Parallel VSP experiment

Paul E. McGee and Robert J. Ferguson

ABSTRACT

Shot record modelling of VSP surveys can be computationally time intensive. To address this issue, we parallelize the running of forward modelling code, and we discuss the performance gains that result. The performance of this parallel implementation is measured on a cluster, and we find that performance gains (see Figure 2) are achieved with only small changes to the original code.

To discuss hardware effects on performance, hardware factors are tabulated with the most expensive factors being number of processors, the processors, cluster memory and the communications backbone. We derive an equation relating total computation time to size of the survey, they are related through hardware dependent co-efficients. Software implementation is most importantly done using Matlab's Parallel Computing Toolbox (PCT) and inline methods. The VSP experiment is done with computation of different shots being distributed to Matlab workers.

INTRODUCTION

The parallelization of the serial VSP experiment code incorporates loop parallelism. The costliest computational loop is implemented in a parallel way, and data generation is then distributed across parallel nodes. The results are then written to the master node (see Figure 1). Software implementation is most importantly done using Matlab's Parallel Computing Toolbox (PCT) and inline methods. Some tricks are neccessary to get the code to run efficiently. In particular, profiling is important to identify bottlenecks in the software design. Computation was conducted on a parallel cluster. The basic architecture of the parallel cluster is a master node with 18 identical slave nodes. More information about the hardware specifics and architecture can be found in the table on page 2, 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 computational techniques can be applied to other experiments and configurations, it is beyond the scope of this report to discuss alternative implementations. The experiment is run using a VSP configuration for the purpose of VSP imaging. VSP imaging is conducted using various techniques such as depth migration as in (Dong and Margrave, 2005), VSP reverse-time migration as in (Sun et al., 2011) and VSP interferometry as in (Wang et al., 2010) as well as VSP migration using the Kirchoff integral as in (Dillon, 1988).

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 on page 2 for our particular parallel environment. For performance analysis, Gilgamesh is the parallel cluster used with an architecture discussed in (Bonham et al., 2008). To measure the performance of the parallel and serial program implementations, Matlab's timing methods (*tic* and *toc*) are called.

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. (Bonham and Ferguson, 2009) show that increasing the number of workers decreases the computation time, although their separate computations were not in general independent of each other as they are in our report. In our case, increasing the number of workers should reduce the computation time even more, as there is no overhead associated with inter-node commmunication during the computation itself. 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 respectively
Serial disk interface	3Gbps
Communications backbone	Gigabit Ethernet
Number of nodes	1 Master and 18 slaves

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. Due to the limited size of VSP shots and surveys, the overhead associated with the over the network communication is seen as not as big a problem as potentially could be for a different survey configuration. Contention with writing the shots to the master node's disk, or network overhead could become problems if the cluster is heavily loaded. This analysis was carried out during periods of little computational loading of the cluster.

Figure 2 shows the total elapsed time plot for parallel application versus number of shots where the line is a linear fit to the data. Symbols are the measured data points. 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

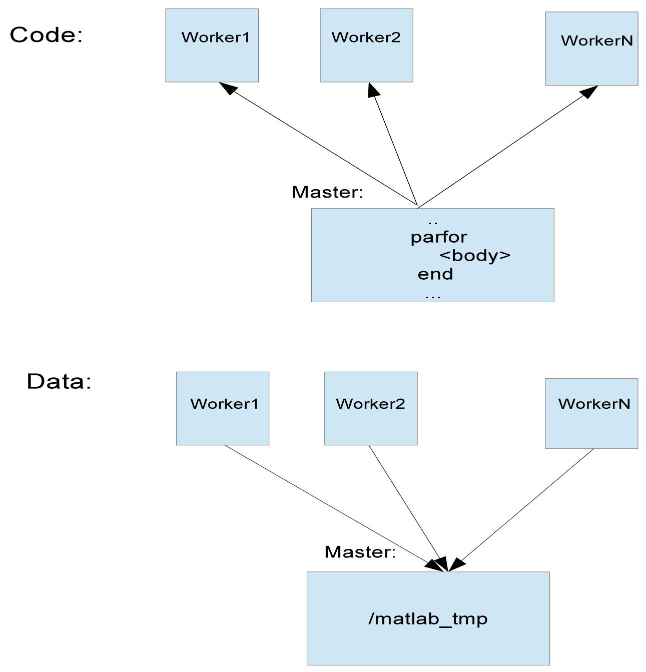

FIG. 1. Code and data distribution

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.

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 also a function of several other 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

We parallelize the running of VSP forward modelling code. We analyze the performance of the parallel program. Parallelizing the code, as well as implementing the changes

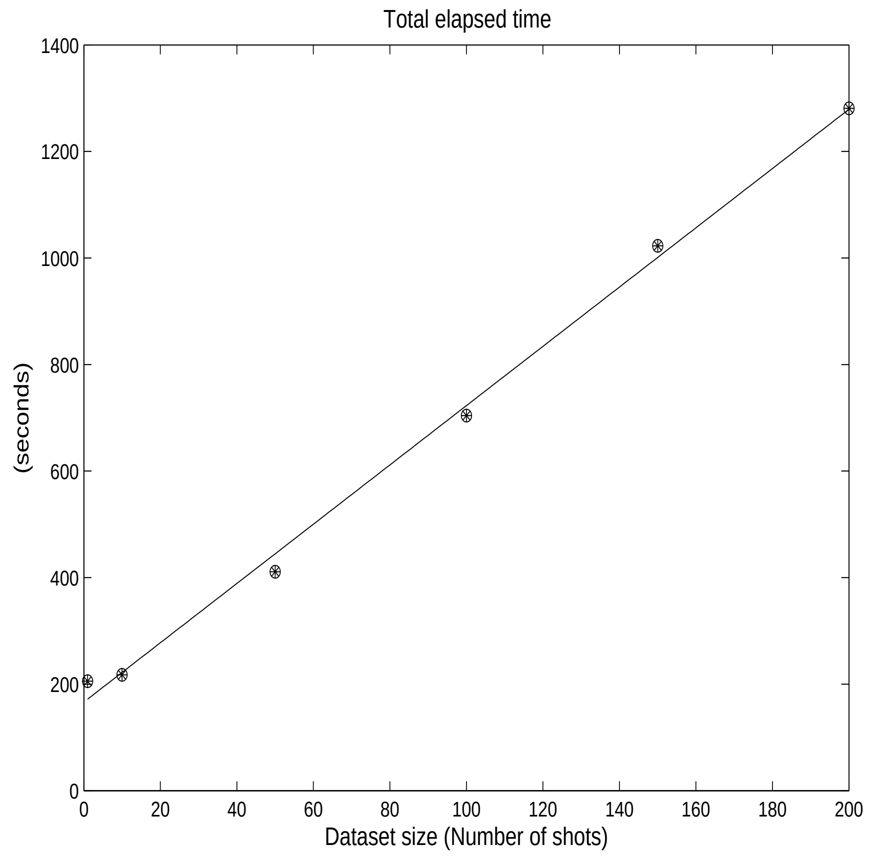

FIG. 2. Computation performance: X-axis is number of shots, Y-axis is the total elapsed time in seconds. The straight line is a linear fit to the performance data, circle and cross symbols are elapsed time measurements.

in Matlab, prove worthwhile. Analyzing the performance of the parallel program is useful as it allowed for identifying how various system factors affect performance. Analyzing the total elapsed time allows us to determine how the software design affected performance. A future direction could be to reduce the elapsed computation time of the plot in Figure 2. Experimenting with changing the scheduling algorithm proved to be not as useful in the short-term, although comparison of computation time using a different scheduling algorithm may prove beneficial as it will potentially reduce the K term in Equation (1). Another future change could be to improve the data handling by having the nodes write out to local disks, then having a collection script collect the data at the end of the computation. This should also reduce the K term in Equation (1) because of reduced networking and master disk write contention, while incurring an additional read and write to disk per shot.

ACKNOWLEDGEMENTS

The authors wish to thank the sponsors of CREWES for their support. In addition, we would like to thank the members of CREWES for their help, in particular Kevin Hall and Rolf Maier.

REFERENCES

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.

Dillon, P., 1988, Vertical seismic profile migration using the Kirchoff integral: Geophysics, **53**, 786–799.

Dong, L., and Margrave, G., 2005, Depth migration of synthetic surface and VSP data: CREWES Research Report, **17**, 1–8.

Sun, W.B., Sun, Z.D., and Zhu, X., 2011, Amplitude preserved VSP reverse time migration for angle-domain CIGs extraction: Applied Geophysics, **8**, 141–149.

Wang, B.L., Zhu, G.Z., and Gao, J., 2010, Joint interferometric imaging of walkaway VSP data: Applied Geophysics, **7**, 41–48.