

CRESCO - SPI.2

MAGO

Deliverable 3° obiettivo realizzativo intermedio

**Metodologia di interazione tra il database ed il gestore di
risorse LSF**

**Relativo al contratto tra ENEA e CRIAI avente per
oggetto: “Analisi e Realizzazione di tool innovativi a
supporto delle funzionalità GRID” stipulato in data 22
Dicembre 2006**

Indice dei contenuti

Indice dei contenuti.....	ii
Indice delle figure.....	iii
Introduzione.....	4
Scopo e Campo di applicazione.....	4
Architettura del sistema.....	6
Funzionalità.....	8
Implementazione del sistema MAGO.....	11
Magoupdate.sh.....	11
MMasterLoader	14
MsiteProvider.....	15
SocketSmista.pl.....	16
ServerDB.pl.....	16
Struttura Componenti “core”.....	17
Mago-Top.....	18
Application Server.....	19
L'istanza tecnologica dell'Application Server: JBoss.....	21
Integrazione tra JBoss e AFS-Enea.....	23
Casi d'uso dello scenario.....	27
Modifiche effettuate sulla macchina virtuale.....	27

Indice delle figure

Figura 1 Architettura del sistema MAGO.....	6
Figura 2 Relazioni tra i componenti del sistema MAGO.....	10
Figura 3 Relazioni e modalità di comunicazione del sistema MAGO.....	17
Figura 4 Ambiente mago sul server Mago-Top.....	18
Figura 5 La struttura della directory di JBoss Application Server.....	22
Figura 6 La struttura della directory top-level di JBoss.....	22
Figura 7 La struttura della directory di configurazione dell'Application Server JBoss.....	23
Figura 8 Modifiche effettuate in jboss-service.xml.....	24
Figura 9 Struttura della pubblicazione web in AFS-ENEA. .	26
Figura 10 Caso d'uso Pubblicazione di Contenuti Web in AFS-ENEA.....	27
Figura 11 Caso d'uso Navigazione nei contenuti web di ENEA-GRID.....	27

Introduzione

In questo documento si descrivono le caratteristiche del sistema denominato genericamente “MAGO: Monitoring All GRID Objects”. Il sistema completo che comprende al suo interno varie funzionalità:

1. La memorizzazione di tutti i dati riguardanti le risorse disponibili nella rete di calcolo ENEA-GRID.
2. Il monitoraggio di tutte le risorse della rete di calcolo ENEA- GRID.
3. La comunicazione tra il database suddetto ed il sistema di ENEA-GRID.

Il focus di questo documento è centrato sul terzo punto, poiché i precedenti punti sono già stati affrontati nelle relative documentazioni di rapporto.

Sarà affrontata la sezione core del sistema, in particolare saranno trattati l'insieme dei demoni che sono necessari al trasferimento delle informazioni lungo la gerarchia architetturale del progetto in questione.

Scopo e Campo di applicazione

Il progetto PON Cresco ha come obiettivo la realizzazione, presso il Centro Ricerche ENEA di Portici (NA), di un importante Polo di calcolo multidisciplinare per lo studio dei sistemi complessi di natura biologica e tecnologica, collegato con facilities di elaborazione, archiviazione e visualizzazione 3D di dati ubicate presso i Centri ENEA di Brindisi e Trisaia (MT).

Di seguito si descrive il capitolo I.2.4 della attività programmatica di CRESCO con titolo “Analisi e Realizzazione di tool innovativi a

supporto delle funzionalità GRID” al fine di esporre il motivo dell'utilizzo di Ganglia e le potenzialità che se ne intendono sfruttare.

Il capitolo I.2.4 della attività programmatica di CRESCO con titolo “Analisi e Realizzazione di tool innovativi a supporto delle funzionalità GRID” si rivolge esplicitamente ai temi del “Monitoring”, “Resource Discovery”, “Accounting” e “Semantic & Knowledge GRID”.

L'obiettivo è estendere in modo innovativo le potenzialità della attuale infrastruttura ENEA-GRID, all'interno della quale viene collocata la realizzazione del progetto CRESCO. A tal fine si assumono come vincoli i componenti costitutivi di base di ENEA GRID, ovvero il sistema di gestione delle risorse basato sul prodotto LSF Multicluster e il file system distribuito OpenAFS.

Questa attività specifica dovrà portare ad arricchire in modo innovativo le funzionalità di ENEA-GRID in modo compatibile con l'infrastruttura esistente, estendendole in particolare in quelle aree che le componenti attualmente utilizzate non riescono a coprire in modo pienamente soddisfacente.

Il progetto si basa su GANGLIA e si propone di integrare le informazioni fornite di default da ganglia con informazioni su demoni, servizi e software che sono specifici delle infrastruttura di ENEA-GRID. L'insieme di tutte le informazioni viene centralizzato su un DB, implementato in MySQL. L'implementazione fa uso di AFS, componente di base di ENEA-GRID per la distribuzione delle informazioni statiche di configurazione ma identifica in modo esplicito tale approccio in modo che sia possibile implementare le stesse funzionalità attraverso altri metodi in ambienti che non fanno uso di AFS.

Architettura del sistema

Il sistema si compone di 3 livelli distinti: Server Centrale, Server di Sito e Nodo.

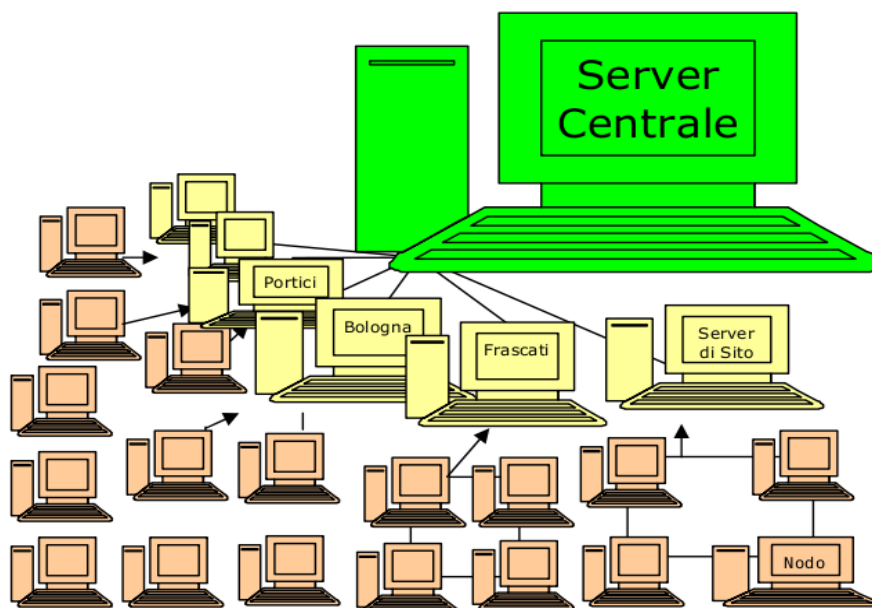


Figura 1 Architettura del sistema MAGO

Come si evince, questo modello architetturale è del tutto centralizzato. Ciò si contrappone ad una delle peculiarità dei sistemi GRID. Tuttavia questo modello gerarchico è relativo esclusivamente al sistema di monitoraggio, in pratica è una visione indipendente da come realmente vengono usati i nodi di calcolo per compiere operazioni.

Tale approccio, si sottolinea, è giustificato dall'obiettivo di rendere la configurazione dell'intero sistema quanto più centralizzata possibile, senza dover fare eccessivi interventi al livello di singolo nodo.

Come si nota l'infrastruttura può essere suddivisa in livelli: Nodo *layer* "2", Server di sito *layer* "1", Server centrale *layer* "0".

Il sistema MAGO implementa tre funzionalità principali: la *memorizzazione* dati di tutti le risorse, il *monitoraggio* di ogni singolo nodo, *l'utilizzo e l'analisi* del database mediante altri tool.

Il raggiungimento di questi obiettivi prevede un discorso abbastanza articolato, che richiede una prima descrizione ad un livello più astratto per poi scendere nelle particolarità di ogni proprietà offerta dal sistema al fine di permettere al lettore una semplice comprensione della modalità di esecuzione.

Per quanto detto procediamo ad una panoramica delle funzionalità del sistema a partire da quelle presenti sul server centrale :

- Un database centrale contenente le risorse dell'intera rete ENEA-GRID
- Raccolta dei dati monitorati in tempo reale di tutte le risorse
- Formattazione dei dati per inserirli correttamente nelle tabelle predisposte sul database
- Inserimento/Aggiornamento dei dati nel database centrale

Funzionalità offerte su tutti i server di sito:

- Raccolta e propagazione dei dati provenienti direttamente dal monitoraggio dalle sottoreti
- Inserimento di nuove informazioni nell'insieme di dati da inoltrare
- Sincronizzazione e comunicazione con il server centrale

Funzionalità su tutti i nodi:

- Installazione e configurazione centralizzata del sistema
- Sonda di prelievo informazione e replica su nodi e server di sito
- Esecuzione degli Script di METRICA
- Inserimento dei risultati degli script di METRICA nei file xml di ganglia

Col termine METRICA si intende la singola richiesta tra:

- Test di ciascuno specifico software che restituisce lo stato dell'eseguibile sul nodo

- Test di ciascuno specifico software che restituisce la versione installata sul nodo
- Test di controllo della corretta esecuzione di un dato demone sul nodo
- Test di controllo di erogazione di un servizio su nodi dedicati all'erogazione di servizi generici verso altri nodi

I precedenti esempi derivano dalle richieste espresse dal committente del progetto, tali script sono soggetti ad un unico vincolo : “Devono ritornare un singolo valore”.

L'obiettivo di fornire le precedenti funzionalità implica delle componenti presenti sui vari elaboratori del sistema atti a svolgere le suddette caratteristiche.

Funzionalità

Vediamo ora quali componenti svolgono quali servizi su ogni livello dell'infrastruttura:

Server Centrale: presente al centro ENEA di Portici, è una macchina dedicata al sistema di monitoraggio MAGO, facile da raggiungere per gli sviluppatori di tale sistema appartenenti al CRIAI, dotata dei seguenti componenti:

- Ambiente locale mago: */usr/local/mago*
- Server di monitoraggio *gmetad*
- Database delle Risorse mysql: *Mago.sql*
- Componente di caricamento del database: ***MMasterLoader***

Server di Sito: macchina locata presso ogni centro ENEA, dedicata al sistema MAGO, che permette di monitorare tutti i nodi appartenenti alla sua sottorete, dotato dei seguenti componenti:

- Ambiente locale mago: */usr/local/mago*
- Server di monitoraggio *gmetad* sempre attivo

- Client di monitoraggio *gmond* sempre attivo
- Componente salvataggio temporaneo di sito: ***MsiteLoader***
- Componente di fornitura dei dati verso il Server centrale:
SiteProvider

Nodo: rappresenta un determinato elaboratore univocamente determinato all'interno dell'ENEA-GRID, e spazialmente locato presso uno dei centri ENEA presenti in Italia, avente i seguenti componenti:

- Ambiente locale MAGO */usr/local/mago*
- Elemento di installazione esecuzione e configurazione:
magoupdate.sh
- Client di monitoraggio *gmond* come servizio sempre attivo
- Metriche aggiuntive che vengono raccolte tramite l'esecuzione di script eseguite dal crontab di un utente locale dedicato (“mago”).

Di seguito è raffigurata l'attuale architettura del sistema MAGO:

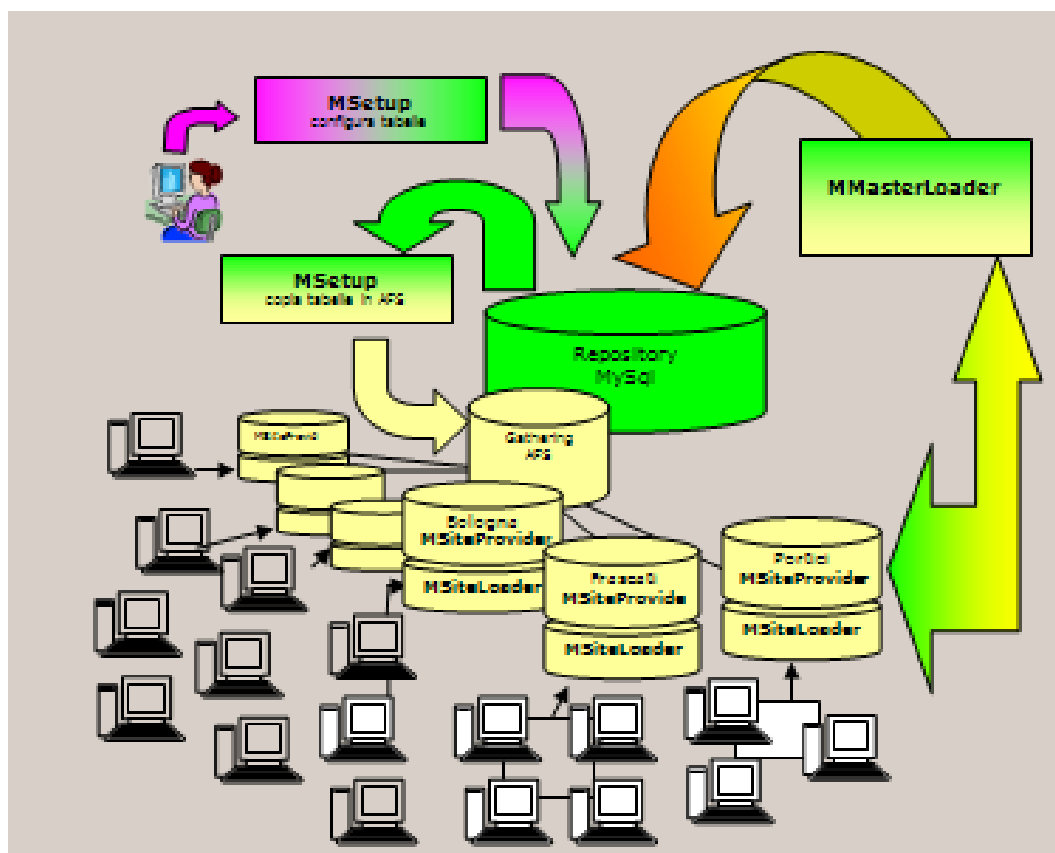


Figura 2 Relazioni tra i componenti del sistema MAGO

Implementazione del sistema MAGO

Di seguito è riportata una lista di componenti necessari al funzionamento del sistema MAGO.

I componenti totali del sistema sono:

- **magosql**
- **Msetup**
- **MmasterLoader--core--**
- **MsiteLoader --core--**
- **MsiteProvider --core--**
- **Magoupdate.sh--core--**
- **MMonitor**
- **MEnquiry**

I componenti trattati in dettaglio sono quelli contraddistinti dalla parola "core", stando ad indicare che rappresentano il cuore dell'intero sistema.

In questo documento è presente una guida descrittiva dell'applicazione server JBOSS, strumento indispensabile per MAGO, e come vedremo per altre applicazioni.

Il sistema prevede un'installazione centralizzata sui nodi, con installazione in loco dell'ambiente mago e verifica dei vincoli per permettere il funzionamento del sistema. Tale compito è esplicato dal seguente script.

Magoupdate.sh

Tale script risiede in una apposita directory del file system distribuito AFS, discusso nel II capitolo di questo lavoro:

`/afs/enea.it/project/cresco/LA1/cresco_sp12_criai/soft/mago/magoupdate.sh`

è uno script in bash shell che deve essere eseguito almeno una volta su ogni macchina che si intende inserire nel monitoraggio.

Prende in input uno di due parametri: *install* e *update*. L'esecuzione con l'argomento *update*

```
XXX:~$ magoupdate.sh update
```

Verifica se la macchina sul quale sta girando usa una versione obsoleta di se stesso: in caso affermativo la sovrascrive con la versione aggiornata residente in AFS e la esegue; in caso negativo la esegue.

Il controllo avviene sulla data di ultima modifica del file effettuato dalla funzione *Verify_up*

Copia da AFS i file specifici per il nodo, i quali risiedono sotto una directory chiamata con il nome dell'host dal quale viene eseguito. Tale directory viene eseguita in fase di configurazione, e verrà creata sotto il path:

```
/afs/enea.it/project/cresco/LA1/cresco_sp12_criai/soft/mago/hosts/nome_host
```

in tale modo da poter univocamente determinare la configurazione dell'host. Non a caso per arrivare da nodo a questo path, viene usato il ritorno del comando "*hostname*". Giunto alla directory tale script copia dei file:

crontab.list : file contenente delle stringhe settate secondo le convenzioni di crontab, contiene gli intervalli temporali con cui le metriche devono essere eseguite sul singolo host

metric.list : file contenente coppie di stringhe del tipo chiave, valore:
chiave=id_script identifica univocamente lo script all'interno del database centrale

valore=path_completo in AFS gestito associando un valore semantico alla directory, cioè `/afs/enea.it/system/mago/scripts` directory contenente tutti gli script di metrica

la cui sottodirectory /type [=software,daemon,service] identifica una classe sul quale ottenere informazione, la cui sottodirectory /group [=AFS,matlab,gcc,...] identifica un oggetto il cui file /nome_script_metrica è un soft-link alla versione corrente dello script.

Aggiorna il crontab locale dell' utente “mago”

Esegue gli script delle metriche relativi al nodo eincapsula tramite gmetric appartenente al tool ganlia il valore di ritorno dello script al blocco delle informazioni relative al nodo. Tali informazioni tramite gmond saranno sparse su tutti i nodi appartenenti alla sottorete, per poi su richiesta essere mandati al server di sito, il quale si occuperà di conservarli fin quando non saranno scritti nel database sul server centrale.

Eseguendolo con l'argomento install

```
XXX:~$ magoupdate.sh install
```

se non esiste crea e poi si sposta nella directory locale mago presente sul nodo in /usr/local/mago, copia l'ultima versione di magoupdate.sh e la esegue in modalità update.

Come conseguenza di tali funzioni si ottengono i seguenti risultati:

Esecuzione dei rispettivi script secondo la tempistica stabilita da crontab

Accodamento del risultato ottenuto da ciascuno script a gmond mediante gmetric.

Invio automatico, grazie all'esecuzione di gmond, del file XML di Ganglia sulla porta 8649

MMasterLoader

Questo componente del Server Centrale esplica le seguenti funzionalità:

Legge le tabelle di configurazione di Mago dal database per conoscere gli indirizzi dei Server di Sito

Contatta i Server di Sito e ottiene dai componenti MSiteProvider tutti i file di risposta disponibili

Formatta i dati nel formato corretto

Crea e salva in locale i file

in /usr/local/mago/XMLCENTRALSERVER

Ordina la cancellazione dei file di risposta correttamente caricati.

Rappresenta il cuore del sistema totale, in questa fase vengono definite una serie di caratteristiche fondamentali per un buon sistema di monitoraggio ad-hoc, come la sincronizzazione e il carico della rete.

Il modello utilizzato per lo scambio è Client/Server, MMasterLoader.c è il client, i server risiedono sui server di sito. Fa uso delle librerie mysql per compiere query sul database e ottenere gli ip dei server di sito.

Questo programma ottiene le informazioni relative ai nodi dai server di sito mediante una socket usando domini PF_INET. L'obiettivo è portare in locale file sensibili per il sistema, residenti sui server di sito garantendo l'integrità. Per quest'ultimo motivo è stato utilizzato il protocollo TCP, poiché prevede la ritrasmissione del pacchetto, senza tale vincolo era più logico utilizzare il protocollo UDP visto la grande quantità di informazione. La quantità di file da scambiare è molto elevata e avviene in maniera continua. Ottenute le informazioni dalla socket le scrive in un file, lo chiude, e poi invia una stringa sulla socket che rappresenta l'avvenuta memorizzazione. A tal punto il server può cancellare il file precedentemente inviato.

Carico della rete: Daver realizzato la sincronizzazione tra i componenti MMasterLoader.c MSiteLoader.c, in fase di test è stato notata il carico della rete nel mandare piccola quantità di file. I risultati non erano brillanti, dunque sorge un problema.

Se per una piccola quantità di file xml la rete usata come test subisce un grosso sforzo, quale sarà stato il risultato quando il sistema dovrà trasferire grossi numeri di pesanti xml? Da notare che il peso in KB dei file xml è direttamente proporzionale alla quantità di nodi appartenenti alla sottorete di cui il nodo fa parte. Senza grossi sforzi di calcolo si è notato che la rete poteva creare non pochi problemi al funzionamento di questo sistema. Durante la fase di realizzazione notando che i file xml non sono altro che file di testo semplici, e gli algoritmi di compressione utilizzati dalle bzlib2 applicati sui file di testo arrivano a diminuire il peso del file dell'80% il problema del carico della rete è stato risolto.

MsiteProvider

Questo componente di ciascun Server di Sito rappresenta il server, attraverso la socket tcp si sincronizza con il server e comincia a trasferire dei pacchetti compressi. Tale compressione è stata realizzata attraverso l'uso della libreria bzlib, accennata nel paragrafo precedente, per approfondimenti : <http://www.bzip.org/index.html>.

Il motivo di questa compressione deriva dalla sua potenza di comprimere il contenuto testuale dei file. Notando che il formato delle informazioni proveniente dalle sonde del tool di monitoraggio ganglia è XML, del tutto equivalente al formato testo, è stato intuitivo il risparmio del tempo di trasmissione su grandi quantità di dati.

Attualmente il componente pretende l'inserimento manuale dell'indirizzo del server.

In collaborazione con l'MMasterLoader esisto due componenti sviluppati in perl: ServerDB.pl e SocketSmista.pl. visti di seguito.

SocketSmista.pl

Questo componente funge sia da client che da server al fine di gestire i dati provenienti dal ParserGangliaXML, il quale come descritto precedentemente, preleva le informazioni dai file XML provenienti dalle sonde.

L'insieme di queste informazioni viene usato sia per gestire il modulo di inserimento nel database, sia per il modulo di gestione degli allarmi. Si preoccupa di creare diverse formattazioni di stringhe, per l'inserimento nel database realizzato dal componente ServerDB.pl descritto di seguito, e per la gestione degli allarmi realizzata da MMonitor appartenente alla sezione web dell'applicazione

ServerDB.pl

Questo componente è un server per il componente precedente, rimane in ascolto attendendo le informazioni contenenti le specifiche metriche dell'host di riferimento.

Il database come descritto nelle precedenti documenti di rapporto è in mysql.

Struttura Componenti “core”

Di seguito è riportata una figura intuitiva che rappresenta gli attuali componenti realizzati, e le loro modalità di iterazione, al fine di rendere semplici anche sviluppi futuri.

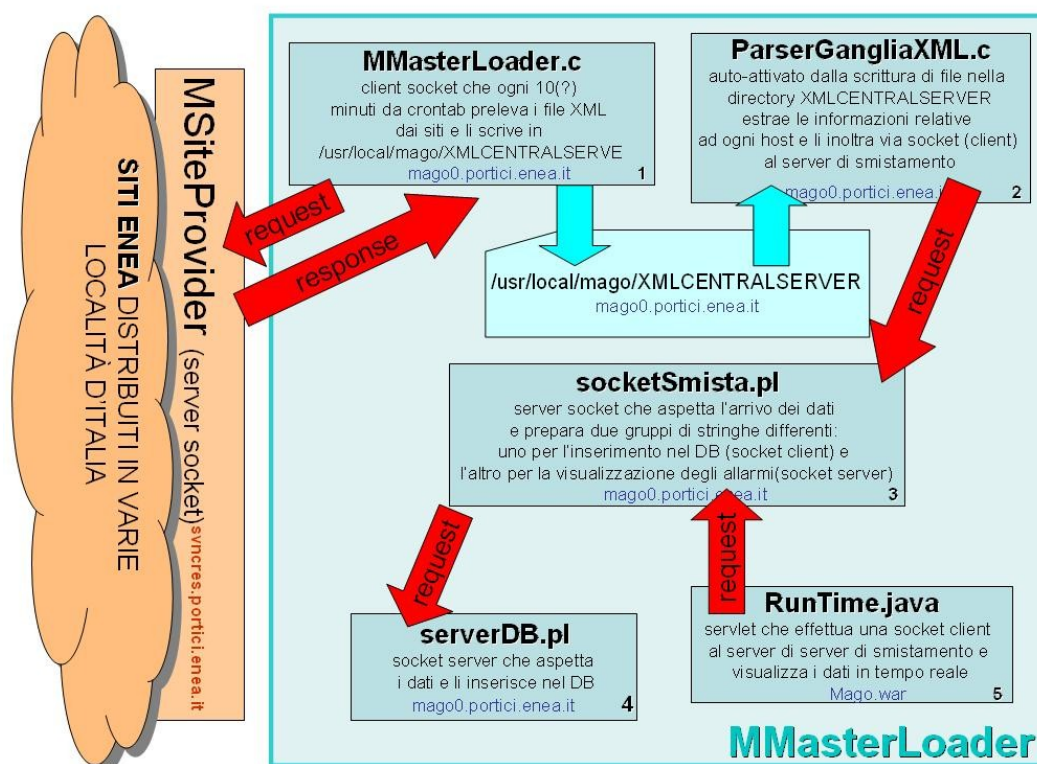


Figura 3 Relazioni e modalità di comunicazione del sistema MAGO

Di seguito è raffigurata l'ambiente e i relativi componenti sul server centrale, denominato mago-top.



Consorzio Campano di Ricerca per l'Informatica e
l'Automazione Industriale – S.c.a.r.l.



Ente per le Nuove tecnologie, l'Energia e l'Ambiente

Mago-Top

Rappresenta il server centrale, residente a portici opportunamente configurato dal CRIAI. Di seguito è riportato l'ambiente MAGO con i relativi componenti installati:

```
root@magotop: /usr/local/mago
File Modifica Visualizza Terminale Schede Ajuto
[root@magotop mago]# hostname
magotop.portici.enea.it
[root@magotop mago]# pwd
/usr/local/mago
[root@magotop mago]# tree
.
|-- XMLCENTRALSERVER
|   |-- XML1212596996986
|   |-- XML121259699742
|-- bin
|   |-- MMasterLoader
|   |-- ParserGangliaXML
|   |-- name.txt
|   |-- serverDB.pl
|   |-- socketSmista.pl
|   |-- valori.txt
|-- crontab.list
|-- doc
|-- magoupdate.sh
|-- metric.list
|-- src
|   |-- MMasterLoader.c
|   |-- ParserGangliaXML.c
|   |-- ParserGangliaXMLv.1.1.c
|   |-- mago_setup.SQL
|   |-- name.txt
|   |-- serverDB.pl
|   |-- socketSmista.pl
|   |-- valori.txt
4 directories, 19 files
[root@magotop mago]#
```

Figura 4 Ambiente mago sul server Mago-Top

Application Server

Un application server è un sistema [software](#) che fornisce l'[infrastruttura](#) e le funzionalità di supporto, sviluppo ed [esecuzione](#) di [applicazioni](#) e componenti [server side](#) in un contesto distribuito. Si tratta di un complesso di servizi orientati alla realizzazione di applicazioni per il [web](#), multilivello ed enterprise, con alto grado di complessità.

L'adozione di un application server offre particolari benefici soprattutto nei settori dello sviluppo, dell'esecuzione e della gestione integrata dei [sistemi](#).

I principali vantaggi possono essere così riassunti:

Semplificazione delle attività di sviluppo: gli application server creano un ambiente nel quale si possono utilizzare gli strumenti di sviluppo più diffusi sul mercato, consentendo di produrre e distribuire rapidamente applicazioni transazionali fortemente scalabili. In generale, questi ambienti comprendono [modelli](#) e strumenti di ausilio per sviluppare le applicazioni, riducendo i tempi di realizzazione e messa in esercizio dei programmi negli ambienti distribuiti.

Supporto di vari [linguaggi](#), strumenti e [piattaforme](#): in funzione dell'application server utilizzato, le applicazioni possono essere scritte nel linguaggio preferito dal programmatore.

Riusabilità del codice: la riusabilità del codice deriva sia dalla [programmazione orientata agli oggetti](#), spesso utilizzata in questi casi, sia dall'utilizzo dell'approccio di sviluppo a componenti. Una volta sviluppata la logica applicativa, essa può essere condivisa e riutilizzata.

Gestione delle [transazioni](#): l'application server facilita la gestione delle operazioni basate su transazioni, assicurando l'integrità transazionale e la gestione affidabile dei [back-end](#) multipli per le risorse e i [dati](#). Il sistema

di gestione delle transazioni gestisce le interazioni con i [database](#) e le funzioni di: [commit](#), [rollback](#) e [recovery](#).

Scalabilità: gli application server supportano il partizionamento delle applicazioni e la distribuzione in [rete](#) dei componenti. I sistemi [multiprocessore](#) e i [cluster](#) di application server assicurano la scalabilità necessaria a gestire anche un gran numero di utenti concorrenti.

Alte prestazioni: gli application server offrono caratteristiche architetturali che permettono di erogare elevate prestazioni. Tra esse: il [multithreading](#), il bilanciamento dinamico dei [carichi](#) di lavoro (load balancing), il caching e il [pooling](#) degli [oggetti](#) e delle [connessioni](#) ai database.

Estensibilità: l'architettura modulare degli application server e il supporto per i server e per i moduli applicativi, che possono essere caricati dinamicamente, consente alle aziende di estendere facilmente le funzionalità dei loro sistemi e delle relative applicazioni.

Robustezza: l'architettura basata sui componenti degli application server e il bilanciamento dinamico dei carichi assicurano l'alta [disponibilità](#) dei sistemi. I componenti del server e la logica applicativa possono essere riconfigurati, aggiunti o rimossi senza interruzioni nell'erogazione dei servizi agli utenti. Queste caratteristiche sono particolarmente importanti per garantire l'alta disponibilità del sistema, requisito necessario per il buon esito delle operazioni mission-critical aziendali.

Sicurezza: gli application server offrono funzioni specifiche di sicurezza [end-to-end](#). Esse sono necessarie per l'esecuzione delle applicazioni aziendali che richiedono particolari misure di sicurezza e riservatezza dei dati. Per le comunicazioni tra [client](#) e server, vengono impiegati [algoritmi](#) standard, ampiamente testati e collaudati sul web, come quelli offerti dal protocollo SSL. Il logging e il tracking degli eventi forniscono protezione dagli accessi non autorizzati.

L'istanza tecnologica dell'Application

Server: JBoss

JBoss [2] è un application server **open source**. E' conforme a tutte le specifiche J2EE imposte dalla Sun Microsystems e può essere installato su tutte le piattaforme (compreso i mainframe AS400 e OS390).

Si tratta di un motore applicativo in grado di contenere e gestire applicazioni Web ed Enterprise scritte in Java secondo le specifiche J2EE.

L'utilizzo di questa tecnologia porta con sé il vantaggio di essere basata su standard e di supportare le ultime novità tecnologiche nell'ambito dello sviluppo java quali il motore di persistenza Hibernate e le specifiche Enterprise Java Bean (EJB) 3.0.

L'installazione di JBoss è molto semplice, ed i dettagli della stessa sono ampiamente documentati [1].

In sintesi l'installazione della distribuzione di JBoss per una architettura hardware (es. INTEL) crea una directory *jboss-**(es. *4.0.4*) che contiene gli script di start del server, JARs, i set di configurazione del server e le working directories.

La figura seguente illustra la directory di installazione del server JBoss.

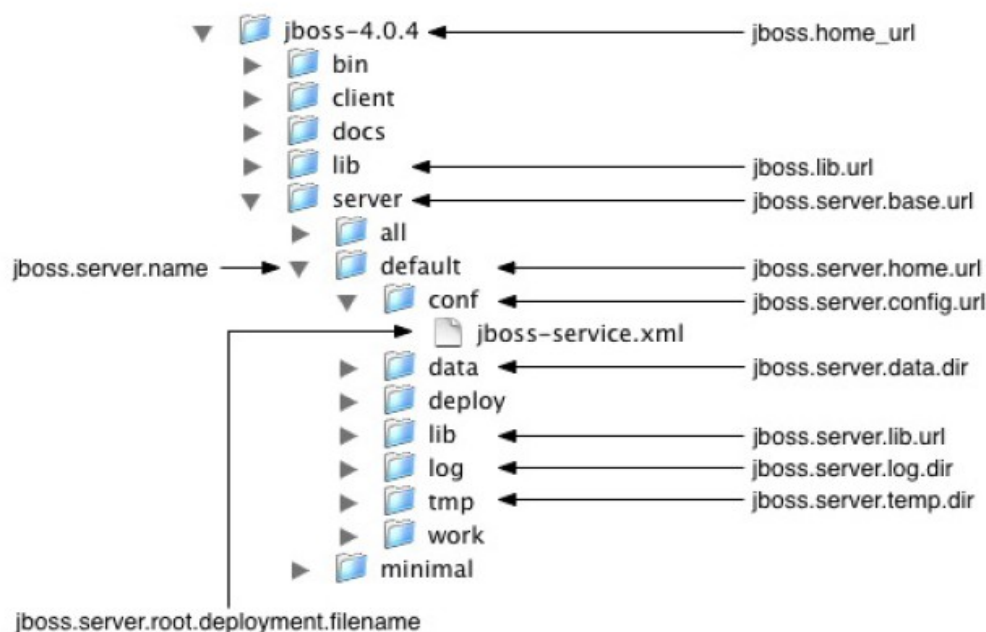


Figura 5 La struttura della directory di JBoss Application Server

La tabella seguente mostra le cartelle di top-level e le loro funzioni principali.

Directory	Description
bin	All the entry point JARs and start scripts included with the JBoss distribution are located in the bin directory.
client	The JARs that are required for clients that run outside of JBoss are located in the client directory.
server	The JBoss server configuration sets are located under the server directory. The default server configuration set is the server/default set. JBoss ships with minimal, default and all configuration sets. The subdirectories and key configuration files contained default configuration set are discussed in more detail after
lib	The lib directory contains startup JARs used by JBoss. Do not place your own libraries in this directory

Figura 6 La struttura della directory top-level di JBoss

La tabella seguente mostra le cartelle interne a quella della configurazione del server (*jboss-*(es. 4.0.4)/server/(es. default)*) e le loro funzioni.

Directory	Description
conf	The <code>conf</code> directory contains the <code>jboss-service.xml</code> bootstrap descriptor file for a given server configuration. This defines the core services that are fixed for the lifetime of the server.
data	The <code>data</code> directory is available for use by services that want to store content in the file system.
deploy	The <code>deploy</code> directory is the default location the hot deployment service looks to for dynamic deployment content. This may be overridden through the <code>URLDeploymentScanner</code> URLs attribute.
lib	The <code>lib</code> directory is the default location for static Java libraries that should not be hot deployed. All JARs in this directory are loaded into the shared classpath at startup.
log	The <code>log</code> directory is the directory log files are written to. This may be overridden through the <code>conf/log4j.xml</code> configuration file.
tmp	The <code>tmp</code> directory is used by JBoss to store temporarily files such as unpacked deployments.

Figura 7 La struttura della directory di configurazione dell'Application Server JBoss

Nella cartella “`deploy`” si trovano le applicazioni pubblicate, che vengono eseguite attraverso un browser.

Integrazione tra JBoss e AFS-Enea

In questo paragrafo si descrivono i dettagli implementativi della sperimentazione effettuata dal CRIAI, al fine di evidenziare l'integrabilità con l'infrastruttura applicativa già esistente in AFS-Enea.

Innanzitutto è stato installato JBoss su un PC dotato di sistema operativo Scientific Linux 4 e di un client AFS.

Nel file *jboss-service.xml*, che è uno dei file di configurazione di JBoss, posizionato in *DIR_INSTALLAZIONE_JBOSS/server/default/conf* è stata

aggiunta nella sezione “Deployment Scanning” all’interno del tag con
attribute name “URLs”, l’indicazione *file:/afs/enea.it/.../public_html* .

```
<!--          Deployment Scanning          -->
<!-- An mbean for hot deployment/undeployment of archives.  -->
<mbean
code="org.jboss.deployment.scanner.URLDeploymentScanner"
name="jboss.deployment:type=DeploymentScanner,flavor=URL">
  <depends optional-attribute-name="Deployer">
    jboss.system:service=MainDeployer
  </depends>
  <attribute name="URLComparator">
    org.jboss.deployment.DeploymentSorter
  </attribute>
  <attribute name="FilterInstance"
    attributeClass="org.jboss.deployment.scanner.DeploymentFilter"
    serialDataType="javaBean">
- <!--
  <attribute name="ScanPeriod">5000</attribute>
  <attribute name="ScanEnabled">true</attribute>

  <attribute name="URLs">
    deploy/,file:/afs/enea.it/por/user/jannace/public_html
  </attribute>
  <attribute name="RecursiveSearch">True</attribute>
</mbean>
```

Figura 8 Modifiche effettuate in jboss-service.xml

Ciò è stato fatto per consentire al server JBoss di estendere la
pubblicazione delle proprie applicazioni anche al sistema AFS-Enea.

L'infrastruttura in questo modo è dotata di 2 server:

il web server Apache all'indirizzo www.afs.enea.it

l'application server JBoss ad un altro indirizzo IP (da definirsi)

In questo modo, richiamando da un browser, l'indirizzo:

http://indirizzo_IP_server_JBoss/Nome_Applicazione

sarà possibile visualizzare qualunque **applicazione web** che si trova nella directory [/afs/enea.it/.../public_html](http://afs.enea.it/.../public_html).

Si sottolinea che il web server Apache può continuare a funzionare secondo le attuali modalità di utilizzo, pubblicando le pagine HTML contenute in [/afs/enea.it/.../public_html](http://afs/enea.it/.../public_html).

Sia il web server che l'application server possono pubblicare contenuti della medesima directory di AFS-ENEA

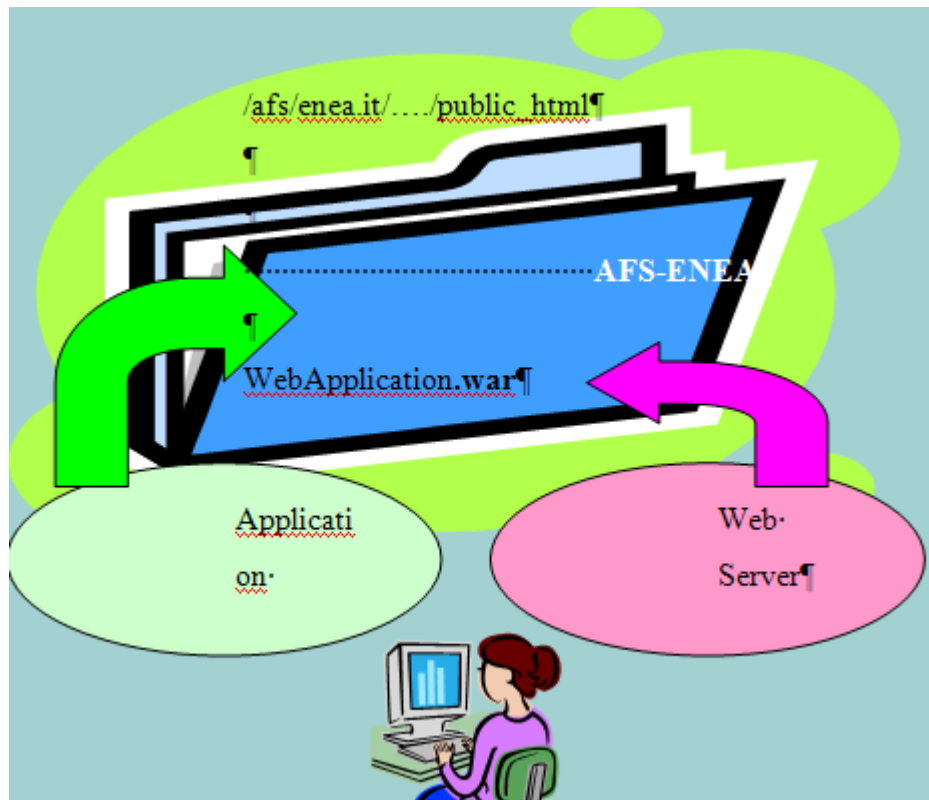


Figura 9 Struttura della pubblicazione web in AFS-ENEA

Ma essendo esso un web server, digitando nella barra degli indirizzi di un browser

`http:// www.afs.enea.it/user/Nome_Applicazione_Dinamica`

la web application non viene ovviamente elaborata e visualizzata come accade nel caso di un Application Server.

Se si richiama il nome completo del file che si trova in public_html, cioè

`http://www.afs.enea.it/user/ Nome_Applicazione_Dinamica.war`

allora il web server Apache ne consente il download al browser che lo sta richiedendo.

Casi d'uso dello scenario.

Un attore è l'utente dell'infrastruttura AFS-ENEA di tipo editore di contenuti web (Editore)

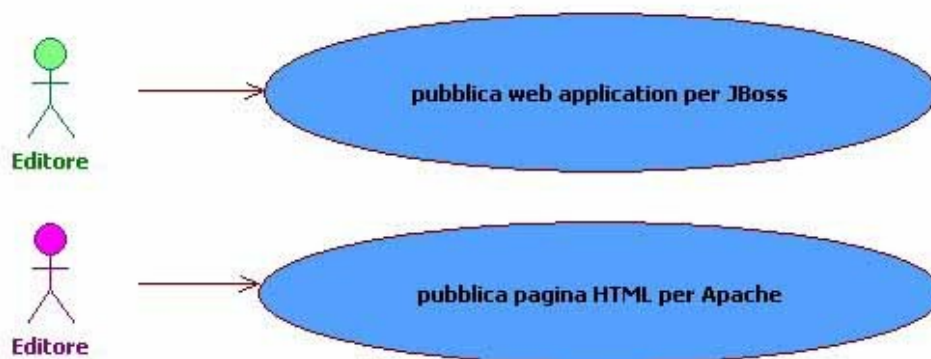


Figura 10 Caso d'uso Pubblicazione di Contenuti Web in AFS-ENEA

Un altro attore è l'utente generico dell'infrastruttura ENEA-GRID di tipo fruitore (Utente).



Figura 11 Caso d'uso Navigazione nei contenuti web di ENEA-GRID

Modifiche effettuate sulla macchina virtuale

Creazione dell'utente jboss sulla macchina ospitante il server

- Protezione per l'applicazione jmx-console, la quale permette tramite web di accedere alle informazioni sensibile dell'application server.
- Modifica dei file necessari ad un livello di sicurezza momentaneo:

JBOSS_HOME/server/default/deploy/jmx-console.war/WEB-INF/web.xml

```
<!-- A security constraint that restricts access
to the HTML JMX console to users with the role
JBossAdmin. Edit the roles to what you want and
uncomment the WEB-INF/jboss-web.xml/security-domain
element to enable secured access to the HTML JMX
console. -->

<security-constraint>
  <web-resource-collection>
    <web-resource-name>HtmlAdaptor</web-
resource-name>
    <description>An example security config that
only allows users with the
        role JBossAdmin to access the HTML JMX
console web application
    </description>
    <url-pattern>/*</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>
  <auth-constraint>
    <role-name>JBossAdmin</role-name>
  </auth-constraint>
</security-constraint>

<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>JBoss JMX Console</realm-name>
</login-config>

<security-role>
```

```
<role-name>JBossAdmin</role-name>  
</security-role>
```

- JBOSS_HOME/server/default/deploy/jmx-console.war/WEB-INF/jboss-web.xml

Abilita il modulo di richiesta password per l'applicazione jmx-console

```
<jboss-web>  
<!-- Uncomment the security-domain to enable security. You  
will need to edit the htmldaptor login configuration to  
setup the login modules used to authentication users. -->  
<security-domain>java:/jaas/jmx-console</security-domain>  
</jboss-web>
```

- modifica dei file:
 - jmx-console-roles.properties
 - jmx-console-users.properties

per definire ruolo e utente di un amministratore non di default

I passi precedenti sono stati ripetuti anche per l'applicazione web-console, la quale permette accesso via web a informazioni sensibili.

L'applicazione web-console risiede:

JBOSS_HOME/server/default/deploy/management/console-mgr.sar/web-console.war

I due sistemi progettati dal CRIAI ed attualmente in fase di implementazione sono ovviamente diversi per la natura stessa delle funzionalità che essi vanno ad esplicare:

MAGO è un sistema di monitoraggio delle risorse presenti

-

nell'infrastruttura ENEA-Grid

ASSOCIA è un sistema di comunicazione tra gli utenti e le

-

simulazioni e tra le simulazioni ed i simulatori.

Tuttavia esiste un unico punto di convergenza tra i due sistemi dovuto alla modularità progettuale con cui essi sono stati ideati.

Si tratta di un punto di convergenza di carattere infrastrutturale, e consiste nell'Application Server in cui risiederanno le web application di entrambi i progetti.

La scelta dell'application server (JBoss) è una esplicita richiesta del SPIII.5, ma una volta effettuata, ritorna comodamente riutilizzabile per ogni altra esigenza di carattere distribuito di ENEA, tra cui quindi anche la fruizione del sistema di monitoraggio MAGO, la cui web

application verrà pubblicata nel medesimo server.

Per i dettagli implementativi della fruizione generale dell'Application Server JBoss all'interno dell'infrastruttura ENEA-Grid si rimanda allo specifico documento.

In particolare il modulo di LOGIN che effettua il “klog” per accedere ad

ENEA-Grid è utilizzato da entrambe le web application di MAGO ed ASSOCIA.

L'interazione con l'LSF

In questa sezione viene affrontato il valore aggiunto delle applicazioni sviluppate dal CRIAI nei confronti dell'infrastruttura ENEA-GRID, in particolare in LSF.

Per quanto detto nei precedenti paragrafi MAGO affronta problematiche relative al monitoraggio, basando la sua forza sulla capacità di includere metriche ad hoc per la rilevazione dei dati significativi.

Tale approccio offre la possibilità di monitorare, e allo stesso tempo valutare strumenti alla base dell'ENEA-GRID, come AFS ed LSF.

L'altra applicazione sviluppata dal CRIAI è ASSOCIA, la quale permette l'esecuzione e l'interazione di simulatori, nonostante le diverse caratteristiche di linguaggio, di INPUT e di OUTPUT.

I componenti dei due sistemi sono modulari e scalabili, progettati per sfruttare le potenzialità dell'infrastruttura di calcolo ENEA-Grid, ma estensibili per ogni sviluppo futuro.

In particolare per quanto riguarda il modulo SLA del progetto ASSOCIA offre un'introduzione all'utilizzo dei ServiceClass introdotti nella versione 6.0 di LSF. Attualmente per instradare l'esecuzione delle simulazioni sui nodi di calcolo più scarichi di lavoro, viene utilizzato LSF.

Una possibilità evoluzione della precedente funzionalità è l'utilizzo del database delle risorse implementato per il sistema MAGO. Secondo tale modalità il modulo di SLA potrebbe effettuare una richiesta al database per stabilire il carico di lavoro ed instradare l'esecuzione del JOB completo di simulazione verso i nodi di calcolo rilevati più idonei attraverso il sistema MAGO stesso.

Offrendo così un approccio Grid della scoperta e gestione delle risorse.