DOTTORATO DI RICERCA

IN

SCIENZE COMPUTAZIONALI E INFORMATICHE

CICLO XVIII

CONSORZIO TRA UNIVERSITÀ DI CATANIA, UNIVERSITÀ DI NAPOLI FEDERICO II, SECONDA UNIVERSITÀ DI NAPOLI, UNIVERSITÀ DI PALERMO, UNIVERSITÀ DI SALERNO

SEDE AMMINISTRATIVA: UNIVERSITÀ DI NAPOLI FEDERICO II

VALERIA MANNA

Grid Workflow per l'Analisi dei Dati Astronomici tramite Web Service

Indice

Introduzione

L'astrofisica è una scienza che ci guida in maniera naturale all'uso della tecnologia computazionale.

Con la simulazione dell'universo e l'analisi di enormi quantità di dati collezionati durante le osservazioni, l'astrofisica ha sempre avuto bisogno del calcolo ad alte prestazioni e di tecniche di elaborazione avanzate. Non c'è da meravigliarsi, quindi se oggi l'astrofisica è una delle guide principali per lo sviluppo delle tecnologie Grid.

L'astrofisica utilizza strumenti di calcolo per analizzare i fenomeni dell'universo, da quelli localizzati, come la formazione delle stelle, la fisica solare, le supernove e i buchi neri, a quelli che nascono dall'insieme di tutti questi eventi, come la formazione e l'interazione tra galassie. Ognuno di questi processi risulta complesso e richiede la conoscenza di diverse materie scientifiche dalla fisica nucleare all'idrodinamica. Di conseguenza gli algoritmi sviluppati sono complessi e i relativi problemi computazionali sono immensi. La maggior parte di questi problemi sono tridimensionali e su larga scala e necessitano di un codice parallelizzato che venga eseguito su calcolatori avanzati. Un altro problema in questo contesto è la gestione dell'output prodotto in enorme quantità dopo le analisi e le elaborazioni dei dati. Lo storage, la gestione, la visualizzazione e l'interpretazione dei dati sono questioni molto rilevanti nell'ambito delle osservazioni.

Poichè la materia è ricca e vasta, gli astrofisici stringono continue collaborazioni e non esiste un unico gruppo che abbraccia tutte le conoscenze necessarie, che abbia sufficiente esperienza da risolvere tutti i problemi che emergono. Di conseguenza gli astrofisici sono coinvolti in numerose organizzazioni che collaborano e condividono risorse dati, codice e conoscenze. Per analizzare i dati provenienti dai nuovi osservatori e i moderni telescopi, i progetti degli astronomi richiedono collaborazioni internazionali. La velocità con cui aumentano i dati da elaborare è impressionante. Le organizzazioni che forniscono servizi agli astronomi devono garantire sia l'accesso ai software che raw data di alta qualità di fronte ai volumi di dati prodotti che si raddoppiano in meno di dodici mesi. Lentezza di CPU, capacità limitate

degli storage e diversità dei dati sono i problemi più urgenti nell'astronomia.

La soluzione a questi problemi è rappresentata dal calcolo e dagli storage distribuiti. L'utilizzo delle infrastrutture Grid permette ai ricercatori astronomi di migliorare le loro elaborazioni e applicazioni. I processi di analisi e simulazione e l'interazione tra le sorgenti dati sono i motivi principali che rendono l'utilizzo di Grid indispensabile per la comunità astrofisica.

In particolare nell'ambito esistono strumenti come i telescopi a grande campo che producono una notevole quantità di dati che devono essere processati. Vi sono diverse iniziative internazionali mirate a far fronte all'enorme mole di dati astronomici che sono e/o saranno prodotti nei prossimi anni dai grandi telescopi e dai rilevatori CCD a mosaico di nuova generazione. Un esempio di queste nuove macchine a grande campo è dato dal VST, un telescopio interamente dedicato all'imaging a grande campo equipaggiato con una camera a mosaico chiamata OmegaCAM (32 CCD per un totale di 256 Megapixels).

La Comunità Astronomica Italiana è fortemente impegnata in questo progetto: il telescopio è interamente costruito dall'Osservatorio Astronomico di Capodimonte (OAC) e gli Osservatori di Padova e Capodimonte prendono parte al consorzio europeo per la realizzazione della camera. Per queste ragioni, una frazione significativa del tempo di osservazione di VST sarà dedicato a progetti nazionali e questi dati verranno stivati in grandi archivi su disco.

Il consorzio europeo ASTRO-WISE, finanziato dall'Unione Europea, cui partecipa l'OAC, si sta occupando delle problematiche legate alla riduzione, all'analisi e allo storage dei dati VST.

La ricerca è incentrata sull'analisi delle problematiche legate all'integrazione in ambito Grid delle risorse sviluppate dalla comunità astrofisica per la riduzione dei dati del telescopio VST e il conseguente approccio innovativo alle metodologie per l'utilizzo di servizi esistenti e futuri.

In fase di test e validazione della Grid, risulta estremamente utile poter contare su metodi oggettivi e ripetibili per valutare l'efficienza dell'esecuzione di applicazioni su Grid, in funzione delle problematiche tipiche delle applicazioni astrofisiche e delle richieste degli utenti, in modo da massimizzare l'efficienza ed individuare eventuali punti deboli.

Una parte iniziale della ricerca è stata dedicata all'aspetto architetturale delle Griglie Computazionali, analizzando, in particolare, le problematiche relative ai quality of service e alla creazione, alla gestione e all'utilizzo della Grid. Una fase è stata dedicata all'installazione e alla configurazione del software di Grid all'OAC, per creare un nodo funzionale della Grid nazionale; in seguito è stato affrontato uno studio sulle metodologie per integrare un cluster in ambiente Grid senza alterarne la configurazione e l'integrità in quanto ambiente massicciamente utilizzato dalla comunità scientifica dell'OAC.

Una seconda parte è relativa alle problematiche legate all'utilizzo di Grid per la riduzione dei dati del telescopio VST , utilizzando un'applicazione modulare che elabora per step successivi i dati astronomici, con una metodologia Pipeline.

L'interesse della comunità astrofisica per l'uso di tecnologie Grid per l'elaborazione di immagini astronomiche wide field ha portato allo studio delle modalità ottime di integrazione di tali applicazioni in Grid.

Da tali studi si è evidenziata la non banalità di integrare la Pipeline negli attuali middleware, a causa della complessità dei task di riduzione dati astronomici, che mal si adattano all'attuale sistema di gestione del workload. Si è riscontrata, quindi, la necessità della creazione di un workflow che descriva tutti i task di cui è costituita un'applicazione astrofisica, come la conversione dei dati, il trasferimento dei file, la sincronizzazione dei job e l'analisi dei risultati. L'obiettivo è quello di permettere all'utente (l'astronomo) di interrogare più centri dati in modo trasparente, di fornire potenti strumenti per nuove analisi e visualizzazioni, di fornire uno standard per i servizi di consegna e di pubblicazione.

Il progetto EGEE, nato per consolidare l'infrastruttura creata con il progetto DATAGRID e rivolto alla sperimentazione del middleware gLite, sta integrando un'architettura basata sui Web Service.

Per integrare la Pipeline VST nel middleware di gLite, utilizzando le nuove tecnologie Service Oriented, si è svolta un'analisi approfondita dei vari gestori dei workflow, compatibili con la struttura dei Web Service, e rispondenti a standard riconosciuti.

Quest'analisi ha portato a focalizzare l'attenzione su un particolare linguaggio di workflow, chiamato BPEL4WS, in grado di orchestrare i Web service coinvolti nell'esecuzione delle applicazioni.

BPEL4WS rappresenta un linguaggio standard e open, che riesce a realizzare l'orchestration dei servizi, in particolare nell'ambito del Grid Services supportando complessi workflow e riuscendo ad esporre ai Web Service tutte le activity che sono eseguite in un workflow.

Nell'utilizzare BPEL4WS workflow è importante fornire una serie di servizi che interagiscono nei process flow. Un modo per garantire i servizi è l'utilizzo di partner astratti. Essi sono servizi invocati per l'esecuzione di workflow, mappati sui Web Service o mappati attraverso il WSDL port type di cui ogni partner è fornito. Un BPEL4WS workflow può interagire con molti partner alcuni dei quali possono essere Web Service, WSRF service, o altri BPEL4WS workflow. Inoltre alcuni partner non sono esposti come gli attuali Web Service ma attraverso il WSDL descrivono le loro classi Java.

L'utilizzo di BPEL4WS non apporta grossi vantaggi se le combinazioni non corrispondo a invocazioni a Web o Grid Service.

Attualmente il progetto EGEE sta sviluppando un'infrastruttura Grid in cui una serie di componenti espongono Web Service. Molte altre architetture, come LCG, a cui fa riferimento sia l'INAF che INFN, non sono ancora basate sui Web Service. Una possibile soluzione per superare le difficoltà dell'attuale stato dell'arte dei servizi e, nello stesso tempo, utilizzare un linguaggio standard per la descrizione dei Web Service, è utilizzare un tool per invocare i partner che non sono accessibili come dei reali Web Service. Tra tali tool, uno in particolare, il WSIF (Web Service Invocation Framework), presenta caratteristiche interessanti, riuscendo a consentire il mapping dei metodi e ad esporre le descrizioni necessarie a Bpel per interagire con i servizi mappati.

In questo modo WSIF permette all'utente di interagire con la rappresentazione astratta dei Web Service e permette l'aggiunta dinamica di nuovi binding. Il tool invoca i partners che non sono accessibili come dei reali Web Service: ad esempio un servizio può essere utilizzato invocando metodi di una classe Java.

Nell'analisi dell'utilizzo dei workflow per le applicazioni astronomiche viene proposto un modo per utilizzare un linguaggio di workflow standard e nel contempo interagire con i Grid service che non sono stati esposti come dei Web Service. L'utilizzo combinato di WSIF e Bpel consentirebbe anche nell'immediato l'adozione di un completo ed affidabile linguaggio di esecuzione per la sottomissione di task nell'ambito della riduzione dei dati astronomici.

Attraverso l'utilizzo delle API WSIF è possibile invocare delle classi Java; in particolare un Java binding descrive il modo in cui una classe Java supporta un port type definito. Il Java binding è un WSDL binding che permette di descrivere la classe Java utilizzando WSDL. Sotto queste condizioni si può accedere a un Grid service attraverso le classi Java utilizzando un tool, chiamato Commodity Grid (CoG) Kits. CoG permette agli utenti di Grid, agli sviluppatori e agli amministratori della Grid, di utilizzare un framework higher-level. In particolare Java CoG fornisce un framework per interagire con una serie di servizi che fanno parte di GT2 come GRAM, GASS, GridFTP, MDS.

In questo modo è possibile utilizzare BPEL4WS workflow per le infrastrutture Grid che non sono ancora esposte come Web service e riusciamo a descrivere workflow astronomici, realizzando la composizione dei servizi nell'ambito dei Grid Service, calandoci nella realtà degli attuali middleware.

Il percorso formativo e l'attività di ricerca hanno contribuito allo studio e alla messa a punto di tecniche di elaborazione e di calcolo distribuito per la comunità scientifica astronomica. In particolare, è emersa l'esigenza di descrivere le applicazioni astronomiche con l'utilizzo di un linguaggio di workflow che permette l'integrazione della Pipeline in Grid, componen-

do le nuove tecnologie Grid Service. L'analisi delle metodologie propone una soluzione integrabile con il middleware attuale e con le sue future evoluzioni.

Capitolo 1

Grid: caratteristiche e funzionalità

1.1 Introduzione alle Griglie Computazionali

Grid è un'infrastruttura hardware e software nata circa un decennio fa con l'intento di supportare il calcolo scientifico coordinando un numero di risorse condivise e distribuite geograficamente. Analogamente al Web, Grid in origine, è nato esclusivamente nell'ambito scientifico per favorire il calcolo distribuito intensivo e l'accesso flessibile a una grande mole di dati, ma negli ultimi tempi sta ricevendo particolari attenzioni anche nel campo industriale, commerciale e finanziario.

Negli ultimi anni c'è stata una notevole evoluzione delle reti di calcolatori e di molte altre teconologie che sono migliorate esponenzialmente in termini di performance, funzionalità e affidabilità (25). Negli ambiti istituzionali, governativi e commerciali, l'affidabilità di Internet e il calcolo distribuito sono diventati di uso comune come avviene per l'utilizzo dell'elettricità . Oggi, sia i professionisti che i ricercatori, si sottoscrivono e utilizzano risorse di calcolo distribuito.

Il concetto di calcolo distribuito non è così recente; basta ricordare infatti i sistemi per il Remote Job Entry (10), le stampanti condivise, le chiamate a procedure remote, il networked file system, i sistemi di calcolo etereogenei e gli ambienti per la simulazione di multiprocessori. Grid rispetto a queste tecnologie viene visto come un successore nell'ambito del calcolo distribuito. Le Griglie computazionali non nascono con lo scopo di sostituire i vari servizi e i sistemi di calcolo presenti già da anni ma per essere un loro completamento. Tuttavia è importante sottolineare la reale differenza che c'e tra Grid e i sistemi distribuiti, al fine di fornire una definizione completa e chiara (42).

Il concetto di griglie computazionali si può riassumere in tre punti: facile accesso alle risorse di computing/storage, accesso alle risorse tramite un provider, condivisione delle risorse per il raggiungimento di uno scopo comune (Virtual Organization). Sebbene le finalità e le motivazioni di Grid siano chiare non esiste una definizione altrettanto chiara. Vari sviluppatori hanno definito Grid come:

- una risorsa coordinata, flessibile e sicura, distribuita tra un'insieme dinamico di individui, istituzioni e risorse (23)
- un unico ambiente nel quale i cicli, le comunicazioni e i dati sono distribuiti e dove le workstation distribuite su scala geografica sono a portata di tutti (29)
- un unico ambiente costituito in maniera trasparente da: personal computer, workstation, supercomputer e dispositivi non tradizionali (30)
- una collezione di risorse geograficamente distribuite (persone, computer, strumenti e database) connessi tra di loro attraverso una rete ad alta velocità, e un software chiamato "middleware" che trasforma una collezione di risorse indipendente in un'unica macchina virtuale (35)

Caratterizzazioni più recenti non hanno contribuito a far emergere una definizione comune: *coordinator resource sharing (22), single-system image, comprhensiveness of resources (33) e utility computing* sono le parole chiavi per descrivere Grid, formulate da esperti della materia. Con vari gradi di precisione, queste definizioni descrivono la nozione fondamentale di un sistema Grid, ma non sono ancora in grado di chiarire la differenza tra Grid e i sistemi di calcolo distribuito convenzionale. La maggior parte dei tentativi di definizione di Grid sono stati del tutto informali e si sono focalizzati su come un sistema Grid possa essere costruito: dalle componenti, alle interfacce e ai protocolli diffusi. Ma per giungere ad una definizione universale bisogna puntare l'attenzione su cosa un sistema Grid dovrebbe fornire: quali sono le funzionalità fondamentali per la creazione di una Grid senza considerare le attuali componenti, i protocolli e altri dettagli di implementazione.

1.2 Differenze tra Grid e un ambiente di calcolo distribuito convenzionale

Una metodo per differenziare Grid e un sistema di calcolo distribuito convenzionale può essere quello di fare una caratterizzazione sulle *macchine virtuali*.

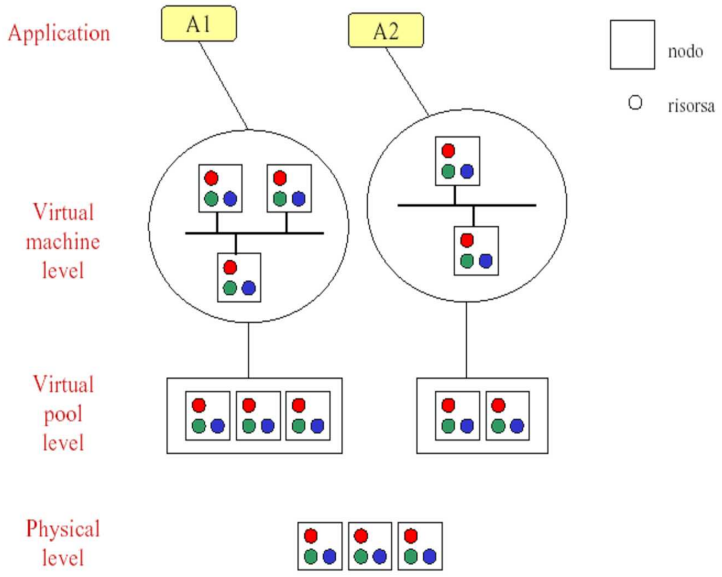

Figura 1.1: Macchina virtuale in un sistema distribuito

Un'applicazione distribuita è composta da un numero finito di processi cooperativi che sfruttano risorse debolmente accoppiate (*loosely coupled resources*). La *macchina virtuale* su cui viene eseguita un'applicazione distribuita differisce nelle sue caratteristiche se consideriamo un sistema di calcolo distribuito convenzionale o un sistema di calcolo Grid.

1.2.1 Caratterizzazione di un sistema di calcolo distribuito convenzionale

Nel calcolo distribuito convenzionale l'applicazione distribuita assume l'esistenza di un pool di nodi di calcolo che implementano una *macchina virtuale* (Fig.1.1). Il pool è costituito da un insieme di PC, supercomputer, workstation e possibilmente supercalcolatori su cui, l'utente che esegue l'applicazione distribuita, ha accesso tramite un account. L'accesso su una *macchina virtuale* consiste nell'autenticazione su ciascun nodo (42). In generale si può assumere che l'utente autenticato su un nodo del pool viene automaticamente autorizzato all'utilizzo di *tutti* i nodi del pool. L'utente di un sistema di calcolo distribuito convenzionale conosce le caratteristiche dei nodi a disposizione come ad esempio il tipo di architettura, la potenza di calcolo, il sistema operativo e la sicurezza. Inoltre il pool di nodi virtuale può essere considerato statico dal momento che il set di nodi sul quale l'utente ha avuto

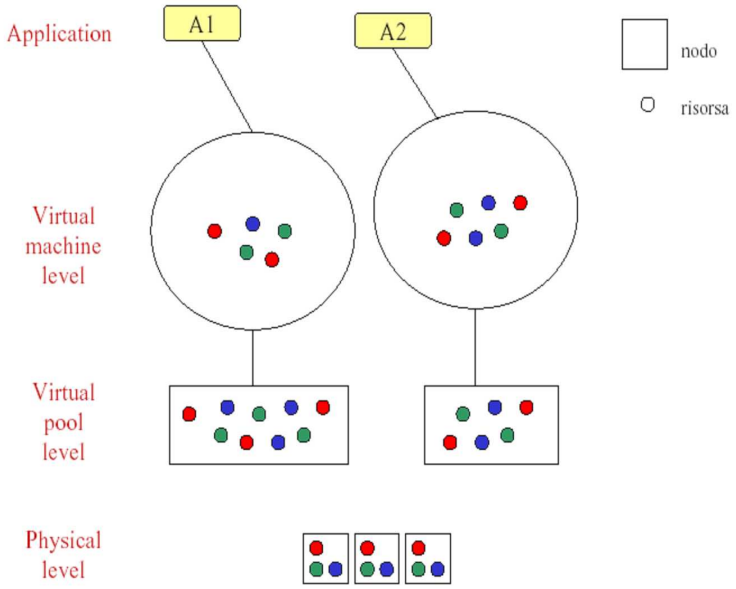

Figura 1.2: Macchina virtuale Grid

accesso cambia molto raramente. Questo tipo di sistemi ha una grandezza che è tipicamente dell'ordine di 10-100 nodi, sia come deployment che come utilizzo.

1.2.2 Caratterizzazione di un sistema di calcolo Grid

In Grid un'applicazione distribuita assume l'esistenza di un pool di risorse come ad esempio processori, memoria, e disco, distribuite su scala geografica. La *macchina virtuale* (Fig.1.2), nel caso di Grid, è costituita da un set di risorse del pool. I sistemi Grid, infatti, sono realizzati per ottenere l'esecuzione su risorse distribuite, come storage, network, dati, software e risorse atipiche come componenti grafici di I/O audio, manipolatori, sensori e così via. Tutte queste risorse vengono tipicamente rappresentate come nodi geograficamente distribuiti e ripartite su diversi domini amministrativi. L'accesso alla *macchina virtuale* presuppone che l'utente possiede delle credenziali accettate dai proprietari delle risorse del pool. Un utente può essere autorizzato all'utilizzo di una risorsa *senza* avere un account sul nodo che ospita la risorsa. Opportune funzioni di mapping permettono di mappare l'utente di Grid in utenti di sistemi locali realizzando un aspetto peculiare di Grid: la *virtualizzazione degli utenti*. Il pool virtuale di risorse è dinamico, quindi le risorse possono essere aggiunte e riservate ogni volta a discrezione del proprietario e le loro performance e il carico possono cambiare di conti-

	Conventional distributed environments	Grids
1	A virtual pool of computational nodes	A virtual pool of resources
2	A user has access (credential) to all the nodes in the pool	A user has access to the pool but not to individual nodes
3	Access to a node means access to all resources on the node	Access to a resource may be restricted
4	The user is aware of the capabilities and features of the nodes	The user has little or no knowledge about each resource
5	Nodes belong to a single trust domain	Resources span multiple trust domains
6	Elements in the pool 10–100, more or less static	Elements in the pool $\gg$100, dynamic

Figura 1.3: Schema sulle differenze tra un convenzionale ambiente distribuito e Grid.

nuo nel tempo. Questo aspetto reallizza l'altro aspetto distintivo di Grid: la *virtualizzazione delle risorse*. La dimensione di un sistema di calcolo Grid convenzionale è dell'ordine di migliaia di nodi. Inotre l'utente di un sistema di calcolo distribuito Grid non ha nessuna conoscenza a priori del tipo, dello stato e delle caratteristiche delle risorse a disposizione.

Nella tabella in figura 1.3 riassumiamo le differenze tra sistemi distribuiti convenzionali e sistemi Grid.

1.2.3 Verso una definizione universale di Grid

Attualmente non esiste ancora, per Grid, una definizione universalmente accettata. Infatti è riconosciuto che Grid è un insieme di risorse condivise, ma non esiste una descrizione formale relativa alle proprietà specifiche di Grid e alle funzionalità che essa supporta. Quello che possiamo aggiungere è che Grid è un insieme di servizi che consentono il calcolo distribuito su risorse appartenenti a diversi domini di gestione e che fornisce virtualizzazione degli utenti e delle risorse. In base alla definizione e alla caratterizzazione di Grid e' possibile determinare se un sistema di calcolo e' "Grid-enabled" (42).

1.3 Classificazione dei sistemi di calcolo Grid-enabled

E' facile confondere alcuni sistemi di calcolo distribuiti con Grid e spesso non è chiaro quali sistemi di calcolo distribuito forniscono effettivamente le funzionalità di Grid e quali no. In accordo con la definizione data, ci sono alcuni sistemi che si qualificano come Grid anche se non sono classificati come tale, e altri che non rispondono ai criteri di qualificazione nonostante loro usino il termine Grid nelle proprie descrizioni. Analizziamo alcuni sistemi dividendoli in sistemi non Grid, sistemi approssimativamente Grid e sistemi Grid.

Esempi di Sistemi non Grid:

- Un sistema di gestione di code batch che utilizza le CPU di un computer multi-processore o di computer inseriti in un cluster o su una LAN. In questo modo esiste un controllo centralizzato delle risorse e la conoscenza completa dello stato del sistema.

- Il Web anche se utilizza protocolli standard, aperti e general-purpose, non gestisce in modo coordinato le risorse per assicurare la migliore quality of service.

Esempi di sistemi approssimativamente Grid:

- I sistemi di *schedulers multi-site* come ad esempio Condor (6), Entropia o SETI@home che distribuiscono risorse in modo non centralizzato sfruttando milioni di CPU inattive e connesse ad Internet assicurando quality of service, seppure in modo limitato.

Esempi di sistemi Grid:

- In Europa i progetti di "Data Grid" per il calcolo intensivo e distribuito nell'ambito accademico e scientifico sono *EDG, CrossGRID, Data Tag, LCG, EGEE*; in USA i più famosi sono *GriPhyN, PPDG, iVDGL* e in Asia c'è *ApGrid*. Essi si propongono di integrare risorse anche non omogenee appartenenti a molte istituzioni che conservano in ogni caso le loro politiche di utilizzo. L'accesso *on-demand* alle risorse viene realizzato utilizzando protocolli aperti, standard e general-purpose per la gestione delle risorse (ad es. il *Globus Toolkit*, l'EDG, l'*Open Grid Service Architecture* - OGSA) che hanno l'obiettivo di garantire quality of service dal punto di vista della sicurezza, dell'affidabilità e delle prestazioni.

Capitolo 2

La riduzione dei dati VST in GRID

2.1 La necessità di una nuova infrastruttura di calcolo per l'astronomia

L'astrofisica è una scienza che ci guida in maniera naturale all'utilizzo della tecnologia computazionale. Con la simulazione dell'universo e l'analisi di enormi quantità di dati collezionati durante le osservazioni, l'astrofisica ha sempre avuto bisogno del calcolo ad alte prestazioni e di tecniche di elaborazione avanzate (25). Non c'è da meravigliarsi, quindi se oggi l'astrofisica è una delle guide principali per lo sviluppo delle tecnologie Grid.

L'astrofisica utilizza strumenti di calcolo per analizzare i fenomeni dell'universo, da quelli localizzati, come la formazione delle stelle, la fisica solare, le supernove e i buchi neri, a quelli che nascono dall'insieme di tutti questi eventi, come la formazione e l'interazione tra galassie. Ognuno di questi processi risulta complesso e richiede la conoscenza di diverse materie scientifiche dalla fisica nucleare all'idrodinamica. Di conseguenza gli algoritmi sviluppati sono complessi e i relativi problemi computazionali sono immensi. La maggior parte di questi problemi sono tridimensionali e su larga scala e necessitano di un codice parallelizzato che venga eseguito sui calcolatori avanzati. Un altro problema in questo contesto è la gestione dell'output prodotto in enorme quantità dopo le analisi e le elaborazioni dei dati. Lo storage, la gestione, la visualizzazione e l'interpretazione dei dati sono questioni molto rilevanti nell'ambito delle osservazioni. Poichè la materia è ricca e vasta, gli astrofisici stringono continue collaborazioni e non esiste un unico gruppo che abbraccia tutte le conoscenze necessarie e che abbia sufficiente esperienza da risolvere tutti i problemi che emergono. Di conseguenza gli astrofisici sono coinvolti in

numerose organizzazioni che collaborano e condividono risorse dati, codice e conoscenze. Per analizzare i dati provenienti dai nuovi osservatori e i moderni telescopi, i progetti degli astronomi richiedono collaborazioni internazionali. La velocità con cui aumentano i dati da elaborare è impressionante. Le organizzazioni che forniscono servizi agli astronomi devono garantire sia l'accesso ai software che raw data di alta qualità di fronte ai volumi di dati prodotti che si raddoppiano in meno di dodici mesi. Lentezza di CPU, capacità limitate degli storage e diversità dei dati sono i problemi più urgenti nell'astronomia. La soluzione a questi problemi è rappresentata dal calcolo e dagli storage distribuiti. L'utilizzo delle infrastrutture Grid permette ai ricercatori astronomi di migliorare le loro elaborazioni e applicazioni. I processi di analisi e simulazione e l'interazione tra le sorgenti dati sono i motivi principali che rendono l'utilizzo di Grid indispensabile per la comunità astrofisica.

2.1.1 L'utilizzo del cluster nelle applicazioni astronomiche

Un'effettiva soluzione per la gestione dell'image processing dei dati prodotti dagli attuali e futuri telescopi a grande campo è rappresentato dai sistemi di calcolo Beowulf. Attualmente all'Osservatorio Astronomico di Capodmonte(OAC) sono presenti due cluster beowulf: uno con 8 nodi computazionali e 2 nodi di gestione/immaganizzamento dati, per un totale di 20 CPU utilizzato per testare applicazioni di calibrazione e preriduzione; un altro più potente con 16 nodi e 32 CPU, dedicato alla riduzione dei dati. Per testare le performance del cluster è stata costruita all'OAC un'applicazione chiamata Astro-Bench. Essa è una serie di benchmarks per stimare le performance single/multi processor come appunto un sistema di cluster Beowulf. Attualmente i test sono stati eseguiti su quattro task astronomici:

- la produzione di un master Bias utilizzando la routine wfi_masterbias (eclipse) su cinque *bias*
- la produzione di un master Flat field di cinque *dome flat* e *sky flat* utilizzando la routine wfi_ff del programma SExtractor su cinque immagini *wfi*
- l'esecuzione di Swarp su quattro immagini *wfi*

I test mostrano che nel caso di applicazioni intensive sulla CPU, come Swarp, il tempo speso nell'I/O è piccolo rispetto a quello speso per la CPU. Invece per quanto riguarda il master bias e il master flat filed possono essere ottenuti alcuni miglioramenti modificando il software in modo tale da usare

maggiormente le risorse di rete e l'I/O. In ogni caso l'ordine dell'esecuzione dei task è fondamentale per il miglioramento delle performance.

Per utilizzare in maniera esaustiva l'hardware disponibile è importante installare sul cluster un sistema di gestione delle risorse, come ad esempio PBS. Una distribuzione efficiente dei job sui nodi di calcolo è uno dei task principali di un sistema di scheduling. In particolare la qualità dello scheduling ha un impatto molto alto sulle performance del calcolo parallelo. Lo scheduling sui nodi deve essere efficiente in modo da ottimizzare i tempi di esecuzione degli algoritmi ma in un ambiente eterogeneo è più complesso avere uno scheduling dinamico e programmare applicazioni parallele. Lo scheduling è una parte fondamentale del software per la gestione del workload e solitamente si occupa di: Queuing, Scheduling, Monitoring, Resource Management, Accounting.

Su entrambi i cluster, presenti all'OAC, non erano mai stati utilizzati sistemi di gestione delle code. A tale scopo all'OAC sono stati effettuati alcuni test sul cluster per studiare i tempi di esecuzione del'analisi delle immagini elaborate utilizzando il batch system (37).

Sistema di Gestione delle Risorse per la riduzione dei dati Astronomici

Sul cluster dell'OAC è stato installato un sistema di gestione open, chiamato Torque. Il master del cluster è stato utilizzato come nodo di sottomissione per lo scheduling dei job, mentre gli altri otto nodi sono stati configurati come nodi di esecuzione. Per testare le funzionalità di Torque sono stati sottomessi alcuni scripts per eseguire l'analisi delle immagini su otto CCD:

- 8 script per sottomettere i task wfi_masterbias e wfi_ff per la produzione del master bias e del master flat field;
- 8 script per sottomettere il task SExtractor;
- 8 script per sottometter il task Swarp per il resampling.

Il master bias e master flat field sono elaborati nello stesso script per ridurre l'accesso alla rete e il carico di I/O. Wfi_masterbias scrive i suoi risultati su una directory temporanea riducendo l'accesso alla rete. Infine, wfi_ff legge l'output del master bias da una directory temporanea locale e lo utilizza come input, evitando un altro accesso alla rete. In questo modo abbiamo ottimizzato il traffico di rete.

Tabella 2.1: Tempo di esecuzione di ogni singolo task della riduzione dei dati astronomici su ogni nodo al quale Torque ha assegnato il job. Il tempo è riportato in secondi

node	Job Name	Time
b01	wfi0	676
b01	wfi1	685
b02	wfi2	732
b02	wfi3	685
b03	wfi4	783
b03	wfi5	761
b04	wfi6	664
b04	wfi7	684
b05	sex0	54
b05	sex1	53
b06	sex2	21
b06	sex3	21
b07	sex4	27
b07	sex5	24
b08	sex6	48
b08	sex7	42
b06	swarp0	280
b07	swarp1	280
b07	swarp2	330
b05	swarp3	324
b06	swarp4	460
b08	swarp5	420
b05	swarp6	420
b08	swarp7	405

Swarp è compilato in modo multi-thread e quindi è capace di utilizzare entrambe le CPU di un nodo nello stesso tempo. Apportando piccole modifiche siamo riusciti a massimizzare l'utilizzo della CPU.

Per misurare i tempi di esecuzione di ogni task sono utilizzati le utilities unix *time* e *date*. I test sono stati effetuati in un ambiente di produzione e il cluster elabora continuamente i task di altri utenti per questo i risultati sono mostrati solo come esempio (tabella 2.1). Per confrontare i risultati è stato lanciato Astrobench con 8 CCD (tabella (2.2)). La tabella (2.3) mostra il tempo totale di esecuzione di un singolo task utilizzando Astrobench.

Tabella 2.2: Tempo di esecuzione totale della riduzione dei dati astronomici su ogni nodo utilizzando Torque e Astrobench. Il tempo è riportato in secondi

# node	Total Time with Torque	Total Time with Astrobench
8	800	1490

Tabella 2.3: Tempo di esecuzione totale per un singlo task della riduzione dei dati astronomici sul cluster utilizzando Astrobench. Il tempo è riportato in secondi

MasterBias	MasterFlat	sExtractor	SWarp
64	183	29	1214

Analisi spazio-temporale delle oscillazioni solari

All'OAC il gruppo dei Solari ha realizzato un'applicazione per la misura delle oscillazioni solari. Le procedure numeriche elaborano l'analisi delle armoniche sferiche delle immagini solari acquisite con lo strumento VAMOS per studi eliosismologici. I residui della velocità sono decomposti in armoniche sferiche e i coefficienti risultanti dalle armoniche sferiche sono utilizzati per la trasformata di Fourier per calcolare lo spettro delle oscillazioni solari. Le routine per il calcolo delle armoniche sferiche sono state sviluppate in IDL (45), (44). L'applicazione è stata eseguita sul cluster Beowulf per misurare i tempi di esecuzione in parallelo rispetto a quelli sequenziali. L'applicazione risulta di tipo "embarassing parallel". Infatti dal punto di vista dell'analisi

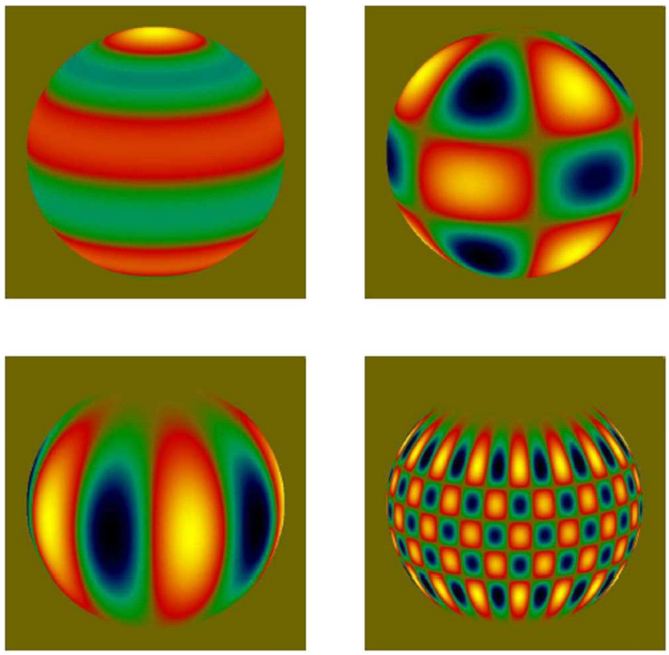

Figura 2.1: Esempi di armoniche sferiche: la velocità residuale è decomposta in armoniche sferiche

spaziale il numero totale delle immagini prese nell'arco di tempo T sono distribuite su m nodi di elaborazione. Dal punto di vista dell'analisi temporale, si possono suddividere le matrici dei coefficienti delle armoniche sferiche in n riquadri da elaborare separatamente lungo l'asse temporale. Il passo finale di riassemblaggio dei risultati non richiede grosse risorse di calcolo.

L'applicazione è stata eseguita sul cluster utilizzando il sistema di gestione dei job SunGridEngine. Attraverso la costruzione di uno script viene richiamato n volte il comando di sottomissione *qsub* osservando una diminuzione significativa del tempo di esecuzione rispetto alla stessa applicazione elaborata su tre computer.

2.1.2 Grid Site all'INAF-OAC

Nell'ambito del progetto DRACO (Datagrid for italian Research in Astrophysics and Coordination with the virtual Observatory) (46) il lavoro all'OAC è stato incentrato sugli aspetti relative all'integrazione delle risorse messe a disposizione per la riduzione dati delle immagini VST nella GRID nazionale, con particolare attenzione per i cluster Beowulf utilizzati per la riduzione dati. Uno degli aspetti principali del progetto è stato installare, integrare e

configurare il front-end con la rete GRID nazionale, per rendere l'OAC un Grid site attivo ed operativo. In questa fase sono state affrontate le problematiche software legate alla configurazione del sistema e alla gestione delle risorse hardware messe a disposizione della GRID. E' stato inoltre configurato il sottosistema di gestione degli account e delle risorse, in modo da garantire al sistema la sicurezza necessaria. Il Grid Site è formato da cinque server linux, uno per ogni elemento base di in nodo Grid:

- LCFG server
- User Interface
- Computing Element
- Storage Element
- Worker Node

Per questa fase di sperimentazione l'OAC, come gli altri membri del progetto DRACO, ha deciso di aderire alla Grid GILDA (Grid Infn Laboratory for Dissemination Activities), una Grid di test creata per ampliare ed dimostrare le possibilita' della tecnologia Grid. GILDA è una risorsa dell' Instituto Nazionale di Fisica Nucleare, legata alle esperienze italiane INFN GRID e il progetto europeo EGEE. Sul quantum Grid è stata inizialmente installata la release INFN-GRID 2.2.0 poi successivamente aggiornata con le nuove release supportate da INFN-GRID. Per il buon funzionamento del Grid site e per favorirne l'utilizzo da parte della comunità astronomica italiana ci sono delle periodiche verifiche dello stato del sito, monitoraggio e manutenzione della configurazione.

2.1.3 Integrazione del cluster in Grid

Un importante aspetto della nostra ricerca riguarda l'integrazione, come worker resource, del cluster Beowulf nella Grid (2.2). L'obiettivo di integrare il cluster dellOAC in Grid come una singola unit di calcolo rispettando i seguenti vincoli:

- soluzione non invasiva per la comunit astronomica dellOAC che utilizza la risorsa per la riduzione dei dati
- mantenere lintegrit del cluster
- mantenere la configurazione esistente

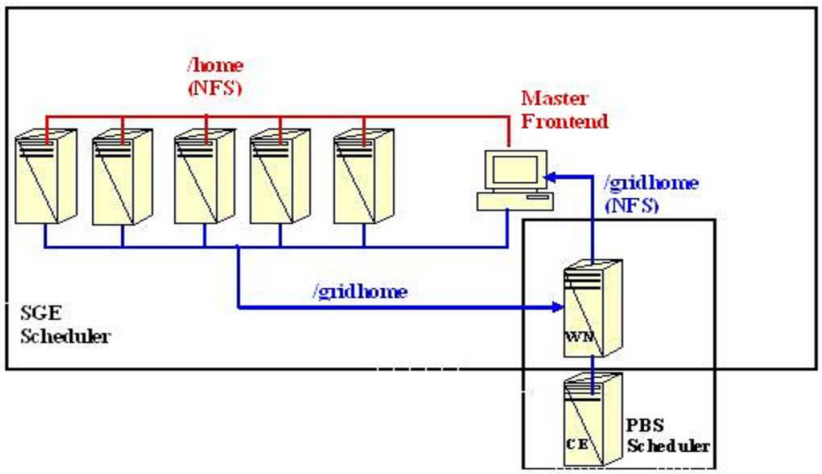

Figura 2.2: Integrazione del Cluster dell'OAC in Grid

Per integrare il cluster in Grid e utilizzare i suoi nodi come Worker Node della Grid è necessario installare su ogni nodo del cluster il sistema operativo richiesto dalla release fornita dall'INFN-GRID insieme a tutti i pacchetti che permettono la configurazioe dei nodi come Worker Node. Questo tipo di integrazione snatura il concetto di cluster e la sua unità come singola risorsa costituita da un master che coordina tutti i nodi e gestisce, tramite dei sistemi di scheduling, le code dei job sui singoli nodi.

Per utilizzare il cluster sia come worker node della Grid sia per altri tipi di applicazioni è necessario che questo rimanga intatto nella sua configurazione e integrità. In questo caso quindi la soluzione scelta è stata quella di effettuare alcune modifiche al Grid environment per consentire ai job sottomessi sul Worker Node di migrare sul cluster. Il worker Node del Quantum Grid è stato configurato con un numero di CPU pari ai nodi del cluster. Inoltre è stato configurato in modo tale che il master node del cluster lo utilizza come nodo di sottomissione all'interno del cluster. Il job manager della Grid continua ad essere OpenPbs mentre come sistema di gestione delle code per il cluster è stato installato SunGrid Engine In questo modo un job sottomesso al nodo Grid OAC di Napoli quando arriva sul Computing Element viene schedulato e sottomesso tramite OpenPbs al WorkerNode che lo risottomette attraverso SunGridEngine al Cluster.

2.2 Il Workload Management System di LCG

Lo scopo del Workload Management System è lo scheduling distribuito dei job in ambiente Grid e le funzionalita' offerte sono la:

- sottomissione dei job
- l'esecuzione dei job
- il monitoring dello stato del job
- il recupero dell'output

Il Workload Management System (WMS) comprende una serie di componenti del middleware di Grid responsabili della distribuzione e della gestione dei task sulle risorse di Grid, in modo tale che le applicazioni vengano eseguite in maniera efficiente e affidabile:

- User interface (UI): il punto di accesso a Grid per gli utenti
- Resource Broker (RB): il servizio di scheduling distribuito
- Job Submission Service (JSS): il servizio di job submission
- Logging and Bookkeeping Service (LB): il servizio di job monitoring

Il Resource Broker ha il compito di individuare la risorsa di computing "migliore" su cui sottomettere il job dell'utente; interagisce con Information Service e Data Management Service. Il Computing Element che viene selezionato deve soddisfare i requirement specificati nella descrizione del job (JDL file) e se più Computing Element soddisfano i requirement allora viene scelto il Computing Element con Rank più alto.

2.2.1 Logging and Bookkeeping: un servizio di job monitoring

Il servizio di Logging and Bookkeeping (L&B) traccia i job in termine di eventi, come ad esempio stato di sottomissione, determinazione di un CE, inizio dell'esecuzione. Gli eventi sono passati ad una componente fisicamente vicina al L&B (locallogger) per evitare problemi di rete. Queste componenti conservano gli eventi in un disco locale e hanno solo la responsabilità di consegnarli in seguito. La destinazione di un evento è uno dei bookkeeping servers assegnato in maniera statica ad un job appena viene sottomesso. Il server processa l'inizio degli eventi per dare una panoramica dello stato

del job (Submitted, Running, Done), che può contenere anche informazioni relative ai vari attributi. L'utente può eventualmente registrarsi per avere le notifiche su un particolare cambiamento di stato del job.

2.3 La Pipeline: un'applicazione astronomica in Grid

Vi sono diverse recenti iniziative internazionali mirate a far fronte all'enorme mole di dati astronomici che saranno prodotti nei prossimi anni dai telescopi a grande campo (*wide-field*) e dai rilevatori CCD a mosaico di nuova generazione. Un esempio di queste nuove macchine è il *VLT Survey Telescope* (VST), un telescopio interamente costruito dall'Osservatorio Astronomico di Capodimonte (OAC), dedicato all'imaging *wide-field* (47). Gli Osservatori di Padova e Capodimonte si sono occupati di realizzare la camera di cui sarà equipaggiato il telescopio, chiamata OmegaCAM. La camera a mosaico è costituita di 32 CCD per un totale di 256 Megapixels e produrrà tipicamente 100 Gbyte di dati grezzi per notte. Il consorzio europeo ASTRO-WISE (**?**), finanziato dall'Unione Europea, cui partecipa l'OAC, si sta occupando delle problematiche legate alla riduzione, all'analisi e allo storage dei dati VST. Una pipeline ad-hoc è in costruzione e si basa su di un'architettura parallela. Le tecnologie di calcolo distribuito sono la naturale soluzione al problema della riduzione delle immagini di grande formato ottenute con camere costituite da un mosaico di CCD. Nel caso di camere a mosaico, ogni immagine consiste di blocchi, anche detti estensioni, acquisiti con CCD indipendenti. Questa struttura per estensioni è particolarmente adatta per l'elaborazione parallela; ogni estensione, infatti, può essere ridotta e calibrata individualmente, prima di essere combinata a formare l'immagine scientifica, ed ha, quindi, una natura che in gergo tecnico viene definita come "embarrassingly parallel". Nel nostro caso poichè OmegaCAM è costituita da 32 CCds, ognuno da 2kx4k, l'immagine grezza in arrivo viene divisa in 32 estensioni, una per ogni CCD, che vengono successivamente elaborate da un nodo computazionale; solo alla fine della pipeline i larghi mosaici sono ricostituiti nella fase di coaddition.

I dati raw saranno sottoposti ad un primo livello di qualità (QC0) che consiste sostanzialmente nel verificare l'integrità dei dati e se corrispondono i requirement degli utenti. Le immagini che passano il primo controllo saranno inserite in un archivio locale e poi processati (47). La figura 2.3 mostra un modello per elaborare le immagini del VST, nel quale si può distinguere la fase di pre-riduzione e calibrazione, che includono la creazione dei frame di calibrazione, la fase della calibrazione astrometrica e fotometrica, la co-

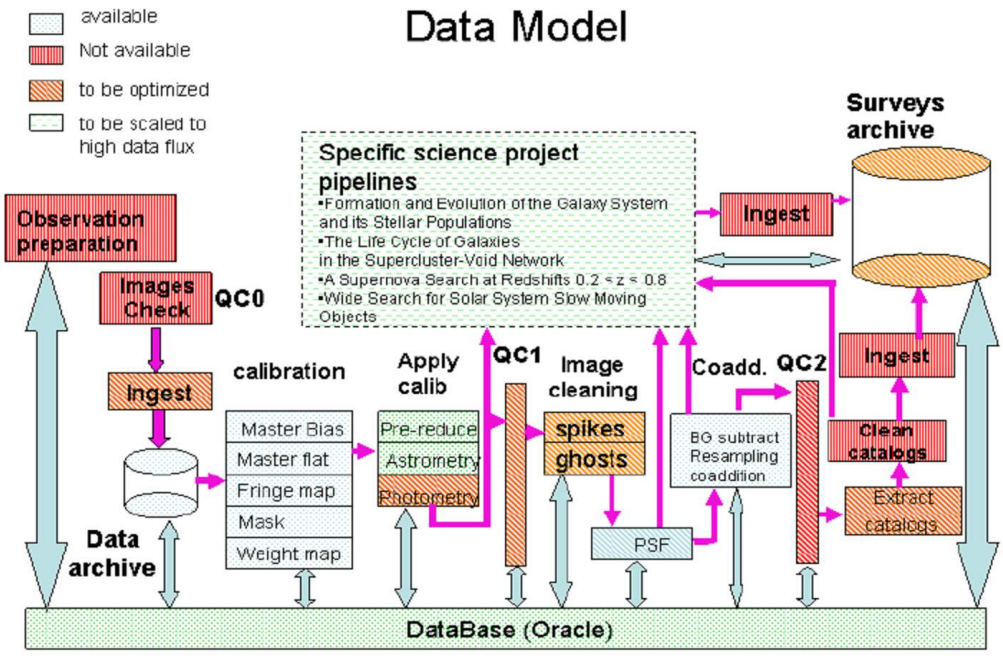

Figura 2.3: Tempi di esecuzione di 32 job in parallelo

addition, e in fine l'estrazione del catalogo. Durante queste fasi, i dati sono sottoposti ad altri due controlli per poter soddisfare ogni specifica dell'utente che produrre una pipeline.

La pipeline è un'applicazione modulare e possiamo schematizzare i suoi step nel modo seguente (4):

- *Master Bias production (parallel)*
- *Master Flat field production (parallel)*
- *Master Calibration Correction (parallel)*
- *Source extraction (parallel)*
- *Astrometric calibration (catalogue space)*
- *Fotometric calibration (catalogue space)*
- *Background subtraction (parallel)*
- *Re-sampling (parallel)*
- *Images co-addition (serial)*

- *Source extraction (catalogue space)*

2.4 Studio di fattibilità della tecnologia Grid per la pipeline Astro-wise

I dati di input di una pipeline sono immagini grezze costituite ognuna da 32 CCD che vengono elaborate seguendo una serie di step, alcuni dipendenti dal precedente altri disaccoppiati. Nell'ambito delle riduzione dei dati VST è stata analizzato l'uso di Grid per l'elaborazioni di immagini astronomiche wide-field. La pipeline Astro-wise (56) include elaborazioni parallele, storage distribuiti e database (fig.2.4). Per quanto riguarda gli storage distribuiti, la pipeline Astro-wise usa un meccanismo peer-to-peer per avere accesso agli storage distribuiti. Si è studiata la possibilità di integrare nella pipeline, elementi come storage e replica manager, richiesti dall'infrastruttura Grid. Questo rappresenta un cambiamento cruciale sia per gli utenti di Grid che per quelli della pipeline, in quanto l'infrastruttura Grid può fornire un accesso standard agli storage, non solo per la pipeline ma anche per altre applicazioni astronomiche Grid-based. Per quanto riguarda l'aspetto del calcolo distribuito o parallelo, l'attuale versione della pipeline Astro-wise fornisce uno scheduler XML-based, per distribuire i job sui diversi nodi computazionali. Contrariamente a quanto si può pensare gli studi di fattibilità sull'inserimento di alcuni elementi dell'infrastruttura Grid all'interno della pipeline Astro-wise hanno portato a concludere che è necessario un impegno considerevole per integrare la pipeline in Grid (28). Questo è dovuto al fatto che la pipeline di Astro-wise usa un approccio orientato agli oggetti, che tiene traccia delle diverse classi di oggetti durante l'image processing. Nel caso di un esecuzione remota, la pipeline mantiene le relazioni tra gli oggetti con un meccanismo implementato dallo scheduler di Astro-wise. Un meccanismo di questo tipo deve essere conservato nell'uso di uno Grid scheduler.

Test e performance in Grid

Nell'ambito dell'integrazione del data reduction nell'infrastruttura Grid sono stati realizzati alcuni test utilizzando un prototipo di pipeline che usa Grid Storage Element (SE) per conservare i dati di input raw, cosi come l'output ottenuto con la preriduzione e la calibrazione (47). Per elaborare i dati in parallelo è stato utilizzato il job scheduler di Grid. In particolare la pipeline è formata da 32 CCD che vengono elaborate in maniera indipendente e in parallelo. Il prototipo della pipeline ha permesso di misurare alcuni critici ove-

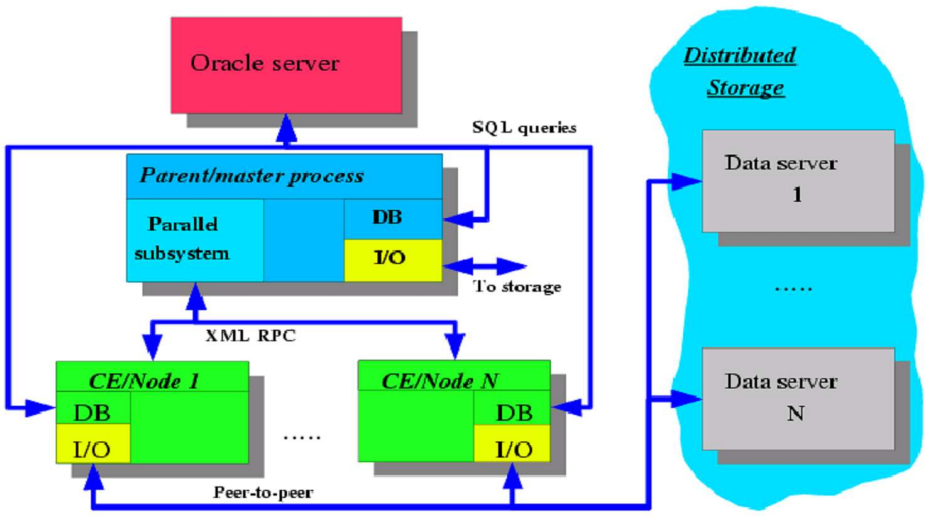

Figura 2.4: AstroWise processing model

rheads dell'esecuzione in Grid. Un primo test è stato eseguito considerando un job "dummy".

Quello che si osserva è un overhead di 5 minuti (Fig.2.5). Un altro parametro critico è la scalabilità per le esecuzioni parallele. Per misurare questo parametro, la calibrazione è stata calcolata usando solo i primi step della pipeline. Il tempo di esecuzione per 32 job lanciati in parallelo è 23 volte più veloce della stessa esecuzione in sequenziale (Fig.2.6).

Resta comunque da sottolineare che poichè i tempi di esecuzione sono significativamente differenti, alcuni job restano in esecuzione mentre la maggior parte dei 32 sono terminati e questo non permette di eseguire step successivi della pipeline. Un altro overhead da non sottovalutare, in particolare per le applicazioni che recuperano ed elaborano grandi quantità di dati, sono le operazioni di I/O sulla rete WAN (Wide Area Network) (Fig.2.7).

Il vantaggio che si vuole apportare è quello di automatizzare la sottomissione di tutti gli step della pipeline descrivendo tutte le informazioni relative ai dati di I/O, agli step da eseguire, alle priorità e, ai percorsi che sono autorizzati all'interno di un flusso di esecuzioni. Tutto questo è realizzabile attraverso la costruzione di workflow che descrivono i task secondo un ordine ben definito e attraverso tecniche di gestione sempre più sofisticate dei sistemi di Grid Workflow.

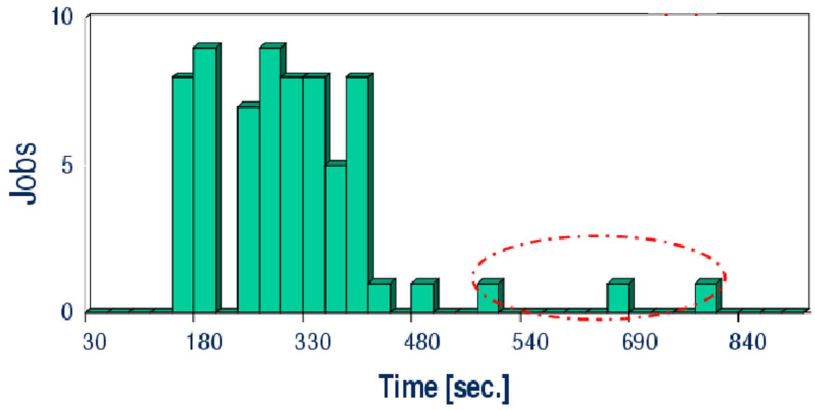

Figura 2.5: Grid scheduler overhead con una sottomissione dei job “dummy”

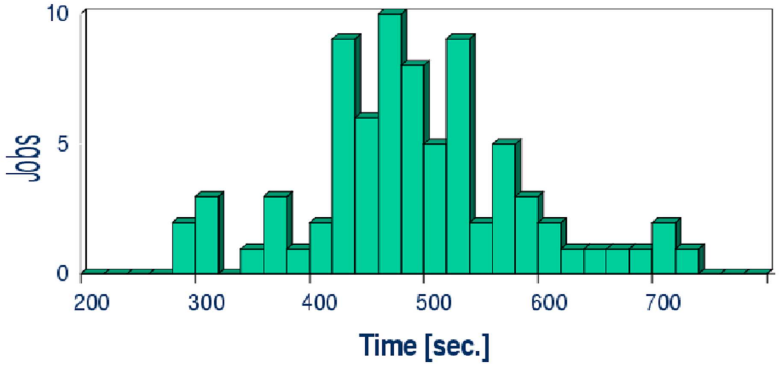

Figura 2.6: Tempi di esecuzione di 32 job in parallelo

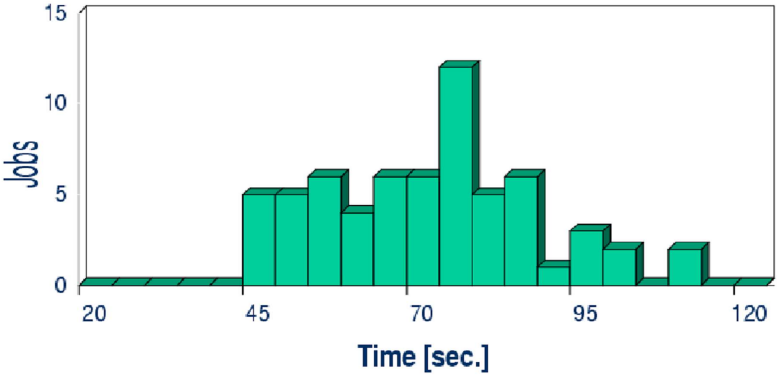

Figura 2.7: I/O overhead per 5 raw-bias

Capitolo 3

I sistemi di management dei workflow

3.1 Classificazione dei sistemi di gestione del Workflow

La nascita e l'espansione delle tecnologie Grid hanno portato allo sviluppo di applicazioni sempre più complesse per gestire ed elaborare dati e per eseguire esperimenti scientifici su risorse geograficamente distribuite. In questo modo è nata la necessità di creare ed eseguire complessi workflow e in particolare di sviluppare sistemi di gestione dei workflow, principalmente per le tecnologie Grid.

Il Grid Workflow è una collezione di task elaborati su risorse distribuite, secondo un ordine ben definito. Negli ultimi anni sono state sviluppate numerose tecniche di gestione dei workflow, specialmente nel campo del business e molti di questi approcci possono essere applicati al Grid Workflow per le applicazioni scientifiche. Si osservi che esistono delle differenze tra un workflow convenzionale e un workflow scientifico per Grid; i workflow scientifici sono basati su lunghe esecuzioni di task che coinvolgono un enorme flusso di dati ed utilizzano risorse dinamiche ed eterogenee con la partecipazione di diverse organizzazioni, mentre i workflow relativi al business coinvolgono brevi transazioni con piccole quantità di dati. I sistemi di gestione dei workflow che si occupano di definire, gestire ed eseguire workflow per applicazioni Grid sono in continua espansione ed offrono molti vantaggi come:

- possibilità di costruire dinamicamente applicazioni che sfruttano risorse distribuite

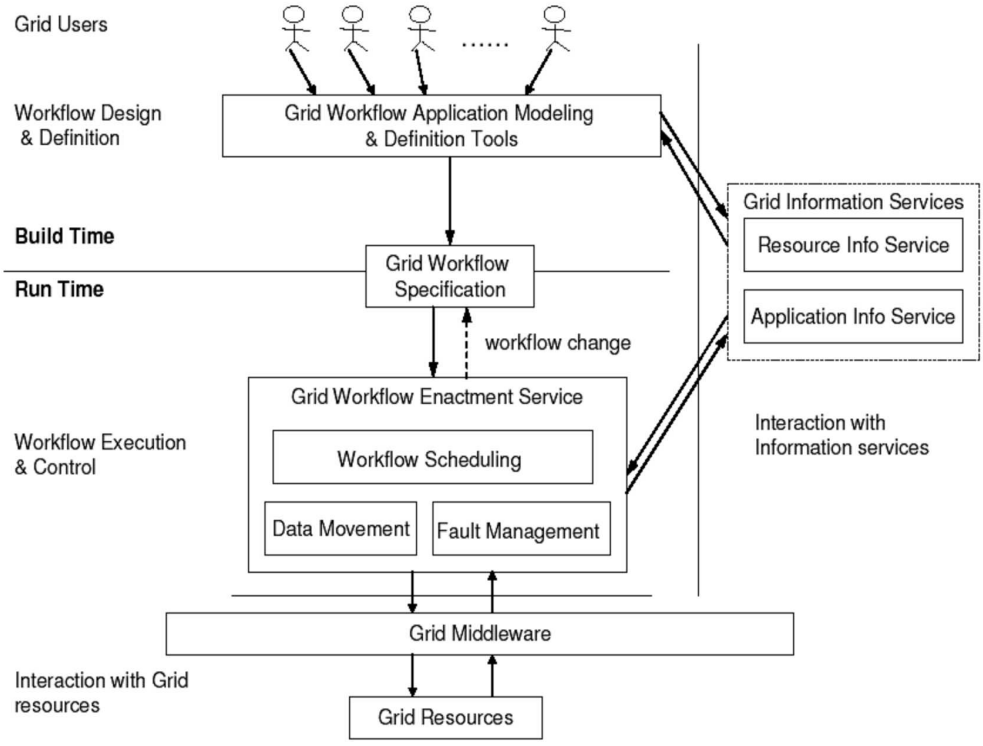

Figura 3.1: Architettura di un Grid Workflow Management System

- utilizzo di risorse localizzate in diversi domini per migliorare il trattamento dei dati o ridurre i costi di esecuzione
- esecuzioni estese su vari domini amministrativi per ottenere specifiche capacità di elaborazione
- integrazione di gruppi coinvolti nella gestione di parti differenti di esperimenti che costituiscono un workflow, promuovendo così le collaborazioni tra le organizzazioni

L'architettura di un sistema per la gestione del Grid Workflow costituita da una parte dedicata alla definizione e alla creazione di modelli di workflow con le loro dipendenze *(built time)* e da una parte dedicata alla gestione dell'esecuzione dei workflow e all'interazione con le risorse Grid per l'elaborazione dei workflow *(run time)*.

La figura 3.1 mostra l'architettura e le funzionalità delle varie componenti di un sistema di workflow per Grid basato su un modello proposto dal Workflow Management Coalition (WfMC) (41). L'utente interagisce con

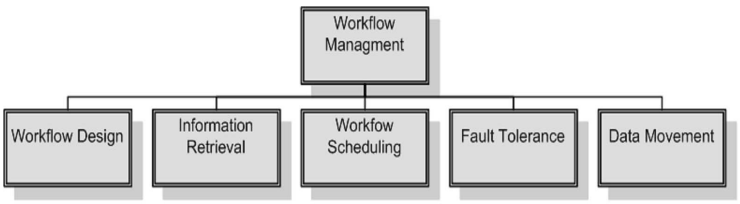

Figura 3.2: Elementi di un sistema di gestione dei Grid workflow

strumenti che generano un modello di workflow (Workflow Design & Definition) che viene poi sottomesso ad un servizio per l'esecuzione (Workflow Execution & Control).

Tale servizio fornisce una serie di funzioni come lo scheduling, la fault tollerance e lo spostamento dei dati ed è costruito su un middleware Grid, come ad esempio Globus Toolkit e UNICORE , attraverso il quale vengono utilizzate le risorse di Grid. Sia nella fase di costruzione del modello che in quella dell'esecuzione, le informazioni sulle risorse e sulle applicazioni sono recuperate interrogando servizi Grid dedicati all'informazione.

Per comprendere meglio l'architettura e gli sviluppi dei sistemi di gestione dei workflow consideriamo la seguente classificazione che caratterizza gli approcci dei vari sistemi di gestione dei workflow nell'ambito del Grid Computing (65).

Un sistema di gestione del Grid workflow si può dividere in cinque elementi (Fig.3.2):

- modello
- recupero delle informazioni
- scheduling
- tolleranza
- spostamento dei dati.

3.1.1 Il design di un workflow

Il design di un workflow management si divide in (Fig.3.3):

Workflow structure indica le relazioni temporali tra i task. In generale possiamo rappresentare un workflow come un Directed Acyclic Graph (DAG)

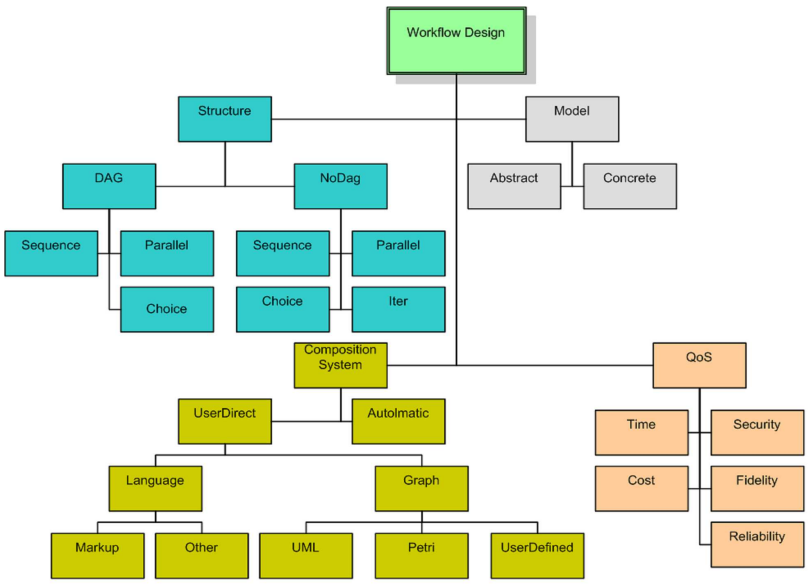

Figura 3.3: Classificazione del Workflow Design

e un non-DAG.

- Un workflow basato su DAG può essere suddiviso in *sequence, parallel*, e *choice*. *Sequence* è una serie di task ordinati in cui un task inizia solo dopo che il precedente è terminato; *parallel* rappresenta dei task eseguiti contemporaneamente; ed infine il controllo di *choice* permette di selezionare l'esecuzione di un task in tempo reale se le condizioni ad esso associate sono vere.

- Il workflow basato su non-DAG ammette, oltre alle condizioni valide per la struttura DAG anche una quarta caratteristica, l'*iter*, attraverso la quale i task possono essere ripetuti all'interno di un blocco di iterazioni. Le iterazioni non sono altro che dei loop o cicli e rappresentano delle strutture molto frequenti nelle applicazioni scientifiche in cui spesso più task devono essere eseguiti ripetutamente.

Workflow Model/Specification definisce un workflow includendo la definizione dei task e della struttura. Si possono considerare due principali tipi di modelli chiamati astratto e concreto (13).

- Il modello astratto descrive un worfkflow in una forma appunto astratta, nella quale non sono specificati riferimenti a particolari risorse di

Grid per l'esecuzione del task. Tale modello offre all'utente un modo flessibile per definire i workflow senza necessariamente conoscere in dettaglio le implementazioni a basso livello. I task di un modello astratto sono portabili e possono essere applicati su molti adeguati servizi di Grid utilizzando meccanismi di discovery e assegnazione.

- Il modello concreto lega i task a specifiche risorse. In alcuni casi il modello concreto include lo spostamento dei dati per rappresentare l'input e l'ouput del calcolo e pubblicare dati recenti in uno spazio di storage dedicato ad una VO (13).

Workflow Composition System permettono all'utente di assemblare le componenti del workflow. Un sistema di composizione di un workflow offre una visione ad alto livello della costruzione di un workflow per le applicazioni di Grid, nascondendo la complessità intrinsecamente propria di ogni sistema di griglie.
Il sistema di composizione può essere costruito seguendo varie opzioni:

- composizione automatica di un workflow
- User-directed permette all'utente di editare direttamente il workflow utilizzando un modello basato su un linguaggio di markup oppure un modello grafico.

Workflow QoS si interessa dell'aspetto relativo alle richieste delle risorse da parte degli utenti, per l'esecuzione dei workflow.
Nell'ambito di Grid, esiste un nutrito numero di risorse simili o persino equivalenti; l'utente Grid può selezionare le risorse più adatte ed usarle per il proprio workflow di applicazioni.
Tali risorse forniscono le stesse funzionalità ma ottimizzano differenti parametri del Quality of Service. Inoltre, gli utenti o le stesse applicazioni possono avere diverse aspettative ed esigenze che non sono banalmente soddisfatte da un sistema di gestione dei workflow.
E' necessario quindi che il sistema di workflow si occupi non solo di gestire le caratteristiche funzionali del workflow ma raccolga anche le richieste per i servizi di QoS.
A tale scopo, l'utente deve essere in grado di specificare le sue richieste di QoS all'interno del disegno del workflow.
I vincoli di un QoS possono essere assegnati a livello di task, dove l'utente specifica le sue richieste per i singoli task che compongono un workflow, oppure a livello di workflow, dove l'utente definisce le richieste su tutto il workflow.
Il modello di QoS comprende 5 dimensioni:

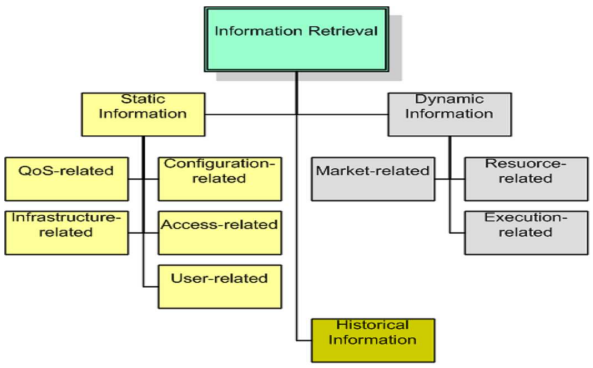

Figura 3.4: Classificazione del Recupero delle Informazioni

- Time è una misura fondamentale per le performance e si riferisce al tempo totale richiesto per completare l'esecuzione di un workflow
- Cost rappresenta il costo associato all'esecuzione del workflow, comprendendo anche il costo per il sistema di gestione dei workflow e il costo d'uso delle risorse Grid per eseguire i task
- Fidelity si riferisce alla misura della qualità dell'output nell'esecuzione di un workflow
- Reliability è relativo al numero di fallimenti per l'esecuzione di un workflow
- Security riguarda l'aspetto confidenziale dell'esecuzione di un workflow e l'affidabilità delle risorse

3.1.2 Recupero delle Informazioni

Un sistema di gestione di workflow non esegue direttamente i task ma rappresenta solo un coordinatore delle esecuzioni per le risorse Grid. Per assegnare i task alle risorse più adatte, esistono alcune tecniche dedicate al recupero delle informazioni relative alle disponibilità di risorse sulla griglia. Tali informazioni possono essere recuperate in tre modi (Fig.3.4):

Static information è una tecnica basata sulle informazioni che non cambiano nel tempo relativamente all'infrastruttura, alla configurazione, agli accessi, agli utenti, e infine alle richieste sulla qualità dei servizi. Parametri statici sono ad esempio il numero di processori, le librerie e i sistemi operativi, le

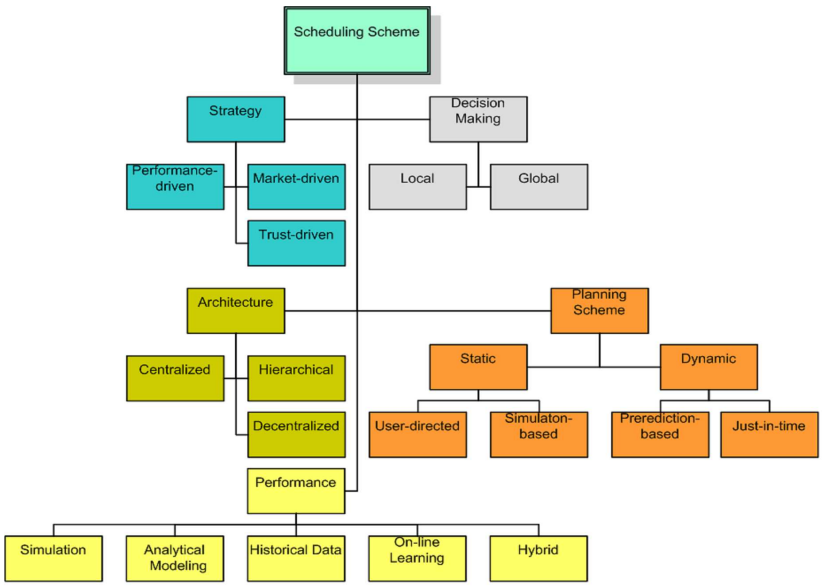

Figura 3.5: Cloassificazione dello Scheduling

autenticazioni e il costo d'uso.

Dynamic information esprime lo stato delle risorse Grid come il carico medio di un cluster, lo spazio disco disponibile, l'utilizzo della CPU, e i processi attivi. Le informazioni dinamiche ragguagliano anche sulle esecuzioni dei task e le informazioni relative al mercato, come i prezzi dinamici delle risorse.

Historical Information si basa sugli eventi precedentemente accaduti sfruttando l'history delle performance e delle esecuzioni delle risorse Grid e delle applicazioni. Solitamente, il sistema di gestione dei workflow analizza le informazioni storiche per effettuare delle previsioni sui comportamenti futuri delle risorse e delle applicazioni su un set assegnato di risorse. Historical information può essere anche usato per migliorare l'affidabilità delle esecuzioni successive.

3.1.3 Lo Scheduling

Il workflow scheduling coordina diversi sistemi di gestione locali data l'eterogeneità delle risorse di grid in termini di configurazioni e politiche locali. Analizziamo lo scheduling del workflow dal punto di vista dell'architettura, delle decisioni, della pianificazione, della strategia e delle performance

(Fig.3.5):

L'architettura di un'infrastruttura di scheduling è molto importante per problemi di scalabilità, qualità e performance del sistema.

- In un'architettura centralizzata un solo scheduler prende le decisioni per tutti i task di un workflow, raccogliendo le informazioni di un intero workflow e delle risorse disponibili. Uno scheduler di questo tipo è molto efficiente perchè contiene tutte le informazioni necessarie, ma non è scalabile rispetto al numero di task e al numero di processori.

- In un'architettura decentralizzata un unico scheduling mantiene le informazioni relative solo ai sotto-workflow. In questo modo, l'architettura risulta molto scalabile su un grande numero di task perchè i task gestiti da ogni scheduler sono limitati. Tuttavia la migliore decisione presa per un workflow parziale può portare ad un peggioramento delle performance per il workflow totale.

- L'architettura gerarchica è costituita da un gestore centrale e va- ri scheduler per i sotto-workflow. Il manager controlla e l'esecuzione del workflow e assegna i sotto-workflow agli scheduler a più basso livello Ogni scheduler di basso livello è responsabile dei propri task. Il vantaggio principale è che possono essere assunte diverse politiche di scheduling anche se il fallimento del gestore centrale comporterebbe il fallimento dell'intero sistema.

Decision. Per una buona politica di scheduling è importante decidere la migliore distribuzione del workflow sulle risorse, per tutti i tipi di applicazioni. Solitamente, decisioni di questo tipo sono fatte in base alle informazioni sul task corrente (local decision) o sull'intero workflow (global decision) (14).

Planning scheme riguarda la trasformazione del workflow da astratto a concreto. Lo schema può essere classificato in statico e dinamico.

- Nel caso statico il modello concreto viene generato prima dell'esecuzione, in accordo con le informazioni in corso sull'ambiente di esecuzione, senza tenere conto dei cambiamenti di stato dinamici delle risorse. Gli approcci in uno schema statico possono *user-directed* e *simulatin-based*. Nel primo caso, gli utenti imitano un processo di scheduling e decidono le risorse su cui distribuire i task, basandosi sulle loro conoscenze e preferenze, e tenendo conto anche di alcuni criteri di performance. Nel secondo caso, viene eseguito il migliore scheduling su un task di

simulazione su un set di risorse prima di far partire l'esecuzione del workflow. La simulazione si può basare sulle informazioni statiche o sui risultati di una stima delle prestazioni.

- In contrapposizione, nel caso dinamico si usano sia le informazioni dinamiche che quelle statiche. Lo schema è di tipo *prediction-based* cioè vengono utilizzate sia le informazioni dinamiche che i risultati di una previsione.

Le strategie della maggior parte degli scheduling sono classificate in tre categorie:

- Performance-driven trova una distribuzione del workflow sulle risorse che raggiunga le performance ottimali di esecuzione, come ad esempio i tempi di esecuzione minimi.

- Market-driven utilizza un modello di mercato per gestire l'allocazione delle risorse applicando principi economico-computazionali e stabilendo un mercato elettronico tra i sistemi di gestione del workflow e il fornitore delle risorse partecipanti.

- Trust-driven seleziona le risorse basandosi sul loro livello di fiducia. Questo modello è basato su attributi come sicurezza, rispettabilità accumulata e capacità di auto-difesa.

Performance estimation è un aspetto critico per la stesura di una buona strategia di scheduling, specialmente per la costruzione di un scheduling di workflow preliminare. La tecnica di stima delle performance permette di pronosticare il comportamento dei task in un workflow su risorse eterogenee e distribuite. In questo modo si può decidere dove e come eseguirli. Le stime possono essere calcolate in base ai seguenti approcci: simulazione, analisi, ibrido, utilizzo dei dati storici, istruzioni on-line. Vediamoli in dettaglio:

- La simulazione mette a disposizione un ambiente di risorse simulato per eseguire i task di un workflow prima della sua reale esecuzione.

- Nel modello analitico, lo scheduler predice le performance di un task su un set di risorse basato su una metrica analitica.

- Historical data si basa su dati storici per fare una stima delle performance (38)

- On line learning cattura informazioni dalle esperienze on-line senza avere conoscenze precedenti sull'ambiente dinamico

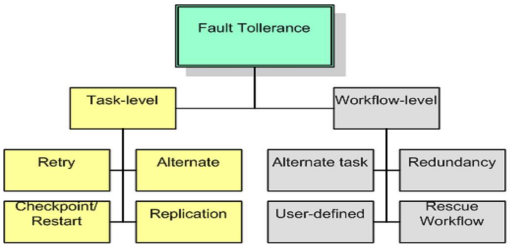

Figura 3.6: Classificazione della Faut Tollerance

3.1.4 La Fault Tolerance

In un ambiente Grid esistono disparati motivi che causano il fallimento dell'esecuzione di un workflow quali la variazione della configurazione dell'ambiente di esecuzione, mancanza di disponibilità dei servizi richiesti, condizioni di sovraccarico delle risorse, errori di calcolo e nelle componenti di rete. Un sistema di gestione dei workflow deve essere in grado di identificare e trattare gli errori e dare un supporto affidabile alle esecuzioni, in presenza di concorrenza e fallimenti. La tecnica di gestione dei fallimenti puo essere divisa in task-level e workflow-level (Fig.3.6).

Il Task-level identifica gli errori dei task in un workflow e si può catalogare in quattro dimensioni:

- Retry è una delle tecniche più semplici per il recupero degli errori in quanto cerca di eseguire il task sulla stessa risorsa dopo l'avvenuto fallimento (43)
- La tecnica Alternate sottomette i task falliti ad un'altra risorsa
- Il checkpoint/restart migra il task in maniera trasparente su un'altra risorsa in modo tale da continuare l'esecuzione dal punto in cui è fallita (26)
- La tecnica di replication esegue il task contemporaneamente su diverse risorse Grid per assicurarsi la conclusione senza errori di almeno uno dei task

Il Workflow-level modifica la struttura del workflow come il flusso di esecuzione, per trattare le condizioni di fallimento. Una tecnica di questo tipo prende in considerazione task alternati, ridondanza, gestione utente delle eccezioni e workflow di salvataggio.

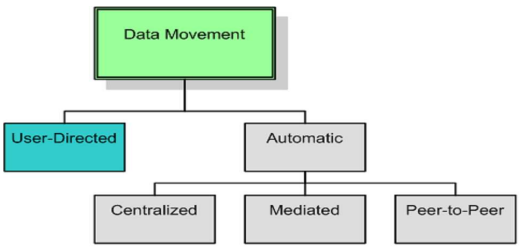

Figura 3.7: Classificazione del Data Movement

- I task alternati prevedono più implementazioni per l'elaborazione di un task con diverse caratteristiche di esecuzione. In questo caso, se un task fallisce viene eseguita un'altra implementazione di quel task
- La tecnica della ridondanza, invece, esegue le varie implementazioni contemporaneamente
- La gestione delle eccezioni permette all'utente di specificare un trattamento speciale per alcuni fallimenti
- Il salvataggio di un workflow ignora i task falliti e continua il resto dell'esecuzione fino al termine di tutti i task. La descrizione di tutti i fallimenti viene salvata in un file che puo essere consultato successivamente per una nuova sottomissione.

3.1.5 Il Data Movement

In un'applicazione per le griglie computazionali sia i dati di input che i dati di output devono essere organizzati in un'area di storage prima di essere elaborati. In particolare i dati di output possono essere richiesti da altri sotto-task che sono elaborati su risorse diverse. Questo implica che i dati intermedi non devono essere conservati e organizzati nello stessa postazione Grid sulla quale vengono eseguiti i task. La gestione dei file può essere eseguita in due modi diversi (Fig.3.7):

User-directed implica che un sistema richiede direttamente all'utente di gestire il trasferimento dei dati intermedi.

Automatic è una tipologia di gestione in cui il sistema automaticamente dirige il trasferimento dei dati. Un approccio automatico può essere affron-

tato considerando tre tipi di architettura: centralizzata, decentralizzata o peer-to peer.

- L'approccio centralizzato utilizza un punto centrale che colleziona i risultati dell'esecuzione di un task e li trasferisce all' entità che contiene l'elaborazione del task successivo. Un approccio di questo tipo è molto semplice da implementare se il workow non richiede grandi elaborazioni di dati.

- L'architettura decentralizzata prevede un sistema di gestione dati distribuito. Ogni step è registrato in un servizio di replica catalog in modo tale che i dati di input, per ogni task, sono disponibili interrogando il servizio. Quest'approccio è molto scalabile e adatto alle applicazioni che prelevano i dati per usi successivi.

- La tecnica peer-to-peer trasmette i dati alle risorse destinatarie senza passare per risorse intermedie o coinvolgere servizi esterni. Questo ci permette di migliorare in modo considerevole i tempi di trasmissione e riduce i colli di bottiglia, ma è sufficientemente difficile da implementare in quanto richiede ai nodi di Grid di essere capaci di gestire i dati e i servizi per il trasferimento.

3.1.6 Lo stato dell'arte

I sistemi di workflow che gestiscono le applicazioni sono stati trattati in maniera esaustiva nell'ambito di processi dei business, ma risultano ancora un campo inesplorato relativamente al Grid Computing. Sono stati fatti molti sforzi per creare e migliorare i linguaggi di workflow e dell'architettura ed esistono vari progetti sui sistemi di workflow. Molte di queste architetture sono indipendenti dal middleware e ne sfruttano i vantaggi di alcuni servizi, come l'accesso alle risorse, la sicurezza e la gestione delle repliche. Di seguito riportiamo i vari progetti attualmente esistenti nell'ambito dei workflow:

DAGMan (26) (6) è stato sviluppato per effettuare lo scheduling dei job su Condor system, utilizzando i DAG. Mentre Condor ha lo scopo di determinare le risorse disponibili per l'esecuzione dei job, DAGMan gestisce le dipendenze tra i job. DAGMan utilizza i DAG come struttura dati per rappresentare le dipendenze tra i job. Ogni job è un nodo nel grafo e le connessioni rappresentano le loro dipendenze. Ogni nodo può avere un numero di nodi genitori o figli. I figli non possono essere eseguiti fino a quando i loro genitori non sono terminati. I cicli in cui due job discendono entrambi l'uno dall'altro sono proibiti perchè potrebbero causare una situazione di stallo.

Inoltre DAGMan non supporta in maniera automatica il data movement intermedio, quindi gli utenti devono specificare i comandi per il trasferimenti prima e dopo l'elaborazione.

Pegasus (14) mappa un workflow astratto su una serie di risorse Grid disponibili, generando un workflow eseguibile. Un workflow astratto descrive il calcolo in termini di file logici e applicazioni logiche indicando le dipendenze nella forma di un DAG. Prima di effettuare il mapping, Pegasus riduce il workflow astratto riutilizzando un dataset prodotto da altre Virtual Organization. Inoltre è dimostrato che è molto più economico accedere ai risultati processati che produrre un nuovo dataset. Pegasus consulta i vari servizi di informazione di Grid per trovare le risorse, il software e i dati che sono utilizzati nel workflow. In Pegasus il workflow può essere generato descrivendo i dati che si desiderano attraverso la tecnologia dell'intelligenza artificiale, indicando le dipendenze di calcolo nella forma dei DAG.

Il progetto **Taverna** (43) ha sviluppato un tool per la creazione di workflow bioinformatici per la comunità di life-science. Il tool non è basato su Globus. In Taverna i modelli di dati possono essere rappresentati sia in formato grafico che con un linguaggio XML-based. Il modello dei dati consiste di input, output, processori, flusso di dati e flussi di controllo. Taverna fornisce anche un ambiente a finestre user-friendly, per manipolare, eseguire e monitorare i workflow e selezionare le risorse disponibili.

Allo stesso modo **GrADS** (51) (7)(Grid Application Development Software) fornisce dei tool e un ambiente per sviluppare e eseguire le applicazioni su Grid utilizzando delle funzioni che, richiamate all'interno del proprio codice, permettono la costruzione del workflow tramite una struttura DAG. Anche Grads non si basa sul toolkit di Globus.
GrADS fornisce uno scheduling per creare una mappa dei task delle applicazioni del workflow su una serie di risorse, basandosi su Grid scheduling e metodi di rescheduling. Questo metodi si basano su una funzione che minimizza la complessità di tempo dei job dei workflow. Lo scheduler ottiene le informazioni sulle risorse attraverso un servizio come MDS e NWS e localizza il software necessario attraverso delle query a GrADS Information Service (GIS). Grads utilizza Autopilot per monitorare le performance dell'accordo tra la richiesta delle applicazioni e la capacità delle risorse. Se l'accordo viene violato il rescheduling di GrADS esegue le dovute correzioni.

Altri progetti per la gestione dei workflow nell'ambito del Grid Computing sono **GridFlow** (8) e **UNICOREplus** (1). Entrambi non si basano sul

middleware di Globus e in particolare GridFlow ha alcune limitazioni sul flusso di controllo della struttura in quanto è basata sui DAG. UNICORE-plus fornisce un accesso sicuro alle risorse distribuite. UNICORE plus è una conseguenza del progetto. UNICORE provvede alla creazione di una ambiente di programmazione per l'utente per disegnare ed eseguire un flusso di job. Con UNICORE un job o un gruppo di job che può essere eseguito su un sito UNICORE può contenere altri job e/o gruppi di job. Con la versione 4 di UNICORE sono stati aggiunti più avanzati flussi di controllo, che includono esecuzione di condizioni come "if-then-else", esecuzioni di ripetizioni come "do-n" ed esecuzioni di ripetizioni condizionali come "do-repeat".

Il progetto **Triana** (53) (54) è un ambiente visivo per l'analisi dei dati che permette di costruire un modello di workflow con un'interfaccia visiva basata su una struttura aciclica (NonDAG). E' stato integrato in Grid attraverso l'interfaccia Gridlab GAT (Grid Application Toolkit). Il sistema di composizione non è basato sul un linguaggio di markup come XML ma su una struttura grafica.

Nella figura 3.8 riportiamo il posto assunto dai vari progetti sopraelencati, relativamente alla tassonomia esposta nei paragrafi precedenti.

3.2 I requirement degli astrofisici e i limiti degli attuali Sistemi di Workflow

Ci sono una serie di progetti focalizzati sullo sviluppo del linguaggio di workflow. Attualmente l'architettura Workflow Enactment Engine (WFEE) è basata su un linguaggio di workflow chiamato xWFL mentre il Grid Service Flow Language (GSFL) supporta le specifiche per la costruzione dei workflow nell'ambito della Grid per OGSA.
Entrambi i linguaggi però non supportano i cicli e le limitazioni sul flusso di controllo.
Il workflow di DAGMan è limitato alla progettazione di grafi aciclici e non supporta costrutti avanzati per l'esecuzione parallela. La maggior parte dei lavori esistenti soffre di inconvenienti come limitazioni del flusso di controllo, limitazioni per i meccanismi di espressioni parallele, meccanismi per il flusso dei dati ristretto e costrutti a basso livello.
Basandoci sulla tassonomia esposta, la struttura di un sistema di gestione dei workflow che più si adatta alle esigenze delle applicazioni astrofisiche è definita scegliendo solo alcuni aspetti della classificazione (Fig.3.9).

Project	Workflow Design			Workflow Scheduling				Fault-tolerance	Data Movement
	Structure	Model	Composition System	Architecture	Decision Making	Planning Scheme	Strategies		
DAGMan [19]	DAG	Abstract	User-directed -Language-based	Centralized	Local	Dynamic -Just in-time	Performance-driven	Task Level -Migration -Retrying Workflow Level -Rescue workflow	User-directed
Pegasus [6]	DAG	Abstract	User-directed -Language-based Automatic	Centralized	Local Global	Static -user-directed Dynamic -Just in-time	Performance-driven	Based on DAGMan	Mediated
Triana [20]	Non-DAG	Abstract	User-directed -Graph-based	Decentralized	Local	Dynamic -Just in-time	Performance -driven	Based on GAT manger	Peer-to-Peer
ICENI [16]	Non-DAG	Abstract	User-directed -Language-based -Graph-based	Centralized	Global	Dynamic -Prediction-based	Performance-driven Market-driven	Based on ICENI middleware	Mediated
Taverna [17]	DAG	Abstract Concrete	User-directed -Language-based -Graph-based	Centralized	Local	Dynamic -Just in-time	Performance-driven	Task Level -Retry -Alternate Resource	Centralized
GrADS [3]	DAG	Abstract	User-directed -Language-based	Centralized	Local Global	Dynamic -Prediction-based	Performance-driven	Task Level in rescheduling work in GrADS, but not in workflows.	Peer-to-Peer
GridFlow [4]	DAG	Abstract	User-directed -Graph-based -Language-based	Hierarchical	Local	Static -Simulation-based	Performance-driven	Task Level -Alternate resource	Peer-to-Peer
UNICORE [1]	Non-DAG	Concrete	User-directed -Graph-based	Centralized	User-defined*	Static -User-directed	User-defined*	Based on UNICORE middleware	Mediated
Gridbus workflow [22]	DAG	Abstract Concrete	User-directed -Language-based	Hierarchical	Local	Static -User-directed Dynamic -Just in-time	Market-driven	Task Level -Alternate resource	Centralized Peer-to-Peer
Askalon [7]	Non-DAG	Abstract	User-directed -Graph-based -Language-based	Decentralized	Global	Dynamic -Just in-time -Prediction-based	Performance-driven Market-driven	Task Level -Retry -Alternate resource Workflow level -Rescue workflow	Centralized User-directed

Figura 3.8: Schema completo dei sistemi di workflow relativamente alla classificazione precedente

Relativamente ai problemi per la creazione di un workflow abstract, il sistema di workflow per gli astrofisici necessita di un linguaggio che copra la complessità dell'infrastruttura Grid e l'ambiente di esecuzione; un linguaggio che offra una serie di controlli e di costrutti per eseguire applicazioni astronomiche su Grid. Quindi, relativamente a questo problema, il workflow management system deve supportare la struttura Non-Dag per permettere la dinamicità del workflow con la possibilità di eseguire loop e istruzioni di controllo. Il sistema di composizione, deve essere di tipo user-direct, basandosi su un linguaggio di markup come XML, semplice e portabile.
La maggior parte dei progetti sono quasi indipendenti nell'esecuzione e nella gestione del workflow, per lo scheduling per la sottomissione e la gestione dei dati. L'architettura di WFEE relativa al progetto Gridbus ad esempio, è costituita dalle seguenti componenti: workflow submission, workflow language

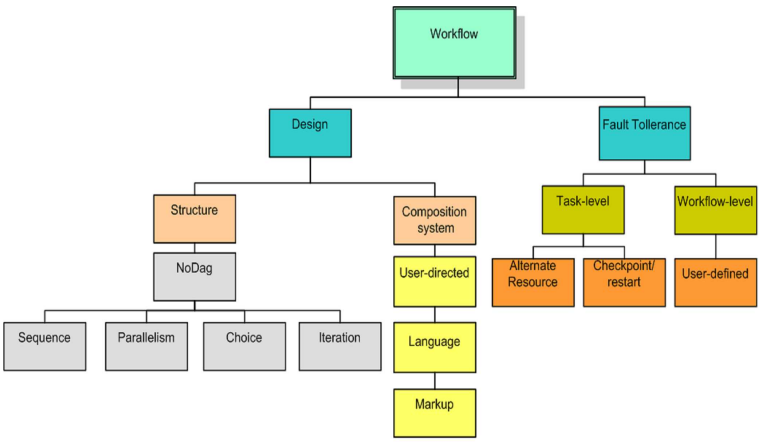

Figura 3.9: Un ipotetico sistema di workflow per i requirement astrofisici

parser, resource discovery, data movement e workflow scheduler. Il resto dei servizi sono ottenuti interagendo con le varie componenti di un'infrastruttura Grid.
Diversamente dagli attuali progetti, l'architettura di un sistema di workflow per le applicazioni astrofisiche vuole fornire un sistema per la creazione di complessi workflow orientato al Globus Toolkit e ai middleware esistenti, senza sostituire le componenti e i servizi forniti dalle attuali infrastrutture, come lo scheduler e il data movement.

3.3 AstroGrid

Un workflow ha l'obiettivo di completare le parti complesse di un lavoro. Se il lavoro potesse essere svolto in un unico step (ad esempio l'esecuzione di un singolo programma lanciato a linea di comando) diventerebbe difficilmente un workflow. Il primo aspetto di un workflow è che l'obiettivo necessiti di più di uno step per essere portato a termine, quindi abbia un grado di complessità. Un semplice esempio di workflow nell'ambito delle applicazioni astronomiche consiste nei seguenti passi:

- Fare una query ad un catalogo per cercare un'area del cielo
- Fare una query simile in un altro catalogo
- Unire i contenuti
- Analizzare il risultato

In AstroGrid (63) questo rappresenta un'indagine astronomica. Astrogrid è un progetto dell'UK eScience in collaborazione con numerosi gruppi dei maggiori centri di data archive dell'UK e sta realizzando il primo Osservatorio Virtuale del Regno Unito. Insieme con altri importanti progetti sugli osservatori virtuali mondiali (IVOA: International Virtual Observatory Alliance) AstroGrid sta creando una serie di sistemi software interagenti che permetteranno all'utente di interrogare più centri dati in modo trasparente, che forniranno potenti strumenti per nuove analisi e visualizzazioni e che forniranno uno standard per i servizi di consegna e di pubblicazione (59). AstroGrid sta sviluppando una struttura che permetterà:

- di migliorare la qualità, l'efficienza, la velocità e i costi di una ricerca astronomica on-line
- di fare confronti e integrazioni di dati
- di rimuovere le barriere dell'analisi di dati per una ricerca interdisciplinare
- di fare manipolazione di grandi set di dati in maniera facile e potente

L'obiettivo è quello di eliminare la complessità dei workflow (62). Ogni step di un lavoro solitamente rappresenta l'invocazione a programmi distinti e ogni tool richiede i propri input e output. Alcuni input possono essere dati astronomici e altri saranno informazioni di controllo per selezionare altri tool. Per AstroGrid il risultato finale dell'esecuzione del workflow è solitamente un file conservato in MySpace. Da questo punto di vista, il workflow di AstroGrid è esso stesso uno step intermedio.
Il disegno del workflow è organizzato come un documento XML e conservato in MySpace (Fig.3.10).

3.3.1 L'architettura di AstroGrid

Il linguaggio di workflow utilizzato da AstroGrid è completo di tutte le activities base per scrivere un workflow; esso consiste infatti di strutture gerarchiche costituite da parti chiamate steps, tools, flows, sequences. Inoltre contiene variabili, controlli logici e script per costruire i loop e branch. Quando un workflow inizia la sua esecuzione si parla di job. In particolare, un workflow diventa un job quando viene sottomesso al *Job Execution System* (JES). Ogni sottomissione rappresenta un job separato. In effetti, un job è un particolare workflow che non solo contiene il disegno così come è stato sottomesso, ma anche delle informazioni run-time sull'esecuzione.

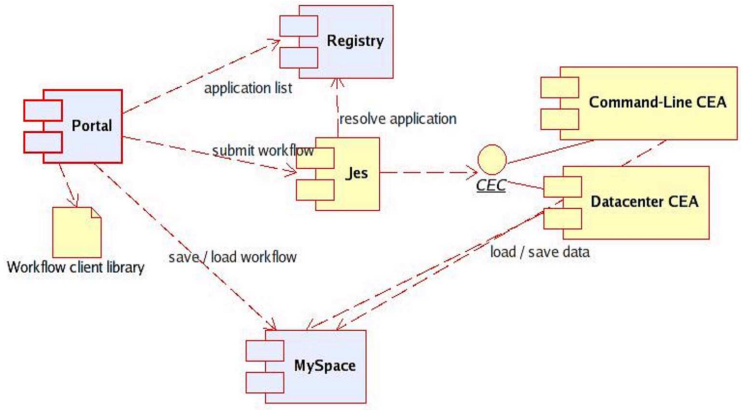

Figura 3.10: Architettura di AstroGrid

JES è una parte di AstroGrid che controlla l'esecuzione del workflow. E' una struttura che può gestire job costituiti da molti step, dove ogni step può essere lanciato su vari computer attraverso la rete. JES sarà in grado di eseguire in parallelo o in sequenziale i job a seconda di come sono descritti nel workflow. Poichè gli step possono essere eseguiti in diverse parti, deve essere gestito sia il flusso di controllo che il flusso di dati. Una componente chiamata *Common Execution Controller* (CEC), gestisce l'esecuzione degli step e il flusso di dati richiesto. Tutti gli step del workflow sono eseguiti in maniera asincrona. Questo significa che uno step è basato su uno schema non bloccante. CEC informa JES quando uno step è completato, questo perchè gli step sono inviati in maniera asincrona e quindi non bisogna aspettare che un step finisca una volta che è stato inviato. Un'altra componente dell'architettura è il *Common Execution Architecture* (CEA) (32)che nasce per creare un piccolo set di interfacce e schemi per determinare come eseguire una tipica applicazione astronomica con il VO. In questo contesto l'applicazione può essere una serie di processi che elaborano o producono dati, includendo applicazioni a linea di comando, query di database e Web Service. CEA è stato in primo luogo disegnato per lavorare con meccanismi di chiamate a web service, sebbene è possibile avere un linguaggio specifico usando la stessa interfaccia; per esempio nel caso di AstroGrid l'implementazione è in Java.

3.3.2 Conclusioni

L'architettura di AstroGrid è studiata esattamente per le applicazioni astronomiche e prevede una serie di problematiche strettamente legate a queste applicazioni. Il linguaggio di workflow è molto ricco di costrutti anche se non

è un linguaggio universale e standard come ad esempio WS-BPEL (Business Process Execution Language for Web services) (60) . La potenza e l'innovazione di questa architettura sono realizzate pensando a tutti i complessi aspetti di un workflow astronomico, risolvendo problematiche come l'accesso ai cataloghi, il recupero e lo storage dei dati in fasi intermedie dell'esecuzione e infine ma non meno importante, la possibilità di esprimere complesse esecuzioni con un linguaggio di workflow completo di costrutto di controllo. Uno dei grandi limiti di AstroGrid è non avere ancora una metodologia di accesso ai servizi Grid; l'architettura non prevede ancora la sottomissione di job ad un middleware esistente (Glite, LCG, UNICORE e altri). Attualmente si cerca di passare da un esecuzione in locale del workflow ad un esecuzione su Cluster, sempre in locale. Si prevede in seguito il passaggio di esecuzione ad un campus Grid, in particolare quello dell'Università di Cambridge CamGrid. Il campus in questione è un Condor flock. Il flocking in Condor agisce come un broker e i job sottomessi a CamGrid possono essere spediti in una qualsiasi parte dell'Università. In ogni caso siamo ancora lontani da una completa integrazione dell'architettura di AstroGrid con gli attuali grid middleware europei.

Capitolo 4

I Web Service

4.1 Introduzione alla Service Oriented Architecture

Dal punto di vista tecnologico, gli ultimi 15 anni sono stati spesi nella realizzazione di middleware standard e di architetture avanzate, imparando dai successi e dai fallimenti dei sistemi distribuiti e dai middleware basati su message oriented architecture. Dal punto di vista dell'architettura, SOA (Service-Oriented Architecture) permette diversi approcci per la realizzazione e lo sviluppo di sistema distribuiti. SOA è un particolare tipo di architettura che permette il disaccoppiamento e il binding dinamico tra i servizi. Le basi principali che costituiscono la SOA sono mostrate in figura 4.1.

In primo luogo, è necessario fornire una definizione astratta dei servizi, includendo i dettagli che permettono, a chi vuole utilizzare il servizio, di comporlo in modo appropriato (*bind*). Inoltre, i provider del servizio devono pubblicare i dettagli dei loro servizi in modo tale che chi li utilizza può capire precisamente cosa fare, ottenendo le informazioni necessarie per utilizzarli (*publish*) (61). Infine, coloro che richiedono i servizi hanno bisogno di avere a disposizione dei metodi per determinare la disponibilità dei servizi a loro necessari(*find*). Per far in modo che l'approccio *bind/publish/find* lavori bene, devono essere definiti gli standard per gestire come e cosa pubblicare, come trovare le informazioni e i dettagli su come comporre i servizi. Il provider dei servizi descrive le sue semantiche, cioè le funzioni che supporta in aggiunta alle sue tipiche operazioni e richiede i dettagli su come accedere al servizio. Le informazioni descrittive sui servizi sono pubblicate in una directory o in un registro. Il richiedente utilizza una facility per il discovery associato con i registri per trovare ciò a cui è interessato, basandosi sulle informazioni pubblicate. Il richiedente può comporre i servizi selezionati. Le

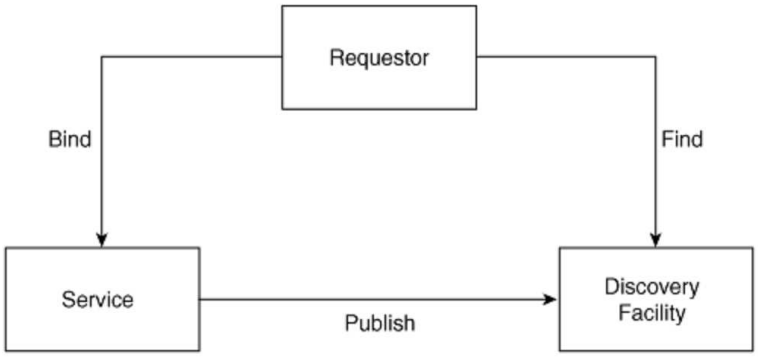

Figura 4.1: Le basi principali di una SOA

informazioni sulle descrizioni dei servizi sono basate sul WSDL (Web Service Description Language); il meccanismo di pubblicazione / identificazione di un servizio tramite la consultazione di registry dedicati è ottenuto tramite UDDI (Universal Description, Discovery and Integration) che è una specifica che definisce come pubblicare e ricercare dei servizi. I vari step che scoprono e compongono i servizi sono i seguenti:

- il richiedente descrive i servizi necessari attraverso delle funzioni
- basandosi su questi input, la facility per il discovery produce una lista di servizi candidati che soddisfano tali necessità
- da questa lista di candidati, il richiedente sceglie un servizio e determina il suo indirizzo e i dettagli per il binding, come il protocollo da utilizzare e quale formato dei messaggi è richiesto. Queste informazioni sono contenute nella lista dei candidati o sono ricevute in un uno step successivo.

La figura 4.2 mostra l'architettura high-level della SOA.

4.2 I Web Service

Con il passaggio da un sistema di gestione del workload ad un sistema per la gestione del worflow non abbiamo più un mapping di tipo task-job, ma un mapping di tipo task-servizio. Le applicazioni astronomiche sono costituite principalmete da sequenze di task ed ognuna delle esecuzioni coinvolge uno o più servizi. Oggi parlare di servizi significa parlare di Web Service e, un errore comune è pensare di combinare insieme Web Service e SOA. Le architetture

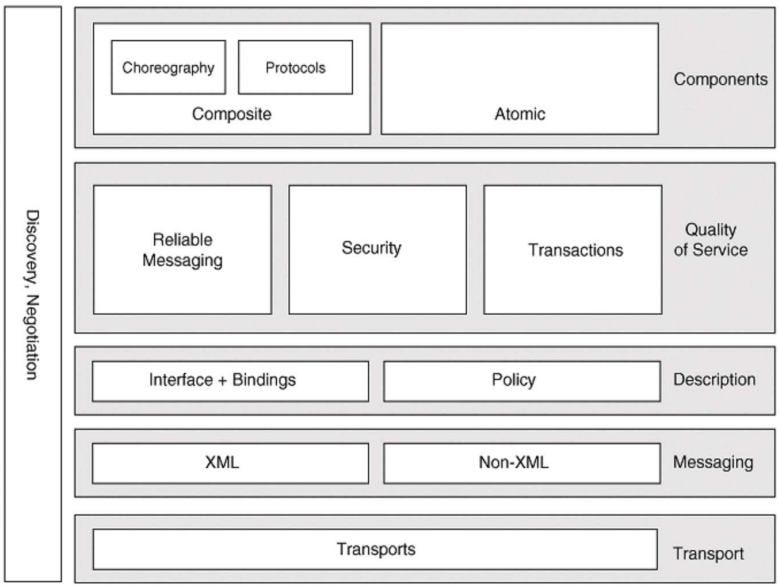

Figura 4.2: L'architettura dei Web Service

Service-Oriented sono un approccio per la costruzione di sistemi software, basato sul disaccoppiamento dei servizi, che sono stati opportunamente descritti in unico modo e che possono essere composti in diversi modi. I Web Service, invece, sono un approccio innovativo per realizzare un'architettura SOA.

I web services sono un servizio disponibile tramite Internet/Intranet che utilizza un sistema standard di messaging basato su XML e mette a disposizione dei metodi per invocare dei servizi. Inoltre, cosa importante, non sono legati ad un particolare sistema operativo o linguaggio di programmazione (9).

I web service godono di alcune proprietà aggiuntive:

- self-describing, per la descrizione delle caratteristiche offerte tramite una grammatica XML
- discoverable, meccanismo di pubblicazione/identificazione di un web service tramite la consultazione di registry dedicati

Con l'utilizzo dei Web Services ci si sposta da un Web centrato sull'utente (*human-centric Web*), dove gli utenti sono gli attori principali per le richieste su Web, a un Web incentrato sulle applicazioni (*application-centric Web*). Questo non significa che gli utenti sono completamente fuori dal contesto, ma che le comunicazioni dirette tra le applicazioni avvengono facilmente come tra

i browser e i server. L'*application-centric Web* non è un concetto nuovo, ma i sistemi sviluppati precedentemente fornivano soluzioni fatte ad hoc. Con i Web Service, invece, abbiamo la standardizzazione che favorisce l'integrazione delle applicazioni. L'architettura Web Service fornisce un'alternativa interessante per il disaccoppiamento.

Ci sono due modi di vedere l'architettura Web Service: il primo è quello di esaminare le singole regole che li costituiscono; il secondo è quello di esaminare i protocolli emergenti (9).

1. Le tre regole che costituiscono un'architettura Web Services sono:
   - service provider, per implementare e rendere disponibile i web service
   - service requestor, per utilizzare i web service tramite delle richieste XML
   - service registry, per contenere le informazioni su come localizzare i web services in base a delle chiavi di ricerca
2. Dall'altro punto di vista possiamo esaminare l'insieme di protocolli che la costituiscono:
   - Service transport, responsabile del trasporto dei messaggi tra le applicazioni; attualmente include HTTP, SMTP, FTP
   - XML messaging, responsabile per la codifica dei messaggi in formato XML; attualmente questo layer include anche XML-RPC e SOAP
   - Service description, responsabile della descrizione delle interface da pubblicare per uno specifico Web Service; attualmente è gestito da Web Service Description Language (WSDL)
   - Service discovery, responsabile della centralizzazione dei servizi in un registro comune e fornisce le funzionalità di publish/find. Attualmente, il servizio di discovery è gestito da Universal Description Discovery and Integration (UDDI)

4.2.1 SOAP: un protocollo per il trasporto

SOAP (Simple Object Access Protocol) (fig.4.3) e' un protocollo XML-based per l'accesso ai metodi di un oggetto remoto. Sebbene può essere utilizzato in diversi sistemi di messaging ed è indipendente dal livello di trasporto,

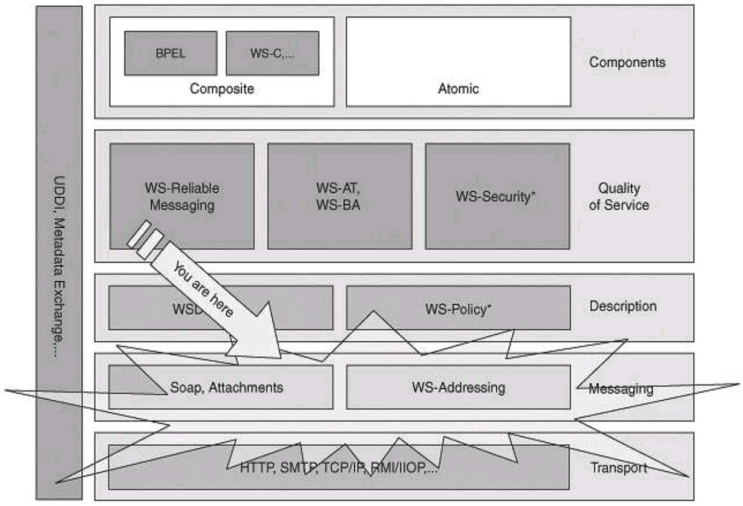

Figura 4.3: Il livello di SOAP nell'architettura dei Web Service

SOAP utilizza, in generale, il protocollo HTTP (61). Un server SOAP è l'implementazione delle regole di elaborazione descritte con le specifiche di SOAP e può trasmettere, ricevere ed elaborare i messaggi SOAP. Sebbene i nodi implementano i modelli SOAP per l'elaborazione, essi possono anche accedere ad altri servizi che devono fornire i protocolli standard come HTTP, SMTP o TCP. La specifica SOAP definisce tre elementi:

- *SOAP envelope specification*, che definisce le regole per l'incapsulamento dei dati da trasferire, del metodo da invocare, dei parametri e/o dei valori di ritorno
- *data encoding rules* che definisce le regole per la codifica delle strutture dati in un messaggio XML (array, hash table, int, double, float, ...)
- *RPC convention*, che definisce la convenzione per implementare una RPC call: messaging one-way, messaging two-way

Envelope. Un messaggio one-way, una richiesta da un client o una risposta da un server sono messaggi SOAP. Un messaggio è l'unita base della comunicazione tra nodi SOAP. Ogni messaggio SOAP ha due elementi obbligatori, *envelope* e *body*, e uno opzionale, *header*. Ognuno di questi elementi ha una serie di regole associate e la comprensione ne facilita l'utilizzo. L'elemento *envelope* può contenere zero o più *header* e un l'elemento *body* che contiene l'informazione . L'innesto di questi elementi è mostrato in figura 4.4.

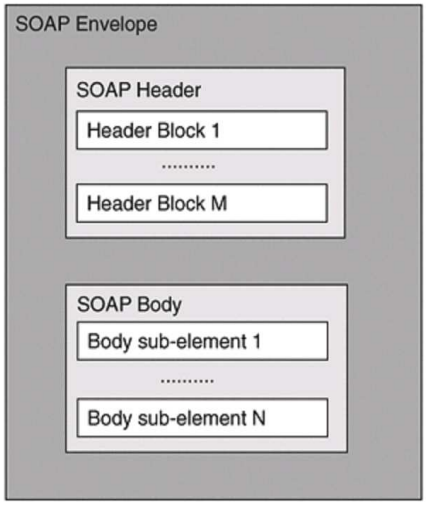

Figura 4.4: Gli elementi che costituiscono un messaggio SOAP

Ogni messaggio SOAP ha un elemento di root envelope. Rispetto ad altre specifiche come HTTP e XML, SOAP non definisce un modello di versione tradizionale basato sul numero di release, ma utilizza un namespace XML per differenziare le versioni. L'elemento envelope specifica la versione utilizzata, ad esempio:

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
```

Header. L'elemento opzionale *header* offre flessibilità per specificare informazioni aggiuntive application-level, come ad esempio la firma digitale, il numero di account e altro.

Body. L'element *body* è obbligatorio per tutti i messaggi SOAP e include richieste e risposte RPC. Esso contiene le informazioni relative al metodo da invocare.

Fault. Nel caso in cui si verifica un errore, l'elemento *body* conterrà un elemento *fault* che contiene a sua volta dei sottoelementi come *faulCode*, *faultString* e altri. Esso contiene le informazioni inviate dal provider al requestor in caso di errore, ad esempio se un metodo non esiste o se l'accesso è negato

4.2.2 WSDL: un linguaggio per la descrizione

WSDL è una specifica che definisce come descrivere i Web Service tramite una grammatica XML. Mentre SOAP è il formato per scrivere i messaggi

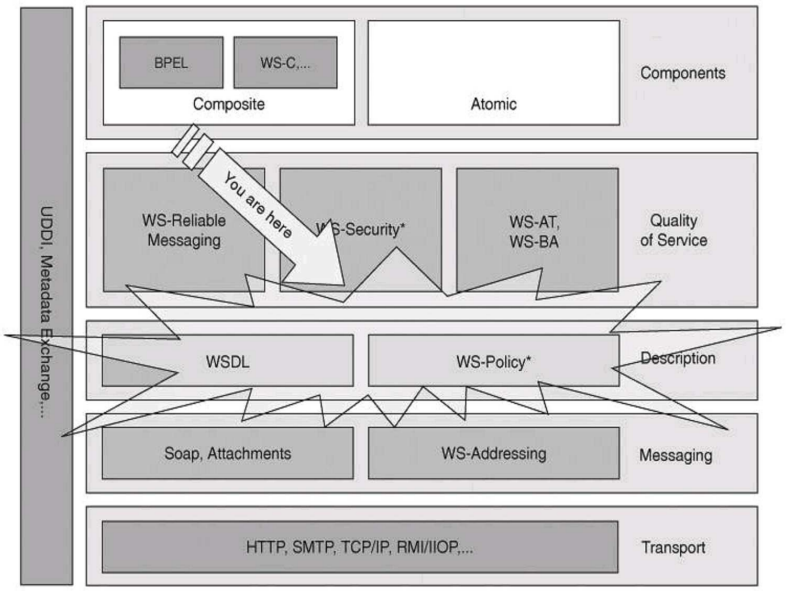

Figura 4.5: Il livello di WSDL nell'architettura dei Web Service

nelle interazioni tra Web Service, WSDL è il linguaggio per sapere quali metodi sono esposti. Attraverso questo linguaggio gli autori dei servizi danno informazioni sul servizio in modo tale che gli altri possano utilizzarlo. WSDL è stato progettato in modo tale da adattarsi a fornire la descrizione dei servizi utilizzando diversi type system, come XML Schema o Java. E' una piattaforma indipendente dal linguaggio e descrive quattro fondamentali parti relative ai dati:

- L'interfaccia per descrivere tutte le funzioni disponibili
- Le informazioni sui data type per tutti i messaggi di richiesta e di risposta.
- Il binding information per i protocolli di trasporto utilizzati
- Le informazioni sugli indirizzi per localizzare i servizi specificati

Un documento WSDL è costituito da due parti: un abstract e una parte concreta. L'abstract descrive le caratteristiche dei Web Service relative alle operazioni includendo i messaggi in uscita e in entrata dai servizi; la parte concreta invece descrive come e dove accedere ai servizi implementati.

La specifica WSDL definisce sei elementi:

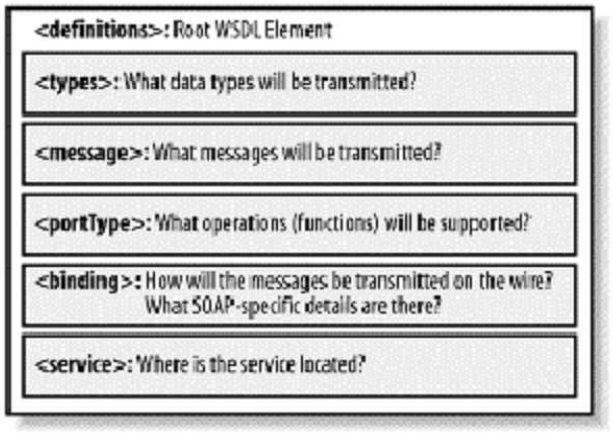

Figura 4.6: Le specifiche WSDL

- *definitions* è il root element del documento WSDL; specifica il nome del web service e i *namespace* utilizzati nel documento
- *types* descrive i tipi di dati utilizzati tramite la specifica XML-schema. WSDL non è legato ad un particolare sistema di tipi ma usa le specifiche del W3C XML Schema come scelta di default.
- *messages* descrive ciascun messaggio di tipo one-way, come singolo messaggio di richiesta o come singolo messaggio di risposta
- *portType* descrive le operazioni supportate dal web service combinando piu' one-way message
- *binding* descrive la modalita' di trasmissione dei messaggi (SOAP, ...)
- *service* descrive l'indirizzo da utilizzare per invocare il web service. Di solito questi includono un URL per l'invocazione a SOAP

La figura 4.6 chiarisce il significato di ogni elemento con una semplice rappresentazione delle specifiche WSDL. Oltre ai sei principali elementi, WSDL ha altre due specifiche: *documentation*, utilizzato per fornire la documentazione leggibile per l'utente, che può essere inclusa nel documento WSDL; *import* è utilizzato per poter importare altri documenti WSDL o XML Schema. Questo rende il documento WSDL più modulare. Ad esempio due documenti WSDL possono importare lo stesso elemento base e includere anche i propri elementi per rendere lo stesso servizio disponibile su due indirizzi fisici.

Definitions

L'elemento *definitions* specifica il nome del Web Service e i *namespaces* che saranno utilizzati nel documento. L'uso dei namespace è importante per differenziare gli elementi e permette ai documenti di fare riferimento a diverse specifiche esterne, includendo le specifiche WSDL, le specifiche SOAP, e le specifiche XML Schema. L'elemento *definitions* specifica anche l'attributo targetNamespace che è una convenzione di XML Schema che permette a un documento WSDL di fare riferimento a se stesso. Notiamo comunque che il *namespace* non richiede che i documenti esistano veramente in quella locazione, ma permette di specificare un valore che e' unico, diverso da tutti gli altri *namespace* che sono definiti. Poichè l'elemento *definitions* può specificare un *namespace* di default, gli elementi senza un *namespaces* fissato, come ad esempio *message* o *portType*, possono fare riferimento a quello di default.

Message

Messages descrive ciascun messaggio di tipo one-way, come singolo messaggio di richiesta o come singolo messaggio di risposta. Ogni messaggio contiene un elemento chiamato *part*. Per la richiesta, l'elemento *part* specifica i parametri della funzione; per la risposta, specifica il valore di ritorno della funzione. L'elemento *part* ha l'attributo *types*, che specifica l'XML Schema data type. Il valore del *types* deve essere specificato come un XML Schema Qname , questo significa che il valore dell'attributo deve essere un *namespace* qualificato. Ad esempio il *firstName types* è definito come *xsd:string*; il prefisso *xsd* fa riferimento al *namespace* di XML Schema precedentemente definito con l'elemento *definitions*. Se la funzione ammette diversi argomenti come parametri o molti valori di ritorno, si possono specificare più elementi *part*.

PortType

PortType descrive le operazioni (funzioni) supportate dal web service combinando più *one-way message*. Cosi come per l'elemento type definito precedentemente, l'attributo *message* deve essere specificato come un XML Schema Qname. Questo significa che il valore dell'attributo deve essere un *namespace* qualificato. WSDL supporta quattro *operation patterns*:

- *One-way*, in cui il servizio riceve un messaggio; l'operazione quindi ha un singolo *input*.
- *Request-response* in cui il servizio riceve un messaggio e spedisce la risposta; l'operazione quindi ha un *input* seguito da un *output*. Per incapsulare gli errori può essere specificato l'elemento opzionale *fault*

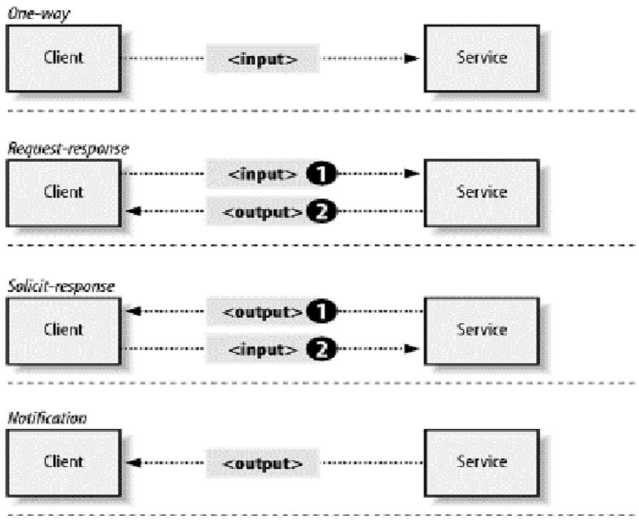

Figura 4.7: Le operation patterns supportate da WSDL

- *Solicit-response*, in cui il servizio spedice un messaggio e riceve una risposta. L'operazione quindi ha un *output* seguito da un *input*. Per incapsulare gli errori può essere specificato l'elemento opzionale *fault*
- *Notification* in cui il servizio spedisce un messaggio. L'operazione quindi ha un *output*.

La figura 4.7 mostra i vari *operation pattern*;

Binding

L'elemento *binding* fornisce i dettagli su come l'operazione *portType* sarà offerta sulla rete. *Binding* può essere disponibile tramite vari servizi di trasporto come HTTP GET, HTTP POST, o SOAP. Infatti è possibile specificare molti *binding* per un singolo *portType*. L'elemento *binding* specifica gli attributi *name* e *type*.

4.3 SOA e i Web Services

I Web Service espongono metodi atraverso l'utilizzo dei servizi e virtualizzano le risorse, in particolare in ambienti etereogenei e distribuiti. Le interazioni tra i richiedenti e i servizi devono assicurare Quality of Service (QoS), come l'affidabilità dei servizi per il trasporto dei messaggi, la protezione delle transazioni, la sicurezza a livello di messaggi, e tante altre garanzie. Le QoS

non devono essere assicurate solo tra due partecipanti ma tra un discreto numero di partecipanti in un ambiente eterogeneo. Da questo punto di vista è giustificata l'applicabilità di SOA ,in generale, e dei Web Service in particolare. SOA non risulta una soluzione efficace nel caso in cui le applicazioni da gestire sono piccole o i cui requirement non ottengono abbastanza benefici dall'uso di SOA. Inoltre SOA non da alcun beneficio in un tipo di applicazione statico. Se invece il problema da risolvere è focalizzato sull'integrazione di componenti in ambienti eterogenei o su configurazioni che cambiano in maniera dinamica, allora ha senso utilizzare SOA e la tecnologia Web Service. SOA offre dei notevoli benefici anche nelle applicazioni distribuite, che sono sottoposte a molti cambiamenti, come ad esempio diversi merge e acquisizioni, o frequenti cambi di servizi. Se il riutilizzo di una funzione, nel senso di renderla disponibile a tutti i tipi di richiedenti, è importante, allora un buon approccio è quello di fornirla tramite Web Service (61).

Capitolo 5

Grid Computing e i Web Services

5.1 Introduzione ai servizi Grid Computing

Le Griglie computazionali non nascono con lo scopo di sostituire i vari servizi e sistemi di calcolo presenti da anni nel contesto internet ma per essere un loro completamento (33). Per questo motivo nel progetto di un infrastruttura Grid bisogna utilizzare protocolli e interfacce standard e open, essenziali per assicurare le varie funzionalità di base e assicurare l'interoperabilità tra le diverse Virtual Organization. L'accesso alle risorse e ai servizi offerti deve essere totalmente trasparente, quindi occorre un meccanismo che nasconda all'utente finale la distribuzione delle risorse e le differenze tra i vari ambienti di calcolo, dandogli l'impressione di utilizzare un'unica macchina dedicata alle sue esigenze. Lo strato di *software* necessario a questo ed altri scopi è chiamato *middleware*.

In ambito *Grid* la maggior parte dei middleware sono implementati su una base di *Globus Toolkit* (GT). *Globus Toolkit* raccoglie servizi e librerie che riguardano il *resource monitoring, il discovery, il management*, la sicurezza e la gestione dei *file*; esso rappresenta il nucleo dell'infrastruttura *Grid* e può anche essere utilizzato per creare applicazioni *Grid enabled* (25). La politica *open source* con cui è portato avanti il progetto ne facilita l'utilizzo e contemporaneamente contribuisce in maniera significativa alla continua crescita del prodotto. La sua rappresentazione è il modello a tre infrastrutture, ognuna delle quali è una componente primaria di Globus e condividono un'unica infrastruttura di sicurezza denominata *Grid Security Infrastructure* (GSI). L'infrastruttura *Resource Management* realizza la gestione e l'allocazione delle risorse (GRAM), l'*Information Services* implementa le funzionalità di

monitoraggio e *discovery* delle risorse (MDS), mentre il *Data Management* realizza i servizi ed i protocolli di gestione dati come *GridFTP* e *Globus Replica Catalogue*. I protocolli base di comunicazione si basano su protocolli classici come IP, TCP, UDP, DNS, RSVP

Il *Globus Toolkit* utilizza i protocolli GSI (Grid Security Infrastructure) per l'autenticazione, la protezione della comunicazione e l'autorizzazione. GSI si basa sulla crittografia a chiave pubblica, certificati X.509 e sul protocollo di comunicazione SSL (Secure Socket Layer).

Vengono definiti anche i protocolli per negoziare, iniziare, monitorare, controllare e addebitare l'utilizzo di risorse non distribuite; essi si dividono in *information protocol*, utilizzati per ottenere informazioni sulla configurazione e lo stato di una risorsa e *management protocol* per negoziarne l'accesso.

Globus Toolkit utilizza i seguenti protocolli:

- LDAP (Lightweight Directory Access Protocol): utilizzato per l'accesso a *directory*
- GRIP (Grid Resource Information Protocol): si basa su LDAP ed è utilizzato per recuperare informazioni sulle risorse e per definire un modello informativo
- GRRP (Grid Resource Registration Protocol): utilizzato per registrare le risorse
- GRAM (Grid Resource Access Management): si basa su HTTP ed è utilizzato per l'allocazione ed il controllo delle risorse di calcolo; utilizza il *Resource Specification Language* (RSL) per la specifica delle richieste
- GridFTP (Grid File Transfer Protocol): si basa su FTP estendendo le sue funzionalità; utilizza i protocolli di sicurezza dello strato *Connectivity* e gestisce una sorta di parallelismo per i trasferimenti ad alta velocità, inoltre è sicuro, efficiente, affidabile e flessibile

Vengono compresi anche servizi come:

- GRIS (Grid Resource Information Service): fornisce lo stato di una particolare risorsa Grid.
- GIIS (Grid Index Information Service): è un servizio che incorpora un insieme di server GRIS per fornire informazioni su un'intera organizzazione.
- GIS (Grid Information Service): riceve ad intervalli di tempo notifiche dalle risorse Grid e fornisce meccanismi di base per scoprirne l'esistenza e le caratteristiche.

Il *Globus Toolkit* utilizza anche il *monitoring and directory service* (MDS) per la gestione informativa delle risorse. Questo servizio offre gli strumenti necessari per costruire un'infrastruttura informativa basata su LDAP e permette di ottenere informazioni sulle varie risorse Grid. MDS si basa sui servizi GRIS, GIIS e sull'*Information providers*, che rappresenta un servizio locale utile ad ogni risorsa per pubblicare informazioni su se stessa. In ambiente Grid i principali tipi di risorse sono quelle di calcolo e quelle di storage; le prime sono dette *Computing Element* (CE), mentre le seconde *Storage Element* (SE). Servizi utili per la gestione delle SE sono il *Replica Catalogue*, che permette di conoscere quante copie esistono di un particolare file, ed il *Replica Location Service* in grado di individuare i dischi che le contengono. Un utente interagisce con una griglia computazionale attraverso il *Workload Management System* (WMS); questo servizio si occupa dello *scheduling* distribuito dei job, offre funzionalità per la loro sottomissione, esecuzione e *monitoring*; inoltre fornisce l' output dell'elaborazione all'utente e provvede ad ottimizzare l'utilizzo delle risorse (5). Attualmente il WMS si compone delle seguenti parti:

- User Interface (UI): è il punto di accesso alla griglia e fornisce un'interfaccia con la quale gli utenti possono colloquiare con il sistema.
- Resource Broker (RB): è il mediatore delle risorse di Grid ed è responsabile di trovare le migliori risorse utili ai job. Effettua l'operazione di *matching* tra job requirements e risorse disponibili
- Job Submission Service (JSS): consegna i job ai Computing Element scelti dal RB.
- Information Index (II): è un server LDAP utilizzato dal Resource Broker per selezionare le risorse che soddisfano i requisiti di un job; questo servizio rappresenta un filtro per il Grid Information Service.
- Logging and Bookkeeping service (LB): fornisce le informazioni sui job se richieste dagli utenti.

5.2 Dai Web service ai Grid service

Il Globus Toolkit (GT) è un progetto open source, che permette di condividere risorse di calcolo, dati, database ed applicazioni tra istituzioni, in modo sicuro, preservando l'autonomia organizzativa, gestionale ed amministrativa di ogni singola organizzazione. L'architettura del GT è modulare e indirizza alcune delle problematiche presenti in un ambiente di Grid Computing

Attualmente, il team del Globus project sta sviluppando la terza versione del GT (GT3), che implementa l'Open Grid Service Architecture, basata sul concetto di GridService, derivante da quello di Web Service.

Prima di entrare nel dettaglio di quello che è un Grid service, occorre descrivere meglio due concetti: OGSA e OGSI.

OGSA

Open Grid Services Architecture (OGSA) è un insieme di specifiche e di standard che combinano i vantaggi del Grid Computing e dei Web Services. Grazie ad OGSA, gli utenti hanno ora la possibilità di accedere e condividere on demand le risorse di elaborazione su Internet, basandosi su un'infrastruttura flessibile, che può essere gestita in modo autonomo (52). OGSA definisce meccanismi standard per la creazione, il naming ed il discovering delle istanze transienti di un servizio; fornisce la possibilità di accedere le risorse senza conoscere dove esse siano allocate ed i protocolli binding multipli per le istanze di un servizio; supporta l'integrazione con le funzionalità della piattaforma sottostante e supporta la creazione, la gestione e l'utilizzo di insiemi di servizi gestiti dalle organizzazioni virtuali. OGSA :

- definisce le caratteristiche fondamentali che una Grid dovrebbe avere
- definisce il modello di programmazione di una Grid
- fornisce informazioni su come costruire un servizio di Grid
- precisa quali sono le componenti essenziali per costruire e distribuire un valido prodotto di Grid

OGSI

Open Grid Service Infrastructure (OGSI), utilizzando le tecnologie che stanno alla base dei Grid e Web Services, definisce i meccanismi per la creazione, la gestione e lo scambio di informazioni tra i Grid Service stessi (57). OGSI è una specifica tecnica e formale dei concetti descritti in OGSA e di un Grid Service. Le specifiche di OGSI estendono gli schemi WSDL e XML al fine di gestire le seguenti possibilità:

- WS con stato
- estensione delle interfacce dei WS
- notifica asincrona dei cambiamenti di stato; o riferimenti ad istanze di servizi

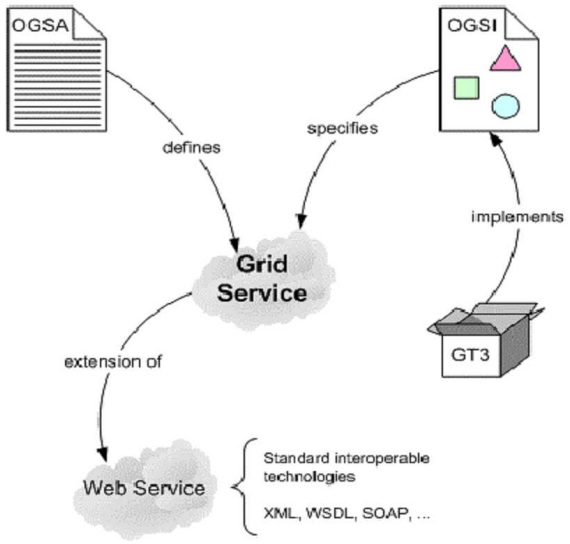

Figura 5.1: Relazione tra OGSA, OGSI e GT3

- collezione di istanze

La figura 5.1 seguente mostra le relazioni esistenti tra OGSA, OGSI e GT3.

Grid Service

Per virtualizzare e comporre dei servizi sono necessarie delle definizioni di interfacce standard e delle semantiche standard per le interazioni di un servizio in modo che altri servizi seguano le stesse convenzioni. Per questo scopo, OGSA definisce il Grid Service, cioè un Web service che fornisce un insieme di interfacce ben definite e che segue specifiche convenzioni. Le interfacce sono rivolte al discovery, alla creazione dinamica, al lifetime management, alla notifica ed alla gestione di un servizio; le convenzioni, invece, sono rivolte al naming ed all'espandibilità di un servizio. Le interfacce e le convenzioni che definiscono un Grid Service sono rivolte, in particolare, ad aspetti riguardanti la gestione di istanze di servizi. La necessità di utilizzare in Grid le istanze di servizio deriva dal fatto che i Web Services permettono di scoprire ed invocare servizi persistenti e questo può creare un problema di condivisione degli accessi, in quanto è possibile invocare contemporaneamente lo stesso metodo dello stesso servizio. Inoltre, i Web Services sono stateless, cioè non è possibile mantenere informazioni tra un'invocazione e l'altra del servizio, mentre in Grid ad ogni istanza di servizio è associato uno stato logico persistente (5.2).

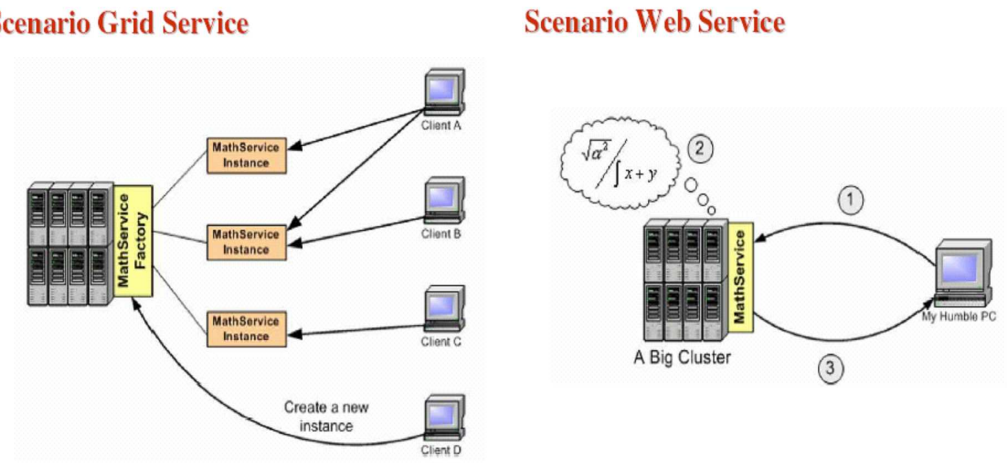

Figura 5.2: Scenario Web Service e Grid Service

WSRF

OGSI non è stato adottato in molti progetti a causa dei problemi delle sue specifiche, infatti:

- non e' una soluzione integrabile con gli attuali Web Service Framework
- l'architettura object-oriented è molto diversa dall'architettura dei Web Service

Nel 2004 le specifiche di OGSI sono state revisionate in una serie di specifiche chiamate WS-Resource Framework (WSRF). WSRF distingue in maniera esplicita i servizi dalle risorse *stateful*.

Dal punto di vista tecnico il più grande cambiamento è dovuto al fatto che il Grid Service Handle (GSH) e il Grid Service References (GSR) sono sostituiti da WS -Addressing (WSA) Endpoint References (EPRs). EPR sono molto simili ai servizi di OGSI ma possono contenere proprietà XML. Il messaggio viene depositato in una SOAP Envelope, etichettato con le label di WS-Addressing Endpoint Reference, che indicano la destinazione del messaggio attraverso servizi intermediari di Grid. Date le dipendenze di OGSI e WSRF con le specifiche dei Web Service create ad hoc per il business, è naturale aspettarsi l'utilizzo dei business workflow, ed in particolare del Business Workflow Language for Web Service, in Grid.

5.3 Il progetto EGEE e lo sviluppo di gLITE

La gestione delle risorse e lo scheduling per la distribuzione del calcolo in un ambiente Grid sono problemi ancora aperti e impegnativi. Sebbene siano stati raggiunti ottimi risultati negli anni precedenti, lo sviluppo e la messa in piedi di componenti affidabili e standard sono obiettivi che ancora devono essere raggiunti del tutto. I domini di interesse sono in particolare:

- Workload Management
- Discovery delle risorse
- Matchmaking e il Brokering delle risorse
- problemi di accounting e politica delle autorizzazioni

Tuttavia una delle attività che ha una particolare attenzione è l'evoluzione delle architetture service-oriented, supportata dall'aumento degli standard. Tali evoluzioni sono state affrontate nel progetto EU-funded EGEE (Enabling Grids for E-science in Europe), che nasce principalmente con lo scopo di fornire componenti per un middleware robusto e per creare un'infrastruttura Grid affidabile per il supporto della applicazioni scientifiche. Il progetto ha l'obiettivo di costruire tecnologie Grid avanzate e di mettere a disposizione un set di Grid Service (3). Il Workload Management è uno degli elementi principali tra i Grid Services. Negli anni precedenti sono stati ottenuti risultati significativi relativamente ai problemi di scheduling e di gestione di grandi quantità di job su una Grid. Il Workload Management System implementato nel progetto DATAGRID, Condor System (6) e EURO GRID Unicore sono esempi di soluzioni per il calcolo su Grid e su risorse eterogenee. Ma il problema della gestione delle risorse e dello scheduling in Grid non è da considerarsi completamente risolto e richiede ancora studio e attenzione.

Prendendo in considerazione le precedenti esperienze dei progetti Grid, i requirement delle applicazioni e l'attuale avanzamento verso la standardizzazione delle specifiche, è stata progettata e implementata un'architettura per il Workload Management System di EGEE.

Il Workload Management System (WMS) comprende una serie di componenti del middleware di Grid responsabili della distribuzione e della gestione dei task sulle risorse di Grid, in modo tale che le applicazioni vengano eseguite in maniera efficiente e affidabile.

Funzionalità

Il nucleo delle componenti del WMS è il Workload Manager che ha il compito di accettare e soddisfare le richieste per la gestione dei job, espresse attraverso

il Job Description Language (JDL). Un'altra componente fondamentale è il Job Logging and Bookkeeping Service. Durante l'esecuzione di un job ci sono due tipi di richieste: la sottomissione e l'eliminazione.

Il significato della richiesta di sottomissione è di passare la responsabilità del job al WM, che lo passerà a sua volta ad un appropriato Computing Element per l'esecuzione prendendo in considerazione i requirement e le preferenze espresse nella descrizione del job. La scelta di quali risorse utilizzare deriva dal processo di matchmaking che avviene tra la richiesta di sottomissione e la disponibilità delle risorse. La disponibilità delle risorse per un particolare task non dipende solo dallo stato delle risorse, ma anche dalla politica di utilizzo scelta dagli amministratori delle risorse o delle Virtual Organization.

Scheduling

Il Workload Management può adottare diverse politiche di scheduling: un job può essere direzionato su una risorsa il più rapidamente possibile, e una volta deciso, il job viene passato alla risorsa selezionata per l'esecuzione, dove probabilmente sarà messo in coda; in alternativa, un job viene tenuto nel Workload Management fin quando una risorsa non diventa disponibile; in tal caso la risorsa viene confrontata con il job sottomesso e il job che si adatta meglio viene passato a tale risorsa, per l'esecuzione.

A livello di matchmaking la differenza fondamentale tra i due tipi di scheduling è che il primo implica un confronto del job con molte risorse, mentre nel secondo caso il confronto è tra una risorsa e molti job.

Information Supermarket

Il meccanismo che permette l'applicazione flessibile di differenti politiche è il disaccoppiamento tra la collezione di informazioni relative alle risorse e il loro uso. La componente che implementa questo meccanismo è l'Information Supermarket (ISM) e rappresenta uno dei miglioramenti del Workload Management.

ISM è un deposito di informazioni sulle risorse, disponibili in sola lettura per il motore di matchmaking, il cui contenuto si aggiorna con l'inoltro da parte delle risorse di notifiche relative al proprio stato.

Task Queue

Il secondo importante miglioramento nel Workload Management è la possibilità di accettare una richiesta di sottomissione mentre le risorse non sono ancora disponibili. La richiesta viene recuperata periodicamente o appena la

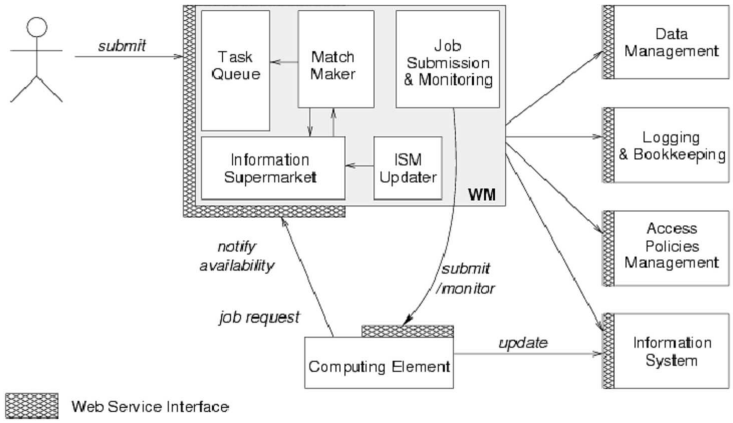

Figura 5.3: L'architettura del Workload Manager

notifica di risorsa disponibile appare nel ISM. La componente che implementa questo aspetto è chiamato Task Queue (TQ).

La fig 5.3 mostra l'architettura del Workload Manager e le interazioni con gli altri servizi. Sia il Workload Management che gli altri servizi in futuro esporranno un'interfaccia Web Service.

Computing Element

Il Computing Element è il servizio che rappresenta una risorsa di calcolo; la sua funzione principale è la gestione dei job. Il CE può lavorare sia in *push model* dove il job è inviato a un CE per essere eseguito, che in *pull model*, dove è il CE a chiedere al WMS dei job da eseguire. Quando un job viene inviato a un CE, esso viene accettato solo se ci sono risorse che soddisfano i requirement specificati dall'utente. Nel *pull model*, invece, quando un CE è pronto a ricevere un job, fa richiesta al WMS. Nella richiesta devono essere incluse le caratteristiche delle risorse, in modo tale che il WMS possa selezionare il job più adatto.

Esistono diverse tecniche di scheduling per il pull model, che possono essere analizzate per determinare quali di esse forniscono le migliori performance in situazioni differenti. Le possibili tecniche sono:

- Il CE richiede un job da tutti i WMS conosciuti. Se due o più WMS offrono dei job, solo il primo che arriva è accettato dal CE, mentre gli altri vengono rifiutati.

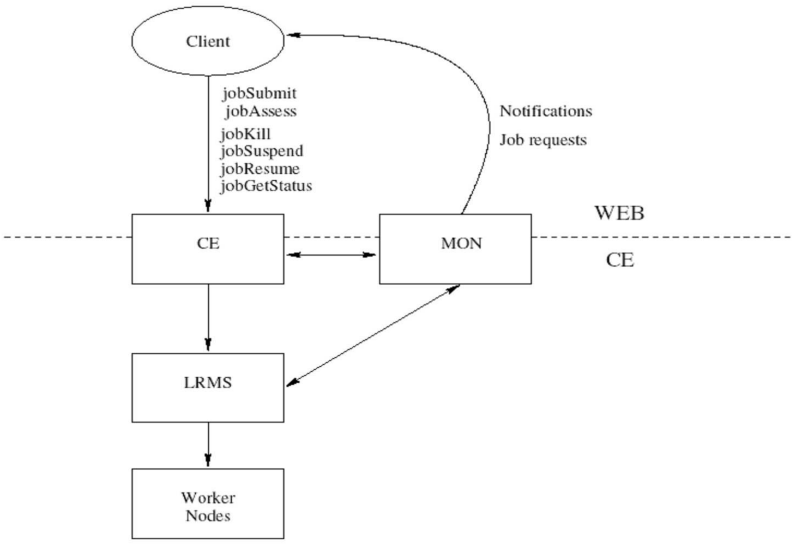

Figura 5.4: L'architettura del Computing Element

- Il CE richiede un job da un solo WMS. Se il WMS contattato non ha job disponibili viene notificato un altro WMS.

Inoltre, il CE espone un'interfaccia Web Service e può essere usato da un generico client. L'architettura di un CE è rappresentata in fig.5.4. Il servizio di Monitor (CEMON) notifica gli utenti asincronicamente sullo stato del job e notifica le caratteristiche e lo stato del CE. In particolare se il CE sta lavorando in *pull model*, il servizio è utilizzato per richiedere i job al WMS.

Capitolo 6

Grid Service Orchestration e BPEL4WS

6.1 Web Service e BPEL4WS: standard per l'interoperabilità

Il workflow è l'automazione di una parte o di un intero process flow, dove i documenti, le informazioni e i task sono trasferiti da un partner all'altro per essere elaborati, in accordo con una serie di regole procedurali. (41)

I workflow scientifici sono uno degli elementi più importanti per l'esecuzione delle applicazioni. Sebbene siano stati sviluppati vari linguaggi per descrivere i workflow, sino ad ora è sempre mancato uno standard completo per orchestrare i servizi. L'utilizzo di uno standard industriale per l'*orchestrazione* dei business flow e dei Web service non permette soltanto di velocizzare l'implementazione e il deployment dei nuovi progetti, ma riduce anche il costo complessivo di gestione, modifica, estensione e reimplementazione dei servizi esistenti. Il termine Web service si riferisce a una serie di standard di interoperabilità (WSDL, XML, XML Schema, SOAP, JMS) che semplificano l'integrazione con sistemi eterogenei. In modo analogo alla rivoluzione portata da standard come SQL nel settore dei dati strutturati e HTTP/ HTML nelle modalità di accesso ai contenuti e alle applicazioni, i Web service hanno il potenziale di trasformare Internet in una vera piattaforma di computing distribuito e di consentire a sistemi eterogenei di cooperare in modo semplice e affidabile.

Attualmente i Web Service sono ancora isolati e poco visibili. Per superare questo problema si possono coordinare i Web Service descrivendo come utilizzare le loro composizioni per eseguire workflow. La composizione (fig.6.1) dei Web Service è un paradigma che consente di eseguire un work-

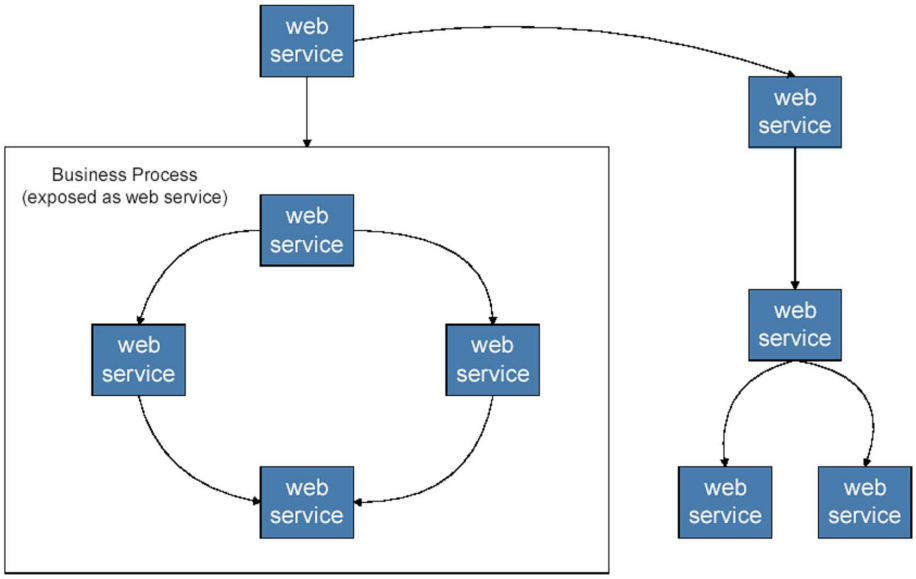

Figura 6.1: Composizione di Web Services

flow specificando l'ordine di esecuzione in una collezione di Web Service, la distribuzione dei dati nella composizione di Web Service, i partner coinvolti e le relazioni di dipendenza tra i partner. Possiamo distinguere due tipi di composizione:

- *Service Orchestration.* E' l'interazione tra web service a livello di messaggi, considerando la logica dell'elaborazione e l'ordine dell'esecuzione; il controllo dei workflow è gestito da una delle parti coinvolte (17). L'orchestration è supportata da Business Process Execution Language for Web Service (BPEL4WS).

- *Service Choreography.* Il controllo dei workflow non è gestito da un singolo partner ma ognuno dei partner descrive il proprio ruolo nell'interazione.

BPEL4WS offre alla comunità scientifica uno standard per l'orchestration e l'esecuzione di workflow. Dal punto di vista delle caratteristiche, BPEL4WS offre un linguaggio standard per definire le modalità per: inviare messaggi XML a servizi remoti, manipolare strutture di dati XML, ricevere messaggi XML in modo asincrono da servizi remoti, gestire eventi ed eccezioni, definire sequenze parallele di esecuzione e l'annullamento delle parti dei processi nelle quali si verificano delle eccezioni. Questi sono i costrutti necessari per combinare una serie di servizi per l'esecuzione di process flow.

BPEL4WS si basa su XML Schema, SOAP e WSDL (48). A differenza degli standard di processo proposti nel passato, BPEL4WS, sostenuto dall'organismo normativo OASIS, ha ottenuto il supporto delle industrie. BPEL4WS è uno standard completo che soddisfa i requisiti e le esigenze delle applicazioni scientifiche a differenza dei precedenti linguaggi che non sono riusciti a sviluppare un unico standard assoluto.

Negli ultimi anni le industrie sono sempre più interessate all'utilizzo delle architetture di tipo Web Services per il calcolo distribuito. In particolare stanno acquisendo metodi e strumenti per lo sviluppo, il supporto e l'integrazione dei Web Services nelle tecnologie del Grid Computing. Infatti attualmente si parla di Grid Services per indicare un'infrastruttura Grid Computing orientata ai servizi.

6.2 Definizioni e caratterizzazioni

BPEL4WS è un linguaggio basato su XML che consente di descrivere formalmente un workflow per coordinare i Web Service che interagiscono nel flusso di elaborazione. La modalità di interazione è ottenuta descrivendo l'interfaccia di composizione come una collezione di WSDL portType. La composizione è ricorsiva, in modo tale che i partner espongono l'interfaccia WSDL a coloro che interagiscono con essi.

BPEL4WS è stato sviluppato per lavorare in un ambiente altamente dinamico nel quale i servizi possono cambiare continuamente ed è intenzionalmente disaccoppiato da particolari istanze. Esso si posiziona al livello più alto dell'architettura dei Web Services (fig.6.2), utilizzando direttamente WSDL per definire le funzionalità che espone e che richiede ai servizi interagenti (61). Per comprendere meglio l'utilizzo di BPEL4WS mostriamo il seguente esempio:

> *Un cliente vuole acquistare un pacchetto di servizi per un set di computer. I pacchetti richiesti corrispondono ad una serie di servizi di supporto, come ad esempio, riparazioni a domicilio e garanzie hardware. Prima che il cliente possa procedere con l'acquisto, è necessario validare l'ordine e questo consiste nel controllo di alcuni aspetti, come ad esempio, la disponibilità dei servizi richiesti nel contesto del cliente (posizione geografica, data dell'ordine e altro), la compatibilità dei servizi con le macchine possedute, e con altri servizi richiesti.*

Descrivere un workflow con BPEL4WS risolve i problemi relativi ad una transazione di questo tipo. BPEL4WS può accettare la richiesta di validare

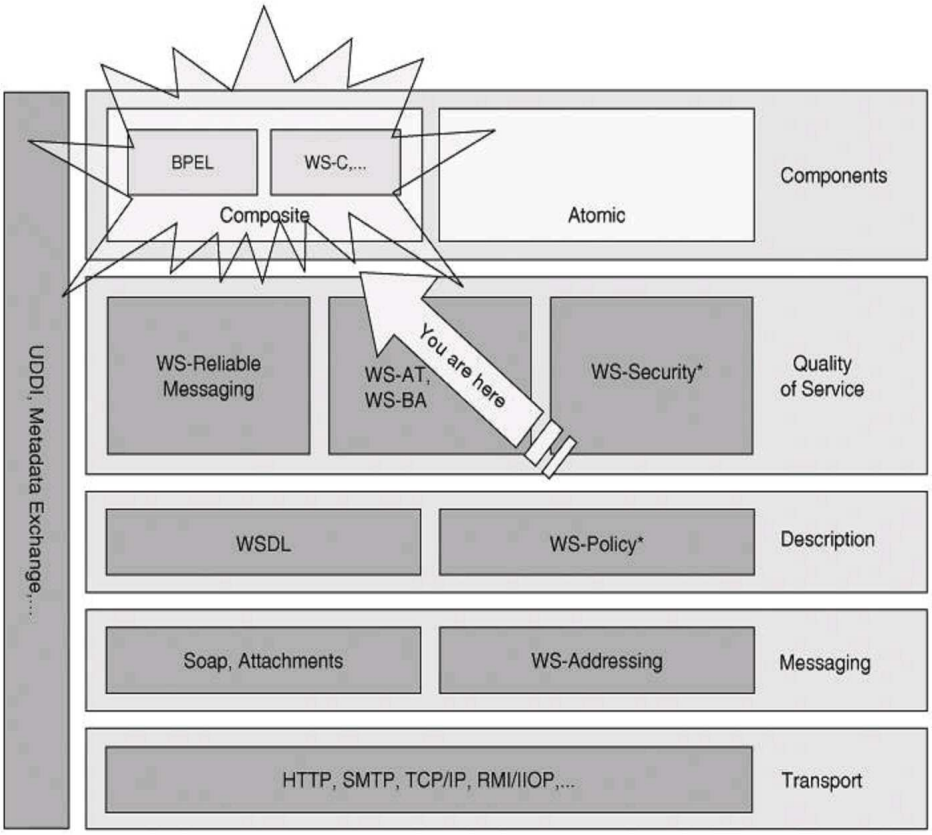

Figura 6.2: Il ruolo di Bpel nell'architettura dei Web Services

l'ordine d'acquisto con una chiamata ai Web Services esposti. La richiesta viene scomposta in tante sottorichieste ognuna delle quali è spedita ad un diverso Web Service che a sua volta ne può validare un particolare aspetto. In seguito si creano le risposte alla richieste di validazione, basate sulle risposte che provengono dai vari servizi che sono stati chiamati. Le risposte contengono i risultati della validazione per tutti i pacchetti dei servizi richiesti e viene inoltrata al chiamante. Solo in questa fase, BPEL4WS può accettare l'ordine d'acquisto per comprare i servizi. Infine il workflow descrive anche l'invocazione al servizio che si occupa di fatturare i clienti e di aggiornarne il profilo (61).

Quello che abbiamo voluto sottolineare con questo esempio è che BPEL4WS riesce a realizzare l'orchestration dei servizi, supportando complessi workflow ed esponendo ai Web Service tutte le activities di un workflow. BPEL4WS può mantenere strutture dati temporanee durante le esecuzioni. Questi dati possono essere richiesti e manipolati dal workflow di BPEL4WS per passarli

da un servizio ad un altro. Tra i numerosi linguaggi che sono stati pensati e creati per la composizione di oggetti, nessuno è stato in grado di integrarsi e operare completamente in SOA. Quello di cui si necessita in un ambiente Service Oriented è un linguaggio capace di disaccoppiare le entità in gioco, che gestisca le interazioni on demand, e la variazione continua di alcuni parametri come la posizione, la disponibilità e la qualità del servizio. Ciò che si richiede e si realizza con BPEL4WS si può riassumere cosi:

- integrazione flessibile
- composizioni ricorsive
- divisione e composizione dei workflow
- gestione del ciclo di vita e delle comunicazioni
- ricuperabilità

In particolare BPEL4WS può essere utilizzato per l'orchestration dei servizi nell'ambito dei Grid Service.

6.3 L'architettura di Bpel

BPEL4WS ha un sistema di type che assicura che i dati scambiati tra i servizi, utilizzando messaggi SOAP, sono compatibili con i parametri dichiarati nella descrizione WSDL dei servizi. I workflow sono nidificati in *scope* che contengono le relazioni tra i partner esterni, le dichiarazioni dei dati del processo, la gestioni delle varie richieste, e le attività che devono essere eseguite. I dati sono scritti e letti attraverso l'utilizzo di *variabili*. Ogni parte del processo da eseguire in BPEL4WS è un *activity*.

Activity Base

Le **activity base** gestiscono le ricezioni, le risposte e le invocazioni ai servizi esterni e sono *receive, reply, invoke* ed *event handlers*. Una delle più importanti *activity* base in BPEL4WS è descritta da *invoke*. Mostriamo un esempio:

```
<invoke name= ‘‘submitGS ’’
 partnerLink= ‘‘JobSubmissionPartner ’’
 portType= ‘‘gs:JobSubmissionPortType"
 operation= ‘‘submitJob"
inputVariable= ‘‘submitreq"
outputVariable= ‘‘submitresp" />
```

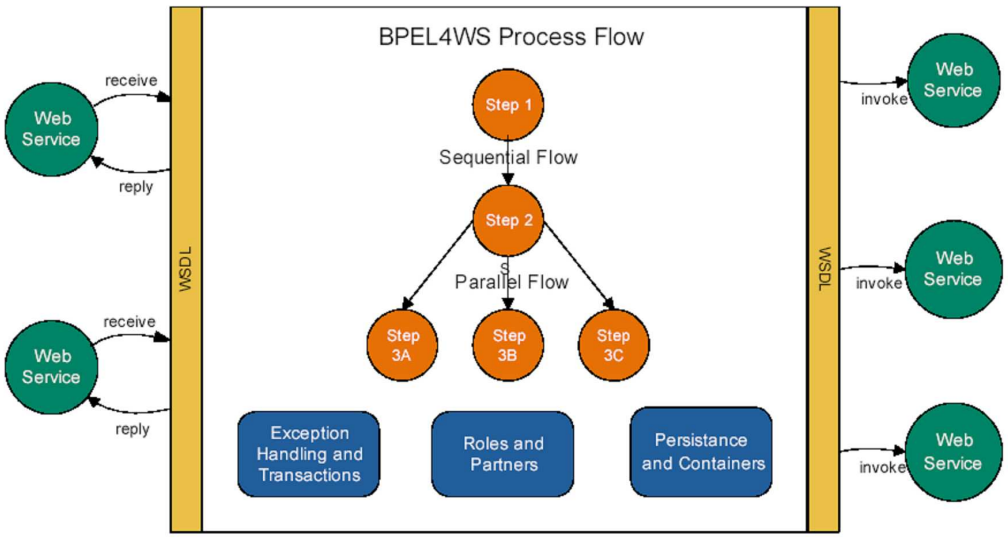

Figura 6.3: Esempio di workflow con l'utilizzo di BPEL4WS

Dall'esempio si osserva che JobSubmissionPartner invoca submitJob e fornisce il port Type JobSubmissionPortType definito nel WSDL. Inoltre osserviamo che il binding del partner, che descrive la modalità di trasmissione dei messaggi, non è compito di BPEL4WS. I partner devono definire i port Type WSDL per ogni interfaccia utilizzata nel workflow e BPEL4WS chiede le definizioni WSDL per avere i link con il partner, specificando il ruolo di ogni interfaccia (portType)(fig.6.3). BPEL4WS può fare riferimento a questi ruoli per descrivere le interazioni tra i servizi, ad esempio per distinguere la spedizione di una richiesta dalla spedizione di una risposta a un'istanza del workflow. BPEL4WS permette le invocazioni ad un servizio sia di tipo sincrone che di tipo asincrone. Ogni process flow BPEL4WS è di fatto un Web Service che può essere invocato spedendo un messaggio SOAP ad un Bpel engine. L'*activity* utilizzato è il *receive*.

```
<receive partnerLink=‘‘client"
 portType=‘‘tns:PolySearchPT"
 operation=‘‘runPolySearch"
 variable=‘‘analysisName"
createInstance=‘‘yes"/>
```

Un'altra importante *activity* che permette di manipolare i dati conservati in variabili è ottenuto utilizzando *assignments*. Un assignment consiste in un

numero di copie di dati da una sorgente a una destinazione. La destinazione può essere un documento XML, il risultato della valutazione di un espressione, o una query che estrae dati da un'altra variabile. La destinazione di una copia è descritta in una variabile nel messaggio WSDL e può, in maniera opzionale, includere una query che determina dove deve essere copiato l'elemento in una struttura dati XML

```
<assign>
 <copy>
  <from expression=‘‘/tmp/molpakout0"/>
  <to variable=‘‘submitreq"
   part=‘‘JobDescription"
   query=‘‘/gs:JobDescription/jsdl:JobDefinition \
           /jsdl:JobDescription/jsdl:Application \
           /jsdl:Output" />
 </copy>
 ...
</assign>
```

Activity Strutturate

Le **activity strutturate** gestiscono il flusso di un processo specificando le attività da elaborare e l'ordine con le quali devono essere elaborate. Esse forniscono costrutti logici quali *sequence, switch, while, flow* e *pick*.

La *sequence* è una semplice aggregazione di attività eseguite nell'ordine specificato. Lo *switch* fornisce una scelta di decisioni e il *while* è un loop. I costrutti in parallelo sono realizzati con il *flow*. *Pick* invece permette di definire scelte non deterministiche.

Di seguito mostriamo l'esecuzione di un'operazione che aspetta 30 secondi e poi invoca l'operazione che restituisce lo stato del job da un port type per il monitoring.

```
<while
  name=‘‘waitForGS"
  condition=‘‘count(bpws:getVariableData('statusresponse',
            'JobStatus','//gs:Property'))&lt; 2">
  <sequence>
   <wait for=‘‘'PT30S'" name=‘‘wait4Property"/>
   <invoke
     name=‘‘statusGS1"
     partnerLink=‘‘JobMonitoringPartner"
```

```
      portType=‘‘gs:JobMonitoringPortType"
      operation=‘‘getJobStatus"
      inputVariable=‘‘statusrequest"
      outputVariable=‘‘statusresponse"/>
  </sequence>
</while>
```

Possiamo concludere che BPEL4WS soddisfa un gran numero di richieste per poter definire workflow scientifici e fornisce un sistema che rilascia i types definiti nei documenti WSDL dei servizi che costituiscono la composizione.

Utilizzando BPEL4WS nella descrizione delle sequenze della Pipeline possiamo avere numerosi vantaggi. In questo modo è possibile sottomettere i job che coinvolgono l'esecuzione di un gran numero di task come la conversione dei dati, il trasferimento dei file, la sincronizzazione dei job e l'analisi dei risultati.

6.3.1 La correlazione

Durante l'esecuzione un processo deve interagire con differenti servizi e non c'è un modo sicuro per sapere a quale particolare istanza è stato indirizzato. BPEL4WS riesce a gestire i task in maniera automatica e orchestra i servizi utilizzando proprio la nozione di correlazione. Una correlazione è l'unico modo di identificare le interazioni di un processo con le altre parti. Una correlazione può essere composta da una o più proprietà; essa è dichiarata durante il primo messaggio scambiato tra il processo e le altre parti e viene riutilizzata ogni volta che si spedisce un messaggio a quella parte (50).

6.3.2 Bpel engine

Active Bpel (17) è un'implementazione open source di Bpel rilasciata da Active Endpoints. E' un ambiente runtime per BPEL scritto in Java. Active Bpel non contiene un tool di validazione per controllare il type system di Bpel ma ha un servizio free per la validazione che può essere utilizzato per trovare eventuali errori. Per quanto riguarda il coordinamento dei servizi e le installazioni, ActiveBpel utilizza un formato che archivia la descrizione dei processi insieme al descrittore e a ogni altra dichiarazione WSDL che non è disponibile on-line. L'archivio è compresso e questo semplifica i trasferimenti con il motore. Una volta che l'archivio è conservato nella directory di installazione, ActiveBpel installa il processo. Se un archivio viene cancellato dalla directory, esso rimuove i file, e se un file viene cambiato esso aggiorna il workflow. Questo permette l'integrazione del processo in qualsiasi ambiente Grid.

Active Bpel riesce a soddisfare le necessità di un Grid service orchestation relativamente ai workflow scientifici.

Altri motori sviluppati sono i seguenti:

- **Active Endpoints ActiveWebflow Server** un motore BPEL completo basato sull' application server J2EE o come applicazione standalone integrata su di un server web.
- **Bexee BPEL Execution Engine** è un motore BPEL open source, basato su J2EE-based BPEL engine.
- **Biztalk Server** il prodotto Microsoft per la gestione dei messaggi e l'integrazione applicativa è stato reso compatibile con lo standard BPEL dalla versione 2004. Si basa su tecnologia COM+ ma è perfettamente integrabile con la piattaforma .NET.
- **Collaxa BPEL Orchestration Server** è un motore basato su application server J2EE. Dalla versione 2 è incluso un ambiente di sviluppo visuale basato su Eclipse. In seguito all'acquisizione di questo prodotto da Oracle Corporation l'ambiente di sviluppo è ora basato anche sulla JDeveloper IDE di Oracle.
- **FiveSight PXE** è un motore BPEL modulare particolarmente focalizzato sull'affidabilità e sulla flessibilità.
- **Oracle BPEL Process Manager** è un motore BPEL che si basa su Oracle Application Server.

6.4 Lo stato dell'arte

Negli ultimi anni ci sono stati grandi interessi sia nel campo scientifico che in quello industriale sulla progettazione e la promulgazione dei workflow. Esistono attualmente molti lavori sulla gestione dei workflow scientifici basati su differenti approcci. DAGMan (26), Taverna (43), GridFlow (8), e GridAnt (2) supportano workflow scientifici senza considerare il punto di vista dei Web Service. DAGMan è utilizzato in Condor scheduling per gestire le dipendenze tra job. Taverna è stato sviluppato per le applicazioni bioinformatiche ed ha un supporto limitato per le invocazioni ai Web Service perchè non permette la manipolazione dei costrutti e le assegnazioni delle variabili che derivano da WSDL. Questo significa che una semplice conversione dei dati richiede la creazione e l'invocazione di altri servizi per le trasformazioni. GridANT è un ambiente per la definizione e il monitoring dei workflow

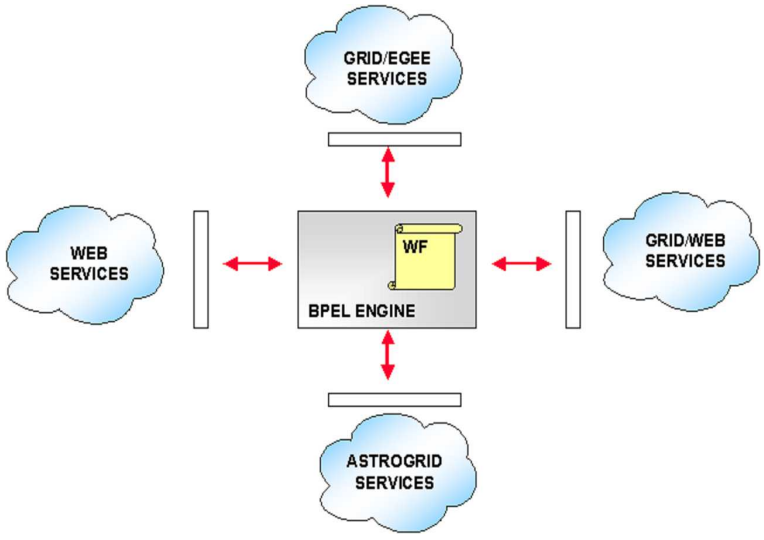

Figura 6.4: Scenario ipotetico con BPEL4WS

basati su Ant. Il linguaggio Ant, comunque, non ha primitive per gestire la sincronizzazione di job asincroni e non supporta molto bene la gestione delle invocazioni ai Grid Service. GridFlow focalizza l'attenzione sull'allocazione delle risorse. In realtà dal nostro punto di vista si vuole separare la nozione di workflow con l'allocazione delle risorse che è delegata invece ai Grid Service nella sottomissione dei job. Rispetto a questi approcci l'utilizzo di Bpel non produce alcun vantaggio se la activity base che devono essere combinate non sono invocazioni ai web service o ai Grid service. Triana (54) e GSFL(34) sono due workflow engine per le applicazione Grid orientate ai servizi. La costruzione di un sistema di workflow scalabile e robusto è molto difficile e le pubblicazioni fatte non includono alcuna discussione sulla scalabilità e la fattibilità di questi approcci. Non è possibile quindi confrontare la loro robustezza con Bpel.

6.5 Conclusioni

Nell'utilizzare BPEL4WS workflow è importante fornire una serie di servizi che interagiscono nei process flow. Un modo per garantire i servizi è l'utilizzo di partner astratti. Essi sono servizi invocati per l'esecuzione di workflow, mappati sui Web Service o mappati attraverso il WSDL port type di cui ogni partner è fornito. Un BPEL4WS workflow può interagire con molti partner alcuni dei quali possono essere Web Service, WSRF service, o altri BPEL4WS workflow (fig. 6.4). Inoltre alcuni partner non sono esposti come

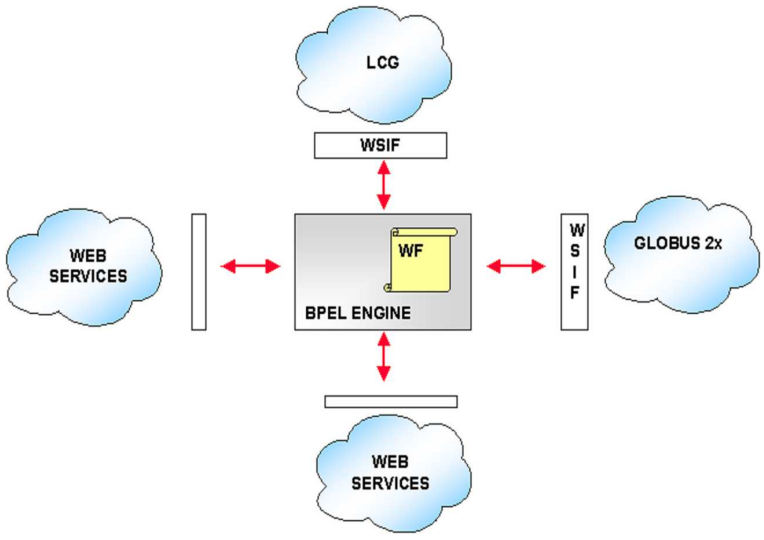

Figura 6.5: Scenario attuale con WSIF

gli attuali Web Service ma attraverso il WSDL descrivono le loro classi Java. L'utilizzo di BPEL4WS non apporta grossi vantaggi se le combinazioni non corrispondo a invocazioni a Web o Grid Service. Attualmente il progetto EGEE sta sviluppando un'infrastruttura Grid in cui una serie di componenti espongono Web Service. Molte altre architetture, come LCG, a cui fa riferimento sia l'INAF che INFN, non sono ancora basate sui Web Service. Una soluzione per superare le difficoltà dell'attuale stato dell'arte dei servizi e, nello stesso tempo, utilizzare un linguaggio standard per la descrizione dei Web Service, è rappresentato da un tool chiamato WSIF (Web Services Invocation Framework) (fig. 6.5).

Il tool riesce ad effettuare il mapping dei metodi di una classe e ad esporre le descrizioni necessarie a BPEL4WS per interagire con i servizi mappati. In questo modo WSIF permette all'utente di interagire con la rappresentazione astratta dei Web Service e permette l'aggiunta dinamica di nuovi binding (16). Il tool invoca i partners che non sono accessibili come dei reali Web Service: ad esempio un servizio può essere utilizzato invocandoo metodi di una classe Java. Nell'analisi dell'utilizzo dei workflow per le applicazioni astronomiche viene proposto un modo per utilizzare un linguaggio di workflow standard e nel contempo interagire con i Grid service che non sono stati esposti come dei Web Service (36).

Attraverso l'utilizzo delle API WSIF è possibile invocare delle classi Java; in particolare un Java binding descrive il modo in cui una classe Java supporta un port type definito. Il Java binding è un WSDL binding che permette di descrivere la classe Java utilizzando WSDL. Sotto queste condizioni si può

accedere a un Grid service attraverso le classi Java utilizzando un tool, chiamato Commodity Grid (CoG) Kits (58). CoG permette agli utenti di Grid, agli sviluppatori e agli amministratori della Grid, di utilizzare un framework higher-level. In particolare Java CoG fornisce un framework per interagire con una serie di servizi che fanno parte di GT2 come GRAM, GASS, GridFTP, MDS. In questo modo è possibile utilizzare BPEL4WS workflow per le infrastrutture Grid che non sono ancora esposte come Web service.

Ringraziamenti

Ringrazio l'Osservatorio Astronomico di Capodimonte, dove ho lavorato per quasi tutta la durata del dottorato, con l'appoggio del Centro di Calcolo ed in particolare dell'Ing. Giulio Capasso.

Ringrazio il mio tutor scientifico, il ricercatore astronomo Enrico Cascone, che con la sua disponibilità e la sua maturità scientifica, ha contribuito alla crescita del mio percorso nel campo della ricerca integrandomi completamente nel mondo delle tecnologie applicate all'astronomia.

Ringrazio il mio tutor burocratico, il prof. Almerico Murli le cui critiche costruttive hanno consentito la mia maturità professionale.

Ringrazio il prof. Leonardo Merola che mi ha dato la possibilità di sviluppare parte dei miei studi all'INFN aprendo una collaborazione con l'OAC.

In particolare ringrazio Gennaro Tortone che ha reso possibile questa collaborazione grazie alla sua disponibilità, e che mi ha insegnato in questi mesi, con la sua grande scrupolosità e le sua innata capacità didattica, le metodologie di studio e di linguaggio, e le nozioni fondamentali sulle nuove tecnologie di calcolo distribuito.

Bibliografia

[1] Jim Almond, Dave Snelling: *UNICORE: Secure and Uniform Access to Distributed Resources via the World Wide Web*, A White Paper, 1998.

[2] Kaizar Amin, Gregor Von Laszewski, Michael Hategan, Nestor J. Zaluzec Shawn Hampton, Albert Rossi: *GridAnt: A Client-Controllable Grid Workflow System*, 37th Hawaii International Conference on System Science, Island of Hawaii, 2004.

[3] P. Andretto, et al.: *Pratical approaches to Grid workload and resource management in the EGEE project*, Proceedings of the International Conference on Computing in High Energy Physics (CHEP2004), Interlaken, Switzerland 2004.

[4] *Astro-Wise Environment, User development manual* November 2004.

[5] G. Avellino et al.: *The EU DataGrid Workload Management System: towards the second major release*, Proceedings of the International Conference on Computing in High Energy Physics (CHEP2003), La Jolla, California, 2003.

[6] J. Basney and M. Livny, *Deploying a High Throughput Computing Cluster* i n R. Buyya (ed.), High Performance Cluster Computing, Vol. 1, Chapter 5, Prentice Hall PTR, 1999.

[7] Francine Berman, et al.: *The GrADS Project: Software Support for High-Level Grid Application Development*, Journal of High Performance Computing Applications, Vol. 15, Number 4, pp. 327-344 Winter 2001.

[8] Junwei Cao, Stephen A. Jarvis, Subhash Saini, Graham R. Nudd: *GridFlow: Workflow Management for Grid Computing*, 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2003), Tokyo, Japan, 2003.

[9] Ethan Cerami: *Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI & WSDL*, O'Reilly, 2002

[10] G. Coulouris, J. Dollimore and T. Kindberg, *Distributed Systems: Concepts and Design* Addison-Wesley, Pearson Education, 2001.

[11] Karl Czajkowski et. al.: *The WS-Resource Framework Version 1.0* http://www.globus.org/wsrf , 2004.

[12] Ewa Deelman, et al.: *Mapping Abstract Complex Workflow onto Grid Environments*, Journal of Grid Computing, 25-39, 2003.

[13] E. Deelman, et al.: *Pegasus: Mapping Scientific Workflow onto the Grid* Across Grids Conference 2004, Nicosia, Cyprus, 2004.

[14] E. Deelman, J. Blythe, Y. Gil, and C. Kesselman. *Workflow Management in GriPhyN.* The Grid Resource Management, Kluwer, Netherlands, 2003.

[15] Rubing Duan, Thomas Fahringer, Radu Prodan, Jun Qin, Alex Villazon, Marek Wieczorek: *Real World Workflow Applications in the Askalon Grid Environment*, Advances in Grid Computing - EGC, European Grid Conference, 464-473, Amsterdam, The Netherlands, 2005.

[16] Matthew J. Duftler, Nirmal K. Mukhi, Aleksander Slominski and Sanjiva Weerawarana: *Web Services Invocation Framework (WSIF)*, Proceedings of the OOPSLA Workshop on Object-Oriented Web Services, October 2001.

[17] Wolfgang Emmerich et. al: *Grid Service Orchestration using the Business Process Execution Language (BPEL)*, UCL-CS, Research Note RN/05/07. Gower St, London WC1E 6BT, UK.

[18] T. Fahringer, S. Pllana, and A. Villazon: *AGWL: Abstract Grid Workow Language* , International Conference on Computational Science, Programming Paradigms for Grids and Metacomputing Systems, Krakow, Poland, 2004.

[19] Thomas Fahringer, Jun Qin, Stefan Hainzer: *Specification of Grid Workflow Applications with AGWL: An Abstract Grid Workflow Language*, IEEE International Symposium on Cluster Computing and the Grid 2005 (CCGrid 2005), Cardiff, UK, 2005.

[20] Thomas Fahringer, Alexandru Jugravu, Sabri Pllana, Radu Prodan, Clovis Seragiotto Junior and Hong-Linh Truong: *ASKALON: a tool set for cluster and Grid computing*, Concurrency and Computation: Practice & Experience, 17/2-4, 2005.

[21] Ian Foster: *What is the Grid? A Three Point Checklist* Grid Today, July 20, 2002.

[22] I. Foster: *What is the Grid? A Three Point Checklist* , Grid Today, Vol. 1, No. 6, 2002.

[23] I. Foster, C. Kesselman, S. Tuecke: *The Anatomy of the Grid: Enabling Scalable Virtual Organizations* International J. Supercomputer Applications, 15(3), 2001.

[24] Ian Foster: *The Grid: A New Infrastructure for 21st Century Science*, Physics Today, February, 2002.

[25] I. Foster, C. Kesselman: *The Grid 2: Blueprint for a New Computing Infrastructure* , 2nd edition, 2003.

[26] J. Frey (2002): *Condor DAGMan: Handling Inter-Job Dependencies* Tech. rep., University of Wisconsin, Dept. of Computer Science, 2002.

[27] Tom Goodale et al.: *The Cactus Framework and Toolkit: Design and Applications*, Vector and Parallel Processing - 5th International Conference, Lecture Notes in Computer Science, 2002.

[28] A.Grado, M. Capaccioli, R. Silvotti, M. Pavlov, F. Getman, A. Volpicelli, J.M. Alcal, M. Radovich, E. Puddu, G.Capasso, G. Busarello, E. Cappellaro, M. Marconi and G. Longo: *Pipeline and data flow for the INAF-Capodimonte guaranteed observing time at VLT Survey Telescope*, Astron.Nachr./AN 325(2004) 6/7

[29] A.S. Grimshaw, W.A. Wulf, J.C. French, A.C. Weaver and P.F. Reynolds *Legion: The Next Logical Step Toward a Nationwide Virtual Computer* Technical report No. CS-94-21, June 1994.

[30] A.S. Grimshaw, W.A. Wulf *Legion - a View From 50,000 Feet* in Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing. IEEE Computer Society Press: Los Alamitos, California, August 1996.

[31] Elliotte Rusty Harold & W. Scott Means: *XML in a Nutshell* O'Reilly Second Edition, 2002.

[32] P. Harrison: *A Proposal for a Common Execution Architecture Version 1.2* http://www.ivoa.net , IVOA Grid Working Group - Common Execution Architecture Internal Draft.

[33] W.E. Johnston, A. Ferrari and K. Holcomb *A Different Perspective on the Question of what Is a Grid?* Grid Today, Vol. 1, No. 9, 2002.

[34] P. W. S. Krishnan & G. v. Laszewski: *GSFL: A workflow framework for grid services* , Technical Report Preprint ANL/MCS-P980-0802, Argonne National Laboratory, 2002.

[35] G. Lindahl, A. Grimshaw, A. Ferrari and K. Holcomb *Metacomputing - what's in it for me* White paper, 1998.

[36] V. Manna, G. Tortone, E. Cascone, G.Capasso:*Definition of astronomical data analysis workflows on a service-oriented grid architecture using Business Process Execution Language*, Proceedings of the Astronomical Data Analysis Software and System XV, San Lorenzo Escorial en El Madrid, Spain, 2005.

[37] Valeria Manna, Giulio Capasso, Enrico Cascone: *Astronomica lmage Processing using System Resource Management*, Memorie della SAIt, 2004.

[38] A. Mayer, S. McGough, N. Furmento, W. Lee, S. Newhouse, and J. Darlingto *ICENI Dataflow and Workflow: Composition and Scheduling in Space and Time* In UK e-Science All Hands Meeting, Nottingham, UK, IOP Publishing Ltd, Bristol, UK, 627-634, 2003.

[39] Karen M. McCann, Maurice Yarrow, Adrian DeVivo, Piyush Mehrotra: *ScyFlow: An Environment for the Visual Specification and Execution of Scientific Workflows*, In Proceedings of Workflow in Grid Systems Workshop in GGF10, at Berlin, Germany, 2004.

[40] Stephen McGough, Laurie Young, Ali Afzal, Steven Newhouse, John Darlington: *Workflow Enactment in ICENI*, UK e-Science All Hands Meeting, p. 894-900, Nottingham, UK, 2004.

[41] Derek Miers, Enix Consulting, United Kingdom: *Workflow Handbook*, Workflow Management Coalition, 2005.

[42] Zsolt Nemeth,Vaidy Sunderam: *Characterizing Grids: Attributes, Definitions, and Formalisms*, Journal of Grid Computing, Vol 1 No. 1., pp. 9-23.

[43] Tom Oinn, et al.: *Taverna: a tool for the composition and enactment of bioinformatics workflows* Bioinformatics Vol.20 no. 17, pages 3045-3054, 2004.

[44] M. Oliviero, G. Severino, Th. Straus *The VAMOS Data Analysis Pipeline* Proceedings of the SOHO VI/GONG workshop, p. 275, Boston, 1998.

[45] Maurizio Oliviero *Misura delle Oscillazioni Solari con Filtro Magneto-Ottico* http://vamos.na.astro.it/

[46] F. Pasian, L. Benacchio *DRACO and the Italian Participation in Virtual Observatory Activities* Proceedings of Astronomical Data Analysis Software and Systems XIII, Vol. 314, pg. 257, San Francisco, 2003.

[47] M. Pavlov, J.M. Alcal, A.Grado, E. Cascone, G.Capasso, V.Manna: *The VST data reduction: an application for the GRID infrastructure*, Astronomical Data Analisys Software and Systems XV - ASP Conference Series, Vol.XXX, Pasadena California, USA, 2004.

[48] Chris Peltz: *Web Services orchestration a review of emerging technologies, tools, and standards*, Hewlett Packard Co, January 2003.

[49] Peter J. Quinn: *Virtual Observatories: the Future of Astronomical Information*, Proceedings of a Conference held at Charles University Prague, Czech Republic July 2-5, 2002.

[50] Matthieu Riou: *WS-BPEL Guide*, http://www.smartcomps.org, 10 Dec, 2004.

[51] S. Sathish and Jack J. Dongarra: *Self Adaptivity in Grid Computing*, Concurrency and Computation: Practice and Experience, Volume 17, Number 2-4, , 235-257, 2005.

[52] Aleksander Slomiski: *On Using BPEL Extensibility to Implement OGSI and WSRF Grid Workflows*, GGF10 Grid WorkFlow Workshop, 25 January 2004.

[53] Dr. Ian Taylor, Matthew Shields, Dr. Ian Wang: *Resource Management of Triana P2P Services*, Grid Resource Management, 451-462, Kluwer Academic Press, 2004.

[54] I. Taylor, et al.: *Triana Applications within Grid computing and Peer to Peer Environments*, Journal of Grid Computing 1(2):199-217, 2003.

[55] S. Tuecke et. al.: *Open Grid Services Infrastructure (OGSI) Version 1.0* http://www.ggf.org/ogsi-wg, 2003.

[56] Edwin A. Valentijn and Konrad Kuijken: *ASTRO-WISE An Astronomical Wide-Field Imaging System for Europe*, Conference Towards an International Virtual Observatory, pg. 19, Garching 2004,

[57] Gregor von Laszewski: *The Grid-Idea and Its Evolution*, Technical Reports, Argonne National Laboratory, USA, 2005

[58] Gregor von Laszewski: *The Java CoG Kit User Manual, version 4.0* Mcs technical memorandum, Argonne National Laboratory, Mathematics and Computer Science Division, Argonne, USA, 2004.

[59] Nicholas A. Walton, Andrew Lawrence, Tony Linde, : *AstroGrid: Initial Deployment of the UK's Virtual Observatory*, Astronomical Data Analysis Software and Systems XIII, ASP Conference Series, Vol. 314, Strasbourg, France, 2003.

[60] Sanjiva Weerawarana, Francisco Curbera: *Business Process with BPEL4WS: Understanding BPEL4WS*, http://www-128.ibm.com/developerworks/

[61] Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey, Donald F. Ferguson: *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More* Prentice Hall, 2005

[62] R. Williams, B. Hanisch, T. Linde, J. McDowell, R. Moore, F. Ochsenbein, M. Ohishi, G. Rixon, A. Szalay, D. Tody: *Virtual Observatory Architecture Overview Version 1.0* http://www.ivoa.net

[63] R. Williams, B. Hanisch, T. Linde, J. McDowell, R. Moore, F. Ochsenbein, M. Ohishi, G. Rixon, A. Szalay, D. Tody: *AstroGrid Job Management User Guide* , http://www.astogrid.org

[64] Jia Yu, Rajkumar Buyya: *A Novel Architecture for Realizing Grid Workflow using Tuple Spaces*,Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, USA, 2004.

[65] Jia Yu, Rajkumar Buyya: *A Taxonomy of Workflow Management Systems for Grid Computing*, Technical Report, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, 2005.