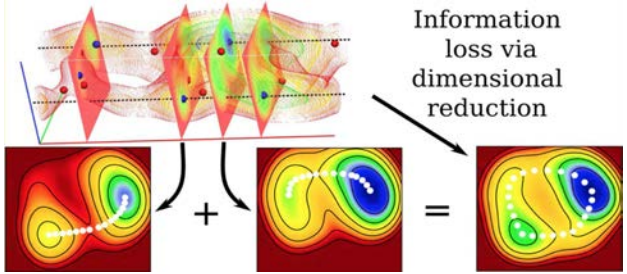
The Power of Many: Scalable Execution of Heterogeneous Workloads

Shantenu Jha
Research in Advanced DIstributed Cyberinfrastructure & Applications Laboratory (RADICAL)
http://radical.rutgers.edu & http://radical-cybertools.github.io

Landscape of Biomolecular Simulations (BMS)

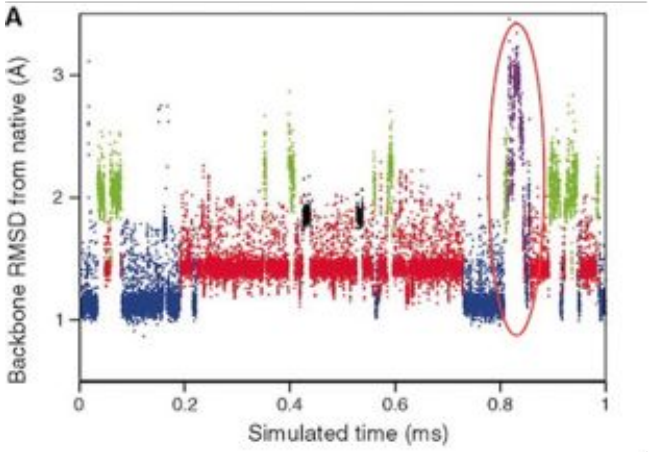
- Larger biological systems
  - Weak scaling
  - Status Quo: Size of systems: > 10M atoms
- Long time scale problem
  - Strong scaling
  - Status Quo: Duration of systems: > 10 ms
- Scaling challenges > than either single-partition strong and weak scaling.
  - Accurate estimation of complex physical processes, e.g., M-REMD
  - “New” Moore’s Law: **gap between weak scaling and strong scaling** capabilities will grow.

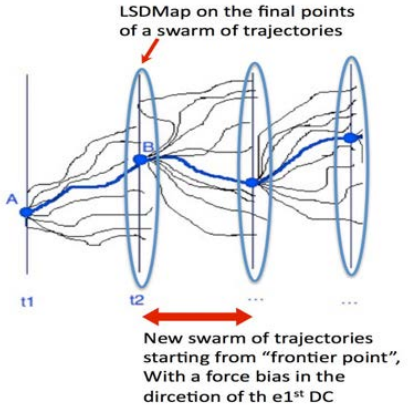


Multidimensional replica exchange umbrella sampling (REUS) simulations of a single uracil ribonucleoside.

Brief Introduction to Sampling

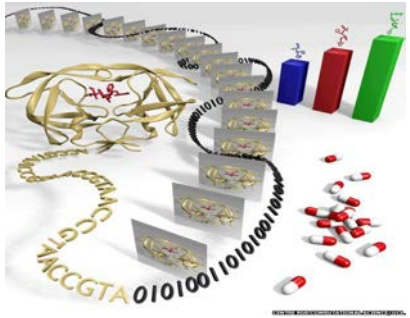
- Sampling: BPTI, 1ms MD simulation = 3 months on Anton (Shaw *et al*, Science 2010).
- *More* sampling, *Better* sampling, *Faster* sampling
- ***More sampling:*** Hundreds or thousands of concurrent MD jobs (nice introduction to Tom Cheatham)
- ***Better Sampling:*** Drive systems towards unexplored regions, don't waste time sampling behaviour already observed
  - DM-d-MD methods, AMBER-COCO

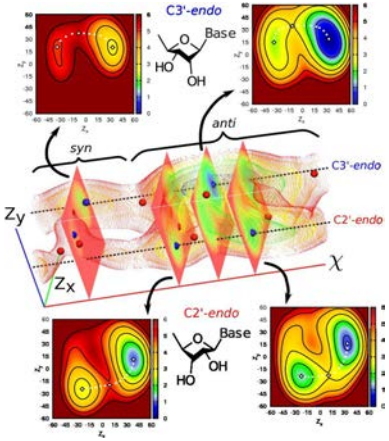


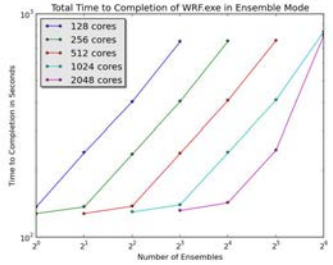


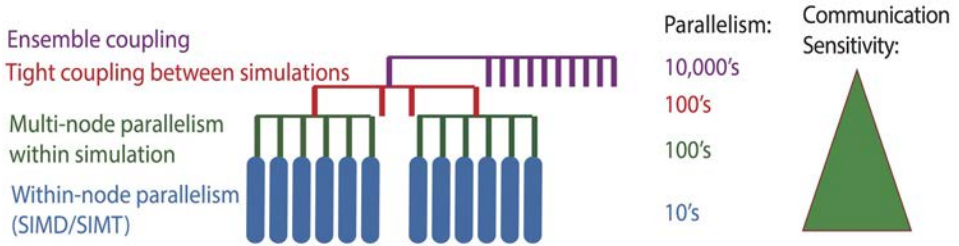
The Power of Many: Ensemble Approaches

- Many complex physical processes and utilize ensemble based approach:
  - Ensemble members often interact!
- Replica-exchange well known; many others interacting ensembles:
  - Adaptive Markov State Models
  - Adaptive biasing from ensembles
- Different degrees and levels of coupling between ensembles members:
  - RE: tight-coupling simulations
  - AMS: Coupling between ensembles









Ensemble Simulations: Computational Requirements

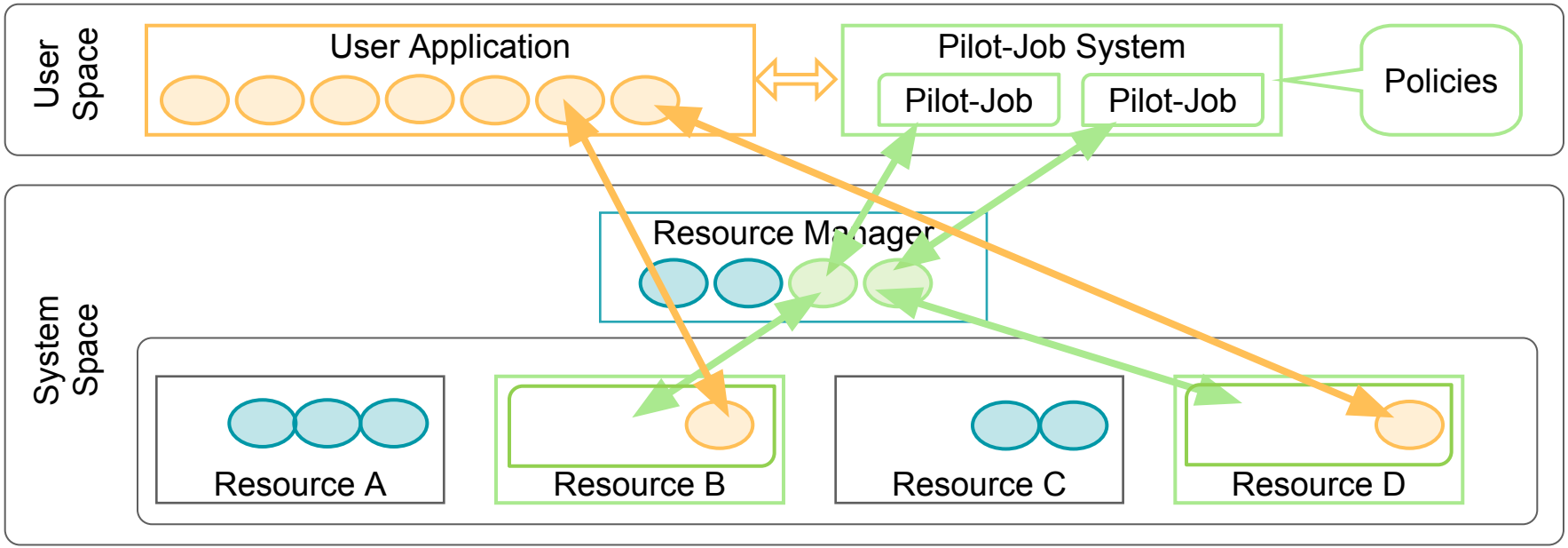
- Support for **heterogeneous** tasks
  - Multi-node and sub-node, application kernels, MPI/non-MPI
- Dynamic workload → set of tasks unknown *a priori*
  - Dynamic workload: Set of tasks and task relations
- Control over concurrency and coupling of tasks
  - Range of coupling (levels and degree) between ensemble members
- Multiple dimensions of scalability:
  - Concurrency: O(100K) tasks
  - Task size: O(1) - O(1000) cores
  - **Launch: O(1000+) tasks per second**
  - **Task duration: O(1) - O(10,000) seconds**

Why not the batch-queue?

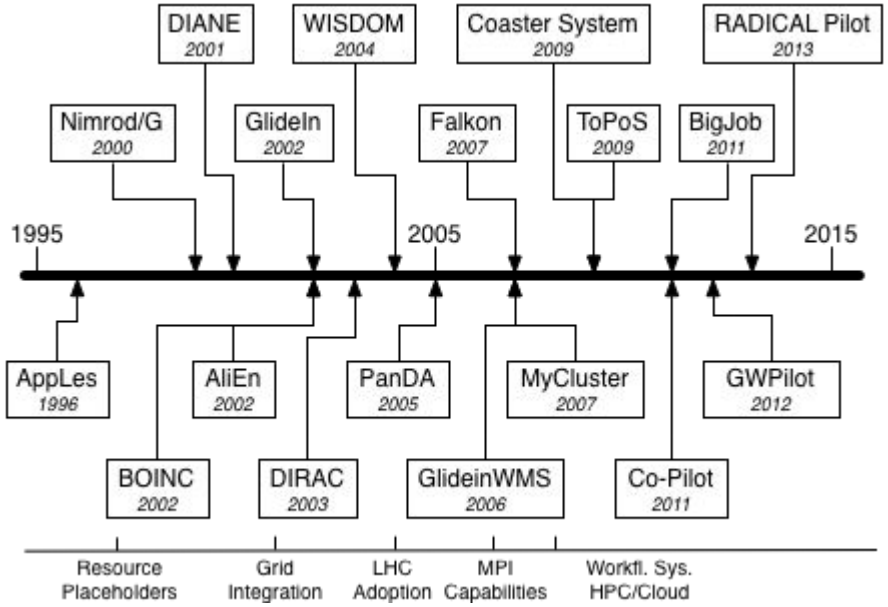
- Low throughput
  - Every job needs to queue
  - Breaks especially in dynamic workload situations
- No control over concurrency
- Limit on total concurrency
- Maximum of one task per node
- Job arrays are too inflexible (nor available on BW)
- Too many flavours ...

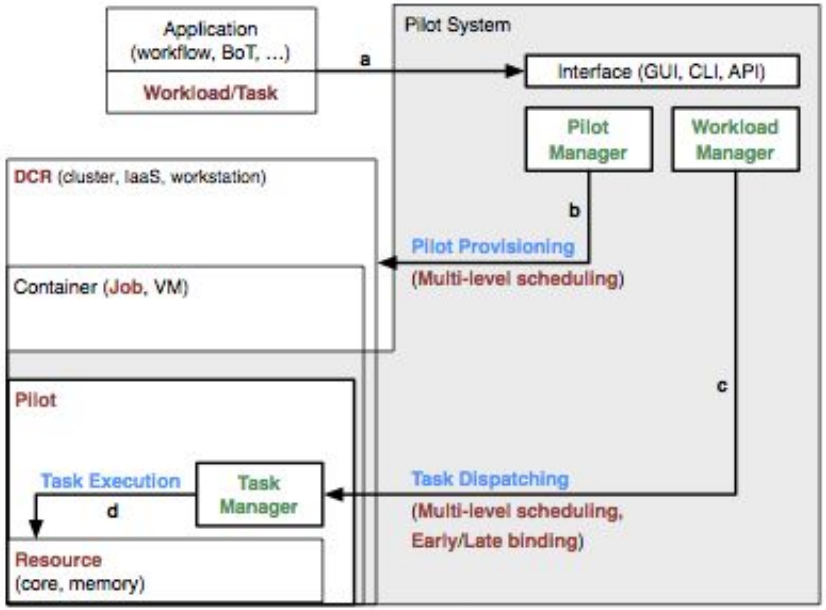
Pilot Abstraction: Schematic

- An abstraction that generalizes the reoccurring concept of utilizing a placeholder job as a container for a set of compute tasks.
- Finer grained spatio-temporal resource management.



P* Model for Pilot Abstraction: Conceptual Model

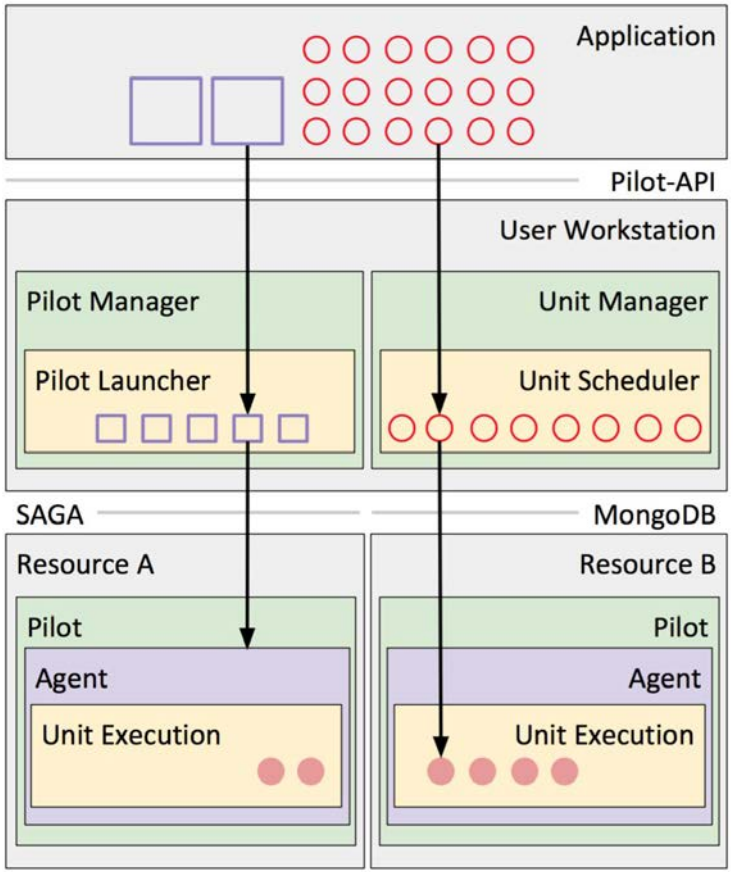




- P* Model of Pilot-Jobs
  - "P*: A Model of Pilot-Abstractions", *8th IEEE International Conference on e-Science 2012*
  - *A Comprehensive Perspective on Pilot-Jobs* http://arxiv.org/abs/1508.04180 *(2015)*

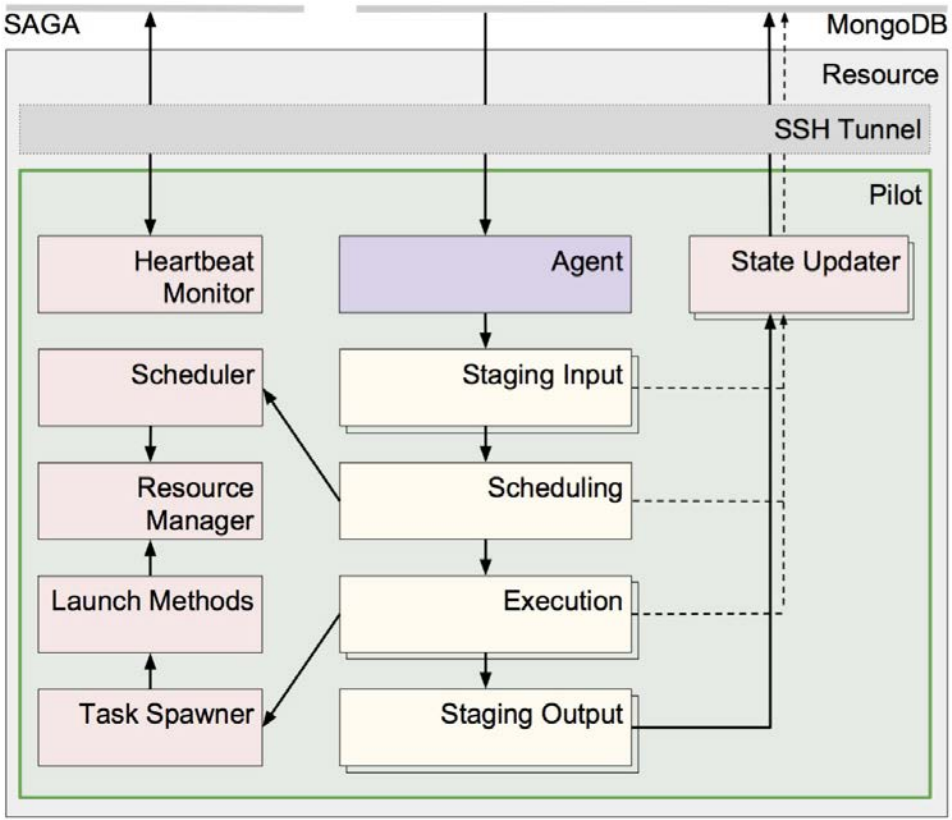
RADICAL-Pilot Overview

- **Programmable interface (arguably unique)**
  - Defined state models for pilots and units.
- **Supports research whilst supporting production scalable science:**
  - Agent, communication, throughput.
  - Pluggable components; introspection.
- **Portability and Interoperability:**
  - SAGA (batch-queue system interface)
  - Modular pilot agent for diff. Architectures.
  - Works on Crays, All XSEDE resources, most clusters, OSG, Amazon EC2...



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

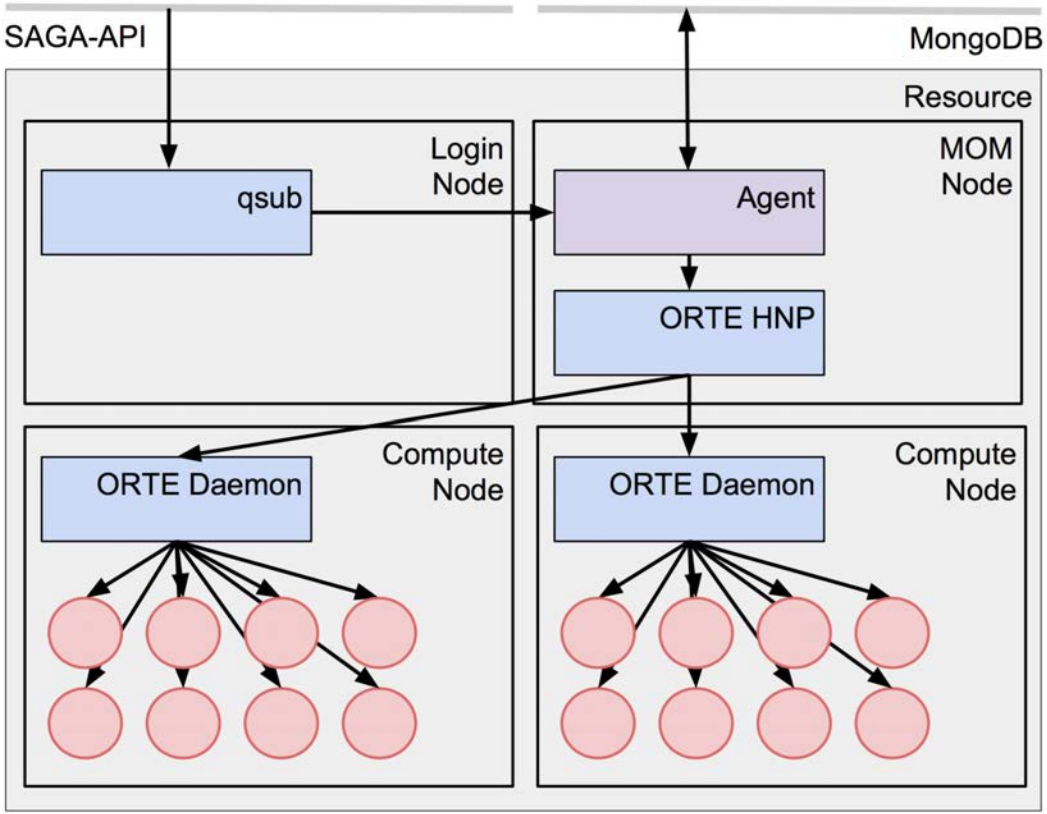
RADICAL-Pilot + ORTE-CLI (a bit better)

- 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, i.e., single core tasks.*

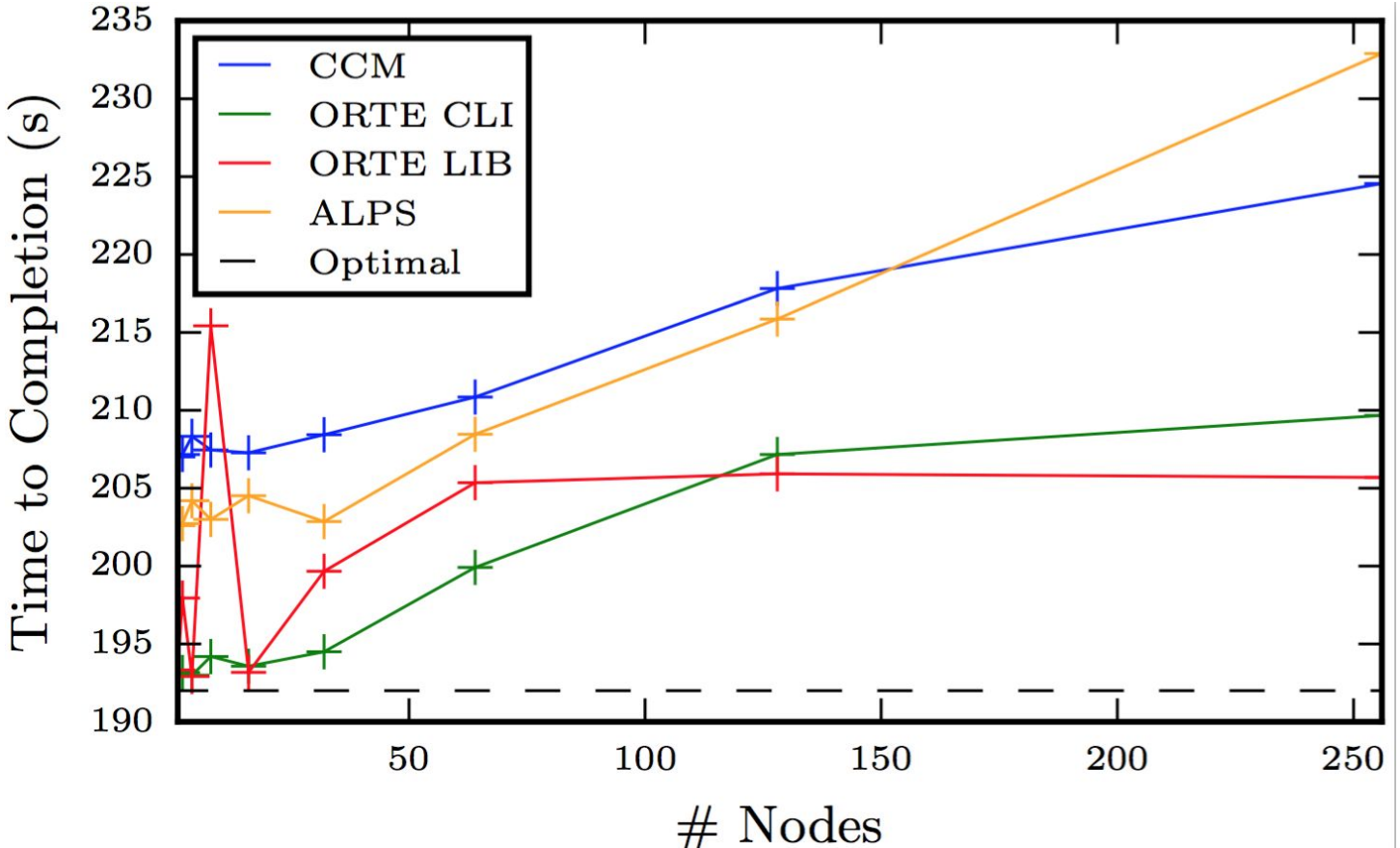
- `orte-submit` is CLI which submits tasks to those daemons
  - ‘sub-agent’ on compute node that executes these
  - Limited by fork/exec behavior
  - Limited by open sockets/file descriptors
  - Limited by file system interactions

RADICAL-Pilot + ORTE-LIB (much better)

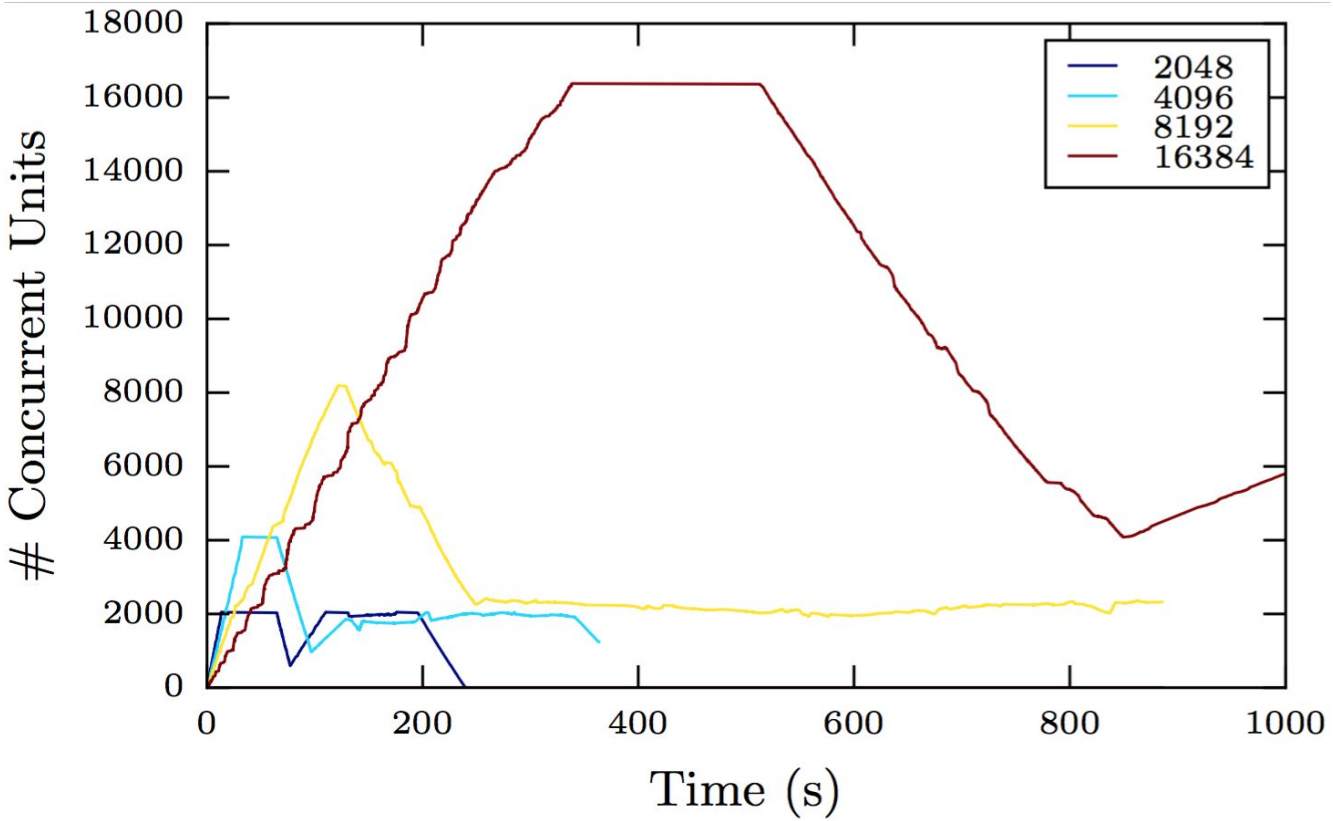
- All the same as ORTE-CLI, but
  - Uses library calls instead of `orte-submit` processes
  - No central fork/exec limits
  - Shared network socket
  - (Hardly) no central file system interactions



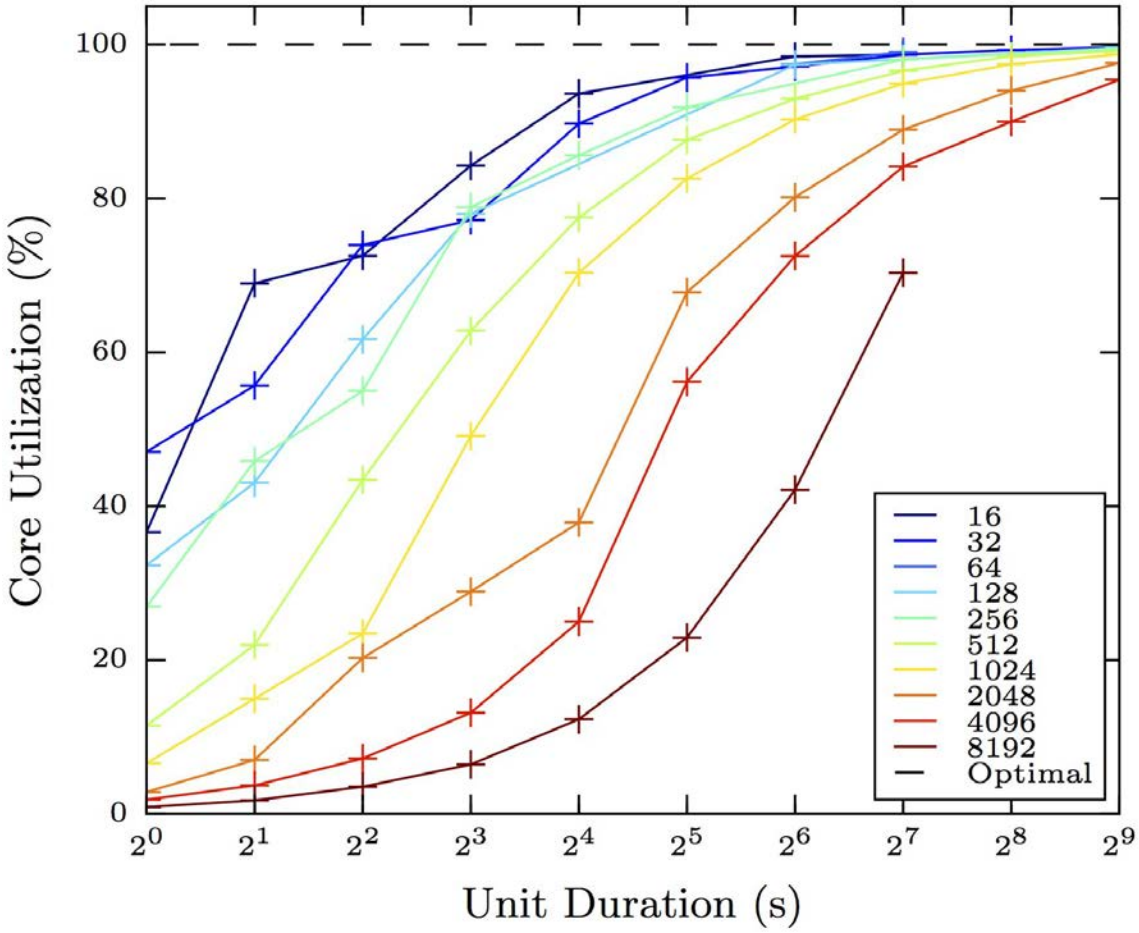
Agent Performance: Full Node Tasks (3 x 64s)



Agent Performance: Concurrent Units (3x)



Agent Performance: Resource Utilization



ExTASY on Blue Waters

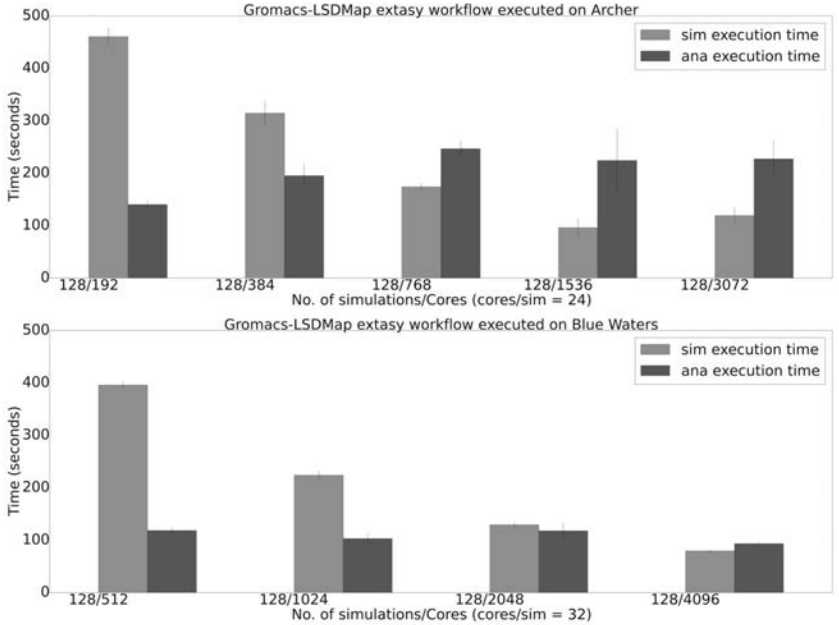


Fig. 7. Strong scaling of DM-d-MD workflow on ARCHER (**top**) and Blue Waters (**bottom**). The number of simulations is held constant at 128, number of cores per simulation at 24 on ARCHER and 32 on Bluewaters. The total number of cores used is varied with a constant workload, hence measuring strong scaling performance of the framework.

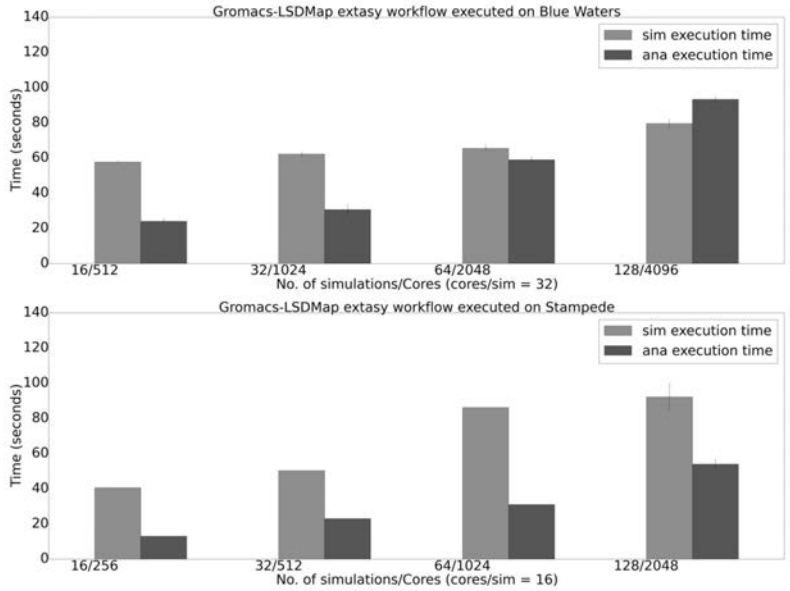


Fig. 9. Weak scaling of DM-d-MD workflow on Blue Waters (**top**) and Stampede (**bottom**). The number of cores per simulation is held constant at 32 on Blue Waters and 16 on Stampede. The total number of simulations is varied from 16-128 and the cores used are increased proportionally. By keeping the ratio of the workload to the number of resources constant, we observe the weak scaling performance of the framework.

Challenges of O(100K) Concurrent Tasks

- Agent communication layer (ZMQ) has limited throughput
  - limit is not yet reached
  - bulk messages (is implemented now)
  - separate message channels
  - code optimization
- Agent scheduler (node placement) does not scale well with number of cores
  - bulk operations (schedule bag of tasks at once)
  - good scheduling algorithms and implementations exist
  - code optimization
- Collecting complete jobs is just as hard as spawning new ones
  - decouple
- Interaction with DB and client side has limited scalability
  - replace with proper messaging protocol (also ZMQ?)

So why is it a challenge...

… to build “Scalable, Extensible and Flexible” tools?

- Every high-performance computing system/supercomputing environment is different, if not unique:
  - **Interoperability** across distinct interfaces and semantics: Batch queue systems, data access/movement etc.
  - **Portability**: Software environments, Simulation Kernel, Python, compilers, libraries, etc.
- **Generality is often orthogonal to performance:**
  - **Extensibility** + **Flexibility**: You can either make it general-purpose or high-performant.
  - **In practice**: find the sweet spot between general-purpose and performance

Conclusion

- Many problems will benefit from ensemble-based “simulation algorithms”. More than bag-of-tasks! Requires more than traditional strong/weak scaling!
- Efficiency large dependent on task count and duration
  - Achieved 16K concurrent tasks
  - Launch rate of ~100 tasks / second
- There is no “one size fits all” in HPC: General tools extend functionality of Cray HPC systems (with NCSA/OpenMPI)
- Cray specific PMI excludes running Cray MPI linked applications
- **Working towards a runtime that will support O(100K), while reasoning about asynchronous, non-linear and adaptive workloads.**

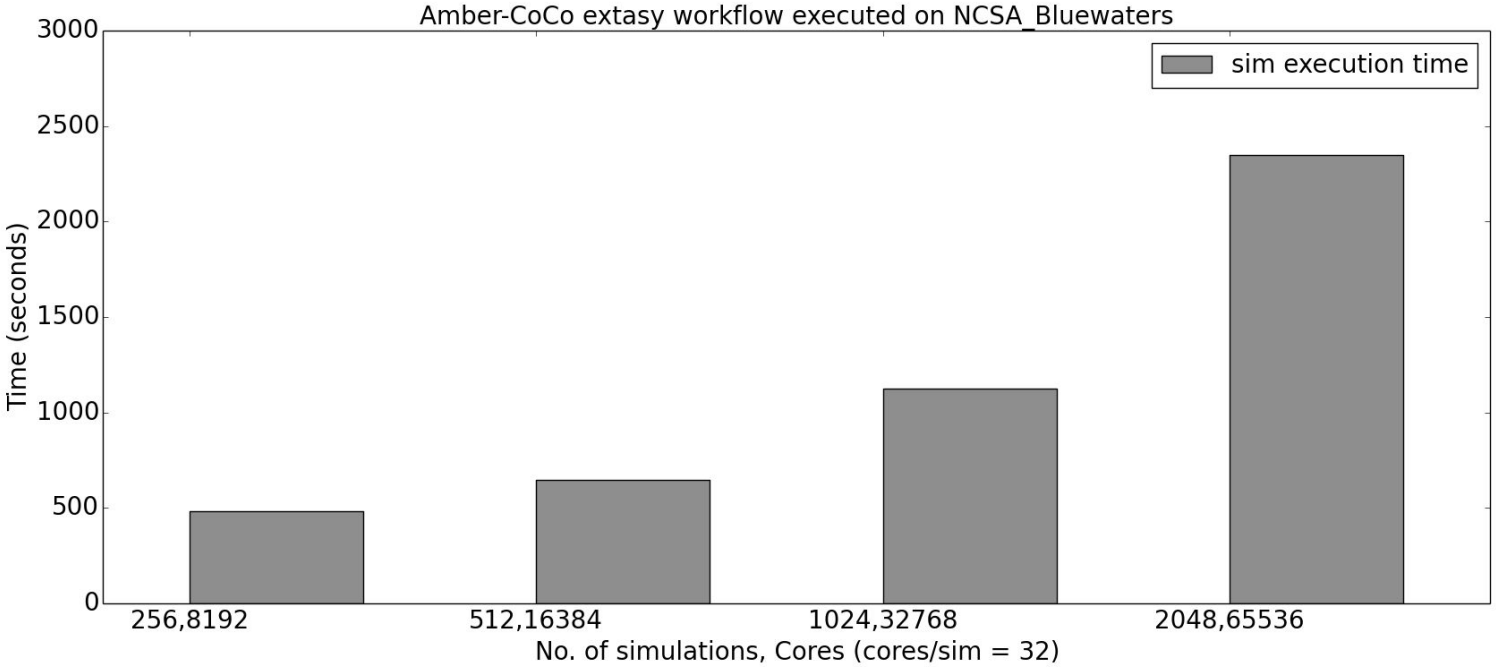
References

- 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-Pilot Github
  - https://github.com/radical-cybertools/radical.pilot
- RADICAL-Pilot Documentation
  - http://radicalpilot.readthedocs.org/
- Executing dynamic heterogeneous workloads on BW with RADICAL-Pilot
  - https://cug.org/proceedings/cug2016_proceedings/includes/files/pap130.pdf
- ExTASY: Scalable and Flexible Coupling of MD and Advanced Sampling.
  - http://arxiv.org/abs/1606.00093

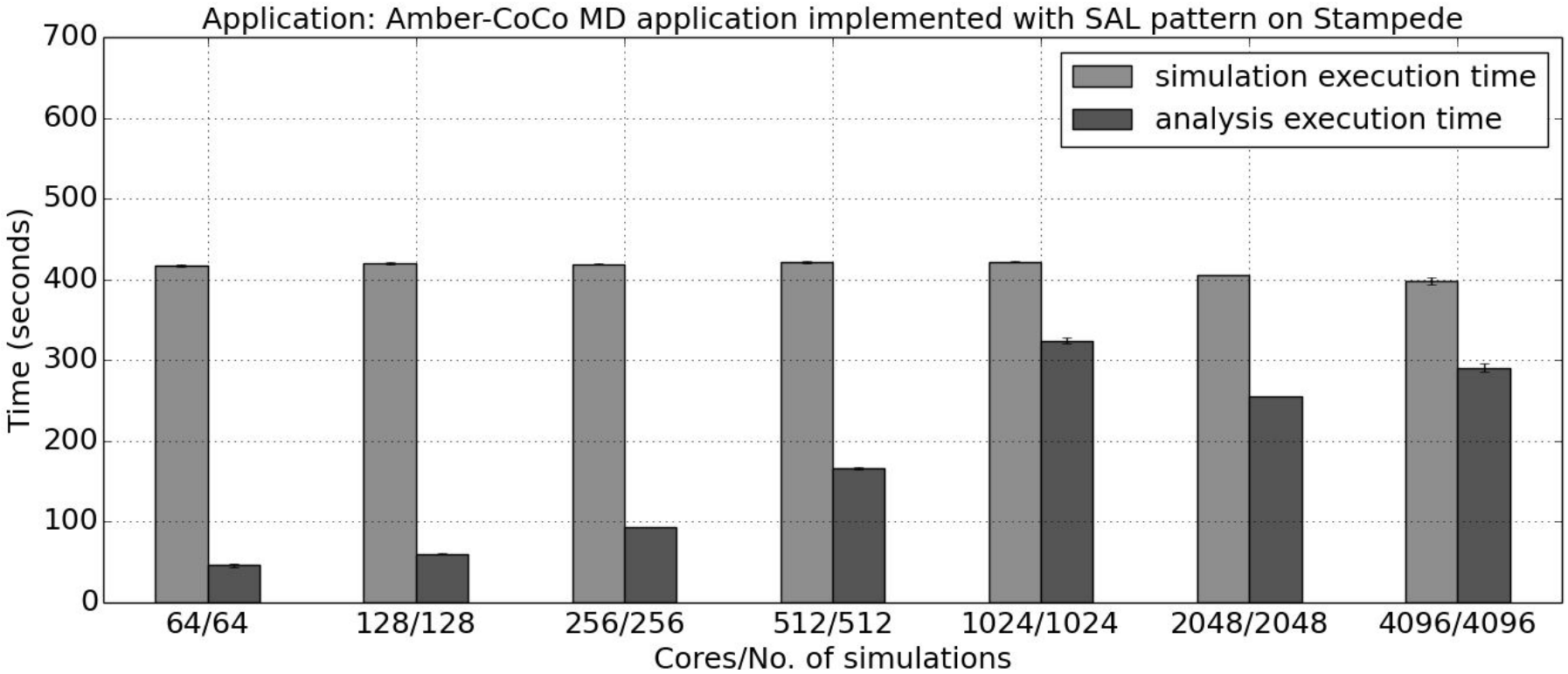
Acknowledgements

- Mark Santcroos, Andre Merzky and RADICAL Group (http://radical.rutgers.edu)
- Greg Bauer (Blue Waters/NCSA)
- Ralph Castain (OpenMPI Forum), Iain Bethune
- Kasson Lab (http://faculty.virginia.edu/kassonlab/Research_Interests.html)
- Michael Shirts (Colorado) and Tom Cheatham (Utah) for useful discussions
- Funding agencies: NSF and DOE

ExTASY on Blue Waters

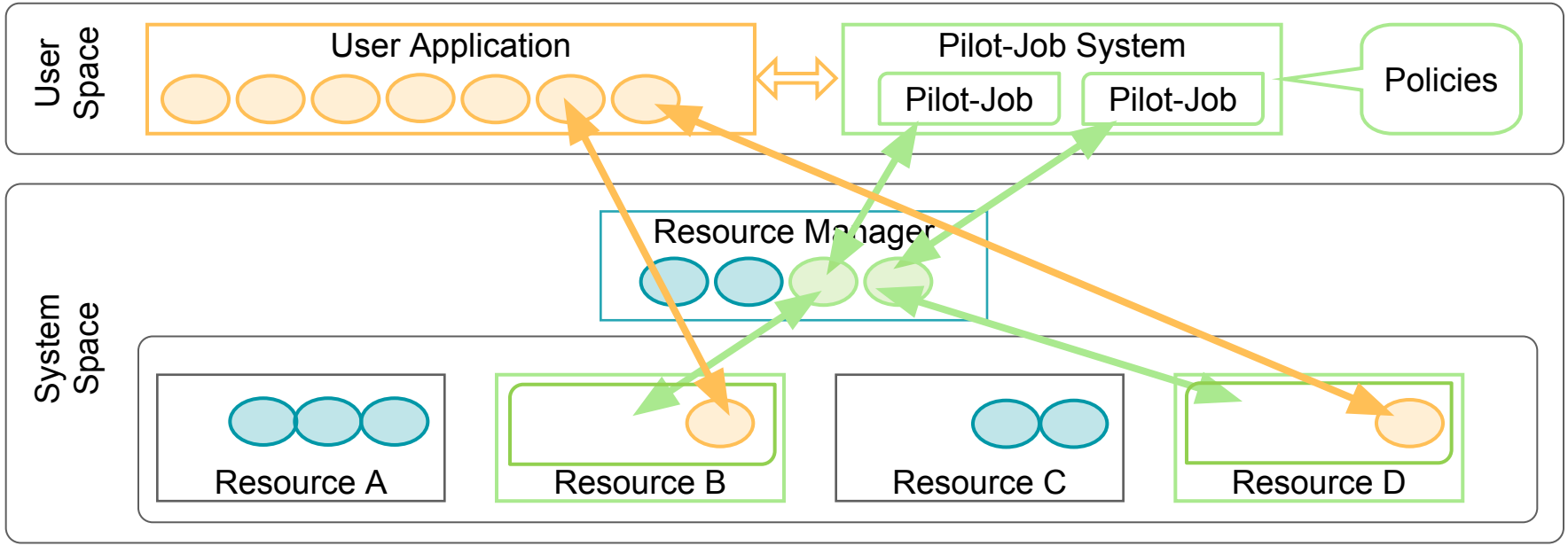


Ensemble Toolkit for Ensemble Approaches



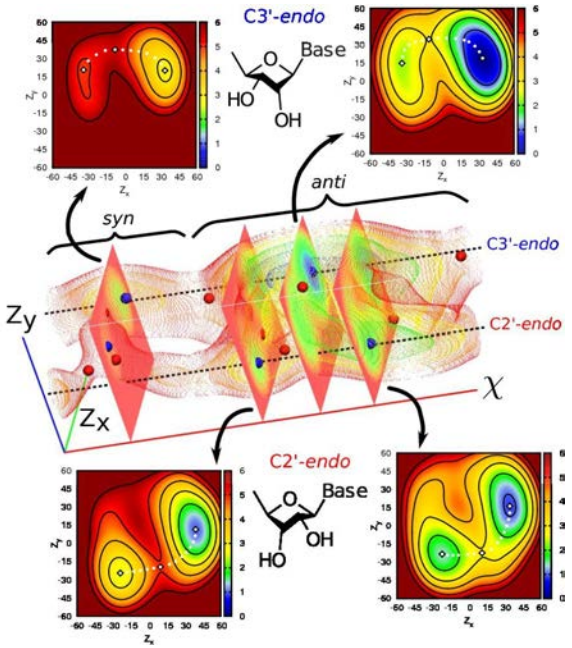
Abstraction: Pilot

- An abstraction that generalizes the reoccurring concept of utilizing a placeholder job as a container for a set of compute tasks.
- Finer grained spatio-temporal resource management.



Landscape of Biomolecular Simulations (BMS)

- Larger biological systems
  - Weak scaling
  - Status Quo: Size of systems: > 10M atoms
- Long time scale problem
  - Strong scaling
  - Status Quo: Duration of systems: > 10 ms
- Capture “Real” complexity
  - Molecular force-fields i.e., interactions
- Accurate estimation of complex physical processes
  - Robust uncertainty quantification



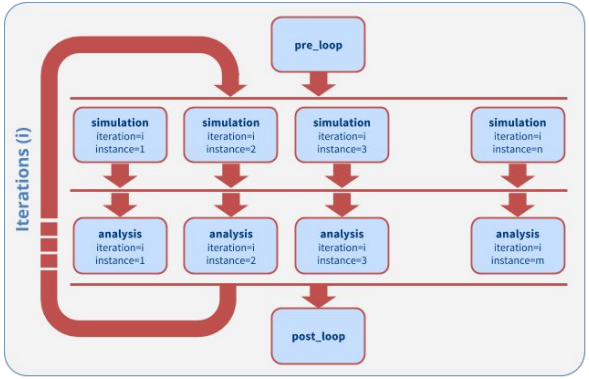
Scaling challenges are greater than single-partition strong and weak scaling.

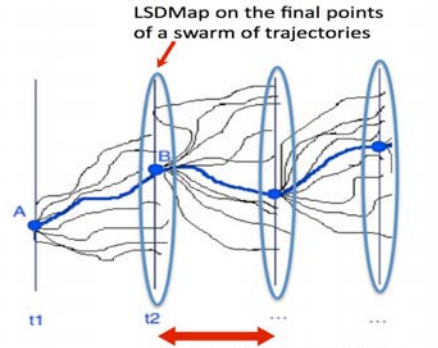
BMS: Status Quo

- Existing systems and solutions have imbalances in the “capability point”.
- Linear extrapolation of existing systems will lead to a different “capability point”, but not necessarily more balanced.
  - “New” Moore’s Law: **gap between weak scaling and strong scaling** capabilities will grow.
- Accurate estimation of complex physical processes is currently limited in sophistication and scale, will likely become more acute:
  - **Traditional use of HPC: Optimize “ecosystem” for single large job.**
  - Missing abstractions, gaps in capabilities when number of concurrent > 1.

Ensemble Toolkit for Ensemble Approaches

- Ensemble ToolKit (EnTK) to support the large-scalable execution of ensembles
  - http://arxiv.org/abs/1602.00678
- Promotes ensemble “execution patterns” independent of simulation kernel and analysis
  - API, “translation layer” and runtime system
  - Most common execution patterns
    - Iterative simulation → analysis
- The coupling between ensemble members, and their evolution is essentially pre-defined
  - **Static** execution patterns





ExTASY Project. Courtesy Cecilia Clementi

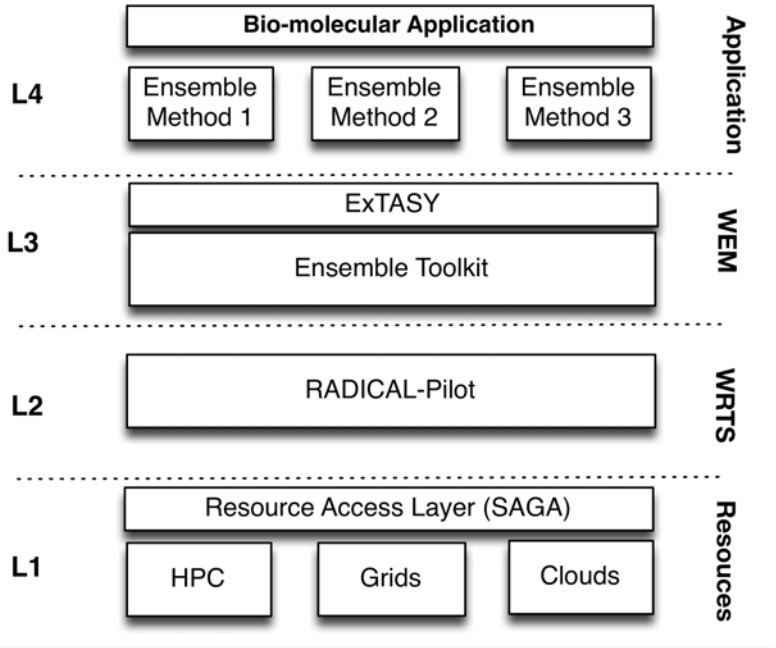
Computational Requirements: L3

L3: Framework for dynamic workloads and many adaptive execution scenarios.

- Support methods that have asynchronous, adaptive and non-linear execution.
- How to best support dynamic workloads? Capture adaptive scenarios?
  - Ease and reuse of components to capture the richness of adaptivity
  - Copernicus (Data-flow) and Ensemble Toolkit (Control-flow)?
- Computationally steered execution?
  - Internal versus externally steered (streaming data)
- **“Special purpose” or “general purpose” workflow approaches?**
  - When to build a domain specific language versus adapt/extend existing?

Computational Requirements: Layered View

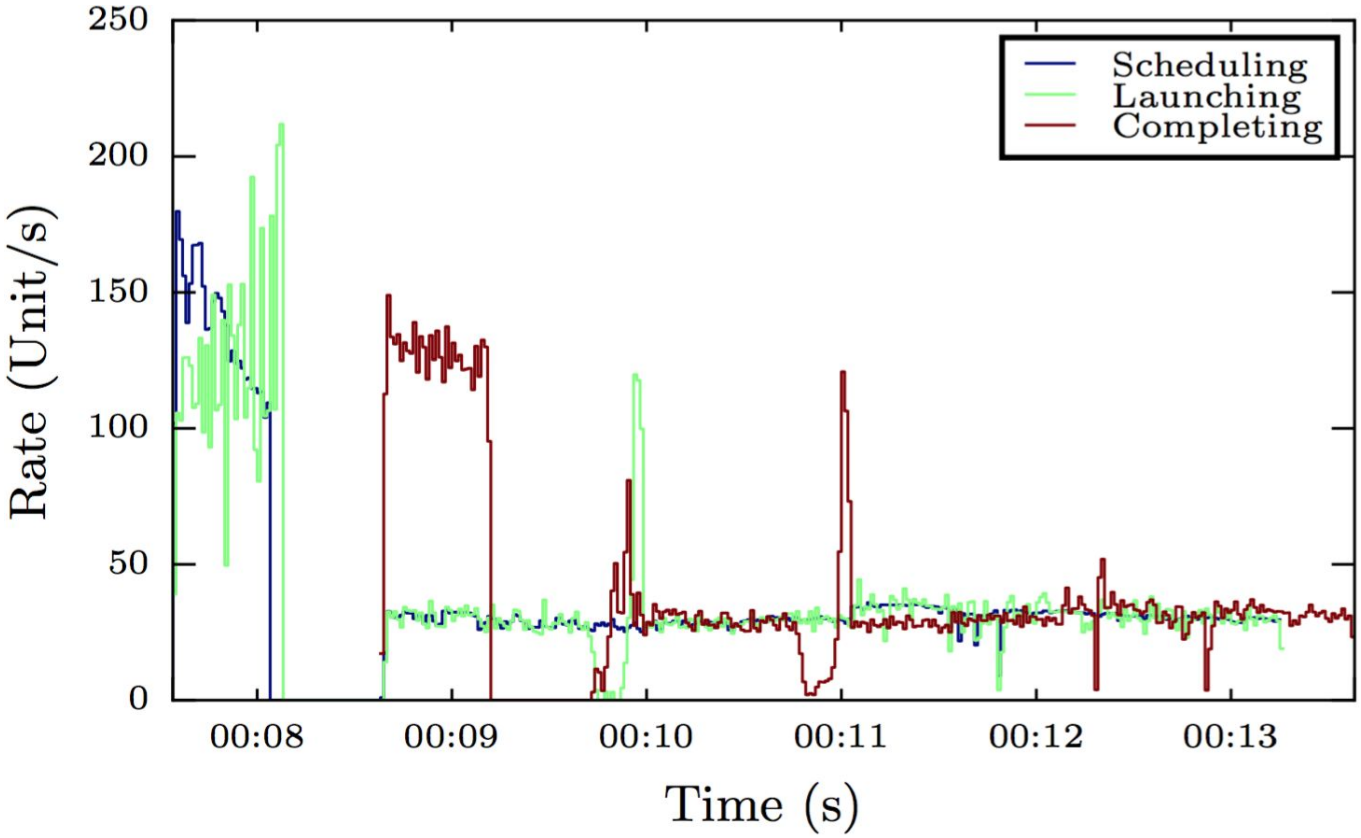
- To systematically analyze the research questions, propose four layers:
  - L4: Application/Workflows
  - L3: Workload execution and management (**WEM**)
  - L2: Task runtime system (**WRTS**)
  - L1: Resource layer
- Entities at each layer:
  - L4: Application semantics
  - L3: Workloads
  - L2: Tasks
  - L1: Jobs



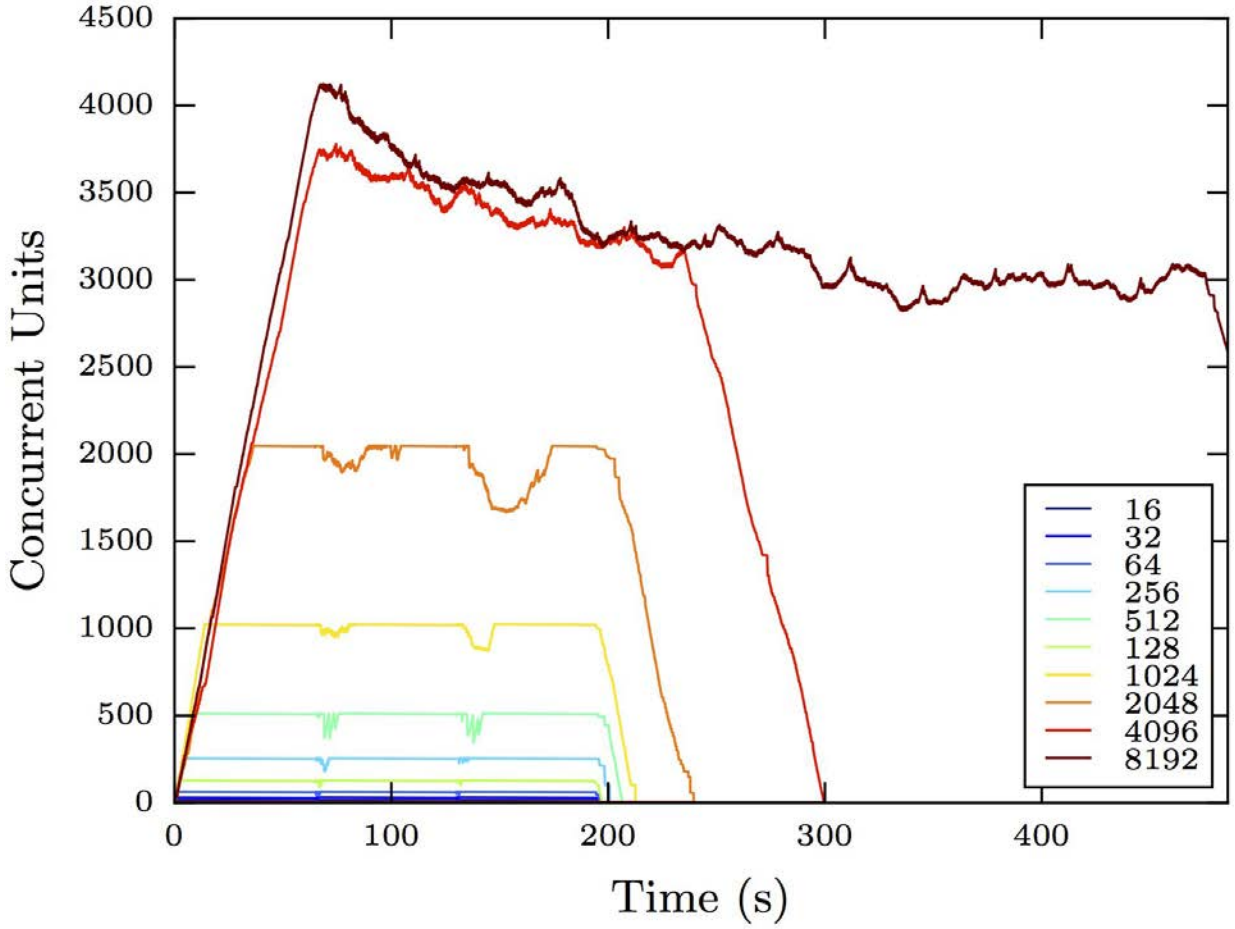
Pilot Abstraction: Overview

- Advantages of Pilot-Abstractions:
  - Decouples workload from resource management
  - Flexible Resource Management
    - Enables the fine-grained (ie “slicing and dicing”) of resources
    - Tighter temporal control and other advantages of application-level Scheduling (avoid limitations of system-level only scheduling)
  - Build higher-level frameworks without explicit resource management

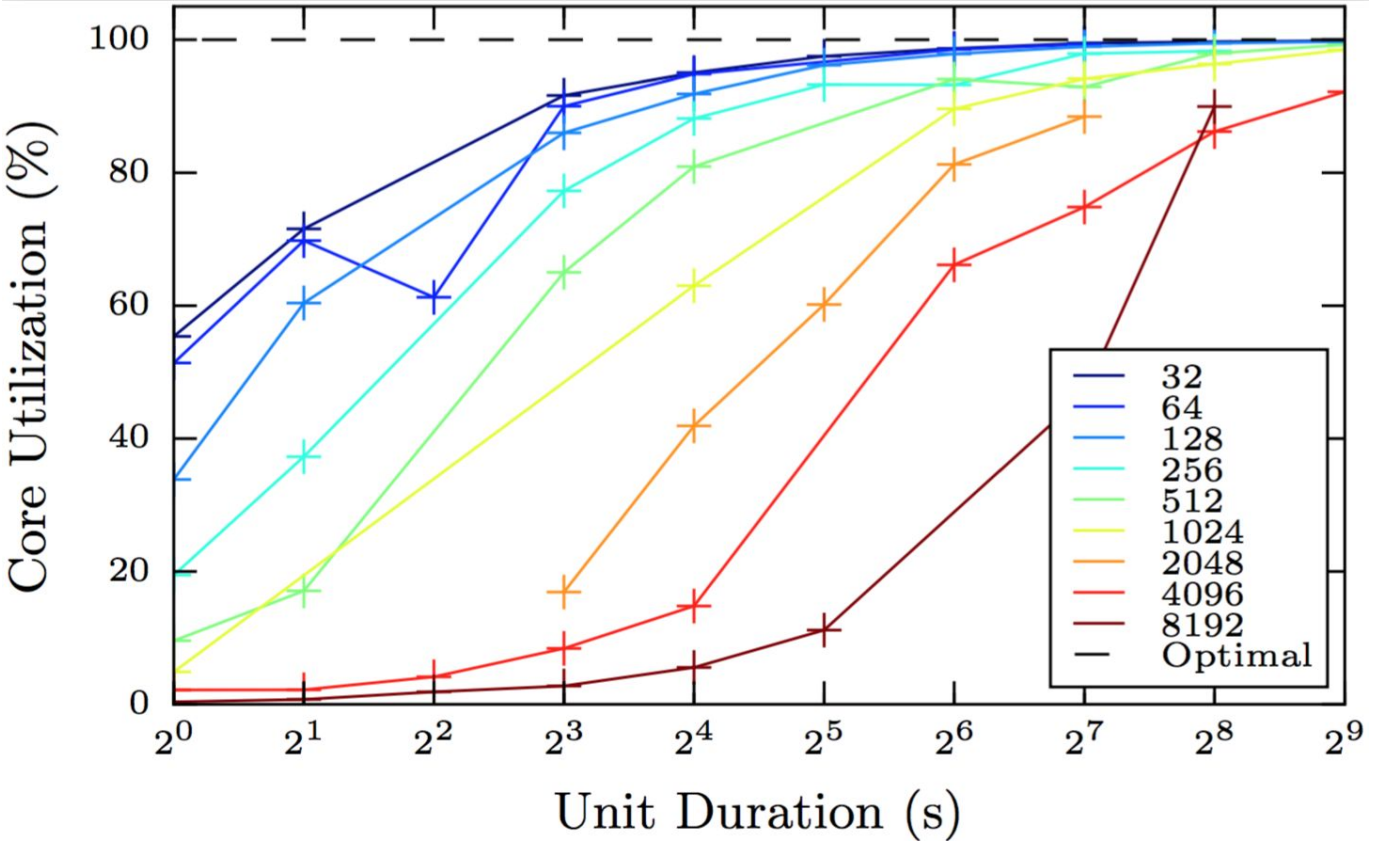
Agent Performance: Turnaround (3 x 4k x 64s)



Agent Performance: Concurrent Units



Agent Performance: Resource Utilization



The Power of Many: Scalable Execution of Heterogeneous Workloads

Shantenu Jha

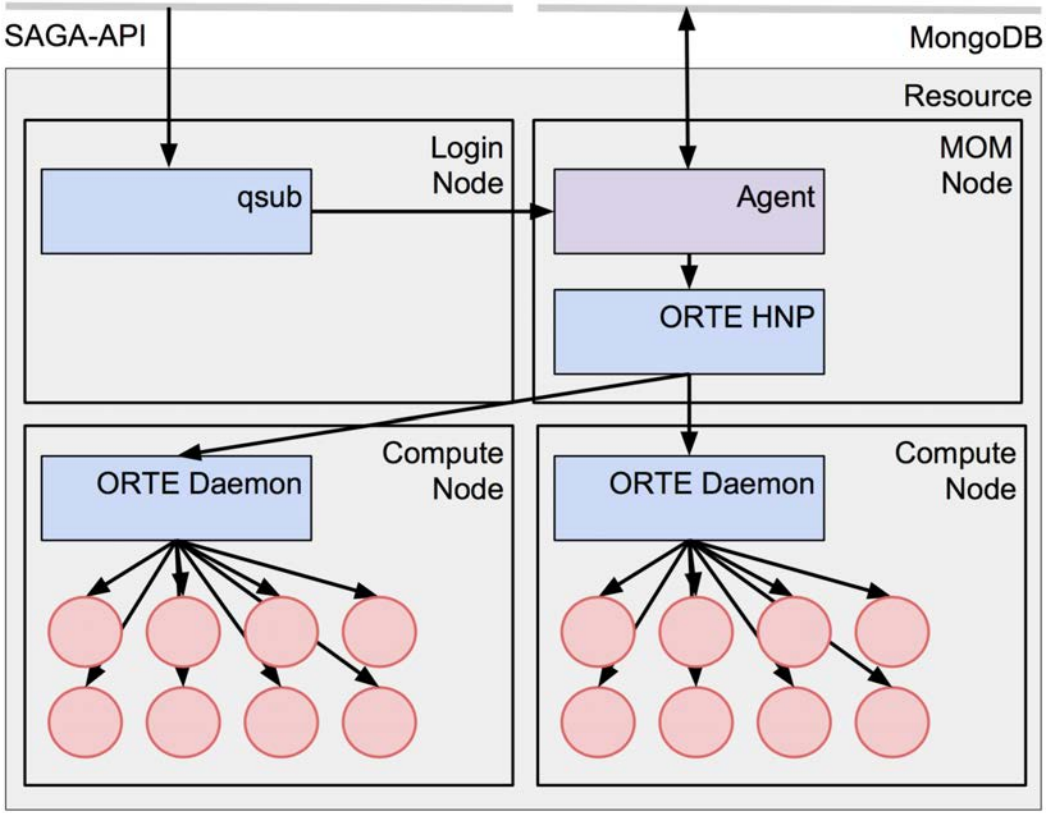
http://radical.rutgers.edu

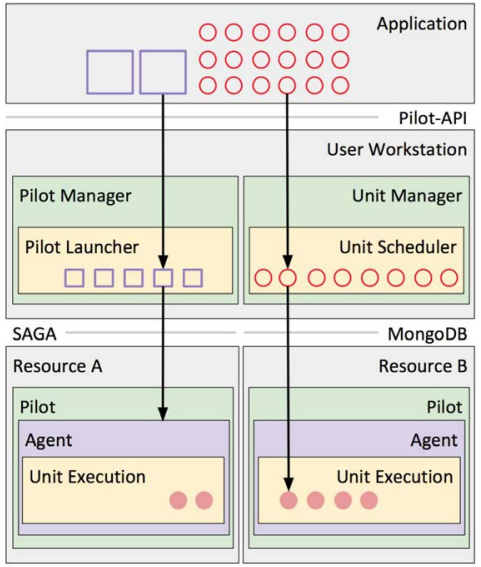
Blue Waters Symposium 2016

Overview of Sampling

- *More* sampling via more simulations
  - Hundreds or thousands of concurrent MD jobs
  - Manage execution, data movement, efficient HPC resource utilisation
- *Better* sampling via biased simulations
  - Don’t waste time sampling behaviour already observed
  - Drive systems towards unexplored regions
  - … and still obtain true thermodynamic distribution of states?
- *Faster* sampling via better algorithms
  - Couple the latest integration algorithms with existing MD codes
  - Increased timestep without loss of accuracy

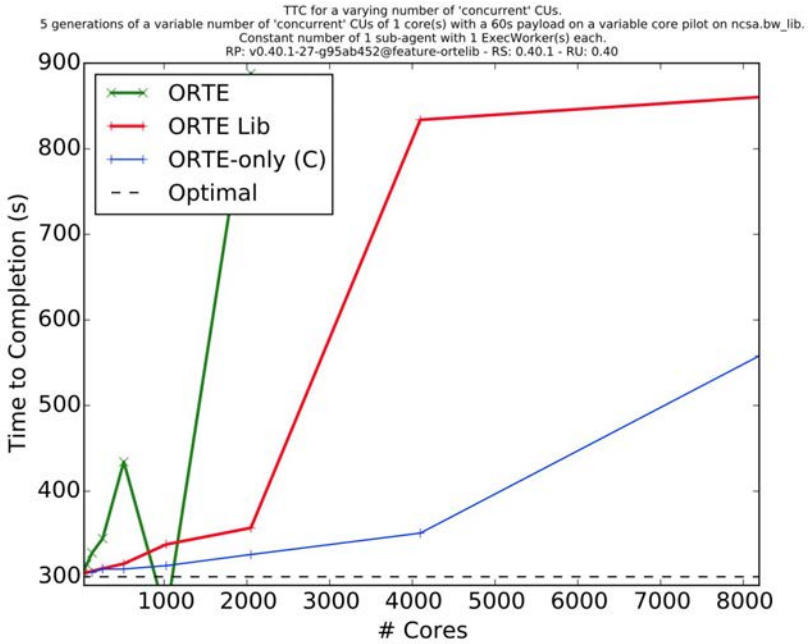
RADICAL-Pilot + ORTE on Cray





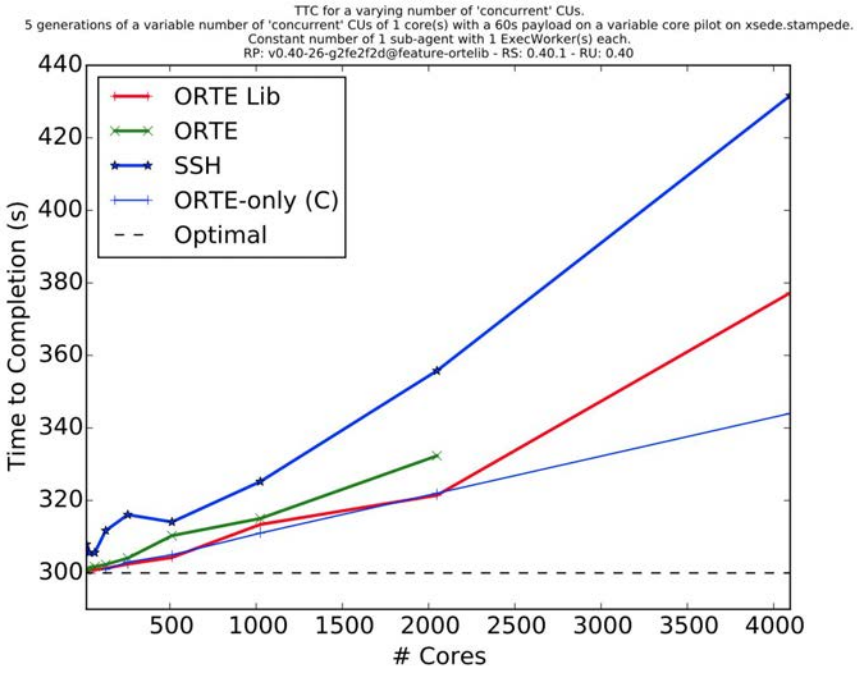
TTC Scalability: Variability

Blue Waters



!=

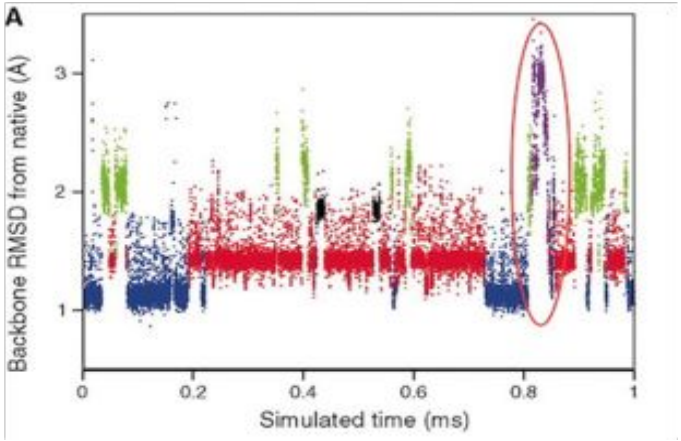
Stampede



Brief Introduction to Sampling

- Nice introduction to sampling by Tom Cheatham
- *More* sampling, *Better* sampling, *Faster* sampling

- Example: BPTI
  - 1ms MD simulation = 3 months on Anton (Shaw *et al*, Science 2010)



- 1 heroic simulation
  -> 1 (lucky?) observation of a new conformation
- Good sampling is a real problem!

Brief Introduction to Sampling

- Sampling: BPTI
  - 1ms MD simulation = 3 months on Anton (Shaw *et al*, Science 2010)
- *More* sampling, *Better* sampling, *Faster* sampling
- *More sampling:* Nice introduction to Tom Cheatham
- *Better Sampling:* Drive systems towards unexplored regions
  - ExTASY DM-d-MD methods, AMBER-COCO methods