A Lightweight Log-based Deferred Update for Linux Kernel Scalability

Joohyun Kyong

School of Computer Science
Kookmin University
Thesis advisor: Sung-Soo Lim

November 5, 2016

CPU trends

- 1

CPU trends

- 2005

CPU trends

- power

CPU trends

CPU trends

Performance Scalability

Performance Scalability

OS Kernel Scalability

OS Kernel Scalability

OS Kernel Scalability

OS Kernel Scalability

OS Kernel Scalability Histroy

OS Kernel Scalability Histroy

OS Kernel Scalability Histroy

OS Kernel Scalability Histroy

OS Kernel Scalability Histroy

Update-heavy Data structure

Update-heavy Data structure

Non-blocking Data Structure

Non-blocking Data Structure

Non-blocking Data Structure

Non-blocking Data Structure

Cache communication bottlenect

Cache communication bottlenect

Cache communication bottlenect

Cache communication bottlenect

Cache communication bottlenect

Approach: Log-based Concurrent Update

Approach: Log-based Concurrent Update

Approach: Log-based Concurrent Update

Advantages of eliminating time-stamp counters

Contributions

- Background of research
- Our new method and Evaluation
- Future plans and Summary

Outline

- Design
  - Approach
  - Example
- Applying the Linux kernel
- Implementation
- Evaluation

Design

Why the OpLog needs the time-stamp counter?

Log Example

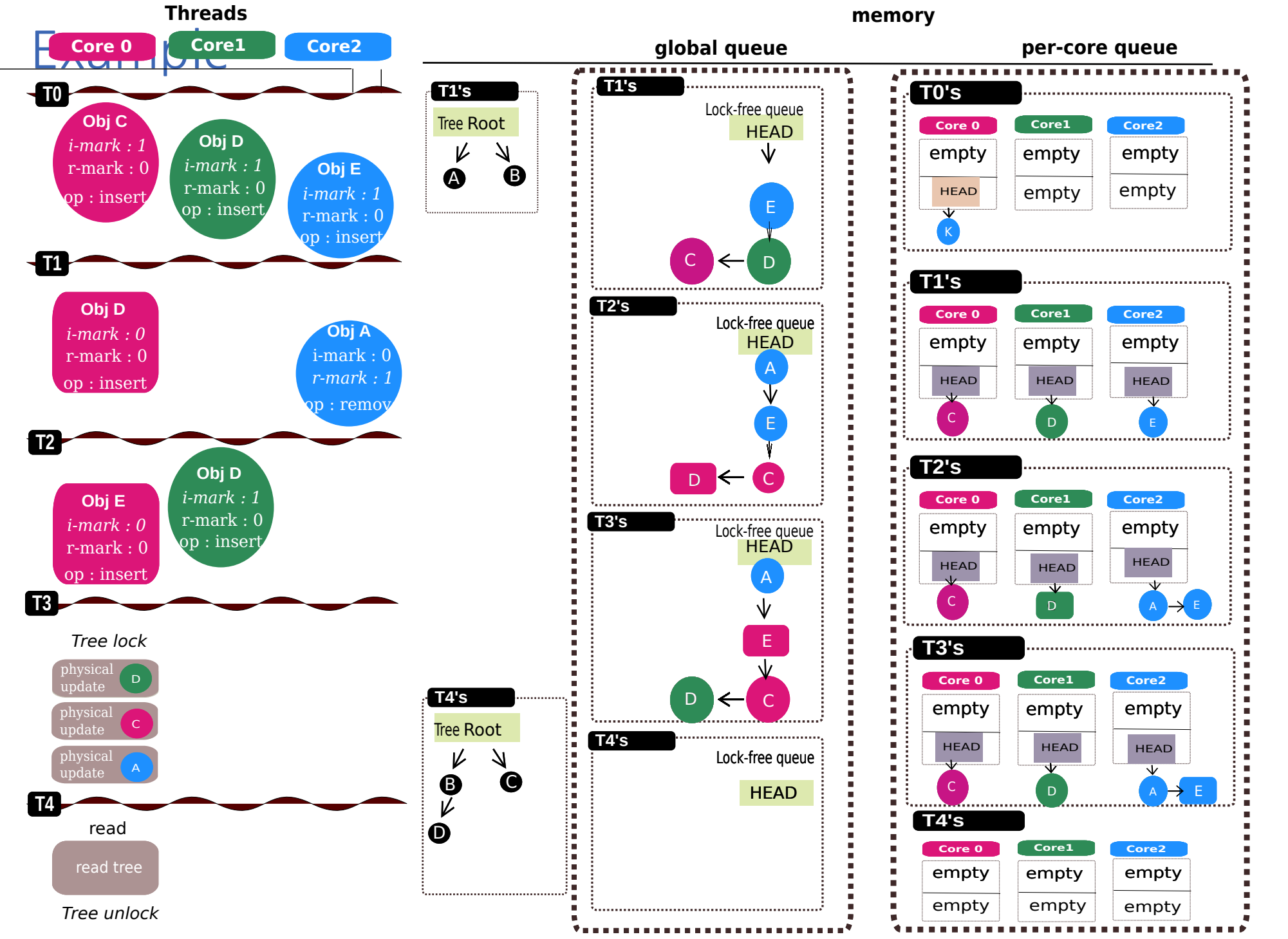
Update-side removing

Garbage log

Reusing logs

Approach

Example


Threads
memory
global queue
per-core queue
Example
Core 0
Core1
Core2
T0
Obj C
i-mark : 1
r-mark : 0
op : insert
Obj D
i-mark : 1
r-mark : 0
op : insert
Obj E
i-mark : 1
r-mark : 0
op : insert
T1
Obj D
i-mark : 0
r-mark : 0
op : insert
Obj A
i-mark : 0
r-mark : 1
op : remov
T2
Obj E
i-mark : 0
r-mark : 0
op : insert
Obj D
i-mark : 1
r-mark : 0
op : insert
T3
Tree lock
physical update
D
physical update
C
physical update
A
T4
read
read tree
Tree unlock
T1's
Tree Root
A
B
T4's
Tree Root
B
C
D
T1's
Lock-free queue
HEAD
E
C
D
T2's
Lock-free queue
HEAD
A
E
D
C
T3's
Lock-free queue
HEAD
A
E
D
C
T4's
Lock-free queue
HEAD
T0's
Core 0
Core1
Core2
empty
HEAD
K
T1's
C
D
E
T2's
C
D
A
E
T3's
C
D
A
E
T4's

Example

Example

Example

Example

Example

Example

Example

Applying the Linux kernel

anonymous reverse mapping

anonymous reverse mapping

file mapping

file mapping

Evaluation

Non-blocking algorithm – Harris linked list

Test-bed

Test-bed

AIM7

AIM7 – CPU utilization

EXIM

EXIM – CPU utilization

Lmbench

Lmbench – CPU utilization

Update ratio

Related work

Papers

Conclusion

- Background of research
- LDU method and Evaluation
- Future plans and Summary

Conclusion

- Background of research
- LDU method and Evaluation
- Future plans and Summary
- https://github.com/KMU-embedded/scalablelinux