CENTER FOR

COMPUTER AND INFORMATION SCIENCES

AND

DIVISION OF ENGINEERING

COMPUTATIONAL WORK AND EFFICIENT COMPUTATION ON GENERAL PURPOSE MACHINES*

by

J. E. Savage

BROWN UNIVERSITY

Providence
Rhode Island

COMPUTATIONAL WORK AND EFFICIENT COMPUTATION ON GENERAL PURPOSE MACHINES*

by

J. E. Savage

COMPUTATIONAL WORK AND EFFICIENT COMPUTATION ON GENERAL PURPOSE MACHINES*

by

J. E. Savage
Division of Engineering
Brown University
Providence, R. I., U.S.A. 02912

ABSTRACT

A new measure of computational work which was introduced recently is applied to three problems: 1) measurement of the work required to fetch from a store, 2) determination of operating principles for a typical general purpose machine on the basis of the computing power of its storage units and 3) formulation of a minimization problem which provides criterea for determining the cost of a mismatch between storage units. Computational work is a measure of the equivalent number of logical operations performed by machines.

* This work was supported by grants NGR40-002-082 and NGR40-002-090 from the National Aeronautics and Space Administration and grant GK-13162 from the National Science Foundation.

Computational Work and Efficient Computation on General Purpose Machines

by

J. E. Savage
Brown University
Providence, Rhode Island

1. Introduction

In a recent paper [1], a measure of "computational work" introduced earlier in the study of error-correcting decoders [2] has been applied to the computation of finite functions on general purpose machines. In [1] it is shown that a product relation exists between the storage and time required to compute finite functions and that the product must be large for complex functions. In that paper we also give a useful measure of the "computing power" of a storage device and find expressions for this quantity for random-access, tape and disk (or drum) storage units. Using these results, it is shown that a clear ordering of devices exists on the basis of the number of cycles required to compute functions when no storage limit is imposed.

In this article, we review the definitions of computational work and computing power and illustrate one use of work by bounding the work required to fetch data from a store. We also calculate the computing power of storage devices in a typical computing faculty and demonstrate that the power of disk and drum are so much smaller than that of core that marked improvements in run times may be possible through changes in system operating principles. In addition, we consider a machine with a main and an auxiliary memory and consider the minimization of run time subject to a restriction on the amount of computational work required. For the type of main and auxiliary memory considered, it is always desirable to run jobs in main memory when sufficient storage capacity is available. When capacity is limited, we give conditions under which the penalty for use of auxiliary storage is large and these conditions involve a comparison of a work to storage ratio with system parameters.

We expect that the use of the computational work measure will lead to a clearer understanding of exchanges possible between storage, time and other parameters and that it will lead to more efficient use of machines.

2. Work and Computing Power

Model a general purpose (GP) machine by a collection of k sequential machines S_1 , --- , S_k each with its own clock. We assume without excessive loss of generality that the clock cycle for each machine is a multiple of some basic cycle. Thus, the model preserves the identity of the several components of a GP machine and approximates its asynchronous character.

We assume that each of the k sequential machines excutes a fixed number of cycles to do its share of a computation and note that the complex defines a function f which is a map from the initial states of the machines (if they can be freely chosen) and external inputs (if they are provided) to the set of external outputs produced sequentially by the complex. We say that "f is computed by S_1 , -- , S_k ."

To measure the amount of work required to compute f , we create a model for each sequential machine using logic elements from some universal set Ω and using compatible memory cells which have individual inputs and outputs and which act as delay units. If a model for S_i contains X_i logic elements and if S_i executes T_i cycles, then we have [1]

$$W = \sum_{i=1}^{k} X_i T_i \geq C_\Omega(f) \qquad (1)$$

where $C_\Omega(f)$ is the minimum number of logic elements from Ω required to realize f with a combinational (logic) circuit.

We call W the computational work done by the models for S_1 , -- , S_k since it is an equivalent number of logic uses. It should be noted that f which are complex (for which $C_\Omega(f)$ is large) will require a large W, and for a single sequential machine this means the XT product must be large. It should also be noted that the storage media employed in the models can do no work and that all the work is done by logic elements.

If one of the sequential machines is a bulk storage device, we say that it has <u>computing power</u> P if P is the minimum number of logic elements required to model the device with elements from Ω. If the device has a small control (a CPU, perhaps) with X_o logic elements and if it is used to compute f in a fixed number of cycles T, then

$$(X_o + P)T \geq C_\Omega(f) \qquad (2)$$

If $P >> X_o$, we find that a product inequality holds on P and T.

Expressions for the computing power of random-access, tape, disk and drum units have been obtained [1] under the assumptions that Ω is the set of all 2-input binary connectives. If the random-access and tape units have S bits of storage and if the tape, disk and drum units can have access to any one of m bits of information in one cycle (the time required to access one bit in parallel), then

$$\begin{aligned} S &\leq P_{ra} \leq 9S \\ S &\leq P_t \leq 9S + \alpha \log S \qquad (3) \\ m &\leq P_d \leq 5m + \beta \log m \end{aligned}$$

where α and β are constants and P_d is the power of drum or disk, whichever is appropriate. We note that P_{ra} and P_t are proportional to total storage since in the model at least one logic element must be used per binary cell to provide access. On the other hand P_d is proportional to m since the disk and drum units always rotate in the same direction and access is only required to cells under the heads. The terms $\alpha \log S$ and $\beta \log m$ measure the logic required in addressing circuits.

Combining (2) and (3), we find that ST is bounded below approximately by $C_\Omega(f)$ for a small control on tape and random access machines. This is the kind of exchange inequality which programmers have found to hold empirically. It is important to note however, that ST must increase with increasing complexity $C_\Omega(f)$.

Combining (2) with (3) for drum and disk and using additional arguements for tape machines we have

$$T_t \geq \sqrt{\frac{C_\Omega(f)}{9m(1+\varepsilon_1)}} \qquad (4)$$

$$T_d \geq \frac{C_\Omega(f)}{5m(1+\varepsilon_2)}$$

when their controls have a number of equivalent logic elements small by comparison with m and total storage, in the case of the tape machine. Combinational complexity $C_\Omega(f)$ for functions of n variables can be very large; in fact it can be nearly exponential in n. Thus, T_t and T_d can be very large with the lower bound on T_d exponentially larger than that on T_t. By contrast with these results, any function of n variables can be computed on a random access machine of sufficient storage capacity in a number of cycles T_{ra} proportional to n using "table look-up". Consequently, a clear ordering of these storage types exists for functions whose complexity is large by comparison with n.

Let $S_1, S_2, ---, S_k$ have cycle times $\tau_1, ---, \tau_k$. Then the maximum amount of work which they can do in t seconds, W(t), if X_i is the minimum number of equivalent logic elements in S_i, is

$$W(t) = \sum_{i=1}^{k} \frac{X_i}{\tau_i} t \qquad (5)$$

If S_i is a bulk storage unit of computing power P_i and a small control with X_{ci} equivalent logic elements, then

$$W(t) = \sum_{i=1}^{k} \left(\frac{X_{ci}}{\tau_i} + \frac{P_i}{\tau_i} \right) t \qquad (6)$$

and P_i/τ_i, or the normalized computing power of the i-th storage unit, is a measure of the rate at which S_i can do work. The larger P_i/τ_i, the smaller will t need be to do a fixed amount of computational work.

3. The Work Required to Fetch

Assume that a storage device has M locations each containing a b bit word. Then, an instruction to fetch the work in location j, w_j, involves the calculation of a function $f(j; w_1, w_2, ---, w_m) = w_j$. This should be viewed as a function of j with $w_1, --, w_M$ fixed, as would be the case in a read-only memory. Note that the complexity of f depends on the values of $w_1, --, w_M$ since if they all have the same value no work is required to compute f; the output of the minimal circuit is constant.

The number of different fetch functions is clearly b^M or the number of ways to choose $w_1, w_2, --, w_M$. Using just this fact, we can apply a counting argument given in [1] to lower bound $C_\Omega(f)$ by

$$C(f) \geq \frac{1}{2} \frac{Mb}{\log_2(Mb)} (1-\varepsilon) \qquad (7)$$

for almost all fetch functions, ε fixed, $0<\varepsilon<1$, and Ω the set of 2-input binary gates. We also assume that the addresses are represented in binary form using $\log_2 M$ bits.

Returning now to the expressions for computing power in (3) and the inequality of (2), we see that the number of cycles required to compute $f(j; w_1, --, w_M)$ on a random access machine is lower bounded by 1 while the lower bound for drum and disk grows almost linearly with Mb. This is consistent with practical methods of fetching from these units and confirms the accuracy of the results.

4. A Typical GP Machine

Consider a machine which has three types of storage, core (or random access) disk and drum. Using typical values for parameters of these units, we determine their relative computing powers and deduce certain principles for thier efficient use.

Let the core contain $S_c = 4 \times 10^6$ bits of storage and have cycle time $\tau_c = 10^{-6}$ sec. Let the drum have 200 tracks, each with a capacity of 1.6×10^5 bits arranged serially and rotational speed of 3600 rpm. The bits on each track are arranged serially so that the number of accessible bits is $m_{dr} = 200$ and the cycle length is $\tau_{dr} = 1/(60 \times 1.6 \times 10^5)$ sec. $= 10^7$. Let the disk unit have 16 disks each with 4000 tracks and each track containing 5.8×10^4 bits organized in serial. Let each disk rotate at 3600 rpm. Then, the number of accessible tracks or bits $m_d = 6.4 \times 10^4$, the cycle length $\tau_d = 3 \times 10^{-7}$ sec, and the total storage capacity $S_d = 3.7 \times 10^9$ bits. The tracks on a disk are arranged in groups called cylinders and the time required to move a head between adjacent cylinders we assume to be 24×10^{-3} sec. and between extremes cylinders on a disk to be 170×10^{-3} sec. These numbers should be compared to 16×10^{-3} sec., which is the time required to read a complete track or cylinder.

To compute the normalized computing power of the units, we use the upper bounds given in (3) and neglect the additive terms proportional to $\log_2 m$. Then, for core, drum and disk we have

$$\frac{P_c}{\tau_c} = 36 \times 10^{12} \quad \frac{P_{dr}}{\tau_{dr}} = 10^{10} \quad \frac{P_d}{\tau_d} = 10^{12} \tag{8}$$

respectively. Thus, the core storage unit can do work at a rate which is 36 times that of disk and 3,600 times that of drum. In addition, P_d/τ_d may be reduced markedly, in practice, because of the time required to move reading heads or disk; an additional loss of one order of magnitude is possible.

On the basis of these calculations, it appears there is serious mismatch between storage devices and that jobs might be run as much as 36 times faster if they could be executed completely in core with no use made of disk or drum. If it were known that a factor of 30 or even 10 in run time is the price to be paid for the flexibility now available in many computing facilities (which requires the use of the inexpensive storage available on drum and disk), then many of these facilties might exchange drum and disk for core storage and small in-core operating systems.

Suppose now that the disk unit in our typical facility is exchanged for a one microsecond core unit with 1/1,000 of the storage capacity (at no increase in cost). The new core unit would then have about 4×10^6 bits of storage or the capacity of the old core unit and a normalized computing power equal to that of the old core. If more core storage could be purchased, the normalized computing power would be proportionately increased and run time decreased.

These observations and the inequalities of (4) suggest that the use of a virtual memory machine may result in gross inefficiencies on jobs which are complex. This follows from the small normalized computing power of the auxiliary storage units and the fact that the serial organization of data in them necessitates a number of cycles which may grow linearly with the complexity of jobs. It does not suggest, however, that virtual memory machines are inefficient when the job mix has a uniform complexity.

5. Work on a GP Machine with 2-Level Storage

Assume that a GP machine with main and auxiliary storage units of different cycle lengths and computing powers is used to compute a function f which requires a work W_f. Under the assumption that f is computed in a batch mode we find approximate expressions for the minimum number of main memory cycles required to compute f. These expressions are obtained both when the capacity of main memory is limited and when it is not.

Let main memory have cycle length τ_m and computing power P_m. Let the corresponding parameters of auxiliary memory be τ_a and P_a and define $\rho = \tau_a/\tau_m$. Let the main and auxiliary memories execute T_m and T_a cycles, respectively, to compute f. Then the total time in main memory cycles, T, and the work done, W, to compute f are given by

$$\begin{aligned} T &= T_m + \rho T_a \\ W &= P_m T_m + P_a T_a \end{aligned} \tag{9}$$

Suppose also, that main memory has a capacity of M bits and that S_f bits, of program must be accessed to compute f. Then, if $S_f \le M$ and if the program can be put into main memory, the minimum value of T under the condition that $W = W_f$ is

$$T_{\min}^{(1)} = \begin{cases} W_f/P_m & P_m \ge P_a/\rho \\ W_f/P_a & P_m \le P_a/\rho \end{cases} \tag{10}$$

Thus, if the normalized computing power of main memory exceeds that of auxiliary memory, the function should be computed entirely in main memory and in auxiliary memory otherwise. This result is not very surprising.

Now suppose that $M < S_f$ and $P_m > P_a/\rho$ so that f would be computed in main memory if that were possible. But this is not possible since the program for f cannot fit into main memory and it implies that $T_a \ge (S_f - M)/b$ where b is the number of bits which can be read from auxiliary memory in one cycle. With this restriction on T and assuming that $W = W_f$, the minimum value of T,

$$T_{\min}^{(2)} \ge \frac{W_f}{P_m} + \left(\frac{S_f - M}{b}\right)\left(\rho - \frac{P_a}{P_m}\right) \tag{11}$$

The penalty for use of auxiliary memory will be large if the second term in the right hand side of (11) is much larger than W_f/P_m. If ρ is much larger than P_a/P_m, which is it for our typical machine described above, then the penalty is large if

$$\frac{W_f}{S_f} << \rho \frac{P_m}{b}\left(1 - \frac{M}{S_f}\right) \tag{12}$$

Again if $S_f >> M$, we compare W_f/S_f with $\rho P_m/b$ $\overset{which}{\wedge}$ is equal to 5.4×10^5 for disk auxiliary storage since $b = 20$ is the number of tracks from which the unit can read simultaneously. Thus, if the <u>work per bit of storage</u> W_f/S_f is less than 10^5 , the number of cycles required because auxiliary storage is used will be much larger than the number necessary when only main memory is used.

6. Conclusion

Through the use of computational work we have exhibited that the rate at which work can be done by storage devices can lead to mismatches between units which can result is gross inefficiencies. We have also shown that the work per bit of storage required to compute a funcion can be used to measure the penalty that must be incurred through use of auxiliary storage. If the work per bit is large,the penalty is small. We have also measured the work required to fetch from a store and shown that this leads to lower bounds on the number of cycles required to fetch a word in random-access and disk or drum units which is in agreement with practice.

Computational work and the notion of efficiency which it implies should be useful in machine design and use.

REFERENCES

1. J. E. Savage, "Computation on Finite Machines," submitted to JACM, September, 1970.

2. J. E. Savage, "The Complexity of Decoders: Part II: Computational work and Decoding Time," to be published, IEEE Transactions on Information Theory, January, 1971.