



IBM

VFS hot tracking Development Overview

Zhi Yong Wu – wuzhy@linux.vnet.ibm.com
Kernel Team
IBM Linux Technology Center

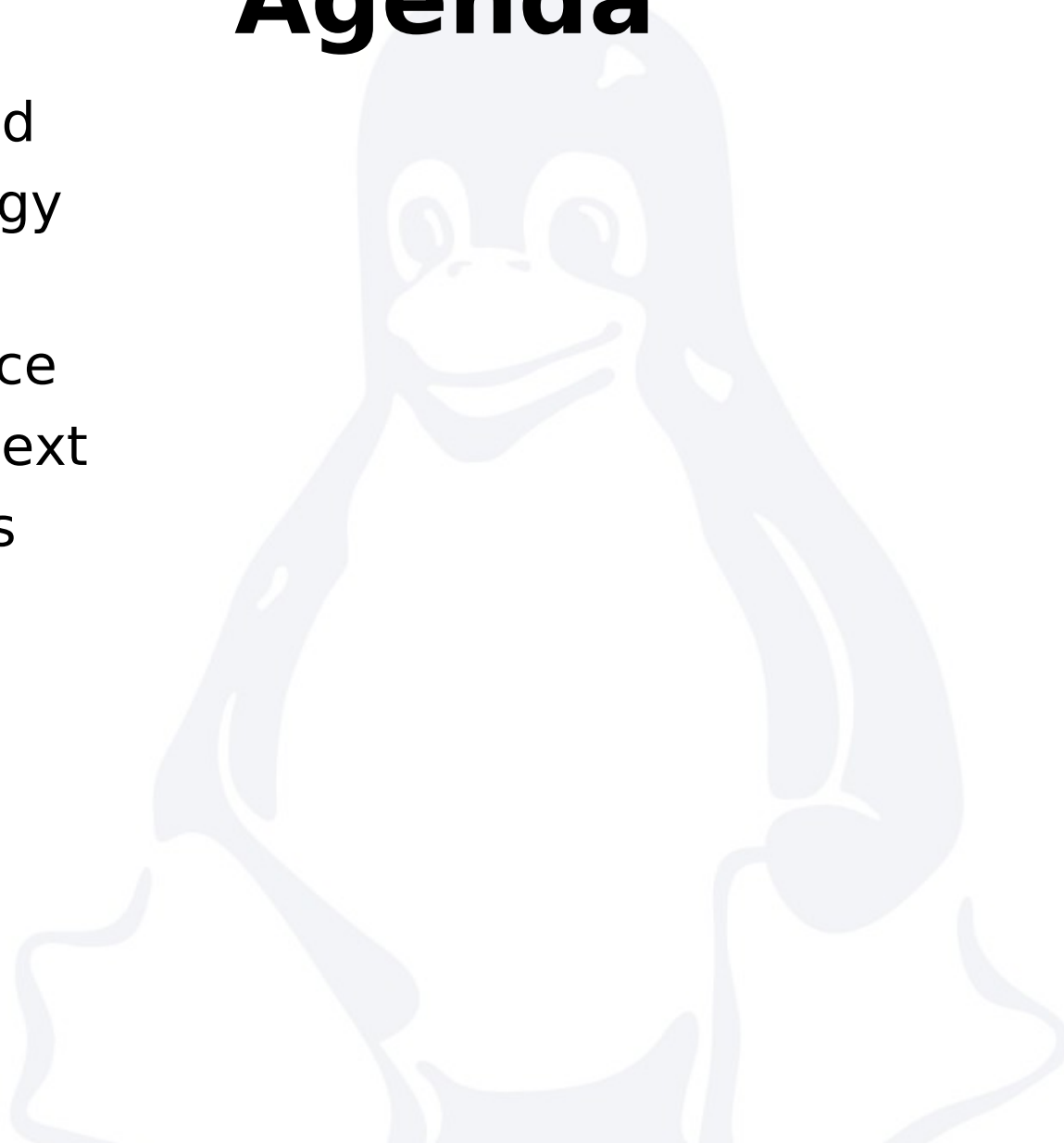
October 19, 2013

IBM



Agenda

- Background
- Methodology
- Internals
- Performance
- Status & Next
- References



Background

- Problem:
 - SSD disks has high IOPS(Input/Output Per Second) and low capability, while traditional disks has opposite peculiarities.
 - Some data are highly accessed, while some are rarely.
- Vision:
 - Place hot data on fast disks as far as possible.
 - Defrag hot files as first as possible
- Proposal:
 - Trace and detect hot data on the filesystem
 - Relocate hot data to fast disks
 - Defrag the files based on hot information



How to track?

- Track real disk I/O access, not I/O hit in page cache
- Track accessed inodes and its ranges whose granularity is 1 MegaByte
- The key is
 - ino → inode
 - offset → range
- Track each read/write on I/O path, including buffered and direct mode



How to find hot spots?

- Each hot item is stored in
 - Inserted into rb_tree
 - Linked to hot map list
- Each hot item is indexed in two ways
 - One by ino or offset in rb_tree, used to quickly update data access frequency
 - One by temperature in hot map array, used to quickly lookup hot spots
- One delayed worker is queued periodically to update the temperature of each hot item, and move it to irresponding hot map list based on its temperature.



Data Structures

```
struct hot_freq {
    struct timespec last_read_time;
    struct timespec last_write_time;
    u32 nr_reads;
    u32 nr_writes;
    u64 avg_delta_reads;
    u64 avg_delta_writes;
    u32 last_temp;
};

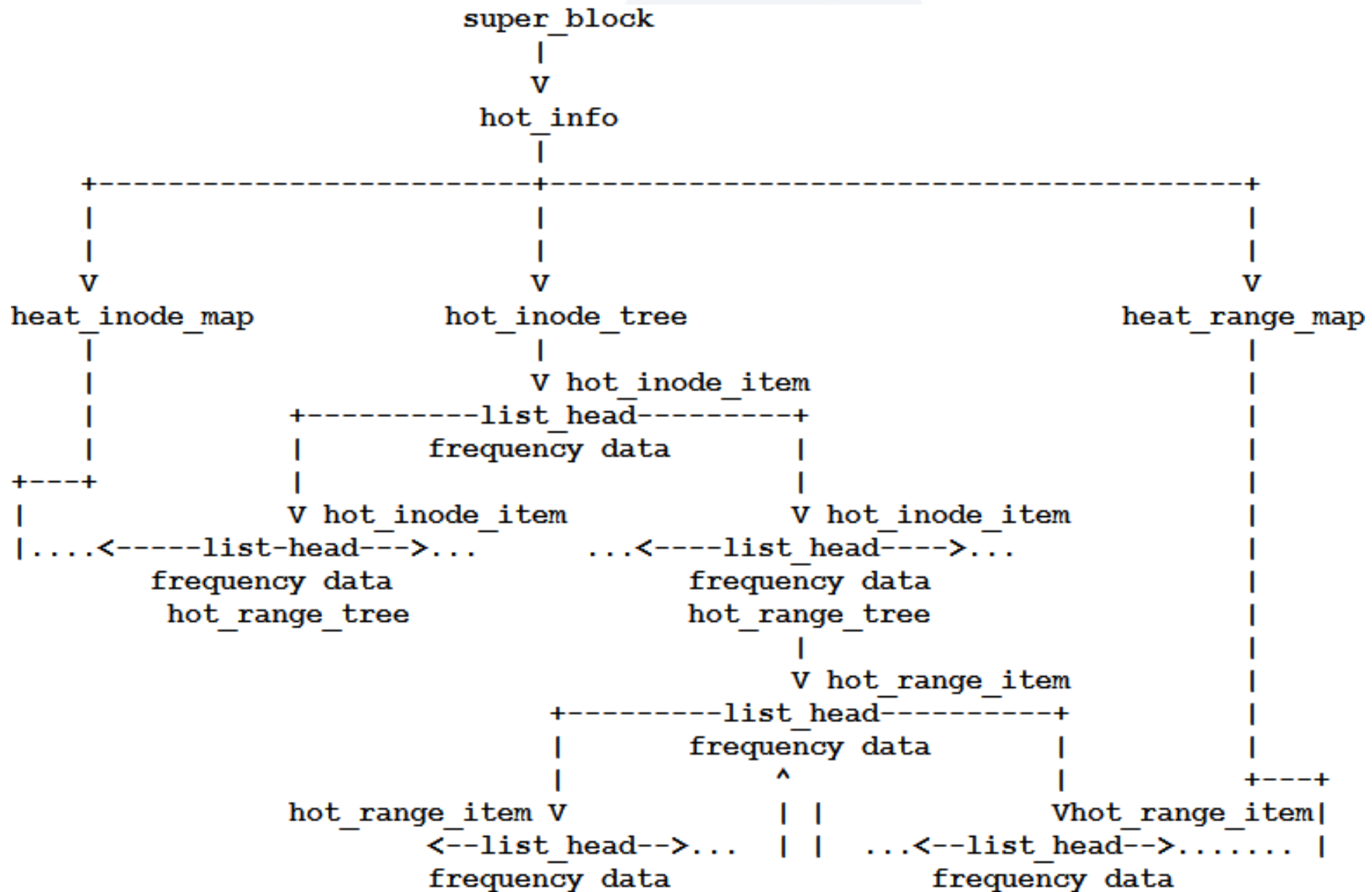
struct hot_inode_item {
    struct hot_freq freq;           /* frequency data */
    struct rb_node rb_node;        /* rbtree index */
    struct list_head track_list;   /* link to *_map[] */
    struct rb_root hot_range_tree; /* tree of ranges */
    u64 ino;                       /* inode number from inode */
    .....
};

struct hot_range_item {
    struct hot_freq freq;           /* frequency data */
    struct rb_node rb_node;        /* rbtree index */
    struct list_head track_list;   /* link to *_map[] */
    loff_t start;                 /* offset in bytes */
    size_t len;                   /* length in bytes */
    .....
};

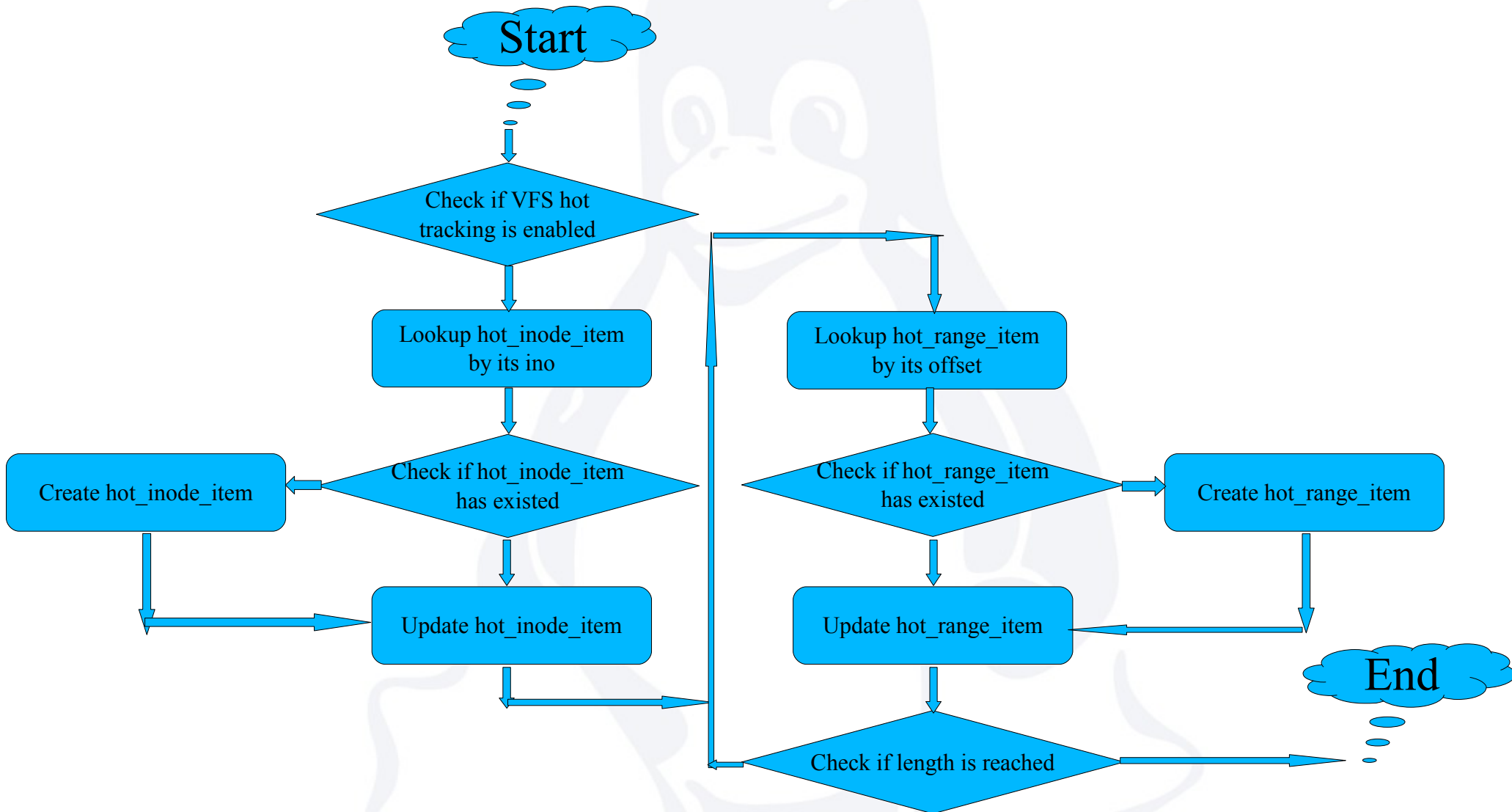
struct hot_info {
    struct rb_root hot_inode_tree;
    struct list_head hot_map[MAX_TYPES][MAP_SIZE]; /* map of inode temp */
    atomic_long_t hot_cnt;
    struct workqueue_struct *update_wq;
    struct delayed_work update_work;
    struct shrinker hot_shrink;
    atomic_long_t mem;
    .....
};
```



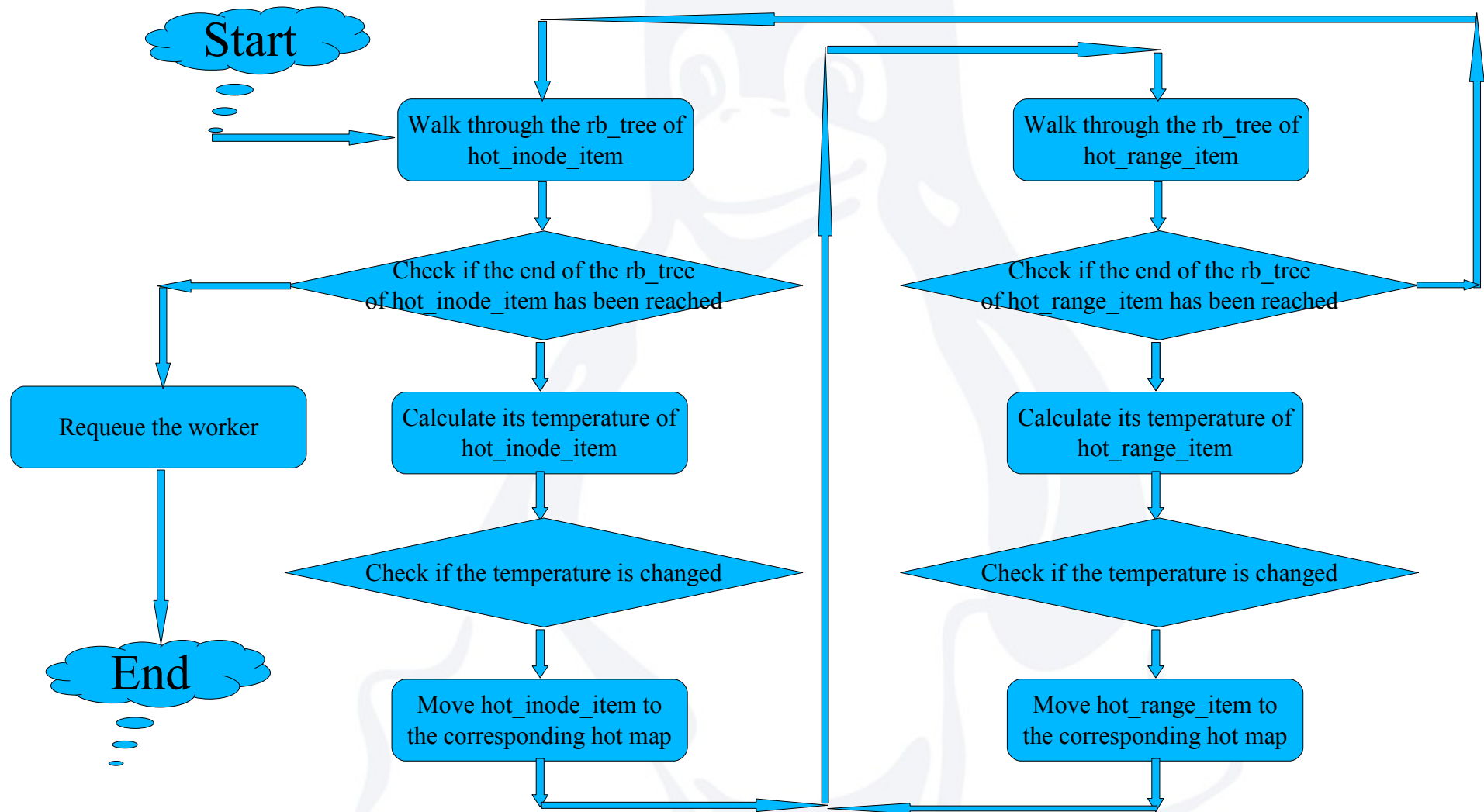
Structure Relationship



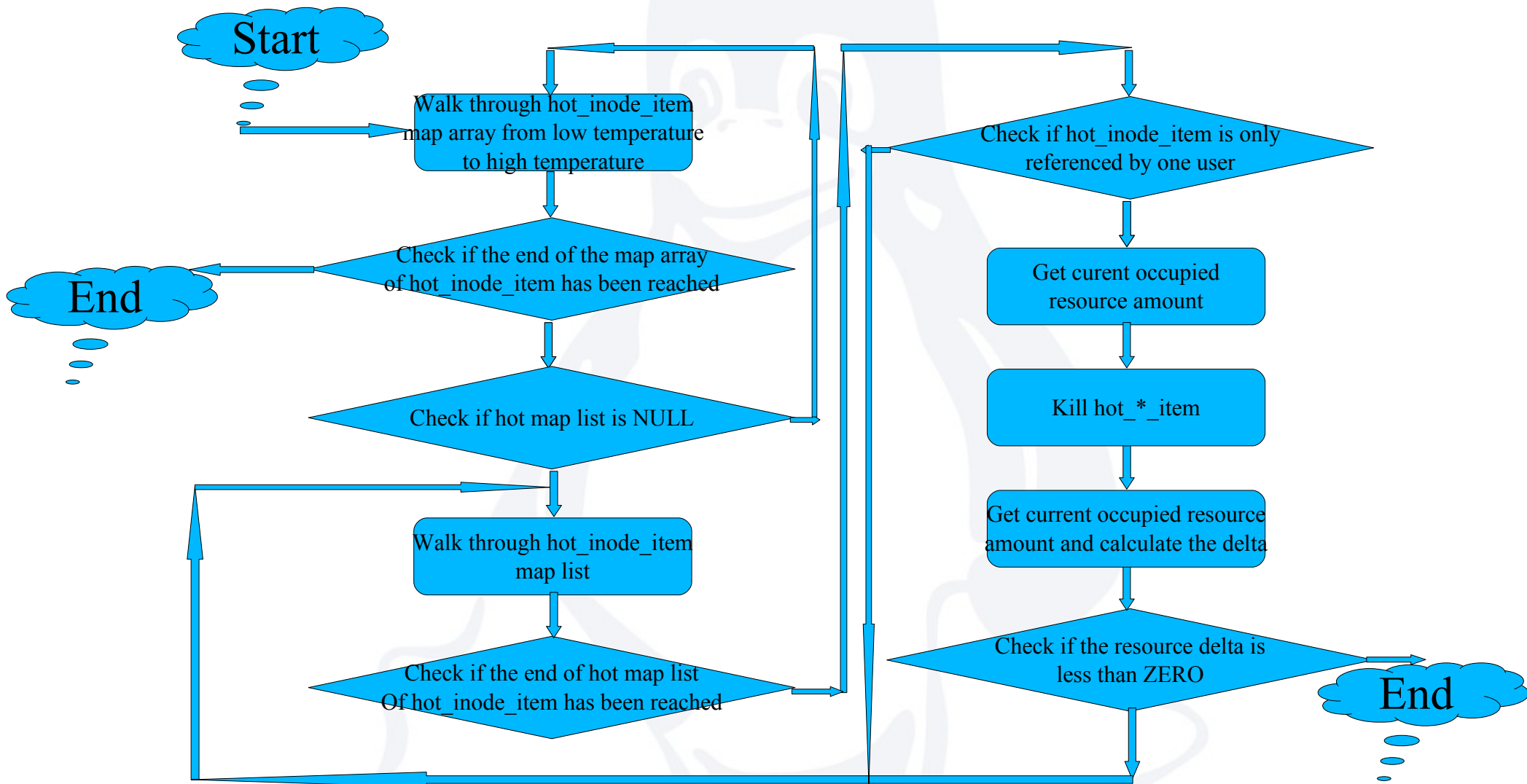
Record I/O access info



Update hot map periodically



Curtail cache by shrinker



Control cache by proc interface

- Approximately same as the shrinker
- Invoke the public interface
 - static unsigned long hot_item_evict(struct hot_info *root, unsigned long work, unsigned long (*work_get)(struct hot_info *root))
- Only passed work_get() is different



FFSB - large_file_create

	w/ hot tracking v1	w/o hot tracking v2	ratio (v1-v2)/v2
large_file_create			
1 thread			
- Trans/sec	28091.76	28126.31	-0.12%
- Throughput	110MB/sec	110MB/sec	+0.00%
- %CPU	10.7%	11.2%	-4.47%
- Trans/%CPU	2625.4	2511.28	-4.54%
8 threads			
- Trans/sec	27980.47	28140.34	-0.57%
- Throughput	109MB/sec	110MB/sec	-0.91%
- %CPU	12.3%	12.8%	-3.90%
- Trans/%CPU	2274.83	2198.46	+3.47%
16 threads			
- Trans/sec	27764.36	27940.96	-0.63%
- Throughput	108MB/sec	109MB/sec	-0.92%
- %CPU	12.8%	13.7%	-6.57%
- Trans/%CPU	2169.09	2039.49	+6.35%
32 threads			
- Trans/sec	27461.82	27624.48	-0.59%
- Throughput	107MB/sec	108MB/sec	-0.93%
- %CPU	13.7%	14.4%	-4.86%
- Trans/%CPU	2004.51	1918.37	+4.49%



FFSB - large_file_seq_read

	w/ hot tracking v1	w/o hot tracking v2	ratio (v1-v2)/v2
large_file_seq_read			
1 thread			
- Trans/sec	34121.46	34838.65	-2.06%
- Throughput	133MB/sec	136MB/sec	-2.21%
- %CPU	8.8%	8.8%	+0.00%
- Trans/%CPU	3877.44	3958.94	-2.06%
8 threads			
- Trans/sec	10883.15	11679.40	-6.82%
- Throughput	42.5MB/sec	45.6MB/sec	-6.80%
- %CPU	3.3%	3.4%	-2.94%
- Trans/%CPU	3297.92	3435.12	-3.99%
16 threads			
- Trans/sec	5760.16	6193.20	-6.99%
- Throughput	22.5MB/sec	24.2MB/sec	-7.02%
- %CPU	1.8%	1.9%	-5.26%
- Trans/%CPU	3200.09	3259.58	-1.83%
32 threads			
- Trans/sec	5470.50	5490.12	-0.36%
- Throughput	21.4MB/sec	21.4MB/sec	+0.00%
- %CPU	1.7%	1.7%	+0.00%
- Trans/%CPU	3217.94	3229.48	-0.36%



FFSB - random_write

	w/ hot tracking v1	w/o hot tracking v2	ratio (v1-v2)/v2
random_write			
1 thread			
- Trans/sec	1611.99	1582.57	+1.86%
- Throughput	220MB/sec	216MB/sec	+1.85%
- %CPU	0.6%	0.6%	+0.00%
- Trans/%CPU	2686.65	2637.62	+1.86%
8 threads			
- Trans/sec	2215.59	2292.57	-3.36%
- Throughput	303MB/sec	313MB/sec	-3.39%
- %CPU	1.4%	1.5%	-6.67%
- Trans/%CPU	1582.56	1528.38	+3.35%
16 threads			
- Trans/sec	2068.52	1935.96	+6.85%
- Throughput	283MB/sec	265MB/sec	+6.79%
- %CPU	1.3%	1.3%	+0.00%
- Trans/%CPU	1591.17	1464.8	+8.63%
32 threads			
- Trans/sec	1764.28	1875.23	-5.92%
- Throughput	241MB/sec	256MB/sec	-5.86%
- %CPU	1.2%	1.3%	-7.69%
- Trans/%CPU	1470.23	1442.48	+1.92%



FFSB - random_read

	w/ hot tracking v1	w/o hot tracking v2	ratio (v1-v2)/v2
random_read			
1 thread			
- Trans/sec	222.84	224.28	-0.64%
- Throughput	891KB/sec	897KB/sec	-0.67%
- %CPU	1.1%	1.0%	+10.0%
- Trans/%CPU	202.58	224.28	-9.68%
8 threads			
- Trans/sec	143.30	136.47	+5.01%
- Throughput	573KB/sec	546KB/sec	+4.95%
- %CPU	0.5%	0.5%	+0.00%
- Trans/%CPU	286.60	272.94	+5.01%
16 threads			
- Trans/sec	105.17	103.75	+1.37%
- Throughput	421KB/sec	415KB/sec	+1.45%
- %CPU	0.5%	0.5%	+0.00%
- Trans/%CPU	210.34	207.5	+1.37%
32 threads			
- Trans/sec	105.78	103.39	+2.31%
- Throughput	423KB/sec	414KB/sec	+2.17%
- %CPU	0.5%	0.5%	+0.00%
- Trans/%CPU	211.56	206.78	+2.31%



FFSB - mail_server

		w/ hot tracking v1	w/o hot tracking v2	ratio (v1-v2)/v2
mail_server				
1 thread				
-	Trans/sec [read]	433.23	446.68	-3.01%
-	Throughput [read]	1.7MB/sec	1.75MB/sec	-2.86%
-	Trans/sec [write]	224.06	213.84	+4.78%
-	Throughput [write]	889KB/sec	848KB/sec	+4.83%
-	%CPU	0.8%	0.8%	+0.00%
-	Trans/%CPU [read]	541.54	558.35	-3.01%
-	Trans/%CPU [write]	280.08	267.3	+4.78%
8 threads				
-	Trans/sec [read]	430.47	435.84	-1.23%
-	Throughput [read]	1.69MB/sec	1.71MB/sec	-1.17%
-	Trans/sec [write]	198.18	207.61	-4.54%
-	Throughput [write]	786KB/sec	823KB/sec	-4.50%
-	%CPU	0.9%	0.9%	+0.00%
-	Trans/%CPU [read]	478.3	484.27	-1.23%
-	Trans/%CPU [write]	220.2	230.68	-4.54%
16 threads				
-	Trans/sec [read]	326.05	347.85	-6.27%
-	Throughput [read]	1.28MB/sec	1.37MB/sec	-6.57%
-	Trans/sec [write]	187.69	177.59	+5.69%
-	Throughput [write]	744KB/sec	705KB/sec	+5.53%
-	%CPU	0.9%	0.9%	+0.00%
-	Trans/%CPU [read]	362.28	386.5	-6.27%
-	Trans/%CPU [write]	208.54	197.2	+5.75%
32 threads				
-	Trans/sec [read]	388.04	419.52	-7.50%
-	Throughput [read]	1.53MB/sec	1.65MB/sec	-7.27%
-	Trans/sec [write]	204.70	207.50	-1.35%
-	Throughput [write]	811KB/sec	823KB/sec	-1.46%
-	%CPU	1.2%	1.2%	+0.00%
-	Trans/%CPU [read]	323.37	349.6	-7.50%
-	Trans/%CPU [write]	170.58	172.92	-1.35%



Status & Next

- The patchset is currently reviewed by VFS maintainer
 - Focus on performance
- Latest patchset
 - `git://github.com/wuzhy/kernel.git hot_tracking`
- The debugfs support
 - Lively dump hot information
- Btrfs hot relocation support
 - `git://github.com/wuzhy/kernel.git hot_reloc`
- XFS/BTRFS Defragment improvement
 - Based on hot information



References

- The lwn editor's article
 - <http://lwn.net/Articles/525651/>
- An intro in-kernel document
 - Documentation/filesystems/hot_tracking.txt
- Mingming Cao's slides
 - http://www.linuxplumbersconf.org/2010/ocw/system/presentations/219/original/hot_cold_data_LPC.pdf



Thank you!

Questions?

