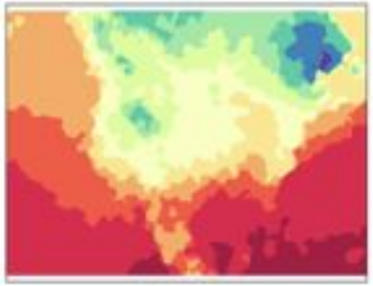
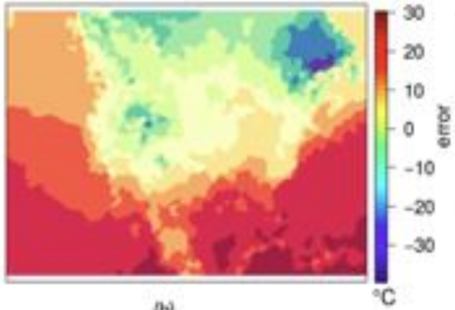
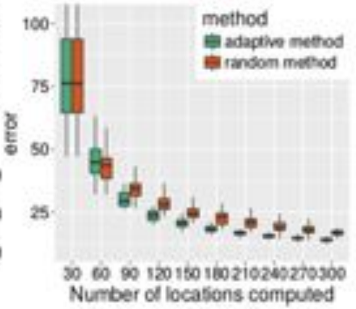
Middleware Building Blocks for Workflow Systems

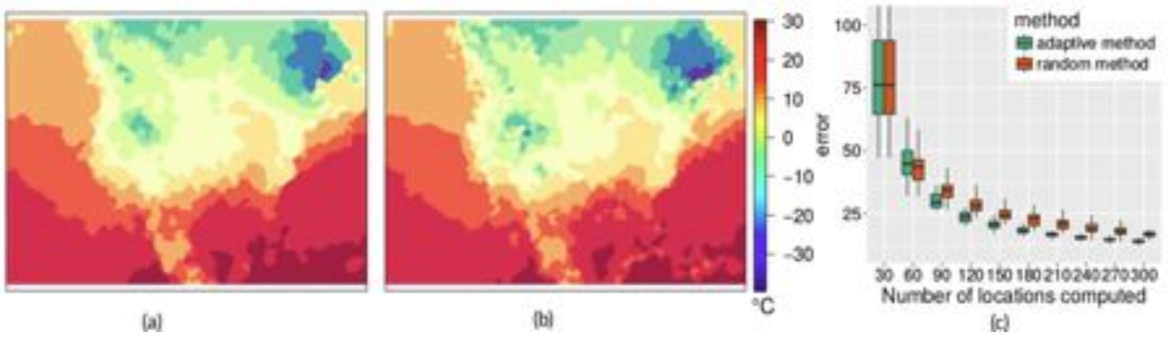
Shantenu Jha
Brookhaven National Laboratory and Rutgers University

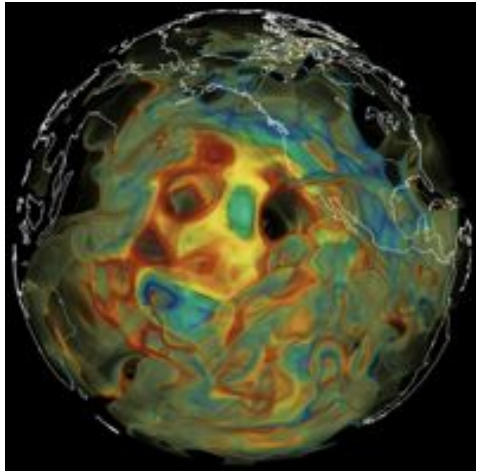
http://radical.rutgers.edu

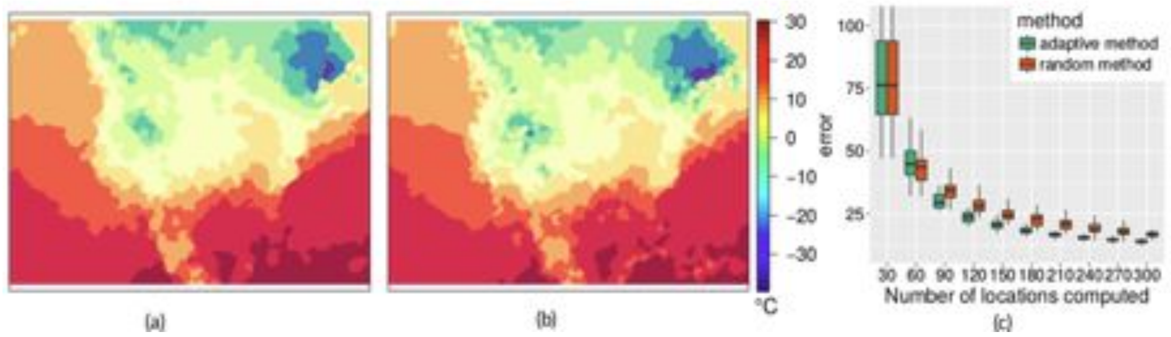

(a)

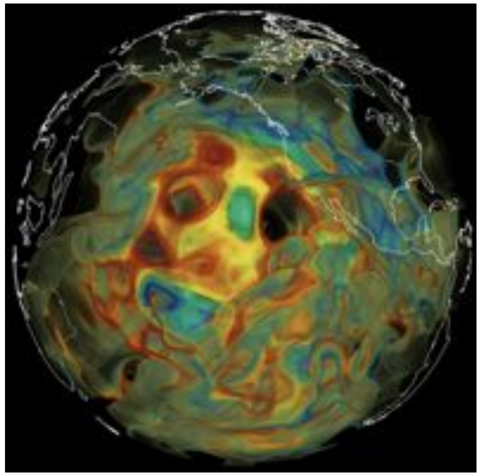

30
20
10
0
−10
−20
−30
°C
(b)

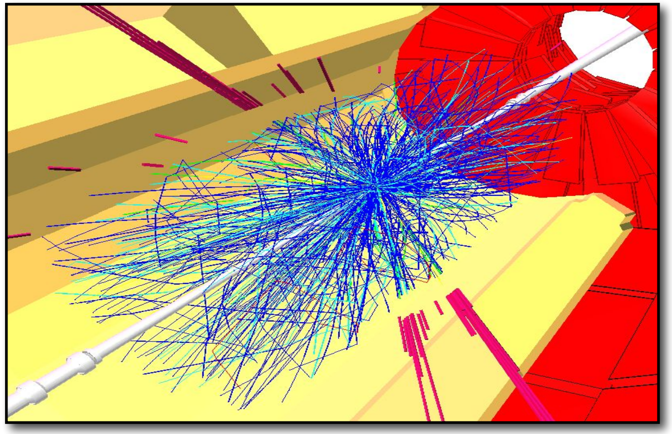

method
adaptive method
random method
error
100
75
50
25
30
60
90
120
150
180
210
240
270
300
Number of locations computed
(c)

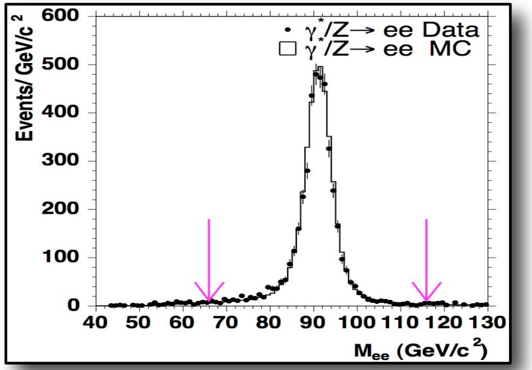

30
20
10
0
−10
−20
−30
°C
(a)
(b)
method
adaptive method
random method
error
100
75
50
25
30 60 90 120 150 180 210 240 270 300
Number of locations computed
(c)

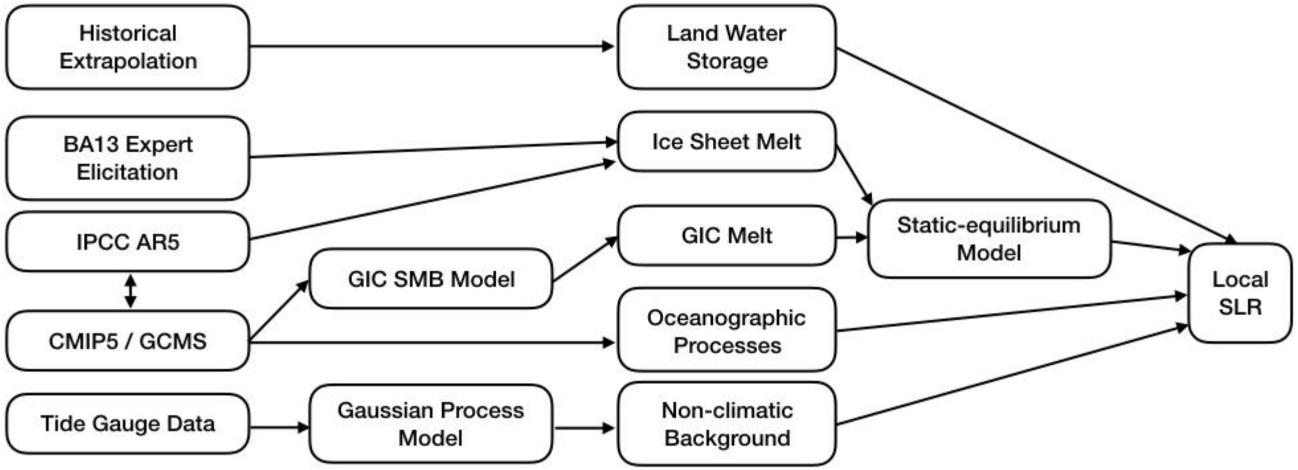



30
20
10
0
−10
−20
−30
°C
(a)
(b)
method
🟩 adaptive method
🟥 random method
100
75
50
25
error
30 60 90 120 150 180 210 240 270 300
Number of locations computed
(c)

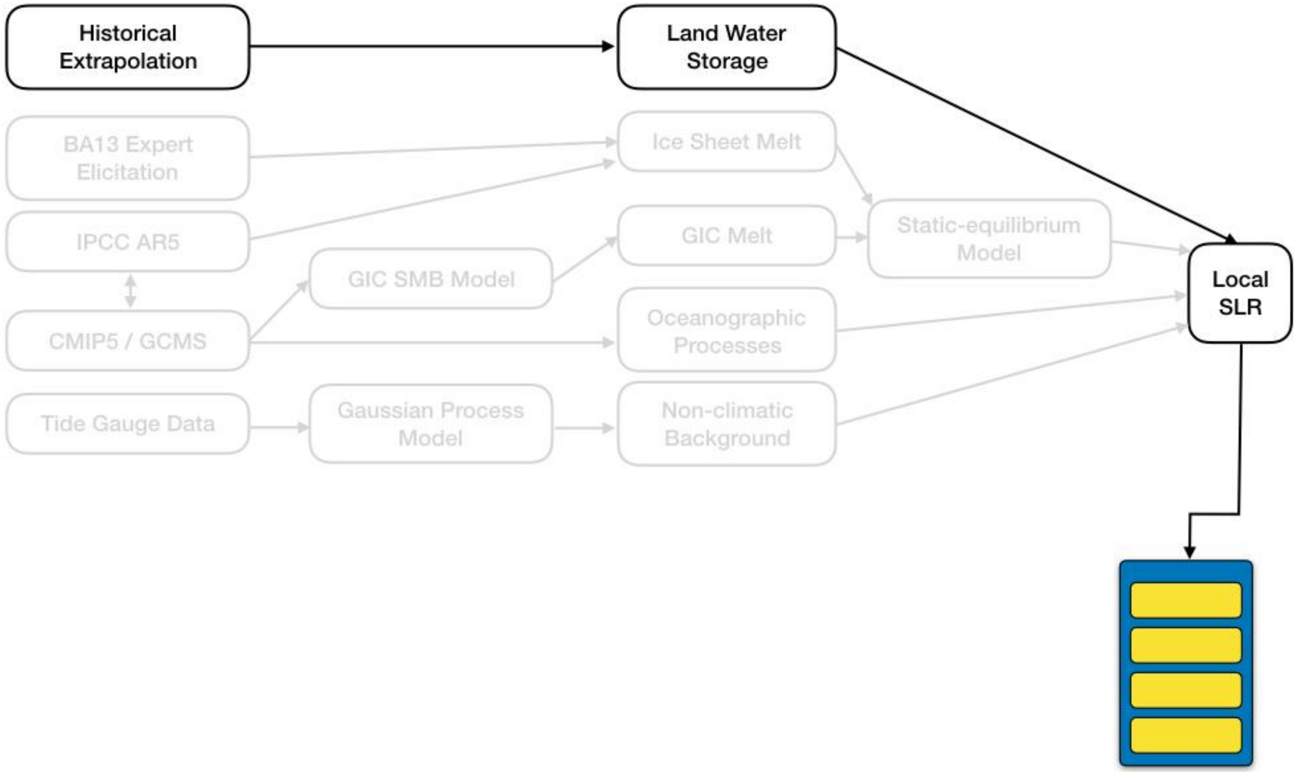




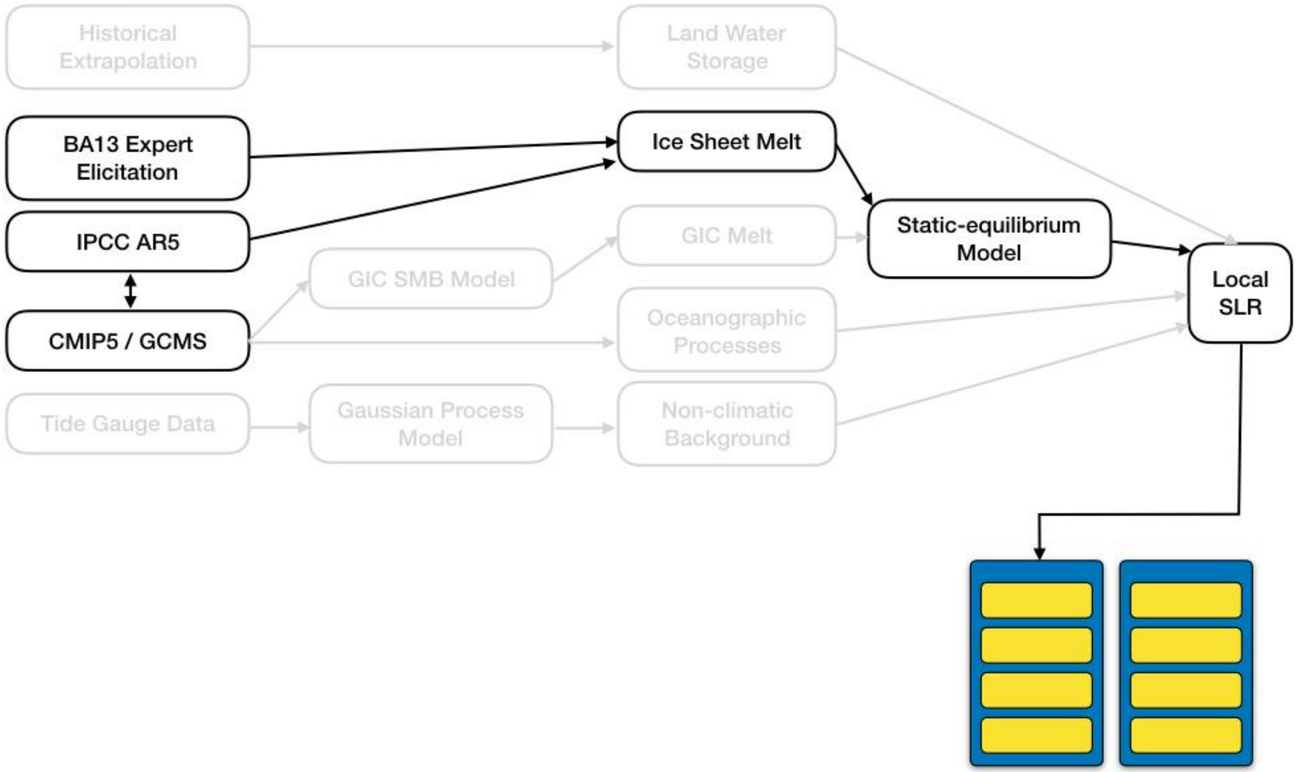

● $\gamma^*/Z \rightarrow ee$ Data
□ $\gamma^*/Z \rightarrow ee$ MC
Events/GeV/c^2
600
500
400
300
200
100
0
40 50 60 70 80 90 100 110 120 130
M_{ee} (GeV/c^2)



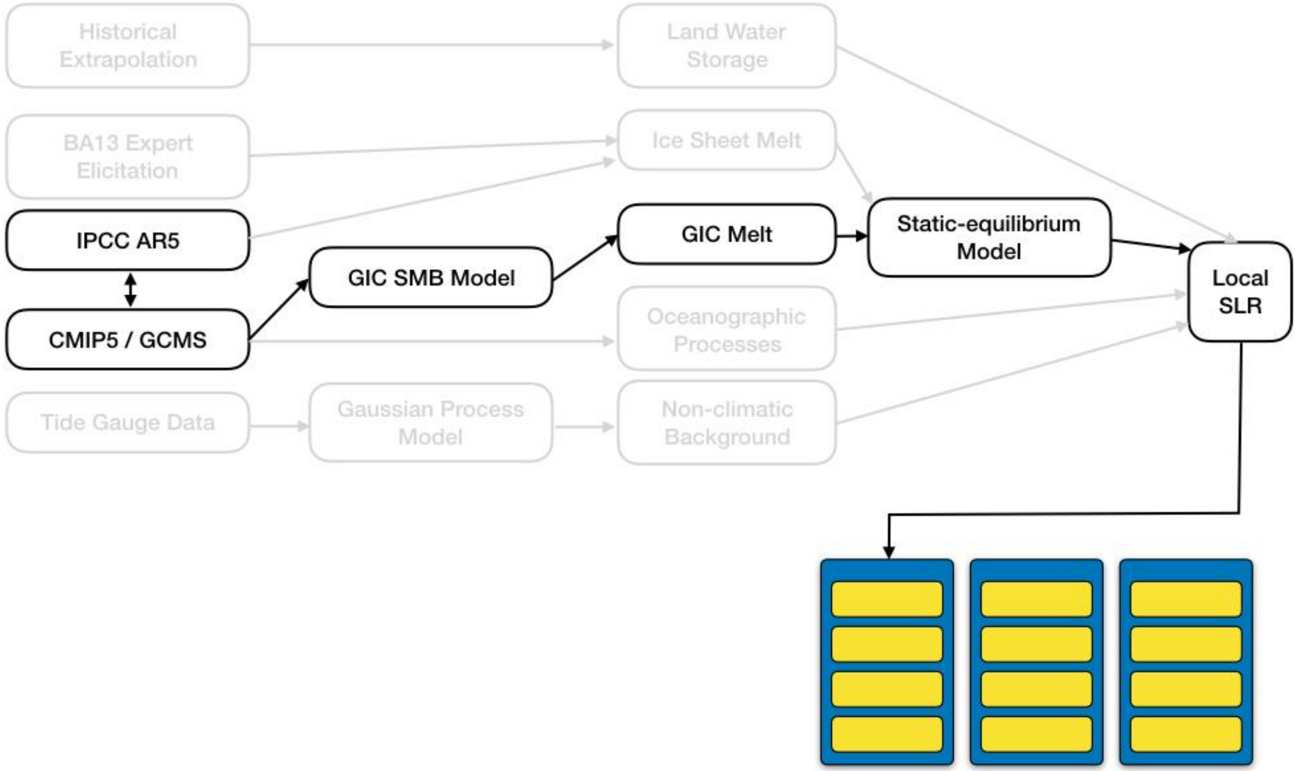
Kopp 2014 Workflow



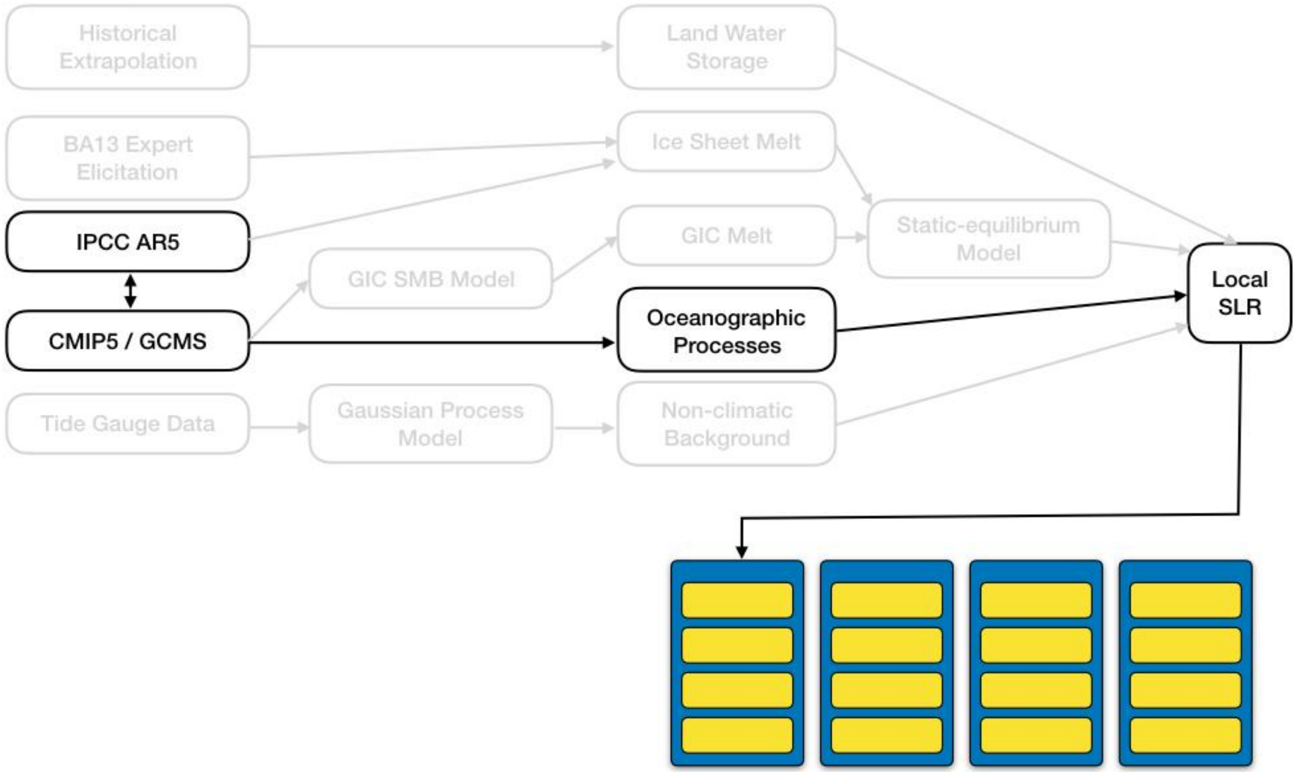
Kopp 2014 Workflow



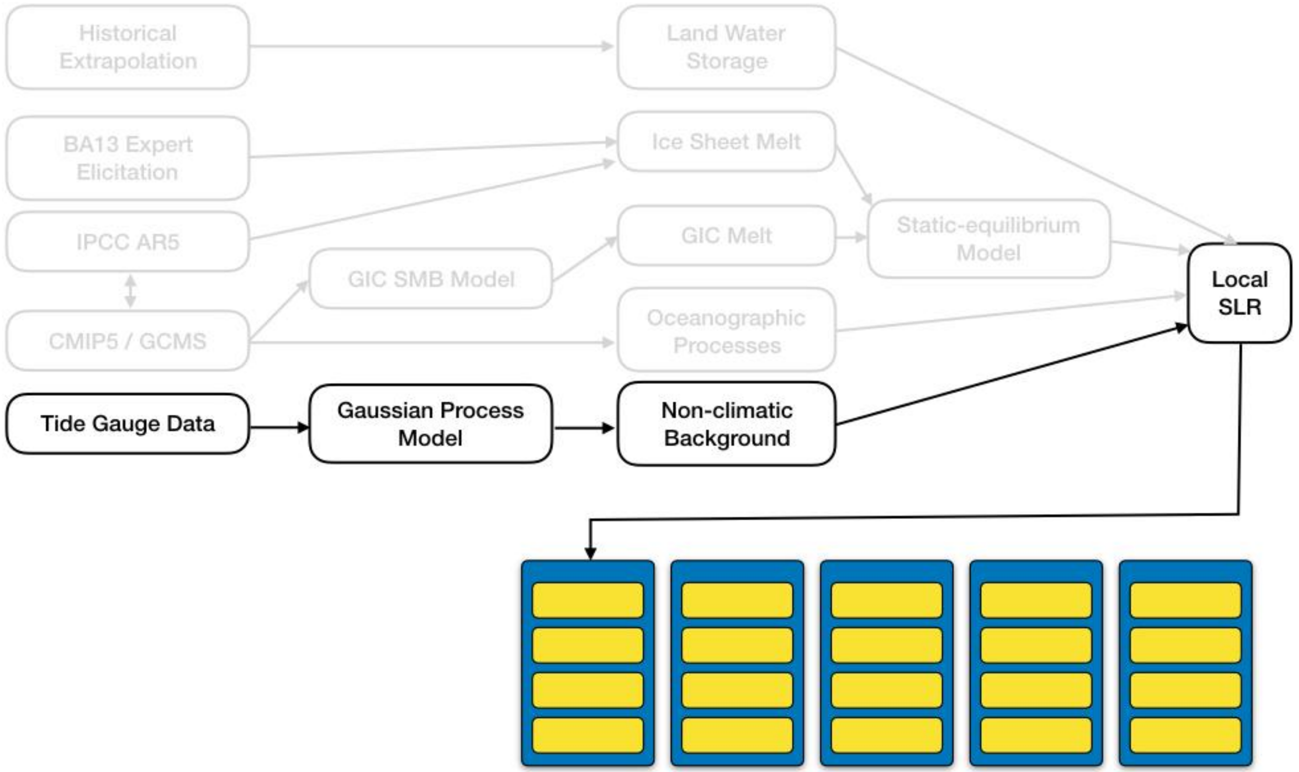
Kopp 2014 Workflow



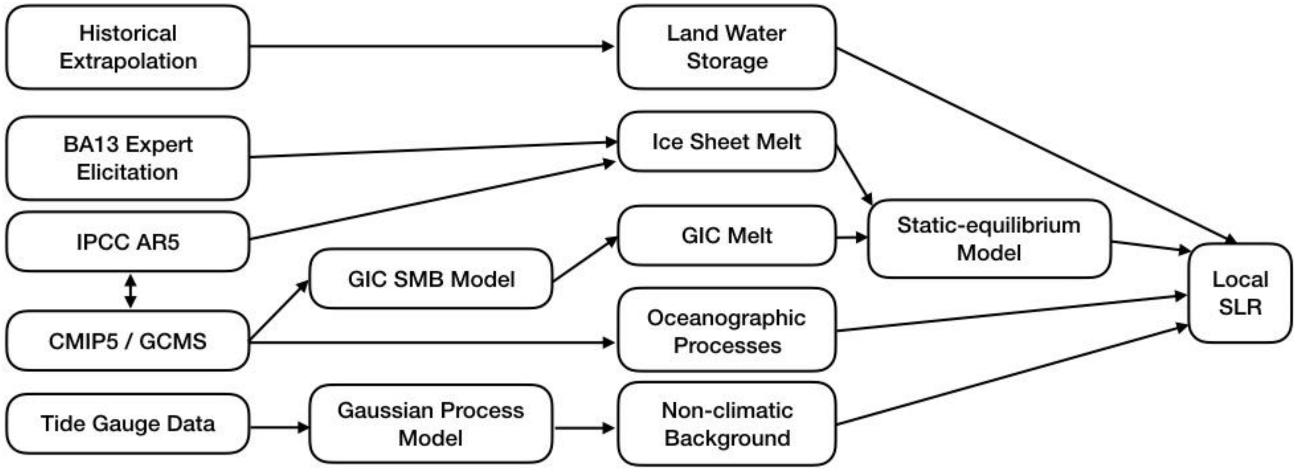
Kopp 2014 Workflow

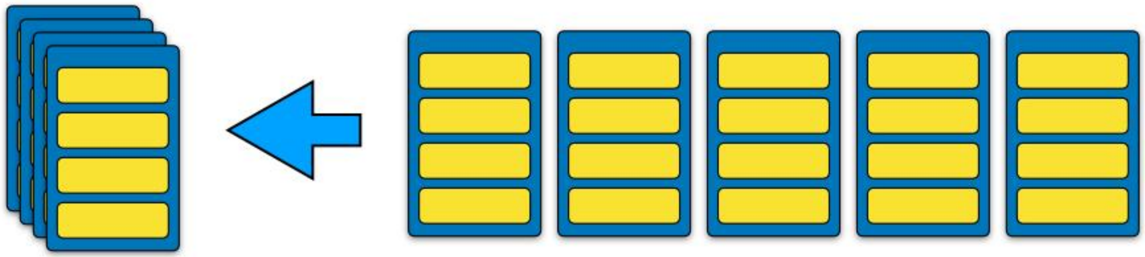


Kopp 2014 Workflow



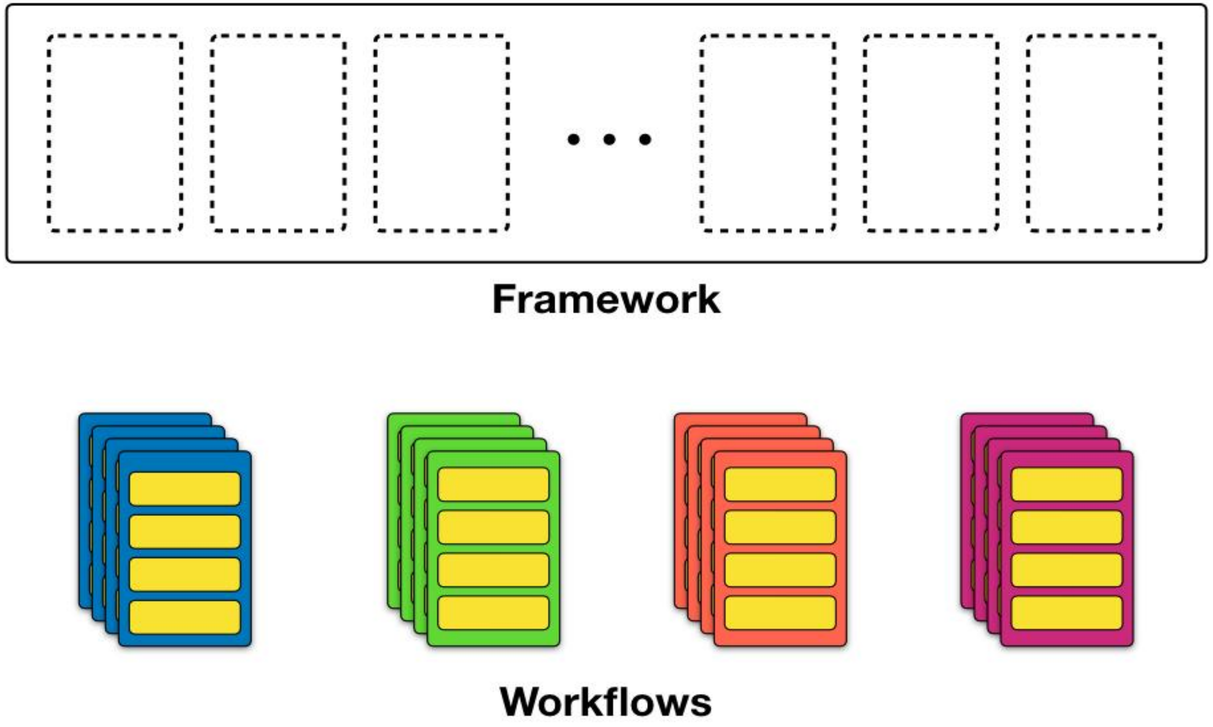
Kopp 2014 Workflow



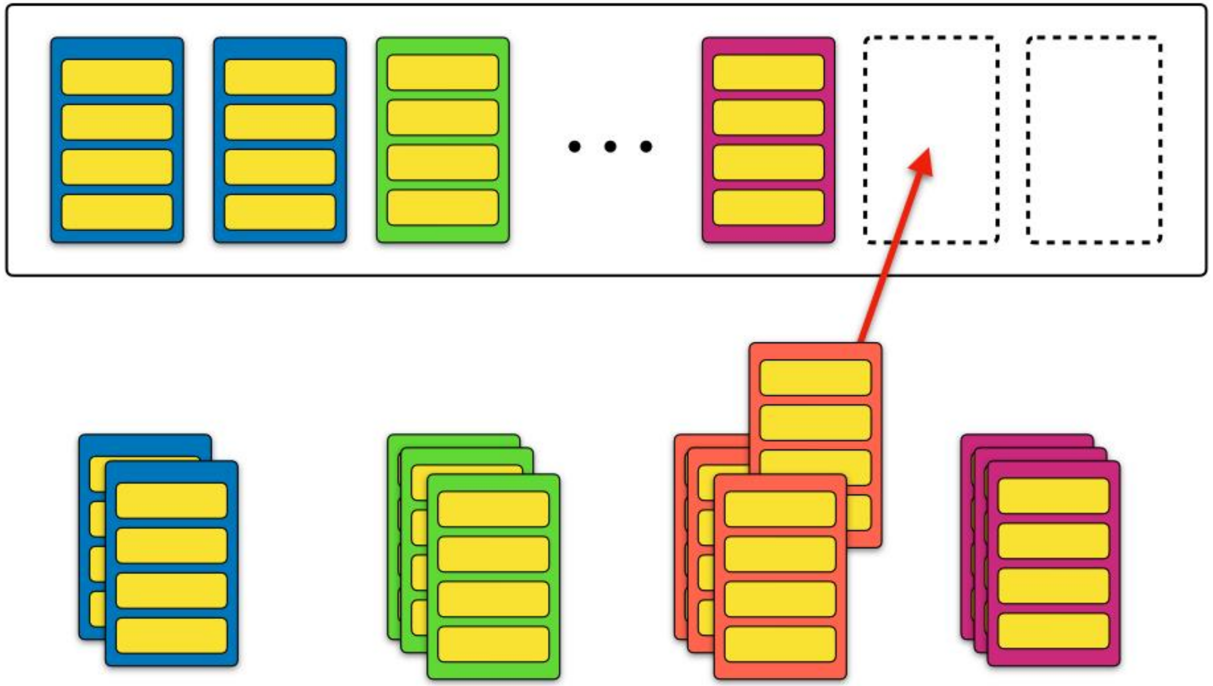


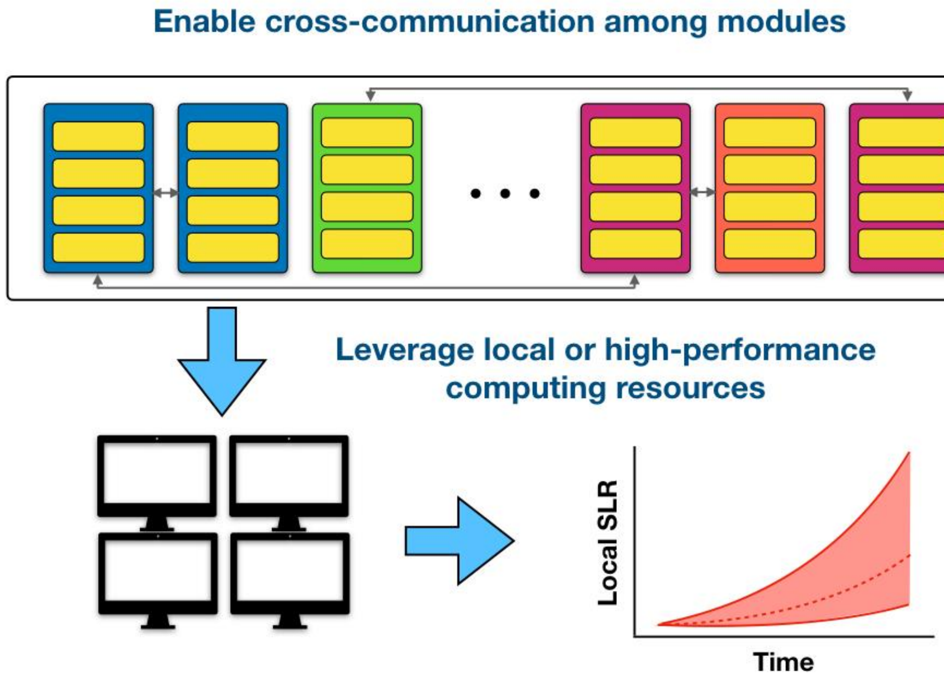
Kopp 2014 Workflow

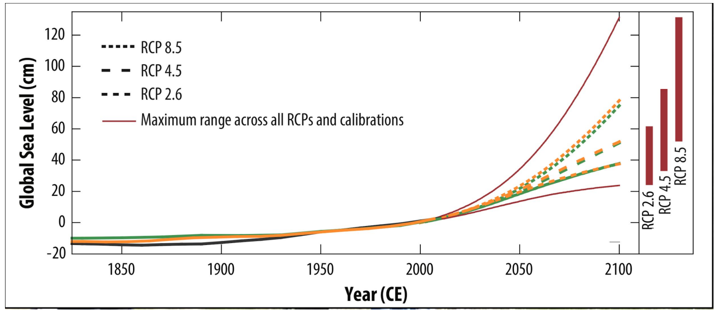
FACTS: Framework for Analyzing Changes To Sea-level



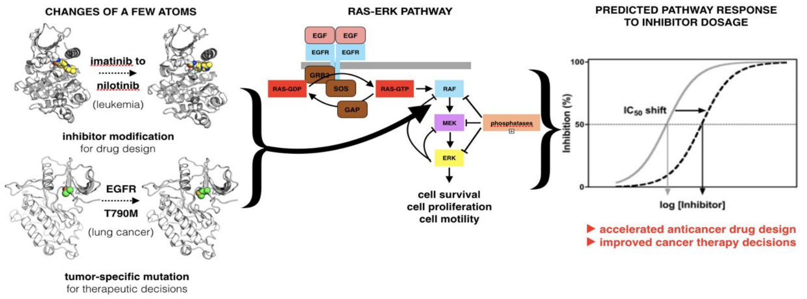
Mix and match sub-models from multiple workflows

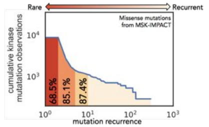


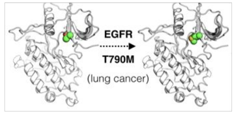

Enable cross-communication among modules
Leverage local or high-performance computing resources
Local SLR
Time

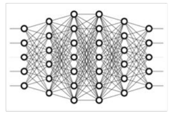


built upon RADICAL-Cybertools





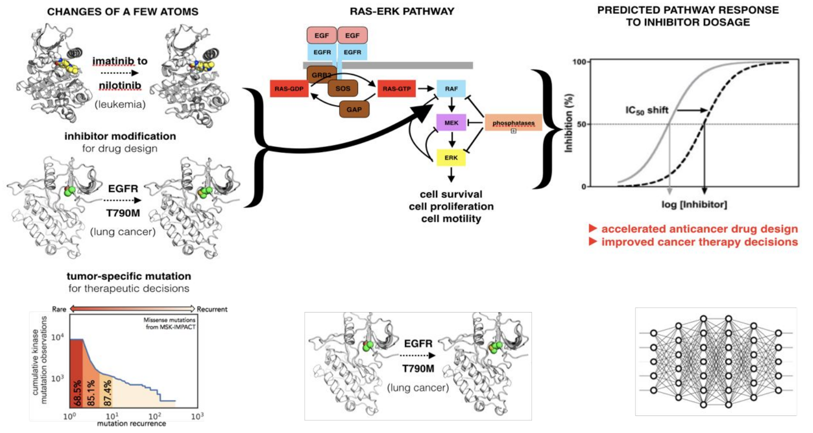




55,000+ clinical cancer kinase mutations

MD simulations to compute kinase inhibitor resistance

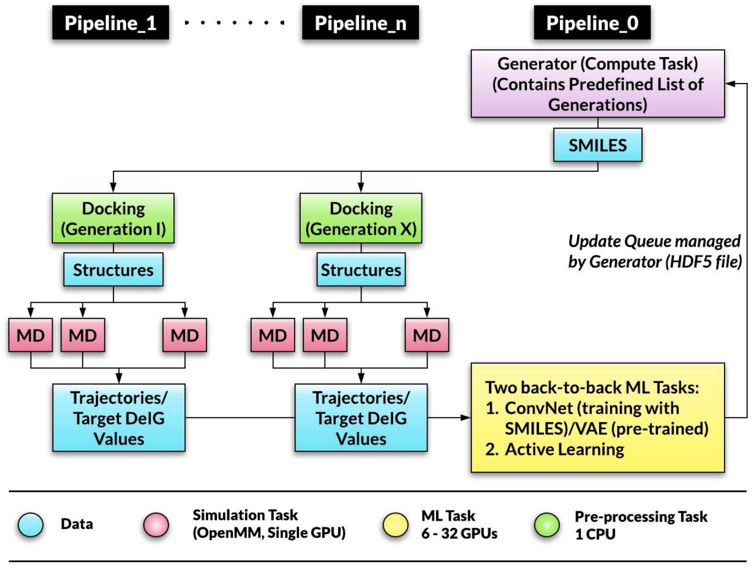
Train machine learning (ML) models to predict therapeutic effectiveness

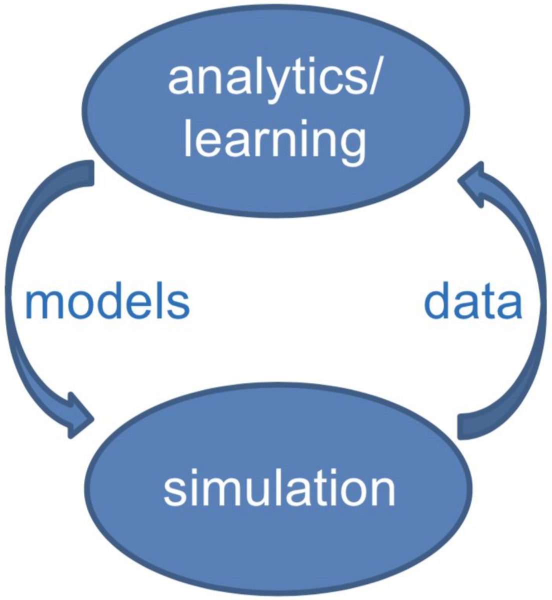

CHANGES OF A FEW ATOMS
imatinib to nilotinib
(leukemia)
inhibitor modification for drug design
EGFR T790M
(lung cancer)
tumor-specific mutation for therapeutic decisions
RAS-ERK PATHWAY
EGF
EGFR
GRB2
SOS
GAP
RAS-GDP
RAS-GTP
RAF
MEK
ERK
phosphatases
cell survival
cell proliferation
cell motility
PREDICTED PATHWAY RESPONSE TO INHIBITOR DOSAGE
Inhibition (%)
IC50 shift
log [Inhibitor]
▶ accelerated anticancer drug design
▶ improved cancer therapy decisions
Rare
Recurrent
Missense mutations from MSK-IMPACT
cumulative kinase mutation observations
68.5%
85.1%
87.4%
mutation recurrence

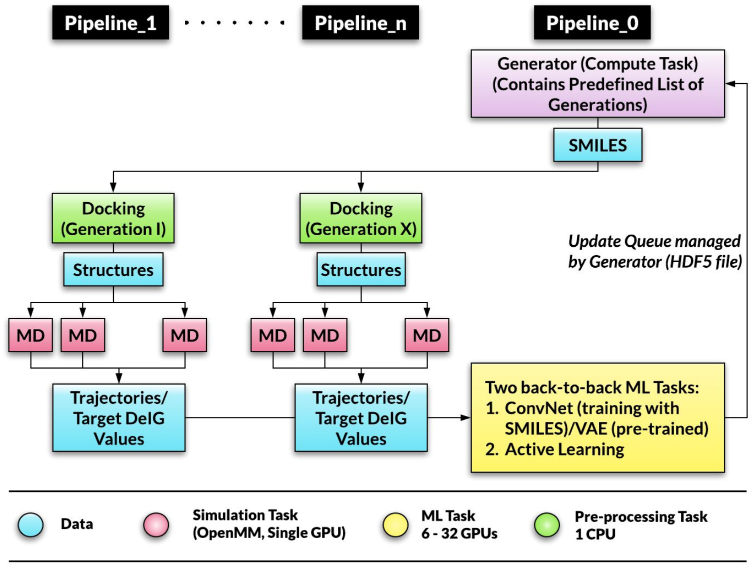
55,000+ clinical cancer kinase mutations

MD simulations to compute kinase inhibitor resistance

Train machine learning (ML) models to predict therapeutic effectiveness

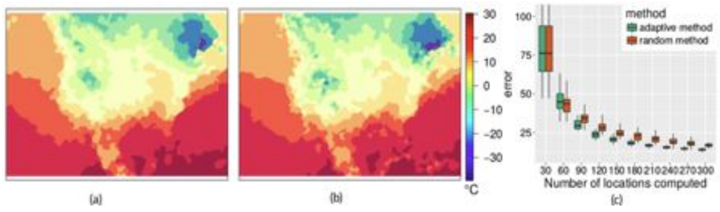

Pipeline_1
Pipeline_n
Pipeline_0
Generator (Compute Task) (Contains Predefined List of Generations)
SMILES
Docking (Generation I)
Docking (Generation X)
Structures
MD
Trajectories/ Target DelG Values
Update Queue managed by Generator (HDF5 file)
Two back-to-back ML Tasks:
1. ConvNet (training with SMILES)/VAE (pre-trained)
2. Active Learning
Data
Simulation Task (OpenMM, Single GPU)
ML Task 6 - 32 GPUs
Pre-processing Task 1 CPU


analytics/ learning
models
data
simulation


Pipeline_1
Pipeline_n
Pipeline_0
Generator (Compute Task) (Contains Predefined List of Generations)
SMILES
Docking (Generation I)
Docking (Generation X)
Structures
Structures
MD
MD
MD
MD
MD
MD
Trajectories/ Target DelG Values
Trajectories/ Target DelG Values
Update Queue managed by Generator (HDF5 file)
Two back-to-back ML Tasks:
1. ConvNet (training with SMILES)/VAE (pre-trained)
2. Active Learning
Data
Simulation Task (OpenMM, Single GPU)
ML Task 6 - 32 GPUs
Pre-processing Task 1 CPU

Some Statements From NYSDS19 ….

- “*All computational problems require workflows*”



Some Statements From NYSDS19 ….

- *“All computational problems require workflows*
- *“Everyone has a different workflow”*

PANGEO

A community platform for Big Data geoscience

Some Statements From NYSDS19 ….

- *“All computational problems require workflows”*
- *“Everyone has a different workflow”*
- *“The optimization of the end-to-end performance of a workflow is important (and different..)”*

CRAY
THE SUPERCOMPUTER COMPANY

Some Statements From NYSDS19 ….

- "*All computational problems require workflows*"
- *"Everyone has a different workflow"*
- *"The optimization of the end-to-end performance of a workflow is important (and different..)"*
- *"**Nothing tends so much to the advancement of knowledge as the application of a new instrument.** The native intellectual powers of people in different times are not so much the causes of the different success of their labors, as the peculiar nature of the means and artificial resources in their possession" -- Humphrey Davy*

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:** DOE 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, …

Perspective on Workflows & Systems

- Initially workflow **management** systems provided **end-to-end** capabilities:
    - “Big Science”; software infrastructure was fragile, missing services
    - Run many times, for many users: amortisation of development overhead
- Workflows aren’t what they used to be
    - HTC important but other design points: automation, sophistication, …
    - **The workflow** is a manifestation of algorithmic & methodological innovation
- The infrastructure is not what it used to be either!
    - Python ecosystem, e.g., task distribution and coordination systems
    - Apache (big) data stack of analysis tools; container technologies ..

Status Quo: Workflows & Workflow Systems

- Need **sustainable** ecosystem of both existing and new software components from which **tailored workflow systems** can be composed
  - Lower barrier to integration of components
- Separate system and performance sensitive (e.g., advanced scheduling, process / resource management etc.) from application facing components
  - Engineer for design points: Usability vs Functionality vs Scalability
- **A systems approach** which addresses both **technical** & **social factors**
  - Incentivize sharing and collective community capability
  - Enable expert contributions, while lowering the breadth of expertise required of workflow system developers.

Outline

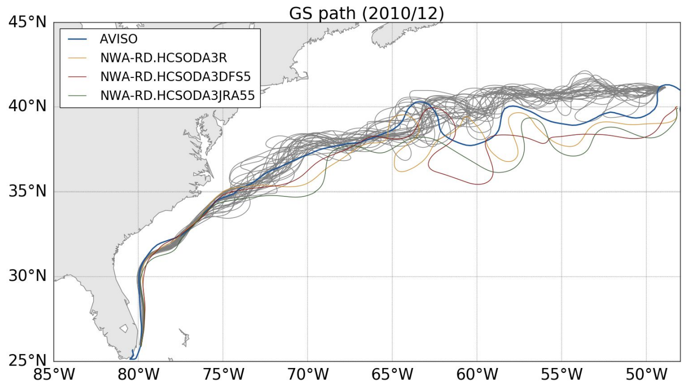
- **Historical Perspective on Workflows**

- **Ensemble Computational Model**
  - System & performance sensitive components versus user-facing component

- RADICAL-Cybertools: Building Blocks for Middleware for Workflow System
  - Building Blocks: RADICAL-Pilot and Ensemble ToolKit (EnTK)
  - Performance Challenges: Pilot-Abstraction and RADICAL-Pilot
  - Software Challenges: Extensibility and Middleware Building Blocks

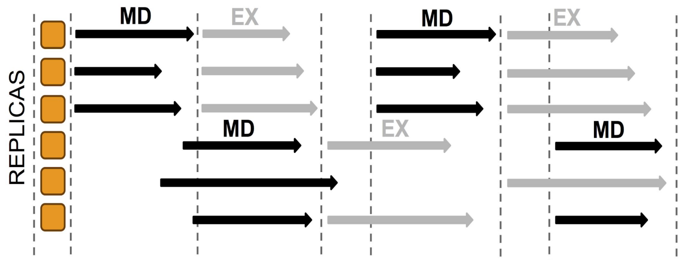
Ensemble Computational Model

- Many applications formulated as **multiple** tasks, as opposed to large but **single** task.
- When the collective outcome of a set of tasks is important, defined as ensemble:
    - Distinct from HTC, typically tasks are I^4: (Independent, Idempotent, Identical, Insensitive to order)
- Performance is mix of HPC and HTC
    - Challenges go beyond traditional strong and weak scaling
    - Concurrent $N_E(t)$, total N_E, communication frequency ..
- Complexity of dependence resolution typically less than workflows



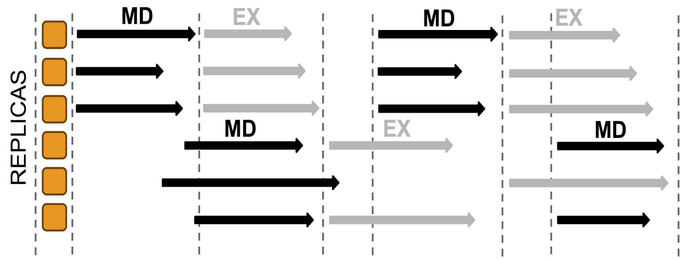
Adaptive Ensemble Algorithms: Variation on a theme

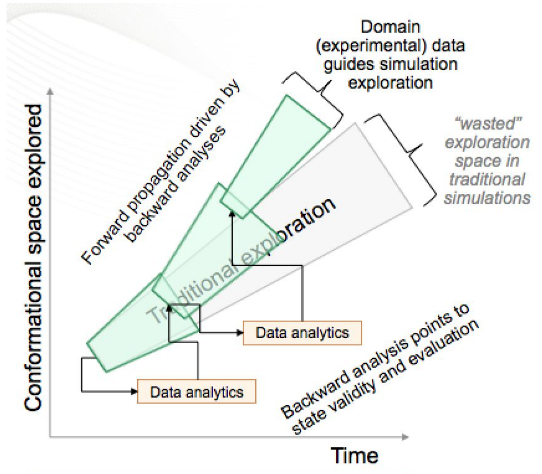
- Ensemble-based methods necessary, but not sufficient !
- **Adaptive** Ensemble-based Algorithms: Intermediate data, determines next stages



Adaptive Ensemble Algorithms: Variation on a theme

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

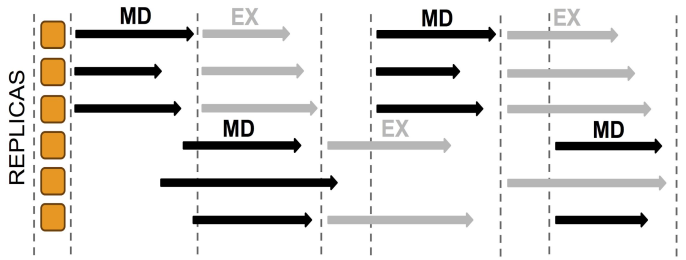


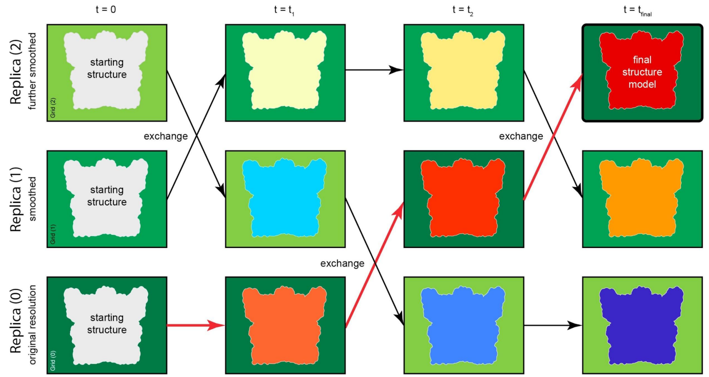

Domain (experimental) data guides simulation exploration
"wasted" exploration space in traditional simulations
Forward propagation driven by backward analyses
Traditional exploration
Data analytics
Data analytics
Backward analysis points to state validity and evaluation
Conformational space explored
Time

*Chodera, J.D., Noe, F., Curr. Opin. Struct. Biol. (2014)

Adaptive Ensemble Algorithms: Variation on a theme

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



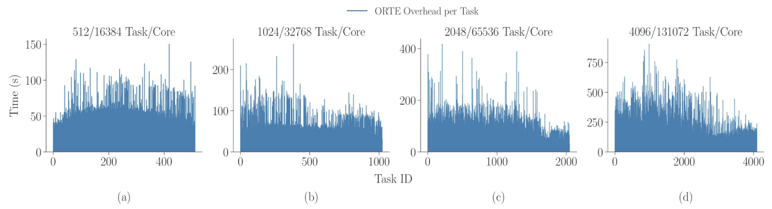


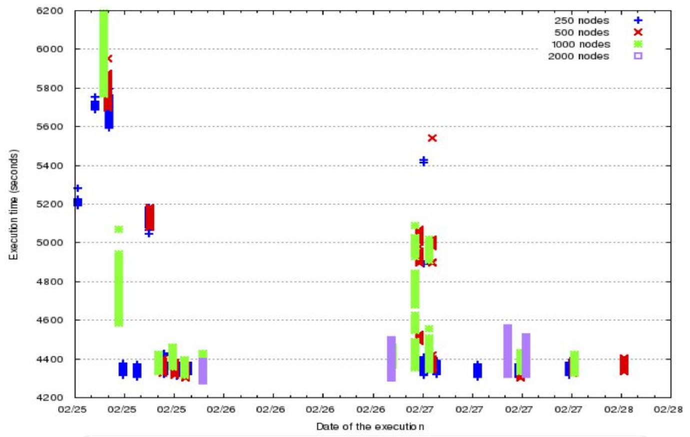
Ensemble Simulations at Scale: Challenges

- **Resource Management** for O(10^{5-6}) tasks -- each is independent executing program!
  - Exascale ~O(10^{6-9})

Ensemble Simulations at Scale: Challenges

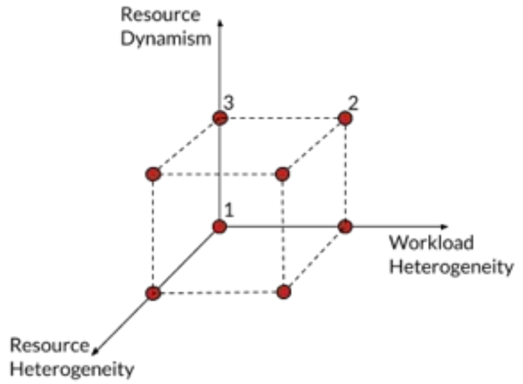
- **Resource Management** for O(10^{5-6}) tasks -- each is independent executing program!
  - Exascale ~O(10^{6-9})
- **Application requirements and resource performance must be dynamic**
  - Abstraction of static perf. is inadequate!
  - Implications on perf. portability & scaling

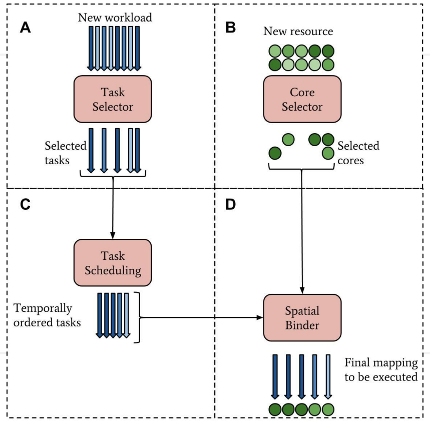




Ensemble Simulations at Scale: Challenges

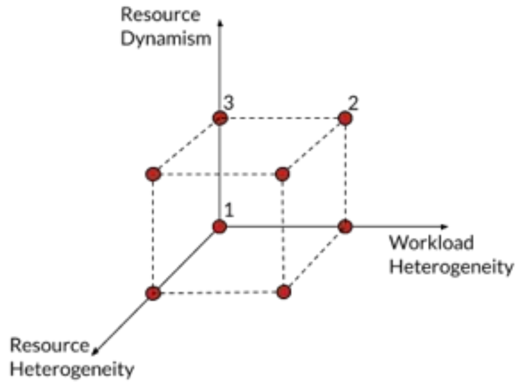
- **Resource Management** for O(10^{5-6}) tasks -- each is independent executing program!
  - Exascale ~O(10^{6-9})
- **Application requirements and resource performance must be dynamic**
  - Abstraction of static perf. is inadequate!
  - Implications on perf. portability & scaling
- **Execution Model** of heterogeneous tasks on heterogeneous and dynamic resources.
  - Early-binding: A->B->C->D
  - Late-binding: C->B->A->D

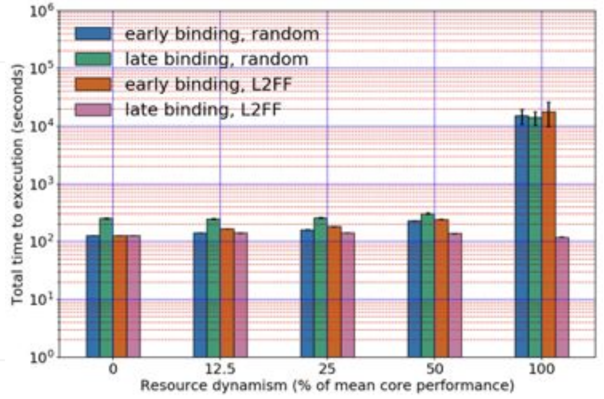




Ensemble Simulations at Scale: Challenges

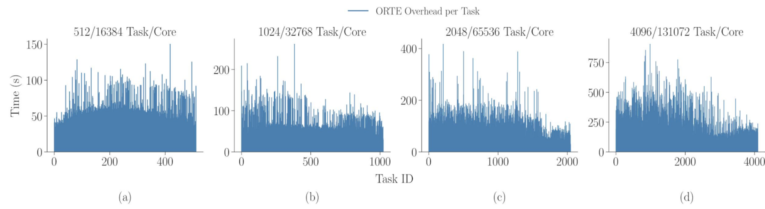
- **Resource Management** for O(10^{5-6}) tasks -- each is independent executing program!
  - Exascale ~O(10^{6-9})
- **Application requirements and resource performance must be dynamic**
  - Abstraction of static perf. is inadequate!
  - Implications on perf. portability & scaling
- **Execution Model** of heterogeneous tasks on heterogeneous and dynamic resources.
  - Early-binding: A->B->C->D
  - Late-binding: C->B->A->D

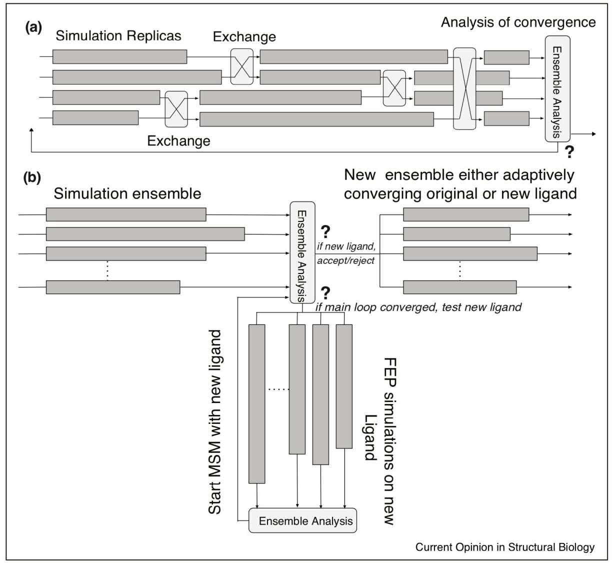




Ensemble Simulations at Scale: Challenges

- **Resource Management** for O($10^{5\text{-}6}$) tasks -- each is independent executing program!
  - Exascale ~O($10^{6\text{-}9}$)
- **Application requirements and resource performance must be dynamic**
  - Abstraction of static perf. is inadequate!
  - Implications on perf. portability & scaling
- **Execution Model** of heterogeneous tasks on heterogeneous and dynamic resources.
- **Adaptive Ensemble Algorithms:** Encoding algorithms that express adaptivity, even statistically (“approximately”)?
  - Managing interactions (coupling) between tasks
  - …..




(a)
Simulation Replicas
Exchange
Analysis of convergence
Ensemble Analysis
Exchange
?
(b)
Simulation ensemble
New ensemble either adaptively converging original or new ligand
Ensemble Analysis
?
if new ligand, accept/reject
?
if main loop converged, test new ligand
Start MSM with new ligand
FEP simulations on new Ligand
Ensemble Analysis
Current Opinion in Structural Biology

Outline

- Historical Perspective on Workflows

- Ensemble Computational Model
  - System & performance sensitive components versus user-facing component

- **Building Blocks for Middleware for Workflow System**
  - **RADICAL-Cybertools:** RADICAL-Pilot and Ensemble ToolKit (EnTK)
  - Performance vs Extensibility

Middleware Building Blocks for Workflow Systems

- Building Block Approach
  - Principled approach to the **architectural** design of middleware systems;
  - Applies traditional notion of modularity at the **software systems level**
  - Enable composability among **independent software systems**

- The four design principles:
  - **Self-sufficient:** Implements set of functionalities; not dependent 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

Middleware Building Blocks for Workflow Systems https://arxiv.org/abs/1903.10057

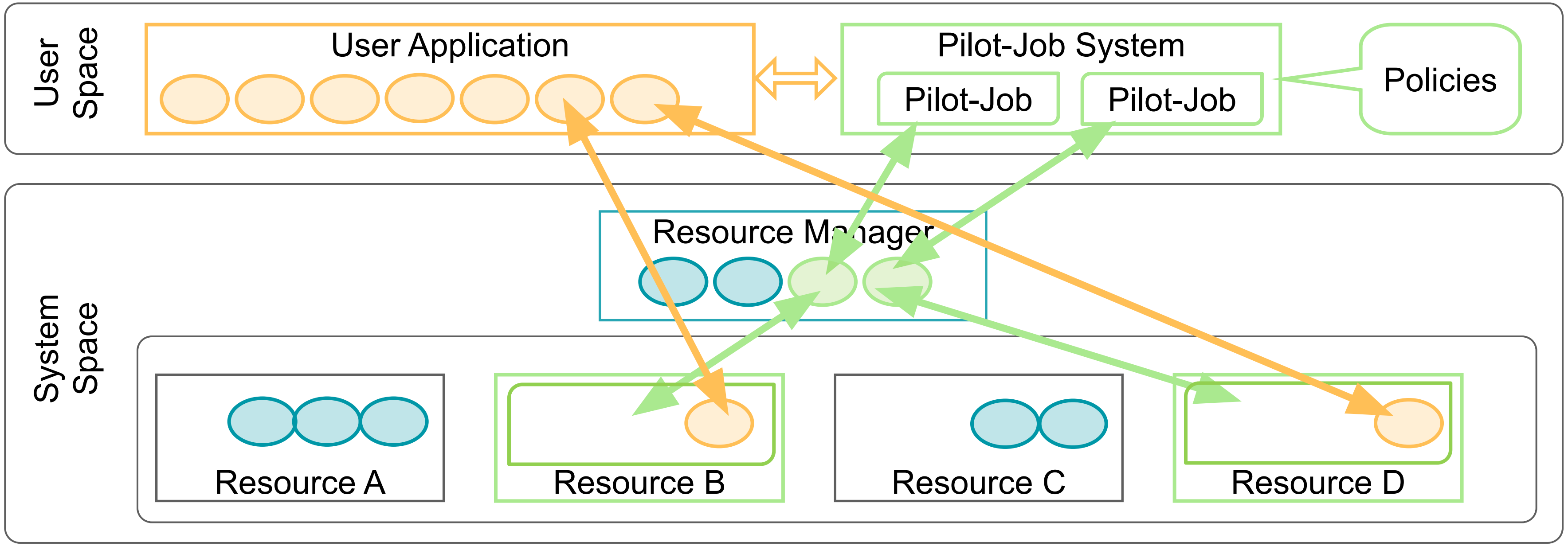
Building Blocks (BB): Properties

- A BB is a semantically well-defined independent software system, agnostic to coordination, and communication **patterns**, and exposed via an API.
  - **Stronger** (stringent) property than modularity
- BB have well defined state, event, and error models
  - Reduce challenge of composability of independent components
- BB work stand-alone, or integrated with other BB, or with 3rd party software
- Architecturally building blocks require:
  - Stable interfaces & distinction between computation and composition
  - Conversion layers -- multiple representation of the same entity

Pilot Abstraction: Schematic

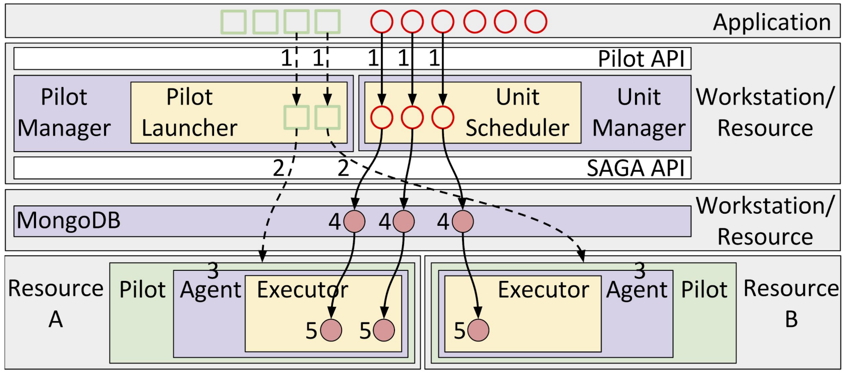
A system that allows application-level control of acquired resources via a scheduling overlay and a placeholder job.

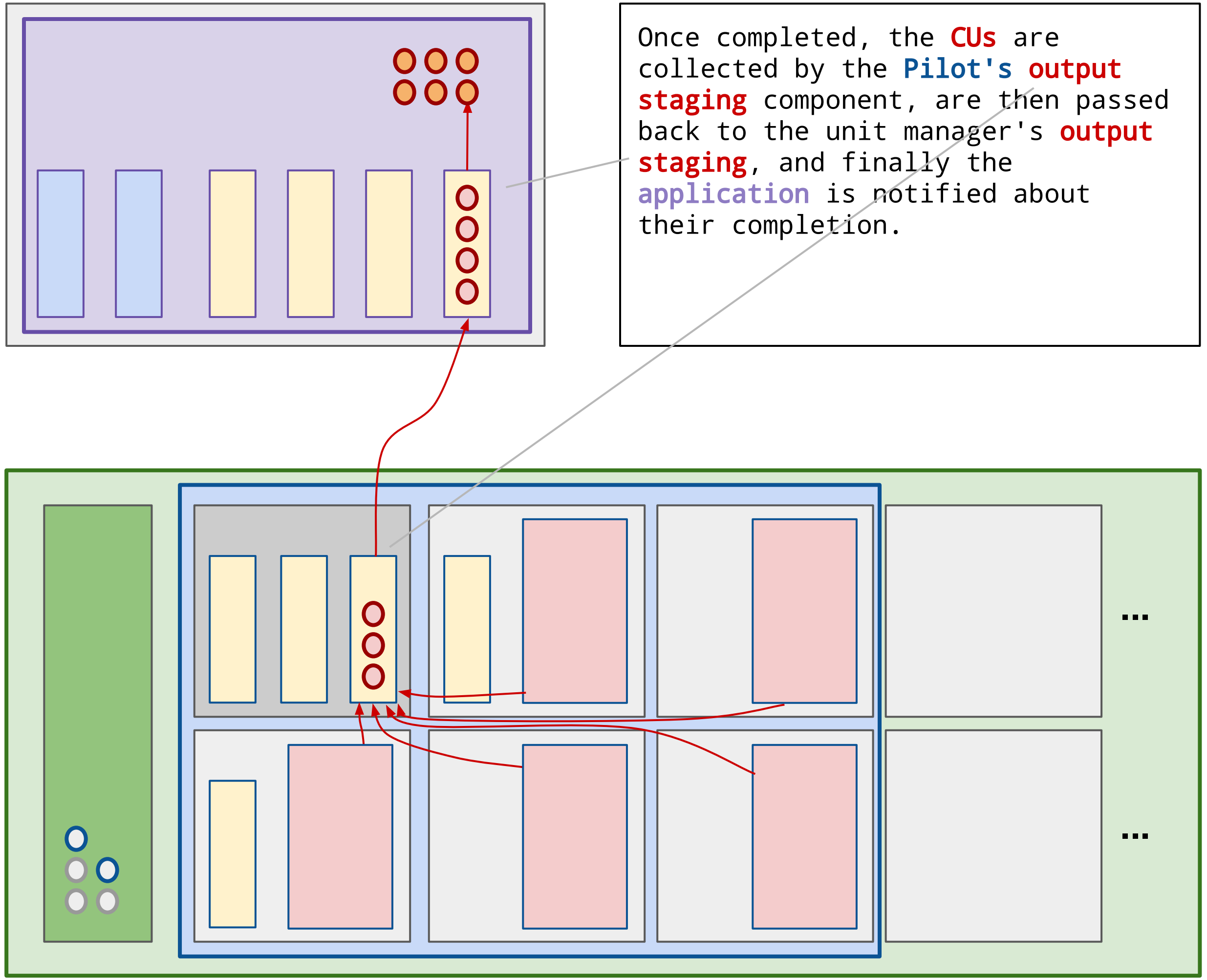
- Enables the fine-grained “slicing and dicing” of resources
- Provides **late-binding** of workloads to resources



Pilot-Abstraction: Recap

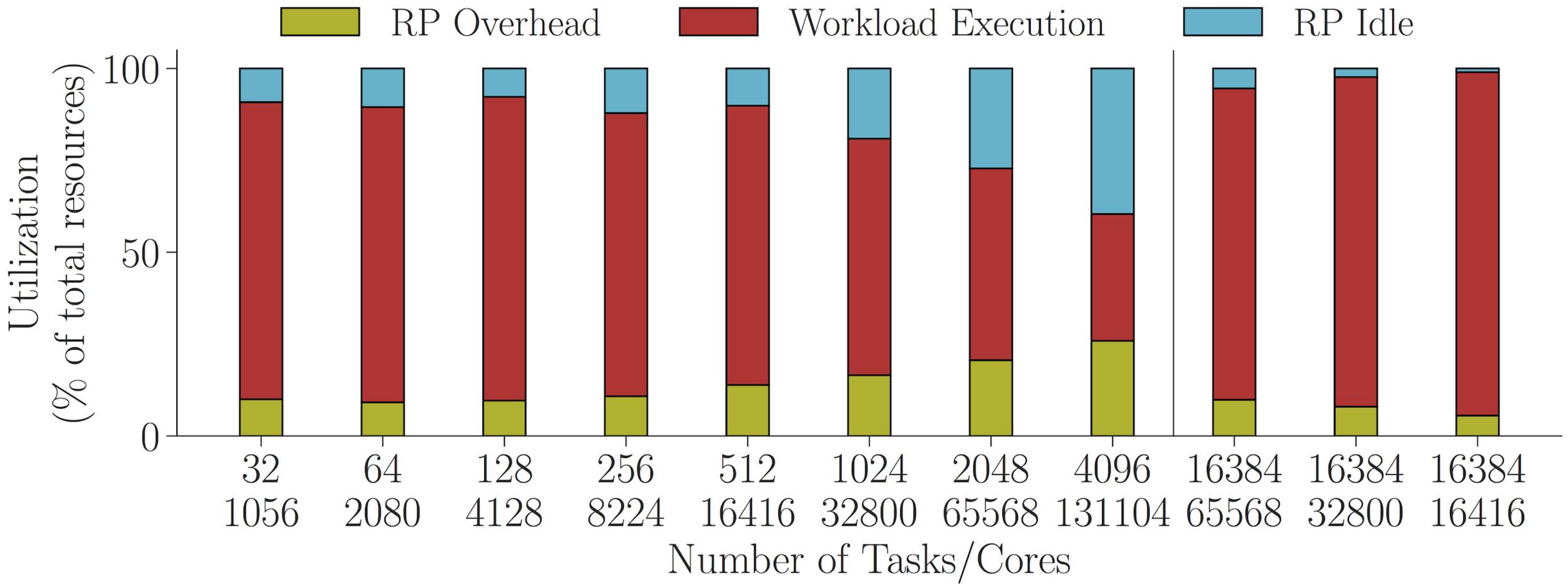
- Run multiple tasks **concurrently** and **consecutively** in a **SINGLE** batch job:
  - Tasks are programs, i.e., executables, not methods, functions, threads
  - Tasks executed within the batch job
- Late binding:
  - Tasks are NOT packaged into the batch job before submission.
  - Tasks are scheduled and then placed within the batch job at runtime.
- Task and resource heterogeneity:
  - Scheduling, placing and running CPU/GPU/OpenMM/MPI tasks





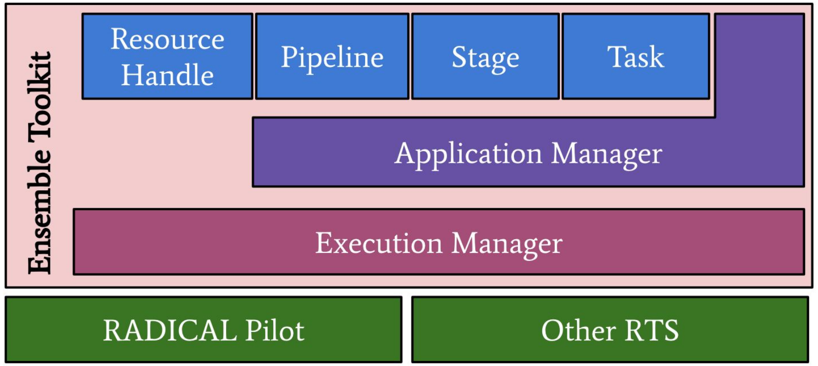
Once completed, the **CUs** are collected by the **Pilot's** **output staging** component, are then passed back to the unit manager's **output staging**, and finally the **application** is notified about their completion.

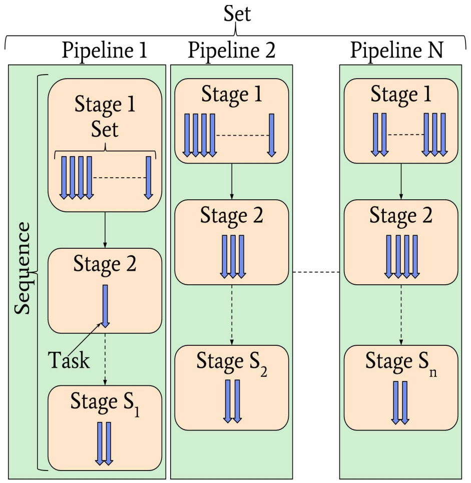
RADICAL-Pilot: Resource Utilization Performance (Titan)



EnTK: Building Block for Ensemble based Applications

- **Ensemble-Toolkit (EnTK):** Promote ensembles as a first-class programming and execution entity.
  - (i) Facilitate expression of ensemble based applications, (ii) manage complexity of resource acquisition, and (iii) 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"





Summary

- **Many advances in workflows, but many challenges in workflow systems**
  - Landscape is changing in many ways; also need focus on systems developers
- **RADICAL-Cybertools BB for Ensemble Computational Model**
  - RCT compatible with performance, extensibility and self-sufficient
  - Strong preference for functional specialization, as opposed to interoperability.
- **Community BB for Workflow Systems as components of "Open Workflow" ?**
- **BB Approach: Promise but many open questions**
  - *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?

Thank you!