GSB/RF
October 15, 1999

Project 2 — TPIE (Transparent Parallel I/O Environment)

The goal of the second project is to perform experiments with the TPIE library (Transparent Parallel I/O Environment) in a few simple programs, and to compare the performance of using TPIE with the performance of the "hand coded" implementations from Project 1. The report of the Project 2 should be handed Friday November 5.

The following short description of TPIE is taken from the homepage of the TPIE project at Duke University:

> *TPIE is designed to bridge the gap between the theory and practice of parallel I/O systems. It is intended to demonstrate all of the following simultaneously:*
>
> - *Abstract away the details of how I/O is performed so that programmers need only deal with a simple high level interface.*
> - *Implement I/O-optimal paradigms for large scale computation that are efficient not only in theory, but also in practice.*
> - *Remain flexible, allowing programmers to specify the functional details of computation taking place within the supported paradigms. This will allow a wide variety of algorithms to be implemented within the system.*
> - *Be portable across a variety hardware platforms.*
> - *Be extensible, so that new features can be easily added later.*
>
> *TPIE is implemented as a set of templated classes and functions in C++. It also includes a small library and a set of test and sample applications.*

The current version of TPIE (version 0.9.01a) together with the TPIE manual (112 pages) can be downloaded from the homepage of the TPIE project (www.cs.duke.edu/~tpie/).

The tasks of the second project are the following:

1. Download and install the current version of TPIE.

2. Implement MergeSort using TPIE (see *e.g.* Section 4.6.1 of the TPIE manual). Compare the performance of the implementation of MergeSort using TPIE with your most efficient MergeSort implementation from Project 1.

3. Implement the classical linear time selection algorithm (see *e.g.* [1, Section 10.3]) using TPIE such that the number of I/Os is $O(N/B)$. Compare the performance of the linear time and I/O implementation for selection with the performance of an implementation for selection based on the MergeSort implementation from 2.

4. Implement the random permutation algorithm of Sanders [2] using TPIE. Make a second random permutation implementation where each element is assigned a random integer in a scan of the input followed by sorting the elements with respect to the assigned integers using the MergeSort implementation from 2. Compare the two implementations.

References

[1] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[2] Peter Sanders. *Random Permutations on Distributed, External and Hierarchical Memory.* Information Processing Letters 56:305–309, 1998.