



Reading and Interpreting Stall Counters

Murali Vijayasundaram
Sujoy Saraswati

© 2004 Hewlett-Packard Development Company, L.P.
The information contained herein is subject to change without notice

Agenda

- Cycles accounting
- Collecting stall breakdown
- For different stall types
 - What causes it?
 - How to gather more detail?
 - How to reduce it?
- Summary
- Questions?

Cycle accounting

- Exception/Branch mispredict flush - BE_FLUSH_BUBBLE
- Data/FPU access - BE_L1D_FPU_BUBBLE
- Execution Latency - BE_EXE_BUBBLE
- RSE spill/fill - BE_RSE_BUBBLE
- Front end stalls - BE_BUBBLE.FE
- Unstalled

- Front end stalls
 - instruction TLB miss stalls
 - BE_LOST_BW_DUE_TO_FE.TLBMISS
 - instruction cache miss stalls
 - BE_LOST_BW_DUE_TO_FE.IMISS

Collecting stall breakdown

- Lots of PMU events
 - requires multiple measurement runs
 - tools should hide the complexity
- How caliper collects stall metrics
caliper cpu -m stall ...
 - Collects
 - stall data in one measurement run
 - uses multiplexing supported by perfmon 2.2 or later
 - uses 7 sets in Itanium2 and 2 sets in Montecito
 - Reports
 - summary data for the entire program run
 - sample detail for each sec of program execution

caliper cpu -m stall ...

-----FE Components-----

Stats	RAW	-----laccess-----		
	CPI	Itlb	Icache	Branch

MEAN	1.102	0.23%	4.17%	1.87%
------	-------	-------	-------	-------

-----BE Components-----

Unstall	BE	Score	-----Daccess-----			RSE
Execute	Flush	Board	L1Dtlb	L2Dtlb	Dcache	Active

47.79%	4.55%	1.00%	0.50%	0.53%	38.05%	1.29%
--------	-------	-------	-------	-------	--------	-------

BE Flush stalls

- Branch misprediction and exception flush
- collect branch prediction profile
 - caliper branch
 - Reducing branch misprediction flushes
 - Use Profile Guided Optimizations (PGO) and higher optimization levels
- collect traps/interrupts/faults profile
 - caliper traps ...
 - Reducing exception flush
 - Modify source based on trap type

Scoreboard stalls

- Register dependency stalls & FPU SIR stalls
- Usually caused by long latency instructions
 - Floating point and multimedia
 - Compiler can not hide all long instruction latencies
- Reducing scoreboard stalls
 - Use PGO and higher Compiler optimizations levels
 - Intersperse long latency operations (e.g., multimedia, fsqrt) with other computations

Scoreboard stalls – FPU SIR stalls

- SIR hardware in CPU is used to report precise exceptions
 - inspects arguments to FP operations
 - stalls the main pipeline if it determines that an exception might occur
- Collect FPU SIR stalls using `caliper cpu -m fp ...`
 - Reports SIR false and SIR true stalls
 - SIR false - only stalls and no trap taken
 - SIR true - stalls and trap taken
- Reducing FPU stalls
 - Use PGO and higher optimization levels
 - Compile with flush to zero option (-ftz in the Intel compiler)
 - Use wider floating point types
 - Change algorithms to avoid extremely larger/small numbers

Data access stalls – dcache

- Cycles stalled due to data cache misses
- May count FP scoreboard stalls as data access
 - Can not breakup FRALL into FRFR and FRLOAD
- Compiler can not “cover” all load latencies
- Explore dcache stalls using:
 - caliper dcache ...
 - Provides source line and instruction level dmiss data
 - Bucketizes access latencies
 - L2, L3, Cell local, C2C, Memory etc.,

Data access stalls – L1 dtlb, L2dtlb

- L1 dtlb stalls
 - stalls due to L1 DTLB miss that hits the L2 DTLB
- L2dtlb stalls
 - stalls due to L2 DTLB miss while HPW is running
 - time spend in the trap handler is not counted here
- Explore DTLB misses using:
caliper dtlb
 - Provides source line and instruction level DTLB miss data

Reducing data access stalls

- Use PGO and higher optimization levels
- Manually insert prefetches to bring data into cache
- Design/develop to:
 - Work on smaller number of data items at a time
 - Work on chunks of data that fit in cache-line/cache
 - Access data items sequentially
 - Avoid mixing integer & FP data within same cache line
 - Put hot data in the same cache line

RSE Active – RSE spill/fill stalls

- Stalls due to RSE spilling/filling registers to/from memory
- Insufficient free registers to satisfy alloc or br.ret
 - Due to: deep call stack, functions with heavy register needs, recursive functions
- Reducing RSE stalls
 - Simplify algorithms to use fewer general registers
 - Reduce number of arguments in heavily used function chains
 - Decompose large complex functions into small leaf functions
 - Eliminate or avoid recursion
 - Use PGO and higher optimization levels

Instruction access stalls – icache, itlb

- Front end stalled due to:
 - instruction cache miss or Instruction TLB miss
 - no backend stalls/flushes and instruction buffer is empty
- Explore instruction access stalls using:
 - caliper icache ...
 - caliper itlb ...
- Reducing instruction access stalls
 - Use PGO and higher optimization levels

Summary

- Collect stall breakdown using:
caliper cpu -m stall ...
- Analyze stalls using:
dcache, dtlb, branch, icache, itlb, traps
- Reduce stalls using:
 - Compiler PGO and higher optimizations
 - Source modifications

Resources

- <http://hp.com/go/caliper>
 - Caliper external website
 - Downloads, documentation, tech notes
- <http://devresource.hp.com>
 - Developers resources website
 - Training webinars
- Intel PMU document (Montecito)
 - <http://www.intel.com/design/itanium2/manuals/308065.htm>