ASSIGNMENT 1: Simple Sort

Out: 1/31/02; Due: 2/7/02
Programming Parallel and Distributed Systems
Computer Science 178, Spring 2002
Steven P. Reiss

OBJECTIVE

The purpose of this assignment is to ensure that you are familiar with Java and to give you a performance baseline of a single threaded program for later comparison with multi-threaded applications.

THE PROBLEM

In my work, both in trace data generation and in consulting, I keep generating these small (generally between 2G and 100G) files that need to be sorted. The UNIX sort utility was written long ago in the days of small memories, small files, limited file descriptors, and slow processors. It can take upwards of 24 hours (and lots of extra disk space) to have it sort these files. Thus, I am looking for a faster sorting program that will take a single file as input and produce a single sorted output file.

A couple of things to note. First, the size of the files precludes their being kept in memory and using traditional sort algorithms (such as quicksort). The classic way of doing file sorting involves initially creating small sorted files, merging these files to get larger files, merging these files to get still larger ones, and repeating this until a single file results. You should be able to find reasonable file-based sorting algorithms in any good algorithms book.

SPECIFICATIONS

Your program will be given the name of the file to be sorted and the name of the result file on the command line. It may also be given the name of a temporary directory to be used for any intermediate files. If no temporary directory is given, the program should create a subdirectory in the current directory, use that, and remove it when done. The file to be sorted will consist of lines of text (with newline (\n) terminators). Lines are to be compared lexicographically. The result file should be in ascending order.

You can assume that the file system has at least as much free space as the twice the size of the original file. You should also learn the use of the UNIX limit command to set the maximum number of file descriptors, and the program size. When using Java, you should know both the -Xmx option to set memory size and the -d64 option (available with Java 1.4) that runs using a 64 bit data model and hence can use more the 2G of memory.

TESTING

Sample files of various sizes will be provided in */map/aux0fred/cs178/rawdata*. You can create your own directories in */map/aux0fred/cs178/temp* for testing purposes. Your goal is to see how large a file you can sort within ten minutes of CPU time. On UNIX you can use the system sort routine with the -c option to check if you sorted the file correctly. You should also record the times is takes you to sort various size files within this ten minute limit.

MECHANICS

You should hand in you source code and a graph showing how your program scales run time with file size (either total size or number of lines).

If you decide to work on this in teams, I would suggest that each member of the team implement a different sorting algorithm and you compare the results.