smart systems
from Science to Solutions

Scalable Services for Digital Preservation

A Perspective on Cloud Computing

Rainer Schmidt, Christian Sadilek, and Ross King

AUSTRIAN RESEARCH CENTERS

Digital Preservation (DP)

- Providing long-term access to growing collections of digital assets.
  - Not just a question of storage
  - Not just a question of files
- Software preservation rather than hardware preservation
  - Prevent objects from becoming uniterpretable bit-streams.
  - Requires establishment of research infrastructures and networks.
    - » Not an out-of-the-box solution
- A number of large EU projects and initiatives are dealing with the implications of digital preservation.
  - FP6/FP7: Planets, CASPAR, DPE, SHAMAN

Planets

- "Permanent Long-term Access through NETworked Services"
- Addresses the problem of digital preservation
  - driven by National Libraries and Archives
- Project instrument: FP6 Integrated Project
- 5. IST Call
- Consortium: 16 organisations from 7 countries
- Duration: 48 months, June 2006 – May 2010
- Budget: 14 Million Euro
- http://www.planets-project.eu/

Why does DP need HPC resources?

- Digital object management systems, repositories, or archives are designed for storing and providing access to large amounts of data.
  - Many different data streams and metadata models
  - Not designed to support continuous data modification.
  - Focus on storage, no support for HPC.
- Digital preservation is an e-science problem
  - Processing vast amounts of complex data (e.g. analyse, migrate),
  - Experiments in distributed and heterogeneous environments,
  - Crossing administrative and institutional boundaries.

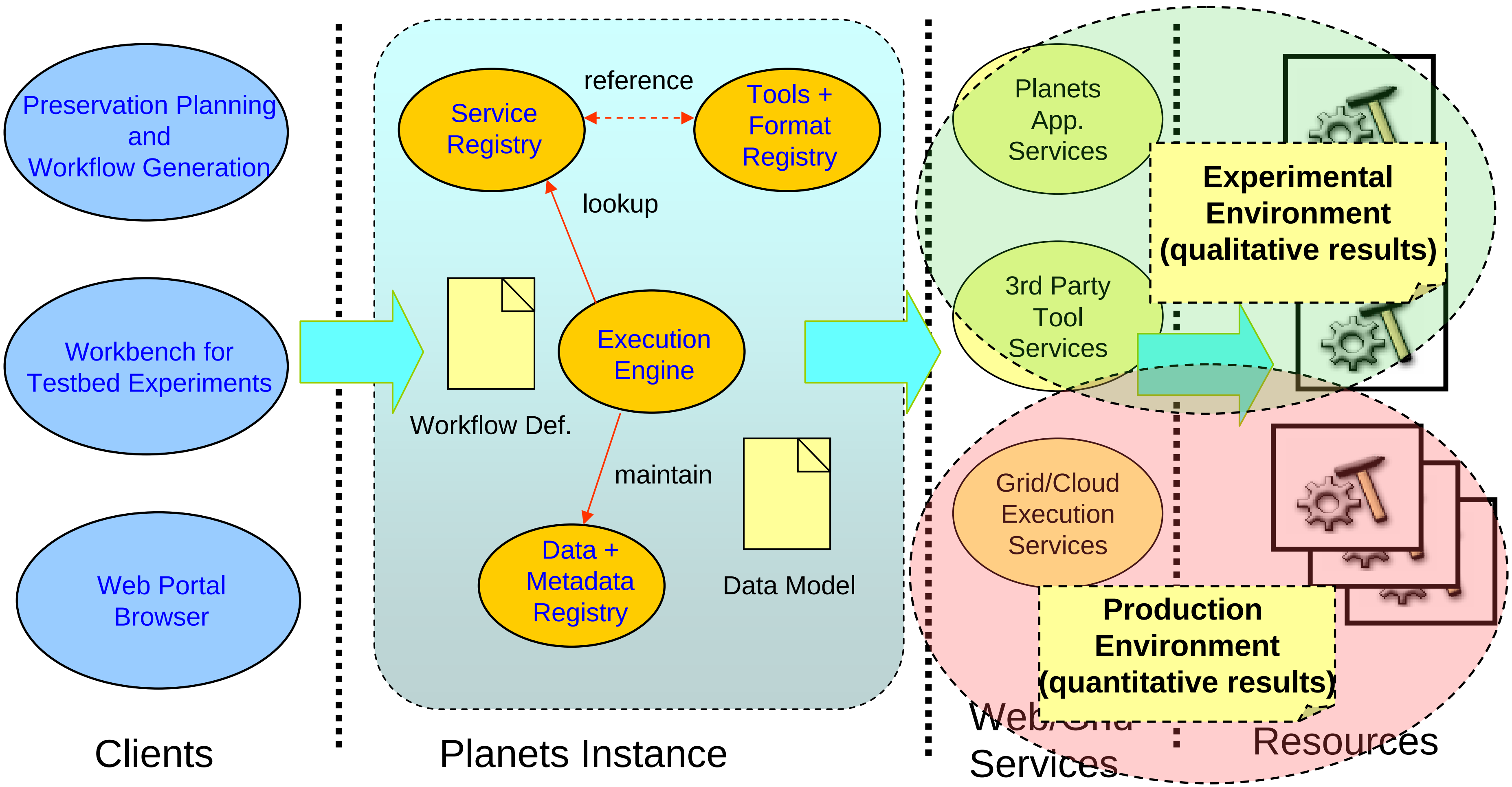
Towards Grid and Cloud Computing

- Planets preservation architecture is based on 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
  - Holding not sufficient computational resources in-house
- There is a clear requirement for incorporating HPC facilities

  -> Grid and Cloud Computing

smart systems
from Science to Solutions

Execution – Tiered Architecture

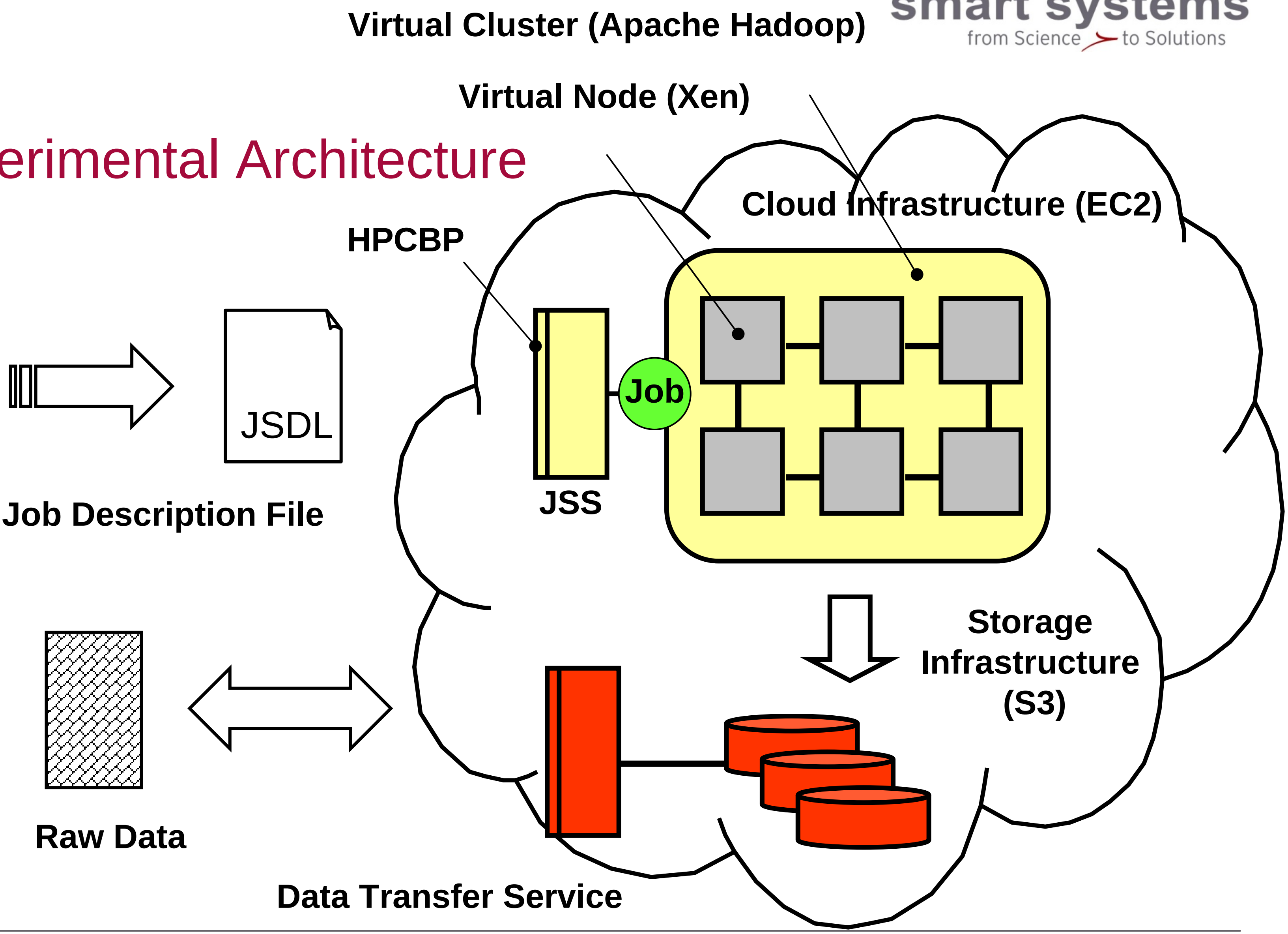


Integrating Clouds and Virtual Clusters

- 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)
  - standard Grid protocols/interfaces (SOAP, HPC-BP, JSDL)
- Cluster nodes are instantiated from pre-configured 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

smart systems
from Science to Solutions

Experimental Architecture



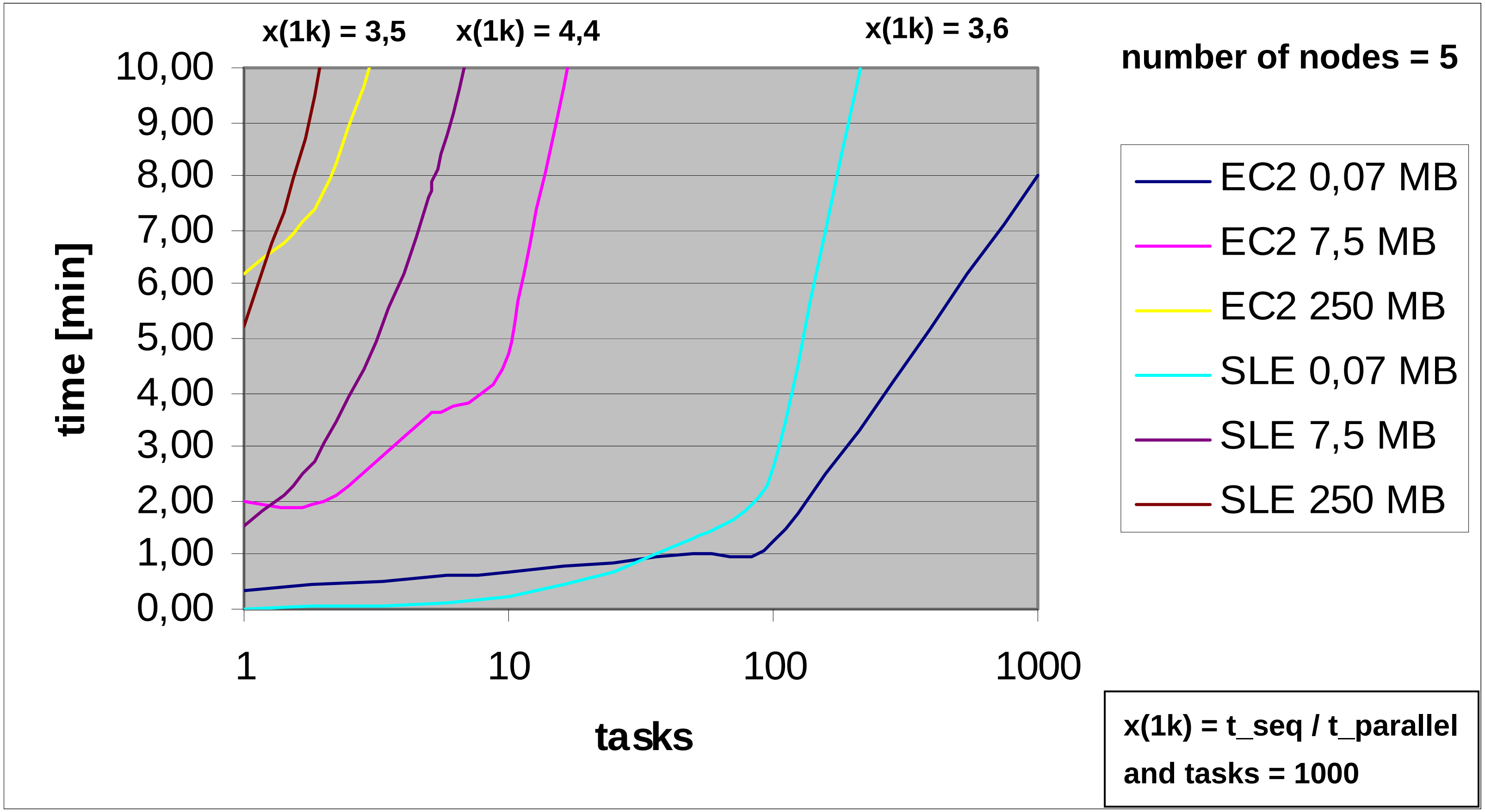
Mass Migration of Digital Objects

- 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

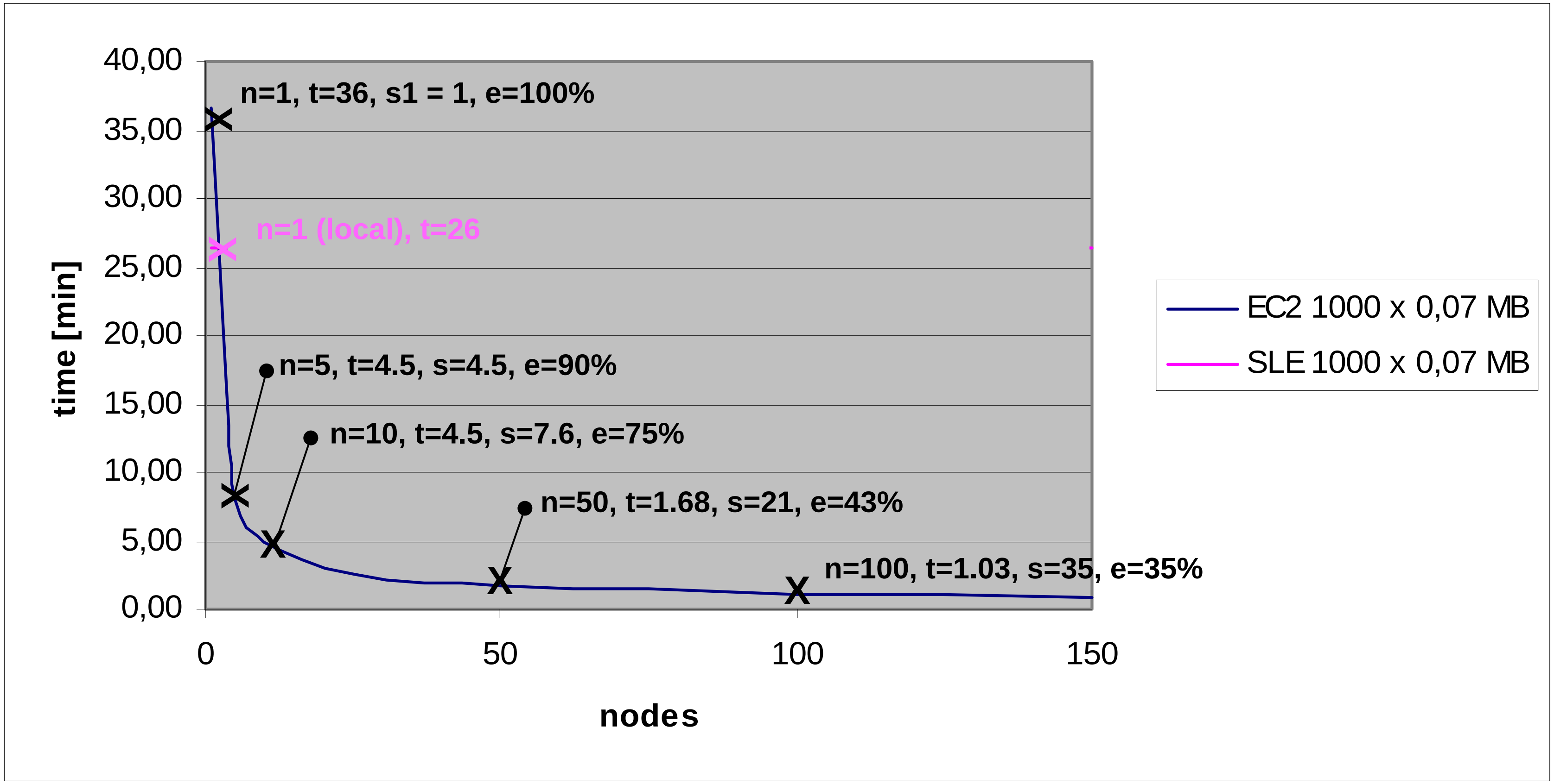
Experimental Results - Setup

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

Experimental Results 1 – Scaling Job Size



Experimental Results 2 – Scaling #nodes



Conclusion

- Preservation systems need to employ HPC resources.
  - Content holders and data repository systems are not ready to utilize computational Grids.
  - There is a need to bridge research communities in the areas of digital preservation and e-science.
- Cloud Computing provides a powerful solution for getting on-demand access to appropriate HPC resources.
  - Many integration issues: Security, Legal Aspects, Reliability, Standardization.
- Planets IF Job Submission Service, a first step.
  - Submission to virtual cluster of DP nodes based on Grid protocols/interfaces.

Fin

smart systems
from Science to Solutions

The Planets Service Framework

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

planets

```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>
```