UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

MONAD: Self-adaptive Micro-service Infrastructure for Heterogeneous Scientific Workflows

Phuong Nguyen and Klara Nahrstedt
University of Illinois at Urbana-Champaign

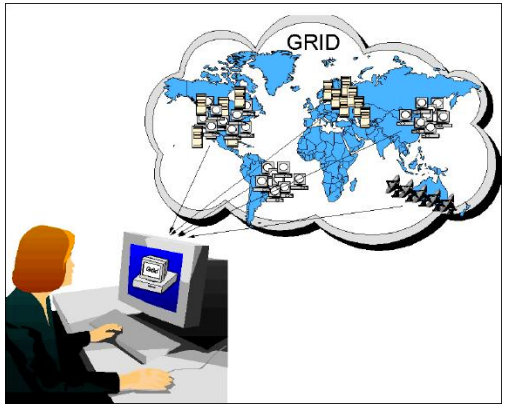
IEEE ICAC, 07/20/2017

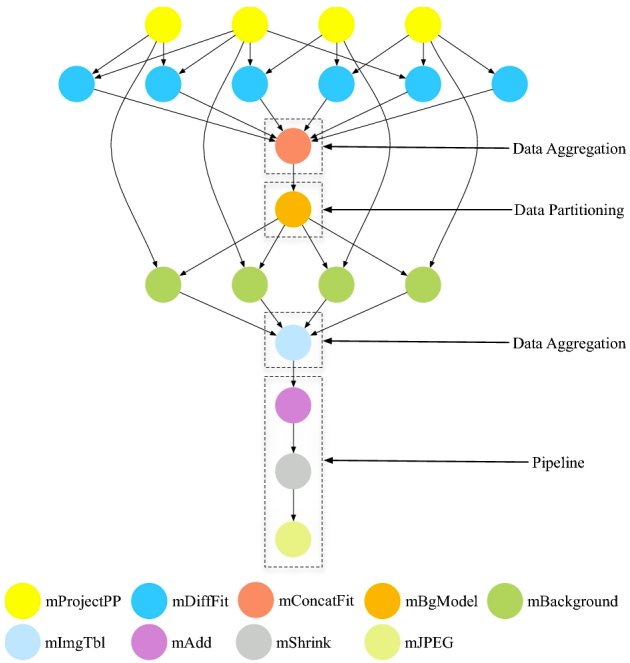
Outline

- **Background & Motivations**
- Our Solution: MONAD System
  - Execution Layer
  - Adaptation Layer
- Evaluation
- Conclusions & Future Work

Background: Scientific workflows

- Scientific workflow is a popular computational model for a variety of application domains
  - E.g.: Astronomy, biology, physics, and earth sciences
- Workflows are traditionally executed on cluster or grid computing infrastructure
  - Best-effort resource offerings

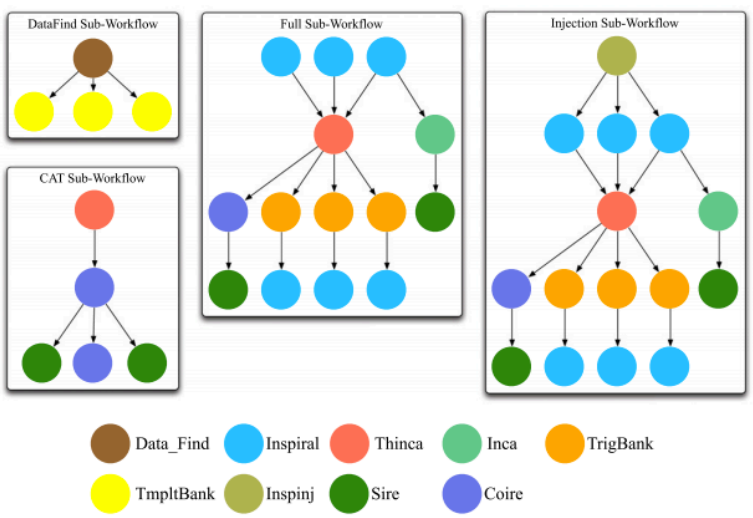




Montage workflow used by to combine multiple input images to create custom mosaics of the sky

Cloud-based scientific workflow system: Challenges & Opportunities

- Support heterogeneous workflows
- Support performance guarantees
  - E.g.: *"Average workflow processing time under 5s"*
  - Existing approaches often require advanced knowledge about workflow structures
- Advanced cloud technologies offer new resource abstraction
  - Existing scheduling approaches often deal with physical resources (e.g., CPU, RAM)

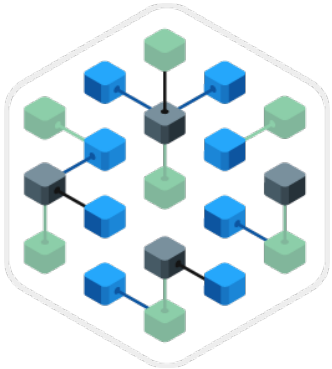


LIGO workflow ensemble

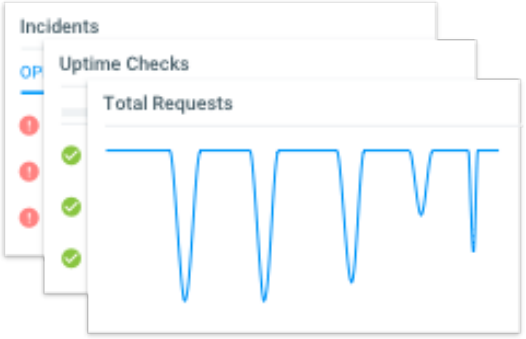
Long-tail scientific data
Source: Ferguson et al. *Nature neuroscience* 17.11 (2014)

Our approach

- We design MONAD* - Self-adaptive Micro-service Infrastructure for Heterogeneous Scientific Workflows



Micro-service execution environment



Non-intrusive workflow monitoring

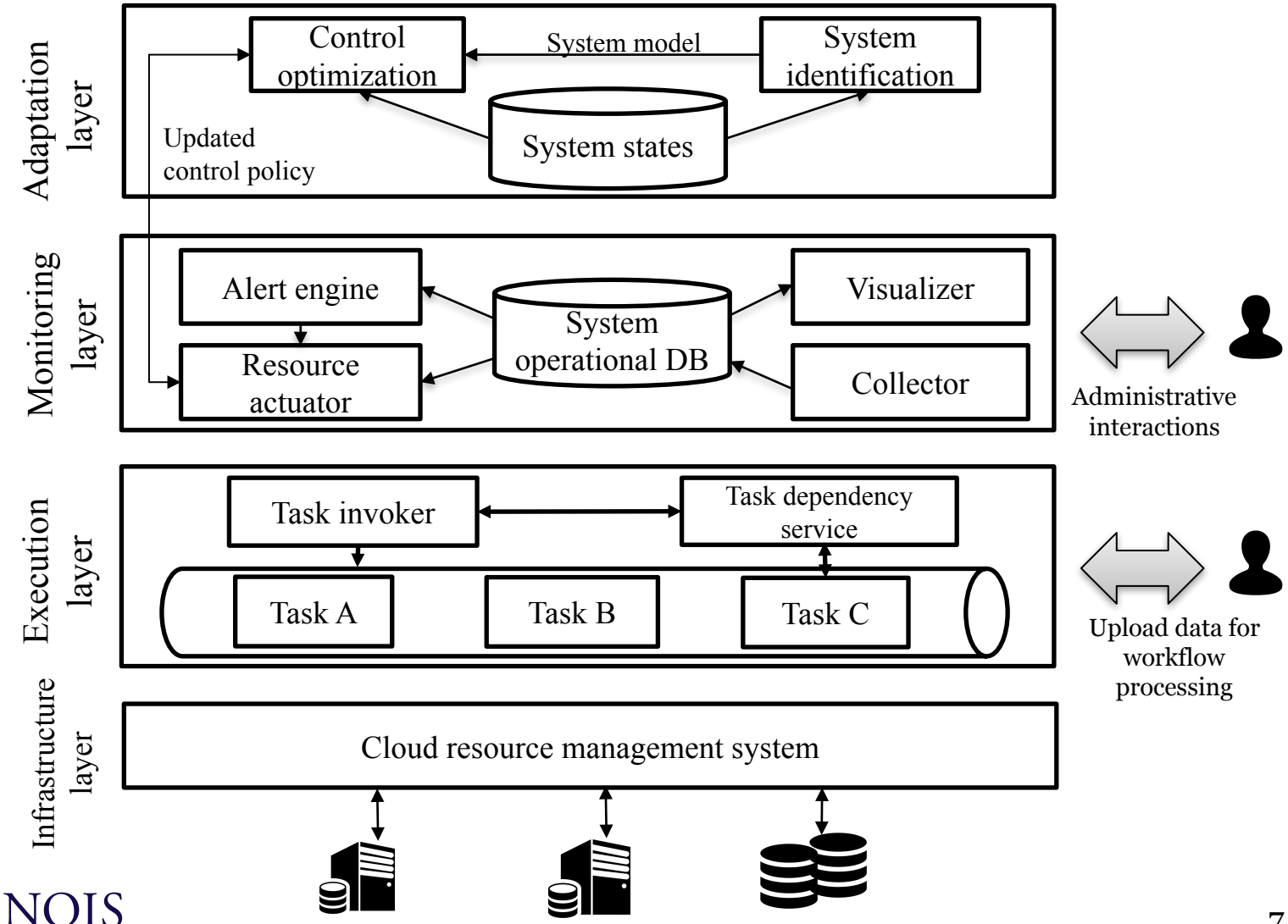
Feedback control resource adaptation

* MONAD stands for **MON**itoring and **AD**aptation. MONAD is also a concept used in functional programming to refer to "a way to build computer programs by joining simple components in robust ways" (*Wikipedia*).

Outline

- Background & Motivations
- **Our Solution: MONAD System**
  - Execution Layer
  - Adaptation Layer
- Evaluation
- Conclusions & Future Work

MONAD system architecture

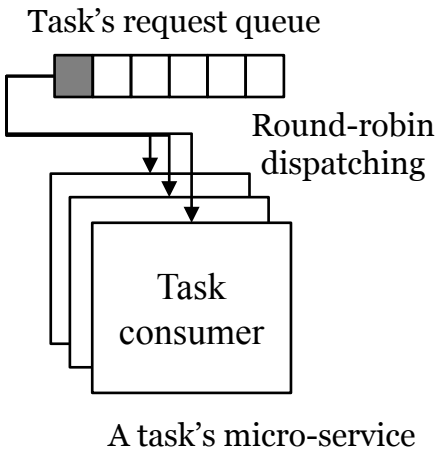


Outline

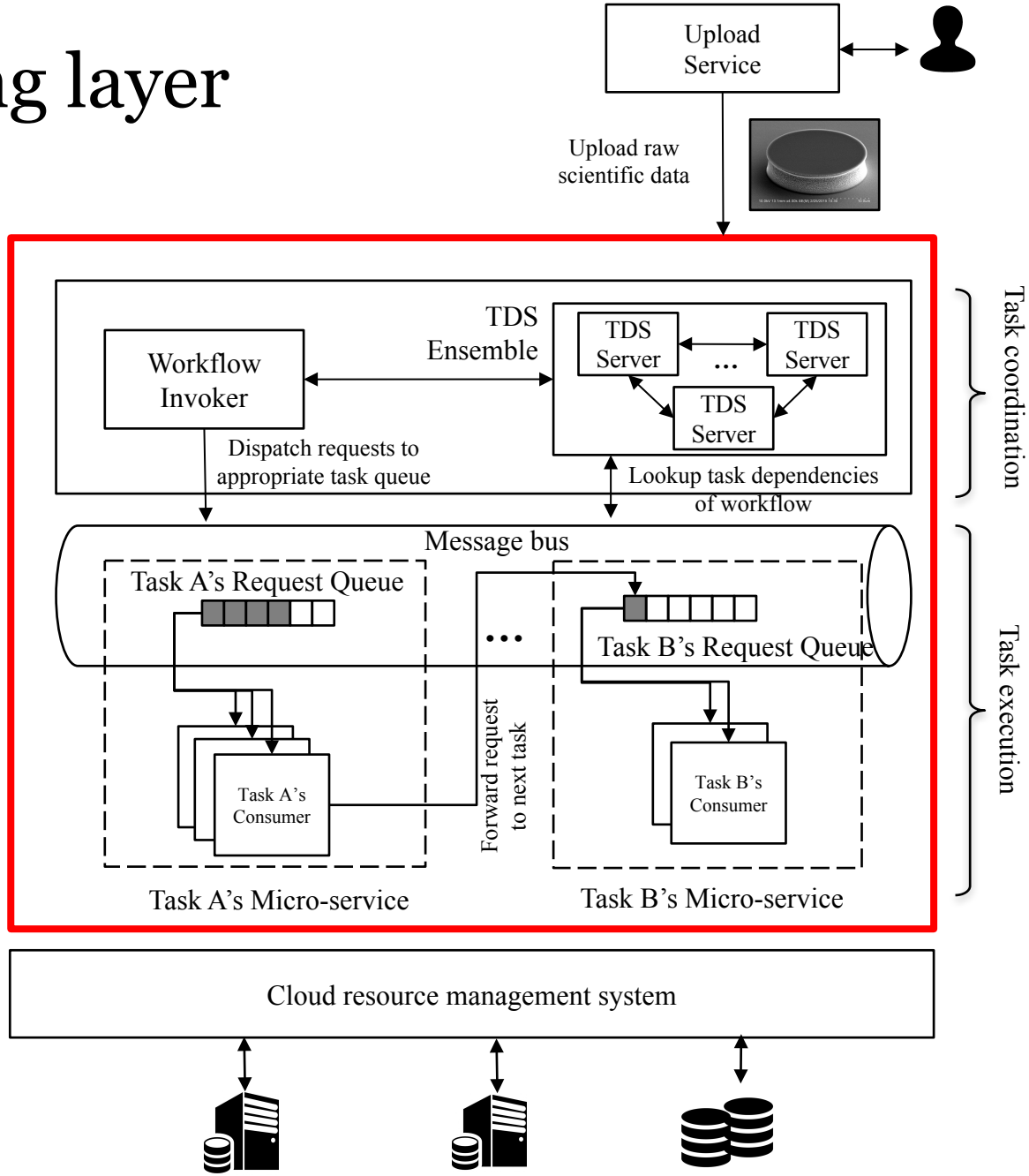
- Background & Motivations
- Our Solution: MONAD System
  - **Execution Layer**
  - Adaptation Layer
- Evaluation
- Conclusions & Future Work

Micro-service execution environment for scientific workflows

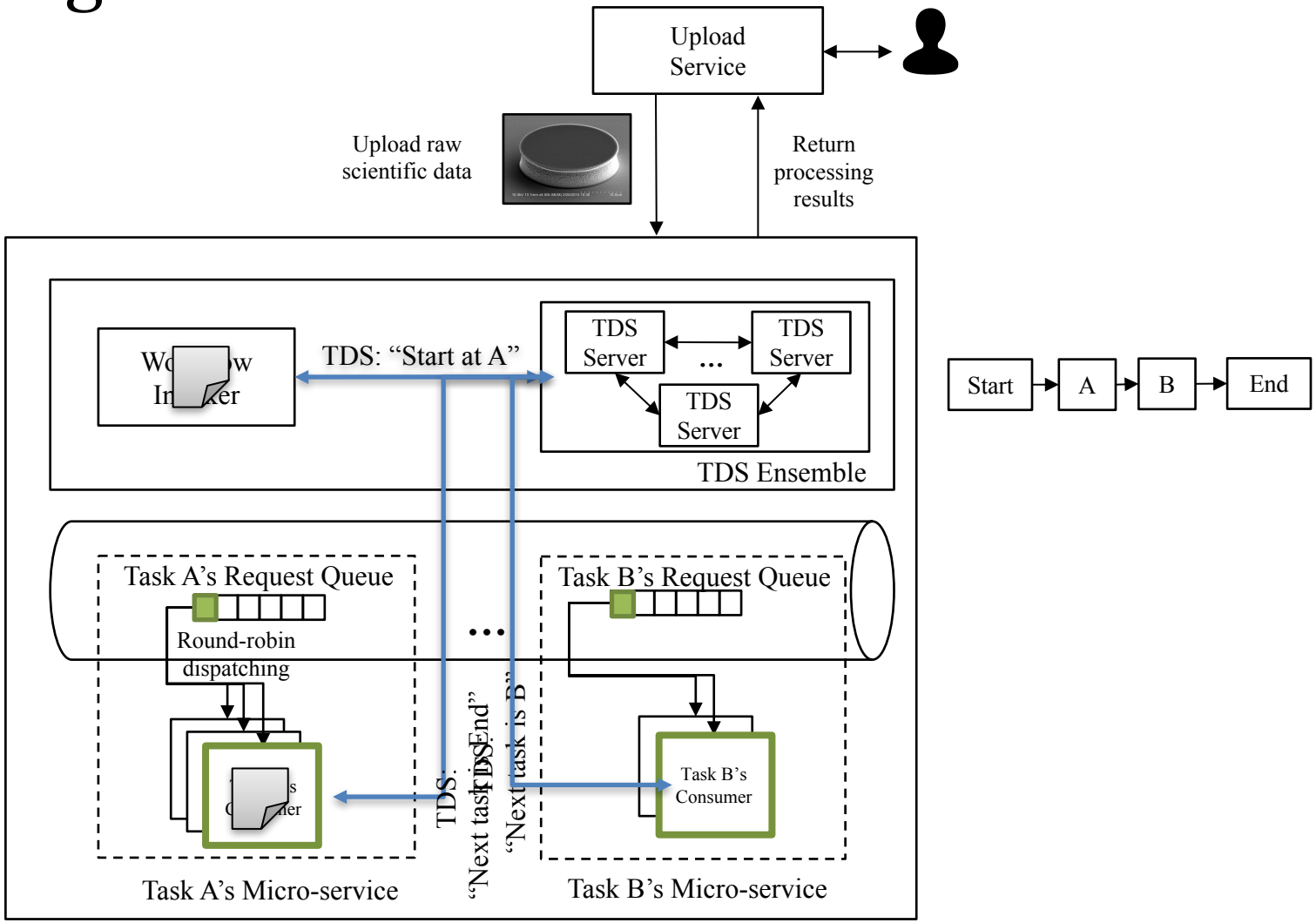
- Micro-services over monoliths
  - Building system from small, independent units of functionality that focus on doing one thing well
- Each task is modeled as a micro-service with its own request queue & servers
  - Use publish-subscribe middleware to connect micro-services
- Separate task dependencies from task implementation & deployment
  - Allow scalable workflow execution & coordination
  - Enable continuous deployment & flexible workflow composition



Executing layer



Example: Executing scientific data processing workflow

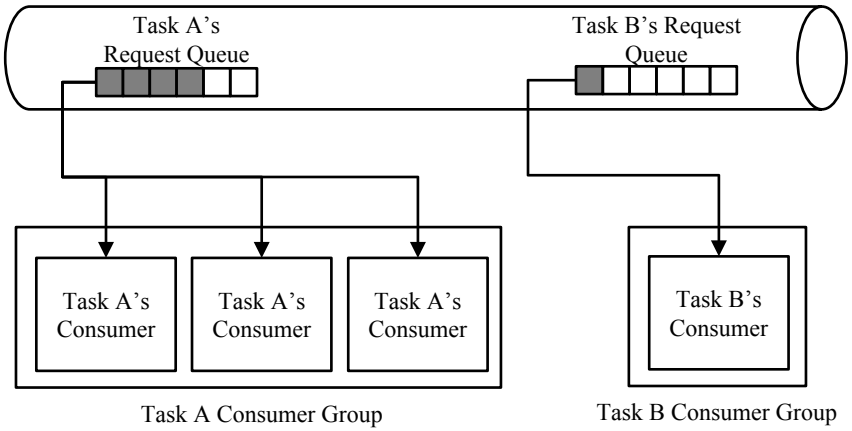


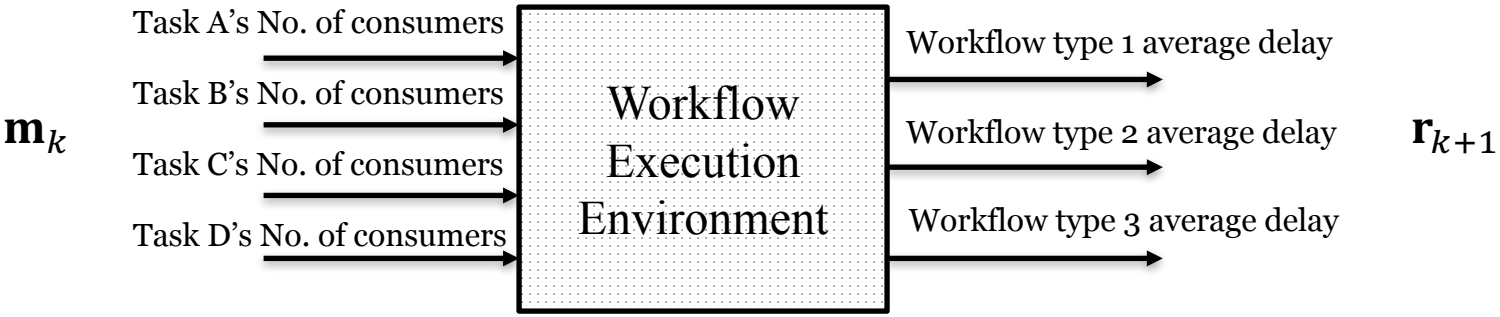
Outline

- Background & Motivations
- Our Solution: MONAD System
  - Execution Layer
  - **Adaptation Layer**
- Evaluation
- Conclusions & Future Work

Performance metrics & resource model

- Performance metrics:
  - Average delay of each workflow type & over all types
- Resource model:
  - Resources are represented by task consumers
  - Control inputs are numbers of consumers per task

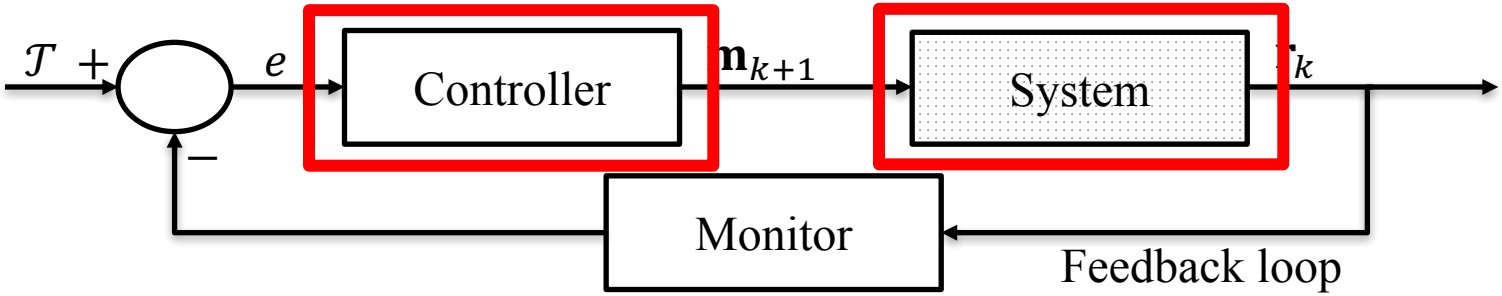




$\mathbf{m}_k$: Number of consumers over tasks $\mathbf{r}_{k+1}$: Avg. delay over workflow types

Feedback control-based resource adaptation

- Employ a control-theoretic approach in designing the adaptation layer

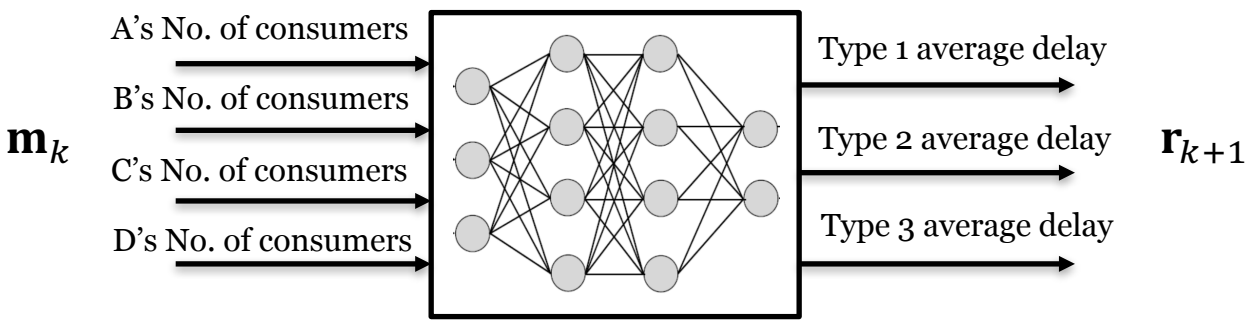


$\mathbf{m}_{k+1}$: Number of consumers over tasks	$\mathbf{r}_k$: Avg. delay over workflow types
$\mathcal{T}$: Reference performance	e: Difference between reference & real performance

System identification of workflow system

- ***Objective***: To derive a mathematical model of the system to predict system performance, given control inputs
- System identification design considerations:
  - *White-box* vs. *black-box* approach
  - Single predicted output vs. multiple predicted output

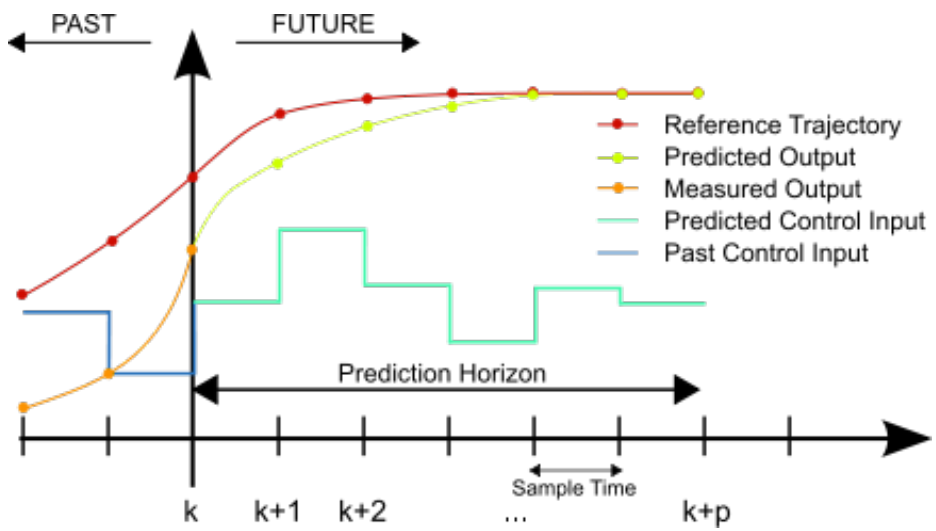
- We use *multilayer neural network* to model the system:
  - Proven approximation power
  - Successfully applied in identification of complex & dynamic systems



$$\mathbf{r}_{k+1} = \boldsymbol{f}(\mathbf{r}_k, \mathbf{m}_k)$$

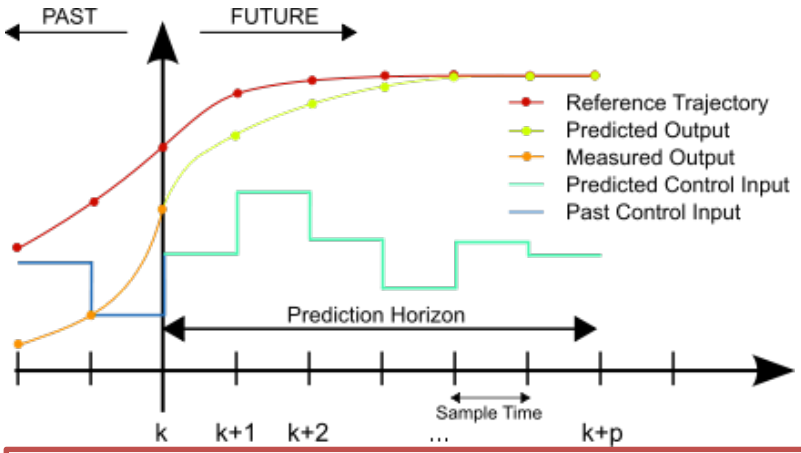
Controller design

- ***Objective***: To derive a controller that can produce control inputs to guide the system follows a desired output
- We use model predictive control (MPC) to design controller, given the learned system identifier



MPC illustration (source: *Wikipedia*)

Model predictive controller design



Minimize the aggregated cost over the horizon of next T time windows

$$\underset{\substack{\mathbb{M}=\{\mathbf{m}_\tau\} \\ k+1\leq\tau\leq k+\mathtt{T}}}{\text{argmin}} \sum_{\tau=k+1}^{k+\mathtt{T}} l(\mathbf{r}_\tau, \mathbf{m}_\tau)$$

subject to $\mathbf{r}_{\tau+1} = \boldsymbol{f}(\mathbf{r}_\tau, \mathbf{m}_\tau), k \leq \tau \leq k+\mathtt{T}-1$

Cost constraint

$$\sum_{j=1}^{J} m_\tau^j \leq \mathcal{C}, k+1 \leq \tau \leq k+\mathtt{T}$$

$$\mathbf{m}_\tau \in \mathbb{Z}_+^J, k+1 \leq \tau \leq k+\mathtt{T}$$

where $l(\mathbf{r}_\tau, \mathbf{m}_\tau)$ is defined as:

Instantaneous cost function at time τ

$$l(\mathbf{r}_\tau, \mathbf{m}_\tau) = \sum_{i=1}^{N} \lambda_i \cdot (\mathcal{T}_i - \mathbf{r}_\tau^i)^2 + \sum_{j=1}^{J} \mu_j \cdot (\Delta m_\tau^j)^2$$

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Model predictive controller design

- Model predictive control problem is a *constrained non-linear integer programming problem*
  - NP-hard problem

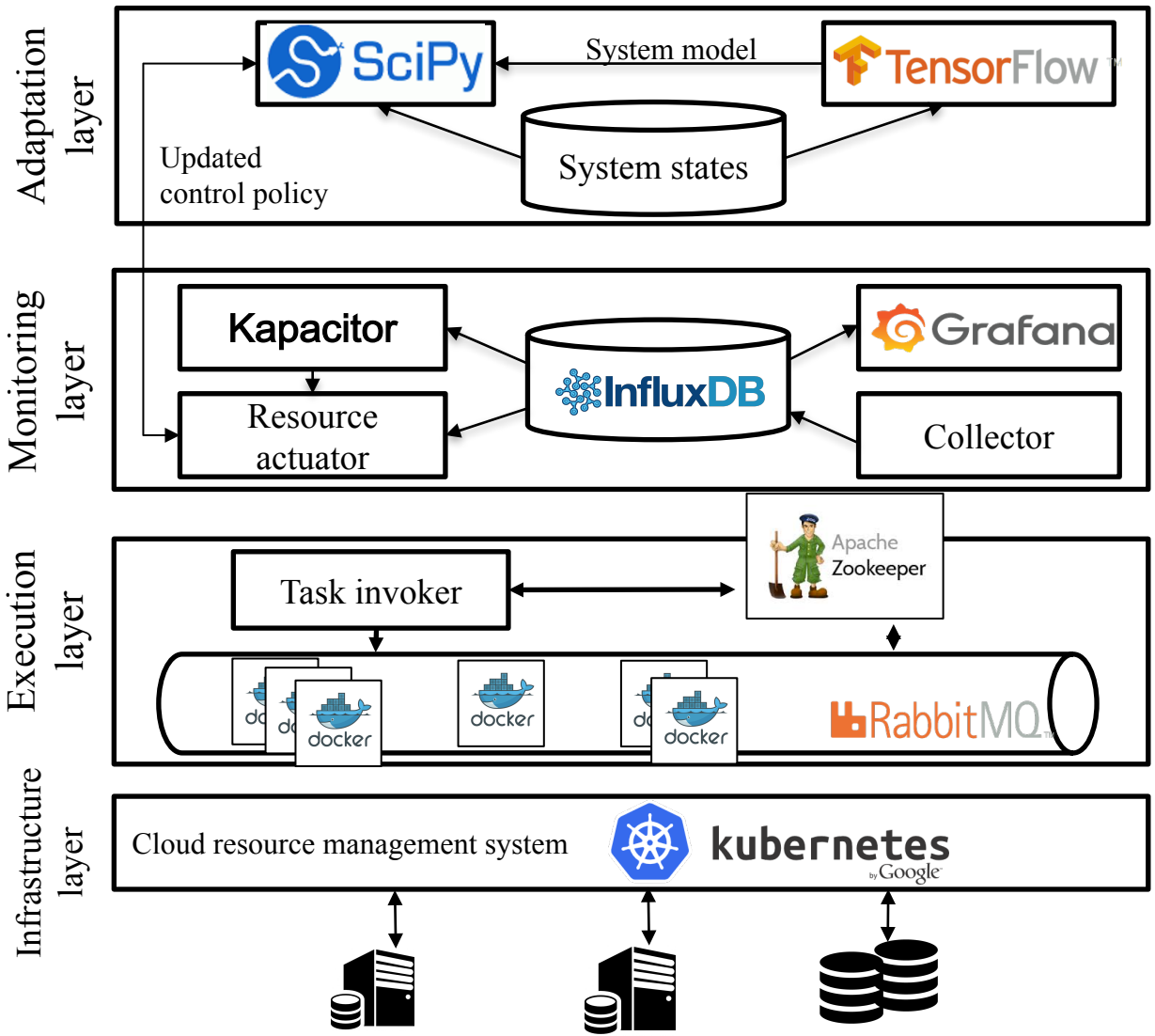
- Approximate solutions (`MPCAdapt`):
  - Relax the integrality constraint of control inputs
  - Solve the problem using *Sequential Least Squares Programming optimization algorithm (*an iterative method)
  - Non-integer solution then can be used to approximate the integer solution of the original problem

- Heuristic-based approach (`HeuristicAdapt`):
  - Only consider the next time window, instead of looking ahead T time windows

Outline

- Background & Motivations
- Our Solution: MONAD System
  - Execution Layer
  - Adaptation Layer
- **Evaluation**
- Conclusions & Future Work

MONAD system implementation



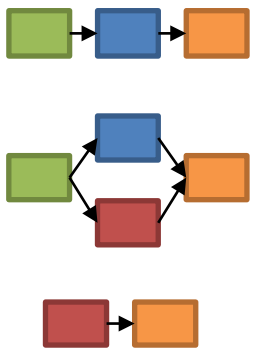
ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

MONAD system implementation

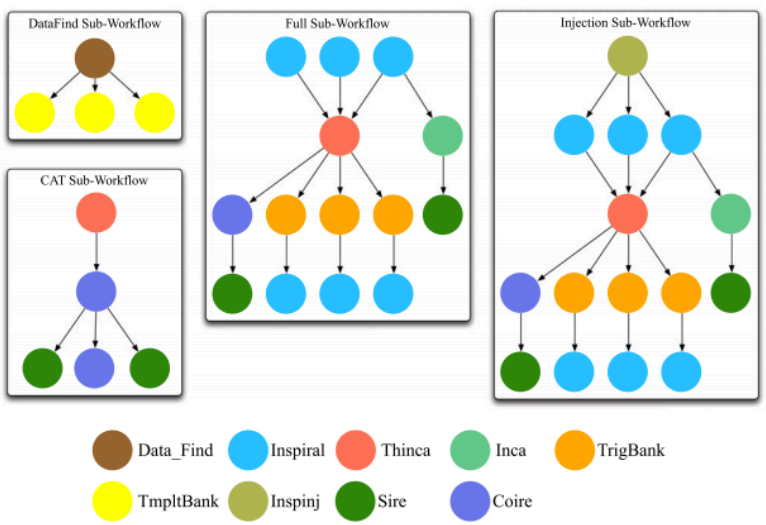
Monitoring layer's real-time dashboard

Evaluation settings

- Data processing workflows:
  - MDP: material data processing workflows (to process output of digital microscopy, such as DM3, AFM, etc.)
  - LIGO: analyze data to study stars and black holes



MDP workflows

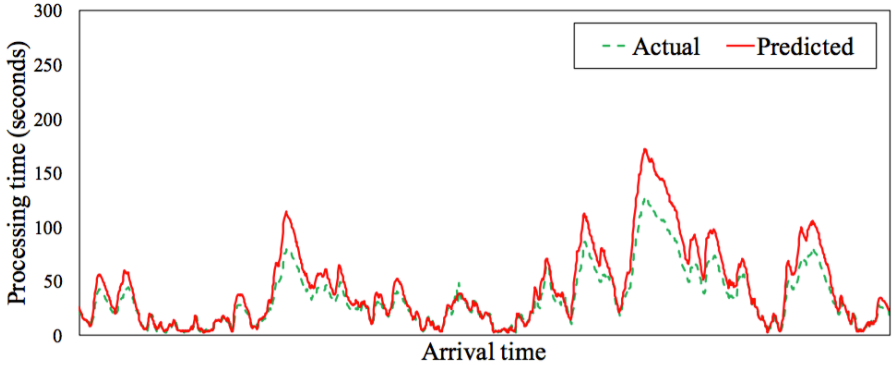


LIGO workflows

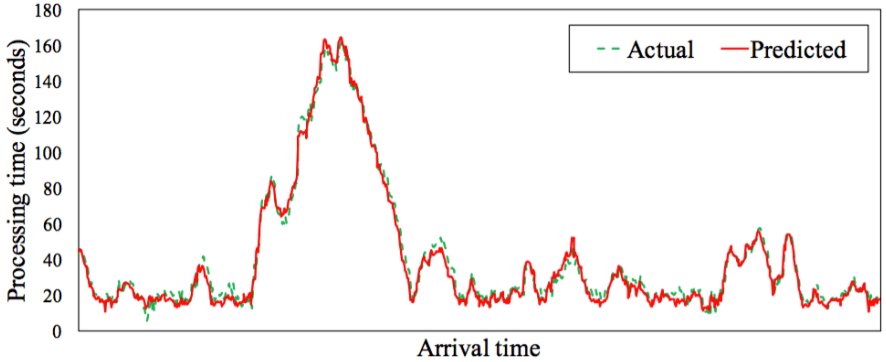
Evaluation settings – System identification

- Randomly generate a workload by varying the arrival rates of requests of different workflow types and vary the allocation of resources over tasks
  - A dataset of 60K requests of MDP and 15K requests of LIGO workflows
- Window length is set at 10s (to balance between the prediction accuracy and the data collection overhead)
- Neural network hyper-parameters:
  - 32 neurals in each hidden layer for MDP, 64 neurals for LIGO workflows
  - Learning rate as 0.001, batch size as 100, and we use 100 training epochs

Effectiveness of neural network-based system identification



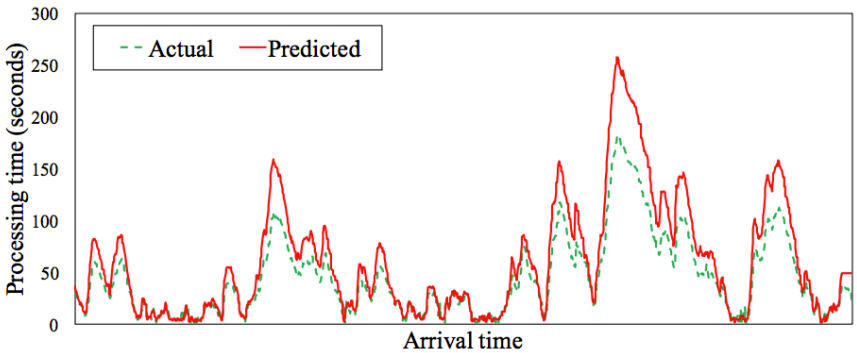
(a) MDP



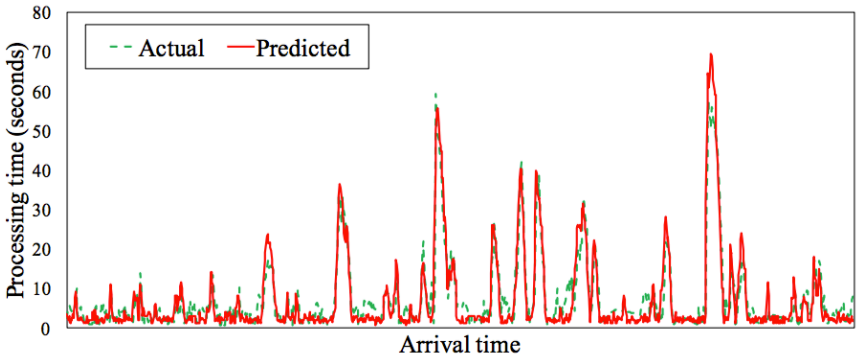
(b) LIGO

Neural network-based system identification can accurately predict average performance on different workflow ensembles

Effectiveness of neural network-based system identification



(a) MDP's DM3 file processing workflow



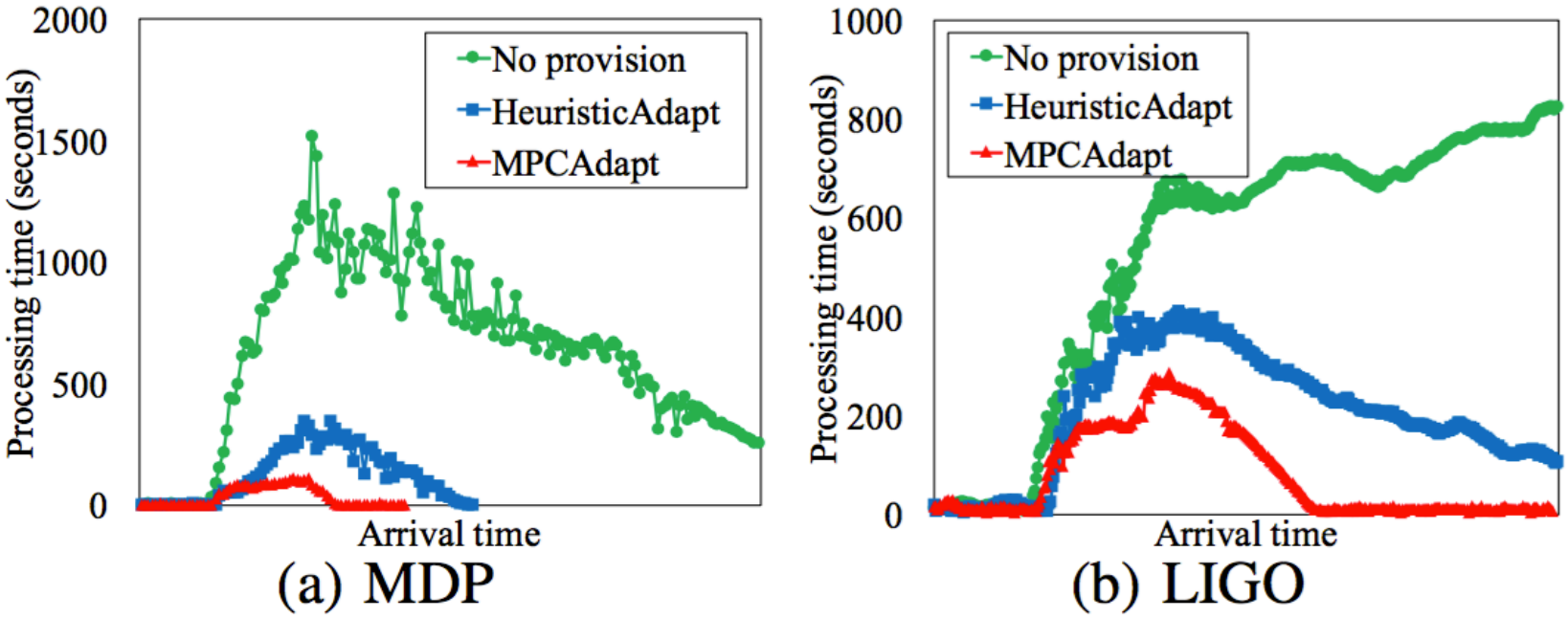
(b) MDP's experiment classification workflow.

Neural network-based system identification can accurately predict average processing time of individual MDP workflows

Evaluation settings – Resource adaptation

- Emulate the bursty workload situation by abnormally increasing the arrival rates of requests to up to 10 and 5 times higher on MDP and LIGO workflows
- Use absolute delay guarantee on average processing time: 10s and 30s for the MDP and LIGO workflows respectively
- Resource constraint of 15 and 90 maximum number of (homogeneous) consumers for the MDP and LIGO worklows

Effectiveness of feedback control-based resource adaptation



Feedback control-based adaptation algorithms demonstrate good effectiveness on adapting system performance when dealing with bursty workloads

Outline

- Background & Motivations
- Our Solution: MONAD System
  - Execution Layer
  - Adaptation Layer
- Evaluation
- **Conclusions & Future Work**

Conclusions & Future Work

- We present a novel system architecture of MONAD - a self-adaptive micro-service infrastructure for heterogeneous scientific workflows.
- We design *the first* feedback control-based resource adaptation mechanism for workflow system
  - Effective neural network-based system identification
- Future work:
  - Incorporate admission control, priority queue, multiple queues
  - Support scientific workflow-as-a-service model (i.e., ad-hoc workflow composition)

Thank you!

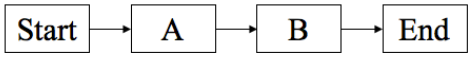
Back-up slides

Related work

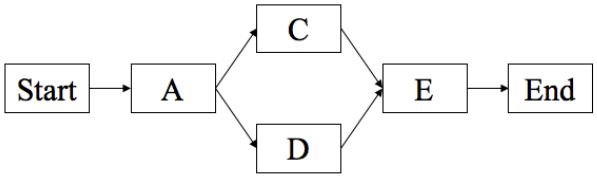
- Monolithic approach in workflow implementation & deployment
  - Pegasus, Taverna, Triana, Kepler: static translation to execution plan, require explicit placement of tasks on compute nodes
  - Shock/AWE: centralized coordination server
- Workflow scheduling focuses on task placement & ordering
  - Requires advanced knowledge of workflow structures
- Workflow execution environment:
  - VMs, virtual resources, RAM & CPU
- Workflow execution monitoring requires extra implementation efforts

Scalable task coordination service

- Maintain workflow's task dependencies & synchronization information between tasks
- Requirements:
  - Highly available
  - Fast response to queries
  - Scalable



Workflow type 1



Workflow type 2

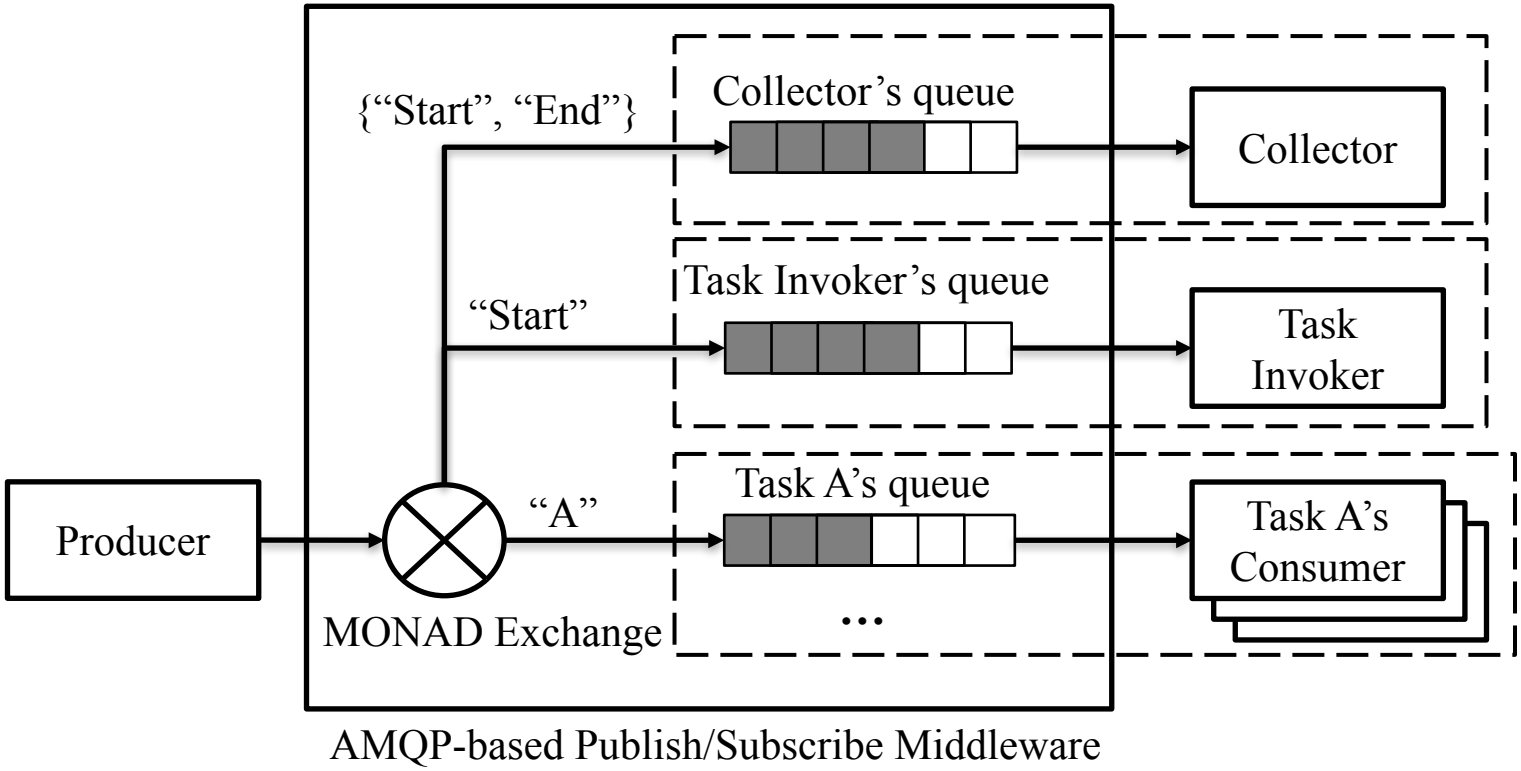
Type	From	To
1	Start	A
1	A	B
1	B	End
2	Start	A
2	A	C
2	A	D
2	C	E
2	D	E
2	E	End
…	…	…

Task dependencies

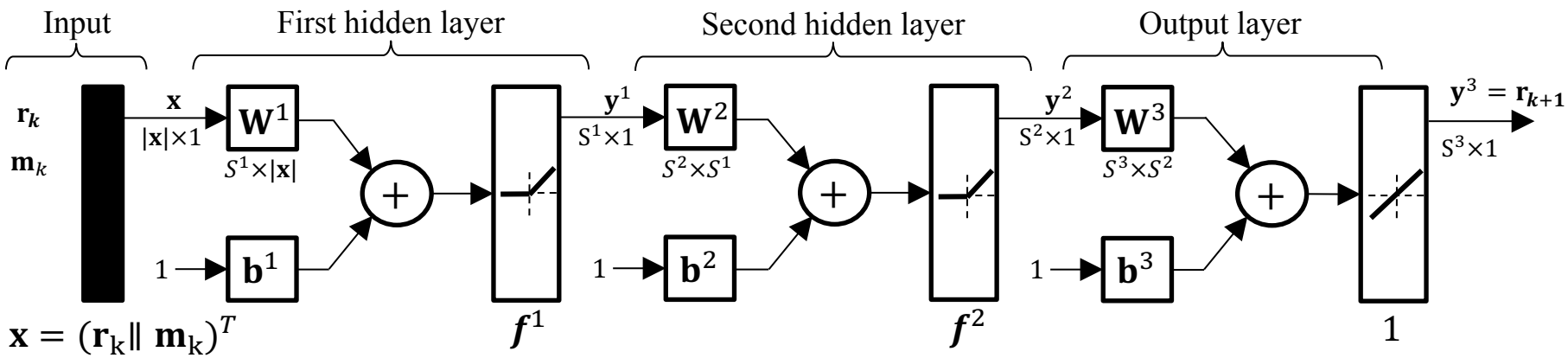
- Our approach:
  - Using an ensemble of multiple TDS servers, each maintain a replica of data
  - For *read* requests: Any of TDS servers can respond (for availability)
  - For *write* requests: Quorum-based write mechanism with leader election (for consistency)

Workflow and task monitoring

- Leverage the publish/subscribe middleware to monitor workflow performance without any *interference* and *any modification* to implementation of tasks



Neural network-based identification of workflow system



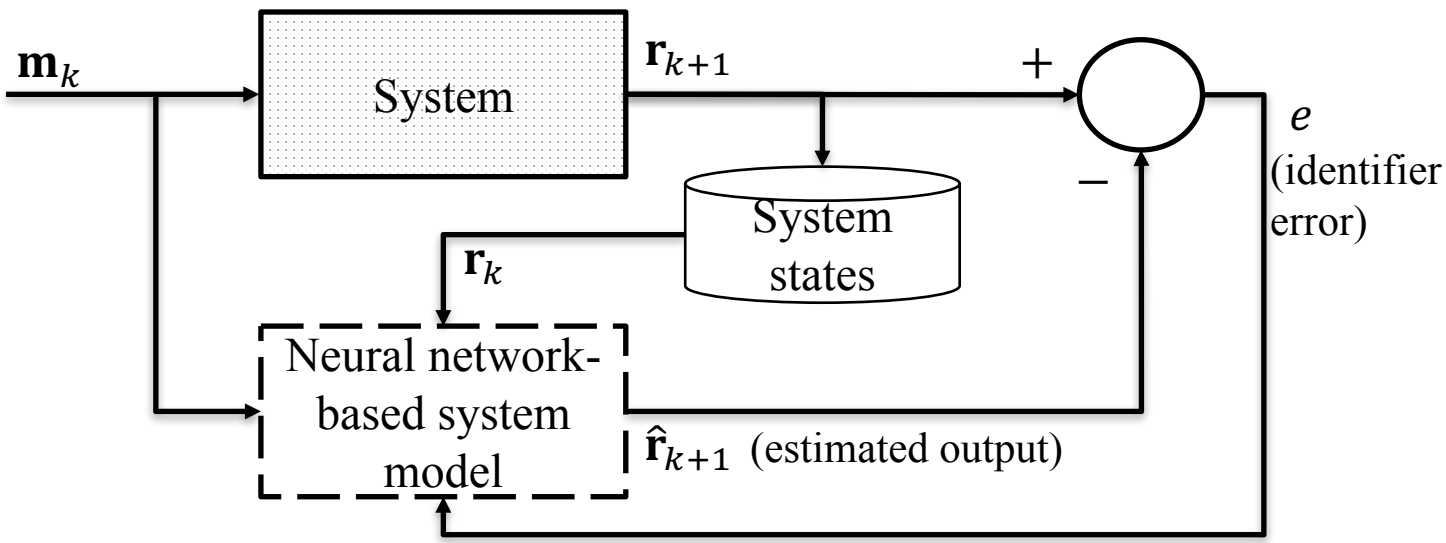
- Objective: To learn a function f

$$\begin{aligned}\mathbf{r}_{k+1} &= \boldsymbol{f}(\mathbf{r}_k, \mathbf{m}_k)\\ &= \mathbf{W}^3\boldsymbol{f}^2(\mathbf{W}^2\boldsymbol{f}^1(\mathbf{W}^1(\mathbf{r}_k \parallel \mathbf{m}_k)^T + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3\end{aligned}$$

- Neural network design:
  - Two hidden layers and one output layer
  - Use rectified linear unit (or ReLU) as the non-linear activation function

Training neural network-based system identifier

- Use backpropagation algorithm for training
- Training data obtained by using historical data, or by bootstrapping system with emulated workload



Model predictive control-based adaptation (`MPCAdapt`)

Algorithm 2 Model Predictive Control-based Adaptation

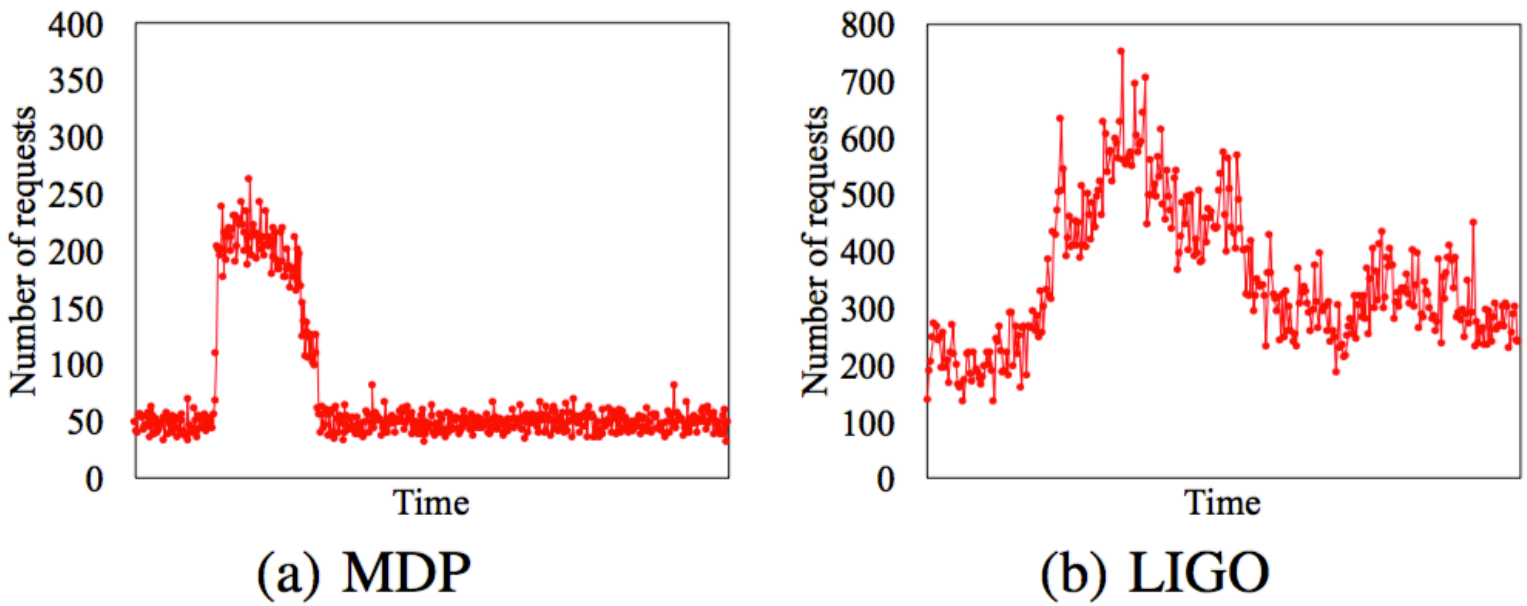
procedure $\texttt{MPCAdapt}(\mathbf{m}_k, \texttt{T}, \{\mathcal{T}_i\}, \mathcal{C})$
  $\mathbb{M} = \{\}$
  $\mathbb{C} = \{\}$
  for ι in $[0, \texttt{T} - 1]$ **do** *# Initialize* $\mathbb{M}$
    $\mathbb{M}.append(\mathbf{m}_k)$
  for ι from $[0, \texttt{T} - 1]$ **do** *# Add cost constraints*
    $\mathbb{C} = \mathbb{C} \cup \{\mathcal{C} - \texttt{sum}(\mathbb{M}[\iota]) \geq 0\}$
  for ι from $[0, \texttt{T} - 1]$ **do** *# Add positive resource constraints*
    $\mathbb{C} = \mathbb{C} \cup \{\mathbb{M}[\iota][j] \geq 0, \forall j \in [0, J - 1]\}$
  # Solve the optimization problem as described in (2):
  $\mathbb{M}^* = \texttt{minimize}(\texttt{mpc\_obj\_func}_{\{\{\mathcal{T}_i\}, \texttt{T}\}}, \mathbb{M}, \mathbb{C})$
  # Only return the first control move:
  Return $\mathbb{M}^*[0]$

Heuristic-based dynamic control algorithm (`HeuristicAdapt`)

Algorithm 3 Heuristic Adaptation Algorithm

procedure `HeuristicAdapt`$(\mathbf{m}_k, \mathbf{r}_k, \{\mathcal{T}_i\}, \mathcal{C})$
  while $\{\exists i \in [1, N] : \mathbf{r}_k^i > \mathcal{T}_i\}$ and $\sum_{j=1}^{J} m_k^j \leq \mathcal{C}$ **do**
    $cur\_min = -1.0$
    $j\_min = -1$
    for j from 1 to J **do**
      $m_k^j = m_k^j + 1$
      if `instant_cost`$(\mathbf{r}_k, \mathbf{m}_k) < cur\_min$ **then**
        $cur\_min =$ `instant_cost`$(\mathbf{r}_k, \mathbf{m}_k)$
        $j\_min = j$
      $m_k^j = m_k^j - 1$
    $m_k^{j\_min} = m_k^{j\_min} + 1$
  Return $\mathbf{m}_k$

Efficiency of task coordination service



TDS service demonstrates good efficiency when dealing with increasing requests to during the bursty workload (aggregated every 5 seconds). TDS maintains maximum latency of responses by of only 22ms for MDP and 37ms for LIGO workflows.