On the behaviour of the MKL library in multicore shared-memory systems

Domingo Giménez
Departamento de Informática y Sistemas
Universidad de Murcia

Alexey Lastovetsky
School of Computer Science and Informatics
University College Dublin

Jornadas de Paralelismo, Valencia, Septiembre 2010
Matrix multiplication on platforms composed of multicore

The goal:
- To identify the shape matrix multiplication has in a multicore as a function of the problem size and the number of threads, to decide the number of threads to use to obtain the lowest execution time.
- To use this information to develop two-level (OpenMP+BLAS) versions of the multiplication, and select the number of threads in each level.
- To use this information to develop three-level (MPI+OpenMP+BLAS) versions, and select the number of processes and threads in each level.
- To use this information to develop heterogeneous/distributed three-level (MPI+OpenMP+BLAS) versions, and select the number of processes and its distribution or the data partition, and in each processor the number of threads in each level.
Matrix multiplication on platforms composed of multicore

The goal:
- To identify the shape matrix multiplication has in a multicore as a function of the problem size and the number of threads, to decide the number of threads to use to obtain the lowest execution time
- To use this information to develop two-level (OpenMP+BLAS) versions of the multiplication, and select the number of threads in each level
- To use this information to develop three-level (MPI+OpenMP+BLAS) versions, and select the number of processes and threads in each level
- To use this information to develop heterogeneous/distributed three-level (MPI+OpenMP+BLAS) versions, and select the number of processes and its distribution or the data partition, and in each processor the number of threads in each level
Matrix multiplication on platforms composed of multicore

The goal:

- To identify the shape matrix multiplication has in a multicore as a function of the problem size and the number of threads, to decide the number of threads to use to obtain the lowest execution time.
- To use this information to develop two-level (OpenMP+BLAS) versions of the multiplication, and select the number of threads in each level.
- To use this information to develop three-level (MPI+OpenMP+BLAS) versions, and select the number of processes and threads in each level.
- To use this information to develop heterogeneous/distributed three-level (MPI+OpenMP+BLAS) versions, and select the number of processes and its distribution or the data partition, and in each processor the number of threads in each level.
Matrix multiplication on platforms composed of multicore

The goal:

- To identify the shape matrix multiplication has in a multicore as a function of the problem size and the number of threads, to decide the number of threads to use to obtain the lowest execution time
- To use this information to develop two-level (OpenMP+BLAS) versions of the multiplication, and select the number of threads in each level
- To use this information to develop three-level (MPI+OpenMP+BLAS) versions, and select the number of processes and threads in each level
- To use this information to develop heterogeneous/distributed three-level (MPI+OpenMP+BLAS) versions, and select the number of processes and its distribution or the data partition, and in each processor the number of threads in each level
## Systems, basic components

<table>
<thead>
<tr>
<th>name</th>
<th>architecture</th>
<th>icc</th>
<th>MKL</th>
</tr>
</thead>
<tbody>
<tr>
<td>rosebud05</td>
<td>4 Itanium dual-core</td>
<td>11.1</td>
<td>10.2</td>
</tr>
<tr>
<td></td>
<td>8 cores</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rosebud09</td>
<td>1 AMD quad-core</td>
<td>11.1</td>
<td>10.2</td>
</tr>
<tr>
<td></td>
<td>4 cores</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hipatia8</td>
<td>2 Xeon E5462 quad-core</td>
<td>10.1</td>
<td>10.0</td>
</tr>
<tr>
<td></td>
<td>8 cores</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hipatia16</td>
<td>4 Xeon X7350 quad-core</td>
<td>10.1</td>
<td>10.0</td>
</tr>
<tr>
<td></td>
<td>16 cores</td>
<td></td>
<td></td>
</tr>
<tr>
<td>arabi</td>
<td>2 Xeon L5450 quad-core</td>
<td>11.1</td>
<td>10.2</td>
</tr>
<tr>
<td></td>
<td>8 cores</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ben</td>
<td>HP Integrity Superdome</td>
<td>11.1</td>
<td>10.2</td>
</tr>
<tr>
<td></td>
<td>128 cores</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bertha</td>
<td>IBM 16 Xeon X7460 hexa-core</td>
<td>11.0</td>
<td>11.0</td>
</tr>
<tr>
<td></td>
<td>96 cores</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Systems

- **Rosebud (Polytechnic Univ. of Valencia):**
  - Cluster with 38 cores
  - 2 nodes single-processors, 2 nodes dual-processors, 2 nodes with 4 dual-core, 2 nodes with 2 dual-core, 2 nodes with 1 quad-core

- **Hipatia (Polytechnic Univ. of Cartagena):**
  - Cluster with 152 cores
  - 16 nodes with 2 quad-core, 2 nodes with 4 quad-core, 2 nodes with 2 dual-core

- **Ben-Arabi (Supercomputing Centre of Murcia):**
  - Shared-memory + cluster: 944 cores
  - Arabi: cluster of 102 nodes with 2 quad-core
  - Ben: HP Superdome, cc-NUMA with 128 cores

- **Bertha (INRIA Bordeaux Ouest):**
  - Shared-memory cc-NUMA: 96 cores
  - 4 nodes, each node 4 processors, each processor hexa-core
Ben architecture

Hierarchical composition with crossbar interconnection. Two basic components: the computers and two backplane crossbars. Each computer has 4 dual-core Itanium-2 and a controller to connect the CPUs with the local memory and the crossbar commuters. The maximum memory bandwidth in a computer is 17.1 GB/s and with the crossbar commuters 34.5 GB/s. The access to the memory is non uniform and the user does not control where threads are assigned.
Bertha architecture

Machine (191GB)

NUMANode #0 (48GB)

Socket #0

L3 #0 (16MB)

L2 #0 (3072KB)  L2 #1 (3072KB)  L2 #2 (3072KB)

L1 #0 (32KB)  L1 #1 (32KB)  L1 #2 (32KB)  L1 #3 (32KB)  L1 #4 (32KB)  L1 #5 (32KB)

Core #0  Core #1  Core #2  Core #3  Core #4  Core #5

PU #0  PU #1  PU #2  PU #3  PU #4  PU #5
The library is multithreaded.

Number of threads established with the environment variable MKL_NUM_THREADS or in the program with the function mkl_set_num_threads.

Dynamic parallelism is enabled with MKL_DYNAMIC=true or mkl_set_dynamic(1). The number of threads to use in dgemm is decided by the system, and is less or equal to that established.

To enforce the utilisation of the number of threads, dynamic parallelism is turned off with MKL_DYNAMIC=false or mkl_set_dynamic(0).
MKL, results

- rosebud05
- arabic
- rosebud09
- arabic
MKL, results

**hipster8**

**ben**

**bertha**
## MKL, results

<table>
<thead>
<tr>
<th>size</th>
<th>Seq.</th>
<th>Max.</th>
<th>Low.</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>0.0081</td>
<td>0.0042</td>
<td>0.0019 (11)</td>
</tr>
<tr>
<td>250</td>
<td>0.0042</td>
<td>0.0050</td>
<td>0.0012 (5)</td>
</tr>
<tr>
<td>250</td>
<td>0.0035</td>
<td>0.0021</td>
<td>0.0011 (7)</td>
</tr>
<tr>
<td>500</td>
<td>0.026</td>
<td>0.0088</td>
<td>0.0056 (9)</td>
</tr>
<tr>
<td>750</td>
<td>0.087</td>
<td>0.021</td>
<td>0.017 (9)</td>
</tr>
<tr>
<td>250</td>
<td>0.0080</td>
<td>0.0015</td>
<td>0.0013 (9)</td>
</tr>
<tr>
<td>500</td>
<td>0.034</td>
<td>0.063</td>
<td>0.0049 (12)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>size</th>
<th>Seq.</th>
<th>Max.</th>
<th>Low.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.25</td>
<td>0.50</td>
<td>0.058 (16)</td>
</tr>
<tr>
<td>2000</td>
<td>1.8</td>
<td>0.35</td>
<td>0.15 (80)</td>
</tr>
<tr>
<td>3000</td>
<td>6.2</td>
<td>1.2</td>
<td>0.67 (32)</td>
</tr>
<tr>
<td>4000</td>
<td>15</td>
<td>1.9</td>
<td>1.3 (32)</td>
</tr>
<tr>
<td>250</td>
<td>0.021</td>
<td>0.017</td>
<td>0.0014 (10)</td>
</tr>
<tr>
<td>500</td>
<td>0.042</td>
<td>0.033</td>
<td>0.0044 (19)</td>
</tr>
<tr>
<td>750</td>
<td>0.14</td>
<td>0.063</td>
<td>0.010 (22)</td>
</tr>
<tr>
<td>1000</td>
<td>0.32</td>
<td>0.094</td>
<td>0.019 (27)</td>
</tr>
<tr>
<td>2000</td>
<td>2.6</td>
<td>0.39</td>
<td>0.12 (37)</td>
</tr>
<tr>
<td>3000</td>
<td>8.6</td>
<td>0.82</td>
<td>0.30 (44)</td>
</tr>
<tr>
<td>4000</td>
<td>20</td>
<td>1.4</td>
<td>0.59 (50)</td>
</tr>
<tr>
<td>5000</td>
<td>40</td>
<td>2.1</td>
<td>1.0 (48)</td>
</tr>
</tbody>
</table>
Two-level parallelism

It is possible to use two-level parallelism: OpenMP + MKL. The rows of a matrix are distributed to a set of OpenMP threads ($nthomp$). A number of threads is established for MKL ($nthmkl$). Nested parallelism must be allowed, with OMP_NESTED=true or omp_set_nested(1).

```
omp_set_nested(1);
omp_set_num_threads(nthomp);
mkl_set_dynamic(0);
mkl_set_num_threads(nthmkl);
#pragma omp parallel
    obtain size and initial position of the submatrix of A to be multiplied
    call dgemm to multiply this submatrix by matrix B
```
Two-level parallelism, results

Motivation

Systems

Using MKL
Two-level parallelism, results

- **rosebud05**
  - Speed-up for different thread configurations
  - # threads OpenMP - # threads MKL

- **hipatia16**
  - Speed-up for different thread configurations
  - # threads OpenMP - # threads MKL

- **arabi**
  - Speed-up for different thread configurations
  - # threads OpenMP - # threads MKL

- **ben**
  - Speed-up for different thread configurations
  - # threads OpenMP - # threads MKL
Two-level parallelism, results

![Graph showing speed-up for different thread configurations.](image1)

![Graph showing speed-up for different thread configurations.](image2)
Two-level parallelism, conclusions

- In Hipatia (MKL version 10.0) the nested parallelism seems to disable the dynamic selection of threads.
- In the other systems, with dynamic assignment the number of MKL threads seems to be one when more than one OpenMP threads are running.
- When the number of MKL threads is established in the program bigger speed-ups are obtained.
- Normally the use of only one OpenMP thread is preferable.
- In large systems it is preferable to use a higher number of OpenMP threads: in Ben a speed-up between 1.2 and 1.8 is obtained with 16 OpenMP and 4 MKL threads, in Bertha between 1.4 and 1.6 with 8 and 8 threads.
## Two-level parallelism, results

<table>
<thead>
<tr>
<th>size</th>
<th>MKL</th>
<th>ben</th>
<th>Sp.</th>
<th>MKL</th>
<th>bertha</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2-levels</td>
<td></td>
<td></td>
<td>2-levels</td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>0.0014 (10)</td>
<td>0.0014 (1-10)</td>
<td>1.0</td>
<td>0.058 (16)</td>
<td>0.014 (2-24)</td>
</tr>
<tr>
<td>500</td>
<td>0.0044 (19)</td>
<td>0.0043 (4-11)</td>
<td>1.0</td>
<td>0.15 (80)</td>
<td>0.053 (5-16)</td>
</tr>
<tr>
<td>750</td>
<td>0.010 (22)</td>
<td>0.0095 (4-11)</td>
<td>1.1</td>
<td>0.67 (32)</td>
<td>0.51 (16-3)</td>
</tr>
<tr>
<td>1000</td>
<td>0.019 (27)</td>
<td>0.015 (4-10)</td>
<td>1.3</td>
<td>1.3 (32)</td>
<td>0.98 (5-16)</td>
</tr>
<tr>
<td>2000</td>
<td>0.12 (37)</td>
<td>0.072 (4-16)</td>
<td>1.6</td>
<td>1.9 (48)</td>
<td>1.7 (3-32)</td>
</tr>
<tr>
<td>3000</td>
<td>0.30 (44)</td>
<td>0.18 (4-24)</td>
<td>1.7</td>
<td>5.0 (32-4)</td>
<td>2.0</td>
</tr>
<tr>
<td>4000</td>
<td>0.59 (50)</td>
<td>0.41 (5-16)</td>
<td>1.4</td>
<td>12 (32-4)</td>
<td>2.1</td>
</tr>
<tr>
<td>5000</td>
<td>1.0 (48)</td>
<td>0.76 (6-20)</td>
<td>1.3</td>
<td>22 (16-8)</td>
<td>3.0</td>
</tr>
<tr>
<td>10000</td>
<td>10 (64)</td>
<td>5.0 (32-4)</td>
<td>2.0</td>
<td>44 (16-8)</td>
<td>3.0</td>
</tr>
<tr>
<td>15000</td>
<td>25 (64)</td>
<td>12 (32-4)</td>
<td>2.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20000</td>
<td>65 (64)</td>
<td>22 (16-8)</td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25000</td>
<td>130 (64)</td>
<td>44 (16-8)</td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Two-level parallelism, surface shape, in Ben

Execution time with matrix size 5000
only times lower than 1/10 the sequential time
Similar results are obtained with other compilers and libraries.

Ben: gcc 4.4 and ATLAS 3.9.
Matrix multiplication: research lines

- Development of a 2D BLAS prototype, and application to scientific problems
- Simple MPI+OpenMP+MKL version
  - Experiments in large shared-memory (ben), large clusters (arabi), and heterogeneous (rosebud)
- ScaLAPACK style MPI+OpenMP+MKL version
  - Determine number of processors, and OpenMP and MKL threads
  - From the model and empirical analysis or with adaptive algorithm
  - In heterogeneous platform the number of processes per processor
- HoHe ScaLAPACK style MPI+OpenMP+MKL version
  - Determine volume of data for each processors, and OpenMP and MKL threads
  - From the model and empirical analysis or with adaptive algorithm
- Distributed style MPI+OpenMP+MKL version
... and if somebody has access to large cc-NUMA systems, you could repeat some of the tests (code in http://www.um.es/pcgum) and send me (domingo@um.es) the results

thanks!