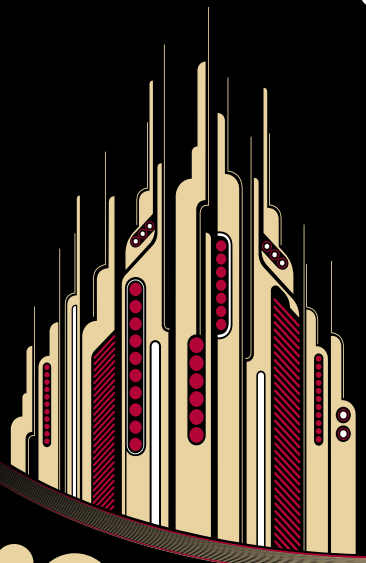
Experience with Adapting a WS-BPEL Runtime for eScience Workflows

Thilina Gunarathne,
Chathura Herath, Eran Chinthaka, Suresh Marru
Pervasive Technology Institute
Indiana University



Look to the future of high-performance computing.

PERVASIVE TECHNOLOGY INSTITUTE
INDIANA UNIVERSITY

SC Portland, OR 2009

Introduction

- Scientist communities are solving deeper larger problems spanning across domains
- Share & combine multiple applications in flexible yet planned order
  - Orchestrate together using workflows
- Most of the scientific workflow systems use custom formats
  - Interoperability challenge

WS-BPEL

- Business Process Execution Language for Web Services (WS-BPEL)
  - De-facto standard for specifying web service based business processes and service compositions
- Basic activities
  - Invoke, Receive, Assign..
- Structured activities
  - Sequence, Flow, ForEach,..

LEAD: an NSF funded large Information Technology Research Project

L I N K E D
ENVIRONMENTS
FOR ATMOSPHERIC
D I S C O V E R Y

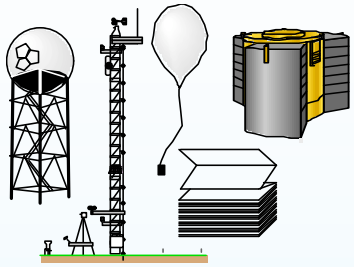
PERVASIVE TECHNOLOGY
INSTITUTE
INDIANA UNIVERSITY

Linked Environments for Atmospheric Discovery

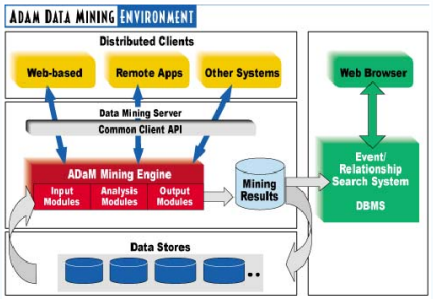
LEAD –

- researched to better understand the atmosphere, educate more effectively about it, and forecast more accurately by adapting technologies as the weather occurs.
- improved mesoscale forecasts
- ingested more local observations to enhance the initial conditions.
- Used in NOAA Storm Prediction Center Hazardous Weather Testbed Spring Experiments.
- Used in National Collegiate Forecast Competition
- Used by USDA for crop planning
- Used as Educational Tools

LEAD Dynamic Adaptive Infrastructure



Streaming Observations



Data Mining

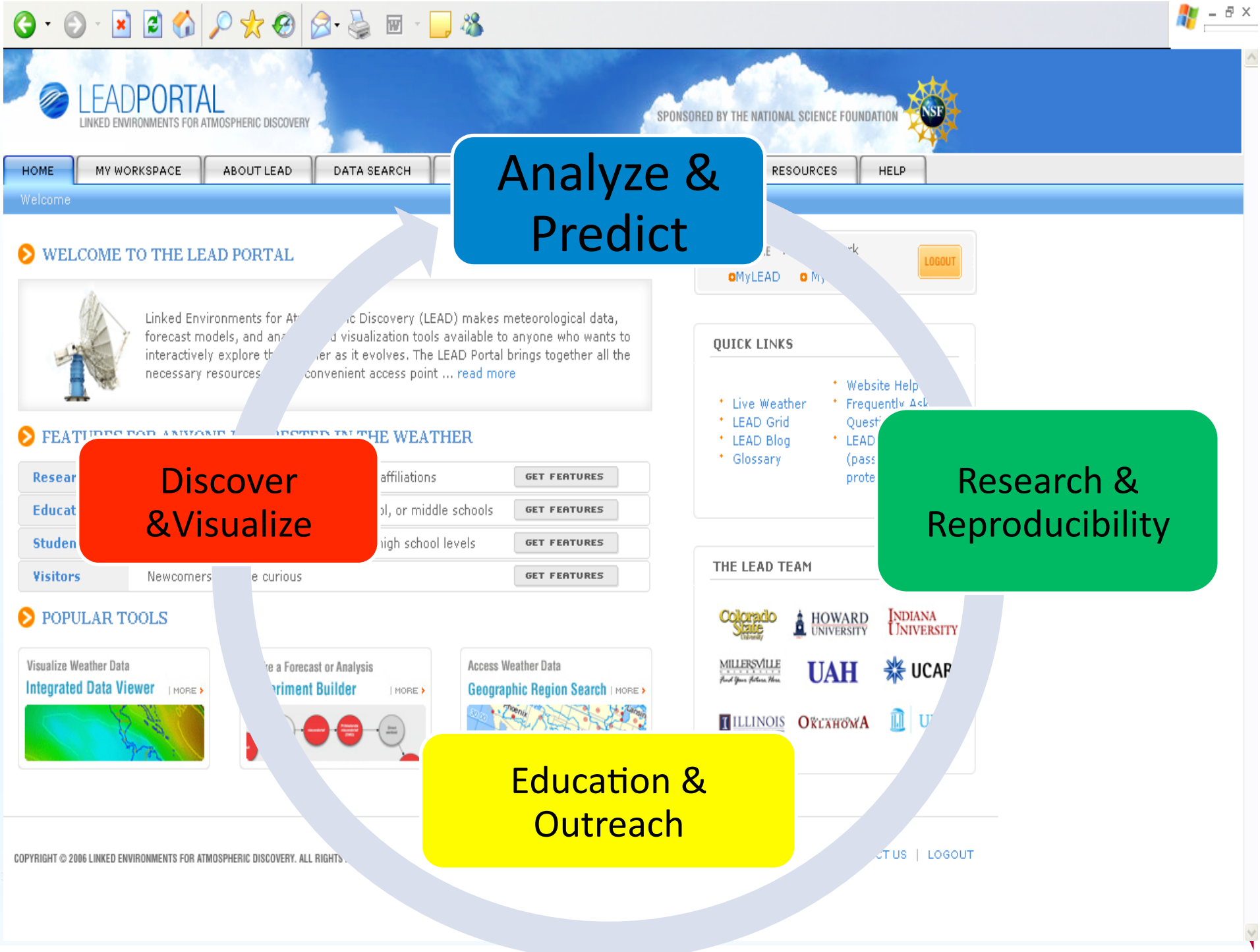
Storms Forming

Forecast Model

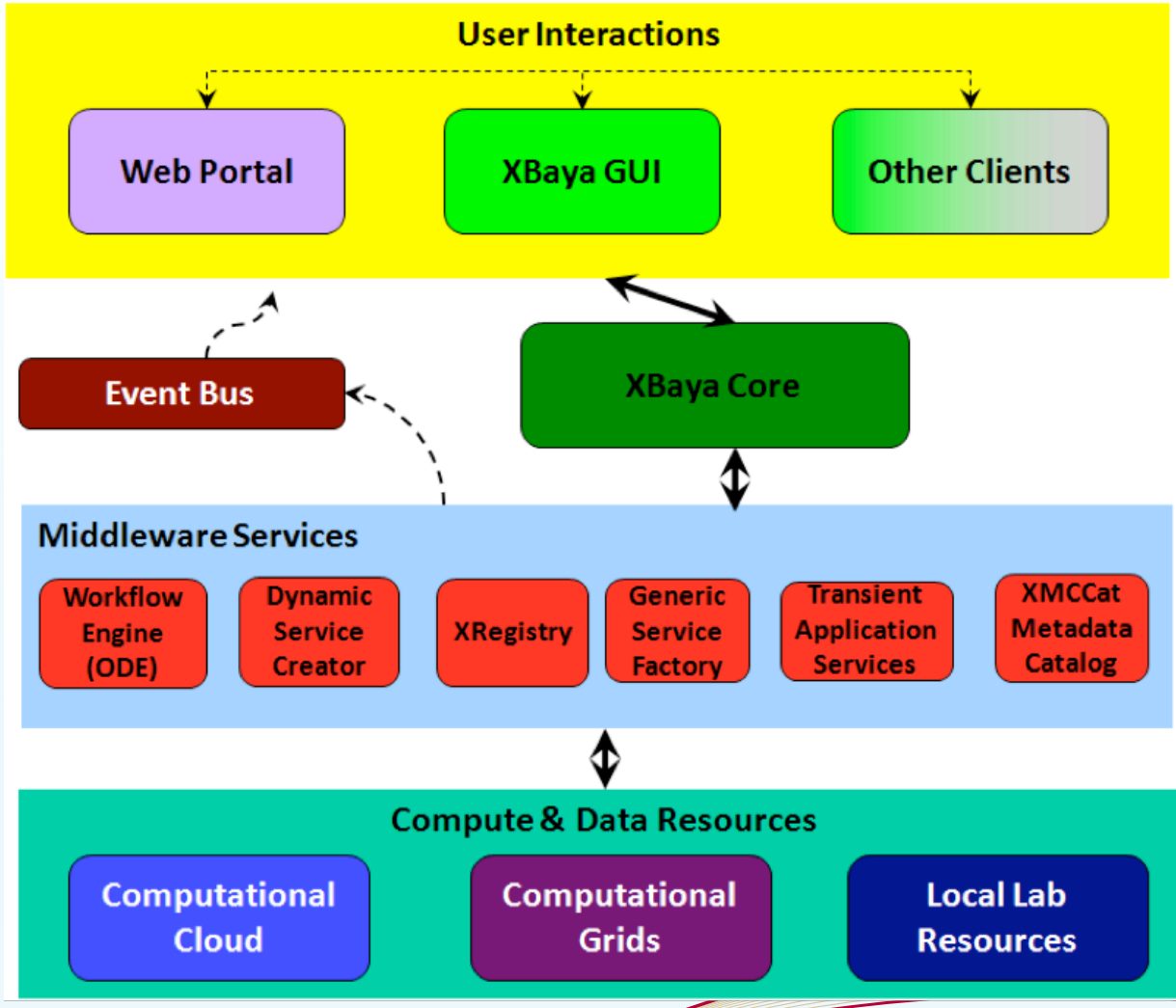
Instrument Steering

Refine forecast gri

PERVASIVE TECHNOLOGY INSTITUTE
INDIANA UNIVERSITY


LEADPORTAL
LINKED ENVIRONMENTS FOR ATMOSPHERIC DISCOVERY
SPONSORED BY THE NATIONAL SCIENCE FOUNDATION
HOME
MY WORKSPACE
ABOUT LEAD
DATA SEARCH
RESOURCES
HELP
Welcome
WELCOME TO THE LEAD PORTAL
Linked Environments for At ic Discovery (LEAD) makes meteorological data, forecast models, and an d visualization tools available to anyone who wants to interactively explore th er as it evolves. The LEAD Portal brings together all the necessary resources convenient access point ... read more
LOGOUT
MyLEAD
QUICK LINKS
Live Weather
LEAD Grid
LEAD Blog
Glossary
Website Help
Frequently Ask
Quest
LEAD
(pass
prote
THE WEATHER
Resear affiliations GET FEATURES
Educat l, or middle schools GET FEATURES
Studen igh school levels GET FEATURES
Visitors Newcomers e curious GET FEATURES
THE LEAD TEAM
POPULAR TOOLS
Visualize Weather Data
Integrated Data Viewer | MORE
e a Forecast or Analysis
riment Builder | MORE
Access Weather Data
Geographic Region Search | MORE
Analyze & Predict
Discover &Visualize
Research & Reproducibility
Education & Outreach
COPYRIGHT © 2006 LINKED ENVIRONMENTS FOR ATMOSPHERIC DISCOVERY. ALL RIGHTS
CT US | LOGOUT

LEAD Architecture



GPEL

- Grid Process Execution Language
  - BPEL4WS based home grown research workflow engine
  - Supports a subset of BPEL4WS 1.1
  - One of the very early adaptations of BPEL efforts
- Specifically designed for eScience Usage
  - Long running workflow support
  - Decoupled client

Goals

WS-BPEL 2.0 features

Sustainability

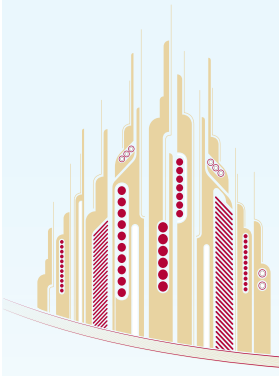
- Well supported run time with minimal custom changes

Improved Scalability & performance

Minimize changes to legacy components

Portability & avoid lock in

- Adhering to widely used open standards
- Avoid using runtime specific features



Challenges

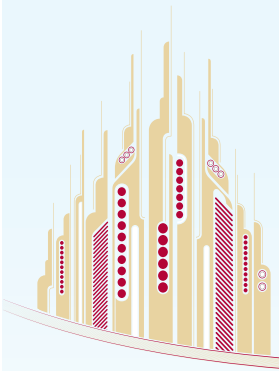
- Propagation of Workflow Context Information
- Asynchronous Invocation
- Notifications & Monitoring
- Workflow Instance Creation
- Variable Initializing
- Deployment
- Workflow Client

Propagation of Workflow Context Information

- Lead Context Header (LCH)
  - Unique identifiers
  - End point references
  - Configurations information
  - Security information
- Lead mandates propagation of LCH with application specific SOAP messages
  - Workflow runtime need to propagate the LCH received in input message to every service invocation message

Propagation of Workflow Context Information

- Implemented using auto-generated WS-BPEL logic
- Define LCH in the WSDL of the workflow, by binding a message part to a SOAP header block
  - Allows us to access LCH as a variable inside WS-BPEL process



```
<definitions ...>
  <message name="requestMessage">
    <part name="params" element="tns:payload"/>
    <part name="leadHeader" element="lc:context"/>
  </message>
  ....
  <binding ...
    <operation name="Run">
      <input message="tns:requestMessage">
        <soap:body parts="params" use="literal"/>
        <soap:header message="tns:requestMessage"
                part="leadHeader" use="literal"/>
      </input>
    </operation>
  </binding> ...
</definitions>
```

Asynchronous Invocation

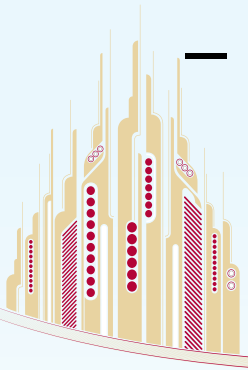
- Prohibitive to invoke LEAD services in a synchronous blocking manner
  - Long running tasks => long running web service invocations
  - Multi hops
- No standard compliant unambiguous mechanism for asynchronous req/resp web service operation invocations in WS-BPEL
  - No integrated support for WS-Addressing

Asynchronous Invocation

- Two popular mechanisms
  - Implement as dual one way messages
    - Requires reply address information to be propagated using a proprietary mechanisms
    - Requires services to be modified
  - Make invoke inherently asynchronous
    - Non-portability of the WS-BPEL process behavior
- Proposal for WS-Addressing based WS-BPEL extension

Notification & Monitoring

- Two types of monitoring
  - Workflow engine state monitoring
  - Service invocation state monitoring
- Generating Notifications from BPEL Engine
  - Out of scope of WS-BPEL specification
  - Almost all the popular WS-BPEL runtimes provide plug-in mechanisms for notification generation
  - LEAD workflow tracking library based Apache ODE notification handler



Notification & Monitoring – Assigning Service Identifiers

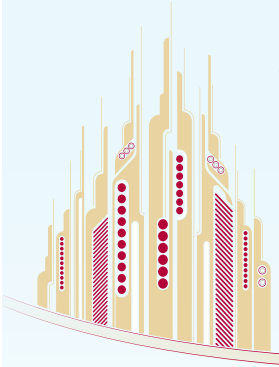
- Fine grained monitoring
- Necessitates unique identifiers for each service invocation
- XBaya generated identifiers
  - Node-id
  - Workflow time step
- Rely on WS-BPEL logic to copy the identifiers to LCH of service invocation messages

Workflow Instance Creation

- In GPEL, separate workflow instance creation and execution steps
  - Workflow engine creates the identifiers
- In WS-BPEL, workflow instances are created implicitly when messages are received <receive> activities marked with “createinstance=true”
  - Workflow client creates the Workflow-Instance-ID
  - Different from Apache ODE internal process instance id
    - Correlation between two ids in the first notification

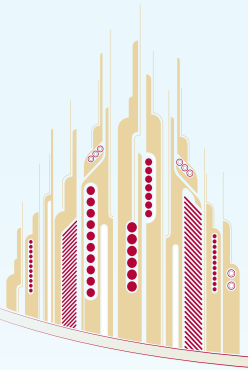
Variable Initializing

- WS-BPEL requires complex variables to be initialized before using or copying in to them
  - Xbaya workflow composer automatically adds WS-BPEL logic for initialization steps using its domain knowledge
- Some engines initialize variables automatically
  - But cannot be done accurately for all the cases without the domain knowledge

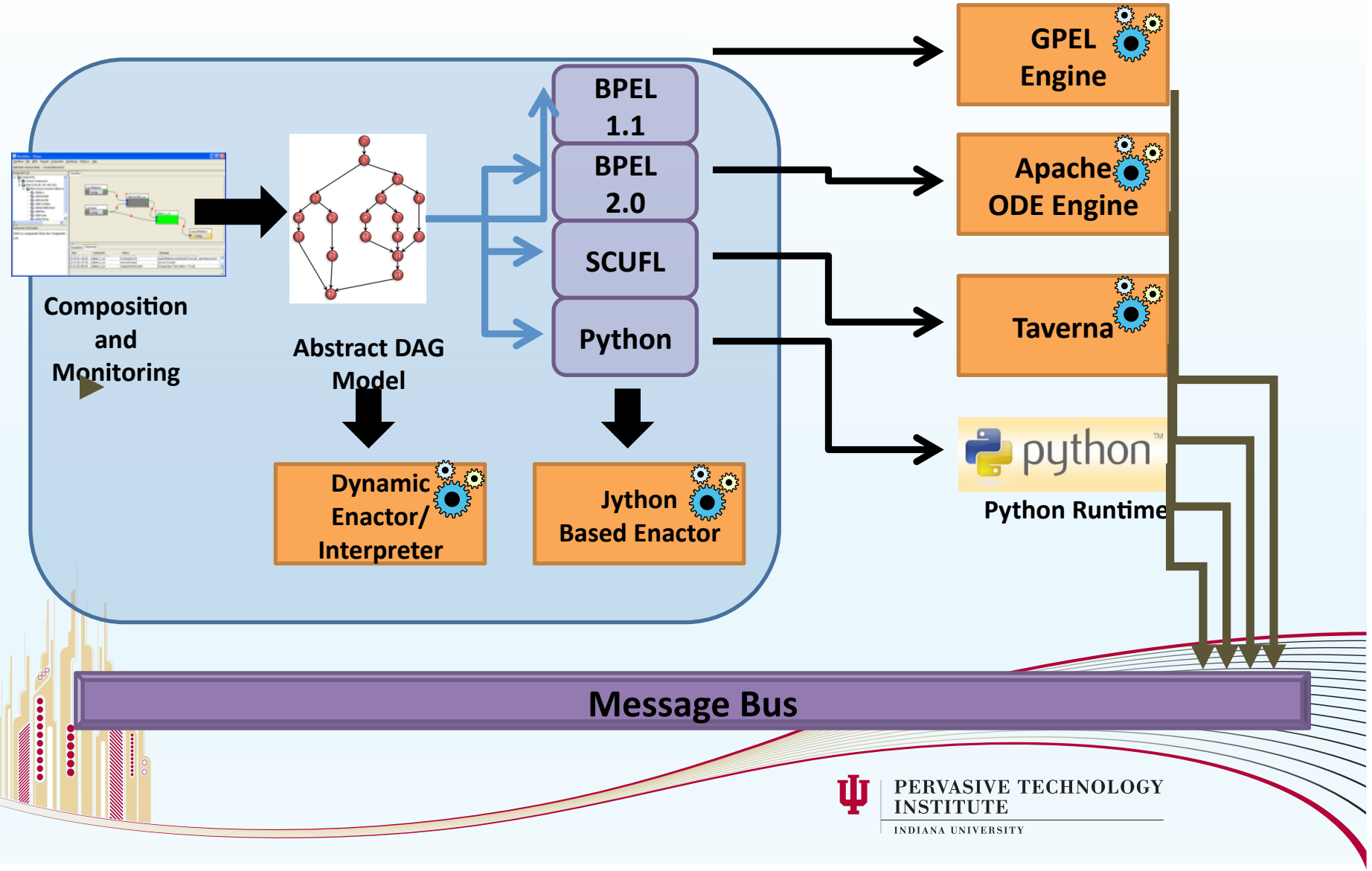


Workflow Deployment

- No standard way of packaging and deploying WS-BPEL processes
- Xbaya workflow composer is designed to support many workflow engines
  - Engine specific decoupled workflow proxy services
  - Generic workflow description from XBaya to the proxy service
    - Currently XBaya contains few ODE specific changes, which will be removed soon.

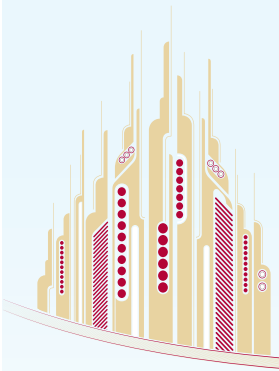


Workflow Client

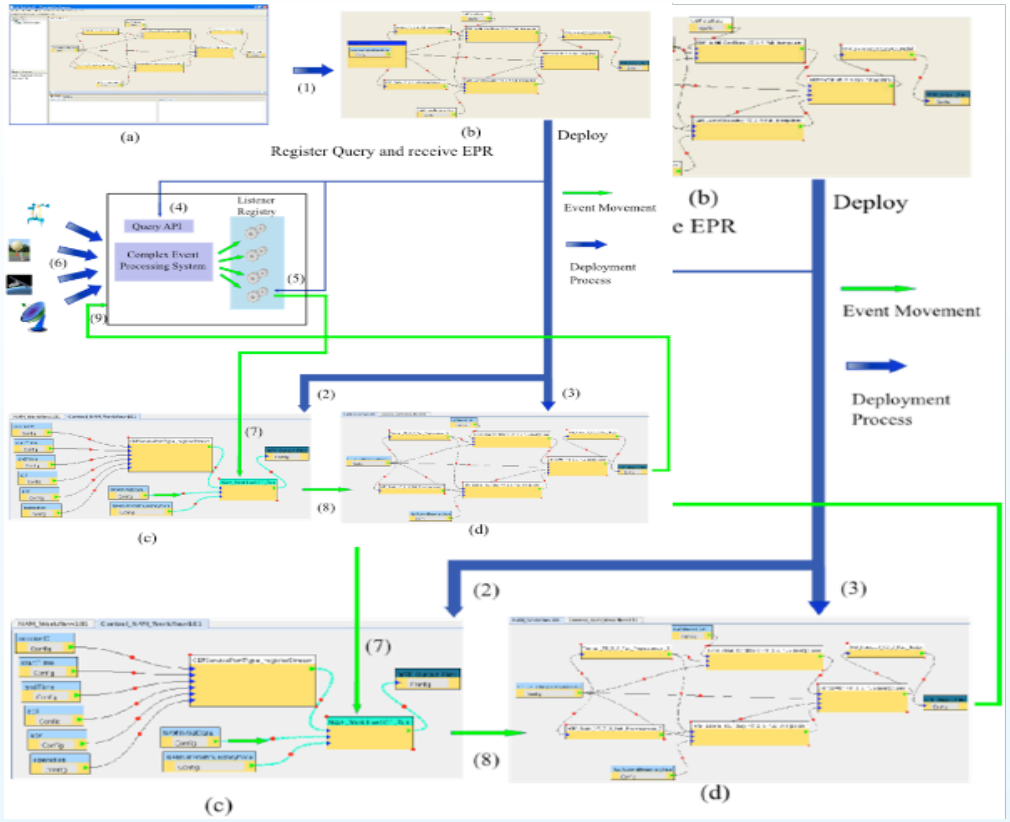


New Use Cases

- Stream Mining
- Parametric Studies
- Workflow Instance Recovery

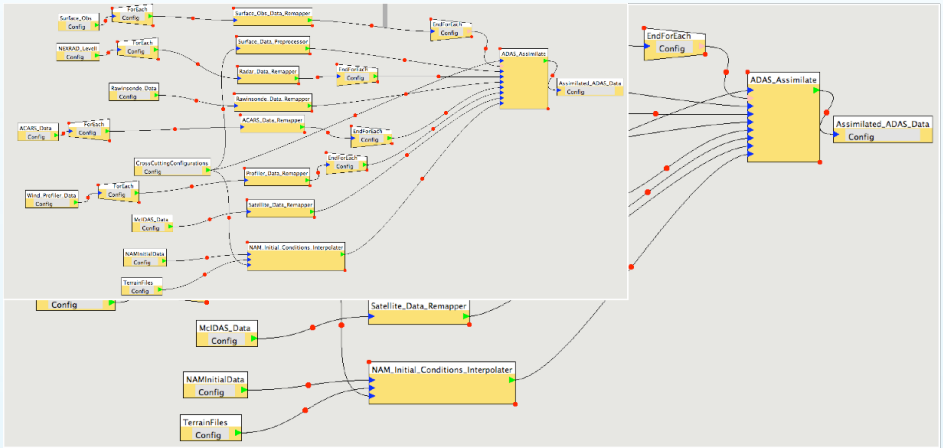


Stream Mining



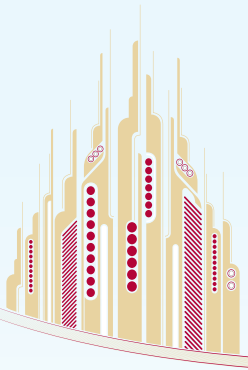
<for-each> for parametric studies

- Exploring a parameter space to identify or optimize a solution



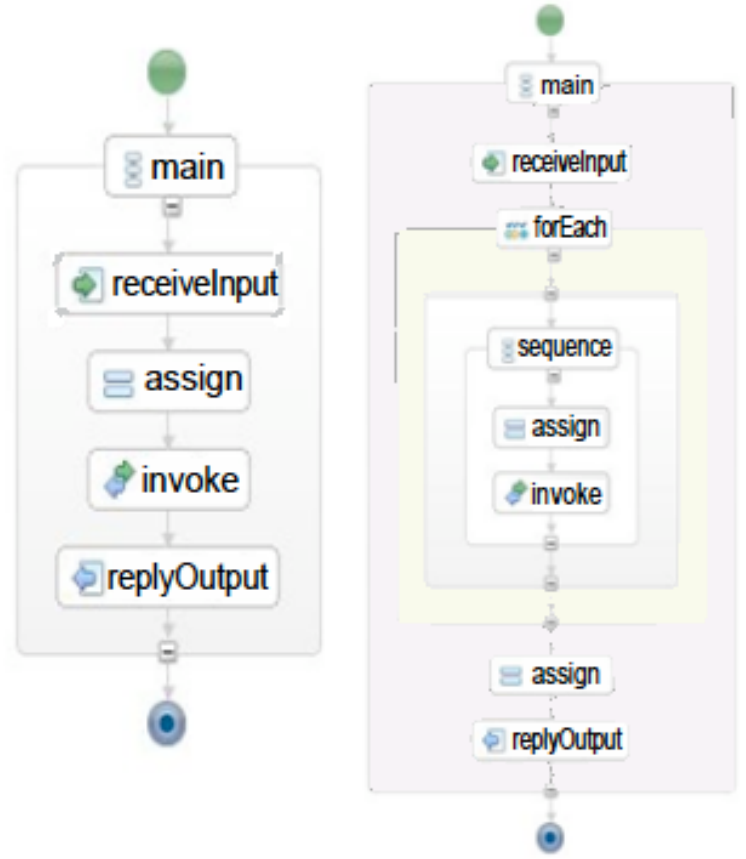
Workflow instance recovery

- Multi-level fault tolerance support in LEAD
  - No need to use WS-BPEL exception handling
    - WS-BPEL error recovery Vs Scientific workflows
- Only infrastructure failures are handled at the working engine level
- Hasthi monitoring infrastructure
  - Performs corrective actions in the event of an infrastructure failure
    1. Recovery of infrastructure components
    2. Resurrects workflow instances by replaying input messages

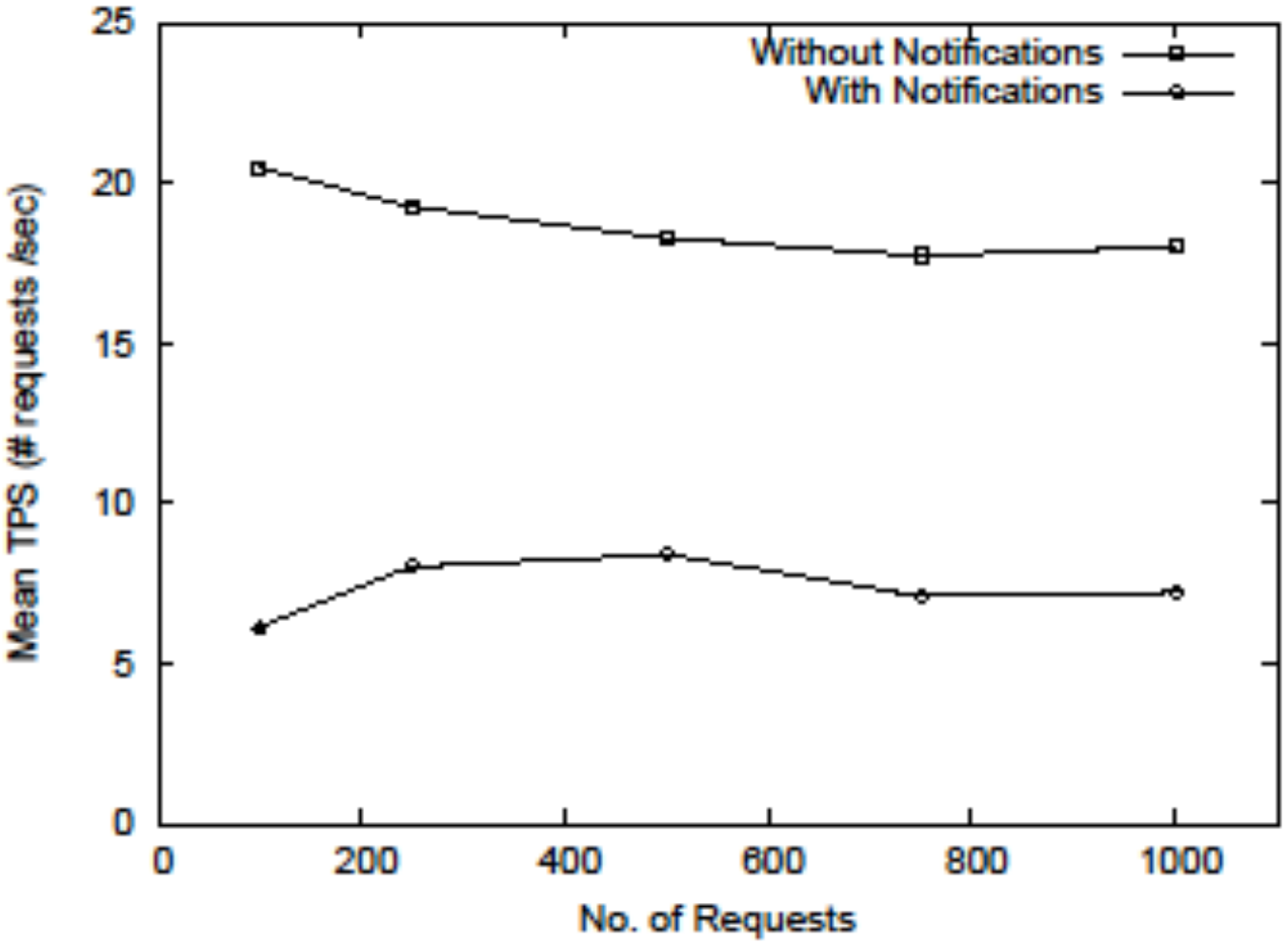


Performance & Reliability

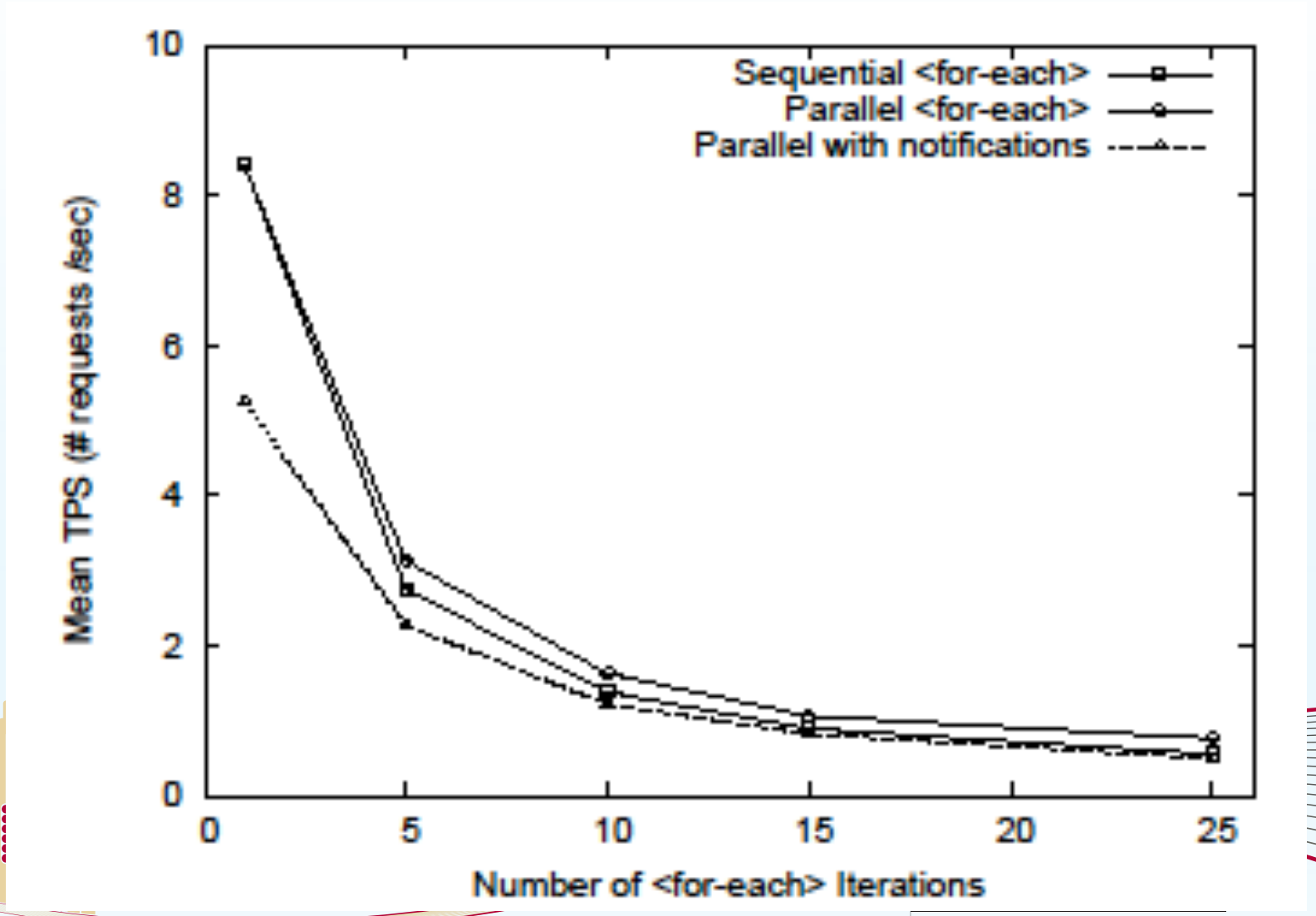
- Evaluate the workflow system with regards to LEAD performance and scalability requirements
- Scalable back end service
  - TPS >4000
- Same configuration as LEAD production ODE



Simple Service Invocation Workflow

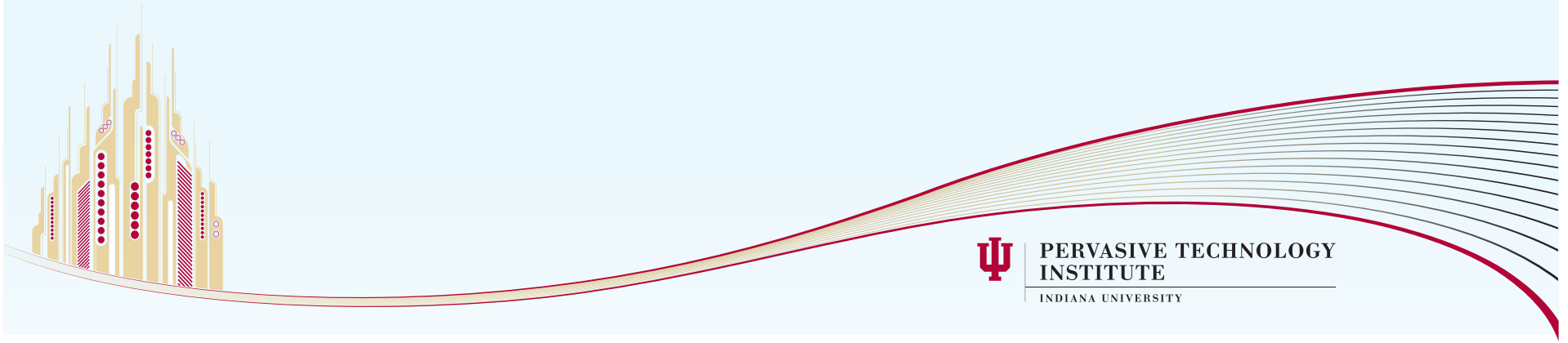


<for-each> workflow



Performance Comments

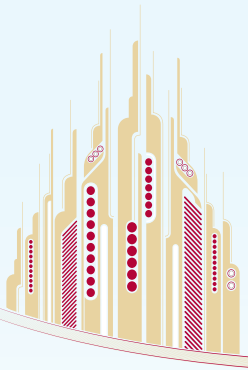
- Parallel speedup results
  - More than factor of 4 speedup when 5 way parallelism is used with a long running service
- Notification overhead is smaller in complex workflows



Experience

- Currently deployed in the LEAD production & development stacks
- More than 1000 workflow deployments
- Available open source in OGCE – Xbaya & the scientific workflow extensions
  - Minimal changes to ODE as most of the requirements were implemented using auto generated WS-BPEL logic.

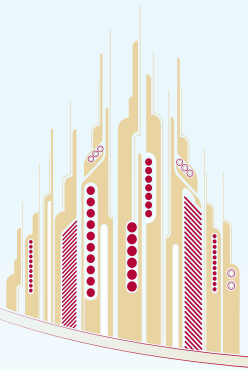
Questions



Acknowledgement

- Aleksander Slominski and Satoshi Shirasuna for the foundation they laid.
- Prof. Dennis Gannon & Prof. Beth Plale for the mentoring
- Srinath Perera and Lavanya Ramakrishnan for their direct and indirect constibutions.
- NSF funding for LEAD project and the NSF funding for OGCE

Thanks



PERVASIVE TECHNOLOGY INSTITUTE
INDIANA UNIVERSITY

Service Monitoring via Events

- The service output is a stream of events
  - I am running your request
  - I have started to move your input files.
  - I have all the files
  - I am running your application.
  - The application is finished
  - I am moving the output to you file space
  - I am done.
- These are automatically generated by the service using a distributed event system (WS-Eventing / WS-Notification)
  - Topic based pub-sub system with a well known “channel”.