LAMBDA
The LSDF Execution Framework for Data Intensive Applications

Thomas Jejkal¹,
V. Hartmann¹, R. Stotzka¹, J. Otte², A. García³, J. van Wezel³, A. Streit³

¹Institute for Dataprocessing and Electronics
²Institute of Toxicology and Genetics
³Steinbuch Centre for Computing
LSDF – The Large Scale Data Facility

Motivation

• Growing need for reliable, secured and long term storage of huge amounts of scientific data

Goal

• Long term storage of structured data (data + meta data)
  • Sustainable services
  • Strategies for data migration and lifecycle management

• Support for various communities
  • Collaborative and federated access

Status

• Basic services up and running
  • Disk storage 2 PB (KIT) and 1.4 PB (Univ. of Heidelberg)
  • Computing cluster next to storage (480 cores, 2 TB memory)
LSDF Architecture

Access Layer
- Web-Portal
- High-Level Service APIs
- Fat Clients
- Basic Service APIs

High-Level Services
- (Meta-) Data Management
- Search
- Lifecycle
- Data Intensive Computing
- Monitoring

Basic Services
- GridFTP
- SFTP
- CIFS
- NFS
- Hadoop
- Ganglia
- Lucene

Resources
- (Online-) Disk Storage
- Tape Archive
- Computing Cluster
- Relational Databases
LSDF Architecture

**Access Layer**
- Web-Portal
- High-Level Service APIs
- Fat Clients
- Basic Service APIs

**High-Level Services**
- (Meta-) Data Management
- Search
- Lifecycle
- Data Intensive Computing
- Monitoring

**Basic Services**
- GridFTP
- SFTP
- CIFS
- NFS
- Hadoop
- Ganglia
- Lucene

**Resources**
- (Online-) Disk Storage
- Tape Archive
- Computing Cluster
- Relational Databases
A Basic Use Case (1)

- Scientist A creates data using a special device
  - Data format is device specific

- Scientist B wants to access the data, but is not able to read format

- Local data conversion not feasible
A Basic Use Case (2)

- Scientist A tags meta data before upload
  - Tag points to preconfigured data transformation application
- If upload has finished, processing is started automatically
- After processing, original and transformed data available
Aims

- Easy to use for scientists
- Generic user application description format
- Automatic, asynchronous processing of datasets
- Ensure a proper execution environment
- Decoupling from technologies as far as possible
  - Computing cluster is in constant change
  - Computing service probably changes over time
  - Meta data storage has to be migrated
Institute for Data Processing and Electronics

User Application Meta Data

Oracle JDK 6

java

$INPUT_DIR $OUTPUT_DIR

Dependencies

Execution

Runtime Arguments Executable Arguments

-Xm1024M -jar

Convert.jar

Where does this information come from?
Institute for Data Processing and Electronics

User Application Deployment Cycle (1)

- Application developer requests preparation of dependencies, if needed
- Administrator installs dependencies and registers according meta data
- LSDF uses *Environment modules* package\(^1\) for dependency management
- Notification of application developer on how to use them

\(^1\)http://modules.sourceforge.net/
User Application Deployment Cycle (2)

- Application developer has to validate application, e.g. via SSH login
- Defines runtime (-arguments) and application (-arguments)
  - Parameterization by scientist / end user is not intended
- If application is ready, administrator is contacted to perform deployment
User Application Deployment Cycle (3)

- User application including description sent to administrator
- User application is packed and stored at special location of LSDF
- Administrator registers user application meta data with unique name and version
Automatic Data Processing

- User has to tag and upload data
- Cron job checks meta data periodically
- Wrapper script created from meta data
- Script submission via submitter interface
- Results written back to LSDF storage
Conclusions

The LAMBDA execution framework provides:

- Tools for managing, executing and monitoring data intensive applications
- Meta data schemas for describing data intensive applications
- Definition of responsibilities and corresponding workflows

It is used in production for two use cases:

- Data conversion for microscopy project
- Execution of image processing workflows
Future Work

Dariah

- Complex pattern recognition workflow
- Processing of approx. 180,000 images
  - Repeated for new versions of algorithms

Biology

- New microscope producing 7 TB of data per day
- Image registration has to be applied after upload

Biological image from:
Execution by auto-generated scripts

- Meta data is transformed into execution wrapper script
  - Environment modules package¹ for dynamic software enabling

```
#!/bin/sh
#This script is generated automatically, please do not modify!
#User application developer contact: developer@kit.edu
#Administrator contact: administrator@kit.edu
echo "Starting wrapper script execution..."

#Dependencies
echo "Adding dependency 'JDK6"
module load java6
#Start of user application execution
echo "Using script parameters '$INPUT_DIR $OUTPUT_DIR'
echo "Changing current directory to working directory '$PROJECT_PATH/workingDir'
cd "$PROJECT_PATH/workingDir"
echo "Starting execution at: 'date"
java -Xmx2048M -jar ImageJ.jar $INPUT_DIR $OUTPUT_DIR
EXIT_CODE=$?
echo "Received exit code $EXIT_CODE at: 'date"
exit $EXIT_CODE
#End of user application execution
```

¹http://modules.sourceforge.net/

Institute for Data Processing and Electronics
Configuration and Access

- Access currently via prototypical fat client(s)
- Different views depending on user role
  - Definition by developer, configuration by administrator, usage by scientist
Challenges for Data Intensive Computing

• Users from different disciplines
  • Mostly no experts in Linux, Clusters, Grid/Cloud computing, Hadoop…
    …sometimes very detailed pretending
  • Users will change over time (training would be time consuming)
• Technologies changed/updated during “real” long term storage
  • Execution environment will change
  • Job submission technologies will change
• Documentation and reproducibility
  • Which applications in which versions were applied to the data?
  • Can you repeat this, please?
What is LAMBDA?

LAMBDA is a framework to allow the asynchronous execution of large scale applications. [...] The main goal of LAMBDA is to simplify large scale data processing for scientific users by reducing complexity, responsibility and error-proneness.

Aims:

- Sustainability due to decoupling from technologies
  - Simple adaptation to new technologies
- Reliability due to automatically configured execution environment
- Self-describing applications and versioning by using meta data
- Support for different types of users
LAMBDA User Roles

• Scientist
  • Own field of expertise
  • Focus on research, not on tooling
  • Can be expected to select pre-configured applications

• Developer
  • Application developer in a specific domain
  • Must be trusted concerning application quality and behavior
  • Can be expected to name prerequisites

• Administrator
  • Expert in computing center administration
  • Is allowed to deploy applications and libraries
  • Can be expected to manage the execution environment, install dependencies, analyze system-specific errors
LAMBDA Meta Data

Generic Workflow Meta Data
- Describes execution of one or more applications on dataset
- Execution can be either sequentially or in parallel

Configuration Meta Data
- Describes how to execute an application, including needed dependencies, runtime environment and arguments

Dependency Meta Data
- Describes dependencies (e.g. system libraries) and how they can be loaded dynamically (opt.)

Runtime Meta Data
- Describes the runtime environment needed to execute an application and how to call it (opt.)

User Meta Data
- Describes users by distinguished name
Decoupling from Technologies

Basic Services
- LAMBDA Control implemented as CL application started via CRON daemon
- IDataAccessor interfaces meta data storage (currently RDBMS via Hibernate)
- Abstract Data Layer Access API (ADALAPI) used for access to LSDF
  - Currently direct access via NFS, may change in future
- ISubmitter interfaces computing technology

Resources
- (Online-) Disk Storage
- Tape Archive
- Computing Cluster
- Relational Databases
Conclusions

- Sustainability due to decoupling from technologies
  → Can be used with any technology capable of executing shell scripts
  → Used technology can be chosen by configuration

- Simple adaptation to new technologies
  → All dependent technologies interfaced

- Reliability due to automatically configured execution environment
  → Dependencies configured in beforehand via meta data,
    scientist/developer can rely on functional runtime environment

- Self-describing applications and versioning by using meta data
  → Execution described by meta data and can be reproduced at any time

- Support for different types of users
  → Different user roles determine responsibilities