UNIVERSITÁ DEL SALENTO

FACOLTÁ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA DELL'INFORMAZIONE

Teodoro Montanaro
Matricola n. 10041267

TECNICHE DI BENCHMARKING APPLICATE AI MODELLI CLIMATICI

TESI DI LAUREA IN
SISTEMI DI ELABORAZIONE

Tesista:

Teodoro Montanaro

Relatore:

Prof. Italo Epicoco

ANNO ACCADEMICO 2009/2010

INDICE

1 Introduzione

Il presente lavoro di tesi si è focalizzato su tutti gli aspetti relativi al benchmarking di applicazioni parallele applicate al campo della climatologia.

Un benchmark viene definito come un insieme di programmi (o micro programmi) eseguiti su sistemi diversi per fornire una misura delle prestazioni. I risultati ottenuti dai benchmark standard, perciò, riflettono solo le performance del set di programmi presenti sul sistema che si sta testando. In realtà, c'è spesso una differenza tra i programmi presenti nei benchmark standard e le applicazioni reali, a causa della quale i risultati dei benchmark possono essere in alcuni casi disinformativi, o addirittura fuorvianti.

Si possono individuare due cause di questo divario:

In primo luogo, l'insieme dei programmi inclusi in un benchmark standard è fisso e può accadere che i programmi inclusi non siano rappresentativi delle applicazioni reali alle quali sono interessati gli utilizzatori finali.
Inoltre il problema può essere aggravato dalla sovra-ottimizzazione introdotta dai costruttori: quanto più i risultati dei benchmark diventano un fattore importante nella decisione di acquisto, tanto più i venditori si preoccupano di progettare i loro sistemi intorno ai benchmark più popolari. Ciò che accade di frequente è che alcuni fornitori si preoccupano di ottimizzare i loro sistemi per alcuni benchmark allo scopo di guadagnare pochi punti percentuali sul punteggio di riferimento, tutto ciò a discapito delle prestazioni delle applicazioni reali. Tali problemi si sono resi evidenti nei benchmark di grafica, dove i venditori di schede grafiche hanno impiegato una tecnica di "miglioramento" per migliorare i loro risultati nel benchmark WinBench Graphics WinMark, compromettendo gravemente le prestazioni di altri dispositivi in situazioni realistiche. Il fatto che l'insieme dei programmi inclusi nel benchmark standard sia generalmente fisso, agevola questa pratica nel corso di ottimizzazione, rendendo ancora più difficile giudicare il rendimento reale di un sistema in presenza di una applicazione reale.

In secondo luogo, la tendenza ad una crescita esponenziale di velocità in hardware e di complessità in software, ha portato lo sviluppo di benchmark a rimanere indietro con i tempi a causa di motivi sia economici che tecnici. Il costo da sostenere per produrre benchmark è stato in costante aumento a causa della crescente complessità delle applicazioni. Come risultato i cicli di sviluppo dei software di benchmark si sono notevolmente allungati.
D'altro canto, la feroce concorrenza ha spinto le aziende a implementare nuove versioni delle applicazioni ad una velocità notevole. Il risultato è implicito: i benchmark standard sono spesso perdenti nella corsa tecnologica e rappresentano i carichi di lavoro di "ieri".

Il benchmark SPEC WEB99 è un buon esempio. SPEC WEB99 è un benchmark standard per misurare le prestazioni dei server web ed è approvato dai principali fornitori di web-server. La prima versione di SPEC WEB (SPEC WEB96) è stata introdotta nel 1996, tre anni dopo il decollo del world-wide-web, e conteneva solo i file statici nel carico di lavoro. La seconda versione, SPEC WEB99, è stata introdotta nel 1999 includendo la generazione dinamica dei contenuti. Tuttavia tale aggiornamento è avvenuto nuovamente due anni dopo l'uso popolare di pagine web dinamiche. Questo squilibrio tra lo sviluppo di applicazioni e lo sviluppo di benchmark porta inevitabilmente alla creazione di un divario tra i benchmarks disponibili e le applicazioni allo "stato dell'arte".

1.1 Obiettivi generali

Lo scopo di questo lavoro di tesi è quello di fornire supporto ad enti e comunità di ricerca sul clima che, tramite l'applicazione realizzata, potranno usufruire di uno strumento facilmente portabile per eseguire benchmark di architetture parallele utilizzando modelli numerici per la descrizione di fenomeni climatologici.

Gli obiettivi perseguiti nell'arco dell'intero progetto possono essere schematizzati nei seguenti punti:

- Permettere una maggiore collaborazione tra iniziative di supporto ai modelli;

- Valutare le performance dei vari modelli climatologici esistenti su architetture parallele;
- Effettuare un benchmarking delle architetture parallele e definire gli aspetti critici che i modelli climatologici stressano;
- Definire e ottimizzare gli attuali ESMs (modelli che descrivono il sistema terra), andando alla ricerca di portabilità dei modelli chiave in modo da poter essere utilizzati su un ampio numero di piattaforme;
- Evidenziare i colli di bottiglia, i punti di forza e di debolezza di ciascuno dei modelli per guidare la progettazione e lo sviluppo di futuri ESMs ottimizzati per le imminenti peta ed exascale architecture.
- Permettere di sviluppare strumenti per facilitare la composizione di nuovi ESMs (Earth System Models) partendo da quelli già esistenti e accoppiare tecnologie allo scopo di abbassare le barriere tecniche che attualmente incontrano le piccole organizzazioni di ricerca sul clima.

Il presente lavoro di tesi risulta inoltre inquadrato all'interno di un progetto di ricerca denominato **IS-ENES** finanziato dalla Comunità Europea sul FP7.

Le problematiche affrontate sono state organizzate come di seguito indicato.

Nei primi capitoli si evidenzia l'importanza che il benchmarking ha nei casi in cui si deve prendere una decisione riguardo le risorse da investire in un progetto, o il tempo da spendere per ottenere dei risultati dal software realizzato o comunque valutare le performance di un'architettura di cui si vuole fare utilizzo.

Successivamente, invece, vengono messi in risalto i vantaggi derivanti dall'utilizzo del software di benchmarking su architetture parallele.

Le difficoltà generalmente incontrate dagli utenti che cercano di eseguire tali software, ci hanno spinto a realizzare un'applicazione che possa aiutare chi provi a interfacciarsi al mondo del benchmarking. A tale scopo si sono inseriti: un capitolo riportante un esempio di configurazione del software realizzato e un'appendice

all'interno del quale si spiega in che modo è possibile selezionare i risultati da visualizzare attraverso la suite da noi creata.

1.2 Cos'è il benchmarking

Con il termine **benchmark** si intende un insieme di test software volti a fornire una misura delle prestazioni della particolare macchina sulla quale i test vengono effettuati.

Esistono 2 diverse fondamentali categorie di benchmark in termini di complessità e spazio utilizzato dai benchmark stessi:

- **Microbenchmarks (o benchmark sintetici)**
- **Macrobenchmarks (o benchmark applicativi)**

I microbenchmarks misurano le prestazioni delle operazioni primitive, sia hardware che software, supportate da una piattaforma di base. Inoltre, includono brevi sequenze di codice (kernel) che risolvono problemi piccoli e ben definiti.
Esempi di operazioni primitive sono: il tempo necessario per recuperare un dato dalla cache / memoria, o il tempo necessario per disegnare una linea su un terminale grafico. I microbenchmarks rivelano una grande quantità di informazioni sui costi fondamentali di un sistema e sono utili per confrontare le operazioni di basso livello dei diversi sistemi, ma è difficile metterle in relazione alle prestazioni delle applicazioni reali.

I macrobenchmarks, invece, sono costituiti da uno o più programmi generalmente derivati da applicazioni reali. Essi vengono normalmente forniti assieme ai dati di input che si presume possano essere rappresentativi di situazioni tipiche. Quando il benchmark è in esecuzione sono proprio i dati di input ad "alimentare" i vari programmi che lo compongono. Generalmente, i valori restituiti da tali benchmark sono rappresentativi del tempo totale di esecuzione. Visto che i macrobenchmarks sono in genere derivati da applicazioni reali, essi sottopongono la macchina ad una notevole quantità di stress esattamente come farebbero le applicazioni reali. Di conseguenza, a condizione che il carico normale del sistema sia lo stesso dei programmi inclusi nel

benchmark, il benchmark stesso può essere un buon indicatore della performance del sistema.

Come già detto, sia che si tratti di microbenchmarks, sia che si tratti di macrobenchmarks, il programma di test restituisce un indice prestazionale attraverso il quale è possibile confrontare le diverse architetture sulle quali è stato eseguito.

A tale scopo è utile, anzi è indispensabile, fornire insieme ai risultati ottenuti, tutti i dettagli relativi alla macchina e quindi all'architettura sulla quale il benchmark è stato eseguito; senza tali descrizioni e dettagli, i risultati sarebbero privi di significato.

I dettagli devono essere forniti in modo che qualsiasi utente possa rieseguire il test nelle stesse condizioni per riottenere pressappoco gli stessi risultati.

Per maggiori dettagli sul tipo di benchmark e sulla metrica utilizzata nella suite descritta in questo documento si veda il capitolo 2.

1.2.1 SPEC (Standard Performance Evaluation Corporation)

Esistono diversi istituti responsabili della definizione e distribuzione di benchmark standard. Alcuni tra i più conosciuti sono: SPEC (System Performance Evaluation Corporation) e TPC (Transaction Processing Performance Council).

SPEC definisce una vasta gamma di benchmark standard, che vanno dal calcolo ad alte prestazioni ai server di rete. Tra questi, la suite di benchmark SPEC CPU è probabilmente il benchmark più utilizzato. SPEC CPU è un macrobenchmark che è ulteriormente diviso in due macrobenchmarks, la "integer suite" e la "floating-point suite". La prima è composta da popolari applicazioni desktop UNIX come gzip (un programma di compressione GNU), gcc (un linguaggio compilatore GNU C) e Perl (un linguaggio di scripting creato da Larry Wall). La seconda invece è composta da applicazioni Floating point, tra cui applicazioni per la grafica 3D.

TPC è un'altra entità standard che si specializza in parametri di riferimento per l'elaborazione delle transazioni e dei sistemi di database. TPC ha prodotto benchmark ampiamente utilizzati come TPC-A, TPC-B, e i loro successori TPC-C e TPC-D.

TPC-C è un benchmark OLTP (OnLine Transaction Processing) che emula un sistema interattivo di elaborazione degli ordini.

Mentre TPC-D è un benchmark di supporto alle decisioni che contiene una serie di complesse query business-oriented (orientate al business).

La caratteristica che contraddistingue i benchmark TPC è che viene riportato il rapporto prezzo / prestazioni nei risultati di benchmark, un parametro utile che aiuta i clienti a bilanciare le prestazioni supplementari e i costi del sistema. Il rapporto è riportato in forma di dollari al TPM (transazioni al minuto). Le attuali versioni di benchmark TPC richiedono uno sforzo significativo per la configurazione e l'esecuzione.

La suite più vicina alle nostre esigenze, in quanto composta da applicazioni parallele, è la suite denominata HPC2002.

1.2.2 SPEC HPC2002

Le applicazioni di benchmark incluse all'interno della suite HPC2002 sono derivate da reali applicazioni HPC (High performance computing) e misurano le prestazioni complessive dei computer (CPU, memoria condivisa o distribuita, compilatori, librerie MPI e OPENMP, ecc) sui quali la suite viene eseguita.

I risultati ottenuti dalla suite HPC2002 rappresentano il numero di volte in cui la suite è stata eseguita nell'arco delle 24 ore.

Questa metrica permette di confrontare architetture diverse valutando tutti i parametri che influenzano le prestazioni.

Tale suite, con la sua metrica e le sue caratteristiche, è stata la base dalla quale siamo partiti per sviluppare la nostra suite di benchmark.

1.3 Il contesto di ricerca

L'attività di tesi si colloca all'interno delle attività di ricerca svolte dal **Centro Euro-Mediterraneo per i Cambiamenti Climatici.** Il CMCC è un consorzio di enti di ricerca a livello nazionale, con unità locali a Bologna, Venezia, Capua, Sassari, Milano e Benevento.

Il CMCC ha per oggetto la promozione e il coordinamento delle ricerche e delle diverse attività scientifiche e applicative nel campo dello studio dei cambiamenti climatici, favorendo anche collaborazioni tra Università, Enti di ricerca nazionali e internazionali, Enti territoriali e il Settore industriale.

Il CMCC e' organizzato in 5 divisioni di ricerca:

- Applicazioni Numeriche e Scenari;
- Impatti Economici e Politiche dei Cambiamenti Climatici;
- Formazione Documentazione e Divulgazione;
- Impatti sull'Agricoltura, Foreste ed Ecosistemi naturali terrestri;
- Impatti sul Suolo e sulle Coste;
- Calcolo Scientifico e Operazioni.

L'attività di ricerca della divisione **Calcolo Scientifico ed Operazioni (SCO) del Centro Euro-Mediterraneo** per i Cambiamenti Climatici comprende tre macro attività: **High End Computing, Gestione Avanzata di Dati, Gestione di Sistemi HPC**.

La missione della macro attività High End Computing (HEC) consiste nel supportare l'ambiente della simulazione scientifica da utente ad utente previsto dal piano strategico del CMCC. Seguendo questo scopo la macro attività HEC comprende: (I) l'ottimizzazione e la parallelizzazione di modelli disponibili e di adozione (sia di modelli climatici che di modelli che si riferiscono agli impatti climatici); (II) valutazione delle prestazioni e valutazione delle nuove tecnologie di calcolo emergenti per imparare ad ottenere da esse le migliori prestazioni; (III) aumentare gli strumenti per

la visualizzazione e l'analisi dei dati; (IV) creare strutture di software per facilitare il progresso scientifico attraverso collaborazioni interdisciplinari.

La macro attività Gestione Avanzata di Dati riguarda la gestione (classificazione, distribuzione, accesso ottimizzato, ecc.) di set di dati ed esperimenti climatici. E' stata scelta un'architettura/ soluzione di griglie di dati col fine di (I) facilitare la gestione di un ambiente di dati così distribuiti per i cambiamenti climatici; (II) installare un set di servizi di griglie di dati (un piano di griglie di dati) che sia uniforme (dal punto di vista della sicurezza); (III) fornire la base adeguata per un livello più alto ed attività ripartite; (IV) facilitare la gestione di diversi set di utenti attraverso il concetto di organizzazione virtuale; (V) creare un ambiente unificato, all'avanguardia e completo per gli studiosi del clima.

L'intero HPC, l'immagazzinamento dei dati e l'infrastruttura per l'utilizzazione della rete del CMCC sono gestiti dal gruppo "Gestione di Sistemi". Il compito principale di questo gruppo é quello di installare ed azionare l'infrastruttura di calcolo avanzato per attivare le attività di ricerca sul calcolo delle divisioni di ricerca del CMCC.

1.4 Progetto IS-ENES

Questo elaborato verrà inserito in un progetto più vasto quale è l'**IS-ENES**.

IS-ENES (**I**nfra**S**tructure for the **E**uropean **N**etwork for the **E**arth **S**ystem Modelling) è un progetto aderente all'FP7 (Seventh Framework Programme), programma che raggruppa tutti i progetti europei che giocano un ruolo fondamentale nel raggiungimento degli obiettivi di crescita, competitività e occupazione.
Esso è finanziato dalla Commissione Europea ed ha l'obiettivo di promuovere lo sviluppo di una infrastruttura distribuita comune di modellazione per la ricerca in Europa che agevoli lo sviluppo e la valorizzazione dei modelli climatici per meglio soddisfare le esigenze della società in relazione alle problematiche inerenti i cambiamenti climatici.

Il progetto IS-ENES, partito l'1 Marzo 2009 con termine previsto per il 28 Febbraio 2013, raggruppa 18 partner provenienti da 10 paesi europei e include i principali 6 modelli europei sul clima globale.

Il progetto è strutturato intorno a 3 principali attività:

Attività di rete (Networking Activities):

Una migliore integrazione si realizzerà attraverso una serie di attività di rete che si concentrerà sullo sviluppo della futura strategia Enes, lo scambio di esperienze e la creazione di attività di formazione.

Servizi sui modelli e sui dati (Service on models and data):

La IS-ENES e-infrastructure offrirà un servizio su modelli climatici e sosterrà la diffusione dei risultati.

Attività di ricerca congiunta (Joint Research Activities):

Le attività di ricerca congiunta miglioreranno l'uso efficiente dei computer ad alte prestazioni, degli strumenti di valutazione dei modelli, dell'accesso ai risultati del modello e dei prototipi relativi ai servizi climatici d'impatto per la comunità.

1.5 Stato dell'arte

Molti ricercatori hanno proposto diverse soluzioni al problema riscontrato con i benchmark a carico di lavoro fisso.

In un recente lavoro, Gustafson ha presentato un nuovo approccio per misurare le prestazioni della macchina che, invece di fissare la dimensione del lavoro dei programmi di benchmark, fissa il tempo e misura la quantità di lavoro durante il periodo di tempo fissato. Il programma di benchmark (chiamato HINT: Hierarchical INTegration) cerca di usare la suddivisione degli intervalli per trovare dei limiti razionali all'integrale della funzione $(1-x) / (1 + x)$ dove $x \in [0,1]$. La quantità di lavoro

svolto in ogni passo è misurata mediante il miglioramento della qualità del limite (cioè la distanza tra il limite superiore e inferiore). Poiché non esiste alcun limite superiore per la strettezza del limite (fatta salva la precisione), HINT è in grado di scalare tale intervallo misurando così la potenza del computer. Una differenza fondamentale tra HINT e le normali applicazioni di benchmark (un esempio è HBench) è che il programma HINT è costituito da un mix fisso di istruzioni (ad esempio, il rapporto delle operazioni di calcolo per le operazioni di memoria è di circa 1). HINT è un buon indicatore di prestazioni per le applicazioni che hanno un mix di istruzioni simile tra loro, ma non necessariamente per le applicazioni con un differente mix di istruzioni.

Dujmovic propone, invece, un approccio sistematico per la costruzione di programmi di benchmark. Essi impiegano un modello BFK (block-frame-kernel) che costruisce ricorsivamente programmi di benchmark utilizzando elementi di base come strutture di controllo e kernel, che sono o semplici dichiarazioni o brevi sequenze di codice che implementa una funzione specifica. Un programma di benchmark è costituito da un numero di blocchi che contengono uno o più frame, ciascuno dei quali a sua volta, può comprendere i kernel o i blocchi. I parametri dei modelli includono l'ampiezza del blocco, che indica il numero di fotogrammi inclusi nel blocco, e la profondità del blocco, che indica il livello di nidificazione. Un programma di benchmark si caratterizza per la distribuzione di probabilità di questi due parametri, la distribuzione di probabilità delle strutture di controllo, e il codice del kernel. Gli autori suggeriscono che, selezionando attentamente i kernel da includere nel benchmark, si possono costruire programmi di benchmark che modellano ogni tipo di carico di lavoro.

Questo approccio, concettualmente, potrebbe ridurre notevolmente i tempi di sviluppo di benchmarking, con un conseguente molto più breve ciclo di produzione di riferimento. Esso, tuttavia, rappresenta solo un passo incrementale nella scienza della valutazione comparativa. Anche se i programmi di benchmark possono essere prodotti più rapidamente, hanno ancora bisogno di essere eseguiti su tutte le configurazioni possibili per determinare le prestazioni. Se cambia il comportamento dell'applicazione, deve essere costruito e poi eseguito un nuovo benchmark per ogni configurazione possibile.

Il principale vantaggio dei tradizionali software di benchmark (come HBench), è, quindi, che una volta che i dati sulle prestazioni del sistema vengono raccolti su una macchina, non è necessario rieseguire l'applicazione o il benchmark su tale macchina anche se l'applicazione dovesse essere modificata nel tempo. Questo però potrebbe non compensare i vantaggi che si otterrebbero con i nuovi approcci proposti.

2 Panoramica della suite (ESM Bench)

La Suite da noi realizzata (dal nome ESM Bench), raccoglie i modelli esistenti di valutazione delle prestazioni sviluppati nell'ambito dell'HPC, con il vantaggio di avere un'applicazione orientata alla comprensione e al miglioramento delle prestazioni dei modelli ESMs per gli attuali sistemi informatici e per il futuro (come per la PETA e le exascale architecture).

L'ESM Bench si compone di una suite di applicazioni che implementano i modelli numerici di simulazione dei fenomeni climatici. Una prima serie di applicazioni è stata definita tenendo conto di una serie di criteri di selezione descritti in seguito. La Suite di valutazione include due diverse categorie di modelli:

1. **modelli accoppiati**, spesso usati dai climatologi per valutare il sistema nel suo complesso, come la gamma di componenti del modello in questione (per esempio: fisiche, chimiche e biologiche) e le loro interazioni,

2. **modelli stand-alone**, generalmente utilizzati per effettuare simulazioni ad alta risoluzione nel breve periodo.

Ogni modello componente la suite fornisce un risultato espresso in termini di **execution time** e poi l'applicazione realizzata nell'ambito di questa tesi (chiamata ESM Bench) prenderà tutti questi risultati per farne una media e fornire un valore di benchmark indicante le prestazioni dell'intero sistema.

2.1 Suite: ESM Bench

2.1.1 Definizione della metrica per la misura delle prestazioni

L'**SYPD** rappresenta la quantità di anni che si riuscirebbero a simulare in 24 ore di run, sapendo che per simulare la finestra temporale definita (TIMESIMULATED) abbiamo impiegato il tempo restituito (execution time).

Per arrivare alla formula finale analizziamo bene i risultati ottenuti dalla suite di benchmark.

Al momento dell'esecuzione della suite viene indicato il numero di giorni che si vogliono simulare (indichiamo questo valore tramite il termine TIMESIMULATED), e il programma (ESM Bench, da noi realizzato) restituisce un valore in secondi indicante quanto tempo in media è stato impiegato per simulare TIMESIMULATED giorni.

Per questo motivo, se vogliamo risalire a quanti anni si possono simulare eseguendo il modello per 24 ore, utilizziamo la seguente proporzione:

*TIMESIMULATED giorni : (execution time) s = X giorni : 24h * 60m * 60 s*

Di conseguenza

$$X\ giorni = \frac{\text{TIMESIMULATED} * 24\text{h} * 60\text{m} * 60\text{s}}{\text{execution time}}$$

E la formula dell'**SYPD** è

$$\text{SYPD} = \frac{\text{X giorni}}{365} = \frac{\text{TIMESIMULATED} * 24\text{h} * 60\text{m} * 60\text{s}}{\text{execution time} * 365}\ \text{anni}$$

Ovviamente se TIMESIMULATED è in mesi, si otterrà X mesi e di conseguenza SYPD sarà uguale a X mesi / 12 e la stessa cosa vale nel caso in cui TIMESIMULATED sia espresso in anni (si divide per uno).

2.1.2 Composizione della suite

La suite qui descritta misura le prestazioni complessive del sistema sul quale viene eseguita.

Essa è composta da diverse applicazioni parallele (per maggiori dettagli si vedano i paragrafi successivi), ciascuna delle quali, al termine dell'esecuzione, restituisce un valore prestazionale in termini di tempo di esecuzione.

L’intera suite viene eseguita diverse volte, su un numero sempre diverso di processori, calcolando ogni volta un **SYPD** diverso.

I passi che descrivono il modo in cui il valore finale di benchmark di ogni singola esecuzione è ottenuto sono i seguenti:

1. Si configura e si compila ogni modello componente la suite;
2. Si esegue ogni modello sulla macchina in fase di test;
3. Ogni applicazione restituisce un valore indicante il tempo di esecuzione (execution time) che rappresenta il tempo impiegato per simulare i processi climatici in una finestra temporale definita (generalmente si simula un anno, ma per completezza chiameremo tale valore TIMESIMULATED);
4. Si calcola una media pesata (al momento il peso è pari a 1 per ogni applicazione) dei tempi di esecuzione ottenuti;
5. Si calcola l’SYPD (Simulated Year per day) attraverso la formula indicata nel successivo paragrafo.

Tale procedura viene eseguita diverse volte, facendo in modo che il numero di processori a disposizione sia sempre diverso ed incrementato rispetto al test precedente.

Al termine dell’intera procedura la suite restituisce 2 valori:

1. **SYPD massimo**, con il numero di processori con il quale si è ottenuto;
2. **SYPD massimo** tra quelli **con efficienza relativa ≥ 0.3** , con il numero di processori con il quale si è ottenuto.

2.1.3 Configurazione dei processori da utilizzare

Essendo composta da applicazioni parallele, è bene eseguire la suite diverse volte, ogni volta con un numero di processori diverso, prima di restituire un risultato indicativo delle prestazioni del sistema.

L'utente che esegue tale suite è sicuramente interessato a conoscere quale sia la migliore configurazione di sistema da usare per ottenere le prestazioni migliori.

A tale scopo vengono forniti 2 valori:

1. **SYPD massimo**, è il valore che indica in quale circostanza si è ottenuto il minor tempo di esecuzione. Tale valore però non prende in considerazione l'efficienza del sistema.
2. **SYPD massimo** tra quelli **con efficienza ≥ 0.3**, che invece considera quale sia l'efficienza dell'intero sistema (per i motivi di tale scelta si veda il paragrafo: **2.1.6**).

2.1.4 Efficienza

Il calcolo dell'efficienza nel caso di architetture parallele è molto importante, in quanto misura l'utilizzo del processore nel calcolo.

L'efficienza è calcolata attraverso la seguente formula:

$$\varepsilon = \frac{\text{Speedup}}{\#\text{processori}} = \frac{T_0}{T_p * \#\text{processori}}$$

Dove:

- **Speedup**: è un rapporto (T_0/T_p) che indica quanto un algoritmo parallelo sia più veloce del corrispondente algoritmo sequenziale;
- **# processori**: indica il numero di processori utilizzati.

2.1.5 Efficienza relativa

Come è possibile notare nella formula precedente, l'efficienza viene generalmente calcolata conoscendo il tempo di esecuzione nel caso di un solo processore (T_0).

Nel nostro caso, però, abbiamo un problema: se provassimo ad eseguire la suite su un unico processore, non solo dovremmo perdere molto tempo in attesa del risultato, ma potrebbe accadere addirittura che l'intero processo si blocchi a causa della mancanza di risorse.

Per tale motivo abbiamo scelto di considerare **l'efficienza RELATIVA**, che prende come valore di riferimento quello ottenuto con un numero di processori diverso da 1.

Quindi potremmo partire con l'esecuzione su 16 processori, ponendo come tempo T_0 quello ottenuto e calcolare l'efficienza con questa formula:

$$\varepsilon = \frac{T_0 * P_0}{T_p * \#\text{processori}}$$

Dove P_0 sarebbe pari a 16!

2.1.6 Andamento dell'Efficienza

E' noto, da diversi studi effettuati a riguardo, che l'efficienza tende a diminuire man mano che aumenta il numero di processori.

Questo fenomeno è giustificato dal fatto che il tempo necessario per gestire le operazioni di gestione del calcolo parallelo tende ad aumentare con il numero di processori, tanto da non essere più compensato dalla diminuzione del tempo di esecuzione dell'intera suite (overhead).

Per tale motivo è bene fornire, oltre all'SYPD massimo, anche l'SYPD ottenuto con efficienza > 0.3 (valore stabilito in base alle necessità).

2.1.7 Perchè media PESATA dei valori piuttosto che altri tipi di media

Per ottenere il tempo di esecuzione da usare nella formula dell'SYPD, abbiamo scelto di calcolare la media dei valori ottenuti da ciascun modello componente la suite.

La media calcolata è una **media pesata** perché nel futuro gli sviluppatori potranno scegliere di attribuire più peso ad una piuttosto che a un'altra applicazione.

Al momento attuale, si è preferito lasciare il peso di ogni modello pari ad 1.

Si è scelto di calcolare la media pesata piuttosto che la media geometrica o la mediana dei valori ottenuti per i seguenti motivi:

1. La media geometrica viene utilizzata generalmente quando si hanno dei valori ottenuti tramite dei rapporti, e questo non è il nostro caso;
2. La media geometrica, essendo calcolata come radice n-esima del prodotto degli n valori disponibili, viene usata se si vuole dare una maggiore importanza ai valori più piccoli, mentre il nostro obiettivo è proprio il contrario: ottenere un valore che indichi quanto tempo impiega la macchina ad eseguire il modello! Ed è ovvio che l'utente non vuole conoscere qual è il caso migliore (il caso migliore potrebbe non verificarsi mai nella realtà)!
3. La mediana prende il valore intermedio tra quelli trovati. Questo vuol dire che il valore intermedio potrebbe anche essere un valore molto lontano dalla reale media (si pensi ad esempio al caso in cui si sono trovati i seguenti valori: 1 ; 2 ; 3 ; 54 ; 100 ; Il valore mediano è 3, anche se la media dei valori è 32!)

2.1.8 Perché si è scelto di considerare l'SYPD piuttosto che altri indicatori

Il nostro obiettivo era quello di valutare le performance totali della macchina testata. Per questo motivo si è scelto di **non** utilizzare parametri come:

1. Rapporto qualità/prezzo che non rende la reale idea delle prestazioni offerte dalla macchina;
2. M/FLOPS (**FL**oating point **O**perations **P**er **S**econd) che indica solo quante operazioni in virgola mobile esegue la macchina in un secondo;
3. Tempo di CPU, che non include il tempo speso nell'attesa di compiere operazioni di I/O o nell'esecuzione di altri programmi;
4. altri indicatori di performance.

Inoltre, valutando il tempo di esecuzione di ogni modello componente la suite e calcolandone l'SYPD possiamo considerare, e quindi valutare, le performance durante le fasi di pre e post processing, ovvero le fasi che precedono e seguono la reale fase di esecuzione del programma (quindi tempo necessario per caricare il programma in memoria, tempo impiegato dal sistema operativo per assegnare le risorse al processo ecc.).
In aggiunta, in questo modo, possiamo scegliere di valutare le prestazioni avendo una macchina carica o non carica, ovvero con o senza altri processi e programmi in esecuzione contemporaneamente.

Un altro aspetto molto importante che è stato tenuto in considerazione è la necessità di avere un indice che crescesse al crescere delle prestazioni della macchina analizzata:

> Se invece dell'SYPD avessimo usato il semplice tempo di esecuzione (*execution time*) avremmo ottenuto un indice che è inversamente proporzionale alle prestazioni della macchina.
> Cioè:

un indice di prestazioni *execution time* maggiore indica una minore efficienza della macchina.

Mentre con SYPD:

Un SYPD maggiore indica migliori prestazioni della macchina testata

2.1.9 Perchè non c'è una macchina di riferimento

Una macchina di riferimento darebbe la possibilità agli utenti finali di avere un quadro di riferimento per giudicare le metriche che altrimenti sarebbero solo un insieme di numeri senza senso.

Noi però usiamo il tempo come riferimento e non la velocità di una particolare macchina.

2.2 Modelli

2.2.1 Aspetti generali

I modelli Climatici che descrivono il sistema terra (Earth System Models: ESMs) sono gli unici strumenti analitici disponibili per predire l'evoluzione futura del clima sia se sottoposto ai soli eventi naturali, sia se sottoposto all'influenza degli esseri umani.

Lo sviluppo e l'uso di modelli climatici realistici richiede un'infrastruttura sofisticata di software e un accesso ai più potenti supercomputers e sistemi di gestione dati.

Molte istituzioni scientifiche, università, organizzazioni governative e partner industriali in Europa hanno sviluppato una classe di competenza mondiale esperta in differenti aspetti della modellazione del sistema terra ed hanno contribuito all'internazionale assestamento del cambiamento climatico.

I complessi processi componenti un modello sul sistema clima ad alta definizione, che meglio simulano le interazioni e le retroazioni tra i processi fisici e biologici sono: l'atmosfera, l'oceano, la terra, il suolo, il permafrost, la vegetazione, il ghiaccio marino, il ghiaccio terrestre, il carbone e altri cicli biogeochimici, le nuvole e la microfisica

connessa, l'idrologia, la chimica atmosferica, gli aerosol, le lastre di ghiaccio, i sistemi umani.

I modelli attuali prevedono già una migliorata simulazione e previsione delle variazioni di temperatura e precipitazione e di eventi meteorologici estremi.

2.2.2 Criteri di selezione dei modelli da utilizzare

Sono molti i modelli climatici attualmente disponibili a livello mondiale. Essi differiscono sia per i requisiti computazionali e l'affidabilità scientifica delle previsioni fatte, che per la modellazione dei comportamenti climatici con il loro impatto.

Per definire i dettagli della suite di valutazione si è discusso su differenti criteri di selezione:

- Interesse per il modello. Il principale criterio adottato si basa sulla stima di quanto il modello possa essere d'interesse per i partner dell'IS-ENES. Da un punto di vista pragmatico, tutti i modelli appartenenti alla suite di valutazione dell'IS-ENES, possono essere conservati, utilizzati e ottimizzati sulla maggior parte delle architetture parallele.
  Quanto più un modello è d'interesse tanto più viene mantenuto, ottimizzato e analizzato.
  L'interesse per un modello viene misurato valutando il numero di progetti che i partner hanno per utilizzare, ottimizzare, creare supporto per esso;
- Diffusione del modello. Questo criterio misura quanto il modello può essere rilevante per una più ampia possibile community sul clima;
- Piattaforme supportate: Questo criterio è relativo alla possibilità che ha un modello di essere utilizzato su diverse architetture parallele. Quante più sono le piattaforme sulle quali il modello è già stato utilizzato e analizzato, tanto minore sarà lo sforzo per sostenere nuove architetture;
- Performance richieste: Valutazione qualitativa della memoria, CPU, I / O, uso della rete. La metrica può essere bassa / media / alta;

- Estensioni del modello: Questo criterio si riferisce al numero dei componenti del modello integrati in altri modelli accoppiati;
- Potenzialità per la peta-exascaling: Criterio qualitativo che indica quanto un modello potrà essere scalato fino ad architetture exaflop e quanto difficile è modificarlo per le piattaforme exascale;
- Livello di documentazione del modello: La disponibilità di una buona documentazione allegata al modello è importante per mantenere il modello ed eseguire la suite di valutazione sviluppata.

2.2.3 Descrizione dei modelli

Come già detto, la suite di valutazione include due diverse categorie di applicazioni:

1. **Modelli accoppiati**, spesso usati dai climatologi per valutare il sistema nel suo complesso, come la gamma di componenti del modello in questione (per esempio: fisiche, chimiche e biologiche) e le loro interazioni,
2. **Modelli stand-alone**, generalmente utilizzati per effettuare simulazioni ad alta risoluzione nel breve periodo.

I Modelli utilizzati nella suite sono i seguenti:

Modelli accoppiati:

- CMCC-MED
- ARPEGE-NEMO
- IPSLCM5
- HadGEM2
- MPI-M

Modelli stand alone:

- ECHAM
- NEMO

Le informazioni riguardo ai modelli sono state organizzate in 2 sezioni:

- La sezione 2.2.3.1 è dedicata ai modelli accoppiati (coupled models);
- La sezione 0 è dedicata ai modelli stand-alone.

2.2.3.1 Modelli accoppiati (Coupled models)

2.2.3.1.1 CMCC-MED

Informazioni generali	
Nome del modello	CMCC-MED
Persone ed enti di riferimento	Silvio Gualdi, gualdi@bo.ingv.it, CMCC
Versione	2.0
Breve descrizione	Tre componenti completamente accoppiati per il modello del clima nelle regioni del mar mediterraneo
Infrastruttura (e.g. coupler)	OASIS3 ver. 2.5 (CMCC parallel version)
Download	Per informazioni riguardo il codice per il download, si prega di contattare le persone ed enti di riferimento
Configurazione	
Configurazione di input	Il modello accoppiato è composto da: Echam5 T159L31 OPA 8.2 global Nemo 1/16° for Mediterranean sea

	Oasis configuration: Numero totale di campi di scambio: 35 Campi scambiati: 17 (Echam -> Oasis -> OPA) Campi scambiati: 9 (Echam -> Oasis -> Nemo) Campi scambiati: 6 (OPA -> Oasis -> Echam) Campi scambiati: 3 (Nemo -> Oasis -> Echam) Periodo di accoppiamento: 2h 40' Echam5 Regione: Globale Risoluzione: T159L31 (480 x 240) time step: 240 sec OPA Regione: Globale Risoluzione: 2° (182 x 149) time step: Nemo: Regione: Mediterranean Sea Risoluzione: 1/16° (871 x 253)
Dimensioni dei file di input	Echam5: 912MB OPA: 96MB NEMO: 1.6GB OASIS: 1.1GB restart files (1 month): Echam5: 1.23GB OPA: 102MB Nemo: 1.5GB OASIS3: 30,5MB
Dimensioni dei file di output	Echam5: 9.2GB Nemo: 18GB

	OPA: 1.5GB

Modelli componenti inclusi nel modello accoppiato	
Modello componente	
Nome del modello componente	OASIS
Versione	2.0.1 CMCC parallel version basata sulla versione 3_prism_2-5
License policy	Lesser GNU General Public License (LGPL)
Linguaggi/o di Programmazione	C, Fortran
Librerie	NetCDF, MPI, OpenMP,
Metodo di parallelizzazione supportato	MPI1, MPI2, OpenMP
Modello componente	
Nome del modello componente	Echam
Versione	v. 5
License policy	"MPI-M Software Licence Agreement", che deve essere registrata da ogni utente
Linguaggi/o di Programmazione	Fortran
Librerie	NetCDF, MPI, OpenMP, Lapack, Blas
Metodo di parallelizzazione supportato	MPI2, OpenMP
Modello componente	
Nome del modello componente	OPA
Versione	v. 8.2
License policy	CeCILL license (public license)
Linguaggi/o di Programmazione	F95
Librerie	NetCDF, MPI
Metodo di parallelizzazione supportato	MPI1, MPI2
Modello componente	
Nome del modello componente	Nemo

Versione	v. 9
License policy	CeCILL license (public license)
Linguaggi/o di Programmazione	F95
Librerie	NetCDF, MPI
Metodo di parallelizzazione supportato	MPI

2.2.3.1.2 ARPEGE-NEMO

Informazioni generali	
Nome del modello	ARPEGE-NEMO
Persone ed enti di riferimento	Eric Maisonnave, eric.maisonnave@cerfacs.fr, CERFACS
Versione	ARPEGE-Climat v5.2 – NEMO v3.2
Breve descrizione	CGCM high definition
Infrastruttura (e.g. coupler)	OASIS v3 (pseudo parallel) – OASIS v4 (work in progress)
Download	A causa di restrizioni di licenza che lo compongono, CERFACS non fornisce alcuna versione del modello accoppiato
Riferimenti per informazioni dettagliate	Per informazioni generali sul flusso di lavoro ARPEGE-NEMO, vedere LEGO: Grid Compliant Climate Model Analysis. Per informazioni sul porting e l'ottimizzazione della configurazione ad alta risoluzione leggere il documento seguente.
Configurazione	
Configurazione di input	Il modello accoppiato è composto da: ARPEGE-5 T359L31 (possibly higher resolution) NEMO-3 ¼ degree (possibly 1/12 degree – 1D configuration also available) Oasis configuration:

	OASIS-3 Pseudo-parallel mode (possibly OASIS-4) Periodo di accoppiamento: 3h Campi accoppiati (configurazione ridotta): O2A: temperature in superficie , albedo A2O: flussi di calore, flussi d'acqua (senza calving e deflusso), flusso del caldo non derivato dal sole (non solar heat flux derivative) ARPEGE-5 Regione: Globale Risoluzione: T359L31 (360 x 180) time step: 900 sec NEMO-3 Regione: Globale Risoluzione: 1/4 (1442 x 1021) time step: 1080 sec
Dimensioni dei file di input	ARPEGE: 87 MB NEMO: 384 GB OASIS: 8 GB restart files (1 month): ARPEGE: 48 MB NEMO: 8.4 GB OASIS3: 50 MB
Dimensioni dei file di output	Monthly means: ARPEGE: 4.5GB NEMO: 745 MB

Modelli componenti inclusi nel modello accoppiato	
Modello componente	
Nome del modello componente	ARPEGE-Climat
Versione	v5
License policy	"ARPEGE-Climat Software Licence Agreement", che deve essere registrata per ogni utente (contattare CNRM, Météo-France)
Linguaggi/o di Programmazione	Fortran
Librerie	BLAS, LAPACK
Metodo di parallelizzazione supportato	MPI1 (MPICH, OpenMPI, LAM tested), possibly OpenMP
Modello componente	
Nome del modello componente	NEMO
Versione	v3
License policy	CECIL licence, vedere http://www.nemo-ocean.eu/user/register
Linguaggi/o di Programmazione	Fortran90
Librerie	Netcdf, xml
Metodo di parallelizzazione supportato	MPI1 (MPICH, OpenMPI, LAM tested)
Modello componente	
Nome del modello componente	OASIS
Versione	V3
License policy	LGPL, see https://oasistrac.cerfacs.fr/
Linguaggi/o di Programmazione	Fortran/C
Librerie	Netcdf
Metodo di parallelizzazione supportato	Pseudo parallelism IPSL/CERFACS method

2.2.3.1.3 IPSLCM5 Model

Informazioni generali	
Nome del modello	IPSLCM5
Persone ed enti di riferimento	Arnaud Caubel and Marie-Alice Foujols, IPSL, arnaud.caubel@lsce.ipsl.fr ; foujols@ipsl.jussieu.fr
Versione	IPSLCM5 v3
Breve descrizione	Oceano-mare ghiaccio-atmosfera-terra modello sul clima, pronto per il sistema terra (chimica, biogeochimica marina e ciclo del carbonio aggiunto)
Infrastruttura (e.g. coupler)	OASIS v3 (pseudo parallel)
Download	Per informazioni riguardo il codice per il download, si prega di contattare le persone ed enti di riferimento
Configurazione	
Configurazione di input	Il modello IPSLCM5 consinste in: LMDZ4+ORCHIDEE (CMIP5) NEMO v3_2 Regione: Globale Oasis configuration: Numero totale di campi scambiati: 21 Campi scambiati: 17 (LMDZ4 -> Oasis -> NEMO) Campi scambiati: 4 (NEMO -> Oasis -> LMDZ4) Periodo di accoppiamento: 24h LMDZ4 Risoluzione: 280x280x19 (è possibile anche 280x280x39 se è disponibile più memoria) Time step: 72 s (dynamics) and 30 mn (physics) NEMO

	Regione: Globale Risoluzione: ORCA05 ½° (511x722x31) time step: 40 mn
Dimensioni dei file di input	LMDZ4/ORCHIDEE: 140 MB NEMO: 1.1GB OASIS: 900 MB (9GB with calculated weights for runoff) restart files (5 days): LMDZ4/ORCHIDEE: 600 MB NEMO : 1.5 GB OASIS: 23 MB
Dimensioni dei file di output	LMDZ4/ORCHIDEE: 315 MB (histday.nc and histhf.nc) NEMO: 650 MB

Modelli componenti inclusi nel modello accoppiato	
Modello componente	
Nome del modello componente	LMDZ4 - ORCHIDEE
Versione	CMIP5
License policy	CeCILL licenses
Linguaggi/o di Programmazione	Fortran
Librerie	NetCDF, MPI, OpenMP, Lapack, Blas
Metodo di parallelizzazione supportato	MPI1, OpenMP
Modello componente	
Nome del modello componente	NEMO
Versione	v. 3.2
License policy	CeCILL license
Linguaggi/o di Programmazione	Fortran
Librerie	NetCDF, MPI

Metodo di parallelizzazione supportato	MPI1
Modello componente	
Nome del modello componente	OASIS
Versione	V3
License policy	LGPL, see https://oasistrac.cerfacs.fr/
Linguaggi/o di Programmazione	C, Fortran
Librerie	NetCDF
Metodo di parallelizzazione supportato	MPI1, Pseudo parallelism IPSL/CERFACS method

2.2.3.1.4 HadGEM2

Informazioni generali	
Nome del modello	Unified Model: HadGEM3A configuration
Persone ed enti di riferimento	Steve.Mullerworth@metoffice.gov.uk
Versione	7.4
Breve descrizione	Modello atmosferico
Infrastruttura (e.g. coupler)	Ingresso impostato tramite interfaccia utente. Nessun attacco (può essere usato accoppiato a NEMO tramite OASIS)
Download	Fornito su CD. E' richiesta una Licenza
Riferimenti per informazioni dettagliate	La documentazione è fornita con il disco d'installazione
Configurazione	
Configurazione di input	La configurazione è definita da una User Interface,e possono essere fornite diverse configurazioni.. Una tipica configurazione è una risoluzione di 192 punti E-W da 145 punti N-S su 63 livelli atmosferici, eseguiti con un intervallo di tempo di 20 minuti per un mese
Dimensioni dei file di input	Generalmente 4Gb

Dimensioni dei file di output	Configurabile – minimo 2Gb

Modelli componenti inclusi nel modello accoppiato	
Modello componente	
Nome del modello componente	HadGEM3A
Versione	7.4
License policy	Licenza disponibile per utenti nell'ambito accademico/della ricerca
Linguaggi/o di Programmazione	FORTRAN + un po' di C
Librerie	GCOM
Metodo di parallelizzazione supportato	MPI. Alcuni OpenMP ancora in fase di sviluppo

2.2.3.1.5 MPI-M

Informazioni generali	
Nome del modello	ECHAM5/MPIOM
Persone ed enti di riferimento	Marco Giorgetta, marco.giorgetta@zmaw.de, MPI-M
Versione	COSMOS-1.2.1.1
Breve descrizione	I modelli che descrivono il sistema terra consistono in modelli accoppiati di atmosfera, oceano e terra, incluso il ciclo del carbonio.
Infrastruttura (e.g. coupler)	IMDI SCE/SRE, OASIS3
Download	Disponibile per il download su richiesta fatta a questo URL: http://www.mpimet.mpg.de/en/wissenschaft/modelle/model-distribution.html
Riferimenti per informazioni dettagliate	"The MPI-M special section in J. Climate" Vol.19 No.16
Configurazione	
Configurazione di input	"ASOB" configuration, i.e. coupled climate carbon

	cycle model ECHAM5J/MPIOM Setup like Control run for preindustrial conditions, as done for ENSEMBLES stream 2. Length of integration in this case = 20 years Oasis configuration: Campi scambiati: 17 (ECHAM5J → Oasis → MPIOM) Campi scambiati: 8 (MPIOM → Oasis → ECHAM5J) coupling period: 1 day ECHAM5 Regione: Globale Risoluzione: T31L19 (96 x 48) time step: 2400 sec MPIOM Regione: Globale Risoluzione: 3° time step: 8640 sec
Dimensioni dei file di input	Specificare la dimensione del file di input con questa dimensione: 58 Mbyte
Dimensioni dei file di output	Dimensione dei file di output: o restart: 36 Mbyte o diagnostic:

Modelli componenti inclusi nel modello accoppiato	
Modello componente	
Nome del modello componente	**ECHAM5**

Versione	ECHAM5.4+JSBACH
License policy	"MPI-M Software Licence Agreement", da registrare ogni utente, Vedere http://www.mpimet.mpg.de/en/wissenschaft/modelle/model-distribution.html
Linguaggi/o di Programmazione	Fortran, C, (inkl OpenMP)
Librerie	NetCDF, CDI, MPI (to name a few)
Metodo di parallelizzazione supportato	MPI1, MPI2, OpenMP
Modello componente	
Nome del modello componente	**MPIOM**
License policy	"MPI-M Software Licence Agreement", da registrare ogni utente, Vedere http://www.mpimet.mpg.de/en/wissenschaft/modelle/model-distribution.html
Linguaggi/o di Programmazione	Fortran, C(inkl OpenMP)
Librerie	NetCDF, CDI, MPI (to name a few)
Metodo di parallelizzazione supportato	MPI1, MPI2, OpenMP
Modello componente	
Nome del modello componente	**OASIS3**
Versione	prism_2_3
License policy	Lesser GNU General Public License (LGPL)
Linguaggi/o di Programmazione	Fortran, C (inkl OpenMP)
Librerie	NetCDF, CDI, MPI (to name a few)
Metodo di parallelizzazione supportato	MPI1, MPI2, OpenMP

2.2.3.2 Modelli Stand-alone

2.2.3.2.1 ECHAM

Informazioni generali	
Nome del modello	ECHAM5 ECHAM6 nella seconda metà del 2010
Persone ed enti di riferimento	Marco Giorgetta, marco.giorgetta@zmaw.de, MPI-M, name, email and institution
Versione	ECHAM5.4.02 ECHAM6.0.nn in 2nd half 2010
License policy	Vedere http://www.mpimet.mpg.de/en/wissenschaft/modelle/model-distribution.html
Linguaggi/o di programmazione	Fortran, C (inkl OpenMP)
Metodi di parallelizzazione supportati	MPI, OpenMP, pthreads, hybrid, SHMEM, ...
Download	ECHAM5: si ECHAM6: lo sarà presto
Riferimenti per informazioni dettagliate	Roeckner et al., The atmospheric general circulation model ECHAM 5. PART I: Model description, MPI report 349, 2003. http://www.mpimet.mpg.de/fileadmin/publikationen/Reports/max_scirep_349.pdf
Configurazione	ECHAM5 T63L31
Configurazione di input	192 x 96 horizontal points, 31 levels, Δt = 720 sec. ;6 hrly output, yrly restart files
Dimensioni dei file di input	19,8 MB
Dimensioni dei file di output	Output: 15,5 GB / yr; Restart: 200 MB / yr; Monthly means: 141 MB / yr; $\sum$ = 17,8 GB / yr

2.2.3.2.2 NEMO

Informazioni generali	
Nome del modello	Nucleus for European Modelling of the Ocean
Persone ed enti di riferimento	Claire Lévy, Claire.Levy@locean-ipsl.upmc.fr
Versione	Reference version tag nemo_v3_1
License policy	CeCILL2, free licence
Linguaggi/o di programmazione	Fortran95, partially Fortran2003
Librerie	MPI, NETCDF
Metodi di parallelizzazione supportati	MPI
Download	Disponibile per il download: contattare le persone ed enti di riferimento
Riferimenti per informazioni dettagliate	Vedere: http://www.nemo-ocean.eu/About-NEMO/Reference-manuals.
Configurazione	ORCA12 (global 1/12 degree resolution). Already used as benchmark at CINES, GENCI, France
Configurazione di input	ORCA12 : (Global ocean model with 1/12 deg resolution at the Equator) Grid size: 4322 x 3059 x 50. Time step is 240 sec. Bench runs 360 time steps (1 day).
Dimensioni dei file di input	Bathymetry = 158 Mb Initial conditions : 5.3 Gb
Dimensioni dei file di output	model output = 12 Gb 1 restart file = 78 Gb (likely not in 1 single file)

3 Fase operativa

Per garantire la massima portabilità di un'applicazione, i programmatori, generalmente, preferiscono fornire agli utenti i file sorgente del loro programma, piuttosto che il programma già compilato, e di conseguenza, prima di poter passare alla fase di esecuzione, è necessario effettuare la configurazione e la successiva compilazione degli stessi.

Le fasi preliminari appena menzionate (configurazione e compilazione) generalmente sono fasi lunghe ed impegnative, in quanto l'utente dovrebbe effettuare tutto manualmente: preparare le cartelle necessarie al programma, compilare i file sorgente nelle giuste posizioni, annotare tutte le informazioni che saranno poi necessarie per l'esecuzione del programma, ecc.

I modelli di benchmark analizzati in questa tesi, sono volutamente resi portabili tramite la fornitura dei soli file sorgente, e di conseguenza è necessario, anche per questi, effettuare tutte le fasi preliminari prima descritte.

E' proprio partendo da queste considerazioni, alle quali si aggiunge anche la necessità di compiere tutte le operazioni preliminari per ogni modello componente la suite, che si è deciso di sviluppare un software in grado di semplificare e automatizzare tutto questo processo iniziale.

3.1 Analisi dei requisiti

L'obiettivo perseguito nell'ambito di questa tesi è stato quello di creare, partendo da qualcosa di già esistente, un'applicazione in grado di effettuare tutte le operazioni preliminari e necessarie alla successiva esecuzione dei vari modelli di benchmark.

Partendo dall'applicazione JuBE, sviluppata da Forschungszentrum Juelich GmbH (per tutti i dettagli si rimanda a [5]), è stato creato uno script Perl che, acquisendo una serie di parametri definiti in opportuni file xml di definizione, permette di effettuare

tutte le operazioni di configurazione, compilazione e anche di esecuzione attraverso un unico e semplice comando.

Tale script è poi stato integrato all'interno della suite prima descritta, responsabile dell'esecuzione di tutti i modelli e dell'estrapolazione della media dei valori ottenuti.

Lo scopo è stato perseguito cercando di rendere l'applicazione il più possibile:

- **Portabile**: La portabilità è la possibilità di utilizzare l'applicazione su architetture diverse. Questo obiettivo è stato raggiunto tramite la parametrizzazione di tutte le informazioni necessarie ai compilatori e ai vari script che intervengono nelle varie fasi;
- **Facilmente usabile**: è la proprietà di un oggetto di essere facilmente capito, utilizzato e gradito;

Oltre che

- **Corretta**;
- **Affidabile**;
- **Robusta**;
- **Efficiente**;
- **Scalabile**;

Tutte caratteristiche di un software ben progettato.

Il modo in cui si è pensato di rendere lo script portabile, quindi adattabile a qualsiasi piattaforma, è la definizione di tutte le variabili di ambiente all'interno di alcuni file xml di definizione e settaggio.

I file xml messi a disposizione dell'utente sono:

- **platform.xml**, all'interno del quale vengono dichiarate tutte le librerie, le istruzioni e i parametri da passare alle stesse per poter eseguire, senza alcuna altra modifica, lo script.
- File **.job.in**, all'interno del quale vengono definite tutte le istruzione che lo scheduler dovrà distribuire all'architettura parallela.
- **compile.xml**, all'interno del quale sono dichiarate tutte le variabili necessarie in fase di compilazione, ma che l'utente non è costretto a modificare, in quanto vengono prelevate automaticamente dal file platform.xml al momento della compilazione.
  L'unica modifica necessaria da parte dell'utente è l'indicazione dell'istruzione, diversa per ogni architettura, che il sistema dovrà utilizzare per la compilazione di tutti i file (<command>./configure; make -f Makefile</command>).
- **execute.xml**, nel quale, oltre ai parametri prelevati autonomamente dal file platform.xml, viene indicato il nome da attribuire al file .job (descritto nei paragrafi successivi), e il nome della cartella che lo contiene.
- **prepare.xml**, nel quale vengono indicati i file utilizzati per la formattazione dell'output
- **result.xml**, nel quale si indicano quali campi stampare nella tabella dei risultati
- Vari file **xml di definizione** dei parametri da stampare (si veda l'**appendice A** per tutti i dettagli).

La diretta conseguenza di questo approccio così parametrizzato, è la necessità di effettuare una dettagliata analisi dei requisiti in conseguenza della quale si potrà poi fare una corretta configurazione dell'intera suite.

La procedura da seguire per la corretta configurazione dello script si preferisce illustrarla congiuntamente all'esempio riportato nel paragrafo **3.5** .

3.2 Componenti dello script

Lo script di configurazione si compone di diversi file distribuiti in cartelle atte a rendere il più semplice e parametrica possibile la configurazione.

Le cartelle di cui si compone sono:

- **applications**, all'interno della quale verranno copiati i vari modelli da eseguire con i relativi file di configurazione (ECHAM5, NEMO, ecc…)
- **bench**, contenente l'intero script corredato da tutti i file necessari alla sua corretta esecuzione (e non necessita di modifiche da parte dell'utente).
- **platform**, contenente i file attraverso i quali si definiscono tutte le caratteristiche del sistema sul quale si vuole eseguire il benchmark.
- **skel**, contenente i file utilizzati nella fase finale per la generazione dei risultati. (Per maggiori dettagli si veda **APPENDICE A**).

All'interno di ogni cartella troviamo i diversi file di configurazione da modificare prima dell'esecuzione dello script. Tali file verranno illustrati nel capitolo successivo congiuntamente all'esempio di configurazione.

3.3 Funzionamento dello script

Il funzionamento dello script è suddiviso in diverse fasi, differenti a seconda dei parametri passati come argomento.

All'interno della procedura si farà riferimento all'invio dei comandi per la reale e conclusiva esecuzione dell'applicazione (job), ad uno scheduler.

Lo scheduler non è altro che un componente del sistema in uso che definisce le politiche di scheduling per l'esecuzione dei job (applicazioni) all'interno del cluster parallelo.

Tale software è diverso da architettura ad architettura e richiede l'utilizzo di istruzioni diverse per l'esecuzione; per questo motivo, lo script prevede la definizione di

tutte le istruzioni e parametri da passare a tale sistema, all'interno di un file (che ha estensione .job.in), che verrà elaborato e successivamente usato per lanciare il job.

Lo scheduler viene richiamato tramite un'unica istruzione (nel nostro caso *bsub*), indicata opportunamente nel file platform.xml, ed esso eseguirà tutte le istruzioni indicate nel file specificato (nel nostro caso *ibm_llsubmit.job.in*).

Vediamo ora come funziona realmente lo script nei 2 casi di maggior interesse analizzando le fasi che lo compongono:

Semplice esecuzione senza parametri:

1. Raccolta di tutti i parametri dai vari file xml definiti dall'utente;
2. creazione di tutte le cartelle necessarie all'esecuzione;
3. creazione di una cartella temporanea (con un numero diverso identificativo di ogni esecuzione) all'interno della quale effettuare tutte le operazioni (questo è necessario per evitare danni all'intera suite, ma anche perché in questo modo si possono elaborare i risultati separatamente anche dopo l'esecuzione).
   La cartella verrà creata all'interno della directory tmp che è possibile trovare in ogni modello di benchmark utilizzato (nel modello ECHAM5, ad esempio, il percorso per trovare tale cartella è applications/ECHAM5/tmp);
4. modifica dei file sorgente in base ai parametri acquisiti al passo 1;
5. compilazione dei file sorgente;
6. creazione di tutte le cartelle e i file necessari all'esecuzione;
7. creazione del file eseguibile con estensione .exe (l'estensione è ininfluente sotto sistemi unix ed invece è necessaria sotto sistemi windows, di conseguenza si è preferito definirla staticamente evitando inutili controlli e rallentamenti dell'intero sistema);
8. Invio dei comandi allo scheduler di riferimento per l'esecuzione dei job (le operazioni che verranno eseguite sono raccolte in un file con estensione

.job.in, mentre l'istruzione che lancia il job è indicata nel campo *command* del file execute.xml);

9. Termine dell'applicazione.

Esecuzione con parametri: -update –result

1. Raccolta di tutti i parametri dai vari file xml definiti dall'utente;
2. Analisi dei file di log generati dai job eseguiti (la cartella in cui vengono salvati i file di log è "logs". Nel caso di ECHAM5, ad esempio, la cartella può essere trovata seguendo il seguente percorso: applications/ECHAM5/logs);
3. Creazione di un file con i risultati (salvato nella cartella results con estensione .dat; nel caso di ECHAM5, ad esempio, la cartella può essere trovata seguendo il seguente percorso: applications/ECHAM5/results);
4. Stampa dei risultati a video.

Nel proseguo analizzeremo prima il modo in cui lo script può essere lanciato e successivamente riporteremo un caso di studio applicato ad un'architettura IBM Power6.

3.4 Parametri accettati dallo script

Lo script può essere richiamato tramite il seguente comando:

perl <cartella contenente lo script>/jube <options> <xml-file> <id-range>

Il file xml (xml-file) è obbligatorio quando si deve lanciare il benchmark, mentre non lo è quando il modello è già stato lanciato ma non si sono ancora elaborati i risultati.
L'id è invece indispensabile quando si vogliono estrarre i risultati in seguito all'esecuzione del modello.

Le opzioni previste per lo script sono le seguenti:

***start, submit** ** : per avviare un nuovo set di benchmark (definiti in xml file passato come parametro);

update + : Ricerca di risultati di jobs terminati;

result + : Visualizzazione dei risultati ottenuti dall'esecuzione dei benchmark (tabelle);

force + : Forzare una nuova ricerca di files di output per l'elaborazione di nuovi risultati;

cdir <dir> : cartella contenente i file xml necessari alla suite (default: ./);

pdir <dir> : cartella contenente il file XML di definizione della piattaforma (default: ../platforms);

tmpdir <dir> : cartella usata per eseguire i job; Usare solo path assoluti (default: tmp contenuta nella cartella di benchmark);

verbose level : verbose;

dump : scaricare la struttura del file XML;

showall : visualizzare tutti i risultati, compresi quelli di fallimento o quelli ancora in coda di esecuzione;

debug : non sottomettere i job;

rmtmp : rimuovere direttamente la cartella temporanea;

cmpdir <dir> : cartella usata per avviare la fase di compilazione; Usare solo path assoluti;

Version : Stampa la versione corrente dello script;

* : richiede XML top level file <xml-file>
+ : può essere specificato un range di ID di benchmark (<id-range>)

3.5 Esempio di configurazione ed esecuzione (IBM POWER 6)

Di seguito verrà illustrata la procedura utilizzata nell'ambito del tirocinio svolto presso il CMCC (Centro EuroMediterraneo per i Cambiamenti Climatici), seguita per l'esecuzione di ECHAM5 su piattaforma IBM POWER 6.

Fase 1: Spacchettamento del modello di benchmark (ECHAM5) all'interno della cartella "applications", appositamente creata per contenere i vari modelli di benchmark da utilizzare.

Fase 2: Configurazione dei file di piattaforma:

- Prendiamo il file platform.xml memorizzato nella cartella platform;
- Creiamo, alla fine del file, prima del tag di chiusura </platforms>, lo spazio per la nostra architettura, racchiudendolo tra i tag `<platform name="IBM-P6-calypso">` e `</platform>`;
- Cerchiamo nel file la sezione dedicata all'architettura più simile alla nostra (IBM P6 vip) e copiamo l'intero set di parametri (quelli compresi tra `<platform name="IBM-P6-vip">` e `</platform>`) all'interno dello spazio da noi creato in precedenza alla fine del file;
- Settiamo tutti i parametri necessari in questo modo:

```
    <platform name="IBM-P6-calypso">
            <params
                make            = "gmake"
                rm              = "rm -f"
                ar              = "ar"
                arflags         = "-rs"
                ranlib          = "/usr/bin/ranlib"

                cpp             = "/usr/lib/cpp"
                cppflags        = "-P"

                f77             = "xlf_r"
                f77flags        = "-qtune=pwr6 -qarch=pwr6"

                f90             = "xlf90_r"
                f90flags        = "-qtune=pwr6 -qarch=pwr6"

                cc              = "xlc_r"
                cflags          = "-qtune=pwr6 -qarch=pwr6"

                cxx             = "xlC_r"
                cxxflags        = "-qtune=pwr6 -qarch=pwr6"

                mpi_f90         = "mpxlf90_r"
                mpi_f77         = "mpxlf_r"
                mpi_cc          = "mpcc_r"
                mpi_cxx         = "mpCC_r"

                ldflags         = "-qtune=pwr6 -qarch=pwr6"

                mpi_dir         = ""
                mpi_lib         = ""
                mpi_inc         = ""
                mpi_bin         = ""

                blas_dir        = ""
```

```
                blas_lib        = "-lessl"

                lapack_dir      = "-L/usr/local/lib"
                lapack_lib      = "-llapack -lessl"

                fftw3_dir       = "-L/usr/local/lib"
                fftw3_lib       = "-lfftw3 -lfftw3_threads -lm"
                fftw3_inc       = "-I/usr/local/include"

                fftw2_dir       = "-L/usr/local/lib"
                fftw2_lib       = "-ldfftw -ldrfftw -ldfftw_threads -
                                  ldrfftw_threads    -ldfftw_mpi    -
                                  ldrfftw_mpi -lm"
                fftw2_inc       = "-I/usr/local/include"

                netcdf3_dir     = "-L/usr/local/netcdf-3.6.3/lib"
                netcdf3_lib     = "-lnetcdf"
                netcdf3_inc     = "-I/usr/local/netcdf-3.6.3/include"

                hdf5_dir        = "-L/usr/local/lib"
                hdf5_lib        = "-lhdf5_64 -lz"
                hdf5_inc        = "-I/usr/localinclude"

                module_cmd      = "module load"
            />
        </platform>
```

Fase 3: Creazione del **file di definizione delle istruzioni** che dovranno essere **eseguite dallo scheduler** per la gestione del job.

- All'interno della cartella platform, cerchiamo la sottocartella associata alla piattaforma più vicina (in fatto di hardware) alla nostra (IBM-P6_CINECA), la copiamo e la rinominiamo con il nome dato prima alla piattaforma nel file xml (`IBM-P6-calypso`).
- Modifichiamo il file (.job.in) presente all'interno della cartella (il nome è ininfluente in quanto dovrà essere successivamente dichiarato all'interno del file execute.xml, pertanto possiamo lasciare quello che c'è) inserendo tutti i parametri relativi alla nostra architettura;

  Facciamo però attenzione:

  Anche questo file è parametrico: i parametri preceduti dal # sono tutti parametri definiti nel file execute.xml (nella sezione *<substitute infile="ibm_llsubmit.job.in" outfile="ibm_llsubmit.job">*) che verranno sostituiti

al momento della compilazione e creazione dei file con estensione .job, pertanto non vanno modificati.

La piattaforma utilizzata dal nostro scheduler è LSF, di conseguenza il file *ibm_llsubmit.job.in* conterrà i seguenti parametri:

```
#!/bin/bash
#BSUB -J #BENCHNAME#
#BSUB -B
#BSUB -N
#BSUB -n #NCPUS#
#BSUB -q poe_short
#BSUB -a poe
#BSUB -o #STDOUTLOGFILE#
#BSUB -e #STDERRLOGFILE#
#BSUB -R "span[ptile=64]"

#MODULE_INIT#
module purge
module load profile/advanced
#MODULE_CMD# #MODULE_FILES#

export LSF_PAM_HOSTLIST_USE=#NODERESERVATION#
export OMP_NUM_THREADS=#THREADSPERTASK#

echo "<jobstart at=\"`date`\" />" >> #OUTDIR#/start_info.xml
#PREPROCESS#
#STARTER#      #ARGS_STARTER#      #MEASUREMENT#      #EXECUTABLE#
#ARGS_EXECUTABLE#
#POSTPROCESS#
echo "<jobend at=\"`date`\" />" >> #OUTDIR#/end_info.xml
```

Terminata la fase 3, si è conclusa la personalizzazione dei file relativi alla piattaforma, e di conseguenza, da questo punto in poi, ci si preoccuperà esclusivamente della modifica dei file appartenenti al modello: indicheremo quali sono i file di piattaforma prima dichiarati e pochi altri parametri necessari allo script per interfacciarsi con il modello in questione.

Le fasi successive sono relative alla configurazione dei file appartenenti al modello ECHAM5.

Fase 4: Adattamento del file **compile.xml**

- Apriamo il file compile.xml presente nella cartella applications/ECHAM5/ ;
- Al termine del file (prima del tag di chiusura *</compilation>*) creiamo un'area riservata alla nostra architettura, racchiudendola tra i tag: *<compile cname="IBM-P6-calypso"> e </compile>*;

  ATTENZIONE: Utilizziamo lo stesso nome dichiarato in *platform.xml*;

- Cerchiamo nel file la solita sezione dedicata alla piattaforma simile alla nostra (*IBM-P6*) e se non la troviamo ne prendiamo un'altra qualsiasi (è di poco conto perché, salvo particolari casi, è lo script che si preoccuperà di sostituire i parametri usando quelli dichiarati nel file platform.xml; I parametri da non modificare sono identificabili tramite il carattere $ che precede il nome di una variabile dichiarata in *platform.xml*);
- All'interno della nostra area, copiamo il contenuto della sezione trovata e modifichiamo i parametri in questo modo:

I parametri da modificare sono:

- Tutti quelli non dichiarati dinamicamente (manca il diretto riferimento a variabili dichiarate nel file platform.xml ($variabile));
- Tutto ciò che è contenuto tra il tag <*command*> e </*command*> : qui è necessario indicare i comandi che lo script dovrà utilizzare per la compilazione (ad esempio `configure; gmake -f Makefile`).

```
<compile cname="IBM-P6-calypso">
    <!-- Specification of source files to copy into temporary
    build directory -->
    <src directory="./src/echam-5.4-pre" files="*" />
    <substitute infile="config/mh-nbench.in" outfile="config/mh-
    nbench">
        <sub from="#ARCH#"          to="ibm_power4" />
        <sub from="#MAKE#"          to="$make" />
        <sub from="#CPP#"           to="$cpp" />
        <sub from="#CPPFLAGS#"      to="$cppflags" />
```

```
        <sub from="#F77#"           to="$f77" />
        <sub from="#FFLAGS#"        to="-q64 -O3 $f77flags" />
        <sub from="#F90#"           to="$f90" />
        <sub from="#F90FLAGS#"      to="-q64 -O3 -qstrict –qMAXMEM
                                    =-1     $f90flags     -Q     -
                                    qfloat=fltint  -qsuffix=cpp=f90
                                    -qzerosize     -qessl     -WF,-
                                    D__ibm__                  -WF,-
                                    DHAVE_LIBNETCDF64             -
                                    brename:.flush,.flush_" />
        <sub from="#CC#"            to="$cc" />
        <sub from="#CFLAGS#"        to="-q64 $cflags" />
        <sub from="#CXX#"           to="$cxx" />
        <sub from="#CXXFLAGS#"      to="$cxxflags" />
        <sub from="#MPI_F90#"       to="$mpi_f90" />
        <sub from="#MPI_F77#"       to="$mpi_f77" />
        <sub from="#MPI_CC#"        to="$cc" />
        <sub from="#MPI_CXX#"       to="$mpi_cxx" />
        <sub from="#MPI_DIR#"       to="$mpi_dir" />
        <sub from="#MPI_LIB#"       to="$mpi_lib" />
        <sub from="#MPI_INC#"       to="$mpi_inc" />
        <sub from="#MPI_BIN#"       to="$mpi_bin" />
        <sub from="#BLAS_DIR#"      to="$blas_dir" />
        <sub from="#BLAS_LIB#"      to="$blas_lib" />
        <sub from="#LAPACK_DIR#"    to="$lapack_lib" />
        <sub from="#LAPACK_LIB#"    to="$lapack_dir" />
        <sub from="#NETCDF_DIR#"    to="$netcdf3_dir" />
        <sub from="#NETCDF_LIB#"    to="$netcdf3_lib" />
        <sub from="#NETCDF_INC#"    to="$netcdf3_inc" />
    </substitute>
    <substitute infile="Makefile.in.nbench" outfile="Makefile.in">
        <sub from="#EXECNAME#"      to="$execname" />
        <sub from="#BLAS_DIR#"      to="$blas_dir" />
        <sub from="#BLAS_LIB#"      to="$blas_lib" />
        <sub from="#LAPACK_DIR#"    to="$lapack_lib" />
        <sub from="#LAPACK_LIB#"    to="$lapack_dir" />
        <sub from="#NETCDF_LIB#"    to="$netcdf3_dir $netcdf3_lib"
                                    />
        <sub from="#NETCDF_INCLUDE#"   to="$netcdf3_inc" />
        <sub from="#OTHER_LIBS#"    to=" " />
        <sub from="#F90FLAGS#"      to="-q64 -O3 -qstrict –qMAXMEM
                                    =-1     $f90flags     -Q     -
                                    qfloat=fltint  -qsuffix=cpp=f90
                                    -qzerosize     -qessl     -WF,-
                                    D__ibm__                  -WF,-
                                    DHAVE_LIBNETCDF64             -
                                    brename:.flush,.flush_" />
        <sub from="#LDFLAGS#"       to="-q64 -O3 -qstrict –qMAXMEM
                                    =-1     $f90flags     -Q     -
                                    qfloat=fltint  -qsuffix=cpp=f90
                                    -qzerosize       -qessl       -
                                    brename:.flush,.flush_" />
    </substitute>
    <!-- issue build command -->
    <command>configure; gmake -f Makefile</command>
</compile>
```

ERRORI RISCONTRATI:

Nella fase di compilazione è stato riscontrato il seguente errore:

```
Arguments of the wrong type were specified for the
INTRINSIC procedure "max".
```

Per risolvere tale problema, relativo a tutte le funzioni INTRINSECHE del Perl, è bastato passare il seguente parametro all'istruzione di compilazione:

```
-qrealsize=8
```

Per maggiori dettagli si faccia riferimento a [15].

Fase 5: Adattamento del file **execute.xml**

- Apriamo il file execute.xml presente nella cartella applications/ECHAM5;
- Al termine del file (prima del tag di chiusura *</execution>*) creiamo un'area riservata alla nostra architettura racchiudendola tra i tag: *<execute cname="IBM-P6-calypso">* e *</execute>*;

  Facciamo **ATTENZIONE**: Utilizziamo lo stesso nome dichiarato in platform.xml (*IBM-P6-calypso*);
- Cerchiamo nel file la solita sezione dedicata alla piattaforma simile alla nostra (*IBM-P6*) e se non la troviamo ne prendiamo un'altra qualsiasi (è di poco conto perché, salvo particolari casi, è lo script che si preoccuperà di sostituire i parametri usando quelli dichiarati nel file platform.xml;

  I parametri da non modificare sono identificabili tramite il carattere $ che precede il nome di una variabile dichiarata in platform.xml);
- All'interno della nostra area, copiamo il contenuto della sezione trovata e modifichiamo i parametri in questo modo:

```
<input    files="../../platform/IBM-P6-calypso/ibm_llsubmit.job.in
          job/namelist.echam.in" />
<substitute infile="namelist.echam.in" outfile="namelist.echam">
        <sub from="#NPROCA#"            to="$nproca" />
        <sub from="#NPROCB#"            to="$nprocb" />
        <sub from="#NPROMA#"            to="$nproma" />
        <sub from="#WORK_DIR#"          to="$outdir" />
        <sub from="#NMONTHS#"           to="$nmonths" />
    </substitute>
<substitute infile="ibm_llsubmit.job.in" outfile="ibm_llsubmit.job
           ">
        <sub from="#SHELL#"             to="/usr/bin/ksh" />
        <sub from="#OUTDIR#"            to="$outdir" />
        <sub from="#STDOUTLOGFILE#"     to="$stdoutlogfile" />
        <sub from="#STDERRLOGFILE#"     to="$stderrlogfile" />
        <sub from="#CLASS#"             to="bench" />
        <sub from="#BENCHNAME#"         to="$benchname-
                                        `$nodes*$taskspernode`"
                                        />
        <sub from="#NODEUSAGE#"         to="not_shared" />
        <sub from="#TIME_LIMIT#"        to="04:00" />
        <sub from="#NODES#"             to="$nodes" />
        <sub from="#TASKSPERNODE#"      to="$taskspernode" />
        <sub from="#NOTIFICATION#"      to="never" />
        <sub from="#NOTIFY_EMAIL#"      to="user@host.com" />
        <sub from="#THREADSPERTASK#"    to="$threadspertask" />
        <sub from="#DATA_LIMIT#"        to="1.5GB" />
        <sub from="#STACK_LIMIT#"       to="1.5GB" />
        <sub from="#MEMORYPERTASK#"     to="3GB" />
        <sub from="#EXECUTABLE#"        to="$executable" />
        <sub from="#ENV#"               to="$env" />
        <sub from="#PREPROCESS#"        to="cd ./bigtmp" />
        <sub from="#POSTPROCESS#"       to="" />
        <sub from="#STARTER#"           to="mpirun.lsf " />
        <sub from="#ARGS_STARTER#"      to="" />
        <sub from="#MEASUREMENT#"       to="time hpmcount -a -k "
                                        />
        <sub from="#ARGS_EXECUTABLE#"   to="" />
        <sub from="#NPROCB#"     to="`$taskspernode*$nodes/1`" />
        <sub from="#NPROCA#"            to="1" />
        <sub from="#NCPUS#"             to= "`$threadspertask*
                                        $taskspernode* $nodes`"
                                        />
        <sub from="#NMONTHS#"            to="$nmonths" />
    </substitute>
    <environment>
        <env var="MP_LABELIO" value="yes" />
        <env var="MP_INFOLEVEL" value="2" />
        <env var="MP_SHARED_MEMORY" value="yes" />
        <env var="MP_TASK_AFFINITY" value="MCM" />
        <env var="MEMORY_AFFINITY"  value="MCM" />
        <env var="MP_SINGLE_THREAD" value="yes" />
    </environment>
    <command>bsub &lt; ibm_llsubmit.job</command>
</execute>
```

Qui è stato importante definire bene tutti i parametri:

- Nel tag `<input>` è stato importante dichiarare il nome del file con estensione `.job.in` precedentemente creato nella cartella `platform/IBM-P6-calypso` (*IBM-P6-calypso* è la sottocartella creata in platform nella fase 3);

  Abbiamo fatto **ATTENZIONE** a non cancellare la parte relativa al file *`job/namelist.echam.in`* : noi non dobbiamo toccarlo, ma è indispensabile per il corretto funzionamento dello script;

- Nel tag <*command*> abbiamo definito il comando necessario a richiamare lo scheduler;
  Nel nostro caso, ad esempio, la piattaforma utilizzata per richiamare lo scheduler è LSF ed il comando utilizzato è bsub (il comando va dichiarato assieme al file nel quale sono dichiarate le istruzioni che dovrà eseguire lo scheduler per lanciare i job: *`ibm_llsubmit.job`*).

Fase 6: Modifica di tutti i file utilizzati per la formattazione dell'output (per tutti i dettagli si veda l'**appendice A**)

Fase 7: Creazione e modifica del file nel quale dichiarare i set di benchmark da eseguire

- Creiamo, all'interno della cartella applications/ECHAM5, un file con il nome formato da:

  bench-

  IBM-P6-calypso (nome dell'architettura dichiarato nel file platform.xml)

  .xml

  Quindi il nostro file si chiamerà *bench-IBM-P6-calypso.xml*;

- Cerchiamo il file associato all'architettura più simile alla nostra (*bench-IBM-P6.xml*);
- Lo apriamo e copiamo tutto il suo contenuto;
- Incolliamo il contenuto copiato, all'interno del file da noi creato (*bench-IBM-P6-calypso.xml*) e lo modifichiamo in questo modo:

```
<bench name    = "ECHAM5" platform= "IBM-P6-calypso" >
<benchmark name="echam5-T63-64" active="1">
    <compile     cname="$platform" version="reuse" />
    <tasks       threadspertask="1" taskspernode="64" nodes="1" />
    <params      nproca="8" nprocb="8" nproma="45" res="63"
                  levels="31" nmonths="06"/>
    <prepare     cname="ECHAM5" />
    <execution   iteration="1" cname="$platform" />
    <verify      cname="ECHAM5" />
    <analyse     cname="$platform" />
</benchmark>
<benchmark name="echam5-T63-256" active="1">
    <compile     cname="$platform" version="reuse" />
    <tasks       threadspertask="1" taskspernode="16" nodes="16"
                  />
    <params      nproca="16" nprocb="16" nproma="45" res="63"
                  levels="31" nmonths="06"/>
    <prepare     cname="ECHAM5" />
    <execution   iteration="1" cname="$platform" />
    <verify      cname="ECHAM5" />
    <analyse     cname="$platform" />
</benchmark>
<benchmark name="echam5-T63-496" active="1">
    <compile     cname="$platform" version="reuse" />
    <tasks       threadspertask="1" taskspernode="31" nodes="16"
                  />
    <params      nproca="16" nprocb="31" res="63" levels="31"
                  nmonths="06"/>
    <prepare     cname="ECHAM5" />
    <execution   iteration="1" cname="$platform" />
    <verify      cname="ECHAM5" />
    <analyse     cname="$platform" />
</benchmark>
</bench>
```

- Creiamo una sezione per ogni prova da effettuare (con un numero di processori o di iterazioni diverse).

All'interno di questa sezione ci saranno diversi tag:

- Lasciamo tutto invariato tranne i campi "*nproca*", "*nprocb*", "*res*", "*taskspernode*", "*threadspertask*", "*nodes*", facendo attenzione che il

prodotto tra *nproca* e *nprocb* sia sempre uguale al prodotto tra *threadspertask*, *tasks*pernode e *nodes*;

- Il campo "*version*" serve a non rieffettuare la compilazione ogni volta. In sostanza la prima volta che si esegue il benchmark è bene settarlo a "*new*", così verrà effettuata la compilazione di tutti i file, mentre nelle successive prove è bene settarlo a "*reuse*", in modo che riutilizzi l'eseguibile già creato e non ci sia bisogno di attendere tutto il tempo necessario alla compilazione;
- Il parametro "*active*" è utilizzato in questo modo:
    - Se è settato a *0* il benchmark definito viene saltato e non eseguito;
    - Se è settato a *1* il benchmark definito viene eseguito.

Fase 8: Esecuzione

L'esecuzione è stata ottenuta portandosi nella cartella applications/ECHAM5 e lanciando il seguente comando:

perl ../../bench/jube bench-IBM-P6-calypso.xml

Al termine dell'esecuzione lo script ha restituito l'ID dei processi lanciati dallo scheduler come jobs. E' stato necessario attendere la fine dell'esecuzione di tali jobs per poi lanciare il comando:

perl ../../bench/jube –update –result <ID>

ed ottenere i risultati (per verificare che fosse terminata l'esecuzione dei job, la nostra architettura permetteva di saperlo tramite il comando bjobs).

I risultati, oltre ad essere stampati sullo schermo sono stati stampati in un file salvato nella cartella *application/ECHAM5/results* con il nome composto da:

ECHAM5_
IBM-P6-calypso (nome della piattaforma)

_

echam5-T63-64 (nome associato nel file bench-IBM-P6-calypso.xml al set di benchmark)

_i

<ID>

.dat

Quindi i risultati dell'esecuzione del modello di benchmark lanciato tramite il seguente set di benchmark:

```
<benchmark name="echam5-T63-496" active="1">
    <compile     cname="$platform" version="reuse" />
    <tasks       threadspertask="1" taskspernode="31" nodes="16"
                 />
    <params      nproca="16" nprocb="31" res="63" levels="31"
                  nmonths="06"/>
    <prepare     cname="ECHAM5" />
    <execution   iteration="1" cname="$platform" />
    <verify      cname="ECHAM5" />
    <analyse     cname="$platform" />
</benchmark>
```

saranno salvati nel file

ECHAM5_IBM-P6-calypso_echam5-T63-496_i33.dat

dove 33 è l'ID associato dallo script alla particolare esecuzione ed è stato restituito a video in seguito alla compilazione.

4 CASO DI STUDIO: ARCHITETTURA POWER6

Nell'ambito di questo lavoro di tesi, è stato eseguito lo script su 2 diversi modelli di benchmark:

- ECHAM5;
- NEMO.

L'architettura sulla quale sono stati eseguiti è IBM P575 Power 6. Si tratta di un'architettura IBM P6-575 Infiniband Cluster, dotata di processore IBM Power6 a 4.7 GHz (composto da 960 Computing Cores e 30 Computing Nodes) e Ram da 21 TB (128 GB/nodo). Lo spazio disponibile su Disco è 1,2 PB e l'architettura è dotata di una rete interna "Infiniband x4 DDR". Il sistema operativo è AIX 6 e le performance di picco sono dell'ordine di 101 TFlop/s. I compilatori disponibili sono Fortran90, C, C++, mentre le librerie parallele disponibili sono MPI, OpenMP, LAPI.

Nei paragrafi seguenti vengono riportati i risultati ottenuti, le relative considerazioni, ed infine la media dei risultati restituita in seguito all'esecuzione della suite realizzata.

4.1 Risultati ottenuti da ECHAM5

La tabella seguente contiene tutti i risultati di maggiore interesse relativi all'esecuzione della 5° versione del modello di benchmark denominato ECHAM sull'architettura IBM P575 Power 6 prima descritta.

ID	Numero processori	EXECUTION TIME [secondi] per simulare 6 mesi	SYPD [anni]	Memoria usata [KB*s]	Efficienza
1	**16**	4939,8	8,75	695495025	100,00%
2	**32**	3261,12	13,25	401563049	75,74%
3	**64**	2051,82	21,05	212652838	60,19%
4	**128**	1168,7	36,96	111533874	52,83%
5	**256**	694,23	62,23	91025083	44,47%
6	**496**	635,58	67,97	115134594	25,07%
7	**510**	703,15	61,44	128254086	22,04%
8	**513**	697,48	61,94	109840519	22,09%

Tabella dei risultati ottenuti dall'esecuzione di ECHAM5 su IBM P575 Power 6

SYPD massimo = 61,94 anni ottenuto con 513 processori

SYPD massimo (con efficienza >=0,3) = 62,23 anni ottenuto con 256 processori

Considerazioni sulla metrica

Il tempo di esecuzione riportato in tabella è relativo alla simulazione di 6 mesi di esecuzione del benchmark.
Si è scelto di simulare 6 mesi in quanto ritenuti un buon compromesso tra il tempo di esecuzione e la completezza del run.

Per il calcolo dell'efficienza relativa si è scelto come riferimento il tempo ottenuto con 16 processori, infatti nella prima riga si può notare un'efficienza pari al 100%.

Analisi dei risultati ottenuti:

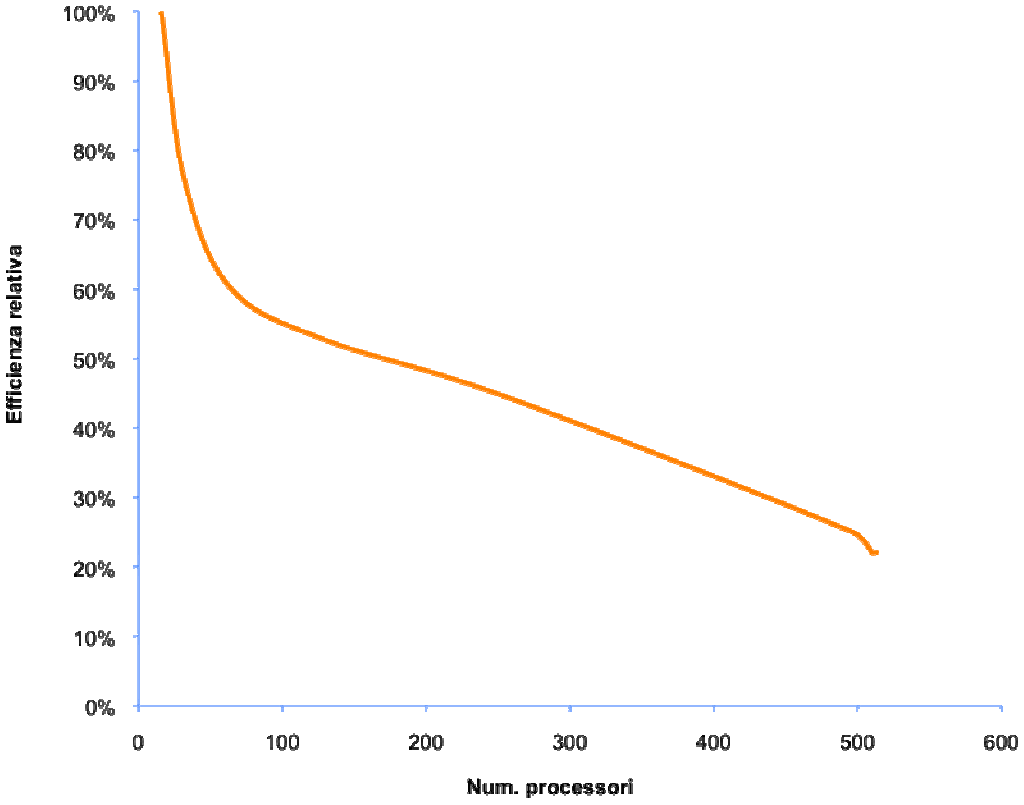

Grafico riportante il rapporto tra il numero di processori e l'efficienza relativa

Analizzando il grafico sopra raffigurato, riportante la variazione dell'efficienza relativa in funzione del numero di processori, si può subito notare quanto già riportato nel capitolo 2, ovvero che l'efficienza tende a diminuire man mano che si aumenta il numero di processori.

Tale decremento è causato dalla gestione del parallelismo: ogni volta che si eseguono istruzioni parallele, il sistema si deve preoccupare di distribuire il carico di lavoro tra i vari processori, e man mano che il numero di processori aumenta, il tempo necessario per espletare tale incarico aumenta sempre di più, a discapito dell'esecuzione del programma lanciato.

Per tale motivo, e come si può notare dal grafico riportato di seguito, anche se il numero di processori aumenta in maniera esponenziale, il tempo di esecuzione rimane pressappoco lo stesso dopo un certo numero di processori, tendendo addirittura a risalire dopo i 500 processori.

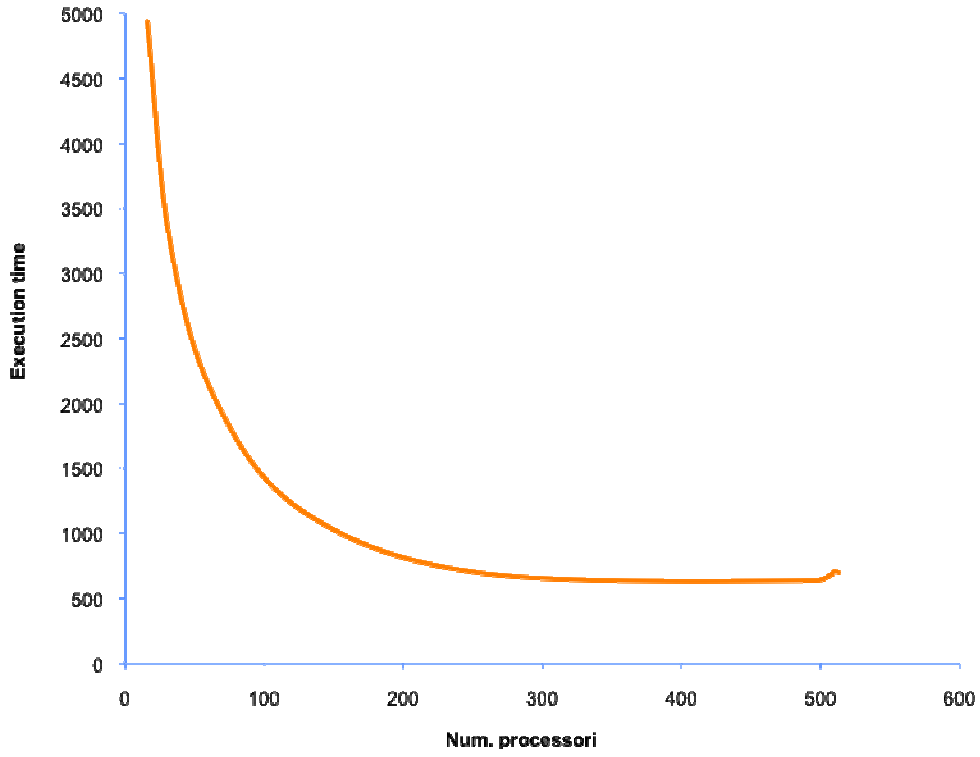

Grafico riportante il rapporto tra il numero di processori e il tempo di esecuzione (execution time)

Analizzando inoltre il consumo di memoria da parte della suite, riportato nel grafico seguente, si può notare che la memoria utilizzata per singolo processo diminuisce all'aumentare del numero di processori; il motivo principale di tale decremento è la scomposizione delle variabili temporanee che vengono distribuite tra i vari nodi con l'aumento del numero di processori.

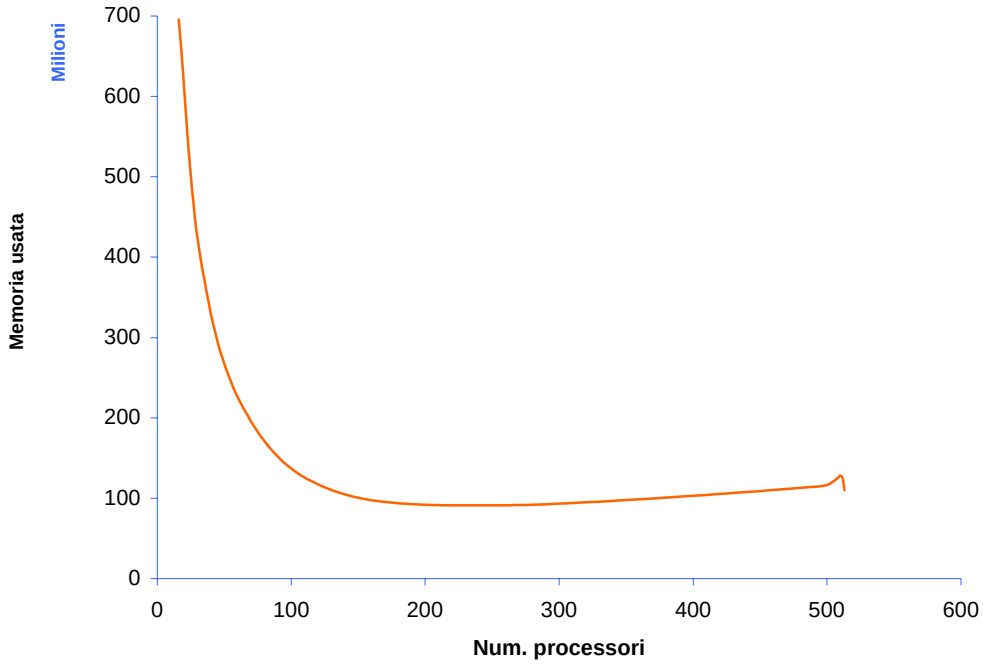

Grafico riportante il rapporto tra il numero di processori e la memoria utilizzata

Tale effetto (di diminuzione della memoria occupata), però, ha un limite: i vari processi condividono una parte di dominio (detta di overlap) che non può essere divisa nella scomposizione; inoltre esistono alcune variabili temporanee, oltre a quelle già mensionate, non dipendenti dalla decomposizione.

Tali limiti hanno un immediato effetto nell'occupazione di spazio in memoria: come è possibile notare dal grafico, superando il limite dei 150 processori, la quantità di memoria utilizzata tende a rimanere costante.

4.2 Risultati ottenuti da NEMO

La tabella seguente contiene tutti i risultati di maggiore interesse relativi all'esecuzione del modello di benchmark denominato NEMO sull'architettura IBM P575 Power 6 descritta all'inizio di questo capitolo.

ID	Numero processori	EXECUTION TIME [secondi] per simulare 125 giorni	SYPD [anni]	Memoria usata [KB*s]	Efficienza
1	**16**	2455,95	12,05	1116089790	100,00%
2	**32**	1919,86	15,41	491966163	63,96%
3	**64**	1650,71	17,93	250183025	37,20%
4	**128**	658,87	44,91	72610214	46,59%
5	**256**	410,44	72,09	34261220	37,40%
6	**512**	330,9	89,42	41266850	23,19%
7	**1024**	288,6	102,93	33439205	13,30%

Tabella dei risultati ottenuti dall'esecuzione di NEMO su IBM P575 Power 6

SYPD massimo = 102,93 anni ottenuto con 1024 processori

SYPD massimo (con efficienza >=0,3) = 72,09 anni ottenuto con 256 processori

Considerazioni sulla metrica

Il tempo di esecuzione riportato è relativo alla simulazione di 125 giorni di esecuzione del benchmark.

Si è scelto di simulare 125 giorni in quanto ritenuti un buon compromesso tra il tempo di esecuzione e la completezza del run.

Per il calcolo dell'efficienza relativa si è scelto come riferimento il tempo ottenuto con 16 processori, infatti nella prima riga si può notare un'efficienza pari al 100%.

Analisi dei risultati ottenuti:

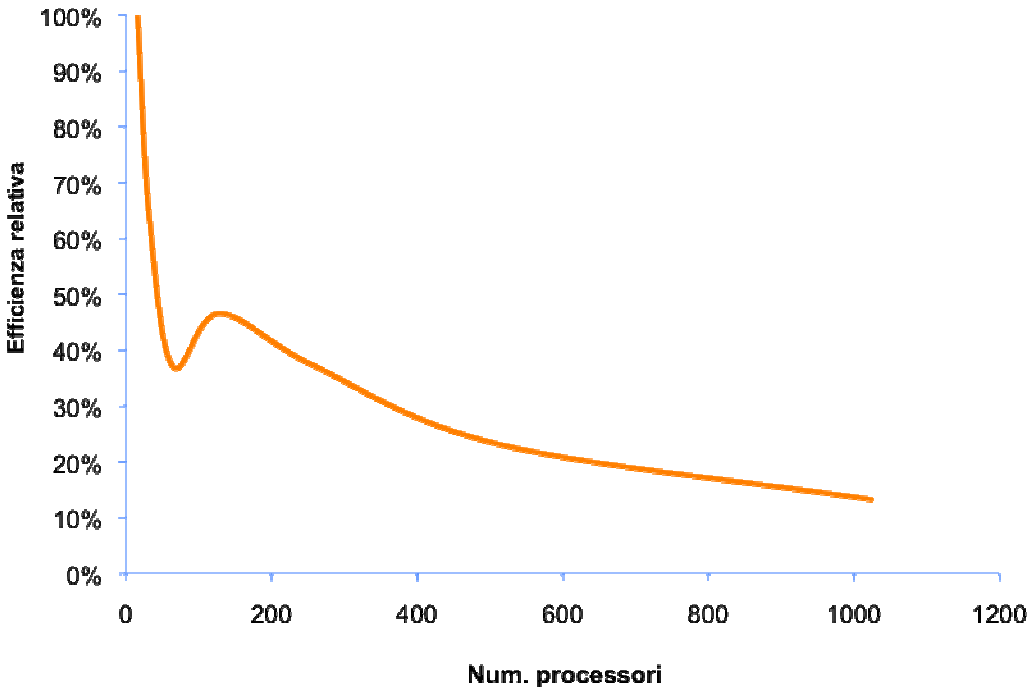

Grafico riportante il rapporto tra il numero di processori e l'efficienza relativa

Anche per il caso di NEMO è possibile fare le stesse considerazioni del caso precedente:

Analizzando il grafico sopra raffigurato, riportante la variazione dell'efficienza relativa in funzione del numero di processori, si può subito notare quanto già riportato nel capitolo 2, ovvero che l'efficienza tende a diminuire man mano che si aumenta il numero di processori.

Tale decremento è causato dalla gestione del parallelismo: ogni volta che si eseguono istruzioni parallele, il sistema si deve preoccupare di distribuire il carico di lavoro tra i vari processori, e man mano che il numero di processori aumenta, il tempo necessario

per espletare tale incarico aumenta sempre di più, a discapito dell'esecuzione del programma lanciato.

Per tale motivo, e come si può notare dal grafico riportato di seguito, anche se il numero di processori aumenta in maniera esponenziale, il tempo di esecuzione rimane pressappoco lo stesso dopo un certo numero di processori.

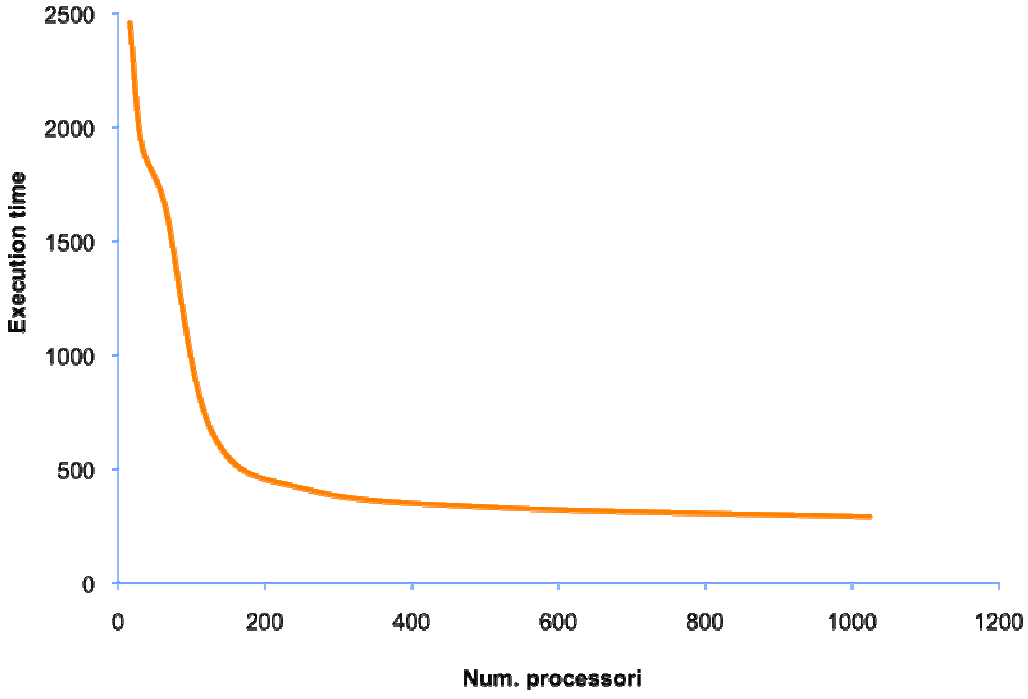

Grafico riportante il rapporto tra il numero di processori e il tempo di esecuzione (execution time)

Analizzando inoltre il consumo di memoria da parte della suite, riportato nel grafico seguente, si può notare che la memoria utilizzata per singolo processo diminuisce all'aumentare del numero di processori; il motivo principale di tale decremento è la scomposizione delle variabili temporanee che vengono distribuite tra i vari nodi con l'aumento del numero di processori.

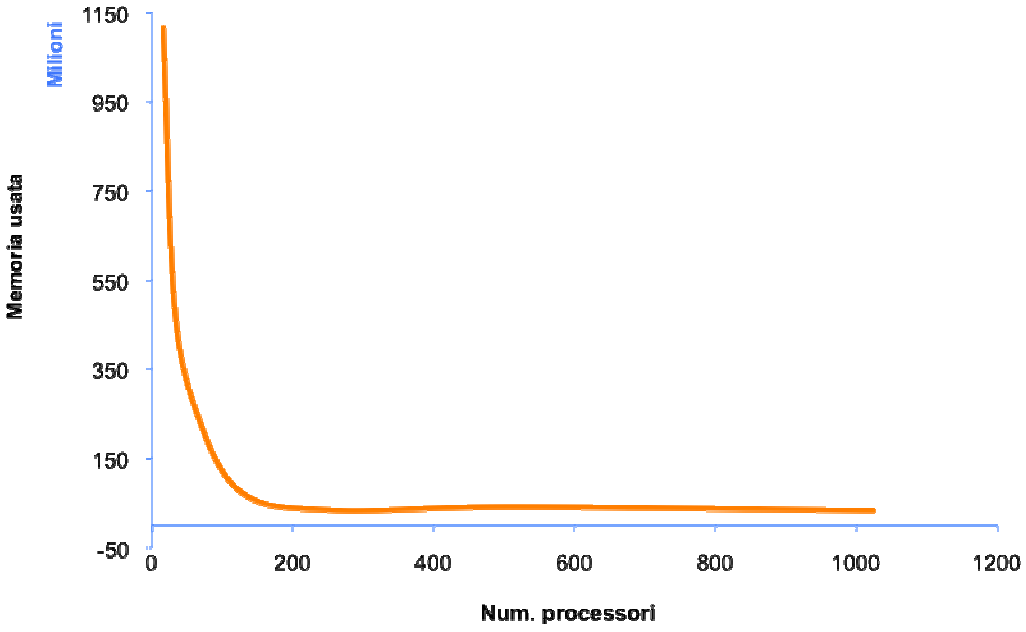

Grafico riportante il rapporto tra il numero di processori e la memoria utilizzata

Tale effetto (di diminuzione della memoria occupata), però, ha un limite: i vari processi condividono una parte di dominio (detta di overlap) che non può essere divisa nella scomposizione; inoltre esistono alcune variabili temporanee, oltre a quelle già mensionate, non dipendenti dalla decomposizione. Tali limiti hanno un immediato effetto nell'occupazione di spazio in memoria: come è possibile notare dal grafico, superando il limite dei 200 processori, la quantità di memoria utilizzata tende a rimanere costante.

4.3 Risultati restituiti dalla suite

Dopo aver eseguito i modelli singolarmente (lo si è fatto per poter valutare in maniera opportuna i risultati ottenuti dalle singole esecuzioni), abbiamo lanciato l'intera suite, ottenendo i seguenti ovvi risultati.

ID	Numero processori	EXECUTION TIME [secondi] per simulare 6 mesi	SYPD [anni]	Efficienza
1	**16**	4311,8625	10,02	100,00%
2	**32**	3070,455	14,07	70,22%
3	**64**	2263,9425	19,08	47,61%
4	**128**	1078,5025	40,06	49,98%
5	**256**	654,945	65,96	41,15%
6	**512**	596,915	72,37	22,57%

Tabella dei risultati ottenuti come media dei risultati ottenuti dall'esecuzione di ECHAM5 e NEMO su IBM P575 Power 6

Come già anticipato in precedenza, in questa tabella sono riportati i valori medi del tempo di esecuzione e dell'SYPD. Tali valori sono i valori che la suite di benchmark calcola come media dei valori ottenuti dai vari modelli. In questo caso, gli unici modelli utilizzati sono ECHAM5 e NEMO.

I valori dell'SYPD restituiti sono:

SYPD massimo = 72,37 anni ottenuto con 512 processori

SYPD massimo (con efficienza >=0,3) = 65,96 anni ottenuto con 256 processori

Considerazioni sulla metrica

Il tempo di esecuzione riportato in tabella è relativo alla simulazione di 6 mesi di esecuzione del benchmark.
Si è scelto di simulare 6 mesi in quanto ritenuti un buon compromesso tra il tempo di esecuzione e la completezza del run.

Per il calcolo dell'efficienza relativa si è scelto come riferimento il tempo ottenuto con 16 processori, infatti nella prima riga si può notare un'efficienza pari al 100%.

Analisi dei risultati ottenuti:

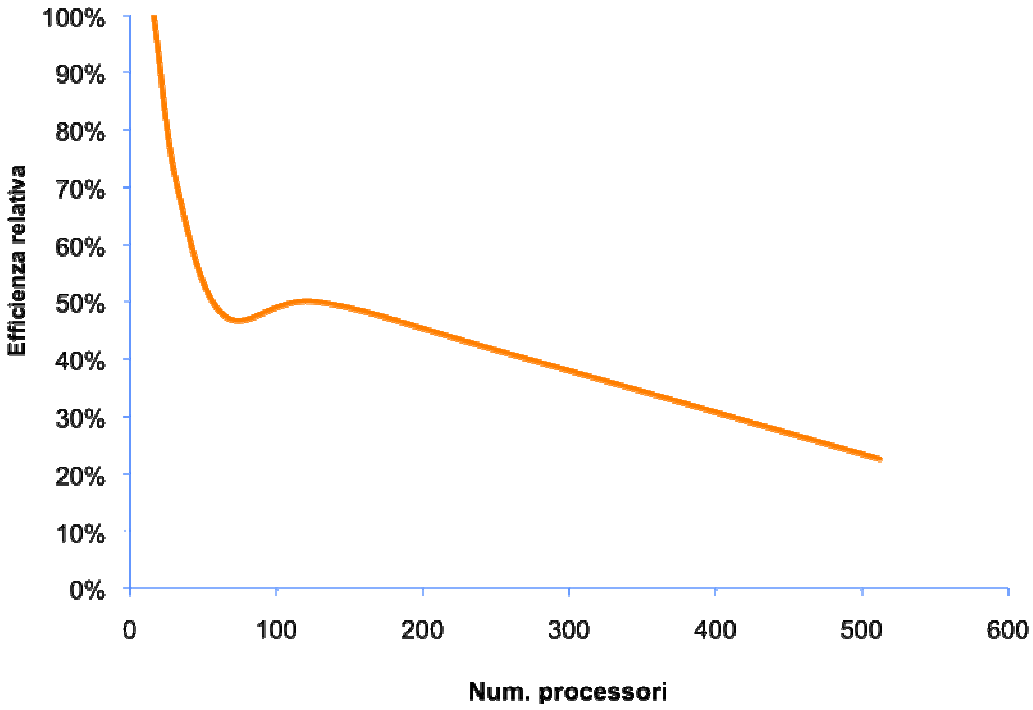

Grafico riportante il rapporto tra il numero di processori e l'efficienza relativa

Anche per il caso della media dei valori ottenuti dai vari modelli è possibile fare le stesse considerazioni fatte nei paragrafi precedenti:

Analizzando il grafico sopra raffigurato, riportante la variazione dell'efficienza relativa in funzione del numero di processori, si può subito notare che l'efficienza tende a diminuire man mano che si aumenta il numero di processori.

Inoltre, come si può notare dal grafico riportato di seguito, anche se il numero di processori aumenta in maniera esponenziale, il tempo di esecuzione rimane pressappoco lo stesso dopo un certo numero di processori.

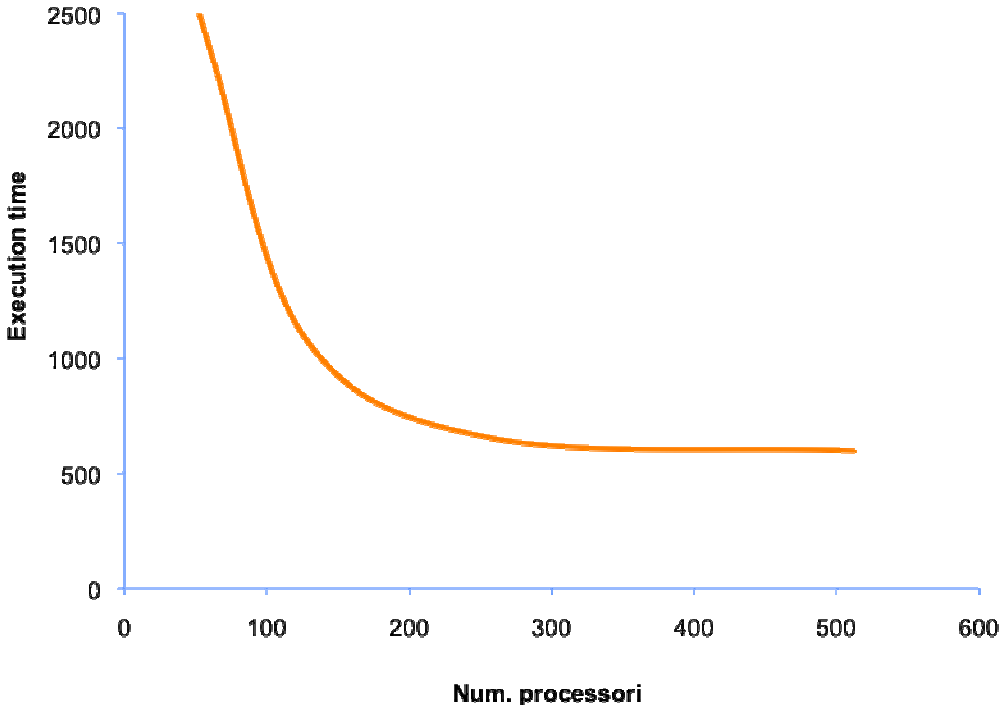

Grafico riportante il rapporto tra il numero di processori e il tempo di esecuzione (execution time)

APPENDICE A: Definizione dei parametri da stampare nei risultati

In questa sezione si illustrerà la procedura da seguire per definire quali risultati estrarre dai file di log generati in fase di esecuzione.

Tutte le informazioni ed i risultati ottenuti in fase di esecuzione della suite di benchmark (come ad esempio: tempo di esecuzione di ogni modello componente la suite, memoria occupata, ecc…) vengono salvati, man mano che si presentano, all'interno di 2 principali file di log.
Tali file è possibile trovarli all'interno della cartella "logs", sottocartella della directory dedicata al modello (ad esempio per ECHAM5 è possibile trovare la cartella seguendo questo percorso: applications/ECHAM5/logs).

Nell'ottica di rendere l'applicazione il più portabile e personalizzabile possibile e considerando il funzionamento dello script, che si preoccupa essenzialmente di estrapolare i risultati dai file di log già generati, anche la fase di raccolta dei risultati è stata resa parametrica tramite la creazione di opportuni file xml di definizione; in questo modo si è permesso agli utenti utilizzatori di scegliere quali valori visualizzare nella tabella finale dei risultati.

I file da modificare per la personalizzazione dei risultati sono i seguenti:

1. **result.xml** (che si trova all'interno della cartella principale dedicata al modello di benchmark in uso : applications/ECHAM5/);
2. **analyse.xml** (che si trova all'interno della cartella principale dedicata al modello di benchmark in uso : applications/ECHAM5/);
3. i **file indicati in analyse.xml** (nel nostro caso *analyse-pattern-echam5.xml* e *../../skel/hpm3patterns.xml*).

Proviamo ad analizzare un caso reale per comprendere bene come abbiamo fatto, ad esempio, ad estrapolare la quantità di memoria usata dal benchmark e l'SYPD.

Per prima cosa guardiamo il contenuto del file **analyse.xml**

```
<analyzer>

<!--
************************************************************************
************ -->
<!-- Input is stdout and stderr of benchmark run -->
<!-- Standard result parameter:
     - walltime
-->

   <analyse cname="IBM-SP4-Jump">
       <includepattern file="./analyse-pattern-echam5.xml" />
       <includepattern file="../../skel/hpm3patterns.xml" />
   </analyse>

   <analyse cname="Cray-XT4-Louhi">
       <includepattern file="./analyse-pattern-echam5.xml" />
       <includepattern file="../../skel/hpm3patterns.xml" />
   </analyse>

   <analyse cname="SGI-Altix-HLRB2">
       <includepattern file="./analyse-pattern-echam5.xml" />
       <includepattern file="../../skel/hpm3patterns.xml" />
   </analyse>

   <analyse cname="IBM-P6-calypso">
       <includepattern file="./analyse-pattern-echam5.xml" />
       <includepattern file="../../skel/hpm3patterns.xml" />
   </analyse>

</analyzer>
```

Come si può notare, esattamente come in tutti gli altri file xml già descritti nel capitolo 3, anche in questo file abbiamo una "sezione" dedicata ad ogni architettura, quindi ogni volta che si vuole utilizzare lo script su una architettura diversa è buona norma inserire una nuova sezione all'interno di questo file.

Nel nostro caso abbiamo aggiunto la sezione *<analyse cname="IBM-P6-calypso"*> all'interno della quale sono stati dichiarati i file da modificare per definire i valori da includere tra i risultati.

Nei campi <*includepattern*> sono stati indicati 2 file:

- **analyse-pattern-echam5.xml**
- **hpm3patterns.xml**

Il primo è un file specifico per il particolare modello di benchmark (ECHAM5), mentre il secondo è relativo all'intera piattaforma.
Ovviamente è stata una nostra scelta quella di prevedere 2 file diversi: uno specifico per il particolare modello e uno utilizzabile da tutti i modelli; di conseguenza si potrebbe tranquillamente dichiarare un unico file all'interno del quale riportare tutto, solo che così si dovrebbe ripetere l'operazione per tutti i modelli di benchmark da utilizzare.

All'interno di questi file indicati in *analyse.xml*, possiamo trovare la definizione di tutti i parametri da estrapolare dai file di log.

Nel proseguo cercheremo di spiegare il modo in cui è possibile definire un parametro in questi file di dichiarazione (*analyse-pattern-echam5.xml* e *hpm3patterns.xml*).

La struttura dei file è la seguente:
Il tag principale è <*patterns*> e al suo interno vengono definiti tutti i parametri da prelevare dai file di log tramite l'utilizzo del tag "*parm*".

Ecco un esempio di come sarà il file:

```
    <patterns>

        <parm name="walltime" unit="s"        mode="line"
                type="float">Wallclock\s*:\s*$patfp s</parm>
        <parm name="tdomain"   unit="s"        mode="line,add"
                type="float">Domains:\s+$patfp</parm>
        <parm name="tbuild"    unit="s"        mode="line,add"
                type="float">Build:\s+$patfp</parm>
```

```
        <parm name="tprefetch" unit="s"        mode="line,add"
                type="float">Prefetch:\s+$patfp</parm>
        <parm name="twalkser"  unit="s"        mode="line,add"
                type="float">Walk serial:\s+$patfp</parm>
        <parm name="twalkcomm" unit="s"        mode="line,add"
                type="float">Walk comm:\s+$patfp</parm>
        <parm name="tforces"   unit="s"        mode="line,add"
                type="float">Forces:\s+$patfp</parm>
        <parm name="tpusher"   unit="s"        mode="line,add"
                type="float">Pusher:\s+$patfp</parm>
        <parm name="tdiag"     unit="s"        mode="line,add"
                type="float">Diag:\s+$patfp</parm>
        <parm name="ttotal"    unit="s"        mode="line,add"
                type="float">Total:\s+$patfp</parm>
        <parm name="nel"      unit="#el"      mode="line" type="int"
                >\s+Electrons:\s+$patint.*$</parm>
        <parm name="nion"     unit="#ions"    mode="line" type="int"
                >\s+Ions:\s+$patint.*$</parm>
        <parm name="mem"  unit="KB"     mode="line,last"
                type="int">Unshared mem use in data
                seg.:\s+\d+\s+\d+\s+\d+\s+$patint</parm>
        <parm name="SYPD" unit="MFlops" mode="derived"
                type="float">7200*$nmonths/$walltime</parm>
        <parm name="NMONTHS" unit="MFlops" mode="derived"
                type="float">$nmonths</parm>
        <parm name="n"        unit="#"        mode="derived"
                type="int">$nel+$nion</parm>
        <parm name="np"       unit="#"        mode="derived"
                type="int">$nodes*$taskspernode*$threadspertask</par
                m>
    </patterns>
```

Cominciamo con l'indicare quali attributi possono essere definiti all'interno dei tag parm:

1. **name** -> indica il nome che verrà utilizzato nella tabella dei risultati per indicare la colonna con quel valore;
2. **unit** -> indica l'unità di misura utilizzata per rappresentare il risultato;

3. **type** -> indica il tipo di dato che verrà estrapolato; può essere:
    - int;
    - float;
    - qualsiasi dato previsto dal Perl;
4. **mode** -> indica il modo in cui può essere acquisito il risultato:
    - line
    - line,add
    - line,last
    - line,min
    - line,max
    - line,statistics
    - line,index()
    - derived

Per comprendere il significato di questi ultimi valori (assumibili dall'attributo mode), è necessario capire il modo in cui i dati vengono memorizzati all'interno dei file di logs.

Lo script, al momento della raccolta dei risultati, si preoccupa di andare a prelevare i risultati all'interno di 2 principali file di log:

/logs/……out.log

/logs/……err.log

(al posto dei puntini verranno sostituiti: il nome della suite di benchmark seguito dal nome del set di benchmark definito in bench-IBM-P6-calypso.xml e il numero ID identificativo della particolare esecuzione).

All'interno di questi file i risultati sono raggruppati in maniera diversa a seconda di ciò che rappresentano.

Proviamo ad esempio a guardare il file *ECHAM5_IBM-P6-calypso_echam5-T63-64_i000399.n1p64t1_t001_i01_stdout.log*

Tra le altre cose possiamo trovare le seguenti righe:

```
……….
Execution time (wall clock time): 2057.83913 seconds on 64 tasks

 ## Resource Usage Statistics -    Average      Total        MIN        MAX ##

 Wall Clock Time (in sec.)    : 2056.921911   131643.002331   2055.882272   2057.839130 s
 Time in user mode (in sec.)  : 1024.075489   65540.831287    1008.007182   1037.973266 s
 Time in system mode (in sec.):   0.667795      42.738883       0.005080      1.980373 s
 Maximum resident set size    :     66598         4262312          65204         109996 KB
 Shared mem use in text seg.  :    6789353       434518616        6778434      6793692 KB*s
 Unshared mem use in data seg.: 129575770    8292849294      126803516   214538337 KB*s
 Page faults w/out IO activity:      16191         1036272          15728          27105
 Page faults with IO activity :        166           10660             96            285
 Times process was swapped out:          0               0              0              0
 Times file system perf. INPUT:          0               0              0              0
 Times file system perf.OUTPUT:          0               0              0              0
 IPC messages sent            :          0               0              0              0
 IPC messages received        :          0               0              0              0
 signals delivered            :          0               0              0              0
 voluntary context switches   :      19020         1217319           9391          25623
 involuntary context switches :      12723          814300           7529          18701

 #######  End of Resource Statistics  ########

 Set: 1
 Counting duration: 2049.454582640 seconds
  PM_FPU_1FLOP (FPU executed one flop instruction )          :   20821241363044
  PM_FPU_FMA (FPU executed multiply-add instruction)         :   8932920839046
  PM_FPU_FSQRT_FDIV (FPU executed FSQRT or FDIV instruction) :   1980385588550
  PM_CYC (Processor cycles)                                  : 616972728689290
  PM_RUN_INST_CMPL (Run instructions completed)              : 228548520958867
  PM_RUN_CYC (Run cycles)                                    : 617228496573225


  Utilization rate                      :      6373.636 %
  Flop                                  :   46608625.395 Mflop
  Flop rate (flops / WCT)               :     22649.305 Mflop/s
  Flops / user time                     :      355.359 Mflop/s
  FMA percentage                        :        60.045 %

…….
```

Come si può notare, alcuni dati sono raccolti in tabelle, mentre altri sono semplicemente riportati accanto al nome attribuitogli;

Per l'estrapolazione verranno comunque considerati tutti allo stesso modo ed in particolare verrà fatto riferimento ad un valore attraverso la riga di appartenenza (identificata con una delle informazioni contenute) e la sua posizione nella riga stessa.

Lo script è implementato utilizzando il linguaggio di programmazione Perl e per questo motivo è semplice comprendere che per estrapolare i dati verranno utilizzati tutti gli strumenti resi disponibili dal Perl per l'elaborazione delle stringhe.

Cerchiamo di capire bene come sia possibile estrapolare i dati.

Cominciamo con l'analizzare i possibili valori associabili all'attributo mode:

- **line** -> usato per estrapolare un valore da una riga in cui è riportato un solo valore;
- **line,add** -> usato per fare la somma di tutti i valori indicati come operandi attraverso l'uso di $patint o $patfp;
- **line,last** -> usato per estrapolare l'ultimo valore da una riga che contiene più valori (considera solo quelli indicati con $patint o $patfp);
- **line,min** -> usato per estrapolare il valore minimo da una riga che contiene più valori (considera solo quelli indicati con $patint o $patfp);
- **line,max** -> usato per estrapolare il valore massimo da una riga che contiene più valori (considera solo quelli indicati con $patint o $patfp);
- **line,statistics** -> usato per calcolare la media statistica dei valori presenti sulla stessa riga (considera solo quelli indicati con $patint o $patfp);
- **line,index(i)** -> usato per estrapolare il valore presente nella posizione "i" da una riga che contiene più valori (considera solo quelli indicati con $patint o $patfp);

- **derived** -> usato per effettuare operazioni tra campi già estrapolati (è usato ad esempio per calcolare l'SYPD nel nostro esempio. Vedi l'esempio riportato sopra per dettagli).
  Il valore derived è anche usato per stampare valori di parametri dichiarati nel file di dichiarazione del set di benchmark da effettuare (*bench-IBM-P6-calypso.xml*). Nel nostro caso, ad esempio è stato utilizzato per riportare il numero di mesi simulati dalla suite (*NMONTHS*).

A questo punto cerchiamo di capire cosa si intende con "considera solo quelli indicati con $patint o $patfp".

Come detto in precedenza, lo script si preoccupa di identificare la linea dalla quale prelevare i risultati attraverso una parola che si trova al suo interno.

Se si guarda ad esempio il file riportato prima si trova la seguente riga:

```
<parm name="mem"  unit="KB*s"     mode="line,last" type="int">Unshared
mem use in data seg.:\s+\d+\s+\d+\s+\d+\s+$patint</parm>
```

Che preleva i risultati dalla seguente riga del file di log:

```
Unshared mem use in data seg.: 129575770    8292849294      126803516  214538337 KB*s
```

Analizziamo la dichiarazione:

1. Il nome ***mem*** è quello che figurerà nella tabella dei risultati;
2. L'unità di misura è ***KB*s***;
3. Si vuole prelevare l'ultimo valore tra quelli indicati (***mode="line,last"***);
4. Il tipo di dato è ***int***:
5. Per identificare la riga da cui prelevare i risultati viene usata l'**espressione** racchiusa tra i tag <*parm …..*> e <*/parm*>;

   Analizziamola:

- La riga viene identificata dalle parole presenti all'inizio: "***Unshared memuse in data seg.:***";
- Per indicare i possibili spazi presenti tra i due punti e il primo valore usiamo **\s** (che si usa per identificare uno spazio) seguito da *, perché non sappiamo se ci sarà o no lo spazio (l'**asterisco** identifica da 0 a infiniti spazi, mentre il + identifica da 1 a infiniti spazi);
- Poi usiamo ***\d*** per identificare un dato che non ci interessa;
- Poi +;
- Poi di nuovo **\s**;
- E così via fino ad indicare tutto ciò che è presente nella riga fino al punto in cui è il risultato che interessa a noi.

  Nel nostro caso il risultato interessato è nella quarta posizione, quindi inseriamo 3 volte \d e poi $patint.

I valori ammessi come parametro per dire allo script che in quella posizione c'è un valore che deve essere considerato per l'elaborazione dei risultati sono:

- **$patint**: pattern per i numeri interi;
- **$patfp**: pattern per i numeri in floating point;
- **$patwrd**: pattern per word (tutti i caratteri non vuoti);
- **$patnint**: pattern per i numeri interi, no ( );
- **$patnfp**: pattern per i numeri in floating point, no ( );
- **$patnwrd**: pattern per word (tutti i caratteri non vuoti) , no ( );
- **$patbl**: pattern per spazi bianchi (di dimensione variabile).

Inoltre si possono usare i seguenti valori per saltare interi blocchi di parole:

- **\d** se il valore da saltare è un intero;
- **\s** se il valore da saltare è uno spazio;

- + usato quando ci si aspetta come minimo un valore/spazio, ma anche di più;
- * usato quando ci si aspetta come minimo 0 valori/spazi, ma anche più;
- ( ) usate per raggruppare valori.

RESULT.XML

Rimane, ora, da stabilire quali tra tutti i parametri dichiarati prima, devono essere realmente stampati nella tabella dei risultati.

Per questo fine è stato creato e messo a disposizione il file **result.xml** nel quale si trovano due principali tag:

- **show**
- **sort**

Il primo serve ad indicare quali campi devono essere stampati.

Il secondo permette di stabilire secondo quale parametro si vogliono ordinare i risultati in tabella.

Guardiamo l'esempio:

```
<result>

    <show active="1" colw="10">
    nodes,taskspernode,threadspertask,
    walltime,
    SYPD,
    NMONTHS,
    mem,
    vcheck,vcomment
    </show>

    <sort>
    Walltime,SYPD
    </sort>
</result>
```

Come si può facilmente vedere, si è deciso di stampare i campi "**nodes**", "**taskpernode**", "**threadperstask**", "**walltime**", "**SYPD**", "**NMONTHS**", "**mem**", "**vcheck**", "**vcomment**", ordinando i risultati in base al campo **walltime** (che in questa tesi si è sempre chiamato execution time) e poi, a parità di walltime, in base a **SYPD**.

BIBLIOGRAFIA

[1] M. Quinn. “Parallel programming in C with MPI and OPEN MP”. Mcgraw-Hill College

[2] D. A. Patterson, J. L. Hennessy. “Struttura e progetto dei calcolatori. L’interfaccia hardware-software”. Zanichelli. Seconda edizione

[3] M. De Blasi. “Sistemi per l’elaborazione dell’informazione. Architettura dei calcolatori”. Edizioni Fratelli LaTerza.

[4] S. Holzner. “Perl Black Book: The Most Comprehensive Perl Reference Available Today”. Coriolis Group Books.

[5] URL: http://www2.fz-juelich.de/jsc/jube/ “JuBE: Jülich Benchmarking Environment“. Contatti: jube-jsc@fz-juelich.de .

[6] URL: http://www.deisa.eu/science/benchmarking “Benchmarking & Benchmark Suite”.

[7] URL: http://www.deisa.eu/science/benchmarking/codes/echam5 . “ECHAM5“

[8] URL: http://www.deisa.eu/science/benchmarking/codes/nemo . “NEMO”

[9] URL: http://www.deisa.eu/science/benchmarking/download_form . “Download page DEISA benchmarks”

[10] URL: http://www.mpimet.mpg.de/en/wissenschaft/modelle/echam/echam5.html . “ECHAM5 Website” Contatti: christof.wilhelm@zmaw.de ; fanny.adloff@zmaw.de .

[11] URL: http://www.nemo-ocean.eu/ . “NEMO Website” Contatti: yka@nocs.soton.ac.uk ; sga@noc.soton.ac.uk .

[12] URL: http://en.wikipedia.org/wiki/Benchmark_%28computing%29 . “Benchmark (computing)”.

[13] URL: http://www.spec.org/ . “Standard Performance Evaluation Corporation”.

[14] URL: http://www.spec.org/hpc2002/ . “The HPC2002 Benchmarks“. Contatti: info@spec.org ; support@spec.org.

[15] URL: www.ibm.com/support/docview.wss?uid=swg27005408&aid=1 . “XL Fortran Enterprise Edition for AIX – User Guide – version 9.1”

[16] URL: http://ubuntuforums.org/archive/index.php/t-157488.html . “HowTo install g95 fortran compiler”.

[17] URL: http://ss64.com/bash/time.html . “UNIX man pages : time ()”.

[18] URL: http://www.ipnom.com/FreeBSD-Man-Pages/pgrep.1.html . “Bash MAN: pgrep”.

[19] URL: http://www.gidforums.com/t-3369.html . “Executing programs with C(Linux)”.

[20] URL: http://www.pluto.it/files/ildp/guide/abs/index.html . “Guida avanzata di scripting Bash”. Contatti: thegrendel@theriver.com .

[21] URL: http://www.coresis.com/extra/linuxcorsobase/6-4.htm . “Guida Linux: Editare testi con vi”.

[22] URL: http://www.unidata.ucar.edu/software/netcdf/docs/other-builds.html#g95 . “Other Builds of the netCDF Package” . Contatti: steve@unidata.ucar.edu ; support@unidata.ucar.edu .

[23] URL: http://www.afs.enea.it/project/eneagrid/Resources/Comandi-LSF/index_LSF.htm . “LSF: comandi per l'esecuzione di programmi in batch.” . Contatti: salvatore.raia@portici.enea.it

[24] URL: http://www.cmcc.it/ . “Centro EuroMediterraneo per i Cambiamenti Climatici” . Contatti: info@cmcc.it ; direzione@cmcc.it .