

Tune Disk I/O To Speed Up MeeGo Boot

Presented by:

Shaohua Li <shaohua.li@intel.com>



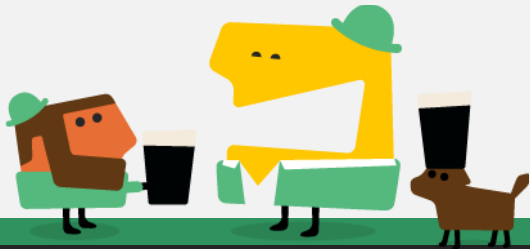
Agenda

- Overview
- Better Defragmentation
- metadata readahead
- Things TO DO



Target Requirements

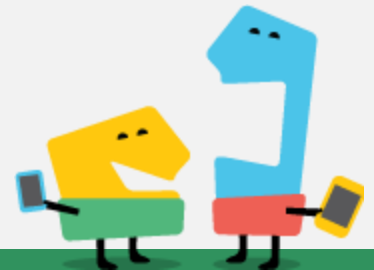
- Only evaluate disk I/O impact
- Hard disk* based system



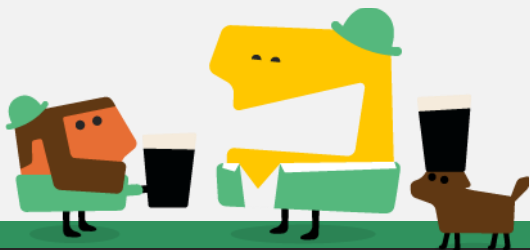
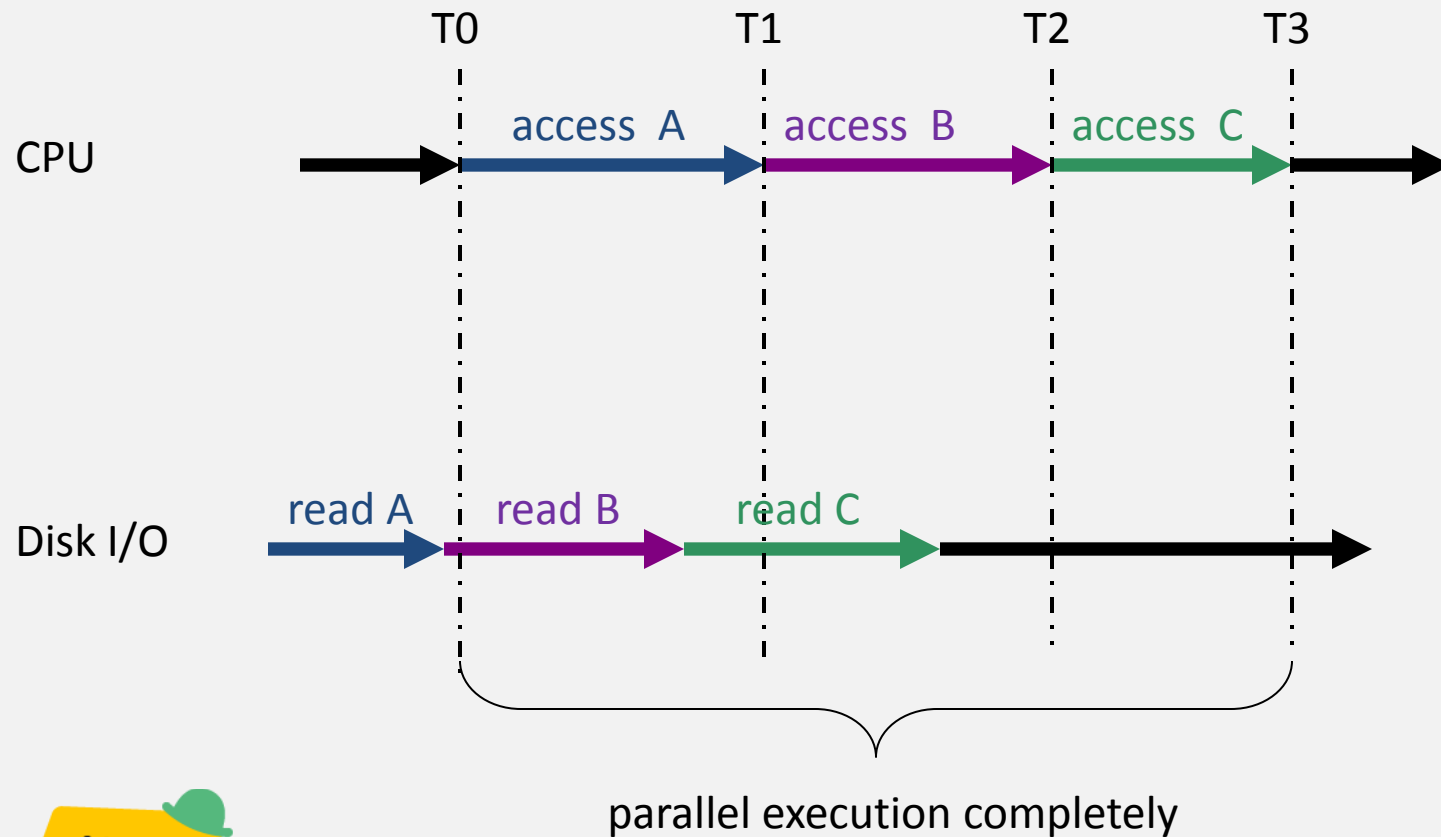
*Hard disk means rotate disk in the talk

Disk I/O Impact for Boot

- Why matters? slow disk vs. fast CPU
- OS generally deploys a kind of readahead to overcome the imbalance
 - CPU and disk I/O run in parallel without interfere
 - For example:
 - ureadahead for Ubuntu
 - sreadahead for MeeGo



How readahead Works



sreadahead

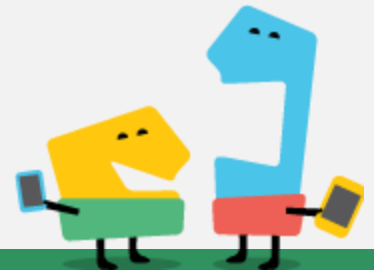
- Similar like other readahead
- How sreadahead works?
 - Collection stage (first run)
 - Collect info about all files required by boot (mincore)
 - do defragment for all the files
 - Readahead stage (later run)
 - read data of the files per collected info (readahead)
 - Do normal boot
- Great for SSD, but not very efficient for rotate disk



Test Environment



- EeeePC 1005PE
- MeeGo 1.0
- sreadahead version 0.10
- Root filesystem is btrfs

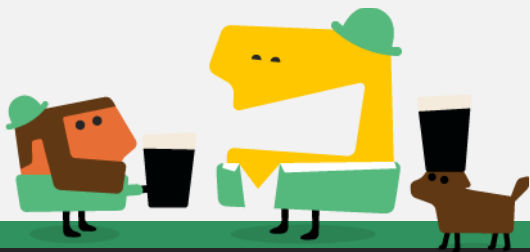


Boot Chart Without Either Optimization



Hard Disk Performance Principles

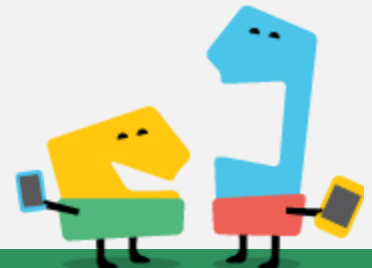
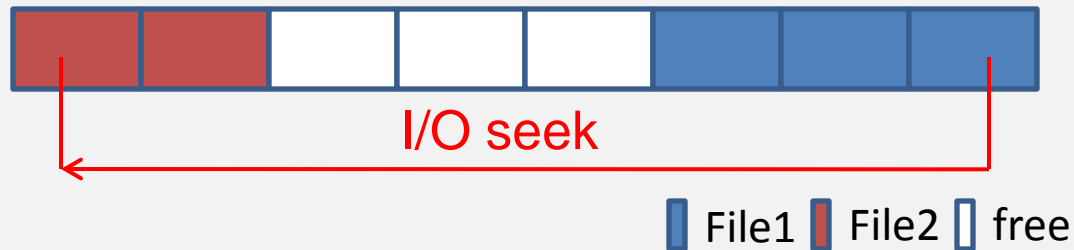
- Avoid spindle seek because seek penalty is big
- Merge I/O requests because small and big requests take the same time usually



Better Defragment



- Boot reads several files, file1, file2, file3...
- sreadahead implementation:
 - Do defragment for the files
 - Disk blocks for one file are adjacent
 - Disk blocks between files are not adjacent
- Not sufficient



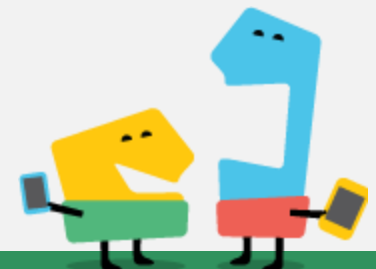
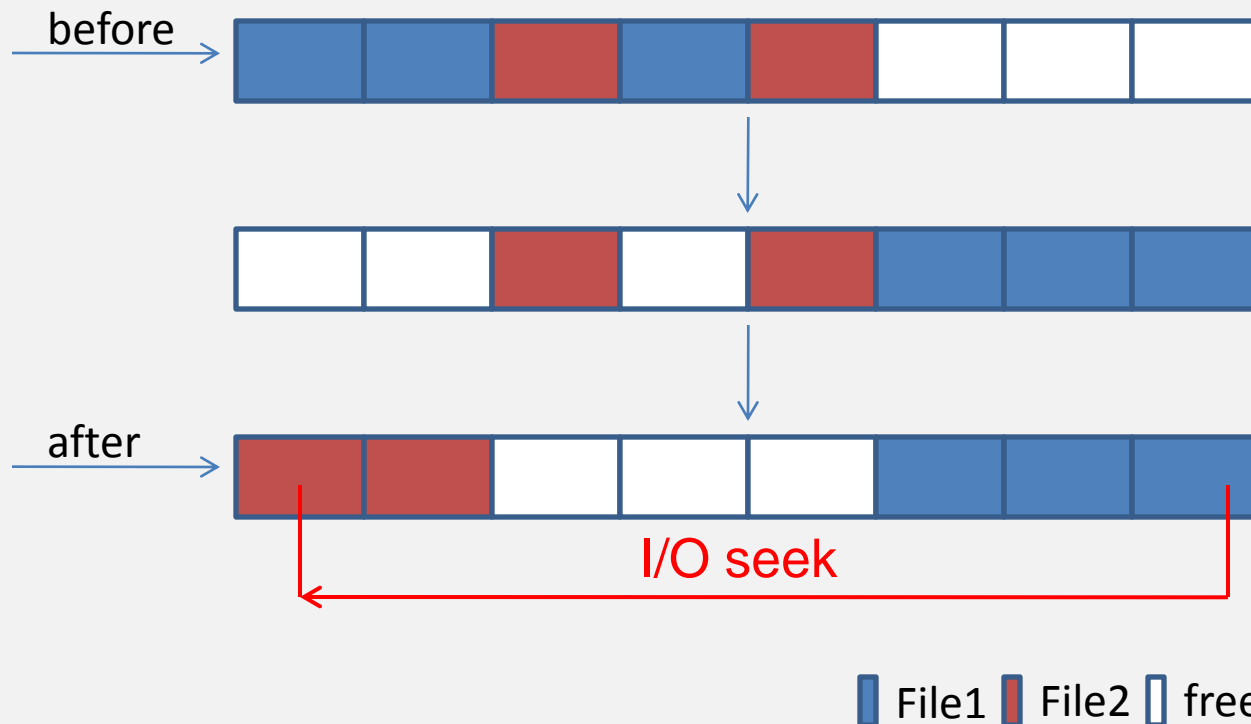
How

- Before doing defragment, mount file system with ‘-o ssd’
 - btrfs uses a different block allocator policy
 - Without ‘-o ssd’: find a free block range fitting a file, and store the file to the block range
 - With ‘-o ssd’: allocate a big range (usually 1G) in a batch, subsequent allocations are all taken from the big range



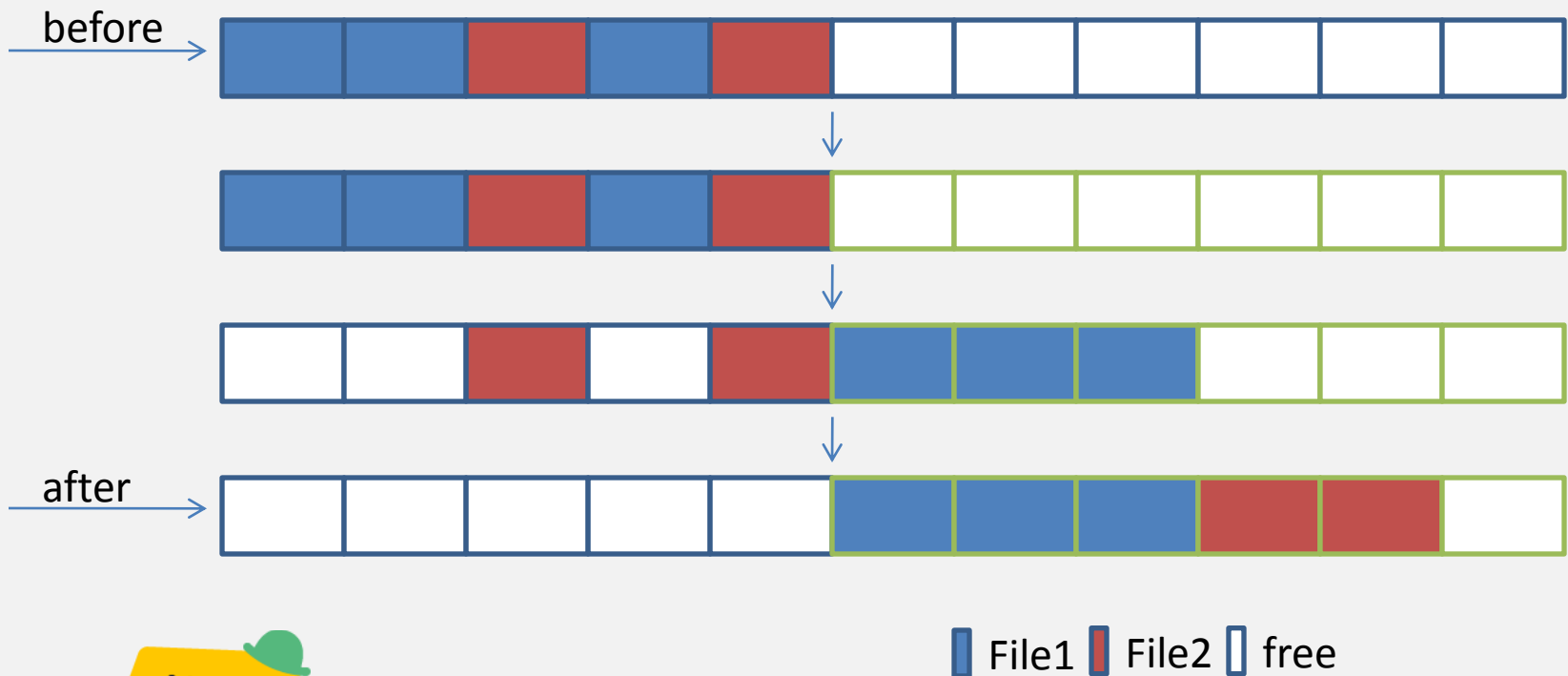
Defrag without '-o ssd'

🍀 Defragmenting File1 and File2




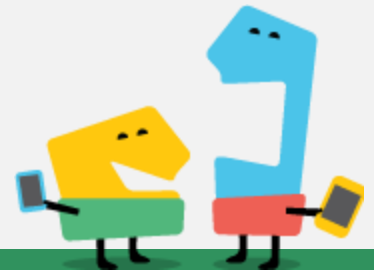
Defrag with '-o ssd'

Defragmenting File1 and File2



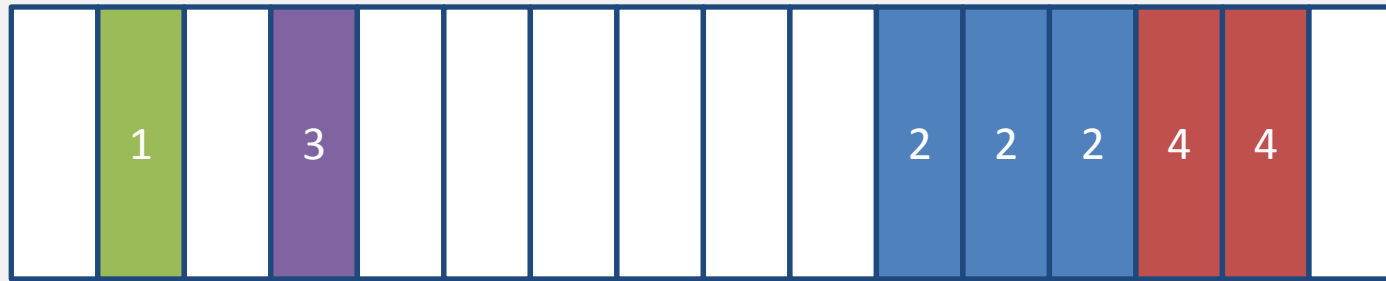
Results

-  Less I/O seek
- Average savings of 1 ~ 3 seconds boot time
- The optimization is merged in MeeGo already

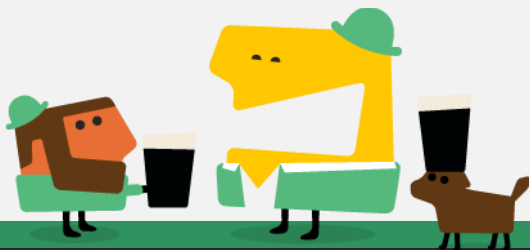


Better Defrag is Still Not Sufficient


- Read file involves both data and metadata (inode info)
- Metadata read is synchronous read and data readahead must wait for metadata read finish



■ File1 metadata ■ File1 data ■ File2 metadata ■ File2 data ■ free



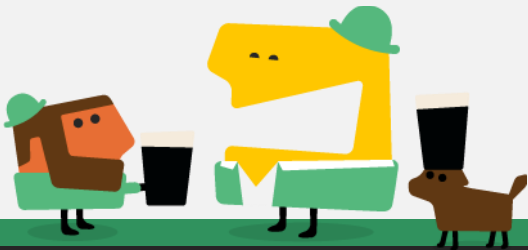
metadata readahead

-  Two new ioctls for btrfs:
 - metadata_incore (find metadata in memory)
 - metadata_readahead (readahead metadata)
- New ioctls looks just like readahead/mincore syscall, but acts for special btree_inode of btrfs
- btree_inode stores all metadata in btrfs



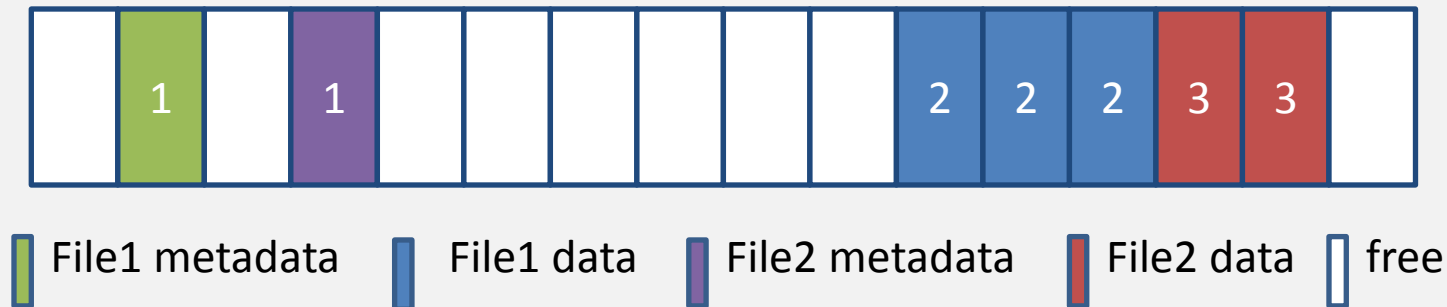
How to utilize them

- Collection stage:
 - metadata_incore collect metadata in memory
 - Store the info to a configuration file
- readahead stage:
 - read metadata info from the configuration file
 - metadata_readahead reads all metadata one time
 - start previous sreadahead process



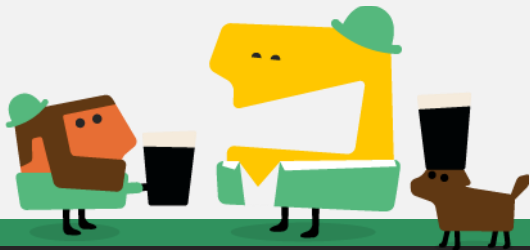
Advantages of metadata readahead

- Metadata read is asynchronous
- Still has seeks but:
 - Seek in one round and there is no back seek
 - No mixed seeks between data and metadata
- In data readahead stage, all metadata are in memory. Data readahead can be fully pumped



Results

- Less seek and no data readahead stall time
- Average savings of 3.5 seconds boot time



Hit Rate

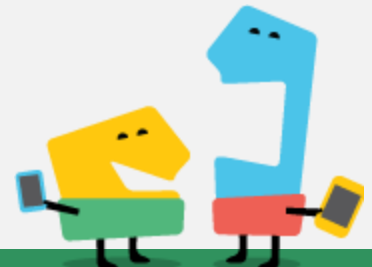
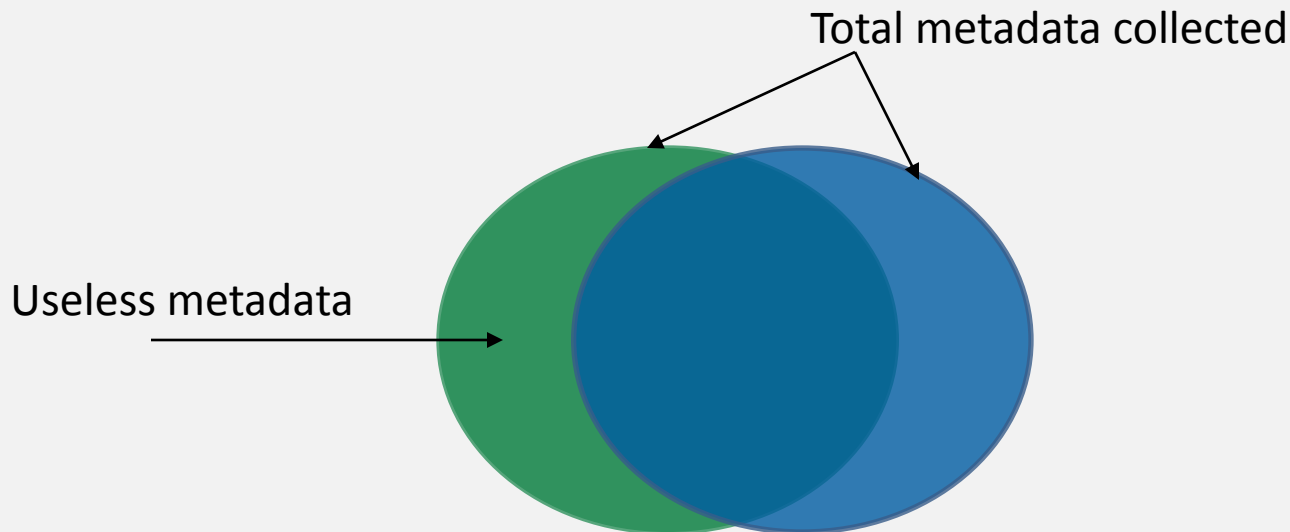
- Is metadata collected before still valid?
- Hit rate = (valid metadata read by sreadahead)/(total metadata required by boot)
- Do a test:
 - Create 4 directories, and create 12 empty files for each directory. Delete 2 directories, and each directory has 12 empty files
 - The hit rate > 80% and the test has no hurt to boot time
- Tip: Do metadata info collect every day or every boot because metadata_incore is optimized to be low overhead



metadata Info Increases?

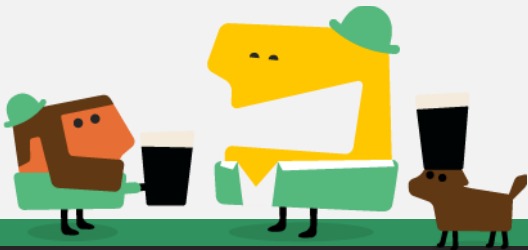


- Considering below scenario:
 - Collect metadata info (metadata set **A**)
 - Do some fs operations (metadata set **B**)
 - Collect metadata info at the second time
 - Total metadata info collected is **AUB**
 - Useless metadata info collected is (**A-B**)



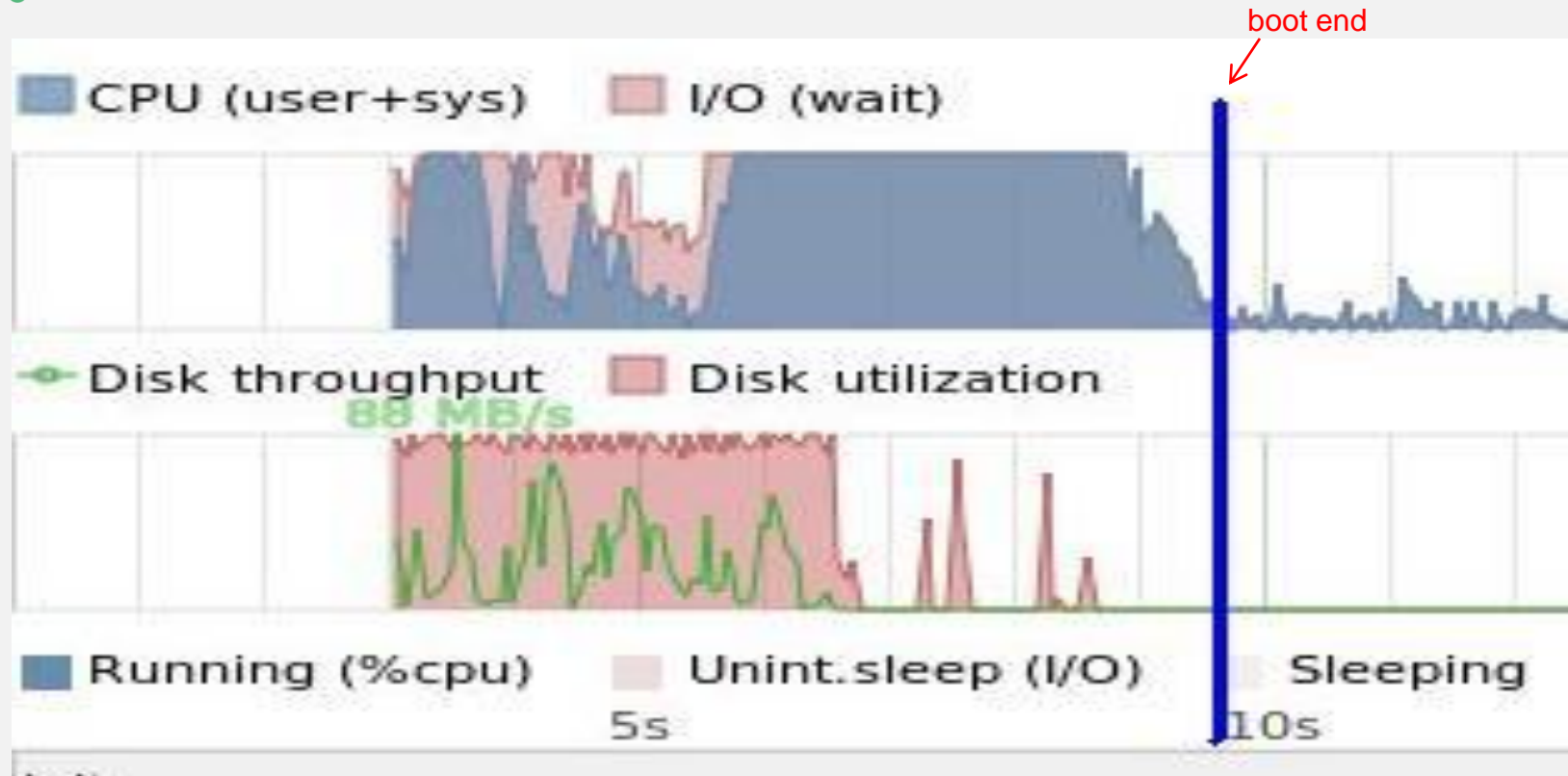
How to Avoid metadata Info Increasing

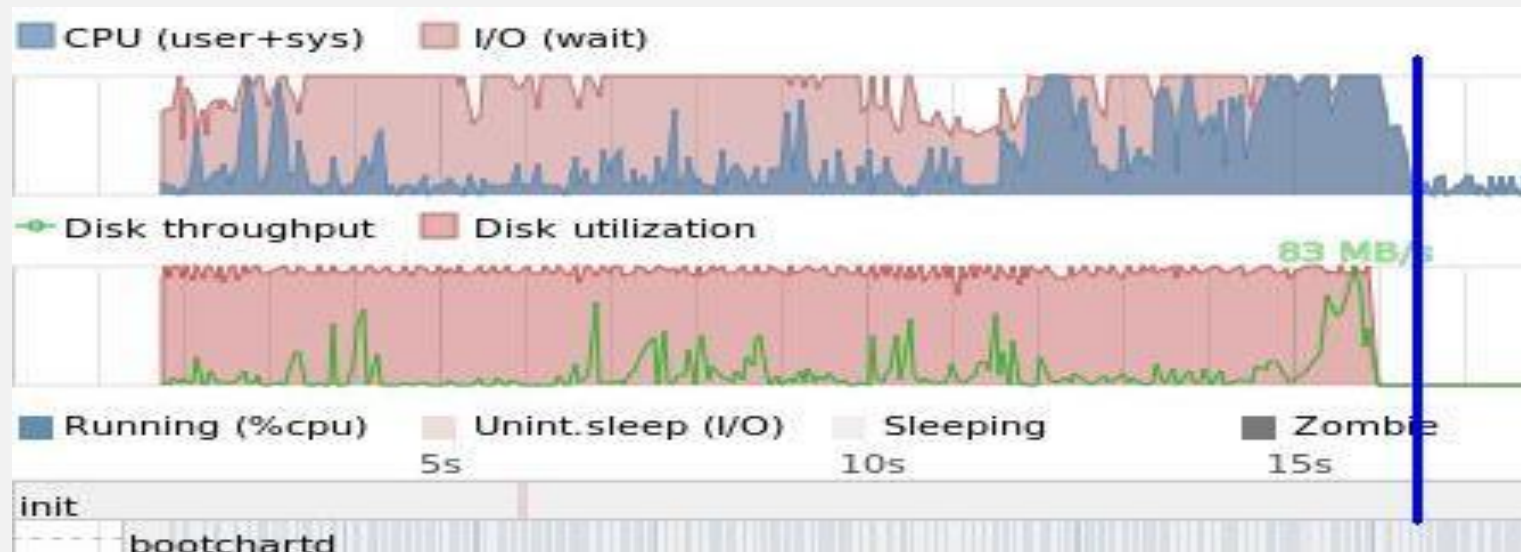
- Before btrfs accesses a metadata page, the 'referenced' bit will be set for the page. Otherwise, not set
- Metadata_incore ignores pages without the bit set
- In this way we can filter useless metadata



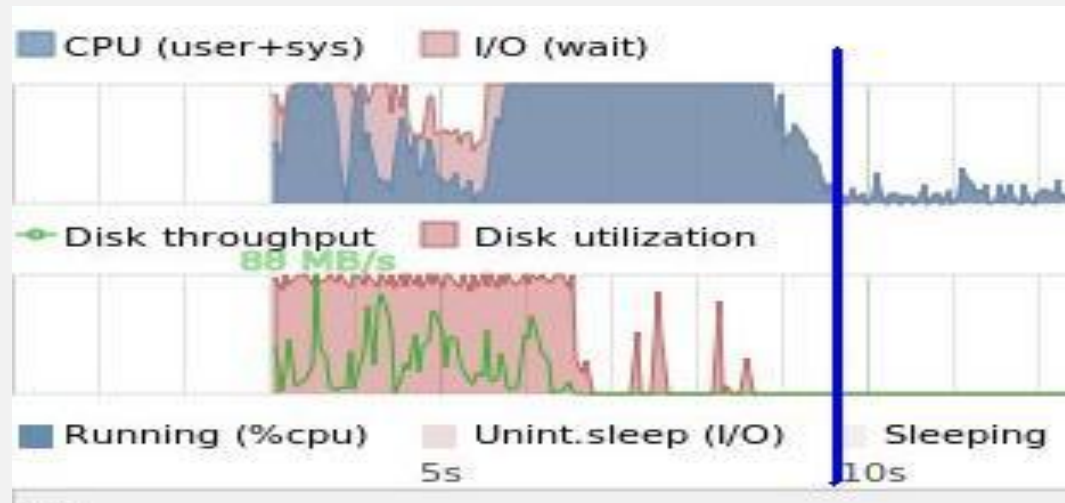
Combined Optimization Results

☘ Total boot time saved is about 6 seconds

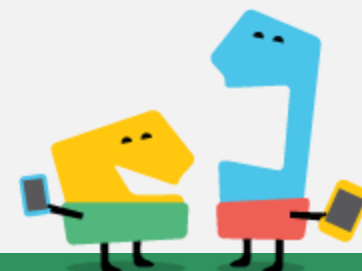




original

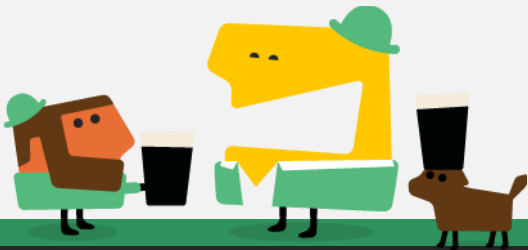


new

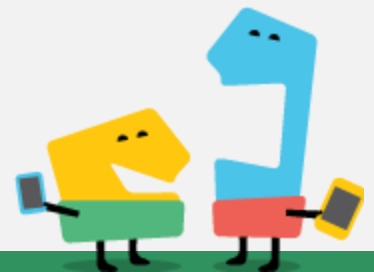


Things TO DO

- Push to upstream and to MeeGo
- Evaluate if the metadata readahead optimization helps SSD
- Evaluate the impact of range defragmentation



Q&A



THANK
YOU

