Linux Benchmarking HOWTO

di André D. Balsa, *andrewbalsa@usa.net* v0.12, 15 agosto 1997

Il Linux Benchmarking HOWTO discute alcune questioni relative al benchmarking ("misura delle prestazioni") di sistemi Linux e presenta un semplice strumento di benchmarking ed un modulo a questo associato che consente di fornire informazioni significative per il benchmarking in un paio d'ore. Probabilmente aiuterà anche a diminuire la quantità di articoli inutili in comp.os.linux.hardware ... Traduzione di Gianluca Pecoraro (*atahualpa@altavista.net*).

Indice

1 Introduzione

What we cannot speak about we must pass over in silence.

Ludwig Wittgenstein (1889-1951), filosofo austriaco

Benchmarking significa **misurare** la velocità con la quale un computer esegue un processo, in una maniera tale da consentire il confronto tra differenti combinazioni di hardware e software. Ciò **non** include la facilità, l'estetica, considerazioni ergonomiche o qualsiasi altro giudizio soggettivo.

Il Benchmarking è un processo tedioso e ripetitivo e necessita dell'attenzione ai dettagli. Molto spesso i risultati non sono quelli che ci si aspettava, e sono soggetto di interpretazione (che attualmente può essere la parte più importante di una procedura di benchmarking).

Infine il benchmarking implica fatti e dati, non opinioni e approssimazioni.

1.1 Perché il benchmarking è così importante?

Oltre alle ragioni spiegate nel BogoMips Mini-HOWTO (sezione 7, paragrafo 2), ci si trova a volte a confrontarsi con un budget limitato e un minimo di prestazioni richieste per il suo sistema Linux. In altre parole, quando ci si pongono le seguenti domande:

- Come massimizzo le prestazioni per un dato budget?
- Come riduco i costi per un dato livello minimo di prestazioni?
- Come ottengo il miglior rapporto prezzo/prestazioni (entro un dato budget o una data richiesta di prestazioni?)

Si dovrà esaminare, confrontare e/o produrre benchmark. Minimizzare i costi senza un tetto minimo di prestazioni richiesto di solito implica il mettere assieme una macchina con rimasugli (quel vecchio 386SX-16 che sta da qualche parte nel garage andrà bene) e non richiede benchmark, e massimizzare le prestazioni senza un limite di costi non è una situazione realistica (a meno che non si voglia mettere un sistema Cray nel proprio soggiorno - gli alimentatori rivestiti in pelle attorno sembrerebbero simpatici, no?)

Il benchmarking di per se è senza senso, una perdita di tempo e soldi; ha solo senso quando è parte di un processo di decisione, p.es. se si deve fare una scelta tra due o più alternative.

Di solito un altro parametro nel processo di decisione è il **costo**, ma potrebbe anche essere la disponibilità, il servizio, la compatibilità, decisioni strategiche o altre caratteristiche misurabili e razionali di un computer. Quando si confrontano, p.es. le prestazioni di differenti versioni del kernel di linux, la **stabilità** è quasi sempre più importante della velocità.

1.2 Considerazioni non valide per il benchmarking

Spesso si leggono nei newsgroup e nelle mailing list, sfortunatamente:

1. Reputazione del costruttore (non misurabile e senza senso).
2. Quote di mercato del costruttore (senza senso e irrilevanti).
3. Parametri irrazionali (p.es. la superstizione o il pregiudizio: compreresti un processore etichettato 171717ZAP e colorato di rosa?)
4. Valori percepiti (senza senso, non misurabili e irrazionali).
5. Presenza di una forte campagna pubblicitaria: questo è uno dei peggiori, penso. Io personalmente sono cresciuto con i logo XXX inside o kkkkkws compatibile (ora aaaaaPowered è entrato a far parte della truppa - chi sarà il prossimo ?). A mio modesto avviso, i miliardi di dollari spesi in queste campagne sarebbero meglio utilizzati dai team di ricerca nel progetto di processori nuovi, più veloci, (economici :-) senza-bug . Nessuna massiccia campagna pubblicitaria potrà rimuovere un bug nel coprocessore in virgola mobile di un nuovo processore che hai appena montato sulla tua scheda madre, ma il cambio con un processore riprogettato lo farebbe.
6. Le opinioni del tipo Tu hai per quello che paghi sono solo quello: opinioni. Fornite i fatti per favore.

2 Procedure di benchmarking e interpretazione dei risultati

Qualche raccomandazione semi-ovvia

1. Prima di tutto, **identificare i propri obiettivi nel benchmarking**. Cosa si sta esattamente tentando di testare? In che modo il processo di benchmarking ci aiuterà nel prendere una decisione? Quanto tempo e risorse si è intenzionati ad impiegare nei propri sforzi di benchmarking?
2. **Usare strumenti standard**. Usare una versione recente ma stabile del kernel, le attuali gcc e libc ed un banco di test standard. In breve usare l'LBT (vedere più avanti)
3. Dare una **completa descrizione** del setup (vedere il modulo LBT più avanti).
4. Provare a**isolare una singola variabile**. Il Benchmark comparativo è più informativo del benchmark assoluto**. Non lo ripeterò mai abbastanza.**
5. **Verificare i propri risultati**. Fare i propri benchmark più di una volta e verificare le variazioni nei risultati, se ce ne sono. Variazioni inspiegate invalideranno i risultati.

6. Se si pensa che i propri sforzi nel benchmarking produrrano informazioni utili, **condividerle** con la comunità Linux in una maniera **precisa** e **concisa**.

7. Per favore **ci si dimentichi dei BogoMips**. Mi sono ripromesso di implementare un giorno un ASIC davvero veloce con il ciclo dei BogoMips dentro. Poi vedremo quello che vedremo!

2.1 Capire le scelte di benchmarking

2.1.1 Benchmark sintetici contro applicazioni

Prima di spendere un bel po' di tempo nei processi di benchmarking va fatta una scelta di base fra benchmark sintetici e applicazioni benchmark.

I benchmark sintetici sono spesso progettati per misurare le performance di una singola componente di un sistema, di solito impiegando questa componente alla sua massima capacità. Un esempio ben conosciuto di benchmark sintetico è la **Whetstone** suite, originariamente programmata nel 1972 da Harold Curnow in FORTRAN (o era ALGOL?) e ancora largamente usata ai giorni nostri. La suite Whetstone misurerà le prestazioni dell'unità in virgola mobile di una CPU.

La maggiore critica che può essere fatta ai benchmark sintetici, è che non rappresentano le prestazioni di un sistema nelle situazioni della vita reale. Prendiamo ad esempio la suite Whetstone: il ciclo principale è molto corto e entrerà facilmente nella cache primaria di una CPU, tenendo la pipeline dell'unità in virgola mobile costantemente riempita in modo tale da esercitare l'FPU alla sua massima velocità. Non possiamo davvero criticare la Whetstone suite se ricordiamo che è stata programmata più di 25 anni fa (e le date della progettazione risalgono ancora a prima!), ma noi dobbiamo essere sicuri di interpretare i suoi risultati con attenzione, quando ci troviamo a testare un moderno microprocessore.

Un altro punto molto importante circa i benchmark sintetici è che, in teoria, loro ci possono dire qualcosa rispetto ad uno **specifico** aspetto del sistema che stiamo provando, indipendentemente da tutti gli altri aspetti: un benchmark sintetico per il troughtput di una scheda Ethernet può avere lo stesso o similare risultato che esso sia effettuato su un 386SX-16 con 4MB di Ram o su un Pentium 200 MMX con 64 MB di RAM. Altrimenti, un test potrebbe misurare le prestazioni generali della combinazione di CPU/Scheda Madre/Bus/Scheda Ethernet/Sottosistema di memoria/DMA: non molto utile dato che il cambio del microprocessore causerebbe un impatto molto più grande rispetto al cambio della scheda di rete Ethernet (questo, ovviamente, prendendo per scontato che stiamo usando la stessa combinazione di kernel e driver, cosa che potrebbe causare una variazione ancora più grande)!

Infine uno degli errori più comuni è fare la media di alcuni benchmark sintetici e affermare che quella media è una buona rappresentazione della vita reale per ogni dato sistema.

Questo è un commento dei benchmark sull'unità in virgola mobile riproposto con il permesso del sito web della Cyrix Corp.:

> *Una unità in virgola mobile (FPU) accelera il software progettato per usare il calcolo matematico in virgola mobile: tipicamente programmi CAD, fogli elettronici, giochi 3D e applicazioni di disegno. Fatto sta, che oggi molte tra le più popolari applicazioni per pc fanno uso di entrambe le istruzioni in virgola mobile e intere. Come risultato, Cyrix ha scelto di enfatizzare il parallelismo nella progettazione dei processori 6x86 per velocizzare le prestazioni dei software che mischiano questi due tipi di istruzioni.*
>
> *Il modello di eccezione del floating point x86 permette alle istruzioni intere di iniziare e completarsi mentre un istruzione in virgola mobile è ancora in elaborazione. In contrasto, una seconda istruzione in virgola mobile non può iniziare la sua esecuzione mentre una precedente istruzione è ancora in esecuzione. Per rimuovere questo limite alle prestazioni, il 6x86 può specularmente*

> *iniziare fino a quattro istruzioni in virgola mobile alla FPU nel chip mentre continua ad iniziare ed eseguire istruzioni intere. Per esempio, in una sequenza di codice con due istruzioni in virgola mobile (FLT) seguite da sei istruzioni intere (INT) seguite da due FLT, il processore 6x86 può inizare ad eseguire tutte e dieci le istruzioni con l'appropriata unità di esecuzione prima ancora del completamento dell'esecuzione del primo FLT. Se nessuna delle istruzioni fallisce (il caso tipico), l'esecuzione continua con entrambe le unità intera ed a virgola mobile che completano in parallelo le istruzioni. Se uno degli FLT fallisce (caso atipico), la capacità di esecuzione speculativa del 6x86 permette che lo stato del processore sia restaurato in una maniera che sia compatibile con il modello di eccezione dell'x86.*
>
> *L'esame dei test di benchmark rivela che i benchmark sintetici dell'unità in virgola mobile usano un 'code stream' in pura virgola mobile che non si trova nelle applicazioni del mondo reale. Questo tipo di benchmark non trae vantaggio dalle possibilità di esecuzione speculativa dei processori 6x86. Cyrix crede che i benchmark non sintetici basati sulle applicazioni del mondo reale riflettano meglio le attuali prestazioni che gli utenti otterranno. Le applicazioni del mondo reale contengono funzioni intere e a virgola mobile miste, e quindi beneficieranno della capacità di esecuzione speculativa del 6x86*

Così la recente moda nel benchmarking è di scegliere applicazioni comuni ed usare queste per provare le prestazioni di un computer completo. Per esempio, **SPEC**, l'organizzazione no-profit che ha progettato le ben conosciute suite di benchmark SPECINT e SPECFP ha lanciato un progetto per una nuova suite di benchmark applicativo. Ma ancora è molto difficile che certi benchmark commerciali includeranno mai codice Linux.

Ricapitolando, i benchmark sintetici sono validi fino a che si capiscono i loro obiettivi e le loro limitazioni. I benchmark applicativi rifletteranno meglio le prestazioni di un computer nel suo insieme, ma nessuno è disponibile per Linux.

2.1.2 Benchmark di alto livello contro quelli di basso livello

I benchmark di basso livello misureranno direttamente le prestazioni dell'hardware: il clock del processore, i cicli di DRAM e SRAM, il tempo medio di accesso del disco rigido, la latenza, il tempo di stepping da traccia a traccia, ecc ... Questo può essere utile nel caso si compri un sistema e si vuole vedere con quali componenti è stato costruito, ma una maniera migliore di controllare questo sarebbe di aprire il case, elencare i numeri di serie di tutti i componenti, e poi ottenere le caratteristiche per ogni componente sul web.

Un altro uso dei benchmark di basso livello è di controllare che il driver del kernel sia correttamente configurato per una specifica componente hardware: se si hanno le caratteristiche dichiarate dal produttore per quel componente si possono confrontare i risultati dei benchmark di basso livello con quelle ...

I benchmark di alto livello tengono maggiormente conto delle prestazioni della combinazione di hardware/driver e sistema operativo per un aspetto specifico di un sistema, per esempio, le prestazioni di input&output, o anche le prestazioni di una singola applicazione rispetto alla combinazione di hardware/driver e sistema operativo, p.es. un benchmark di Apache su sistemi differenti.

Ovviamente tutti i benchmark di basso livello sono sintetici. I benchmark di alto livello possono essere sintetici o applicativi.

2.2 Benchmark standard disponibili per Linux

A mio modesto avviso un semplice test che ognuno può fare nell'aggiornare qualsiasi componente del suo sistema Linux è di lanciare la compilazione del kernel prima e dopo l'aggiornamento di hardware e/o software e confrontare i tempi di compilazione. Se tutte le altre condizioni sono tenute costanti allora il test è valido come una misura delle prestazioni nella compilazione e si può essere tranquilli nel dire che:

> Cambiare A in B porta ad un miglioramento delle prestazioni di x % nella compilazione del kernel di Linux sotto tale e tali condizioni.

Ne più, ne meno!

Dal momento che la compilazione è un processo molto usuale sotto linux, e dal momento che esercita molte funzioni che vengono esercitate dai normali benchmark (eccetto le prestazioni in virgola mobile), esso costituisce un test **individuale** abbastanza buono. In molti casi, comunque, i risultati da test a test non possono essere riprodotti da altri utenti Linux perché ci sono variazioni nelle configurazioni hardware/software e così questo tipo di test non può essere usato come unità campione per confrontare sistemi dissimili (a meno che non ci accordiamo su un kernel standard da compilare - vedi più avanti).

Sfortunatamente, non ci sono strumenti di benchmarking specifici per Linux, eccetto forse i Byte Linux Benchmarks che sono una versione leggermente modificata dei Byte Unix Benchmarks che datano Maggio 1991 (Trasposizione Linux di Jon Tombs, autori originali Ben Smith, Rick Grehan e Tom Yager).

C'è un *sito web* centrale per i Byte Linux Benchmarks.

Una versione migliorata e aggiornata dei Byte Unix Benchmarks è stata messa assieme da David C.Niemi. Questa è chiamata UnixBench 4.01 per evitare confusioni con le versioni precedenti. Questo è ciò che David scrisse circa i suoi mods:

> *Gli originali e leggermente modificati BYTE Unix benchmarks sono rotti in un numero tale di modi che fanno di loro un indicatore stranamente non stabile delle performance del sistema. Intenzionalmente ho fatto sembrare i miei valori indice molto differenti per evitare confusioni con i vecchi benchmarks.*

David ha messo su una mailing list basata su majordomo per discutere del benchmark su Linux e sistemi operativi concorrenti. Per partecipare si invii subscribe bench nel corpo di un messaggio a

majordomo@wauug.erols.com `mailto:majordomo@wauug.erols.com` . Il Washington Area Unix User Group è anche in procinto di mettere su un *Sito Web* per i benchmark Linux.

Inoltre recentemente, Uwe F. Mayer, *mayer@math.vanderbilt.edu* `mailto:mayer@math.vanderbilt.edu` ha portato la BYTE Bytemark suite in Linux. Questa è una moderna suite attentamente assemblata da Rick Grehan del BYTE Magazine per testare CPU, FPU e memoria su un moderno computer (questi sono benchmark strettamente orientati alle prestazioni del processore, né l'I/O né le performance del sistema sono tenute in conto).

Uwe ha anche messo online un *Sito Web* con un database di risultati per la sua versione dei Linux BYTEmark benchmarks.

Mentre si cerca per benchmark sintetici per Linux, si noterà che sunsite.unc.edu ha disponibili diversi strumenti di benchmarking. Per testare la velocità relativa dei server X e delle schede grafiche, la suite xbench-0.2 di Claus Gittinger è disponibile da sunsite.unc.edu, ftp.x.org e altri siti. Xfree86.org si rifiuta (saggiamente) di rendere disponibile o raccomandare alcun benchmark.

L' *XFree86-benchmarks Survey* è un sito web con database di risultati di x-bench.

Per il troughput puro I/O dei dischi il programma hdparm (incluso con molte distribuzioni, altrimenti disponibile da sunsite.unc.edu) misurerà la velocità di trasferimento se avviato con le opzioni -t e -T. Ci sono molti altri strumenti liberamente disponibili su Internet per provare vari aspetti delle prestazioni di un sistema Linux.

2.3 Collegamenti e riferimenti

comp.benchmarks.faq di Dave Sill è ancora il punto di riferimento standard per il benchmarking. Non è specifico per Linux, ma la lettura è raccomandata per chiunque si voglia impegnare seriamente nel benchmarking. È disponibile da un grande numero di FTP e siti web ed elenca **56 differenti benchmark**, con links agli FTP o siti web che li rendono disponibili. Alcuni dei benchmark listati sono commerciali (SPEC per esempio).

Quindi non andrò oltre nell'esaminare ognuno dei benchmark menzionati in comp.benchmarks.faq, ma c'è almeno una suite di basso livello di cui voglio parlare: la *lmbench suite* , di Larry McVoy. Citando David C. Niemi:

> *Linus e David Miller la usano molto perché fa molte utili misurazioni di basso livello e può anche misurare il throughput di rete e la latenza se si hanno 2 sistemi da testare.*

Un *Sito FTP*

piuttosto completo per benchmark **liberamente** disponibili è stato messo su da Alfred Aburto. La Whetstone suite usata nell'LBT può essere trovata a questo sito.

C'è una **FAQ in più parti di Eugene Miya** che viene postata regolarmente su comp.benchmarks; è un eccellente punto di riferimento.

3 The Linux Benchmarking Toolkit (LBT)

Proporrò uno strumento base per testare sistemi Linux. Questa è una versione preliminare di un Linux Benchmarking Toolkit, da essere espansa e migliorata. Prendetela per come è, cioè una proposta. Se si pensa che questa non sia una valida suite di test, si è invitati a mandare una e-mail con le proprie critiche e saranno apportati i cambiamenti e migliorie se possibile. Prima di entrare nell'argomento, comunque, è consigliabile leggere questo HOWTO e i punti di riferimento menzionati: critiche informate sono le benvenute, vuoto criticismo no.

3.1 Razionale

Questo è solo il buonsenso:

1. Non dovrebbe prendere un giorno intero per andare. Quando si va al benchmarking comparativo, nessuno vuole spendere giorni tentando di immaginare il più veloce setup per un dato sistema. Idealmente l'intero set di benchmark dovrebbe prendere circa 15 minuti per completarsi su una macchina media.
2. Tutto il codice sorgente utilizzato per il programma deve essere liberamente disponibile su internet per ovvie ragioni.
3. I benchmark dovrebbero provvedere semplici indici che rispecchino le performance misurate.
4. Ci dovrebbe essere un misto di benchmark sintetici e applicativi
5. Ogni benchmark **sintetico** dovrebbe impiegare il relativo sottosistema alla sua massima capacità.
6. I risultati dei benchmark **sintetici non** dovrebbero essere unificati in un unico indice (che non rispetta l'intera idea che c'è dietro i benchmark sintetici, con considerevole perdita di informazioni).
7. I benchmark applicativi dovrebbero consistere in processi eseguiti comunemente in un sistema Linux

3.2 Selezione dei benchmark

Ho selezionato cinque differenti suite di benchmarking, tentando il più possibile di eliminare overlap nei test:

1. Compilazione del Kernel 2.0.0 (configurazione di default) usando gcc.
2. Whetstone suite versione 10/03/97 (ultima versione di Roy Longbottom).
3. xbench-0.2 (con parametri di esecuzione veloce).
4. UnixBench benchmarks versione 4.01 (risultati parziali).
5. BYTE Magazine's BYTEmark benchmarks beta release 2 (risultati parziali).

Per i test 4 e 5, (risultati parziali) significa che non tutti i risultati prodotti da questi benchmark vanno considerati.

3.3 Durata dei test

1. Kernel 2.0.0, compilazione: da 5 a 30 minuti, dipende dalle **reali** performance del sistema.
2. Whetstone: 100 secondi.
3. Xbench-0.2: < 1 ora.
4. UnixBench benchmarks versione 4.01: approssimativamente 15 minuti.
5. BYTE Magazine's BYTEmark benchmarks: approssimativamente 10 minuti.

3.4 Commenti

3.4.1 Compilazione Kernel 2.0.0:

- **Cos'è:** È l'unico benchmark applicativo in LBT.
- Il codice è largamente disponibile (p.es. finalmente troverete un uso per i vostri vecchi cd di linux).
- Molti utenti Linux ricompilano il kernel abbastanza spesso, così è una misura significante delle performance generali del sistema
- Il kernel è grosso e gcc usa un bel po' di memoria: ciò attenua il miglioramento che si potrebbe avere grazie alla cache di secondo livello svolgendo piccoli test
- Svolge frequenti operazioni di I/O col disco
- Procedura del test: prendi il sorgente originale di un kernel 2.0.0, compilalo con le opzioni di default (make config e premendo Invio ripetutamente). Il tempo riportato dovrebbe essere il tempo speso nella compilazione p.es. dopo che si è digitato make zImage e **non** includendo make dep, make clean. Notare che l'architettura di default per il kernel è la i386, perciò se è compilato su un altra architettura, gcc dovrebbe essere impostato come cross-compile, con i386 come architettura di destinazione.
- **Risultati:** tempo di compilazione in minuti e secondi (per favore, è inutile riportare le frazioni di secondo).

3.4.2 Whetstone:

- **Cos'è:** misura le prestazioni pure dell'unità in virgola mobile con un corto ciclo. Il sorgente (in C) è abbastanza leggibile ed è molto facile vedere quali operazioni in virgola mobile ne prendono parte.
- Test più corto del LBT :-).
- È un vecchio classico test: punti di riferimento sono disponibili, i suoi limiti sono ben conosciuti.
- Procedura del test: il più nuovo sorgente in C dovrebbe essere ottenuto dal sito Aburto. Compilarlo e eseguirlo in modalità doppia precisione. Specificare gcc e -O2 come precompilatore e opzioni del precompilatore.
- **Risultati:** un indice delle prestazioni dell'unità in virgola mobile in MWIPS.

3.4.3 Xbench-0.2:

- **Cos'è:** misura le prestazioni dell'X server.
- La misura xStones fornita da xbench è una media pesata di numerosi test riferiti come indice ad una vecchia Sun station con un display in bianco e nero. Hmmm... non è inappuntabile come test dei moderni X server, ma è ancora il miglior strumento che abbia trovato.
- Procedura del test: compilare con -O2. Specifichiamo qualche opzione per una esecuzione più veloce: `./xbench -timegoal 3 > results/name_of_your_linux_box.out`. Per avere un punteggio in xStones, dobbiamo eseguire uno script awk; il modo più semplice è di digitare `make summary.ms`. Controllare il file summary.ms: il punteggio in xStone del sistema è nell'ultima colonna della linea con cui hai specificato il nome della propria macchina durante il test.
- **Risultati:** le prestazioni di X misurate in xStones.
- Nota: questo test, così com'è è obsoleto. Dovrebbe essere ricodificato

3.4.4 UnixBench versione 4.01:

- **Cos'è:** misura le prestazioni totali di Unix. Questo test impegnerà le prestazioni di I/O e le prestazioni di multitasking del kernel
- Ho scartato tutti i risultati dei test aritmetici, tenendo solo i risultati relativi al sistema.
- Procedura di test: compilare con -O2. Eseguire con `./Run -1` (esegue ogni test una sola volta). Si troverano i risultati in ./results/report file. Calcolare il significato geometrico di EXECL THROUGHPUT, FILECOPY 1, 2, 3, PIPE THROUGHPUT, PIPE-BASED CONTEXT SWITCHING, PROCESS CREATION, SHELL SCRIPTS e gli indici di SYSTEM CALL OVERHEAD.
- **Risultati:** un indice di sistema.

3.4.5 BYTE Magazine's BYTEmark benchmarks:

- **Cos'è:** fornisce una buona misurazione delle prestazioni del processore. Questo è un estratto dalla documentazione: *questi benchmark hanno la funzione di mostrare il limite superiore teorico di CPU, FPU e architettura di memoria di un sistema. Non possono misurare il video, i dischi o il throughput della rete (questi sono dominio di un'altro set di benchmark). Si dovrebbe, comunque, usare il risultato di questi test come una parte di un processo di misurazione di un sistema, non come tutto il processo.*

- Ho scartato i test in virgola mobile dal momento dal momento che il test dei Whetstone è più rappresentativo delle prestazioni in virgola mobile.
- Ho diviso i test interi in due parti: quelli più rappresentativi delle prestazioni di memoria-cache-CPU e i test interi della CPU.
- Procedura del test: compilare con -O2. Eseguire il test con `./nbench > myresults.dat` o simile. Poi da myresults.dat, calcolare la media geometrica degli indici della prova STRING SORT, ASSIGNMENT e BITFIELD ; questo è l'**indice di memoria** ; calcolare la media geometrica degli indici della prova di NUMERIC SORT, IDEA, HUFFMAN and FP EMULATION ; questo è l'**indice intero**.
- **Risultati:** un indice di memoria e un indice intero calcolato come spiegato.

3.5 Miglioramenti possibili

La suite ideale di benchmark dovrebbe terminare in qualche minuto, con benchmark sintetici che provino ogni sottosistema separatamente e benchmark applicativi che forniscano risultati per diverse applicazioni. Dovrebbero anche generare automaticamente un rapporto completo e eventualmente spedirlo via e-mail ad un database centrale sul web.

Qui non siamo davvero interessati alla portabilità, ma dovrebbe almeno funzionare su tutte le recenti (> 2.0.0) versioni e sapori (i386, Alpha, Sparc...) di Linux.

Se qualcuno ha un idea circa il benchmarking delle prestazioni di rete in una maniera semplice e facile, con un test breve (meno di 30 minuti per impostarlo ed eseguirlo), per favore mi contatti.

3.6 Modulo LBT

Dopo i test, la procedura di benchmarking non sarebbe completa senza un modulo che descrivesse il setup, che così dovrebbe essere (seguendo le linee guida di comp.benchmarks.faq):

```
LINUX  BENCHMARKING TOOLKIT REPORT FORM
```

```
CPU
==
Produttore:
Modello:
Frequenza di clock:
Produttore della scheda madre:
Modello della sk.madre:
Chipset della sk.madre:
Tipo di bus:
Freq. di clock del bus:
Cache totale:
Tipo e velocità della cache:
SMP (numero di processori):
```

```
RAM
====
Totale:
Tipo:
Velocità:
```

```
Disco
====
Produttore:
Modello:
Capienza:
Interfaccia:
Driver/Settaggi:
```

```
Scheda video:
===========
Produttore:
Modello:
Bus:
Tipo di Video RAM:
Totale di Video RAM:
Produttore X server:
Versione X server:
Scelta del chipset nell'X server:
Risoluzione/freq di refresh verticale:
Profondità di colore:
```

```
Kernel
=====
Versione:
Dimensione file di swap:
```

```
gcc
===
Versione:
Opzioni:
versione libc:
```

```
Note al test
==========
```

```
RISULTATI
========
Tempo di compilazione del kernel 2.0.0
Tempo di compilazione: (minuti e secondi)
Whetstones: risultati in MWIPS.
Xbench: risultati in xstones.
Unixbench Benchmarks 4.01: indice di systema
BYTEmark: INDEX intero
BYTEmark: INDICE di memoria
```

```
Commenti*
=========
* Questo campo è incluso per possibili interpretazioni dei risultati,
e come specificato è opzionale. Potrebbe essere la parte più
significativa del proprio report, specialmente se si stanno
effettuando benchmark comparativi.
```

3.7 Test delle prestazioni della rete

Provare le prestazioni di una rete è un'ardua sfida dal momento che include almeno due macchine, un server ed un client, quindi il doppio del tempo per impostarlo e molte molte variabili da controllare, ecc... Su una rete ethernet, penso che la migliore scelta sarebbe il pacchetto ttcp. (da espandere)

3.8 Test degli SMP (Multi processori simmetrici)

I test degli SMP sono un'altra sfida ed ogni benchmark specificatamente progettato per testare l'SMP avrà un lungo tempo per dimostarsi valido nelle impostazioni della vita reale, dal momento che gli algoritmi che possono prendere vantaggio dall'SMP sono difficili da progettare. Sembra che le ultime versioni del kernel (> 2.1.30 è successive) faranno un multiprocessing fine-grained. Ma non ho informazioni maggiori in questo momento.

Secondo David Niemi, *... shell8*[parte degli Unixbench 4.01 benchmaks]*svolge un buon lavoro nel confrontare combinazioni simili di hardware e sistemi operativi nei modi SMP e UP*

4 Esempi di esecuzione e risultati

L'LBT è stato eseguito sulla mia macchina di casa, un sistema Linux di classe Pentium con il quale ho scritto questo HOWTO. Qui c'è il modulo di report LBT per il mio sitema

```
LINUX  BENCHMARKING TOOLKIT REPORT FORM

CPU

==

Produttore: Cyrix/IBM

Modello: 6x86L P166+

Frequenza di clock: 133 MHz

Produttore scheda madre: Elite Computer Systems (ECS)

Sk. madre: P5VX-Be

Chipset: Intel VX

Tipo di bus: PCI
```

```
Frequenza di clock del bus: 33 MHz

Cache totale: 256 KB

Cache tipo/velocità: Pipeline burst 6 ns

SMP (numero di processori): 1

RAM

====

Totale: 32 MB

Tipo: EDO SIMMs

Velocità: 60 ns

Disco

====

Produttore: IBM

Modello: IBM-DAQA-33240

Grandezza: 3.2 GB

Interfaccia: EIDE

Driver/Settaggi: Bus Master DMA modo 2

Scheda video

===========

Produttore: Generica S3

Modello: Trio64-V2

Bus: PCI

Tipo Video RAM: EDO DRAM

Totale Video RAM: 2 MB

Produttore X server: XFree86

Versione X server: 3.3
```

```
Scelta chipset dell'X server: S3 accellerato

Risoluzione/rinfresco verticale: 1152x864 @ 70 Hz

Profondità di colore: 16 bit

Kernel

=====

Versione: 2.0.29

File di swap: 64 MB

gcc

===

Versione: 2.7.2.1

Opzioni: -O2

versione libc: 5.4.23

Note del test

==========

Carico molto basso. I seguenti risultati sono stati ottenuti abilitando alcune delle speciali features del Cyrix/IBM 6x86
abilitate con il programma setx86: fast ADS,  fast IORT, Enable DTE, fast LOOP, fast Lin. VidMem.

RISULTATI

========

Kernel Linux 2.0.0 Tempo di compilazione: 7m12s

Whetstones: 38.169 MWIPS.

Xbench: 97243 xStones.

BYTE Unix Benchmarks 4.01 INDICE di sistema: 58.43

BYTEmark integri INDICE: 1.50

BYTEmark INDICE memoria: 2.50

Commenti

=========

Questo è un sistema molto stabile con prestazioni omogenee, ideale per l'uso di casa e/o lo sviluppo in Linux. Riporterò al più presto i risultati con un processore 6x86MX appena ne avrò uno fra le mani!
```

5 Trappole e inesattezze del benchmarking

Dopo aver messo insieme questo HOWTO ho iniziato a comprendere come le parole trappole e inesattezze siano così spesso associate al benchmarking...

5.1 Paragonando mele e arance

O dovrei dire Apple e PC? Questa è una disputa così vecchia e ovvia che non andrò avanti nei dettagli. Dubito che il tempo che impieghi Word a caricarsi su un Mac confrontato con un medio Pentium sia la misura reale di qualcosa. Lo stesso vale per il confronto tra il tempo di boot di Linux e Windows NT, ecc ... Provare a confrontare il più possibile macchine identiche con una singola modifica.

5.2 Informazioni incomplete

Un singolo esempio illustrerà questo errore molto comune. Spesso si legge in comp.os.linux.hardware le seguenti frasi, o simili: Ho appena montato il processore XYZ a nnn MHz e ora compilare il kernel di linux prende solo x minuti (poni pure XYZ , nnn e x come meglio credi). Questo è irritante, perché nessun altra informazione è data, ad.es. non sappiamo neanche l'ammontare della RAM, la dimensione dello swap, gli altri processi in esecuzione nello stesso momento, la versione del kernel, i moduli selezionati, il tipo di hard disk, la versione di gcc, ecc... È consigliato di usare il modulo di report LBT che utilizza almeno una maschera di raccolta dati standard.

5.3 Hardware/software proprietario

Un ben conosciuto produttore di processori una volta pubblicò i risultati dei benchmarks prodotti da una speciale, personalizzata versione di gcc. A parte le considerazioni etiche, questi risultati erano senza senso, dal momento che il 100% della comunità Linux continuerebbe ad usare la versione standard di gcc. Lo stesso vale per l'hardware proprietario. Il Benchmarking è molto più utile quando si rapporta a hardware comune e software libero (nel significato GNU/GPL di libero).

5.4 Rilevanza

Stiamo parlando di Linux, giusto? Così ci dobbiamo scordare dei benchmark prodotti su altri sistemi operativi (questo è un caso speciale della trappola Confrontare mele e arance spiegata prima). Allo stesso modo, se si è intenzionati a testare le prestazioni di un server Web, **non** citare le prestazioni dell'unità in virgola mobile, o altre informazioni irrilevanti. In molti casi meno è meglio. Allo stesso modo, **non** c'è bisogno di menzionare gli anni del tuo gatto, il tuo stato d'animo mentre stavi effettuando il test, ecc...

6 FAQ (Domande Frequenti)

D1.

C'è un qualsiasi indice di merito per i sistemi Linux?

A:

No, fortunatamente nessuno è ancora venuto fuori con una misurazione Lhinuxstone (tm). E se ce ne fosse uno, non avrebbe molto senso: i sistemi Linux sono usati per tanti differenti compiti, dal server web pesantemente caricato alla workstation grafica per uso individuale. Nessun singolo indice di merito può descrivere le prestazioni di un sistema Linux in differenti situazioni.

D2.

Quindi perché non una dozzina di indici riassumenti le prestazioni di diversi sistemi Linux?

A:

Sarebbe la situazione ideale. Vorrei vedere ciò diventare realtà. Nessun volontario per un **Linux Benchmarking Project**? Con un sito web e un database completo e ben progettato?

D3.

... BogoMips ...?

A:

I BogoMips non hanno **niente** a che fare con le prestazioni del sistema. Leggere il BogoMips Mini-HOWTO.

D4.

Qual è il miglior benchmark per Linux?

A:

Dipende da quale aspetto prestazionale di linux si vuole misurare. Ci sono differenti benchmark per misurare la rete (p.es i transfer rate), i file server (NFS), l'I/O dei dischi, le prestazioni sui calcoli in virgola mobile e interi, grafica, 3D, larghezza di banda della memoria del processore, prestazioni CAD, tempo di transazione, prestazioni SQL, prestazioni del server Web, prestazioni in tempo reale, prestazioni del CD-ROM, prestazioni di Quake (!), ecc ... AFAIK nessuna suite di benchmark esiste per linux che supporti tutti questi test.

D5.

Qual è il processore più veloce sotto Linux?

A:

Il più veloce in quale processo? Se si è molto orientati alla masticazione di numeri, un processore Alpha ad alta frequenza di clock (600 MHz in su) dovrebbe essere più veloce di nessun altro, dal momento che gli Alpha sono stati progettati proprio per fornire questo tipo di prestazioni. Se, dall'altro lato, si vuole mettere assieme un server news davvero veloce, è probabile che la scelta di un veloce sottosistema di dischi rigidi e molta RAM risulterà in un migliore aumento di prestazioni che un cambio di processore per la stessa somma di $.

D6.

Fammi riformulare l'ultima domanda, allora: c'è un processore che è il più veloce per applicazioni generiche?

A:

Questa è una domanda trabocchetto ma ha una sola e semplice risposta: **NO**. Si può sempre disegnare un sistema veloce anche per applicazioni generiche, indipendentemente dal processore. Di solito, restando uguali tutte le altre cose, una più alta frequenza di clock risulterà in maggiori prestazioni del sistema (e anche più mal di testa). Tirando fuori un vecchio processore a 100Mhz da una motherboard (di solito non) aggiornabile, e inserendo dentro la versione a 200Mhz, si dovrebbe sentire una grande differenza. Certamente con solo 16 MB di RAM, lo stesso investimento sarebbe stato molto più saggiamente speso in DIMMs aggiuntive ...

D7.

Ma allora la frequenza di clock influenza le prestazioni di un sistema?

A:

Per molti processi, eccetto per i loop NOP vuoti (a proposito, questi stanno per essere rimossi dai moderni compilatori ottimizzanti), una maggiorazione nella frequenza di clock non darebbe un aumento lineare delle prestazioni. Molti piccoli programmi che fanno un uso intensivo del processore entrano interamente nella cache primaria del processore (la cache L1, usualmente 16 o 32K) e avranno un aumento di prestazioni equivalente all'aumento della frequenza di clock, ma molti veri programmi sono molto più larghi di ciò, hanno cicli che non entrano nella cache L1, condividono la cache di secondo livello (L2) con altri processi, dipendono da componenti esterni e daranno un incremento di prestazioni molto minore. Questo è perché la cache L1 funziona alla stessa frequenza di clock del processore, mentre molte cache di secondo livello e tutti gli altri sottosistemi (DRAM per esempio) funzionano in maniera asincrona a minore frequenza di clock.

D8.

OK, poi, un ultima domanda su questa questione: qual è il processore con il miglior rapporto tra prezzo e prestazioni per un uso generico di Linux?

A:

Definire un Linux di uso generico non è una cosa facile! Per ogni particolare applicazione, c'è sempre un processore con IL MIGLIORE rapporto prezzo/prestazioni in ogni momento, ma cambia abbastanza frequentemente così come i produttori rilasciano nuovi processori, così rispondere Processore XYZ a nnn Mhz sarebbe una risposta valida solo in quel momento. Comunque il prezzo di un processore è insignificante se paragonato al prezzo dell'intero sistema che si sta assemblando. Così, davvero, la questione dovrebbe come si può massimizzare il rapporto tra prezzo e prestazioni di un dato sistema? E la risposta a questa domanda dipende fortemente dal minimo di prestazioni richieste e/o dal prezzo massimo stabilito per la configurazione che si sta considerando. A volte l'hardware comune non soddisfa le prestazioni minime richieste e costosi sistemi RISC sarebbero la sola alternativa. Per l'uso di casa si raccomanda un sistema bilanciato e omogeneo per tutte le prestazioni (ora vai ad immaginare che cosa io intendo per bilanciato e omogeneo :-); la scelta di un processore è una decisione importante, ma non più che scegliere il tipo di disco fisso e la capacità, il quantitativo di RAM, la scheda video, ecc ...

D9.

Quando un aumento delle prestazioni è considerato significativo?

A:

Vorrei rispondere che innanzittutto ogni variazione sotto all'1% non è significativa (potrebbe essere descritta come marginale). Noi umani, difficilmente percepiamo la differenza tra due sistemi con un 5% di differenza nel tempo di risposta. Certamente alcuni benchmarkers hard-core non sono umani e ti diranno confrontando un sistema con indici di sistema 65.9 e 66.5 che il secondo è sicuramente molto più veloce.

D10.

Come posso ottenere significativi aumenti nelle prestazioni al costo minore?

A:

Dal momento che molti codici sorgenti per Linux sono disponibili gratuitamente, l'esame attento e il ridisegno algoritmico delle subroutine chiave potrebbe aumentare in alcuni casi le prestazioni. Se ci si trova ad avere a che fare con un progetto commerciale e non ci si vuole addentrare profondamente nello sviluppo in codice C **si dovrebbe chiamare un consulente Linux**. Vedere il Consultants-HOWTO.

7 Copyright, riconoscimenti e varie

7.1 Come è stato scritto questo documento:

Il primo passo è stato leggere la sezione 4 di Writing and submitting a HOWTO dell'HOWTO Index di Tim Bynum.

Non conoscevo assolutamente niente di SGML o LaTeX, ma sono stato tentato di usare un pacchetto automatico di generazione dopo aver letto i vari commenti sugli SGML-Tools. Comunque inserire i tags manualmente in un documento mi ricorda i giorni che assemblai a mano un programma monitor da 512 byte per un, ora defunto, processore a 8bit, così sono entrato in possesso dei codici sorgenti di LyX, li ho compilati, ed ho usato il suo modo LinuxDoc. Combinazione altamente raccomandata: **LyX e SGML-Tools**.

7.2 Copyright

7.3 Nuove versioni di questo documento

Le nuove versioni del Linux Benchmarking-HOWTO saranno depositate su sunsite.unc.edu e siti mirror. Sono disponibili altri formati, come una versione Postscript e dvi, nella directory other-formats. Il Linux Benchmarking-HOWTO è pure disponibile per client WWW come Grail, un Web browser scritto in Python. Sarà pure inviato regolarmente a comp.os.linux.answers.

7.4 Commenti e critiche

Sono richiesti suggerimenti, correzioni e aggiunte. Si cercano contributi e riscontri. Non si cercano flame.

Posso sempre essere raggiunto a andrewbalsa@usa.net.

7.5 Riconoscimenti

David Niemi, l'autore della suite Unixbench, ha provato di essere una fonte inesauribile di informazioni e di (valide) critiche.

Voglio anche ringraziare Greg Hankins uno dei maggiori contributori al pacchetto SGML-tools, Linus Torvalds e l'intera comunità Linux. Questo HOWTO è il mio modo per ringraziare.

7.6 Disclaimer

Your mileage may, and will, vary. Be aware that benchmarking is a touchy subject and a great time-and-energy consuming activity.

7.7 Marchi registrati

Pentium e Windows NT sono marchi registrati rispettivamente da Intel e Microsoft.

BYTE e BYTEmark sono marchi registrati da McGraw-Hill, Inc.

Cyrix e 6x86 sono marchi registrati da Cyrix Corporation.

Linux non è un marchio registrato, e spero che mai lo sarà. :)