Small side-core for efficient I/O virtualization

Chung Lee, Peter Strazdins, Steve Blackburn, Eric McCreath

ANU College of Engineering & Computer Science

INTRODUCTION

• I/O virtualization and *side-core*

I/O performance is a key consideration for virtualized high performance computing (HPC) clusters. Although software approaches are the most flexible way, without special hardware or removing some important virtualization features the performance is still unbearable. Several studies show that the *side-core* approach to schedule I/O tasks to a dedicated core can improve the performance close to that of bare-metal.

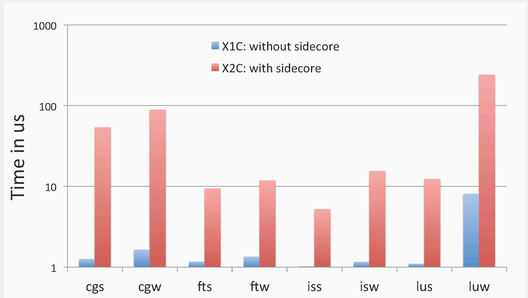

Fig 1. Comparison of the performance of HPC applications of different network I/O patterns: side-core shows big improvement.

• Drawback of (big) side-core

As shown in Fig 1, the side-core makes a big improvement but using a full featured big core for simple I/O tasks can be wasteful in terms of energy and utilization.

• Small side-core

Considering the relative simplicity of IO tasks, scheduling them on small-core(s) while keeping big cores occupied with complex applications is a solution for getting better performance as well as preventing wasted resources.

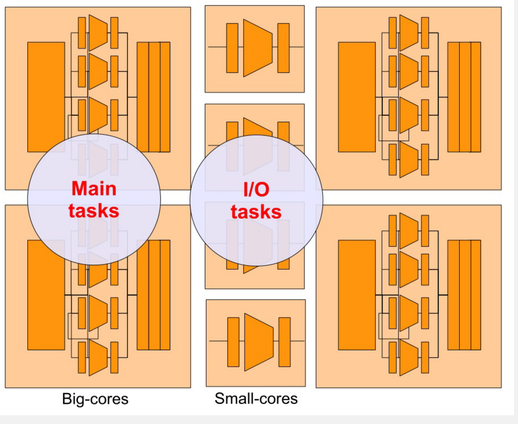

Fig 2. Small side-core: Big cores for virtual machines running complex applications while small cores for I/O tasks

OBJECTIVES

Identify essential micro-architectural features and evaluate the configuration of the small core.

A small core is defined as containing significantly fewer transistor than the big. The goal is to replace the big core with the little, without the performance suffering.

Considerations include:

- Memory system (L1/L2 cache) and TLB
- HW prefetch
- Branch predictor and indirect branch predictor
- Frequency
- In-order/Out-out-order

METHODS

Mimic AMP and compare the performance.

- Page-coloring for examining the effect of the size of L2 cache
- Frequency scaling for sensitivity of the performance to the frequency
- Turning on/off micro-architectural features. Such as the *hw prefetch* and *indirect branch predictor*

Compare the performance and micro-architectural events of different processors.

Table 1. Processors used in experiments

Processor/Spec	Atom (D525)	Phenom (X945)	FX-8350
# of cores	4T/2C	4C	8C
Max. Freq (Mhz)	1800	3000	4000
L1 ic/dc (kB)	32/24	64/64	64/16
L2 cache (kB)	512	512	2048
L3 cache (MB)	None	6	8
L1 TLB (i/d)	32/16	32/48	48/64
L2 TLB (i/d)	None/64	512/512	512/1024
Branch Predictor	2-level adaptive	2-level adaptive	Tornament (hybrid)
BTB	128	2048	512/5120
Indirect BTB	None	512	512
Instruction scheduling	In-order	Out-of-order	Out-of-order
# of transistors	176M	758M	1.2B

• Benchmarks:

- Common network metrics such as MPI bandwidth and latency from the OSU micro-benchmarks suite.
- MPI applications with different communication patterns such as *cg*, *ft* and *lu* from NAS Parallel benchmarks.

RESULTS

• L2 cache size and the performance

Using page-coloring, we partition Phenom L2 cache of 512kB. The size of 256kB shows almost the same performance for all applications.

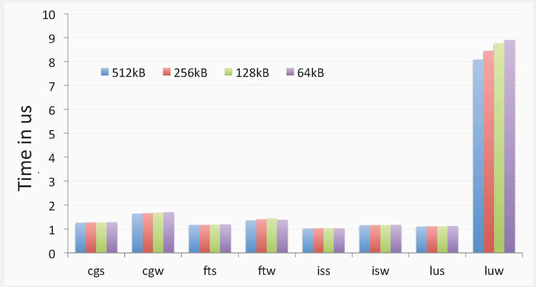

Fig 3. Application performance comparison according to L2 cache size: all configurations show less than 10% degradation.

• Frequency and the performance

Using a higher frequency makes a big performance improvement but don't expect it will continue forever.

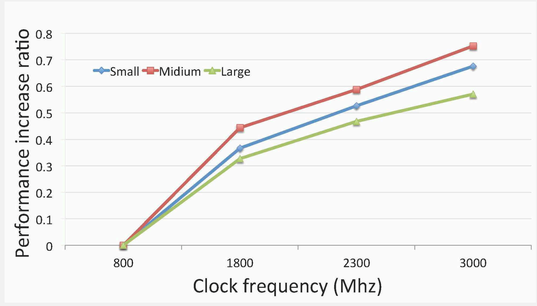

Fig 4. Network throughput increase according to the frequency (800Mhz to 3Ghz)

• Micro-architectural statistics

Other micro-architectural features that cannot be configured are compared using a system profiler and they are summarised in table 2.

Table 2. Miss ratio on each experimental processor

Miss ratio	Atom	Phenom	FX-8350
BP	9.5	4.8	3.2
L1D	7.6	3.9	8.4
L1I	4.5	6.4	10.2
ITLB	2.0	1.9	2.4
DTLB	1.5	1.5	1.2

CONCLUSIONS

Small side-core specification:

Freq	L1 ic/dc	L2 cache	L1 TLB	BP	BTB
On demand	64/16kB	256kB	32/16	2-level adaptive	2048

- HW data prefetch and indirect BP do not contribute to the performance.
- FPU is almost never used.
- Branch predictor: We choose 2K. FX8350 BP gives the best performance but requires huge buffers. Phenom has less than half size of the FX-8350 BTB but The difference of miss ratio is only 1.6%.
- L1D: 16kB has been chosen. The latency can be hidden by exploiting more memory level parallelism.
- L1I: We select 64kB because instruction misses may cause the whole pipe-line to stall and it's penalty is not easy to hide. Thus we treated it with more importance.

FUTURE WORKS

Simulation of the small side-core

For evaluating the whole design of the small side-core and fine tuning, we will run processor simulations. Also, we expect the simulations would give us some insights about following questions.

Is out-of-order (OOO) useful?

From experimental results we are observing OOO processors outperform the in-order execution on the Atom processor, but due to other configuration differences we cannot determine if it is the only contributing factor.

To investigate, we will run a simulation of the processor and measure the distribution of memory access patterns. This will be used to explain the number of isolated and overlapped misses and the associated penalty.

Is the small side-core specification the smallest possible?

There would be some simulations for finding efficient configuration that is impossible on the real machines

CONTACT

E-mail: brian.lee@anu.edu.au