The Maple Tree

Not Just For Delicious Pancakes

Liam R. Howlett
Linux Plumbers Conference
September 9th, 2019

ORACLE

 | Maple Trees are generally deciduous

Talk Agenda

1. Why Another Tree?
2. Maple Tree
3. The VMA Search Problem
4. Node Types and Project Performance
5. Future Growth

Why Another Tree?

Making a more efficient tree

- Radix tree (trie)
  - When compact, Radix searches are quite efficient
  - When sparse, Radix searches are extremely poor
- Rbtree
  - Function pointers are not as fast as they were a few years ago
  - Not cache optimized
  - Not RCU safe
  - API is difficult to use

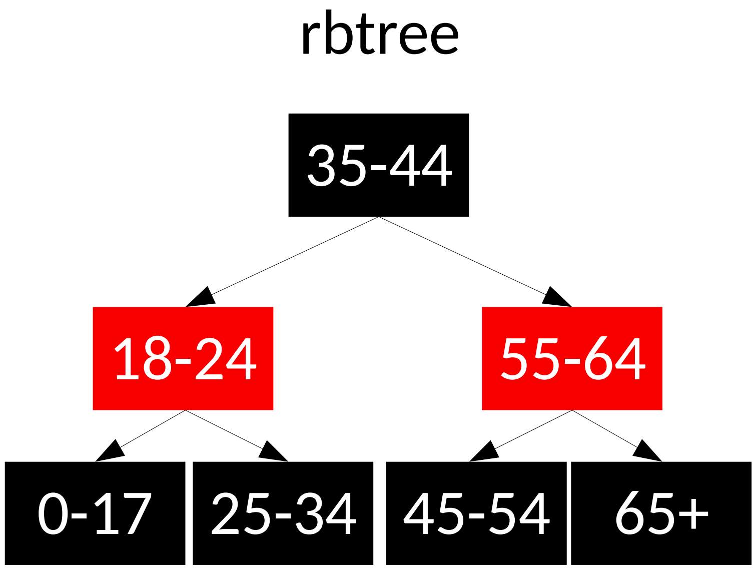
Why Another Tree?

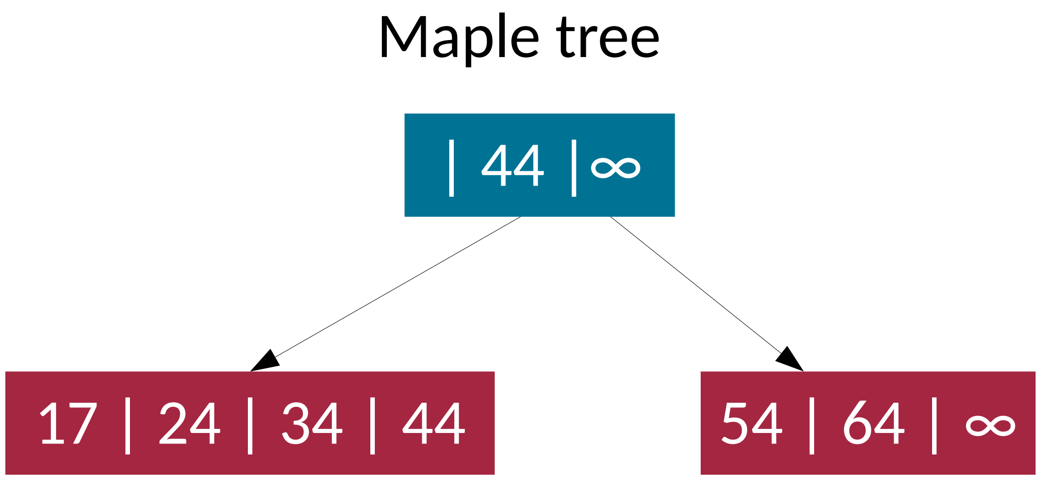
Maple: Trees for the modern CPU

- In-memory, RCU-safe, Range-based B-tree
  - Optimised for contiguous ranges
  - Does not support overlapping ranges (yet)
  - Goal to be faster than rbtrees and the Radix tree (trie)
- Multiple node formats
  - Range
  - Allocating Range (tracks gaps)
  - Dense
  - Other node types in future (compressed pivots, large leaf nodes, ...)

Maple Tree

Looking at a diverse forest



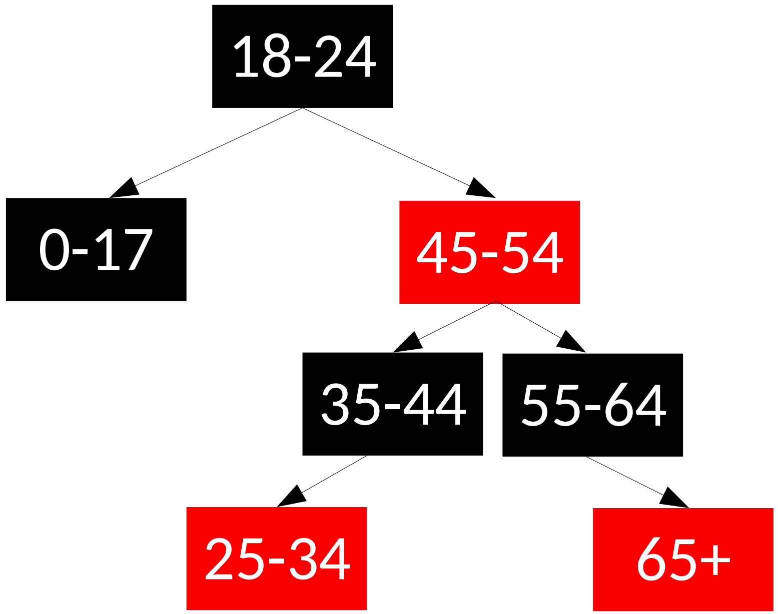


 | Dried maple wood is often used for smoking foods

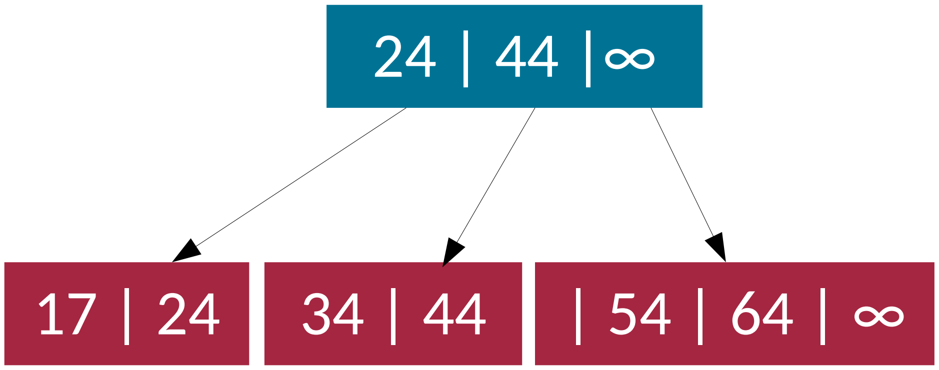
Maple Tree

Looking at a sub-optimal, yet still diverse forest

rbtree



Maple tree

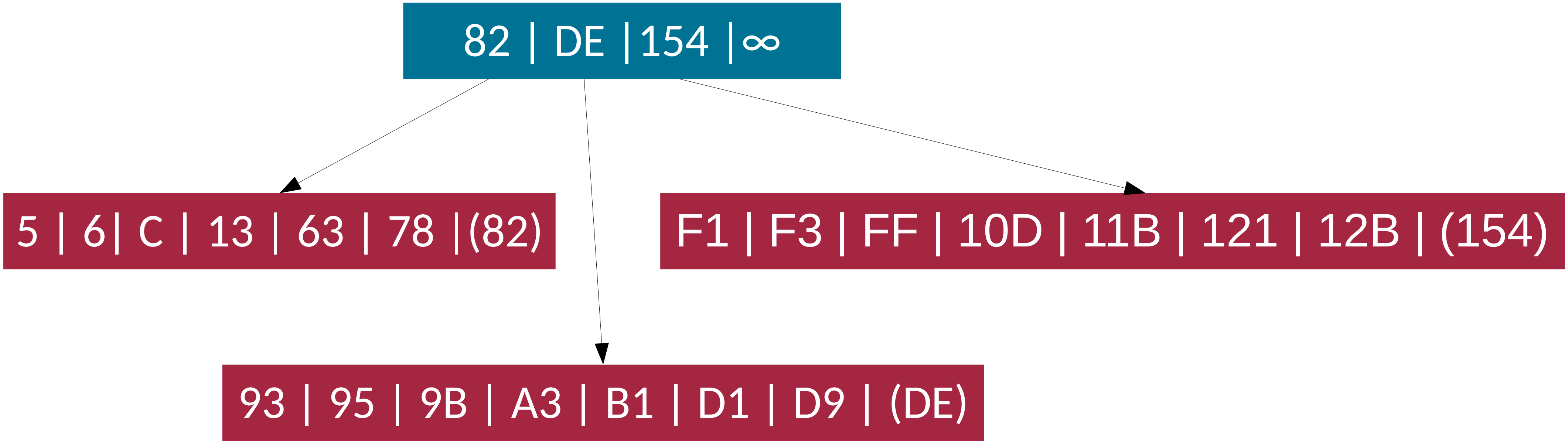


ORACLE

 | Maples can be used to produce print quality paper

Maple Tree

A full sized node maple tree



Maple Tree

Different aspects matter for different reasons

	rbtree	Radix Tree	Maple Tree
RCU Safe	No	Yes	Yes
Range support	Yes	Limited	Non-overlapping
Tree height	Tall	Short*	Medium
API	Hard	Easy	Easy
Node	Embedded	External	External
Node size	24 bytes	576 bytes	128 bytes

* with dense indices

The VMA search problem

Virtual Memory Areas

- A task's address space is a set of non-overlapping Virtual Memory Areas

- Currently stored in an augmented rbtree
  - rbtrees are not RCU safe
    - Requires mmap_sem locking to walk the tree
    - This scalability problem will be further discussed by Laurent Dufour

ORACLE

 | Maple wood is a tonewood – wood that carries sound waves

Node Types

Node Types: Now With Gaps!

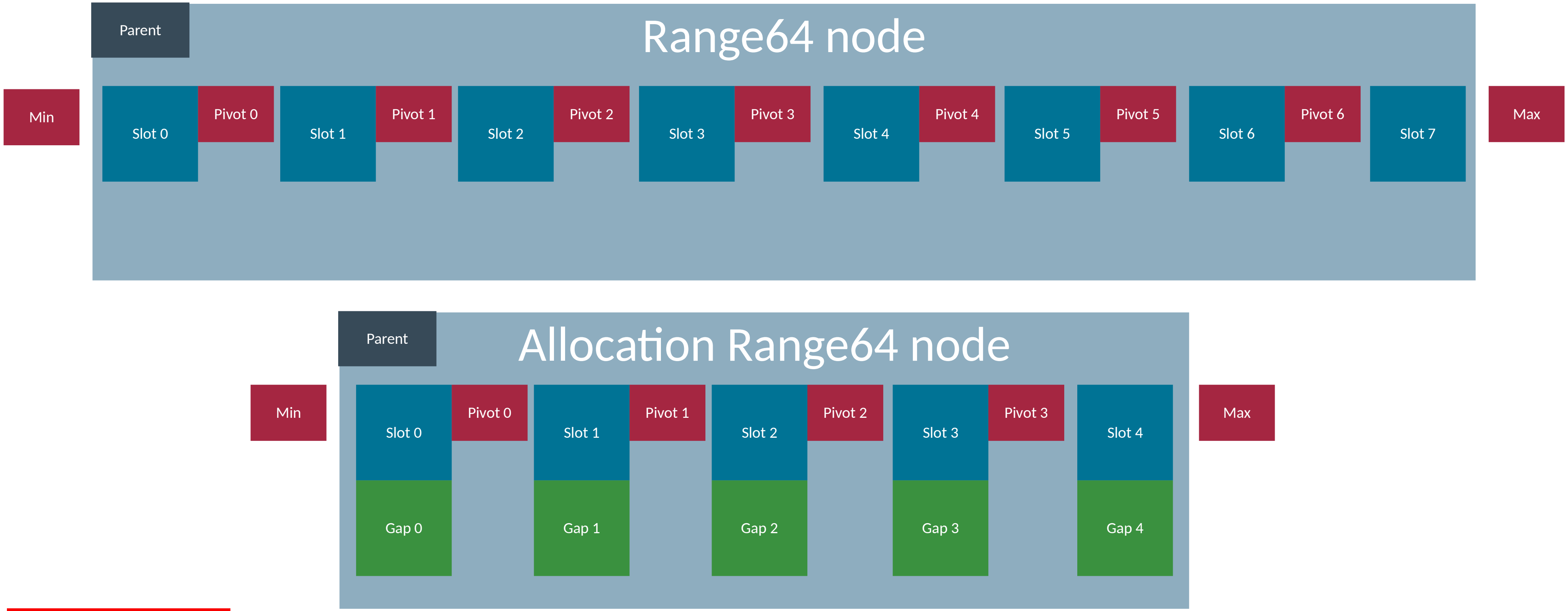
- range64
  - 8 entries (unsigned long), 8 byte indices

- arange64 – allocation range 64, tracks largest gap below
  - 5 entries & 5 gaps (unsigned long), 4 byte indices

- Dense
  - 15 entries

Node Types

Seeing the Nodes for the Trees



 | Sugar maple wood is often know as hard maple wood

Node Types

What makes maple so mapley? Just how dense are they?

- 128 byte aligned allocations
  - 7 bits for metadata

- Struct maple_enode: Encoded nodes (mapley node)
  - Metadata: Internal node, Full, Node type

- Struct maple_pnode: Parent Node
  - Metadata: Parent node type, Slot number in parent

Projected Performance

Dereference count; large process test (Firefox has 1415 VMAs)

- Perfectly balanced rbtree
  - 9.56 dereferences on average to find desired VMA
- Maple tree (~20% more NULLs, 1698 entries total)
  - Most fragmented tree has 4 entries per leaf node and 3 entries per non-leaf node:
    - 8 dereferences
  - Average tree has 6 entries per leaf node and 4 per non-leaf node:
    - 7 dereferences
  - Most compact tree has 8 entries per leaf node and 5 per non-leaf node:
    - 7 dereferences

Projected Performance

Memory usage; large process test (Firefox has 1415 VMAs)

- rbtree Implementation
  - 48 bytes per node; about 66KiB
- Maple tree (~20% more NULLs, 1698 entries total)
  - Most fragmented tree has 4 entries per leaf node and 3 entries per non-leaf node:
    - 633 nodes; about 79KiB
  - Average tree has 6 entries per leaf node and 4 per non-leaf node:
    - 378 nodes; about 47KiB
  - Most compact tree has 8 entries per leaf node and 5 per non-leaf node:
    - 268 nodes; about 34KiB

ORACLE

Current development status

Functions to implement VMA operations

- Handling page faults – mas_load()
- mmap(MAP_FIXED) – mt_store()
  - Also mremap(MREMAP_FIXED)
- Create VMA at lowest possible free location – mt_alloc_range()
- Create VMA at highest possible free location – mt_alloc_rrange()
- Find next/prev VMA – mas_next()/mas_prev()
- Iterate over all VMAs – mas_for_each()
- Grow VMA (eg stack, mremap()) - mas_store()

Solving the PID allocation problem

Also useful for cgroup ID allocation and many other *idr_alloc_cyclic* users

- Radix trees (like the IDR) are good for densely packed IDs
- Once PIDs are freed, radix tree data structure becomes inefficient
- Maple tree will be able to convert between *dense* nodes and *sparse* nodes
  - Dense nodes contain 15 pointers
  - Sparse nodes contain up to 7 pointers and 7 values. All other values in the range covered by this node are implicitly NULL

ORACLE

Large, dense nodes

Useful for the file descriptor table

- Allocate an entire page
  - 512 * 8 bytes in a page
- Store the parent pointer in the *struct page*
- Takes three levels out of the tree for dense regions

Replacing hash tables

- Sparse nodes may let us outperform hash tables
  - More benchmarking!
- Hash tables are often mis-sized
  - Walking long hash chains is expensive
  - Large top-level arrays waste memory

 | Maple flowers are green, yellow, orange, and red

Short Term Plans

Budding interests

- Finish VMA tree conversion
- Benchmark & more testing
- Support 32-bit CPUs
- Add support for search marks
- Dust off dense node implementation
- Use XArray API for Maple Tree
- Implement sparse64 nodes

 | Fall colours are produced by pigments called anthocyanins

Open questions

- How do we remove old shadow entries from the page cache?
- What should the batch API look like?
- Do the benefits of a larger node size outweigh the disadvantages?
- Can we replace rbtrees with overlapping ranges?
- Is it worth adding a new node type for ranges with gaps between them?
- How many search marks is it useful to support?

 | Bigleaf maple trees are able to grow 40m tall or more.