D1.1g - VISHNU Technical Architecture

COLLABORATORS

	TITLE : D1.1g - VISHNU Technical Architecture		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Benjamin Isnard, Benjamin Depardon, Daouda Traore, Ibrahima Cissé, and Kevin Coulomb	December 14, 2011	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
1	08/02/2011	Deliverable version	SysFera
2	11/03/2011	Added chapter 5 for TMS and FMS details In §3.1, added definition for VISHNU machine In §3.4, modified flows VFT, UFT and DFT, and added CFT	SysFera
3	25/03/2011	In §3.4, added flows COS and OSS	SysFera
4	22/07/2011	Added SLURM	SysFera
5	07/12/2011	Added SLURMJobScheduler on Figure 3.3 and replace of ORACLE by MYSQL on the document such as Figure 5.3 and Figure 5.6	SysFera

Contents

List of Figures

List of Tables

Chapter 1

Document presentation

1.1 Document objectives

This document provides a global view of the architecture for VISHNU and its environment. The target audience of this document is composed of software, system and network architects. The main objective of this document is to validate that the solution provided matches the needs and constraints of the target environment. It can also be considered as a first version of a checklist for system deployment but some deployment details may still not be defined at the time this document is written and may be completed during the project development.

1.2 Document structure

- Chapter 1 contains a description of the document itself and definitions of terms and acronyms.
- Chapter 2 contains a global description of the VISHNU project: its objectives, its context including the main constraints and the main technical choices made for the technical solution proposed by SysFera.
- Chapter 3 contains a description of the functional architecture of VISHNU: users, functional units, interfaces, data and security.
- Chapter 4 contains a description of the technical architecture of VISHNU: clients and servers, network, databases.
- Chapter 5 contains details of the architecture that are specific to some VISHNU modules.

1.3 References

- [D1.1a]: VISHNU General specifications
- [D1.1b]: VISHNU "Spécifications techniques des besoins"
- [D1.1c]: VISHNU API Detailed specifications

1.4 Acronyms and abbreviations

- **API**: Application Programming Interface
- **CLI**: Command Line Interface
- **DB**: Database
- **FMS**: File Management Service (VISHNU Module)

- **IMS**: Information Management Service (VISHNU Module)
- **kB**: Kilo-bytes
- **MB**: Mega-bytes
- **n/a**: Not Applicable
- **QoS**: Quality of Service
- **TBC**: To Be Completed
- **SSH**: Secure Shell
- **TMS**: Tasks Management Service (VISHNU Module)
- **UML**: Unified Modeling Language (current version: v2.3)
- **UMS**: Users Management Service (VISHNU Module)
- **WS**: Web Services

1.5 Glossary

- **Computing resource**: a server that is used to process batch jobs. This definition covers supercomputers, the cluster gateway servers and the cluster processing nodes.
- **Host**: represents a physical client or server computer system.
- **Middleware**: an application that sits "in the middle" between application software that may be working on different operating systems. It is similar to the middle layer of a three-tier single system architecture, except that it is stretched across multiple systems or applications. [source: Wikipedia]
- **Network firewall**: a system that filters the network traffic between two or more different network areas based on traffic protocol patterns.
- **POSIX**: international standard that defines an API to access a filesystem.
- **Storage resource**: a server that is used to store data as files. This definition covers the user desktops or laptops and all unix servers.
- **SysFera-DS**: open-source middleware developped by SysFera.
- **SysFera-DS Agent**: SysFera-DS process running permanently (or daemon).
- **VISHNU system**: set of all elements (software, hardware, data) that compose a single instance of the VISHNU application and that work together to provide VISHNU services to the users.

Chapter 2

Global description of VISHNU

This chapter contains a synthetic presentation of the ins and outs of the system architecture.

2.1 Project sheet

Project Title	VISHNU
Version	1.0
Objective	To provide a standardized, integrated, secure, reliable and high performance interface to heterogeneous scientific computing resources used by research and development teams.
Beneficiaries	EDF R&D
Impact	This project is critical for the beneficiary organization. The realized system will be transfered to different enterprise organizations for deployment on their own infrastructure.
Project Duration	7 months
Major constraints	- VISHNU architecture must not be tied to a given deployment environment but must be usable on different system and network environments as long as they are compliant with the requirements defined in [D1.1b]. - The installation and usage of VISHNU must not require root access on the computing or storage resources. - The source code must be under the CeCILL A v2 license and VISHNU must be distributed as a debian package and a source package.
Security constraints	- VISHNU requires user authentication.
Traceability constraints	VISHNU will keep track of all user actions with their date and time, user identifier, client host identifier and action details. VISHNU will provide a log of all user actions that allows the user herself or an administrator to redo past actions.
System services	VISHNU will provide the following services: • User management services (UMS): authentication and session management. • Information management services (IMS): monitoring and control services. • Tasks management services (TMS): submission of tasks (jobs) on computing resources. • File management services (FMS): display and transfer of files between storage resources.

System interfaces	VISHNU will provide the following interfaces: • a Web Services interface • a Python shell interface • a C++ application programming interface (API) • a Unix command line interface
System users	VISHNU will be used by managers, engineers and researchers. A user may either use a man-machine interface (web or unix shell) or a programming interface (Python or C++) to access the system.
Database system	VISHNU will use either a PostgreSQL 8.x or MySQL v5.x database server.
Architecture choices	- VISHNU will be based on a distributed service-oriented middleware for the communication between all clients and servers. - VISHNU will be modular so that new services can be easily integrated with the existing ones and so that each module can have its own version. - VISHNU will use the SSH protocol for file transfers and for tunneling of application flows across network firewalls. - VISHNU will use the mechanisms of the underlying Unix security system to protect access to sensitive information (e.g. SSH keys). - VISHNU will store all internal and user data (except user files) in a central database.

2.2 QoS objectives

Availability	- The individual failure of a computing or storage resource must not impact the global availability of VISHNU but only services located on that resource. - The components of the VISHNU system with the exception of the Web Server and the Database will be monitored internally and restarted automatically within less than 1 minute (the real delay will depend on the monitoring frequency which will be set by the administrator) - VISHNU will keep internal data stored in a central database and use this data to recover its state before a crash or manual termination. - In case of database crash, VISHNU will be restarted using a standby database that contains a copy of the main database.
Performance	- VISHNU will support at least 100 simultaneous users and peaks of 100 simultaneous requests. - The latency for all user requests sent through the VISHNU C++ API or CLI will be less than a similar request done using SSH.

Chapter 3

Functional architecture

3.1 VISHNU concepts

This paragraph introduces important concepts that are used in the definition of the functional and technical architecture

- **Computing server**: a server that provides computing services to the user through a batch scheduler software like LoadLeveler, Torque or SLURM. In the case of a cluster, the computing server is also known as the gateway server.
- **Database Server (DBS)**: represents a database server used by VISHNU.
- **Storage server**: a server that provides filesystem access to the user through a standard POSIX API.
- **VISHNU Deployment Map**: this map defines which physical host is providing which VISHNU services or on which are running VISHNU internal processes. This map will be described physically as a XML file and will be used by the VISHNU Admin System (see below inside the VBS category) to launch, check and stop the VISHNU system.
- **VISHNU Services Servers (VSS)**: this is a generic component of the functional architecture that represents all the VISHNU server components providing services to the user. The different classes of this generic component are the following in the context of the VISHNU 1.0 project, but the list of classes could be extended in the future:
  - FMS Server (File Management Server): this role can be assigned only to a Storage Server
  - IMS Server (Infrastructure Management Server): this role can be assigned to any VSS
  - TMS Server (Tasks Management Server): this role can be assigned only to a Computing Server
  - UMS Server (Users Management Server): this role can be assigned to any VSS
- **VISHNU Backbone servers (VBS)**: this is a generic component of the functional architecture that represents all VISHNU server components that are used by VISHNU to handle user requests (i.e. receive the request and forward it to the appropriate VSS) and to administrate VISHNU. A physical server may host both a VISHNU Backbone server and a VISHNU Services Server as these are two independent functionalities. The different classes of this generic component are the following:
  - **SysFera-DS Forwarder**: represents a SysFera-DS agent that provides connectivity between different networks through a dedicated SSH tunnel.
  - **SysFera-DS MasterAgent (MA)**: represents a SysFera-DS agent that processes requests coming directly from client systems.
  - **VISHNU Admin system (VAS)**: represents a component used by an administrator to launch and stop the VISHNU system and to restart individually VISHNU processes.
- **VISHNU Database**: represents the logical database used by VISHNU to store internal data.
- **VISHNU Machine**: represents a given host that has been registered in the VISHNU Database by a VISHNU administrator. Each machine is identified by a unique identifier that is generated when the machine is registered in VISHNU.

- **VISHNU Module**: macro software component that provides a set of services to the user. It contains both client and server software providing these services. A module may have dependencies to other modules.
- **VISHNU User**: represents the unix user that is used to run all VISHNU internal processes (VBS or VSS) on the different host(s) used by the VISHNU System.

3.2 Functional architecture diagram

The Figure 3.1 shows the different functional parts of VISHNU and their interactions. On the client side (upper part of the diagram) are shown the different interface components that are provided to the users: APIs and command-line interface. These interfaces provide access to the different VISHNU modules (UMS, TMS, FMS, IMS) using a client component, that is itself connected with the corresponding server component through the SysFera-DS Middleware. The server components handle data items that belong to different inter-related databases, each managing a specific kind of data (a more detailed description of the databases is provided in section 3.2). The TMS and FMS Server components also handle some specific interaction with external entities: the batch schedulers (LoadLeveler, Torque or SLURM) for the management of jobs on clusters or supercomputers, and the SCP and RSYNC programs of the underlying Linux system for the transfer of files between two storage servers or between a client system and a storage server.

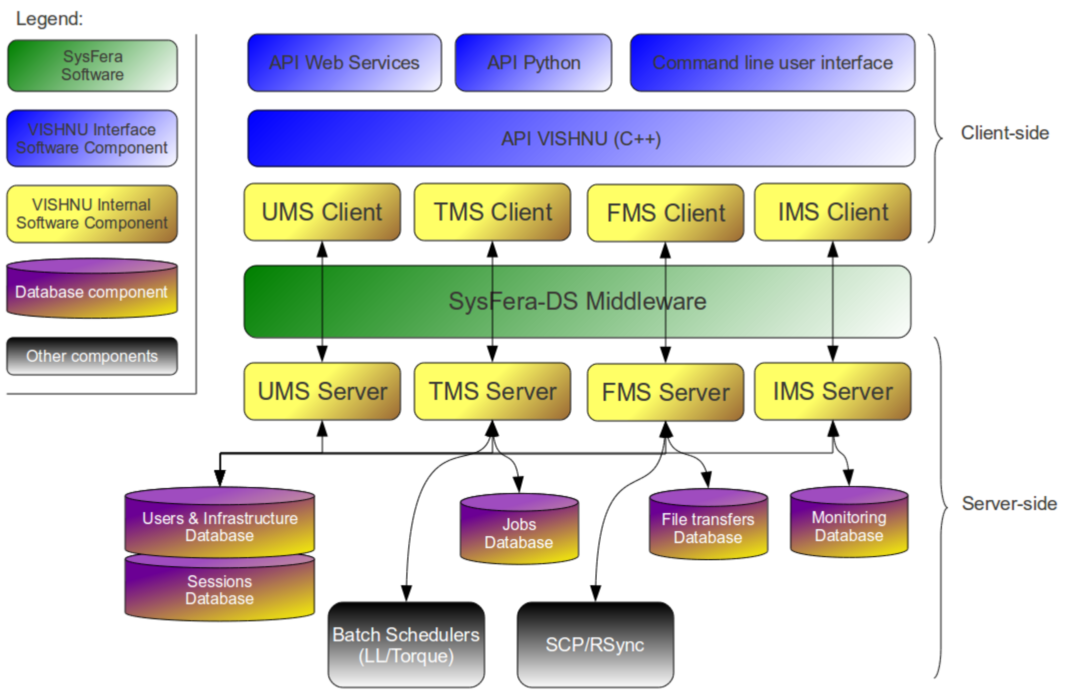

Figure 3.1: VISHNU Functional architecture diagram

3.3 Database description

3.3.1 Functional view

The following database components are used by the VISHNU system to store persistent data and provide historical data to the users:

- **Users and infrastructure database**: contains all data related to users, user configuration and the VISHNU managed infrastructure information.
- **Sessions database**: contains all data related to user sessions and the history of commands sent during the sessions.
- **Jobs database**: contains meta-information about the jobs submitted on the computing servers.
- **File transfers database**: contains meta-information about the file transfers initiated by users of VISHNU between systems that belong to the VISHNU Managed Infrastructure.
- **Monitoring database**: contains historical data about the status of systems that belong to the VISHNU Managed Infrastructure. The status includes system hardware information (cpu usage, disk usage, memory usage, etc.).

3.3.2 Logical view

The Figure 3.2 describes all the tables of the VISHNU database with their relationships.

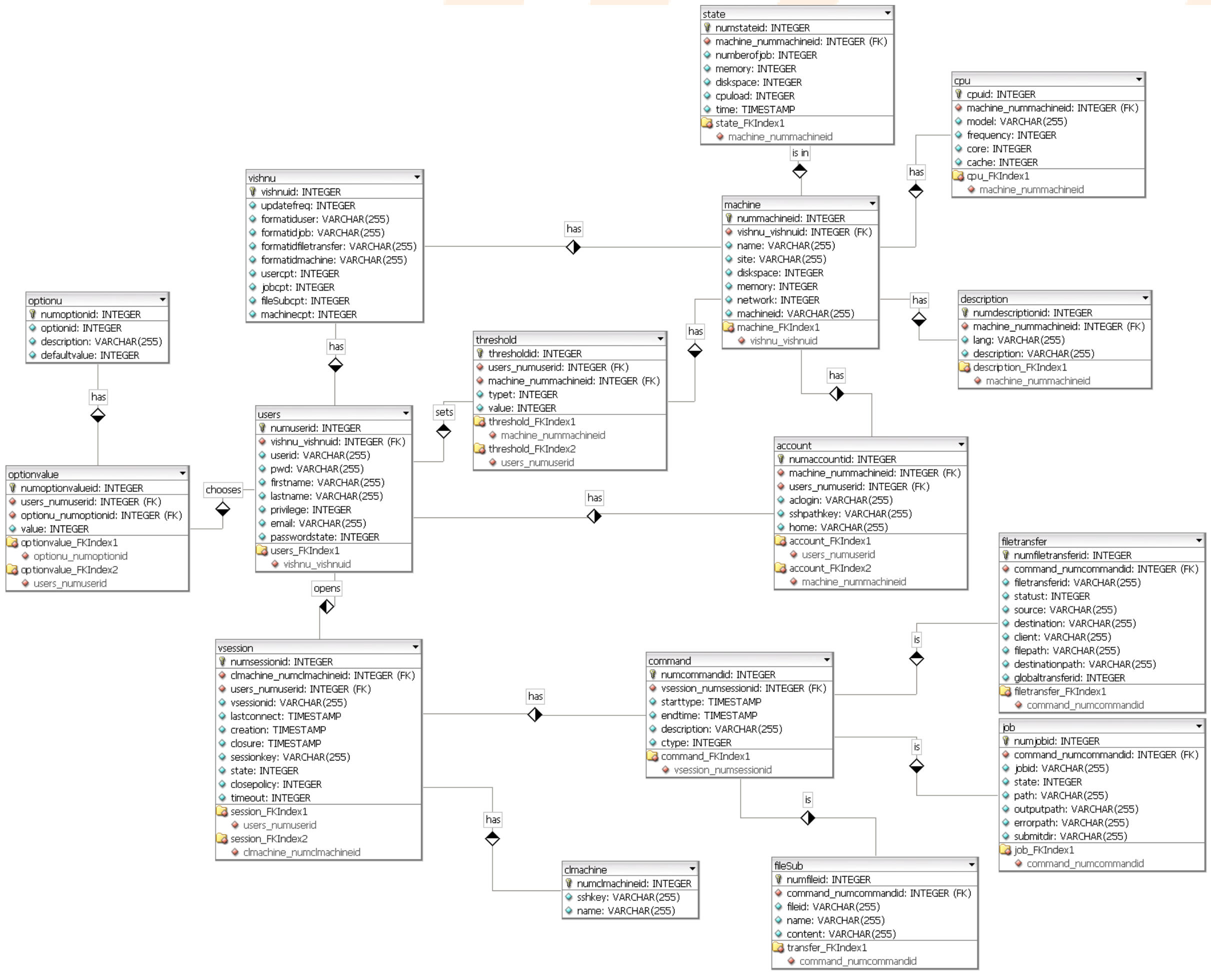

Figure 3.2: VISHNU Database diagram

3.4 Data flows

The Table 3.1 lists all the VISHNU data flows between the different functional elements of the architecture. Please note that the data flows involving the ‘Forwarder’ component are used only in the case of a deployment that requires crossing firewalls between different network areas (see chapter 4).

The bandwidth information provided in the table is an average estimated size for one standard user request sent by the client to the VISHNU system. The bandwidth requirement can be obtained by multiplying this value by the request rate. A ‘standard user request’ is a request that returns data to the user about VISHNU managed objects (users, sessions, commands, jobs, file transfers, accounts) considering that the size of this data set is not more than 10KBytes.

The **NVSS** number represents the number of VISHNU Services Servers in the VISHNU System. Please note that this is the number of logical servers not the number of physical servers.

The **aggregation** term is used when the bandwidth of the flow is the aggregation of the bandwidth of all flows that are tunneled through that flow.

The **CORBA** protocol ports: CORBA uses the IIOP (Internet Inter-ORB Protocol) on port 2809/tcp by default for omniNames and random user tcp ports.

ID	Source	Destination	Protocol	Bandwidth	Description
CMA	Client system	VISHNU MA	CORBA	1kB/request	SysFera-DS Request
COS	Client system	omniNames server	CORBA	1kB/request	SysFera-DS Request
WMA	Web server	VISHNU MA	CORBA	1kB/request	SysFera-DS Request
CWS	Client system	Web server	HTTP (80/tcp)	2kB/request	Web service request
OSS	omniNames server	VISHNU VSS	CORBA	1kB*NVSS /request	Corba request
CVS	Client system	VISHNU VSS	CORBA	15kB/request	SysFera-DS Solve
WVS	Web server	VISHNU VSS	CORBA	15kB/request	SysFera-DS Solve
MAS	VISHNU MA	VISHNU VSS	CORBA	1kB*NVSS /request	SysFera-DS Submit
VSD	VISHNU VSS	Database server	MySQL (3306/tcp) or PostgreSQL (5432/tcp)	11kB/request	Database connection
VDT	VISHNU VSS	VISHNU VSS	CORBA	TBC (<1kB /request)	Data transfer between servers
VFT	VISHNU Machine	VISHNU Machine	SSH (22/tcp)	depends on file size	File transfer between machines
UFT	Client system or Web server	VISHNU Machine	SSH (22/tcp)	depends on file size	Upload file
DFT	VISHNU Machine	Client system or Web server	SSH (22/tcp)	depends on file size	Download file
CFT	VISHNU VSS	VISHNU Machine or Client system or Web server	SSH (22/tcp)	1kB/request	Remote file command
VSE	VISHNU VSS	SMTP server	SMTP (25/tcp)	<10kB per message	Email to users or administrators
ADM	Admin system (VAS)	VISHNU VSS and VBS	SSH (22/tcp)	<1kB/s	Control of VISHNU processes
FWD	Forwarder	Forwarder	SSH (22/tcp)	aggregation	CORBA Tunnel
CWD	Client system or Web server	Forwarder	CORBA	aggregation	CORBA call through forwarder
SWD	VISHNU VSS or VBS	Forwarder	CORBA	aggregation	CORBA call through forwarder

Table 3.1: VISHNU Data Flows

3.5 Software architecture diagram

The Figure 3.3 describes the different software components of VISHNU with their dependencies.

This diagram is a UML component diagram:

- A component can be included inside another parent component (e.g. a package).
- A component can contain internal classifiers that implement the interface provided by the component (internal classifiers are shown by a box without the component icon).
- A dotted line arrow represent a dependency between two components.
- A circle represents a provided interface and is connected to the component that provides it.
- A semi-circle represents a required interface and is connected to the component that requires it.

The following terms and abbreviations are used in the Figure 3.3:

- **CPP**: C++
- **Py**: Python
- **Proxy**: Generic component type described in the ‘proxy pattern’ which is used when a class acts as an interface to another class.
- **SeD**: Server Daemon
- **WS**: Web Services package

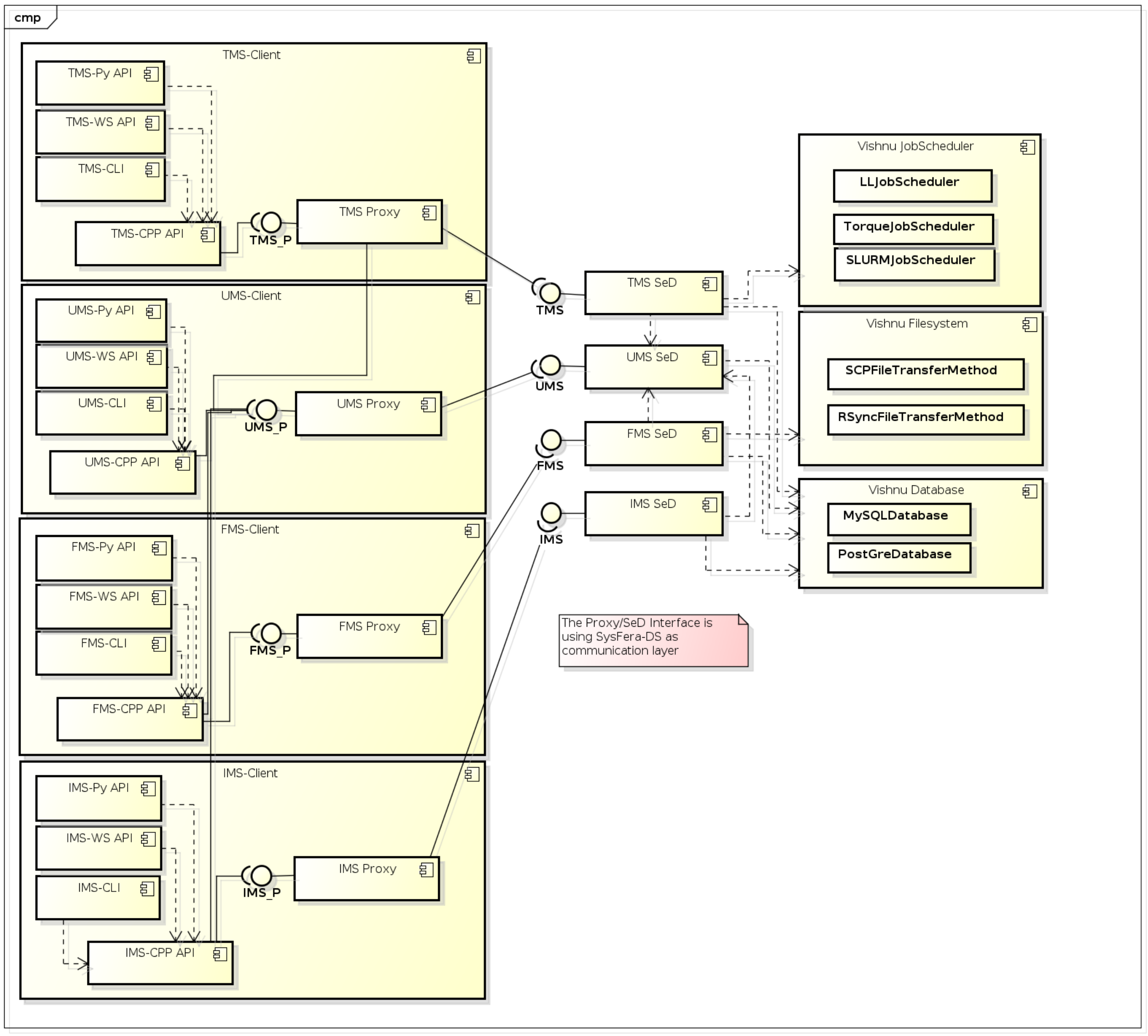

Figure 3.3: VISHNU Software architecture diagram

Chapter 4

Technical architecture

This chapter describes the deployment aspects of a VISHNU System. The first section presents a typical deployment of VISHNU on a single network. The second section presents a more complex deployment of VISHNU where the systems involved are located on different networks isolated by firewalls.

4.1 Technical architecture diagram

The Figure 4.1 shows a VISHNU system deployed on a set of hosts where each class of VSS and VBS is running on one dedicated host and where all clients and servers can connect to each other without crossing network firewalls. The SysFera-DS middleware uses the CORBA Bus for communication between the client systems and the MasterAgent (MA) server or a VSS, and between the MA server and a VSS. All VSS connect directly to the Database Server and can communicate together through CORBA or transfer files through SSH (not shown on the diagram).

During the startup or termination phases of VISHNU, the VISHNU Admin System (VAS) uses SSH connections to all the VBS and VSS hosts to launch the VISHNU internal processes. These connections are not shown on the diagram for better readability. These connections are all using the VISHNU User SSH keys.

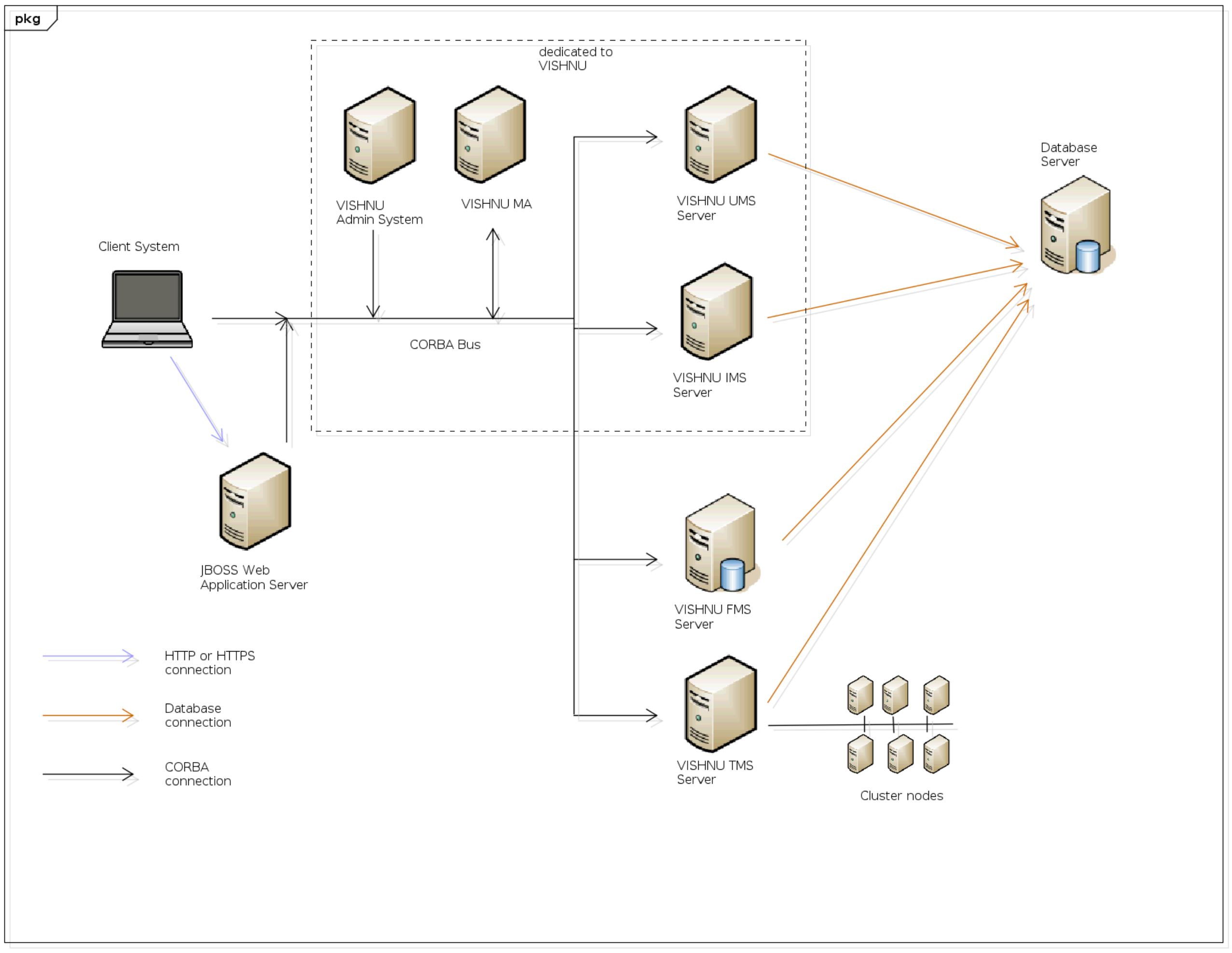

Figure 4.1: VISHNU Technical architecture diagram

4.2 Technical architecture with network firewalls

The Figure 4.2 shows a VISHNU system deployed on a set of hosts located on different networks:

- **Network A**: contains a client system (user laptop) and a SysFera-DS Forwarder host.
- **Network B**: contains VISHNU VBSs (with one hosting a SysFera-DS Forwarder). This network could also contain some VSS servers.
- **Network C**: contains the Web server, the Database Server and a SysFera-DS Forwarder host.
- **Network D**: contains a VSS (that could be a UMS, TMS, IMS or FMS Server) and a SysFera-DS Forwarder on the same host.

In this architecture, all inter-network communications are using the SSH protocol or the HTTP protocol (for client to web server).

The role of the SysFera-DS Forwarder is to allow CORBA communications (described in §4.1) across network firewalls. A SysFera-DS Forwarder agent must be running on both sides of the firewall and be accessible by other hosts on the same network. Within one network, all communications are as described in §4.1.

The startup and termination phases are identical to the architecture of §4.1., the SysFera-DS Forwarders being started and managed by the VISHNU Admin Server.

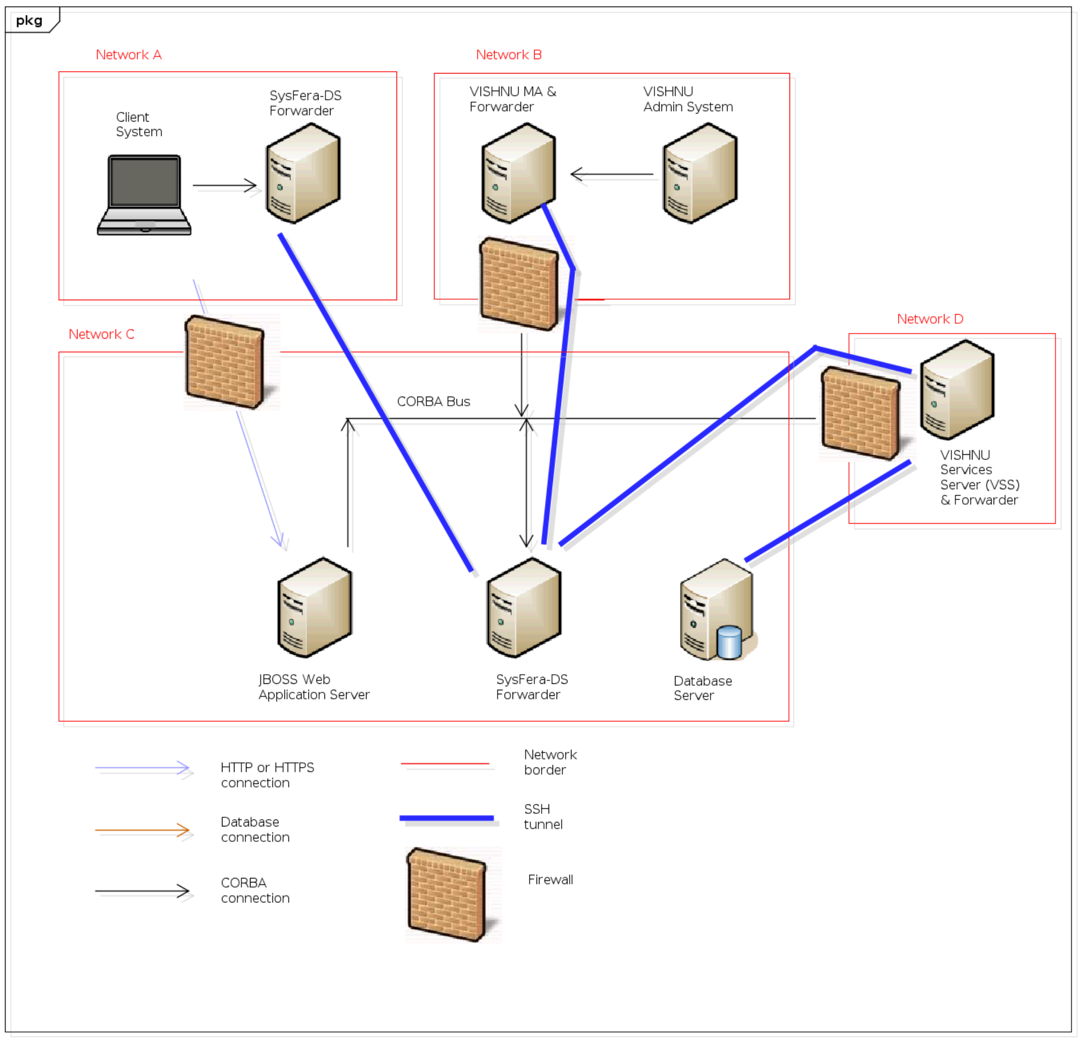

Figure 4.2: VISHNU Technical architecture with firewalls diagram

Chapter 5

VISHNU modules architecture

This chapter contains details of the VISHNU architecture for some VISHNU modules.

5.1 TMS module details

This section presents the specificities of TMS (Tasks Management Service) services architecture and the model of deployment for this module.

5.1.1 TMS submit job call sequence

Among the services provided by the TMS module we present the call sequence for the **submit job function.** For the specifications of this service, see the use case T1.CP1.

The following steps and diagrams represent the different steps of a job submission through vishnu

- A TMS server must be running on each VISHNU Machine where a batch scheduler is installed.
- The user must open the VISHNU session before using the VISHNU submit job function.
- After authentification of the user, a valid VISHNU session key is created, the submit function uses this session key to submit a job through the VISHNU system.
- In addition of the session key value, the VISHNU submit function takes as arguments the identifier of the machine on which the job must be submitted, the path of the script containing the job characteristics and the possible options.
- The machineId must be known to the VISHNU system (i.e. it has been registered by a VISHNU Administrator).
- The user who wants to submit a job must have a **VISHNU local account** configured on the machine. A local account on a machine contains the unix account details of the VISHNU user on the corresponding host (user login, home directory path, ssh key path).
- To submit a job the VISHNU submit function program contacts the Sysfera-DS agent to get the reference of the machine on which the job will be submitted.
- The submit function program contacts the selected machine and sends all input data (sessionKey, machineId, scriptPath and options) and waits for the output values : a valid output valid or a VISHNU exception. The machineId is valid if it exists in the vishnu Data Base and if the submit job owner has an account on this machine.
- The TMS server launched on the selected machine receives the input values, checks the validity of the sessionKey and machineId by contacting the VISHNU Database.
- If the two values are valid, the TMS server calls the underlying batch scheduler API submit function (llsubmit(...) for the LoadLeveler scheduler, pbs_submit(...) for Torque scheduler, slurm_submit_batch_job(...) for SLURM scheduler).

- If the job was performed successfully, the TMS creates a vishnu jobId and sends it back to the user.

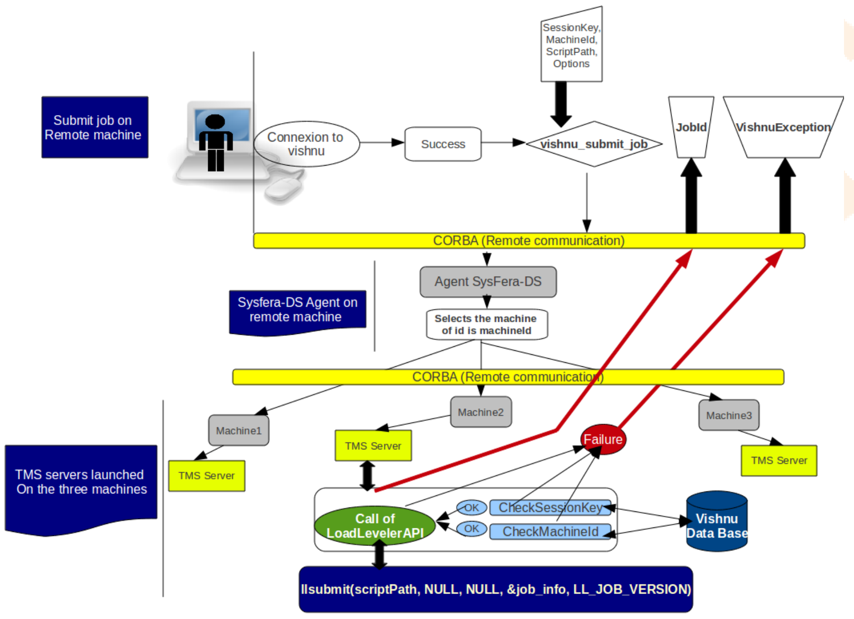
Submit job on Remote machine
Connexion to vishnu
Success
SessionKey, MachineId, ScriptPath, Options
vishnu_submit_job
JobId
VishnuException
CORBA (Remote communication)
Agent SysFera-DS
Sysfera-DS Agent on remote machine
Selects the machine of id is machineId
CORBA (Remote communication)
Machine1
Machine2
Machine3
TMS Server
TMS Server
TMS Server
Failure
TMS servers launched On the three machines
Call of LoadLevelerAPI
OK
OK
CheckSessionKey
CheckMachineId
Vishnu Data Base
llsubmit(scriptPath, NULL, NULL, &job_info, LL_JOB_VERSION)

Figure 5.1: TMS job submission steps

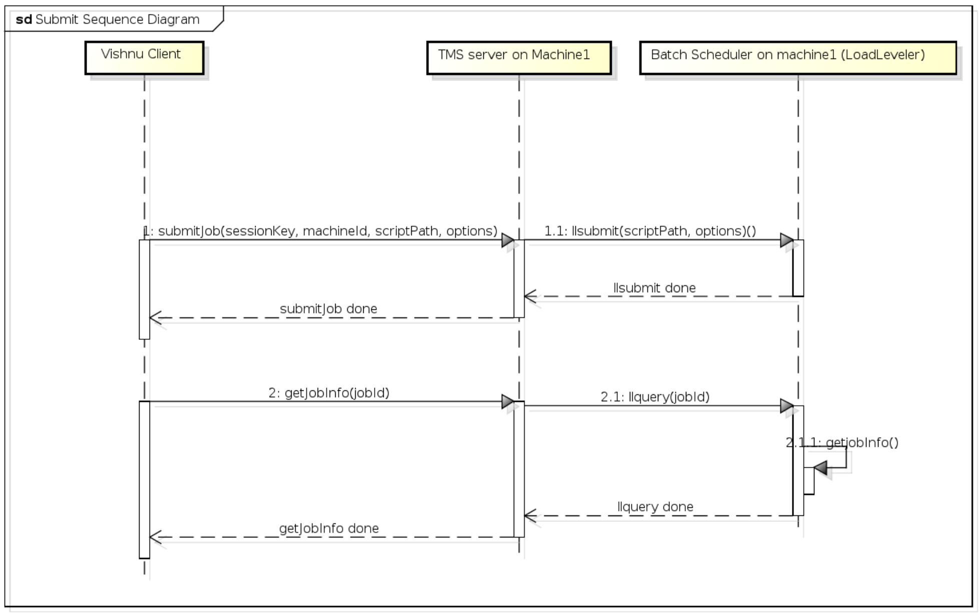
sd Submit Sequence Diagram
Vishnu Client
TMS server on Machine1
Batch Scheduler on machine1 (LoadLeveler)
1: submitJob(sessionKey, machineId, scriptPath, options)
1.1: llsubmit(scriptPath, options)()
llsubmit done
submitJob done
2: getJobInfo(jobId)
2.1: llquery(jobId)
2.1.1: getJobInfo()
llquery done
getJobInfo done

Figure 5.2: TMS submit job sequence diagram

5.1.2 TMS server deployment

The following diagram shows the relationships and their cardinalities between the different hosts involved in the deployment of the VISHNU TMS module. Several TMS Servers can be deployed within a single VISHNU infrastructure and each will provide the same set of services for a given batch scheduler. The TMS Servers will share the same VISHNU database where all logs and dynamical information will be stored.

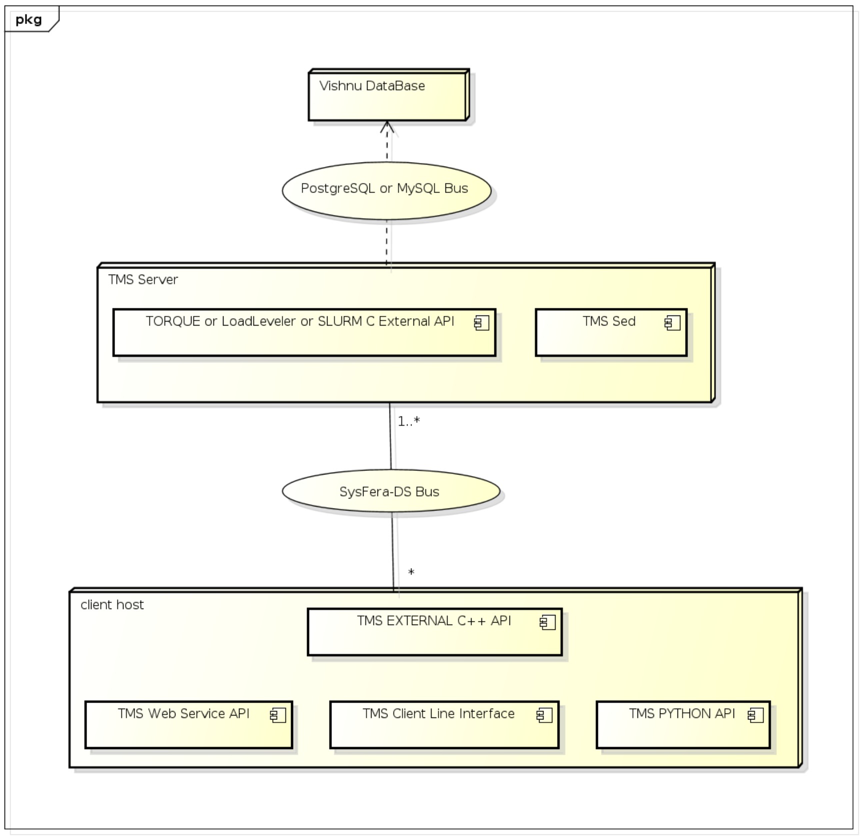

Figure 5.3: TMS deployment diagram

5.2 FMS module details

This section presents the specificities of FMS (File Management Service) services architecture and the model of deployment for this module.

5.2.1 FMS service call sequence

Among the services provided by the FMS module we present the call sequence for three of the most representative ones: a **file copy** from "MachineA" to "MachineB" with the synchronous and asynchronous variants, and the service for file information retrieval. For the specifications of these services, see the use cases F2.CP1, F2.CP2 and F1.DI5 in [D1.1a].

In the following diagrams, the objects represents different processes that may be running on different hosts. The generic names "MachineA" and "MachineB" could be mapped to either a VISHNU Machine or to the Client host itself. The constraints on a given host to be "MachineA" or "MachineB" are the following:

- A ssh server must be running on the host.
- The programs **scp**, **rsync**, **ls**, **rm**, **head**, **tail**, **mkdir**, **chgrp**, **cat** must be available on the host.
- The host must be known to the VISHNU system as either a **Machine** (i.e. it has been registered by a VISHNU Administrator) or a **Client** (i.e it is the host from where the request is sent).

- The VISHNU user who is requesting the file transfer must have a **VISHNU local account** configured on both machines except if the machine is the client host itself. A local account on a machine contains the unix account details of the VISHNU user on the corresponding host (user login, home directory path, ssh key path).
- The ssh configuration on both hosts corresponding to "MachineA" and "MachineB" *and the network* between these hosts must allow ssh connections:
  - from the FMS Server host to "MachineA" and to "MachineB".
  - from "MachineA" to "MachineB".

The call sequence is designed around the usage of the **scp** and **rsync** command-line programs according to the project requirements. Therefore the FMS Server software will use system ""exec"" calls to launch file transfers or other system calls related to file management (chmod, ls, etc.). This solution allows access to many hosts from a single FMS server (see deployment paragraph below), but reduces the interoperability as it depends on unix/linux-specific features.

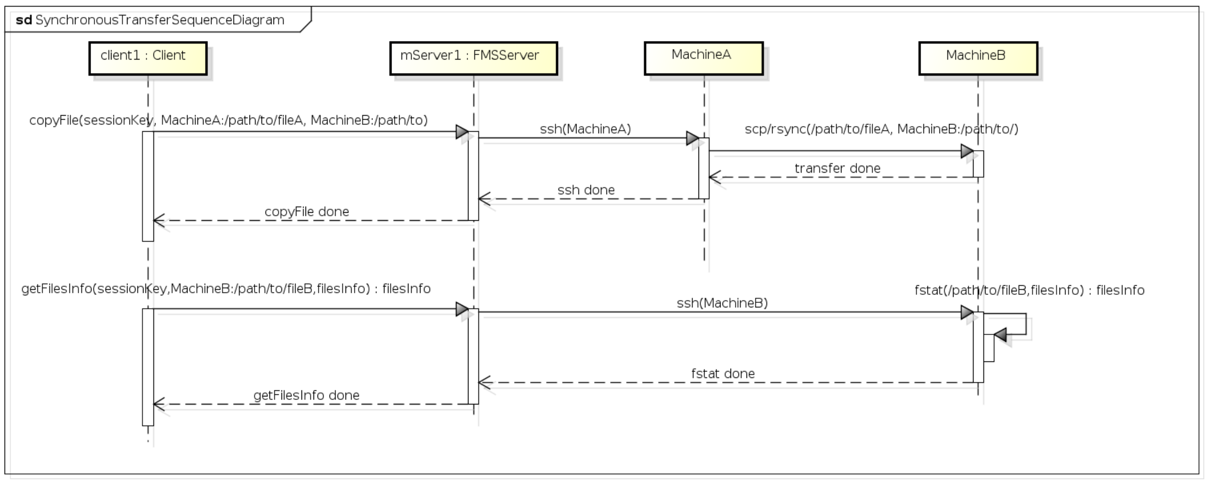
sd SynchronousTransferSequenceDiagram
client1 : Client
mServer1 : FMSServer
MachineA
MachineB
copyFile(sessionKey, MachineA:/path/to/fileA, MachineB:/path/to)
ssh(MachineA)
scp/rsync(/path/to/fileA, MachineB:/path/to/)
transfer done
ssh done
copyFile done
getFilesInfo(sessionKey,MachineB:/path/to/fileB,filesInfo) : filesInfo
ssh(MachineB)
fstat(/path/to/fileB,filesInfo) : filesInfo
fstat done
getFilesInfo done

Figure 5.4: FMS synchronous transfer sequence diagram

In the case of an asynchronous file transfer, the call from client1 to mServer1 terminates as soon as the file transfer is initiated from MachineA. The state of an asynchronous file transfer can be requested using another FMS Service, and the transfer can be stopped asynchronously as shown on the diagram.

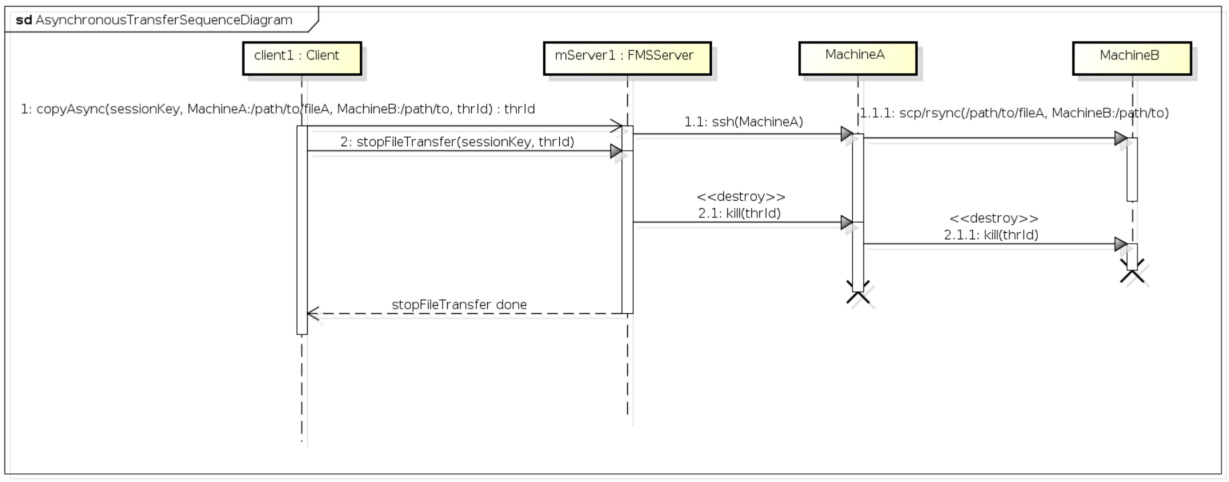
sd AsynchronousTransferSequenceDiagram
client1 : Client
mServer1 : FMSServer
MachineA
MachineB
1: copyAsync(sessionKey, MachineA:/path/to/fileA, MachineB:/path/to, thrId) : thrId
1.1: ssh(MachineA)
1.1.1: scp/rsync(/path/to/fileA, MachineB:/path/to)
2: stopFileTransfer(sessionKey, thrId)
<<destroy>>
2.1: kill(thrId)
<<destroy>>
2.1.1: kill(thrId)
stopFileTransfer done

Figure 5.5: FMS asynchronous transfer sequence diagram

5.2.2 FMS server deployment

The following diagram shows the relationships and their cardinalities between the different hosts involved in the deployment of the VISHNU FMS module. Several FMS Servers can be deployed within a single VISHNU infrastructure and will provide the same services for all the machines configured in VISHNU. The FMS Servers will share the same VISHNU database where all logs and dynamical information will be stored.

Note: the choice of a particular FMS Server will be done automatically by VISHNU and will be transparent for the user.

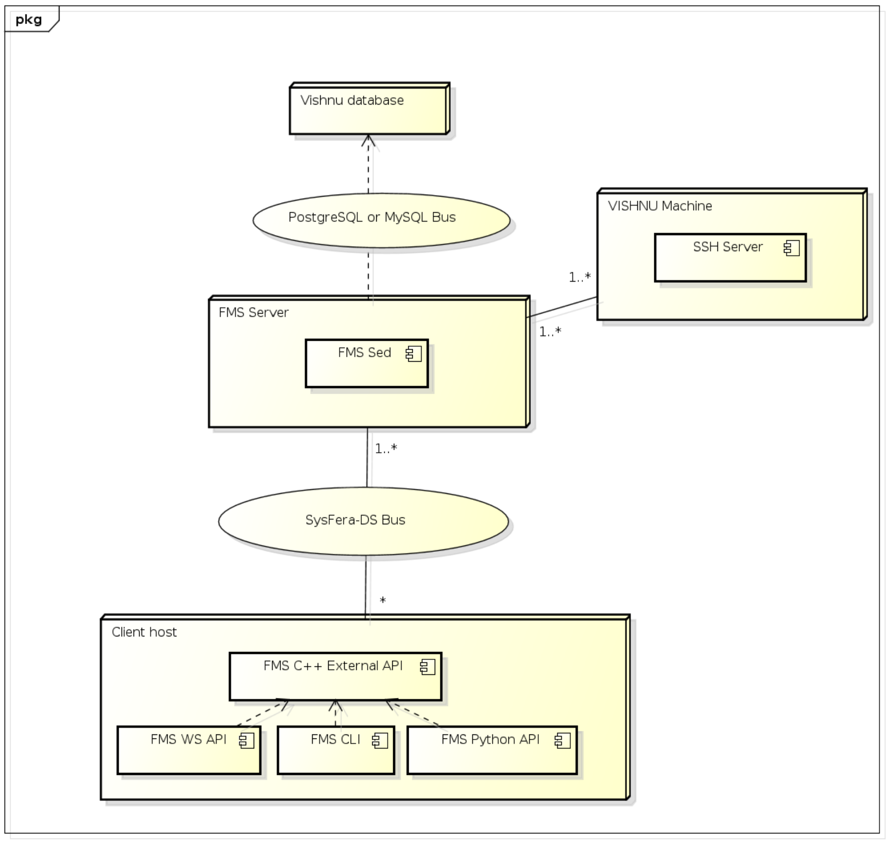

Figure 5.6: FMS deployment diagram