stichting
mathematisch
centrum

AFDELING INFORMATICA
(DEPARTMENT OF COMPUTER SCIENCE)

IW 118/79 SEPTEMBER

L.G.L.T. MEERTENS

RECURRENT ULTRACOMPUTERS ARE NOT LOG N-FAST

Preprint

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).

1980 Mathematics Subject Classification: 68C25

ACM-Computing Reviews-categories: 5.25, 5.22

Recurrent Ultracomputers are not log N-fast*)

by

L.G.L.T. Meertens

ABSTRACT

Ultracomputers are assemblages of processors that are able to operate concurrently and can exchange data through communication lines in, say, one cycle of operation. For physical reasons, the fan in/out of the processors must be limited. This imposes restrictions on the possible communication schemes. In order to have the ultracomputer operate efficiently as a whole, it is desirable that arbitrary exchanges of information between the processors can be effected in a small number of data shifts.

If a really huge ultracomputer is built, it would be nice if it could be constructed by coupling smaller ultracomputers, and so on. It will be shown that the latter desire conflicts to a certain extent with the earlier one.

KEY WORDS & PHRASES: *computational complexity, parallelism, ultracomputers.*

---

*) This research has been done while the author was visiting the Courant Institute of Mathematical Sciences, New York University, New York. It has been published there as Ultracomputer Note #2.
This report will be submitted for publication elsewhere.

1. INTRODUCTION

Ultracomputers [1] are assemblages of processors that are able to operate concurrently and can exchange data through communication lines in, say, one cycle of operation. For physical reasons, the fan in/out of the processors must be limited. This imposes restrictions on the possible communication schemes. In order to have the ultracomputer operate efficiently as a whole, it is desirable that arbitrary exchanges of information between the processors can be effected in a small number of data shifts.

If a really huge ultracomputer is built, it would be nice if it could be constructed by coupling smaller ultracomputers, which in turn are assembled from still smaller ultracomputers, and so on. It will be shown that the latter desire conflicts to a certain extent with the earlier one.

2. PARACOMPUTERS

For the purposes of this note, a *paracomputer* is a sequence of directed graphs. (Ultracomputers are paracomputers satisfying a restriction defined below.) Throughout the paper, the sequence G_D, $D = 0,1,\dots$ stands for a paracomputer. Each G_D is a pair $\langle P_D, L_D \rangle$, where P_D is the set of nodes (or "processors") of G_D, and L_D is a set of edges (or "lines") $\langle p_1, p_2 \rangle \in P_D \times P_D$. We define

$$N_D = \#P_D \quad \text{(the } \textit{size} \text{ of } G_D\text{)},$$

$$\phi_D = \max_{p \in P_D} \#\{\langle p_1, p_2 \rangle \in L_D \mid p_1 = p \text{ or } p_2 = p\}$$
$$\text{(the maximal } \textit{fan in/out} \text{ in } G_D\text{)},$$

$$C_D = \#L_D,$$

$$\Gamma_D = C_D / N_D.$$

To exclude uninteresting cases, it is assumed that $N_D \to \infty$. (Here and in the sequel, where limits or orders of magnitude are concerned, these are always understood to be with respect to $D \to \infty$.)

For a paracomputer to be an ultracomputer, the following requirement is imposed:

(UC) ϕ_D is bounded by some constant ϕ.

LEMMA 1. (UC) *implies that* Γ_D *is bounded.*

PROOF.

$$C_D = \#L_D = \#\{\langle p_1,p_2\rangle \in L_D\} \leq$$

$$\leq \frac{1}{2} \sum_{p \in P_D} \#(\{\langle p_1,p_2\rangle \in L_D \mid p_1 = p \text{ or } p_2 = p\}$$

$$\leq \frac{1}{2} \sum_{p \in P_D} \phi_D = \frac{1}{2} N_D \phi_D,$$

so

$$\Gamma_D = C_D/N_D \leq \frac{1}{2} \phi_D,$$

which by (UC) is bounded. □

The order of magnitude of the number of data shifts required to obtain an arbitrary permutation on P_D will determine how "fast" the paracomputer is. In order to express this in terms of the graph model, we must go through some definitions. The set of *basic permutations* on G_D is defined by

$$BP_D = \{\pi\colon \pi \text{ is a permutation on } P_D \mid \pi(p) = p \text{ or } \langle p,\pi(p)\rangle \in L_D \text{ for all } p \in P_D\}.$$

The permutations $PERM_D^{(d)}$ of *shift depth* d, $d \geq 0$, are inductively defined by:

$$PERM_D^{(0)} = \{\pi_I\}, \text{ where } \pi_I \text{ stands for the identity permutation,}$$

$$PERM_D^{(n+1)} = \{\beta \circ \pi\colon \beta \in BP_D,\ \pi \in PERM_D^{(n)}\} - PERM_D^{(n)}.$$

(Note that $BP_D = PERM_D^{(0)} \cup PERM_D^{(1)}$.)

The *shift depth* $sd_D(\pi)$ of a permutation π on P_D is defined by

$$\pi \in PERM_D^{(sd_D(\pi))} .$$

This definition may leave $sd_D(\pi)$ undefined for a given π, in which case we put $sd_D(\pi) = \infty$.

The *maximal shift depth* of G_D is now

$$M_D = \max_{\pi} sd_D(\pi),$$

where π ranges over all permutations on P_D. (The treatment of ∞'s should be obvious.)

A paracomputer is called $f(N)$-*fast* if $M_D = O(f(N_D))$. For example, the ultracomputer as defined in SCHWARTZ [1] has $N_D = 2^D$ and $M_D \leq 4D-3$ for $D \geq 1$, so it is log N-fast. In fact, it is easily seen to be *strictly* log N-fast, meaning that it is log N-fast but not $f(N)$-fast for any $f(N) = o(\log N)$. This is the best possible since no ultracomputer can improve on log N-fastness. Note that lower orders of $f(N)$ correspond to faster operation.

LEMMA 2. *Let the processors* P_D *of* G_D *be partitioned into two sets* S *and* T. *Let* $n = \min(\#S, \#T)$ *and* $c = \#(L_D \cap S \times T)$. *Then* $n \leq M_D \cdot c$.

PROOF. Let the permutations on P_D be extended in the natural way to map subsets of P_D on subsets. Define

$$a(\pi) = \#(\pi(S) \cap T).$$

We will first show that for $\beta \in BP_D$, $a(\beta) \leq c$. For

$$a(\beta) = \#(\beta(S) \cap T) = \#\{s \in S \mid \beta(s) \in T\}$$

$$= \{s \in S \mid \langle s, \beta(s)\rangle \in L_D \cap S \times T\} \leq \#(L_D \cap S \times T) = c.$$

Let π be a permutation such that $sd_D(\pi) = d$. It is claimed that $a(\pi) \leq d \cdot c$. The claim is easily shown correct by induction on d (and in fact, we have just shown it for the case $d = 1$). For $sd_D(\pi) = 0$, $\pi = \pi_I$, so $a(\pi) = \#(\pi_I(S) \cap T) = \#(S \cap T) = 0$. For $sd_D(\pi) > 0$, π can be written as $\beta \circ \pi'$, where $sd_D(\pi') = sd_D(\pi) - 1$ and $\beta \in BP_D$. Since

$$\pi'(S) = \pi'(S) \cap S \cup \pi'(S) \cap T \subset S \cup \pi'(S) \cap T,$$

$$\pi(S) = \beta \circ \pi'(S) = \beta(\pi'(S)) \subset \beta(S \cup \pi'(S) \cap T) \subset \beta(S) \cup \beta(\pi'(S) \cap T),$$

so

$$\beta \circ \pi'(S) \cap T \subset \beta(S) \cap T \cup \beta(\pi'(S) \cap T) \cap T \subset \beta(S) \cap T \cup \beta(\pi'(S) \cap T).$$

We have

$$a(\pi) = a(\beta \circ \pi') = \#(\beta \circ \pi'(S) \cap T) \leq \#(\beta(S) \cap T \cup \beta(\pi'(S) \cap T))$$

$$\leq \#(\beta(S) \cap T) + \#\beta(\pi'(S) \cap T) = \#(\beta(S) \cap T) + \#(\pi'(S) \cap T)$$

$$= a(\beta) + a(\pi').$$

Using $a(\beta) \leq c$, $sd_D(\pi') = sd_D(\pi) - 1$ and the inductive hypothesis, it follows that

$$a(\pi) \leq c + (sd_D(\pi) - 1) \cdot c = sd_D(\pi) \cdot c.$$

Next, choose (arbitrarily) two subsets $S' \subset S$ and $T' \subset T$, each of size n. Let π be any permutation such that $\pi(S') = T'$. Then

$$n = \#T' = \#(\pi(S') \cap T') \leq \#(\pi(S) \cap T) = a(\pi)$$

so, since M_D is an upper bound of the values of $sd_D(\pi)$,

$$n \leq a(\pi) \leq sd_D(\pi) \cdot c \leq M_D \cdot c,$$

which proves the lemma. □

REMARK. Although it may not be obvious from the formalism of the proof, the crucial idea is that at any shift β at most c items from S' may reach (their destination in) T across the "boundary" between S and T. It follows that the lemma will also hold if the processors are not forced to give up their current contents in passing it on to another processor and receiving data from a third. Even an unlimited memory capacity of the processors will not help; the bottle-neck is not the capacity of the processors but that of the lines.

3. RECURRENT PARACOMPUTERS

A *recurrent* paracomputer is a paracomputer obeying a recurrence relation

$$G_D = \langle P_{D-i_1} \cup \ldots \cup P_{D-i_n}, L_D^+ \cup L_{D-i_1} \cup \ldots \cup L_{D-i_n} \rangle.$$

In this scheme the processors P_{D-i_k} of constituent paracomputers G_{D-i_k} are considered distinct for different values of k, even if i_k is the same (by taking copies if necessary), so the unions involved are disjoint unions. We require, moreover,

$$n \geq 2 \quad \text{and} \quad 1 = i_1 \leq i_2 \leq \ldots \leq i_n.$$

(An additional requirement, which we do not need however, might be that $L_D^+ \subset P_D \times P_D$ is disjoint from each $P_{D-i_k} \times P_{D-i_k}$.) We shall write I for i_n.

To get the sequence started, we take $G_D = \langle\emptyset,\emptyset\rangle$ for $D < 0$ and $G_0 = \langle\{\Lambda\},\emptyset\rangle$. ($\Lambda$ stands for any "atom" to label the processor in the point set P_0, e.g. the null sequence. For the following considerations the choice of P_0 is immaterial, as long as $N_0 > 0$. Moreover, if $N_0 = 1$, the choice of L_0^+ is immaterial.)

For a recurrent paracomputer we have

$$N_D = 0 \qquad \text{for } D < 0;$$

$$N_0 = 1;$$

$$N_D = \sum_{k=1}^{n} N_{D-i_k} \qquad \text{for } D > 0.$$

Obviously, N_D is strictly monotone increasing for $D \geq 0$. The solution to a recurrence relation of this type can be written explicitly as

$$N_D = \sum_{j=1}^{I} a_j \lambda_j^D,$$

where the λ_j are the roots of the equation $\Sigma_{k=1}^{n} \lambda^{-i_k} = 1$. If λ is the largest of these roots, we have

(1) $$N_D = a\lambda^D + O(\mu^D)$$

for some positive a and some μ such that $|\mu| < \lambda$. (If there is a multiple root, the general explicit solution is slightly more complicated. We are concerned with the behaviour of N_D, however, and it can be shown that the largest root is larger than 1 and exceeds the other roots in absolute magnitude, and so has multiplicity 1.)

Putting $C_D = \#L_D$ and $c_D = \#L_D^+$, we also have

$$C_D = 0 \quad \text{for} \quad D < 0,$$

$$C_D = c_D + \sum_{k=1}^{n} C_{D-i_k} \qquad \text{for } D \geq 0.$$

The recurrence relation is solved by

(2) $$C_D = \sum_{q=1}^{D} N_{D-q} c_q.$$

(If $L_0^+ \neq \emptyset$, the summation should start with $q = 0$.)

To give an example of a recurrent paracomputer, consider

$$G_D = \langle P_{D-1}^{(0)} \cup P_{D-1}^{(1)}, L_D^+ \cup L_{D-1}^{(0)} \cup L_{D-1}^{(1)} \rangle.$$

The superscripts (0) and (1) serve to distinguish the two copies of G_{D-1}. If p is a processor of P_{D-1}, the corresponding processors of $P_{D-1}^{(0)}$ and $P_{D-1}^{(1)}$ are written p0 and p1, respectively. L_D^+ is then defined as

$$\{\langle p0,p1\rangle : p \in P_{D-1}\} \cup \{\langle p1,p0\rangle : p \in P_{D-1}\}.$$

So $N_D = 2^D$. Since $\phi_D = 2D$, this recurrent paracomputer is not an ultracomputer. It is easily shown to be strictly log N-fast. G_D is isomorphic to a hypercube (with edges running both ways) of dimension D.

4. MAIN RESULT

<u>THEOREM</u>. *Recurrent ultracomputers are not* log N-*fast.*

<u>PROOF</u>. By contradiction. Let the sequence G_D be a log N-fast recurrent ultracomputer. We have $M_D = O(D)$, so at most a finite number of the values of M_D is infinite. If this should be the case, we augment the corresponding L_D^+ to make M_D finite. This does not influence property (UC). Now, for some $\alpha > 0$, $M_D < \alpha D$.

We can partition P_D into two sets, $S = P_{D-i_1}$ and $T = P_{D-i_2} \cup \ldots \cup P_{D-i_n}$. From $I = \max i_k$, $k = 1,\ldots,n$, we have $\min(\#S,\#T) \geq N_{D-I}$. Each L_{D-i_k} contains members of $P_{D-i_j} \times P_{D-i_j}$ only, so members of $S \times T$ contained in $L_D = L_D^+ \cup L_{D-i_1} \cup \ldots \cup L_{D-i_n}$ are members of L_D^+. Consequently, $\#(L_D \cap S \times T) \leq \#L_D^+ = c_D$. Application of Lemma 2 yields now

$$N_{D-I} \leq M_D c_D.$$

Using $M_D < \alpha D$ and (2), we obtain for Γ_D

$$\Gamma_D > \frac{1}{\alpha} \sum_{q=1}^{D} \frac{N_{D-q} N_{q-I}}{q N_d} .$$

Since $N_{D-q} N_{q-I} / N_D \to a\lambda^{-I}$, we are led to rewrite this as

$$\Gamma_D > \frac{1}{\alpha} a\lambda^{-I} \sum_{q=1}^{D} \frac{1}{q} + \frac{1}{\alpha} \sum_{q=1}^{D} \frac{1}{q} \left(\frac{N_{D-q} N_{q-I}}{N_D} - a\lambda^{-I} \right).$$

From (1) it is clear that the sum in the second term has a finite limit, whereas the first term is clearly unbounded, so Γ_D is unbounded. Together with Lemma 1 this yields a contradiction. □

REMARK. The possibility is still left open that recurrent ultracomputers might exist that are $(\log N)^{1+\varepsilon}$-fast for arbitrarily small $\varepsilon > 0$. Note in fact that $\Sigma q^{-(1+\varepsilon)}$ is bounded. A mere existence proof, e.g. by enumerating combinations, would not be very helpful; for an ultracomputer to be manageable the lines should definitely exhibit some simple pattern. Note, moreover, that the criterion of boundedness of Γ_D as applied is relatively weak; for example, if c_D is constant, the reasoning in the proof of the theorem fails completely to reveal that the corresponding ultracomputer is at best N-fast, for no contradiction is obtained concerning the boundedness of Γ_D for even $(\log N)^{1+\varepsilon}$-fastness (although the contradiction follows immediately from the intermediate $N_{D-I} \le M_D c_D$). It seems therefore entirely plausible that the result of this note could be drastically sharpened.

REFERENCE

[1] SCHWARTZ, J.T., *Ultracomputers*, Preprint, Computer Science Department, Courant Institute of Mathematical Sciences, New York University, New York, 1979.