Efficiency of Universal Parallel Computers
(Extended Abstract)

by

Friedhelm Meyer auf der Heide
Johann Wolfgang Goethe-Universität Frankfurt
Fachbereich Informatik
6 000 Frankfurt a.M.
Fed. Rep. of Germany

Abstract:
We consider parallel computers (PC's) with fixed communication network with bounded degree. We construct a universal PC with $n^{1+O(1/\log\log(n))}$ processors which can simulate each PC with n processors with a time loss of $O(\log\log(n))$. This improves a result of [1] where a time loss of $O(\log(n))$ was achieved but only using $O(n)$ processors. Furthermore we prove a time-processor trade-off for a very general type of universal PC's, which includes thatone above. This generalizes a result for a simpler type of simulations presented in [2], where also all results of this paper are included.

Introduction: Galil and Paul dealt in [1] with parallel computers (PC's) with fixed communication network with bounded degree:

A PC M is specified by a graph with bounded degree and by processors which are attached to the vertices of the graph. We suppose that these processors are Random Access Machines (see [3]) which my in addition to their usual instructions read in one step the content of a fixed register - the communication register - of one of its neighbouring processors. M has fixed input - and outputprocessors. We assume M to be syncronized.

A multi-purpose PC (MPC) is a PC whose processors are universal Random Access Machines (see [3]). We say, M_0 can simulate a PC M with time loss k, if there is a program for M_0 (i.e. for every processor of M_0) such that M_0 initialized with this program simulates M and the time it needs to simulate T steps of M is at most $k \cdot T$, i.e. the time for simulating one step is at most k on an average.
M_0 is called n-universal with time loss k, if it can simulate each PC with n processors and degree c with time loss k, where $c>2$ is fixed all over this paper. In [1], a n-universal PC M_0 with $O(n)$ processors and time loss $O(\log(n))$ is constructed.

In the first chapter, we construct a n-universal PC with $n^{1+O(1/\log\log(n))}$ processors but with a time loss of only $O(\log\log(n))$.

In the second chapter, we present a time -processor trade-off for n-universal

PC's M_0, which use simulations with the following property: Let M be simulated for T steps. Then at every time $t \leq T$, each processor of M is simulated by at least one processor of M_0, its representants at time t. If P and Q are neighbouring processors of M, then for every representant of P at time t, there must be a path to some representant of Q at time (t-1) along which the communication is simulated. The maximal length of such a path for some neighbouring processors P and Q let be K_t. Then the time loss of the simulation is

$$\frac{1}{T} \sum_{i=1}^{T} k_t.$$

The simulations of the universal PC from the first chapter are of this type. We prove that every n-universal PC with m processors and time loss k fullfils that $m \cdot k = \Omega(n \log(n)/\log \log(n))$.
In [2] also a trade-off $m \cdot k = \Omega(n \log(n))$ is proved for the case that M_0 only uses simulations in which the representants at time t for some processor are identical for all t. The n-universal PC from [1] fits to this type but not thatone from the first chapter of this paper.

Chapter 1: A fast universal PC.

The basic network of the universal PC we want to construct is a generalization of a permutation nerwork, we call it a distributor.

This is a MPC M_0 with m distinguished processors, its base $B=[1,m]$ $(=\{1,\ldots,m\})$, which are both input -and output- processors. M_0 has the property, that there is a program for M_0 for an arbitrary disjoint partition $A_1,\ldots,A_m$ of B such that M_0 started with $x_i \in N^*$ (*) in processor i of B, $i=1,\ldots,m$, computes y_j in the j-th processor of B with $y_j=x_i$ iff $j \in A_i$ for all $i,j \in [1,m]$. We than say, M_0 distributes $(x_1,\ldots,x_m)$ according to $A_1,\ldots,A_m$.
Note that some A_j's may be empty.

Let G_m be the graph with vertex set $V=\{c_{ij}, i=1,\ldots,m, j=0,\ldots,\lceil \log(m) \rceil -1\}$. $\{c_{ij}, c_{i'j'}\} \subset V$, $j \leq j'$, is an edge in G_m, if $j'=j+1$ and either $i=i'$ or $|i-i'|=2^{j'}$ or if $j=j'=0$ and $|i-i'|=1$. The MPC W_m is specified by G_m and the base $B=\{c_{i,0}, i=1,\ldots,m\}$. (All its processors are universal Random Access Machines).
Figure 1 shows W_7.

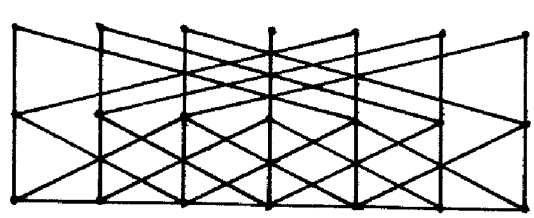

Figure 1: The MPC W_7.

(*) $N^* = \bigcup_{i \geq 0} N^i$, N is the set of non-negative integers.

Because of the similarity of W_m to the Waksman Permutation Network (see [4]) one can prove the following (see [2]):

Theorem 1: W_m is a m-distributor with the properties:

P1: W_m has $m\lceil \log(m) \rceil$ processors and degree 6.

P2: For $a,b \in [1,m]$, $a \leq b$, the MPC whose graph is the subgraph of G_m with vertex set $\{c_{ij}, i=a, a+1,\ldots,b; j=0,\ldots,\lceil \log(b-a+1) \rceil -1\}$ is identical to W_{b-a+1}.

P3: For every disjoint partition $A_1,\ldots,A_m$ of $[1,m]$, $(x_1,\ldots,x_m) \in (N^*)^m$ can be distributed according to $A_1,\ldots,A_m$ in $O(\log(m)+s)$ steps, where s is the maximum length of the x_i's.

Now we shall show how W_m (for a suitable m) can simulate a PC M with n processors $P_1,\ldots,P_n$ and graph G with degree c. (We identify the processors and the correspending vertices of G). For some P_i and $q \in N$ let $U_q(P_i)$ be the set of all processors of M which can be reached along a path of length at most q from P_i. Let $f,g,h:N \rightarrow N$ be functions such that $f(n)<n$ and $g(n) \geq n \cdot g(f(n))$ for all $n>1$ and $\#U_{h(g)}(P_i) \leq q$ for all $q \leq n$ and $\frac{h(n)}{h(f(n))} \in N$ for all $n \geq 1$.

We want to simulate M in $W_{g(n)}$.

For $i=1,\ldots,n$ let M_i be a PC with f(n) processors from $P_1,\ldots,P_n$ including those from $U_{h(f(n))}(P_i)$. The graph of M_i is the restriction of G on the processors of M_i. Let $\overline{W}_1,\ldots,\overline{W}_n$ be n exemplaries of $W_{g(f(n))}$ in $W_{g(n)}$. We can find them because of P2 of theorem 1 and the definition of g.

The following simple lemma is the main observation for our algorithm:

Lemma 1: If for some $i \in [1,m]$, M and M_i execute $h(f(n))$ steps then the processors P_i in M and M_i resp. have executed the same computation.

The simulation of h(n) steps of M by $W_{g(n)}$ works as follows:

If $n \leq c$, then use any simulation which uses a number of steps only dependent on n.
If $n>c$, execute $\frac{h(n)}{h(f(n))}$ times the following three parts: (note that $\frac{h(n)}{h(f(n))} \in N$.)

Part 1: For each $i \in [1,n]$ simulate recursively $h(f(n))$ steps of M_i in $\overline{W}_i$.

Remark: By lemma 1, there is a processor of $\overline{W}_i$ which simulates P_i correctly relative to M. This is the main representant of P_i. But in $\overline{W}_i$ and in other $\overline{W}_j$'s, too, there are processors which simulate P_i but make mistakes during the simulation. These are its potential representants.

Part 2: For each $i \in [1,n]$ transport the information about the last $h(f(n))$ steps P_i has executed from its main representant to its potential representants.

Remark: This can be done because $W_{g(n)}$ is a g(n)-distributor.

Part 3: Each potential representant of some P_i uses the information got in part 2 for computing the right configuration of P_i relative to M.

Obviously we have simulated h(n) steps of M. Let now $p \in N$, $p>1$ be fixed. We may choose $f(n) \approx n^{1/p}$, $h(n) \approx \alpha \cdot \log(n)$ for some suitable $\alpha>0$ and $g(n) = \lfloor n^{1+\frac{1}{p-1}} \rfloor$.

Let T(n) be the time necessary to simulate h(n) steps of a PC with n processors by $W_{g(n)}$.

Then the above algorithm shows

$T(n) \leq a_0$ for some $a_0>0$, if $n<c$. If $n>c$, then

$$T(n) \leq \frac{h(n)}{h(f(n))} \; [T(f(n)) + O\,(\log(g(n)) + h(n))]$$

$$\approx p \cdot T(n^{1/p}) + O(\log(n)).$$

Thus $T(n) = O(\log(n) \log\log(n))$ which quarantees a time loss of $O(\log\log(n))$.

We can improve this result in the following way. At the top of the above recursion we choose $f(n) \approx n^{1/\log\log(n)}$ instead of $n^{1/p}$. For the resulting subproblems of size $n^{1/\log\log(n)}$ we apply the above algorithm.

Thus we obtain for the size of M_0: $g(n) \approx n \cdot g(n^{1/\log\log(n)}) = n \cdot (n^{1/\log\log(n)})^{1+1/(p-1)}$ $\leq n^{1+\beta/\log\log(n)}$ for some $\beta>0$. Thus we may choose $g(n) = \lfloor n^{1+\beta/\log\log(n)} \rfloor$.

As we may choose $h(n) = O(\log(n))$ we obtain:

$$T(n) = O\left(\log(n) \cdot \frac{\log\log(n)}{\log(n)}\right) \cdot (T(n^{1/\log\log(n)}) + O(\log(n)))$$

$$= O(\log\log(n)) \cdot (\log(n^{1/\log\log(n)}) \cdot \log\log(n^{1/\log\log(n)}) + O(\log(n)))$$

$$= O(\log\log(n) \cdot \log(n)).$$

Theorem 2: Let $g(n) = \lfloor n^{1+\beta/\log\log(n)} \rfloor$ for some suitable $\beta>0$.

Then $W_{g(n)}$ is n-universal with time loss $\log\log(n)$.

Chapter 2: The Time - Processor Trade-Off

Let M_0 be n-universal and M a PC with n processors [1,n], the processors of M_0 let be [1,m]. The degree of the graph G_0 of M_0 let be d, thatone of the graph G of M let be c.In the sequel we shall identify the graph of a PC to the PC itself.

A simulation of T steps of M by M_0 is a sequence $(B_{1,t},\dots,B_{n,t},W_t)_{t\leq T}$ with the properties:

For every $t \leq T$, $B_{1,t},\dots,B_{n,t}$ are pairwise disjoint subsets of the vertex set [1,m] of M_0. $B_{i,t}$ is the set of representants of the vertex i of M at time t. W_t is a set of pathes in M_0. It contains for every representant x of some i at time t a path from x to one representant of each neighbour of i in M at time t-1. If k_t is the length of a longest path of W_t, then k_t is called the t-time loss and $\frac{1}{T}\sum_{i=1}^{T} k_t$ the time loss of the simulation. If $h := \max\{\sum_{i=1}^{n} \#B_{i,t},\ t \leq T\}$, then we say that the simulation uses h representants.

A simulation with time loss at most k using at most h representants is called a (h,k)-simulation.

Obviously, the simulations from chapter 1 are (g(n),O(log log(n)) - simulations. It seems to be very unlikely that reasonable simulations can be constructed which are not of this type. Therefore we call a n-universal PC which only uses (h,k)-simulations n-universal of the general type with time loss k using h representants. For such PC's we prove:

Theorem 3: Let M_0 be a n-universal PC of the general type with m processors and time loss k, using h representants, then $h \cdot k = \Omega(n \log(n)/\log\log(n))$ or $m = n^{\Omega(n \log(n)/h)}$.

As $h \leq m$, we obtain the following time-processor trade-off:

Theorem 4: Let M_0 be a n-universal PC of the general type with m processors and time loss k, then $m \cdot k = \Omega(n \log(n))/\log\log(n))$.

Now we prove theorem 3.

The idea of this proof is as follows:

To each (h,k)-simulation of a graph with n vertices by M_0, we attach a fragment, i.e. an object which still specifies the graph being simulated. For technical reasons we only consider graphs which contain a balanced, binary tree. The set of these graphs let be called E_n. Now the number Y of fragments of (h,k)-simulations of graphs from E_n is an upper bound for the number of graphs from E_n which can be simulated by M_0 with time loss k using h representants.

(Note that this bound is smaller then the number of (h,k)-simulations, because different such simulations may have the same fragment.)

On the other hand we bound $\#E_n$ from below. As every graph from E_n must be simulated by M_0 with a (h,k)-simulation, $y \geq \#E_n$, which will prove the theorem.

We first state the bound for $\#E_n$. A proof can be found in [2].

Lemma 2: $\#E_n \geq n^{\frac{c-3}{2}n} \cdot 2^{-an}$ for some a>0.

Before defining and counting the fragments, we state some estimations from [2] which we will need in the sequel.

Lemma 3: a) For all $k,n \in \mathbb{N}$, $1 \leq k \leq n$, $\binom{n}{k} \leq n^k$.

b) $\#\{(a_1,\ldots,a_n) \in (N \setminus \{0\})^n \mid \sum_{i=1}^{n} a_i \leq h\} \leq 2^h$

c) Let $(a_1,\ldots,a_n),(b_1,\ldots,b_n) \in (N \setminus \{0\})^n$.

Let $p \in N$ such that $p \cdot a_i \geq b_i$ for every $i \in [1,n]$,

and $\sum_{i=1}^{n} a_i \leq h$, $\sum_{i=1}^{n} b_i \leq h$. Then $\prod_{i=1}^{n} \binom{p \cdot a_i}{b_i} \leq e^{2h} \cdot p^h$.

Now we define the fragments. Let D be a balanced, binary tree with vertices $[1,n]$. D has depth $\lfloor \log(n) \rfloor$.

Now let $A \in N$ be fixed, $A \leq n$. A will be specified later.

Let $r \in N$ and $V_1,\ldots,V_r$ be r subsets of $[1,n]$ of cardinality A, which cover $[1,n]$, such that for every $i \in [1,r]$, the subgraph of D induced by V_i is a balanced, binary tree of depth $\lfloor \log(A) \rfloor$. Obviously, $V_1,\ldots,V_r$ can be chosen such that $r \leq \frac{2n}{A}$ and every $i \in [1,n]$ is contained in at most two of the V_i's.

We assume that $T \geq 2\lfloor \log(A) \rfloor + 1$. Let $(B_{1,t},\ldots,B_{n,t},W_t)_{t \leq T}$ be a (h,k)-simulation for some graph from E_n. For $t \in [1,T]$ let k_t be the t-time loss of the simulation.

We count the number of graphs for which there is a (h,k)-simulation as follows: For some t_0 we count the number of possible choices of $B_1,\ldots,B_n = B_1^{t_0},\ldots,B_n^{t_0}$ in a strategy. Afterwards we estimate the number of possible choices of sets S of edges of graphs which can be simulated by a strategy with the above representants at time t_0 and (t_0+1)-time loss k_{t_0+1}. Unfortunately, this method, i.e. the choice of $(B_1,\ldots,B_n,S)$ as fragments, is to weak for our purpose, because there are too many choices for $B_1,\ldots,B_n$. Therefore we first fix the representants $B_1',\ldots,B_r'$ of r suitably chosen vertices of G - one from each V_i - at time $t_0-2\lfloor \log(A) \rfloor$. There number is not too large if t_0 is chosen reasonably. As all considered graphs contain a balanced binary tree, after having fixed $B_1' \ldots B_r'$ the number of choices of $B_1,\ldots,B_n$ decreases considerably.

Formally a fragment is defined as follows:

Let $t_0 \in [2\lfloor \log(A) \rfloor, T-1]$ be chosed such that $\sum_{t=t_0-2\lfloor \log(A) \rfloor+1}^{t_0+1} k_t)$ is minimal relative to the choice of t_0. This sum is called R_0.

Now a fragment of $(B_{1,t},\ldots,B_{n,t},W_t)_{t \leq T}$ is specified by a tupel

$(B_1,\ldots,B_n,B_1',\ldots,B_r',S)$ as follows:

$(B_1,\ldots,B_n) = (B_{1,t_0},\ldots,B_{n,t_0})$.

If $j \in [1,r]$ and $i_j \in V_j$ such that $B_{i_j,t_0-2\lfloor \log(A) \rfloor}$ has a minimal cardinality relative to the choice of i_j, then $B_j' = B_{i_j,t_0-2\lfloor \log(A) \rfloor}$.

$S := \{(x,y) \in [1,m]^2 / x \in B_{i,t_0}$ and there is an $i \in [1,n]$,

such that there are two (t_0+1)-transport pathes in W_{t_0+1} which join

x and y to the minimal element of $B_{i,t_0+1}\}$.

Let R be the number of graphs from E_n for which there is a (h,k)-simulation in M_0, and Y the number of fragments of (h,k)-simulations for graphs from E_n.

Obviously a fragment still specifies the graph being simulated. Therefore, the following holds:

Prop. 1: $R \leqq Y$.

Before we bound Y, we state some easy properties of the fragment described above.

Prop. 2: a) $K_{t_0+1} \leqq R_0 \leqq 2k(2\lfloor \log(A) \rfloor + 1)$ b) $\sum_{i=1}^{r} \#B'_i \leqq \frac{2h}{A}$.

c) For every $j \in [1,r]$ and every $i \in V_j$, $B_i \subset U_{R_0}(B'_j)$. (Let G=(V,E) be a graph, $B \subset V$, $a \in N$, then $U_a(B)$ is the set of vertices from V, which can be reached by a path of length at most a from some vertex from B.)

Now we bound Y.

Prop. 3: $y \leqq m^{\frac{2h}{A}} \cdot d^{(h+2cn)(5k \log(A))} \cdot e^{4h} \cdot (\frac{h}{n})^n$.

Proof: First we bound the number Y_1 of all tupels $(B_1,\ldots,B_n,B'_1,\ldots,B'_r)$, which belong to a fragment of a(h,k)-simulation of a graph from E_n.

Claim 1: $y_1 \leqq m^{\frac{2h}{A}} \cdot e^{4h} \cdot d^{h(R_0+1)}$.

Proof: Let the cardinalities $h_1,\ldots,h_n$, $h'_1,\ldots,h'_r$ of $B_1,\ldots,B_n$, $B'_1,\ldots,B'_r$ be fixed.

- By lemma 2.b) there are at most 2^{2h} possible choices of $h_1,\ldots,h_n$, $h'_1,\ldots,h'_r$.
- There are at most $\prod_{i=1}^{r} \binom{m}{h'_i}$ possible choices of $B'_1,\ldots,B'_r$.
- For $j \in [1,r]$ let $V'_j \subset V_j$ chosen such that $V'_1,\ldots,V'_r$ form a disjoint partition of $[1,n]$.

By prop. 2.c) it follows for every $j \in [1,r]$ and every $i \in V'_j$: There are at most

$$\binom{h'_j \cdot d^{R_0+1}}{h_i}$$ possible choices for B_i.

Therefore we obtain:

$$Y_1 \leq 2^{2h} \cdot \prod_{q=1}^{r} \binom{m}{h'_q} \prod_{j=1}^{r} \prod_{i \in V'_j} \binom{h'_j \cdot d^{R_0+1}}{h_i}$$

Applying lemma 3.a) and c) we obtain

$$Y_1 \leq 2^{2h} \cdot m^{\sum_{i=1}^{r} h'_i} \cdot d^{h(R_0+1)} \cdot e^{2h}.$$

By prop. 2.b), $\sum_{i=1}^{r} h'_i \leq \frac{2h}{A}$, which proves claim 1.

Now we bound for some fixed sets $B_1,\ldots,B_n$, $B'_1,\ldots,B'_r$ the number Y_2 of fragments of (h,k)-simulations which can be formed by these sets.

<u>Claim 2</u>: $Y_2 \leq (\frac{h}{n})^n \cdot d^{2(k_{t_0+1}+1)cn}$.

<u>Proof</u>: If $(B_1,\ldots,B_n , B'_1,\ldots,B'_r , S)$ is a fragment of a (h,k)-simulation it follows for S:

- There are at most n different first components of pairs occuring in S, one in each B_i, $i \in [1,n]$.
- At most c second components belong to each first component x.

They are contained in $U_{2(k_{t_0}+1)}(x)$.

For $i \in [1,n]$ let $h_i = \#B_i$. Then there are at most $\prod_{i=1}^{n} h_i \leq (\frac{h}{n})^n$ possible choices for the n first components of the pairs of S.

In order to fix the second components for some first component x, there are at most

$$\binom{d^{2k_{t_0+1}+1}}{c}$$ possible choices.

Therefore it follows by lemma 3.a):

$$Y_2 \leq (\frac{h}{n})^n \cdot \binom{d^{2k_{t_0+1}+1}}{c}^n$$

$$\leq (\frac{h}{n})^n \cdot d^{(2k_{t_0+1}+1)cn}.$$

As $Y \leq Y_1 \cdot Y_2$, prop. 3 is proved by claim 1 and 2 and the bounds for R_0 and k_{t_0+1} from prop. 2.a).

Now we are able to prove theorem 3.

By lemma 2, $\#E_n \geq n^{\frac{c-3}{2} n} \cdot 2^{-an}$.

W.l.o.g. we may assume that $c \geq 4$.

As pointed out when describing the idea of the proof, we get:

$$n^{\frac{c-3}{2} n} \cdot 2^{-an} \leq 2^{\frac{2h}{A}} \cdot e^{4h} \cdot d^{(h+2cn)(5k \log(A))} \cdot (\frac{h}{n})^{n}.$$

Therefore,

$$m \geq 2^{\frac{A}{2h}\left(\left(\frac{c-3}{2}\right) n \log(n) - a n - 4h \log(e)\right)} \cdot 2^{\frac{A}{2h}\left(-\log(d)(h+2cn)(5k\log(A)) - \log(\frac{h}{n}) n\right)}.$$

Let $a_1, a_2 > 0$ be chosen such that $\frac{c-3}{2} > a_2 = a_1(4\log(e) + 5(2c+1)\log(d))$.

and let $h \cdot k \cdot \log(A) \leq a_1 n \log(n)$.

Then $\log(\frac{h}{n}) \leq \log\log(a_1 n)$ and it follows:

$$m \geq 2^{\frac{A}{2h}\left(\frac{c-3}{2} - a_2\right) n \cdot \log(n) - a \cdot n - n \cdot \log\log(a_1 n)}$$

$$\geq n^{\Omega(\frac{A\,n}{h})}.$$

Now we choose $A = \lfloor \log(n) \rfloor$ and obtain theorem 3.

References:

[1] Z. Galil, W.J. Paul: A Theory of Complexity of Parallel Computation. Proc. of the 13th Annual ACM Symp. on Theory of Computing, Milkwaukee, May 1981, pp. 247-262.

[2] F.Meyer auf der Heide: Efficiency of Universal Parallel Computers. Interner Bericht des Fachbereichs Informatik der J.W. Goethe-Universität Frankfurt, 2.82.

[3] W.J. Paul: Komplexitätstheorie. Teubner Verlag, Stuttgart, 1978.

[4] A. Waksman: A Permutation Network. Journal of the ACM, 15(1) (1968) pp. 159-163.