Geophysical Research Abstracts
Vol. 16, EGU2014-10923, 2014
EGU General Assembly 2014

Parallel calculations on shared memory, NUMA-based computers using MATLAB

Marcin Krotkiewski (1) and Marcin Dabrowski (1,2)

(1) University of Oslo, PGP, Physics of Geological Processes, Oslo, Norway (marcink@fys.uio.no), (2) Polish Geological Institute - National Research Institute, Wroclaw, Poland

Achieving satisfactory computational performance in numerical simulations on modern computer architectures can be a complex task. Multi-core design makes it necessary to parallelize the code. Efficient parallelization on NUMA (Non-Uniform Memory Access) shared memory architectures necessitates explicit placement of the data in the memory close to the CPU that uses it. In addition, using more than 8 CPUs ($\sim$100 cores) requires a cluster solution of interconnected nodes, which involves (expensive) communication between the processors.

It takes significant effort to overcome these challenges even when programming in low-level languages, which give the programmer full control over data placement and work distribution. Instead, many modelers use high-level tools such as MATLAB, which severely limit the optimization/tuning options available. Nonetheless, the advantage of programming simplicity and a large available code base can tip the scale in favor of MATLAB.

We investigate whether MATLAB can be used for efficient, parallel computations on modern shared memory architectures. A common approach to performance optimization of MATLAB programs is to identify a bottleneck and migrate the corresponding code block to a MEX file implemented in, e.g. C. Instead, we aim at achieving a scalable parallel performance of MATLABs core functionality. Some of the MATLABs internal functions (e.g., bsxfun, sort, BLAS3, operations on vectors) are multi-threaded. Achieving high parallel efficiency of those may potentially improve the performance of significant portion of MATLABs code base.

Since we do not have MATLABs source code, our performance tuning relies on the tools provided by the operating system alone. Most importantly, we use custom memory allocation routines, thread to CPU binding, and memory page migration. The performance tests are carried out on multi-socket shared memory systems (2- and 4-way Intel-based computers), as well as a Distributed Shared Memory machine with 96 CPU cores and 8 NUMA-nodes interconnected with Infiniband network. We present the results of a MATLAB implementation of the STREAM memory bandwidth benchmark and discuss performance problems that we have identified.