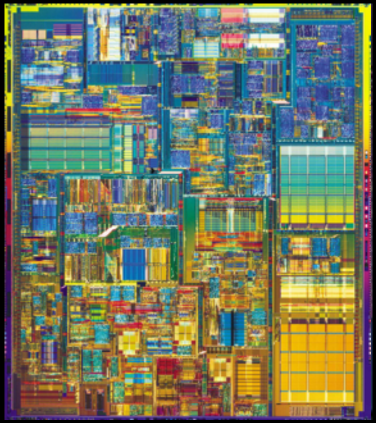
Multithreaded Programming for Mere Mortals

Emery Berger

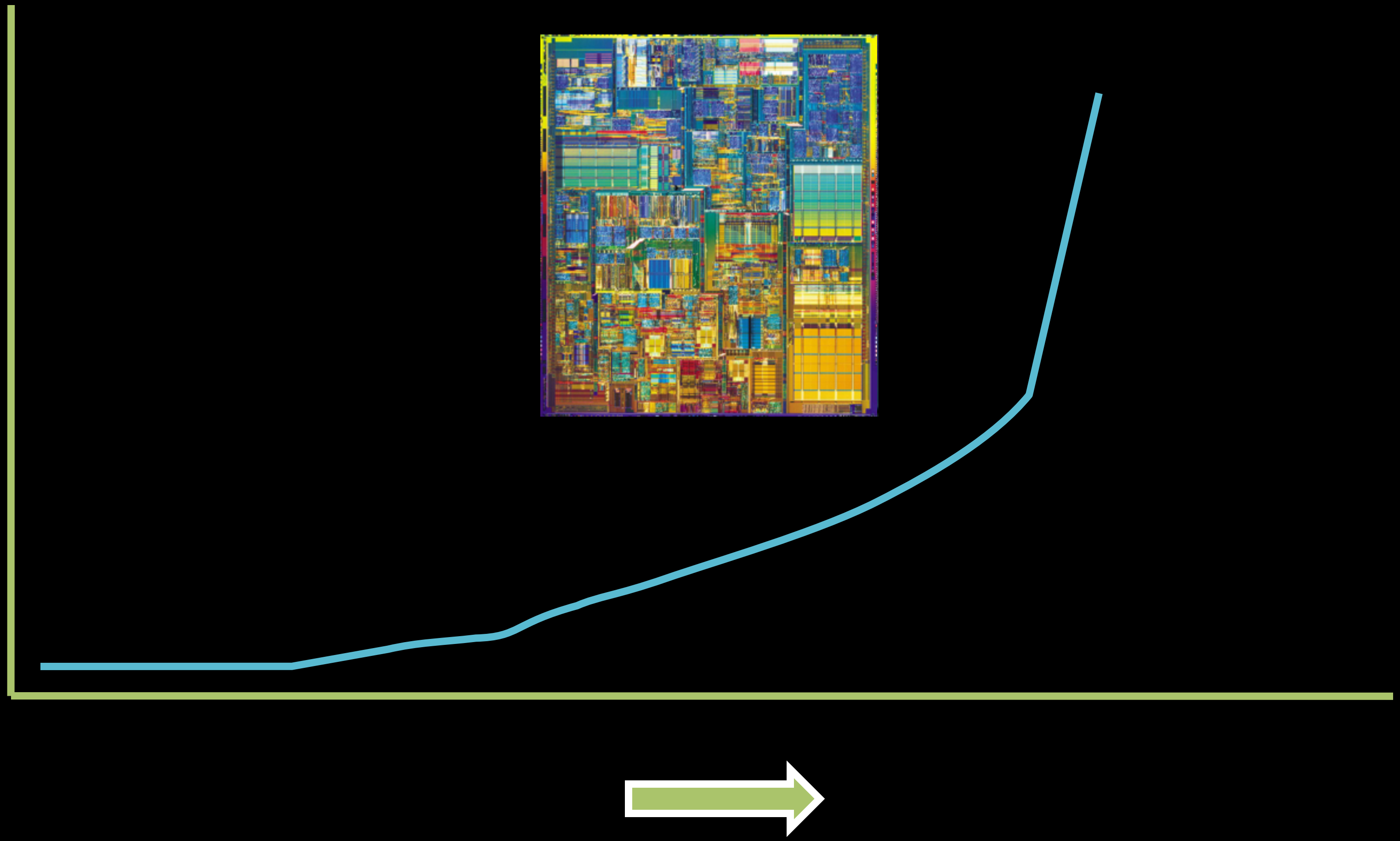
University of Massachusetts, Amherst

In the Beginning…

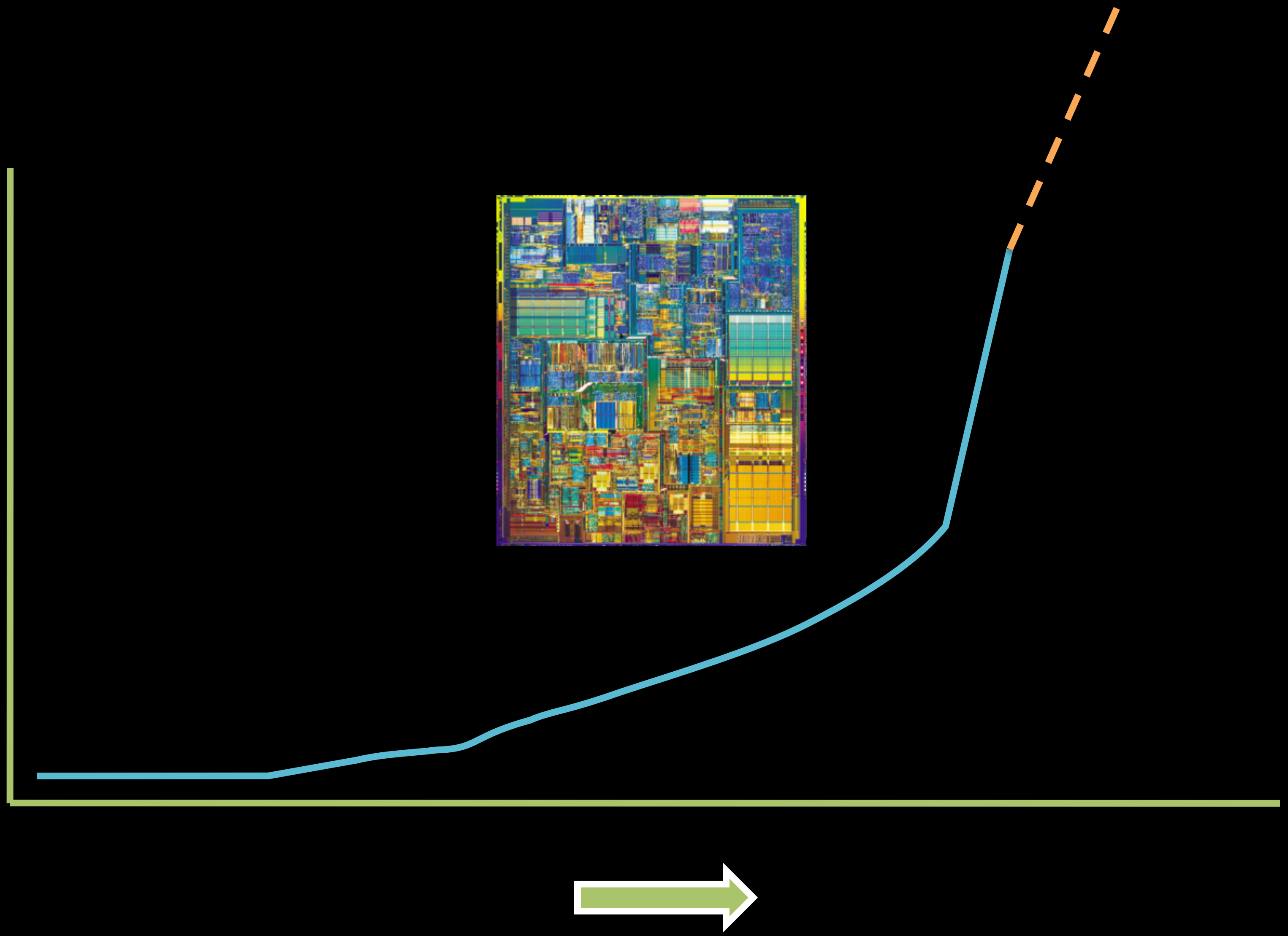
There was the Core.



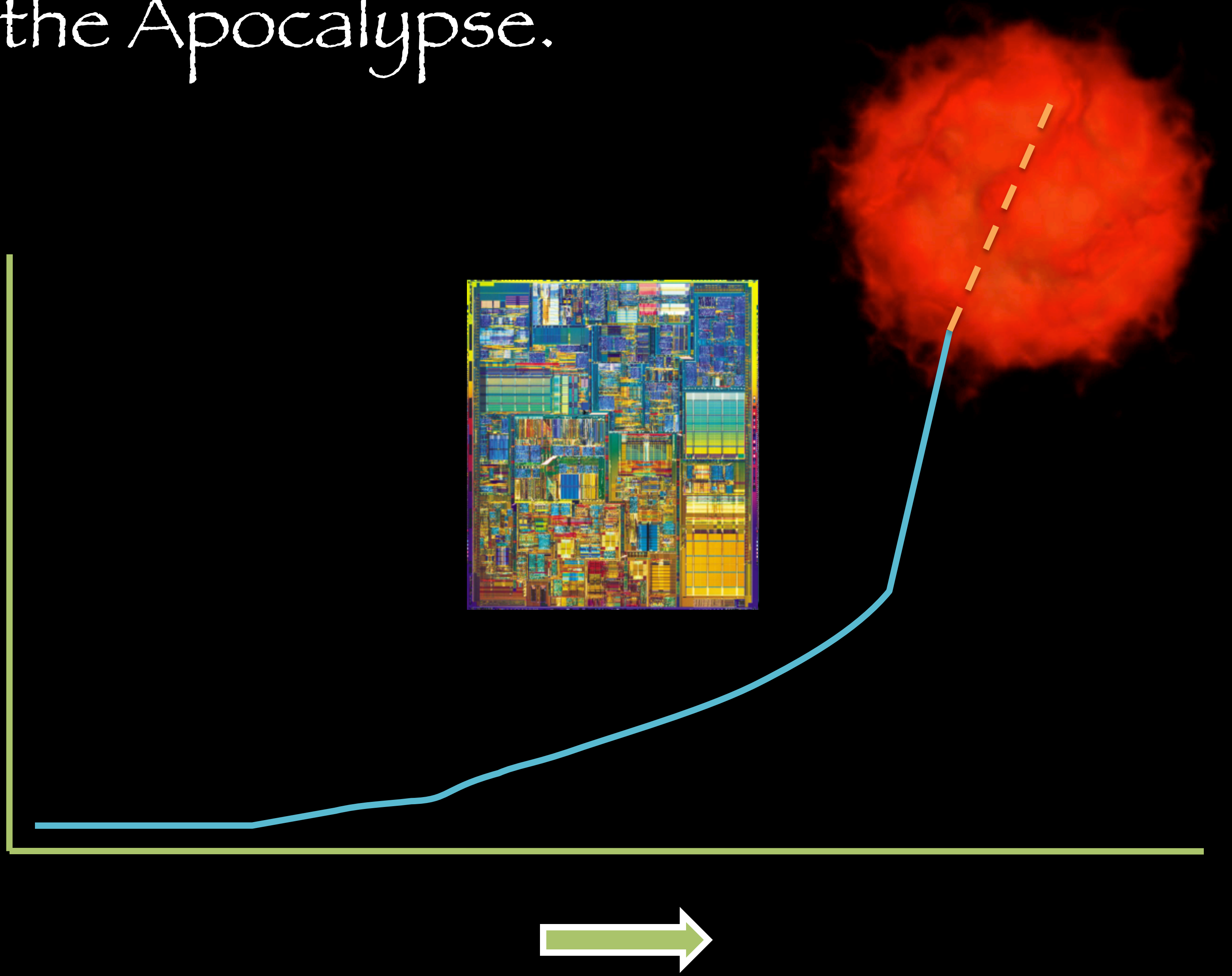
And it was Good.



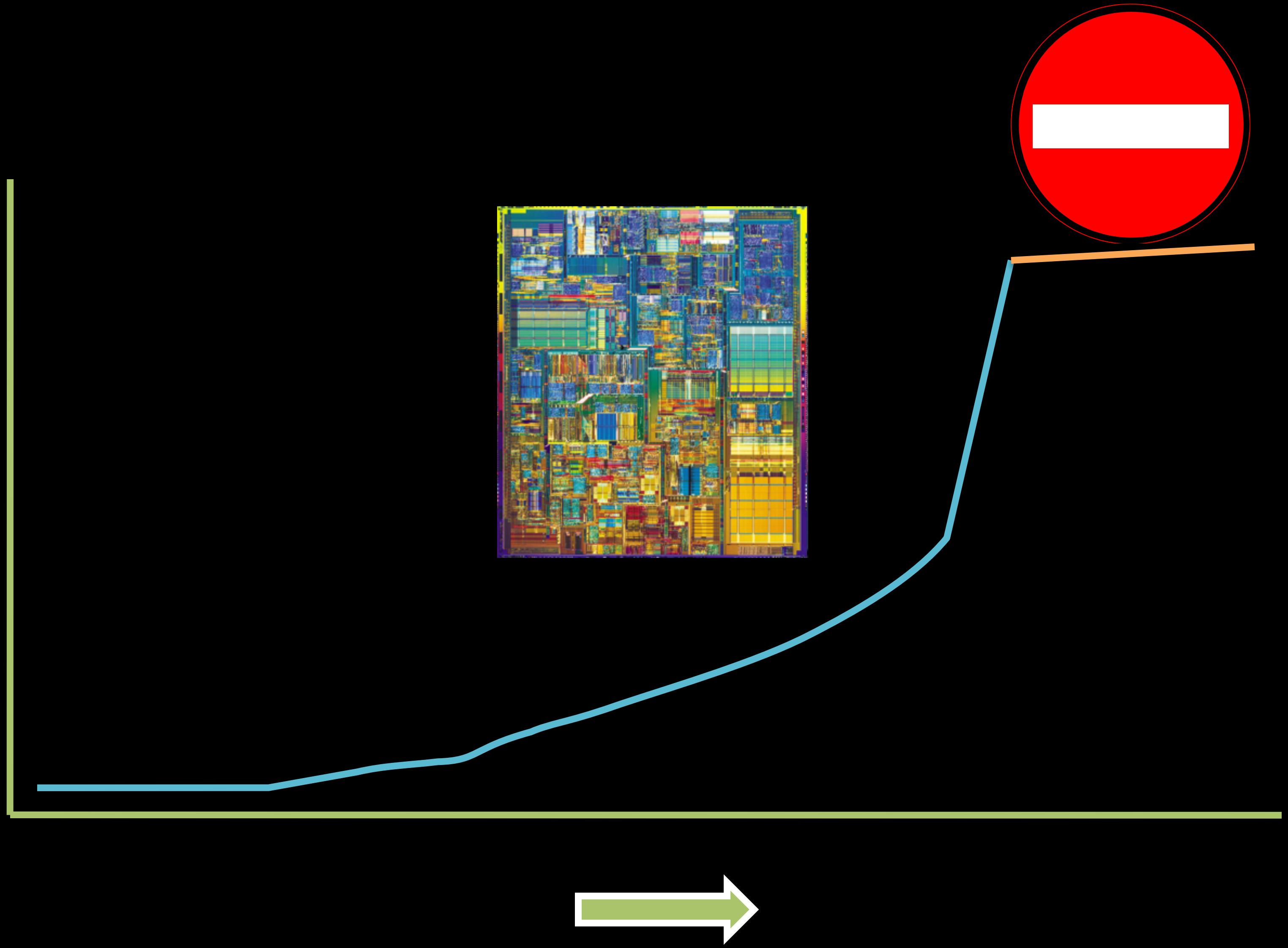
It was Very Good. Until…



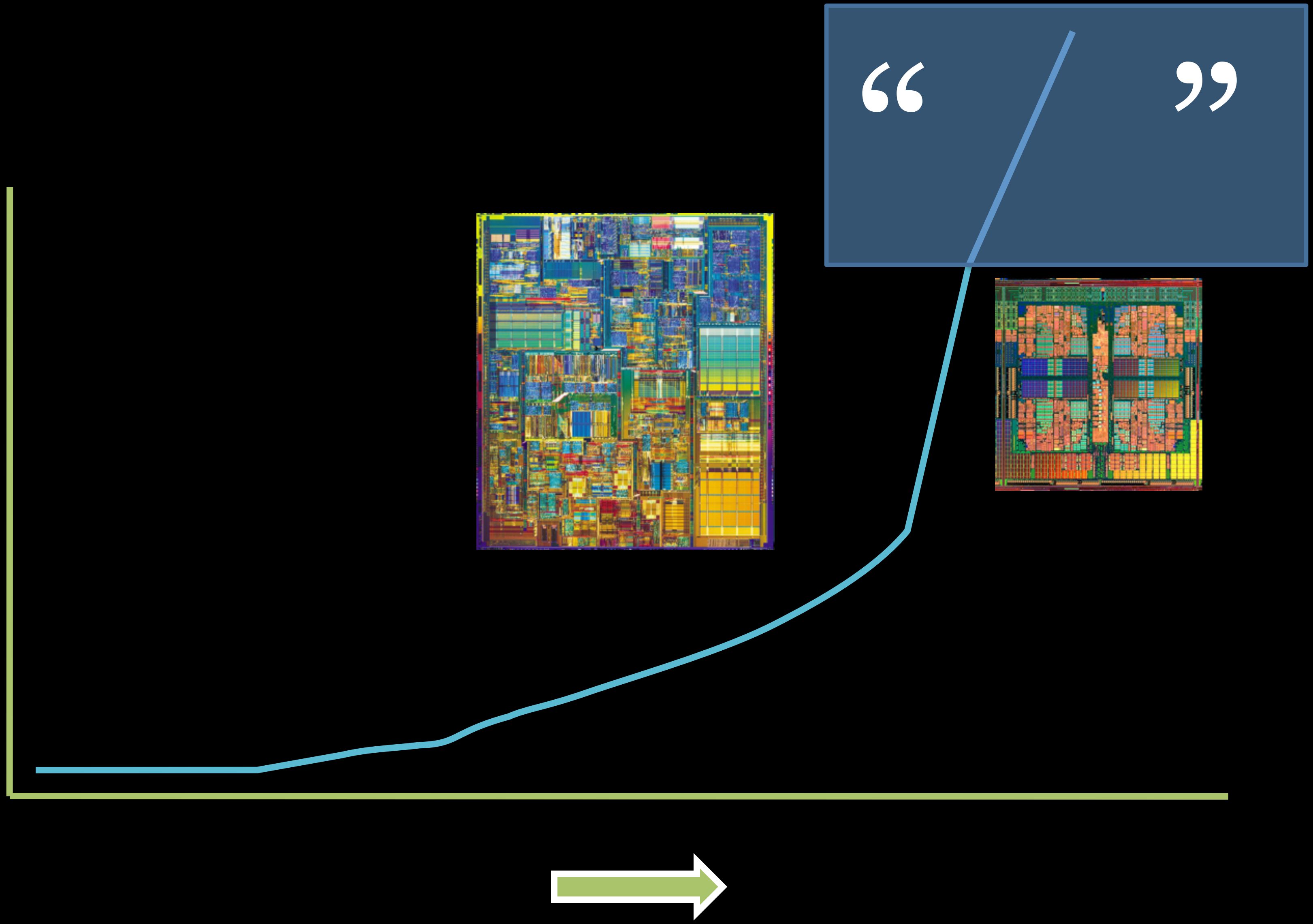
…the Apocalypse.



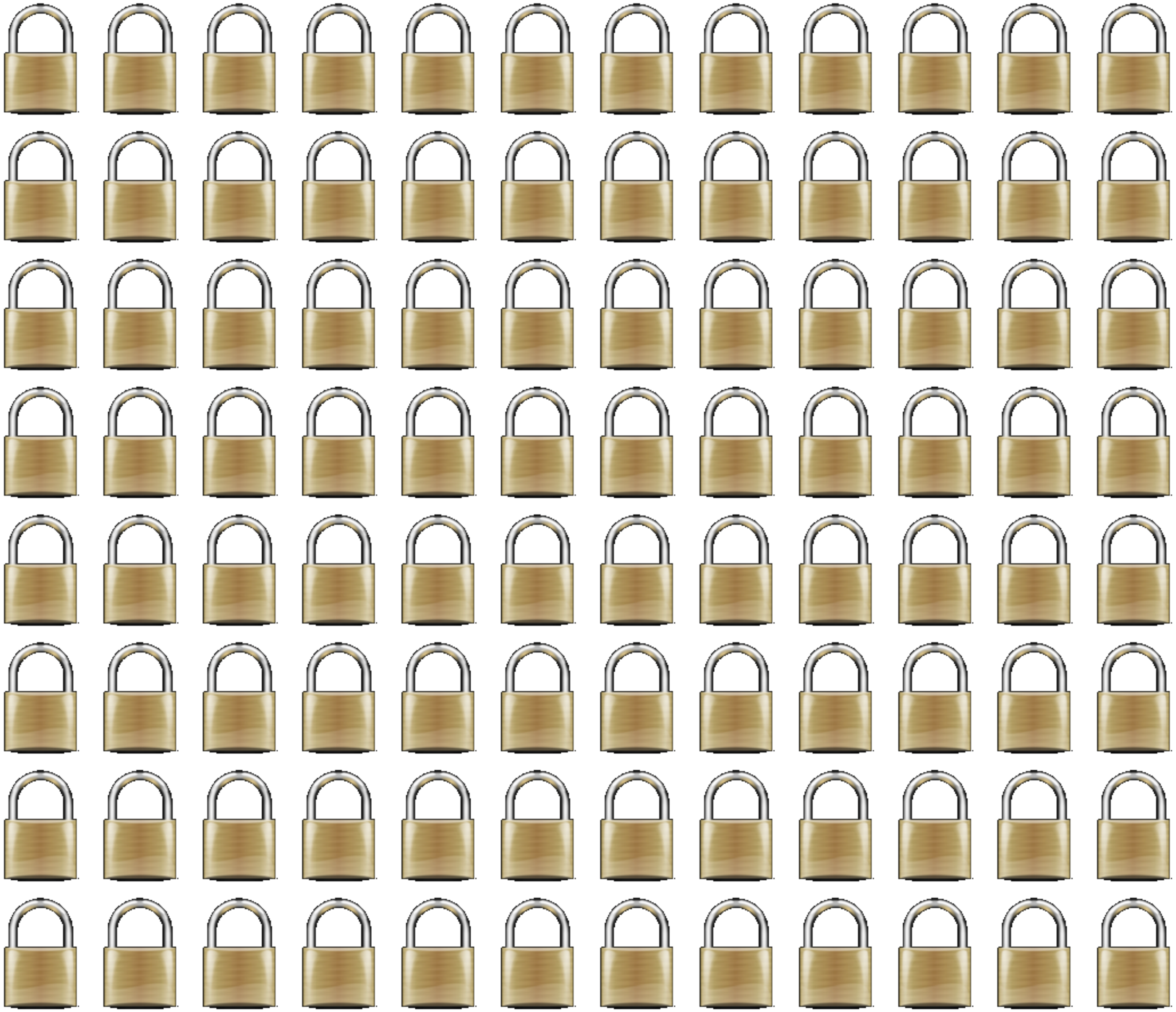
And the Speed was no Moore.



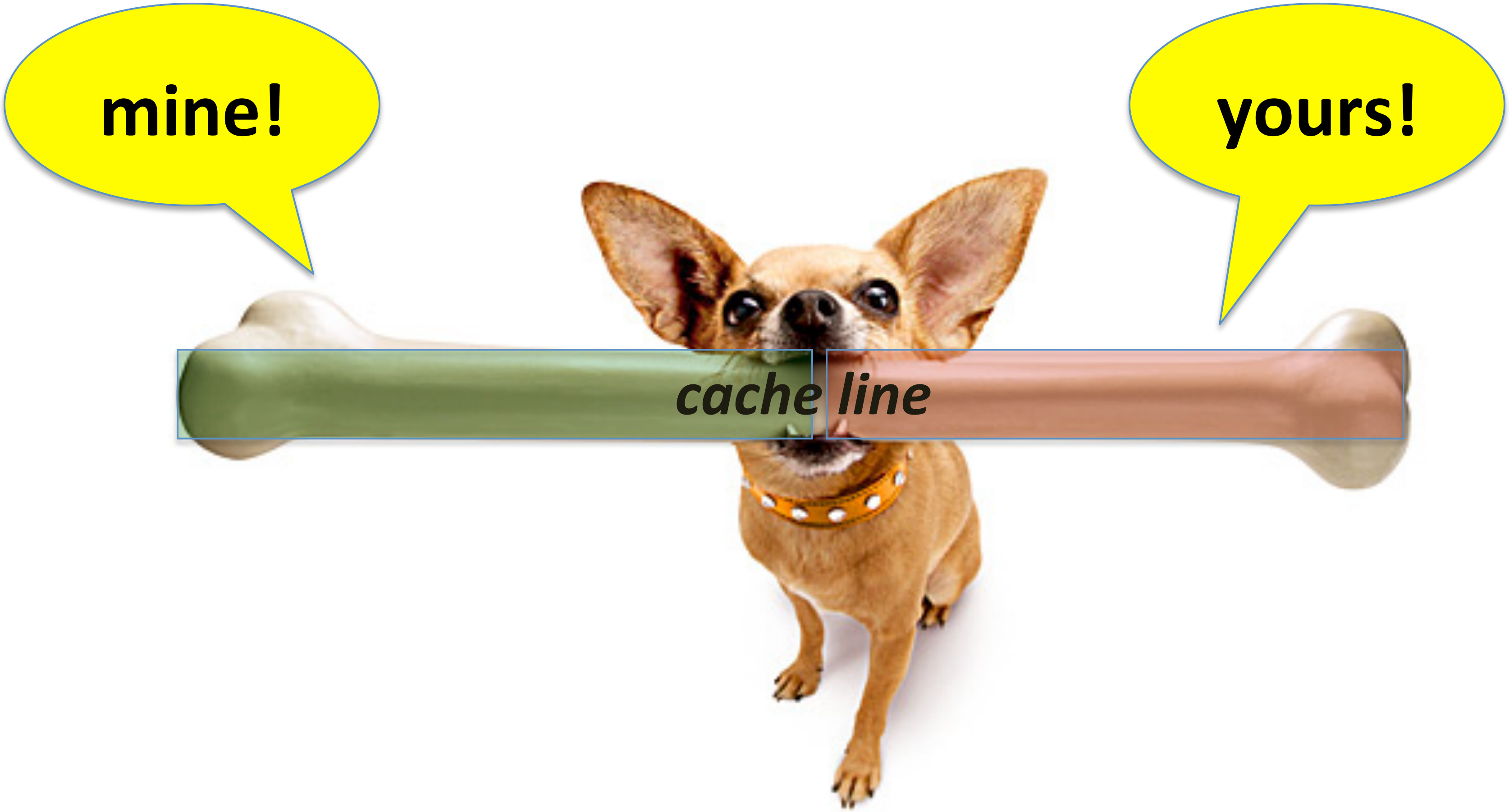
More is Moore (?)...

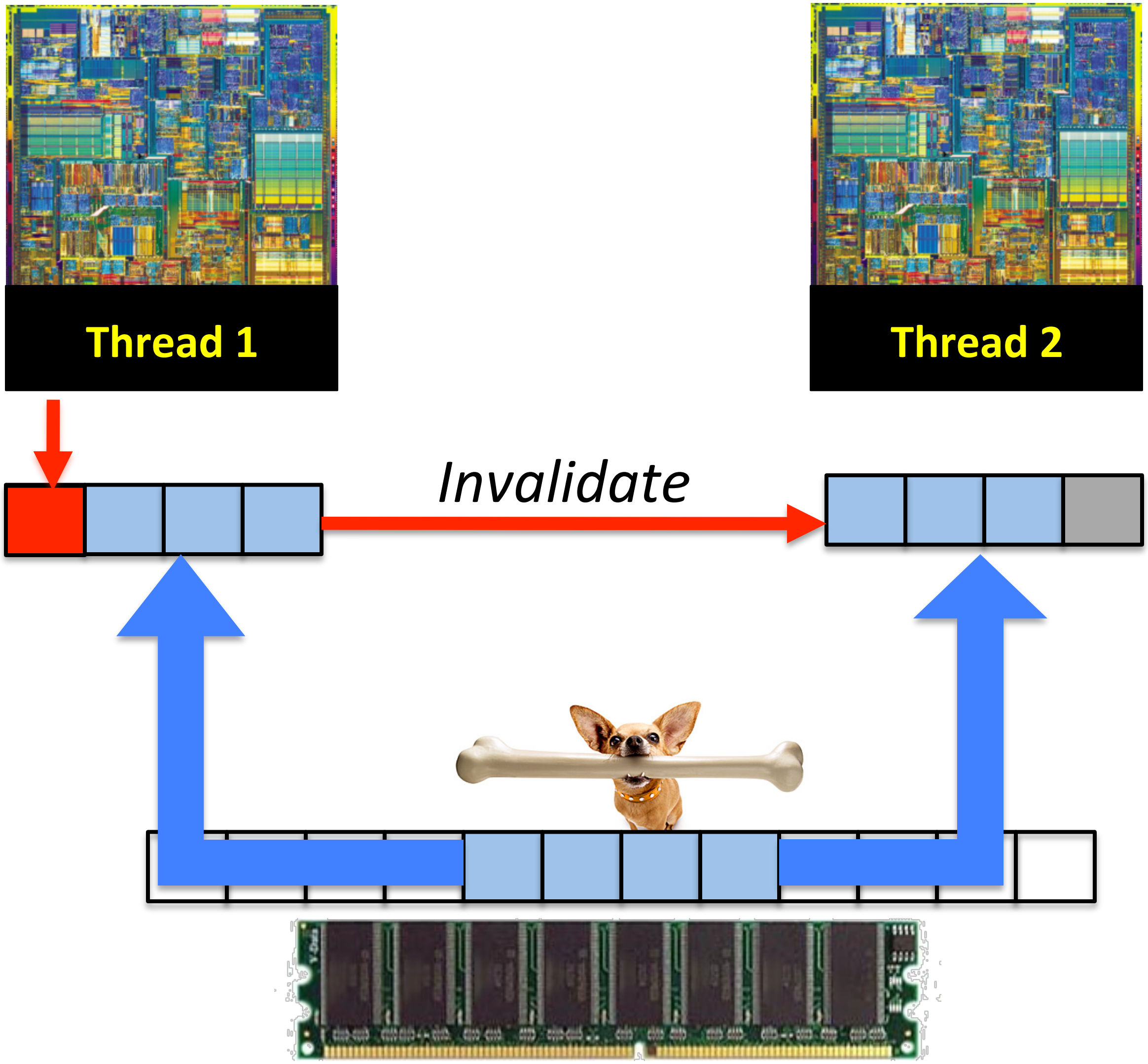


```
color = 🟨; row = 0; // globals
void nextStripe(){
  for (c = 0; c < Width; c++) {
    drawBox (c,row,color);
  }
  color = (color == 🟨)? 🟥 : 🟨;
  row++;
}
for (n = 0; n < 9; n++) {
  spawn nextStripe();
}
sync;
```



FALSE SHARING!

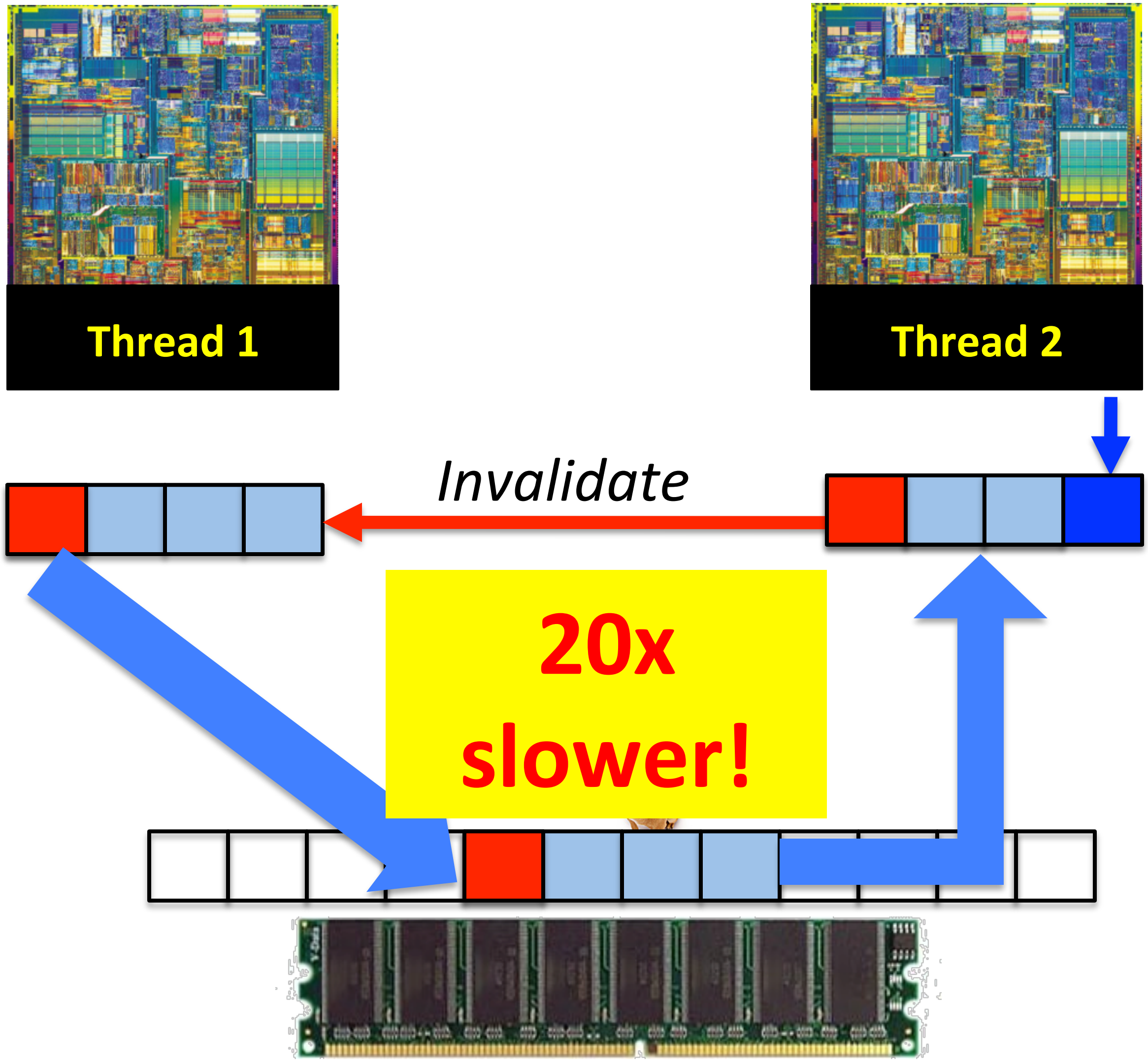

mine!
yours!
cache line



Thread 1

Thread 2

Invalidate



Thread 1

Thread 2

Invalidate

20x slower!

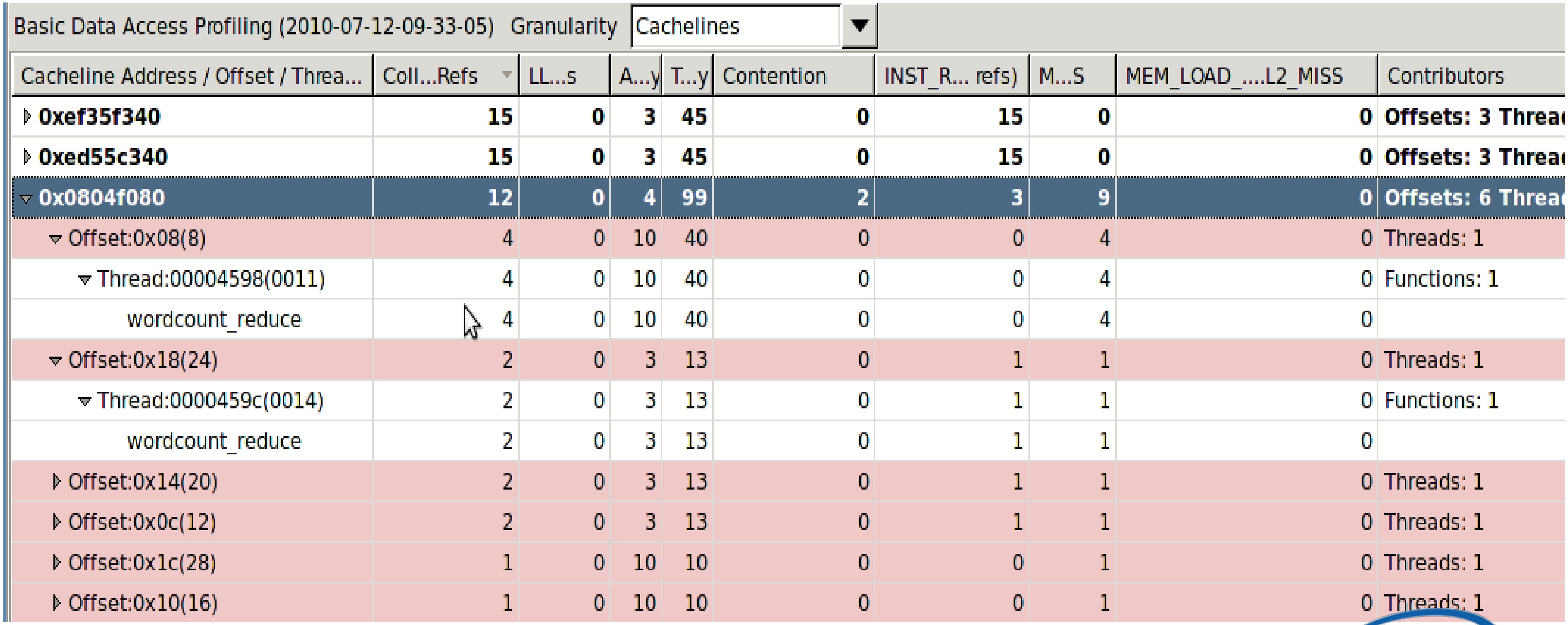
```
me = 1;
you = 1; // globals

me = new Foo;
you = new Bar; // heap

class X {
  int me;
  int you;
}; // fields

arr[me] = 12;
arr[you] = 13; // array indices
```

```
me = 1;
you = 1; // globals

me = new Foo;
you = new Bar; // heap

class X {
  int me;
  int you;
}; // fields

arr[me] = 12;
arr[you] = 13; // array indices
```

```
me = 1;
you = 1; // globals

me = new Foo;
you = new Bar; // heap

class X {
  int me;
  int you;
}; // fields

arr[me] = 12;
arr[you] = 13; // array indices
```

```
me = 1;
you = 1; // globals

me = new Foo;
you = new Bar; // heap

class X {
  int me;
  int you;
}; // fields

arr[me] = 12;
arr[you] = 13; // array indices
```

```
me = 1;
you = 1; // globals

me = new Foo;
you = new Bar; // heap

class X {
  int me;
  int you;
}; // fields

arr[me] = 12;
arr[you] = 13; // array indices
```



Basic Data Access Profiling (2010-07-12-09-33-05) Granularity Cachelines

Cacheline Address / Offset / Threa...	Coll...Refs	LL...s	A...y	T...y	Contention	INST_R... refs)	M...S	MEM_LOAD_....L2_MISS	Contributors
▷ **0xef35f340**	**15**	**0**	**3**	**45**	**0**	**15**	**0**	**0**	**Offsets: 3 Threa**
▷ **0xed55c340**	**15**	**0**	**3**	**45**	**0**	**15**	**0**	**0**	**Offsets: 3 Threa**
▽ **0x0804f080**	**12**	**0**	**4**	**99**	**2**	**3**	**9**	**0**	**Offsets: 6 Threa**
▽ Offset:0x08(8)	4	0	10	40	0	0	4	0	Threads: 1
▽ Thread:00004598(0011)	4	0	10	40	0	0	4	0	Functions: 1
wordcount_reduce	4	0	10	40	0	0	4	0	
▽ Offset:0x18(24)	2	0	3	13	0	1	1	0	Threads: 1
▽ Thread:0000459c(0014)	2	0	3	13	0	1	1	0	Functions: 1
wordcount_reduce	2	0	3	13	0	1	1	0	
▷ Offset:0x14(20)	2	0	3	13	0	1	1	0	Threads: 1
▷ Offset:0x0c(12)	2	0	3	13	0	1	1	0	Threads: 1
▷ Offset:0x1c(28)	1	0	10	10	0	0	1	0	Threads: 1
▷ Offset:0x10(16)	1	0	10	10	0	0	1	0	Threads: 1

+ 850 lines...

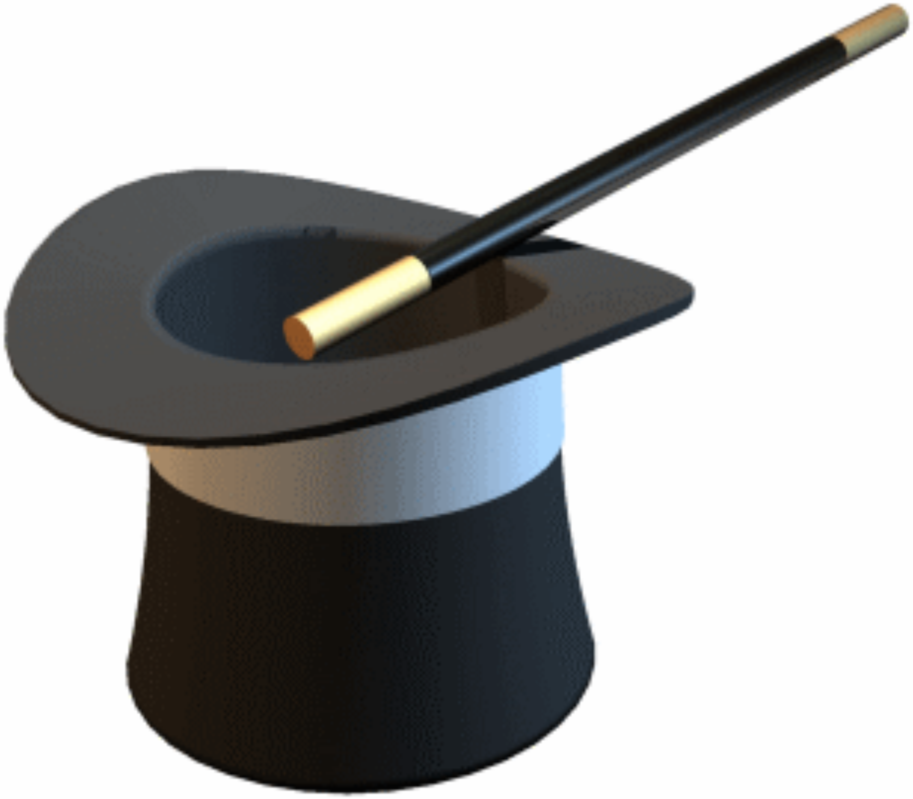
Sheriff-Detect

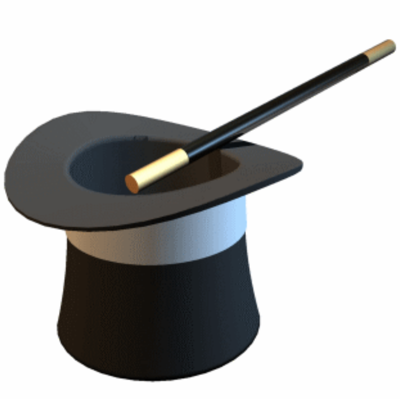
("share-if")

No false positives

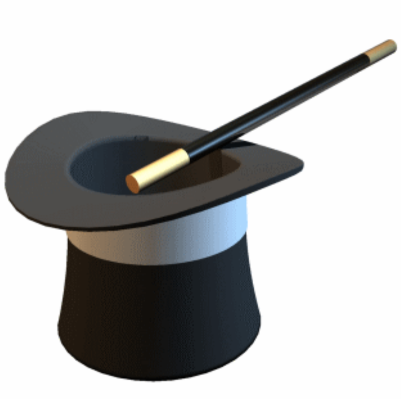
```
Object has 13767 interleaving writes
(starts at 0xd5c8e160, length 32).
Allocation call stack:
   0: wordcount.c: 136
   1: wordcount.c: 444
```

Precise object identification





```
t1 = spawn f(x);
t2 = spawn g(y);
sync;
```



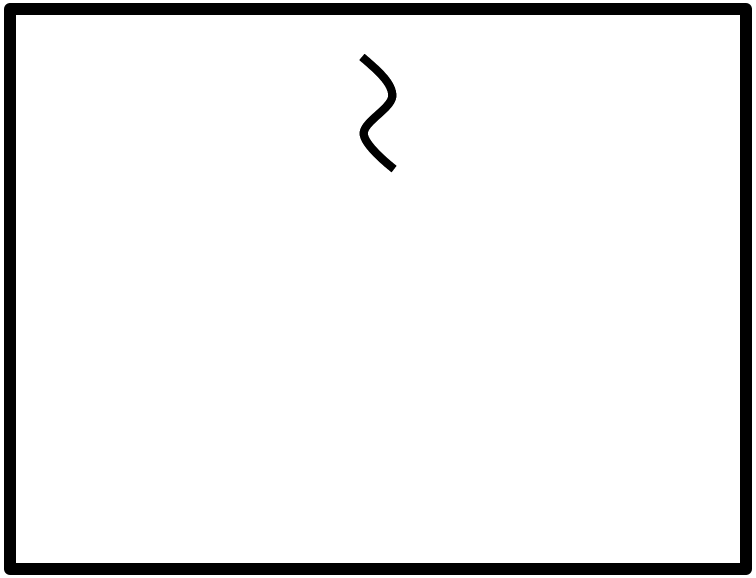
```
t1 = spawn f(x);
t2 = spawn g(y);
sync;
```

```
if (!fork())
  f(x);
if (!fork())
  g(y);
```

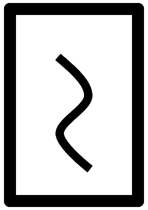
GRACE [OOPSLA 2009]

Isolation

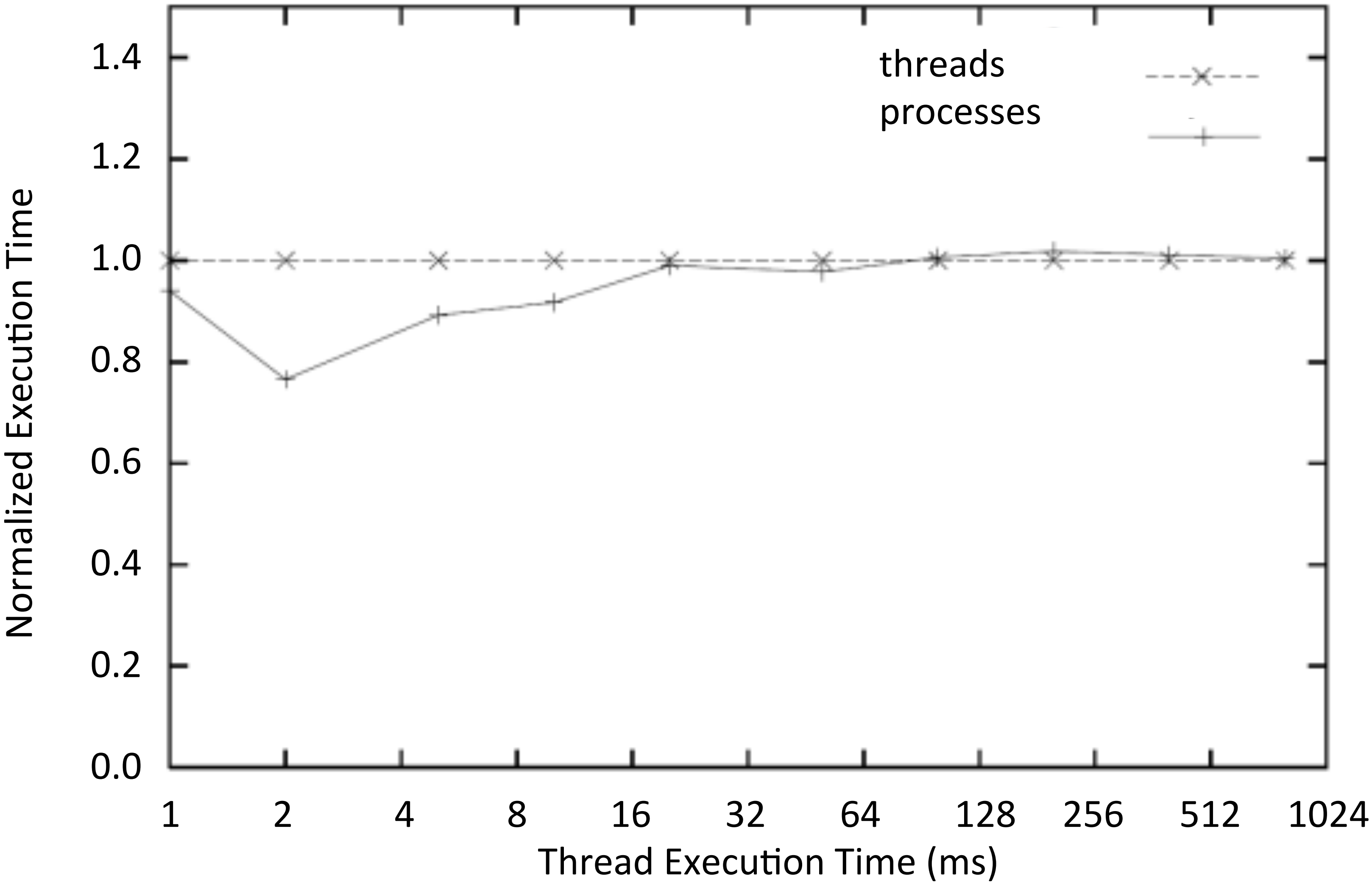
shared address space



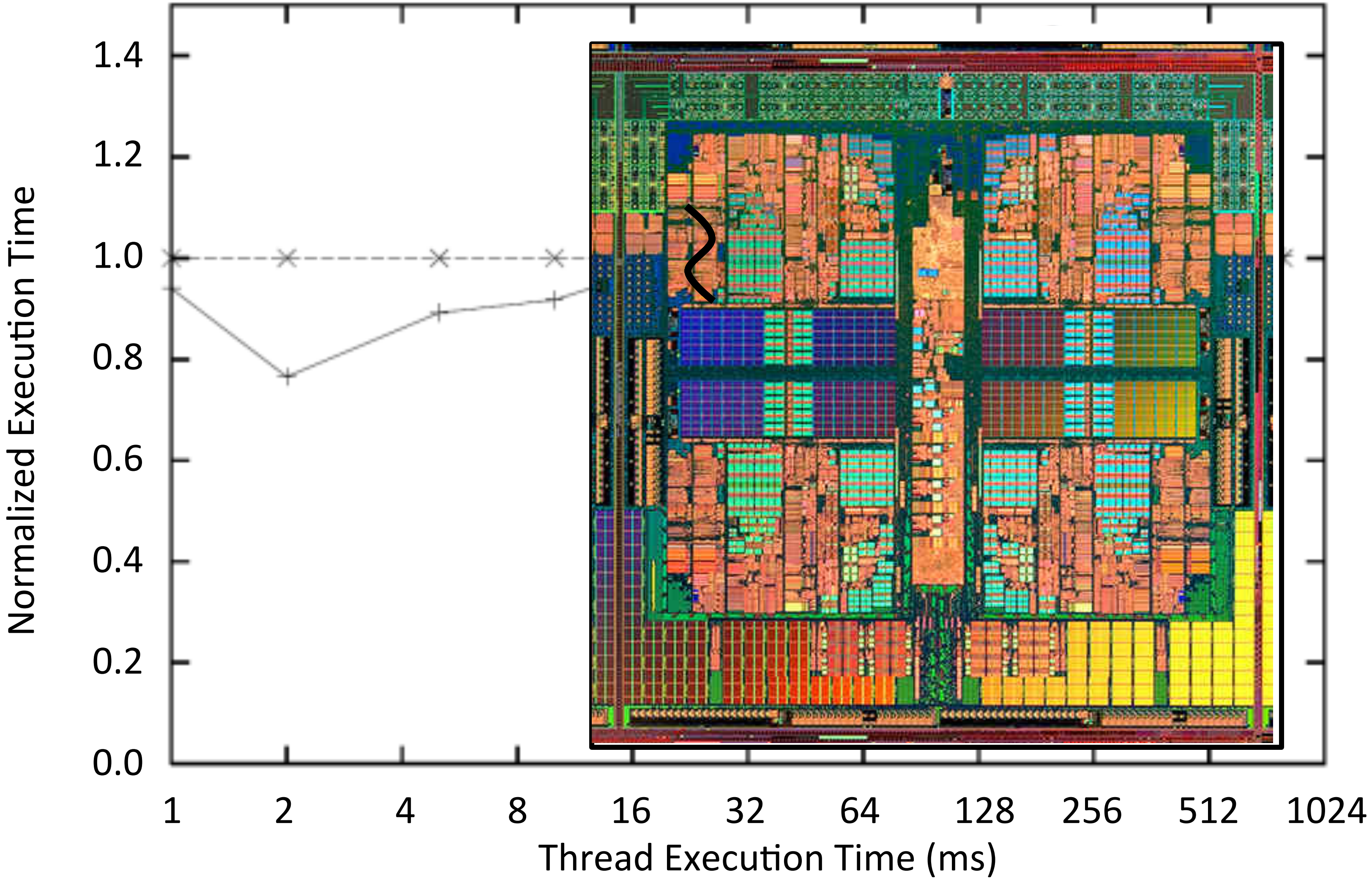
disjoint address spaces



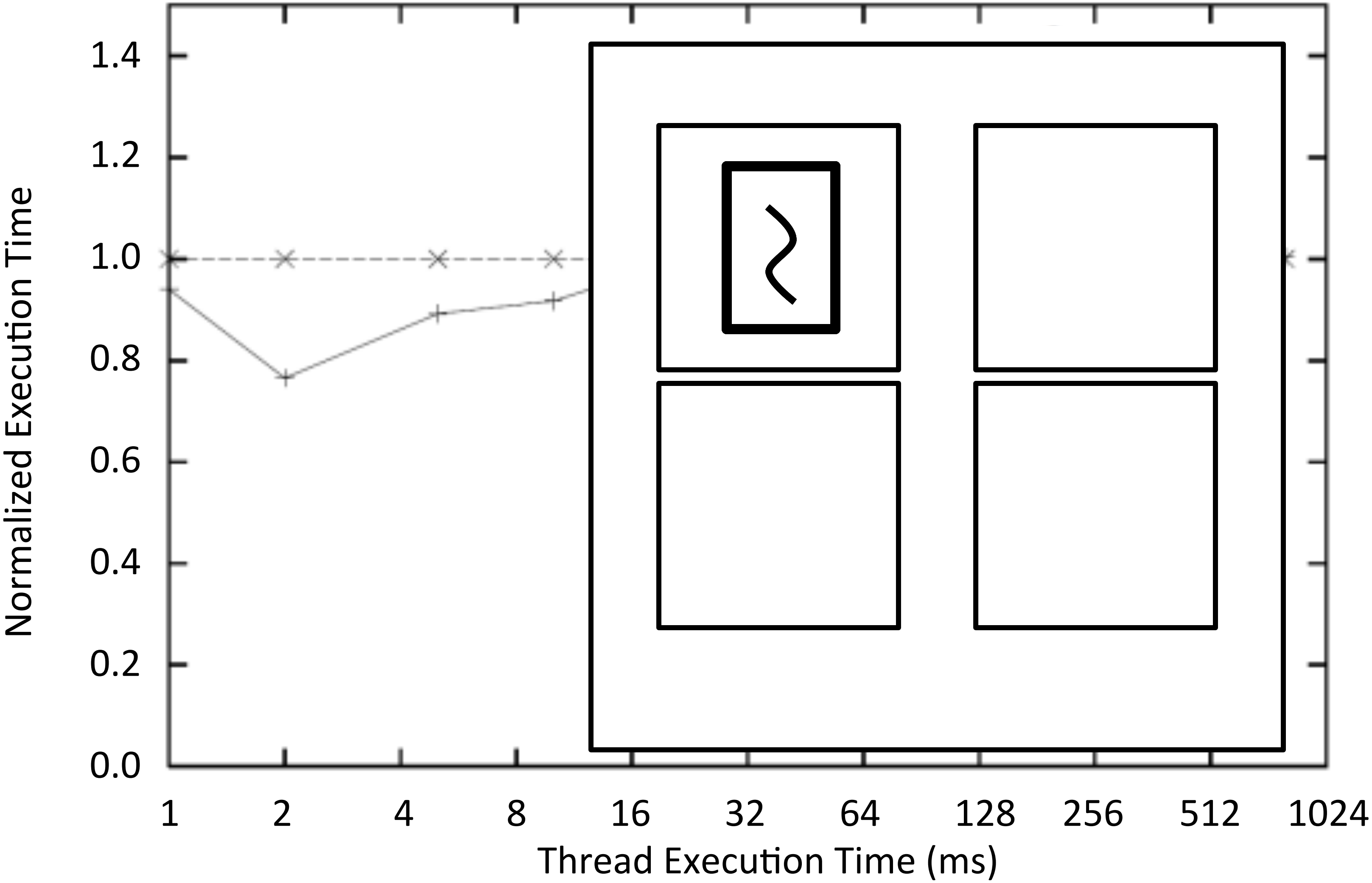
Performance: Processes vs. Threads



Performance: Processes vs. Threads



Performance: Processes vs. Threads

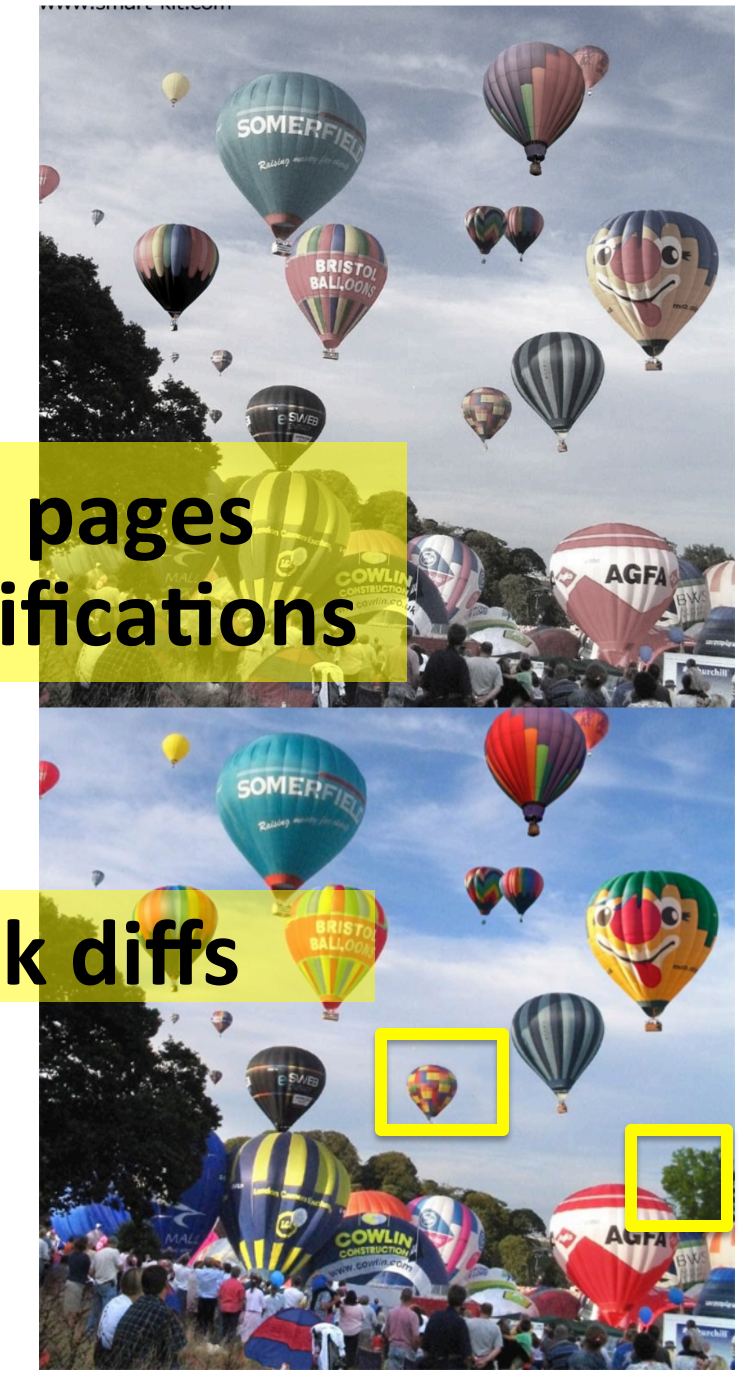


“Shared Memory”

“Shared Memory”

Snapshot pages before modifications

“Shared Memory”



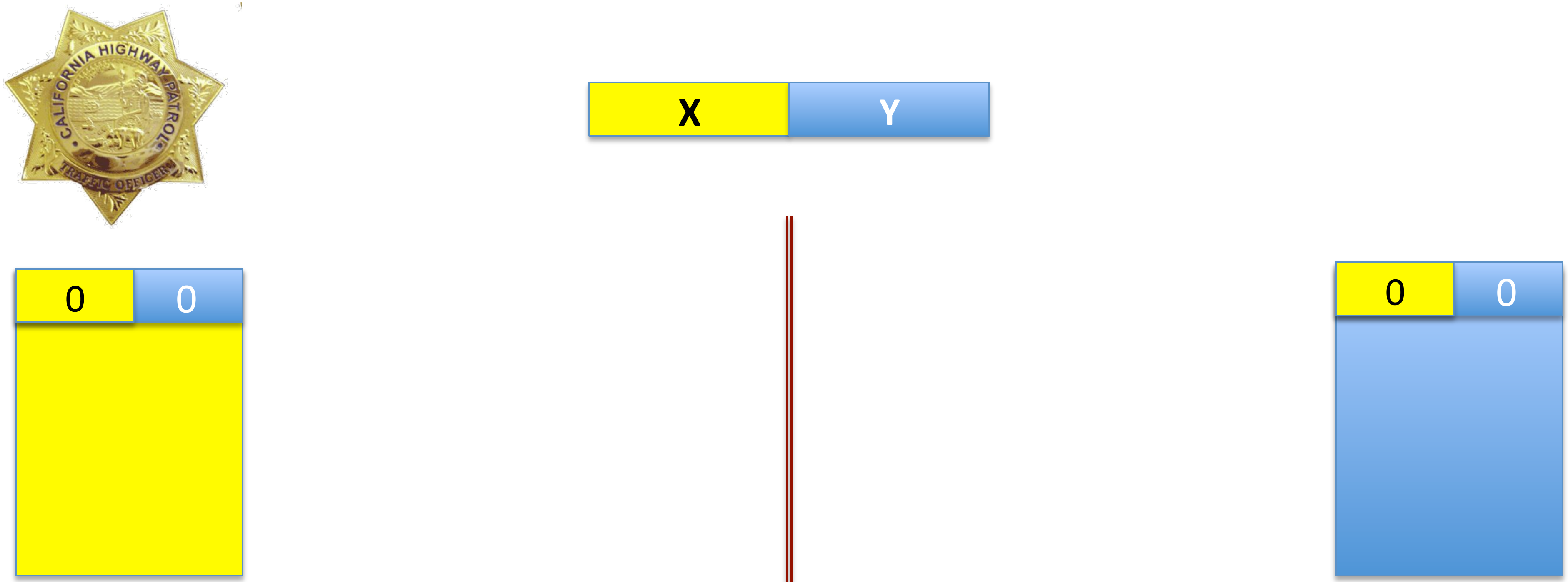
Snapshot pages before modifications

Write back diffs

X	Y

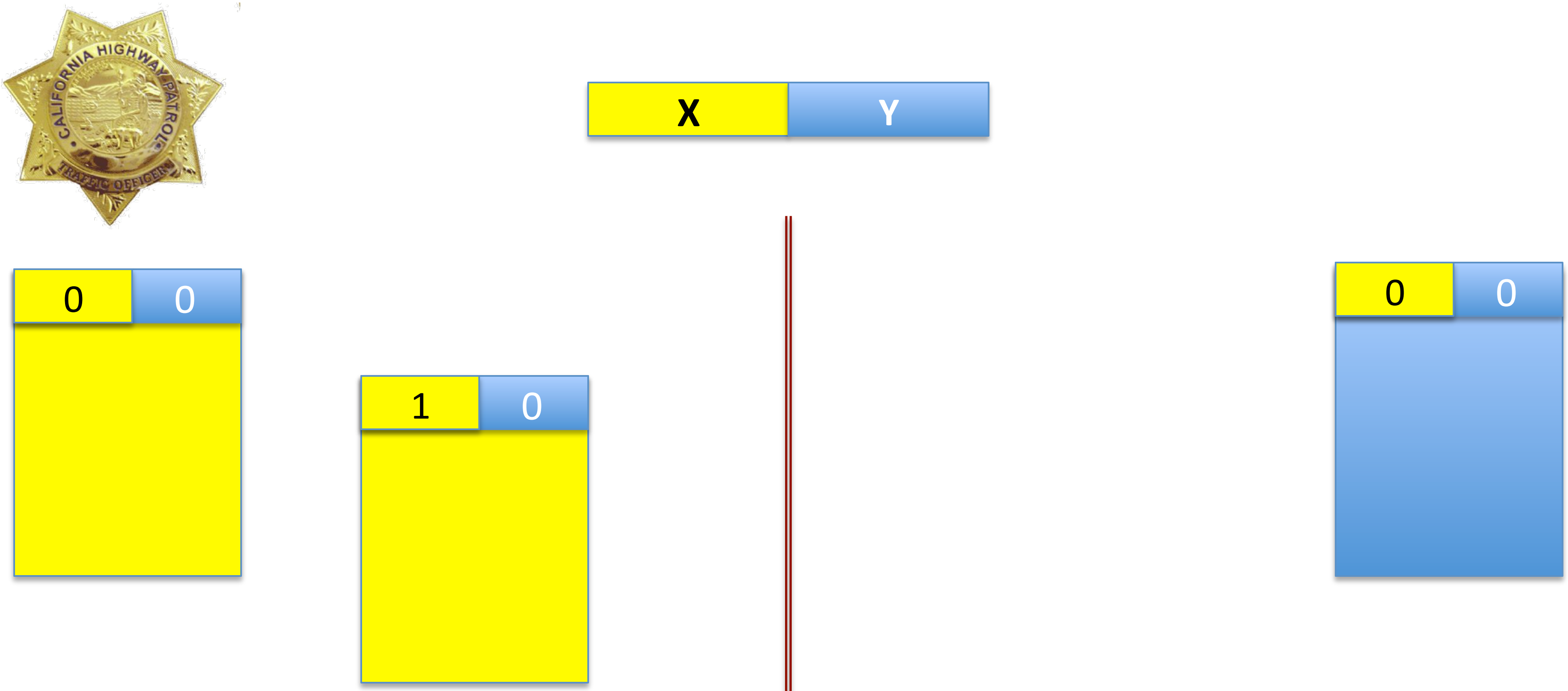
```
for (i = 0; i < M; i++)
{
        x = 1;
}
```

```
for (i = 0; i < M; i++)
{
        y = 2;
}
```



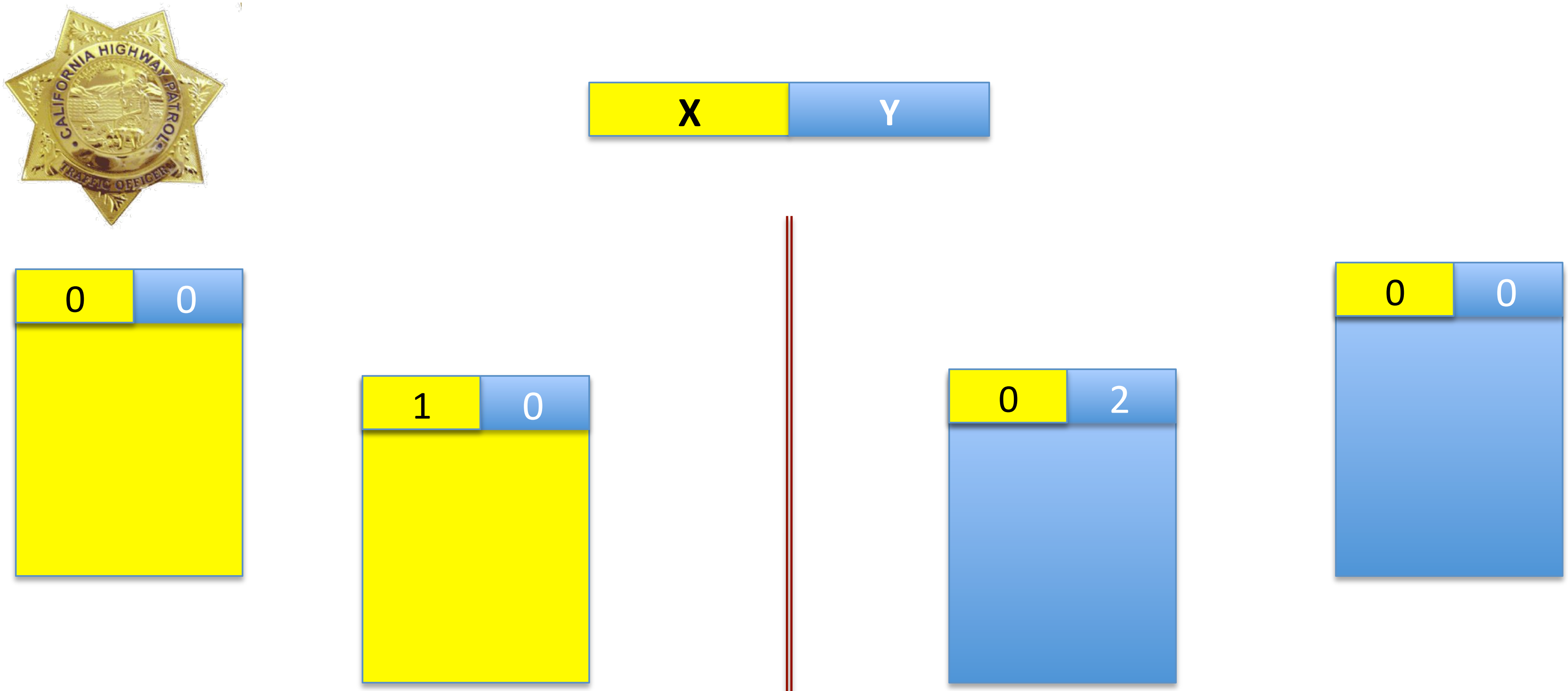
```
for (i = 0; i < M; i++)
{
        x = 1;
}
```

```
for (i = 0; i < M; i++)
{
        y = 2;
}
```



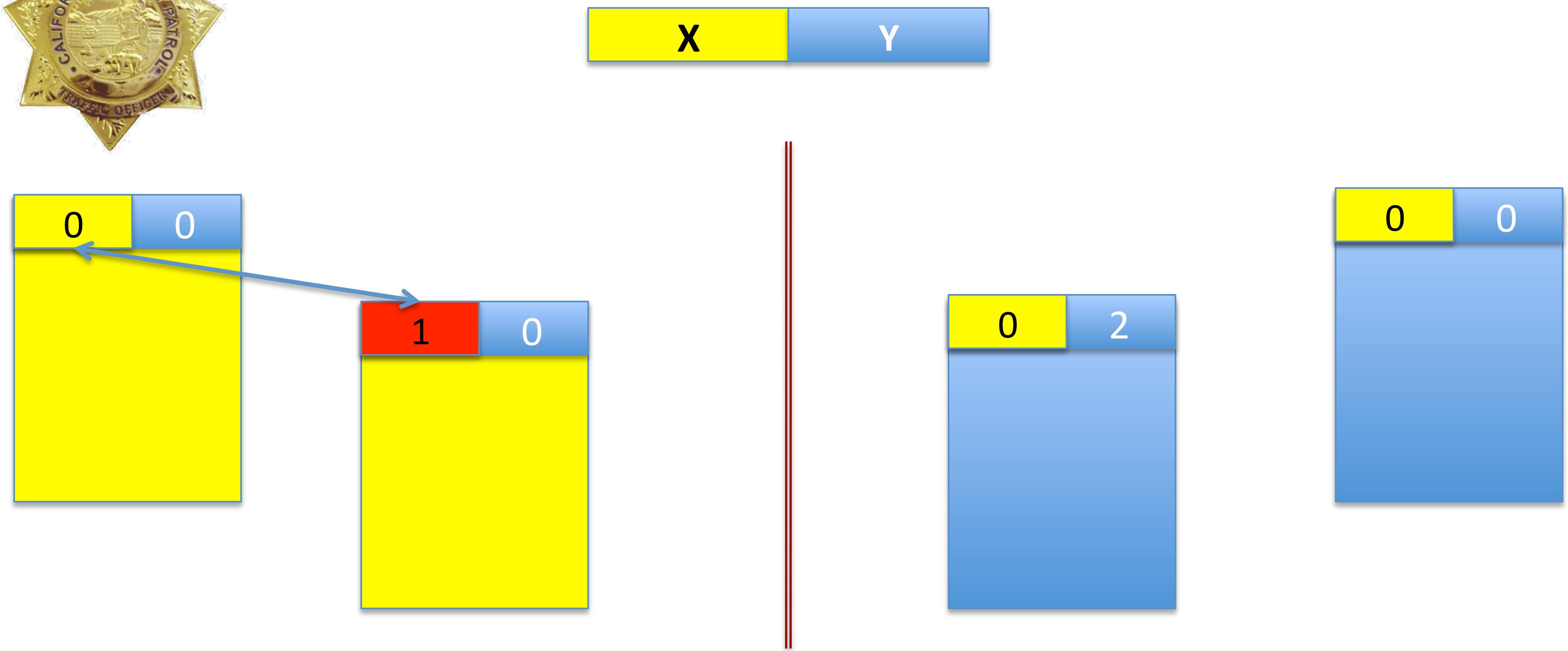
```
for (i = 0; i < M; i++)
{
        x = 1;
}
```

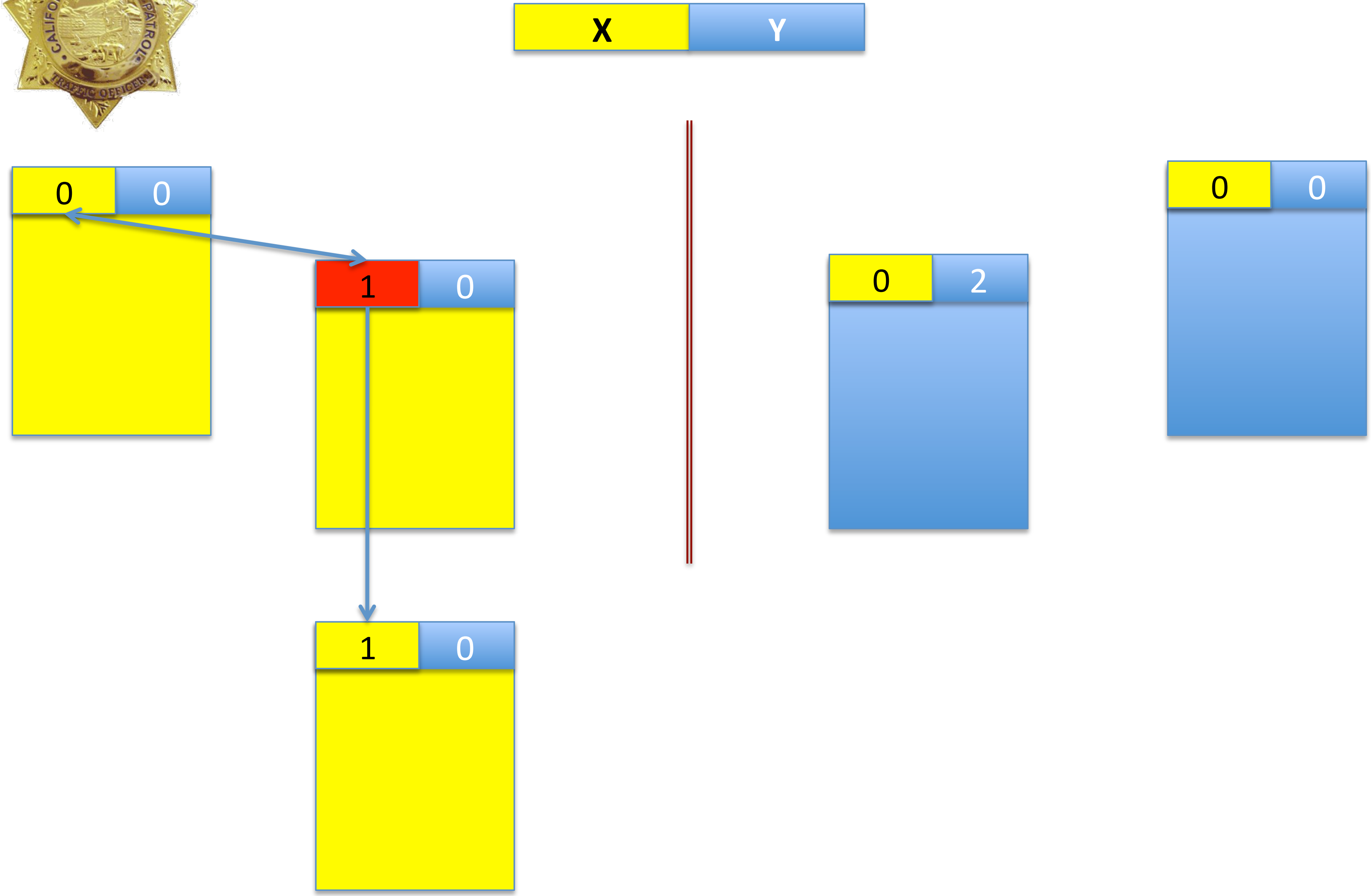
```
for (i = 0; i < M; i++)
{
        y = 2;
}
```



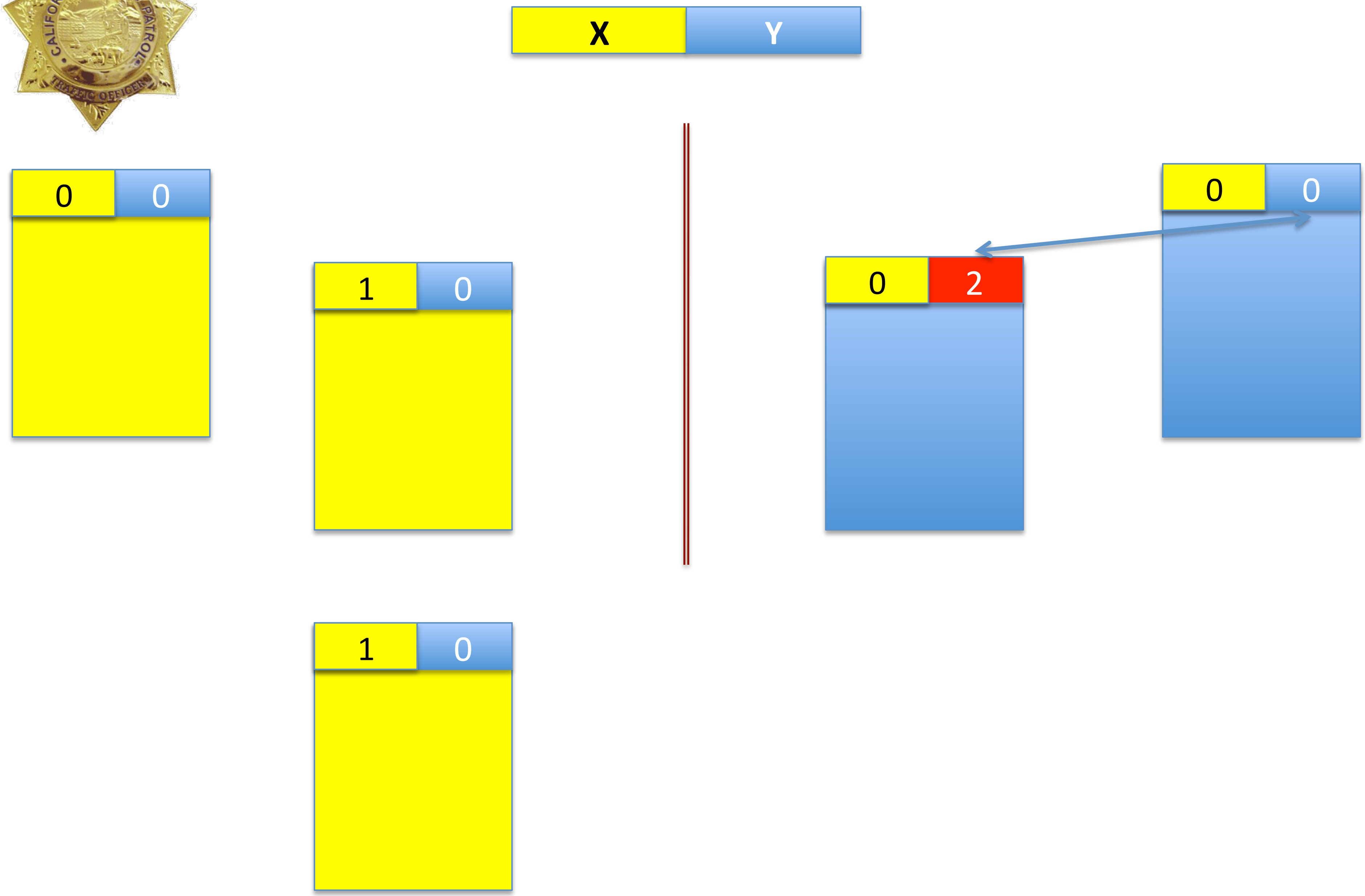
```
for (i = 0; i < M; i++)
{
        x = 1;
}
```

```
for (i = 0; i < M; i++)
{
        y = 2;
}
```


X
Y
0
0
1
0
0
2
0
0



X
Y
0
0
1
0
1
0
0
2
0
0



X Y

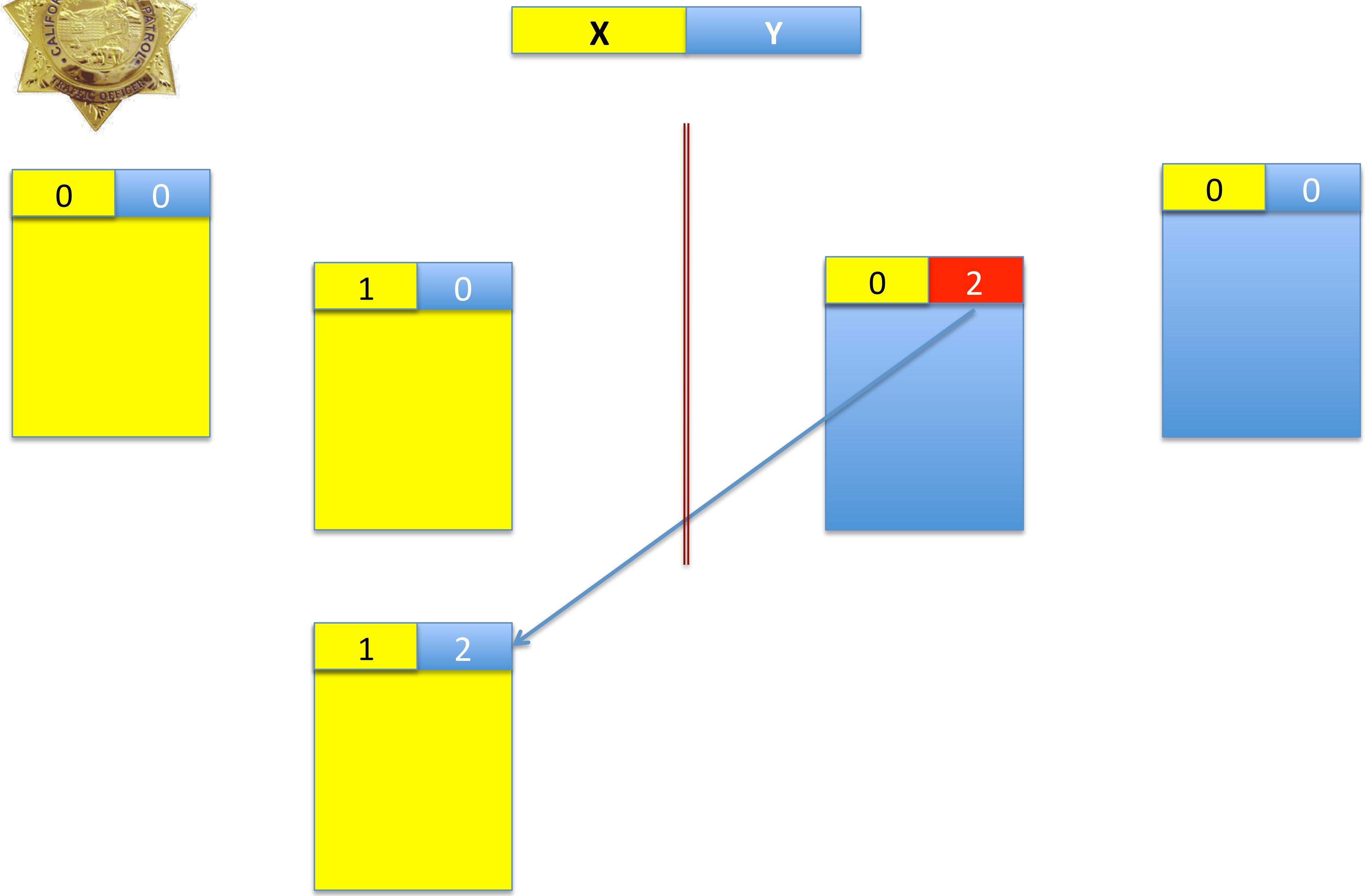
0 0

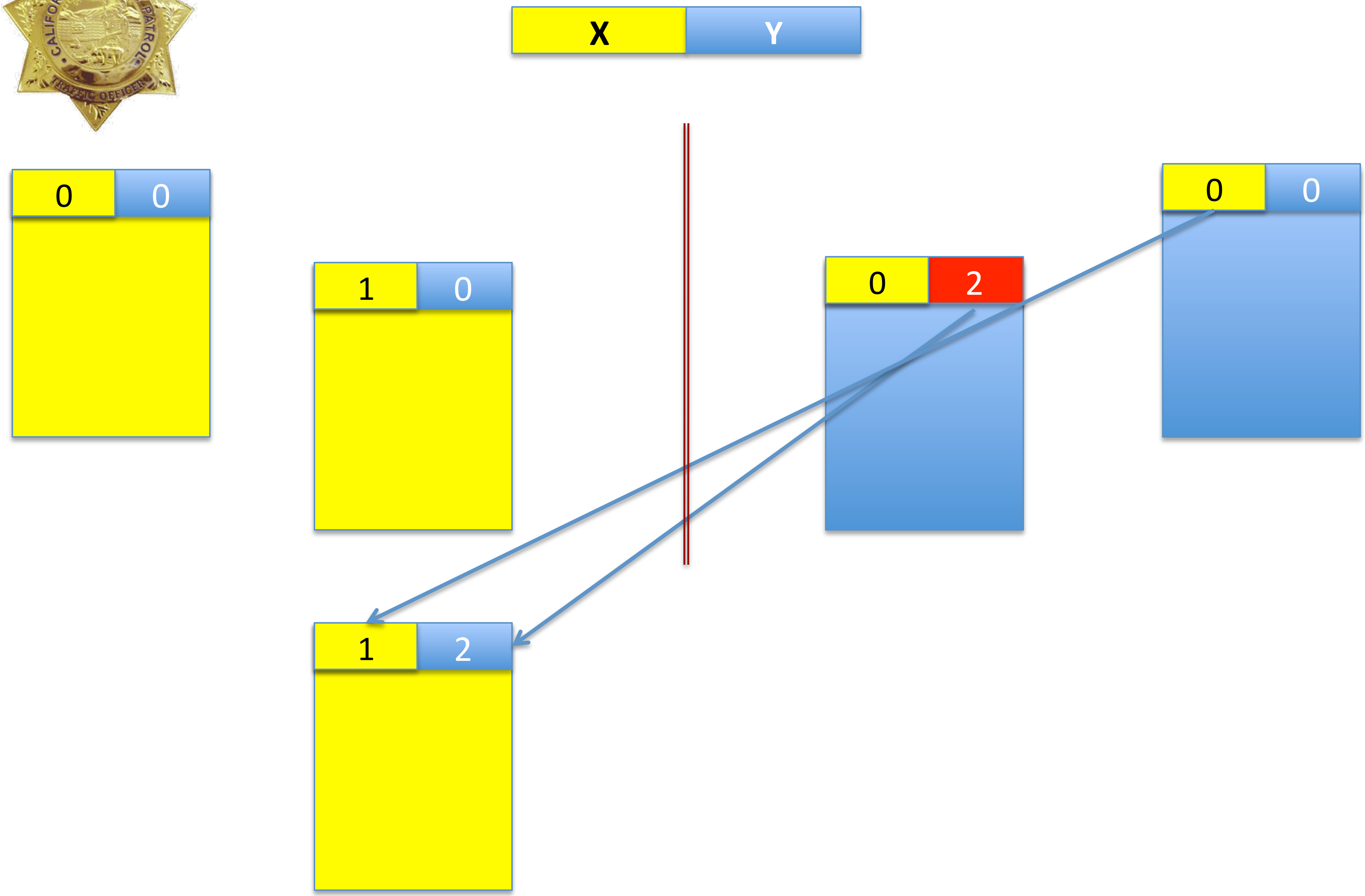
1 0

1 0

0 2

0 0


X
Y
0
0
1
0
0
2
0
0
1
2


X
Y
0
0
1
0
0
2
0
0
1
2

Output: PTU vs. Sheriff-Detect

Benchmarks	**PTU (# shared lines)**	**SHERIFF-DETECT (# shared objects)**
canneal	1	1
kmeans	**1916**	**2**
linear_regression	5	1
matrix_multiply	468	0
pbzip2	14	0
pca	45	0
pfscan	2	0
reverse_index	**N/A**	**5**
streamcluster	9	1
word_count	4	4
swaptions	196	0
Total	**2,664**	**15**

```
typedef struct {
  long long SX;
  long long SY;
  long long SXX;
…
} lreg_args;

void *linear_regression_pthread (void *args_in)
{
   lreg_args* args =(lreg_args*)args_in;
   ……
   for (i = 0; i < args->num_elems; i++)
   {
      args->SX  += args->points[i].x;
      args->SXX += args->points[i].x*args->points[i].x;
      ……
```

```
/home/emery/sheriff$
```

```
/home/emery/sheriff$ gcc lreg.c -lsheriff_detect -o lreg
```

```
/home/emery/sheriff$ gcc lreg.c -lsheriff_detect -o lreg
/home/emery/sheriff$
```

```
/home/emery/sheriff$ gcc lreg.c -lsheriff_detect -o lreg
/home/emery/sheriff$ ./lreg
```

```
/home/emery/sheriff$ gcc lreg.c -lsheriff_detect -o lreg
/home/emery/sheriff$ ./lreg
…
Sheriff-Detect has detected false sharing.
Object 1: Allocation call stack:
   0: lreg.c: line number: 136
```

```
/home/emery/sheriff$ gcc lreg.c -lsheriff_detect -o lreg
/home/emery/sheriff$ ./lreg
…
Sheriff-Detect has detected false sharing.
Object 1: Allocation call stack:
   0: lreg.c: line number: 136
```

(1) Examine allocation site

```
/home/emery/sheriff$ gcc lreg.c -lsheriff_detect -o lreg
/home/emery/sheriff$ ./lreg
…
Sheriff-Detect has detected false sharing.
Object 1: Allocation call stack:
  0: lreg.c: line number: 136
```

(1) Examine allocation site

```
136 tid_args = (lreg_args *)calloc(sizeof(lreg_args),
num_procs);
```

(2) Find references

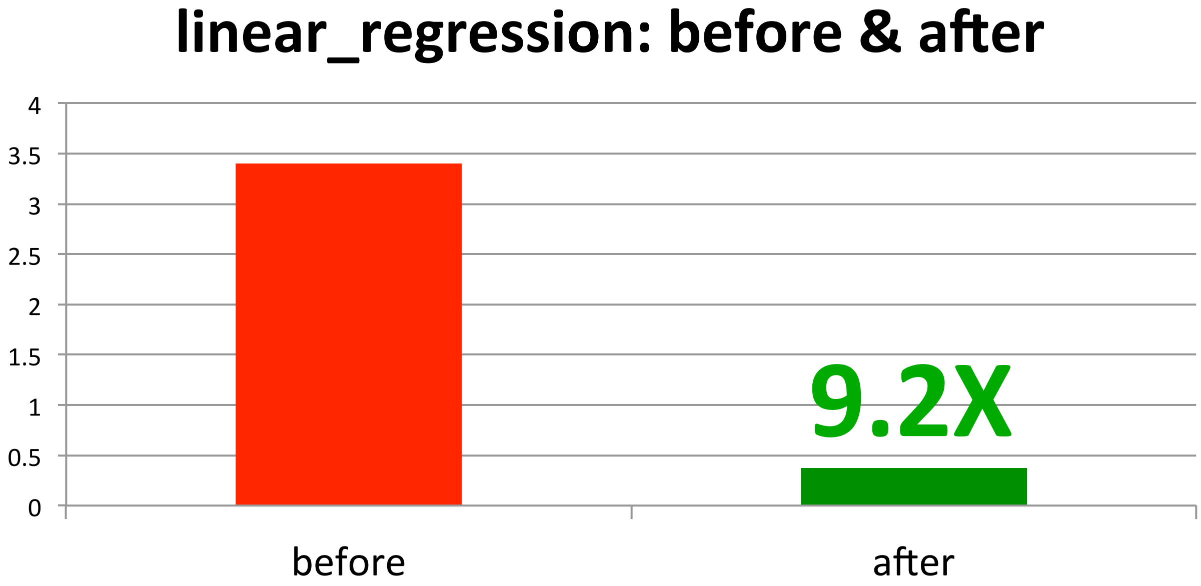
```
152 pthread_create(&tid_args[i].tid, &attr,
linear_regression_pthread, (void*)&tid_args[i]) != 0);
```

```
void *linear_regression_pthread(void *args_in)
{
   lreg_args* args =(lreg_args*)args_in;
   ……
   for (i = 0; i < args->num_elems; i++)
   {
      args->SX  += args->points[i].x;
      args->SXX += args->points[i].x*args->points[i].x;
      ……
```

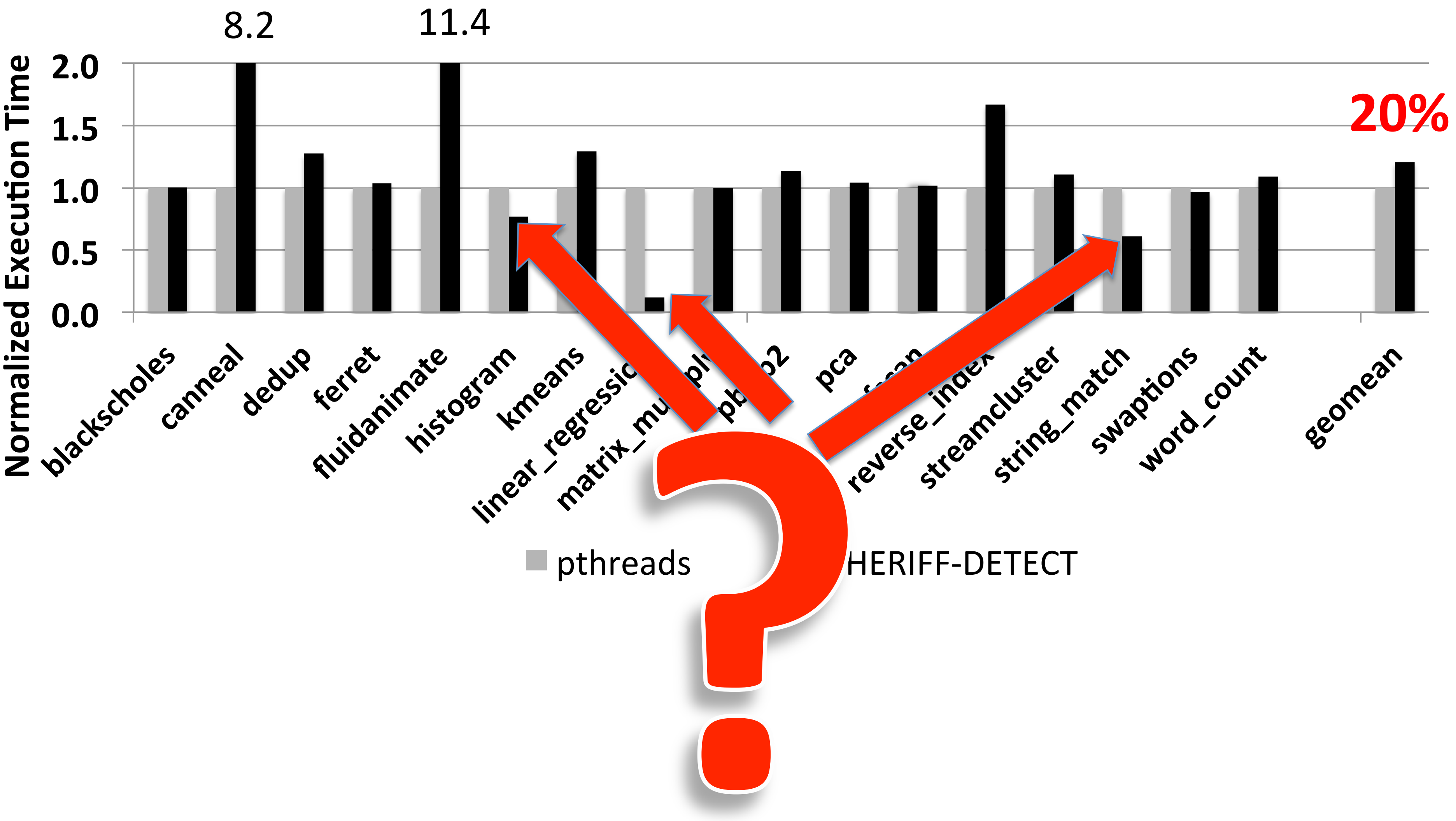
“lreg_args” is not cache-line aligned

(3) Fix false sharing (here: *padding*)

```
typedef struct {
  …..
  char padding[128];  // Padding to avoid false sharing
} lreg_args;
```



SHERIFF-DETECT performance



JAKE GYLLENHAAL
MICHELLE MONAGHAN
VERA FARMIGA
SOURCE CODE
FROM THE DIRECTOR OF MOON
VENDÔME PICTURES PRESENTS A MARK GORDON COMPANY PRODUCTION

SHERIFF-PROTECT

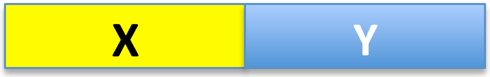
Prevents **all** false sharing

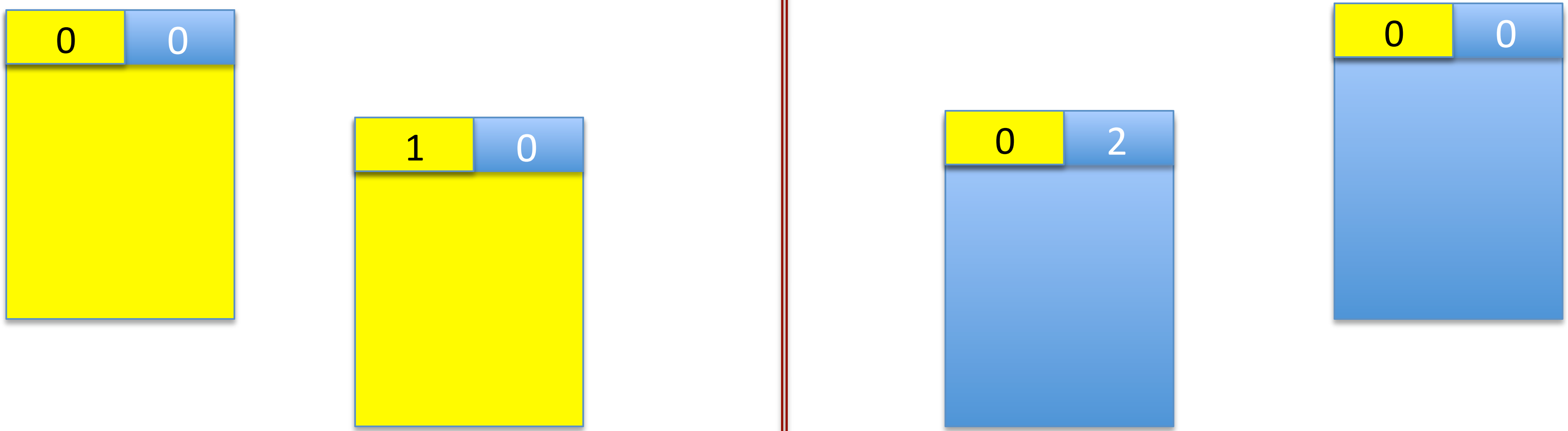
=

−

SHERIFF-PROTECT

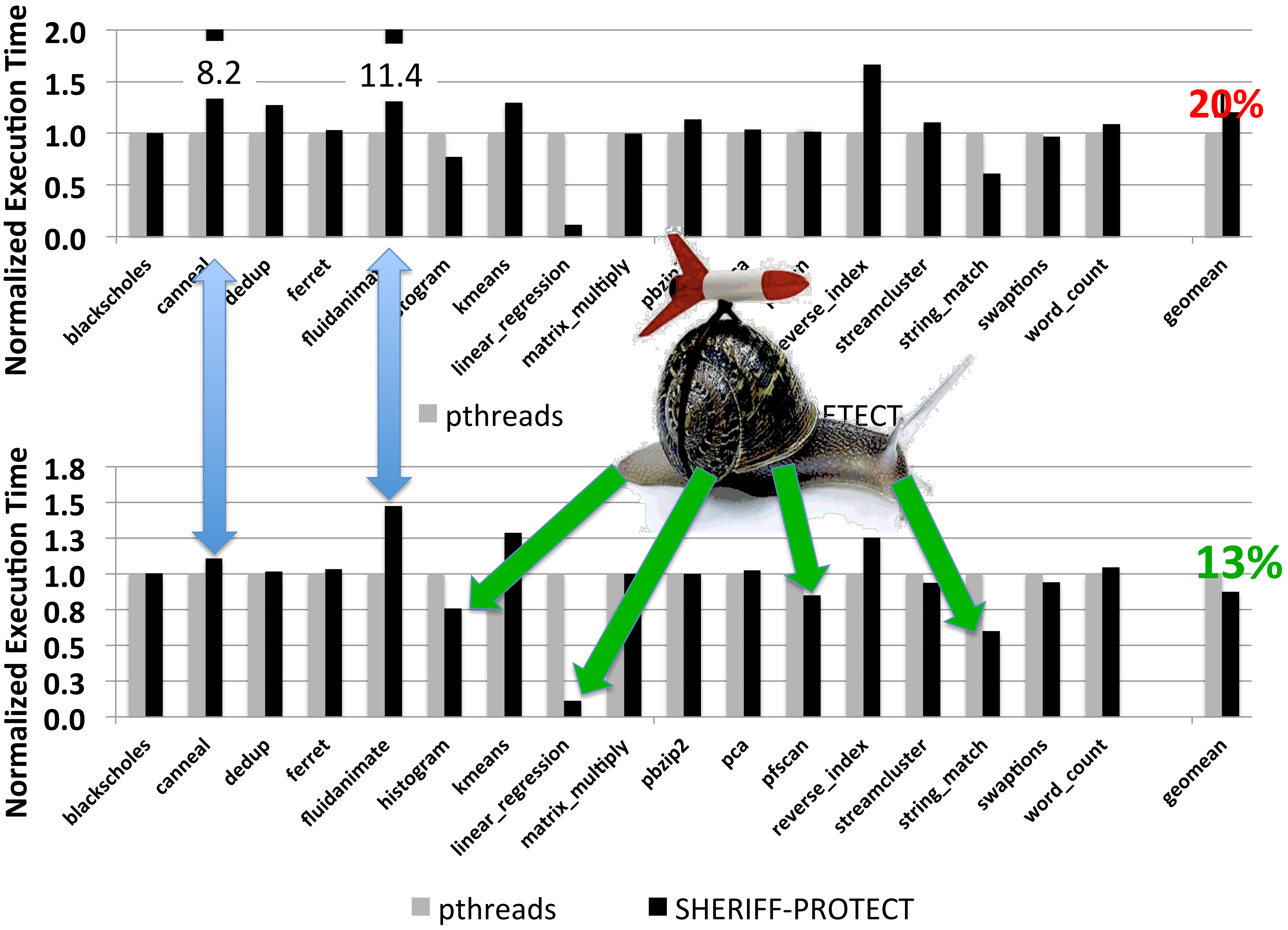
SHERIFF-DETECT





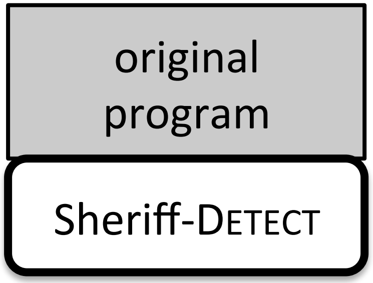
```
for (i = 0; i < M; i++)
{
        x = 1;
}
```

```
for (i = 0; i < M; i++)
{
        y = 2;
}
```


Normalized Execution Time
2.0
1.5
1.0
0.5
0.0
8.2
11.4
20%
blackscholes
canneal
dedup
ferret
fluidanimate
histogram
kmeans
linear_regression
matrix_multiply
pbzip2
pca
pfscan
reverse_index
streamcluster
string_match
swaptions
word_count
geomean
pthreads
SHERIFF-PROTECT
Normalized Execution Time
1.8
1.5
1.3
1.0
0.8
0.5
0.3
0.0
13%
blackscholes
canneal
dedup
ferret
fluidanimate
histogram
kmeans
linear_regression
matrix_multiply
pbzip2
pca
pfscan
reverse_index
streamcluster
string_match
swaptions
word_count
geomean
pthreads
SHERIFF-PROTECT

Sheriff-Detect

false sharing detection

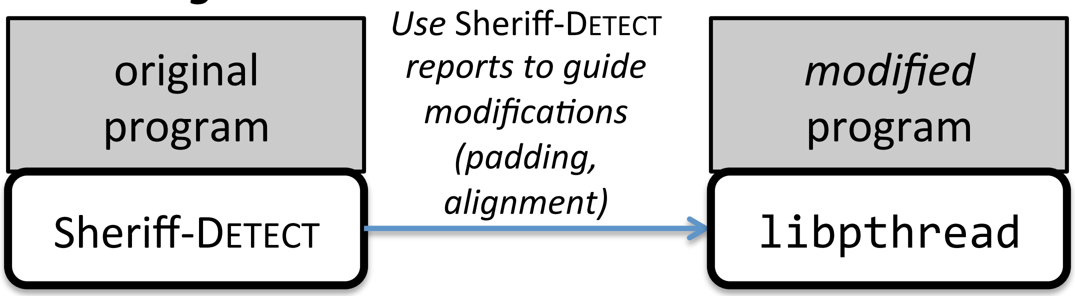


original program

Sheriff-Detect

SHERIFF-DETECT

false sharing detection



original program

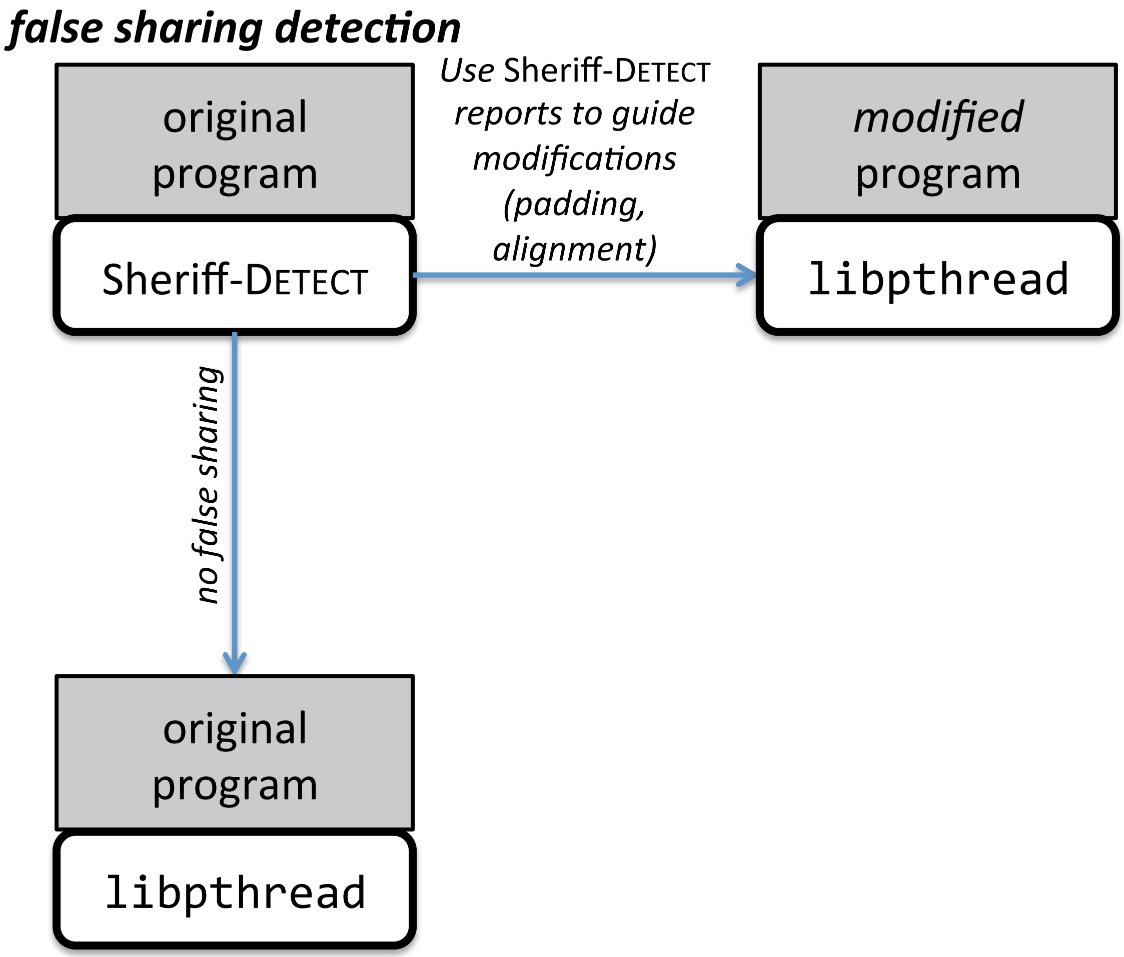
Sheriff-DETECT

Use Sheriff-DETECT *reports to guide modifications (padding, alignment)*

modified program

`libpthread`

Sheriff-Detect



false sharing detection

original program

Sheriff-Detect

Use Sheriff-Detect *reports to guide modifications (padding, alignment)*

modified program

`libpthread`

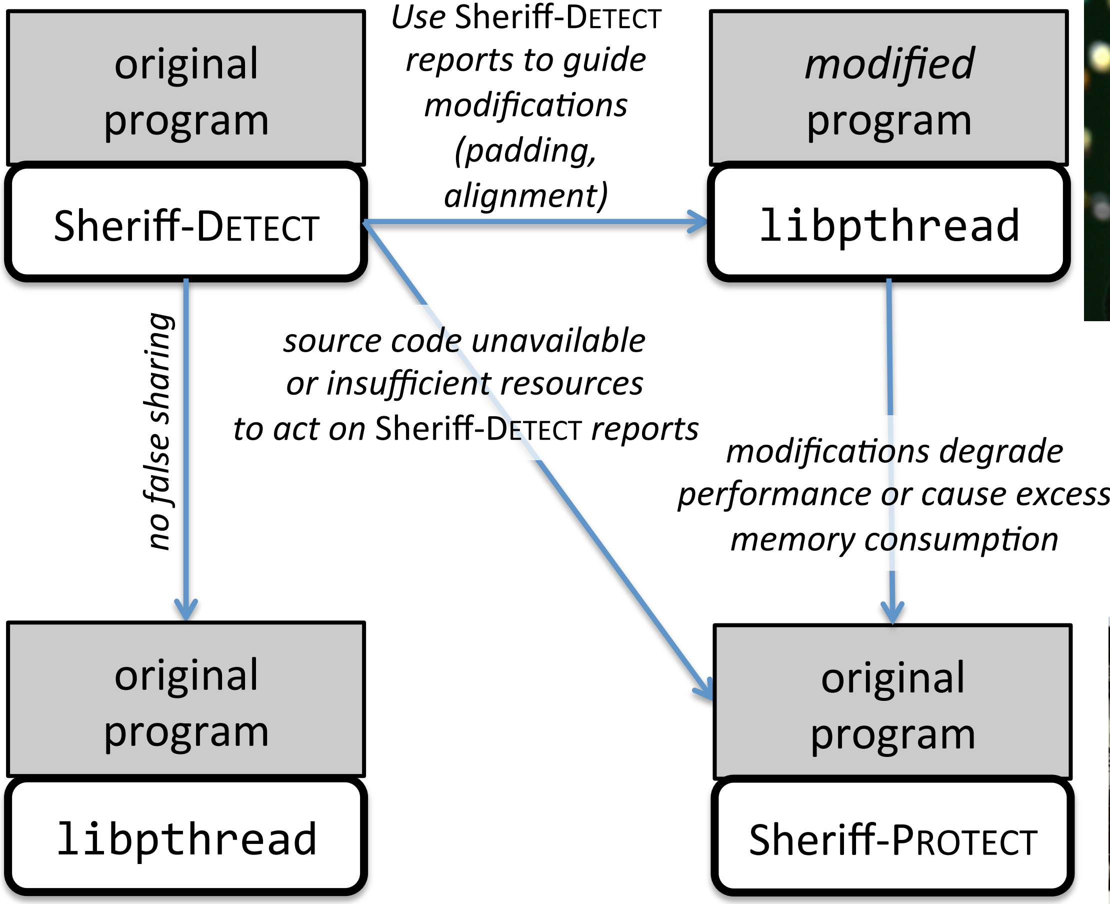
no false sharing

original program

`libpthread`

SHERIFF-DETECT

false sharing detection



original program

Sheriff-DETECT

Use Sheriff-DETECT *reports to guide modifications (padding, alignment)*

modified program

libpthread

no false sharing

original program

libpthread

source code unavailable or insufficient resources to act on Sheriff-DETECT *reports*

modifications degrade performance or cause excess memory consumption

original program

Sheriff-PROTECT

false sharing mitigation

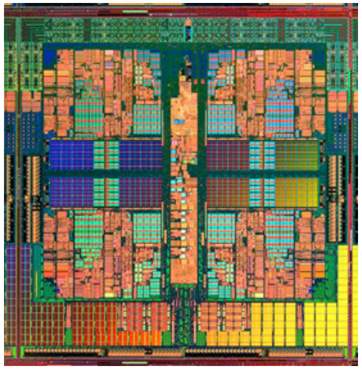
SHERIFF-PROTECT

Ford
R. Montier & Cie
19
E. Bruet
R. Coucher
Martin
RICHARD MILLE

BOSCH
Microsoft
Energizer
TARGE
stone
EXMARK
PURINA
Energizer
reveal
Gillette

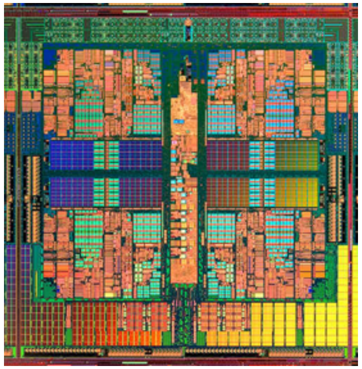
33
TARGET
Pioneer
51

33
PEAK

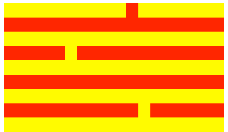


Dthreads

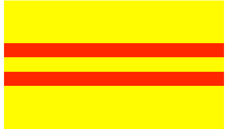
```
% g++ myprog.cpp -ldthreads   -o myprog
```



Dthreads

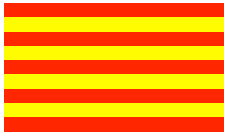


race conditions

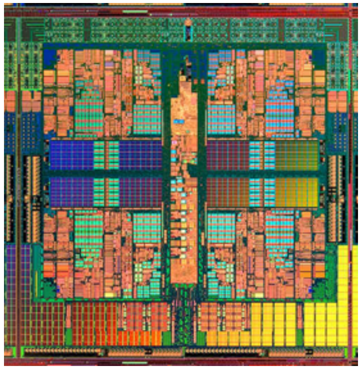


atomicity violations

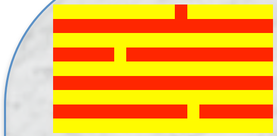
deadlock



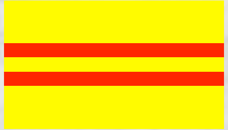
order violations



Dthreads

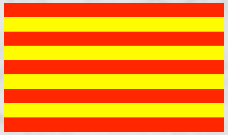


race conditions

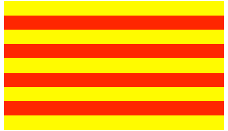


atomicity violations

deadlock



order violations

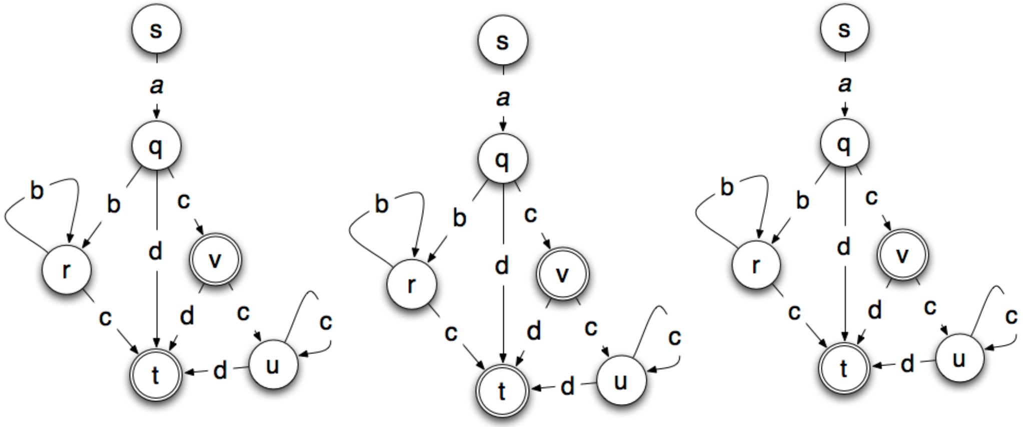


deterministic

DTHREADS = Deterministic Execution =

Race-free Executions

Replay Debugging w/o Logging



Replicated State Machines

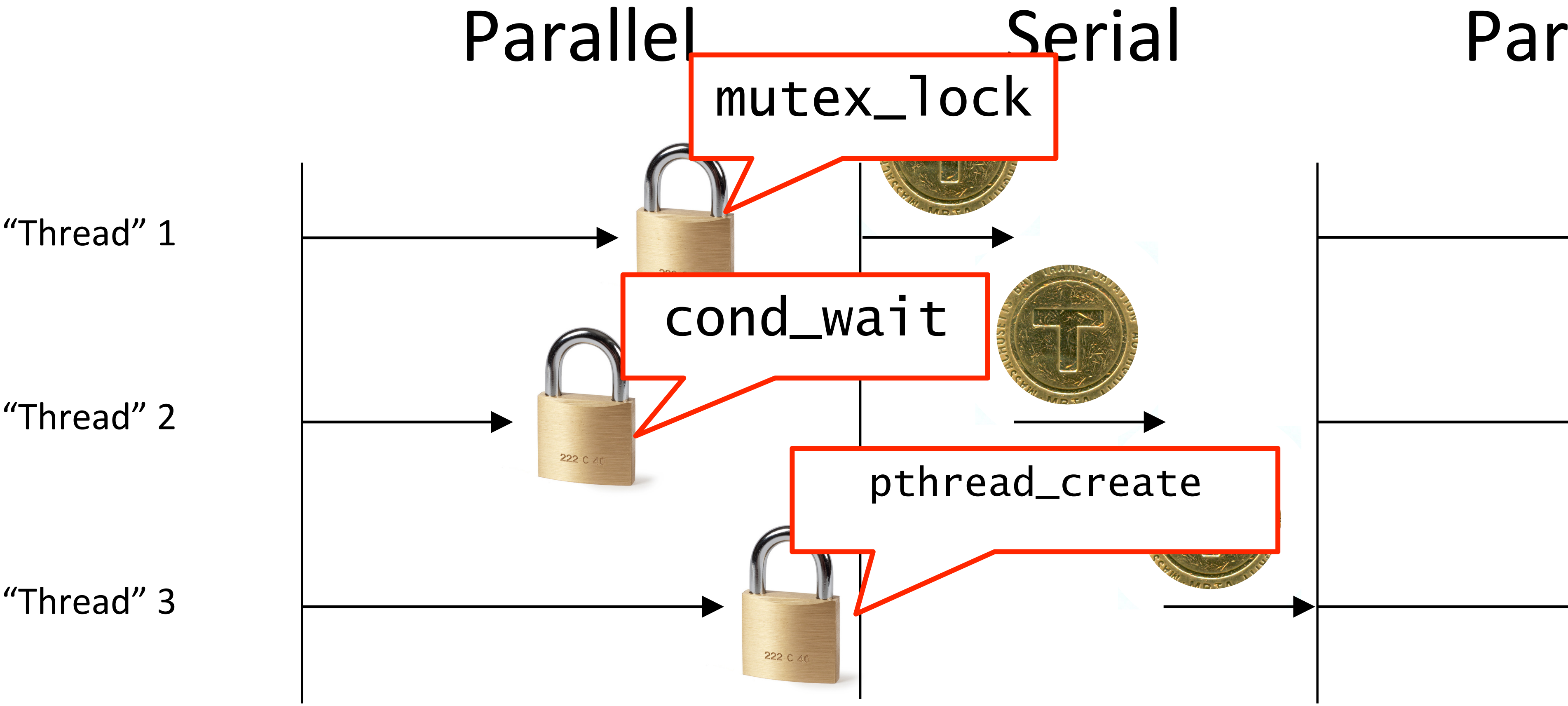
Debuggable Races

Debuggable Races

Debuggable Races

Debuggable Races

Update in Deterministic Time & Order



Parallel

Serial

Par

mutex_lock

cond_wait

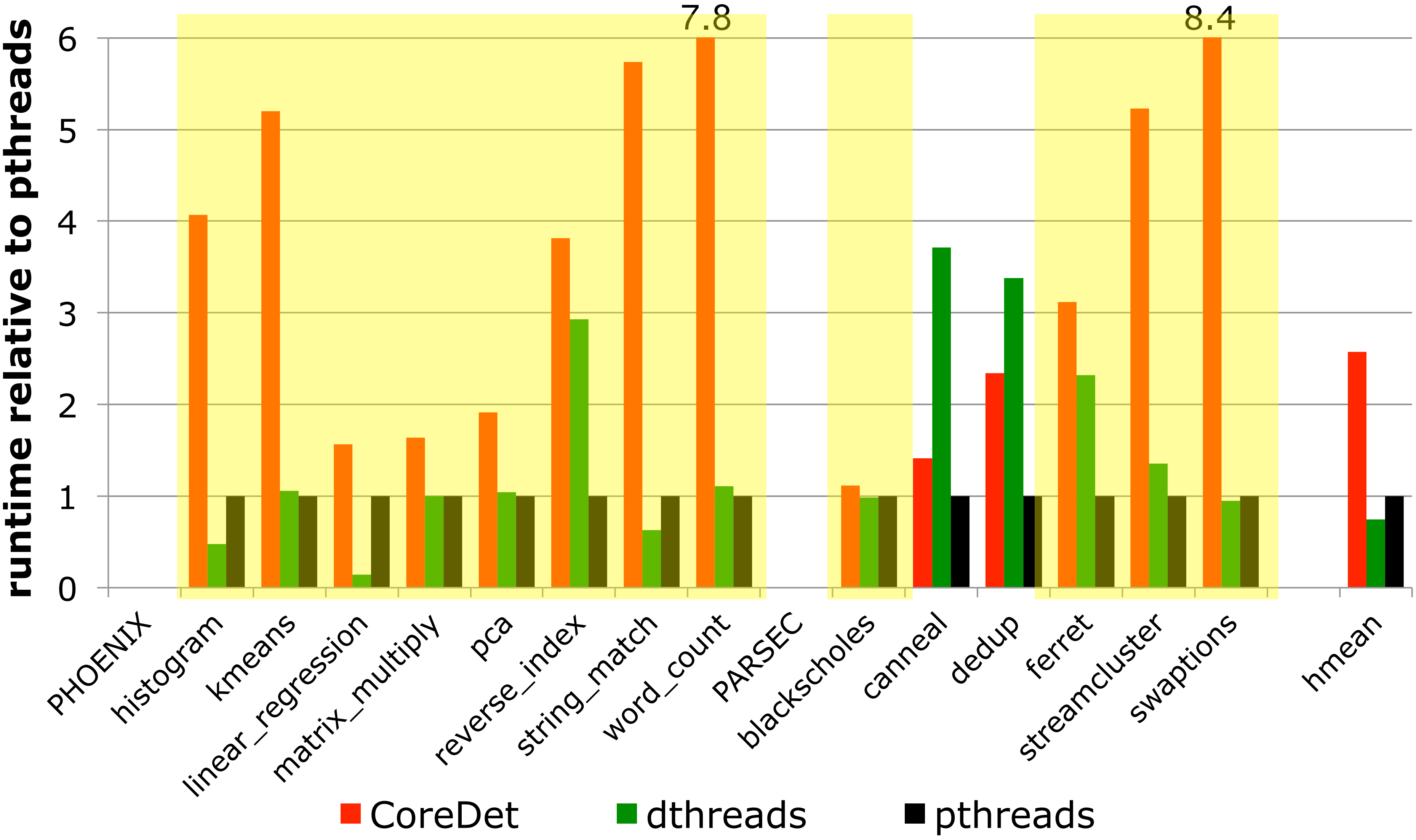
pthread_create

“Thread” 1

“Thread” 2

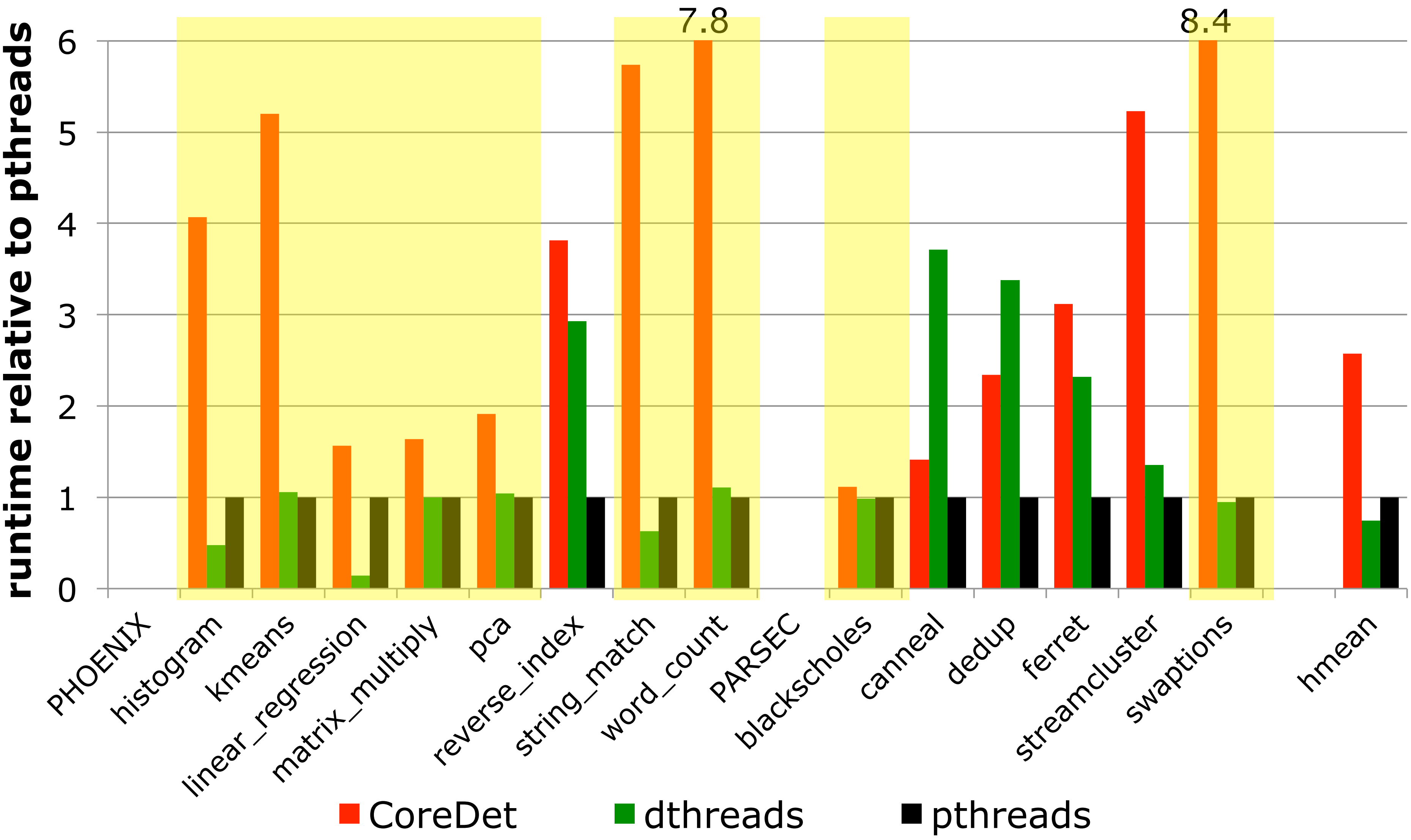
“Thread” 3

DTHREADS: Efficient Determinism



Usually faster than the state of the art

DTHREADS: Efficient Determinism



Generally as fast or faster than pthreads

race conditions

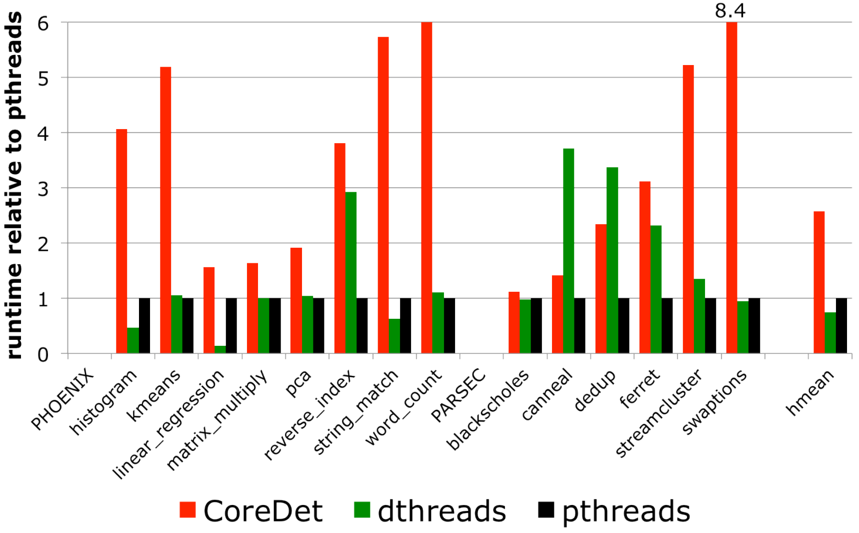
atomicity violations

deadlock

order violations

deterministic

DTHREADS



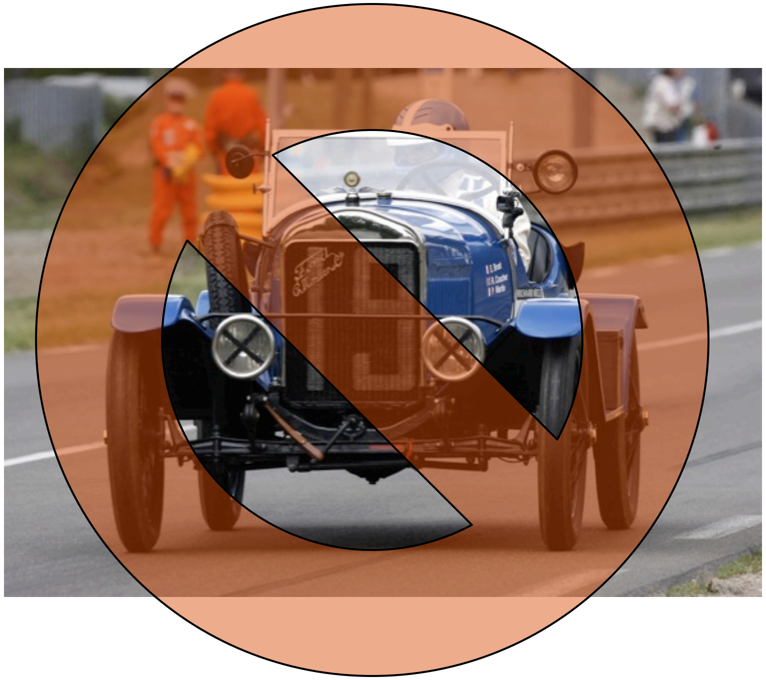
```
% g++ myprog.cpp -lpthread
```

SHERIFF-DETECT

SHERIFF-PROTECT

SHERIFF-DETECT

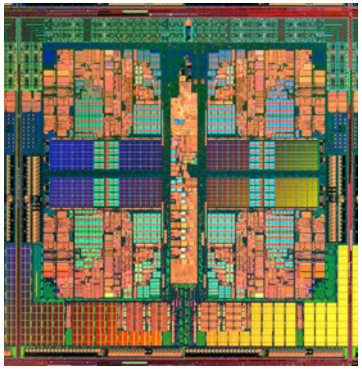
SHERIFF-PROTECT



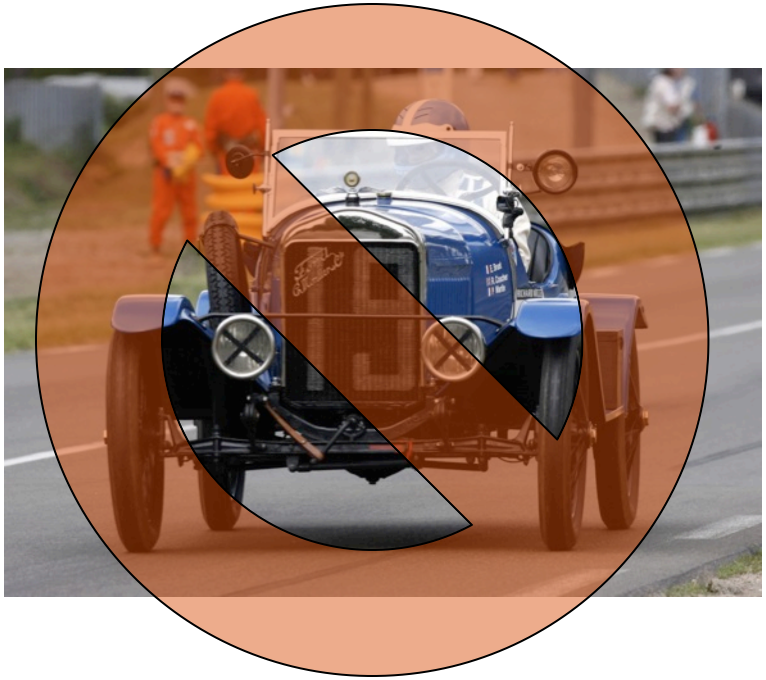
[OOPSLA 2011]

SHERIFF-DETECT

SHERIFF-PROTECT



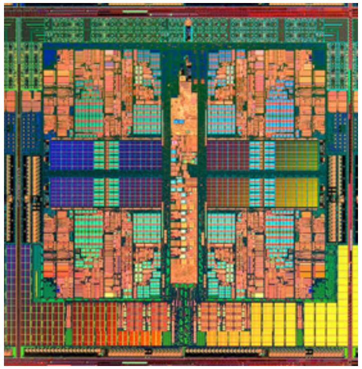
DTHREADS



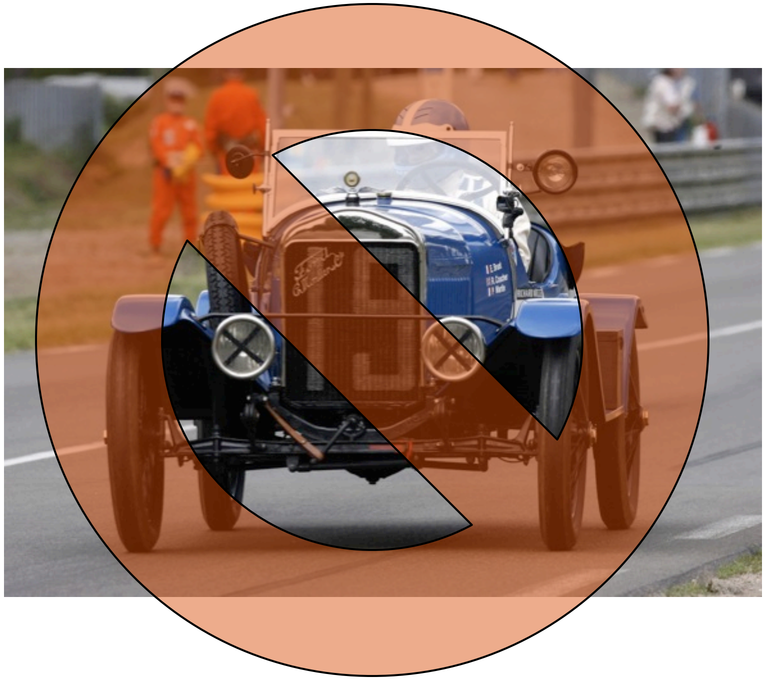
[OOPSLA 2011]

SHERIFF-DETECT

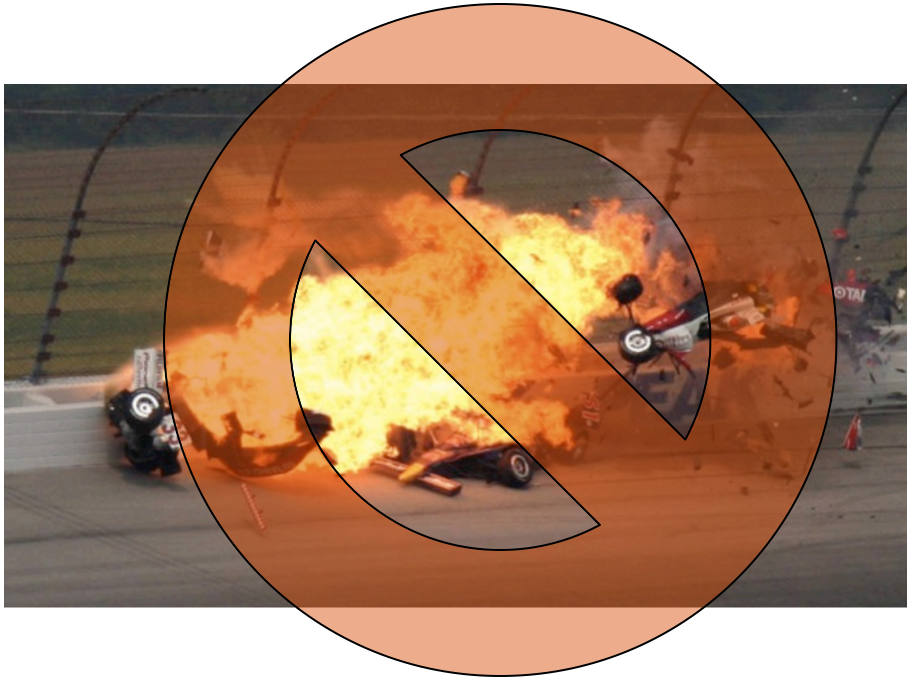
SHERIFF-PROTECT



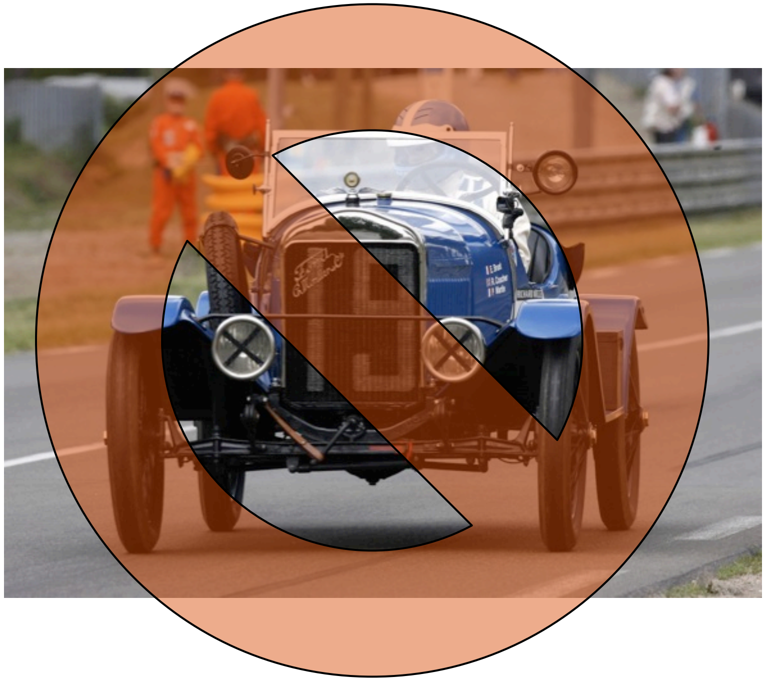
DTHREADS

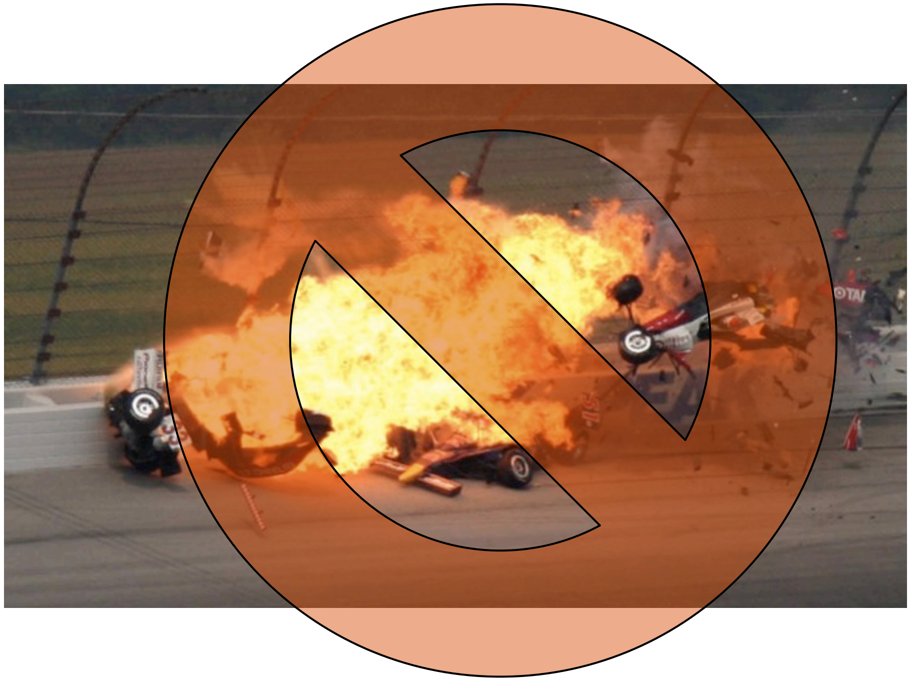


[OOPSLA 2011]



[SOSP 2011]





https://github.com/plasma-umass/sheriff

https://github.com/plasma-umass/dthreads