Parallel Systems(CSC548)
Programming Assignment 1

Vineet Krishnan,
vineet@ncsu.edu

September 26, 2011

- Problem Statement
  Matrix Multiplication - An 4096 × 4096 matrix (A.dat) will be provided. A power n will be passed on the command-line. The program should calculate A^n and write the output C-n.dat to the specified output folder. All IO operations should be performed in MPI-IO

- Need for parallelism
  Matrix Multiplication is a very computationally intensive operation. A lot of CPU horsepower is needed. For this sample input, a openMP parallelized program in C running on a modern-day laptop takes about 200 secs for completion. In case of real world problems where the sample size is much larger and higher powers are needed to be computed, this speed is not acceptable.

- Multiplication Algorithm
  At the core, this program uses the basic naive O(n^3) multiplication algorithm. However, because of the modular approach in the code, more sophisticated multiplications algorithms like the Strassen Algorithm $\approx O(n^2.807)$ can easily be substituted.

- Subdivision and Aggregation of sub-tasks During the multiplication, each node picks up a set of rows depending on the sample size and its rank. Following formula is used -

  start_row $= 0 + rank \times \frac{matrix1.rows}{numProcessors}$

  end_row = start_row $+ \frac{matrix1.rows}{numProcessors}$

  Because the row subset on which the multiplication is to be performed is decided based on rank, no messages are required to be sent. This saves a lot of communication overhead.
  Also, this approach intuitively scales up according to the number of processors.

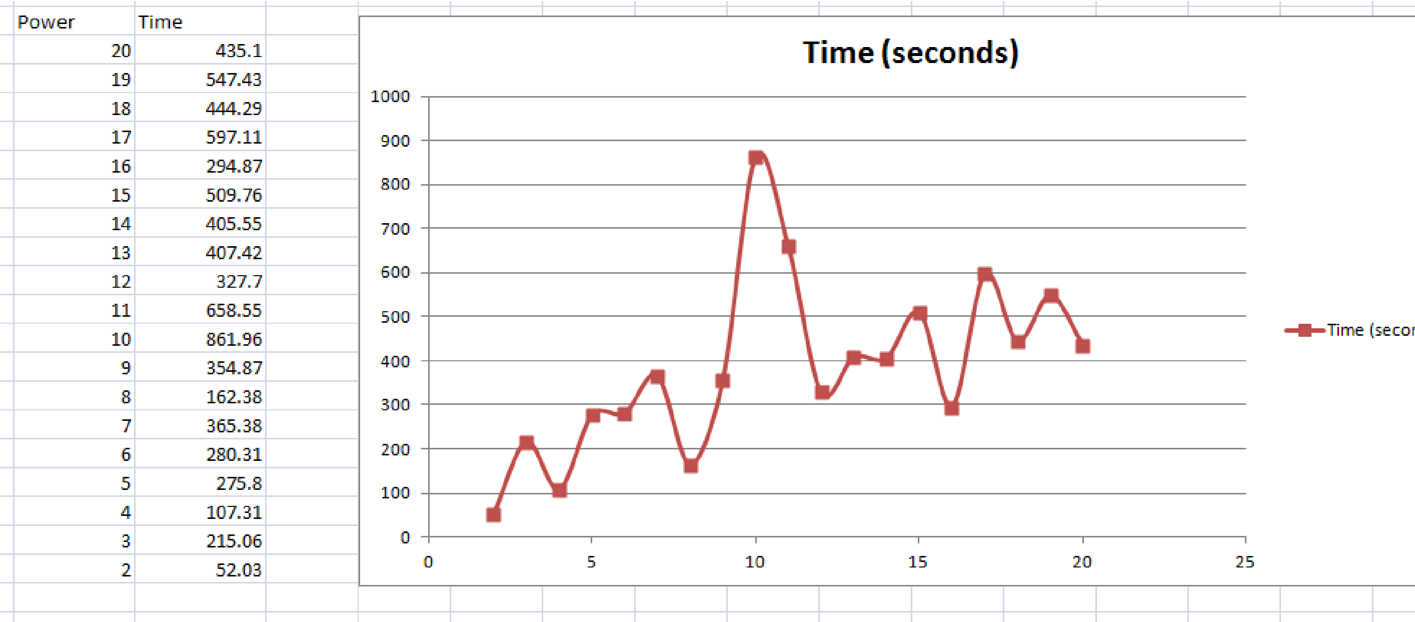
  During aggregation, each node sends accross its computed matrix. The master node then sums all the temporary matrices to form the final matrix.

  Also, to avoid extra multiplication operations, the following algorithm is used.
  Given power = 7
  Binary Representation = $111_2 = 4 + 2 + 1$
  Steps to calculate

  - Save A
  - $A^2 = A \times A$
  - Save A^2
  - $A^4 = A^2 \times A^2$
  - Calculate result as $A^7 = A^4 \times A^2 \times A$

- Scheduling No specific scheduling policies as such are used. Every processor picks up its designated subset of rows and processes it.

- Performance figures All tests were performed interactively with n=4 and openMP turned off.
  All times are in seconds.



Power	Time
20	435.1
19	547.43
18	444.29
17	597.11
16	294.87
15	509.76
14	405.55
13	407.42
12	327.7
11	658.55
10	861.96
9	354.87
8	162.38
7	365.38
6	280.31
5	275.8
4	107.31
3	215.06
2	52.03

As you can see, the time graph changes for the number of total multiplications required for each power.