Parallel VSP experiment

Paul E. McGee*, Robert F. Ferguson
pmcgee@ucalgary.ca, rjfergus@ucalgary.ca

Introduction

Software implementation is most importantly done using Matlab's Parallel Computing Toolbox (PCT) and inline methods. Profiling is important to identify bottlenecks in the software design. The basic architecture of the parallel cluster used for computation is a master node with 18 identical slave nodes. More information about the hardware specifics and architecture can be found in the table below, and in [Bonham et al.(2008)]. We have 96 matlab workers running 8 workers on each of 12 slave nodes. We favor Matlab for it's syntax and data-handling capabilities.
The computation performed is a forward modelling experiment using a vertical seismic profile (VSP) configuration. Computation of the VSP experiment is done with different shots being distributed to different workers. The experiment is run for the purpose of VSP imaging.

Results

Implementation of parallelization is primarily done at the software level using Matlab's Parallel Computing Toolbox (PCT). Most importantly, the *parfor* and *matlabpool* commands as well as inlining of methods are utilized. Details of the hardware and cluster architecture is given in the table below for our particular parallel environment. For performance analysis, Gilgamesh is the parallel cluster used with an architecture discussed in [Bonham et al.(2008)].
The following table is a list of some of the contributing factors to increases in elapsed times. The most important factor is the number of processors or number of workers. Other contributing factors are the processor speed and cluster memory, which together affect the speed of each independent computation. The speed of the interconnect network affects mainly the time to write results to the master node, as the data are sent over the network before being written, as is shown in Figure 1.

Table 1: Some computational aspects of Gilgamesh cluster

Factor	Value
Number of processors	96
Processors	2.66 GHz CPU
Cluster and Node Memory	300 GB and 16GB RAM
Serial disk interface	3Gbps
Communications backbone	Gigabit Ethernet
Number of nodes	1 Master and 18 slaves

Results (Cont'd)

Figure 1 shows the code and data distribution for the application. The parfor body is distributed to N-workers located on slave nodes, the rest of the code runs on the master node. Within the parfor loop runs the finite difference code running in parallel on workers. Each instantiation of finite difference code outputs it's data to master node in a temporary directory, into a seperate data file.

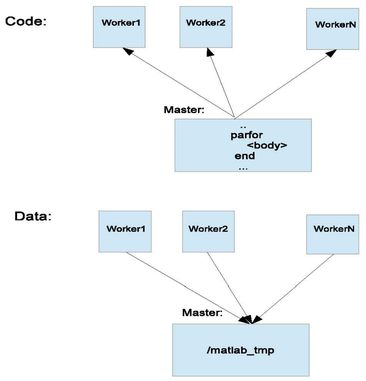

Figure 1 : Code and data distribution

Figure 2 shows the total elapsed time plot for parallel application. The Y-intercept is overhead having to do with setting up the parallel pool of workers. The slope of the line is the incremental cost of adding another shot to the computation. Included in this cost is the cost of distributing the task to a worker, as well as writing the resulting data to the master node's hard-disk.

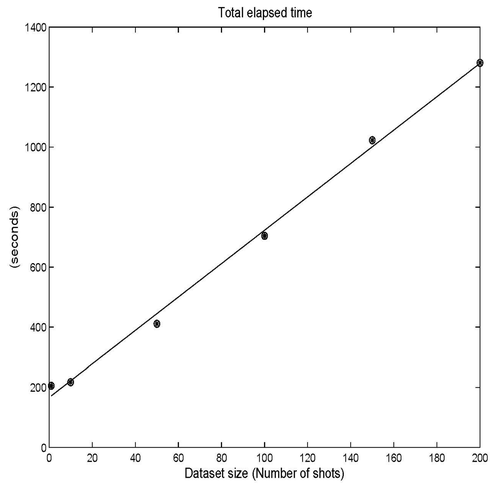

Figure 2 : Computation performance: X-axis is number of shots, Y-axis is the total elapsed time in seconds. The straight line is a linear fit to the performance data, circle and cross symbols are elapsed time measurements.

Results (Cont'd)

The performance equation,

$$T_p = J + K * B \tag{1}$$

is based on a linear fit to the data in Figure 2. T_p is the elapsed time of computation in seconds. J is the overhead cost with the computation. J is a function of a number of factors, one example of which is the cost of setting up the pool of workers . B is the size of the computation, in this case number of VSP shots. K is the incremental cost of adding another shot to the computation. K is a function of several factors, some of which are the overhead associated with distributing a job to a parallel worker on the cluster, as well as the costs associated with writing the data out to the master node. T_p is shown in Figure 2, where J is the Y-intercept and K is the slope of the total elapsed time curve.

Conclusions

Parallelizing the code proves worthwhile. Analyzing the performance of the parallel program allows for identifying how various system factors affect performance. A future direction could be to reduce the elapsed computation time of the plot in Figure 2.

Acknowledgements

The authors wish to thank the sponsors of CREWES for their support.

Bibliography

- Bonham, K., and Ferguson, R. J., 2009, Seismic data modelling using parallel distributed matlab: SEG Expanded Abstracts, **46**, 2692 – 2696.
- Bonham, K., Hall, K., and Ferguson, R. J., 2008, The epic of Gilgamesh: CREWES' new cluster computer: CREWES Research Report, **20**, 1–11.