

Markov Analysis of Multiple-Disk Prefetching Strategies for External Merging

Vinay Sadananda Pai*
Schlumberger Dowell
P.O. Box 2710
Tulsa, Oklahoma 74101

Alejandro A. Schäffer†
Department of Computer Science
Rice University
P.O. Box 1892
Houston, TX 77251

Peter J. Varman‡
Department of Electrical and Computer Engineering
Rice University
P.O. Box 1892
Houston, TX 77251

Abstract

Multiple-disk organizations can be used to improve the I/O performance of problems like external merging. Concurrency can be introduced by overlapping I/O requests at different disks and by prefetching additional blocks on each I/O operation. To support this prefetching, a memory cache is required.

Markov models for two prefetching strategies are developed and analyzed. Closed-form expressions for the average parallelism obtainable for a given cache size and number of disks are derived for both prefetching strategies. These analytic results are confirmed by simulation.

Keywords : Parallel I/O, Prefetching, Disk Cache, External Merging, Declustered Disks, Markov Chains.

To appear in *Theoretical Computer Science*
Short version in 1992 Intl. Conf. on Parallel Processing.

*Partially supported by an NSF Graduate Research Fellowship while at the ECE Department, Rice University.

†Partially supported by NSF Research Initiation Award CCR 9010534.

‡Partially supported by NSF and DARPA Grant CCR 9006300.

1 Introduction

Advances in processor architecture and integration technology have resulted in steady increases in processor speeds over the past several years. The performance of I/O subsystems, in contrast, has generally not kept pace with these improvements in processor performance. The data rates possible from single disks are limited by physical considerations such as the speed of disk rotation and the rate of head movement, and are unlikely to increase dramatically. As a consequence, there have been a number of recent proposals for the use of multiple disks to form high-performance I/O subsystems [9, 16, 18]. Performance evaluation of different multiple-disk systems, and associated management strategies have been studied in [19, 12, 17, 5, 6, 10], for example. A number of analytic studies of I/O performance for specific computational problems have been undertaken previously in [11, 20, 8, 1, 21, 2, 23, 22, 14, 4].

This paper suggests an analytic framework for the study of I/O parallelism by undertaking a specific case study of *prefetching* as a means for improving I/O performance in a multiple-disk environment. In particular, we study the tradeoff between the average disk parallelism and the cache size for a single-pass merge of D sorted runs using D concurrent disk units storing the input. The system model consists of an infinitely fast CPU, a RAM-based disk cache with a capacity of C blocks, and D disks containing one sorted run each. On a single I/O operation, at most D blocks, one from each disk, can be read and transferred to the cache concurrently.

The D -way merge algorithm operates as follows. A block from each run is brought into memory, and the records from each block are extracted and merged together in sorted order. When one of the input blocks is depleted, an I/O request is made for the next block from the run whose block was depleted. In a multiple disk system, the request for this next block can be overlapped with prefetching a block from each of the other disks. These prefetched blocks are held in the cache until they are required by the merge. If the cache is large enough to hold all prefetched blocks, then the total number of I/O accesses will be reduced by a factor of D over the case of a single disk, and a speedup of D in the I/O time will be expected¹.

We use the block random depletion model [11] in which the probability that the next block depleted comes from any particular run is uniformly $1/D$. A worst-case scenario of block depletion, in which all the blocks from run 1 are depleted, then all the blocks from run 2 are depleted, and so on, is not interesting from a prefetching point of view. Once the cache gets filled in the worst-case scenario, only the depleted block can be replaced, and there is no opportunity for parallelism.

¹The seek time reduction that arises when the runs are spread over several disks are not considered.

Therefore, we sought a random model of block depletion that would be useful to study *average-case* behavior, and we chose a model proposed previously in a related paper.

Ideally, we would like the prefetching strategy to read D blocks on each input operation; however, in practice, the cache can become filled, and it may not have space to store D additional blocks. We would like to know the expected number of blocks fetched on an input operation, which we can think of as the average parallelism obtained by using a machine with D disks instead of 1 disk. In particular, we analyze the relationship between the size of the disk cache and the average parallelism for two different prefetching strategies that can be easily implemented with a standard multi-way merge algorithm.

The motivation for the study is to provide a quantitative understanding of some of the tradeoffs involved when resources like main memory are allocated to tasks in a typical DBMS system. We therefore concentrate on analyzing simple enhancements of the standard multiway merge algorithm that is commonly employed in such systems. The speeds of the source of the input data and the destination of the sorted output are assumed to be fast enough that merging is the bottleneck. Such situations arise often in practice, as when the input data is available on multiple disks, and the output is fed to a further processing stage. The larger questions of whether algorithms other than the multiway merge may be more suitable for sorting or merging data are not addressed here. With the same perspective, we assume contiguous placement of the sorted runs on the disks, as is done in several commercial DBMS systems like IBM's DB2, SQL/DS for instance. In contiguous placement, a run is placed on a single disk, and occupies consecutive blocks on that disk. In contrast, in block-interleaved placement, consecutive blocks of a run are interleaved across the D disks. Issues related to block interleaving, including the degree of hardware support provided, the possibility of partial striping and the granularity are still the subject of active research, and beyond the scope of this paper.

Recent investigations of different methods for multiple-disk sorting include the works of Vitter and Shriver [23], and Nodine and Vitter [22, 14], who have presented new algorithms for external-sorting. Vitter and Shriver [23] have shown that a randomized version of distribution sorting can achieve an asymptotically optimal number of I/Os. Vitter and Nodine have discovered an ingenious deterministic, mergesort-like algorithm called Greed Sort. This algorithm achieves optimal asymptotic performance even for small cache sizes, but requires a larger constant in the number of I/Os required in any pass. More recently Nodine and Vitter [14] have discovered a deterministic version of distribution sort called Balance Sort, which reduces the constants associated with Greed Sort. While quite promising, these algorithms need to be evaluated to determine the ranges over which

they are superior to mergesort especially in real systems. For other works on the I/O complexity of sorting and merging, the reader is referred to the works of Kwan and Baer [11] and Salzberg [20] for single-disk systems, and to [15, 24] for multi-disk systems.

The two prefetching strategies are described in detail in Section 2. The first is a *randomized* (and greedy) algorithm that attempts to read one block each from as many disks as is possible, subject to the availability of free cache space. The second is a *deterministic* (and conservative) algorithm that reads concurrently only if all disks can be used; otherwise it does not perform any prefetching. In our analysis, we derive a closed-form expression for the average number of blocks read on an I/O operation as a function of the cache size and the number of runs. Surprisingly, the conservative strategy, which sometimes leaves available cache slots empty, attains slightly more parallelism than the greedy strategy for most reasonable choices of the number of disks and cache size. This theoretical result is confirmed by our simulations using nominal-sized data.

The rest of this paper is organized as follows. In Section 2 the two prefetching strategies are described in detail, and Markov models are developed for their analysis. Section 3 contains some mathematical definitions and combinatorial results used in the analysis. The first prefetching strategy is analyzed in Section 4 and the second in Section 5. In Section 6 we present simulation results which validate the analysis, and we conclude with Section 7.

2 System Models

In this section we describe two prefetching strategies that can be used to improve the I/O performance in implementing the multiway merge. For both these strategies we develop Markov models which are analyzed in later sections.

During the merge, whenever a block is depleted from run i , the cache is checked to see if the next block from run i is in the cache. If no block from run i is in the cache, a fetch operation is required. The block needed will be referred to as the *demand-fetch block*. To exploit the parallelism possible with the D disks, blocks from other disks can be *prefetched* along with the demand-fetch block, provided sufficient space is available in the cache to accommodate them.

The two prefetching strategies we analyze will be referred to as the *randomized prefetch* model and the *deterministic prefetch* model, respectively. Both models have identical behavior when sufficient cache memory is available to fetch a block from each disk. In this case, $D - 1$ blocks are prefetched along with the demand-fetch block. The models differ when the number of free cache blocks, F , is less than $D - 1$. In the randomized model, F of the $D - 1$ disks are randomly chosen

with equal probability; the demand-fetch block is fetched along with one block from each of these F disks. In the deterministic model, no prefetching is performed; only the demand-fetch block is fetched. The deterministic model fetches either D blocks or only 1 block, whereas the randomized model fetches between 1 and D blocks, depending on the amount of cache space available.

An I/O operation that fetches at most D blocks, one from each disk, will be charged unit cost. The D disks can independently fetch blocks from different disk locations, but they will be assumed to initiate and complete I/O operations as a group. This assumption relaxes any temporal transients without invalidating the cache behavior being studied. If the full parallelism of the D disks can be used on every I/O operation, the cost of the merge operation will be reduced by a factor of D over the single-disk case. Ideally, a speedup of D can be obtained with a cache of unbounded size. As the cache size is reduced, the actual speedup will be lower, depending on the average I/O parallelism of each I/O operation.

The two prefetching models are described by the pseudocode in Figure 2.1. Initially, the cache is loaded with one block from each run. At the start of any iteration of the loop, at least one block from each run will be present in the cache. A random model of block depletion is assumed [11]. The leading block from each run is a candidate for being depleted next. One of the D leading blocks is chosen with equal probability $1/D$, and depleted. If that run still has blocks present in the cache, no I/O operation is required (we call this a *d-transition*). However, if that run has no more cached blocks, an I/O operation will be required to retrieve the next block from that run before the merge can continue. If there is enough cache space to read a block from each disk, D blocks are fetched (called an *f-transition*); if not, the action taken is different for the two prefetching models. In the deterministic model only one block from the depleted run is fetched (called an *r-transition*), while in the randomized model enough blocks to fill up the cache are fetched (called a *p-transition*).

The dynamic behavior of both systems can be modeled mathematically by Markov chains. Each state in the model will represent the number of cache blocks allocated to each run at the start of each iteration of the loop in Figure 2.1. Figure 2.2a shows a sample cache state for $D = 5$ disks. Each execution of the loop of the program causes a transition to a different state.

To describe the states of the system and the transitions, we use the following notation.

Definition 2.1 Let C denote the size of the cache in blocks, and D the number of runs. Let $\mathbf{s} = [s_1, \dots, s_i, \dots, s_D]$ be an integer vector with D components. Define $\delta(\mathbf{s}) = (C - \sum_{i=1}^D s_i)$ to be the number of free cache blocks in \mathbf{s} . Define $\gamma(\mathbf{s})$ to be the number of components of \mathbf{s} that are 1.

Every cache state will be a D -component integer vector. From the program one can see that

```

/* C is the cache size in blocks. D is the number of runs */

Initial State:
Add the first block from each run to the cache.
for (i=1..D) a[i]= 1;
/* a[i] represents the number of blocks from run i that reside in the cache. */
num_free_cache= C-D;

do forever
{
  /* RECORD STATE HERE */
  Randomly choose a run, j, from which to deplete a block.
  a[j]= a[j]-1;
  if (a[j]==0) { /* A run has emptied */
    if (num_free_cache  $\geq$  (D-1)) {
      Fetch D blocks, one from each run; /*f-transition */
      for (i=1..D) a[i]= a[i]+1;
      num_free_cache= num_free_cache - (D-1);
      /* Only (D-1) new prefetches have been added. */
      /* The demand-fetch replaces the cache block depleted from run j. */
    }
    else { /* Less than (D-1) free cache blocks */
      switch (prefetching model) {
        case DETERMINISTIC:
          Fetch 1 block from run j; /* r-transition */
          a[j]= a[j]+1;
          /* The demand-fetch replaces the cache block depleted from run j. */
          break;
        case RANDOM:
          Let F= num_free_cache;
          Randomly choose F runs from the D-1 runs, i=1..D, i  $\neq$  j
          Let these be  $i_1, i_2, \dots, i_F$ 
          Fetch 1 block from run j and 1 block from each of these F runs;
          a[j] = a[j] + 1; for (k=1, ..., F) a[i_k] = a[i_k] + 1; /* p-transition */
          num_free_cache= 0;
          break;
      };
    };
  }
  else /* Next block of run j available in cache */
    num_free_cache= num_free_cache+1; /* d-transition */
}

```

Figure 2.1: System model: random and deterministic prefetching models

any vector that represents a cache state must satisfy Definition 2.2 below. Every component must be at least 1, at least one component must be equal to 1, and the number of free cache blocks must be between 0 and $C - D$. However, not all vectors satisfying these three conditions are necessarily reachable from the initial state. The set of reachable states differs for the two prefetching models; the reachable states are characterized precisely in Sections 4 and 5.

Definition 2.2 A *cache state* is an integer vector $\mathbf{s} = [s_1, \dots, s_j, \dots, s_D]$ such that every component $s_j \geq 1$ for $1 \leq j \leq D$, $\gamma(\mathbf{s}) \geq 1$, and $0 \leq \delta(\mathbf{s}) \leq C - D$.

The transitions from state $\mathbf{s} = [s_1, \dots, s_j, \dots, s_D]$ to state \mathbf{t} can be characterized as follows:

1. **Depletion (d-transition):** A block is depleted from run j , and all runs still have at least one cached block. This transition is enabled only if $s_j > 1$. Following the transition, $\mathbf{t} = [s_1, \dots, s_j - 1, \dots, s_D]$ (Figure 2.2(b)).
2. **Fetch (f-transition):** A depletion from run j occurs, and run j no longer has any cached blocks. A fetch is required, and all D blocks are cached. This transition is enabled only if $s_j = 1$ and $\delta(\mathbf{s}) \geq D - 1$. Following the transition, $\mathbf{t} = [s_1 + 1, \dots, s_{j-1} + 1, 1, s_{j+1} + 1, \dots, s_D + 1]$ (Figure 2.2(c)).
3. **Replenish (r-transition):** A depletion from run j occurs, and run j no longer has any cached blocks. However, the cache does not contain enough free blocks for a fetch of D blocks. Instead, the next block from run j is fetched. This transition is enabled only if $s_j = 1$ and $\delta(\mathbf{s}) < D - 1$. Following the transition, $\mathbf{t} = \mathbf{s}$ (Figure 2.2(d)).
4. **Partial Fetch (p-transition):** A depletion from run j occurs, and run j no longer has any cached blocks. However, the cache has only $F < D - 1$ free cache blocks. The next block from run j and one block from each of F runs chosen randomly from the other $D - 1$ runs are fetched. This transition is enabled only if $s_j = 1$ and $\delta(\mathbf{s}) < D - 1$. A p-transition always fills the cache, since exactly $F = \delta(\mathbf{s})$ blocks are fetched in addition to the block from run j .

Every state has exactly D possible depletions that initiate a transition to another state. Each depletion is equiprobable with probability $1/D$. In most cases, the depletion determines which transition is made, and the transition also has probability $1/D$. The only exception is when a p-transition occurs in the randomized model.

Figure 2.3 shows the Markov chain for the case of 3 disks and 7 cache blocks for the randomized model, while Figure 2.4 shows the Markov chain for the deterministic model. In both figures the

three applicable types of transitions are shown. The three states marked as invalid in Figure 2.4 cannot be reached from the initial state. Also, states have been labeled as being either interior or edge states. These classifications are useful in Section 5.

Most states and transitions look the same in both Markov chains. For example, in both Markov chains, the state $[1, 2, 2]$ has three outgoing transitions, each with probability $1/3$. If a block is depleted from the second or third runs, a d-transition is made to $[1, 1, 2]$ or $[1, 2, 1]$, respectively. If a block is depleted from the first run, an I/O operation is required, since the first run has no cached blocks. Since $[1, 2, 2]$ has 2 free cache blocks, an f-transition will be made, leading to state $[1, 3, 3]$. The difference between the chains can be seen by considering the state $[3, 2, 1]$. In the randomized model $[3, 2, 1]$ has four outgoing transitions, while in the deterministic model it has three. If the leading block from the third run is depleted, the models differ. In the randomized model, two p-transitions are possible, to $[4, 2, 1]$ or to $[3, 3, 1]$. The choice of p-transition depends on a random decision on whether to use the last free block in the cache for a block from the first run or the second. Each occurs with equal probability ($1/2$), so the probability of the two p-transitions is $(1/3)(1/2) = 1/6$ for each. In the deterministic model, the depletion of the lone block from the third run forces an r-transition back to $[3, 2, 1]$; this r-transition has probability $1/3$.

3 Mathematical Preliminaries

In this section, some known combinatorial results about binomial coefficients and integer partitions are summarized. These results are used, in turn, to derive related mathematical formulas that will be applied directly in the analysis of Sections 4 and 5. (The reader may prefer to skim through this section and return to it when the results are used in subsequent sections.)

Lemma 3.1 [7, p. 169]

$$\sum_{k=0}^p \binom{p-k}{m} \binom{q+k}{n} = \binom{p+q+1}{m+n+1} \quad \text{integers } p, q \geq 0, n \geq q \geq 0$$

Lemma 3.2 [7, p. 169]

$$\sum_{k=-m}^n \binom{p}{m+k} \binom{q}{n-k} = \binom{p+q}{m+n} \quad \text{integers } m, n, p, q$$

Lemma 3.3 [7, p. 160]

$$\sum_{k=m}^n \binom{k}{m} = \binom{n+1}{m+1} \quad \text{integers } m \geq n \geq 0$$

Lemma 3.4 [7, pp. 173–175]

$$\sum_{j=0}^k \left[\frac{\binom{k}{j}}{\binom{D-1}{j}} \right] = \binom{D}{D-k} \quad \text{integers } D-1 \geq k \geq 0$$

Definition 3.5 [3, p. 1] A *partition* of a positive integer n is a finite nonincreasing sequence of positive integers $\lambda_1, \lambda_2, \dots, \lambda_m$ such that $\sum_{i=1}^m \lambda_i = n$. The λ_i are called **parts** of the partition.

Definition 3.6 [3, p. 54] A *composition* is a partition in which the order of the summands is considered. A composition with m parts will be denoted by $[c_1, c_2, \dots, c_m]$, in which the c_i are the parts of the composition; note that $c_i \geq 1$ for $1 \leq i \leq m$.

The following results on compositions can be found in the text by Andrews [3].

Lemma 3.7 [3, p. 54, p. 63] The number of compositions of n with m parts, denoted by $c(m, n)$, and the number of compositions of n with m parts in which each part is greater than or equal to i , denoted by $c_i(m, n)$, are given by :

$$\begin{aligned} c(m, n) &= \binom{n-1}{m-1} \\ c_i(m, n) &= \binom{n-(i-1)m-1}{m-1} \end{aligned}$$

The following result follows from Lemma 3.7.

Lemma 3.8 The number of compositions of n with m parts of which exactly k parts are equal to 1 is given by :

$$\begin{cases} 1 & \text{if } k = m = n \\ \binom{m}{k} \binom{n-m-1}{m-k-1} & \text{otherwise} \end{cases}$$

Proof: For $n = m$, the number of compositions satisfying the lemma is 1 if $k = m$ and 0 otherwise. For $n \neq m$, the number can be found as follows. The k parts that are 1 can be chosen in $\binom{m}{k}$ ways. The other $m - k$ parts must not equal 1, and they must sum to exactly $n - k$. Lemma 3.7 gives the number of such possible combinations. Consequently, the desired sum is :

$$\binom{m}{k} c_2(m-k, n-k) = \binom{m}{k} \binom{(n-k)-(m-k)-1}{(m-k)-1} = \binom{m}{k} \binom{n-m-1}{m-k-1}$$

■

Definition 3.9 Let \mathbf{s} be a composition of n . Denote by $\gamma(\mathbf{s})$ the number of parts of \mathbf{s} that are 1.

Definition 3.10 Let $\Gamma_m(n)$ be the set of all compositions of n with m parts, in which at least one part is a 1. That is, $\Gamma_m(n) = \{\mathbf{s} = [s_1, s_2, \dots, s_m] \mid \sum_{k=0}^m s_k = n, s_k \geq 1, \gamma(\mathbf{s}) \geq 1\}$.

Lemma 3.11 $|\Gamma_m(n)| = \binom{n-1}{m-1} - \binom{n-m-1}{m-1}$

Proof: The number of compositions in $\Gamma_m(n)$ is the difference between the number of compositions of n with m parts and the number of compositions of n with m parts with every part ≥ 2 . By Lemma 3.7 the first quantity is $c(m, n) = \binom{n-1}{m-1}$, and the second is $c_2(m, n) = \binom{n-m-1}{m-1}$. ■

Lemma 3.12 $\sum_{\mathbf{s} \in \Gamma_m(j)} \gamma(\mathbf{s}) = m \binom{j-2}{m-2}$

Proof: The total number of parts that are 1 must be counted in all compositions in the set $\Gamma_m(j)$. Let S_i be the subset of compositions in $\Gamma_m(j)$ in which part i equals 1, $1 \leq i \leq m$. The total number of ones in all compositions in $\Gamma_m(j)$ with part i equal to one is clearly $|S_i|$. Hence, the desired result can be computed, by summing $|S_i|$, for all i , $1 \leq i \leq m$.

Every composition in S_i has part i equal to 1, and the remaining $m-1$ parts must sum to $j-1$. Therefore, the number of such compositions is $c(m-1, j-1)$. Substituting for $c(m-1, j-1)$ from Lemma 3.7, the resulting expression is: $\sum_{\mathbf{s} \in \Gamma_m(j)} \gamma(\mathbf{s}) = \sum_{i=1}^m |S_i| = \sum_{i=1}^m \binom{j-2}{m-2} = m \binom{j-2}{m-2}$. ■

Definition 3.13 Let $\Psi_m(n) = \bigcup_{j=m}^n \Gamma_m(j)$.

Lemma 3.14 $|\Psi_m(n)| = \binom{n}{m} - \binom{n-m}{m}$

Proof: Since $\Gamma_m(i)$ and $\Gamma_m(j)$ are disjoint for $i \neq j$, $|\Psi_m(n)|$ can be found by summing $|\Gamma_m(j)|$, for all j between m and n .

$$|\Psi_m(n)| = \sum_{j=m}^n |\Gamma_m(j)| = \sum_{j=m}^n \binom{j-1}{m-1} - \sum_{j=m}^n \binom{j-m-1}{m-1} \quad (1)$$

The terms in the second sum are 0 for $m \leq j \leq 2m-1$, and, hence:

$$|\Psi_m(n)| = \sum_{j=m}^n \binom{j-1}{m-1} - \sum_{j=2m}^n \binom{j-m-1}{m-1} \quad (2)$$

Substituting $k = j-1$ in the first sum and $i = j-m-1$ in the second sum we get:

$$|\Psi_m(n)| = \sum_{k=m-1}^{n-1} \binom{k}{m-1} - \sum_{i=m-1}^{n-m-1} \binom{i}{m-1} \quad (3)$$

Lemma 3.3 can be used to evaluate both the sums of equation 3, thereby proving the lemma. ■

Lemma 3.15 $\sum_{\mathbf{s} \in \Psi_m(n)} \gamma(\mathbf{s}) = m \binom{n-1}{m-1}$

Definition 3.16 A Markov chain is *irreducible* if there is a path of transitions from any state to any other state. A Markov chain is *recurrent nonnull* if for every state, the mean time to return to that state is finite. A Markov chain is *aperiodic* if for each state there exists a number k such that for all $j \geq k$, returning to the state can be accomplished in exactly j transitions.

Theorem 3.17 [13, p. 128] (*Ergodic Theorem*) A discrete-time Markov chain is said to be *ergodic* if it is irreducible, recurrent nonnull, and aperiodic. If a Markov chain is ergodic, there exists a unique limiting distribution for the probability of being in a state k denoted as $\pi(k)$, independent of the initial state. These probabilities are called the *steady-state* or equilibrium probabilities.

To find the steady-state probabilities we can solve the steady-state equation $\underline{\pi}\mathbf{M} = \underline{\pi}$, in which $\underline{\pi}$ is the vector of steady-state probabilities and \mathbf{M} is the single-step transition probability matrix.

4 Analysis of the Randomized Prefetching Model

In this section we analyze the randomized prefetching model, with the aim of deriving an expression for the average parallelism of an I/O operation in the steady state. Towards this end, the state space is precisely characterized in Lemma 4.1, and the Markov chain is shown to be ergodic in Lemma 4.3. The steady-state probabilities are then derived in Theorem 4.4. Finally, a closed-form expression for the average I/O parallelism is derived in Theorem 4.6.

In the randomized prefetching model, additional blocks are always prefetched into the cache if possible. When there are insufficient free cache blocks to fetch from all D runs, the runs from which to prefetch are chosen at random from the $D - 1$ runs not containing the demand-fetch block.

Lemma 4.1 Every integer vector $[s_1, s_2, \dots, s_D]$ satisfying the conditions of Definition 2.2 represents a cache state that can be reached from the initial state $[1, 1, \dots, 1]$.

Proof: Let \mathbf{s} be a vector satisfying the conditions of the lemma. Without loss of generality assume that $s_1 = 1$. Choose \mathbf{s}' to be another integer vector with the properties that:

1. $s'_1 = 1$
2. $\delta(\mathbf{s}')=0$; *i.e.*, the cache is full
3. $s'_i \geq s_i, 1 \leq i \leq D$

Since \mathbf{s}' majorizes \mathbf{s} along every component (property 3), \mathbf{s} can be reached from \mathbf{s}' by exactly $\delta(\mathbf{s})$ d-transitions. The lemma can be proven by showing that every state \mathbf{s}' satisfying the three properties is reachable from the initial state.

Our first goal is to reach some state with a full cache. We take $\lfloor \frac{C-D}{D-1} \rfloor$ f-transitions (always depleting the first component) to reach a state \mathbf{t} in which $0 \leq \delta(\mathbf{t}) < D - 1$. If $\delta(\mathbf{t}) > 0$, any p-transition that depletes the first component is taken to reach a state with the first component equal to 1 and no free blocks.

Now, suppose that we want to reach a state \mathbf{q}' with $q'_1 = 1$ and $\delta(\mathbf{q}') = 0$ from state \mathbf{q} with $q_1 = 1$ and $\delta(\mathbf{q}) = 0$. Define the *distance* between \mathbf{q} and \mathbf{q}' as follows:

$$dist(\mathbf{q}, \mathbf{q}') = \sum_{i=1}^D |q_i - q'_i|$$

Since $\mathbf{q} \neq \mathbf{q}'$ and $\delta(\mathbf{q}) = \delta(\mathbf{q}') = 0$, there exist parts i and j such that $q_i > q'_i$ and $q_j < q'_j$. Take a d-transition that depletes part i and a p-transition that increments part j to reach state \mathbf{q}'' such that $dist(\mathbf{q}, \mathbf{q}'') = dist(\mathbf{q}, \mathbf{q}') - 2$. Repeat this process by replacing \mathbf{q} with \mathbf{q}'' and choosing possibly different values of i and j ; after $\frac{1}{2}dist(\mathbf{q}, \mathbf{q}')$ repetitions, state \mathbf{q}' will be reached. ■

The previous Lemma can be used to count the number of states in the Markov chain.

Lemma 4.2 The number of states in the Markov chain is $\binom{C}{D} - \binom{C-D}{D}$.

Proof: By Lemma 4.1 and Definitions 3.13 and 3.10, the set of states is in one-to-one correspondence with the set $\Psi_D(C)$. Therefore, the number of states is given by Lemma 3.14. ■

Lemma 4.3 The Markov chain for the randomized prefetching model is ergodic.

Proof: Let \mathbf{s} and \mathbf{t} be two arbitrary states. To make a path from \mathbf{s} to \mathbf{t} , we first go from \mathbf{s} to the initial state $[1, 1, \dots, 1]$ by a sequence of d-transitions. Then we add the path from the initial state to \mathbf{t} that exists by Lemma 4.1. Therefore, the Markov chain is irreducible.

Each state \mathbf{s} has a finite non-zero probability of returning to itself, since there is a finite-length, non-empty path from \mathbf{s} to itself, and all state transitions have finite, non-zero probabilities. Therefore, the Markov chain is recurrent nonnull.

To show that the Markov chain is aperiodic, note that there is a path from any state \mathbf{s} to itself that passes through a state with no free blocks. In a state with no free blocks, any number of p-transitions can be taken without changing state. As a result there is a constant k such that for any $j > k$, there is a path from \mathbf{s} . ■

Since the Markov chain is ergodic, Theorem 3.17 asserts that each state has a unique steady-state probability. Surprisingly, every state has the same probability.

Theorem 4.4 Every state has steady-state probability $1 / \left[\binom{C}{D} - \binom{C-D}{D} \right]$.

Proof: Consider the steady-state equation $\underline{\pi}\mathbf{M} = \underline{\pi}$ mentioned above. To show that the equation has a solution in which all components of $\underline{\pi}$ are equal, it suffices to show that every column in \mathbf{M} sums to 1. In other words, for every state \mathbf{s} the sum of the probabilities on transitions coming into \mathbf{s} is 1. The proof contains three cases depending on the structure of the state \mathbf{s} .

Case 1: $\delta(\mathbf{s}) > 0$ and $2 \leq \gamma(\mathbf{s}) \leq D$.

Since a p-transition always fills the cache, and since \mathbf{s} has a nonzero number of free cache blocks, \mathbf{s} cannot be reached by a p-transition. Also, \mathbf{s} cannot be reached by an f-transition, because an f-transition always leads to a state \mathbf{t} with $\gamma(\mathbf{t}) = 1$. Thus, \mathbf{s} can be reached by only d-transitions.

Since $\gamma(\mathbf{s}) > 1$, for each component i , $1 \leq i \leq D$, there is a d-transition from $[1, s_2, \dots, s_{i-1}, s_i + 1, s_{i+1}, \dots, s_D]$ to \mathbf{s} . Each such transition has probability $1/D$; therefore, the total probability on transitions into \mathbf{s} is $D(1/D) = 1$.

Case 2: $\delta(\mathbf{s}) > 0$ and $\gamma(\mathbf{s}) = 1$.

Without loss of generality assume that $s_1 = 1$. As in case 1, \mathbf{s} cannot be reached by a p-transition. However, \mathbf{s} can be reached by an f-transition from $[1, s_2 - 1, s_3 - 1, \dots, s_D - 1]$; this f-transition has probability $1/D$. Also, for each i , $2 \leq i \leq D$, there is a d-transition with probability $1/D$ from $[s_1, s_2, \dots, s_{i-1}, s_i + 1, s_{i+1}, \dots, s_D]$ to \mathbf{s} . Note that \mathbf{s} cannot be reached by a d-transition by depleting s_1 , since the previous state would have no parts equal to 1. Thus, \mathbf{s} can be reached by $D - 1$ d-transitions and 1 f-transition, each with probability $1/D$ for a total incoming probability of $1/D + (D - 1)(1/D) = 1$.

Case 3: $\delta(\mathbf{s}) = 0$ and $\gamma(\mathbf{s}) = D - k$.

Since \mathbf{s} has no free blocks, \mathbf{s} can only be reached by an f-transition or a p-transition. Therefore, if there is a transition from \mathbf{s}' to \mathbf{s} , \mathbf{s} majorizes \mathbf{s}' componentwise. In particular, if $s_i = 1$, then $s'_i = 1$, for any component i . Furthermore, if $\delta(\mathbf{s}') = j$, then \mathbf{s}' differs from \mathbf{s} in exactly j components, and for each such component, $s'_i = s_i - 1$; we say that such an \mathbf{s}' is *component compatible* to \mathbf{s} . From any \mathbf{s}' that is component compatible to \mathbf{s} , \mathbf{s} can be reached from \mathbf{s}' by either a p-transition, if $\delta(\mathbf{s}') < D - 1$, or an f-transition, if $\delta(\mathbf{s}') = D - 1$. Since \mathbf{s} has k parts not equal to 1, there are exactly $\binom{k}{j}$ states with j free blocks that are component compatible to \mathbf{s} .

Suppose \mathbf{s}' is component compatible to \mathbf{s} and $\delta(\mathbf{s}') = j$. The probability that \mathbf{s} is reached from \mathbf{s}' by a p-transition or an f-transition depends on the occurrence of two events. First, the component of \mathbf{s}' whose depletion caused the transition must be one of the $D - k$ parts of \mathbf{s} which is 1. Secondly, the j blocks that are prefetched must add to those parts i for which $s'_i = s_i - 1$. Since the next depletion is chosen with equal probability from the D parts, the probability of the first event is $(D - k)/D$. The probability of the second event given the choice of the depleted component is

$1/\binom{D-1}{j}$, because the j parts are chosen at random from the $D - 1$ other parts.

The total probability on transitions into \mathbf{s} is the sum of the product of the number of component compatible states with j free blocks and the probability of the transition into \mathbf{s} :

$$\sum_{j=0}^k \binom{k}{j} \left(\frac{D-k}{D}\right) \left(\frac{1}{\binom{D-1}{j}}\right) = \left(\frac{D-k}{D}\right) \sum_{j=0}^k \left[\frac{\binom{k}{j}}{\binom{D-1}{j}}\right]$$

By Lemma 3.4, this quantity is equal to 1. ■

The aim of this analysis is to determine the relation between the size of the cache and the average number of blocks fetched on each I/O operation. Since an I/O operation is performed on either an f-transition or a p-transition, these transitions are referred to as I/O transitions.

Lemma 4.5 Let $\pi(\mathbf{t})$ denote the steady-state probability of a state \mathbf{t} of the Markov chain, and let $n(\mathbf{t})$ be the number of blocks fetched on an I/O operation from \mathbf{t} . The average number of blocks fetched on an I/O operation in the steady state is:

$$\frac{\sum_{\mathbf{t}} \gamma(\mathbf{t}) \pi(\mathbf{t}) n(\mathbf{t})}{\sum_{\mathbf{s}} \gamma(\mathbf{s}) \pi(\mathbf{s})}$$

where both sums are taken over all states in the Markov chain.

Proof: An I/O transition from a state \mathbf{t} occurs if and only if a component equal to 1 is depleted. Since \mathbf{t} has $\gamma(\mathbf{t})$ parts that are 1 and each of the D parts is depleted with probability $1/D$, the probability of an I/O transition from \mathbf{t} is the product of $\gamma(\mathbf{t})/D$ and the probability of being in state \mathbf{t} (*i.e.* $\pi(\mathbf{t})$). Therefore, the conditional probability that an I/O transition is taken from \mathbf{t} , given that some I/O transition is taken is $\gamma(\mathbf{t})\pi(\mathbf{t})/\sum_{\mathbf{s}} \gamma(\mathbf{s})\pi(\mathbf{s})$. Since the number of blocks fetched on an I/O transition from \mathbf{t} is $n(\mathbf{t})$, the lemma follows. ■

Theorem 4.6 For the randomized prefetching model, the average number of blocks fetched on an I/O operation in the steady state is:

$$\frac{\binom{C}{D} - \binom{C-D}{D}}{\binom{C-1}{D-1}}$$

Proof: By Theorem 4.4, $\pi(\mathbf{t})$ is the same for every state \mathbf{t} in the Markov chain. The number of blocks $n(\mathbf{t})$ fetched on an I/O transition from \mathbf{t} is $\min(D, \delta(\mathbf{t}) + 1)$, since D blocks are fetched if $\delta(\mathbf{t}) \geq D - 1$ and $1 + \delta(\mathbf{t})$ are fetched otherwise.

Thus, the average number of blocks read on any I/O operation is:

$$\frac{\sum_{\mathbf{t}} \gamma(\mathbf{t}) \min(D, \delta(\mathbf{t}) + 1)}{\sum_{\mathbf{s}} \gamma(\mathbf{s})} \tag{4}$$

in which both sums are taken over all states in the Markov chain.

A closed form in terms of binomial coefficients will be derived for (4); first, a closed form will be derived for the denominator. By Lemma 4.1 and definitions 3.13 and 3.9, the denominator is the total number of parts equal to 1 in the set $\Psi_D(C)$. By Lemma 3.15, this quantity is:

$$D \binom{C-1}{D-1} \quad (5)$$

To find a closed form for the numerator, the number of parts equal to 1 must be summed over all states with $C-j$ free blocks. By Lemma 3.12, this quantity is $D \binom{j-2}{D-2}$. The number of blocks fetched is D if $j \leq C-D$ and $(C-j+1)$ otherwise. In the first case, write D as $(C-j+1) - (C-j-D+1)$. The number of blocks fetched is determined by summing over j :

$$\sum_{j=D}^C (C-j+1)D \binom{j-2}{D-2} - \sum_{j=D}^{C-D} (C-j-D+1)D \binom{j-2}{D-2} \quad (6)$$

Before Lemma 3.1 can be used to simplify (6), several transformations must be made. The lower bound of both summation indices can be extended to $j=0$, since the binomial coefficient is 0 when $j < D$. Additionally, the substitution $a = \binom{a}{1}$ will be made, resulting in:

$$D \left[\sum_{j=0}^C \binom{C-j+1}{1} \binom{j-2}{D-2} - \sum_{j=0}^{C-D} \binom{C-j-D+1}{1} \binom{j-2}{D-2} \right]$$

Change the index of summation by making the substitution $i = j - 2$. The upper bound on index i can be extended by 1 without changing the total sum. The result is:

$$D \left[\sum_{i=0}^{C-1} \binom{C-i-1}{1} \binom{i}{D-2} - \sum_{i=0}^{C-D-1} \binom{C-i-D-1}{1} \binom{i}{D-2} \right] \quad (7)$$

Finally, Lemma 3.1 can be applied to simplify each sum in (7), resulting in:

$$D \left[\binom{C}{D} - \binom{C-D}{D} \right] \quad (8)$$

By combining the denominator, (5), and the numerator, (8), the theorem is proved. \blacksquare

5 Analysis of the Deterministic Model

In this section the Markov chain for the deterministic prefetching model is analyzed. Once again, an expression will be derived for the average parallelism of an I/O operation in the steady state. The organization of the section parallels that of Section 4. The state space of the Markov chain is characterized in Theorem 5.4, and shown to be ergodic in Lemma 5.7. The steady-state probabilities

of the states in the Markov chain are then derived in Theorem 5.8. Finally, a closed-form expression for the average I/O parallelism is derived in Theorem 5.11.

Recall that in the deterministic prefetching model a block is fetched from all D runs if there is sufficient cache space. However, when there are insufficient free cache blocks to fetch from all D runs, *only one* block—the demand-fetch block—is fetched.

The state spaces of the Markov chains for this model and the randomized model differ. In particular, Theorem 5.4 will show that not every state satisfying Definition 2.2 is reachable from the initial state.

Lemma 5.1 Any state \mathbf{s} with $\gamma(\mathbf{s}) \geq \delta(\mathbf{s}) + 2$ is not reachable from any other state.

Proof: Since $\gamma(\mathbf{s}) \geq 2$, \mathbf{s} cannot be reached by an f-transition, since an f-transition into \mathbf{s} requires $\gamma(\mathbf{s}) = 1$. Hence, \mathbf{s} must be reached by a d-transition from some state, say \mathbf{t} . Then, $\delta(\mathbf{t}) = \delta(\mathbf{s}) - 1$ and either $\gamma(\mathbf{t}) = \gamma(\mathbf{s}) - 1$ or $\gamma(\mathbf{t}) = \gamma(\mathbf{s})$. In either case, $\gamma(\mathbf{t}) \geq \delta(\mathbf{t}) + 2$.

The lemma is proved by induction on $\delta(\mathbf{s})$. The lemma holds for a state \mathbf{s} with $\delta(\mathbf{s}) = 0$ and $\gamma(\mathbf{s}) \geq 2$, since there are no d-transitions into a state with zero free cache blocks.

Now assume inductively that the lemma holds for all states \mathbf{w} where $\delta(\mathbf{w}) < \delta(\mathbf{s})$. As shown above, any state \mathbf{t} with a transition to \mathbf{s} satisfies $\gamma(\mathbf{t}) \geq \delta(\mathbf{t}) + 2$, and $\delta(\mathbf{t}) < \delta(\mathbf{s})$. Hence, \mathbf{t} satisfies the induction hypothesis, and so \mathbf{t} is unreachable. Hence, \mathbf{s} is unreachable. Therefore, the lemma holds for all \mathbf{w} , $\delta(\mathbf{w}) \leq \delta(\mathbf{s})$. ■

Figure 2.4 shows the Markov chain for the deterministic model. State $[1, 1, 5]$ is marked as invalid i.e. it is not reachable from the initial state. This conforms to Lemma 5.1, since this state has 0 free blocks and two components which are 1.

Lemma 5.2 Let $\mathbf{s} = [1, s_2, s_3, \dots, s_{j-1}, 1, 1, \dots, 1]$ be a state such that $\delta(\mathbf{s}) \geq (D - 1) + n_j$, $n_j \geq 1$, and let \mathbf{t} be the state $[1, s_2, s_3, \dots, s_{j-1}, 1 + n_j, 1, \dots, 1]$. Then, there is a sequence of transitions that takes the system from \mathbf{s} to \mathbf{t} .

Proof: We first describe a sequence of transitions Δ_j that lead from \mathbf{s} to a state in which component s_j is incremented by 1. First, s_1 is depleted, resulting in an f-transition since $\delta(\mathbf{s}) \geq (D - 1) + n_j$. From this state, every component except 1 and j is successively depleted once each by a d-transition, leading to a state $\mathbf{s}' = [1, s_2, s_3, \dots, s_{j-1}, 2, 1, \dots, 1]$; notice that $\delta(\mathbf{s}') \geq (D - 1) + (n_j - 1)$, and hence, the sequence Δ_j can be applied once more to \mathbf{s}' . After a total of n_j such applications the state \mathbf{t} is reached. ■

Lemma 5.3 Every integer vector $\mathbf{s} = [s_1, s_2, \dots, s_D]$ that satisfies Definition 2.2 and satisfies $\gamma(\mathbf{s}) \leq \delta(\mathbf{s}) + 1$ is reachable from state $\mathbf{c} = [1, 1, \dots, 1]$.

Proof: Without loss of generality we rearrange the components of \mathbf{s} in non-decreasing order. That is, $\mathbf{s} = [1, \dots, 1, s_{k+1}, s_{k+2}, \dots, s_D]$ such that $s_i \leq s_{i+1}$ for $1 \leq i \leq D$. As a result of this reordering, $s_i = 1$ for $1 \leq i \leq k$, $k = \gamma(\mathbf{s})$, and $s_i \geq 2$ for $k + 1 \leq i \leq D$.

Since $k = \gamma(\mathbf{s}) \leq \delta(\mathbf{s}) + 1$ we have $\delta(\mathbf{s}) \geq k - 1$. We also have the following bound on $\delta(\mathbf{c})$.

$$\begin{aligned}
\delta(\mathbf{c}) &= \delta(\mathbf{s}) + \sum_{i=k+1}^D (s_i - 1) \\
&= \delta(\mathbf{s}) + \sum_{i=k+1}^D (s_i - 2) + (D - k) \\
&\geq k - 1 + \sum_{i=k+1}^D (s_i - 2) + (D - k) \\
&= (D - 1) + \sum_{i=k+1}^D (s_i - 2) \tag{9}
\end{aligned}$$

Inequality (9) ensures that we can apply the sequence of transitions of Lemma 5.2 to each of the components c_{k+1} through c_D of \mathbf{c} in turn, to get from \mathbf{c} to a state $\mathbf{s}' = [1, \dots, 1, s_{k+1} - 1, s_{k+2} - 1, \dots, s_D - 1]$. Also, $\delta(\mathbf{s}') \geq D - 1$. To reach \mathbf{s} from \mathbf{s}' , an f-transition is made by depleting s'_1 , and then components 2 through k are depleted. Since \mathbf{s} was an arbitrary state satisfying the conditions of the lemma, the proof is complete. ■

Combining, Lemma 5.1 and Lemma 5.3, we have the following theorem.

Theorem 5.4 In the Markov chain for the deterministic model, the set of all states reachable from the initial state are precisely those described in Lemma 5.3.

To derive the steady-state probabilities and the average I/O parallelism, it is useful to classify the states of the Markov chain as follows.

Definition 5.5 A state \mathbf{s} in the Markov chain is an *interior* state if the number of free cache blocks in \mathbf{s} is at least $D - 1$, *i.e.* $0 \leq \delta(\mathbf{s}) \leq D - 1$. A state that is not an interior state is an *edge* state.

Definition 5.6 Let I denote the set of all interior states and E denote the set of all edge states in the Markov chain. Let $E_{f,k}, 0 \leq f \leq D - 2, 1 \leq k \leq f + 1$ denote the subset of edge states that have f free cache blocks and have k parts which are 1.

Before we derive the steady-state probabilities of the states in the Markov chain, we establish that they exist and are unique. This can be done by showing that the Markov chain is ergodic.

Lemma 5.7 Each valid state \mathbf{s} has a unique steady-state probability.

Proof: Theorem 5.4 defines the set of states in the Markov chain for the deterministic model. The proof of ergodicity closely parallels the proof of Lemma 4.3.

Let \mathbf{s} and \mathbf{t} be two arbitrary states. To find a path from \mathbf{s} to \mathbf{t} , use d-transitions to get from \mathbf{s} to the initial state $[1, 1, \dots, 1]$, and then use a path from the initial state to \mathbf{t} , which exists by Lemma 5.3. Therefore, the Markov chain is irreducible.

Each state \mathbf{s} has a finite non-zero probability of returning to itself, since there is a finite-length, non-empty path from \mathbf{s} to itself, and all state transitions have finite, non-zero probabilities. Therefore, the Markov chain is recurrent nonnull.

To show that the Markov chain is aperiodic, note that there is a path from a state \mathbf{s} to itself that passes through an edge state. In an edge state, any number of d-transitions can be made without changing state. As a result, the Markov chain is aperiodic.

Hence, the Markov chain is ergodic. By Theorem 3.17, every state has a unique steady-state probability. ■

We derive below the relative steady-state probabilities of the states, rather than the absolute probabilities. Each relative probability must be scaled by a normalization constant, α , to obtain the actual steady-state probability of that state. α can be chosen uniquely so that the sum of the probabilities of all states is 1, but we do not need to compute it explicitly.

Theorem 5.8 The steady-state probability of state \mathbf{s} , (denoted by $\pi(\mathbf{s})$), is given by:

$$\begin{cases} \alpha(D-1) & \text{if } \mathbf{s} \in I \\ \frac{\alpha(f+1)!(D-k-1)!}{(f-k+1)!(D-2)!} & \text{if } \mathbf{s} \in E_{f,k} \end{cases}$$

in which α is the normalization constant.

Proof: The proof will consist of verifying that the probabilities stated in the theorem satisfy the equation $\underline{\pi}\mathbf{M} = \underline{\pi}$, in which $\underline{\pi}$ is the vector of steady-state probabilities in the theorem and \mathbf{M} is the single-step transition probability matrix given by the transition rules defined earlier. In particular, the probabilities of all incoming transitions to \mathbf{s} weighted by the steady-state probability of the originating state will be summed and will be shown to equal $\pi(\mathbf{s})$.

The proof consists of four cases depending on the structure of state \mathbf{s} . Figure 5.5 shows the transitions for the different types of states. Each transition shown has $1/D$ probability of being taken, except those transitions labeled as having n occurrences which have n/D probability.

Case 1: $\mathbf{s} \in I$. We show that $\pi(\mathbf{s}) = \alpha(D - 1)$.

Figure 5.5a shows the incoming transitions to an interior state state \mathbf{s} with $\gamma(\mathbf{s}) = 1$. State \mathbf{s} can be reached by $D - 1$ d-transitions or an f-transition. The f-transition must be from an interior state, say \mathbf{t} , and $\pi(\mathbf{t}) = \alpha(D - 1)$. The d-transition must be from a state \mathbf{u} , where either $\mathbf{u} \in I$ or \mathbf{u} is an edge state with $\delta(\mathbf{u}) = D - 2$. In both cases, $\pi(\mathbf{u}) = \alpha(D - 1)$. Summing the weighted probabilities on the input edges we get: $\alpha(1/D)(D)(D - 1) = \alpha(D - 1)$.

Figure 5.5b shows an interior state \mathbf{s} with $2 \leq \gamma(\mathbf{s}) \leq D$. Such a state cannot be reached by any f-transitions. All incoming transitions are d-transitions, each having probability $1/D$. Any of these d-transitions originates from a state (say \mathbf{u}) where either $\mathbf{u} \in I$ or \mathbf{u} is an edge state with $\delta(\mathbf{u}) = D - 2$. In either case, $\pi(\mathbf{u}) = \alpha(D - 1)$, and hence, $\pi(\mathbf{s}) = \alpha(D - 1)$.

Case 2: $\mathbf{s} \in E_{f,1}$. We show that $\pi(\mathbf{s}) = \alpha(f + 1)$.

Figure 5.5c shows the incoming transitions to a state $\mathbf{s} \in E_{0,1}$. The sum of the incoming probabilities is $\alpha(f + 1)(1/D) + \alpha(D - 1)(1/D) = \alpha$, where the first term is due to the r-transition, and the second is due to the f-transition into \mathbf{s} . This satisfies the Theorem since $f = 0$.

For $f \geq 1$, Figure 5.5d shows all incoming transitions. Any d-transition into \mathbf{s} must be from a state $\mathbf{u} \in E_{f-1,1}$, with $\pi(\mathbf{u}) = \alpha f$. There is one f-transition into \mathbf{s} from an interior state, and one r-transition. Summing the weighted probabilities of the incoming transitions, we get: $\alpha f \frac{1}{D}(D - 1) + \alpha \frac{1}{D}(D - 1) + \alpha \frac{1}{D}(f + 1) = \alpha \frac{1}{D}(Df + D) = \alpha(f + 1)$.

Case 3: $\mathbf{s} \in E_{f,k}$, $f \geq 1$ and $2 \leq k \leq f$. We show that

$$\pi(\mathbf{s}) = \alpha \frac{(f + 1)!(D - k - 1)!}{(f - k + 1)!(D - 2)!}$$

Figure 5.5e shows the case of an edge state with $2 \leq k \leq f$. The state \mathbf{s} , with $f \geq 2$ and $2 \leq k \leq f$, has a total of D incoming d-transitions; k from states in $E_{f-1,k-1}$, and the remaining $D - k$ from states in $E_{f-1,k}$. Since $\gamma(\mathbf{s}) \geq 2$, \mathbf{s} cannot be reached by an f-transition. Since \mathbf{s} has k parts equal to 1, there are k incoming r-transitions.

Denoting by $P_{f,k}$ the steady-state probability of a state in $E_{f,k}$, the weighted sum of the probabilities of transitions into \mathbf{s} is $[P_{f-1,k-1}](k/D) + [P_{f-1,k}]((D - k)/D) + [P_{f,k}](k/D)$. After substituting for each $P_{f,k}$ and simplifying, the sum reduces to $\frac{(f+1)!(D-k-1)!}{(f-k+1)!(D-2)!}$, as claimed.

Case 4: $\mathbf{s} \in E_{f,f+1}$. We show that

$$\pi(\mathbf{s}) = \frac{(f+1)!(D-f-2)!}{(D-2)!}$$

Figure 5.5f shows an edge state \mathbf{s} with $k = f + 1$. The incoming transitions are very similar to case 3 with $k = f + 1$, Figure 5.5e, except that there are no transitions into \mathbf{s} from states in $E_{f-1,f+1}$, since all states in $E_{f-1,f+1}$ are unreachable by Lemma 5.1.

Thus \mathbf{s} has $f + 1$ incoming d-transitions from states in $E_{f-1,f}$, and $f + 1$ r-transitions.

The weighted sum of all incoming probabilities is therefore, $[P_{f-1,f}] \frac{1}{D}(f+1) + [P_{f,f+1}] \frac{1}{D}(f+1)$. After simplifying and rearranging terms, the sum reduces to $\alpha \frac{(f+1)!(D-f-2)!}{(D-2)!}$. ■

We now determine the average I/O parallelism for the deterministic model, by evaluating the formula in Lemma 4.5. Recall that $\pi(\mathbf{s})$ denoted the probability of state \mathbf{s} . We first compute two sums which are then used in Theorem 5.11 to determine the average I/O parallelism.

Lemma 5.9

$$\sum_{\mathbf{t} \in I} \pi(\mathbf{t})\gamma(\mathbf{t}) = \alpha D(D-1) \binom{C-D}{D-1}$$

in which α is the normalization constant.

Proof: For an interior state \mathbf{t} , $\pi(\mathbf{t}) = \alpha(D-1)$, by Theorem 5.8. By Theorem 5.4 there is a one-to-one correspondence between I and $\Psi_D(C-D+1)$. Hence, by Lemma 3.15 :

$$\sum_{\mathbf{t} \in I} \pi(\mathbf{t})\gamma(\mathbf{t}) = \alpha(D-1) \sum_{\mathbf{t} \in \Psi_D(C-D+1)} \gamma(\mathbf{t}) = \alpha(D-1)D \binom{C-D}{D-1}$$

■

Lemma 5.10 If $C \geq 2D - 1$ so f-transitions make sense,

$$\sum_{\mathbf{t} \in E} \pi(\mathbf{t})\gamma(\mathbf{t}) = \alpha D(D-1) \binom{C-D}{D-1} [(1-D) + (C-D+1)(H(C-D) - H(C-2D+1))]$$

in which α is the normalization constant and $H(n) = \sum_{i=1}^n (1/i)$ is the n^{th} Harmonic number.

Proof: By Theorem 5.8, if $\mathbf{s} \in E_{f,k}$ then $\pi(\mathbf{s}) = \alpha \frac{(f+1)!(D-k-1)!}{(f-k+1)!(D-2)!}$. By definition, $\gamma(\mathbf{s}) = k$. The number of states in $E_{f,k}$ is given by Lemma 3.8, with $n = C - f$, and $m = D$. Also, by Theorem 5.4, the range of values of k for a given f satisfies $1 \leq k \leq f + 1$, and the range of f is $0 \leq f \leq D - 2$.

$$\sum_{\mathbf{t} \in E} \pi(\mathbf{t})\gamma(\mathbf{t}) = \alpha \sum_{f=0}^{D-2} \sum_{k=1}^{f+1} \left[k \left[\frac{(f+1)!(D-k-1)!}{(f-k+1)!(D-2)!} \right] \binom{D}{k} \binom{C-f-D-1}{D-k-1} \right] \quad (10)$$

Note that $C - f - D - 1 > 0$ since $C \geq 2D - 1$. Equation (10) can be simplified by rearranging and combining terms, resulting in:

$$\alpha D \sum_{f=0}^{D-2} \sum_{k=1}^{f+1} \left[\frac{(f+1)(D-1)}{D-k} \binom{f}{k-1} \binom{C-f-D-1}{D-k-1} \right] \quad (11)$$

The $(D-k)$ term in (11) can be combined with $\binom{C-f-D-1}{D-k-1}$ by noting that $\frac{1}{y+1} \binom{x}{y} = \frac{1}{x+1} \binom{x+1}{y+1}$.

$$\alpha D(D-1) \sum_{f=0}^{D-2} \left[\frac{f+1}{C-f-D} \sum_{k=1}^{f+1} \left[\binom{f}{k-1} \binom{C-f-D}{D-k} \right] \right] \quad (12)$$

The inner sum in (12) can be simplified by using Lemma 3.2, with $m = -1, p = f, q = C - f - D, n = D$. The resulting quantity is:

$$\alpha D(D-1) \sum_{f=0}^{D-2} \left[\frac{f+1}{C-f-D} \left[\binom{C-D}{D-1} - \sum_{k=f+2}^D \binom{f}{k-1} \binom{C-f-D}{D-k} \right] \right] \quad (13)$$

By noting that $f+1 \leq k-1 \leq D-1$ and that consequently $k-1 > f$, the second sum in (13) has lower index starting at $f+2$ implying that for all choices of f the first term inside the sum, $\binom{f}{k-1} = 0$. Hence the second sum is 0 and the resulting expression is:

$$\alpha D(D-1) \binom{C-D}{D-1} \sum_{f=0}^{D-2} \frac{f+1}{C-f-D} \quad (14)$$

The goal is to express the terms in (14) in terms of $H(n) = \sum_{i=1}^n (1/i)$ because there are excellent asymptotic approximations to the Harmonic numbers [7].

To simplify (14), the sum will be expanded and rewritten by subtracting 1 and adding 1 to each term; for clarity, let $x = C - D$. The resulting expression is:

$$\begin{aligned} & \sum_{f=0}^{D-2} \frac{f+1}{x-f} \\ &= \sum_{f=0}^{D-2} \left[\frac{f+1}{x-f} + 1 - 1 \right] = \left[\sum_{f=0}^{D-2} \frac{x+1}{x-f} \right] - (D-1) \\ &= (x+1) \left[\frac{1}{x} + \frac{1}{x-1} + \frac{1}{x-2} + \cdots + \frac{1}{x-(D-2)} \right] - (D-1) \\ &= (x+1)[H(x) - H(x-D+1)] - (D-1) \end{aligned} \quad (15)$$

Substituting $x = C - D$ into (15) results in:

$$\sum_{f=0}^{D-2} \frac{f+1}{C-D-f} = (1-D) + (C-D+1)[H(C-D) - H(C-2D+1)] \quad (16)$$

To complete the proof substitute (16) into (14). \blacksquare

Theorem 5.11 For the deterministic prefetching model, the average number of blocks fetched on an I/O operation in the steady state is:

$$1 + \frac{D - 1}{2 - D + (C - D + 1)[H(C - D) - H(C - 2D + 1)]}$$

in which $H(n) = \sum_{i=1}^n (1/i)$ and we assume $C \geq 2D - 1$. If $C < 2D - 1$, no prefetching is done and the average parallelism is 1.

Proof: The formula derived in Lemma 4.5 is directly applied. Note that $n(\mathbf{t}) = D$ if $\mathbf{t} \in I$, and $n(\mathbf{t}) = 1$ if $\mathbf{t} \in E$. The sum over all the states can be divided into a sum over the interior states and a sum over the edge states, resulting in the following expression for the numerator:

$$D \sum_{\mathbf{t} \in I} \gamma(\mathbf{t})\pi(\mathbf{t}) + \sum_{\mathbf{t} \in E} \gamma(\mathbf{t})\pi(\mathbf{t})$$

The expression for the denominator is :

$$\sum_{\mathbf{s} \in I} \gamma(\mathbf{s})\pi(\mathbf{s}) + \sum_{\mathbf{s} \in E} \gamma(\mathbf{s})\pi(\mathbf{s})$$

The individual sums over the interior and edge states are derived in Lemmas 5.9 and 5.10 respectively. By substituting for the individual sums and simplifying, the theorem is proven. ■

6 Simulation and Comparison of the Models

In this section we simplify the formulae for the average parallelism derived previously to obtain the asymptotic performance (for large D) of the two strategies. We also plot a comparison for some small values of D , and compare their performance using simulation.

Random Strategy

Rearranging the expression in Theorem 4.6, the average parallelism is given by:

$$\frac{\binom{C}{D}}{\binom{C-1}{D-1}} \left(1 - \frac{\binom{C-D}{D}}{\binom{C}{D}}\right)$$

Note that

$$\frac{\binom{C}{D}}{\binom{C-1}{D-1}} = \frac{C}{D}$$

- **Case 1:** $C = \alpha D$, for some fixed constant $\alpha > 2$. Then:

$$\frac{\binom{C-D}{D}}{\binom{C}{D}} \leq \left(\frac{\alpha D - D}{\alpha D}\right)^D = \left(1 - \frac{1}{\alpha}\right)^D$$

Hence, the average parallelism is at least:

$$\frac{C}{D}(1 - (1 - \frac{1}{\alpha})^D) \rightarrow \alpha \quad \text{as } D \rightarrow \infty$$

- **Case 2:** $C = \alpha D^2$, for some fixed constant α . Then:

$$\frac{\binom{C-D}{D}}{\binom{C}{D}} \leq \left(\frac{\alpha D^2 - D}{\alpha D^2}\right)^D = \left(1 - \frac{1}{\alpha D}\right)^D$$

Noting that $(1 - \frac{1}{\alpha D})^D \rightarrow e^{-\frac{1}{\alpha}}$ as $D \rightarrow \infty$, the average parallelism in the limit is given by:

$$\alpha(1 - e^{-\frac{1}{\alpha}}) D$$

- **Case 3:** $C = \alpha D^{1+\beta}$, for some fixed constant $0 < \beta < 1$. Then, a similar derivation can be used to show that the average parallelism for large D is approximately αD^β .
- **Case 4:** $C = D^{2+\beta}$, for some fixed constant $\beta > 0$. It is easy to show that the average parallelism is $D - o(D)$.

Deterministic Strategy

Rewriting equation 5.11, the average parallelism is:

$$1 + \frac{D - 1}{2 - D + (C - D + 1)[H(C - D) - H(C - 2D + 1)]}$$

In the following we approximate the term $[H(C - D) - H(C - 2D + 1)]$ using the fact that $H(n) = \ln n + O(\frac{1}{n})$, and $\ln(1 + z) = z - \frac{z^2}{2} + O(z^3)$ [7].

- **Case 1:** $C = \alpha D$, for some fixed constant $\alpha > 3$. Then:

$$[H(C - D) - H(C - 2D + 1)] = \ln\left(1 + \frac{D - 1}{\alpha D - 2D + 1}\right) + O\left(\frac{1}{D}\right)$$

Now

$$\ln\left(1 + \frac{D - 1}{\alpha D - 2D + 1}\right) \approx \frac{D - 1}{\alpha D - 2D + 1} - \frac{D^2 - 2D + 1}{2((\alpha - 2)^2 D^2 + 2(\alpha - 2)D + 1)} \approx \frac{1}{\alpha - 2} - \frac{1}{2(\alpha - 2)^2}$$

Hence,

$$(C - D + 1)[H(C - D) - H(C - 2D + 1)] - D \approx \frac{\alpha - 3}{2\alpha^2 - 8\alpha + 8} \approx \frac{D}{2\alpha}$$

Thus, average parallelism, for large D is approximately: $1 + 2\alpha$. This approximation is very good for $\alpha > 7$ or 8 . For smaller α , one can use more terms in the expansion for $\ln(1 + z)$ to get a more precise bound.

- **Case 2:** $C = \alpha D^2$, for some fixed constant α . Then:

$$\ln\left(1 + \frac{D-1}{\alpha D^2 - 2D + 1}\right) \approx \frac{D-1}{\alpha D^2 - 2D + 1} - \frac{D^2 - 2D + 1}{2(\alpha D^2 - 2D + 1)^2}$$

Using the same method as above to approximate the Harmonic difference and simplifying, we get:

$$(C - D + 1) [H(C - D) - H(C - 2D + 1)] - D \approx D - 1 + \frac{1}{2\alpha}$$

Thus, the average parallelism for large D is approximately: $1 + \frac{D-1}{1+1/(2\alpha)} \approx \frac{D}{1+1/(2\alpha)}$.

- **Case 3:** $C = \alpha D^{1+\beta}$, for some fixed constant $0 < \beta < 1$. Working as before, one can show that the average parallelism for large D is approximately $2\alpha D^\beta$.
- **Case 4:** $C = D^{2+\beta}$, for some fixed constant $\beta > 0$. The average parallelism can be shown to be $D - o(D)$.

Comparing the expressions for the randomized and deterministic case, the asymptotic parallelism for large D is generally twice as large as that for the randomized strategy, for moderate cache sizes and is still noticeably better when $C = O(D^2)$.

A simulator was developed to execute the program model described in Figure 2.1. For a given value of D , the average parallelism was evaluated for various values of C . Each experiment involved 30 trials, and the results of these trials were averaged to produce a plot of the average parallelism.

Figures 6.6 and 6.7 show the simulation results with 12,500 blocks of data for $D = 5$ and 25,000 blocks of data for $D = 10$. The steady-state analysis is a good predictor of the average I/O parallelism even with nominal-sized data. Figure 6.8 shows the average parallelism predicted by Theorem 4.6 as a function of both C and D . It is interesting to note that for small cache sizes the average I/O parallelism drops as the number of disks increases past a limit. As the cache size increases, the parallelism approaches the number of disks. The deterministic model shows a similar trend as well.

7 Conclusions

Choosing a prefetching strategy that maximizes I/O performance is, in general, a difficult task [10]. In this study, we investigated the performance of two prefetching strategies that can be used with external mergesort. To compare the cache requirements of the two strategies, a Markov model was developed for each strategy. Closed-form expressions were derived for the average I/O

parallelism in each model, and the results were confirmed by simulation. For block-random data high disk concurrency can be obtained with a reasonable amount of disk cache using either of the two strategies we studied.

From a practical viewpoint, the two strategies have very similar performance. Intuitively, this is because for a large enough cache, both strategies are usually in a state where it is possible to prefetch from all disks. From a theoretical viewpoint, it is surprising that a greedy strategy which maximizes the concurrency on every access actually performs worse than the deterministic strategy. Intuitively, this happens because the deterministic strategy is better able to stay away from the boundary states where the cache is full by leaving some cache slots empty sometimes. As a result, the deterministic strategy can prefetch from all D disks more often than the greedy strategy, and those extra full prefetches more than make up for the partial prefetches that the greedy strategy does near the boundary.

Acknowledgements: We thank the referees for their insightful comments.

References

- [1] A. Aggarwal and J. S. Vitter. The Input/Output Complexity of Sorting and Related Problems. *Comm. ACM*, 31(9):1116–1127, 1988.
- [2] R. Agrawal, S. Dar, and H. V. Jagadish. Direct Efficient Transitive Closure Algorithms: Design and Performance Evaluation. *ACM Transactions on Database Systems*, 15(3):427–458, 1990.
- [3] G. E. Andrews. *The Theory of Partitions*. Addison-Wesley, Reading, Massachusetts, 1976.
- [4] T. Cormen. Fast Permuting on Disk Arrays. *J. Parallel and Distributed Computing*, 17:41–57, 1993.
- [5] G. A. Gibson. Performance and Reliability in Redundant Arrays of Inexpensive Disks. In *Proc. Int. Conf. on Management and Performance Evaluation of Computer Systems (CMG'89)*, pages 381–391, 1989.
- [6] G. A. Gibson, L. Hellerstein, R. M. Karp, R. H. Katz, and D. A. Patterson. Coding Techniques for Large Disk Arrays. In *Proc. Third Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS III)*, pages 123–132, 1989.
- [7] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley, 1989.
- [8] J.-W. Hong and H. T. Kung. I/O Complexity: The Red-Blue Pebble Game. In *Proc. Thirteenth ACM Symp. on Theory of Computing*, pages 326–333, 1981.
- [9] M. Y. Kim. Synchronized Disk Interleaving. *IEEE Transactions on Computers*, C-35(11):978–988, 1986.
- [10] D. F. Kotz and C. S. Ellis. Prefetching in File Systems for MIMD Multiprocessors. *IEEE Transactions on Parallel and Distributed Computing*, 1(2):218–230, 1990.
- [11] S. C. Kwan and J. L. Baer. The I/O Performance of Multiway Mergesort and Tag Sort. *IEEE Transactions on Computers*, 34(4):383–387, 1985.
- [12] M. Livny, S. Khoshafian, and H. Boral. Multi-Disk Management Algorithms. In *Proc. ACM Sigmetrics Conference on Measurement and Modeling on Computer Systems*, pages 69–77, 1987.

- [13] M. K. Molloy. *Fundamentals of Performance Modeling*. Macmillan Publishing Company, New York, NY, 1989.
- [14] M. H. Nodine and J. S. Vitter. Optimal Deterministic Sorting in Large-Scale Parallel Memories. In *Proc. 1993 ACM Symposium on Parallel Algorithms and Architectures*, 1993.
- [15] V. S. Pai and P. J. Varman. Prefetching with Multiple Disks for External Mergesort : Simulation and Analysis. In *8th International Conference of Database Engineering*, pages 273–282, 1992.
- [16] D. A. Patterson, G. Gibson, and R. H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 109–116, 1988.
- [17] A. L. N. Reddy and P. Banerjee. An Evaluation of Multiple-Disk I/O Systems. *IEEE Transactions on Computers*, 38(12):1680–1690, 1989.
- [18] A. L. N. Reddy and P. Banerjee. Design, Analysis, and Simulation of I/O Architectures for Hypercube Multiprocessors. *IEEE Transactions on Parallel and Distributed Computing*, 1(2):140–151, 1990.
- [19] K. Salem and H. García-Molina. Disk Striping. In *Proc. Second IEEE Int. Conf. on Data Engineering*, pages 336–342, 1986.
- [20] B. Salzberg. Merging Sorted Runs Using Large Main Memory. *Acta Informatica*, 27(3):195–215, 1989.
- [21] J. D. Ullman and M. Yannakakis. The Input/Output Complexity of Transitive Closure. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 44–53, 1990.
- [22] J. S. Vitter and M. H. Nodine. Large-Scale Sorting in Uniform Memory Hierarchies. *J. Parallel and Distributed Computing*, 17:107–114, 1993.
- [23] J. S. Vitter and E. A. M. Shriver. Optimal Disk I/O With Parallel Block Transfer. In *Proc. 22nd ACM Symposium on Theory of Computing*, pages 159–169, 1990.
- [24] L. Q. Zheng and P.-A^o. Larson. Speeding up External Mergesort. Technical Report CS-92-40, University of Waterloo, Department of Computer Science, August 1992.

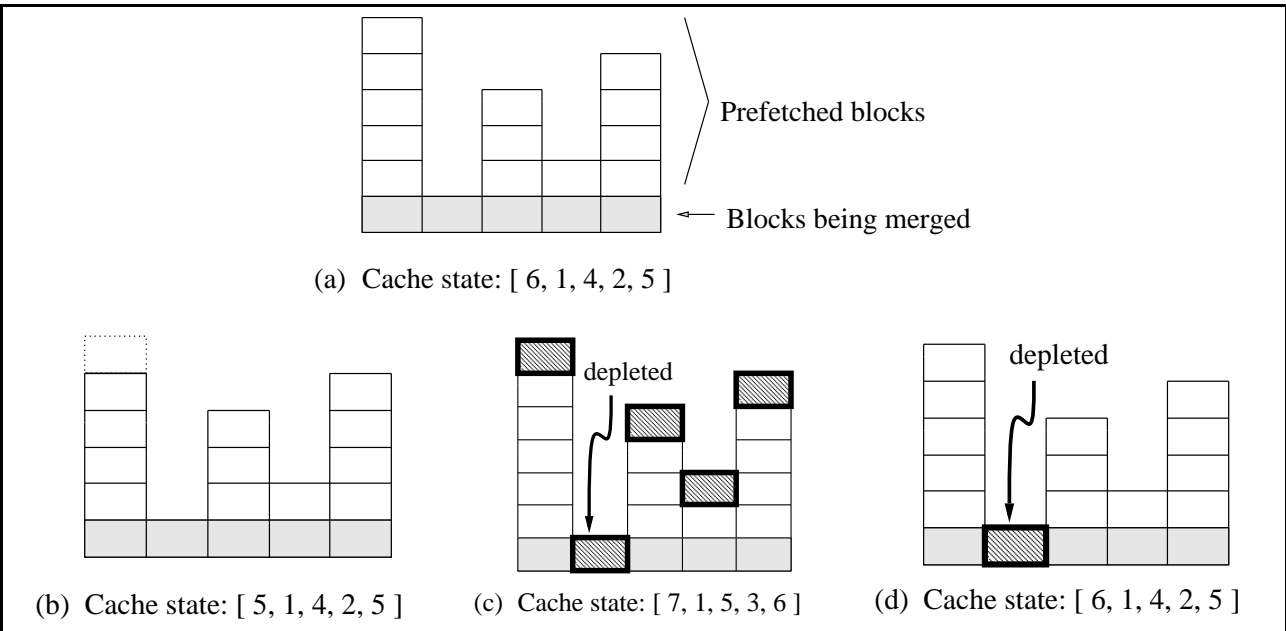


Figure 2.2: (a) A sample cache state for a system with $D = 5$ disks. (b) Cache state after a d-transition. (c) Cache state after an f-transition. (d) Cache state after an r-transition.

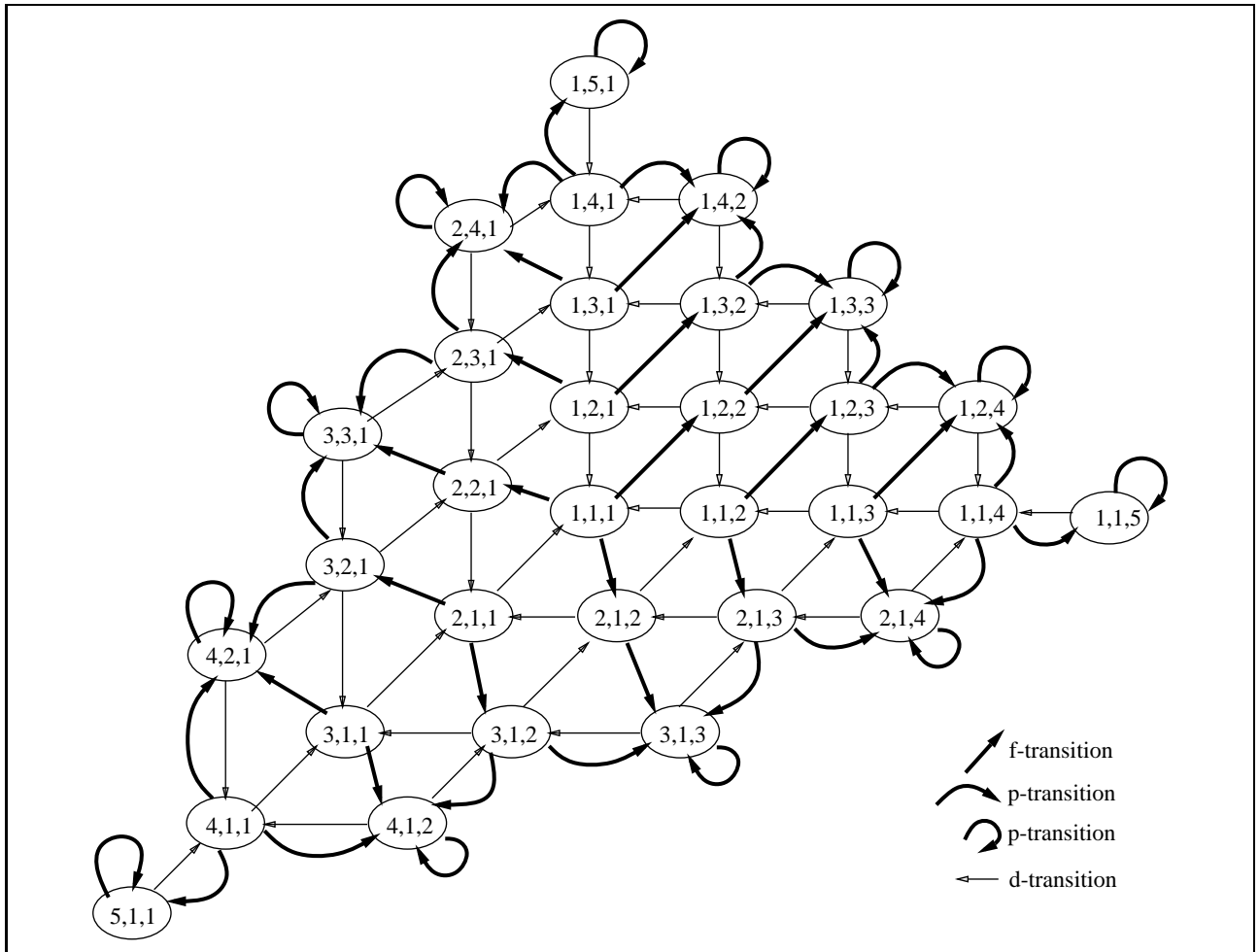


Figure 2.3: The Markov chain for the randomized prefetching model with $D = 3$ disks and $C = 7$ cache blocks.

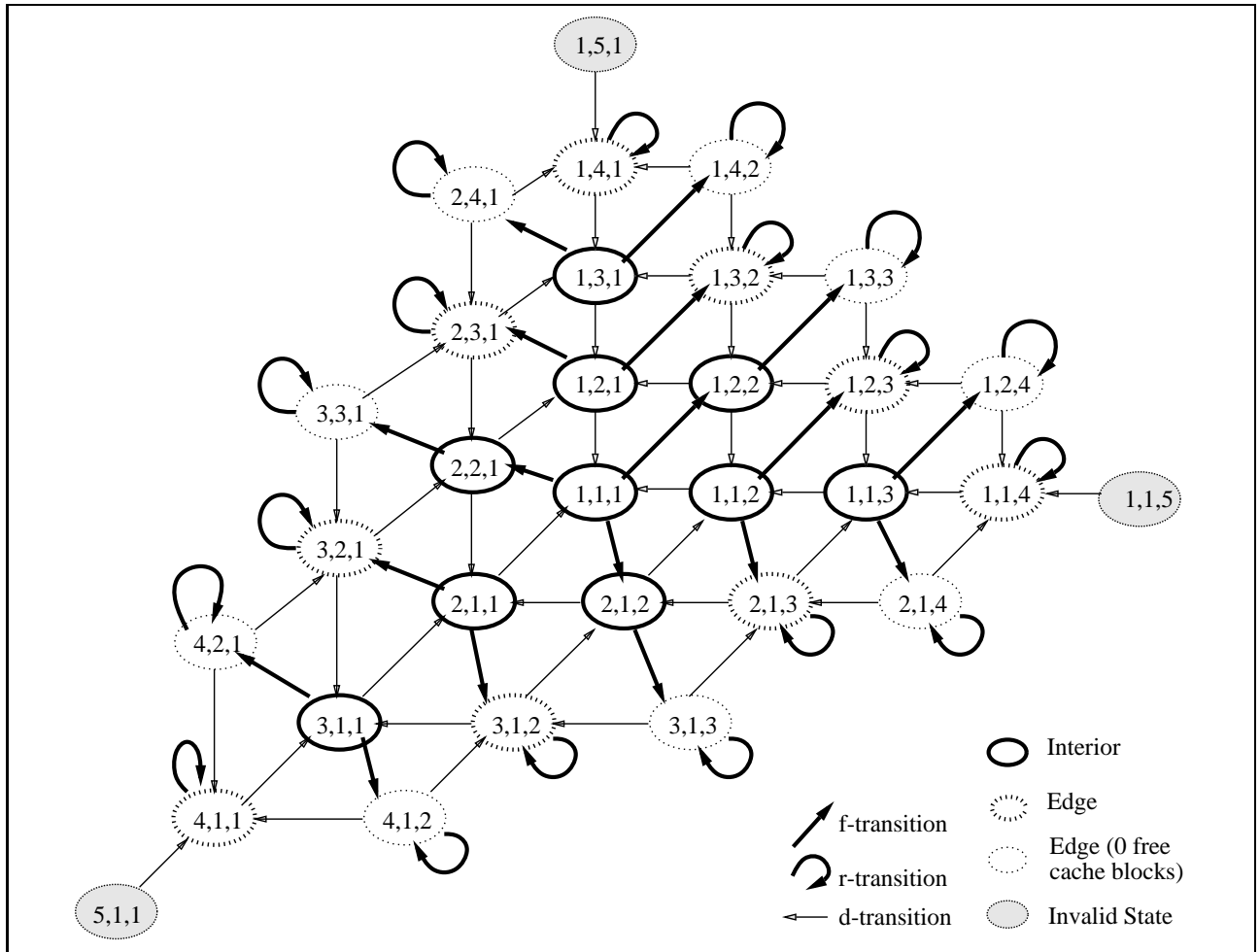


Figure 2.4: The Markov chain for the deterministic prefetching model with $D = 3$ disks and $C = 7$ cache blocks.

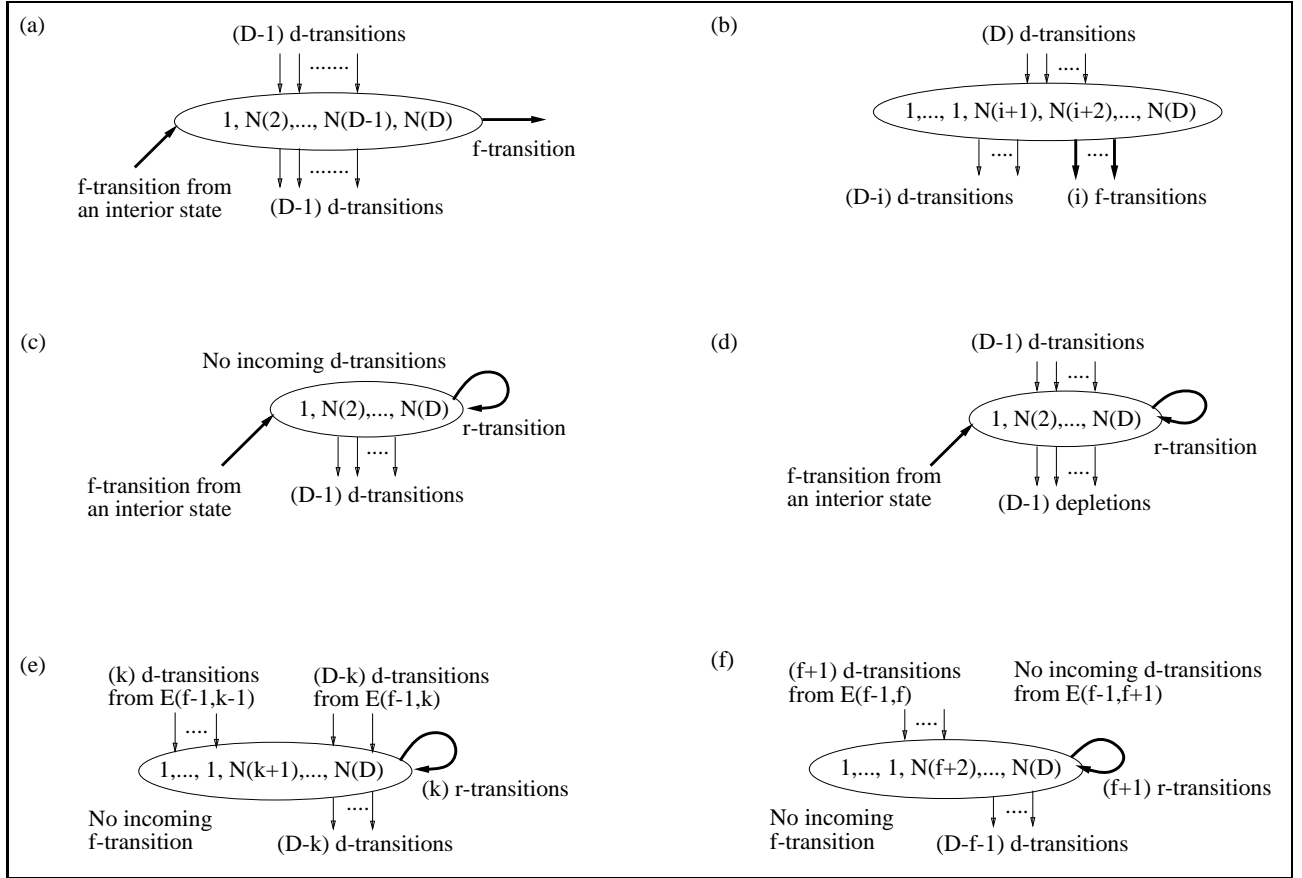


Figure 5.5: **(a)** An interior state with exactly one part equal to 1. **(b)** An interior state with i parts equal to 1. **(c)** An edge state with no free cache blocks. Exactly one part can be equal to 1. **(d)** A state in $E_{f,1}$ with $f \geq 1$ free cache blocks and 1 part which is 1. **(e)** A state in $E_{f,k}$ with $f \geq 1$ free cache blocks and $k \geq 2$ parts which are 1. **(f)** A state in $E_{f,f+1}$ with $f \geq 1$ free cache blocks and $f + 1$ parts which are 1.

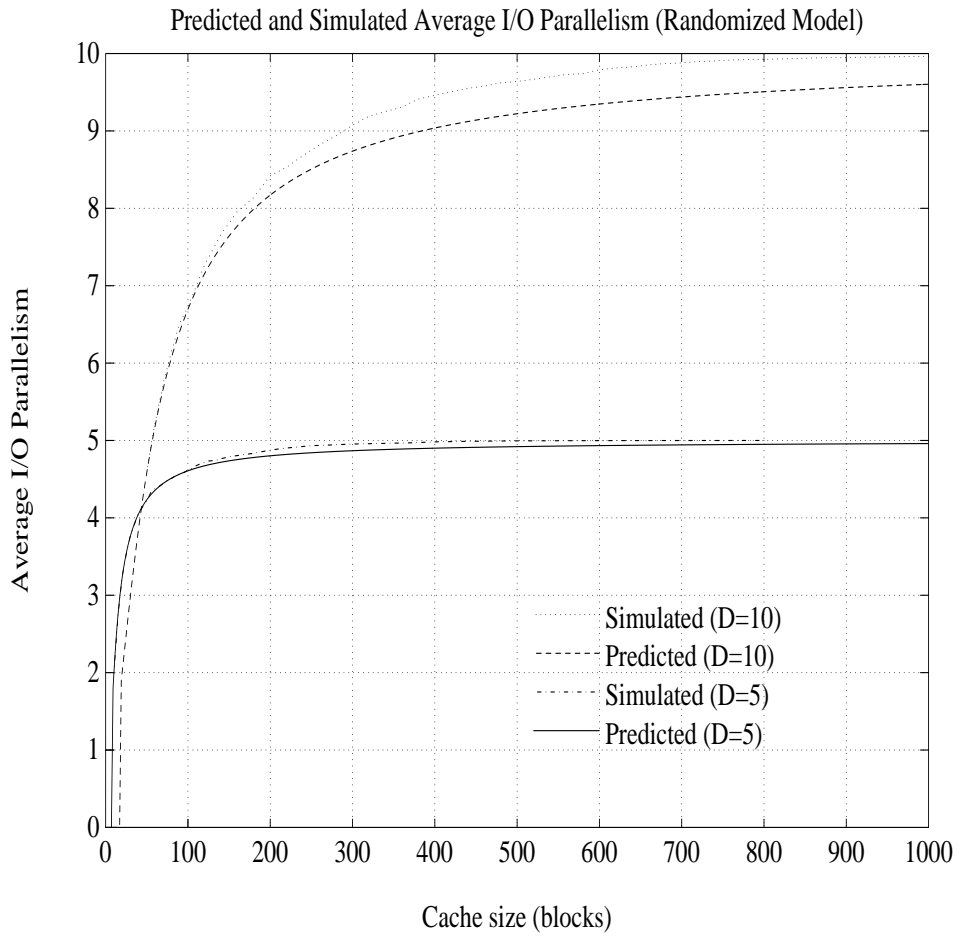


Figure 6.6: *Simulated* and *predicted* average parallelism as a function of the cache size C for the randomized model. For $D = 5$, 12,500 blocks of data are used, and for $D = 10$, 25,000 blocks of data are used.

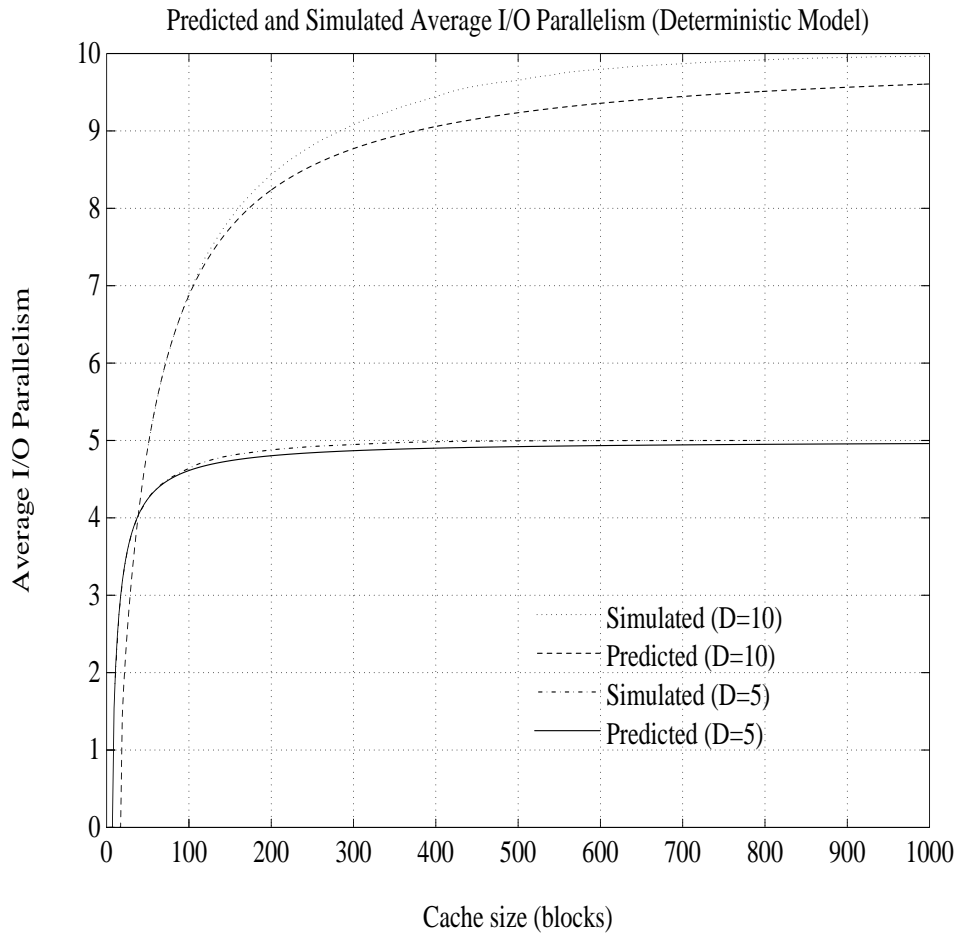


Figure 6.7: *Simulated* and *predicted* average parallelism as a function of the cache size C for the deterministic model. For $D = 5$, 12,500 blocks of data are used, and for $D = 10$, 25,000 blocks of data are used.

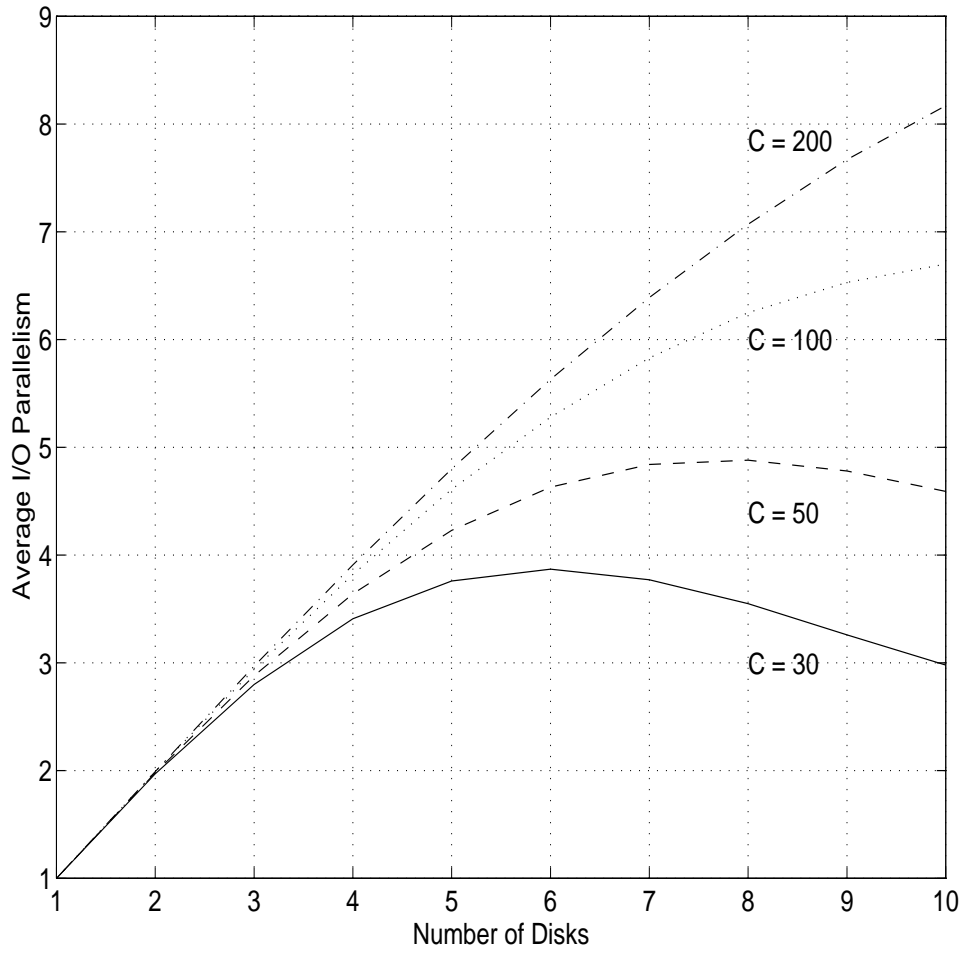


Figure 6.8: Predicted average parallelism as a function of D for the randomized model.