Building Blocks for Adaptive Workflows

Shantenu Jha

Rutgers Advanced DIstributed Cyberinfrastructure & Applications Laboratory (RADICAL)

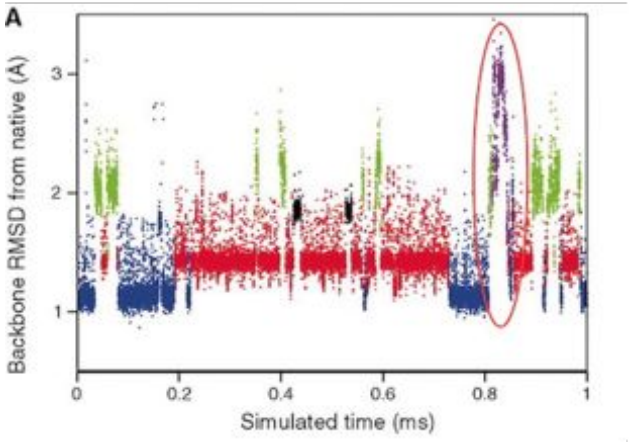
http://radical.rutgers.edu

Outline

- **Design computational experiments to utilize resources more effectively**
  - Research challenge in the design of resource management systems to support flexible and scalable experiments
- **AIMES Model for dynamic resource utilization**
  - Necessary condition for execution of **adaptive** workflows
- **Workflows: Biomolecular simulation algorithms that involve multiple ensembles.**
  - Science-based metrics of efficient resource utilization
- **Building blocks approach for design and implementation of adaptive workflows?**
  - Design and implementation of RADICAL-Cybertools

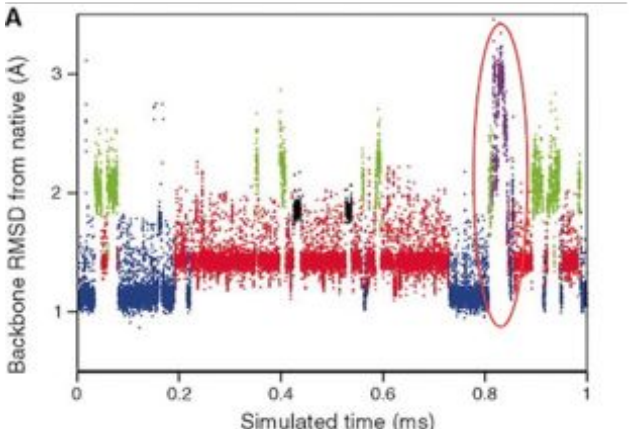
Biomolecular (MD) Simulations: Context

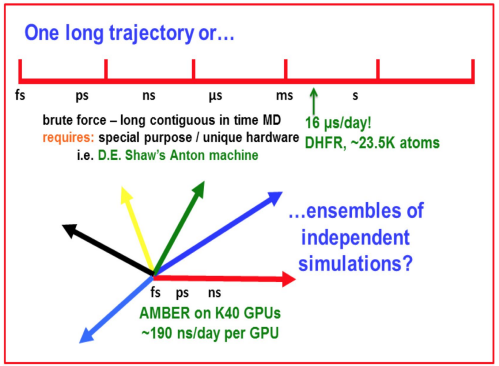
- Larger biological systems
  - Requires **weak** scaling
- Long time scale problem
  - Requires **strong** scaling
  - DE Shaw special purpose computer (Anton)
- Gap between weak and strong scaling capabilities will grow



Biomolecular (MD) Simulations: Context

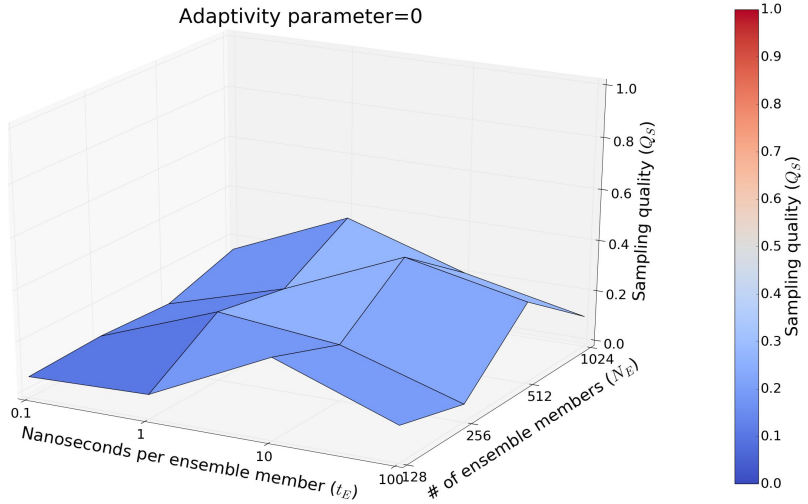
- Larger biological systems
  - Requires **weak** scaling
- Long time scale problem
  - Requires **strong** scaling
  - DE Shaw special purpose computer (Anton)
- Gap between weak and strong scaling capabilities will grow
  - Investigate solutions beyond single-partition strong/weak scaling
  - Ensemble simulations
- Given number of computing hours (= T_A) which is better:
  - Many simulations or longer running simulations?





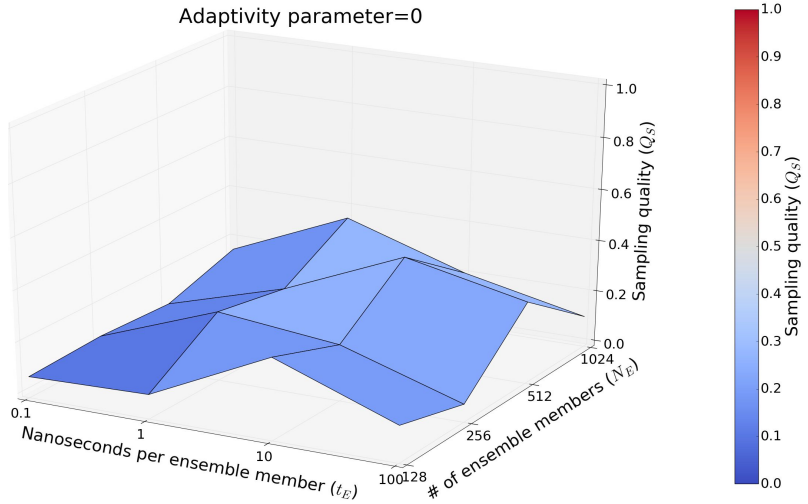
Optimal Resource Utilization Configuration

- Given a finite number of computing hours (= T_A) which is better:
  - Many simulations or longer running simulations?
- How to divide T_A across “many” simulations:
  - For given T_A : Greater N_E or larger t_E ?



Optimal Resource Utilization Configuration

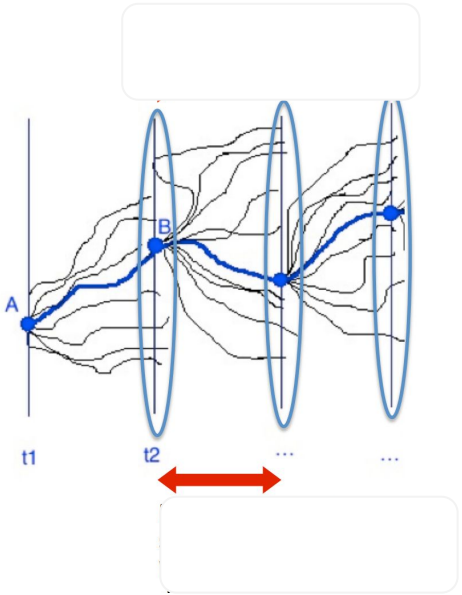
- Given a finite number of computing hours (= T_A) which is better:
  - Many simulations or longer running simulations?
- How to divide T_A across “many” simulations:
  - For given T_A : Greater N_E or larger t_E ?

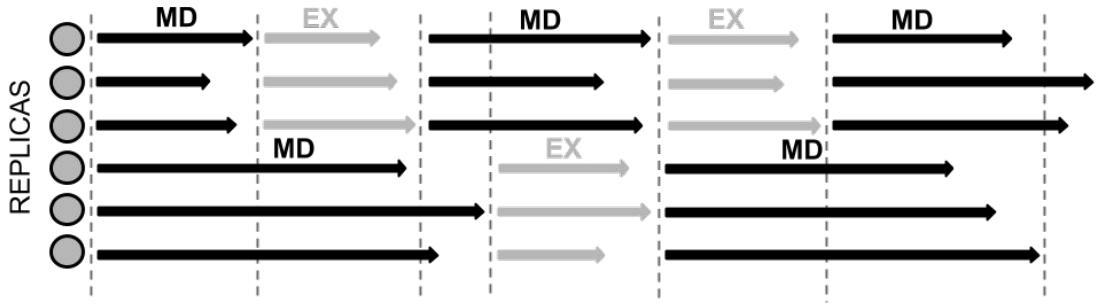


Break free of the static resource utilization and execution models.

Optimal Resource Utilization Configuration

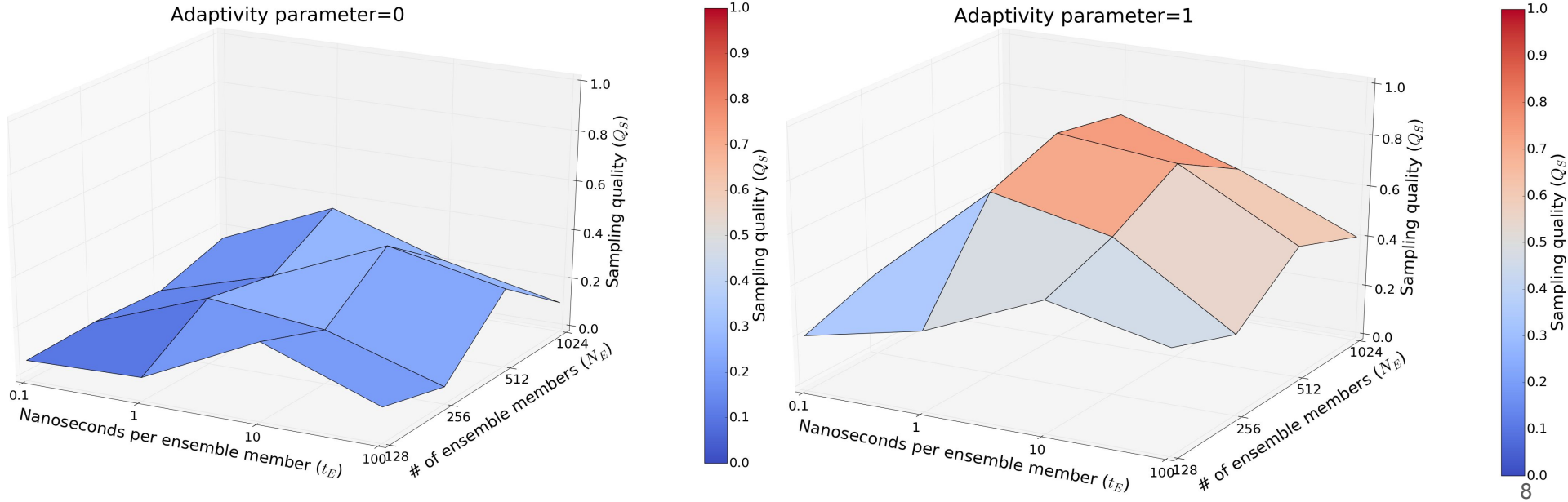
- How to divide T_A across “many” simulations:
  - For given T_A : Greater N_E or larger t_E ?
  - Non-adaptive (static) simulations vs adaptive simulations (replicas)?





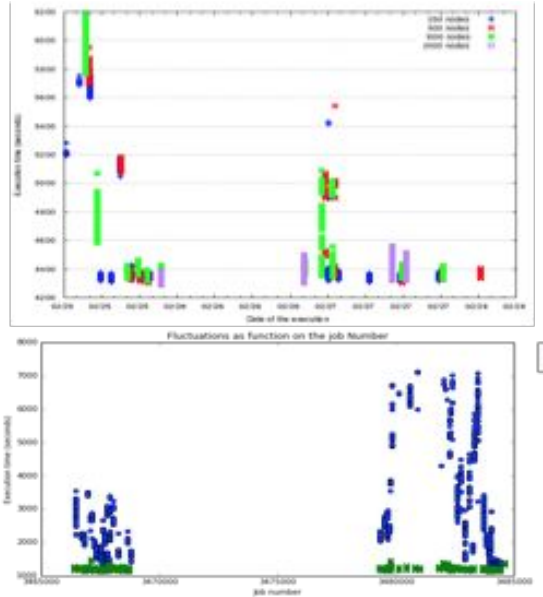
Optimal Resource Utilization Configuration

- How to divide T_A across “many” simulations:
  - For given T_A : Greater N_E or larger t_E ?
  - Non-adaptive (static) simulations vs adaptive simulations (replicas)?



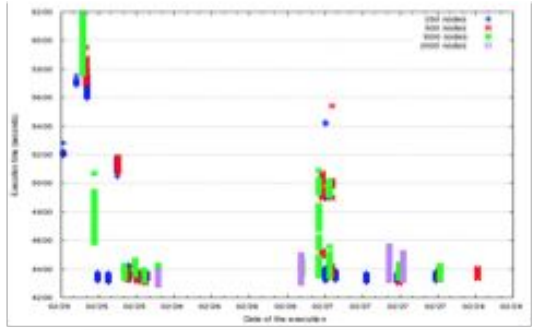
Runtime Fluctuation

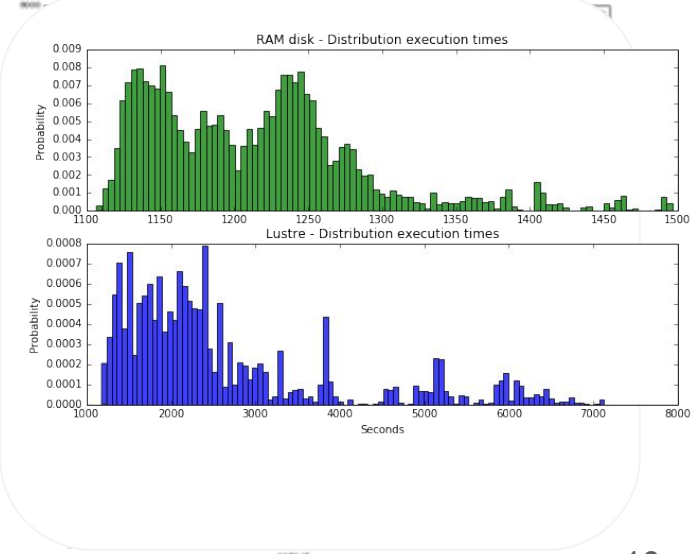
- **Fluctuations** in the task execution time (Tx) of concurrent tasks. Should have same Tx!
  - (Top) Molecular Dynamics (Gromacs)
  - (Middle) Athena-MP (ATLAS).



Runtime Fluctuation

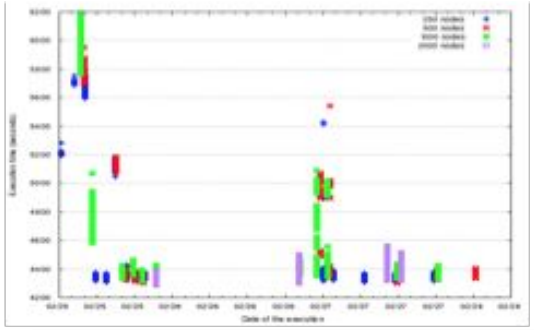
- **Fluctuations** in the task execution time (Tx) of concurrent tasks. Should have same Tx!
  - (Top) Molecular Dynamics (Gromacs)
  - (Middle) Athena-MP (ATLAS).
- Multiple contributing factors:
  - I/O and Lustre
  - Other shared resource contention

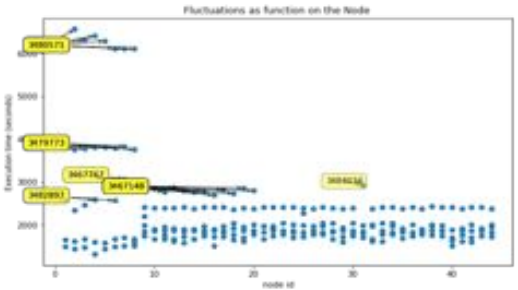




Runtime Fluctuation

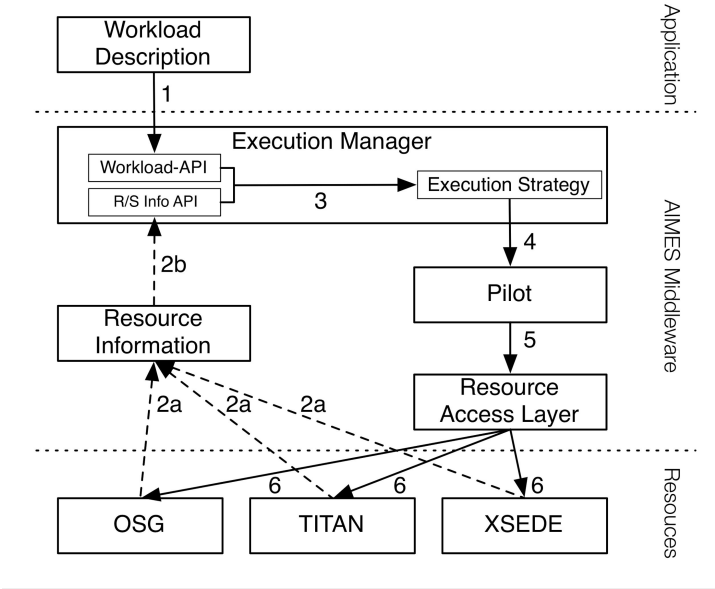
- **Fluctuations** in the task execution time (Tx) of concurrent tasks. Should have same Tx!
  - (Top) Molecular Dynamics (Gromacs)
  - (Middle) Athena-MP (ATLAS).
- Multiple contributing factors:
  - I/O and Lustre
  - Other shared resource contention
- Controlled experiments to isolate factors:
  - Smaller fluctuations when not using Lustre (middle). More than I/O!
  - Repeat experiment on same nodes (bottom): for some workflows, **when** is more important than **where** executed





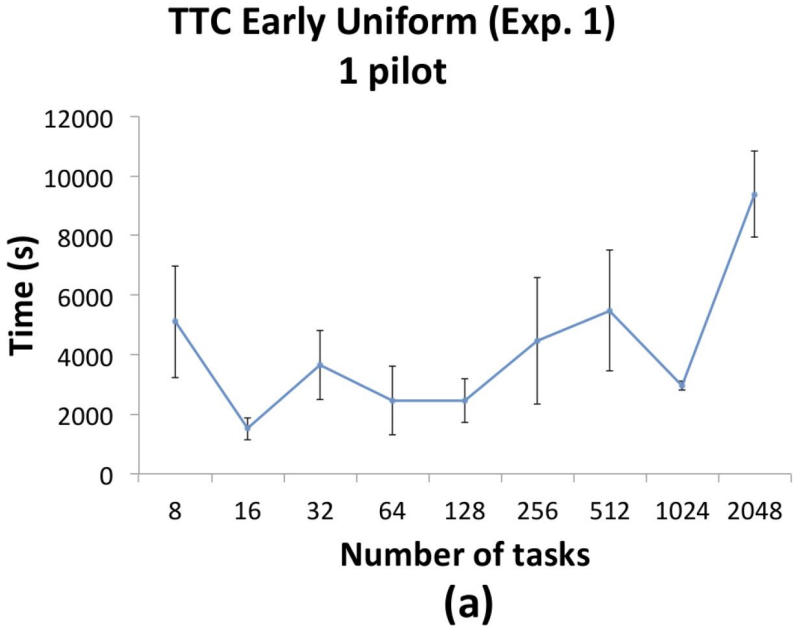
AIMES Execution Model

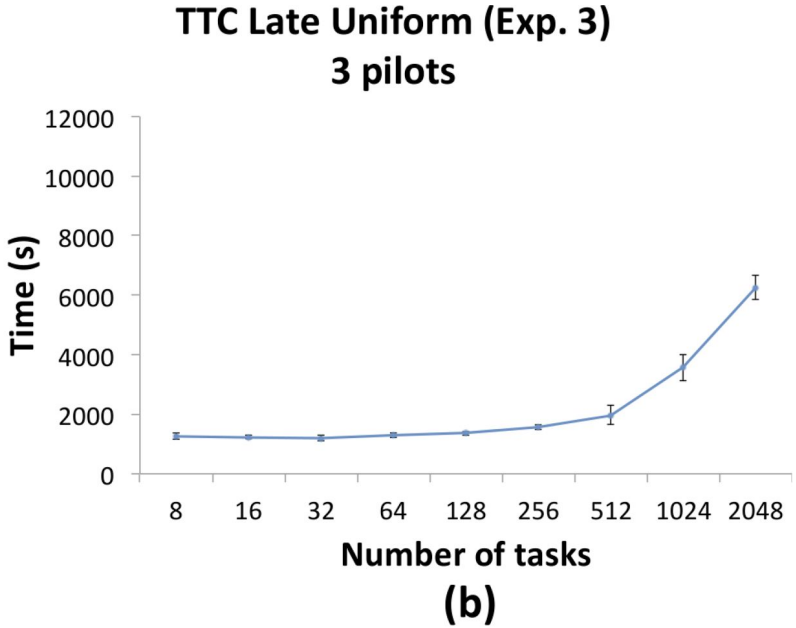
- **AIMES Execution Model** for how to map workloads onto dynamically varying heterogeneous resources, **independent** of type of infrastructural dynamism and heterogeneity.
- Decouple resource acquisition from workload assignment (**late** binding)
  - Requires **dynamic integration** of workload and resource information
  - Beyond multi-level scheduling
- **Execution strategy**: Temporally ordered set of decisions that need to be made to execute a given workload.



Turilli et al, IPDPS'16

Dynamic Workload-Resource Assignment: XSEDE



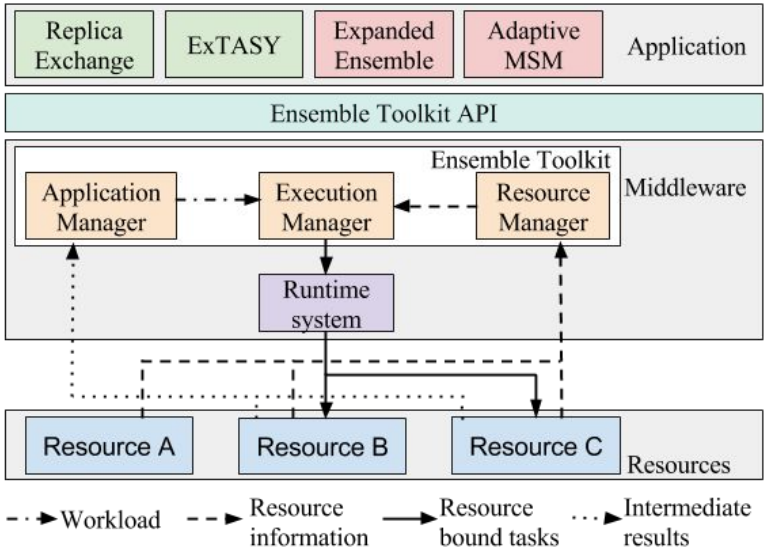


Turilli et al, IPDPS'16

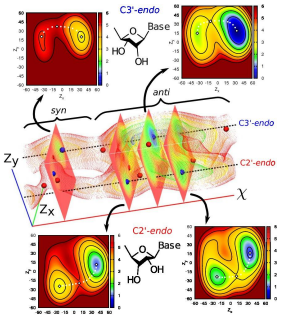
- $TTC = T_x + T_q$
- **Constrained to use same T_A and total number of cores**

AIMES Execution Model

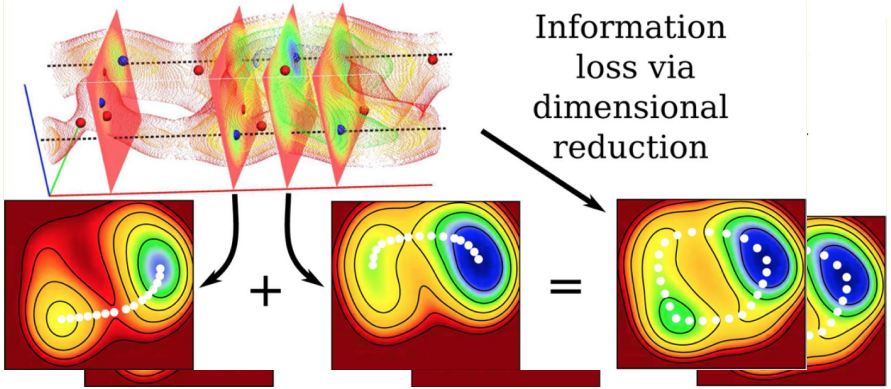
- **AIMES Execution Model** for how to map workloads onto dynamically varying heterogeneous resources, **independent** of type of infrastructural dynamism and heterogeneity.
- **Execution strategy**: Temporally ordered set of decisions that need to be made to execute a given workload.
- Generalize AIMES Execution model of late binding and dynamic integration of information, for “better” (?) and general mapping of workloads to HPC resources.



Ensemble Toolkit (EnTK), VB et al, ICPP’16

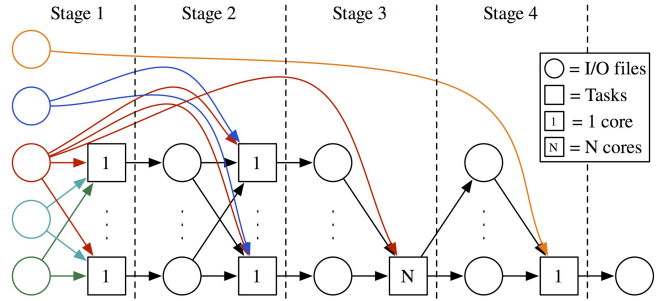


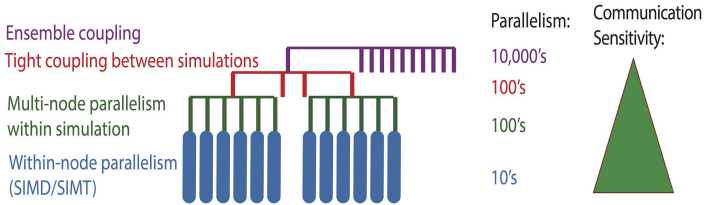
Adaptive Execution of (Ensemble) Workflows



Adaptive Execution

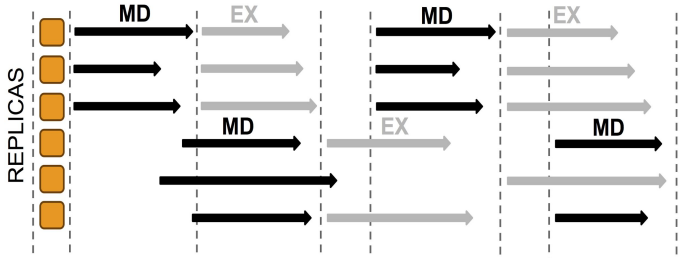
- **Adaptive Execution:** Changes made to the task-graph during runtime on the basis of results generated by executed sections of the task-graph.
  - Cannot be fully enumerated a priori.
- **Task:** scale and granularity
  - Each task is an independent simulation
  - Task often interact (not a “bag-of-tasks”); degrees and levels of coupling
- Adaptive Execution: Types
  - Task parameter(s), order, …
  - Task count, iteration count, ...

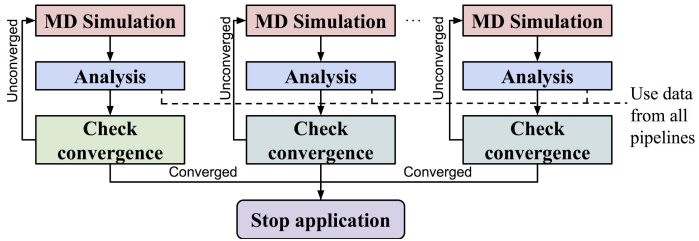


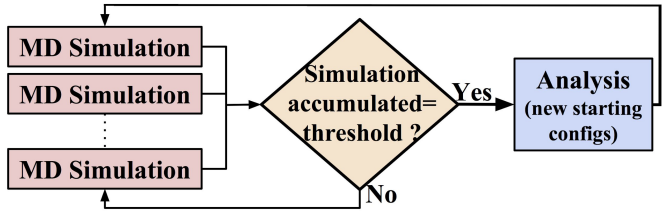


Adaptive Execution of Biomolecular Workflows

- Many biomolecular **sampling algorithms** formulated as adaptive algorithms/methods:
  - Replica-exchange
  - Expanded Ensemble
  - Markov State Models
  - …
- Adaptive algorithms/methods:
  - Scalable utilization of heterogeneous resources; address runtime fluctuations
  - Dynamic resource utilization a **necessary** condition for adaptive execution

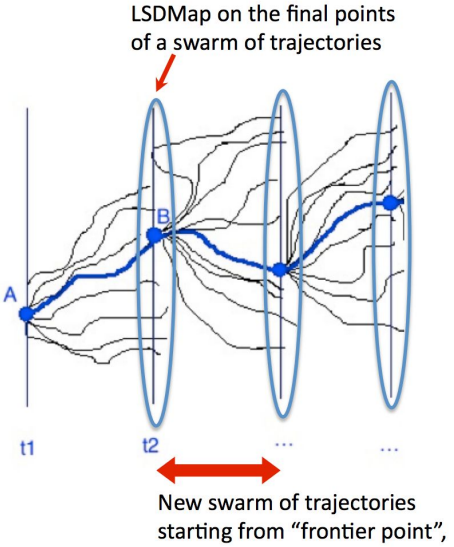


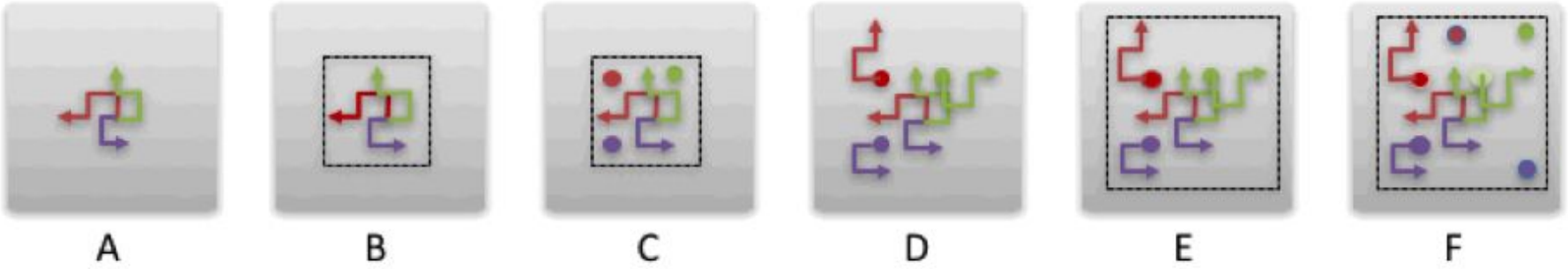




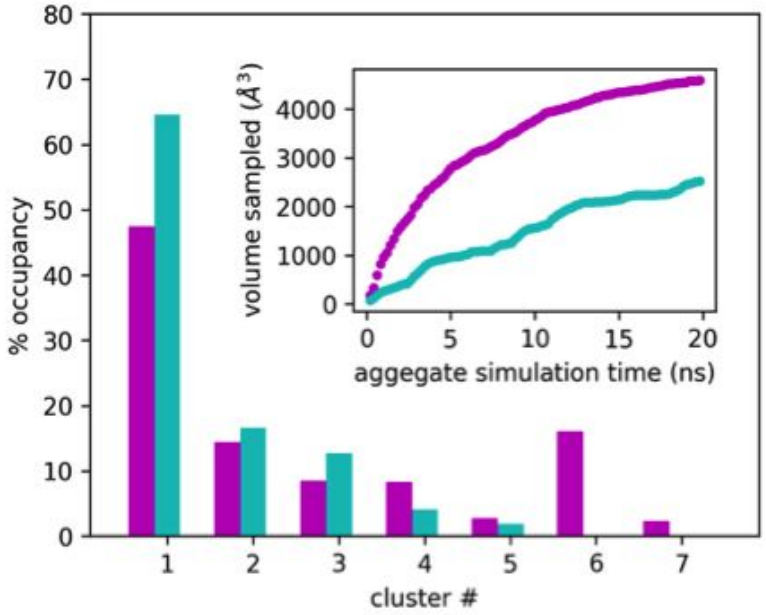
Advanced Sampling: CoCo

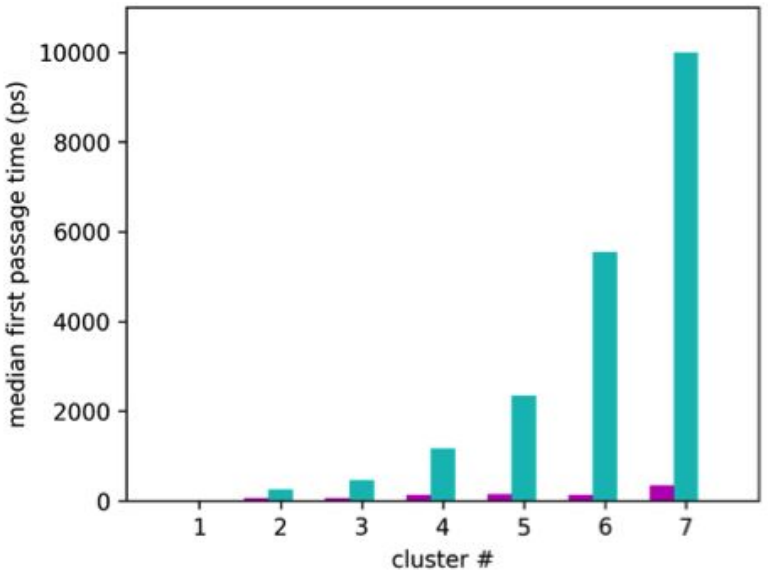
- **Better Sampling:** Drive systems towards unexplored regions, prevent waste time sampling behaviour already observed
  - DM-d-MD (LSDMap)
- Collective Coordinates (**CoCo**):
  - PCA-based unsupervised learning using reduced dimensionality





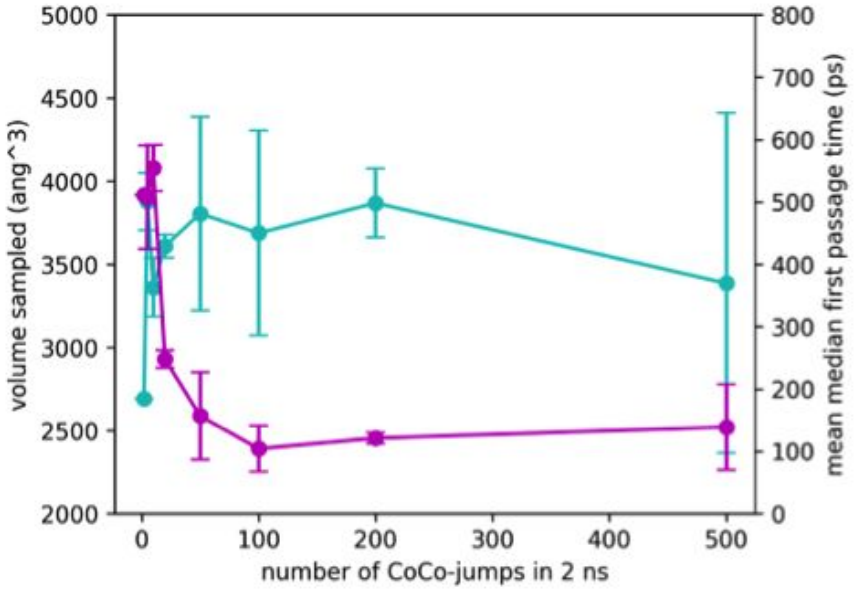
Sampling Quality: Classic versus Adaptive MD

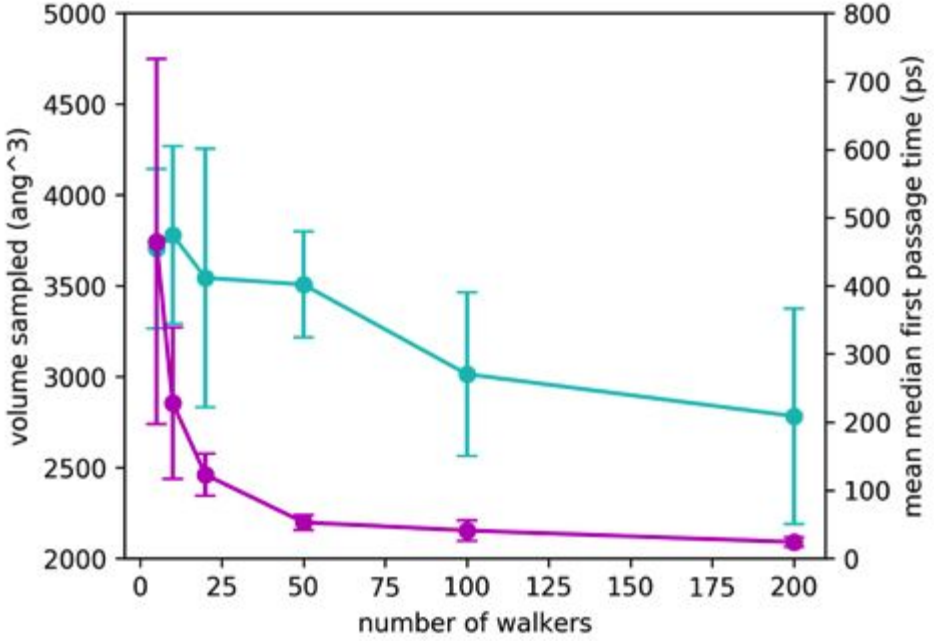




- "Classic" Ensemble MD (Cyan) versus Adaptive MD (Magenta)
- **All points use same T_A by experiment design (though different N_E and T_E)**
  - N_E = 10, t_E= 2 ns

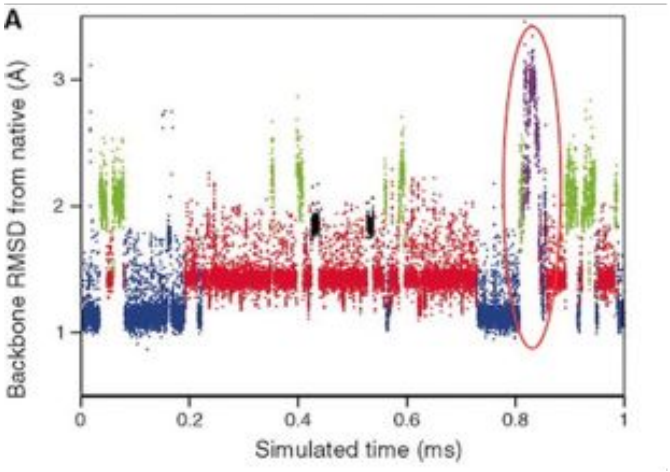
Sampling Quality: Adaptive MD Configuration

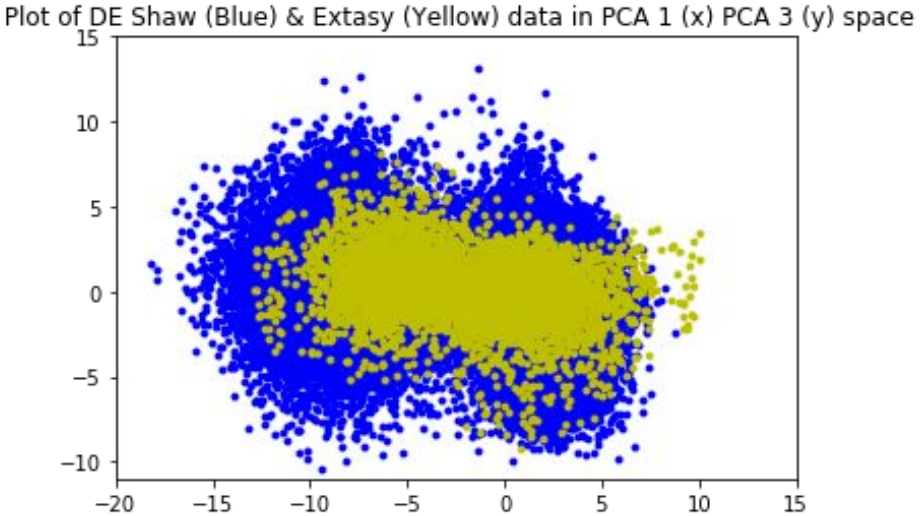




- **All points use same T_A by experiment design (though different N_E and T_E)**

Sampling Quality: Single MD vs Adaptive Execution





1000x ?

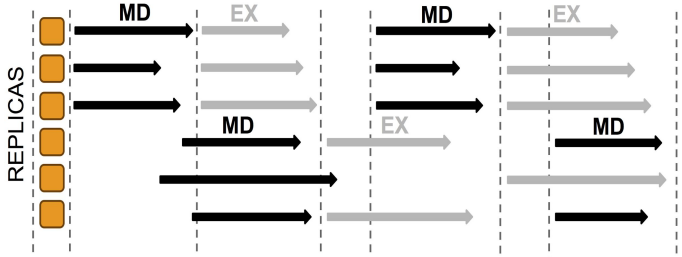
Building Blocks for Adaptive Workflows

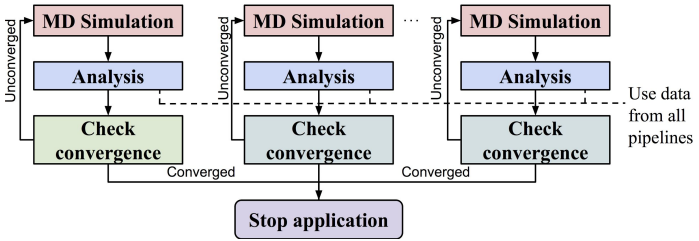
A Fresh Perspective on Workflow Systems?

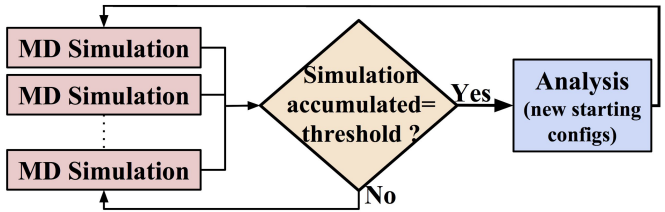
- Initially workflow systems were monolithic with “end-to-end” capabilities
  - Workflow systems were developed to support “big science” projects when software infrastructure was “fragile”, unreliable, missing services
  - Run many times, or many users: amortisation of development overhead
- Workflows aren’t what they used to be!
  - Diverse design points: Automation, scale and sophistication
  - Pervasive and not confined to “big science”. Unlikely one size fits all.
- Need for agile, experimental and often unique workflows
  - The workflow is the (end-user) algorithmic innovation
  - End-users often developers (not of performance critical components)

Requirements for Adaptive Execution

- Fundamental support for ensembles and adaptive execution of ensembles
  - Wide range and types of adaptivity
  - Independent of specific MD engine
- Separate resource management details from execution patterns and algorithmic adaptivity
  - Dynamic resource utilization and late binding of workload
- Rather than adapt to a given workflow system, ask what requirements does this impose on workflow middleware systems?







Building Blocks Approach

Principled approach to the architectural design of middleware systems

- Software component agnostic towards architectural, coordination, and communication patterns, implemented in an arbitrary programming language, and exposed via an API.
  - **Self-contained:** Fully implements a well-defined set of functionalities and design does not depend on other building blocks.
  - **Composable:** Caller can compose functionalities from independent blocks.
  - **Interoperable:** Usable in diverse system architectures without semantic modification. Provides functionalities to every well-formed call.
  - **Extensible:** Building block functionality and entities can be extended
- Beyond modularity: a building block behaves like a black box where no assumptions are made on the design and implementation of the caller.

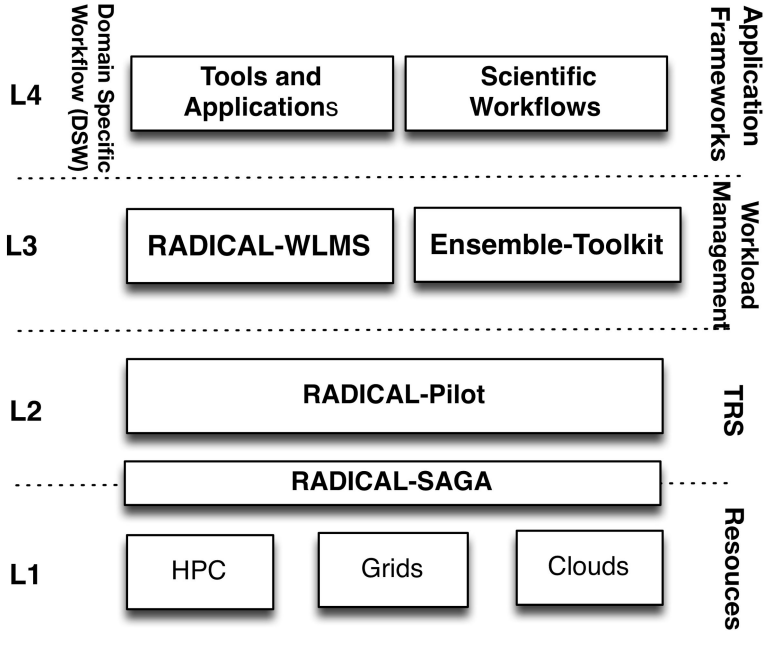
Building blocks approach towards domain specific workflow systems?
Jha and Turilli, https://arxiv.org/pdf/1609.03484.pdf

RADICAL-Cybertools: A Building Blocks Approach

- Four Layers:
  - L4: Adaptive Algorithm/Method
  - L3: Workload Management (WMS)
  - L2: Task Run-time (TRS)
  - L1: Resource Access Layer

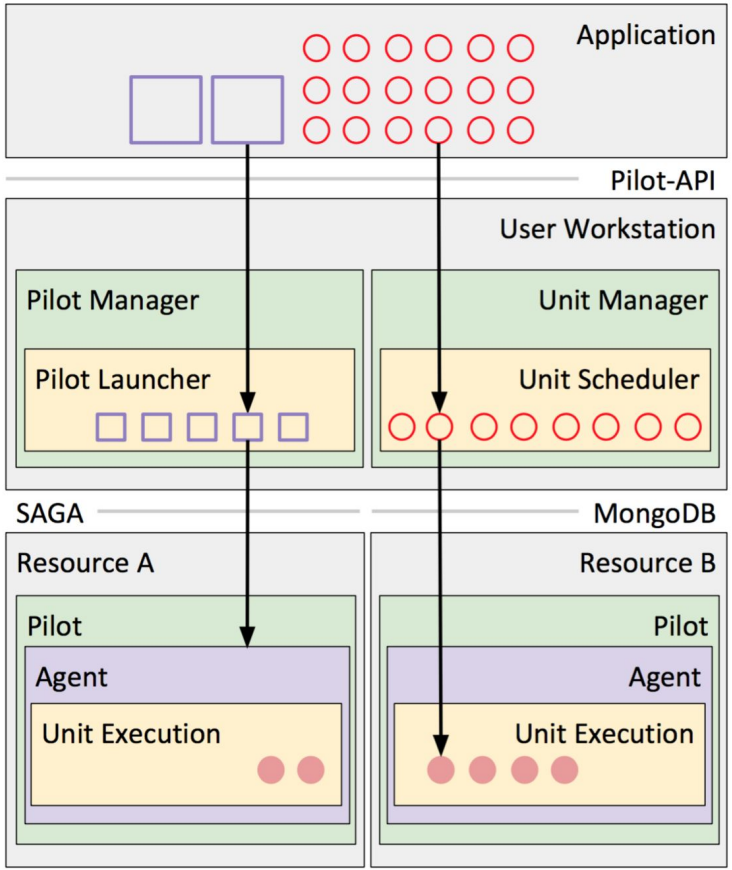
RADICAL-Cybertools: A Building Blocks Approach

- Four Layers:
  - L4: Adaptive Algorithm/Method
  - L3: Workload Management (WMS)
  - L2: Task Run-time (TRS)
  - L1: Resource Access Layer
- Abstractions & Building Blocks:
  - L1: RADICAL-SAGA Distributed job submission & standard interface
  - L2: **RADICAL-Pilot (RP)** Abstraction for Resource Management
  - L3: RADICAL-WLMS, Ensemble Toolkit
- Cross-layer: RADICAL-Analytics
  - Promote reproducibility & consistency

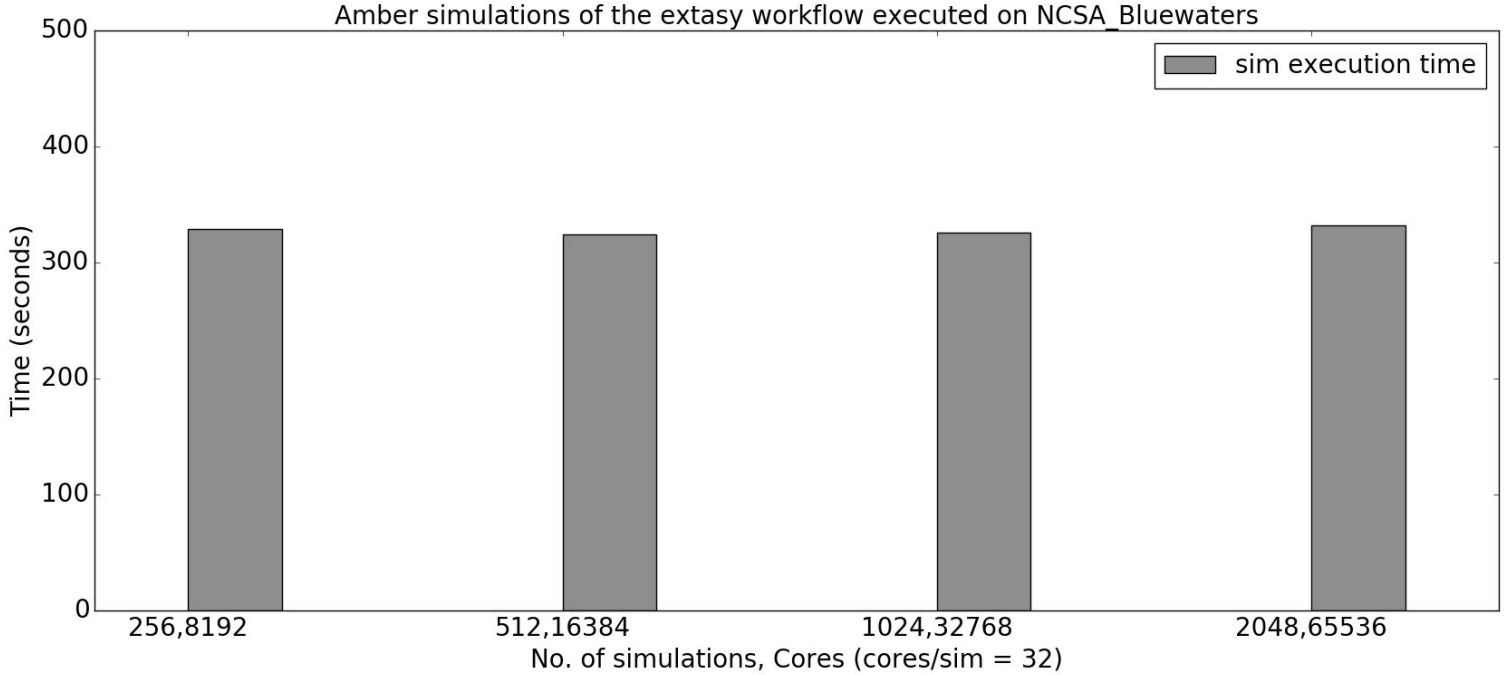


L2: Pilot-Abstraction (P* Model)

- *".. a scheduling overlay which generalizes the reoccurring concept of utilizing a placeholder as a container for compute tasks"*
- Decouples workload from resource management
- Enables the fine-grained spatio-temporal control of resources
- Build higher-level frameworks without explicit resource management
- Provides building block for late-binding of workloads on HPC



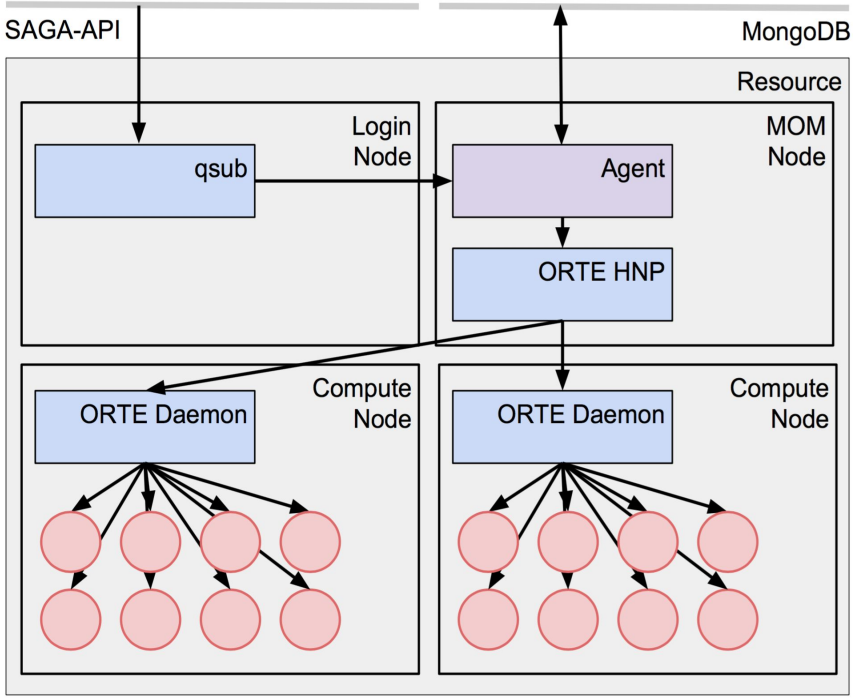
Performance



- RADICAL-Pilot design and implementation supports the efficient launch and management of O(2K) tasks (of 300 seconds) over 64K cores.

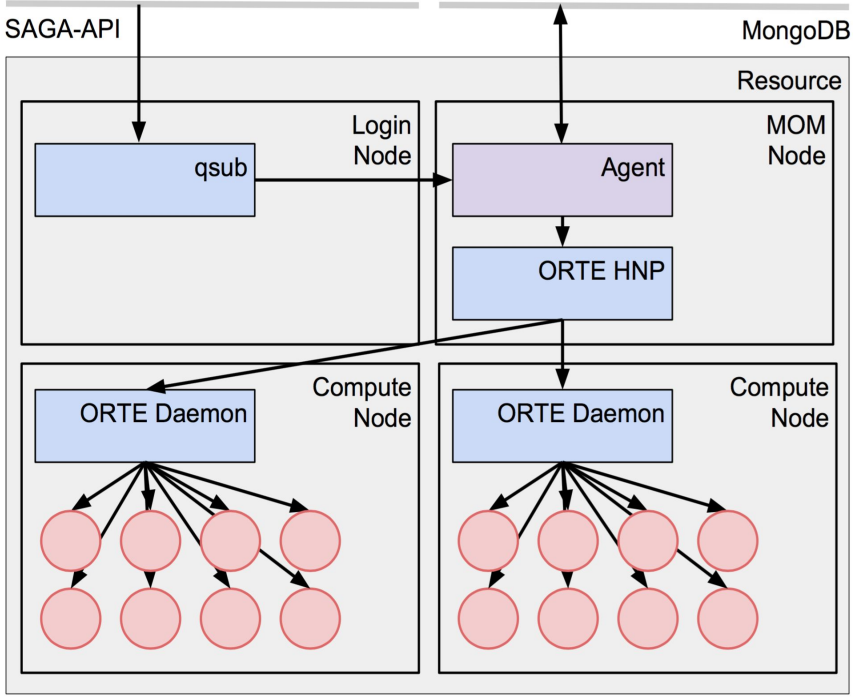
Challenges of O(100K+) Concurrent Tasks

- Agent communication layer (ZMQ) has limited throughput
  - Separate message channels
  - Bulk messages, operations
  - Better scheduling algorithms
  - State information management:



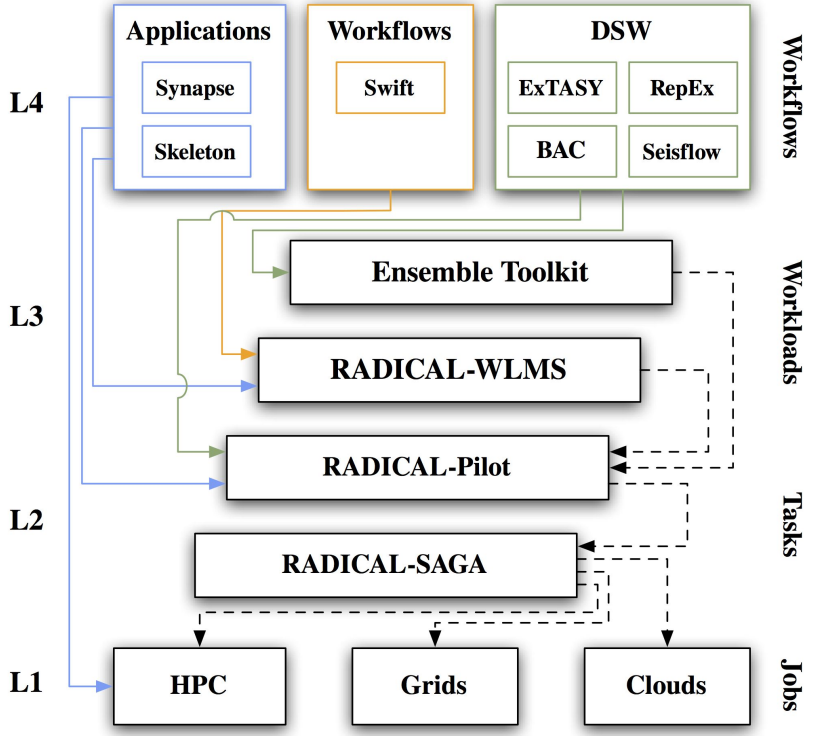
Challenges of O(100K+) Concurrent Tasks

- Agent communication layer (ZMQ) has limited throughput
  - Separate message channels
  - Bulk messages, operations
  - Better scheduling algorithms
  - State information management:
- Interface with ORTE
  - Isolated layer used by Open MPI to coordinate task layout
  - No ALPS concurrency limits
- Supports multiple tasks per node
  - Uses library calls instead of `orterun` processes



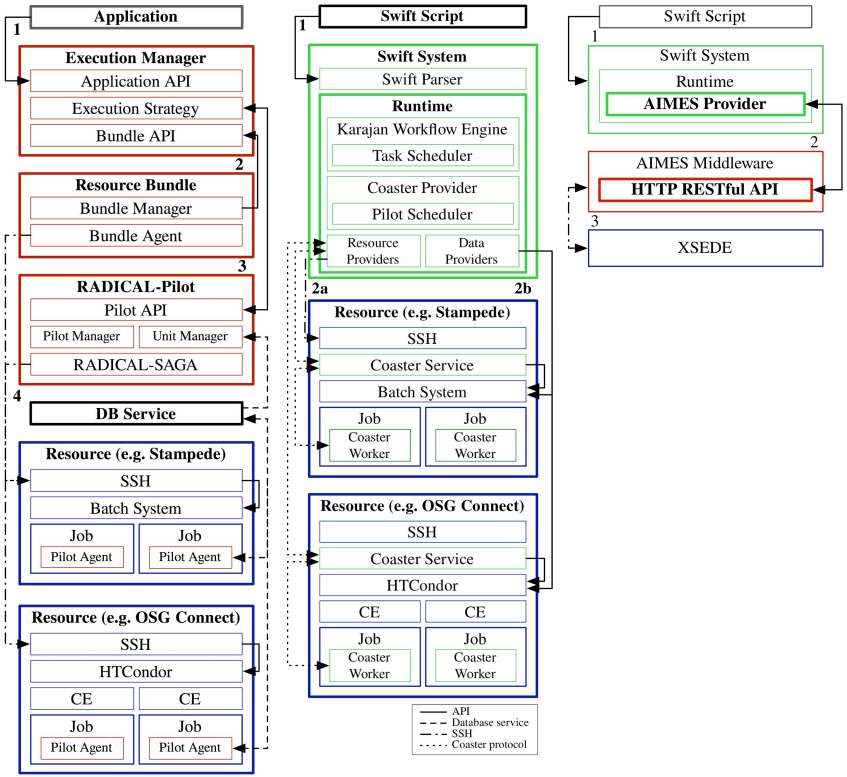
RADICAL-Cybertools: Building Blocks for Workflows

- A “laboratory” while supporting production grade workflows **and** workflow tools.
  - **Consistent with HPC & scale**
- **Integrate with existing tools:**
  - Swift, Fireworks, PanDA, Binding Affinity Calculator (BAC)
  - Distinct points of integration, vertical integration and horizontal extensibility
  - Need “faster” start, “scalable” (more tasks) and “better” (resource utilization)
- **Novel tools and libraries:**
  - **ExTASY,** Replica-Exchange, Seisflow



RADICAL-Cybertools: Building Blocks for Workflows

- A “laboratory” while supporting production grade workflows **and** workflow tools.
  - Consistent with performance & scale
- **Integrate with existing tools:**
  - **Swift,** Fireworks, PanDA, Binding Affinity Calculator (BAC)
  - Distinct points of integration, vertical integration and horizontal extensibility
  - Need “faster” start, “scalable” (more tasks) and “better” (resource utilization)
- **Novel tools and libraries:**
  - **ExTASY,** Replica-Exchange, Seisflow

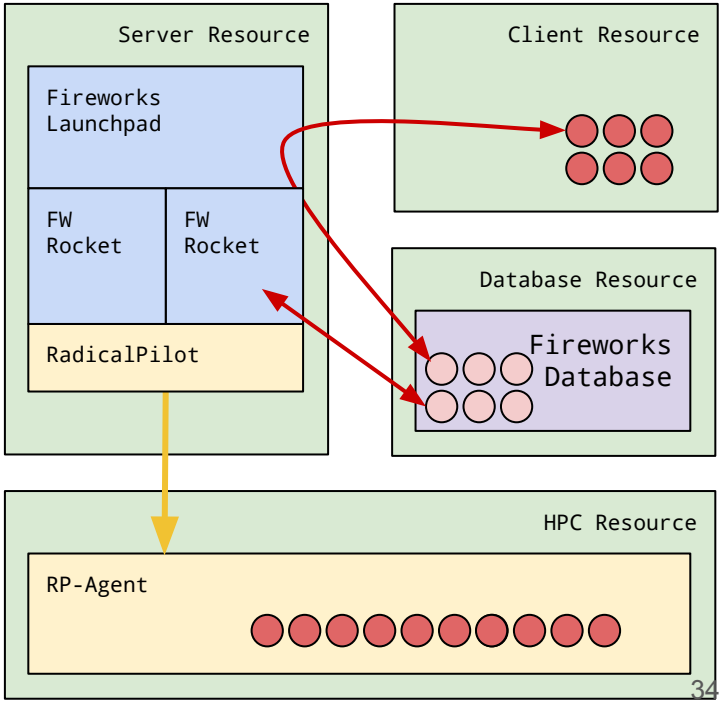


RADICAL-Cybertools: Building Blocks for Workflows

- A “laboratory” while supporting production grade workflows **and** workflow tools.
  - Consistent with performance & scale
- **Integrate with existing tools:**
  - Swift, **Fireworks**, PanDA, Binding Affinity Calculator (BAC)
  - Distinct points of integration, vertical integration and horizontal extensibility
  - Need “faster” start, “scalable” (more tasks) and “better” (resource utilization)
- **Novel tools and libraries:**
  - **ExTASY,** Replica-Exchange, Seisflow

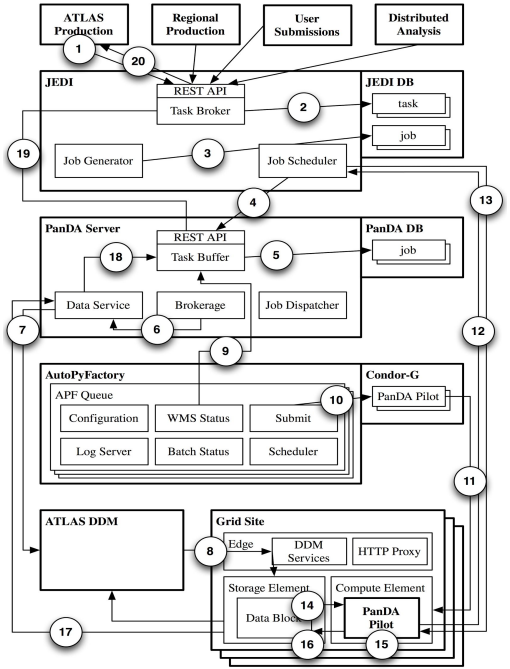
Fireworks + RP:

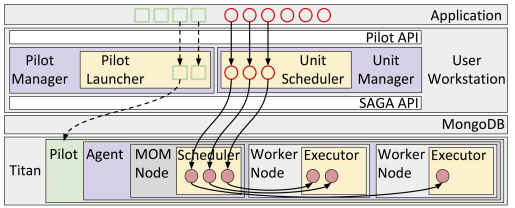
- Rockets start RP pilots on HPC hosts
- Rockets push tasks to RP for execution



RADICAL-Cybertools: Building Blocks for Workflows

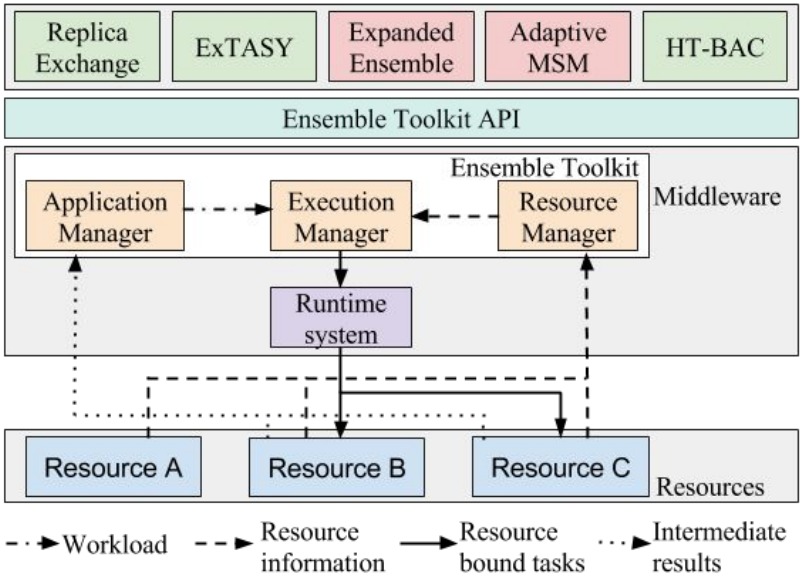
- A “laboratory” while supporting production grade workflows **and** workflow tools.
  - Consistent with performance & scale
- **Integrate with existing tools:**
  - Swift, Fireworks, **PanDA,** Binding Affinity Calculator (BAC)
  - Distinct points of integration, vertical integration and horizontal extensibility
  - Need “faster” start, “scalable” (more tasks) and “better” (resource utilization)
- **Novel tools and libraries:**
  - **ExTASY,** Replica-Exchange, Seisflow



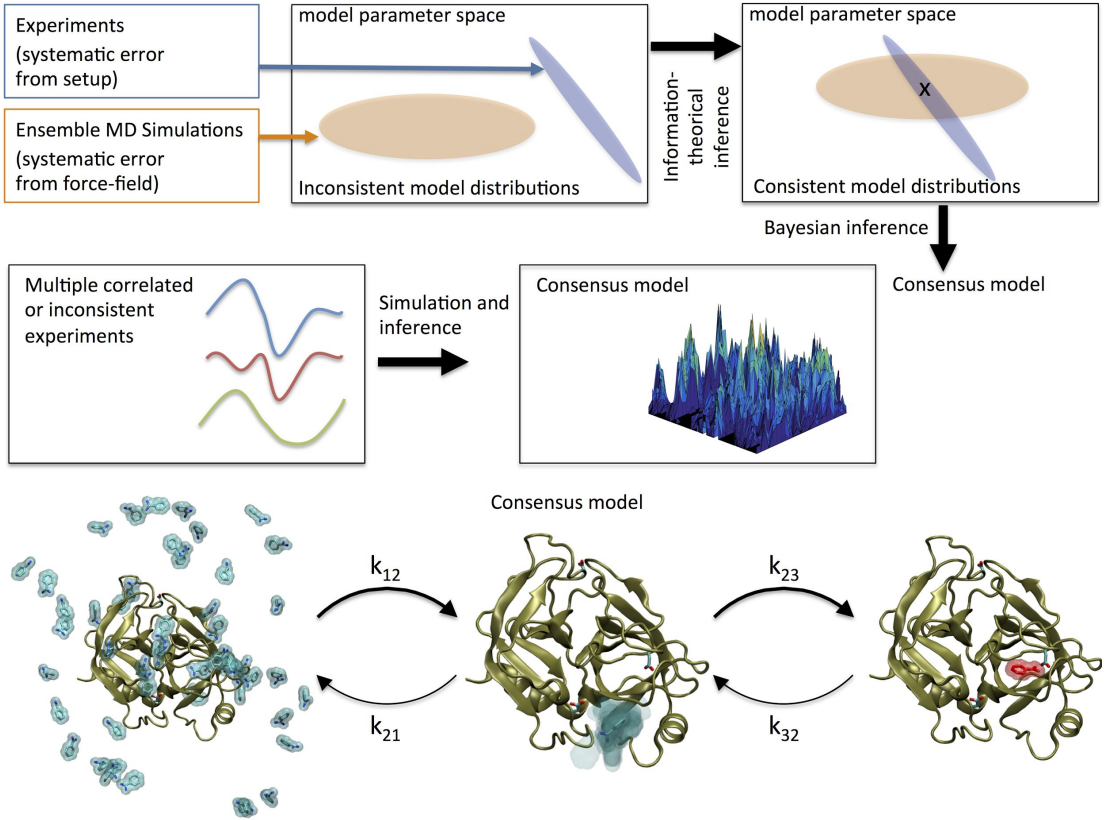


RADICAL-Cybertools: Building Blocks for Workflows

- A “laboratory” while supporting production grade workflows **and** workflow tools.
  - Consistent with performance & scale
- **Integrate with existing tools:**
  - Swift, Fireworks, PanDA, **Binding Affinity Calculator (BAC)**
  - Distinct points of integration, vertical integration and horizontal extensibility
  - Need “faster” start, “scalable” (more tasks) and “better” (resource utilization)
- **Novel tools and libraries:**
  - **ExTASY,** Replica-Exchange, Seisflow

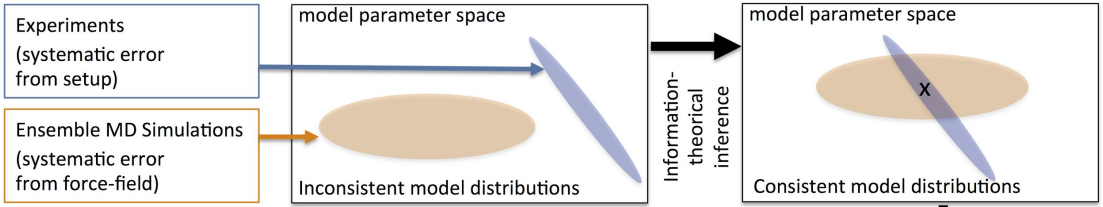


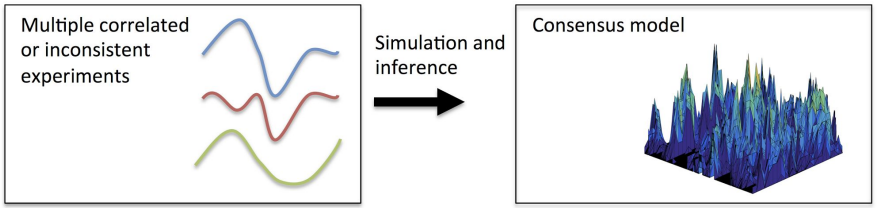
Next-generation Adaptive Workflows

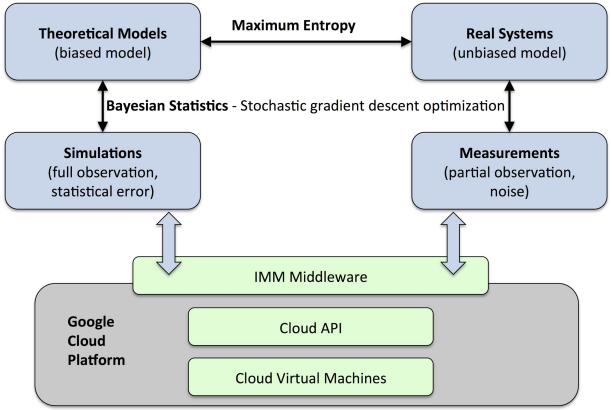


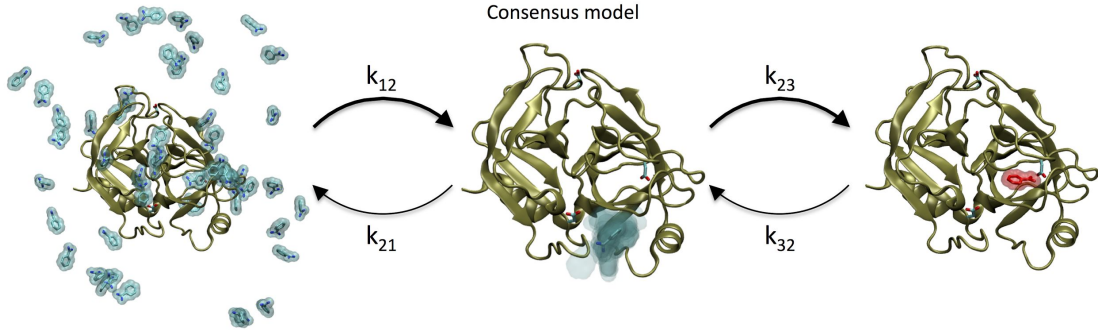
- INtegrated and Scalable PredIction of REsistance (INSPIRE)
- Middleware for Inference of Molecular Models (Kasson)

Next-generation Adaptive Workflows



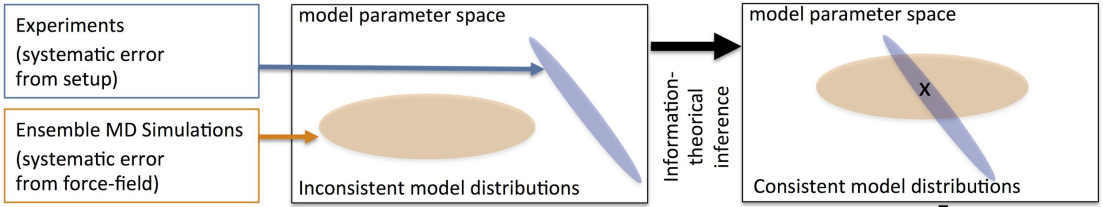


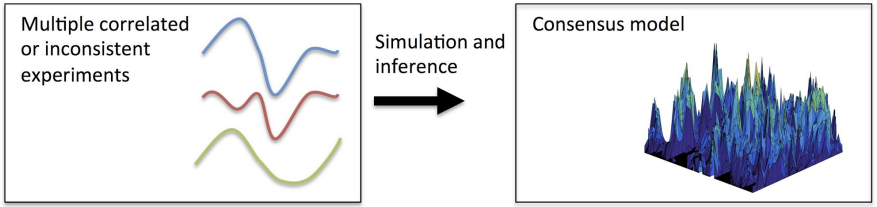


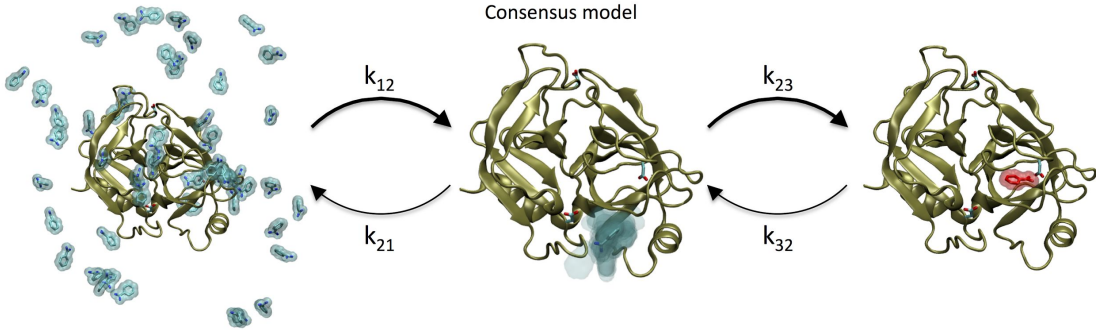


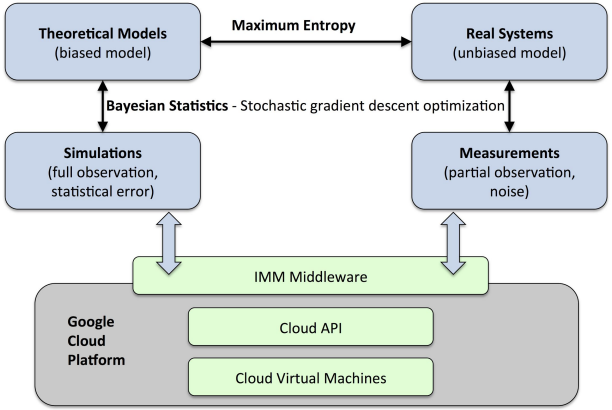
- INtegrated and Scalable PredIction of REsistance (INSPIRE)
- Middleware for Inference of Molecular Models (Kasson)

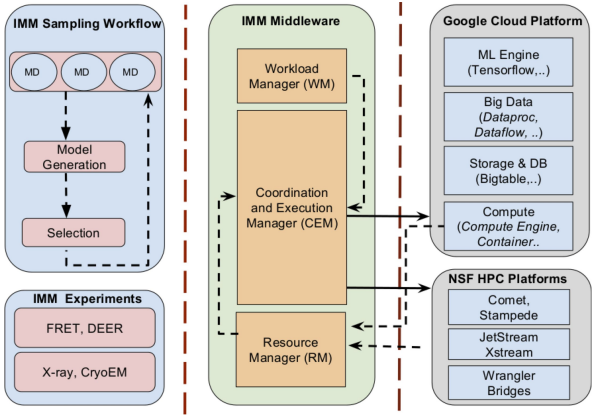
Next generation Adaptive Workflows











Summary

- **Adaptive execution of biomolecular ensembles are useful & important**
  - Demonstrably better than “classic” simulations, but requires care
  - Challenging due to runtime fluctuations
- **Proliferation of middleware systems for “workflows” unsustainable**
  - Distinctions are by-products of specific cyberinfrastructure implementations.
  - Substitute discussions of software with **abstractions & execution models**
- **Building blocks approach to adaptive workflows**
  - Focussed, principled design and development of middleware systems
- **RADICAL-Cybertools: Engineered implementation of building blocks**
  - Multiple adaptive algorithms (replica-exchange, MSM, EE)
  - Prototype and production of enhanced functionality: Swift, Fireworks, and PANDA (ATLAS), HT-BAC

Thank you!

Acknowledgements

- Peter Kasson and Michael Shirts (Adaptive Algorithms/Methods)
- Cecilia Clementi and Charlie Laughton (ExTASY)
- ATLAS Team
- Geoffrey Fox (SPIDAL)
- Peter Coveney (HT-BAC)
- Jon Weissman (AIMES)
- G Cervone, M Mann (PSU)
- M Lefebvre, J Tromp (Princeton)

Research in **A**dvanced **Di**stributed **C**yberinfrastructure and **A**pplications **L**aboratory (RADICAL)

http://radical.rutgers.edu/ - http://radical-cybertools.github.io/

Support:

- NSF and DOE
- IACS at Stony Brook
- CSI at BNL

- Contrary to traditional experiments, where resource utilization is static, can we identify a dynamic execution strategy (**ES**) (= size and time distribution of jobs requested) so as to maximize probability of resource utilization?
- Estimate the utilization of an allocation time ($\mathbf{T_A}$ **core-hours**) as a function of time.