MixApart: Decoupled Analytics for Shared Storage Systems

Madalin Mihailescu, Gokul Soundararajan, Cristiana Amza

University of Toronto and NetApp
Hadoop + Enterprise storage?!

Shared storage (e.g., NAS)
Hadoop+Enterprise: Two Storage Silos

Cross-silo data management $$$

Periodic data ingest

Hardware $$$

Hardware _____

Hadoop
Our Solution: MixApart

- **MapReduce** analytics on **enterprise storage**
  - **Enterprise storage** – **single** reliable data store

![Diagram showing MapReduce Compute and On-disk cache for scalability]

**Transparent** and **on-demand** ingest
Data Flow with MixApart

Map task parallelism:
- Storage bandwidth
- Cache reuse
- Map task I/O rates

Data reuse

Map → Reduce

Reduce → Data reuse
Workload Analysis

- **Extrapolate from recent studies**
  - *Production traces* from Facebook, Bing, Yahoo

- **Insights**
  - High *data reuse* across jobs e.g., $\sim 60\%$
  - Low IO to CPU ratio in input phases e.g., $\sim 25\text{Mbps}$
  - Predictable IO demands

* Ananthanarayanan et al. NSDI ’12, Chen et al. VLDB ’12
Compute Scale Estimates

Map Task I/O Rate 25 Mbps

Shared storage bandwidth 10 Gbps

Data Reuse Ratio

0

0.2

0.4

0.6

0.8

0.95

# of Map Tasks

0

10

100

1000

10000

100000

400 parallel tasks

2000 parallel tasks

Parallel tasks
MixApart Design

- **Storage back-end bandwidth** management
  - Saturate bandwidth with Map I/O streams without impacting job performance
- **Cache** management
  - Ensure high cached data reuse
- **Compute** management
  - Assign Map tasks to nodes with cached data
MapReduce Optimization

- **Predictable** job I/O demands at submission
  - User-specified job *input* data path
  - Derived Map task *I/O rates*

- Just-in-time parallel data prefetch within & across jobs
MixApart Architecture

Co-locates compute and data using:
- Job priorities
- Data in the cache

Issues prefetches using:
- Available storage bandwidth
- Job priorities
- Map I/O rates

JobTracker

Data locations

XDFS NameNode

Location Map

Data Transfer Scheduler

Compute Scheduler

Compute Node

Compute Node

Cache

Cache Node

Job priorities
Job IO demands
MixApart in Action

Job (F₁ F₂ F₃ F₄)

JobTracker

XDFS NameNode

Compute Scheduler

Location Map

Data Transfer Scheduler

Compute Node 1

Cache Node 1

F₁

F₃

Compute Node 2

Cache Node 2

F₃

exchange job input info
MixApart in Action

JobTracker

XDFS NameNode

1. exchange job input info

2. transfer F2

3. create tasks

Job (F1 F2 F3 F4)

Compute Scheduler

Location Map

Data Transfer Scheduler

T1 T2 T3 T4

Compute Node 1

Cache Node 1 F1

Cache Node 2 F3

transfer F4

transfer F2
MixApart in Action

1. JobTracker exchanges job input info with NameNode.

2. XDFS NameNode transfers F2 to Compute Node 1.

3. Create tasks:
   - T2
   - T4

4. Compute:
   - T1 and T3
   - Prefetch F2 and F4

Job (F1 F2 F3 F4)
MixApart in Action

1. JobTracker
   - XDFS NameNode
   - Location Map
   - Data Transfer Scheduler
   - Compute Scheduler

   Job (F₁, F₂, F₃, F₄)

   exchange job input info

2. Transfer
   - F₂
   - F₄

3. Create tasks

4. Compute
   - T₁ and T₃
   - Prefetch F₂ and F₄
MixApart Prototype

- Re-engineered Hadoop MapReduce and HDFS
  - XDFS cache
    - Stateless HDFS + NFS support
  - Compute scheduler
    - FIFO task scheduler + cache aware
  - Data transfer scheduler
    - Module in NameNode
Evaluation on Amazon EC2

- MixApart vs. Hadoop
- **100-core** compute cluster
  - 50 EC2 VM instances
    - 7.5 GB RAM, 850GB local storage
  - *Local VM instance storage* for XDFS cache & HDFS
- **NFS server**
  - EC2 instance
    - 4 EBS volumes in RAID-0 setting
    - **1Gbps** bandwidth for analytics
Microbenchmarks

- **Dataset**
  - 12 days of Wikipedia statistics

- **Workload**
  - MR Job to aggregate page views for regex
  - Job on uncompressed data – I/O intensive
  - Job on compressed data – CPU intensive
Impact of Ingest

MixApart faster: overlap of compute and ingest

Next: MixApart vs. ideal Hadoop with no static ingest
Microbenchmark Job Durations

Data Reuse Ratio

MixApart ~ Hadoop

0.6 reuse: MixApart ~ Hadoop
2 Jobs Co-scheduled

- **MixApart**
- **Hadoop-ideal**

**Job A**
- High priority
- High reuse

**Job B**
- Low priority
- Low reuse

**Time (Normalized to Hadoop)**

- Compute A
- Wait B
- Compute B
- Prefetch B
- Compute B

**Time**
2 Jobs Co-scheduled

MixApart: *work conserving compute scheduling*
Facebook Job Durations

MixApart matches Hadoop when ignoring ingest!

- 0.09 Reuse Trace: +12%
- 0.48 Reuse Trace: +0.2%
- 0.81 Reuse Trace: +0.9%
Facebook Compute Concurrency

CDF

Reduce phase parallelism

Map phase parallelism

MixApart
Hadoop-ideal

Number of Running Tasks
MixApart Summary

- **MapReduce** analytics on enterprise storage
  - *Enterprise storage* – **single** reliable data store
    - Optimized *storage efficiency*
    - Simplified *data management*
  - MixApart *faster* than *ingest-then-compute* Hadoop
  - MixApart *comparable* to Hadoop with no ingest
Thank you!
Questions?