Extending the Hong-Kung Model to Memory Hierarchies*

John E. Savage

Brown University,Providence, Rhode Island 02912

Abstract. The speed of CPUs is accelerating rapidly, outstripping that of peripheral storage devices and making it increasingly difficult to keep CPUs busy. Consequently multi-level memory hierarchies, scaled to simulate single-level memories, are increasing in importance. In this paper we introduce the Memory Hierarchy Game, a multi-level pebble game that simulates data movement in memory hierarchies in terms of which we study space-time tradeoffs.
We provide a) a common generalization of the Hong-Kung and Paterson-Hewitt pebble models to the Memory Hierarchy Game, b) a greatly simplified proof of the Hong-Kung lower bound on I/O complexity that makes their result readily accessible, c) straight-line algorithms for a representative set of problems that are simultaneously optimal at each level in the memory hierarchy in their use of space and I/O and computation time, and d) an extension the game to block transfers of data between memories.

1 Introduction

In this paper we study tradeoffs between the number of storage locations (*space*) at each level of a memory hierarchy and the number of data movements (*I/O time*) between levels in the hierarchy. We develop upper and lower bounds on I/O time in terms of space. We model computations as pebblings of straight-line programs according to the rules of the **Memory Hierarchy Game** (MHG), a pebble game in which different kinds of pebbles represent storage locations at different levels in a memory hierarchy. The MHG generalizes to L levels the two-level game introduced by Hong and Kung [8]. Straight-line computations are modeled by directed acyclic graphs (dags). Not only are many important computational science algorithms described by dags, efficient prefetching in large memory hierarchies may require that all large computations be straight-line.

The rules of the MHG assume that data migrate up and down the hierarchy. Input data resides initially in the highest level memory (the Lth) and the values of all output vertices reside there at the end of a computation. The location of a datum in the jth memory is denoted by placing a level-j pebble on that vertex. Movement up and down the hierarchy is modeled by either replacing a

* This work was supported in part by the Office of Naval Research under contract N00014-91-J-4052, ARPA Order 8225 and by NSF under Grant MIP-902570.

pebble at one level with one at an adjacent level or by possibly adding such a pebble. Level-1 pebbles model data storage in a register. A level-1 pebble can be placed on a vertex (a **computation step**) that has no pebbles only if all of its predecessors carry level-1 pebbles, thereby modeling the requirement that a value of an operation can be computed by a CPU only if all of its operands are present in registers. A **level-l I/O operation** is the placement of a level-$(l-1)$ pebble on a vertex carrying a level-l pebble (an input from level-l) or the placement of a level-l pebble on a vertex carrying a level$(l-1)$ pebble (an output to level-l). We allow an unlimited number of level-L pebbles.

We consider two variants of the MHG, the **standard game** in which highest level pebbles be used on intermediate vertices of a dag $G = (V, E)$ and the **I/O-limited game** in which highest level pebbles cannot be used this way. The latter is appropriate when there is a large gap between the access times of the highest and next-highest level memory units because in this case data-independent prefetching is essential to avoid the large delays required by data-dependent fetching. The two-level I/O-limited game is actually the Paterson-Hewitt [11] **red-pebble game** while the two-level standard game is the Hong-Kung **red-blue pebble game**. Thus, the MHG combines elements of both games and generalizes them to multiple levels.

A pebble strategy is **minimal** if the number of highest level I/O operations is minimized after which the number of I/O operations is minimized at successively lower levels and, finally, the number of computation steps is minimized. This definition of minimality reflects the fact that the time to access data on memories increases very rapidly with their level in the hierarchy.

We develop methods for deriving upper and lower bounds on performance that are applied to a representative set of problems consisting of matrix multiplication, the Fast Fourier Transform and permutation and merging networks. We also generalize the model to block transfers. If the storage unit at level l holds p_l words, and the I/O time at level l, T_l, is the number of times blocks of b_l words move between the storage units at levels $l-1$ and l, then we establish a framework in which to derive lower bounds on T_l in terms of b_l and s_{l-1}, the storage capacity of all units up to and including the $(l-1)$st. Under weak conditions on b_l and for the problems under consideration, we show that our lower bounds on T_l can be achieved up to multiplicative constants for all levels simultaneously.

This approach is illustrated by the problem of multiplying matrices. We show that using any variant of the classical algorithm to multiply two $n \times n$ matrices, T_l is proportional to $\Theta(n^3/(b_l\sqrt{s_{l-1}}))$ when $s_{l-1} = O(n^2)$. If t_l is the time (relative to that of the fastest memory) for a level-l block I/O operation, a memory hierarchy will behave as single flat memory if $t_l \ll b_l\sqrt{s_{l-1}}$ for $2 \leq l \leq L$.

Related Research Aggarwal and Vitter [3] examined a two-level memory in which P B-item blocks can be transferred in each step, obtaining tight bounds for sorting-related problems. They did not handle the I/O-limited case or multiple levels. Aggarwal, Alpern, Chandra and Snir [1] introduced the hierarchical memory model (HMM), which treats memory as a linear array with cost $f(x)$ to access location x in the array, and obtained tight bounds for a number of

problems and a number of cost functions. They don't handle blocks, nor handle the I/O-limited case or large discontinuities in the storage access time between levels.

Aggarwal, Chandra and Snir [2] introduced the BT model, an extension of the HMM model supporting block transfers in which the time to move a block of size b ending at location x is $f(x)+b$. They establish tight bounds on computation time for problems including matrix transpose, FFT, and sorting using a number of cost functions. They allow blocks to be arbitrarily large and problem dependent but do not handle the I/O-limited case nor large discontinuities in access time.

Alpern, Carter and Feig [4] introduced the uniform memory hierarchy (UMH) which has uniform exponential values for memory capacity, block size, and the time to move a block between levels. They allow I/O overlap between levels and determine conditions under which matrix transposition, matrix multiplication and Fourier transforms can and cannot be done efficiently.

Savage and Vitter [12] extend the one-level Paterson-Hewitt model [11] to support parallel pebbling and the Hong-Kung model to support contiguous block I/O. Vitter and Shriver [16] examine block transfers in three parallel disk memory systems and present a randomized version of distribution sort that meets the lower bounds for these models of computation. Nodine and Vitter [10] give an optimal deterministic sorting algorithm for these memory models. The models have the limitations described above.

2 The Memory Hierarchy Game

The Memory Hierarchy Game (MHG) formally defined below captures the essential features of serial computers that use storage units organized into levels and in which data moves between levels from the highest to the lowest level and back. The highest level storage unit models an archival store with large access time whereas the lowest level unit models a fast memory used by the CPU for all its computations.

The L-level Memory Hierarchy Game (MHG) is played on dags with p_l pebbles at level l, $1 \leq l \leq L-1$, and an unlimited number of pebbles at level L. It has **resource vector** $\underline{p} = (p_1, p_2, \ldots, p_{L-1})$, where $p_j \geq 1$ for $1 \leq j \leq L-1$, and uses $s_l = \sum_{j=1}^{l} p_j$ pebbles at level l or less. Its rules are given below.

R1. (Computation Step) A first-level pebble can be placed on or moved from a predecessor to any vertex all of whose immediate predecessors carry first-level pebbles.

R2. (Pebble Deletion) Except for level-L pebbles on output vertices, a pebble at any level can be deleted from any vertex.

R3. (Initialization) A level-L pebble can be placed on an input vertex at any time.

R4. (Input from Level-l) For $2 \leq l \leq L-1$, a level-$(l-1)$ pebble can be placed on any vertex carrying a level-l pebble.

Standard Game

R5. (Output to Level-l) For $2 \leq l \leq L$, a level-l pebble can be placed on any vertex carrying a level-$(l-1)$ pebble.

I/O-limited Game

R5. (Output to Level-l) For $2 \leq l \leq L-1$, a level-l pebble can be placed on any vertex carrying a level-$(l-1)$ pebble.

R6. (I/O-limitation) Level-L pebbles can only be placed on input vertices and output vertices carrying level-$(L-1)$ pebbles.

3 Computational Inequalities

In this section we derive generic lower bounds on the number of I/O and computation steps required by a minimal pebbling of a dag $G = (V, E)$ in the MHG. We assume throughout that all vertices of a dag $G = (V, E)$ are reachable from input and output vertices.

Consider a pebbling $\mathcal{P}(\underline{p}, G)$ of the dag G in the L-level MHG with resource vector $\underline{p}$. A minimal pebbling is one that successively minimizes the number of level-l I/O operations at decreasing levels starting at level L. The **computation time** of a pebbling, $T_1^{(L)}(\underline{p}, G)$, is the minimal number of lowest level pebblings in a minimal pebbling and the **level-l I/O time**, $T_l^{(L)}(\underline{p}, G)$, is the minimal number of level-l I/O operations, $2 \leq l \leq L$.

The following result shows that if the number of pebbles available at or below a given level, is large enough, no I/O operations at the next level are necessary except on input and output vertices.

Lemma 1. *Let S_{min} be the minimum number of pebbles to pebble $G = (V, E)$ in the red-pebble game. If the number of pebbles at level $k < L$ or less, s_k, exceeds $S_{min} + (k-1)$, a minimal pebbling $\mathcal{P}(\boldsymbol{p}, G)$ with resource vector $\boldsymbol{p}$ in the L-level MHG does not perform I/O operations at level $k+1$ or higher except on inputs and outputs.*

Because every input must be read from level L and every output written to level L, $T_l^{(L)}(\underline{p}, G)$ is at least equal to the number of input and output vertices of G for $2 \leq l \leq L$. Also $T_1^{(L)}$ is at least the number of non-input vertices $|V^*|$ of G.

The following definition of the S-span of a graph abstracts ideas used by Hong and Kung [8] and is used to derive lower bounds on the I/O time of minimal pebblings of dags.

Definition 2. Given a dag $G = (V, E)$ the **S-span of** G, $\rho(S, G)$, is the number of computation steps on G with the red-pebble game in a minimal pebbling maximized over all initial placements of S red pebbles. The pebblings are done assuming that pebbles are left on output vertices.

The following theorem generalizes the Hong-Kung [8] lower bound on I/O time for the two-level MHG and provides a new and much simpler proof. Aggarwal and Vitter [3] have given a simple proof of the Hong-Kung lower bound for the FFT dag.

Theorem 3. *Consider a minimal pebbling of the dag $G = (V, E)$ in the standard MHG with resource vector $\underline{p}$ using $s_l = \sum_{j \le l} p_l$ pebbles at level l or less. Let $T_l^{(L)}(\underline{p}, G)$ be the number of I/O operations at level l, $2 \le l \le L$, and let $T_1^{(L)}(\underline{p}, G)$ be the number of computations steps used in this pebbling. Then, the following lower bound on $T_l^{(L)}(\underline{p}, G)$, $2 \le l \le L$, must be satisfied whether the MHG is I/O-limited or not:*

$$\lceil T_l^{(L)}(\underline{p}, G)/s_{l-1} \rceil \rho(2s_{l-1}, G) \ge T_1^{(L)}(\underline{p}, G) \ge |V^*|$$

Proof Sketch. Since fewer pebblings are done at each level for the standard game, lower bounds are derived for this case. s_{l-1} is the number of pebbles at all levels up to and including level $l-1$. Let C be a minimal pebbling of G. Divide C into consecutive sequential sub-pebblings $\{C_1, C_2, \ldots, C_h\}$ where each sub-pebbling has s_{l-1} level-l I/O operations except possibly the last which has no more such operations. Thus $h = \lceil T_l^{(L)}(\boldsymbol{p}, G)/s_{l-1} \rceil$.

We develop an upper bound Q to the number of computation steps in each sub-pebbling. This number multiplied by the number h of sub-pebblings is an upper bound to the total number of computation steps, $T_1^{(L)}(\boldsymbol{p}, G)$, performed by the pebbling C. It follows that $Qh \ge T_1^{(L)}(\boldsymbol{p}, G)$.

The upper bound on Q is developed by adding s_{l-1} new level-$(l-1)$ pebbles and showing that we may use these new pebbles to move all I/O operations at level l or higher in a sub-pebbling C_t to either the beginning or end of the sub-pebbling without changing the number of computation steps or I/O operations at level $l-1$ or less. Thus, we move without changing them all computation steps and I/O operations at level $l-1$ or lower to a *middle interval* of C_t in between the higher-level I/O operations.

An upper bound to Q is obtained by observing that at the start of the pebbling of the middle interval of C_t there will be at most $2s_{l-1}$ pebbles at level $l-1$ or less on G, at most s_{l-1} original pebbles plus a like number of new pebbles. Any output vertex pebbled in this interval must have a level-$(l-1)$ or lower pebble on it at the end of the interval since the pebbling is minimal. Clearly, the number of computation steps on vertices pebbled in the middle interval is largest when all $2s_{l-1}$ pebbles of levels $l-1$ or less on G are treated as level-1 pebbles. It follows that at most $\rho(2s_{l-1}, G)$ computation steps can be done on G in C_t since C_t is part of a minimal pebbling. This completes the proof of this theorem.

In the standard game each vertex of G is pebbled once in a computation step. However, when the I/O limitation applies, some vertices may have to be repebbled. The following theorem relates the number of moves in an L-level

game to the number in a two-level game and allows us to use prior results. The proof is by simulation of an L-level MHG with a two-level MHG.

Theorem 4. *Let $S = s_{L-1} = \sum_{j=1}^{L-1} p_j$. The following inequalities hold for $2 \leq l \leq L-1$ when the graph G is pebbled in the L-level MHG, whether I/O limited or not, with resource vector $\boldsymbol{p}$:*

$$T_l^{(L)}(\boldsymbol{p}, G) \geq T_L^{(L)}(\boldsymbol{p}, G) \geq T_2^{(2)}(S, G)$$
$$T_1^{(L)}(\boldsymbol{p}, G) \geq T_1^{(2)}(S, G) \geq |V^*|$$

Here $T_1^{(2)}(S, G)$ and $T_2^{(2)}(S, G)$ are the number of computation and I/O operations, respectively, in the red-blue pebble game played on G with S red pebbles. If MHG is played with the I/O-limitation, these two measures are computed under the I/O-limitation. $|V^*|$ is the number of non-input vertices of G.

4 Matrix Multiplication

Consider a matrix multiplication algorithm conforming to the classical algorithm. All products of pairs of entries in the two matrices A and B are formed and combined in independent binary addition trees. We develop tight upper and lower bounds on the I/O and computation time for the standard MHG as well as the I/O-limited MHG. Developing good upper bounds under the I/O limitation is related to the challenging problem of deriving fast algorithms for matrix multiplication.

Lemma 5 [8]. *The S-span of any graph G associated with the classical algorithm to multiply two $n \times n$ matrices with the binary operations of addition and multiplication of component values satisfies $\rho(S, G) \leq 2S^{3/2}$ for $S \leq n^2$.*

Theorem 6. *Let G be any graph consistent with the classical algorithm to multiply two $n \times n$ matrices. Let it be pebbled in the standard MHG with the resource vector $\underline{p}$. Let $s_l = \sum_{j=1}^{l} p_j$ and let k be the largest integer such that $s_k \leq 3n^2$. When $p_1 \geq 3$ there is a pebbling of G such that the following bounds hold simultaneously:*

$$T_l^{(L)}(\underline{p}, G) = \begin{cases} \Theta(n^3) & l = 1 \\ \Theta(n^3/\sqrt{s_{l-1}}) & 2 \leq l \leq k+1 \\ \Theta(n^2) & k+2 \leq l \leq L \end{cases}$$

Proof Sketch. We note that any graph G consistent with the classical matrix multiplication algorithm has $|V^*| = \Theta(n^3)$ non-input vertices. The first lower bound follows from the fact that we have to pebble $\Theta(n^3)$ vertices with level-1 pebbles to pebble the graph. The second follows from Theorem 3 and Lemma 5. The third lower bound follows from Theorem 4.

The pebbling strategy that simultaneously achieves these lower bounds up to constant multiplicative factors uses the standard representation of a $n \times n$

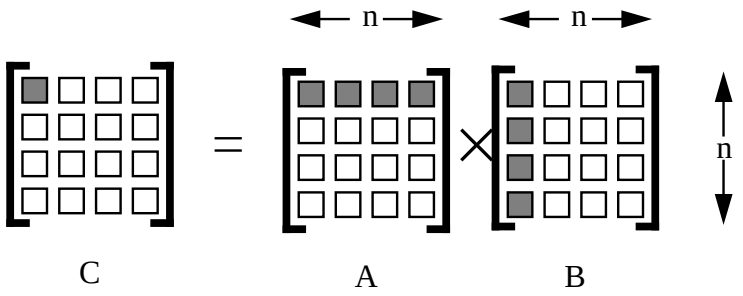

Fig. 1. Pebbling schema for matrix multiplication.

as an $n/m \times n/m$ matrix of $m \times m$ matrices when m divides n, as suggested by Figure 1. In turn the submatrices are themselves recursively decomposed as submatrices where the matrix sizes are chosen to minimize the amount of memory fragmentation. The innermost matrices are $r_1 \times r_1$ and are contained in $r_2 \times r_2$ matrices which are finally contained in $r_k \times r_k$ where the r_i are set as shown below and k is chosen to be the largest integer such that $r_k^2 \leq n^2$. (Without loss of generality we assume that $r_k = n$, that is, we assume that A, B and C are $r_k \times r_k$ matrices.)

$$r_i = \begin{cases} \lfloor \sqrt{s_1/3} \rfloor & i = 1 \\ r_{i-1} \lfloor \sqrt{(s_i - i + 1)}/(\sqrt{3} r_{i-1}) \rfloor & i \geq 2 \end{cases}$$

Our pebbling strategy is recursive. It is done using two sets of pebbles, a reserve set containing one pebble per level except the first, and the remainder. With the remaining pebbles, recursively pebble the product of two $r_j \times r_j$ submatrices A_1 and B_1 of A and B under the assumption that A_1 and B_1 have pebbles at level j or lower on their entries initially. When pebbles at levels j or lower are placed on inputs of A_1 and B_1, we exhaust pebbles at each level before using pebbles at a lower level. Since $s_j \geq 3r_j^2 + j - 1$, this guarantees that if a submatrix of A_1 or B_1 does not have all its input vertices covered with pebbles at level $j - 1$ or less, there will be low-level pebbles that can be moved to them using the reserve pebbles without having to remove pebbles from other vertices.

We close this section with a lower bound for the I/O-limited version of the game using a new lower bound [13] of the Grigoryev style [7] on the I/O time in the red-pebble game for matrix multiplication of the form $T_2^{(2)}(S, G) \geq n^3/\sqrt{32(S+1)}$.

Theorem 7. *Let G be any dag associated with matrix multiplication of $n \times n$ matrices and let it be pebbled under the* **I/O limitation**. *If $S = s_{L-1} \leq n$, then the time to pebble G at the lth level, $T_l^{(L)}(\underline{p}, G)$, must satisfy the following relation for $1 \leq l \leq L$:*

$$T_l^{(L)}(\underline{p}, G) = \Omega(\frac{n^3}{\sqrt{S}})$$

5 The Fourier Transform

The discrete Fourier transform (DFT) on n inputs is defined by the matrix-vector multiplication $A\underline{x}$ with a Vandermonde matrix. The fast Fourier transform (FFT) graph is the well known fast implementation of the DFT. As suggested in Figure 2, for $1 \leq j \leq d-1$, the FFT graph $F^{(d)}$ on 2^d inputs can be represented as the composition of 2^{d-j} disjoint "top" FFT graphs on 2^j inputs $\{F_{t,p}^{(j)} \mid 0 \leq p \leq 2^{d-j}-1\}$ with 2^j disjoint "bottom" FFT graphs on 2^{d-j} inputs $\{F_{b,m}^{(d-j)} \mid 0 \leq m \leq 2^j - 1\}$, where the mth input vertex of $F_{t,p}^{(j)}$ is the pth output vertex of $F_{b,m}^{(d-j)}$ for $0 \leq m \leq 2^j - 1$ and $0 \leq p \leq 2^{d-j} - 1$.

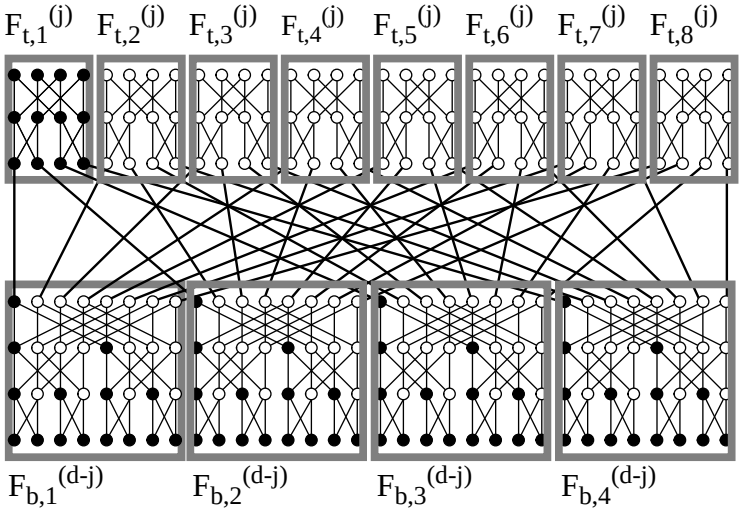

Fig. 2. Decomposition of the FFT graph $F^{(5)}$. Edges between shaded boxes identify vertices common to two FFT subgraphs. The ordering of their endpoints defines a matrix transposition.

The following bound on the S-span of the FFT dag is implicit in the work of Hong and Kung [8] and explicit in the work of Aggarwal and Vitter [3].

Lemma 8. *$\rho(S, F^{(d)})$ on the 2^d-input FFT graph $F^{(d)}$ satisfies $\rho(S, F^{(d)}) \leq 2S \log S$ when $S \leq n$.*

Theorem 9. *Let the FFT graph on $n = 2^d$ inputs, $F^{(d)}$, be pebbled in the standard MHG with resource vector $\underline{p}$. Let $s_l = \sum_{j=1}^{l} p_j$ and let k be the largest*

integer such that $s_k \leq n$. When $p_1 \geq 3$ there is a pebbling of $F^{(d)}$ such that the following relations are simultaneously satisfied:[2]

$$T_l^{(L)}(\underline{p}, F^{(d)}) = \begin{cases} \Theta(n \log n) & l = 1 \\ \Theta(n \log n / \log s_{l-1}) & 2 \leq l \leq k+1 \\ \Theta(n) & k+2 \leq l \leq L \end{cases}$$

Proof. The lower bounds follow directly from elementary considerations and Theorem 3 and Lemma 8. We exhibit a pebbling strategy giving upper bounds matching these lower bounds for all $1 \leq l \leq L$. Our pebbling strategy is based on the decomposition of Figure 2.

If e divides d, decompose $F^{(d)}$ into a collection of subgraphs $F^{(e)}$ each of which can be pebbled level by level with 2^e+1 level-1 pebbles without repebbling any vertex. This observation is used in our hierarchical pebbling strategy.

The following non-decreasing sequence $\underline{d} = (d_1, d_2, \ldots, d_{L-1})$ of integers is used to describe an efficient pebbling strategy for $F^{(d)}$. Let $d_0 = 1$ and $d_1 = \lfloor \log(s_1 - 1) \rfloor \geq 1$ where $s_1 = p_1 \geq 3$. Define m_r and d_r for $1 \leq r \leq L-1$ by

$$m_r = \lfloor \frac{\lfloor \log \min(s_r - 1, n) \rfloor}{d_{r-1}} \rfloor, \; d_r = m_r d_{r-1}$$

Note that d_r is divisible by d_{r-1} and $s_r \geq 2^{d_r} + 1$ when $s_r \leq n+1$. From this it follows that $d_r \geq (\log \min(s_r - 1, n))/4$. The sizes of the sub-FFT graphs are chosen so that a sufficient number of pebbles is available, including $l-1$ reserve pebbles, to pebble $F^{(d_r)}$ using s_{l-1} pebbles at levels $l-1$ or less.

Through induction it is possible to show that the number of level-l I/O operations on $F^{(d_r)}$, $n_l^{(r)}$, satisfies the following inequality

$$n_l^{(r)} \leq 2 \frac{d_r}{d_{l-1}} 2^{d_r}$$

for $2 \leq l \leq r+1$. In addition, each input vertex is pebbled once with pebbles at each level.

Let k be the largest integer such that $s_k \leq n = 2^d$. If $s_k \neq 2^d$, it is possible to extend the above argument. It follows that the total number of level-l I/O operations, $2 \leq l \leq k+1$, is at most

$$(\lceil d/d_k \rceil) 2^{d-d_k} \frac{2d_k}{d_{l-1}} 2^{d_k} = \lceil d/d_k \rceil \frac{2d_k}{d_{l-1}} 2^d \leq \frac{4d2^d}{d_{l-1}}$$

The desired conclusions follows from the observation above that $d_l \geq (\log(s_l - 1))/4$ when $s_l - 1 \leq n$.

We now state lower bounds on the number of I/O and computation steps for DFTs realized by straight-line algorithms for the DFT. Corresponding upper bounds are stated for FFT graphs. The lower bounds are larger than the lower bounds of Theorem 9 when the total number of pebbles at all levels but the

[2] All logarithms are to base 2.

highest, S, satisfies $S \leq n/\log n$ where n is the number of inputs to the DFT. The lower bounds apply to any straight-line program for the DFT, not just linear straight-line programs, as shown by Tompa [14], due to a new unpublished result [13].

Theorem 10. *Let $FFT(n)$ be any dag associated with the DFT on n inputs when realized by a linear straight-line program. Let $FFT(n)$ be pebbled under the I/O limitation with resource vector $\underline{p}$. Let $s_l = \sum_{j=1}^{l} p_j$. If $S = s_{L-1} \leq n$, then the time to pebble $FFT(n)$ at the lth level, $T_l^{(L)}(\underline{p}, FFT(n))$, must satisfy the following relation for $1 \leq l \leq L$:*

$$T_l^{(L)}(\underline{p}, FFT(n)) = \Omega(\frac{n^2}{S})$$

Also, when $n = 2^d$, there is a pebbling of the FFT graph $F^{(d)}$ such that the following relations hold simultaneously:

$$T_l^{(L)}(\underline{p}, F^{(d)}) = \begin{cases} O(\frac{n^2}{S} + n \log S) & l = 1 \\ O(\frac{n^2}{S} + n\frac{\log S}{\log s_{l-1}}) & 2 \leq l \leq L \end{cases}$$

when $S \geq 2 \log n$.

6 Permutation and Merging Networks

Consider merging networks $BS^{(d)}$ based on bitonic sorting networks [9, pp. 632] and permutation networks $P^{(d)}$ based on three back-to-back FFT graphs [17]. Replacing comparators in $BS^{(d)}$ with two-input butterfly graphs produces an FFT with edge directions reversed. It is immediate from the layered cross-product representation of FFT graphs [6] that $BS^{(d)}$ is isomorphic to the FFT graph. Thus, the bounds for the FFT apply directly to $BS^{(d)}$.

It can be shown that lower bounds for the FFT apply to $P^{(d)}$ because $P^{(d)}$ reduces to $F^{(d)}$ be eliminating edges and coalescing chains to single edges. Since our pebbling strategy for the FFT graph in the standard MHG advances pebbles level-by-level, using higher-level pebbles sparingly to achieve the lower bounds, it follows that the pebbling strategy for the FFT graph is directly applicable to this graph.

Theorem 11. *Let $P^{(d)}$ be pebbled in the standard L-level MHG with resource vector $\underline{p}$. Let $s_l = \sum_{j=1}^{l} p_j$ and let k be the largest integer such that $s_k \leq n$. When $p_1 \geq 3$ there is a pebbling of $P^{(d)}$ such that the number of pebblings at each of the L levels, $\{T_l^{(L)} \mid 1 \leq l \leq L\}$, simultaneously satisfy the following relations:*

$$T_l^{(L)}(\underline{p}, P^{(d)}) = \begin{cases} \Theta(n \log n) & l = 1 \\ \Theta(n \log n / \log s_{l-1}) & 2 \leq l \leq k+1 \\ \Theta(n) & k+2 \leq l \leq L \end{cases}$$

When the I/O limitation applies and s_{L-1} is too small, the I/O and computation time to pebble the permutation and sorting networks can be much larger than that to pebble the FFT graph. For example, Carlson and Tompa together show the following result, which implies that $P^{(d)}$, which has three FFT graphs back-to-back, requires at least as much I/O time:

Lemma 12 [5,14,15]. *Let G be the graph consisting of two back-to-back copies of the FFT graph $F^{(d)}$ on $n = 2^d$ inputs. Then the number of second-level I/O operations when the* **I/O limitation** *applies in the red-pebble game when played with S level-1 pebbles satisfies the following inequality:*

$$T_2^{(2)}(S, G) \geq \Omega(n^3/S^2 + (n^2 \log n)/S)$$

7 Generalization to Block-I/O

Data is typically moved in blocks between memories in a hierarchy; data must fetched from the same block in which it was stored. Our lower bounds can be generalized to the block-I/O case by dividing the number of I/O operations by the size b_l of blocks moving between levels $l-1$ and l. This lower bound can be achieved for matrix multiplication because data is always read from the higher-level memory in the same way every time. For the FFT graph in the standard MHG instead of pebbling FFT subgraphs on 2^{d_r} inputs, we pebble b_l FFT subgraphs on $2^{d_r}/b_l$ inputs (assuming that b_l is a power of 2). This allows all the data moving back and forth between memories in blocks to be used and accommodates the transposition mentioned in the caption to Figure 2. This provides an upper bound of $O(n \log n/(b_{l-1} \log(s_{l-1}/b_{l-1})))$ on the I/O time at level l. Clearly, when b_{l-1} is much smaller than s_{l-1}, say $b_{l-1} = O(\sqrt{s_{l-1}})$, the upper and lower bounds match.

8 Conclusions

The Memory Hierarchy Game has been introduced and new lower bounds developed on the computation and I/O time needed at each level of a memory hierarchy to compute functions from straight-line programs. We have studied two variants of the game, one in which the highest level memory can be used for intermediate results and another in which it cannot. We have demonstrated the utility of this new game by showing that our lower bounds can be met by pebbling strategies for matrix multiplication, the Fourier transform, as well as merging and permutation networks.

Since it is very expensive to increase the speed of CPUs, parallel machines are becoming increasingly more attractive. To fully exploit parallelism it will be necessary to provide high-speed parallel memory hierarchy systems. We need a much better understanding of parallel I/O systems if we are going to meet this challenge.

References

[1] A. Aggarwal, B. Alpern, A. Chandra, and M. Snir, "A Model for Hierarchical Memory," *Procs. 21st Annual ACM Symposium on Theory of Computing* (May 15-17, 1989), 305–314.

[2] A. Aggarwal, A. Chandra, and M. Snir, "Hierarchical Memory with Block Transfer," *Proc. 28th Annl. Symp. on Foundations of Computer Science* (October 1987), 204–216.

[3] A. Aggarwal and J. S. Vitter, "The Input/Output Complexity of Sorting and Related Problems," *Communications of the ACM* 31 (September 1988), 1116–1127.

[4] B. Alpern, L. Carter, and E. Feig, "Uniform Memory Hierarchies," *Proc. 31st Annual Symposium on Foundations of Computer Science* (October 22-24, 1990), 600–608.

[5] D. A. Carlson, "Time-Space Tradeoffs for Back-to-Back FFT Algorithms," *IEEE Trans. Computing* C-32 (1983), 585–589.

[6] S. Even and A. Litman, "Layered Cross Product - A Technique to Construct Interconnection Networks," *Proc. 4th Ann. ACM Symp. on Parallel Algorithms and Architectures* (June 29 - July 1, 1992), 60–69.

[7] D. Y. Grigoryev, "An Application of Separability and Independence Notions for Proving Lower Bounds of Circuit Complexity," *Notes of Scientific Seminars, Steklov Math. Inst.* 60 (1976), 35–48.

[8] J. -W. Hong and H. T. Kung, "I/O Complexity: The Red-Blue Pebble Game," *Proc. 13th Ann. ACM Symp. on Theory of Computing* (May 11-13, 1981), 326–333.

[9] F. T. Leighton, in *Introduction to Parallel Algorithms and Architectures*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1992.

[10] M. H. Nodine and J. S. Vitter, "Large-Scale Sorting in Parallel Memories (Extended Abstract)," *Procs. 3rd Annual ACM Symposium on Parallel Algorithms and Architectures* (July 21-24, 1991), 29–39.

[11] M. S. Paterson and C. E. Hewitt, "Comparative Schematology," *Proc. Proj. MAC Conf. on Concurrent Systems and Parallel Computation* (June 1970), 119–127.

[12] J. E. Savage and J. S. Vitter, "Parallelism in Space-Time Tradeoffs," in *Advances in Computing Research*, F. P. Preparata, ed., 1987, 117–146.

[13] J. E. Savage, *A Generalization of Grigoryev's Space-Time Tradeoff Method*, Unpublished manuscript, March 1995.

[14] M. Tompa, "Time-Space Tradeoffs for Computing Functions, Using Connectivity Properties of Their Circuits," *JCSS* 20 (1980), 118–132.

[15] M. Tompa, "Corrigendum: Time-Space Tradeoffs for Computing Functions, Using Connectivity Properties of Their Circuits," *JCSS* 23 (1981), 106.

[16] J. S. Vitter and E. A. M. Shriver, "Optimal Disk I/O with Parallel Block Transfer," *Procs. 22nd Annual ACM Symposium on Theory of Computing* (May 1990), 159–169.

[17] C. L. Wu and T. Y. Feng, "The Universality of the Shuffle-Exchange Network," *IEEE Trans. Computing* C-30 (May 1981), 324–332.

This article was processed using the LaTeX macro package with LLNCS style