ORACLE

The Maple Tree

Condensing 40 Liters of Data into 1

Liam R. Howlett
Linux Kernel Developer
September 14th, 2022

Overview

1. B-tree variant
    - Cache Efficient
    - Self Balancing
    - RCU-safe

2. Initially designed to track Virtual Memory Areas
    - Replacing augmented rbtree + doubly linked list + vmacache

3. Multiple node types
    - Currently supports 3 node types
        - arange 64 (allocation range)
        - range 64 and leaf 64

Other Users
Why use Maple Tree?

- Easy to Use
- Fast
- Non-overlapping Data
- Pre-allocation Support

Raison D'Être
The Data Structure Conversation

- The linked list
    - The most widely used data structure in the kernel
- rbtree
- Interval tree
    - rbtree, but with search already written
- Radix tree

 The wood of maples varies among species and is generally light in colour

Update
Internal Changes

- Settled on 256B nodes
- Pre-allocation support
- 32 bit testing
    - Next to be upstreamed

I haven't seen any issues attributed to maple tree in 2+ weeks. Unless there be weighty objections, I plan to move this series into mm-stable soon after mglru is added. Perhaps a week from now.

– **akpm**, 11/09/2022

PID Allocator
Different from VMA Tracking

PID Allocator

1. Singletons

2. Cursor
   - Tracked outside of the tree

3. Read/write balance is unclear

VMA Tracking

1. Ranges

2. Next sufficient gap
   - “Next” is arch dependent

3. Mostly reads

PID Allocator
Cyclic Allocator of Singletons

- Currently handled by Radix tree
    - Becomes sparse over time

- Dynamic node types

PID Allocator
Sparse Nodes

- All indexes without a value are NULL
- Does not support ranges
- Stores 15 indices → entry singletons
- Room for internal metadata

PID Allocator

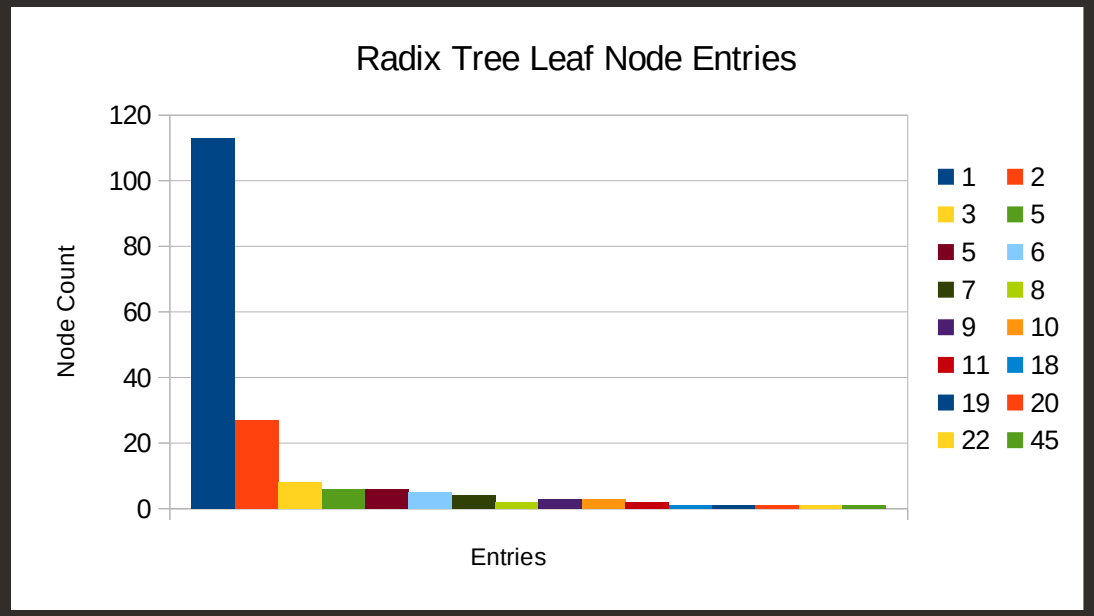
A Real Capture of 521 PIDs

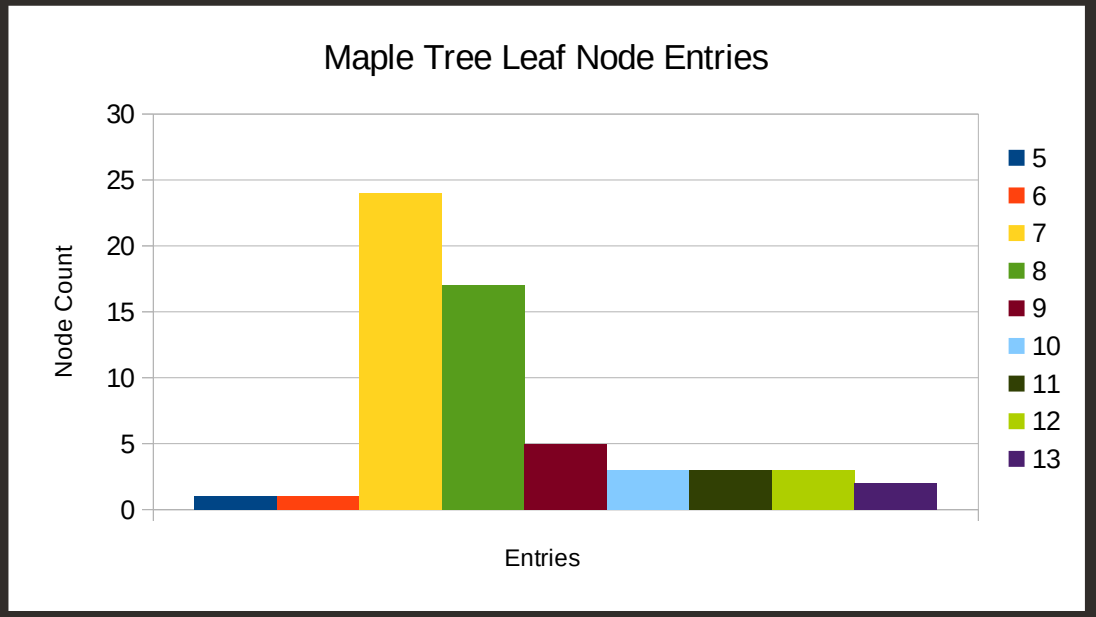
Radix tree

- 147kB
- 253 nodes
- Nodes can hold 64 entries

Maple Tree

- 16kB
- 64 nodes
- Nodes can hold 15 entries





PID Allocator
Dense Node

- Array of singletons
- Does not support ranges
- Index is implied
- Stores 31 entries
    - vs 15 in range64

Page Cache
Different from VMA Tracking

Page Cache

1. Ranges (folios) or Singletons

2. Search Marks
   - Tracked inside the tree
   - 4 bits per entry
   - Leaves will support marks

3. Mostly reads

VMA Tracking

1. Ranges

2. Gaps
   - Tracked inside the tree
   - uint64_t per entry
   - Gaps are calculated on leaf level

3. Mostly reads

Page Cache
Mark Support

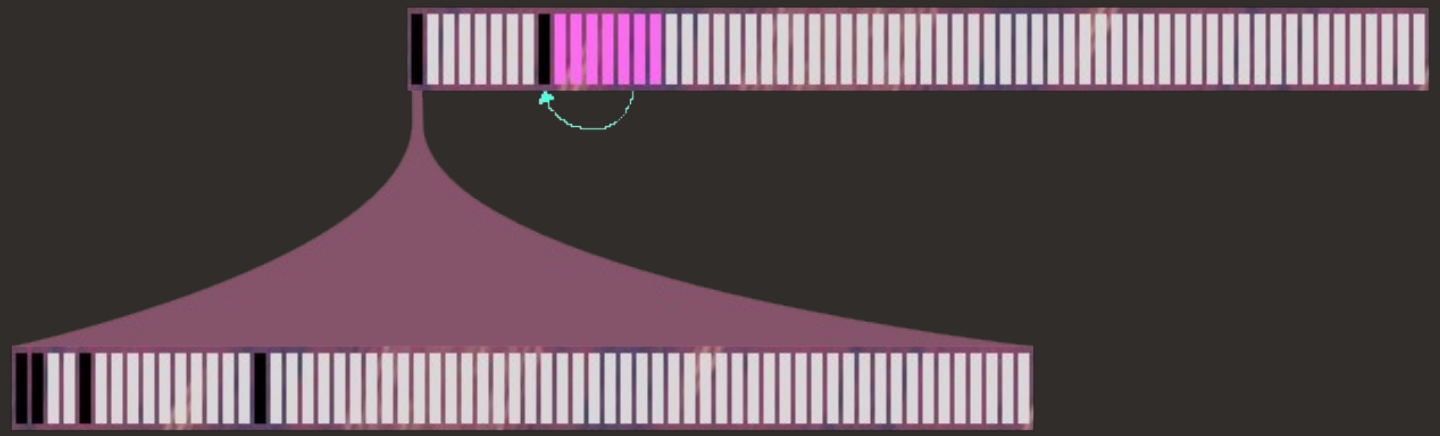
- Search for marks instead of gaps
    - Very similar to gap searching
- Potential range64 node layout
    - 14 pivots, 15 entries, 4 marks per entry

Page Cache

Large Folio Example, Store 0,1, 4, 15, 512-1023

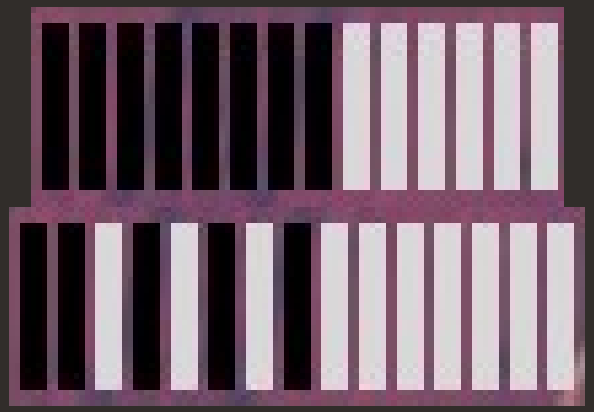
Radix Tree

1. Two nodes
2. Leaf (bottom) node has 4 entries, 60 empty
3. Root (top) node has 2 entries, 7 “Buddy” entries



Maple Tree

1. Single node
2. Top part represents the pivots
3. Bottom part represents the slots



File Descriptors
Probably Not Worth It

- Very important users allocate huge number of FD
- Any slow down on FD allocation is unacceptable

Filesystems
Extents

- Add 32 → 64 for less height
  - 16TB files aren't very common today
- Allow 32 → 32 regardless of host
- Add 64 → 32 for 32bit maybe?

Filesystems
Compressed Indexes

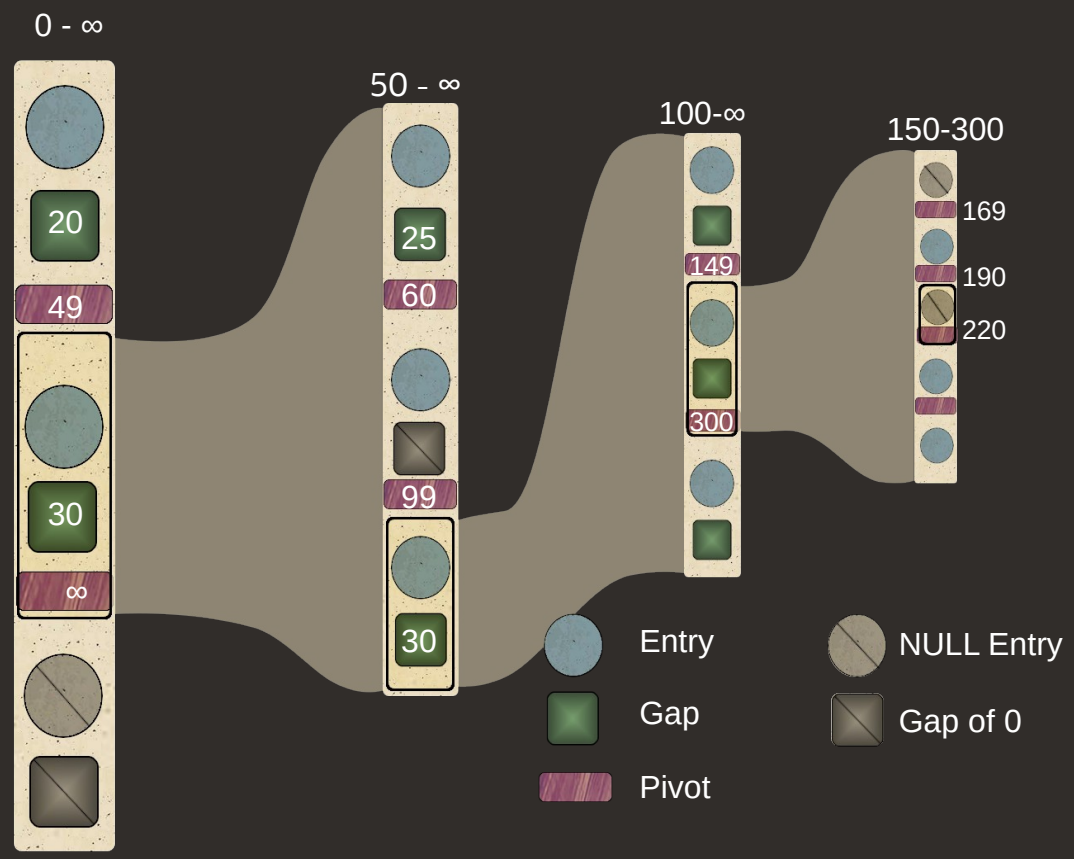
- Inherited part of Index
- Increase number of entries in the node
- Added benefit of making other uses faster

Thank you

Liam R. Howlett


ORACLE

Maple Tree Example



Maple Tree Huge Dense
4K Nodes

- Allocate a page
- Treat it like a giant array

Maple Tree Internals
Additional Improvements

- Memory Pools
- Tree Duplication
- Maple Splitting State