Capitolo 8

Implementazione e verifica

Alla fase di progettazione è seguita la realizzazione di un primo prototipo del sistema DEMI. Obiettivo fondamentale è stato quello di verificare il corretto funzionamento del sistema, e individuare eventuali problemi o anomalie. Comunque sono già stati implementati la maggior parte dei requisiti definiti inizialmente.

In questo capitolo vengono anche presentati alcuni esempi di utilizzo del sistema. I risultati di queste verifiche sono senz'altro positivi: è stata confermata la "validità" del progetto, e sono state individuate le direzioni verso cui proseguire lo sviluppo.

8.1 Implementazione

Il primo prototipo del sistema DEMI è stato implementato su piattaforma Digital Unix, utilizzando il linguaggio C++. Quindi i moduli Global Interface, Local Interface, DEMI Global Control System e DEMI Local Control System possono essere eseguiti su un computer Alpha con sistema operativo Digital Unix. Il porting ad una qualsiasi altra piattaforma Unix, comunque, non comporta nessun problema, in quanto non si sono utilizzate specifiche funzionalità implementate solamente nel sistema operativo Digital Unix. Per l'esecuzione dei vari job, invece, possono essere usati computer di tutte le piattaforme su cui è installato il sistema di load sharing considerato.

Per l'implementazione della base di dati si è impiegato il sistema MySQL [5], un database server relazione di pubblico dominio. L'accesso al database da parte del DEMI Global Control System e del DEMI Local Control System è stato realizzato attraverso l'API ODBC (Open Data Base Connectivity), considerando il driver manager iODBC (intrinsic Open Data Base Connectivity) [2] e il driver ODBC per il sistema MySQL (la descrizione delll'API ODBC è riportata in appendice D).

Come sistema di load sharing si è utilizzato il sistema LSF (presentato nel paragrafo 5.1.2), di cui si sono usate le relative API.

Obiettivo fondamentale nella realizzazione del prototipo è stato quello di verificare la correttezza delle fasi di analisi e progettazione, individuare eventuali problemi e, più in generale, valutare l'effettiva "validità" del sistema complessivo. Comunque già in questo prototipo sono stati implementati la maggior parte dei requisiti definiti inizialmente. Il sistema è infatti in grado di gestire in maniera "automatica" elaborazioni (anche complesse) in un ambiente eterogeneo costituito da workstation Unix e/o PC con sistema operativo Windows NT. L'utente non deve occuparsi direttamente dei vari dettagli implementativi: deve solo definire l'elaborazione, e attendere i risultati da parte del sistema.

Anche il requisito di un sistema self managing è stato realizzato: tutte le informazioni, infatti, vengono gestite in maniera persistente, e quindi, in caso di crash, quando il sistema si "riavvia", esso è in grado di individuare qual era lo stato al momento in cui si è verificato il problema, e può quindi agire di conseguenza.

Non è invece stata ancora realizzata un'interfaccia grafica: al momento l'utente interagisce con il sistema attraverso un'interfaccia di tipo testuale.

8.2 Analisi di un caso reale

Per verificare l'effettiva funzionalità del sistema, lo si è voluto innanzittutto provare con un'applicazione reale di fisica delle alte energie.

Viene di seguito descritto il tipo di elaborazione considerata, analizzato l'ambiente di esecuzione, per presentare infine i risultati.

8.2.1 Descrizione dell'elaborazione

In questo primo test si è considerata una classica elaborazione simulazione-ricostruzione dell'esperimento DELPHI.

La produzione utilizzata è mostrata in figura 8.1.

Il task *Delsim* è il task che gestisce la fase di simulazione. All'input *Inp1*, di tipo singleton, viene associato un file di titoli, che definisce il tipo, il numero di eventi da generare, e altri parametri. Il data box *Out1*, di tipo stream, rappresenta i raw data simulati generati da *Delsim*. All'inizio viene generato un numero casuale, che viene usato nel corso dell'elaborazione di Delsim: in questo modo viene generata una sequenza di file "diversi". I dati generati da Delsim vengono usati dal task *Delana*, che effettua la ricostruzione, producendo così eventi simulati ricostruiti,

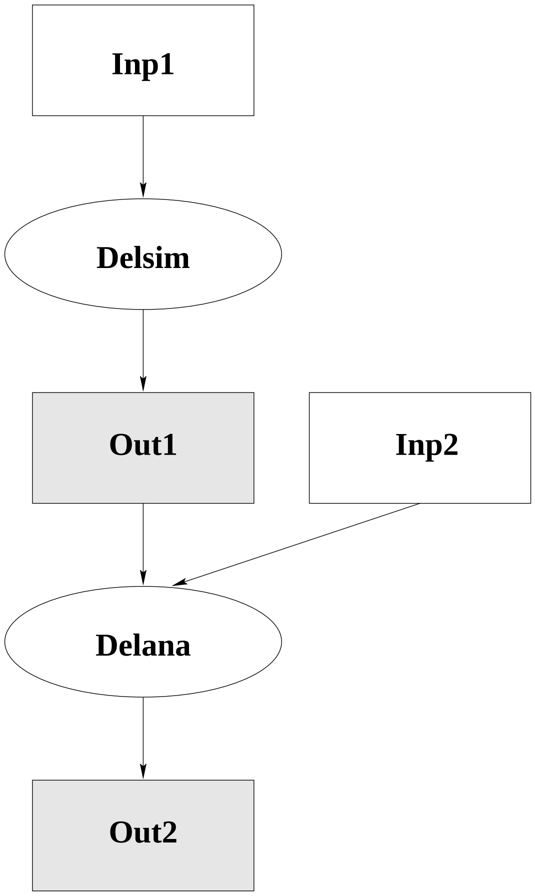

Figura 8.1: La produzione considerata nel primo test

associati al data box *Out2*. *Delana* ha anche un secondo input (rappresentato dal data box *Inp2*): ad esso viene associato un file di titoli che definisce alcune costanti che devono essere considerate nell'elaborazione.

In questo test si sono considerate due architetture: Digital-Unix e Hp-Ux: i task *Delsim* e *Delana* sono stati implementati per entrambe le piattaforme. Il "costo" dei task è lo stesso per le due architetture, poiché il tempo di esecuzione dipende esclusivamente dalle prestazioni dei computer, e non da caratteristiche specifiche del tipo di architettura.

Considerando questa produzione, si è sottomessa una richiesta computazionale. Non essendovi input di tipo stream si è specificato il numero di iterazioni: è stato scelto un valore pari a 80. All'input di tipo singleton *Inp1* si è associato come singleton input file un file di titoli, in cui si indica di generare 50 eventi di tipo QQPS (quindi in totale nelle 80 iterazioni vengono prodotti e ricostruiti 4000 eventi).

Mediamente per eseguire *Delsim* su un'alpha station 3000/600, per la produzione di 50 eventi QQPS, sono necessari circa 100 minuti, mentre per la ricostruzione (attraverso *Delana*) di questi 50 eventi su un calcolatore dello stesso modello, sono necessari circa 30 minuti. Quindi, se si considerasse un'elaborazione sequenziale su una singola alpha station 3000/600 utilizzata a tempo pieno, sarebbero necessari più di sette giorni.

Per quel che riguarda la dimensione massima dei singoli file associati ai data box di output, questa è pari a 28 Mbyte per il data box *Out1*, e 6 Mbyte per il data box *Out2*.

8.2.2 Descrizione dell'ambiente

Per l'esecuzione si è considerato l'ambiente rappresentato in figura 8.2.

Si sono utilizzate in totale otto workstation, di due diverse architetture: sei alpha station con sistema operativo Digital Unix 4.0B (architettura *AXP-DU*), e due workstation HP con sistema operativo HP-UX 10.20 (architettura *HP-UX*). Il Local Scheduler è stato installato su un'altra alpha station. In tabella 8.1 sono riportate le caratteristiche di questi computer (il campo 'Performances' rappresenta il corrispondente attributo del Local Computer Database, e i relativi valori fanno riferimento ai CERN benchmark).

Per la memorizzazione dei file si sono utilizzati due disk buffer, di capacità rispettivamente di 2 Gbyte e 1 Gbyte, fisicamente installati su due alpha station. Questi disk buffer sono stati condivisi con le altre workstation (sia Digital che HP) via NFS.

Per quel che riguarda la configurazione del batch system LSF, si è definita

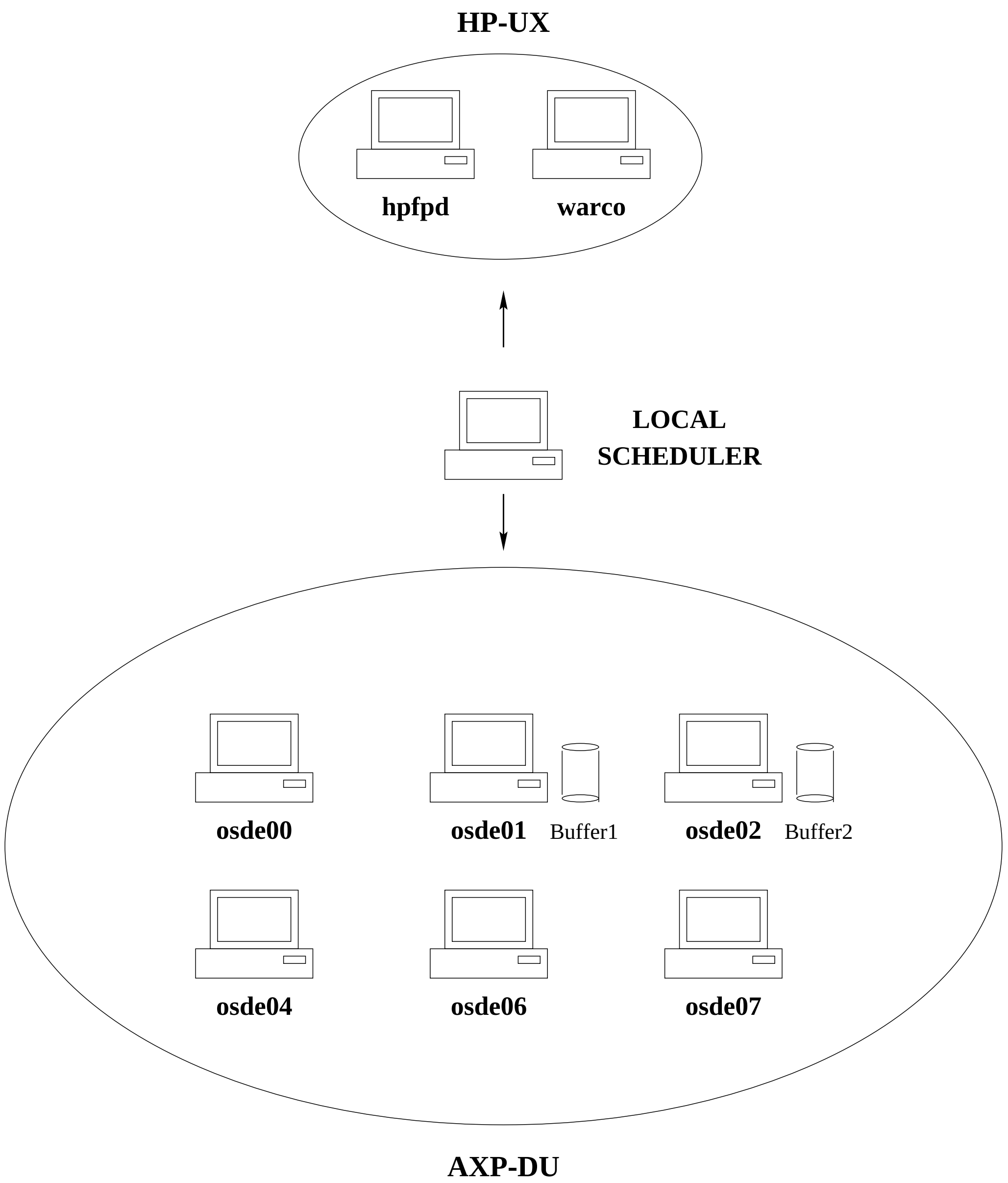

Figura 8.2: Ambiente considerato nel primo test

Nome	Modello	Sistema Operativo	Architettura	Performances
osde00	Alpha Station 600 5/266 128 MB	Digital Unix 4.0B	AXP-DU	34
osde01	Alpha Station 3000/600 64 MB	Digital Unix 4.0B	AXP-DU	12
osde02	Alpha Station 3000/200 64 MB	Digital Unix 4.0B	AXP-DU	7
osde04	Alpha Station 3000/400 64 MB	Digital Unix 4.0B	AXP-DU	9
osde06	Alpha Station 3000/600 64 MB	Digital Unix 4.0B	AXP-DU	12
osde07	Alpha Station 3000/600 64 MB	Digital Unix 4.0B	AXP-DU	12
hpfpd	HP Workstation 9000/735 96 MB	HP-UX 10.20	HP-UX	12
warco	HP Workstation 9000/735 96 MB	HP-UX 10.20	HP-UX	12

Tabella 8.1: Computer usati nel test

un'unica coda per tutte queste workstation, configurata in modo che i job del sistema DEMI possono essere eseguiti su un computer solamente se esso non è usato per altre elaborazioni batch, o in maniera interattiva. Più un dettaglio ai job di questa coda è stata assegnata una priorità più bassa rispetto a quelli delle altre code, e questi job vengono eseguiti con un valore di nice pari a 20. Non ci possono essere due o più task DEMI in esecuzione contemporaneamente. Un job di questa coda può iniziare l'esecuzione (o se ne può fare il resume da uno stato di suspend) se la media, calcolata sull'ultimo minuto, del numero di processi in attesa di usare il processore è minore o uguale a 1, e se non c'è attività interattiva da almeno cinque minuti. Un job viene invece sospeso quando la media dell'ultimo minuto del numero di processi in attesa di utilizzare il processore è maggiore o uguale a 2.5, o quando c'è attività interattiva da almeno un minuto.

Non si è utilizzato il meccanismo del checkpointing: quindi viene gestito il suspend e il resume dei job, che però non vengono "migrati" tra i diversi computer.

8.2.3 Analisi dei risultati

L'elaborazione è iniziata alle ore 12:15 di un giovedì. L'ultimo task ha completato la sua esecuzione alle ore 17.15 del giorno successivo (quindi 29 ore dopo).

Nelle figure 8.3 e 8.4 si riporta su che calcolatori sono stati eseguiti rispettivamente gli 80 task *Delsim* e gli 80 task *Delana*.

Nelle figure 8.5 e 8.6 si indica invece quando questi task sono stati completati sui vari computer.

La tabella 8.2 indica, per i vari calcolatori, il tempo totale speso per l'esecuzione dei task *Delsim/Delana* (il campo 'Tempo esecuzione'), e il tempo invece in cui i task sono rimasti in stato di suspend (il campo 'Tempo susped') poiché i relativi computer erano usati per altre attività (sostanzialmente uso interattivo da parte degli utenti). Il grafico di figura 8.7 mostra quando questi computer hanno elaborato i task *Delsim/Delana* (linee verdi), e quando questi task sono rimasti in stato di suspend (linee rosse).

Come ci si poteva aspettare, in genere i job sono stati in esecuzione ininterrottamente di notte, mentre di giorno le workstation sono spesso state usate dagli utenti (per *osde00*, la workstation a più elevate prestazioni, lo stato di suspend è stato prevalente rispetto allo stato di esecuzione).

Le prestazioni misurate sono piuttosto elevate. Per la simulazione e ricostruzione il throughput è stato infatti di circa 3300 eventi al giorno mentre, se si fosse utilizzata a tempo pieno una singola alpha station 3000/600, il throughput sarebbe stato di circa 550 eventi al giorno (quindi con il sistema DEMI è stato fatto il lavoro di sei alpha station 3000/600, utilizzate a tempo pieno). È importante sottolineare

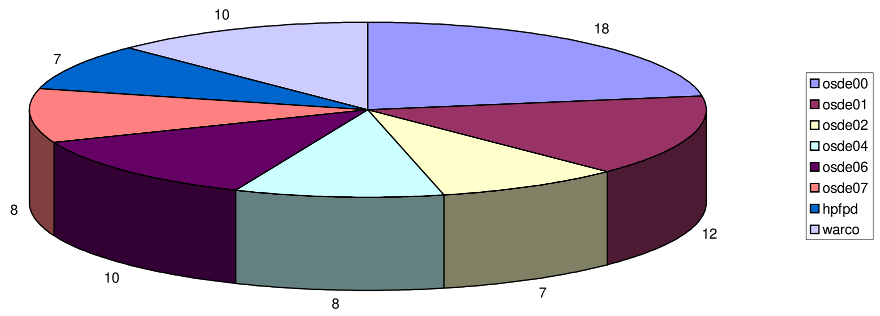

Figura 8.3: Task *Delsim* eseguiti nei vari calcolatori

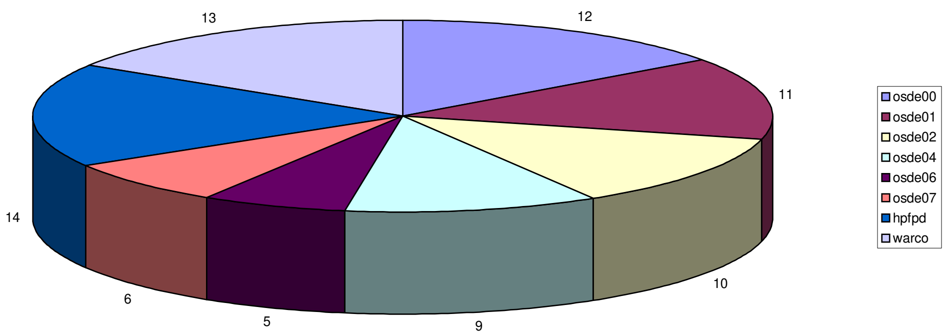

Figura 8.4: Task *Delana* eseguiti nei vari calcolatori

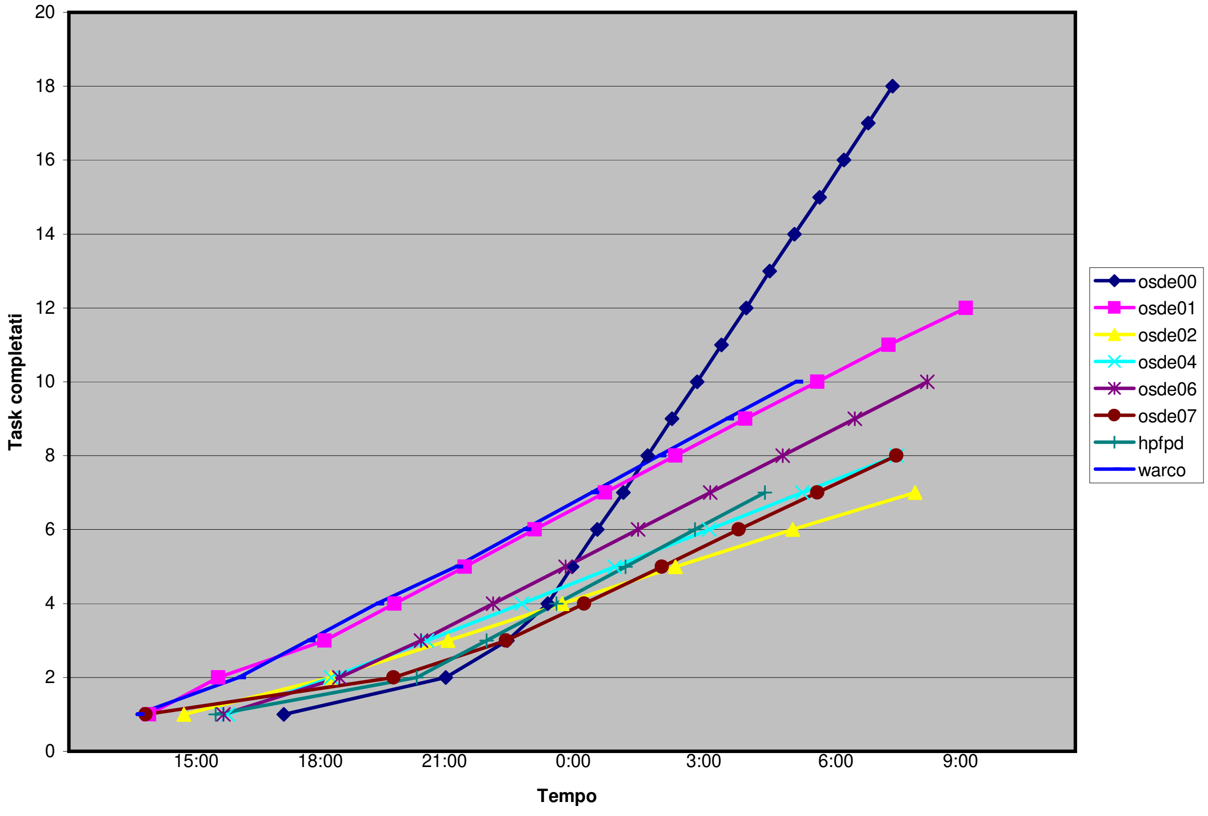

Figura 8.5: Task *Delsim* completati sui vari calcolatori

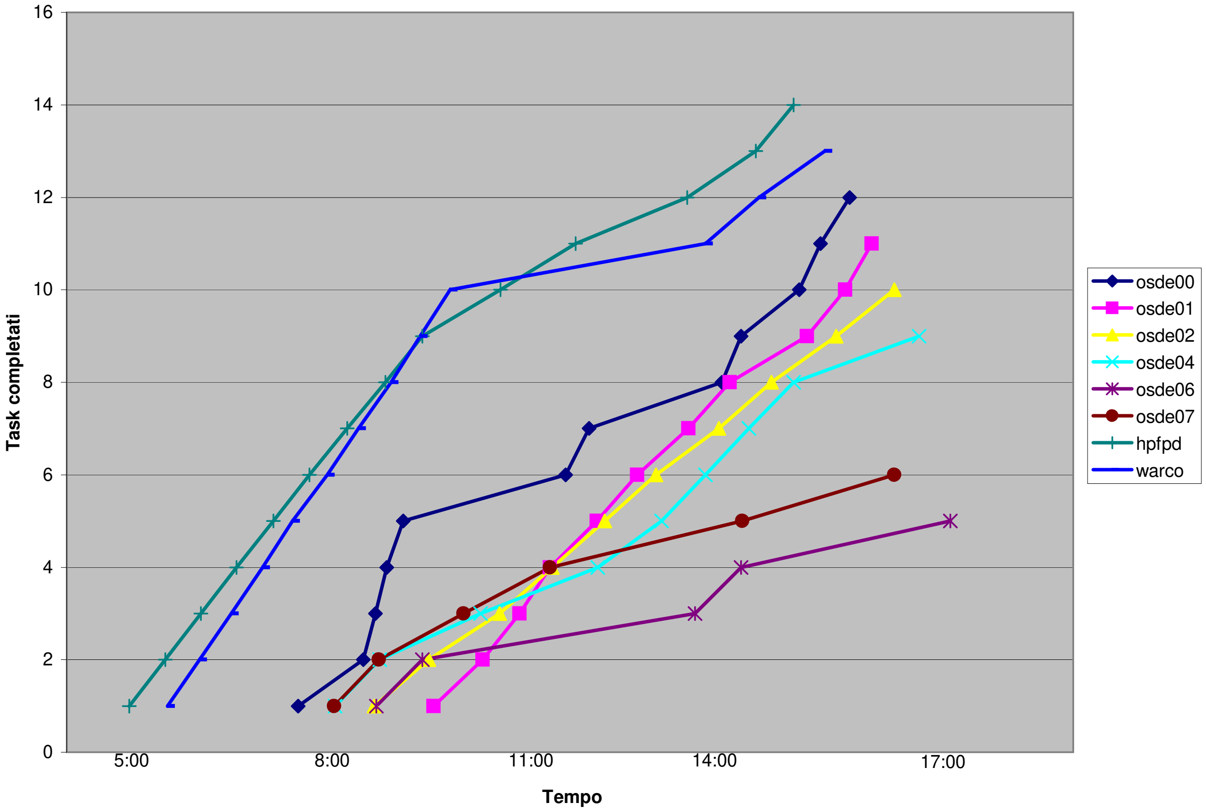

Figura 8.6: Task *Delana* completati sui vari calcolatori

Computer	Tempo esecuzione	Tempo suspend
osde00	47503 s.	51379 s.
osde01	90445 s.	9780 s.
osde02	99987 s.	1060 s.
osde04	84330 s.	18171 s.
osde06	71267 s.	32930 s.
osde07	66425 s.	34640 s.
hpfpd	69735 s.	25762 s.
warco	79576 s.	18175 s.
	609268 s.	191897 s.

Tabella 8.2: Utilizzo dei vari calcolatori nel primo test

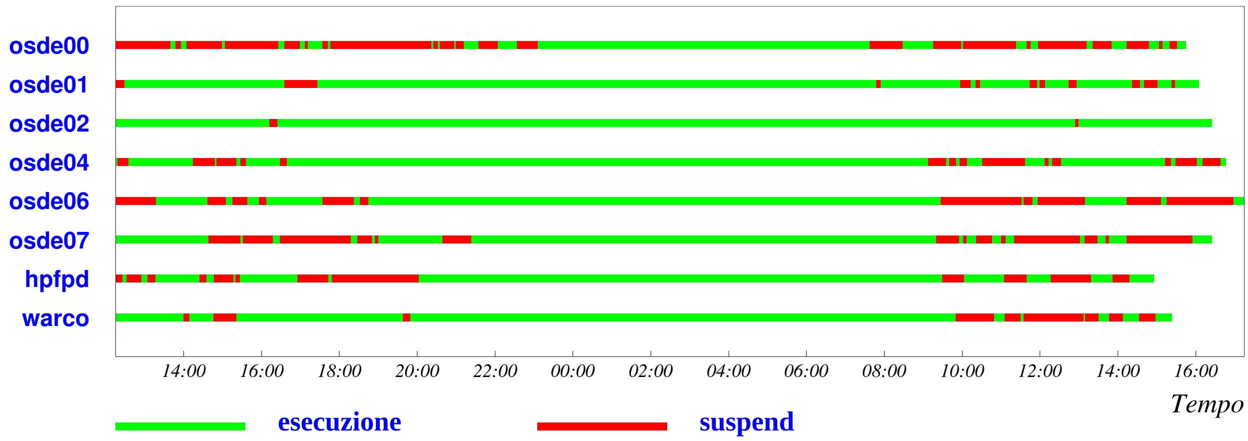

Figura 8.7: Esecuzione e suspend dei job DEMI

che nel test non si è interferito minimamente con le attività degli utenti, che hanno potuto usare senza problemi le varie workstation.

Questi risultati potrebbero essere ulteriormente migliorati con alcuni raffinamenti. Dal grafico si può notare come le elaborazioni sulle workstation dell'architettura Hp-Ux (*hpfpd* e *warco*) siano terminate prima rispetto alle elaborazioni sulle alpha station. Il problema è dovuto a come viene effettuato lo scheduling locale: il Local Scheduler (e in particolare il modulo che decide a che architetture assegnare i task da eseguire) viene "attivato" quando c'è un'elementary request che deve essere elaborata. Decidere a che architettura assegnare un job da eseguire, quando ci sono già centinaia o migliaia di altri job in coda, non è la scelta ottimale, in quanto non si può sapere come evolverà l'elaborazione: non si può prevedere quando e per quanto tempo i vari calcolatori delle diverse architetture saranno disponibili, o usati dagli utenti. Si potrebbe allora modificare questa parte del sistema DEMI, in modo da decidere a che architettura assegnare un task, solo quando c'è un calcolatore "libero", pronto per eseguire elaborazioni[1].

Può succedere, inoltre, che vi siano dei lunghi intervalli di tempo tra il termine delle elaborazioni dei job DEMI sui computer della stessa architettura. Questo problema è legato all'elaborazione dell'ultimo job su ogni computer: il processing può procedere ininterrottamente, oppure il job, una volta iniziata l'elaborazione, può restare in stato di 'suspend' per un lungo periodo. Si potrebbe ovviare a questo inconveniente utilizzando il meccanismo del checkpointing: se un job resta in stato di suspend per un lungo periodo, può essere migrato su un altro calcolatore disponibile. D'altro canto bisogna ricordare che salvare periodicamente su disco lo stato dei vari job può comportare un sensibile degrado delle prestazioni.

8.3 Altro esempio

È stata verificata la funzionalità del sistema considerando anche altre applicazioni.

In questo paragrafo si esamina un secondo test del prototipo, in cui si è definita appositamente una produzione più "articolata" rispetto a quella del test precedente, e in cui si sono utilizzati per l'esecuzione calcolatori di diverse architetture (Digital-Unix, Hp-Ux, Windows-NT). Anche se non si tratta di un'applicazione reale, si ritiene quindi questa prova molto significativa, vista la complessità dell'elaborazione e dell'ambiente di calcolo.

Si sono considerati tre tipi di task, rappresentati in figura 8.8, utilizzati poi per definire la produzione.

[1]Per verificare quando un computer è disponibile, si possono usare le apposite procedure del sistema di load sharing

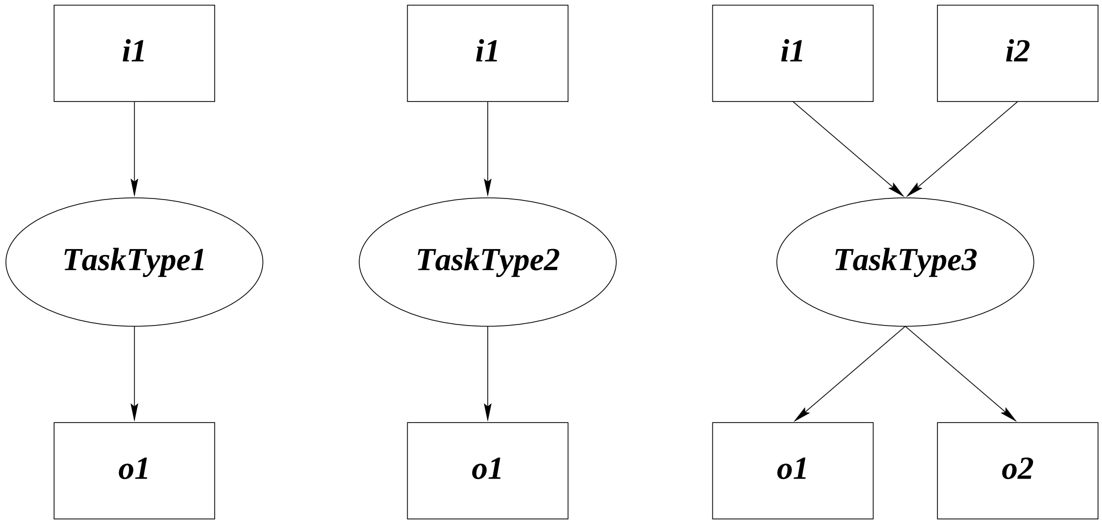

Figura 8.8: Tipi di task utilizzati nel secondo test

Il tipo di task *TaskType1* e il tipo di task *TaskType2* hanno un input e un output, mentre il tipo di task *TaskType3* ha due input e due output.

In questi task si legge il valore degli input, e si eseguono delle computazioni che fanno uso intenso di CPU. Infine vengono creati i corrispondenti file di output. Per quel che riguarda l'output *o1* del tipo di task *TaskType3*, non viene creato un nuovo file, ma viene fatto un append ad un file già esistente (il file viene creato solo la prima volta).

Approssimativamente il tempo necessario per l'esecuzione di un task di tipo *TaskType2* è circa un terzo rispetto al tempo richiesto per un task di tipo *TaskType1*, e metà rispetto ad un task di tipo *TaskType2*. Considerando ad esempio un'alpha station 3000/600, per eseguire un task di tipo *TaskType1*, un task di tipo *TaskType2*, e un task di tipo *TaskType3* sono necessari rispettivamente circa 18 minuti, 6 minuti e 12 minuti.

Per rendere più significativo il test, non si sono implementati tutti i tipi di task per tutte le architetture disponibili: il tipo di task *TaskType1* è stato realizzato per le architetture Hp-Ux e Windows-NT, il tipo di task *TaskType2* è stato implementato per le architetture Digital Unix, Hp-Ux e Windows-NT, mentre il tipo di task *TaskType3* è stato realizzato per le architetture Digital Unix e Hp-Ux.

Usando questi tre tipi di task, si è definita la produzione mostrata in figura 8.9, costituita da quattro task: *Task1* di tipo *TaskType1*, *Task2* di tipo *TaskType2*, *Task3* di tipo *TaskType3*, e *Task4* di tipo *TaskType2*.

Per quel che riguarda i data box, *Inp1* e *Inp3* sono input di tipo singleton. *Out2*, *Out4* e *Out7* sono output di tipo stream, mentre *Out5* è un output di tipo

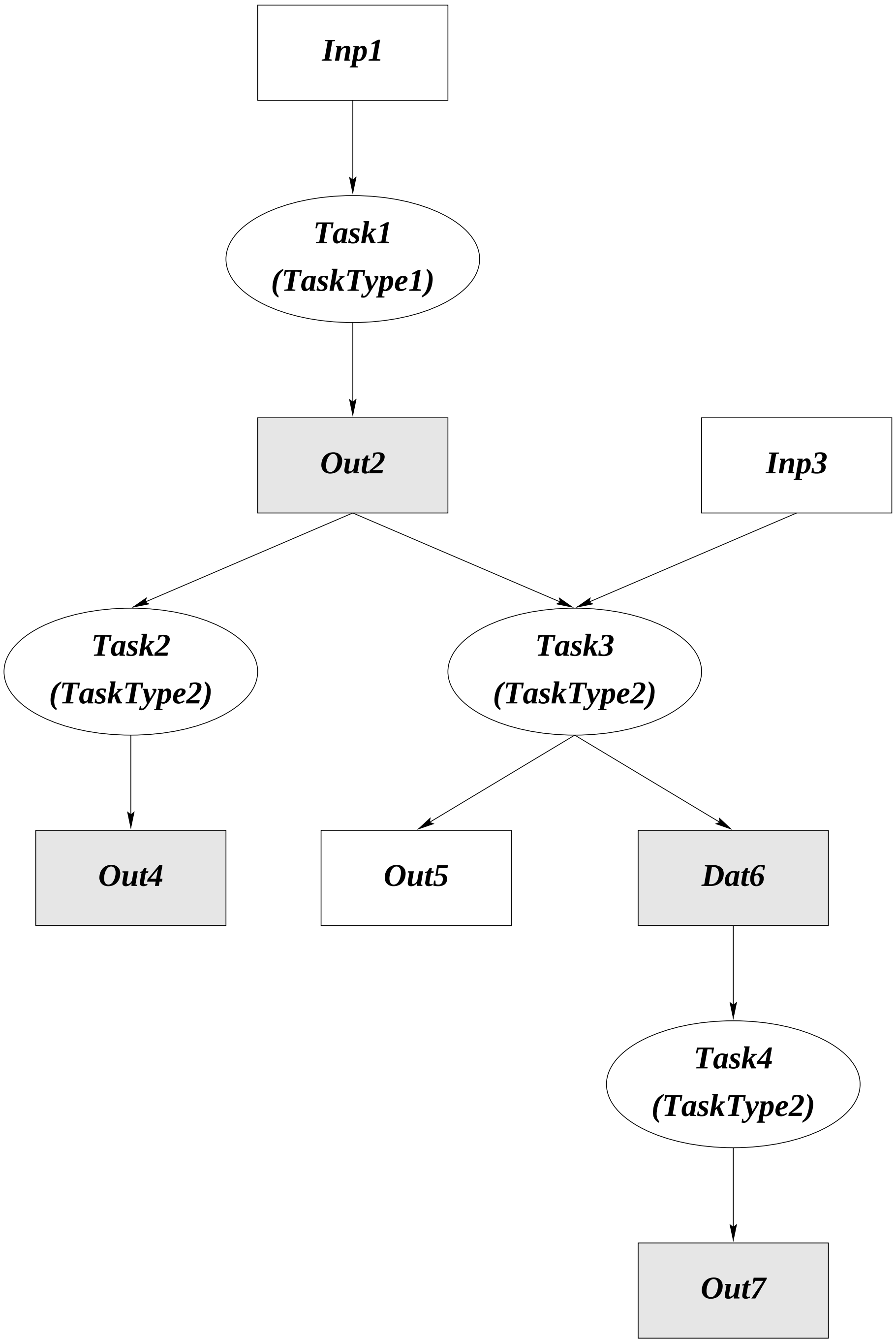

Figura 8.9: Produzione considerata nel secondo test

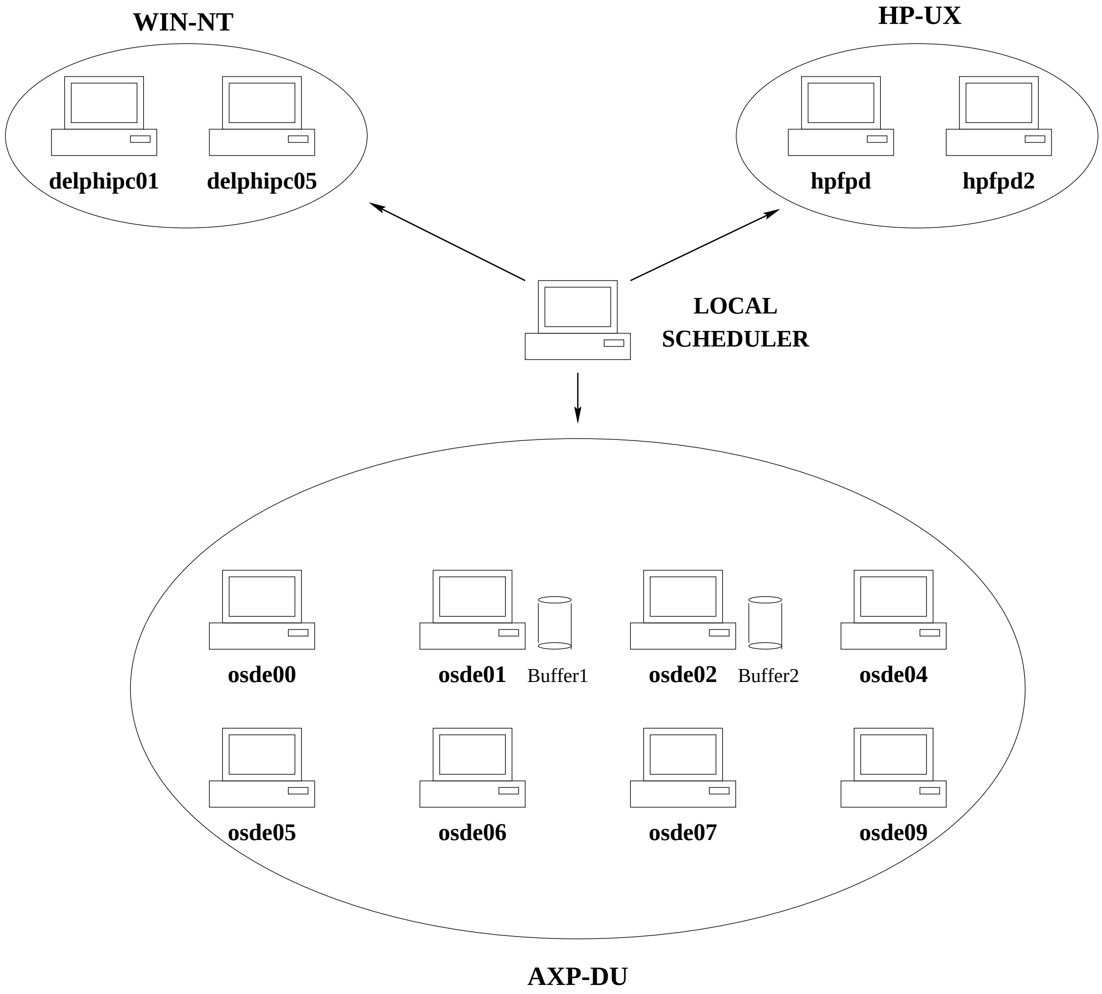

Figura 8.10: Ambiente considerato nel secondo test

singleton. Infine *Dat6*, di tipo stream, rappresenta dati temporanei (i relativi file vengono quindi cancellati dopo essere stati usati come input dal task *Task4*).

Considerando questa produzione, si è "sottomessa' una richiesta computazionale: si è specificato un numero di iterazioni pari a 300 (in totale, quindi, sono stati eseguiti 1200 task: 4 task per ogni iterazione).

Per quel che riguarda la dimensione massima dei singoli file associati ai vari data box della corrispondente produzione, questa è pari a 1 Mbyte per i file associati ai data box *Inp1*, *Inp3*, *Out2* e *Out5*, 2 Mbyte per i file associati ai data box *Out4* e *Out7*, e 4 Mbyte per i file associati al data box *Dat6*.

Per l'esecuzione si è considerato l'ambiente rappresentato in figura 8.10.

Si sono utilizzate in totale dodici workstation, di tre diverse architetture:

- otto alpha station con sistema operativo Digital Unix 4.0B (architettura *AXP-DU*), di cui:
  - un'alpha station 600 5/266: *osde00*;
  - cinque alpha station 3000/600: *osde01*, *osde05*, *osde06*, *osde07*, *osde09*;
  - un'alpha station 3000/400: *osde04*;
  - un'alpha station 3000/300: *osde02*;
- due workstation HP con sistema operativo HP-UX 10.20 (architettura *HP-UX*), di cui:
  - una workstation HP 9000/780: *hpfpd2*;
  - una workstation HP 9000/735: *hpfpd*;
- due PC Pentium Pro 200 con sistema operativo Windows NT 4 Workstation: *delphipc01* e *delphipc05* (architettura *WIN-NT*).

Si sono considerati due disk buffer, di capacità rispettivamente 1.1 Gbyte e 950 Mbyte, fisicamente installati su due alpha station. Questi disk buffer sono stati condivisi con le altre workstation Unix (sia Digital che HP) via NFS, mentre per i PC Windows NT si è utilizzato Samba.

Per quel che riguarda il load sharing system LSF, si è considerata la stessa configurazione del primo test.

L'elaborazione è iniziata alle ore 20:30 di un lunedì sera. L'ultimo task ha completato la sua esecuzione alle ore 12:45 del giorno successivo (quindi circa 16 ore dopo).

Non vengono analizzati in dettaglio i risultati dell'elaborazione (visto che non si tratta di un'applicazione reale, in questo test si era più interessati a valutare la funzionalità del sistema, considerando un'elaborazione e un ambiente di esecuzione complessi): si riporta solo, nelle figure 8.11, 8.12, 8.13, e 8.14, su che calcolatori sono stati eseguiti rispettivamente i task *Task1*, *Task2*, *Task3* e *Task4*.

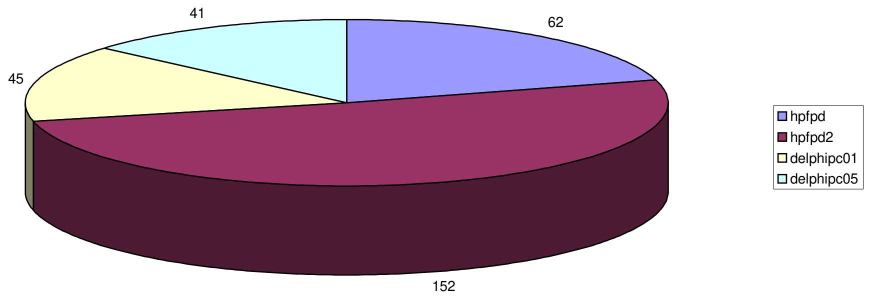

Figura 8.11: Task *Task1* eseguiti nei vari calcolatori

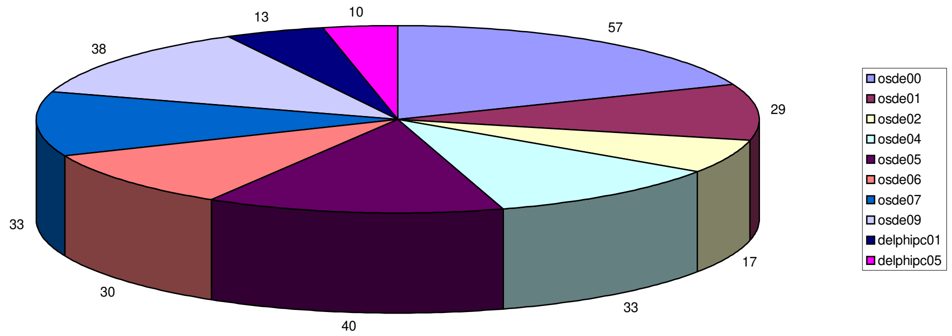

Figura 8.12: Task *Task2* eseguiti nei vari calcolatori

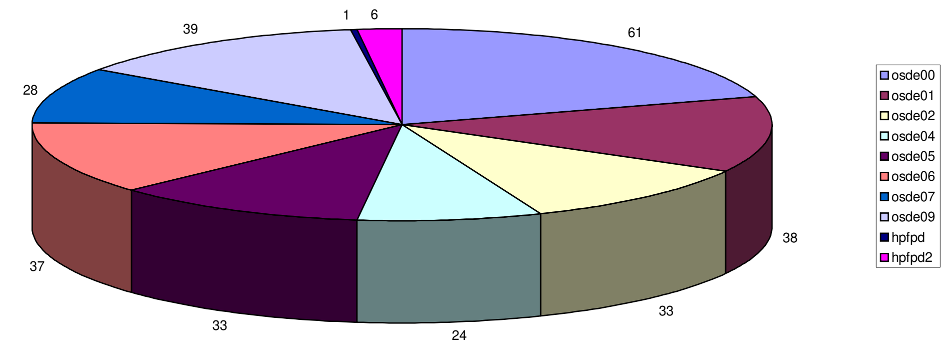

Figura 8.13: Task *Task3* eseguiti nei vari calcolatori

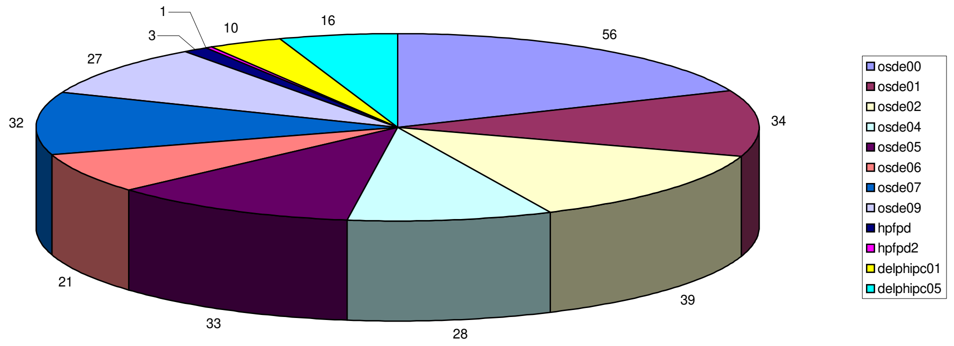

Figura 8.14: Task *Task4* eseguiti sui vari calcolatori

8.4 Conclusioni

È stato quindi verificato il corretto funzionamento del sistema, considerando diversi tipi di elaborazioni, sia reali, che costruite appositamente per esaminare casi più complessi. Negli esempi descritti si sono analizzate "singole" produzioni, ma si è verificata la funzionalità del sistema considerando anche diversi tipi di elaborazioni, sottomesse contemporaneamente al sistema.

Si sono considerati anche diversi ambienti di esecuzione, costituiti da calcolatori di architetture diverse. In particolare si è verificata la piena integrazione tra le workstation Unix, e i PC Windows NT.

Per quel che riguarda l'interoperabilità tra il sistema DEMI e il load sharing system LSF, non si sono riscontrati particolari problemi o anomalie. È probabilmente opportuna una maggiore "integrazione" tra i vari task definiti dagli utenti, e la configurazione del pool di workstation (ad esempio per definire vincoli del tipo: il task *taskx* può essere eseguito solamente sui calcolatori che hanno una memoria libera almeno pari a y, o per decidere se usare o meno il meccanismo del checkpointing, a seconda dei casi). Questi raffinamenti, comunque, comporterebbero piccole modifiche del codice, "isolate" a specifici moduli.

I risultati dei test sul prototipo del sistema DEMI sono quindi da considerarsi senz'altro positivi. Le verifiche sono state utili anche perché si sono potuti individuare gli elementi che, se raffinati, possono comportare un ulteriore miglioramento delle prestazioni complessive del sistema.