Capitolo 2

L'ambiente HEP

L'obiettivo di un generico esperimento di fisica delle alte energie è quello di scoprire quali sono le particelle costituenti la materia, e le forze che esistono tra esse. In questi studi è necessario elaborare un numero molto elevato di dati, in parte acquisiti dagli apparati sperimentali, e in parte generati attraverso opportuni programmi di simulazione.

In questo capitolo, dopo una breve introduzione sugli esperimenti HEP, si analizzano quali sono le caratteristiche e i requisiti per le elaborazioni che devono essere gestite. Oltre a considerare gli esperimenti attuali, si fa riferimento anche agli esperimenti della prossima generazione, in cui il volume dei dati da gestire sarà dell'ordine dei Pbyte: per questi esperimenti i problemi computazionali da gestire saranno quindi molto più complessi.

2.1 Gli esperimenti HEP

In un generico esperimento di fisica delle alte energie (*HEP: High Energy Physics*) vengono fatti collidere fasci di particelle con bersagli fissi, o con particelle di altri fasci. Studiando i frammenti che vengono prodotti in queste collisioni, è possibile scoprire quali sono i costituenti della materia, e le forze che operano tra questi elementi [20]. A tal fine si utilizzano appositi *detector*, costituiti da più "strati" tra i quali passano le particelle: attraverso questi apparati è possibile identificare tali particelle, ricostruire le loro tracce e calcolare i loro parametri fisici (masse, energie, momenti, ecc...). Ogni detector è in grado di fornire solo specifiche informazioni, e può identificare solo particolari tipi di particelle o interazioni: per questo motivo si devono considerare più tipi di detector [49].

Un grande esperimento HEP di solito coinvolge una collaborazione che consiste in centinaia o migliaia di fisici, appartenenti a università e istituti di tutto il mondo. Il detector e gli altri apparati sperimentali relativi ad un esperimento HEP sono installati in uno specifico laboratorio, in genere chiamato il *sito principale* dell'esperimento.

In tutti gli esperimenti di fisica delle alte energie l'unità di informazione è l'**evento**: fondamentalmente un evento è una "registrazione", in forma digitale, delle tracce lasciate dalle particelle dopo una collisione [49].

I vari eventi sono indipendenti tra loro: ogni evento rappresenta un'entità a sé stante, e non è in nessun modo correlato agli altri eventi. L'elaborazione che viene fatta su un evento è quindi del tutto indipendente dall'elaborazione che viene eseguita sugli altri eventi.

Poiché la ricerca, la verifica e lo studio delle varie teorie fisiche viene fatta utilizzando metodologie di tipo statistico, è fondamentale poter disporre di un numero molto elevato di eventi da poter elaborare. In un generico esperimento HEP dell'ultima generazione vengono generalmente acquisiti dagli apparati sperimentali ogni anno un numero di eventi che può variare tra 10^6 e 10^{12}. Ogni evento ha una dimensione che di solito varia tra 1 Kbyte e 1 Mbyte, a seconda del tipo di analisi fisica che deve essere fatta, e a seconda del tipo di detector utilizzato. Per quel che riguarda i requisiti in termini di risorse computazionali, per elaborare un evento è necessario un tempo di calcolo che può variare tra qualche millisecondo fino a centinaia di secondi, a seconda del tipo di analisi considerata e, ovviamente, a seconda del tipo di processore utilizzato.

Questi numeri relativi ai "volumi" di dati che devono essere considerati in un esperimento HEP, cresceranno approssimativamente per un ulteriore ordine di grandezza nei prossimi anni: in esperimenti come CDF, D0, LEP, HERA e le B factory a SLAC e a KEK si acquisiranno dagli apparati sperimentali ogni anno una quantità di dati dell'ordine delle decine o centinaia di Tbyte. Si stima che con i collider dell'esperimento LHC al Cern, a partire dalla metà del prossimo decennio, verranno acquisiti ogni anno dati dell'ordine dei Pbyte [11]. In tabella 2.1 vengono mostrati i volumi di dati che saranno acquisiti ogni anno per alcuni esperimenti HEP [56].

Per poter elaborare tutti questi dati è necessario poter disporre di una enorme potenza di calcolo. L'ambiente HEP "lavora" sempre ai limiti della tecnologia informatica: lo studio e il progetto di un esperimento di fisica delle alte energie deve necessariamente considerare quali sono i vincoli posti dalle risorse di calcolo disponibili.

È quindi chiaro come sia di fondamentale importanza poter gestire nel modo più efficiente possibile queste risorse computazionali.

Esperimento	Laboratorio	Inizio acquisizione dati	Raw Data da acquisire
BELLE	KEK Tsukuba (Giappone)	1998	30 Tbyte / anno
NA45	CERN Ginevra (Svizzera)	1999	30 - 50 Tbyte / anno
BaBar	SLAC San Francisco (USA)	1999	200 Tbyte / anno
Esperimenti RHIC (PHENIX, STAR)	BNL Long Island (USA)	1999	200 - 300 Tbyte / anno
COMPASS	CERN Ginevra (Svizzera)	1999	300 Tbyte / anno
D0	FNAL Chicago (USA)	2000	100 Tbyte / anno
CDF (RUN II)	FNAL Chicago (USA)	2000	250 Tbyte / anno
Esperimenti LHC (ATLAS, CMS, ALICE)	CERN Ginevra (Svizzera)	2005	5 Pbyte / anno

Tabella 2.1: Dati da acquisire per alcuni esperimenti HEP

2.2 Elaborazioni in HEP

Esistono molti tipi di elaborazioni che vengono fatte sugli eventi, nei vari esperimenti di fisica delle alte energie. Queste computazioni possono comunque essere divise in tre categorie principali [20]:

- **Simulazione** (o **Monte Carlo**)

  Per poter verificare le varie teorie fisiche, i ricercatori devono confrontare i *raw data* acquisiti dagli apparati sperimentali, con eventi simulati, generati attraverso specifici programmi. Di solito è necessario poter disporre di un numero di eventi simulati molto superiore al numero di eventi reali acquisiti (almeno il doppio). I dati reali (acquisiti dagli apparati sperimentali) e i dati simulati hanno la stessa struttura: in questo modo è poi possibile utilizzare gli stessi programmi per elaborare i due insiemi di dati.

  La simulazione viene anche usata per valutare l'efficienza del detector utilizzato.

- **Ricostruzione**

  In questa fase i raw data (reali e simulati) vengono elaborati, in modo da ottenere valori significativi dal punto di vista fisico. Ad esempio, per trovare la carica di una particella si ricostruisce la corrispondente traccia nel detector: analizzando poi la curvatura di questa traiettoria, dovuta alla presenza del campo magnetico, si determina appunto la carica. In questa fase si identificano anche le particelle generate nelle collisioni (una volta calcolati le corrispondenti cariche e masse).

- **Analisi**

  L'analisi può essere scomposta in varie fasi, nelle quali gli eventi reali ricostruiti e gli eventi simulati ricostruiti vengono confrontati tra loro, per verificare differenze e/o analogie.

Semplificando il problema, una tipica elaborazione può quindi essere schematizzata dal seguente pseudo-codice:

```
1  begin
2    if (necessary == true)
3      GenerateMonteCarloEvents();
4    end
5    while (necessary == true)
6          begin
7            MCRawSubset = GetMCRawSubSet();
8            RealRawSubset = GetRealRawSubSet();
9            MCReconSubset = Reconstruction(MCRawSubset);
10           RealReconSubset = Reconstruction(RealRawSubset);
11           MCOut = SpecificAnalysisProg(MCReconSubset);
12           RealOut = SpecificAnalysisProg(RealReconSubset);
13           Compare(MCOut, RealOut);
14         end
15   end
```

- Linee 2 - 4

  Se necessario (se non sono già disponibili) vengono generati eventi Monte Carlo, attraverso un opportuno programma di simulazione.

- Linee 7 - 8

  Dall'insieme costituito da tutti gli eventi reali, acquisiti dagli apparati sperimentali, si estrae un sottoinsieme di eventi: la scelta dipende dal tipo di problema fisico da considerare. Analoga operazione viene fatta per l'insieme dei raw data simulati.

- Linee 9 - 10

  Si effettua la ricostruzione (utilizzando lo stesso programma) dei due insiemi di eventi reali e simulati prima selezionati.

- Linee 11 - 12

  Utilizzando lo stesso programma di analisi, scritto per la "gestione" dello specifico problema considerato, si elaborano gli eventi "ricostruiti", reali e simulati, ottenuti nella precedente fase.

- Linea 13

  I risultati dell'analisi, sui dati reali e su quelli simulati (che solitamente sono sotto forma di distribuzioni statistiche), vengono confrontati tra loro, per verificare differenze e/o analogie.

Nella simulazione e nella ricostruzione i requisiti computazionali in termini di I/O sono modesti: l'elaborazione è "limitata" dalla CPU disponibile. Nelle fasi dell'analisi, invece, può essere necessaria anche un'elevata attività di I/O.

I programmi che vengono utilizzati per la simulazione, la ricostruzione e l'analisi sono specifici ai tipi di detector considerati, e possono essere costituiti anche da milioni di righe di codice. Alcuni di questi programmi possono essere utilizzati per molti anni; altri, invece, devono essere frequentemente modificati, quando vi è una qualsiasi modifica nei relativi detector.

In tutti questi programmi si fa uso di librerie standard di procedure, utilizzate in tutti gli esperimenti HEP (ad esempio le *librerie CERNlib*) [49]. Anche l'I/O è gestito attraverso routine standard: in questo modo non è necessaria nessuna conversione esplicita quando i dati sono prodotti da un computer, e devono poi essere utilizzati da un altro calcolatore di architettura diversa.

Quando è disponibile una nuova versione di un programma di ricostruzione/analisi, è spesso necessario usarlo per elaborare dati che erano stati già processati da una precedente release del programma. Questa procedura è chiamata *reprocessing*.

Poiché, come detto, gli eventi sono indipendenti tra loro, è possibile elaborarli in maniera autonoma e asincrona [48]. È quindi possibile "parallelizzare" le attività di simulazione, ricostruzione e analisi, utilizzando più processori. L'insieme totale degli eventi da elaborare può essere partizionato in un numero di insiemi pari al numero dei processori disponibili, e il programma che esegue l'elaborazione considerata viene "replicato" su ognuno di questi processori. In questo modo ogni processore esegue l'elaborazione considerata su un proprio insieme di dati. In questa forma di *data parallelism*, il programma considerato non è un programma parallelo: semplicemente viene eseguito in parallelo su più processori [20]. In figura 2.1 è mostrato un esempio di questo parallelismo sui dati: l'insieme dei dati di input viene partizionato in due insiemi, e due processori (*Host 1* e *Host 2*) eseguono lo stesso programma, ognuno su uno di questi due insiemi di dati.

Oltre al data parallelism, in alcuni casi si può considerare anche un'altra forma di parallelismo: il *parallelismo temporale*. In questi casi la computazione viene "scomposta" in più task, ognuno dei quali viene assegnato ad uno specifico processore [53]. In figura 2.2 è mostrato un esempio di parallelismo temporale: la computazione è composta di due task (*Task 1* e *Task 2*), che vengono associati a due processori (*Host 1* e *Host 2*).

Per questa seconda forma di parallelismo si considera una granularità di tipo *coarse-grain*, in quanto le componenti che vengono eseguite in parallelo sono applicazioni "complete", e non singole istruzioni di uno stesso programma [49].

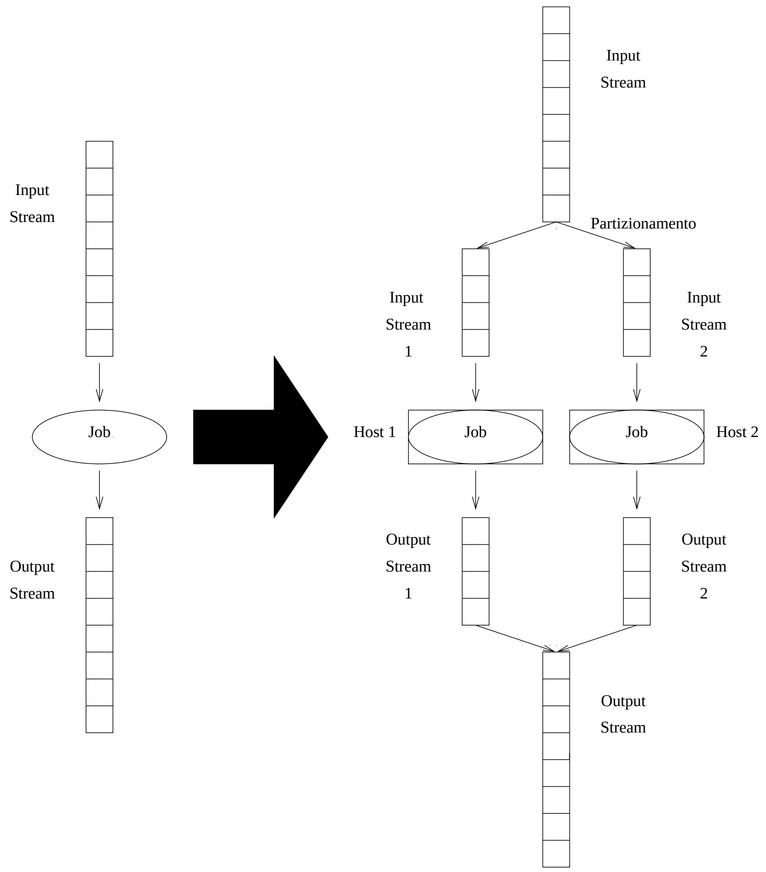

Figura 2.1: Data parallelism, usando due processori

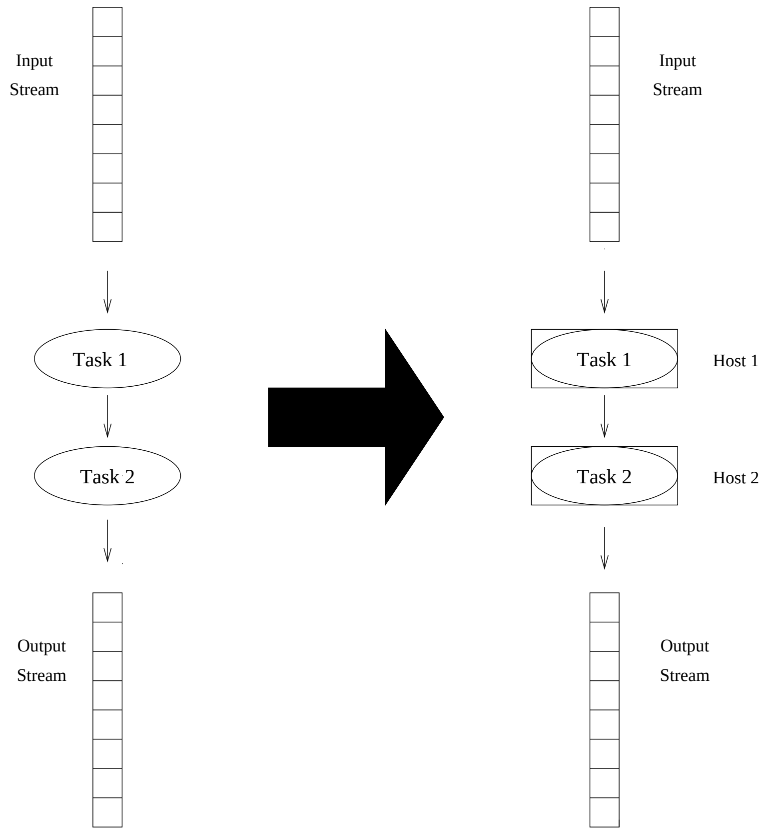

Figura 2.2: Parallelismo temporale, usando due processori

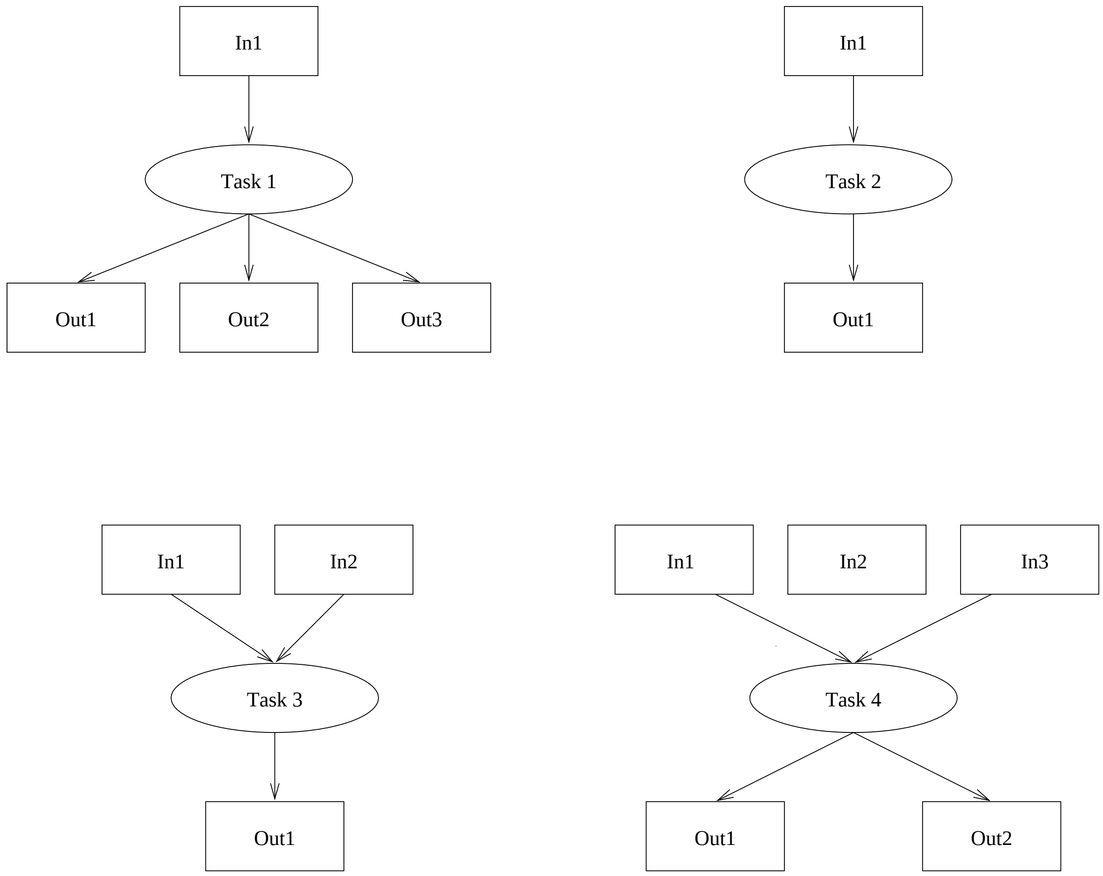

Figura 2.3: Esempi di rappresentazioni grafiche di task

2.2.1 Task e Production

Di solito un'elaborazione fisica non consiste di un singolo programma elementare, con un input e un output, ma può avere una struttura più complessa: può essere composto da più elementi computazionali elementari, che chiamiamo **task**. Ogni task rappresenta un'unità di elaborazione, responsabile di "gestire" un singolo aspetto del problema complessivo (simulazione, ricostruzione, una singola fase dell'analisi, ecc...). Un task può avere zero o più input, e zero o più output.

Graficamente si può rappresentare un task con un'ellisse, e i suoi input-output con rettangoli, come rappresentato negli esempi di figura 2.3.

I task vengono "combinati" tra loro, costituendo quella che viene chiamata **production**: una production gestisce quindi un job "completo" (ad esempio la generazione di eventi simulati, la loro ricostruzione, la loro analisi, costituita da diverse fasi). Per descrivere una production bisogna quindi specificare i task che la compongono, e le relazioni che esistono tra essi.

La struttura logica di una production può anche essere molto complessa. Ad esempio la production rappresentata in figura 2.4 è composta dai quattro task (*Task1*, *Task2*, *Task3* e *Task4*) definiti in figura 2.3.

L'input del task *Task 1* è anche l'input (*Input 1*) della production. Il primo output del task *Task 1* coincide con l'input del task *Task 2* (*Dat 2*). Gli altri due output del task *Task 1* sono input del task *Task 3* (*Dat 3* e *Dat 4*). Il secondo output del task *Task 1* è anche input del task *Task 4*. Gli output dei task *Task 2* e *Task 3* sono usati come input dal task *Task 4* (*Dat 5* e *Dat 6*). Gli output del task *Task 4* sono anche output della production (*Output 1* e *Output 2*).

I dati di input possono essere classificati in due categorie: gli *event data* e i *long term data*. Gli event data rappresentano sequenze di eventi indipendenti, che devono essere elaborati uno alla volta. I long term data, invece, sono singoli file, che rappresentano "costanti" che devono essere usate nell'elaborazione di tutti gli eventi che costituiscono gli event data. Ad esempio le *costanti di calibrazione* sono input di tipo long term.

Ogni task può avere input sia di tipo long term, che di tipo event, e produce output di tipo event [53]. Ogni task può avere zero o al più un input di tipo event, e zero o più input di tipo long term: un task non può avere due o più input di tipo event. Considerando ad esempio un task per la ricostruzione, vi sarà un input di tipo event (per la sequenza di eventi che devono essere elaborati, uno alla volta), e uno o più input di tipo long term, rappresentanti costanti da usare per l'elaborazione di tutti gli eventi.

In figura 2.5 è rappresentato un altro esempio, meno complesso, di production. In questo caso la production è costituita da due task: *Task X* e *Task Y*. Il task *Task X* non ha input e ha un output (*Output 1*), che è anche un output della production. Il task *Task Y* ha un input (*Output 1*) che è output del task *Task 1*, e un output (*Output 2*), anch'esso un output della production.

In questo caso, quindi, il risultato intermedio *Output 1* deve essere considerato un output della production (ad esempio deve essere memorizzato su nastro), e non può essere cancellato dopo essere stato usato come input dal task *Task X*. Si considera questo approccio quando è necessario gestire i reprocessing: in questi casi gli output dei task intermedi devono essere "salvati" su supporti di memoria secondaria, per poter poi eventualmente elaborare nuovamente questi dati in futuro, utilizzando nuove versioni di programmi di analisi.

Prima di eseguire un generico task di simulazione/ricostruzione/analisi, sono in genere necessarie alcune operazioni preliminari (ad esempio il load di un database di calibrazioni): queste operazioni possono richiedere per la loro esecuzione un tempo non trascurabile. Non è quindi conveniente elaborare un evento alla volta, in

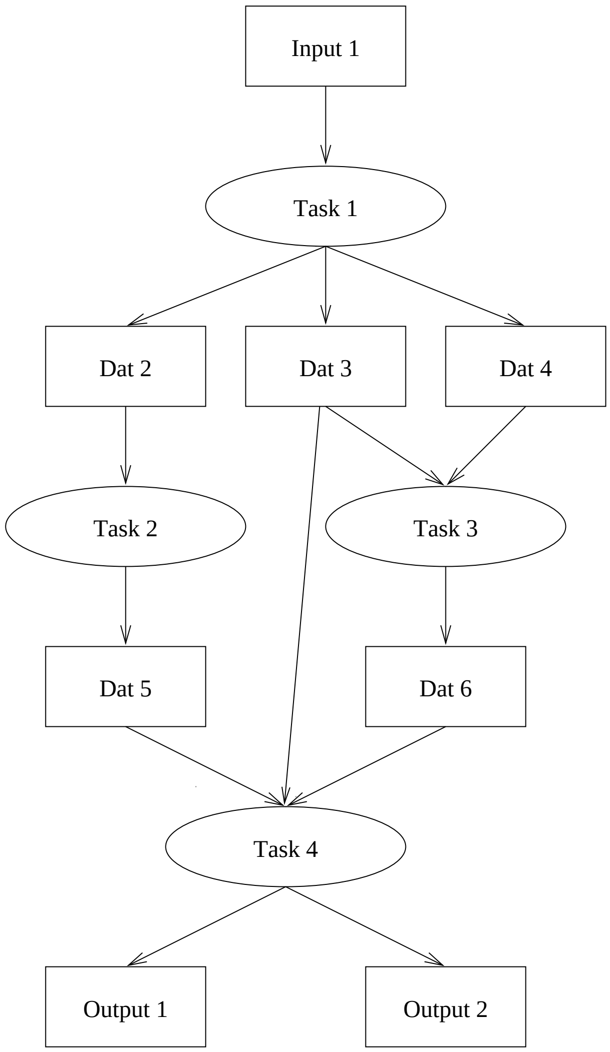

Figura 2.4: Esempio di production

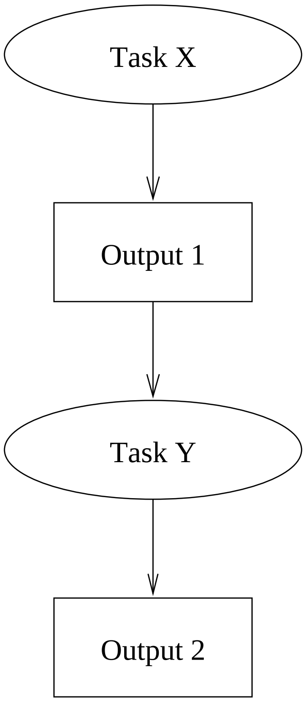

Figura 2.5: Un altro esempio di production

quanto queste operazioni di "setup" dovrebbero essere eseguite tante volte quanti sono gli eventi da elaborare. Per migliorare le performance, gli eventi vengono organizzati in *run*, ognuno costituito da qualche centinaia o migliaia di eventi. In questo modo le operazioni preliminari devono essere eseguite solo per ogni run, e non per ogni evento: questo permette di ridurre notevolmente i tempi totali richiesti per l'elaborazione.

Nel problema considerato si può supporre che la "partizione" dell'insieme di eventi in run sia già stata eseguita: quindi per ogni input di tipo event viene specificata una sequenza di file, ognuno dei quali rappresenta un run.

2.2.2 Architetture usate nell'ambiente HEP

Anche l'ambiente della fisica delle alte energie è una di quella realtà in cui è cambiato il modello di computing: si è verificata la "migrazione" da singoli computer mainframe, a cluster di workstation. Le workstation, infatti, si sono dimostrate adeguate per le elaborazioni tipiche che si devono considerare, con un miglior rapporto prestazioni/prezzo rispetto ai computer di tipo mainframe [7] [20].

Fino a qualche anno fa in quasi tutti gli esperimenti di fisica delle alte energie si è utilizzato un modello standard di workstation, per le attività di simulazione, ricostruzione e analisi: l'architettura Digital-VMS. Si è poi ritenuto più conveniente l'utilizzo di sistemi operativi più "aperti", rispetto al sistema proprietario VMS, in modo da poter utilizzare un numero superiore di architetture hardware. Per questa ragione è stato scelto il sistema operativo Unix, considerando le sue varie implementazioni (Digital-Unix, Hp-Ux, Sun-Solaris, Ibm-Aix, ecc...).

Probabilmente le workstation Unix saranno usate ancora per un certo periodo di tempo, ma i PC, che offrono prestazioni elevate e ad un prezzo molto competitivo, saranno quasi sicuramente la piattaforma di riferimento per il prossimo futuro. È stato provato che sia il sistema operativo Linux, che il sistema operativo Windows NT sono adeguati per eseguire programmi di simulazione/ricostruzione/analisi fisica [10].

2.3 Organizzazione

I fisici che collaborano ad un certo esperimento appartengono a università e istituti di tutto il mondo; il sito principale dell'esperimento e questi altri istituti sono connessi tra loro attraverso una rete geografica (WAN: Wide Area Network): di solito questa non è una rete privata, ma è una rete che deve essere condivisa con altre organizzazioni.

Tutti i dati reali e i dati simulati sono di solito memorizzati su supporti di memoria di massa presenti nel sito principale, disponibili all'intera collaborazione. Vista la grande quantità di eventi da gestire, è possibile memorizzare su dischi solo un sottoinsieme di questi dati, mentre per gli altri si utilizzano supporti di tipo *off-line* (in genere nastri).

Negli altri istituti possono essere presenti sottoinsiemi di questi dati, usati da quei fisici per le loro specifiche analisi. Questi dati vengono copiati su nastri dai dispositivi di storage del sito principale, e vengono poi inviati all'istituto remoto: solo per piccoli volumi si utilizza la Wide Area Network per queste copie.

L'altro possibile approccio per i ricercatori è quello di sottomettere i job di analisi sui calcolatori presenti nel sito principale dell'esperimento, senza quindi la necessità di avere repliche locali dei dati.

Di solito i singoli ricercatori (o gruppi di ricercatori) studiano solo un particolare aspetto del problema fisico generale considerato: a tal fine essi scrivono appositi programmi di analisi, che vengono usati per elaborare specifici sottoinsiemi dei dati disponibili. Tuttavia alcune elaborazioni non vengono "gestite" direttamente dai singoli fisici, ma vengono invece affidate ad una *struttura centralizzata*, che riceve le richieste dagli utenti, "schedula" e sottomette i relativi job sui calcolatori disponibili. Questo approccio è usato in genere per quelle elaborazioni che coinvolgono grandi volumi di dati, per cui è necessaria una grande quantità di risorse di calcolo. In questi casi è infatti fondamentale ottimizzare l'uso delle risorse computazionali, disponibili nel sito principale e/o nei siti remoti. Attraverso una struttura centralizzata di questo tipo viene inoltre ridotto il *manpower* richiesto per la gestione dell'intero sistema [7] [19].

Questa struttura viene usata ad esempio per la produzione di eventi Monte Carlo, ma può essere utilizzata anche per altri tipi di elaborazioni.

2.4 Un esempio: l'esperimento DELPHI

DELPHI (DEtector with Lepton, Photon and Hadron Identification) [3] è uno degli esperimenti LEP.

Il collider LEP, che con una circonferenza di 27 Km è il più grande acceleratore che sia mai stato costruito, è operativo al CERN (Ginevra, Svizzera) dall'agosto 1989. In questo collider gli elettroni e le loro anti-particelle, i positroni, vengono accelerati in direzioni opposte in un condotto in cui è stato creato il vuoto, attorno al quale vi è un anello di magneti. Gli elettroni e i positroni vengono fatti collidere tra loro: viene rilasciata un'energia in un piccolo volume con una densità paragonabile a quella che si presume fosse presente nell'Universo una frazione di secondo dopo il Big Bang. C'è quindi una "sintesi" tra il "molto grande" (l'Universo) e il "molto piccolo" (le particelle elementari).

L'energia iniziale del collider LEP è stata inizialmente posta a circa 91 GeV, in modo che in queste collisioni venissero prodotte particelle Z^0. Le particelle Z^0, e le particelle W^{+-}, entrambe scoperte al CERN nel 1983, sono responsabili delle cosiddette "forze elettrodeboli", responsabili della sopravvivenza del nostro sole. Studiando la creazione e il decadimento delle particelle Z^0, si è studiato e si sta ancora studiando l'attuale teoria delle particelle fondamentali, chiamata *modello standard*. La particella Z^0 ha una vita molto breve: la sua esistenza può essere "dedotta" solo dalla presenza dei frammenti della sua disintegrazione, che possono variare in numero da due fino a un centinaio.

Dalla fine del 1995, è iniziata la seconda fase del collider LEP. L'energia viene aumentata per:

- studiare la produzione delle coppie Z^0Z^0 e delle coppie W^+W^-, per nuovi studi e test del modello standard;

- ricercare nuove particelle, in particolar modo il bosone di Higgs, e/o particelle supersimmetriche.

Si sta procedendo in maniera progressiva:

- nel Novembre 1995 l'energia è stata portata a 130 - 136 GeV;
- nel Luglio - Agosto 1996 l'energia è stata posta a 161 GeV;
- nell'Ottobre - Novembre 1996 a 172 GeV;
- nel 1997 a 183 GeV;
- nel 1998 si sta operando ad un'energia pari a 189 GeV.

Nell'acceleratore LEP i fasci di elettroni e di positroni vengono fatti collidere varie volte. Esistono quattro regioni di intersezione, dove sono posti detector di particelle (ALEPH, DELPHI, L3, OPAL), che misurano le proprietà delle particelle secondarie prodotte nella collisione. Ognuno di questi detector è stato ottimizzato in modo da studiare specifici aspetti e fenomeni fisici.

DELPHI è uno di questi detector. Ha un'alta precisione e un'elevata "granularità", e permette di differenziare le varie particelle secondarie cariche. Attraverso un apposito detector di silicio è possibile individuare anche particelle dalla vita molto breve.

Per la progettazione e la costruzione del detector DELPHI sono stati necessari sette anni.

Attualmente la collaborazione dell'esperimento DELPHI consiste di circa 550 fisici, provenienti da 56 università e istituti di 22 stati.

2.4.1 Elaborazioni in DELPHI

DELSIM (DELphi SIMulation) è il programma standard di simulazione usato nell'esperimento DELPHI. L'unico input di DELSIM è costituito da un *file di titoli* (un input di tipo long term), che permette di selezionare il tipo di eventi da produrre, l'energia, e altri parametri. Si può quindi utilizzare lo stesso programma eseguibile DELSIM per produrre eventi simulati di tutti i tipi: è solo necessario modificare in maniera opportuna il file di titoli. Tuttavia, in alcuni casi particolari (per produzioni speciali) è necessario modificare anche il codice DELSIM, usando quindi un programma non standard. Di solito ogni anno viene rilasciata una nuova versione del programma DELSIM.

DELANA (DELphi ANAlysis) è il programma standard di ricostruzione dell'esperimento DELPHI. DELANA elabora i raw data reali (acquisiti dagli apparati sperimentali) o i raw data simulati (generati dal programma DELSIM), producendo

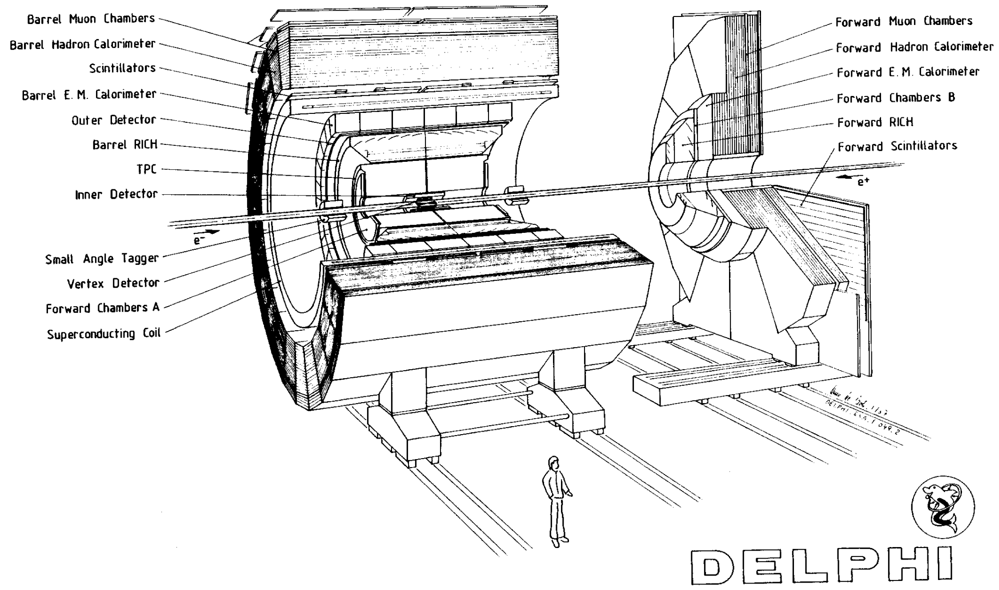

Figura 2.6: Il detector DELPHI

file di tipo *DST* (*Data Summary Tape*). Tutti gli eventi, reali e simulati, di tutti i tipi, vengono analizzati utilizzando lo stesso programma eseguibile DELANA. Di solito ogni anno vengono rilasciate una o più nuove versioni del programma DELANA.

Esistono poi altri programmi standard di analisi, come il programma *SHORT-DST* (l'input di questo programma è costituito da event data di tipo DST, e l'output è costituito da event data di tipo SHORTDST), il programma *MINIDST* (che elabora event data di tipo SHORTDST, producendo event data di tipo MINIDST), ecc...

Si devono considerare poi anche altri programmi "non standard", utilizzati per specifiche analisi. Di solito questi job di analisi sono gestiti direttamente dai singoli fisici, che ne scrivono il codice, e li sottomettono poi sulle varie workstation per la loro esecuzione.

Nell'esperimento DELPHI la produzione degli eventi simulati (usando DELSIM) e la produzione dei corrispondenti file DST (usando DELANA) viene fatta in diversi istituti che fanno parte della collaborazione: Lione, Milano, Padova, Rutherford, Saclay, Stoccolma, ecc... I file DST prodotti in questi istituti vengono memorizzati su nastri, che vengono poi inviati al CERN, il sito principale dell'esperimento.

Al CERN è presente una *farm*, composta da decine di workstation, che viene usata per la ricostruzione e l'analisi dei dati reali. Questa farm viene utilizzata per la produzione di eventi Monte Carlo solamente se non ci sono richieste per l'analisi, o se in quel momento la simulazione è considerata un'attività a più alta priorità.

Nell'esperimento DELPHI non esiste una vera e propria "struttura centralizzata" che gestisce l'esecuzione di tutti questi job: in ogni istituto vi sono una o più persone che si occupano della gestione delle attività di simulazione e ricostruzione, usando le risorse computazionali disponibili localmente. Non esiste però un modello standard a cui fare riferimento: in ogni istituto problemi quali il mapping dei task ai processori, la gestione delle risorse, ecc... vengono gestiti in maniera indipendente e autonoma.

2.5 Esperimenti della prossima generazione

Negli esperimenti HEP della prossima generazione (primi fra tutti gli esperimenti LHC al CERN) si considererà un modello di computing molto diverso rispetto a quello usato negli attuali esperimenti.

Ad esempio, non necessariamente i dati dovranno essere elaborati dove essi sono fisicamente presenti. Infatti un problema che si verifica attualmente è l'*overload* dei calcolatori presenti nel sito principale (dove sono presenti tutti i dati, reali e simulati). Se negli esperimenti della prossima generazione si continuasse ad utilizzare questo approccio, il problema sarebbe ancora più grave, visto che il volume dei dati da elaborare sarà molto più elevato. È più conveniente sfruttare anche le risorse di calcolo presenti nei vari siti, "spostando" i dati se necessario, considerando anche che reti geografiche veloci ed affidabili (come le reti ATM a 622 Mbps) saranno presto disponibili.

Sarà quindi necessario implementare un sistema che, a seconda dei casi, dovrà valutare dove è più conveniente eseguire il job sottomesso dall'utente. Un elemento di questo sistema sarà quindi un *data mover*, che dovrà gestire, in maniera trasparente e dinamica, la copia dei dati tra i vari siti, decidendo le strategie migliori: quando eseguire la copia, che sistema di copia utilizzare (la rete geografica, o la spedizione di nastri attraverso corriere), ecc...

Gli utenti non dovranno più preoccuparsi di conoscere dove i dati sono fisicamente memorizzati: accederanno ad essi usando semplicemente "nomi logici", anziché "nomi fisici" (i pathname dei file).

La "struttura centralizzata" probabilmente non esisterà più: i singoli utenti (o gruppi di utenti autorizzati) potranno definire proprie production, e sottomettere direttamente richieste computazionali al sistema.