Hardware Support for Out-of-Order Instruction Profiling on Alpha 21264a

Lance Berc & Mark Vandevoorde

Joint work with: Jennifer Anderson, Jeff Dean, Sanjay Ghemawat, Shun-Tak Leung, Mitch Litchenberg, Gerard Vernes, Carl Waldspurger, William E. Weihl, and Jonathan White
dcpi@pa.dec.com
http://www.research.digital.com/SRC/dcpi/
Compaq Systems Research Center
Palo Alto, CA 94301 USA

www.compaq.com

COMPAQ
Better answers

Related Trends

- **More aggressive CPUs**
  - More performance puzzles
- **Widening gap in peak-vs-actual performance**
- **More sophisticated performance tools**
  - Digital Continuous Profiling Infrastructure (DCPI)
    - Where CPU cycles went
    - Where peak performance was lost and why
  - Others: Morph, SGI Speedshop, VTune

COMPAQ
Better answers

Detailed Information Matters

DCPI experience on the Alpha

- TPC-D: 10% speedup

- Duplicate filtering for AltaVista: part of 19X

- Compress program: 22%

- Compiler improvements: 20% in several SPEC benchmarks

All required instruction-level information

COMPAQ
Better answers

Traditional Performance Counters

- Count events, interrupt when counter rolls over
  - cycles, issues, loads, L1 Dcache misses, branch mispredicts, uops retired, ...
    - Alpha 21064, 21164; Ppro, PII; R10k, …
    - Easy to measure total cycles, issues, CPI, etc.
- Basic information is restart PC

COMPAQ
Better answers

DCPI on In-Order Alpha 21164

Address	Instruction	CPI
9618	addq s0,t6,t6	1.0 cycles

b *(b = data dep on t6)*

D   **3.5 cycles** *(D = DTLB miss)*

⋮

D

| 961c | ldl   t4,0(t6) | 3.5 cycles |

a

a   *(a = data dep on t4)*

a   **21.0 cycles**

di   *(d = d-cache miss)*

⋮   *(i = i-cache miss)*

di

| 9620 | xor   t4,t12,t5 | 21.0 cycles |

Where peak performance is lost and **why**

COMPAQ
Better answers

Problem: Inaccurate Attribution

- **Experiment**
  - count data loads
  - loop: single load + hundreds of nops
- **In-Order Processor**
  - Alpha 21164
  - skew
  - one large peak

Skew

782

Histogram of Restart PCs

load

COMPAQ
Better answers
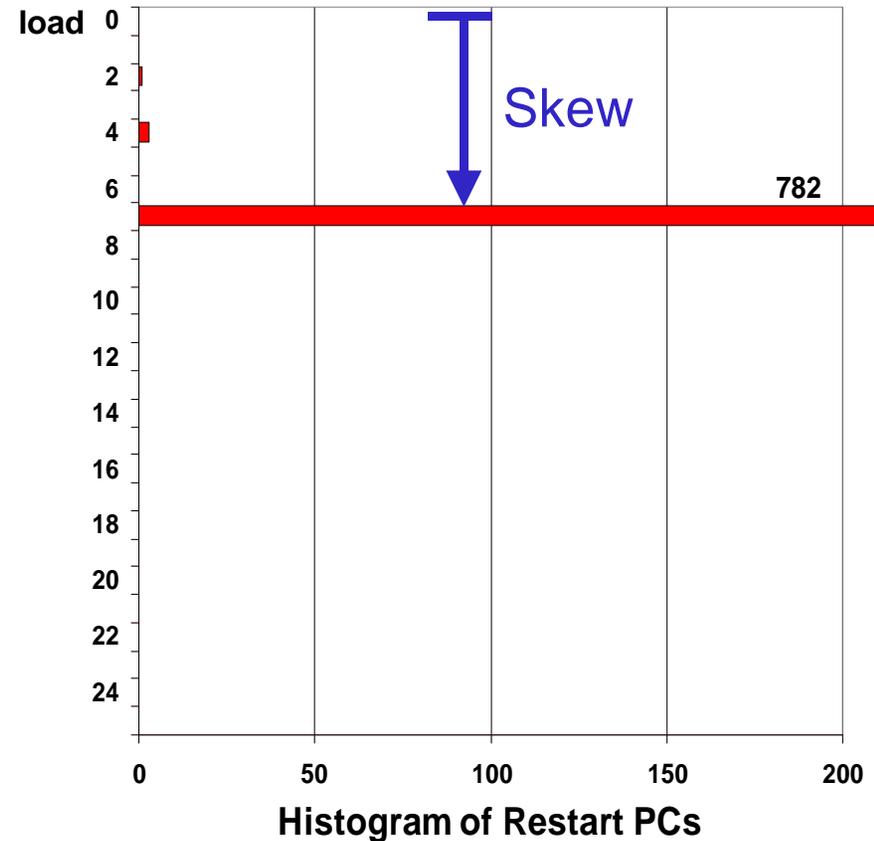
Problem: Inaccurate Attribution

- Experiment
  - count data loads
  - loop: single load + hundreds of nops
- In-Order Processor
  - Alpha 21164
  - skew
  - one large peak
- Out-of-Order Processor
  - Intel Pentium Pro
  - skew
  - smear
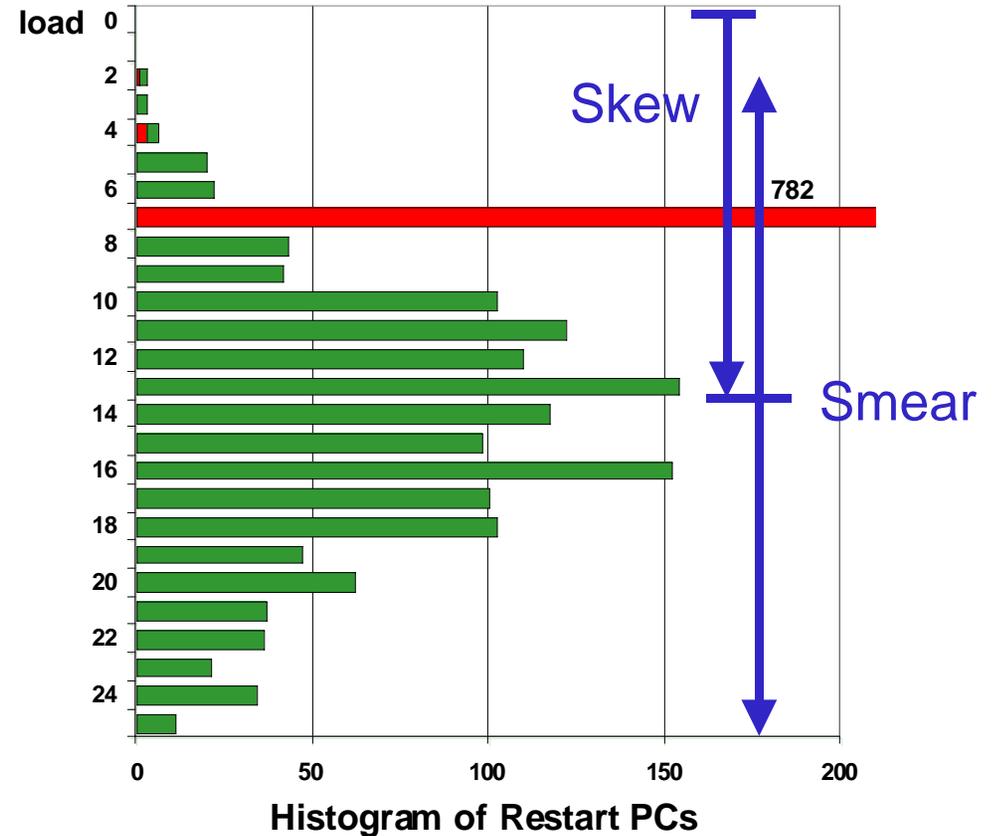
Histogram of Restart PCs

Impact of Misattribution

- No skew or smear
  - Instruction-level analysis is easy!
- If skew is a constant number of cycles
  - Instruction-level analysis may be possible: DCPI
  - Adjust sampling period by amount of skew
  - Infer execution counts, CPI, stalls, and stall explanations from cycles samples and program
- Smear
  - Instruction-level analysis seems hopeless
  - Examples: PII, StrongARM

COMPAQ

Better answers

ProfileMe on Alpha 21264a

- Count fetched instructions instead of events
- Save PC of sampled instruction
    - Interrupt handler reads Internal Processor Register
    - Makes skew and smear irrelevant
- Save execution information in IPRs

COMPAQ
Better answers

ProfileMe: Instruction Sampling

Fetch counter

overflow?

random selection

ProfileMe tag!

fetch

map

issue

exec

retire

branch predictor

icache

interrupt

tagged?

capture!

| pc | miss? | prediction | | trapped? | trap type | | retired? | delay |

internal processor registers
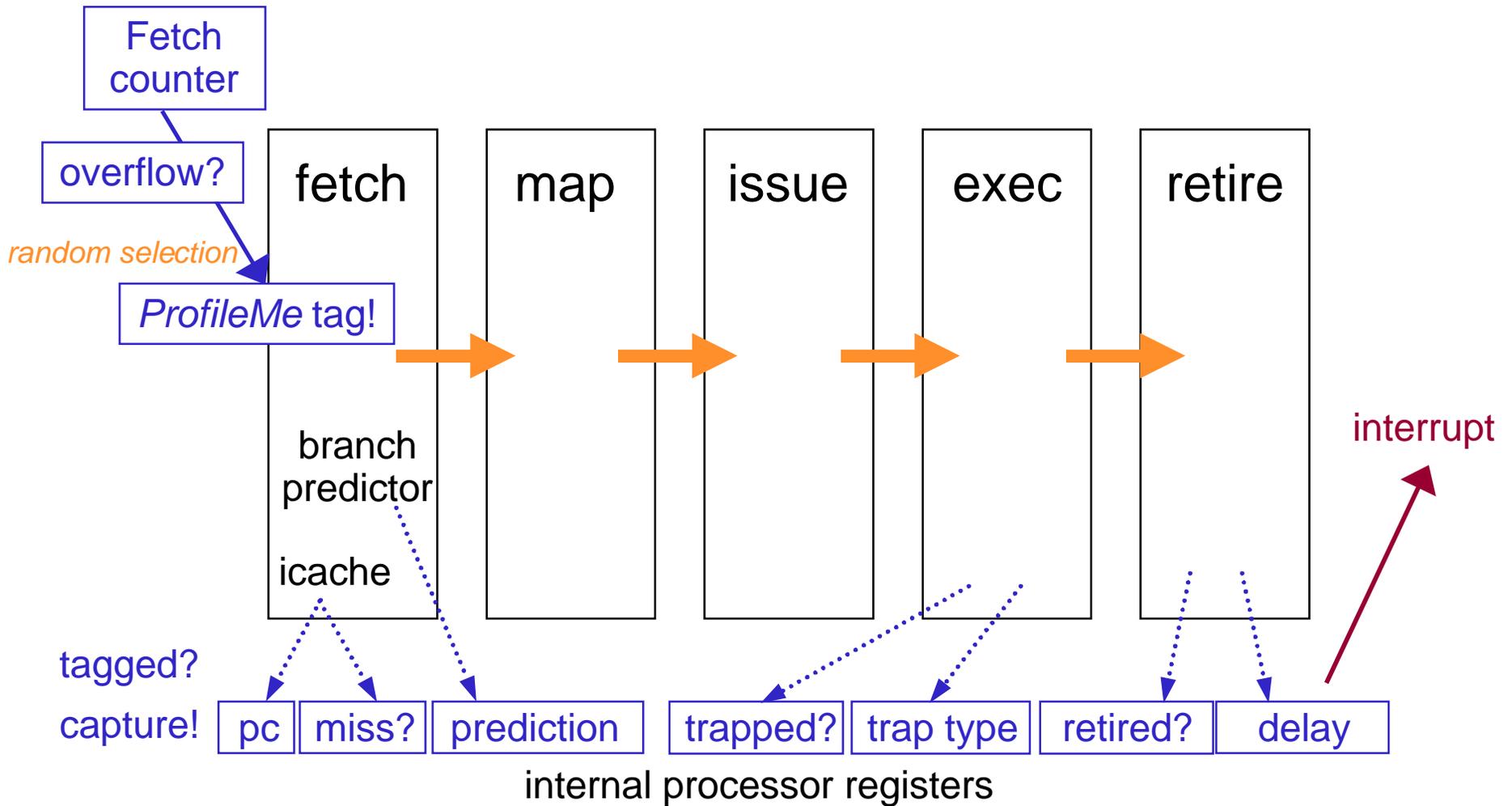
COMPAQ
Better answers

Instruction-Level Information

- PC + Retire Status ⇨ execution frequency
- PC + Cache Miss Flag ⇨ cache miss rates
- PC + Mispredict ⇨ mispredict rates
- PC + Event Flag ⇨ event rates
- PC + Retire Delay ⇨ ILP

COMPAQ
Better answers

Pointer-Chasing Loops

```
while (p) p = *p;
```

```
retires    delay  misp    PC
   8537     1.99          0x30  ldq  a0, 0(a0) # L1 hit
   8661     1.01  .006 0x34  bne  a0, 0x30

    610     2.00          0x30  ldq  a0, 0(a0) # L2 hit
    605    23.06  .007 0x34  bne  a0, 0x30

    241     1.99          0x30  ldq  a0, 0(a0) # Main Mem
    205   138.13  .005 0x34  bne  a0, 0x30
```

```
retires   delay      PC
   2563              0xed0  lds   $f14, 13052(a0)
   2515              0xed4  addq  t11, s4, t11
   2519              0xed8  adds  $f1,$f15,$f1
   2577    1.0       0xedc  muls  $f17,$f3,$f17
   2485              0xee0  ldq_u zero, 0(sp)
   2490    2.0       0xee4  subs  $f19,$f20,$f20
   2525              0xee8  muls  $f18,$f4,$f18
   2546              0xeec  muls  $f27,$f4,$f3
   2546              0xef0  adds  $f0,$f10,$f0
   2501              0xef4  lds   $f15, 13072(a0)
   2498              0xef8  lds   $f10, 13060(a0)
   2521              0xefc  muls  $f13,$f4,$f13
   2456              0xf00  adds  $f16,$f11,$f16
   2440              0xf04  adds  $f17,$f12,$f17
   2579              0xf08  ldq_u zero, 0(sp)
   2502    1.0       0xf0c  lds   $f12, 13068(a0)

        16 instructions in 4 cycles
```

Poor ILP in OpenGL

```
retires  delay          PC
    2262                0x210   addq      t5, 0x10, t5
*   2308                0x214   cmptlt    $f7,$f3,$f10
    2231                0x218   subq      t4, 0x1, t4
*   2285     1.0        0x21c   cmptlt    $f4,$f7,$f11
*   2224     0.9        0x220   cmptlt    $f8,$f5,$f12
*   2227     1.0        0x224   cmptlt    $f6,$f8,$f13
*   2257     1.0        0x228   cmptlt    $f9,$f1,$f14
*   2390     1.0        0x22c   cmptlt    $f2,$f9,$f15
    2265                0x230   lds       $f7, 0(t5)
*   2343     1.0        0x234   adds      $f10,$f16,$f16
    2357                0x238   lds       $f8, 4(t5)
*   2249     1.0        0x23c   adds      $f11,$f17,$f17
    2309     0.1        0x240   lds       $f9, 8(t5)
*   2214     1.0        0x244   adds      $f12,$f18,$f18
*   2292     1.0        0x248   adds      $f13,$f19,$f19
*   2234     1.0        0x24c   adds      $f14,$f20,$f20
*   2282     1.0        0x250   adds      $f15,$f21,$f21
    2278     1.0        0x254   bgt       t4, 0x210
```

12 Floating point adds in 12 cycles

Eight Queens Code

```
while ((! *q) && (j != 8)) {
    j = j + 1; *q = false;
    if (b[j] && a[i+j] && c[i-j+7]) {…}
```

retires	delay	mispred	pc
3807			20 cmpeq s6, 0x8, v0
3858			24 lda a1, 4(a1)
3922			28 lda a0, 4(a0)
3909	0.28		2c lda a3, -4(a3)
3757			30 bis a2, v0, v0
3919			34 addl s6, 0x1, s6
3952	1.01	0.13	38 bne v0, f0
3543	0.31		3c stl zero, 0(s0)
3457	0.42		40 ldl v0, 0(a1)
3573	0.25		44 ldl a2, 0(a0)
3547	0.39		48 ldl t0, 0(a3)
3509	0.70		4c cmoveq a2, zero, v0
3573			50 bis zero, zero, a2
3392	2.06		54 cmoveq t0, zero, v0
3543	2.00	0.14	58 beq v0, 20

	7.42 cycles		

Optimized Eight Queens Code

```
retires delay   pc
   3910          20 cmpeq    s6, 0x8, v0
   3999          24 lda      a1, 4(a1)
   3998          28 lda      a0, 4(a0)
   3883   0.26   2c lda      a3, -4(a3)
   3873          30 bis      a2, v0, v0
   3926          34 addl     s6, 0x1, s6
   3869   1.06   38 bne      v0, f0
   3476   0.52   3c stl      zero, 0(s0)
   3481   0.52   40 ldl      v0, 0(a1)
   3413   0.26   44 ldl      a2, 0(a0)
   3439   0.39   48 ldl      t0, 0(a3)
   3448          4c and      a2, v0, v0   # was cmov
   3415          50 bis      zero, zero, a2
   3561   0.45   54 and      t0, v0, v0   # was cmov
   3518   2.16   58 beq      v0, 20
          ----
          5.62 cycles
```

Profiling Aborted Instructions

retires	aborts	traps	ldstore	PC		
24	0	0	0	0xca8c	stt	$f16, 56(sp)
36	0	0	0	0xca90	addssu	$f25,$f28,$f16
27	0	0	0	0xca94	ldl	at, 8(a2)
26	0	0	0	0xca98	cmpule	at, 0xc, at
29	0	0	0	0xca9c	bne	at, 0xdcb0
21	0	0	0	0xcaa0	stt	$f20, 72(sp)
17	23	23	23	0xcaa4	ldt	$f20, 56(sp)
29	28	0	0	0xcaa8	sts	$f20, 60(a2)
39	20	0	0	0xcaac	ldl	at, 8(a2)
21	21	0	0	0xcab0	cmpule	at, 0x4, at
21	26	0	0	0xcab4	bne	at, 0xdcb4
20	29	8	7	0xcab8	ldt	$f20, 72(sp)
31	37	0	0	0xcabc	sts	$f20, 28(a2)
28	33	0	0	0xcac0	ldl	at, 8(a2)

Trap shadow is 70 instructions long!
Fixed with better register allocation

Avoiding Traps

- Traps (pipeline aborts) are expensive
  - Mispredicts, DTB misses, replays, etc.
- Profiles identify trapping PCs and trap type
  - Aborted instructions are profiled, too
  - But how many cycles were lost?
  - Can analysis deduce missing information?

COMPAQ
Better answers

Status

- ProfileMe works on Alpha 21264a prototypes
- Initial experience
  - 17% speedup for a Java Virtual Machine
- Still working on enhancements
  - Analysis software for 21264a
  - Counters on future Alphas

COMPAQ
Better answers

Summary

- Instruction-level information improves program optimizations
- ProfileMe: Sample instructions rather than events
  - Event sampling inaccurate on O-O-O machines
  - Direct observation eliminates misattribution
- ProfileMe first implemented in Alpha 21264a
  - Has led to significant insight and improvements

COMPAQ
Better answers

COMPAQ

Better answers

www.compaq.com