Visualizing the out-of-order CPU model

Ryota Shioya Nagoya University

Introduction

This presentation introduces:

 \bigcirc the visualization of the out-of-order CPU model in gem5

🖉 Konata	- • ×
File Window View Help	
trace-all-fastfwd.out 🗙	
46: s47 (t0: r37): 0x004063f0: ldr 1 Is Cm 1 2 Mc 1	
47: s48 (t0: r38): 0x004063f4: cbz Rn 1 Ds 1 Is Cm 1	
48: s49 (t0: r39): 0x004063f8: ldr Rn 1 Is Cm 1 Mc 1	
49: s50 (t0: r40): 0x004063fc: ldr Dc Rn 1 Is Cm 1 Mc 1	
50: s51 (t0: r41): 0x00406400: orr F Dc Rn 1 Is Cm 1	
51: s52 (t0: r42): 0x00406404: ldr F Dc Rn 1 Is Cm 1 Mc 1	
52: s53 (t0: r43): 0x00406408: ldr F Dc Rn 1 Is Cm 1 Mc	
53: s54 (t0: r44): 0x0040640c: ldr F Dc Rn 1 Is Cm 1 2	
54: s55 (t0: r45): 0x00406410: add F Dc Rn 1 Ds 1 2	● Konta – □ ×
55: s56 (t0: r46): 0x00406414: str F Dc Rn 1 Ds 1 2	File Window View Help
56: s57 (t0: r47): 0x00406414: addxi_ F Dc Rn 1 Is Cm	trace-all-fastfwd.out 🗙
57: s58 (t0: r48): 0x00406418: ubfm F Dc Rn 1 Ds 1	
58: s59 (t0: r49): 0x0040641c: str F Dc Rn 1 Ds	
59: s60 (t0: r50): 0x00406420: subs F Dc Rn 1 Is	
60: s61 (t0: r51): 0x00406424: add F Dc Rn 1	
61: s62 (t0: r52): 0x00406428: b.ls F Dc Rn 1	
62: s63 (t0: r53): 0x0040642c: subs	
63: s64 (t0: r54): 0x00406430: csel F Dc Rn	
64: s65 (t0: r55): 0x00406434: str	
65: s66 (t0: r56): 0x00406438: movz	
66: s67 (t0: r57): 0x0040643c: str	
67: s68 (t0: r58): 0x00406440: orr 68: s69 (t0: r59): 0x00406444: ret	
69: s70 (t0: r60): 0x00406568: adds	
70: s71 (t0: r61): 0x00406566: b.ne	
71: s72 (t0: r62): 0x00406570: movz	
72: s73 (t0: r63): 0x00406574: addxi_	
73: s74 (t0: r64): 0x00406574: ldp_uo	
74: s75 (t0: r65): 0x00406578: ldp_uo	
75: s76 (t0: r66): 0x00406578: addxi_	[957, 333]
76: s77 (t0: r67): 0x0040657c: ret	

Introduction

Let's suppose

- \diamond you come up with an excellent idea and
- \diamond try to extend the CPU model in gem5 for adding your new method.
- You will probably tackle the following issues:
 - ◇ difficult bugs, especially performance related ones
 - a situation where your method cannot improve the performance as expected

• You probably validate your *modified* gem5 as follows:

- Check counters outputted by gem5
 - □ e.g., the number of LLC misses / branch mispredictions
 - □ These counters sometimes give us clues.
- \diamond Check the behavior by using a debugger and step execution
- However, it is difficult to fix issues in the following situations:
 - \diamond You have no idea what causes it
 - You recognize some counters show that something is wrong, but you have no idea what happened

Visualizing the pipeline behavior

- In such situations, pipeline visualization is very useful.
- In general, visualization is a powerful tool for investigating bugs or behavior.
 - If you have developed hardware with HDL such as Verilog, you may have used a waveform viewer.
 - In a waveform view, you can easily see signal transitions and relations between signals.
 - □ Such viewers may have helped you a lot.
- This is also true for gem5!

A text-based pipeline viewer is provided for gem5

This viewer is very useful to investigate the pipeline behavior.

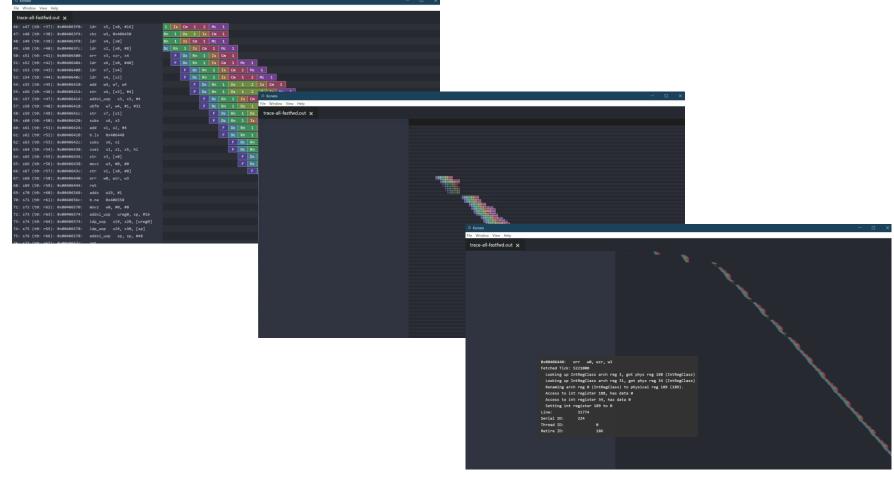
 \diamond But, you can see only a limited range of instruction sequences at once \diamond This is the "less" command itself, it is not very user-friendly.

🖲 🔿 👘 Thanks for flying Vim — le	ss — 162×44	
<mark>n. [c</mark>]-(480000) 0x120007bf4.0 ldq r2,0(r16) [328]
in <mark>p. c n</mark> .	480000) 0x120007bf8.0 cmpeq r2,5,r1 [329]
<mark>/n.p]e.n</mark>	480000)0x120007bfc.0 bne r1,0x120007c4c [330]
- <mark>fdn. 1<mark>2 n</mark> </mark>	520000)0x120007c00.0 cmple r2,5,r1 [331]
<mark>fdn.p<mark>ll</mark>e.n</mark>	520000) 0x120007c04.0 beq r1,0x120007c14 [332]
<mark>fdn.lc</mark>	520000) 0x120007c08.0 cmpeq r2,3,r1 [333]
<mark>f dn . p l<mark>l c . n</mark> </mark>	520000) 0x120007c0c.0 bne r1,0x120007c40 [334]
	520000) 0x120007c40.0 ldq r1,8(r16) [349]
	520000)0x120007c44.0 stq r1,0(r4) [350]
	520000)0x120007c48.0 br 0x120007c54 [351]
	520000) 0x120007c54.0 lda r16,16(r16) [358]
	520000) 0x120007c58.0 ldq r1,0(r16)	359]
[dn.p]-(520000) 0x120007c5c.0 bne r1,0x120007bf4 [360]
	520000)0x120007bf4.0 ldq r2,0(r16) [377]
	520000) 0×120007bf8.0 cmpeq r2,5,r1 [378]
[dn.p]-(520000) 0x120007bfc.0 bne r1,0x120007c4c [379]
	520000) 0x120007c4c.0 ldq r1,8(r16) [399]
	520000)0x120007c50.0 stq r1,0(r3) [400]
	520000) 0x120007c54.0 lda r16,16(r16) [401]
f <mark>dn.p∥₂</mark>	520000) 0x120007c58.0 ldq r1,0(r16)	402]
l=-(520000) 0x120007c5c.0 bne r1,0x120007bf4 [403]
	520000) 0x120007bf4.0 ldq r2,0(r16) [404]
i <mark>dn.pl</mark> c.n.	520000) 0x120007bf8.0 cmpeq r2,5,r1 [405]
	520000) 0x120007bfc.0 bne r1,0x120007c4c [406]
i <mark>dn. <mark>cn</mark></mark>	520000) 0x120007c00.0 cmple r2,5,r1 [407]
<u>[]_(</u>	520000) 0x120007c04.0 beq r1,0x120007c14 [408]
	520000) 0x120007c14.0 cmpeq r2,6,r1 [409]
[dn.p][c.f.	520000) 0x120007c18.0 bne r1,0x120007c28	410]
	520000) 0x120007c1c.0 cmpeq r2,17,r1 [426]
[-(520000) 0x120007c20.0 bne r1,0x120007c34 [427]
n	520000) 0x120007c24.0 br 0x120007c54 [443]
l <mark>e le le constant de la const</mark>	520000) 0x120007c54.0 lda r16,16(r16) [444]
P <mark>∎e-i</mark> CC	520000) 0x120007c58.0 ldq r1,0(r16) [445]
•••••]••••	520000) 0x120007c5c.0 bne r1,0x120007bf4 [446]
international statements in the statement of the statemen	560000) 0x120007bf4.0 ldq r2,0(r16) [463]
······································	560000) 0x120007bf8.0 cmpeq r2,5,r1 [464]
[]-(560000) 0x120007bfc.0 bne r1,0x120007c4c [465]
i-c	560000) 0x120007c00.0 cmple r2,5,r1 [478]
i(<u>idu-bic-u</u>	560000) 0x120007c04.0 beq r1,0x120007c14 [479]
······································	560000) 0x120007c14.0 cmpeq r2,6,r1 [480]
i-(560000) 0x120007c18.0 bne r1,0x120007c28 [481]
	560000) 0x120007c1c.0 cmpeq r2,17,r1 [482]
]-(560000) 0x120007c20.0 bne r1,0x120007c34 [483]

Konata: a new GUI based viewer

You can see the pipeline behavior as a map app.

 \diamond This presentation introduces Konata and best practices in gem5.



- 1. A brief explanation of how to use
- 2. Typical visualization examples
- 3. Use cases

- 1. Install: All you have to do is to download the package and unpack it.
 - https://github.com/shioyadan/Konata/releases
 - ◇ Windows/Linux/Mac packages are provided.
 - \diamond No additional runtime is not required.
- 2. Start the executable file such as Konata.exe

- 1. Generate a trace log from gem5 with the O3 CPU model
 - \bigcirc Execute gem5 with the following flags
 - ./build/ARM/gem5.opt --debug-flags=03PipeView --debug-start=<first
 tick of interest> --debug-file=trace.out configs/example/se.py
 --cpu-type=detailed --caches -c <path to binary> -m <last cycle of
 interest>
 - This example is from http://www.m5sim.org/Visualization
- 2. Load the generated "trace.out" to Konata
 - ◇ from the menu in the window or using drag&drop

How to use

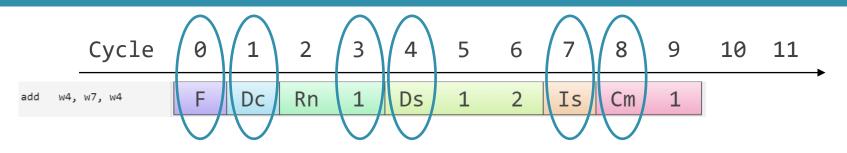
After loading the file, contents like the following are shown.

◇ Left side: instruction information such as a PC and mnemonic

◇ Right side: the image of visualized pipeline behavior

🕸 Konata	
File Window View Help	
trace-all-fastfwd.out 🗙	
47: s48 (t0: r38): 0x004063†4: cbz w3, 0x406450	Dc Rn 1 Ds 1 Is Cm 1
48: s49 (t0: r39): 0x004063f8: ldr x4, [x0]	Dc Rn 1 Is Cm 1 Mc 1
49: s50 (t0: r40): 0x004063fc: ldr x2, [x0, #8]	F Dc Rn 1 Is Cm 1 Mc 1
50: s51 (t0: r41): 0x00406400: orr x3, xzr, x4	F DC Rn 1 Is Cm 1
51: s52 (t0: r42): 0x00406404: ldr x6, [x0, #40]	F DC Rn 1 IS Cm 1 Mc 1
52: s53 (t0: r43): 0x00406408: ldr x7, [x4]	F Dc Rn 1 Is Cm 1 Mc 1
53: s54 (t0: r44): 0x0040640c: ldr x4, [x2]	F Dc Rn 1 Is Cm 1 2 Mc 1
54: s55 (t0: r45): 0x00406410: add w4, w7, w4	F Dc Rn 1 Ds 1 2 Is Cm 1
55: s56 (t0: r46): 0x00406414: str x4, [x3], #4]	F Dc Rn 1 Ds 1 2 3 Is Mc 1
56: s57 (t0: r47): 0x00406414: addxi_uop x3, x3, #4	F Dc Rn 1 Is Cm 1 2 3 4
57: s58 (t0: r48): 0x00406418: ubfm w7, w4, #1, #31	F Dc Rn 1 Ds 1 2 Is Cm 1
58: s59 (t0: r49): 0x0040641c: str x7, [x1]	F Dc Rn 1 Ds 1 2 Is Mc 1
59: s60 (t0: r50): 0x00406420: subs x6, x3	F Dc Rn 1 Is Cm 1 2 3 4
60: s61 (t0: r51): 0x00406424: add x1, x2, #4	F Dc Rn 1 Is Cm 1 2 3
61: s62 (t0: r52): 0x00406428: b.ls 0x406448	F Dc Rn 1 Is Cm 1 2 3
62: s63 (t0: r53): 0x0040642c: subs x6, x1	F DC Rn 1 Is Cm 1 2
63: s64 (t0: r54): 0x00406430: csel x1, x1, x5, hi	F DC Rn 1 DS IS Cm 1
64: s65 (t0: r55): 0x00406434: str x3, [x0]	F Dc Rn 1 Is Mc 1
65: s66 (t0: r56): 0x00406438: movz w3, #0, #0	F Dc Rn 1 Is Cm 1
66: s67 (t0: r57): 0x0040643c: str x1, [x0, #8]	F Dc Rn 1 Is Mc 1
67: s68 (t0: r58): 0x00406440: orr w0, wzr, w3	F DC Rn 1 Is Cm 1
68: s69 (t0: r59): 0x00406444: ret	F DC Rn 1 Is Cm 1
69: s70 (t0: r60): 0x00406568: adds w19, #1	F Dc Rn 1 Is Cm 1 2
70: s71 (t0: r61): 0x0040656c: b.ne 0x406558	F Dc Rn 1 Ds Is Cm 1
71: s72 (t0: r62): 0x00406570: movz w0, #0, #0	F Dc Rn 1 Is Cm
72: s73 (t0: r63): 0x00406574: addxi_uop ureg0, sp, #16	F Dc Rn 1 Is Cm
73: s74 (t0: r64): 0x00406574: ldp_uop x19, x20, [ureg0]	F DC Rn 1 IS Cm
74: s75 (t0: r65): 0x00406578: ldp_uop x29, x30, [sp]	F Dc Rn 1 Is Cm
75: s76 (t0: r66): 0x00406578: addxi_uop sp, sp, #48	F Dc Rn 1 Is
76: s77 (t0: r67): 0x0040657c: ret	F Dc Rn 1 Ds

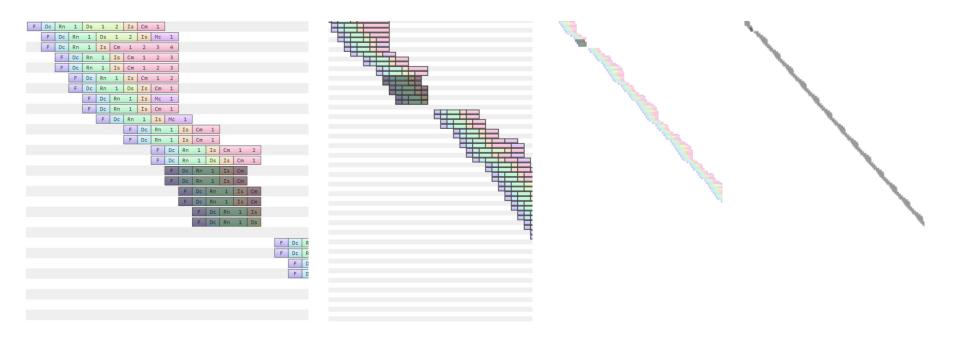
How to see the visualized image



- The clock cycle proceeds from left to right
 - ◇ F : Instruction fetch
 - ◇ Dc : Instruction Decode
 - 🔷 Rn : Rename
 - \bigcirc Ds : Dispatch
 - IS : Issue
 - \diamond Cm : Completion of execution
 - □ The execution stage is not explicitly shown
 - The execution seems to be started at the first stage of "Cm" stages
 - \diamond (The end of Cm stages) : Retire

Zoom in/out

You can zoom in/out as follows:

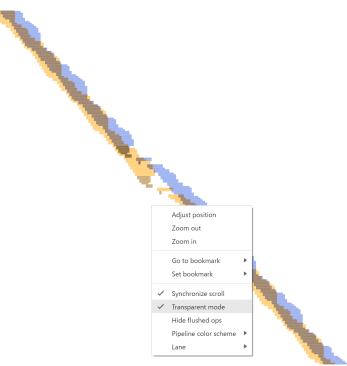


zoom-in

zoom-out

Compare Two Pipelines

- Konata can show two pipelines overlapping.
 (Currently, it requires manual operations as follows:
 - 1. Load two files
 - 2. Right click -> "Transparent mode" & "Synchronize scroll"
 - 3. Right click -> "Pipeline color scheme" to change a pipeline color



- 1. A brief explanation of how to use
- 2. Typical visualization examples
- 3. Use cases

Typical Visualization Examples

- Introduce how the following things are shown:
 - 1. Out-of-order execution
 - 2. Branch misprediction
 - 3. Cache miss
 - 4. Execution speed

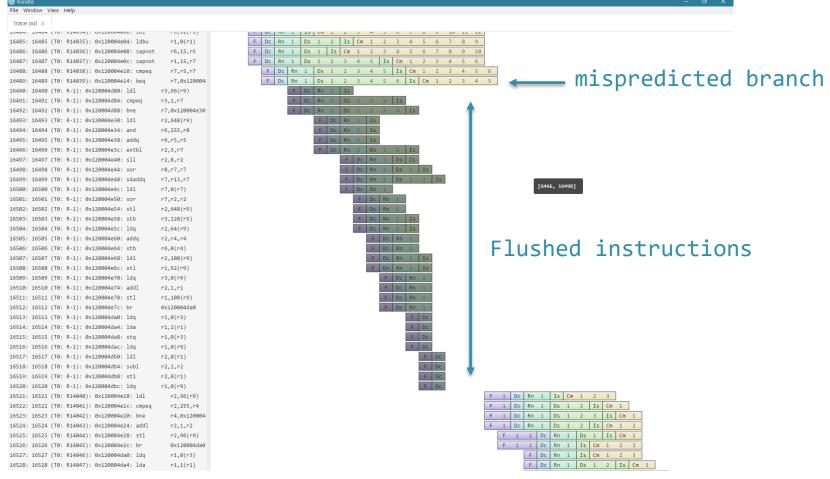
Example: Out-of-order Execution

- Fetch and retirement, marked with the blue circles, are performed in-order
 - \Diamond Instruction issue, marked with the red circles, is performed out-of-order

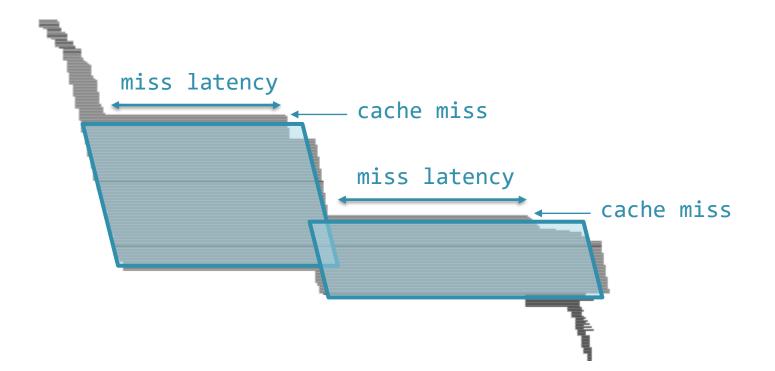
🞯 Konata	- 🗆 X
File Window View Help	
trace.out 🗙	\frown
6461: 6461 (T0: R2421): 0x120016d44: cmpeq	r10,r8,r7 F D Rn 1 Ds 1 2 3 4 Is Cm 1
6462: 6462 (T0: R2422): 0x120016d48: cmpeq	r7,0,r7 F DC Rn 1 Ds 1 2 3 4 5 IS Cm 1
6463: 6463 (T0: R2423): 0x120016d4c: and	r3,r7,r7 F DC R 1 Ds 1 2 3 4 5 6 Is C 1
6464: 6464 (T0: R2424): 0x120016d50: beq	r7,0x12631 F 1 DC Rn 1 Ds 1 2 3 4 5 6 Is [4674, 6463] Cm[2]
6465: 6465 (T0: R2425): 0x120016d54: ldq	r3,568(115 F 1 Dc 2n Is cm 1 2 4 5 [40/4, 6183] Cm[2]
6466: 6466 (T0: R2426): 0x120016d58: bne	r3,0x1201 F 1 Dc R Ds 1 2 Is Cm 2 3 4 5
6467: 6467 (T0: R2427): 0x120016d5c: subl	r10,48,r F 1 Dc Rn 1 D5 1 2 . Is Cm 2 3 4
6468: 6468 (T0: R2428): 0x120016d60: zapnot	r3,15,r3 F DC Rn 1 Ds 1 2 3 Is Cm 2 3
6469: 6469 (T0: R2429): 0x120016d64: cmpule	r3,9,r3 F Dc Rn Ds 1 2 3 4 5 Is Cm 1 2
6470: 6470 (T0: R2430): 0x120016d68: beq	r3,0x1200. F Dc Rn 1 D5 1 2 3 4 5 th Is Cm 1
6471: 6471 (T0: R2431): 0x120016d6c: ldq	r0,616(r19 F Dc Rn L Is Cm 1 2 2 4 5 6 7 8
6472: 6472 (T0: R2432): 0x120016d70: subl	r10,47,r3 F 1 Dc R 1 Ds 1 2 Is Cm 2 3 4 5
6473: 6473 (T0: R2433): 0x120016d74: cmple	r3,r0,r3 F 1 Dc Rn Ds 1 2 Is Cm 1 2 3 4
6474: 6474 (T0: R2434): 0x120016d78: bne	r3,0x12001 F 1 Dc Rn 1 D5 2 3 4 Is 0 1 2 3
6475: 6475 (T0: R2435): 0x120017440: cmpeq	r12,r2,r7 F Dc Rn Is m 1 2 4 5 6 7
6476: 6476 (T0: R2436): 0x120017444: sextb	r10,r3 F Dc Rn 1 Ds 1 Is Cm 1 2 3 4
6477: 6477 (T0: R2437): 0x120017448: bne	r7,0x12001 F Dc Rn 1 Ds Is Cm 1 2 3 4 5 6
6478: 6478 (T0: R2438): 0x12001744c: addq	r4,r12,r22 F Dc Rn 1 Is Cm 1 2 3 4 5 6 7
6479: 6479 (T0: R2439): 0x120017450: subl	r6,1,r8
6480: 6480 (T0: R2440): 0x120017454: lda	r7,1(r10) F Dc Rn 1 Ds 1 Is Cm 1 2 3 4 5 6
6481: 6481 (T0: R2441): 0x120017458: lda	r12,1(r12) F Dc Rn 1 Is Cm 1 2 3 4 5 6 7 a
6482: 6482 (T0: R2442): 0x12001745c: stb	r3,0(r22) F Dc Rn 1 Ds 1 2 Is Cm 1 2 3 4 5
6483: 6483 (T0: R2443): 0x120017460: cmovgt	r6,r8,r6
6484: 6484 (T0: R2444): 0x120017464: beq	r7,0x12001 F Dc Rn 1 Ds 1 Is Cm 1 2 3 4 5
6485: 6485 (T0: R2445): 0x120017468: ldq	r7,8(r11) F Dc Rn 1 Is Cm 1 2 3 4 5 6 7
6486: 6486 (T0: R2446): 0x12001746c: ldq	r3,16(r11) F Dc Rn 1 Is Cm 1 2 3 4 5 6 7
6487: 6487 (T0: R2447): 0x120017470: cmpult	r7,r3,r3 F Dc Rn 1 Ds 1 2 Is Cm 1 2 3 4
6488: 6488 (T0: R2448): 0x120017474: beq	r3,0x12001 F Dc Rn 1 Ds 1 2 3 Is Cm 1 2 3 4 5 6 7
6489: 6489 (T0: R2449): 0x120017478: lda	r3,1(r7) F Dc Rn 1 Ds 1 Is Cm 1 2 3 4 5 6 7 8
6490: 6490 (T0: R2450): 0x12001747c: ldbu	r10,0(r7) F Dc Rn 1 Ds 1 Is Cm 1 2 3 4 5 6 7 8 9
6491: 6491 (T0: R2451): 0x120017480: stq	r3,8(r11) F Dc Rn 1 Ds Is Cm 1 2 3 4 5 6 7 8

Example: Branch Misprediction

\diamond Flushed instructions are shown as dark ones



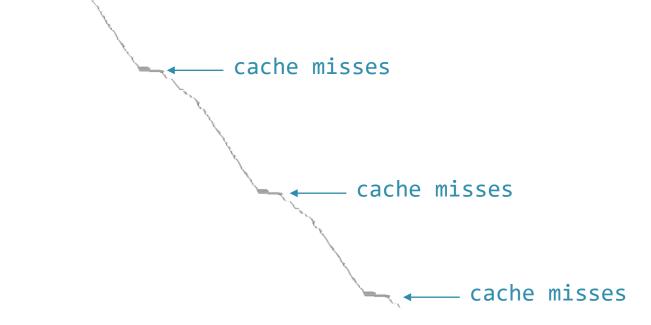
A cache miss is typically shown as a diamond-like shape when the image is zoomed out as follows



Example: Cache Misses

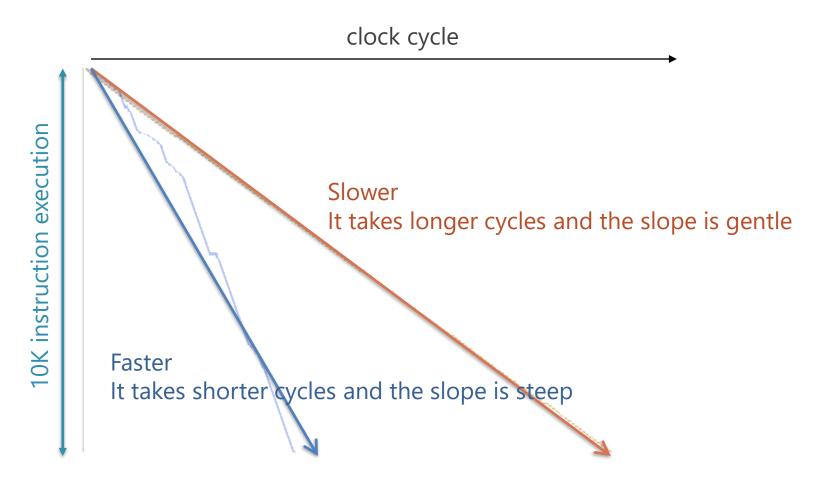
As it is zoomed out more, the pipeline is typically shown as follows

- ◇ This is the pipeline behavior of MCF in SPECCPU 2006.
- \bigcirc This figure shows the performance is degraded by the cache misses.



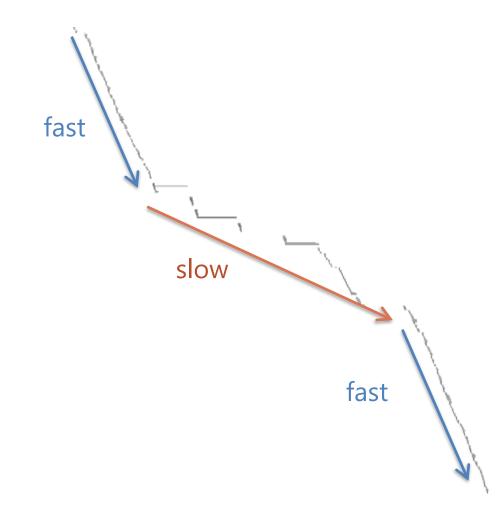
The slope of a pipeline shape roughly represents the execution speed (IPC)

The following two pipelines show the execution of the same 10K instructions



The slope of a pipeline shape roughly represents the execution speed (IPC)

You can see the transition in the execution speed for each part of the program as follows



- 1. A brief explanation of how to use
- 2. Typical visualization examples
- 3. Use cases
 - 1. Grasping the pipeline behavior
 - 2. Comparing pipelines

Grasping the pipeline behavior

- The pipeline visualization makes it easy to grasp the pipeline behavior.
 - Explain this by using some examples
- Let's suppose you newly add speculative execution with branch prediction.
 - \diamond (Of course, gem5 already has this feature
 - ◇ Something wrong happens in recovery from mispredictions.

Investigating with a log

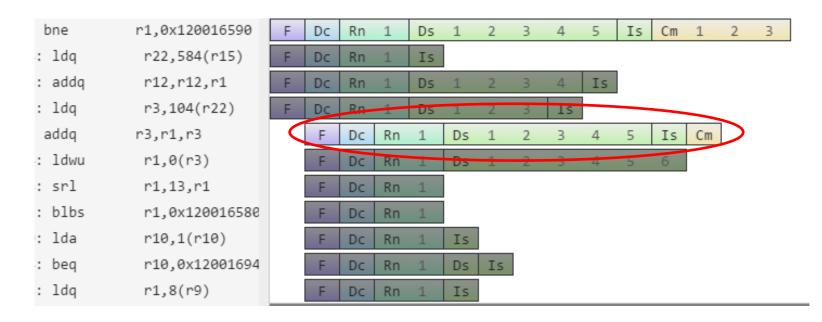
- 15 O3PipeView:fetch:25069250:0x120016358:0:177:addq r12,r12,r1 16 O3PipeView:decode:25069500 O3PipeView:rename:25069750 17 18 O3PipeView:dispatch:25070250 O3PipeView:issue:25071500 19 O3PipeView:complete:25071750 20 O3PipeView:retire:0:store:0 21 O3PipeView:fetch:25069250:0x12001635c:0:178:ldq r3,104(r22) 22 23 O3PipeView:decode:25069500 24 O3PipeView:rename:25069750 O3PipeView:dispatch:25070250 25 O3PipeView:issue:25071250 26 27 O3PipeView:complete:25071500 O3PipeView:retire:0:store:0 28 O3PipeView:fetch:25069500:0x120016360:0:179:addq r3,r1,r3 29 O3PipeView:decode:25069750 30 31 O3PipeView:rename:25070000 32 O3PipeView:dispatch:25070500 O3PipeView:issue:25072000 33 34 O3PipeView:complete:25072250 O3PipeView:retire:25072500:store:0 35 36 O3PipeView:fetch:25069500:0x120016364:0:180:ldwu r1,0(r3) O3PipeView:decode:25069750 37 38 O3PipeView:rename:25070000 O3PipeView:dispatch:25070500 39 O3PipeView:issue:25072250 40 41 O3PipeView:complete:0 42 O3PipeView:retire:0:store:0 43 O3PipeView:fetch:25069500:0x120016368:0:181:srl r1,13,r1 O3PipeView:decode:25069750 44 45 O3PipeView:rename:25070000 46 O3PipeView:dispatch:25070500 O3PipeView:issue:0 47 O3PipeView:complete:0 48
- 49 03PineView:retire:0:store:0

For investigating your implementation, you probably:

- Check custom logs or your "printf" outputs such as the left example
- It records when/what instructions are flushed.
- It's very difficult to detect which point is incorrect from such text logs.

25

Investigating with visualization



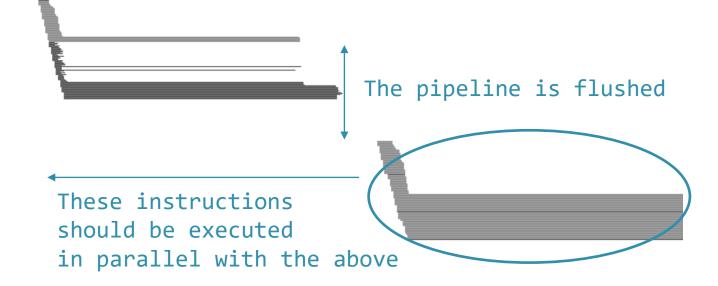
- By visualizing it, you can easily notice the incorrect point.
 - There is the light instruction (not flushed) between the dark flushed instructions.
- Although this is an artificial example,
 - \diamond visualization gives us a lot of hints intuitively!

Another example: memory level parallelism

- One of my friends tackled a topic related to memory level parallelism.
 In short, his method improves the performance by performing multiple memory accesses in parallel.
- He enlarged the size of the OoO scheduling window so that more memory accesses are performed in parallel.
 - \diamond But, the performance is not improved...

Another example: memory level parallelism

- He realized a curious behavior from the following zoomed-out image,
 because the shape is unnatural
- He realized that the pipeline was flushed on a cache miss
 - In this sequence, memory accesses should be performed in parallel



- He examined the flushed instruction in detail and found the cause.
- This was because he used Alpha ISA.
 - In Alpha architecture, TLB miss causes a trap and the pipeline is flushed.
 - \diamond On a cache miss, a TLB miss often occurs.
 - \diamond So, memory accesses cannot be performed in parallel.
- It is not easily noticed simply by observing the counters in gem5.
 - \diamond The shape or pattern of visualized pipelines often tell us hints.

- 1. A brief explanation of how to use
- 2. Typical visualization examples
- 3. Use cases
 - 1. Grasping the pipeline behavior
 - 2. Comparing pipelines

Let's suppose

- \diamond your new method seems to work correctly,
- \diamond but it does not improve the performance as you expected.

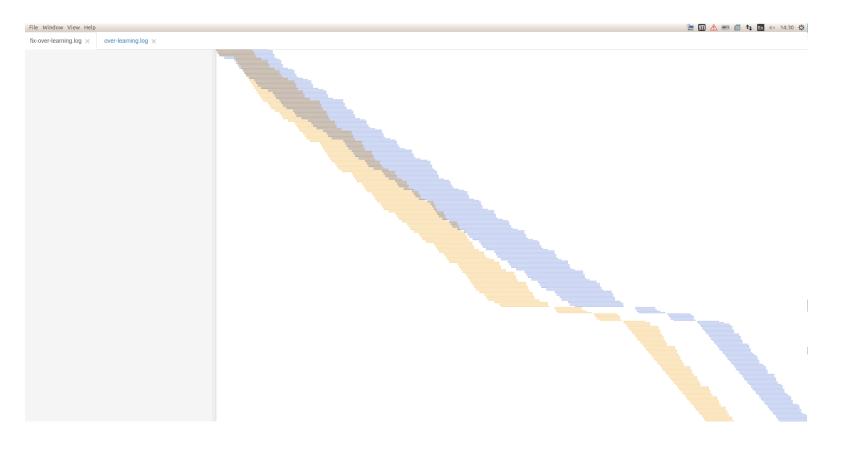
Konata can compare two pipelines!

 \diamond It is useful when investigating the above situation.

Example of comparing

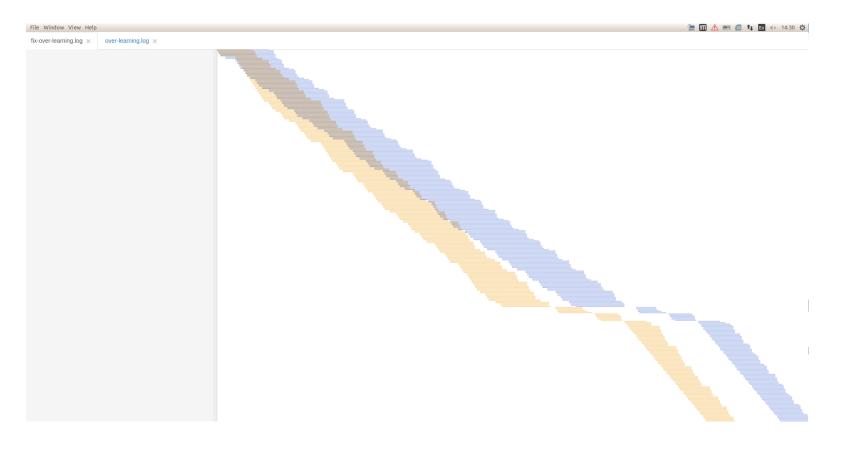
Konata can show two pipelines overlapping.

- \diamond Blue shows a baseline processor pipeline.
- ◇ Orange shows a pipeline with a proposed method.



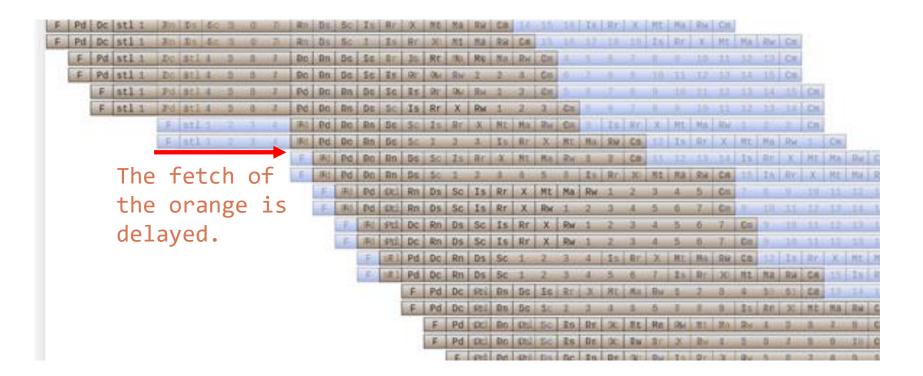
Example of comparing

The orange one (proposed) is basically faster than the blue one (baseline) The insn. fetch and retirement of the orange one are clearly fast.



Example of comparing

- In the zoomed-in image,
 - \diamond in some places, the fetching of the orange is unreasonably delayed.
 - \diamond This was caused by a bug, and this bug degraded the performance.



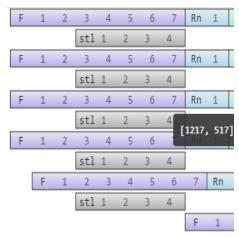
- Visual comparison is very effective for analysis when adding new features to gem5.
 - If the performance is not improved as expected, something is delayed.
 - \diamond You can detect such parts by visual comparison.
 - □ It is easy to see which part is different.

Conclusion

- It is generally difficult to investigate the cause of a bug related to the performance.
 - \diamond Especially, when you have no idea what happened.
- In such cases, visualization is very useful.
 - This presentation introduced a new pipeline viewer Konata and best practice in gem5
- Please try it!
 - https://github.com/shioyadan/Konata/releases

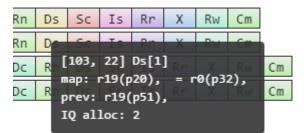
Appendix

- \diamond Add new features:
 - 1. Showing concurrent events such as pipeline stall etc.
 - 2. Dependency arrows between instructions
 - 3. Custom messages for each stage
- \diamond Konata has already had these features.
 - Because Konata was developed for other simulators.
 - □ I have a plan to add support for these features to gem5.



Y	19	X	ŏ	9	STC	1	1	2	3	. Y	1	2	WD	T	т	Ţ	2	5
	X	WB	1	f	1	2	3	4	5	6	7	8	9	10	11	12	13	1
	2	3	X	146	1	-f-	1	2	3	4	5	6	X	8	9	10	11	1
	2	A	×	1	2	3	-4	5	6 -	Wb	1	f	1	2	3	4	5	6
	3	4	5	6	7	8	9	10	11	Slc	T	1	2	3	X	1	2	Wb
	2	3	X	1	2	3	4	5	6=	Wb	1	4	4	2	3	4	5	6
	2	3	4	5	6	7	8	9	10	Slc	I	1	2	3	► X	1	2	Wb
	1	2	À	Х	1	2	3	4	5 ~	Wb	×	f	1	2	3	4	5	6
	2	3	4	5	6	7	8	9	10	Wku	SIC	I	1	2	3	► X	1	2
	1	2	3	X	1	2	3	4	5 ~	Wb	1	f	1	1	3	4	5	б
	1	2	3	4	5	6	7	8	9	Wku	Ste	I	1	2	3	► X	1	2
	Slc	I	1	2	3	Х	Wb	1	f	1	2	3	4	5	6	7	8	9
	1	Slc	I	1	2	3	Х	1	2 -	Wb	1	f	1	2	3	4	5	6
	1	Sw	1	2	3	4	5	6	7	Wku	Wat	SIC	Ι	1	2	3	×Χ	1
	1	Slc	I	1	2	3	Х	1	2	Wb	1	f	1	2	3	4	5	6
	1	Sw	1	2	3	4	5	6	7	Wku	Wat	Slc	Ι	1	2	3	×Χ	1
	D	1	Slc	Ι	1	2	3	Х	1	2	3	4	5	6	7	8	9	1
	D	1	Sw	1	Slc	I	1	2	3	ţХ	1	2	Wb	1	f	1	2	3

2.Dependency arrow



3.Custom log message 38

1.Multiple stage lanes

The slope of a pipeline shape roughly represents the execution speed (IPC)

- \diamond It is not accurate because flushed instructions are also shown.
- If you want to compare accurately, use "Hide flushed ops" option from the right click menu

