Host-side Filesystem Journaling for Durable Shared Storage

Andromachi Hatzieleftheriou, Stergios V. Anastasiadis
Department of Computer Science & Engineering
University of Ioannina, Greece
Outline

Motivation

Design

Implementation

Evaluation

Conclusions
Datacenter Storage

Multi-tier distributed systems on clusters of commodity servers and disk drives

- client-side frontend
- caching layer
- backend storage
Multi-tier distributed systems on clusters of commodity servers and disk drives

- client-side frontend
- caching layer
- backend storage
Datacenter Storage

Multi-tier distributed systems on clusters of commodity servers and disk drives

- client-side frontend
- caching layer
- backend storage

Frontend tier: client-side

- stateless for reduced cross-layer communication
- recent updates kept in volatile memory
- lost data in case of client failure/reboot
Ceph object-based scale-out file system
  • client-side memory-based caching

Experiment
  • Filebench fileserver
  • writeback every 5 sec dirty data older than 30 sec (default)

Outcome
  • on average, 24.3MB of dirty data only in volatile memory over time
Storage Interfaces

Block-based
- ✓ virtualization flexibility
- × no sharing
- × translation overhead
- × semantic gap

File-based
- ✓ file sharing
- ✓ improved performance
- ✓ semantics awareness
- × compatibility limitations

Object-based
- ✓ scalability
The Problem of Block-based Caching

Strengths

Performance
• improved throughput
• reduced latency

Efficiency
• reduced server & network load

Weaknesses

Functionality
• no file sharing

Overhead
• translation overhead
• metadata persistence

Consistency
• semantic gap
• ordering of requests
• grouping of operations
Key Insight

Improve

• the durability of the memory-based cache at the client of shared storage systems

Gain

• performance
• efficiency
Design Goals

Interface
• POSIX-like file-based interface

Sharing
• native file-sharing support within and across hosts

Durability
• recent updates survive client reboots

Performance
• sequential disk throughput for writes

Scalability
• scale-out backend servers
The Arion System Architecture

Distributed filesystem
- object-based
- multiple data & metadata servers
- virtualized or bare-metal client
- multiple backend replicas

Local journal at the client-side
- heterogeneous replication
- reliable directly attached storage
- log both data and corresponding metadata
- one journal per client
Journaling and Writeback

Local journal device attached to client at mount-time

Commit
- synchronously transfer data updates from memory to journal
- periodically or explicit flush request

Writeback
- occasionally copy data blocks from memory to filesystem servers
- periodically, under memory or journal space pressure

Written-back and invalidated pages removed from journal
Consistency

Shared file access with tokens

Normal operation - conflicting writes from different clients
- checkpoint pending writes
- invalidate related journal entries
- revoke write token

Failure - client reconnection or reboot after a crash
- acquire required tokens
- replay file updates only if journaled metadata is newer than metadata fetched from the server
Implementation

Prototype implementation

- CephFS kernel-level client (Linux kernel v3.6.6)
- Linux JBD2

During commit

- include metadata attributes in the journal tags of the descriptor block

During recovery

- compare journaled metadata attributes with those newly fetched from MDS
- replay writes only for files not accessed after the transaction commit
Experiments

Experimentation Environment:

- Backend Servers - Ceph v0.80.1 (3 OSDs, 1 MON, 1 MDS)
  - 3GB RAM, 2 x 300GB 15KRPM SAS disks, 2 x quad-core x86-64 2.66GHz
  - separate OSD journal device
  - replication factor 3

- Virtualized client
  - 2GB RAM, 2 x VCPUs
  - journal size 2GB, commit interval 1s

- Host - XEN v4.2.0
  - 2 x 300GB 15KRPM SAS disks (RAID0), 2 x quad-core x86-64 2.66GHz

Workloads

- Filebench fileserver & mail server
- Flexible I/O Tester
Arion flushes dirty data to local journal every second

Reduced amount of vulnerable data in memory
  • from 24.3MB to 5.4MB over time
Filebench Mail Server

Varying writeback and expiration intervals

Arion achieves up to 58% higher operation throughput than Ceph

OSD network traffic normalized by the number of completed operations
- 30% reduction of the received network load
- 27% reduction of the transmitted network load
Arion-60 achieves 22% reduced write latency

Received OSD network traffic reduced by 42%
Flexible IO Tester

Reduced disk utilization at the servers
  • filesystem device utilization reduced by 82%
Conclusions & Future Work

Durable shared storage through host-side journal integration at the client of a distributed filesystem

Tunable control over
- amount of dirty pages staged at the host
- time period for dirty pages to reach the backend servers

Benefits
- improved durability of frontend memory caching
- increased performance
- resource efficiency (network & disk)

Future work
- further experimentation
- extend host-based journaling to support disk-based caching