Applied Mathematics Letters 25 (2012) 398–400

Contents lists available at SciVerse ScienceDirect

Applied Mathematics Letters

journal homepage: www.elsevier.com/locate/aml

On the efficiency and capacity of computers

Boris Ryabko

Institute of Computational Technology of Siberian Branch of Russian Academy of Science, Siberian State University of Telecommunications and Informatics, Novosibirsk, Russia

ARTICLE INFO

Article history:
Received 27 June 2010
Accepted 1 September 2011

Keywords:
Computer capacity
Computer efficiency
Shannon entropy
Information theory

ABSTRACT

We address the problems of estimating the computer efficiency and the computer capacity. We define the computer efficiency and capacity and suggest a method for their estimation, based on the analysis of processor instructions and kinds of accessible memory. Obtained results can be of some interest for practical applications.

1. Introduction

We address the problem of what the efficiency (or performance) and the capacity of a computer are and how they can be estimated. More precisely, we consider a computer with a certain set of instructions and several kinds of memory [1]. What is the computer capacity, if we know the execution time of each instruction and the speed of each kind of memory? What is the computer efficiency if the computer is used for solving problems of a certain kind (say, matrix multiplications)? On the one hand, the questions about the computer efficiency and capacity are quite natural, but, on the other hand, to the best of our knowledge, the computer science does not give answers to these questions. The goal of this paper is to suggest a reasonable definition of the computer efficiency and capacity and methods of their estimation. We will mainly consider computers, but our approach can be applied to all devices which contain processors, memories and instructions. (Among these devices we mention mobile telephones and routers.)

2. The computer efficiency and capacity

First, we briefly describe the main point of the suggested approach and give definitions. For a start, we will consider the simplified variant of a computer, which consists of a set of instructions I and an accessible memory M. It is important to note that any instruction $x \in I$ contains not only its name (say, JUMP), but memory addresses and indexes of registers. For example, all instructions JUMP which deal with different memory addresses are contained in I as different instructions.

A computer task P determines a certain sequence of instructions $X(P) = x_1 x_2 x_3 \ldots$, $x_i \in I$. For example, if the program P contains a loop which will be executed ten times, then the sequence X will contain the body of this loop repeated ten times. It is important to note that we do not suppose that all possible sequences of instructions are allowable. In principle, there can be sequences of instructions which are prohibited. For example, it is possible that some pairs of instructions are prohibited. In other words, it is possible that sequences of instructions should satisfy some limitations (or some rules). We define the set of all allowable sequences of instructions by S_C and consider two different sequences of the computer instructions from S_C as two different computer tasks. So, any computer task can be presented as a sequence of instructions from S_C. Moreover, the

E-mail address: boris@ryabko.net.

0893-9659/$ – see front matter
doi:10.1016/j.aml.2011.09.021

opposite is also true: any sequence of instructions from S_C can be considered as a computer task. Indeed, using a so-called assembly language any sequence of instructions from S_C can be presented as a computer program; see, for example, [1].

Let us denote the execution time of an instruction x by $\tau(x)$. For the sake of simplification, we suppose that all execution times $\tau(x), x \in I$, are integers and the greatest common divisor of $\tau(x), x \in I$, equals 1. (This assumption is valid for many computers if the time unit equals a so-called clock rate; see [1]. In this paper, this assumption gives a possibility to use lim instead of lim sup, when the capacity will be defined.) Then the execution time $\tau(X)$ of a sequence of instructions $X = x_1 x_2 x_3 \dots x_t$ is given by $\tau(X) = \sum_{i=1}^{t} \tau(x_i)$.

The key observation is as follows: the number of different computer tasks, whose execution time equals T, is equal to the size of the set of all sequences of instructions, whose execution time equals T, i.e. $\nu(T) = N(T)$, where $\nu(T)$ is the number of different problems, whose execution time equals T, and

$$N(T) = |\{X : \tau(X) = T\}|. \tag{1}$$

In other words, the total number of computer tasks executed in time T is equal to (1). Based on this consideration we give the following definition.

Definition 1. Let there be a computer with a set of instructions I and let $\tau(x)$ be the execution time of an instruction $x \in I$. The computer capacity $C(I)$ is defined as follows:

$$C(I) = \lim_{T\to\infty} \frac{\log N(T)}{T}, \tag{2}$$

where $N(T)$ is defined in (1), $\log x \equiv \log_2 x$.

(That this limit exists can be proven based on the lemma by Krichevsky [2].)

Let us consider a simple example. There are two computers C_1 and C_2 which have identical sets of instructions I_1 and I_2, but their execution times are different in such a way that the speed of the first computer is twice more than that of the second one, i.e. $\tau_1(x) = \tau_2(x)/2$ for any $x \in I$. From definition (2) we immediately obtain that $C(I_1) = 2C(I_2)$. Apparently, this equation is quite natural.

The next question to be investigated is the definition of the computer efficiency (or performance), when a computer is used for solving problems of a certain kind. For example, one computer can be a Web server and another can be used for solving differential equations. Certainly, the computer efficiency depends on the problems the computer has to solve. In order to model this situation, we suggest the following approach: there is an information source which generates a sequence of computer tasks in such a way, that the computer begins to solve each next task as soon as the previous task is finished. We will not deal with a probability distribution on the sequences of the computer tasks, but consider sequences of computer instructions, determined by sequences of the computer tasks, as stochastic processes. In what follows, we will consider the model when this stochastic process is stationary and ergodic, and we will define the computer efficiency for this case.

The definition of efficiency will be based on results and ideas of information theory, which we introduce in what follows. Let there be a stationary and ergodic process $z = z_1 z_2 \dots$ generating letters from a finite alphabet A (the definition of stationary ergodic process can be found, for ex., in [3]). The n-order Shannon entropy and the limit Shannon entropy are defined as follows:

$$h_n(z) = -\frac{1}{n+1} \sum_{u\in A^{n+1}} P_z(u) \log P_z(u), \qquad h_\infty(z) = \lim_{n\to\infty} h_n(z) \tag{3}$$

where $n \geq 0$, $P_z(u)$ is the probability that $z_1 z_2 \dots z_{|u|} = u$ (this limit always exists; see [3,4]). We will consider so-called i.i.d. sources. By definition, they generate independent and identically distributed random variables from some set A. Now we can define the computer efficiency.

Definition 2. Let there be a computer with a set of instructions I and let $\tau(x)$ be the execution time of an instruction $x \in I$. Let this computer be used for solving such a randomly generated sequence of computer tasks, that the corresponding sequence of the instructions $z = z_1 z_2 \dots, z_i \in I$, is a stationary ergodic stochastic process. Then the efficiency is defined as follows:

$$c(I, z) = h_\infty(z) / \sum_{x\in I} P_z(x)\tau(x), \tag{4}$$

where $P_z(x)$ is the probability that $z_1 = x,\ x \in I$.

Informally, the Shannon entropy is a quantity of information (per letter), which can be transmitted and the denominator in (4) is the average execution time of an instruction. So, $c(I, z)$ is a quantity of information per time unit.

3. Methods for estimating the computer capacity

The efficiency, in principle, can be estimated based on statistical data, which can be obtained by observing a computer which solves tasks of a certain kind.

The simplest estimate of computer capacity can be obtained if we suppose that all sequences of the instructions are admissible. In other words, we consider the set of instructions I as an alphabet and suppose that all sequences of letters (instructions) can be executed. In this case, the method of calculation of the lossless channel capacity, given by Shannon in [4], can be applied. He showed that the capacity $C(I)$ is equal to the logarithm of the largest real solution X_0 of the following equation:

$$X^{-\tau(x_1)} + X^{-\tau(x_2)} + \cdots + X^{-\tau(x_s)} = 1, \tag{5}$$

where $I = \{x_1, \ldots, x_s\}$. In other words, $C(I) = \log X_0$.

It is easy to see that the efficiency (4) is maximal, if the sequence of instructions $x_1 x_2 \ldots, x_i \in I$ is generated by an i.i.d. source with probabilities $p^*(x) = X_0^{-\tau(x)}$, where X_0 is the largest real solution to Eq. (5), $x \in I$. Indeed, having taken into account that $h_\infty(z) = h_0(z)$ for i.i.d. source (see [3]) and the definition of entropy (3), the direct calculation of $c(I, p^*)$ in (4) shows that $c(I, p^*) = \log X_0$ and, hence, $c(I, p^*) = C(I)$. So, we obtain

Theorem 1. *Let there be a computer with a set of instructions I and let $\tau(x)$ be the execution time of $x \in I$. Suppose that all sequences of instructions are admissible computer programs. Then the following equalities are valid:* (i) *the alphabet capacity $C(I)$ (2) equals $\log X_0$, where X_0 is the largest real solution to Eq. (5) and* (ii) *the efficiency (4) is maximal if the sequences of instructions are generated by an i.i.d. source with probabilities $p^*(x) = X_0^{-\tau(x)}$, $x \in I$.*

As an example, we briefly consider the MMIX computers suggested by Knuth [5]. The point is that this computer is described in detail and can be considered as a model of a modern computer. The MMIX computer has 256 general-purpose registers, 32 special-purpose ones and 2^{64} bytes of virtual memory [5]. The MMIX instructions are presented as 32-bit words and in this case the "computer alphabet" consists of almost 2^{32} words (almost, because some combinations of bits do not make sense). In [5], the execution (or running) time is assigned to each instruction in such a way that each instruction takes an integer number of υ, where υ is a unit that represents the clock cycle time. Besides, it is assumed that the running time depends on the number of memory references (*mems*) that a program uses. For example, it is assumed that the execution time of each of the *LOAD* instructions is $\upsilon + \mu$, where μ is an average time of memory Ref. [5]. If we consider υ as the time unit and define $\hat{\mu} = \mu/\upsilon$, we obtain from the description of MMIX [5] and (5) the following equation for finding an upper bound on the MMIX capacity:

$$2^{24}\left(\frac{139}{X} + \frac{32}{X^2} + \frac{5}{X^3} + \frac{17}{X^4} + \frac{3}{X^5} + \frac{4}{X^{10}} + \frac{2}{X^{40}} + \frac{4}{X^{60}}\right) + 2^8 2^{64}\left(\frac{46}{X^{1+\hat{\mu}}} + \frac{2}{X^{1+20\hat{\mu}}} + \frac{46}{X^{2+2\hat{\mu}}}\right) = 1. \tag{6}$$

The value $\hat{\mu}$ depends on the realization of MMIX and is larger than 1 for modern computers [5]. Now we can estimate the capacity for different $\hat{\mu}$. For example, if $\hat{\mu}$ is small (say, $\hat{\mu} = 1$), the capacity is large (about 35 bits). If $\hat{\mu}$ is large ($\hat{\mu} = 5$), the capacity is around 31.5 bits.

It is natural to use estimations of the computer capacity at the design stage. We consider an example of such estimations that are intended to illustrate some possibilities of the suggested approach. Suppose a designer has decided to use the MMIX set of registers. Suppose further, that he/she has a possibility to use two different kinds of memory, such that the time of one reference to the memory and the cost of one cell are τ_1, c_1 and τ_2, c_2, correspondingly. It is natural to suppose that the total price of the memory is required not to exceed a certain bound C. As in the example with MMIX we define $\hat{\mu}_1 = \tau_1/\upsilon$, $\hat{\mu}_2 = \tau_2/\upsilon$, where, as before, υ is a unit that represents the clock cycle time.

As in the case of MMIX, we suppose that there are instructions for writing and reading information from a register to a cell. The set of these instructions coincides with the corresponding set of the MMIX computer. If we denote the number of the memory cells by S_i ($S_i = C/c_i$), we can see from (6) that the designer should calculate the maximal roots for the following equations

$$\left(2^{24}\left(\frac{139}{X} + \frac{32}{X^2} + \frac{5}{X^3} + \frac{17}{X^4} + \frac{3}{X^5} + \frac{4}{X^{10}} + \frac{2}{X^{40}} + \frac{4}{X^{60}}\right)\right) + 2^8 S_i\left(\frac{46}{X^{1+\hat{\mu}_i}} + \frac{2}{X^{1+20\hat{\mu}_i}} + \frac{46}{X^{2+2\hat{\mu}_i}}\right) = 1 \tag{7}$$

$i = 1, 2$. Then he/she can choose that kind of memory for which the solution is larger. It will mean that the computer capacity will be larger for the chosen kind of memory.

In conclusion, we note that the suggested approach can be extended to multi-core processors and special kinds of cache memory.

References

[1] A.S. Tanenbaum, Structured Computer Organization, Prentice Hall PTR, 2005.
[2] R. Krichevsky, Universal Compression and Retrieval, Kluwer Academic Publishers, 1993.
[3] T.M. Cover, J.A. Thomas, Elements of Information theory, Wiley, 2006.
[4] C.E. Shannon, A mathematical theory of communication, Bell Syst. Tech. J. 27 (1948) 379–423. 623–656.
[5] D.E. Knuth, The Art of Computer Programming, in: Fascicle 1, MMIX: A RISC Computer for the New Millennium, vol. 1, 2005.