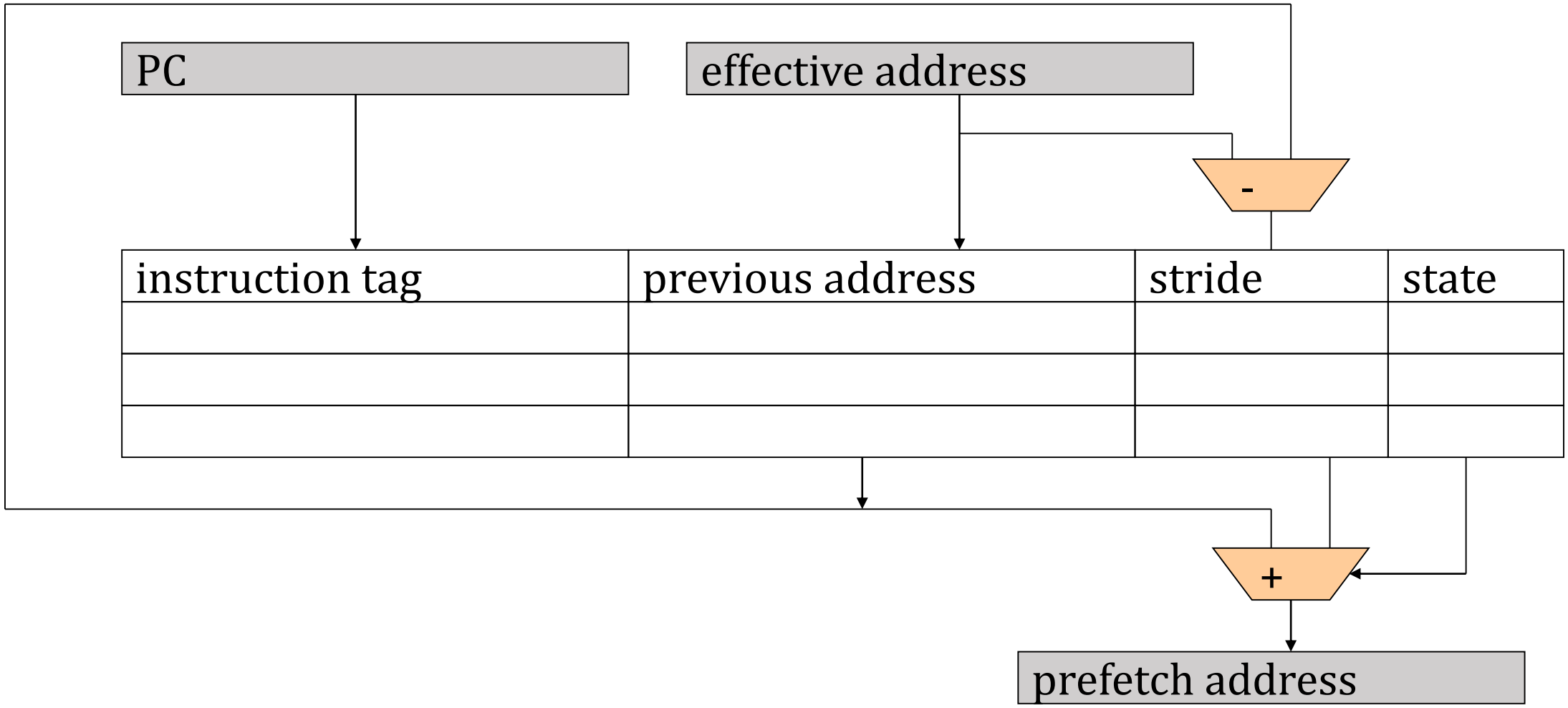
CASPER

CS230: Digital Logic Design and Computer Architecture

Lecture 21: Multicore-Caches

https://www.cse.iitb.ac.in/~biswa/courses/CS230/autumn23/main.html

IP-stride prefetcher



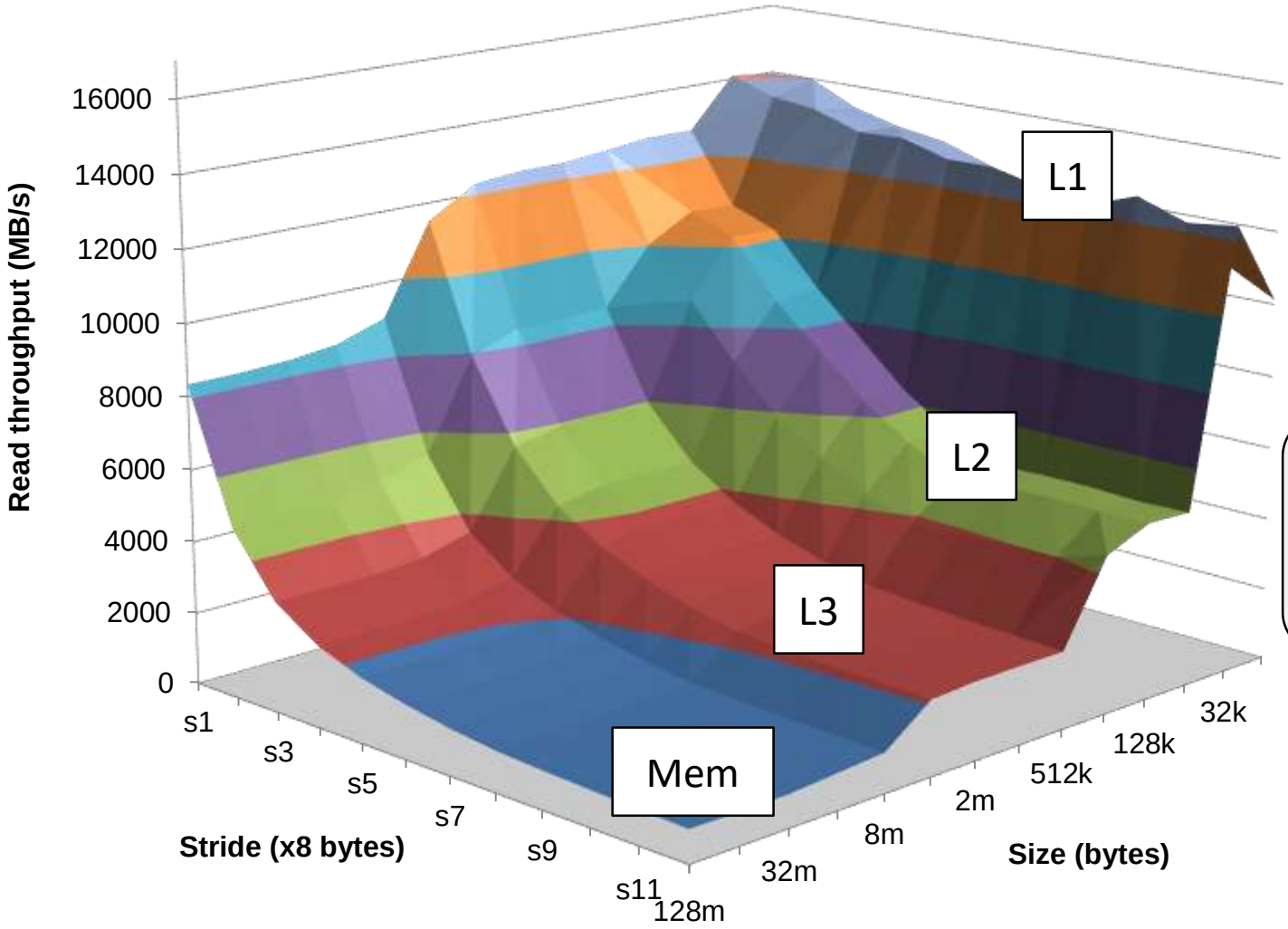
Metrics of interest

Accuracy

Coverage

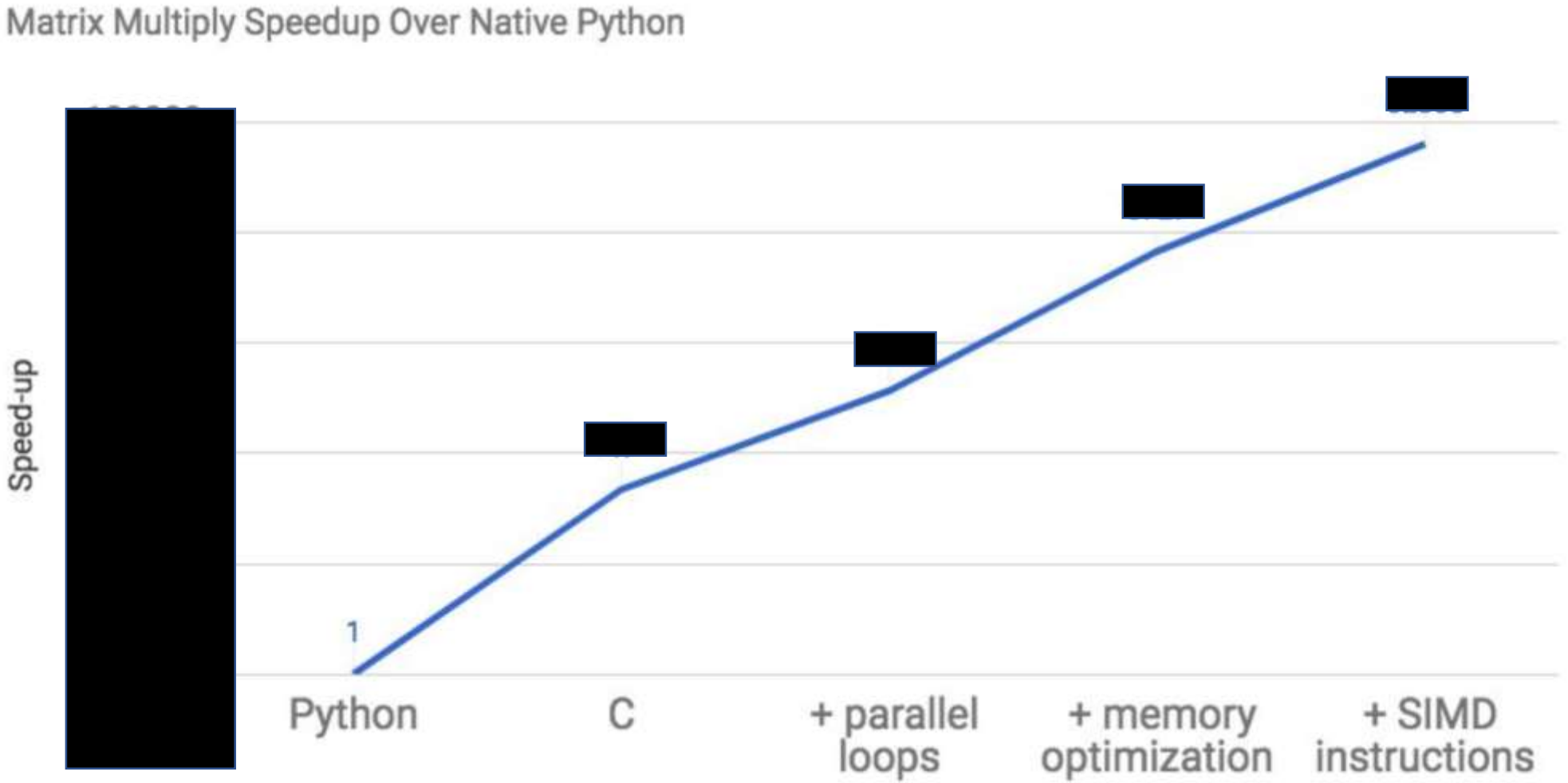
Timeliness

Memory Mountain

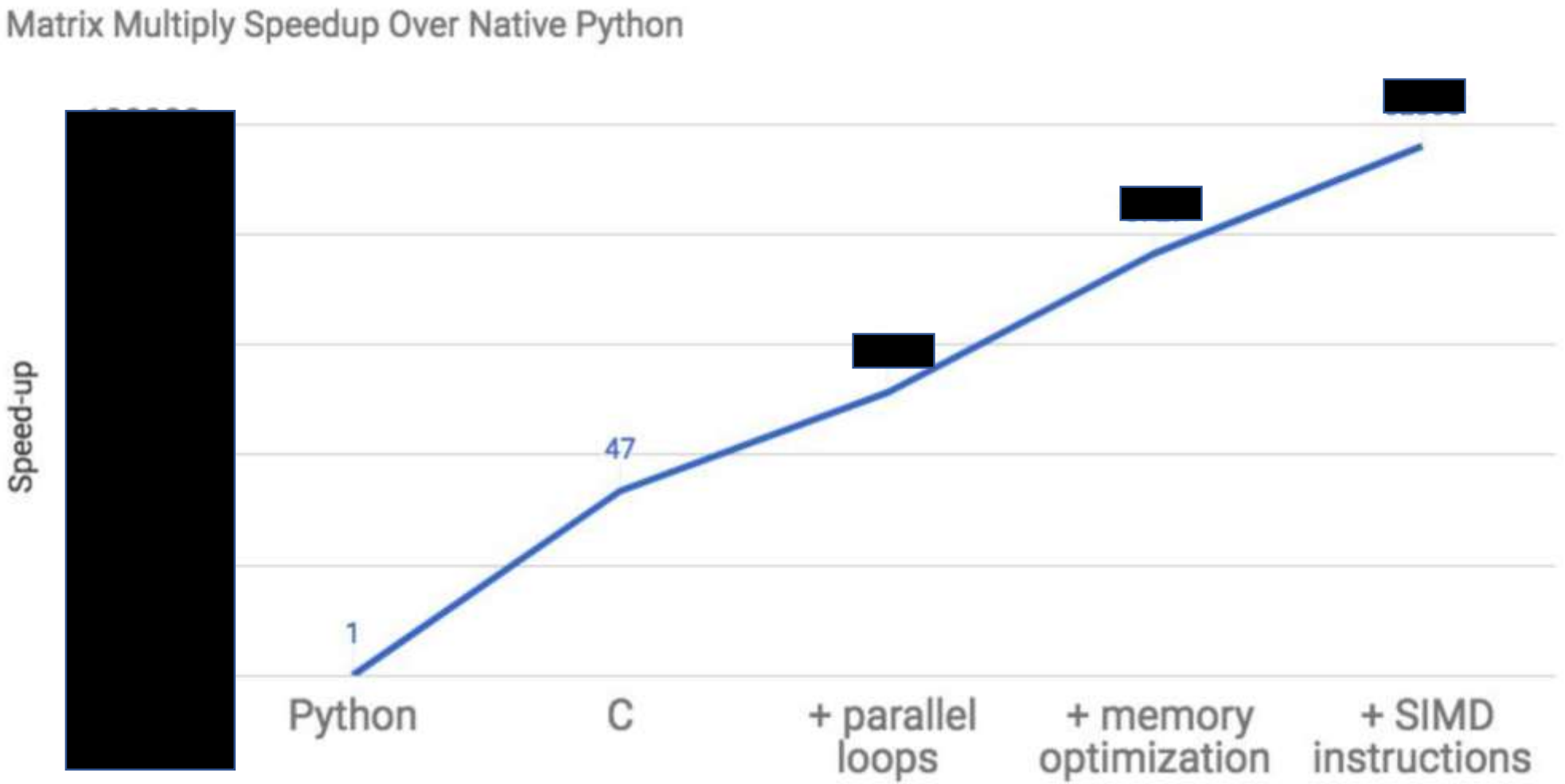


Think about it, before you
write your program

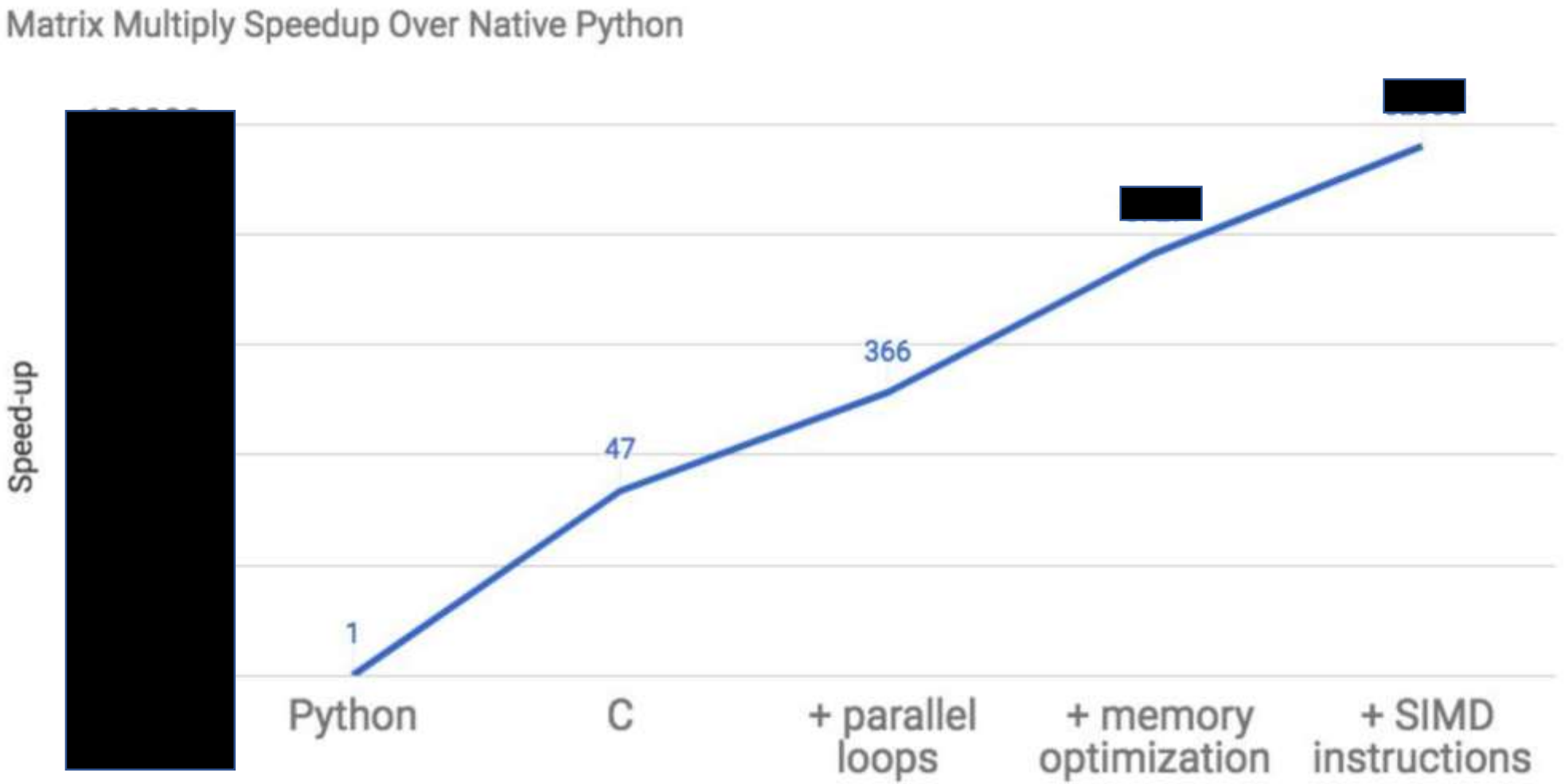
Does programming languages matter?



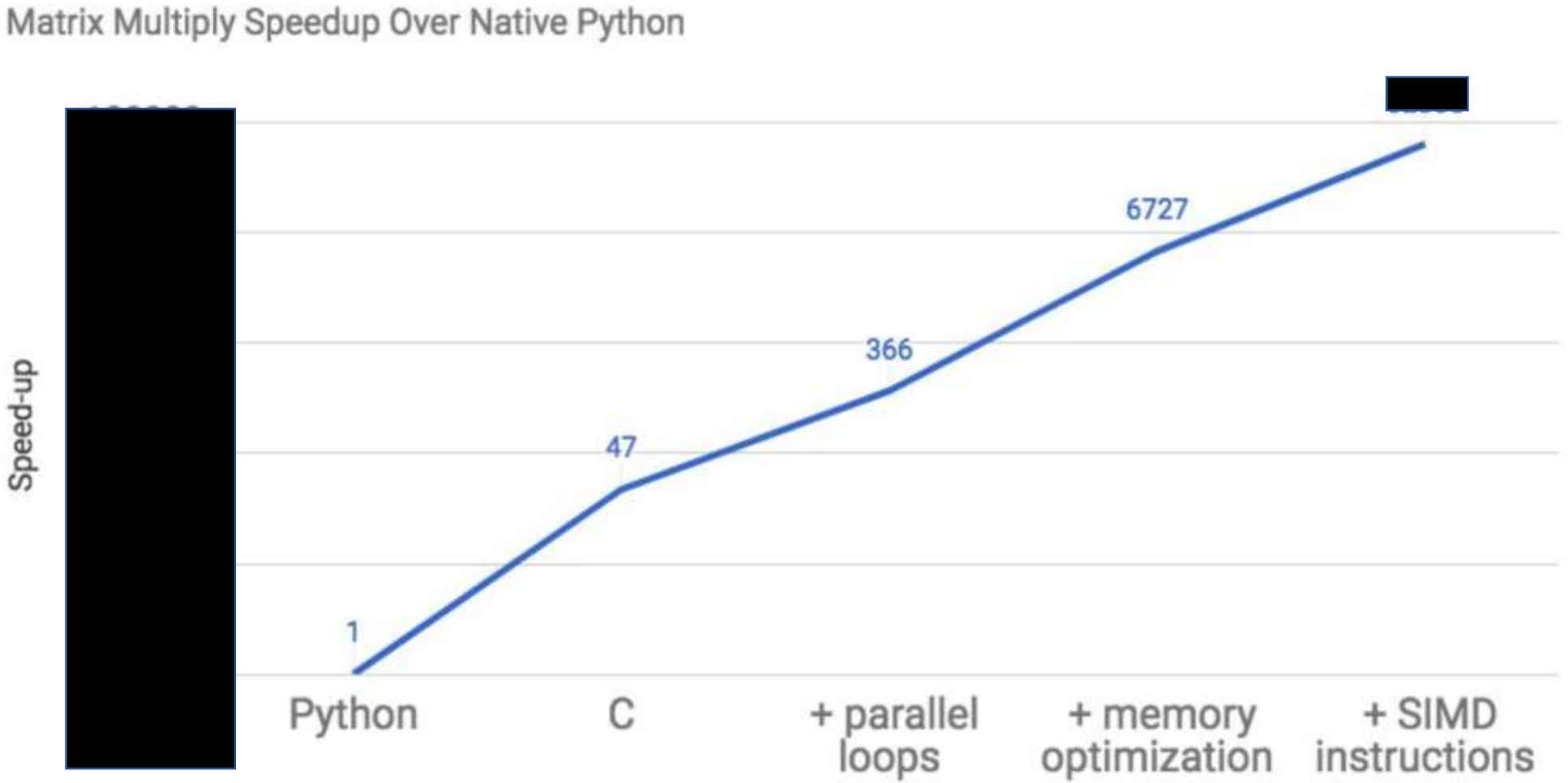
Seriously?


Matrix Multiply Speedup Over Native Python
Speed-up
47
1
Python
C
+ parallel loops
+ memory optimization
+ SIMD instructions

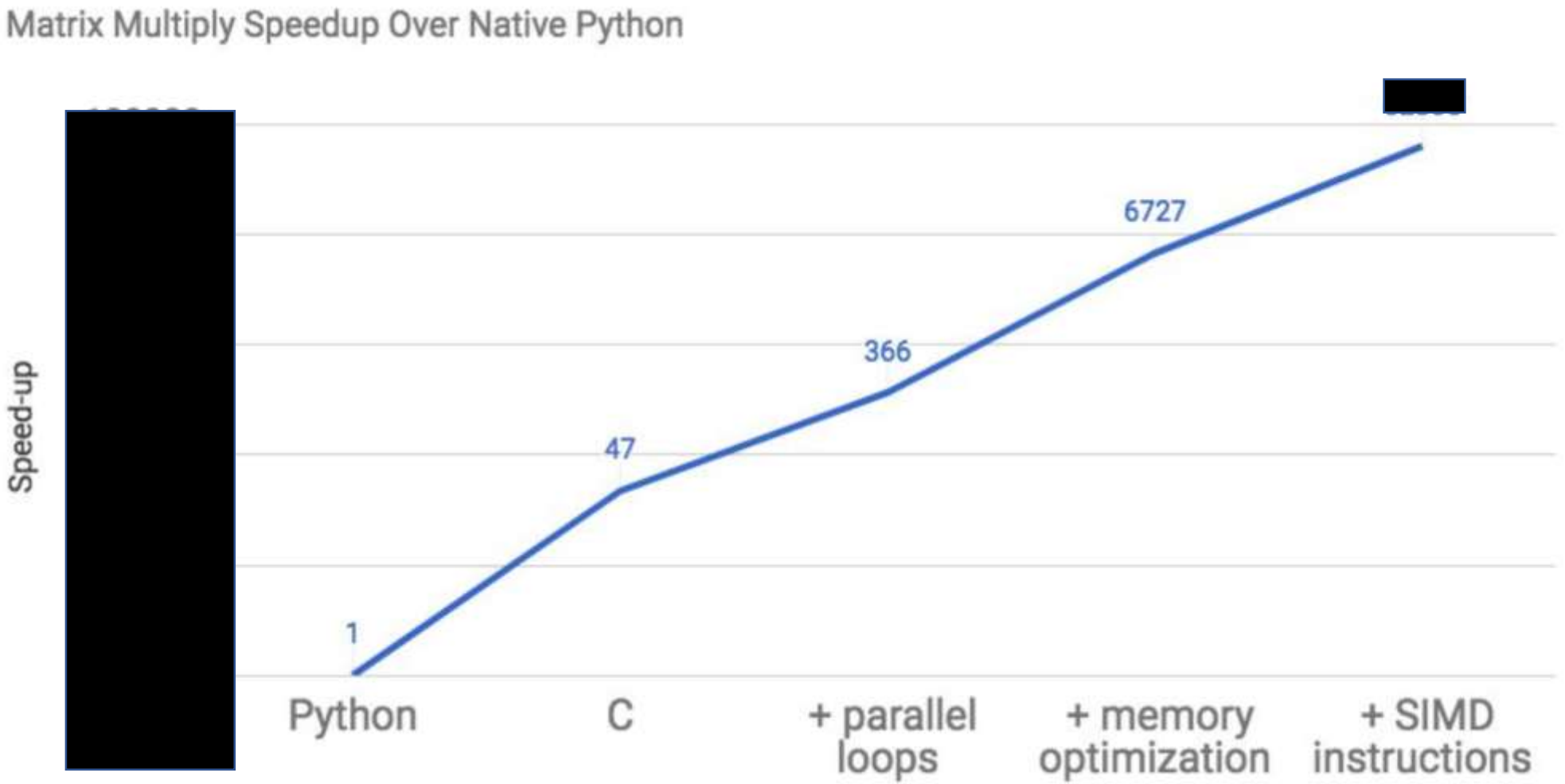
What?


Matrix Multiply Speedup Over Native Python
Speed-up
1
47
366
Python
C
+ parallel loops
+ memory optimization
+ SIMD instructions

Insane

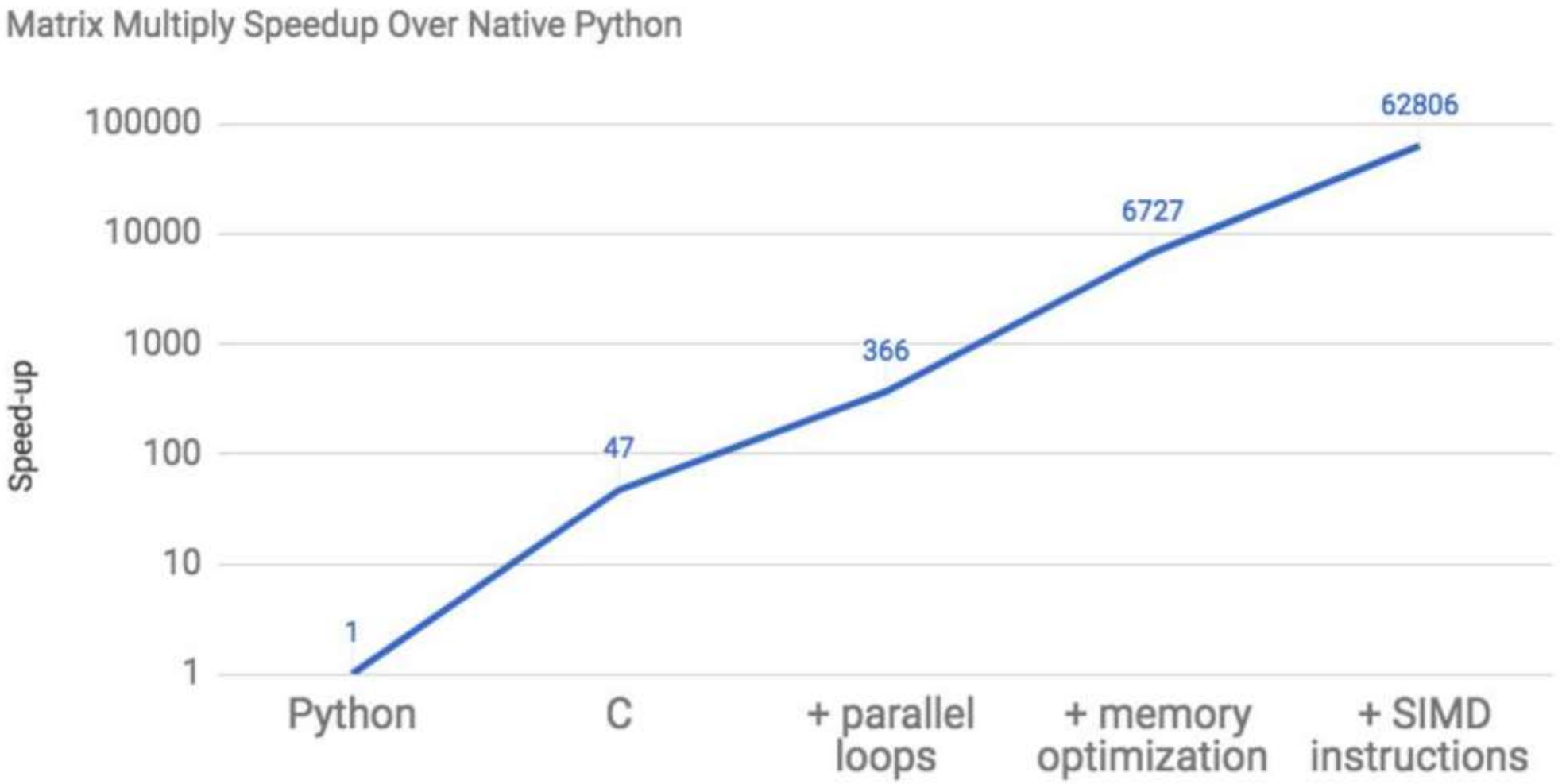

Matrix Multiply Speedup Over Native Python
Speed-up
1
47
366
6727
Python
C
+ parallel loops
+ memory optimization
+ SIMD instructions

Still?


Matrix Multiply Speedup Over Native Python
Speed-up
1
47
366
6727
Python
C
+ parallel loops
+ memory optimization
+ SIMD instructions


62,806X

Ohhhhh!!


Matrix Multiply Speedup Over Native Python
Speed-up
100000
10000
1000
100
10
1
62806
6727
366
47
1
Python
C
+ parallel loops
+ memory optimization
+ SIMD instructions

Can Compilers/programmers exploit locality?

Matrix Multiplication: 101

```
/* ijk */
for (i=0; i<n; i++)  {
  for (j=0; j<n; j++) {
    sum = 0.0;
    for (k=0; k<n; k++)
      sum += a[i][k] * b[k][j];
    c[i][j] = sum;
  }
}
```

$$4\times3 + 2\times2 + 7\times5 = 51$$

4	2	7
1	8	2
6	0	1

$\times$

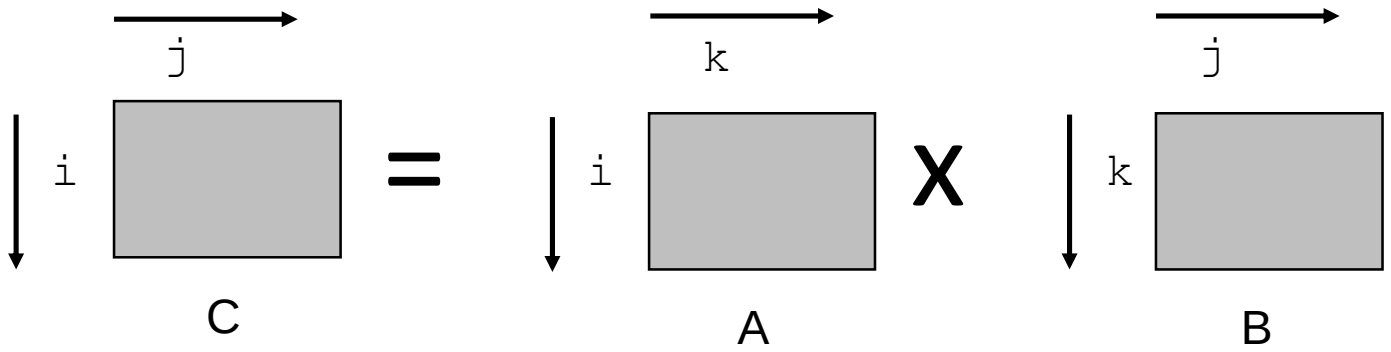
3	0	1
2	4	5
5	9	1

$=$

51		

Miss Rate analysis

- Assume:
  - Block size = 32B (big enough for four doubles)
  - Matrix dimension (N) is very large
    - Approximate 1/N as 0.0
  - Cache is not even big enough to hold multiple rows
- Analysis Method:
  - Look at access pattern of inner loop



Effect of Cache Layout

C arrays allocated in row-major order

- each row in contiguous memory locations

Stepping through columns in one row:

- **for (i = 0; i < N; i++)**
  sum += a[0][i];
- accesses successive elements
- if block size (B) > sizeof(a_{ij}) bytes, exploit spatial locality
  - miss rate = sizeof(a_{ij}) / B

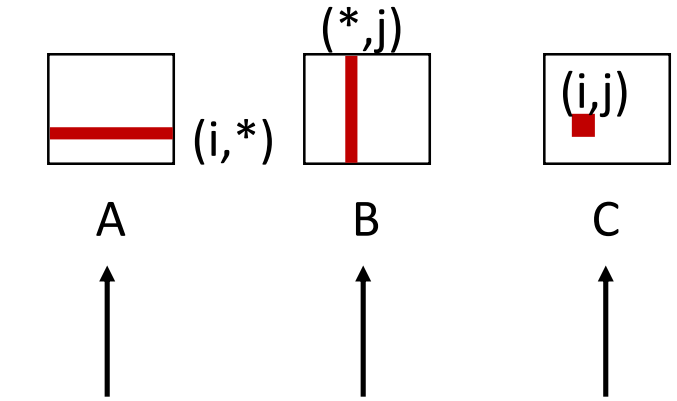
Stepping through rows in one column:

- **for (i = 0; i < N; i++)**
  sum += a[i][0];
- accesses distant elements
- no spatial locality!
  - miss rate = 1 (i.e. 100%)

Effect of loop order (ijk)

```
/* ijk */
for (i=0; i<n; i++)  {
  for (j=0; j<n; j++) {
    sum = 0.0;
    for (k=0; k<n; k++)
      sum += a[i][k] *
b[k][j];
    c[i][j] = sum;
  }
}
```

Inner loop:

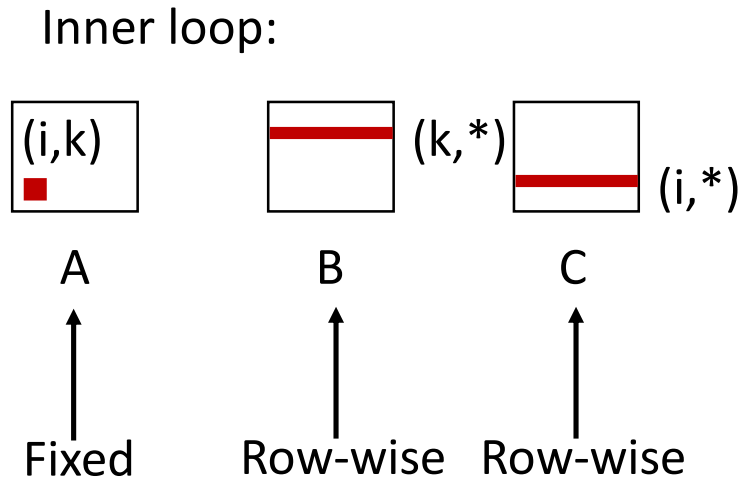


Miss rate for inner loop iterations:

A	B	C
0.25	1.0	0.0

Effect of loops (kij)

```
/* kij */
for (k=0; k<n; k++) {
  for (i=0; i<n; i++) {
    r = a[i][k];
    for (j=0; j<n; j++)
      c[i][j] += r * b[k][j];
  }
}
```

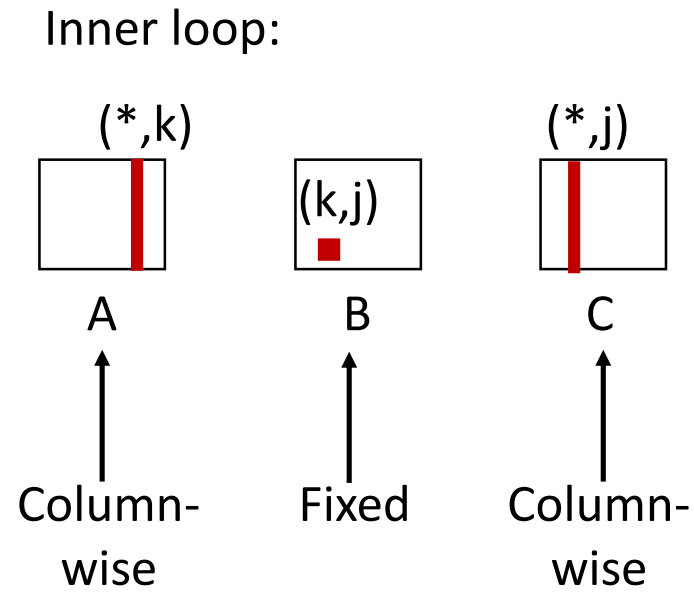


Miss rate for inner loop iterations:

A	B	C
0.0	0.25	0.25

Effect of loops (jki)

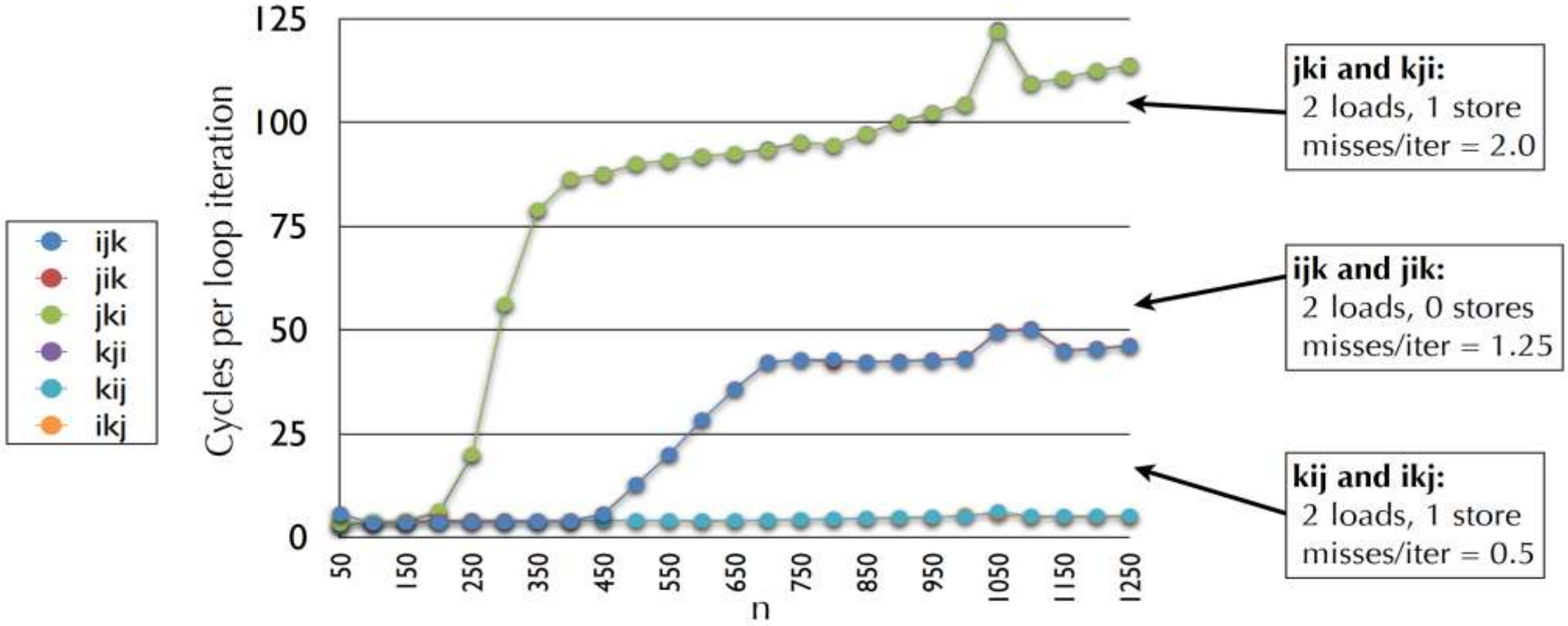
```
/* jki */
for (j=0; j<n; j++) {
  for (k=0; k<n; k++) {
    r = b[k][j];
    for (i=0; i<n; i++)
      c[i][j] += a[i][k] * r;
  }
}
```



Miss rate for inner loop iterations:

A	B	C
1.0	0.0	1.0

Effect of loops



- Miss rate better predictor or performance than number of mem. accesses!
- For large N, kij and ikj performance almost constant.
  Due to **hardware prefetching**, able to recognize stride-1 patterns.

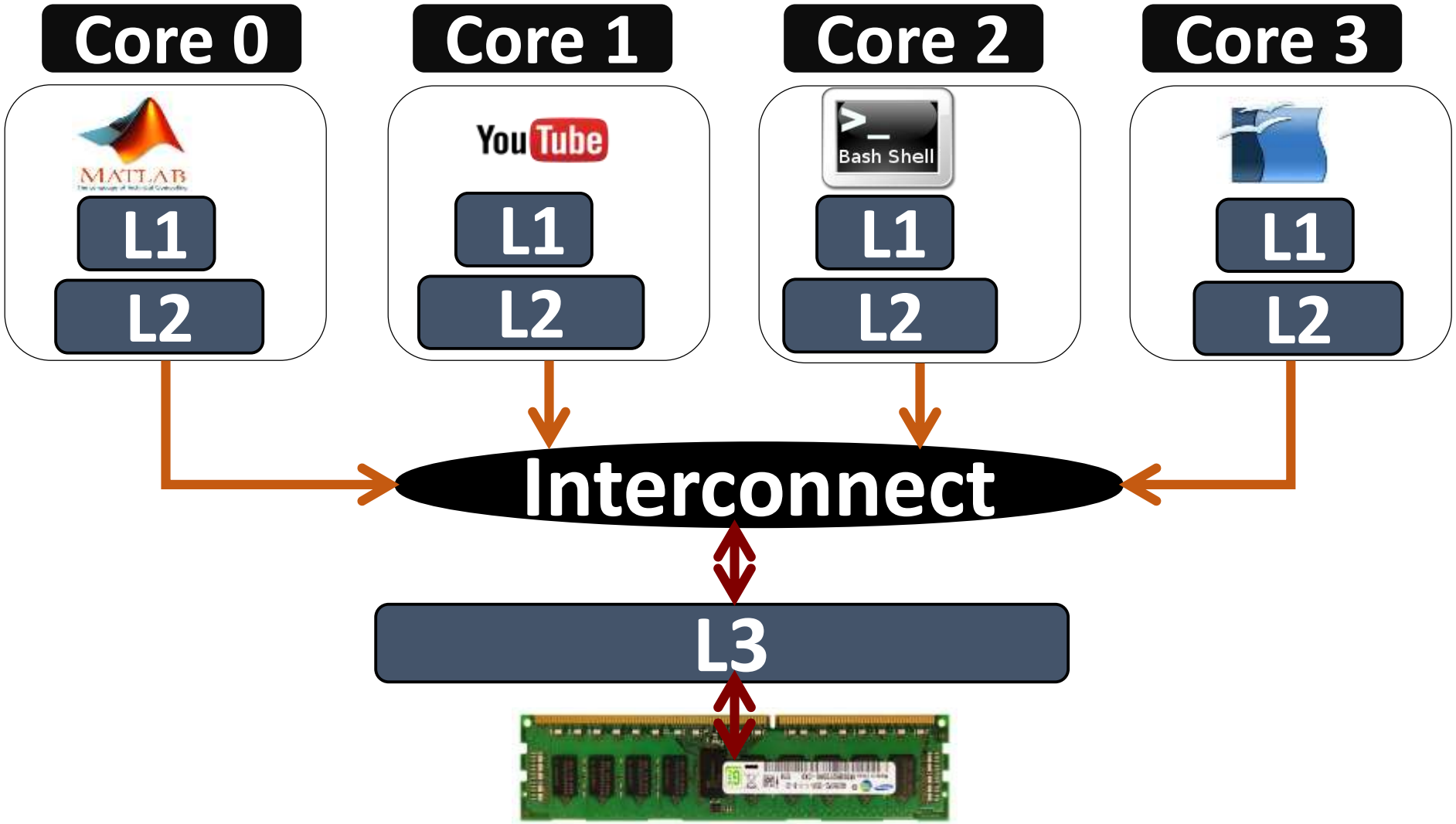
Few Linux commands of interest

perf: https://perf.wiki.kernel.org/index.php/Tutorial#Counting_with_perf_stat

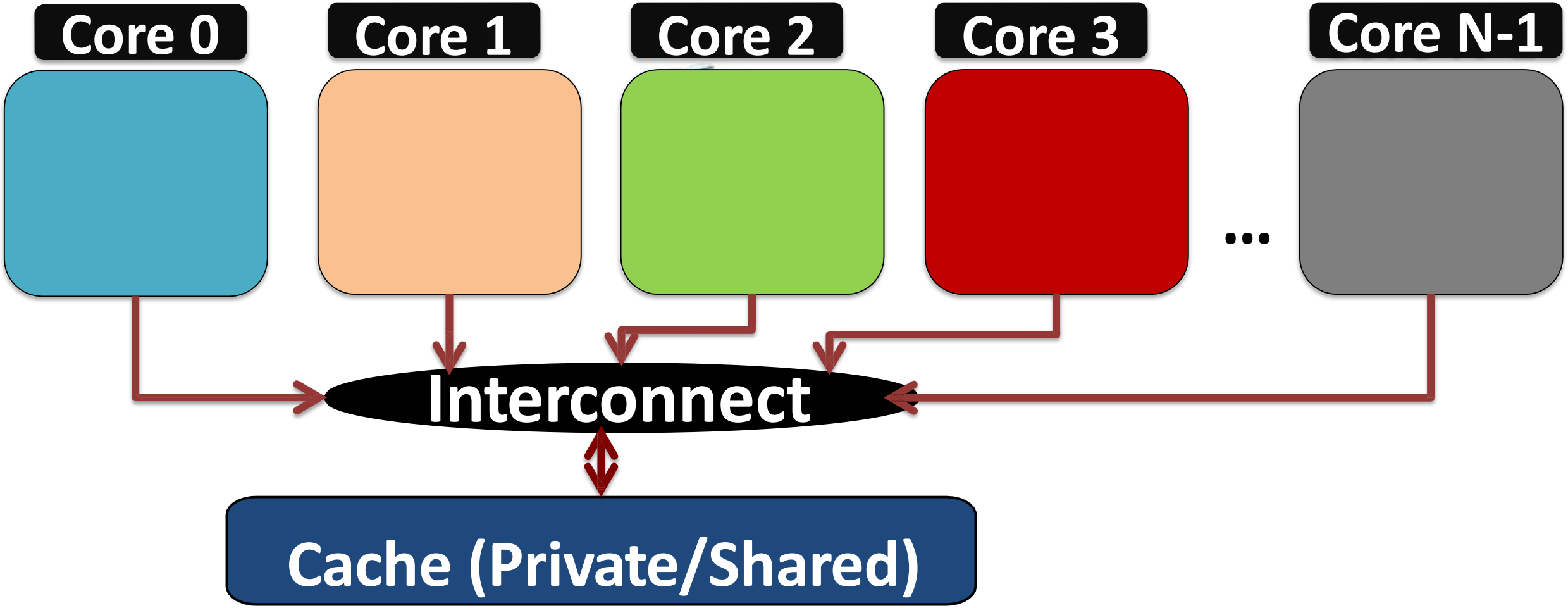
dmidecode

/proc/cpuinfo

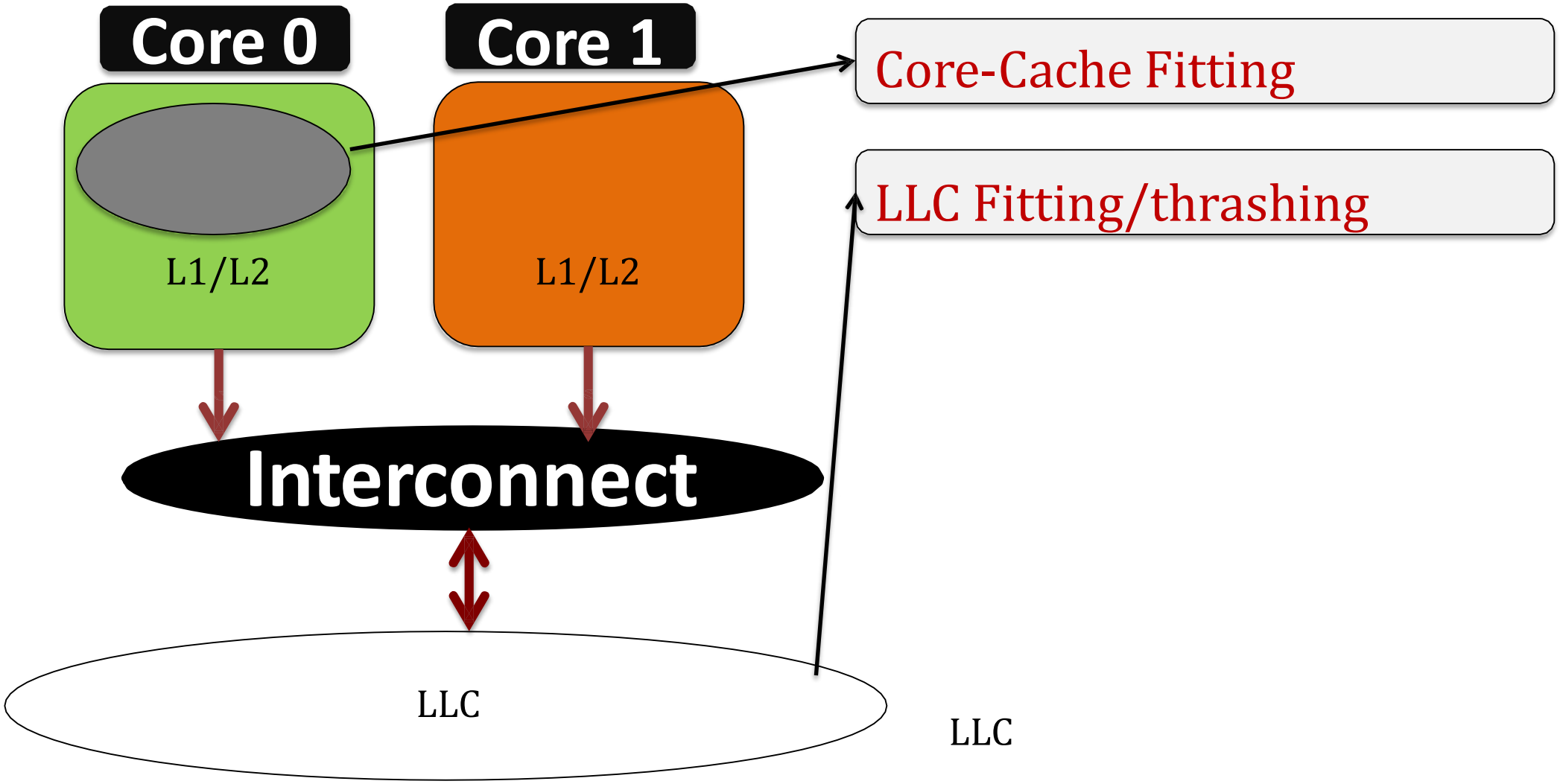
Multicore


Core 0
Core 1
Core 2
Core 3
MATLAB
YouTube
Bash Shell
L1
L2
L1
L2
L1
L2
L1
L2
Interconnect
L3

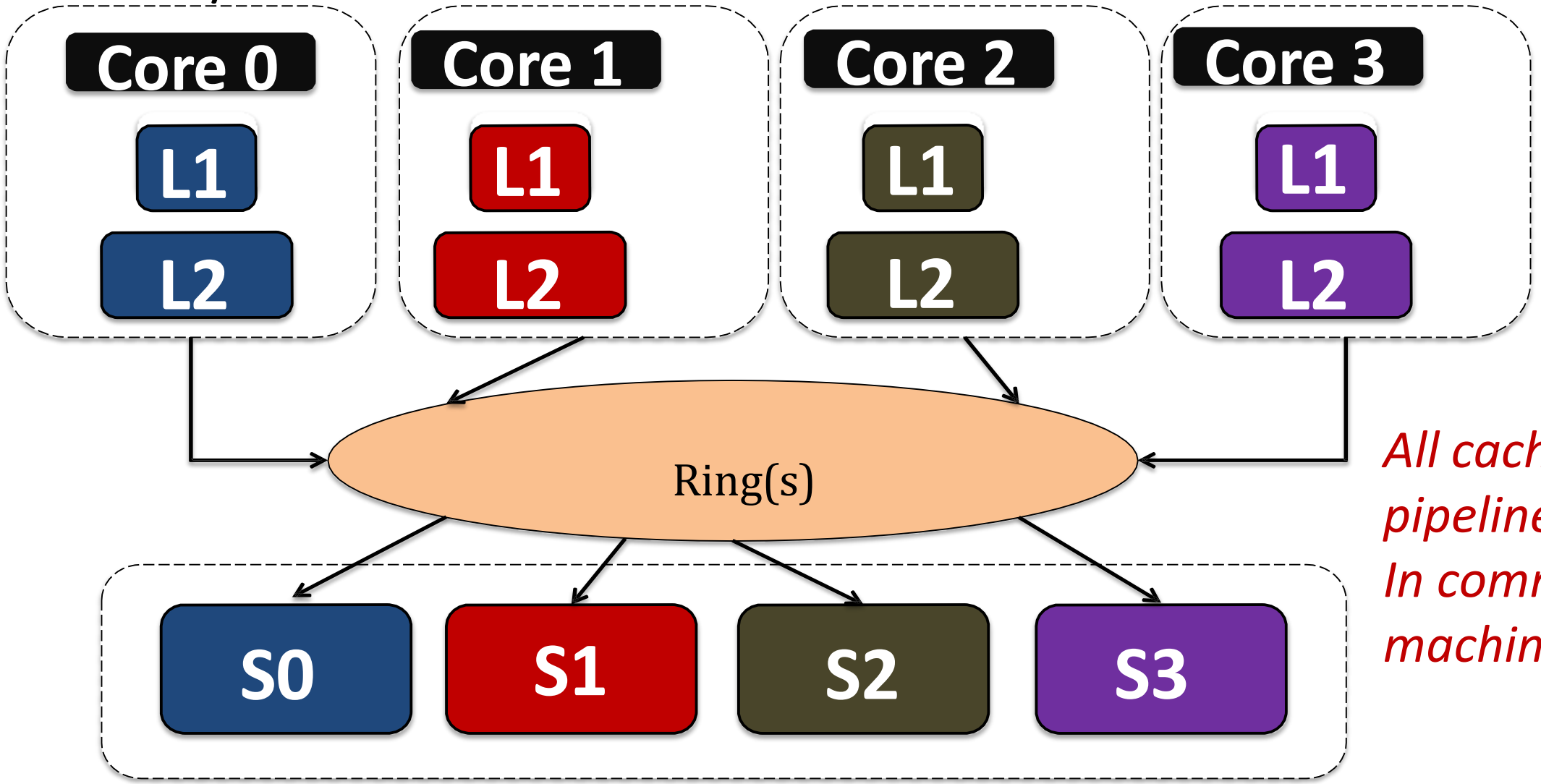
Caches: Private/Shared



Application behavior

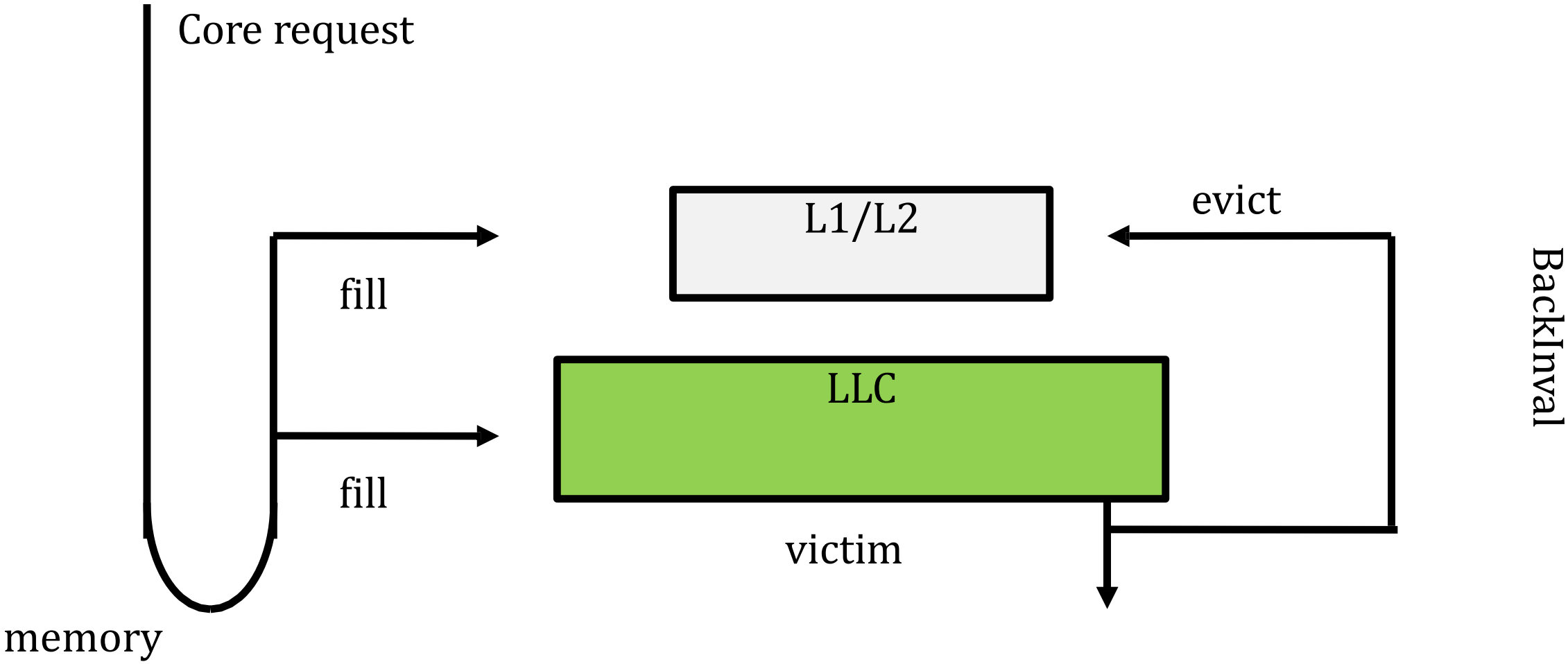


Sliced/Banked LLC

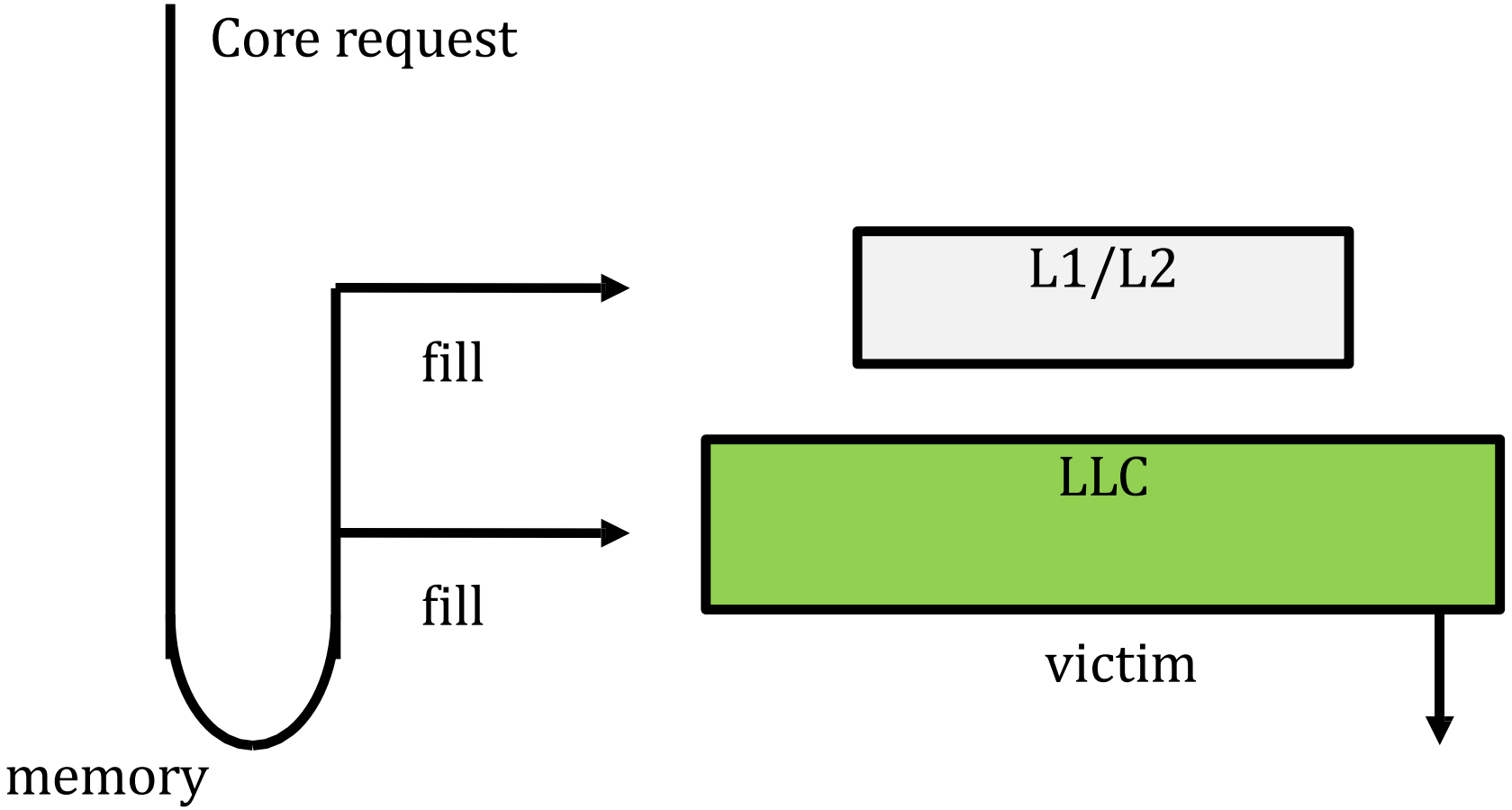


All caches are pipelined
In commercial machines

Inclusive Cache Hierarchy



Non-inclusive (many commercial machines)



Exclusive hierarchy