Assignment #4
35 points
Due Tuesday, Feb 12, 2008 in class

I. Sorting [15 pts]
The bin sort algorithm sorts N keys distributed over P processors in three phases. In the first phase, each processor places its keys in one of P buckets, where all the keys in bucket \( k \) are strictly less than all the keys in bucket \( k+1 \) (We assume no duplicates). In the second phase, the processors collectively call Alltoallv, the “vector” form of all to all, to route the keys to the owning processor. In the third phase, each processor sorts its local keys. All the keys are now sorted across the P processors.

In what follows, assume that \( N \gg P \gg 2 \).

1. Why do we need the “vector” form of Alltoallv, rather than Alltoall? Some arguments to the Alltoallv are set up using additional MPI calls. Explain the purpose of these arguments, and the MPI calls needed to set them up.

2. In the best expected case, keys are taken from a uniform random distribution over the entire keyspace, and distributed uniformly and randomly over the processors. Under these conditions, how many keys will end up in each bucket after phase 1? What are the running times for phases 1, 2, and 3 for this best expected case?

3. Consider the following case. At the end of phase 2, all the processors end up with the same number of keys, however, two of the processors exchange twice as many keys as in the best expected case of part 2. Explain how this case arises, and compare the running time of phase two with that of the ideal expected case in part (2).

4. Consider a different case where, at the end of the third phase, two of the processors end up with twice the number of the keys as compared with the would be expected case in part (2). What is the running time of phase 3?

II. Performance [10 pts]

A. We have a parallel computer with 100 available processors, and are given an application comprising two phases. The first phase of the application is perfectly parallelizable. The second phase runs on only 50 of the 100 processors because there wasn’t time to fully parallelize it. We aren’t told how much time is spent in each phase. Nevertheless, determine the maximum fraction of the time that can be spent in the second phase if our goal is a parallel speedup (weak scaling) of 80 for the whole application, that is, phases 1 and 2 taken together.

B. What is the speedup of a workload running on 8 processors, which incurs a 20% communication overhead when running in parallel, but no other overheads?
III. Scalability [10 pts]

We are given an algorithm with a serial running time of \( cN^3 \), where \( c \) is a constant ("Best serial program"). The parallel running time of the algorithm running on \( p \) processors is \( T_A \), where

\[
T_A = \frac{c N^3}{p} + b N^2 / \sqrt{p},
\]
and \( b \) is a constant

A. Give the expression for the parallel efficiency of the algorithm.

B. What is the parallel overhead \( T_o \)?

C. What is the isoefficiency function for the algorithm?

D. Are there any limitations to weak scaling?

IV. Extra Credit [5 pts]

Let \( A \) and \( B \) be two nodes in a \( d \)-dimensional hypercube. Define the Hamming distance \( H(A,B) \) as the number of non-zero bits in \( A \otimes B \), where \( \otimes \) is exclusive OR, and by an abuse of notation, \( A \) and \( B \) represent the Gray codes for nodes \( A \) and \( B \).

1. Prove the following theorem using an informal proof.
   \( H(A,B) = \) number of hops in a path from \( A \) to \( B \).

2. Now, prove that there are exactly \( d \) non-overlapping paths between \( A \) and \( B \), that is, sharing no common nodes other than \( A \) and \( B \).