# The Planets Interoperability Framework

## Scalable Services for Digital Preservation

**DPE, Planets and CASPAR Third Annual Conference:**
**Costs, Benefits and Motivations for Digital Preservation**

30. October 2008

**Ross King**, Christian Sadilek, Rainer Schmidt
Austrian Research Centers GmbH – ARC

# Outline

- Planets Interoperability Framework
- Grids and Clouds
- Initial Experimental Results

# The Planets Interoperability Framework

## Motivation

- There are a number of functions that all (or nearly all) software applications commonly need. These include functions such as
  - Data persistence
  - User management
  - Authentication and Authorization
  - Monitoring, Logging, and Notification
- The Interoperability Framework (IF) software components provide these commonly required functions.

# The Planets Interoperability Framework

- Defines an Service-Oriented Architecture for Digital Preservation
  - Set of Services, Interfaces, a common Data Model
- Implements Common Services
  - Authentication and Authorization, Monitoring, Logging, Notification, …
  - Service Registration and Lookup
- Provides APIs for Applications that *use* Planets
  - Testbed Experiments, Executing Preservation Plans
- Provides Workflow Enactment Service and Engine
  - Components-based, XML serialization

# The Problem of Scalability

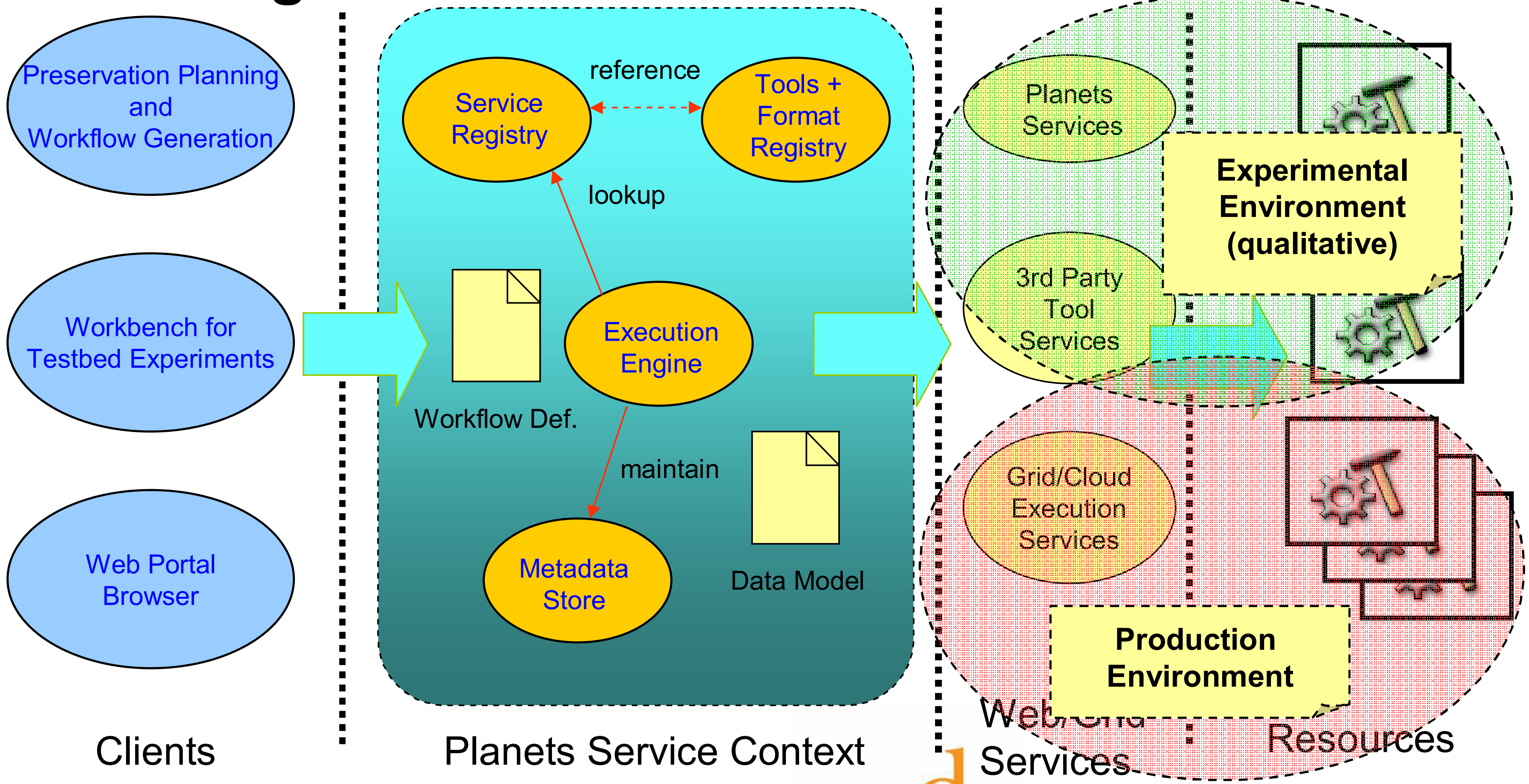
- Planets is a preservation architecture based on Web Services
  - Supports interoperability and a distributed environment
  - Sufficient for a controlled experiments (Testbed)
- Not sufficient for handling a *production environment*
  - Massively, uncontrolled user requests
  - Mass migration of hundreds of TBytes of data
- Content Holders are faced with loosing vast amounts of data
  - Sufficient computational resources in-house?
- There is a clear demand for incorporating Grid or Cloud Technology

wePreserve

# Integrating Virtual Clusters and Clouds

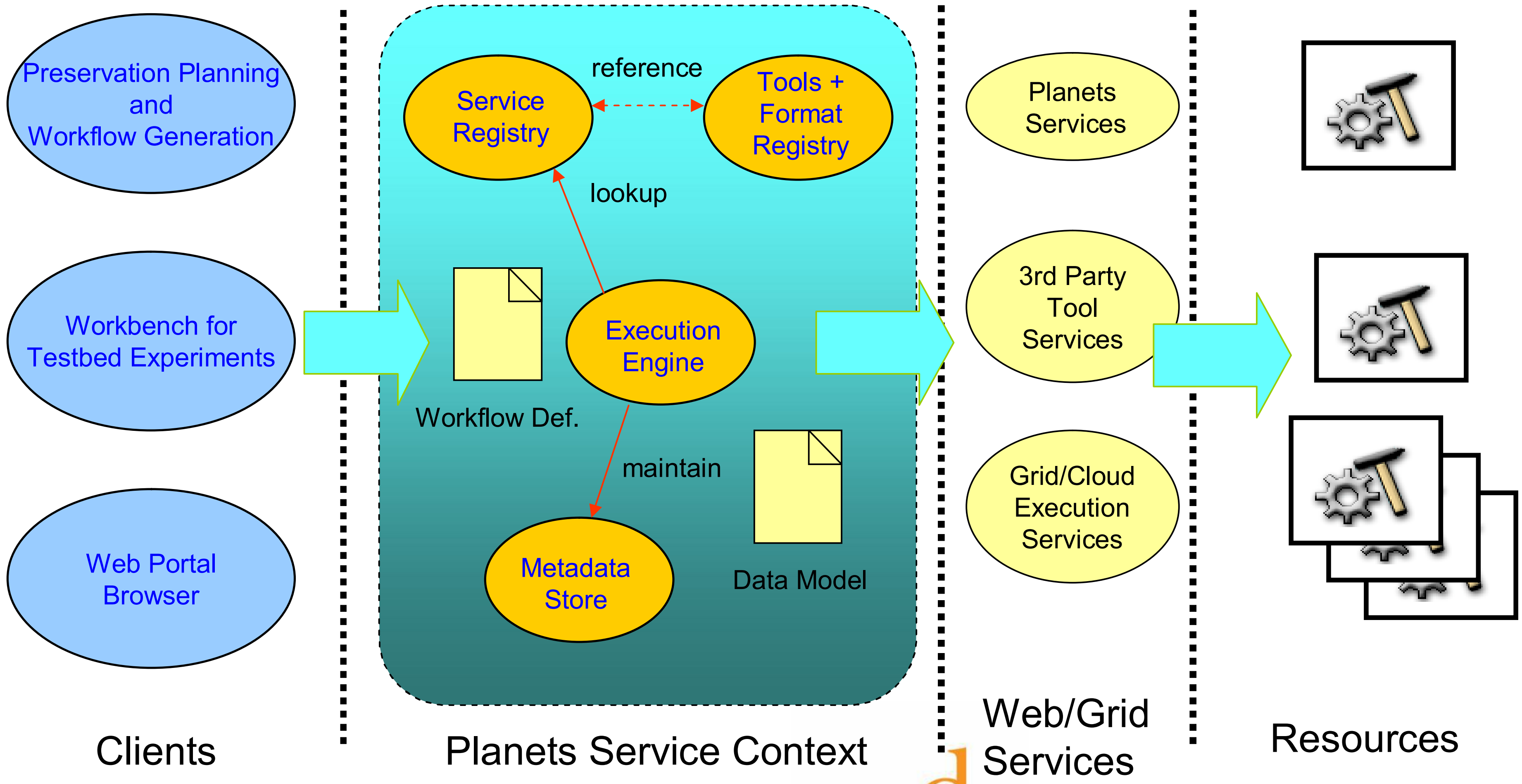
- Basic Idea: Extending Planets SOA with Grid Services
- The Planets IF Job Submission Services
  - Allow Job Submission to a PC cluster (e.g. Hadoop, Condor)
  - Grid approach/standards (SOAP, HPC-BP, JSDL)
- Cluster nodes are instantiated from specific system images
  - Most Preservation Tools are 3rd party applications
  - Software need to be preinstalled on cluster nodes
- Cluster and JSS be instantiated *in-house* (e.g. a PC lab) or on top of (leased) cloud resources (AWS EC2).
  - Computation be moved to data or vice-versa

# Integration – Planets Tiered Architecture

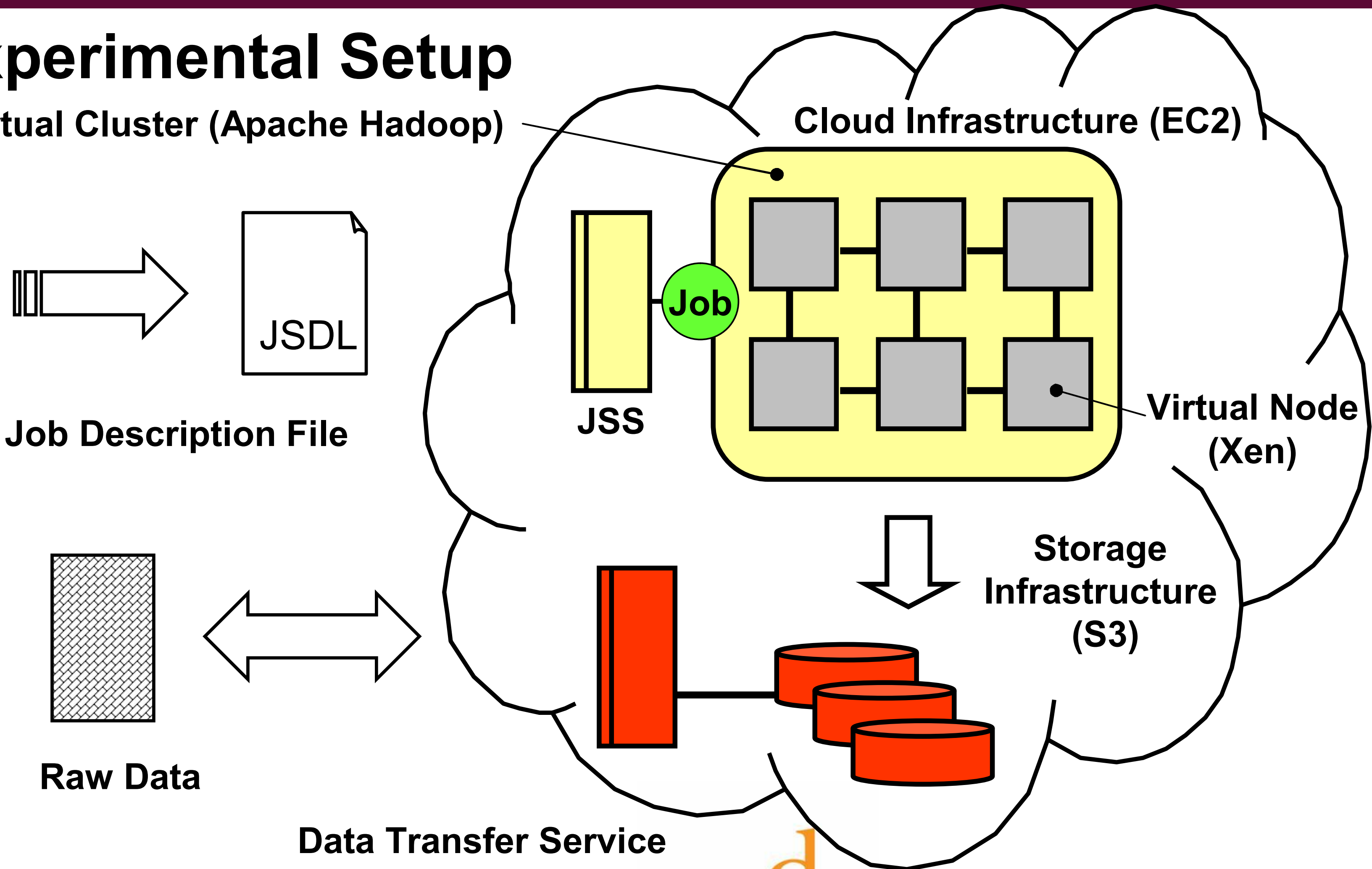


# Experimental Setup

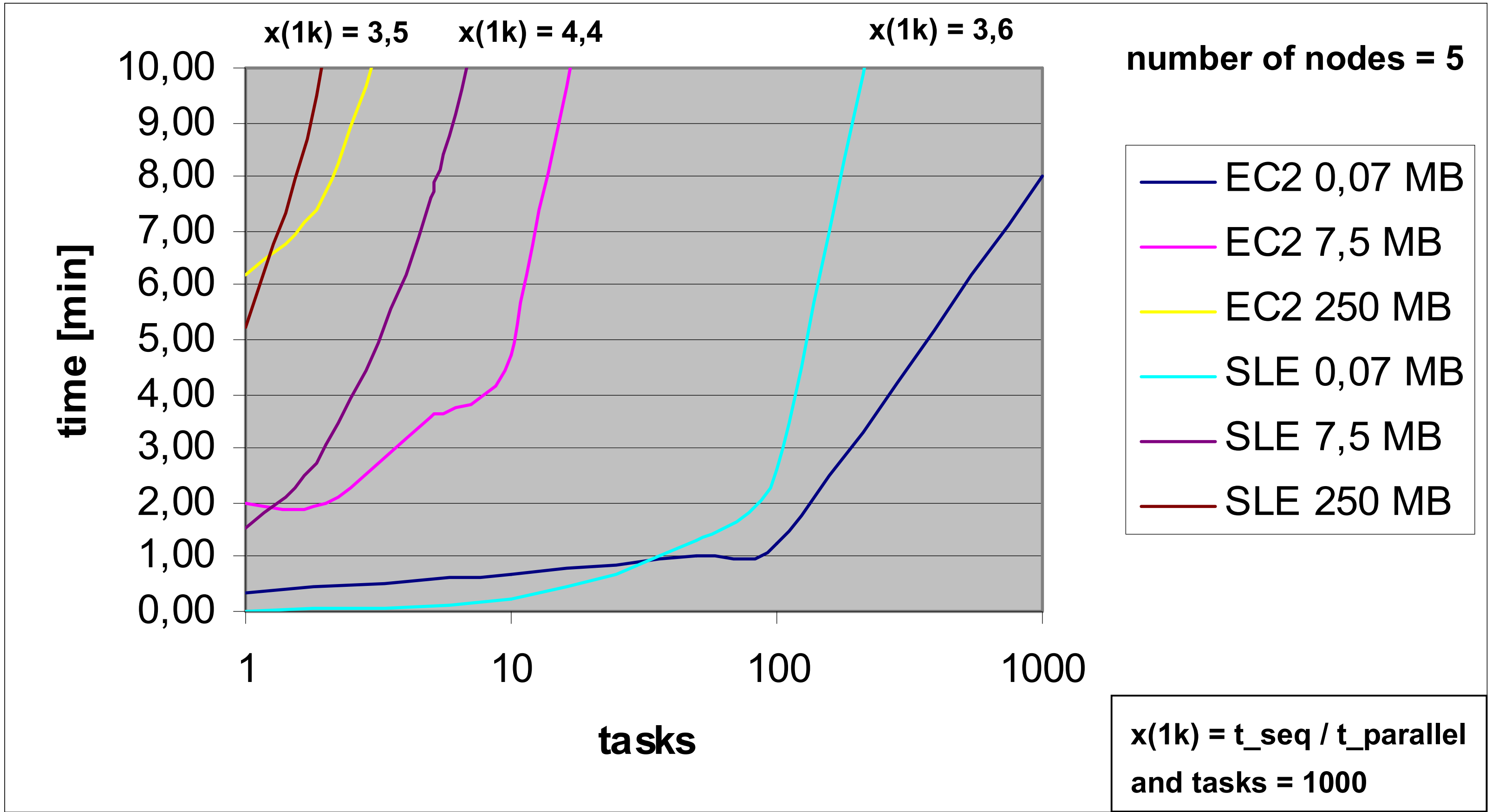
- Amazon *Elastic Compute Cloud (EC2)*
  - 1 – 150 cluster nodes
  - Custom image based on RedHat Fedora 8 i386
- Amazon *Simple Storage Service (S3)*
  - max. 1TB I/O,
  - ~32,5MBit/s download / ~13,8MBit/s upload (cloud internally)
- Apache Hadoop (v.0.18)
  - MapReduce Implementation
- Pre-installed command line tools (e.g, ps2pdf )


Preservation Planning and Workflow Generation
Workbench for Testbed Experiments
Web Portal Browser
Service Registry
reference
Tools + Format Registry
lookup
Workflow Def.
Execution Engine
maintain
Metadata Store
Data Model
Planets Services
3rd Party Tool Services
Grid/Cloud Execution Services
Clients
Planets Service Context
Web/Grid Services
Resources

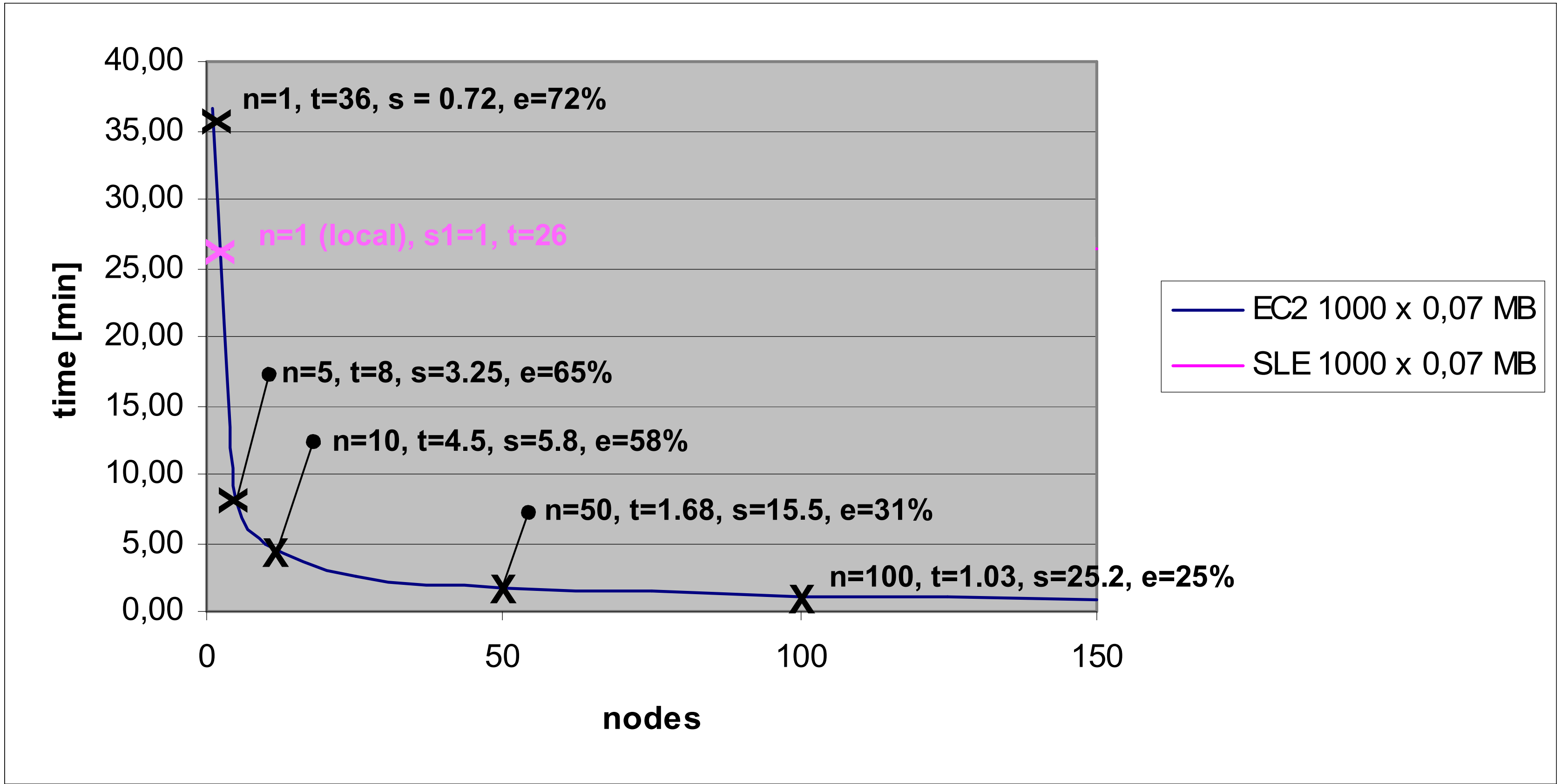
# Experimental Setup



# Experimental Results 1 – Scaling Job Size



# Experimental Results 2 – Scaling #Nodes



# Conclusions

- Preservation systems will need to employ Grid/Cloud resources
  - Therefore there is a need to bridge communities in the areas of digital libraries and e-science.
- Cloud and virtual infrastructures provide a powerful solution for obtaining on-demand access to computational resources.
- Planets IF Job Submission Service provides a first step
  - Submission to virtual cluster of preservation nodes using Grid protocols.
  - Performance scales roughly with the number of nodes, accounting for expected overheads
  - Many open issues remain! Security, reliability, standardization, legal aspects...

# Thank you for your attention!

- Planets Project

  http://www.planets-project.eu

- Contacts


Ross King ross.king@arcs.ac.at

Rainer Schmidt rainer.schmidt@arcs.ac.at

# Sample JSDL code

```xml
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:JobDefinition xmlns="http://www.example.org/"
    xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"
    xmlns:jsdl-posix="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://schemas.ggf.org/jsdl/2005/11/jsdl jsdl.xsd ">
  <jsdl:JobDescription>
    <jsdl:JobIdentification>
      <jsdl:JobName>start vi</jsdl:JobName>
    </jsdl:JobIdentification>
    <jsdl:Application>
      <jsdl:ApplicationName>ls</jsdl:ApplicationName>
      <jsdl-posix:POSIXApplication>
        <jsdl-posix:Executable>/bin/ls</jsdl-posix:Executable>
        <jsdl-posix:Argument>-la file.txt</jsdl-posix:Argument>
        <jsdl-posix:Environment name="LD_LIBRARY_PATH">/usr/local/lib</jsdl-posix:Environment>
        <jsdl-posix:Input>/dev/null</jsdl-posix:Input>
        <jsdl-posix:Output>stdout.${JOB_ID}</jsdl-posix:Output>
        <jsdl-posix:Error>stderr.${JOB_ID}</jsdl-posix:Error>
      </jsdl-posix:POSIXApplication>
    </jsdl:Application>
  </jsdl:JobDescription>
</jsdl:JobDefinition>
```

wePreserve

# Map-Reduce for Migrating Digital Objects

- Map-Reduce implements a framework and prog. model for processing large documents (Sorting, Searching, Indexing) on multiple nodes.
  - Automated decomposition (split)
  - Mapping to intermediary pairs (map), optionally (combine)
  - Merge output (reduce)
- Provides implementation for data parallel problems, i/o intensive,
- Example: Conversion digital object (e.g website, folder, archive)
  - Decompose into atomic pieces (e.g. file, image, movie)
  - On each node, convert piece to target format
  - Merge pieces and create new data object

Information Society digital preservation europe planets CASPAR