Automatic Single Core Performance Analysis on IBM POWER6

Yury Oleynik
oleynik@in.tum.de
Technische Universität München

Outline

- Motivation for single core performance analysis automation
- Performance Property
- IBM POWER6 CPU
  - Overview
  - Execution pipeline and Periscope Perfromance Properties derivation
    - Instruction fetching
    - Instruction dispatching
    - Memory access
    - Floating Point instructions
    - Memory access pattern analysis
- Measuring single core performance with Periscope
- POWER6 Search Strategies
- Demonstration of Automatic Single Core Performance Analysis with Periscope
- Conclusion

Motivation

- Single core performance tuning on POWER6 is tough!
  - Complicated microarchitecture
    - Lack of documentation
  - Ultra-high frequency
  - In-order execution
  - Weird execution specifics
  - Complicated HW counters interpretation
    - 552 HW events available
    - 2 lines of event description
    - Only 4 counters available
  - Poor single core performance analysis tools support
    - General problem for single core perf. analysis
- Goal: Single Core Performance Analysis Automation
- Tasks:
  - Understand architecture
  - Identify potential bottlenecks
  - Map bottlenecks to available HW events
  - Validate bottlenecks against test kernels
  - Automate bottlenecks evaluation

Periscope Performance Property

- Based on APART Specification Language:

“ ***Performance Property:*** A performance property (e.g. load imbalance, communication, cache misses, redundant computations, etc.) characterizes a specific performance behavior of a program (or part) and can be checked by a set of *conditions*. Conditions are associated with a *confidence value* (between 0 and 1) indicating the degree of confidence about the existence of a performance property. In addition, for every performance property a *severity measure* is provided the magnitude of which specifies the importance of the property. The severity can be used to focus effort on the important performance issues during the (manual or automatic) performance tuning process. Performance properties, confidence and severity are defined over performance-related data. “

Hypothesis

Property: Cycles lost due to cache misses

Condition:
PM_MISS_L1_CYC/PM_RUN_CYC>a
Confidence: 1.0
Severity:
PM_MISS_L1_CYC/PM_RUN_CYC

Context

Process 17

Thread 1

Execute

Found property

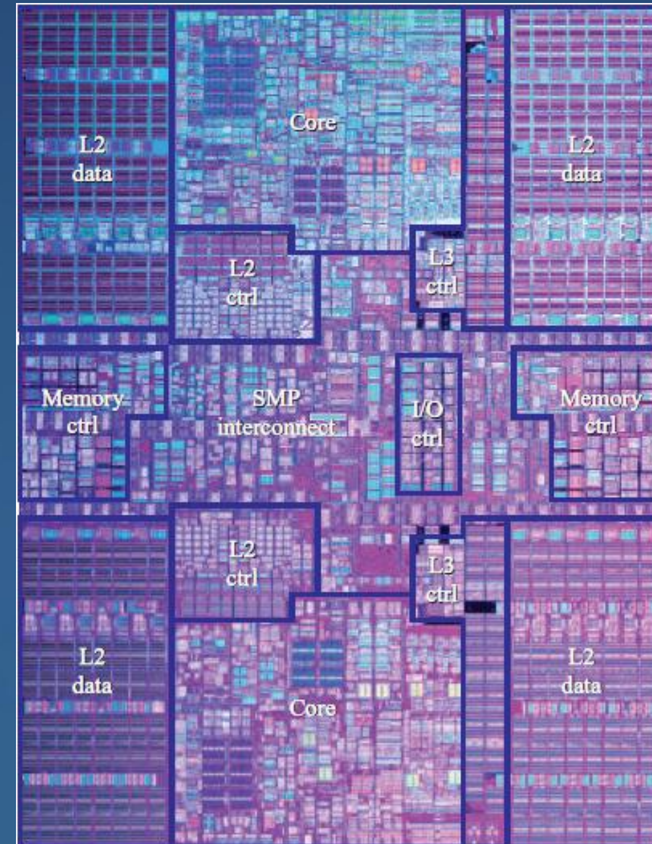
Property: Cycles lost due to cache misses

Condition: *TRUE*
Confidence: 1.0
Severity: 21%

Region 1 | Region 2 | Region 3 | Region 4 | Region 5

POWER 6 Overview

- Ultra-high frequency dual-core chip
  - 8 execution units
    - 2LS, 2FP, 2FX, 1BR, DFU
  - private 64K L1 D and I cache
  - private 4MB on-chip L2
  - On-chip L3 directory and controller
  - Symmetric MultiProcessor (SMP) coherence and data interconnect
  - Simultaneous MultiThreading (SMT)
  - Powerfull prefetch engine
- Performance sacrifices:
  - **In-order** execution model



POWER6 Execution pipeline

P1 P2 P3 P4 IFA IC0 IC1 BHT BHT

Instruction fetch pipeline

IB0 IB1 PD DISP

Instruction dispatch pipeline

RF RF EX RF AG DC0 DC1 FMT

BR/CR

FX

Load

BR/FX/load pipeline

ISS RF

EX1 EX2 EX3 EX4 EX5 EX6 EX7

Floating-point pipeline

ECC ECC

Checkpoint recovery pipeline

- Pre-decode stage
- Instruction fetch/branch stage
- Delayed/transmit stage
- Instruction decode stage
- Instruction dispatch/issue stage
- Operand access/execution stage
- Writeback stage
- Completion stage
- Checkpoint stage
- Cache access stage
- FX result bypass
- Load result bypass
- Float result bypass

AG: address generation; BHT: branch history table; BR: branch; DC: data cache access; DISP: dispatch; ECC: error-correction code; EX: execute; FMT: formatting; IB: instruction buffer; IC0/IC1: instruction cache access; IFA: instruction fetch access; ISS: issue; P1-P4: pre-decode; PD: post decode; RF: register file access

H.Q. Le et al

POWER6 Execution pipeline

Periscope properties:

- *Cycles lost due to no instruction to dispatch*
- *Cycles lost due to branch miss prediction*
- *Cycles lost due to I-Cache miss*
- *Branch miss prediction rate*

P1 P2 P3 P4 IFA IC0 IC1 BHT BHT

Instruction fetch pipeline

IB0 IB1 PD DISP

Instruction dispatch pipeline

EX FX

RF AG DC0 DC1 FMT Load

BR/FX/load pipeline

ISS RF

EX1 EX2 EX3 EX4 EX5 EX6 EX7

Floating-point pipeline

ECC ECC

Checkpoint recovery pipeline

Pre-decode stage; Instruction fetch/branch stage; Delayed/transmit stage; Instruction decode stage; Instruction dispatch/issue stage; Operand access/execution stage; Writeback stage; Completion stage; Checkpoint stage; Cache access stage; FX result bypass; Load result bypass; Float result bypass

AG: address generation; BHT: branch history table; BR: branch; DC: data cache access; DISP: dispatch; ECC: error-correction code; EX: execute; FMT: formatting; IB: instruction buffer; IC0/IC1: instruction cache access; IFA: instruction fetch access; ISS: issue; P1-P4: pre-decode; PD: post decode; RF: register file access

H.Q. Le et al

POWER6 Execution pipeline

Periscope properties:

- *Cycles lost due to dispatch held*
- *Cycles lost due to GPR dependencies*
- *Cycles lost due to FP comparisons*
- *Cycles lost due to other thread active*

P1 P2 P3 P4 IFA IC0 IC1 BHT BHT

Instruction fetch pipeline

IB0 IB1 PD DISP

Instruction dispatch pipeline

RF AG DC0 DC1 FMT Load

BR/FX/load pipeline

ISS RF

EX1 EX2 EX3 EX4 EX5 EX6 EX7

Floating-point pipeline

ECC

ECC

Checkpoint recovery pipeline

Pre-decode stage
Instruction fetch/branch stage
Delayed/transmit stage
Instruction decode stage
Instruction dispatch/issue stage
Operand access/execution stage
Writeback stage
Completion stage
Checkpoint stage
Cache access stage
FX result bypass
Load result bypass
Float result bypass

AG: address generation; BHT: branch history table; BR: branch; DC: data cache access; DISP: dispatch; ECC: error-correction code; EX: execute; FMT: formatting; IB: instruction buffer; IC0/IC1: instruction cache access; IFA: instruction fetch access; ISS: issue; P1-P4: pre-decode; PD: post decode; RF: register file access

H.Q. Le et al

POWER6 Execution pipeline

Periscope properties:
- *Cycles lost due to FX multiply/divide*
- *Cycles lost due to FX multiply dependency on GPR*

Periscope properties:
- *Cycles lost due to FPQ full*
- *FPU issue stall due to FPR dependencies*
- *FPU issue stall due to store*
- *FPU issue out of order*
- *FP divide/sqrt instructions rate*
- *FMA instructions rate*

AG: address gen… cache access; DISP: dispatch; ECC: error-
correction code; E… C1: instruction cache access; IFA: instruction
fetch access; ISS… ter file access

POWER6 Execution pipeline

Periscope properties:

- Load performance
  - *Cycles lost due to demand load cache miss*
  - *Average amount of cycles lost per cache miss*
  - *L1 demand load miss rate*
  - *L2 demand load miss rate*
  - *L3 demand load miss rate*
  - *L2 total load miss rate*
  - *Prefetch for load to L1 rate*
  - *Prefetch for load to L2 rate*
- Store performance
  - Store queue full per store operation
  - *Non-chained stores rate*
- Address translation performance
  - *Cycles lost due to DERAT misses*
  - *DERAT 4K page miss rate*
  - *DERAT 64K page miss rate*

AG: address generati… s; DISP: dispatch; ECC: error-correction code; EX: … n cache access; IFA: instruction fetch access; ISS: iss…

POWER6 Execution pipeline

BR/CR
FX
Load
IFA
BHT
RF
EX
AG
DC0
DC1
FMT
ECC
Instruction dispatch pipeline
Checkpoint recovery pipeline

Periscope property

- *Memory Access Pattern:*
  - *Streaming from memory*
  - *Non-local data access*
  - *Poor cached data reusage*
  - *Non-optimal prefetch engine performance*
  - *Too many consequent stores*

Pre-decode stage
Instruction fetch/branch stage
Delayed/transmit stage
Instruction decode stage
Instruction dispatch/issue stage
Operand access/execution stage
Writeback stage
Completion stage
Checkpoint stage
Cache access stage
FX result bypass
Load result bypass
Float result bypass

AG: address generation; BHT: branch history table; BR: branch; DC: data cache access; DISP: dispatch; ECC: error-correction code; EX: execute; FMT: formatting; IB: instruction buffer; IC0/IC1: instruction cache access; IFA: instruction fetch access; ISS: issue; P1-P4: pre-decode; PD: post decode; RF: register file access

H.Q. Le et al

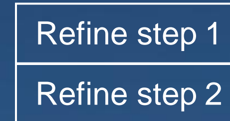
Optimization of hardware counters utilization

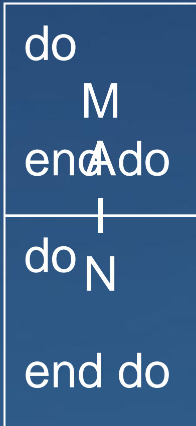
- POWER6 hardware counters:
  - 6 counters available
  - Cycles and instruction completed
  - 4 programmable counters
    - 552 hardware events total
    - grouped into 200 Hardware counters sets
- Total number of events used: 74
  - 43 iterations per region
- Cross-iterations performance deviation
- Countermeasures:
  - Performance properties hierarchy
  - Optimal hardware events grouping
  - Search strategies
  - Phase deviation control

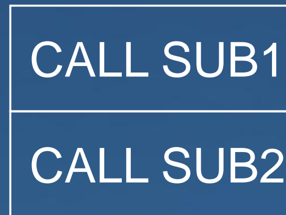
- Total cycles spent
  - *Memory access pattern*
  - *Cycles lost due to demand load miss L1*
    - *Average amount of cycles lost per L1 miss*
    - *L1 demand load miss rate*
      - *Prefetch for load to L1 rate*
    - *L2 demand load miss rate*
      - *Prefetch for load to L2 rate*
    - *L3 demand load miss rate*
    - *L2 total load miss rate*
  - *Cycles lost due to D-ERAT miss*
    - *D-ERAT 4K/64K/16M/16G page miss rates*
  - *Cycles lost due to store queue full*
    - *L2 store miss rate*
    - *Percentage of not chained stores*
  - *Cycles lost due to FPQ full*
    - *FPU issue stall due to FPR dependencies*
    - *FPU issue stall due to store*
    - *FPU issue out of order*
    - *FP divide/sqrt instructions rate*
    - *FMA instructions rate*
  - *Cycles lost due to FX multiply divide*
    - *Cycles lost due to FX multiply dependencies*
  - *Cycles lost due to load-hits-store*
  - *Cycles lost due to no instruction to dispatch*
    - *Cycles lost due to branch miss prediction*
    - *Cycles lost due to I-Cache miss*
    - *Branch miss prediction rate*

…

P6 Breadth First Strategy







Running Periscope on Power575

- Available POWER6 Strategies:
  - POWER6 Depth First "--strategy=P6" – many iterations, minimum memory usage
  - POWER6 Breadth First "--strategy=P6BF" – min iterations, some memory overhead
  - POWER6 Breadth First + Memory Access Pattern Analysis "--strategy=P6BF_Memory"
- Phase performance deviation control:
  - Enabled with "--with-deviation-control"
- Access:

```
% module load periscope
% export MP_HOSTFILE=<hostfile path>
% cp $(PERISCOPE_ROOT)/etc/periscope.sample ~/.periscope
```

- Instrumentation:
  - Substitute FC or CC in makefile by:

```
IFC=psc_instrument -t "all user loop mpi par sync sub call" $(FC)
```

- Run (both interactively and under LoadLeveler):

```
% psc_regsrv &
% psc_frontend --apprun=./stream --sir=stream.sir --mpinumprocs=1
        --strategy=P6BF_Memory --with-deviation-control
```

Experiment 1: COPY and Divide loop size=8M

```
DO j=1,n
        c(j)=a(j)/scalar
END DO
```

Properties found:

- Cycles lost FPQ full **73%**
  - Costly FP div/sqrt rate 100%

Total cycles: 321M

Experiment 2: COPY and FP2INT conv loop

```
DO j=1,n
        c(j)=integers(j)*scalar
END DO
```

Properties found:

•LHS reject per load	99%
•SRQ full per store	5%

Total cycles: 252M

Experiment 3: STREAM COPY loop size=8M

```
DO j=1,n
        c(j)=a(j)
END DO
```

Properties found:

- Cycles lost cache miss 21%
  - Average cyc per miss 270
  - L1 miss rate 1%
  - L2 miss rate 16%
  - L3 miss rate 100%
  - Prefetch to L1 99%
  - Prefetch to L2 99%
- SRQ full per store **40%**

Total cycles: 114M

- **Memory access pattern:** Streaming from memory; Poor cached data reusage; Store queue flooded, too many consequent stores

Experiment 4: LOAD loop size=8M

```
DO j=1,n
        scalar=a(j)
END DO
```

Properties found:

- Cycles lost cache miss **56%**
  - Average cyc per miss 203
  - L1 miss rate 3%
  - L2 miss rate 6.5%
  - L3 miss rate 100%
  - Prefetch to L1 99%
  - Prefetch to L2 99%

Total cycles: 84M

Experiment 5: non-local COPY loop size=8M

```
DO j=1,n
    DO i=0,m-1
        c(i*n+i)=a(i*n+j)
    END DO
END DO
```

Properties found:

Cycles lost cache miss	12**%**		Total cycles: 801M
•Average cyc per miss	22		
•L1 miss rate	57**%**		
•L2 miss rate	0.04%		
•L3 miss rate	55%		
•Prefetch to L1		0.02%	
•Prefetch to L2		**1.597%**	
•Cycles lost DERAT miss	**68%**		
•64K DERAT page miss		23%	

•**Memory access pattern:** Severe non-local data access; L1 miss rate is too high; Non-optimal prefetch engine performance; Store queue flooded, too many consequent stores; L2 dominate data access

Experiment 6: GENE memory intensive subroutine

```
do n=ln1,ln2
       do m=lm1,lm2
          do l=ll1,ll2
             f_s(li0*lj0*nzb:li0*lj0*(lk0+nzb)-1,l,m,n) =&
                  &g_s(:,l,m,n) + pre_s(:,l,m,n)* &
          &psi_s(li0*lj0*nzb:li0*lj0*(lk0+nzb)-1)
          End Do
          f_s(:,lbv:ll1-1,m,n)=0.
          f_s(:,ll2+1:ubv,m,n)=0.
       enddo
enddo
```

Properties found:

- Cycles lost cache miss 8.5% ← Memory access problem...
  - Average cyc per miss 250 ← beyond L2 latencies
  - L1 miss rate 0.26%
  - L2 miss rate 24%
  - L3 miss rate 95% ← streaming from memory
  - Prefetch to L1 118% ← slightly non-local access
  - Prefetch to L2 118%
- SRQ full per store **49%** ← **Bottleneck**
- Cycles lost FPQ 17%
- Cyc lost DERAT 3%
- **Memory access pattern:** Streaming from memory; Poor cached data reusage; Non-optimal prefetch engine performance; Store queue flooded, too many consequent stores

Conclusion

- Single core performance analysis is much easier now with Periscope!
  - More then 30 performance properties cover the majority of potential bottlenecks
  - Automatically searches for inefficiencies through the whole code, pointing to important ones
  - Hides hardware details from user
  - Automatic search for memory access patterns
- Also MPI ill communication pattern and OpenMP overheads analysis available
- Performance Analysis smoothly integrated into development environment
- Easy to run both interactively and under LoadLeveller

- In case found properties are not correct, please, send us properties.psc, appl.sir + source or short description of what the code is doing.

Your feed back will help us to implement new properties for you!

Thank you for your attention!
Questions?

Yury Oleynik <oleynik@.in.tum.de>