Tackling The Challenges of Big Data
Big Data Systems

Nickolai Zeldovich

Associate Professor

Massachusetts Institute of Technology

Tackling The Challenges of Big Data
Big Data Systems
Multicore Scalability
Introduction

Nickolai Zeldovich

Associate Professor

Massachusetts Institute of Technology

Goal: Scalability

- **With n cores, do n times as much work**
- **One way to think about multi-core performance**
- **Limit to scalability: serial sections**

- **Parallel time: P**
- **Serial time: S**

Sharing vs. Partitioning

- **One approach to multi-core systems: partitioning**
- **Advantage: avoids sharing between partitions**
- **Disadvantage: may be hard to choose a partitioning**
- **Disadvantage: may require load-balancing**

Outline

- **Hardware: cache coherence**
- **Cache coherence and scalability**
  - Example: implementing a lock
  - Avoiding performance collapse
- **Improving scalability by avoiding locks**
  - Example: lock-free reads in a stack
  - Generalizing lock-free reads with RCU

Tackling The Challenges of Big Data

Big Data Systems

Multicore Scalability

Introduction

THANK YOU

PROFESSIONAL EDUCATION

CSAIL

Tackling The Challenges of Big Data

Module: Big Data Systems

Nickolai Zeldovich

Associate Professor

Massachusetts Institute of Technology

PROFESSIONAL EDUCATION

CSAIL

Tackling The Challenges of Big Data

Module: Big Data Systems

Topic: Multicore Scalability

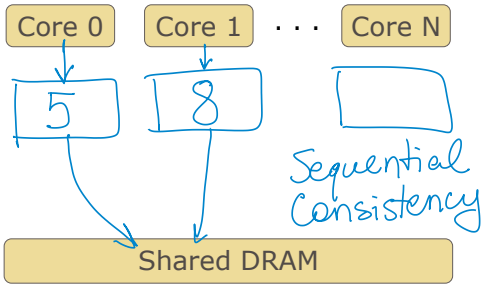
Cache coherence

Nickolai Zeldovich

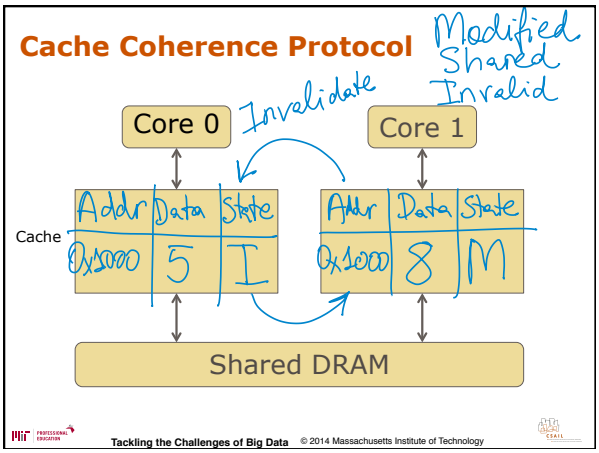
Associate Professor

Massachusetts Institute of Technology

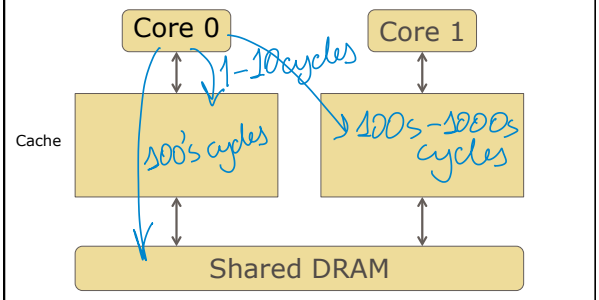
Multi-Core Hardware



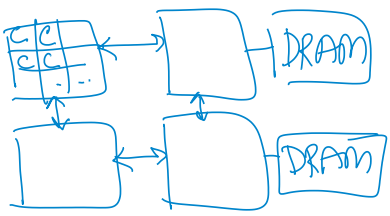
Cache Coherence Protocol



Performance With Cache Coherence



Multi-core hardware in practice



Tackling The Challenges of Big Data

Big Data Systems

Multicore Scalability

Cache coherence

THANK YOU

Tackling The Challenges of Big Data

Big Data Systems

Nickolai Zeldovich

Associate Professor

Massachusetts Institute of Technology

Tackling The Challenges of Big Data

Big Data Systems

Multicore Scalability

Implementing a lock

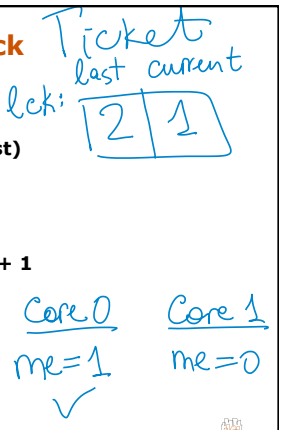
Nickolai Zeldovich

Associate Professor

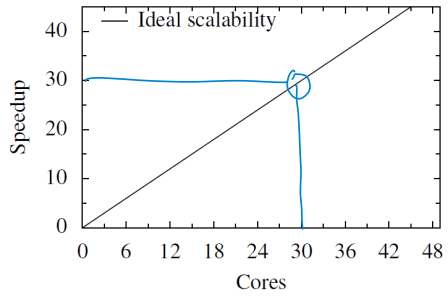
Massachusetts Institute of Technology

Implementing A Lock

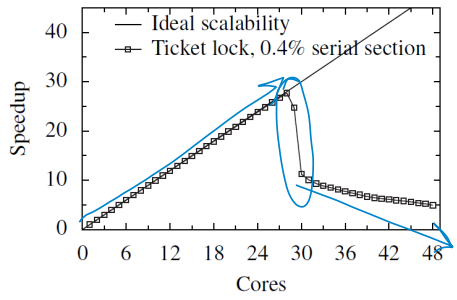
- **acquire(lck):**
  me = atomic_inc(&lck.last)
  while lck.current != me:
  pass /* spin */
- **release(lck):**
  lck.current = lck.current + 1



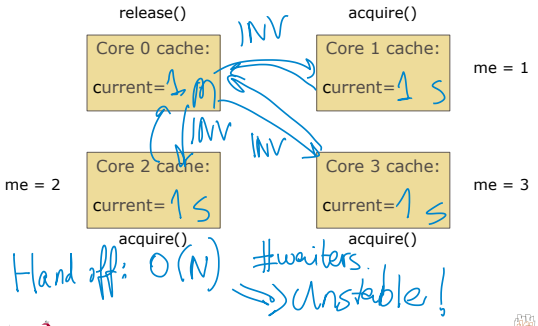
Performance Of Ticket Locks



Performance Of Ticket Locks



Why Do Ticket Locks Collapse?



Tackling The Challenges of Big Data

Big Data Systems

Multicore Scalability

Implementing a lock

THANK YOU

PROFESSIONAL EDUCATION

CSAIL

Tackling The Challenges of Big Data

Big Data Systems

Nickolai Zeldovich

Associate Professor

Massachusetts Institute of Technology

PROFESSIONAL EDUCATION

CSAIL

Tackling The Challenges of Big Data

Big Data Systems

Multicore Scalability

Non-collapsing locks

Nickolai Zeldovich

Associate Professor

Massachusetts Institute of Technology

Avoiding Lock Collapse

- **What's the ideal performance for a contended lock?**
  - Constant hand-off time: no collapse
  - Total performance at best remains flat: still have a serial section, so no scalability

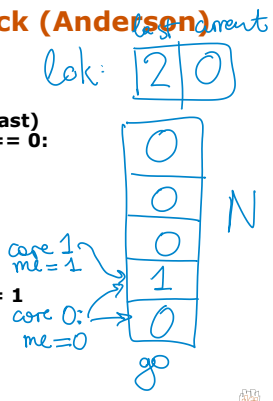
Non-Collapsing Lock (Anderson)

- **acquire(lck):**

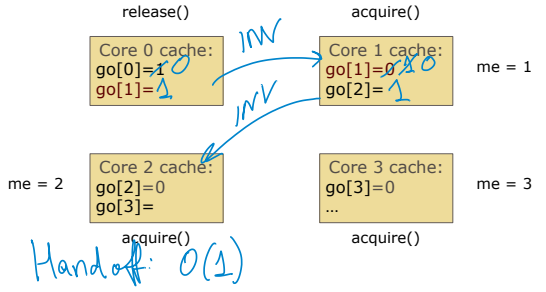
```
me = atomic_inc(&lck.last)
while lck.go[me % N] == 0:
  pass  /* spin */
lck.current = me
```

- **release(lck):**

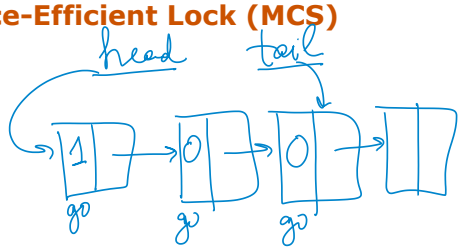
```
me = lck.current
lck.go[me % N] = 0
lck.go[(me + 1) % N] = 1
```



Non-Collapsing Lock: Cache Coherence



Space-Efficient Lock (MCS)



Non-Collapsing Locks

- **Helps avoid collapse under contention**
- **Does not improve scalability: serial section remains**
- **To improve scalability, eliminate contention**
  - Fine-grained locks
  - Avoid locks altogether

Tackling The Challenges of Big Data
Big Data Systems
Multicore Scalability
Non-collapsing locks

THANK YOU

Tackling The Challenges of Big Data
Big Data Systems

Nickolai Zeldovich

Associate Professor

Massachusetts Institute of Technology

Tackling The Challenges of Big Data
Big Data Systems
Multicore Scalability
Lock-free synchronization

Nickolai Zeldovich

Associate Professor

Massachusetts Institute of Technology

Example Data Structure: Stack

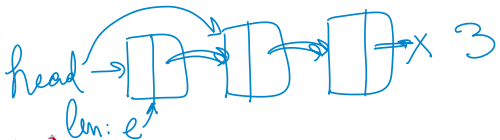
- length():

```
e = head
while e != NULL:
  i = i + 1
  e = e.next

return i
```

- pop():

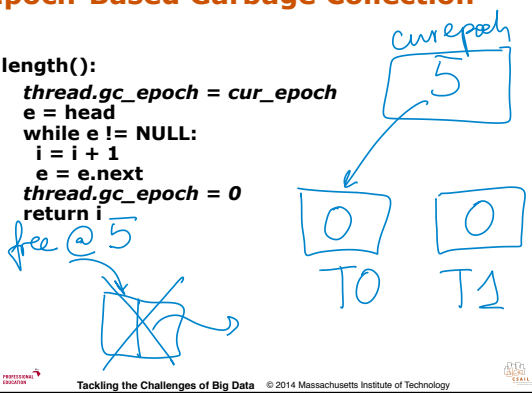
```
e = head
head = e.next
free(e)
```



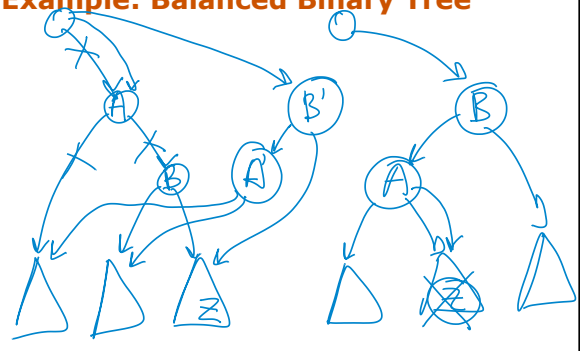
Epoch-Based Garbage Collection

- length():

```
thread.gc_epoch = cur_epoch
e = head
while e != NULL:
  i = i + 1
  e = e.next
thread.gc_epoch = 0
return i
```



Example: Balanced Binary Tree



RCU Enables Scalable Readers

- **Three rules:**
  - Lock-free readers
  - Single pointer update in writers (update a copy)
  - Delayed free
- **Readers do not modify any shared cache lines**
- **Concurrent readers can execute in private caches**
  - No cache-coherence traffic!
- **Good scalability for read-heavy workloads**

Conclusion: Multicore Scalability

- **Cache-coherence protocols in hardware**
- **Lock can collapse due to sharing one cache line**
- **Multiple cache lines help avoid collapse due to a lock**
- **RCU: powerful technique for read-heavy workloads**

Tackling The Challenges of Big Data

Big Data Systems

Multicore Scalability

Lock-free synchronization

THANK YOU

PROFESSIONAL EDUCATION

CSAIL

Tackling The Challenges of Big Data

Big Data Systems

Multicore Scalability

THANK YOU

Nickolai Zeldovich

Associate Professor

Massachusetts Institute of Technology