Università degli Studi di Roma "La Sapienza"

Facoltà di Ingegneria

Tesi di Laurea in

Ingegneria Telecomunicazioni

Dicembre, 2004

Piattaforme software distribuite per il recupero di hardware obsolescente

Ruggero Russo

Università degli Studi di Roma "La Sapienza"

Facoltà di Ingegneria

Tesi di Laurea in Ingegneria Telecomunicazioni

Sessione Autunnale – Dicembre, 2004

Piattaforme software distribuite per il recupero di hardware obsolescente

Ruggero Russo

Relatore Prof. Roberto Baldoni

.....

Co-Relatore Ing. Domenico Davide Lamanna

.....

Indirizzo dell'Autore:
Ruggero Russo
Via Giovanni Caselli 67
Roma 00149
ITALIA
e-mail: rughero1@virgilio.it
www:

Non c'é progresso se non é per tutti.

Indice

Introduzione

Il presente progetto di tesi nasce da una proposta di Ingegneria Senza Frontiere che riguarda tre tematiche fondamentali: il riutilizzo di hardware datato (disciplina che verrà indicata nelle pagine che seguono col nome di Trashware); l'uso di tecnologie appropriate e sostenibili in progetti con Paesi in via di sviluppo (Cooperazione allo sviluppo) e la condivisione di risorse computazionali in un sistema distribuito (Cluster di macchine).

I problemi connessi al Trashware, ai Cluster e alla Cooperazione sono stati interrelati con lo scopo di studiare e discutere delle tematiche che uniscono aspetti tecnologici, sociali, economici ed anche ecologici. Le conclusioni cui si è perventuti sono di interesse tanto per le finalità sociali di Ingegneria Senza Frontiere (ISF), quanto per la ricerca accademica, nell'ambito del calcolo distribuito e dell'efficienza delle varie configurazioni di batterie di calcolatori.

Nel capitolo 1 si descrive l'impatto sociale ed economico che le nuove tecnologie hanno sulla vita delle persone, portando dei cambiamenti radicali nel modo in cui pensano, comunicano, agiscono, lavorano ecc. Tanto grandi sono le possibilità di sviluppo che le tecnologie dell'informazione e della comunicazione (ICT) hanno introdotto negli ultimi anni, che ormai non si può che parlare di Rivoluzione Digitale. Proprio l'enorme importanza che tali tecnologie rivestono per lo sviluppo della società, rende ancora più stridente il divario tra paesi ricchi e paesi poveri, cioè tra chi ha la possibilità di accompagnare quello sviluppo traendone i massimi benefici, e chi invece, a causa di una arretratezza economica, culturale ed infrastrutturale, rimane tagliato fuori. In questo contesto, il superamento del cosiddetto divario digitale appare come uno dei problemi più urgenti da affron-

tare per la comunità internazionale. Vengono presentate statistiche ed indicatori che mostrano come quello scarto tra i paesi più sviluppati e quelli che invece versano in condizioni di povertà, sta aumentando di anno in anno, aggravando una situazione dicotomica, per la quale nei Paesi ricchi si assiste al radicamento di un atteggiamento generale di "consumismo" anche a livello informatico, mentre i Paesi in via di sviluppo sono sempre più esclusi dalla fruizione delle nuove tecnologie.

Nei Paesi più sviluppati, la corsa al materiale informatico più nuovo e dalle elevate prestazioni è alimentata da una sorta di duopolio costituito dai produttori di hardware e di software, che impongono un'obsolescenza programmata ai dispositivi e ai programmi informatici. Questa tendenza, che non sembra destinata a rallentare, ha, tra le altre cose, un impatto molto grande sull'ambiente. Più di 150 milioni di computer sono dismessi ogni anno nel mondo; essi contengono sostanze altamente inquinanti, tra le quali piombo, mercurio, cadmio e cromo esavalente e costituiscono, perciò, una minaccia di inquinamento senza precedenti. L'Unione Europea ha varato direttive sul riciclaggio di tali materiali, ma di fatto nessuno degli stati dell'Unione si è adeguato. Inoltre, la difficoltà nello smaltimento delle componenti elettroniche e i suoi costi elevati scoraggiano l'iniziativa privata. Nel capitolo 1, si presentano nel dettaglio anche le statistiche di quella che è riconosciuta come una emergenza ambientale, presentando le quantità di materiale dismesso e le cifre sull'inquinamento che lo smaltimento dei rifiuti elettronici produce.

Lo studio del problema ecologico ed il tentativo di trovare una soluzione ad una questione che è quanto mai attuale e che, come detto, ha assunto la connotazione di una vera e propria emergenza, ha portato ISF a proporre la strada del riutilizzo del materiale dismesso per ammortizzare i tempi di obsolescenza dei computer (ormai inferiori ai 2-3 anni). Più in particolare, ISF ha sviluppato un progetto di recupero e riutilizzo di materiale informatico obsolescente da destinare ad associazioni che ne facciano richiesta o da utilizzare in progetti di cooperazione allo sviluppo, come quello che ha visto l'associazione impegnata ad allestire un laboratorio di informatica in Kosovo (capitolo 2). Oltre all'attività di

cooperazione internazionale, ISF ha avviato un progetto di Trashware in Italia, che ha portato alla formazione di un gruppo di lavoro che inizialmente si è raccolto intorno a questo lavoro di tesi, contribuendo in modo determinante al suo sviluppo, attraverso la condivisione e la trasmissione di competenze informatiche, hardware e software.

In molti iniziano ad interessarsi al riutilizzo di hardware datato. Negli ultimi anni in Italia il numero di associazioni o di semplici appassionati che si dedicano all'attività di Trashware è aumentato incredibilmente; le motivazioni che muovono i gruppi che si occupano di ricondizionare computer dismessi sono sostanzialmente legate alla questione ecologica di cui si è già accennato e soprattutto al corretto utilizzo dei dispositivi hardware.

La crescita di questo movimento non può essere pensata separata dalla crescita di un altro fenomeno incredibilmente vitale: quello del software Libero e Open Source. Nel Capitolo 2 si descrivono nel dettaglio in cosa consiste l'attività di Trashware, e le motivazioni profonde di quanti si occupano di questa attività. Viene dedicato ampio spazio alla descrizione del movimento OpenSource e Free Software, cui il Trashware è inscindibilmente legato per diversi motivi che nello sviluppo della tesi verranno trattati nel dettaglio e che qui verranno accennati. Il primo motivo è sicuramente di carattere economico; tra le prime considerazioni di chi deve fare i conti con la sostenibilità di un intervento dalla disponibilità economica esigua, rientra sicuramente la necessità di svincolarsi dalle costose licenze d'uso; il secondo motivo è legato al fatto che per conseguire uno sviluppo sostenibile autonomo è necessario il raggiungimento dell'indipendenza da attori stranieri, che può essere garantita esclusivamente dalla soluzione OpenSource; l'ultima motivazione è di carattere tecnico: solo il software Open Source può garantire massima adattabilità e la capacità di ottimizzare le prestazioni dell'hardware obsoleto che viene messo a disposizione in tutti quei progetti di sviluppo e cooperazione che hanno come scopo la riduzione di quel Divario Digitale di si è parlato sopra.

Rispetto ad un lavoro di Trashware tradizionale (che consiste nel ricondizionare le macchine obsolete ottimizzandone le prestazioni), questo lavoro trae la

sua originalità dall'affrontare tale problema attraverso la speculazione accademica riguardante la condivisione di risorse computazionali (Capitolo 3). Quello che solitamente è visto come una sfida per l'implementazione di risorse di supercalcolo per scopi scientifici, è stato calato in applicazioni del mondo reale. Questo aspetto originale del lavoro di tesi è conseguenza diretta di una delle peculiarità di ISF, che nasce all'interno dell'università e in essa opera organizzando seminari e conferenze che hanno come obiettivo la promozione di uno sviluppo tecnologico che sia realmente sostenibile ed accessibile a tutti.

In una fase propedeutica alla stesura di questo lavoro, è stato condotto uno studio sullo stato di sviluppo dei sistemi di condivisione di risorse computazionali, sia in ambiente accademico sia per quanto concerne i progetti di riuso e di cooperazione tecnica nell'ambito delle tecnologie per l'informazione e la comunicazione.

L'attenzione è stata focalizzata su due piattaforme software distribuite: OpenMosix ed LTSP. Nel capitolo 3, si mostra come queste hanno delle caratteristiche diametralmente diverse per i contesti in cui sono nate e si sono sviluppate, e per le finalità che ciascuno dei due sistemi persegue. OpenMosix (su cui maggiormente si è concentrata la nostra attenzione) è un sistema sviluppato in ambiente accademico e nasce appositamente per eseguire calcoli ad alte prestazioni ed efficienza. Perciò si presta molto bene ad essere utilizzato in tutte quelle applicazioni scientifiche che presentano costi computazionali molto elevati. LTSP, invece, nasce con finalità più marcatamente sociali, con lo scopo di favorire la diffusione delle tecnologie informatiche in tutti quei contesti in cui si hanno scarse risorse, come scuole, progetti di cooperazione, ecc. Una differenza negli scopi che non può non avere ripercussioni sull'architettura dei due sistemi.

Nel capitolo-3 si mostrano nel dettaglio i principi di funzionamento dei due sistemi di cluster, mettendo, inoltre, in evidenza i pregi ed i difetti di ciascuno dei due. Particolare rilievo, è stato dato a Migshm, un elemento aggiuntivo di openMosix, ancora in fase sperimentale, che consente al sistema di estendere le proprie caratteristiche e le proprie funzionalitè anche alle comuni applicazioni utente.

La parte più importante di questo lavoro è costituita dal capitolo 4, nel quale sono descritti nel dettaglio tutti gli esperimenti che sono stati condotti sui sistemi di cluster che sono stati studiati. I test, sono stati eseguiti su batterie di macchine obsolete fornite da ISF, all'interno dei locali messi a disposizione dal BugsLab, presso lo Spazio Pubblico Autogestito Strike. Ingegneria Senza Frontiere ha avuto in tutto questo lavoro di tesi un importanza fondamentale; infatti, oltre a ideare il progetto di Trashware di cui questo lavoro, come detto, è solo uno degli aspetti, e fornire il materiale hardware su cui sono stati condotti gli esperimenti, ha messo a disposizione competenze ed esperienze tecniche ed umane, di fondamentale importanza. Lo sviluppo del progetto è stato possibile grazie alla stretta collaborazione di tutti i membri del gruppo Trashware di ISF, che hanno cooperato condividendo saperi e competenze. La fase di studio e sperimentazione si è avvalsa, inoltre dell'aiuto prezioso degli operatori del BugsLab che, oltre ad offrire supporto logistico, hanno condiviso con noi le loro competenze contribuendo alla nostra formazione e consentendo di generare un ambito di interscambio di esperienze tecniche, molto utile per la nostra attività.

Di fondamentale importanza è anche il continuo contatto con le comunità di sviluppatori di openMosix ed LTSP, attraverso le due liste elettroniche di discussione:

- openmosix-general@lists.sourceforge.net;
- ltsp-discuss@lists.sourceforge.net;
- trashware@lists.linux.it.

Tramite questi due strumenti abbiamo potuto apprendere tutte le novità sui due sistemi; essi, inoltre, ci hanno consentito di chiarire dubbi e risolvere problemi tecnici.

L'implementazione di openMosix, in particolare, si è avvalsa della collaborazione diretta di due degli sviluppatori del sistema: Mirko Caserta e Cristiano De Michele. Il primo ha partecipato alla fase di allestimento del cluster, mentre il secondo ha fornito il suo aiuto ed ha messo a disposizione la sua esperienza rivelatasi di grande utilità nella sperimentazione su Migshm.

Grazie allo IASI (Istituto dell'automazione dei sistemi e informatica del CNR), che ci ha messo a disposizione un laboratorio informatico di computer di ultima generazione, è stato anche possibile valutare con alcuni test il comportamento di openMosix in condizioni ideali di funzionamento (macchine identiche e molto potenti).

Nel capitolo 4 vengono, inoltre, mostrati grafici che illustrano il comportamento del nostro cluster in rapporto a macchine con architetture moderne (Pentium IV). Da essi si evince un comportamento eccellente del nostro cluster "povero", in particolare per tutte quelle applicazioni che richiedono una grande quantità di calcoli. Meno brillanti sono i risultati legati all'applicazione di Migshm, a causa, anche, del livello di maturità del progetto.

Infine, nel capitolo dedicato alle conclusioni si dà ampio spazio ai commenti su questo lavoro e sui risultati ottenuti, delineando, inoltre, possibili sviluppi futuri nel lavoro del gruppo Trashware di ISF.

Capitolo 1

Rivoluzione digitale e Digital Divide

1.1 Rivoluzione Digitale

La rivoluzione digitale, generata ed alimentata dallo sviluppo delle tecnologie dell'informazione e della comunicazione (nel seguito indicate con l'acronimo inglese "ICT"), ha modificato in maniera radicale il modo in cui le persone pensano, comunicano, agiscono, lavorano. Si sono generati nuovi modi di creare conoscenze, di educare e di diffondere informazioni di ogni tipo. Si è modificato il modo in cui nel mondo si conducono gli affari economici e si gestiscono i governi. L'importanza centrale dell'ICT nello sviluppo mondiale è riconosciuto e ribadito anche dall'ONU quando afferma che l'accesso alle informazioni ed alla conoscenza è prerequisito fondamentale per il raggiungimento degli obiettivi di sviluppo della dichiarazione del millennio[1]. Nella dichiarazione dei principi del WSIS (summit mondiale sulla società dell'informazione) viene riconosciuto l'immenso impatto

[1]L'estirpazione dell'estrema fame e povertà, il raggiungimento di un livello di educazione primaria; la promozione dell'uguaglianza tra i sessi e l'empowerment delle donne; la riduzione della mortalità infantile; la lotta all'HIV/AIDS, alla malaria e alle altre malattie; la garanzia della sostenibilità ambientale; e lo sviluppo di una partnership mondiale per l'ottenimento di un mondo più pacifico, giusto e prosperoso

che l'ICT ha su ogni aspetto della vita degli esseri umani. Il rapido progresso di queste tecnologie apre opportunità completamente nuove per il raggiungimento di più alti livelli di sviluppo. La capacità dell'ICT di abbattere alcuni ostacoli tradizionali, primo tra tutti quello della distanza geografica, rende possibile per la rpima volta nella storia, l'utilizzo delle potenzialità delle nuove teconlogie informatiche a vantaggio di milioni di persone in ogni angolo della Terra. Sotto condizioni favorevoli, queste tecnologie possono essere strumenti molto potenti per aumentare la produttività, generare crescita economica, ridurre la disoccupazione, migliorando così la qualità della vita per tutti. Inoltre l'ICT può favorire la partecipazione e l'inclusione delle popolazioni nella vita politico-economica in ciascun paese, consentendo un reale intervento delle persone sulle decisioni che le riguardano. L'ICT può garantire la creazione di network e quindi di spazi pubblici di dibattito tra le persone, canali attraverso cui far condividere conoscenze ed esperienze ed in cui far circolare notizie. Lo sviluppo delle tecnologie dell'informazione potenzialmente aprirà nuove strade per la diffusione e la socializzazione di servizi: è il caso della telemedicina che consentirebbe agli abitanti anche di luoghi isolati ed irraggiungibili di avere, ad esempio, diagnosi on-line o la possibilità di ottenere analisi mediche senza dover necessariamente accedere fisicamente ad un centro specializzato. Anche nel campo dell'educazione si aprono strade tutte nuove che potrebbero garantire un accesso più ampio all'istruzione. L'e-commerce garantirebbe la possibilità di comunicazione tra realtà locali isolate e il mercato globale, che le nuove tecnologie stanno contribuendo a sviluppare. Per far questo sarebbe necessario che molti degli ostacoli attualmente esistenti che impediscono l'accesso alle tecnologie, vengano abbattute tramite l'innovazione tecnologica, la competizione di mercato e l'iniziativa statale. Da questo punto di vista è molto importante trovare un modo per ridurre i costi legati alle tecnologie, e superare i problemi legati all'assenza di infrastrutture fondamentali come le reti telefoniche cablate. E' per questo che sarà particolarmente importante riuscire a mettere a disposizione delle popolazioni computer ancora perfettamente funzionanti ma a prezzi più bassi (pensiamo a macchine di penultima generazione opportunamente ottimizzate), e fornire loro la capacità di accesso a Internet tramite sistemi senza

fili (wireless). Le tecnologie digitali potrebbero contribuire ad eliminare alcune delle disparità che si osservano a livello mondiale, dando più forza alla voce dei paesi in via di sviluppo, dissolvendo le barriere nazionali e rafforzando il processo di democratizzazione. Proprio la grande importanza che l'ICT riveste per lo sviluppo della società in ogni suo aspetto, rende ancora più stridente il divario tra paesi ricchi e paesi poveri, cioè tra chi ha la possibilità di accompagnare quello sviluppo traendone i massimi benefici, e chi invece a causa di una arretratezza economica, culturale ed infrastrutturale, rimane tagliato fuori. Appare evidente quindi che il superamento del cosidetto **divario digitale** (digital divide) tra il Nord ed il Sud del mondo diventi in questo contesto una delle sfide più urgenti che si pongono all'attenzione della comunità internazionale.

1.2 Divario Digitale

Divario digitale è il termine utilizzato per indicare le disuguaglianze nella fruizione (accesso ed utilizzo) delle tecnologie informatiche. Ciò che si osserva, nonostante l'importanza che tali tecnologie ricoprono per lo sviluppo economico e sociale a livello globale, è una sempre maggiore difficoltà nella loro utilizzazione per alcune categorie sociali e addirittura per interi Paesi. Se da un lato la rivoluzione tecnologica in atto, favorita dalla crescita del World Wide Web, offre alla popolazione mondiale enormi possibilità di sviluppo, dall'altra contribuisce ad alimentare nuove forme di disuguaglianza. Tale rivoluzione presuppone infatti grandi investimenti e la presenza di infrastrutture e servizi spesso assenti in numerosi paesi. Inoltre, la possibilità di utilizzare strumenti sempre più efficaci è legata alla capacità dei diversi paesi di educare le popolazioni al loro utilizzo. Dal punto di vista culturale ed educativo possiamo dire che va aumentando il divario tra società alfabetizzate e società in cui il tasso di analfabetismo è ancora molto alto. Inoltre, oltre a questo fenomeno che rigurda le differenze tra paesi con diversi gradi di sviluppo, anche nei paesi più sviluppati si vanno generando le condizioni per una nuova forma di analfabetismo: l'analfabetismo elettronico.

1.2.1 Alcuni dati sul Divario Digitale

Per una analisi della situazione mondiale rispetto alla questione della fruizione delle tecnologie digitali, si prenderà in considerazione uno studio molto dettagliato condotto da Orbicon (Unesco Chairs in Communication)[2], pubblicato nell'Ottobre del 2003[50] . Questo studio è basato sull'analisi dello stato di ciascun paese in termini di fruizione e diffusione dell'ICT, di investimenti nelle nuove tecnologie e di livello medio di istruzione, nel quinquennio 1996-2001. In questa analisi, infatti, per ciascun Paese vengono introdotti tre indicatori:

- Infodensity: somma di tutti gli indici della capacità produttiva in riferimento all'ICT (capitale, forza lavoro, infrastrutture ma anche il livello medio di istruzione della popolazione)
- Info-use: flussi di consumo delle ICT
- Infostate: aggregazione di infodensity e info-use

Sulla base del valore dell'infostate (che viene assunto come indice di sviluppo tecnologico dei paesi) è stato possibile stilare una graduatoria su base mondiale di tutti gli stati[3]. E' possibile suddividere le nazioni in cinque gruppi (da A ad E). Al gruppo A appartengono i paesi dell'Europa dell'Ovest (comprese le nazioni Scandinave) e del Nord America ed in più l'Australia, la Nuova Zelanda, Hong Kong, Singapore, Giappone e Corea del Sud (tutti questi paesi hanno un valore per l'Infostate che è il 160% del valore medio mondiale). Il gruppo B contiene i paesi dell'Europa del Sud (Portogallo, Italia, Spagna; Grecia; Malta e Cipro), i paesi dell'Europa dell'est, il Chile, l'Argentina e l'Uruguay, il Bahrain e gli Emirati Arabi, Malesia e Macau (tutti paesi che hanno un indice di sviluppo intorno a quello medio). Il gruppo C contiene paesi dell'America Latina (Brasile, Costa Rica, Panama, Venezuela, Colombia e Perù), paesi Arabi (Qatar, Kuwait, Giordania, Arabia Saudita), il Sud Africa, le Mauritius e Thailandia. Al gruppo D

[2] Orbicom: creata nel 1994 congiuntamente dall'UNESCO e dalla Universitè du Quèbec à Montrèal

[3] cfr. Appendice A

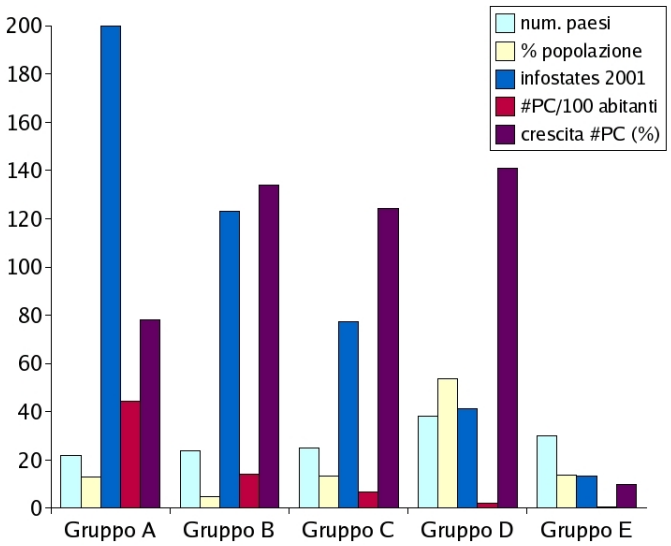

Figura 1.1: dati per gruppi di paesi

apprtengono Paesi nei quali è concentrato circa il 50% della popolazione mondiale ma con valori dell'Infostate compresi tra il 25% e il 60% del valore medio. In questo gruppo troviamo la Cina le Filippine e l'Indonesia, alcune nazioni dell'Africa del Nord, i Paesi restanti dell'America Latina, alcuni Paesi asiatici (Iran, Armenia, Kirgikistan, Mongolia e Vietnam) e alcune nazioni dell'Africa Subsahariana. Infine al Gruppo E appartengono sostanzialmente Paesi Africani ed alcuni Paesi Asiatici (Pakistan, Bangladesh, Nepal). Questi gruppi con le percentuali di popolazione e l'indice di sviluppo teconlogico che li rappresentano sono mostrati nella figura-1.1. In essa compaiono anche i dati che fanno riferimanto al numero di PC per 100 abitanti ed alla percentuale di crescita di personal computer nelle case nel quinquennio oggetto dello studio.

Lo studio condotto da Orbicom evidenzia una crescita dell'indice di sviluppo in tutti i gruppi di Paesi (figura-1.2). Tale crescita evidenzia una tendenza mondiale, che però non ci dà una misura del Digital Divide. Sono le differenze nell'Infostate a fornire i dati per monitorare l'evoluzione del divario tra i vari stati. Le differenze

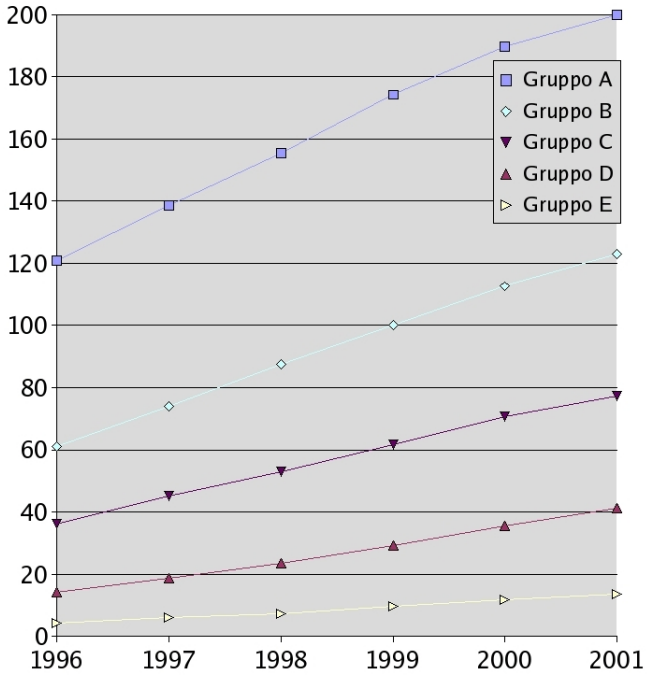

Figura 1.2: andamento dell'indice di sviluppo tecnologico per i gruppi

tra i Paesi più sviluppati e le nazioni più arretrate **crescono in valore assoluto ogni anno**. L'unico dato confortante è che il tasso di crescita annuo dei Paesi con un livello di sviluppo più basso è più grande di quello dei Paesi maggiormente sviluppati come mostrato nella figura 1.3

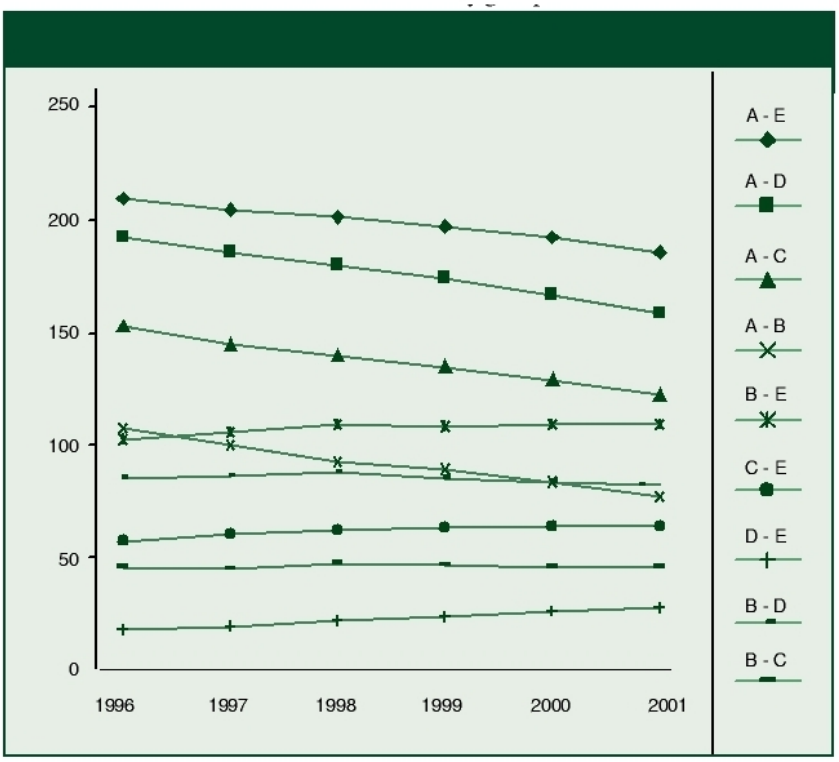

Figura 1.3: Evoluzione del grado di divario tra gruppi di paesi

La figura sembra mostrare che il Digital Divide si sia ridotto con gli anni; in realtà non è così: da un lato osserviamo che le nazioni più arretrate dal punto di vista tecnologico hanno un tasso di crescita nei cinque anni che è maggiore di quello dei paesi del gruppo A, per contro però la differenza in valore assoluto tra i valori dell'Infostate è in continuo aumento. Se volessimo interpretare ottimisticamente il dato, potremmo dire che il fatto che i paesi meno sviluppati abbiano un tasso di crescita maggiore, è un indicatore del fatto che in essi si assiste ad una

crescita percentuale degli investimanti nell'ICT, con uno sviluppo particolarmente accelerato per la Cina ed il Brasile che in cinque anni hanno aumentato il loro tasso di sviluppo del 300%. In generale molti esempi possono essere citati per mostrare l'espansione delle opportunità digitali nei paesi in via di sviluppo. Molti paesi del Sud-Est Asiatico stanno provando ad emulare il modello di sviluppo adottato dal Giappone nel periodo della ricostruzione post-bellica. La Malaysia, per esempio, presenta un tasso di penetrazione di telefoni cellulari di uno su dieci abitanti, scuole sempre più cablate e 21 host di internet su 1000 persone. Così se da un lato gli Stati Uniti sono il primo produttore mondiale di apparecchiature elettroniche, dall'altro la classifica vede le nazioni del Sud Est Asiatico ai primissimi posti. Per quanto riguarda altre regioni, l'area del sud dell'India è spesso citata per la sua importanza nello sviluppo di software. Nell'Europa Centrale e dell'Est, Slovenia, Estonia, Slovacchia e Lituania hanno fatto moltissimi sforzi per far in modo che la popolazione potesse utilizzare computer ed avere accesso alla rete. Il governo Estone ha fornito punti di accesso ad Internet pubblici distribuiti su tutto il territorio, all'interno di scuole, uffici postali, biblioteche ed ospedali. Tra tutti i continenti quello africano appare ancora quello più indietro nello sviluppo.

Nonostante questa tendenza ad un aumento degli investimenti, dall'interpretazione dei dati appare chiaro che la maggior parte della popolazione mondiale è ancora esclusa da questa rivoluzione tecnologica a da tutti quei benefici che essa può portare (telemedicina, telelavoro, e-learning, ecc).

Un dato di estremo interesse per i nostri studi è la diffusione dei PC rispetto al numero di abitanti. Il dato ci dice che nei paesi del gruppo A si assiste ad una crescita del 15,6% su base annua, mentre nei paesi meno teconologicamente sviluppati si ha un numero di PC per 100 abitanti assolutamente insignificante e una crescita molto bassa (circa il 2%).

Facendo una valutazione generale possiamo dire che nei nei paesi del Sud del Mondo la difficoltà nello sviluppo e nella diffusione dell'ICT è data dalla carenza di infrastrutture di telecomunicazione spesso concentrate per lo più nelle grandi città e totalmente assenti nelle zone rurali (nelle quali vive invece la maggior parte

della popolazione), dai costi molto elevati delle connessioni alla rete telefonica e , per l'appunto, dalla scarsa disponibilità di computer. Oltre a queste motivazioni di carattere tecnico si dovrà aggiungere anche il fatto che il livello di istruzione è in molti casi assolutamente deficitario tanto da indurre gli analisti a parlare di divario conoscitivo (knowledge divide). E' indicativo, infatti, che proprio nei Paesi in cui l'accesso ad Internet è più basso, si abbia anche un livello di istruzione inferiore. In quei paesi si osserva una stretta correlazione tra la diffusione dell'ICT e la fruizione dei media tradizionali come giornali, telefoni e televisioni. Questo ci dice che i problemi di accesso ad Internet (per esempio) nei paesi meno sviluppati non sono dovuti alla natura del medium stesso, ma evidentemente ai problemi endemici delle società più povere.

1.2.2 I Paesi più sviluppati ed il Divario Digitale

Problema diverso si ritrova nei Paesi più sviluppati. In essi si ha una grande diffusione delle tecnologie digitali. Tale diffusione è legata, ovviamente al livello di ricchezza presente, che oltre a consentire la fruizione delle tecnologie, ha anche incentivato la ricerca con opportuni investimenti. Non è un caso, ad esempio, che proprio nei Paesi in cui sono presenti Nokia ed Ericsson ci sia un utilizzo di telefoni cellulari tra i più alti d'Europa. Il termine Divario Digitale assume qui una coloritura un po' diversa: indica infatti lo scarto tra tutti coloro che hanno possibilità di accesso alle tecnologie e tutti quelli che invece si vedono negata tale opportunità. Uno studio del Dipartimento del Commercio USA, condotto a partire dal 1993[4] , dimostra che c'è una mancanza di accesso ai computer da parte delle classi più povere, delle persone in possesso di un titolo di istruzione secondaria, delle popolazioni di colore ed Ispaniche, delle comunità rurali e delle donne. Vediamo quali sono i fattori discriminanti:

- **Reddito**. Il Divario Digitale è un fenomeno sfaccettato, ed è legato a molti aspetti diversi, ma sicuramente la differenza di risorse è comunemente assunta come una delle cause più importanti. Cioè la tendenza delle persone

[4]Falling Through the Net. Washington DC: US Department of Commerce

a usare le tecnologie è molto legata al reddito, al tipo di occupazione e alla educazione. Nello studio del dipartimento del Commercio Statunitense si evidenzia come il numero dei PC nelle case sia quadruplicato tra il 1984 e 1997, ma che allo stesso modo questo periodo ha visto crescere le disparità nel possesso dei dispositivi hardware tra strati sociali. Uno studio dell'OCSE[42], che ha raccolto dati dalla Francia, dal Giappone e dagli USA, ha confermato la sostanziale disparità nell'accesso ai PC in base ai differenti redditi familiari;

- **Occupazione**. Anche il tipo di lavoro sembra essere una discriminante molto importante. Uno studio condotto dai governi di Gran Bretagna, Finlandia e Danimarca, riporta un dato inequivocabile: i manager e i liberi professionisti che hanno più mezzi messi loro a disposizione fanno un utilizzo di tecnologie che è doppio rispetto a quello degli altri comuni impiegati e triplo rispetto agli operai ed ai disoccupati;

- **Educazione**. Il livello di educazione è una variabile molto importante, visto che scuole ed università educano all'utilizzo dei computer e forniscono talvolta la possibilità di navigare gratuitamente. In più il livello di istruzione conseguito è strettamente legato al livello del susseguente impiego, cosa che, come visto, ha a sua volta una grande importanza dal punto di vista del divario. Ed infatti si può osservare che il divario che si ha in base all'educazione è assolutamente analogo a quello che si vede rispetto all'occupazione;

- **Sesso**. In questo campo possiamo dire che le differenze tra uomini e donne nell'utilizzo del PC e nell'accesso ad Internet rispecchiano sostanzialmete le differenze sociali tra i sessi;

- **Età**. In Europa si evidenzia una situazione per la quale gli utilizzatori delle tecnologie digitali sono sostanzialmente i giovani tra 18 e 25 anni mentre la percetuale di utenti decresce vertiginosamente al crescere dell'età con una percentuale di utenti over 65 che in tutta Europa è molto vicino allo zero.

1.2.2.1 Accessibilità e consumismo informatico

Dal quadro delineato nel paragrafo precedente si evince che accanto alle persone che si possono permettere soluzioni hardware di ultima generazione, abbiamo tutta una serie di categorie sociali che invece per carenza di istruzione (per esempio, anziani), per problemi economici, per problemi legati a barriere culturali (immigrati che hanno difficoltà a reperire software nella propria lingua), per disabilità fisiche, rimangono escluse da questo meccanismo. E' opportuno inoltre evidenziare una certa tendenza cui assitiamo negli ultimi anni nei paesi più sviluppati, e cioè al radicamento di un atteggiamento generale di "consumismo" anche a livello informatico. Vengono messi sul mercato computer sempre più potenti con una capacità di calcolo che raddoppia ogni 18 mesi (legge di Moore[5]). Ad HW sempre più performanti corrispondono software sempre più complessi e potenti, che hanno bisogno di girare su macchine migliori e viceversa. Si è, cioè, generato una sorta di circolo vizioso nel quale l'hardware insegue il software e questo quello. In tale meccanismo, ogni applicativo è pensato per una determinata macchina e non per le precedenti. Si genera, così, una specie di inaccessibilità all'indietro per la quale, per avere accesso alle nuove funzioni di un sistema appena prodotto si è obbligati a sostituire anche la macchina.

In questo sistema di duopolio anche utenti casalinghi con limitate necessità di calcolo si vedono costretti ad acquistare delle macchine assolutamente sovradimensionate, quando invece computer ancora perfettamente funzionanti (anche se non di ultima o penultima generazione) vengono dismessi.

1.3 Il problema dell'hardware obsoleto

Durante l'anno 2002 sono state dismesse, perché guaste o semplicemente perché sostituite con modelli più potenti ed aggiornati, circa 12.000 tonnellate di

[5]Gordon Moore, uno dei fondatori di Intel dichiarò nel 1965 di aspettarsi che il numero di transistor per microprocessore sarebbe raddoppiato ogni 18 mesi. La sua previsione, ben presto nota come legge di Moore, si sarebbe dimostrata accurata per quasi 40 anni

monitor, 12.400 tonnellate di computer tra desktop e portatili (compresi mouse, tastiere, modem ecc.), 1240 di servers e workstations, 900 di scanner, 2600 di stampanti, 13.800 di fax e copiatrici, 5000 di toner, superando così la soglia del milione di tonnellate di spazzatura elettronica prodotta nel nostro Paese nell'ultimo decennio. Di tutto questo meteriale il 90% finisce in discarica. I dati riguardanti l'unione Europea ci dicono che in media ogni cittadino dell'unione produce ogni anno 20 Kg di spazzatura elettronica e che tale cifra è destinata ad aumentare con un tasso complessivo che oscilla nelle previsioni tra il 16% e il 28% annuo. Secondo il WWF, l'incenerimento dei RAEE (rifiuti di apparecchiature elettriche ed elettroniche) emette nell'atmosfera circa 36 tonnellate di mercurio e 16 di cadmio all'anno, e contribuisce per più di metà del piombo immmesso negli inceneritori. A queste cifre sull'inquinamento legato allo smaltimento dei rifiuti elettronici, dobbiamo aggiungere anche quelle legate alla produzione dei computer. Da uno studio condotto dalle Nazioni Unite[48] è emerso che per fabbricare un PC ed un monitor sono necessarie 1,8 tonnellate di materie prime. In particolare per produrre un normale desktop equipaggiato con un monitor CRT da 17" sono richiesti 240 kg di combustibili fossili, 22 kg di sostanze chimiche e 1500 kg di acqua. Sembra certo poi, che alla fine di questo decennio, il numero di apparecchi finiti nelle discariche arriverà a quota 3 miliardi (cifra che comprende computer, cellulari, televisori, frigoriferi, stampanti e tutte le periferiche). Vediamo nel dettaglio alcune nozioni sui fattori di rischio, per l'ambiente e per gli operatori, collegati alle principali sostanze presenti all'interno delle apparecchiature elettriche ed elettroniche.

- **Piombo**. Nelle apparecchiature elettroniche viene utilizzata una percentuale compresa tra l'1,5 e il 2,5% di tutto il piombo estratto attualmete al mondo, principalmente per il vetro dei tubi catodici e per le saldature indispensabili nell'assemblaggio dei circuiti stampati. Gli effetti nocivi riscontrati in persone esposte per motivi professionali a questa sostanza coinvolgono il sistema nervoso centrale e periferico, quello circolatorio, quello endocrino ed i reni. Lo stoccaggio anche temporaneo, di apparecchiature elettriche ed elettroniche in siti di smaltimento o discariche può

provocare l'inquinamento delle falde acquifere sottostanti se il terreno non è adeguatamente impermeabilizzato;

- **Cadmio**. Secondo le stime dell'agenzia nazionale per la protezione dell'ambiente (ANPA), la dismissione di oltre 315 milioni di computer prevista tra il 1997 e il 2004 sul territorio dell' Unione Europea renderà necessario avviare a trattamenti specifici 900 tonnellate di questo elemento. Il cadmio è nocivo perché compromette le funzioni renali e riproduttive e causa una demineralizzazione dell'apparato scheletrico;

- **Mercurio**. Nelle apparecchiature elettroniche viene utilizzato il 22% di tutto il mercurio prodotto nel mondo, esso è utilizzato soprattutto nei termostati, nei sensori di posizione, nei relais e negli interruttori; è presente poi nelle batterie e nei circuiti stampati. Nelle persone che vi entrano in contatto produce danni al cervello ed in particolare alle zone che regolano la vista, l'equilibrio ed il coordinamento;

- **Cromo esavalente**. Viene usato da alcuni produttori di PC per difendere contro la corrosione le placche d'acciaio non trattate e galvanizzate. Può essere assorbito facilmente dalle cellule e può portare forti reazioni allergiche. L'incenerimento non controllato di spazzatura contenente cromo è stato bandito tramite apposite convenzioni internazionali proprio a causa dell'alta nocività delle ceneri che da tale incenerimento si possono sprigionare;

- **Materie plastiche**. L'ANPA prevede che per il 2004 ci saranno da smaltire circa 1.935.000 tonnellate di materie plastiche contenute nei computer e nelle periferiche giunti alla fine del loro ciclo vitale. La maggior parte di tale quantità è costituita da PVC, usato fino alla metà degli anni novanta per la sua resistenza e la non infiammabilità. La successiva scoperta della sua alta nocività durante l'incenerimento ed il riciclaggio ne ha causato la messa al bando successiva. Oggi al posto del PVC si usa l'ABS ma le grandi

quantità di rifiuti elettronici da smaltire ne fanno ancora una minaccia per l'ambiente;

- **Ritardanti di fiamma Bromurati**. I brominati sono una classe di sostanze chimiche utilizzate nei pc per garantire protezione contro l'infiammabilità per la loro proprietà di ritardare la combustione dei materiali plastici innalzando la temperatura d'innesco. Presentano una tossicità e producono degli effetti comparabili a quelli delle diossine.

La presenza e gli effetti degli elementi e delle sostanze elencati rendono il processo di riciclaggio dei Raee molto pericoloso per l'ambiente e per gli operai addetti allo smaltimento. Attualmente, in Europa il 90% dei rifiuti elettronici viene avviato alla discarica, incenerito o frantumato senza che sia stato precedentemente trattato. Come se non bastasse, le componenti di piccole dimensioni vengono trattate come normali rifiuti domestici e, come tali finiscono direttamente negli inceneritori.

1.3.1 L'apparato normativo

Le cifre non possono non preoccupare, anche in considerazione del fatto che l'Italia non sembra disporre di misure adeguate a fronteggiare un tale volume di scarti, spesso pericolosi per l'uomo e per l'ecosistema. Si rende necessaria una corretta sensibilizzazione del cittadino e delle imprese ma anche l'emanazione di leggi che regolamentino in modo chiaro lo smaltimento dei RAEE, oltre alla creazione di centri specializzati per la separazione dei componenti delle apparecchiature elettroniche e per il loro riutilizzo. La gestione dei rifiuti elettronici è regolamentata a livello europeo dalle direttive n. 2002/96/Ce e n. 2003/108 secondo le quali gli Stati membri devono provvedere affinchè, entro il 13 Agosto 2005 i produttori prevedano il finanziamento dei costi di raccolta, trattamento, recupero e smaltimento ecologicamente corretto dei Raee provenienti da utenti diversi dai nuclei domestici e originati da prodotti immessi sul mercato dopo il 13 Agosto 2005. Inoltre la normativa sancisce che saranno gli stessi fornitori a farsi

carico del trattamento dei rifiuti elettronici prodotti prima di quella data e sostituiti per obsolescenza o malfunzionamento. Le direttive imponevano anche agli stati membri di mettere in vigore entro il 13 agosto 2004 le disposizioni legislative, regolamentari e amministrative necessarie per la loro attuazione. A tutt'oggi, solo la Grecia si è adeguata. Dal 1 luglio 2006, inoltre, sarà vietato costruire computer, telefonini ed elettrodomestici usando piombo, mercurio, cadmio e cromo esavalente. Il passo ulteriore che la UE ha previsto per gli stati membri è che entro il 31 dicembre 2006 venga obbligatoriamete raggiunto un tasso minimo di raccolta selettiva di apparecchi a fine vita provenienti dai nuclei domestici pari a 4 kg/anno procapite. L'Italia non ha ancora recepito la direttiva UE, per questo nella pratica dal punto di vista legislativo, è il decreto Ronchi (D.lgs. 5 febbraio 1997, n.22) a definire l'argomento. Tale decreto è emanato in recepimento delle direttive comunitarie in materia di rifiuti, rifiuti pericolosi, imballaggi e rifiuti di imballaggio. La disciplina impone la responsabilizzazione e la cooperazione di tutti i soggetti coinvolti e favorisce la riduzione della produzione di rifiuti e all'incentivo del loro recupero e reciclaggio. Fa divieto di abbandonare rifiuti sul suolo, sul sottosuolo e nelle acque superficiali e sotterranee ed impone, sostanzialmente, di portare i rifiuti tecnologici in discaricarica per essere avviati alle procedure di recupero o di smaltimento. Il legislatore ha operato in modo da favorire ed incentivare il reimpiego ed il riutilizzo dei rifiuti, il recupero dei rifiuti al fine di ottenerne materia prima, l'utilizzo dei rifiuti come combustibile o altra forma di energia. Ancora fissa una serie di obblighi e divieti a carico dei produttori e dei detentori dei rifiuti come l'obbligo di autorizzazione per la gestione di impianti fissi e mobili di gestione dei rifiuti, il divieto di miscelazione dei rifiuti pericolosi, l'obbligo di tenuta di un formulario di identificazione per il trasporto e di un registro di carico. Sono stati inoltre istituiti i consorzi obbligatori per il recupero di alcuni materiali specifici (polietilene, batterie esauste, oli usati). Il decreto ormai emanato più di sette anni fa, in uno scenario molto più arretrato di quello attuale, appare inadeguato rispetto a quello che è il tasso di sviluppo tecnologico attuale in Italia. In esso non viene fatto nessun riferimento alla modalità corretta con cui i rifiuti elettronici (e-waste) dovrebbero essere trattati.

1.3.2 La politica delle aziende

Mentre i paesi dell'UE si preparano ad adeguarsi alle nuove normative comunitarie, alcune aziende hanno già da qualche tempo iniziato a sviluppare politiche di riciclaggio adottando nuove iniziative ed investendo denaro per renderle visibili alla maggioranza delle persone. Fu Hewlett-Packard la prima azienda ad offrire ai propri clienti la possibilità di dar indietro la macchina vecchia con l'acquisto di quella nuova. Una soluzione semplice che consente al produttore di avviare un computer inservibile allo smaltimento, dopo aver recuperato i componenti ancora utilizzabili. Alla HP si sono poi aggiunti tutti i grandi colossi del settore informatico, che con il tempo hanno mostrato una grande attenzione per soluzioni più eco-compatibili. In particolare HP si muove su più fronti: ha avviato un programma per la ricerca di materiali meno inquinanti da utilizzare nella progettazione dei prodotti HW, inoltre viene data molta attenzione alla costruzione di apparati facilmente smontabili, ovvero le cui componenti, una volta arrivate a fine vita, possano essere facilmente separate. Per quanto riguarda i consumabili, Hp ha istituito un servizio di ritiro di toner e cartucce esauste dai propri clienti, purchè essi abbiano una grande quantità di apparecchiature. Il servizio di recupero è affidato ad aziende partner selezionate tra i trasportatori ed i riciclatori autorizzati. In più all'utente finale viene offerto un servizio a pagamento per la restituzione dell'hardware, che consiste nella distribuzione su tutto il territorio italiano dei kit di restituzione, ossia dei box che possono contenere fino a 25 kg di materiale hardware obsoleto e che vengono ritirati, una volta pieni, dalla società partner. Per stessa ammissione della Direzione Affari governativi e pubblici di HP Italia, tale iniziativa ha avuto finora una risposta molto fredda da parte dell'utente finale che continua ancora a preferire il cassonetto. Da parte sua, Epson ha avviato un progetto per il recupero dei consumabili (cartucce e toner). Anch'essa affida ad una società terza il recupero del materiale dai grandi utenti e dai clienti; rimangono comunque fuori da iniziative di recupero gli utenti finali. Accanto ad iniziative volte al recupero ed allo stoccaggio di materiale esausto, si sta anche cercando di ridurre al massimo l'utilizzo di materiale nocivo come il piombo. Molto attiva

nel campo è anche la Fuji-Siemens, che ha creato in Germania un centro per lo stoccaggio di computer e periferiche dismesse; tutti i pezzi vengono controllati, e se in buone condizioni vengono rivenduti in un'ala della struttura adibita a questo scopo. Molte altre aziende dichiarano di avere progetti in stand-by o comunque di stare ancora nella fase di studio dei progetti. Come si vede lo scenario è molto complesso. Viene data grande enfasi a due aspetti fondamentali: l'eliminazione di materiali dannosi dal processo di fabbricazione dei dispositivi tecnologici e il corretto smaltimento dei rifiuti elettronici (recupero, riciclo, smaltimento). Non viene data invece la giusta enfasi ad un terzo aspetto findamentale: l'utilizzo più corretto dei PC e di tutte le periferiche e il riuso del materiale obsolescente ma ancora perfettamente funzionante.

1.3.3 Recupero, Riciclo e Smaltimento

Le politiche di gestione dei rifiuti fino ad oggi hanno concentrato la loro attenzione esclusivamente sulla fase finale: lo smaltimento. Le imprese da parte loro non hanno grande interesse ad alimentare il mercato o comunque la diffusione del computer usato, e conseguentemente hanno esclusivamente pensato al modo più eco-compatibili di eliminare i prodotti dismessi. In realtà, di fronte ad un problema così articolato e complesso, bisogna comprendere quale sistema offra una sufficiente capacità di recupero, minimizzi gli impatti ambientali ed eviti danni alla salute. Nonostante la tendenza a contrapporre le diverse possibilità di gestione dell'elettronica dismessa, è indubbiamente da una loro combinazione che possono venire i migliori risultati. Secondo alcuni dati forniti da WWF ed Ecoqualit[6] è possibile recuperare fino al 90% dei materiali presenti in un PC, avviando poi il restante 10% alla discarica per lo smaltimento.

Dal punto di vista dello smaltimento e del riciclaggio di un personal computer, assume grande importanza l'analisi della composizione dei materiali in esso presenti. Avvalendoci dei risultati di una ricerca condotta dalla Microelectronic and

[6]Consorzio per la tutela dell'ambiente

Computing Technology Corporation[7], possiamo dire che all'interno di un normale desktop (approssimativamente 27 kg di peso) ci sono oltre trenta elementi, caratterizzati da possibilità di riciclaggio anche molto differenti come mostrato nellla tabella1.1.

Di fronte ad una tale pluralità di elementi presenti in quantità anche minime e con tassi di efficienza di riciclaggio talvolta molto bassi, si intuisce come soltanto con volumi alti di hardware smaltito si riescono a ottenere quantità significative di materiali da recuperare. E' evidente allora che il riciclaggio da solo non è un buon modo di affrontare la dismissione dell'hardware obsoleto. Esso dovrà essere solo l'ultimo passo prima dello smaltimento vero e proprio dei materiali irrecuperabili. Un corretto processo di trattamento dei dispositivi elettronici obsoleti, dovrà constare di diverse fasi:

- **Raccolta**;
- **Cernita** e collaudo. In questa fase l'hardware raccolto viene esaminato per separare le componenti in base alle loro caretteristiche; poi viene collaudato per vedere quali componenti sono ancora funzionanti. Se questa fase dà esito negativo, si provvede ad un primo smontaggio per eliminare tutto ciò che non è riutilizzabile;
- **Prelievo componenti per il reimpiego**. Si tratta, in questa fase, di testare e recuperare tutto ciò che può essere riusato;
- **Frantumazione e selezione per il recupero dei materiali**. Questa è la fase in cui le schede vengono macinate ed in cui si provvede al recupero dei materiali che possono essere riciclati (come i metalli);
- **Smaltimento**. Solo quando tutto ciò che è recuperabile è stato preso, si può passare allo smaltimento vero e proprio.

In questo modo si riuscirà ad ottenere il massimo in termine di profitti per le aziende, abbattendo, inoltre, il tasso di inquinamento di cui abbiamo parlato

[7]Elecronics Industry Environmental Roadmap, Austin (Tx) 1996

Materiale	Quantità (kg)	Efficienza di riciclo	Uso/posizione
Materie plastiche	6,21	20%	Incl. materie organiche e silicee
Piombo	1,71	5%	Giunture dei metalli, CRT
Alluminio	3,83	80%	Struttura, CRT, connettori
Germanio+Gallio	<0,1	0%	Semiconduttori
Ferro	5,54	80%	Struttura e CRT
Stagno	0,27	70%	Saldature, CRT
Rame	1,89	90%	Conduttività, Crt, connettori
Nichel	0,23	80%	Struttura, sedi magnetiche, CRT
Zinco	0,59	60%	Batteria, emettitore al fosforo, CRT
Tantalio	<0,1	0%	Condensatore
Vanadio	<0,1	0%	Emettitore al fosforo del rosso/CRT
Berillio	<0,1	0%	Conduttore termico, connettori
Oro	<0,1	99%	Connettività, connettori
Europio	<0,1	0%	Attivatori al fosforo
Titanio	<0,1	0%	Colore, Agenti alogeni
Rutenio	<0,1	80%	Resistenza
Cobalto	<0,1	85%	Struttura, CRT
Palladio	<0,1	95%	Connettività, conduttori, connettori
Manganese	<0,1	0%	Struttura, CRT
Argento	<0,1	98%	Conduttività connettori
Antimonio	<0,1	0%	Diodo, CRT
Cromo	<0,1	0%	Decorativo, induritore
Cadmio	<0,1	0%	Batteria, emettitore al fosforo
Selenio	<0,00096	70%	Rettificatori
Radio	<0,1	50%	Spessa pellicola conduttrice
Platino	<0,1	95%	Spessa pellicola conduttrice
Mercurio	<0,1	0%	Batterie, interruttori
Silicio	6,75	0%	Vetro, dispoditivi di stato solido

Tabella 1.1: Elementi in un PC ed efficienza di riciclaggio

all'inizio di questo paragrafo e consentendo all'Europa di risparmiare 120 milioni di gigajoule, equivalente a 2,8 milioni di tonnellate di petrolio ogni anno, con un risparmio energetico pari al 60-80% rispetto all'utilizzo di materia vergine.

Dal punto di vista commmerciale, per le aziende il problema è ricavare profitto dal disassemblaggio e recupero di componenti, oppure dal ricondizionamento e riutilizzo dei computer. La prima strada è praticata da alcune imprese con profitto, mentre la seconda è di più difficile realizzazione, sostanzialmente perché il valore commerciale di un PC obsoleto è molto basso e, di solito, non copre le spese necessarie per ricondizionarlo. E' per questo che l'attività di recupero dei PC obsoleti è svolta in Italia da associazioni senza fini di lucro che destinano le macchine ad associazioni no profit o a paesi con economie più povere.

L'aspetto del recupero dell'hardware obsoleto in vista di un suo riutilizzo è per noi di centrale importanza. Sono due le idee che ci guidano nel nostro studio: la prima è legata alla sempre più pressante necessità di smaltire grandi quantità di computer; l'altra, meno ovvia ma altrettanto importante, è dare all'obsolescenza tecnologica il suo corso naturale, e non il frenetico ritmo artificiale (di cui abbiamo precedentemente parlato all'interno del paragrafo**??**) imposto negli ultimi anni. Spesso, infatti, i computer vengono dismessi dalle aziende ancora funzionanti benché con prestazioni non al passo con il software di ultima generazione. Hanno un valore residuale che però non è sfruttabile dalle imprese perché per esse tenere un computer lento in produzione fa indirettamente crescere altri tipi di costo, specialmente in termini di produttività di un impiegato. Tale valore può però essere utilizzato convenientemente in tutte le realtà in cui la lentezza non causa l'aumento dei costi (come associazioni, scuole ecc). Riutilizzando in particolari realtà i computer dismessi si ottimizza l'utilizzo del valore economico totale della macchina.

In sostanza, si vorrebbe promuovere un corretto utilizzo della risorsa tecnologica: il pc ormai ammortizzato, lento, poco produttivo, viene sostituito da uno più competitivo ed efficiente. E' importante che le aziende imparino a liberarsi subito dei pc che non usano e che, altrimenti, occupano inutilmente spazio, ma soprattutto perdono quel valore residuo che può essere socialmente utile. Così

facendo si ottengono due vantaggi economici: il risparmio dello smaltimento immediato e la possibilità di avere a disposizione risorse informatiche accessibili a tutti.

Capitolo 2

Il Trashware

Con il termine **Trashware** si intende il riutilizzo proficuo di computer dismessi ed altrimenti destinati allo smaltimento. La parola coniata dal Gruppo Operativo Linux Empoli (Golem)[1], nasce dall'unione dei termini hardware e trashing e sta ad indicare proprio che si vanno a ricondizionare computer che altrimenti finirebbero in discarica. Chi fa Trashware riutilizza le macchine dismesse da privati, enti pubblici ed aziende per donarli, dopo essere stati opportunamente controllati e rimessi a punto, ad associazioni di volontariato o a progetti di solidarietà internazionale e cooperazione allo sviluppo.

Per molto tempo, come detto nel capitolo precedente, si è realizzato lo smaltimento dell'hardware senza troppe preoccupazioni. Pur essendo l'e-waste molto difficile da trattare, questo tipo di rifiuto è stato tenuto abbastanza facilmente sotto controllo, viste le ridotte quantità di materiale da gestire. Negli ultimi anni, però, la quantità di macchine da dismettere ha raggiunto un numero molto grande (vedi tabella 2.1), tanto che quello dei rifiuti tecnologici è diventato un problema ecologico di grande attualità (come già detto nel paragrafo 1.3).

Le motivazioni che muovono i gruppi che in Italia si dedicano al Trashware sono sostanzialmente legate alla questione ecologica di cui si parlava nel capitolo precedente, e soprattutto al corretto utilizzo dei dispositivi hardware. Spesso i computer vengono dismessi sebbene ancora funzionanti, perché troppo lenti, e

[1] Gruppo fondato nel 2000, allo scopo di diffondere l'uso del sistema operativo GNU/Linux

Anni	Server	PC	Totale
1998	72.600	615.000	688.200
1999	105.500	1.109.500	1.215.000
2000	86.500	1.176.500	1.263.000
2001	116.000	861.500	977.500
2002	127.550	1.042.500	1.170.050

Tabella 2.1: Computer avviati alla dismissione in Italia 1998-2002 (dati in unità)

tale lentezza è vista come un fattore di rallentamento della produttività di un lavoratore. Questi computer hanno comunque un valore a volte ancora molto alto (non dimentichiamo che i computer più performanti di oggi saranno considerati obsoleti tra 18 mesi[2]) che può essere sfruttato in tutte quelle realtà in cui la supposta lentezza non è un grande problema. Per chi non può permettersi un computer, come associazioni di volontariato, scuole, associazioni culturali, studenti, privati cittadini con redditi bassi ecc, avere una macchina di penultima generazione opportunamente ottimizzata e riconfigurata è una possibilità molto appettibile. In tal modo si fornisce un servizio utile alla comunità ed al contempo si ottimizza l'utilizzo del valore economico totale dei PC. Il riutilizzo non elimina, alla fine, la necessità di smaltimento chimico dei computer, però sicuramente minimizza gli sprechi e l'impatto sull'ambiente e sulla società. In ultima analisi possiamo dire che anche per le aziende può essere un vantaggio non indifferente, in quanto le macchine dismesse vengono spesso tenute ammassate nei magazzini facendo crescere i costi storici, mentre così facendo si promuove l'aggiornamento delle risorse tecniche di un'azienda, senza che questo sia un incentivo al consumo ed allo spreco. Le aziende investono, in termini di efficienza, la parte residuale di questo investimento (scarsamente sfruttabile economicamente), che può essere usata socialmente per la formazione di nuovi tecnici, i quali, a loro volta, potranno essere utili alle imprese.

[2]Legge di Moore: vedi nota 5 del paragrafo 1.3

Condizione fondamentale affinchè il Trashware possa attechire è che le aziende abbandonino alcuni comportamenti che sono dannosi per il processo di riutilizzo. Spesso, infatti, i computer dismessi vengono stipati per lunghissimi periodi di tempo in luoghi non adatti alla loro conservazione; vengono tolte loro parti vitali da tenere per casi di necessità (che quasi mai si realizzano) attuando quella che in gergo tecnico viene chiamata "cannibalizzazione dei Pc".

L'attività di Trashware è inscindibilmente legata all'utilizzo di Software Libero o OpenSource. Il software prorpietario ha, infatti, diversi tipi di incompatibilità con la natura stessa del progetto: un'incompatibilità economica, poichè l'acquisto delle licenze sarebbe troppo oneroso per associazioni di volontariato e per le popolazioni dei paesi in via di sviluppo; un'incompatibilità ideologica, poichè per la comunità Linux è di fondamentale importanza mantenere l'indipendenza dalle politiche delle grandi aziende, mentre installare software proprietario significherebbe diventare uno strumento commerciale di diffusione dei loro prodotti alle associazioni italiane e alle popolazioni del mondo; infine un'incompatibilità di carattere tecnico nel senso che è estremamente difficile reperire versioni di qualunque software non libero in grado di funzionare in modo corretto sulle macchine oggetto del recupero e, anche qualora questo fosse possibile, risulterebbe spesso inutile poichè i prodotti nuovi e aggiornati spesso non hanno caratteristiche accettabili di retrocompatibilità, ma anzi contribuiscono, come descritto nel paragrafo 1.3, in modo determinante all'obsolescenza tecnica e all'esclusione sociale.

Alla luce di questo, eploreremo nel paragrafo seguente le caratteristiche e le modalità operative del Software Libero e dell'Open Source

2.1 Free Software e software Open Source

Il concetto di software libero discende naturalmente da quello di libertà di scambio di idee ed informazioni. Negli ambienti scientifici, quest'ultimo principio è tenuto in alta considerazione per la fecondità che ha dimostrato; ad esso infatti è generalmente attribuita molta parte dell'eccezionale ed imprevedibile crescita del sapere negli ultimi tre secoli. La libertà di scambio di idee non è tuttavia

una questione puramente pratica: essa è infatti alla base dei concetti di libertà di pensiero e di espressione. Analogamente alle idee, il software è immateriale, e può essere riprodotto e trasmesso facilmente. In modo simile a quanto avviene per le idee, parte essenziale del processo che sostiene la crescita e l'evoluzione del software è la sua libera diffusione. Ed ogni giorno di più, come le idee, il software permea il tessuto sociale e lo influenza, produce effetti etici, economici, politici e in un senso più generale culturali.

Il primo a formalizzare il concetto di software libero fu Richard M. Stallman nei primi anni Ottanta. La definizione che egli ha prodotto, assume la forma di quattro principi di libertà:

- **libertà 0, o libertà fondamentale**: libertà di eseguire il programma per qualunque scopo, senza vincoli sul suo utilizzo
- **libertà 1**: libertà di studiare il funzionamento del programma e di adattarlo alle proprie esigenze.
- **libertà 2**: libertà di redistribuire copie del programma.
- **libertà 3**: libertà di migliorare il programma, e di distribuirne i miglioramenti

Il software distribuito con una licenza che rispetti questi principi è detto libero (in inglese free software). Nel 1984 Richard M. Stallman diede vita al progetto GNU, con lo scopo di tradurre in pratica il concetto di software libero, e creò la Free Software Foundation per dare supporto logistico, legale ed economico al progetto GNU. La licenza del progetto GNU, la Licenza Pubblica Generica GNU (GNU GPL), al contrario delle normali licenze d'uso, concede all'utilizzatore di un programma di godere di tutte e quattro le libertà suddette. Inoltre si occupa di tutelarle nel senso che chi modifichi un programma protetto da GPL e lo distribuisca lo deve fare ancora sotto licenza GPL. Con un gioco di parole, il nome dato a questo tipo di protezione è permesso d'autore (in inglese **copyleft**): è il criterio che prevede che le modifiche ad un programma possano essere distribuite solo con

la stessa licenza del programma originale. Le licenze usano le norme sul diritto d'autore (copyright) per togliere libertà agli utenti di un programma; il permesso d'autore usa le stesse norme per garantire quelle libertà e per proteggerle.

Nel 1998 Bruce Perens, Eric Raymond ed altre personalià nel campo del software libero si convinsero che i principi di libertà associati ad esso fossero mal visti nel mondo degli affari, a causa della loro carica ideologica. Lanciarono, così, una campagna di promozione del free software che mettesse in luce i suoi numerosi vantaggi pratici come la facilità di adattamento, l'affidabilità, la sicurezza, la conformità agli standard, l'indipendenza dai singoli fornitori ecc. Scrissero a tal fine la Open Source Definition, il documento fondamentale del movimento **Open Source**. In più la nuova definizione che essi trovarono per il software prodotto consentiva loro di superare un'ambiguità fondamentale, quella legata alla parola "free" nella lingua Inglese che può significare "libero" ma anche "gratuito". Tale ambiguità è di non secondaria importanza per chi si vuole guadagnare da vivere con la scrittura di programmi. La definizione ufficiale di "software open source" come pubblicata dalla Open Source Iniziative, si avvicina molto a quella di free software; tuttavia è per alcuni aspetti un po' più ampia ed in più ha accettato alcune licenze che vengono considerate dal movimento di Stallman come inaccettabilmente restrittive per gli utenti.

La voluta neutralità del movimento open source verso gli aspetti etici e politici del software libero è la caratteristica sostanziale che lo distingue dalla filosofia di quest'ultimo. Possiamo dire, quindi, che la differenza che c'è tra i due movimenti è politica più che pratica, visto che essi concordano sulle licenze accettabili ed hanno obiettivi e mezzi comuni. I due movimenti sono cioè in disaccordo sui principi di base, ma sono più o meno d'accordo sugli aspetti pratici. Per questo essi possono lavorare insieme su molti progetti specifici.

Al di là delle questioni ideologiche, possiamo sicuramente dire che la diffusione del software libero ed open source è diventato uno dei fenomeni più interessanti dell'intero panorama delle tecnologie informatiche, generando un livello di interesse e producendo un numero di appassionati che è paragonabile a quello dei primi anni della diffusione di Internet. Sebbene, come visto, il fenomeno non sia

recentissimo, solo negli ultimi anni ha raggiunto un numero critico di persone, che gli ha consentito di entrare nel mercato del software tradizionale.

Negli ultimi anni l'impatto di questo nuovo tipo di programmi sta diventando molto forte nell'industria del software e nella intera società. Sta diffondendo un nuovo tipo di modello di sviluppo, che trae vantaggio dal lavoro di sviluppatori sparsi per il mondo; proprio la presenza di comunità che contribuiscono allo sviluppo del software o anche che semplicemente risolvono problemi per i nuovi utenti è uno degli aspetti più affascinanti di questa realtà tutto sommato nuova. Questa tecnologia Open Source consente un modello di affari che è completamente nuovo e che ha in generale un impatto molto positivo nella creazione di nuovi mercati e di nuove opportunità di lavoro.

2.1.1 Modello di sviluppo

Oltre all'impatto che l'Open Source sta avendo sul mercato dell'Information Technology, esso sta producendo molti cambiamenti nel campo dello sviluppo del software, abbattendo quello che era il concetto classico del software engineering, per il quale solo una gestione centralizzata ed un forte controllo sull'accesso al codice sorgente garantivano un prodotto di alta qualità. Il modello proposto e diffuso dall'Open Source è per alcuni aspetti opposto; in esso, infatti, un gran numero di sviluppatori sparsi nel mondo collaborano per costruire un prodotto dall'alto standard qualitativo.

Le motivazioni che inducono a usare e sviluppare il software libero sono molte e molto diverse tra loro e sono di natura etica, filosofica o semplicemente dettate da aspetti pratici. Degli aspetti etici abbiamo già in qualche modo parlato, mentre di quelli pratici faremo ora una veloce disamina.

- L'accessibilità al codice sorgente e il diritto di modificarlo. Garantisce la possibilità di mettere a punto e di migliorare il prodotto. Consente di adattare il codice a nuovi harware (aspetto fondamentale per la nostra ricerca), di adattarlo al variare delle condizioni, e di raggiungere una piena comprensione di come il sistema lavora. E' per questo che molti esperti sono

arrivati alla conclusione che per estendere la vita di un'applicazione, deve essere disponibile il suo codice sorgente. Per ultimo dobbiamo sottolineare come l'avere a disposizione il codice sorgente consente agli sviluppatori di isolare i bachi e di fissarli;

- Il diritto di redistribuire le modifiche e le migliorie del codice. Consente a tutti di fruire dei miglioramenti apportati dagli sviluppatori al codice;

- Il diritto di utilizzo del software. Questo combinato con l'aspetto precedente garantisce che ci sia una grande quantità di utenti potenziali, quindi un mercato molto ampio che richiede sempre più supporto e "customizzazione" per il software prodotto, cosa che in generale attira sempre nuovi sviluppatori, che a loro volta contribuiscono a migliorare la qualità del prodotto e le sue funzionalità;

- Non c'è nessuno che abbia il potere di porre restrizioni all'uso del software. Cosa che invece si ha ogni qualvolta il management di una software house decide di non aggiornare i programmi per vecchie piattaforme, costringendo gli utenti a rimanere legati alle vecchie versioni, o a passare ad un nuovo prodotto o direttamente ad intervenire sull'hardware (acquistandone di nuovo);

- Non c'è un unico soggetto da cui dipenda il futuro del software. Questa cosa accade al contrario nel caso di software proprietario: se un'azienda decide di non sviluppare un programma o semplicemente chiude, nessuno può continuare lo sviluppo di quel prodotto. Questo problema è amplificato negli ultimi tempi dalle acquisizioni o dalle fusioni a cui assistiamo nel mercato del software, che spesso portano alla cannibalizzazione di un prodotto per instaurare un regime di monopolio. L'Open Source fattivamente protegge da questo pericolo, poichè è sempre possibile trovare qualcuno che si occupi di continuare lo sviluppo senza che ci siano limitazioni legali o pratiche;

- C'è sempre la possibilità di "forking". Cioè la possibilità di iniziare un codice alternativo se si percepisce che quello corrente sta intraprendendo una

gestione sbagliata (un esempio di questo è proprio il progetto OpenMosix nato dal progetto mosix nel momento in cui quest'ultimo stava diventando non open). Così nel momento in cui una release di un prodoto è rilasciata sotto licenze proprietarie si può usare quella immediatamente precedente come base di partenza per le successive rilasciate ancora sotto GPL.

Ovviamente il modello di sviluppo appena descritto porta con sè anche quelli che vengono percepiti come svantaggi. Per esempio non cè nessuna garanzia che un prodotto continui ad essere effettivamente sviluppato. In altre parole non è possibile sapere se un progetto arriverà a raggiungere uno stadio in cui sia usabile e anche se dovesse arrivare ad essere utilizzabile non è comunque possibile prevedere se il prodotto morirà per mancanza di interesse oppure no. Ovviamente questo è un problema presente anche per il software proprietario per il quale, però, è la legge del mercato a decretare l'abbandono di un progetto.

Altro limite è che alcune volte è difficile poter sapere se un progetto esiste e qual è il suo status corrente. Questo è particolarmente vero per i progetti più piccoli. A questo problema si ovvia grazie all'apparato informativo che si è creato intorno al fenomeno GNU/Linux, grazie al Web, ai giornali specializzati, ai forum, alle mailing-list, ecc.

Questo modello di sviluppo e le licenze che sono alla base del prodotto Open Source hanno generato un meccanismo di cooperazione e competizione che in ultima analisi ha portato all'alta qualità e all'alta efficienza che i progetti Open Source hanno raggiunto. Abbiamo dunque un meccanismo di cooperazione che si palesa tra gli sviluppatori all'interno di un progetto ma anche tra programmatori impegnati in progetti differenti. E' molto comune che uno sviluppatore legga e corregga i bachi del codice sviluppato da un altro. In più i miglioramenti apportati ad un codice da un programmatore sono in generale fruibili da tutti quelli che, anche in progetti diversi, stanno utilizzando lo stesso codice.

Per quanto riguarda la competizione possiamo dire che i diversi progetti Open Source competono tra loro per emergere all'interno della stessa nicchia di mercato, in tal modo sono forzati a mantenere dei livelli di qualità alti in modo da non perdere utenti e sviluppatori.

Le caratteristiche fin qui descritte sono collegate alla nostra ricerca per quello che concerne gli aspetti tecnici dell'adattabilità e della flessibilità del software allo scopo della riqualificazione dell'hardware. Rimane da comprendere come venga recepita la scelta del software libero dall'utente finale, ovvero dai fruitori della soluzione tecnologica.

Perciò faremo una valutazione di quale sia l'atteggiamento delle pubbliche amministrazioni rispetto alla diffusione del sistema operativo GNU/Linux. E successivamente ci concentreremo su alcuni progetti in cui il software open viene usato in Paesi in via di sviluppo (indicati con l'acronimo PVS) nel tentativo di colmare quel divario digitale di cui abbiamo parlato nel capitolo 1.

2.1.2 I governi e l'Open Source

La diffusione dell'Open Source Software (OSS) nel mondo è un fenomeno molto ampio a cui molti governi stanno guardando con grande interesse. Vediamo alcuni esempi delle politiche adottate da alcuni paesi in tutto il mondo:

- **Australia**. Il governo di Adelaide ha emanato una legge nel 2003 per la quale, ovunque sia possibile, l'autorità pubblica dovrebbe privilegiare l'OSS rispetto al software proprietario;

- **Brasile**. Il presidente Brasiliano Luiz Ignacio Lula da Silva sta conducendo una politica con cui si raccomanda che i ministeri federali, le agenzie e le imprese di stato installino programmi open source, invece di software proprietari sui nuovi computer. L'obiettivo di questa politica è di avere Linux installato su almeno l'80% dei computer dei pubblici uffici[3];

- **Cina, Giappone, Sud Corea**. Nel settembre 2003 i governi annunciano la loro intenzione di lavorare con il settore privato per sviluppare nuovi software basati su Linux per dispositivi come telefoni cellulari, camere digitali e sistemi di navigazione satellitare;

[3]Nell'ottobre 2003 in proposito è stato pubblicato il documento contenente le linee guida per l'implementazione del free software nel Governo Federale

- **Danimarca**. Nell'Ottobre 2002, il ministero per le tecnologie nel suo report dal titolo "Software Open Source nell'e-govenment", raccomanda che il governo assuma un ruolo attivo nel promuovere l'utilizzo di formati di file standardizzati alternativi a quelli proprietari;

- **Commissione Europea**. Nell'Agosto 2003, la commissione Europea ha iniziato uno sforzo per incoraggiare la pratica dell'uso di software open nella pubblica amministrazione, focalizzando l'attenzione in una prima fase, sulla creazione di centri nazionali e regionali in cui sviluppare competenze per facilitare il passaggio al nuovo sistema;

- **Germania**. Nel Maggio 2003 la città di Monaco ha annunciato il suo piano per migrare i 14000 computers dell' amministrazione al sistema Gnu/Linux. Il ministro degli interni dichiara che l'operazione avviata in tutta la Germania, consente di raggiungere un maggior livello di sicurezza, evitanto una monocultura e riducendo la dipendenza da un singolo venditore, con un drastico taglio dei costi;

- **India**. Il governo indiano ha lanciato "the Linux India Initiative", un progetto che si concentrerà sullo sviluppo di centri di ricerca, di gruppi di studio, di progetti pilota che accompagnino la ricerca e lo sviluppo di software Open Source;

- **Filippine**. In questo paese il governo ha dato avvio ad un progetto di sviluppo di pacchetti di prodotti Open Source rivolti principalmente alle agenzie governative, alle scuole ed alle piccole aziende. L'Advance Science and Technology Institute (ASTI) che afferisce al Dipartimento delle Scienze e delle Tecnologie (DOST) ha rilasciato una versione semplificata di Linux chiamata Bayanihan Linux. ASTI sta anche sviluppando un sistema thin client e soluzioni embedded basate sull'OSS;

- **Thailandia**. Il ministero dell'ICT incoraggia l'uso dei sistemi GNU/Linux, con l'obiettivo a lungo termine di avere fatto migrare al nuovo sistema operativo almeno il 50% dei computer delle agenzie governative;

- **Spagna**. Nel 2002 il Ministero dell'Educazione della Scienza e della Tecnologia dell'Extremadura, ha avviato un progetto di conversione di tutti i computer pubblici dal software proprietario a quello Open Source, prevedendo anche la creazione di una distribuzione specifica chiamata GNU/LinEx. Secondo il ministro Luis Millan Vazquez due sono le motivazioni che hanno portato ad avviare questa piccola rivoluzione nella regione: risparmiare i 30 milioni di euro necessari per pagare le licenze di software proprietari e non dipendere da una singola compagnia. L'obiettivo del progetto è anche quello di fornire un computer ogni sei studenti di scuola primaria ed uno per ogni alunno delle scuole secondarie;

- **Sud Africa**. Nel Gennaio 2003 il governo riconosce i benefici commerciali ed educativi del software open, e raccomanda la creazione di partnership tra aziende, ambienti accademici, industria e istituzioni governative al fine di favorire l'utilizzo del sistema GNU/Linux ovunque gli OSS e i software proprietari abbiano gli stessi vantaggi e svantaggi.

- **Svezia**. Nel Febbraio 2003,l'Agenzia Svedese per Amministrazione Pubblica dichiara che l'OSS presenta in molti casi una qualità che è pari o superiore al software proprietario, e che per questo è opportuno dare al primo le stesse opportunità di quest'ultimo. In quest'ottica ha imposto di adottare standar open e formati di file compatibili con entrambi i sistemi;

- **Gran Bretagna**. Nel 2002 il governo Anglosassone giunge alle stesse conclusioni di quello svedese, promuovendo l'uso di formati di file non proprietari per garantire a tutti l'accesso ai dati, e ponendo l'OSS sullo stesso piano dei sistemi operativi proprietari dal punto di vista delle prestazioni e della qualità.

La cosa straordinaria che si vede in questa panoramica è che si sta diffondendo sempre più una nuova attenzione per il software libero ed Open Source, e che tale attitudine è alimentata dai governi di Paesi molto diversi tra di loro in termini di sviluppo e di avanzamento economico.

Molto interessante a questo punto è vedere quali sono le caratteristiche che possono rendere l'OSS appetibile in particolare nei paesi del Sud del mondo e che ha fatto sì che si sviluppassero intorno al prodotto open numerosi progetti di cooperazione con lo scopo di colmare quel Digital Divide di cui abbiamo parlato nel capitolo1.

2.1.3 L'impatto nei paesi in via di sviluppo

L'impatto dell'Open Source nei paesi in via di sviluppo può essere ancora maggiore che nelle regioni in cui il settore dell'information technology è ben sviluppato. Tra le ragioni principali che ci hanno indotto questa valutazione le più importanti sono:

- Facilità d'accesso ai prodotti software. I programmi che girano sul sistema GNU/Linux possono essere facilmente messi a disposizione di aziende e singoli utenti. E' sufficiente che vengano distribuite copie di CD contenenti il software da utilizzare (cosa assolutamente vietata per le licenze dei programmi proprietari). La distribuzione dei CD può essere fatta da un ente pubblico, così come da un'organizzazione privata, con i soldi provenienti dalle tasse o dai fondi della cooperazione internazionale. Poichè non ci sono costi per ciascuna copia (se non quelli legati alla distribuzione), poichè non ci sono tasse da pagare, l'accesso della popolazione più povera al software può essere possibile praticamente in tutti i paesi;

- Contributo da parte dei paesi sviluppati. I paesi delle regioni più sviluppate potrebbero contribuire praticamente a costo zero al trasferimento del software, aiutando i PVS ad adottare soluzioni Open Source. Poichè infatti non ci sono costi di licenza ed anche le copie sono gratuite, il costo per i paesi che danno il loro contributo è molto basso;

- Accesso diretto al software. I paesi in via di sviluppo possono accedere ai programmi senza dover pagare per questo costose royalties. Solitamente

quei paesi non hanno facile accesso al software; essi possono ottenere il prodotto finito a prezzi alti senza poter mai accedere al processo che ha portato alla generazione di quel prodotto. Al contrario grazie al sistema GNU/Linux possono accedere al codice sorgente, inclusi i suoi dettagli implementativi;

- Possibilità di partecipare allo sviluppo di tecnologie avanzate. Nel mondo Open Source, ogni sviluppatore in ogni parte del mondo ha, almeno in teoria, la possibilità di accedere ad ogni progetto (se ha a disposizione una connessione ad Internet). Di fatto è interessante notare come importanti progetti di free software siano stati condotti in paesi con poca tradizione dello sviluppo di programmi.

In conclusione, possiamo dire che questo tipo di software rispetto a quello proprietario consente di ridurre le disparità tra chi è in grado di accedere alle risorse tecnologiche, e chi invece non ha quella possibilità, e rende più facile per i paesi meno sviluppati non solo accedere ai programmi ma anche contribuire al loro sviluppo.

Vediamo alcuni dei progetti più interessanti sviluppati da paesi in via di sviluppo. Tutti hanno nell'utilizzo di Linux un elemento in comune.

Il primo è stato sviluppato in Brasile specificatamente nella municipalità di San Paolo. Il progetto consiste nella realizzazione di centri informatizzati che forniscano l'uso gratuito di computer e l'accesso ad internet alle zone più povere e periferiche della città. Alla fine del 2003 erano 128 i centri funzionanti in tutto il territorio urbano. Ciascuno di essi fornisce servizi a circa 3000 utenti. Il costo per la realizzazione di ogni centro è stato di 10.000 USD ed il loro accesso è completamente gratuito per la comunità. I servizi offerti sono innumerevoli; si va dai corsi di web-designer ed image processing fino a quelli di giornalismo con una particolare cura ad incoraggiare le attività artistiche in genere. In ogni centro è presente uno staff per il supporto tecnico e la formazione. Uno dei capisaldi del progetto è che dopo la fase di start-up siano gli stessi membri della comunità a fornire il supporto tecnico necessario all'interno di ciascun centro. Dal punto

di vista tecnologico possiamo dire che in ciascun centro c'è un cluster LTSP[4] costituito da un server e 20 nodi diskless connessi tramite una rete a 100MB. Gli applicativi previsti sono Gnome, un browser Web, Gimp, un processatore di testo e un programma per la gestione dei fogli di calcolo. Il progetto si è rivelato di grande successo. Ha permesso di confutare un luogo comune sulla difficoltà del sistema GNU/Linux, che invece è risultato molto semplice da imparare per i ragazzi in età scolare ed in generale per tutte le persone con un'istruzione anche minima. Ciascun centro ha il suo Consiglio formato ed eletto dagli abitanti della comunità locale, con il compito di stabilire le regole per l'uso del centro stesso.

Il secondo progetto preso in esame per il nostro studio è stato sviluppato in Tajikistan. Questo paese è il più povero tra tutti quelli dell'ex Unione Sovietica, e tra i vari problemi che ha rispetto l'accesso alle tecnologie c'è quello della lingua; non esistono cioè traduzioni di software nell'idioma Tajik. Il progetto finanziato dal governo è stato pensato proprio per il superamento di questo problema. E' stato formato un gruppo di sviluppatori volontari cui è stato dato il compito di tradurre programmi utente nella loro lingua nativa. Dal punto di vista tecnico possiamo dire che i volontari hanno lavorato su 30 computers assemblati in US e configurati con una prima versione in Tajik di Mandrake 9.1[5]. Questi computer sono stati usati principalmente dall'equipe di volontari e da studenti di alcune scuole del paese. Una volta completata l'opera di traduzione i centri allestiti con i PC diventeranno accessibili ai cittadini. Questo progetto di dimensioni abbastanza ridotte ci consente comunque di mettere l'accento su un aspetto fondamentale di Linux e cioè il fatto che questo sistema operativo è disponibile in un numero di lingue che è più del doppio rispetto a quelle del prodotto più aggiornato della Microsoft, ad esempio, e questo proprio grazie al contributo di un numero molto grande di sviluppatori ed appassionati sparso in tutto il mondo che possono accedere al codice sorgente.

Il terzo ed ultimo progetto è stato sviluppato in Laos per aiutare le comunità agricole di Phon Kham e di altri quattro villaggi ad ottenere le informazioni sui

[4]Del sistema Ltsp parleremo nel capitolo3

[5]Una distribuzione GNU/Linux

prezzi del mercato agricolo ed aumentare, così, le entrate finanziarie dei villaggi. Il Jhai Remote Village IT system (ancora in fase di sviluppo), è stato disegnato per fornire il supporto per le chiamate locali e facilitare la connessione ad Internet. I villaggi sono situati in una valle in cui sono completamente assenti l'elettricità e il servizio telefonico ed in cui l'accesso alla telefonia cellulare è limitato dalle montagne intorno. Oltre a questi problemi, la tecnologia ICT doveva affrontare anche la questione delle condizioni atmosferiche assolutamente avverse a causa dei monsoni, delle alte temperature e dell'altissimo tasso di umidità. Il sistema Jhai è basato su un computer embedded alimentato da poca corrente. Su di esso è installata una versione ottimizzata e tradotta di KDE, chiamata LaoNux. Il computer del villaggio si collega ad Internet con un sistema wireless che comunica con un'antenna fissata sul tetto di una delle case, che invia il segnale ad un access point alimentato ad energia solare. Il computer funziona con 12 watts di potenza, fornita dall'energia cinetica prodotta da una bicicletta. E' stata scartata l'energia solare perché più costosa e per la lunghezza del periodo delle piogge. Nel progetto ha una grandissima importanza la crescente disponibilità di OSS che supportano i sistemi wireless. In più è stato scelto il sistema operativo GNU/Linux perché il sistema Jhai è stato pensato per funzionare 10 anni, e solo un software Open Source può disporre di persone che sanno come fare la manutenzione per tutto quel tempo.

Questi tre esempi ci danno una misura molto chiara di quanto il sistema GNU/Linux sia importante in tutti quei progetti che hanno come obiettivo la riduzione dello scarto digitale nei paesi meno sviluppati. I casi che abbiamo esaminato sono particolarmente interessanti perché sono molto diversi tra di loro e tutti mettono in evidenza aspetti partiolari del sistema operativo Open Source che vanno ben oltre il semplice risparmio in termini di costi.

2.2 Esempi di Trashware

Il Trashware è un fenomeno molto nuovo ed in grande espansione in Italia, sotto la spinta di associazioni di volontariato e di appassionati utilizzatori del sistema

GNU/Linux. Il Golem, come già detto, è stato il gruppo che per primo ha iniziato un progetto integrale e strutturato per il ricondizionamento di computer obsoleti. Il suo esempio è stato poi seguito da numerosi altri gruppi sparsi in tutta Italia.

2.2.1 Il Golem

I colleghi del Golem hanno, tra le altre cose, tracciato le linee guida da seguire per realizzare un progetto di riutilizzo di hardware[25]. Riducendo il problema ai minimi termini sono state individuate tre attività principali che richiedono attrezzature, competenze ed esperienze sostanzialmente diverse e specifiche. Per questo tre sono anche i soggetti protagonisti del progetto. Le attività principali sono la **raccolta** dei computer dismessi, il **ricondizionamento** con Linux e Software Libero e la **redistribuzione** dei computer a chi può utilizzarli con vantaggio. Per ciascuna delle attività è stato individuato il soggetto più competente, attrezzato e predisposto. Il soggetto chiamato a recuperare fisicamente i computer dismessi dall'azienda è il cosiddetto **Soggetto Collettore**. La presenza di quest'ultimo è, ovviamente, meno importante quando si devono gestire piccole quantità di computer, ma diventa particolarmente importante nel momento in cui l'attività di Trashware cresce. Più i volumi di materiale da raccogliere crescono più risulterà di fondamentale importanza l'apporto di un soggetto che abbia attrezzature ed organizzazione logistica. Più di tutto è necessario che il Soggetto Collettore sia già il riferimento istituzionale per chi smaltisce i computer, perché in questo modo anche chi non sia a conoscenza del progetto, contatterà comunque il soggetto giusto. Sembra in questo senso opportuno coinvolgere l'azienda municipalizzata o consortile che ritira i rifiuti ingombranti, almeno nell'aspetto logistico, organizzativo ed informativo. In più diventa davvero essenziale la presenza di un soggetto dotato di adeguati mezzi di trasporto sia per raccogliere i computer recuperati, sia per gestire opportunamente tutto il materiale inutilizzabile. La parte informativa è molto importante non solo per la pubblicità all'iniziativa, ma anche perché è importante che le aziende conoscano il modo corretto di dismettere macchine usate, affinchè si possa evitare il fenomeno della "cannibalizzazione" (di cui ab-

baimo parlato all'inizio di questo capitolo) e che i PC siano tenuti tanto tempo ammassati prima di essere dismessi. Se le aziende dismetterrano correttamente i loro computer, allora si avranno enormi facilitazioni nel recupero e nel ricondizionamento. Il soggetto collettore è importante anche perché sa come trattare un tipo di 'rifiuto' particolare come il RAEE. L'azienda consortile diventa l'interfaccia unica nei confronti di chi dismette. Essendo questa la sua funzione naturale, può ricoprirla senza sforzo, essendo già attrezzata per farlo (per esempio con call center, numero verde, dipendenti formati a dialogare con chi dismette). Quindi il meccanismo della raccolta può essere il seguente: chi dismette prende contatto con il Collettore, che si informa sulla presenza di macchine riutilizzabili. A questo punto, se le macchine sono visionabili, il Soggetto Riqualificatore sarà chiamato per scegliere i computer riutilizzabili. Si applicheranno, allora, due metodologie di ritiro differenti: i computer non recuperabili saranno ritirati come rifiuti, e i computer ancora buoni prenderanno la via del riutilizzo. Chi dismette dovrà pagare solo per le macchine non più riutilizzabili.

Il **Soggetto Riqualificatore** è il secondo soggetto che viene previsto nel progetto di Golem. Esso è il fulcro attorno al quale ruota tutta l'attività. Questo soggetto non è nè solo un'azienda nè una semplice associazione di volontariato. perché, per installare vecchi computer, si ha bisogno di competenze superiori e tecnici più competenti e preparati rispetto a quanto può bastare per un computer nuovo. Ricorrendo ad un'azienda i costi di recupero lievitano enormemente, perché essa dovrà impiegare sistemisti molto competenti solo per installare linux sulle vecchie macchine. Il soggetto riqualificatore non può neanche essere solo un gruppo di volontari che non possono garantire la necessaria 'produttività' del recupero. La scelta del Golem è ricaduta, allora, sui LUG[6]. Nei gruppi di utenti Linux infatti ci sono persone che per la loro competenza e la loro passione, possono risolvere numerosi incovenienti hardware e software; per contro però possono offrire un apporto che è comunque saltuario. Attualmente è molto acceso il dibattito su quale debba essere lo status legale migliore per un gruppo che voglia dedicarsi ad un'attività di Trashware, che dovrà riuscire a conciliare le alte com-

[6]Linux Users Group

petenze informatiche necessarie con la bassa spesa sostenibile. Una possibilità è che sia una cooperativa di lavoro a svolgere il ruolo del riqualificatore. Essa effettuerebbe la raccolta di rifiuti per conto dell'azienda consortile, che invece funzionerebbe solo da collettore. La coperativa di lavoro avrebbe sicuramente le strutture e la forza lavoro necessarie e verrebbe affiancata dai volontari del LUG per le competenze informatiche.

Infine il terzo soggetto è il **Soggetto Ridistributore**, che potrebbe essere chi conosce e coordina le associazioni di volontariato presenti sul territorio, o chi gestisce e coordina gli aiuti alle associazioni o a gruppi di cittadini (anziani o studenti, per esempio). Quest'ultimo soggetto dovrebbe conoscere le esigenze di chi richiede un computer ricondizionato in modo che si possa operare sulle applicazioni utente da inserire nelle diverse macchine affinchè i richiedenti ne traggano massimo vantaggio. Questo soggetto dovrà anche definire i criteri di redistribuzione e gestire la consegna dei PC. Un criterio base che Golem considererà come preferenziale è fornire i computer a tutte quelle realtà che possono contribuire in qualche modo alla diffusione della conoscenza e dell'uso delle risorse informatiche e comunicative, in modo da contribuire a quel processo di alfabetizzazione informatica che è alla base della missione del gruppo. Al crescere del numero di computer recuperati crescerà sempre più la portata del progetto e le conseguenze che questo avrà sulla società. Quando i volumi trattati saranno tali da non poter essere smaltiti dai volontari del Gruppo Linux, allora sarà necessario avviare (sempre tramite il soggetto ridistributore) corsi di formazione che possano consentire ad altri di aggiungersi al progetto e al Gruppo Linux di autofinanziarsi.

2.2.1.1 Progetti

Il Golem fornisce computer ad associazioni di volontariato presenti sul territorio empolese-valdelsa, ma anche a popolazioni di paesi in via di sviluppo. Nel Gennaio 2003 sono stati inviati al popolo Saharawi sei computer, mentre in collaborazione con ISF Trento, nell'Aprile dello stesso anno, sono stati inviati tre computer in una scuola in Somalia. Oltre ai servizi offerti alle associazioni e ai progetti di cooperazione allo sviluppo, Golem ha in cantiere progetti futuri che

riguardano soprattutto la possibilità di tirocinii formativi per giovani ed interventi nelle scuole. La scuola investe sempre più sull'informatica, che ormai è presente ad ogni grado ed in ogni indirizzo. Quello che accade è che si trascurano le effettive possibilità di accesso degli studenti ai computer al di fuori dell'ambito scolastico. Spesso i programmi applicativi sono non-liberi e con costi di acquisto proibitivi per molti studenti e famiglie; non sempre inoltre sono previsti spazi di approfondimento e utilizzo di tali strumenti all'interno delle strutture scolastiche al di fuori dell'attività frontale. Altro problema è l'estrema limitazione dell'offerta formativa dal punto di vista dell'hardware, a causa dei costi elevati per l'acquisto di PC. In questo campo il Trashware potrebbe essere di grande aiuto ed in effetti come vedremo più avanti, vi sono in corso molti progetti che hanno lo scopo di recuperare hardware obsoleto per consegnarlo alle scuole.

2.2.2 Progetto Lazzaro

Altro esempio di progetto di Trashware molto attivo è il **Progetto Lazzaro**, un'iniziativa internazionale promossa dalla EdOsNet.org[7] che si propone di favorire il riuso dei Pc dismessi dalle aziende e dalla Pubblica Amministrazione nell'ambito della scuola primaria e secondaria.

Il progetto Lazzaro ha una serie di finalità tra le quali:

- il recupero del patrimonio hardware detenuto dalla Pubblica Amministrazione ed erroneamente considerato obsoleto;
- il sostegno alla diffusione dell'uso nella Scuola del software gratuito, con particolare attenzione a quello Open Source: Linux e OpenOffice.org;
- il sostegno alla diffusione dell'uso nella scuola di prodotti software che generano, esportano e importano file nel formato aperto XML;

[7]La Educational Opensource Network Organization - in sigla EdOsNet.org - è una organizzazione di carattere internazionale la cui missione consiste nel sostenere la diffusione del know-how tecnico per costruire delle reti didattiche a basso costo basate sui server Linux

- la diffusione della cultura del riuso di componenti elettronici ancora tecnicamente efficienti e/o rigenerati;
- la tutela dell'ambiente, mediante il riuso di componenti elettronici ancora perfettamente funzionanti e lo smaltimento in modalità controllata dei rifiuti elettronici classificati nocivi.

Nel progetto Lazzaro la società Didasca Technologies propone alle scuole l'acquisto di hardware rigenerato provvisto del sistema GNU/Linux. Vengono fornite due soluzioni differenti a seconda che la scuola desideri una rete di macchine oppure PC Stand-alone. In entrambi i casi il prodotto fornito alle scuole è ottenuto mediante l'assemblaggio di componenti nuovi di fabbrica e di componenti rigenerati provenienti dal disassemblaggio di apparecchiature informatiche dismesse eseguito da aziende terze. Per le apparecchiature è garantita l'assenza di software proprietario, ciò significa che le scuole non devono pagare per esse il canone annuale forfettario per l'uso dei prodotti Microsoft. Grazie ad una convenzione stipulata con il MIUR nel Luglio 2003 per la fornitura di beni e servizi informatici, le scuole possono acquistare le macchine ricondizionate nell'ambito del progetto Lazzaro a prezzi ridotti.

2.2.3 Prodigi-progetto Tunisia

Altro esempio di progetto di riuso di materiale hardware obsoleto, è quello sviluppato da Prodigi in due città della Tunisia[28].

Il gruppo Prodigi è nato in Italia con l'obiettivo di creare un organismo capace di sviluppare strategie per combattere quell'esclusione digitale di cui più volte abbiamo parlato in questa tesi di laurea, e in grado di aprire la strada a nuovi percorsi di sviluppo autonomi e nel rispetto delle diversità culturali.

Come detto il progetto in questione è stato sviluppato in Tunisia nell'Agosto del 2003 in collaborazione con la ONG Alisei e la Union Tunisienne de Solidarietè Sociale (UTSS), ed ha permesso di installare due laboratori informatici nella città di Gabes e nell'oasi di Kerchaou, nel Sud-Est della Tunisia. Oltre a questo, ha permesso di formare 30 persone all'utilizzo delle tecnologie informatiche. I

fondi necessari alla realizzazione dell'intervento sono stati raccolti con attività di autofinanziamento da parte di Prodigi e grazie all'appoggio anche finanziario delle due ONG partner.

Sono stati realizzati tre corsi di formazioni, per rispondere alle esigenze delle comunità locali e per garantire la sostenibilità dei laboratori al termine della fase di start-up del progetto. Il primo corso ha formato dei mediatori, cioè delle persone che facessero da tramite tra le tecnologie e tutte quelle persone che non intendono imparare ad usare il computer ma che possono averne bisogno occasionalmente.

Il secondo corso, per formatori, era rivolto a tutte quelle persone (insegnanti soprattutto) già attive nella cooperazione o nella formazione. L'obiettivo era quello di affidare la trasmnissione delle competenze informatiche a individui il cui ruolo sociale istituzionale era già quello di trasmettere conoscenza.

Infine il terzo corso è stato ristretto ad un gruppo di sole quattro persone allo scopo di fornire loro le competenze necessarie per la manutenzione dei laboratori informatici, e per rendere i centri il più possibile autosufficienti.

I due laboratori, allestiti con computer dismessi da enti pubblici e privati ma ancora funzionanti, sono stati concepiti non solo come luoghi di accesso alle nuove tecnologie, ma come spazi di pubblica utilità aperti alle comunità locali, luoghi di aggregazione e formazione. In ciascun laboratorio sono stati installati 5 computer ed una stampante connessi in rete. Su ogni macchina è stato installato il sistema operativo GNU/Linux ed alcuni degli applicativi più comuni come Mozilla, OpenOffice e Gimp. Particolare attenzione è stata messa sull'utilizzo della lingua Francese e Araba per le tastiere, il sistema operativo e il software installato.

2.2.4 ISF-Roma Progetto Kosovo

Il progetto ha visto impegnata ISF Roma nel 2002, ed ha avuto come obiettivo principale l'allestimento di un laboratorio di informatica per l'istituto tecnico

Anton Cetta di Skenderaj in Kosovo. Lo scopo era quello di fornire la scuola di uno strumento didattico come supporto all'insegnamento dell'Informatica.

Il progetto è nato dalla collaborazione tra l'associazione PlanetNoProfit di Milano e il Gruppo Sprofondo Imperia, già da tempo impegnate sul territorio e che hanno messo a disposizione la loro rete di contatti e un supporto logistico già collaudato.

Per il laboratorio è stato usato esclusivamente software libero ed in particolare è stato scelto il sistema operativo GNU/Linux e la suite di applicazione per ufficio OpenOffice.

Nella fase preliminare ISF Roma è stata impegnata in una serie di sopralluoghi per valutare la fattibilità del progetto e prendere contatti con la comunità locale. Sono stati, inoltre, realizzati questionari per gli studenti ed i docenti della scuola per comprendere che tipo di intervento effettuare e come preparare e strutturare i corsi. Tramite questo strumento è stato possibile comprendere in particolr modo, il livello di conoscenza della lingua Inglese e dei concetti di Informatica di base.

L'hardware utilizzato nel progetto è stato messo a disposizione dall'azienda Metropolis del gruppo F.S. S.p.A., mentre SuSE Italia ha fornito i pacchetti GNU/Linux SuSE 7.1 che sono stati installati.

Dal punto di vista tecnico il laboratorio è stato dotato di una rete di 12 computer Intel Celeron 266 e 3 stampanti laser donati da Trenitalia. Sui PC è stato installato GNU/Linux, distribuzione SuSE 7.1. A causa della discontinuità della fornitura di energia dalla rete elettrica i l laboratorio è stato alimentato attraverso un gruppo elettrogeno dedicato (8kW) acquistato da ISF Roma e donato alla scuola. I computer sono interconnessi tramite un HUB, su rete Ethernet a 100Mbps.

ISF ha poi curato 2 cicli di corsi di alfabetizzazione informatica di 2 settimane ciascuno, sia per gli studenti che per i professori, che hanno riguardato i principi di funzionamento dei calcolatori, della rete e di alcuni pacchetti applicativi. Sono stati tenuti 3 corsi al giorno della durata di 3 ore ciascuno a circa 25 studenti per corso. La lezione veniva tenuta in Inglese da un volontario ISF per volta e tradotta dai docenti della scuola, mentre gli altri volontari facevano attività di supporto tra

i tavoli. Grande attenzione è stata messa nell'istruire i docenti sull'architettura e la manutenzione dei PC e dell'intera rete, per garantire il raggiungimento di una reale autosufficienza della scuola nell'utilizzo del laboratorio. Sono state inoltre implementate e spiegate procedure di ripristino in caso di inconvenienti causati dalle deficienze nell'erogazione di corrente.

Per dare continuità nel tempo al suo intervento ISF Roma ha anche previsto un servizio di assistenza on-line per monitorare la stato di funzionamento del laboratorio, e che ha permesso ai volontari di ISF di mantenere quotidianamente un ruolo attivo e diretto nella gestione del laboratorio stesso, anche dopo aver terminato la fase di implemantazione in loco.

Abbiamo voluto brevemente descrivere questi progetti, perché molto diversi tra loro e molto esplicativi di come sia possibile ricondizionare computer dismessi nelle attività più varie. In più tramite esperienze concrete hanno cosentito ulteriormente di mettere in evidenza alcune delle caratteristiche del Trashware e di dare un'idea delle grandi implicazioni che un'attività del genere può avere.

Dal prossimo capitolo, entriamo nel dettaglio del nostro progetto di tesi, dapprima attraverso uno studio delle piattaforme software utilizzate nei nostri esperimenti, e successivamente passando a decrivere il nostro test-bed e i risultati e le conclusioni a cui siamo pervenuti.

Capitolo 3

I Cluster

La parola "Cluster" è usata nel computer networking per riferirsi genericamente alla condivisione delle risorse di calcolo tra i nodi della rete di macchine. Tipicamente un cluster integra le risorse di due o più computer (che potrebbero in altra maniera funzionare separatamente) che lavorano insieme al fine di aumentare la potenza di calcolo o l'affidabilità dell'intero sistema. L'idea che è alla base del clustering consiste nel distribuire il carico su tutti i computer disponibili, usando le risorse che sono libere sulle macchine che compongono il sistema. Il singolo computer è l'unità base di un cluster ed è chiamato "nodo". I cluster possono crescere in dimensioni con l'aggiunta di più macchine, ed ovviamente hanno una potenza ed una velocità dipendente da quella dei loro nodi e dalla velocità della rete. In più un sistema di clustering deve saper utilizzare al meglio l'hardware anche al variare delle condizioni. Questo diventa una vera sfida quando il cluster è composto da tipi di hardware differenti (cluster eterogeneo), e quando la configurazione cambia in modo impredicibile con macchine che si aggiungono o abbandonano il cluster ed il carico di lavoro non può essere predetto. Sostanzialmente ci sono tre tipi di clusters, Fail-over, Load-balancing ed High Performance Computing, con i primi due che sono probabilmente i più diffusi.

- **Fail-over Cluster**: consiste di due o più computer interconnessi, il funzionameto delle macchine è continuamente monitorato, e quando uno dei due host smette di funzionare l'altra macchina subentra;

- **Cluster con load-balancing**: è un sistema nel quale quando arriva una richiesta di lavoro questa viene inviata alla macchina con meno carico;
- **HPC Cluster**: i computer sono configurati per fornire specialmente a centri di elaborazione dati prestazioni estremamente alte; le macchine suddividono i processi di un job su più macchine, al fine di guadagnare in prestazioni. La peculiarità saliente di questo tipo di cluster è che i processi sono parallelizzati e che le routines che possono girare separatamente saranno distribuite su macchine differenti invece di aspettare di essere eseguite una dopo l'altra.

Il nostro studio si è focalizzato in particolare su due sistemi di cluster molto diversi tra di loro: **openmosix** ed **LTSP**. Molto diversi perché sono nati e si sono sviluppati per due scopi differenti. Openmosix è un High-Performance Computing Cluster perciò è stato pensato per fare calcoli fornendo alte prestazioni, mentre LTSP è un sistema Thin Client molto più orientato del primo alle normali applicazioni utente. In questo capitolo analizzeremo le caratteristiche più importanti di ciascuno dei due e ne evidenzieremo le differenze salienti.

3.1 OpenMosix

.

OpenMosix è un software che consente di trasformare in un cluster una rete di computer interconnessi e dotati di sistema operativo GNU/Linux. Esso automaticamente bilancia il carico tra i diversi nodi, e questi ultimi possono aggiungersi

Figura 3.1: logo openMosix.

o lasciare il cluster senza che ci sia interruzione del servizio. Il carico è distribuito tra i nodi sulla base della velocità della loro CPU.

OpenMosix è una patch del Kernel Linux; questo significa che un'applicazione utente, i file e le altre risorse continuano a funzionare senza alcun cambiamento. L'utente casuale non noterà la differenza tra un sistema Linux semplice ed uno con openMosix; per questi, l'intero cluster funzionerà come un normale sistema GNU/Linux molto veloce. OpenMosix non ha un controllo centralizzato di tipo master/slave. Ogni nodo agisce come un sistema autonomo, le sue decisioni sono indipendenti e si basano su una conoscenza parziale degli altri nodi. Questa struttura consente di avere una configurazione molto dinamica nella quale nodi possono essere aggiunti o rimossi in modo trasparente, consentendo così un'agevole scalabilità del sistema e assicurando un funzionamento ottimale indipendentemente del numero dei nodi.

Per garantire la scalabilità del cluster, si adotta uno schema di diffusione delle informazioni sullo stato di ogni nodo di tipo probabilistico: ad intervalli regolari, ogni macchina del cluster invia ad un sottoinsieme casuale di altri nodi informazioni sulle risorse che ha a disposizione, mantenendo contemporaneamente in una tabella gli ultimi valori ricevuti. Questo scambio avviene ogni secondo, utilizzando il protocollo UDP.

Il funzionamento di openMosix si basa su un sistema di rilevazione dell'utilizzo delle risorse, un insieme di algoritmi adattativi per la gestione delle risorse ed un sistema di migrazione dei processi trasparente. Vediamoli nel dettaglio.

Il **sistema per la rilevazione dello stato del cluster** è quello per cui ogni nodo è in grado di conoscere il reale stato di utilizzo delle risorse del sistema. Ciascuno, come detto, invia le informazioni sul suo stato di carico tramite UDP ad un numero di nodi scelti in modo casuale, e mantiene una tabella con le notizie che riceve dagli altri nodi. Le informazioni vengono aggiornate molto frequentemente, tipicamente ad ogni system call oppure ogni secondo. La creazione di un nuovo processo invalida tutte le informazioni visto che modifica il carico sui nodi.

Gli **algoritmi per la condivisione delle risorse** sono gli strumenti che consentono ad openMosix l'ottimizzazione e il bilanciamento del carico. Sulla

base delle informazioni che i nodi si scambiano, un algoritmo di **load balancing** tenta di ridurre la differenza di carico tra le coppie di nodi, facendo migrare i processi dai nodi più carichi a quelli con più risorse disponibili. Questo sistema è decentralizzato e tutti i nodi utilizzano lo stesso algoritmo; la riduzione delle differenze di carico viene eseguita da coppie di nodi in modo autonomo. L'algoritmo determina le potenzialità di ogni nodo sulla base del numero delle CPU disponibili e della velocità della singola CPU. L'algoritmo è in grado di rispondere dinamicamente a variazioni del carico. Il **load balancing** è affiancato da un altro algoritmo fondamentale per openMosix, che è quello di **memory ushering**, a cui è demandato il compito di bilanciare il carico di memoria dei nodi. Serve per evitare che ci sia un drastico peggioramento delle prestazioni a causa della mancanza di memoria libera. Quando un nodo inizia ad usare il file di swap, l'algoritmo va in esecuzione e tenta di migrare il processo verso un altro nodo con sufficiente memoria libera, anche se così facendo si dovesse ottenere un bilanciamento del carico non ottimale. Il bilanciamento della memoria ha la priorità sul bilanciamento del carico della CPU. Oltre a questi algoritmi, ce ne è un altro che serve a determinare su quale nodo un processo dovrebbe essere eseguito. Il modello matematico alla base di questo algoritmo deriva dalla scienza economica. Determinare il nodo ottimo non è una cosa semplice. La complicazione principale è che le risorse di ogni nodo sono eterogenee e non direttamente confrontabili, visto che neanche sono misurate nella stessa unità di misura. L'algoritmo utilizzato da openMosix è interessante in quanto tenta di ridurre queste differenze basandosi su principi economici di mercato. L'idea è, infatti, di associare a ciascuna risorsa un "costo" omogeneo e migrare i processi dove il costo è più basso. Questa strategia basata sull'economia più che sulle tecnologie informatiche, consente di sviluppare algoritmi che ottimizzano l'allocazione delle risorse nei nodi.

L'ultimo elemento che caratterizza fortemente il funzionamento di openMosix è il **sistema di migrazione dei processi**. Il meccanismo di Preemptive Process Migration (PPM) consente la migrazione **trasparente** dei processi, e possiamo dire che è quello che consente a openMosix di funzionare. Ogni processo ha un unico nodo di origine (UHN-Unique Home Node) in cui è stato creato; normal-

mente questo è il nodo da cui l'utente ha lanciato il processo. Finché l'utilizzo delle risorse del nodo rimane al di sotto di una certa soglia, il processo è vincolato a rimanere sul nodo home; quando tale soglia viene superata, alcuni processi possono essere migrati verso altri nodi. I processi che vengono mandati ad un altro nodo sfruttano le risorse locali, ma continuano a interagire con l'ambiente dell'utente tramite l'UHN. L'obiettivo totale è quello di massimizzare l'efficienza dell'utilizzazione delle risorse dell'intera rete di calcolatori. Per implementare PPM occorre dividere i processi in due parti:

- **la parte utente**: è la parte che può essere migrata; in termini openMosix, si chiama anche "remote". Contiene anche l'eseguibile del programma, il suo stack, i dati, le mappe di memoria e i registri che competono al processo;
- **la parte di sistema**: è la parte che rimane nel nodo d'origine; si chiama per openMosix anche "deputy". E' vincolata a rimanere nell' home node e non può essere migrata. Contiene una descrizione delle risorse che il programma sta utilizzando e un sistema di system call per conto del processo remoto.

I processi migrati verso un nodo non sono accessibili dagli altri processi che girano su di esso. L'unica cosa che un utente remoto può fare è forzare il processo a lasciare il nodo e migrare altrove.

Il processo è in esecuzione immediatamente dopo il suo trasferimento, anche se non ha ancora tutte le pagine di memoria necessarie; le pagine di cui il processo ha bisogno sul nuovo nodo vengono trasferite fino a che sul nodo remoto non è stato ricostruito il working set del processo. Grazie alla tecnologia di "Demand paging" le pagine vengono trasferite solo quando sono effettivamente necessarie, riducendo così la dimensione del resident set.

La comunicazione tra deputy e remote avviene grazie a un protocollo molto leggero basato su UDP che usa porte oltre le prime 1024 (cioè non di sistema). L'interfaccia tra deputy e remote è ben definita, quindi è possibile intercettare tutte le interazioni tra le due parti e inviarle sulla rete. In tal modo è possibile captare tutte le system call eseguite dal processo e quelle che non dipendono dal nodo sono eseguite in remoto senza ulteriori ritardi, mentre quelle che dipendono

dal nodo di origine vengono trasferite al deputy che le esegue per conto del remote. Il deputy invia quindi i risultati a quest'ultimo che riprende l'esecuzione del codice utente. E' in questo modo che si realizza la trasparenza nella migrazione dei processi. La divisione dei processi in deputy e remote comporta però un overhead nell'esecuzione delle system call e nella notifica dei segnali.

L'I/O sui file eseguito via rete è un notevole collo di bottiglia, rallenta le prestazioni del nodo ed intasa la rete con traffico inutile, per questo è preferibile tenere tutte le operazioni di I/O locali. Così al contrario di tutti i network file system esistenti (per esempio NFS), che trasferiscono i dati attraverso la rete fino al nodo remoto in cui si trova il processo, il cluster openMosix tenta di migrare il processo nel nodo in cui il file effettivamente risiede.

Questo ha richiesto l'utilizzo di un file system proprio: openMosix File System (oMFS). L'oMFS è un filesystem del cluster e consente ad ogni nodo di accedere al filesystem di qualunque altro nodo come se fosse montato localmente; si occupa solo di rendere accessibili i filesystem ma non di utilizzarli in modo efficiente. All'oMFS (che verrà abbandonato a partire dal kernel 2.4.26) è stato aggiunto un Direct File System Access (DFSA). Il DFSA è stato pensato per ridurre l'overhead dovuto all'esecuzione di operazioni di I/O da parte dei processi migrati. In aggiunta a questo file system per prevenire il problema di intasamento della rete, sono stati modificati gli algoritmi di load balancing in modo che un processo che deve eseguire molte operazioni sui file (processo I/O bound), venga migrato verso il nodo su cui deve compiere le suddette operazioni. I processi I/O bound hanno quindi maggiore flessibilità nel migrare dall'home node, per garantire il migliore utilizzo delle risorse dell'intero cluster.

Vediamo in conclusione quali sono i pro ed i contro di questo sistema di clustering.

I Pro:

- Trasparenza: il sistema di migrazione dei processi è totalmente trasparente sia all'utente sia alle applicazioni; i processi standard di Linux vengono migrati verso i nodi più performanti ogni volta che è necessario;

- Nessuna ricompilazioni del codice per le applicazioni: è possibile usare le normali applicazioni che funzionano su una singola macchina senza doverle riscrivere o ricompilare;

- Nessuna manutenzione aggiuntiva: una volta che il cluster è stato configurato tutte le operazioni sono automatiche e non richiedono interventi da parte dell'utente.

I Contro:

- Le applicazioni composte da un solo processo non traggono vantaggi dall'uso del cluster, eccetto ovviamente per il fatto che quel processo sarà migrato sul nodo più performante;

- Non tutti i processi possono essere migrati: alcuni tipi di processi sono vincolati al proprio home node e non possono essere migrati, come i processi con necessità real-time e i processi che hanno accesso diretto ai dispositivi di I/O e soprattutto i processi a memoria condivisa;

- La minima unità di lavoro di openMosix è il processo: questo significa i thread non vengono migrati ad eccezione dei Java Green Thread e degli FSU Pthread[23].

Il fatto che openMosix non consenta la migrazione dei processi a memoria condivisa è stato uno dei problemi più grandi che abbiamo dovuto affrontare nelle nostre ricerche e nei nostri test. La nostra intenzione, come detto, è sempre stata quella di usare openMosix per scopi diversi rispetto a quelli per cui era stato progettato (HPC), cioè per fare in modo che le normali applicazioni utente traessero vantaggio dal girare su un cluster piuttosto che su un singolo computer. Il problema, per noi, nasce nel momento in cui abbiamo verificato che molte delle applicazioni che volevamo far funzionare erano a memoria condivisa.

Vediamo ora come funziona la condivisione di memoria nel sistema GNU/Linux e su quale soluzione sperimentale sta lavorando la comunità openMosix per ovviare a questo inconveniente.

3.1.1 La memoria condivisa

Sotto Linux la condivisione della memoria è gestita utilizzando i meccanismi di comunicazione tra processi implementati nello standard UNIX System V; esso consente a due o più processi di accedere ad alcune strutture di dati comuni inserendole all'interno di un segmento di memoria. La memoria condivisa in Linux è una porzione di memoria contigua accessibile o in lettura o in scrittura da un certo numero di processi che abbiano gli appropriati permessi di accesso. Ogni porzione di memoria condivisa ha un identificatore unico all'interno del sistema (numero intero) denominato chiave. Tale chiave è lo strumento che i processi utilizzano per accedere alla memoria condivisa.

L'accesso avviene nel seguente modo: un processo apre la memoria condivisa attraverso la chiave e l'attacca al proprio spazio di indirizzamento, da dove la staccherà quando non ne avrà più bisogno. Per la prima operazione ci sarà bisogno di una chiamata di sistema apposita. La chiamata per ottenere dal Kernel una porzione di memoria centrale da utilizzare in modalità condivisa è la **shmget()**. Se la chiamata ha successo, allora la funzione shmget restituisce un intero con cui il Kernel identifica univocamente l'area di memoria condivisa, mentre restituisce -1 in caso di fallimento. La chiamata di sistema opposta a questa è la **shmdt()**, essa rimuove la regione di memoria dallo spazio di indirizzamento del processo.

Per poter utilizzare la memoria condivisa, dovremo collegarla allo spazio di indirizzi di ciascun processo che ne abbia bisogno. La chiamata di sistema necessaria per fare questo è **shmat()**. Se la chiamata ha successo, allora la funzione restituisce il puntatore alla prima locazione di memoria condivisa. In questo modo si potrà accedere alla memoria condivisa come se fosse un semplice array.

Per eseguire operazioni di controllo su una porzione di memoria condivisa usiamo la **shmctl()**.

Ora che abbiamo visto come la memoria condivisa è gestita nel sistema GNU/Linux andiamo ad analizzare la soluzione (ancora in fase di sperimentazione) che viene proposta alla comunità openMosix.

3.1.2 Migshm

Migshm (Migration of Shared Memory) è una patch del kernel addizionale a quella di openMosix. E' stata creata da 5 studentesse di ingegneria dell'Università di Pune in India, per cercare di risolvere uno dei problemi maggiori di openMosix, cioè, come detto, consentire la migrazione di processi che condividono la memoria. La patch è ancora in fase di sperimentazione, ed ancora non si sa quando verrà inserita stabilmente nella patch ufficiale openMosix.

Migshm consiste sostanzialmente di quattro moduli. Il primo è quello più importante visto, che consente ai processi a memoria condivisa di essere presi per la migrazione in ogni momento durante la loro esecuzione, sia prima che dopo la chiamata della shmget() che li attacca alla regione di memoria condivisa. Per fare questo, si deve innanzitutto fare in modo che oM riconosca tali processi e li consideri come migrabili. A tale scopo è necessario fornire a ciascuno di questi processi un unico identificatore e un flag che dica al cluster se il processo è fortemente o debolmente legato alla regione di memoria condivisa. Oltre a questi è necessario registrare anche alcuni attributi della memoria condivisa stessa come l'identificatore unico necessario per indicare la regione di memoria, il conteggio degli accessi alla regione. Tali informazioni sono inizializzate quando viene invocato shmget() dal processo e quando questi accede alla regione di memoria. Questo modulo viene chiamato di "Migrazione dei processi a memoria condivisa" e nasce per superare sostanzialmente due problemi che openMosix presenta nella gestione della memoria. Il primo problema è che oM gestisce la memoria solo dei processi a memoria non condivisa. Quando un processo migra su un nodo remoto, la sua mappa di memoria viene distrutta e ricreata li dove il processo è migrato. Questo modo di gestione è insufficiente per i nostri scopi. Quando proviamo a migrare un processo shared memory, non possiamo distruggere le pagine di memoria condivisa sul nodo home perché attaccate ad esse abbiamo degli altri processi. Il secondo problema è legato alla migrazione di due processi che condividono la memoria su un nodo. Quando questi passano su un altra macchina del cluster openMosix ricrea nuove pagine di memoria per ciascuno di essi

mentre essi dovrebbero condividere la memoria anche sul nodo remoto. Il primo modulo di migshm garantisce che le system calls di Sys V IPC possano essere fatte trasparentemente da remoto quando i processi migrano.

Il secondo modulo è il cosiddetto "Communication Module". Quando un processo migra su un cluster openMosix è richiesto un canale di comunicazione tra la parte deputy e la parte remote, questo canale non può essere utilizzato, però, quando due processi su due nodi devono comunicare tra di loro. Per superare questo problema e consentire ai processi che condividono memoria di comunicare e coordinarsi, migshm prevede un demone MigShareMemD che è attivato su ogni nodo.

Il terzo modulo è il "Consistency Module". Quando processi che utilizzano memoria condivisa migrano su differenti nodi, lavorano su copie locali della memoria condivisa. Così è necessario mantenere la consistenza tra le differenti copie locali della stessa porzione di memoria. Questo rende necessario assicurare che venga letta sempre l'ultima copia dei dati e che le modifiche fatte in scrittura da un processo vengano fornite a tutti i processi remoti.

Il modello di consistenza che è stato adottato per migshm è il cosiddetto "Eager release consistency" nel quale un processo propaga le modifiche da lui apportate ai dati in comune solo quando rilascia quella zona di memoria che condivide con gli altri processi. Usare questo modello garantisce che il nodo proprietario della memoria condivisa ha sempre la copia più aggiornata.

Per ottenere la consistenza vengono usate due operazioni: *writeback* ed *invalidate*. La prima operazione è avviata dal nodo remoto per scrivere nella memoria condivisa; essa consente l'invio al nodo owner delle pagine di memoria scritte dal processo remoto appena prima che il processo rilasci la memoria stessa. Nel nodo proprietario i dati ricevuti sono inseriti nell'opportuna locazione di memoria garantendo così che il nodo owner abbia sempre i dati più aggiornati. Quando il processo remoto scrive una pagina condivisa, invia al nodo proprietario le informazioni sui dati da inserire e sull'indirizzo di memoria relativo, oltre a questi invia l'ID e l'home node per identificare univocamente la memoria condivisa sul nodo di origine. Il nodo proprietario, dopo avere inserito i dati nella giusta loca-

zione di memoria, invia a tutti i processi remoti, che condividono la memoria che è stata appena modificata, un messaggio di *Invalidate* con cui avverte i processi remoti che la pagina di memoria è cambiata; in questo modo la prossima volta che un processo remoto accede alla memoria condivisa potrà prendere la versione più aggiornata. Il messaggio di *invalidate* è gestito in remoto da un daemon (il mig-shm-daemon()) che gira su ogni nodo.

Quando un processo a memoria condivisa migra su un altro nodo, ogni volta che viene mandato il messaggio *invalidate*, una nuova versione aggiornata della pagina di memoria deve essere trasferita dal nodo home attraverso la rete. Oltre a queste informazioni, passeranno anche tutte le quelle che vengono inviate dal nodo remoto ad ogni riscrittura della pagina di memoria da parte di un processo che è migrato. Se gli accessi alla memoria da parte dei proccessi remoti sono più frequenti di quelli dei processi che girano sul nodo home, allora il traffico sulla rete cresce molto. Allo scopo di massimizzare gli accessi locali, la memoria condivisa è migrata con un processo, a patto che esso sia strettamente vincolato alla memoria stessa.

Il quarto modulo è il cosidetto "Access log and migration decisions". Esso registra gli accessi alle pagine di memoria condivisa. Usando queste informazioni viene arricchito l'algoritmo di load-balancing, che consente ad openMosix di prendere decisioni sul nodo più opportuno in cui far migrare un processo. OpenMosix usa un demone chiamato *memsorter*, che raccoglie le informazioni necessarie per l'algoritmo di *memory ushering*; migshm ha esteso le funzionalità di quel demone, rendendo possibile la registrazione degli accessi di tutti i processi legati ad una regione di memoria condivisa. Viene utilizzato un counter per ciascun processo; ogni volta che una pagina di memoria condivisa viene scritta, il counter viene incrementato di tre, ogni volta che viene letta è incrementato di uno. Il counter viene confrontato con un valore di soglia: se il conto degli accessi è inferiore rispetto a quello di soglia, allora il processo verrà considerato debolmente legato; in caso contrario, verrà considerato fortemente legato alla regione di memoria condivisa. Nel primo caso openMosix migrerà il processo da solo, nel secondo caso abbiamo due possibilità: se il processo accede alla memoria condivisa più

di tutti gli altri processi messi insieme, allora la memoria verrà migrata con lui, incrementando così il numero degli accessi locali e riducendo quelli il numero di quelli remoti; se invece accade il contrario, allora il processo in questione viene semplicemente migrato.

Il quinto ed ultimo modulo è il "migration of shared memory" e gestisce la migrazione della regione condivisa. Migshm usa una migrazione logica, cioè il processo migrato possiede la memoria condivisa, ma manda sempre il contenuto aggiornato all'owner node. Questo approccio consente di aumentare gli accessi locali, senza però modificare il setup della memoria condivisa sull'home node. Quando una regione di memoria viene migrata con il processo che è fortemente legato ad essa, il nuovo nodo diventa l'owner della memoria. Il possessore precedente notifica il cambiamento a tutti i nodi in cui si trovano processi che condividono quella regione di memoria, questi manderanno i messaggi di *writeback* al nuovo owner.

Con la descrizione di questo quinto modulo chiudiamo la trattazione teorica su migshm, sottolineando ancora una volta come questa patch di openMosix sia assolutamente ancora in una fase sperimentale. Il suo sviluppo sarà molto importante e ci permetterà di dotare il cluster di uno strumento molto potente che ci consentirà auspicabilemente di colmare quella che in questo momento è forse la più grande lacuna di openMosix: l'impossibilità di migrare applicazioni a memoria condivisa.

Accanto ad openMosix, abbiamo studiato e sperimentato un altro sistema per la creazione e la gestione di un cluster: LTSP. Questo sistema è pensato in un modo che potremmo dire diametralmente opposto a quello di oM. Vediamo nel dettaglio come funziona LTSP e poi cercheremo di fare un confronto tra i due sistemi, per poter comprendere come una loro integrazione possa rivelarsi un vantaggio per le finalità della nostra ricerca.

3.2 LTSP

LTSP (Linux Terminal Server Project) è un sistema **Thin Client**, cioè consiste di un certo numero di workstations diskless che sono connesse ad un server. Tutte le applicazioni girano sul server, che realizza tutte le funzioni di calcolo e di gestione dei dati. Le workstation sostanzialmente forniscono all'utente tastiera, video e mouse per l'accesso alle applicazioni ed ai dati. Per questo motivo, LTSP presenta un'architettura di rete molto diversa da quella di openMosix. La figura 3.3 mostra il fatto che nel cluster LTSP c'è un nodo master, che è il server che svolge tutte le operazioni del sistema, mentre le macchine destinate agli utenti si comportano come semplici terminali. Mentre in openMosix tutti i nodi condividono il carico computazionale dell'intero cluster.

LTSP fornisce un modo semplice per riutilizzare computer a basso costo come terminali grafici, collegandoli ad un server GNU/Linux. Questo progetto, ideato da James A. McQuillan, è stato pensato e realizzato per tutte quelle situazioni in cui c'è bisogno di un numero elevato di computer ma si hanno a disposizione poche risorse. Usando LTSP, infatti, c'è la possibilità di acquistare PC datati, rimuovere il disco fisso, il floppy e il cdrom, aggiungere una scheda di rete che abbia il supporto per la partenza da rete, ottenendo così un cluster a basso costo. Durante la fase di partenza, la workstation ottiene il suo indirizzo IP ed il suo kernel dal server e poi monta un filesystem tramite la rete ed il protocollo NFS. Questa workstation può essere configurata in tre modi:

- Grafico. Usando X window, la workstation può essere usata per lanciare qualsiasi applicazione sul server di partenza, o su qualunque altro server della rete.

- Testo basato su sessioni telnet. La workstation può collegarsi in telnet sul server. Ogni sessione sarà messa su un differente schermo virtuale per un totale di 9 sessioni separate.

- Prompt dei comandi. Questa modalità è molto utile durante la fase di configurazione per risolvere i vari problemi che possono presentarsi.

La cosa interessante è che si possono avere molte workstation che funzionano con un unico server GNU/Linux; la dimensione del cluster dipenderà solo dalle dimensioni e dalle prestazioni del server e dal tipo di applicazioni usate.

3.2.1 Teoria del funzionamento

Far partire un client senza disco include molte fasi diverse. Nel momento in cui accendiamo la workstation collegata al server su cui è istallato il pacchetto LTSP, questa inizierà il Post durante il quale il bios cercherà rom di espansione. La scheda di rete ne contiene una, la eeprom Etherboot[1], ed il bios ne prenderà atto. Una volta che il POST sarà terminato, viene eseguito il codice Etherboot contenuto nella eeprom; esso cerca una scheda di rete che, una volta trovata, viene inizializzata. Il codice a questo punto fa una richiesta DHCP sulla rete locale. La richiesta includerà il MAC address della scheda di rete. Il processo dhcpd sul server legge la richiesta e cerca nel suo file di configurazione (/etc/dhcp.conf) se è presente un MAC address che corrisponde a quello della scheda di rete che ha fatto la richiesta. Se il MAC esiste, il processo dhcpd costruisce un pacchetto di risposta contenente molte informazioni che sarà inviato indietro alla workstation. In esso sono inclusi:

- Indirizzo IP per la workstation;
- Settaggio NETMASK per la rete locale;
- Dirctory e nome del Kernel da scaricare (dal server);
- Directory e nome del filesystem NFS da montare come root;
- Parametri opzionali da passare al kernel.

Il codice Etherboot riceve la risposta dal server e configura l'interfaccia TCP/IP nella scheda di rete con i parametri ricevuti. A questo punto, usando il protocollo

[1]Pacchetto software per creare immagini ROM che possono scaricare codice tramite un collegamento di rete

TFTP (Trivial File Transfer Protocol), il codice Etherboot contatta il server e scarica il kernel, una volta che avrà finito di scaricarlo lo metterà nella corretta locazione di memoria. Il controllo a questo punto passa al kernel che provvede ad inizializzare l'intero sistema e tutte le periferiche che può riconoscere. Attaccata alla fine del kernel scaricato dalla rete c'è l'immagine di un filesystem. Questa viene caricata in memoria come se fosse un normale ramdisk e montata temporaneamente come filesystem di root. Quando il kernel finisce l'operazione di boot, solitamente lancia il programma init; in questo caso è progettato per lanciare uno script apposito chiamato **linuxrc**, che esegue la scansione del bus PCI cercando la scheda di rete e caricando il modulo del driver corrispondente.

A questo punto viene eseguito il **dhclient** per fare un'altra richiesta al server DHCP. Questa seconda richiesta è necessaria perché la prima viene utilizzata dal kernel. Quando il dhclient riceve una risposta dal server viene eseguito il file **/etc/dhclient-script** che consente di recuperare le informazioni per la configurazione di eth0.

Fino ad ora il filesystem di root è stato montato in ramdisk. Ora, lo script /linuxrc ne monterà uno nuovo su root attraverso NFS, esportando dal server la directory **/opt/ltsp/i386**. In realtà non potendo montare il nuovo fielsystem su /, lo monterà prima su /mnt, eseguirà poi un operazione detta di **pivot-root**, che scambierà il root filesystem corrente con uno nuovo. A operazione ultimata avremo il filesystem NFS montato su / e su /oldroot quello che era montato in ramdisk. Solo a questo punto viene avviato il programma **init** che comincerà a configurare l'ambiente della workstation tramite il file /etc/inittab. Questo file lancia il comando rc.local che crea un ramdisk montato sulla directory /tmp di 1Mb, con lo scopo di contenere tutti i file che necessitano di essere modificati durante il normale funzionamento della workstation. Quando questa operazione sarà ultimata, se la workstation è stata configurata per eseguire lo swap della ram su NFS, viene montata la directory **/var/opt/ltsp/swapfiles** su /tmp/swapfiles e se qui non è presente nessun file di swap ne verrà automaticamente creato uno, di grandezza stabilita all'interno del file di configurazione della workstation chiamato **/lts.conf** e che sarà stato opportunamente editato in fase di progettazione del

cluster. Successivamente viene configurata l'interfaccia di loopback per la rete.

In generale, per ciascuna workstation che costituisce il sistema sarà possibile anche prevedere alcune applicazioni che girano in locale; il nome dell'applicazione dovrà essere indicato nel file /lts.conf di configurazione del client. Nel caso in cui le applicazioni locali siano abilitate, viene montata la directory **/home** in modo che le applicazioni possano accedere alle directory degli utenti.

Il sistema X Windows viene ora configurato. Nel file lts.conf, c'è un parametro chiamato **XSERVER** che ci consente di specificare la scheda video utilizzata da ciascuna workstation; se non é specificato nulla, allora verrà avviata la ricerca automatica della scheda. Se essa è supportata da XFree86, verrà caricato il driver X da usare e generato il file di configurazione XF86Config che consente di settare opportunamente X. Al termine di questa fase, viene creato lo script /tmp/start-WS, che è responsabile della corretta partenza del server X. A questo punto il controllo passa di nuovo ad **init**, che guarderà al parametro initdefault per determinare in quale runlevel entrare. Ora la workstation è pronta per essere utilizzata.

perché tutto ciò funzioni correttamente sono necessarie alcune operazioni previe di fondamentale importanza. Il primo passo consiste nell'installare i pacchetti di LTSP; una volta fatto questo, è necessario configurare il sistema LTSP. Ciò vuol dire modificare i file di configurazione in modo da abilitare il server ad offrire i servizi necessari alle varie workstation. A questo scopo si devono editare in modo opportuno tutti i file che consentono di specificare al server le caratteristiche delle workstation, in particolare:

- /etc/dhcpd.conf che indica l'indirizzo IP e l'hostname delle workstation, l'IP del server, la dirctory ed il nome del kernel da caricare ed infine la directory da montare come filesystem di root;

- /etc/host che contiene l'IP ed il nome delle workstation;

- /opt/ltsp/i386/etc/lts.conf che contiene tutte le caratteristiche delle workstation connesse al server.

Dopo aver configurato il server, è necessario passare alle workstation.

Il pacchetto LTSP riguarda tutto quello che succeda dopo che il kernel è stato montato sul client; ci sono diversi modi per far entrare il kernel in memoria: noi abbiamo utilizzato un floppy contenente il codice del progetto **Etherboot**. Questo è un pacchetto software Open Source che serve per creare un immagine Rom che possa scaricare, tramite un collegamento di rete, del codice eseguibile su macchine x86. Una volta che abbiamo a disposizione il floppy con l'immagine già compilata[2], la workstation non ha più bisogno di niente e tutto è pronto per avviare il cluster.

Ora che abbiamo visto come LTSP funziona, vediamo quali sono i vantaggi e gli svantaggi di un sistema thin client rispetto ad un normale sistema Windows del tipo PC workstation/server.

Tra i vantaggi

- Costi nulli delle licenze per il software;
- Costi dei terminali ridotti, perché i thin client sono meno costosi di un normale pc;
- Istallazione più veloce ed economica, visto che non serve configurazione dell'hardware dei terminali ed anche la configurazione del profilo utente sul server è un'operazione abbastanza agevole;
- Aggiornamento del software più economico e veloce, perché appena le applicazioni sono aggiornate sul server sono disponibili automaticamente a tutti gli utenti;
- Allungamento dei tempi di obsolescenza dell'hardware dei terminali, visto che non devono soddisfare a particolari specifiche prestazionali;
- Costi di manutenzione ridotti, perché i terminali hanno meno componenti e quindi meno probabilità che si abbia la rottura di alcune parti;

[2]Scaricabile dal sito www.Rom-o-Matic.net

- Costo ridotto per assicurare il sistema dagli attacchi di virus, perché i terminali sono immuni e perché ci si può concentrare sulla sicurezza del solo server;

- I terminali richiedono meno energia dei PC, in più sono più leggeri, meno ingombranti, meno rumorosi e più robusti agli ambienti ostili rispetto alle PC Workstation.

L'unico svantaggio saliente che possiamo rilevare nel sistema è quello legato alle prestazioni ed alla robustezza del server. Cioè all'aumentare del numero di client, deve aumentare anche la RAM e la velocità della CPU del nodo centrale. Questo potrebbe divenire un problema per chi come noi lavora con sistemi di cluster ottenuti con hardware obsolescente. In più, un problema che si può avere è che nel momento in cui il server va fuori uso tutto il cluster diventa inutilizzabile.

Al di là di quest'ultima valutazione, possiamo, in generale, dire che per le caratteristiche favorevoli che abbiamo appena elencato, LTSP è utilizzato in moltissimi progetti di alfabetizzazione informatica nelle scuole (che dispongono di pochi fondi) o in paesi in via di sviluppo in cui si hanno poche risorse finanziarie.

Per concludere, vediamo di sottolineare alcune delle differenze che presentano i due sistemi che abbiamo analizzato.

3.2.2 OpenMosix vs LTSP

Come già accennato i due sistemi di clustering nascono per scopi molto diversi e per questo sono sistemi che presentano innumerevoli differenze.

OpenMosix è pensato e sviluppato per fare calcolo ad alte velocità e prestazioni, mentre LTSP è stato progettato per tutte quelle situazioni in cui si ha la necessità di avere più computer connessi a formare una rete (come in un ufficio o in una scuola), ma non sia hanno a disposizione molte risorse economiche. Ovviamente questa differenza negli scopi per cui i due sistemi sono stati progettati ha come diretta e naturale conseguenza una differenza nelle loro architetture, come mostrato in figura 3.2 e in figura 3.3.

In openMosix non è previsto un sistema master/slave, non c'è, cioè, un nodo centrale del cluster che coordina gli altri o che faccia da server, tutti i nodi contribuiscono in ugual misura al calcolo ed ogni nodo agisce come un sistema autonomo che prende decisioni indipendenti da quelle degli altri nodi. In LTSP, invece, c'è un server da cui tutti i nodi dipendono; i nodi diskless forniscono solo un terminale che consente agli utenti di accedere a quelle che sono le funzionalità del server; su quest'ultimo girano tutte le applicazioni ed è eseguito tutto il calcolo ed il controllo della rete.

Va fatto un ultimo accenno alla possibilità di fondere i due sistemi. E' stato sviluppato un pacchetto apposito (ltsp-mosix-core-1.0beta4) che consente ad una rete LTSP di trasformarsi in un cluster openMosix. Molto semplicemente, quando i terminali fanno il boot dalla rete, caricano un kernel "patchato" openMosix invece del kernel normale; in più ciascun client carica dal server tutti i file di configurazione e di gestione necessari per far parte del cluster. Al livello attuale di sperimentazione, questo sistema semplicemente consente di avere un cluster openMosix perfettamente funzionante senza preoccuparsi del settaggio dei nodi.

Figura 3.2: architettura openMosix.

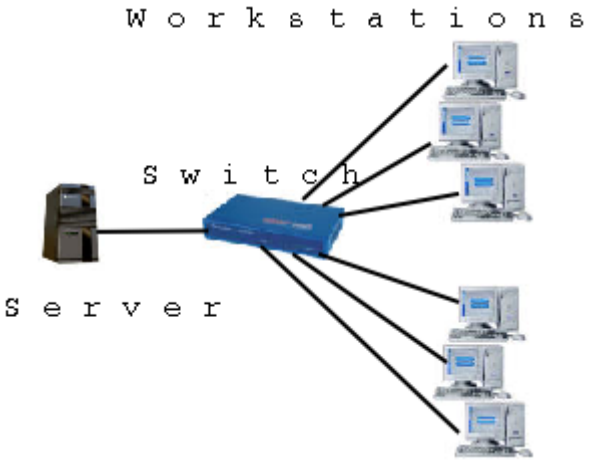

Figura 3.3: architettura LTSP.

Capitolo 4

I test

La sperimentazione è stata sviluppata in più fasi nelle quali abbiamo installato e provato due diversi sistemi di clustering (openMosix ed LTSP) e differenti versioni dello stesso sistema nel caso di openMosix. La maggior parte dell'attività è stata condotta in un laboratorio di informatica allestito con la collaborazione di Ingengeria Senza Frontiere nei locali messi a disposizione dal **BugsLab**, presso lo Spazio Pubblico Autogestito **Strike**. ISF ha fornito l'hardware su cui sono stati svolti i test e gli esperimenti, oltre ad un costante supporto tecnico ed un fondamentale sostegno umano. Fondamentale per lo svolgimento del lavoro (e per suoi eventuali sviluppi futuri), è stato il ruolo ricoperto dal BugsLab, nato da un progetto di due laboratori informatici autogestiti ed aperti a tutti, con la finalità di promuovere un uso critico della tecnologia informatica e di trasmettere la conoscenza posseduta o acquisita. Gli operatori del BugsLab sono impegnati in progetti di Trashware e di alfabetizzazione informatica ed oltre a fornire supporto logistico ospitando il nostro test-bed, hanno messo a disposizione il loro prezioso supporto tecnico e il materiale hardware necessario in diverse circostanze.

Nel BugsLab, oltre a sperimentare diverse piattaforme software distribuite, è stato possibile anche lavorare sulle componenti hardware delle macchine obsolete a nostra disposizione, risolvendo i problemi piccoli o grandi che di volta in volta sono occorsi, prendendo contatto con il lavoro di Trashware vero e proprio e acquistando sempre maggiori competenze tecniche.

4.1 Il nostro test-bed

Il nostro test-bed per la maggior parte della sperimentazione è stato costituito da quattro nodi eterogenei:

- Il nodo 1: Pentium MMX 166MHz con 32Mb Ram e scheda di rete intell eepro100
- Il nodo 2: Pentium Celeron 733 MHz con 64Mb Ram e scheda di rete 3com 3c905
- Il nodo 3: Pentium MMX 200MHz con 64Mb Ram e scheda di rete Realtek RTL-8139C
- Il nodo 4: Pentium 133MHz con 48Mb Ram e scheda di rete ISA 3com 3c509

La scelta dei nodi differenti ha reso molto più complessa la gestione del cluster, ma ci ha consentito di lavorare in un ambiente che riproduce in modo fedele quello in cui si trova chi lavora con macchine obsolete al fine di recuperarle al meglio. I quattro nodi sono connessi attraverso uno switch da 100Mbit/s, che ha consentito di avere una connessione abbastanza efficiente e veloce. Su tutte le macchine è stata installata una distribuzione GNU/Linux Debian 3.0 e sui nodi 2 e 4 è stato installato il server X. Infine, su ciascuna delle macchine connesse in rete, sono stati installati i kernel 2.4.21 e 2.4.22 entrambi "patchati" con openMosix. Per i dettagli sull'installazione e per la descrizione dei tools che sono stati utilizzati per il monitoraggio del cluster, si veda l'Appendice C.

Le prestazioni del nostro cluster sono state confrontate con quelle di una macchina più performante con le seguenti caratteristiche:

- AMD Athlon XP 1700+ con 1Gb di RAM e con distribuzione GNU/Linux Gentoo.

Nel corso della sperimentazione, è stato possibile anche valutare il comportamento di un cluster openMosix di macchine di ultima generazione, avendo avuto

la possibilità di installare openMosix 2.4.22 su sei postazioni in un laboratorio messo a disposizione dallo IASI (Istituto dell'automazione dei sistemi e informatica del CNR). Qui abbiamo lavorato su computer identici con la seguente configurazione:

- Pentium IV con CPU a 2400 MHZ e 1Gb di RAM con distribuzione GNU/Linux Debian 3.0; su tutti i nodi era installato il server X.

Vedremo come l'ambiente del CNR sia stato utile non tanto per produrre confronti quantitativi con il nostro cluster "povero", quanto per il fatto che ha permesso di valutare quello che è il comportamento del sistema openMosix nelle sue condizioni ideali di funzionamento, cioè con una configurazione in cui il fatto di avere nodi uguali consente di ridurre il tempo in cui l'algoritmo di load-balancing di openMosix decide su quale nodo migrare il generico processo.

Per la valutazione delle prestazioni di LTSP, infine, abbiamo utilizzato la seguente configurazione server-client:

- Server: Pentium IV 1700 MHz con 384Mb di RAM con distribuzione GNU/Linux RedHat 9.0

- Client: Pentium 133MHz con 48Mb di RAM con distribuzione GNU/Linux Debian 3.0

I due computer sono connessi in modo diretto con cavo di rete incrociato.

Dopo aver descritto i diversi ambienti in cui sono stati condotti gli esperimenti, vediamo di mostrare nel dettaglio i test effettuati ed i risultati prodotti.

4.2 I test su openMosix-2.4.22

Iniziamo col descrivere gli esperimenti condotti sul cluster openMosix-2.4.22, che d'ora in avanti chiameremo **cluster-OM1**.

4.2.1 Test 1: i cicli dummy

Il primo test consiste nel misurare il tempo impiegato dal cluster per completare un certo numero di cicli "dummy". Il singolo ciclo è fatto partire tramite l'istruzione da riga di comando:

time awk 'BEGIN {for(i=0;i< n;i++)for(j=0;j< m;j++);}' &

Entrando nel dettaglio dell'istruzione possiamo dire che AWK è un linguaggio di programmazione interpretato , che presenta direttive BEGIN-END (che vengono eseguite prima e dopo il processamento dell'input), cicli IF-ELSE, variabili speciali e funzioni varie.

Nel caso del nostro test è semplicemente il linguaggio che ci permette di iniziare da riga di comando un ciclo FOR "dummy", cioè che non restituisce nessun risultato e non esegue alcuna operazione.

Tale ciclo semplicemente ci consente di stressare la CPU del nodo che è chiamato ad eseguirlo per un tempo che è legato al valore che noi diamo alle due variabili n ed m indicate nella linea di comando.

Il **time** all'inizio della riga di comando ci consente di avere in output il tempo impiegato dalla CPU a completare il ciclo. Infine il simbolo & alla fine della linea ci consente di mandare il processo in background e di avere la possibilità di mandare in esecuzione un'altra istruzione.

Alla fine dell'esecuzione del ciclo l'output ottenuto è di questo tipo:

real 0m4.581s

user 0m4.570s

sys 0m0.000s

Cioè abbiamo in output il tempo reale che intercorre tra l'istante in cui è stata lanciata l'istruzione e quello in cui è stato completato il ciclo, il tempo utente che è quello impiegato per eseguire il processo ed infine il tempo di sistema. Per un solo ciclo i primi due tempi sono molto simili, ma quando il numero di processi cresce, il tempo reale risulterà essere quello totale necessario per l'esecuzione di tutti i processi lanciati, mentre il tempo utente sarà quello che compete ad ogni

singolo processo. Nei test eseguiti viene preso il real time dell'ultimo dei processi completati come misura delle prestazioni.

Per quanto riguarda l'esecuzione di queste istruzioni da parte del cluster, risulta che ciascuna di esse è un processo; per come è progettato openMosix (si veda il capitolo-3), l'elemento minimo di lavoro è un processo. Questo significa che le istruzioni non possono essere "spezzettate" in più parti per essere migrate sui diversi nodi, ma saranno eseguite ciascuna su un nodo. Per valutare con questo tipo di test il comportamento di openMosix e per assistere a migrazioni dei job sui vari nodi del cluster, è stato necessario lanciare una certa quantità di queste istruzioni/processi.

Nel caso di un cluster eterogeneo come il nostro, lanciando una singola istruzione si ha comunque un vantaggio, e cioè che il processo sarà migrato verso il nodo che presenta le prestazioni migliori in termini di CPU e RAM. In questo modo, il processo lanciato, per esempio, dal nodo4 del cluster-OM1 sarà eseguito dal nodo2, che è molto più performante, con ovvi vantaggi.

Questo tipo di test è stato fatto lanciando un numero di processi variabile e modificando, inoltre, il numero di iterazioni all'interno dei cicli.

In un primo test ho lavorato con le seguenti due istruzioni[1] lanciate sempre dal nodo2, ciascuna da una a cinque volte:

Istruzione-1:

time awk 'BEGIN {for(i=0;i<10000;i++)for(j=0;j<10000;j++);}' &

Istruzione-2:

time awk 'BEGIN {for(i=0;i<20000;i++)for(j=0;j<10000;j++);}' &

Come si vede le due istruzioni si differenziano per il numero di iterazioni di uno dei due cicli *for*. All'aumentare dei questo numero aumenta anche il peso computazionale di ciascuna istruzione.

Alla fine dell'esecuzione di ciascun processo viene mostrato in output sul nodo2 (da cui tutte le istruzioni sono state mandate) il tempo intercorso tra l'istante in

[1] I test sono stati, in realtà eseguiti su un numero molto grande di istruzioni differenti per numero di iterazioni in un ciclo, ma poi la scelta è ricaduta su queste due perché fornivano risultati più significativi.

cui il job è stato lanciato nel cluster e il momento in cui è stato completato. Il tempo impiegato dal cluster-OM1 per completare l'esecuzione dei processi è stato poi confrontato con quello del nodo più performante avendo, però disattivato openMosix.

La figura-4.1 mostra che mentre nel caso del nodo2 all'aumentare del numero di processi si ha un aumento quasi linerare del tempo impiegato dalla macchina per portare a compimento il job, invece con il cluster si ha una crescita nel tempo molto ridotta. Più precisamente, facendo i calcoli otteniamo che nel caso del nodo singolo il tempo che si impiega ad eseguire 5 processi è circa 5 volte quello necessario per eseguire un solo job, e questo è vero sia per la Istruzione-1 che per la Istruzione-2. Per il cluster-OM1, invece, se indichiamo con T_i e T_f i tempi per l'esecuzione di un processo e di cinque processi rispettivamente, allora possiamo trovare le seguenti relazioni per i due tipi di istruzione:

Istruzione-1: $T_f = 3.65T_i$

Istruzione-2: $T_f = 3.34T_i$

Queste due semplici relazioni sono di notevole importanza perché ci dicono che utilizzando openMosix ottengo prestazioni migliori sia all'aumentare del numero dei processi lanciati, sia all'aumentare del costo computazionale delle operazioni.

La figura-4.2, invece, mostra lo speed-up che si ha nell'utilizzare il cluster rispetto al solo nodo2. E' da sottolineare come il confronto mostrato in figura sia fatto rispetto al nodo più performante del cluster; ciò vuol dire che il confronto con uno degli altri nodi darebbe un risultato ancora più favorevole per il cluster.

Molto interessante è stato anche andare a fare dei confronti tra l'hardware obsoleto, rappresentato dal nostro cluster-OM1, e l'Athlon 1700+ che nel seguito chiameremo **PIV**; le misure sono state fatte sempre con le solite due istruzioni già descritte. Questa volta, però, è stato lanciato un numero di processi superiore (nove). Il diagramma mostrato in figura-4.3 ha un'importanza centrale per il nostro studio:

fa vedere, infatti che all'aumentare del numero di processi (e quindi delle migrazioni sul cluster) ed all'aumentare del costo computazionale (cioè delle iterazioni in un ciclo), il cluster-OM1 tende a superare il PIV, cioè offre prestazioni

migliori. In particolare, nella figura, per ciascuna delle due istruzioni lanciate, sono messi in evidenza i due cross-point (evidenziati dalle X rosse), ovvero i punti precisi in cui il cluster supera nelle prestazioni il PIV. Dal cross-point in poi, si nota come la differenza di tempocresca sempre più al crescere del numero dei processi; si evidenzia dunque uno speed-up sempre maggiore, e quindi un vantaggio sempre crescente dell'hardware povero rispetto all'architettura moderna.

Altro test fatto sui cicli dummy, riguarda un confronto tra il nostro cluster ed il PIV, questa volta al variare del numero dei nodi della rete. Il risultato dell'esperimento è mostrato in figura 4.4.

Per poter correttamente leggere la figura, dobbiamo descrivere la configurazione che è stata scelta per il cluster al variare dei nodi:

- il Cluster2 è formato dal nodo1 e dal nodo3 come descritti all'inizio del paragrafo
- il Cluster3 è formato dai nodi 1-2-3 (è escluso quello meno performante)
- il Cluster-OM1 è quello completo con i 4 nodi, ed è quello che abbiamo utilizzato fin qui per i test già mostrati

Dalla figura si evincono alcune informazioni importanti: il Cluster2 realizzato con nodi molto poco prestazionali non riesce mai ad avvicinare i tempi del PIV; le cose diventano interessanti quando si passa a considerare il Cluster3 (aggiungendo il nodo2). Come si vede, i tempi si abbattono enormemente grazie al maggior numero di nodi che si suddividono il lavoro e grazie alle superiori capacità di calcolo del nodo2. Il divario tra il Cluster3 ed il PIV decresce sempre più al crescere del numero di processi lanciati, fino a che, per 5 processi, assistiamo ad un'inversione del trend con il Cluster3 che impiega meno tempo del PIV e con una differenza nei tempi di esecuzione delle istruzioni che tende questa volta ad aumentare al crescere del numero dei processi. Comportamento analogo si osserva nel confronto tra Cluster-OM1 e PIV, questa volta il cross-point si ha già per quattro processi (come già indicato nella figura 4.3). Nel confronto tra il cluster con 4 macchine e quello con 3, notiamo una differenza che è abbastanza

ridotta a causa delle scarse prestazioni del nodo4. E' molto interessante, , infine, notare che la differenza di tempo tra il Cluster3 ed il Cluster-OM1 ha un andamento molto irregolare, legato al modo in cui la migrazione dei processi è gestita dall'algoritmo di load-balancing di openMosix. L'algoritmo infatti tende a direzionare la migrazione verso i nodi più performanti, inviando processi verso i nodi meno potenti solo quando il carico dei primi è tale da non consentire loro di gestire in modo efficiente l'arrivo e l'esecuzione di nuovi processi. Quanto appena detto è evidente osservando che i tempi per l'esecuzione dei primi tre processi è sostanzialmente uguale per le due configurazioni di clustering e che comincia ad esserci un differenza maggiore solo dal quarto processo in poi; infatti è solo dal quarto processo che il nodo4 viene chiamato ad eseguire job, essendo tra tutte le macchine quella con meno risorse.

E' stato di estremo interesse, l'ultimo test eseguito confrontando il cluster-OM1 di macchine obsolete ed il PIV preso da riferimento. Questo esperimento è stato condotto per avere una misura qualitativa della maggiore scalabilitá del cluster rispetto all'architettura moderna. Il test è consistito nel lanciare un numero molto grande di processi (ancora una volta tramite l'istruzione-2), valutando quando i due sistemi raggiungevano la saturazione. Nell'esperimento si è considerato come indicatore dell'avvenuta saturazione della macchina singola e del cluster, il tempo impiegato dal sistema, per dare risposta a delle istruzioni banali, come *ps*, per esempio, che dà in output la lista dei processi attivi.

Abbiamo considerato avvenuta la saturazione nel momento in cui il tempo di risposta ai comandi della tastiera e poi, all'istuzione ps superavano il minuto.

Dal test è risultato che il PIV era bloccato già a 150 processi, mentre il cluster risultava avere un comportamento analogo solo oltre i 250 processi, come graficato in figura-4.5. I risultati a cui siamo pervenuti attraverso questo test, dimostrano che in applicazioni in cui il carico per la CPU è molto elevato, il cluster-OM1, non solo presenta prestazioni migliori in termini di tempi di esecuzione, ma consente, anche, di processare un numero molto maggiore di istruzioni.

Prima di concludere con la descrizione del **test 1** è necessario dedicare qualche commento al cluster che è stato installato e testato al CNR (che indicheremo con

Cluster-CNR). Come detto all'inizio del capitolo a causa delle caratteristiche dei nodi dell'ambiente creato al CNR, non è possibile fare un confronto diretto tra le prestazioni del Cluster-CNR e del Cluster-OM1; comunque i test sono stati molto utili perché ci hanno permesso di avere una idea migliore di quale sia il comportamento di openMosix in condizioni ideali. Anche al CNR il test è consistito nell'esecuzione delle Istruzioni 1 e 2 (lanciate sempre da uno stesso nodo) per un numero di volte variabile da uno a sei, sul cluster di 6 macchine. I risultati sono mostrati in figura 4.6

L'elemento nuovo ed interessante che osserviamo rispetto al nostro cluster di macchine obsolete, è che il CLuster-CNR ha un comportamento molto regolare (dovuto al fatto che tutti i nodi sono identici), in cui le curve del tempo per entrambe le istruzioni si mantengono quasi costanti al crescere del numero dei processi. La leggera salita che si nota è legata al fatto che intercorre un certo tempo tra l'istante in cui il processo è lanciato da un nodo e il momento in cui inizia ad essere eseguito dopo che c'è stata la migrazione. Diverso sarebbe stato l'andamento se i sei processi fossero stati lanciati contemporaneamente dalle sei macchine, abbattendo così i tempi necessari alla migrazione. Il grafico presenterà poi una crescita più marcata tra il sesto ed il settimo processo per effetto del fatto che si avrà un numero di istruzioni maggiore di quello dei nodi del cluster; successivamente, fino al dodicesimo processo si avrà sempre un andamento quasi constante analogo a quello che si riscontra nella figura per i primi sei processi.

4.2.2 Test 2: i frattali

Il secondo test è simile al primo dal punto di vista computazionale con la differenza che si è lavorato con un'applicazione: **KFract**. Questa, è contenuta nel pacchetto di grafica di KDE[2] (Kdegraphics) e serve per produrre frattali, ovvero curve che godono della cosiddetta proprietà di *autosimilitudine*, per la quale ingrandendo un tratto di curva qualsiasi, si visualizza un insieme di particolari altrettanto ricco e complesso del precedente, con un procedimento di "zoom" che può proseguire

[2] E' l'ambiente grafico che abbiamo utilizzato nel nostro server X

all'infinito, con la conseguenza che presi due punti anche vicinissimi sulla curva, essi hanno distanza, misurata lungo la curva stessa, infinita.

Tornando a Kfract, si deve sottolineare come anche questa applicazione così come le istruzioni che abbiamo utilizzato precedentemente stressa molto la CPU toccando poco la RAM.

L'istruzione che è stata lanciata da riga di comando è:

time kfract & xkill &

time è l'istruzione che restituisce in output il tempo necessario al processore per eseguire il comando che viene subito dopo sulla linea. Così come detto per il test precedente il simbolo & manda in background l'istruzione consentendo di lanciare immediatamente un altro comando senza dover aspettare la fine dell'applicazione stessa. Così subito dopo aver lanciato *kfract*, viene avviato xkill che è un comando che consente di terminare un'applicazione cliccando sulla sua finestra. L'ultimo & restituisce ancora una linea di comando per lanciare altre applicazioni.

Ciascuna istruzione kfract lanciata su un nodo costituisce un processo e come tale viene eseguito per intero, cioè senza essere scomposto in diverse parti. Per cui si ripetono anche in questo caso le valutazioni fatte nel test precedente sulla modalità di gestione da parte di openmosix delle istruzioni che sono state eseguite.

Il frattale che è stato scelto per essere realizzato con kfract è **Mandelbrot** che è computazionalmente più pesante rispetto all'altro disponibile (Julia). Anche il frattale, così come le due istruzioni del test precedente, è stato eseguito da una a cinque volte (che significa lanciare sul cluster da uno a cinque processi). L'applicativo Kfract dà la possibilità di settare il numero di iterazioni con cui eseguire il frattale stesso, aumentando la quantità di pixel di cui la figura è composta e consentendo, così, di modificare il carico computazionale nell'esecuzione dell' istruzione. Ciascuna istruzione, dunque, è stata lanciata (ancora a partire dal nodo2) modificando il numero di iterazioni necessarie per il calcolo del frattale.

Per semplicità di trattazione indichiamo con Kfract1 l'istruzione kfract in cui si sia settato pari a 10000 il numero di iterazioni, con Kfract2 l'istruzione in cui il tale numero sia posto pari a 40000 ed infine Kfract3 l'applicazione in cui si esegue

il frattale con 60000 iterazioni. Ciascuna di queste tre istruzioni costituisce un processo ed è stata lanciata da una a cinque volte.

La misura del tempo impiegato dal cluster per eseguire i job è stata presa considerando il valore di output dell'istruzione **time**, riferito all'ultimo processo lanciato. In questo modo è stato possibile calcolare il tempo intercorso tra l'istante in cui è stata lanciata la sequenza di istruzioni e quello in cui l'ultimo processo è stato completato. In ultimo si deve sottolineare che l'aggiunta di xkill è stata necessaria perché il valore di "time" viene restituito solo quando la finestra dell'applicazione viene richiusa dall'utente e con xkill è possibile fare questo istantaneamente.

La figura-4.7 mostra i risultati del test. Anche in questo caso il comportamento del cluster-OM1 è rapportato a quello del nodo2, che offre le migliori prestazioni. Quello che possiamo notare è che il nodo singolo ha un comportamento migliore rispetto all'intero cluster per l'istruzione meno pesante dal punto di vista computazionale ed in generale quando sono lanciati pochi processi. Invece all'aumentare del costo computazionale e all'aumentare del numero di processi, cioè al crescere del carico complessivo, il cluster-OM1 offre prestazioni sempre migliori.

4.2.3 Test 3: Povray

Il terzo test è concettualmente diverso dagli altri due; si è trattato di andare a misurare le prestazioni del Cluster-OM1 nel realizzare **rendering grafico** di immagini. Il programma utilizzato nel test è chiamato **Pov-ray** ed è in assoluto l'applicativo Open Source più utilizzato da chi si occupa di "renderizzazione" di immagini. E' stato scelto Povray perché ne esiste una versione realizzata appositamente per funzionare con cluster, attraverso l'uso delle librerie di calcolo parallelo (PVM) e chiamata appunto pvmpovray. Quando pvmpovray lavora in un cluster, distribuisce il carico su tutti i nodi. Quando lavora in un cluster come quello che è stato testato, è openMosix stesso a realizzare la migrazione e la distribuzione del carico sui diversi nodi. Oltre a questa un'altra motivazione che ci ha guidato nella scelta di condurre test su questa appplicazione, è che

il rendering grafico tipicamente richiede moltissimi calcoli ed impegna molto la CPU, prestandosi, perciò molto bene a costituire ambiente per il benchmarking.

Il test che è stato realizzato è consistito nel renderizzare alcune immagini di complessità crescente, fornite direttamente all'interno del pacchetto di povray. L'istruzione lanciata per realizzare il test è stata:

povray nome-immagine.pov +V +W640 +H480

dove +V stabilisce la modalità "verbose" tramite la quale possiamo avere tutte le informazioni che ci servono sullo stato di avanzamento del processo, mentre "+W640 +H480", è l'opzione che consente di stabilire le dimensioni dell'immagine. Il tempo impiegato per il rendering viene fornito dal programma stesso alla fine dell'esecuzione.

Vale la pena di accennare il principio di funzionamento di PVMPOV, per comprendere in che modo il programma trae vantaggio dall'esecuzione su un cluster di macchine piuttosto che su un solo PC. Brevemente possiamo dire che usando il modello PVM, c'è un task master e diversi task slave. Il master ha la responsabilità di dividere l'immagine in piccoli blocchi, che sono assegnati agli slave. Quando gli slaves hanno concluso il rendering dei blocchi che competono loro, li reinviano al master che li combina per formare l'immagine finale. Il master non fa rendering ma solo coordinamento, tuttavia, c'è uno slave che gira sulla stessa macchina del master e questo è possibile poiché esso utilizza poca CPU.

Come detto sono state renderizzate diverse immagini ed i tempi impiegati dal cluster-OM1 per la loro realizzazione sono stati confrontati con quello impiegato dal PIV e dal nodo 4 del cluster stesso, ripetendo la metodologia già adottata per gli altri test. Questo è stato fatto allo scopo di osservare, se ed in che modo, il cluster di hardware obsoleto riusciva a superare in prestazioni il computer moderno e quanto vantaggio il PI (il nodo 4) traeva dal fatto di trovarsi all'interno di un'architettura in cui il carico è distribuito su più macchine.

La figura-4.8 mostra i risultati del test; da essa si possono ricavare due informazioni molto interessanti: la prima è che il PI trae un vantaggio enorme quando diviene parte del cluster, vista la differenza che c'è tra il tempo da esso impiegato

e quello impiegato dal cluster; la seconda informazione di grande importanza è che rispetto ai test visti precedentemente non si è riscontrata, in questo caso e con le immagini che abbiamo "renderizzato", una situazione in cui il cluster supera nelle prestazioni il PIV preso come riferimento. Per contro si può notare che la differenza percentuale tra i tempi del cluster-OM1 e del PIV tende a diminuire all'aumentare della complessità dell'immagine da trattare. Questa tendenza che è stata riscontrata, è mostrata nella figura-4.9.

Con il test 3 si conclude la serie di esperimenti eseguiti sul cluster-OM1. Il commento che si può fare sispetto ai risultati che sono stati prodotti, è che openMosix conferma le sue caratteristiche di alta efficienza computazionale anche in condizioni di funzionamento diverse da quelle per le quali è stato progettato; cioè con cluster costituito da macchine eterogenee e con poche risorse. Dobbiamo sottolineare, a conclusione di questa parte, due risultati fondamentali: il primo è che in tutte quelle applicazioni in cui vengono fatti molti calcoli, il cluster povero si avvantaggia della distribuzione delle risorse, che openMosix rende possibile, superando nelle prestazioni, il PIV; il secondo riguarda la scalabilità; ciò che possiamo dire è che il cluster riesce a sopportare, rispetto al PIV, un carico molto maggiore prima di arrivare alla saturazione, con conseguenze molto interessanti sulla possibilità di utilizzo di un cluster povero in contesti in cui si hanno diversi utenti che lanciano numerose applicazione sui nodi del cluster, come accade, per esempio, all'interno di laboratori informatici.

4.3 I test su openMosix e Migshm

In questo paragrafo si mostreranno i test che sono stati realizzati sul cluster openMosix con l'aggiunta delle nuove funzionalità (ancora totalmente sperimentali) che la patch Migshm fornisce al sistema. Come visto nel capitolo-3 openMosix non è in grado da solo di gestire la migrazione delle applicazioni a memoria condivisa che sono, invece, della massima importanza per le finalità dello studio condotto, visto che gran parte dei programmi utente fanno uso della memoria condivisa. Gli esperimenti sono stati fatti sul test-bed visto all'inizio di questo

capitolo, ma con la differenza fondamentale che per applicare la nuova patch ad openMosix è stato necessario cambiare il kernel, passando dal 2.4.22 al 2.4.21, a causa del fatto che al tempo in cui i test hanno avuto inizio l'unica versione stabile e di migshm era fornita solo per quel kernel.

Chiamerò il nuovo ambiente dotato del nuovo kernel e di Migshm **cluster-Migshm**. Su di esso sono stati ripetuti i test già visti per il cluster-OM1, per verificare che le prestazioni e tutte le funzionalità standard di openMosix rimanessero inalterate in situazioni normali di funzionamento del sistema. Dopo aver verificato che le prestazioni del cluster rimanevano sostanzialmente le stesse, si è passati alla sperimentazione specifica su cluster-Migshm.

4.3.1 Test: migrazione di applicazioni a memoria condivisa

Il test che abbiamo eseguito sul cluster nella nuova configurazione è consistito nell'aprire diverse istanze di alcune applicazioni per verificare la presenza di migrazione di processi (che con il cluster-OM1 invece rimanevano vincolati al nodo d'origine), e per prendere delle misure dei tempi di apertura delle applicazioni stesse per verificare in che modo la presenza di migshm incidesse su di esse.

Sono state fatte misure su **Kword**, **Konqueror**, **OpenOffice** e **The Gimp**[3]. Le prime due sono applicazioni che girano nell'ambiente grafico KDE e sono rispettivamente un programma di scrittura ed un browser; OpenOffice è, invece, la più importante e completa suite da ufficio Open Source. Infine The Gimp è un programma per il trattamento delle immagini.

Su Kword e Konqueror il test è stato il seguente: apertura di cinque istanze di un file di testo (openMosix-HOWTO.txt) tramite l'utilizzo delle due applicazioni e verifica della migrazione di processi tramite openmosixmigmon (si veda

[3]Numerose sono le applicazioni testate per valutarne la migrabilità con Migshm, ma queste sono sembrate più interessanti di altre perché hanno messo in evidenza diversi comportamenti di openMosix modificato.

l'appendice C per la descrizione del tool) e confronto dei tempi di apertura con il cluster-OM1. Le istruzioni lanciate da linea di comando sono le seguenti:

time kword openMosix-HOWTO.txt & xkill &

time konqueror openMosix-HOWTO.txt & xkill &

Ciascuna delle due istruzioni è stata ripetuta cinque volte per poter stressare la CPU e soprattutto la RAM del nodo su cui venivano lanciate e favorire la migrazione dei processi sugli altri nodi. I tempi di apertura delle istruzioni sono stati presi grazie al comando **time** che restituisce un valore appena la quinta iterazione delle due applicazioni viene chiusa tramite il comando xkill.

Nell'esecuzione del test abbiamo potuto assistere alla migrazione di processi dal nodo da dui le applicazioni sono state lanciate, verso gli altri nodi del cluster. Questo è già un risultato di estrema importanza ed interesse per la nostra ricerca. Per quanto riguarda i tempi di apertura delle applicazioni ed il confronto con il cluster-OM1, i risultati sono mostrati nella figura 4.10. Da essa si evince che il cluster-Migshm impiega circa lo stesso tempo del cluster-OM1 per aprire le applicazioni, e quindi che l'aggiunta di migshm in questo stadio di sviluppo non produce effetti apprezzabili.

Il test su OpenOffice consiste nell'aprire una sola istanza dell'applicazione che comunque richiede moltissima RAM. Questo semplice esperimento ha mostrato che,nonostante la presenza di Migshm, OpenOffice rimane vincolato al nodo che l'ha generato senza che ci sia migrazione di processi ad esso legati. Nonostante questo, è stato riscontrato un comportamento del cluster molto interessante e cioè che tutti i processi che potevano essere migrati (compresi quelli delle applicazioni a memoria condivisa che erano state aperte in precedenza), venivano spostati su altri nodi del cluster per, consentire alla macchina impegnata nell'onerosa apertura di OpenOffice, di avere più risorse a disposizione.

Comportamento ancora diverso è assunto dal cluster-Migshm nell'apertura di The Gimp. Si è notato che durante l'avvio del programma alcuni processi migravano su altri nodi, venivano elaborati e poi tornavano indietro; solo a questo punto avveniva l'apertura dell'applicazione.

Come si è visto, i test puramente numerici che consentono di avere una misura

delle prestazioni non mostrano un vantaggio nell'utilizzo della patch Migshm. Il risultato piú interessante che possiamo ricavare dai test che abbiamo realizzato, è legato sostanzialmente al fatto che si sono prodotte migrazioni di processi che competono ad applicazioni a memoria condivisa, cosa, questa, che apre nuove strade allo sviluppo di openMosix. Appare chiaro che Migshm non è ancora matura e che ancora molti test e molta ricerca va fatta, prima di raggiungere quella stabilità che le consenta di poter essere inserita come parte integrante della release ufficiale di openMosix.

In quest'ottica i test che abbiamo condotto e quelli che continueremo a fare in futuro, sono molto importanti per il contributo che possono dare allo sviluppo di questo progetto.

4.4 I test su LTSP

L'ultima serie di test è stata realizzata sul sistema LTSP (Linux Terminal System Project). Questo, come detto nel capitolo-3, è un sistema thin client e perciò molto diverso da quello fin qui visto. E' un sistema che ha un'architettura meno avanzata rispetto a quella di openMosix, ma che è, per le finalità di questa tesi, di estrema importanza, poiché si presta molto bene ad essere utilizzato in tutti quei contesti in cui si hanno poche risorse disponibili.

L'ambiente su cui le prove sono state fatte è diverso rispetto ai precedenti; infatti è stata utilizzata una configurazione a due soli computer (server-client) come descritta nella sezione-4.1. Su questo sistema sono stati fatti test analoghi a quelli appena visti per il cluster-Migshm, consistenti nell'appertura di diverse istanze di alcune diverse applicazioni. In particolare sono state prese misure sul tempo necessario per l'avvio di Konqueror e di OpenOffice con le modalità già descritte nella sezione precedente. Dal PI sono state lanciate più volte (da una a quattro), due istruzioni analoghe a quelle utilizzate nel precedente test:

```
# time konqueror openMosix-HOWTO.txt & xkill &
# time ./soffice openMosix-HOWTO.txt & xkill &
```

Alla fine dell'apertura dell'ultima istanza sono stati presi i tempi impiegati

dal client per avviare le applicazioni. Questi tempi sono stati poi confrontati con quelli del PIV da solo, e con quelli del PI da solo.

La figura 4.11 mostra il tempo necessario al client per aprire diverse istanze di Konqueror, confrontato con quello che impiegherebbe se non fosse all'interno della rete LTSP e con quello impiegato dal PIV. Dal grafico si riscontra una netta diminuzione del tempo impiegato dal client LTSP rispetto a quello che si ha quando questo non si avvale delle risorse del server, cioè quando è al di fuori della rete LTSP. Inoltre, viene mostrato come le prestazioni del client stesso siano molto vicine a quelle del PIV.

Per quanto riguarda i tempi di OpenOffice, abbiamo un comportamento molto simile a quello appena messo in evidenza per l'applicazione precedente. E', però, importante sottolineare un aspetto fondamentale, e cioè che OpenOffice per poter essere installato e poter girare richiede una quantità di RAM ed una velocità del processore tali da non permettere ad un PI di poterlo utilizzare. Questo fatto è stato reso graficamente in figura facendo crescere le barre dei tempi del PI fino a farle uscire fuori scala.

Il commento che può essere fatto, è che i nostri test confermano che LTSP si presta molto bene ad essere utilizzato in tutti quei contesti in cui si hanno a disposizione risorse computazionali o di RAM molto scarse, consentendo di allestire laboratori informatici pienamente funzionanti, utilizzando macchine molto povere. L'aspetto forse più interessante è che LTSP consente di utilizzare applicazioni anche molto onerose (come OpenOffice) su macchine che, altrimenti, mai avrebbero potuto farne uso.

Con questo test si conclude questo capitolo e tutto il lavoro di studio e sperimentazione e si rimananda alla prossima sezione per le conclusioni, i commenti e le valutazioni sul lavoro futuro.

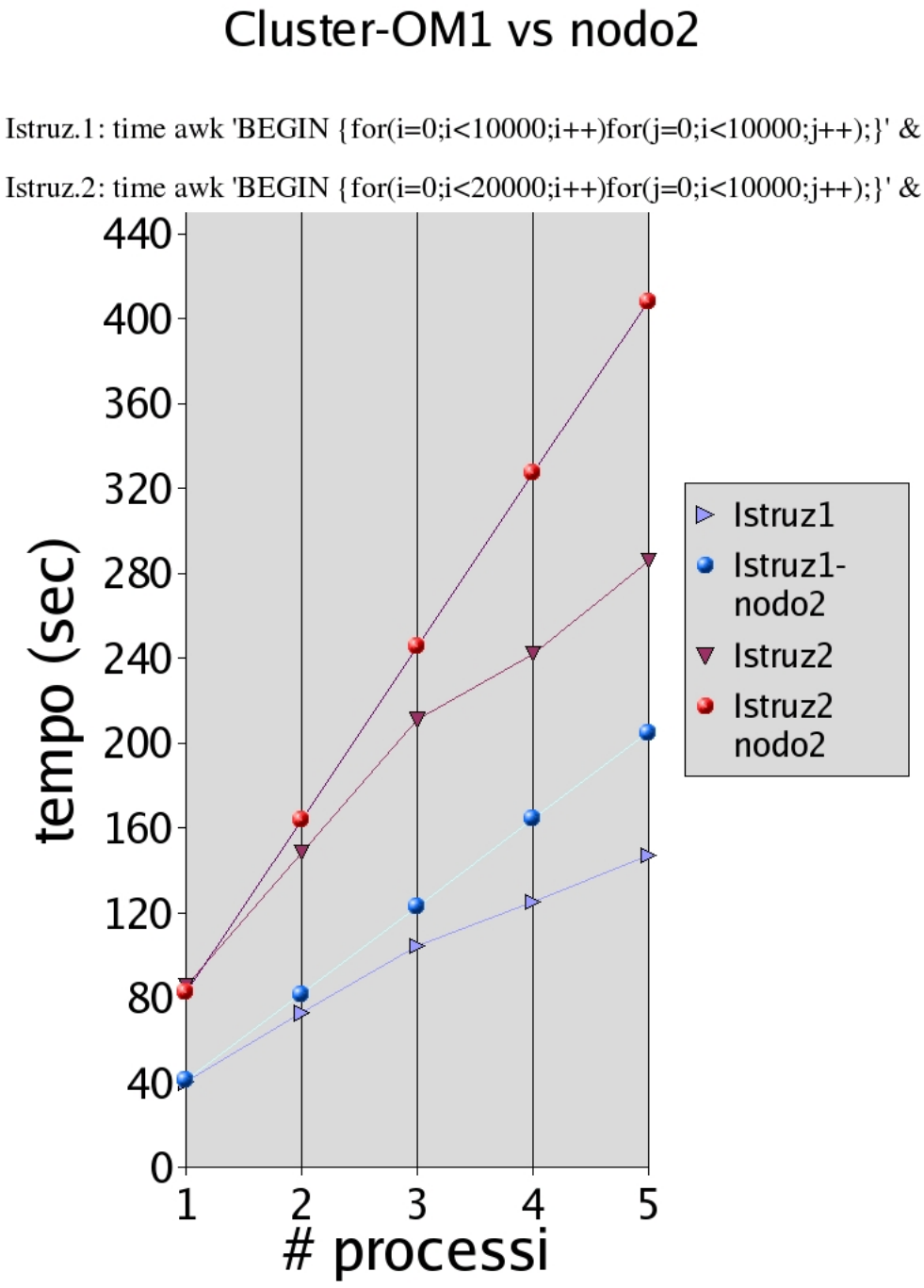

Figura 4.1: Tempo impiegato dal cluster-OM1 per eseguire le due istruzioni al variare del numero di processi lanciati, confrontato con il tempo impiegato dal nodo2.

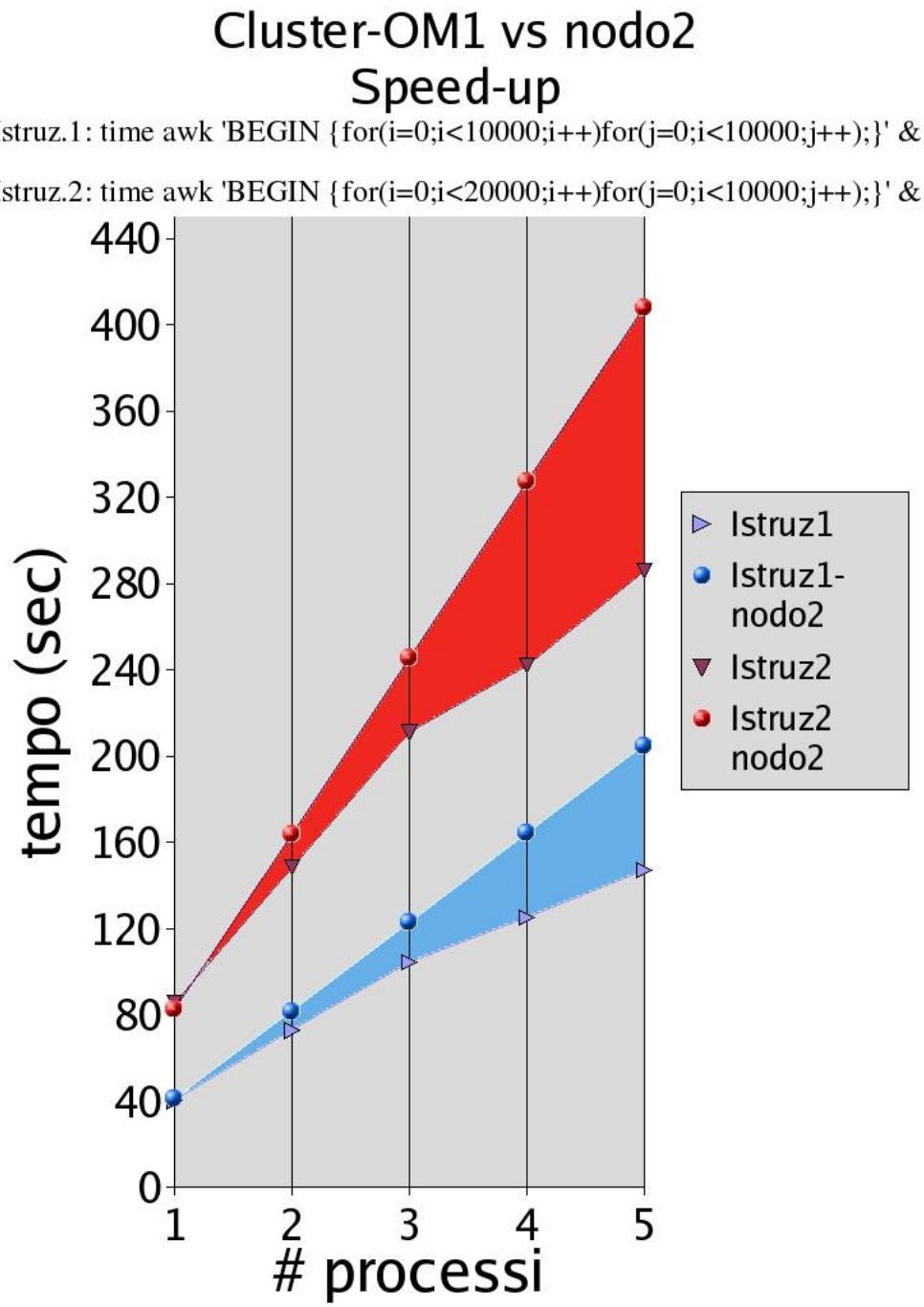

Figura 4.2: Speed up ottenuto con il cluster-OM1 rispetto al nodo2 da solo.

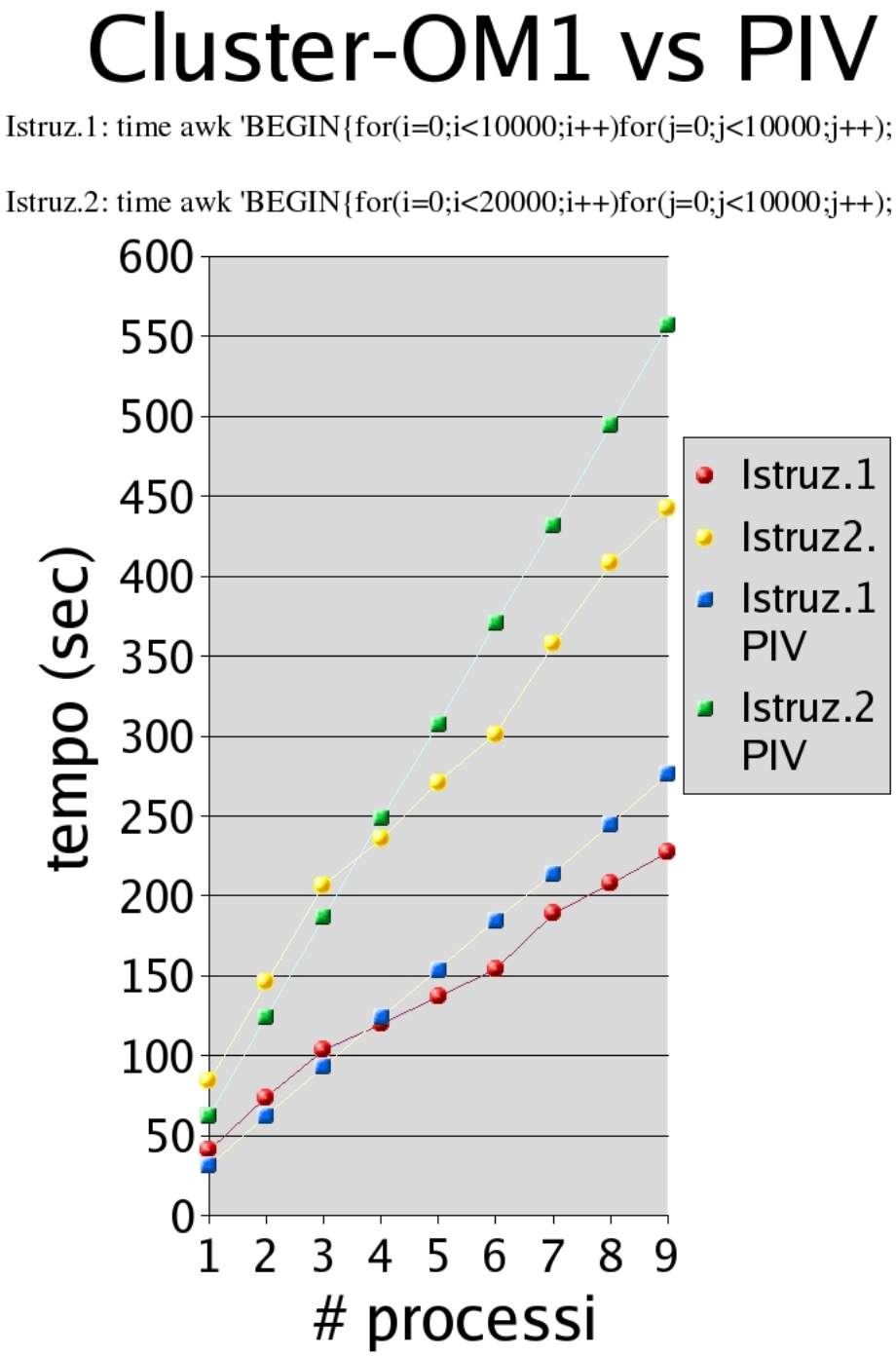

Figura 4.3: Confronto tra il cluster-OM1 di macchine povere ed il PIV.

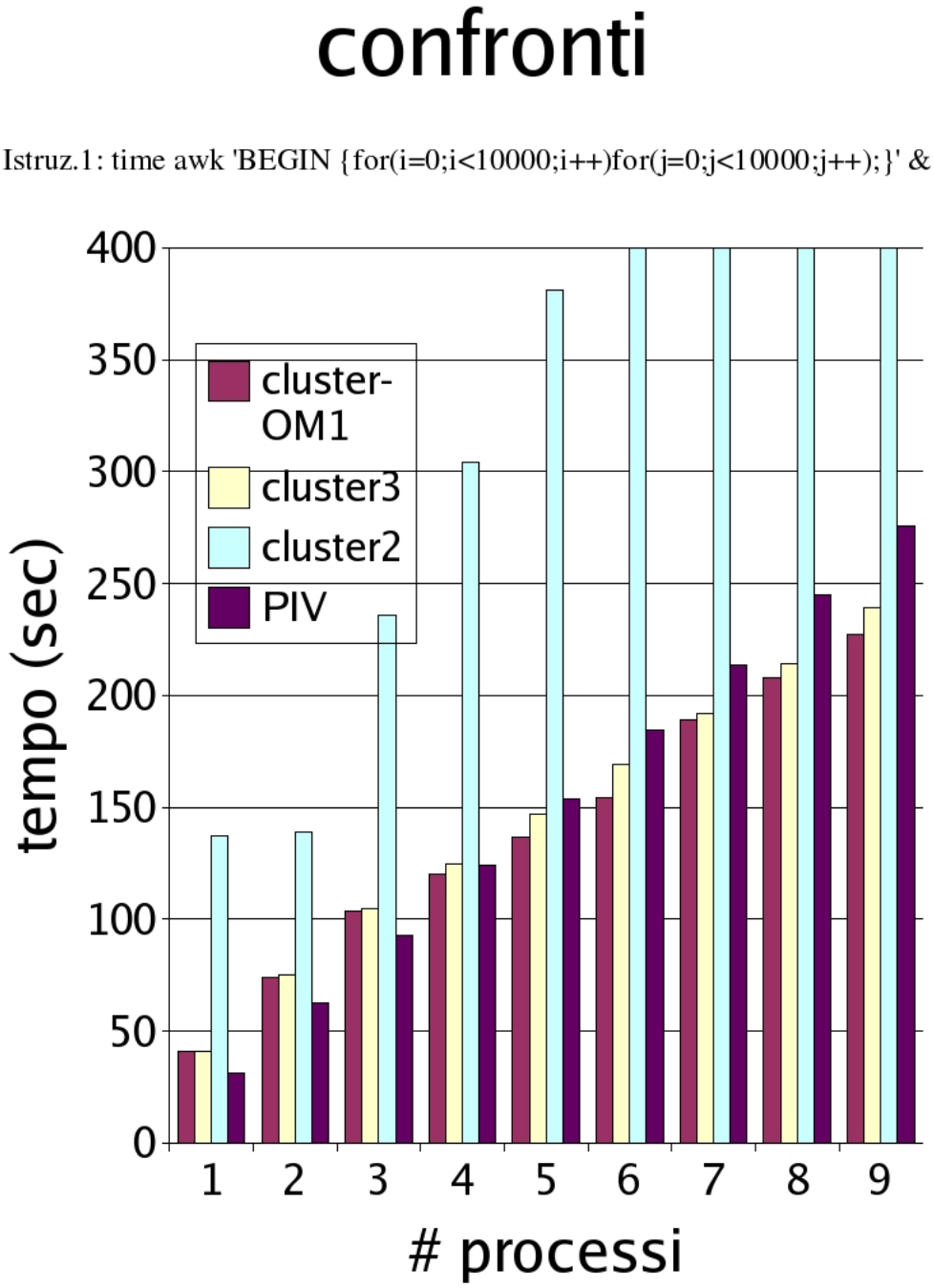

Figura 4.4: Confronto tra il cluster di macchine povere ed il PIV al variare del numero dei nodi del cluster.

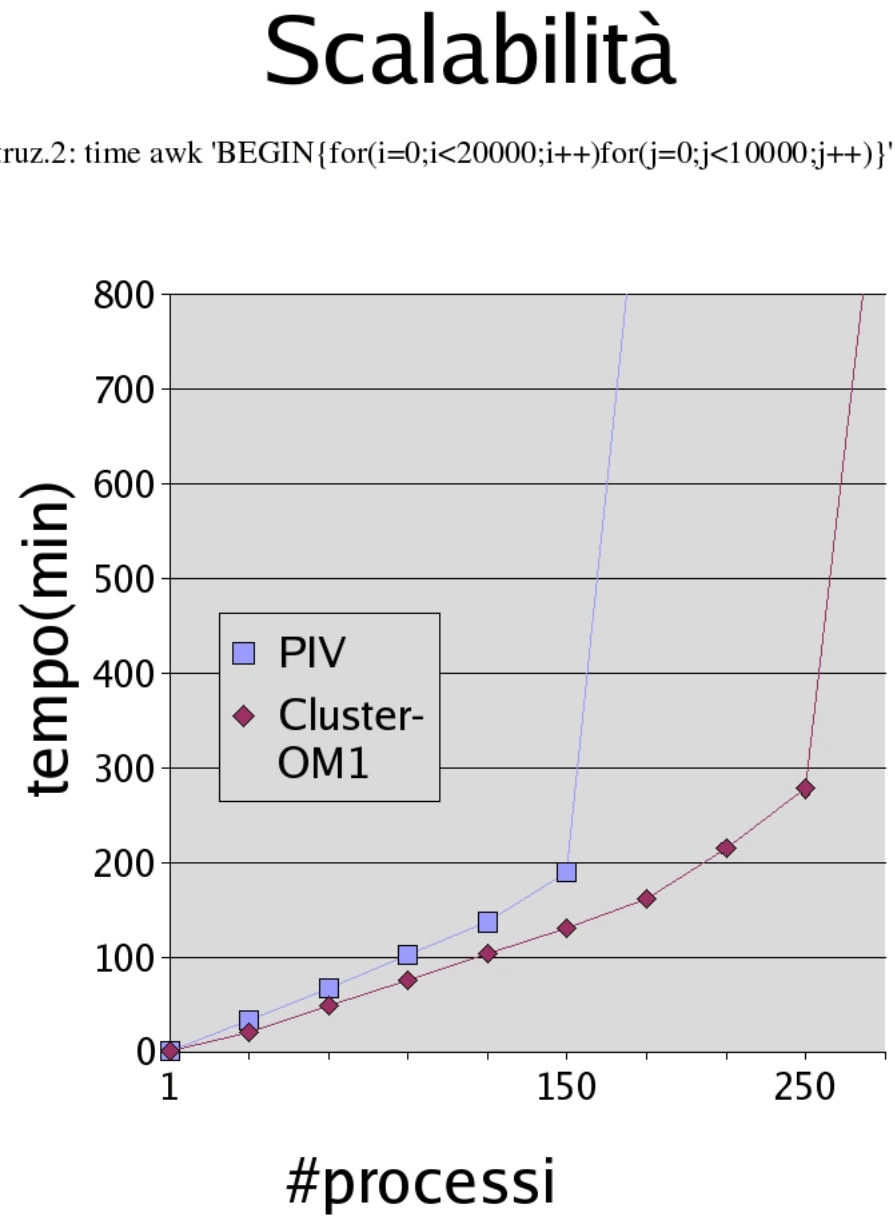

Figura 4.5: Confronto in termini di scalabilità tra il cluster di macchine povere ed il PIV.

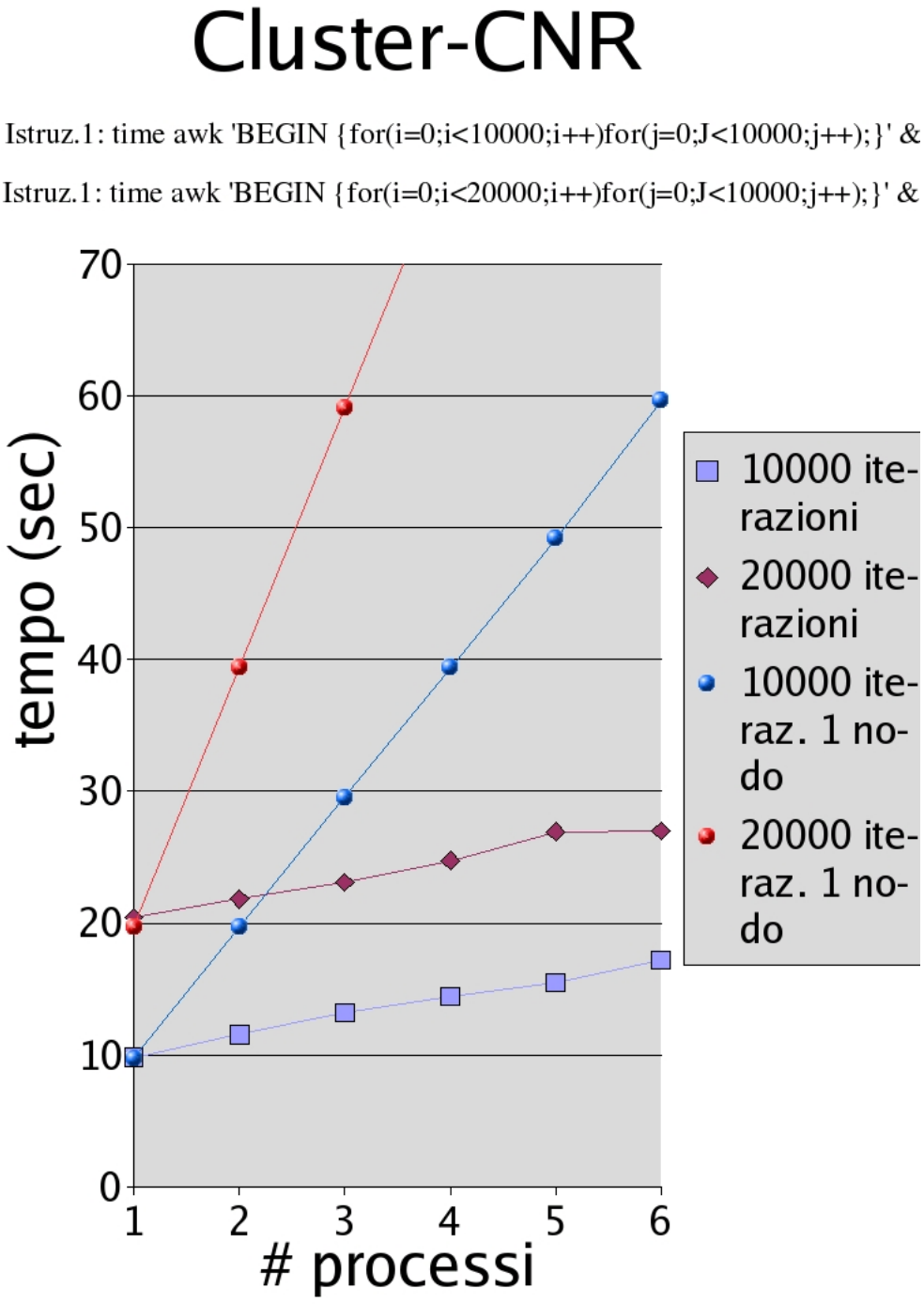

Figura 4.6: Tempo impiegato dal cluster-CNR per eseguire le due istruzioni al variare del numero di processi lanciati, confrontato con il tempo impiegato dalla singola macchina.

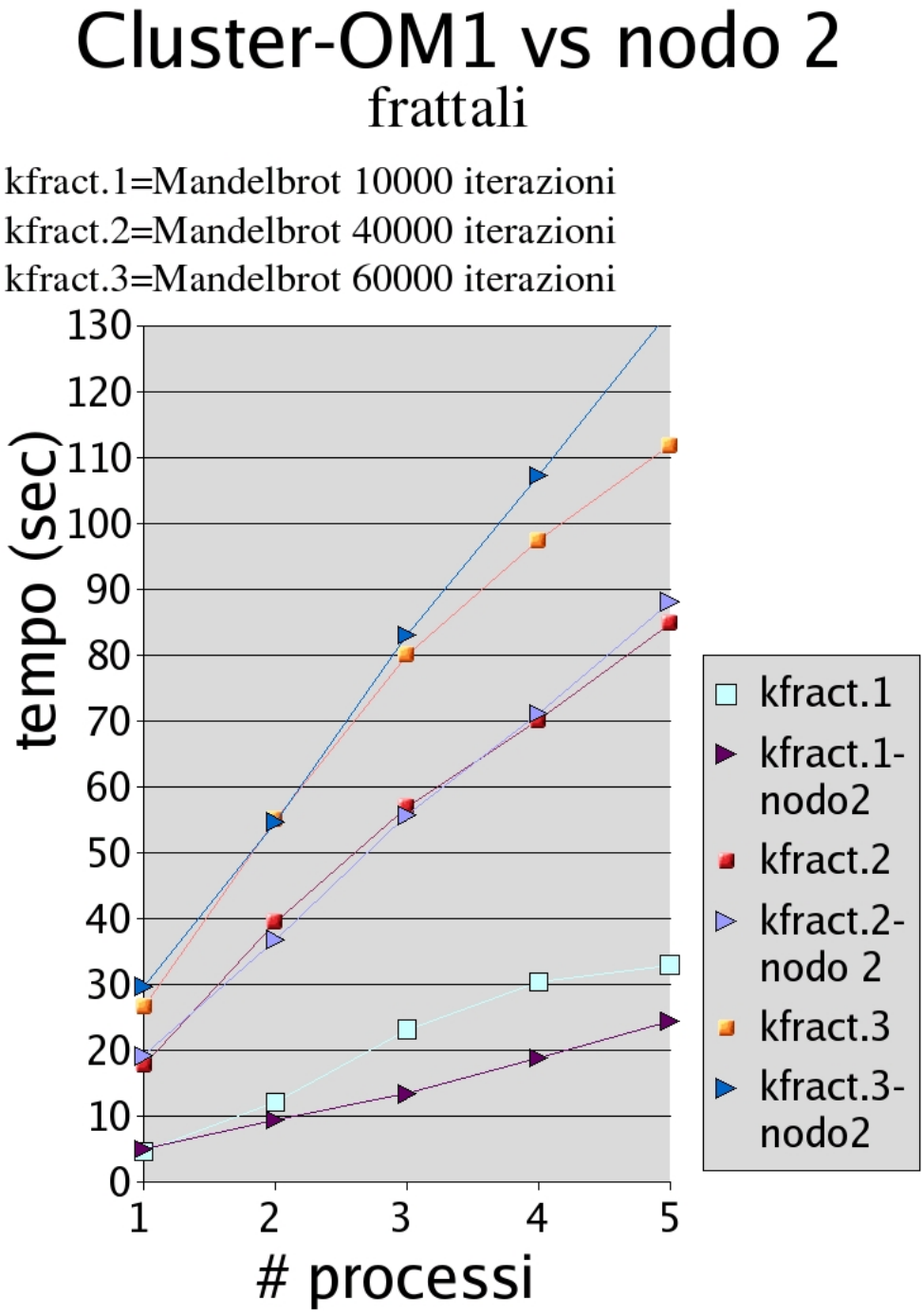

Figura 4.7: Tempo impiegato dal cluster-OM1 per eseguire le tre istruzioni al variare del numero di processi lanciati, confrontato con il tempo impiegato dal nodo 2.

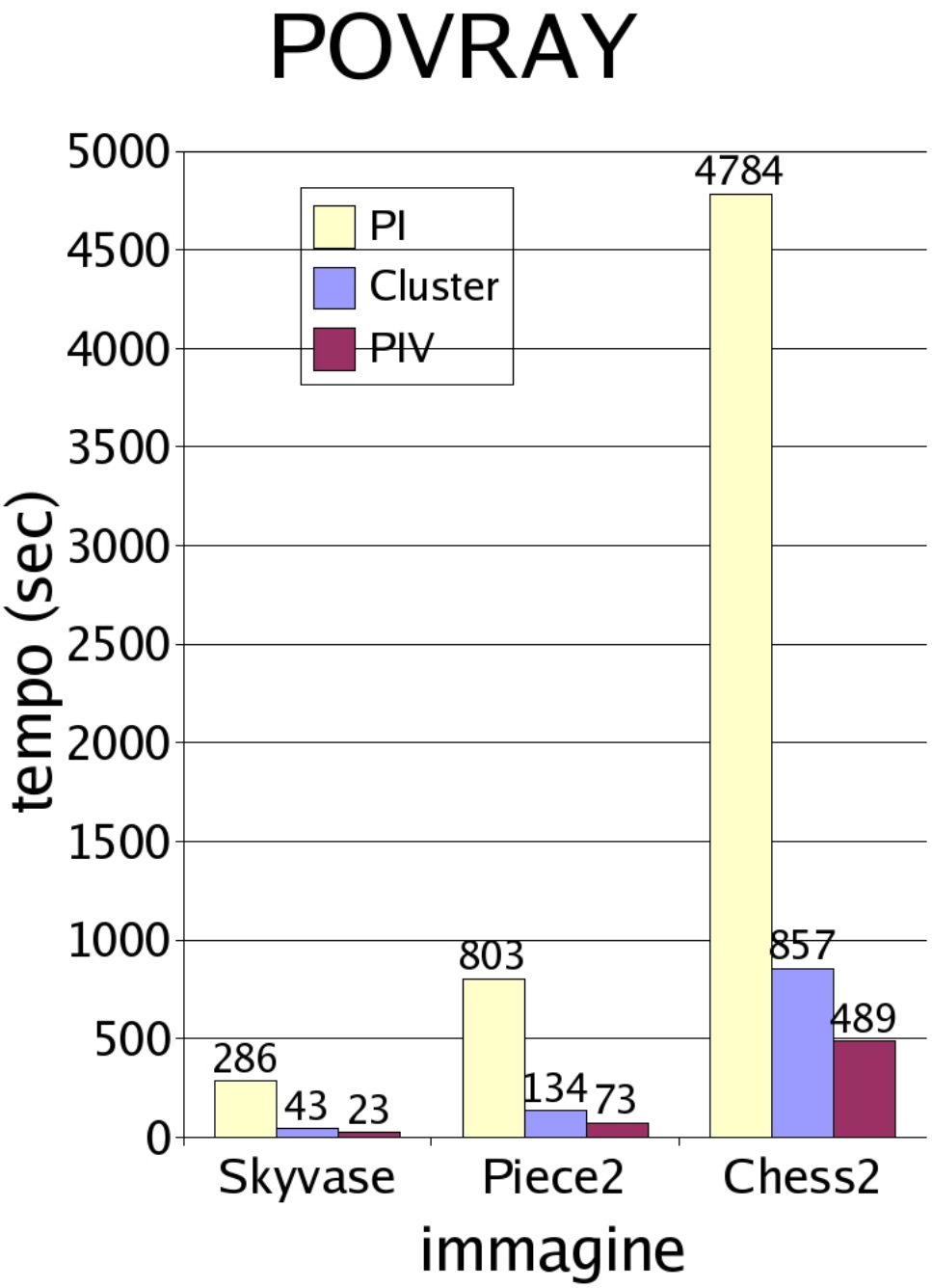

Figura 4.8: Tempo impiegato dal cluster-OM1 per eseguire la renderizzazione di tre immagini di complessità crescente, confrontato con il tempo impiegato dal PI (nodo4) e dal PIV.

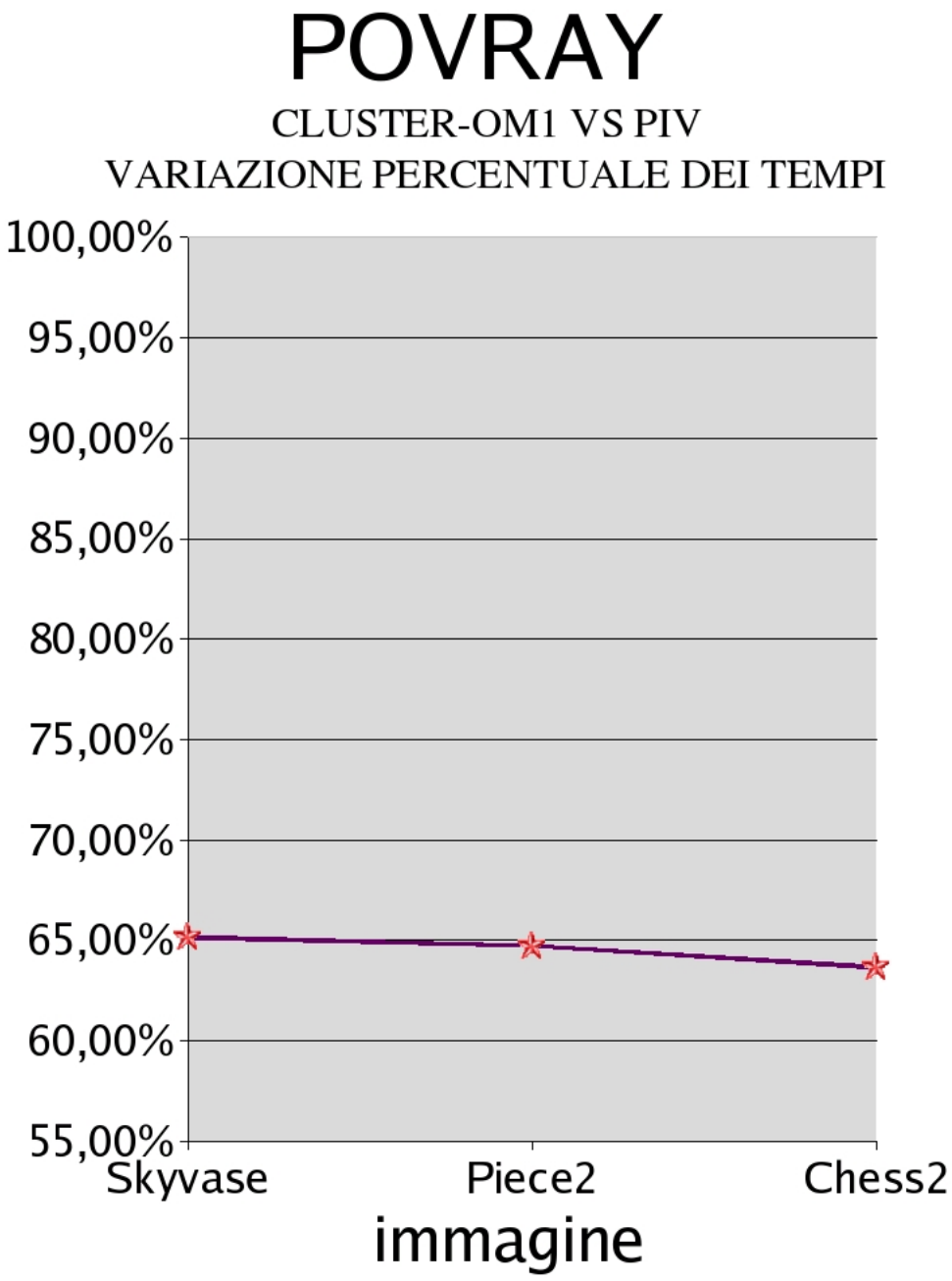

Figura 4.9: Variazione percentuale della differenza del tempo di esecuzione del job tra PIV e cluster-OM1.

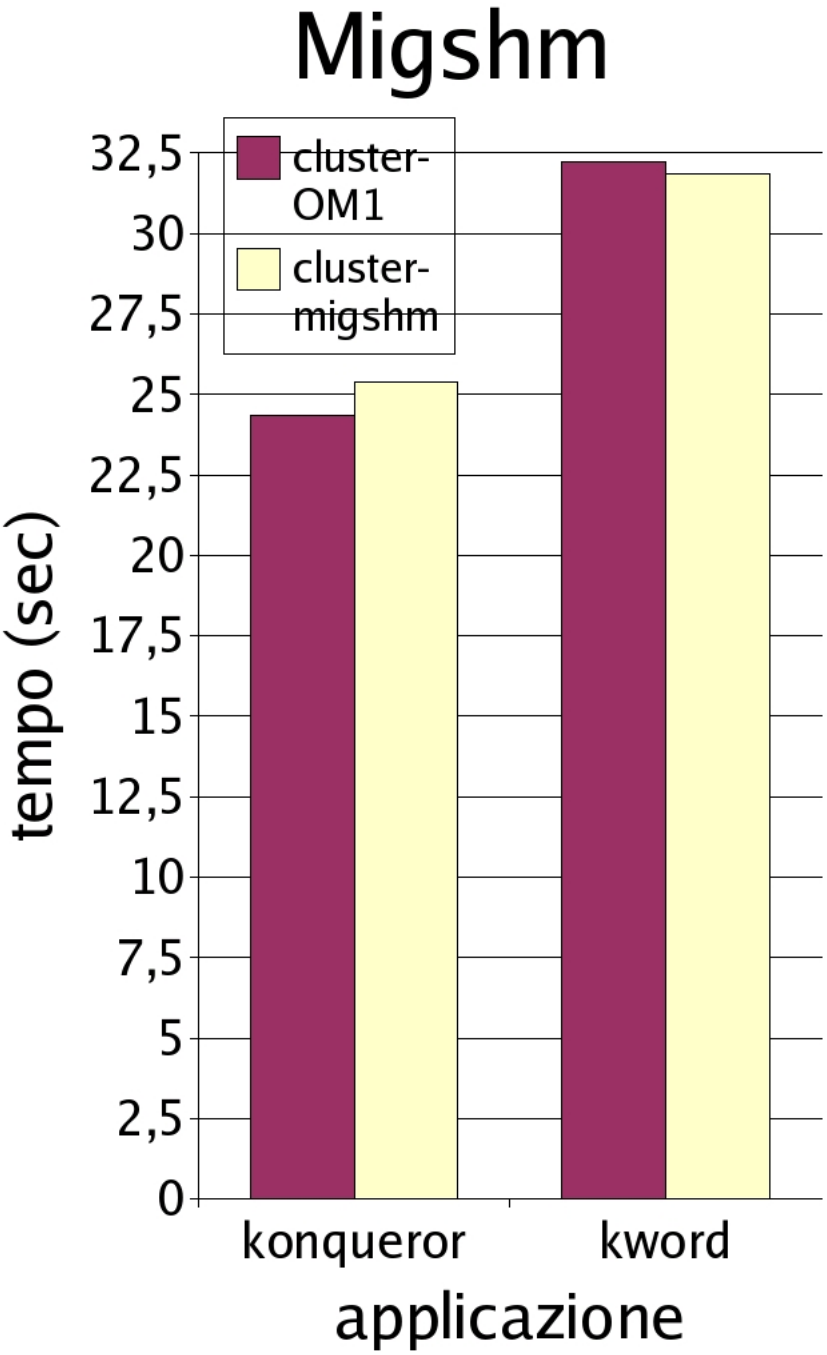

Figura 4.10: Confronto dei tempi di apertura di applicazioni a memoria condivisa tra un cluster openmosix tradizionale ed uno abilitato alla migrazione di processi shared-memory tramite Migshm.

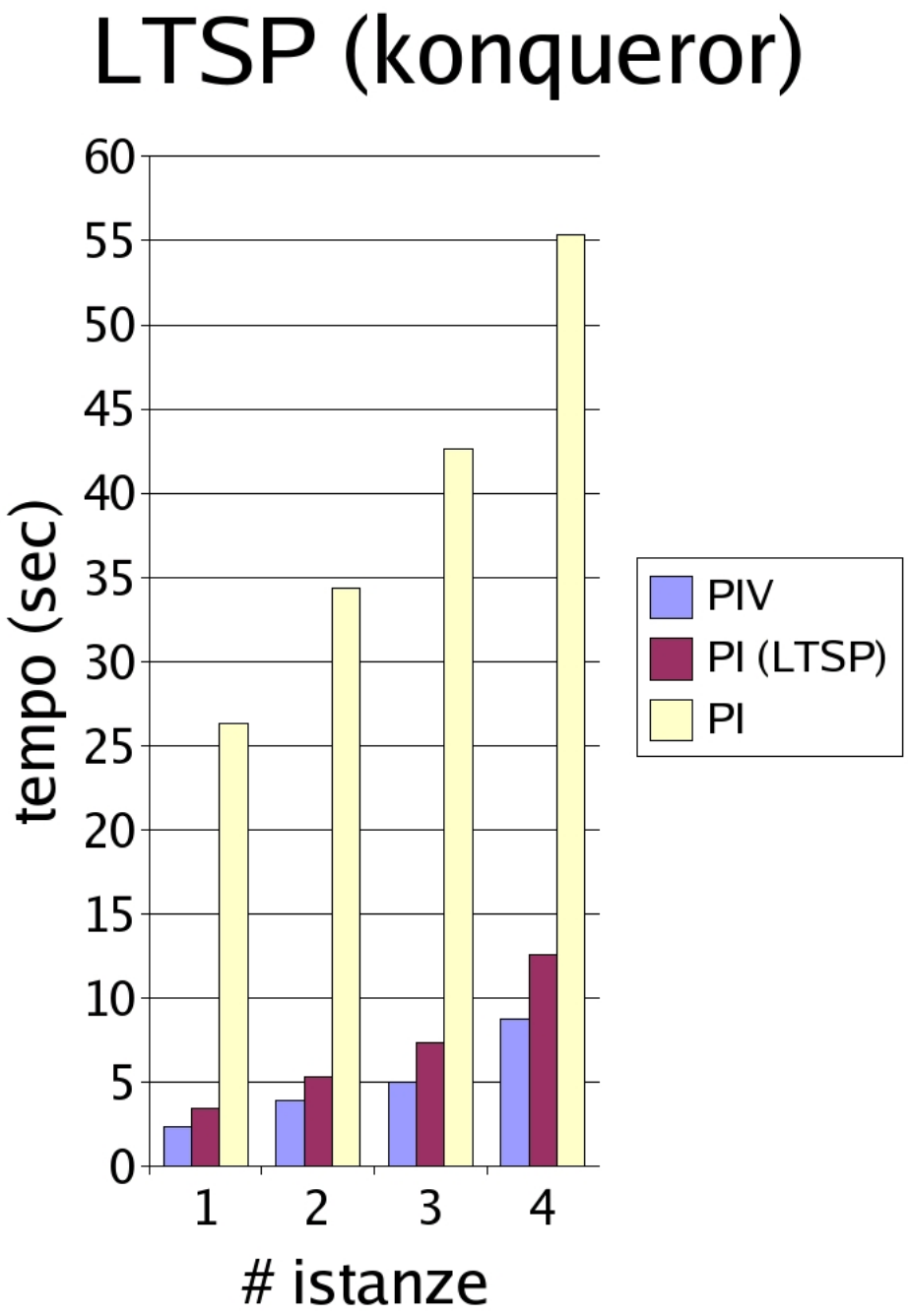

Figura 4.11: Tempo impiegato dal client LTSP per aprire diverse istanze di Konqueror confrontato con quello del PIV e con quello del PI.

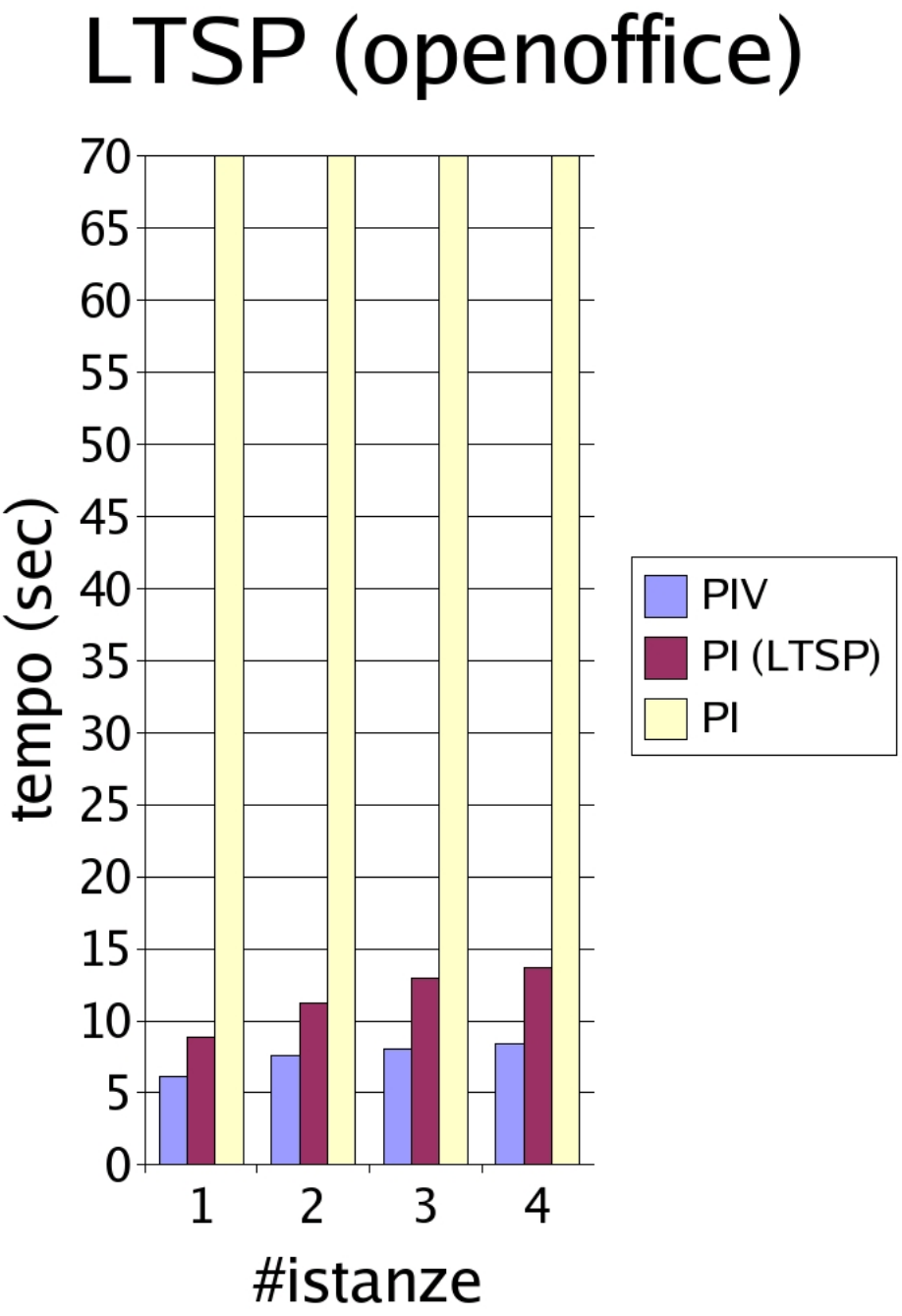

Figura 4.12: Tempo impiegato dal client LTSP per aprire diverse istanze di OpenOffice confrontato con quello del PIV e con quello del PI.

Conclusioni

Questo lavoro di tesi, proposto da Ingegneria Senza Frontiere (ISF) e sviluppato all'interno del progetto di riutilizzo di hardware datato (Trashware), in cui l'associazione è impegnata, ha prenso le mosse dalla necessità di trattare in maniera organica e congiunta tre tematiche fondamentali: il corretto utilizzo dell'hardware, con lo scopo di dare il giusto corso all'obsolescenza delle teconlogie informatiche; l'uso di tecnologie appropriate per progetti di cooperazione in paesi in via di sviluppo, nel tentativo di contribuire all'abbattimento del Divario Digitale; infine la speculazione accademica sulle piattaforme software distribuite.

Si è analizzata la situazione mondiale rispetto divario nella fruizione delle tecnologie tra Paesi del Sud e Paesi del Nord del Mondo. Evidenziando come a fronte di Nazioni in cui la popolazione dispone di meno di 1 computer ogni 100 abitanti, ci sono Paesi in cui vengono avviate a dismissione in discariche, macchine ancora perfettamente funzionanti ma considerate obsolete a causa di un mercato, quello dell'informatica, che impone hardware sempre più potente per poter far funzionare software sempre più sofisticato e viceversa. Questo lavoro, ha, in più, descritto il fenomemo del Trashware e mostrato come esso sia strettamente legato allo sviluppo del software Libero ed Open Source, mettendo particolare attenzione ad evidenziare il modo in cui il sistema GNU/Linux ed il modello di sviluppo che ne è alla base possono costituire il fulcro di progetti di cooperazione che abbiano come obiettivo il superamento del divario digitale.

La caratteristica realmente peculiare del progetto di Trashware che è stato sviluppato all'interno di questo lavoro di tesi, è data dal fatto che si affronta il problema del riuso dell'hardware obsoleto, attraverso la speculazione accademi-

ca riguardante lo studio della condivisione delle risorse di computazione di più macchine organizzate in modo da formare un cluster. Si cerca, in altre parole, di trattare il problema dell'obsolescenza dei computer e del loro riuso, in un modo del tutto originale, ottimizzando le risorse che si hanno a disposizione attraverso l'allestimento di batterie di computer in cui le macchine possano condividere risorse di calcolo e distribuirsi il carico computazionale. Dal punto di vista della speculazione accademica, quindi, si sono studiate alcune piattaforme software che per le loro caratteristiche e per il loro grado di sviluppo e diffusione meglio si prestavano a costituire il nostro ambiente di sperimentazione. In particolare si è studiato il funzionamento di openMosix e di LTSP in contesti di Trashware; lo si è fatto allestendo un laboratorio di macchine "povere" in cui sono stati eseguiti numerosi test con differenti configurazioni hardware e diverse versioni delle piattaforme software. Nei test che abbiamo eseguito, il nostro cluster povero è stato confrontato con architetture hardware moderne (un PIV 1700MHz).

Nelle prove fatte con openMosix si mostra che in tutte quelle applicazioni caratterizzate dall'elevato costo computazionale, il cluster riesce a raggiungere e superare il computer moderno, con uno speed-up che aumenta al crescere della complessità e del peso delle applicazioni.

Nonostante i risultati eccellenti che sono stati ottenuti, il sitema openMosix ha palesato un limite fondamentale rispetto alle finalità del nostro progetto: non era ideato in modo tale che anche le applicazioni che fanno uso di memoria condivisa (ovvero la maggior parte dei normali programmi di uso comune), potessero avvantaggiarsi degli algoritmi di condivisione delle risorse e di distribuzione del carico propri del sistema. In altre parole, openMosix, per come era progettato, non consentiva di distribuire su tutto il cluster, il carico computazionale dovuto alle applicazioni di uso comune. Per ovviare a questo problema è stato necessario utilizzare una versione di openMosix, modificata tramite l'aggiunta di un elemento totalmente sperimentale chiamato Migshm, che abbiamo a lungo testato per studiarne il funzionamento e per verificare che effettivamente il cluster, nella nuova configurazione, trattasse le applicazioni a memoria condivisa come fa con qualunque altro processo: consentendone la migrazione verso nodi diversi da

quello in cui l'applicazione stessa è stata lanciata.

Dalla sperimentazione, si è evidenziato che effettivamente Migshm consente di trasferire i processi che competono ad applicazioni utente, sui diversi nodi del cluster.

Dopo aver verificato che effettivamente i programmi di uso comune potevano migrare tra i nodi della rete di macchine, nei test che abbiamo condotto, si è cercato di valutare se si traeva un vantaggio da questo trasferimento,oppure no. Il risultato che abbiamo trovato e che abbiamo illustrato nei grafici del capitolo-4, dimostrano che non c'è un reale vantaggio dall'aggiunta di Migshm.

E' necessario comunque sottolineare che è, questo, un elemento del tutto nuovo, che rivoluziona il comportamento di openMosix, ma che ancora è in fase di test e sperimentazione e dunque non è ancoro molto maturo. Saranno necessari più esperimenti da fare su versioni aggiornate progettate per kernel più moderni, per avere una idea veramente chiara dell'impatto che Migshm potrà avere sullo sviluppo futuro di openMosix.

Risultati molto buoni, sono stati ottenuti dagli esperimenti che abbiamo condotto sul Linux Terminal Server Project (LTSP), che nasce appositamente per essere utilizzato in contesti caratterizzati da poche risorse economiche. E' per questo motivo che questo sistema ha un così grande successo in moltissimi progetti di cooperazione allo sviluppo o di alfabetizzazione informatica (per esempio, laboratori di informatica nelle scuole). Nelle prove che sono state fatte si è misurato il tempo che un Pentium I impiega ad aprire applicazioni di uso comune all'interno di una rete LTSP (in cui utilizza le risorse computazionali di un server). Tali tempi sono stati poi confrontati con quelli che la stessa macchina impiega da sola, riscontrando un grande miglioramento delle prestazioni. In LTSP i nodi della rete di calcolatori utilizzano le risorse computazionali del server, per questo è possibile per una macchina con pochissime risorse in termini di CPU e di RAM, utilizzare applicazioni che, in condizioni normali di funzionamento, non potrebbero neanche essere installate.

Lavoro futuro Nel futuro, il gruppo Trashware continuerà a condurre attività di ricerca e di sperimentazione, per ottenere soluzioni che consentano un

livello di ottimizzazione delle risorse sempre più alto, nel tentativo di raggiungere prestazioni sempre migliori con macchine obsolete. Ci sono diverse strade che il gruppo ha intenzione di percorrere.

La prima riguarda sicuramente la sperimentazione su Migshm; nel futuro se ne proveranno le nuove versioni che nel frattempo sono state prodotte o che usciranno nei prossimi mesi. Si proveranno nuovi kernel più efficienti e maturi nel desiderio di fornire anche un contributo attivo allo sviluppo del progetto, fornendo un aiuto a quella comunità di sviluppatori che ha seguito il nostro lavoro di tesi.

La seconda via che vorremmo percorrere è quella di studiare in che modo è possibile fondere i sistemi LTSP ed openMosix ottenendo così un cluster ibrido che possa unire le caratteristiche di entrambi i sistemi cioé la scalabilità e l'efficienza di openMosix, con la semplicità realizzativa di LTSP.

L'ultima strada, infine, ancora tutta da valutare e da sviluppare consiste nella fusione di openMosix con un progetto chiamato User Mode Linux (UML). UML è uno strumento che rende possibile far girare Linux dentro se stesso in modo sicuro. Dà la possibilità di installare e provare software con bachi, di sperimentare nuovi kernel e nuove distribuzioni, osservare le parti più interne del kernel Linux, senza che il setup della nostra macchina possa correre il minimo rischio. In altre parole, è una macchina virtuale che può avere risorse hardware e sofware maggiori rispetto a quelle del computer fisico. Alla macchina viene consentito accesso limitato alle risorse hardware, in modo tale che nulla di quello che viene fatto sulla macchina virtuale, possa danneggiare il computer reale o il suo software. La fusione di openMosix con UML, consentirebbe di ottenere un cluster virtuale openMosix che funziona in modalità utente, premettendo uno studio approfondito del sistema e garantendo la possibilità di modificarlo senza che il cluster rischi di essere danneggiato.

Appendice A

Graduatoria

Come visto nel primo capitolo Orbicom ha stilato nell'ottobre del 2003 un graduatoria dello sviluppo tecnologico di tutti gli stati sulla base del valore dell'Infostate.

Di seguito mostriamo la calssifica aggiornata al 2001.

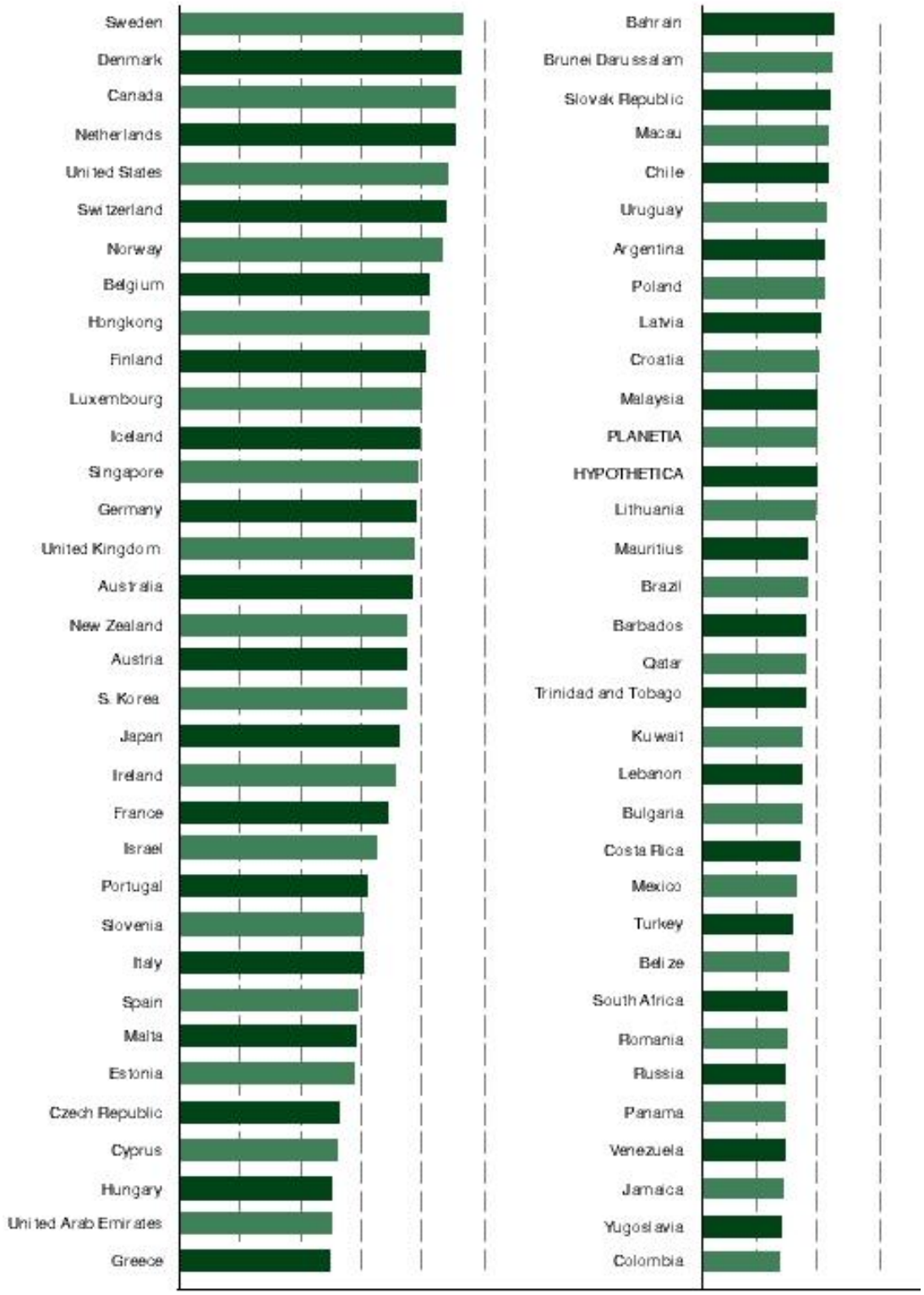

Figura A.1: Graduatoria.

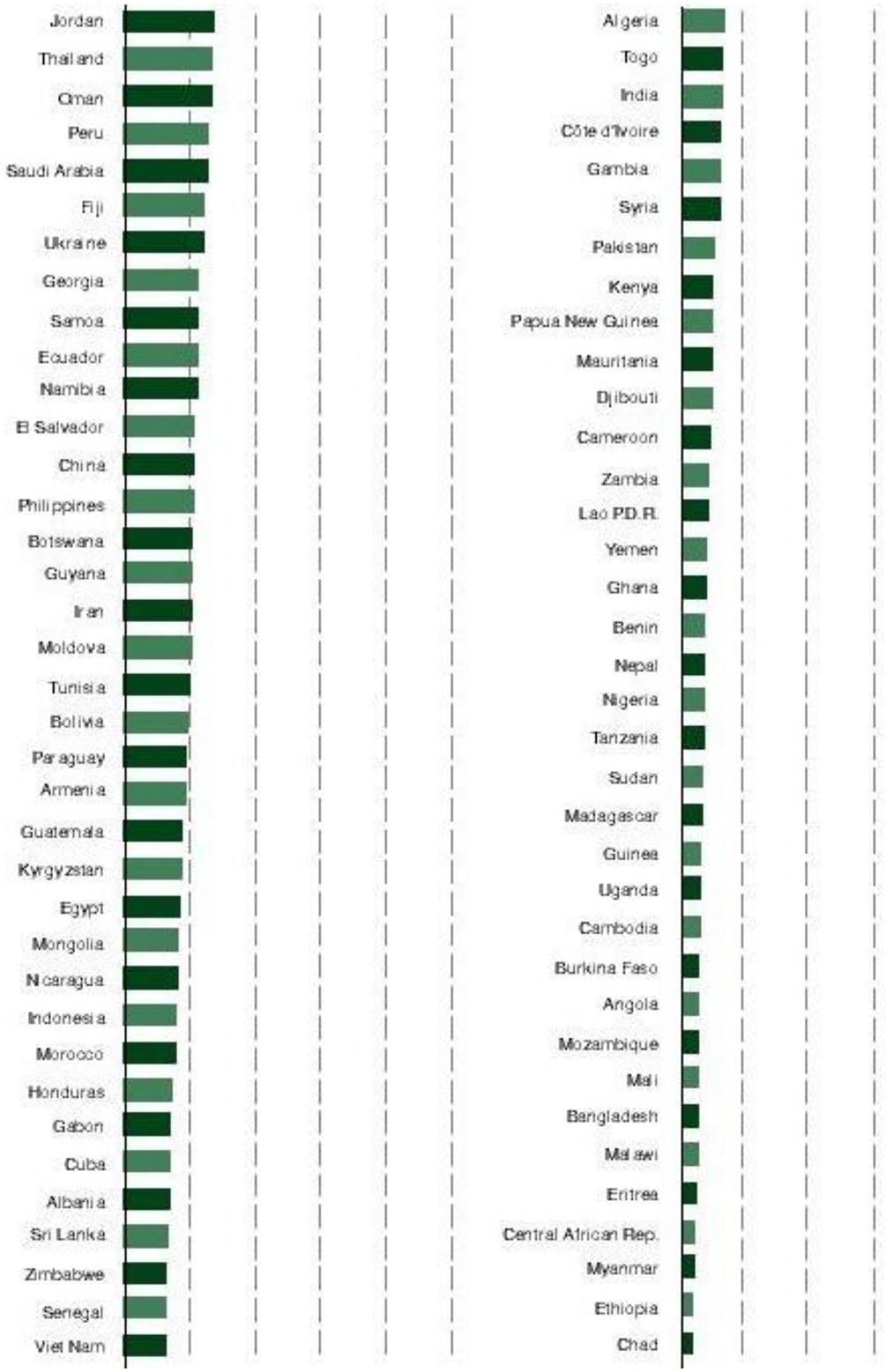

Figura A.2: Graduatoria.

Appendice B

La normativa Europea

Direttiva 2003/108/Ce (Rifiuti di apparecchiature elettriche ed elettroniche - modifiche alla direttiva 2002/96/Ce)

Direttiva 8 dicembre 2003, n. 2003/108/Ce (Gazzetta ufficiale dell'Unione europea 31 dicembre 2003 n. L 345)

Direttiva 2003/108/Ce del Parlamento europeo e del Consiglio dell'8 dicembre 2003 che modifica la direttiva 2002/96/Ce sui rifiuti di apparecchiature elettriche ed elettroniche (Raee)

Il Parlamento europeo e il Consiglio dell'Unione europea,

visto il trattato che istituisce la Comunità europea, in particolare l'articolo 175, paragrafo 1,

vista la proposta della Commissione,

visto il parere del Comitato economico e sociale europeo,

previa consultazione del Comitato delle regioni,

deliberando secondo la procedura di cui all'articolo 251 del trattato,

considerando quanto segue:

(1) Durante la procedura di adozione della direttiva 2002/96/Ce del Parlamento europeo e del Consiglio, del 27 gennaio 2003, sui rifiuti di apparecchiature elettriche ed elettroniche (Raee), hanno destato preoccupazioni le possibili conseguenze finanziarie dell'enunciato dell'articolo 9 della medesima direttiva per i produttori delle apparecchiature interessate.

(2) Nella riunione del Comitato di conciliazione del 10 ottobre 2002 dedicata a detta direttiva, il Parlamento europeo, il Consiglio e la Commissione hanno espresso l'intenzione, in una dichiarazione congiunta, di esaminare quanto prima le questioni relative all'articolo 9 della direttiva 2002/96/Ce concernenti il finanziamento relativo ai Raee provenienti da utenti diversi dai nuclei domestici.

(3) Ai sensi della dichiarazione congiunta, la Commissione ha esaminato le conseguenze finanziarie per i produttori derivanti dall'enunciato attuale dell'articolo 9 della direttiva 2002/96/Ce e ne ha concluso che l'obbligo di ritiro di Raee immessi sul mercato in passato crea un onere retroattivo non considerato che potrebbe esporre determinati produttori a gravi rischi economici.

(4) Per prevenire tali rischi, è opportuno che la responsabilità finanziaria per la raccolta, il trattamento, il riutilizzo, il recupero e il riciclaggio dei Raee provenienti da utenti diversi dai nuclei domestici immessi sul mercato prima del 13 agosto 2005 incomba ai produttori che forniscono prodotti nuovi in sostituzione di prodotti di tipo equivalente o adibiti alle medesime funzioni. Ove tali rifiuti non siano sostituiti da prodotti nuovi, la responsabilità dovrebbe ricadere su detti utenti. Gli Stati membri, i produttori e gli utenti possono stipulare accordi alternativi.

(5) Ai sensi dell'articolo 17 della direttiva 2002/96/Ce, gli Stati membri devono mettere in vigore le disposizioni legislative, regolamentari e amministrative necessarie per conformarsi alla direttiva entro il 13 agosto 2004. Per evitare che sia necessario modificare atti legislativi adottati dagli Stati membri entro quella data, è opportuno adottare la presente direttiva quanto prima, in modo che sia recepita negli ordinamenti nazionali degli Stati membri contemporaneamente alla direttiva 2002/96/Ce.

(6) E' opportuno modificare di conseguenza la direttiva 2002/96/Ce,

hanno adottato la presente direttiva:

Articolo 1

L'articolo 9 della direttiva 2002/96/Ce sostituito dal seguente:

"Articolo 9

Finanziamento relativo ai Raee provenienti da utenti diversi dai nuclei domestici

1. Gli Stati membri provvedono affinché entro il 13 agosto 2005 i produttori debbano prevedere il finanziamento dei costi di raccolta, trattamento, recupero e smaltimento ecologicamente corretto dei Raee provenienti da utenti diversi dai nuclei domestici e originati da prodotti immessi sul mercato dopo il 13 agosto 2005.

Gli Stati membri provvedono affinché entro il 13 agosto 2005, per Raee di prodotti immessi sul mercato prima del 13 agosto 2005 ("rifiuti sotorici"), il finanziamento dei costi di gestione obbedisca alle modalit di cui al terzo e quarto comma.

Per i rifiuti storici sostituiti da nuovi prodotti equivalenti o da nuovi prodotti adibiti alla medesima funzione, il finanziamento dei costi incombe ai produttori di detti prodotti all'atto della fornitura. Gli Stati membri possono, in alternativa, disporre che gli utenti diversi dai nuclei domestici siano resi anch'essi parzialmente o totalmente responsabili di tale finanziamento.

Per gli altri rifiuti storici, il finanziamento dei costi incombe agli utenti diversi dai nuclei domestici.

2. I produttori e gli utenti diversi dai nuclei domestici possono, fatta salva la presente direttiva, concludere accordi che stabiliscano altre modalit di finanziamento."

Articolo 2

Gli Stati membri mettono in vigore le disposizioni legislative, regolamentari e amministrative necessarie per conformarsi alla presente direttiva entro il 13 agosto 2004. Essi ne informano immediatamente la Commissione.

Quando gli Stati membri adottano tali disposizioni, queste contengono un riferimento alla presente direttiva o sono corredate di un siffatto riferimento all'atto della pubblicazione ufficiale. Le modalità di tale riferimento sono decise dagli Stati membri.

Articolo 3

La presente direttiva entra in vigore il giorno della pubblicazione nella Gazzetta ufficiale dell'Unione europea.

Articolo 4

Gli Stati membri sono destinatari della presente direttiva.

Fatto a Bruxelles, l'8 dicembre 2003.

Appendice C

Installazione del software e monitoraggio del cluster

C.1 Installazione e configurazione di openMosix

Per prima cosa sono stati scaricati dalla rete i sorgenti del kernel 2.4.22 ottenuti da http://www.kernel.org sono stati copiati nella directory /usr/src e scompattati. L'operazione genera una directory linux-2.4.22/ all'interno della quale abbiamo tutti i sorgenti del kernel. Successivamente è stato copiato in /usr/src/ il file openmosix-2.4.22.bz2 scaricato da uno dei mirror di http://sourceforge.net. A questo punto sono stati patchati i sorgenti del kernel con il comando:

bzcat /usr/src/openmosix-2.4.22.bz2 | $patch - Np1$

Dopo essere stato patchato, il kernel è stato configurato e ricompilato abilitando le voci specifiche di openMosix oltre a quelle specifiche di ciascuna macchina tramite le istruzioni:

make mrproper

make clean

make menuconfig

L'ultima istruzione in particolare avvia un menù che consente di configurare il kernel openMosix e di adattarlo alle esigenze hardware del cluster che ha fatto da test-bed. Sono state abilitate, quindi le voci specifiche di openMosix:

(*) openMosix process migration support

( ) Support cluster with a complex nethwork topology

(*) Stricter security on openMosix ports

(3)Level of process-identity disclosure (0-3)

(*) openMosix File-system

(*) Poll/Select exceptions on pipes

( ) Disable OOM Killer

( ) Load Limit

Una volta conclusa l'operazione di scelta dei moduli di cui avremmo avuto bisogno è seguita la compilazione vera e propria del kernel con le istruzioni:

make dep

make clean

make bzImage

make modules

make modules_install

Succesivamente è stato installato il kernel patchato, e modificato il **lilo** per poterlo avviare:

cp arch/i386/boot/bzImage /boot/vmlinuzOM

cp System.map /boot/System.map-2.4.22-OM

vi /etc/lilo.conf

In lilo è stato necessario aggiungere le righe che ci hanno consentito di avviare il nuovo kernel:

image=/boot/vmlinuzOM

label=Linux-OM

read-only

Il lilo è stato successivamente testato e reinstallato.

Per poter far partire openMosix sul nuovo kernel sono necessari alcuni tool di gestione e monitoraggio che sono stati installati con i seguenti comandi:

tar xvfz openmosix-tools-0.3.5.tar.bz2

cd openmosix-tools-0.3.5/

./configure

make

make install

Questi tools generano per prima cosa lo script con il quale abbiamo potuto lanciare openMosix sul generico nodo del cluster, e poi forniscono alcuni strumenti per il monitoraggio che vedremo più avanti in questa appendice.

L'ultima operazione che è stata eseguita è stata editare il file openmosix.map in cui sono contenuti gli indirizzi IP dei nodi del cluster.

Il file del cluster è fatto come segue:

# openmosix-node_ID	IP-Address (or hostname)	Range-size
1	192.168.0.1	1
2	192.168.0.2	1
3	192.168.0.3	1
4	192.168.0.5	1

Con questa operazione si conclude la parte che illustra l'installazione e la configurazione del cluster.

C.1.1 Monitoraggio del Cluster

Per monitorare l'ambiente di sperimentazione sono stati utilizzati molti strumenti; alcuni forniti direttamente con openMosix (come gli openmosix_tools di cui ho mostrato l'installazione nel precedente paragrafo), ed altri ideati e sviluppati dalla nutritissima comunità di developers che si è raccolta intorno al progetto openMosix.

Il pacchetto openmosix-tools fornisce quattro strumenti di gestione mosctl, mosmon, mosrun, mps.

- mosmon: monitorizza lo stato del cluster, fornendo un grafico aggiornato in tempo reale sulle condizioni di carico dei diversi nodi o sull'occupazione di memoria dei diversi processori.

- mosctl: tool di amministrazione del cluster, consente di gestire la migrazione dei processi verso determinati nodi del cluster e l'accesso al filesystem di openmosix

- mosrun: esegue un comando su un particolare nodo del cluster
- mps: costituisce un potenziamento di ps, offrendo la possibilità di vedere tutti i processi lanciati nel cluster e i nodi su cui si trovano.
- setpe: consente la gestione del pool di indirizzi del cluster (mappatura dei nodi)

Tra tutti questi strumenti mosmon è quello che abbiamo utilizzato più diffusamente nel nostro studio sul cluster e nei nostri test poichè consente di osservare in modo semplice ed immediato lo stato di utilizzo di un nodo.

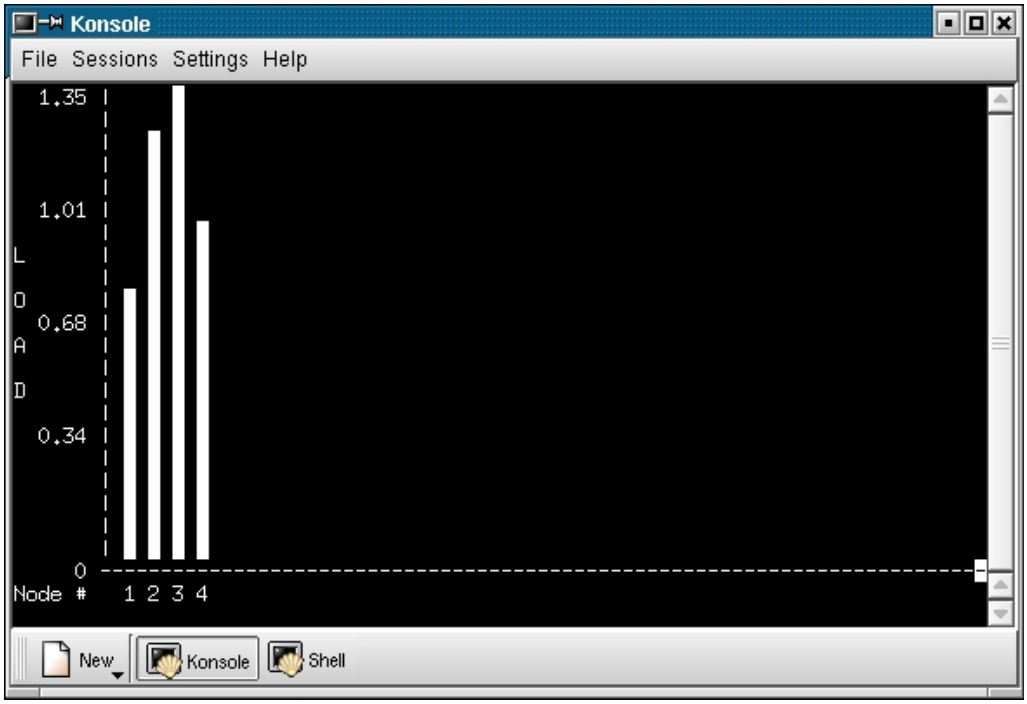

Figura C.1: mosmon

Oltre a quelli menzionati è stato utilizzato un altro strumento molto importante che è mtop. Questa applicazione rappresenta la versione per il cluster di "top" ,applicazione che fornisce in tempo reale istantanee dellattivit del processore. In particolare mostra il numero totale di processi in esecuzione ed il loro

stato (running, sleeping, stopped), il PID (process ID) di ciascuno, da quanto tempo sono in esecuzione, ecc ecc. Mtop si differenzia da top per il fatto che mostra queste informazioni non per il singolo nodo ma per tutto il cluster, perciò viene fatto vedere anche il numero del nodo in cui un processo sta girando, ed il numero di migrazioni tra nodi del cluster che il task ha effettuato.

Come detto all'inizio di questo paragrafo, oltre a questi tools utente la comunità openMosix ha sviluppato una suite composta da cinque applicazioni per la gestione, il monitoraggio ed il controllo del cluster con veste grafica molto più accattivante e più facilmente interpretabili: **openmosixview** . Gli applicativi contenuti nel pacchetto sono:

- **openMosixview**: l'applicazione principale di amministrazione e monitoraggio
- **openMosixprocs**: applicazione per la gestione dei processi
- **openMosixcollector**: applicazione per raccolta delle informazioni sullo stato dei nodi e dell'intero cluster
- **openMosixanalyser**: applicazione per analizzare i dati ottenuti da openMosixcollector
- **openMosixhistory**: applicazione che registra la storia dei processi del cluster

Tutte le applicazioni sono accessibili dalla finestra di openMosixview, le cui caratteristiche sono mostrate nella figuraC.2. In essa si possono vedere tre file, ciascuna delle quali consta di diverse parti. Percorrendo idealmente la finestra da sinistra a destra troviamo un rettangolo il cui colore ci dice se quel nodo fa parte del cluster (verde) oppure no (rosso). Il numero all'interno del rettangolo rappresenta l'ID del nodo che viene monitorato. A fianco abbiamo un bottone con l'indirizzo IP del nodo; spingendolo si apre una finestra di dialogo con la quale possiamo eseguire delle operazioni sui processi come per esempio veicolarne la migrazione verso specifici nodi. Percorrendo la riga abbiamo un cursore che

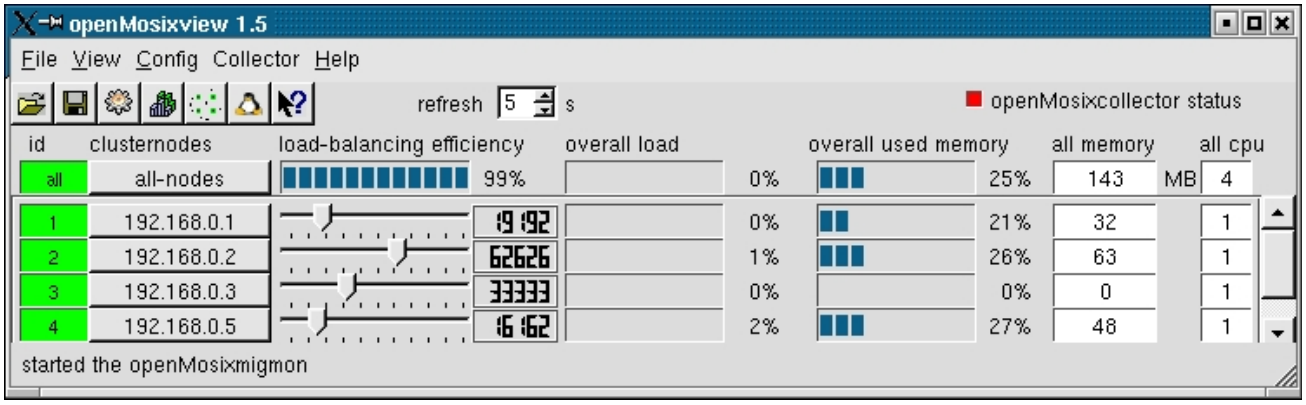

Figura C.2: openMosixview-finestra principale

consente di modificare la velocità di ciascun nodo, fissando un valore che viene indicato dal numero LCD che troviamo immediatamente dopo. Successivamente abbiamo due barre di avanzamento che indicano per ciascun nodo le percentuali di carico complessivo e di memoria utilizzata. Infine le ultime due informazioni si riferiscono alla RAM di ogni nodo ed al numero di CPU presenti.

Altro tool molto utile è **openMosixmigmon**. Si tratta di un monitor per le migrazioni all'interno del cluster openMosix. Mostra tutti i nodi come piccoli pinguini seduti in cerchio intorno ad un nodo centrale che è quello su cui gira openMosixmigmon. I processi che competono al nodo centrale sono disposti in circolo intorno ad esso. Quando un processo migra verso un altro nodo, viene indicato con il colore verde e di dispone in cerchio intorno al nodo verso cui si è spostato. Questo strumento consente non solo di osservare le migrazioni, ma anche di conoscere il PID del processo e soprattutto di spostare a piacimento i processi in qualunque degli altri nodi semplicemente con l'utilizzo del mouse.

Nell'ambiente così configurato sono stati effettuati i test descritti nel capitolo-4.

Come detto nello stesso capitolo, openMosix fornisce delle ottime prestazioni in tutte quelle applicazioni che eseguono molti calcoli, ma non consente la migrazione dei processi che competono alle applicazioni che fanno uso di memoria condivisa. Per consentire a queste applicazioni di migrare è necessario aggiungere

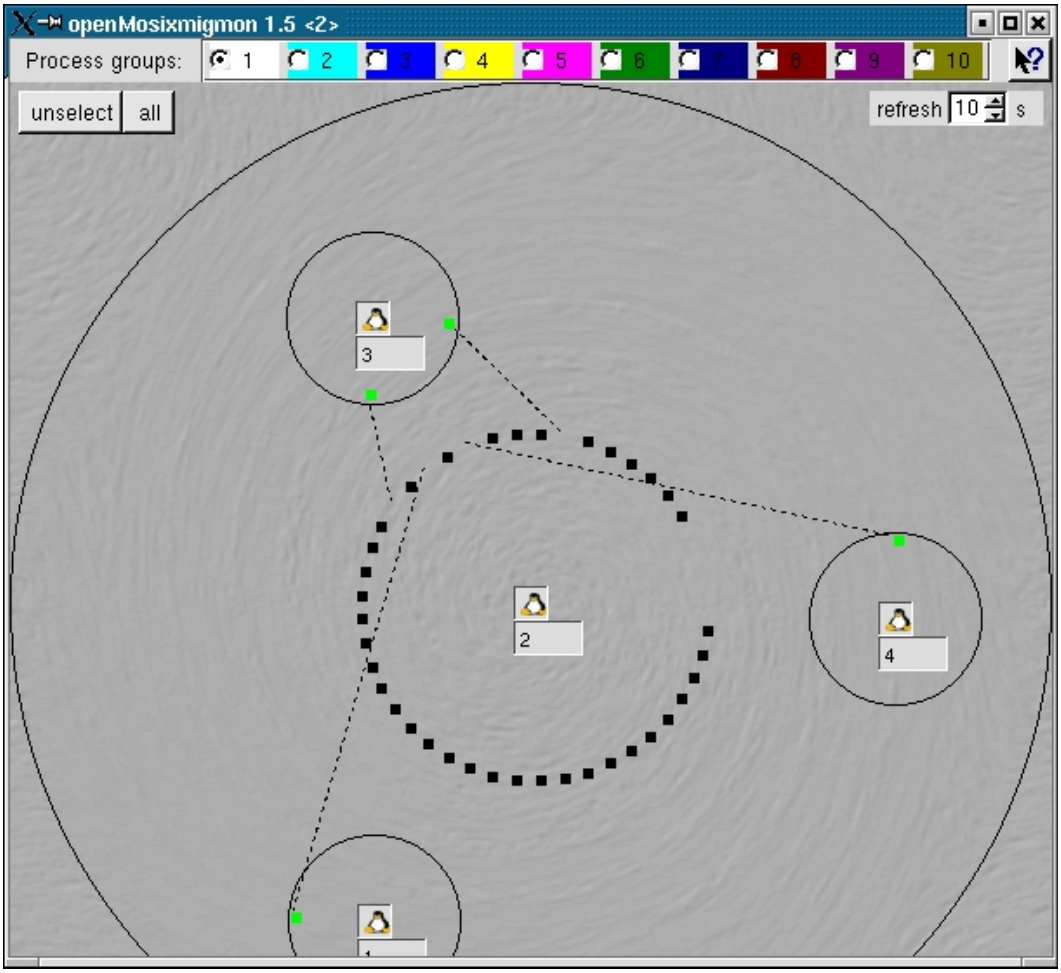

Figura C.3: openMosixmigmon

una patch ancora sperimentale al kernel già "patchato" con openmosix. E' quello che è stato fatto nella seconda fase di studio e di test del software openmosix.

C.1.2 Aggiunta di Migshm ad openMosix

Come detto precedentemente, migShm è una patch di openMosix, che a sua volta è una patch del vanilla kernel di Linux, per questo motivo per poter costruire il nuovo kernel con openMosix e migShm è stato necessario ripercorrere i passi mostrati nel paragrafo precedente, con l'aggiunta delle istruzioni specifiche per

questa nuova configurazione.

Il kernel è stato "patchato" con le istruzioni

openmosix-2.4.22.bz2 | $patch - Np1$

patch-Migshm-2.4.21.diff | $patch - Np1$

Successivamente è stato necessario abilitare alcuni parametri aggiuntivi nel kernel per rendere operative le nuove proprietà di migrazione di applicazioni a memoria condivisa:

#Shared memory migration support(Exper.)(CONFIG-SHM)(N/y/?) y

#flush() support for consistency(CONFIG-SHM-FLUSH)(N/y/?)(NEW) y

#Kernel Debug messages (CONFIG-SHM-DEBUG)(N/y/?)(NEW) y

Dopo avere ricompilato il kernel ed averlo avviato correttamente sul nodo centrale, l'installazione sulle altre macchine è stata più agevole, si è deciso infatti di non procedere alla ricompilazione anche sugli altri nodi. Semplicemente è stata copiata l'immagine del kernel su ciascuno degli altri nodi, e con essa sono state portate la cartella contenente i moduli del kernel che vengono caricati all'avvio (/lib/modules/2.4.21), e quella che contiene tutti i sorgenti e i tools di monitoraggio (/usr/src/linux-2.4.21). A questo punto è stato sufficiente modificare il lilo per rendere il kernel avviabile.

In questo modo è stato possibile avere due ambienti diversi (nonostante il software fosse lo stesso) che hanno reso possibile la realizzazione dei due tipi di test concettualmente e qualitativamente molto differenti che ho mostrato nel capitolo 4.

C.1.3 Installazione del cluster openMosix-CNR

Questa stessa tecnica è stata utilizzata anche per l'istallazione ed il settaggio del **cluster openMosix-CNR**; l'unico commento aggiuntivo che si può fare è che in questo caso l'installazione è stata resa molto più agevole per il fatto che tutti i nodi erano identici, per cui è stato possibile installare un kernel estemamente leggero che però è stato possibile esportare su tutte le macchine del cluster.

C.2 Installazione LTSP

Rispetto a quella di openMosix l'installazione di LTSP è stata decisamente meno complessa, sostanzialmente perchè non ci sono kernel da ricompilare ma solo pacchetti da installare e solo sulla macchina che funziona da server.

Per poter avviare LTSP sono necessari quattro pacchetti:

- LTSP Core package;
- Kernel package;
- Core package;
- X Fonts package.

Dopo aver terminato l'istallazione dei pacchetti, è stato necessario inizializzare i file di configurazione dei file di sistema all'interno del server.

I file di configurazione di LTSP sono contentuti nella directory /opt/ltsp/templates, perciò dopo essere entrati nella directory è stato lanciato il comando che automaticamente consente l'inizializzazione:

#./ltsp-inizialize

Dopo l'inizializzazione del sistema è necessario fornire al server le informazioni riguardanti il thin-client attraverso tre file di configurazione:

- /etc/dhcp.conf
- /etc/hosts
- /opt/ltsp/i386/etc/lts.conf

LTSP usa DHCP per fornire alle workstations tutti i dati di cui hanno bisogno per funzionare correttamente, il primo dei file appena elencati è quello tramite il quale vengono forniti alla workstation il suo indirizzo IP, l'hostname, l'IP del server ,il default gateway, la directory ed il nome del kernel da scaricare ed in fine il server e la directory da montare come filesystem di root. La figura C.4 mostra il file dhcp.conf che è stato usato per i nostri esperimenti.

L'ultimo passo consiste nel modificare opportunamente il file /opt/ltsp/i386/etc/lts.conf che ha una sintassi molto semplice ed è diviso in più sezioni. In particolare ce ne è una chiamata di **default** che contiene informazioni generali sul server, sulla risoluzione, sul tipo di display manager da utilizzare, ecc,ecc. Oltre a questa sezione nel file ne possiamo trovare una per ciascuna delle workstation del cluster contenenti informazioni sui moduli da caricare, sul tipo di scheda di rete, sulla estensione del file di swap di ciascun client, ecc, ecc.

A questo punto il settaggio è concluso ed il sistema è pronto a partire secondo la modalità di funzionamento descritta nel capitolo 3.

```
ddns-update-style         none;

default-lease-time            21600;
max-lease-time                21600;

option subnet-mask            255.255.255.0;
option broadcast-address      192.168.0.255;
option routers                192.168.0.1;
option domain-name-servers    212.216.112.222;
option domain-name            "virgilio.it";
option root-path              "192.168.0.1:/opt/ltsp/i386";

option option-128 code 128 = string;
option option-129 code 129 = text;

shared-network WORKSTATIONS {
    subnet 192.168.0.0 netmask 255.255.255.0 {
    }
}

group   {
    use-host-decl-names       on;
    option log-servers        192.168.0.1;

    host ws001 {
        hardware ethernet     00:60:97:95:7D:C1;
        fixed-address         192.168.0.2;
        filename              "/lts/vmlinuz-2.4.24-ltsp-4";
        option option-128     e4:45:74:68:00:00;
        option option-129     "NIC=3c509 ";
```

Figura C.4: dhcp.conf

Ringraziamenti

Questa tesi è dedicata a tutti quelli che mi sono stati vicini in questi anni. Alla mia famiglia, che mi ha sostenuto per tutto questo tempo. A manu, che da tre anni ha il mio cuore, che in questi mesi ha sopportato la mia agitazione e la mia lontananza, standomi sempre vicina. Ad ISF che ha ideato, proposto e sostenuto il progetto. Al prof. Baldoni che mi ha portato a questo momento così importante. A Davide che mi è sempre stato vicino, mi ha aiutato, ha condiviso le sue conoscenze con me ed è stato sempre presente in questi mesi. Alla squadra trashware con la quale faremo grandi cose ed in particolare a Emanuele al quale devo gran parte di quello che so. A Matteo e Daniele che mi hanno aiutato tanto. Ai ragazzi del BugsLab che ci hanno dato una grande ospitalità ed una grande mano. Ai ragazzi del Golem che hanno iniziato tutto. A Chiara compagna di mille avventure, a Michele che la pensa come me, che mi ha insegnato tanto e mi ha aiutato quando ero in difficoltà. A Francesco con il quale ho condiviso tante esperienze in quindici anni di vita. Ad Alessandra, Anna e Milena alle quali sono legato da un sentimento di amicizia intenso e speciale. A Paolo, Greta, Benny e Silvia con la speranza che prima o poi ci vediamo.

A tutti quelli che in questi anni hanno intrecciato le loro vite alla mia, regalandomi tante emozioni.

A me stesso;

Grazie.

Bibliografia

[1] Moshe Bar and MAASK team. Mihshm. In *Linux Congress*, 2003.

[2] Amon Barak, Avner Braverman, Ilia Gilderman, and Oren Laden. Performance of pvm with the mosix preemptive process migration scheme. Technical report, Institute of Computer Science, The Hebrew University of Jerusalem, 1997.

[3] Amon Barak and Oren La'adan. The mosix multicomputer operating system for high performance cluster computing. Technical report, Institute of Computer Science, The Hebrew University of Jerusalem, 1998.

[4] Amon Barak and Ron Lavi. The home model and competitive algorithms for load balancing in a computing cluster. Technical report, Institute of Computer Science, The Hebrew University of Jerusalem, 1998.

[5] Amon Barak and Amnon Shiloh. Scalable cluster computing with mosix for linux. Technical report, Institute of Computer Science, The Hebrew University of Jerusalem, 1998.

[6] Marc Baudoin. *Impara Latex*, 1998.

[7] BGLug. Panoramica sul kernel linux. In *Linux Day*, Bergamo 2003.

[8] Giampaolo Bonora. Digital divide. Technical report, Osseratorio Internet, 2001.

[9] Todd Burgesst. *Trashware-mini-HowTo*, 2004.

[10] Kris Buyeart. *OpenMosix howto*, 2001.

[11] Kris Buyeart. *OpenMosix-migshm howto*, 2003.

[12] Kris Buytaert. *Introducing OpenMosix*, 2004.

[13] Carlo Buzzi. I numeri ed i rischi della spazzatura informatica. *Punto Informatico*, 2003.

[14] Alberto Cammozzo. Il progetto pluto refun. In *Webbit03*, 2003.

[15] Richard Camp. *K12Ltsp-Mosix howto*, 2002.

[16] John Clay. *Reduced Cost Network Computing Through The Use Of Thin Clients and Low Cost Software*, 2003.

[17] Maritn Daniau. *Ltsp-Mosix*, 2001.

[18] Paul Dravis. Opensource software. Technical report, InfoDev, 2003.

[19] WWF-Consorzio Ecoqual'IT. L'e-waste ladri di futuro-le cause e gli effetti della mancata gestione dei rifiuti tecnologici. Technical report, WWF, 2002.

[20] Richard Ferri. *The seacrets of OpenMosix*, 2003.

[21] Daniele Frigioni. Elememnti di programmazione concorrente in ambiente gnu/linux. Technical report, Dip Ingegneria Elettrica, Universitá degli studi dell'Aquila, 2004.

[22] Gian Paolo Ghilardi. Considerations on openmosix. Technical report, Universitá degli studi di Cremona, 2002.

[23] GianPaolo Ghilardi. *FSU PThreads and OpenMosix*, 2003.

[24] Corrado Giustozzi. Obsolescenza programmata. *Byte Italia*, 1999.

[25] Golem. *Trashware HowTo*, 2001.

[26] Jesus M. Gonzales. Free software/open source: Information society opportunities for europe. Technical report, Information Society Directorate General of UE, 2001.

[27] Werner Heuser and Wade Hampton. *Linux Ecology How-to*, 1999.

[28] Alessandro Inzerilli and Igino Gagliardone. Prodigi-il progetto tunisia, 2003.

[29] ITU. World telecommunication development report. Technical report, ITU, 2003.

[30] Jemes Jensen. *Ltsp+OpenMosix mini-Howto*, 2002.

[31] Linux Journal. Discussing shared memory. *Linux Journal*, 2002.

[32] Brad Kline and Andreas Dilger. *PMVPOV HowTo*, 1994.

[33] Helmut Kopka and Patrick W. Daly. *A guide to LATEX*. Addison-Wesley, 1995.

[34] MAASK. Migshm. Master's thesis, University of Pune,India, 2002.

[35] Alberigo Massucci. Computer a caccia di ecologia. *Punto informatico*, 2003.

[36] Steve McClure and Richard Wheeler. Mosix:how linux clusters solve real world problems. Technical report, EMC2 Corporation, 2001.

[37] James McQuillan. *LTSP-Linux Terminal Server Project-v3.0*, 2002.

[38] Davide Meloni. *OpenMosix@openLabs HowTo*, 2004.

[39] Miur. Ricondizionamento di vecchi pc. Technical report, Servizio Osservatorio Tecnologico del Miur, 2001.

[40] Andrea Mosca. Lo smaltimento dei componenti elettrici ed elettronici obsoleti: aspetti normativi e tecnologici. Master's thesis, Universitá degli studi di Torino; Facoltá di economia, 2001.

[41] Pippa Norris. Digital divide. *New York:Cambridge University Press*, 2001.

[42] OCSE. Information technology outlook. Technical report, Parigi:OCSE, 2000.

[43] ONU. Information and communication technology development indices. Technical report, UNCTAD, 2003.

[44] Paolo Palmerini and Tommaso Pucci. Gnu/linux e il software libero nei paesi in via di sviluppo. In *Atti del Terzo Linuxday Italiano*, Firenze 2003.

[45] Matteo Papadopulo. Il progetto linx, 2004.

[46] Roberto Premoli. *Test Prestazionali di un Cluster di Calcolo OpenMosix*, 2001.

[47] Mathias Rochenburg. Monitoring and administration of openmosix clusters. In *Fosdem*, Bruxells 2004.

[48] Kuehr Ruediger and Eric Williams. Computers and the environment: Understanding and managing their impacts. Technical report, Kluwer Academic Publishers, October 2003.

[49] Mulyadi Santosa. Checkpointing and distributed shared memory in openmosix, Aprile 2004.

[50] George Sciadas. Monitoring the digital divide...and beyond. Technical report, Orbicom, 2003.

[51] Avi Silberschatz and Peter Galvin. *Operating System Concepts*. Wiley, 1999.

[52] Sistemisti-indipendenti.org. *Load Balancing Cluster-OpenMosix con GNU/Linux*, 2004.

[53] Richard Stallman. The gnu project. *Linux Journal*, 1998.

[54] Andrew S. Tanenbaum. *Architettura del Computer*. Jackson Libri, 1999.

[55] Barry Wellman Wenhong Chen. Charting and bridging digital divide. Technical report, NetLab, 2003.

[56] WSIS. Declaration of principles. In *world summit on the information society*, Ginevra 12 Dicembre 2003.

Elenco delle figure