LA-UR-23-29120

Approved for public release; distribution is unlimited.

Title:	Elastic Workflows with PMIx
Author(s):	Pritchard, Howard Porter Jr. Bhattarai, Rajat
Intended for:	HPC Intern summer showcase 2023
Issued:	2023-08-08

Los Alamos
NATIONAL LABORATORY

Los Alamos
NATIONAL LABORATORY

Elastic Workflows with PMIx

Rajat Bhattarai
Howard Pritchard (Mentor)

08/10/2023

Problem Statement

- Scientific workflows increasing in complexity
- Elastic workflows promise improved application and system performance
- Parsl[1] and Dask[2] support elasticity but at job level
  - Cloud-like elasticity model
  - Resources as block, each job allocation corresponds to a block
  - Resource change means adding and canceling jobs
  - Problems:
    - Scheduling logic of resource managers in DOE systems might affect, users may be allowed to have limited number of jobs running at a time
    - Spanning same HPC applications particularly MPI applications across multiple jobs for expansion gets complicated
    - Need for common standard for resource management for workflows

Proposed Solution

- Grow within the same job allocation
- Use advanced middleware like PMIx[3]
- Advantages of this approach:
  - Fine grained resource management and flexible resource allocations from resources level
  - Standardize how workflow manager should talk to resource managers
  - Extra capabilities like launching processes, event notification, job control and monitoring, queries and logging

PMIx

- Works as messenger between applications and resource manager in client server model
- Allows process management, **flexible resource management**, event notification, job control and monitoring, queries and logging
- Work on *PMIx_Allocation_request* API ongoing to enable application directed resource changes (EU Deep SEA project)
- PMIx Reference Runtime Environment(PRRTE)[4] has another resource changing capabilities: DVM(Distributed Virtual Machine) extension and pruning while launching applications (Used in this work)

Parsl

- DataFlowKernel
  - Orchestrator of tasks
- Executor
  - Handles execution of tasks on remote nodes e.g. HighThroughputExecutor
- Provider
  - Provides resources e.g. SlurmProvider
- App decorators
  - @bash_app and @python_app
- Future Object

Listing 1: Running MPI apps in Parsl example

```
config = Config(
    executors=[
        HighThroughputExecutor(
            ...........
            max_workers=4,
            provider=SlurmProvider(
                ...........
                nodes_per_block=4,
                walltime='00:20:00',
                ...........
            ),
        )
    ],
)
parsl.load(config)
@bash_app
def echo_hello(n: int, stderr='std.err', stdout='std.out'):
    return f'mpiexec -n {n} --ppn 1 hostname'
```

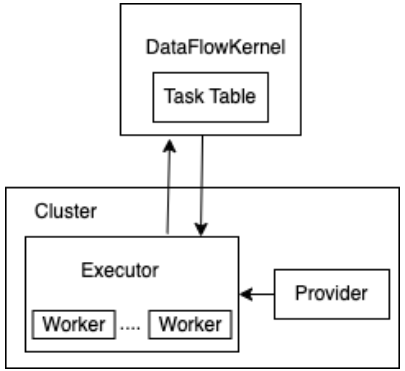


Fig. 1: Parsl Example and architecture

Parsl with PMIx

- SlurmPMIxProvider
- Modification to HighThroughputExecutor
  - Starting and stoping PRRTE DVM
  - Adding more Parsl workers for expansion
- Modification to Bash App decorator
  - Modifying app launch command options in prun
  - Adding --dvm-uri, --hostfile and --add-hostfile option

Listing 2: Running Pmix enabled Parsl implementation

```
config = Config(
    executors=[
        HighThroughputExecutor(
            ...........
            max_workers=3,
            provider=SlurmPMIxProvider(
                ...........
                nodes_per_block=3,
                walltime='00:20:00',
                ...........
            ),
        )
    ],
)
parsl.load(config)
@bash_app
def echo_hello(n: int, stderr='std.err', stdout='std.out'):
    return f'prun -n {n} hostname'
```

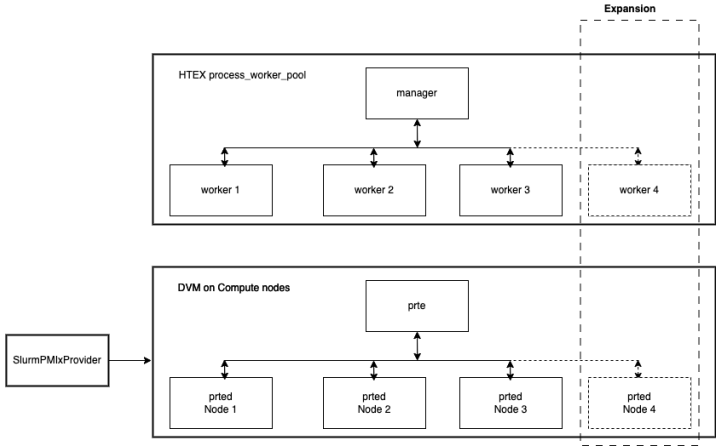


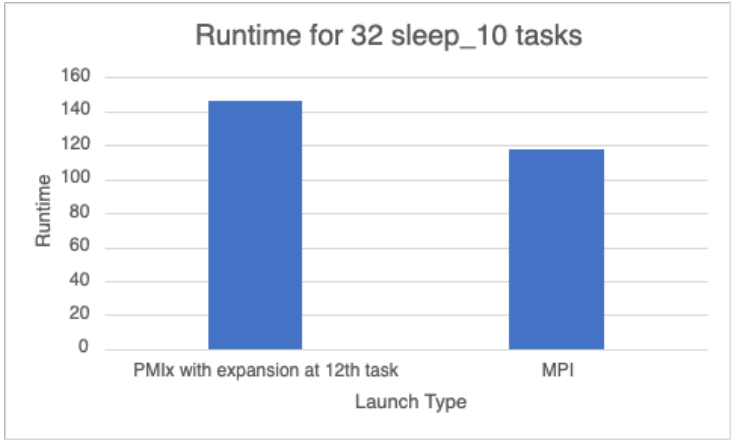
Fig. 2: Parsl with PMIx example and atchitecture

Challenges

- Parsl relies on PRRTE to spread jobs across nodes which does not automatically ensure all nodes are being used
  - Use hostfiles
- Once tasks are serialized using pickle, the task's launch command cannot be modified
  - Modify before serializing inside bash app decorator
- Task with expand command when run in parallel with normal tasks causes race condition(PRRTE bug)
  - Run expanding task (*prun --add-localhost*) alone: adds time overhead
- After DVM extension, PRRTE shows non-deterministic spawn behaviour (PRRTE bug)

Evaluation

- Created two dummy workflows using two dummy apps: hostname and sleep_10
  - Launched with MPI launcher with 3 nodes
  - Launched with PMIx launcher with expansion feature from 3 to 4 nodes at certain task iteration
  - Ran hostname for 256 times
  - Ran sleep_10 for 32 times
- Working on a real computational chemistry workflow using NWChem



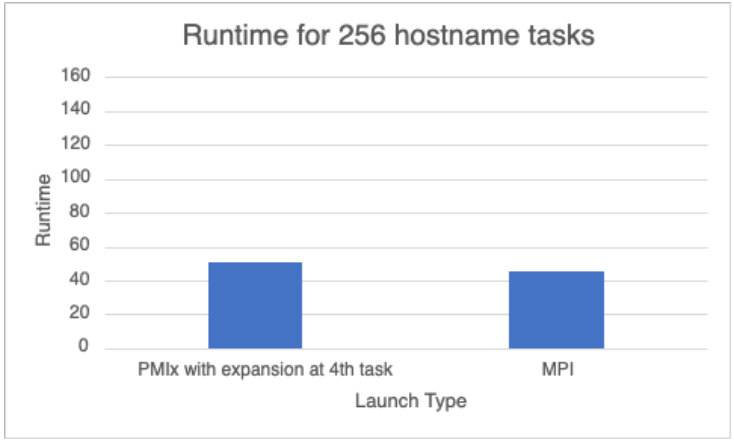


Fig. 3: Some Measurement results

Conclusion

- Created an elastic Parsl implementation with PMIx
- Suitable for workflows with large number tasks than Parsl workers
  - Parsl suggests configuring the executor to launch a number of workers equal to the number of MPI tasks per block [5]
- Lot of time overhead which can be avoided
- **Future Work**
  - Fix existing PRRTE issues
  - Add resource shrinkage support
  - Leverage work done by the EU Deep SEA project for dynamic allocation API
  - Utilize other PMIx capabilities to create PMIx as a standard for exchanging information between Workflow Manager, Resource Manager and Applications

References

1. Babuji, Y., Woodard, A., Li, Z., Katz, D. S., Clifford, B., Kumar, R., ... & Chard, K. (2019, June). Parsl: Pervasive parallel programming in python. In Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing (pp. 25-36).
2. Rocklin, M. (2015, July). Dask: Parallel computation with blocked algorithms and task scheduling. In Proceedings of the 14th python in science conference (Vol. 130, p. 136). Austin, TX: SciPy.
3. Castain, R. H., Solt, D., Hursey, J., & Bouteiller, A. (2017, September). Pmix: process management for exascale environments. In Proceedings of the 24th European MPI Users' Group Meeting (pp. 1-10).
4. https://docs.prrte.org/en/latest/#
5. https://parsl.readthedocs.io/en/stable/userguide/mpi_apps.html

Los Alamos
NATIONAL LABORATORY

Thank You.

Questions?