AUTOPLACER: Scalable Self-Tuning Data Placement in Distributed Key-value Stores
ICAC’13

João Paiva, Pedro Ruivo, Paolo Romano, Luís Rodrigues

Instituto Superior Técnico / Inesc-ID, Lisboa, Portugal

June 27, 2013
Outline

Introduction

Our approach

Evaluation

Conclusions
Motivation

Collocating processing with storage can improve performance.

- Using random placement, nodes waste resources due to node-intercommunication.
- Optimize data placement to improve locality and to reduce remote requests.
Motivation

Collocating processing with storage can improve performance.

- Using random placement, nodes waste resources due to node-intercommunication.

- Optimize data placement to improve locality and to reduce remote requests.
Motivation

Collocating processing with storage can improve performance.

- Using random placement, nodes waste resources due to node-intercommunication.
- Optimize data placement to improve locality and to reduce remote requests.
Approaches Using Offline Optimization

Algorithm:

1. Gather access trace for all items
2. Run offline optimization algorithms on traces
3. Store solution in directory
4. Locate data items by querying directory

- Fine-grained placement
- Costly to log all accesses
- Complex optimization
- Directory creates additional network usage
Approaches Using Offline Optimization

Algorithm:

1. Gather access trace for all items
2. Run offline optimization algorithms on traces
3. Store solution in directory
4. Locate data items by querying directory

- Fine-grained placement
- Costly to log all accesses
- Complex optimization
- Directory creates additional network usage
Main challenges

**Cause:** Key-Value stores may handle large amounts of data

**Challenges:**

1. **Collecting Statistics:** Obtaining usage statistics in an efficient manner.
2. **Optimization:** Deriving fine-grained placement for data objects that exploits data locality.
3. **Fast lookup:** Preserving fast lookup for data items.
Approaches to Data Access Locality

1. Consistent Hashing (CH):
   The “don’t care” approach

2. Distributed Directories:
   The “care too much” approach
Consistent Hashing

Don’t care for locality: items placed deterministically according to hash functions and full membership information.

- Simple to implement
- Solves **lookup challenge** by using local lookups

- No control on data placement $\rightarrow$ bad locality
- Does not address **optimization challenge**
Consistent Hashing

Don’t care for locality: items placed deterministically according to hash functions and full membership information.

- Simple to implement
- Solves **lookup challenge** by using local lookups
- No control on data placement → bad locality
- Does not address **optimization challenge**
Distributed Directories

Care too much for locality: nodes report usage statistics to centralized optimizer, placement defined in a distributed directory (may be cached locally)

- Can solve **statistics challenge** using coarse statistics
- Solves **optimization challenge** with precise data placement control

Hindered by **lookup challenge**:
- Additional network hop
- Hard to update
Distributed Directories

Care too much for locality: nodes report usage statistics to centralized optimizer, placement defined in a distributed directory (may be cached locally)

- Can solve **statistics challenge** using coarse statistics
- Solves **optimization challenge** with precise data placement control

Hindered by **lookup challenge**:
- Additional network hop
- Hard to update
Our approach: beating the challenges

Best of both worlds

- **Statistics Challenge**: Gather statistics only for hotspot items
- **Optimization Challenge**: Fine-grained optimization for hotspots
- **Lookup Challenge**: Consistent Hashing for remaining items
Algorithm overview

Online, round-based approach:

1. Statistics: Monitor data access to collect hotspots
2. Optimization: Decide placement for hotspots
3. Lookup: Encode / broadcast data placement
4. Move data
Algorithm overview

Online, round-based approach:

1. **Statistics**: Monitor data access to collect hotspots
2. Optimization: Decide placement for hotspots
3. Lookup: Encode / broadcast data placement
4. Move data
Statistics: Data access monitoring

**Key concept:** Top-K stream analysis algorithm

- Lightweight
- Sub-linear space usage
- Inaccurate result... But with bounded error
Key concept: Top-K stream analysis algorithm

- Lightweight
- Sub-linear space usage
- Inaccurate result... But with bounded error
Key concept: Top-K stream analysis algorithm

- Lightweight
- Sub-linear space usage
- Inaccurate result... But with bounded error
Algorithm overview

Online, round-based approach:

1. Statistics: Monitor data access to collect hotspots
2. **Optimization**: Decide placement for hotspots
3. Lookup: Encode / broadcast data placement
4. Move data
Optimization

Integer Linear Programming problem formulation:

\[
\min \sum_{j \in \mathcal{N}} \sum_{i \in \mathcal{O}} X_{ij} (c_r^{r} r_{ij} + c_r^{w} w_{ij}) + X_{ij} (c_l^{r} r_{ij} + c_l^{w} w_{ij}) \tag{1}
\]

subject to:

\[
\forall i \in \mathcal{O} : \sum_{j \in \mathcal{N}} X_{ij} = d \land \forall j \in \mathcal{N} : \sum_{i \in \mathcal{O}} X_{ij} \leq S_j
\]

Inaccurate input:

- Does not provide optimal placement
- Upper-bound on error
Accelerating optimization

1. ILP Relaxed to Linear Programming problem
2. Distributed optimization

LP relaxation
- Allow data item ownership to be in \([0 - 1]\) interval

Distributed Optimization
- Partition by the \(N\) nodes
- Each node optimizes hotspots mapped to it by CH
- Strengthen capacity constraint
Accelerating optimization

1. ILP Relaxed to Linear Programming problem
2. Distributed optimization

LP relaxation

- Allow data item ownership to be in $[0 - 1]$ interval

Distributed Optimization

- Partition by the $N$ nodes
- Each node optimizes hotspots mapped to it by CH
- Strengthen capacity constraint
Accelerating optimization

1. ILP Relaxed to Linear Programming problem
2. Distributed optimization

LP relaxation
- Allow data item ownership to be in $[0 - 1]$ interval

Distributed Optimization
- Partition by the $N$ nodes
- Each node optimizes hotspots mapped to it by CH
- Strengthen capacity constraint
Algorithm overview

Online, round-based approach:

1. Statistics: Monitor data access to collect hotspots
2. Optimization: Decide placement for hotspots
3. **Lookup**: Encode / broadcast data placement
4. Move data
Probabilistic Associative Array (PAA)

- Associative array interface (keys→values)
- Probabilistic and space-efficient
- Trade-off space usage for accuracy
Probabilistic Associative Array: Usage

Building

1. Build PAA from hotspot mappings
2. Broadcast PAA

Looking up objects

- If item not in PAA, use Consistent Hashing
- If item is hotspot, return PAA mapping
Probabilistic Associative Array: Usage

Building

1. Build PAA from hotspot mappings
2. Broadcast PAA

Looking up objects

- If item not in PAA, use Consistent Hashing
- If item is hotspot, return PAA mapping
PAA: Building blocks

- **Bloom Filter**
  Space-efficient membership test (is item in PAA?)

- **Decision tree classifier**
  Space-efficient mapping (where is hotspot mapped to?)
PAA: Building blocks

- **Bloom Filter**
  Space-efficient membership test (is item in PAA?)

- **Decision tree classifier**
  Space-efficient mapping (where is hotspot mapped to?)
PAA: Properties

Bloom Filter:

- **False Positives**: match items that it was not supposed to.
- **No False Negatives**: never return \(\perp\) for items in PAA.

Decision tree classifier:

- **Inaccurate values** (bounded error).
- **Deterministic response**: deterministic (item \(\rightarrow\) node) mapping.
PAA: Properties

Bloom Filter:

- **False Positives**: match items that it was not supposed to.
- **No False Negatives**: never return ⊥ for items in PAA.

Decision tree classifier:

- **Inaccurate values** (bounded error).
- **Deterministic response**: deterministic (item→node) mapping.
PAA: Properties

Bloom Filter:
- **False Positives**: match items that it was not supposed to.
- **No False Negatives**: never return ⊥ for items in PAA.

Decision tree classifier:
- **Inaccurate** values (bounded error).
- **Deterministic response**: deterministic (item→node) mapping.
Algorithm Review

Online, round-based approach:

1. Statistics: Monitor data access to collect hotspots
   - Top-k stream analysis
2. Optimization: Decide placement for hotspots
   - Lightweight distributed optimization
3. Lookup: Encode / broadcast data placement
   - Probabilistic Associative Array
4. Move data
Algorithm Review

Online, round-based approach:

1. Statistics: Monitor data access to collect hotspots
   Top-k stream analysis
2. Optimization: Decide placement for hotspots
   Lightweight distributed optimization
3. Lookup: Encode / broadcast data placement
   Probabilistic Associative Array
4. Move data
Algorithm Review

Online, round-based approach:

1. Statistics: Monitor data access to collect hotspots
   Top-k stream analysis
2. Optimization: Decide placement for hotspots
   Lightweight distributed optimization
3. Lookup: Encode / broadcast data placement
   Probabilistic Associative Array
4. Move data
Algorithm Review

Online, round-based approach:

1. Statistics: Monitor data access to collect hotspots
   Top-k stream analysis
2. Optimization: Decide placement for hotspots
   Lightweight distributed optimization
3. Lookup: Encode / broadcast data placement
   Probabilistic Associative Array
4. Move data
Outline

Introduction

Our approach

Evaluation

Conclusions
Experimental settings

- Integrated in Distributed Key-Value store (JBoss Infinispan)
- 40 Virtual Machines (10 physical machines)
- Gigabit network
Modified TPC-C benchmark

Induce controllable locality:

- Probability $p$: Nodes access data associated with a given warehouse.
- Probability $1 - p$: Nodes access data associated with a random warehouse.
Remote operations

![Graph showing the percentage of remote operations over time for different locality levels: 100%, 90%, 50%, and 0% locality, as well as a baseline. The graph plots time in minutes on the x-axis and the percentage of remote operations on the y-axis.]
Directory effects

Transaction per second (tx/s)

Autoplacer
Directory
Baseline

100% Locality
90% Locality
0% Locality

Transaction per second (tx/s)

100% Locality
90% Locality
0% Locality
Outline

Introduction

Our approach

Evaluation

Conclusions
Conclusions

- Gather statistics only for hotspots
- Fine-grained hotspot placement
- Retain Local lookups using PAA
- Effective locality improvement
- Good network usage
- Considerable performance improvements
Conclusions

- Gather statistics only for hotspots
- Fine-grained hotspot placement
- Retain Local lookups using PAA
- Effective locality improvement
- Good network usage
- Considerable performance improvements
Thank you