Characterizing the Performance of Executing Many-tasks on Summit

Matteo Turilli, Andre Merzky, Thomas Naughton, Wael Elwasif, Shantenu Jha

RUTGERS

OAK RIDGE National Laboratory

BROOKHAVEN NATIONAL LABORATORY

SC19 - November 22, 2019, Colorado Convention Center, Denver

Outline

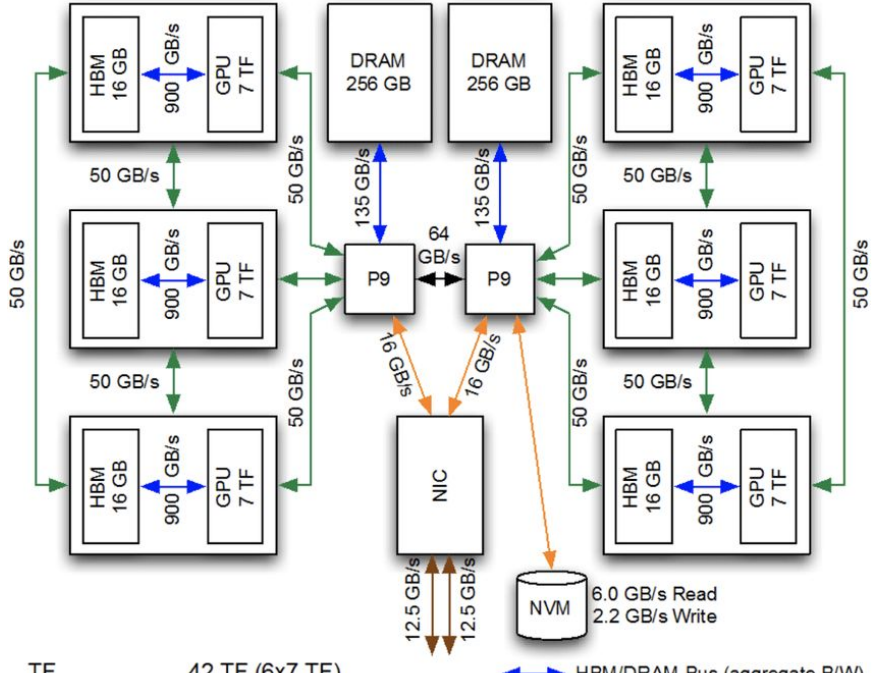
- Many-tasks applications on high performance computing platforms
- Summit architecture
- Implementing the pilot abstraction on Summit
- Performance comparison between JSM and PRRTE-based implementations
- Overheads decomposition and performance improvement

Many-tasks Applications on HPC

- Workloads of many scientific applications are comprised of many tasks: molecular dynamics, seismology, climate sciences, particle physics, etc.
- Tasks are independent, self contained computing unit, i.e., executables
- Tasks can be heterogeneous: type/amount of resources and execution time
- Requires HPC platforms for both scale and performance
- Adoption of HPC platforms presents many challenges:
  - Departing from traditional paradigms: single, large MPI executable
  - Acquisition and dynamic management of heterogeneous resources
  - Coping with resource policies designed for traditional HPC use cases

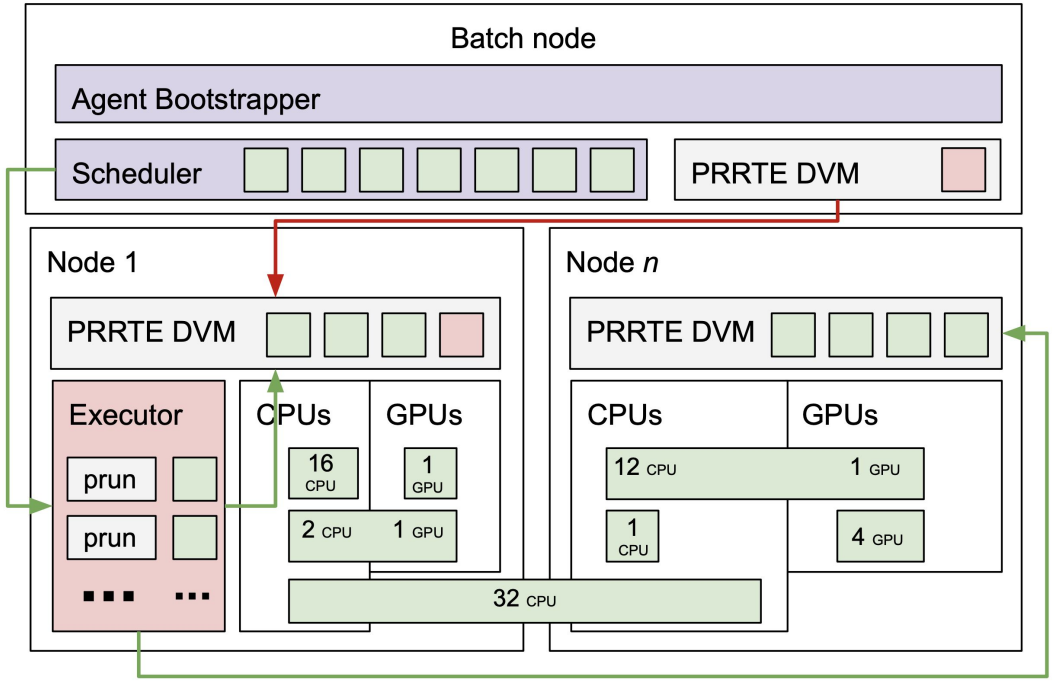
Summit at Oak Ridge National Lab

- 4,600 compute nodes IBM Power System AC922
- Each node:
  - 2 IBM POWER9 22C 3.07GHz, each supporting 4 HW threads
  - 6 NVIDIA Volta GV100
  - 512 GB of DDR4 for POWER9 and 96 GB of High Bandwidth Memory (HBM2) for GPU



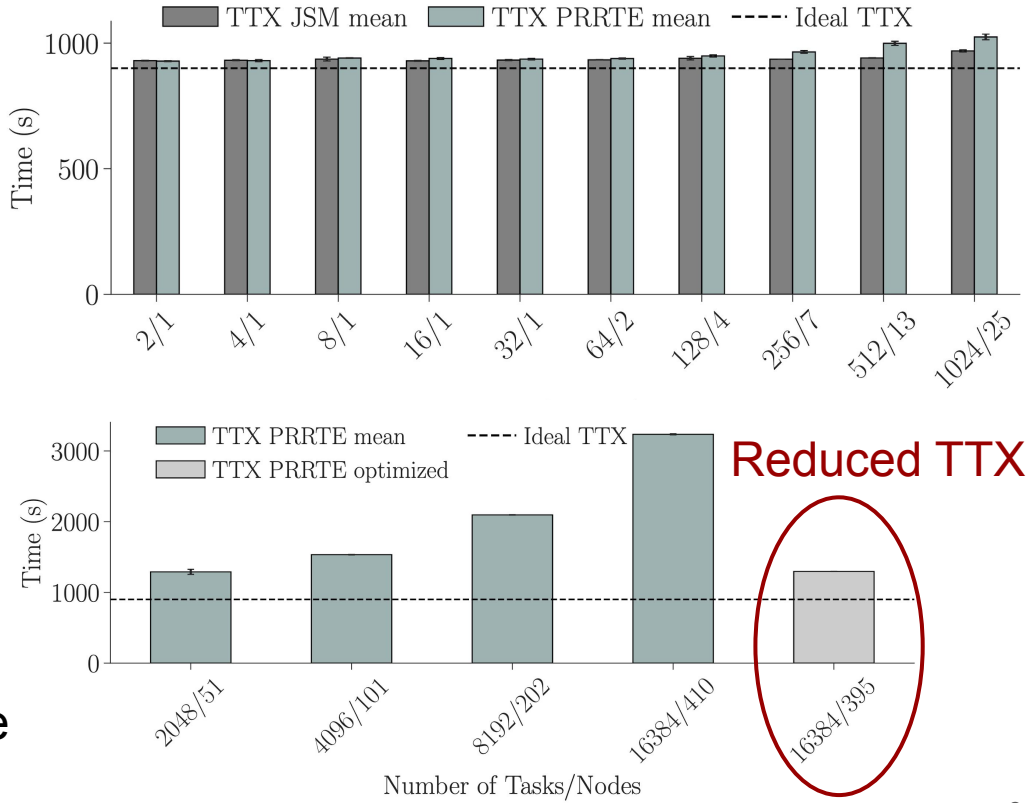
Implementing the Pilot Abstraction on Summit

- Pilot: scheduling, placing and launching tasks on job resources
- BM Job Step Manager (JSM)
- Process Management Interface for Exascale (PMIx)
- PMIx Reference RunTime Environment (PRRTE)
- RADICAL-Pilot (RP)
- Integration of: RP+PRRTE and RP+JSM



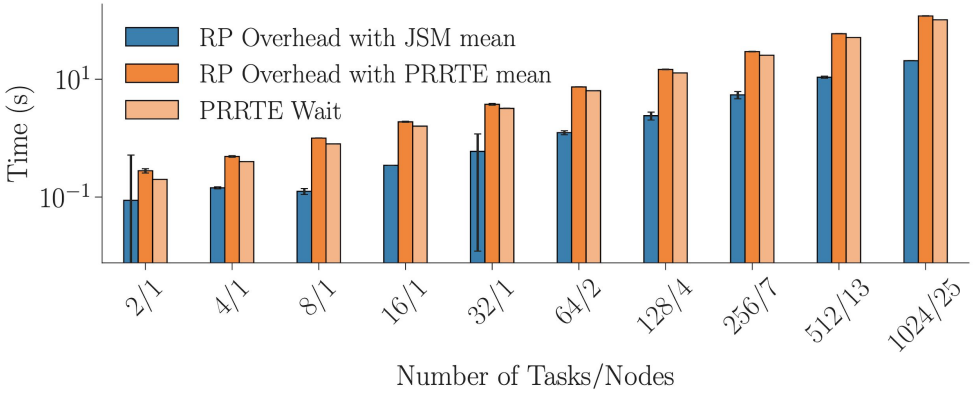
Performance comparison: Total Execution Time (TTX)

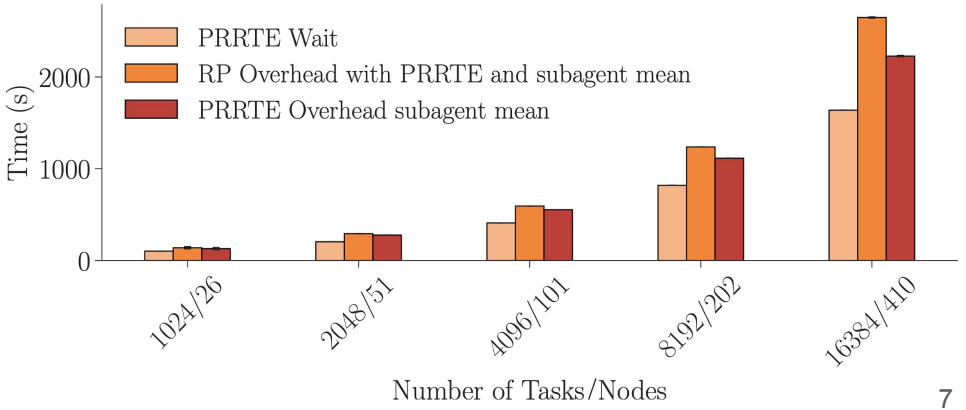
- Global performance view of TTX with RP+JSM and RP+PRRTE
- Many-tasks application:
  - 2-16384, 1 core, 15m, tasks
  - 1-410 Summit work nodes
  - Worse case scenario
- Scaling of JSM: 1024 tasks
- Scaling of PRRTE: ~20K tasks
- RP+PRRTE TTX slightly higher than PR+JSM TTX
- RP+PRRTE TTX exponential due to RP implementation (reduced)



Performance comparison: TTX Decomposition

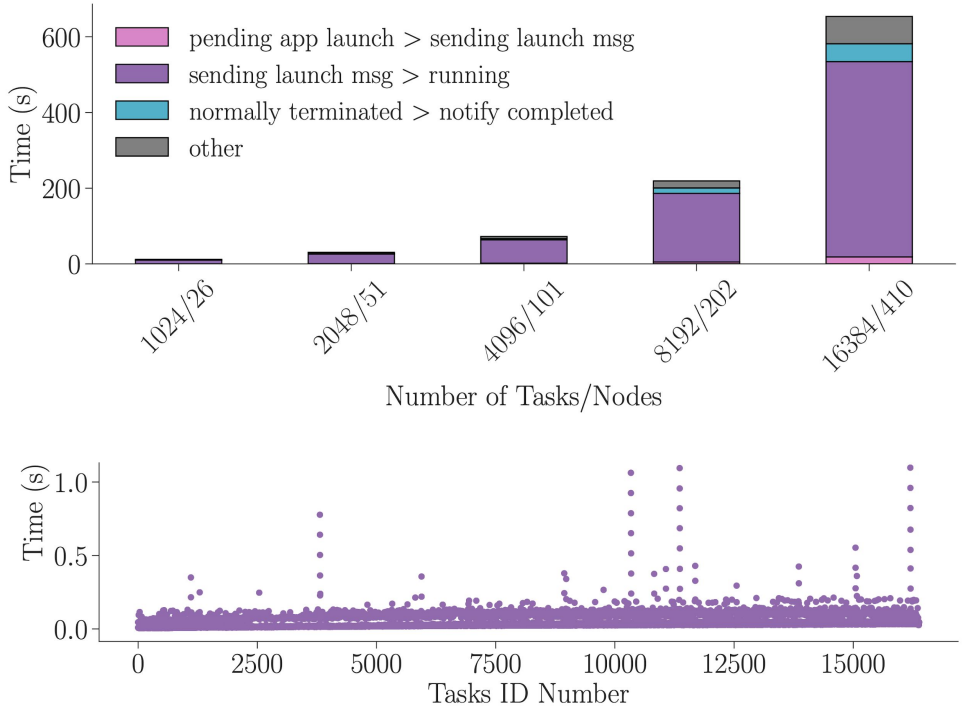
- Overhead view of TTX with RP+JSM and RP+PRRTE
- RP+JSM overhead:
  - Higher variance than RP+PRRTE
  - Lower mean than RP+PRRTE
- RP+PRRTE overhead:
  - Almost no variance
  - Scales ~linearly
  - Mostly due to RP wait time
- Without RP wait time, PRRTE overhead is negligible compared to other overheads.



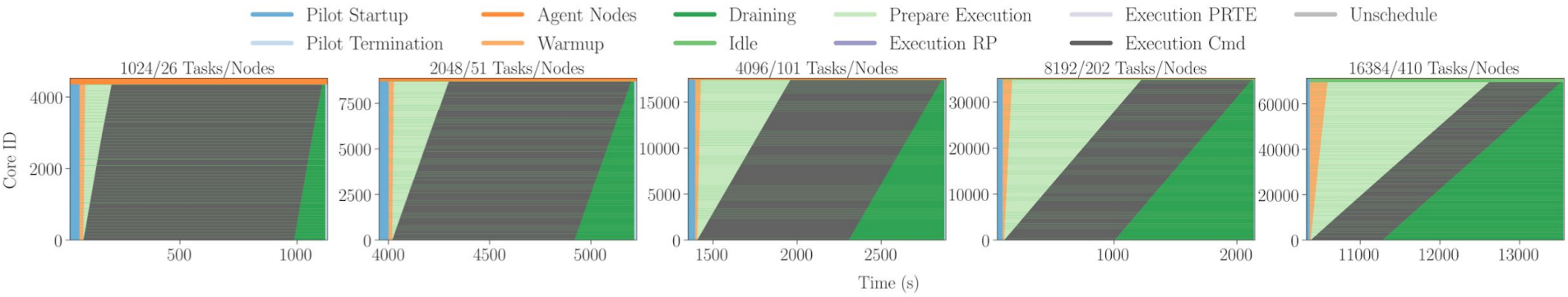


Decomposition of PRRTE Execution Time

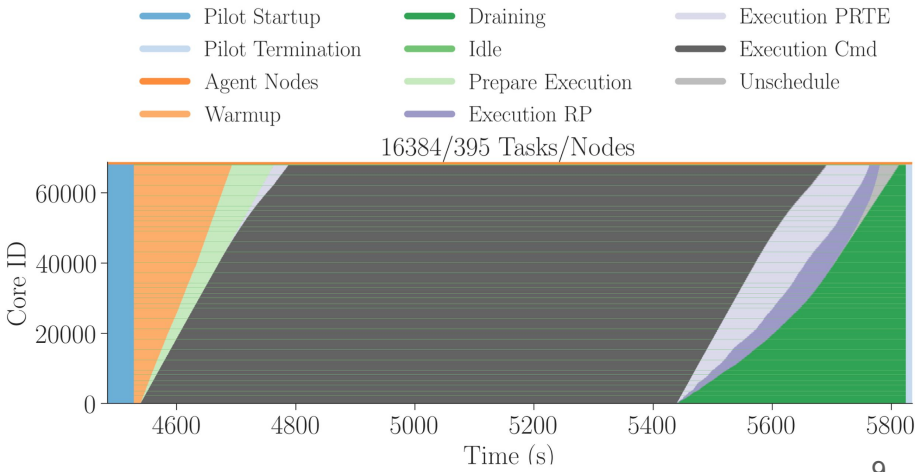
- Implemented tracing of PRRTE task states at runtime
- Scheduling task across DVM becomes dominant at scale
- Average scheduling time of a task across the DVM is 0.2s
- Sequential scheduling times aggregate at runtime
- Opportunity for concurrent task scheduling across DVMs



RP+PRRTE Performance: Resource Utilization (RU)



- RU inversely proportional to TTX
- RU progressively dominated by preparing execution, i.e., RP wait time
- Removing wait time improves RU from 25.6% to 63.5% for 16k tasks
- RP scheduling is the new dominant overhead: addressed in latest release.



Conclusions

- Baseline characterization of the performance of RP, JSM and PRRTE integration on Summit when executing many-task workloads.
- Results:
  - Better scalability of PRRTE for > 987 concurrent task executions
  - PRRTE overheads are negligible when compared to aggregated overheads
  - PRRTE open source code enables reduction of aggregated overheads
- Planned work:
  - Support of several INCITE and Exascale compute projects
  - Partitioning of RP on Summit to support scaling up to whole Summit
  - Improvement of RP scheduler to support heterogeneous/dynamic workloads

Tasks / Nodes	Agent Nodes	Pilot Startup	Warmup	Prep. Execution	Exec. RP	Exec. PRRTE	Exec. Cmd	Unschedule	Draining	Pilot Termination	Idle
1024 / 26	3.846%	3.630%	1.680%	4.510%	0.016%	0.002%	**73.999** %	0.001%	6.149%	0.812%	5.355%
2048 / 51	1.961%	3.622%	1.603%	9.800%	0.011%	0.004%	**65.313** %	0.000%	11.356%	0.867%	5.462%
4096 / 101	0.990%	2.698%	1.398%	16.178%	0.013%	0.002%	**54.797** %	0.000%	17.798%	0.534%	5.593%
8192 / 202	0.495%	2.076%	1.954%	23.375%	0.021%	0.002%	**39.990** %	0.001%	25.570%	0.396%	6.120%
16384 / 410	0.244%	1.271%	3.309%	28.779%	0.021%	0.002%	**25.596** %	0.001%	32.752%	0.256%	7.771%
16384 / 410	1.013%	3.265%	6.314%	2.345%	2.421%	4.988%	**63.557** %	0.286%	11.526%	0.800%	3.485%

Table 1: Experiments 3–4: Resource utilization (RU) expressed as the percentage of resources used of blocked by RP, PRRTE and the workload. Last line shows optimized run of Experiment 4.

Thank you

To the other members of the PMIx community, and **Ralph Castain** in particular, for the excellent work that we build upon.

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. At Rutgers, this work was also supported by NSF “CAREER” ACI-1253644, RADICAL-Cybertools NSF 1440677 and 1931512, and DOE Award DE-SC0016280. We also acknowledge DOE INCITE awards for allocations on Summit.

- Experiments data: https://github.com/radical-experiments/summit_jsrun_prte
- OpenPMIx PRRTE: https://github.com/openpmix/prrte
- RADICAL-Pilot: https://github.com/radical-cybertools/radical.pilot
- RADIAL Cybertools: https://radical-cybertools.github.io/