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

![Montage workflow diagram](image-url)
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)

Over 50% of scientific findings do not appear in the published literature

Our approach

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

* MONAD stands for MONitoring and ADaptation. 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

Adaptation layer
- Control optimization
- System model
- System identification
- Updated control policy

Monitoring layer
- Alert engine
- Resource actuator
- System operational DB
- Visualizer
- Collector

Execution layer
- Task invoker
- Task dependency service
- Task A
- Task B
- Task C

Infrastructure layer
- Cloud resource management system

Administrative interactions
- Upload data for workflow processing
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

- Task A's Request Queue
- Task B's Request Queue
- Task A's Consumer
- Task B's Consumer
- Task A's Micro-service
- Task B's Micro-service
- Cloud resource management system

Workflow Invoker
- Dispatch requests to appropriate task queue
- TDS Ensemble
- Lookup task dependencies of workflow

Message bus
- Forward request to next task

Upload raw scientific data

TDS Server
- ...
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

\[ \mathbf{m}_k: \text{Number of consumers over tasks} \quad \mathbf{r}_{k+1}: \text{Avg. delay over workflow types} \]
Feedback control-based resource adaptation

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

\[ \mathcal{T} + e \rightarrow \text{Controller} \xrightarrow{m_{k+1}} \text{System} \]

\[ \text{Monitor} \xrightarrow{\text{Feedback loop}} \]

\( m_{k+1} \): Number of consumers over tasks
\( r_k \): 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

\[ r_{k+1} = f(r_k, m_k) \]
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:

$$\arg\min_{M = \{m_\tau\}} \sum_{\tau = k + 1}^{k + T} l(r_\tau, m_\tau)$$

subject to:

$$r_{\tau + 1} = f(r_\tau, m_\tau), \, k \leq \tau \leq k + T - 1$$

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

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

where $l(r_\tau, m_\tau)$ is defined as:

$$l(r_\tau, m_\tau) = \sum_{i=1}^{N} \lambda_i \cdot (T_i - r_\tau^i)^2 + \sum_{j=1}^{J} \mu_j \cdot (\Delta m_\tau^j)^2$$

Cost constraint

Instantaneous cost function at time $\tau$
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

Adaptation layer

System model

Updated control policy

System states

Monitoring layer

Kapacitor

Resource actuator

InfluxDB

Collector

Execution layer

Task invoker

Apache Zookeeper

Infrastructure layer

Cloud resource management system

kubernetes

cloud resource management system

monad system implementation
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
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

Neural network-based system identification can accurately predict average performance on different workflow ensembles.
Effectiveness of neural network-based system identification

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 workflows
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
- 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

![Workflow and task monitoring diagram](image)
Neural network-based identification of workflow system

- Objective: To learn a function $f$

$$r_{k+1} = f(r_k, m_k)$$

$$= W^3 f^2(W^2 f^1(W^1 (r_k \parallel m_k)^T + b^1) + b^2) + b^3$$

- 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

\[ e = \text{identifier error} \]

\[ r_{k+1} = \text{estimated output} \]

\[ m_k \rightarrow \text{System} \]

\[ r_k \rightarrow \text{Neural network-based model} \]

\[ r_k \rightarrow \text{System states} \]

\[ e = \text{identifier error} \]
Model predictive control-based adaptation (MPCAdapt)

Algorithm 2 Model Predictive Control-based Adaptation

1: procedure MPCAdapt(mₖ, T, {Tᵢ}, C)
2:     M = {}
3:     C = {}
4:     for i in [0, T - 1] do # Initialize M
5:         M.append(mₖ)
6:     for i from [0, T - 1] do # Add cost constraints
7:         C = C ∪ {C − sum(M[i]) ≥ 0}
8:     for i from [0, T - 1] do # Add positive resource constraints
9:         C = C ∪ {M[i][j] ≥ 0, ∀j ∈ [0, J - 1]}
10:    # Solve the optimization problem as described in (2):
11:    M* = minimize(mpc_obj_func({Tᵢ}, τ), M, C)
12:    # Only return the first control move:
13:    Return M*[0]
Heuristic-based dynamic control algorithm (HeuristicAdapt)

Algorithm 3 Heuristic Adaptation Algorithm

1: procedure HeuristicAdapt($m_k$, $r_k$, $\{T_i\}$, $C$)
2: while $\exists i \in [1, N]: r_k^i > T_i$ and $\sum_{j=1}^{J} m_k^j \leq C$ do
3:     $cur\_min = -1.0$
4:     $j\_min = -1$
5:     for $j$ from 1 to $J$ do
6:         $m_k^j = m_k^j + 1$
7:         if instant\_cost($r_k$, $m_k$) < $cur\_min$ then
8:             $cur\_min = \text{instant\_cost}(r_k, m_k)$
9:             $j\_min = j$
10:        end if
11:    end for
12:    $m_k^j_{\_min} = m_k^j_{\_min} + 1$
13: return $m_k$
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.