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
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
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

---

**Radix Tree Leaf Node Entries**

<table>
<thead>
<tr>
<th>Node Count</th>
<th>Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>2</td>
</tr>
<tr>
<td>90</td>
<td>3</td>
</tr>
<tr>
<td>70</td>
<td>5</td>
</tr>
<tr>
<td>50</td>
<td>6</td>
</tr>
<tr>
<td>30</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>17</td>
</tr>
</tbody>
</table>

**Maple Tree Leaf Node Entries**

<table>
<thead>
<tr>
<th>Node Count</th>
<th>Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Page Cache</th>
<th>VMA Tracking</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Ranges (folios) or Singletons</td>
<td>1. Ranges</td>
</tr>
<tr>
<td>2. Search Marks</td>
<td>2. Gaps</td>
</tr>
<tr>
<td>• Tracked inside the tree</td>
<td>• Tracked inside the tree</td>
</tr>
<tr>
<td>• 4 bits per entry</td>
<td>• uint64_t per entry</td>
</tr>
<tr>
<td>• Leaves will support marks</td>
<td>• Gaps are calculated on leaf level</td>
</tr>
<tr>
<td>3. Mostly reads</td>
<td>3. Mostly reads</td>
</tr>
</tbody>
</table>

---

Maple drums are favoured for their bright resonant sound
Mark Support

• Search for marks instead of gaps
  • Very similar to gap searching

• Potential range 64 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?
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
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