Building Blocks for Open Workflow Systems: The RADICAL-Cybertools Experience

Shantenu Jha
Brookhaven National Laboratory and Rutgers University

http://radical.rutgers.edu

Outline

- **A biased perspective of Workflows and Workflow Systems**
  - Diversity of workflow systems; proliferation needs to be managed
  - Understand needs of workflow system development

- **RADICAL-Cybertools: Building Blocks for Middleware for Workflow System**
  - Example of common components: Pilot-Jobs and Ensemble ToolKit (EnTK)
  - Lessons from Adaptive Ensemble-based Applications
  - https://arxiv.org/abs/1903.10057

- **Facilities perspective: Requirements & Challenges for Workflow Systems**
  - Distinguish “what” versus “how”, expressing “workflow” vs “executing”
  - Importance of Workflows at Extreme Scale: Lessons from OpenHPC ?

A Historical Perspective on Workflows & Systems

- **1970s:** “Business Workflows”, early 1990s: Workflow Management Coalition
- **Late 1990’s** (Early 2000s):
  - Grid/Distributed workflows -- driven by LHC
  - HPC Workflows (ASCI Program)
- **2001:** MyGrid / MyExperiment emphasized provenance and reproducibility,
  - Advances in workflow sharing, e.g., Taverna (cross-disciplinary WMS)
  - Implementations rely upon changing technologies. Sustainability?
- **2014:** ASCR Workflow Modelling Program (Rich Carlson)
- **2019:** Approximately 240 computational & data analysis Workflow Systems
  - https://s.apache.org/ existing- workflow- systems
  - Most workflow users don’t use a “formal” WMS, but “roll their own”
  - Caveat: Diverse systems; complete - partial; extensible - standalone, …

A (Biased) Perspective on Workflows & Systems (2)

- Initially workflow **management** systems provided "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!
  - High-throughput computing is important, but many other features
  - Diverse design points: Automation, scale, sophistication, …
  - Need for agile, experimental and often unique workflows
  - Pervasive and not confined to "big science". Unlikely one size fits all.
  - **The workflow is the (end-user) algorithmic innovation**
  - End-users often developers (not of performance critical components)
  - ….

A (Biased) Perspective on Workflows & Systems (3)

- Neither is the Infrastructure what it used to be, either!
  - Python ecosystem, e.g., task distribution and coordination systems
  - Apache (big) data stack of analysis tools
  - Containers technologies
  - …
  - Infrastructure more reliable, better supported, consistently available
- Other reasons for “roll your own” workflow systems worth examining
  - Many are not HPC oriented --- which has unique needs & challenges,
  - More complex than infrastructure enhancements, and proverbial “last mile customization” of workflows

A (Biased) Perspective on Workflows & Systems (4)

- Proliferation cannot be reversed, it must be managed.
- Many questions / implications of proliferation:
  - How to implement workflow systems in an agile fashion to provide flexibility and sharing of capabilities while not constraining functionality, performance, or sustainability?
  - Given the rich ecosystem of capabilities, how can the barrier to the integration of workflow systems with these capabilities be lowered?
  - ….
- **Upshot:** Need a sustainable ecosystem of **both** existing and new software components from which tailored workflow systems can be composed
  - Realize agile development and composition of workflow systems that leverage rich ecosystem of existing capabilities
  - **Understand needs of workflow system development, not just workflows!**

Outline

- **A (biased) perspective of Workflows and Workflow Systems**
  - Diversity of workflow systems; proliferation needs to be managed
  - Understand needs of workflow system development

- **RADICAL-Cybertools: Building Blocks for Middleware for Workflow System**
  - Example of common components: Pilot-Jobs and Ensemble ToolKit (EnTK)
  - Lessons from Ensemble-based Applications

- **Facilities perspective: Requirements & Challenges for Workflow Systems**
  - Distinguish “what” versus “how”, expressing “workflow” vs “executing”
  - Importance of Workflows at Extreme Scale: Lessons from OpenHPC ?

Middleware Building Blocks for Workflow Systems

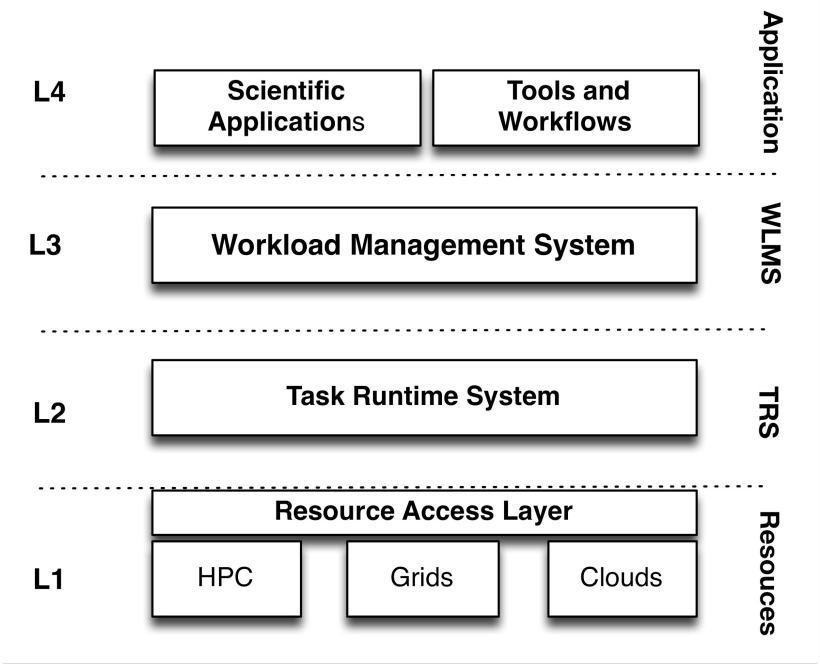
Building Block Approach: Principled approach to the architectural design of middleware systems and applies traditional notion of modularity at the systems level to enable composability among independent software systems

- The four design principles:
  - **Self-sufficient:** Implements set of functionalities; not depend on other blocks
  - **Composable:** Caller can compose functionalities from independent blocks.
  - **Interoperable:** Usable in diverse system without semantic modification
  - **Extensible:** Building block functionality and entities can be extended
- Work both individually, or integrated, or with third party software.

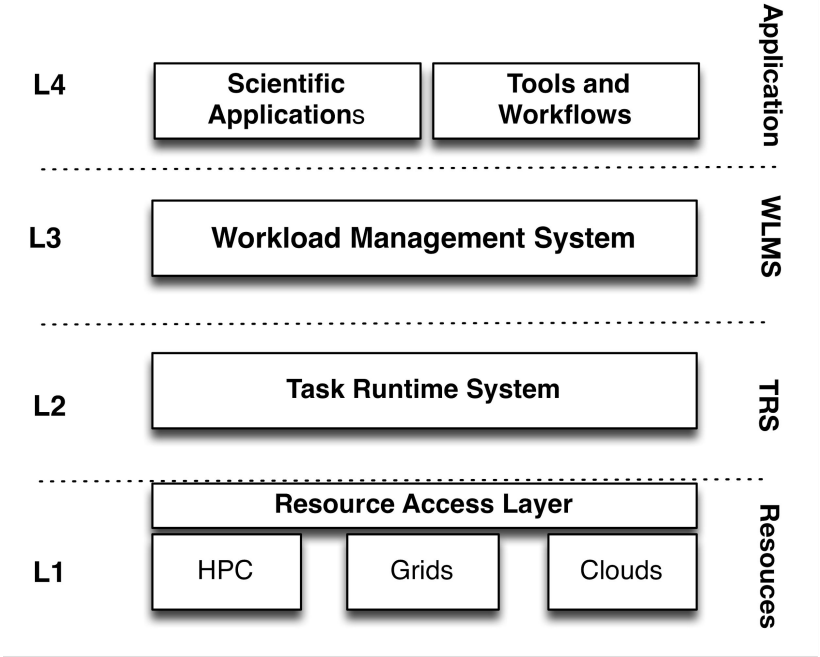
Building Blocks: Properties

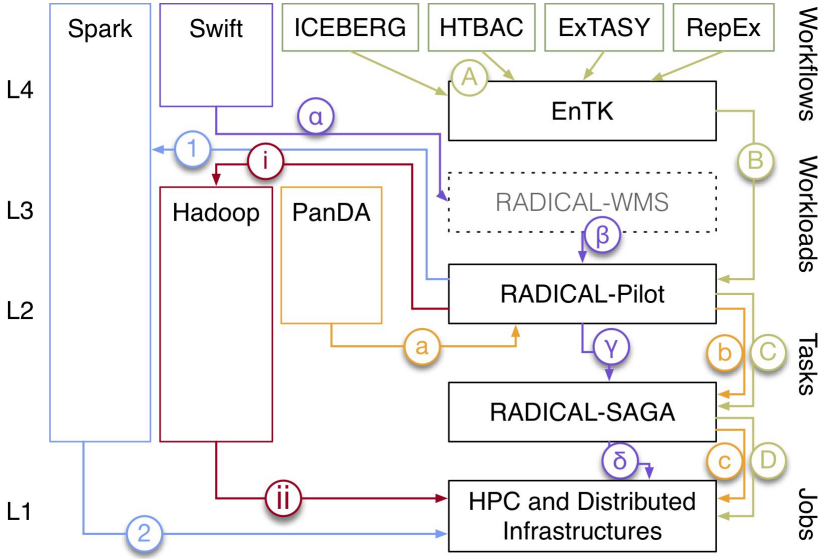
- A BB is an independent software system that is agnostic towards design, coordination, and communication **patterns**, and exposed via an API.
  - **Stronger** (stringent) property than modularity
- Architecturally building blocks require:
  - Stable interfaces permit distinction between computation and composition
  - Conversion layers enabling operation on multiple representation of the same entity
- Composability of Independent components remains challenging. Reasons: (i) lack of semantic uniformity across interface specification, (ii) error handling, (iii) I/O validation..
  - BB address some by specifying state, event, and error models for each block.

Developing Workflow Tools Using Building Blocks



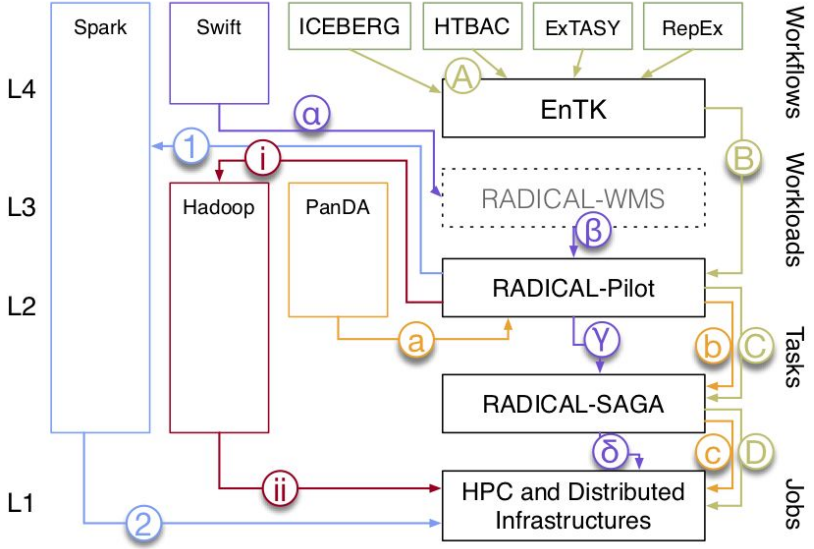
Developing Workflow Tools Using Building Blocks





RADICAL-Cybertools: Middleware Building Blocks

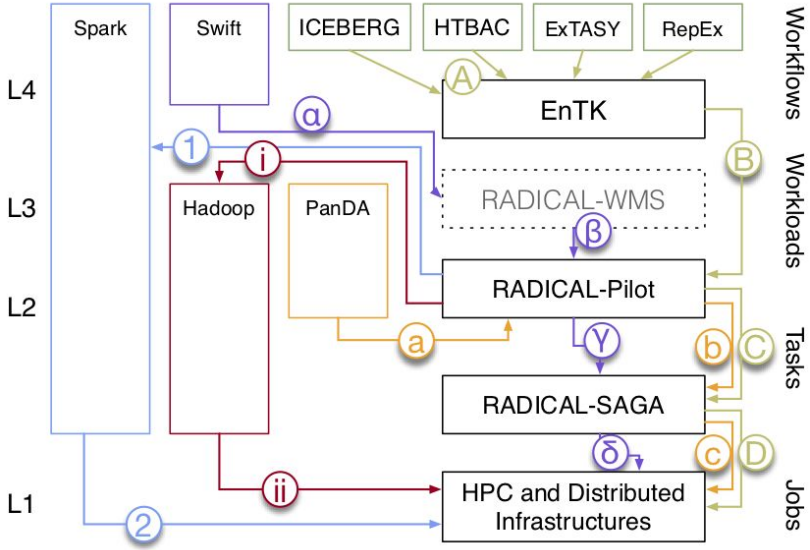
- A “laboratory” while supporting production grade workflows **and** workflow systems
- Stand alone, vertical and horizontal extensibility
- Integrate with existing tools:
  - PanDA, Swift, Fireworks, ...
  - Distinct points of integration
  - Need “faster” start, “scalable” (more tasks) and “better” (resource utilization)
- Novel tools and libraries:
  - ExTASY, RepEx, HTBAC, Seisflow,..



RADICAL-Cybertools as Building Blocks: Experience

Our preliminary experience with RADICAL-Cybertools (**RCT**) suggests:

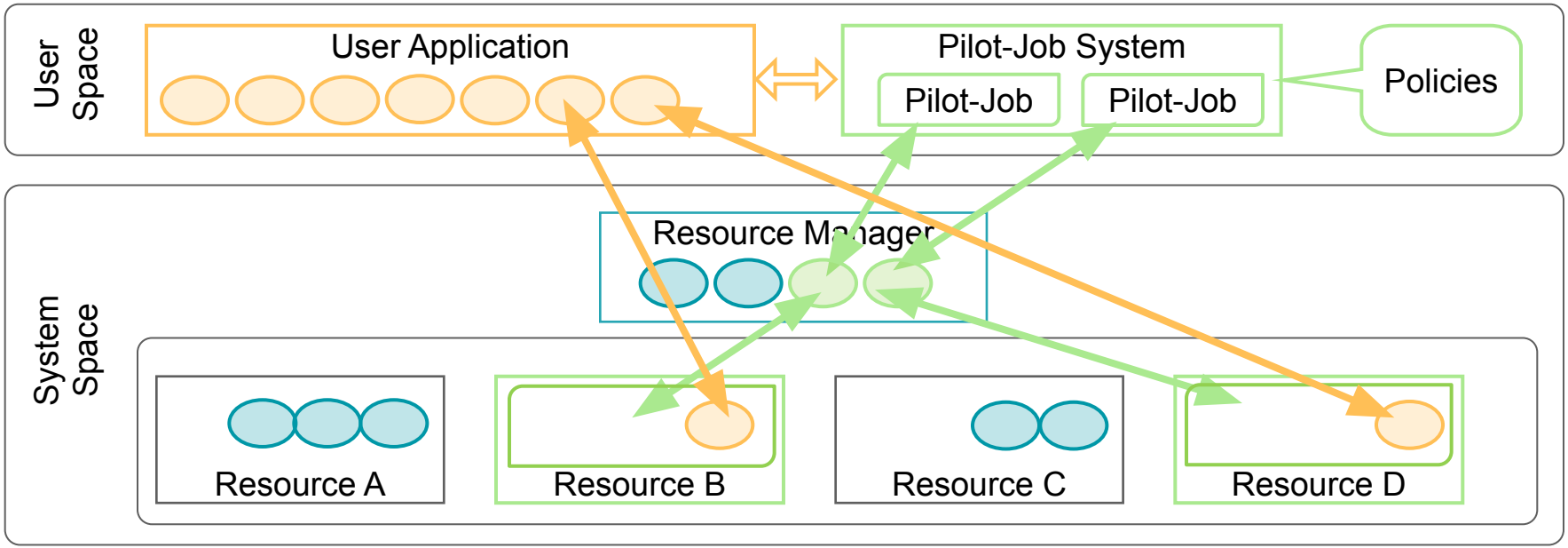
- Building Blocks are easily extensible & can be used for proverbial last mile customization:
  - EnTK is used to support > 6 Domain Specific Workflows
  - RepEx, ExTASY, ICEBERG, HTBAC, SCALE-MS, INSPIRE, ExaLearn
- Building Blocks are compatible with HPC challenges and requirements
  - RP used on 66% of Titan; now Summit
- RCT integrate with many other components -- some HPC, others non-HPC (ABDS)
- ….



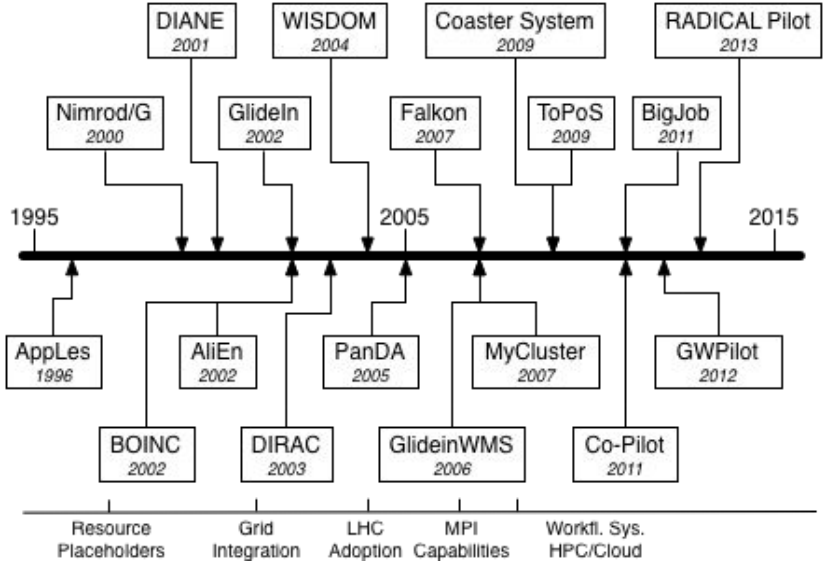
Pilot Abstraction: Schematic

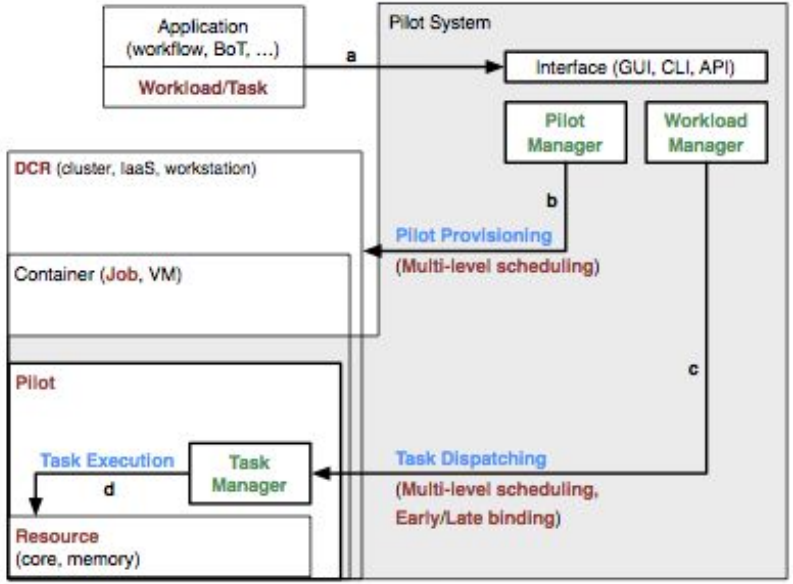
A system that allows application-level control of acquired resources via a scheduling overlay and a placeholder job.

- Enables the fine-grained “slicing and dicing” of resources
- Provides late-binding of workloads to resources



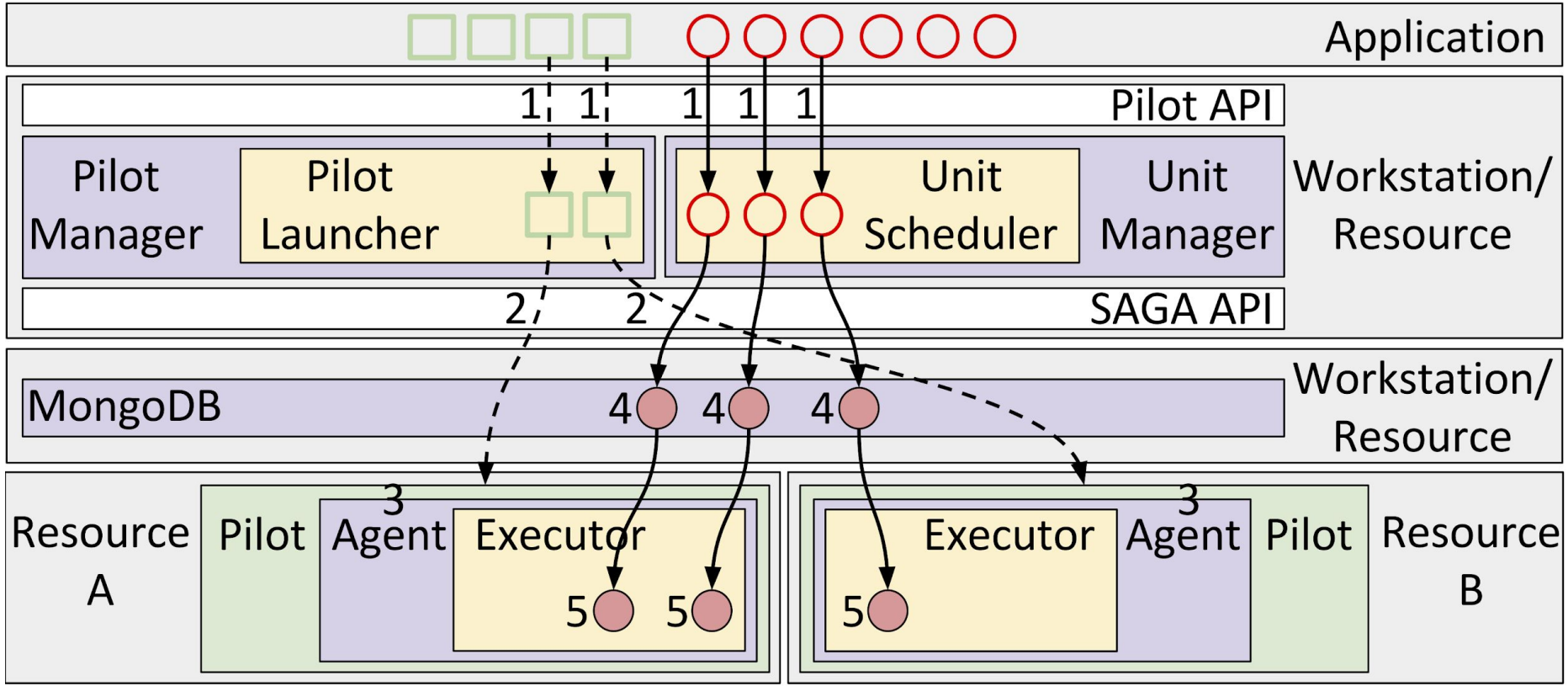
Pilot Jobs: Embarrassment of Many ?





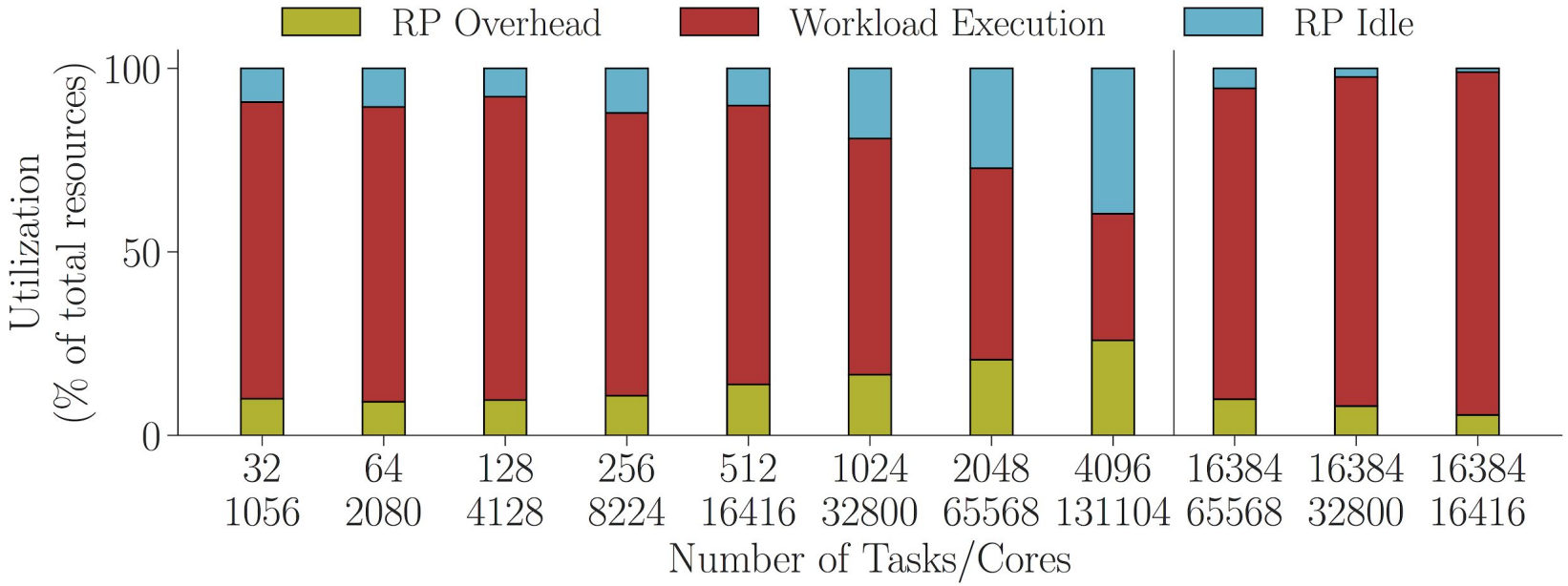
- "P*: A Model of Pilot-Abstractions", *8th IEEE International Conference on e-Science (2012)*
- *A Comprehensive Perspective on Pilot-Jobs* http://arxiv.org/abs/1508.04180 *(ACM Computing Surveys, 2018)*

RADICAL-Pilot: Execution Model



RADICAL-Pilot: Resource Utilization Performance (Titan)

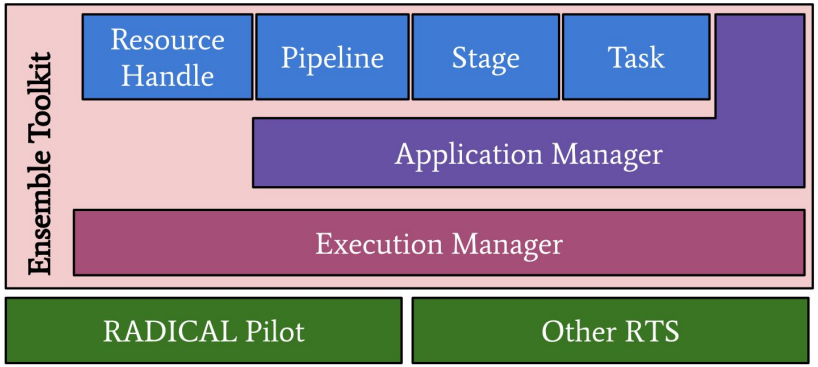
RADICAL-Pilot: Resource Utilization Performance (Titan)

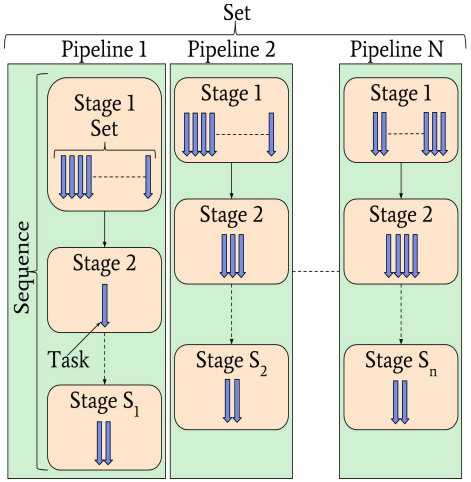


"... The PMIx community has committed to reducing or eliminating the time spent in these stages to achieve an overall goal of launching and connecting exascale applications in under 30 seconds. For purposes of tracking this goal, the community has adopted its baseline test as being the time ***required to start an application and complete MPI Init, with all processes having all required information to communicate at that point, using an application size of 50K nodes supporting up to 1M individual processes****…"* ***Castain et al, Parallel Computing 2018***

EnTK: Building Block for Ensemble based Applications

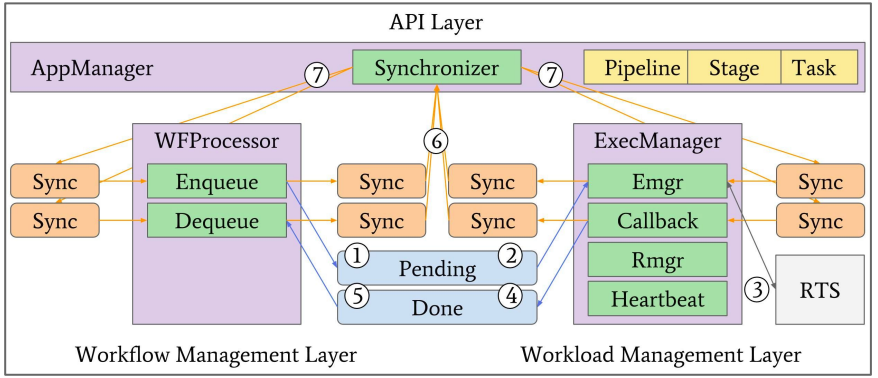
- **Ensemble-Toolkit (EnTK):** Toolkit to facilitate expression of ensemble based applications, manage complexity of resource acquisition and task execution.
- **Architecture:**
  - User facing components (blue)
  - Workflow management components (purple)
  - Workload management components (red) via runtime system (green)
- **PST Programming Model:**
  - **Task:** an abstraction of a computational process and associated execution information
  - **Stage:** a set of tasks without dependencies, which can be executed concurrently
  - **Pipelines:** a list of stages, where stage “i” can be executed after stage “i−1”

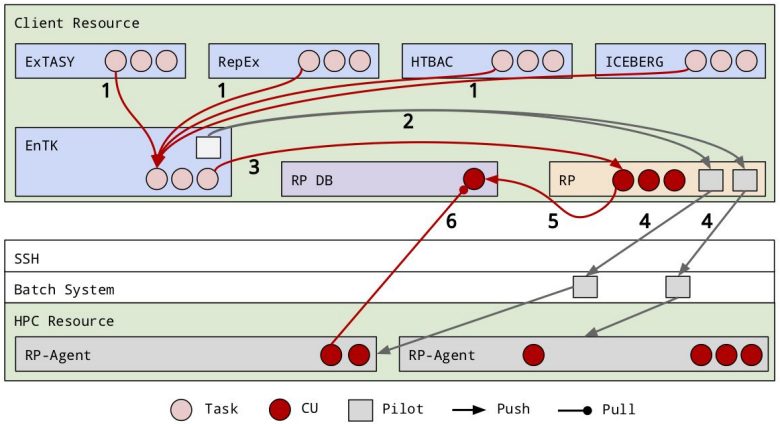




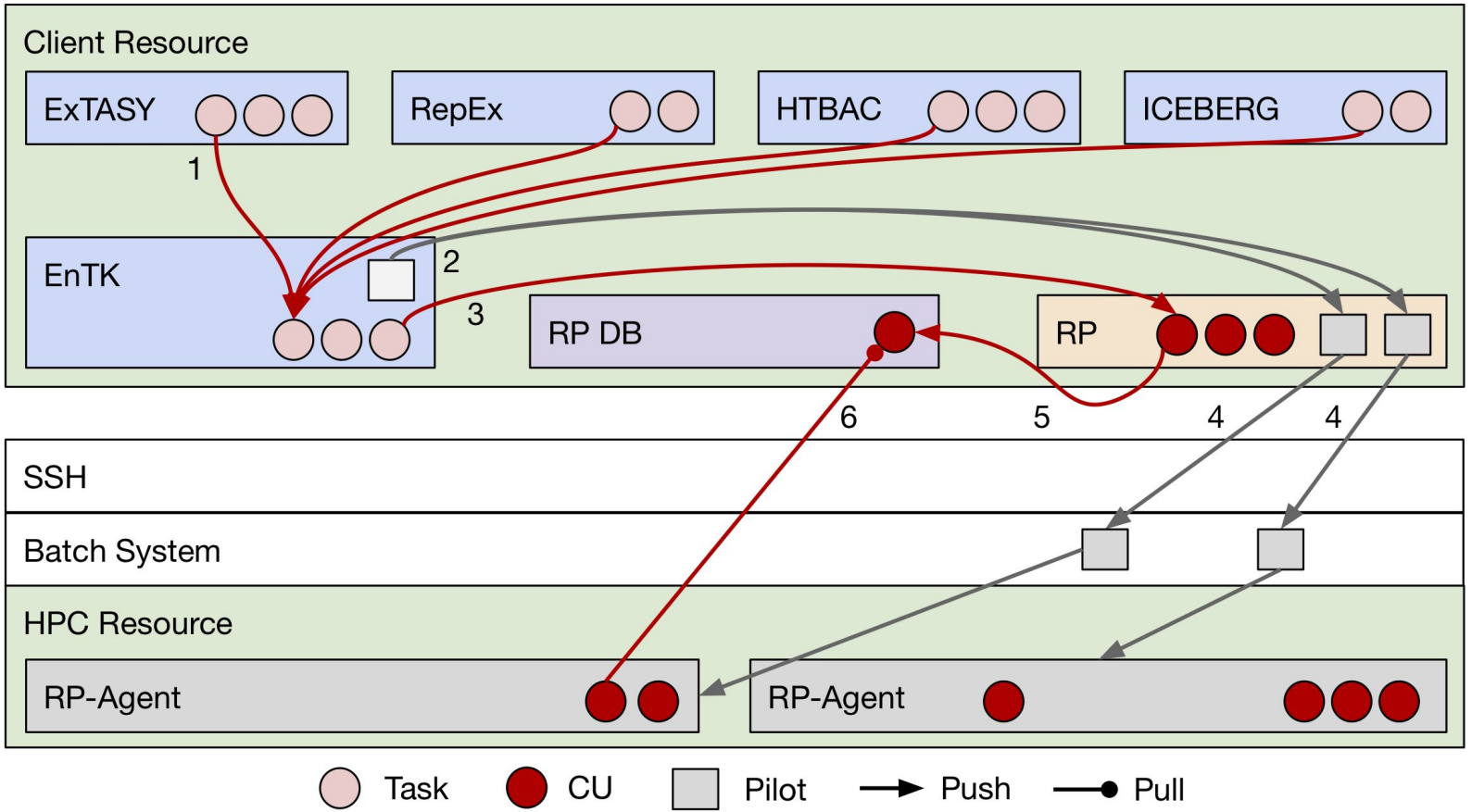
EnTK: Building Block for Ensemble based Applications

- **Design and Implementation:**
  - Simple programming model (P-S-T model)
  - *Workflow Management Layer* to manage the execution order of the individual tasks of the application: (i) AppManager, (ii) WFProcessor
  - *Workload Management Layer* to manage resources and task execution via runtime system: ExecManager
  - Defined execution model and interfaces with different runtime systems
- **Extensible, Integrate with existing tools:**
  - Provides generic building block component used by many “workflow systems” (HTBAC, ICEBERG, ExTASY, SEISFLOW, RepEx…)



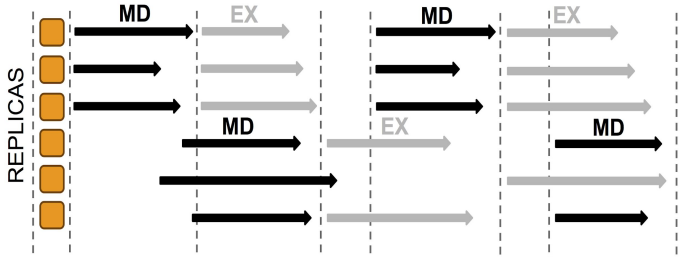


EnTK: Supporting Several Domain Specific Workflows



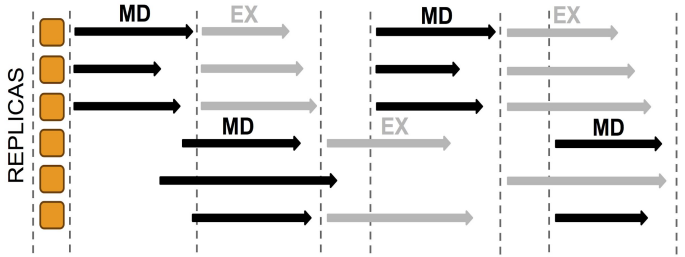
Adaptive Ensemble Algorithms: Variation on a theme

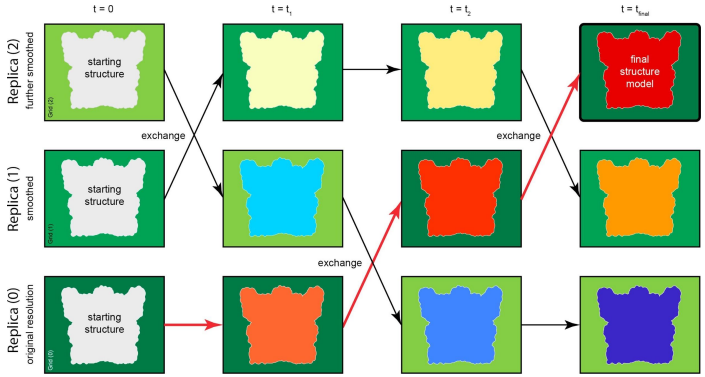
- Ensemble-based necessary, not sufficient !
- Adaptive Ensemble-based Algorithms: **Intermediate data, determines next stages**
- Adaptivity: How and What
  - Internal data: Simulation generated data used to determine “optimal” adaptation
  -



Adaptive Ensemble Algorithms: Variation on a theme

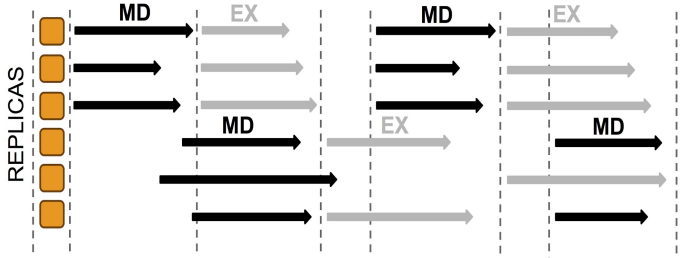
- Ensemble-based necessary, not sufficient !
- Adaptive Ensemble-based Algorithms: **Intermediate data, determines next stages**
- Adaptivity: How and What
  - Internal data: Simulation generated data used to determine “optimal” adaptation
  - External data used, e.g., experimental or separate computational process.
  - What: Task parameter(s), order, count, ….





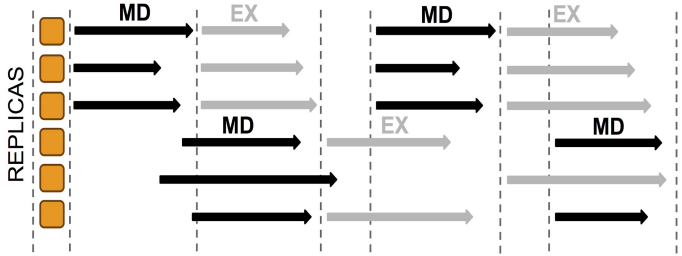
Adaptive Ensemble Algorithms: Variation on a theme

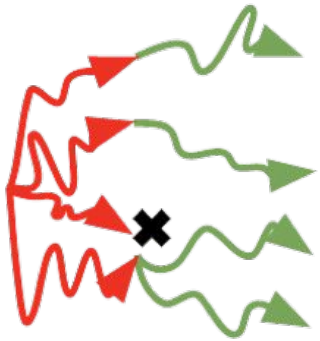
Better, Faster, Greater sampling



Adaptive Ensemble Algorithms: Variation on a theme

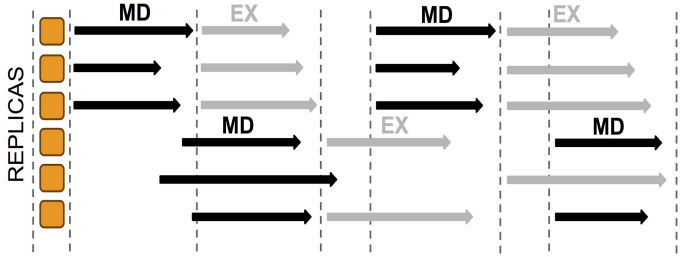
Better, Faster, Greater sampling

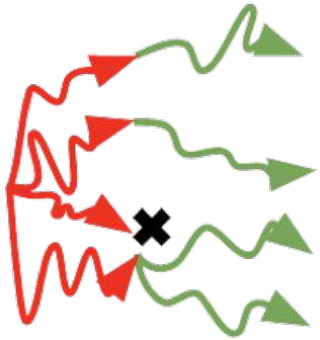


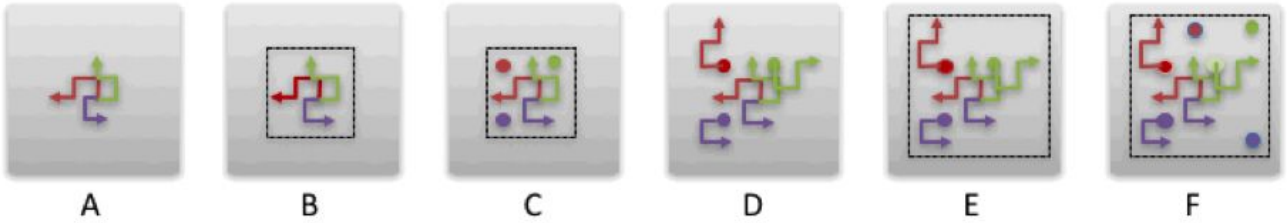


Adaptive Ensemble Algorithms: Variation on a theme

Better, Faster, Greater sampling

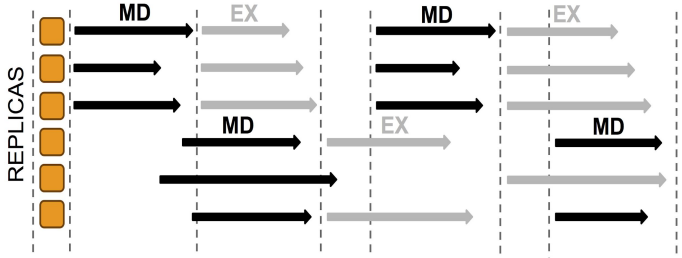


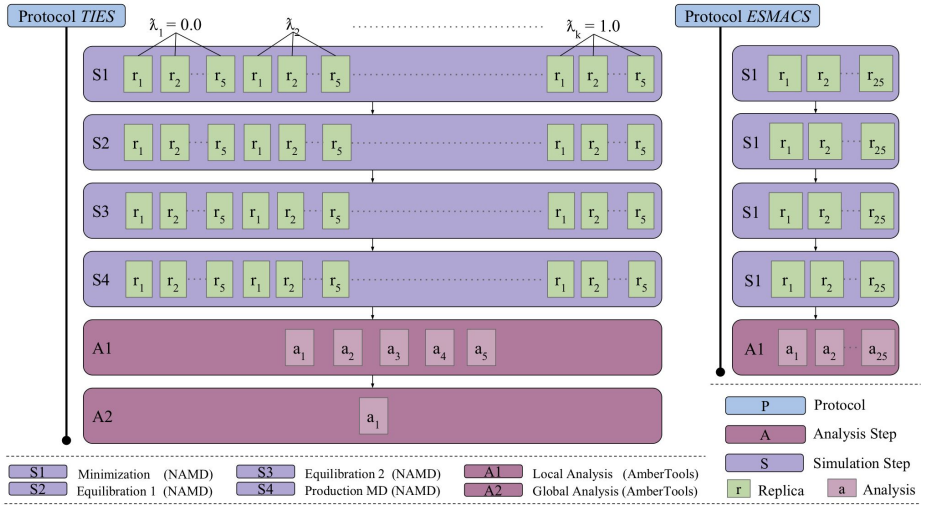


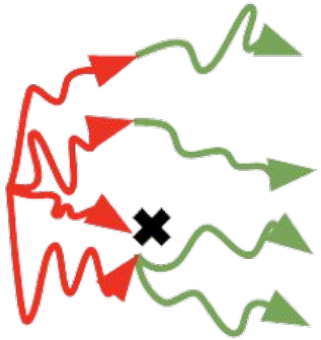


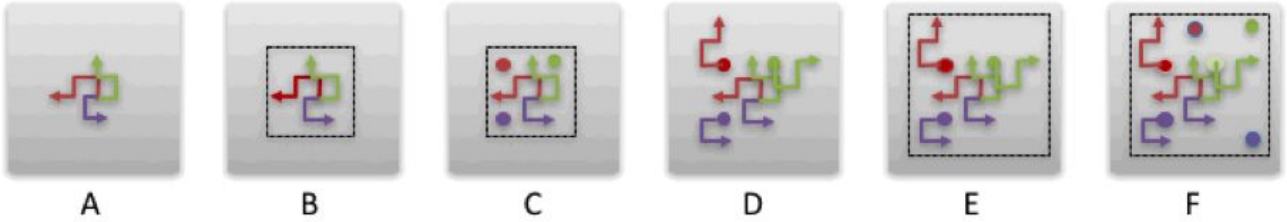
Adaptive Ensemble Algorithms: Variation on a theme

Better, Faster, Greater sampling

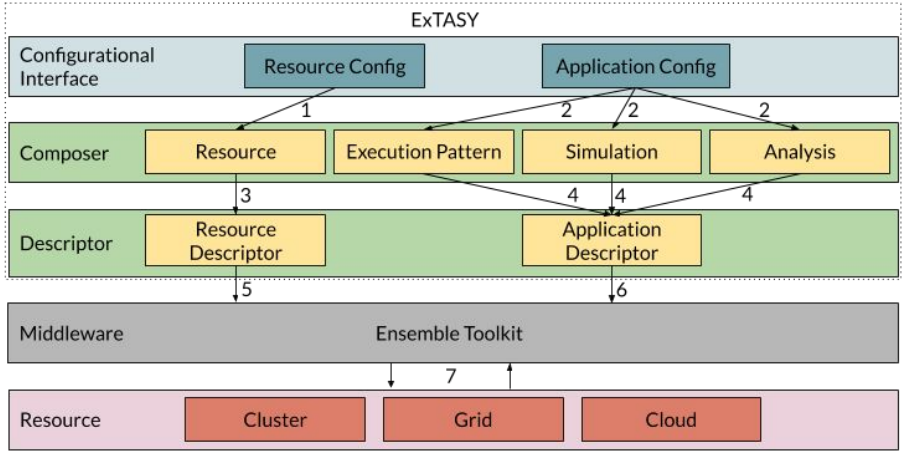


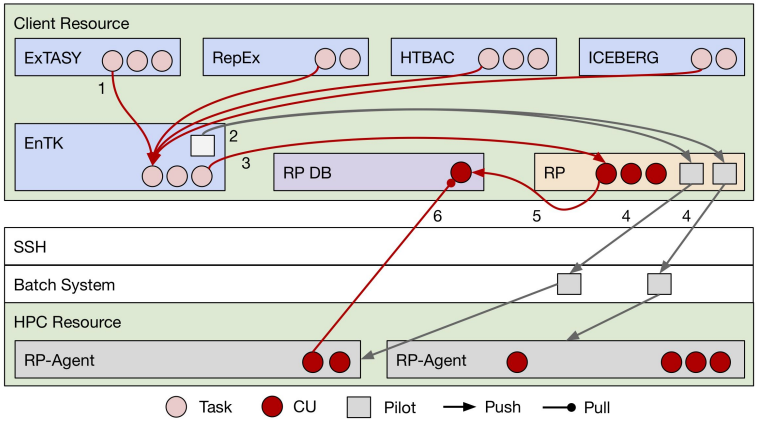


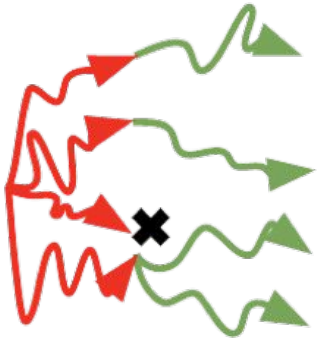




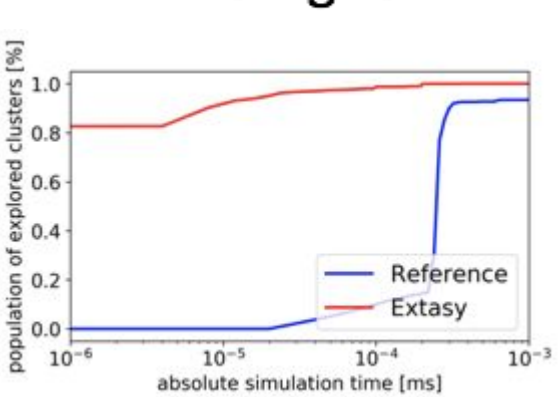
EnTK: Supporting Several Domain Specific Workflows

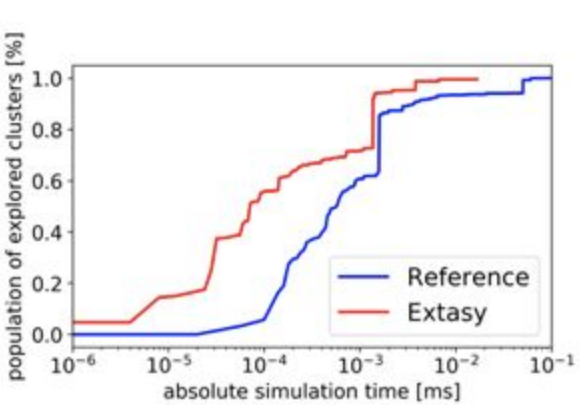


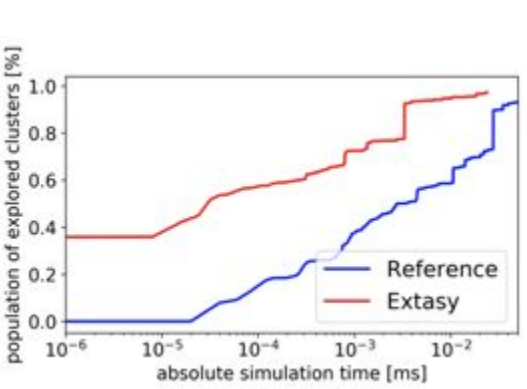


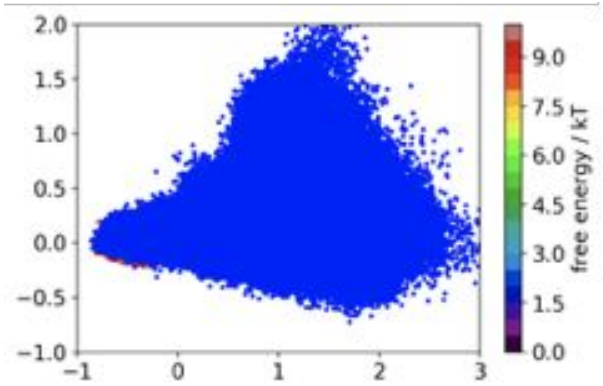


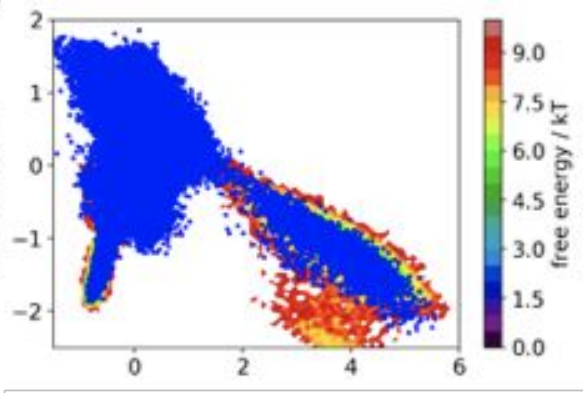
Adaptive Ensemble (MSM) vs Conventional MD

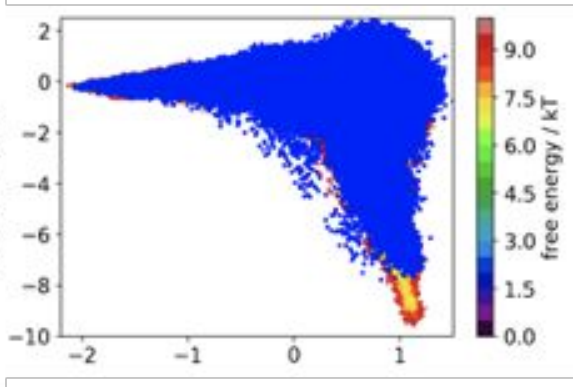




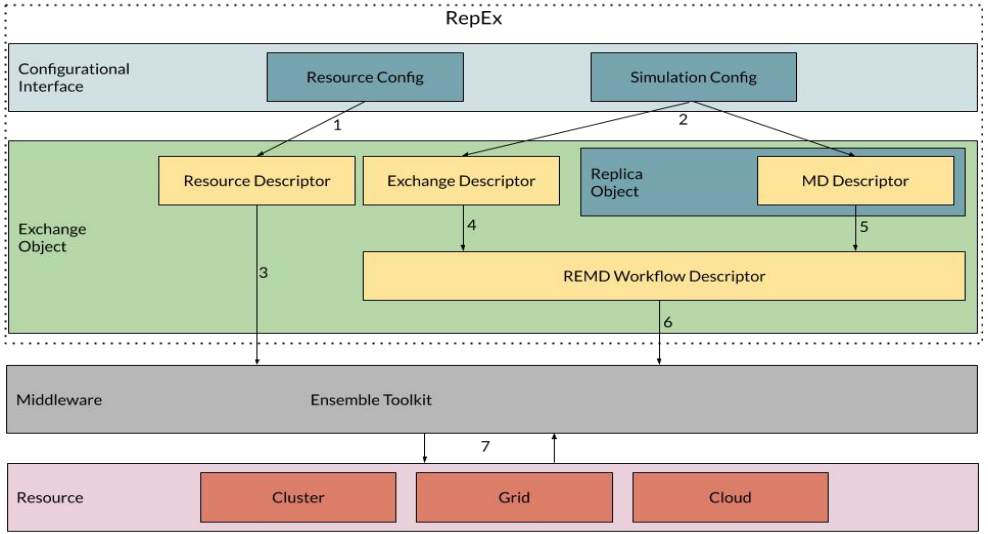


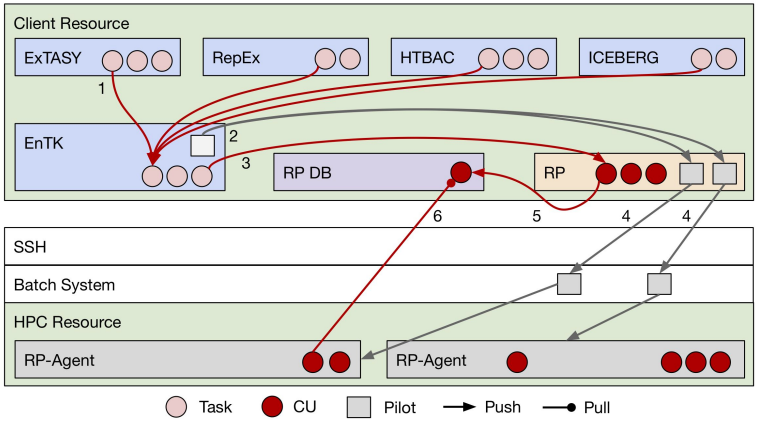


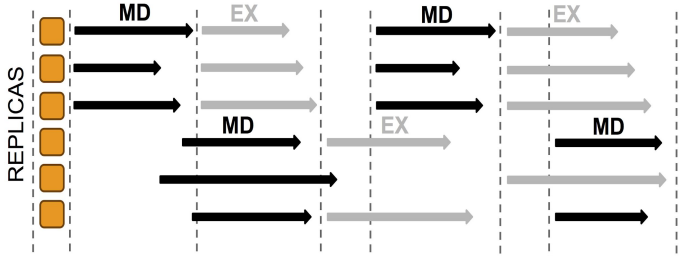




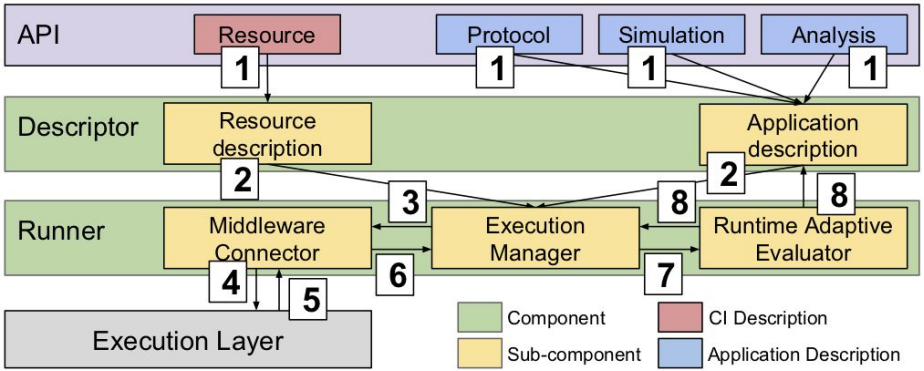
EnTK: Supporting Several Domain Specific Workflows

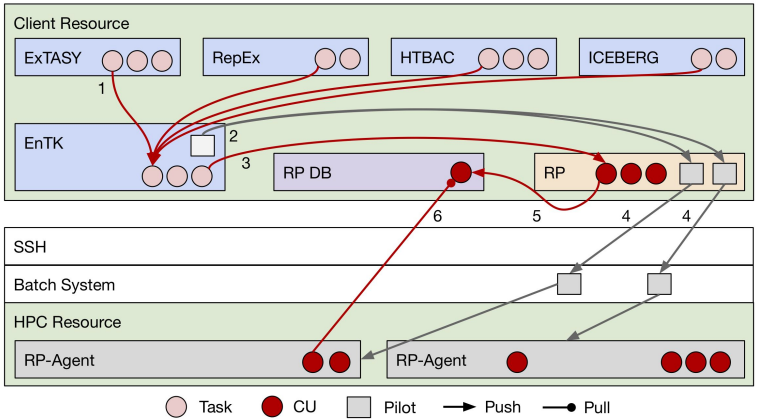


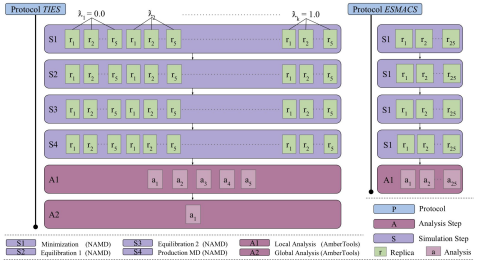




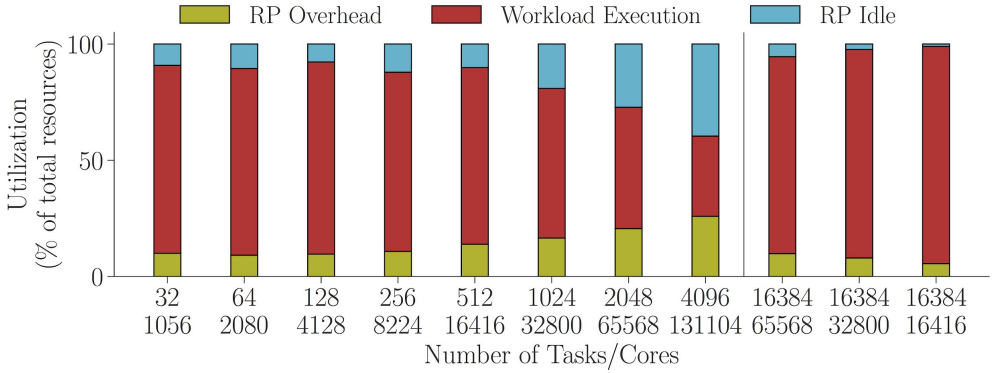
EnTK: Supporting Several Domain Specific Workflows

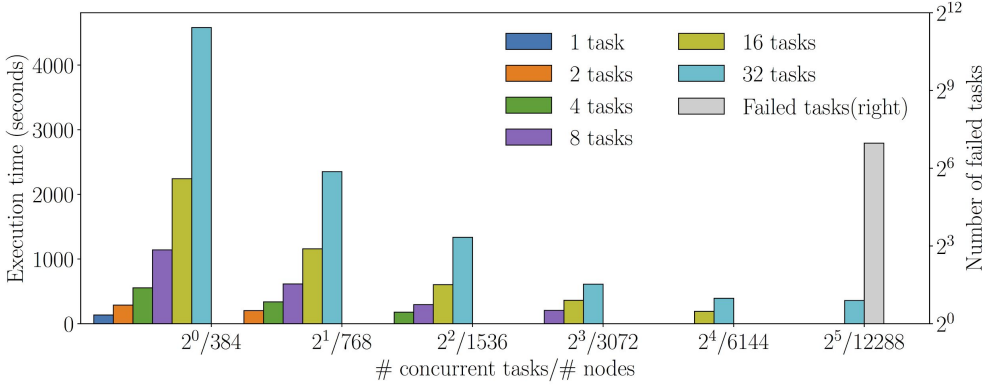


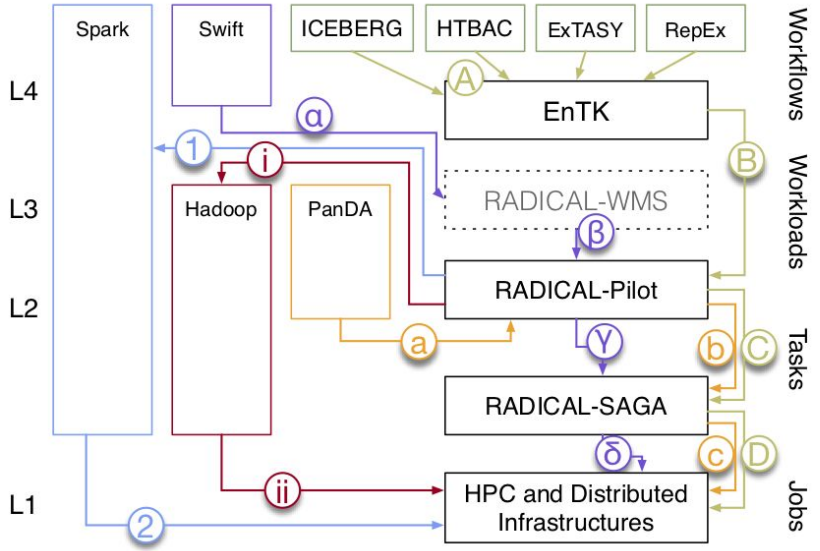




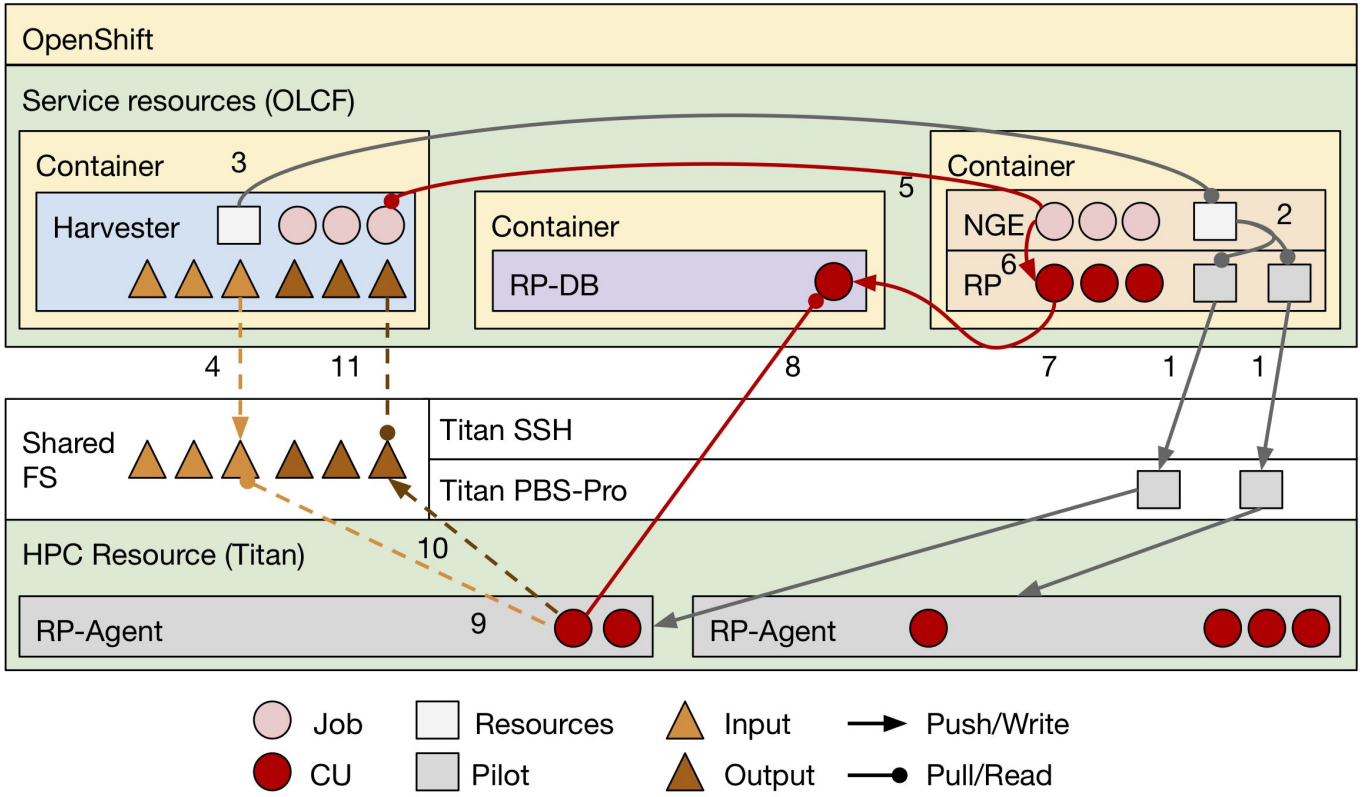
Software Systems Challenge: Specificity with Performance



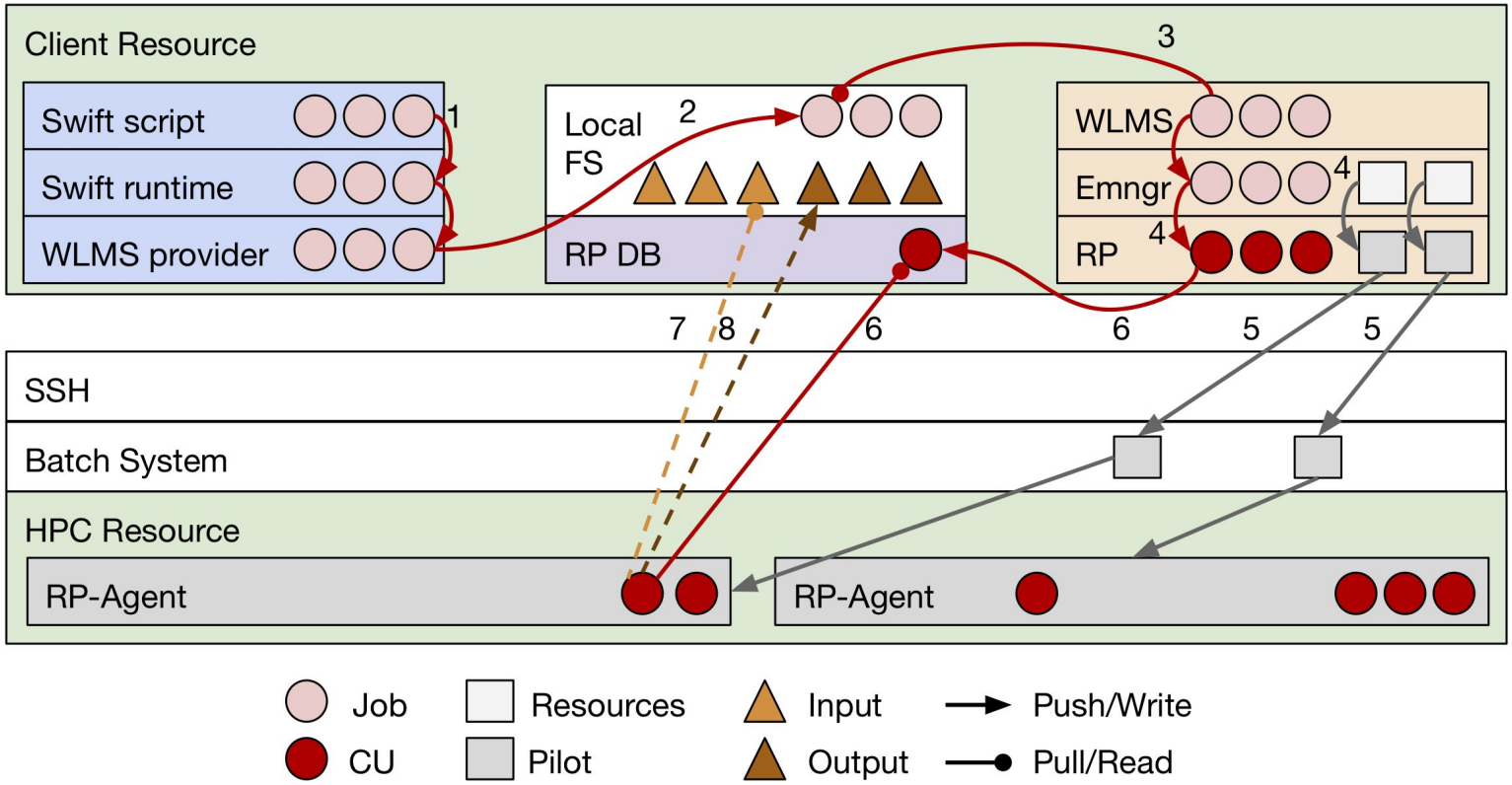




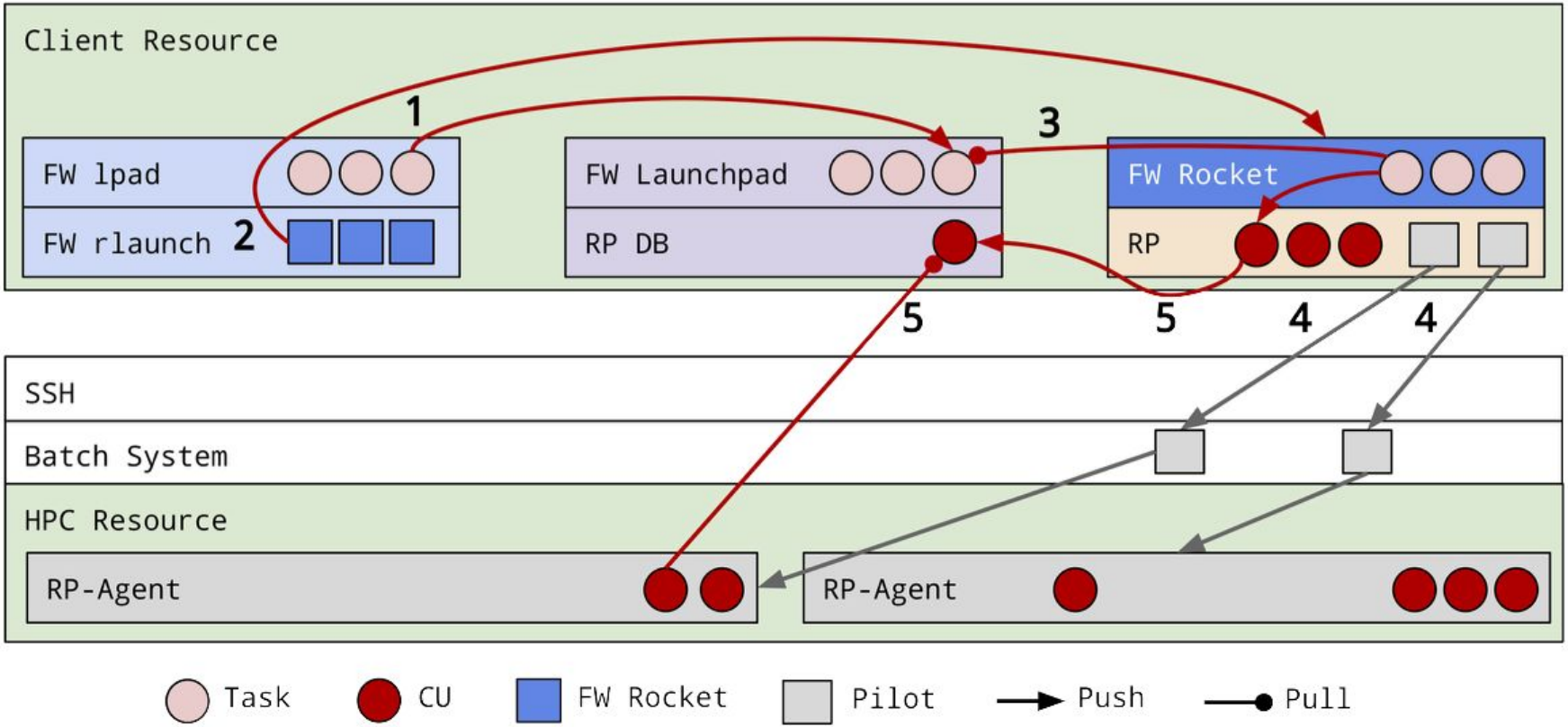
RADICAL-Pilot: Integration with ATLAS PanDA



RADICAL-Pilot: Integration with Swift



RADICAL-Pilot: Integration with Fireworks



Reformulating the Requirements

- Need a sustainable ecosystem of **both** existing and new software components from which tailored workflow systems can be composed
  - Realize agile development and composition of workflow systems that leverage rich ecosystem of existing capabilities
- **Understand needs of workflow system development, not just workflows!**
- Argue that BB changes both the **technical** and **social factors** (incentives)
- **Social:** If ecosystem of components from which workflow systems are composed, less incentive to claim “my WMS” is `better`[ ] than “your WMS”!
  - Incentivizes sharing and collective community capability
  - Disincentivizes against faux objective of interoperability of workflows
- **Technical:** BB enables expert contributions, while lowering the breadth of expertise required of workflow system developers.

Summary

- **Many advances but many challenges in workflow systems**
  - “Roll your own” systems suggest no single workflow system will serve all needs;
  - Landscape is changing in many ways; also need focus on systems developers
- **BB Approach: Promise but many open questions**
  - RCT compatible with performance, extensibility and self-sufficient
  - *Qualitative:* Need more formally rigorous definitions building block? Differentiability?
  - *Quantitative:* Develop a hypothesis & validation of how/when/if BB are more scalable and sustainable than monolithic approaches?
  - *Best Practice:* A formal understanding of granularity, type and how domain specific?
- **What can we learn from Apache Big Data Stack (ABDS)?**
  - Strong preference for functional specialization, as opposed to interoperability!
- **Well scoped BB for Workflow Systems as components of “Open Workflow”needed**
  - Will not solve all, but will solve problems of proliferation and interoperability