Comparing the behaviour of basic linear algebra routines on multicore platforms

Javier Cuenca, Luis P. García, Domingo Giménez, Manuel Quesada
jcuena@um.es, luis.garcia@sait.upct.es, domingo@um.es, manuel.quesada@alu.um.es

Motivation
Multicore architectures are everywhere and can be found in all market segments. There is great interest in the scientific community in using multicore systems efficiently, but without the great effort of reprogramming existing sequential codes.

One possibility is to describe parallelism of the compiler using OpenMP, with a directive-based syntax. OpenMP is incremental and relatively non-invasive. It is not clear which is the best OpenMP compiler in a particular system. It is possible to obtain executables which perform well with fewer threads than cores in the node, although when the number of threads increases, the performance greatly decreases.

Linear algebra functions are widely used in the solution of scientific and engineering problems, and these libraries are used by non-experts in the solution of sequential codes.

Need for a Poly-Compilation Engine: Compilers experimented are:

• P2c: Intel Pentium, 2.8 GHz, with 2 cores. Compilers: icc 10.1 and gcc 4.3.2.
• A4c: Alpha EV68CB, 1 GHz, with 4 cores. Compilers: cc 6.3 and gcc 4.3.
• X4c: Intel Xeon, 3 GHz, with 4 cores. Compilers: icc 10.1 and gcc 4.2.3.
• X8c: Intel Xeon, 2 GHz, with 8 cores. Compilers: icc 10.1 and gcc 3.4.6.

Routines

Routines experimented are:

• R-mvomp: matrix-vector multiplication, $O(n^2)$.
• R-Jacobi: a single iteration of the Jacobi relaxation method for a 2D mesh of $n \times n$ points, $O(n^2)$.
• R-mmmomp: matrix-matrix multiplication, $O(n^3)$.
• R-strassen: an implementation of the Strassen algorithm for matrix multiplication, $O(n^\log_2{7})$.

Experience

Need for a Poly-Compilation Engine: with the information generated with a Poly-Compilation Benchmarking tool, and with theoretical execution time of the routine, the compiled version to use in the solution of a particular problem in a particular system could be decided at running time.

Poly-Compilation Engine

To obtain the values of the parameters in the model, OpenMP primitives ([18]) for threads creation and management have been used:

• R-generate: creates a series of threads with a fixed quantity of work to do per thread, to compare the time of creating and managing threads.
• R-pfor: a simple for loop where there is a significant work inside each iteration, to compare the time of dynamically distributing a set of homogeneous tasks.
• R-barriers: a barrier primitive set after a parallel working area, to compare the times to perform a global synchronization of all the threads.

Example, model of R-jacobi:

• Number of threads ≤ number of cores:

$$P \cdot T_{gen} + \frac{1}{P} \frac{n^2}{T_{work}} = \frac{P}{C} \cdot T_{swrap} + T_{pwr}$$

• Number of threads > number of cores:

$$P \cdot T_{gen} + \frac{n^2}{F} \cdot T_{work} = \frac{P}{C} \cdot T_{swrap} + T_{pwr}$$

Funding
This work has been partially supported by the Consellería de Educación de la Región de Murcia (Fondación Séneca, 08763/PI/08), and by the Ministerio de Ciencia e Innovación (TIN2008-06570-C04-02/TIN).