Characterize the Performence of cache and TLB

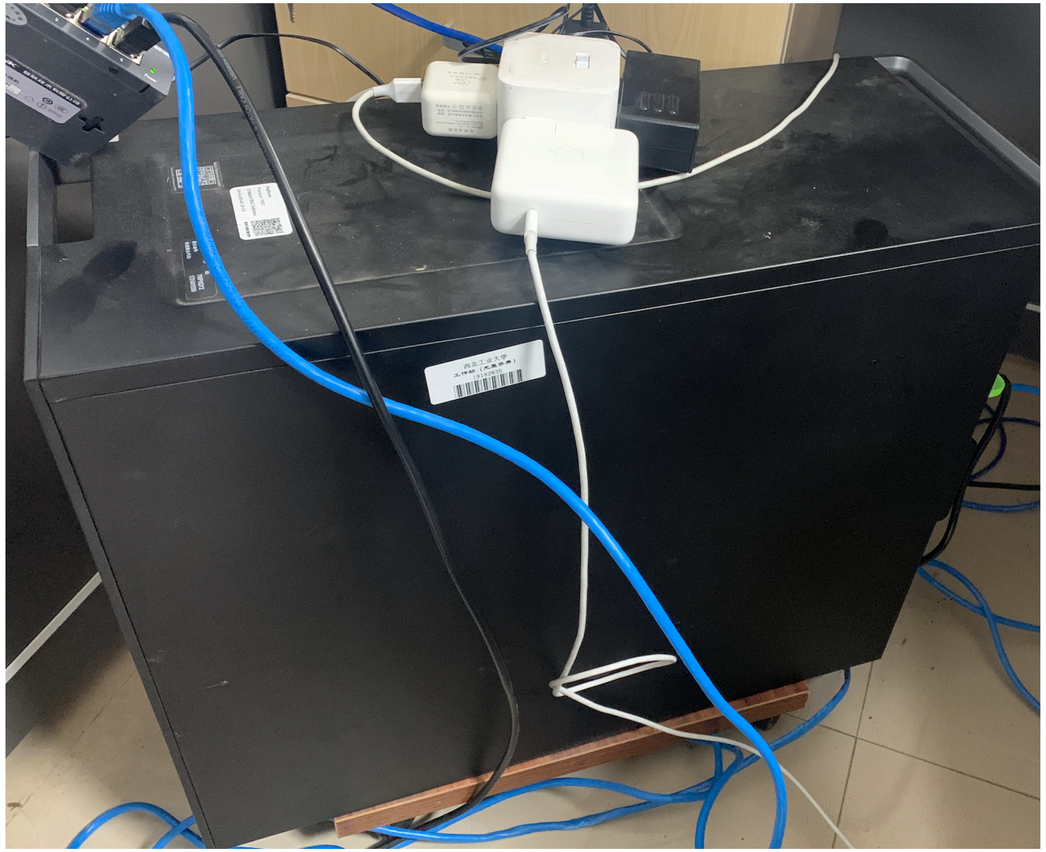
袁浩程
圣贤

2021/11/22

Problem Statement

Measure cache and TLB

- I have a computer



- I want to know the cpu's....

	Cache Parameters	*TLB Parameters*
cache size	?	?
associativity	?	?
line size	?	?
miss penalty	?	?

Experimental Methodology

Basic idea

- a: associatity
- b: line size
- D: cache size
- M: miss penalty
- N: array size
- S: stride

- Compute many times a simple floating-point function on a subset of elements taken from an array.
- This subset is given by
  $1, s+1, 2s+1, \ldots, N-s+1 (s < N/2)$
- Each iteration the cache gets a read request, time differs when miss does/doesn't happen.

Experimental Methodology

Four regimes

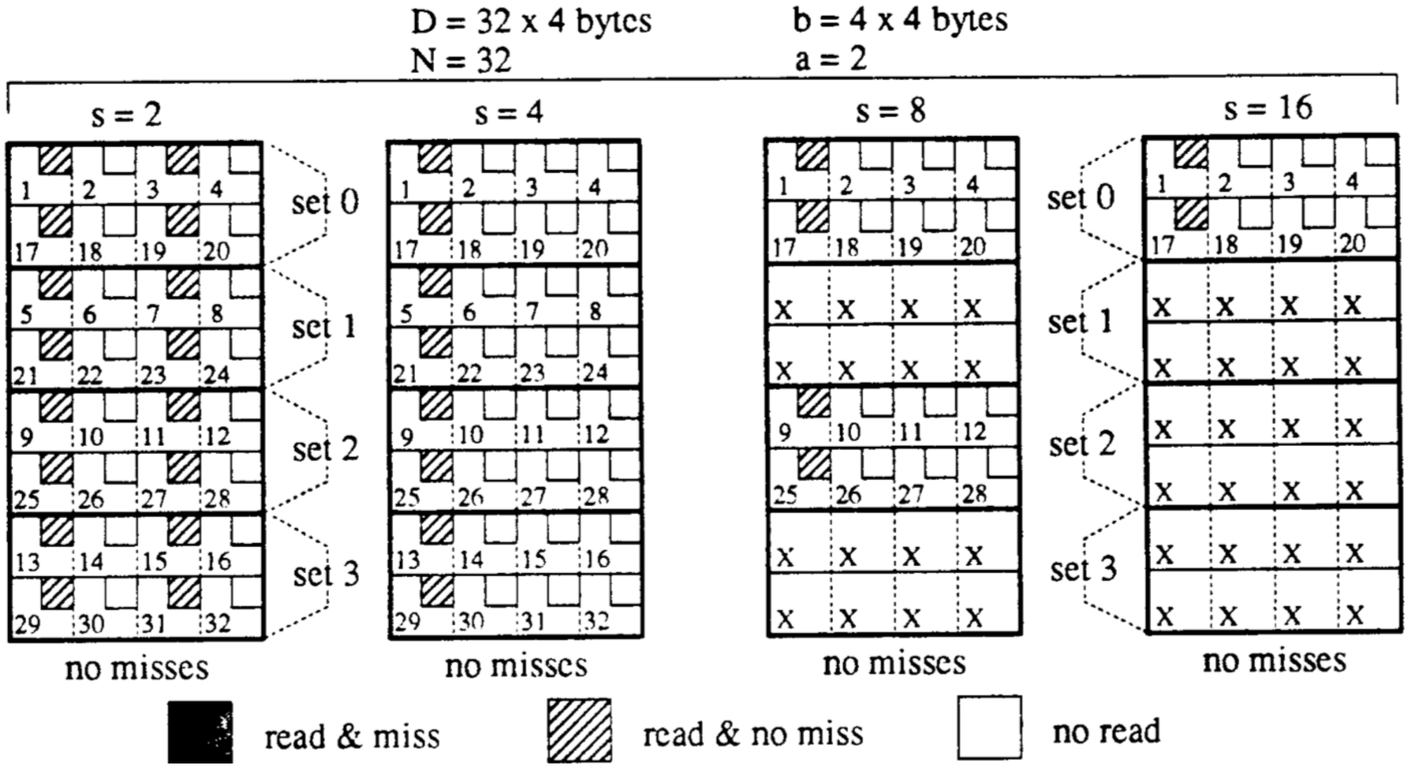
- a: associatity
- b: line size
- D: cache size
- M: miss penalty
- N: array size
- S: stride

Regime	*Size of Array*	*Stride*	*Frequency of Misses*	*Time per Iteration*
1	1<N<D	1< s <N/2	no misses	T_no_miss
2.a	D<N	1<= s <b	one miss every b/s elements	T_no_miss+ Ms/b
2.b	D<N	b<= s <N/a	one miss every element	T_no_miss+ M
2.c	D<N	N/a<= s <=N/2	no misses	T_no_miss

Experimental Methodology

Four regimes

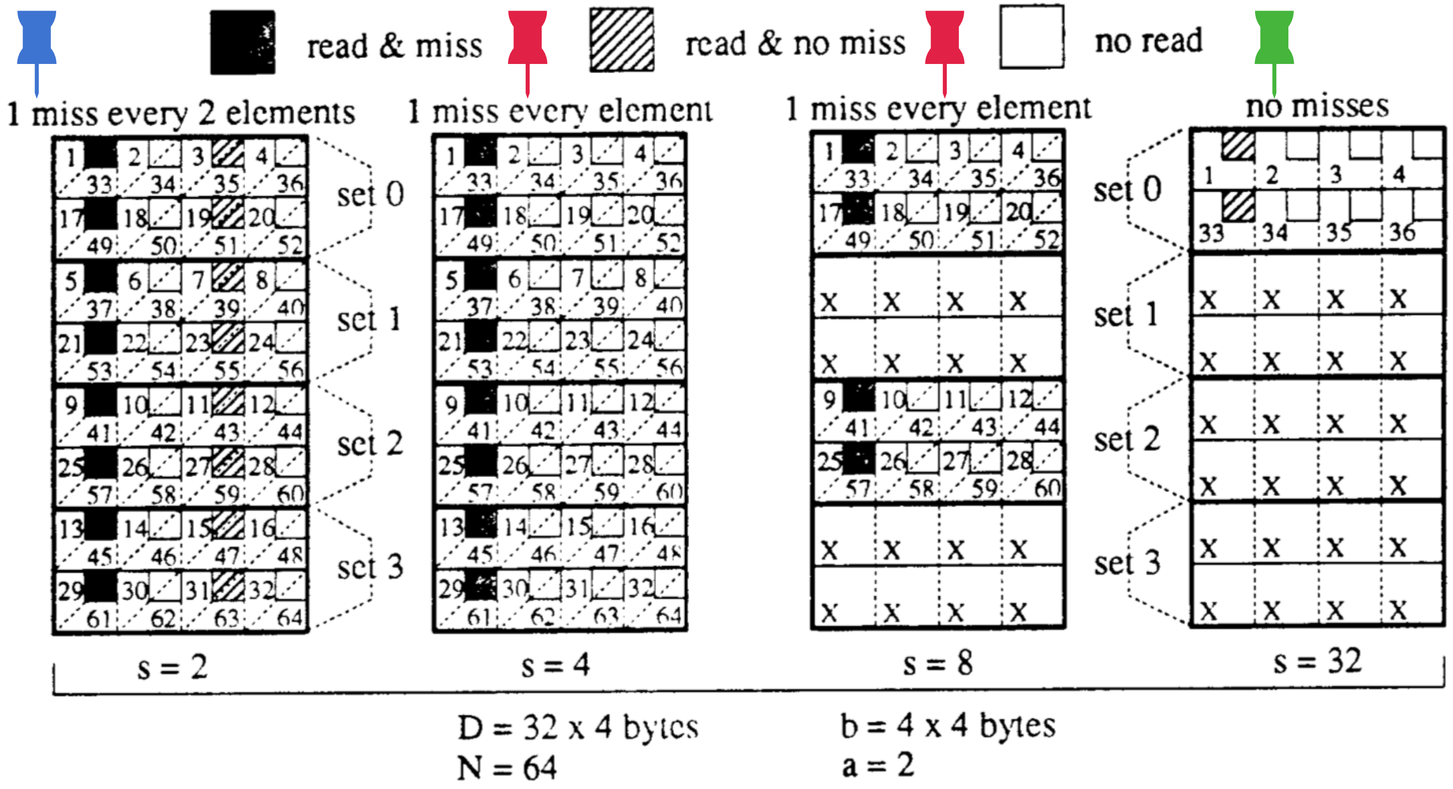
Regime	*Size of Array*	*Stride*	*Frequency of*	*Time per Iteration*
1	1<N<D	1< s <N/2	no	T
2.a	D<N	1<= s <b	one every b/	T+Ms/b
2.b	D<N	b<= s <N/a	one every	T+M
2.c	D<N	N/a<= s <=N/2	no	T



Experimental Methodology

Four regimes

Regime	*Size of Array*	*Stride*	*Frequency of*	*Time per Iteration*
1	1<N<D	1< s <N/2	no	T
2.a	D<N	1<= s <b	one per b/s	T+Ms/b
2.b	D<N	b<= s <N/a	one	T+M
2.c	D<N	N/a<= s <=N/2	no	T

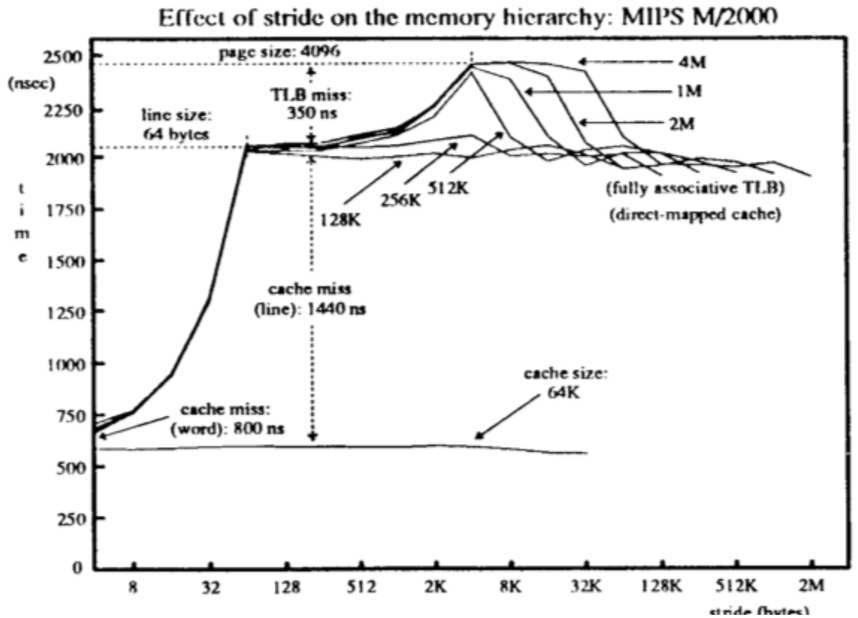


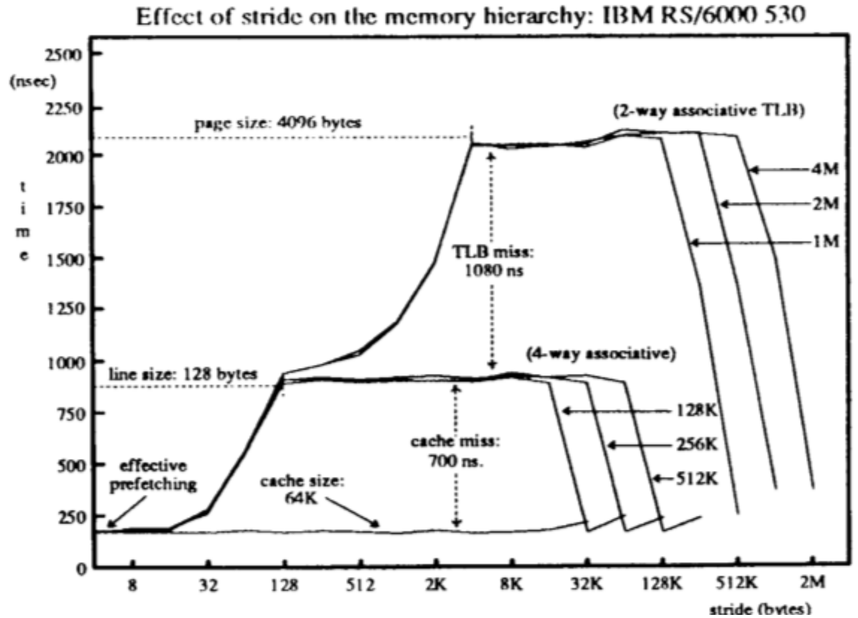
Experimental Methodology

My implementation

- C
  build the main part of the benchmark
- Shell
  change N & S automatically
- Matlab
  draw the graph

- Use benchmark_N to identify the cache size
- Use benchmark_S to get the result

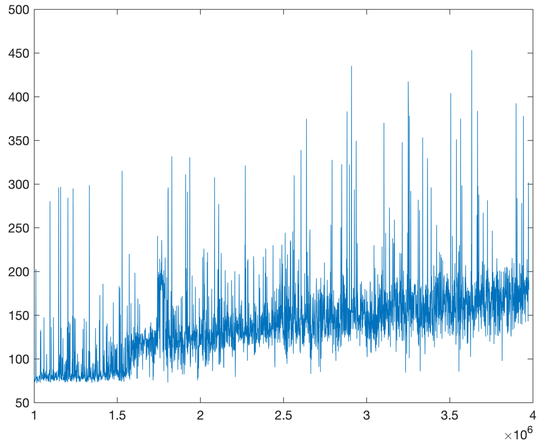


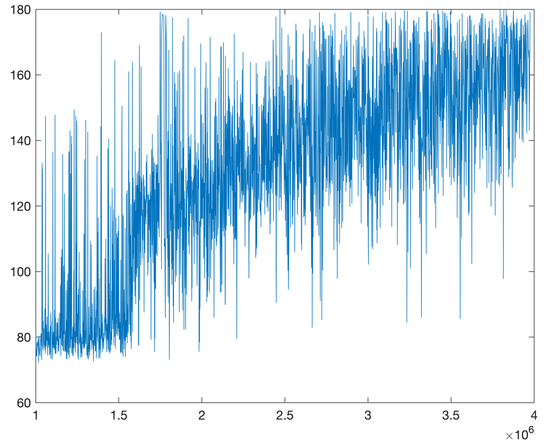


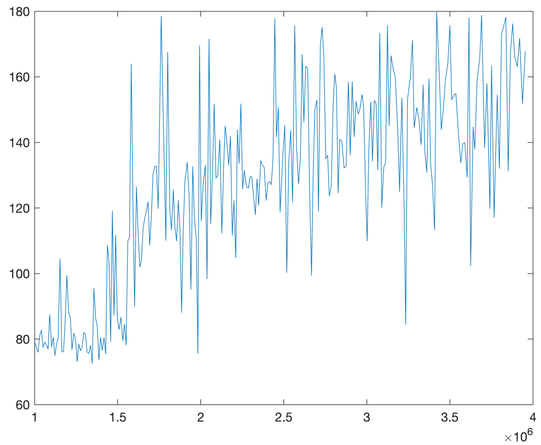
Result

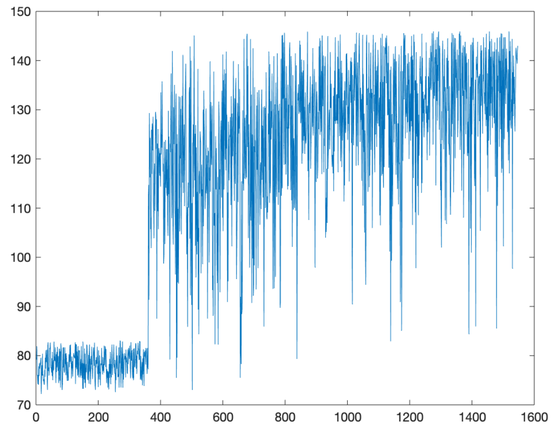
Benchmark_N

- benchmark_n.c
  - *build the main loop*
  - *count the time*
  - *write the result*
- benchmark_n.sh
  - *change N automatically*







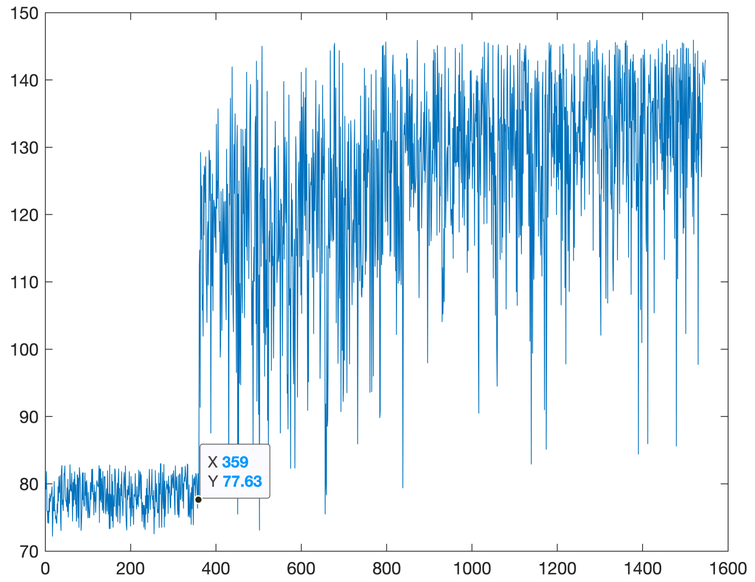


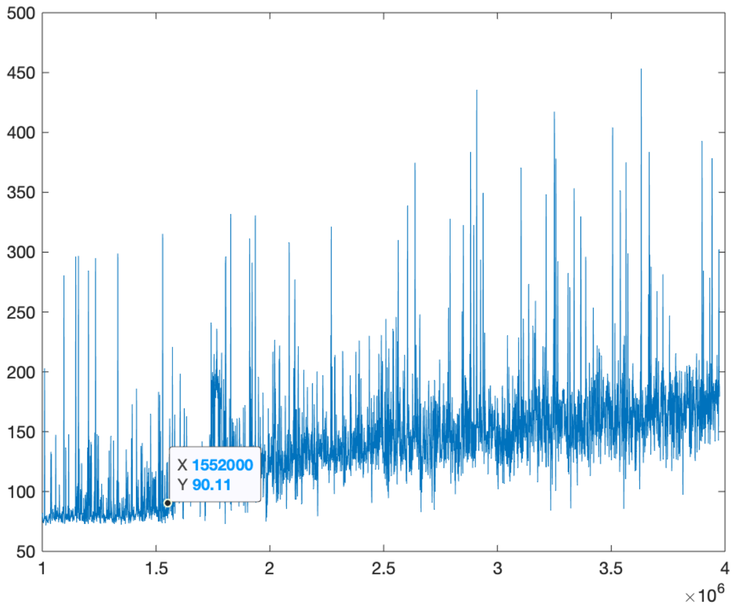
Result

Benchmark_N

- benchmark_n.c
  - *build the main loop*
  - *count the time*
  - *write the result*
- benchmark_n.sh
  - *change N automatically*

- 359 or 400 ?
- 1552000 4-bytes words = 6062KB



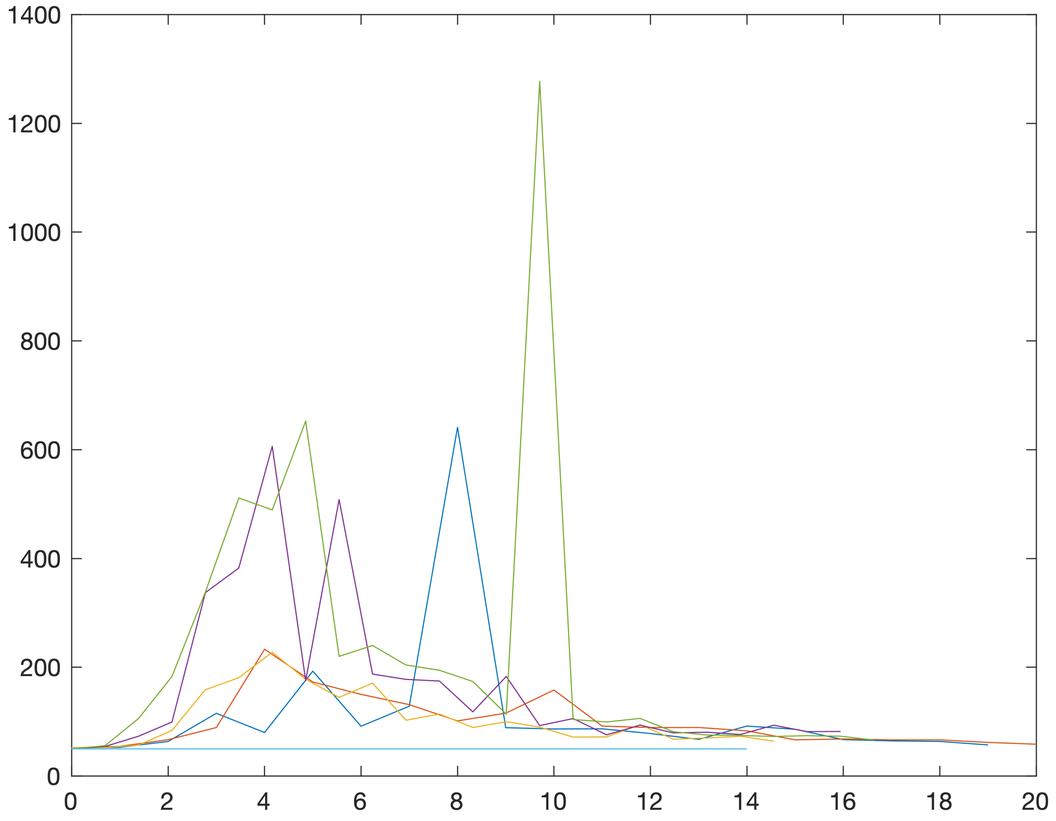


Result

Benchmark_S

- benchmark_s.c
  - *build the main loop*
  - *count the time*
  - *write the result*
- benchmark_n.sh
  - *change N & S automatically*

- s+=s
- log10(s) - time
- Too many noise !
- Hard to identify the s value

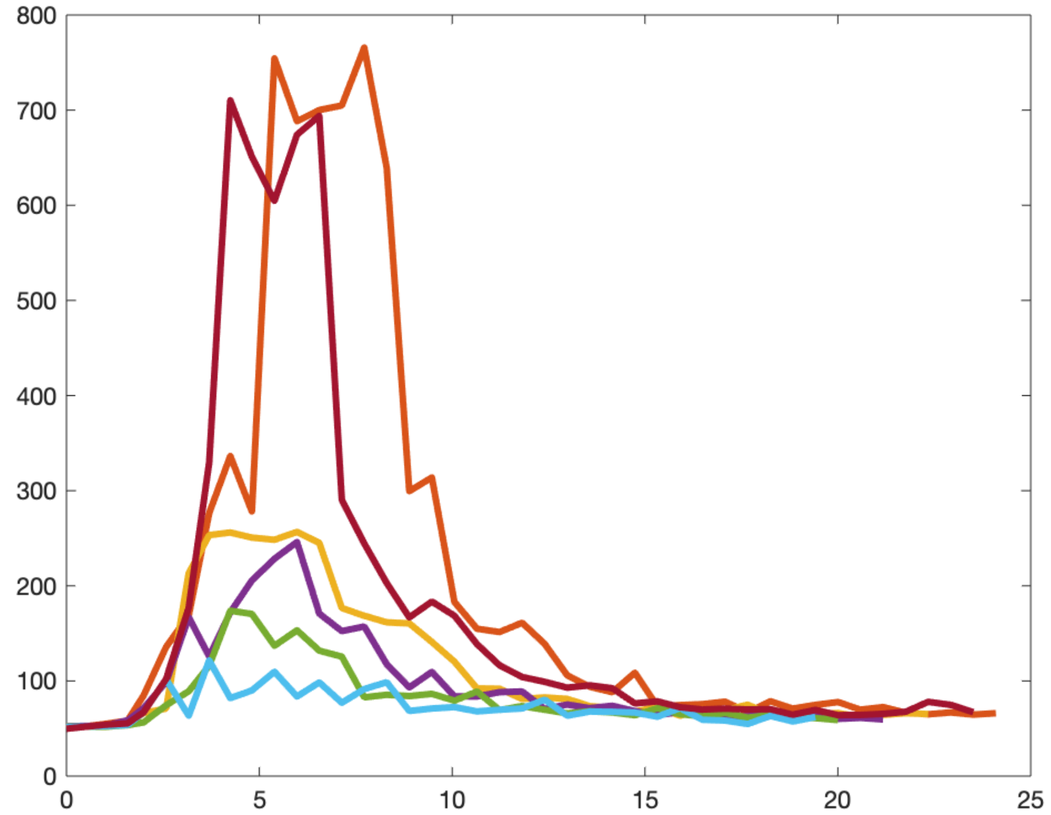


Result

Benchmark_S

- benchmark_s.c
  - *build the main loop*
  - *count the time*
  - *write the result*
- benchmark_n.sh
  - *change N & S automatically*

- s+=s/2
- log2(s) - time
- Remove outlier
- Beautiful~

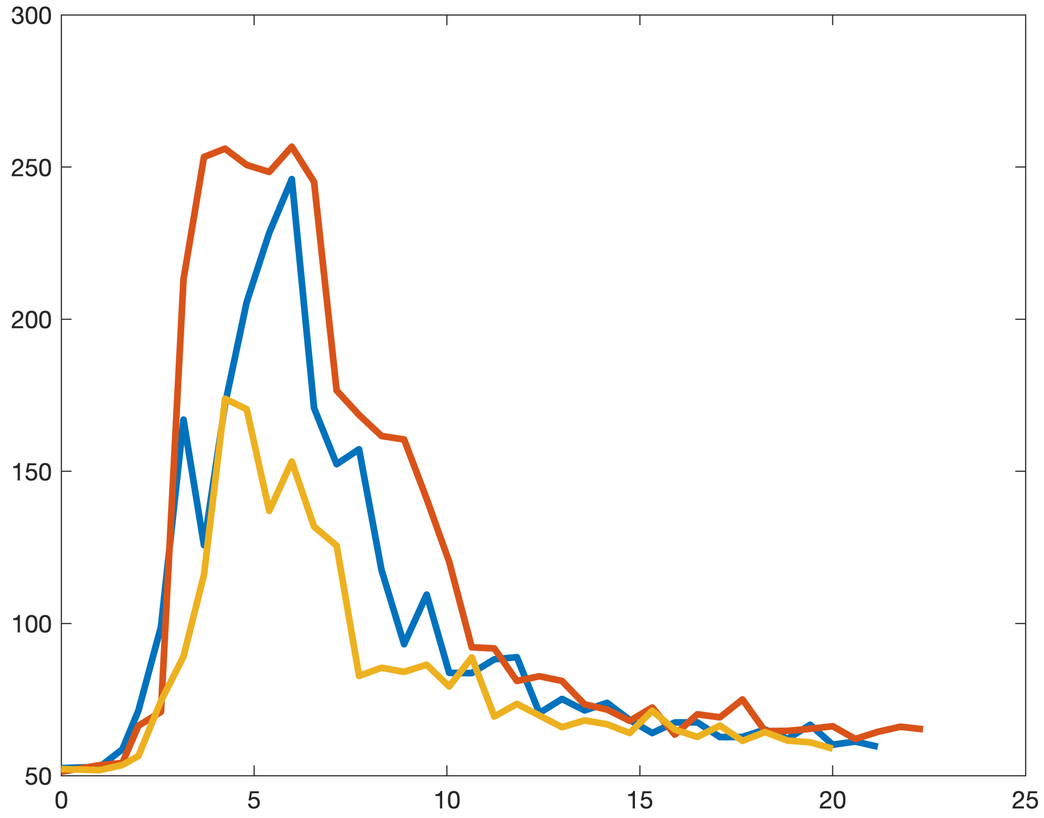


Result

Cache Line Size b

- Regime 2.*a*
- Line size b is the s value when the time value goes to maximum (under a relative small N)

- N = 2/4/8 D
- log2(s) - time
- B = 32 * 4byte
  = 128 bytes

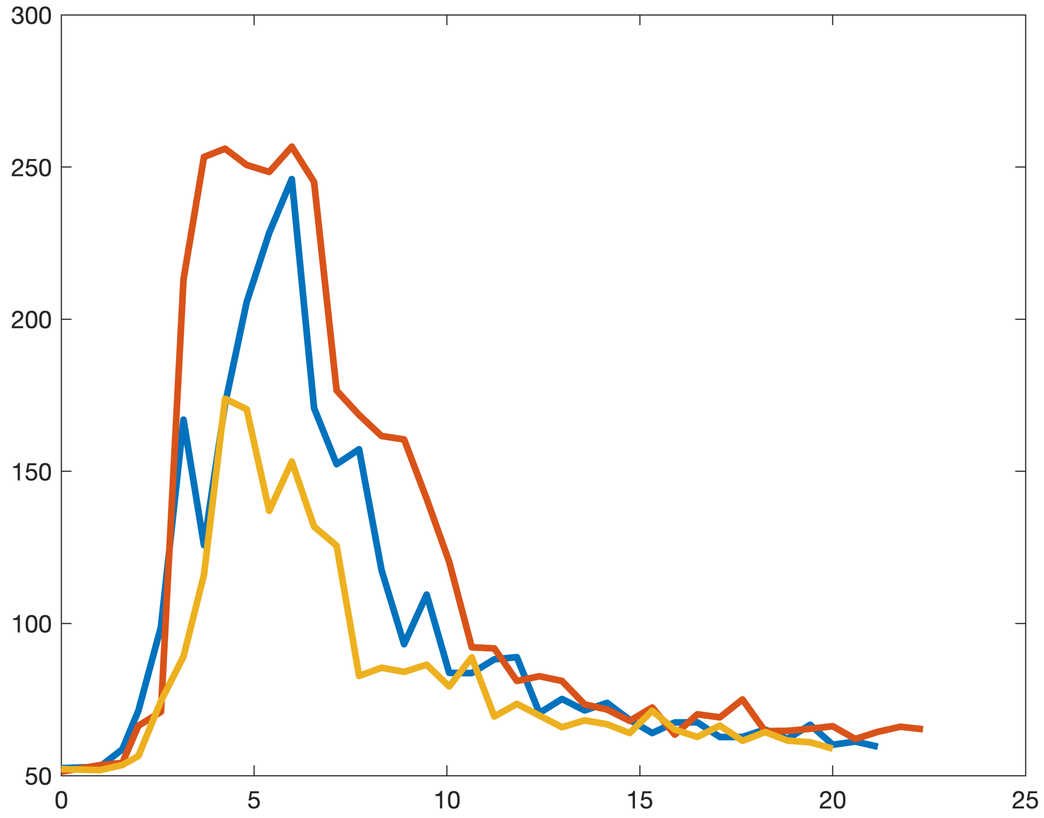


Result

Cache Associativity a

- Regime 2.*b*
- Cache associativity b is the N/s value when the time value drop to minimum (under a relative small N)

- N = 2/4/8 D
- log2(s) - time
- a = N//s_a
  = 4

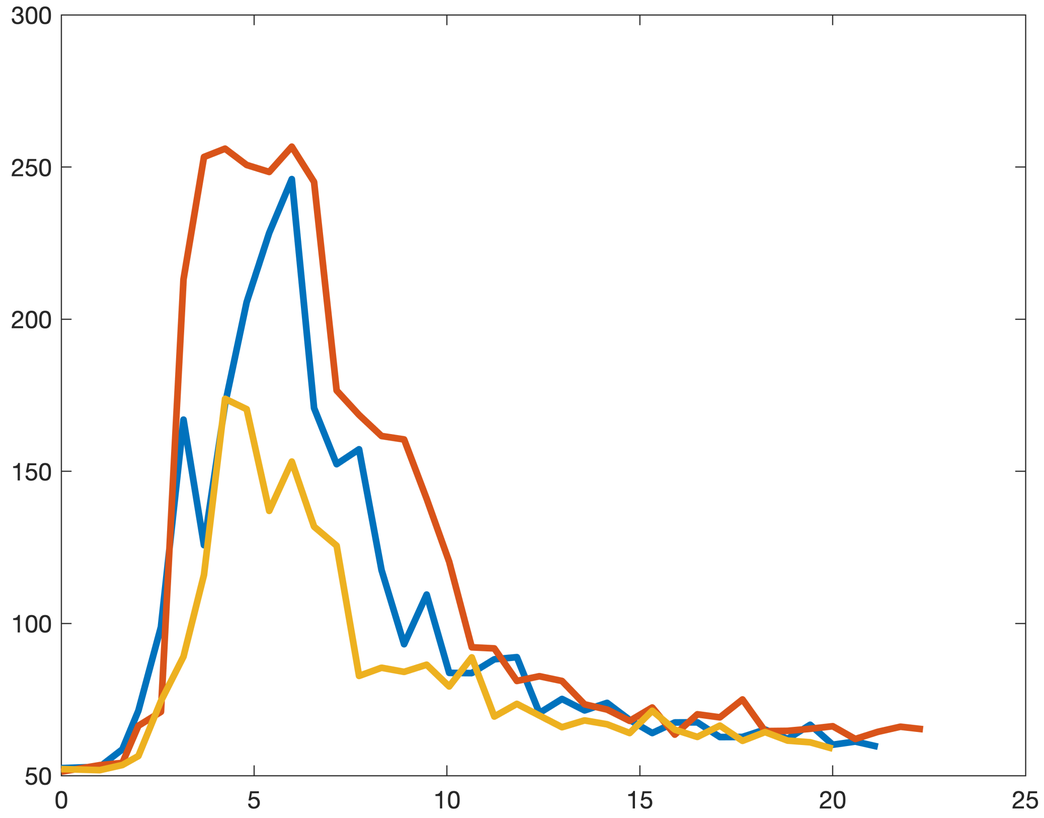


Result

Cache Miss penalty M

- Regime 2.a
- M is
  the
  time_n - time_0
  value
  when
  the time value
  goes to maximum
  (under a relative
  small N)

- N = 2/4/8 D
- log2(s) - time
- M = 200 - 70
  = 130 ns

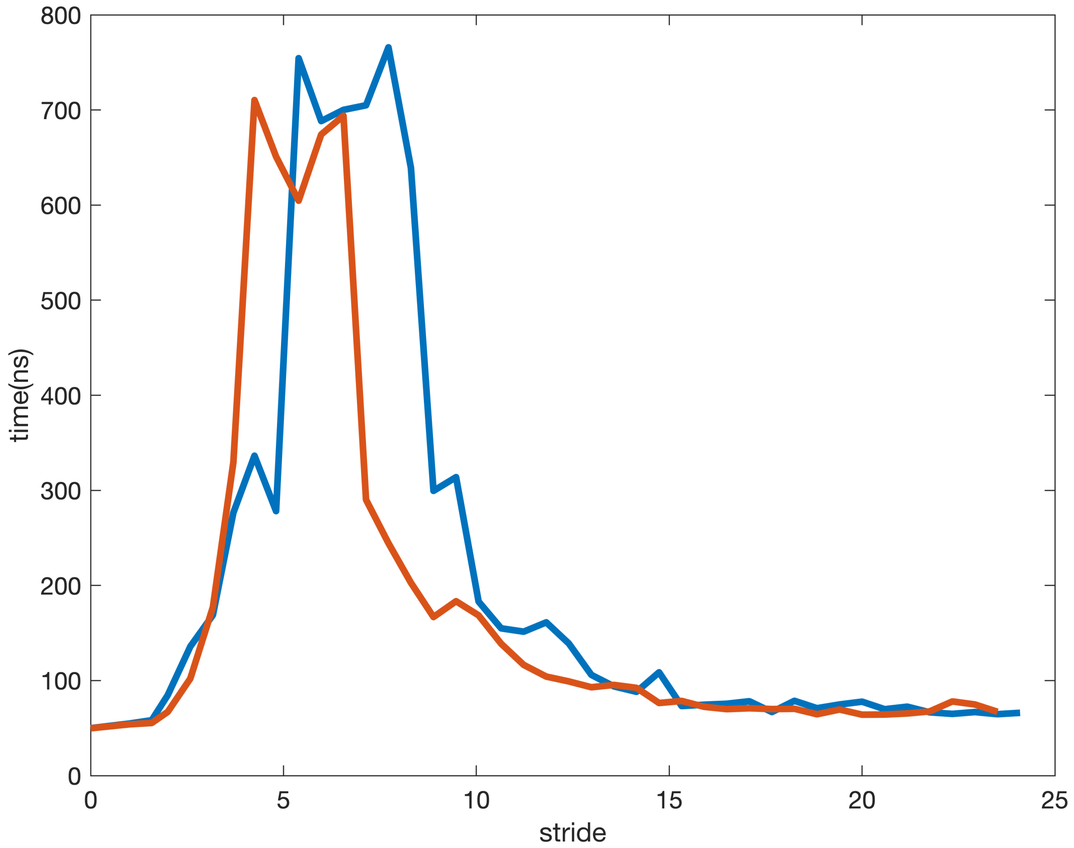


Result

Page Size p

- Regime 2.*a*
- Page size p is the s value when the time value goes to maximum (under a relative large N)

- N = 16/32 D
- log2(s) - time
- p = 1024 * 4byte
  = 4K bytes



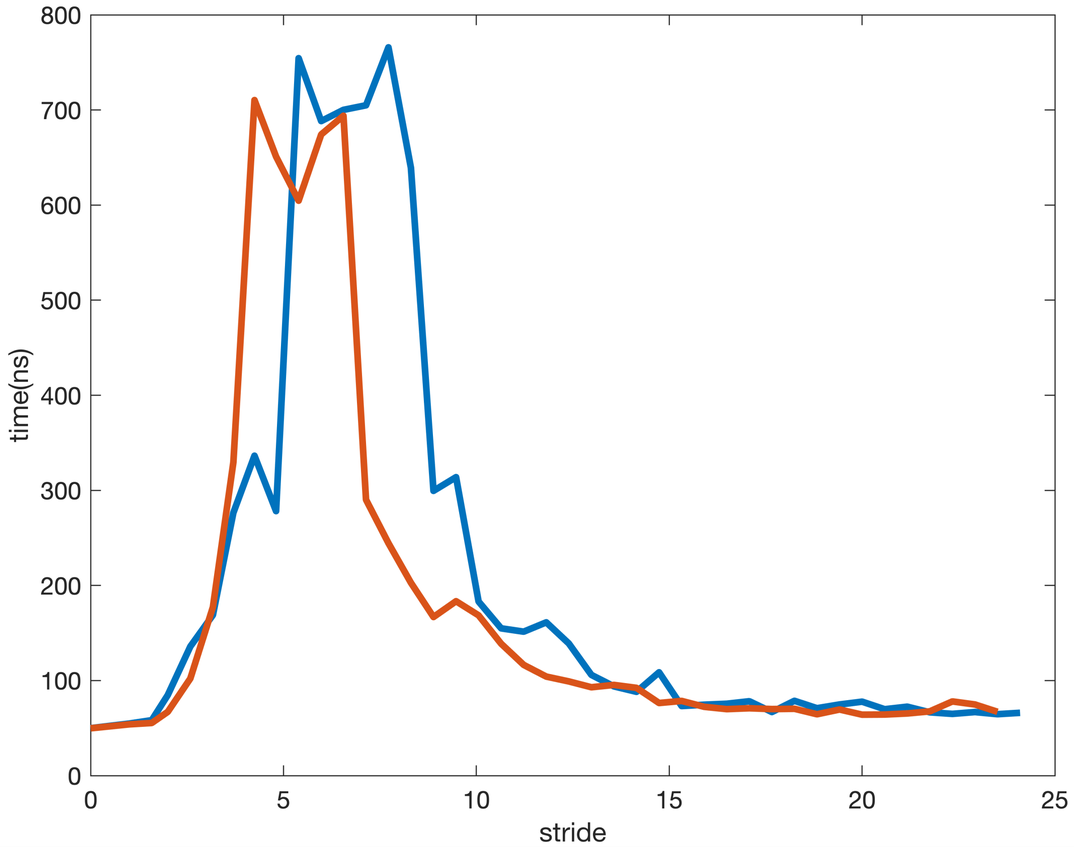
Result

TLB Associativity tlb_a

- Regime 2.*b*
- TLB associativity tlb_a is the N/s value when the time value drop to minimum

  (under a relative large N)

- N = 1/2/4/8 D
- log2(s) - time
- tlb_a = N//s_a
  = 4

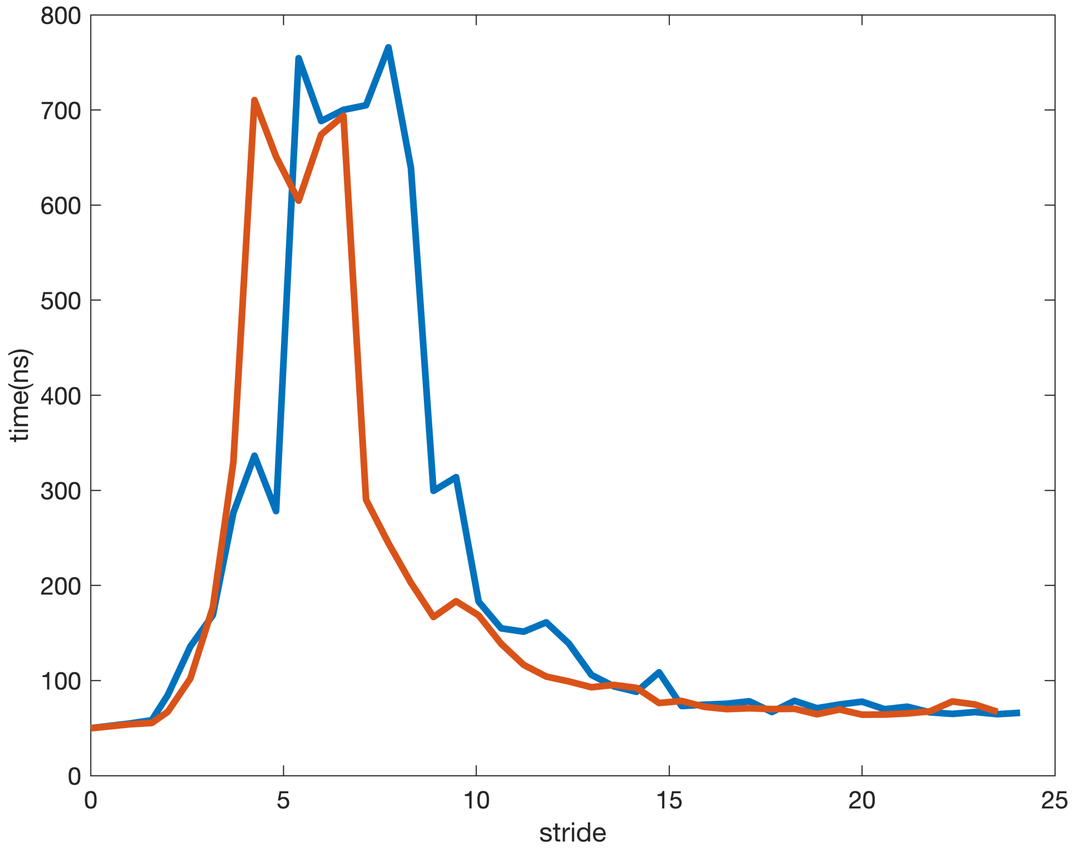


Result

TLB Miss penalty tlb_M

- Regime 2.a
- tlb_M is
  the
  time_n' - time_n
  value
  when
  the time value
  goes to maximum
  (under a relative
  smaller N)

- N = 1/2/4/8 D
- log2(s) - time
- M = 700 - 130
  = 570 ns



Result

Conclusion

- I want to know the cpu's....

	Cache Parameters	*TLB Parameters*
cache size	6062KB	4KB
associativity	4	4
line size	128bytes	128bytes
miss penalty	130ns	570ns

- Check the result

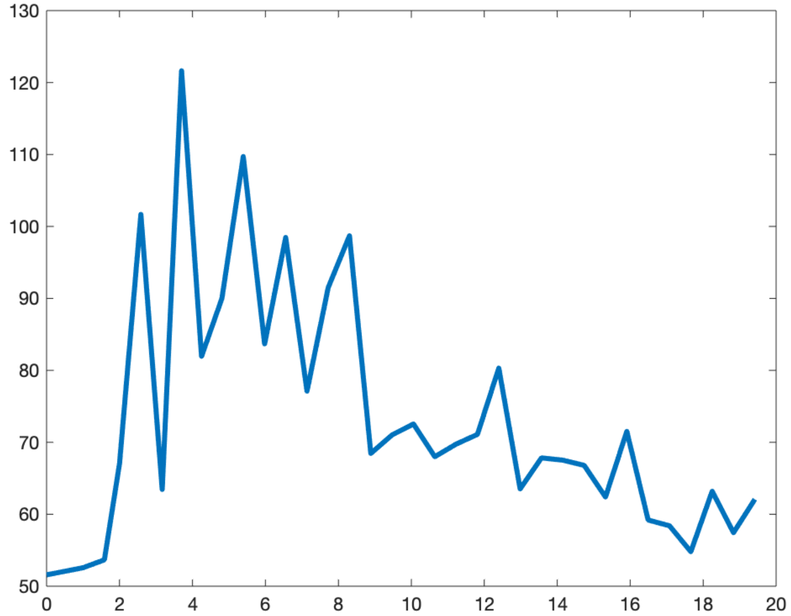
```
(base) yhc@yhc-Precision-7920-Tower:~/Desktop/os/cachesize$ cpuid | grep -i tlb
   cache and TLB information (2):
      0x63: data TLB: 2M/4M pages, 4-way, 32 entries
            data TLB: 1G pages, 4-way, 4 entries
      0x03: data TLB: 4K pages, 4-way, 64 entries
      0x76: instruction TLB: 2M/4M pages, fully, 8 entries
      0xb6: instruction TLB: 4K, 8-way, 128 entries
      0xc3: L2 TLB: 4K/2M pages, 6-way, 1536 entries
   L1 TLB/cache information: 2M/4M pages & L1 TLB (0x80000005/eax):
   L1 TLB/cache information: 4K pages & L1 TLB (0x80000005/ebx):
   L2 TLB/cache information: 2M/4M pages & L2 TLB (0x80000006/eax):
   L2 TLB/cache information: 4K pages & L2 TLB (0x80000006/ebx):
```

```
(base) yhc@yhc-Precision-7920-Tower:~/Desktop/os/cachesize$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 85
model name      : Intel(R) Xeon(R) Bronze 3104 CPU @ 1.70GHz
stepping        : 4
microcode       : 0x2006b06
cpu MHz         : 1700.000
cache size      : 8448 KB
physical id     : 0
siblings        : 6
core id         : 0
cpu cores       : 6
```

Problems.....

Fight with the ‘prefetch’

- Prefetch: fetch more than you want
- In our case:
  - Set stride=1 and get very strange result
- My solution:
  - Use stride=1000(100 may work as well)



Problems.....

How to measure the time

- Which api to use
- clock(): only ms level
- My solution: search online
  - A good api: clock_gettime

- Which measurement to use
- Count average time of each loop
  - Failed when N went bigger.
- Count the time of each iteration
  - Cost a lot.
- My solution: spend some time.....

实验体会

- 实验难度不小
  - 论文非常清晰，但是复现起来困难重重
  - Debug很痛苦，尤其是预取问题刚出现时

- 能力提升很大
  - 理论-实现的过程，非常有趣
  - 锻炼分析问题的思维
  - 感谢老师~

Thanks For Listening!

- Github id：yyyyyhc