SE 4F03 – Assignment 2

Ned Nedialkov

1 February 2013

Due date: 13 February

Performance I (10 points). Suppose that you want to study the performance of your implementation of the matrix-vector multiplication. For this problem, you can consider square matrices only, say of size $n \times n$.

You need to produce timing results for various number of processes and problem sizes. When measuring CPU time, measure the time for the parallel matrix-vector product and the time for collecting the result on process 0. Do not include the time for generating and distributing the data. That is, your timing should be

- Start timing
- Multiply A and b in parallel
- Collect the result on process 0
- Stop timing

Plot scalability versus number of processes and also plot the running time versus n for a fixed number of processes, say 8 or 16.

Submit a hard copy of your programs and also submit them to the SVN repository under directory A2. When `make` is typed, an executable `A2perf` should be created. When

```
mpirun -np p ./A2perf n
```

is executed, it should output rows in the format

```
    p     n    time
```

Here `p` is the number of processes, and `n` is the problem size. Output the time accurate up to two digits after the decimal point.

If your programs in A1 do not work properly, or if you wish, you can use the code from the posted solution to A1.

Keep the same file `matvecres.c` from A1 and call the same `getResult` function in this and in the next problem.

Performance II (10 points) Now we want to multiply again an $n \times n$ matrix A times an n vector b. For this problem, assume that they are already distributed. To simulate this, each process will generate an $n/p \times n$ matrix and an n/p vector.

Write a program that does the following:

a. Read n from the standard input; e.g. `mpirun -np 4 10000`.

b. Each process generates a random $n/p \times n$ matrix and an n/p vector. Note: you have to take care of cases when p does not divide n.

c. Start timing

d. Multiply A and b in parallel

e. Collect the result on process 0

f. Stop timing

Submit a hard copy of your programs and also submit them to the SVN repository under directory A2. When make is typed, an executable `A2perf2` should be created. When

```
mpirun -np p ./A2perf2 n
```

is executed, it should output rows in the format

```
p     n     time
```

Produce the same type of plots as in the previous problem.

Discussion (10 points) Discuss the performance of the above two programs. Explain why you obtain or do not obtain good scalability results.

What is the largest n for which your programs in I and II do not crash?