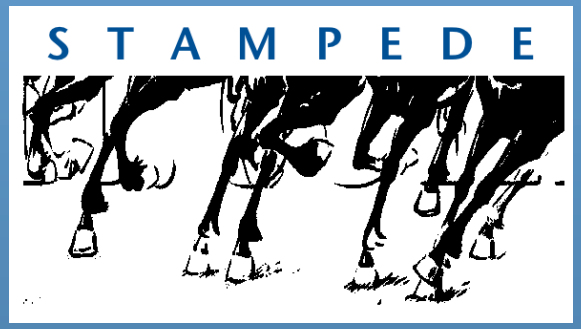


A General Approach to Real-time Workflow Monitoring

Karan Vahi , Ewa Deelman, Gaurang Mehta, Fabio Silva

USC Information Sciences Institute

Ian Harvey, Ian Taylor, Kieran Evans, Dave Rogers, Andrew Jones, Eddie El-Shakarchi

School of Computer Science, Cardiff University

Taghrid Samak, Dan Gunter, Monte Goode

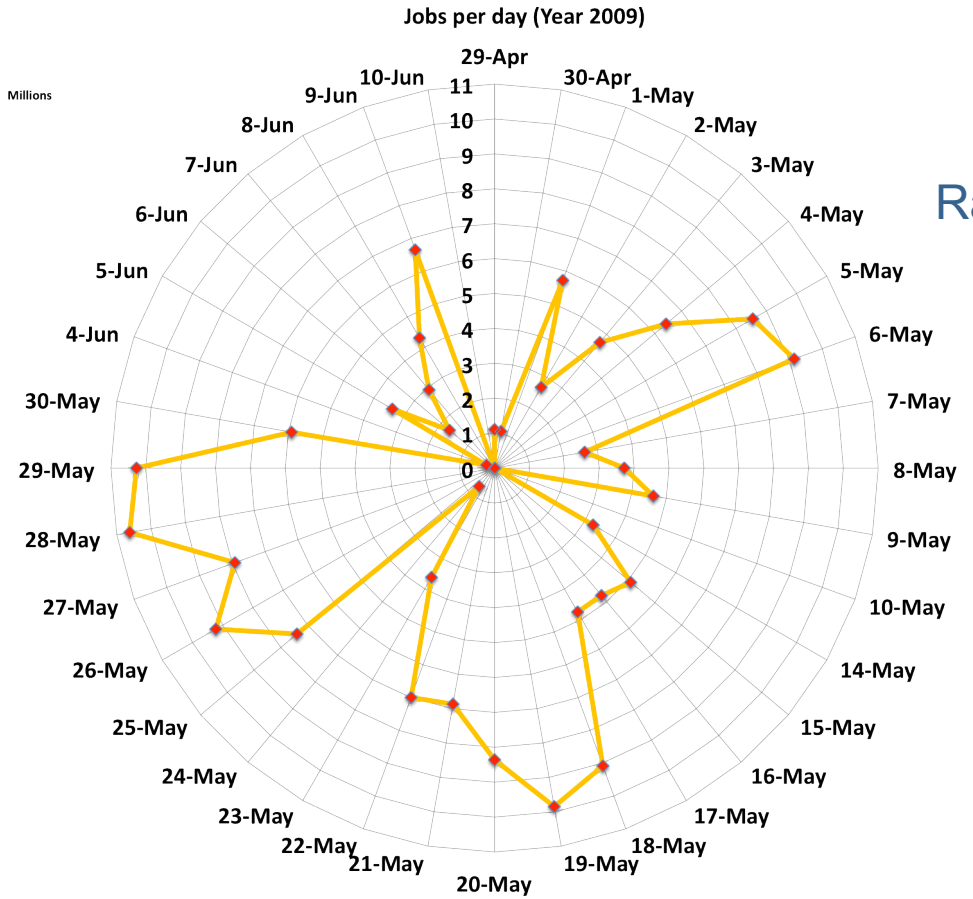
Lawrence Berkeley National Laboratory

Outline

- Background
- Stampede Data Model
- Triana and Stampede Integration
- Experiments and Analysis Tools
- Conclusions and Future Work

Domain: Large Scientific Workflows

SCEC-2009: Millions of tasks completed per day



Radius = 11 million

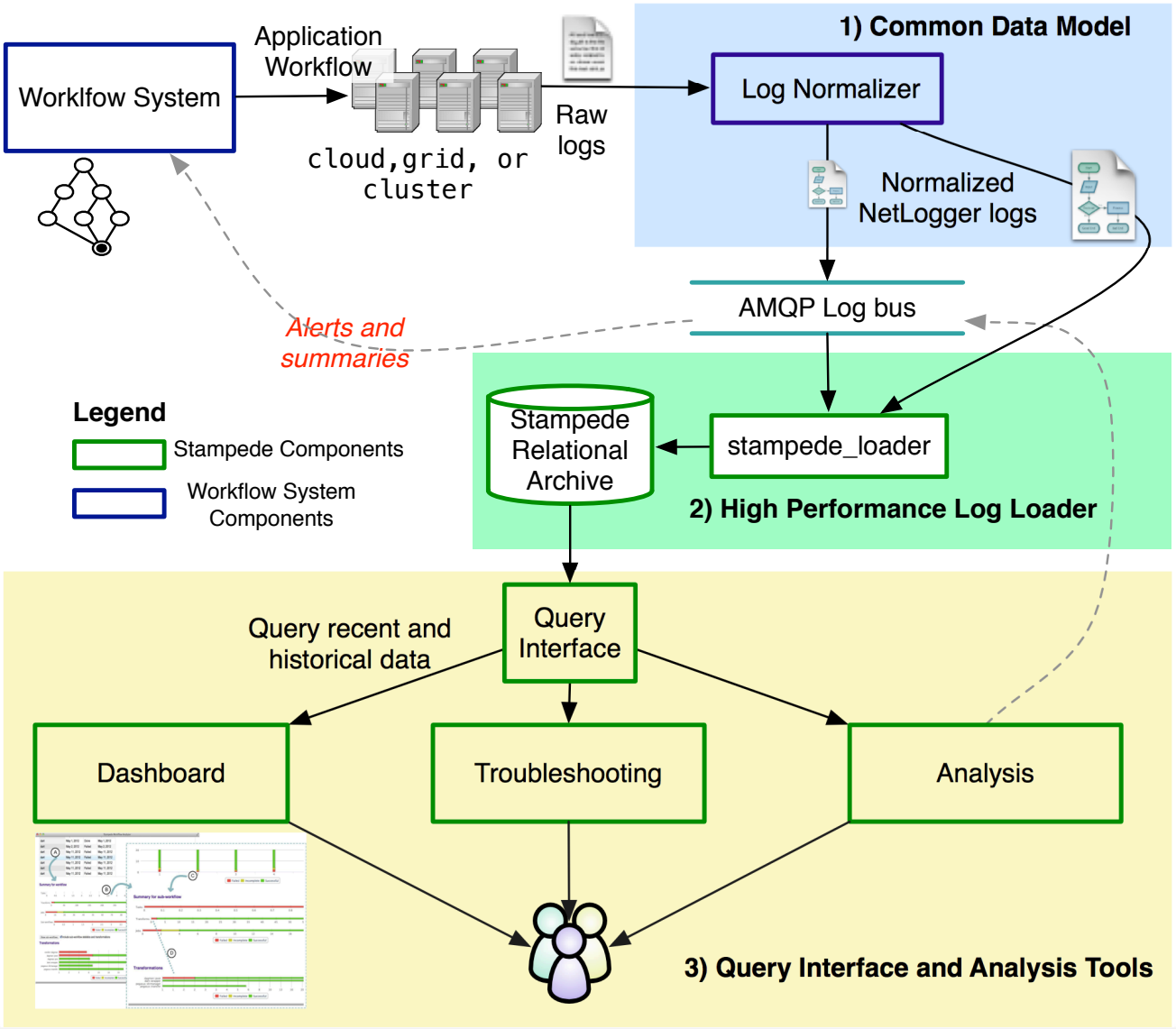
Goal: Real-time Monitoring and Analysis

1. Monitor Workflows in real time
   - Scientific workflows can involve many sub-workflows and millions of individual tasks
   - Need to correlate across workflow and job logs
   - Provide realtime updates on the workflow – how many jobs completed, failed etc
2. Troubleshoot Workflows
   - Provide users with tools to debug workflows, and provide information of why a job failed
3. Visualize Workflow performance
   - Provide a workflow monitoring dashboard that shows the various workflows run
4. Provide Analysis tools
   - Is a given workflow going to "fail"?
   - Are specific resources causing problems?
   - Which application sub-components are failing?
   - Is the data staging a problem?
5. **Do all of this as generally as possible: Can we provide a solution that can apply to all workflow systems?**

Outline

- Background
- Stampede Data Model
- Triana and Stampede Integration
- Experiments and Analysis Tools
- Conclusions and Future Work

How Does Stampede Provide Interoperability



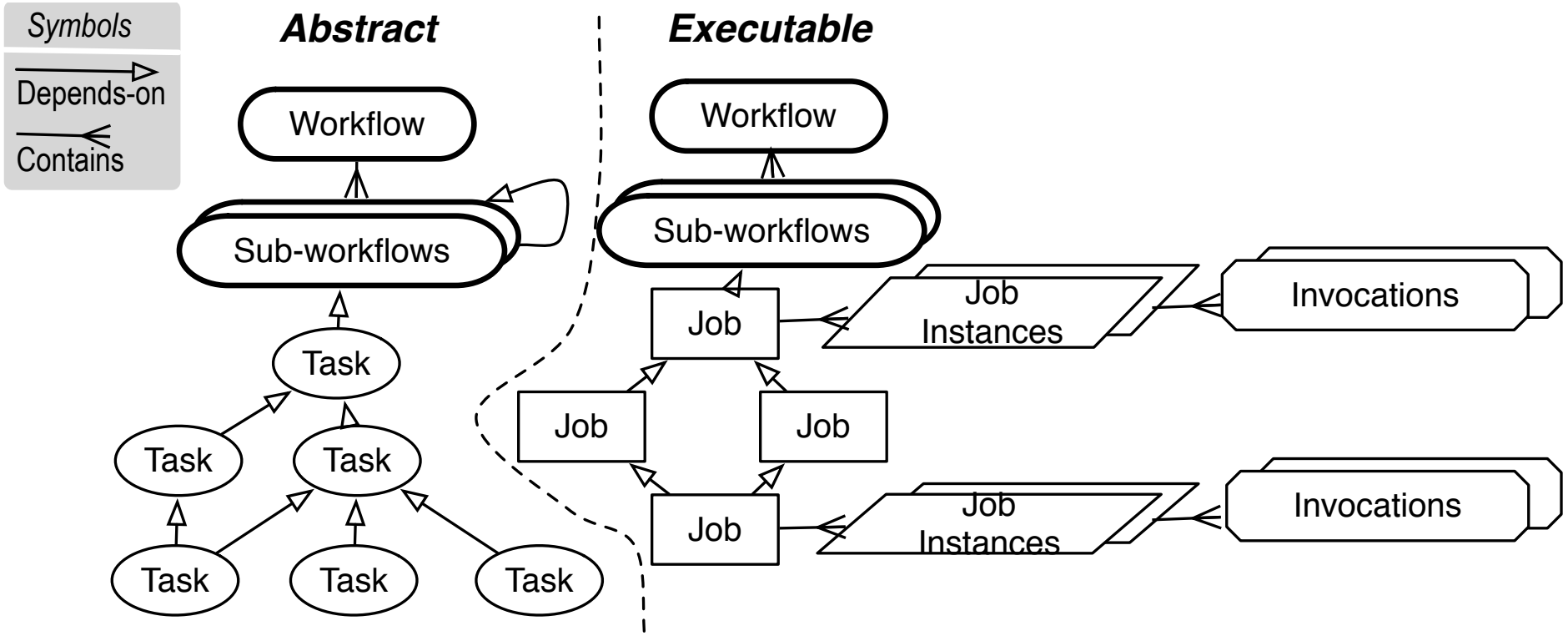
Abstract and Executable Workflows

- Workflows start as a resource-independent statement of computations, input and output data, and dependencies
  - This is called the Abstract Workflow (AW)
- For each workflow run, workflow systems may plan the workflow, adding helper tasks and clustering small computations together
  - This is called the Executable Workflow (EW)
- Note: Most of the logs are from the EW but the user really only knows the AW. The model allows us to connect jobs in the user specified (AW) with the jobs in EW executed through Workflow Systems

Entities in Stampede Data Model

- **Workflow**: Container for an entire computation
- **Sub-workflow**: Workflow that is contained in another workflow
- **Task**: Representation of a computation in the AW
- **Job**: Node in the EW
  - May represent one or more tasks in the AW. Or can represent jobs added by Workflow System (e.g., a stage-in/out),
- **Job instance**: Job scheduled or running by underlying system
  - Due to retries, there may be multiple job instances per job
- **Invocation**: captures actual invocation of an executable on
  - When a job instance is executed on a node, one or more invocations can be associated. The invocations capture the runtime execution of tasks specified in the AW

Relationship between Entities in Stampede Data Model



Logs Normalization

- Logging Methodology
  - Workflow Systems generate logs in the netlogger format
    - Timestamped, named, messages at the *start* and *end* of significant events, with additional *identifiers* and metadata in a std. line-oriented ASCII format (Best Practices or BP)
    - APIs are provided

Example Log Message

```
ts=2012-03-13T12:35:38.000000Z event=stampede.xwf.start
level=Info xwf.id=ea17e8ac-02ac-4909-b5e3-16e367392556
restart_count=0
```

- Yang schema to describe the events in netlogger format
  - YANG schema documents and validates each log event

http://acs.lbl.gov/projects/stampede/4.0/stampede-schema.html

Snippet of schema

```
container stampede.xwf.start {
  description “Start of executable workflow”;
  uses base-event;
  leaf restart_count {
    type uint32;
    description  "Number of times workflow was restarted (due to
failures)”; }}
```

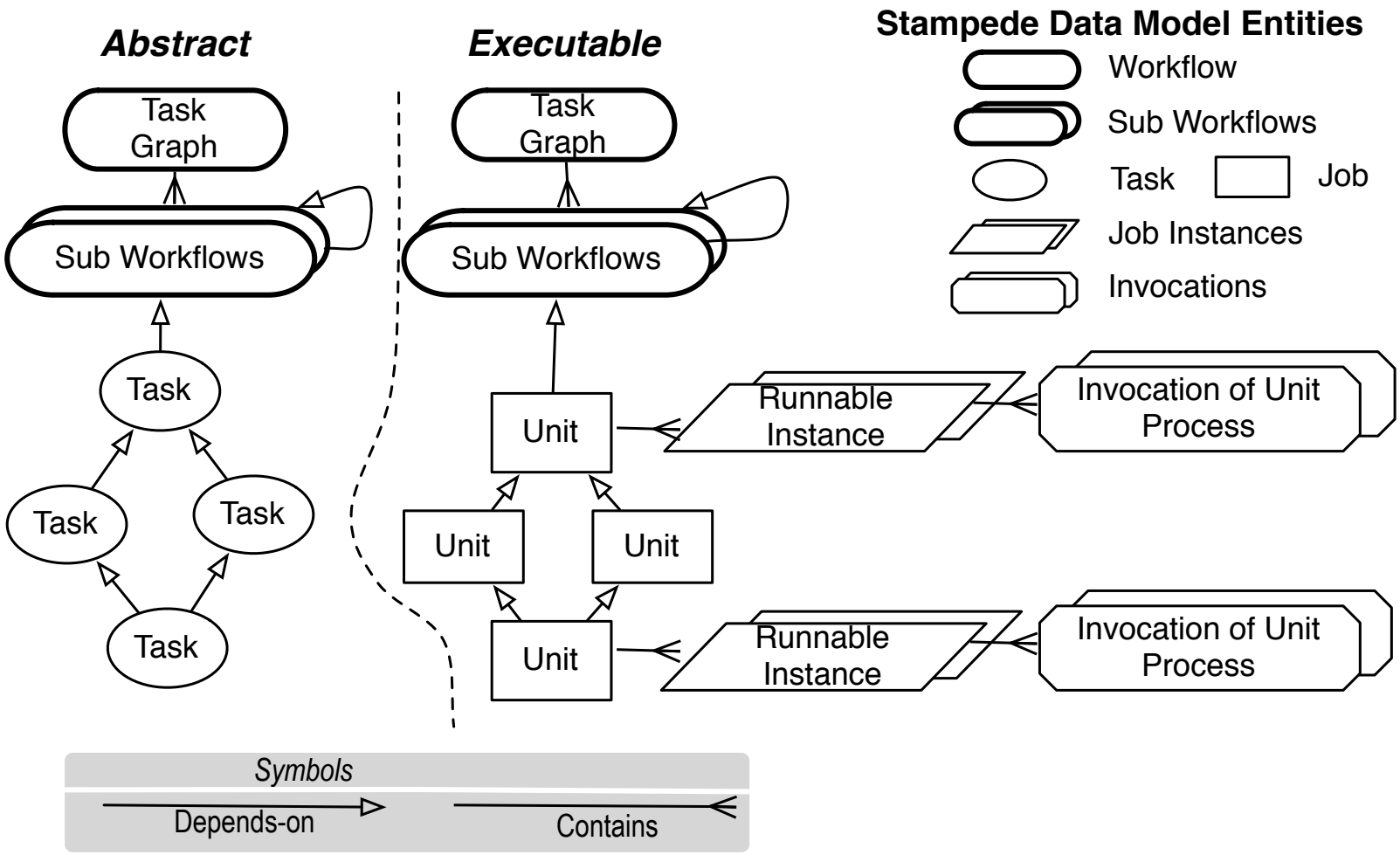
Outline

- Background
- Stampede Data Model
- Triana and Stampede Integration
- Experiments and Analysis Tools
- Conclusions and Future Work

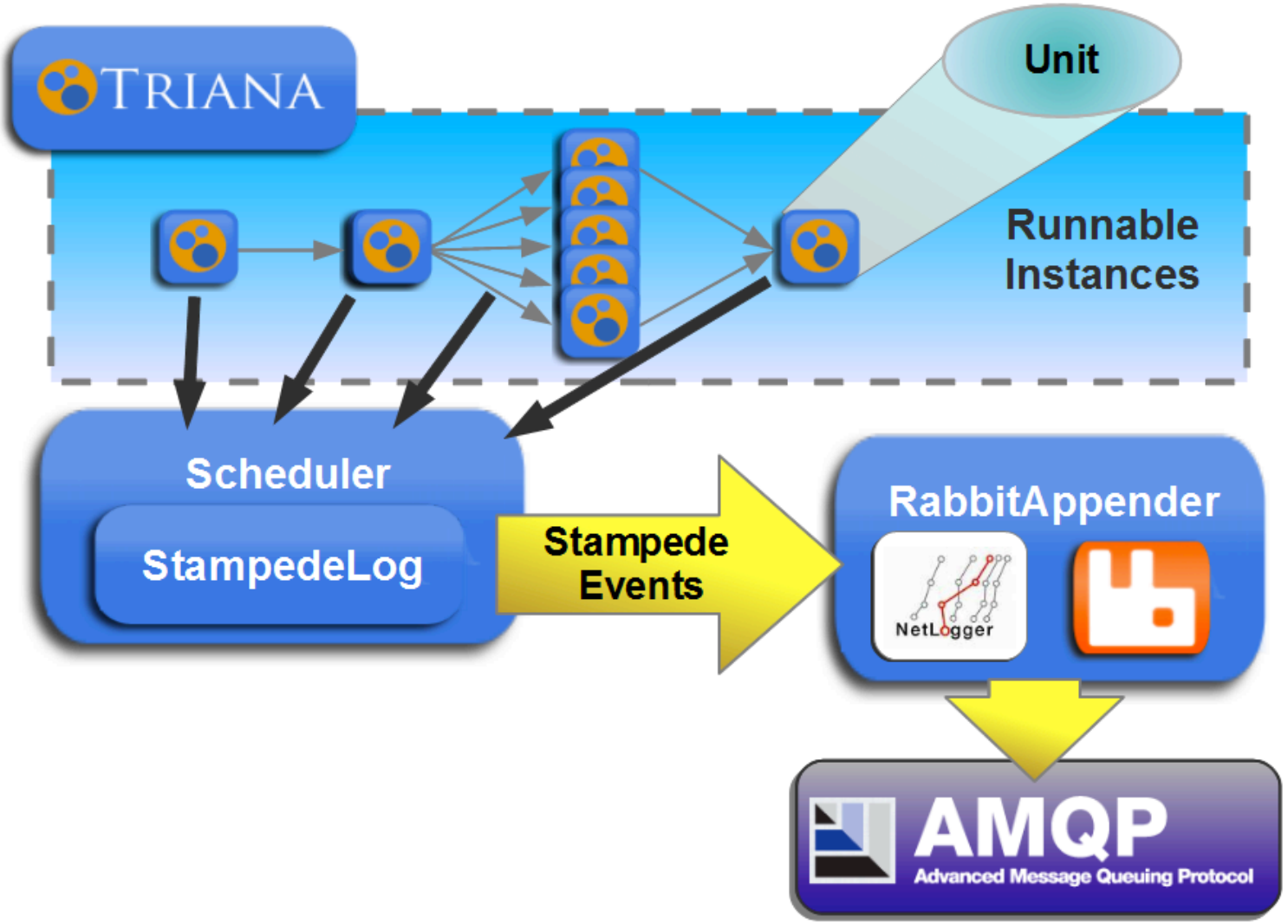
Triana Workflow System

- Workflow and Data Analysis Environment
- Interactive GUI to enable workflow composition
- Focused on data flows of Java components
- Has a wide range palette of tools that can be used to design applications
- Can distribute workloads to remote Cloud VM's

Mapping With Stampede Data Model



Triana Integration With Stampede

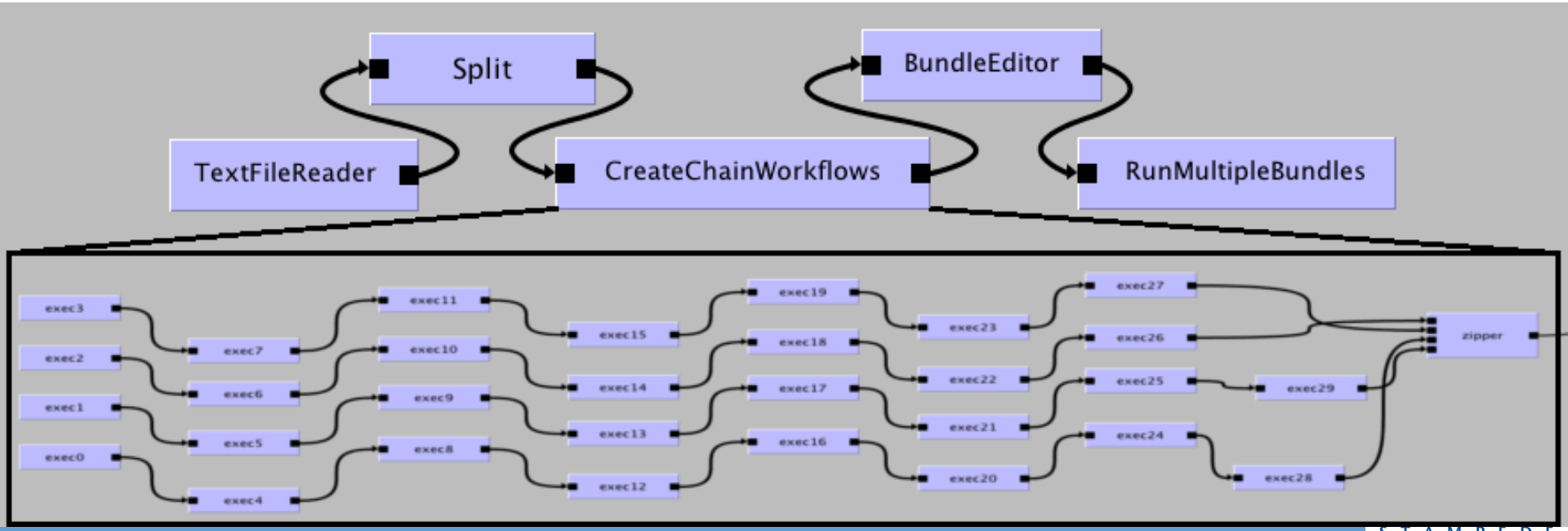


Outline

- Background
- Stampede Data Model
- Triana and Stampede Integration
- Experiments and Analysis Tools
- Conclusions and Future Work

Scientific Experiment

- DART Audio Processing Workflow
  - DART algorithm is portable and package as a jar file.
  - Parameter sweep resulting in 306 DART application invocations
  - Executed on a Triana Cloud Deployment in Cardiff



Performance Statistics - *stampede-statistics*

Workflow Statistics

Type	Succeeded	Failed	Incomplete	Total	Ret-ries	Total
Tasks	367	0	0	367	0	367
Jobs	367	0	0	367	0	367
Sub WF	20	0	0	20	0	20

Workflow wall time : 11 mins, 1 sec, (661 seconds).
Workflow cumulative job wall time : 11 hrs, 10 mins, (40224 seconds).

TABLE I
SUMMARY OUTPUT FROM STAMPEDE-STATISTICS FOR DART WORKFLOW

TASK Statistics Per Sub Workflow

Type	Count	Success	Failed	Min	Max	Mean	Total
115-119	1	1	0	1.0	1.0	1.0	1.0
Output_0	1	1	0	1.0	1.0	1.0	1.0
exec0	1	1	0	74.0	74.0	74.0	74.0
exec1	1	1	0	75.0	75.0	75.0	75.0
exec2	1	1	0	74.0	74.0	74.0	74.0
exec3	1	1	0	75.0	75.0	75.0	75.0
exec4	1	1	0	36.0	36.0	36.0	36.0
zipper	1	1	0	1.0	1.0	1.0	1.0

TABLE II
BREAKDOWN.TXT DESCRIBING THE TASKS IN A SUB-WORKFLOW

Performance Statistics - *stampede-statistics*

Job	Try	Site	Invocation Duration
unit:304-305	1	trianaworker6	1.0
exec1	1	trianaworker6	64.0
file.Output_0	1	trianaworker6	1.0
file.zipper	1	trianaworker6	1.0
processing.exec0	1	trianaworker6	51.0

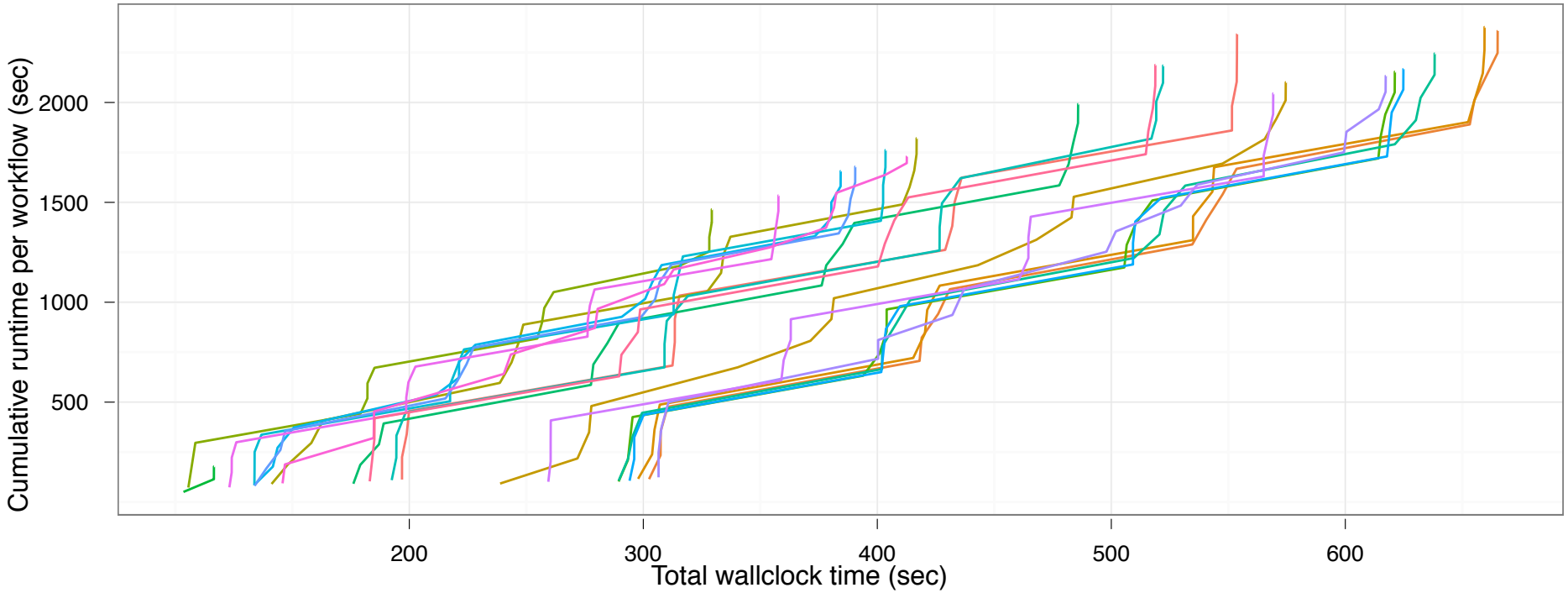
TABLE III
SECTION OF JOBS.TXT FOR A SINGLE SUB WORKFLOW

Job Level Statistics

Job	Queue Time	Runtime	Exit	Host
unit:304-305	0.06	1.0	0	None
exec1	0.04	64.0	0	trianaworker6
file.Output_0	0.0	1.0	0	trianaworker6
file.zipper	0.0	1.0	0	trianaworker6
processing.exec0	0.07	51.0	0	trianaworker6

TABLE IV
SECTION OF JOBS.TXT FOR A SINGLE SUB WORKFLOW

Workflow Analysis using R



Troubleshooting and Dashboard

Stampede-analyzer

- Interactive debugging tool
- Identifies what jobs failed and why they failed
- Drill down functionality for hierarchal workflows

Stampede Dashboard – Available Soon

- Lightweight Performance Dashboard
- Online monitoring and status of workflows
- Troubleshoot failed jobs
- Charts and statistics online

Outline

- Background
- Stampede Data Model
- Triana and Stampede Integration
- Experiments and Analysis Tools
- Conclusions and Future Work

Conclusions and Future Work

- Real-time failure prediction for scientific workflows is a challenging and important task
- Stampede provides a 3 layer model for integration with different workflow systems
  - Has been integrated with Pegasus WMS and now with Triana
  - Provides users useful real-time monitoring , debugging and analysis tools
- Apply Workflow and Job Failure prediction models to Triana Runs
- Dashboard for easier visualization of collected data

Thank you!

USC Viterbi
School of Engineering
Information Sciences Institute

CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD