Efficient Data Race Detection of Async-Finish Programs Using Vector Clocks

Shivam Kumar
IIT Kanpur

Anupam Agrawal
IIT Kanpur

Swarnendu Biswas
IIT Kanpur

Programming Models and Applications for Multicores and Manycores (PMAM 2022)

A Racy Java Program

```
Object x = null;
Boolean done = false;
```

Thread T1

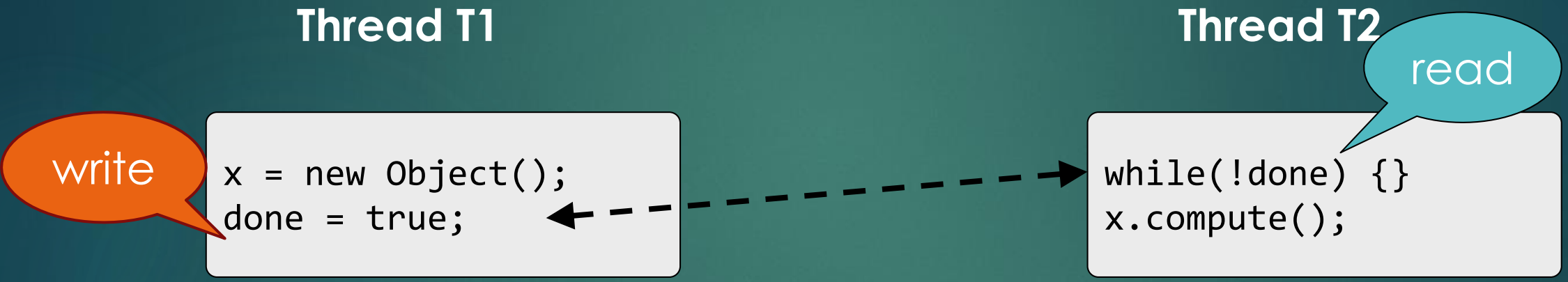
```
x = new Object();
done = true;
```

Thread T2

```
while(!done) {}
x.compute();
```

Data Race

```
Object x = null;
Boolean done = false;
```



Data race

Conflicting accesses – two threads access the same shared variable where at least one access is a write

Concurrent accesses – accesses are not ordered by synchronization operations

Data Races Are Bad!!!

```
Object x = null;
Boolean done = false;
```

Thread T1

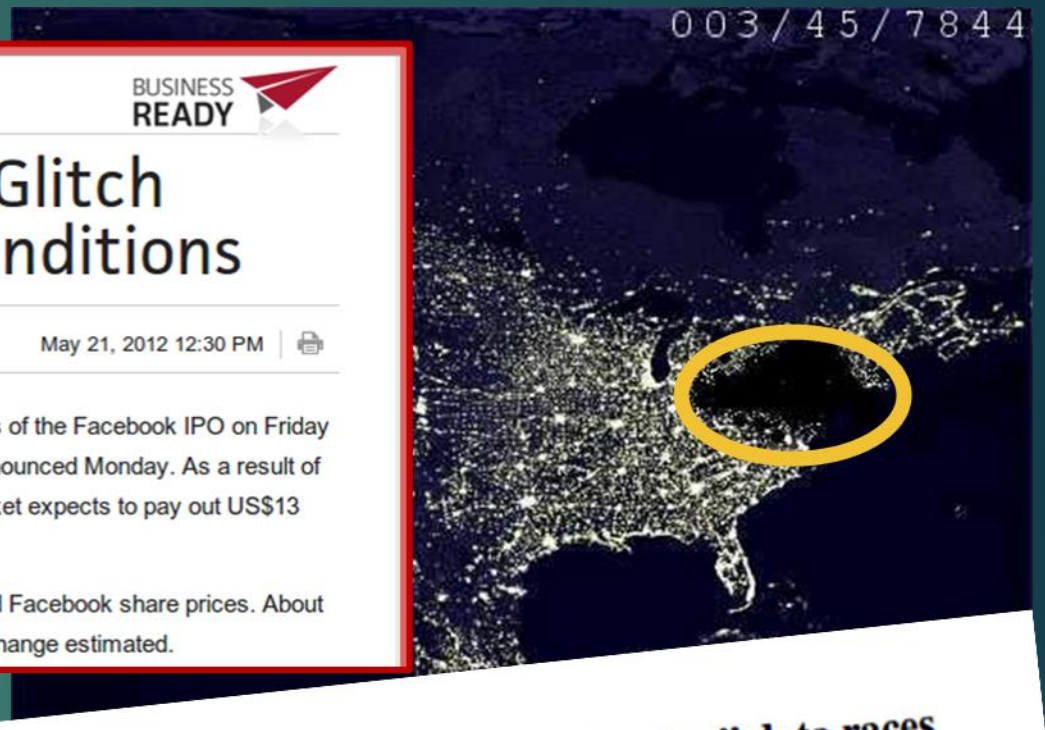
```
done = true;

x = new Object();
```

Thread T2

```
while(!done) {}
x.compute();
```

Impact of Data Races

Therac-25 accidents, 1985-87



BUSINESS SOFTWARE
business

BUSINESS READY

Nasdaq's Facebook Glitch Came From Race Conditions

Joab Jackson
@Joab_Jackson

May 21, 2012 12:30 PM

The Nasdaq computer system that delayed trade notices of the Facebook IPO on Friday
change announced Monday. As a result of
n, the market expects to pay out US$13

nismatched Facebook share prices. About
ed, the exchange estimated.

research highlights

Technical Perspective
Data Races are Evil with No Exceptions

By Sarita Adve

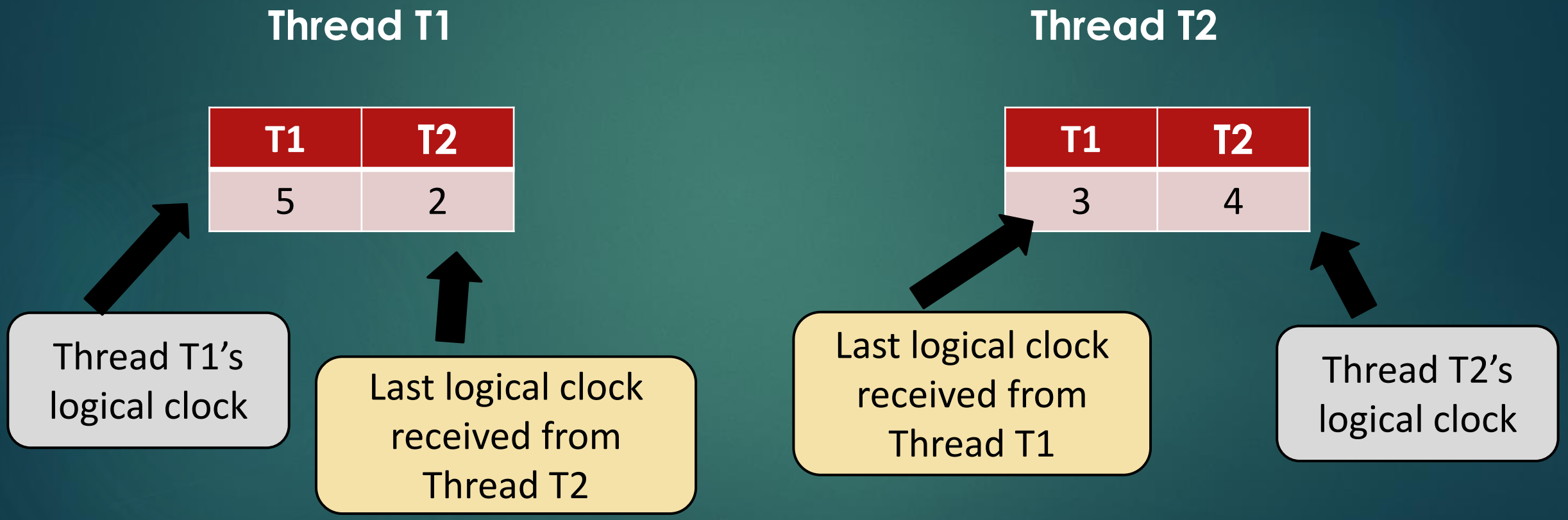
EXPLOITING PARALLELISM HAS become the primary means to higher performance.

racy code. Java's safety requiremer
preclude the use of "undefined" beh

How to miscompile programs with "benign" data races

Hans-J. Boehm
HP Laboratories

Vector Clock Based Race Detection



Detecting Data Races Using FastTrack

NULL

Var x

Thread T1

T1	T2
5	2

Thread T2

T1	T2
3	4

C. Flanagan and S. Freund. FastTrack: Efficient and Precise Dynamic Data Race Detection. PLDI, 2009.

Detecting Data Races Using FastTrack

5@T1

Var x

Thread T1

T1	T2
5	2

write(x)

Thread T2

T1	T2
3	4

Detecting Data Races Using FastTrack

5@T1

Var x

Thread T1

T1	T2
5	2

write(x)

Thread T2

T1	T2
3	4

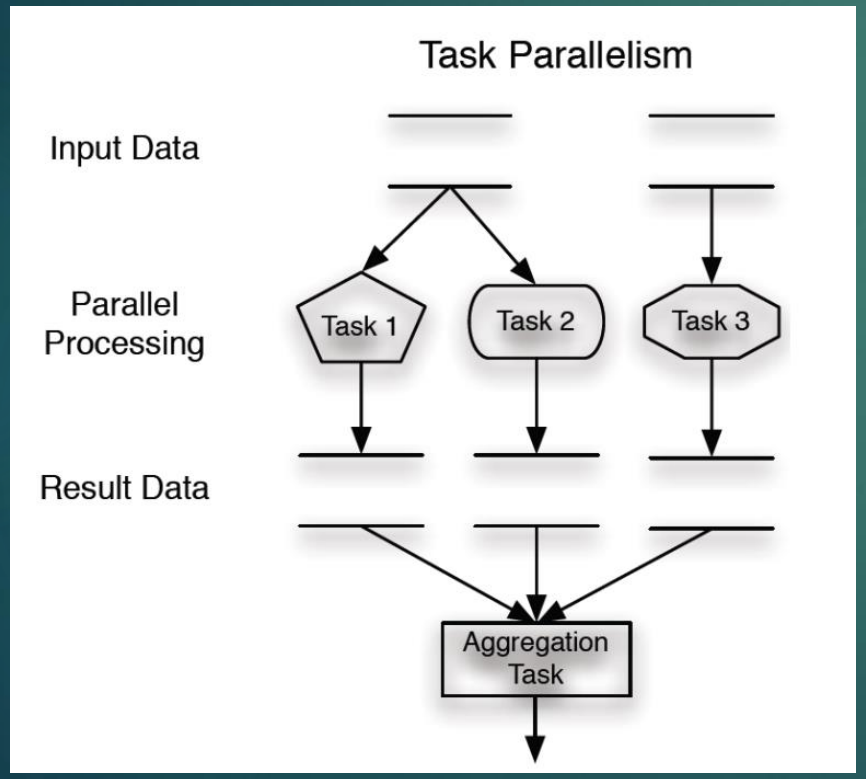
5@T1 > VC_{T2} [T1]

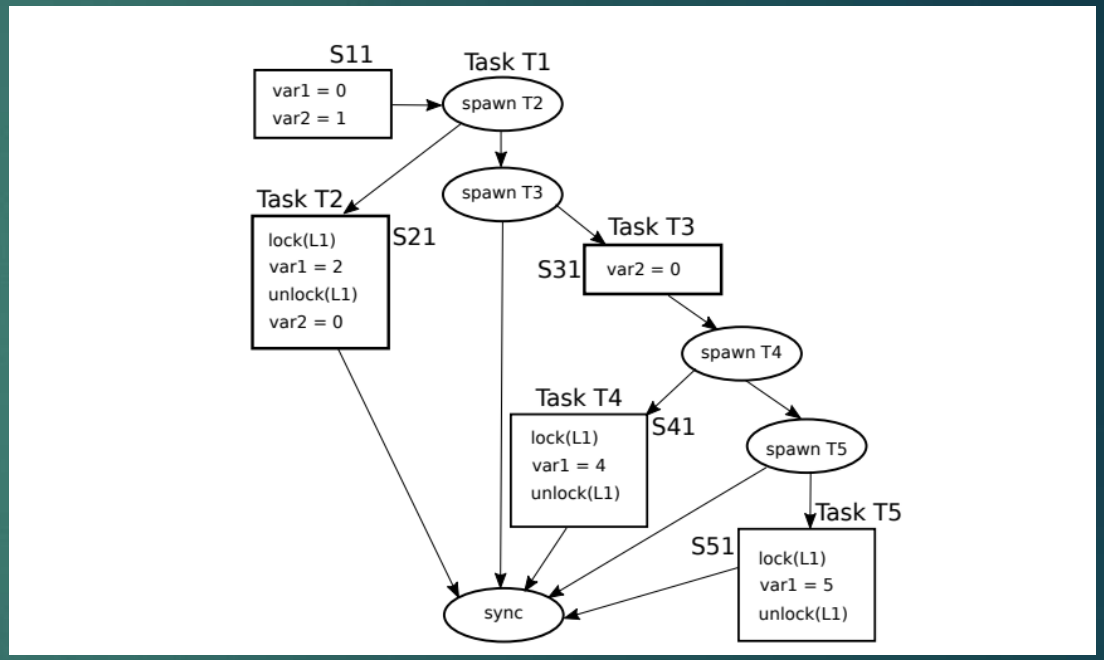
Data Race

write(x)

Task Parallel Programs

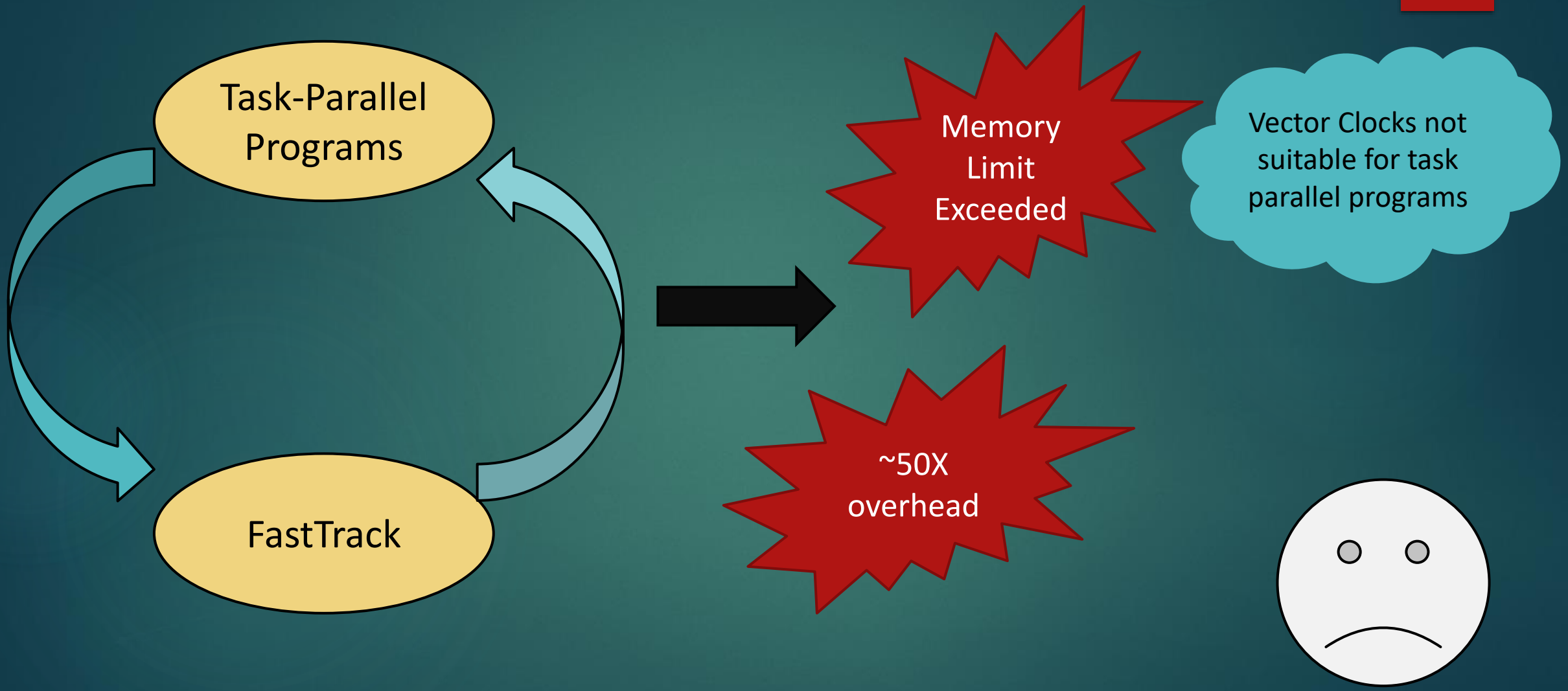
Task-Parallel Programs





Async-Finish Programs

FastTrack on Task Parallel Programs



Data Race Detection

Multi-threaded Programs

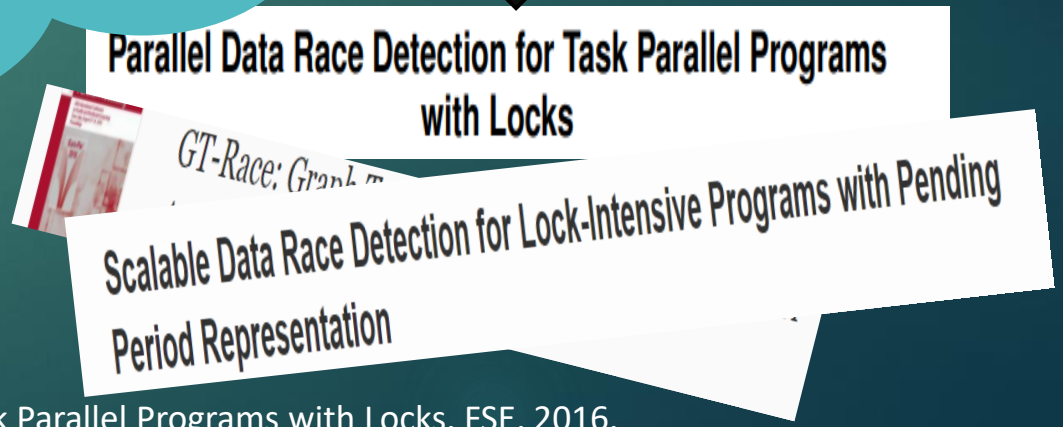
Task-Parallel Programs

Vector Clo

Based

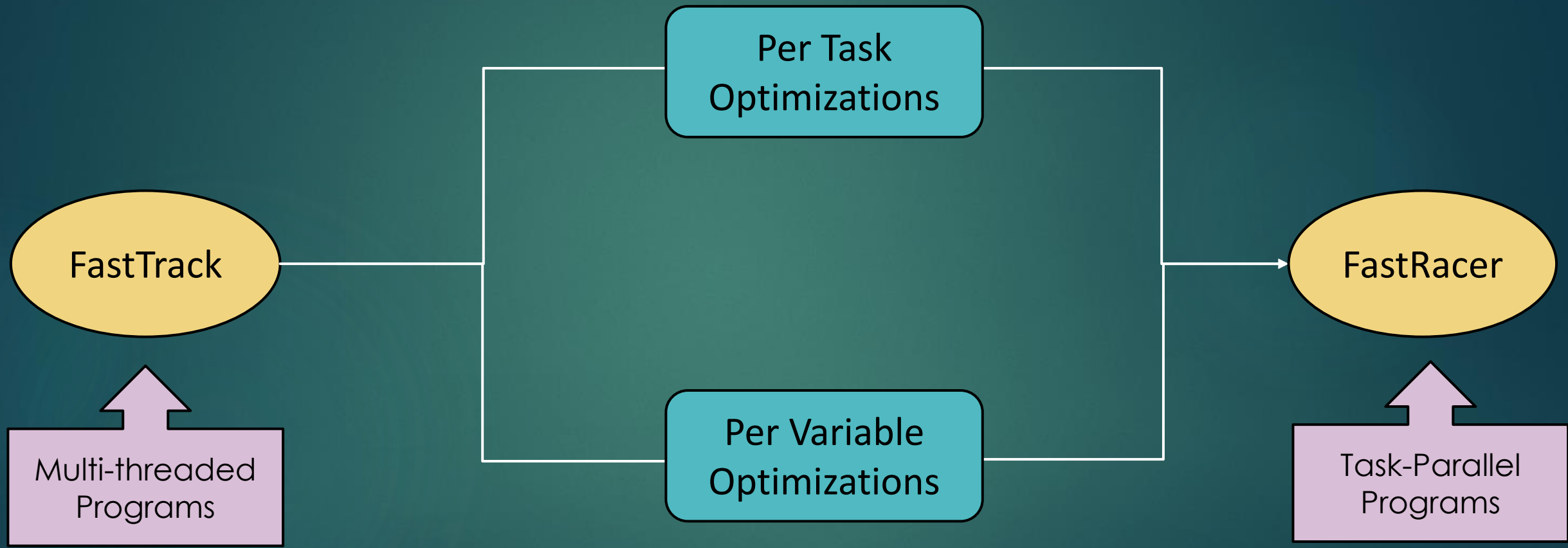
Vector clock analyses are generic, inherently parallel, and have better data locality than tree-based data structures.

FastTrack: Efficient and Precise Dynamic Race Detection

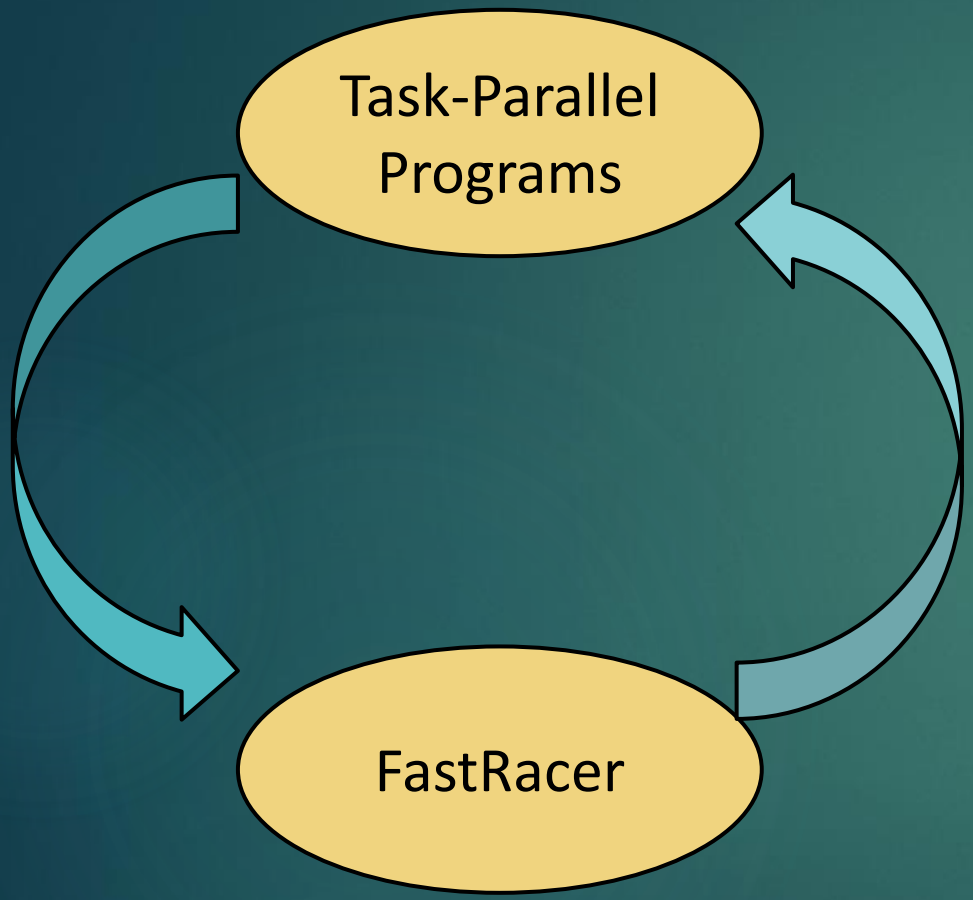


Adarsh Yoga, Santosh Nagarakatte, and Aarti Gupta. Parallel Data Race Detection for Task Parallel Programs with Locks. FSE, 2016.

FastRacer: An Efficient Dynamic Data Race Detector



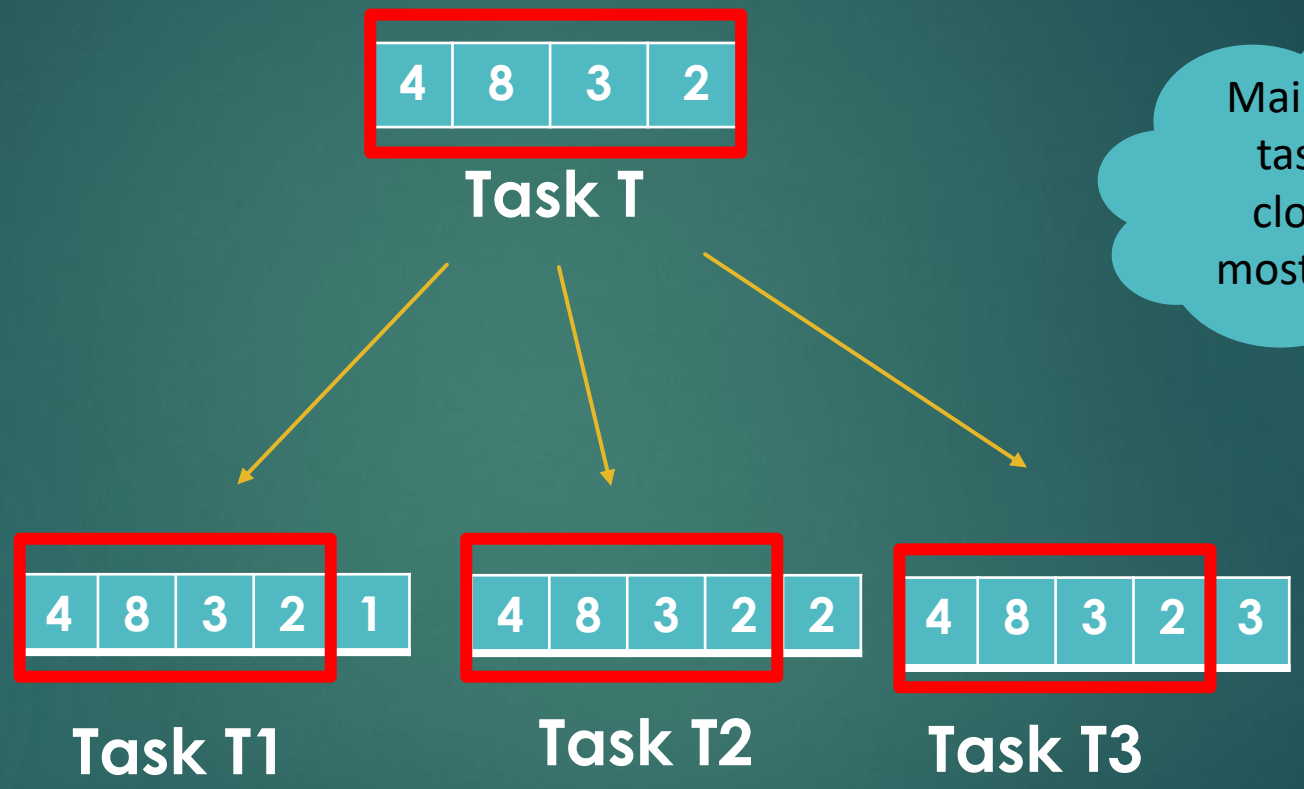
FastRacer on Task Parallel Programs



Vector clocks viable for efficient analysis of task-based programs

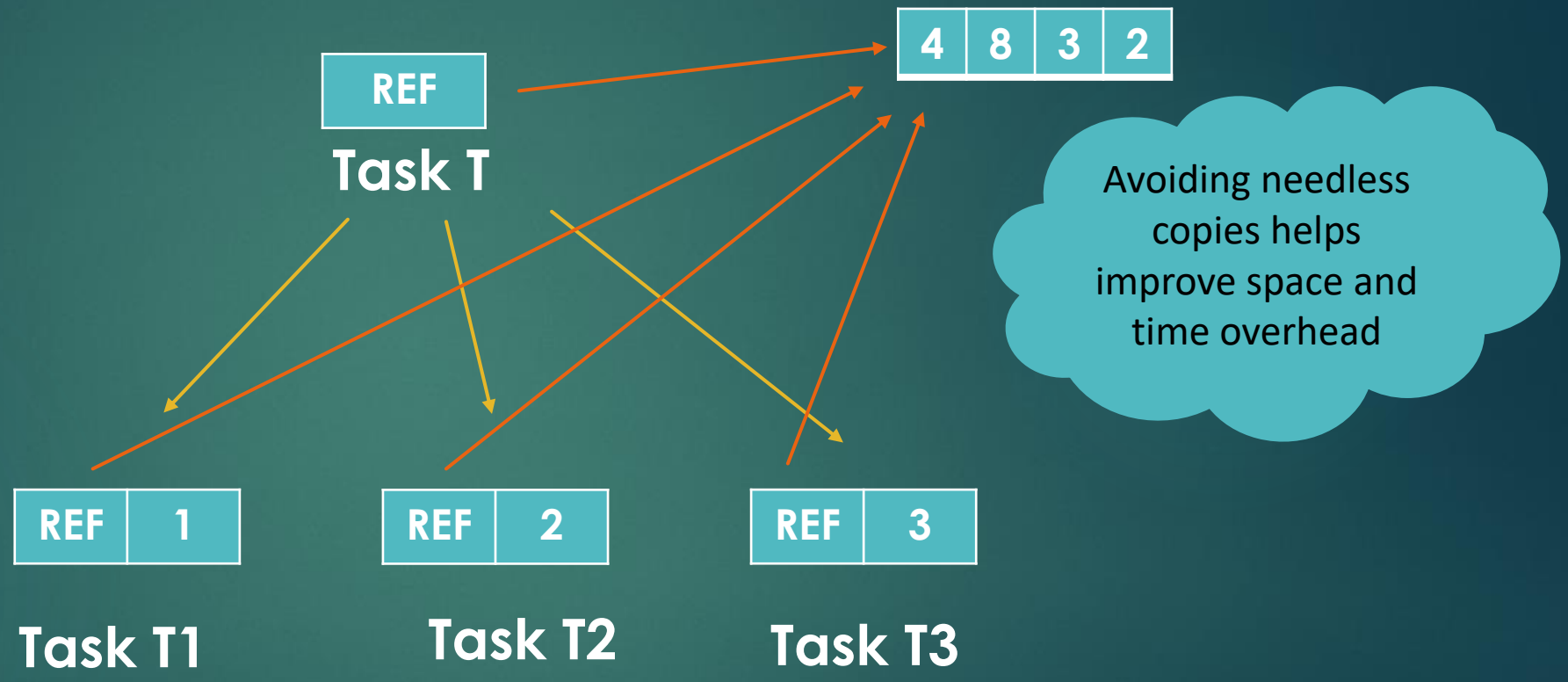
Outperforms SOTA algorithms e.g. PTRacer, C-Racer

FastTrack: Task Spawn

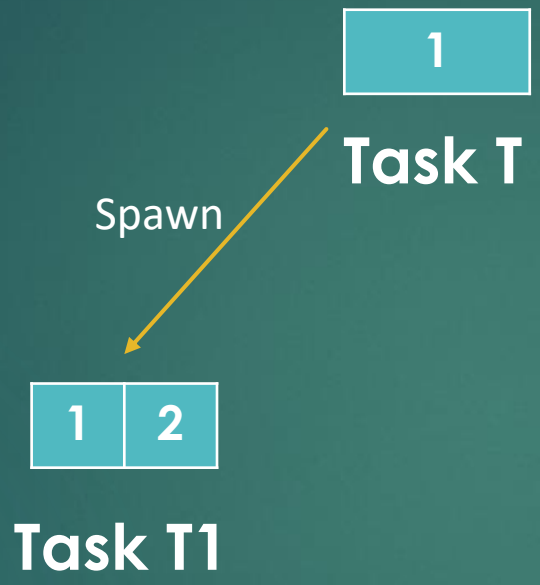


Maintaining per-task copies of clock values is mostly redundant

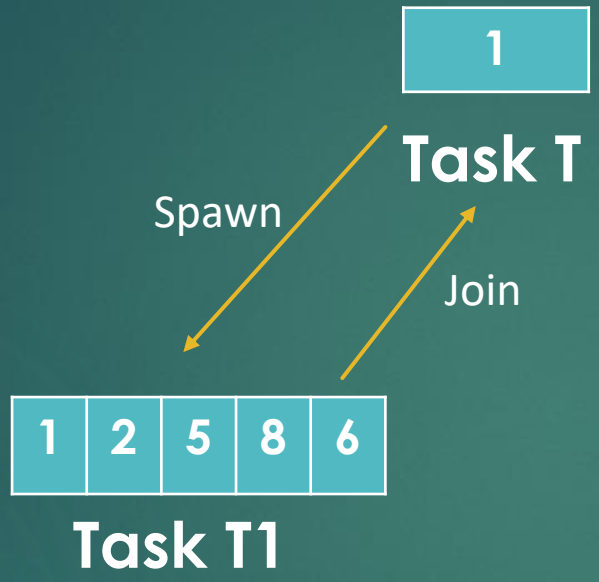
FastRacer: Task spawn



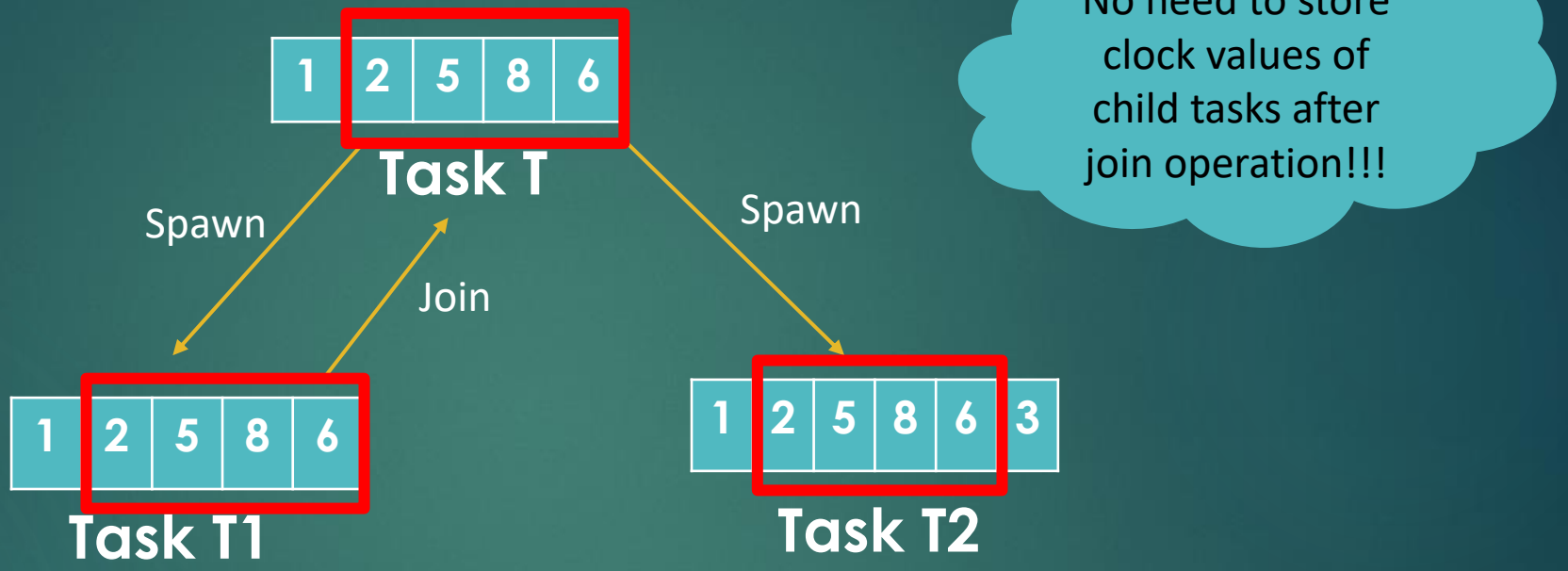
FastTrack: Task Join



FastTrack: Task Join

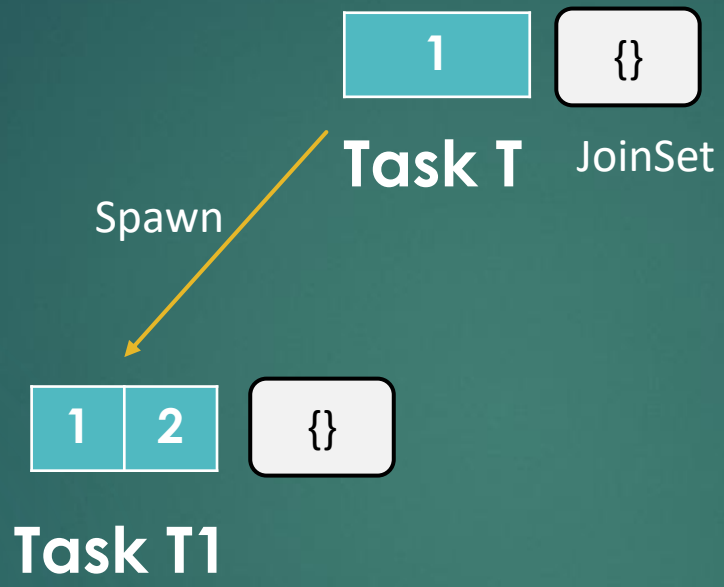


FastTrack: Task Join

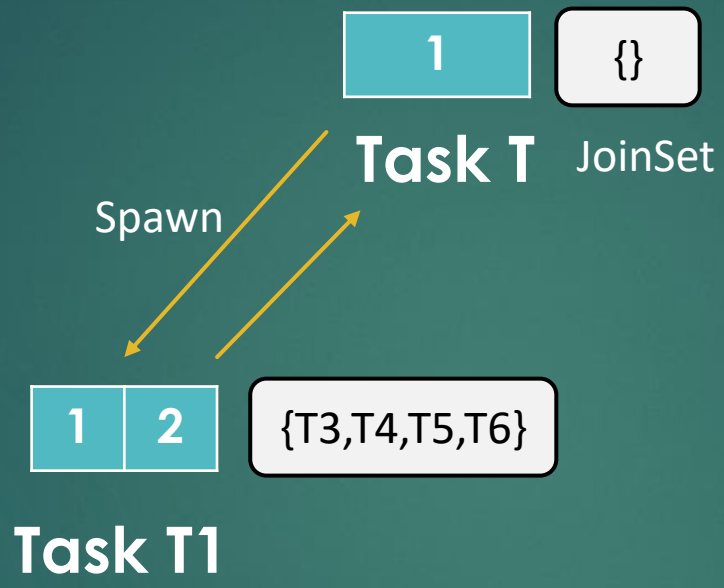


No need to store clock values of child tasks after join operation!!!

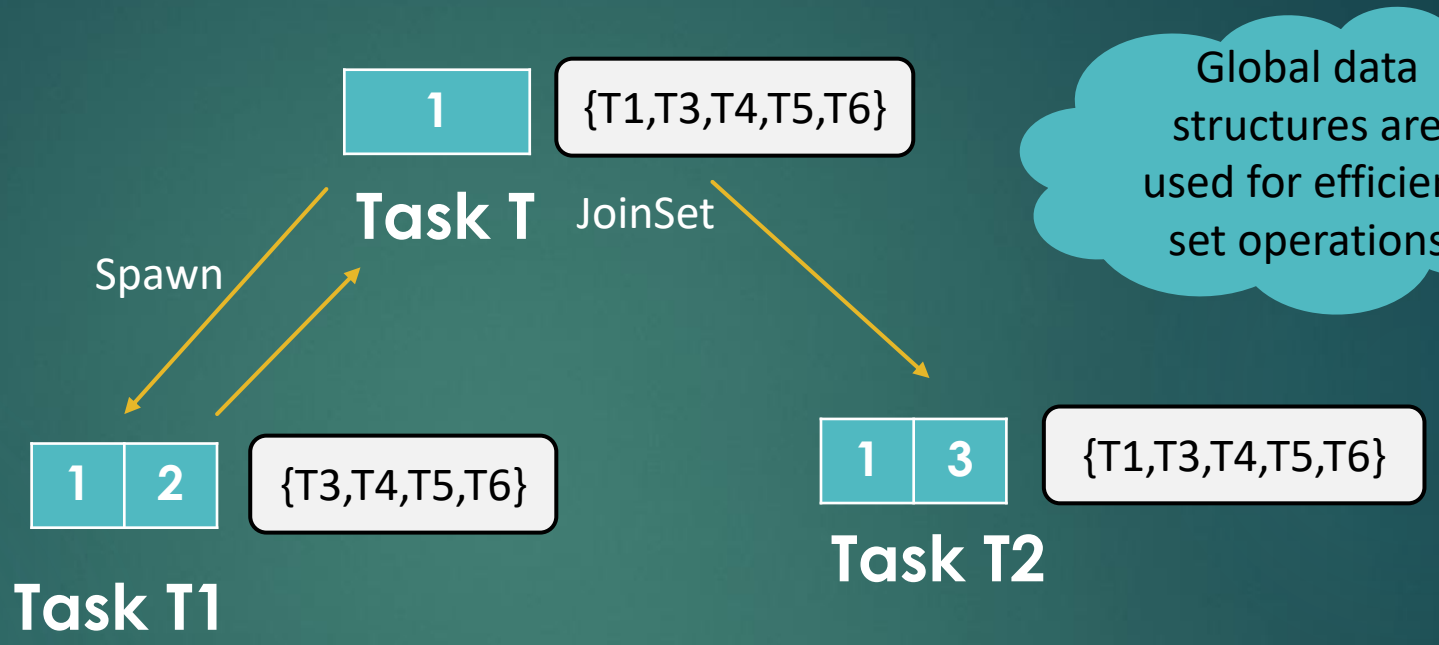
FastRacer: Task Join



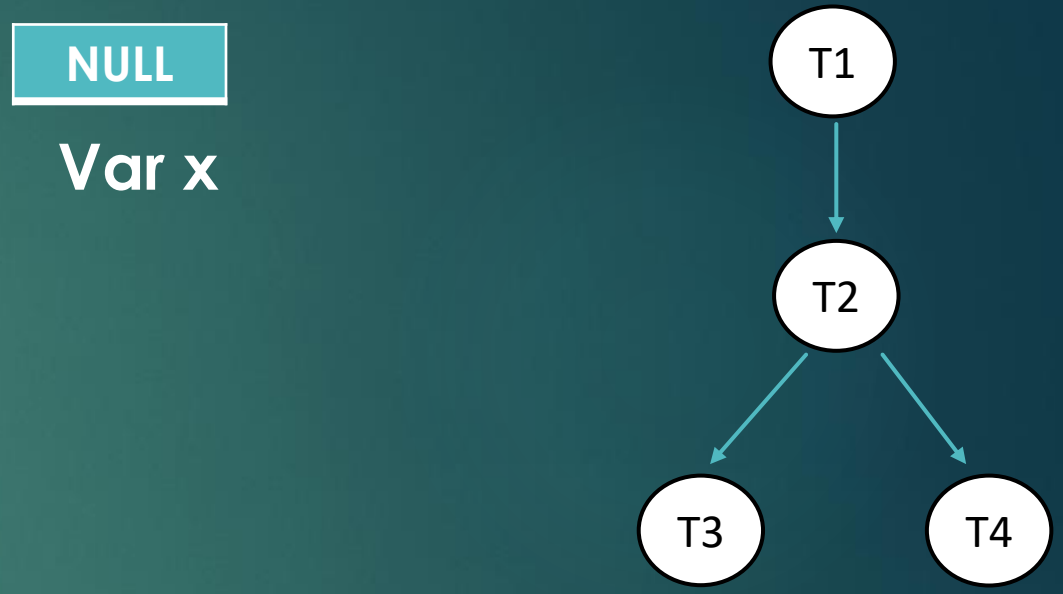
FastRacer: Task Join



FastRacer: Task Join



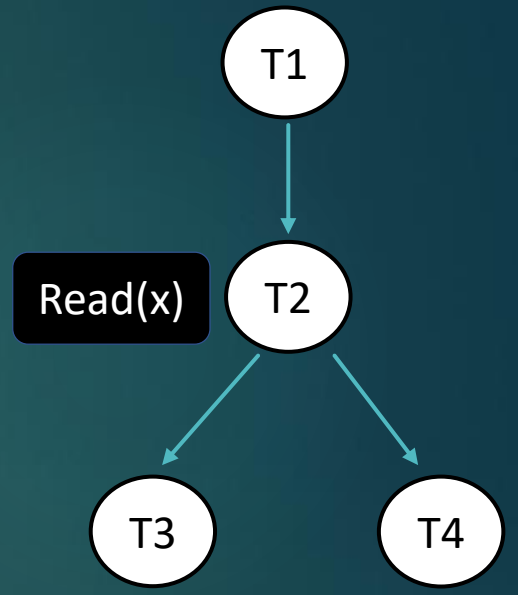
FastTrack: Variable Access



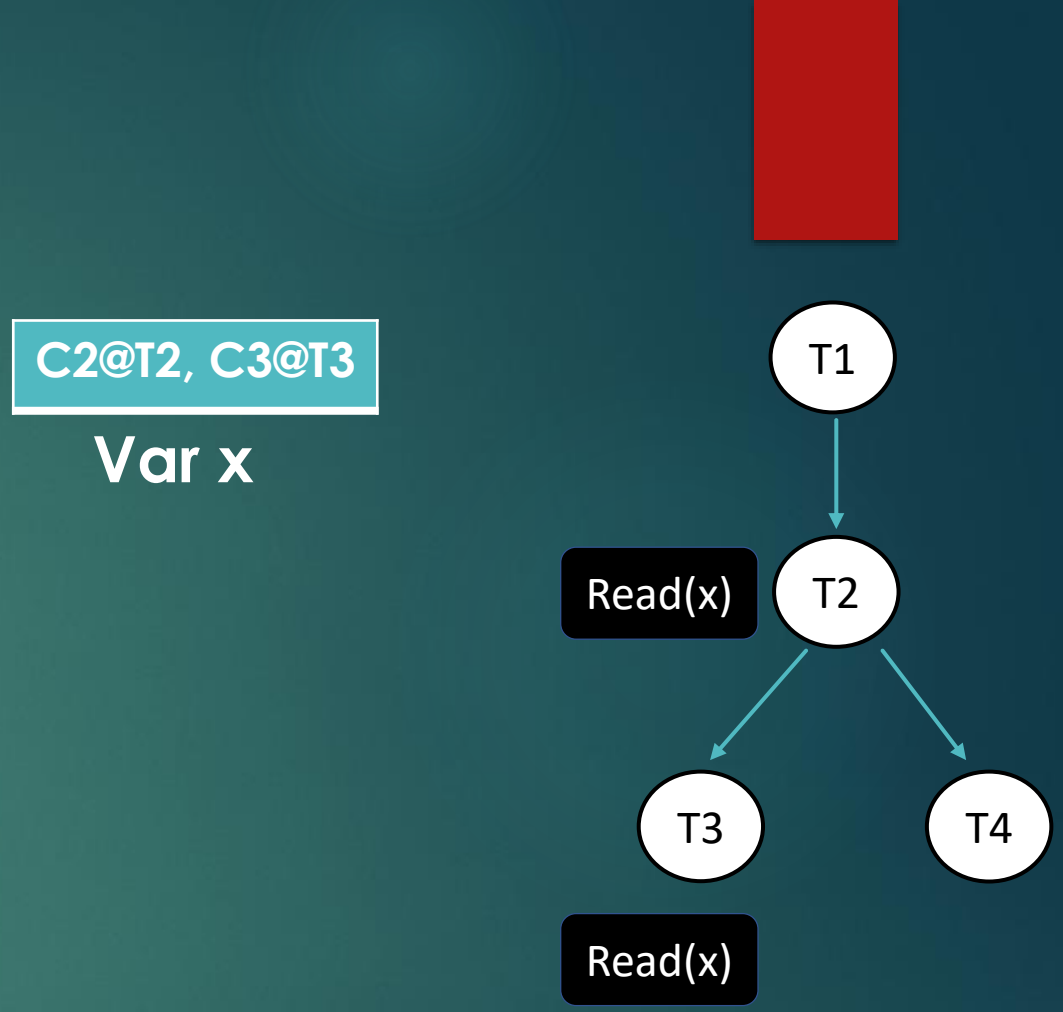
FastTrack: Variable Access

C2@T2

Var x



FastTrack: Variable Access

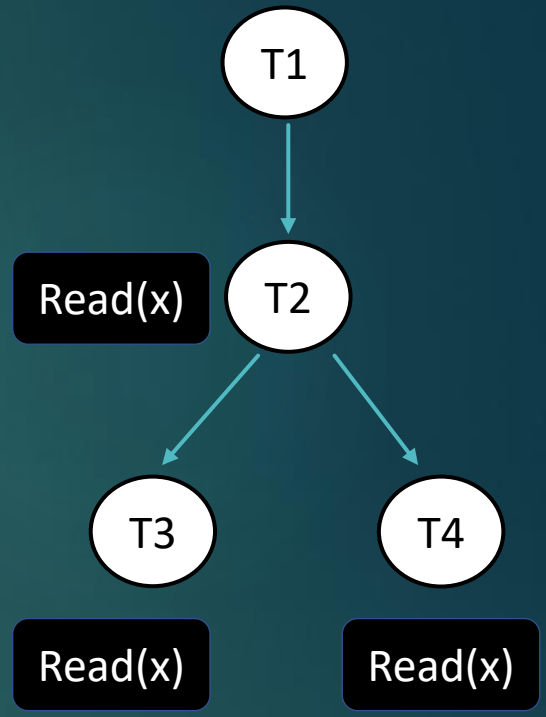


FastTrack: Variable Access

Variable metadata directly proportional to number of parallel accesses

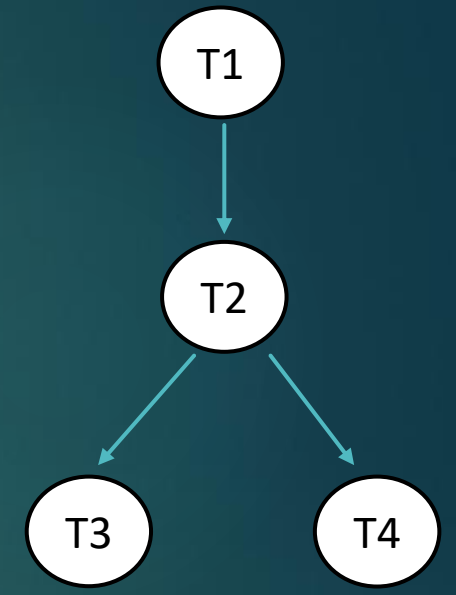
C2@T2, C3@T3, C4@T4

Var x



FastRacer: Variable Access

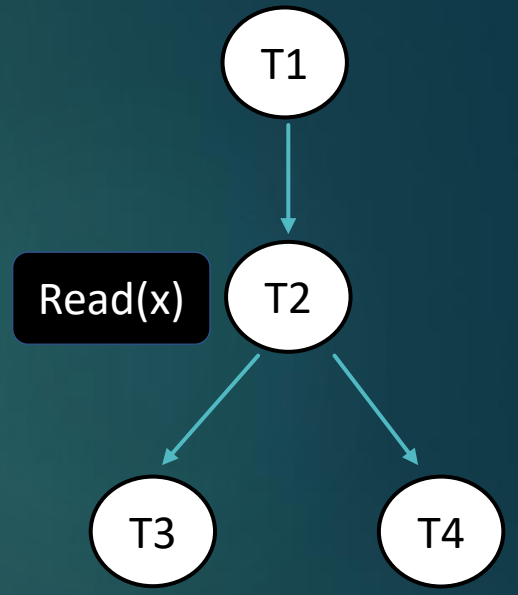
Var x



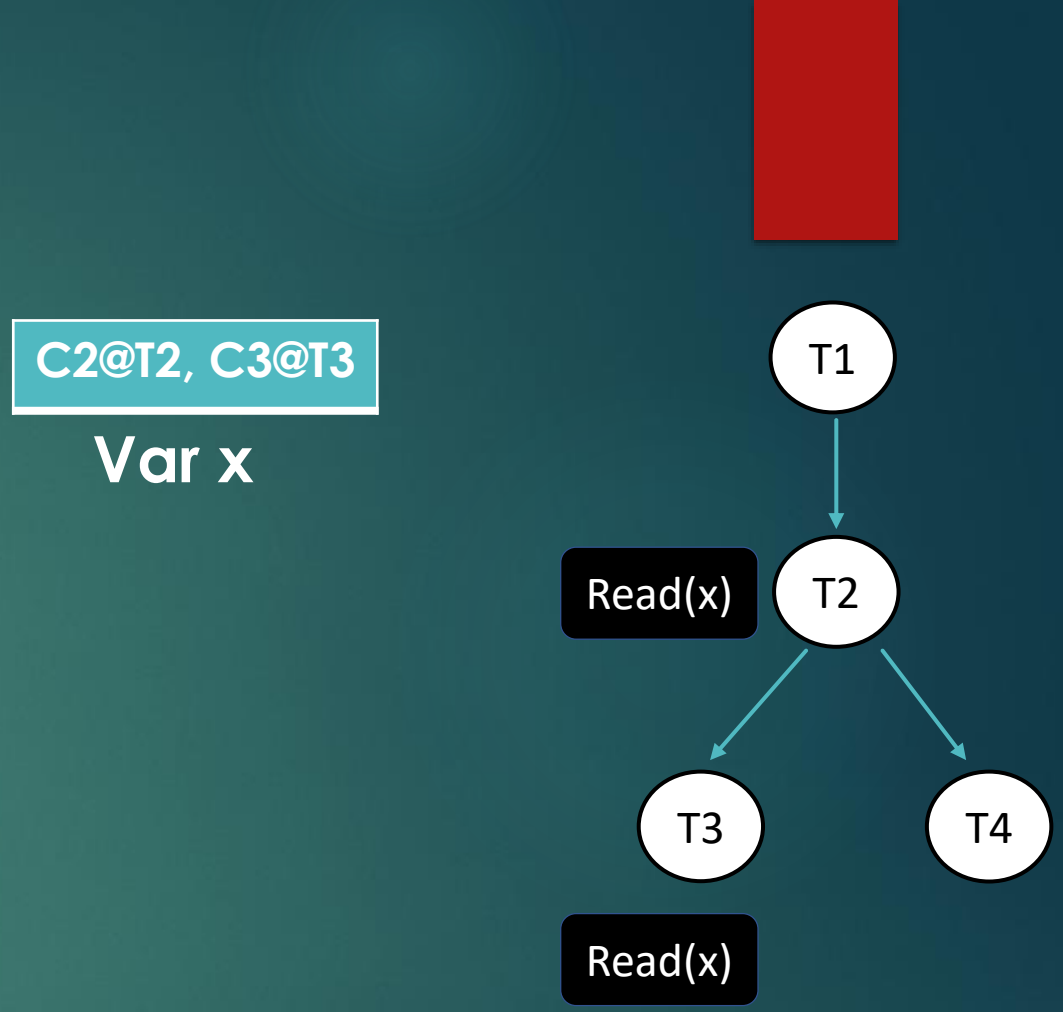
FastRacer: Variable Access

C2@T2

Var x



FastRacer: Variable Access

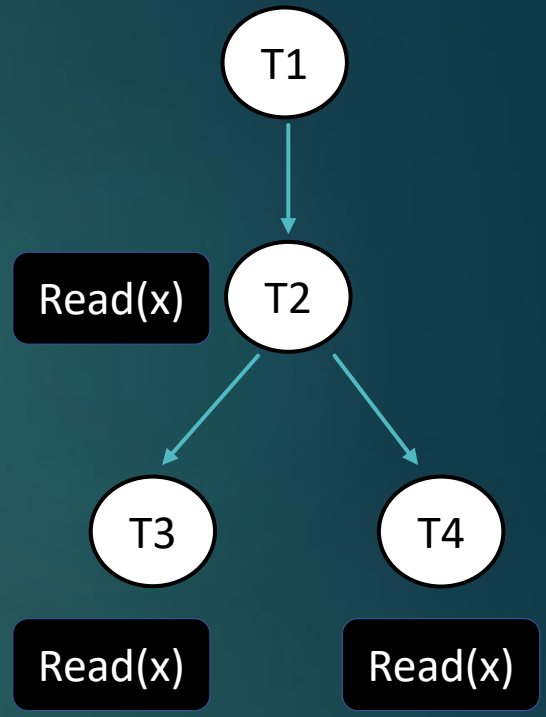


FastRacer: Variable Access

Constant variable metadata for a given lockset

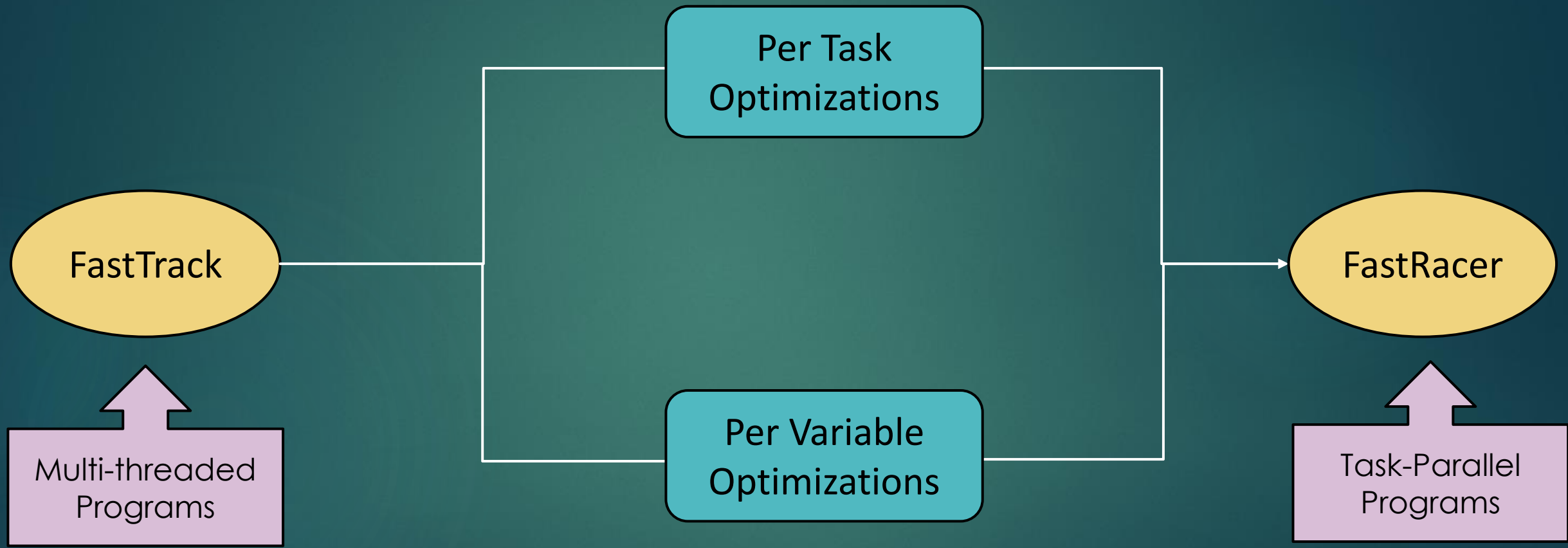
C2@T2, C3@T3

Var x

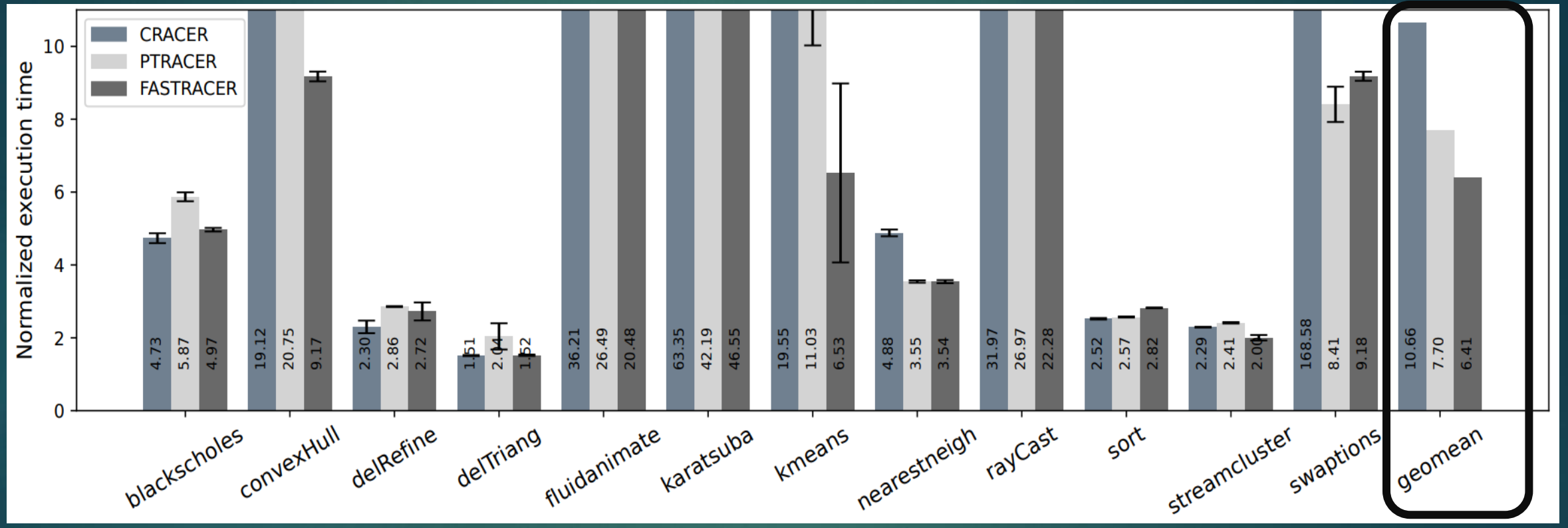


TIP: Select the two accesses with highest LCA in inheritance tree

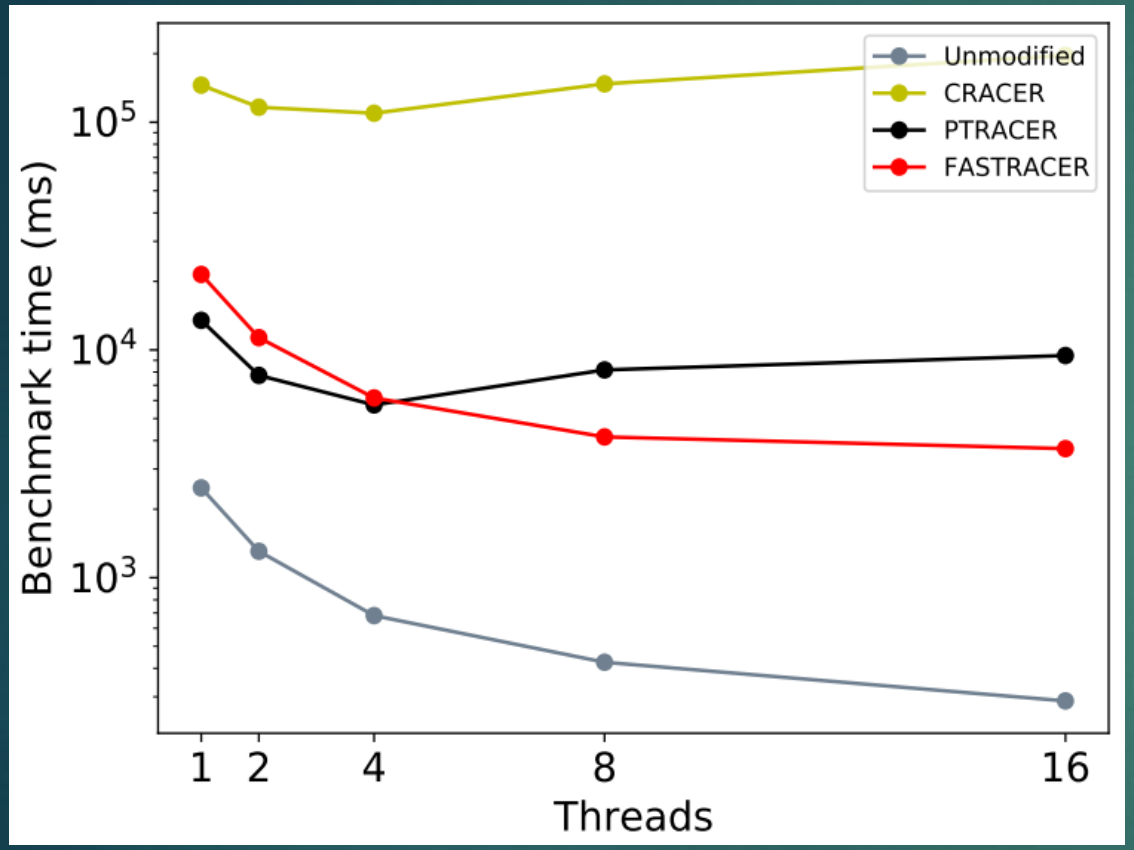
FastRacer: An Efficient Dynamic Data Race Detector



Performance Results



Scalability Plots and Race Reports



Scalability Plots

	# Tasks ($\times 10^3$)	ACC ($\times 10^6$)		Data Races		
		RDs	WRs	CR	PT	FR
blackscholes	0.20	90	50	21	21	21
fluidanimate-r	1.60	26	0.7	40	40	40
streamcluster-r	180	363	13	80	80	80
swaptions	960	77	77	0	0	0
convexHull	8.50	30	0	0	0	0
delRefine	1000	15	0	0	0	0
delTriang	790	30	20	0	0	0
nearestNeigh	2800	51	8	0	0	0
rayCast	1900	160	0	0	0	0
karatsuba	1.98	3.4	0.8	0	0	0
kmeans-r	35	570	10	75	75	75
sort	0.70	11	0.06	1024	1024	1024

Race Reports

Contributions

First to show viability of using vector clocks for efficient dynamic analysis of task-based programs

Optimizations discussed are generic enough for several popular frameworks e.g. OpenMP

Publicly available implementations of FastRacer* and related techniques

* https://github.com/prosper/fastracer-pmam-2022

Efficient Data Race Detection of Async-Finish Programs Using Vector Clocks

Shivam Kumar
IIT Kanpur

Anupam Agrawal
IIT Kanpur

Swarnendu Biswas
IIT Kanpur

Programming Models and Applications for Multicores and Manycores (PMAM 2022)