XtreemOS

Enabling Linux
for the Grid

XtreemOS Linux-based Grid Operating System

Christine Morin, INRIA
XtreemOS Scientific coordinator

MasterClass, Amsterdam, VUA, October 23, 2008

XtreemOS IP project
is funded by the European Commission under contract IST-FP6-033576

Outline

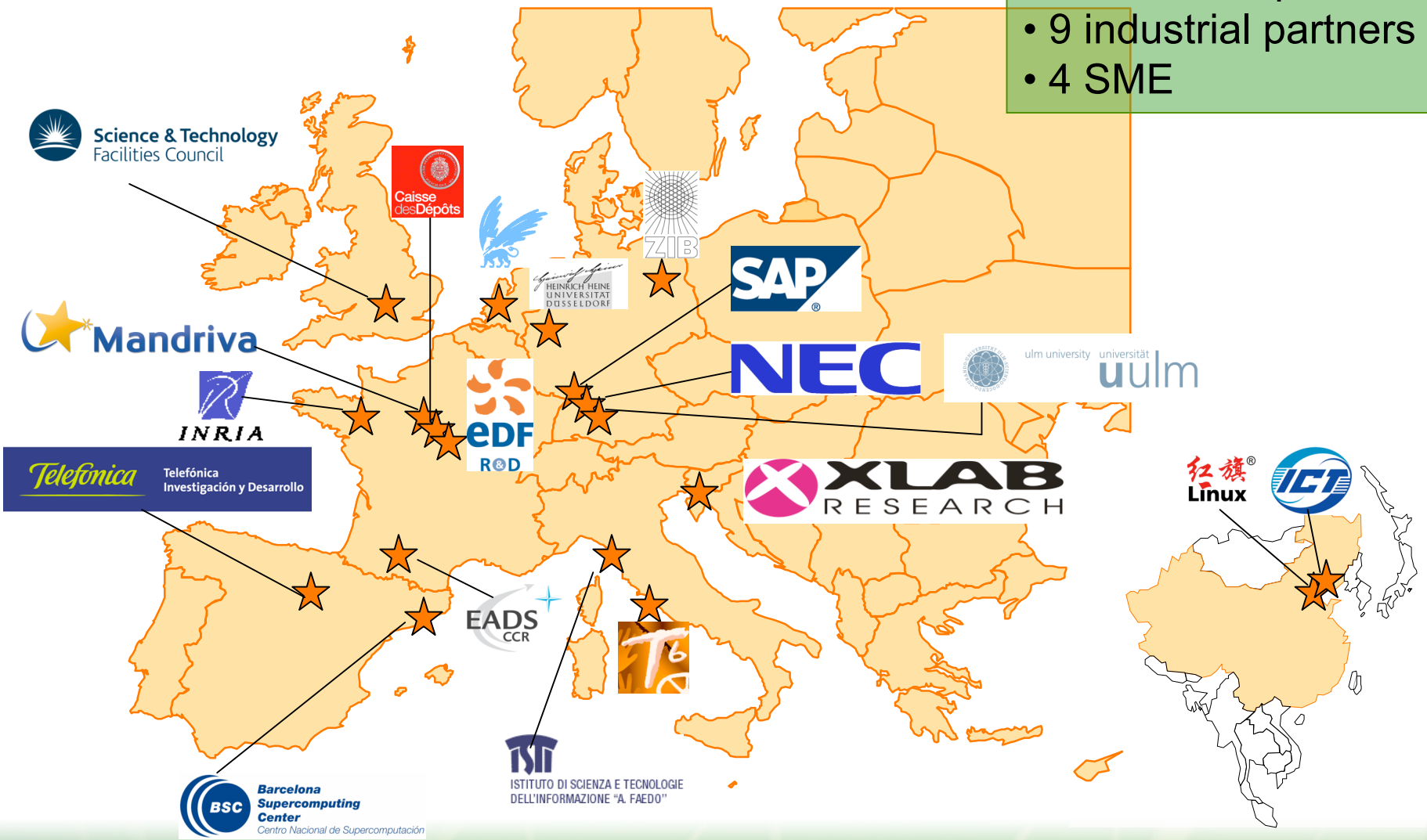
- XtreemOS project
- XtreemOS vision
- VO & security management
- *Demo VO & security*
- Job Management
- Data management
- *Demo user session*
- Infrastructure for scalable & highly available services
- XtreemOS API
- XtreemOS flavours
- Concluding remarks

XtreemOS European R&D Project

- **4-year IP project started in June 2006 in the FP6 framework**
- **30 M€ budget, 14.2 M€ EC grant**
- **19 partners**
  - From 7 European countries & China

XtreemOS Consortium

- 9 academic partners
- 9 industrial partners
- 4 SME



XtreemOS Objectives

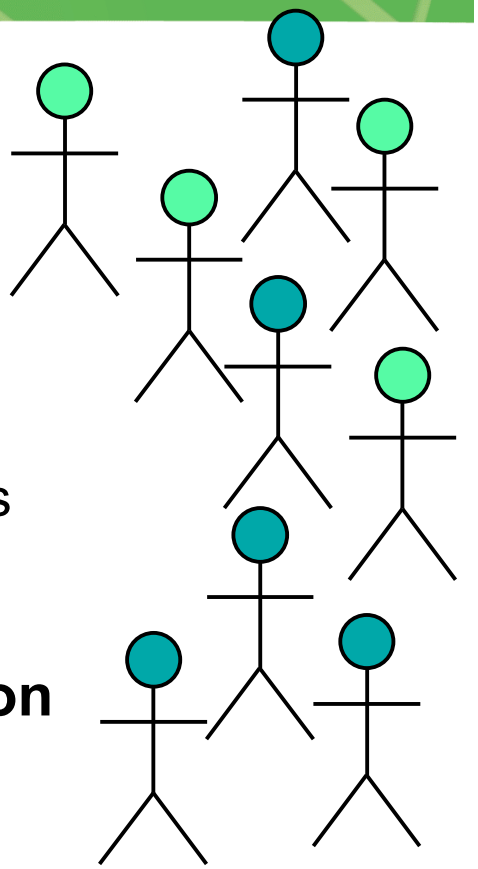
- **Design & implementation of an open source Linux-based Grid Operating System with native VO support**

- **Two fundamental properties: transparency & scalability**
  - Bring the Grid to standard users
  - Scale with the number of entities and adapt to evolving system composition

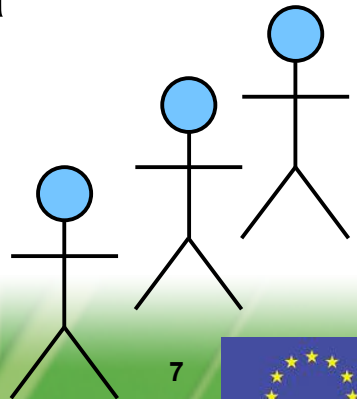
Outline

- XtreemOS project
- XtreemOS vision
- VO & security management
- *Demo VO & security*
- Job Management
- Data management
- *Demo user session*
- Infrastructure for scalable & highly available services
  XtreemOS API
- XtreemOS flavours
- Concluding remarks

Users
End users - Service Administrators

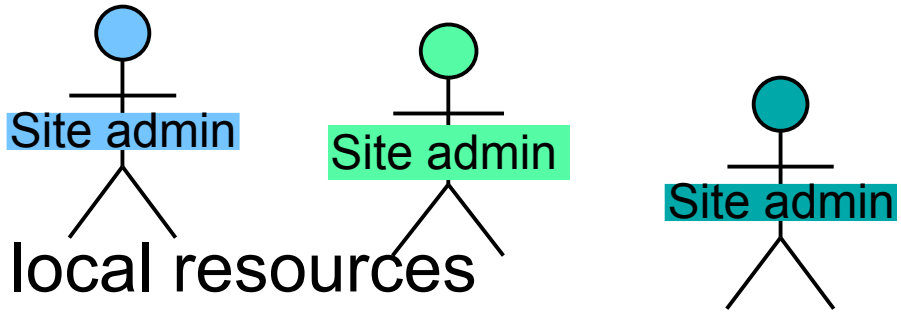


- **Ease of use**
  - Do not want to care with Grid issues
  - Want to work with familiar interfaces
  - Want to use their non Grid-aware legacy applications
  - Simple login as a Grid user in a VO
- **Secure and reliable application/service execution**
- **High performance**
- **Ubiquitous access** to services, applications & data

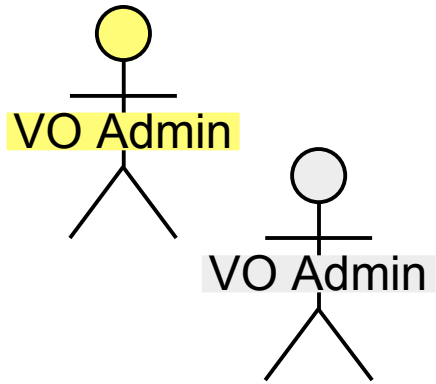


Administrators

- **Site administrators**
  - Ease of management
  - Autonomous management of local resources
  - Should not be impacted by every single change in a VO



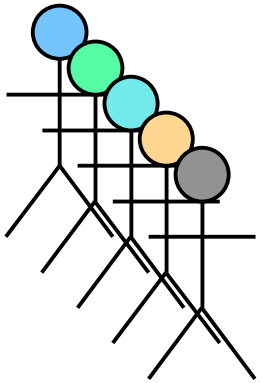
- **VO administrators**
  - Ease of management
  - Flexibility in VO policies
  - Accounting

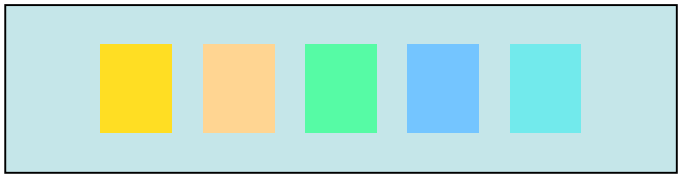


Developers' Needs

- **Ease of development of Grid applications**
  - Reuse existing code
- **Stable API**
- **Conformance to standard API**
  - Familiar API Posix
  - Grid application standards

Operating System





Application

Set of integrated services (user account, process, file, memory segment, sockets, access rights)

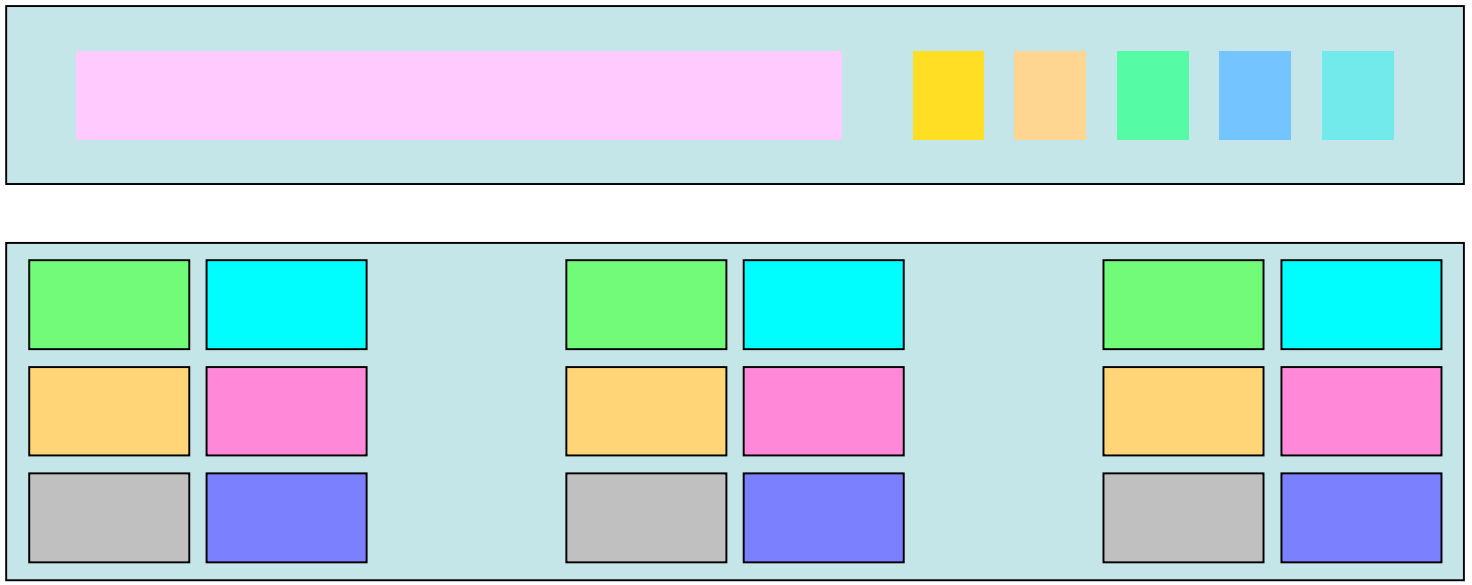
Operating System


Single computer

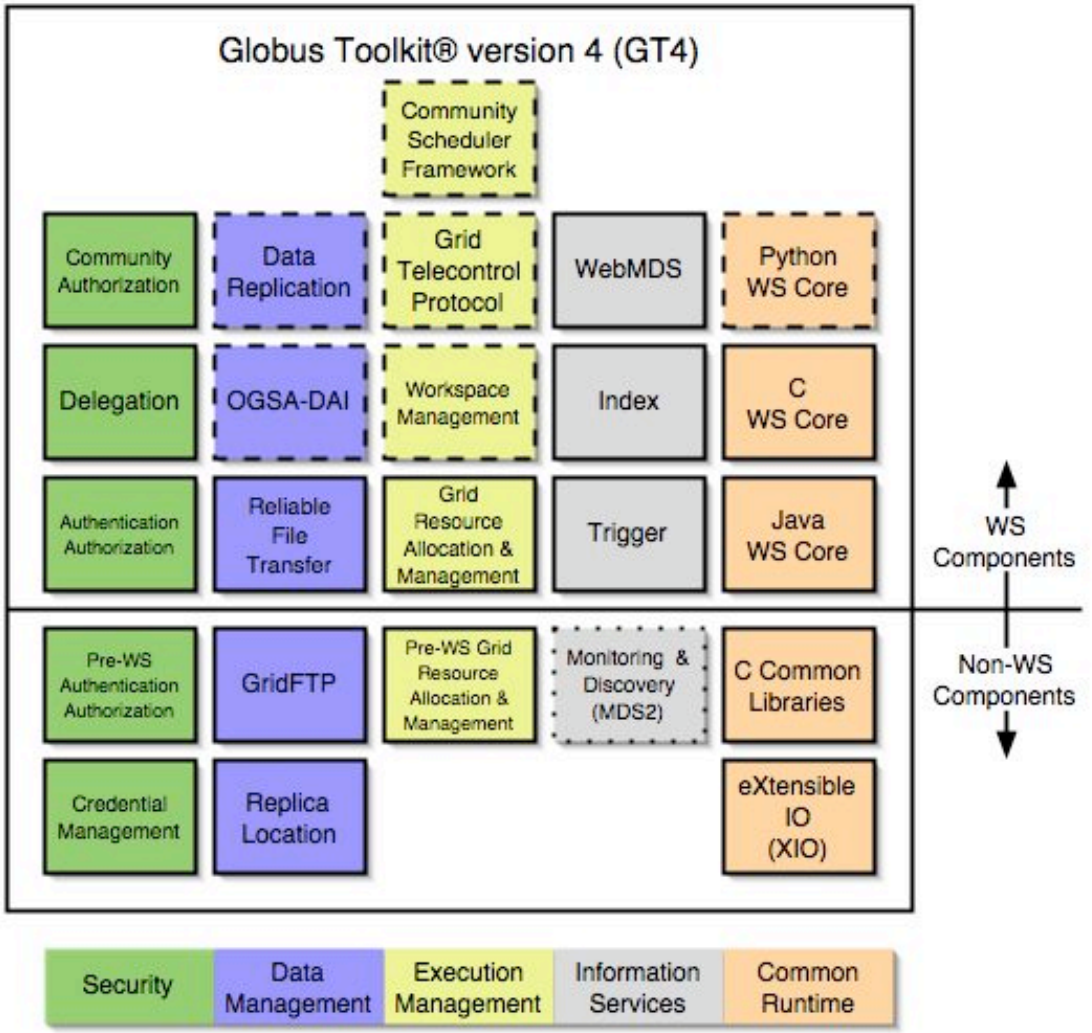
Hardware

Why a Grid Operating System?

Middleware Approach

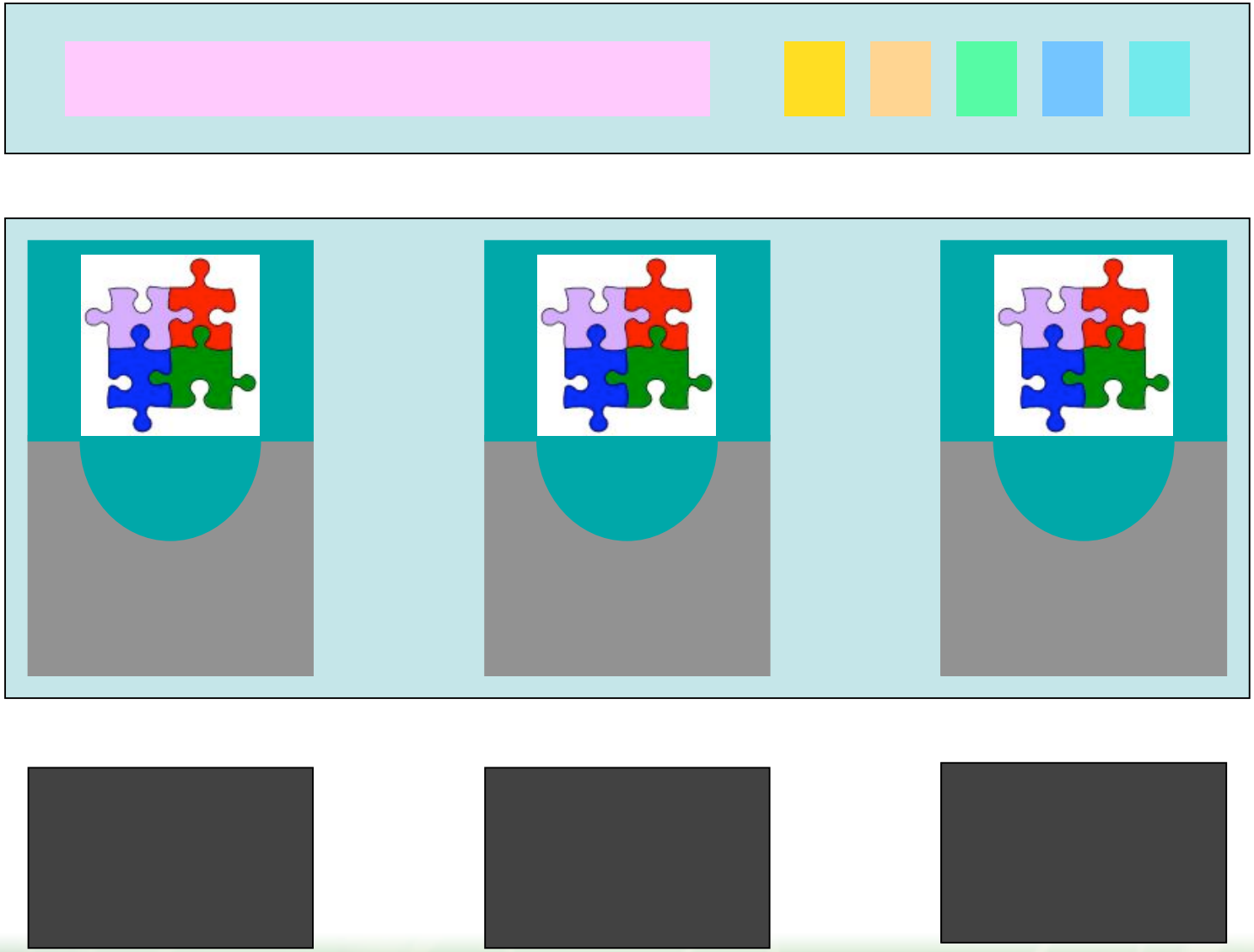


Example: Globus Toolkit



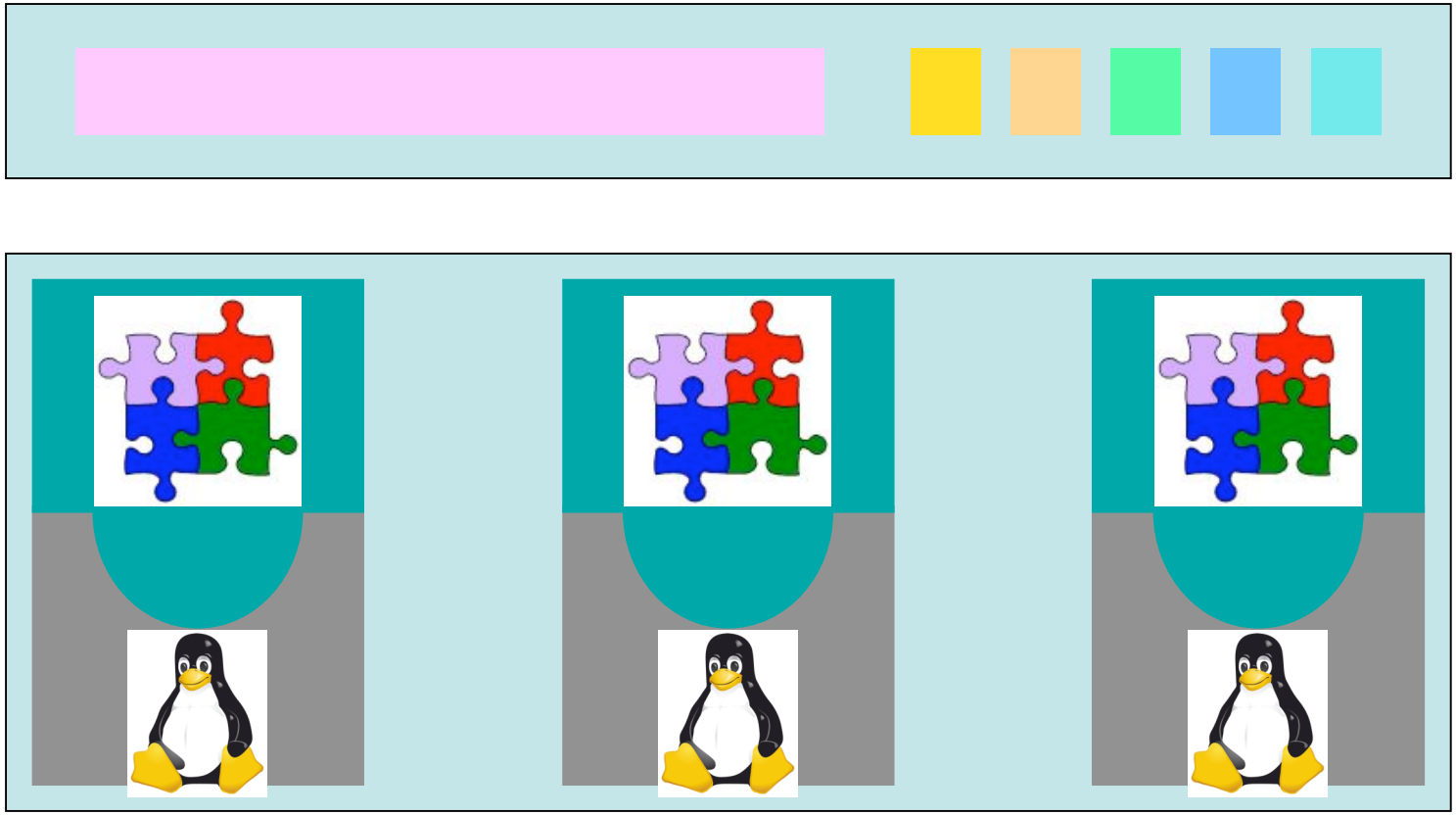
Grid Operating System

Grid OS



XtreemOS
A Grid OS based on Linux with Native VO Support

XtreemOS



Scalability

- **Scale**
  - Thousands of nodes in thousands sites in a wide area infrastructure
  - Thousands of users

- **Consequences of scale**
  - Heterogeneity
    - Node hardware & software configuration
    - Network performance
  - Multiple administrative domains
  - High churn of nodes

XtreemOS Service Scalability

- **Scalability with the number of entities & their geographical distribution**
  - Avoid contention points & save network bandwidth (performance)
  - Run over multiple administrative domains (security)

- **Adaptation to evolving system composition (dynamicity)**
  - Run with partial vision of the system
  - Self-managed services
    - Transparent service migration
  - Critical services highly available
    - No single point of failure

Transparency
User's Point of View

- **Bring the Grid to standard Linux users**
  - **Feeling to work with a Linux machine**
    - Standard way of launching applications
    - `ps` command to check status of own jobs
  - **No limit on the kind of applications supported**
    - Interactive applications
  - **Grid-aware user sessions**
    - **Grid-aware shell** taking care of Grid related issues
  - **VO can be built to isolate or share resources**
    - Parameter defined by VO administrator

Transparency
Application & Application Developer's Point of View

- **Make Grid executions transparent**
  - Hierarchy of jobs in the same way as Unix process hierarchy
  - Same system calls: wait for a job, send signals to a job
  - Processes in a job treated as threads in a Unix process
- **Files stored in XtreemFS Grid file system**
  - Posix interface and semantics to access files regardless of their location
- **Transparent fault tolerance to applications**
- **Clusters transparent to applications**
  - Single System Image

Grid Operating System

A **comprehensive** set of **cooperating** **system services**

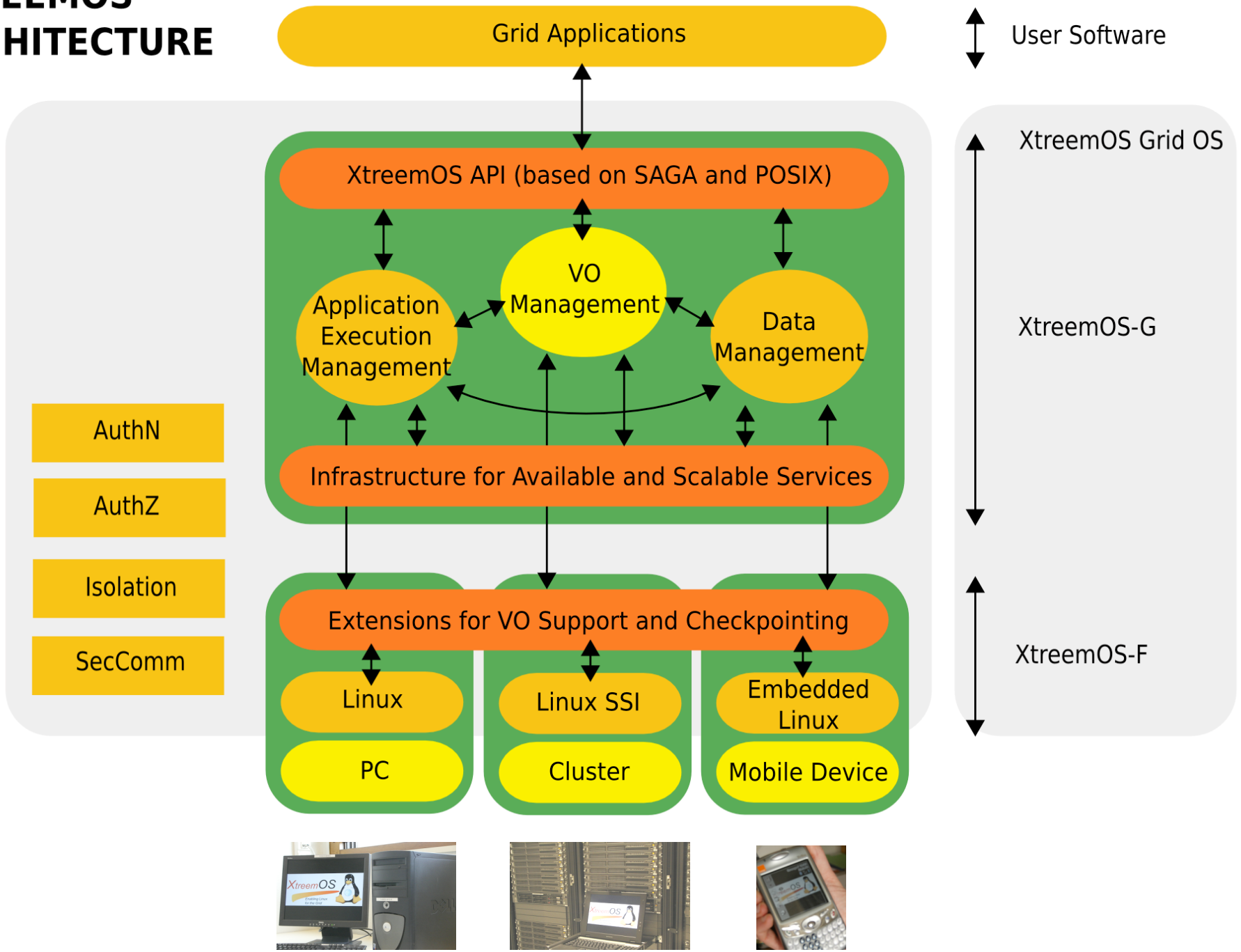
providing a **stable** interface

for a **wide-area** **dynamic distributed infrastructure**

composed of **heterogeneous resources**

spanning **multiple administrative domains**

XTREEMOS ARCHITECTURE



Outline

- XtreemOS project
- XtreemOS vision
- VO & security management
- *Demo VO & security*
- Job Management
- Data management
- *Demo user session*
- Infrastructure for scalable & highly available services
  XtreemOS API
- XtreemOS flavours
- Concluding remarks

XtreemOS

Enabling Linux for the Grid

VO

Organization 1

Site 1

VO A

Organization 3

Site 3

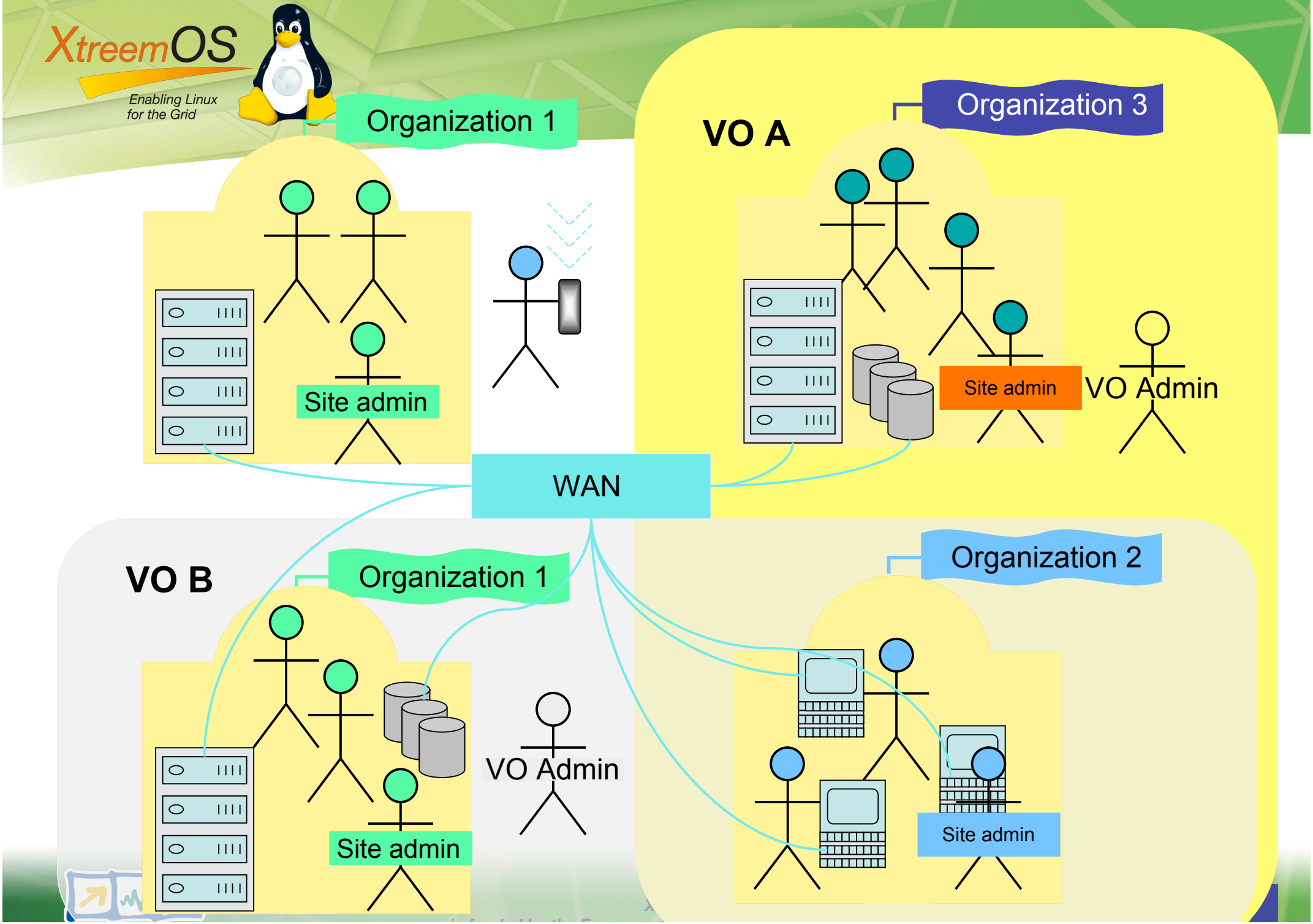
VO B

Organization 1

Site 2

Organization 2

Site 4


XtreemOS
Enabling Linux for the Grid
Organization 1
Site admin
VO A
Organization 3
Site admin
VO Admin
WAN
VO B
Organization 1
VO Admin
Site admin
Organization 2
Site admin

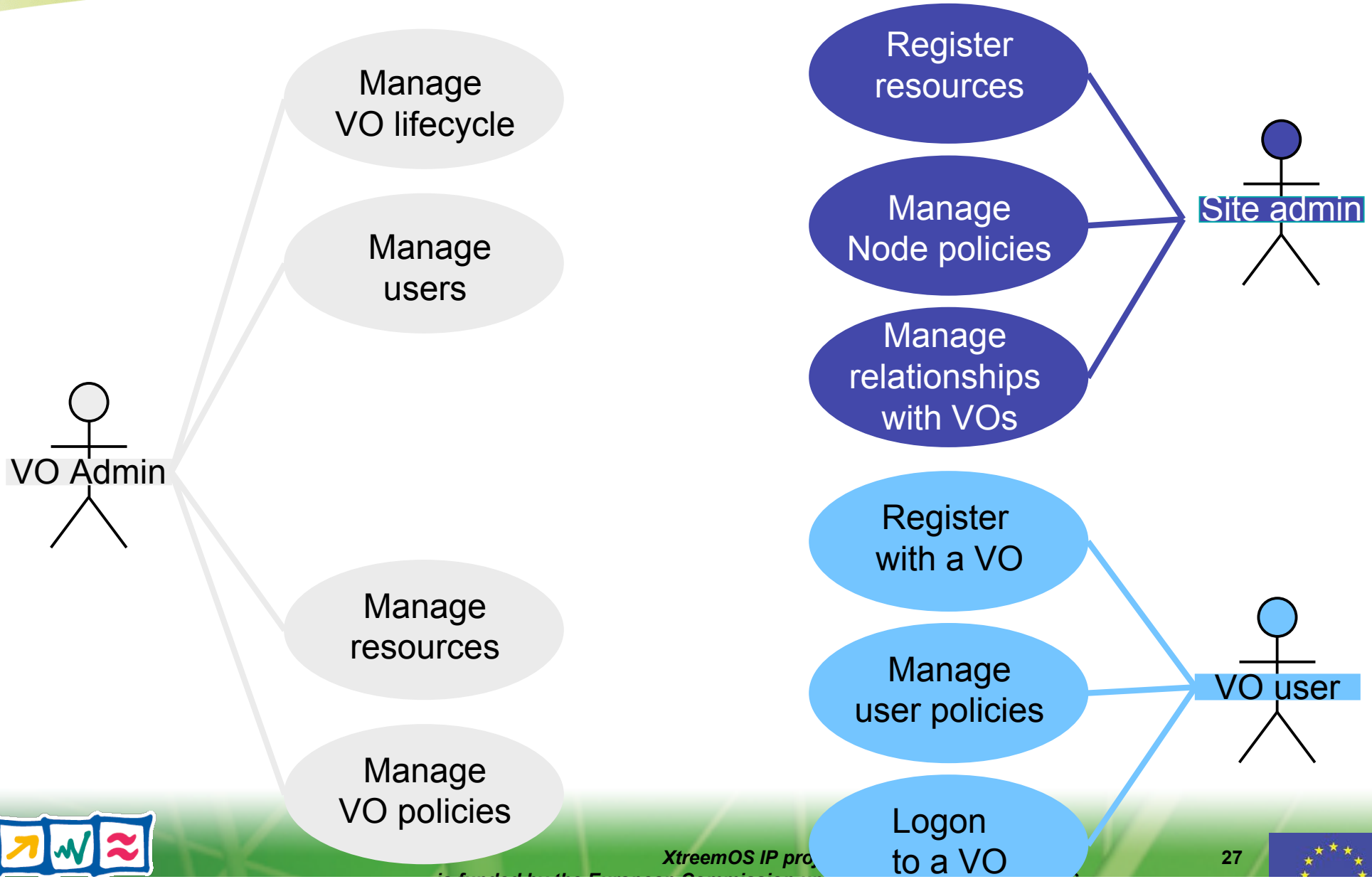
Requirements

- **Secure VO management & application execution**
  - Grid user and service mutual authentication
  - Confidentiality and integrity of stored and communicated data
  - Authorized access to data, services, resources
  - Isolation
  - Accountability of data access and service execution

VO Management & Security

- **Scalability of management of dynamic VOs**
  - **VO-centric security architecture**
  - Dynamic mapping between Grid VO users & Linux entities with no modification to Linux kernel
    - No centralized Grid wide data base, no grid map file needed
- **Flexible administration of VOs**
  - Multiple VO models supported (on-going research)
  - Hierarchical policy management (VO, resource, user)
  - Accountability of data access and service execution (on-going)
- **Interoperability with third party security infrastructures**
  - Kerberos, LDAP, Shibboleth…
- **Single-Sign-On**

VO-related Use Cases



Node Level VO Support

- **Policies specified by a VO finally checked & ensured at resource nodes by the local instance of the OS**
  - Standard Linux unaware of VOs
  - Isolation & access control mainly rely on user accounts, process id, file permission bits
- **What is needed for Linux OS to be able to enforce VO policies**
  - OS kernel should deal with VO & VO users identities
  - Identity information should be exploited in standard access control mechanisms
  - Linux OS should supply identity information to Grid level services (XtreemFS, AEM)
- **NO modification of Linux kernel**
  - Mapping of VO level identities & policies into local ones fully recognized by Linux

System-Level VO Support

- **VO-customizable, dynamic mapping of Grid users onto local accounts**
  - Integration of Grid user management into Linux using
    - Pluggable Authentication Modules (PAM)
      - Multiple low level authentication technologies into a common high level API
    - Name Service Switch (NSS)
- **Interfacing with the Grid authentication services**
  - Development of PAM modules to accommodate multiple VO models
    - Authentication, authorization, session management
- **User space credential translation**
  - NS-Switch
- **Access control & logging**
  - Caching of authentication data related to a process within the kernel

Purposes of VO and Security Services

- **VO services managing**
  - VO lifecycle (VO creation, evolution, termination)
  - VO members: users, resources, their membership in VOs
- **Security services managing**
  - Credentials (e.g. identity and attribute credentials)
  - Policies (e.g. VO and node level rules/policies)

Types of VO and Security Services

- **VO and security services are operated in two scopes:**
  - Node scope services (aka. node services): deal with the requests from a single Linux box
  - Global scope services (aka. global services): deal with the requests from multiple Linux boxes
- **All VO and security services can serve more than one VO.**

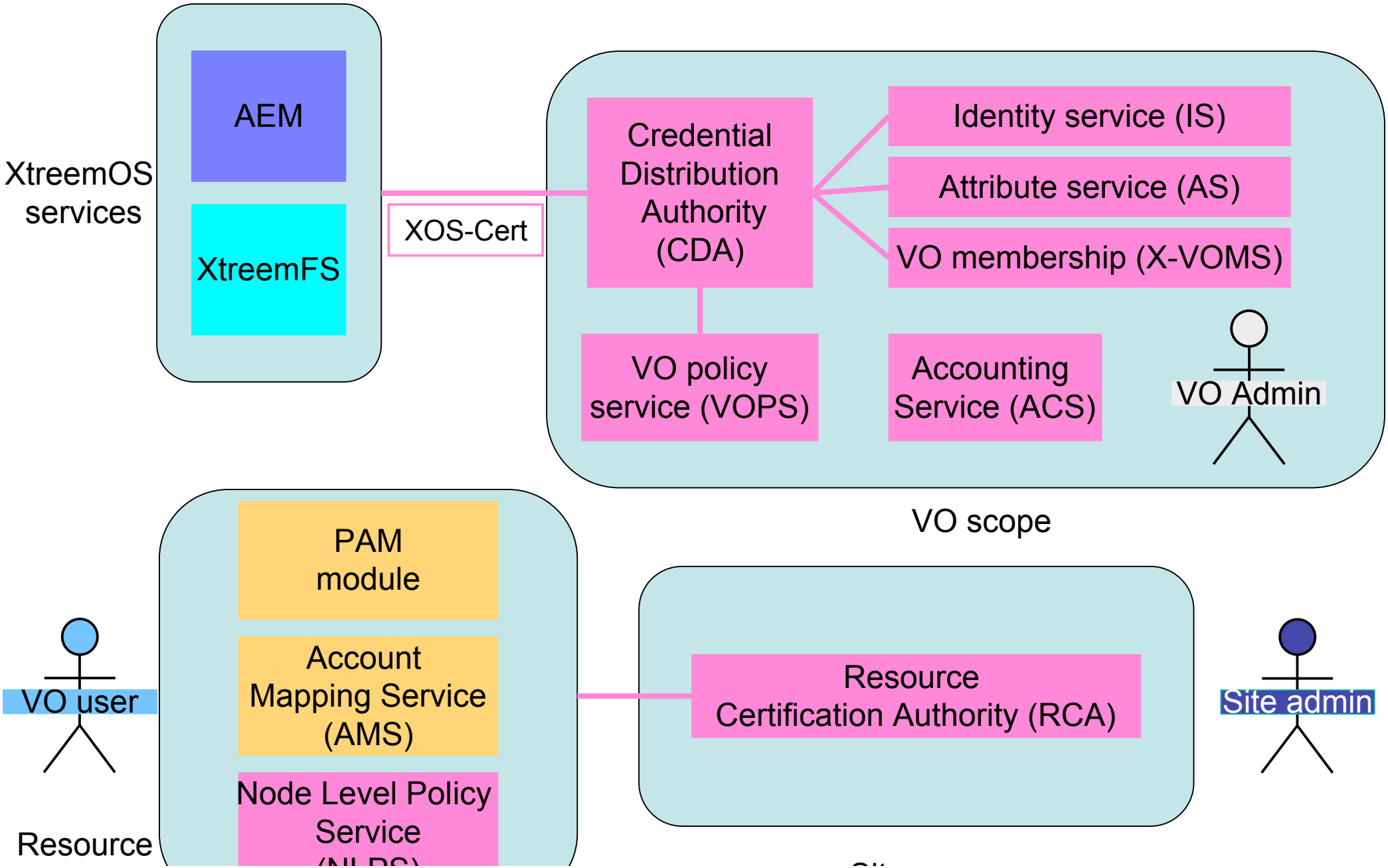
Global Services

- **X-VOMS**
  - VO database management system, containing details of users' VO membership and VO attributes (e.g. VO groups they belong to);
- **CDA: Credential Distribution Authority**
  - provides users with XOS-Certificates
  - contain their public key and VO attributes,
  - used to authenticate user and check their authorisation.
- **RCA: Resource Certification Authority**
  - provides machines with a certified list of the resources (CPU, RAM etc) they can provide for use in AEM resource allocation
- **VOPS: VO Policy Service**
  - allows flexible selection of nodes during AEM resource matching according to VO-defined policies
- **VOLife**
  - a web application providing a convenient front-end to the services above,
  - allowing initial registration of users, creating of VOs and allocation of users to VOs

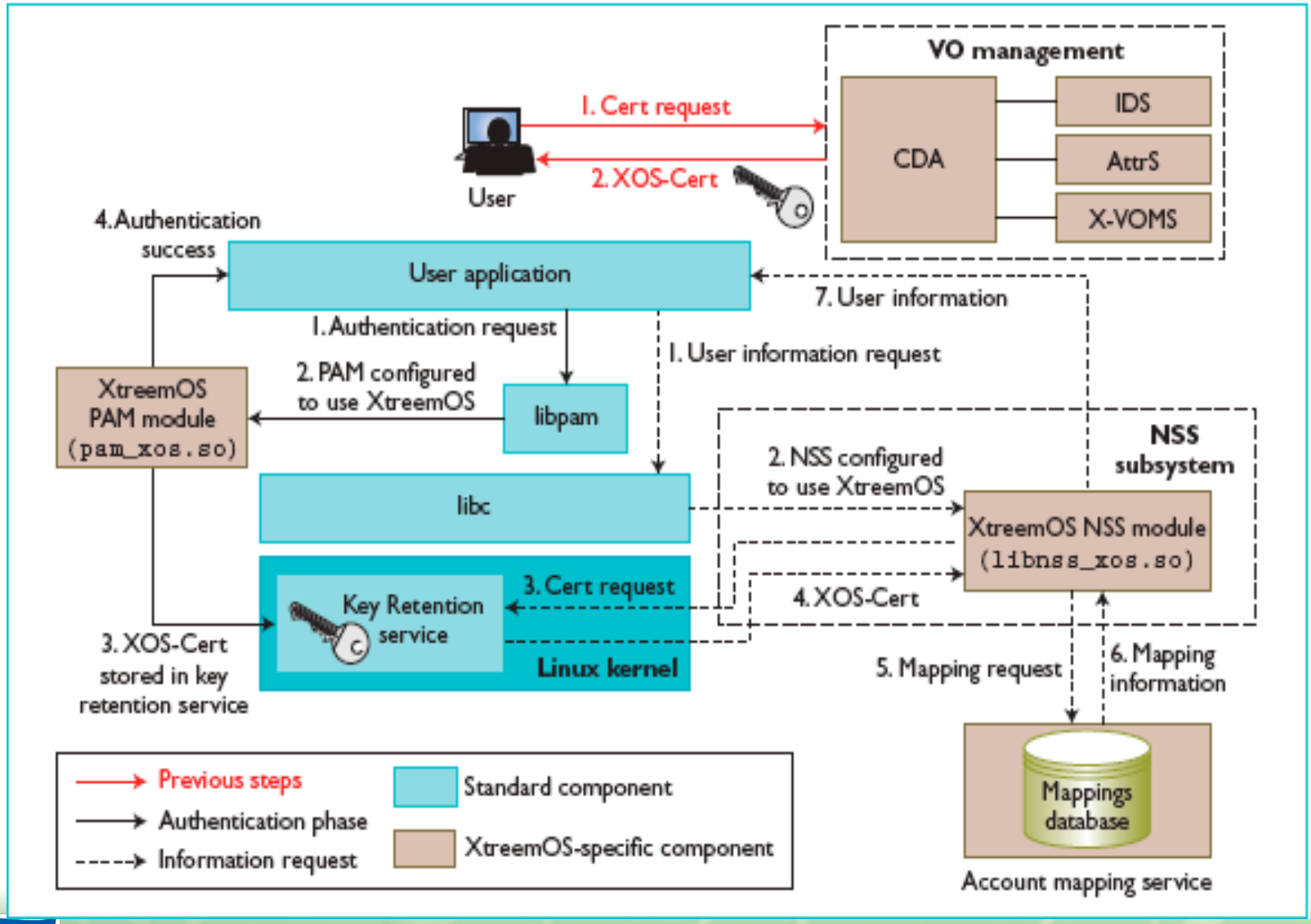
Node Services

- **PAM (Pluggable Authentication Module extensions)/XOS-SSH (XtreemOS Secure Shell)**
  - Certificate-based authentication recognizing VO attributes
- **Name Switch Service (NSS extensions)**
  - Supporting bi-directional mapping between Global User IDentity (GUID) and local user identity (UID)
- **AMS (Account Mapping Service)**
  - Dynamic (and session-based) account creation/deletion
- **KKRS (Kernel Key Retention Service)**
  - Allows key pairs, authentication tokens, user mappings, to be cached in the kernel for use of file systems and other kernel services.
- **NLPS (Node-level Policy Service)**
  - Allows resource owners to specify how their Linux boxes should be exploited

VO-centric Security Architecture



Node Services



Key Contributions

- **Maximum transparency**
  - Grid unaware applications & tools can be used without being modified or recompiled
- **Integration of Grid level authentication with system level authentication**
  - Creation of dynamic on-the-fly mappings for Grid users in a clean & scalable way
  - No centralized Grid wide data base
- **Grid user mappings invisible to local users**
- **VO are easier to setup and manage**
  - No grid map file needed
  - User management does not necessitate any resource reconfiguration

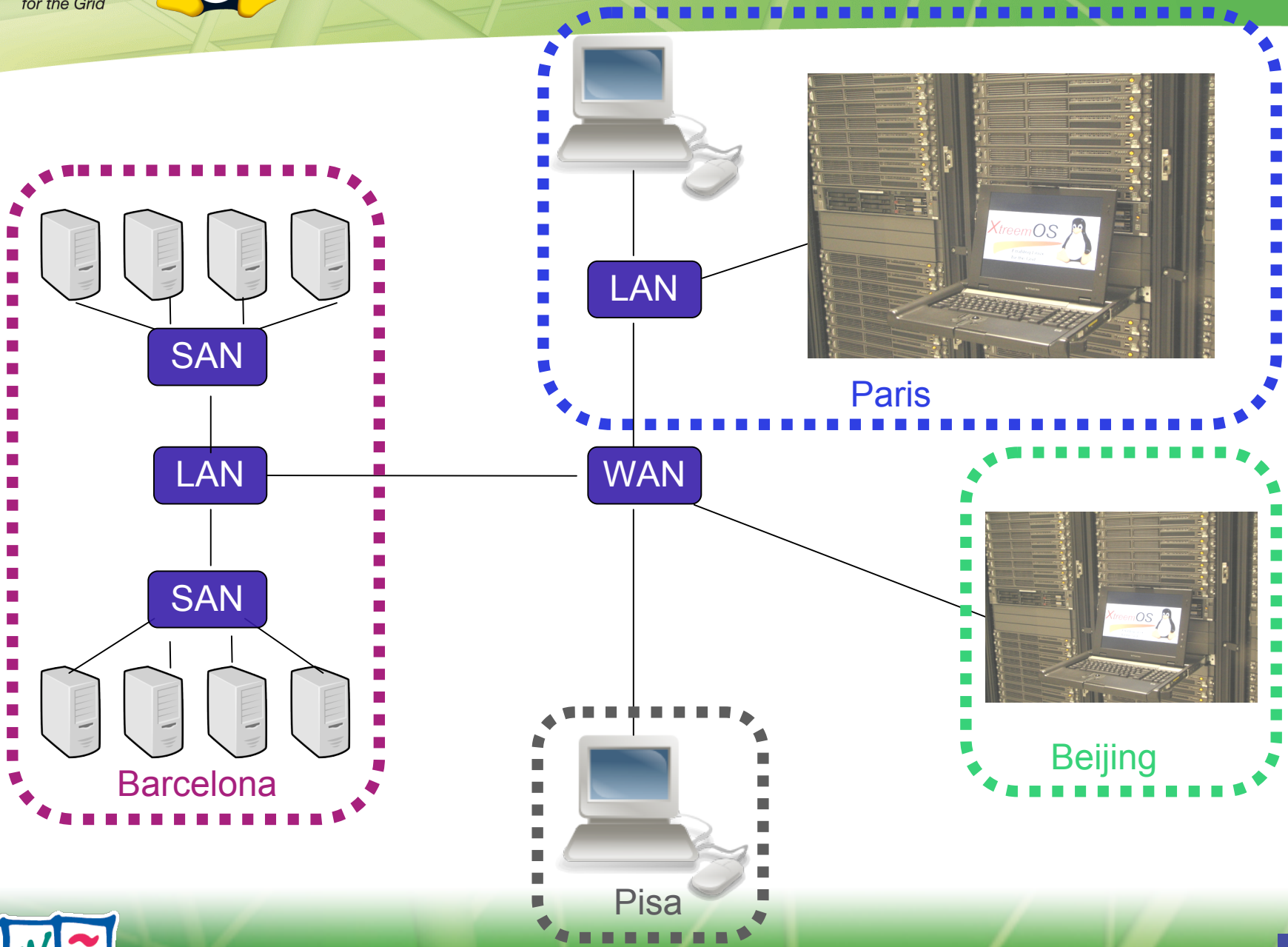
Outline

- XtreemOS project
- XtreemOS vision
- VO & security management
- *Demo VO & Security*
- Job Management
- Data management
- *Demo user session*
- Infrastructure for scalable & highly available services
  XtreemOS API
- XtreemOS flavours
- Concluding remarks

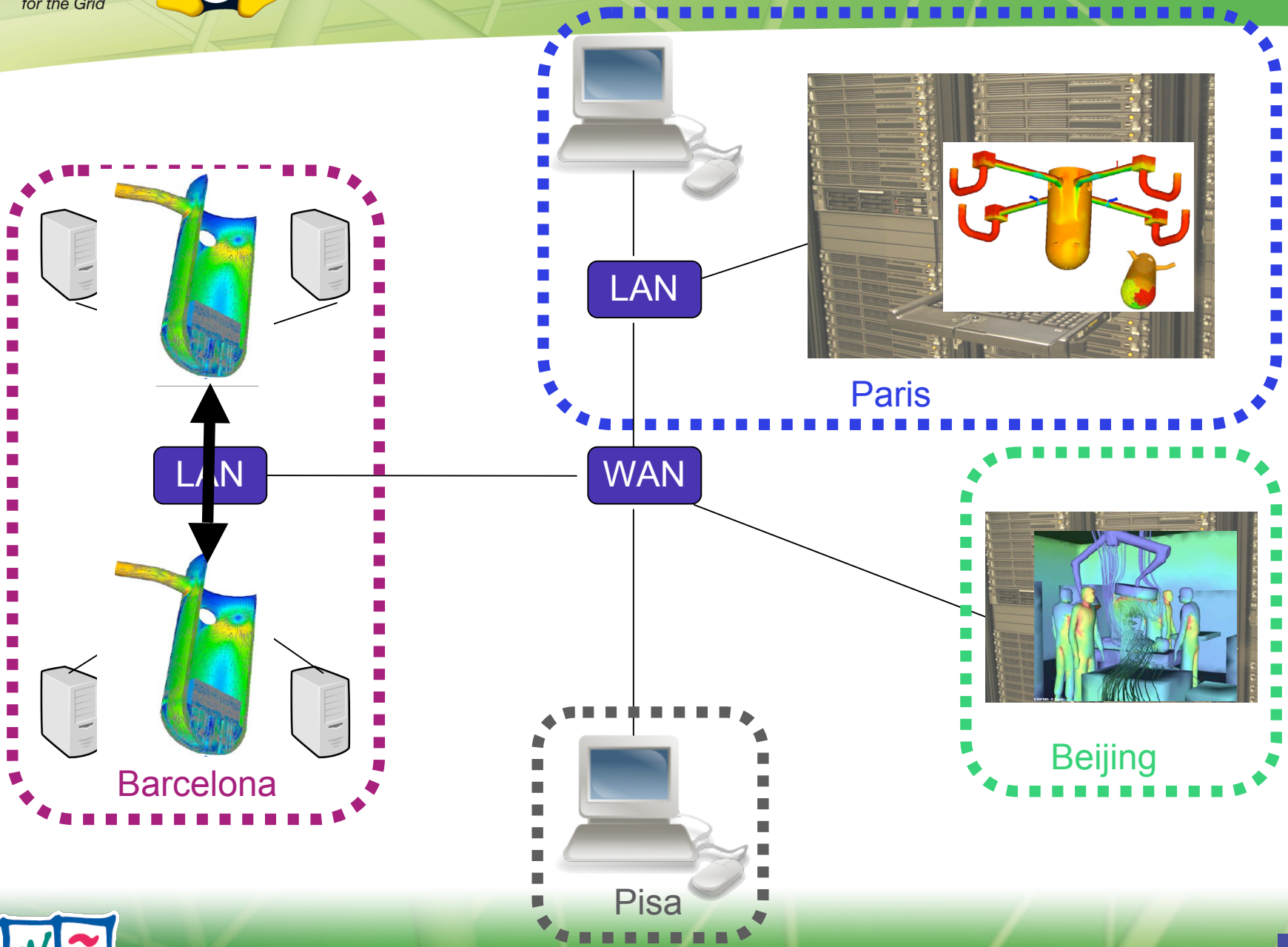
Outline

- XtreemOS project
- XtreemOS vision
- VO & security management
- *Demo VO & security*
- Job management
- Data management
- *Demo user session*
- Infrastructure for scalable & highly available services
  XtreemOS API
- XtreemOS flavours
- Concluding remarks

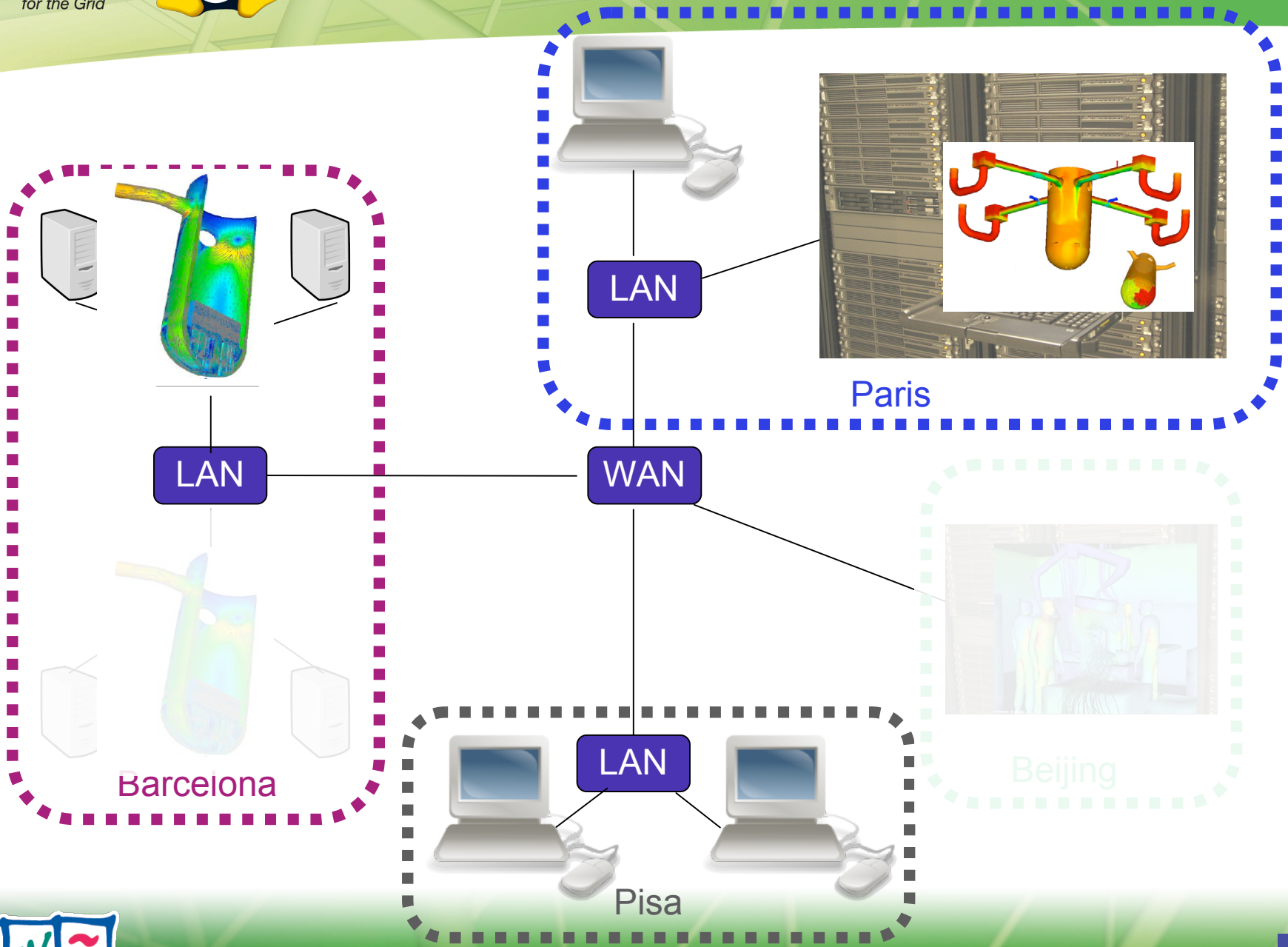
Grid Environment



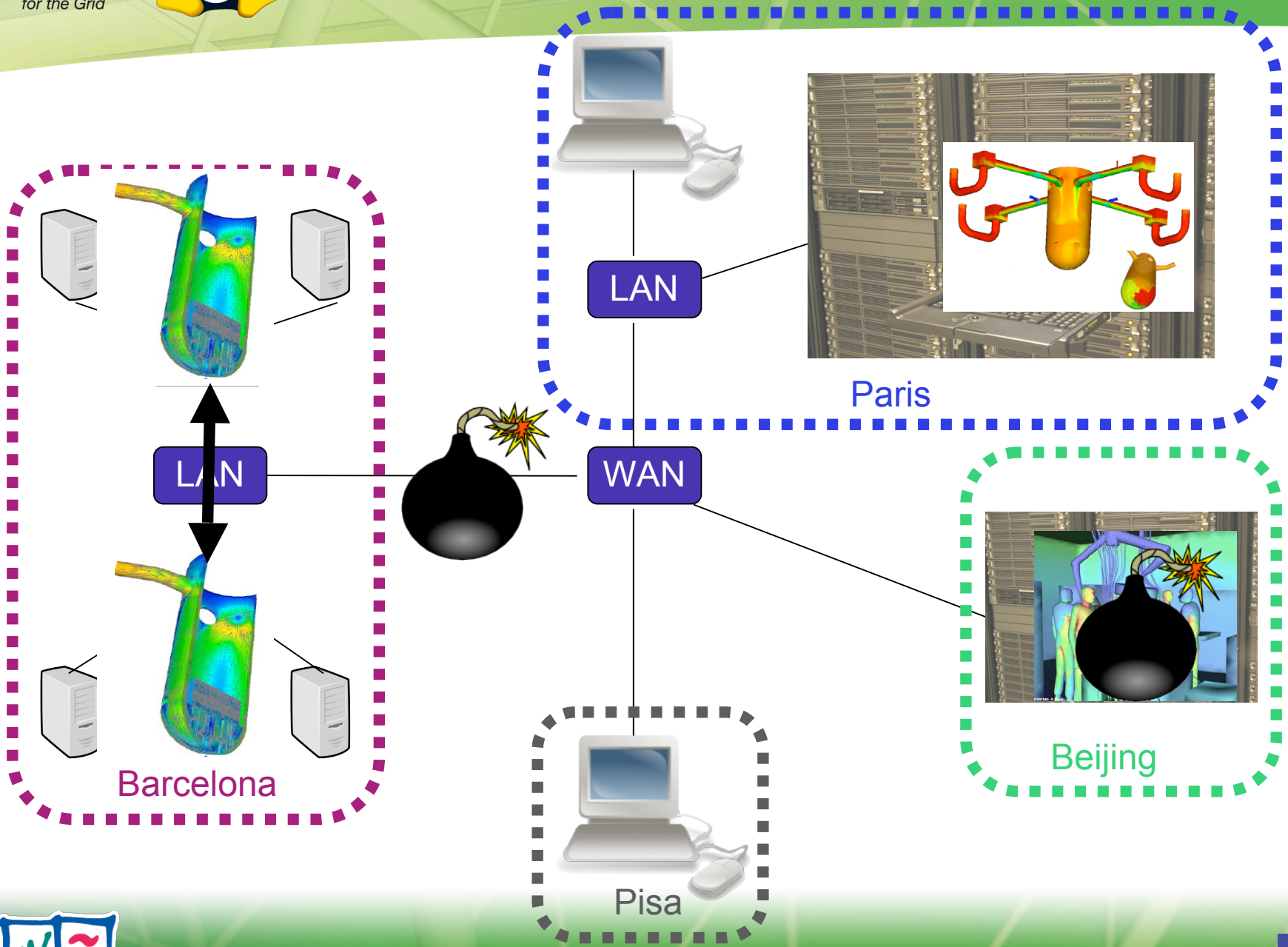
Application Execution



Node volatility

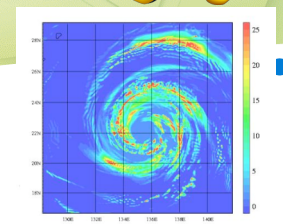


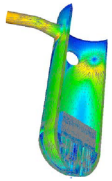
Failures

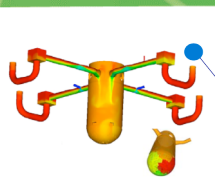


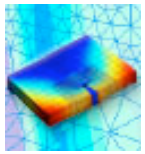
XtreemOS

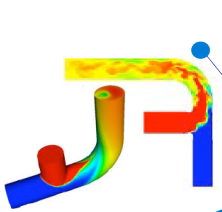
Enabling Linux for the Grid

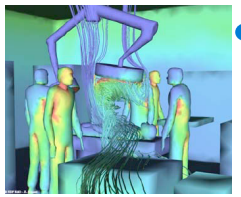


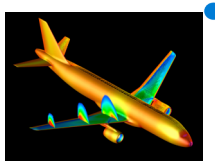


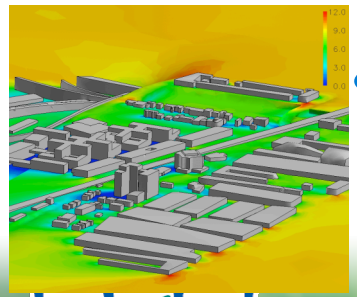


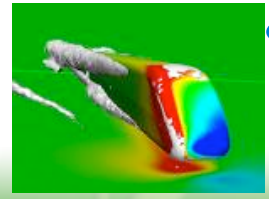


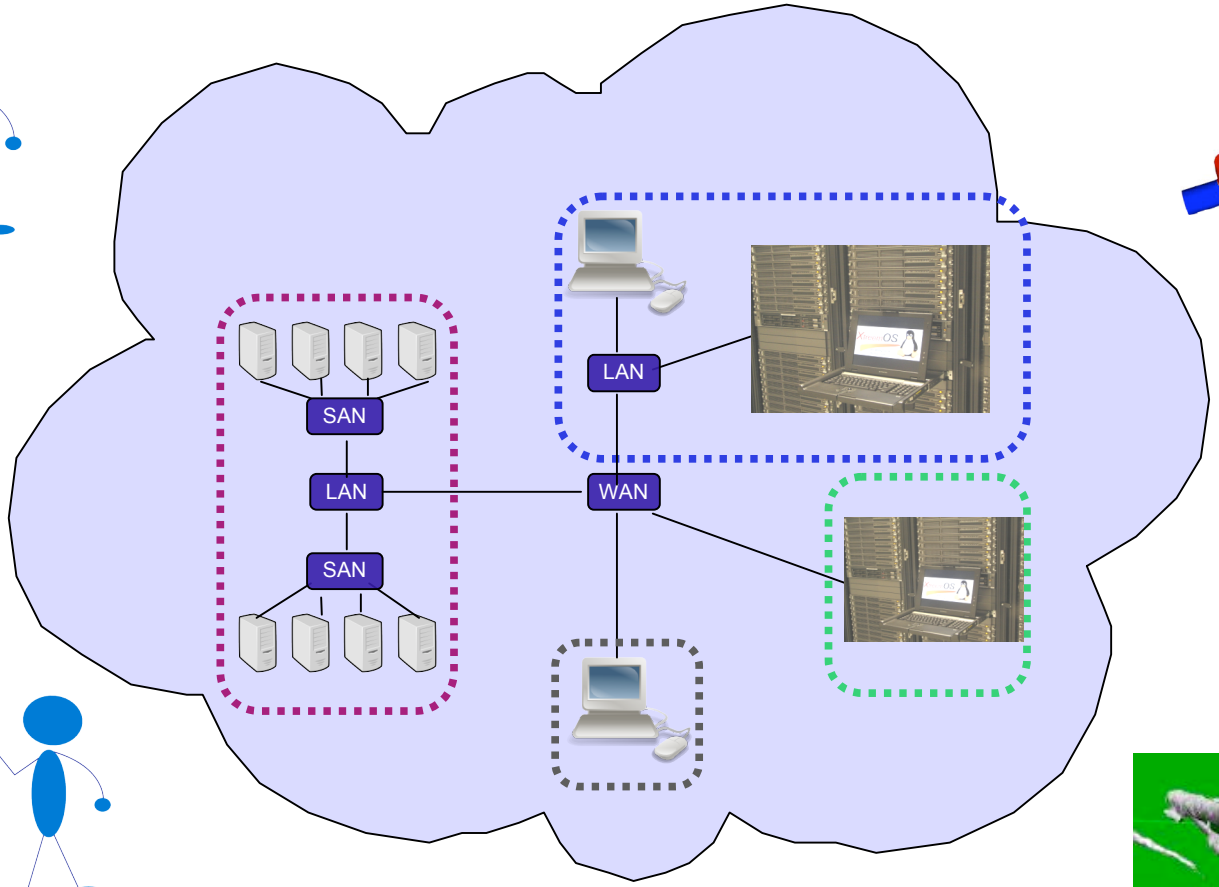


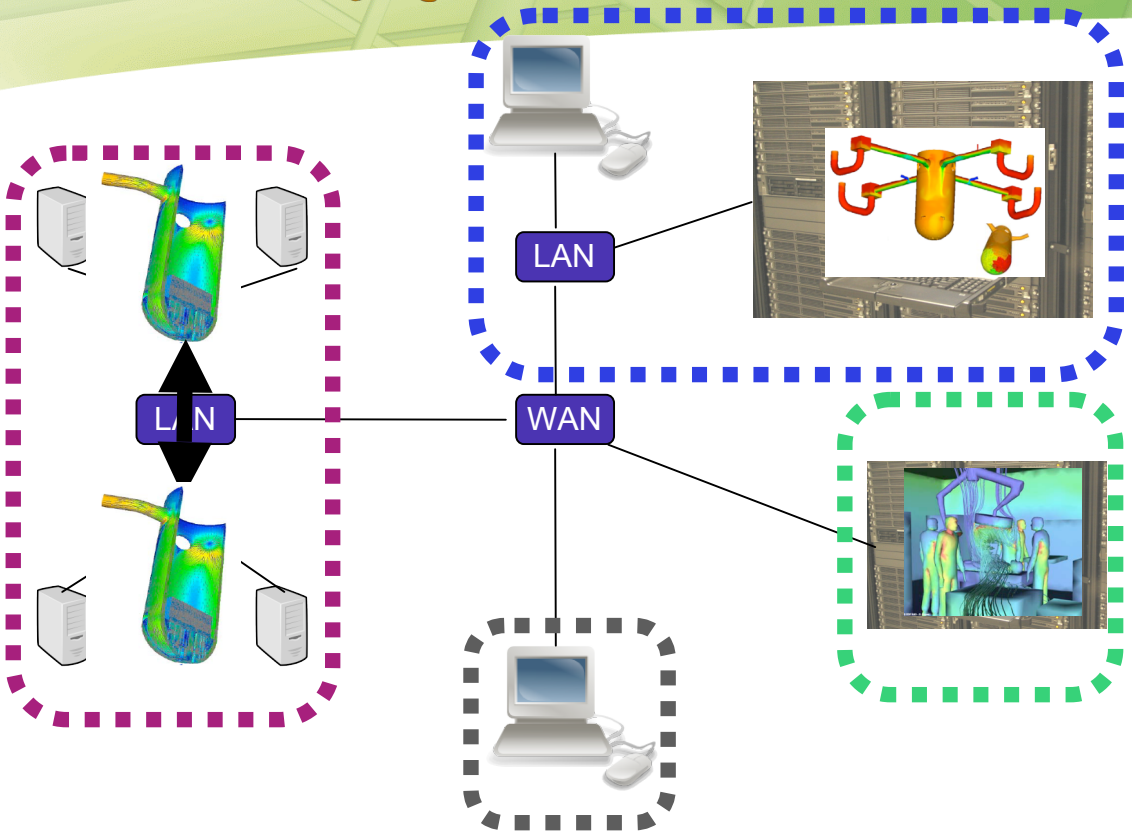




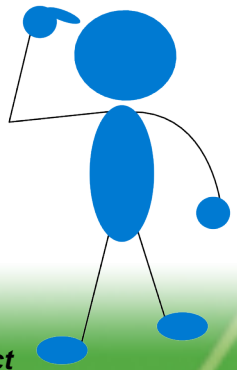


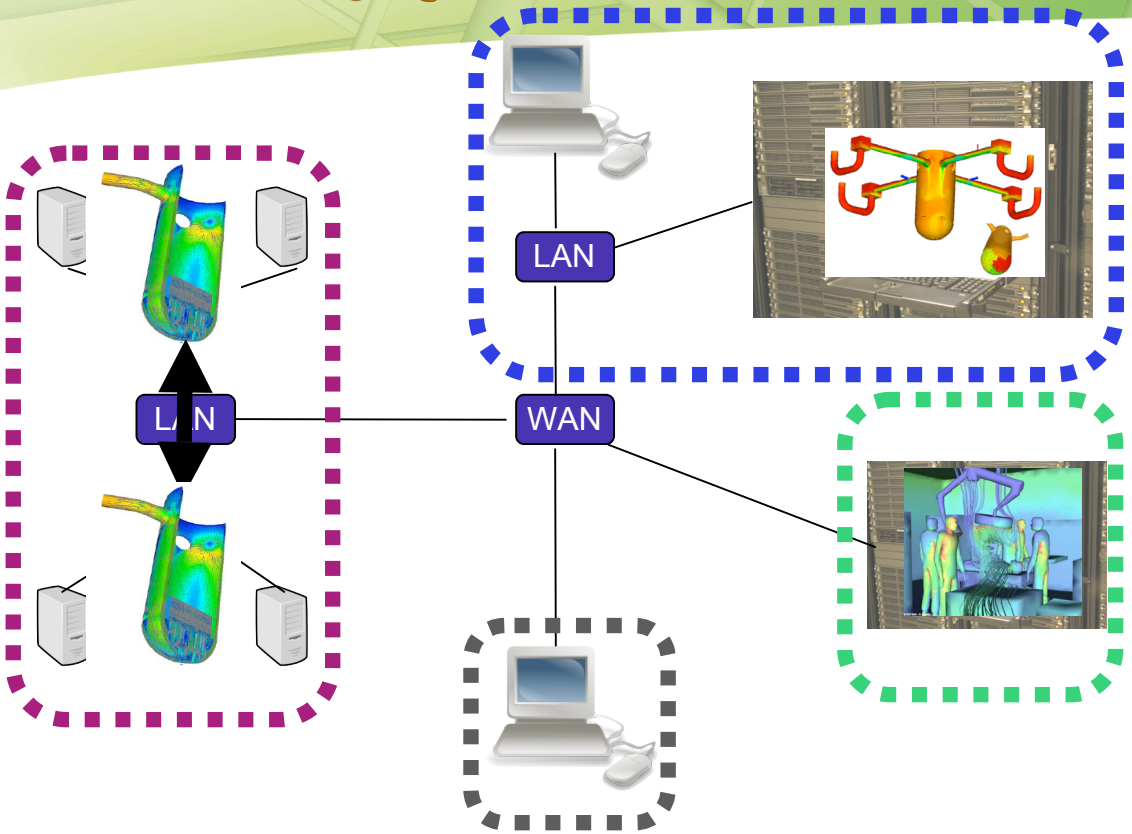




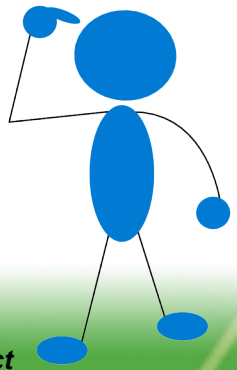


How to submit my application to the grid ?

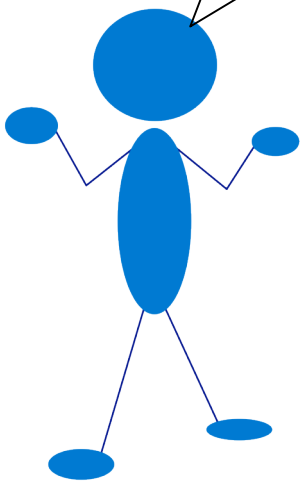


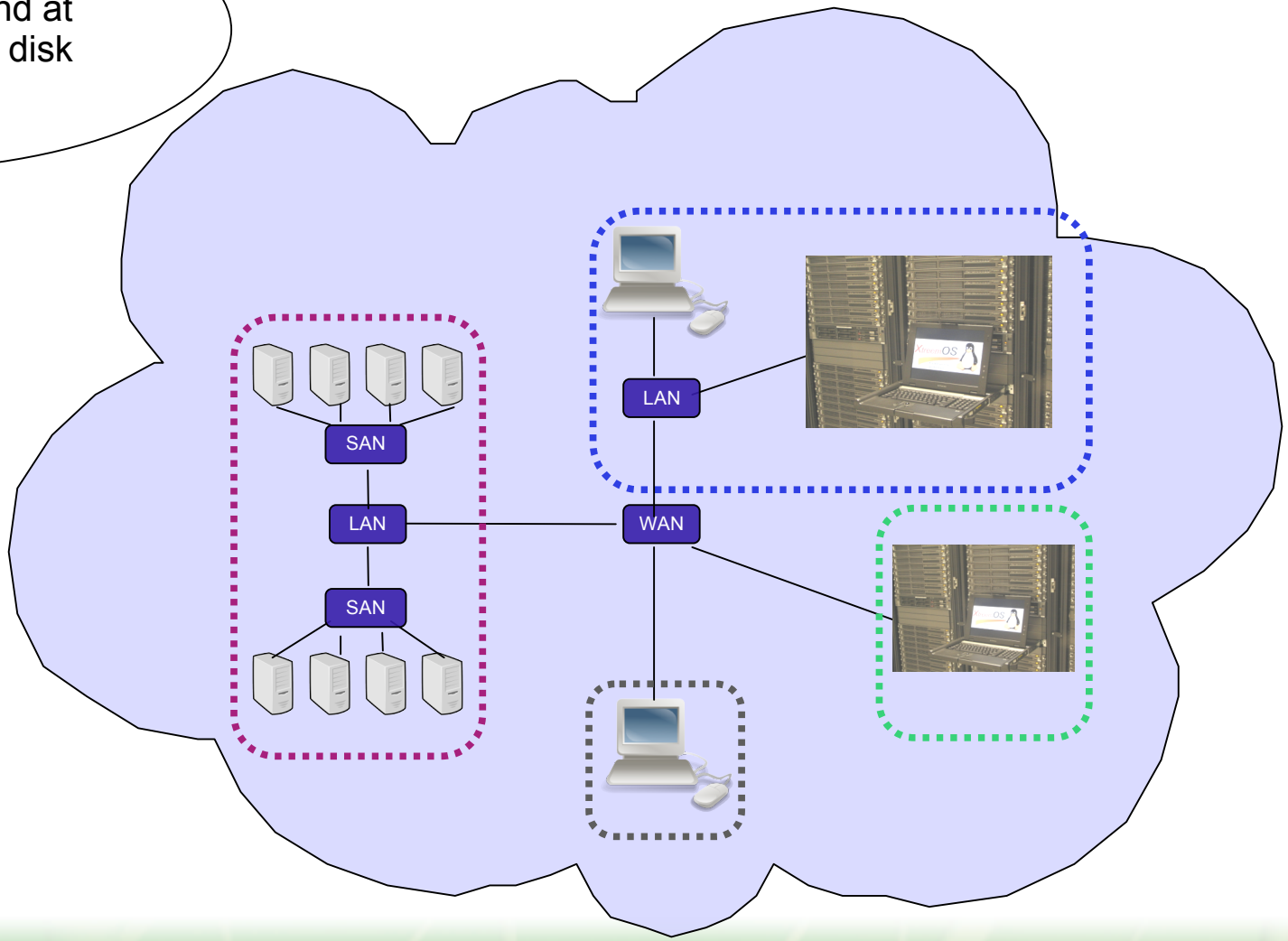


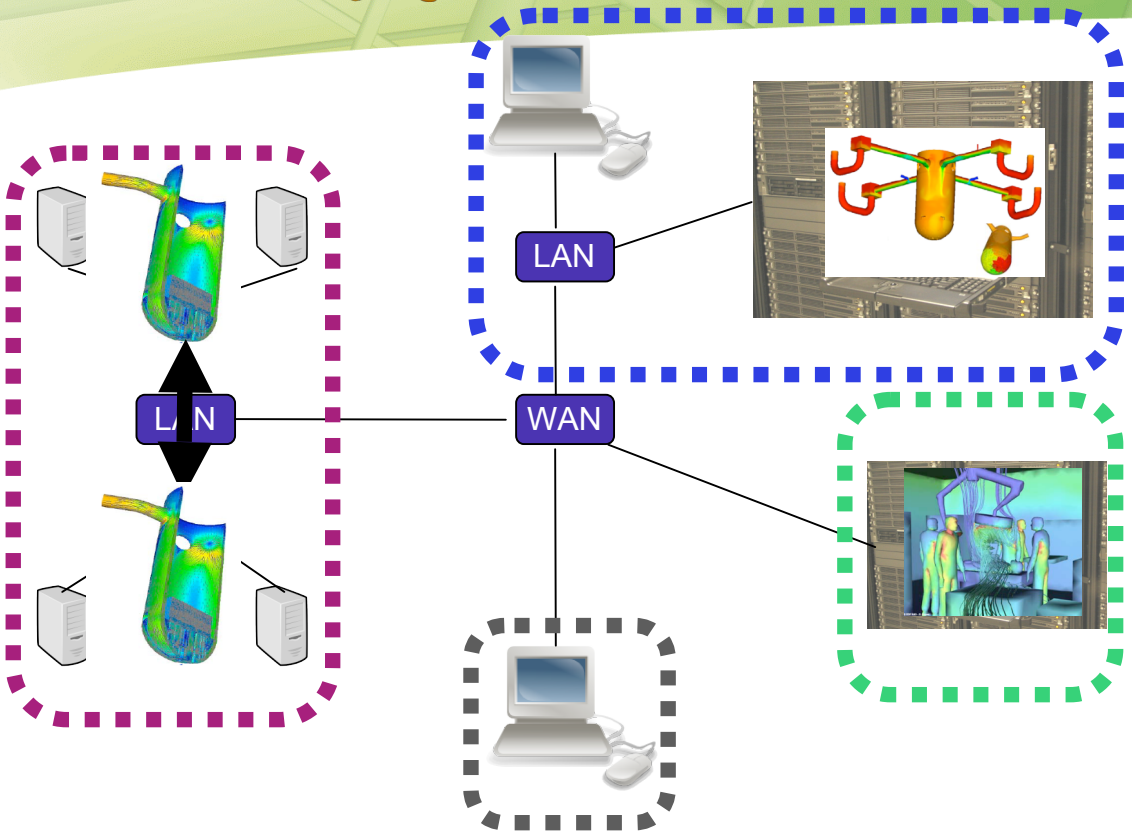
There are different kinds of resources in the grid, how can I find the most suitable ones ?



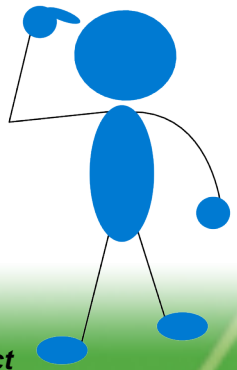
How to find 5 nodes Opteron, with 4 Go of RAM and at least 200 Mo of free disk space?

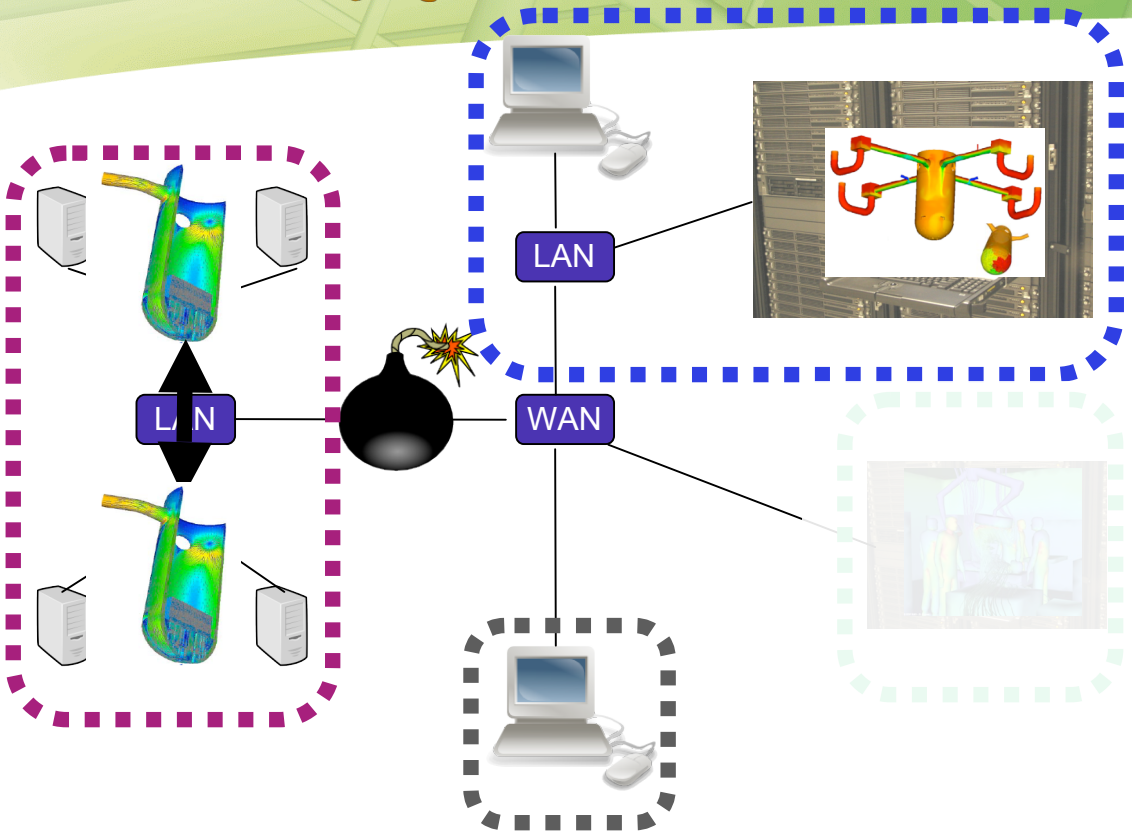




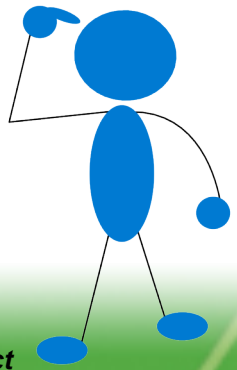


Will my application be efficiently executed if other people use the grid at the same time?





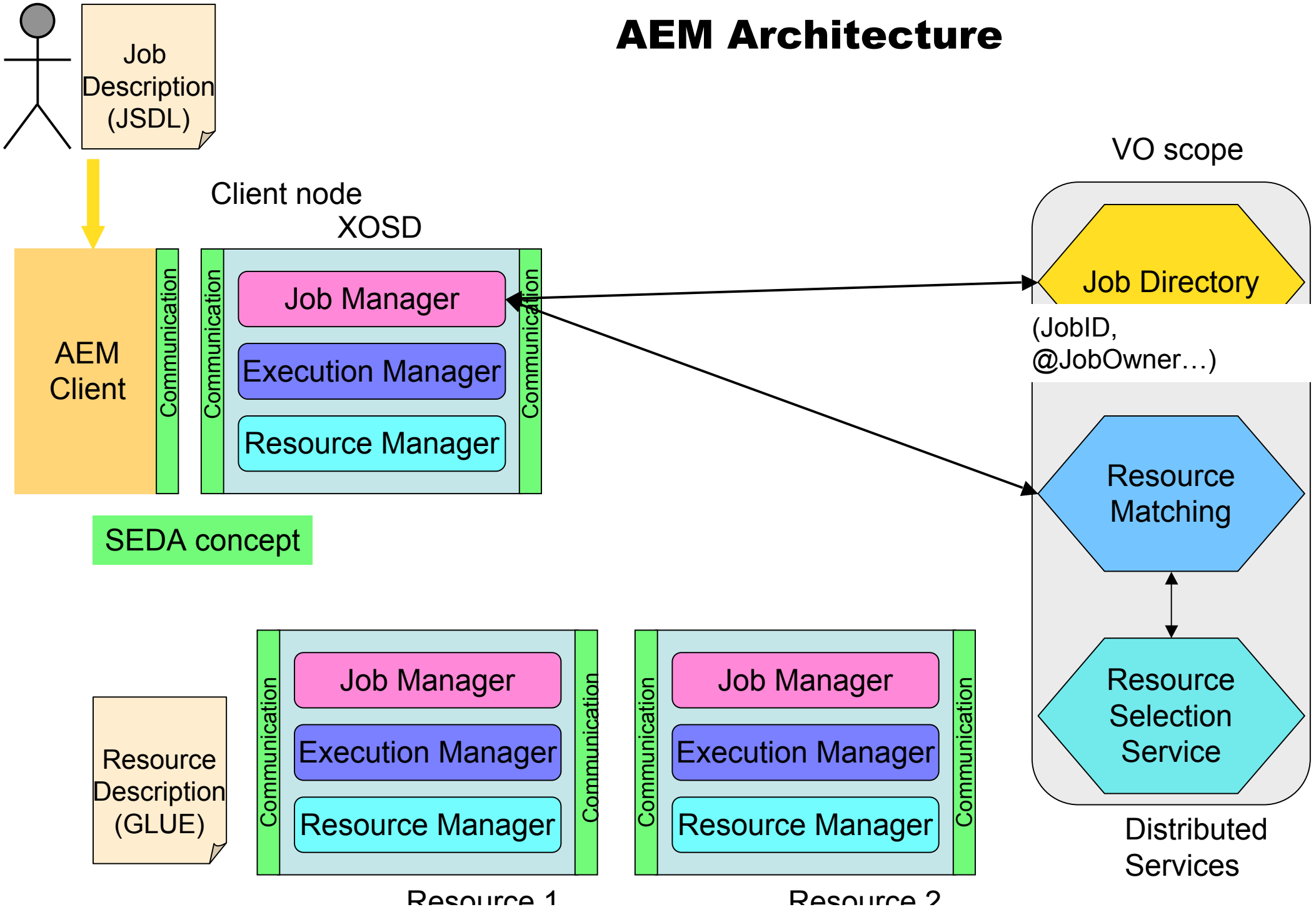
Will my application be executed despite the grid dynamic behaviour?



Application Execution Management

- **Objectives**
  - Start, monitor, control applications
  - Discover, select, allocate resources to applications
- **Features**
  - No assumption on local node RMS
    - AEM can be used without any batch system
  - Job “self-scheduling”
    - No global job scheduler
  - Resource discovery based on overlay networks
  - Unix-like job control
  - Accurate and flexible monitoring of job execution
  - Checkpointing service for grid jobs

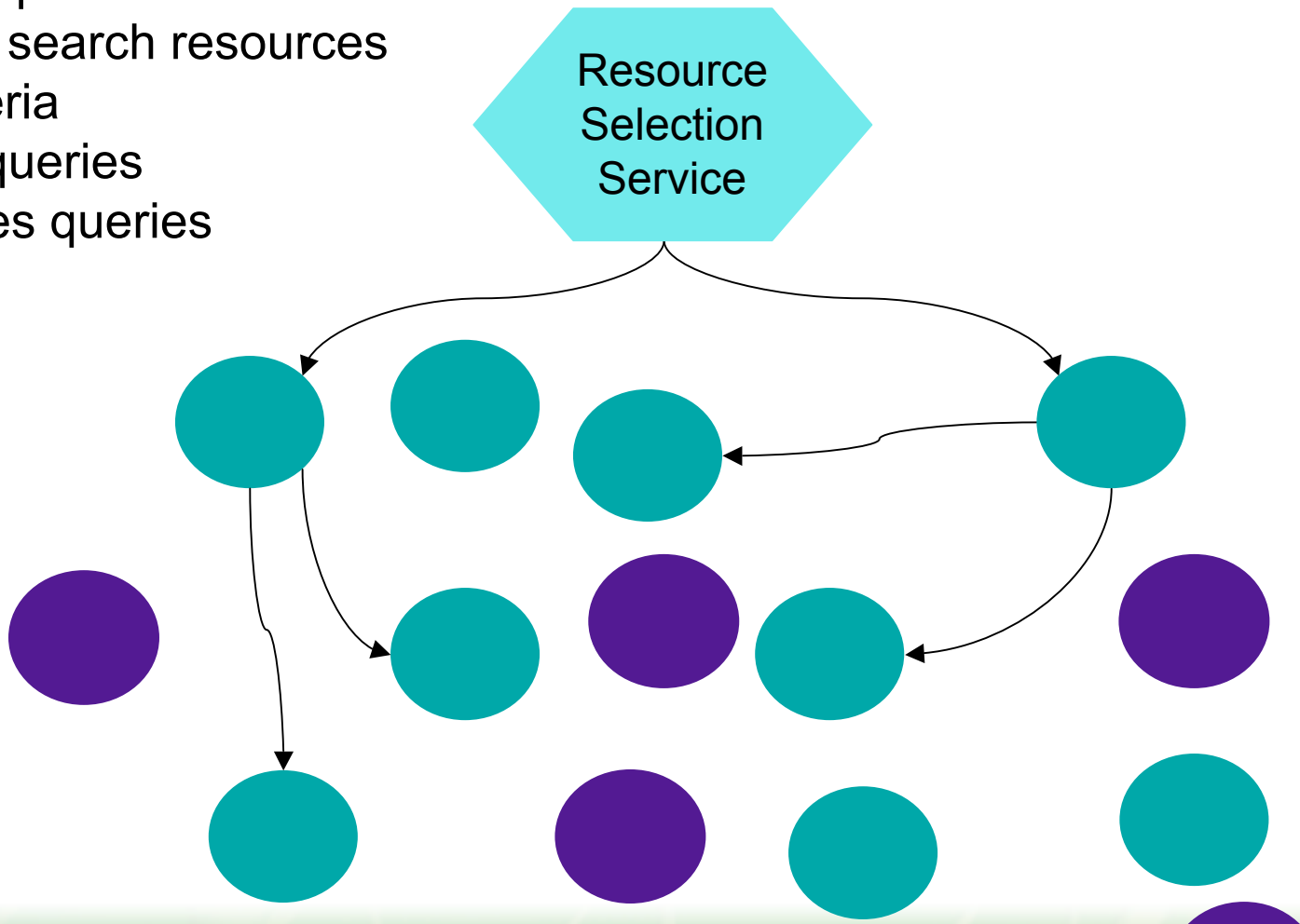
AEM Architecture



Resource Discovery

Resource discovery protocol on overlay networks to search resources matching static criteria

- multi-criteria queries
- range of values queries



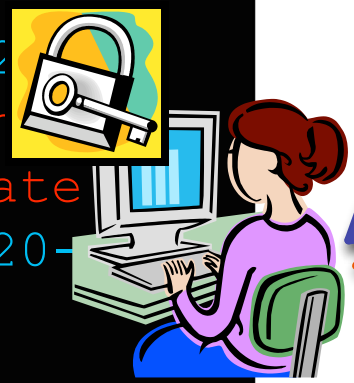
Job Control

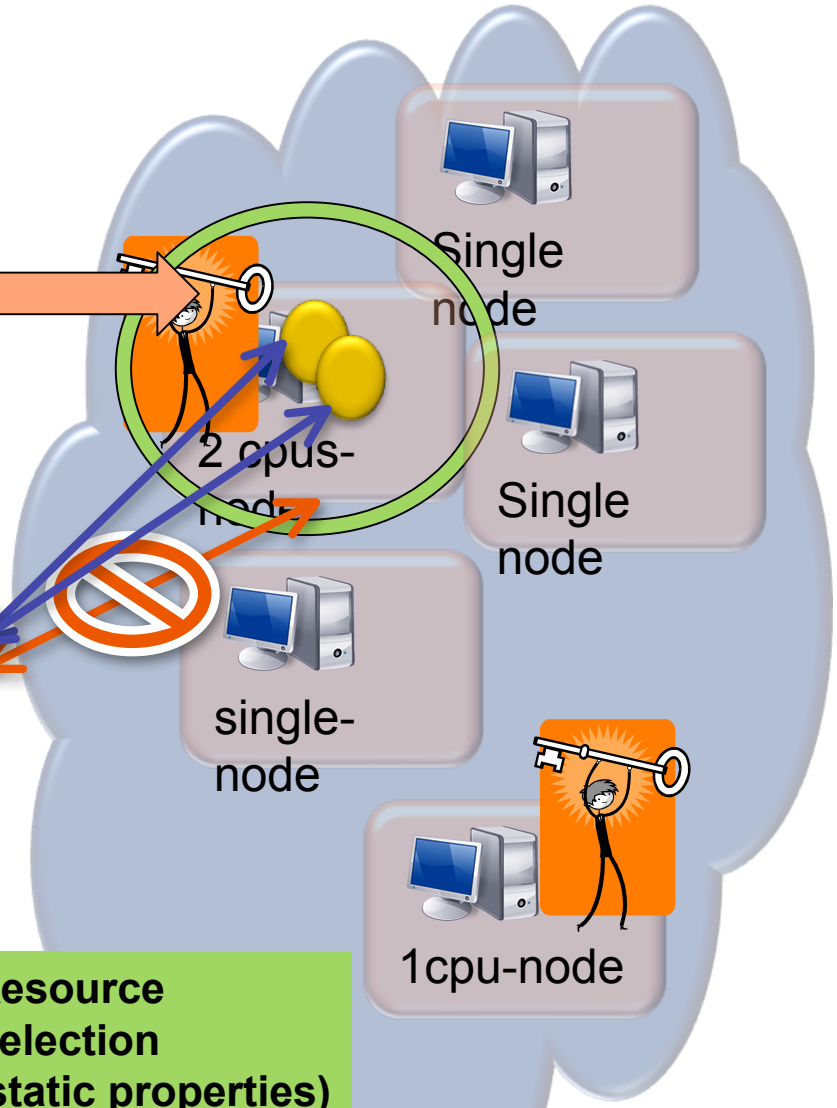
- **xps**
- **Send signals to a job**
  - To all its processes
  - Eg. kill
- **Graceful job termination**
  - Exit and wait job

Job Control: xps

```
➢./runclient localhost 9090
mykey.pem mycert.pem
➢xsub kmines.jsdl -c
mycert.pem
Job submitted succesfully: 53c92b29-
9a58-4620-b7ab-2dc6d363de35
>xps -j 53c92b29-9a58-462
b7ab-2dc6d363de35 -c wror
XOSERROR::Wrong certificate
>xps -j 53c92b29-9a58-4620-
b7ab-2dc6d363de35 -c
mycert.pem
JOB ID - COMMAND - JOB STATE * RESOURCE
ADDR:PORT
+ PID - USER TIME - SYS TIME - PROC STATE
53c92b29-9a58-4620-b7ab-2dc6d363de35 -
/usr/games/kmines - Done
95f6ad50-b918-4b71-8795-18ab78efe67a -office -
LocalSubmited * 84.88.50.104:60000

+ 9164 - 00:01.29 - 00:00.13 – S
```





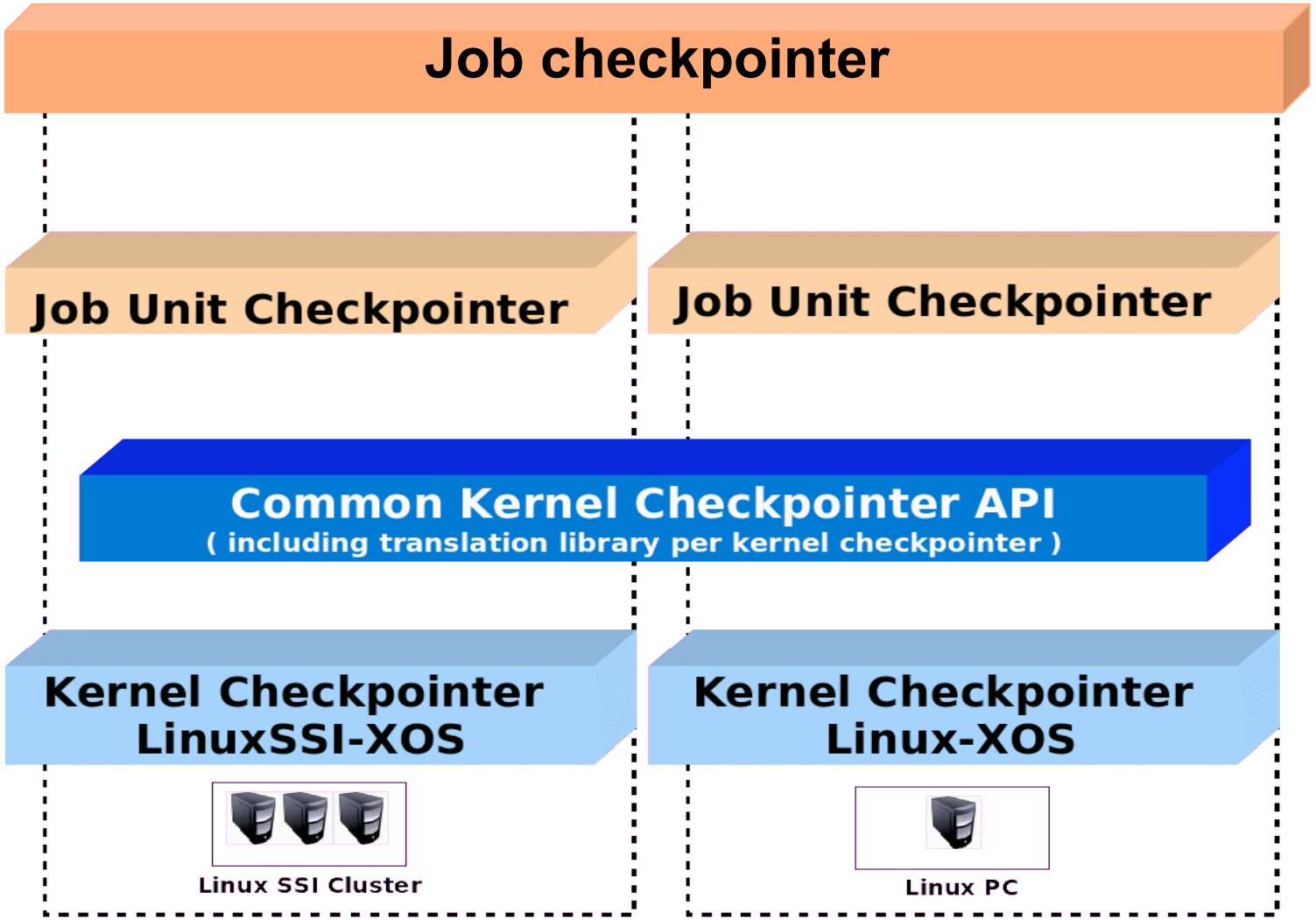
Job Monitoring Service

- Automatically provide information associated to jobs
- Provide mechanisms to limit the type and granularity of information collected
- Provide mechanisms to add new information
- Provide mechanisms to be notified when certain monitoring events occur (callbacks)

Checkpointing Service

- **Goal: checkpointing and restart for grid jobs**
  - Fault tolerance
  - Migration (scheduling / load balancing)

Grid Checkpointing Architecture



Work Directions

- **Checkpointing protocols for Grid applications**
  - Coordinated checkpointing
  - O2P protocol (optimistic message logging protocol)
- **Kernel checkpointers**
  - BLCR & Kerrighed checkpointer
    - Adapted for Grid usage (callbacks)
    - Steps for applications running on several Grid nodes
  - Kerrighed checkpointer
    - Message-passing based parallel applications
  - Monitoring the evolution of Linux kernel
    - cgroups, containers, name spaces
- **Security issues**
- **Checkpoint storage**

On-going Work

- Resource reservation & co-allocation
- Interface for workflow engine
- Monitoring & accounting

Infrastructure for programming AEM Services

- **DIXI (DIstributed Xtreemos Infrastructure)**
  - SEDA based AEM Infrastructure
  - Used to develop HIGHLY DISTRIBUTED Services
  - Provides communication layer abstraction, where services do not handle the sessions or connections
  - Exception h
- **XATI (XtreemO**
  - Automatic g
  - Support for
  - Fundamenta
    - Coding
    - Redunc

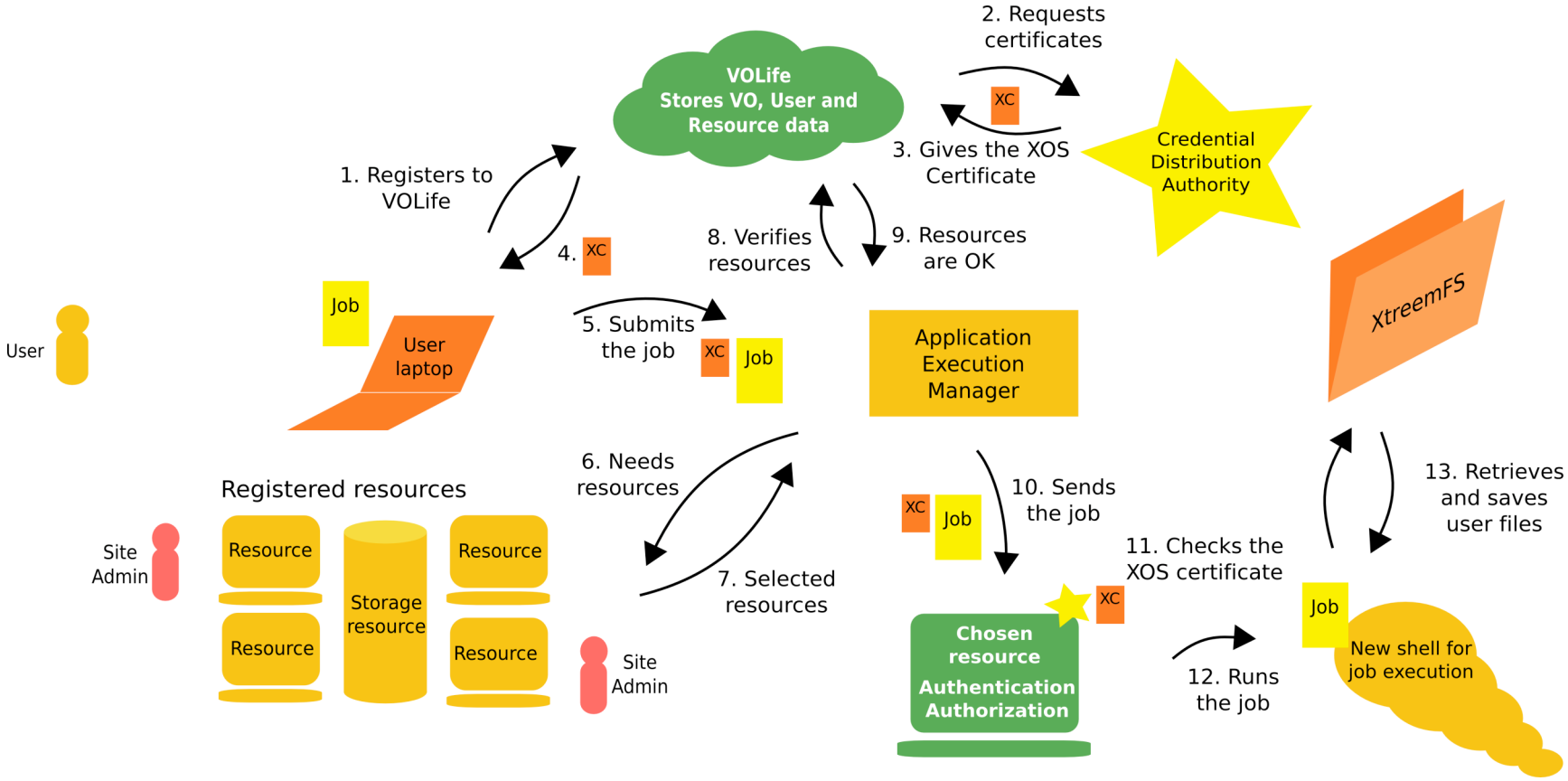
```
@XOSDSERVICE
public class JobMng extends Abstract2wayStage {

    private Hashtable<String, XJob> jobsList;
    ...

    @XOSDXATI(returntype = "String")
    public String createJob(String jsdlFile,
                            Boolean startJob){

        ....
        return job.getJobId();
    }
```

Putting Everything Together



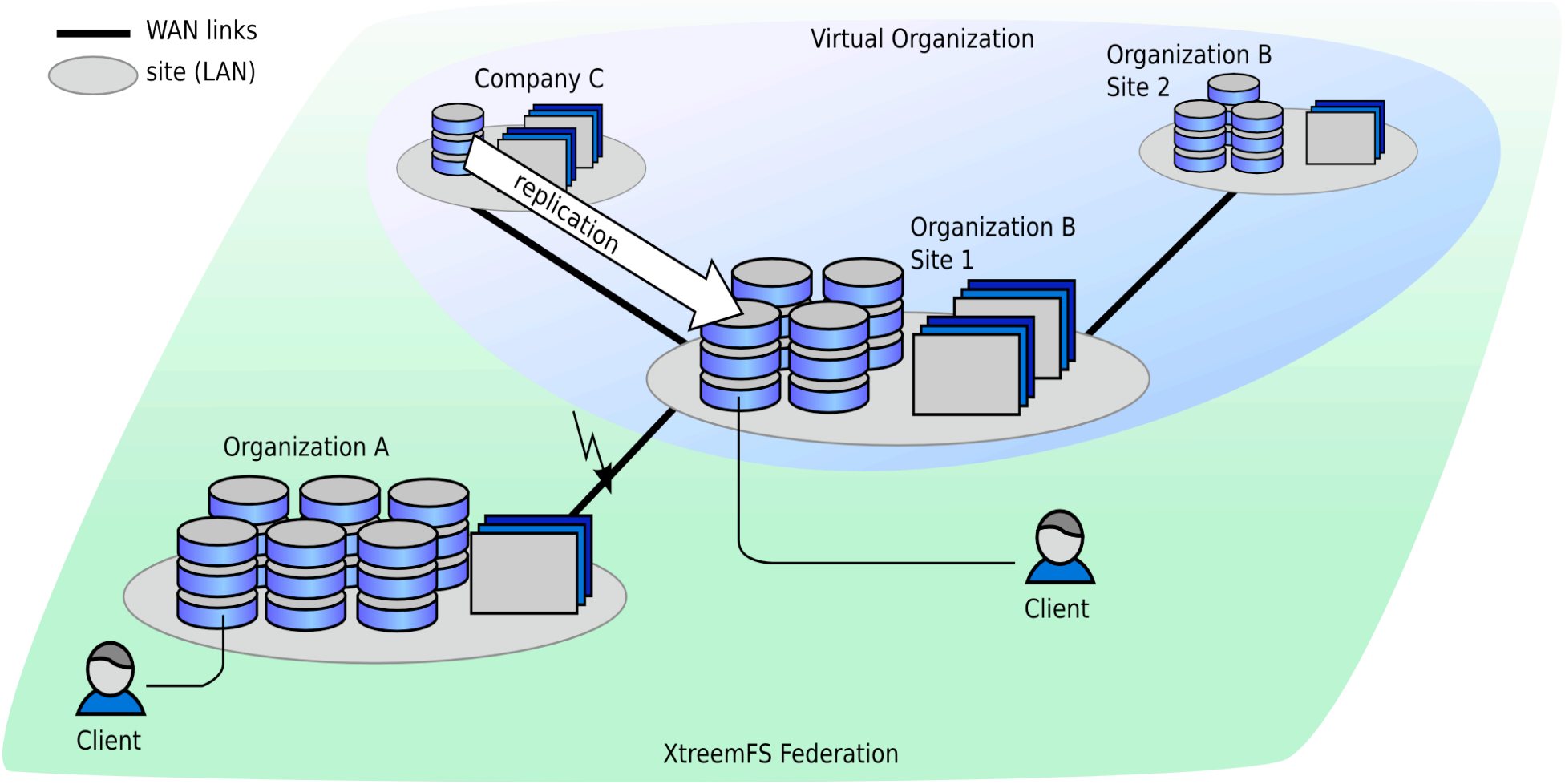
Outline

- XtreemOS project
- XtreemOS vision
- VO & security management
- *Demo VO & security*
- Job Management
- Data management
- *Demo user session*
- Infrastructure for scalable & highly available services
- XtreemOS API
- XtreemOS flavours
- Concluding remarks

Data Management in XtreemOS

- **XtreemFS Grid file system**
  - Persistent data
- **Oject Sharing System (OSS)**
  - Shared objects in memory

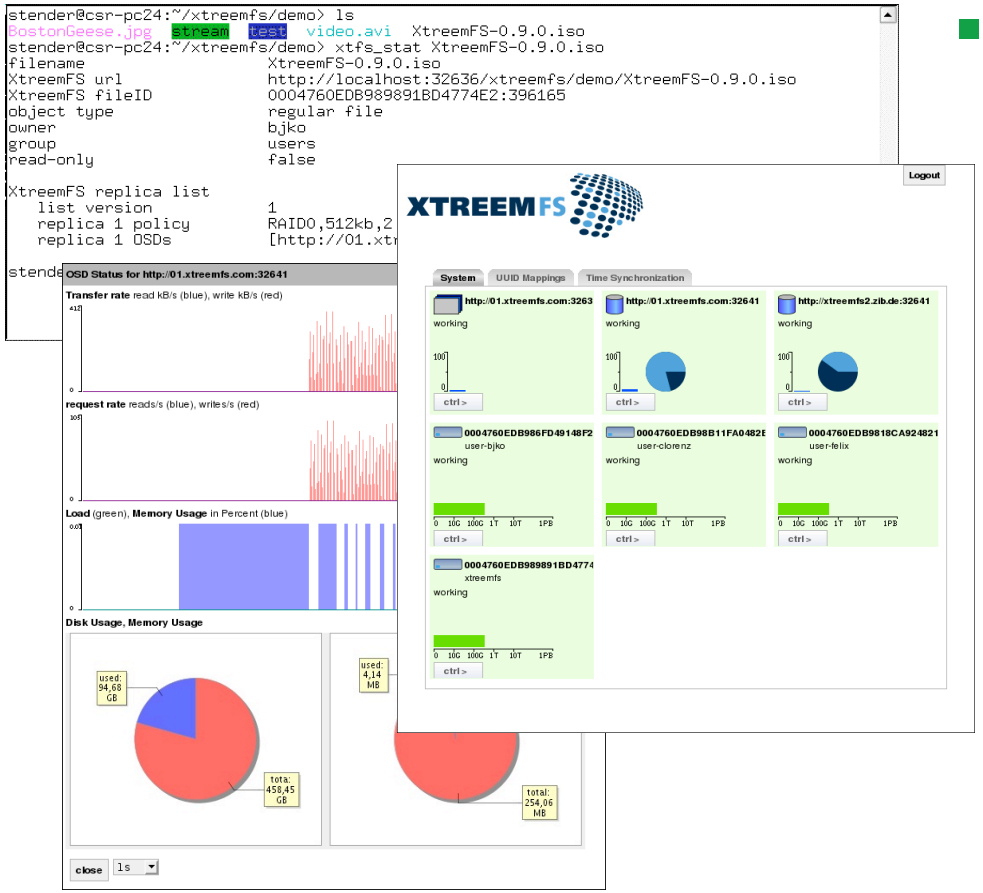
XtreemFS: Environment



XtreemFS: Environment

- federation: clusters can join/leave/fail
  - no centralized services at an organization
- connected over the Internet
  - complex failure cases (like network splits)
  - no control over hardware
- spanning administration domains
  - cross-organization authentication
  - virtual organization (VO) support necessary

XtreemFS: Overview



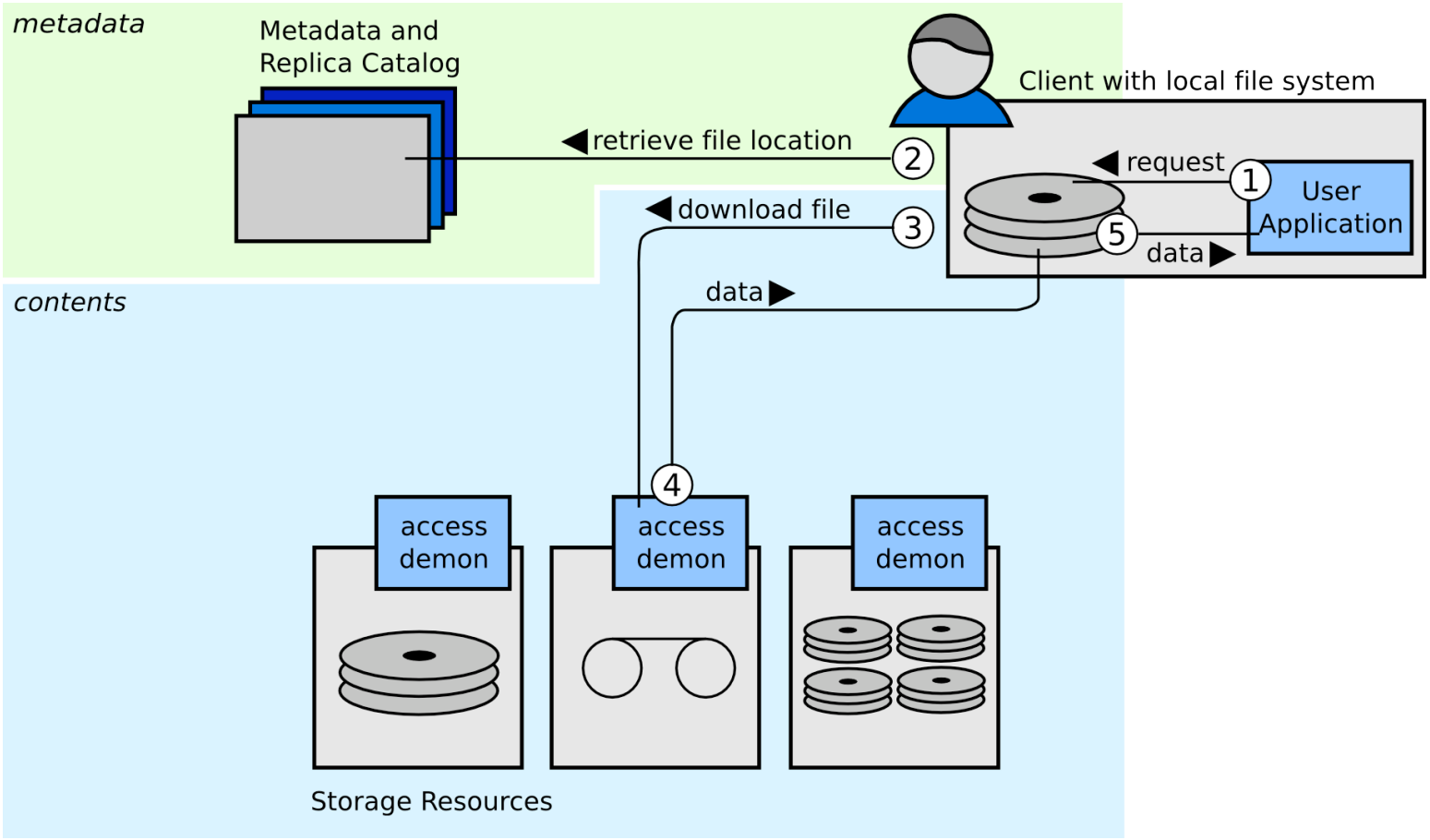
- What is XtreemFS?
  - A **distributed**, **replicated POSIX file system** for wide area networks
  - Mountable on any **Linux** machine
  - **Secure** (X.509 and SSL-based)
  - **Easy** to set up and maintain

XtreemFS Grid File System

- A Grid file system providing users with a global view of their files
- Posix interface
- Efficient location-independent access to data in a Grid
  - Grid users from multiple VO
  - Data storage in different administrative domains
- Autonomous data management with self-organized replication and distribution
- Consistent data sharing

XtreemFS vs. Traditional Grid Data Management

Traditional Grid Data Management



Object-based File Systems

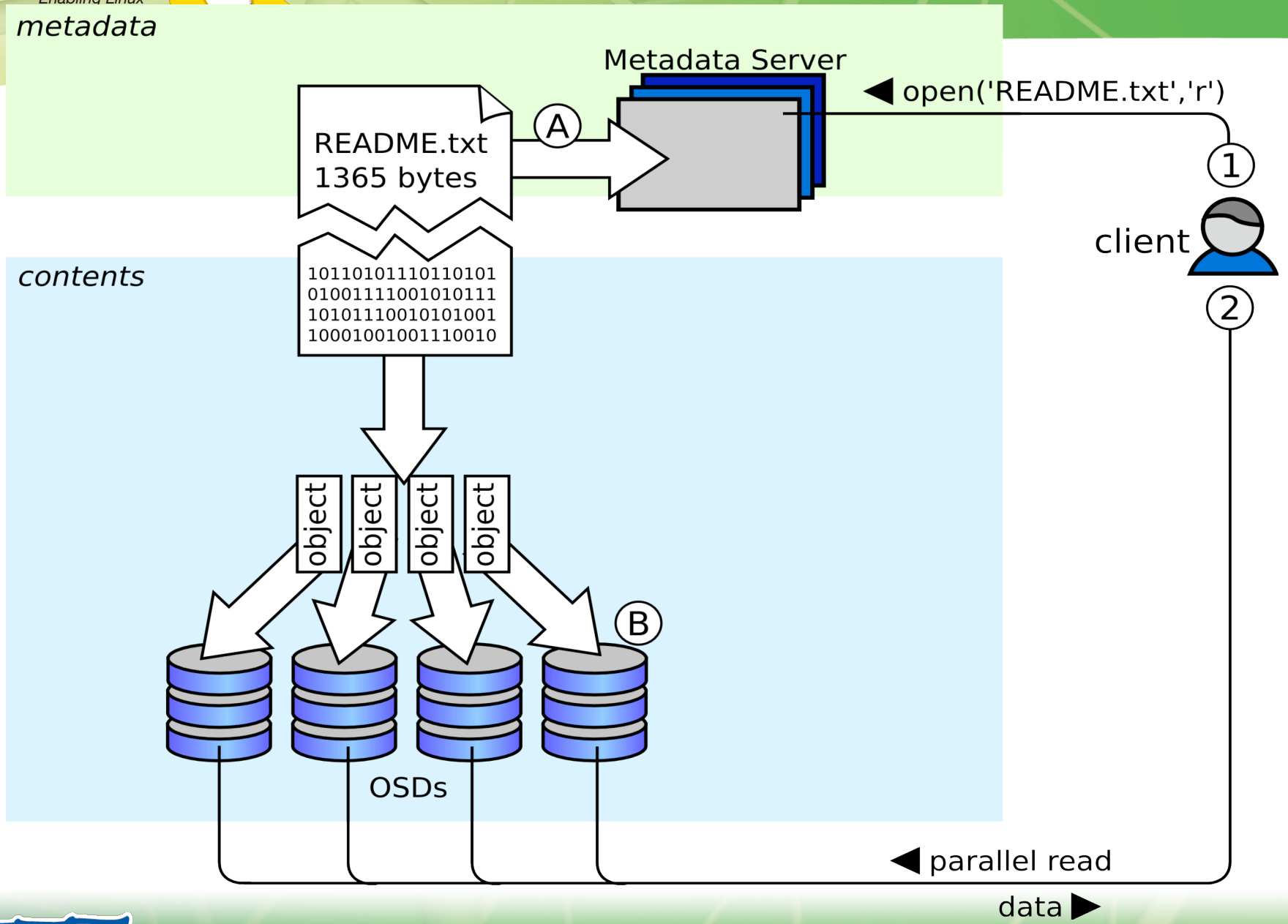
Block-based **File Systems**

- **Unit of distribution are disk blocks**
- **File system addresses blocks over the network**
- **Metadata and block-management at central server**

Object-based **File Systems**

- **Storage devices can be more intelligent today**
- **Split file in parts and distribute & address them**
- **Only metadata at server, block management by storage devices**

Object-based File Systems



Object-based File Systems

several available ...

- Lustre (Open-Source)
- Panasas ActiveStore (commercial)
- Ceph (Research, Open-Source)

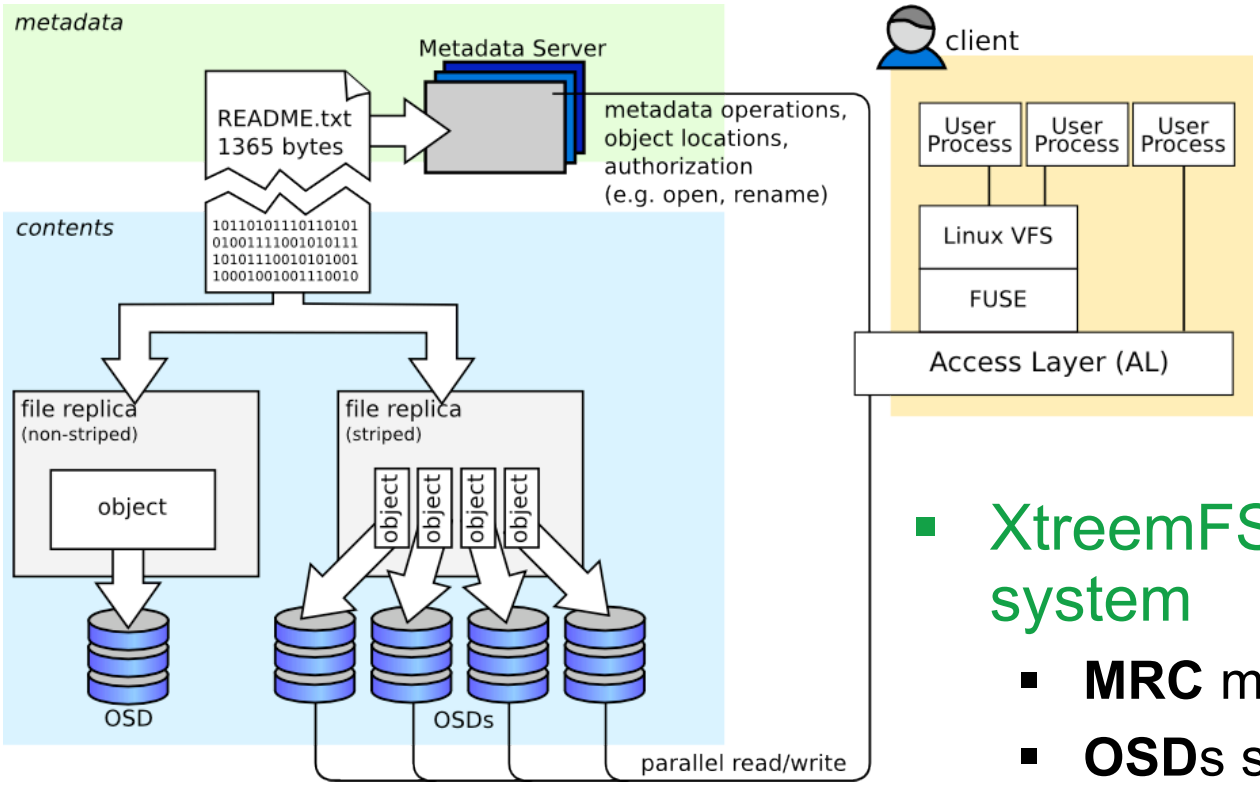
common properties:

- parallel designs for high-performance LAN access
- centralized, one data center, one organization
- control over failures of hardware

XtreemFS vs. Traditional Grid Data Management

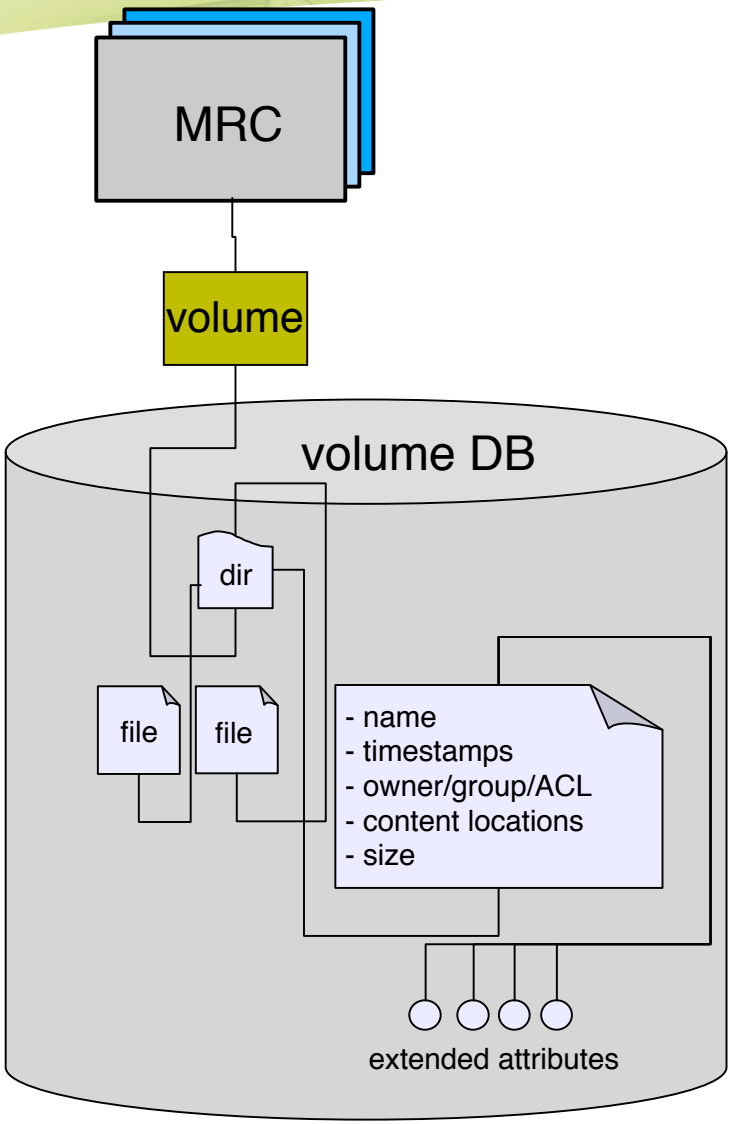
- Traditional Grid data management:
  Simple access to heterogeneous storage resources, but ...
  - in general, whole files have to be transferred and stored locally
    - high latency to first access
    - potential waste of network and storage resources
    - local access might be slower than network access
  - no automatic replica consistency
    - usually restriction to write-once usage patterns: download of input files, upload of output files
  - no access control on downloaded copies

XtreemFS: Architecture

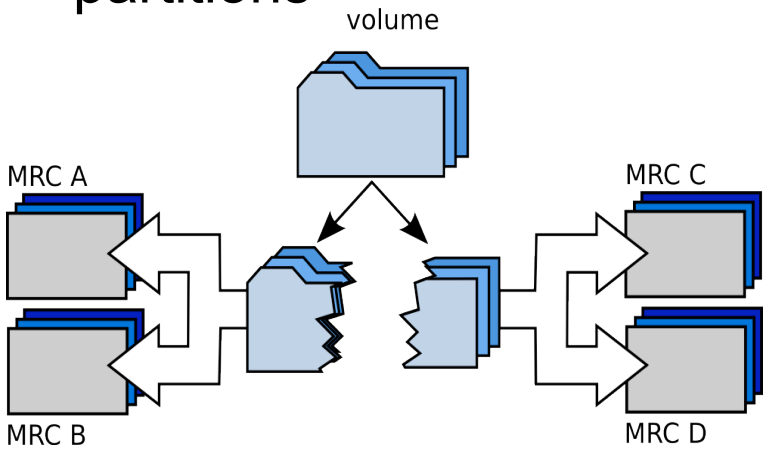


- XtreemFS: an object-based file system
  - **MRC** maintains metadata
  - **OSD**s store file content
  - **Client** (Access Layer) provides client access

Features - Metadata Management

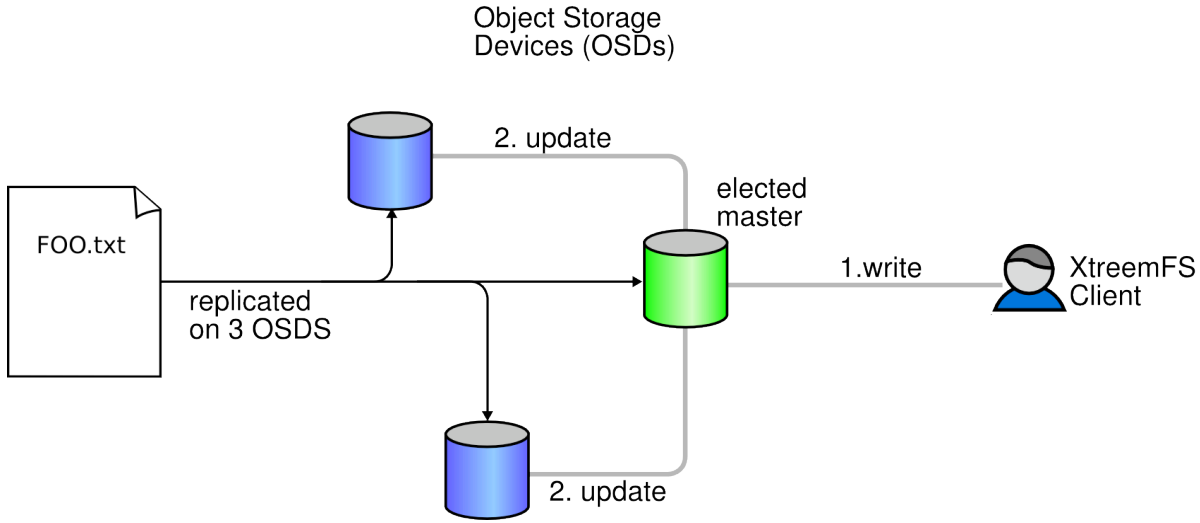


- **partitioning**
  - split up volume (DB) into smaller parts
- **replication**
  - primary/secondary with fail-over
  - granularity: volumes / volume partitions



XtreemFS: Core Features

- Replication
  - OSD locations of replicas are part of the file metadata
  - replica consistency is ensured by XtreemFS OSDs
  - current state: read-only file replication in development, will be released with version 1.0 (early 2009)



Consistency Coordination

replication of files

- read/write replication
- fully transparent to client
- guarantees sequential consistency
- primary/secondary approach with fault-tolerant lease negotiation

consistency coordination

- currently at object level
- synchronous, asynchronous or on-demand

Consistency Coordination

synchronous

- **writing**: acknowledge after all updates have been acknowledged
- **reading**: on any replica

Consistency Coordination

asynchronous

- **writing**: acknowledge when performed locally
- **reading**: check and fetch latest data

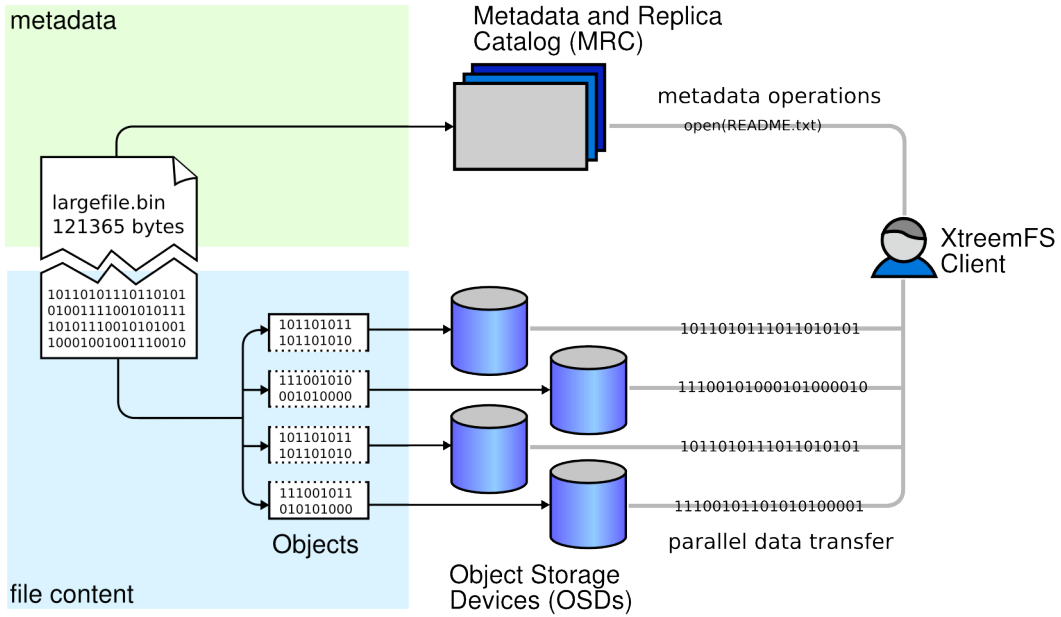
Consistency Coordination

on-demand

- **writing**: acknowledge when performed locally, do not disseminate updates
- **reading**: check and fetch latest data

XtreemFS: Core Features

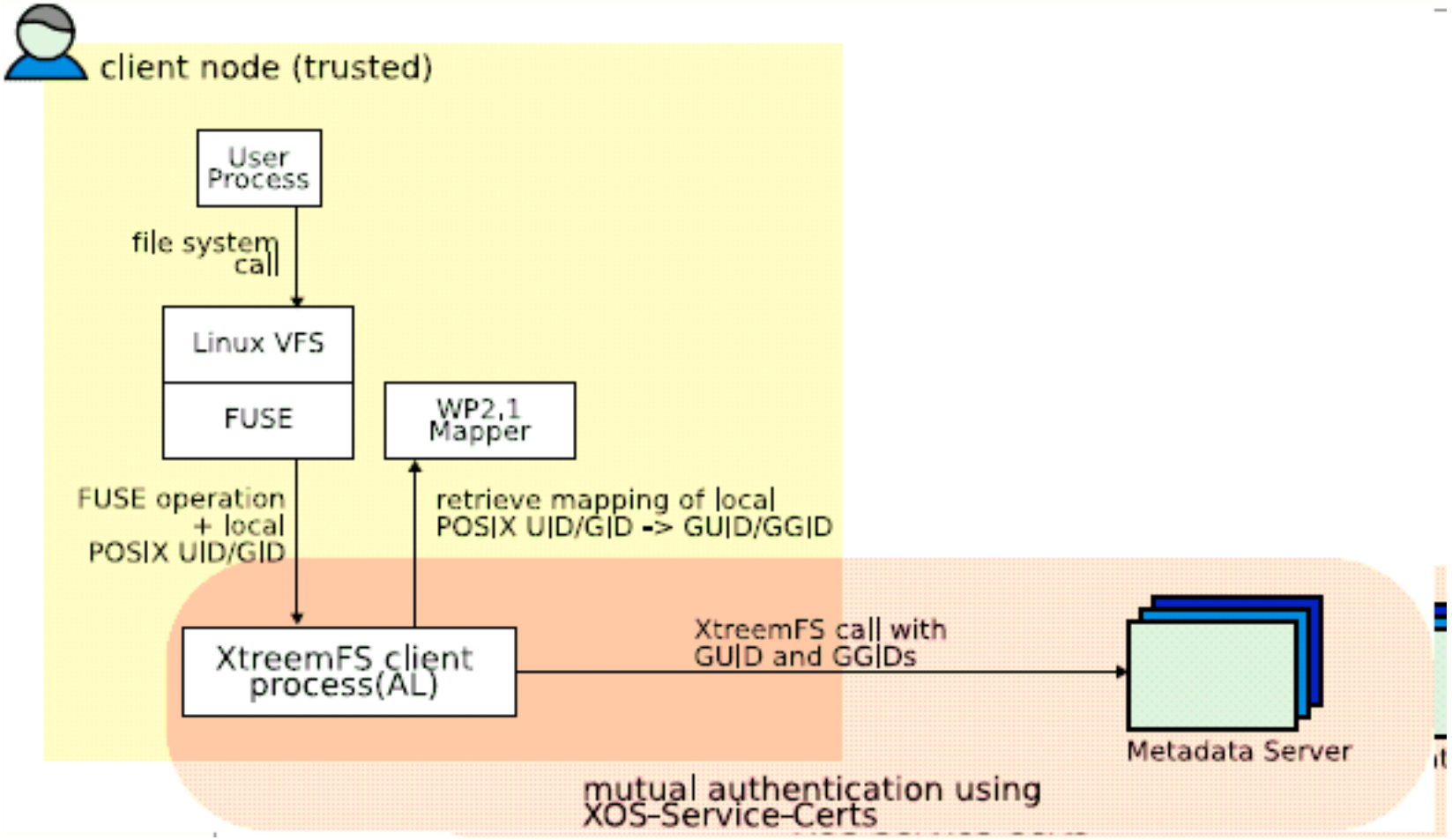
- Striping
  - MRC stores a per-file list of OSDs
  - parallel data transfer from/to multiple OSDs
  - current state: RAID-0 supported (release 0.9.0)



XtreemFS: Features

- Features (current release 0.9.0)
  - POSIX-compliant interface and semantics ✔
  - Access control via POSIX permissions and POSIX ACLs ✔
  - X.509-based authentication, SSL encryption ✔
  - Master-slave replication of metadata ✔
  - Extended metadata ✔
  - Plug-in architecture for user-defined policies ✔
  - Monitoring and management tools ✔

XtreemFS Security



Implementation

Protocol

- HTTP (with JSON encoding for RPCs)

MRC, OSD, Directory Service

- staged server implementation (non-blocking I/O)
- Java (~40.000 LOC) + BerkeleyDB (MRC)

File System Client

- FUSE-based implementation (for now)
- C (~13.000 LOC)

Object Sharing Service (OSS)

- **Objective**
  - Simplify data exchange and consistency maintenance in Grid applications
- **OSS**
  - Provide storage for volatile objects
    - Unstructured storage: applications may store anything in objects
    - Scalability: many nodes can allocate and free any number of objects
    - Consistency: application can choose between different consistency models
  - Offer familiar memory management interface
  - Integrate neatly with Linux

OSS

- **Features**
  - Transactional memory
    - Several memory operatins bundled in a TA (ACID properties)
  - Scalability
    - P2P techniques, weakly consistent object, multiple consistency domains, pipelined TA

OSS Interface

• void *alloc(size, consistency) • void free(void *object) • void *mmap(URL, offset, len, cons)	Memory management
• transaction *bot() • void eot(transaction *) • void abort()	Transaction management
• void startup(own_ip, bootstrap_ip)	Node management

Outline

- XtreemOS project
- XtreemOS vision
- VO & security management
- *Demo VO & security*
- Job Management
- Data management
- *Demo user session*
- Infrastructure for scalable & highly available services
- XtreemOS API
- XtreemOS flavours
- Concluding remarks

Outline

- XtreemOS project
- XtreemOS vision
- VO & security management
- *Demo VO & security*
- Job Management
- Data management
- *Demo user session*
- Infrastructure for scalable & highly available services
- XtreemOS API
- XtreemOS flavours
- Concluding remarks

Infrastructure for scalable & highly available services

- **Three kinds of services:**
  - Services to store/query structured data
    - Resource Selection Service
    - Application Directory Service
  - Services to communicate in a scalable fashion
    - Publish-subscribe
  - Services to (partially) hide the effects of scale
    - Hide resource distribution: distributed servers
    - Hide resource failures: virtual nodes
- **Main challenge: scalability!**

Services

- **Resource Selection Service (RSS)**
  - A new P2P overlay to select resources from properties
  - No delegation: each node represents itself in the overlay
- **Application Directory Service (ADS)**
  - Support for dynamic information and data lifespan
- **Publish/subscribe**
  - DHT-based structure for scalability
  - Unique feature: transactional guarantees
- **Virtual nodes (VN)**
  - Transparent replication of Java-based services
  - Dynamic choice of replication protocol
- **Distributed servers (DS)**
  - Transparent client handoff even in case of node failure

Use Cases

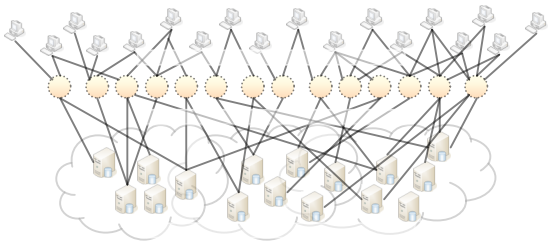
- **Services offered to applications**
- **Services used by XtreemOS system services**
  - RSS & ADS used by AEM
  - Pub/Sub, ADS to be used by XtreemFS
  - VN and DS to be used for critical components of AEM/XtreemOS/VO-Security services

Outline

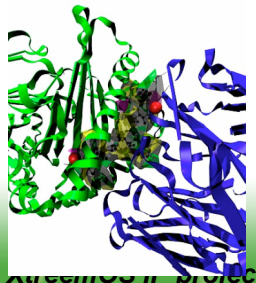
- XtreemOS project
- XtreemOS vision
- VO & security management
- *Demo VO & security*
- Job Management
- Data management
- *Demo user session*
- Infrastructure for scalable & highly available services
- XtreemOS API
- XtreemOS flavours
- Concluding remarks

Application Spectrum

- **Wide range of applications…**
  - Grid aware distributed applications
  - Grid unaware (legacy) applications executed in a Grid



- **… in different domains**
  - E-business
    - Services…
  - Scientific applications

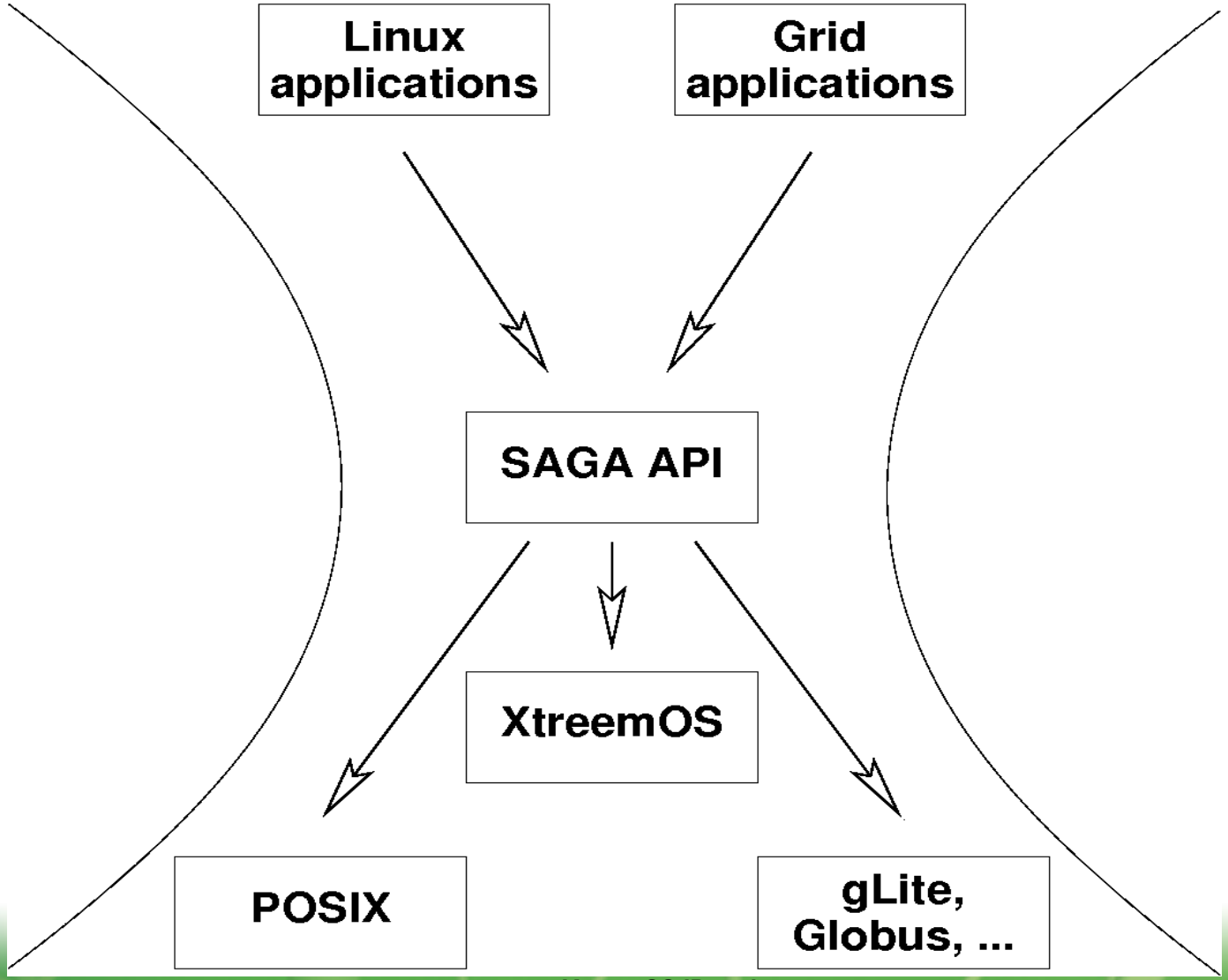


… XtreemOS is an OS!

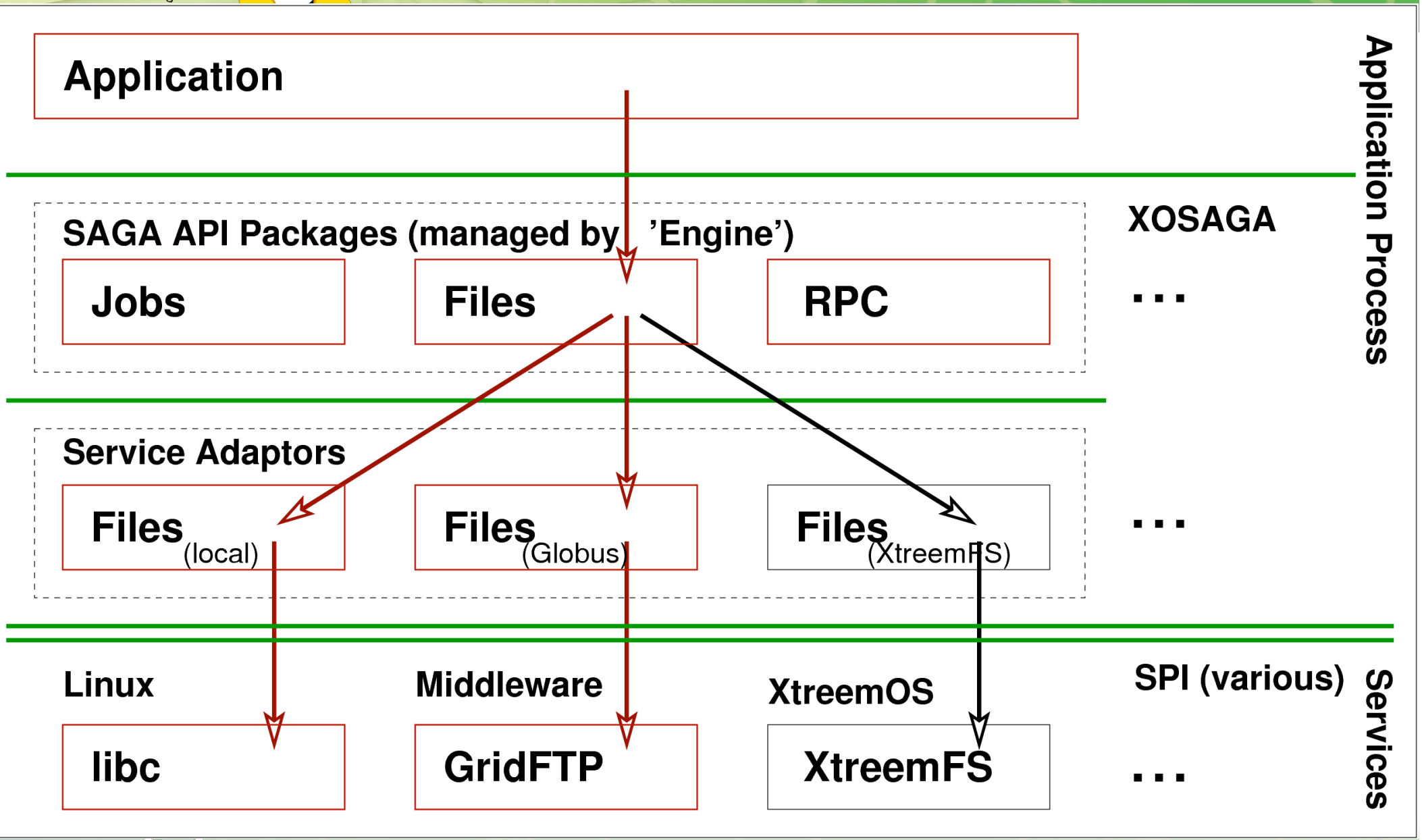
Running Applications in XtreemOS

- Linux applications should run with little (no) modifications
- Grid applications should run with little (no) modifications
- XtreemOS functionality must be provided to applications

SAGA Bridging Linux & Grids



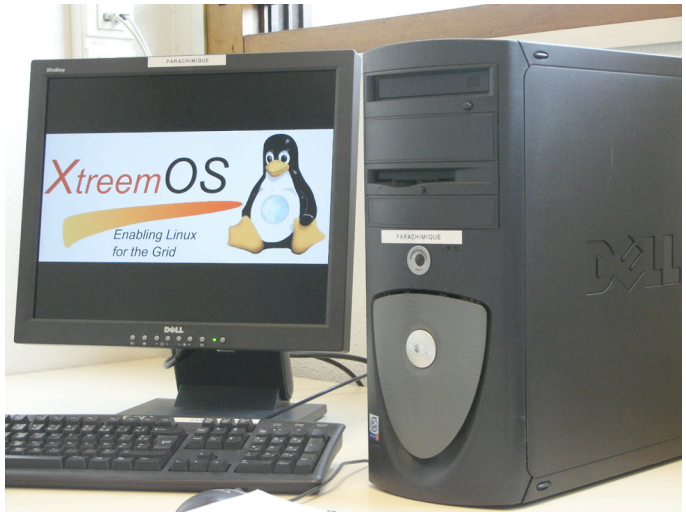
SAGA API C++ Implementation Structure



Outline

- XtreemOS project
- XtreemOS vision
- VO & security management
- *Demo VO & security*
- Job Management
- Data management
- *Demo user session*
- Infrastructure for scalable & highly available services
- XtreemOS API
- XtreemOS flavours
- Concluding remarks

XtreemOS Flavours

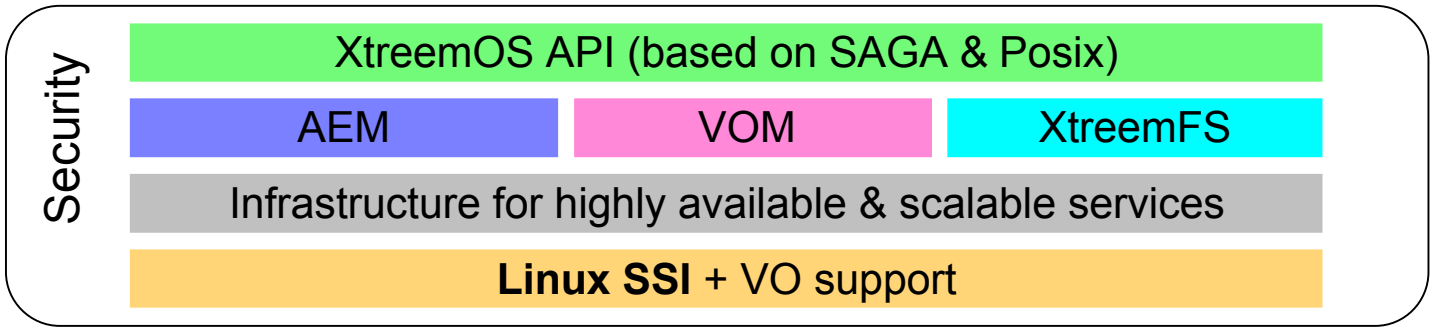


Stand-alone PC

Cluster

Mobile device

XtreemOS Cluster Flavour

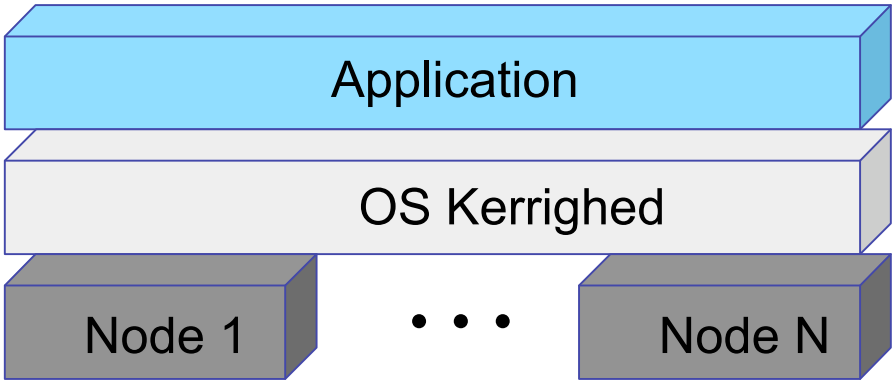


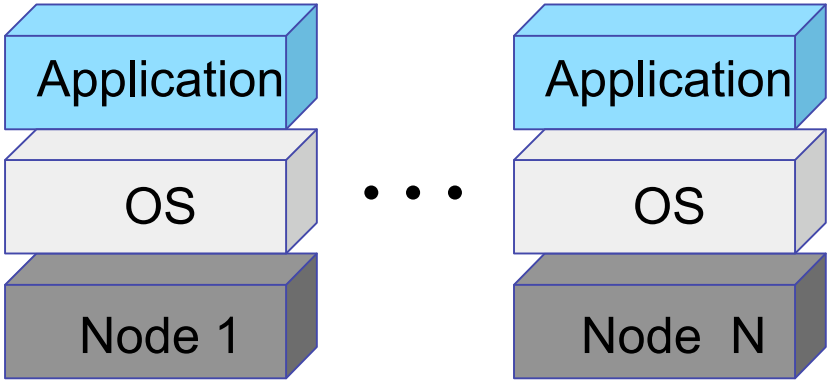
- **Based on LinuxSSI foundation layer**
  - Linux based Single System Image cluster OS
    - Illusion of a powerful SMP machine running Linux
  - Leverage Kerrighed full SSI
    - Posix compliant interface validated by successfully running the standard Linux Test Suite

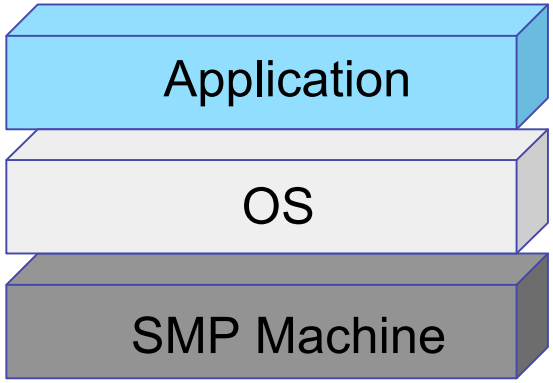
Kerrighed
Linux clusters made easy

Kerrighed Cluster OS

- **SSI**
  - Virtual SMP
  - Legacy applications executed without any modification or recompilation
  - Standard OS interface







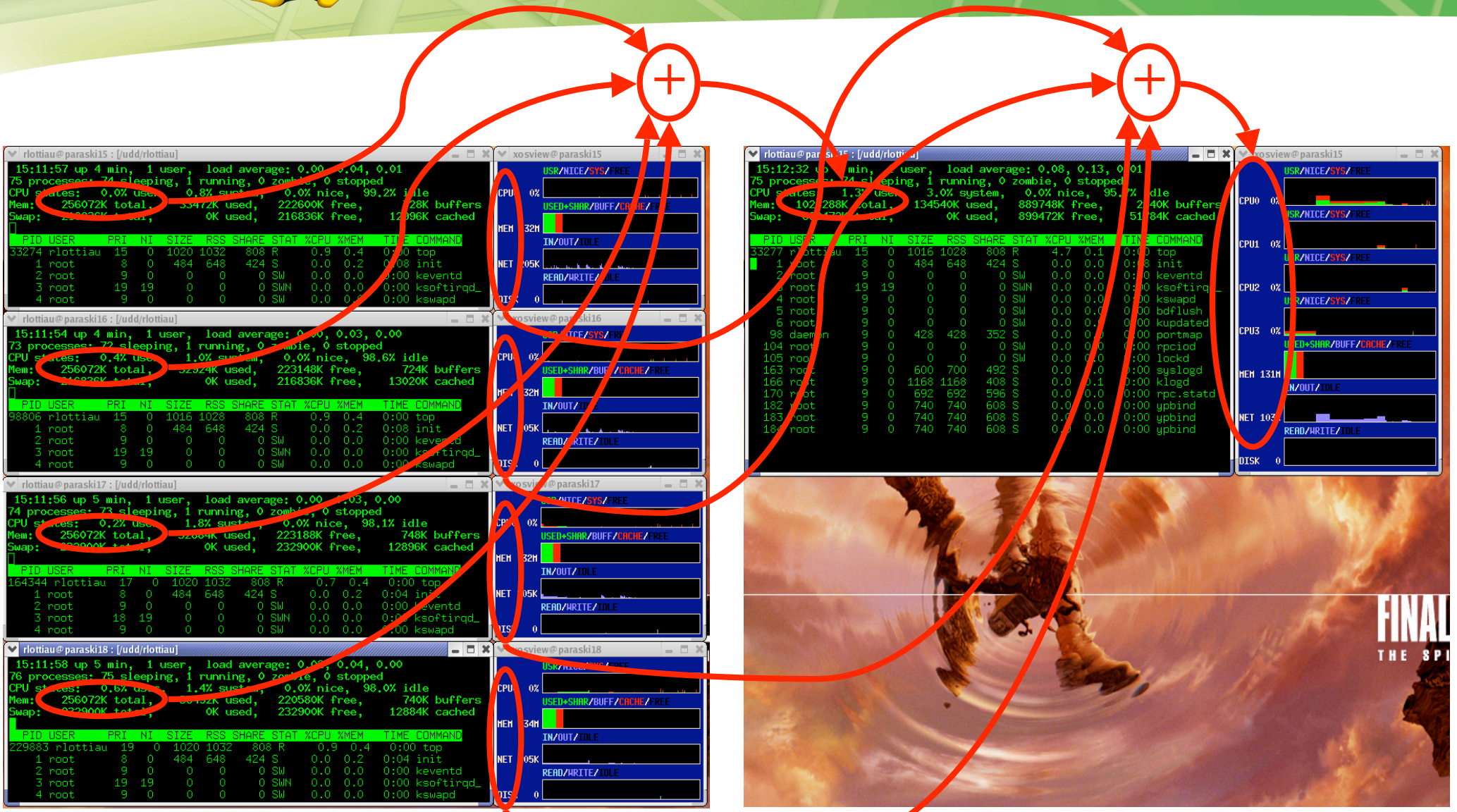
Why a full Linux SSI?

- **Ease of use for non expert users**
  - Standard Linux commands
    - Interactive use of a cluster (without a batch)
  - Execution of unmodified legacy applications
    - Including parallel applications based on the shared memory model
- **Ease of management**
  - Single instance of the OS for the whole cluster
  - Hot node addition or eviction without stopping the whole cluster
- **Unique Features**
  - kDFS distributed/parallel file system for efficient data accesses exploiting compute node storage
  - Framework for customized load balancing & scheduling policies
  - Built-in efficient checkpoint/recovery mechanisms

XtreemOS

Enabling Linux for the Grid

Standard Linux vs Linux/Kerrighed



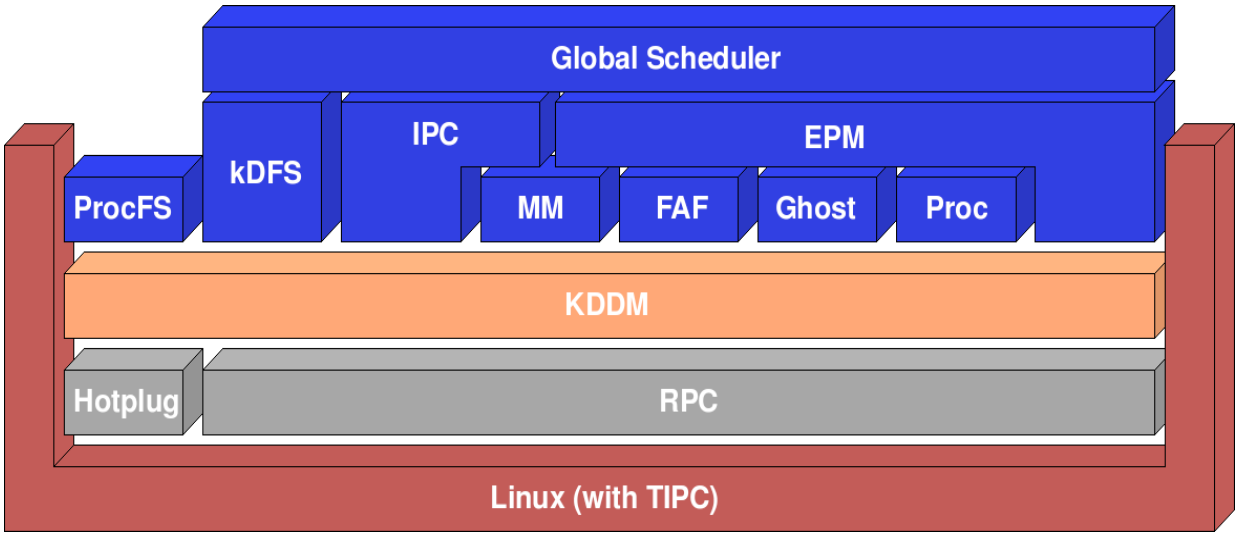
Standard Linux

Linux/Kerrighed

XOSVIEW **unmodified**

LinuxSSI Implementation

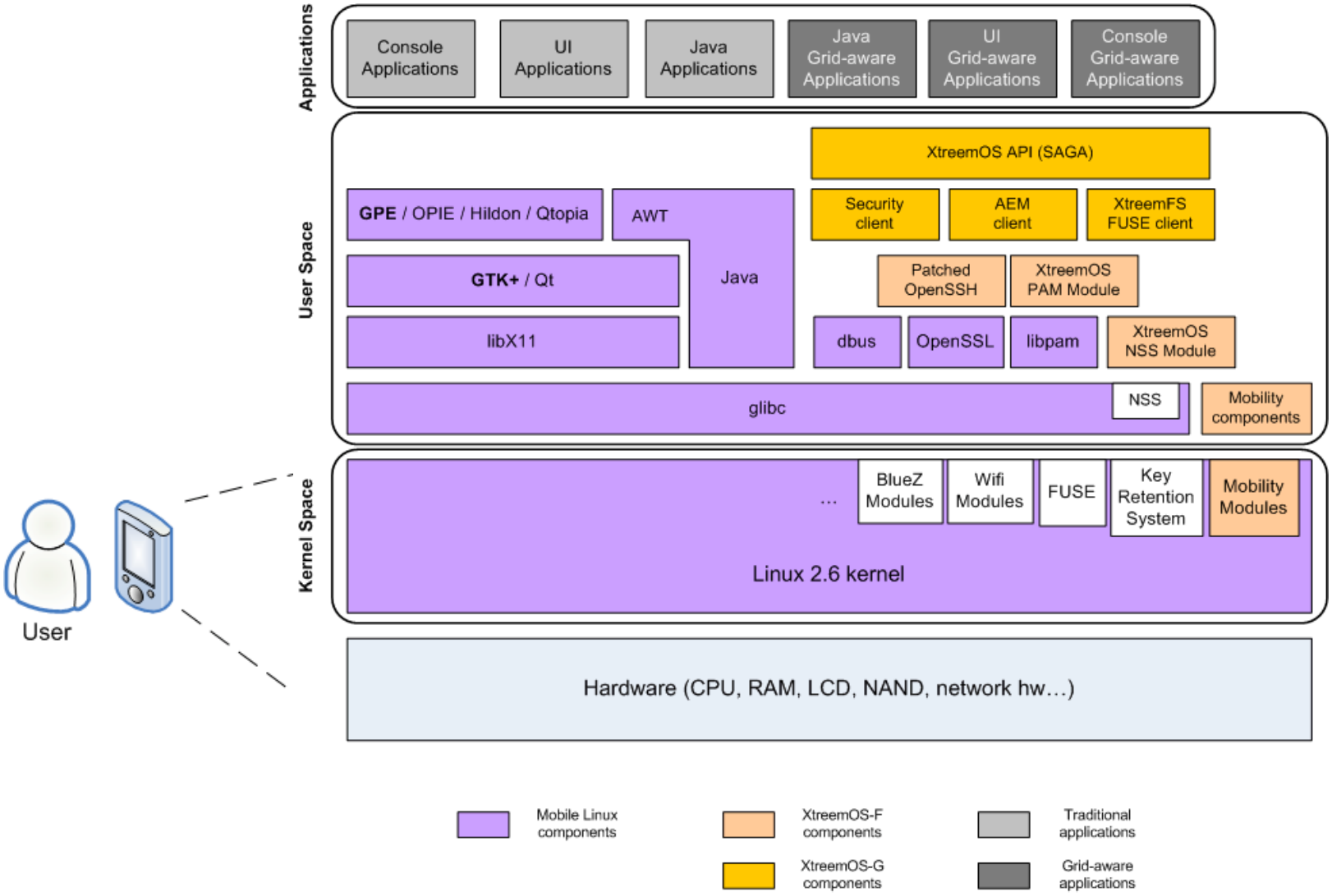
- Set of distributed services for global resource management
- Implemented by a set of patches to the Linux kernel & kernel modules



XtreemOS Mobile Device Flavour

- **Objectives**
  - Integration of XtreemOS services in mobile Linux OS enabling grid operation in an efficient and transparent way
  - Targets
    - Grid aware use cases
      - Grid users on the move
    - Grid-transparent use cases
      - Services given through a Grid infrastructure without the end users knowing it (Mobile Linux integrators)
  - Portability

Mobile Device Architecture



Outline

- XtreemOS project
- XtreemOS vision
- VO & security management
- *Demo VO & security*
- Job Management
- Data management
- *Demo user session*
- Infrastructure for scalable & highly available services
- XtreemOS API
- XtreemOS flavours
- Concluding remarks

XtreemOS Software

- **First public release of XtreemOS software (open source)**
  - Fall 2008
  - Mandriva & RedFlag Linux distributions
  - http://www.xtreemos.eu & sf.net
- **Demonstrations**
  - SC '08, Austin, USA, November 16-20, 2008 (XtreemOS booth #3019)
  - ICT'08, Lyon, France, November 25-27, 2008
- **Contributions welcome**
  - Beta-testers, developers

Reading List

- **XtreemOS**
  - ISORC 2007 paper
  - Vision paper (XtreemOS technical report #4)
  - Architecture deliverable D3.1.4 (updated version by the end of December 2008)
- **VO & Security**
  - D3.5.x (VO & security services)
  - D2.1.x (VO support in Linux)
  - IEEE Internet 2008 paper

Reading List

- **AEM**
  - D3.3.x deliverables
  - PDCAT 2008 paper on Grid checkpointer
- **XtreemFS & OSS**
  - D3.4.x deliverables
  - HPDC 2008 paper on XtreemFS
- **Infrastructure for scalable & highly available services**
  - D3.2.x deliverables
- **XtreemOS API**
  - D3.1.x deliverables
  - SAGA (see OGF documents)

Reading List

- **Cluster flavour**
  - D2.2.x deliverables
  - Europar 2008 paper on kDFS
  - XtreemOS technical report on customizable scheduler
  - Kerrighed
    - http://www.irisa.fr/paris for scientific papers
    - http://www.kerrighed.org for software & technical documentation
- **Mobile device flavour**
  - D2.3.x & D3.6.x deliverables

Information & Contact

- **Information**
  - http://www.xtreemos.eu
- **Contact**
  - info@xtreemos.eu
- **Teachers**
  - Christine.Morin@inria.fr
  - Sylvain.Jeuland@inria.fr