Kuda:
The Split
Race Checker

Can Bekar (Koç U), Tayfun Elmas (UC Berkeley),
Semih Okur (UIUC), Serdar Tasiran (Koç U)

On-the-fly Race Checking

Traditional Race Checking

Thread 1

a1: Write(x)
Record(a1)
Analyze(a1)
a2:Read(y)
Record(a2)
Analyze(a2)
a3: Lock(l)
Record(a3)
Analyze(a3)
a4: Read(x)
Record(a4)
Analyze(a4)
a5: Unlock(l)
Record(a5)
Analyze(a5)
…

Thread 2

b1: Write(y)
Record(b1)
Analyze(b1)
b2:Read(z)
Record(b2)
Analyze(b2)
b3: Write(x)
Record(b3)
Analyze(b3)
…

Race Checking in Parallel

Thread 1

a1
Record
a2
Record
a3
Record
a4
Rec
a5
Rec
…

Thread 2

b1
Rec
b2
Rec
b3
Rec
…

Helper thread

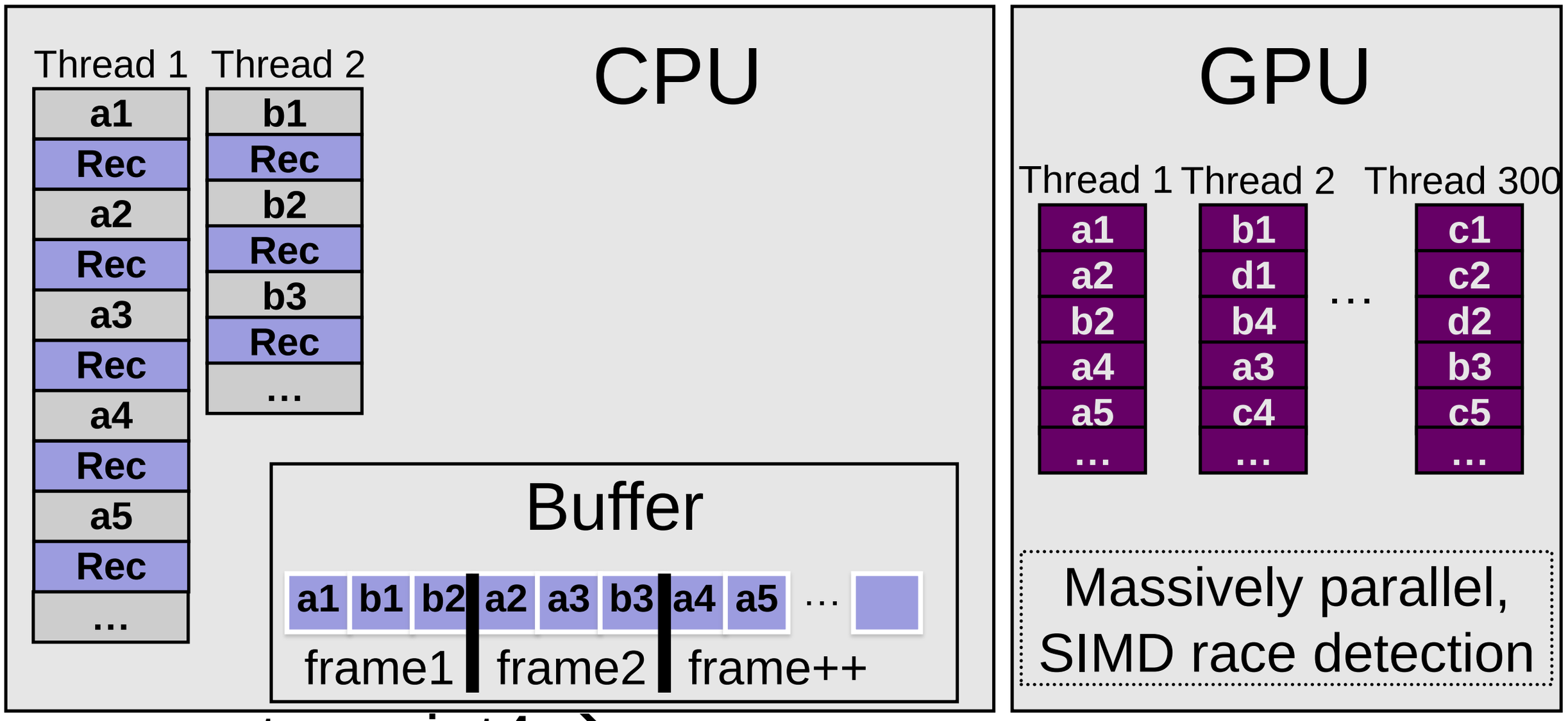
A(a1)
A(b1)
A(a2)
A(a3)
A(b2)
A(b3)
A(a4)
A(a5)
…

Looks so nice!

Intel's *Thread Checker,* **Google's** *ThreadSanitizer and* **Helgrind** → each tool may cause several hundreds times slowdown!

(see papers)

Producer/Consumer Relation on all CPU Records



an event = int4 → event{varID,threadID,type,index}

a frame = 1024 events

the buffer = 2048 frames = 16 MB

Does It Work?

Baseline Program (computation only) 3.8 sec

\+ Runtime instrumentation 3x

\+ Cheap Algorithm (on CPU)

- Eraser – Simple/Fast & Imprecise ~400x

\+ Event buffering 78x

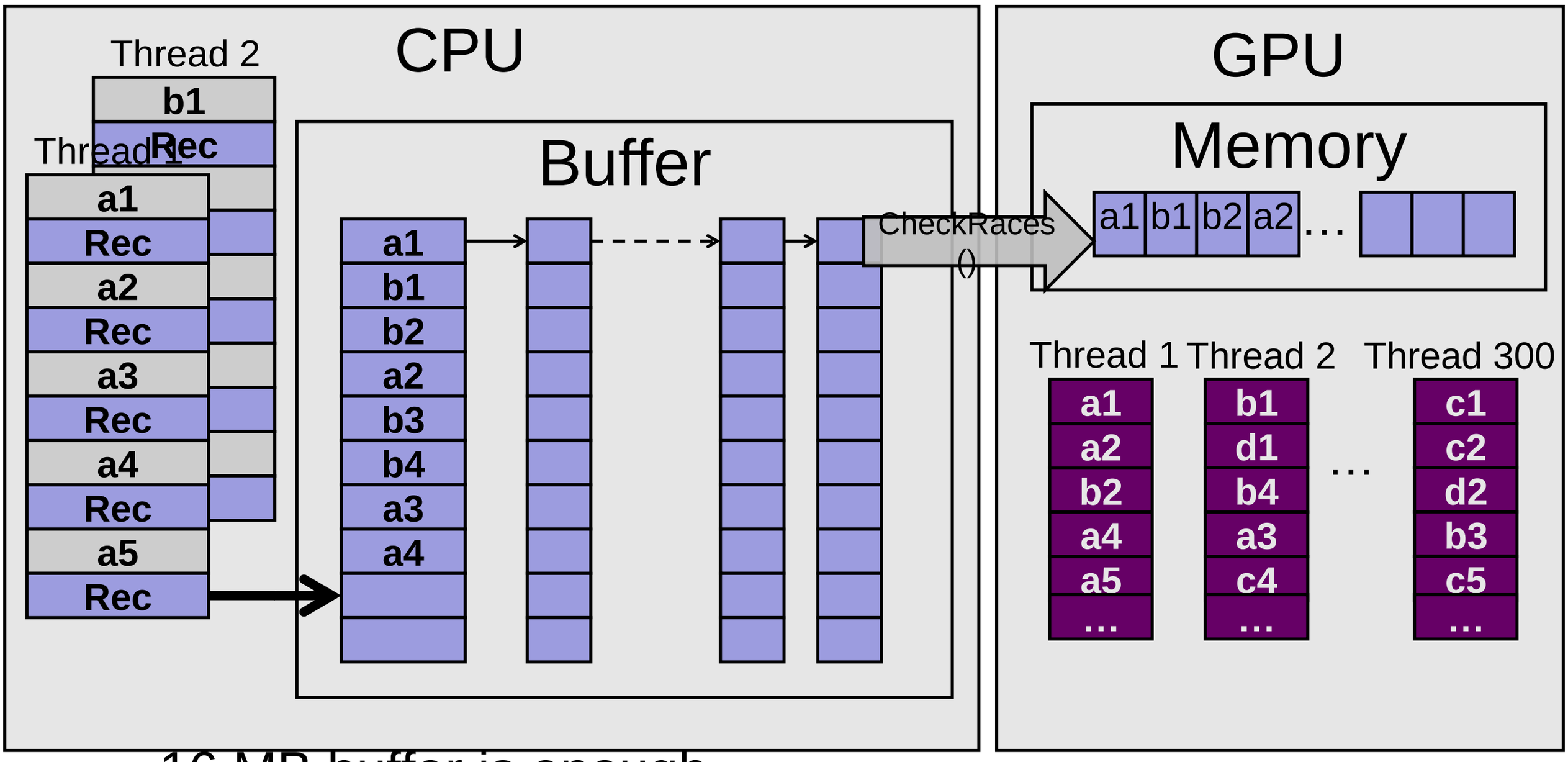
\+ Expensive Algorithm (on GPU)

- Goldilocks – Complex & Precise 83.6x

= Precise race detection on GPU is 5x faster than imprecise race detection on CPU

(see experiments, 2 slides later)

Non-blocking, Lock-free Buffer



16 MB buffer is enough

Always room for the next event to be recorded (GPU is fast)

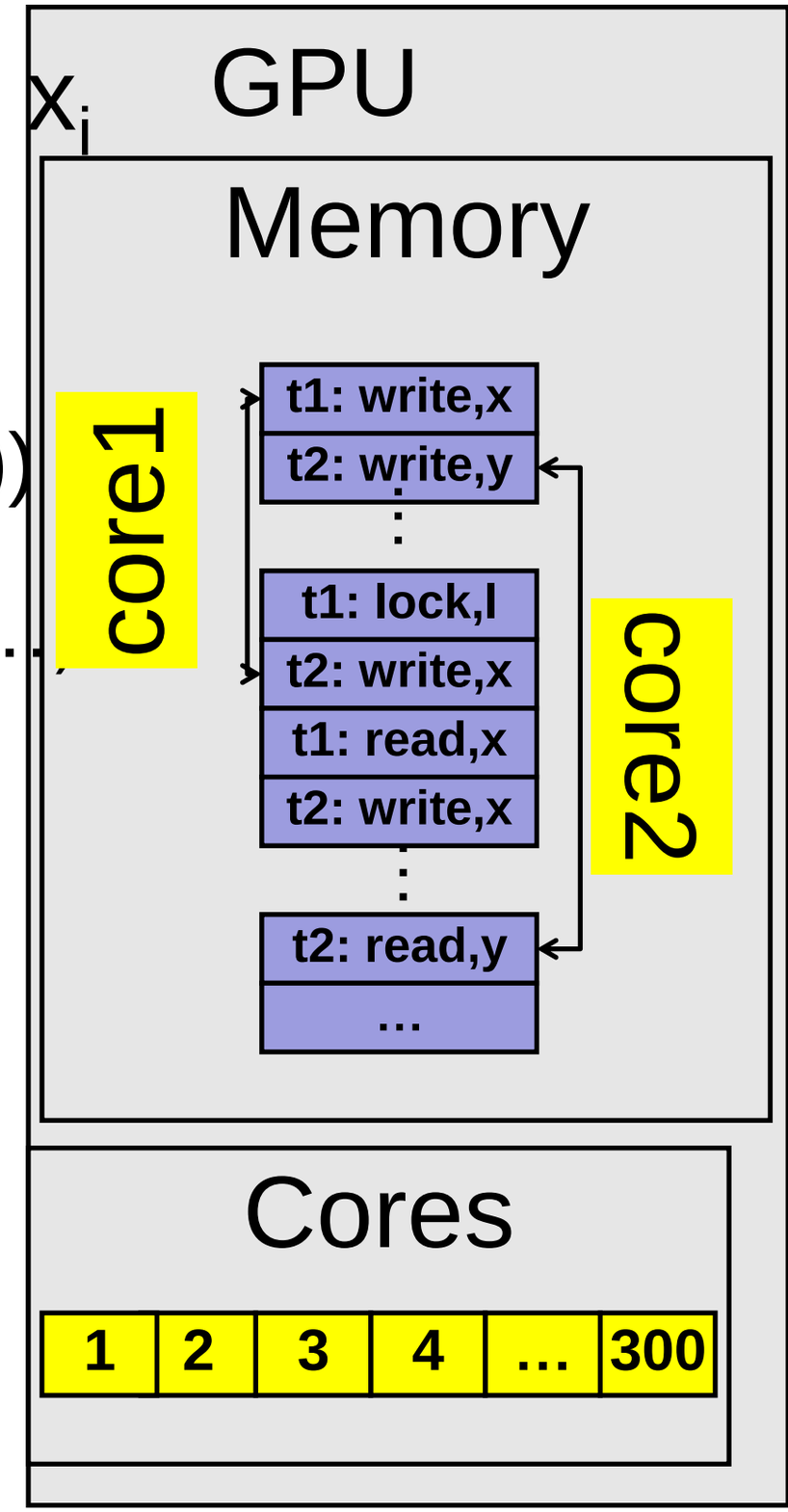
Parallel Race Checking

- Suppose event[i] accesses var x_i
- GPU core[i]

→ handle(event[i] *until* next_event(x_i))

when done → handle(event[i+N]…)

- Read-only device memory!
- No shared data besides the frame



Benchmarks	Normal exe	Instrumented	Eraser@CPU	Event buffer	Goldilocks@GPU
	Reference	Slowdown	Slowdown	Slowdown	**Slowdown**
blackscholes	1	2.1	101	14.7	**21.1**
bodytrack	1	2.6	251	72.5	**75.6**
canneal	1	1.6	47	6.9	**7.4**
dedup	1	3.1	429	88	**101.6**
fluidanimate	1	2.5	308	83.9	**88.1**
raytrace	1	2	123.7	5.7	**6.2**
swaptions	1	3	437	117.5	**119.9**
x264	1	7.1	645.2	151.7	**155.8**
barnes	1	4	499.1	111.5	**116.1**
cholesky	1	3.2	216.2	32.4	**33.2**
fmm	1	2.9	1455.8	89.6	**98.6**
fft	1	2.1	222.7	45.3	**48.1**
lu	1	5.9	742.2	190	**201.9**
ocean	1	4	301.6	63.4	**68.1**
radix	1	2	126.6	31.1	**33.1**
raytrace	1	2	122.2	6.9	**7.2**
water-n2	1	3.5	707.6	182.1	**191.4**
w-spatial	1	4.6	485.2	114.9	**131.5**
average	**1x**	**3.2x**	**401x**	**78x**	**83x**

Future Work

- Profiling + Pruning of thread local vars
- Compare state-of-the-art RC in C(Fast-track)
- Offloading other concurrent analysis to GPU
- Reduce slowdown due to buffering
  - Custom HW

Q&A

Our codebase is online:

kuda.codeplex.com