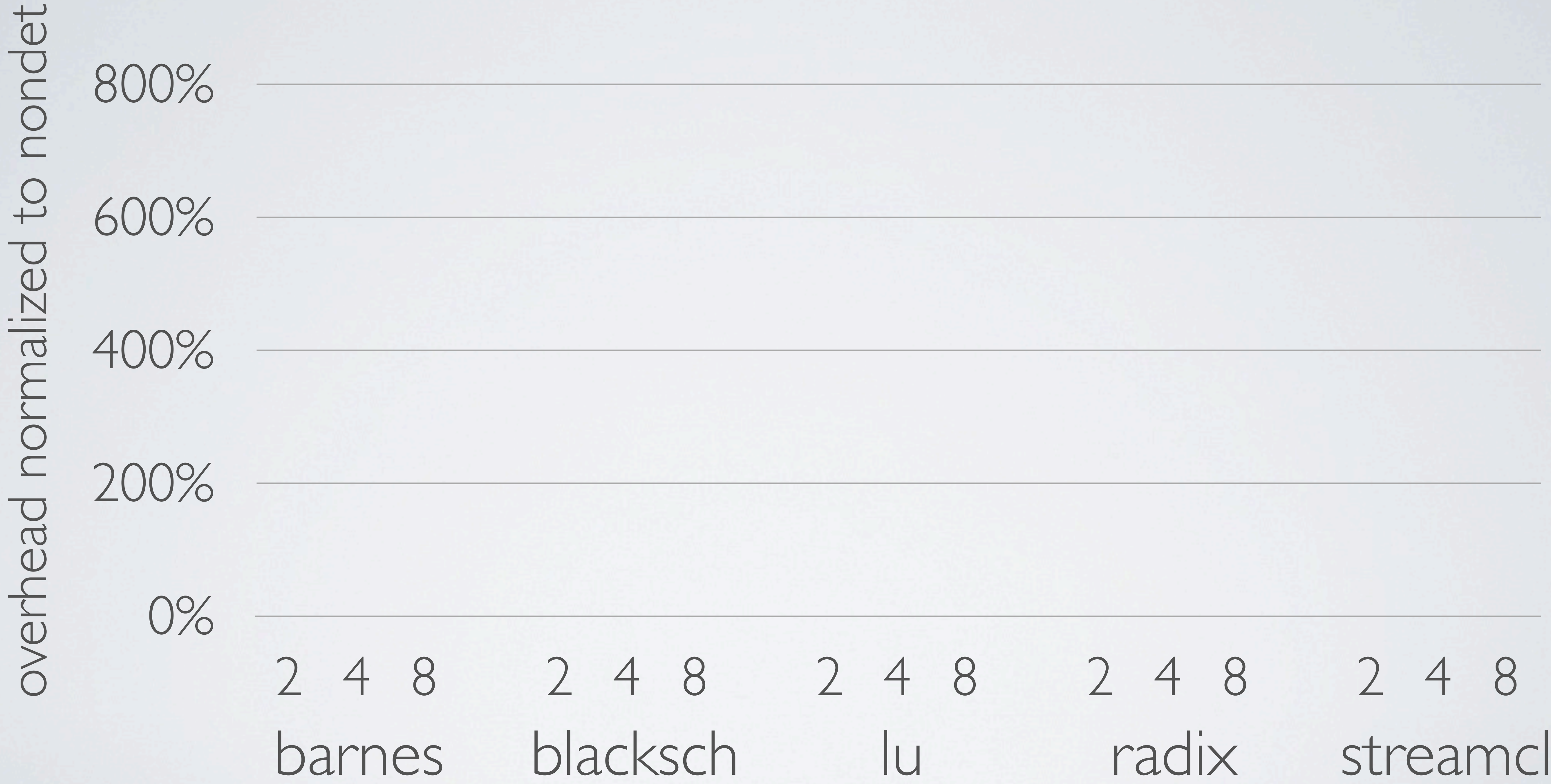
MELD:

Merging Execution- and Language-level Determinism

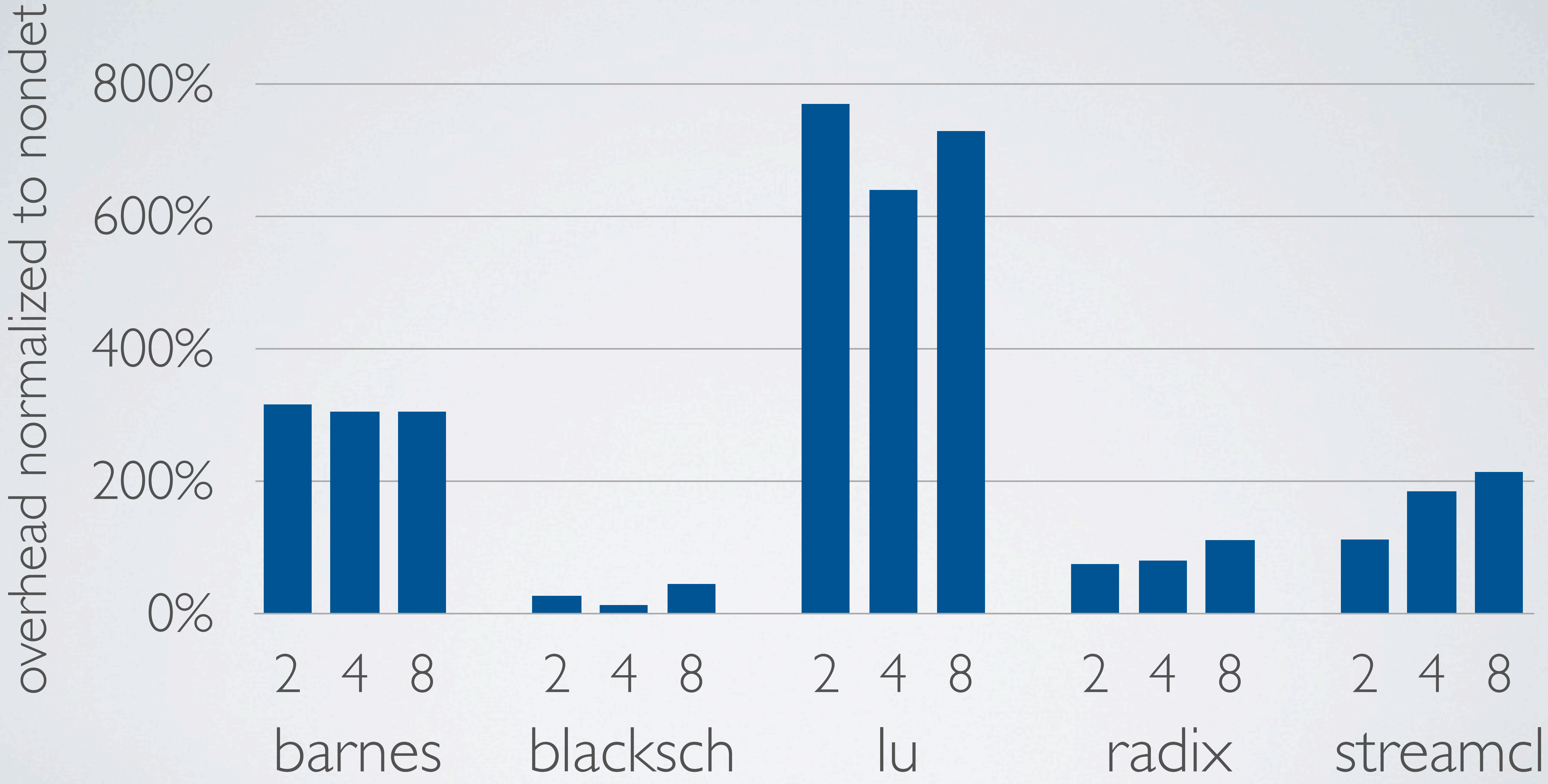
Joe Devietti, Dan Grossman, Luis Ceze
University of Washington

CoreDet results



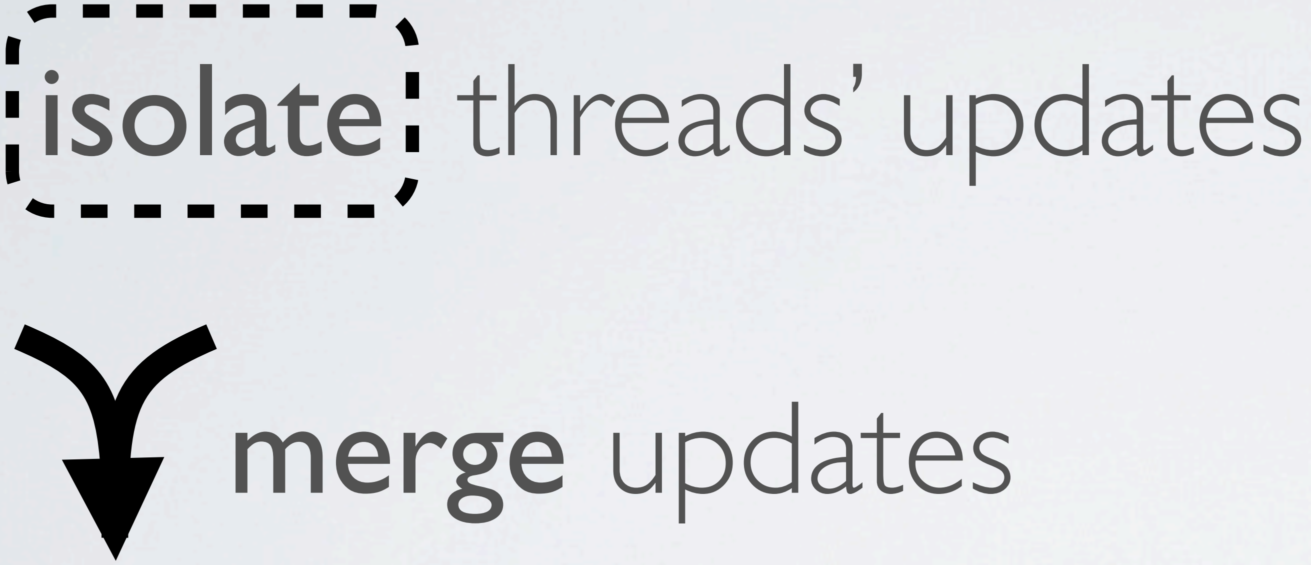
Bergan et al., ASPLOS '10;
Devietti et al. ASPLOS '11

CoreDet results



Bergan et al., ASPLOS '10;
Devietti et al. ASPLOS '11

determinism recipe



isolate threads' updates

merge updates

i. in a deterministic way

ii. at deterministic times

determinism recipe

isolate threads' updates

use store buffers to buffer updates

merge updates

i. in a deterministic way

ii. at deterministic times

determinism recipe

isolate threads' updates

use store buffers to buffer updates

merge updates

i. in a deterministic way

parallel merge algorithm

ii. at deterministic times

determinism recipe

isolate threads' updates

use store buffers to buffer updates

merge updates

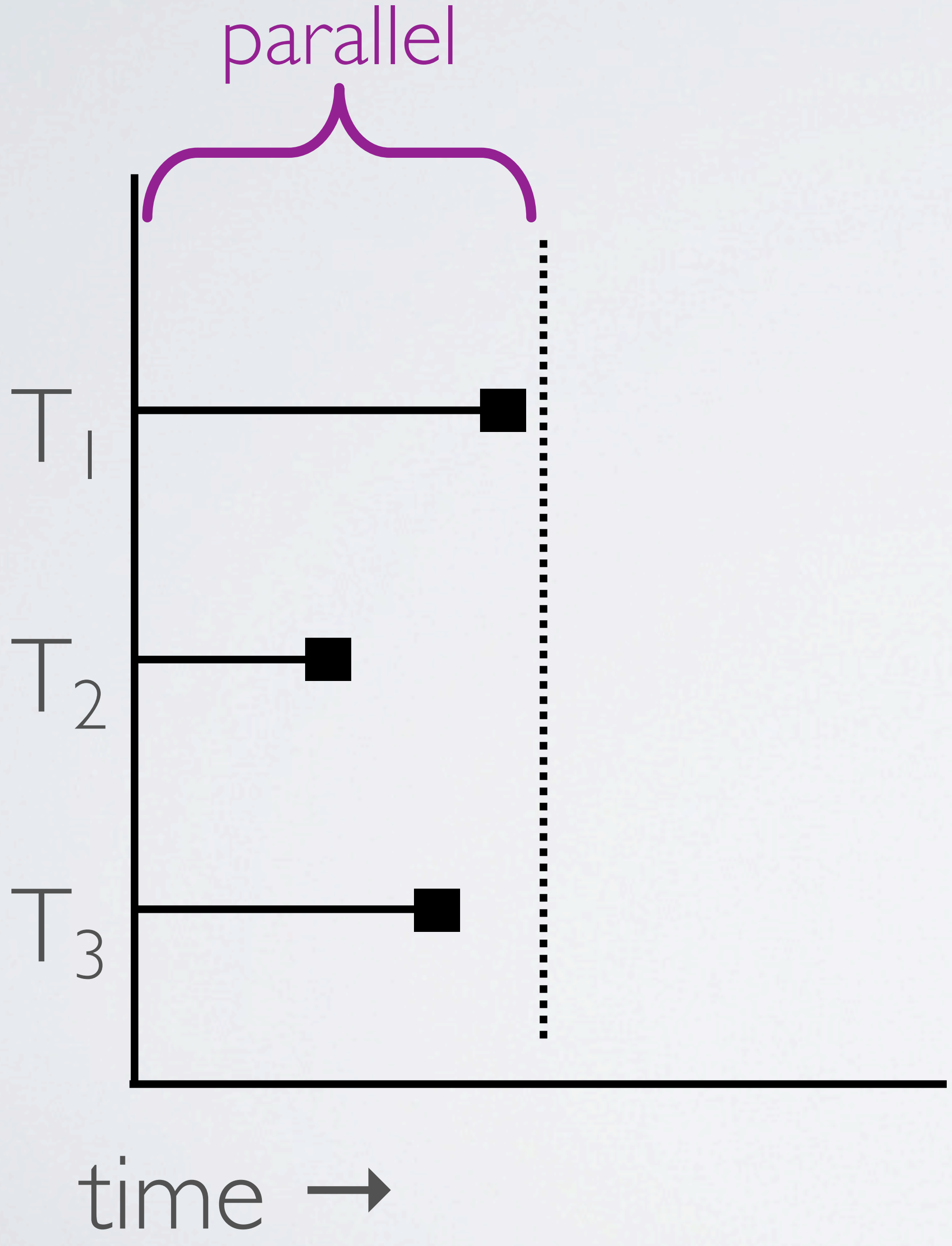
i. in a deterministic way

parallel merge algorithm

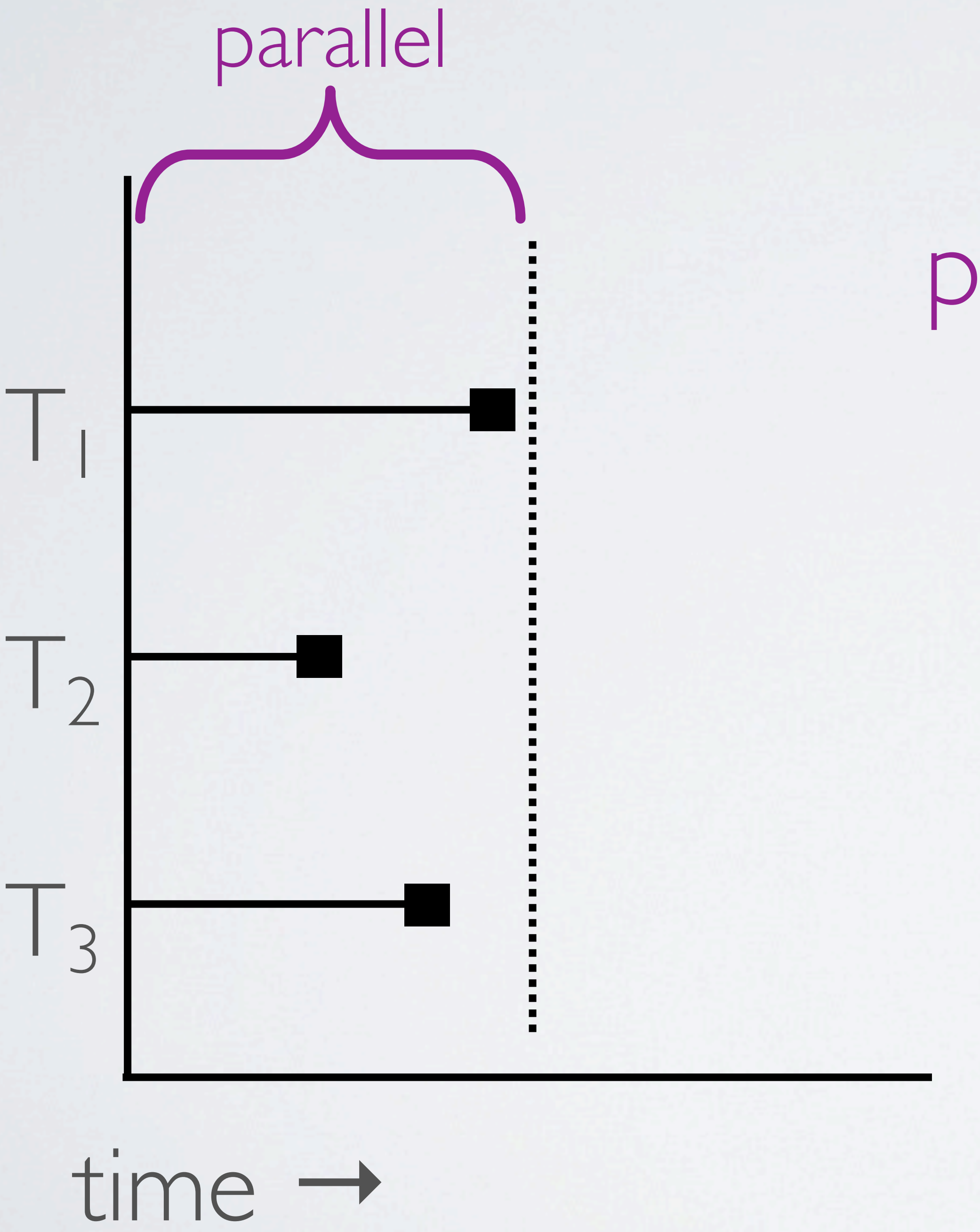
ii. at deterministic times

count fixed # of instructions

determinism via store buffers

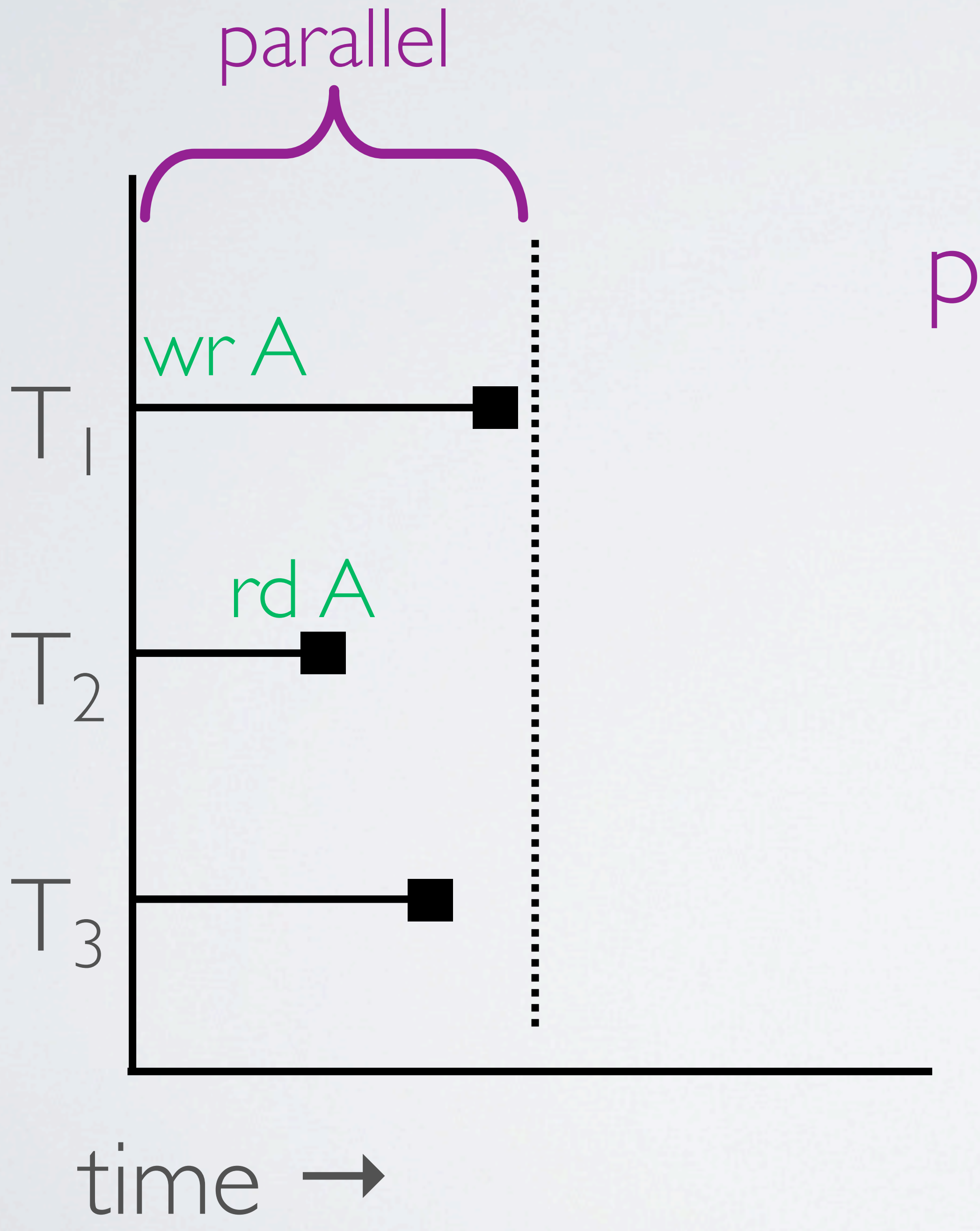


determinism via store buffers



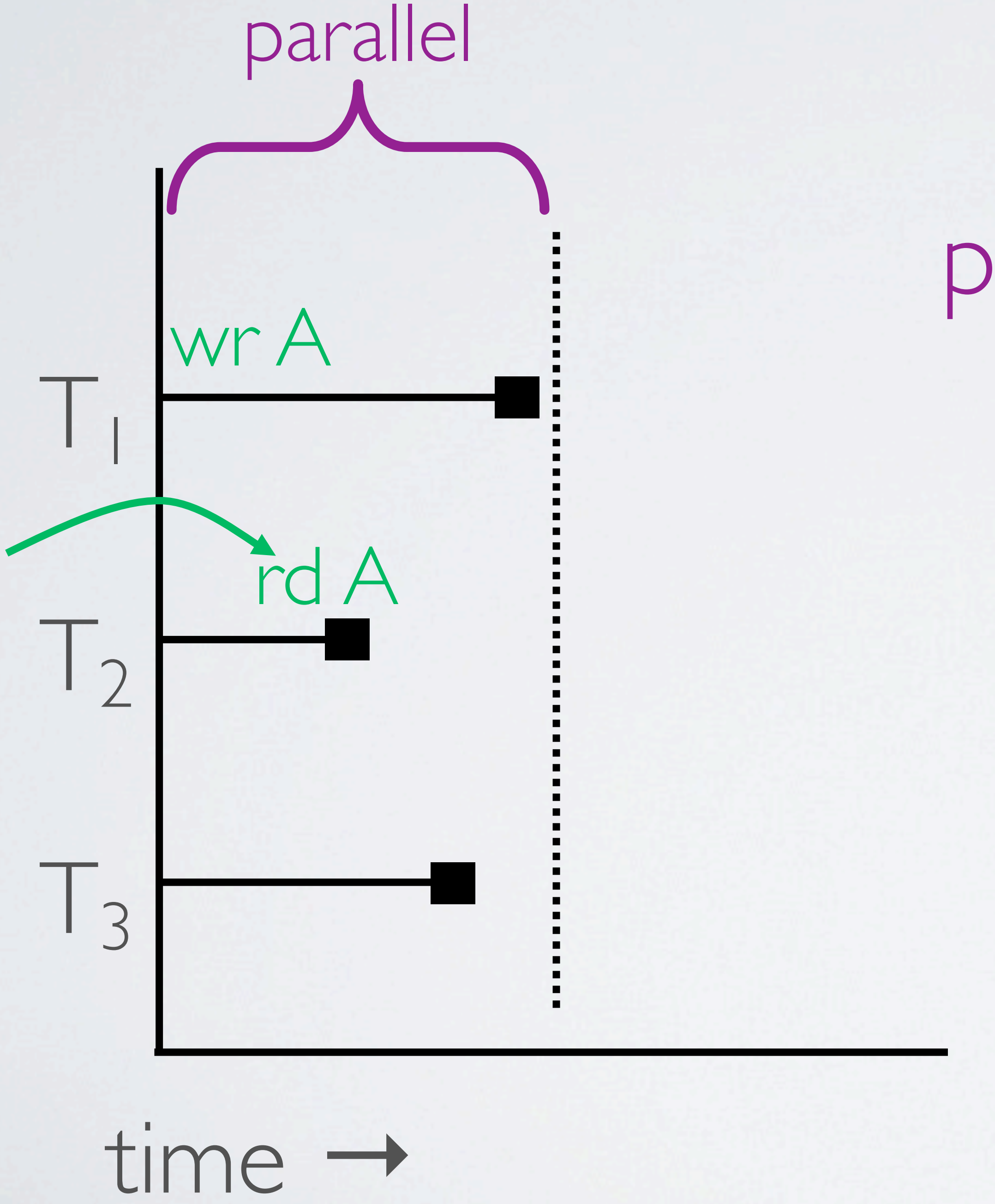
parallel mode: buffer all stores, sync via [Olszewski et al, ASPLOS '09]

determinism via store buffers



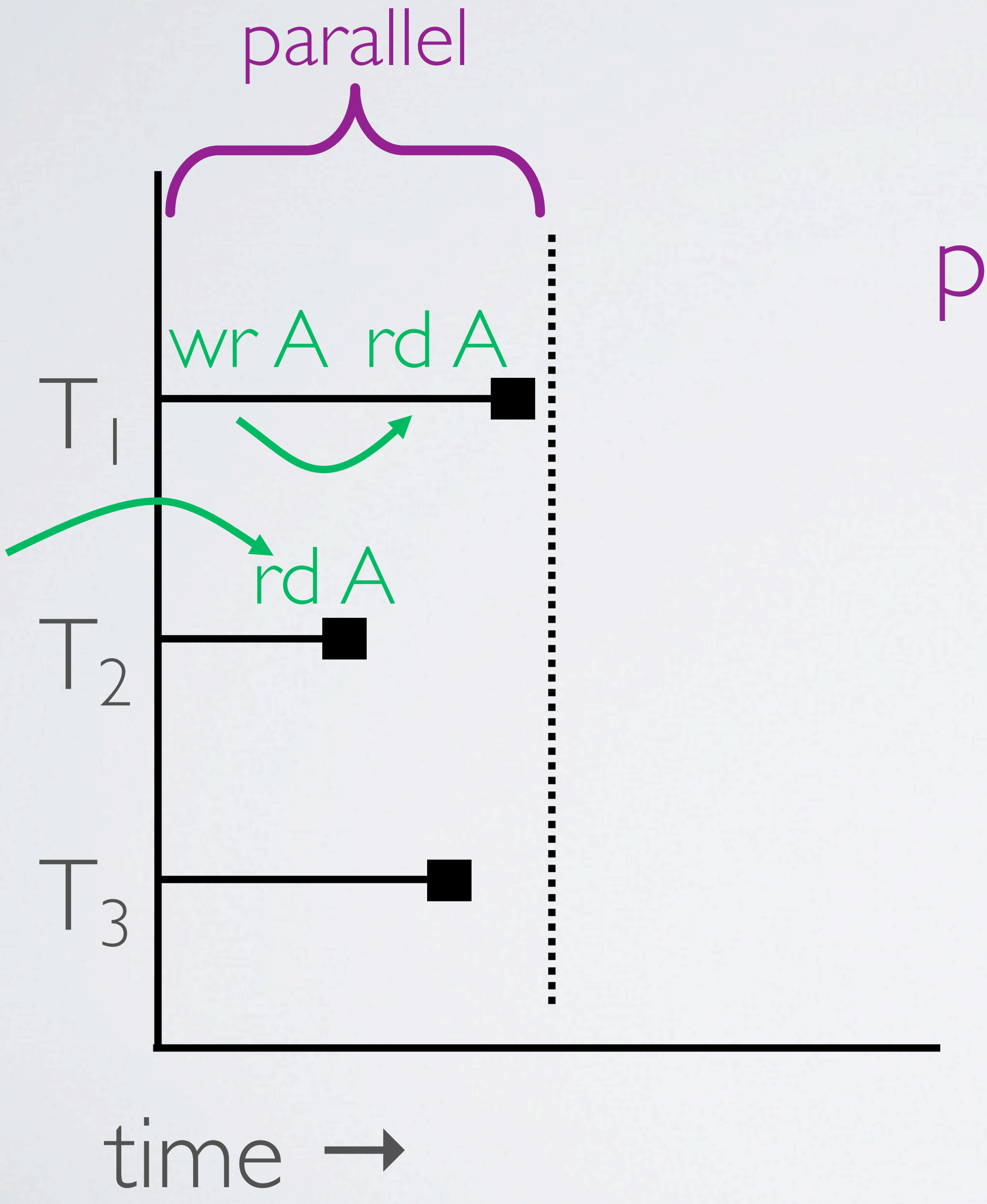
parallel mode: buffer all stores,
sync via [Olszewski et al, ASPLOS '09]

determinism via store buffers



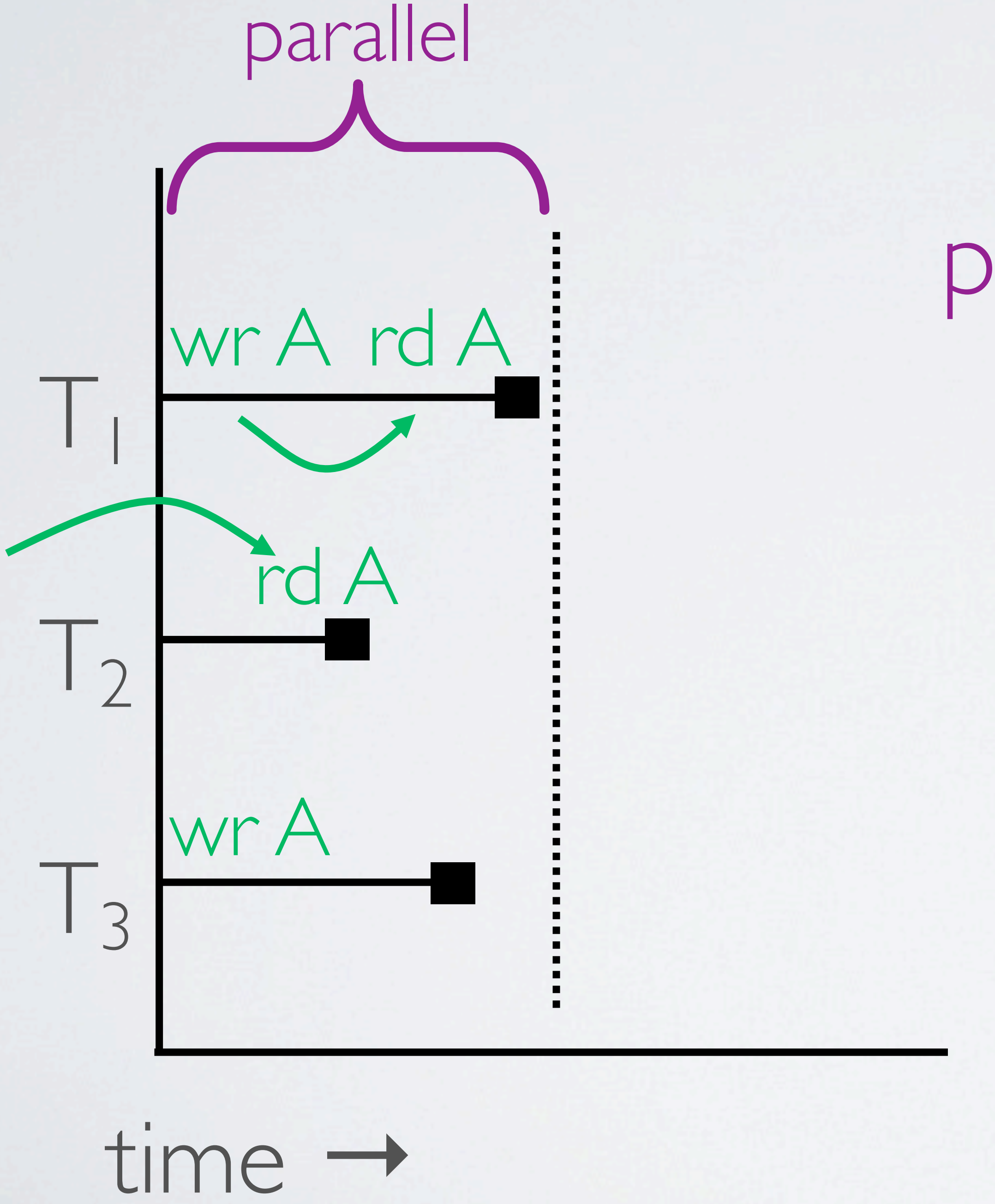
parallel mode: buffer all stores,
sync via [Olszewski et al, ASPLOS '09]

determinism via store buffers



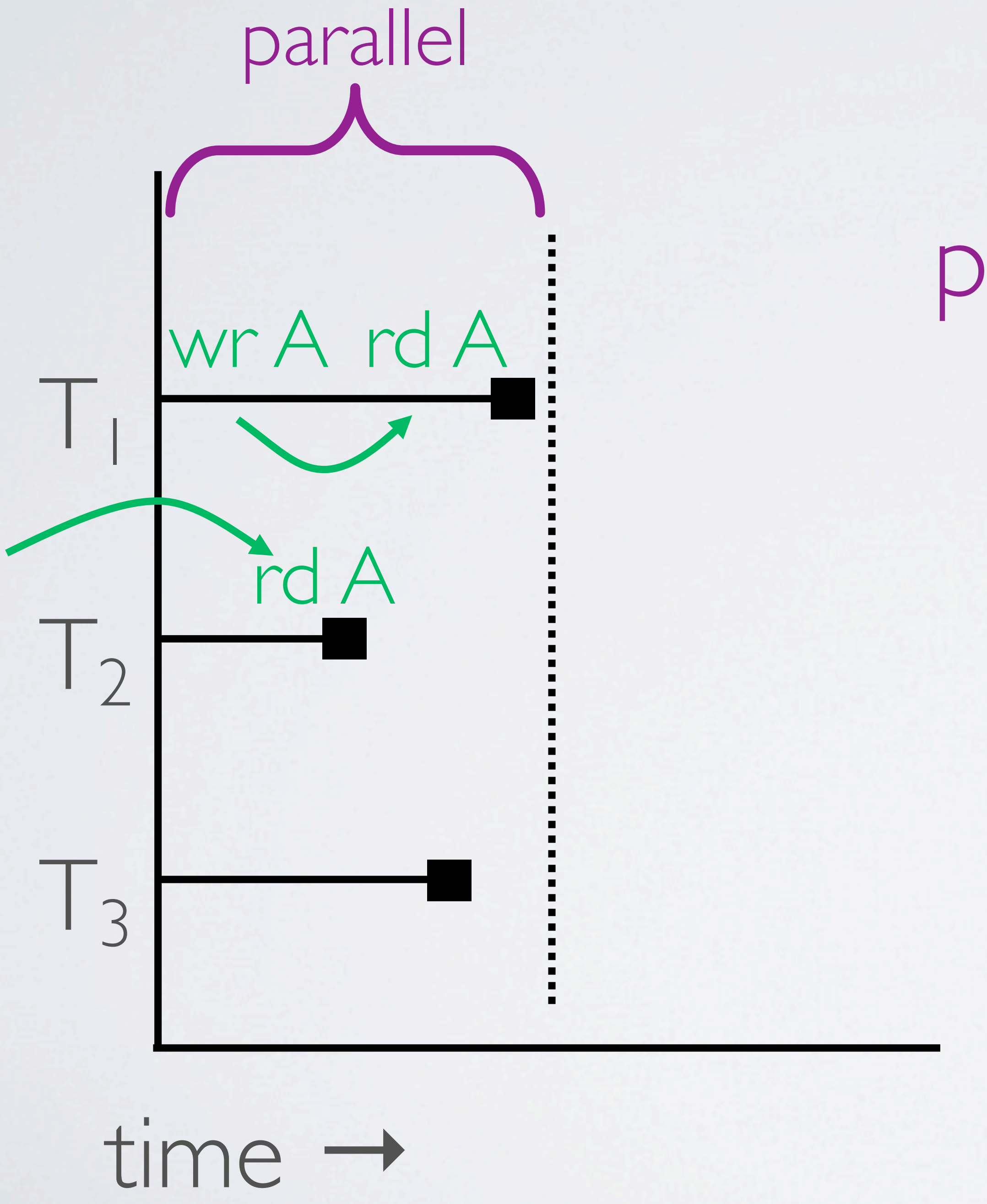
parallel mode: buffer all stores,
sync via [Olszewski et al, ASPLOS '09]

determinism via store buffers



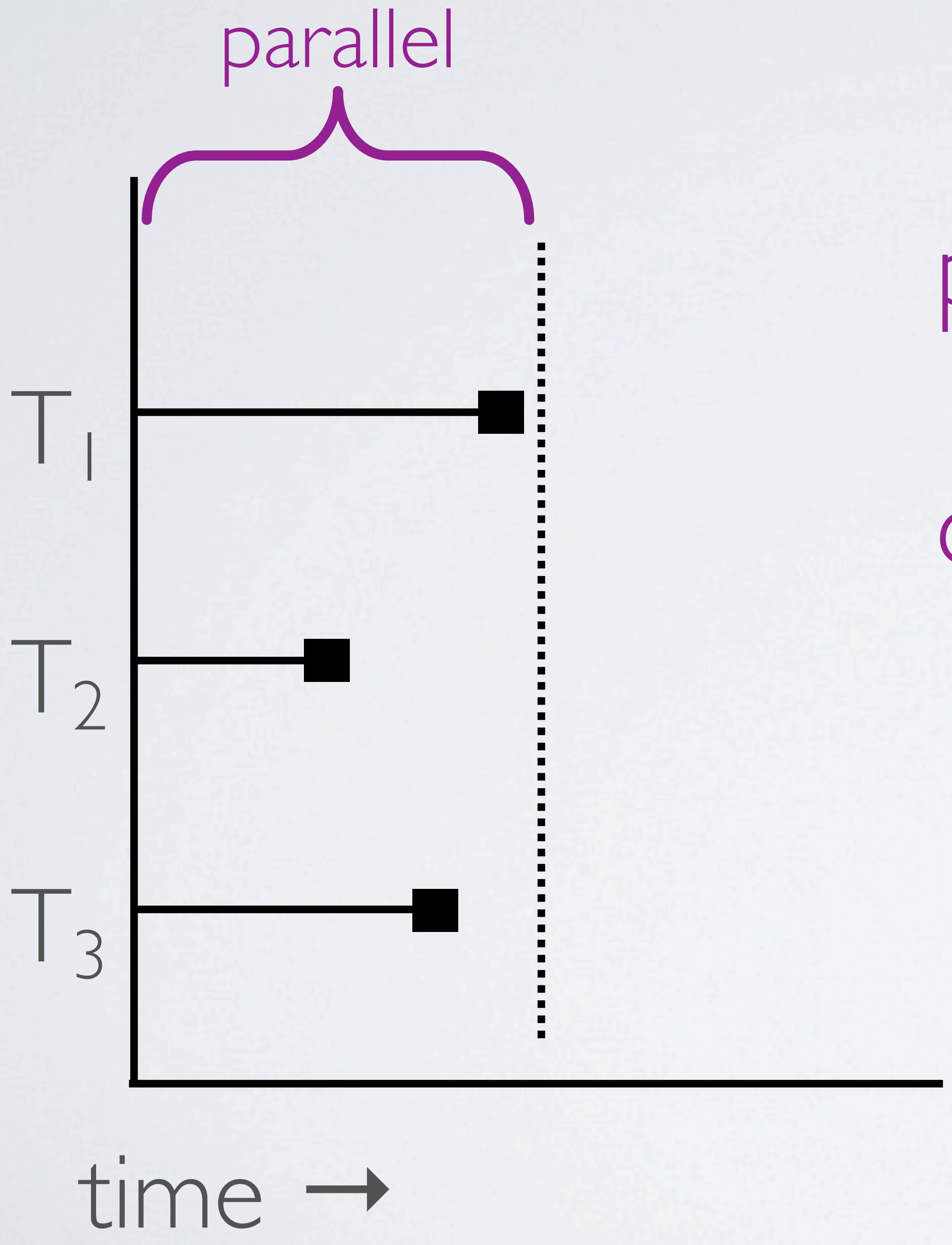
parallel mode: buffer all stores, sync via [Olszewski et al, ASPLOS '09]

determinism via store buffers



parallel mode: buffer all stores, sync via [Olszewski et al, ASPLOS '09]

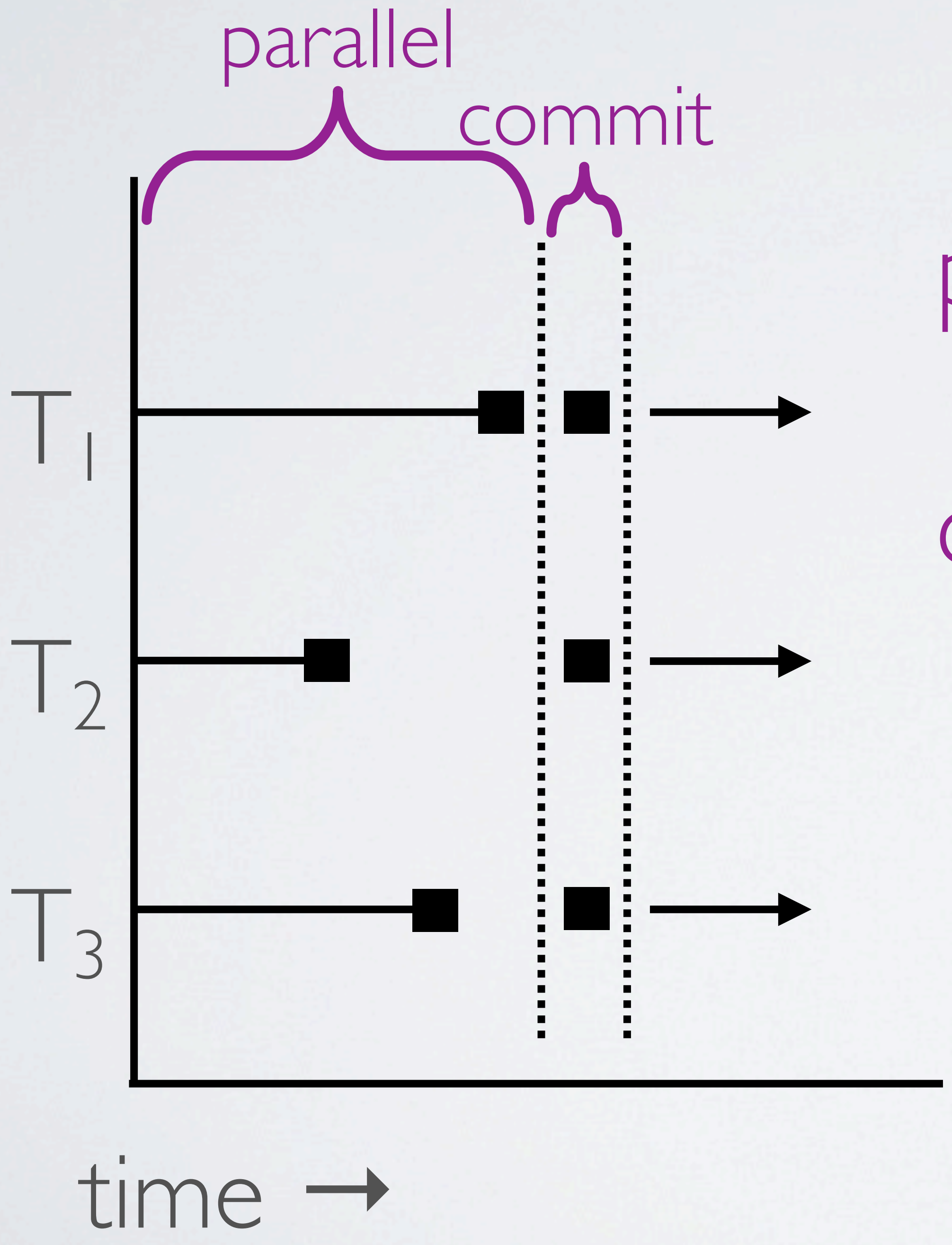
determinism via store buffers



parallel mode: buffer all stores, sync via [Olszewski et al, ASPLOS '09]

commit mode: deterministically publish buffers

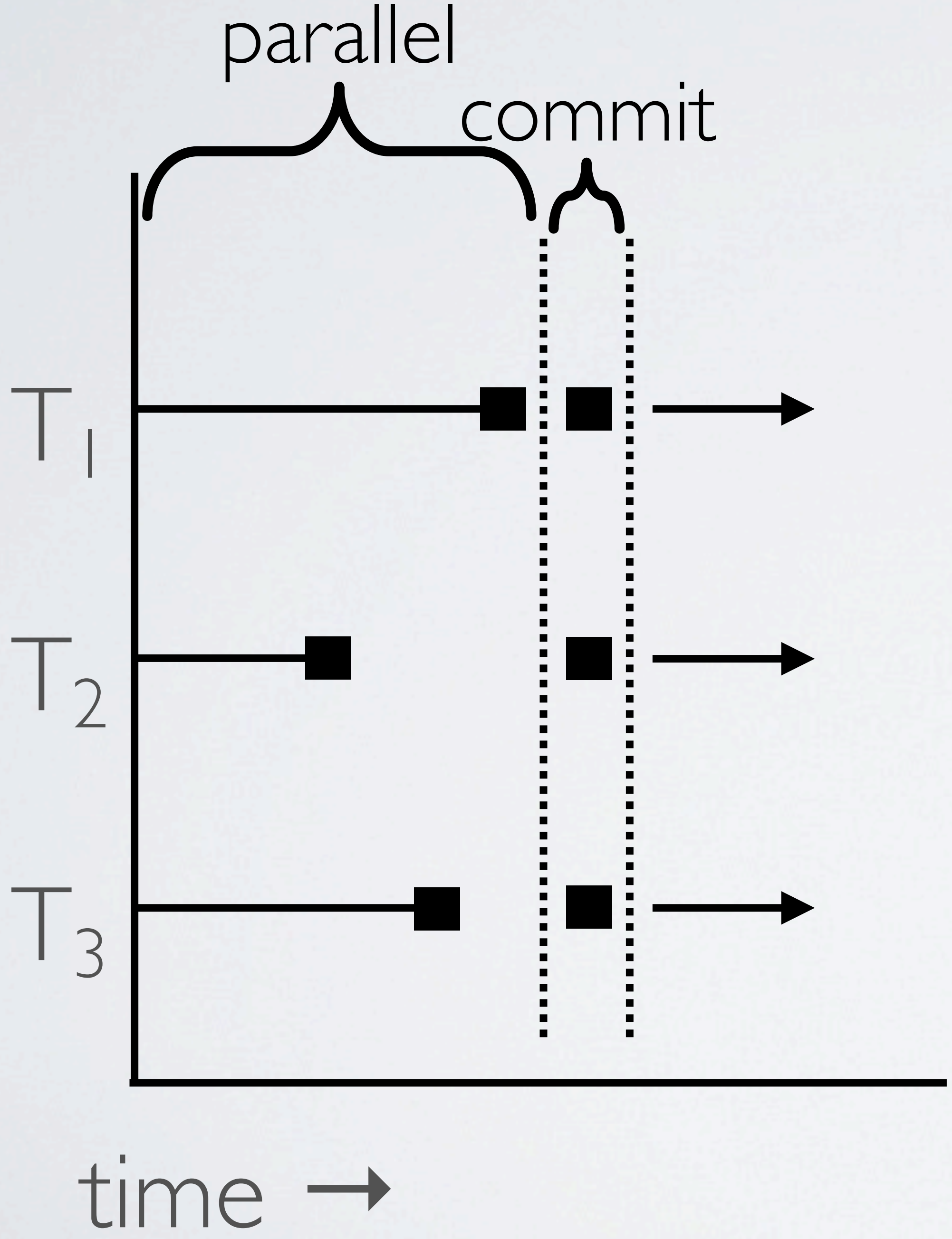
determinism via store buffers



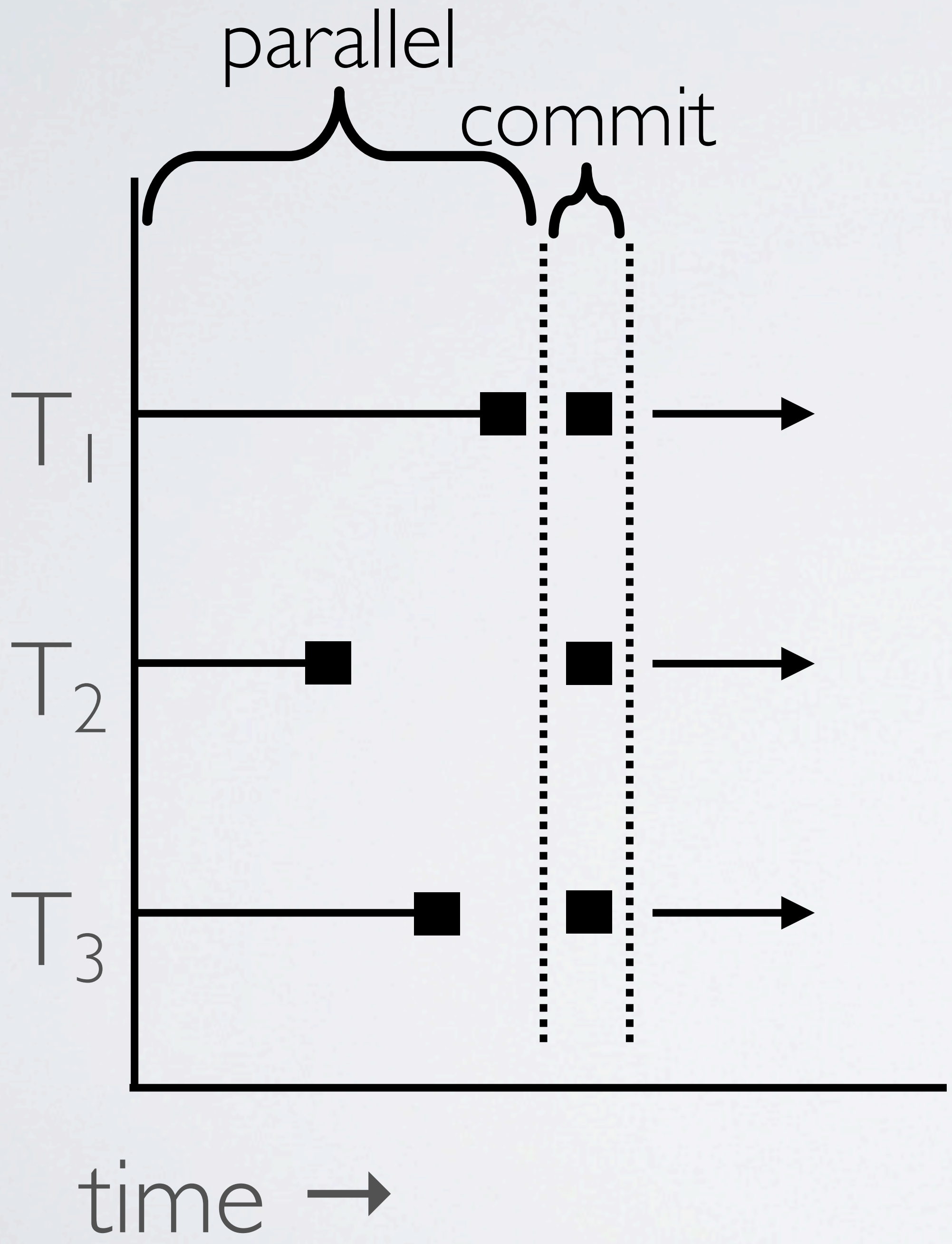
parallel mode: buffer all stores, sync via [Olszewski et al, ASPLOS '09]

commit mode: deterministically publish buffers

sources of overhead

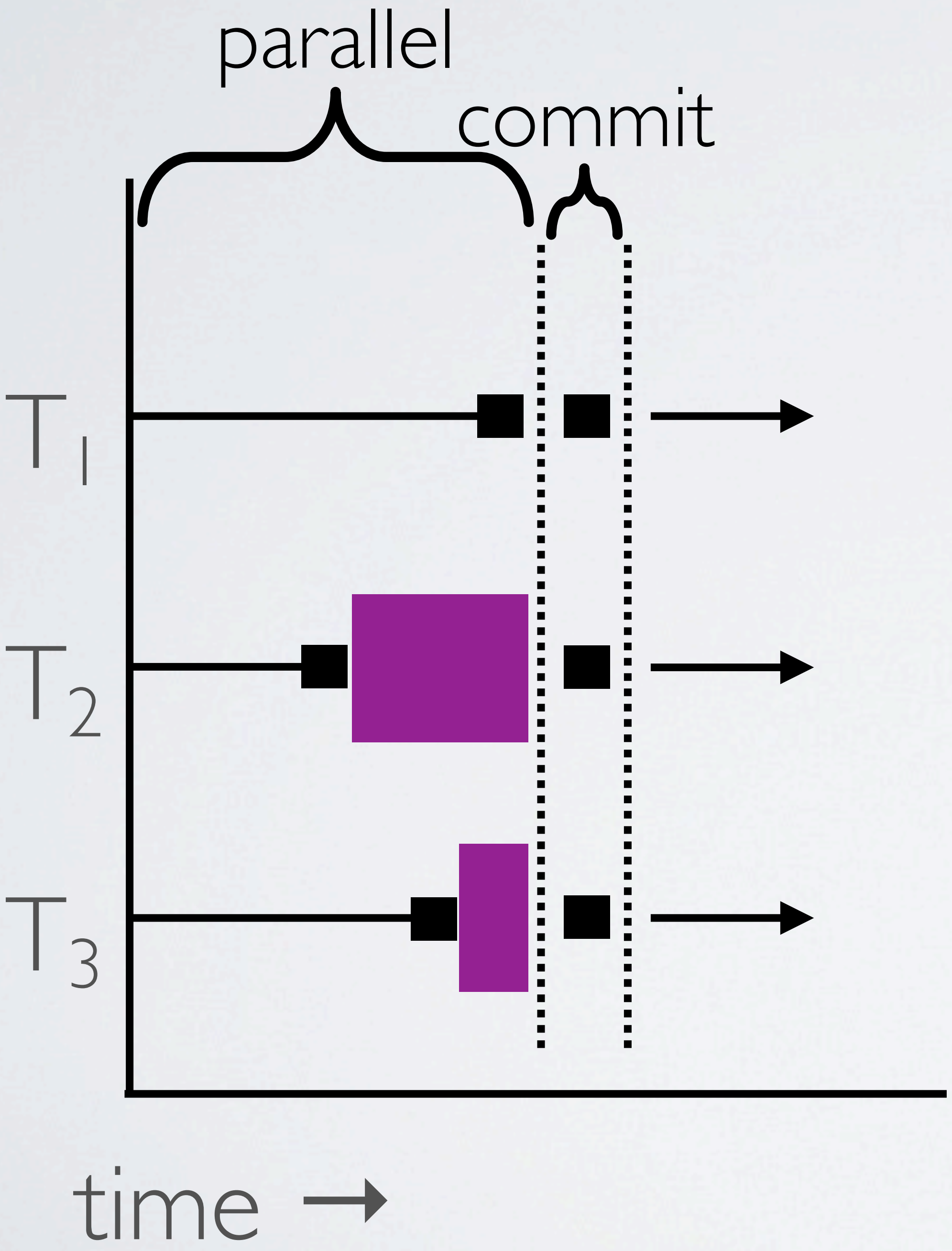


sources of overhead



store buffer instrumentation

sources of overhead

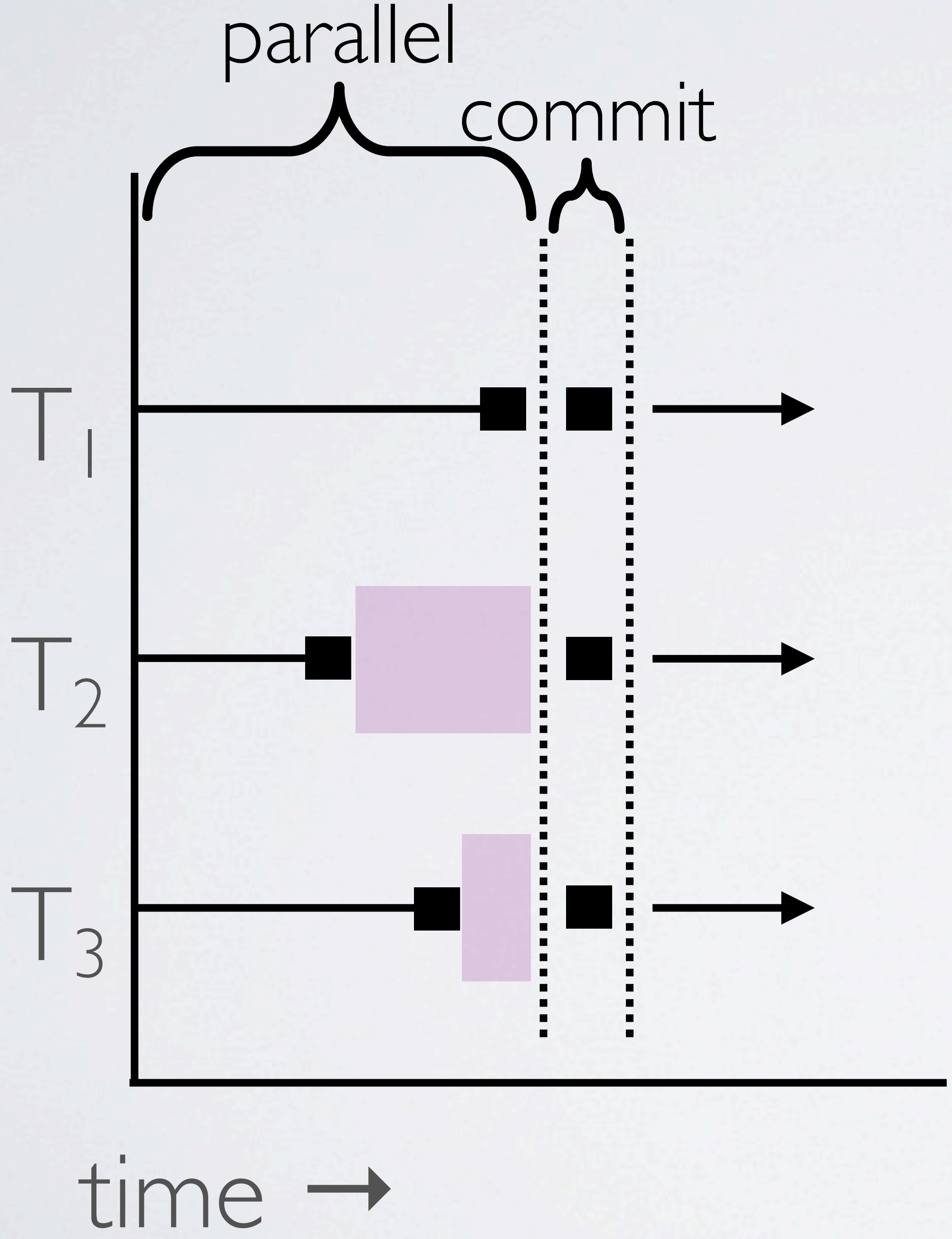


store buffer

instrumentation

imbalance

sources of overhead



store buffer

instrumentation

imbalance

MELD

Merging Execution- and Language-level Determinism

	language-level	execution-level
examples	Jade, DPJ	DMP, Kendo

MELD

Merging Execution- and Language-level Determinism

	language-level	execution-level
examples	Jade, DPJ	DMP, Kendo
runtime overhead?	none	moderate-high

MELD

Merging Execution- and Language-level Determinism

	language-level	execution-level
examples	Jade, DPJ	DMP, Kendo
runtime overhead?	none	moderate-high
supports all code?	no	yes

MELD

Merging Execution- and Language-level Determinism

	language-level	execution-level
examples	Jade, DPJ	DMP, Kendo
runtime overhead?	none	moderate-high
supports all code?	no	yes
sequential semantics?	yes	no

MELD

Merging Execution- and Language-level Determinism

	language-level	execution-level	hybrid
examples	Jade, DPJ	DMP, Kendo	MELD
runtime overhead?	none	moderate-high	low
supports all code?	no	yes	yes
sequential semantics?	yes	no	no

90% of execution time is spent in 10% of the code

90% of execution time is spent in 10% of the code

often data-parallel

program composition

program composition

locks

condition
variables

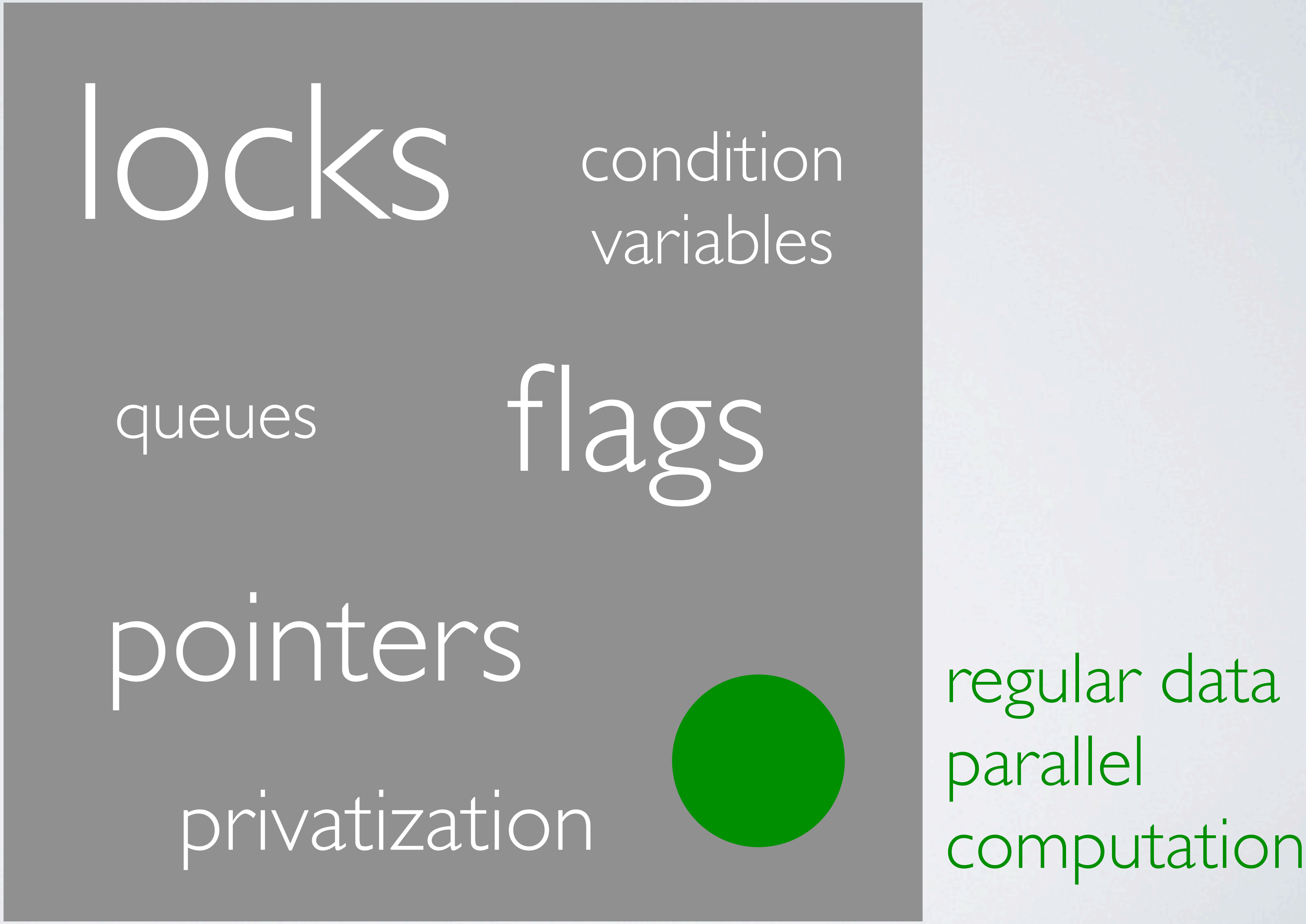
queues

flags

pointers

privatization

program composition



locks

condition
variables

queues

flags

pointers

privatization

regular data
parallel
computation

program composition

locks

condition variables

queues

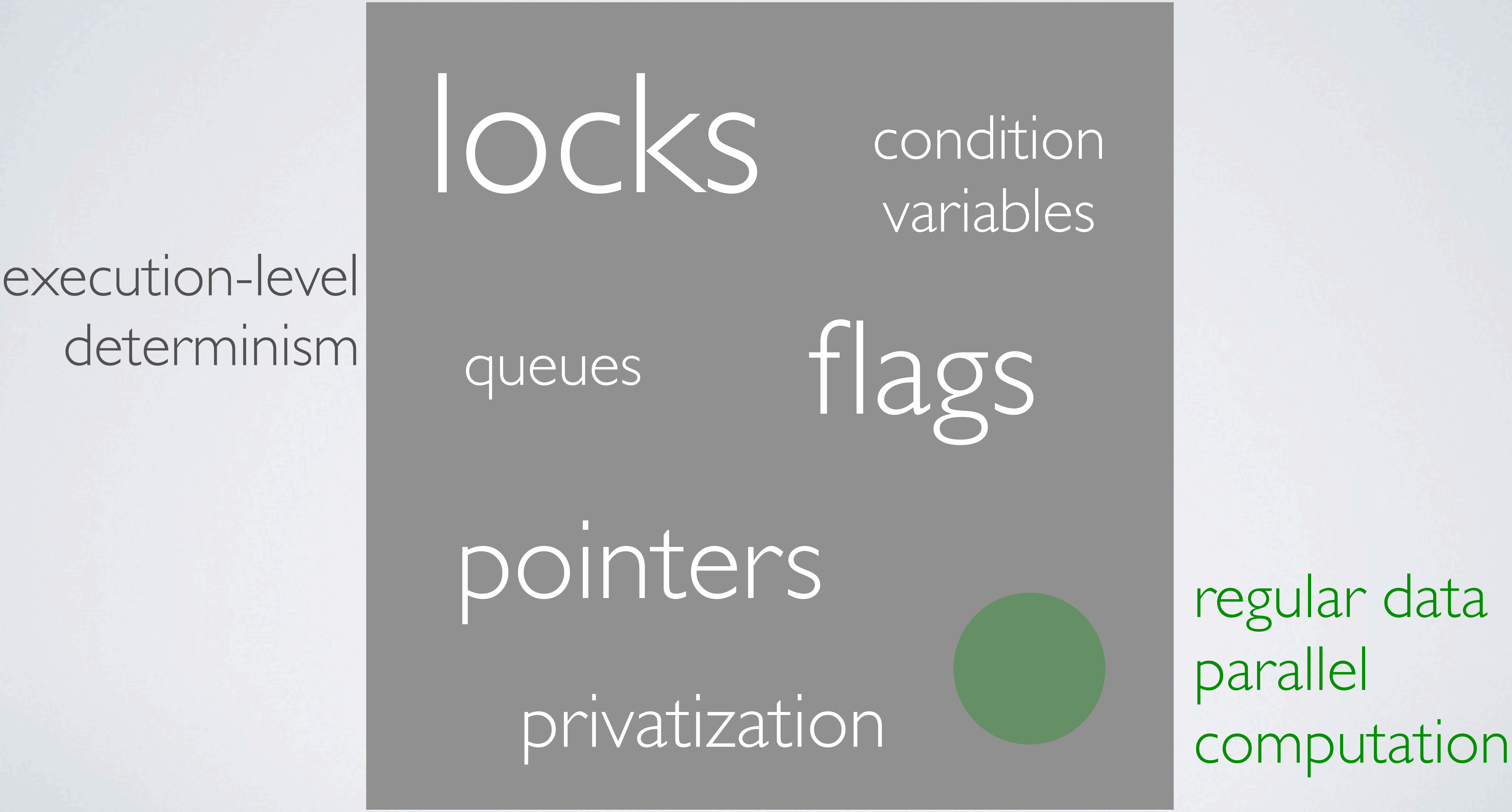
flags

pointers

privatization

regular data parallel computation

program composition



locks

condition variables

execution-level determinism

queues

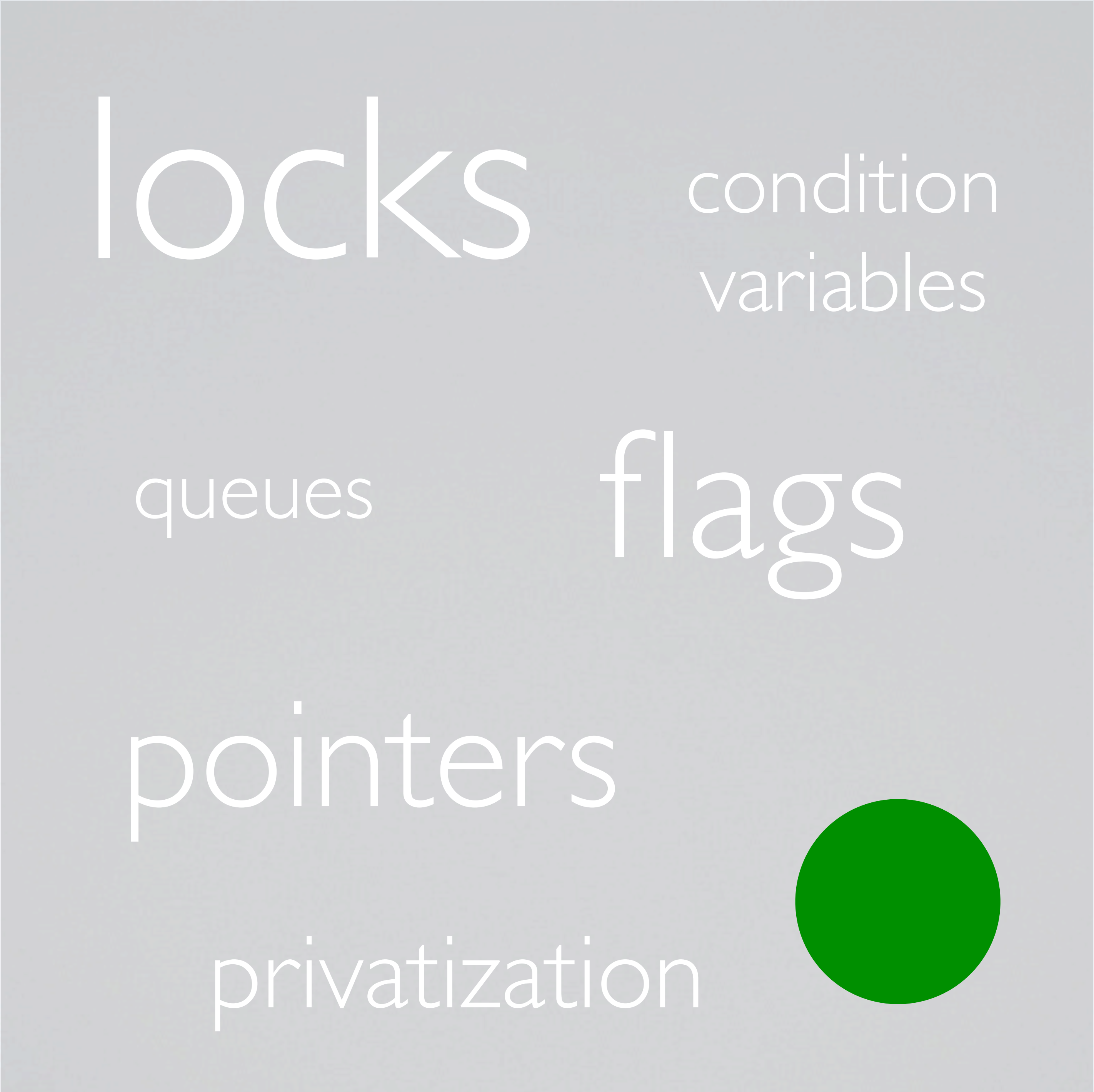
flags

pointers

privatization

regular data parallel computation

program composition



locks

condition variables

queues

flags

pointers

privatization

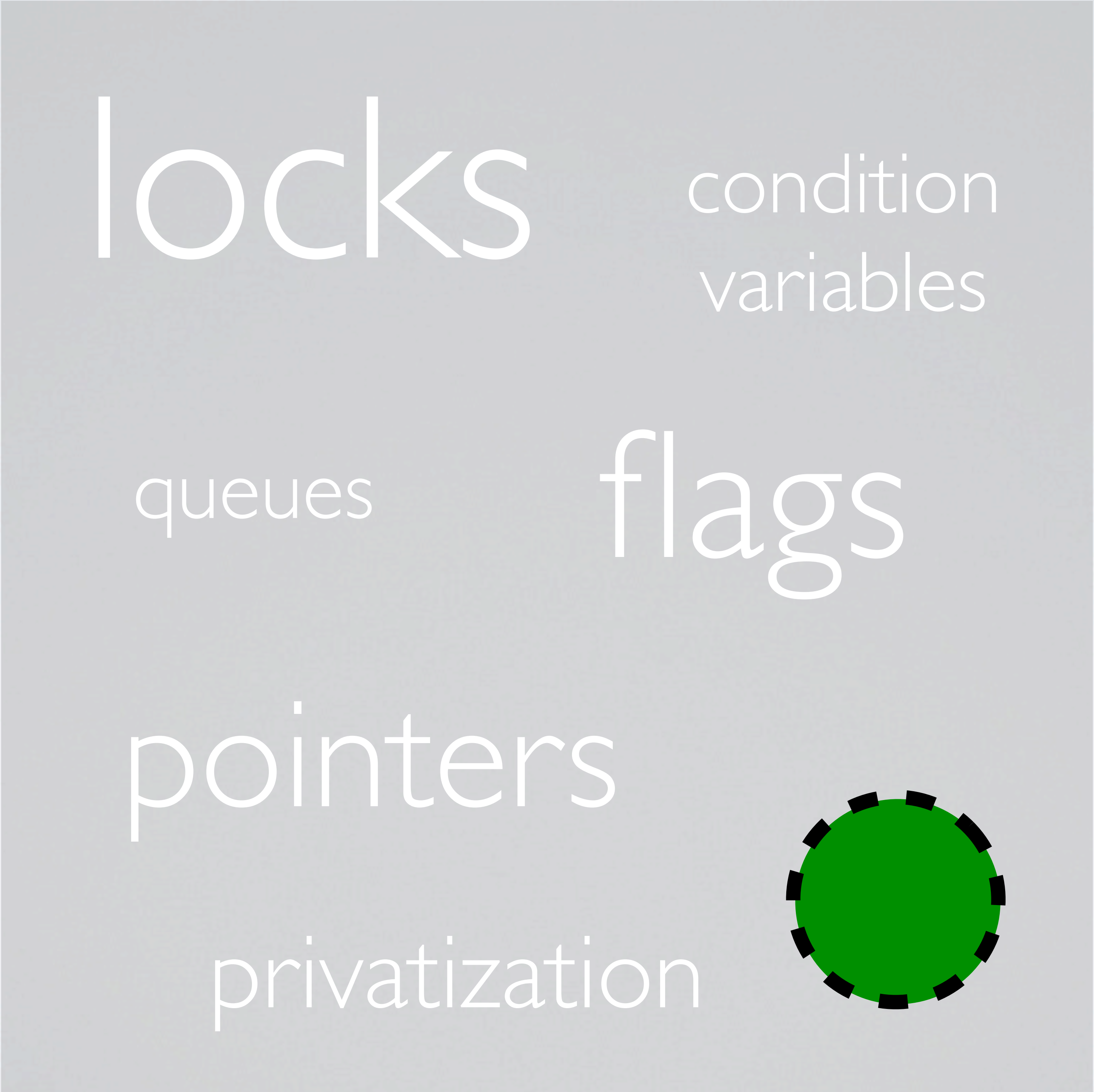
execution-level determinism

language-level determinism

regular data parallel computation

program composition

execution-level
determinism



locks

condition
variables

queues

flags

pointers

privatization

language-level
determinism

regular data
parallel
computation

what could possibly go wrong?

```
mergesort(int* array) {
  // verified by det lang
}
```

what could possibly go wrong?

what other threads call `mergesort` concurrently?

```
mergesort(int* array) {
  // verified by det lang
}
```

what aliases `array`?

can other threads concurrently access `array`?

what could possibly go wrong?

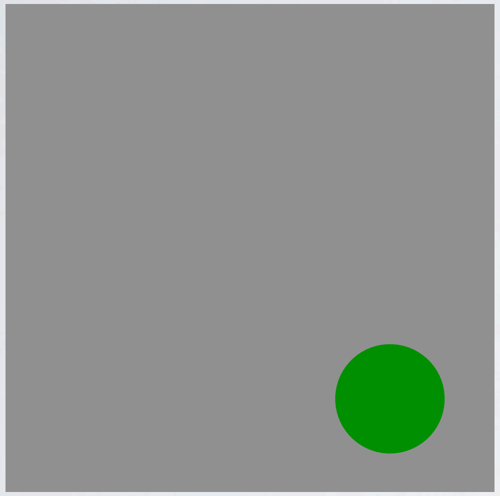
what other threads call `mergesort` concurrently?

```
        langdet
mergesort(int* array) {
  // verified by det lang
}
```

what aliases `array`?

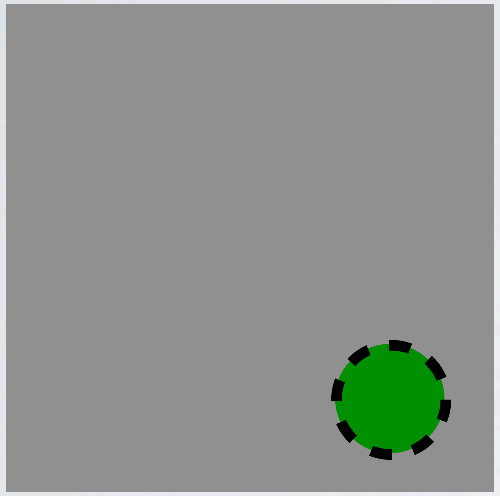
can other threads concurrently access `array`?

compilation flow

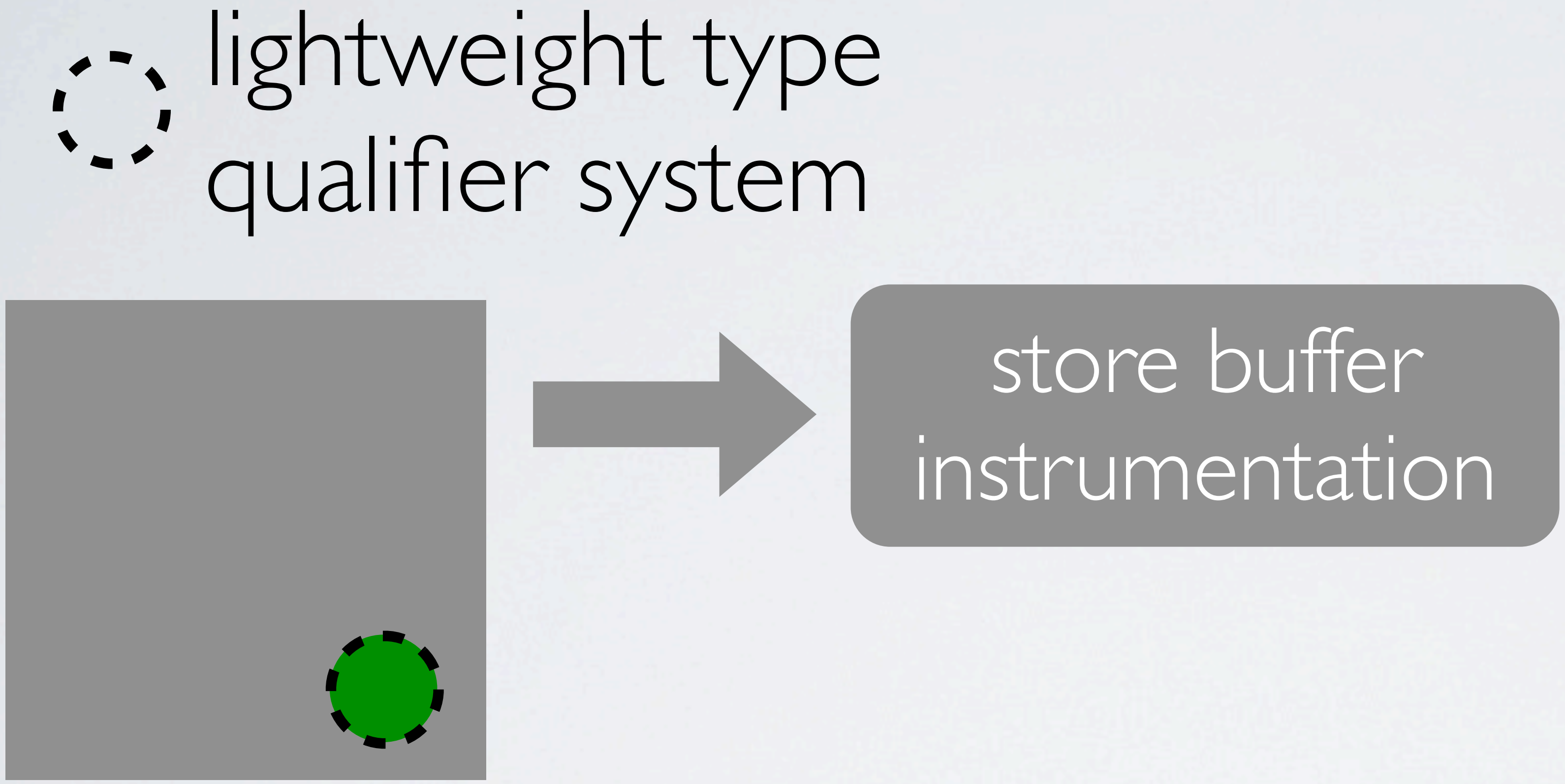


compilation flow

◌ lightweight type qualifier system



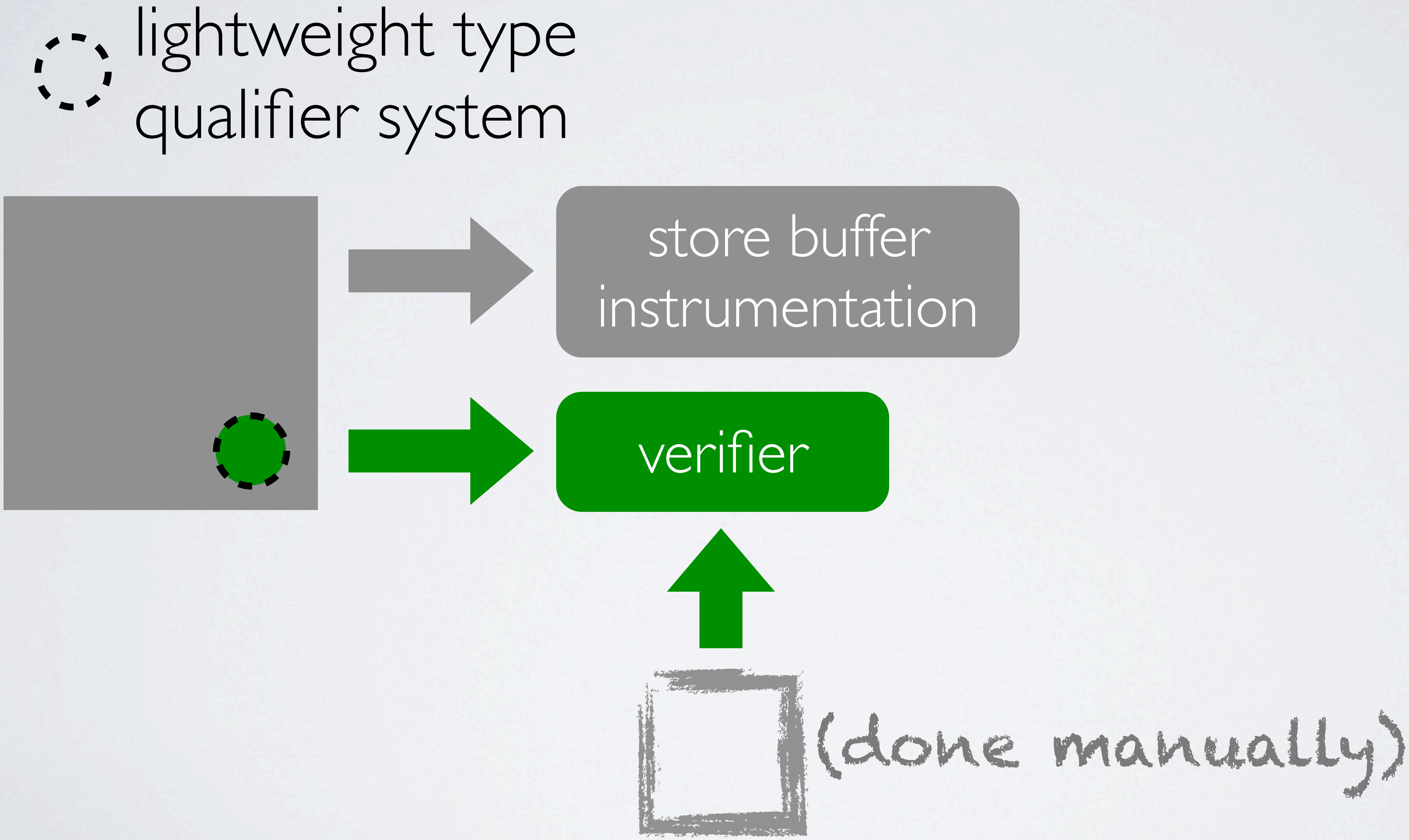
compilation flow



lightweight type qualifier system

store buffer instrumentation

compilation flow



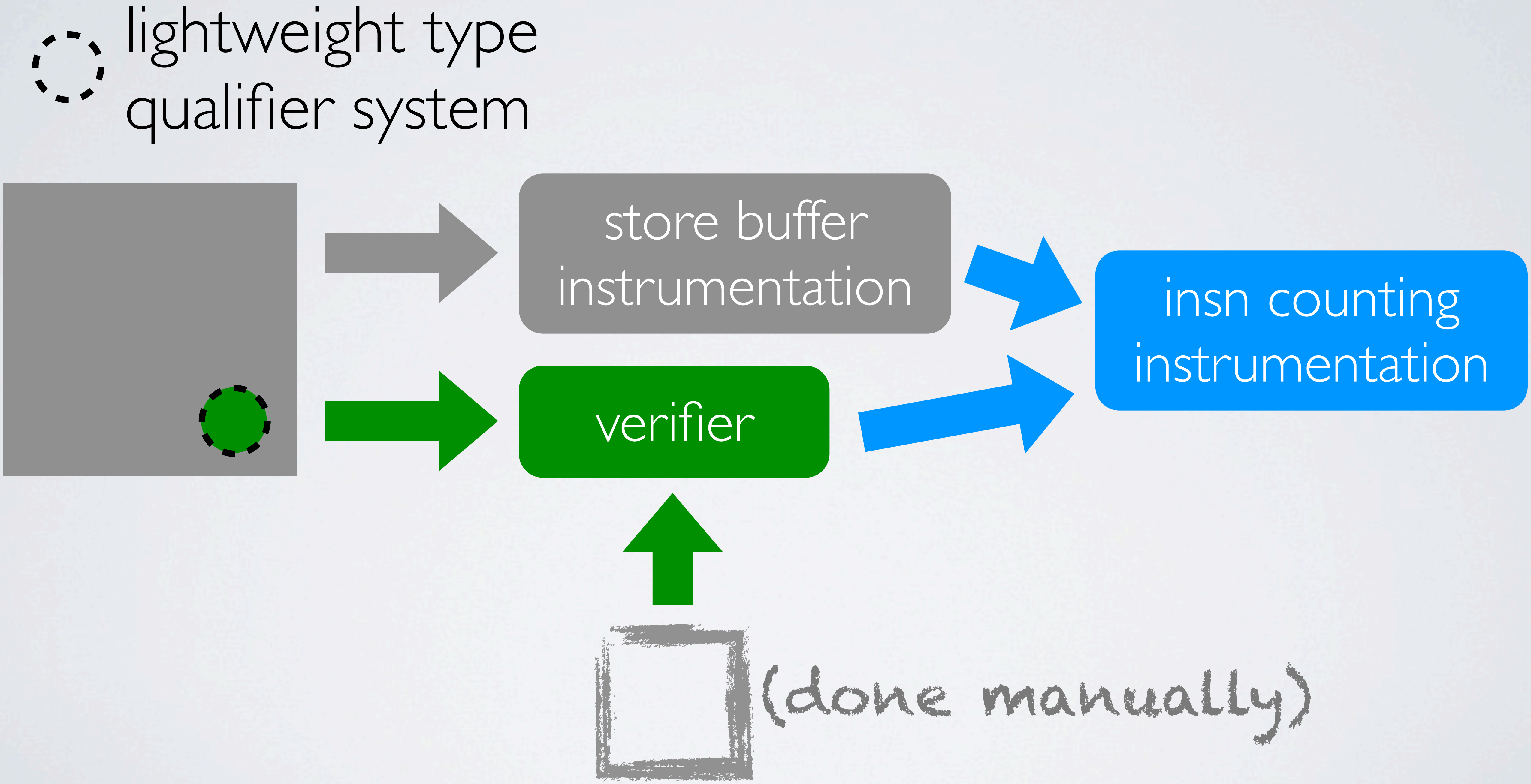
lightweight type qualifier system

store buffer instrumentation

verifier

(done manually)

compilation flow



lightweight type qualifier system

store buffer instrumentation

verifier

insn counting instrumentation

(done manually)

example: radix

```
int *dest = ...; // implicitly “exdet”
```

example: radix

```
int *dest = ...; // implicitly “exdet”

langdet int langdet *_source = cast(source);
```

example: radix

```
int *dest = ...; // implicitly “exdet”

langdet int langdet *_source = cast(source);

BARRIER();
for (int i = ...) {
  dest[COMPLICATED] = _source[i];
}
BARRIER();
```

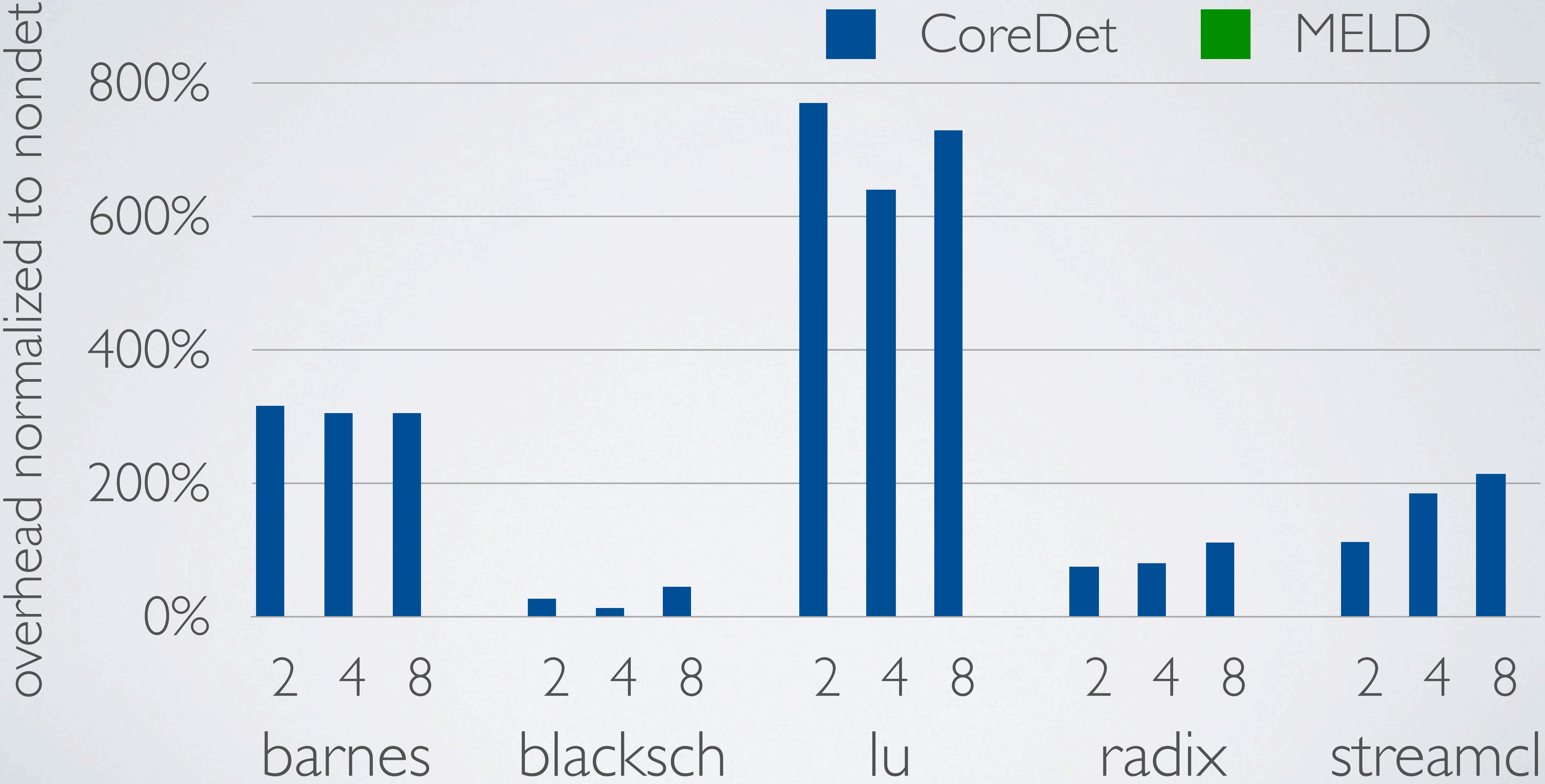
experimental setup

- 8-core 2.4GHz Intel Nehalem, 10GB RAM
- C benchmarks from SPLASH2, PARSEC
- CoreDet compiler with consistency optimizations from RC/DC [Devietti et al., ASPLOS '11]

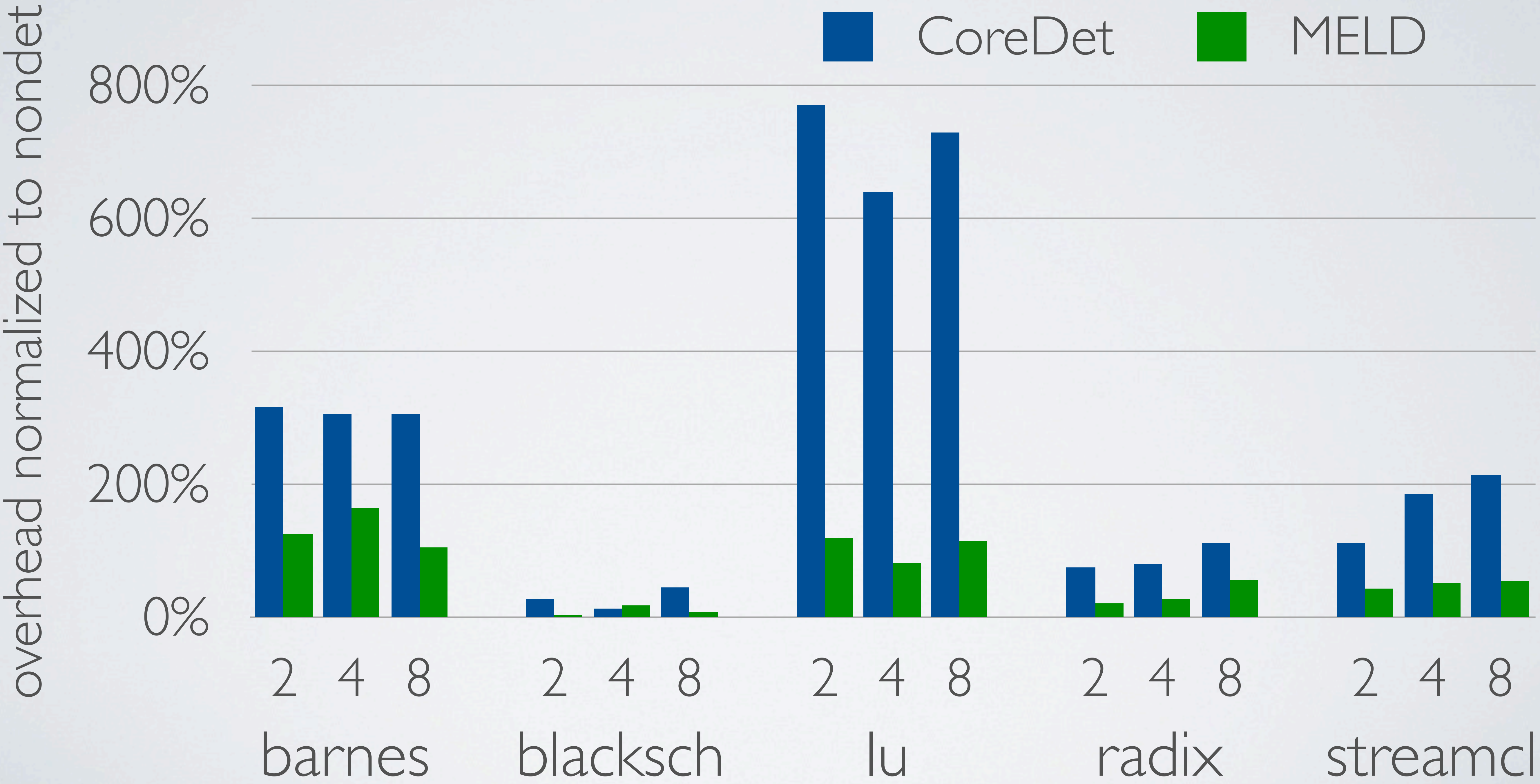
MELD results

🟦 CoreDet
🟩 MELD
overhead normalized to nondet
800%
600%
400%
200%
0%
2
4
8
barnes
2
4
8
blacksch
2
4
8
lu
2
4
8
radix
2
4
8
streamcl

MELD results



MELD results



characterization

workload	LOC	explicit sync ops (static)
barnes	2964	6
blackscholes	420	0
lu	993	1
radix	878	3
streamcluster	2347	4

usability

workload	annotations	casts
barnes	6	7
blackscholes	8	0
lu	10	3
radix	2	2
streamcluster	3	1

future work

- build fully integrated system
- supporting nondeterminism via information flow tracking type system
- find gainful employment

Questions?