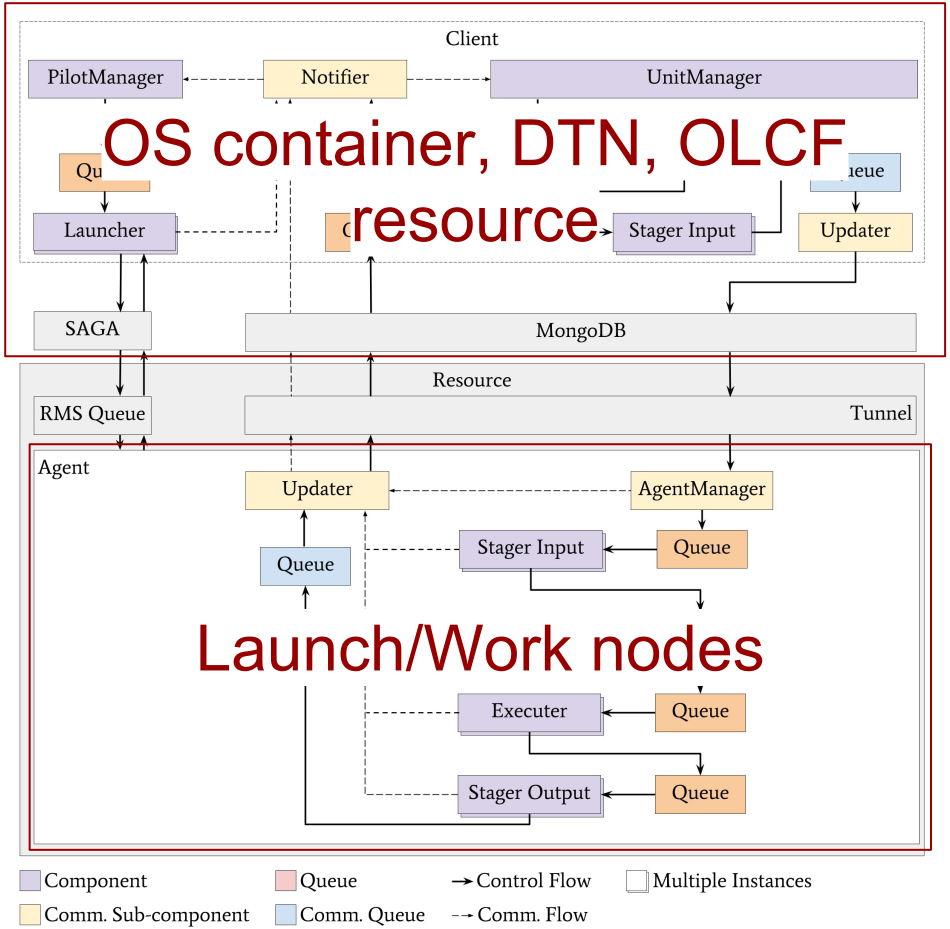
Code available at:

NGE: https://github.com/radical-cybertools/radical.nge

RP: https://github.com/radical-cybertools/radical.pilot

RS: https://github.com/radical-cybertools/saga-python

Deployment of NGE and RADICAL-Pilot on Summit



- NGE:
  - OpenShift container, DTN or any linux-based machine within OLCF.
- RADICAL-Pilot:
  - Client: same as NGE.
  - Agent: MOM node + work nodes.
  - MongoDB: OpenShift container.
- RADICAL-SAGA:
  - Same as NGE.

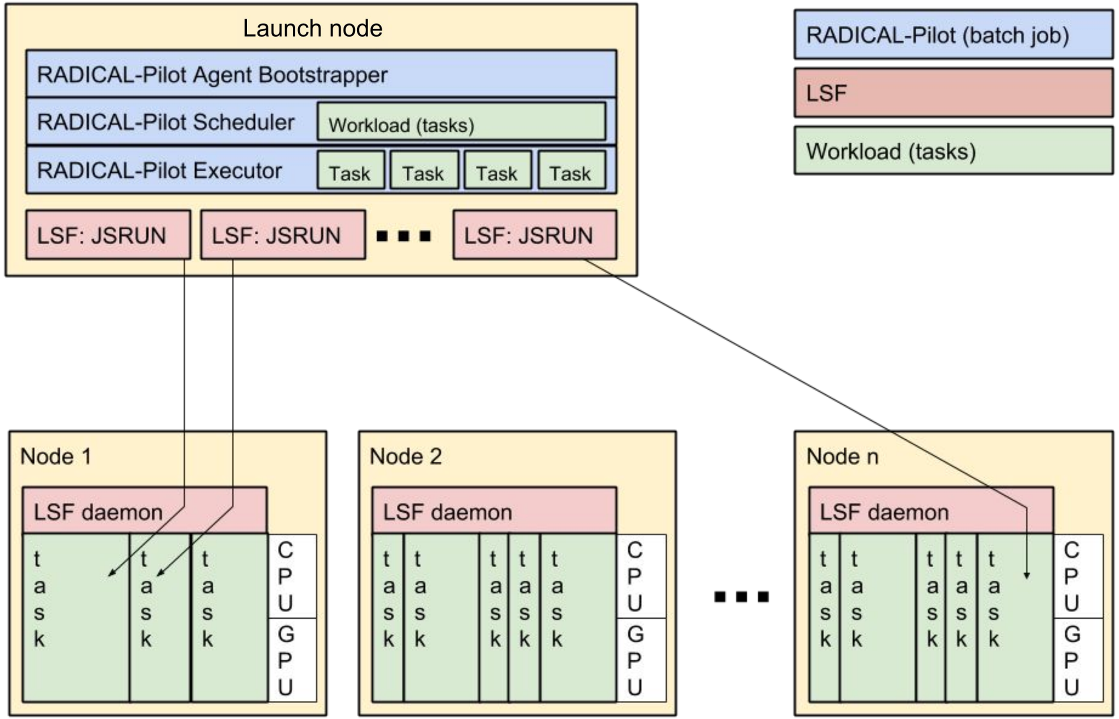
Code available at:

NGE: https://github.com/radical-cybertools/radical.nge

RP: https://github.com/radical-cybertools/radical.pilot

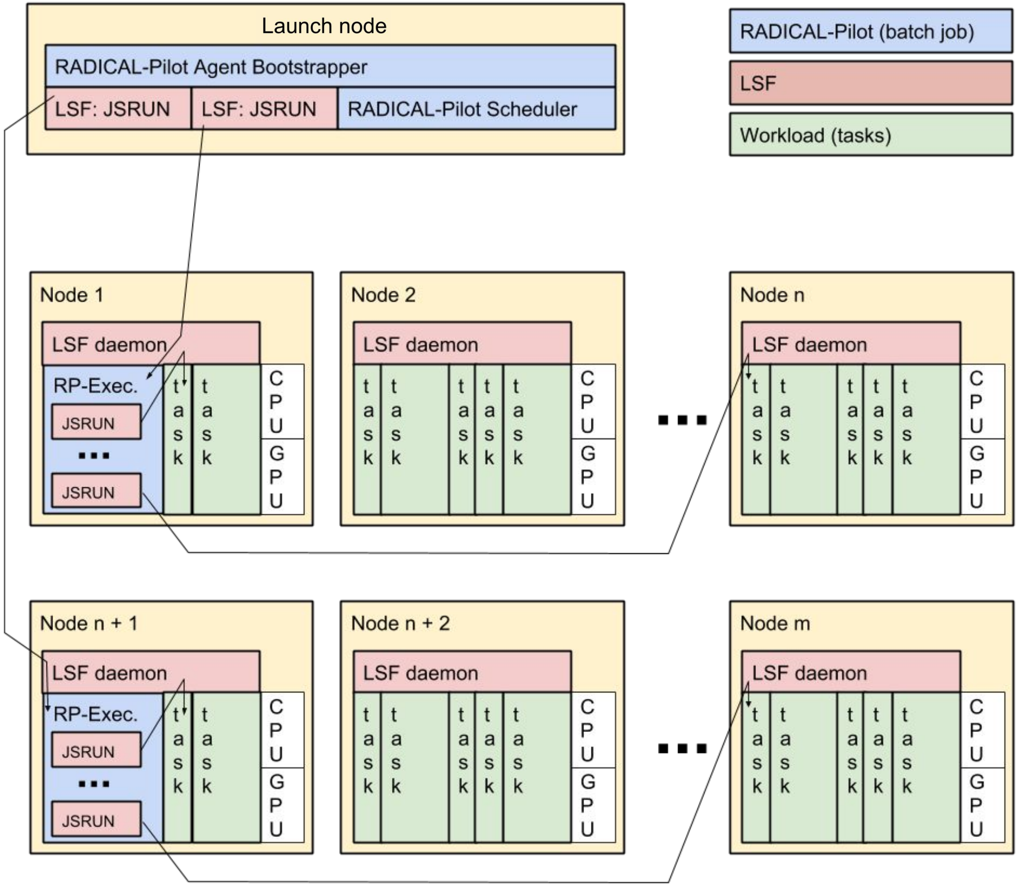
RS: https://github.com/radical-cybertools/saga-python

Status of Scheduling/Launching Jobs on Summit



- LSF and JSRUN used to place and launch tasks on Summit nodes
- Reliability to be tested with many concurrent instances
- Scalability bounded by MOM node resources

Scheduling and Launching Jobs on Summit



- Scalability issues addressed by executing RP Executor on work nodes
- Each RP Executor can serve multiple work nodes, up to external scalability boundaries
- MOM node:
  - 1 JSRUN command launches one RP Executor on a work node
- Work node:
  - 1 JSRUN command launches 1 task on the same or another work node.

JSRUN Resource Spec and Placement

- Summit: 44x4 ‘cores’, 6 GPU, 512GB+96GB ram/HBM2 per ‘node’
- JSRUN Explicit Resource File (ERF) examples:

  -
    ```
    task0: my_app
    rank : 0-3: { host: 1; cpu: {0-3},{4-7},{8-11},{12-15}} : task0
    ```

  -
    ```
    task1: mygpu_app
    rank : 0-5: { host: 1; cpu: {16},{20},{24},{28},{32},{36};
                           gpu: {0,1,2,3,4,5}} : task1
    ```

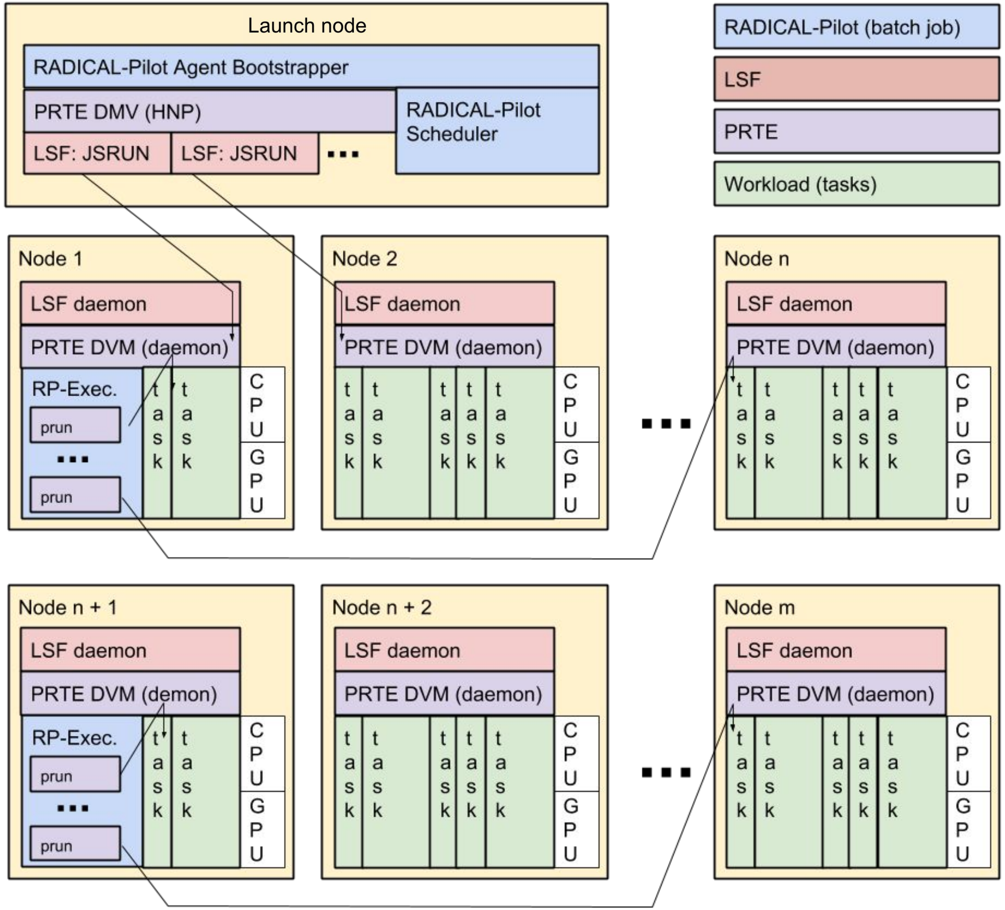
  -
    ```
    task2: mympi_gpu_app
    rank : 0-1: { host: 2; cpu: {0-15},{16-31};
                           gpu: {0,1}} : task2
    rank : 2-4: { host: 3; cpu: {32-47},{48-63},{64-79}} : task2
    ```

- Task 0: 4 processes, each with 4 threads, run on node 1
- Task 1: 6 processes, each with 1 GPU running, run on node 1
- Task 2: 6 processes, each with 16 threads, 2 with 1 GPU, run on nodes 2,3

RADICAL-Pilot and JSRUN

- Who supports it: only LSF batch system vendor
- How we use it: running many concurrent instances in 1 or 2 layered arch.
- How do we manage heterogeneity: ERF per task for placement control
- pros & cons
  - pros: system tool, supported, fine control over placement, heterogeneous tasks
  - cons: buggy, slow, LSF only

Scheduling and Launching Jobs on Summit



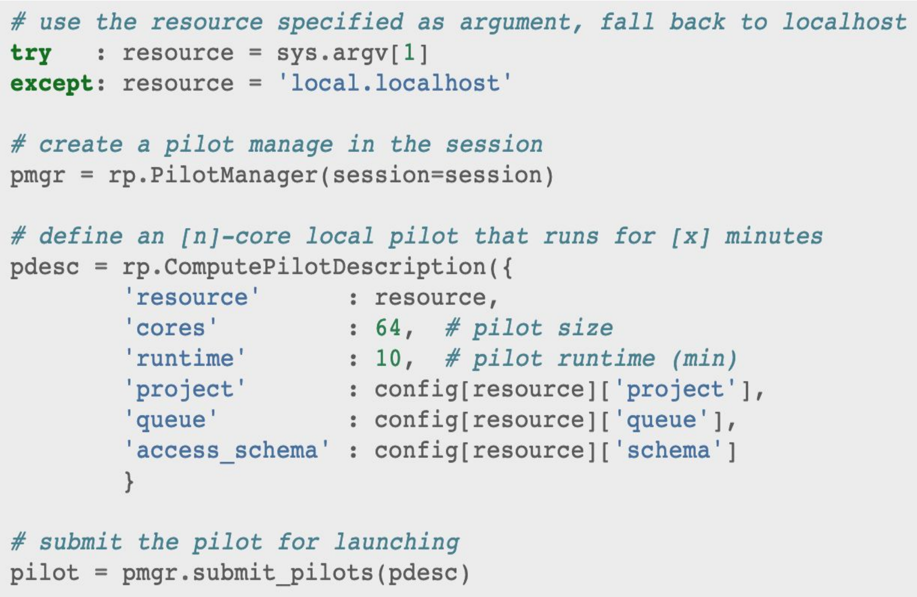
- PRRTE project abstracts away LSF and JSRUN
- PRTE DVM
  - Head Node Process
  - Daemons (1 per node)
- PRUN: uses PRTE DVM to place and execute tasks on nodes
- PRTE daemons use PMI-X as interface to LSF (or SLURM, etc.)

PMI-X and PRRTE

- **PMI-X : P**rocess **M**anagement **I**nterface for E**X**ascale
  https://github.com/pmix/pmix/wiki
- **PRRTE : P**MI-X **R**eference **R**un**T**ime **E**nvironment
  https://github.com/pmix/prrte

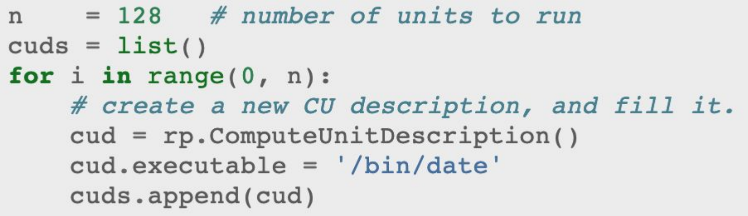
- Who supports it: MPI implementations, batch system vendors
- How we use it: private DVM, concurrent tasks
- How do we manage heterogeneity: CL parameters for placement
- pros & cons:
  - pros: heterogeneous tasks (as with JSRUN), (potentially) fast, **portable**
  - cons: young code, no official support

RADICAL-Pilot API

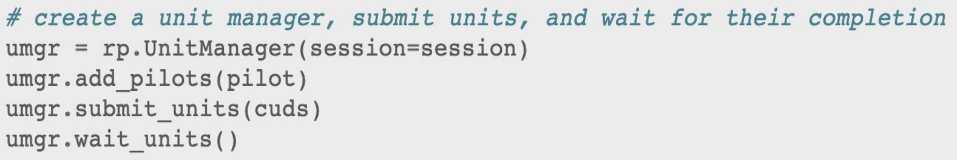


```
# use the resource specified as argument, fall back to localhost
try   : resource = sys.argv[1]
except: resource = 'local.localhost'

# create a pilot manage in the session
pmgr = rp.PilotManager(session=session)

# define an [n]-core local pilot that runs for [x] minutes
pdesc = rp.ComputePilotDescription({
        'resource'      : resource,
        'cores'         : 64,  # pilot size
        'runtime'       : 10,  # pilot runtime (min)
        'project'       : config[resource]['project'],
        'queue'         : config[resource]['queue'],
        'access_schema' : config[resource]['schema']
        }

# submit the pilot for launching
pilot = pmgr.submit_pilots(pdesc)
```

- rp.PilotManager()
- rp.UnitManager()
- rp.ComputePilotDescription()
- rp.ComputeUnitDescription()

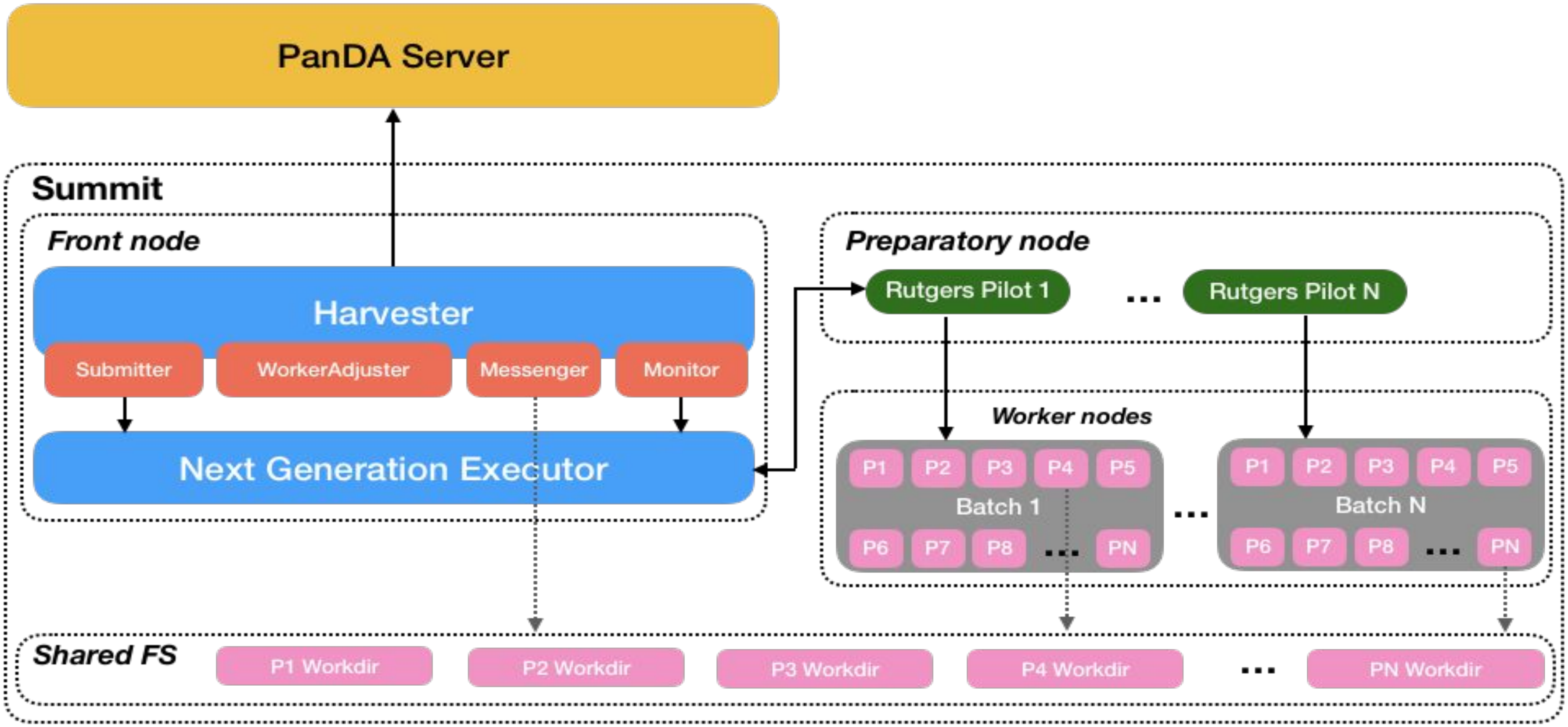


```
n    = 128   # number of units to run
cuds = list()
for i in range(0, n):
    # create a new CU description, and fill it.
    cud = rp.ComputeUnitDescription()
    cud.executable = '/bin/date'
    cuds.append(cud)
```



```
# create a unit manager, submit units, and wait for their completion
umgr = rp.UnitManager(session=session)
umgr.add_pilots(pilot)
umgr.submit_units(cuds)
umgr.wait_units()
```

RADICAL-Pilot, NGE and Harvester



NGE/Harvester integration status and roadmap

- Status:
  - NGE and RADICAL-Pilot can support execution of workloads on Summit
  - Interface between Harvester and NGE tested for bulk tasks (jobs in Harvester lingo) submissions and monitoring
  - Corresponding Submitter, Monitor and Messenger modules for Harvester have been implemented
  - Concurrent CPU/GPU payloads were executed
- Roadmap:
  - Continuing test executions of ATLAS payload via NGE on Titan and comparing results with executions on Summit
  - Test execution of ATLAS CPU/GPU payload via NGE on Summit
  - Evaluate reliability and performance on Summit
  - Estimated effort: 1 Harvester, 1 NGE devs, 3-6 months depending on time allocation
  - Follow up: whole chain ATLAS payload execution.

Thank you

Code available at:

- NGE: https://github.com/radical-cybertools/radical.nge
- RP: https://github.com/radical-cybertools/radical.pilot
- RS: https://github.com/radical-cybertools/saga-python

RADICAL:

- Projects: http://radical.rutgers.edu/projects/
- Publications: http://radical.rutgers.edu/publications/

RADICAL-Pilot Architectural Components

- Client:
  - PilotManager: enables submitting a batch job to the indicated machine, requesting a number of cores/GPUs for a defined walltime.
  - UnitManager: enables scheduling a set of compute units on a pilot for execution, including staging in of the input files required by each unit.
  - StagerOutput: enables staging out of unit output files.
- Agent:
  - StagerInput: retrieves staging in files from UnitManager and makes them available to the units
  - Scheduler: schedules units on available executor(s). Different scheduling algorithms available
  - Executor: place and execute units on the required resources (fraction of worknode, full worknode, multiple worknodes, CPU, GPU or CPU+GPU)
  - StagerOutput: sends unit output files to the client