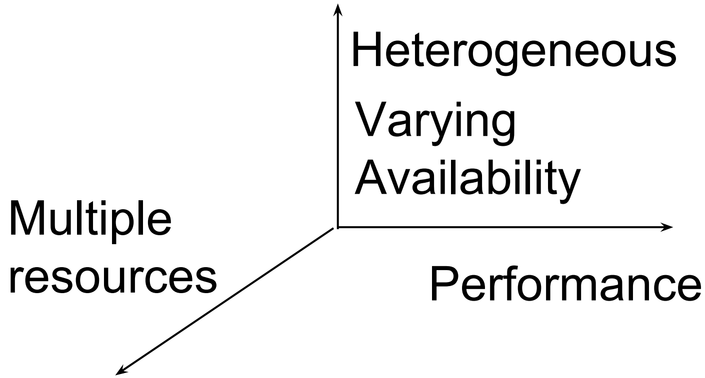
Executing dynamic heterogeneous workloads on Blue Waters with RADICAL-Pilot

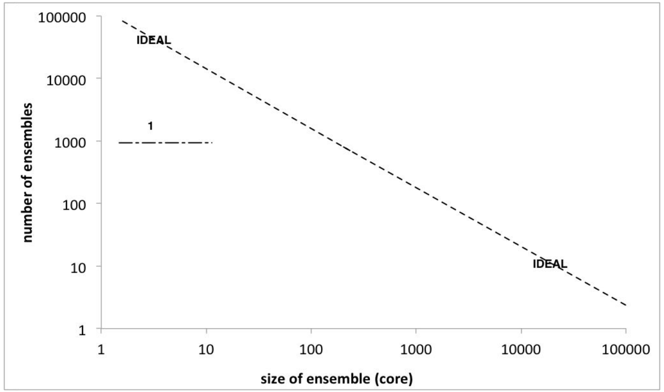
Mark Santcroos*, Ralph Castain†, Andre Merzky*, Iain Bethune‡ and Shantenu Jha*
* *School of Electrical and Computer Engineering, Rutgers University, New Brunswick, New Jersey, USA*
† *Intel Corporation, USA*
‡ *EPCC, The University of Edinburgh, Edinburgh, UK*

Research in **A**dvanced **DI**stributed **C**yberinfrastructure & **A**pplications **L**aboratory (RADICAL)
Rutgers University
http://radical.rutgers.edu
http://radical-cybertools.github.io

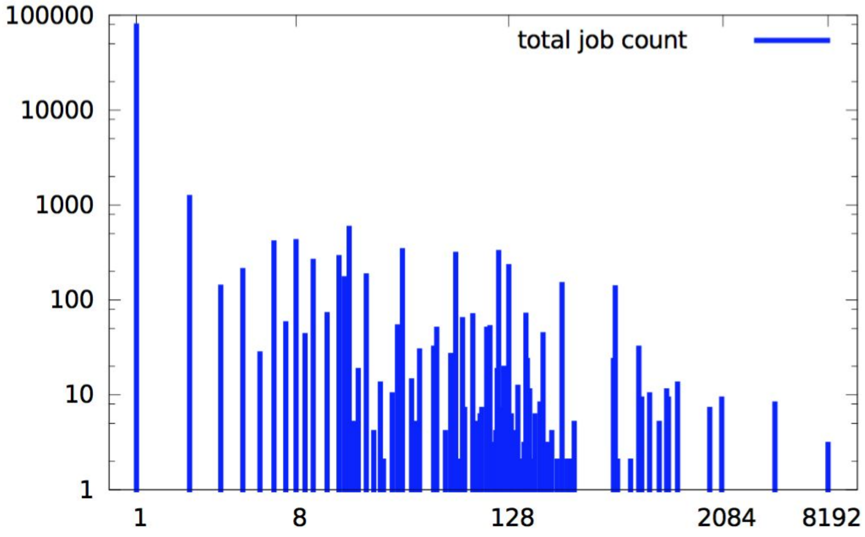
Extreme Scale “Task-level Parallelism” on HPDC

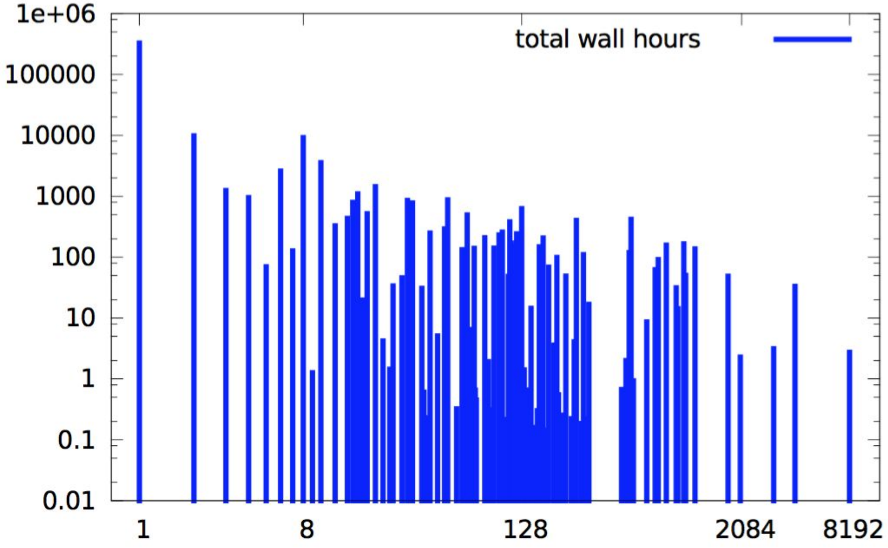
- Problems in computational science *naturally* amenable to “task level” parallelism computing
- Beyond HTC vs HPC
- Given access to X cores/nodes – slice/dice or distribute as needed.
- Resources and workloads are characterised by a range of properties:





Blue Waters Job Size Distribution





Requirements / goals

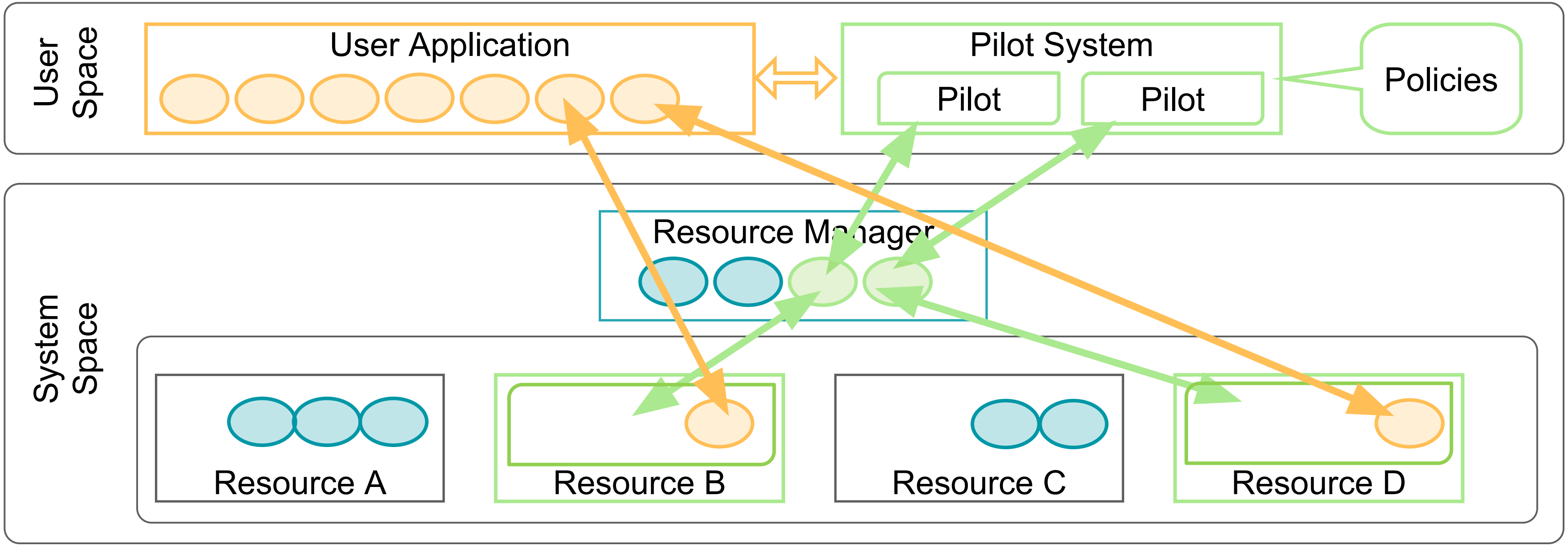
- Workload with heterogeneous tasks
  - Varying core count
  - Varying application
  - MPI / non-MPI
- Dynamic workload with workload unknown in advance
  - Task N+1 depends on task N
- Control over concurrency of tasks
  - Might be loosely coupled (e.g. replica exchange)
- ~10k concurrent tasks

(Why not) batch queue jobs

- Low throughput
  - Every job needs to queue
  - Breaks especially in dynamic workload situations
- No control over concurrency
- Limit on total concurrency
- Maximum of one task per node
- Job arrays are too inflexible (nor available on BW)
- Too many flavours

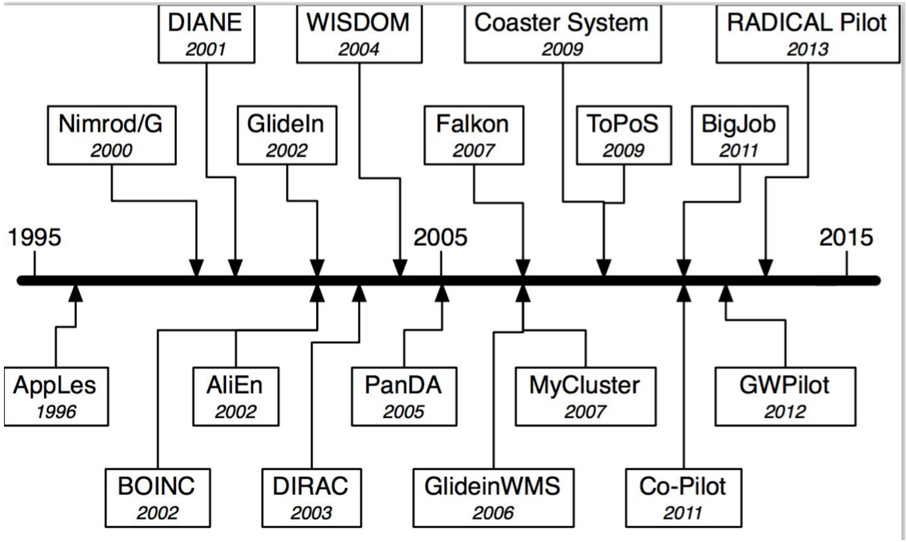
Pilot Abstraction

Working definition: A system that generalizes a placeholder to allow application-level control over acquired resources.



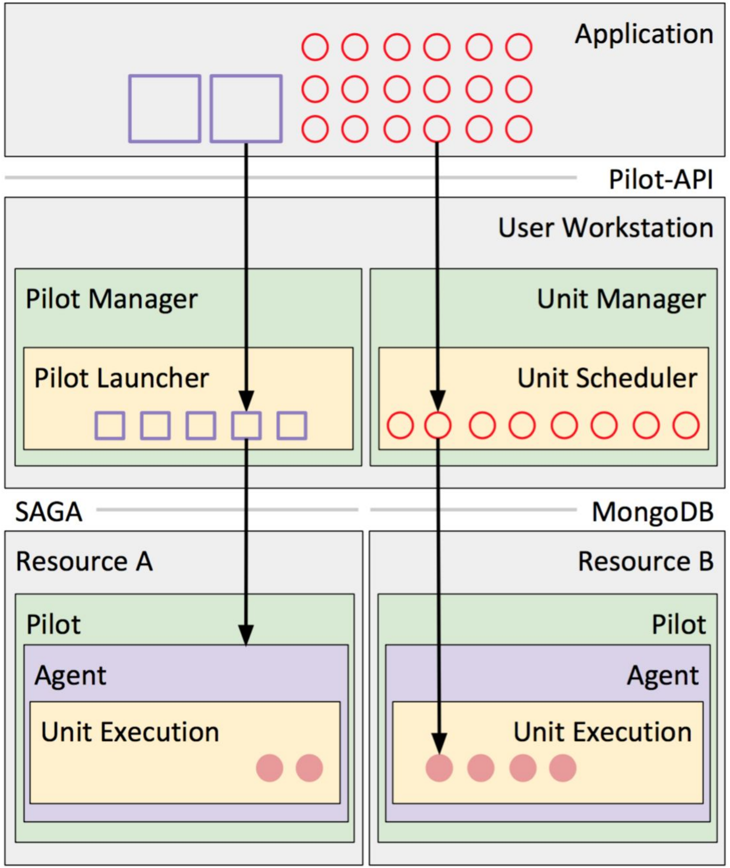
Advantages of Pilot-Abstraction

- Decouples workload management from resource management
- Flexible Resource Management
  – Enables the fine-grained "slicing and dicing" of resources
  – Tighter temporal control and other advantages of application-level Scheduling (avoid limitations of system-level scheduling)
- Build higher-level frameworks without explicit resource management



RADICAL-Pilot Overview

- Programmable interface, arguably unique:
  - Well defined state models for pilots and units.
- Supports research whilst supporting production scalable science:
  - Pluggable components; introspection.
- Portability and Interoperability:
  - Works on Crays, most known clusters, XSEDE resources, OSG, and Amazon EC2.
  - Modular pilot agent for different architectures.
- Scalable:
  - Agent, communication, throughput.



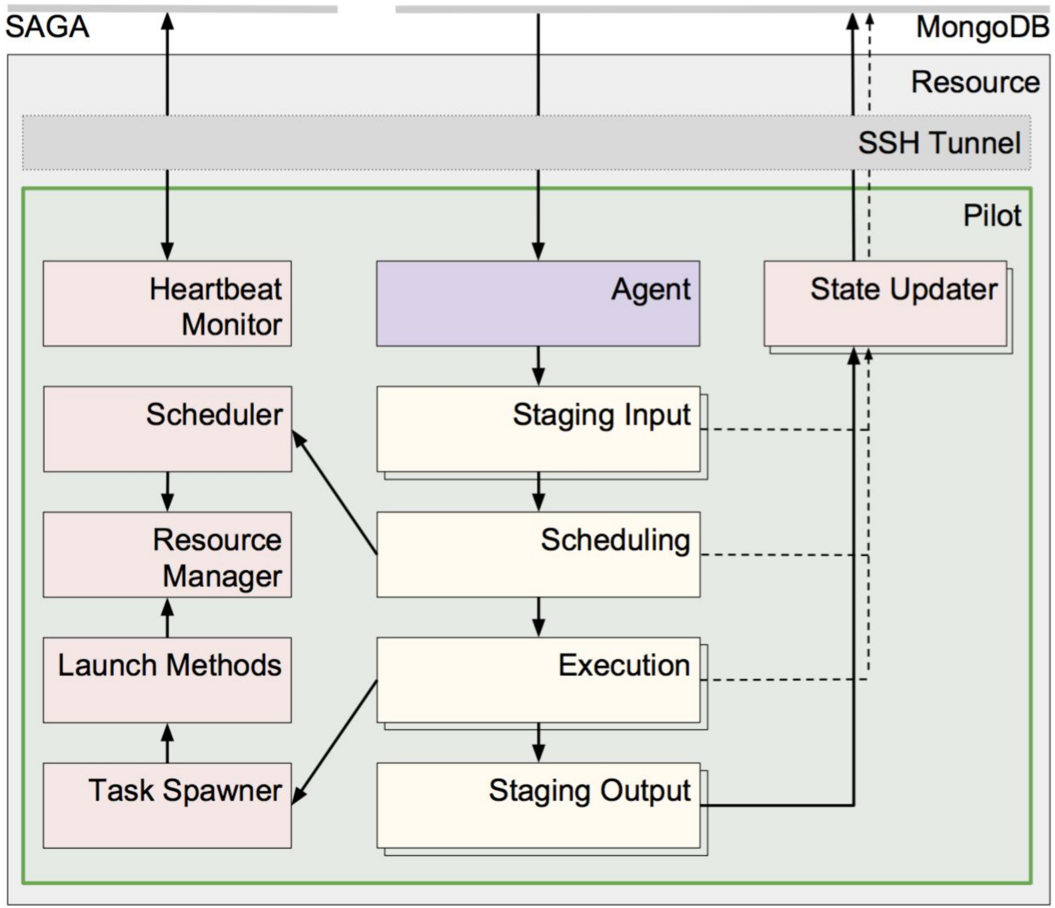
```
# create a pilot manage in the session
pmgr = rp.PilotManager()

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

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

```
n = 42 # Number of units to run
cuds = []
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()
```

Agent Architecture

- **Components:**
  Enact state transitions for Units
- **State Updater:**
  Communicate with client library and DB
- **Scheduler:**
  Maps Units onto compute nodes
- **Resource Manager:**
  Interfaces with batch queuing system, e.g. PBS, SLURM, etc.
- **Launch Methods:**
  Constructs command line, e.g. APRUN, SSH, ORTE, MPIRUN
- **Task Spawner:**
  Executes tasks on compute nodes



(Why not) RADICAL-Pilot + APRUN

- RP Agent runs on MOM node
- Uses aprun to launch tasks onto the worker nodes

- Low throughput (ALPS not designed for short/small tasks)
- Limit on total concurrency (1000 aprun instances)
- Maximum of one task per node

(Why not) RADICAL-Pilot + CCM

- Bootstrapper runs on MOM node
- Bootstrapper creates “cluster”
- Uses ccmrun to launch RP Agent into the “cluster”

- Not universally available

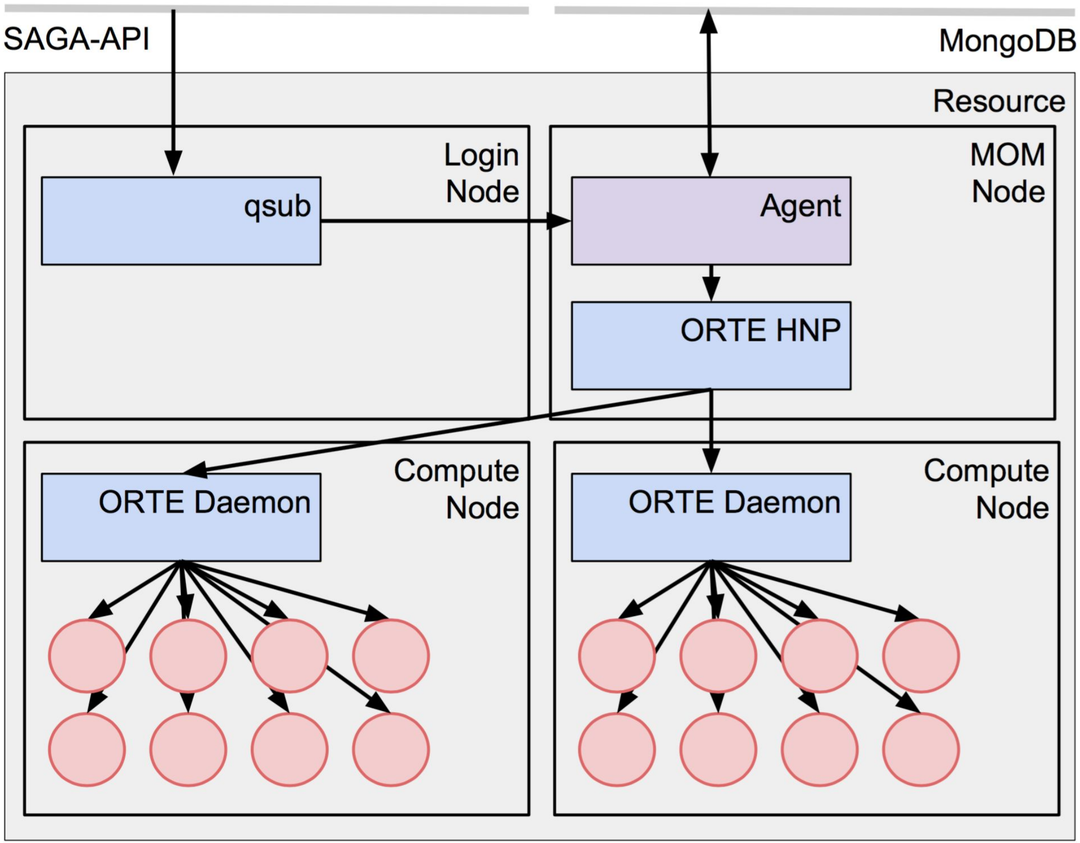
RADICAL-Pilot + ORTE-CLI (a bit better)

- ORTE: **O**pen **R**un**T**ime **E**nvironment
  - Isolated layer used by Open MPI to coordinate task layout
  - Runs a set of daemons over compute nodes
  - No ALPS concurrency limits
  - Supports multiple tasks per node
- orte-submit is CLI which submits tasks to those daemons
  - ‘sub-agent’ on compute node that executes these
  - Limited by fork/exec behavior
  - Limited by open sockets/file descriptors
  - Limited by file system interactions

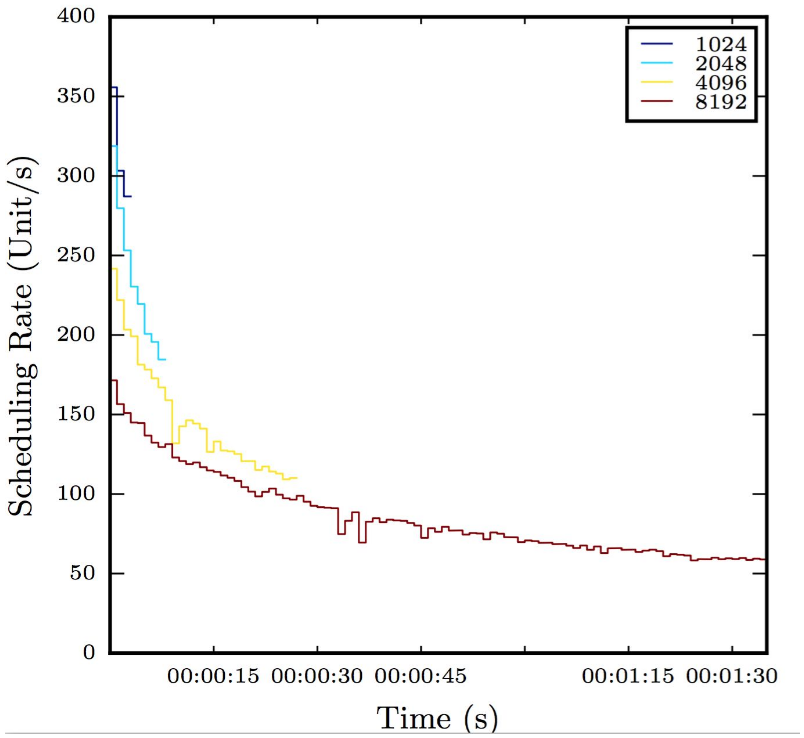
RADICAL-Pilot + ORTE-LIB (much better)

- All the same as ORTE-CLI, but
  - Uses library calls instead of orte-submit processes
  - No central fork/exec limits
  - Shared network socket
  - (Hardly) no central file system interactions

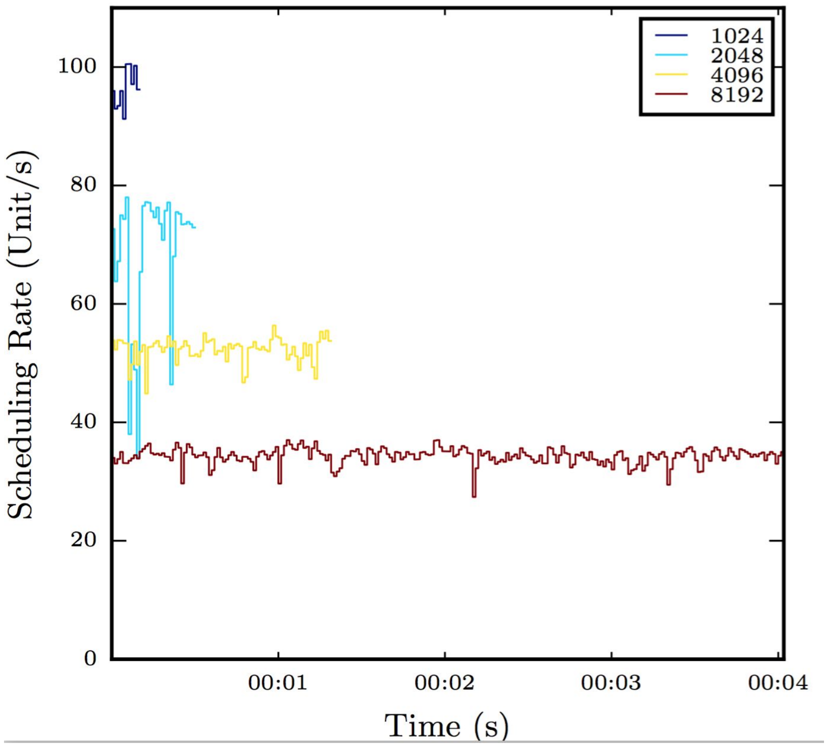
RADICAL-Pilot + ORTE on Cray



Micro Benchmark: Scheduler

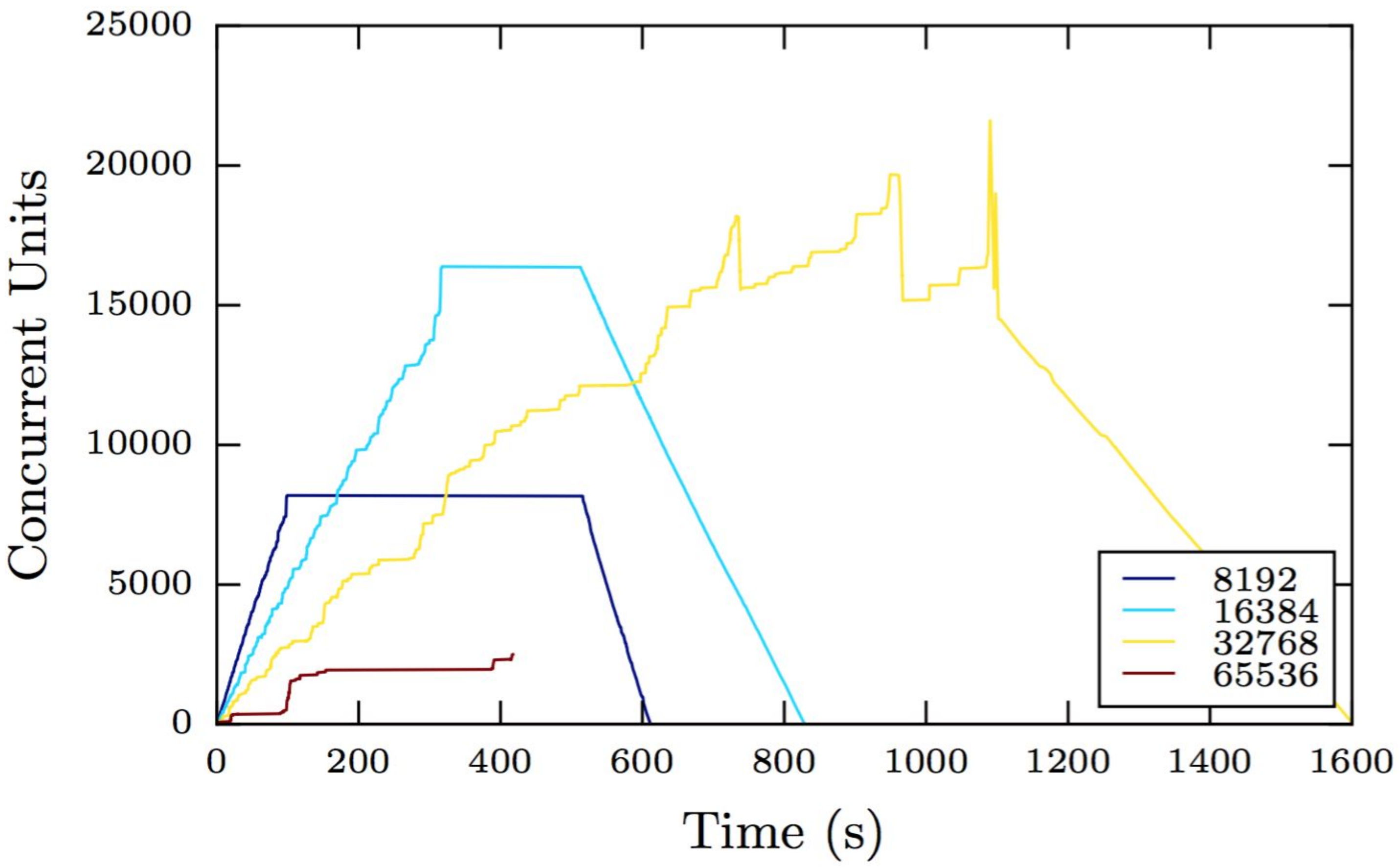


Scheduling only

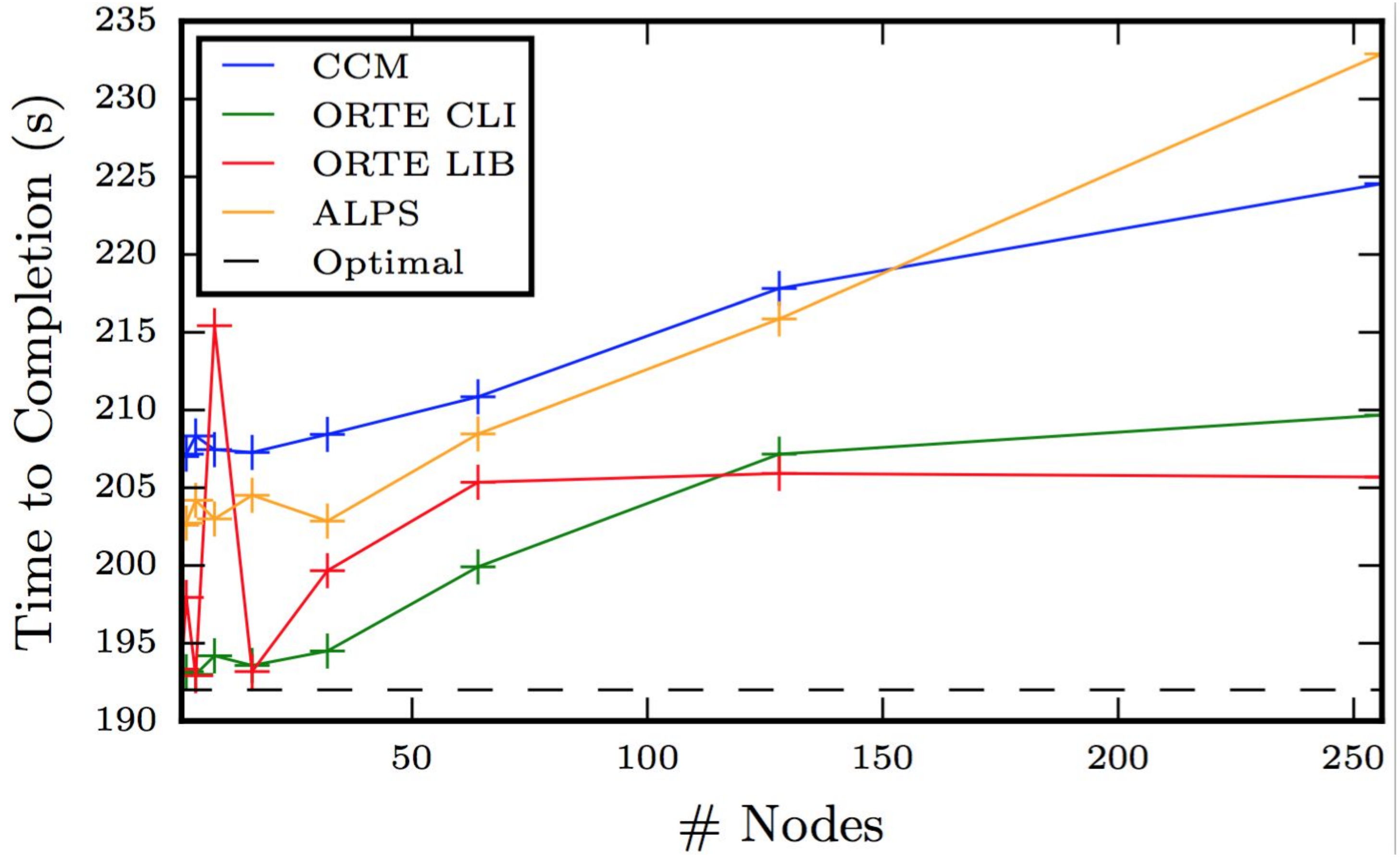


Scheduling and unscheduling

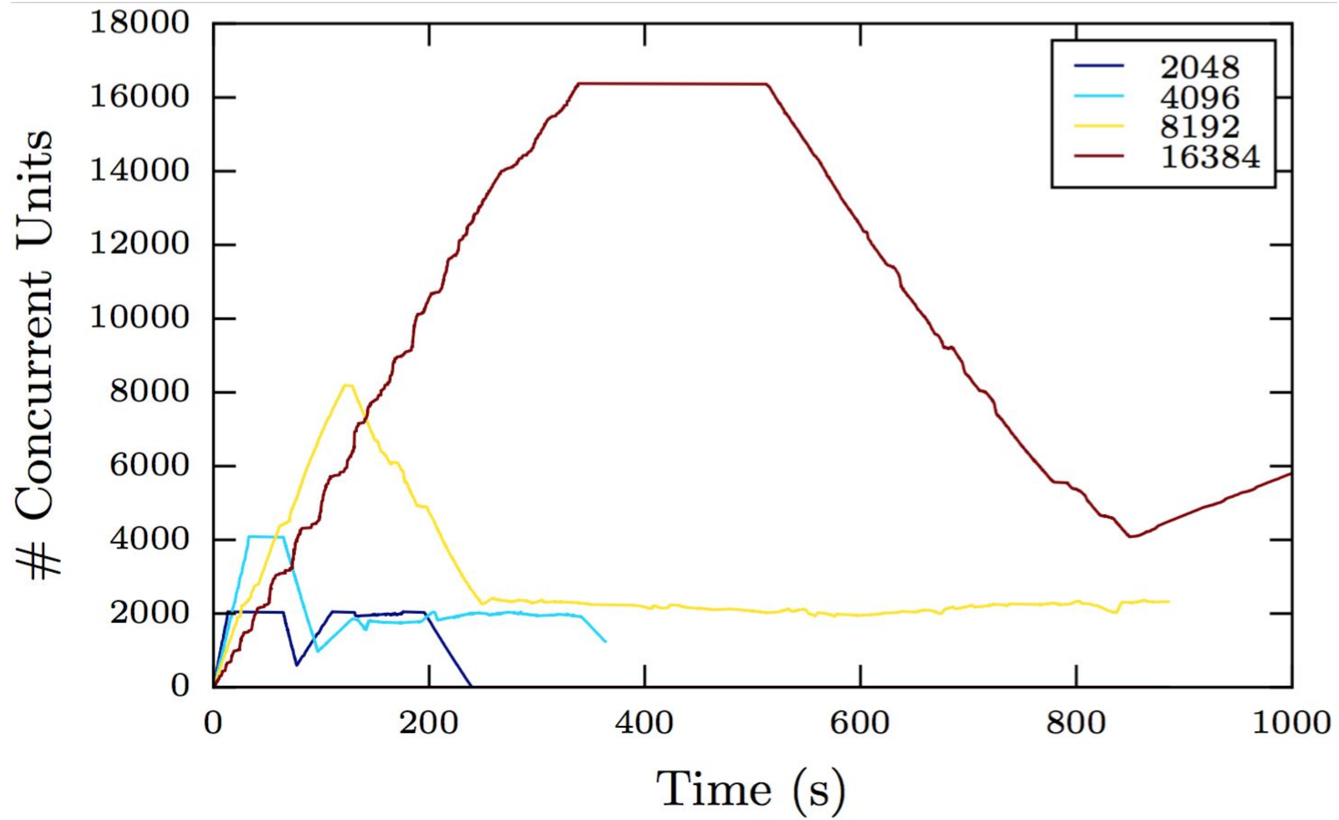
Micro Benchmark: Executor Scaling



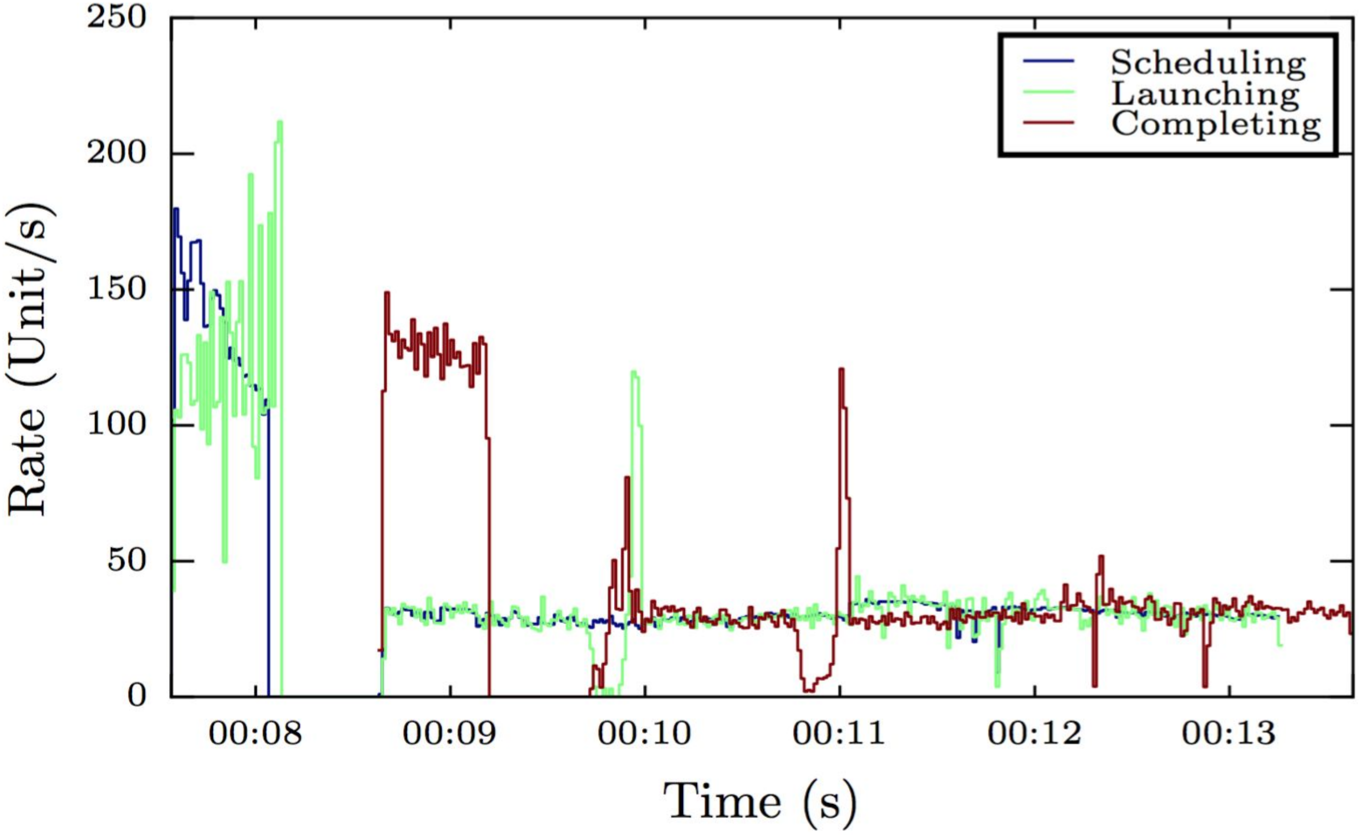
Agent Performance: Full Node Tasks (3 x 64s)



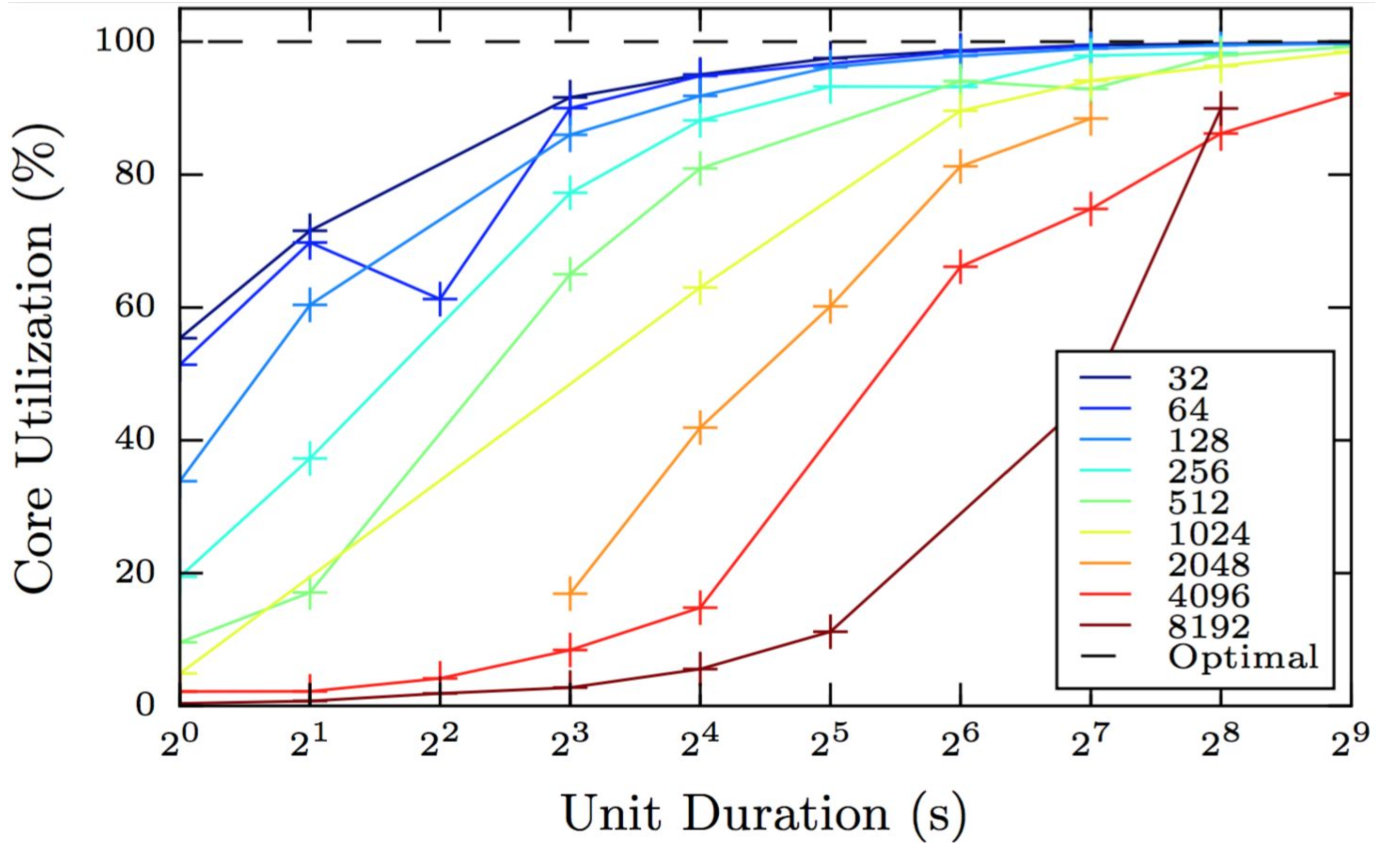
Agent Performance: Concurrent Units (3x)



Agent Performance: Turnaround (3 x 4k x 64s)



Agent Performance: Resource Utilization



Conclusion

- There is no “one size fits all” in HPC
- With general tools extend functionality of Cray HPC systems
- Achieved 16k concurrent tasks
- Launch rate of ~100 tasks / second
- Efficiency large dependent on task count and duration
- Cray specific PMI excludes running Cray MPI linked applications

Future work

- RADICAL-Pilot
  - Bulks all the way
  - Agent scheduler overhaul
  - Topology aware task placement
  - Heterogenous node scheduling (I.e. GPU)

- ORTE
  - Fabrics-based inter-ORTE communication
  - Optimize ORTE communication topology

References

- RADICAL-Pilot: Scalable Execution of Heterogeneous and Dynamic Workloads on Supercomputers
  - http://arxiv.org/abs/1512.08194
- A Comprehensive Perspective on the Pilot-Job Systems
  - http://arxiv.org/abs/1508.04180
- RADICAL-Cybertools overview
  - http://radical-cybertools.github.io/
- RADICAL-Pilot Github
  - https://github.com/radical-cybertools/radical.pilot
- RADICAL-Pilot Documentation
  - http://radicalpilot.readthedocs.org/

Micro Benchmark: Exec Rate + Concurrency (1x4k)