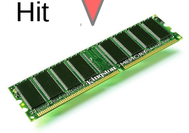
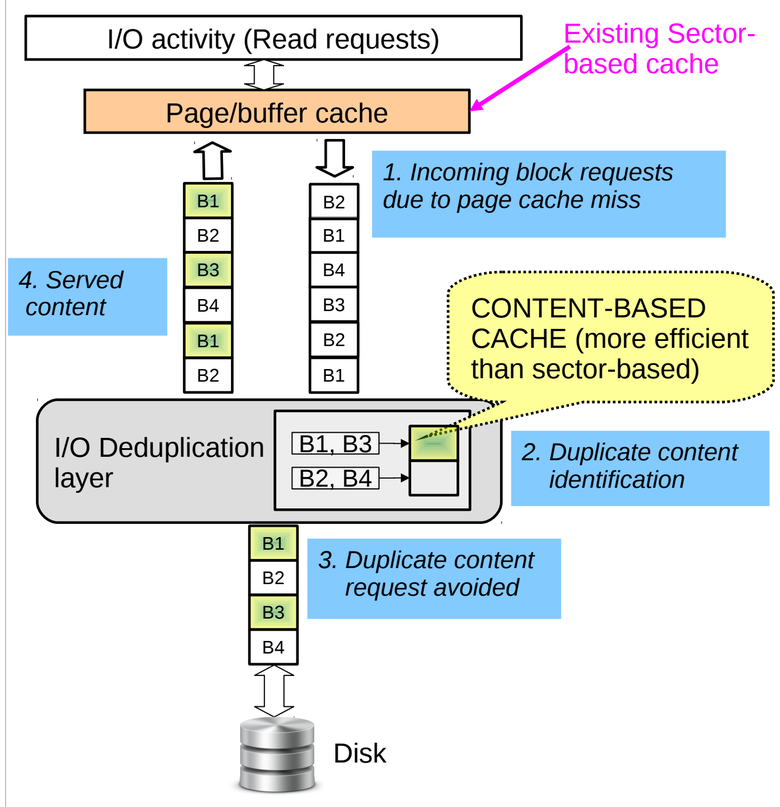
Improving I/O Performance using Integrated Deduplication Cache and Variable-sized Chunks

Sujesha Sudevalayam, Rahul Balani, Akshat Verma, Purushottam Kulkarni

De-duplication: Eliminate duplicates

Cache Hit 	*Memory deduplication* saves memory space	[ESX Server Memory Management] [Difference Engine] [Satori] [Singleton]
Cache Miss		
I/O Request	*I/O deduplication* saves I/O access time	[I/O Deduplication]
Disk read 	*Storage deduplication* saves disk space	[Data Domain] [Decentralized Deduplication] [LiveDFS] [SiLo] [Sparse Indexing]

I/O De-duplication [1]

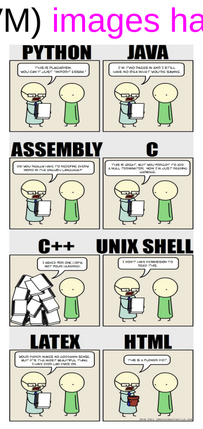


[1] *I/O Deduplication: Utilizing Content Similarity to Improve I/O Performance*

I/O Deduplication in Virtualized Environment

Many virtual machine (VM) images have similar content:-

operating systems



Applications, libraries and development environments.

In virtualization context, I/O de-duplication can be beneficial in following scenarios:-

- Mass instantiation of VMs (simultaneous requests and/or batched requests)
- Disaster recovery (recovery from machine breakdown, network outage, power failure, etc)
- VM-hosted application run-time

BACKGROUND

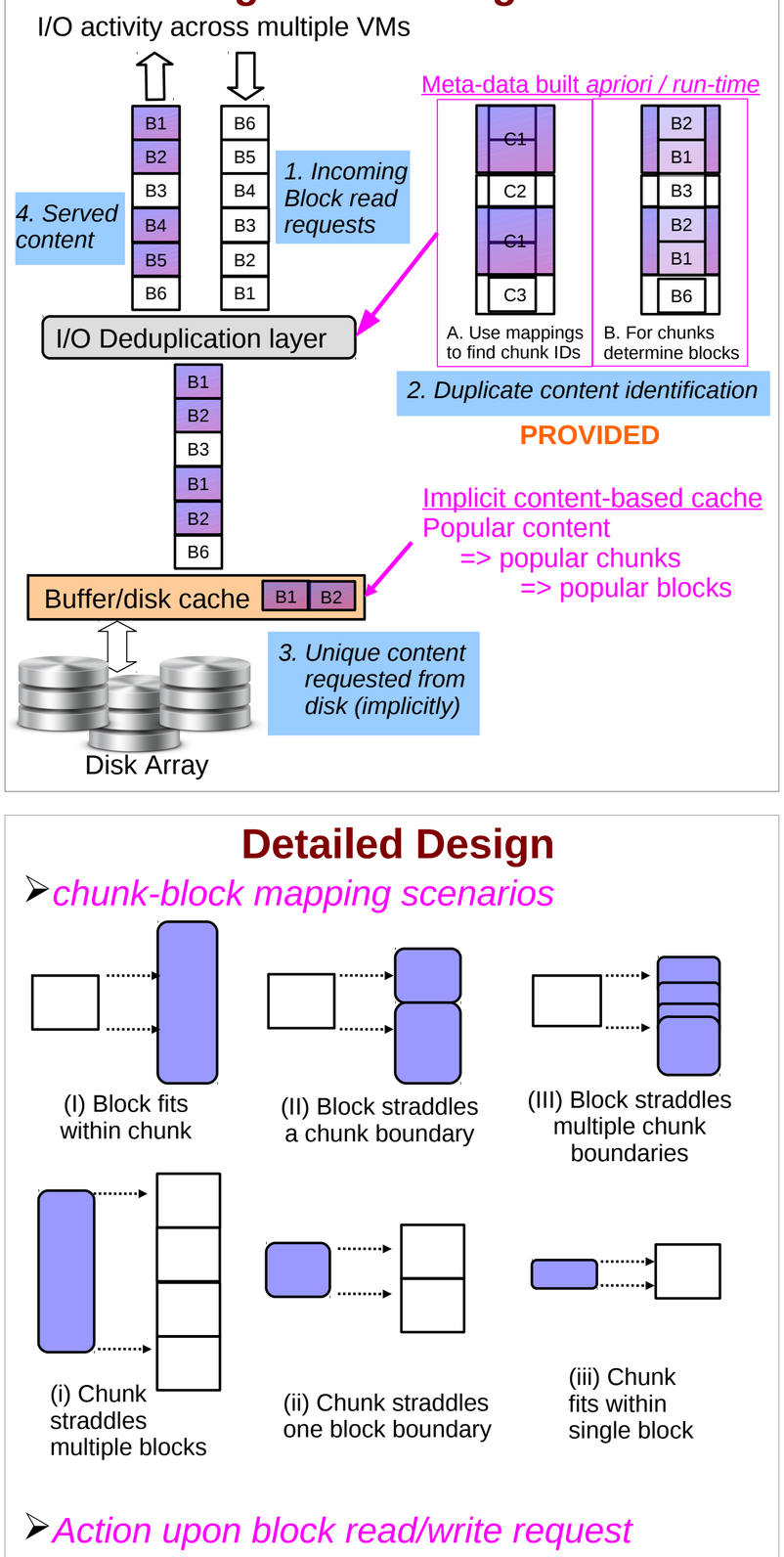
Motivation

- Block requests always specify a fixed number of sectors, i.e. fixed-size blocks [2]
- Disk also handle only fixed-size sector or blocks [2]
- However, de-duplication efficiency better with variable-chunks than fixed-size blocks – 25% more [3]
- In virtualized environment, disk requests (virtual blocks) mapped into physical blocks by storage virtualization engine [4,5]
- Such virtual-to-physical (V2P) mapping can exist at multiple levels – hypervisor, object-store, storage volume controller.

Problem Statement

Can we improve I/O de-duplication performance by
1) Augmenting the virtual-to-physical mapping with variable chunk-size handling?
2) Integrating buffer cache and de-duplication cache

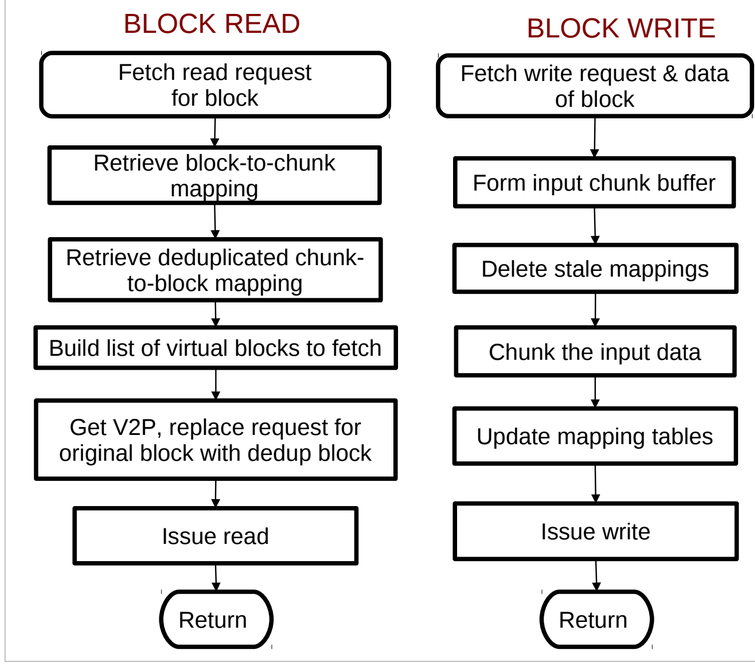
High-level Design



Detailed Design

➢ *chunk-block mapping scenarios*

➢ *Action upon block read/write request*



SYSTEM DESIGN

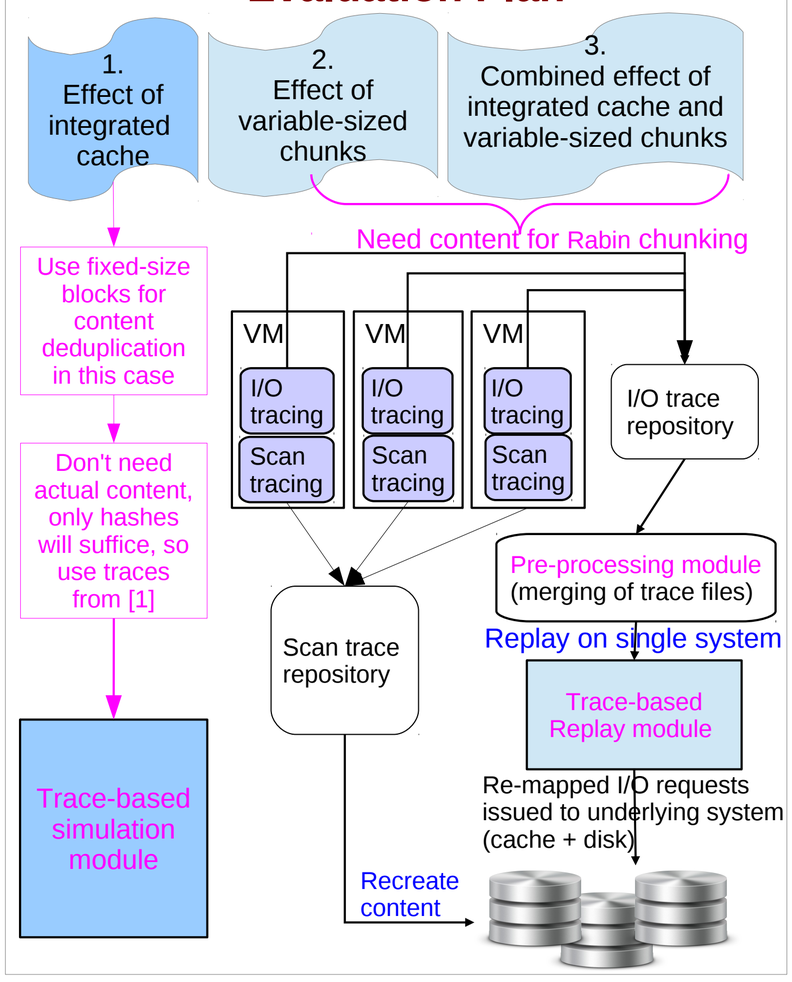
Contributions

- Redirection of I/O requests for using buffer cache as a content-deduplicated cache
- Identification of duplicates using mapping metadata between Fixed-size blocks and variable-sized chunks
- Management of metadata by building mapping between blocks and chunks at run-time

Normalized similarity of variable wrt fixed (%)	Centos 6.4	Fedora 18	Opensuse 12.3	Ubuntu 10.04	Ubuntu 12.04	Ubuntu 12.10	Ubuntu 13.04
Centos 6.4	0						
Fedora 18	12	0					
OpenSUSE 12.3	58	72	0				
Ubuntu 10.04	66	11	77	0			
Ubuntu 12.04	68	72	124	1	0		
Ubuntu 12.10	54	91	133	10	8	0	
Ubuntu 13.04	71	87	144	28	17	-7	0

Benchmarking fixed versus variable for VM booting traces

Evaluation Plan



Build metadata apriori vs runtime

	Apriori	Run-time
Entails what?	Mapping is built during an initial offline scan of content	Mapping is built online during every block read/write
Benefit	Metadata building is fairly straight-forward due to sequential scan of blocks	Metadata size proportional to number of blocks read/written => can fit in RAM
Drawback	Metadata size proportional to total number of blocks => may not fit in RAM	Metadata building relatively complex since blocks may not be read/written sequentially
Useful when?	When "working set" will eventually span all blocks (Eg., SAN)	When "working set" is a small fraction of total blocks, metadata size can be limited (Eg. host)
To handle metadata size?	Store index in RAM and disk using bloom-filter based techniques s.t. 99% lookups served from RAM [6]	
Our work	Algorithm for building block to chunk and chunk to block mapping upon sequential scan, and for updating mapping upon every bock write	Algorithm for building/updating block to chunk and chunk to block mapping upon every block read and write

Work-in-progress

1.Simulation module for testing effect of integrated cache, READY. Testing & evaluation in progress.
2.Trace-based replay module for testing effect of variable versus fixed, with apriori map building READY. Runtime map building to be done.
3.Trace collecting kernel module READY. Trace collection to be done.

CONTRIBUTIONS & DISCUSSION

References

[1] *I/O Deduplication: Utilizing Content Similarity to Improve I/O Performance.* Ricardo Koller and Raju Rangaswami. FAST 2010
[2] *Linux Device Drivers*. Jonathan Corbet, Greg Kroah-Hartman, Alessandro Rubini. Online as on 6-July-2012. http://www.makelinux.net/ldd3/chp-16
[3] *An Empirical Analysis of Similarity in Virtual Machine Images.* K.R. Jayaram et al. Middleware 2011.
[4] *The Definitive Guide to the Xen Hypervisor.* David Chisnall. Published by Prentice Hall.
[5] *Kernel Virtual Machine (KVM)*. www.linux-kvm.org
[6] *Avoiding the Disk Bottleneck in the Data Domain Deduplication File System.* Zhu B, Li K, Patterson H. FAST 2008.