LDU: A Lightweight Concurrent Update Method with Deferred Processing for Linux Kernel Scalability

Joohyun Kyong and Sung-Soo Lim

School of Computer Science
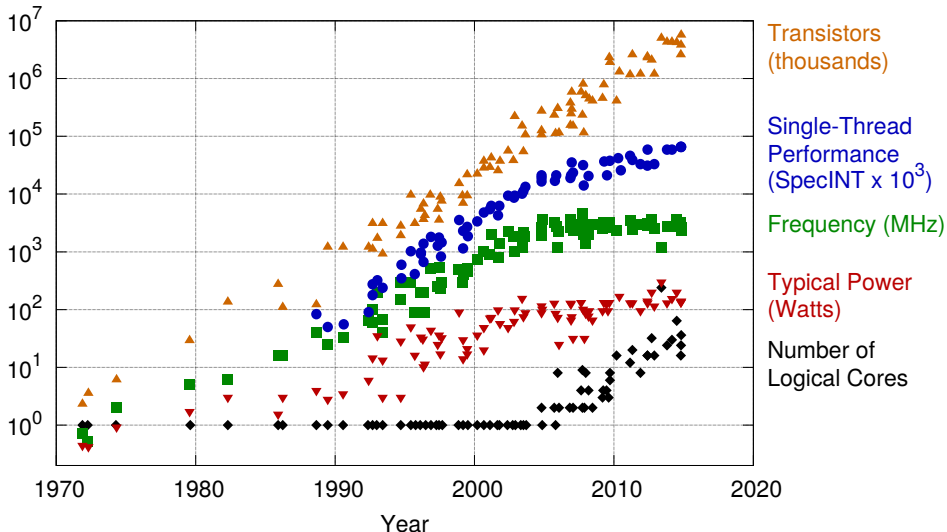Kookmin University

March 3, 2016

Outline

- Background of research and History of the Linux scalability
- Linux Scalability Problems
- Our new method and Evaluation
- Future plans and Summary

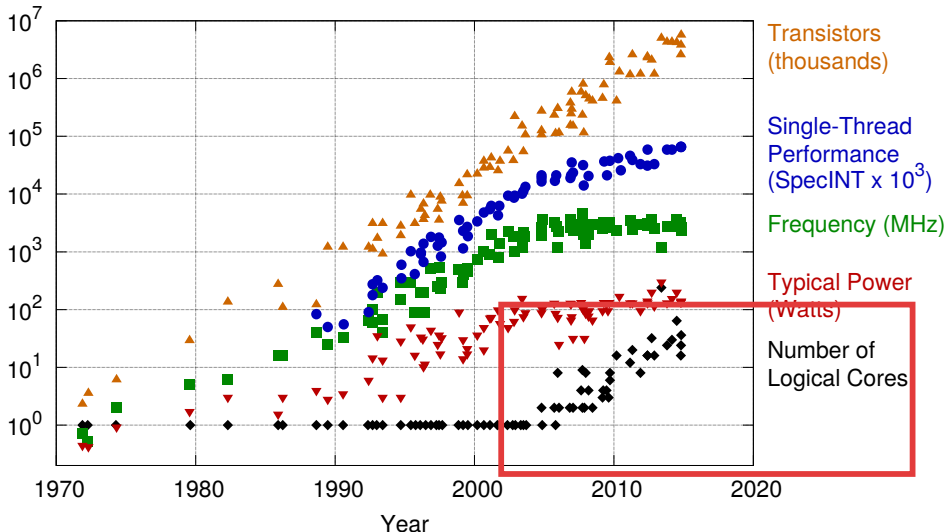# 40 Years of Microprocessor Trend Data

40 Years of Microprocessor Trend Data

Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

40 Years of Microprocessor Trend Data

40 Years of Microprocessor Trend Data

Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

History of the Linux scalability research

Boyd-Wickizer, Silas, et al. "An Analysis of Linux Scalability to Many Cores." OSDI, 2010.

History of the Linux scalability research

Boyd-Wickizer, Silas, et al. "An Analysis of Linux Scalability to Many Cores." OSDI, 2010.

Austin T. Clements., M. Frans Kaashoek, and Nickolai Zeldovich. "Scalable address spaces using RCU balanced trees." ASPLOS, 2012.

Austin T. Clements, M. Frans Kaashoek, Nickolai Zeldovich, Robert T. Morris, Eddie Kohler.
"The Scalable Commutativity Rule: Designing Scalable Software for Multicore Processors," SOSP 2013.

Austin T. Clements, M. Frans Kaashoek, and Nickolai Zeldovich. "RadixVM: Scalable address spaces for multithreaded applications" , EuroSys 2013
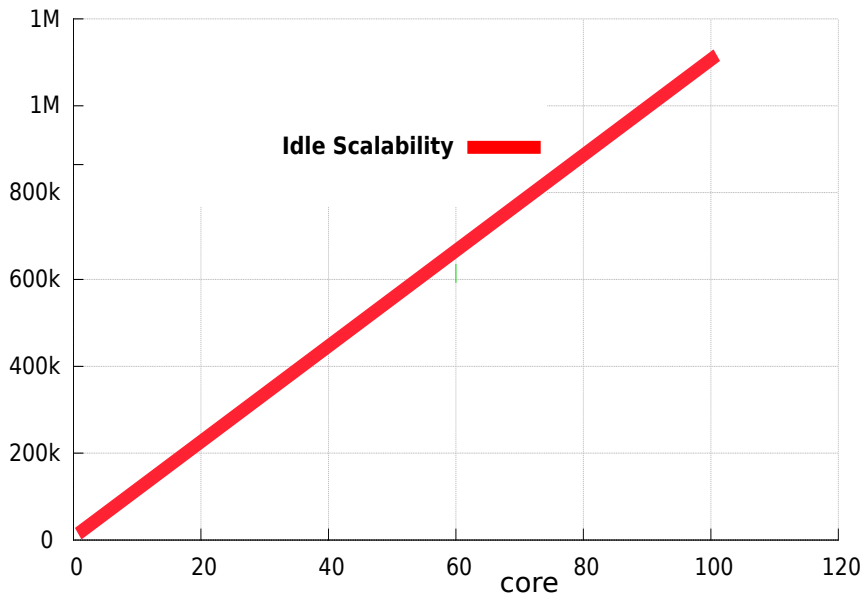
What about update?

Linux Scalability Problems – AIM7– multiuser
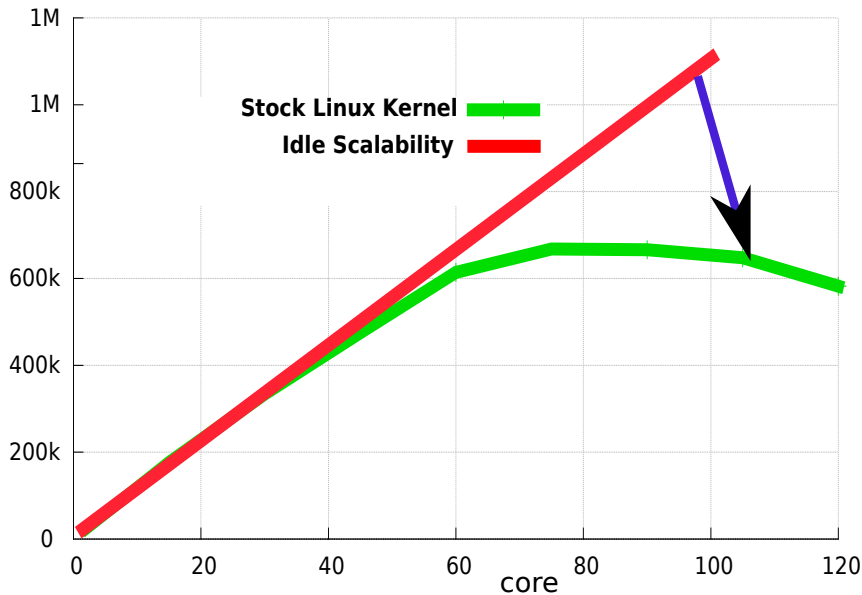

jobs/min

Stock Linux Kernel

Problem after 60 core

core
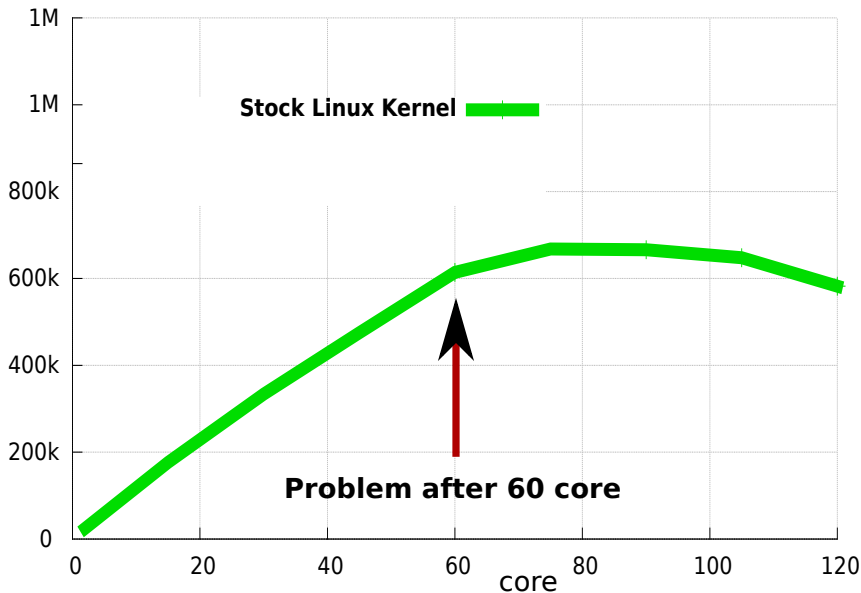
Linux Scalability Problems

jobs/min

Linux Scalability Problems
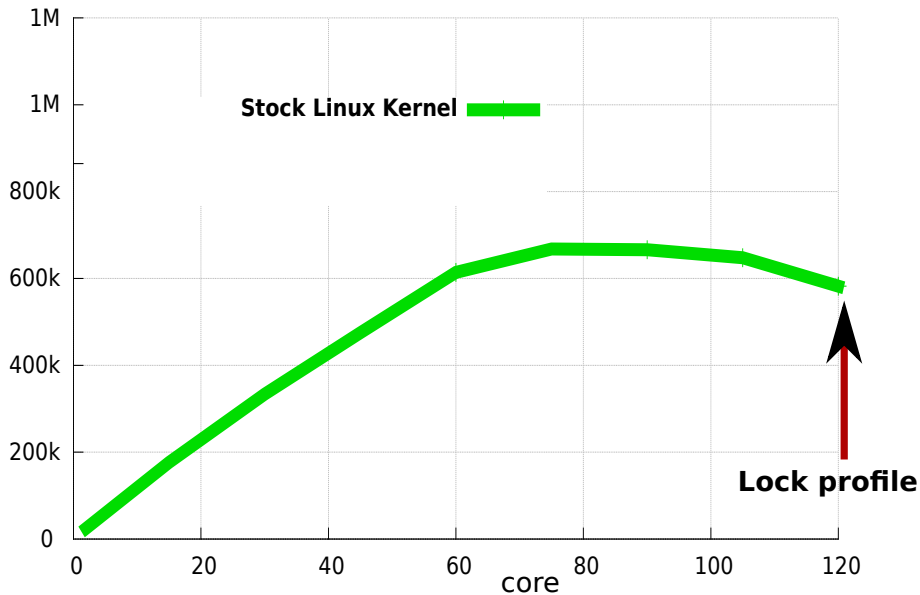
jobs/min

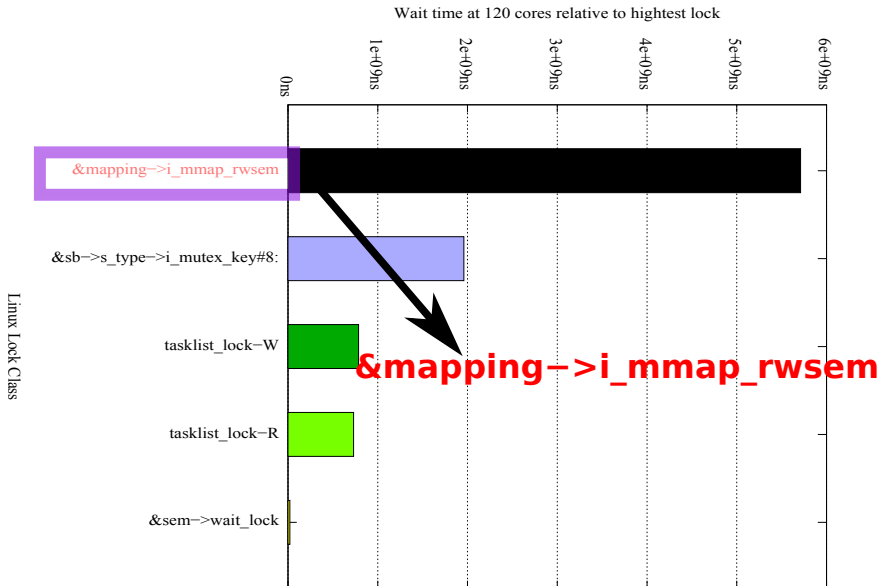**Stock Linux Kernel**
Idle Scalability

core

Linux Scalability Problems

Linux Scalability Problems
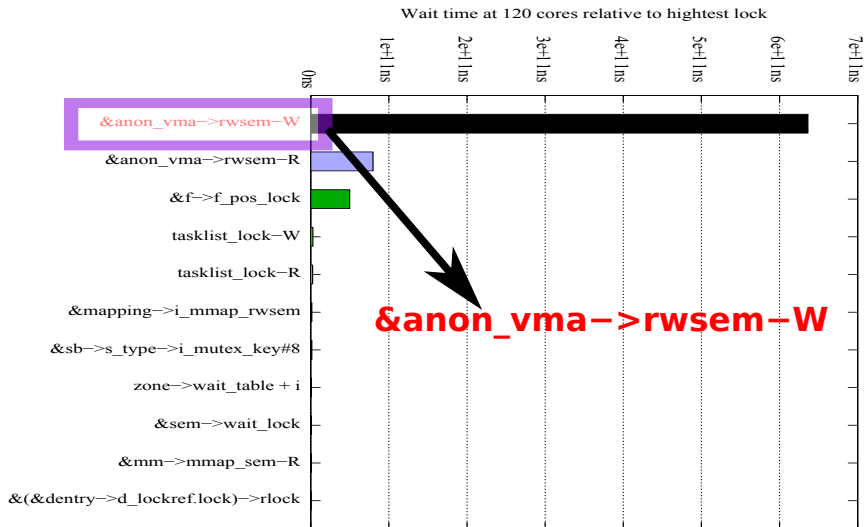
Linux Scalability Problems

Update Serialization

Core 0	spin or block
Core1	spin or block · **Update** · spin or block · **Update Lock**
Core2	**Update Lock** · spin or block
Core3	spin or block · **Update Lock**

Time

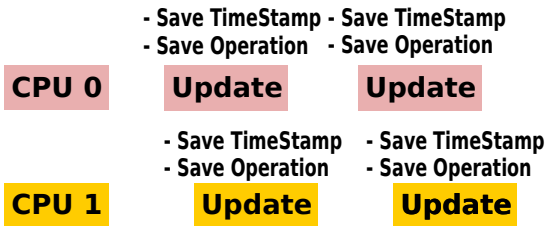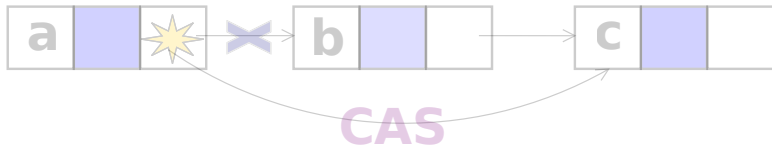**&anon_vma−>rwsem−W**
&mapping−>i_mmap_rwsem-W
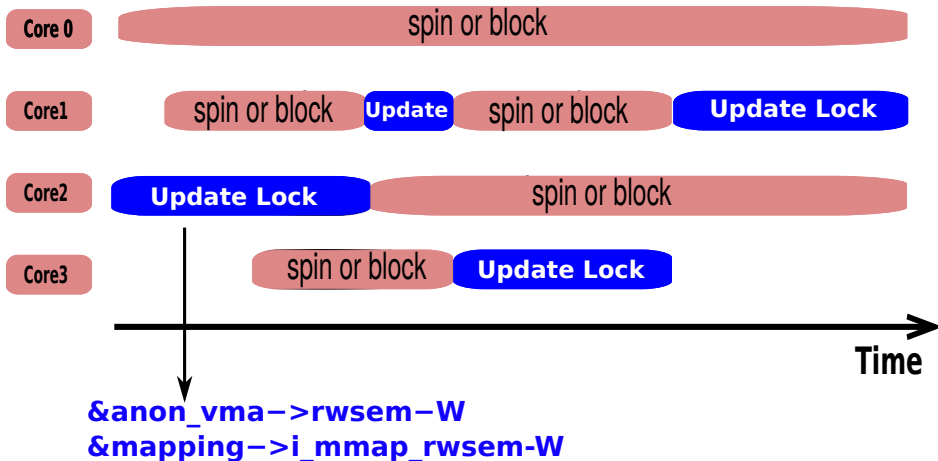
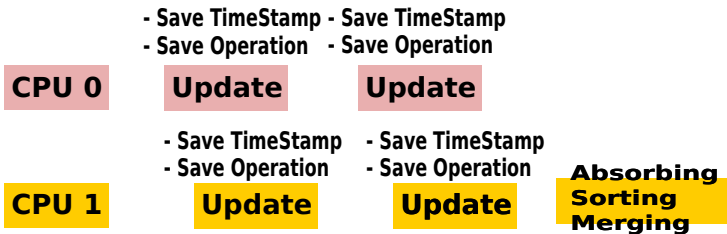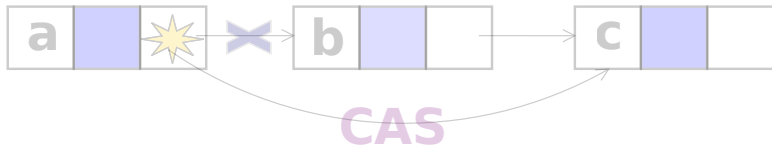# Solution – Non blocking Data Structure

CAS
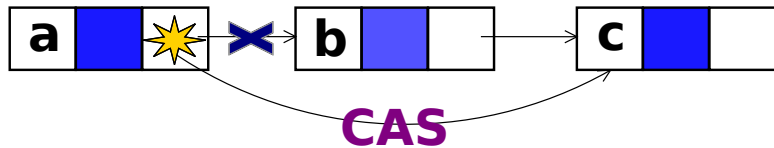
- Harris' linked list : linked list
- Fraser skip list : skip list
- Lock-free tree : binary tree
- Michael's hash table : hash table
- Cliff Click's hash map : hash table
- No hot spot skip list : skip list
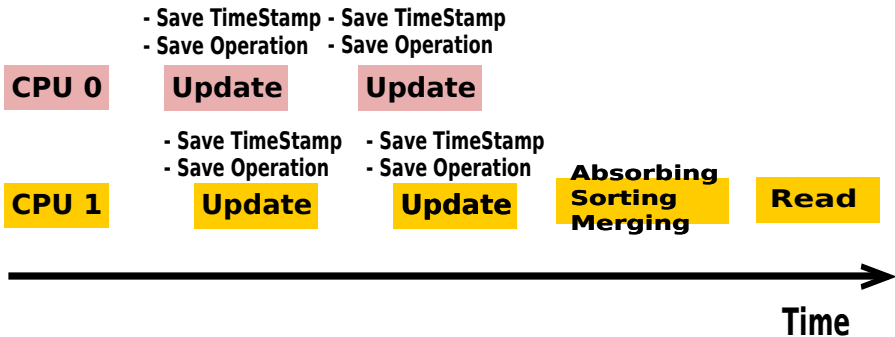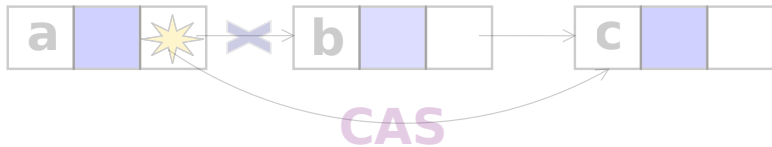- Rotating skip list : skip list
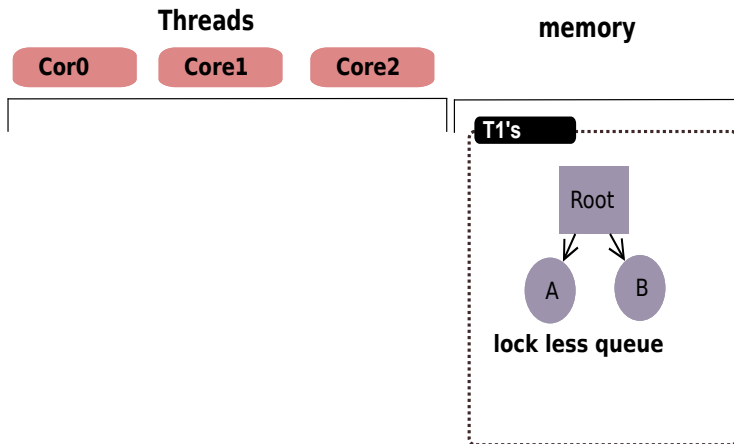
Per-core processing
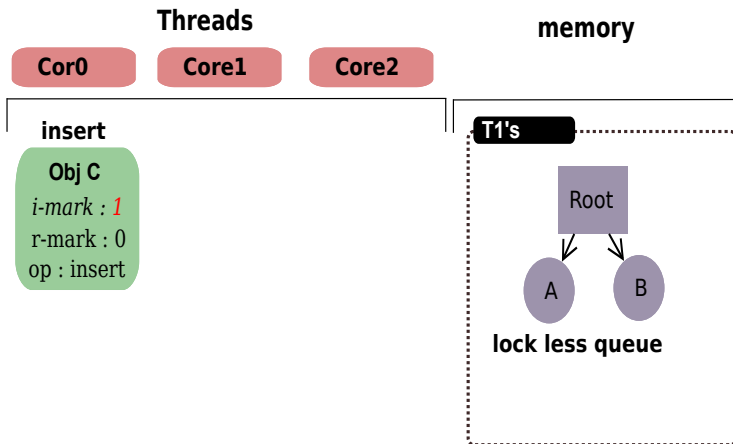
Per-core processing

a ✷ ✕ b c

CAS

- Save TimeStamp   - Save TimeStamp
- Save Operation   - Save Operation

CPU 0   **Update**   **Update**

- Save TimeStamp   - Save TimeStamp
- Save Operation   - Save Operation

CPU 1   **Update**   **Update**   **Absorbing Sorting Merging**

Time

Per-core processing

CAS

	- Save TimeStamp	**- Save TimeStamp**
	- Save Operation	**- Save Operation**
CPU 0	**Update**	**Update**

	- Save TimeStamp	**- Save TimeStamp**	**Absorbing**	
	- Save Operation	**- Save Operation**	**Sorting**	
CPU 1	**Update**	**Update**	**Merging**	**Read**

Time

LDU: An example of Lightweight Deferred Update

LDU: An example of Lightweight Deferred Update

Threads

memory

Cor0　　**Core1**　　**Core2**

insert

Obj C
i-mark : 1
r-mark : 0
op : insert

T1's

Root

A　　B

lock less queue

LDU: An example of Lightweight Deferred Update

LDU: An example of Lightweight Deferred Update

Threads

Cor0   Core1   Core2

insert

Obj C
i-mark : 1
r-mark : 0
op : insert

insert

Obj D
i-mark : 1
r-mark : 0
op : insert

memory

T1's

Root

A   B

lock less queue

C

LDU: An example of Lightweight Deferred Update

Threads

memory

| Cor0 | Core1 | Core2 |

insert

Obj C
i-mark : 1
r-mark : 0
op : insert

insert

Obj D
i-mark : 1
r-mark : 0
op : insert

T1's

Root

A   B

lock less queue

D → C

LDU: An example of Lightweight Deferred Update

LDU: An example of Lightweight Deferred Update

remove

Obj C
i-mark : 1
r-mark : 0
op : insert

remove

Obj C
i-mark : 0
r-mark : 0
op : insert

Update-side Absorbing
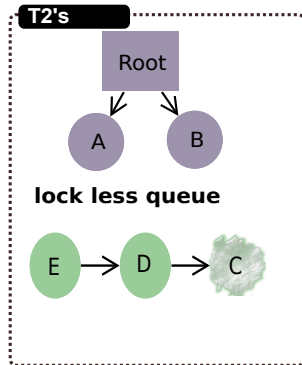
LDU: An example of Lightweight Deferred Update
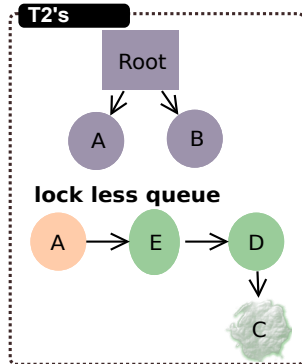
remove

Obj C
i-mark : 0
r-mark : 0
op : insert

remove

Obj A
i-mark : 0
r-mark : 1
op : remove

LDU: An example of Lightweight Deferred Update

remove

Obj C
i-mark : 0
r-mark : 0
op : insert

remove

Obj E
i-mark : 0
r-mark : 0
op : insert

remove

Obj A
i-mark : 0
r-mark : 1
op : remove

T2's

Root

A   B

lock less queue

A → E → D

C
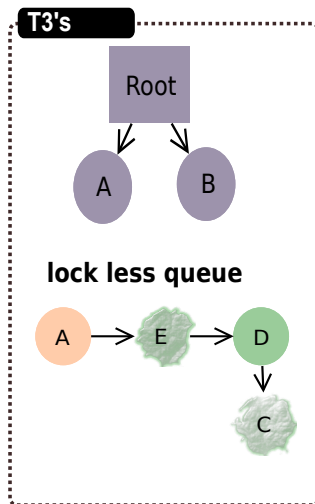
T2

Tree lock

LDU: An example of Lightweight Deferred Update

LDU: An example of Lightweight Deferred Update

Tree lock
synchronize

physical update D

T3's

Root

A   B

D

lock less queue

A → E

LDU: An example of Lightweight Deferred Update

LDU: An example of Lightweight Deferred Update

Tree lock
synchronize

physical update (D)

physical update (A)

T3

read

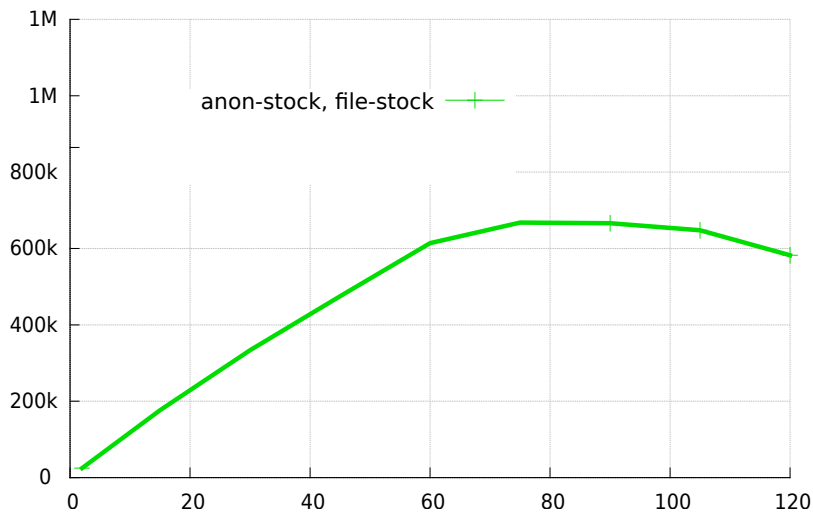read tree

Tree unlock
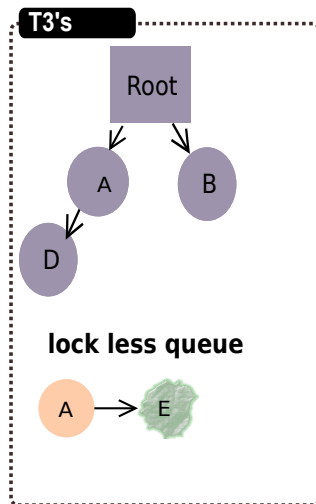
T3's

Root

D   B

lock less queue

NULL
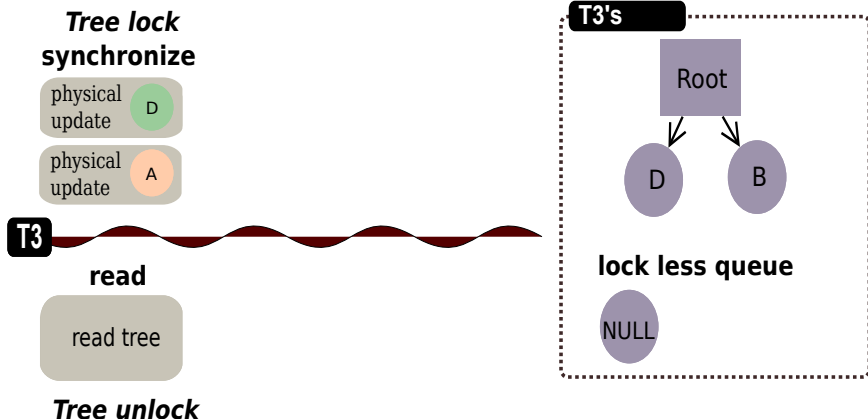
Concurrent update for Linux kernel
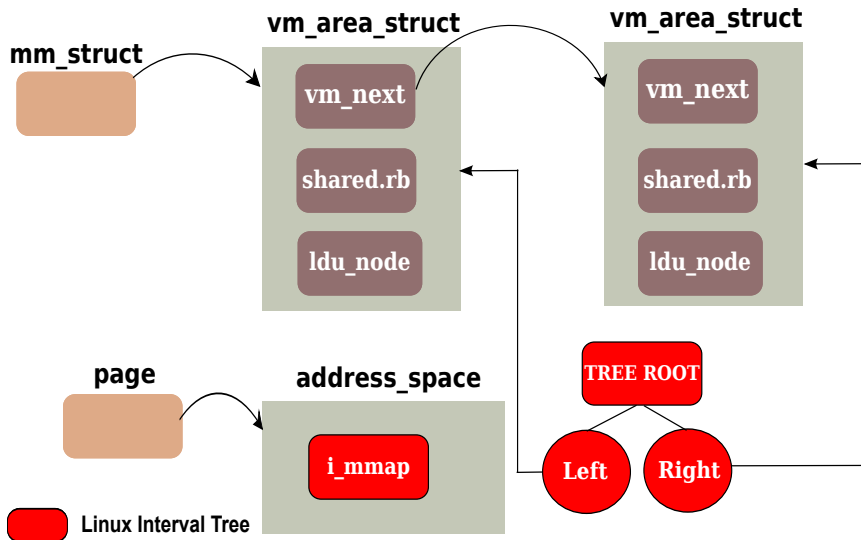
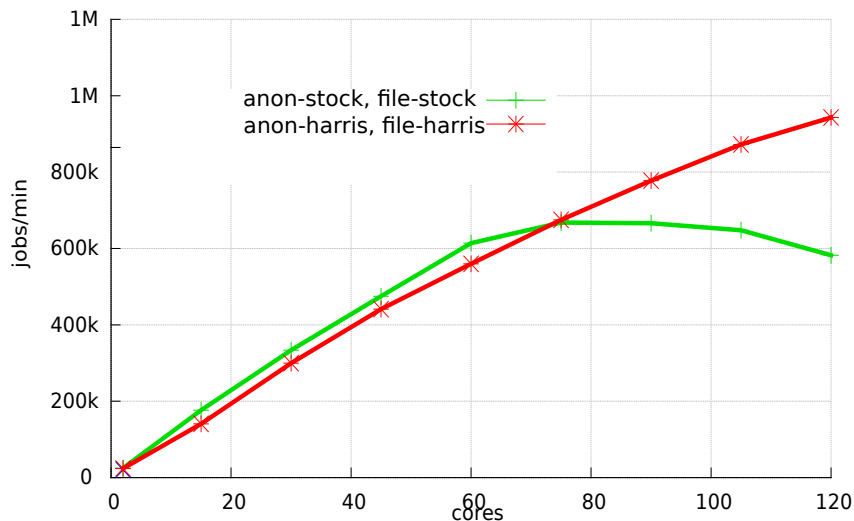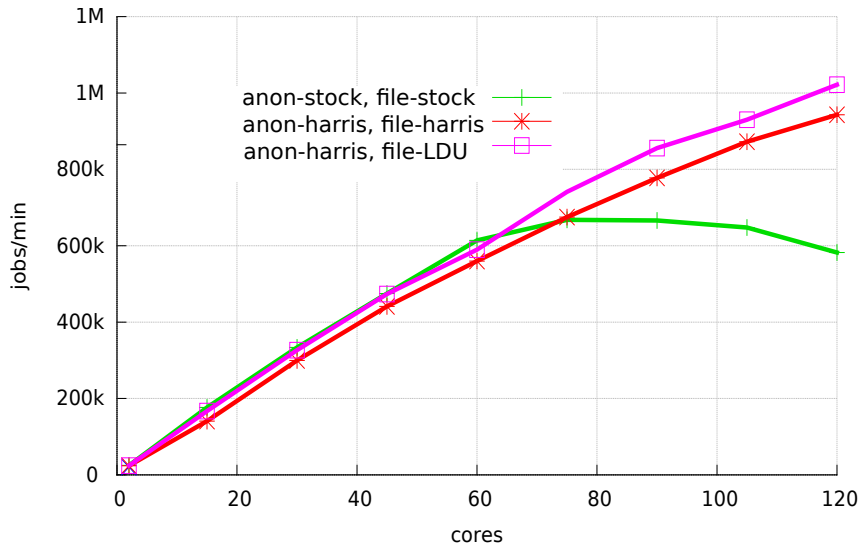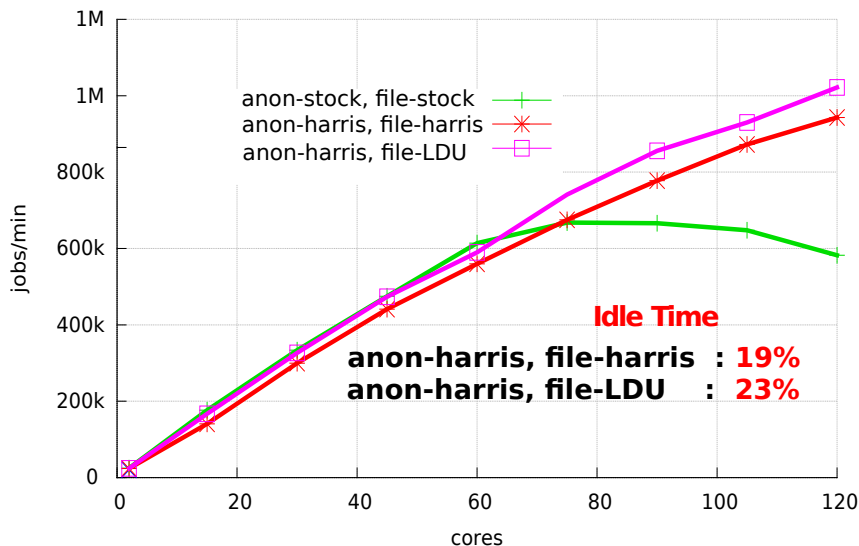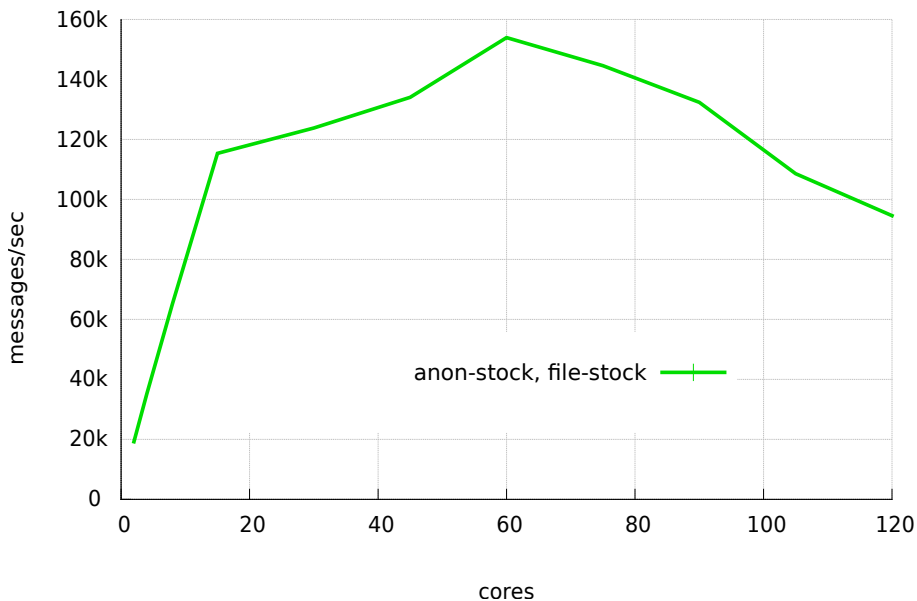# Concurrent update for Linux kernel
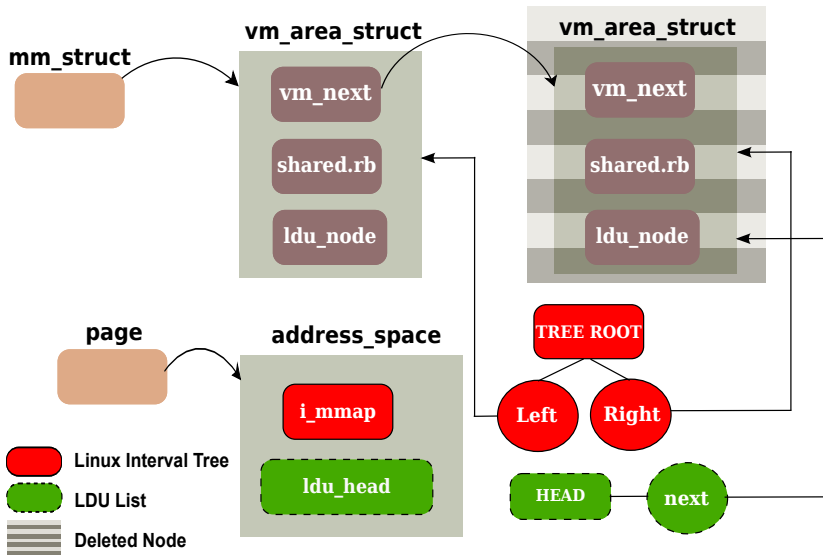
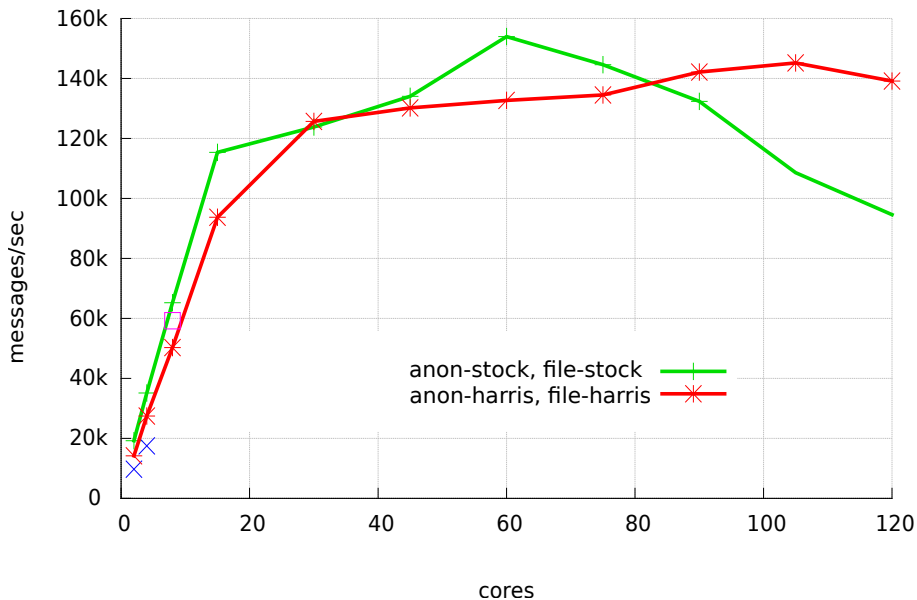# Evaluation – AIM7
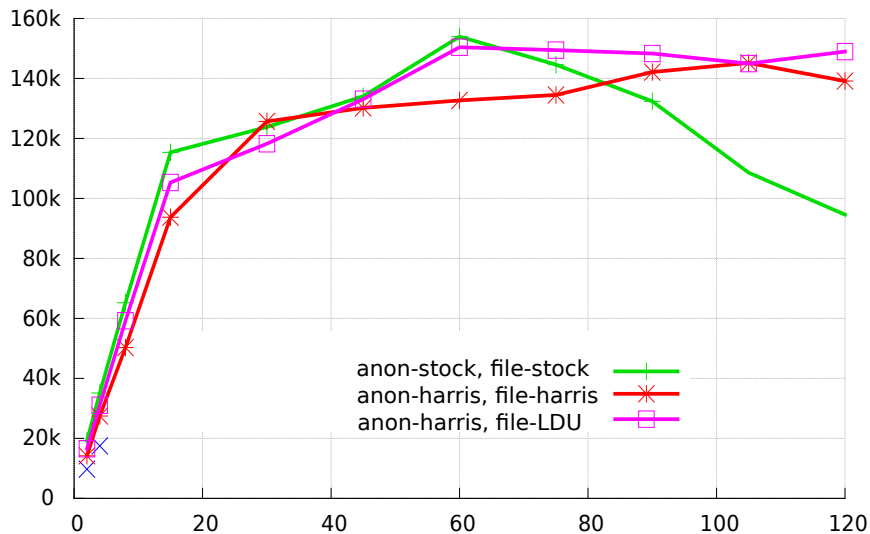
Evaluation – AIM7
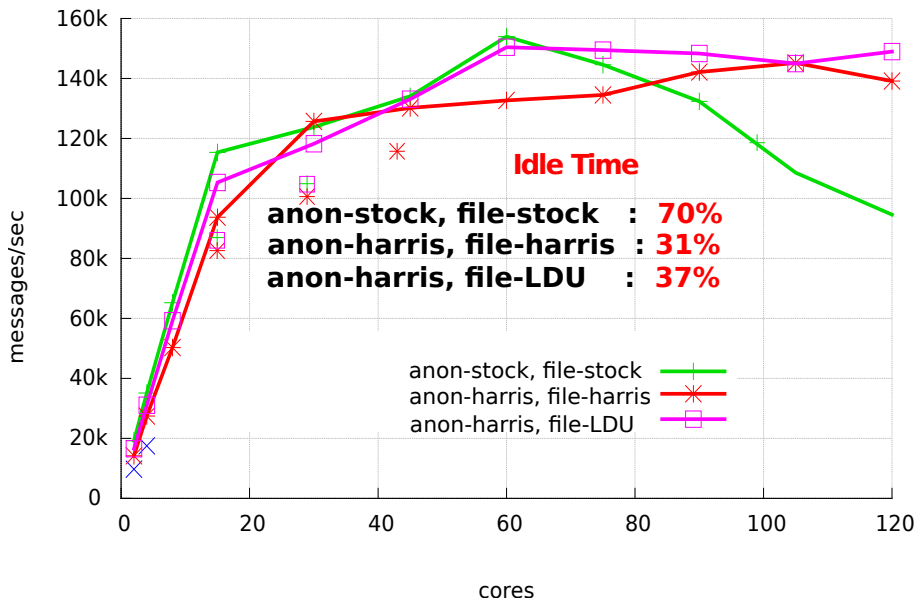
Evaluation – AIM7

Evaluation – AIM7

Evaluation – EXIM

Evaluation – EXIM

Evaluation – EXIM

Evaluation – EXIM

Idle Time

anon-stock, file-stock : 70%
anon-harris, file-harris : 31%
anon-harris, file-LDU : 37%

Summary

- History of the Linux scalability

- Linux Scalability Problems

- LDU and Evaluation

- Future plans and Summary

- https://github.com/KMU-embedded/scalablelinux

Q & A