SLASH 24

CPU Observability 높이는 Hyperthread 톺아보기

최준우
Server Developer

본 발표자료의 저작권은 연사에 있으며, 저작권자의 사전 서면 동의 없이 자료의 일부 또는 전부를 이용하거나 배포할 수 없습니다.

또한 해당 자료를 복제하여 SLASH 행사 홈페이지를 제외한 온라인상에 게재하는 행위는 연사가 동의한 저작권 및 배포전송권에 위배됩니다.

토스가 다루는 모든 개인정보는 고객에게 동의를 받은 후에 처리되고 있으며, 접근 권한이 분리되어 있습니다.
개발자는 모든 데이터가 아닌 담당 영역에 한하여 접근·이용할 수 있습니다.

toss

시작

김이슈 10:10 AM

전에 구두로 하이퍼쓰레딩 관련해서 문의드린 적 있는데요.
노드의 CPU를 50%를 넘어서 사용할 때 경합 때문에 성능이 떨어지는 문제가 있는지 확인해주시기로 했구요.
관련해서 정리 부탁드립니다.

시작

이토스 10:10 AM

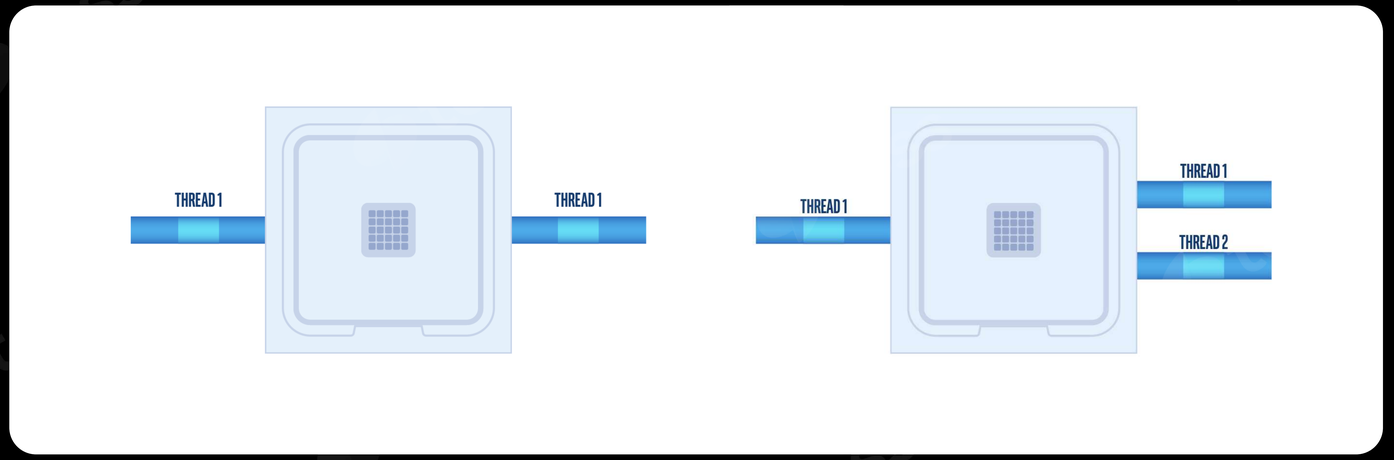
우선 고객 사례중에 비슷한 케이스도 확인되었고, 말씀주신것처럼 CPU 사용률이 50% 이상인 경우 HyperThreading 아키텍쳐가 적합한 모델이 아니므로 HT사용을 권고하지 않는다는 답변이네요.

SLASH 24

시작

Hyperthread??

좀 더 알아보자



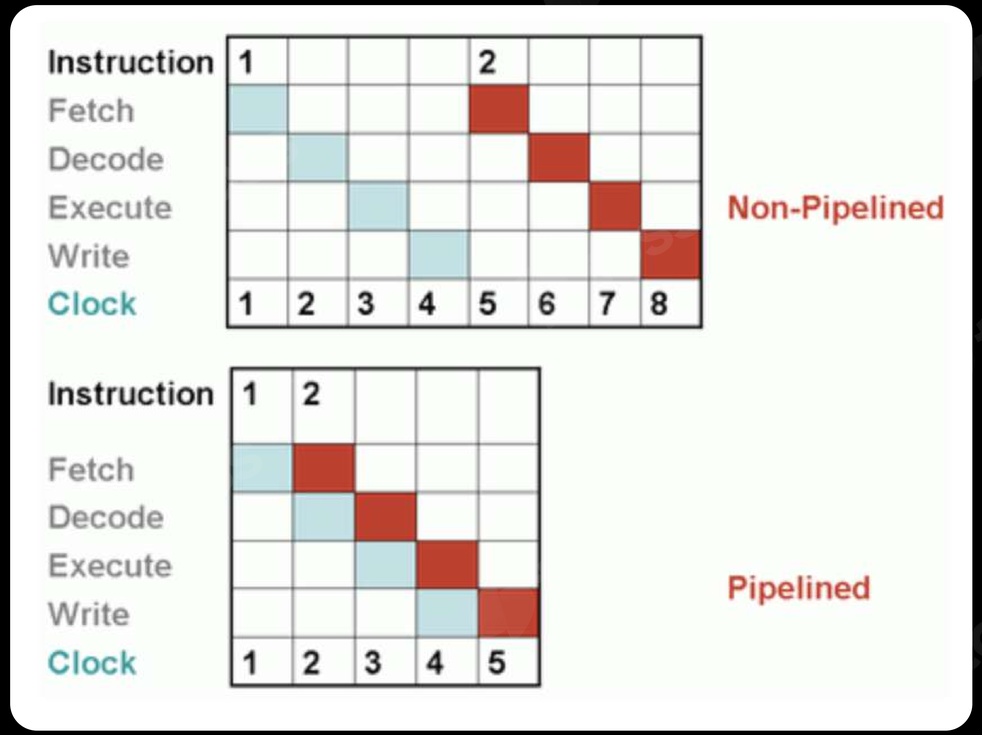
출처 : https://www.intel.co.kr/content/www/kr/ko/gaming/resources/hyper-threading.html

Hyperthread??

pipeline

out of order

super scalar



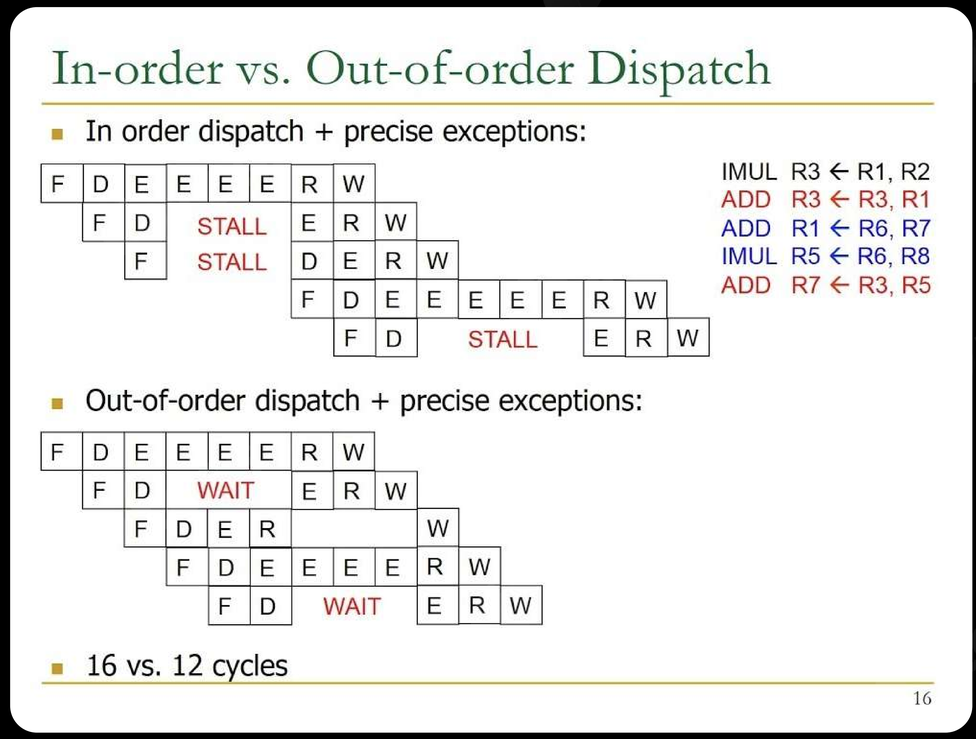
출처 : https://stackpointer.io/hardware/how-pipelining-improves-cpu-performance/113/

Hyperthread??

pipeline

out of order

super scalar



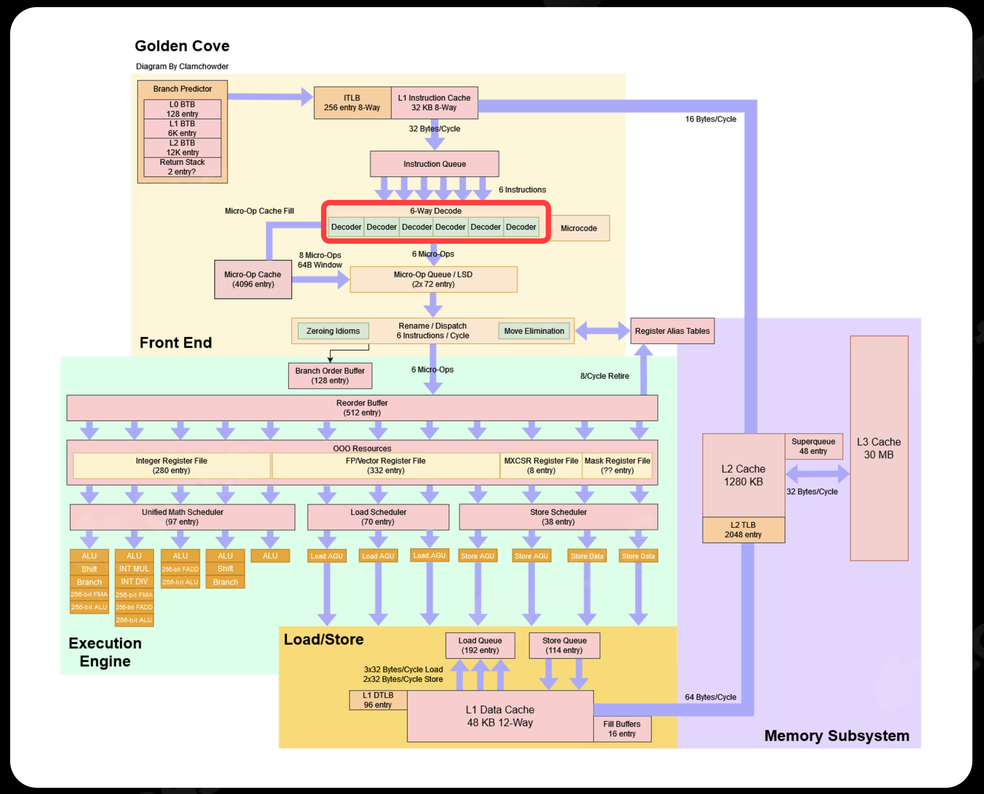
출처 : https://youtu.be/AiEsYzTDB5c?si=13kUSjijdtd_hgak

Hyperthread??

pipeline

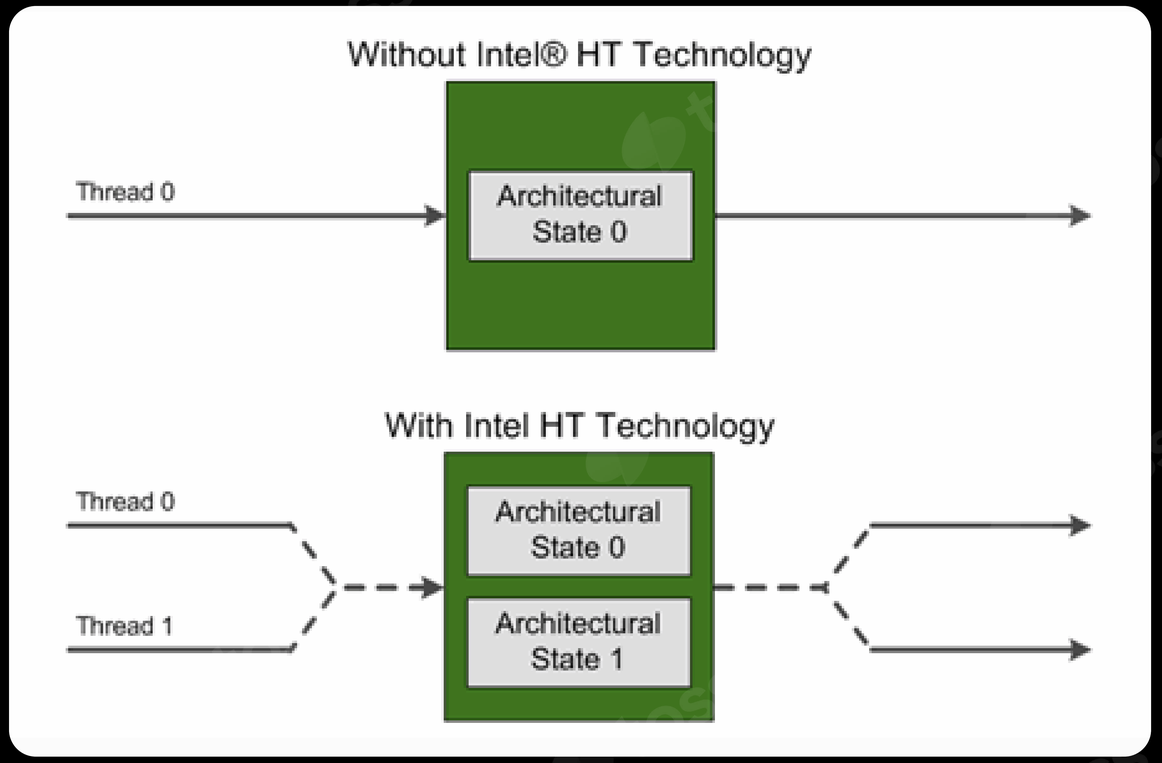
out of order

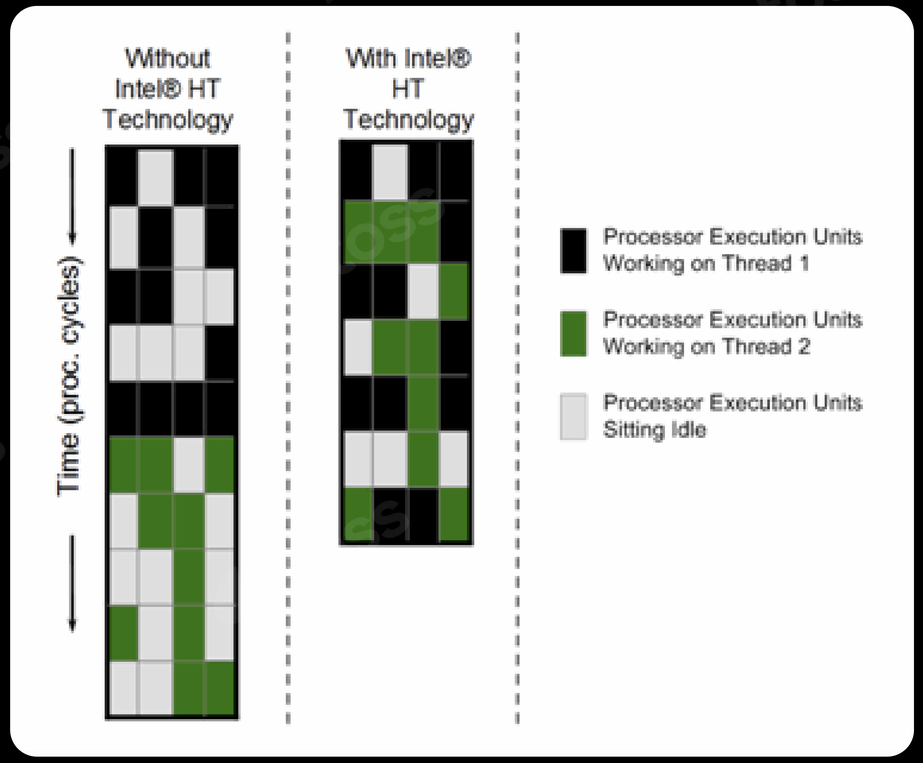
super scalar



출처 : https://chipsandcheese.com/2021/12/02/popping-the-hood-on-golden-cove/

Hyperthread??





출처 : https://web.archive.org/web/20150217050949/https://software.intel.com/en-us/articles/performance-insights-to-intel-hyper-threading-technology/

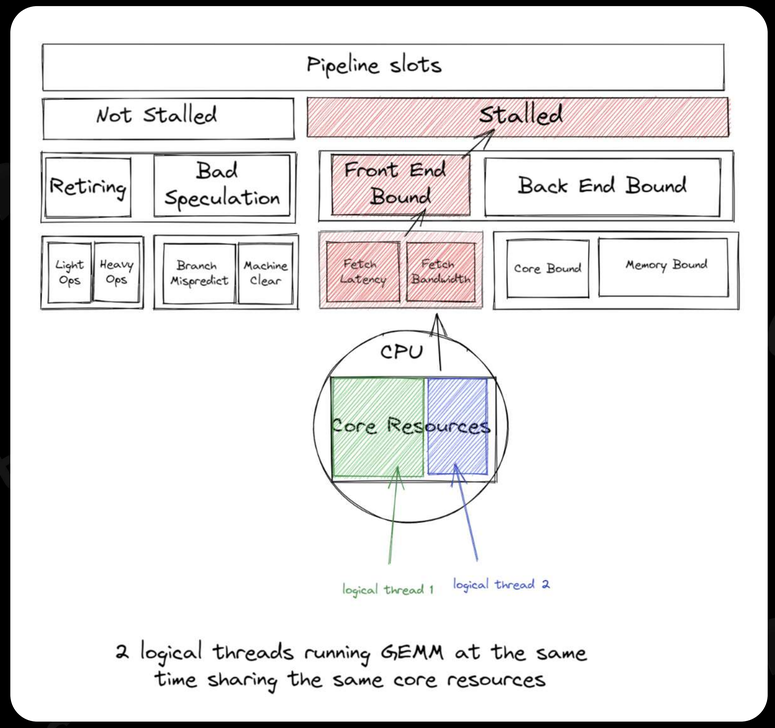
Hyperthread??

그렇게 생각했던 시기가 저에게도 있었지요

Hyperthread??

Noisy Neighbor

Hyperthread??



출처 : https://tutorials.pytorch.kr/intermediate/torchserve_with_ipex.html

Noisy Neighbor

Hyperthread??

From: Linus Torvalds <torvalds-AT-linux-foundation.org>
To: Thomas Gleixner <tglx-AT-linutronix.de>
Subject: Re: [patch V2 27/28] x86/speculation: Add seccomp Spectre v2 user space protection mode
Date: Sun, 25 Nov 2018 12:40:14 -0800
Message-ID: <CAHk-=whtiX45YPjFPMFuktZ3WB23zqBZR-rL6RewrTyvrFj2Fw@mail.gmail.com>
Cc: Linux List Kernel Mailing <linux-kernel-AT-vger.kernel.org>, "the arch/x86 maintainers" <x86-AT-kernel.org>, Peter Zijlstra <peterz-AT-infradead.org>, Andrew Lutomirski <luto-AT-kernel.org>, Jiri Kosina <jkosina-AT-suse.cz>, thomas.lendacky-AT-amd.com, Josh Poimboeuf <jpoimboe-AT-redhat.com>, Andrea Arcangeli <aarcange-AT-redhat.com>, David Woodhouse <dwmw-AT-amazon.co.uk>, Tim Chen <tim.c.chen-AT-linux.intel.com>, Andi Kleen <ak-AT-linux.intel.com>, dave.hansen-AT-intel.com, Casey Schaufler <casey.schaufler-AT-intel.com>, "Mallick, Asit K" <asit.k.mallick-AT-intel.com>, "Van De Ven, Arjan" <arjan-AT-linux.intel.com>, jcm-AT-redhat.com, longman9394-AT-gmail.com, Greg KH <gregkh-AT-linuxfoundation.org>, david.c.stewart-AT-intel.com, Kees Cook <keescook-AT-chromium.org>

```
[ You forgot to fix your quilt setup.. ]

On Sun, 25 Nov 2018, Thomas Gleixner wrote:
>
> The mitigation guide documents how STIPB works:
>
>    Setting bit 1 (STIBP) of the IA32_SPEC_CTRL MSR on a logical processor
>    prevents the predicted targets of indirect branches on any logical
>    processor of that core from being controlled by software that executes
>    (or executed previously) on another logical processor of the same core.

Can we please just fix this stupid lie?

Yes, Intel calls it "STIBP" and tries to make it out to be about the
indirect branch predictor being per-SMT thread.

But the reason it is unacceptable is apparently because in reality it just
disables indirect branch prediction entirely. So yes, *technically* it's
true that that limits indirect branch prediction to just a single SMT
core, but in reality it is just a "go really slow" mode.

If STIBP had actually just keyed off the logical SMT thread, we wouldn't
need to have worried about it in the first place.

So let's document reality rather than Intel's Pollyanna world-view.

Reality matters. It's why we had to go all this. Lying about things
and making it appear like it's not a big deal was why the original
patch made it through without people noticing.

          Linus
```

출처 : https://lkml.org/lkml/2018/11/26/58

Vulnerability

Hyperthread??

1. 쓰레드간 불균형이 심한경우
2. 이미 높은 메모리 대역폭
3. 이미 높은 Instruction Per Cycle
4. false sharing, 잠금 / 동기화 가 많은 경우
5. OS CPU 스케줄러가 SMT 미지원일때

Hyperthread??

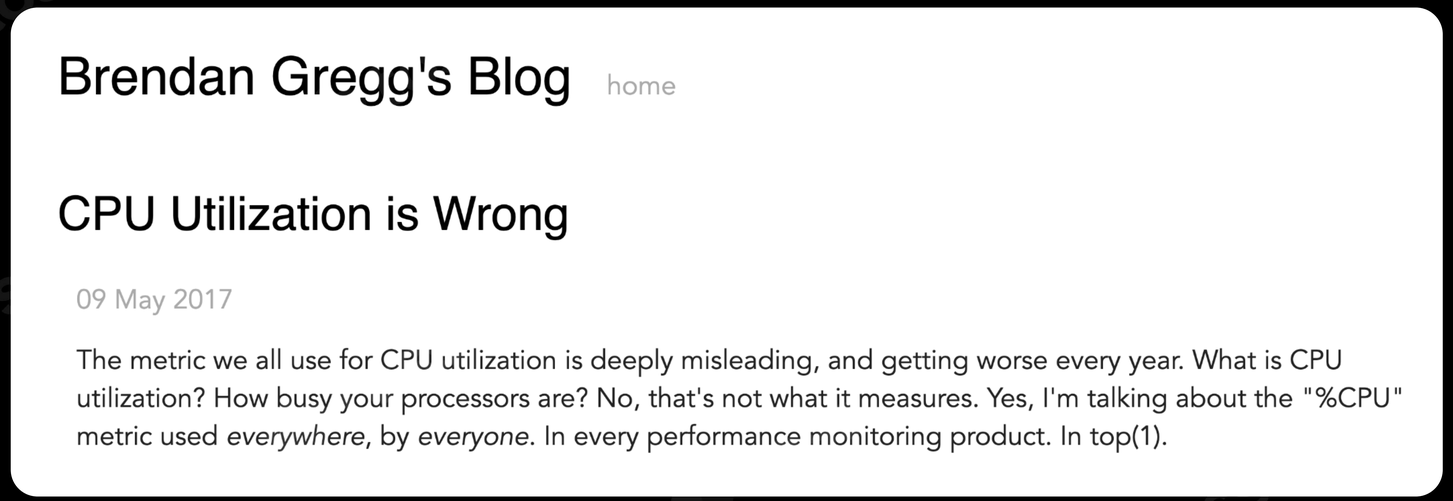
"There's no such thing as a free lunch."

성능 측정

Elasticsearch

- JVM netty 기반
- http api & json parsing
- 실서비스 노드와 비슷한 서버 사양
- 높은 CPU 사용률

성능 측정



Brendan Gregg's Blog home

CPU Utilization is Wrong

09 May 2017

The metric we all use for CPU utilization is deeply misleading, and getting worse every year. What is CPU utilization? How busy your processors are? No, that's not what it measures. Yes, I'm talking about the "%CPU" metric used *everywhere*, by *everyone*. In every performance monitoring product. In top(1).

출처 : https://www.brendangregg.com/blog/2017-05-09/cpu-utilization-is-wrong.html

CPU usage는 노이즈가 있으니 Instruction 처리량을 확인해야한다

성능 측정

Linux-KI	hp에서 개발한 커널 프로파일링 double busy라는 hyperthread 전용 메트릭이 나와 처음 측정에 사용한 툴
perf	리눅스에서 제공하는 프로파일링 도구

성능 측정

1. 다른 노드에서 on vs off
2. 같은 노드에서 on vs off

기타 성능에 영향을 주는 Numa, CPU pinning, C-State 는 동일하게 설정

성능 측정

다른 노드에서 측정

on / off 노드 둘 다 CPU 100%

성능 측정

다른 노드에서 측정

on / off 노드 둘 다 CPU 100%

성능 측정

Linux-KI 비교

1.1.4 Hyperthread CPU Usage

[Prev Subsection][Next Subsection]---[Prev Section][Next Section]

PCPU	double idle	lcpu1 busy	lcpu2 busy	double busy
[0 32]:	2.8%	4.6%	7.9%	84.6%
[1 33]:	0.8%	5.7%	4.2%	89.2%
[2 34]:	2.4%	7.2%	7.5%	82.9%
[3 35]:	1.6%	5.1%	4.2%	89.2%
[4 36]:	4.7%	5.5%	3.3%	86.4%
[5 37]:	0.7%	5.2%	4.4%	89.7%
[6 38]:	5.2%	6.8%	4.3%	83.8%
[7 39]:	0.6%	5.5%	4.1%	89.7%
[8 40]:	5.2%	5.8%	4.8%	84.2%
[9 41]:	0.7%	5.6%	3.3%	90.4%
[10 42]:	2.3%	4.6%	8.0%	85.1%
[11 43]:	0.9%	4.0%	4.5%	90.6%
[12 44]:	2.4%	4.4%	7.6%	85.7%
[13 45]:	1.0%	4.8%	4.0%	90.3%
[14 46]:	2.5%	6.2%	5.6%	85.7%
[15 47]:	0.8%	5.1%	4.5%	89.6%
[16 48]:	4.6%	6.1%	4.2%	85.1%
[17 49]:	0.7%	4.6%	4.0%	90.7%
[18 50]:	2.3%	8.2%	4.7%	84.8%
[19 51]:	0.8%	4.7%	4.4%	90.0%
[20 52]:	2.2%	4.3%	7.5%	85.9%
[21 53]:	1.0%	4.9%	5.0%	89.2%
[22 54]:	4.2%	6.8%	4.5%	84.5%
[23 55]:	0.7%	4.6%	4.3%	90.3%
[24 56]:	3.4%	9.5%	3.0%	84.0%
[25 57]:	1.1%	3.5%	4.5%	90.9%
[26 58]:	4.9%	6.6%	3.2%	85.4%
[27 59]:	0.4%	4.6%	4.9%	90.1%
[28 60]:	3.0%	5.6%	6.2%	85.3%
[29 61]:	0.9%	5.6%	3.5%	90.0%
[30 62]:	2.6%	7.6%	5.2%	84.6%
[31 63]:	0.6%	3.0%	5.6%	90.8%
Total	**2.1%**	**5.5%**	**4.9%**	**87.5%**

성능 측정

Linux-KI 비교

Warning: CPU Bottleneck (Idle < 10%) [Next]

1.1.1 Global CPU Usage by CPU

[Prev Subsection][Next Subsection]---[Prev Section][Next Section]

cpu	node	Total Busy	sys	usr	idle
0	[0] :	98.04%	4.10%	93.58%	1.91%
1	[1] :	97.72%	4.11%	93.28%	2.22%
2	[0] :	97.40%	3.69%	91.28%	2.51%
3	[1] :	98.14%	4.12%	93.72%	1.81%
4	[0] :	97.65%	3.91%	93.42%	2.31%
5	[1] :	97.74%	3.70%	93.79%	2.21%
6	[0] :	97.18%	3.18%	91.05%	2.73%
7	[1] :	97.88%	2.86%	94.79%	2.08%
8	[0] :	97.32%	4.01%	91.42%	2.60%
9	[1] :	96.74%	3.74%	92.79%	3.21%
10	[0] :	97.26%	4.34%	92.68%	2.70%
11	[1] :	97.29%	3.87%	93.18%	2.66%
12	[0] :	97.90%	4.81%	91.58%	2.04%
13	[1] :	97.41%	3.49%	93.72%	2.54%
14	[0] :	98.55%	3.94%	92.95%	1.39%
15	[1] :	97.35%	3.73%	93.38%	2.59%
16	[0] :	97.67%	4.35%	92.21%	2.26%
17	[1] :	98.35%	4.58%	93.47%	1.60%
18	[0] :	97.81%	4.00%	93.44%	2.15%
19	[1] :	98.05%	2.98%	94.83%	1.91%
20	[0] :	97.47%	4.26%	89.20%	2.41%
21	[1] :	97.95%	3.69%	94.00%	2.00%
22	[0] :	98.09%	3.97%	93.79%	1.87%
23	[1] :	96.84%	3.79%	92.80%	3.11%
24	[0] :	97.59%	3.74%	93.45%	2.36%
25	[1] :	98.17%	3.25%	94.68%	1.78%
26	[0] :	98.25%	4.02%	93.88%	1.71%
27	[1] :	96.58%	4.05%	92.28%	3.38%
28	[0] :	97.80%	4.68%	92.80%	2.15%
29	[1] :	96.53%	3.37%	92.91%	3.42%
30	[0] :	98.00%	4.20%	93.22%	1.94%
31	[1] :	95.75%	3.13%	92.38%	4.20%
Total		**97.58%**	**3.86%**	**93.00%**	**2.37%**

[CSV]

성능 측정

perf 비교

```
#           time             counts unit events
     5.018804079        320,194.08 msec cpu-clock                 #  454.155 CPUs utilized
     5.018804079           210,252      context-switches          #    0.657 K/sec
     5.018804079             3,559      cpu-migrations            #    0.011 K/sec
     5.018804079             2,299      page-faults               #    0.007 K/sec
     5.018804079   736,991,437,457      cycles                    #    2.302 GHz                      (30.82%)
     5.018804079   767,075,591,420      instructions              #    1.04  insn per cycle           (38.50%)
     5.018804079   126,641,577,808      branches                  #  395.515 M/sec                    (38.51%)
     5.018804079     1,184,935,985      branch-misses             #    0.94% of all branches          (38.52%)
     5.018804079   225,660,868,259      L1-dcache-loads           #  704.763 M/sec                    (38.52%)
     5.018804079    11,846,988,264      L1-dcache-load-misses     #    5.25% of all L1-dcache hits    (38.51%)
     5.018804079     1,836,339,823      LLC-loads                 #    5.735 M/sec                    (30.80%)
     5.018804079       472,689,945      LLC-load-misses           #   25.74% of all LL-cache hits     (30.78%)
     5.018804079   <not supported>      L1-icache-loads
     5.018804079    15,421,958,457      L1-icache-load-misses                                         (30.77%)
     5.018804079   225,812,175,248      dTLB-loads                #  705.235 M/sec                    (30.77%)
     5.018804079       220,544,271      dTLB-load-misses          #    0.10% of all dTLB cache hits   (30.77%)
     5.018804079       813,335,479      iTLB-loads                #    2.540 M/sec                    (30.77%)
     5.018804079       150,078,315      iTLB-load-misses          #   18.45% of all iTLB cache hits   (30.78%)
     5.018804079   <not supported>      L1-dcache-prefetches
     5.018804079   <not supported>      L1-dcache-prefetch-misses
```

```
#           time             counts unit events
     5.008640330        159,864.14 msec cpu-clock                 #  226.747 CPUs utilized
     5.008640330           128,407      context-switches          #    0.803 K/sec
     5.008640330             2,638      cpu-migrations            #    0.017 K/sec
     5.008640330            26,019      page-faults               #    0.163 K/sec
     5.008640330   374,374,768,189      cycles                    #    2.342 GHz                      (52.27%)
     5.008640330   576,251,944,616      instructions              #    1.54  insn per cycle           (58.98%)
     5.008640330    94,582,799,370      branches                  #  591.645 M/sec                    (59.72%)
     5.008640330       839,477,856      branch-misses             #    0.89% of all branches          (60.36%)
     5.008640330   168,508,555,554      L1-dcache-loads           # 1054.074 M/sec                    (60.38%)
     5.008640330     7,684,421,148      L1-dcache-load-misses     #    4.56% of all L1-dcache hits    (60.39%)
     5.008640330     1,355,535,703      LLC-loads                 #    8.479 M/sec                    (55.41%)
     5.008640330       272,723,960      LLC-load-misses           #   20.12% of all LL-cache hits     (55.23%)
     5.008640330   <not supported>      L1-icache-loads
     5.008640330     8,479,351,277      L1-icache-load-misses                                         (55.22%)
     5.008640330   168,763,460,136      dTLB-loads                # 1055.668 M/sec                    (55.32%)
     5.008640330        25,348,735      dTLB-load-misses          #    0.02% of all dTLB cache hits   (50.07%)
     5.008640330       254,573,771      iTLB-loads                #    1.592 M/sec                    (50.80%)
     5.008640330        65,842,183      iTLB-load-misses          #   25.86% of all iTLB cache hits   (51.52%)
     5.008640330   <not supported>      L1-dcache-prefetches
     5.008640330   <not supported>      L1-dcache-prefetch-misses
```

성능 측정

다른 노드에서 on vs off

하이퍼쓰레드를 켰을때 30%정도 더 좋은 성능을 보인다

캐시 미스, 브랜치 미스가 안좋아지는게 보임

cycle 이 거의 2배가 되었지만 instruction 처리량은 비례해서 늘지 않음

성능 측정

동일 노드에서 측정

성능 측정

Linux-KI 비교

1.1.4 Hyperthread CPU Usage

[Prev Subsection][Next Subsection]---[Prev Section][Next Section][Tabl

PCPU		double idle	lcpu1 busy	lcpu2 busy	double busy
[0	32]:	4.5%	6.5%	8.3%	80.8%
[1	33]:	6.2%	9.0%	6.6%	78.1%
[2	34]:	3.9%	6.8%	6.5%	82.7%
[3	35]:	5.7%	9.3%	7.9%	77.0%
[4	36]:	5.1%	6.7%	6.4%	81.8%
[5	37]:	4.5%	7.9%	7.0%	80.7%
[6	38]:	3.4%	8.7%	5.7%	82.2%
[7	39]:	4.8%	7.3%	10.1%	77.8%
[8	40]:	4.1%	8.2%	5.5%	82.2%
[9	41]:	5.8%	7.4%	8.2%	78.6%
[10	42]:	4.6%	7.2%	7.4%	80.8%
[11	43]:	4.2%	8.7%	7.4%	79.7%
[12	44]:	4.7%	7.9%	6.9%	80.4%
[13	45]:	6.3%	10.5%	7.2%	76.1%
[14	46]:	4.2%	7.7%	6.5%	81.6%
[15	47]:	4.1%	9.7%	6.7%	79.5%
[16	48]:	3.6%	7.4%	7.6%	81.4%
[17	49]:	4.6%	7.4%	7.5%	80.5%
[18	50]:	4.9%	9.0%	5.7%	80.4%
[19	51]:	4.8%	7.1%	9.0%	79.1%
[20	52]:	4.4%	8.2%	6.4%	80.9%
[21	53]:	5.8%	7.1%	8.5%	78.6%
[22	54]:	6.0%	9.6%	7.0%	77.4%
[23	55]:	4.7%	6.4%	6.5%	82.4%
[24	56]:	4.1%	6.5%	7.2%	82.2%
[25	57]:	6.0%	7.8%	7.1%	79.1%
[26	58]:	4.0%	6.3%	6.8%	83.0%
[27	59]:	6.7%	9.2%	6.8%	77.2%
[28	60]:	4.0%	7.1%	6.2%	82.7%
[29	61]:	6.2%	9.5%	7.3%	76.9%
[30	62]:	4.7%	8.0%	6.2%	81.1%
[31	63]:	4.9%	9.0%	8.2%	77.9%
Total		**4.9%**	**8.0%**	**7.1%**	**80.0%**

1.2 CPU Usage by Task

성능 측정

Linux-KI 비교

1.1.1 Global CPU Usage by CPU

[Prev Subsection] [Next Subsection] --- [Prev Section] [Next Section]

cpu	node	Total Busy	sys	usr	idle
0	[0] :	96.11%	4.61%	89.04%	3.77%
1	[1] :	95.69%	3.23%	92.28%	4.23%
2	[0] :	95.08%	4.22%	88.91%	4.79%
3	[1] :	96.73%	2.87%	93.76%	3.21%
4	[0] :	97.19%	4.00%	91.29%	2.71%
5	[1] :	96.55%	3.68%	92.75%	3.38%
6	[0] :	97.24%	3.95%	91.36%	2.67%
7	[1] :	96.97%	3.21%	93.67%	2.96%
8	[0] :	95.44%	4.21%	89.31%	4.43%
9	[1] :	96.54%	3.37%	93.07%	3.39%
10	[0] :	96.16%	4.10%	91.84%	3.78%
11	[1] :	97.05%	3.36%	93.58%	2.88%
12	[0] :	96.00%	3.52%	92.32%	3.93%
13	[1] :	97.03%	3.17%	93.76%	2.90%
14	[0] :	96.46%	4.60%	91.57%	3.48%
15	[1] :	96.02%	4.52%	91.39%	3.92%
16	[0] :	96.20%	3.89%	90.93%	3.70%
17	[1] :	97.18%	3.43%	93.65%	2.75%
18	[0] :	96.86%	5.44%	89.79%	3.05%
19	[1] :	95.33%	3.61%	91.61%	4.62%
20	[0] :	95.21%	4.70%	90.29%	4.73%
21	[1] :	95.70%	4.06%	91.53%	4.24%
22	[0] :	97.29%	3.19%	93.97%	2.66%
23	[1] :	95.26%	4.03%	91.13%	4.68%
24	[0] :	96.81%	4.02%	92.55%	3.14%
25	[1] :	96.15%	3.30%	92.77%	3.79%
26	[0] :	97.00%	3.81%	91.46%	2.92%
27	[1] :	95.83%	3.78%	91.94%	4.10%
28	[0] :	95.82%	3.66%	91.98%	4.11%
29	[1] :	96.44%	3.06%	93.29%	3.50%
30	[0] :	96.28%	4.00%	92.13%	3.66%
31	[1] :	96.92%	3.52%	93.30%	3.02%
Total		**96.33%**	**3.82%**	**91.94%**	**3.60%**

[CSV]

성능 측정

perf 비교

```
15.027127069        319,436.91 msec cpu-clock                 #  453.081 CPUs utilized
15.027127069           225,632      context-switches          #    0.706 K/sec
15.027127069             4,096      cpu-migrations            #    0.013 K/sec
15.027127069           123,218      page-faults               #    0.385 K/sec
15.027127069   711,285,391,560      cycles                    #    2.225 GHz                      (27.01%)
15.027127069   772,095,349,581      instructions              #    1.10  insn per cycle           (33.82%)
15.027127069   130,972,902,929      branches                  #  409.632 M/sec                    (33.66%)
15.027127069     1,178,235,590      branch-misses             #    0.91% of all branches          (33.76%)
15.027127069   226,401,972,698      L1-dcache-loads           #  708.097 M/sec                    (33.87%)
15.027127069    11,464,018,902      L1-dcache-load-misses     #    5.06% of all L1-dcache hits    (33.92%)
15.027127069     1,664,979,146      LLC-loads                 #    5.207 M/sec                    (27.29%)
15.027127069       460,857,429      LLC-load-misses           #   27.97% of all LL-cache hits     (27.27%)
15.027127069   <not supported>      L1-icache-loads
15.027127069    14,447,604,323      L1-icache-load-misses                                        (27.20%)
15.027127069   225,441,688,126      dTLB-loads                #  705.094 M/sec                    (27.17%)
15.027127069       181,443,520      dTLB-load-misses          #    0.08% of all dTLB cache hits   (27.15%)
15.027127069       752,727,342      iTLB-loads                #    2.354 M/sec                    (27.03%)
15.027127069       146,376,797      iTLB-load-misses          #   19.45% of all iTLB cache hits   (27.04%)
15.027127069   <not supported>      L1-dcache-prefetches
15.027127069   <not supported>      L1-dcache-prefetch-misses
```

```
15.011297485        159,658.08 msec cpu-clock                 #  226.455 CPUs utilized
15.011297485            82,416      context-switches          #    0.516 K/sec
15.011297485             1,352      cpu-migrations            #    0.008 K/sec
15.011297485           175,802      page-faults               #    0.001 M/sec
15.011297485   377,295,189,588      cycles                    #    2.362 GHz                      (31.14%)
15.011297485   598,506,750,506      instructions              #    1.60  insn per cycle           (35.60%)
15.011297485    95,644,410,362      branches                  #  598.797 M/sec                    (38.06%)
15.011297485       821,205,638      branch-misses             #    0.87% of all branches          (40.15%)
15.011297485   186,127,306,143      L1-dcache-loads           # 1165.280 M/sec                    (40.15%)
15.011297485     5,196,651,048      L1-dcache-load-misses     #    2.99% of all L1-dcache hits    (40.15%)
15.011297485       925,179,670      LLC-loads                 #    5.792 M/sec                    (23.41%)
15.011297485       396,808,337      LLC-load-misses           #   32.42% of all LL-cache hits     (22.87%)
15.011297485   <not supported>      L1-icache-loads
15.011297485     3,590,942,396      L1-icache-load-misses                                        (22.86%)
15.011297485   183,938,491,296      dTLB-loads                # 1151.576 M/sec                   (23.24%)
15.011297485        14,444,707      dTLB-load-misses          #    0.01% of all dTLB cache hits   (23.72%)
15.011297485        95,141,611      iTLB-loads                #    0.596 M/sec                    (26.18%)
15.011297485        30,686,226      iTLB-load-misses          #   15.60% of all iTLB cache hits   (28.65%)
15.011297485   <not supported>      L1-dcache-prefetches
15.011297485   <not supported>      L1-dcache-prefetch-misses
```

성능 측정

같은 노드에서 on vs off

여전히 30%정도 더 좋은 성능을 보인다

결론(?)

Hyperthread 켜면 좋다

결론 아니고..

모니터링

모니터링

무엇이 문제인가?

자원 경합으로 인한 stall 증가

모니터링

문제 상황을 만들어보자

```
int main() {
    int i,j,k,l,m,n;
    long count;
    while(count < 10000000000) {
        i++;
        j++;
        k++;
        l++;
        m++;
        n++;
        i--;
        j--;
        k--;
        l--;
        m--;
        n--;
        count++;
    }
    return 0;
}
```

모니터링

다른 코어에서 실행시

```
time taskset -c 47 ./test

real    1m0.504s
user    1m0.399s
sys     0m0.000s
```

```
time taskset -c 6 ./test

real    1m0.304s
user    1m0.159s
sys     0m0.003s
```

모니터링

같은 코어에서 실행시

```
time taskset -c 47 ./test

real    2m4.867s
user    2m4.521s
sys     0m0.000s
```

```
time taskset -c 23 ./test

real    2m4.850s
user    2m4.533s
sys     0m0.001s
```

모니터링

EC2 Dedicated Host PMCs

The PMCs available are the architectural PMCs listed in the Intel 64 and IA-32 Architectures Developer's Manual: vol. 3B, in section 18.2.1.2 "Pre-defined Architectural Performance Events", Table 18-1 "UMask and Event Select Encodings for Pre-Defined Architectural Performance Events". I've drawn my own table of them below with example event mnemonics.

Architectural PMCs

Event Name	UMask	Event Select	Example Event Mask Mnemonic
UnHalted Core Cycles	00H	3CH	CPU_CLK_UNHALTED.THREAD_P
Instruction Retired	00H	C0H	INST_RETIRED.ANY_P
UnHalted Reference Cycles	01H	3CH	CPU_CLK_THREAD_UNHALTED.REF_XCLK
LLC Reference	4FH	2EH	LONGEST_LAT_CACHE.REFERENCE
LLC Misses	41H	2EH	LONGEST_LAT_CACHE.MISS
Branch Instruction Retired	00H	C4H	BR_INST_RETIRED.ALL_BRANCHES
Branch Misses Retired	00H	C5H	BR_MISP_RETIRED.ALL_BRANCHES

출처 : https://www.brendangregg.com/blog/2017-05-04/the-pmcs-of-ec2.html

Gregg know the answer

모니터링

CHAPTER 20
PERFORMANCE MONITORING

Intel 64 and IA-32 architectures provide facilities for monitoring performance via a PMU (Performance Monitoring Unit).

NOTE

Performance monitoring events can be found here: https://perfmon-events.intel.com/.

Additionally, performance monitoring event files for Intel processors are hosted by the Intel Open Source Technology Center. These files can be downloaded here: https://download.01.org/perfmon/.

출처 : https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html

Performance Monitoring Unit

모니터링

역시나 인텔에서 perf 에 넣어둠

https://github.com/torvalds/linux/blob/master/tools/perf/pmu-events/arch/x86/skylake/skl-metrics.json

6th Generation Intel® Core™ Processor

This section provides reference for hardware events that can be monitored for the CPU(s):

- Events for Intel® microarchitecture code name Skylake
- Events for Intel® microarchitecture code name Skylake (: Skylake_ULT)

Event Name	Description	Additional Info
CPU_CLK_UNHALTED.THREAD	Counts the number of core cycles while the thread is not in a halt state. The thread enters the halt state when it is running the HLT instruction. This event is a component in many key event ratios. The core frequency may change from time to time due to transitions associated with Enhanced Intel SpeedStep Technology or TM2. For this reason this event may have a changing ratio with regards to time. When the core frequency is constant, this event can approximate elapsed time while the core was not in the halt state. It is counted on a dedicated fixed counter, leaving the four (eight when Hyperthreading is disabled) programmable counters available for other events.	IA32_FIXED_CTR1 Architectural, Fixed
CPU_CLK_UNHALTED.THREAD_ANY	Core cycles when at least one thread on the physical core is not in halt state.	IA32_FIXED_CTR1 Architectural, Fixed
CPU_CLK_UNHALTED.THREAD_P	This is an architectural event that counts the number of thread cycles while the thread is not in a halt state. The thread enters the halt state when it is running the HLT instruction. The core frequency may change from time to time due to power or thermal throttling. For this reason, this event may have a changing ratio with regards to wall clock time.	EventSel=3CH UMask=00H Counter=0,1,2,3 CounterHTOff=0,1,2,3,4,5,6,7 Architectural
CPU_CLK_UNHALTED.THREAD_P_ANY	Core cycles when at least one thread on the physical core is not in halt state.	EventSel=3CH UMask=00H AnyThread=1 Counter=0,1,2,3 CounterHTOff=0,1,2,3,4,5,6,7 Architectural

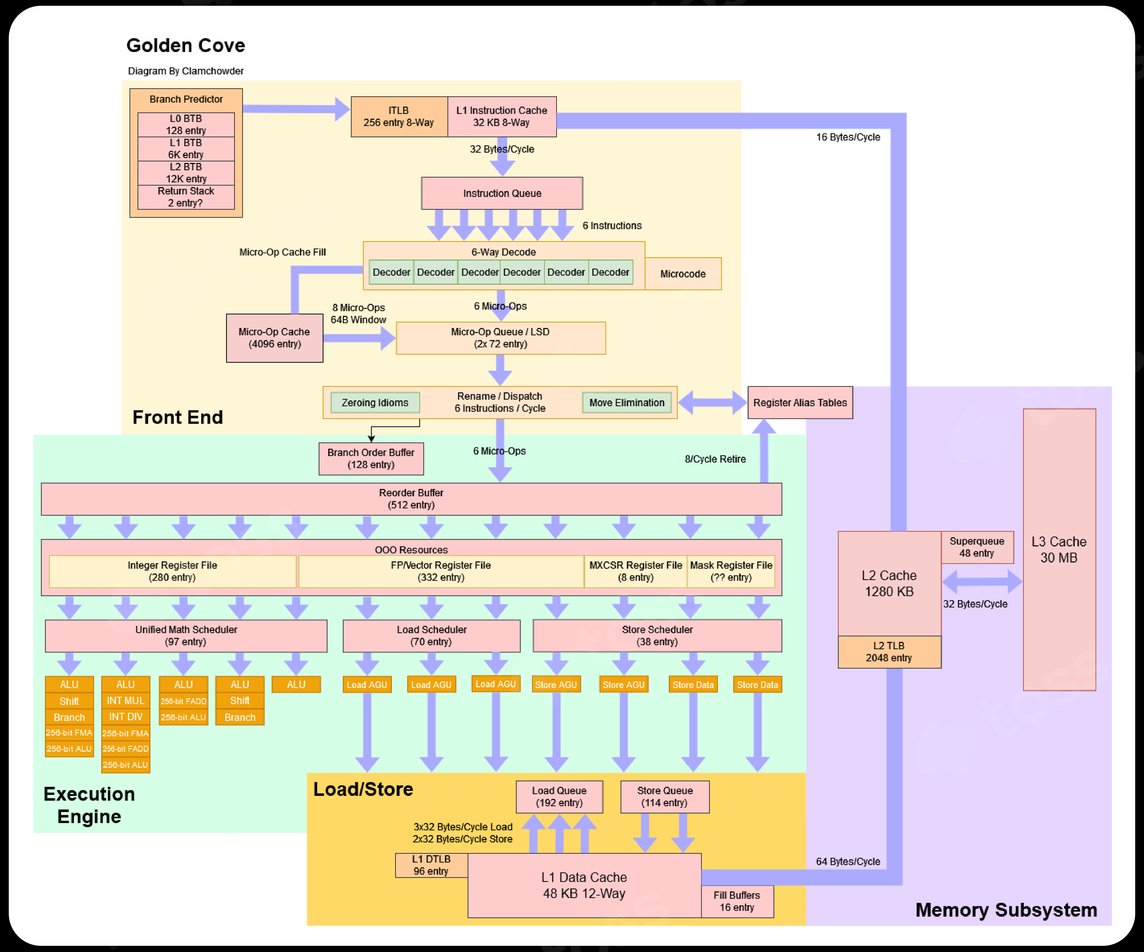
출처 : https://perfmon-events.intel.com/index.html?pltfrm=skylake.html&evnt=CPU_CLK_UNHALTED.THREAD

모니터링

CPU Topdown Analysis

CPU 프론트엔드에서 백엔드까지 performance counter 를 통해 병목 지점을 분석

모니터링



출처 : https://chipsandcheese.com/2021/12/02/popping-the-hood-on-golden-cove/

모니터링

```
# perf stat -a -C 15 -Mtma_L1_group -euops_executed.stall_cycles  taskset -c 15 ./test

Performance counter stats for 'system wide':

364,335,968      uops_executed.stall_cycles                                  (35.71%)
1,611,154,226      CPU_CLK_UNHALTED.REF_XCLK    # 618997959839.45 SLOTS
                                                #   67.9 %  tma_retiring
                                                #    0.0 %  tma_bad_speculation       (21.43%)
24,273,540       INT_MISC.RECOVERY_CYCLES_ANY                                (21.43%)
1,611,583,468      CPU_CLK_UNHALTED.ONE_THREAD_ACTIVE                        (21.43%)
154,728,878,608      CPU_CLK_UNHALTED.THREAD                                 (28.57%)
420,425,669,644      UOPS_RETIRED.RETIRE_SLOTS                               (35.72%)
420,444,801,462      UOPS_ISSUED.ANY                                         (35.72%)
1,612,182,102      CPU_CLK_UNHALTED.REF_XCLK    #    0.1 %  tma_frontend_bound
                                                #   31.9 %  tma_backend_bound       (35.72%)
900,131,965       IDQ_UOPS_NOT_DELIVERED.CORE                                (35.72%)
23,508,714       INT_MISC.RECOVERY_CYCLES_ANY                                (28.57%)
1,611,951,671      CPU_CLK_UNHALTED.ONE_THREAD_ACTIVE                        (28.57%)
154,773,425,775      CPU_CLK_UNHALTED.THREAD                                 (35.71%)
420,647,011,006      UOPS_ISSUED.ANY                                         (35.71%)
430,034,883,695      INST_RETIRED.ANY          # 430034883695.00 Instructions
                                                #   2.78 CoreIPC                 (21.43%)
1,611,366,729      CPU_CLK_UNHALTED.REF_XCLK                                 (21.43%)
1,611,279,125      CPU_CLK_UNHALTED.ONE_THREAD_ACTIVE                        (21.43%)
154,691,221,053      CPU_CLK_UNHALTED.THREAD                                 (21.43%)

64.955153257 seconds time elapsed
```

모니터링

```
BE/Core     Backend_Bound.Core_Bound.Ports_Utilization.Ports_Utilized_3m.ALU_Op_Utilization          % Core_Execution    50.2
BE/Core     Backend_Bound.Core_Bound.Ports_Utilization.Ports_Utilized_3m.Load_Op_Utilization.Port_2  % Core_Clocks       65.2
BE/Core     Backend_Bound.Core_Bound.Ports_Utilization.Ports_Utilized_3m.Load_Op_Utilization.Port_3  % Core_Clocks       66.3
BE/Core     Backend_Bound.Core_Bound.Ports_Utilization.Ports_Utilized_3m.Store_Op_Utilization        % Core_Execution    93.9
BE/Core     Backend_Bound.Core_Bound.Ports_Utilization.Ports_Utilized_3m.Store_Op_Utilization.Port_4 % Core_Clocks       93.9
RET         Retiring.Light_Operations.Memory_Operations                                            % Slots            42.5
```

모니터링

```
# perf stat -a -C 7 -Mtma_L1_group -euops_executed.stall_cycles  taskset -c 7 ./test

 Performance counter stats for 'system wide':

    21,576,372,155      uops_executed.stall_cycles                                    (35.71%)
     2,719,105,840      CPU_CLK_UNHALTED.REF_XCLK    # 522313131452.00 SLOTS
                                                     #   80.5 %  tma_retiring
                                                     #    0.1 %  tma_bad_speculation    (21.43%)
        74,869,131      INT_MISC.RECOVERY_CYCLES_ANY                                  (21.43%)
                 0      CPU_CLK_UNHALTED.ONE_THREAD_ACTIVE                            (21.43%)
   261,156,565,726      CPU_CLK_UNHALTED.THREAD                                       (28.57%)
   420,681,760,926      UOPS_RETIRED.RETIRE_SLOTS                                     (35.72%)
   420,859,463,825      UOPS_ISSUED.ANY                                               (35.72%)
     2,721,194,985      CPU_CLK_UNHALTED.REF_XCLK    #   18.7 %  tma_frontend_bound
                                                     #    0.7 %  tma_backend_bound      (35.72%)
    97,860,876,494      IDQ_UOPS_NOT_DELIVERED.CORE                                   (35.72%)
        75,907,506      INT_MISC.RECOVERY_CYCLES_ANY                                  (28.57%)
                 0      CPU_CLK_UNHALTED.ONE_THREAD_ACTIVE                            (28.57%)
   261,234,803,515      CPU_CLK_UNHALTED.THREAD                                       (35.71%)
   420,933,337,909      UOPS_ISSUED.ANY                                               (35.71%)
   430,198,407,698      INST_RETIRED.ANY             # 430198407698.00 Instructions
                                                     #   3.30 CoreIPC                (21.43%)
     2,719,504,896      CPU_CLK_UNHALTED.REF_XCLK                                     (21.43%)
                 0      CPU_CLK_UNHALTED.ONE_THREAD_ACTIVE                            (21.43%)
   261,072,462,886      CPU_CLK_UNHALTED.THREAD                                       (21.43%)

   109.840303999 seconds time elapsed
```

모니터링

모니터링

```
{
        "BriefDescription": "Probability of Core Bound bottleneck hidden by SMT-profiling artifacts",
        "MetricExpr": "100 * ( 1 - ( ( 1 - ( IDQ_UOPS_NOT_DELIVERED.CORE…",
        "MetricGroup": "Cor;SMT",
        "MetricName": "tma_info_botlnk_core_bound_likely"
    },
```

Hyperthread로 인해 백엔드 병목 현상이 가려질 수 있다

모니터링

```
# perf stat -ddd -a -A --percore-show-thread -MSMT -C 15 taskset -c 15 ./test

 Performance counter stats for 'system wide':

CPU15       2,722,070,704      CPU_CLK_UNHALTED.REF_XCLK        # 1305410274441.50 CORE_CLKS
CPU15                                                            #     0.98 Core_Bound_Likely       (15.39%)
CPU15         128,281,290      EXE_ACTIVITY.EXE_BOUND_0_PORTS                                      (15.39%)
CPU15      94,162,481,359      IDQ_UOPS_NOT_DELIVERED.CORE                                         (15.39%)
CPU15           4,226,412      EXE_ACTIVITY.BOUND_ON_STORES                                        (15.39%)
CPU15      62,606,826,579      EXE_ACTIVITY.1_PORTS_UTIL                                           (15.39%)
CPU15          75,813,583      INT_MISC.RECOVERY_CYCLES_ANY                                        (15.39%)
CPU15       2,720,254,455      CPU_CLK_UNHALTED.REF_XCLK_ANY    #     1.00 SMT_2T_Utilization      (15.39%)
CPU15                   0      CPU_CLK_UNHALTED.ONE_THREAD_ACTIVE                                    (15.39%)
CPU15     261,082,054,883      CPU_CLK_UNHALTED.THREAD                                             (19.23%)
CPU15     420,519,490,221      UOPS_RETIRED.RETIRE_SLOTS                                           (19.23%)
CPU15      21,118,437,490      CYCLE_ACTIVITY.STALLS_MEM_ANY                                       (19.23%)
CPU15      80,208,279,784      EXE_ACTIVITY.2_PORTS_UTIL                                           (19.23%)
CPU15      21,671,886,837      CYCLE_ACTIVITY.STALLS_TOTAL                                         (19.23%)
CPU15     420,853,102,333      UOPS_ISSUED.ANY                                                     (15.38%)
CPU15          16,006,164      ARITH.DIVIDER_ACTIVE                                                (15.38%)
CPU15     430,318,900,660      INST_RETIRED.ANY                 #     3.30 CoreIPC                 (11.54%)
CPU15       2,720,475,382      CPU_CLK_UNHALTED.REF_XCLK                                           (11.54%)
CPU15                   0      CPU_CLK_UNHALTED.ONE_THREAD_ACTIVE                                    (11.54%)
CPU15     261,165,670,208      CPU_CLK_UNHALTED.THREAD                                             (11.54%)
CPU15     560,572,502,189      UOPS_EXECUTED.THREAD             #     2.34 Execute                 (7.69%)
CPU15     239,364,849,044      cpu/UOPS_EXECUTED.THREAD,cmask=1/                                     (7.69%)
CPU15     140,096,870,707      L1-dcache-loads                                                     (7.69%)
CPU15          16,733,177      L1-dcache-load-misses            #     0.01% of all L1-dcache accesses  (11.54%)
CPU15             833,275      LLC-loads                                                           (11.54%)
CPU15               2,534      LLC-load-misses                  #     0.30% of all LL-cache accesses   (15.39%)
CPU15     <not supported>      L1-icache-loads
CPU15          25,818,357      L1-icache-load-misses            #     0.00% of all L1-icache accesses  (15.39%)
CPU15     140,158,263,939      dTLB-loads                                                          (15.39%)
CPU15             111,366      dTLB-load-misses                 #     0.00% of all dTLB cache accesses (15.39%)
CPU15              52,401      iTLB-loads                                                          (15.39%)
CPU15             215,186      iTLB-load-misses                 #   410.65% of all iTLB cache accesses (15.39%)
CPU15     <not supported>      L1-dcache-prefetches
CPU15     <not supported>      L1-dcache-prefetch-misses

     109.923637381 seconds time elapsed
```

성능 측정

pmu-tools

perf 기반 인텔 CPU 성능 분석 도구

모니터링

```
# ./toplev -l3 --single-thread --nodes '!+Core_Bound*/3,+Backend_Bound,+MUX' taskset -c 15 ./test
# 4.7-full on Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz [skx/skylake]
BE               Backend_Bound             % Slots                       31.9   [16.0%]
BE/Core          Backend_Bound.Core_Bound  % Slots                       31.6   [16.0%]<==
This metric represents fraction of slots where Core non-
memory issues were of a bottleneck...
MUX                                        %                             16.00
Run toplev --describe Backend_Bound.Core_Bound^ to get more information on bottleneck
Add --run-sample to find locations
```

--single-thread

모니터링

```
# ./toplev --single-thread -l1 -C 7,15 taskset -c 7 ./test
# 4.7-full on Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz [skx/skylake]
BE               Backend_Bound  % Slots                          50.5   [50.0%]<==
This category represents fraction of slots where no uops are
being delivered due to a lack of required resources for
accepting new uops in the Backend...
MUX                             %                                50.00
Run toplev --describe Backend_Bound^ to get more information on bottleneck
Add --run-sample to find locations
Add --nodes '!+Backend_Bound*/2,+MUX' for breakdown.
```

--single-thread

모니터링

Elasticsearch 에서 CPU Bound 를 보면 되겠다!

모니터링

```
# 4.7-full on Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz [clx/skylake]
5.015660735 FE              Frontend_Bound.Fetch_Latency.MS_Switches       % Clocks_est            1.0   [ 5.9%]
5.015660735 RET             Retiring.Heavy_Operations.Microcode_Sequencer  % Slots                 1.0   [ 5.9%]
5.015660735 BAD             Bad_Speculation.Machine_Clears                 % Slots                 0.0   [ 5.9%]
5.015660735 BE/Mem          Backend_Bound.Memory_Bound                     % Slots                21.3   [ 5.9%]
5.015660735 BE/Core         Backend_Bound.Core_Bound                       % Slots                20.3   [ 5.9%]
5.015660735 BE              Backend_Bound                                  % Slots                41.5   [ 5.9%]<==
```

Hyperthread off

모니터링

```
# 4.7-full on Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz [clx/skylake]
5.054301612 S0-C0     FE               Frontend_Bound                % Slots                    22.2   [ 1.6%]
5.054301612 S0-C0     BE               Backend_Bound                 % Slots                    20.8   [ 1.6%]
5.054301612 S0-C0     FE               Frontend_Bound.Fetch_Latency  % Slots                    13.3   [ 1.6%]
5.054301612 S0-C0     BE/Core          Backend_Bound.Core_Bound      % Slots                    10.4   [ 1.6%]
```

Hyperthread on

모니터링

- **pcm Grafana dashboard** : front-end for Grafana (in scripts/grafana directory). Full Grafana Readme is here

- **pcm-sensor** : front-end for KDE KSysGuard
- **pcm-service** : front-end for Windows perfmon

출처 : https://github.com/intel/pcm

PCM exporter

모니터링

ebpf_exporter

Prometheus exporter for custom eBPF metrics and OpenTelemetry traces.

- Metrics:

출처 : https://github.com/cloudflare/ebpf_exporter

EBPF exporter

결론

하이퍼스레드 켜면 (대부분) 좋다

사용중인 워크로드의 특성을 잘 알아야 한다

SLASH 24