The Abdus Salam
International Centre for Theoretical Physics

United Nations
Educational, Scientific and
Cultural Organization

IAEA
International Atomic Energy Agency

2068-25

Advanced School in High Performance and GRID Computing - Concepts and Applications

30 November - 11 December, 2009

Grid middleware and arc overview

S. Maffioletti
University of Zurich
Switzerland

Part I
Introduction to ARC:
Advance Resource Connector

Sergio Maffioletti
Grid Computing Competence Center GC3
University of Zurich
sergio.maffioletti@gc3.uzh.ch

Table of content

- What is ARC and the community behind
- Application support
- Main features
- Interoperability
- Future perspectives

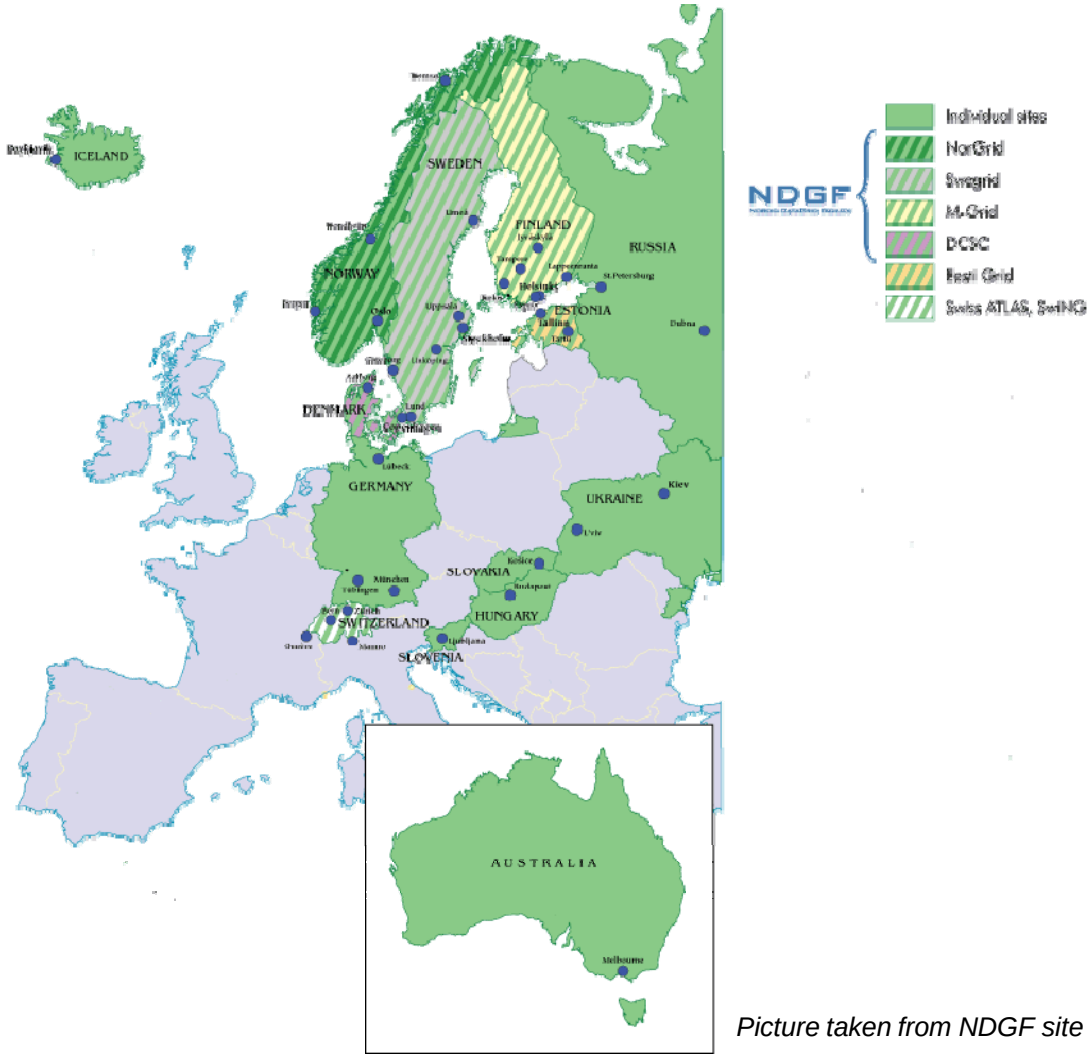
What is ARC

- **NorduGrid** project started in 2001 as a solution for aggregating computing and storage resources from the Nordic countries (Lund, Uppsala, Copenhagen, Oslo, Helsinki) within the scope of the **CERN/LHC** computing project
- NorduGrid built its own Grid middleware
- Named **ARC**, for Advanced Resource Connector

Since May 2002 ARC is extensively used
in **ATLAS** production and
other scientific computing projects

•Switzerland started using ARC in 2005 for a national project **SwissBioGrid**

Supporting countries



Picture taken from NDGF site

Scientific domains support

- ARC is used to make a distributed computing center for High Energy Physics: the NDGF “Tier1”
- Several other scientific domains supported thanks to its ease of use

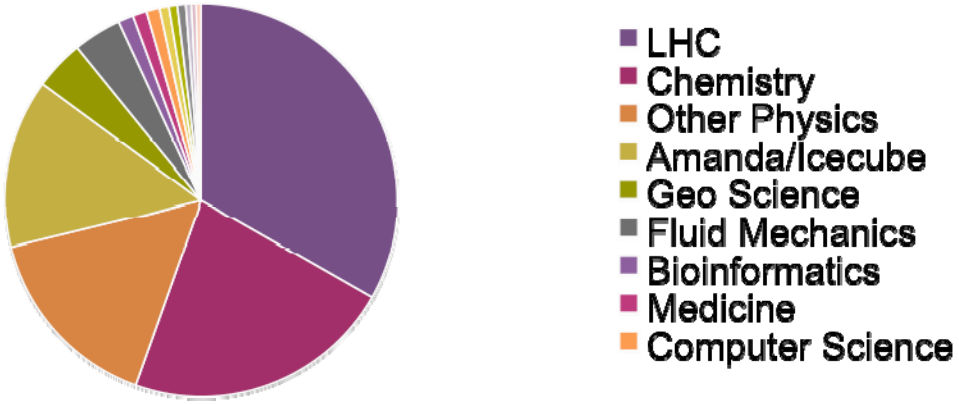


chart taken from NDGF application report 2009

ARC in Switzerland

- **SwiNG**
  - SMSCG (national grid)
  - Grid Portal working group
  - Campus Grid working group

- **Swiss Atlas Grid**
  - UniGE (Tier3)
  - UniBE (Tier3)
  - ETH/CSCS (Tier2)
  - Switch (regional GIIS)

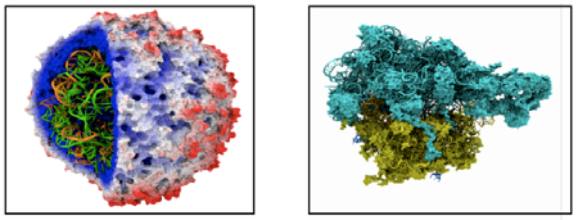
Partners

- **Haute Ecole Spécialisée de Suisse Occidentale** (HES-SO)
  - Grid and Ubiquitous Computing Group
- **University of Geneva** (UniGE)
  - Proteome Informatics Group (PIG)
- **Swiss Institute of Bioinformatics** (SIB) - Vital-IT
- **SWITCH**
- **Università della Svizzera Italiana** (USI)
- **University of Bern** (UniBE):
  - Laboratory for High Energy Physics (LHEP)
  - Informatikdienste
- **University of Zurich** (UZH):
  - Grid Computing Competence Center – GC3
  - Informatikdienste
- **Eidgenössische Forschungsanstalt für Wald, Schnee und Landschaft** (WSL)
- **Eidgenössisches Institut für Schnee- und Lawinenforschung** (SLF)
- **ETH Zurich - Swiss National Supercomputing Centre** (CSCS)
- **Ecole Polytechnique Federale de Lausanne** (EPFL)

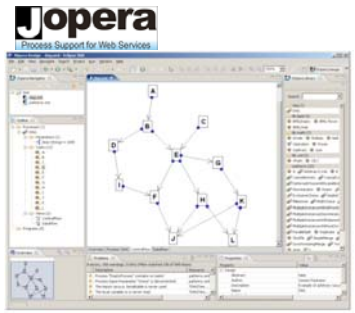
Application support

On the SMSCG Grid the following applications are supported and/or in the progress of being ported:

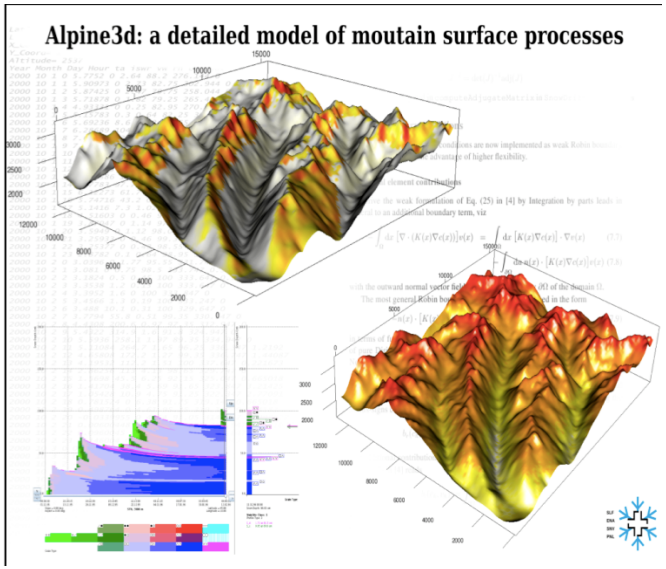
- **NAMD and GROMACS**: biochemistry applications
- **Alpine 3D**: an application for the high resolution simulation of alpine surface processes
- **swissPIT**: Swiss Protein Identification Toolbox
- **POP-C++**: Parallel Object Programming framework
- **JOpera**: open grid workflow management system based on the Eclipse platform
- **RSA768**: cryptographic application

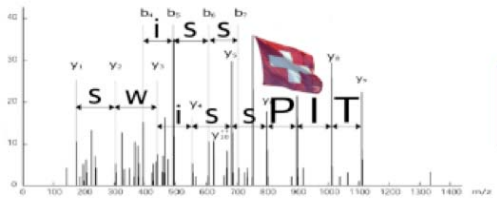


Nanoscale Molecular Dynamics and Ribosome Computed by NAMD
(http://www.computerweekly.com)

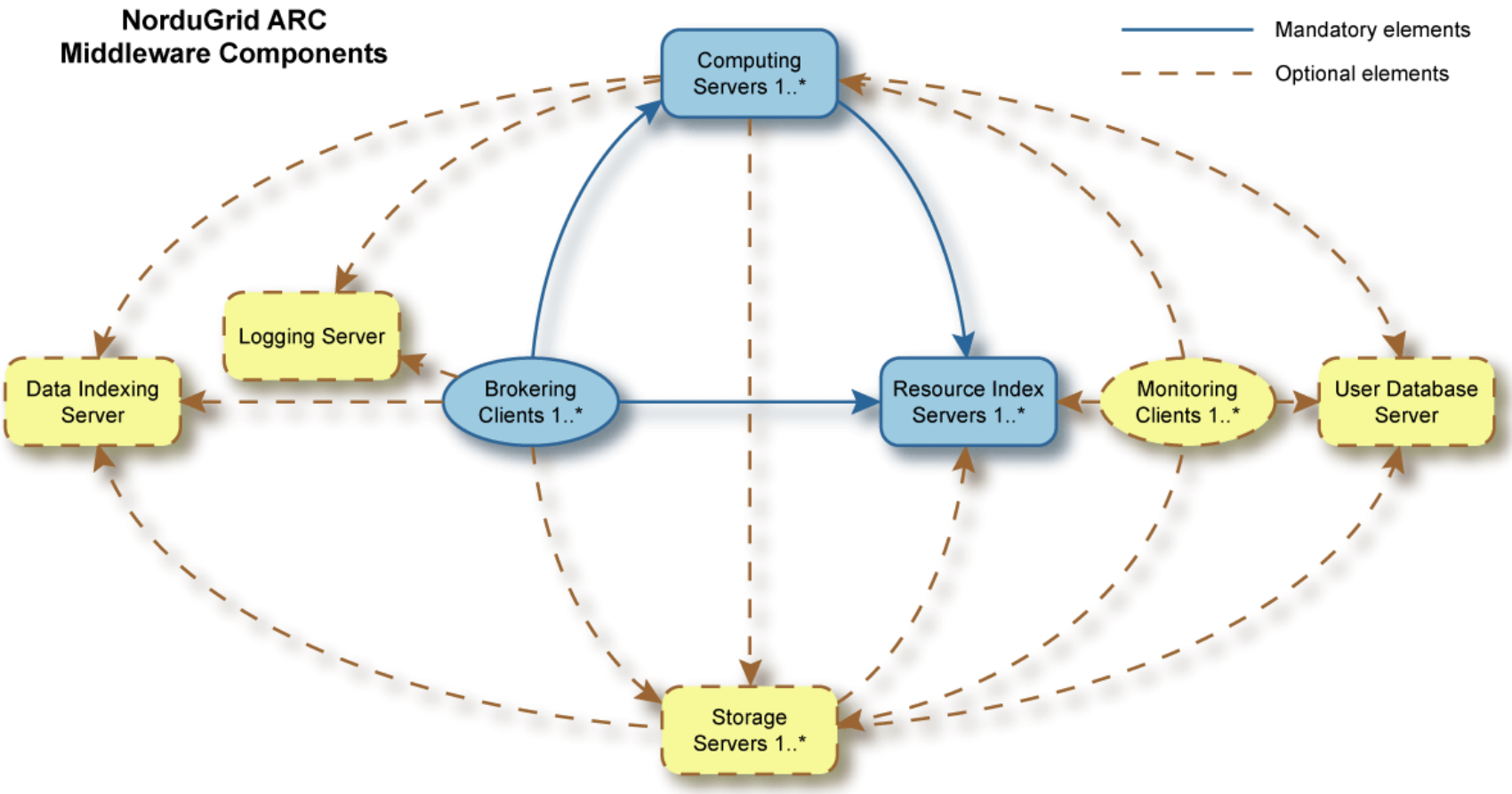


JOpera Design Perspective on a complex DAG Control flow





ARC in a nutshell



Picture taken from "ARC meet SwiNG" workshop 2008

ARC main features

- Lightweight **standalone** client package, easy to install and use
- **Reliable resource** for scientific applications in many research fields
- Available on a **wide range of Linux platforms**
- **Non-centralized** architecture
- Needs **no centralized operations** infrastructure
- **Non-intrusive**, coexists with other software and configurations

Production ARC overview

- **Current ARC version in production: v0.6.5**
  - http://www.nordugrid.org - Open Source (GPL v2, next versions - Apache 2.0)
  - Binary packages for several Linux flavors (RH, Fedora, SuSE, Debian, Ubuntu)
- **Reliable implementation of basic Grid functionalities**
  - *De-facto* standard Grid security: GSI, VOMS, GACL
  - Job submission: by matchmaking/brokering or direct
  - Job monitoring, logging and life cycle management
  - Information services: resource aggregation, discovery and monitoring
  - Basic data management:
    - Interface to data indexing services (e.g. LFC), client-side data movement
    - Storage Elements (GridFTP, SRM – own or 3rd party) – NDGF uses d-Cache
- **Builds upon standard Open Source solutions and protocols**
  - Globus Toolkit® pre-WS API and libraries
  - OpenLDAP, OpenSSL, SASL, SOAP, GridFTP, GSI

Interoperability

- **Strategy**: interoperability via open standards
  - BES, JSDL, GLUE2, SRM, GridFTP, X509, SAML etc
- **Shorter term**: transitional gateway-like solutions are available (ARC-gLite)
- **Currently in development**:
  - ARC client library addresses the ARC → other middleware direction
  - CLI will offer transparent access capability to 3rd party services
- **Primary target platforms**: gLite, Unicore
  - New ARC client can already now submit jobs to CREAM, Unicore compute elements

Future perspectives

- ARC evolves from a pre-WS solution to a Web Service based one
- ARC consortium (NorduGrid, NDGF, KnowARC et al), together with gLite and Unicore, contribute to creation of the ***Universal Middleware Distribution (UMD)*** for the European Grid Initiative (EGI)
  - Sites and VOs that use ARC will get an access to the European e-Science infrastructure, just like those that use gLite or Unicore

Further information

- Lots of documentation, presentations and tutorials on the NorduGrid web site: http://www.nordugrid.org
- ARC mailing lists:

  nordugrid-support@nordugrid.org

  Nordugrid-discuss@nordugrid.org

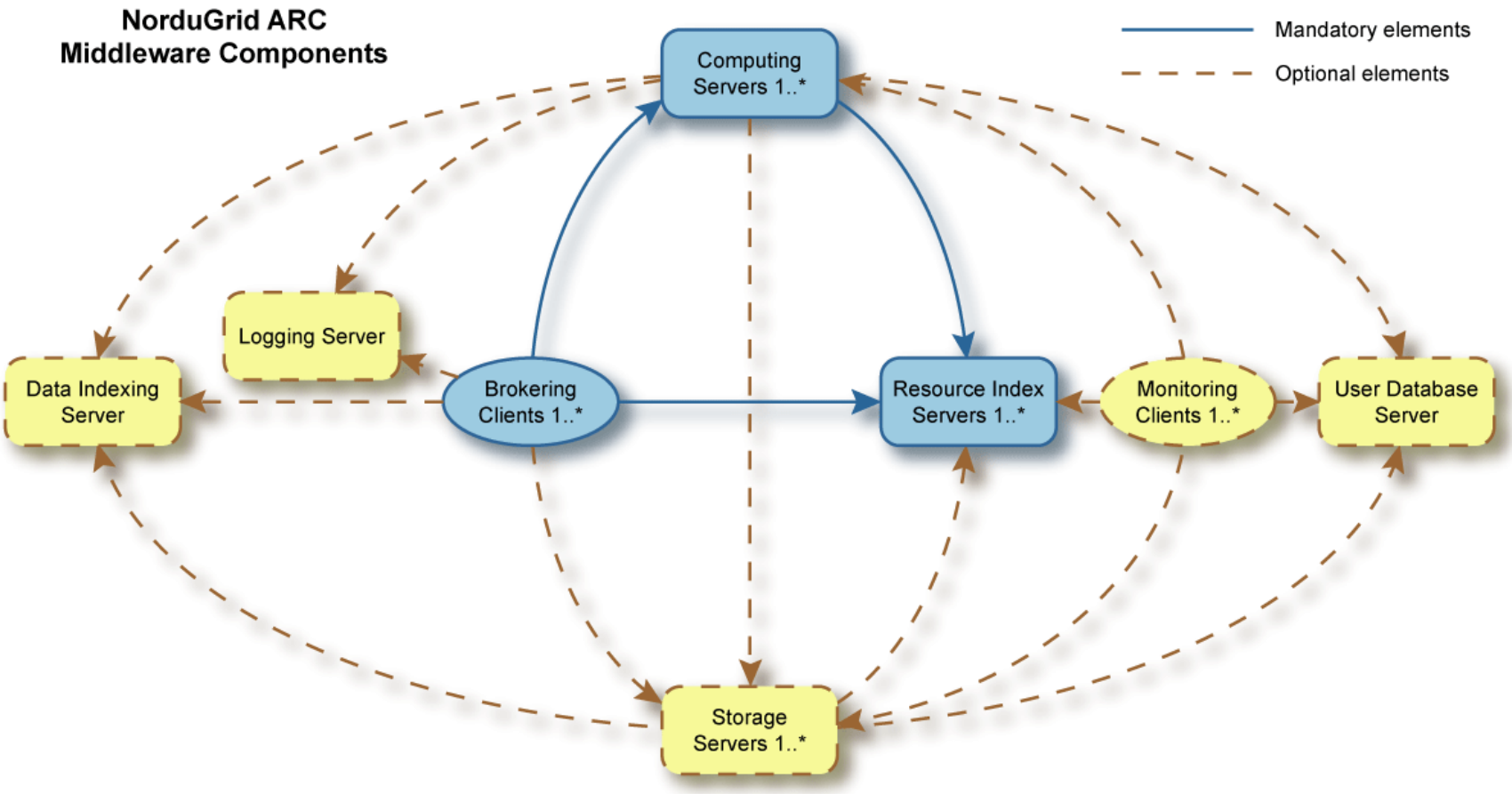
- SwiNG website: http://www.swing-grid.ch
- SMSCG site: https://www.smscg.ch

Part II
Inside ARC

Sergio Maffioletti
Grid Computing Competence Center GC3
University of Zurich
sergio.maffioletti@gc3.uzh.ch

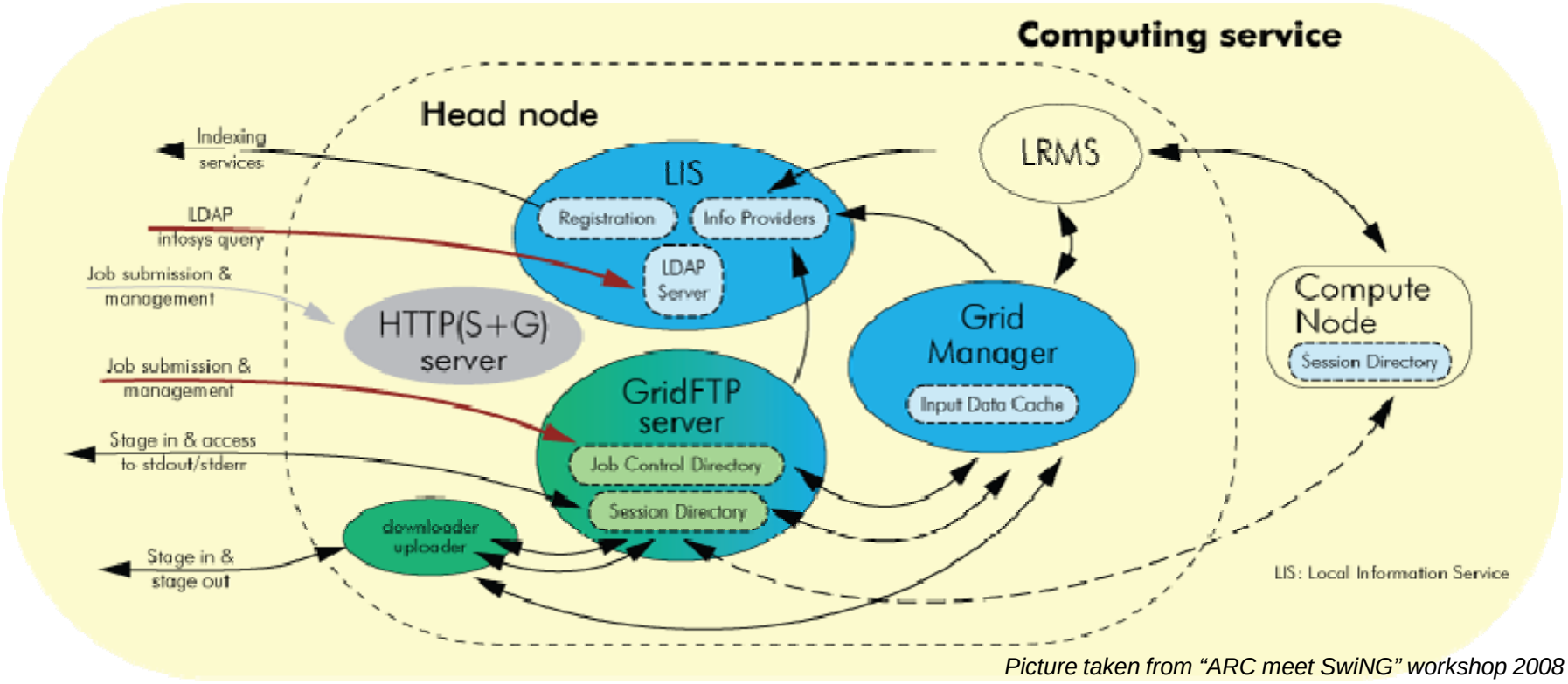
Table of content

ARC in a nutshell



Picture taken from "ARC meet SwiNG" workshop 2008

ARC Computing Service



Picture taken from "ARC meet SwiNG" workshop 2008

- Computing resources: Grid-enabled via ARC layer on head node (front-end):
  - Custom GridFTP server for all the communications
  - Grid Manager handles job management upon client request, interfaces to LRMS
  - Performs most data movement (stage in and out), cache management
  - Publishes resource and job information via LDAP

ARC Client

Lightweight User Interface with the built-in Resource Broker

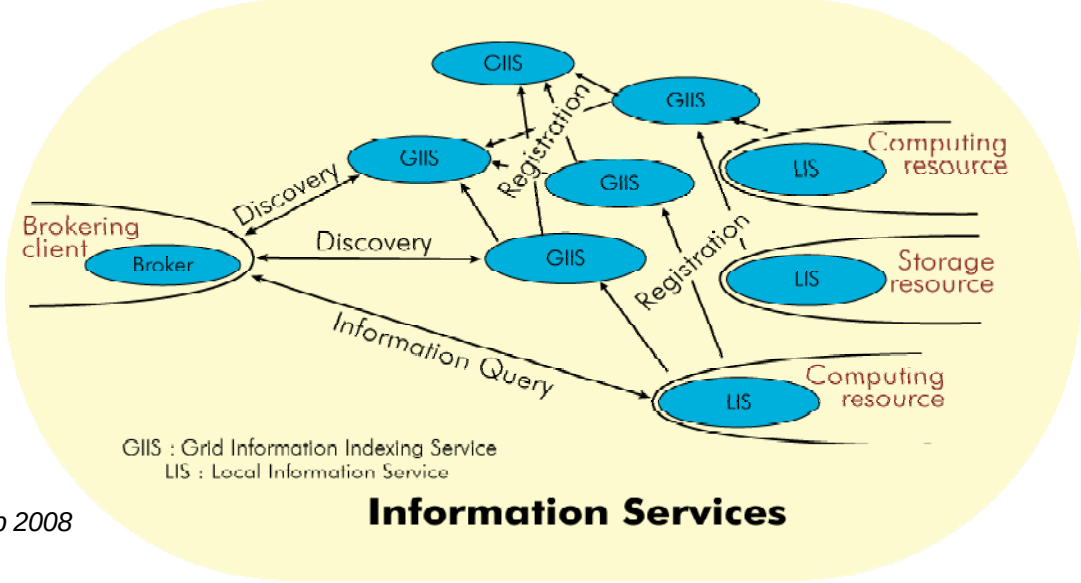
- A set of command line utilities
- Minimal and simple
- Under the hood: resource discovery, matchmaking, optimization, job submission
- Complete support for single job management
- Basic functionality for multiple job management
- Built upon ARCLIB

Standalone binary client package possible to be installed in user space

ARC Information System

Information System: based on Globus-patched OpenLDAP

- It uses GRIS and GIIS back-ends
- Effectively provides a pseudo-mesh architecture, similar to file sharing networks
- Information is only kept on the resource; never older than 30 seconds
- Keeps strict registration hierarchy
- Own schema and providers

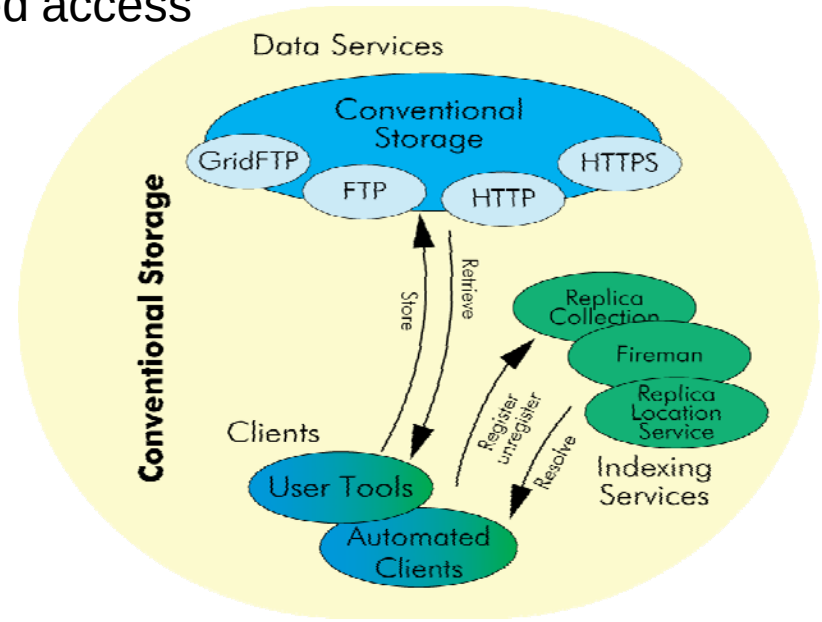


Picture taken from "ARC meet SwiNG" workshop 2008

ARC Storage Service

Conventional Storage:

- Own GridFTP server implementation with pluggable back-ends
- Ordinary file system access
- Grid Access Control Lists (GACL) based access



Picture taken from "ARC meet SwiNG" workshop 2008

ARC for users: Getting started

1. Obtain access to a User Interface (ARC client software)
2. Request a user certificate from a Certification Authority
3. Deploy the signed certificate on the User Interface
4. Create grid proxy
5. Write a job description
6. Submit job
7. Monitor the progress of the job
8. Fetch the results

User Interface

ngsub – find suitable resources and submit a job

ngstat – check the status of jobs and resources

ngcat – display stdout, stderr of a running job

ngget – retrieve the results of a finished job

ngkill – stop a job

ngclean – delete a job from a computing resource

ngsync – find user's jobs

ngls – list files on a storage resource or in job's sandbox

ngcp – transfer files to and from cluster and storages

Basic Job Workflow

- Create proxy: grid-proxy-init
- Writing a job descrption: job.xrsl
- Submitting the job: ngsub
- Checking the status: ngstat / ngcat
- Retrieving the result files: ngget
- Detroy proxy: grid-proxy-destroy

Installing ARC Client

- Required to submit jobs to ARC
- Could be downloaded from http://ftp/nordugrid.org/download
- Various binary packages as well as source code
- Easiest way to get started it to install the standalone package
  - Uncompress in a directory (no root privileges required)

```
tar zxvf nordugrid-standalone-0.6.5-1.i386.tgz
```

  - Run the environment setup script

```
cd nordugrid-standalone-0.6.5-1
. ./setup.sh
```

- RPM packages are recommended for multi user installation

Writing a Job Description File

- Resource Specification Language (RSL) files are used to specify job requirements and parameters for submission
  - ARC uses an extended language (xRSL) based on the Globus RSL
- Similar to scripts for local queuing systems, but includes some additional attributes
  - Job name
  - Executable location and parameters
  - Runtime Environment requirements

xRSL Example

- helloWorld.sh

```
#!/bin/sh
echo “Hello World”
```

- helloWorld.xrsl

```
& (executable=helloWorld.sh)
  (jobname=hellogrid)
   (stdout=std.out)
   (stderr=std.err)
   (gmlog=gridlog)
   (architecture=i686)
   (cputime=10)
   (memory=32)
```

Basic Operations

- Submit the job

```
ngsub -f helloWorld.xrsl
 => Job submitted with jobid gsiftp://arc-
ce.grid.seed:2811/jobs/455611239779372141331307
```

- Query the status of the submitted job

```
ngstat hellogrid
Job gsiftp://arc-ce.grid.seed:2811/jobs/
455611239779372141331307
Jobname: hellogrid
Status: INLRMS:Q
```

- Most common status values are: ACCEPTED, PREPARING, SUBMITTING, INLRMS:Q, INLRMS:R, EXECUTED, FINISHED

Basic Operations

- Print the job output

```
ngcat hellogrid
```

  - Shows the standard output of the job
  - This can be done also during job execution

- Fetch the results

```
ngget hellogrid
ngget: downloading files to
/home/gridseed01/results/455611239779372141331307
ngget: download successful - deleting job from
gatekeeper.
```

Runtime environment - RTE

- Software packages which are pre-installed on a computing resource and made available through ARC
- Avoid the need of sending the binaries together with the job
- Allows local platform specific optimization
- Provides to the users a common environment for the specific application
- Implemented by shell scripts which initialize the environment and are placed in specific directory
- Required RTE can be specified in the job description file:
  `(runtimeenvironment=APPS/LIFE/TANDEM-09.08)`
- Every infrastructure should provide a registry for the supported RTEs and the conventions followed

Runtime environment - RTE

Deployment and RTE: APPS/LIFE/TANDEM-09.08

```
..
export TANDEM_LOCATION=$application_base_path
Export TANDEM_TAXONOMY=$TANDEM_LOCATION/bin

# Set the specific mdrun commands for this system.
export TANDEMRUN="$TANDEM_LOCATION/bin/tandem.exe”
..
```

Runtime environment - RTE

In xrsl job description file

```
(runtimeenvironment="APPS/LIFE/TANDEM-09.08")
```

Within job execution

```
..
$TANDEMRUN input.xml
..
```

ARC for sysadmins

Installation of ARC packages:

- For most rpm-based Linux distributions, RPMs for ARC and for most of its dependencies are provide through nordugrid repository
- Possible to install via apt or yum
  - yum groupinstall “ARC Server”
  - Provided deb packages
  - Non-official (but working) support for Gentoo distribution

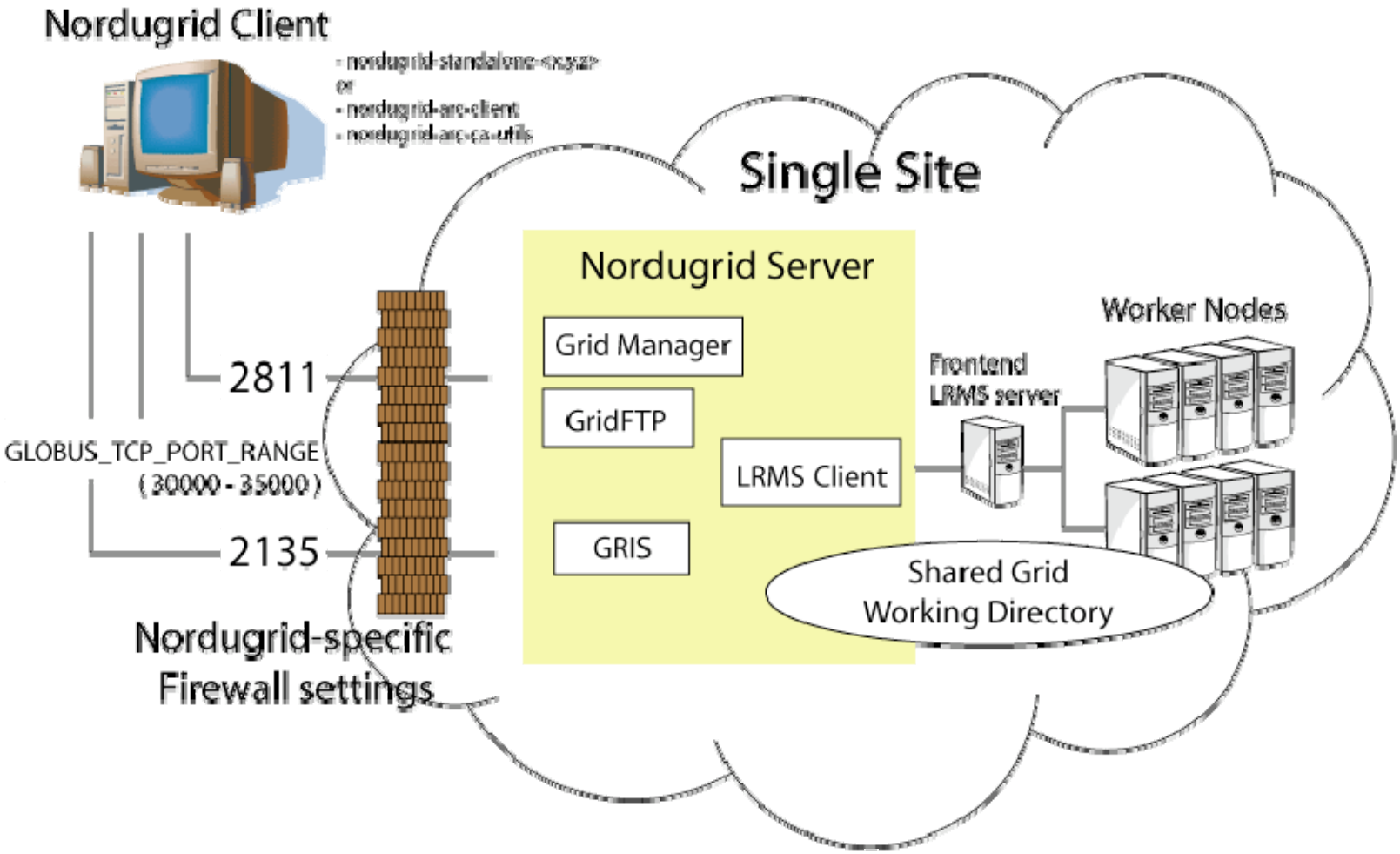
ARC system requirements

- ARC can be see as made of four main service type:
- ARC_CE: interface with the computing farm
- ARC_UI: client interface
- ARC_SE: interface with the storage farm
- ARC_GIIS: top level information system

- Each of them can be installed either separately or altogether on the same node
- RPMs are provided for ARC server and ARC client
- ARC server includes components for CE,SE,GIIS
- System administrator decides which service configure and enable through configuration files

ARC_CE

- Given a computing farm controlled and managed by a Local Resource Management System (LRMS)
- ARC_CE is the interface to the LRMS
- ARC_CE needs to be an authorized client of the LRMS
- ARC_CE needs to share at least one filesystem with the rest of the computing farm
- Submission to the LRMS is done by ARC_CE on behalf of the users
- ARC_CE checks the status of the LRMS jobs and retrieves the results on behalf of the user
- Results form the LRMS submission are stored on ARC_CE for manual retrieval or transfer to a storage resource


Nordugrid Client
- nordugrid-standalone-<x.y.z>
or
- nordugrid-arc-client
- nordugrid-arc-ca-utils
Single Site
Nordugrid Server
Grid Manager
GridFTP
LRMS Client
GRIS
Worker Nodes
Frontend
LRMS server
Shared Grid
Working Directory
2811
GLOBUS_TCP_PORT_RANGE
(30000 - 35000)
2135
Nordugrid-specific
Firewall settings

Resource selection

- ARC_UI embodies a resource broker that is responsible of selecting the resources to match the requirements of a submitting job
- Broker first queries the GIIS it knows to get a list of sites
- Then queries the sites to check whether the user is authorized to the site
- Then filters the resources according to the ARC_job's resource specifications
- Then ranks the filtered resources according to its policy (random, fastest cpus, …)
- The top rank resource is selected
- Submission to selected resource

Lifecycle of a job on ARC_CE

- An ARC_job is submitted from ARC_UI
- On ARC_CE, the Gridftp server accept the request
- Authentication and authorization (GSI,VOMS)
  - Request is mapped to local user account
- An ARC_jobID is created (this will be the unique reference for the job)
- A session folder is created within $sessiondir (as specified in arc.conf) named as the ARC_jobID
- Downloader process is started to fetch input data
- Input data are stored in ARC_job's session dir
- submit-$LRMS-job script is started to translate ARC_job into a local submission
  - There are several LRMS backend: PBS, SGE, LL, LSF, Condor,…

Lifecycle of a job on ARC_CE

- Translated job is submitted to LRMS using local user account
- Lifecycle of LRMS_job is supervised by grid-manager
- Information system updates information on the status of the job (INLRMS:R means submitted to LRMS and running there)
- Once LRMS_job is terminated, results are retrieved in job's session dir
- Uploader process takes care of staging results to a designated storage resource (if specified in xrsl)
- ARC_job status is reported as FINISHED