Surf

XYZ

July 17, 2014

Abstract

1 Introduction

- Why in-memory caching tier for Big Data Analytics?
- Our new approach: a generic, transparent caching service backed by a backend distributed file system

  cache: soft-state

  support for multi-frameworks

  how file system clients use surf? Use surf:// instead of hdfs:// in their file system call, and the rest of things are handled by surf.
- Unique features of Surf
  - Application-driven data replication : per-file replication policy
  - Elastic cluster-level memory pool provisioning: increase or decrease the number of caching servers depending on caching load
- Implementation: built upon REEF [], lines of code; In-production use, contributed to Apache
- Evaluation: how Surf beats HDFS, HDFS caching
- Paper roadmap

2 Overview

We need a nice transition between Intro and Design

- Maybe an overview of Surf?

3 Surf Design

- Basic Design - non-elastic, non-replication version
  - Pinning
  - Cache eviction
- Adding flexible replication - cache all, cache one, cache a few
- Adding elasticity
- Discussion?

4 Implementation

Surf is built atop REEF [**?**].

- Lines of code
- Client library exposing the file system interface, Thrift between client and caching task
- Server

5 Evaluation

Experiment setup - how many nodes, the spec. of each node
What do we compare with Surf: HDFS, HDFS caching
Summary of what we want to show from the experiments

- Microbenchmarks - explain
  - Result 1 (Each graph should make a point.)
  - Result 2
  - ...
- Macrobenchmarks - Hadoop MR jobs
  - Result 1
  - Result 2
  - ...
- Macrobenchmarks - SKT workloads (Shark/Spark jobs)
  - Result 1
  - Result 2
  - ...

6 Related Work

- In-memory work in Big Data Analytics
  - HDFS caching: tied to OS page cache, not elastic, not flexible
  - Spark RDD [**?**]: tied to a particular framework, surf: independent of frameworks
  - PacMan [**?**]
  - Tachyon - in-memory file system [**?**]: only a single copy in memory, file system semantics, recovery etc., surf: soft-state
- Caching in other domains: Memcache, Web caching, CDN, etc.

7 Conclusion and Future Work

- Contribution
- Summary of the numbers
- Say it's in production use in SKT
- Say it's contributed to the Apache incubation project
- Future work
  - write path? - anything we do to improve write performance?
  - How to handle intermediate data (written) that's not backed by the distributed file system
  - running code in the same JVM that hosts the cached data,

References