radical

RADICAL-Pilot and Parsl (RPEX)

Executing Heterogeneous Workflows on HPC Platforms

Aymen Alsaadi

The RADICAL Group
http://radical.rutgers.edu
Rutgers, The State University of New Jersey

Parsl Fest 2022

Motivation: Why the Integration approach in general?

- Possibility of achieving better capabilities by merging both system capabilities.
- New capabilities serve new use cases or existing use cases in a better way.
- Only develop new capabilities when they are not available in both systems.
- Integrating two independent systems → we are preventing effort duplications → better collaboration.
- Less code.
- RADICAL-Pilot (RP) is a workload runtime system based on the building block approach.
- By design, both Parsl and RP allow the ease of integration with other systems.

RADICAL-Pilot (RP)

radical

- A scalable, modular, and interoperable pilot system that enables the asynchronous execution of heterogeneous workloads on heterogeneous HPC resources.
- RP provides methods for efficiently and effectively scheduling, placing, and launching independent tasks across multiple nodes.
- RP supports the concurrent execution of up to 10^6 tasks on 10^3 compute nodes with low overheads.

Parsl and RADICAL-Pilot

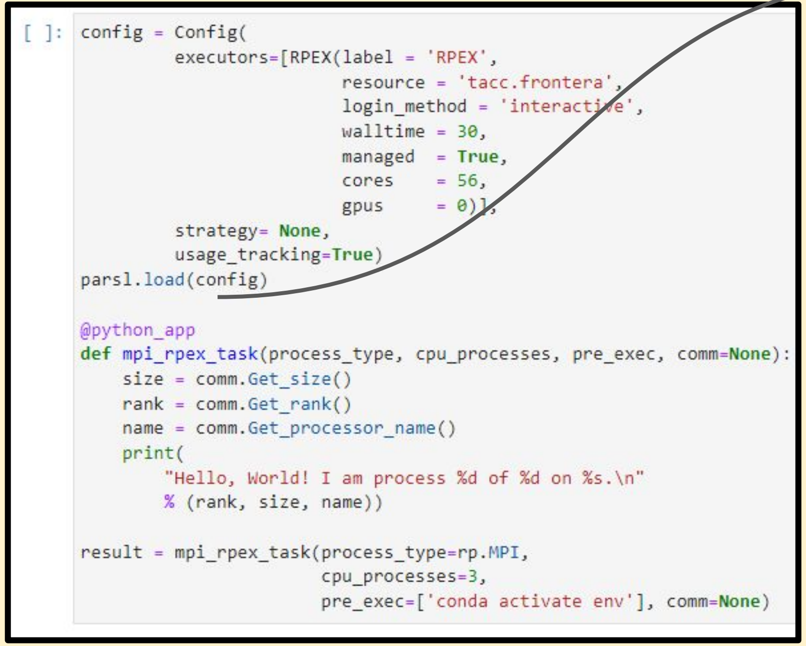
What Parsl offers to RADICAL-Pilot:

- Workflows management capabilities.
- Construct and build complex dynamic workflows
- Flexible and lightweight API and programming model.

What RADICAL-Pilot offers to Parsl:

- Single/multi cores/GPUs with MPI, OpenMP, and single/multi-[threaded|process] tasks.
- Heterogeneous resources management (CPUs/GPUs)
- Granular resource specification at the pilot/task level (processes/ranks/threads).

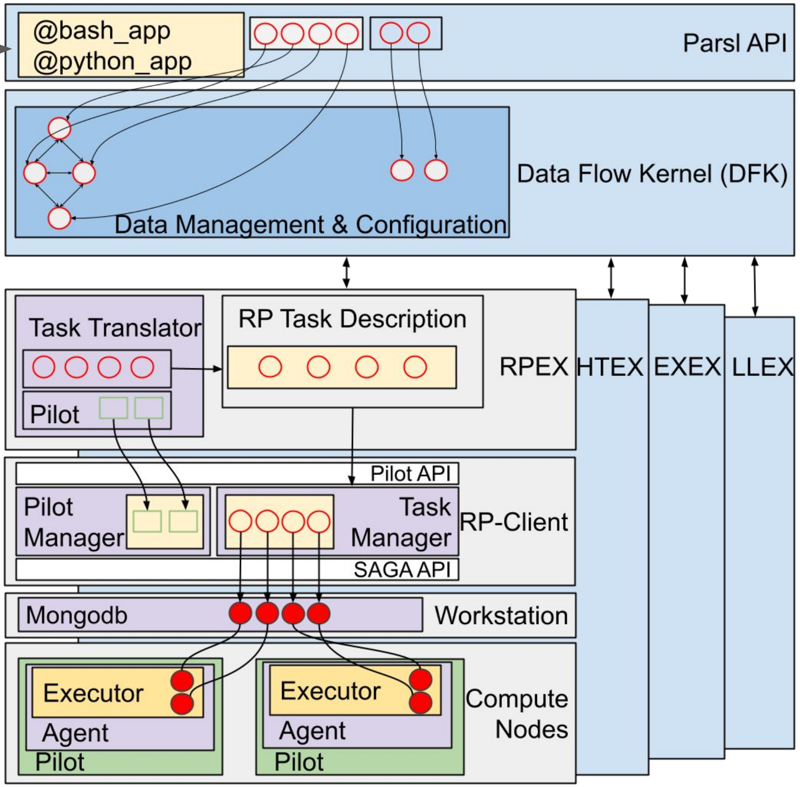
RP Executor (RPEX) Architecture



```
[ ]: config = Config(
              executors=[RPEX(label = 'RPEX',
                              resource = 'tacc.frontera',
                              login_method = 'interactive',
                              walltime = 30,
                              managed  = True,
                              cores    = 56,
                              gpus     = 0)],
              strategy= None,
              usage_tracking=True)
     parsl.load(config)

     @python_app
     def mpi_rpex_task(process_type, cpu_processes, pre_exec, comm=None):
         size = comm.Get_size()
         rank = comm.Get_rank()
         name = comm.Get_processor_name()
         print(
             "Hello, World! I am process %d of %d on %s.\n"
             % (rank, size, name))

     result = mpi_rpex_task(process_type=rp.MPI,
                            cpu_processes=3,
                            pre_exec=['conda activate env'], comm=None)
```



Supported Use Cases

Colmena:

- A Python package for machine learning based steering of ensemble computations on HPC platforms.
- Requires RPEX capabilities to launch large MPI workflows (executables and functions) via Parsl programming model.

Ice Wedge Polygons:

- Detect and classify wedge-shaped ice masses create polygonized land surface patterns called Ice Wedge Polygons (IWP) across large Arctic areas.
- Requires RPEX capabilities to launch MPI CPU and GPU tasks across multiple nodes.

Performance results on TACC Frontera

Colmena:

- A single node MPI executables and non-MPI python functions workload.
- Executing 3600 tasks on 256 nodes.
- A resource utilization of ~98% on 256 nodes (~14K CPU cores)
- RPEX OVH of ~500s (RP overheads + Parsl overheads)
- Throughput of ~15 tasks/sec.

Ice Wedge Polygons:

- A multi-node MPI functions workload.
- Executing 32 tasks on 16 nodes.
- A resource utilization of ~99% (~900 cores and 64 GPUs)
- RPEX OVH of 6.5s.
- Throughput of ~1 tasks/sec.

Future work and new use cases

- Develop a bulk mode submission for RPEX to boost the throughput of:
  - Task Submission
  - Task Scheduling
  - Task Execution

- Cylon is a new upcoming use case that requires RPEX capabilities.
  - Cylon is a python and C++ parallel data framework that provides fast and scalable data aggregation while offering the possibility to integrate with other data distribution frameworks for additional capabilities.