Increasing Concurrency in Deterministic Runtimes with Conversion

Timothy Merrifield and Jakob Eriksson (UIC)

Why Determinism?

- Multi-core processors are the new normal

- Better performance will hinge on making use of these cores

Yet…shared memory programming is hard!


races


deadlock

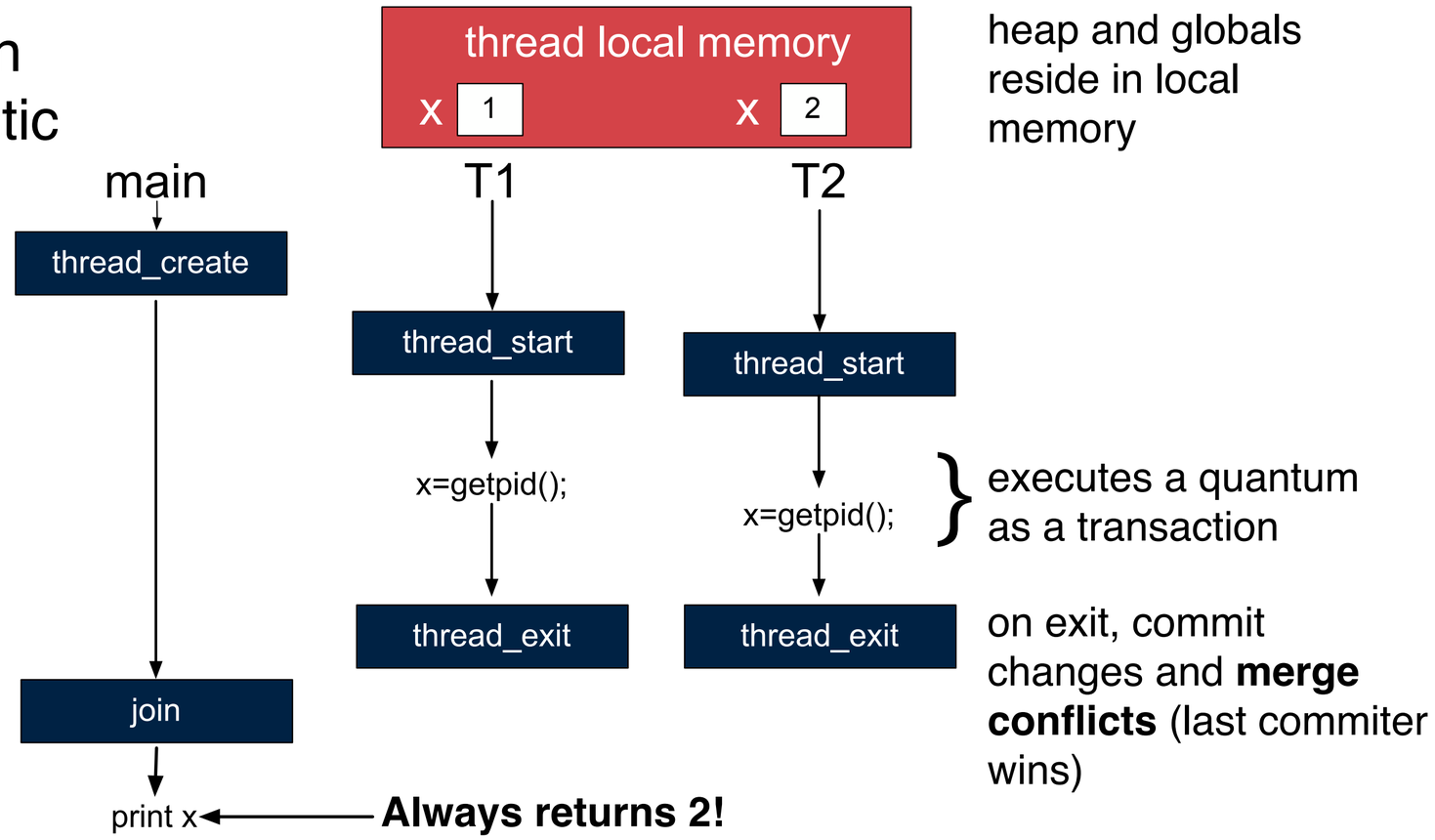

atomicity violations

A Solution: Enforce Determinism

- For a given input, a program written with pthreads/OpenMP is deterministic

An example program

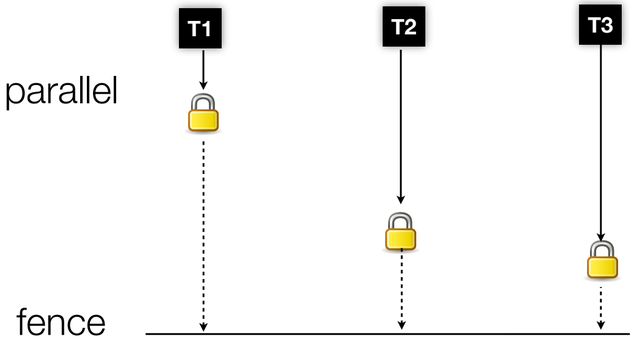
```
int x=0;
run(){
    x=get_pid();
}
main(){
    for (int i=0;i<2;i++)
        thread_create(run);
    join();
    print x;
}
```

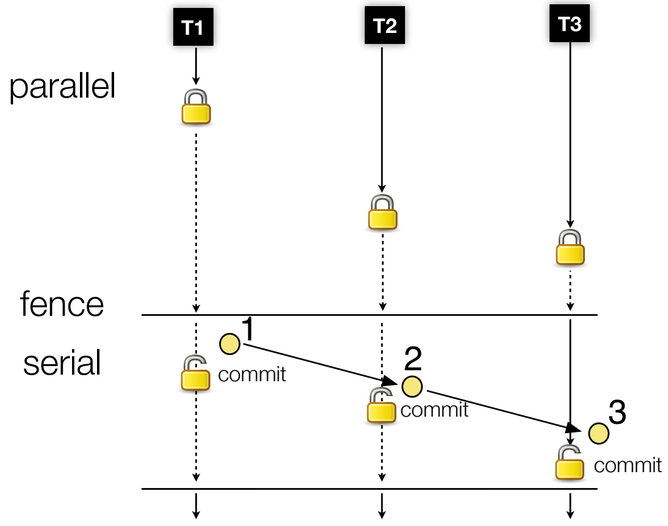


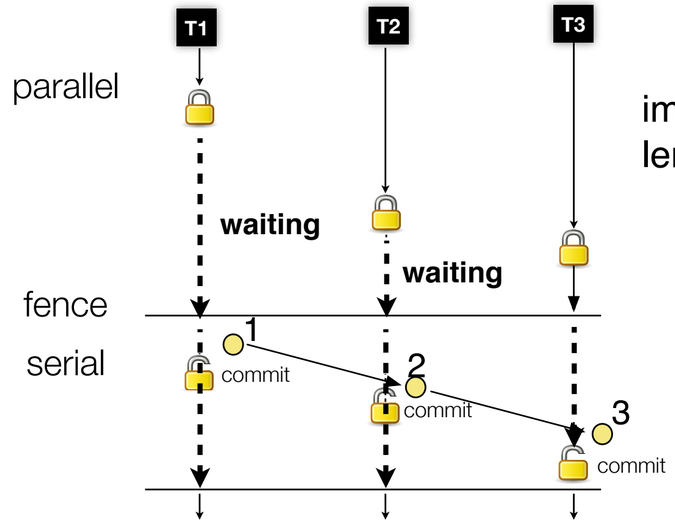
UIC UNIVERSITY OF ILLINOIS AT CHICAGO
Bits Networked Systems Laboratory
COLLEGE OF ENGINEERING

Dthreads* : The State of the Art in Software Determinism

- Dthreads delimits quanta with pthreads operations
  - pthread_mutex_lock, pthread_create, pthread_barrier_wait, etc...
- At the end of the quantum, thread wait at the fence for the arrival of other threads.
- Communication between threads is done using a token
  - once a thread owns the token, it can acquire a lock or commit its changes
  - the token is passed round-robin
- Dthreads is susceptible to long waits if workloads have imbalanced quanta lengths



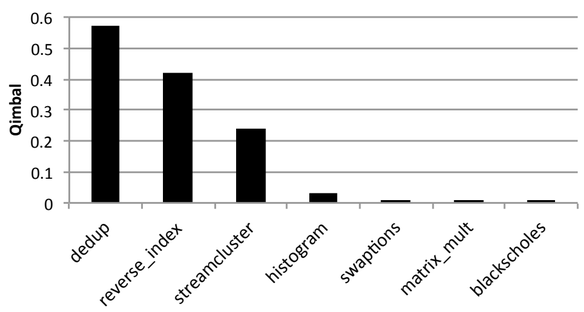


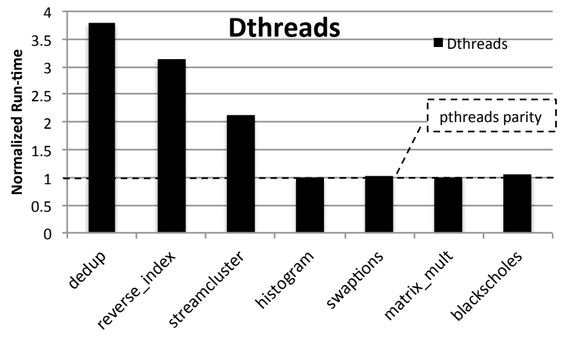


imbalance in quanta length leads to waiting

Quanta Imbalance and Runtime Performance

$$Qimbal_r = \frac{1}{N-1} \sum_{q=1}^{N} \frac{len_{max} - len_q}{len_{max}}$$

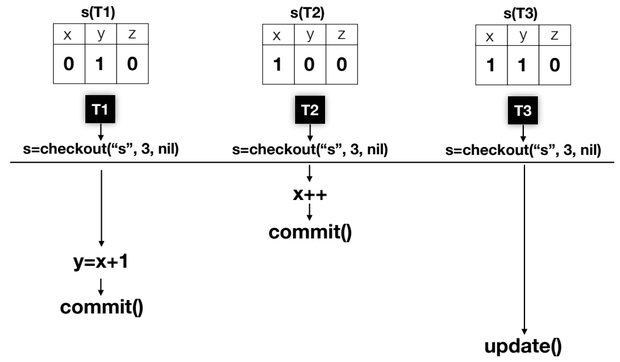




*Tongping Liu, Charlie Curtsinger, and Emery D. Berger. 2011. Dthreads: efficient deterministic multithreading. In (SOSP '11)

Conversion* : A Memory Model for Determinism

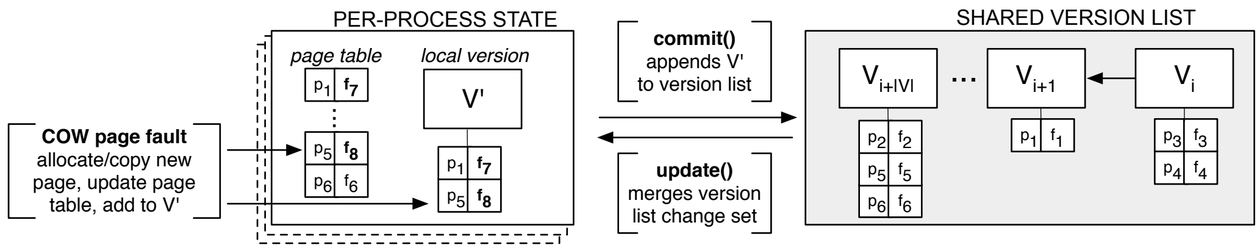
- Version-controlled memory (CVS/SVN-like)*
- Kernel support for Multi-versioning of shared memory segments
- Linux kernel module
  - a few small changes needed to the kernel



Conversion API

s=checkout(name, size, flags)	Create a new Conversion segment, or map in an existing one (anonymous or file-backed)
commit(s)	Commits your local copy's changes to the repository
update(s)	Update your local copy from the repository

Conversion Architecture



Micro-benchmarks

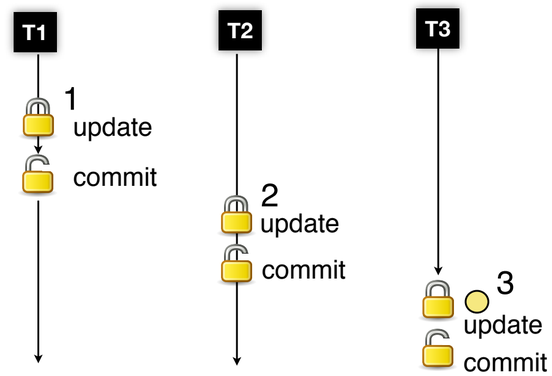
Operation	latency
COW fault w/o CONVERSION	$2.4\ \mu s$
COW fault w/ CONVERSION	$2.5\ \mu s$
`commit()`	$3-6\mu s+0.8\ \mu s*pages$
persistent `commit()`	$3-6\mu s+1.2\ \mu s*pages$
`update()`	$3-6\mu s+0.4\ \mu s*pages$
`update()` w/ merging	$3-6\mu s+5.2\ \mu s*pages$

Using Conversion in Dthreads

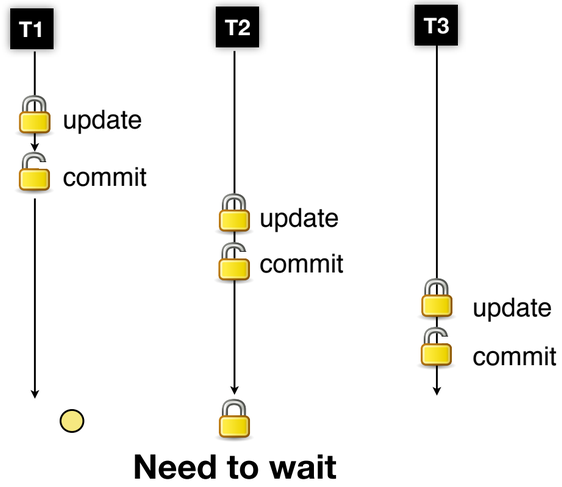
Benefits of Conversion

- Writes to shared memory in parallel with thread execution
  - If you own the token, just commit your changes
  - Rely on the token to maintain determinism
- Benefits of residing in the kernel
  - Faster page fault handling
  - Perform operations in bulk
- Simpler (and more intuitive) code

No more parallel/serial phases
Threads commit immediately



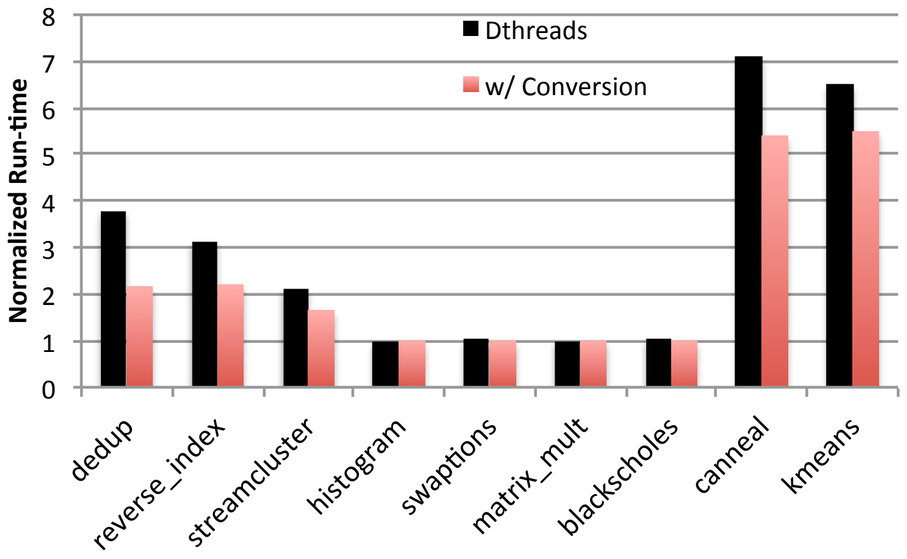
May still need to wait for token



*Timothy Merrifield and Jakob Eriksson. 2013.
Conversion: Multi-version Concurrency Control for Main Memory Segments. In (EuroSys '13)

Performance with Conversion

- Conversion helps programs with imbalance
  - dedup and reverse index most notably
- Other programs benefit because of Conversion performance
  - faster page faults and efficient page table operations
  - canneal and kmeans

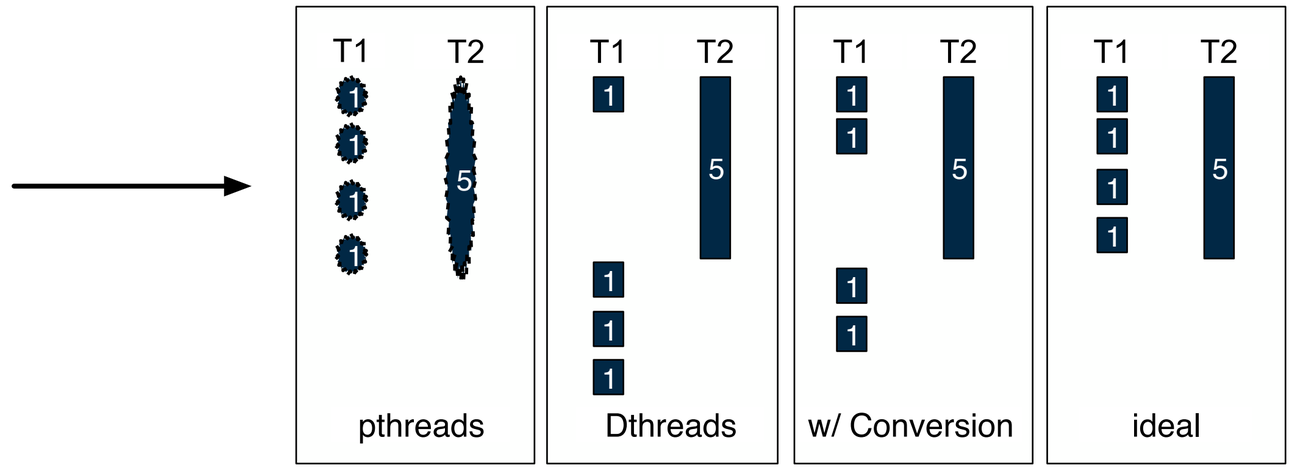


benchmark	token (ms)	fence (ms)	commit (ms)	trans. count	page faults
`dedup`	6.3K	17K	251	91K	359K
w/ CONVERSION	10K	0	1.2K	”	434K
`canneal`	49K	25K	14K	2.1K	3.6M
w/ CONVERSION	63K	0	24K	”	3.6M
`reverse_index`	2.8K	5.5K	125	76K	132K
w/ CONVERSION	3.6K	0	415	”	152K
kmeans	17K	112K	127	8.3K	49K
w/ CONVERSION	22K	8.7K	120	”	55K
streamcluster	5.5K	11K	409	260K	135K
w/ CONVERSION	7.8K	0	3.0K	”	137K

Future work

Determinism

- Token acquisition based on time (not round-robin)
- Kendo-style deterministic clock
- Each thread maintains their own lock
  - Uses retired instruction count
- Thread with lowest clock time holds the token



Conversion

- Overall performance improvements
- Pulling work off the critical path and performing updates/commits in the background
- More intelligent garbage collection of unused versions
- NUMA-aware Conversion

Other Conversion Applications

- Applications that can work with a (slightly) out-of-date local copy
- Concurrent Data Structures
  - Snapshot isolation for long running readers
- Concurrent Garbage Collection

https://github.com/tmerrifi/conversion

UIC UNIVERSITY OF ILLINOIS AT CHICAGO
Bits Networked Systems Laboratory
COLLEGE OF ENGINEERING