Scalable self-tuning data placement for distributed transactional memory systems

João Paiva, Pedro Ruivo, Paolo Romano, Luís Rodrigues

Inesc-ID / IST, Portugal

April 14, 2013

Outline

Why is data placement important?

Motivation

Powerful machines are wasting resources due to node-intercommunication.

Optimize data placement to reduce remote requests by improving data locality

Why is data placement important?

Motivation

Powerful machines are wasting resources due to node-intercommunication.

Optimize data placement to reduce remote requests by improving data locality

Classical Approaches

Algorithm

- Run offline optimization algorithms over all data items
- Store solution in directory
- Locate data items by querying directory

Cons

Complex optimization
Directory creates additional network usage

Classical Approaches

Algorithm

- Run offline optimization algorithms over all data items
- Store solution in directory
- Locate data items by querying directory

Cons

Complex optimization
Directory creates additional network usage

Main challenges

Distributed Transactional Memories may potentially handle a large amount of data

Implications

- Hard to obtain usage statistics
- Hard to optimize placement
- Hard to broadcast new placements

Existing Approaches

- Consistent Hashing
- Distributed Directories

Consistent Hashing

Data items placed according to hash functions.
Location derived locally, using full membership list.

Pros

- Simple
- Efficient

Cons

- No control on data placement $\rightarrow$ bad locality

Distributed Directories

Nodes report usage statistics to centralized optimizer
Placement is defined in a distributed directory (may be cached locally)

Pros

- Precise data placement control

Cons

- Additional network hop
- Hard to update

Outline

Our approach

Best of both worlds

Focus on hotspot items: optimized placement for selected data item

Consistent Hashing for most objects

Algorithm overview

Online, round-based approach

1. Monitor data access to collect hotspots
2. Decide on placement for hotspots
3. Encode placement
4. Broadcast placement
5. Move data

Algorithm overview

Online, round-based approach

1. **Monitor data access to collect hotspots**
2. Decide on placement for hotspots
3. Encode placement
4. Broadcast placement
5. Move data

Data access monitoring

- Top-K based algorithm instances:

Sub-linear space usage.

Inaccurate statistics.

... with bounded error.

Data access monitoring

- Top-K based algorithm instances:

Sub-linear space usage.

Inaccurate statistics.

... with bounded error.

Algorithm overview

Online, round-based approach

1. Monitor data access to collect hotspots
2. **Decide on placement for hotspots**
3. Encode placement
4. Broadcast placement
5. Move data

Optimization

ILP problem formulation $\rightarrow$ relaxed to LP

$$min \sum_{j \in \mathcal{N}} \sum_{i \in \mathcal{O}} \overline{X}_{ij}(cr^r r_{ij} + cr^w w_{ij}) + X_{ij}(cl^r r_{ij} + cl^w w_{ij}) \qquad (1)$$

subject to:

$$\forall i \in \mathcal{O} : \sum_{j \in \mathcal{N}} X_{ij} = d \wedge \forall j \in \mathcal{N} : \sum_{i \in \mathcal{O}} X_{ij} \leq S_j$$

Does not provide optimal placement
Bounded error

Algorithm overview

Online, round-based approach

1. Monitor data access to collect hotspots
2. Decide on placement for hotspots
3. **Encode placement**
4. Broadcast placement
5. Move data

Encoding placement

Probabilistic Associative Array (PAA)

- Space-efficient associative array (keys→values) instantiated for (data items→nodes)
- Trade-off space usage for accuracy

PAA: Building blocks

- **Bloom Filter**
- **Decision tree classifier**

PAA: Building blocks

- **Bloom Filter**
  Space-efficient membership test (is item in PAA?)
- **Decision tree classifier**
  Space-efficient (item$\rightarrow$node) association

PAA: Usage

- Build PAA from hotspot locations
- If item not in PAA, use Consistent Hashing

Consistent Hashing for most objects

- If item is hotspot, query decision tree

Focus on hotspot items: optimized placement for select data item

Outline

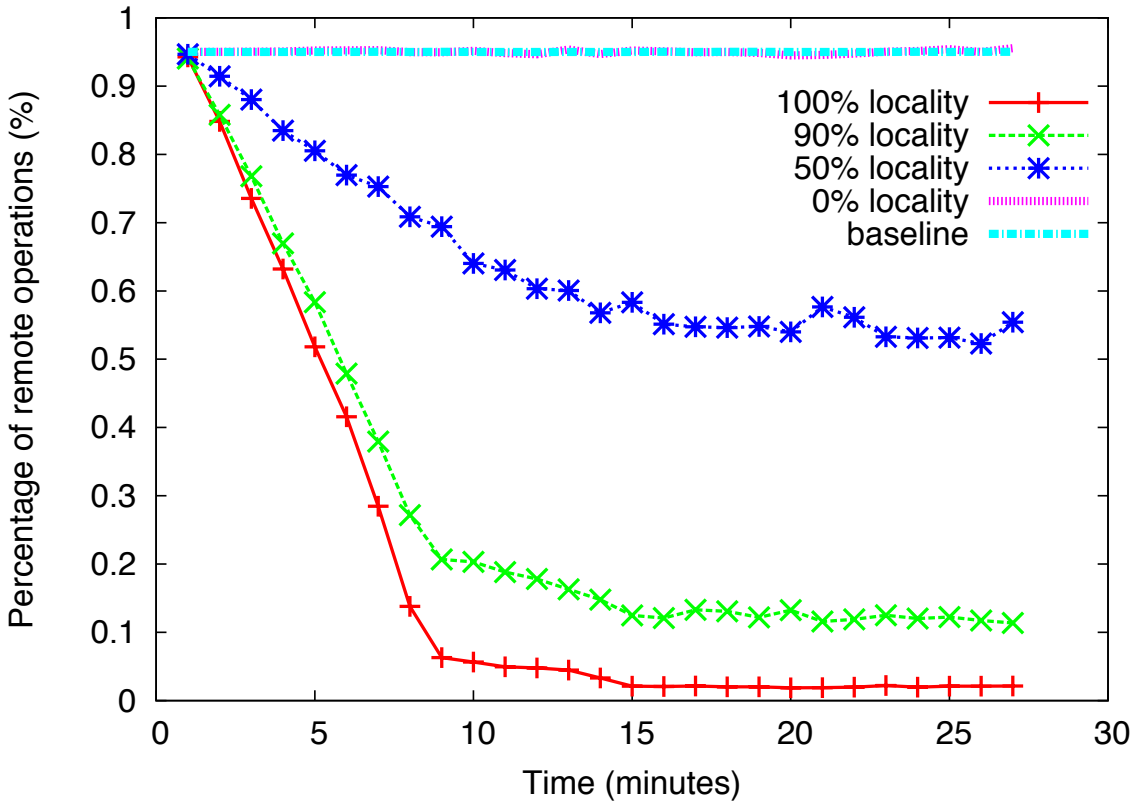
Experimental settings

Integrated in Distributed TM.

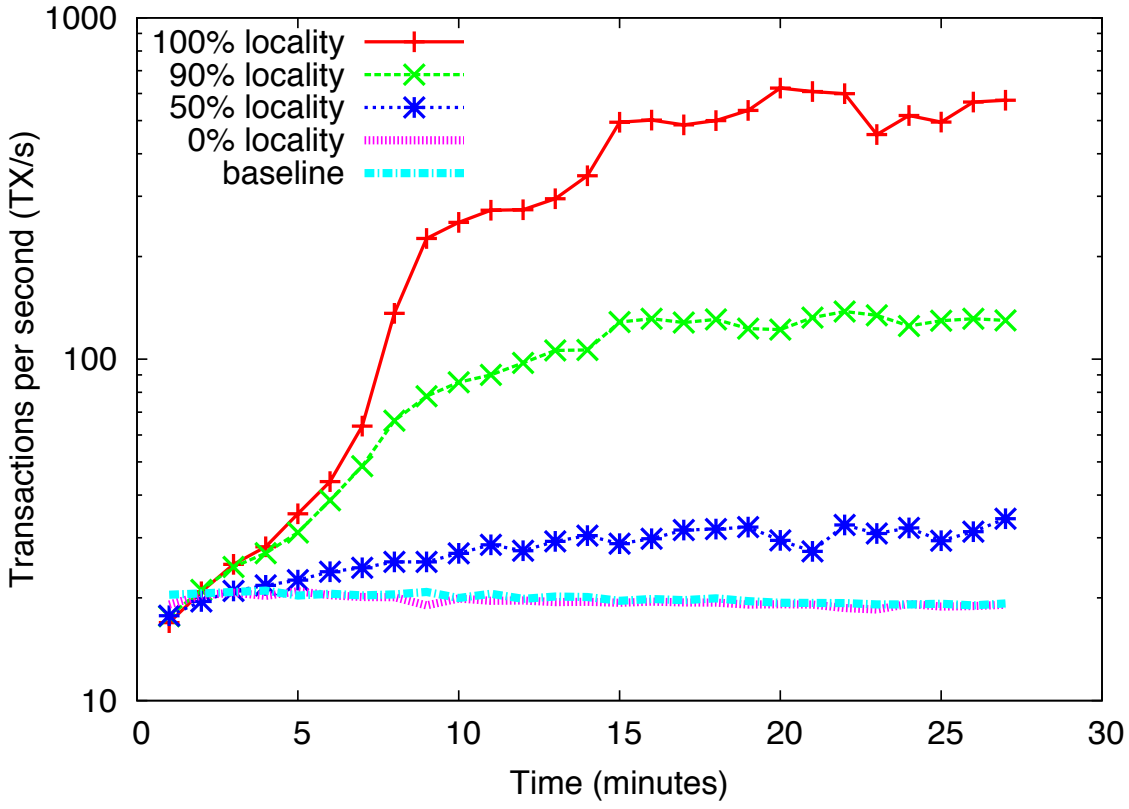
TPC-C benchmark

Modified to induce controllable locality:
Probability p: Transactions access data associated with a given warehouse.
Probability $1 - p$: Transactions acess data associated a random warehouse.

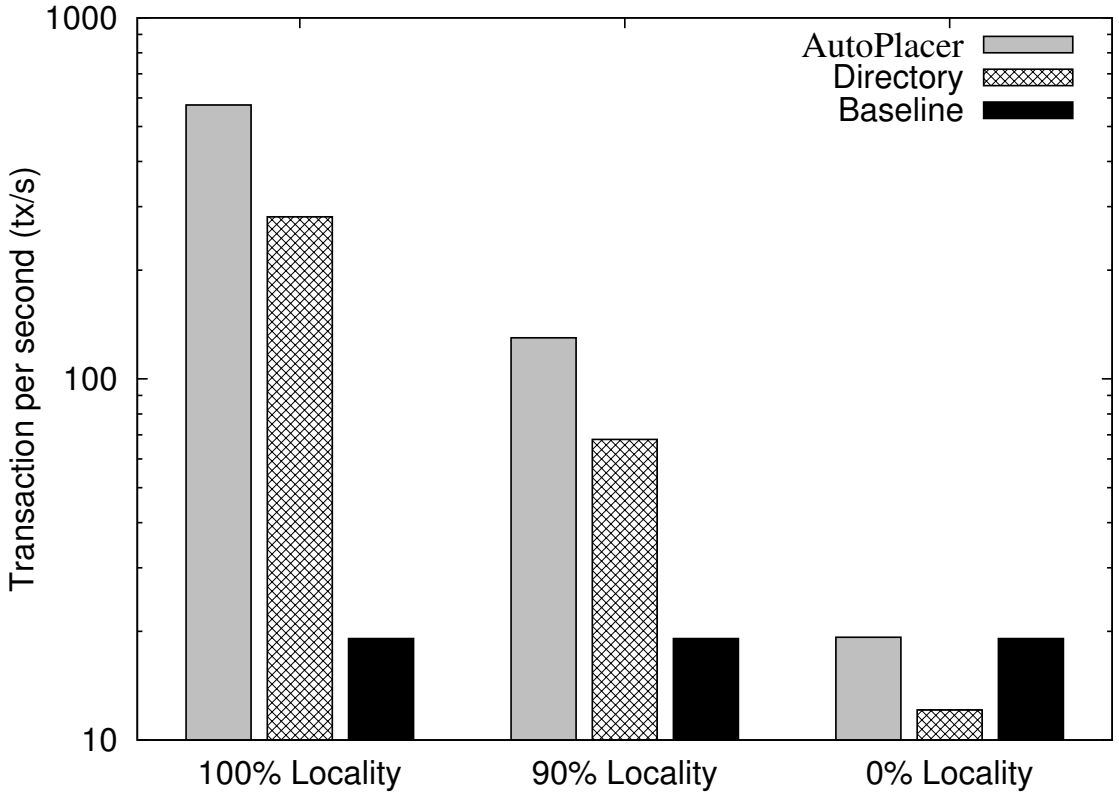
Remote operations



Throughput


1000
100
10
Transactions per second (TX/s)
0
5
10
15
20
25
30
Time (minutes)
100% locality
90% locality
50% locality
0% locality
baseline

Throughput


1000
100
10
Transaction per second (tx/s)
AutoPlacer
Directory
Baseline
100% Locality
90% Locality
0% Locality

Outline

Conclusions

- Hotspot optimization approach
- Consistent Hashing for most items

- Retains local lookups from Consistent Hashing
- Takes advantage of data locality

Thank you