Simultaneous Multi-Threading, performances and power/energy consumption

Flavia Monti, 1632488
Seminars in Advanced Topics in Computer Science Engineering

1. INTRODUCTION

Simultaneous Multi-Threading (SMT) is the capability of a single physical processor to simultaneously execute instructions from more than one single hardware thread. For a single core there are two threads (each represented by a separate architectural state) which share the processor's execution engine and the bus interface. SMT uses logical processors from different processors, so on a single core I can execute on one thread (the physical core) or on two threads (the physical and the logical cores).
SMT has been shown to be energy efficient most of the time increasing however power consumption.

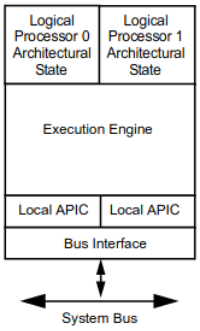


Figure 1. Processor with Two Logical Processors Supporting SMT

In this document is analyzed the relation between energy, power and performance of SMT. Important benchmarks are executed with different level of SMT and with different P-state frequencies, performance counters are measured using the tool LIKWID (Like I Knew What I Am Doing) to collect information and to analyze it.

2. RELATED WORK AND BACKGROUND

Many researchers studied the area of performance and energy efficiency of SMT processors, all these studies bring to the fact that increased processor utilization in STM will impact power. Li et al. study the area overhead, focusing on understanding the power-performance efficiency of SMT in the context of a Power4-like architecture. Seng et al. study power-aware optimizations for multi-threaded Alpha processor. Sasanka et al. study energy efficiency of CMP and SMT for multimedia workloads, and also Kaxiras et al. do the same for mobile phone workloads on a digital signal processor.

When we need to measure performances of a system the main aspect to consider are performance counters. There are many events that hardware designer look at in order to measure them and make decisions. PCM (Performance Counter Monitor) is an hardware supported method that controls these events and measure the CPU utilization of the system.
Performance counter registers are those registers that can be read by a user and can write only in supervisor mode. MSR are registers used for debugging, program execution tracing, computer performance monitoring and toggling certain CPU features.
There exist many software that are able to read and write MSRs like msr-tools package, perf, Intel Vtune, Amplifier XE, PTU, Oprofile and LIKWID. The last one is the tool used in this study, it is a lightweight, simple and efficient command line tool and it is compatible with a big number of architectures.

3. GOALS AND METHODOLOGY

In this study we want to find some indicators that tell us if it is convenient or not to switch on STM and schedule both on physical and logical cores. We test different benchmarks with a specialized tool named LIKWID that access to performance counters of the system.

LIKWID is a command line tool that accesses to MSR module and can read out hardware performance counters. MSR registers are located in `\dev\cpu\*\msr` where * indicates the number of the `cpu`.
`likwid-perfctr` is the specific tool to access to performance counters of the system and it can be used as wrapper application to get the results relative to the execution of a particular application. LIKWID provides different events and different performance groups to be measured that need to be specified with the option `-g` when `likwid-perfctr` is called. Together with this option we need also to specify a core id lists so that the application will be pinned to these cores. With the option `-C` we set these cores following the variants of `likwid-pin`. We can see the thread topology of the system with `likwid-topology` command, looking at it we can create a right core id list for our needs.

Likwid gives the possibility to create custom performance groups in this way we can measure what we need for our purpose. In our study we are going to measure metrics related to power and energy consumption and below

is reported the event set and all the metrics we are going to use.

- Event Set:
  - FIXC0 = INSTR_RETIRED_ANY
  - FIXC1 = CPU_CLK_UNHALTED_CORE
  - FIXC2 = CPU_CLK_UNHALTED_REF
  - TMP0 = TEMP_CORE
  - PWR0 = PWR_PKG_ENERGY
  - PWR1 = PWR_PP0_ENERGY
  - PWR3 = PWR_DRAM_ENERGY
  - UBOXFIX = UNCORE_CLOCK
  - PMC0 = L2_TRANS_ALL_REQUESTS
  - PMC1 = L2_RQSTS_MISS
  - PMC2 = MEM_LOAD_RETIRED_L3_ALL
  - PMC3 = MEM_LOAD_RETIRED_L3_MISS
- Metrics:
  - Runtime (RDTSC) [s] = time
  - Runtime unhalted [s] = FIXC1*inverseClock
  - Clock [MHz] = 1.E-06*(FIXC1/FIXC2)/inverseClock
  - Uncore Clock [MHz] = 1.E-06*UBOXFIX/time
  - CPI = FIXC1/FIXC0
  - Temperature [C] = TMP0
  - Energy [J] = PWR0
  - Power [W] = PWR0/time
  - Energy PP0 [J] = PWR1
  - Power PP0 [W] = PWR1/time
  - Energy DRAM [J] = PWR3
  - Power DRAM [W] = PWR3/time
  - L2 request rate = PMC0/FIXC0
  - L2 miss rate = PMC1/FIXC0
  - L2 miss ratio = PMC1/PMC0
  - L3 request rate = PMC2/FIXC0
  - L3 miss rate = PMC3/FIXC0
  - L3 miss ratio = PMC3/PMC2

The event set list all the counters with the relative events. The first three counters, FIXC0, FIXC1 and FIXC2, are fixed counter. FIXC0 counts the number of instruction retired, FIXC1 counts the unhalted cycles of the processor core and FIXC2 counts the number of reference cycles at the TSC (Time Stamp Counter) rate when the core is not in a halt state. Then there are many PWR# counters that measure the current energy consumption through the RAPL interface. Depending on # the energy is relative to different RAPL domains: 0 is for Package (PKG), 1 is for Power Plane 0 (PP0) and 3 is for DRAM. Generally PKG refers to the processor package while PP0 refers to processor cores. The processor provides one register to measure the current core temperature (TMP0) and provides measurements for the global uncore management unit (UBOXFIX). At the end there are 4 general-purpose counters associated to events involving L2 and L3 caches. PMC0 is associated to the event relative to the transactions that requests L2 cache, PM1 to the requests that miss L2 cache, PM2 counts the retired load instruction that hit L3 cache and PM3 counts all the retired instruction that missed in L3 cache.

Benchmarks are one of the most important aspect to consider in evaluating the results of new studies. We are going to test different benchmarks of NAS Parallel Benchmark suite.
NPB suite provide benchmarks derived from computational fluid dynamics (CFD) applications and also benchmarks for unstructured adaptive meshes, parallel I/O, multi-zone applications and computational grids. Depending on the version of NPB there are different implementations that uses OpenMP, MPI, serial,... and there are also different sizes of each application. For each problem size there is a different class that we can specify when compiling benchmarks.

1) BT (Block Tridiagonal): Solves a synthetic system of nonlinear PDEs using an algorithm involving block tridiagonal.
2) CG (Conjugate Gradient): This benchmark uses the invers epower method to find an estimate of the largest eigenvalue of asymmetric positive definite sparse matrix with a random pattern of nonzeros.
3) EP (Embarrassingly Parallel): Generates pairs of Gaussian random deviates according to a specific scheme and tabulate the number of pairs in successive square annuli.
4) FT (Fast Fourier Transform): Numerically solves a certain partial differential equation PDE using forward and inverse FFTs.
5) IS (Integer Sort): Sorts N keys in parallel.
6) LU (Lower-Upper symmetric Gauss-Seidel): Solves PDEs using symmetric successive over-relaxation (SSOR) solver kernels.
7) MG (MultiGrid): Four iterations of the V-cycle multigrid algorithm are used to obtain an approximate solution u to the discrete Poisson problem $\Delta^2 u = v$ on a $256 \times 256 \times 256$ grid with periodic boundary conditions.
8) SP (Scalar Pentadiagonal): Solves a synthetic system of nonlinear PDEs using an algorithm involving scalar pentadiagonal.
9) UA (Unstructured Adaptive): Solves Heat equation with convection and diffusion from moving ball. Mesh is adaptive and recomputed at every 5th step.

All the benchmarks have been tested under different levels of SMT and on different P-state frequencies. To execute all the scripts we used a machine with 2 NUMA nodes each with 10 cores but only one NUMA node is used for our study. The machine has 10 physical cores and 10 logical cores and there are 65 combinations of these 20 cores to test the benchmarks activating SMT on different level or not. The machine has the possibility to set P-state on 14 different frequencies varying from 1000000kHz to 2201000kHz.

4. EXPERIMENTAL EVIDENCE

We measured the performance in terms of the metrics with respect to the number of physical and logical cores.

Results for the case of energy consumption are reported in Figure 2, with some plots of *bt.A.x* benchmark. It is possible to see from these plots how the energy consumption

in all the cases decreases when the number of cores increase and also decrease when we set greater P-state frequencies. In particular we can state that in all the cases in which we use only physical cores the energy consumption is smaller with respect to the case in which we use both physical and logical cores. This results are valid also for the other benchmarks except for *mg.A.x*, in Figure 3, where we can see that the energy initially decrease, remain constant and then increase. Looking at the collection of results relative to the same number of total cores we can see that if the number of logical cores is far from the number of physical cores, the energy is greater than when the number of logical cores is like the number of physical cores.

The behaviour of the energy is the same also for energy consumption of DRAM, it decrease with increasing the number of cores.

In Figure 4 there are the results of power consumption of *ft.A.x* benchmark. Recalling what we said in the introduction we expect to find an increasing of the power. In the plots we can see how the power consumption increases with the number of total core and obviously increases with the value of frequencies. An important point is the fact that when using only physical cores, power consumption is higher than using both physical and logical cores. As before there is a special case for *mg.A.x* where the trend of the power increase, stabilized and then decrease.

Power consumption of DRAM behave like what said now but, increasing the number of cores, it increases, remains constant on high value and when we have higher number of cores values seems to decrease.

Another important result is relative to runtime execution. Time decreases very fast with increasing the number of cores used. This is what we expected to find by looking at the results of power and energy. Usage of SMT with all the cores is faster than using only one core. In Figure 5 we have an example with a P-state at 1500000kHz, for *sp.A.x* we have a running time over 40s using 1 physical core on one side and we have about 5s using 10 physical cores and 10 logical cores. The point where the values decrease very fast is when we pass from the usage of only 1 physical core to the usage of more than 3 cores. High values of power consumption are due to the fact that we reach small values of runtime and this values are much much smaller than the values of the energy.

Results for the metrics CPI are reported in Figure 6, where there are example of plots relative to *bt.A.x*, *cg.A.x* and *ep.A.x*. CPI formula involves the number of instruction retired by a benchmark and the clock cycles when core is not in halt state. In all the cases CPI varies little with different values of P-state frequency. Analyzing all the results we can say that when we use only physical cores without any logical cores, values of CPI are lower than using both physical an logical cores. When using both physical and logical core there are some aspect to consider. On one side if the number of logical and physical core are equal the values of CPI per each core does not vary a lot and remain around some value. On the other side if the number of physical and logical core are not equal (so physical cores are greater then logical ones), the values of CPI per core vary a lot. Overall CPI increase with the number of cores used.

Results on L2 cache show how the request rate values increase with the number of cores. Values vary much more when not using same number of physical and logical cores while does not vary a lot when using all physical cores or same number of physical and logical cores while values per cores. In particular when using only physical cores L2 request rates assume lower values than the rest. Results on miss rate have not a particular behaviour, there are cases where we have a lot of values that vary a lot and cases in which values are constant.

L3 cache's results show that request rate and also miss rate assume very low values. We have a little increment with increasing the number of cores but most of the values concentrate on low values.

5. CONCLUSION

This study analyze the effect of SMT on the execution of NAS benchmarks.
The results show that energy consumption is very efficient but at the same time the power consumption increase.
An important aspect is the fact the fact that when we do not activate SMT, energy consumption is lower than activating SMT and execute on both logical and physical cores, but the discussion is contrary when we analyze power.
Overall, when using same number of total cores it is better to use both physical and logical cores and not only physical cores.

References

[1] D. Baily, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, S. Weeratunga, *The NAS Parallel Benchmarks*, NAS Technical Report RNR-94-007, NASA Ames Research Center, Moffett Field CA, March 1994

[2] C. Cao Minh, J. Chung, C. Kozyrakis, and K. Olukotun, *STAMP: Stanford Transactional Applications for Multi-processing*, IISWC '08: Proceedings of The IEEE International Symposium on Workload Characterization

[3] *Intel® 64 and IA-32 Architectures Software Developer's Manual. Volumes 3A, 3B, 3C, and 3D: System Programming Guide, Part 2*

[4] *Intel® Performance Counter Monitor - A Better Way to Measure CPU Utilization*, http://www.intel.com/software/pcm

[5] Y. Li, D. Brooks, Z. Hu, K. Skadron, and P. Bose,*Understanding the energy efficiency of simultaneous multithreading*, in Proc. ISLPED'04, August 2004

[6] *NAS Parallel Benchmarks Changes*, NASA Advanced Supercomputing Division, https://www.nas.nasa.gov/publications/npb.html

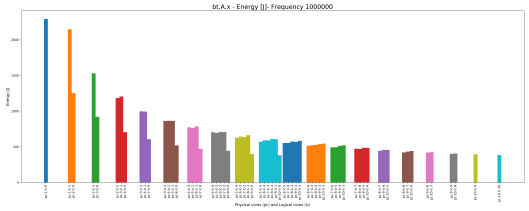
[7] R. Sasanka, S. V. Adve, Y. K. Chen, and E. Debes, *The energy efficiency of CMP vs. SMT for multimedia workloads*, in Proc. 18thICS, June 2004

[8] C. Sakalis, C. Leonardsson, S. Kaxiras, and A. Ros, *Splash-3: A properly synchronized benchmark suite for contemporary research*, in Performance Analysis of Systems and Software (ISPASS) 2016, IEEE International Symposium On 2016

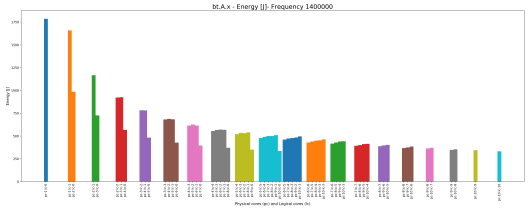
[9] J. Seng, D. Tullsen, and G. Cai, *Power-sensitive multithreaded architecture*, in Proc. ICCD 2000

[10] J. Treibig, G. Hager and G. Wellein, *LIKWID: A lightweight performance-oriented tool suite for x86 multicore environments. Proceedings of PSTI2010, the First International Workshop on Parallel Software Tools and Tool Infrastructures*, San Diego CA, September 13, 2010. DOI: 10.1109/ICPPW.2010.38 Preprint: http://arxiv.org/abs/1004.4431

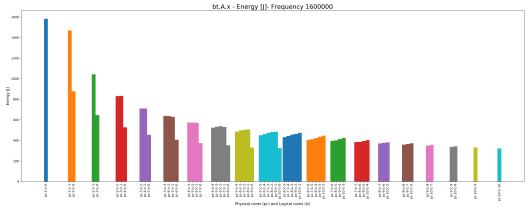
Figure 2. Energy consumption of bt.A.x



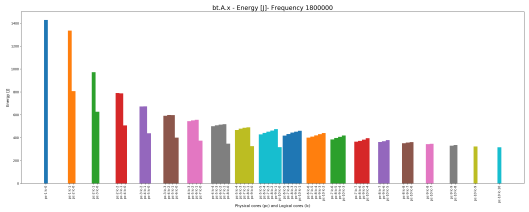
(a) Frequency = 1000000kHz



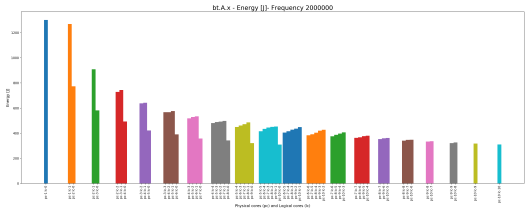
(b) Frequency = 1400000kHz



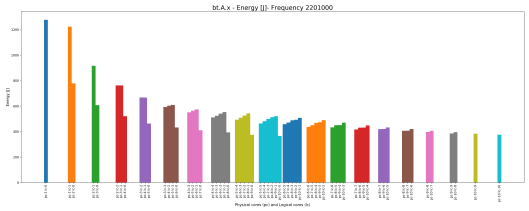
(c) Frequency = 1600000kHz



(d) Frequency = 1800000kHz

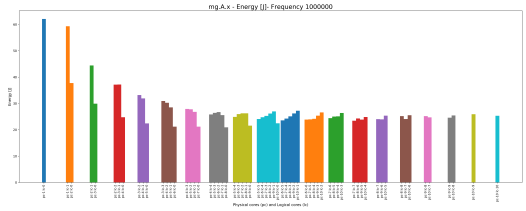


(e) Frequency = 2000000kHz

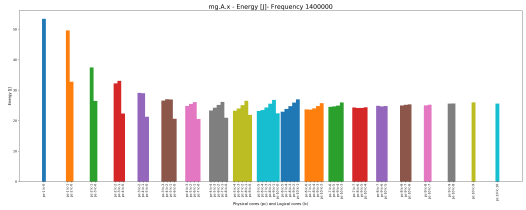


(f) Frequency = 2201000kHz

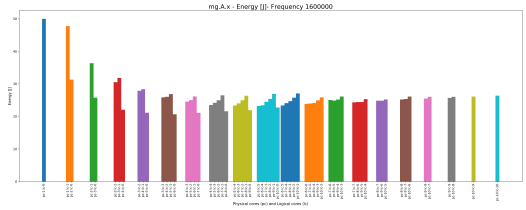
Figure 3. Energy consumption of mg.A.x



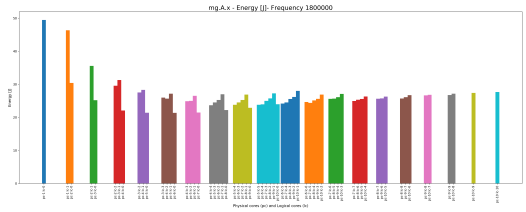
(a) Frequency = 1000000kHz



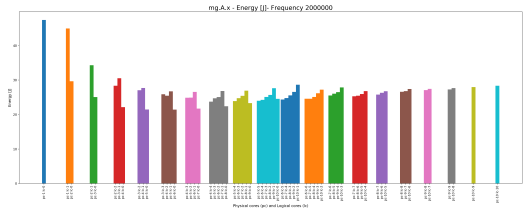
(b) Frequency = 1400000kHz



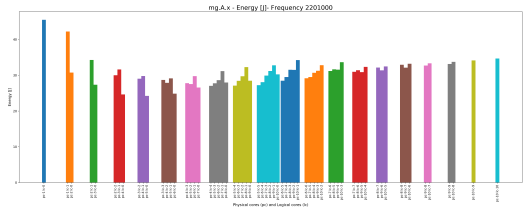
(c) Frequency = 1600000kHz



(d) Frequency = 1800000kHz

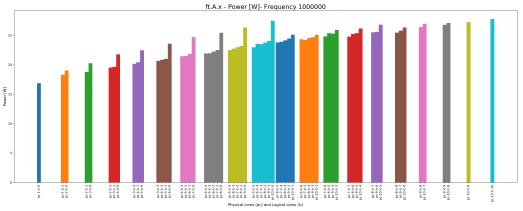


(e) Frequency = 2000000kHz

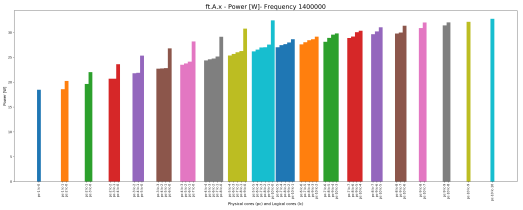


(f) Frequency = 2201000kHz

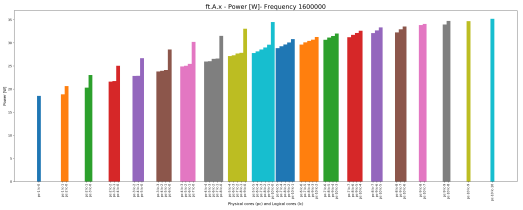
Figure 4. Power consumption of ft.A.x



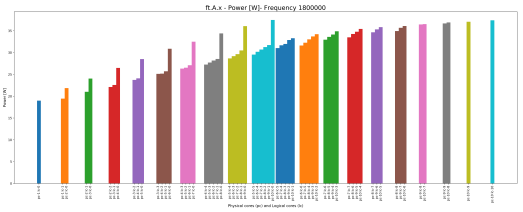
(a) Frequency = 1000000kHz



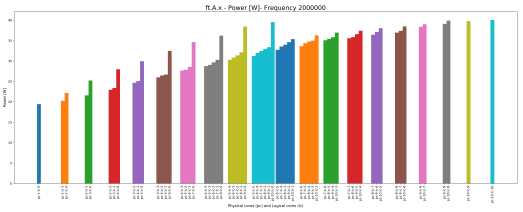
(b) Frequency = 1400000kHz



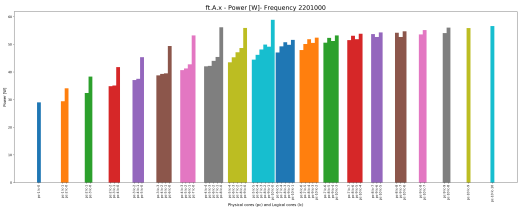
(c) Frequency = 1600000kHz



(d) Frequency = 1800000kHz



(e) Frequency = 2000000kHz



(f) Frequency = 2201000kHz

Figure 5. Runtime of sp.A.x with P-state at 1500000kHz

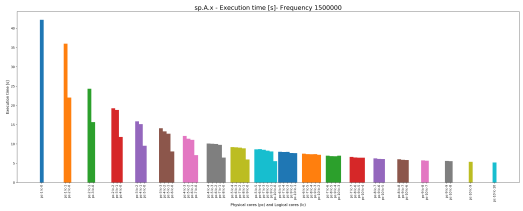
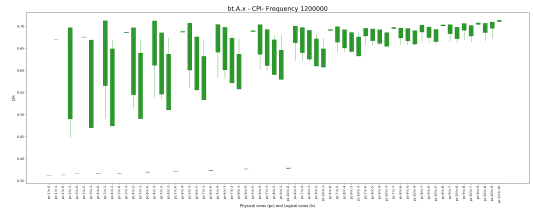
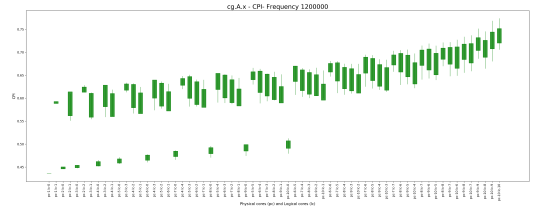


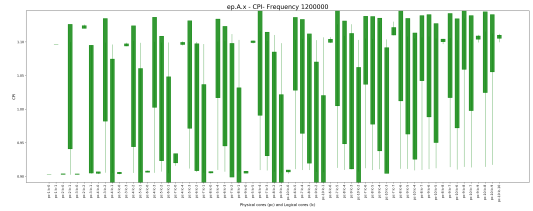
Figure 6. CPI with P-state at 1200000kHz



(a) bt.A.x



(b) cg.A.x



(c) ep.A.x