DN2264: Lab Project

Michael Hanke

December 10, 2012

In the present lab work, you will solve a problem by implementing it in a parallel environment using MPI and evaluate the performance of your implementation. You can choose among two different problems:

(A) Evaluation of the eigenvalues of a dense matrix;

(B) Comparision of the performance of parallel sorting routines.

As a result, you will hand in a well-written report and all of your source code (plus makefiles etc.). Your report should contain the following sections:

1. Problem description;
2. Description of your algorithm (using pseudo-code);
3. Theoretical performance estimation;
4. Implementational details (those parts which are not obvious);
5. Results of a typical problem run;
6. Experimental speedup investigations (similarly to homework 3).

Deadline for the report will be April 19, 2013.

Problem A: The Power Method

Implement the power method for the computation of an eigenvector and an eigenvalue of a dense matrix! For testing purposes, you can choose a so-called strictly positive matrix A, that means, $a_{ij} > 0$ for all i, j. The Perron-Frobenius theory ensures that the spectral radius $\rho(A)$ is an eigenvalue with (algebraic) multiplicity 1. So the power method will converge towards an eigenvector to that eigenvalue. Such a matrix can easily be generated using a random number generator. If the matrix has uniformly distributed elements, the speed of convergence will be rather fast (why?). Try to use other distributions such that the number of iterations reaches 100–1000.

In order to test your program you can use a stochastic matrix which is characterized by $\sum_{j=1}^{N} a_{ij} = 1$ for all i. For those matrices, $\rho(A) = 1$ and the corresponding eigenvectors contains only ones (so choose $x^{[0]}$ randomly and not as a constant vector).

In a first step you can assume that you have a $P \times P$ process mesh with identical data distributions for both the rows and columns of A. If you intend to obtain better marks, make you program working for any number of processors.

Problem B: Paralle Sorting

Implement Merge Sort and/or Bucket Sort under the same conditions as you did with Odd-Even Sort in homework 3. Compare the efficiency of all these three methods!