Perfomance Analysis and Introduction to Optimization of ParFlow

Kevin Sanders, Dr. Timothy Kaiser
Colorado School of Mines
Summer 2012

ParFlow

ParFlow is an open-source, object-oriented, parallel watershed flow model. It includes fully-integrated overland flow, the ability to simulate complex topography, geology and heterogeneity and coupled land-surface processes including the land-energy budget, biogeochemistry and snow (via CLM).

It is multi-platform and runs with a common I/O structure from laptop to supercomputer. ParFlow is the result of a long, multi-institutional development history and is now a collaborative effort between CSM, LLNL, UniBonn and UCB.

- Dr. Reed Maxwell

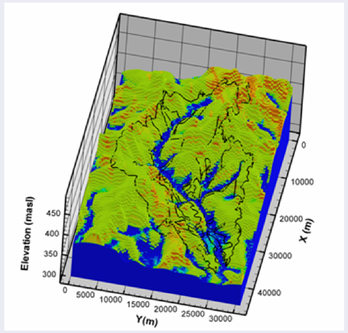


Figure: Little Washita Watershed in Oklahoma (Chow 2006)

Objectives

- Get ParFlow to build and compile with with instrumentation in order to collect data about the runtime and event timeline of the program.
- Use the collected data to analyze scalability, computation vs. communication and the overall efficiency of the program.
- Using this information, pinpoint possible weak points in the code that can be improved and make the necessary changes in an attempt to make the program run faster.

Acknowledgements

Many thanks to Dr. Timothy Kaiser for his assistance and support throughout the project. Also to Dr. Chuck Stone and all the Mines faculty that put this summer experience together. Thanks to the National Science Foundation, as well, for making this possible with the DMR-0820518 award.

Sources

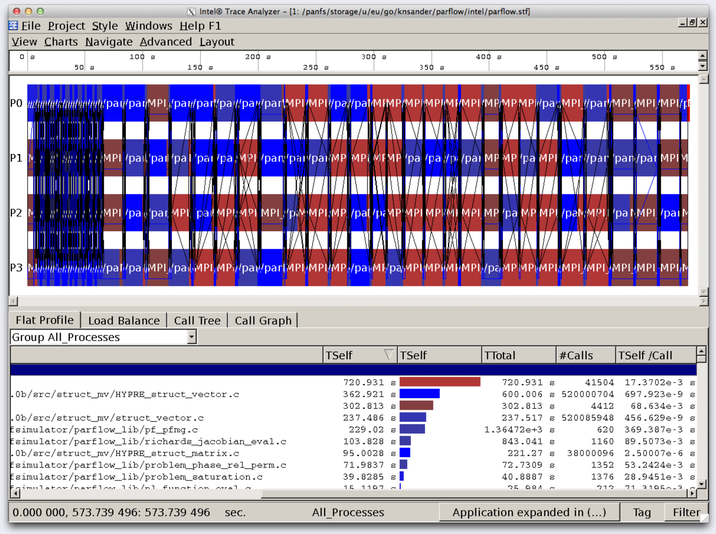
Chow F.T., Kollet, S.J., Maxwell, R.M., and Duan, Q. (2006), Effects of soil moisture heterogeneity on boundary layer flow with coupled groundwater, land-surface, and mesoscale atmospheric modeling, AMS 17th Symposium on Boundary Layers and Turbulence, San Diego.

Technique

The code profiler that provided the most success with instrumenting the ParFlow code was the Intel Trace Analyzer and Collector (ITAC). Instrumentation allows someone to monitor a section of code for the purpose of profiling performance and debugging errors. To instrument with ITAC all it takes is a couple compiler flags, an example:

```
$ mpicc -trace -tcollect example.c
```

If compiled with instrumentation the program can be run with the normal run command and it will create trace files containing all the information about the performance of the program. This data can viewed using the ITAC graphical user interface (GUI):



Difficulties

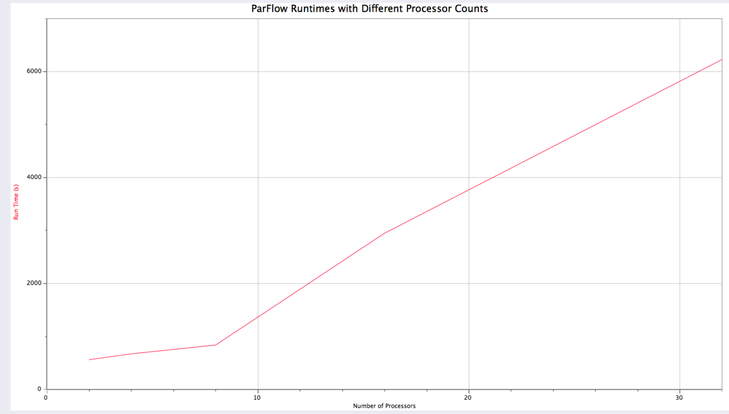
There was a substantial number of setbacks involved with instrumenting and profiling ParFlow.

- Of the three different code profilers tested, two caused the program not to work.
  - TAU would compile the code but would cause ParFlow to get an error when running.
  - Allinea OPT never really worked with its server based implementation.
- The Intel Trace Analyzer GUI would crash with anything other than a really small data set.

Fortunately, because of how ParFlow was designed, it runs similarly on a small scale as it does on a large scale, allowing for some useful information to be gathered even with the numerous setbacks.

Results

Scalability



For the limited trials it shows that the run time appears to increase linearly when adding more processors, however, with ParFlow it is difficult to analyze this without a more in depth look at what is happening. This is due to the fact that increasing the number of processors also increases the work load by a factor that cannot be determined from run time and parameters because it depends on how long it takes for the problem to converge.

Efficiency

This bar graph shows that most of the time is being spent in a function MPI_Waitall, meaning the program is spending most of its time waiting, which is quite problematic. This is likely happening while the processors are waiting on messages from each other. If this is the case, fixing it would involve better synchronizing the processes so that they are ready to send and receive at the same time.

Future Work

Once the updated, bug-free ITAC is released, work can be done to better narrow down what is slowing down ParFlow and attempts can be made to make changes that will increase the speed and efficiency.