Intelligent Cache Hinting

3x Improvement in Small File Performance with I/O Hinting in Lustre*

Micah Bhakti – Product Manager / ISV Partnerships

Why Provide Caching in Lustre*?

What problems are users experiencing?

- Small file and random I/O
- Complex to managing multiple file systems
- Pure SSD storage is expensive

Problems with traditional Caching

- Caches are inefficient at keeping “important” data
- Application context is lost in storage interfaces
  (e.g., SCSI, REST)
- End-to-end Quality of Service is increasingly difficult

Conceptual Approach

I/O Classification

#1 Classification

Application data

Classify each storage I/O request (in-band)

#2 Policy assignment

Storage policies

Assign policies to I/O classes (out-of-band)

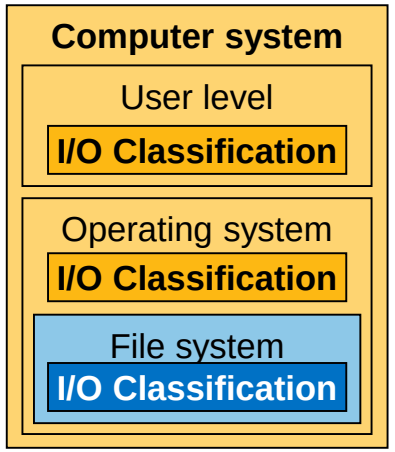
#3 Policy enforcement

*Lustre**

Enforce policies in storage system

Classification Architecture

Classification



Policy assignment

Classifier	Group #
Metadata	*0*
Boot files	*1*
Small files	*3*
Media files	*2*
...	...

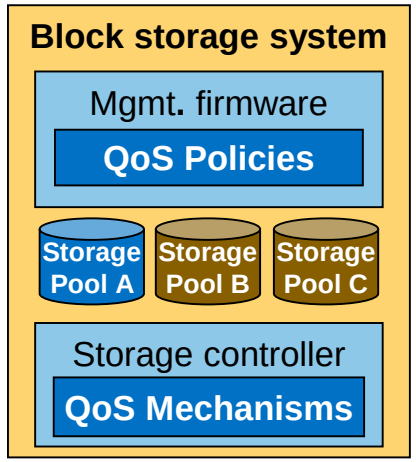
(offline)

Table 75 — WRITE (10) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Ah)							
1	WRPROTECT			DPO	FUA	Reserved	FUA_NV	Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	Reserved			GROUP NUMBER				
7	(MSB)	TRANSFER LENGTH						
8								(LSB)
9	CONTROL							

Classify each I/O in-band (e.g., in SCSI command descriptor block)

Policy enforcement



intel Software

Cache Implementation – How do we pass hints?

Classifying I/O using SCSI Command Descriptor Block

Table 75 — WRITE (10) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Ah)							
1	WRPROTECT			DPO	FUA	Reserved	FUA_NV	Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	Reserved			GROUP NUMBER				
7	(MSB)	TRANSFER LENGTH						
8								(LSB)
9	CONTROL							

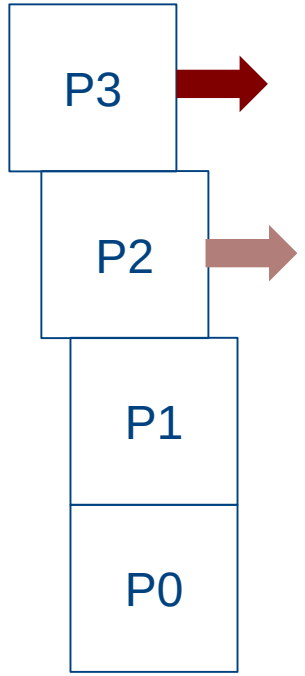
5 bits → up to 32 classes

Cache Implementation – What do we need?

Caching HW/SW must do the following –

1. Implement a ***selective allocation*** algorithm
   - Multiple LRUs, each with for a unique class (priority)
   - I/O is assigned to LRUs (based on their class)
   - Low priority I/O is “fenced off” under cache pressure

2. Implement a ***selective eviction*** algorithm
   - Data classes can be soft-pinned (priority-based eviction)
   - Data classes can be hard-pinned (never evicted)



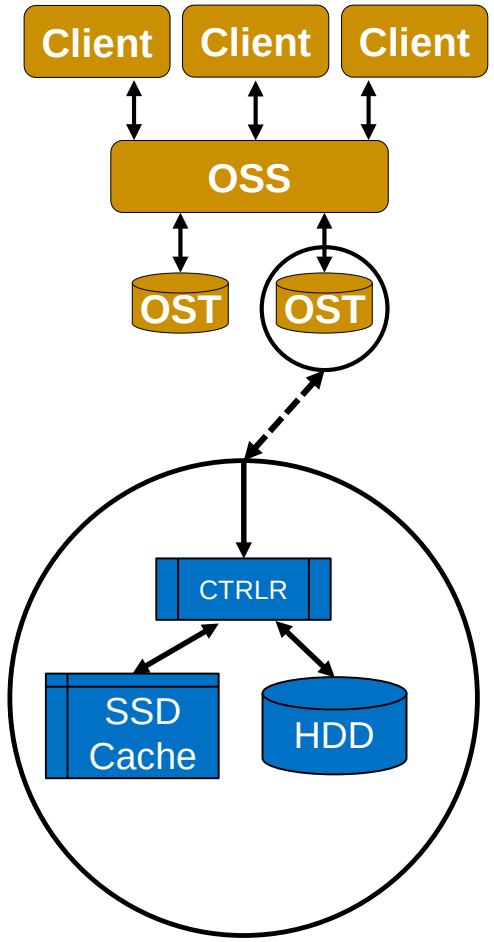
Cache Implementation – Where do we cache?

Client-side Caching is Difficult

- More clients increases the cost
- Cache Coherency Issues
- Moving data from cache to storage can cause problems without QoS

Caching on the Server is Preferable

- Fewer devices, reduced cost
- Transparent to Lustre*
- Fails over with OST



intel Software

Integration with Lustre*

Ldiskfs changes

- Add classifier to I/O requests
- Only coalesce like-class requests
- Copy classifier into SCSI CDB
- 18 classes identified →
- Optimized for a file server
  - ➢***Small files & metadata***

One-time change to FS

Ldiskfs Class	Group Number	Cache priority
Unclassified	0	12
Superblock	1	0
Group desc.	2	0
Bitmap	3	0
Inode	4	0
Indirect block	5	0
Directories	6	0
Journal	7	0
File <= 4KB	8	1
File <= 16KB	9	2
File <= 64KB	10	3
…	…	…
File > 1GB	18	11

Performance Analysis - Objective

Combine traditional Lustre* IO (IOR) with Small File and Random I/O (Postmark) to assess the benefits of Cache Hinting

- Is 10% Cache storage significant?
- Performance compared to LRU cache?
- Performance gap to pure SSD?

What is the impact on overall Lustre performance?

Test Cluster Configuration

All nodes R2000GZ* servers

- 2-socket Intel® Xeon® processors - E5-2680 v2 @ 2.8 GHz, 96 GB DRAM
- 10 GbE network
- SSD system disk
- 10 TB additional storage
- Centos* 6.4, 2.6.32 kernel, Lustre* 2.5.x

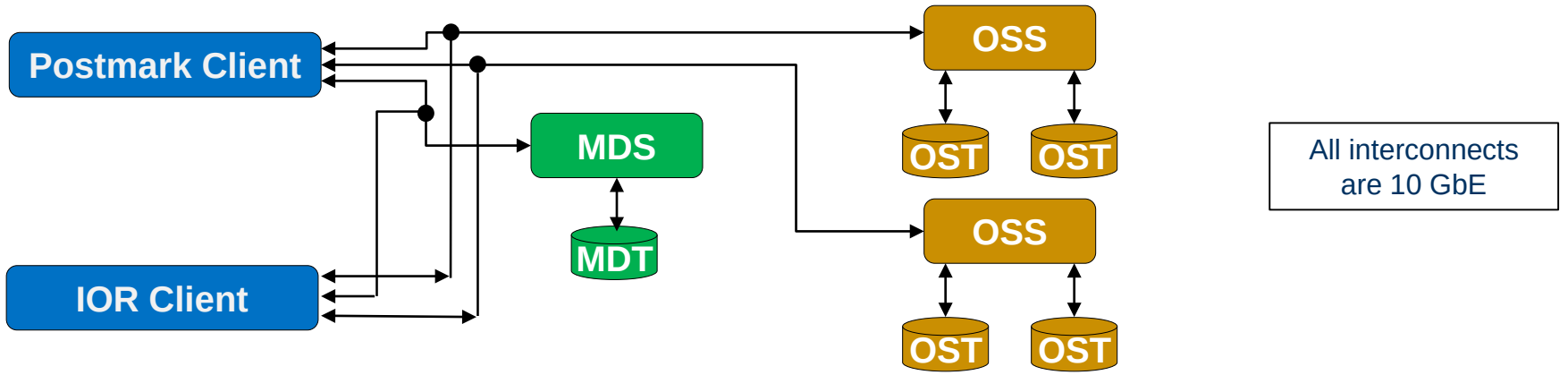
OSS(s) have 2 12TB RAID-6 OSTs (via JBOD)

2 Clients, 1 MGS/MDS, 2 OSS X 2 OSTs

Initial testing

- 1 OSS with single OST; 1 OSS with 2 OSTs
- Results are consistent for both configurations

Benchmark Configuration



All interconnects are 10 GbE

Postmark

- For our tests
  - 300,000 Files (1KB to 32MB)
  - 10,000 Subdirectories
  - 30,000 Transactions

The SPECsfs file server configuration

- ~180 GB of data created

IOR (Interleaved-Or-Random)

For our tests:

BEGIN_LOOP
- Write 20GB “checkpoint” file
- Write 8MB “data set 1” file
- Write 8MB “data set 2” file
- Read checkpoint file
- Read data set 1
- Read data set 2

END_LOOP

Runs concurrently with Postmark

intel Software

Caching – Results and Optimization

Caching vs. HDD

3.9x
Performance

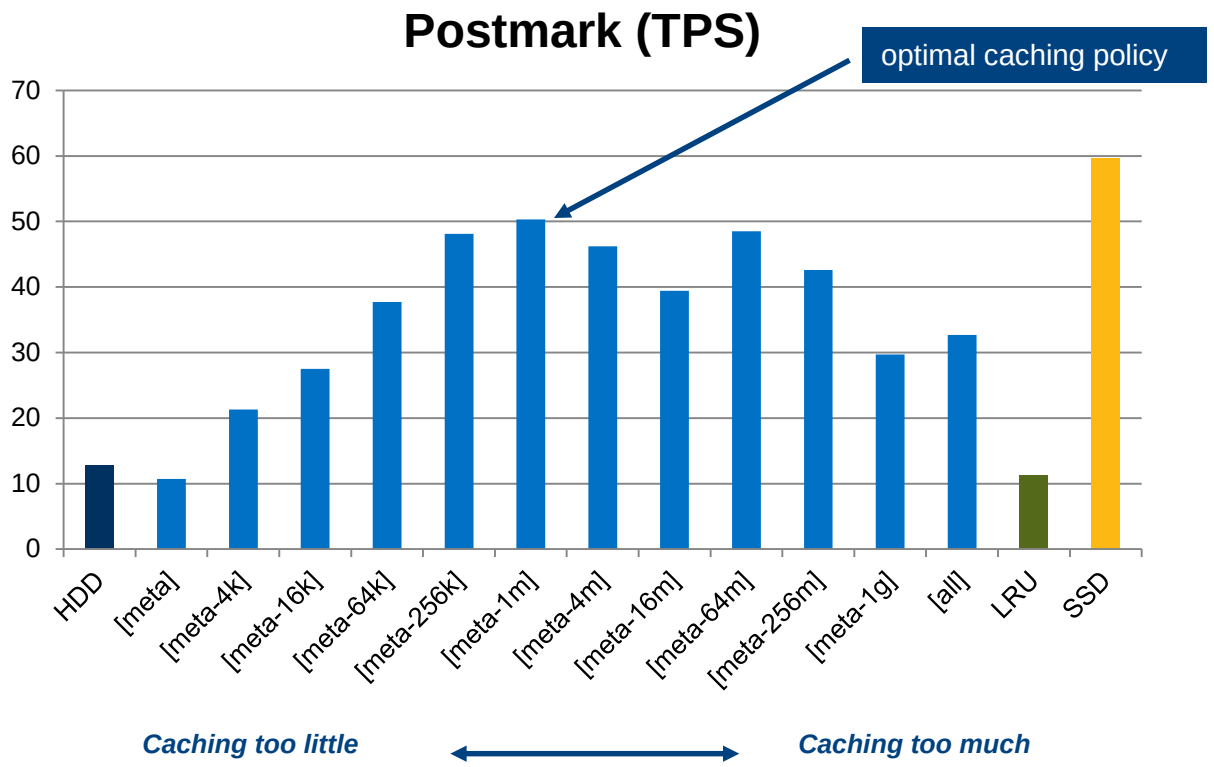
Cache too little?

-Underutilized SSD

Cache too much?

-Reduced HDD parallelism

-Cache thrash



I/O hints enable "hill-climbing" optimization

intel Software

Potential Future Enhancements

Future Improvements – Standardization

Standardize reference classifications with T10

- Standard set of base classifiers and group numbers for cache Hinting
- Reference schemas certified
- Provides vendors requirements to use for development

Classifier	Group #
Metadata	*0*
Boot files	*1*
Small files	*3*
Media files	*2*
...	...

Table 75 — WRITE (10) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Ah)							
1	WRPROTECT			DPO	FUA	Reserved	FUA_NV	Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	Reserved			GROUP NUMBER				
7	(MSB)	TRANSFER LENGTH						
8								(LSB)
9	CONTROL							

Upstream hinting capabilities into Linux kernel

Future Improvements

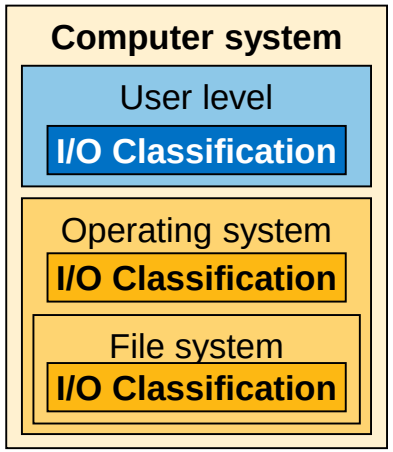
Provide Hints directly from User Applications

Files tagged with cache priority by specific applications

- Further optimization of hinting
- File system hints can still be used in lieu of application hints
- Difficult to assess for proprietary apps

What would be helpful in this space?

Classification



Software

Special thanks to the following people for enabling this work with cache hinting in Lustre* –

- Michael Mesnier – Intel Labs
- John Keys – Intel Labs
- Andreas Dilger – Intel HPDD
- Hongchao Zhang – Intel HPDD

More Information –

- Emerging SCSI standard for I/O hinting – SBC-3: Accessible LBA Access Hints (12-061r0).
- Christian Black, Michael Mesnier, Terry Yoshii. Solid-state Drive Caching with Differentiated Storage Services. Intel Whitepaper (IT Best Practices). July, 2012.
- T. Luo, R. Lee, M. Mesnier, F. Chen, and X. Zhang. hStorage-DB: Heterogeneity-aware Data Management to Exploit the Full Capability of Hybrid Storage Systems. PVLDB 5(10):1076-1087,2012.
- Michael Mesnier, Jason Akers, Feng Chen, Tian Luo. Differentiated Storage Services. 23rd ACM Symposium on Operating Systems Principles (SOSP). October 2011.

Legal Information

- No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.
- Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.
- This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.
- The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.
- Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.
- Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at http://www.intel.com/content/www/us/en/software/intel-solutions-for-lustre-software.html.
- Intel and the Intel logo, are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others

intel Software


intel
Software