Journey to the Bottom of the Storage Stack

Tracing, Visualizing, and Debugging File System Behavior

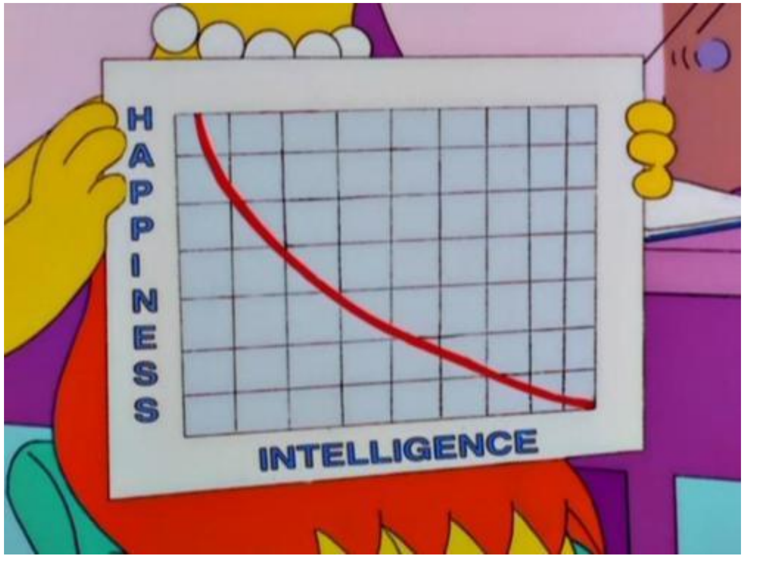
Mark Gritter -- @markgritter

Tintri, Inc. -- @TintriInc

#minnebar

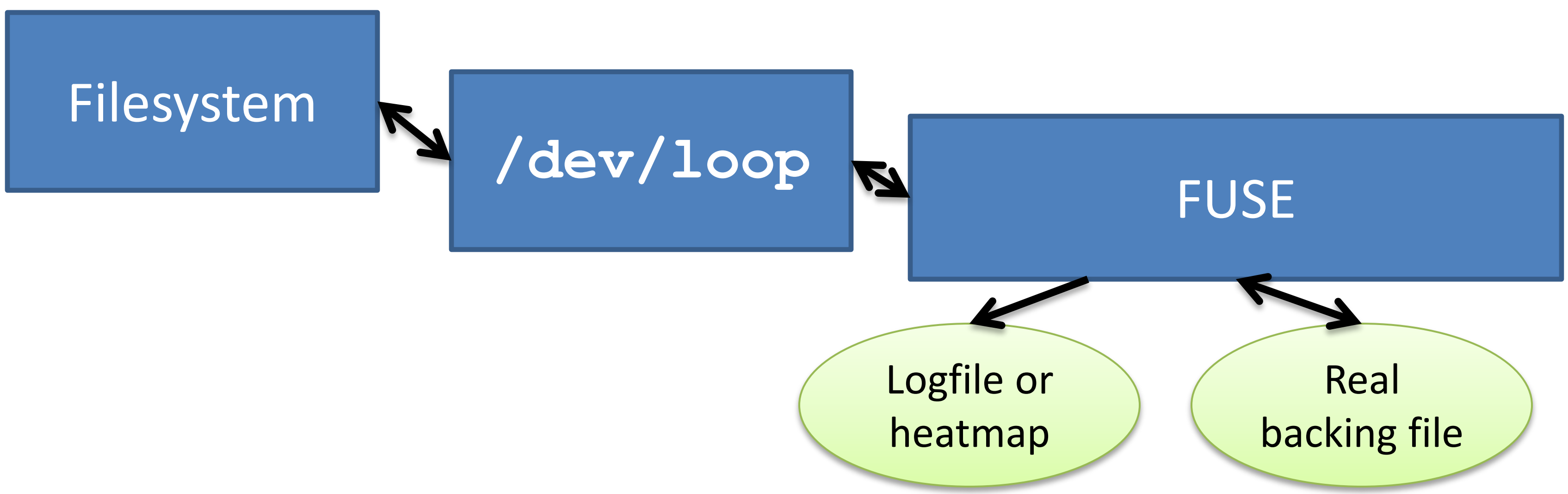
20,000 I/Os Under the File System

- What does application behavior end up looking like, at the level of a block device?
- What can we understand about system behavior from block-level operations?
- “Medium-to-big” data
- Lots of graphs



Tracing

- Linux loopback device exposes file as block device (so we can mount it)
- Could put the file on a FUSE file system to monitor read/write activity.



Or… get the OS to help out

- If you have DTrace, tracing becomes a lot easier.
  - Probe exists to dump all SCSI commands
- On Linux, systemtap has a similar probe

Or… virtualize It!

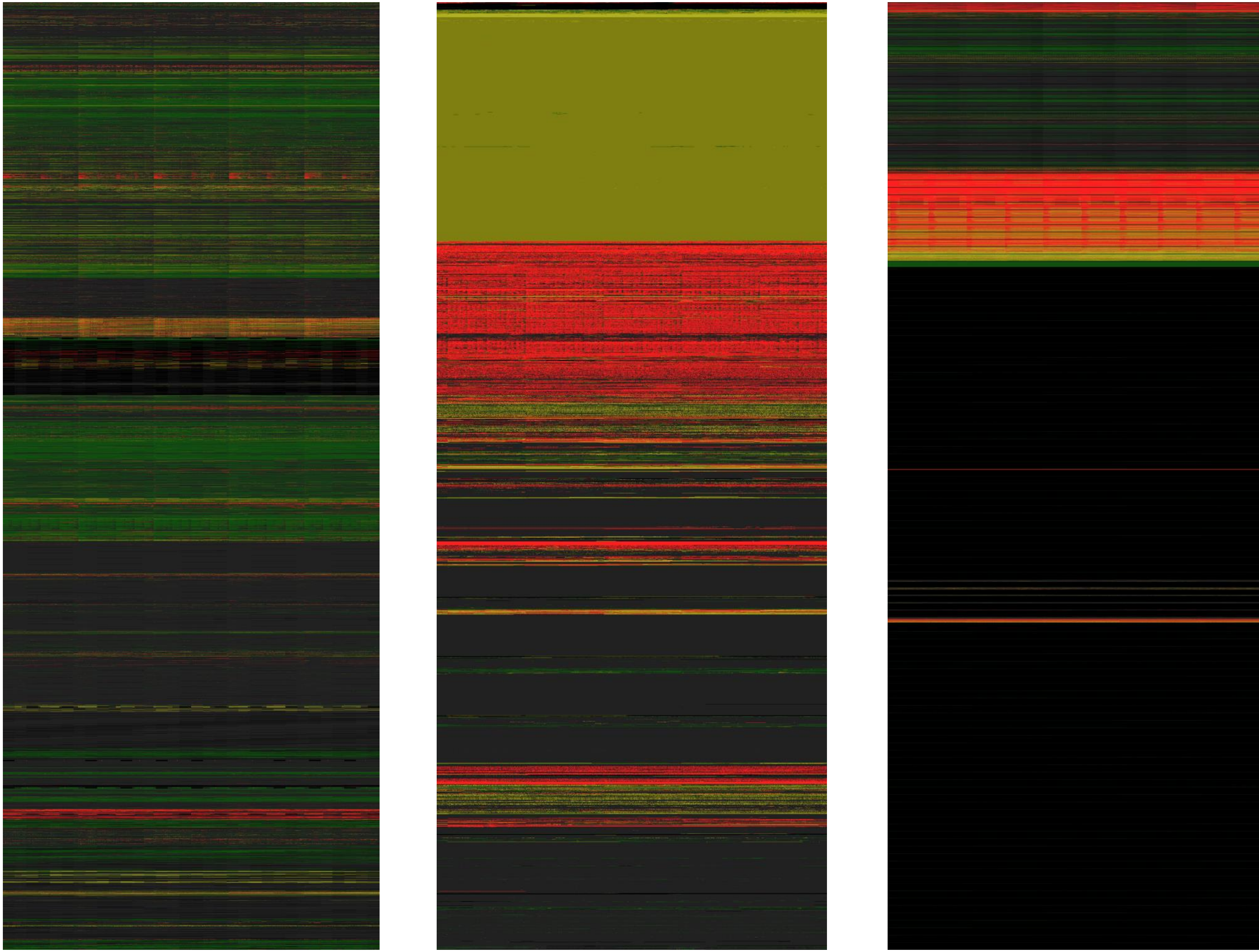
- Vmware Workstation as a ‘VProbes’ feature that lets you log any kernel function…
  - Like the one that issues SCSI block requests
- …and backs block I/O with a file
  - Can look at traffic over NFS, or trace via other mechanisms described here
  - or have a storage appliance that already has these features built in!
- Virtualization helps analysis by being able to attach a clone of a virtual disk.

Example data in this talk

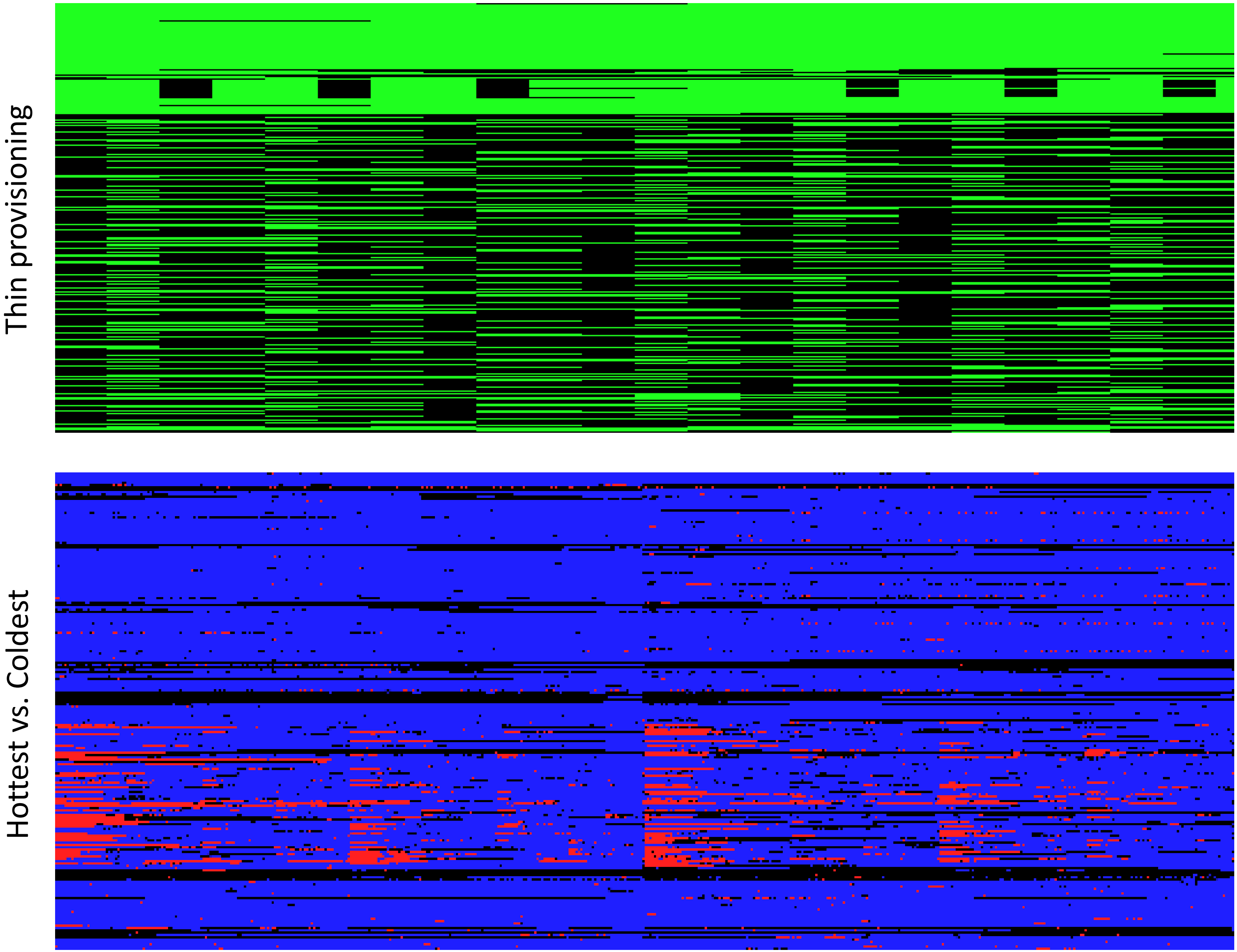
- Mark’s development VM
  - build working area
  - system disk
- QA database server (Oracle)

Heatmaps

- For every 8KB block in the system, plot the number of access counts (both read + write)
  - On Tintri systems, we also get # of times that data block is shared (deduped)
- “Static” view of the system, long-term trends visible



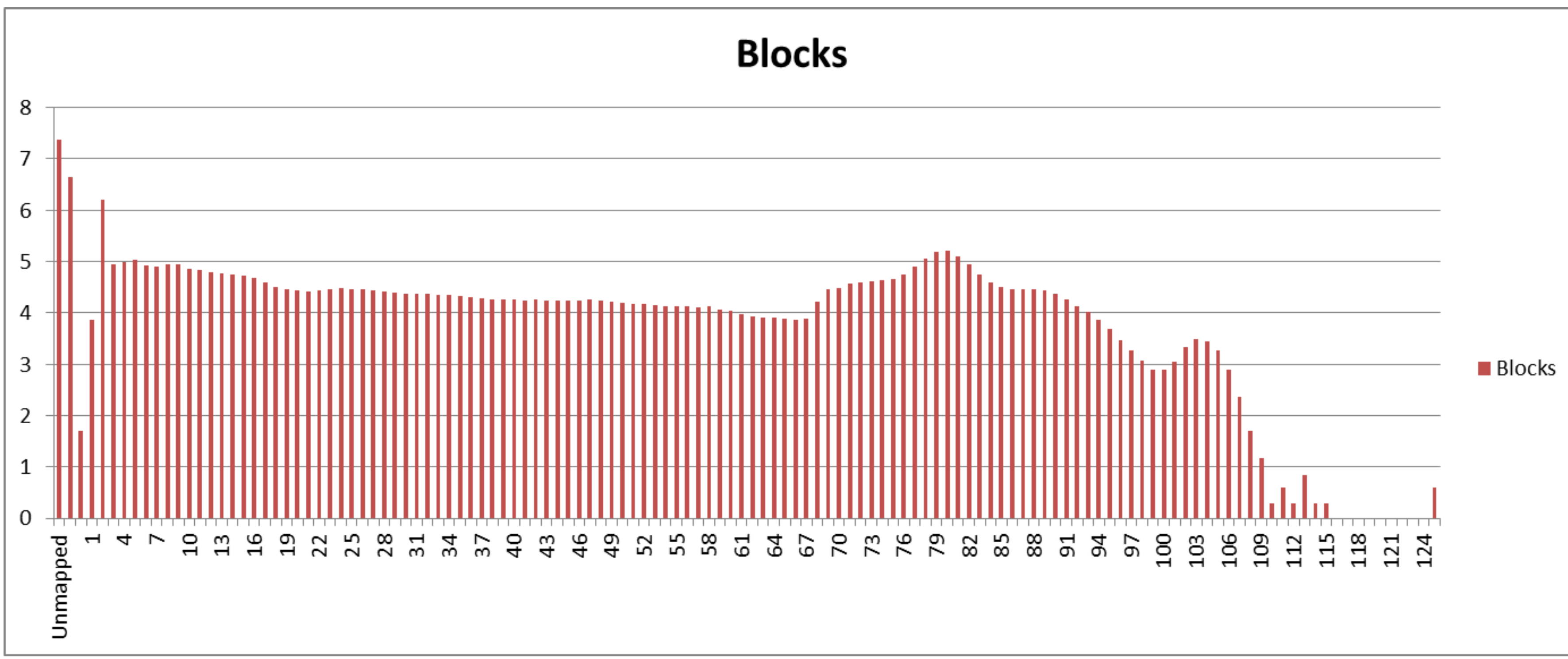
working system DB

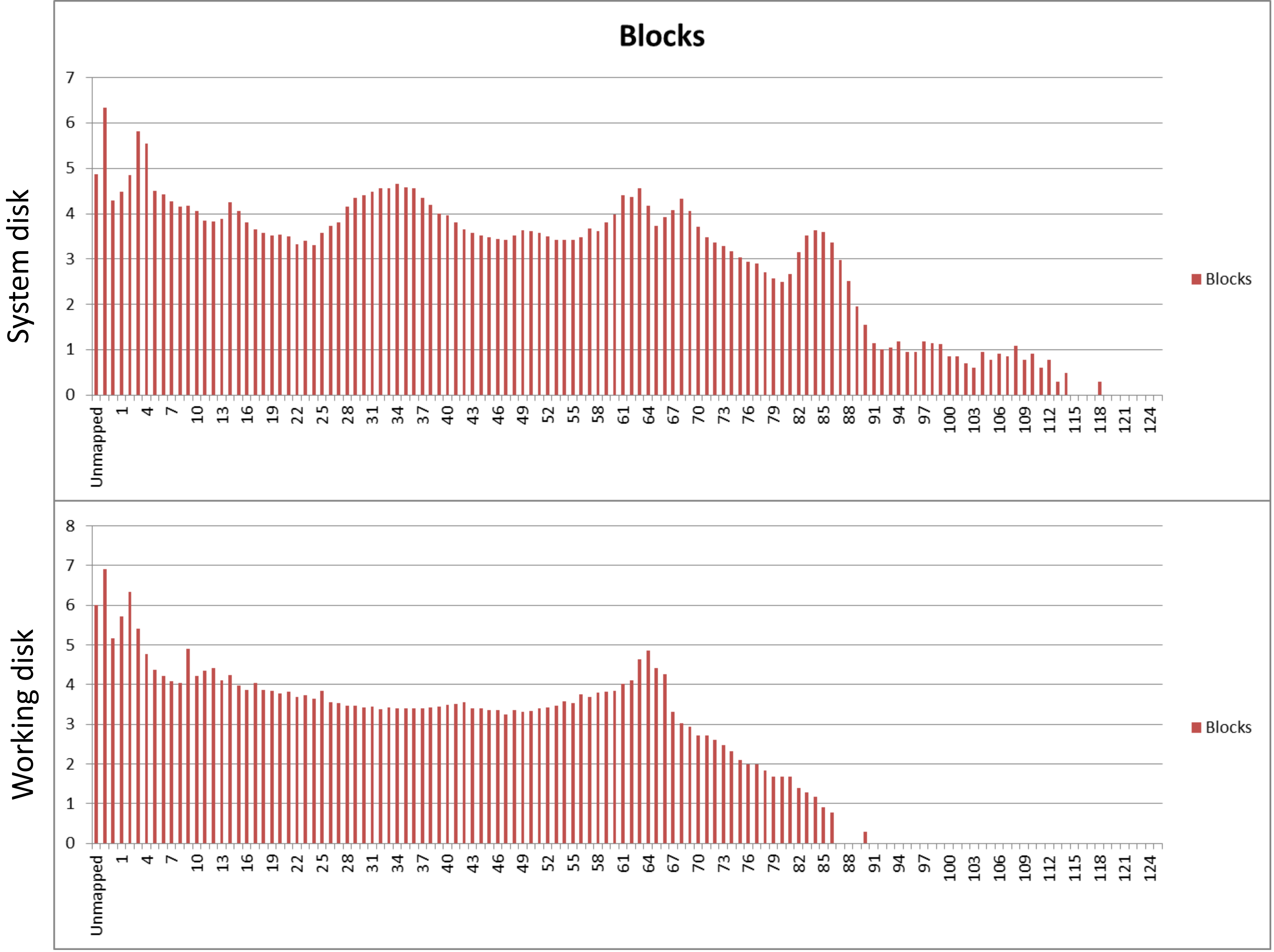

Thin provisioning
Hottest vs. Coldest

Histograms

- Log (base 10) scale to make the data visible.

Database





Blocks
System disk
Working disk
Blocks
Unmapped

2-D Histograms

Database

		Access Counts									
		0	1	2	3-4	5-7	8-11	12-16	17-32	33-64	65-128
Reference Counts	Disk	4452654									
	1	39	6018	102977	81300	209048	256159	228003	383231	384597	716726
	2	3	1264	223822	51085	63094	64032	49527	62593	115868	701177
	3-4	0	22	50517	8961	841	544	624	1280	1497	4046
	5-8	0	43	85959	4878	853	379	231	393	270	872
	9-16	0	47	19974	4330	779	102	35	167	66	132
	17-32	0	2	111230	3275	659	111	58	58	84	219
	33-64	0	0	242189	10144	195	17	29	57	9	14
	65-128	8	63	234196	9845	17	23	74	64	2	21
	129-256	0	1	408090	10684	30	27	42	16	14	18
	257-506	0	31	119684	2187	142	99	93	122	26	15

	Cold	Hot
Low dedupe	54.7%	31.2%
High dedupe	14.0%	0.1%

Lots of hot, undedupable data --- unique entries
In the database.

System

		Access Counts									
		0	1	2	3-4	5-7	8-11	12-16	17-32	33-64	65-128
Reference Counts	Disk	2149496									
	1	125	11555	13017	137308	7852	4030	12713	6783	18230	5687
	2	156	1847	8470	849045	19678	1475	18324	9316	9983	18884
	3-4	581	4597	9735	3634	1844	1116	1834	14670	16277	3034
	5-8	1093	2658	11270	4865	4775	3260	1399	54241	91849	14802
	9-16	494	3208	14572	14467	29597	17010	3912	63929	179983	29388
	17-32	444	413	6945	3076	8896	14498	5843	9140	49222	18850
	33-64	512	378	2082	898	1179	1717	1391	5515	12314	820
	65-128	571	379	1583	1386	1996	2194	1700	2180	9182	448
	129-256	13619	4185	1269	834	1086	1809	1131	1703	4814	666
	257-506	1802	976	951	602	1031	970	1265	1378	898	671

	Cold	Hot
Low dedupe	77.7%	2.6%
High dedupe	4.0%	15.7%

Hot, deduped data--- common OS files!

Working

		Access Counts									
		0	1	2	3-4	5-7	8-11	12-16	17-32	33-64	65-128
Reference Counts	Disk	8322146									
	1	97	6699	948768	58070	12915	26530	35018	28885	61403	28662
	2	32	3440	644721	46320	5420	9207	8972	12374	19196	8647
	3-4	149	4546	200171	36418	4870	3824	5413	6859	13087	2806
	5-8	893	26645	118554	21653	4726	4620	4743	6702	14583	2822
	9-16	2213	30958	59795	27306	6235	7391	6067	8417	15925	1895
	17-32	5981	10600	58851	35075	5933	4452	3611	6422	13292	1758
	33-64	3095	9589	37123	48271	3431	4498	4759	4500	9786	1666
	65-128	3177	11520	37956	12548	2815	5077	2371	2772	8712	1229
	129-256	79887	295579	26620	23251	4787	62763	658	2225	60866	875
	257-506	48310	136087	8048	1314	1047	722	717	1399	15218	389

	Cold	Hot
Low dedupe	83.3%	2.0%
High dedupe	12.1%	2.7%

Cold data, in a mixed of deduped (source files) and undeduped (builds?)

What’s in those hot blocks?

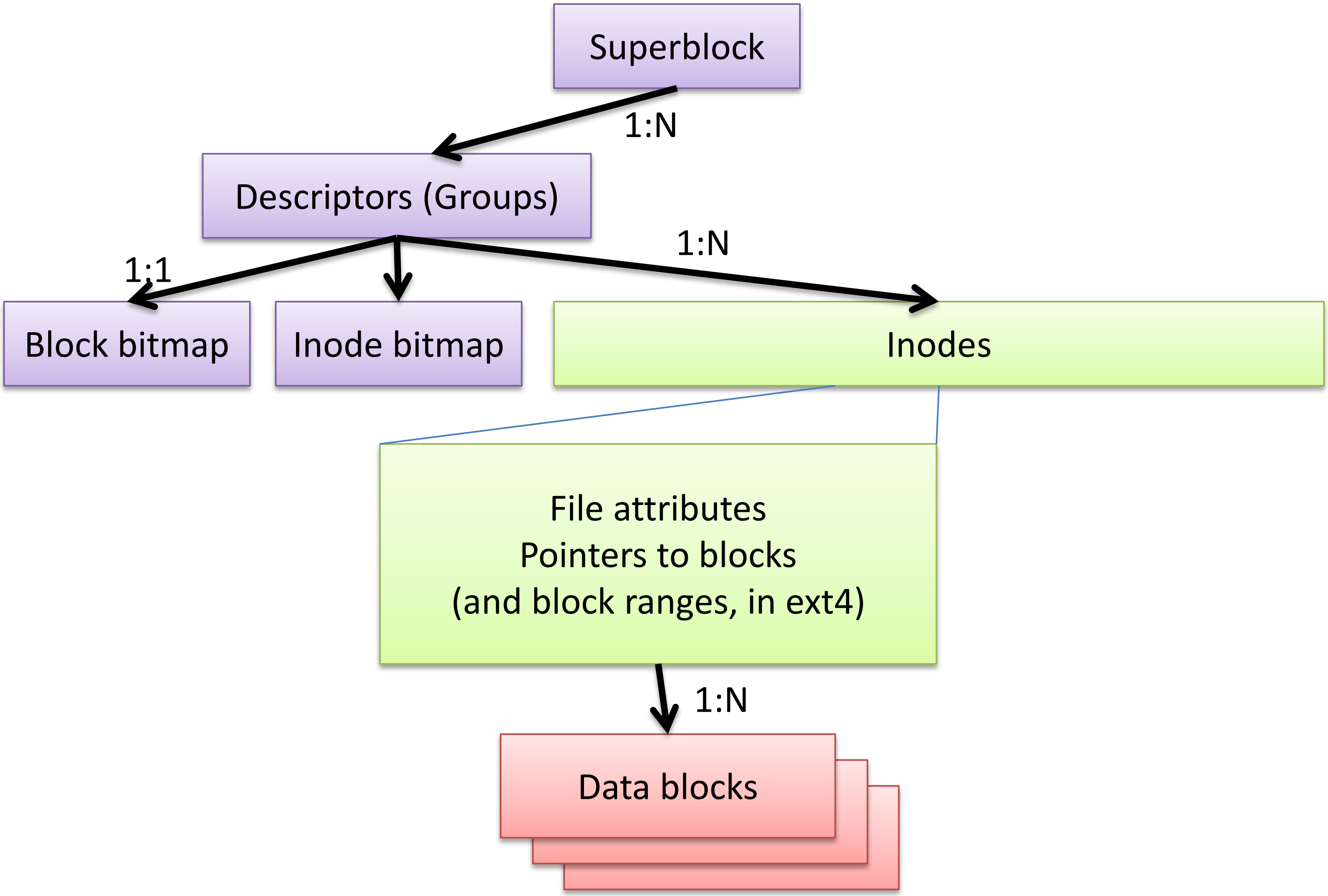
- Can we tell by examining the content?
- One experiment --- run “file” utility on top 1% of hot blocks (of build disk)
- Some right answers, but a lot of strange ones

data	60.1%
ISO-8859 text	27.4%
raw G3 data	2.9%
ASCII C++ program text	1.4%
ASCII C program text	1.2%
ASCII English text	1.0%
PCX ver. 2.5 image data	0.9%
ASCII text	0.8%
ACB archive data	0.3%
DOS executable (COM)	0.3%
PDP-11 UNIX/RT ldp	0.3%
ASCII assembler program text	0.2%
ASCII Pascal program text	0.2%
8086 relocatable (Microsoft)	0.1%
ASCII Java program text	0.1%
DOS executable (device driver)	0.1%
COM executable for DOS	0.1%
X11 SNF font data	0.1%
DBase 3 data file	0.1%
DBase 3 index file	0.1%
ELF 64-bit LSB relocatable	0.1%

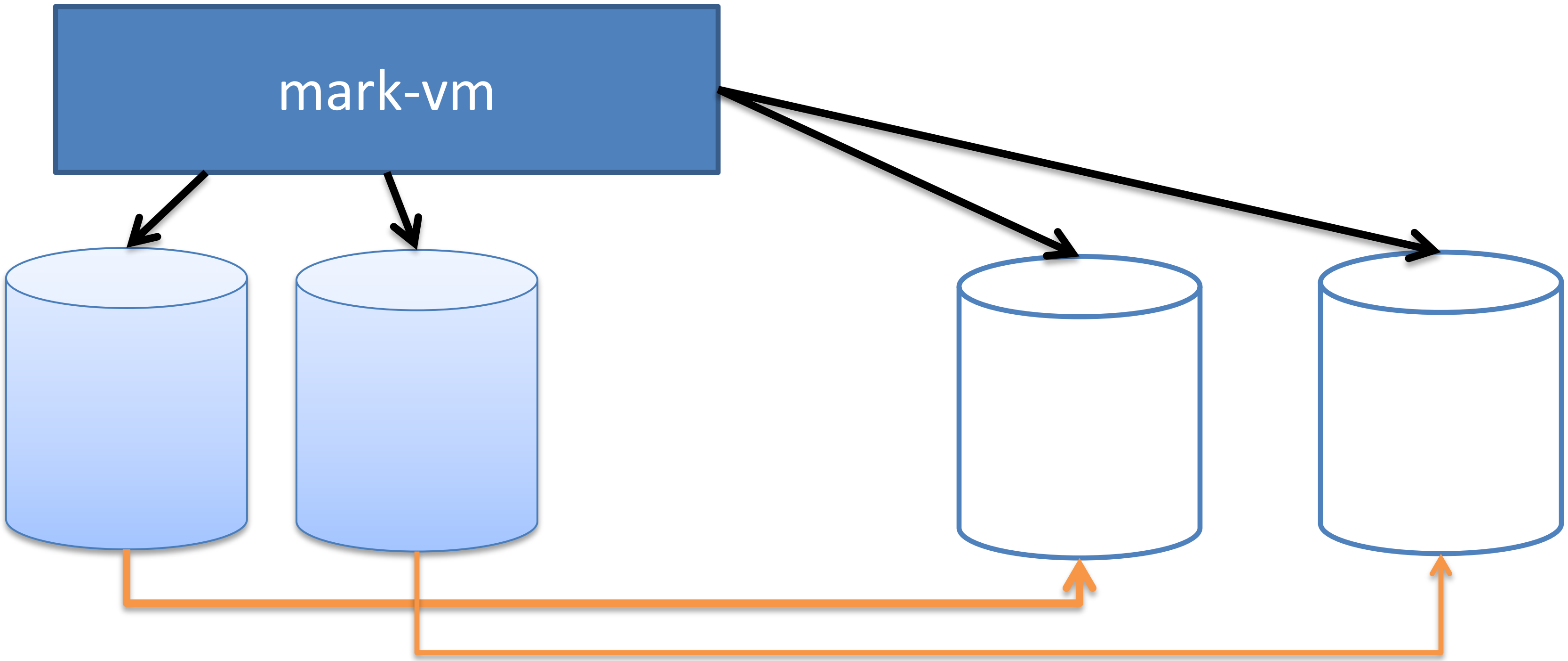
Detour: ext4 debugger

- "debugfs" program lets you interactively explore an ext2/ext3/ext4 file system
  - `stats`: show superblock information
  - `icheck`: search for references to a data block, within inodes
  - `ncheck`: search for names (directory entries) for an inode
  - `testb`: check whether a data block is free

EXTn structure



Experimental Setup



Capture heatmap of real disks, then clone them…

then attach the new disks to the original VM, and run debugfs against the clones.

Which files contain the hot blocks?

(again, just looking at top 1%)

- Much more useful! Analyzing system disk shows us:
  - File system structures
  - Mark's favorite editor
  - Libraries frequently used

.ext-journal	39.8%
free	15.3%
inode	6.3%
/var/cache/yum/x86_64/12/fedora/4b	3.2%
/usr/bin/emacs-23.1	2.7%
/var/cache/yum/x86_64/12/updates/5	2.3%
/var/cache/yum/x86_64/12/updates/a	1.9%
/var/cache/yum/x86_64/12/fedora/5c	1.3%
/var/lib/rpm/Packages	1.2%
/usr/bin/postgres	0.6%
/lib64/libc-2.11.2.so	0.4%
/usr/lib64/libcrypto.so.1.0.0b	0.4%
/var/log/tintri/support.log	0.3%
/opt/build-tools/.hg/store/data/apach	0.3%
/lib64/libdb-4.7.so	0.3%
/usr/lib64/libnss3.so	0.2%
unknown	0.2%
/lib64/libglib-2.0.so.0.2200.5	0.2%
/var/log/audit/audit.log.1	0.2%

- Breakdown by extension more useful when there are lots of files…
  - Shared libraries
  - Log files
  - Can look at inode to identify programs with executable bit set

.ext-journal	39.8%
free	15.3%
other	12.5%
.sqlite	8.8%
.so	6.8%
inode	6.3%
.1	*2.7%*
.jar.d	1.1%
.html	0.6%
.log	0.5%
.pyo	0.3%
.swc.d	0.3%
executable	0.3%
.mo	0.3%
.jar	0.2%
.d	0.2%
unknown	0.2%
.log.1	0.2%
.html.i	0.2%
.py	0.2%
.linux.d	0.2%
.pyc	0.1%
.cache	0.1%
.jar.i	0.1%
.4.jar.d	0.1%

Exploring the obvious:

Working set of builds has object files and source files, but 53% of the hottest blocks are free!

free	53.4%
executable	28.6%
.o	5.1%
.ko	1.9%
inode	1.1%
.c	1.0%
.c.i	0.7%
.so	0.6%
.cpp	0.6%
.h	0.5%
.i	0.3%
.py	0.3%
directory	0.3%
.cgs.i	0.3%
.S	0.3%
.cgs	0.3%
.java	0.3%
.a	0.3%
.java.i	0.3%
.h.i	0.3%
.exp	0.2%
.d	0.2%
.cpp.d	0.2%
.exp.i	0.2%
unknown	0.2%

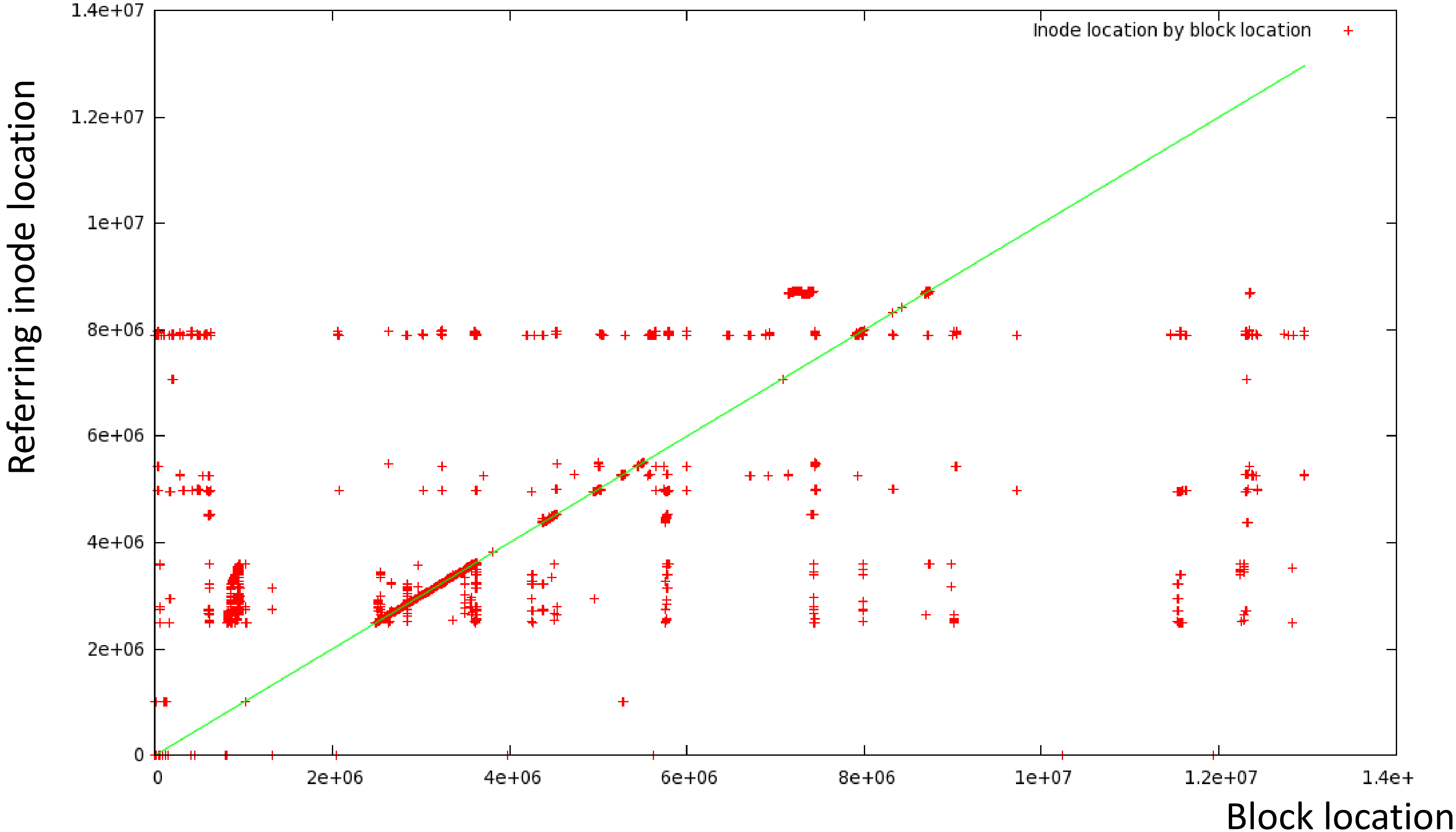
Working set of database server is… database files. The temporary database dominates.

/oradata/orcl/temp01.dbf	41.5%
/oradata/orcl/undotbs01.dbf	17.4%
/oradata/orcl/qavcsa1	17.0%
/oradata/orcl/redo02.log	6.0%
/oradata/orcl/redo03.log	6.0%
/oradata/orcl/redo01.log	5.9%
EXT-JOURNAL	4.0%
/oradata/orcl/sysaux01.dbf	0.8%
/product/11.2.0/dbhome_1/lib/libclntsh.so.11.1	0.2%
/oradata/orcl/system01.dbf	0.2%
/product/11.2.0/dbhome_1/jdk/jre/lib/amd64/server/libjvm.so	0.1%
/diag/rdbms/orcl/orcl/trace/orcl_dbrm_1730.trm	0.1%
inode	0.1%
free	0.1%
/product/11.2.0/dbhome_1/lib/libnnz11.so	0.1%
/flash_recovery_area/orcl/control02.ctl	0.1%
/oradata/orcl/control01.ctl	0.1%
/product/11.2.0/dbhome_1/bin/oracle	0.0%

(I'm told that this happens if you don't configure temporary space per database.)

Does ext4 place data and metadata together on disk?

(build disk)



Not really (but sometimes!)

Does data block hotness predict metadata hotness?

	Colder data									Hotter data	
Cold	99058	0	124	52	410	1070	1401	1024	1998	2496	16
metadata	43	0	0	0	0	0	0	0	0	0	0
	2141	0	15	1	1	0	0	0	0	10	0
	6945	0	21	17	2	4	17	4	44	86	0
	96867	2	1149	62371	3174	467	980	320	563	2009	3579
	107117	185	2842	94040	5712	2126	1794	1103	3003	11632	3114
	77686	22	21612	159151	4303	2074	2696	1908	5390	10785	2583
	57421	104	4498	40505	3180	3130	3461	3384	7187	7681	1351
	87058	160	8852	31448	4848	7132	5949	5123	10453	20753	5297
Hot	57502	16	4441	51987	2862	4654	4631	2278	1972	9373	6185
metdata	60250	688	2990	5099	1855	1322	1257	2014	2615	3170	2542

Not really, in fact why do hot blocks with cold metadata even exist?
What is the appropriate statistical test here?

Traces

- For every read and write, record the location and time.
- Dynamic view of the file system (+larger volume of data!)

compiling “Hello, World!”

```
 0.264248 R         0+0000  superblock
 0.265235 R         1+0000  descriptor table
 0.267531 R      1057+0000  inodes 1-16
 0.268537 R   2129920+0000  journal read, in inode 8
 0.270083 W         0+0000  superblock
 1.298729 R      9249+0000  root directory, in inode 2 (block 1057)
10.742560 W   2129920+0000  journal
10.743471 W   2129921+0000  journal
10.744055 W   2129923+0000  journal
26.853277 R     33793+0000  helloworld.c, in inode 12 (only 92 byte!)
27.094870 R      1041+0000  inode bitmap
32.772513 W   2129924+0000  journal
32.773320 W   2129929+0000  journal
34.684074 W         1+0000  descriptor table
34.685649 R      1026+0000  block bitmap
34.686604 W      1041+0000  inode bitmap
34.687324 W      1057+0000  inode 1-16
34.688026 W      9249+0000  root directory (in inode 2)
34.689338 W     33794+0000  helloworld (in inode 13)
34.689454 W   2129930+0000  journal
34.690194 W   2129934+0000  journal
34.783723 W   2129935+0000  journal
34.784501 W         1+0000  descriptor table
34.784621 W      1057+0000  inodes 1-16
34.784625 W      1026+0000  block bitmap
```

Mounting the file system

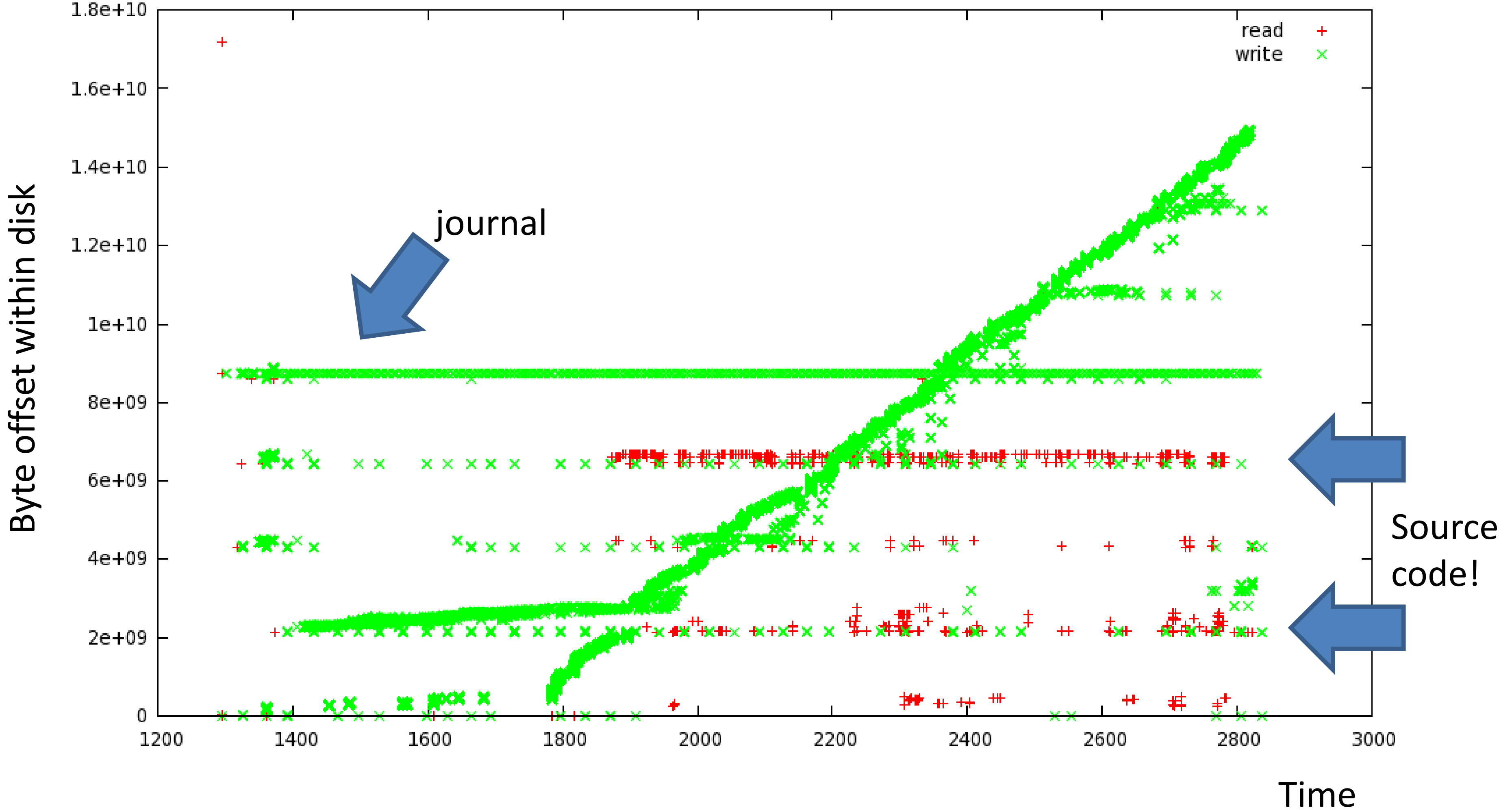
New version of inode 13 found in this portion of the journal

Things to do with a trace

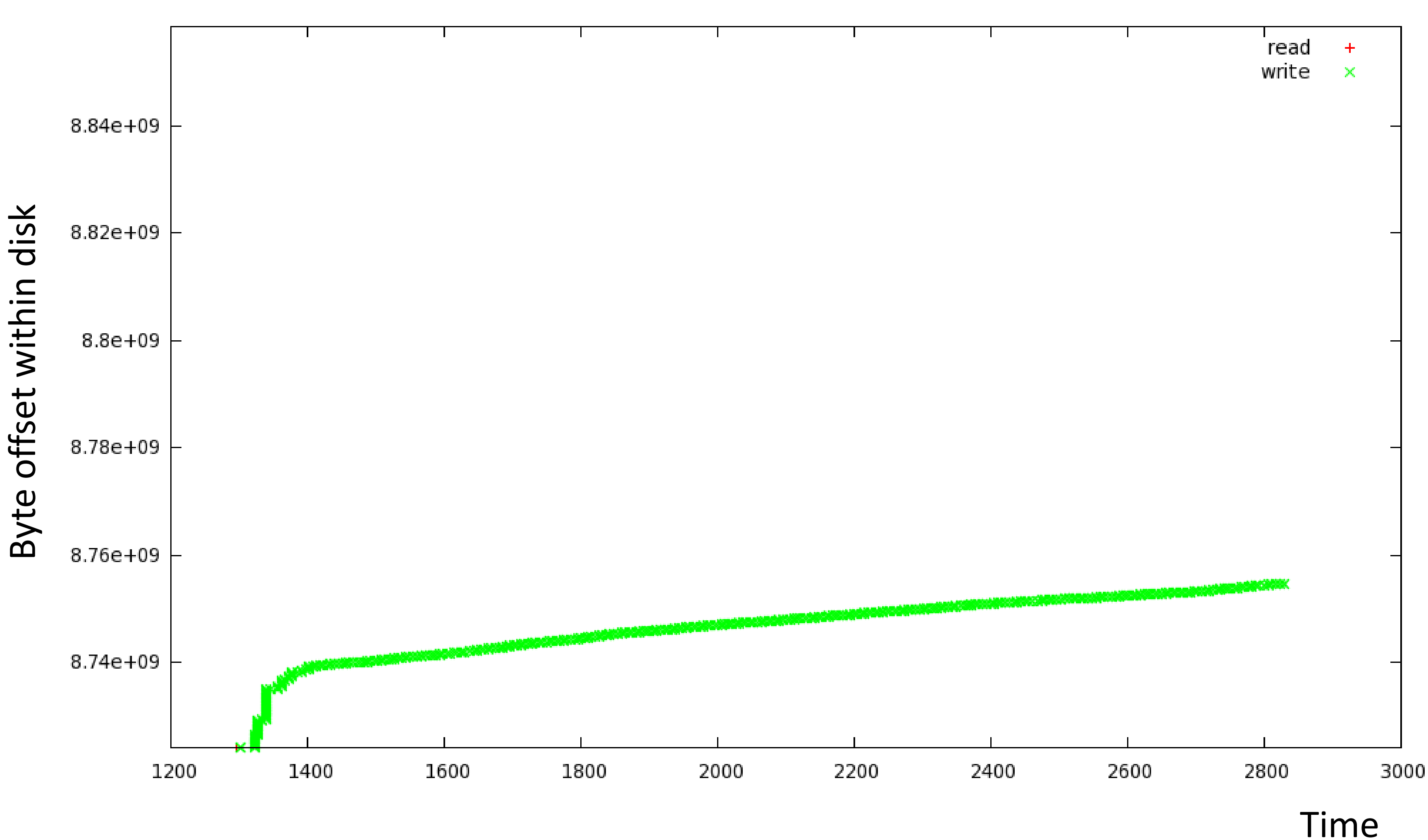
- Working-set analysis
- Heatmaps (again)
- Queue depth--- how many concurrent I/Os does an application generate? (need latency measurements)
- Look for redundant I/Os
- Read/write mix

Compiling a filesystem!

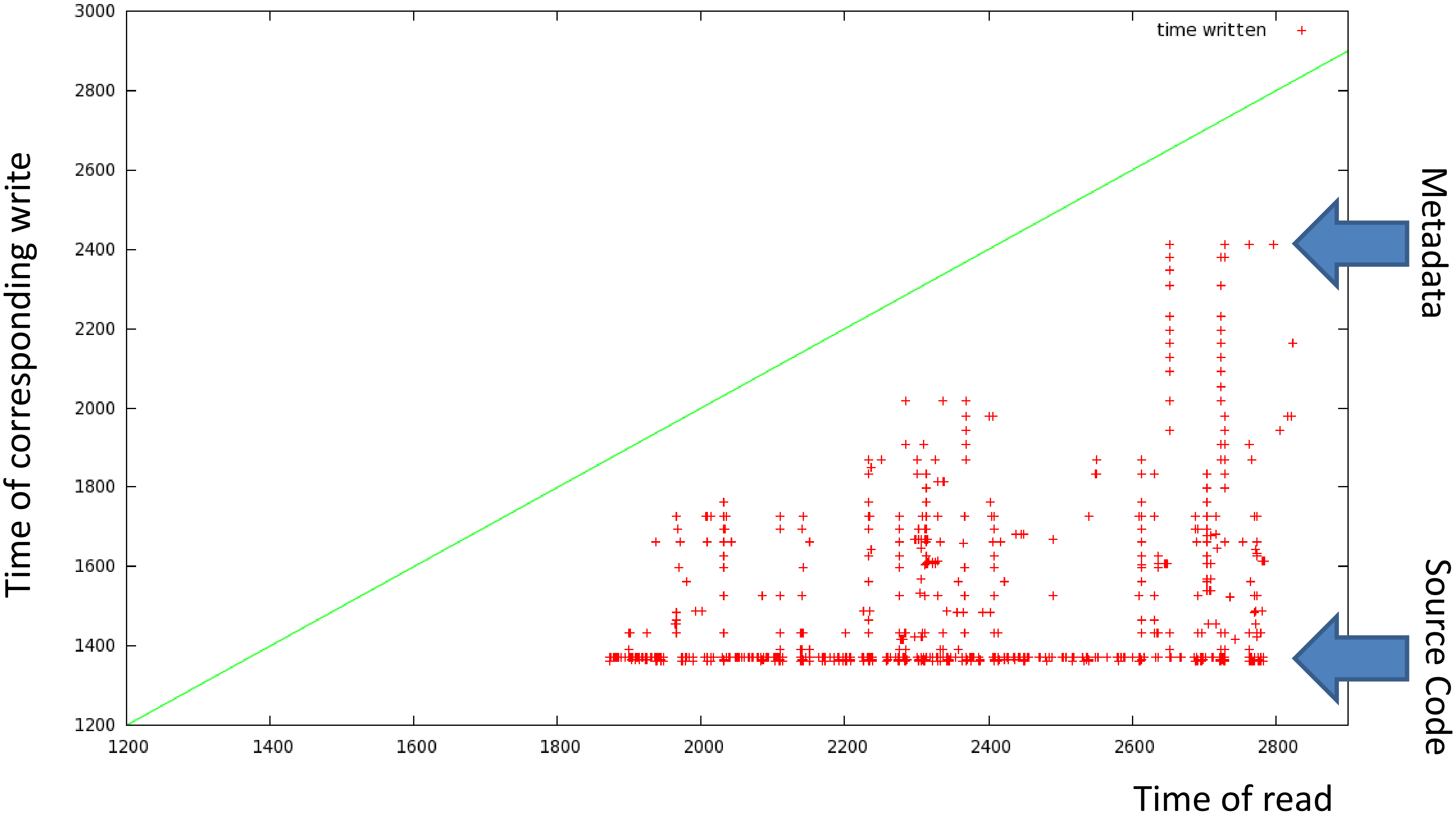
(on a fresh ext4 volume)



Writes to the ext4 journal



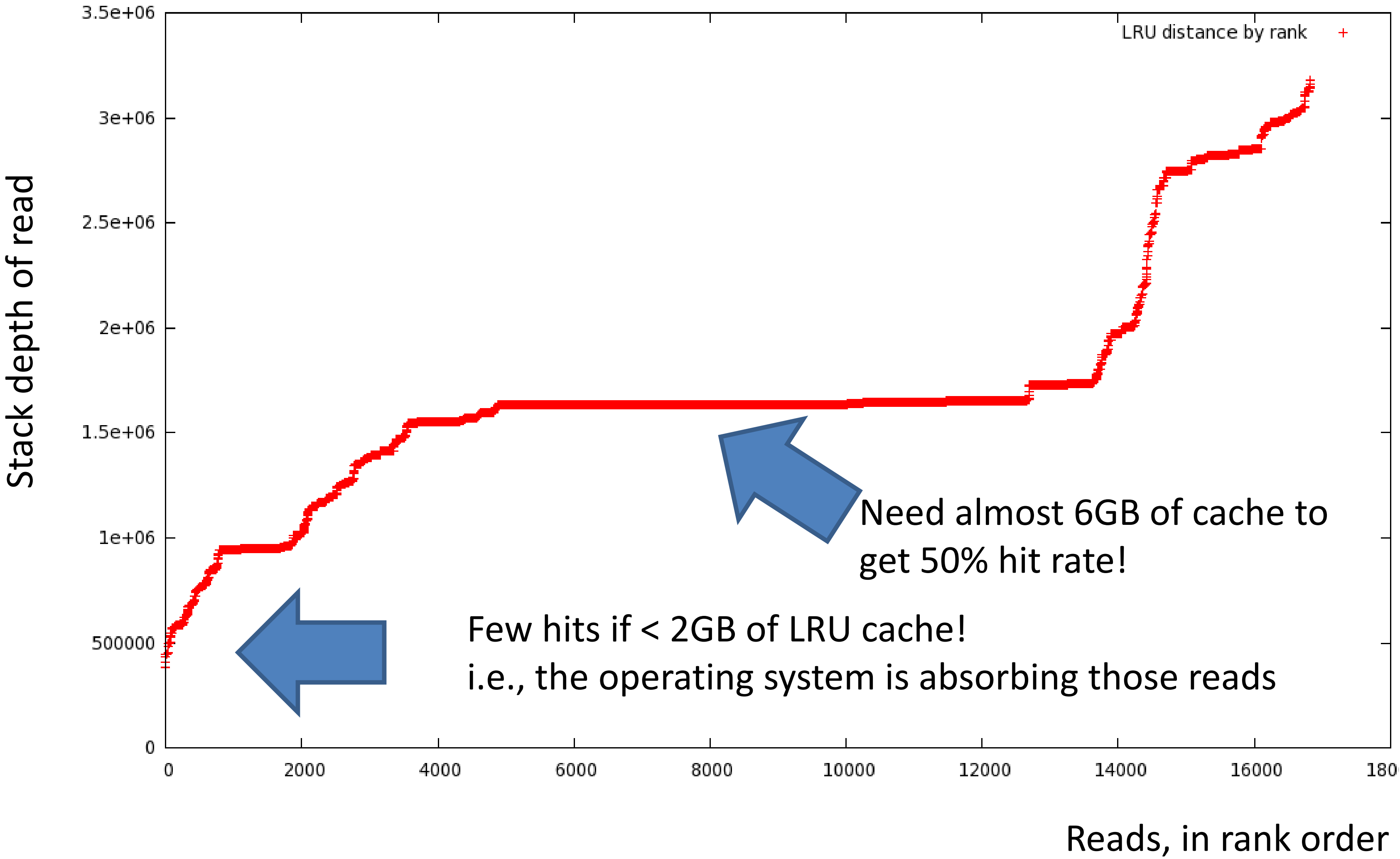
How old is the data being read?



LRU stack model

- When a block is read or written, record its depth in the stack, then pop it to the top
  - For an LRU cache of size N blocks:
    - Depth <= N is a cache hit
    - Depth > N is a cache miss
- Next graph shows stack depths for reads, sorted by rank

LRU cache analysis



What does an SQL INSERT look like?

(in a fresh Postgres database)

```
210.155231 W   2658076+0000 free
210.155450 W   2659476+0000 in inode 656051, /sql/global/pgstat.stat
214.111312 W     36124+0000 in inode 655380, /sql/pg_xlog/000000010000000000
214.111578 W     36125+0000 in inode 655380
215.540158 W   2132355+0000 journal
215.541465 W   2132356+0000 journal
215.542217 W   2132367+0000 journal
221.537726 W   2658070+0000 in inode 656259, /sql/base/16385/16386
221.537983 W   2621518+0000 inodes, group 80
221.537984 W   2621515+0000 inodes, group 80
221.537993 W   2659506+0000 in inode 656262, /sql/base/16385/16392
221.538003 W         0+0000
221.538079 W   2658071+0000 in inode 656259, /sql/base/16385/16386
221.538083 W   2621472+0000 inodes, group 80
221.538086 W   2621456+0000 inode bitmap 80
221.538088 W   2621520+0000 inodes, group 80
221.538731 W   2629677+0000 in inode 655374, /sql/pg_stat_tmp
221.539004 W   2132368+0000 journal
221.539586 W   2132372+0000 journal
221.590715 W         1+0000
221.590833 W   2621441+0000 block bitmap 81
221.590934 W   2621528+0000 inodes, group 80
```

Transaction log

DB page

Index?

SQL Insert (cont.)

INSERT INTO messages (id, content, time) VALUES (1, 'Hello, World!', '2013-01-01 01:01:01');

First Log write (4KB)

```
08d1c000  63 d0 01 00 01 00 00 00  00 00 00 00 00 c0 51 00  |c.............Q.|
08d1c010  7d 00 00 00 6f 6e 73 74  6c 65 6e 20 34 20 3a 63  |}...onstlen 4 :c|
08d1c020  6f 6e 73 74 62 79 76 61  6c 20 74 72 75 65 20 3a  |onstbyval true :|
08d1c030  63 6f 6e 73 74 69 73 6e  75 6c 6c 20 66 61 6c 73  |constisnull fals|
08d1c040  65 20 3a 6c 6f 63 61 74  69 6f 6e 20 35 33 33 35  |e :location 5335|
[snip]
08d1c780  02 40 00 00 00 00 00 00  01 00 b4 02 00 03 00 02  |.@..............|
08d1c790  08 18 00 01 00 00 00 1d  48 65 6c 6c 6f 2c 20 57  |........Hello, W|
08d1c7a0  6f 72 6c 64 21 00 00 00  00 00 00 40 4d 96 6e 2e  |orld!......@M.n.|
08d1c7b0  75 01 00 00 00 00 00 00  11 c2 75 d7 00 00 00 00  |u.........u.....|
08d1c7c0  58 c7 51 00 93 02 00 00  34 00 00 00 14 00 00 00  |X.Q.....4.......|
08d1c7d0  a0 0b 00 00 00 00 00 00  7f 06 00 00 01 40 00 00  |.............@..|
08d1c7e0  08 40 00 00 01 00 00 00  00 00 00 00 00 00 00 00  |.@..............|
```

Second Log write (4KB)

```
08d1d000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
08d1e000
```

SQL Insert (cont.)

DB table write (1st)

```
288f16000  00 00 00 00 b8 c7 51 00  01 00 00 00 1c 00 c8 1f  |......Q.........|
288f16010  00 20 04 20 00 00 00 00  c8 9f 70 00 00 00 00 00  |. . ......p.....|
288f16020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
288f17000
```

Index (?) file write

```
2894b2000  00 00 00 00 38 c8 51 00  01 00 00 00 1c 00 e0 1f  |....8.Q.........|
2894b2010  f0 1f 04 20 00 00 00 00  e0 9f 20 00 00 00 00 00  |... ...... .....|
2894b2020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
2894b3fe0  00 00 00 00 01 00 10 00  01 00 00 00 00 00 00 00  |................|
2894b3ff0  00 00 00 00 00 00 00 00  00 00 00 00 03 00 00 00  |................|
2894b4000
```

DB table write (3rd)

```
288f17000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
288f17fc0  00 00 00 00 00 00 00 00  93 02 00 00 00 00 00 00  |................|
288f17fd0  00 00 00 00 00 00 00 00  01 00 03 00 02 08 18 00  |................|
288f17fe0  01 00 00 00 1d 48 65 6c  6c 6f 2c 20 57 6f 72 6c  |.....Hello, Worl|
288f17ff0  64 21 00 00 00 00 00 00  40 4d 96 6e 2e 75 01 00  |d!......@M.n.u..|
288f18000
```

Some References Worth Following

- Neeraja J. Yadwadkar, Chiranjib Bhattacharyya, K. Gopinath, Thirumale Niranjan, Sai Susarla. "Discovery of Application Workloads from Network File Traces", FAST 2010
  - http://static.usenix.org/events/fast10/tech/full_papers/yadwadkar.pdf
  - Identify which application somebody is running from the sequence of NFS operations that are sent

- Duy Le et al, "Understanding Performance Implications of Nested File Systems in a Virtualized Environment", FAST 2012
  - http://static.usenix.org/events/fast12/tech/full_papers/Le.pdf
  - What happens when you run one file system virtualized on top of another file system? **Bad things.**
- Yuanyuan Zhou, Zhifeng Chen, and Kai Li. "Second-Level Buffer Cache Management", IEEE Transactions on Distributed and Parallel Systems, Vol. 15, No. 7, July 2004
  - http://opera.ucsd.edu/paper/TPDS-final.pdf
  - Page-replacement algorithms don't work well as storage caches!

- Tyler Harter, Chris Dragga, Michael Vaughn, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau. "A File is Not a File: Understanding the I/O Behavior of Apple Desktop Applications" *Transactions on Computing Systems (TOCS),* August 2012, v. 30:3
  - http://research.cs.wisc.edu/wind/Publications/ibench-tocs12.pdf
  - Application I/O behavior is increasingly complex
  - "Sequential is not Sequential"?!?
  - Looks at a lot of aggregate stats
  - (Lots of other neat stuff coming out of the Arpaci-Dusseaus at UW-Madison!)

- Zhenmin Li, Zhifeng Chen, Sudarshan M. Srinivasan and Yuanyuan Zhou. "*C-Miner*: Mining Block Correlations in Storage Systems", FAST 2004
  - http://static.usenix.org/events/fast04/tech/full_papers/li/li_html/paper.html
  - Storage system that tries to predict which block will be accessed next and prefetch it
  - Some visualizations of block correlation from traces

Things I Would Love To Do If This Were Actually My FT Job Instead of a Side Project

- Other file systems (ZFS, Windows)
  - Log-structured file systems look a lot different!
  - Is there a debugger for NTFS?
- Larger database traces
  - Can we distinguish indexed from non-indexed queries?
- Better visualizations and statistical tests
- Block-by-block correlations
- Clustering techniques?

Thanks for Attending!

Mark Gritter
mgritter@gmail.com
Twitter: @markgritter

- Rejected presentation titles:
  - "Around the Disk in 80 Milliseconds"
  - "From the Earth to the SCSI Bus"
  - "The Mysterious Block Device"