X.systems.press

# Leistungsbewertung bei Computersystemen

Praktische Performance-Analyse von Rechnern und ihrer Kommunikation

Bearbeitet von
Hans Günther Kruse

1. Auflage 2009. Buch. vii, 201 S. Hardcover
ISBN 978 3 540 71053 0
Format (B x L): 15,5 x 23,5 cm

# 2 Einfache Leistungsmaße

In diesem Kapitel geht es um einfache, direkt messbare Leistungsgrößen oder -maße, welche eine erhebliche Bedeutung über alle Systemklassen und Größenordnungen hinweg haben.

## 2.1 Takt, CPI und Ablaufzeit

Das am leichtesten zugängliche Leistungsmaß einer Hardware, um genauer zu sein, eines Prozessors, ist das diskrete feste Zeitraster, nach dem sich alle Aktionen ausrichten – der Takt!

So entspricht der 100-MHz-Takt längst Vergangenheit gewordener Maschinen einem Zeitabstand zwischen zwei Ereignissen – der Basistaktzeit bzeit – von 10 nsec, das sind $10*10^{-9}$ sec. Die heutigen Gigahertz-Prozessoren entsprechen einem Abstand von 1 nsec. Zählen wir die Takte, welche eine vorgegebene Last benötigt, mit der Variable #**takte** (# bezeichnet immer eine Anzahl), dann ergibt sich die Ablaufzeit zcpu der Last zu

$$\mathrm{zcpu} = \#\,\mathrm{takte} * \mathrm{bzeit} = \frac{\#\,\mathrm{takte}}{\frac{1}{\mathrm{bzeit}}} = \frac{\#\,\mathrm{takte}}{\mathrm{takt}}$$

denn klarerweise ist **takt = 1/bzeit**. Hierbei ist die Variable #**takte** zwar unmittelbar einsichtig, aber schlecht einer direkten Messung zugänglich, weiterhin ist damit die Last nicht vor, sondern nach der Verarbeitung definiert – eine zu große Abweichung von unserem Konzept. Deshalb wird wie folgt verfahren.

Die Zahl der Instruktionen der Last #**last** lässt sich im Prinzip vor der Verarbeitung bestimmen. Wir messen die Zahl der benötigten Takte #takte und bestimmen mit der Größe cpi, wie viel Takte für eine Instruktion benötigt werden:

$$\text{cpi} = \frac{\#\,\text{takte}}{\#\,\text{last}}$$

Für die Ablaufzeit folgt danach:

$$\text{zcpu} = \frac{\#\,\text{takte}}{\text{takt}} = \frac{\#\,\text{takte}}{\#\,\text{last}} * \frac{\#\,\text{last}}{\text{takt}}$$

$$\textbf{\textit{zcpu}} = \text{cpi} * \frac{\#\,\text{last}}{\text{takt}} = \#\textbf{\textit{last * cpi * bzeit}}$$

Diese Umformungen sind keine überflüssige Spielerei, sondern widerspiegeln die drei wichtigsten Ebenen in einem DV-System:

- #**last** = Zahl der Instruktionen, hängt von der Anwendung und Compilertechnologie ab;
- **cpi** = Zahl der Basistakte, die für eine Instruktion benötigt wurde, hängt von der Maschinenarchitektur ab;
- **bzeit** = Basistakt, ist direkt von der eingesetzten Hardware- Technologie abhängig.

So verkürzt eine steigende Taktrate die Basiszeit **bzeit** und damit in der Theorie die Ablaufzeit **zcpu**. Eine RISC-Architektur versucht, **cpi** auf den Wert 1 zu drücken, also einen Takt pro Instruktion – in superskalaren Maschinen liegt dieser Wert sogar darunter –, und hochoptimierende Compiler versuchen die Anzahl der Maschineninstruktionen #**last** zu minimieren.

Die Größe **cpi** muss noch etwas genauer analysiert werden. Hintergrund ist die Tatsache, dass Maschinenbefehle unterschiedliches Zeitverhalten von den Hardware-Ressourcen her haben, und es daher kein exaktes globales **cpi** geben kann. Vielmehr muss man die Maschinenbefehle in Gruppen von Instruktionen einteilen, welche dann innerhalb der Gruppe k alle die gleiche Taktzahl pro Befehl besitzen:

$$\{\text{Instruktionen}\}_{k=1\ldots n} => \text{cpi}_{k=1\ldots n}$$

Kennt man dann das prozentuale Gewicht $p_k$ jeder Gruppe mit $p_1 + p_2 + p_3 + \ldots + p_n = 1$ (n ist die Gruppenanzahl), dann gilt für die globale Größe

$$\text{cpi} = \sum_{k=1}^{n} p_k * \text{cpi}_k$$

Das folgende Beispiel soll dies etwas verdeutlichen.

**B1**
Wir betrachten eine Modell-RISC-Maschine mit den Befehlsgruppen

| | | $cpi_k$ | $p_k$ |
|---|---|---|---|
| Arithmetisch-logisch | ALU | 1 | 0,4 (40%) |
| Speicher | LOAD/STORE | 2 | 0,4 (40%) |
| Test & Verzweigung | JMP | 3 | 0,2 (20%) |

$$cpi = 1*0{,}4 + 2*0{,}4 + 3*0{,}2 = 1{,}8$$

An der Größe **cpi** kann man jetzt sehr schön den Einfluss verschiedener Maßnahmen analysieren. Angenommen wir ändern durch den Einsatz eines besseren Compilers das Gewicht $p_n = I_n / I$ der n-ten Gruppe $I_n \rightarrow \alpha * I_n$, wobei $0 \le \alpha \le 1$.

Aus dem alten Wert

$$cpi = \sum_{k=1}^{n} cpi_k \left( \frac{I_k}{I} \right), \quad I = \sum_{k=1}^{n} I_k$$

wird dann

$$cpi' = \sum_{k=1}^{n-1} cpi_k \left( \frac{I_k}{I - \left( I_n - \alpha I_n \right)} \right) + cpi_n * \left( \frac{\alpha * I_n}{I - \left( I_n - \alpha I_n \right)} \right)$$

$$= \sum_{k=1}^{n-1} cpi_k * \left( \frac{I_k}{I} \right) * \frac{1}{\left( 1 - (1-\alpha) * p_n \right)} + \left( \frac{\alpha * p_n * cpi_n}{1 - (1-\alpha) * p_n} \right)$$

$$= \frac{1}{1-(1-\alpha)p_n} \left( \sum_{k=1}^{n-1} \underbrace{p_k * cpi_k + p_n * cpi_n}_{cpi} - p_n * cpi_n + \alpha * p_n * cpi_n \right)$$

Also folgt

$$cpi' = \frac{1}{1-(1-\alpha) * p_n} (cpi - (1-\alpha) * p_n * cpi_n)$$

Da $0 \le p_n$, $1 - \alpha \le 1$, ist der Vorfaktor $1/(1 - (1 - \alpha)\, p_n) \ge 1$ und signalisiert damit erst einmal eine Verschlechterung des neuen cpi-Wertes. Diese kann nur durch die Subtraktion in der Klammer kompensiert werden.

Diese Formel können wir jetzt benutzen, um Änderungen im Instruktionsumfang abzuschätzen.

**B2**
In der Beispielmaschine von B1 wird $p_{ALU}$ um die Hälfte auf 0,2 reduziert, wie ändert sich cpi?

$$p_{ALU} = 2/5;\ p'_{ALU} = 1/5 \text{ d.h } \alpha = 1/2$$

$$cpi' = \frac{1}{1-(1-\frac{1}{2})*\frac{2}{5}}(\frac{9}{5}-(1-\frac{1}{2})*\frac{2}{5}*1) = \frac{5}{4}*\frac{8}{5} = 2$$

Dieses Beispiel zeigt, dass eine an sich positive Maßnahme sogar einen schlechten **cpi-**Wert zur Folge haben kann. Dies ist aber klar, weil die größeren Werte ($cpi_{LS} = 2$, $cpi_{JMP} = 3$) stärker gewichtet werden.
Reduzieren wir etwa in der Gruppe der Sprungbefehle um die Hälfte (0,2 → 0,1), folgt $\overline{cpi} = \frac{1}{1-(1-\frac{1}{2})*\frac{1}{5}}*(\frac{9}{5}-(1-\frac{1}{2})*\frac{1}{5}*3) = \frac{10}{9}*\frac{15}{10} = \frac{5}{3}$ und die Verbesserung beträgt immer hin ca. 8%.

Der andere Parametertyp, den man verändern kann, ist $cpi_k$ von einzelnen Gruppen, ohne dass der Umfang der Maschineninstruktionen verändert wird. Im Prinzip bedeutet dies eine Architekturänderung im Leitwerk (μ-Programm). Wir greifen wieder die n-te Gruppe heraus und reduzieren über $cpi_n' = \alpha * cpi_n$, $0 \le \alpha \le 1$:

$$\begin{aligned} <cpi'> &= \sum_{k=1}^{n-1} p_k * cpi_k + p_n * \alpha * cpi_n \\ &= \sum_{k=1}^{n-1} \underbrace{p_k * cpi_k + p_n * cpi_n}_{cpi} - p_n * cpi_n + p_n * \alpha * cpi_n \end{aligned}$$

$$<cpi'> = cpi - (1-\alpha) * p_n * cpi_n$$

Hier fehlt der vergrößernde Faktor $1/(1 - (1 - \alpha)\ p_n)$, sodass deutlich höhere Chancen bestehen, den cpi-Wert mit dieser Maßnahme zu verkleinern.

**B2'**
Wir setzen die Situation wie in B1 und B2 voraus, schaffen es durch eine Änderung des Mikroprogramms, den Wert von $cpi_{JMP}$ von 3 auf 1,5 zu reduzieren. Dann folgt

$$\begin{aligned}\overline{cpi} &= cpi_{ALU} * p_{ALU} * cpi_{LS} * p_{LS} + cpi_{JMP} * p_{JMP} \\ &= 1 * 0{,}4 + 2 * 0{,}4 + 1{,}5 * 0{,}2 \\ &= 1{,}5\end{aligned}$$

Dies bedeutet eine 20-prozentige Verbesserung und liegt damit deutlich über den Anstrengungen von B2!

Das dritte Beispiel zu der cpi-Thematik ist etwas ausführlicher und diskutiert auch die Abhängigkeit der Ablaufzeit.

**B3**
In zwei Maschinen A und B braucht die Verzweigung zwei Zyklen und alle anderen Befehle einen Zyklus. In der auszumessenden Last sind 20% Verzweigungen. Die Taktzeit (Basiszeit) von A ist 25% günstiger als die von B. Welche Maschine ist hinsichtlich dieses Benchmarks günstiger, wenn ein optimierender Compiler 20% der Instruktionen auf B in der Gruppe der Einzyklusbefehle einspart?

$zcpu_A = \#last_A * cpi_A * bzeit_A$

$cpi_A = 2 * p_{JMP} + 1 * p_{Rest}$

$= 2*0{,}2 + 1*0{,}8 = 1{,}2 = \frac{6}{5} \Rightarrow zcpu_A = \frac{6}{5} * \#last_A * bzeit_A$

Für die Maschine B schließen wir analog

$zcpu_B = \#last_B * cpi_B * bzeit_B$

$cpi_B = 2 * p_{JMP} + 1 * p_{Rest} = 1{,}2 = \frac{6}{5}$

$bzeit_B = \frac{5}{4} * bzeit_A$ nach Voraussetzung

Damit folgt ohne optimierenden Compiler ($\#last_B = \#last_A$):

$$zcpu_B = (\frac{5}{4}) * \#last_B * (\frac{6}{5}) * bzeit_B = (\frac{5}{4})\, zcpu_A$$

Der Gewinn (Speed-Up) $Sp(B/A) = (zcpu_B/zcpu_A)$ berechnet sich zu 5/4, d. h. die Maschine B ist um 25% langsamer.

Der Einsatz des optimierenden Compilers bringt $\#last_B = 0{,}8 * last_A$, allerdings muss $cpi_B$ neu berechnet werden. Wie in B2 folgt

$$cpi'_B = \frac{1}{1-0{,}2}\left(\frac{6}{5} - 0{,}2 * 1\right) = \frac{5}{4}$$

$$\begin{aligned} zcpu_B &= 0{,}8\ \#last_A * (\frac{5}{4}) * (\frac{5}{4})\ bzeit_A \\ &= (\frac{4}{5}) * (\frac{5}{4}) * (\frac{5}{4}) * \#last_A * bzeit_A \end{aligned}$$

Damit können wir wie oben die Rechner A und B vergleichen:

$$Sp(B/A) = \frac{zcpu_B}{zcpu_A} = \frac{\left(\frac{5}{4}\right) * \#last_A * bzeit_A}{\left(\frac{6}{5}\right) * \#last_A * bzeit_A} = \frac{25}{24} > 1$$

Rechner B wird also um ca. 5% langsamer sein, obwohl er 20% weniger Instruktionen verarbeiten muss.
Konzentriert man sich statt der eben diskutierten Maßnahme auf eine 20%ige Verbesserung von $cpi_{REST}$, dann resultiert ein neuer $cpi_B$-Wert von 26/25. Der Instruktionsumfang bleibt gleich, sodass ein Vergleich der zcpu-Zeiten 13/12 ergibt, d. h. die Maschine ist um 8% langsamer.

## 2.2 Durchsatz, MIPS und MFLOPs

Unter dem Aspekt, die Leistung direkt aus der Taktzeit und der Anzahl der Takte pro Instruktionen zu erschließen und das physikalische Leistungsmaß „Arbeit pro Zeiteinheit“ möglichst gut abzubilden, sind zwei Begriffe entwickelt worden, die

ein gewisses Renommee erworben haben und damit etwas eingehender diskutiert werden sollen.

Das eine Maß ist **MIPS** (**M**illionen **I**nstruktionen **P**ro **S**ekunde) und das heißt, wenn man ein Programm als mögliche Last vorgibt (#last ist die Zahl der Instruktionen), dann folgt:

$$\begin{aligned} \text{MIPS} &= \frac{\#\text{last}}{\text{zcpu}} * 10^{-6} \\ &= \frac{\#\text{last} * 10^{-6}}{\#\text{last} * \text{cpi} * \text{bzeit}} \\ &= \frac{10^{-6}}{\text{cpi} * \frac{1}{\text{takt}}} = \frac{\text{takt}}{\text{cpi}} * 10^{-6} \\ &= \frac{\text{takt}}{\text{cpi}} * 10^{-6} \end{aligned}$$

Dies sieht als Leistungsmaß sehr vernünftig aus, es wächst mit dem Takt (der Taktrate) und fällt mit der Komplexität der Architektur – dann nämlich, wenn ich im Mittel immer mehr Takte für einen Maschinenbefehl verbrauche (z. B. CISC-Maschinen).

Von Hennessy & Patterson [6] werden eine Reihe von Einwänden gegenüber diesem Maß angeführt, die wichtigsten sind:

- MIPS ist abhängig von der Menge der Instruktionen, also der Architektur.
- MIPS ist extrem lastabhängig.
- MIPS kann invers zur gemessenen Leistung sein.

Während die beiden ersten Punkte eigentlich nur eine Reflexion des Standardproblems beim Benchmarking darstellen, muss der dritte Punkt wesentlich ernster genommen werden.

Betrachten wir Gleitkommaoperationen, diese können in Hardware oder Software realisiert sein. Nehmen wir zunächst die Hardware-Variante; im Allgemeinen bedeuten sie ein komplexeres Mikroprogramm, d. h. der Wert von cpi wird größer und damit MIPS kleiner – obwohl die Ausführungszeit sinkt.

Bei der Software-Variante werden einfache Instruktionen mit kleinerem cpi verwendet, dieses bewirkt ein Anwachsen von MIPS – obwohl die Ausführungszeit **zcpu** ansteigt.

**B4**

Wir greifen auf unsere RISC-Maschine von Beispiel B1 zurück, unterstellen eine Taktrate von 1 GHz = takt. Der Einsatz eines optimierenden Compilers bringt eine 50-prozentige Reduktion der ALU-Operationen. Man berechne den Effekt auf MIPS und Ablaufzeit!

$$cpi_{normal} = 1{,}8$$

$$MIPS_{normal} = \frac{takt}{cpi_{normal}} * 10^{-6} = \frac{1*10^9}{1{,}8} * 10^{-6} = \frac{1000}{1{,}8} = \frac{10000}{18} = \frac{5000}{9}$$

$$cpi_{opt} = \frac{(\frac{0{,}4}{2}) * 1 + 0{,}4 * 2 + 0{,}2 * 3}{1 - (0{,}4/2)} = \frac{0{,}2 + 0{,}8 + 0{,}6}{0{,}8} = \frac{1{,}6}{0{,}8} = 2$$

$$MIPS_{opt} = \frac{takt}{cpi_{opt}} * 10^{-6} = \frac{1*10^9}{2} * 10^{-6} = \frac{1000}{2} = 500$$

$$\frac{MIPS_{opt}}{MIPS_{normal}} = \frac{500}{\frac{500}{1 - \frac{1}{10}}} = 1 - \frac{1}{10} = \frac{9}{10}$$

Die optimierte Version hat also eine um 10% kleinere MIPS-Rate!
Wie sieht es bei den Ausführungszeiten aus?

$$zcpu_{normal} = \#last * cpi_{normal} * bzeit\ [sec]$$

$$= \#last * 1{,}8 * 1{,}0 * 10^{-9} = 1{,}8 * 10^{-9} * \#last\ [sec]$$

$$zcpu_{opt} = 0{,}8 * \#last * cpi_{opt} * bzeit$$
$$= 0{,}8 * \#last * 2 * 1 * 10^{-9} = 1{,}6 * 10^{-9} * \#last\ [sec]$$

D. h. obwohl die optimierende Version mehr als 10% schneller läuft, hat sie eine kleinere MIPS-Rate → Widerspruch im Leistungsmaß!

Das zweite populäre Maß, welches eingeführt wurde, um die Leistungsfähigkeit von Maschinen zu beurteilen, ist **MFLOPS** (**M**illion **F**loating **P**oint **O**perations per **S**econd) und in ähnlicher Weise wie MIPS definiert:

$$\text{MFLOPS} = \frac{\#\text{FLOPS}}{\text{zcpu}} * 10^{-6}$$

Es hängt jedoch noch weit mehr als MIPS von der Hardware und dem Anwendungsprogramm ab. Enthält dies keine oder wenig Gleitkommaoperationen (FLOPS), so wird die gemessene Leistung ziemlich irrelevant sein. Es wird mit MFLOPS/GFLOPS/TFLOPS halt nur eine einzige Komponente eines Rechnersystems ausgemessen und hinsichtlich dieser kann man die Leistung vergleichen. Für wissenschaftliche Anwendungen mit stark numerischem Charakter, z. B. aus der Flüssigkeits- und Gasdynamik der Theoretischen Chemie und Theoretischen Physik, der Astronomie und der Gentechnik, sind TFLOPS allerdings ein geeignetes Maß.

Standard-Last ist der sogenannte Linpack-Benchmark, ein Paket zur Lösung linearer Gleichungssysteme, welches von Jack Dongarra/Knoxville-Tennessee geschrieben wurde. Es erlaubt nicht nur die maximale Leistung $R_{max}$, sondern auch die Leistung $N_{max}$ bei der halben Problemgröße (Matrixdimension) zu messen; die Implementierung muss so erfolgen, dass die Zahl der Gleitkommaoperationen sich wie $(2/3)\ n^3 + 0\ (n^2)$ verhält.

Wir führen einige Beispiele aus der Liste von November 2004 an:

| | | |
|---|---|---|
| 1. | IBM BlueGene/L (32768 * PowerPC) | 70,72 TFLOPS |
| 2. | SGI Columbia (10160 * SGI Altix) | 51,87 TFLOPS |
| 3. | NEC Earth Simulator (5120 * NEC) | 35,86 TFLOPS |
| 10. | NCSA Tungsten (2500 * Dell PowerEdge/Xeon) | 9,819 TFLOPS |
| 85. | MPI Garching (822 * IBM p Serie 690 Turbo) | 2,198 TFLOPS |
| 168. | Hitachi Leibniz-RZ München (168 * Hitachi) | 1,653 TFLOPS |

Um unter die Top 10 zu kommen, brauchte eine Installation zu diesem Zeitpunkt mehr als 3 TERAFLOPS = $3*10^{12}$ Gleitkommaoperationen pro Sekunde. Im Frühsommer 2008 waren ca. neun TFLOPS nötig, um überhaupt in die Liste zu kommen, und das IBM-Roadrunner-System durchbrach die Peta-FLOP-Schranke.

Man muss sich aber stets vor Augen halten, dass dies kein allgemeingültiges Leistungsmaß darstellt – die Gründe sind ähnlich wie bei MIPS gelagert.

Abbildung 2.1 zeigt die MFLOP-Raten der jeweils leistungsfähigsten Gleitkommamaschinen (auch Supercomputer genannt) im Zeitraum 1982–2002. Man beachte den logarithmischen Maßstab auf der vertikalen Achse und somit den Technologiesprung um drei Zehnerpotenzen in einer Dekade.

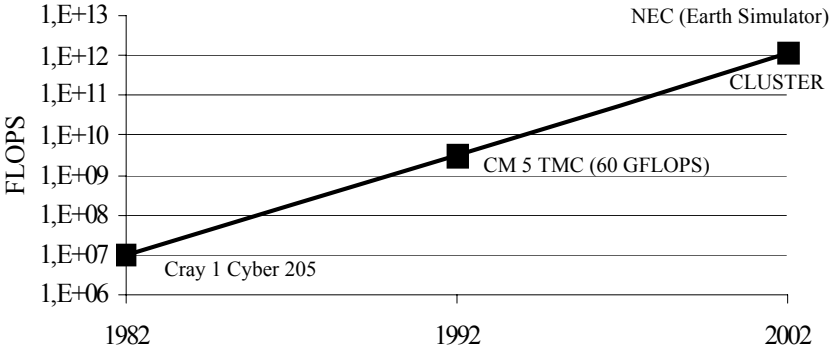


**Abb. 2.1** Leistungsentwicklung von Supercomputern

Ohne auf Einzelheiten einzugehen, widerspiegelt diese Entwicklung auch die Veränderung von einer monolithischen klassischen Architektur wie die Cray 1 oder Cyber 205 über die einheitliche Vernetzung einer großen Zahl identischer Prozessoren wie in der CM 5 von Thinking Machines bis hin zur aktuellen Clusterung kompletter Systeme wie bei NEC Earth Simulator oder IBM SP2 Power3 in Livermore (über 6000 Knoten).

Damit einher geht auch eine Veränderung in der Herstellerlandschaft. Waren es zu Beginn der 1980er-Jahre noch Konstrukteure, die aus dem Großrechnerbereich kamen, änderte sich dies zu Beginn der 1990er fundamental. Eine Reihe junger innovativer Firmen und Designer erschien auf der Bühne und löste mit neuen Architekturkonzepten einen Leistungsboom aus. Viele von diesen sind Ende der Dekade schon wieder verschwunden; durchgesetzt hat sich das Cluster-Prinzip, nach dem eine große Anzahl konventioneller Billigprozessoren oder Standardsysteme (dazu zählt auch der α-Prozessor von DEC, später Compaq bzw. hp) durch ein Hochleistungsnetz verbunden wird.

Einen sehr schönen Überblick der Szene gibt die

*www.top500.org*

Website, welche zweimal jährlich die 500 leistungsfähigsten Maschinen listet und vergleicht – wie bereits erwähnt auf der Basis des Linpack-Benchmarks.