IRL P Programming

Erik Hagersten
Uppsala University

UPPSALA
UNIVERSITET

- CPU/Pipeline videos asap
- End game: Multicores, GPUs, Future
- Reading instructions, X-tenta etc.

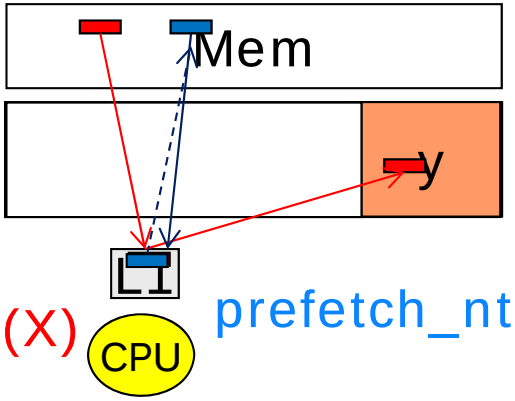
www.it.uu.se/edu/course/homepage/avdark/ht13/

- PhD student?
- X-jobb?

AVDARK
2013

Discussing Quizzes

Cache Waste




```
/* Unoptimized */
for (s = 0; s < ITERATIONS; s++){
   for (j = 0; j < HUGE; j++)
        x[j] = x[j+1];          /* will hog the cache but not benefit*/
   for (i = 0; i < SMALLER_THAN_L2_CACHE; i++)
        y[i] = y[i+1];          /* will be evicted between usages /*
}
/* Optimized */
for (s = 0; s < ITERATIONS; s++){
   for (j = 0; j < HUGE; j++) {
        PREFETCH_NTA x[j+1] /* will be installed
         x[j] = x[j+1];
   for (i = 0; I < SMALLER_THAN_L2_CACHE; i++)
        y[i] = y[i+1];      /* will always hit in the cache*/
}
```

What kind of misses are removed?

- ❑ Capacity
- ❑ Conflict
- ❑ Compulsory
- ❑ Coherence

➔ Can also be beneficial if applications are co-scheduled on MC and share cache with other applications.

Avoiding coherence traffic

UPPSALA UNIVERSITET

What kind of misses are removed?

- ❑ Capacity
- ❑ Conflict
- ❑ Compulsory
- ❑ Coherence

ORIG:

Thread 0:

```
int a, i, total;
spawn_child()
for (int i; i< HUGE; i++) {
  /* do some work */
   a++;
 }
join()
total = a;
```

Child:

```
int i;
for (int i; i< HUGE; i++) {
   /* do some work*/
   a++;
 }
end_child()
```

OPT:

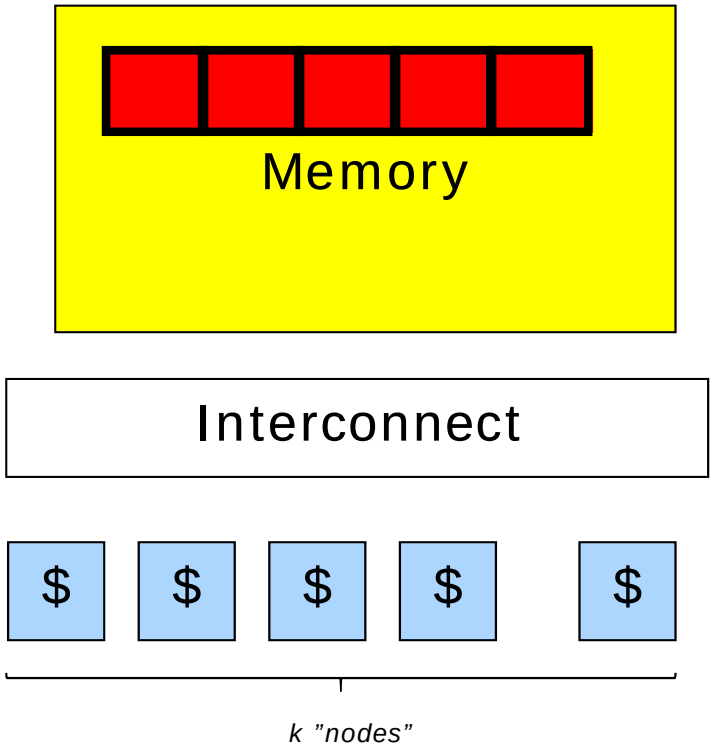
Thread 0:

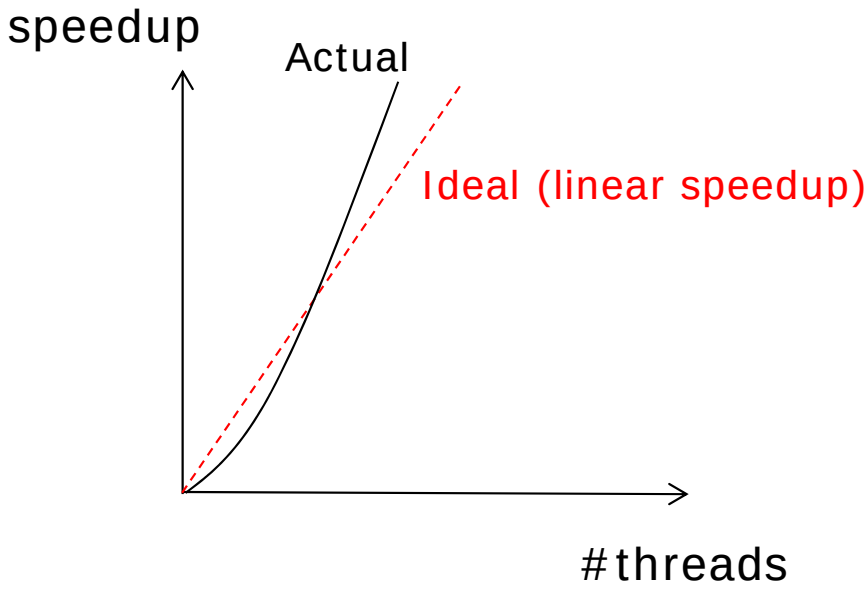
```
int a, i, total;
spawn_child()
for (int i; i< HUGE; i++) {
   /* do some work */
   a++;
 }
join()
total += a;
```

Child:

```
int b, i;
for (int i; i< HUGE; i++) {
   /* do some work */
   b++;
 }
total = b;
end_child()
```

Super-linear speedup





What kind of scaling will most likely see super-linear speedup?

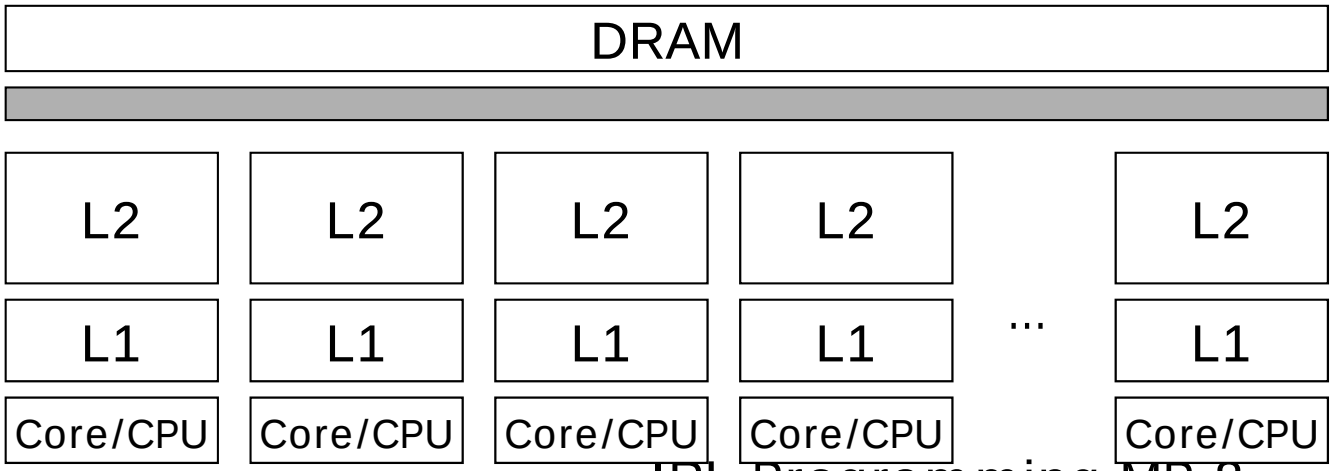
- ❑ Strong scaling
- ❑ Weak scaling
- ❑ Mixed throughput workloads

Problems / Discussion

UPPSALA UNIVERSITET

Assumptions

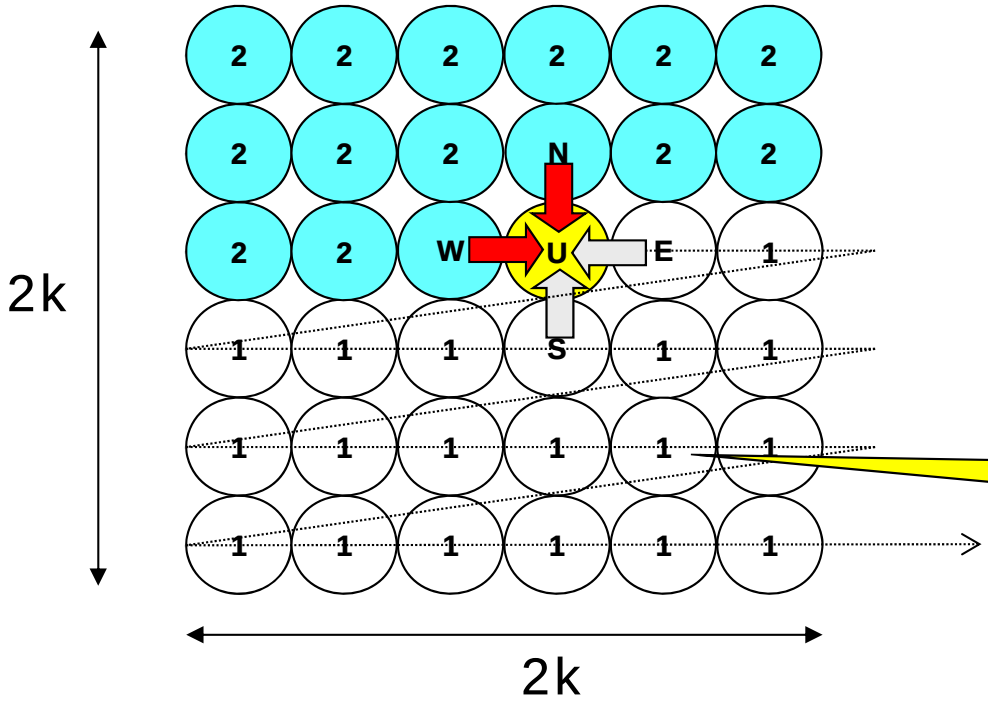
- ”Perfect” 3MHz pipeline with CPI =1,0 for ALU operations and L1 cache hits
- Store buffer hides write latency
- L1 = 32kB, CL=64B, 8-way LRU
- L2 = 2MB, CL=64B, 8-way LRU (HW pref optional)
- L2 cache read: 10 extra cycles latency (over L1)
- L2 Cache2cache: extra 100 cycles latency
- DRAM: 200 extra cycles latency (over L1)



Gauss Seidel, natural order

No HW pref

How many cycles per instruction (CPI)?
What DRAM bandwidth?



Reading old data
Reading new data
X Iterated X times
Execution path
Cell updated now

Double precision FP
8 FP (64b) per CL (64B)

Line of thought:
instr / loop
L1/L2 miss/loop
Avg cycles/loop
➔ CPI
Cycles/miss
DRAM accesses/s
➔ BW

```
for i = 0 to 2k
   for j = 0 to 2k {
      tmp = a[i,j];
      a[i,j] = a[i,j]+a[i-1,j]+a[i,j-1]+a [i+1,j]+a[i,j+1];
      convergence += abs(tmp-a[i,j]);
   }
```

UPPSALA UNIVERSITET

```
for i = 0 to 2k                                                    // (2 ALU  -- rare)
  for j = 0 to 2k {                                                // 2 ALU
    tmp = a[i,j]                                                   // 1LD
    a[i,j] = a[i,j]+a[i-1,j]+a[i,j-1]+a [i+1,j]+a[i,j+1]           // 5LD, 1ST, 7ALU
    convergence += abs(tmp-a[i,j])                                 // 3 ALU
  }
```

GS natural

Cache misses:

S: L2 miss 1/8, L1 hit 7/8
E: L2 hit 1/8, L1 hit 7/8
N: L2 hit 1/8, L1 hit 7/8
U: L1 hit
W: L1 hit

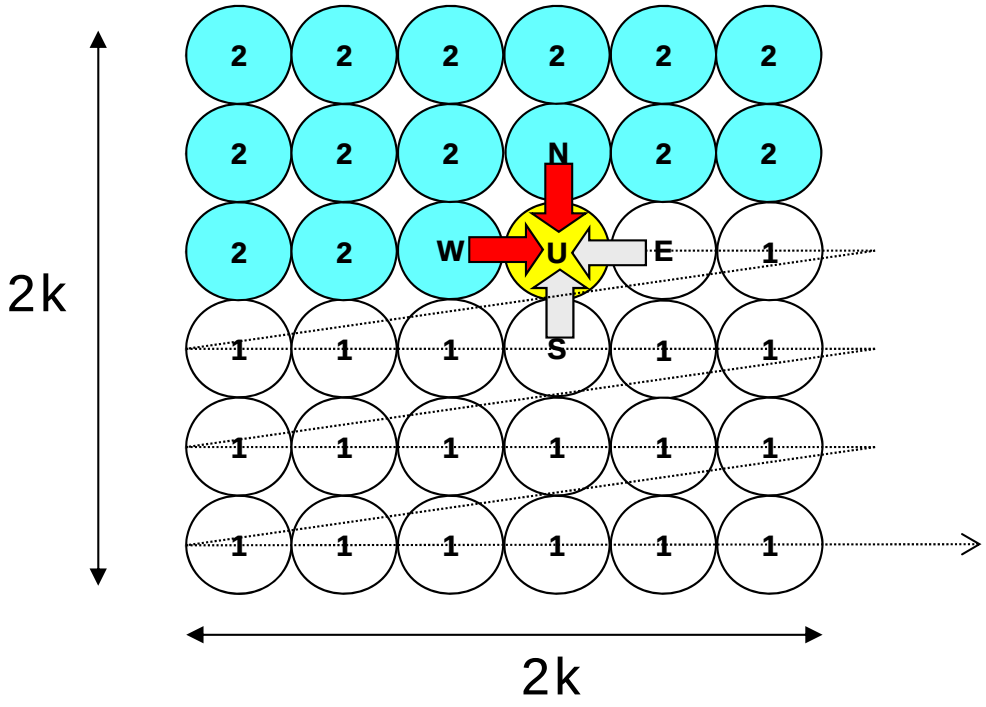
7/8 iterations: 19 cycles
1/8 iteration: 19 cycles + ”2 L2 hits” + ”1 L2 miss”= 239 cycles
CPI = (7/8*19 + 1/8 * 238) cycles / 19 instructions = **2,45**

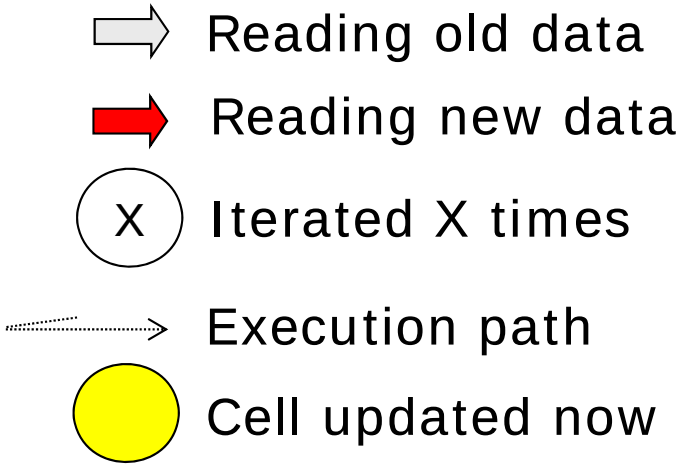
Bandwidth

1 L2 miss every 8*19 instr = 152 instr.
WB ratio = 100%
Cycles per DRAM access = 152 * CPI / (1 + WB ratio) = 186
DRAM accesses/s = 3G / 186 = 16M accesses /s ➔ BW = **1 GB/s**

GS natural. With L2 HW prefetcher

How many cycles per instruction?
What DRAM bandwidth?






```
for i = 0 to 2k
  for j = 0 to 2k {
    tmp = a[i,j];
    a[i,j] = a[i,j]+a[i-1,j]+a[i,j-1]+a [i+1,j]+a[i,j+1];
    convergence += abs(tmp-a[i,j]);
  }
```

Steps:

instr / loop
L1/L2 miss/loop
Avg cycles/loop
➔ CPI
Cycles/miss
DRAM accesses/s
➔ BW

UPPSALA UNIVERSITET

GS natural

Cache misses:

S: L2 miss 1/8, L1 hit 7/8
E: L2 hit 1/8, L1 hit 7/8
N: L2 hit 1/8, L1 hit 7/8
U: L1 hit
W: L1 hit

7/8 iterations: 19 cycles
1/8 iteration: 19 cycles + ”2 L2 hits” + ”1 L2 miss”= 239 cycles
CPI = (7/8*19 + 1/8 * 238) cycles / 19 instructions = **2,45**

Bandwidth

1 L2 miss every 8*19 instr = 152 instr.
WB ratio = 100%
Cycles per DRAM access = 152 * CPI / (1 + WB ratio) = 186
DRAM accesses/s = 3G / 186 = 16M accesses /s ➔ BW = **1GB/s**

UPPSALA UNIVERSITET

GS nat, L2 pref

Cache misses:

S: **L2 hit,** L1 hit 7/8
E: L2 hit 1/8, L1 hit 7/8
N: L2 hit 1/8, L1 hit 7/8
U: L1 hit
W: L1 hit

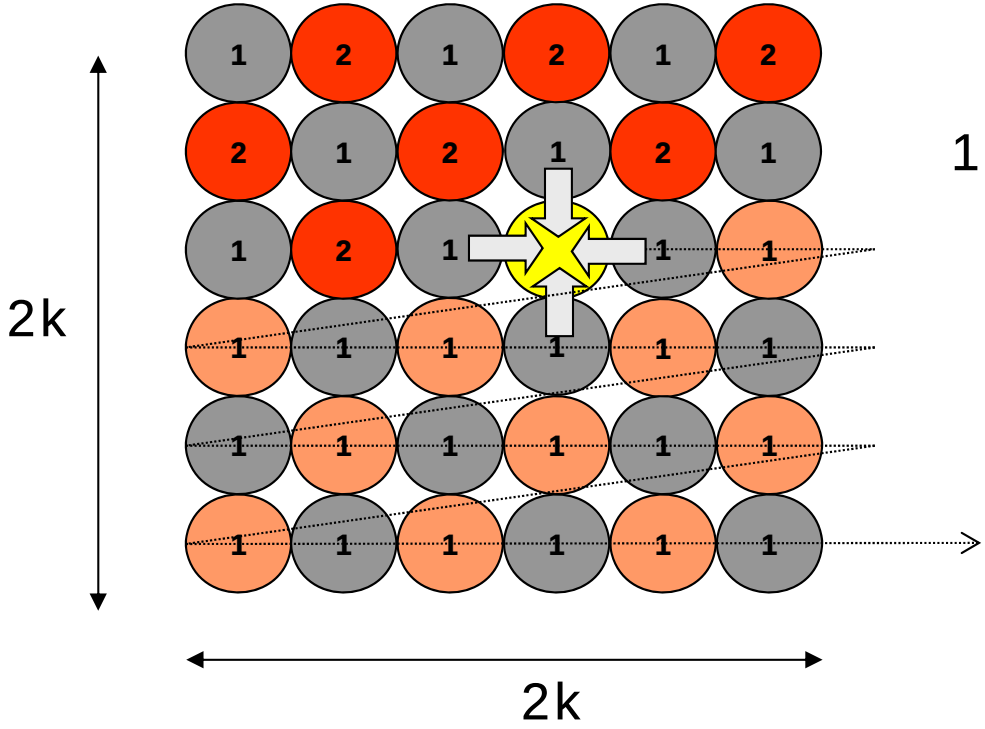
7/8 iterations: 19 cycles
1/8 iteration: 19 cycles + "**3** L2 hits" + "**0** L2 miss"= **49** cycles
CPI = (7/8*19 + 1/8 * **49**) cycles / 19 instructions = **1,20**

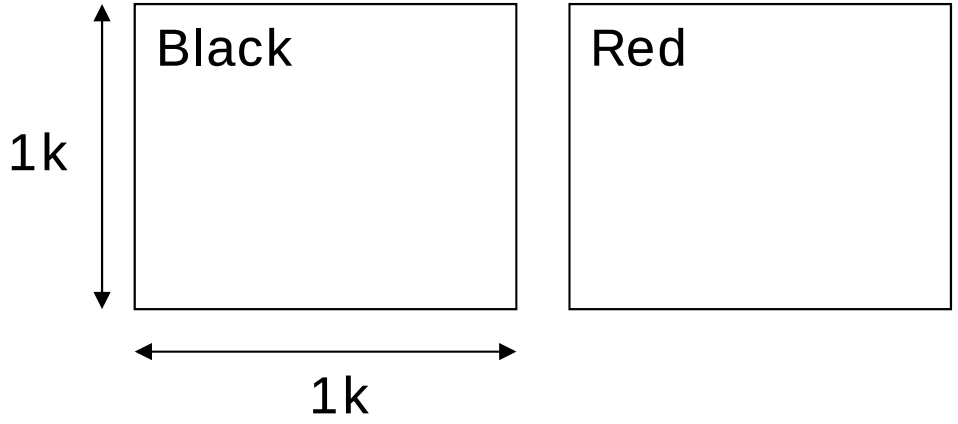
Bandwidth

1 **DRAM read** every 8*19 instr = 152 instr.
WB ratio = 100%
Cycles per DRAM access = 152 * CPI / (1 + WB ratio) = **91**
DRAM accesses/s = 3G / **91** = **32**M accesses /s ➔ BW = **2,1 GB/s**

Removing Dependencies: Red/Black

How many cycles per instruction (CPI)?
What DRAM bandwidth?





LOOP:
  UPDATE ALL **RED** POINTS
  UPDATE ALL **BLACK** POINTS
  IF (convergence_test)
    <done>

Cache misses (GS natural)

S: L2 miss 1/8, L1 hit 7/8
E: L2 hit 1/8, L1 hit 7/8
N: L2 hit 1/8, L1 hit 7/8
U: L1 hit
W: L1 hit

WB ratio = 100% (GS natural)

UPPSALA UNIVERSITET

GS natural

Cache misses:

S: L2 miss 1/8, L1 hit 7/8
E: L2 hit 1/8, L1 hit 7/8
N: L2 hit 1/8, L1 hit 7/8
W: L1 hit
U: L1 hit

7/8 iterations: 19 cycles
1/8 iteration: 19 cycles + ”2 L2 hits” + ”1 L2 miss”= 239 cycles
CPI = (7/8*19 + 1/8 * 238) cycles / 19 instructions = **2,45**

Bandwidth

1 L2 miss every 8*19 instr = 152 instr.
WB ratio = 100%
Cycles per DRAM access = 152 * CPI / (1 + WB ratio) = 186
DRAM accesses/s = 3G / 186 = 16M accesses /s ➔ BW = **1GB/s**

UPPSALA UNIVERSITET

Red / black

Cache misses:

S: L2 miss 1/8, L1 hit 7/8
E: L2 hit 1/8, L1 hit 7/8
N: L2 hit 1/8, L1 hit 7/8
W: L1 hit
U: L2 miss 1/8, L1 hit 7/8

7/8 iterations: 19 cycles
1/8 iteration: 19 cycles + ”2 L2 hits” + ”**2** L2 miss”= **439** cycles
CPI = (7/8*19 + 1/8 * 238) cycles / 19 instructions = **3,76**

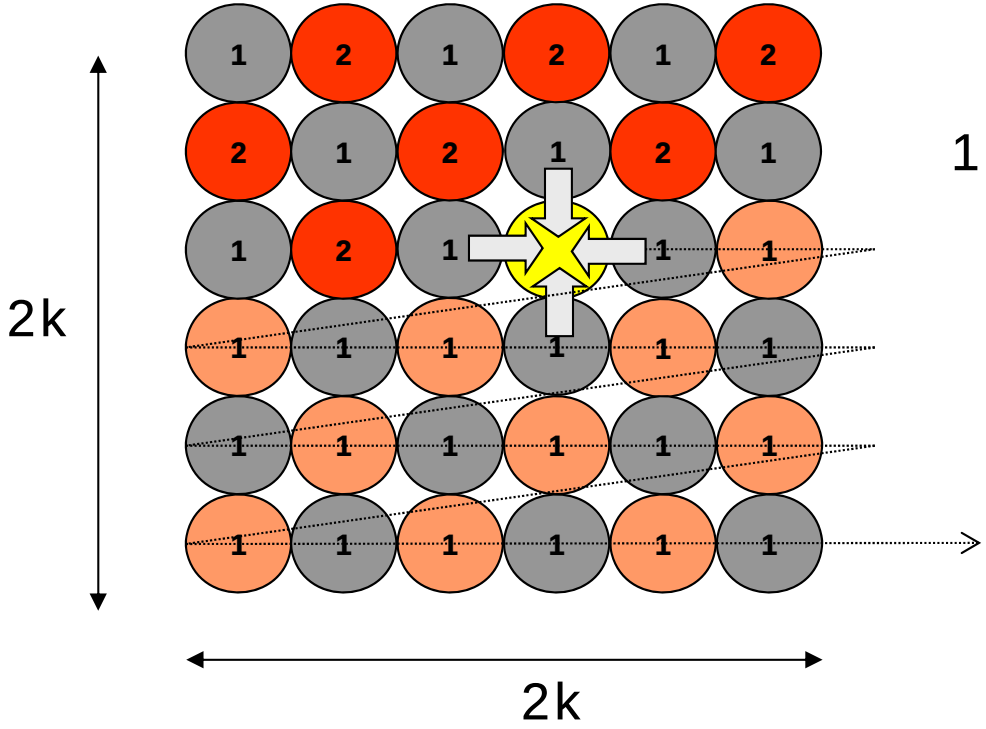
Bandwidth

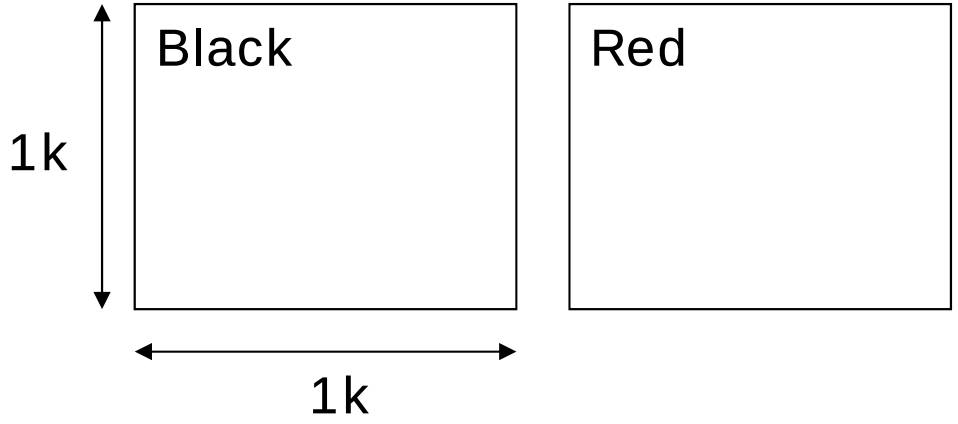
2 L2 miss every 8*19 instr => **76** instr/miss
WB ratio = **50%**
Cycles per DRAM access = **76** * CPI / (1 + WB ratio) = **190**
DRAM accesses/s = 3GHz / 186 = **15,7**M accesses /s ➔ BW = **1 GB/s**

AVDARK 2013

Red/Black w HWP?

How many cycles per instruction (CPI)?
What DRAM bandwidth?





LOOP:
  UPDATE ALL **RED** POINTS
  UPDATE ALL **BLACK** POINTS
  IF (convergence_test)
    <done>

UPPSALA UNIVERSITET

GS nat, L2 pref

Cache misses:

S: **L2 hit,** L1 hit 7/8
E: L2 hit 1/8, L1 hit 7/8
N: L2 hit 1/8, L1 hit 7/8
U: L1 hit
W: L1 hit

7/8 iterations: 19 cycles
1/8 iteration: 19 cycles + "**3** L2 hits" + "**0** L2 miss"= **49** cycles
CPI = (7/8*19 + 1/8 * **49**) cycles / 19 instructions = **1,20**

Bandwidth

1 **DRAM read** every 8*19 instr = 152 instr.
WB ratio = 100%
Cycles per DRAM access = 152 * CPI / (1 + WB ratio) = **91**
DRAM accesses/s = 3G / **91** = **32**M accesses /s ➔ BW = **2,1GB/s**

UPPSALA UNIVERSITET

Red/Black, L2 pref

Cache misses:

S: **L2 hit,** L1 hit 7/8
E: L2 hit 1/8, L1 hit 7/8
N: L2 hit 1/8, L1 hit 7/8
U: **L2 hit**
W: L1 hit

7/8 iterations: 19 cycles
1/8 iteration: 19 cycles + "**4** L2 hits" + "**0** L2 miss"= **59** cycles
CPI = (7/8*19 + 1/8 * **59**) cycles / 19 instructions = **1,26**

Bandwidth

2 DRAM read every 8*19 instr = 152 instr.
WB ratio = **50**%
Cycles per DRAM access = 152 * CPI / (1 + WB ratio) = **64**
DRAM accesses/s = 3G / **64** = **47**M accesses /s ➔ BW = **3,0GB/s**

AVDARK 2013

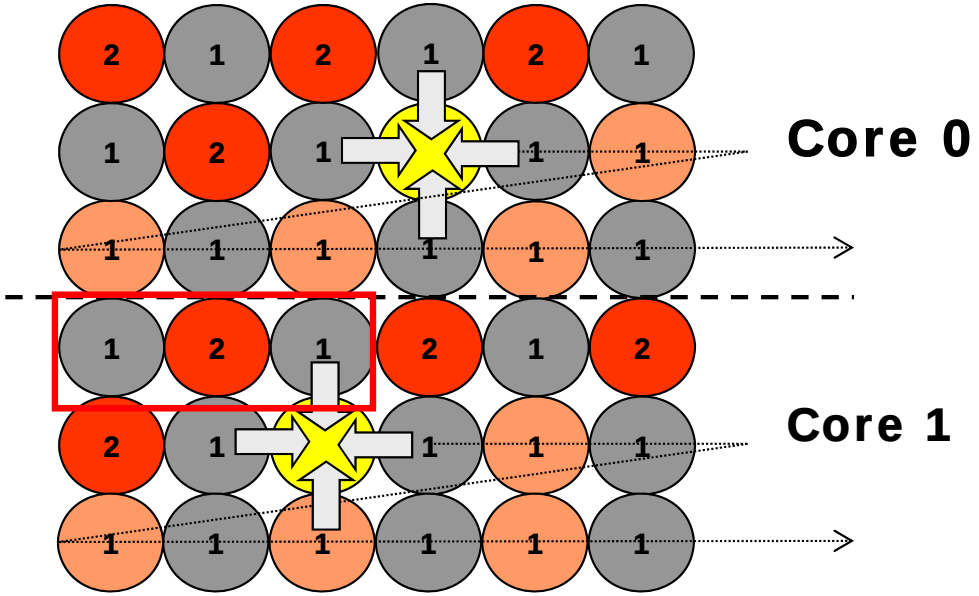
UPPSALA UNIVERSITET

	CPI	DRAM BW (GB/s)
GS natural	2,45	1,03
GS natural HWP	1,20	2,11
RB	3,76	1,01
RB HWP	1,26	3,00

UPPSALA UNIVERSITET

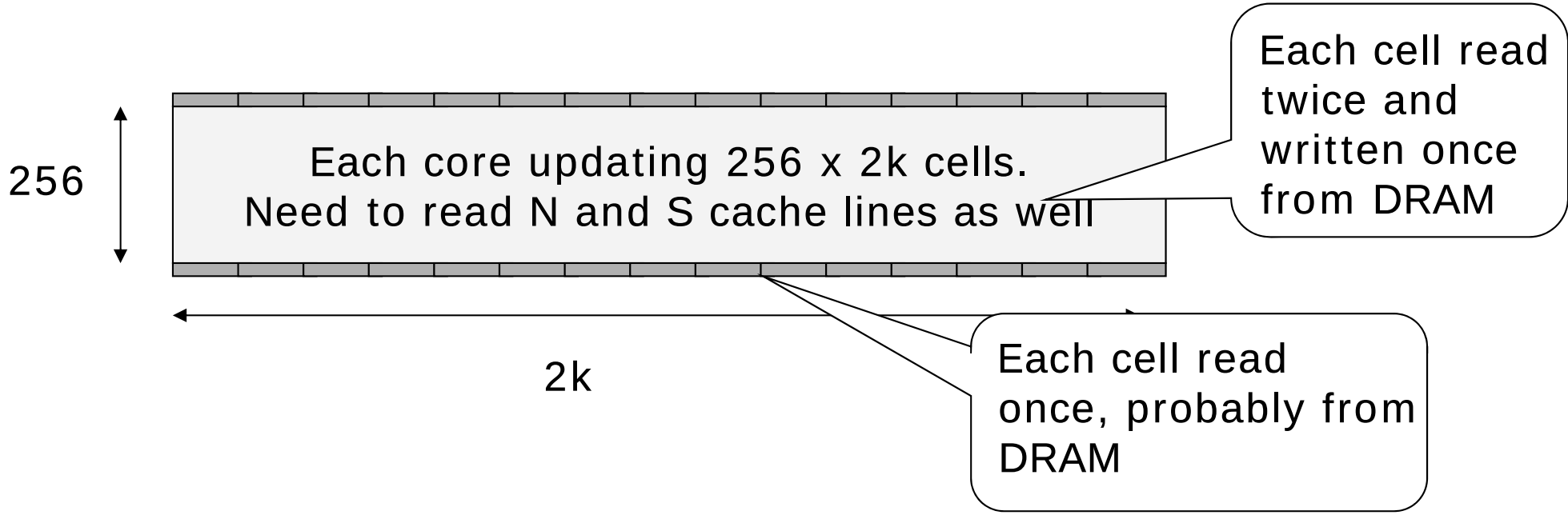
	CPI	DRAM BW (GB/s)
GS natural	2,45	1,03
GS natural, HWP	1,20	2,11
RB	3,76	1,01
RB, HWP	1,26	3,00
GS block=4, HWP	1,20	0,53

Red/Black, 8 cores (with HWP) (here only showing 2)



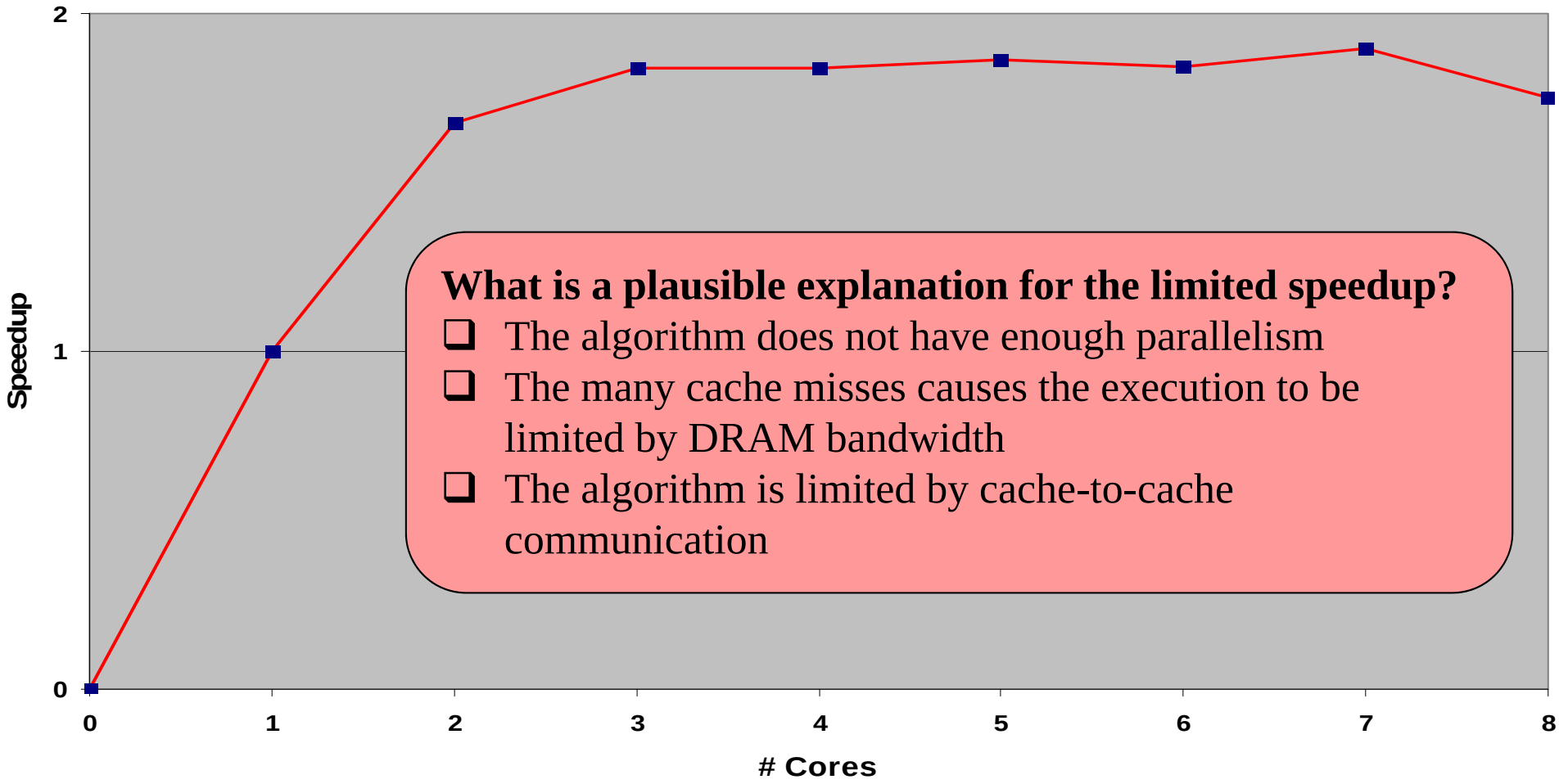
"1/8" the execution time
How many DRAM accesses?
How many 2

LOOP:
  IN PARALLEL: UPDATE ALL **RED** POINTS
  <barrier>
  IN PARALELL: UPDATE ALL **BLACK** POINTS
  <barrier>
  IF (convergence_test)
    <done>



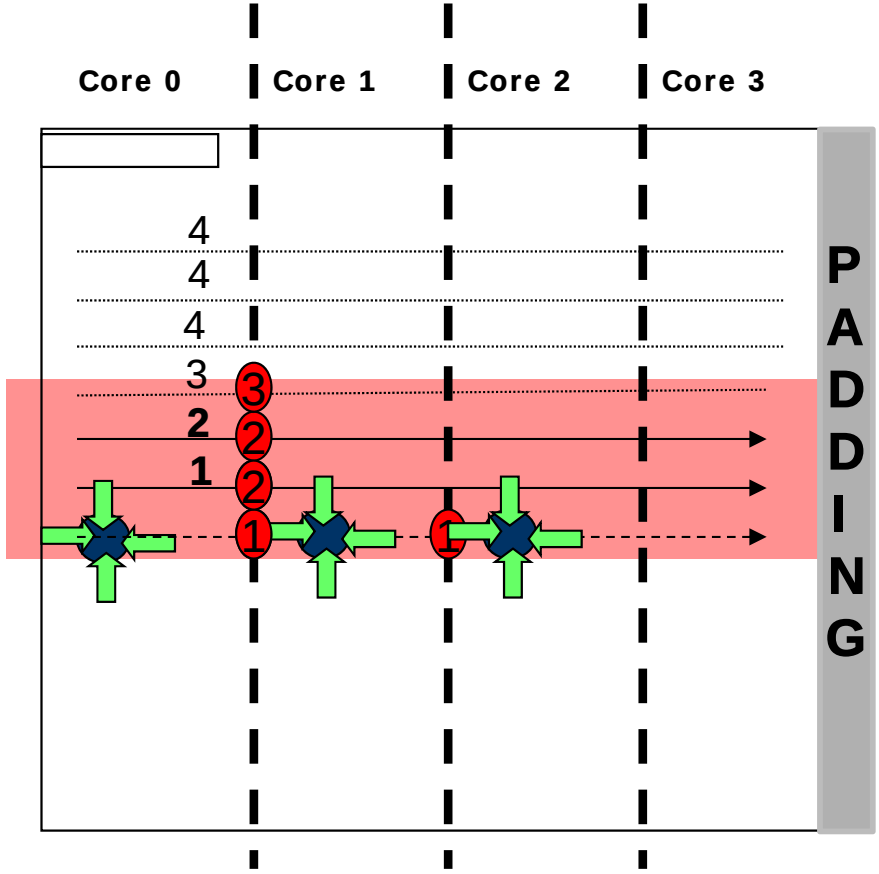
Previous DRAM accesses: ~256 * 3
New DRAM accesses: ~2 * 1
Increas in DRAM accesses: 0,3%

Only One Problem: Performance



Read/Black can double the number of cache misses
Sweeps the array twice per iteration

G-S, Block=4, HWP, 8 cores



= sweep path

= current

= data dependence

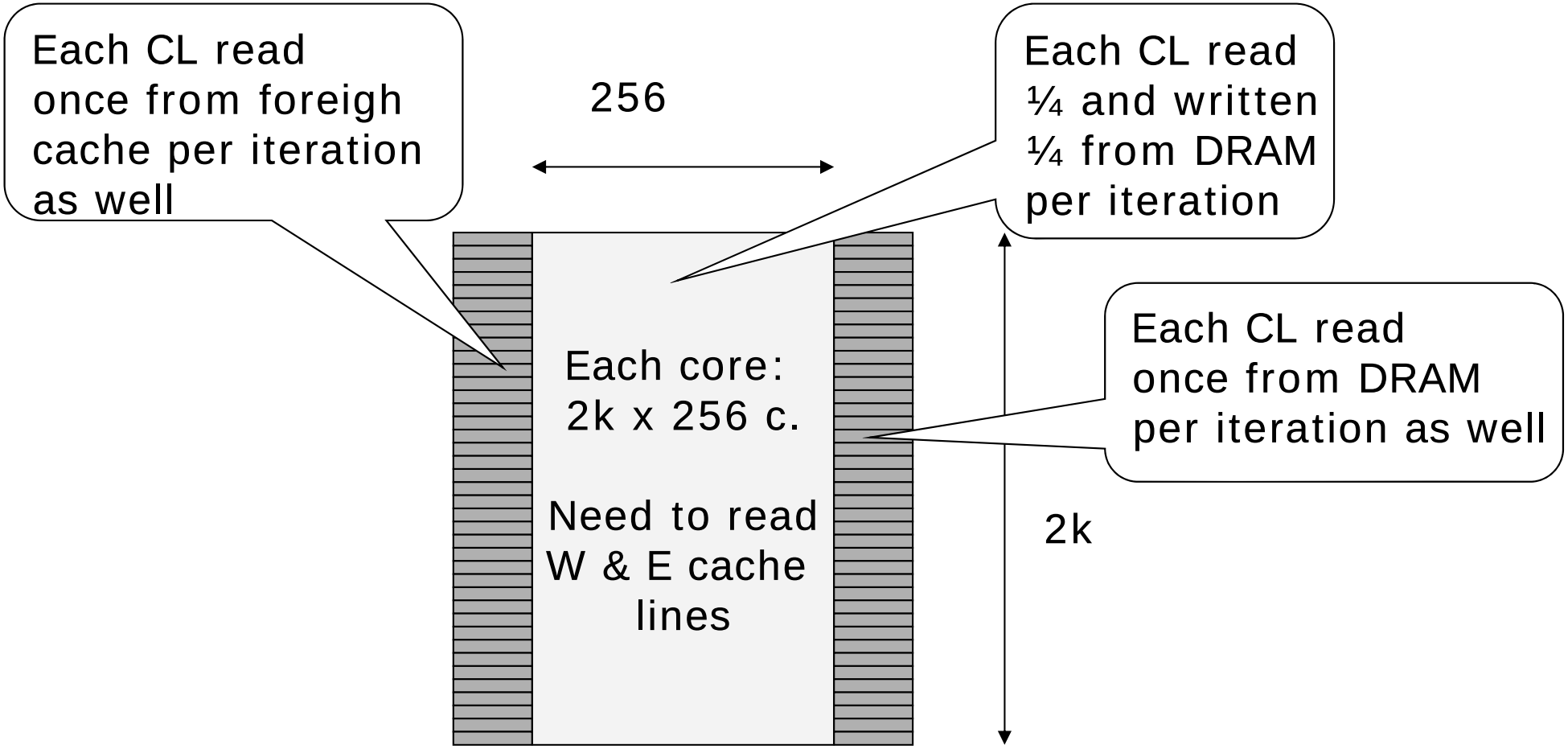
1,2,3,4 = iteration number

= cacheline layout

= active region

1 = sync flag iteration no

”1/8” the execution time
How much DRAM access increase?
How many 2



Previous DRAM accesses: ~32 CL * ¼
E: New DRAM accesses: ~1 CL * 1
W: New cache2cache: ~1 CL *1
W: Read flag ~1 CL *1
Increas in DRAM + 2 accesses: 37%

UPPSALA UNIVERSITET

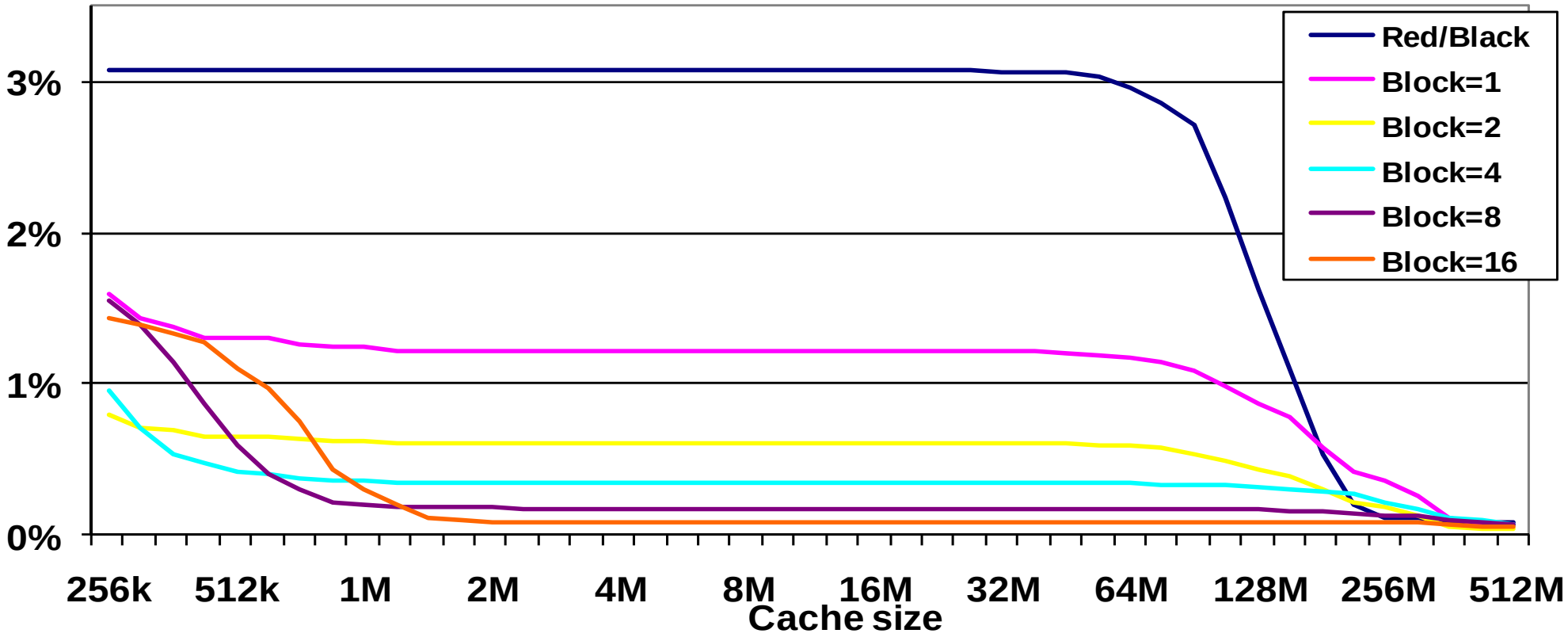
	CPI	DRAM + 2 BW (GB/s)
GS natural	2,45	1,03
GS natural, HWP	1,20	2,11
RB	3,76	1,01
RB, HWP	1,26	3,00
GS block=4, HWP	1,20	0,53
8x RB, HWP	**0,16**	**3,0 * 8 = 24**
8x GS block=4, HWP	**0,15**	**0,53 * 8 * 1,37 = 5,8**

DRA

Larger blocks (active region) limits DRAM traffic for large caches, but seem to increase traffic for small caches. Why?

- ❑ The cache line size increase and cause more false sharing
- ❑ The size of the active region increases and cannot fit in the cache
- ❑ Spatial locality gets worse with larger blocks

Fetch Ratio,
i.e, fraction of mem_ops generating DRAM traffic



Note: This graph only counts read misses, not write backs.