Cost of Process Migration

Problem Description

We are estimating the time taken(cost) to migrate the process from one core to another. We are using some libraries like <sched.h>, <sys/types.h>, and <unistd.h> to bind different cores at different stages of execution of the program. Also, we need to measure the Level 2 cache miss rate using the performance API library available in C/C++. Therefore, we are using the below PAPI counters to measure the L2 cache miss rate.

PAPI_L2_TCM - Level 2 data cache misses
PAPI_L2_TCA - Level 2 data cache accesses

Solution Strategy

We need to measure the cost of migration of process from one core to another. We use simple and block size of 16 matrix multiplication. We saw significant difference in the matrix multiplication of 1024 dimension. So, for this experiment we use the matrices of size 1024.

I have come up with 4 versions of my code.

First version of code implements simple matrix multiplication for single core i.e. core 0, it initializes the PAPI library, creates event set with PAPI_L2_TCM and PAPI_L2_TCA as 2 events. It binds the current process (i.e. the program of simple multiplication) to core 0. We start PAPI event set and the timer and perform simple matrix multiplication. We stop the time counter and PAPI event set, read the PAPI counter values. We measure the execution time and miss rate for L2 cache and display the L2 cache miss, total L2 cache accesses, miss rate and total execution time.

Similarly, second version of code implements block size 16 matrix multiplication for single core i.e. core 0, it initializes the PAPI library, creates event set with PAPI_L2_TCM and PAPI_L2_TCA as 2 events. It binds the current process (i.e. the program of block size 16 multiplication) to core 0. We start PAPI event set and the timer and perform block size 16 matrix multiplication. We stop the time counter and PAPI event set, read the PAPI counter values. We measure the execution time and miss rate for L2 cache and display the L2 cache miss, total L2 cache accesses, miss rate and total execution time.

For migrating the process, my code 2 versions each for simple and block size 16 matrix multiplication as follows:

Initialize the PAPI library, create 2 event set for measuring L2 cache miss rate for core 0 and core 7 as mentioned before. It binds the current process (i.e. the program of simple multiplication) to core 0. We start PAPI event set and the timer and perform simple matrix multiplication. The program will run only for half of the rows of the matrix in core 0. That means the most outer for loop runs from 0 to 511. We stop PAPI event set, read the PAPI counter values and reset the PAPI event set.

We zero out the mask and set the mask to 7, to bind the process to core 7. We then start the PAPI event set and perform simple matrix multiplication. The program will run only for next half of the rows of the matrix in core 7. That means the most outer for loop runs from 512 to 1023. We stop the time counter and PAPI event set, read the PAPI counter values. We measure miss rate for L2 cache for core 0 and core

Cost of Process Migration

7. Display the L2 cache miss, total L2 cache accesses, miss rate, and total execution time. We reset the PAPI event set.

Similar version of program is used for block size 16 matrix multiplication.

Before executing the program, I made sure to acquire a node with 8 cores using the below command:

salloc -N 1 -n 8 –exclusive

And after all the programs are ran we can use “exit” command, which helps in releasing the resources i.e. 1 node that has 8 cores.

To note a significant difference in the execution time between process ran on single core and process ran in 2 cores. We cannot estimate the cost from a single run for both versions. So, we perform 5 runs for each program and record the measurements. We find the average time taken by the single core and 2 cores for a simple and block size 16 matrix multiplication.

Description of Resources

We are using the whale cluster for conducting this experiment.

The hardware information that was used when the experiment was conducted are as follows:

PAPI Version – 5.6.0.0

Operating System – Linux

Vendor – AMD

Model – QUAD Core AMD Opteron Processor

CPU Max MHz – 2211

Number of Cores – 8

Number of Hardware Counters – 3

Cores per socket – 4

Socket – 2

Cache – Level 1 and Level 2 cache are available.

The above information can be fetched using the “papi_avail” command or we can use the “PAPI_get_hardware_info” method.

Cost of Process Migration

Measurements

Simple Matrix Multiply (1024)

Level 2 Cache Simple Multiply 1024 Core 0				
Index	Cache misses	Total cache access	Miss rate	Total execution time in micro sec
1	1112751840	2069656024	0.537650618	64657455
2	1109315962	2145141882	0.517129413	64447558
3	1109755551	2076147568	0.534526335	64264395
4	1117671321	2178122029	0.513135309	63665243
5	1113023801	2213082262	0.502929249	64306973

Average Miss rate for L2 cache = 0.521074185

Average Execution time = 64268324.8 = 64.27 sec

Level 2 Cache Simple Multiply 1024 Core 0 and Core 7							
	Core 0			Core 7			
Index	Cache misses	Total cache access	Miss rate	Cache misses	Total cache access	Miss rate	Total execution time in micro sec
1	557948516	1096196852	0.508985694	556658155	1109425605	0.501753477	64376446
2	559813968	1087316821	0.514858188	556324083	1104620925	0.503633482	65008550
3	556903686	1094180083	0.508968948	560678109	1029007561	0.544872681	64024647
4	558708300	1088781572	0.513150033	556179486	1124752316	0.494490634	63926634
5	554493431	1112826502	0.498274825	551560829	1144862608	0.481770323	67322329

Average Miss rate for L2 cache for Core 0 = 0.508847538

Average Miss rate for L2 cache for Core 7 = 0.50530412

Average Execution time = 64931721.2 = 64.93 sec

Time difference = 663396.4 = 0.66 sec

Block size 16 Matrix Multiply (1024)

Level 2 Cache Block 16 Multiply 1024 Core 0				
Index	Cache misses	Total cache access	Miss rate	Total execution time in micro sec
1	12630055	1296482868	0.009741783	20117924
2	12549937	1203363402	0.01042905	19840624
3	12650303	1303483859	0.009704994	20111464
4	12662319	1306953234	0.009688425	20094707
5	12637617	1307637336	0.009664466	20091184

Average Miss rate for L2 cache = 0.009845744

Average Execution time = 20051180.6 = 20.05 sec

Cost of Process Migration

Level 2 Cache Block 16 Multiply 1024 Core 0 and Core 7							
	Core 0			Core 7			
Index	Cache misses	Total cache access	Miss rate	Cache misses	Total cache access	Miss rate	Total execution time in micro sec
1	6204233	959518594	0.006465985	6244693	961714906	0.006493289	21267260
2	5903548	907612183	0.006504483	6018318	857219091	0.007020747	21037894
3	6159083	960329003	0.006413513	6123044	962148170	0.00636393	21279686
4	6040927	998617572	0.00604929	6001436	981322195	0.006115663	21225824
5	5752144	909167844	0.006326823	5788009	872030698	0.006637391	21070697

Average Miss rate for L2 cache for Core 0 = 0.006352019

Average Miss rate for L2 cache for Core 7 = 0.006526204

Average Execution time = 21176272.2 = 21.18 sec

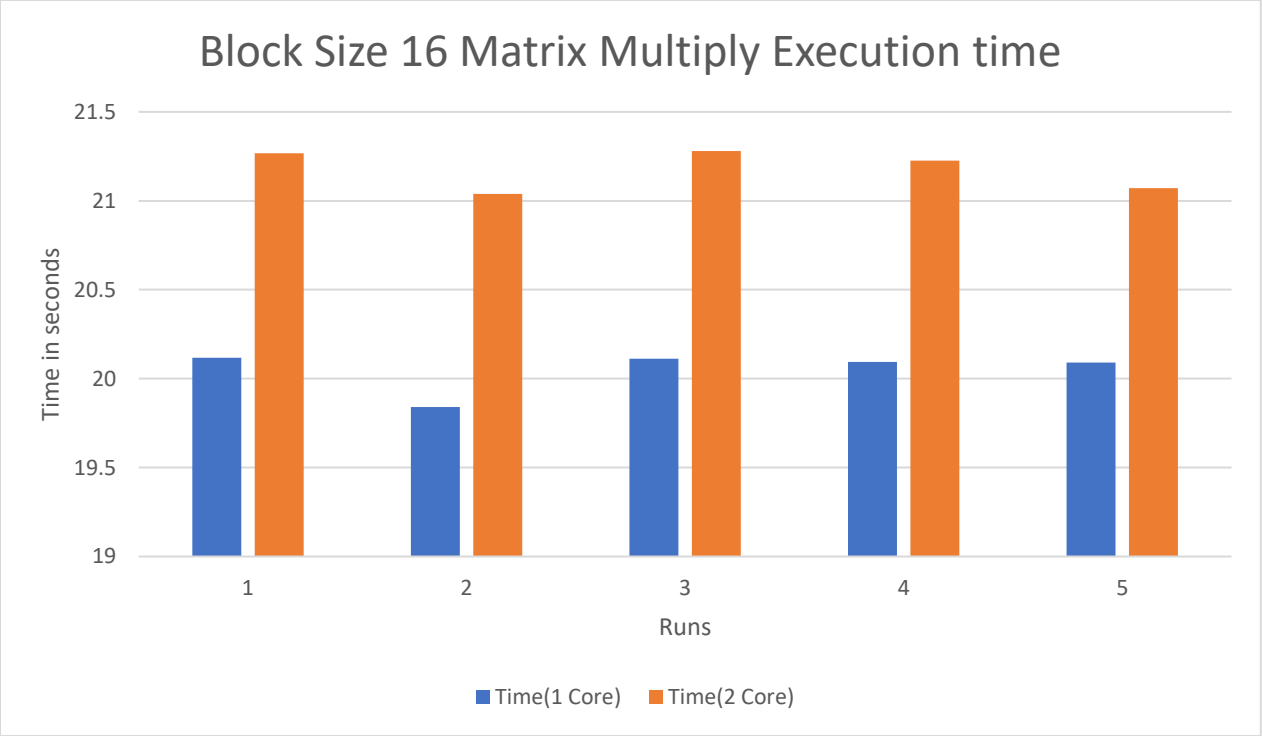
Time difference = 1125091.6 = 1.13 sec

Comments

From the above measurements we deduce that process migration comes with a cost associated with it. From the experimental results, the time taken for migration takes approximately around 1 second.

Execution Time

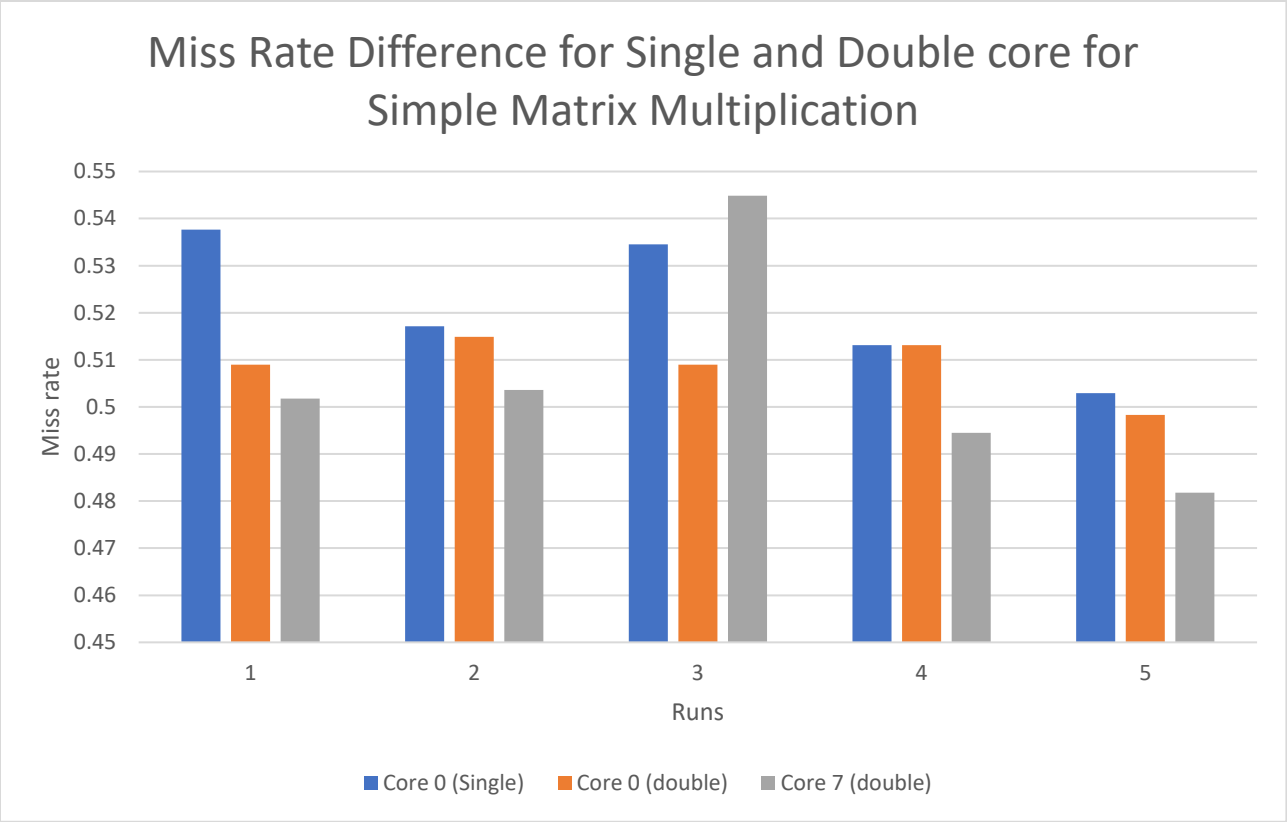
Cost of Process Migration



As expected, we observe that the average execution time for 5 runs for both simple and block size 16 matrix multiplication adds up a time cost for migrating a process from one processor to another.

Cost of Process Migration

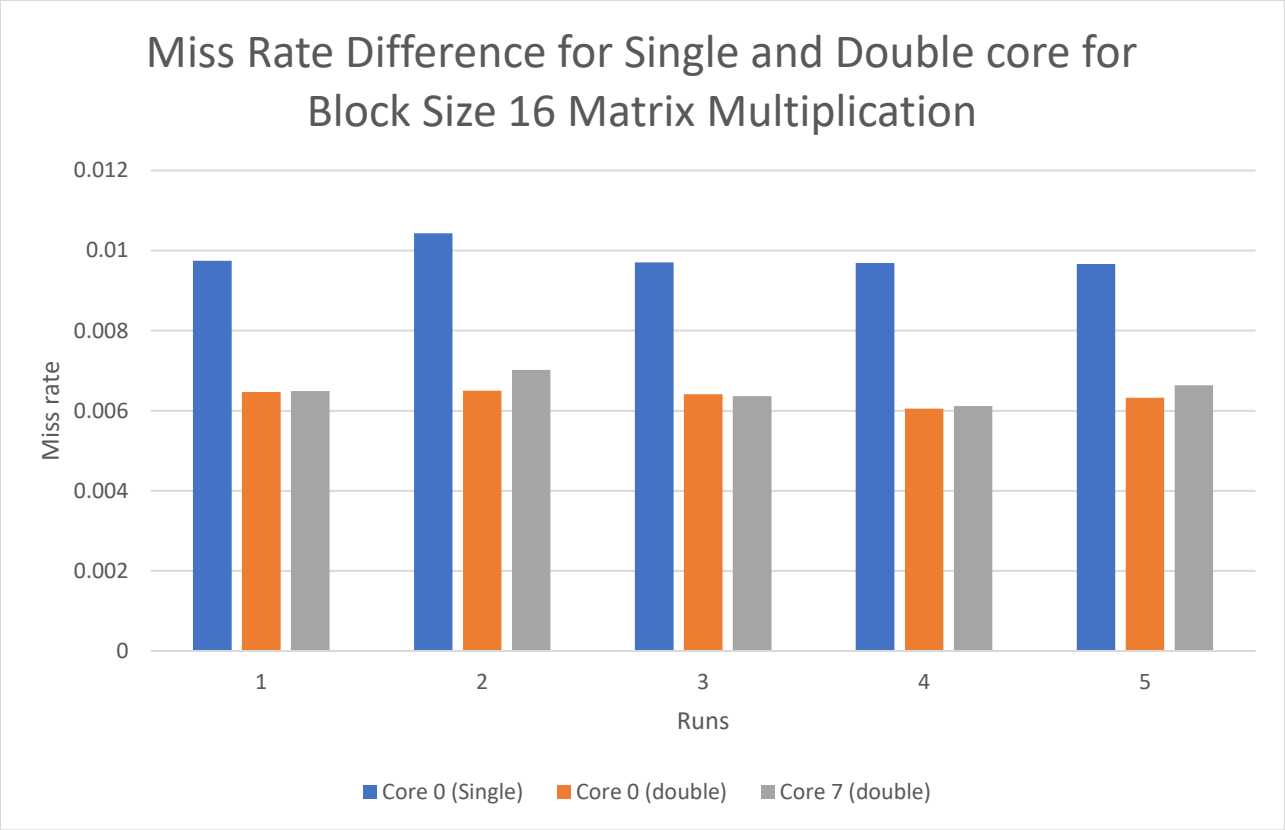
Miss Rate



We observe that level 2 cache miss rate is usually lower when we run the program in 2 different cores as compared to running in a single core for simple matrix multiplication.

The miss rate for core 0 and core 7 vary significantly.

Cost of Process Migration



Similar implications can be derived from the above diagram.

We observe that level 2 cache miss rate for Core 0 is higher as compared to when ran with 2 cores for block size 16 matrix multiplication. Additionally, miss rate of core 0 and core 7 are similar for a run.

We observe that overall the miss rate is greater for Simple matrix multiplication than for block size matrix multiplication which is what we expect.

We observe what was expected. We expected that there might be some cost of migrating the process from one core to another. Our experiments suggest that it is true.