COMMIT/

Scientific workflow management a way to enable e-science on both Grids and Clouds

Adam Belloum
Institute of Informatics
University of Amsterdam
a.s.z.belloum@uva.nl

SHIWA summer school MTA STAKI, Budapest HU July 2012

UvA Universiteit van Amsterdam

Outline

- Introduction
- Life cycle of e-Science Workflow
- Different approaches to workflow scheduling
  - Workflow Process Modeling & Management In Grid/Cloud
  - Workflow and Web services (intrusive/non intrusive)
- Provenance

University of Amsterdam

Computational Science

- Objective of the group
  - address the research issues related to **building an *e-Science* framework** which enables scientist to *share*, use *knowledge* add use geographically distributed resources (grids, clouds)

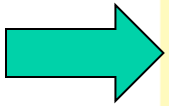
- Keywords:
  - Grid, Scientific workflow, SOA, provenance, interoperability

Collaborative e-Science experiments: from scientific workflow to knowledge sharing *A.S.Z. Belloum, Vladimir Korkhov, Spiros Koulouzis, Marcia A Inda, and Marian Bubak JULY/AUGUST, IEEE Internet Computing, 2011*

The project: COMMIT

- COMMIT is a public-private research community solving grand challenges in information and communication science shaping tomorrow's society.
- COMMIT has **15 projects** and **200 people** in **80 organisations** such as universities, TNO, Thales, Logica, Philips, AMC, and SME's like DevLab, Hyves, Waag.
- COMMIT delivers science, disseminates its results, measures its impact, generates synergy.

www.Commit-nl.nl




Information retrieval for information services


Interaction for universal access


Virtual worlds for well-being


Socially-enriched access to linked cultural media


SWELL: Smart reasoning systems for well-being at work and at home


Sensor networks for public safety


Very large wireless sensor networks for well-being


Composable embedded systems for healthcare


Metis: Dependable cooperative systems for public safety


Trusted healthcare services


Spatiotemporal data warehouses for trajectory exploitation


e-Infrastructure virtualization for e-science applications

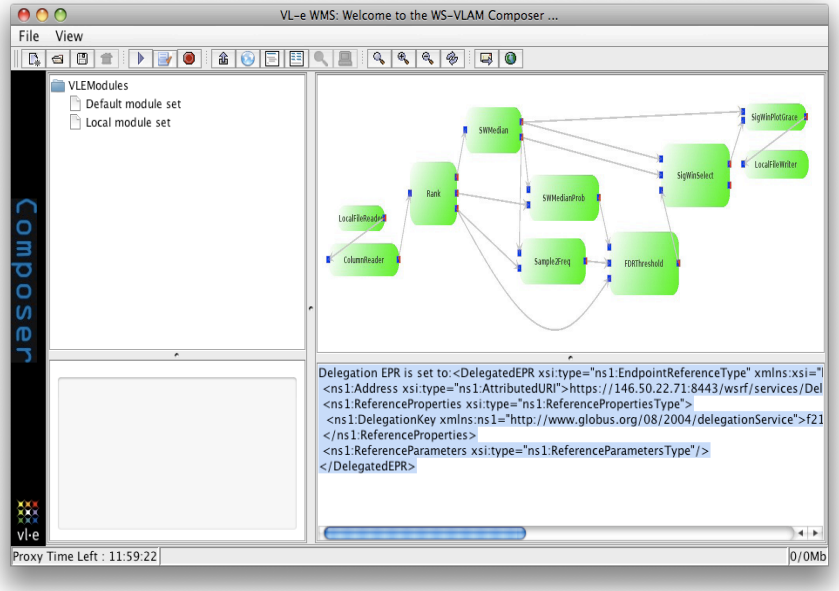

From data to semantics for scientific data publishers

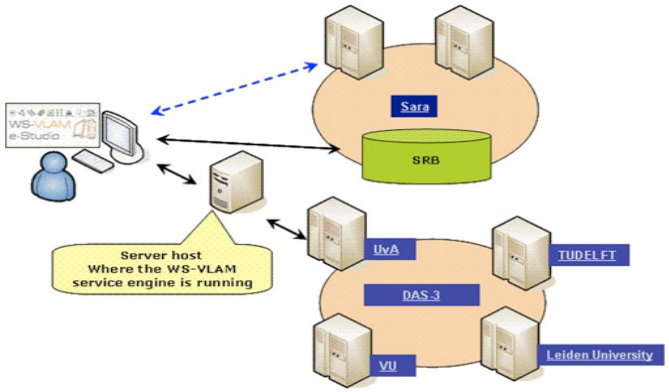

e-Biobanking with imaging for healthcare


e-Foodlab

Workflow management system

- **Workflow management system** is a computer program that manages the execution of a workflow on a set of computing resources.



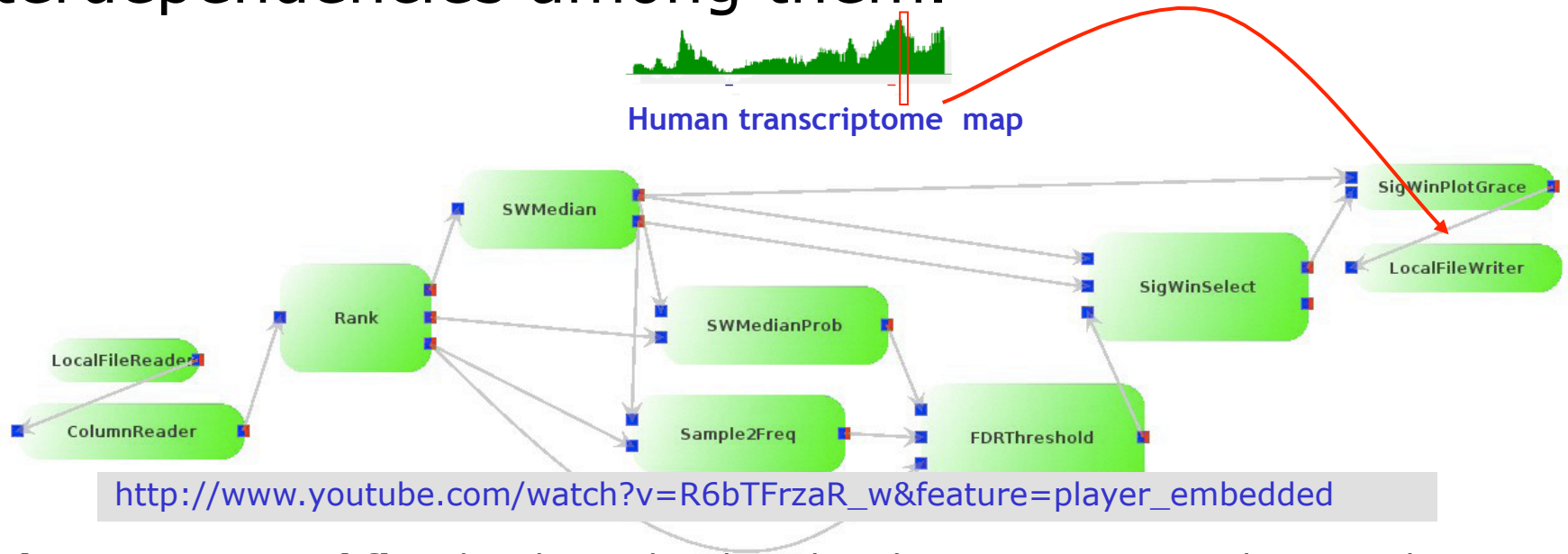


The user interface of the WS-VLAM a workflow management system developed in the VL-e project to execute application workflow on geographically distributed computing resources

Deployed as service on Dutch super Computer (DAS3), and Dutch NGI (BigGrid) Clusters

Workflow

A workflow is a model to represent a reliably repeatable sequence of operations/ tasks by showing explicitly the interdependencies among them.



http://www.youtube.com/watch?v=R6bTFrzaR_w&feature=player_embedded

SigWin-Detector workflow has been developed in the VL-e project to detect ridges in for instance a Gene Expression sequence or Human transcriptome map, BMC Research Notes 2008, 1:63 doi:10.1186/1756-0500-1-63.

List of applications developed using WS-VLAM

- sigWin detector ***[Micro-Array Dept-UvA]***
- *Affymetrix Permutation* ***[Micro-Array Dept-UvA]***
- *Omnimatch* ***[UU/Leiden]***
- wave propagation ***[TUE]***
- Blast ***[AMC]***
- gut microbiota ***[CWI]***
- Smart Infrastructure ***[SNE-UvA]***
- Dynamic network control ***[SNE-UvA]***
- GridSFEA, ***[TU Munchen]***

More applications www.science.uva.nl/~gvlam/wsvlam/Applications

Let us Start with an UvA-Slide: Virtual Laboratory for eScience

Step1: designing an experiment

Step2: performing the experiment

Step3: analyzing the experiment results

success

Scientific Workflow Management Systems

Process Flow Template

Taverna, Kepler, and Triana: model processes for computing tasks.

Process Flow Template

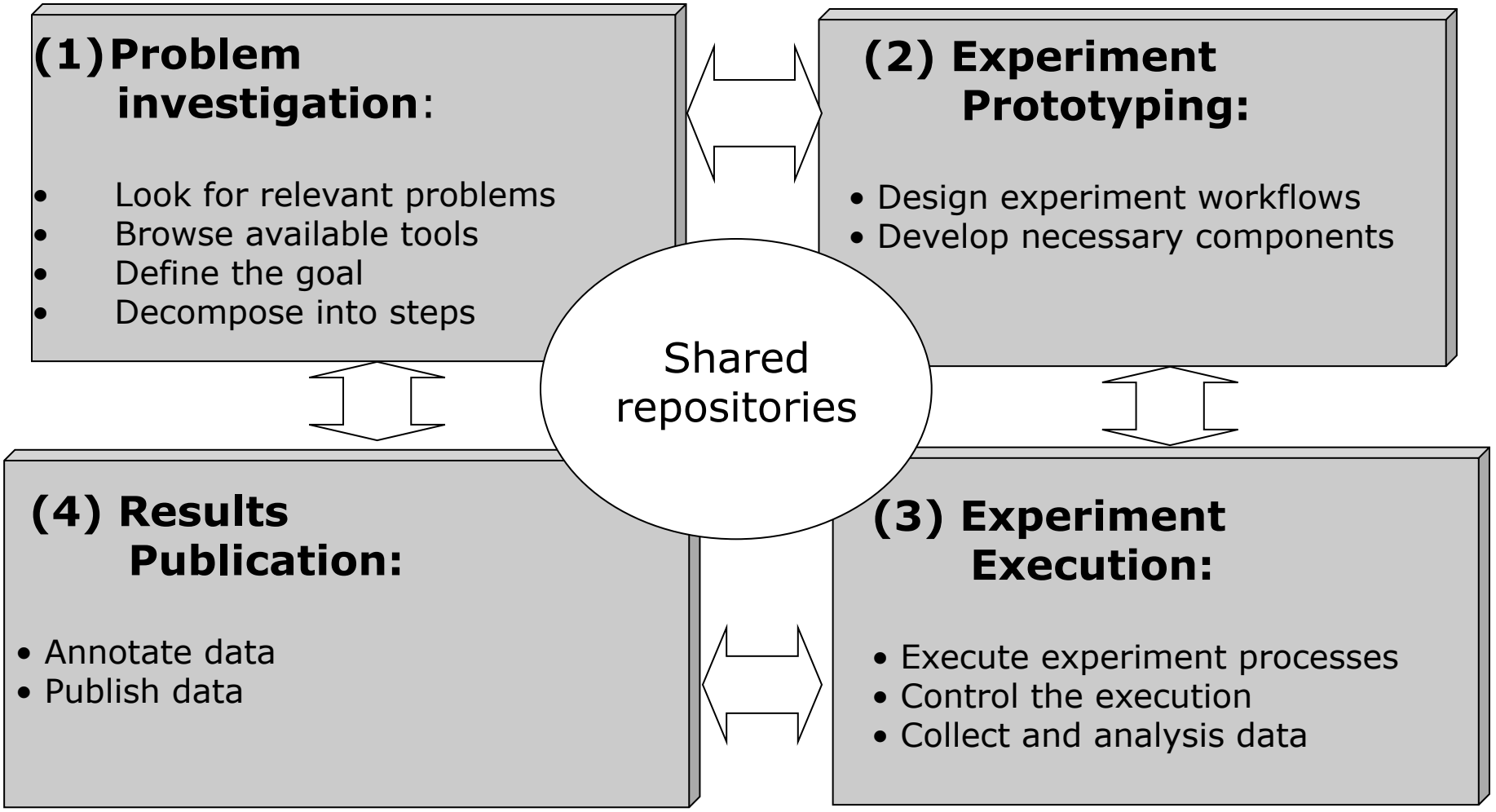
PFT in VLAMG

PFT instances in VLAMG

In VL-e: model both computing tasks and human activity based processes, and model them from the perspective of an entire lifecycle. It tries to support

- Collaboration in different stages
- Information sharing
- Reuse of experiment

Complex Scientific experiments model



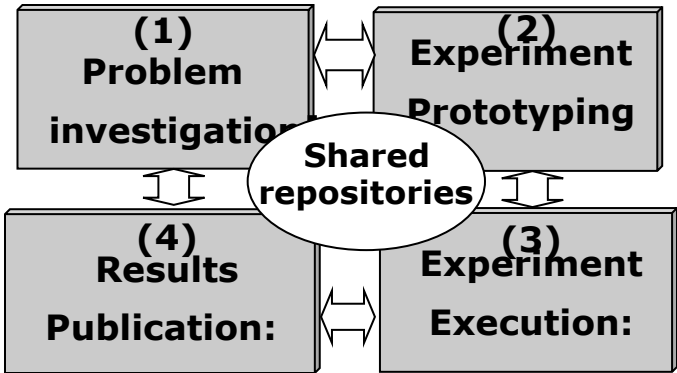
Collaborative e-Science experiments: from scientific workflow to knowledge sharing *A.S.Z. Belloum, Vladimir Korkhov, Spiros Koulouzis, Marcia A Inda, and Marian Bubak JULY/AUGUST, IEEE Internet Computing, 2011*

Targets

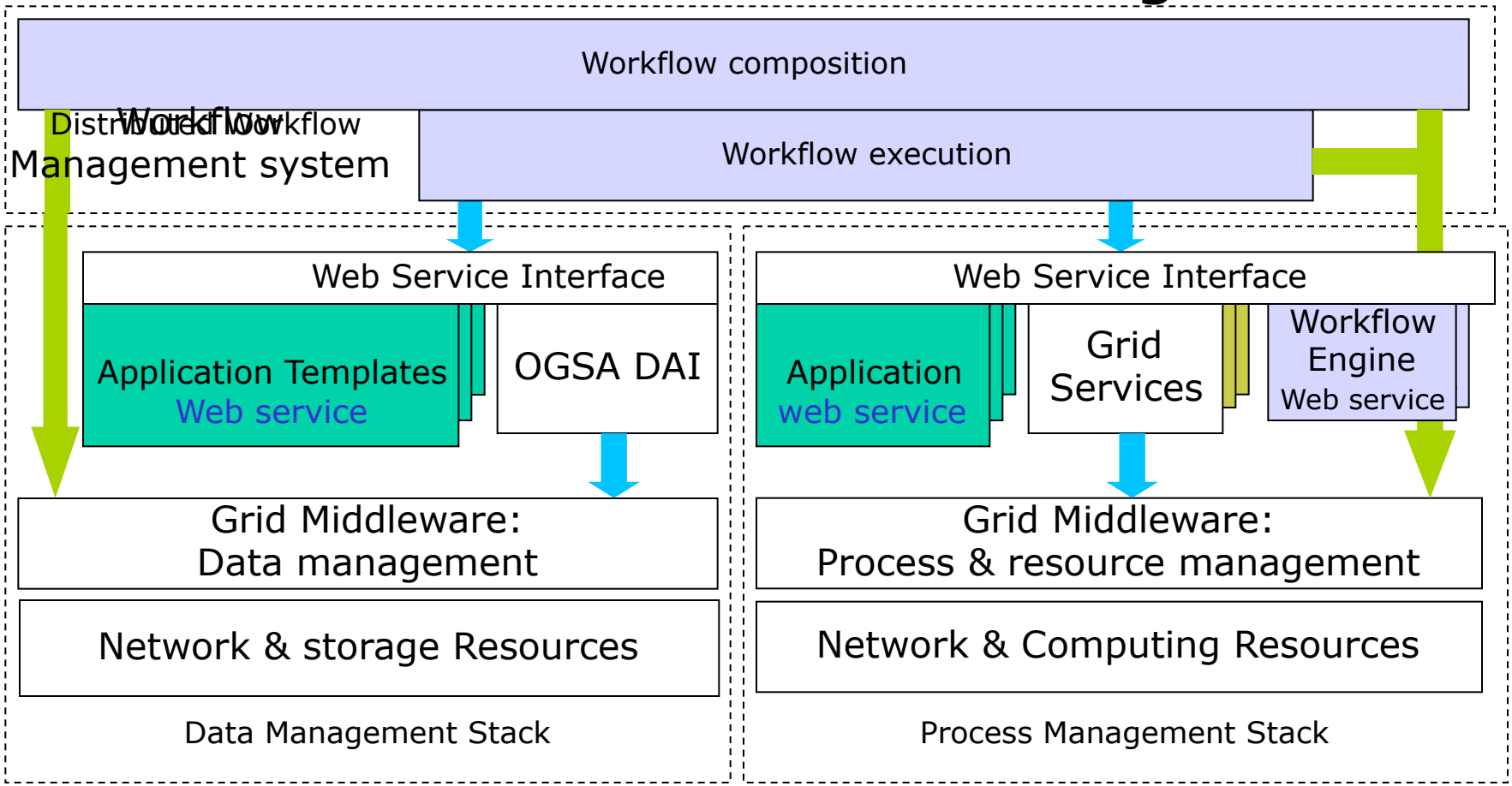
- ***co-allocate*** *resources needed for workflow enactment across multiple domains*?
- *achieve* ***QoS*** *for data centric application workflows that have special requirements on network connections*?
- *achieve* ***Robustness*** *and fault tolerance for workflow running across distributed resources*?
- *increase* ***re-usability*** *of Workflow, workflow components, and refine workflow execution*?

Outline

- Introduction
- Lifecycle of an e-science workflow
- Different approach to workflow scheduling
  - Workflow Process Modeling & Management In Grid/Cloud
  - Workflow and Web services Workflow and Web services (intrusive/non intrusive)
- Provenance

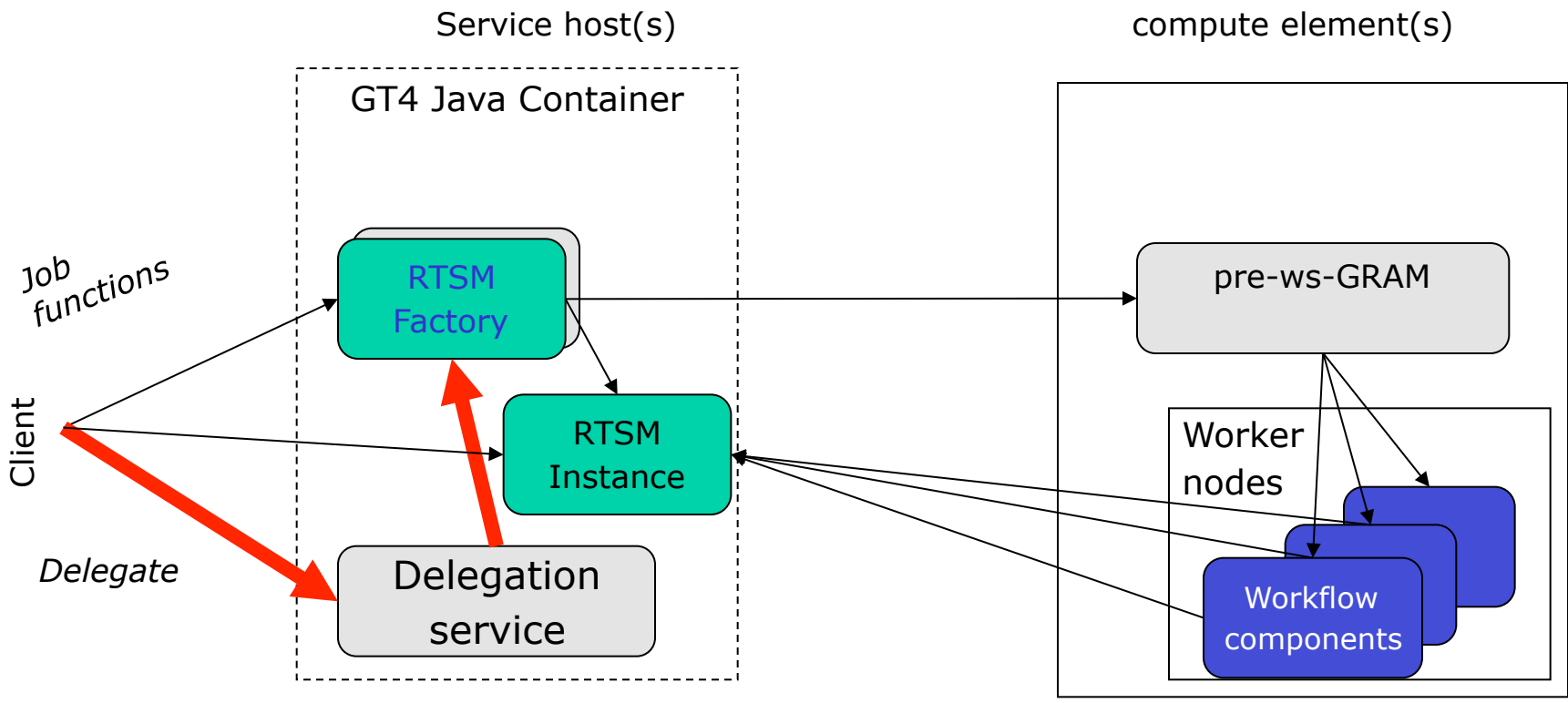


A WSRF enabled workflow engine

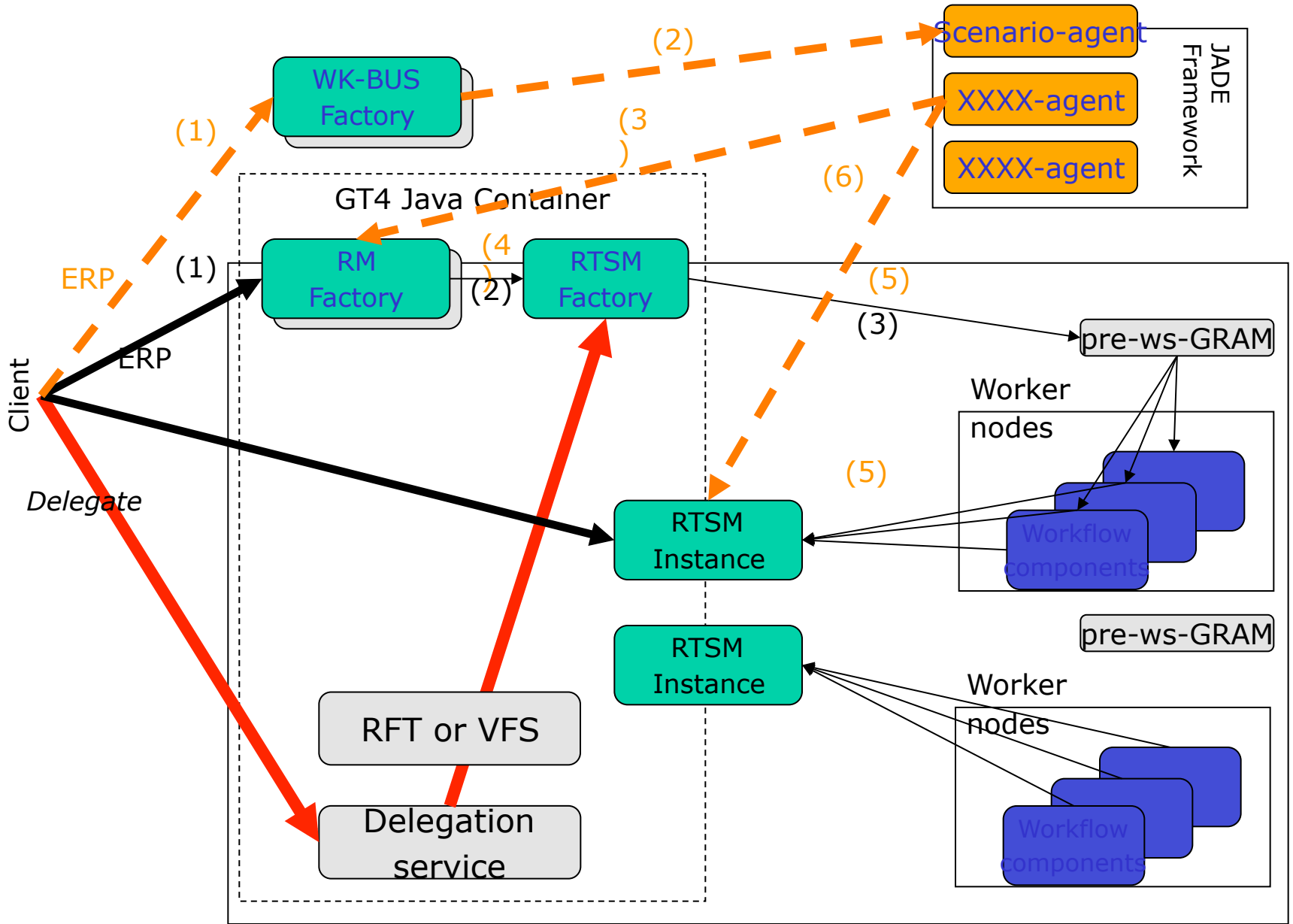


Bob Hertberger keynote talk at *2nd IEEE Conf on eScience & grid computing , Amsterdam 2006*

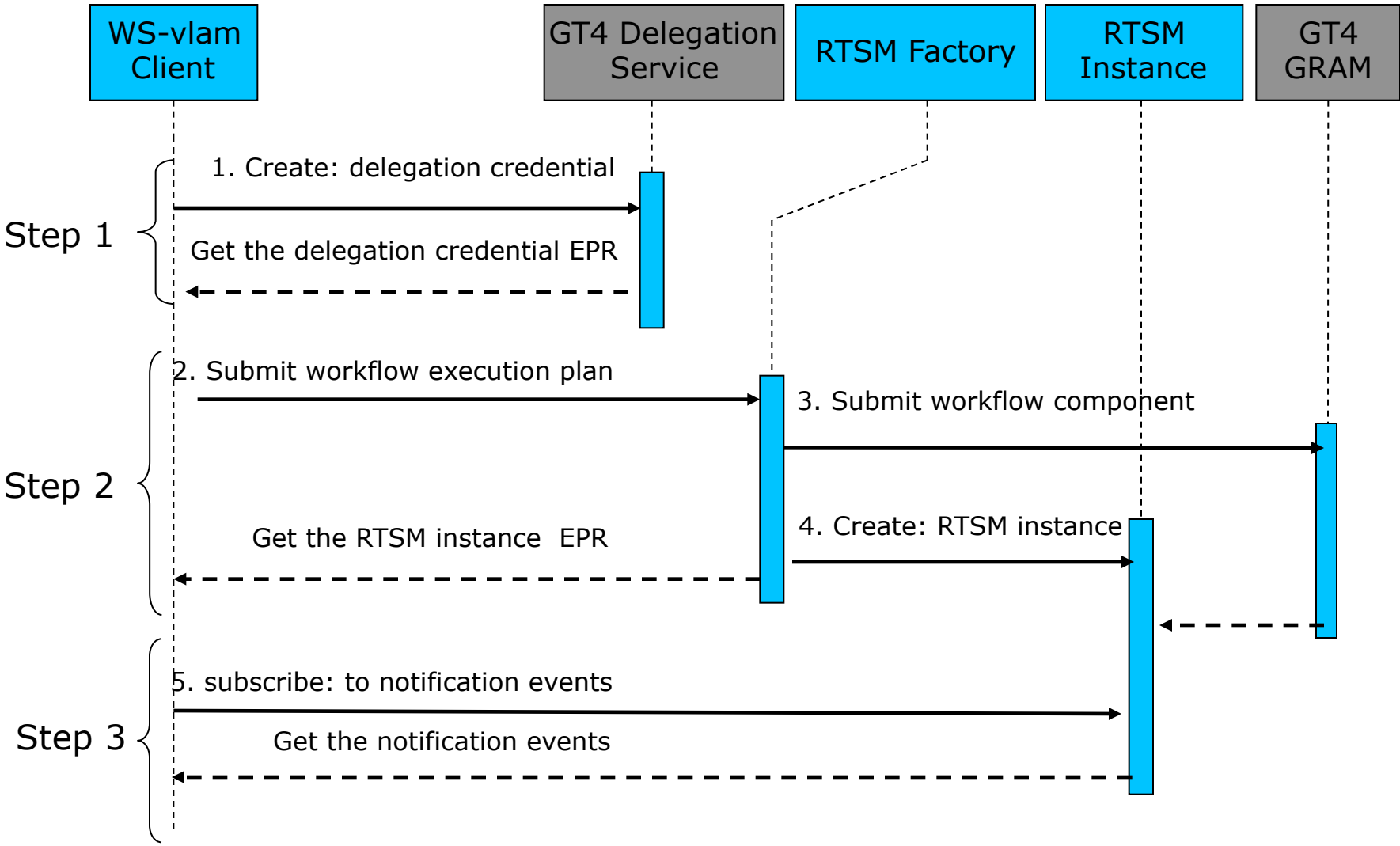
WS-VLAM Engine: architecture (1/2)



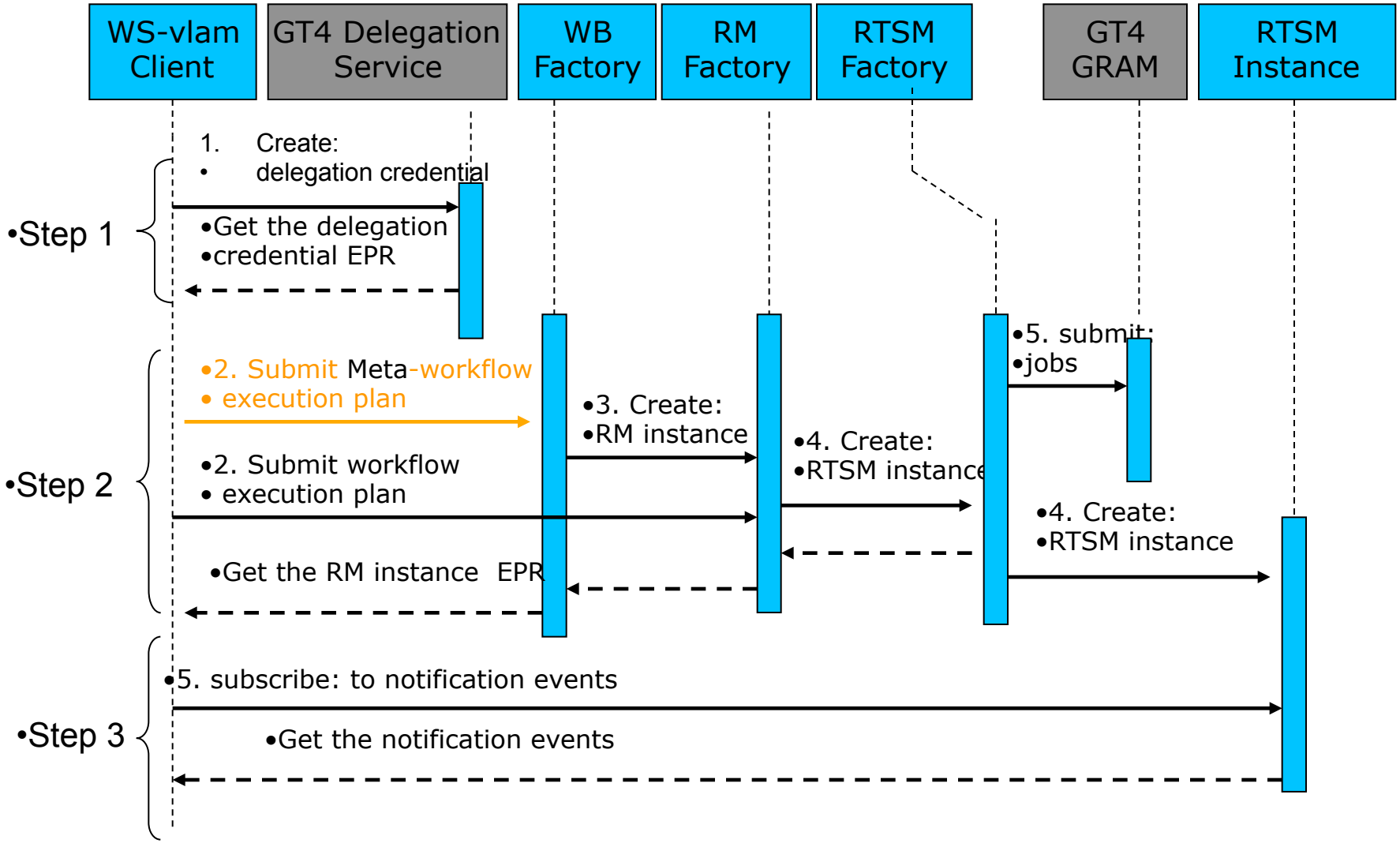
WS-VLAM Engine: architecture (2/2)



Sequence-diagram



Sequence-diagram

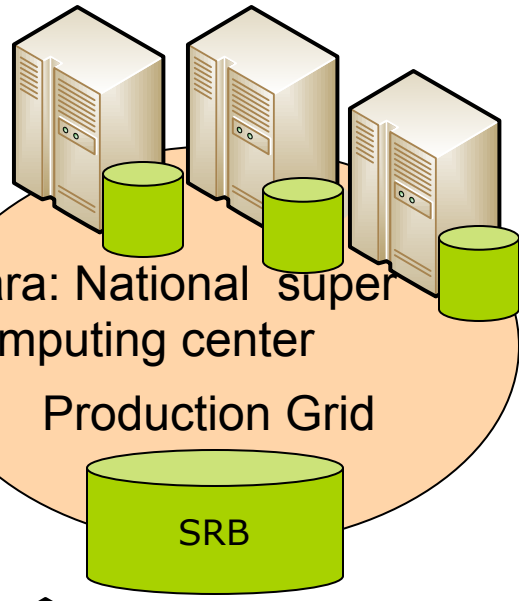


Current depl

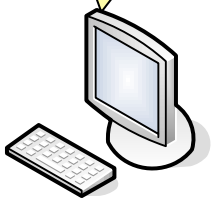
WSRF Services
- WS-VLAM engine
- workflow component repository

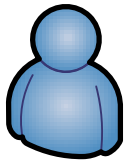
VLe Studio
- WS-VLAM composer
- VBrowser
- Semantic tools

SAW: **S**emantic **A**nnotation for **W**orkflow
CLAMP: Connecting **LA**nguage for **M**odules & **P**rograms
HAMMER: Hybrid-b**A**sed **M**atch**M**aker for e-Science **R**esources

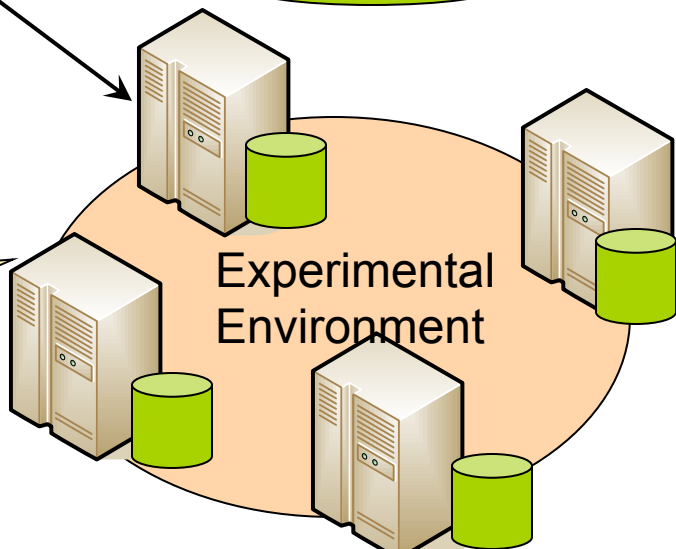


Server host





Computing Nodes
- Workflow components
- Grid Middleware → GT4



Model of computation

- Model of computation: stream-based process network.
  - Engine **co-allocates** all workflows.
  - Components waste time idling.
  - Co-allocation difficult.
- Communication: time **coupled**
  - Assumes components are running
  - Simultaneously
  - Synchronized p2p
  - Fixed TCP/IP

WS-VLAM communication library

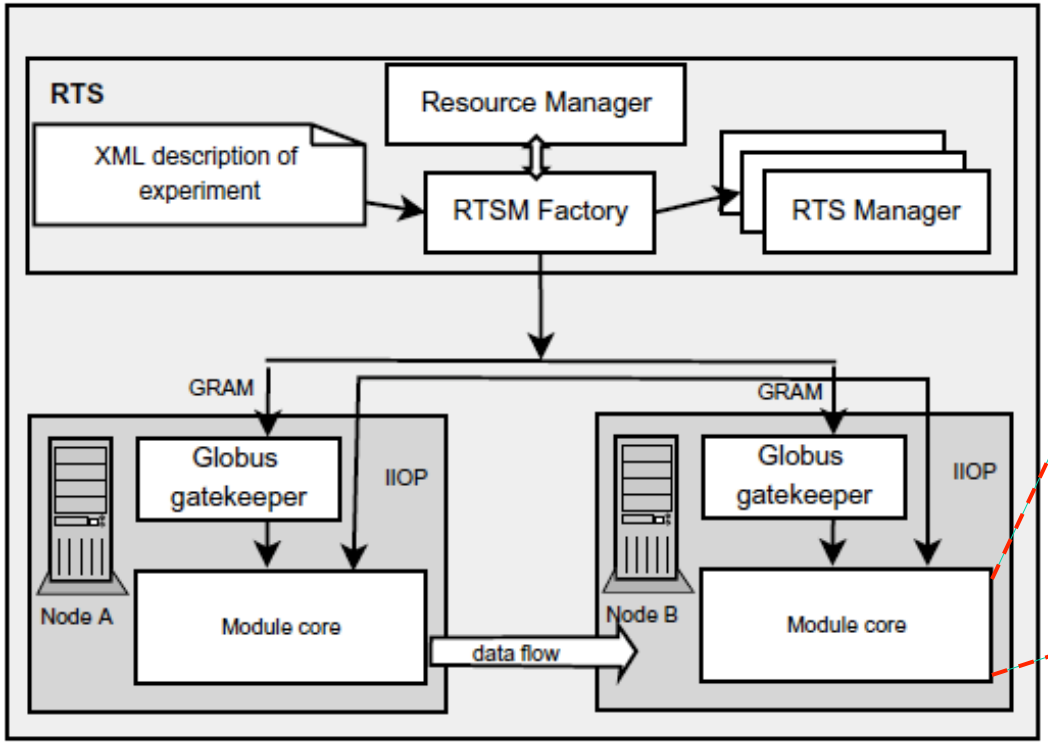


Fig. 1. Run-Time System Architecture.

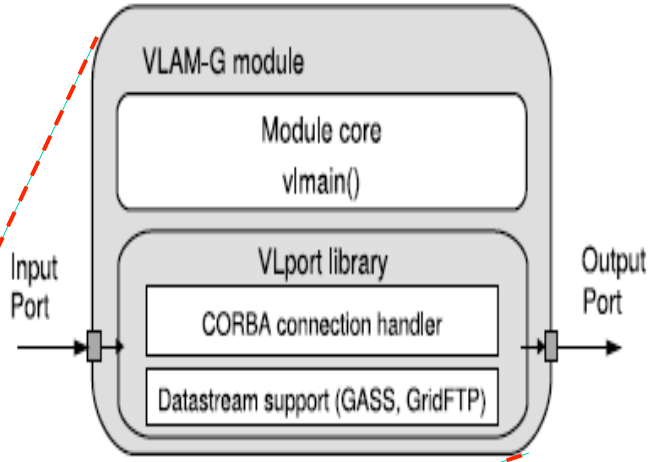


Fig. 4. Port library component architecture.

V. Korkhov et al. VLAM-G: Interactive data driven workflow engine for Grid-enabled resources, Scientific Programming 15 (2007) 173–188 173 IOS Press

WS-VLAM communication library

- Data transfer rate as a function of the data block size (average of 10 measurements per each data-block
- with the deviation not exceeding 5 percent)

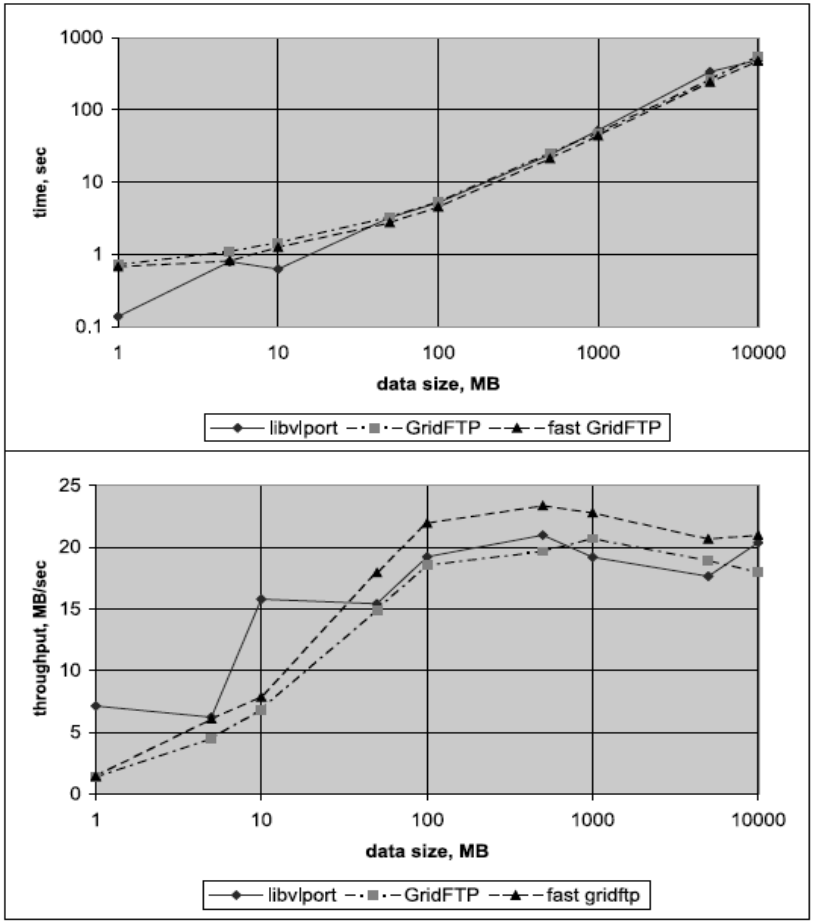


Fig. 7. Average performance of the RTS library on WAN compared with standard Globus data transfer tool.

V. Korkhov et al. VLAM-G: Interactive data driven workflow engine for Grid-enabled resources, Scientific Programming 15 (2007) 173–188 173 IOS Press

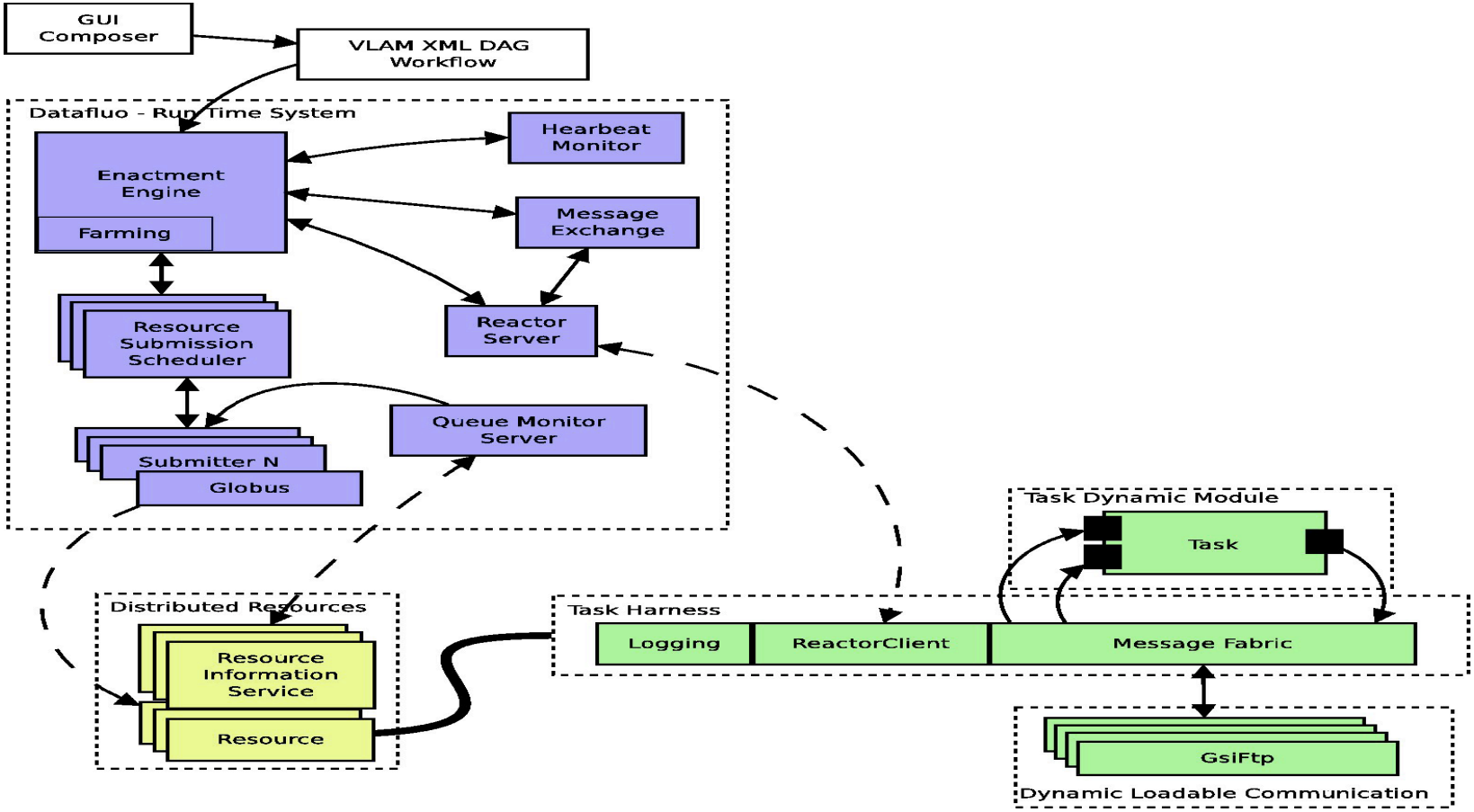
Model of computation

- Model of computation: dataflow network
  - components **scheduled** depending on data
  - components **only activated** when data is available
  - **no need** for **co-allocation**

- Communication: time decouples
  - messaging communication system.
  - components not synchronized
  - communication not strictly TCP/IP

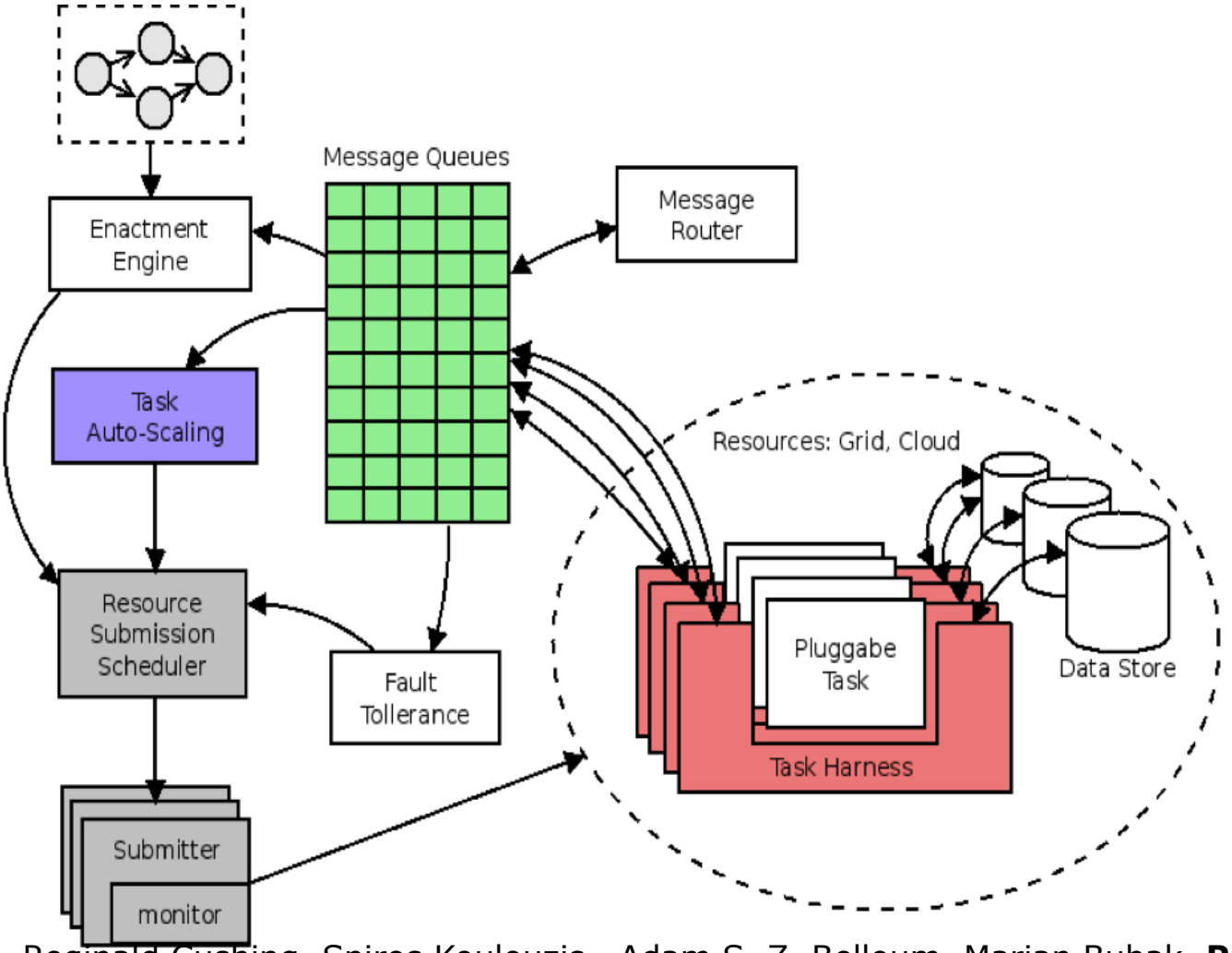
Additional features-Farming

- Task farming: task replication.
- Increases data consumption and production.
- Implements 3 types of farming:
  - Auto Farming: The engine decides on farm size depending on port load.
  - One-to-One Farming: A task replicated for every message received.
  - Fixed Farming: Statically defined.
- Allows parameter sweep studies.
- A task becomes a parameter engine



Reginald Cushing, Spiros Koulouzis, Adam S. Z. Belloum, Marian Bubak, **Prediction-based Auto-scaling of Scientific Workflows**, 7th IEEE International Conference on e-Science, December 2011, Stockholm, Sweden

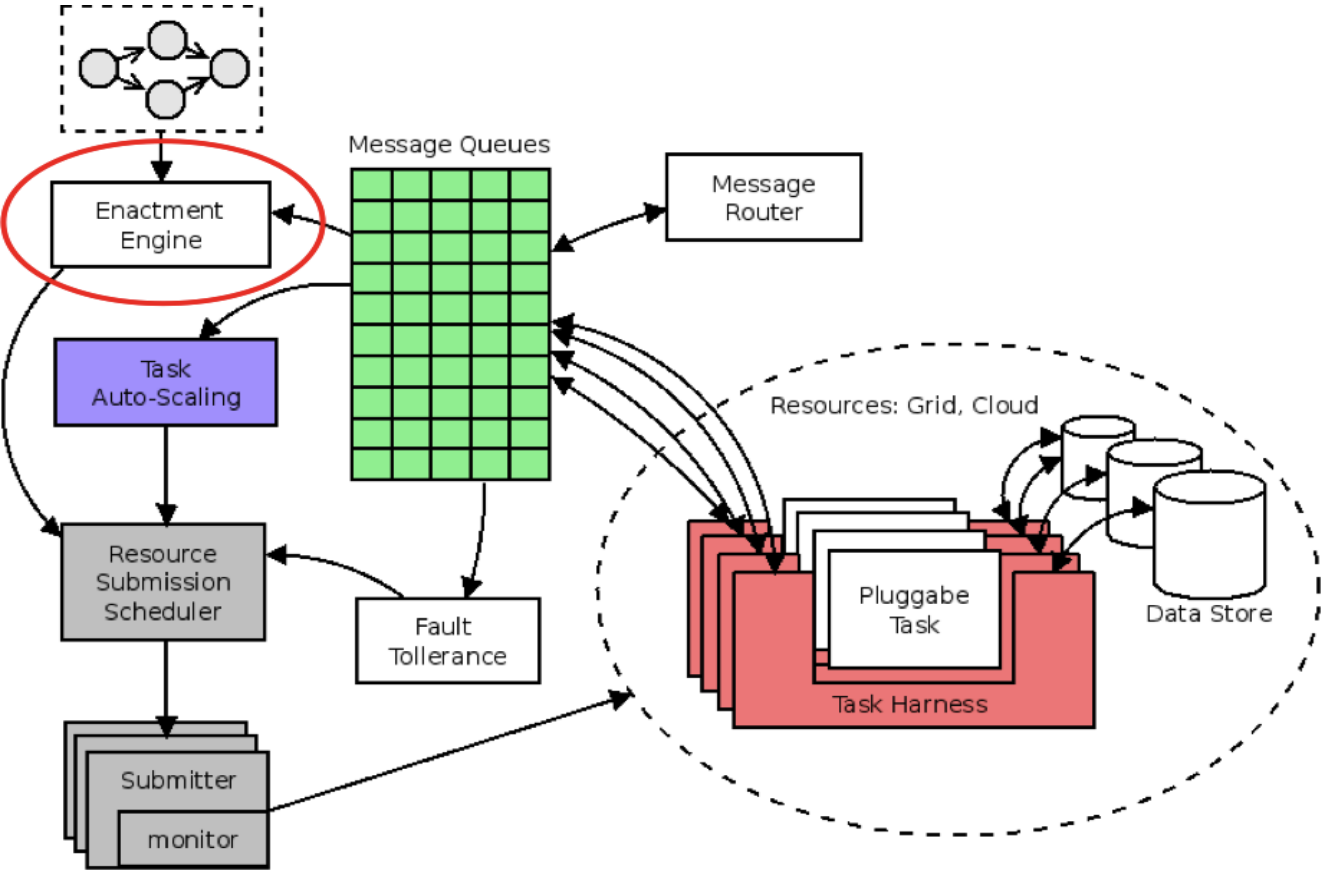
System Overview



Loosely coupled modules revolving around a message broker

Reginald Cushing, Spiros Koulouzis, Adam S. Z. Belloum, Marian Bubak, **Prediction-based Auto-scaling of Scientific Workflows**, Proceedings of the 9th International Workshop on Middleware for Grids, Clouds and e-Science, ACM/IFIP/USENIX December 12th, 2011, Lisbon, Portugal

Enactment Engine



Dataflow engine (**top-level** scheduler) based on Freefluo[§]

Models workflows as dataflow graphs

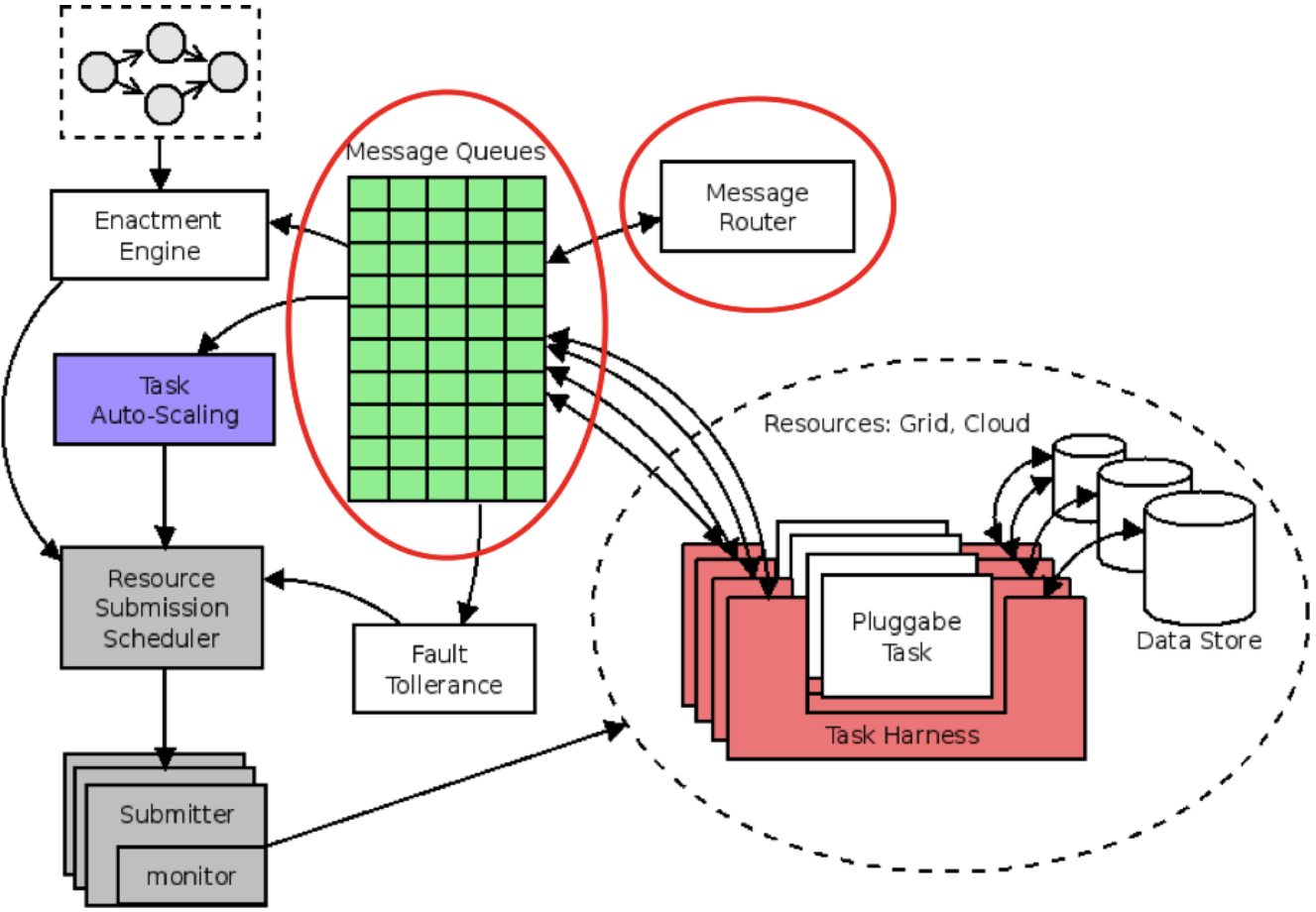
Vertices are **tasks** while **edges** are **dependencies**(data

Tasks have **ports** to simulate data channels

Dataflow model dictates that **only** tasks which have input are scheduled for execution.

[§]http://freefluo.sourceforge.net

Message Broker



Message broker plays a pivotal role in the system

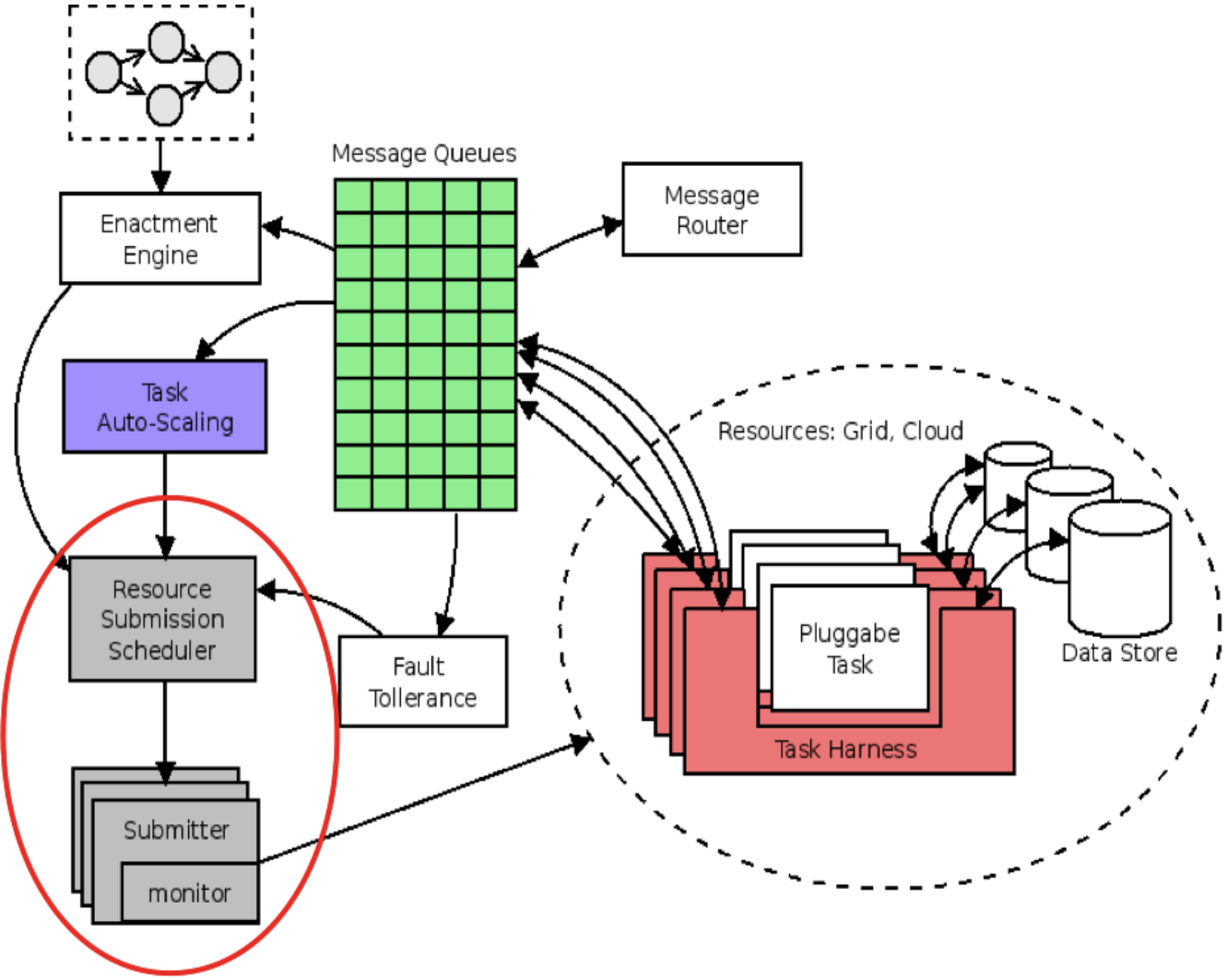
Message broker acts as a data buffer

Communicating tasks are **time decoupled**

Through queue sharing we can achieve scaling

Tasks **communicate** through messaging where messages contain **references** to actual data

Submission System



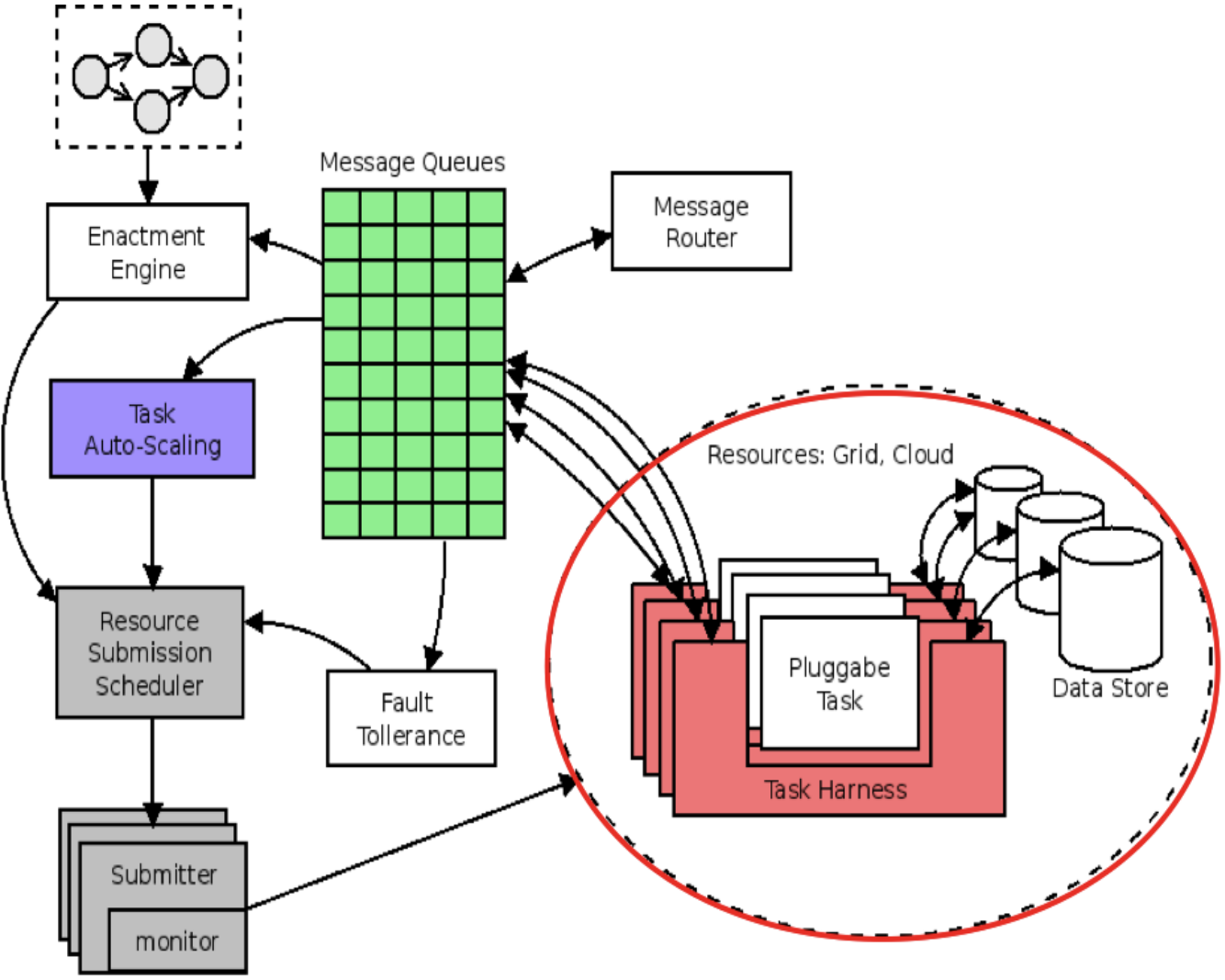
Pluggable schedulers (**bottom-level**) for task match-making

Submitters (drivers) abstract actual resources such as cluster, grid, cloud

Scheduler matches a task to a submitter

Submitter does actual task/job submission

Task Harnessing



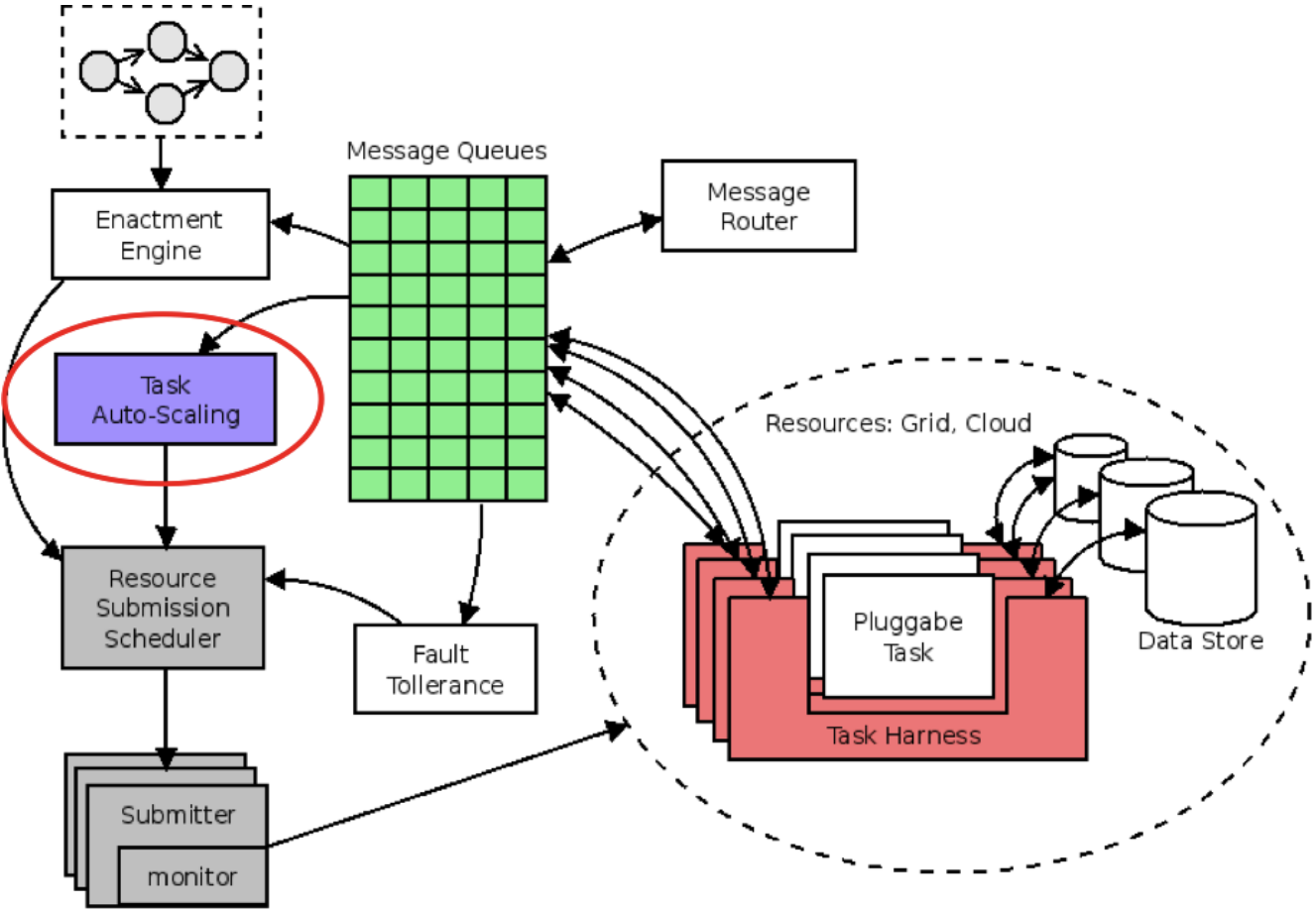
Task harness is a **late binding**, **pilot-job** mechanism

A harness is submitted which will **pull** the actual job

The harness separates data transport from scientific logic

Better control of tasks

Task Auto-scaling



Messages between tasks are monitored

Queued data and mean data processing time are used to calculate **task load**

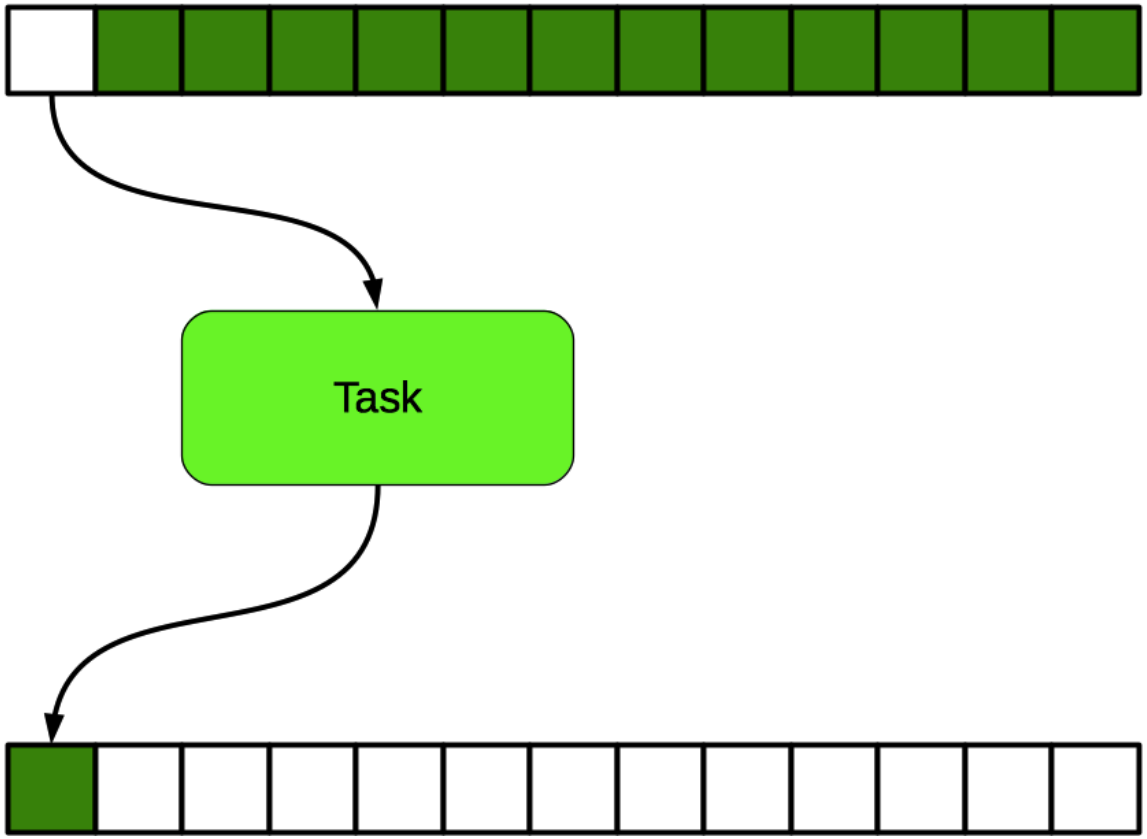
Auto-scaling **replicates** a particular task to ameliorate the task load

Replicated tasks (*clones*) **partition** data by sharing same input message queues

Scaling Concepts

Data organized in atomic parcels(messages)

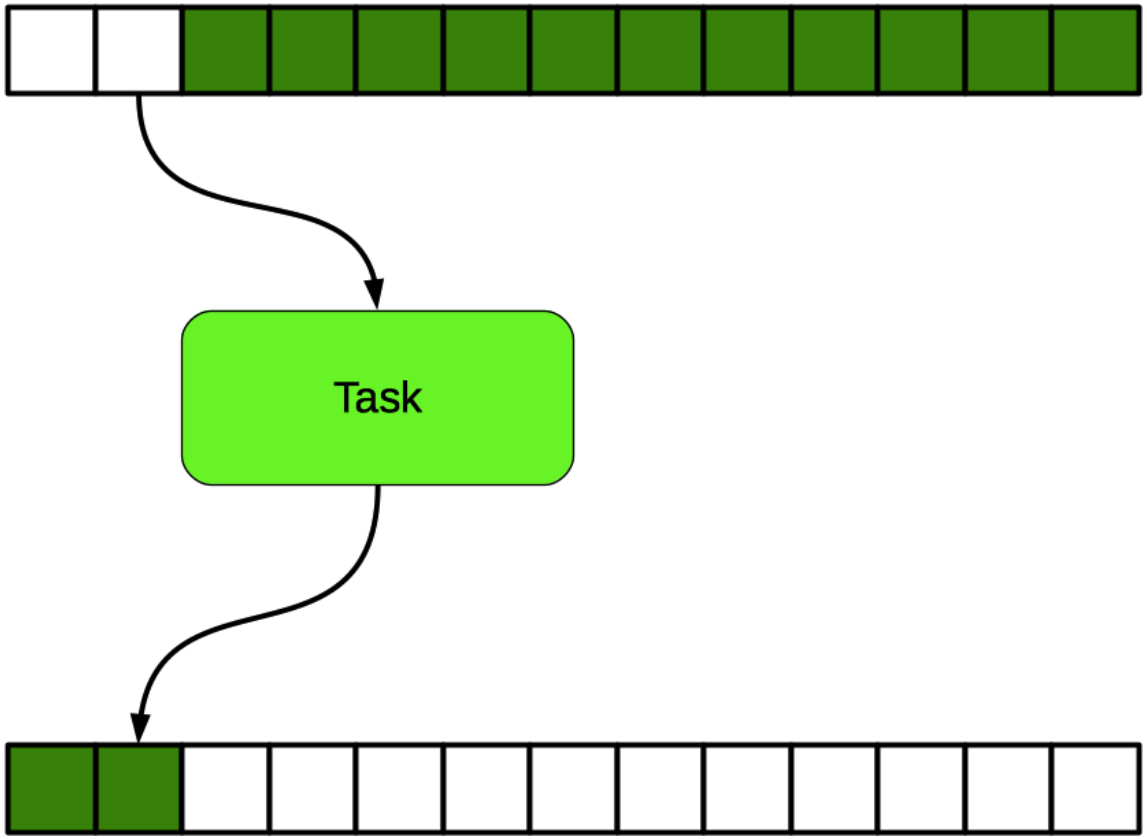
Scaling Concepts



Data organized in atomic parcels(messages)

Task processes data sequentially

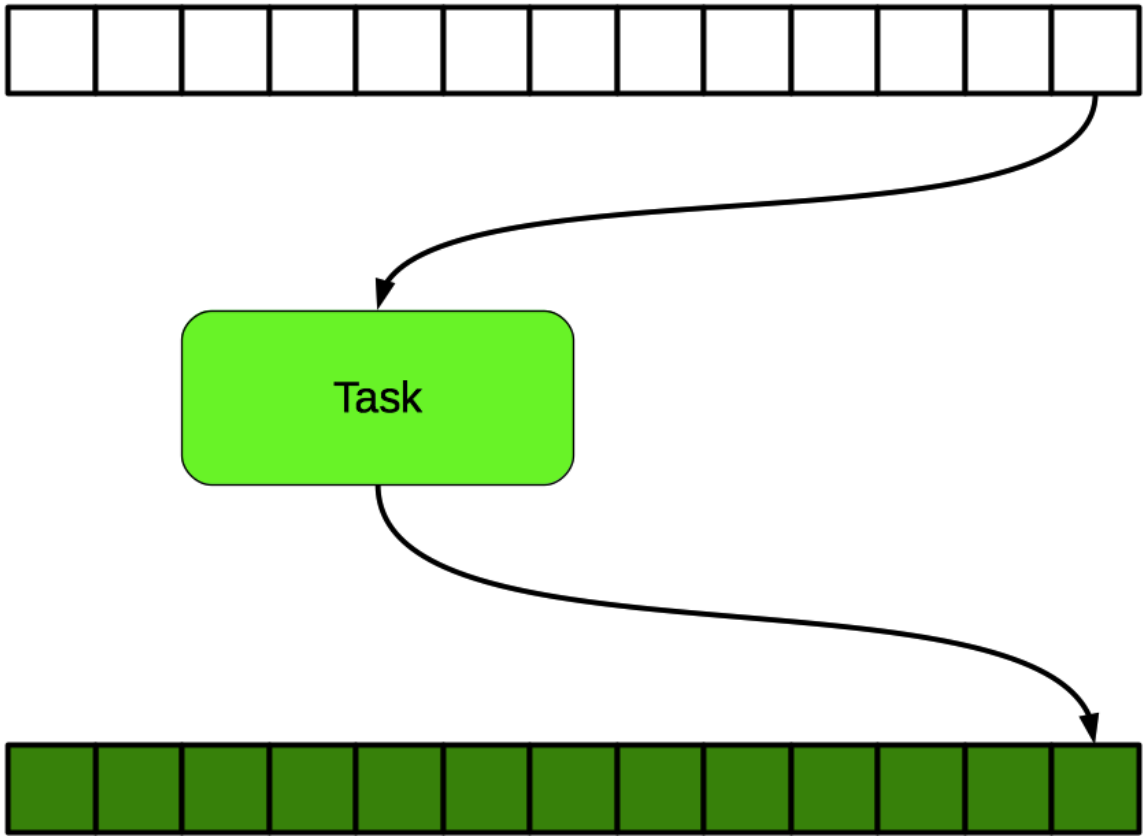
Scaling Concepts



Data organized in atomic parcels(messages)

Task processes data sequentially

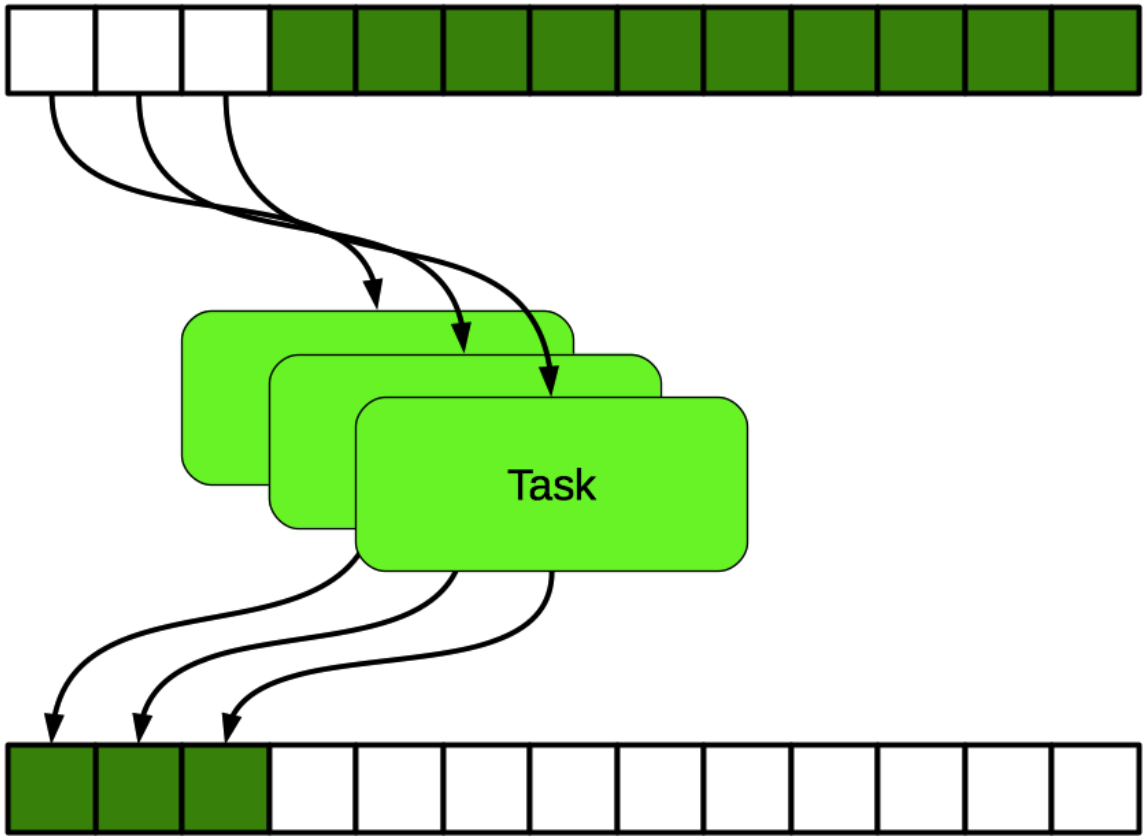
Scaling Concepts



Data organized in atomic parcels(messages)

Task processes data sequentially

Scaling Concepts

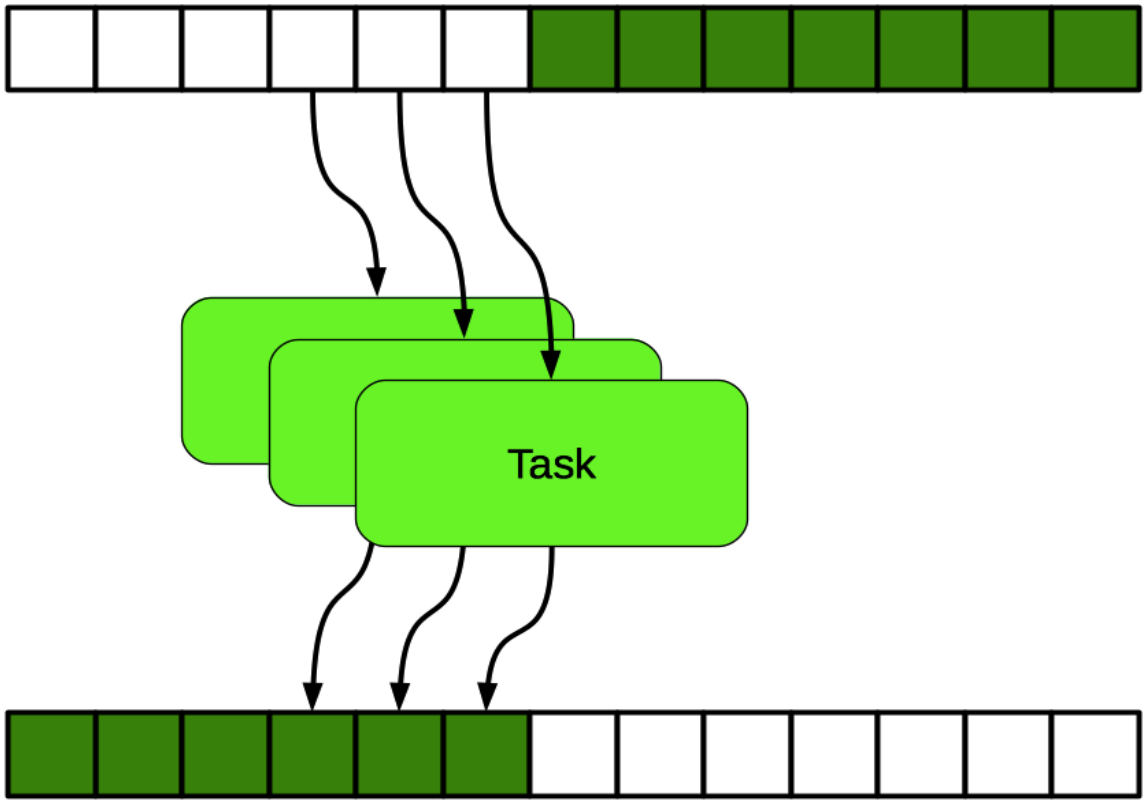


Data organized in atomic parcels(messages)

Tasks processes data **concurrently**

Adding more tasks increases **message consumption** rate

Scaling Concepts



Data organized in atomic parcels(messages)

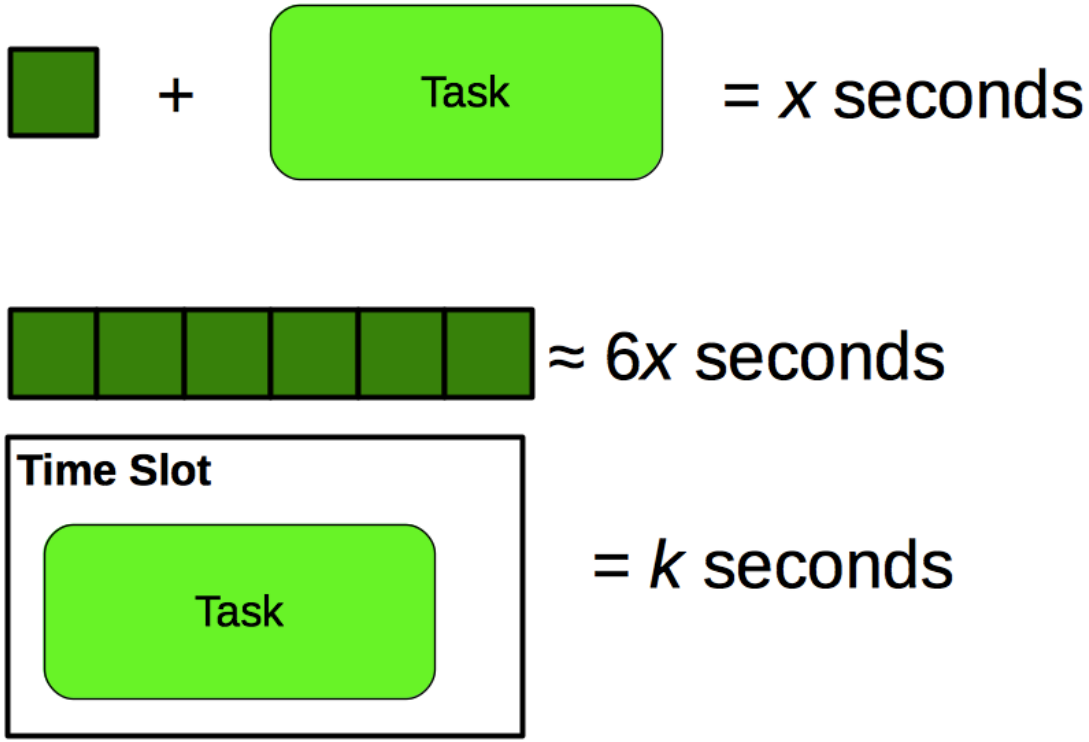
Task processes data sequentially

Adding more tasks increases **message consumption** rate

Challenge: How many tasks to create?

Too **many** and tasks get stuck on queues. Too **few** and optimal performance not achieved

Load Prediction



Simplified Load = 6*x*/*k* time slots

Assumption: Size of data directly proportional to computation time. May not always be the case

Auto-scaling steps

Calculate data processing rate for designated port

$$proc(ip_{t_k}) = \frac{size(C_{ip.t_k})}{\sum_{j=2}^{n} mit_{ip.t_k}(m_{j-1}, m_j)}$$

Auto scaling steps

Calculate data processing rate for designated port

$$proc(ip_{t_k}) = \frac{size(C_{ip.t_k})}{\sum_{j=2}^{n} mit_{ip.t_k}(m_{j-1}, m_j)}$$

Size of Consumed Data

Data Processing time

Auto scaling steps

Calculate data processing rate for designated port

$$proc(ip_{t_k}) = \frac{size(C_{ip.t_k})}{\sum_{j=2}^{n} mit_{ip.t_k}(m_{j-1}, m_j)}$$

Predict processing time for queued data

$$pred(ip_{t_k}) = size(Q_{ip.t_k}) \times proc(ip_{t_k})$$

Auto-scaling Steps

Calculate data processing rate for designated port

$$proc(ip_{t_k}) = \frac{size(C_{ip.t_k})}{\sum_{j=2}^{n} mit_{ip.t_k}(m_{j-1}, m_j)}$$

Predict processing time for queued data

$$pred(ip_{t_k}) = size(Q_{ip.t_k}) \times proc(ip_{t_k})$$

Data size of queued data

Auto-scaling Steps

Calculate data processing rate for designated port

$$proc(ip_{t_k}) = \frac{size(C_{ip.t_k})}{\sum_{j=2}^{n} mit_{ip.t_k}(m_{j-1}, m_j)}$$

Predict processing time for queued data

$$pred(ip_{t_k}) = size(Q_{ip.t_k}) \times proc(ip_{t_k})$$

Calculate needed replicated instances

$$repl(t_k) = \frac{pred(ip_{t_k})}{t_k^{st}}$$

Auto-scaling Steps

Calculate data processing rate for designated port

$$proc(ip_{t_k}) = \frac{size(C_{ip.t_k})}{\sum_{j=2}^{n} mit_{ip.t_k}(m_{j-1}, m_j)}$$

Predict processing time for queued data

$$pred(ip_{t_k}) = size(Q_{ip.t_k}) \times proc(ip_{t_k})$$

Calculate needed replicated instances

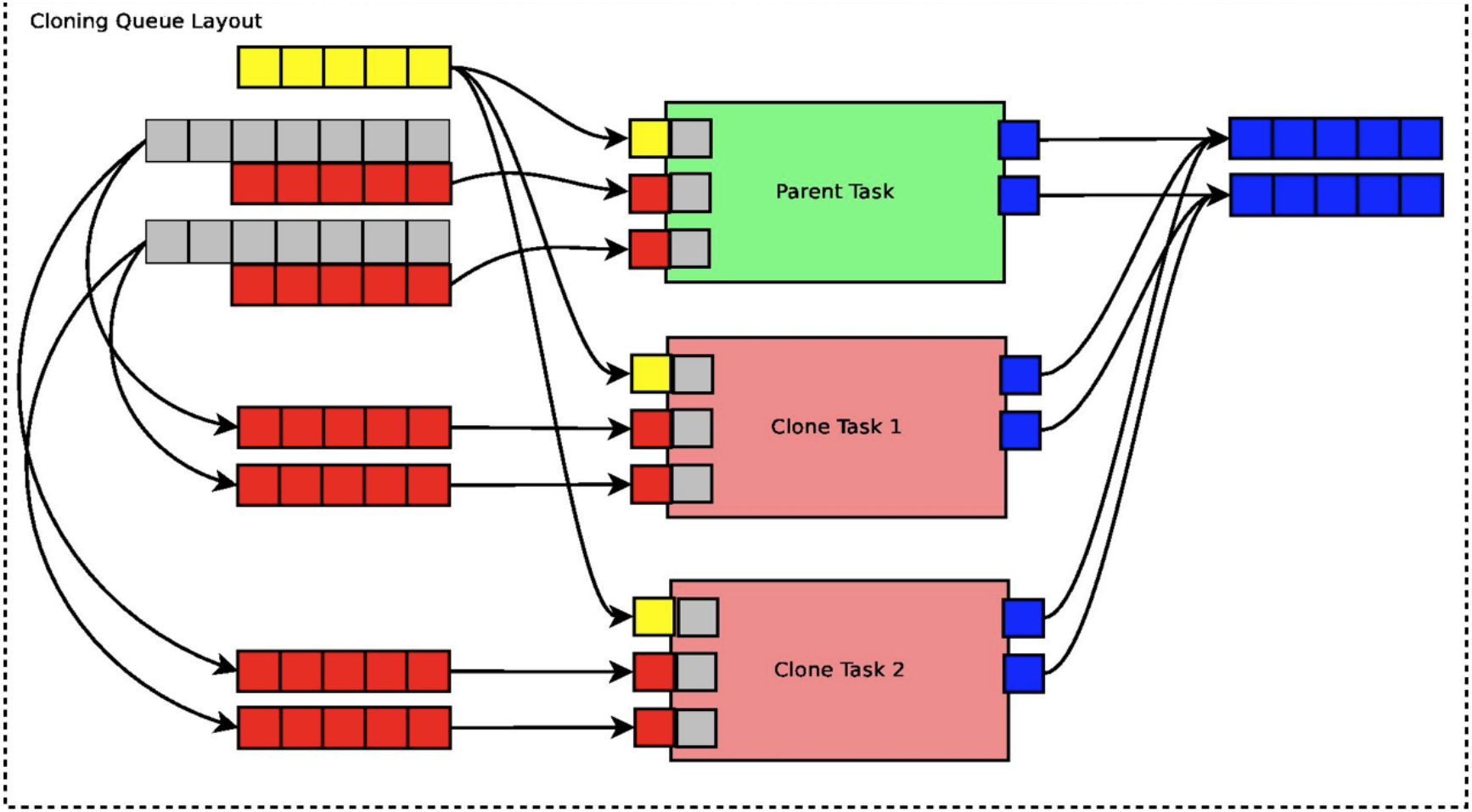
$$repl(t_k) = \frac{pred(ip_{t_k})}{t_k^{st}}$$

Time Quantum for task

Auto-scaling Steps (summary)

- Each task **port** is monitored to calculate the data **processing rate**
- Data is parceled in messages. Tasks consume messages
- Using the **mean data processing rate** and the amount of **data queued** on the port we extrapolate the proc. time for all data
- Based on the **current** resource we can estimate how many clones are needed to process all data within a time quantum
- Clones are submitted in **bursts** so not to flood resources
- Port is **continuously** monitored and further bursts can be submitted
- Once clones are active, message consumption is **faster** since clones share **same queues**

Queue sharing

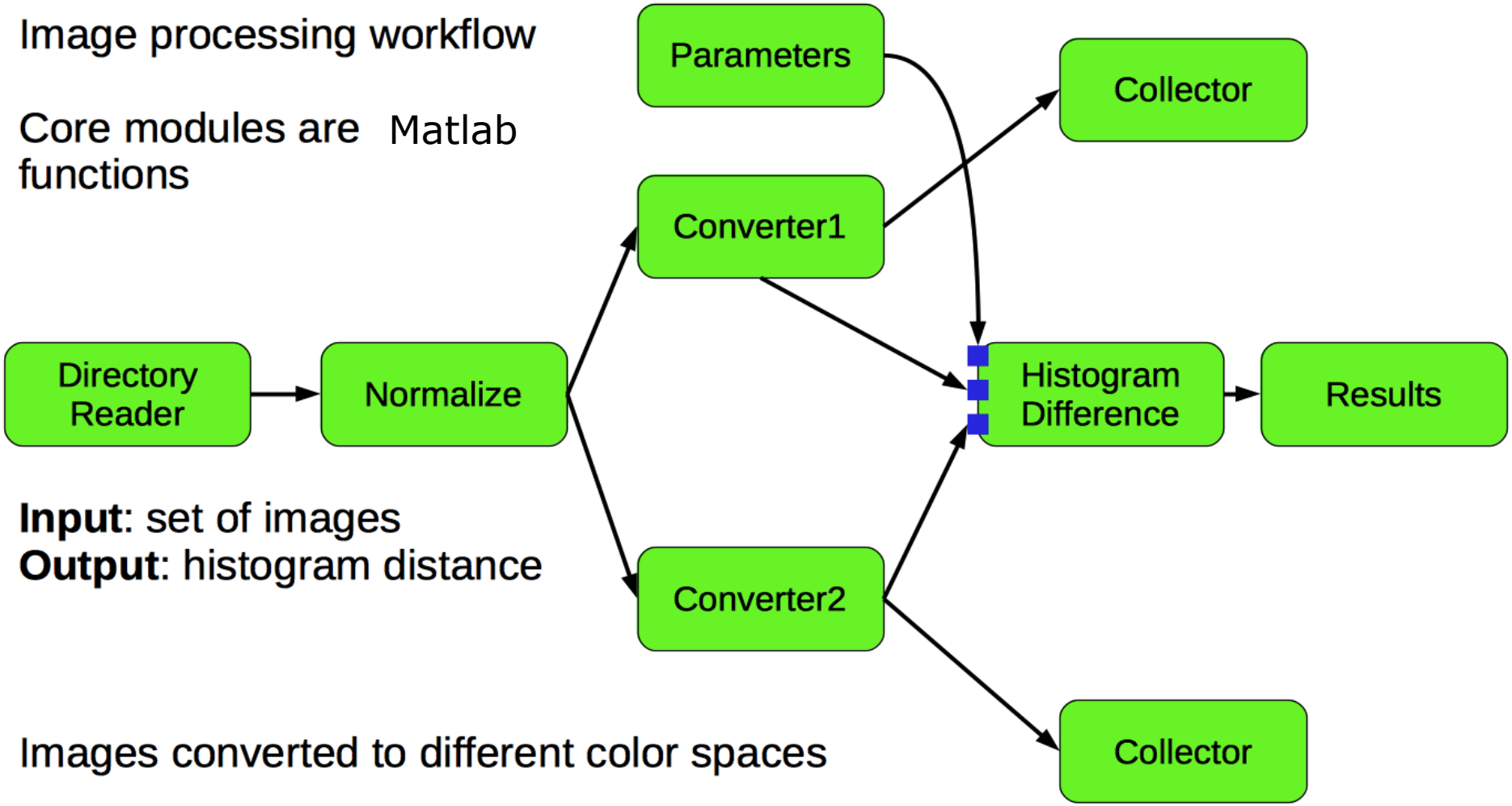


Data partitioning using message queues. **Shadow**(gray) queues allow non partitionable data to be received by all clones

Use case

Image processing workflow

Core modules are Matlab functions



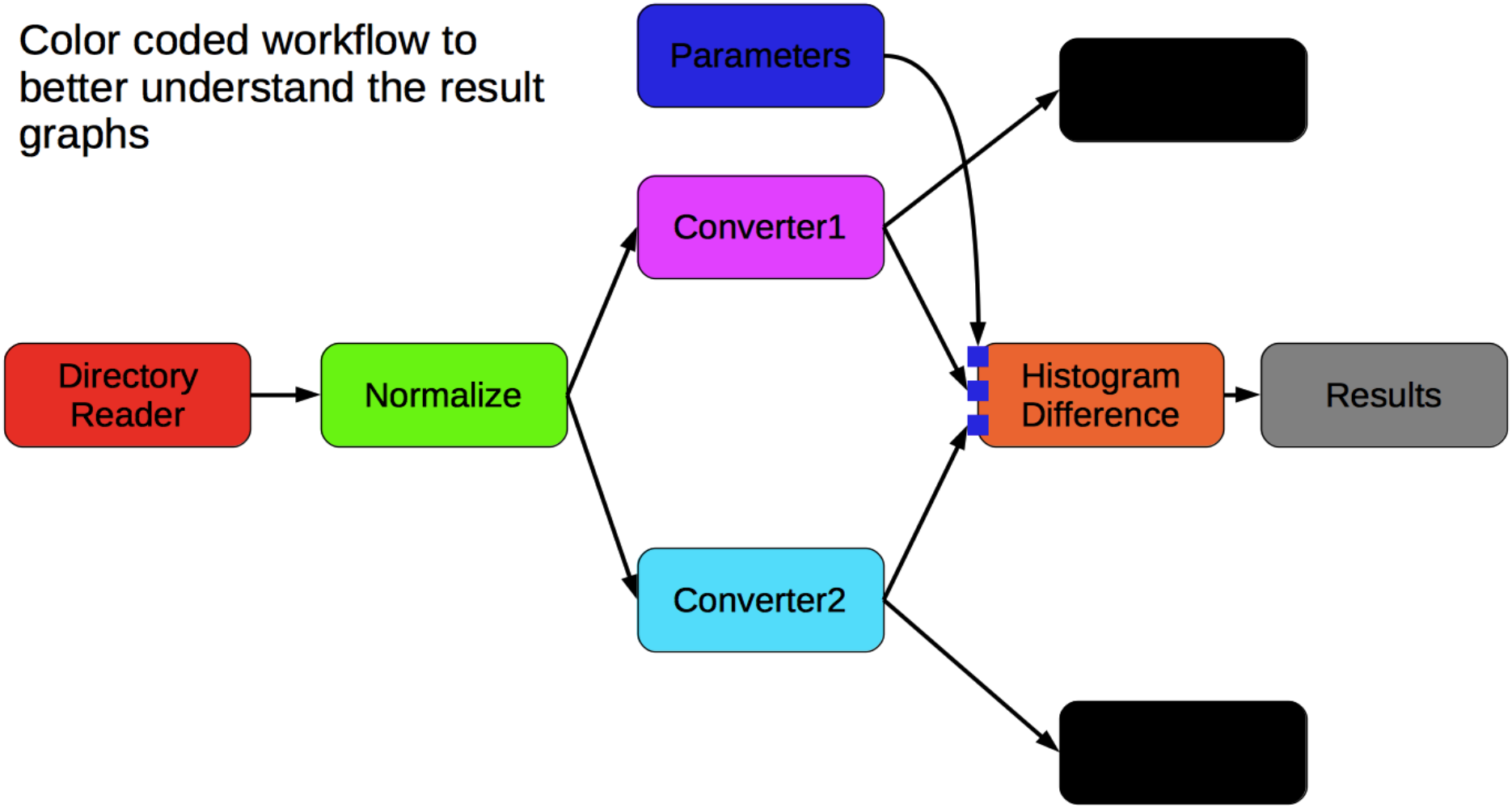
Input: set of images
Output: histogram distance

Images converted to different color spaces

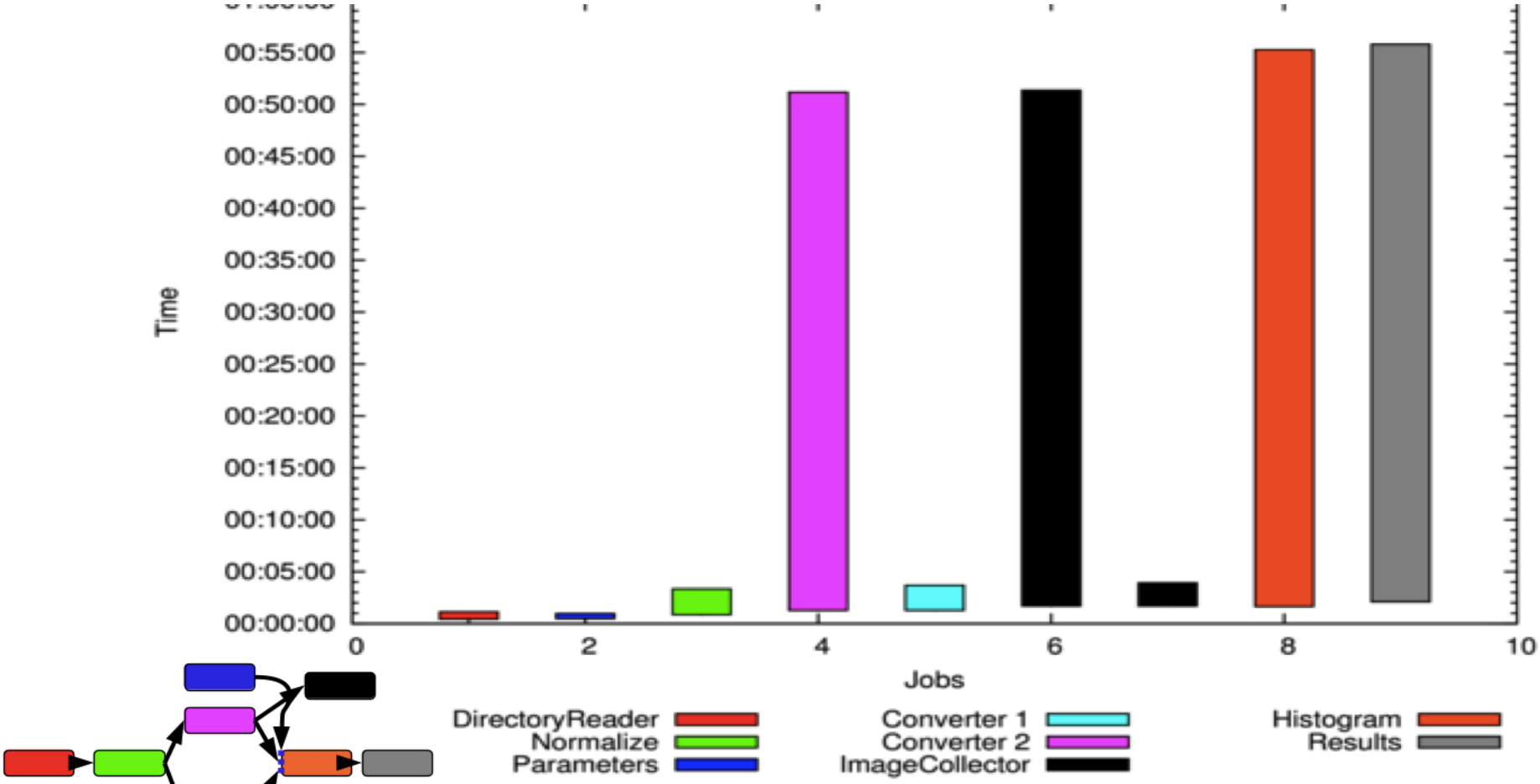
Histogram difference is calculated between color spaces

Use case

Color coded workflow to better understand the result graphs

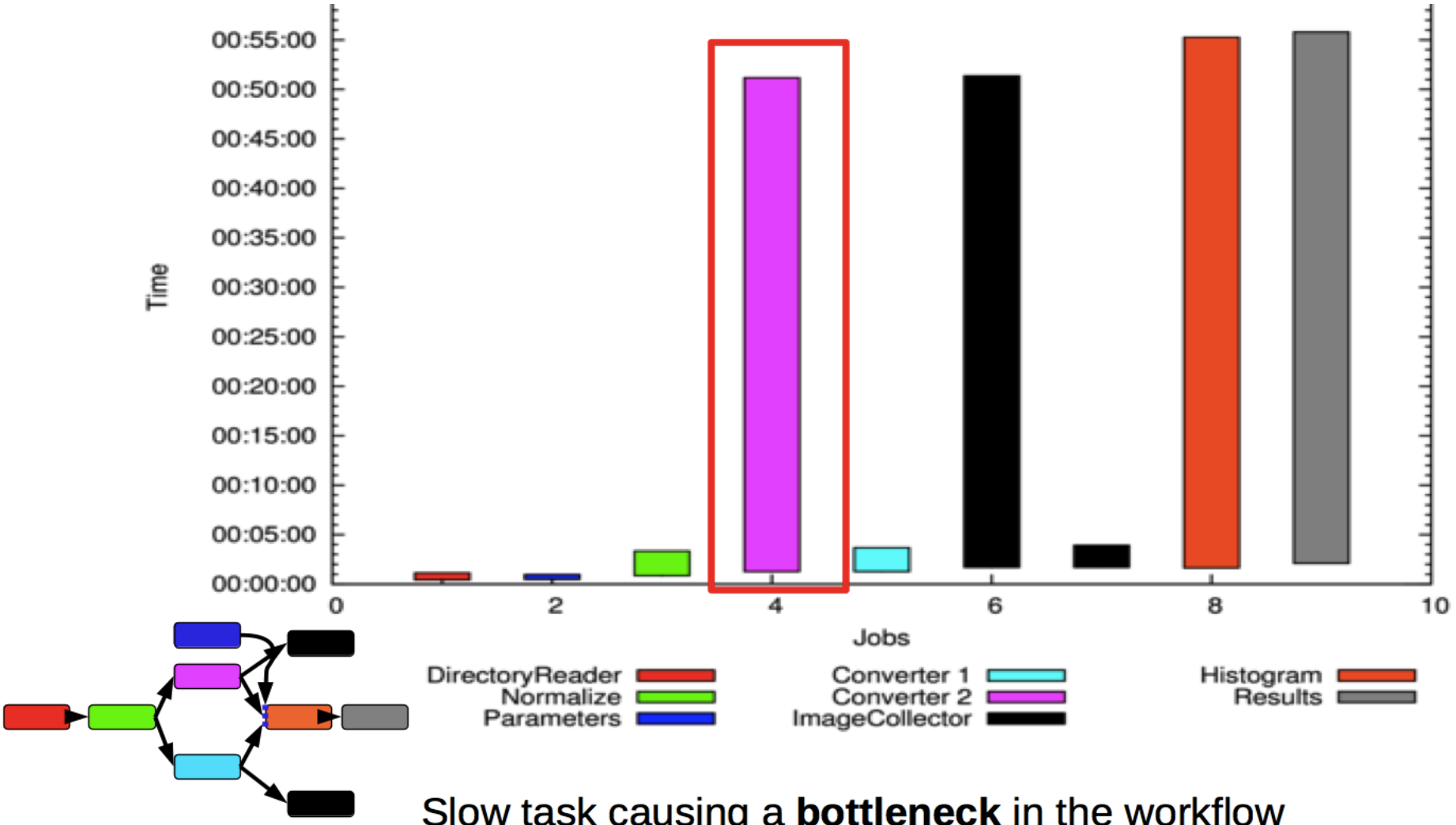


Workflow Without Scaling



Normal workflow execution. **Parallelism** only achieved through workflow structure

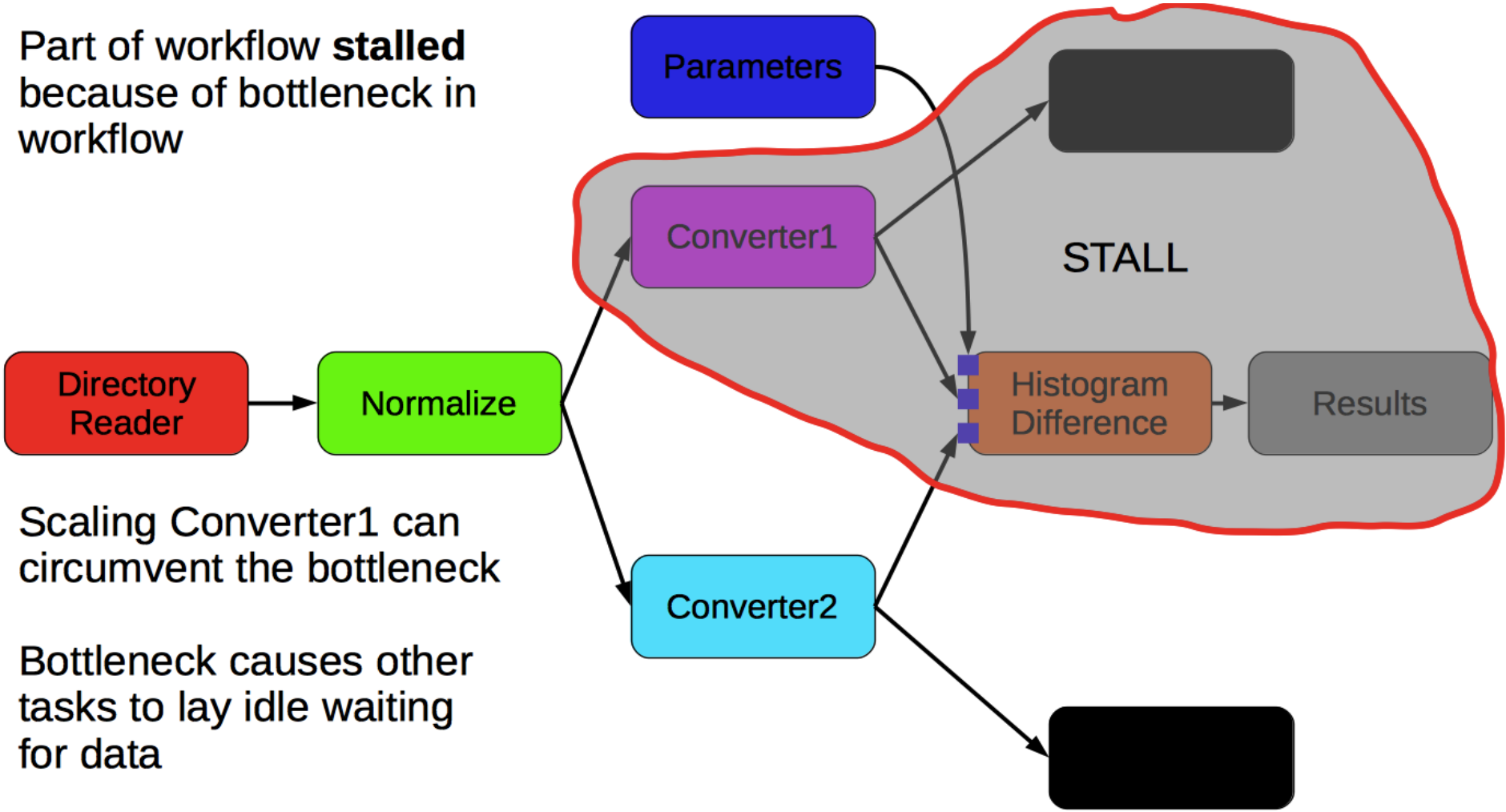
Workflow Without Scaling



Slow task causing a **bottleneck** in the workflow

Use Case

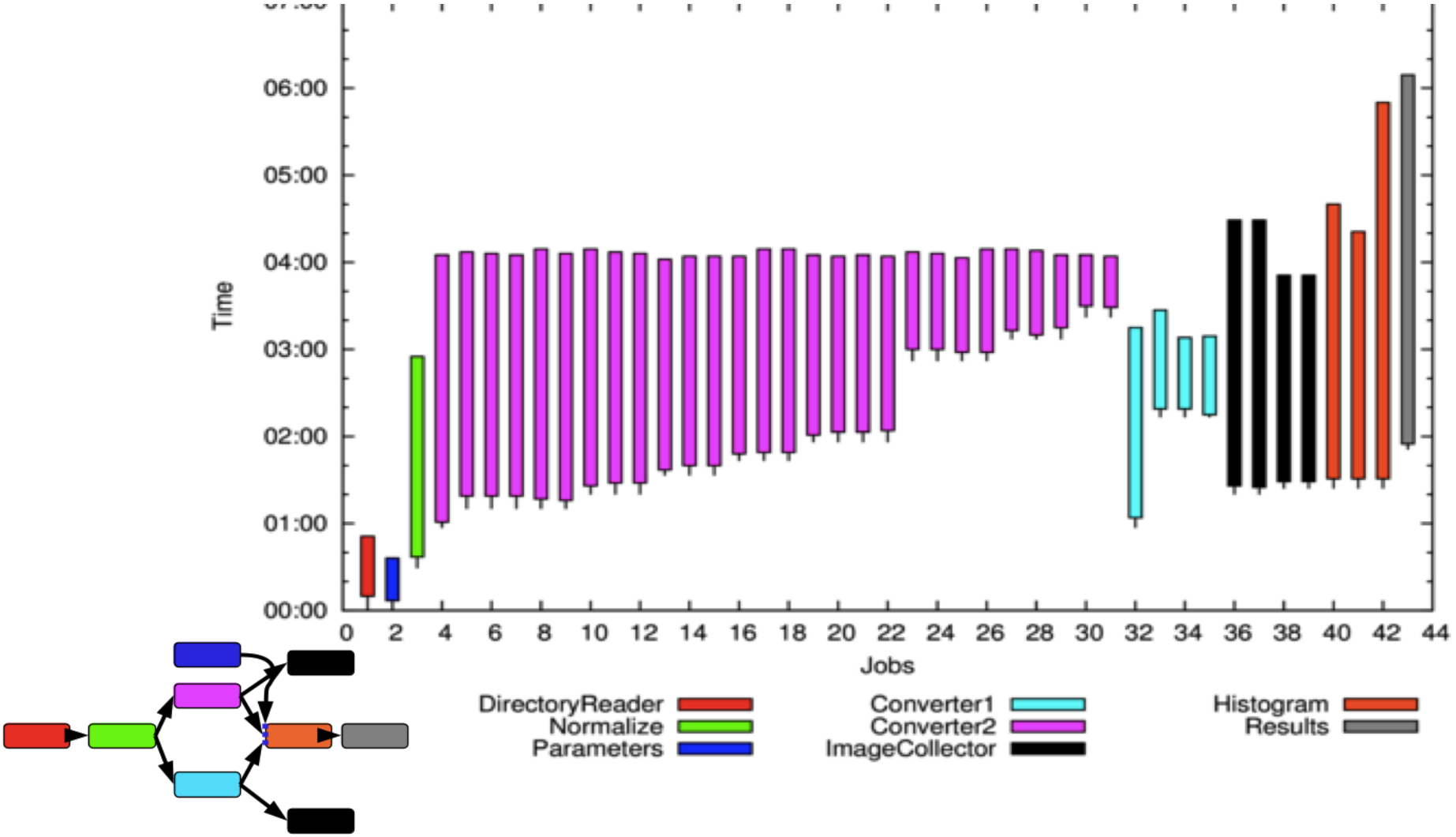
Part of workflow **stalled** because of bottleneck in workflow



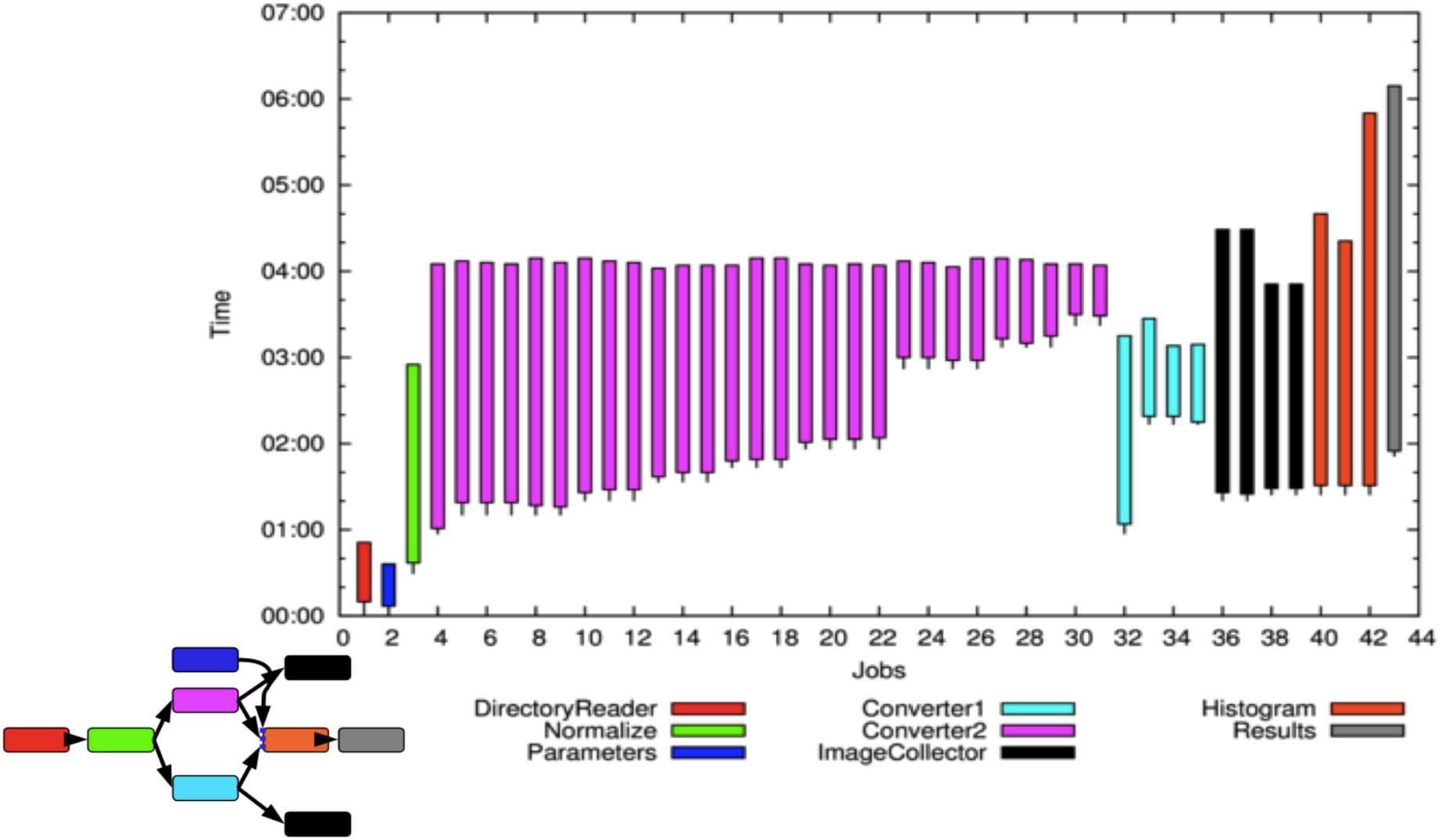
Scaling Converter1 can circumvent the bottleneck

Bottleneck causes other tasks to lay idle waiting for data

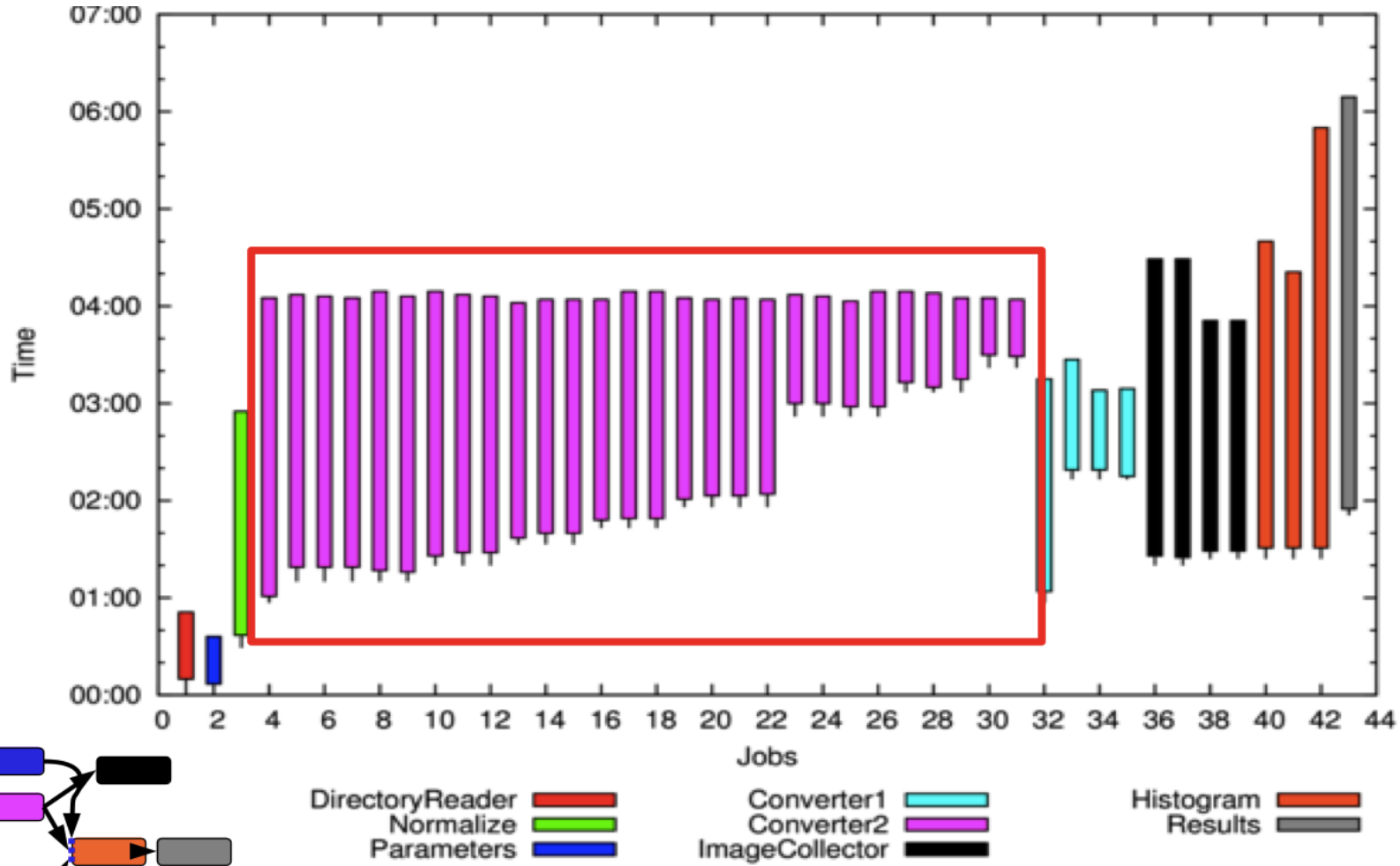
Workflow execution with Scaling

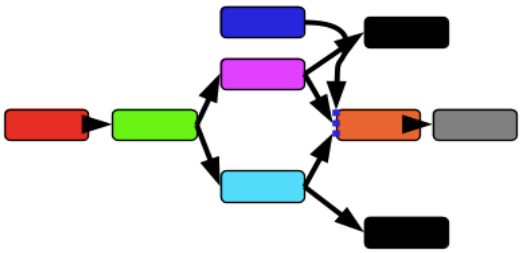


Workflow execution with Scaling

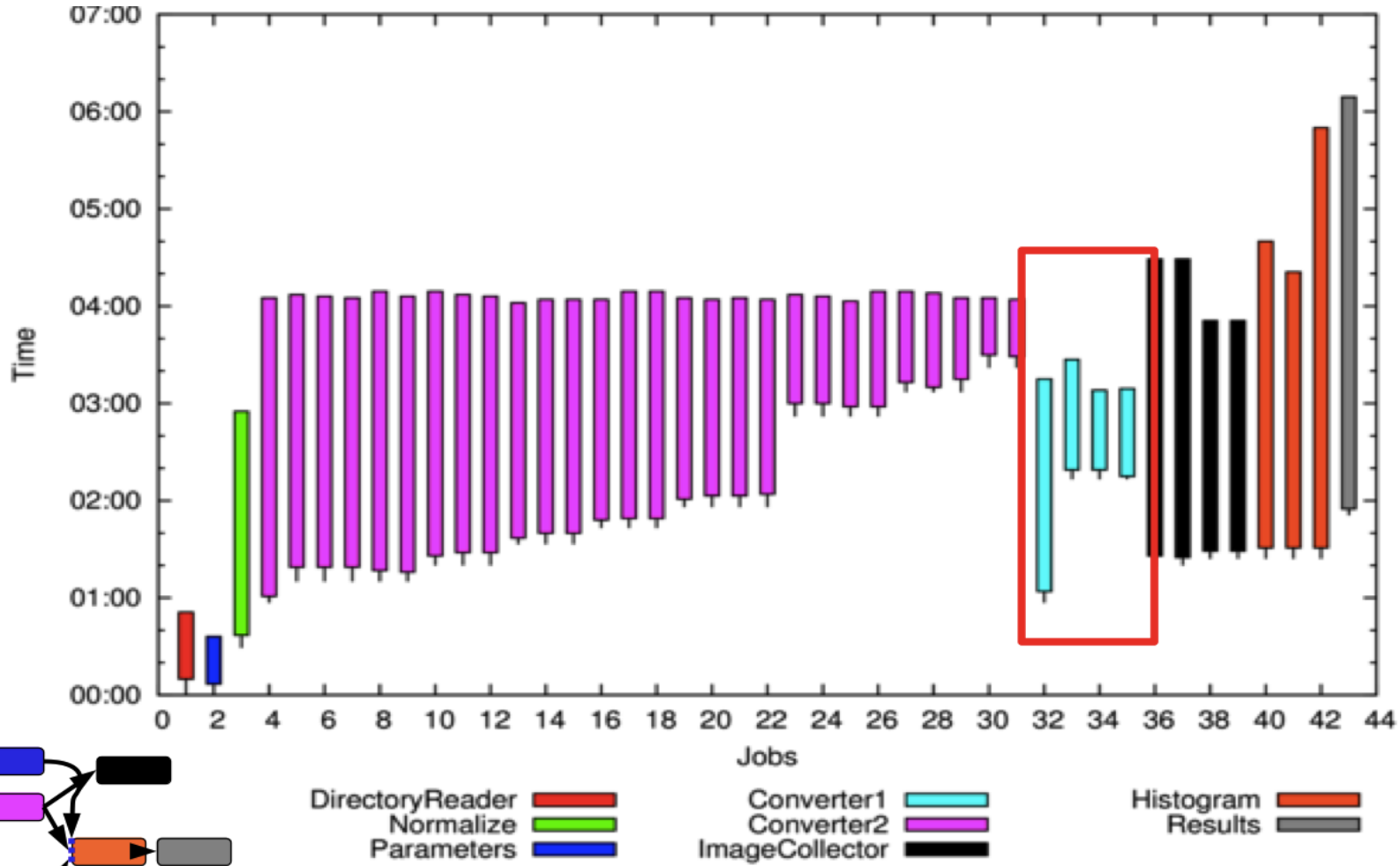


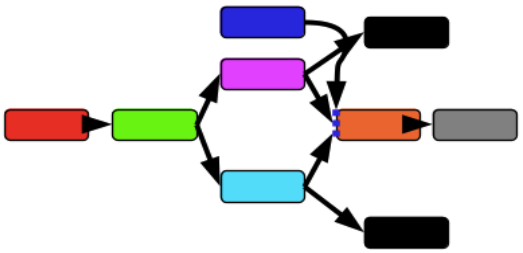
Auto Scaling Task -1



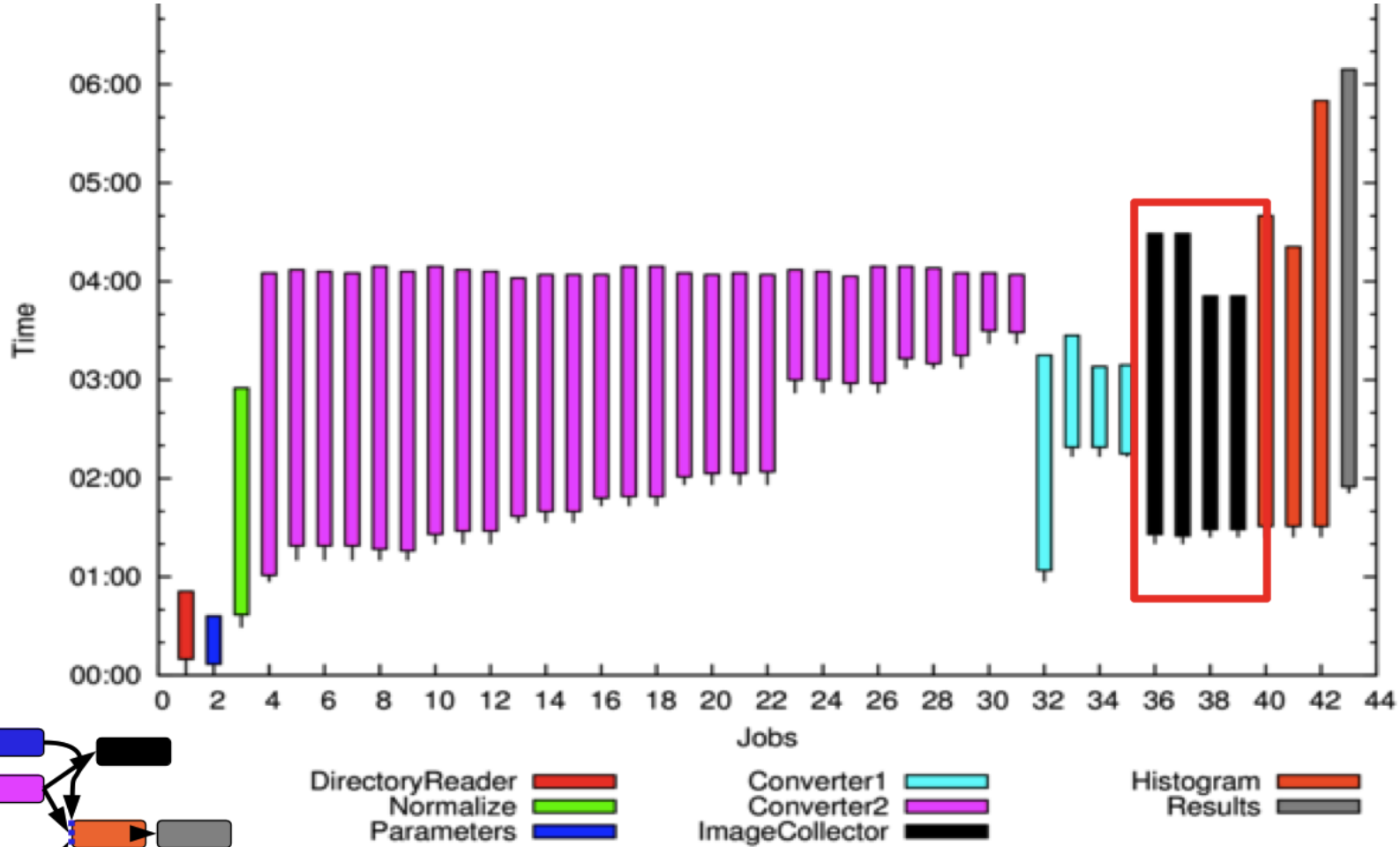


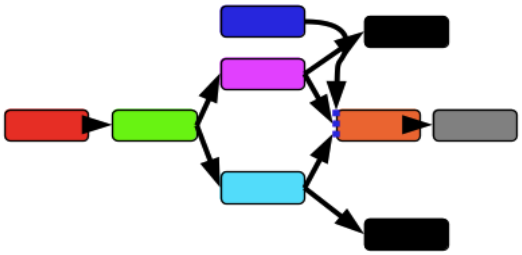
Auto Scaling Task -2





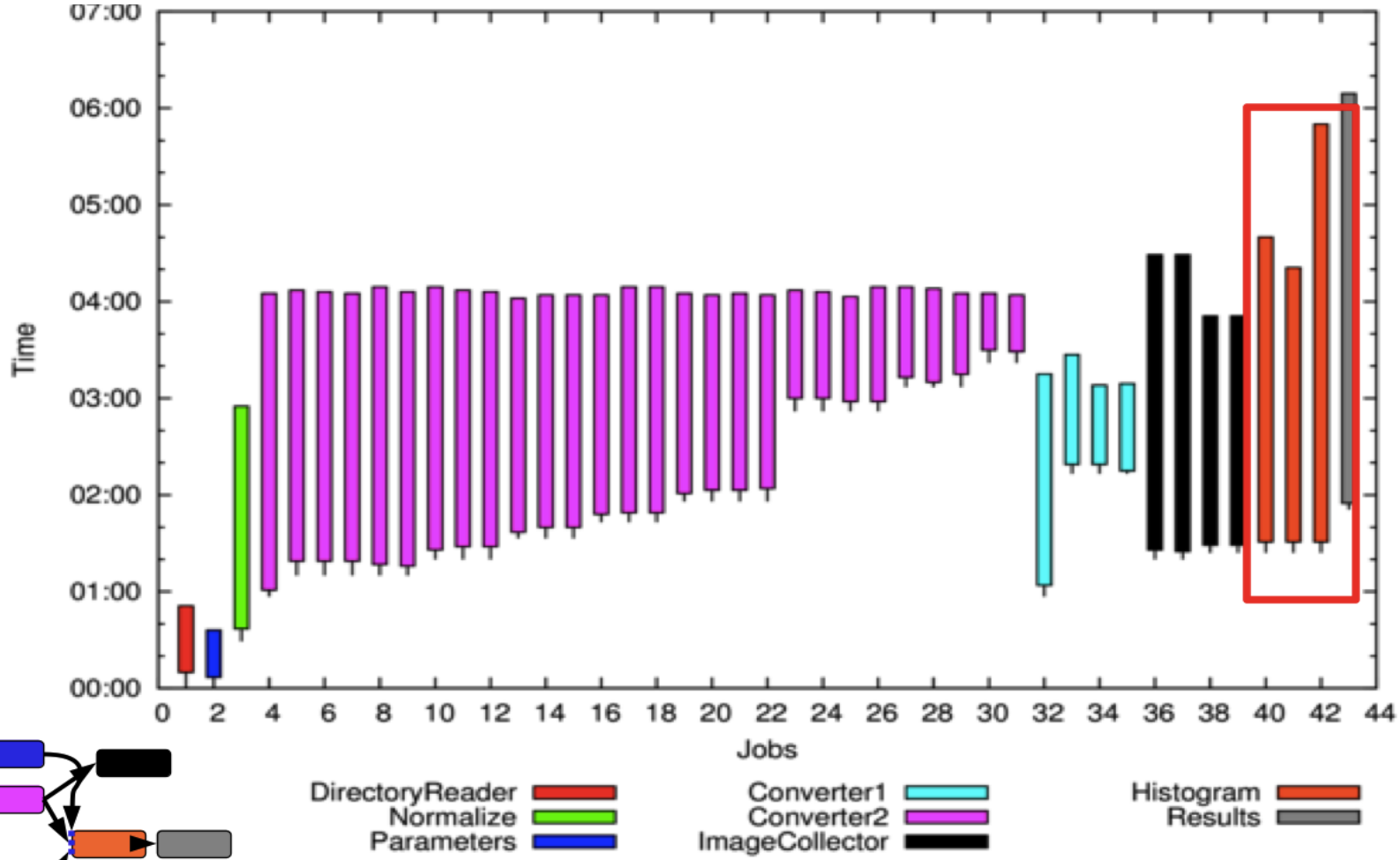
Other Scaled Task -1

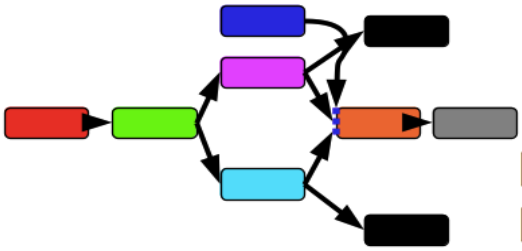




ImageCollector was set to a **fixed** amount (4)

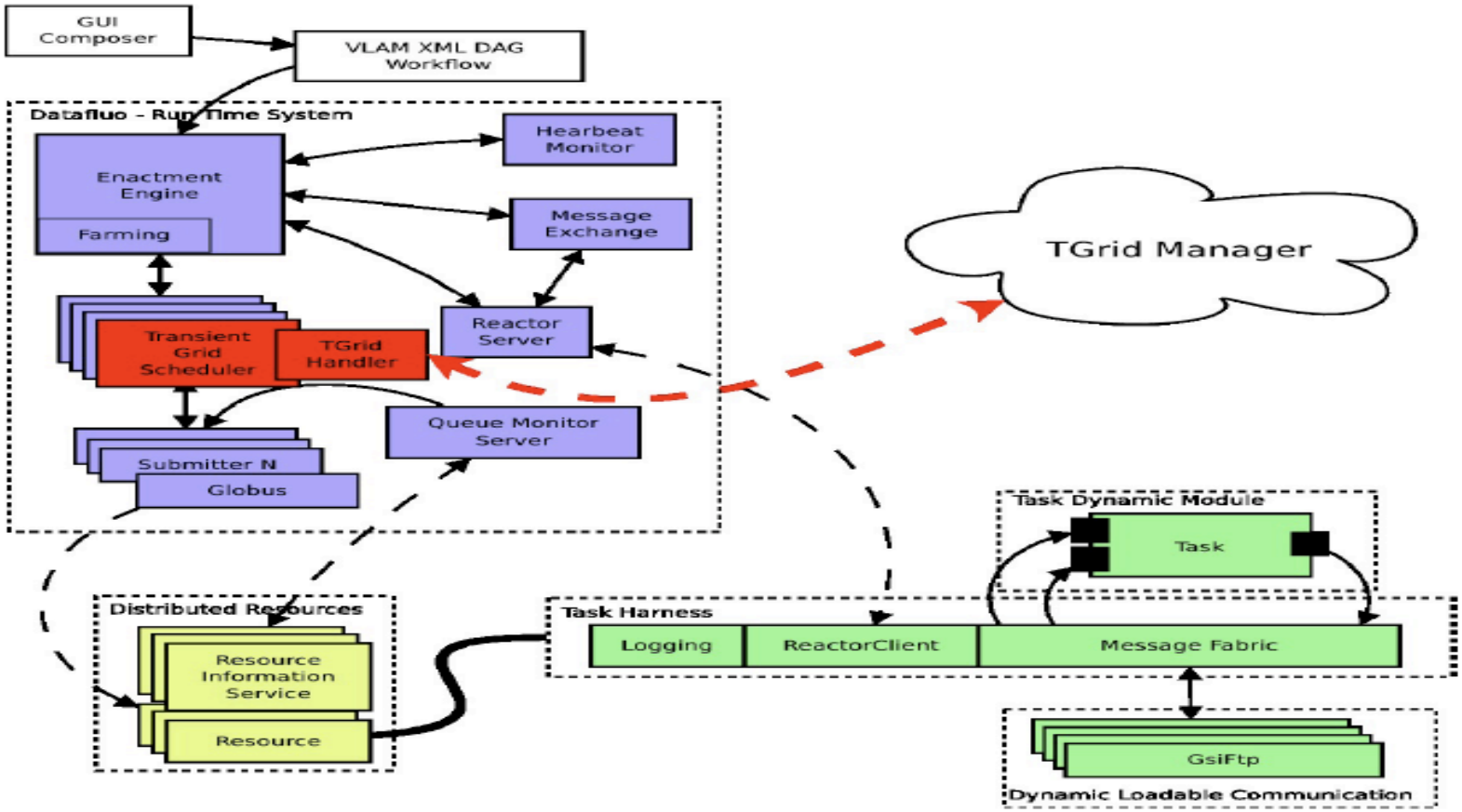
Auto Scaling Task -2





HistogramDifference was set to **one-to-one** scaling.
Each parameter generates a new task

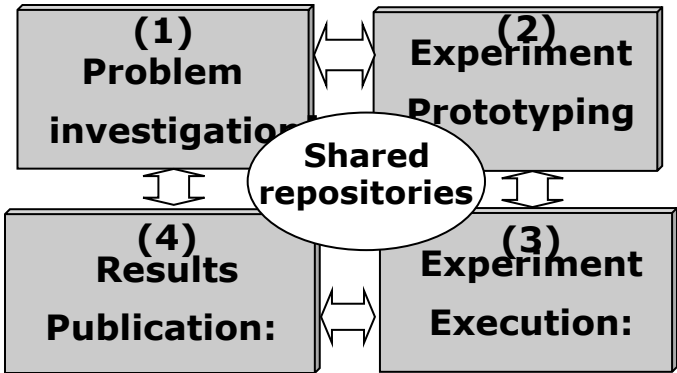
Extension to support Cloud resources



Resource on-demand using multiple cloud providers, Super-computing 2010, and SCALE 2012

Outline

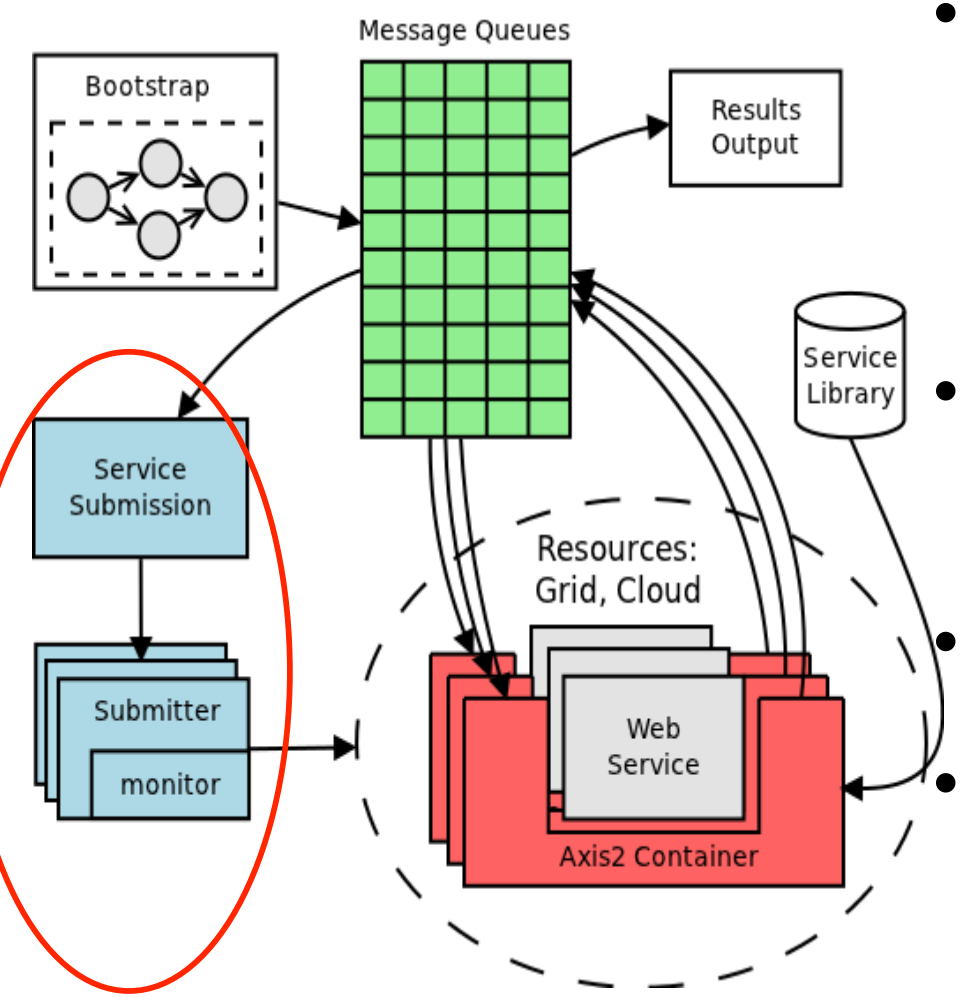
- Introduction
- Lifecycle of an e-science workflow
- Different approach to workflow scheduling
  - Workflow Process Modeling & Management In Grid/Cloud
  - Workflow and Web services (Workflow and Web services (intrusive/non intrusive)
- Provenance



Usage of Web Services in e-science

- WS offer interoperability and flexibility in a large scale distributed environment.
- WS can be **combined** in a **workflow** so that more complex operations may be achieved,
- but any workflow implementation is potentially **faced** with a *data transport problem*

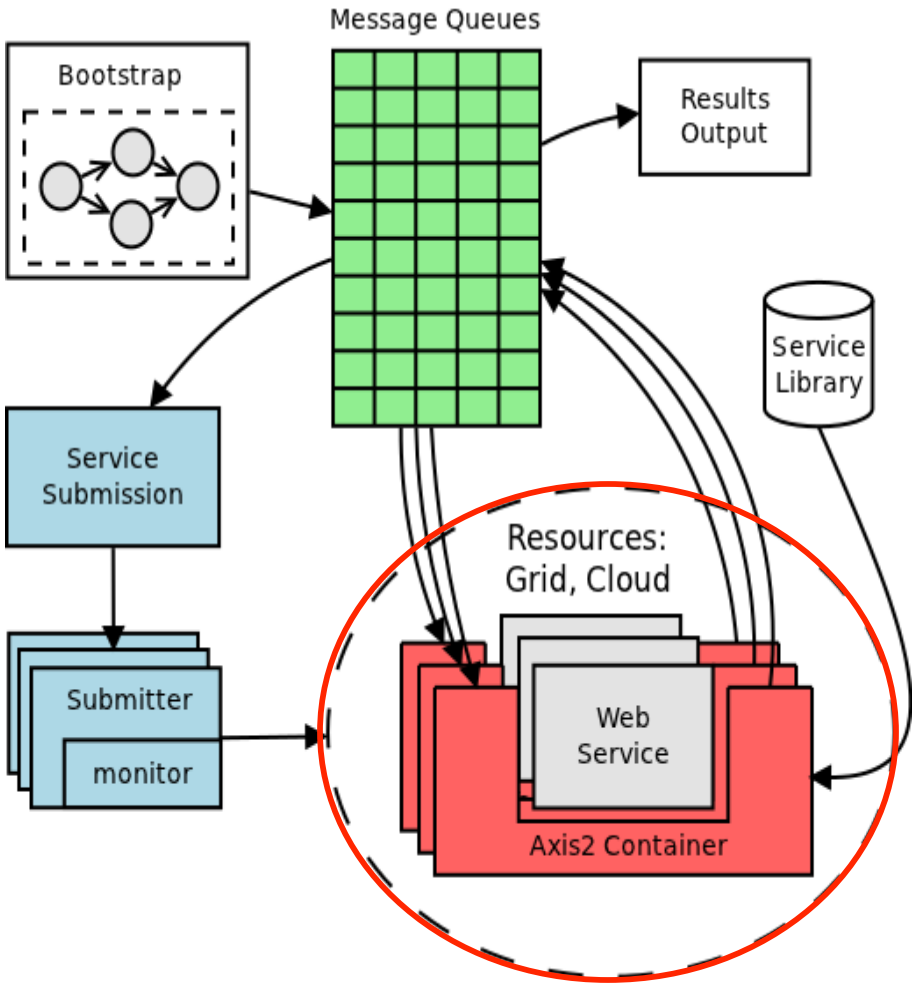
Service Submission



- Tasks/Jobs can be **queued** on the runqueue by **any** entity. The service submission **listens** on the runqueue and picks up new tasks to submit
- Resources such as Grid or Cloud are abstracted using submitters plugins
- Enabling a new resource is a matter of writing its submitter
- Service Submission performs **matchmaking** between services and resources to run on

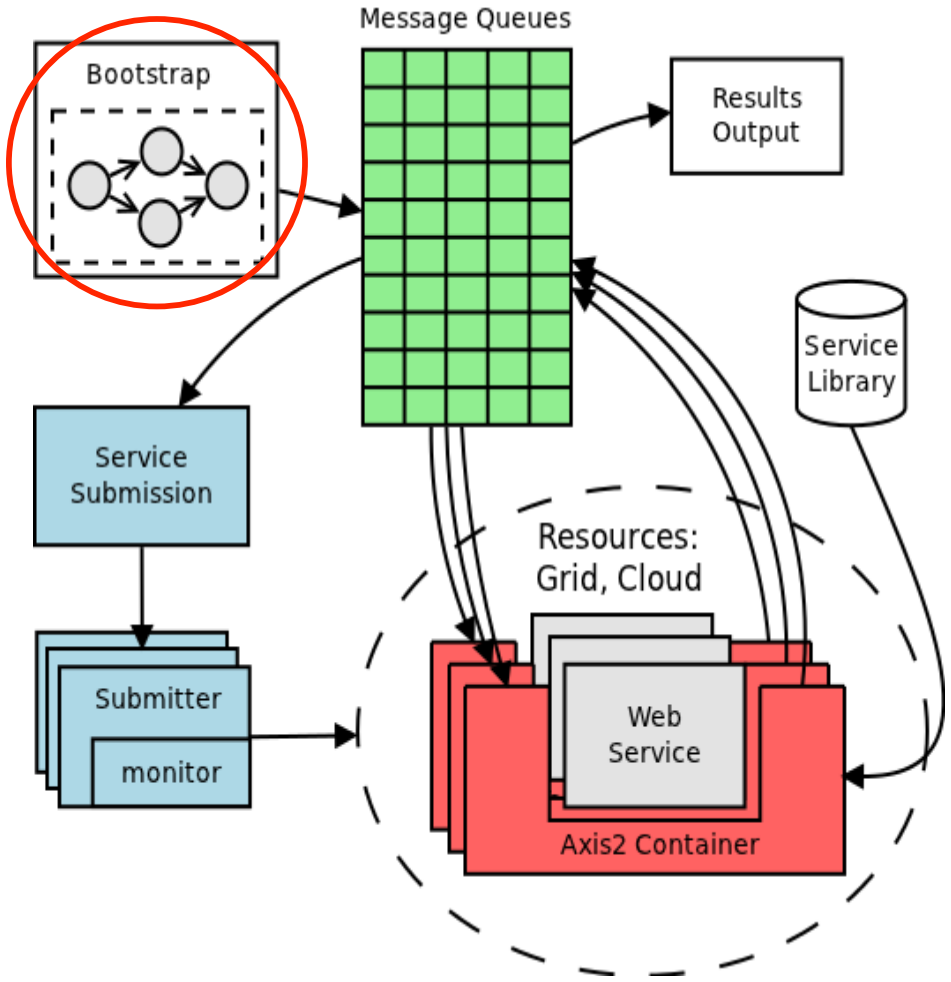
Reginald Cushing, Spiros Koulouzis, Adam S. Z. Belloum, Marian Bubak, **Dynamic Handling for Cooperating Scientific Web Services**, 7th IEEE International Conference on e-Science, December 2011, Stockholm, Sweden

Service Container Module



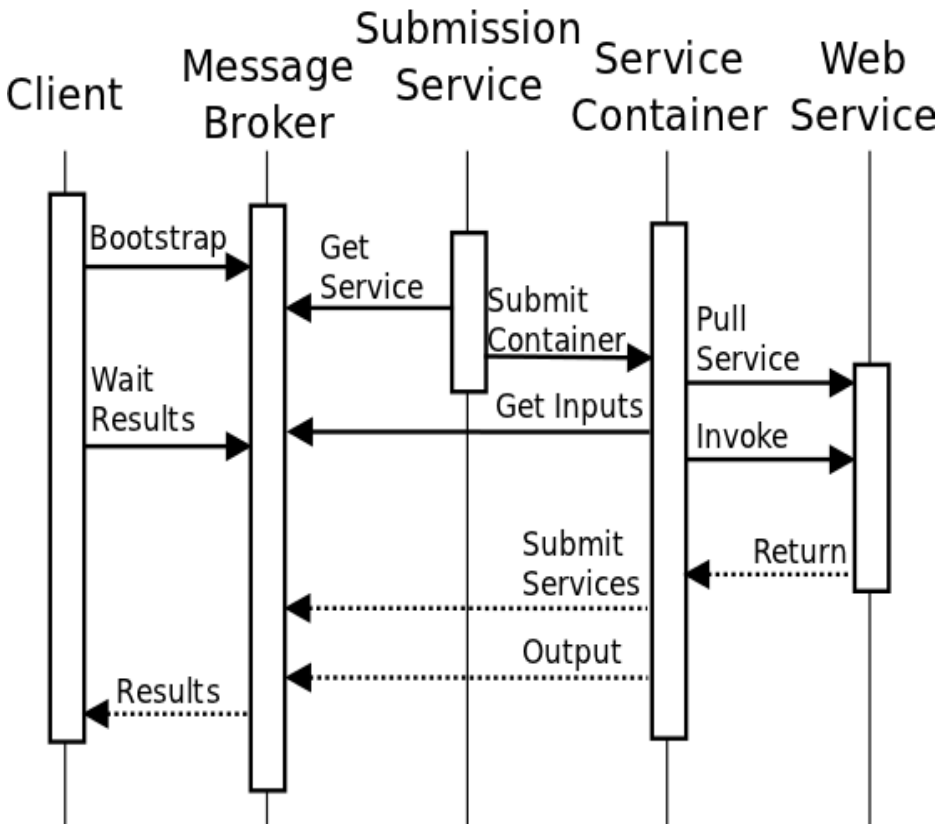
- The **service container** (**Axis2**) is the actual task that is submitted to a resource.
- The service container acts as a **pilot-job** mechanism Once active it will pull a web service to host.
- Axis2 is heavily modified to invert web service invocation from passive to active.
- **Scaling**, **orchestration** and **communication** are all handled within the service container

Bootstrapping Workflows



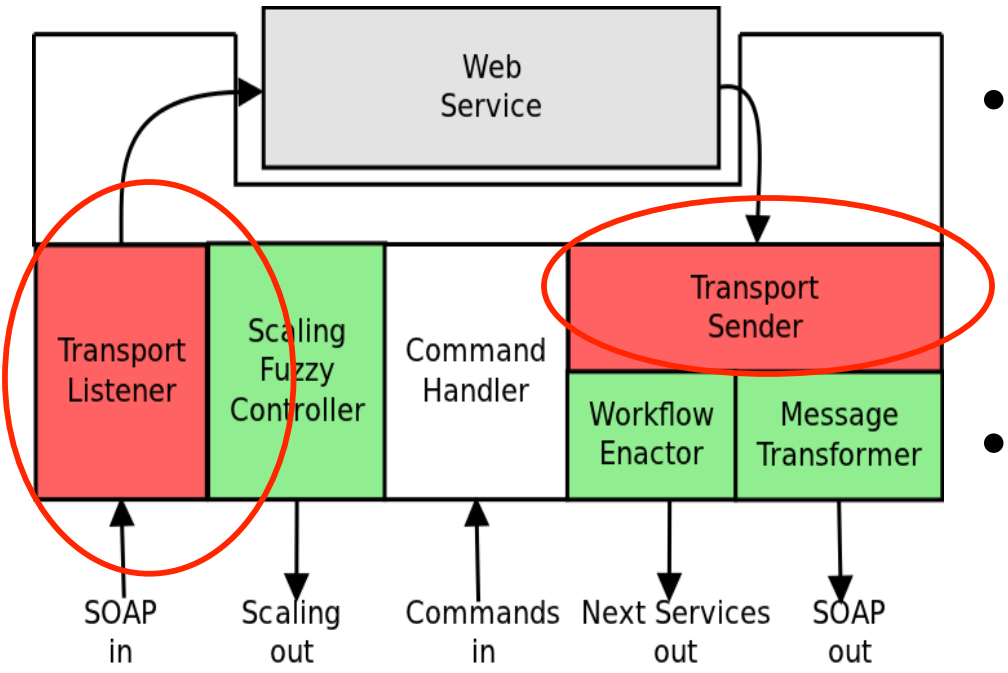
- The architecture has **no central coordinator** to orchestrate a workflow. Hence a workflow is only **bootstrapped** i.e. submit the starting services. The rest are **autonomously** scheduled by the service containers on the resources.
- The bootstrap client submits the **first** service and waits for output of the last service.

Orchestration Steps



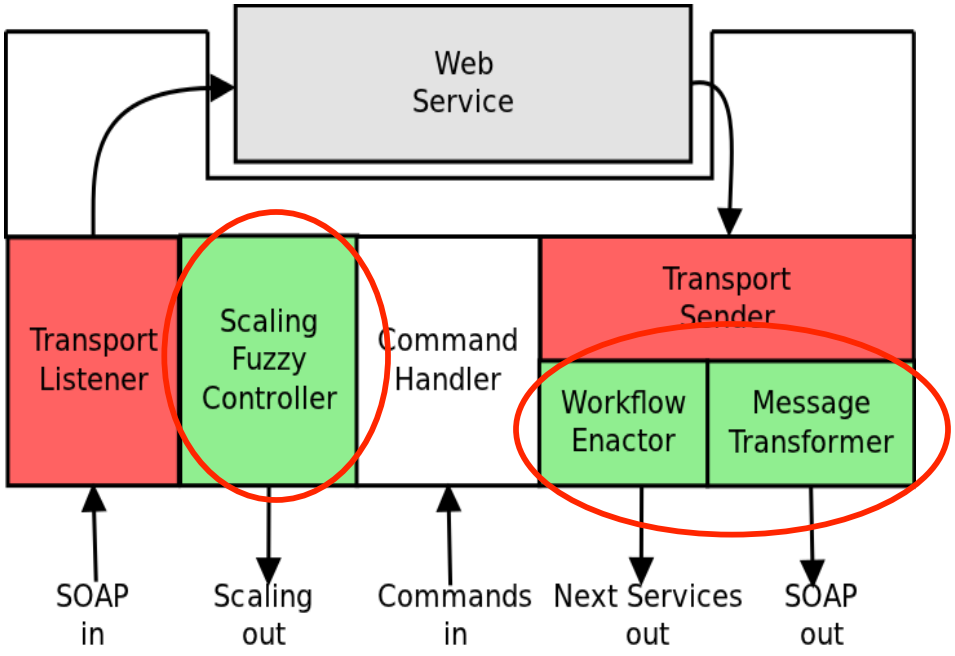
1. Workflow is **bootstrapped** by submitting the workflow entry points onto the queue
2. Submission service picks up the queued service and submits to a resource.
3. Service container starts executing on a resource
4. Service container **pulls** a web service and polls for data to be consumed by the service.
5. Service container outputs data to the next service
6. Service container queues the next service if none exist

Service Container - Transport



- **Transport handler** requests SOAP from message broker queues instead of passively listening for HTTP
- **Pull model** allows web services to “bypass” firewalls and thus can be deployed within networks
- **Transport Sender** picks up the return SOAP message and sends it to the message broker

Service Container - Control



- **Fuzzy controller** implements auto-scaling routines
- **Workflow enactor** implements autonomous orchestration which makes a **central coordinator redundant**

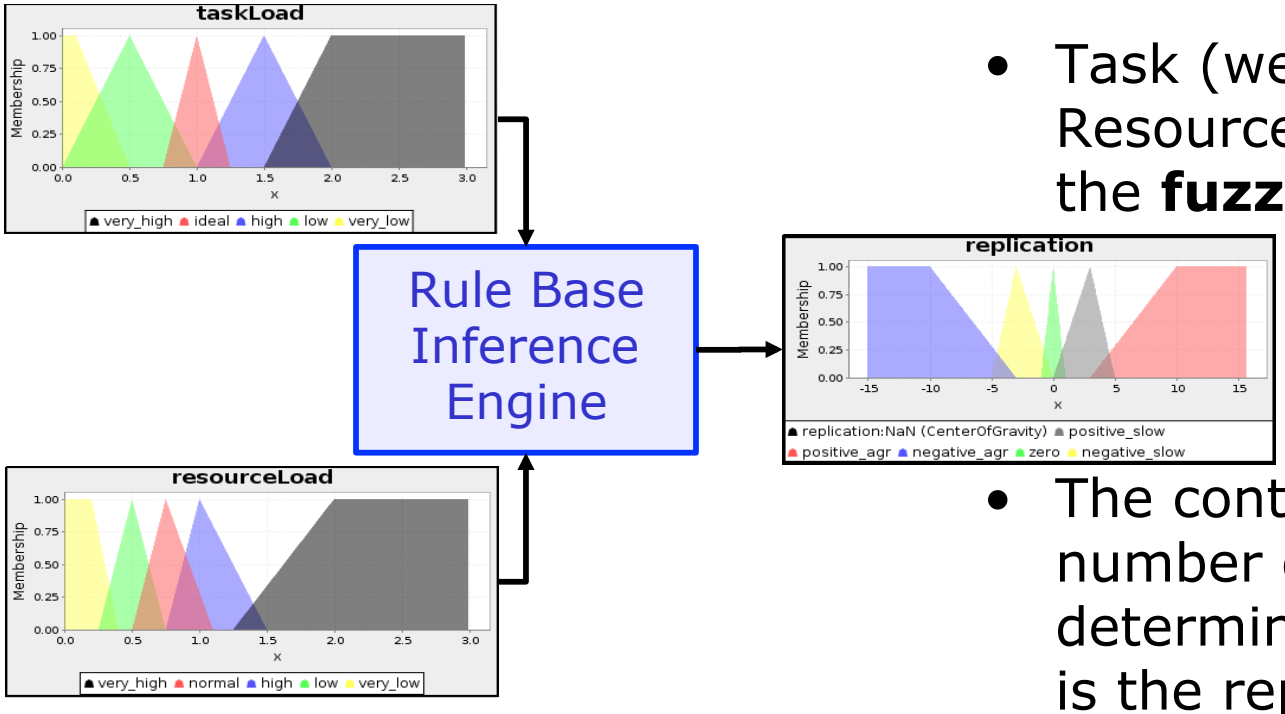
- **Message transformer** transforms a SOAP output to SOAP input for other services in the workflow
- The message transformer allow back-to-back service communication

Resource management

- Within a single workflow services are **competing** for resources.
- Scaling one service without any regard to the whole workflow may starve parts of the workflow and hamper progress
- It would be ideal to have a mechanism to **greedily** consume resources if **no one** is using them but **donate back** resources once they are requested.

Fuzzy controller tries to do just that.

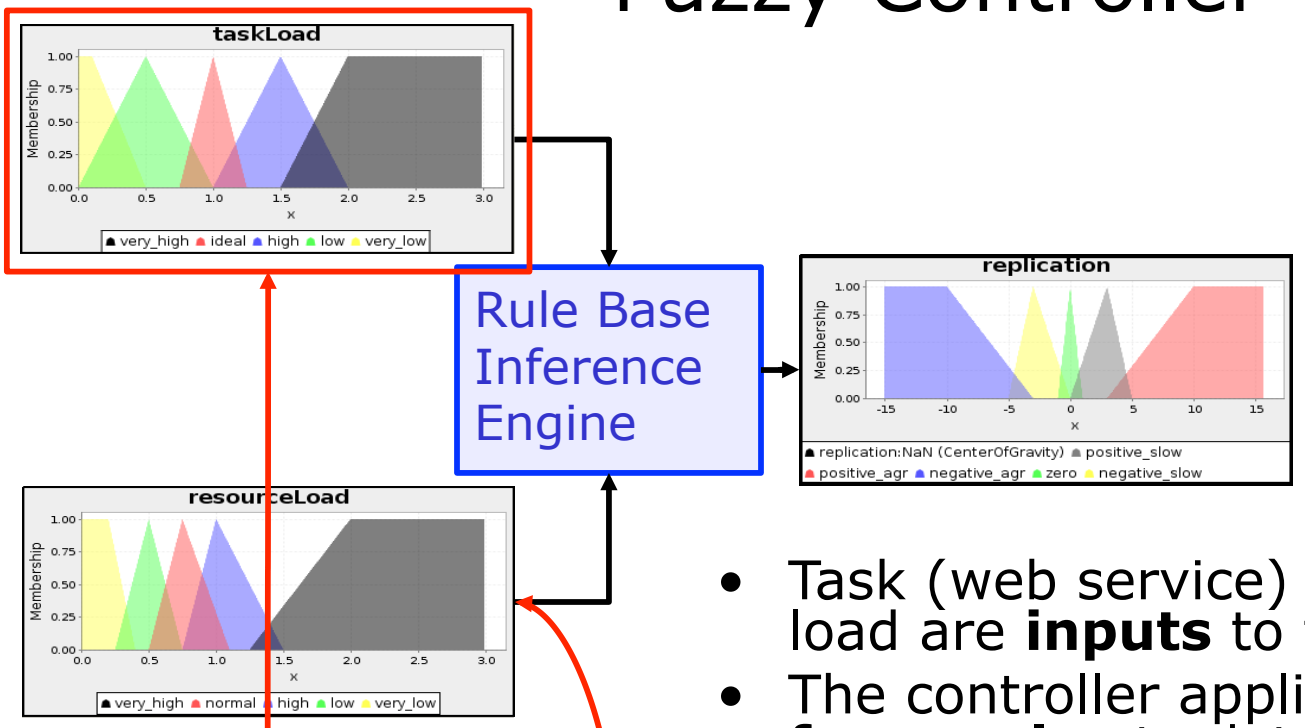
Fuzzy Controller



- Task (web service) load and Resource load are **inputs** to the **fuzzy** controller.
- The controller applies a number of **fuzzy rules** to determine the output which is the replication factor.

- **IF** taskLoad **IS** very_high **AND** resourceLoad **IS** very_low **THEN** replication **IS** positive_aggressive.
- **IF** taskLoad **IS** very_low **AND** resourceLoad **IS** high **THEN** replication **IS** negative_aggressive.

Fuzzy Controller

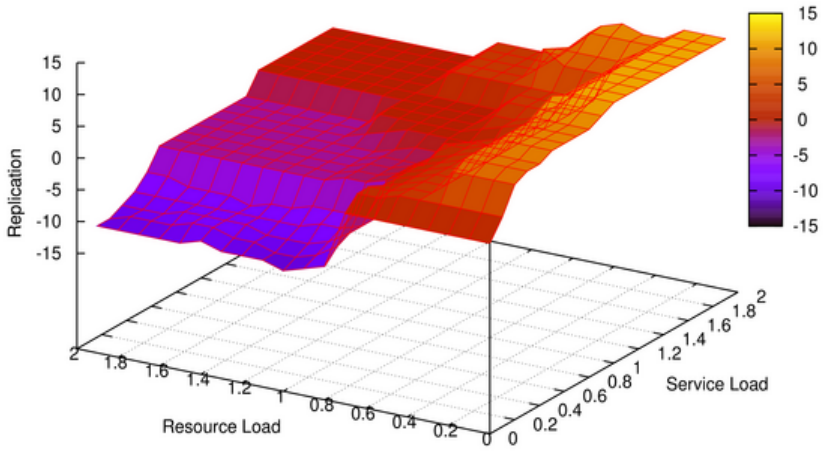


- Task (web service) load and Resource load are **inputs** to the **fuzzy** controller.
- The controller applies a number of **fuzzy rules** to determine the output which is the replication factor.

- **IF** taskLoad **IS** very_high **AND** resourceLoad **IS** very_low **THEN** replication **IS** positive_aggressive.

- **IF** taskLoad **IS** very_low **AND** resourceLoad **IS** high **THEN** replication **IS** negative_aggressive.

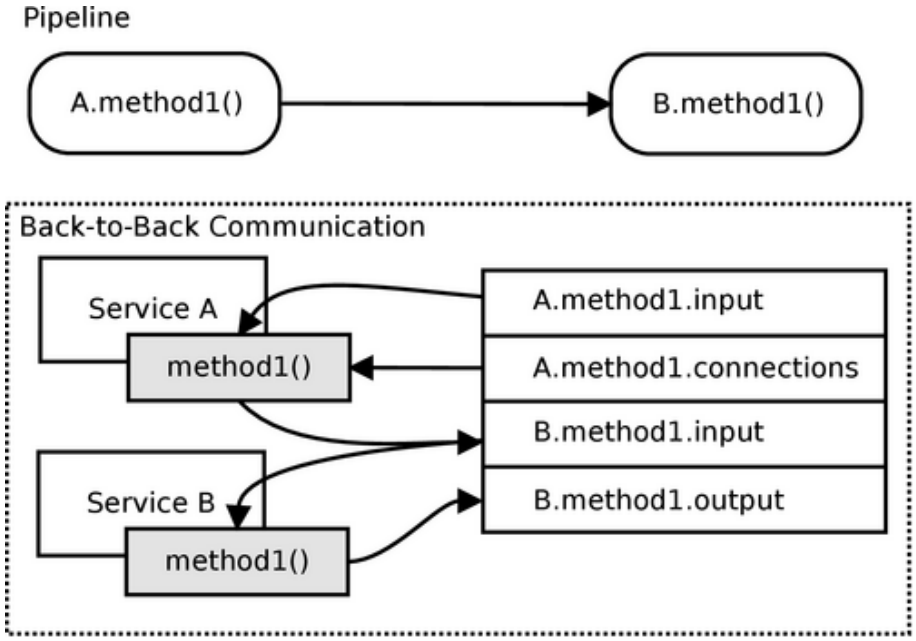
Fuzzy Rule Map



- Service load is based on the amount of **data** being **queued** on the service and the **time quantum** for the service to run
- The service container **continuously** monitors the **data processing rate** and estimates the computation time needed to process all the queued data within a time frame of the data and the processing time are directly proportionate. This might not be the case for all problems.

The **estimated processing time** and the **time quantum** given by the resource for executing the service are used to **derive** the **service load**. Thus a service load of 2 means that it will take twice as much time as the allocated quantum to process the data.

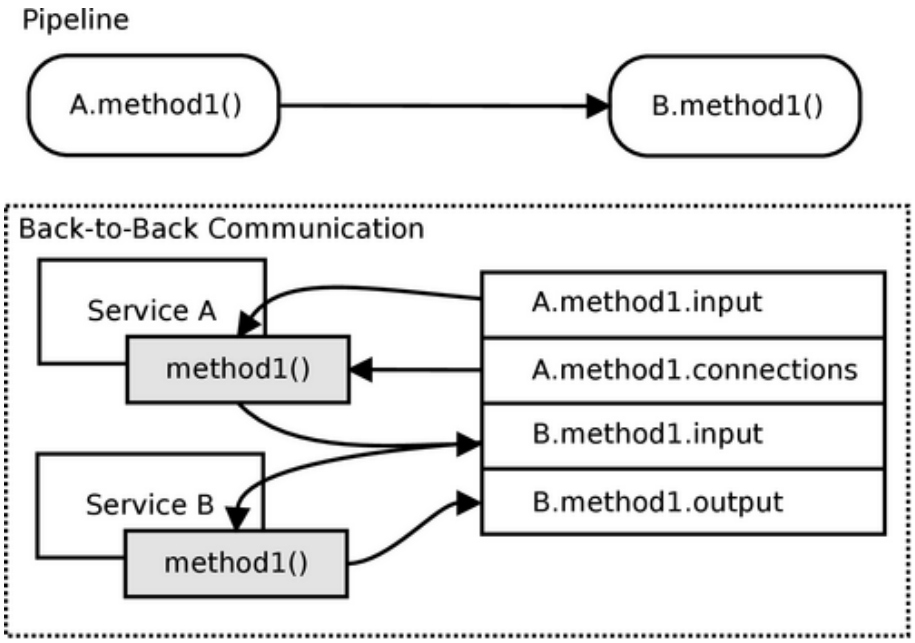
Back-to-Back Communication



- Back-2-Back communication allows web services to **communicate directly** without the need for an **intermediate client**.
- This is achieved through the message broker which exposes **dedicated connections queues**.

Foreach connection **in** A.method1.connections
    SOAPTemplate = getTemplate(connection);
    destinationQueue = getDestination(connection);
    newSOAP = transformSOAP(A.method1.output, SOAPTEmplate);
    write(newSOAP, destinationQueue);

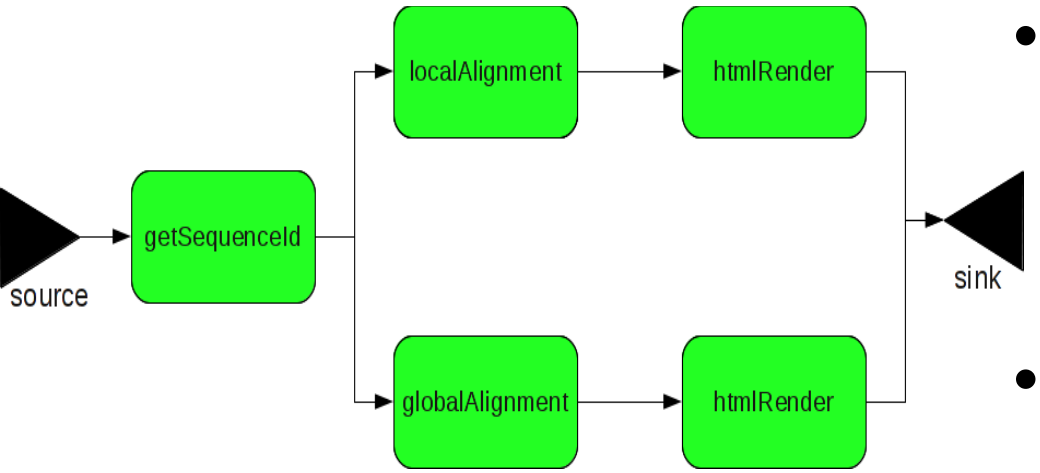
Autonomous Orchestration



- The service container can query the message broker to deduce if and instance of **B** is running.
- If no instance of **B** is running, the service container for **A** submits **B** to the runqueue.
- Service containers are myopic

```
Foreach connection in A.method1.connections
        SOAPTemplate = getTemplate(connection);
        destinationQueue = getDestination(connection);
        newSOAP = transformSOAP(A.method1.output, SOAPTEmplate);
        write(newSOAP, destinationQueue);
        If not active(destinationQueue)
                submit( getService(connection) );
```

Use Case

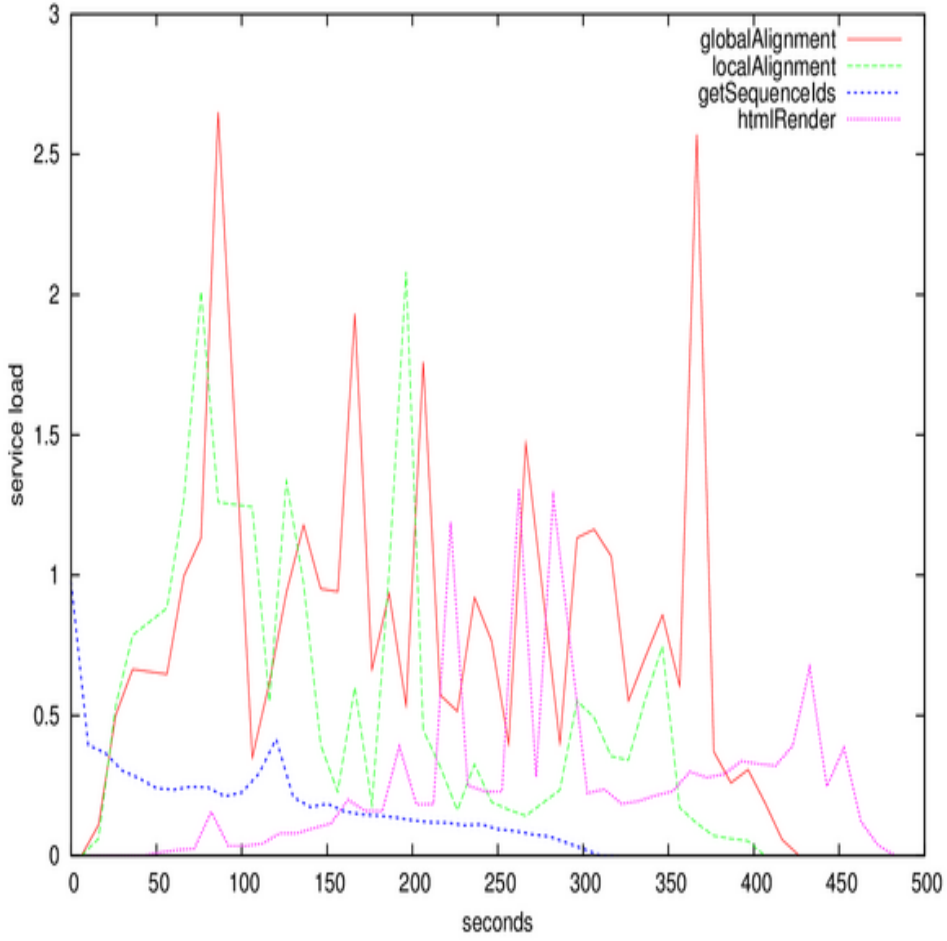


- Workflow with 2 pipelines. The pipelines perform sequence alignments using data from UniProtKB
- Each pipeline performs 22500 alignments i.e. 45100 total alignments in all

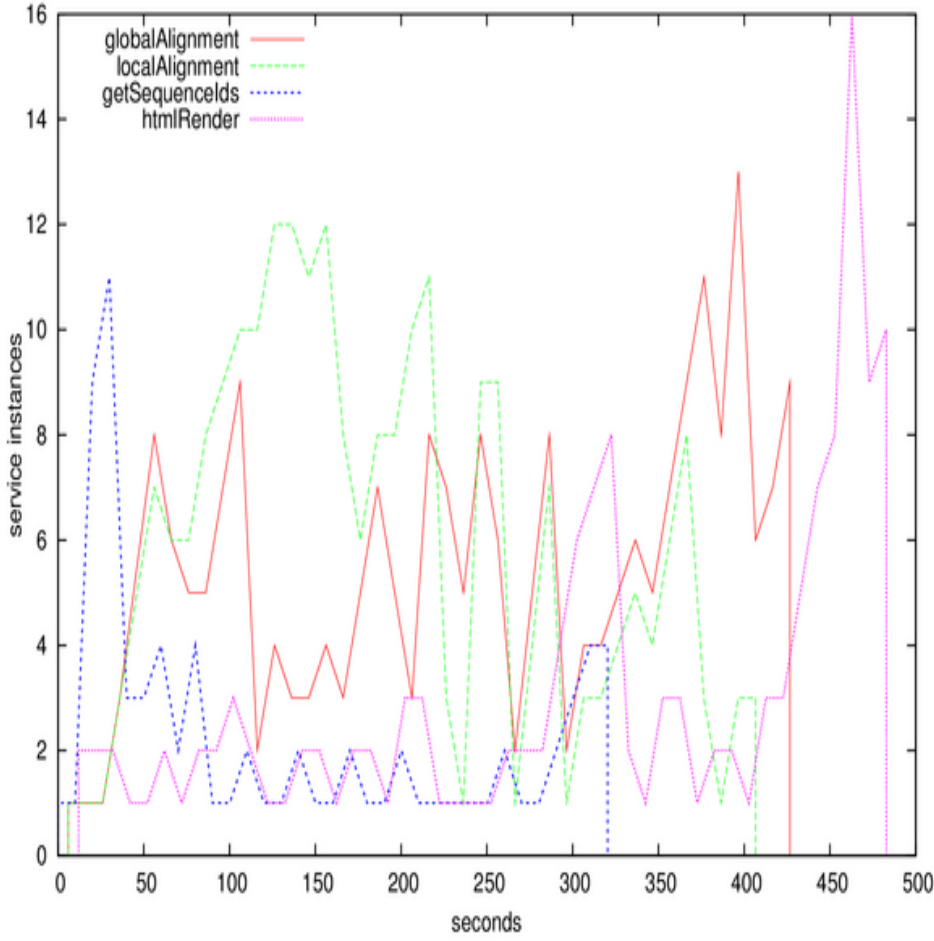
- All modules are standard web services which are hosted in the modified Axis2 container
- The alignments where performed using BioJava api
- Source and sink are part of the bootstrapping sequence. Source submits the getSequenceId service while sink waits for output from the htmlRenderer
- The Distributed ASCI Computer 3 (DAS3) was used as the resource pool.

www.uniprot.org

Evaluating Auto-Scaling

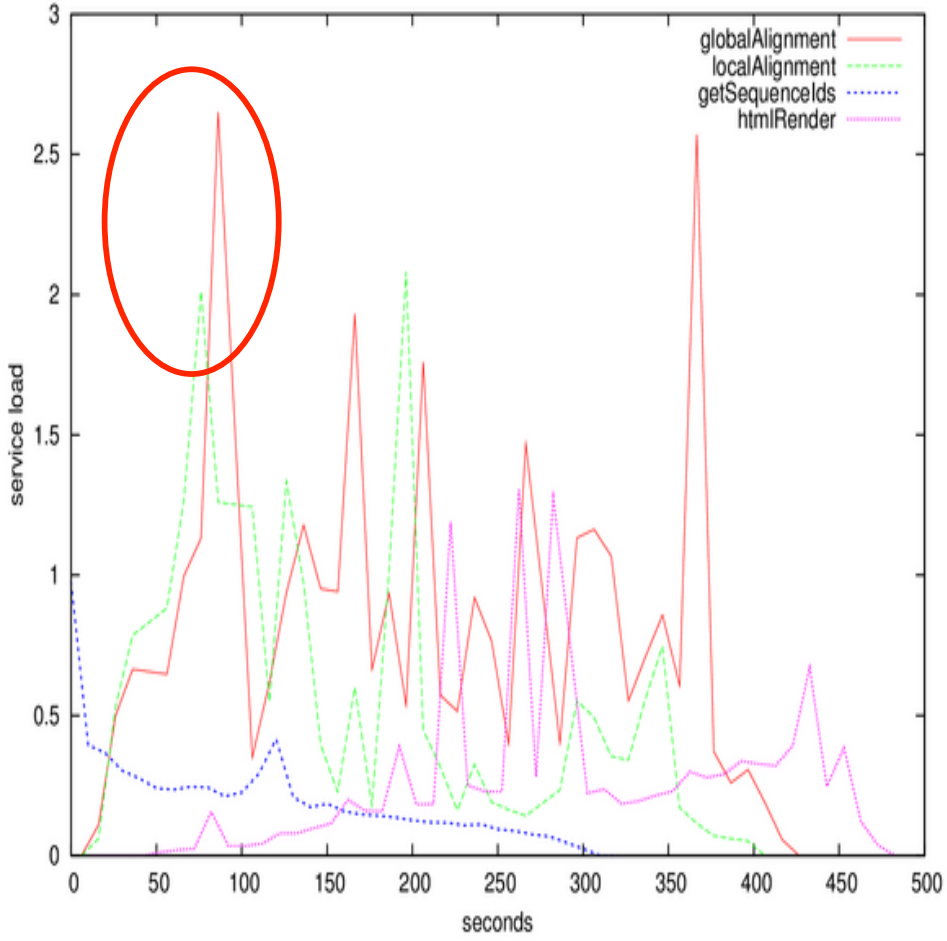


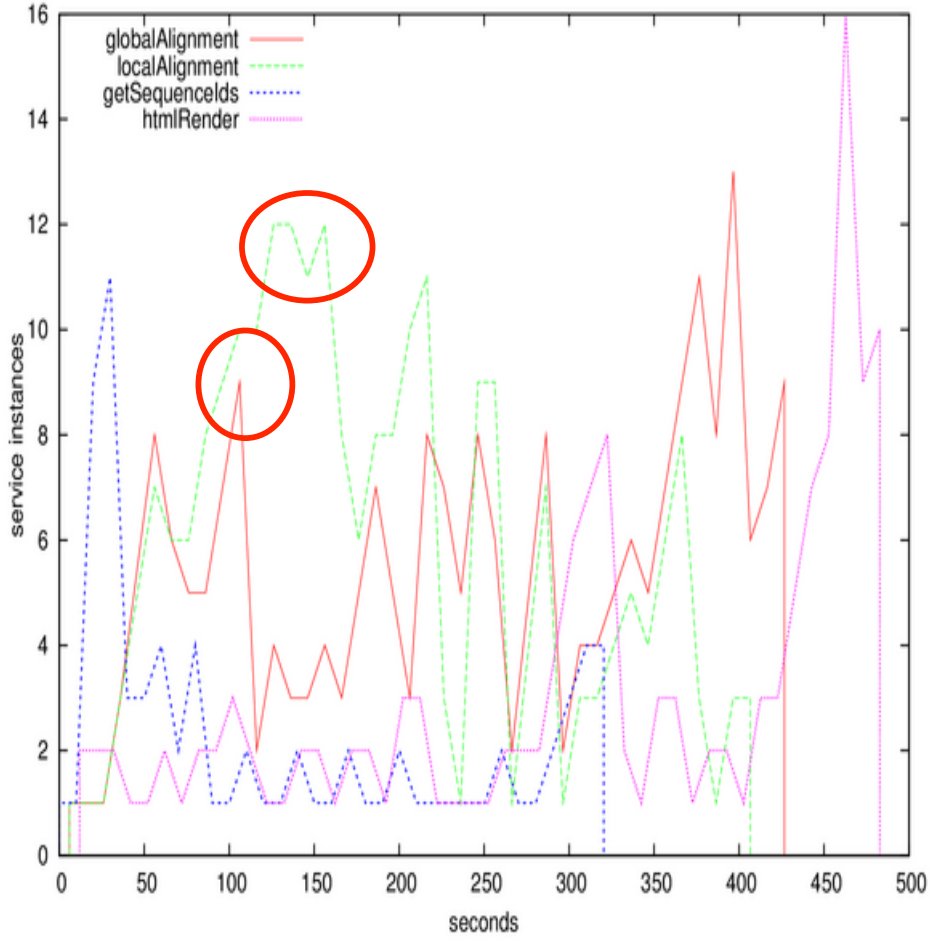
Service Load



Running Service instances

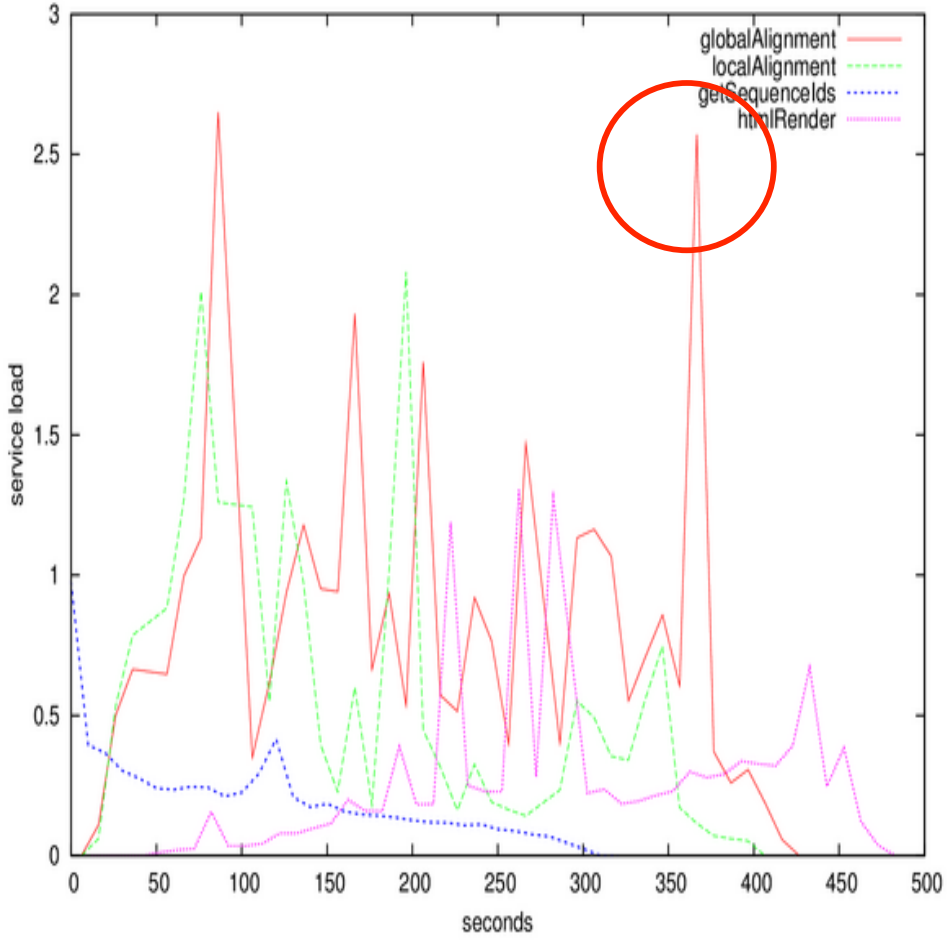
Scale up web services

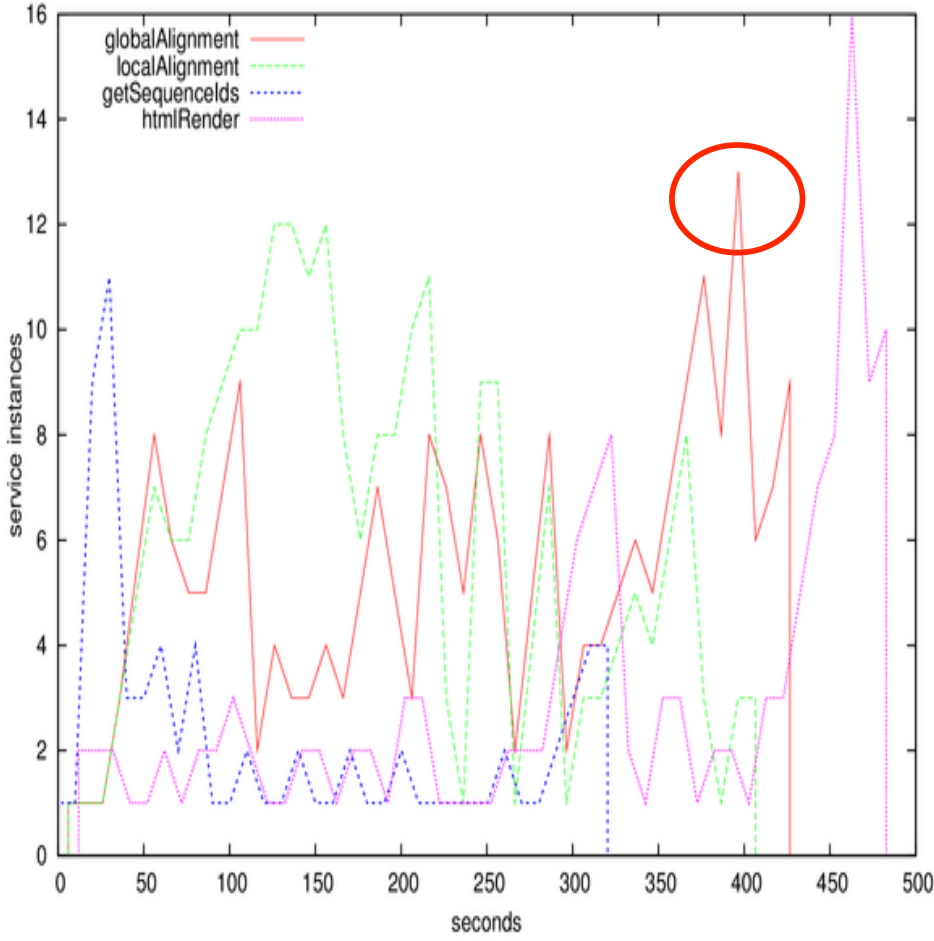




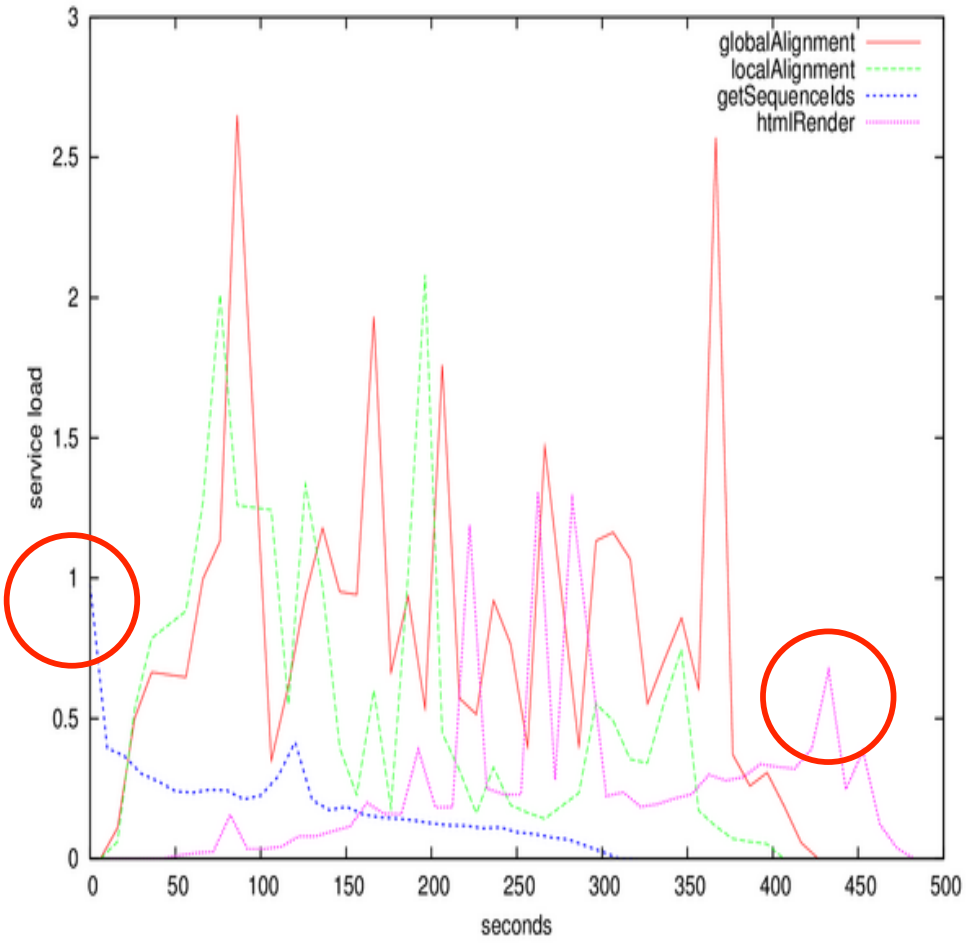
Peaks in load(left) will result in peaks in instances(right).
The fuzzy controllers **scale up** the web services to meet the demands

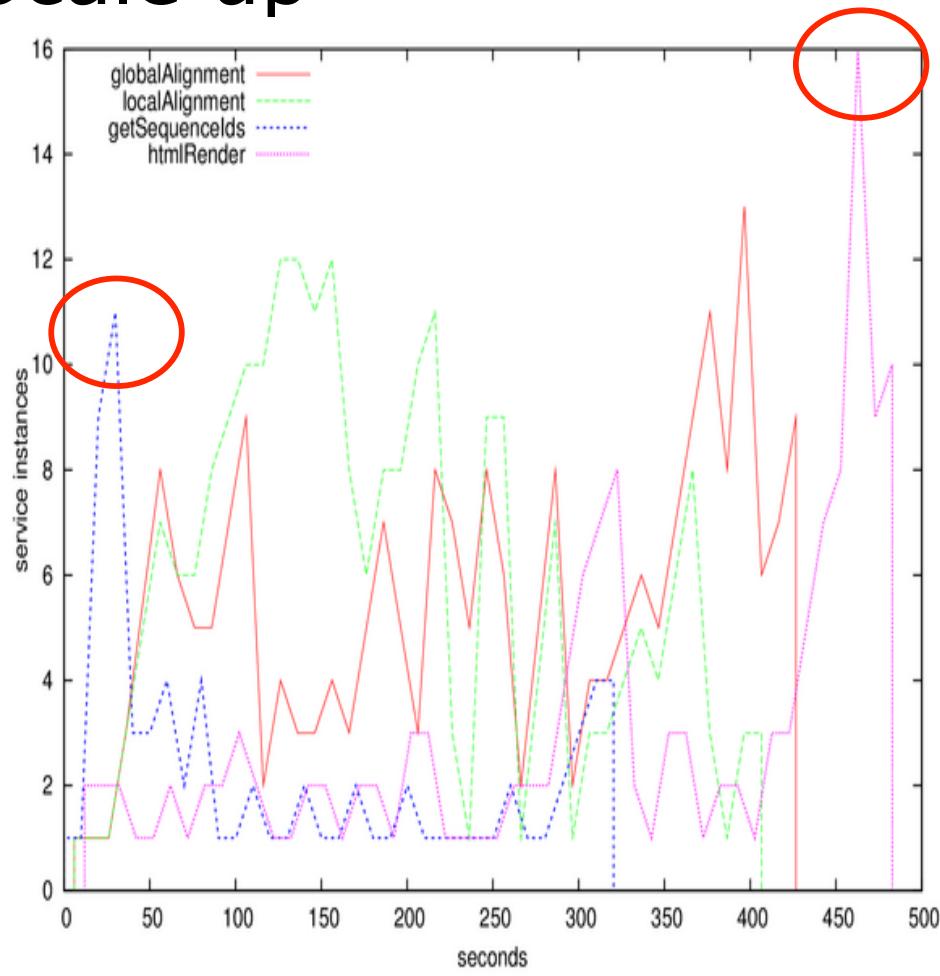
Scale up web services



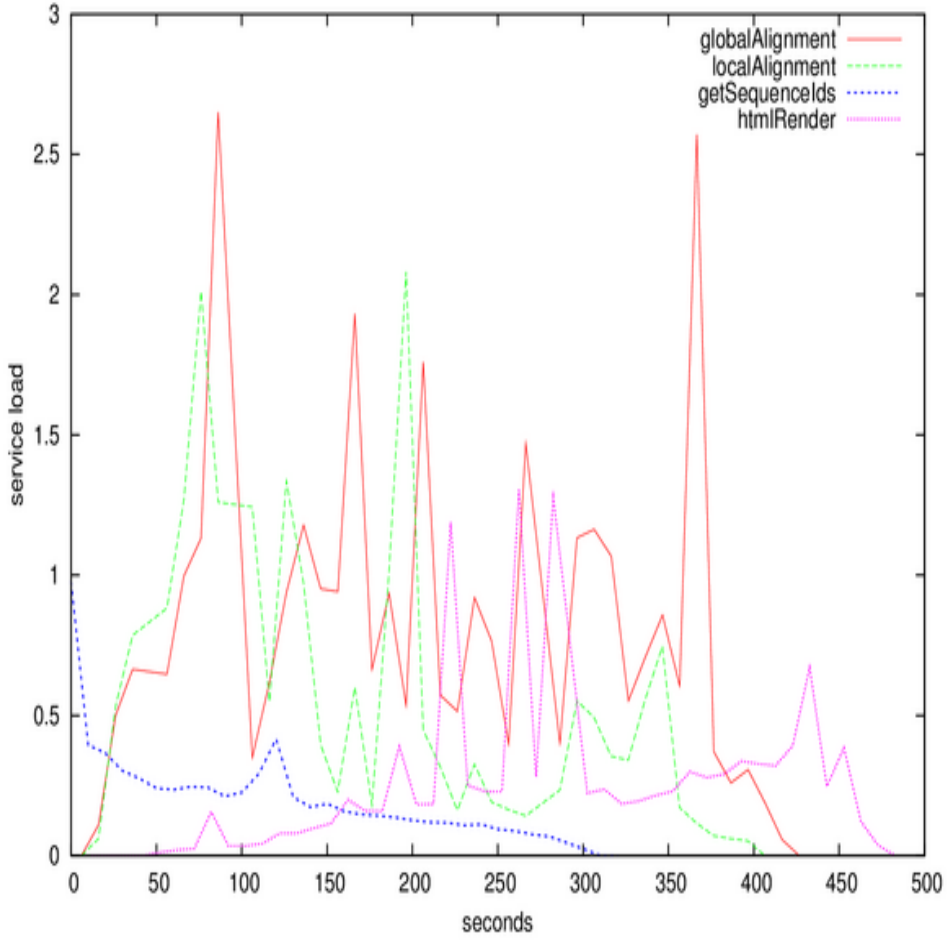


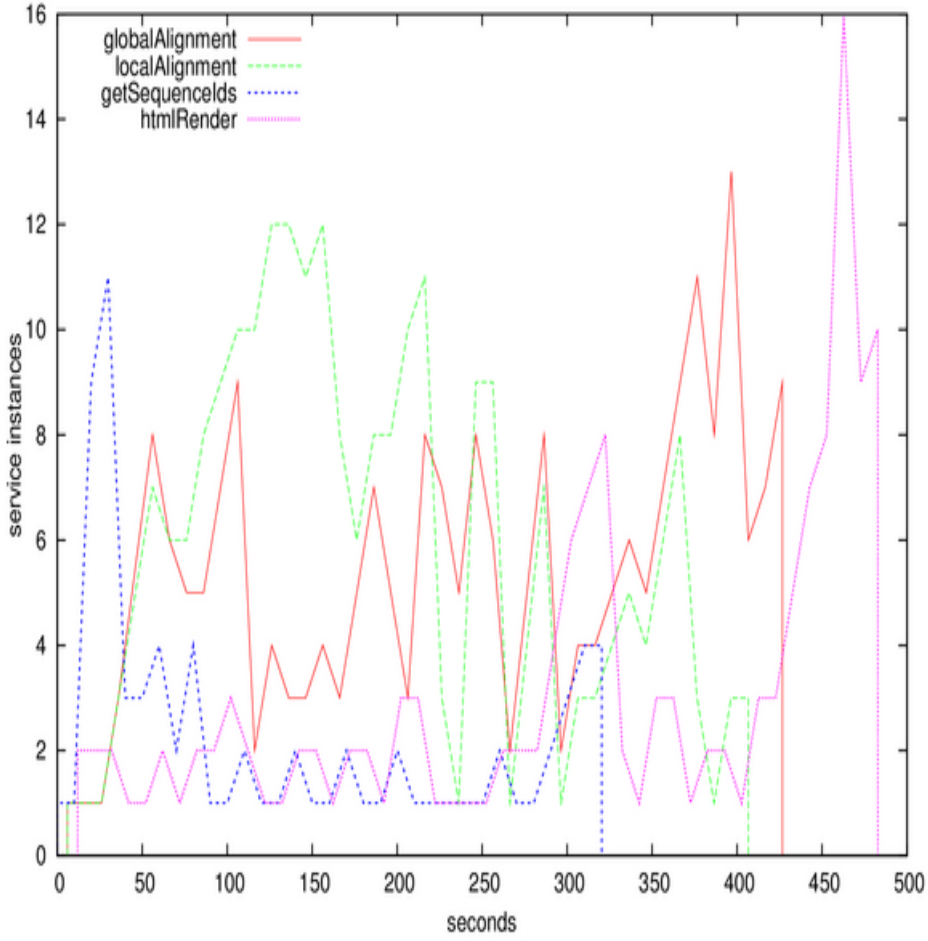
Greedy Scale up





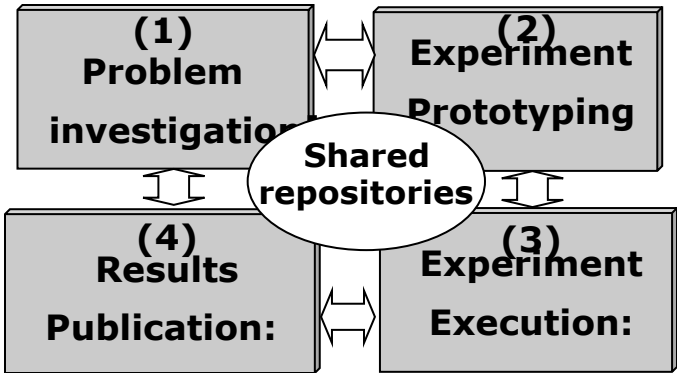
Scale down web services





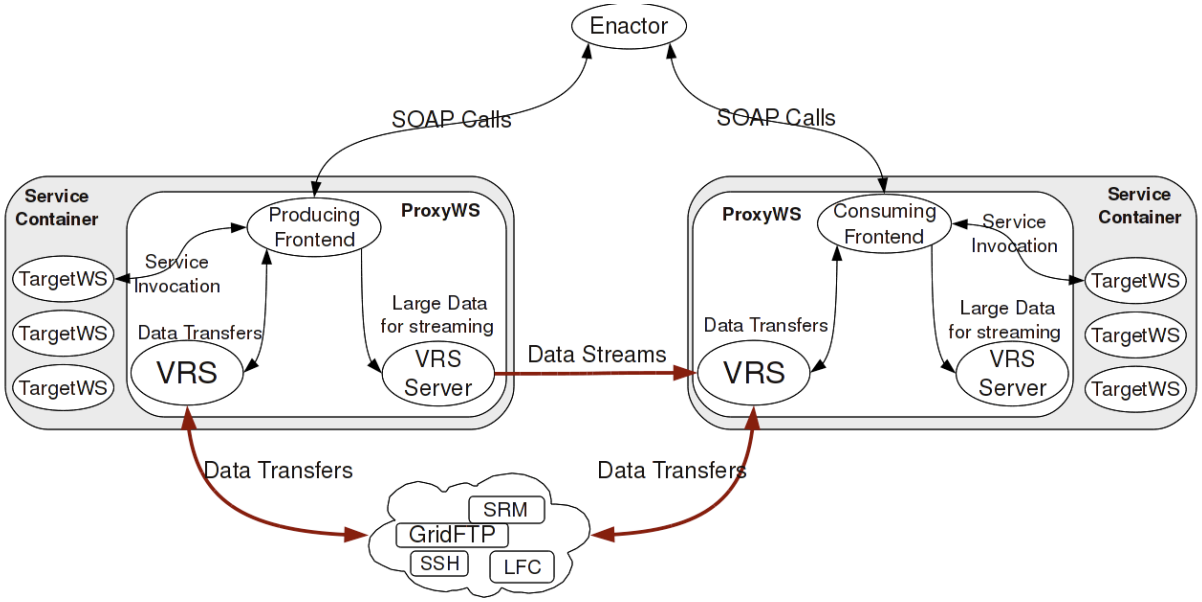
Outline

- Introduction
- Lifecycle of an e-science workflow
- Different approach to workflow scheduling
  - Workflow Process Modeling & Management In Grid/Cloud
  - Workflow and Web services (Workflow and Web services (intrusive/non intrusive)
- Provenance



Usage of Web Services in e-science

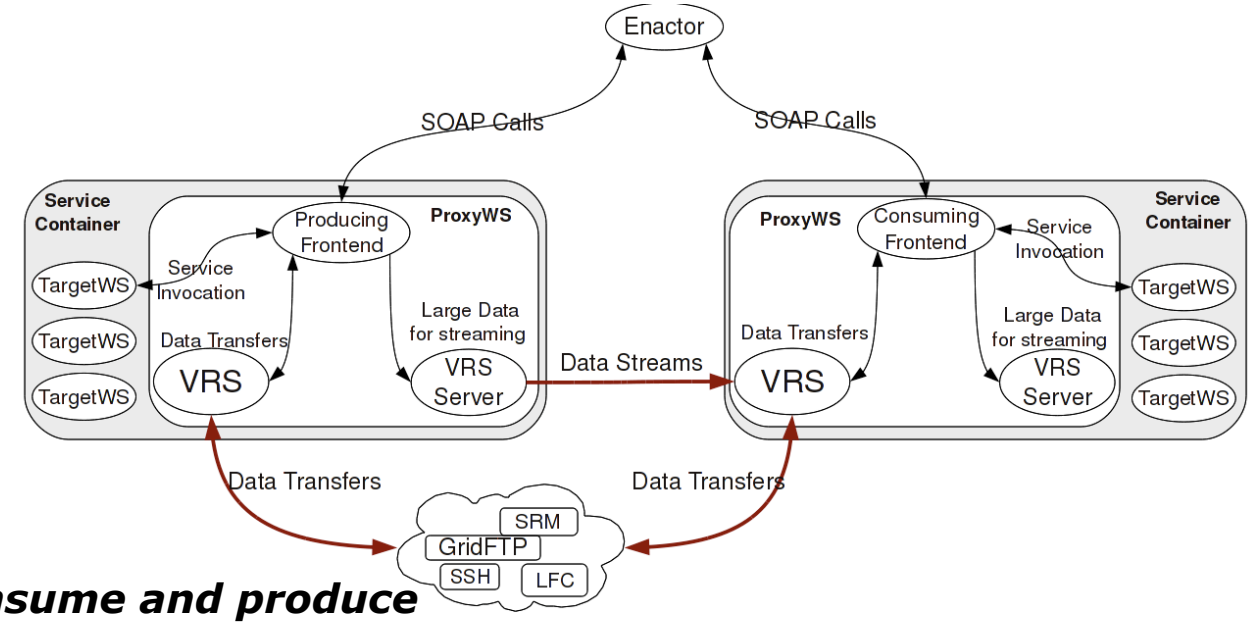
- In service orchestration, **all data is passed to the workflow engine** before delivered to a consuming WS
- Data transfers are made through **SOAP**, which **is unfit for large data transfers**



Enabling web services to consume and produce large distributed *datasets Spiros Koulouzis, Reginald Cushing, Konstantinos Karasavvas, Adam Belloum, Marian Bubak to be published JAN/FEB, IEEE Internet Computing, 2012*

ProxyWS

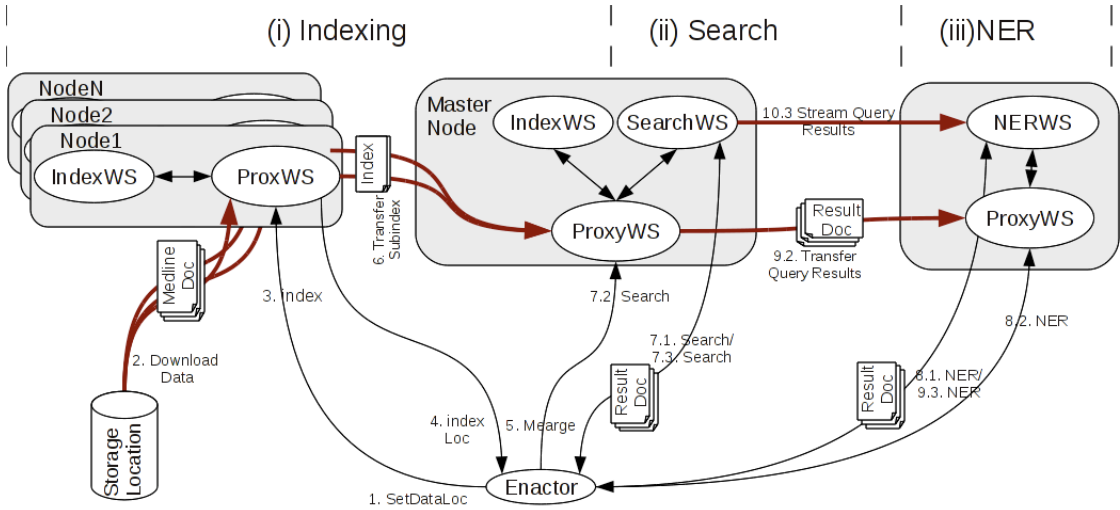
- uses multitude of protocols to transport large data
  - used as an interface for developing WSs able to stream data.
  - Or as enabler for legacy web services to stretch their current potential by referencing data that would otherwise be delivered via SOAP



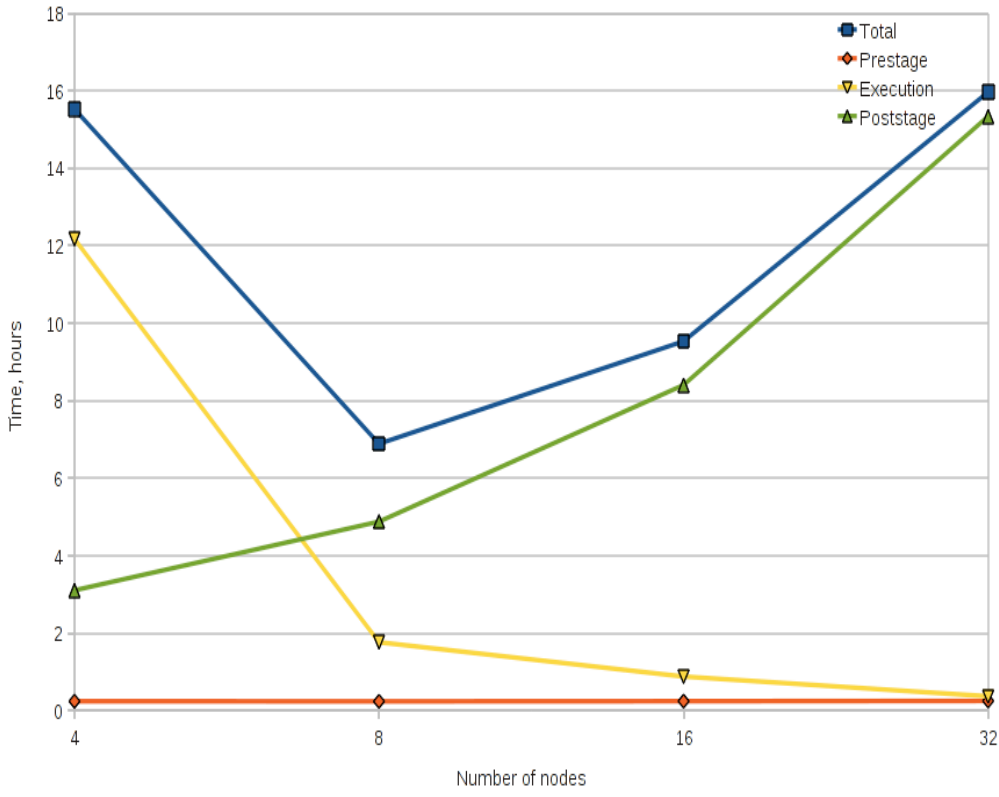
Enabling web services to consume and produce large distributed *datasets Spiros Koulouzis, Reginald Cushing, Konstantinos Karasavvas, Adam Belloum, Marian Bubak to be published JAN/FEB, IEEE Internet Computing, 2012*

Indexing Name Entry Recognition

- AIDA provides a set of components which enable the indexing of text documents in various formats.
- AIDA's Indexer component, called IndexerWS is a WS able to index document with the use of the Streaming library.

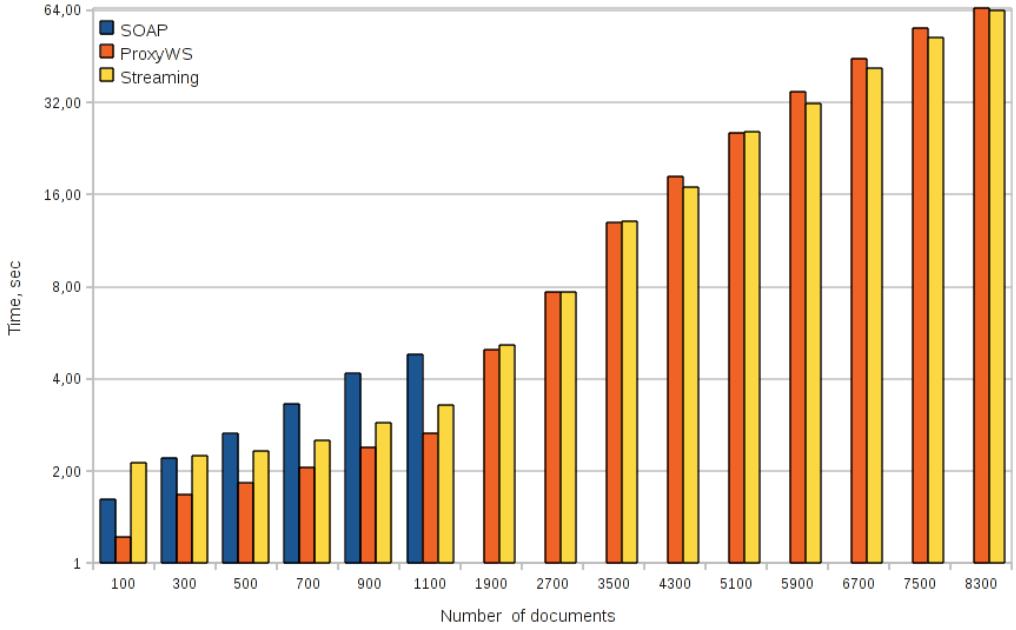


Results Indexing Web Services for Information Retrieval (Indexing)



Enabling web services to consume and produce large distributed *datasets Spiros Koulouzis, Reginald Cushing, Konstantinos Karasavvas, Adam Belloum, Marian Bubak to be published JAN/FEB, IEEE Internet Computing, 2012*

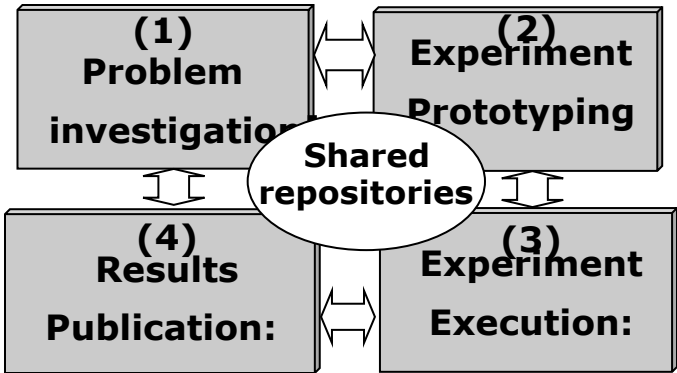
Results Indexing Web Services for Information Retrieval (NER)



Enabling web services to consume and produce large distributed *datasets Spiros Koulouzis, Reginald Cushing, Konstantinos Karasavvas, Adam Belloum, Marian Bubak to be published JAN/FEB, IEEE Internet Computing, 2012*

Outline

- Introduction
- Lifecycle of an e-science workflow
- Different approach to workflow scheduling
  - Workflow Process Modeling & Management In Grid/Cloud
  - Workflow and Web services (intrusive/non-intrusive)
- provenance



Provenance/ reproducibility

- "A complete provenance record for a data object allows the possibility to reproduce the result and reproducibility is a critical component of the scientific method"

- Provenance: The recording of metadata and provenance information during the various stages of the workflow lifecycle

Workflows and e-Science: An overview of workflow system features and capabilities Ewa Deelmana, Dennis Gannonb, Matthew Shields c, Ian Taylor, Future Generation Computer Systems 25 (2009) 528540

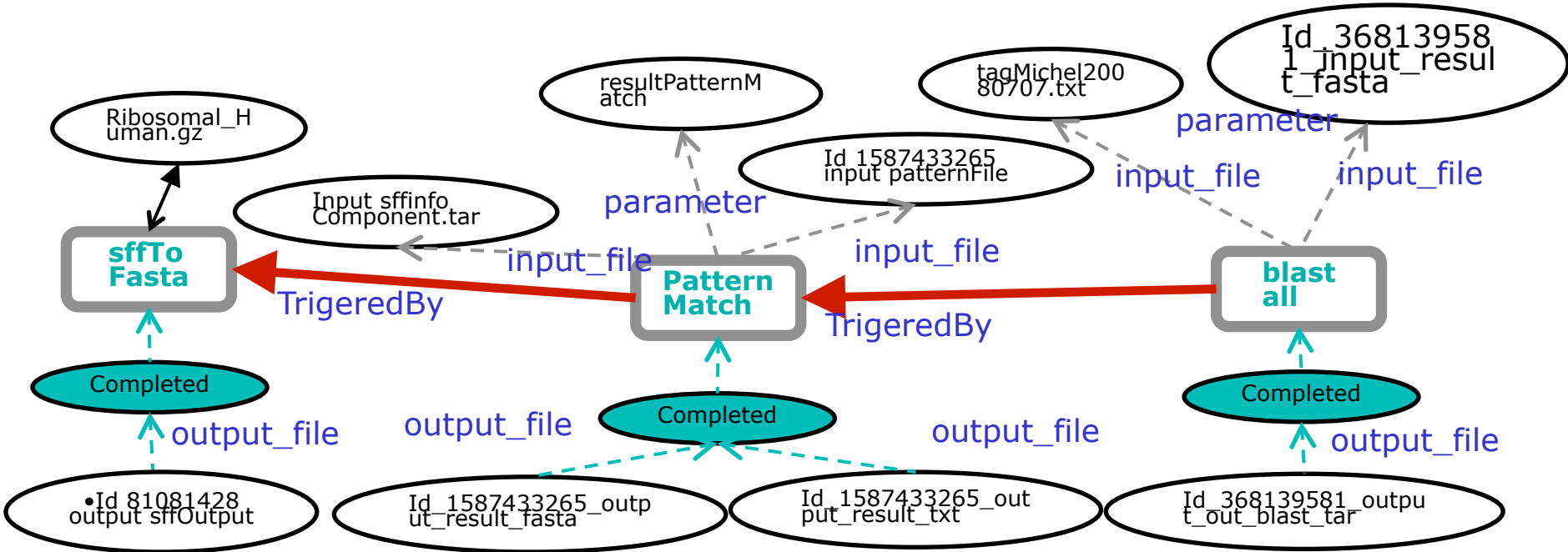
History-tracing XML (FH Aachen)

- provides data/process provenance following an approach that
  - maps the workflow graph to a layered structure of an XML document.
  - This allows an intuitive and easy processable representation of the workflow execution path,
  - which can be, eventually, electronically signed.

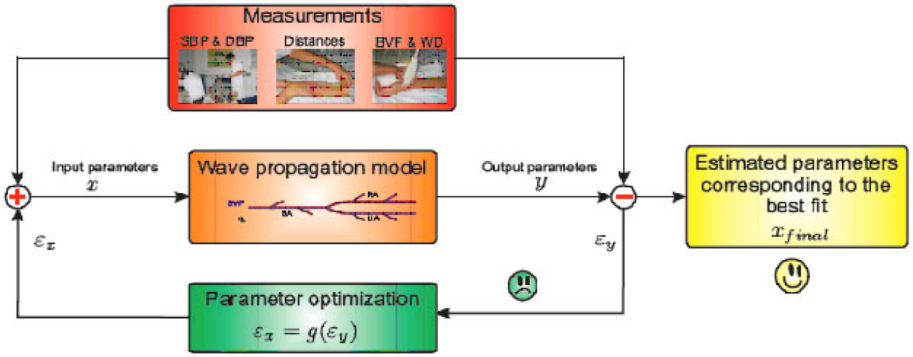
```
<patternMatch>
  <events>
    <PortResolved> provenance
data</PortResolved>
    <ConDone>provenance data
              </ConDone>
    ...
  </events>
  <fileReader2>
    <events> ... </events>
    <sign-fileReader2> ...
          </signfileReader2>
  </fileReader2>
  <sffToFasta>
    Reference
  </sffToFasta>
  <sign-patternMatch> ...
        </sign-patternMatch>
</patternMatch>
```

M. Gerards, Adam S. Z. Belloum, F. Berritz, V. Snder, S. Skorupa, **A History-tracing XML-base Proveannce Framework for workflows**, WORKS 2010, New Orleans, USA, November 2010

- PLIER is an implementation of the OPM 1.1 specifications.
- It's API provides a set of functions to build, store, and share workflow experiments as graphs.
- It also implements an optimal relational database as back-end storage that captures the concepts of the OPM model, using the Java Persistence API (JPA 2.0) and Hibernate.
- In addition, the PLIER API provides specific interfaces, using JDO 3.1, to transform, or serialize, the provenance data into specific formats (e.g. RDF, XML, and DOT).



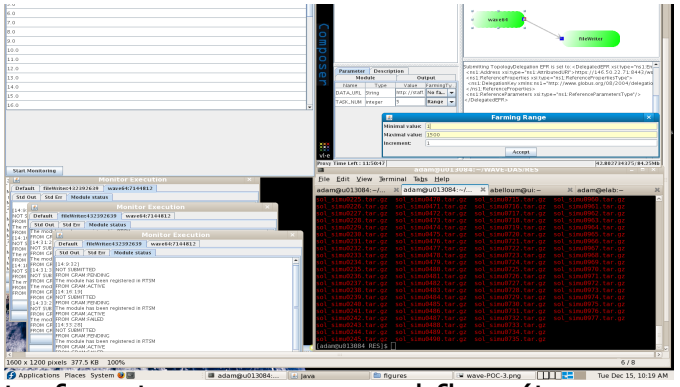
wave propagation model applications



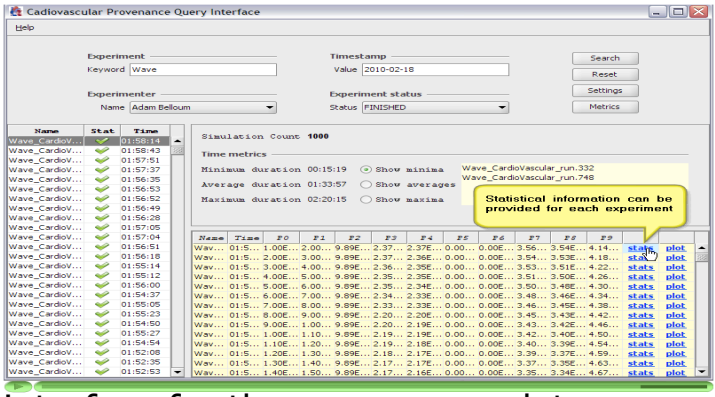
[Biomedical engineering Cardiovascular biomechanics group TUE])

wave propagation model of blood flow in large vessels using an approximate velocity profile function:

a biomedical study for which **3000 runs** were required to perform a global sensitivity analysis of a blood pressure wave propagation in arteries



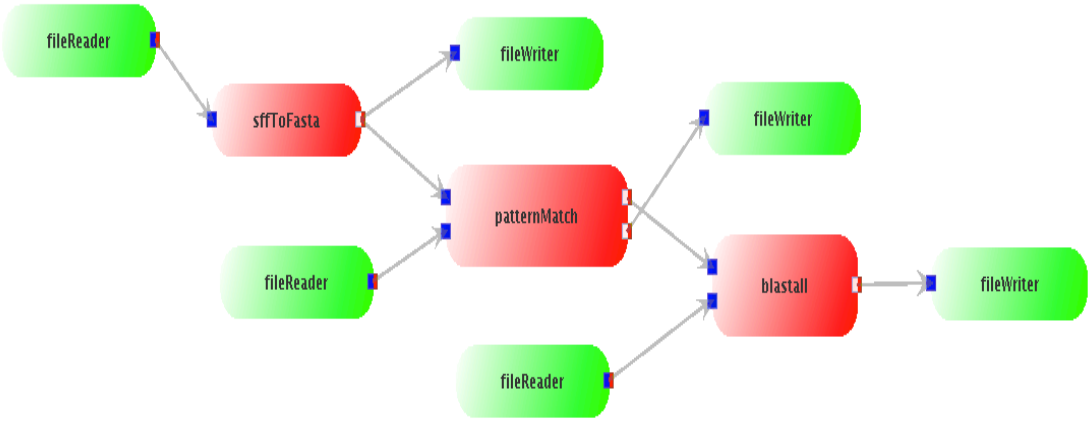
User Interface to compose workflow (top right), monitor the execution of the farmed workflows (top left), and monitor each run separately (bottom left) data



Query interface for the provenance data collected from 3000 simulations of the "*wave propagation model of blood flow in large vessels using an approximate velocity profile function*"

BigGrid project 2009, presented EGI/ BigGrid technical forum 2010

Blast Application



[Department of Clinical Epidemiology, Biostatistics and Bioinformatics (KEBB), AMC]

The aim of the application is the **alignment of DNA sequence** data with a given reference database. A workflow approach is currently followed to run this application on distributed computing resources.

For Each workflow run

- The provenance data is collected an stored following the XML-tracing system
- User interface allows to reproduce events that occurred at runtime (replay mode)
- User Interface can be customized (User can select the events to track)
- User Interface show resource usage

on-going work UvA-AMC-fh-aachen

More References

1. A.S.Z. Belloum, V. Korkhov, S Koulouzis, M. A Inda, and M. Bubak *Collaborative e-Science experiments: from scientific workflow to knowledge sharing* JULY/AUGUST, IEEE Internet Computing, 2011
2. Ilkay Altintas, Manish Kumar Anand, Daniel Crawl, Shawn Bowers, Adam Belloum, Paolo Missier, Bertram Ludascher, Carole A. Goble, Peter M.A. Sloot, Understanding Collaborative Studies Through Interoperable Workflow Provenance, IPAW2010, Troy, NY, USA
3. A. Belloum, Z. Zhao, and M. Bubak Workflow systems and applications , Future Generation Comp. Syst. 25 (5): 525-527 (2009)
4. Z. Zhao, A.S.Z. Belloum, et al., Distributed execution of aggregated multi domain workflows using an agent framework The 1st IEEE International Workshop on Scientific Workflows, Salt Lake City, U.SA, 2007
5. Zhiming Zhao, Adam Belloum, Cees De Laat, Pieter Adriaans, Bob Hertzberger Using Jade agent framework to prototype an e-Science workflow bus Authors Cluster Computing and the Grid, 2007. CCGRID 2007

COMMIT/

http://www.commit-nl.nl/

http://www.science.n/~gvlam/wsvlam/

vl-e

http://www.vle.nl/