rtree

RCU based
self-balancing tree

Manuel Messner <mm@skelett.io>

BONSAI tree

- Research project of MIT CSAIL
- By Austin T. Clements, M. Frans Kaashoek, Nickolai Zeldovich
- Presented in 2012

Address space management did not scale
=> Locking of used tree

Solution: RCU based tree
=> BONSAI tree

BONSAI tree

- RCU
- Self-balancing
- < 500 lines of code
- No general purpose data structure

- Borrows idea of constant data from functional programming.

BONSAI tree

Tree modifications are done by partially*
recreating the tree next to the existing one!

Result:
=> RCU-friendly (atomic) insertion of new subtrees.
=> Lock-less rotations are possible.

*) partially: all nodes under the modified one

BONSAI tree

Consequences:

- Multiple subtrees can exist in parallel:
  => Potentially high memory usage
- Potentially expensive rotation
  => Lots of nodes might be recreated

Solution:

=> Weight calculation supports parameter to configure threshold.
Large trees rotate less!

rtree

- Implementation of the BONSAI tree:

- Adds correct RCU usage:
  borrowed from kernel/bpf/lpm_trie

- General purpose data structure:
  partially borrowed from lib/llist

- Adds error handling

rtree

Offers:

- Insertion/deletion functionality
- Lookup functionality
- Traversing functionality

Needs:

- Comparison callback
- Creation callback
- Deletion callback

rtree

```
File                     | Blank | Cmnts | Code | Sum
-------------------------|-------|-------|------|-----
include/linux/rtree.h    |    23 |     5 |   43 |   71
lib/rtree.c              |   131 |    43 |  367 |  541
lib/test_rtree.c         |   118 |     9 |  308 |  435
-------------------------|-------|-------|------|-----
Sum                      |   272 |    56 |  718 | 1047
```

A usage example is in lib/test_rtree.c