Chapter 24

SORTING AND SELECTION ON PARALLEL DISK MODELS

Sanguthevar Rajasekaran

Department of CISE, University of Florida, Gainesville, FL 32611, USA
raj@cise.ufl.edu

Abstract Data explosion is an increasingly prevalent problem in every field of science. Traditional out-of-core models that assume a single disk have been found inadequate to handle voluminous data. As a result, models that employ multiple disks have been proposed in the literature. For example, the Parallel Disk Systems (PDS) model assumes $D$ disks and a single computer. It is also assumed that a block of data from each of the $D$ disks can be fetched into the main memory in one parallel I/O operation.

In this article, we survey sorting and selection algorithms that have been devised for out-of-core models assuming multiple disks. We also consider practical implementations of parallel disk models.

Keywords: Parallel disks, Sampling, Sorting and selection algorithms.

1. Introduction

Computing applications have advanced to a stage where the data involved is humongous. The volume of data calls for the use of secondary storage devices such as disks. Even a single disk may not be enough to handle I/O operations efficiently. Thus researchers have proposed models with multiple disks.

One of the models (which refines prior models) that has been studied well is the Parallel Disk Systems (PDS) (Vitter and Shriver 1994). In this model there are $D$ disks and a single computer (sequential or parallel). In one parallel I/O operation, a block of data from each of the $D$ disks can be brought into the main memory of the computer. A block consists of $B$ records. We usually require that $M \geq 2DB$, where $M$ is the internal
memory size. At least $DB$ amount of memory is needed to store the data fetched from the disks and the remaining part of the main memory can be used to overlap local computations with I/O operations. Algorithmists have designed algorithms for various fundamental problems on the PDS model. In the analysis of these algorithms, only I/O operations are counted since the local computations are usually very fast.

In this article, we survey sorting and selection algorithms that have been proposed for models with multiple disks. We also investigate the issue of implementing such models in practice. In Section 2 we present details of the PDS model. Sections 3 and 4 are devoted to a survey of sorting and selection algorithms, respectively. In Section 5 we study the problem of implementing parallel disk models in practice. Section 6 concludes the article.

2. Parallel Disk Systems

Here we give more details of the PDS model. A PDS consists of a computer (this could be sequential or parallel) together with $D$ disks. For any given problem, the input will be given in the disks and the output also is expected to be written in the disks. In one I/O operation, a block of $B$ records can be brought into the core memory of the computer from each one of the $D$ disks. In analyzing the time complexity of any algorithm on the PDS model, we consider only the number of parallel I/O operations and neglect the local computation time since the later is usually much smaller.

If $M$ is the internal memory size of the computer, then one usually requires that $M \geq 2DB$. A portion of this memory is used to store operational data whereas the other portion is used for storing prefetched data that enables overlap of local computations with I/O operations. From hereon, $M$ is used to refer to only $DB$.

The sorting problem on the PDS can be defined as follows. There are a total of $N$ records to begin with so that there are $\frac{N}{D}$ records in each disk. The problem is to rearrange the records such that they are in either ascending order or descending order with $\frac{N}{D}$ records ending up in each disk. For the selection problem also, each disk will have $\frac{N}{D}$ input keys to begin with. The output is through any standard I/O device associated with the computer.

3. Sorting Results on the PDS Model

The problem of external sorting has been widely explored owing to its paramount importance. Given a sequence of $n$ keys, the problem of sorting is to rearrange this sequence in either ascending or descending