POLITECNICO DI MILANO
SCUOLA DI INGEGNERIA DELL'INFORMAZIONE
Corso di Laurea Specialistica in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

Sistema a supporto del design-time per l'efficienza energetica di processi di business basati su servizi

Relatore: Ing. Pierluigi PLEBANI

Tesi di Laurea di:
Tommaso CODELLA
Matricola 739376

Anno Accademico 2010-2011

Ringraziamenti

All'ingegner **Pierluigi Plebani**, per tutta la disponibilità e il supporto dimostrati durante questo lavoro di tesi.

A mia **madre**, per avermi dato la possibilità di seguire la strada che più mi rendeva felice, per l'enorme pazienza dimostrata in tutti questi anni e per tutto l'incoraggiamento e il sostegno che mi ha sempre dato.

Ad **Agnese**, per essere stata al mio fianco durante tutti gli anni dell'università e avermi incoraggiato nei momenti più difficili.

A **nonna Vincenza**, per avermi sempre spronato a proseguire gli studi e per tutto quello che ha fatto per me.

A **zia Tamara** e **Chris**, per avermi accolto nella loro casa e al piccolo **Luca** per avermi distratto nei lunghi pomeriggi di studio.

A **zio Nico**, per tutto l'aiuto che mi ha dato e che non potrò mai dimenticare.

A mio **padre** e a tutta la mia **famiglia**, per la fiducia dimostratami durante questo cammino.

A **Julio**, per l'incoraggiamento di questi anni.

A **Davide**, per essersi sempre dimostrato un buon amico.

Alla mia **staff**, presente e passata, per tutto il sostegno che mi hanno dato.

A tutti coloro che hanno condiviso con me il cammino universitario.

Abstract

Valutare l'impatto energetico dei processi di business, o più in generale delle applicazioni, è un compito non facile ma molto utile in un'ottica di ottimizzazione energetica.
La gran parte dei modelli fino ad ora proposti in letteratura si preoccupano principalmente del problema della valutazione dell'energia dissipata a runtime utilizzando informazioni che però non sono disponibili a design-time. In questa fase è possibile solamente stimare il tempo di esecuzione del processo ed eventualmente quali risorse utilizzerà e per quanto tempo verranno usate.

Scopo di questo lavoro di tesi è mostrare come, partendo da quanto proposto in letteratura, si è via via andato a creare un modello capace di calcolare l'energia richiesta per l'esecuzione di un processo basandosi solamente su dati del processo plausibilmente disponibili a design-time (o al massimo stimabili) e sulle caratteristiche tecniche delle macchine a disposizione per l'esecuzione.
A supporto del modello proposto è stata inoltre sviluppata una applicazione che funge da supporto al designer grazie alla quale è possibile valutare l'impatto energetico dell'esecuzione di un processo in un datacenter.
Un altro aspetto preso in considerazione in questa tesi è la possibilità di utilizzo delle soluzioni proposte in qualsiasi tipologia di datacenter: dai più moderni datacenter virtualizzati che introducono diversi livelli di ottimizzazione energetica ai più vecchi datacenter per i quali migrare a soluzioni energeticamente più efficienti (quali la virtualizzazione) potrebbe rappresentare un costo troppo elevato.

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Valutare l'impatto energetico dei processi di business, o più in generale delle applicazioni, è un compito non facile ma molto utile in un'ottica di ottimizzazione energetica.
La gran parte dei modelli fino ad ora proposti in letteratura si preoccupano principalmente del problema della valutazione dell'energia dissipata a runtime utilizzando informazioni che però non sono disponibili al designer, il quale può solamente stimare il tempo di esecuzione del processo ed eventualmente quali risorse utilizzerà e per quanto tempo verranno usate. Per questo motivo è importante strutturare un modello dei consumi energetici del server che sia sufficientemente dettagliato da permettere di avere una stima il più possibile accurata ma che usi solamente informazioni che non richiedano l'esecuzione del processo.
Scopo di questo lavoro di tesi è mostrare come, partendo da quanto proposto in letteratura, si è via via andato a creare un modello capace di calcolare l'energia richiesta per l'esecuzione di un processo basandosi solamente su dati del processo plausibilmente disponibili a design-time (o al massimo stimabili) e sulle caratteristiche tecniche delle macchine a disposizione per l'esecuzione. Quest'ultime vengono ricavate utilizzando esclusivamente i dati presenti sui datasheet delle varie componenti hardware presenti nel sistema in esame: non è quindi richiesto alcun tipo di sistema di misurazione installato sulle macchine.
A supporto del modello proposto è stata inoltre sviluppata una applicazione che funge da supporto al designer grazie alla quale è possibile valutare l'impatto energetico dell'esecuzione di un processo in un datacenter.
Dovendo valutare l'impatto energetico a design-time non è per forza necessario che il valore calcolato sia coincidente con il valore reale ma basta che sia una buona approssimazione che possa servire al designer per valutare in che modo (ed eventualmente con che tempi) eseguire i vari processi. Per esempio, non avendo particolari vincoli sulla durata temporale di un processo, talvolta potrebbe essere più efficiente dal punto di vista energetico utilizzare un server meno performante, quindi con tempi di esecuzione maggiori, ma più economico dal punto di vista del consumo energetico rispetto a un altro che riduce il tempo di esecuzione ma impatta maggiormente sulla potenza dissipata. Ovviamente a un livello così alto di valutazione non è necessario

avere il valore preciso del consumo energetico del processo ma è sufficiente offrire una stima ed, eventualmente, delle informazioni sul possibile errore introdotto così da permettere al designer di fare le opportune valutazioni.
Bisogna anche tenere presente che volendo ottenere poi un'ulteriore efficienza energetica è anche possibile combinare le strategie proposte in questa tesi a opportuni modelli di valutazione dell'efficienza energetica a runtime per ottenere una allocazione dei processi ancora migliore riducendo ulteriormente i consumi energetici.
Un altro aspetto preso in considerazione in questa tesi è la possibilità di utilizzo delle soluzioni proposte in qualsiasi tipologia di datacenter: dai più moderni datacenter virtualizzati che introducono diversi livelli di ottimizzazione energetica ai più vecchi datacenter per i quali migrare a soluzioni energeticamente più efficienti (quali la virtualizzazione) potrebbe rappresentare un costo troppo elevato.
Il modello qui proposto quindi può aiutare anche datacenter più vecchi e non efficienti a ridurre il loro impatto energetico semplicemente valutando come dislocare i processi in base alla stima dell'assorbimento energetico fatta a design-time.
Dover tenere in conto di tutti questi aspetti ha fatto sì che si sia approfondito solamente il caso in cui si ha un solo processo in esecuzione senza preoccuparsi di eventuali altri processi.
Nei prossimi capitoli verrà illustrato il lavoro svolto per ottenere un modello che potesse rispondere al meglio a tutte queste esigenze.
Nel capitolo 1 verrà anzitutto fatta un'analisi dello stato dell'arte illustrando gli articoli scientifici, o più in generale,le soluzioni proposte per risolvere problemi simili a quello qui affrontato. Si cercherà di mettere in luce come questi lavori siano serviti per realizzare il modello che verrà proposto nel capitolo 2. L'esposizione del modello inizierà con una analisi volta a identificare le componenti con cui modellizzare un generico server sia esso fisico o virtuale.
Nel capitolo 2 verrà inoltre proposto un semplice modello di allocazione dei servizi di un processo di business che utilizza il modello energetico sviluppato in questa tesi.
Nel capitolo 3 verranno esposte le metodologie adottate per valutare la validità del modello proposto illustrando i test effettuati, i server utilizzati e valutando i risultati ottenuti.
Nel capitolo 4 verrà presentata un'applicazione web per il designer che utilizza il modello proposto in questa tesi e realizzata per fornire un prototipo di applicazione funzionante che permetta la valutazione del consumo energetico a design-time dei processi di business.
Infine nell'ultimo capitolo verrà fatta un'analisi critica del lavoro svolto cercando di identificare i punti della tesi che potrebbero essere approfonditi ulteriormente servendo come partenza per futuri lavori.

Capitolo 1

Stato dell'arte

GAMES è un progetto di ricerca europeo che ha come obiettivo quello di realizzare una serie di linee guida, metodologie e strumenti per la costruzione e la gestione di data center efficienti dal punto di vista energetico.
La grande diffusione di internet e dei servizi informatici negli ultimi dieci anni ha fatto sì che i data center si siano moltiplicati e ingranditi notevolmente.
L'EPA (U.S. Environmental Protection Agency), l'Agenzia Statunitense per la Protezione dell'Ambiente, a fine del 2006 ha ricevuto il mandato dal Congresso degli Stati Uniti d'America (tramite la Public Law 109-431) di preparare un report sui consumi energetici dei Data Center in territorio americano.
Il report in questione ha denotato come nel 2006 fosse stato stimato un consumo energetico di circa 61 miliardi di kWh pari all'1.5% del consumo totale di energia elettrica negli USA. L'impatto economico è stato di 4.5 miliardi di dollari.
E' stato valutato che nel 2006 l'energia consumata sia raddoppiata rispetto al 2000, è stato inoltre calcolato che il consumo si sarebbe duplicato ulteriormente nei successivi cinque anni per raggiungere nel 2011 quota 100 miliardi di kWh per un costo totale di 7.4 miliardi di dollari.
E' interessante notare come il picco di carico di potenza richiesta sulla rete fosse nel 2006 di circa 7GW cioè l'equivalente di 15 centrali elettriche e le stime indicavano per il 2011 un picco di 12GW equivalenti a 10 ulteriori centrali.
Le previsioni qui riportate si riferiscono al caso peggiore che gli studiosi avevano stimato ovvero quello in cui il trend si fosse mantenuto costante e non fossero state adottate misure per l'efficienza energetica.
Questi dati, sebbene valutati cinque anni fa e relativi ai soli data center sul territorio degli Stati Uniti, fanno intuire l'importanza che ha il problema dell'efficienza energetica dei data center.
Un recente studio [6], pubblicato in agosto del 2011, analizza i dati stimati dall'analisi dell'EPA e li confronta con ciò che si è realmente verificato nel

2010.
La stima dell'EPA era di un incremento rispetto dell'energia consumata pari all'1.7% per il consumo mondiale e del 2.8% per il consumo degli Stati Uniti. I dati del 2010 invece sono risultati migliori del previsto: l'incremento è stato stimato tra l' 1.1 e l'1.5% per il consumo mondiale mentre per gli USA lo stesso valore è stato stimato tra l'1.7% e il 2.2%.
Il fatto che il trend di crescita dei consumi energetici si sia attestato a un livello inferiore del previsto è stato causato dalla crisi economica inziata nel 2008 ma è anche dovuto agli accorgimenti energetici adottati dalle aziende in questi ultimi anni.
Oltre ai dati economici un'altro aspetto importante sono le emissioni di CO_2 che l'utilizzo di queste strutture provoca, infatti le emissioni mondiali dei data center vengono stimate come equivalenti alla metà di quelle prodotte dalle compagnie aeree e maggiori di quelle prodotte dall'Argentina e dai Paesi Bassi insieme. Le cause di questi consumi energetici non sono però da riferire ai soli server in sè ma a tutte le infrastrutture ad essi collegate quali ad esempio gli impianti di raffreddamento.
In questo scenario il progetto GAMES vuole proporre un'insieme di strumenti che spaziano dalle metodologie fino a tools software atti a realizzare dei data center efficienti sotto ogni punto di vista. L'innovazione di questo progetto consiste nel fare in modo che il risparmio energetico sia considerato come un vero e proprio obiettivo primario mantenendo buoni i rapporti tra l'efficienza energetica, la qualità del servizio e le prestazioni.
All'interno di tutto ciò questa tesi si pone come obiettivo quello di identificare un modello per analizzare a design-time il consumo di energia dei singoli servizi che verranno posti in esecuzione nel data center. Questo modello verrà poi implementato in un tool reso disponibile ai designer dei processi e che servirà loro per gestire l'allocazione ottima dei servizi sui server del data center al fine di ottenere, anche sotto questo aspetto, una maggiore efficienza.

In questo primo capitolo verrà spiegato, anzitutto, il concetto di efficienza energetica a design-time e quali caratteristiche sono richieste da un applicativo che sia di supporto a questa fase.
Verrano successivamente esposti i principali tool e modelli presenti in letteratura, per ognuno verrà prima fatta una rapida presentazione alla quale seguirà un'analisi volta a portare alla luce le problematiche a cui porterebbe l'uso di tale modello in fase di design .

1.1 Efficienza energetica a design-time

Valutare l'efficienza energetica a design-time è un compito che richiede di avere particolare attenzione sia nell'utilizzare un buon modello che nello sti-

mare i parametri caratteristici dei processi necessari al modello.
La valutazione dell'energia dissipata può essere una discriminante per allocare successivamente i servizi e ottenere, rispettando i vincoli imposti, una disposizione dei servizi sui server che permetta un risparmio energetico.
Il problema principale è quello di realizzare un modello delle componenti hardware che permetta di valutare la potenza dissipata in funzione di parametri noti a priori e forniti dal costrutture. Questo è un punto fondamentale per ottenere uno strumento che sia di supporto al designer il quale non deve essere costretto ad eseguire i servizi per valutare l'energia dissipata.
Oltrettutto basare un modello sui soli parametri noti dai datasheet delle componenti fa sì che non sia necessario dover, per ciascun server, installare sensori di rilevazione dei consumi energetici per le singole componenti al fine di profilare ogni server nel datacenter. Tra l'altro ipotizzare di utilizzare un sistema di sensori che profili ciascun server in un datacenter per ottenere dati sui consumi non forniti dai produttori è una soluzione altamente costosa che potrebbe non essere praticabile.
Da quanto appena esposto è facile dedurre che l'elaborazione di un modello che riesca a essere di supporto al design-time è un problema delicato che richiede soprattuto di saper identificare i dati necessari al suo funzionamento oltre che a capire bene come funzionano le componenti hardware interessate così da poterle modellare al meglio.

1.2 Modelli e strumenti proposti in letteratura

In letteratura si possono trovare diverse proposte di modelli progettati appositamente per calcolare il consumo energetico di un server.
Il principale problema dei modelli proposti è che valutano l'efficienza energetica a runtime mentre il modello che verrà presentato nel prossimo capitolo è stato sviluppato per valutare l'impatto energetico di un processo di business a design-time quindi con informazioni talvolta limitate e senza dubbio diverse da quelle che si possono avere durante l'esecuzione di tale processo.
Anzitutto verrà presentato un modello proposto per valutare il consumo energetico della CPU al fine di realizzare degli scheduler efficienti dal punto di vista energetico. Sucessivamente saranno illustrati due tool proposti in letteratura e i relativi modelli sui quali questi strumenti si basano.
I modelli e gli strumenti che verranno esposti nelle prossime sezione sono particolarmente interessanti per questa tesi in quanto utilizzano dei modelli simili a quello che verrà proposto successivamente ma che in alcuni punti risultano non utilizzabili a design-time.

1.2.1 Modello per scheduler Energy Efficient

Nell'articolo [2] viene proposto un modello per descrivere il consumo energetico della CPU al fine di realizzare uno scheduler efficiente da questo punto di vista.
Il modello proposto si pone l'obiettivo di predirre il consumo energetico di un processo in base alla tipologia e al numero di cicli della CPU utilizzati.
Alla base vi è il fatto che la potenza dissipata da un computer è proporzionale alla potenza dei singoli componenti e quindi può essere valutata mediante un semplice modello additivo qui riportato.

$$\begin{aligned} &P(System) \propto P(CPU) + P(Memory) + P(Fans) + P(HDDs) + \\ &+ P(Northbridge) + P(Southbridge) + P(Graphics) + \\ &+ P(OtherComponents) \end{aligned} \tag{1.1}$$

Dovendo valutare l'energia consumata in un datacenter il modello viene semplificato ignorando ciò che viene usato di meno e raggruppando tale valore nel termine P(Bias) come si può notare nella relazione (1.2)

$$P(system) \propto P(CPU) + P(Memory) + P(Bias) \tag{1.2}$$

$$\begin{aligned} &P(Bias) = P(Fans) + P(HDDs) + P(Northbridge) + P(Southbridge) \\ &+ P(Graphics) + P(OtherComponents) \end{aligned} \tag{1.3}$$

Come veniva spiegato all'inizio l'idea alla base è quella di modellare la potenza della cpu come proporzionale alla tipologia e al numero dei cicli utilizzati da un task ottenendo quindi:

$$P(Task_i) \propto Cycle(FPU) + Cycle(INT) + Cycles(Cache) \tag{1.4}$$

$$P(system) \propto \sum_{i=1}^{N} P(Task_i) + P(Bias) \tag{1.5}$$

Nella relazione (1.4) $Cycle(FPU)$ rappresenta il numero di cicli floating point mentre $Cycle(INT)$ il numero di cicli interi.
Il problema di questo modello è che necessita di conoscere la tipologia di cicli e il numero per ciascun processo dato che a design-time è difficile ottenere.
Senza dubbio però questo modello può essere adoperato per costruire uno scheduler efficiente che, abbinato a una valutazione a design-time del consumo energetico, può aiutare a ridurre la potenza necessaria a un processo.

1.2.2 pTop

Il tool pTop presentato in [5] offre una soluzione software-based per la misurazione del consumo energetico.
Questo strumento nasce dalla volontà di creare un servizio per la profilazione del consumo energetico a livello di processo.
L'idea alla base di questo tool è quella di poter calcolare il consumo energetico di un qualsiasi processo in esecuzione su un computer.
pTop quindi agisce come servizio del sistema operativo ed avendo in tempo reale le informazioni sulle risorse impiegate da ogni applicazione e possedendo le informazioni sull'hardware del computer su cui è in esecuzione è in grado di valutare quanta energia viene dissipata dal processo monitorato.
Il problema di questo tool è che necessita delle informazioni in tempo reale dei consumi delle risorse del sistema cosa non possibile quando si cerca di conoscere il conumo energetico di una applicazione prima che questa venga eseguita.
Il tool qui presentato, come nel caso esposto nella sezione precedente, utilizza un modello additivo non considerando però la componente Bias precedentemente identificata come la potenza dei componenti secondari.

$$\begin{aligned} E_{SYS} &= E_{CPU} + E_{HDDs} + E_{NETs} = \\ &= \sum_j P_j * t_j + \sum_k n_k * E_k + \\ &+ \sum_i t_{sendi} * P_{send} + t_{recvi} * P_{recv} + \\ &+ \sum_l t_{writel} * P_{write} + t_{readl} * P_{read} + \end{aligned} \tag{1.6}$$

Nell'equazione (1.6) P_j e t_j rappresentano la potenza dissipata per una certa frequenza j e il tempo durante il quale la CPU funziona a quella frequenza.
n_k e E_k rappresentano il numero di transazioni k tra gli stati della CPU occorsi e la rispettiva energia consumata.
Per valutare la potenza dissipata dalle componenti pTop utilizza il valore della TDP fornita dal costruttore dell'hardware. Sebbene la TDP rappresenti la massima potenza termica che il sistema di raffreddamento deve essere in grado di dissipare, gli autori di [5] sostengono che sia una buona approssimazione.
Si nota subito che alcuni parametri richiesti dal modello (ad esempio n_k o l'esatta frequenza di a cui viene usata la CPU) non sono conosciuti a design-time mentre sono rintracciabili facilmente a runtime. Inoltre questo modello

non considera i consumi energetici delle altre componenti che sebbene secondarie possono portare a un impatto energetico non trascurabile.

1.2.3 Joulemeter

Joulemeter proposto in [4] è stato sviluppato per valutare il consumo energetico delle macchine virtuali installate nei data center.
Questo tool utilizza un modello matematico che valuta l'impatto energetico della macchina virtuale basandosi sulle risorse utilizzate a runtime.
Il modello energetico utilizzato da Joulemeter è rappresentato dalla seguente equazione:

$$\begin{aligned} E_{sys} &= E_{cpu} + E_{mem} + E_{disk} + E_{static} = \\ &= \alpha_{cpu} u_{cpu}(p) + \gamma_{cpu} + \alpha_{mem} N_{LLCM} \\ &+ \gamma_{mem} + \alpha_{io} b_{io} + \gamma_{disk} + E_{static} \end{aligned} \tag{1.7}$$

La eq.(1.7) presenta anzitutto un limite per l'utilizzo a design-time che è dato dal parametro N_{LLCM} rappresentante il numero di LLC miss durante l'utilizzo della virtual machine. Questo parametro è impossibile da valutare se non durante l'esecuzione perciò creare un modello che utilizzi questo parametro rende tale modello non utilizzabile in fase di design dei processi di business.
Il modello utilizzato da Joulmeter prevede inoltre di utilizzare una serie di parametri (α_{cpu}, α_{io}, γ_{disk}, γ_{mem}) che devono essere valutati in sito sul server su cui verranno installate le macchine virtuali delle quali si farà la valutazione energetica.
Joulemseter risulta essere un ottimo strumento per valutare il consumo energetico di macchien virtuali e offre un valido modello che però è utilizzabile solo a runtime e soltanto per server di cui sono stati valutati i parametri caratteristici richiesti dal modello.

1.2.4 Altri modelli e metodologie

In letteratura si possono trovare altri modelli sviluppati appositamente per valutare il consumo energetico delle applicazioni. Purtroppo questi richiedono necessariamente l'esecuione dei processi o perlomeno informazioni che sono deducibili unicamente dopo aver eseguito almeno una volta il software. Un esmpio è il modello proposto da Wendt, Grumer et al. [7] pensato in particolar modo per i sistemi embedded il quale valuta l'energia consumata dalla CPU come la somma dell'energia spesa durante ogni ciclo di esecuzione. E' chiaro che un modello della CPU di questo genere richiede che il software sia eseguito almeno una volta per valutare quanti e quali cicli del microprocessore utilizza.

Kansal e Zhao [1] propongono invece un metodo per la valutazione dell'impatto energetico a design-time però utilizzano un approccio diverso da quello adottato in questa tesi.
Nel loro articolo gli autori illustrano come sia possibile valutare a design-time l'impatto energetico identificando le risorse utilizzate e i pattern adottati nello sviluppo dell'applicazione in esame.
Il designer ha a disposizione le informazioni su quanto una scelta di design rispetto a un'altra può impattare a livello energetico basandosi su misurazioni fatte da uno specifico tool. A questo punto il deisgner può modificare il codice sorgente utilizzando un pattern più efficiente o una libreria dal ridotto consumo energetico.
Come si può capire però i problemi sono due: il primo è che non sempre il designer può essere disposto a modifica il codice sorgente e in secondo luogo ci si basa sempre su dati misurati mentre ciò che si vorrebbe ottenere è un modello che non richieda alcun tipo di esecuzione o misurazione per stimare l'imaptto energetico di una applicazione.

1.3 Conclusioni

Gli articoli in letteratura che affrontano il problema dell'efficienza energetica sono molteplici e le tematiche affrontate sono svariate. In questo capitolo sono stati presentati solo alcuni dei lavori, quelli che più di altri sono risultati inerenti a questa tesi e che rappresentano un buon riassunto dei molti articoli individuati.
Gli spunti tratti dai lavori qui riportati hanno aiutato lo sviluppo del modello di consumo energetico del server proposto nel prossimo capitolo. In particolar modo si sono utilizzate le informazioni reperite in letteratura per capire come progettare il modello.

Capitolo 2

Modello Energetico

In questo capitolo si illustrerà il modello matematico sviluppato per valutare il consumo energetico di un processo di business basato su servizi.
Nelle sezioni seguenti verrà riproposta la metodologia seguita per definire il modello: si inizierà individuando quali siano le componenti principali nel bilancio energetico di un server proseguendo poi con l'esposizione del modello di consumo energetico proposto per valutare l'assorbimento energetico del server in esame. Sarà successivamente spiegato come questo modello possa essere utilizzato per calcolare l'impatto energetico dell'esecuzione di un processo su un determinato computer sia esso fisico o virtuale.
Successivamente si prenderà in esame il modello di allocazione dei servizi facenti parte del processo di business e si illustrerà come questo modello possa aiutare ad ottere l'efficenza energetica dell'esecuzione del processo di business.
Il capitolo terminerà con l'analisi delle annotazioni necessarie per un corretto uso del modello.

2.1 Identificazione delle componenti

Il primo passo per la creazione di un modello del consumo energetico, e ancor prima della variazione di potenza assorbita, di un processo è quello di modellare la macchina su cui l'attività dovrà essere eseguita.
Prima di tutto è stato necessario individuare le componenti che concorrono principalmente nel bilancio energetico del server.
L'individuazione delle componenti principali è stata possibile partendo dalle considerazioni fatte in [5], [4] e [1]. In questi articoli gli autori hanno proposto diversi modelli per valutare il consumo energetico di una applicazione e per farlo hanno modellato il calcolatore rappresentandolo come l'insieme di alcune componenti hardware trascurandone altre dal ridotto impatto energetico o che non comportano variazioni nella potenza assorbita in relazione

al differente utilizzo del calcolatore.
Dovendo realizzare un modello che potesse dare un supporto a design-time le componenti individuate negli articoli sopra citati sono state confrontate con i dati reperibili sui datasheet forniti dai prodotturi di componenti hardware per verificare se vi erano dati a sufficienza per modellare correttamente tali componenti.
E' molto importante tenere presente che il mdodello deve essere facilmente utilizzabile a design-time e utilizzare solo informazioni disponibili in tale fase.
Utilizzare le informazioni presenti nei datasheet permette di evitare di dover profilare ciascun server che si vuole modellare. Inoltre questi dati possono essere inseriti in una knowledge base per essere accessibili a tutti gli sviluppatori e facilmente utilizzabili.
Un generico server può essere dunque analizzato come l'insieme di:

- Microprocessore
- Memoria centrale (RAM)
- Dischi rigidi e/o unità allo stato solido
- Adattatore RAID
- Dispositivi di rete

Si noti che alcune componenti sono comuni in tutti i modelli proposti in letteratura mentre altre, quali ad esempio la scheda di rete o l'adattatore RAID, non sempre vengono considerate.
Nello scenario che viene qui preso in considerazione, cioè la valutazione del consumo energetico di un processo di business basato su servizi ed in esecuzione in un cluster all'interno di un IT Service Center, una componente particolarmente importante è la scheda di rete.
Questa componente non sempre risulta modellata e molte volte l'impatto energetico di questo dispositivo viene trascurato (spesso perché si valuta l'impatto energetico di una applicazione in esecuzione su un computer senza valutarne l'eventuale trasmissione in rete di informazioni) o viene stimato in valore medio e sommato all'assorbimento di potenza di altre componenti minori.
Dovendo lavorare su processi di business basati su servizi ed essendo plausibile che i servizi che compongono il processo possano essere disposti su diverse macchine, la comunicazione di rete gioca un ruolo fondamentale nell'esecuzione del compito assegnato al processo quindi impatterà più o meno fortemente sul bilancio energetico. Si pensi inoltre all'architettura tipica di un datacenter in cui vi sono diverse unità di storage di rete alle quali afferiscono diversi rack: il salvataggio di dati e la successiva lettura oltre ad utilizzare i dischi in cui i dati sono memorizzati provoca un intenso utilizzo

dell'adattatore di rete provocando così un consumo energetico non trascurabile.

2.2 Modello energetico del server

Identificate le componenti principali di un server è necessario analizzarle singolarmente così da identificare un modello utilizzabile a design-time per ciascuna componente.
I due punti chiave per identificare correttamente i modelli delle singole componenti sono i seguenti:

- l'obiettivo è di poter calcolare il consumo energetico stimato del processo basandosi solo su ciò che si può sapere a design-time e non dovendo obbligatoriamente eseguire il processo sul server;
- l'impatto energetico va valutato modellando le componenti hardware basandosi solamente sui dati reperibili dai datasheet dei produttori senza quindi costringere il designer a rilevare dati sul consumo di potenza dal server.

Tenendo presenti questi due punti si illustreranno ora i modelli delle componenti principali di un server.

2.2.1 Modello del microprocessore

Il microprocessore è probabilmente una delle componenti interne di un computer o di un server che consuma più energia delle altre.
I diversi articoli citati in precedenza ([5], [4] e [1]) si avvalgono di modelli molto simili a quello proposto servendosi però di informazioni disponibili a runtime o addirittura eseguendo il processo da monitorare tenendo traccia delle risorse utilizzate e delle relative quantità di utilizzo.
Anche in [2] viene proposto un modello molto accurato per descrivere l'assorbimento di potenza della CPU ma anche in questo caso è necessario avere informazioni sull'esecuzione dell'applicazione in esame.
Il modello per la valutazione del consumo energetico del microprocessore è stato concretizzato mediante l'equazione:

$$E_{cpu} = \sum_{m \in M} P_{cpu@m} * t_{exec@m} \tag{2.1}$$

L'equazione (2.1) esprime come l'energia elettrica consumata dal microprocessore sia determinata dalla potenza dissipata dal processore a seconda della modalità di funzionamento ($P_{cpu@m}$) e da quanto tempo il processore viene utilizzato a quella modalità ($t_{exec@m}$).

Nella eq. (2.1) M rappresenta l'insieme di tutte le modalità m di funzionamento della CPU.
L'idea alla base del modello è quella di poter vautare l'impatto energetico che ha l'esecuzione di un servizio afferente a un processo in base a come utilizza la CPU.
Per qualsiasi processore sono solitamente definite almeno due modalità di funzionamento: idle e attiva. A sua volta la modalità attiva può essere suddivisa in più modalità per ognuna delle quali è definita la potenza dissipata. Spesso le modalità active differiscono l'una dall'altra per le caratteristiche attive della CPU, per la frequenza del clock e per la tensione in ingresso richiesta: vi sono dunque modalità a risparmio energetico che limitano le funzionalità del processore e modalità ad altre prestazioni che non attuano politiche di risparmio energetico.
Non sempre a design-time è possibile conoscere l'esatto tempo di esecuzione del processo in esame a ciascuna modalità di funzionamento della CPU, si può invece stimare il differente impatto energetico del processo variando la modalità di funzionamento della CPU ed eventualmente adattando i tempi di esecuzione.
Grazie a questo modello è possibile dunque valutare anche come cambia l'esecuzione di più servizi concorrenti sulla stessa macchina se uno dei servizi richiede al microprocessore di variare la modalità di funzionamento.
Al contrario di altri modelli, ad esempio quello proposto in [5] per la CPU in questo modello non si considera la variazione di potenza dovuta al passaggio tra una modalità di funzionamento e l'altra in quanto non è quantificabile a priori sia il numero di variazioni di modalita sia l'effettivo avvenimento di tali variazioni.

2.2.2 Modello della memoria

Il modello della memoria centrale segue un'equazione più semplice rispetto a quello del microprocessore e risulta anche notevolmente semplificato rispetto ai modelli solitamente presentati in letteratura.
L'energia richiesta dalla memoria centrale viene caratterizzata dalla seguente relazione:

$$E_{mem} = \sum_{i=1}^{n} P_{memi} * t_{exec} \tag{2.2}$$

L'equazione (2.2) modella l'energia consumata dalla memoria RAM come la potenza dissipata da ogni singolo banco di memoria (P_{memi}) valutata su tutto l'intervallo di tempo di esecuzione (t_{exec}) del servizio in esecuzione sul server.
Calcolare il corretto impatto energetico che può avere un servizio durante la sua esecuzione non è affatto semplice in quanto la memoria centrale non è

un'entità gestibile dalle applicazioni ma viene amministrata dal sistema operativo con strategie non facilmente predicibili e dipendenti da troppi fattori per essere determinate a livello di design.
Il problema principale risiede nell'utilizzo dello swap: il sistema operativo potrebbe decidere immediatamente prima dell'esecuzione del processo di cui si vuole valutare l'impatto energetico di liberare parte della memoria per poter allocare uno o più servizi che stanno per essere eseguiti.
Per liberare spazio deve quindi copiare parte di RAM sulla memoria di massa generando quindi un consumo energetico difficile da prevedere a priori.
Altro aspetto da non trascurare è l'accuratezza dei dati presenti sui datasheet resi pubblici dai produttori di memorie: prendendo ad esempio la banca dati [14] che contiene i dati di tutti banchi di memoria prodotti da Kingston Technology si può subito notare che solo per alcuni modelli è riportato l'assorbimento di potenza e che per praticamente nessun prodotto è disponibile la differenziazione tra potenza assorbita in scrittura, lettura e idle ma è quasi sempre disponibile solo un valore medio o massimo di potenza assorbita. Questo fatto impedisce dunque di poter modellare la memoria RAM in modo più accurato.
Va ricordato inoltre che la memoria viene utilizzata non solo dal sistma operativo ma anche dalla CPU quando si verifica un LLC miss quindi, come proposto da [4], bisognerebbe conoscere anche il numero di LLC misses che occorrono durante l'esecuzione.
Considerando tutti questi fattori appena esposti il modello per la descrizione dell'energia richiesta dalla RAM risulterebbe:

$$\begin{aligned} E_{mem} = {} & P_{memwrite} * t_{memread} + P_{memread} * t_{memread} \\ & + P_{memidle} * t_{memidle} + P_{diskwrite} * t_{diskwrite} \\ & + P_{diskread} * t_{diskread} + P_{memwrite} * t_{memswapwrite} \\ & + P_{memread} * t_{memswapread} + N_{LLCm} * (P_{memread} * t_{memLLCread}) \end{aligned} \tag{2.3}$$

Nell'equazione (2.3) i termini: $P_{memwrite}$, $P_{memread}$ e $P_{memidle}$ rappresentano rispettivamente la potenza dissipata dalla memoria in scrittura, lettura e idle; ognuno di questi paramentri è legato al rispettivo tempo in cui l'attività usa la memoria nella modalità in esame.
I termini: $P_{diskwrite}$ e $P_{diskread}$ si riferiscono all'eventuale utilizzo del disco per la liberare la RAM. Si riporta l'energia utilizzata sia in scrittura (quando dalla RAM i dati vengono trasferiti sul disco) che in lettura (i dati precedentemente scaricati sul disco vengono caricati nuovamente in RAM) poiché anche quest'ultima operazione deve essere eseguita a causa del servizio di cui si stava valutando l'impatto energetico il quale richiedendo di essere eseguito ha fatto si che il sitema operativo trasferisse parte del contenuto della memoria centrale sulla memoria di massa. Per lo stesso motivo è stato inserito l'impatto energetico dato dalla lettura, e successiva scrittura, in fase di ripristino dei dati presenti nella memoria RAM; per poter valutare questo

fattore è necessaario conoscere per quanto tempo viene utilizzata la memoria RAM in lettura ($t_{memswapread}$) per spostare i dati sul disco rigido e in scrittura ($t_{memswapwrite}$) per riportare i dati dal disco in memoria. Nel modello caratterizzato dall'equazione (2.3), $N_{LLCm} * (P_{memread} * t_{memLLCread})$ rappresenta l'energia dissipata dagli eventuali LLC misses.
Sebbene l'equazione (2.3) rappresenti una modellazione più accurata della memoria RAM analizzando le sue componenti si deduce come questa possa essere utilizzata per valutare l'energia dissipata a runtime e non a design-time.

2.2.3 Modello della memoria di massa

L'energia richiesta memoria di massa, sia essa rappresentata da un disco rigido, da una unità allo stato solido o da unità di backup su nastro, può essere modellata sulla base della richiesta della risorsa dal processo secondo l'equazione proposta:

$$E_{disk} = P_{write} * t_{write} + P_{read} * t_{read} + P_{wait} * t_{wait} \tag{2.4}$$

La eq. (2.4) rappresenta dunque l'energia consumata da una unità di memoria di massa (un disco) ed è caratterizzata da tre componenti rappresentanti rispettivamente: la potenza dissipata per la scrittura dei dati (P_{write}), la potenza dissipata per la lettura dei dati (P_{read}) e la potenza dissipata dall'inutilizzo dell'unità (P_{wait}) ciascuna valutata per il relativo tempo di utilizzo.
Dopo una attenta analisi dei datasheet dei dischi rigidi e delle unità allo stato solido è emerso che la potenza in lettura e la potenza in scrittura solitamente sono uguali, nel modello i due dati sono rimasti separati sia per caratterizzare meglio il comportamento del disco e il relativo utilizzo da parte del processo che per includere eventuali dischi che prevedono ponteze in lettura e scrittura diverse.
Alcuni modelli di unità di memoria di massa permettono di suddividere ulteriormente l'ultimo fattore della eq. (2.4) ($P_{wait} * t_{wait}$) può essere rappresentato dalla seguente relazione:

$$\begin{aligned} E_{wait} &= P_{wait} * t_{wait} = \\ &= P_{idle} * t_{idle} + P_{sleep} * t_{sleep} + P_{standby} * t_{standby} \end{aligned} \tag{2.5}$$

I valori: P_{idle}, P_{sleep} e $P_{standby}$ indicano rispettivamente la potenza dissipata in idle, quella dissipata in modalità sleep e quella richiesta durante lo standby.
L'equazione (2.5) può essere quindi utilizzata solo se si possiedono i dati del disco per i tre parametri appena citati e se si è in grado di stimare per quanto tempo il processo in esame utilizzerà il disco nelle varie modalità.

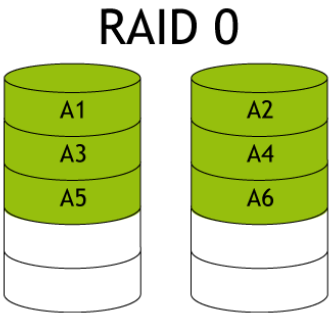


Figura 2.1: Struttura di un array con due dischi in modalità RAID 0

Array di dischi

Una particolare attenzione va posta se si vuole applicare l'equazione (2.4) a un array di dischi RAID.
Anzitutto bisogna aggiungere al modello l'energia assorbita dal controller dell'array (anch'esso considerato sia per il periodo in cui è inutilizzato che per quello in cui è utilizzato) ottenendo dunque:

$$E_{disk} = P_{write} * t_{write} + P_{read} * t_{read} + \\ + P_{wait} * t_{wait} + P_{controlleridle} * t_{wait} + P_{controller} * (t_{write} + t_{read}) \tag{2.6}$$

A questo punto si possono presentare due casi di diversa difficoltà: in uno i dischi dell'array sono tutti perfettamente identici (stessa capienza, stesso produttore e stesso modello) nell'altro l'array di dischi contiene almeno un disco diverso dagli altri.
Si illustrerà ora come l'equazione (2.6) deve essere modificato a seconda della modalità RAID utilizzata; questa analisi verrà fatta solo per tre configurazioni e per ciascuna si mostrerà quali accortezze avere sia nel caso che i dischi siano uguali sia che vi sia uno o più dischi differenti.
L'analisi riportata può essere facilmente estesa a qualsiasi tipologia di RAID analizzandone le peculiarità e adattando il modello proposto dalla eq. (2.6).
L'analisi che verrà presentata è valida a design-time ma potrebbe non essere utilizzabile se si volesse valutare il consumo energetico a runtime in quanto il sistema operativo potrebbe non fornire direttamente informazioni sui dischi dell'array se non attraverso tools di terze parti.

RAID 0 Striping In figura 2.1 è riportata l'architettura standard di un array formato da due dischi in RAID 0.
In questa configuranzione le informazioni da memorizzare vengono suddivise

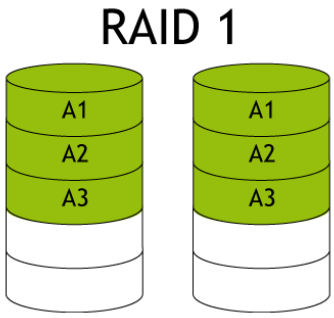


Figura 2.2: Struttura di un array con due dischi in modalità RAID 1

tra i vari dischi, in questo modo, se i dischi dell'array risultano essere identici tra loro, ogni disco verrà utilizzato per un tempo pari a una frazione del tempo totale di utilizzo della memoria di massa quindi l'equazione (2.6) può essere utilizzata senza particolari modifiche.
Il problema si presenta quando l'array è composto da dischi diversi ognuno dei quali può avere un suo consumo energetico bisognerà quindi valutare l'impatto energetico di ciascun disco dell'array supponendo che le informazioni da memorizzare siano suddivise equamente tra tutti i dischi utilizzando per ciascuno la eq. (2.4) e andando a termine ad aggiungere il costo energetico del controllore dato da: $P_{controlleridle} * t_{wait} + P_{controller} * (t_{write} + t_{read})$ in cui le componenti temporali rappresentano il tempo totale di uso dell'array da parte del processo.
Nel caso sia possibile avere le informazioni relative alla quantità di informazioni che il processo richiede di memorizzare è possibile valutare accuratamente i tempi di utilizzo di ciascun disco sulla base dei dati tecnici forniti dai produttori delle velocità di accesso e di rotazione dei dischi.

RAID 1 Mirroring In figura 2.2 è riportata l'architettura standard di un array formato da due dischi in RAID 1.
In questo caso il calcolo energetico deve tenere conto del fatto che ogni informazione memorizzata viene replicata su ogni disco dell'array ma in lettura viene utilizzato un solo disco.
Definito N il numero dei dischi presenti nell'array e considerando i dischi identici tra loro, l'equazione (2.6) per l'array RAID 1 diventa:

$$\begin{aligned} E_{disk} =& N * P_{write} * t_{write} + P_{read} * t_{read} + \\ &+ P_{wait} * t_{wait} + P_{controlleridle} * t_{wait} + P_{controller} * (N * t_{write} + t_{read}) \end{aligned} \tag{2.7}$$

Come già ricordato nel caso di RAID 0 se i dischi non sono perfettamente identici tra loro bisongerà avere qualche accorgimento aggiuntivo in partico-

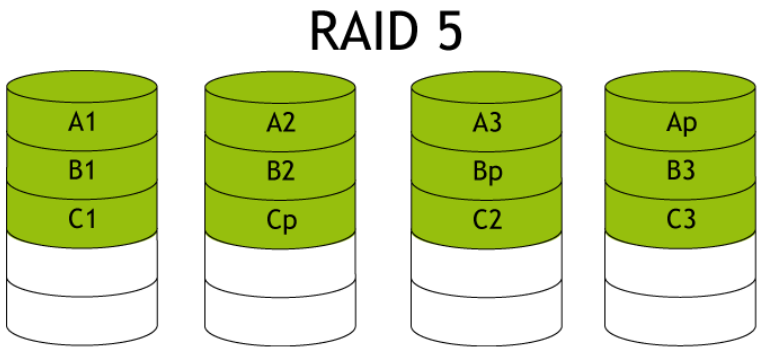


Figura 2.3: Struttura di un array con quattro dischi in modalità RAID 5

lare, bisognerà calcolare l'energia dissipata in scrittura da ciascun disco in base agli assorbimenti di potenza specifici. La potenza dissipata in lettura invece rimarrà invariata in quanto riferita a un singolo disco cioè quello rappresentante il master da cui vengono letti i dati.

RAID 5 In figura 2.3 è riportata l'architettura standard di un array formato da quattro dischi in RAID 5.
Il calcolo del consumo energetico della memoria di massa composta da n dischi in array RAID 5 è complesso a design time e dovuto al fatto che in questa fase non sono stimabili gli errori in lettura che causerebbero una ulteriore richiesta al disco per il calcolo della parità.
Tralasciando questa problematica il modello può essere adattatto tenendo conto del numero di dischi interessati in lettura e in scrittura avendo cura di considerare il numero di dischi in scrittura come il numero N totale dei dischi dell'array in quanto per ogni operazione di scrittura è necessario calcolare il valore di parità dei dati memorizzati. In lettura bisognerà invece considerare N-1 dischi in quanto non viene letto il blocco di parità se non sono identificati errori di CRC.
Il problema principale a design-time vi è se l'array non è composto da dischi identici. In questo caso se vi fossero dischi con diversi assorbimenti di potenza all'interno dell'array sarebbe pressochè impossibile una buona stima dell'energia dissipata durante l'esecuzione di uno o più servizi poiché in lettura non si saprebbe quale disco escludere dal bilancio energetico (verrebbero esclusi alternativamente a seconda di dove è memorizzato il blocco di parità per il dato richiesto) e in scrittura si dovrebbero calcololare i tempi di utilizzo dei singoli dischi sulla base dei blocchi scritti in ogni disco.
Ciò non preclude comunque la possibilità di realizzare un modello statistico ad hoc se vi fosse l'esigenza di estendere il modello qui presentato all'uso in

sistemi con array RAID composti da dischi differenti tra loro.

Dischi virtuali

Valutare l'impatto energetico delle operazioni di I/O sulla memoria di massa all'interno di un sistema virtuale è un compito assai difficoltoso in quanto in una virtual machine i dischi vegnono modellati tramite l'ausilio di uno o più file memorizzati sul disco fisico.
Il problema è di lieve entità quando l'immagine del disco della macchina virtuale è memorizzato su un singolo disco in quanto le operazioni di I/O effettuate dal processo in esecuzione sulla macchina virtuale corrisponderanno a letture e scritture dei file rappresentanti il disco virtuale e quindi a delle operazioni di I/O sul disco fisico.
Il modello rimane valido anche nel caso di dischi virtuali memorizzati in array RAID in quanto si potrà modificare il modello utilizzando gli accorgimenti illustrati in precedenza.
Nel caso di utilizzo di macchine virtuali è necessario essere a conoscenza durante la fase di design di quali dishi verranno allocati alla macchina virtuale così da poter applicare correttamente il modello.

2.2.4 Modello della scheda di rete

Come già illustrato all'inizio di questa sezione l'adattatore di rete ricopre un ruolo fondamentale durante l'esecuzione di un processo di business basato su servizi in quanto ciascun servizio può trovarsi in esecuzione su diversi server e per comunicare con gli altri deve trasferire informazioni mediante l'interfaccia di rete.
Il modello proposto per la scheda di rete è riassunto dalla seguente equazione:

$$E_{net} = P_{send} * t_{send} + P_{rcv} * t_{rcv} + P_{idle} * t_{idle} \tag{2.8}$$

Si noti come il modello presentato dalla eq. (2.8) sia simile a quello proposto per lo storage; l'adattatore di rete, come lo storage, è un'unità di I/O e come tale prevederà un'assorbimento di potenza dovuto alle operazioni di I/O e uno relativo al rimanere attiva in attesa di essere utilizzata.
Il modello proposto nella eq. (2.8) e nella eq. (2.4) potrebbe essere dunque generalizzato per una qualsiasi risorsa di I/O con la seguente:

$$E_{IO} = P_{input} * t_{input} + P_{output} * t_{output} + P_{idle} * t_{idle} \tag{2.9}$$

Tornando all'equazione (2.8) rappresentante l'energia richiesta dalla scheda di rete si può notare che questo valore è dato dalla potenza dissipata in ricezione (P_{rcv}), dalla potenza dissipata in invio (P_{send}) e dalla potenza di idle

(P_{idle}) ciascuna moltiplicata per il rispettivo tempo di utilizzo della scheda di rete nelle varie modalità.
L'utilizzo della eq. (2.8) per modellare la comunicazione di rete tra le virtual machine deve essere fatto tenendo in considerazione il fatto che la comunicazione di rete tra due macchine virtuali non sempre avviene utilizzando l'adattatore di rete fisico poiché se le macchine virtuali sono istanziate sullo stesso server fisico queste possono comunicare tra loro utilizzando una rete virtuale e gli adattatori virtuali.
Capire se un servizio comunicherà con altri servizi presenti in macchine virtuali installate sullo stesso server o con macchine virtuali installate su server remoti è facile da fare a design-time quando cioè viene progettato dove far eseguire ogni servizio poiché il designer è a conoscenza di quali macchine (fisiche e virtuali) vengono allocate al processo di business; è più complesso a runtime perché ogni virtual machine dovrebbe essere a conoscenza di quali host con cui sta comunicando sono nella stessa rete virtuale e quali invece si trovano su server remoti.

2.2.5 Modello completo

Grazie all'analisi effettuata fino a questo punto è possibile comporre il modello che rappresenta il consumo energetico totale di un server fisco o virtuale. L'energia assorbita dal server sarà dunque data da:

$$\begin{aligned} E_{server} =& E_{cpu} + E_{mem} + E_{disk} + E_{network} = \\ &= \sum_{m \in M} P_{cpu@m} * t_{exec@m} + \sum_{i=1}^{n} P_{memi} * t_{exec} + \\ &+ P_{write} * t_{write} + P_{read} * t_{read} + P_{wait} * t_{wait} + \\ &+ P_{send} * t_{send} + P_{rcv} * t_{rcv} + P_{idle} * t_{idle} + P_{bias} * t_{exec} \end{aligned} \tag{2.10}$$

Nell'equazione (2.10) appare il termine P_{bias} che rappresenta la potenza dissipata da tutte le componenti non considerate in dettaglio quali: ventole di raffreddamento, scheda madre, scheda video, ecc.
L'introduzione di P_{bias} nasce dal fatto che il consumo energetico di certi componenti può essere ignorato nel senso che si tratta di componenti che non variano sensibilmente la potenza assorbita durante l'esecuzione dei processi rispetto a quando sono in attesa. Si considera quindi la potenza di queste compoenti costante come fatto in [2] e modellato da P_{bias}.

2.3 Applicazione del modello

Il modello presentato può essere utilizzato a design-time per svolgere principalmente due compiti:

- valutare l'energia complessiva consumata dal datacenter per eseguire i processi di business che verranno eseguiti;

- valutare l'incremento di energia provocato dall'esecuzione del processo valutando quindi durante il design dell'allocazione dei servizi del processo quanta energia consuma ciascun servizio e di conseguenza l'esecuzione del processo.

Il modello descritto dall'equazione (2.10) può essere anche utilizzato come funzione costo per realizzare un modello di allocazione ottima a design-time dei servizi di un processo.
Per calcolare l'energia consumata da un datacenter durante l'esecuzione di un processo di business è sufficiente raccogliere i dati necessari illustrati nella sezione 2.5 ed applicarli all'equazione (2.10) in questo modo si avrà il consumo energetico totale durante l'esecuzione del processo.
Valutare invece l'energia consumata dal singolo processo o dal singolo servizio è leggermente più complicato.
Anzitutto bisogna valutare il consumo energetico del server in idle quindi senza carico. La stima di questo valore avviene utilizzando la eq. (2.10) modificata per valutare solo il comportamento in idle considerando un tempo t di esecuzione pari a un ora così da valutare l'energia assorbita in Wh.

$$\begin{aligned} E_{serveridle} =& E_{cpuidle} + E_{mem} + E_{diskidle} + E_{networkidle} = \\ =& P_{cpuidle} * t + \sum_{i=1}^{n} P_{memi} * t + P_{diskidle} * t + P_{netidle} * t + P_{bias} * t \end{aligned} \tag{2.11}$$

Calcolato il valore della eq. (2.11) si procede valutando il consumo energetico del server con il processo in esecuzione utilizzando la eq. (2.10) ottenendo quindi il valore di E_{server} misurato nella stessa unità di misura di $E_{serveridle}$. L'energia richiesta dal processo sarà quindi pari a:

$$E_{process} = E_{server} - E_{serveridle} \tag{2.12}$$

Analogamente è possibile ottenere lo stesso risultato valutando la differenza tra la potenza dissipata dal server durante l'esecuzione del processo e la potenza dissipata dal server senza carico. Ottenuta la potenza richiesta dal processo si può facilmente ottenere l'energia integrando questo valore con il tempo di esecuzione che il designer stima che avrà il servizio in esame. Ovviamente questo tempo di esecuzione può essere stimato oppure potrebbe essere un valore vero e proprio nel caso in cui il designer sappia che per un tempo ben definito il servizio sarà in esecuzione.

2.4 Allocazione ottima dei servizi

Il modello illustrato nella sezione 2.2 può essere utilizzato per decidere come allocare i servizi per ottenere un'esecuzione efficente dal punto di vista energetico.
In questa sezione verrà illustrato un modello di programmazione lineare per l'allocazione dei servizi che utilizza come funzione costo l'equazione (2.10).
L'obiettivo del modello è quello di trovare l'allocazione ottima dei servizi che costituiscono i processi sulle virtual machine a disposizione nel datacenter.
Il modello prevede che i servizi siano eseguiti su virtual machine ma sarebbe applicabile ugualmente se si volessero eseguire i servizi su dei server non virtualizzati.

2.4.1 Formulazione del modello

M: insieme delle virtual machine caratterizzate da $i = 1, ..., m$
S: insieme dei servizi $j = 1, ..., s$
P: insieme dei processi di business $k = 1, ..., p$

Variabili di decisione

$e_{i,j}$: energia consumata dal servizio j in esecuzione sulla virtual machine i

$$x_{i,j} = \begin{cases} 1 & \text{Se il servizio } j \text{ è in esecuzione sulla virtual machine } i, \\ 0 & \text{altrimenti.} \end{cases}$$

$$y_{j,k} = \begin{cases} 1 & \text{Se il servizio } j \text{ appartiene al processo } k, \\ 0 & \text{altrimenti.} \end{cases}$$

$$w_{j1,j2} = \begin{cases} 1 & \text{Se il servizio } j1 \text{ deve attendere il servizio } j2, \\ 0 & \text{altrimenti.} \end{cases}$$

Parametri del modello

$c_{i,j}$: capacità del processore della VM i richiesta dal servizio j
$r_{i,j}$: capacità della ram della VM i richiesta dal servizio j
$d_{i,j}$: capacità del disco della VM i richiesta dal servizio j
$t_{i,j}$: tempo di esecuzione del servizio j sulla VM i
t_{mink}: tempo minimo di esecuzione del processo k
t_{maxk}: tempo massimo di esecuzione del processo k

Funzione obiettivo

$$min \sum_{i=1}^{i=m} x_{i,j} * e_{i,j} \quad \forall j \in S \tag{2.13}$$

L'obiettivo del modello è quello di minimizzare l'energia richiesta dai processi di business per essere eseguiti.
Il termine $e_{i,j}$ rappresenta l'energia richiesta dal servizio j per essere eseguito sulla macchina virtuale i. Questo valore è calcolato utilizzando il modello dato dall'equazione (2.12). E' possibile anche valutare $e_{i,j}$ in termini di energia dissipata dalla VM j per eseguire il servizio i determinabile con la eq. (2.10) quindi, la funzione obiettivo minimizzerebbe l'energia totale richiesta dal datacenter: l'allocazione risulterà in ambedue i modi identica.

Vincoli

$$\sum_{i=1}^{i=m} x_{i,j} = 1 \quad \forall j \in S \tag{2.14}$$

$$\sum_{j=1}^{j=s} x_{i,j} * c_{i,j} \leq 1 \quad \forall i \in M \tag{2.15}$$

$$\sum_{j=1}^{j=s} x_{i,j} * r_{i,j} \leq 1 \quad \forall i \in M \tag{2.16}$$

$$\sum_{j=1}^{j=s} x_{i,j} * d_{i,j} \leq 1 \quad \forall i \in M \tag{2.17}$$

Il vincolo (2.14) impone che ogni servizio sia associato a un server.
I successivi tre vincoli rappresentano i vincoli derivanti dai limiti di capacità delle componenti considerate nel modello del server.
Il vincolo (2.15) chiede che su una VM vengano allocati servizi fino al raggiungimento della capacità massima della cpu.
Il vincolo (2.16) chiede che su una VM vengano allocati servizi fino al raggiungimento della capacità massima della memoria centrale.
Il vincolo (2.17) chiede che su una VM vengano allocati servizi fino al raggiungimento della capacità massima dello storage.

$$\sum_{i=1}^{i=m} \sum_{j=1}^{j=s} [p_{j,k} x_{i,j} [t_{i,j} + [\sum_{i1=1}^{i1=m} \sum_{j1=1}^{j1=s} w_{j,j1} t_{i1,j1}]]] \geq t_{mink} \quad \forall k \in P \tag{2.18}$$

$$\sum_{i=1}^{i=m}\sum_{j=1}^{j=s}[p_{j,k}x_{i,j}[t_{i,j}+[\sum_{i1=1}^{i1=m}\sum_{j1=1}^{j1=s}w_{j,j1}t_{i1,j1}]]] \leq t_{maxk} \quad \forall k \in P \tag{2.19}$$

I vincoli (2.18) e (2.19) servono per definire il tempo massimo e minimo di esecuzione di un processo.
Il vincolo impone che il tempo di esecuzione totale del processo di business dato dall'esecuzione di tutti i servizi facentene parte, deve essere minore di un tempo massimo e allo stesso tempo durate più di un tempo minimo.
Ovviamente ciascun vincolo proposto può essere considerato o meno dal designer a seconda dello scenario di utilizzo del modello e delle esigenze specifiche.
E anche possibile per il designer aggiungere vincoli specifici per particolari esigenze o modificare i vincoli esistenti per adattarli alle caratteristiche dei processi di business e delle architetture delle macchine virtuali a disposizione.
L'obiettivo del modello di allocazione qui proposto è quello di mostrare un semplice modello che utilizzando le equazioni energetiche definite nella sezione 2.2 permetta di progettare una allocazione efficiente dal punto di vista energetico dei processi di business basati su servizi.
Il modello non preclude nemmeno la possibilità che il designer decida che per particolari motivi alcuni servizi debbano essere eseguiti su una particolare macchina: in questo caso basterà aggiungere i vincoli necessari al modello qui proposto.

2.5 Modello concettuale delle annotazioni

In quest'ultima sezione verrà introdotto il modello concettuale delle annotazioni necessarie a design-time per poter utilizzare il modello proposto nelle sezioni precedenti.

2.5.1 Annotazione del server

Di seguito verranno illustrate per ciascuna componente del modello del server quali annotazioni sono necessarie per poter utilizzare le equazioni energetiche proposte per modellare il consumo di un server.
La figura 2.4 riassume le annotazioni necessarie per il modello che verranno illustrate nei prossimi paragrafi.

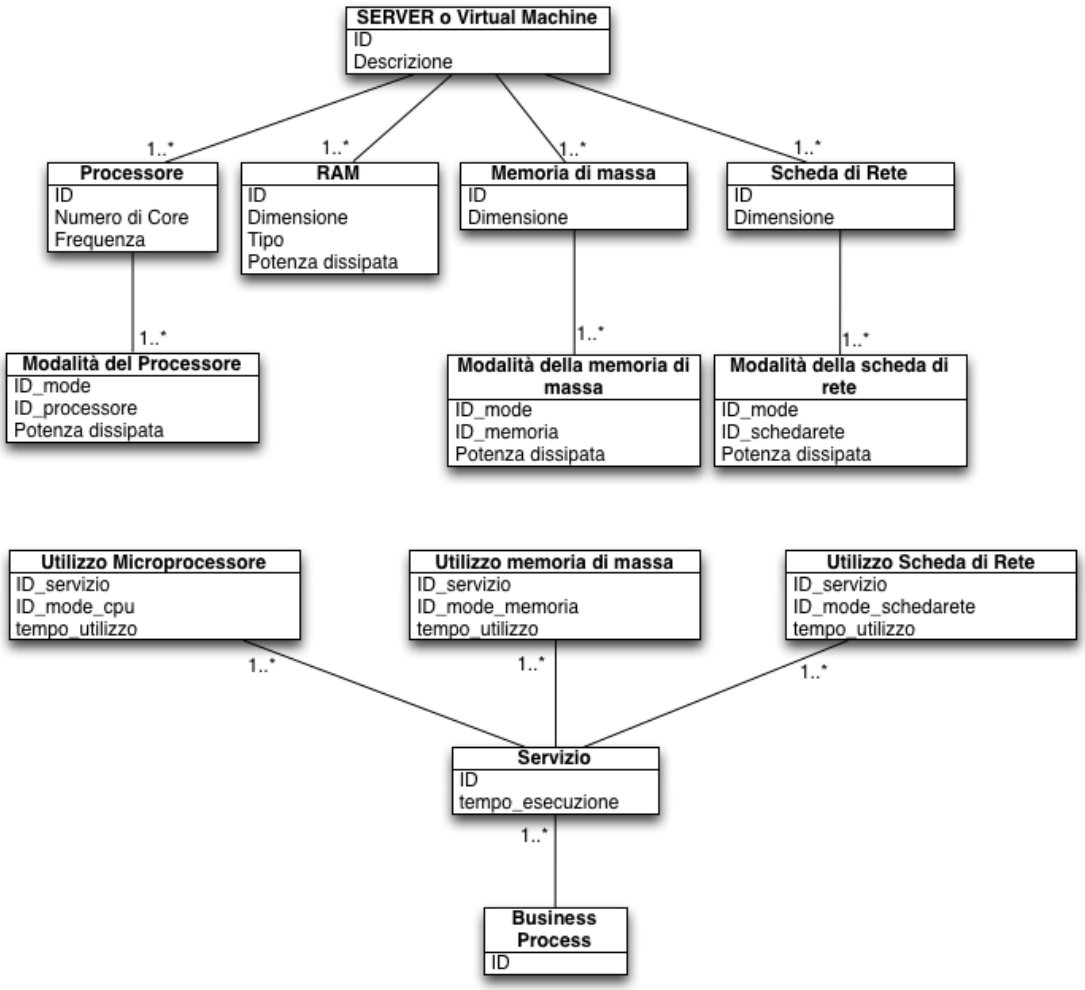


Figura 2.4: Modello concettuale delle annotazioni

Microprocessore

Per poter modellare correttamente il comportamento energetico del microprocessore è necessario fornire le informazioni sul consumo energetico del processore nelle sue diverse modalità di funzionamento.
Il numero minimo di modalità necessarie per valutare la viariazione di energia è due: cpu inattivo e cpu in uso.
Più modalità vengono rese disponibili maggiore sarà il dettaglio raggiungibile dal modello.
Per ciascuna di queste modalità è necessario avere il valore della potenza dissipata durante il funzionamento della cpu in quella modalità.

Memoria centrale

Per la memoria centrale non sono necessarie particolari informazioni: l'unica informazione richiesta è la potenza media richiesta durante il funzionamento.

Memoria di massa

Nel caso della memoria di massa le informazioni richieste sono maggiori in quanto si deve riuscire a modellare i dischi in tutte le loro fasi di funzionamento.
Anzitutto è necessario conoscere la potenza richiesta per eseguire le operazioni di I/O: nella maggior parte dei dispositivi di archiviazione la potenza richiesta in lettura è la stessa richiesta in scrittura mentre può variare la potenza dissipata nelle varie modalità di attesa e inattività.
In generale è sufficiente conoscere oltre alla potenza di I/O la potenza di inutilizzo del disco cioè quanto consuma il disco per rimanere semplicemente acceso. Si può però suddividere questa potenza in più parti permettendo dunque di caratterizzare meglio l'energia consumata.
In generale la potenza di inutilizzo può essere più specificatamente vista come: potenza di inutilizzo vera e propria, potenza di stand-by e potenza di sleep.

Scheda di rete

Come per il disco anche la scheda di rete richiede almeno due valori di potenza: uno che rappresenti il consumo durante le operazioni di I/O (che può coincidere tra lettura e scrittura o essere diverso) e uno che rappresenti la potenza dissipata durante l'inutilizzo della scheda.

Altre compoenti

Essere in possesso di altre informazioni relative all'hardware permette di modellare meglio il server ma non sempre di avere una valutazione più accurata del consumo energetico dei processi.
Sicuramente se il sistema in esame prevede uno storage organizzato con array RAID un dato fondamentale è a potenza dissipata dal controllore RAID.
Altri valori di potenza quali quello della scheda madre o delle ventole di raffreddamento, non comportano miglioramenti nell'accuratezza della stima dell'energia dissipata dai servizi in esecuzione in quanto sono valori di potenza che rimangono tipicamente invariati sia durante l'inutilizzo della macchina che durante le fasi di pieno carico.

2.5.2 Thermal Design Power vs Electrical Power

Durante tutta l'esposizione del modello e delle annotazioni ad esso necessarie si è spesso parlato di potenza dissipata dalle varie componenti.
Volendo fornire le annotazioni necessarie al modello della CPU i dati richiesti si riferiscono alle potenze assorbite dalle varie modalità di funzionamento del processore.
Cercando le informazioni di potenza nei datasheet dei processori spesso ci si imbatte in un valore denominato TDP o Thermal Design Power.
Questo parametro rappresenta un'indicazione del calore dissipato dal processore e serve per dimensionare il sistema di raffreddamento della CPU così da mantenere la temperatura d'esercizio all'interno dei limiti di sicurezza per un buon funzionamento.
Benché questo valore non rappresenti la vera potenza dissipata, in quanto solo una parte dell'energia fornita al processore si trasforma in calore, questo valore può essere considerato abbastanza buono per valutare il consumo energetico della CPU come supposto anche in [5].
Si tenga presente che il valore della TDP è inferiore rispetto alla reale potenza richiesta dal processore.
Un altro modo per calcolare la potenza dissipata da ciascuna modalità di funzionamento della CPU è quello di calcolarla come proporzionale alla corrente fornita in input alla CPU (dato che è possibile trovare nei datasheet).
Il miglior risultato si ottiene quando si riesce ad avere il valore esatto della potenza elettrica assorbita dal processore; questo valore può essere riportato nel datasheet (il dato è presente o facilmente calcolabile soprattutto nei datasheet di processori che prevedono diverse modalità di funzionamento per garantire il risparmio energetico) oppure può essere reperito attraverso banche dati specializzate (un esempio è [13]) o altre pubblicazioni in cui questo dato può essere frutto di misurazioni pratiche o di calcolo attraverso appositi

modelli.

2.5.3 Annotazione dei processi

In questa sezione verranno elencate le informazioni necessarie sui processi, e in particolare sui servizi che compongono ciascun processo, per poter applicare con successo il modello descritto dalla eq. (2.10).
Come per le annotazioni del server si suddivideranno le informazioni in base alle varie componenti hardware.
Le informazioni temporali richieste sono informazioni che devono essere stimate dal designer e che quindi possono provenire da benchmark effettuati sui servizi o da valutazioni fatte dal designer.

Microprocessore

Per poter applicare il modello del processore è necessario conoscere perlomeno il tempo di esecuzione del servizio in esame (dato che sarà richiesto per il calcolo del consumo energetico della memoria centrale).
Volendo ottere un modello più aderente al consumo energetico reale del microprocessore sarebbe auspicabile avere l'informazione relativa al tempo di esecuzione del servizio in ciascuna modalità di funzionamento della CPU e, nel caso in cui il servizio preveda di cambiare modalità di funzionamento della CPU durante la sua esecuzione, è bene avere i tempi di utilizzo per ciascuna modalità.

Memoria centrale

Per modellare la memoria centrale, come si può vedre dall'equazione (2.2), è sufficiente conoscere il tempo totale dell'esecuzione del servizio in analisi.

Memoria di massa

Le informazioni richieste sui servizi per la valutazione del consumo energetico dello storage richiedono di sapere per quanto tempo vengono utilizzati i dischi per le operazioni di I/O e per quanto invece si lasciano i dischi inutilizzati.
Come già detto per i parametri di potenza anche il tempo di inutilizzo può essere suddiviso in tre valori più specifici che permettono di ottenere una maggiore precisione nel calcolo dell'energia dissipata in particolare i valori richiesti sono: il tempo per il quale il disco viene effettivamente lasciato inutilizzato, il tempo durante il quale il disco risulta in stato di sleep e quello

in cui il disco risulta in stand-by.

Scheda di rete

Per la scheda di rete i parametri richiesti sono analoghi a quelli necessari per la memoria di massa; in particolare è richiesto il tempo speso in ricezione e in invio dal servizio valutato come tempo assoluto di ricezione e tempo assoluto di invio, nel caso si stia analizzando il consumo di un servizio in esecuzione su un server fisico o come tempo effettivo di trasmissione e ricezione, cioè il tempo effettivamente speso nell'utilizzo della scheda di rete del server fisico nel caso si tratti di esecuzione su macchina virtuale (quindi considerando solo il tempo in cui avvengono comunicazioni tra macchine virtuali disposte su server fisici diversi).
Inoltre è necessario il tempo in cui la scheda di rete viene lasciata inutilizzata. In generale i tempi di inutilizzo sono parametri che sono facilmente deducibili avendo i tempi di utilizzo delle risorse e il tempo totale dell'esecuzione di ciascun servizio.

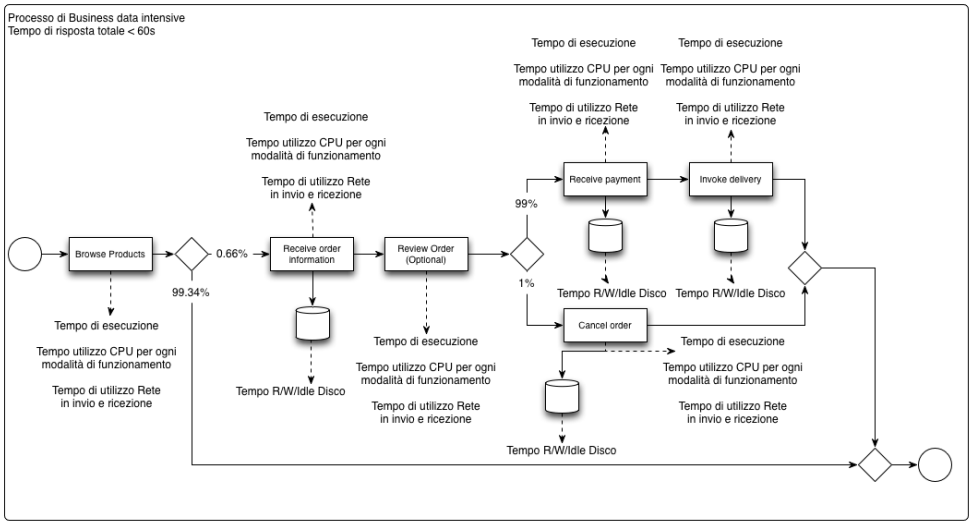


Figura 2.5: Esempio di processo di business con annotazioni

2.6 Conclusioni

In questo capitolo sono stati illustrati i modelli matematici proposti per fornire un sistema di supporto al designer di processi di business basati sui servizi per ottenere l'efficenza energetica dell'esecuzione di quest'ultimi.

Dovendo fornire dei modelli che possano essere utilizzati in fase di design del processo è stato necessario porre molta attenzione a quelli che sono i dati che verosimilmente il designer può conoscere in questa fase.
Talvolta il modello può sembrare semplice o addirittura semplicistico il motivo è che la complessità di un modello va di pari passo con le informazioni necessarie al buon funzionamento del modello. A design-time purtroppo non sono disponibili molte informazioni che invece possono essere reperite a runtime e che permettono una migliore valutazione dell'effettivo consumo energetico di ciascun servizio e di ciascun processo.
Nel successivo capitolo si illustreranno le metodologie adottate per validare il modello e l'architettura di test utilizzata.
Nell'ultimo capitolo verrà poi valutato nuovamente il modello sulla base dei dati ottenuti nei test e si illustrerà ampiamente come il modello qui proposto, benché semplice, sia un'ottimo strumento per stimare l'impatto energetico dei processi in un datacenter e quali possibili sviluppi potrà avere questo lavoro.

Capitolo 3

Validazione del modello

Nelle prossime sezioni verrà affrontato il problema della validazione del modello proposto nel capitolo precedente.
Per poter definire se il modello è valido e realmente utilizzabile si è proceduto effettuando una serie di test volti a raccogliere i dati del consumo energetico di alcuni processi di benchmark appositamente realizzati.
I test sono stati effettuati utilizzando il cluster di server a disposzione per il progetto GAMES presso l'High Performance Computing Center (HLRS) di Stoccarda.
Nella prima sezione di questo capitolo verrà illustrata la configurazione dei server di test soffermandosi sui sensori installati su ciascun server per la raccolta dei dati sul consumo energetico. In questa fase verrà anche analizzata la struttura di fcim, il database incaricato di memorizzare i dati raccolti dai sensori.
Nella seconda sezione verranno invece presentati lo script di test ideato e in quale modo esso avrebbe messo alla prova il server e il modello.
Infine si analizzeranno i dati che è stato possibile misurare con certezza e si discuteranno i risultati ottenuti.

3.1 Struttura del cluster

L'infrastruttura di rete dei server coinvolti nel progetto GAMES presso HLRS è riportata nella figura 3.1 Di tutte le aree rappresentate in figura ne sono state utilizzate principalmente solamente due per i test: il front-end server e il cluster dei 18 nodi di computazione.
Il punto di accesso tramite ssh al sistema è il Front-end Server talvolta indicato con il nome GAMES head o GAMES head server. Attraverso l'accesso a questo server è poi possibile utilizzare i nodi del cluster.
Il GAMES head e i singoli nodi condividono la home directory dell'utente con cui si accede quindi lo spazio di memorizzazione è condiviso. Questo fat-

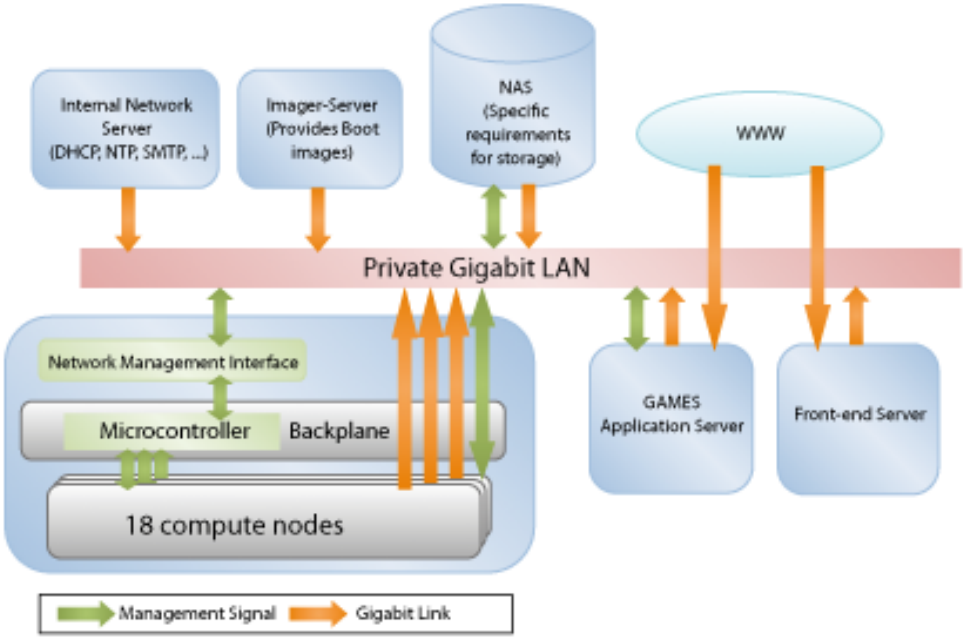


Figura 3.1: Struttura dell'infrastruttura di test

to è importante poiché, non avendo piena conoscenza del disco su cui sono memorizzati i dati, è difficile valutare accuratamente il consumo energetico dello storage.
Il GAMES head è un server dotato di un doppio processore dual core intel Xeon 5160 è privo di hard disk in quanto utilizza lo storage condiviso con i nodi del cluster.
Per quanto riguarda la memoria centrale l'unica informazione a disposzione è che il server dispone di 6 GB di RAM Kingston in banchi da 2GB ciascuno.
L'unità cluster a cui si può avere accesso via ssh tramite GAMES head è composta da 18 unità ognuna fornita di un processore intel dual core P8400 e 4GB di RAM DDR3 in configurazione dual-channel (quindi 2 banchi da 2GB).
I datasheet delle componenti citate nella descrizione hardware dei server sono riportati nell'appendice A; i consumi energetici a cui si farà riferimento in questo capitolo sono tutti ricavati dai dati presenti nella suddetta appendice.

3.1.1 Rilevazione dei consumi e database fcim

Al fine di conoscere il reale consumo di potenza delle varie macchine sono stati installati su ciascun server e ciascun nodo del cluster una serie di sensori utili a valutare diversi parametri tra cui gli assorbimenti di potenza.
I dati vengono rilevati ogni 3 minuti e memorizzati all'interno di un database

denominato fcim.
Nelle successive tabelle vengono riportate le informazioni rilevate per le componenti utilizzate durante i test.
Le informazioni raccolte dai sensori installati sul front-end server vengono memorizzate nella tabella 'fcim_server_consumption' mentre i dati relativi ai nodi del cluster nella tabella 'fcim_recs_board'.

Campo	Descrizione	Unità
server	nome del server	-
power	potenza dissipata	W
timestamp_id	id dell'istante di rilevazione	-

Tabella 3.1: Struttura della tabella fcim_server_consumption

Campo	Descrizione	Unità
board_id	identificativo del nodo	-
temperature	temperatura della mainboard	°C
power	potenza dissipata dalla mainboard	W
timestamp_id	id dell'istante di rilevazione	-

Tabella 3.2: Struttura della tabella fcim_recs_board

E' importante notare la differenza tra il dato contenuto nel campo power della tabella fcim_server_consumption e quello nello stesso campo della tabella fcim_recs_board: nel primo caso per potenza dissipata si intende la potenza totale assorbita da tutto il sistema, nel secondo la sola potenza dissipata dalla scheda madre.
Nel database fcim sono contenute molte informazioni ma qui sono state presentate solo quelle utilizzate per controllare la validità del modello proposto. La struttura completa del database con la descrizione di ogni singolo campo è riportata nell'appendice B.

3.2 Script di benchmark

Per poter valutare se il modello proposto stima correttamente o meno il consumo energetico di un server si è realizzato uno script di benchmark in grado di utilizzare al massimo le risorse del sistema (o almeno di parte del sistema) al fine di valutare la variazione di potenza dissipata rilevata dai sensori installati sulle macchine presenti nel datacenter HLRS.
Nello sviluppo di questo script si è tenuto conto anzitutto della durata dell'esecuzione dello stesso: premesso che i dati vengono campionati con un intervallo di 3 minuti è stato necessario progettare uno script che permettes-

se di avere dati stabili e attendibili quindi che avesse un tempo di esecuzione tale da permettere di rilevare più volte la potenza dissipata durante l'esecuzione.

3.2.1 Struttura dello script

Lo script sviluppato doveva inizialmente permettere di valutare due aspetti principali e facilmente verificabili: la variazione del consumo energetico del sistema quando viene utilizzata la CPU al massimo e quella del sistema quando viene utilizzato il disco in scrittura e in lettura.
I dati forniti dal sistema hanno permesso di valutare solamente la variazione di consumo energetico della CPU e non del disco poiché, come detto in precedenza, il GAMES head server è privo di dischi mentre sui nodi del cluster viene rilevata solo la potenza assorbita dalla scheda madre.
Inoltre, avendo sia sul front-end server che sui nodi interni, accesso alla sola cartella home che è condivisa non è stato possibile trovare l'esatto disco sullo storage utilizzato per questa cartella e conseguentemente, utilizzare la funzionalità di stress del disco messa disposizione dallo script.
L'esecuzione dello script può essere suddivisa in due macro fasi. La prima fase é suddividibile in quattro cicli, in ciascuno di essi viene inizializzata una variabile con un numero ben definito e via via maggiore di caratteri. I caratteri vengono accodati alla variabile a piccoli blocchi (10 caratteri alla volta). Il contenuto delle variabili così inizializzate (1024, 10240, 20240 e 40240 caratteri ciascuna) viene memorizzato su file.
Terminata la prima fase viene eseguita la seconda in cui ciascun file creato nella prima fase viene letto e il suo contenuto è stampato a video.

3.2.2 Modalità di esecuzione degli script

La garanzia sull'accuratezza dei dati raccolti durante l'esecuzione degli script è data non solo dalla precisione dei sensori ma anche dal fatto che i test sono stati fatti su macchine completamente prive di carico. Durante i test sia il front-end server che i singoli nodi risultavano non utilizzati da altri processi quindi i dati raccolti durante l'esecuzione dello script si riferiscono alla variazione della potenza assorbita a causa della sola presenza del processo di benchmark.

3.3 Esito dei test

Lo script di benchmark è stato eseguito diverse volte sia sul server head che su alcuni dei nodi del cluster interno ottenendo sempre dati molto simili se

non uguali.
Purtroppo le informazioni fornite da fcim non consento di valutare singolarmente ciascuna componente considerata nel modello però è possibile fare un confronto tra il valore stimato dal modello per il consumo complessivo del server e quello rilevato da fcim.
Volendo dunque rilevare il consumo complessivo di una delle macchine lo script è stato eseguito sul front-end server del quale viene monitorata la potenza assorbita.
L'esecuzione dello script è durata meno di nove minuti e nella tabella seguente vengono riportati i valori di potenza rilevati durante una delle esecuzioni dello script.

Timestamp	Server	Power [W]
2011-11-03 00:23:28	head	262.0
2011-11-03 00:26:28	head	289.4
2011-11-03 00:29:28	head	290.1
2011-11-03 00:32:28	head	262.8

Tabella 3.3: Esempio di dati rilevati durante l'esecuzione del benchmark sul head server

I dati presenti nella tabella 3.3 mettono subito in luce una variazione della potenza richiesta dal server quando viene avviata l'esecuzione di un processo. Poiché i dati in possesso non sono continui ma discreti e campionati ogni tre minuti è impossibile vedere la variazione di potenza nella transizione tra inutilizzo e utilizzo e viceversa.
Sempre a causa della frequenza di campionamento bisogna considerare un valore medio dei dati rilevati sia in fase di inutilizzo del server che durante l'esecuzione. Dover mediare i dati si rende necessario anche perché il valore rilevato è un valore complessivo dei consumi di diverse componenti che variano la loro richiesta energetica a seconda del compito che devono svolgere. Inoltre il modello presentato non valuta la potenza dissipata in un preciso istante dal server bensì il consumo medio di potenza (e conseguentemente di energia) di un processo.

Server	Power [W]	Descrizione
head	259.9741	Potenza media dissipata in Idle
head	285.0527	Potenza media dissipata durante il benchmark
head	25.0786	Incremento medio di potenza

Tabella 3.4: Dati medi sul consumo di potenza del front-end server

In tabella 3.4 vengono riportati i valori medi della potenza in idle del sistema (valore medio calcolato sull'intera storia disponibile) e durante l'esecuzione degli script di benchmark (valore medio calcolato utilizzando le rilevazioni fatte per tutti i test).
Con i dati disponibili è quindi facile calcolare l'aumento medio di potenza che vi è quando il server passa dallo stato di riposo allo stato di lavoro semplicemente calcolando la differenza dei valori medi delle rispettive potenze ottenendo un incremento di circa 25W (25.0786W).
E' con questo valore sperimentale che deve essere messo alla prova il modello presentato in precedenza; anzitutto si valuta con il modello il consumo di potenza che si ha in idle applicando l'equazione (2.10) utilizzando i parametri caratteristici di ciascun componente.
I parametri in possesso sul server head sono solamente quelli presenti in tabella 3.5 poiché il sistema è privo di dischi e non si hanno informazioni nè sulle schede di rete (vi sono due schede di rete presenti nel sistema) nè sulla scheda madre.

Componente	**Power Idle [W]**	Power Used [W]
CPU Intel Xeon 5160	104.84	130.83
RAM Kingston	2.025	2.025

Tabella 3.5: Consumo energetico delle componenti del front-end server

La presenza di pochi dati non non crea alcun problema per il meccanismo di verifica del modello.
Secondo quanto esposto al capitolo precedente il server in esame viene scoposto nelle sue componenti principali: cpu, ram, dischi e schede di rete.
La ram in realtà viene trattata come una componente particolare: è sostanzialmente un valore costante che serve più che altro per il calcolo dell'energia complessiva del server ed è utile quando si decide di accendere un ulteriore server per allocare i servizi da eseguire.
Il modello dei dischi non può invece essere valutato in quanto sul server in esame non vi sono dei dischi installati inoltre, il tempo richiesto per eseguire l'operazione di scrittura per file di dimensioni ridotte come quelli creati dal benchmark è trascurabile.
La scheda di rete risulta difficilmente valutabile in quanto non si hanno informazioni accurate sul modello esatto della scheda ma solo sul costruttore (MSI).
A questo punto ciò che rimane facilmente verificabile è il consumo della CPU la quale, monitorata con il tool top, risulta essere utilizzata al 100% praticamente per tutto il tempo di esecuzione del processo di benchmark. Inoltre, essendo il processo single thread, viene usata una sola delle due CPU al 100% mentre l'altra rimane in idle.

Server	Power [W]	Descrizione
head	209.68	Potenza stimata in Idle della CPU
head	235.67	Potenza stimata durante il benchmark della CPU
head	25.99	Incremento stimato di potenza

Tabella 3.6: Stime potenza dissipata utilizzando modello energetico

Come si può notare usando il modello proposto si ipotizza che durante il test l'incremento di potenza assorbita sia da imputarsi solamente alla CPU e valutandone l'aumento di richiesta energetica si può osservare che questa si discosta di poco dall'aumento misurato.

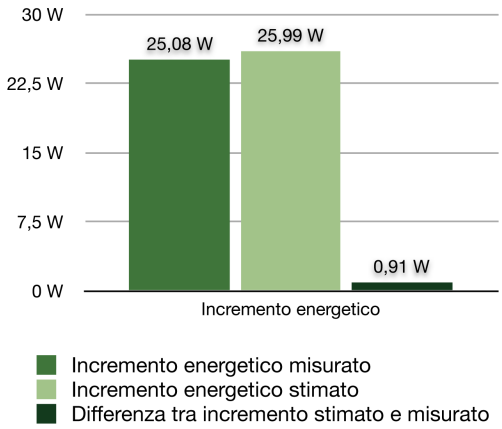


Figura 3.2: Confronto tra incremento energetico stimato e misurato

Il risultato dimostra la validità del modo di ragionare del modello: stimando quali componenti del sistema utilizzerò e per quanto tempo posso, a design-time, valutare la differenza di potenza generata e quindi l'energia consumata.
Il modello ovviamente non può a priori decidere di non considerare alcune equazioni perché la componente in questione non è usata dal servizio in esecuzione (scelta adottata in questa sezione per snellire l'esposizione dei dati raccolti); quindi esegue questa operazione indirettamente.
Come già illustrato in precedenza una delle modalità per calcolare l'energia dissipata dal singolo servizio è: valutare l'energia richiesta in idle dal server, calcolare l'energia richiesta durante l'esecuzione del servizio e fare la differenza tra questi due valori. Così facendo in effetti è come se si tenesse conto solo delle componenti che variano la potenza richiesta durante l'esecuzione del servizio. Lo stesso ragionamento ma riferito alla potenza è ciò che è stato

presentato in questa sezione; utilizzare la potenza invece che l'energia è una scelta derivata dal fatto che le misurazioni sul sistema reale avvengono in termini di potenza e se si fosse deciso di lavorare in termini di energia si sarebbe dovuto calcolare tale valore a partire dal valore di potenza misurato andando inevitabilimente a rendere un valore di controllo frutto di sole misurazione in un valore in parte calcolato.

3.3.1 Validità del modello del disco

Come più volte ricordato nè il front-end server nè i nodi del cluster hanno dei dischi rigidi o delle unità allo stato solido collegate poiché la memorizzazione dei dati è condivisa e dislocata su un particolare server di storage (chiamato appunto storage).
A questo stadio dello sviluppo dell'infrastruttura hardware presso hlrs il server di storage monta due dischi WD3000HLFS i cui consumi di potenza sono riportati nella tabella 3.7.

Modalità	Power [W]
read/write	6.00
idle	4.50
variazione idle-in uso	1.5
standby	0.40
sleep	0.40

Tabella 3.7: Caratteristiche energetiche dell'hard disk WD3000HLFS

Non scrivendo grandi quantità di dati su disco, lo script di test risulta inadeguato per verificare il modello della memoria di massa.
Per questo motivo è stato creato un file molto grande su disco (dell'ordine del Gigabyte di grandezza) e si è misurata la variazione di potenza durante la copia di tale file da una cartella a un'altra.
Procedendo in modo analogo a quanto fatto con la CPU in precedenza si è valutato il valore medio della potenza di idle e il valore medio della potenza durante l'utilizzo dello storage server.

Descrizione	Power [W]
Valore medio storage in uso	231.63
Valore medio storage in idle	226.48
Incremento medio	5.15

Tabella 3.8: Consumo di potenza storage server

E' immediatamente visibile come il valore rilevato presente in tabella 3.8 (5.15W) si scosti fortemente dal valore atteso della variazione di potenza assorbita dal disco riportato in tabella 3.7 (1.5W).
Il primo motivo di questa differenza deriva dalla configurazione dei dischi: sul server sono installati due dischi identici ma non si hanno informazioni su come vengono utilizzati. Nel caso fossero configurati in RAID 1 si dovrebbe considerare il fatto che ciascun disco consuma esattamente il doppio che in configurazione RAID 0 o senza RAID quindi, la differenza di potenza richiesta stimata passerebbe da 1.5W a 3W ancora però al di sotto del valore rilevato.
Un'altra considerazione riguarda la potenza dissipata dal controllore: le specifiche hardware danno solo un valore tipico di assorbimento di potenza stimato 4.62W; la variazione potrebbe dunque dipendere da una richiesta maggiore di potenza da parte del controller.
Bisogna inoltre ricordare che lo storage server altro non è che un normalissimo server sul quale sono allocati diversi dischi. L'operazione di copia oltre a utilizzare il disco potrebbe utilizzare anche la CPU la quale verrebbe a introdurre un aumento della potenza dissipata.
Purtroppo non è possibile analizzare il consumo del disco stimato con maggiore accuratezza a causa della mancanza di maggiori specifiche informazioni tecniche.

3.3.2 Componenti con dissipazione costante

Oltre alla semplice validazione del modello fatta fino a questo punto, grazie ai sensori di fcim è stato possibile verificare se le componenti scelte per modellare il server fossero sufficienti.
Si è detto che vi sono in un server una serie di componenti il cui consumo energetico è pressoché costante nel tempo e non varia durante l'esecuzione di un servizio.
La scheda madre, con i vari bus e collegamenti alle periferiche, è l'unico componente che potrebbe far pensare che abbia un consumo energetico variabile nel tempo. Se così fosse la modellazione proposta non sarebbe valida in quanto mancherebbe di fornire un'equazione per la stima dell'energia dissipata dalla mainboard.
Per verificare se il consumo della scheda madre non varia nel tempo si sono usate le unità del cluster presente nell'architettura GAMES presso HLRS. Ciascun nodo è provvisto di sensori che rilevano la potenza dissipata dalla scheda madre.
Andando ad eseguire lo script di benchmark su vari nodi del cluster si è potuto notare che la potenza dissipata dalla scheda madre rimane sostanzialmente invariata (a meno di qualche variazione fisiologica di circa 1W).
Grazie a questa verifica si è potuta escludere la necessità di modellare anche

la mainboard.
Non vi sono sensori per eseguire lo stesso test su altre parti però all'interno di un server non vi sono altre componenti che potrebbero, durante l'esecuzione di un processo, variare la loro richiesta di potenza.

3.4 Conclusioni

In questo capitolo si è mostrata la metodologia con cui sono stati effettuati i test sul modello proposto.
Gli esiti ottenuti hanno messo in luce come il modello possa fornire una buona stima della potenza dissipata da un generico processo basandosi solamente su informazioni note a design-time.
Avendo a disposizione più server ed eventualmente sensori su ogni componente è possibile verificare con maggiore precisione il modello proposto. Inoltre se il server venisse assemblato con componenti di cui si hanno tutti i datasheet contenenti i valori di potenza per ciascuna componente sarebbe possibile stimare ancora meglio la potenza dissipata.

Capitolo 4

Designer application

In questo capitolo verrà presentata l'applicazione sviluppata a partire dal modello energetico presentato nel capitolo 2.
Come precedentemente illustrato, lo scopo di questa tesi è quello di produrre un sistema a supporto del design-time per l'efficienza energetica dei processi di business; a partire quindi dal modello del consumo energetico di un generico server in un datacenter si è progettata e realizzata una applicazione J2EE web based che potesse aiutare il designer ad allocare i servizi di ciascun processo sui singoli server.
Nelle prossime sezioni verranno anzitutto mostrate le caratteristiche dell'applicazione, si proseguirà analizzando le scelte di progetto adottate e mostrando infine il design del software realizzato.

4.1 Specifiche

Il software implementato fornisce al designer un aiuto nello scegliere come allocare i processi di business in un datacenter cercando di ottimizzare l'impatto energetico che questi avranno durante la loro esecuzione.
La prima feature necessaria è che sia possibile per il designer descrivere la struttura del processo di business in esame rappresentando tramite grafi i legami tra i servizi che compongono il processo.
Sviluppato il processo, il designer deve essere in grado di fornire all'applicazione tutti i parametri di base che la descrivono e definire eventuali altri processi in esecuzione sulle singole macchine.
L'applicazione, preso in input il processo di business da analizzare, deve poter fornire un valore stimato dell'impatto energetico di tale processo sia riferito al consumo totale del datacenter nel momento dell'esecuzione del processo che all'assorbimento proprio di quest'ultimo.

4.1.1 EPKB

L'Energy Practice Knowledge Base (EPKB) è un database sviluppato all'interno del progetto GAMES che contiene tutte le informazioni sulla struttura e composizione del datacenter.
Sono inoltre presenti le descrizioni di tutte le applicazioni in esecuzione nel datacenter con la loro struttura e i legami con l'hardware su cui sono in esecuzione.
Grazie a questo database è possibile conoscere la struttura del datacenter e utilizzarla nell'applicazione per allocare i processi. E' inoltre possibile conoscere i processi già in esecuzione e la macchina che li sta eseguendo.
Oltre a queste informazioni nell'EPKB si suppone di avere tutti i dati dell'hardware necessari al modello e forniti dai datasheet delle componenti installate nei vari server del datacenter.
L'EPKB, insieme al modello energetico del server, rappresenta il mattone sul quale si sviluppa il tool che si sta presentando in questo capitolo.

4.2 Scelte di progetto

L'applicazione da realizzare ha al suo interno due macro componenti fondamentali: la prima, come per ogni applicazione, è l'interfaccia grafica mentre la seconda è il motore di calcolo del modello matematico.
L'interfaccia grafica rappresenta uno dei due principali punti critici in quanto è ciò che differenzia un'applicazione usabile da una non usabile. Quando però si sviluppa una applicazione web il problema dell'interfaccia si accentua a causa delle problematiche di compatibilità tra browser delle librerie javascript, degli attributi CSS e dalle differenti interpretazioni di alcuni tag HTML.
Per risolvere queste criticità sono stati sviluppati diversi framework tra questi vi è Vaddin che verrà presentato nella prossima sezione.
Il problema del motore per il calcolo del modello è altrettanto importante poiché avere un'applicazione usabile ma non funzionante è completamente inutile: di per sè il modello del consumo energetico del server non è difficile da modellare in Java ma ciò che bisogna garantire è anche l'allocazione dei servizi di un processo sui server disponibili e questo è un problema difficilmente risolvibile usando solo Java.
A questo scopo è stato usato lp_solve che verrà presentato successivamente.

4.2.1 Vaadin

Vaadin è un framework Java per la realizzazione di applicazioni web.
Questo framework offre alcune caratteristiche molto interessanti che lo contraddistinguono dall'utilizzo di librerie javascript (ad esempio jQuery o Moo-

Tools).
Anzittutto Vaadin offre un'architettura server-side ciò significa che le pagine web vengono compilate lato server completamente così che la gran parte della logica viene eseguita al sicuro sul server. Al contrario, utilizzando librerie come jQuery, parte della logica applicativa viene eseguita dal client in quanto l'architettura dominante in queste soluzione è client-side.
Secondo aspetto molto importante è quello riassunto dal motto di Vaadin:- Think of U and I. Il framework permette al programmatore di pensare all'utente e allo stesso tempo all'interfaccia infatti è possibile sviluppare complesse logiche applicative e renderle facili grazie a Vaadin e alla sua capacità di realizzare facilmente interfaccie grafiche complete e usabili.
Vaadin può essere inoltre facilmente esteso con la grande libreria di add-ons disponibili sul sito ufficiale o realizzando specifici plug-in per le proprie esigenze.
Il vero punto di forza di questo framework però è la compatibilità: il programmatore può sviluppare tranquillamente l'applicazione utilizzando solo ed esclusivamente linguaggio Java, Vaadin provvederà a generare le pagine, gli script lato client e a rendere l'aspetto grafico compatibile con ogni browser caratteristica resa possibile grazie all'utilizzo del Google Web Toolkit.
Vaddin è stato utilizzato per poter realizzare le interfacce grafiche del software: la scelta è ricaduta su questo framework per le notevoli semplificazioni che introduce nella programmazione di applicazioni web e per il fatto che permette di realizzare tutto usando sempre il medesimo linguaggio potendo quindi integrare in maniera pulita la logica di programma alla rappresentazione grafica come si riesce a fare con le applicazioni desktop Java.

4.2.2 lp_solve

lp_solver è un motore per la soluzione di problemi di programmazione lineare; per risolverli utilizza l'algoritmo del simplesso e il branch and bound per i problemi interi.
Questo motore può essere utilizzato come risolutore abbinato a linguaggi di modellazione come ad esempio AMPL.
Oltre a questo lp_solve dispone di un wrapper Java ovvero di un un insieme di classi capaci di invocare il risolutore passandogli i parametri del problema e ritornando la soluzione calcolata.
Il fatto che questo motore possa essere utilizzato sia abbinato a linguaggi di modellazione che a linguaggi di programmazione fa sì che si possa modellare un comportamento utilizzando un linguaggio apposito e poi generalizzare e dinamicizzare tale modello mediante un linguaggio di programmazione continuando a poterlo risolvere.
Una caratteristica utile del wrapper Java per lp_solve è quella di poter esportare su file i modelli creati tramite il linguaggio di programmazione: in

questo modo è possibile sia tenere un archivio dei modelli creati che elaborarli all'interno di altri strumenti.

4.3 Il design dell'applicazione

In questa sezione verrà presentato il design dell'applicazione ovvero come è stato implementato il software di cui sono state esposte le caratteristiche in precedenza.
Lo strumento realizzato può essere analizzato dividendolo in tre macro sezioni: interfaccia grafica, comunicazione con il database e gestione del modello. Ciascuna parte è caratterizzata nell'applicazione reale da un package che contiene tutte le classi atte ad implementare le funzionalità richieste.

4.4 Architettura

Figura 4.1: Architettura dell'applicazione

4.4.1 Il package ui e la gestione dell'interfaccia grafica

Come già illustrato nella sezione precedente per la gestione dell'interfaccia grafica ci si è affidati a Vaadin, un framework Java per la realizzazione di applicazioni web-based.
Il problema principale dell'interfaccia è stata la realizzazione di un editor che consentisse al designer di definire i processi di business come insieme di servizi.
Si è pensato di rappresentare un processo come il grafo dei servizi che lo compongono. Si è dovuto quindi implementare un editor di grafi all'interno dell'interfaccia grafica che permettesse sia di caricare le strutture dei processi presenti nell'EPKB che di crearne di nuovi.
Per fare ciò si è utilizzato il plug-in Node Graph Widget [11] implementato da Mauro Monti e reso disponibile sul sito ufficiale di Vaadin.

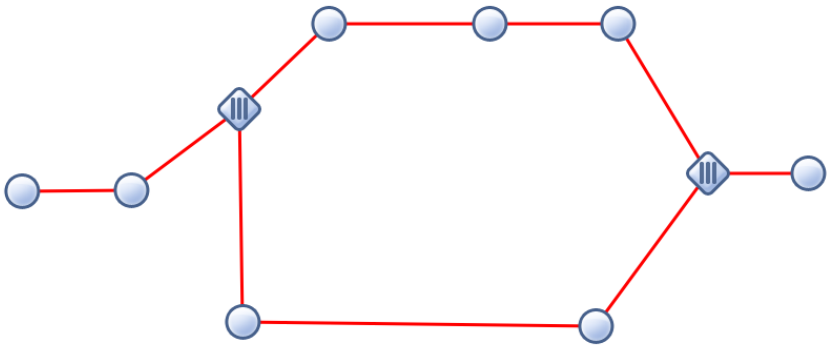


Figura 4.2: Grafo di un processo di business realizzato con Node Graph Widget modificato

Questo widget permette di generare e visualizzare grafi dinamicamente, non permette però l'aggiunta o la modifica di nodi e relazioni una volta che il grafo è stato creato.
Si è quindi provveduto, partendo dal codice sorgente del widget, a implementare le funzionalità mancanti gestendo sia l'aggiunta dinamica di nuovi nodi e relazioni che la loro successiva modifica.
Avendo quindi a disposizione un editor di grafi perfettamente funzionante è stato implementata una categoria di nodi per rappresentare le varie entità definibili all'interno del designer: la classe GAMESNode. Questa classe, estesa da ogni nodo utilizzato in questo tool, modella il generico servizio e mantiene traccia delle annotazioni necessarie al modello.

4.4.2 Il package EPKB e la comunicazione con il database

La comunicazione con il database dell'EPKB è il secondo punto fondamentale per la realizzazione dell'applicazione in quanto l'EPKB è l'archivio di tutte le informazioni necessarie al modello per funzionare.
Per comunicare con il database si è deciso di utilizzare le Java Persistence API (JPA) un framework per Java che si occupa di gestire i dati dei database relazionali come è appunto l'EPKB.
Nel package `EPKB.beans` sono inserite le classi Java che implementano gli entity bean delle rispettive tabelle del database gestite da JPA. Nel package `EPKB` vi è invece la classe `EPKBManager` contenente i metodi per leggere e scrivere informazioni sulla base dati.
L'uso di JPA offre il vantaggio di non dover interagire direttamente con il database relazionale ma di poter mantenere una certa astrazione da esso utilizzando gli entity bean che si presentano come dei veri e propri oggetti Java.

4.4.3 Il package models e la gestione del modello

Il package models è la terza parte fondamentale del tool qui presentato.
In questo package è presente la classe che implementa il modello energetico proposto nel capitolo 2 chiamata `EnergyModel`.
Isolare questa parte del software in un package apposito non è frutto della sola volontà di mantenere un codice pulito ma è dovuto anche dal voler predisporre il codice sorgente (e il software) a poter essere ampliato con l'aggiunta di ulteriori modelli. In questo modo si può offrire al designer un applicativo che, con il tempo e le future ricerche, può crescere e permettere valutazioni energetiche diverse ed eventualmente più accurate.
Più che analizzare la struttura complessiva della classe `EnergyModel` si mostreranno i punti salienti dell'implementazione del modello di allocazione ottima dei servizi presentato nella sezione 2.4 e il modello del consumo energetico della sezione 2.2.

Definizione della variabili del modello energetico

Prima di tutto sono state create le variabili di impostazione del modello con le quali è possibile definire il numero dei server disponibili e dei servizi da allocare.

```
int servers;
int services;
```

Sono state inoltre definite le variabili contenenti le informazioni sulle caratteristiche hardware di ciascun server: il vettore indicante per ciascun server

il numero di dischi e quello contenente le indicazioni sul numero di modalità di funzionamento della cpu di ciascuna macchina.

```
int[servers] disks;
int[servers] cpuPowerProfiles;
```

Vengono inoltre dichiarati gli array con i valori del consumo energetico di ciascuna component hardware per ciascun server.

```
float[servers][cpuPowerProfiles] cpuPower;
float[servers] ramPower;
float[servers] controllerPower;
float[servers][disk] diskWritePower;
float[servers][disk] diskReadPower;
float[servers][disk] diskIdlePower;
float[servers][disk] diskStbPower;
float[servers][disk] diskSleepPower;
```

Ed infine i vettori contenenti le annotazioni di ciascun servizio.

```
float[services] serviceExecTime;
float[services] serviceHDDReadTime;
float[services] serviceHDDWriteTime;
float[services] serviceHDDIdleTime;
float[services] serviceHDDStbTime;
float[services] serviceHDDSleepTime;
float[services][servers][cpuPowerProfiles]
        serviceExecuteTimeAtPowerProfile;
```

Metodo per il Calcolo dell'energia

Successivamente viene implementato il metodo per il calcolo della funzione costo utilizzando l'equazione (2.10): dati in input l'identificativo del server e l'identificativo del servizio viene calcolata l'energia consumata dal servizio in esecuzione su quello specifico server.

```
public float calcEnergy(int server, int service){
        float energy = 0;

        for(int i = 0; i < cpuPowerProfiles[server]; i++){
         energy += cpuPower[server][i] *
         * serviceExecuteTimeAtPowerProfile[service][server][i];
        }

        energy += ramPower[server] *
                *serviceExecTime[service];
```

```
        energy += controllerPower[server] *
                * serviceExecTime[service];

        for(int i=0; i < disks[server]; i++){
                energy += diskWritePower[server][i] *
                        * serviceHDDWriteTime[service];
                energy += diskReadPower[server][i] *
                        * serviceHDDReadTime[service];
                energy += diskIdlePower[server][i] *
                        * serviceHDDIdleTime[service];
                energy += diskStbPower[server][i] *
                        * serviceHDDStbTime[service];
                energy += diskSleepPower[server][i] *
                        * serviceHDDSleepTime[service];
        }

        return energy;
}
```

Implementazione del modello di allocazione

L'implementazione del modello di allocazione avviene in forma matriciale come richiesto dalla sintassi di lp_solve.
Si dovrà anzitutto definire un nuovo problema di programmazione lineare con il comando:

```
lp = LpSolve.makeLp(0, nVar);
```

Dove 0 indica il numero di vincoli da definire al momento dell'allocazione del modello e nVar il numero di variabili del modello.
Istanziato il nuovo modello rappresentato dalla variabile `lp` è possibile definire i vincoli e la funzione obiettivo.
La possibilità di aggiungere dinamicamente vincoli al problema fa sì che sia possibile dare la possibilità al designer di decidere quali vincoli considerare durante l'allocazione dei servizi del proprio processo di business oppure di aggiungere ulteriori vincoli non previsti.
I nuovi vincoli vengono aggiunti grazie al metodo `addConstraintex` mentre la funzione obiettivo viene definita con `setObjFnex`.

Ad esempio il vincolo (2.15) viene realizzato da:

```
colCounter = 0;
//Le celle degli array vengono impostate a zero
row = setZeroDoubleArray(row);
```

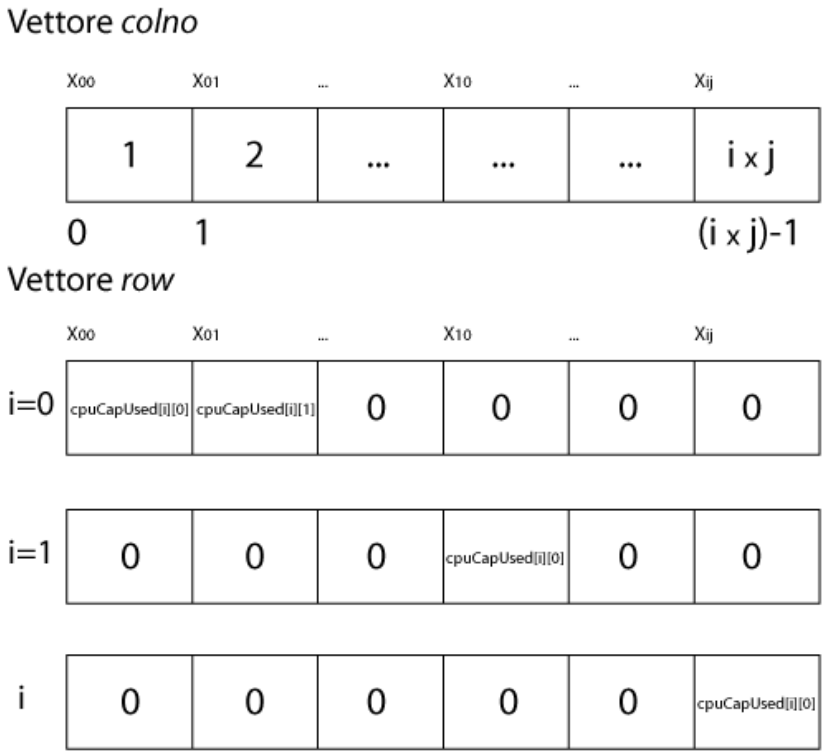


Figura 4.3: Struttura dei vettori colno e row

```
colno = setZeroIntArray(colno);

//Ciclo di creazione del vincolo
for(int i=0; i<nServers;i++){
        for(int j=0;j<nServices;j++){
                colno[colCounter] = colCounter + 1;
                row[colCounter] = cpuCapUsed[i][j];
                colCounter++;
        }
        /*Per ciascun server viene creato
          il vincolo sulla capacità*/
        lp.addConstraintex(colCounter, row, colno,
        LpSolve.LE, cpuCap[i]);

        row = setZeroDoubleArray(row);
}
```

La variabile `colCounter` rappresenta l'indice del vettore `colno[]` rappresentante le colonne della matrice del modello ovvero le variabili (nel modello vi sono i x j variabili $x_{i,j}$ che indicano che il servizio j è in esecuzione sulla virtual machine i). L'array `row[]` contiene invece i valori assunti dalle singole variabili per tale vincolo.
Con l'invocazione di `addConstraintex` viene aggiunto il vincolo avente calCounter variabili, rappresentato dalla matrice identificata da `row` e `colno`. Il vincolo deve essere minore o uguale (`LpSolve.LE`) al valore `cpuCap[i]`. Per capire meglio la struttura della matrice di lp_solve e in particolare del vincolo sopra riportato si faccia riferimento alla figura 4.3. Come si è visto impostare un modello di programmazione lineare con lp_solve non è a prima vista semplicissimo ma senza dubbio è possibile rendere la composizione del modello altamente dinamica.

Capitolo 5

Conclusioni e sviluppi futuri

L'obiettivo di questa tesi è dimostrare come sia possibile realizzare uno strumento che permetta di valutare l'efficienza energetica a design-time.
In particolar modo l'attenzione si è soffermata su come sia possibile realizzare un modello che sia allo stesso tempo abbastanza semplice da poter essere utilizzato a design-time che potente al punto di stimare con la maggior accuratezza possibile il consumo energetico di un processo.
E' stato proposto quindi un possibile modello che permetta al designer di valutare quanto efficiente è un processo di business e che possa essere implementato in un modello di allocazione al fine di permettere a design-time di studiare una disposizione ottima dei servizi che compongono il processo in modo da consumare meno energia possibile.
Per quanto riguarda l'allocazione dei processi è stato sviluppato un modello di programmazione lineare in cui i servizi vengono disposti sulla base di una serie di vincoli avendo come obiettivo la minimizzazione dell'energia assorbita dal server. Questo modello però è il più semplice possibile in quanto vuole essere solo un esempio di una possibile applicazione del modello di stima energetica proposto in questa tesi.
Un primo sviluppo futuro di questa tesi può essere quindi l'inserimento del modello energetico qui proposto all'interno di un più complesso e articolato modello di allocazione dei servizi in modo da migliorare ulteriormente l'efficienza energetica disponendo al meglio i singoli servizi sulle macchine disponibili.

Considerando che il modello per la stima dell'impatto energetico di un processo è stato progettato pensando a un singolo processo e non ad eventuali altri processi in esecuzione, in seguito il modello potrà essere esteso verificando dapprima se e come viene stimata l'energia di un processo in esecuzione insieme ad altri processi, identificandone le criticità e cercando di risolverle.

La validazione del modello è stata fatta utilizzando i server a disposizione del progetto GAMES e ha prodotto ottimi risultati mostrando come il modello proposto fornisca una stima molto vicina al valore reale misurato con appositi sensori.
Si deve tenere presente che sia la rilevazione dei dati che il calcolo del valore stimato utilzzando le equazioni energetiche proposte può essere migliorato: le misurazioni aggiungendo sensori sui singoli componenti così da poter validare le equazioni dei singoli componenti del server, i calcoli avendo informazioni più dettagliate sui singoli hardware installati nel server.

Il modello è stato anche integrato in un prototipo di applicazione web per il design dei processi di business e il contestuale calcolo dell'energia dissipata. L'applicazione sviluppata rappresenta solo un punto di partenza e può essere successivamente ampliata implementando nuove funzionalità e adattandola alle specifiche esigenze del singolo designer.

In conclusione il modello proposto risulta essere un primo approccio per la valutazione a design-time dell'energia richiesta da un processo, le soluzioni esposte in questa tesi possono comunque risultare un buon punto di partenza per future ricerche volte a migliorare la stima dell'efficienza energetica dei processi da parte del designer.

Appendice A

Specifiche hardware

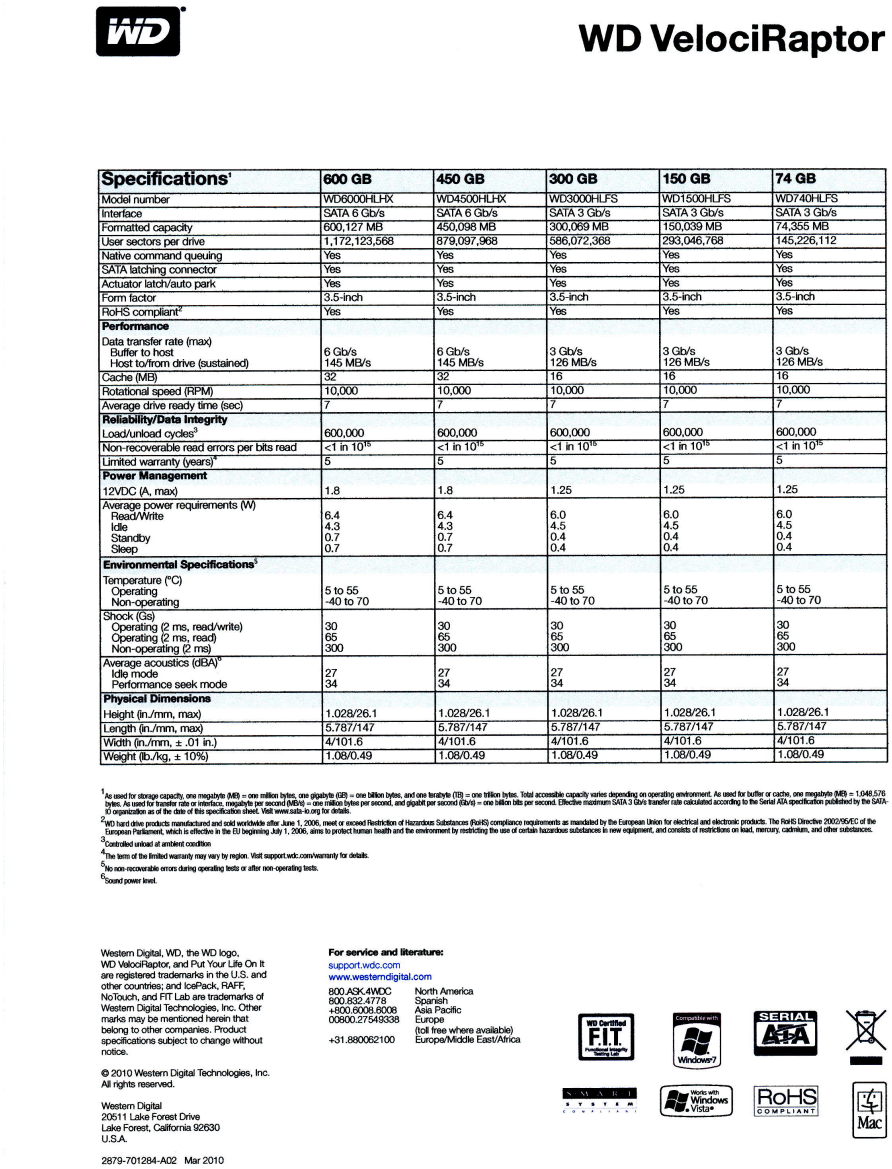
In questa appendice vengono riportati gli estratti di alcuni datasheet relativi a componenti installate sul GAMES front-end server e sui nodi del cluster.

1. Each processor is programmed with a maximum valid voltage identification value (VID), which is set at manufacturing and cannot be altered. Individual maximum VID values are calibrated during manufacturing such that two processors at the same frequency may have different settings within the VID range. Note that this differs from the VID employed by the processor during a power management event (Intel Thermal Monitor 2, Enhanced Intel SpeedStep Technology, or Enhanced Halt State).
2. The voltage specifications are assumed to be measured across V_{CC_SENSE} and V_{SS_SENSE} pins at socket with a 100-MHz bandwidth oscilloscope, 1.5-pF maximum probe capacitance, and 1-MΩ minimum impedance. The maximum length of ground wire on the probe should be less than 5 mm. Ensure external noise from the system is not coupled in the scope probe.
3. Specified at 105 °C T_J.
4. Specified at the nominal V_{CC}.
5. Measured at the bulk capacitors on the motherboard.
6. $V_{CC,BOOT}$ tolerance shown in Figure 7 and Figure 8.
7. Based on simulations and averaged over the duration of any change in current. Specified by design/ characterization at nominal V_{CC}. Not 100% tested.
8. This is a power-up peak current specification that is applicable when V_{CCP} is high and V_{CC_CORE} is low.
9. This is a steady-state I_{CC}current specification that is applicable when both V_{CCP} and V_{CC_CORE} are high.
10. Processor I_{CC} requirements in Intel Dynamic Acceleration Technology mode are lesser than I_{CC} in HFM
11. The maximum delta between Intel Enhanced Deeper Sleep and LFM on the processor will be lesser than or equal to 300 mV.
12. Instantaneous current $I_{CC\_CORE\_INST}$ of 57 A has to be sustained for short time (t_{INST}) of 35 µs. Average current will be less than maximum specified I_{CCDES}. VR OCP threshold should be high enough to support current levels described herein.

Table 8. Voltage and Current Specifications for the Dual-Core, Low-Power Standard-Voltage Processors (25 W) in Standard Package

Symbol	Parameter		Min	Typ	Max	Unit	Notes
V_{CCDAM}	V_{CC} in Enhanced Intel® Dynamic Acceleration Technology Mode		0.9		1.3	V	1, 2
V_{CCHFM}	V_{CC} at Highest Frequency Mode (HFM)		0.9		1.25	V	1, 2
V_{CCLFM}	V_{CC} at Lowest Frequency Mode (LFM)		0.85	—	1.025	V	1, 2
V_{CCSLFM}	V_{CC} at Super Low Frequency Mode (Super LFM)		0.75	—	0.95	V	1, 2
$V_{CC,BOOT}$	Default V_{CC} Voltage for Initial Power Up		—	1.2	—	V	2, 6
V_{CCP}	AGTL+ Termination Voltage		1.0	1.05	1.1	V	
V_{CCA}	PLL Supply Voltage		1.425	1.5	1.575	V	
$V_{CCDPRSLP}$	V_{CC} at Deeper Sleep		0.65	—	0.85	V	1, 2
V_{DC4}	V_{CC} at Intel® Enhanced Deeper Sleep State		0.6	—	0.85	V	1, 2
$V_{CCDPPWDN}$	V_{CC} at Deep Power Down Technology State (C6)		0.35	—	0.7	V	1, 2
I_{CCDES}	I_{CC} for Processors Recommended Design Target		—	—	38	A	12
I_{CC}	I_{CC} for Processors		—	—	—		
	Processor Number	Core Frequency/Voltage	—	—	—		
	P9700 P9600 P8800 P9500 P8700 P8600 P8400	2.8 GHz & V_{CCHFM} 2.667 GHz & V_{CCHFM} 2.667 GHz & V_{CCHFM} 2.53 GHz & V_{CCHFM} 2.53 GHz & V_{CCHFM} 2.4 GHz & V_{CCHFM} 2.267 GHz & V_{CCHFM} 1.6 GHz & V_{CCLFM} 0.8 GHz & V_{CCSLFM}	—	—	38 38 38 38 38 38 38 27.7 17.5	A	3, 4, 10

Figura A.1: Estratto del datasheet della CPU Intel P8400



WD VelociRaptor

Specifications[1]	600 GB	450 GB	300 GB	150 GB	74 GB
Model number	WD6000HLHX	WD4500HLHX	WD3000HLFS	WD1500HLFS	WD740HLFS
Interface	SATA 6 Gb/s	SATA 6 Gb/s	SATA 3 Gb/s	SATA 3 Gb/s	SATA 3 Gb/s
Formatted capacity	600,127 MB	450,098 MB	300,069 MB	150,039 MB	74,355 MB
User sectors per drive	1,172,123,568	879,097,968	586,072,368	293,046,768	145,226,112
Native command queuing	Yes	Yes	Yes	Yes	Yes
SATA latching connector	Yes	Yes	Yes	Yes	Yes
Actuator latch/auto park	Yes	Yes	Yes	Yes	Yes
Form factor	3.5-inch	3.5-inch	3.5-inch	3.5-inch	3.5-inch
RoHS compliant[2]	Yes	Yes	Yes	Yes	Yes
Performance					
Data transfer rate (max) Buffer to host Host to/from drive (sustained)	 6 Gb/s 145 MB/s	 6 Gb/s 145 MB/s	 3 Gb/s 126 MB/s	 3 Gb/s 126 MB/s	 3 Gb/s 126 MB/s
Cache (MB)	32	32	16	16	16
Rotational speed (RPM)	10,000	10,000	10,000	10,000	10,000
Average drive ready time (sec)	7	7	7	7	7
Reliability/Data Integrity					
Load/unload cycles[3]	600,000	600,000	600,000	600,000	600,000
Non-recoverable read errors per bits read	<1 in 10^{15}	<1 in 10^{15}	<1 in 10^{15}	<1 in 10^{15}	<1 in 10^{15}
Limited warranty (years)[4]	5	5	5	5	5
Power Management					
12VDC (A, max)	1.8	1.8	1.25	1.25	1.25
Average power requirements (W) Read/Write Idle Standby Sleep	 6.4 4.3 0.7 0.7	 6.4 4.3 0.7 0.7	 6.0 4.5 0.4 0.4	 6.0 4.5 0.4 0.4	 6.0 4.5 0.4 0.4
Environmental Specifications[5]					
Temperature (°C) Operating Non-operating	 5 to 55 -40 to 70	 5 to 55 -40 to 70	 5 to 55 -40 to 70	 5 to 55 -40 to 70	 5 to 55 -40 to 70
Shock (Gs) Operating (2 ms, read/write) Operating (2 ms, read) Non-operating (2 ms)	 30 65 300	 30 65 300	 30 65 300	 30 65 300	 30 65 300
Average acoustics (dBA)[6] Idle mode Performance seek mode	 27 34	 27 34	 27 34	 27 34	 27 34
Physical Dimensions					
Height (in./mm, max)	1.028/26.1	1.028/26.1	1.028/26.1	1.028/26.1	1.028/26.1
Length (in./mm, max)	5.787/147	5.787/147	5.787/147	5.787/147	5.787/147
Width (in./mm, ± .01 in.)	4/101.6	4/101.6	4/101.6	4/101.6	4/101.6
Weight (lb./kg, ± 10%)	1.08/0.49	1.08/0.49	1.08/0.49	1.08/0.49	1.08/0.49

[1] As used for storage capacity, one megabyte (MB) = one million bytes, one gigabyte (GB) = one billion bytes, and one terabyte (TB) = one trillion bytes. Total accessible capacity varies depending on operating environment. As used for buffer or cache, one megabyte (MB) = 1,048,576 bytes. As used for transfer rate or interface, megabyte per second (MB/s) = one million bytes per second, and gigabit per second (Gb/s) = one billion bits per second. Effective maximum SATA 3 Gb/s transfer rate calculated according to the Serial ATA specification published by the SATA-IO organization as of the date of this specification sheet. Visit www.sata-io.org for details.

[2] WD hard drive products manufactured and sold worldwide after June 1, 2006, meet or exceed Restriction of Hazardous Substances (RoHS) compliance requirements as mandated by the European Union for electrical and electronic products. The RoHS Directive 2002/95/EC of the European Parliament, which is effective in the EU beginning July 1, 2006, aims to protect human health and the environment by restricting the use of certain hazardous substances in new equipment, and consists of restrictions on lead, mercury, cadmium, and other substances.

[3] Controlled unload at ambient condition

[4] The term of the limited warranty may vary by region. Visit support.wdc.com/warranty for details.

[5] No non-recoverable errors during operating tests or after non-operating tests.

[6] Sound power level.

Western Digital
20511 Lake Forest Drive
Lake Forest, California 92630
U.S.A.

2879-701284-A02 Mar 2010

For service and literature:
support.wdc.com
www.westerndigital.com

800.ASK.4WDC	North America
800.832.4778	Spanish
+800.9006.8008	Asia Pacific
00800.27549338	Europe
	(toll free where available)
+31.880062100	Europe/Middle East/Africa

Figura A.2: Estratto del datasheet dell'Hard Disk dello storage server

Memory Module Specifications

Kingston TECHNOLOGY ValueRAM

KVR1066D3E7/2G

2GB 2Rx8 256M x 72-Bit PC3-8500
CL7 ECC 240-Pin DIMM

Important Information: The module defined in this data sheet is one of several configurations available under this part number. While all configurations are compatible, the DRAM combination and/or the module height may vary from what is described here.

DESCRIPTION

This document describes ValueRAM's 256M x 72-bit (2GB) DDR3-1066 CL7 SDRAM (Synchronous DRAM), 2Rx8 ECC memory module, based on eighteen 128M x 8-bit FBGA components. The SPD is programmed to JEDEC standard latency DDR3-1066 timing of 7-7-7 at 1.5V. This 240-pin DIMM uses gold contact fingers. The electrical and mechanical specifications are as follows:

FEATURES

- JEDEC standard 1.5V (1.425V ~1.575V) Power Supply
- VDDQ = 1.5V (1.425V ~ 1.575V)
- 533MHz fCK for 1066Mb/sec/pin
- 8 independent internal bank
- Programmable CAS Latency: 8, 7, 6
- Programmable Additive Latency: 0, CL - 2, or CL - 1 clock
- Programmable CAS Write Latency(CWL) = 6 (DDR3-1066)
- 8-bit pre-fetch
- Burst Length: 8 (Interleave without any limit, sequential with starting address “000” only), 4 with tCCD = 4 which does not allow seamless read or write [either on the fly using A12 or MRS]
- Bi-directional Differential Data Strobe
- Internal(self) calibration: Internal self calibration through ZQ pin (RZQ : 240 ohm ± 1%)
- On Die Termination using ODT pin
- Average Refresh Period 7.8us at lower than TCASE 85°C, 3.9us at 85°C < TCASE ≤ 95°C
- Asynchronous Reset
- PCB: Height 1.18” (30mm), double sided component

SPECIFICATIONS

CL(IDD)	7 cycles
Row Cycle Time (tRCmin)	50.63ns (min.)
Refresh to Active/Refresh Command Time (tRFCmin)	110ns (min.)
Row Active Time (tRASmin)	37.5ns (min.)
Power (Operating)	1.890 W*
UL Rating	94 V - 0
Operating Temperature	0° C to 85° C
Storage Temperature	-55° C to +100° C

*Power will vary depending on the SDRAM used.

Continued >>

Figura A.3: Datasheet RAM

Appendice B

Struttura database FCIM

In questa appendice viene riportata la struttura integrale del database contente le informazioni raccolte dai sensori installati sui server del progetto GAMES presso HLRS.

Tables	Columns	Type	Sample value	Unit
fcim_timestamp				
It stores the timestamp of the Nagios plugin checks (i.e. the measurements).	id	int	1234	.
	timestamp	timestamp	2010-05-22 20:45:00	.
fcim_temperature				
It monitors the temperature of the CPU or of the chipset/mainboard.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	status	string	OK	.
	component	string	cpu1	.
	temperature	int	58	°C
fcim_cpu_fan				
It monitors the speed of a CPU fan.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	status	string	OK	.
	component	string	fan1	.
	speed	int	2486	rpm
fcim_cpu_ssh				
It monitors the % of CPU utilization that occurred while executing at the application level (user_perc), in particular with nice priority (nice_perc), or at the kernel level (sys_perc). It monitors also the % of time that the CPU(s) were idle during which the system had an outstanding disk I/O request (iowait_perc) or when the system did not have any outstanding request (idle_perc). The hardware (irq_perc) and software IRQs (softirq_perc) are reported, too. Data collected via SSH.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	status	string	OK	.
	user_perc	float	0.5	%
	nice_perc	float	0.5	%
	sys_perc	float	0.99	%
	iowait_perc	float	0.5	%
	idle_perc	float	100	%
	irq_perc	float	0.5	%
	softirq_perc	float	0.5	%
fcim_cpu_one				
It monitors the CPU utilization getting info from OpenNebula about the total CPU (number of cores of the CPU multiplied with 100), the CPU allocated for the VMs and the remaining free one, expressed in % (100 means 1 core free, 150 means 1 core free and one with 50% free). The used CPU is collected as weel.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	allocated	int	69	#cores * %
	total	int	400	#cores * %
	free	int	399	#cores * %
	used	int	0	#cores * %
fcim_cpu_load				
It monitors the average number of processes that are using or waiting for CPU, occurred in the last period (in minutes), expressed as Unix load. Common or equivalent data in this table, fcim_cpu_ssh and fcim_cpu_one should be defined and moved into fcim_cpu to avoid redundances.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	status	string	OK	.
	average	float	0.06	unix load
	period	int	15	min

Figura B.1: Struttura database fcim 1/5

Tables	Columns	Type	Sample value	Unit
fcim_cpu				
It stores information about the CPU percentage reserved for the VMs (allocation_perc), the amount of CPU used from the total CPU (usage_perc). Values calculated from data collected via OpenNebula. They could be collected via SSH too, but percentages must be referred to CPU allocated for the VM: is it the case?	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	allocation_perc	float	17,25	%
	usage_perc	float	0	%
fcim_cpu_frequency				
It stores information about the CPU frequency. Data collected via OpenNebula.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	frequency	float	3101,08	MHz
	host???	string	192.168.1.97	.
fcim_kernel				
It stores information about the kernel performance in terms of FLOPS.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	performance	float		FLOPS
fcim_task_manager				
It monitors the total number of processes currently opened.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	status	string	OK	.
	nr_processes	int	131	.
fcim_process				
It monitors the activities of a process providing its name (process), its propertary (user), the percentage of cpu (cpu_usage_perc) and memory (ram_usage_perc) used by the process, its start time (start_time) and the amount of time it used the CPU (cpu_time). It is used for simulation processes.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	status	string	OK	.
	process	string	fmps_default_static	.
	user	string	hpcralf	.
	cpu_usage_perc	float	39	%
	ram_usage_perc	float	3.4	%
	start_time	string	14:45	.
	cpu_time	int	87	min
fcim_recs_controller				
It monitors the RECS microcontroller in terms of number of mainboards checked (boards_count) and its central voltage (central_voltage).	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	status	string	OK	.
	nr_boards	int	18	.
	central_voltage	float	11.84	V
fcim_recs_board				
It monitors the polling data from RECS microcontroller about single mainboards, idendified by boards_id, in terms of status, temperature and power consumption.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	board_id	int	19	.
	status	string	OK	.
	temperature	int	28	°C
	power	int	24	W

Figura B.2: Struttura database fcim 2/5

Tables	Columns	Type	Sample value	Unit
fcim_memory_ssh				
It monitors the cashed and uncashed memory occupation of the entire node. Data collected via SSH.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	status	string	OK	.
	cached	int	1260	MB
	uncached	int	2349	MB
fcim_memory_one				
It monitors the amount of memory reserved (allocated) for the current server's virtual machines. Data collected via OpenNebula.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	allocated	int	1048576	Bytes
	allocation_perc	float	26,7835504469987	%
fcim_memory				
It monitors the amount of memory available (total), currently used by the server (used) and the remaining part (free). It stores also the memory percentage reserved (allocation_perc) and currently used (usage_perc) from the total memory. Data collected via OpenNebula or via SSH.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	total	float	3915	MB
	used	float	65	MB
	free	float	3849	MB
	usage_perc	float	1,6602809706258	%
fcim_hdd_one				
It monitors the disk utilization getting info from OpenNebula about the amount of total storage available on the server's hdd, the amount of storage allocated, the remaining free amount, the amount of storage used by the server and the usage of storage expressed in percentage from the total storage. Data collected via OpenNebula.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	allocated	int	0	MB
	free	int	0	MB
	total	int	0	MB
	used	int	0	MB
	usage_perc	float	0	%
fcim_server_ping				
It monitors the other information resulting from the ping to the server, in terms of percentage of packets lost (loss) and rta.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	status	string	ON	.
	packet_loss	float	0	%
	rta	float	0.76	ms
fcim_server_consumption				
It monitors the power consumption of a server, and its critical and warning thresholds.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	server	string	internal	.
	power	float	285,5	W
	warning_threshold	float	300	W
	critical_threshold	float	500	W

Figura B.3: Struttura database fcim 3/5

Tables	Columns	Type	Sample value	Unit
fcim_vms_one				
It monitors the number of VMs running on the node (nr_running_vms) identified by the OpenNebula id assigned to it (node_id) and its IP address (ip). Data collected via OpenNebula.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	ip	string	192.168.1.25	.
	node_id	int	2	.
	nr_running_vms	int	1	.
fcim_virtualmachine_one				
It monitors information about the VM identified by the OpenNebula id vm_id, running on a server, getting info about its status(ON/OFF),its current memory usage, the total time it used the CPU from its creation (cpu_time) and the percentage of CPU used from the total available on the host (cpu_usage_perc). Data collected by OpenNebula Libvirt.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	vm_id	int	1	.
	status	string	ON	.
	memory	float	1048,576	MB
	cpu_time	float	9620000000	ns
	cpu_usage_perc	float	10,24	%
fcim_vm_consumption				
It monitors the estimated power consumption of a specific VM (easy to measure if it's the only VM on one node, hard to estimate if there are more VMs on one node). Neither dump or documentation available about data strusture.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	power	float	30	W
fcim_storage				
It monitors storage performance and power	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	status	string	OK	.
	storage_type	string	device	.
	space_allocated	int		MB
	space_free	int		MB
	space_total	int		MB
	space_used_perc	float		%
	average_measured_power	float		W
	averaging_interval	int		ms
	iops_read	float		IOPS
	iops_write	float		IOPS
	iops_total	float		IOPS
	sequential_portion_perc	float		%
	responsetime_measured	int		ms
	queue_length	float		.
	throughput_read	float		MB/s
	throughput_write	float		MB/s
	throughput_total	float		MB/s
	allocation_status	boolean		.
	storage_mode	int	1	.
fcim_env_humidity				
It monitors the environmental humitity and its status, as reported by the plugin.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	percentage	float	24.1	%
	status	string	OK	.

Figura B.4: Struttura database fcim 4/5

Tables	Columns	Type	Sample value	Unit
fcim_env_consumption				
It monitors the enviromnental consumption in terms of current, energy, kva, kwatts, voltage and their status, as reported by the plugin.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	current	float	0.4	A
	current_status	string	CRITICAL	.
	energy	int	0	J
	energy_status	string	OK	.
	kva	int	1	kVA
	kva_status	string	OK	.
	kwatts	int	0	kW
	kwatts_status	string	OK	.
	voltage	int	230	V
	voltage_status	string	OK	.
fcim_env_temperature				
It monitors the environmental temperature and its status, as reported by the plugin.	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	temperature	float	26.4	°C
	status	string	OK	.
fcim_env_air				
It monitors the actual temperature of the intake air (intake_temperature) and of the oulet air (outlet_temperature). It's not clear how to retrieve this data from CLM5	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	intake_temperature	float	60	°C
	outlet_temperature	float	40	°C
fcim_network				
It monitors the network power consumption. It's not clear how to retrieve this data from CLM5	id	int	567	.
	timestamp_id	int	1234	.
	service_id	string	19	.
	power	int	22	W

Figura B.5: Struttura database fcim 5/5

Bibliografia

[1] A. Kansal, F. Zhao: *Fine-Grained Energy Profiling for Power-Aware Application Design*. ACM SIGMETRICS Performance Evaluation Review 36(2) (2008)

[2] A. Jaiantilal, Y. Jiang, S. Mishra: *Modeling CPU Energy Consumption for Energy Efficient Scheduling*. Proceedings of the 1st Workshop on Green Computing (2010)

[3] S. Gurumurthi, A. Sivasubramaniam, M. J. Irwin, N. Vijaykrishnan, M. Kandemir, T. Li, L. K. John: *Using Complete Machine Simulation for Software Power Estimation: The SoftWatt Approach*. Proceedings of the eighth International Symposium on High-Performance Computer Architecture (2002)

[4] A. Kansal, F. Zhao, J. Liu, N. Kothari, A. A. Bhattacharya: *Virtual Machine Power Metering and Provisioning*. Proceedings of the 1st ACM symposium on Cloud computing (2010)

[5] T. Do, S. Rawshdeh, W. Shi: *pTop: A Process-level Power Profiling Tool*. (2009)

[6] J. G. Koomey: *Growth in data center electricity use 2005 to 2010*. Analytics Press (2011)

[7] M. Wendt, M. Grumer, C. Steger, R. Weiss, U. Neffe and A. Muehlberger: *Tool for Automated Instruction Set Characterization for Software Power Estimation*. IEEE transactions on instrumentation and measurement (2010)

[8] U.S. Environmental Protection Agency: *Report to Congress on Server and Data Center Energy Efficiency - Public Law 109-431* (2007)

[9] C. Cappiello, M.G. Fugini, G.R. Gangadharan, A. Mello Ferreira, B. Pernici, P. Plebani: *Toward Energy-aware Adaptive Business Processes*

[10] Vaadin: *Vaadin web site*. http://vaadin.com

[11] M. Monti: *Node Graph Widget*. http://code.google.com/p/nodegraph-widget

[12] Michel Berkelaar: *lp_solve 5.5*. http://lpsolve.sourceforge.net/5.5

[13] G. Shvets: *CPU World*. http://www.cpu-world.com

[14] Kingston Technology: *ValueRAM Datasheets*. http://www.valueram.com/datasheets/default.asp

[15] Games project: *GAMES*. http://www.green-datacenters.eu

[16] Intel: *Intel web site and datasheets*. http://intel.com

[17] Western Digital: *Western Digital web site and datasheets*. http://www.wdc.com