Capitolo 1

Introduzione

Il modello di computing è radicalmente cambiato negli ultimi anni. Sono infatti sempre di più i centri di calcolo, sia nel mondo industriale che scientifico, in cui il singolo computer centrale di tipo mainframe, a cui tutti gli utenti accedevano attraverso terminali remoti, è stato sostituito con workstation, collegate tra loro attraverso una rete locale (*LAN: Local Area Network*). Grazie infatti ai notevoli sviluppi tecnologici, è ora possibile acquistare workstation con un rapporto prestazioni/prezzo nettamente superiore a quello dei supercalcolatori di tipo mainframe [8] [9] [22] [30]. Anche l'impiego di personal computer è sempre più diffuso, visto che sono in grado di fornire elevate prestazioni a costi molto competitivi.

L'utilizzo di un sistema di calcolo distribuito offre altri vantaggi, oltre alla semplice convenienza economica: rispetto ai sistemi costituiti da un unico calcolatore, è infatti più facile gestire variazioni della "dimensione" dei problemi da trattare, è più semplice configurare il sistema secondo le proprie specifiche esigenze, ecc... Se poi si considera un ambiente eterogeneo, dove cioè i calcolatori possono essere di diversa architettura, è possibile soddisfare in modo più efficiente le diverse richieste computazionali. Non esiste infatti, e probabilmente non esisterà nemmeno in futuro, un singolo modello di calcolatore adatto a risolvere tutti i tipi di problemi: certe architetture si prestano meglio di altre ad eseguire particolari tipi di codice (per fare un esempio, le architetture SIMD sono più adatte per le computazioni matriciali, mentre le architetture MIMD sono particolarmente indicate per applicazioni costituite da moduli con una limitata intercomunicazione [27]) [24].

Vi sono però varie difficoltà e problematiche, ancora non del tutto risolte, che devono essere considerate in un ambiente di calcolo distribuito ed eterogeneo: è necessario trattare opportunamente le differenze tra i sistemi considerati (incompatibilità binaria, diversa rappresentazione interna dei dati, ecc...), il programmatore deve gestire direttamente diversi dettagli implementativi (la sincronizzazione e la comunicazione tra i vari moduli del programma, il mapping dei processi ai processori disponibili, ecc...), sono necessari adeguati strumenti per il *code profiling* (il procedimento che permette di fornire una stima del tempo di esecuzione delle varie applicazioni sui diversi calcolatori), per il *benchmarking*, per il *mapping* (l'assegnazione dei processi ai processori disponibili), ecc... Inoltre l'attività di amministrazione di un ambiente costituito da decine, o centinaia, o migliaia di workstation può essere molto complessa e richiedere molte risorse.

Una delle realtà in cui il modello di calcolo è cambiato, e in cui il problema della gestione efficiente di un sistema di calcolo distribuito ed eterogeneo è molto sentito, è l'ambiente della fisica delle alte energie (*HEP: High Energy Physics*). La ricerca fisica di tipo teorico e di tipo sperimentale è infatti fortemente vincolata dalle risorse computazionali disponibili limitate: se non vi fossero questi limiti potrebbe essere fatta ricerca ad un diverso livello. È quindi di fondamentale importanza poter gestire nel miglior modo possibile le risorse di calcolo disponibili.

Un tipico programma di elaborazione che deve essere considerato in un esperimento di fisica delle alte energie si compone di unità computazionali elementari, chiamate *task*, ognuna delle quali gestisce un particolare aspetto del problema. I vari task vengono poi combinati tra loro, costituendo le cosiddette *production*. Solitamente una production è costituita da una serie di task posti in "cascata" l'uno con l'altro, ma a volte è necessario considerare strutture più complesse. Le production vengono in genere utilizzate per elaborare un numero di dati molto elevato.

Attualmente non esiste un modello unico per la gestione di queste elaborazioni. Esistono specifici sistemi, che sono però strettamente vincolati al tipo di job da eseguire e alla configurazione dell'ambiente di calcolo considerato. Questi sistemi presentano poi vari problemi, quali il livello di automatizzazione piuttosto limitato (è richiesto un controllo e una gestione praticamente costante da parte dell'utente), un utilizzo spesso non ottimale delle risorse di calcolo disponibili, ecc...

Per gli esperimenti della prossima generazione (primi fra tutti gli esperimenti LHC al Cern) i problemi saranno ancora più complessi, poiché il volume dei dati da gestire sarà molto superiore a quello degli esperimenti attuali.

In questa tesi viene proposto un sistema, chiamato DEMI (Distributed hEterogeneous coMputing envIronment) per la programmazione e la gestione di elaborazioni in un ambiente di calcolo distribuito ed eterogeneo. Attraverso questo sistema l'utente ha la possibilità di definire propri programmi (production), costituiti da un insieme di processi sequenziali (task). I vari task possono essere codificati usando linguaggi di programmazione diversi, e ognuno di essi può essere implementato anche per più piattaforme.

I task di una produzione possono essere eseguiti in maniera concorrente, rispettando però le relazioni di dipendenza sui dati. Un task, infatti, può aver bisogno per l'elaborazione di dati prodotti da altri task: finché l'esecuzione di questi task non viene completata, e quindi finché non sono stati prodotti i relativi output, il task considerato deve rimanere in stato di attesa.

Le produzioni possono essere rappresentate attraverso grafi "data-flow" di tipo aciclico (*DAG: Directed Acyclic Graph*), in cui i nodi rappresentano i processi (task), mentre gli archi rappresentano i vincoli di dipendenza sui dati. La regola di firing associata a ciascun nodo del grafo, cioè la condizione che definisce quando l'esecuzione del processo associato al nodo può cominciare, è *data-driven*: l'elaborazione può iniziare quando "arrivano" al nodo i dati, ovvero quando tutti gli input del task sono disponibili.

Nel formalismo grafico adottato per la rappresentazione delle produzioni, gli archi del grafo intersecano dei "box", che rappresentano i dati (sotto forma di file) prodotti dai task, in attesa di essere usati come input da altri task.

L'ambiente di calcolo che deve essere considerato è composto da un insieme di siti, distribuiti geograficamente, connessi tra loro attraverso una rete geografica. Ogni sito rappresenta un metasistema: un sistema di elaborazione distribuito, costituito da calcolatori indipendenti ed eterogenei, collegati tra loro attraverso una rete locale. La gestione di ogni sito è affidata ad un amministratore locale, che in particolare deve configurare e gestire le risorse locali, mentre un amministratore globale è responsabile delle operazioni che riguardano il sistema complessivo.

Anche se si fa riferimento alle elaborazioni tipiche di un esperimento di fisica delle alte energie, quanto detto può essere applicato ad altre realtà in cui si devono affrontare problemi analoghi.

La tesi presenta lo studio e l'analisi del problema, e la progettazione e implementazione di un primo prototipo del sistema.

Come metodologia di sviluppo è stato adottato il ciclo di vita a cascata: dopo un'analisi preliminare del problema, sono stati definiti in maniera dettagliata i requisiti da considerare. È stata poi realizzata la progettazione del sistema, definendo i vari moduli e le relative funzionalità.

Per gestire in maniera persistente e in modo efficiente le varie informazioni (riguardanti l'ambiente di calcolo, le configurazioni delle elaborazioni, lo stato dell'esecuzione) è stata definita un'apposita base di dati, per la cui progettazione concettuale si è utilizzato il modello a oggetti. Lo schema ottenuto è stato poi trasformato in uno schema logico relazione, per permettere l'implementazione della base di dati in un sistema di questo tipo. La base di dati è stata organizzata gerarchicamente in una parte globale, per le informazioni che riguardano il sistema complessivo e, in ogni sito, in una parte locale, per i dati specifici del sito considerato.

È stato implementato un primo prototipo del sistema: in questa prima fase si è considerato un ambiente costituito da un singolo sito. Il sistema è stato integrato con un *load sharing system*, in grado di gestire in maniera efficiente l'esecuzione di job in un ambiente di calcolo distribuito. Infine si è verificata la funzionalità del sistema, considerando diversi tipi di test.

Nel capitolo 2 viene descritto l'ambiente della fisica delle alte energie, considerando quali sono i problemi computazionali che si devono gestire, con riferimento sia agli esperimenti attuali, che a quelli della prossima generazione. Nel capitolo 3 viene introdotto il sistema DEMI, le cui funzionalità e caratteristiche sono descritte nel capitolo 4. Nel capitolo 5 vengono presentati alcuni sistemi di load sharing, che permettono di gestire in maniera ottimale le elaborazioni in un ambiente di calcolo distribuito. I capitoli 6 e 7 descrivono la progettazione del sistema DEMI: nel capitolo 6 viene presentato il progetto della base di dati utilizzata per la gestione delle varie informazioni, mentre nel capitolo 7 viene descritta l'architettura software del sistema. Nel capitolo 8 si descrive la fase di implementazione, e si presentano esempi di utilizzo del sistema DEMI. Il capitolo 9 conclude la tesi.