



# Cloud-TM: Scalable, Self-tuning, Transactional Cloud Data Store

**Paolo Romano**

inesc id
lisboa

# Cloud-TM at a glance

**Partners:**


INESC ID (PT)

Algorithmica (IT)


C.I.N.I. (IT)

Red Hat (IE)

**Project coordinator:**
Paolo Romano, INESC ID (PT)

**Duration:**
From June 2010 to May 2013

**Programme:**
FP7-ICT-2009-5 – Objective 1.2

**Further information:**
http://www.cloudtm.eu

# Cloud: the bright side…


Unprecedented scalability levels

Minimize operating costs & carbon footprint via elastic provisioning

Leverage economies of scale for both services providers and users

Lower barriers to entry via usage-based pricing schemes

# …and the dark side!

- Lack of programming models effectively hiding the issues of:
  - concurrency
  - distribution
  - fault-tolerance
  - elasticity

**Complexity**

- Need for mechanisms to ensure efficiency at any scale and for any workload:
  - no-one-size-fits-all solution
  - manual tuning is costly, error prone and suboptimal
  - how to ensure QoS in highly dynamic environment?

**Complexity**

# Main project motivation

# Key project goals

Develop a data-centric PaaS aimed to minimize:

1. **developments costs**:

➪ introducing abstractions aimed to hide complexity

2. **administration costs**:

➪ aiding/replacing sys admins via self-tuning

3. **operational costs**:

➪ maximizing efficiency via self-tuning


# The Cloud-TM Solution

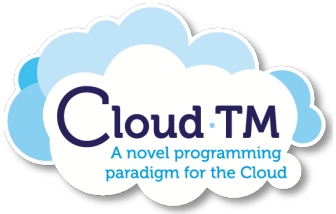
…but first some background…

# From Transactional Memory…

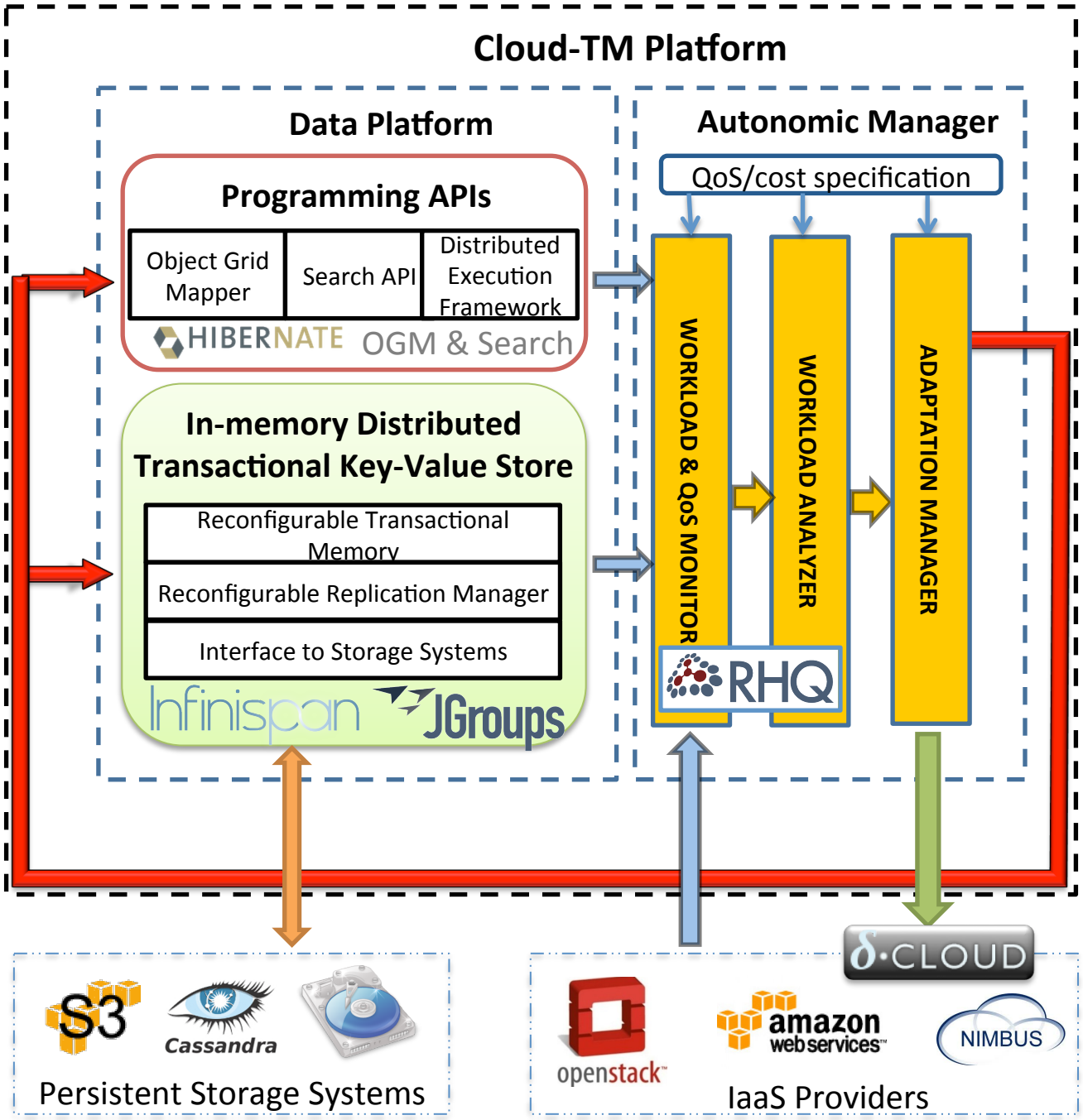

- Transactional Memory (TM):
  - replace locks with atomic transactions in the programming language
  - hide away synchronization issues from the programmer
    - avoid deadlocks, priority inversions, debugging nightmare
    - simpler to reason about, verify, compose
  - **simplify development of parallel applications**

# …to Distributed Transactional Memory…

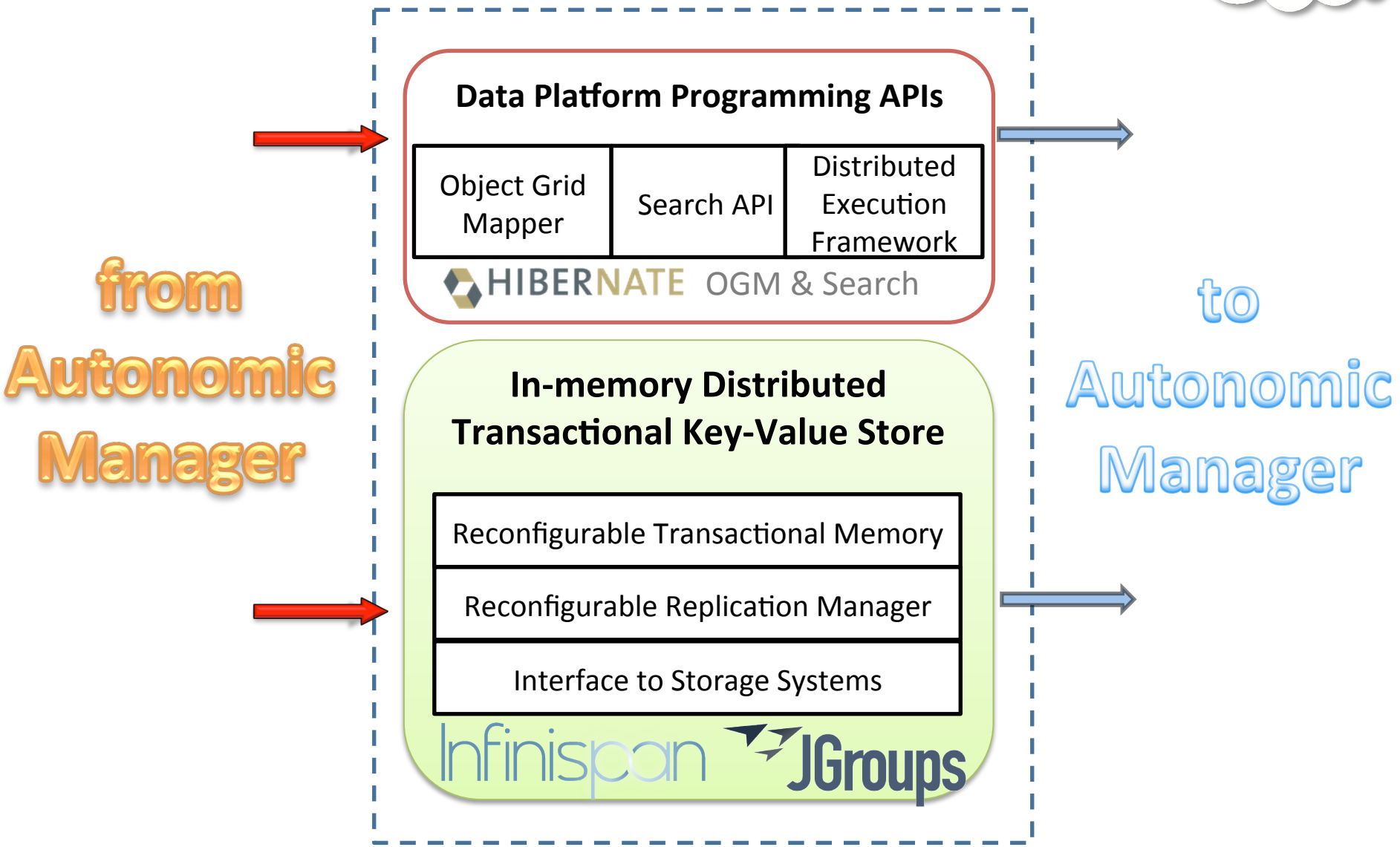

- Distributed Transactional Memory (DTM):
  - extends TM abstraction over the boundaries of a single machine:
    - enhance scalability
    - ensure fault-tolerance
  - maximize scalability and efficiency via:
    - efficient data replication protocols
    - speculation and batching of consistency actions



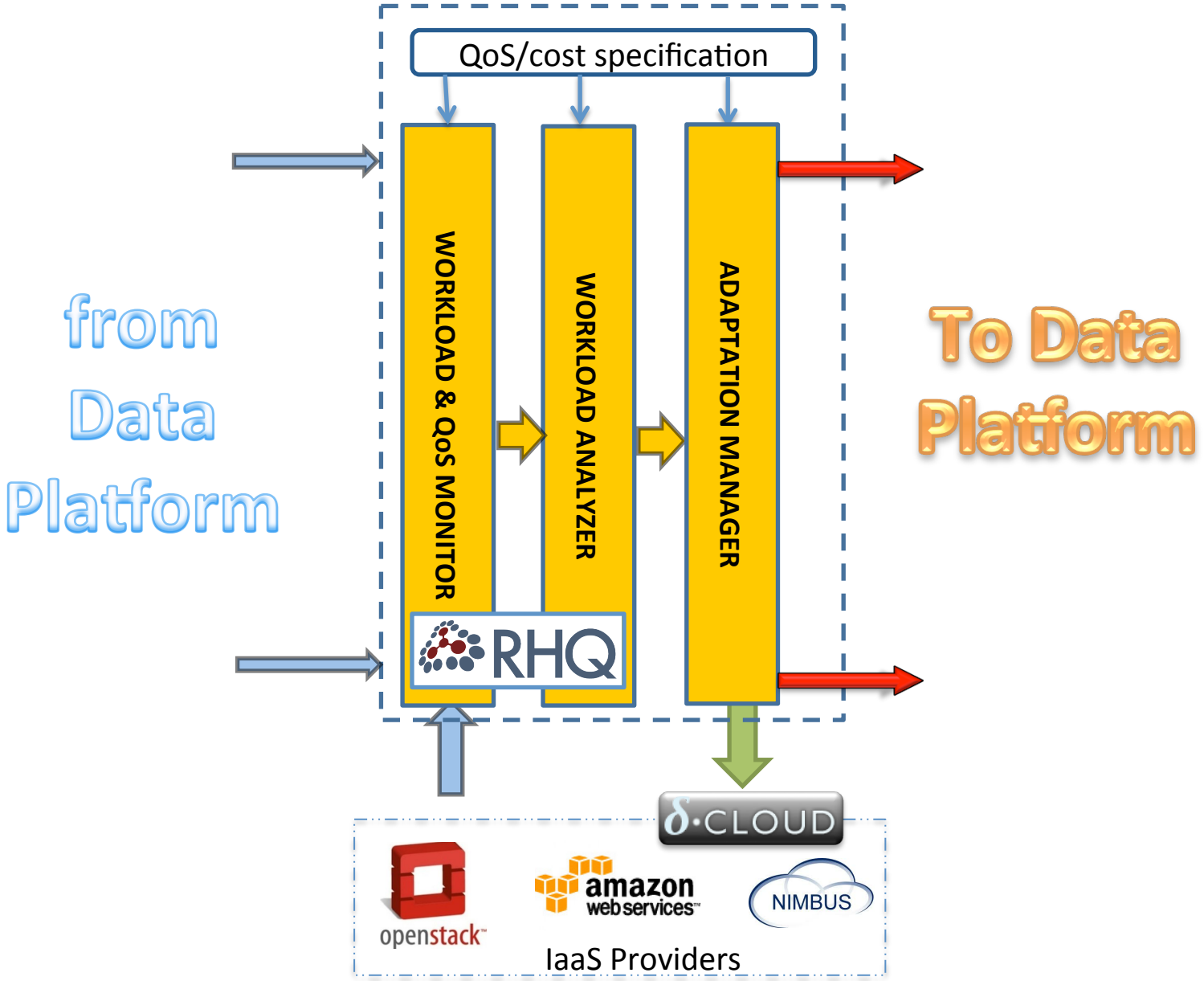

# …to the Cloud-TM platform!


Cloud-TM Platform
Data Platform
Programming APIs
Object Grid Mapper
Search API
Distributed Execution Framework
HIBERNATE OGM & Search
In-memory Distributed Transactional Key-Value Store
Reconfigurable Transactional Memory
Reconfigurable Replication Manager
Interface to Storage Systems
Infinispan
JGroups
Autonomic Manager
QoS/cost specification
WORKLOAD & QoS MONITOR
WORKLOAD ANALYZER
ADAPTATION MANAGER
RHQ
S3
Cassandra
Persistent Storage Systems
δ-CLOUD
openstack
amazon web services
NIMBUS
IaaS Providers

# Cloud-TM Data Platform



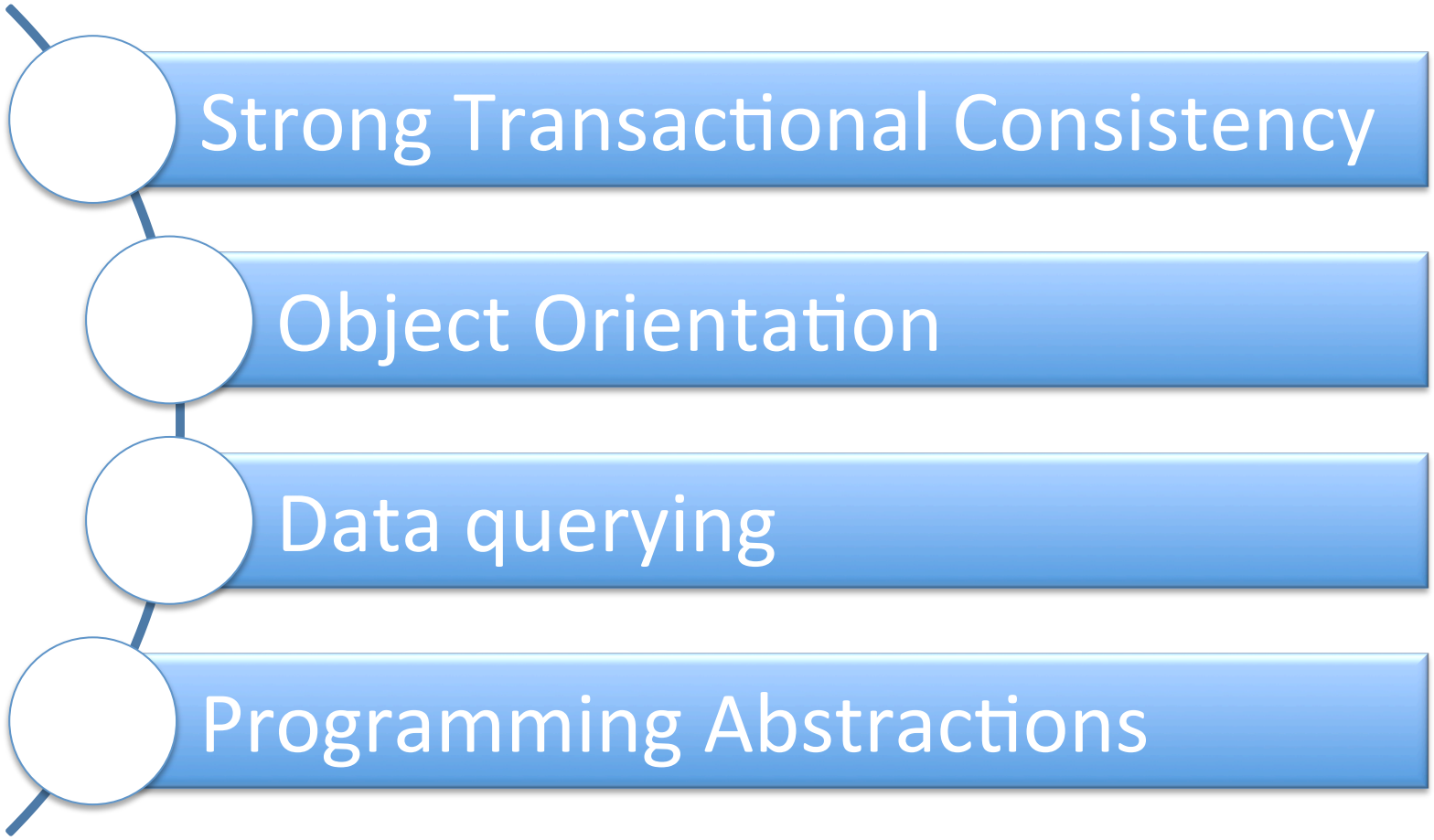
# Autonomic Manager



# The Cloud-TM approach

- Innovation in three main areas:

# Programming Model



## Programming model

# Strong transactional consistency


- Transactional manipulation of *in-memory* objects:
  - atomicity and isolation guarantees
  - primary mechanism for durability ➔ **replication**

- Goal:
  - **shelter programmers from complexity** of weak consistency models

## Programming model

# Object orientation


- Full support for object-oriented data model:
  - transparent mapping of OO model to Key-Value model
- Integration with the Java ecosystem:
  - JAVA Persistence API (JPA)
    - the **STANDARD** way of persisting JAVA objects (Hibernate OGM)
  - Fenix Framework
    - higher level abstraction API, allows for more agile experimentation
- …as well as with Ruby!

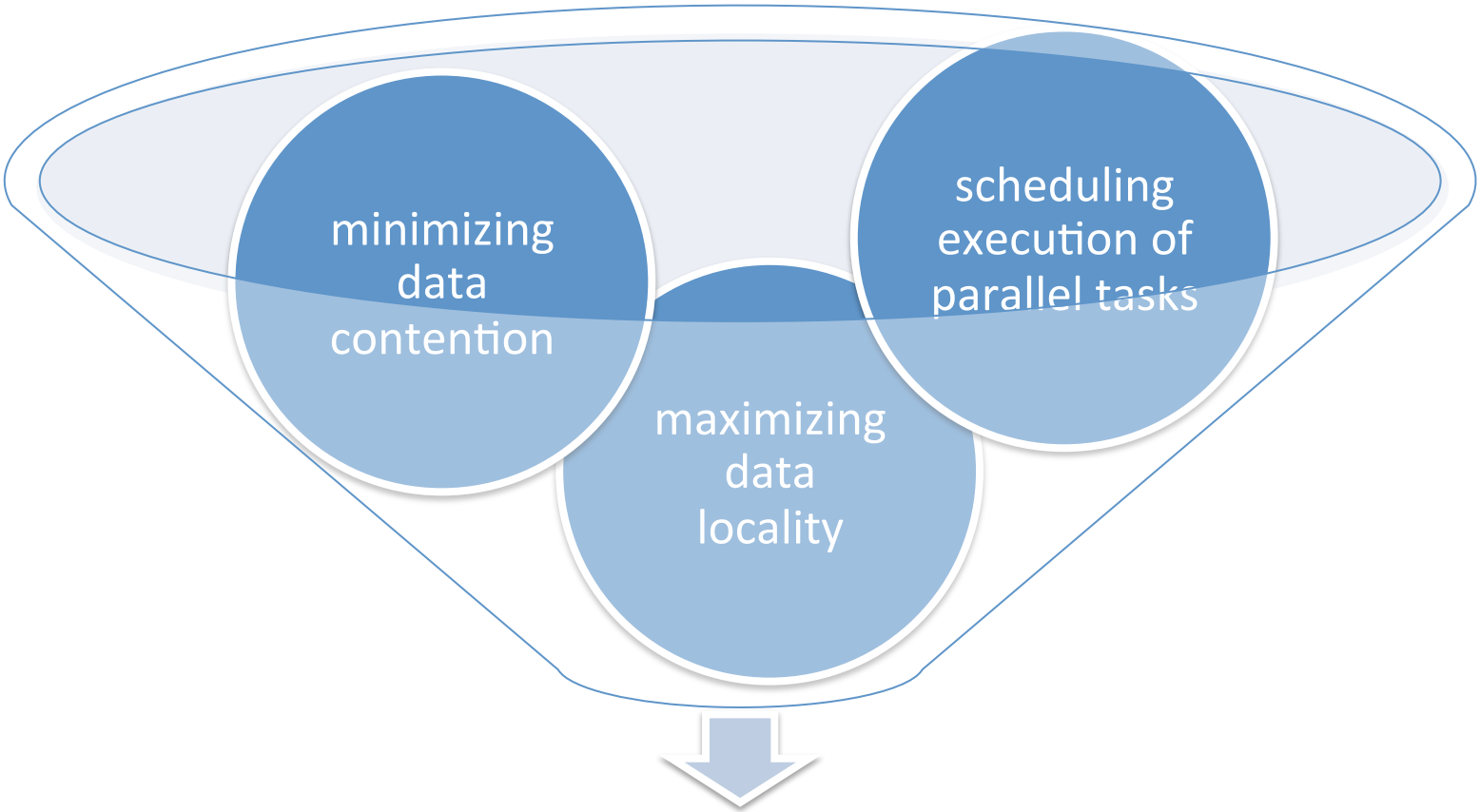
## Programming model

# Data Querying


- Support for querying object-oriented domain:
  - automatic indexing of the data maintained by the platform
  - subset of industry standard JP-QL interface:
    - exact/approximate values queries
    - polymorphic queries
    - by range, aggregation functions, by association

## Programming model

# Programming Abstractions

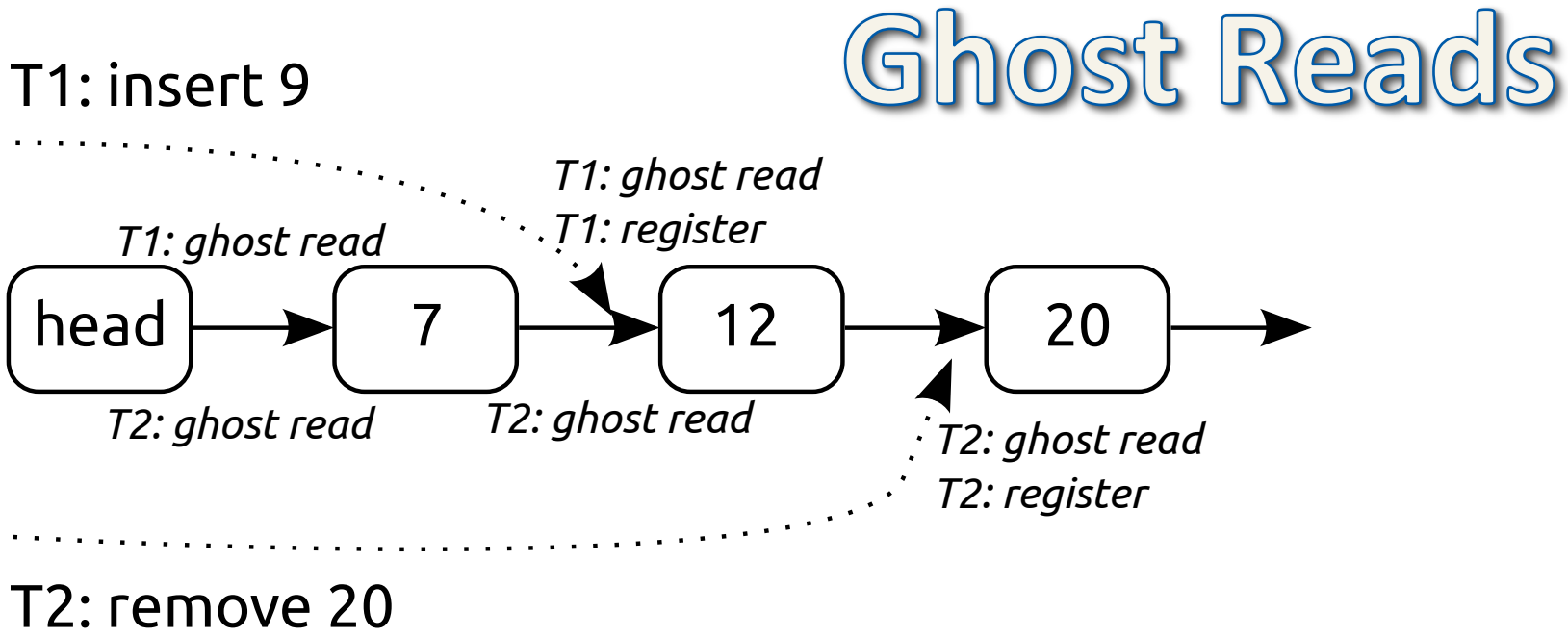



**simpler and better performing cloud applications**

## Programming abstractions

# Minimize data contention

- Let expert programmers exploit appl. semantics
- Avoid aborting txs upon "benign" conflicts
- Killer application: **collections**



## Programming abstractions

# Maximize data locality


- Locality hints (LH):
  - let programmers specify which objects' attributes should be used to determine their placement
  - LH define a multi-dimensional hyperspace
  - Objects with common LHs get co-located

- Object → Point in LH space → Platform Node

programmer defined

automatized by the platform

## Programming abstractions

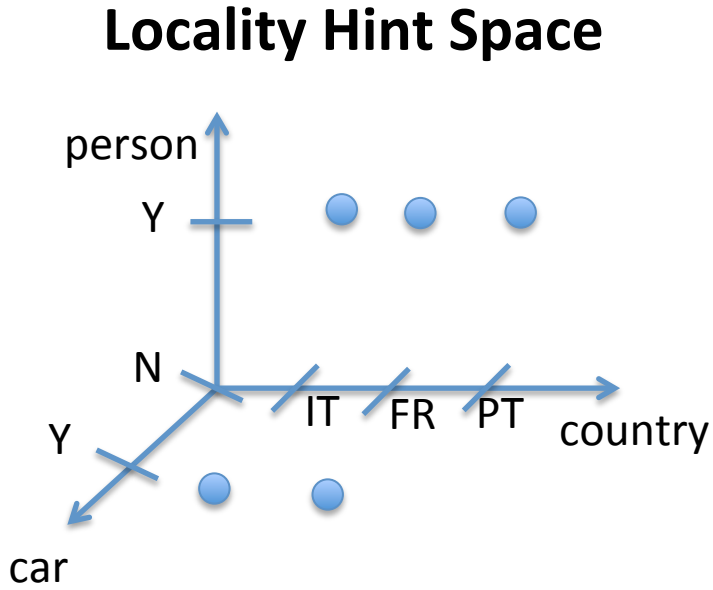
# Maximize data locality

- Object ➔ Point in LH space ➔ Platform Node

programmer defined

automatized by the platform

```
Class Person {
        @localityHint
        Country nation;
        String Name, Surname;
}
Class Car{
        @localityHint
        Country nation;
        String Model, Descrip;
}
```



## Programming abstractions

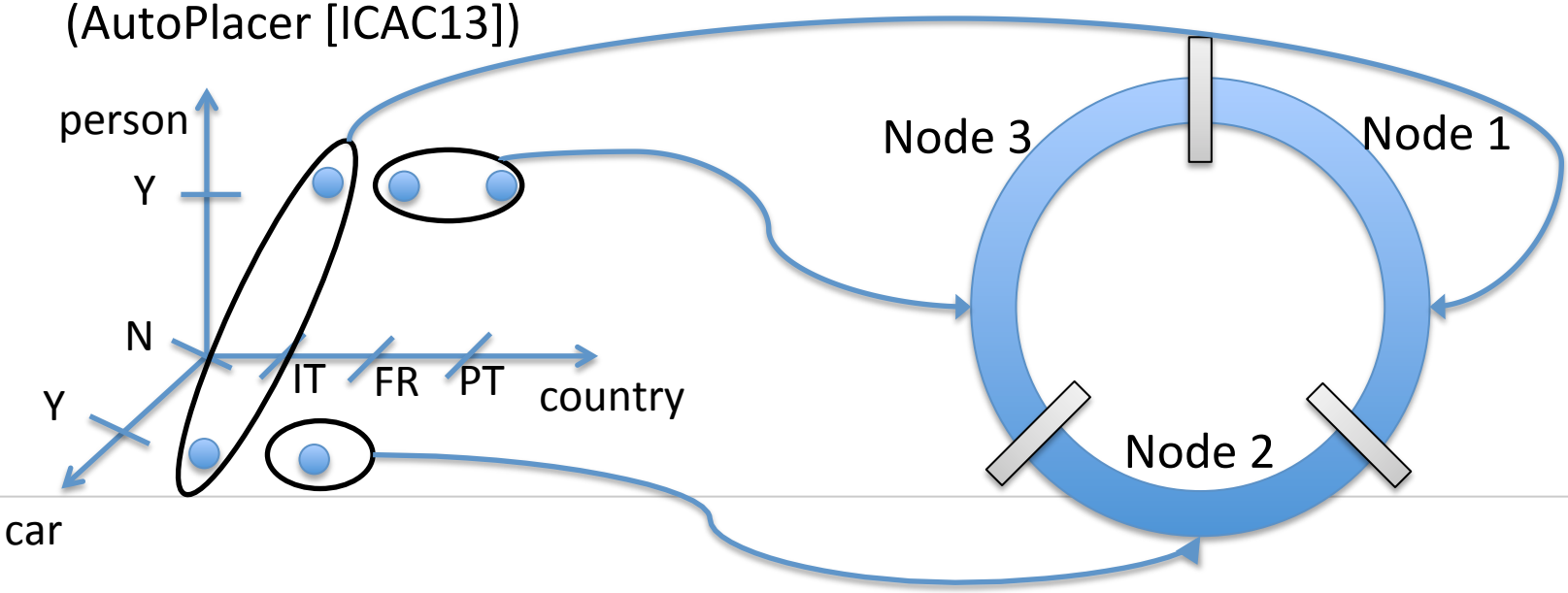
# Maximize data locality

- Object → Point in LH space → Platform Node

programmer defined

automatized by the platform

1. based on consistent hashing (random)
2. based on nodes' access patterns (AutoPlacer [ICAC13])



# Programming abstractions
# …more tricks!


**Delayed Actions** [SRDS13]:

- postpone data manipulations till commit time

**Distributed Execution Framework** [INESC13]:

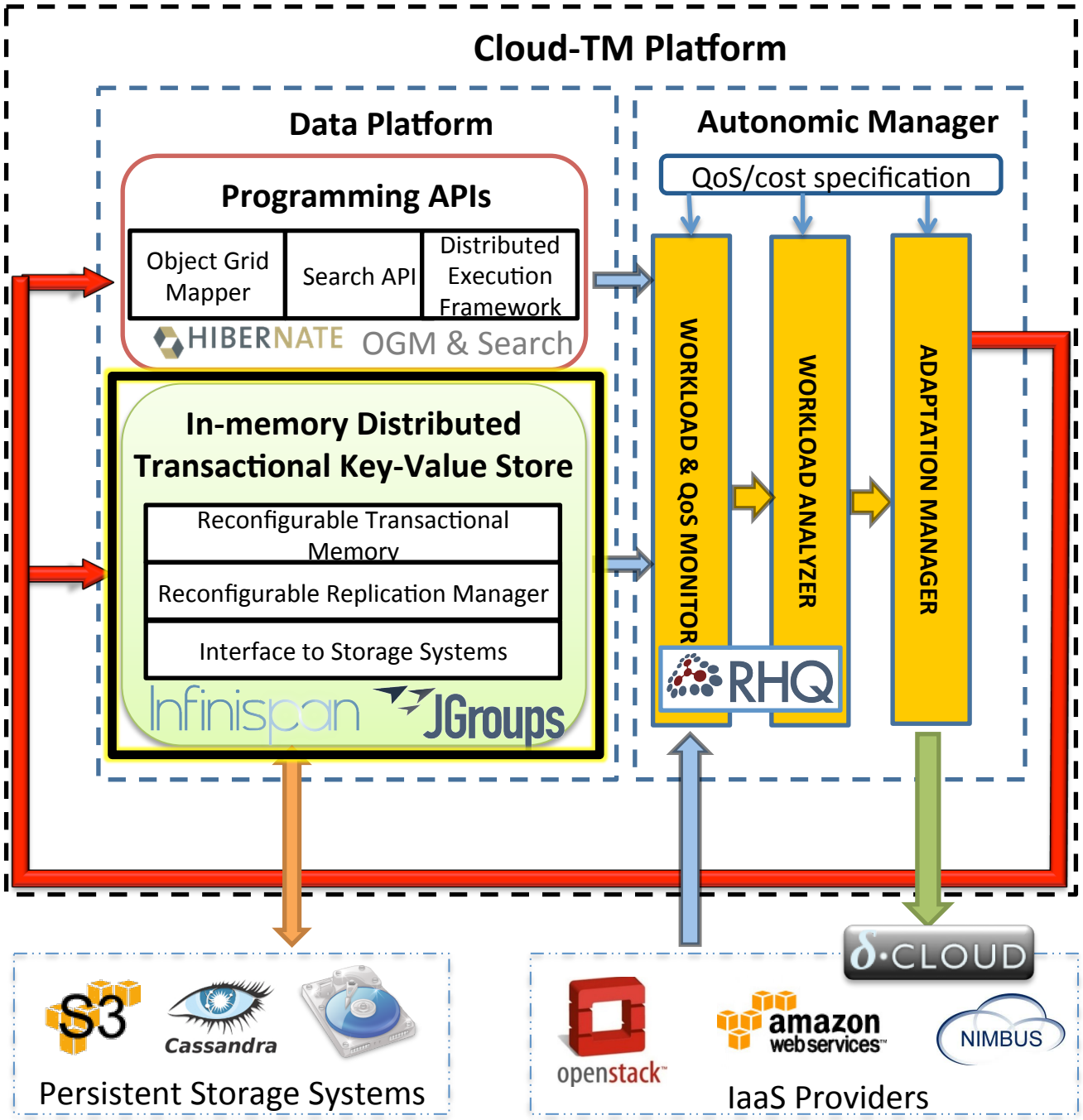
- exploit locality to route transactions to the nodes where the data is stored

**Transaction Migration** [INESC13]:

- allow a tx running on node n to execute arbitrary code on a node n' and resume execution on n

# …to the Cloud-TM approach

- Innovation in three main areas:

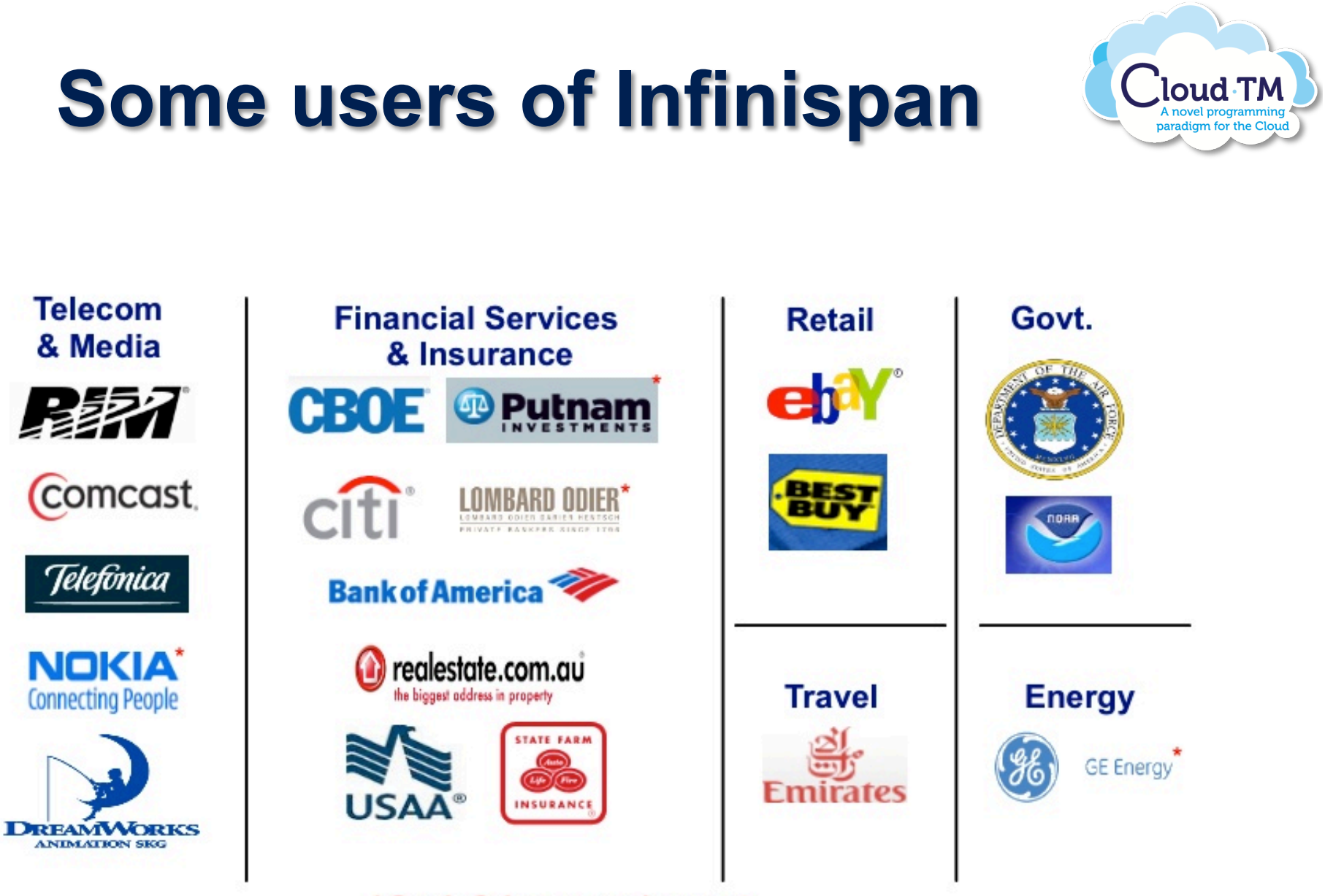

Cloud-TM Platform
Data Platform
Programming APIs
Object Grid Mapper
Search API
Distributed Execution Framework
HIBERNATE OGM & Search
In-memory Distributed Transactional Key-Value Store
Reconfigurable Transactional Memory
Reconfigurable Replication Manager
Interface to Storage Systems
Infinispan
JGroups
Autonomic Manager
QoS/cost specification
WORKLOAD & QoS MONITOR
WORKLOAD ANALYZER
ADAPTATION MANAGER
RHQ
S3
Cassandra
Persistent Storage Systems
δ-CLOUD
openstack
amazon web services
NIMBUS
IaaS Providers

# Infinispan

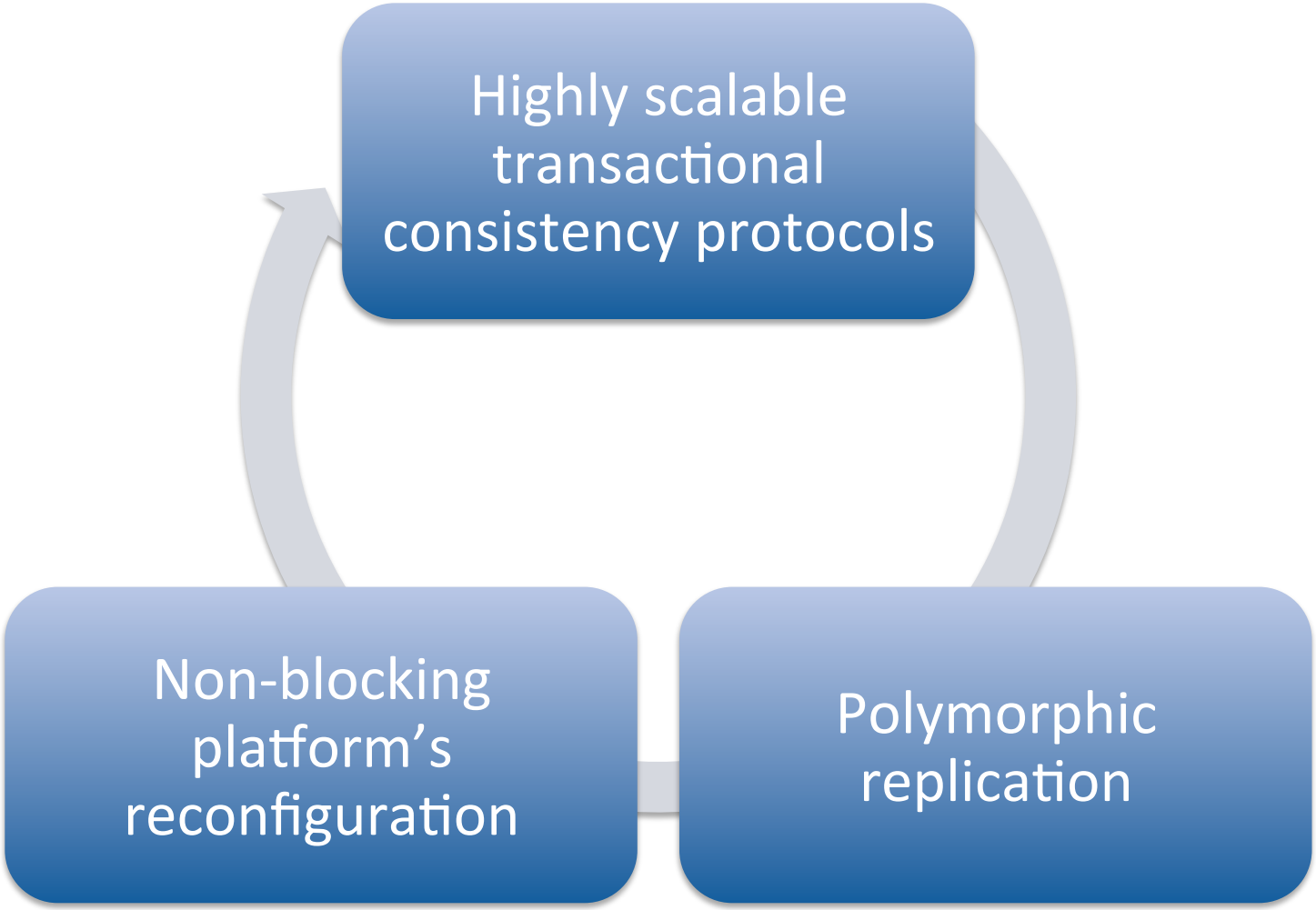
- Cloud-TM reference DTM platform:


  - open source project by JBoss/Red Hat
  - in-memory transactional key-value store

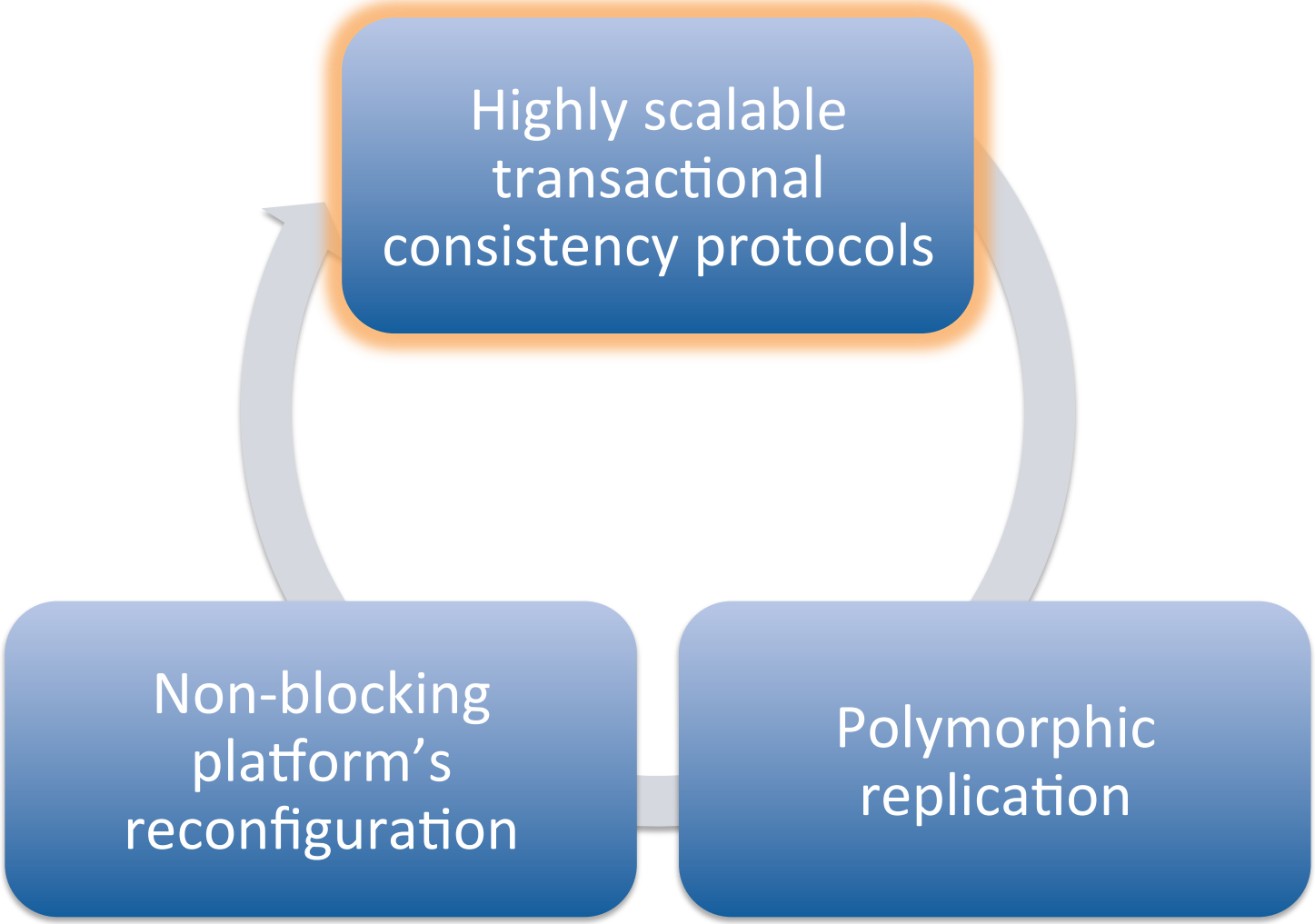
# Some users of Infinispan



# Distributed Data Management



# Distributed Data Management



# Highly scalable replication


- Genuine partial replication schemes:
  - #copies of data items << #nodes in the system
  - involve in the transaction processing only nodes maintaining copies of accessed data:
    - no centralized components/no global broadcasts
- Explored various tradeoffs in the consistency spectrum:
  - Weak consistency: TOM [PRDC11]
  - Strong consistency: GMU & extensions [ICDCS12, Middleware12, SRDS13…]

# Scalability or Consistency?


- Consistency is often sacrified in cloud data stores to maximize scalability, e.g.:
  - eventual consistency:
    - a posteriori application driven reconciliation
  - read committed/repeatable read isolation levels:
    - applications can observe non-serializable snapshots
- Scalability comes  at the cost of complexity

# Scalability and consistency?

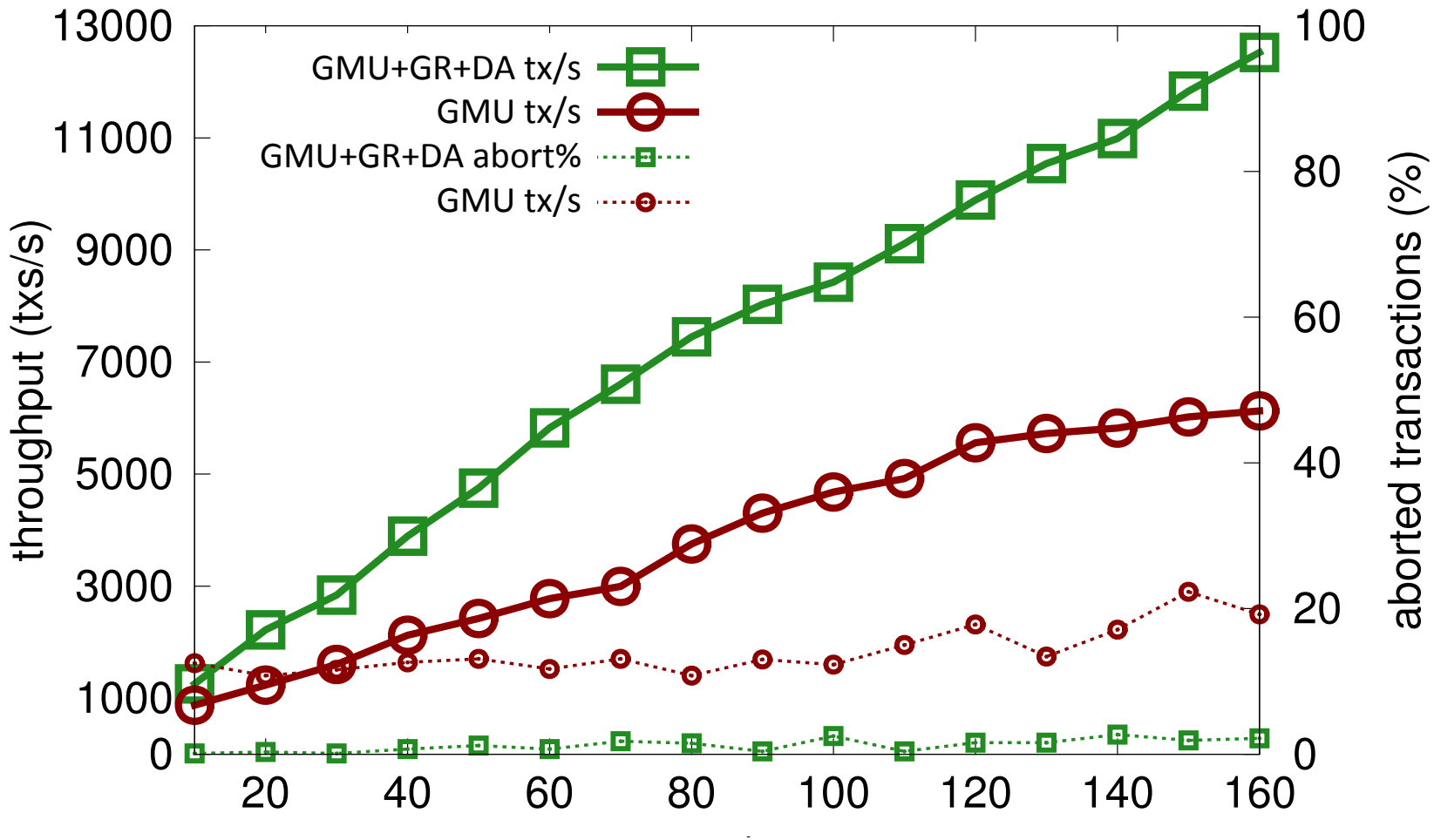

*Are there any sweet spots in the scalability vs consistency trade-off?*

- Do we need to give up consistency to achieve scalability?

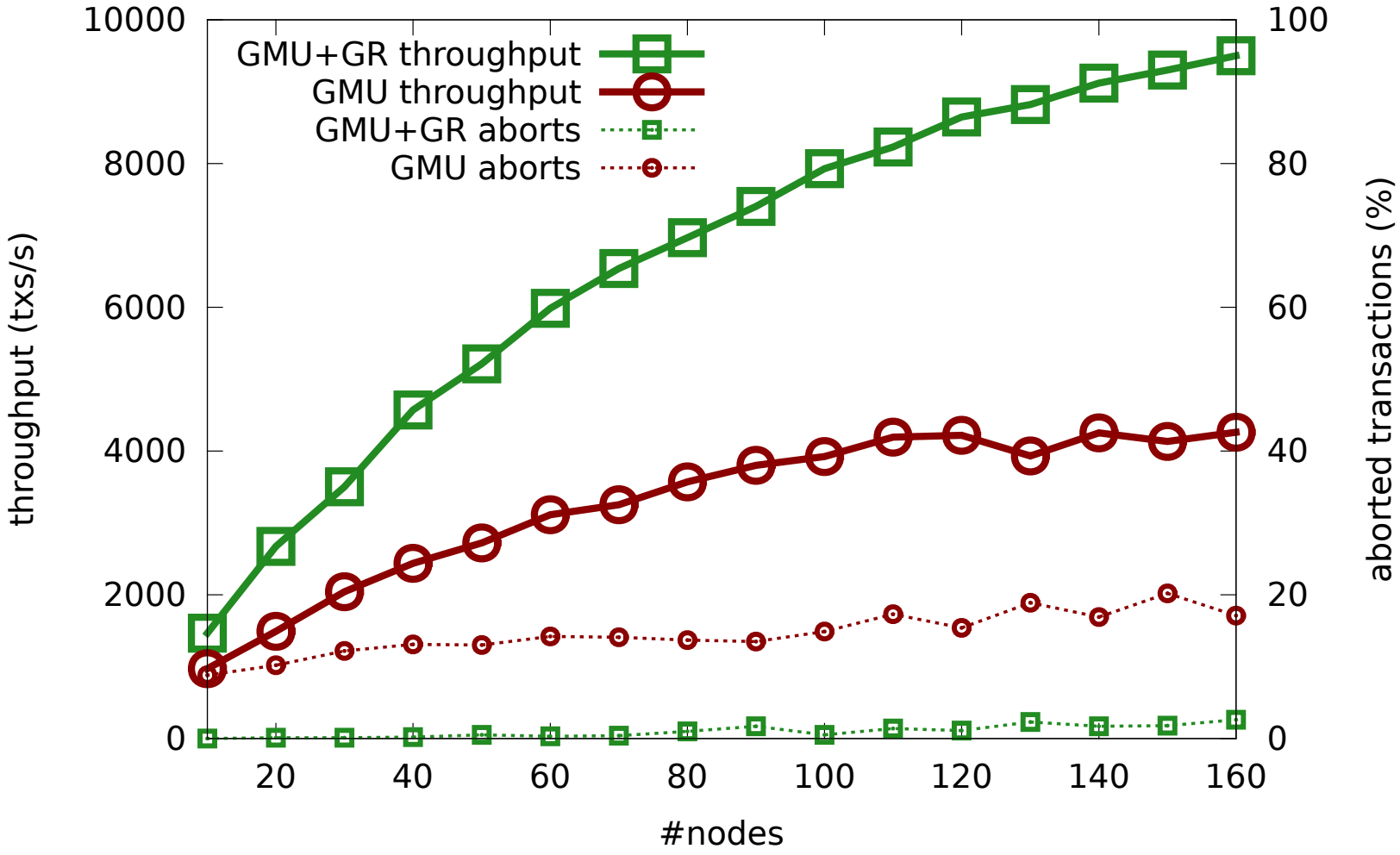
# GMU [ICDCS12]


- Genuine
  - fully decentralized/no centralized coordinator
- Multiversion
  - read-only txs never aborted nor validated
- (Extended) Update Serializable:
  - 1-Copy Serializability for update tx
  - Read-only txs:
    - must observe *some* serializable snapshot
    - can witness non-conflicting update txs in diff. orders
  - compliant with ANSI SQL SERIALIZABLE

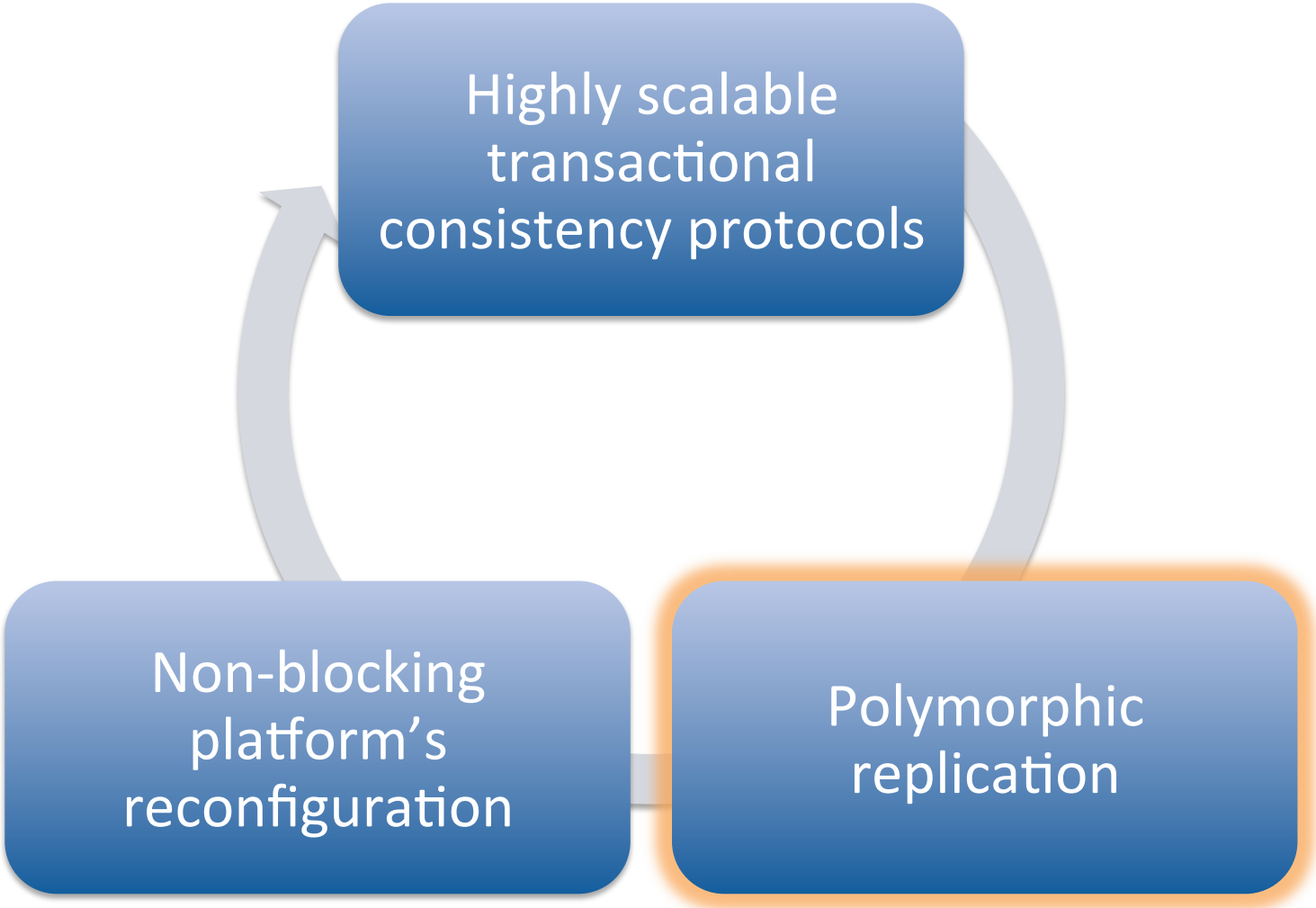
# Vacation


GMU+GR+DA tx/s
GMU tx/s
GMU+GR+DA abort%
GMU tx/s
throughput (txs/s)
aborted transactions (%)
13000
11000
9000
7000
5000
3000
1000
0
100
80
60
40
20
0
20
40
60
80
100
120
140
160

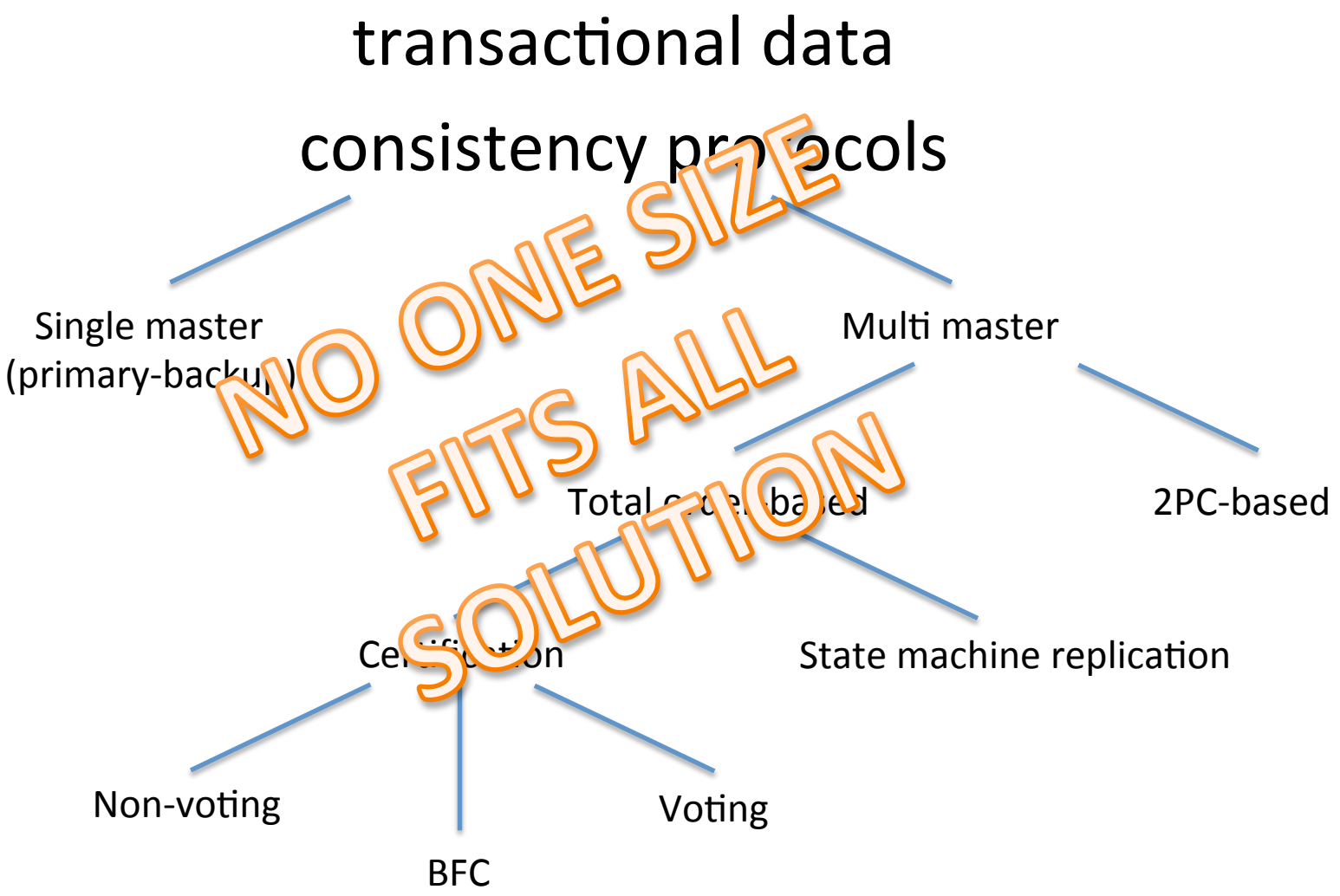
# Skiplist


GMU+GR throughput
GMU throughput
GMU+GR aborts
GMU aborts
throughput (txs/s)
aborted transactions (%)
#nodes
10000
8000
6000
4000
2000
0
100
80
60
40
20
0
20
40
60
80
100
120
140
160

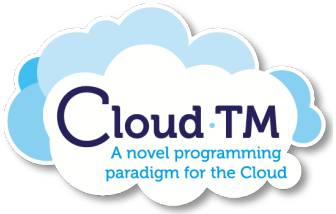
# Distributed Data Management



# The search for the holy grail



# No one size fits all

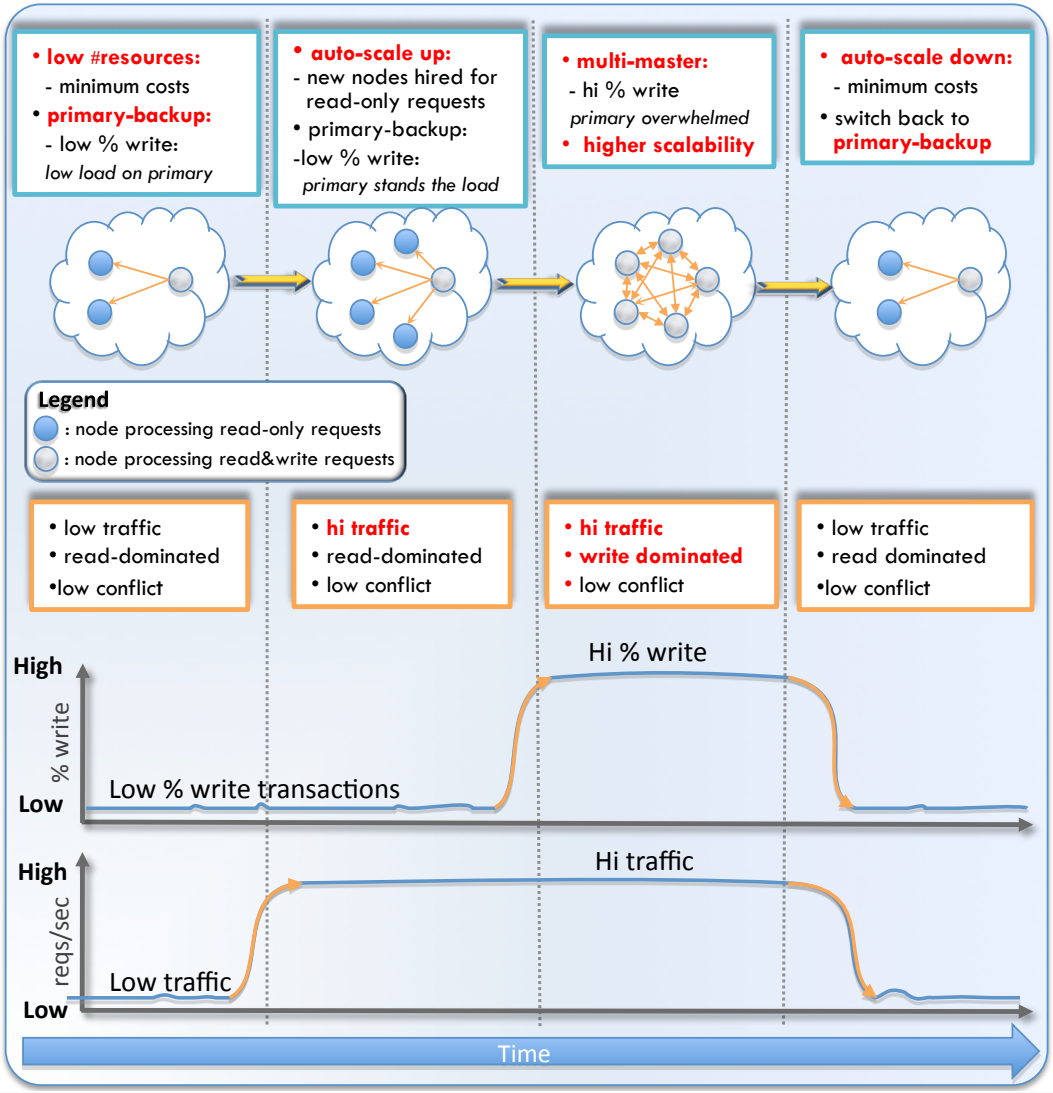



- Existing solutions are optimized for specific workload/scale scenarios

- In **dynamic** environments where both:
  1. the workload characteristics, and
  2. the amount of used resources

  vary over time, **self-tuning is the only way to achieve optimal efficiency**

# The Cloud-TM approach

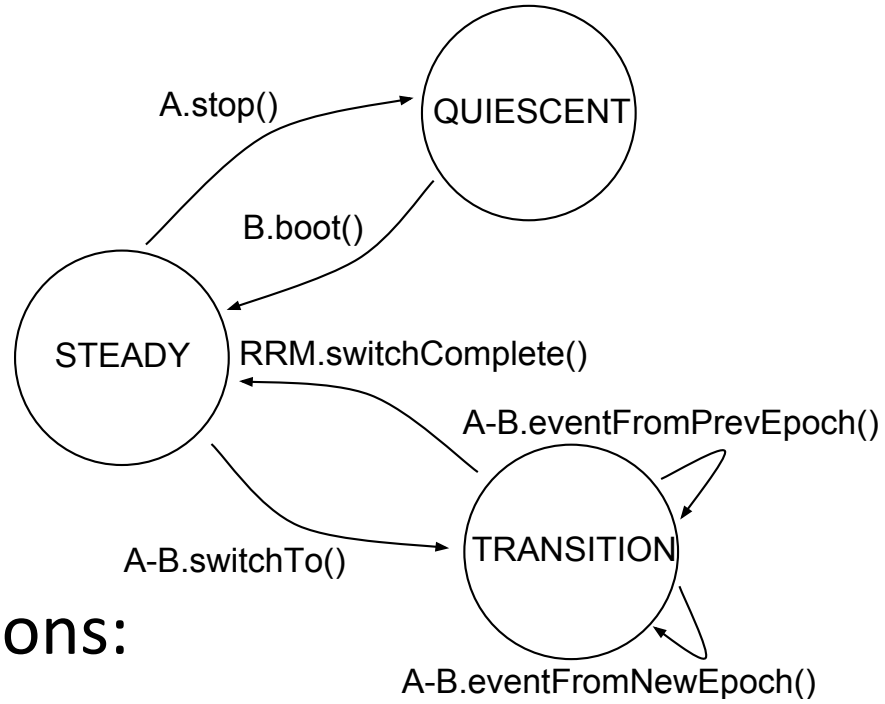


# MorphR [Middleware11, DSN13]

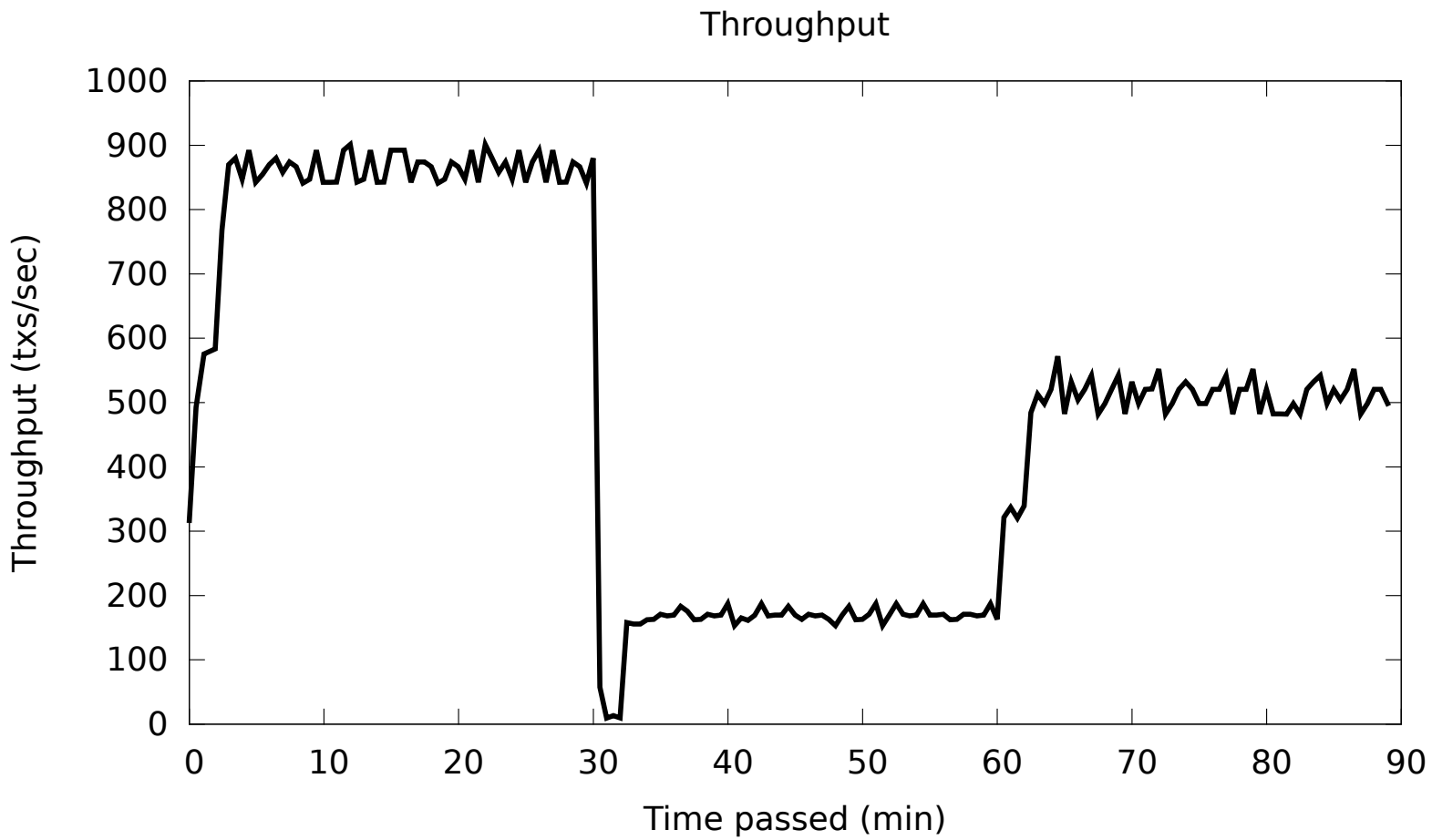

- Generic framework supporting **dynamic switching** between **arbitrary replication protocols**
- Protocol switching phases encapsulated in abstract FSMs with neatly defined interfaces
- Problem of determining the right replication protocol is delegated to the Cloud-TM Autonomic Manager

# Key idea

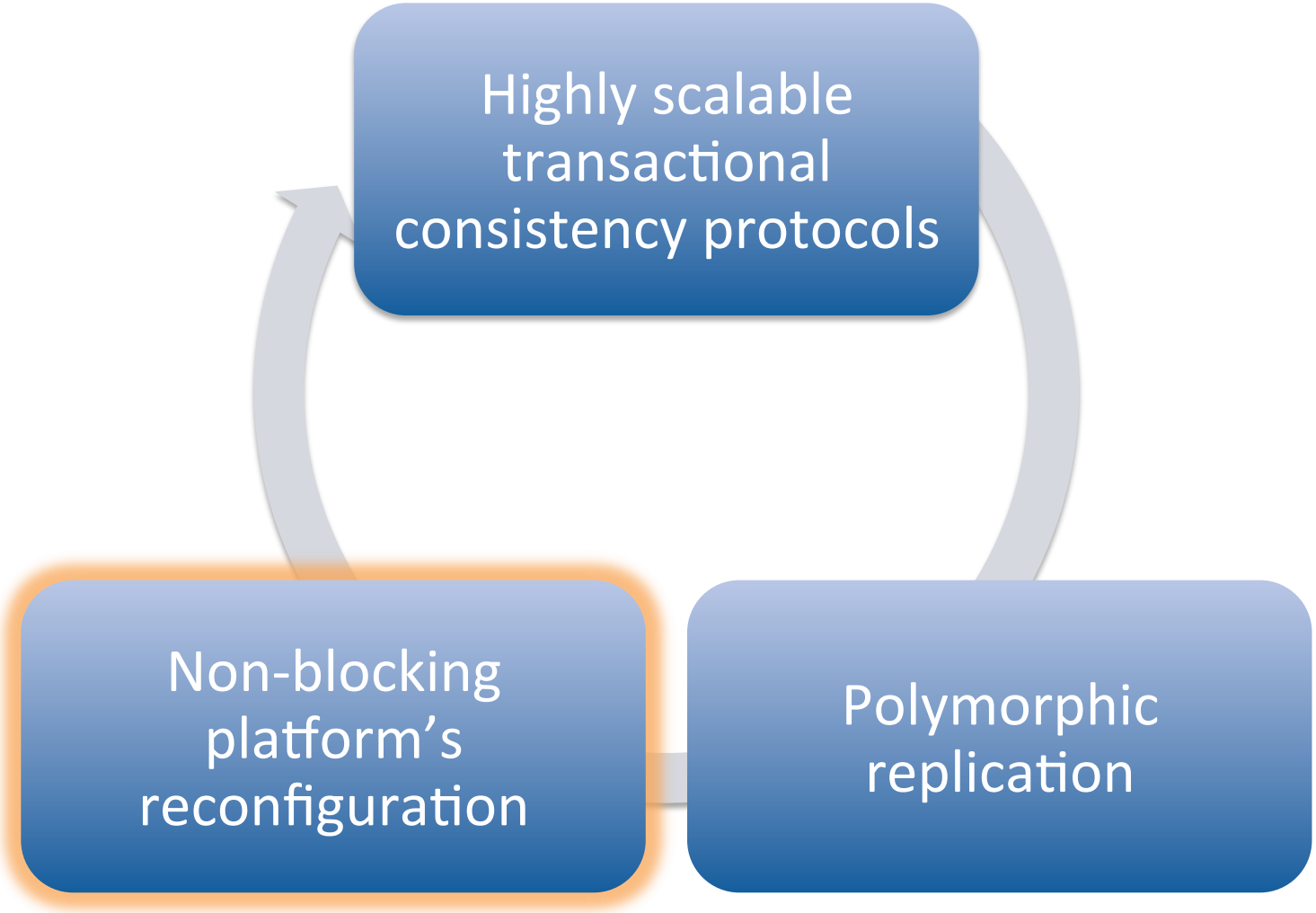
- Support for:
  - blocking (stop&go) reconfigurations:
    - **+ generic**
    - **- less efficient**
  - non-blocking reconfigurations:
    - **+ efficient**
    - **- specific for each pair of protocol**



# MorphR in action



# Distributed Data Management

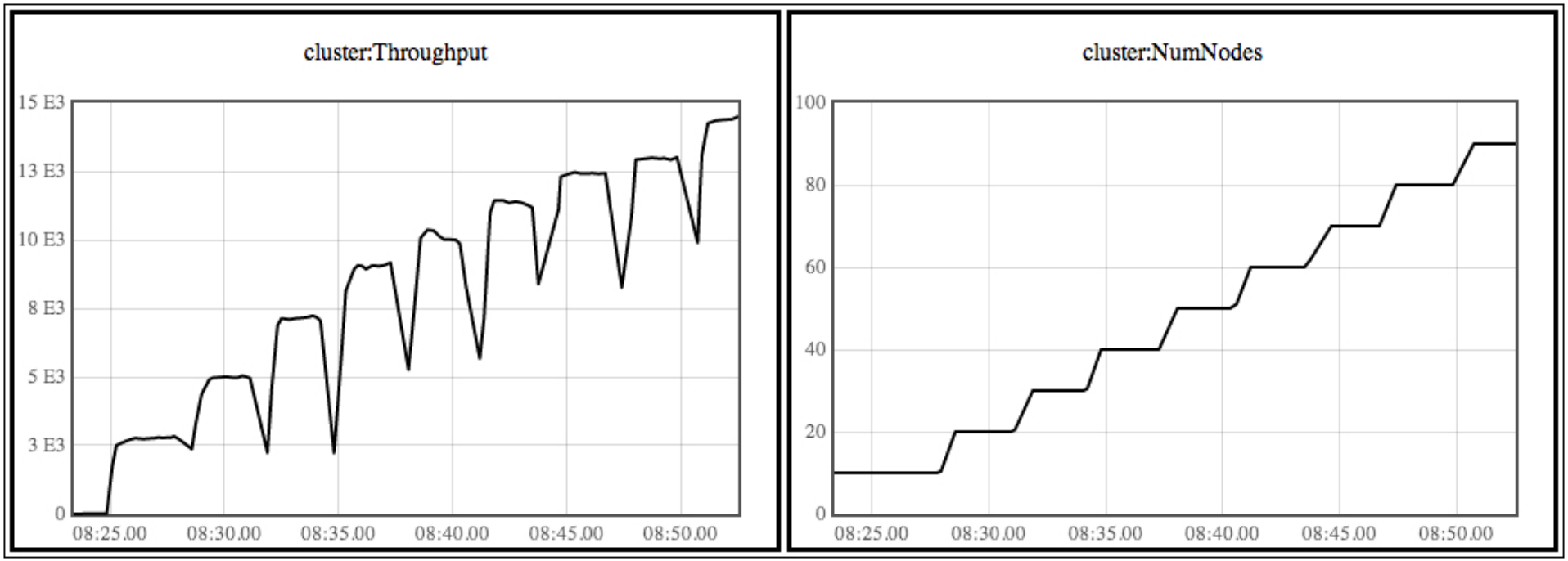


# Non-blocking reconfigurations

- Adding/removing nodes while transactions are in progress is not a trivial issue:
  - data may have to be migr… nodes
    - a process th…
    - …
  - …we want to
    - …rmance:
      - …nsactions should not be aborted
      - new transactions should progress w/o impairment

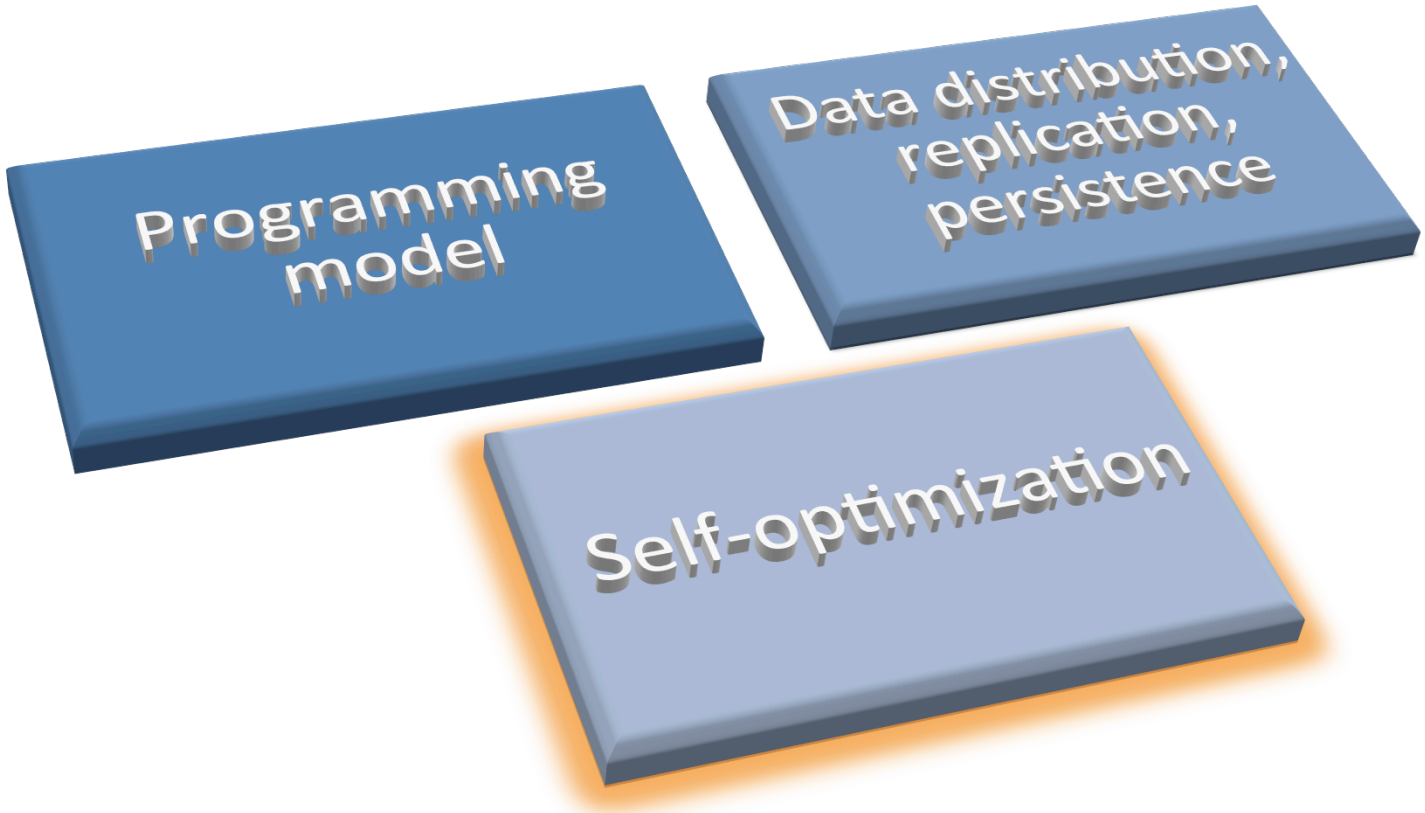
…of course transactional consistency need to be guaranteed whatsoever!

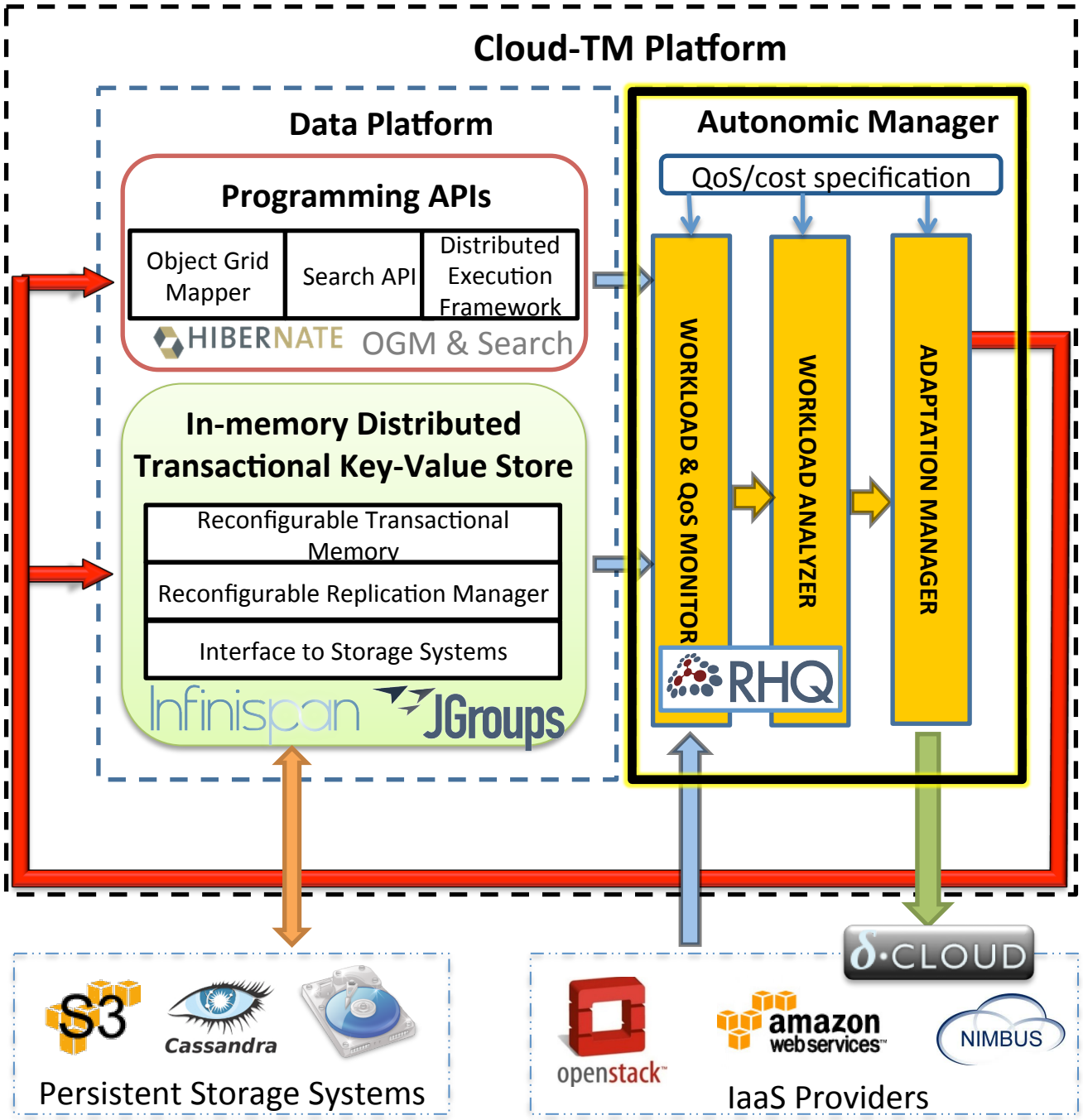
# Non-blocking state transfer



# …to the Cloud-TM approach

- Innovation in three main areas:




Cloud-TM Platform
Data Platform
Programming APIs
Object Grid Mapper
Search API
Distributed Execution Framework
HIBERNATE OGM & Search
In-memory Distributed Transactional Key-Value Store
Reconfigurable Transactional Memory
Reconfigurable Replication Manager
Interface to Storage Systems
Infinispan
JGroups
Autonomic Manager
QoS/cost specification
WORKLOAD & QoS MONITOR
WORKLOAD ANALYZER
ADAPTATION MANAGER
RHQ
S3
Cassandra
Persistent Storage Systems
δ-CLOUD
openstack
amazon web services
NIMBUS
IaaS Providers

# Self-optimization


pervasive workload and performance monitoring

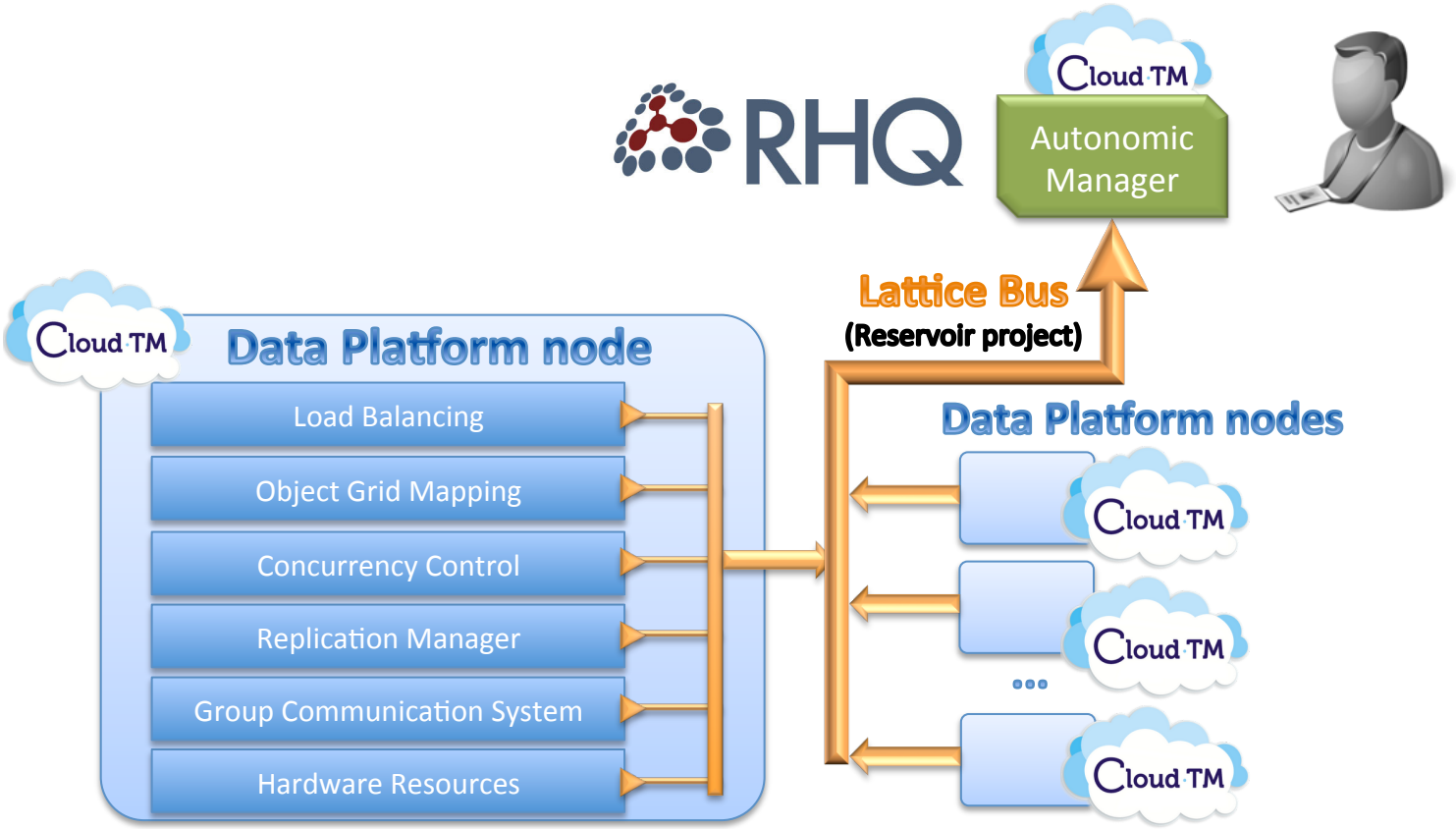
innovative performance forecasting methodologies

multi-objective optimization

QoS/cost driven self-optimization

# Self-optimization

- Pervasive workload and performance monitoring
  - workload characterization across all platform's layers



# Self-optimization

- Pervasive workload and performance monitoring
  - lightweight algorithms from stream analysis literature to pinpoint hot spots for data locality and contention

Top-K Abort Inducing Keys

| Key | Freq. |
|---|---|
| Balance_101 | 12345 |
| Balance_202 | 654 |
| Stock_Item_234 | 543 |
| … | |

Top-K Remotely Acc. Keys

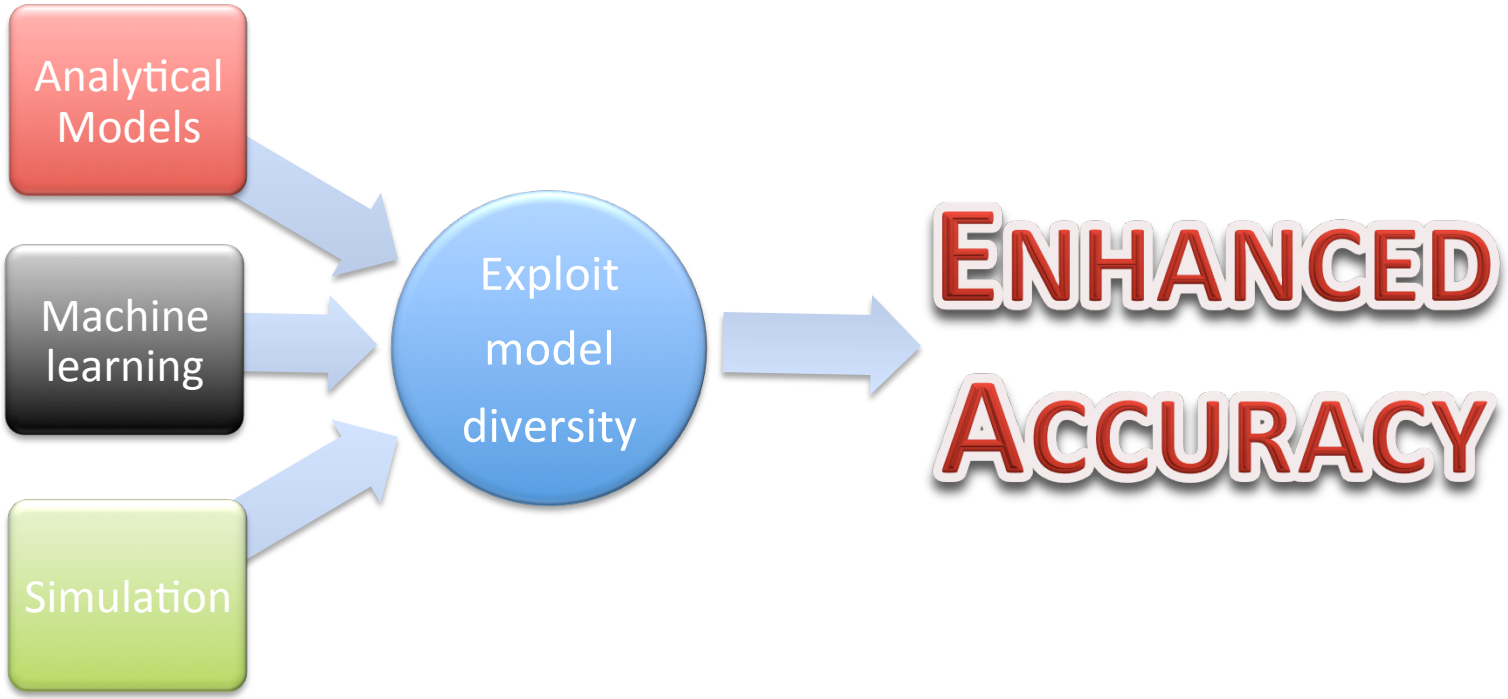
| Key | Freq. |
|---|---|
| Inventory_132 | 45321 |
| Agent_432 | 12302 |
| Stock_Item_234 | 9000 |
| … | |

Top-K Locally Acc. Keys

| Key | Freq. |
|---|---|
| Balance_101 | 12345 |
| Balance_202 | 654 |
| Stock_Item_234 | 543 |
| … | |

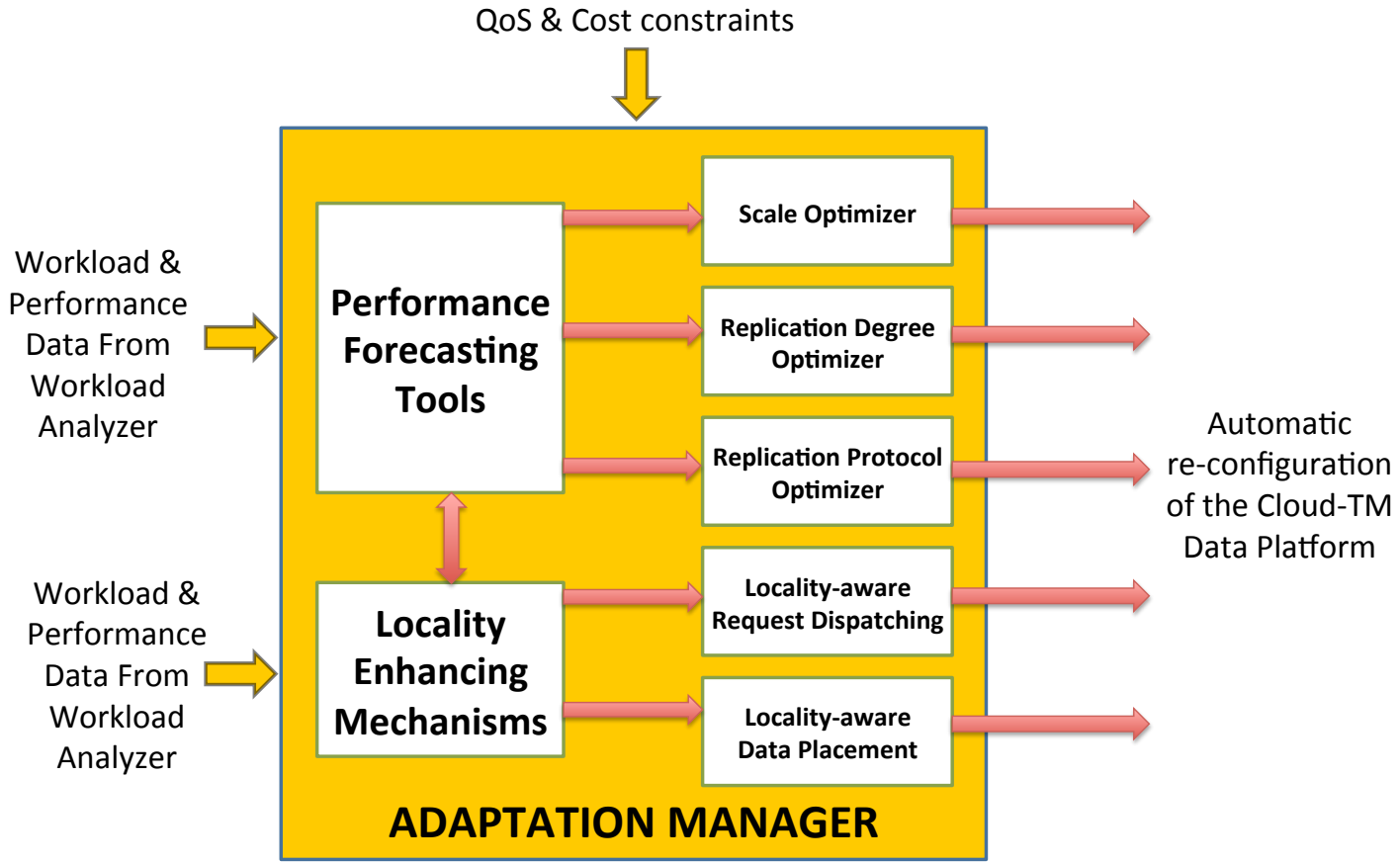
# Self-optimization

- innovative performance forecasting methodologies
  - novel methodologies for the performance modeling of distributed transactional platforms



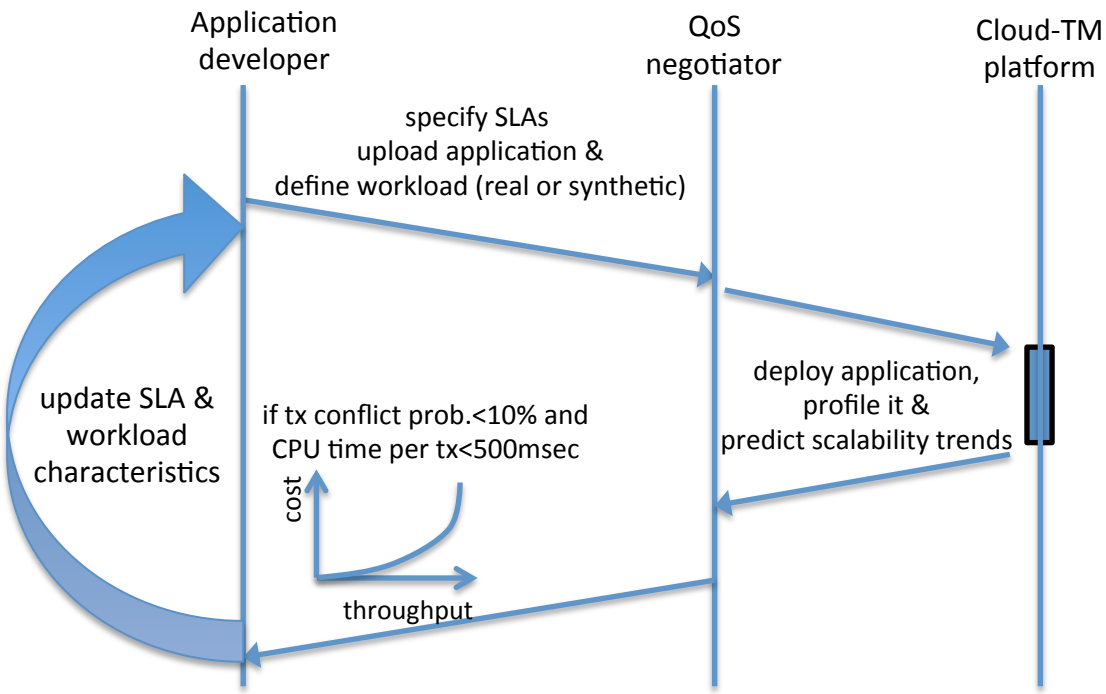
# Self-optimization

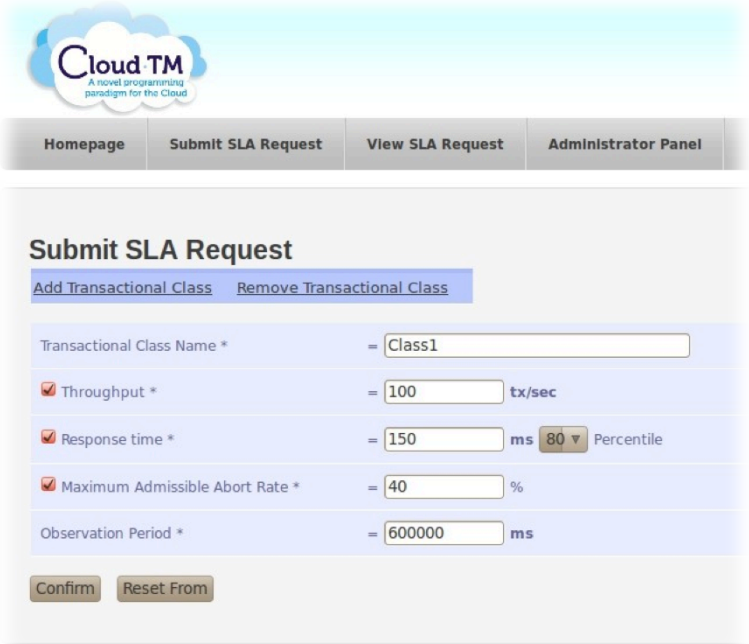
- Multi-objective optimization



# Self-optimization

- QoS/cost driven self-optimization



# The open source way

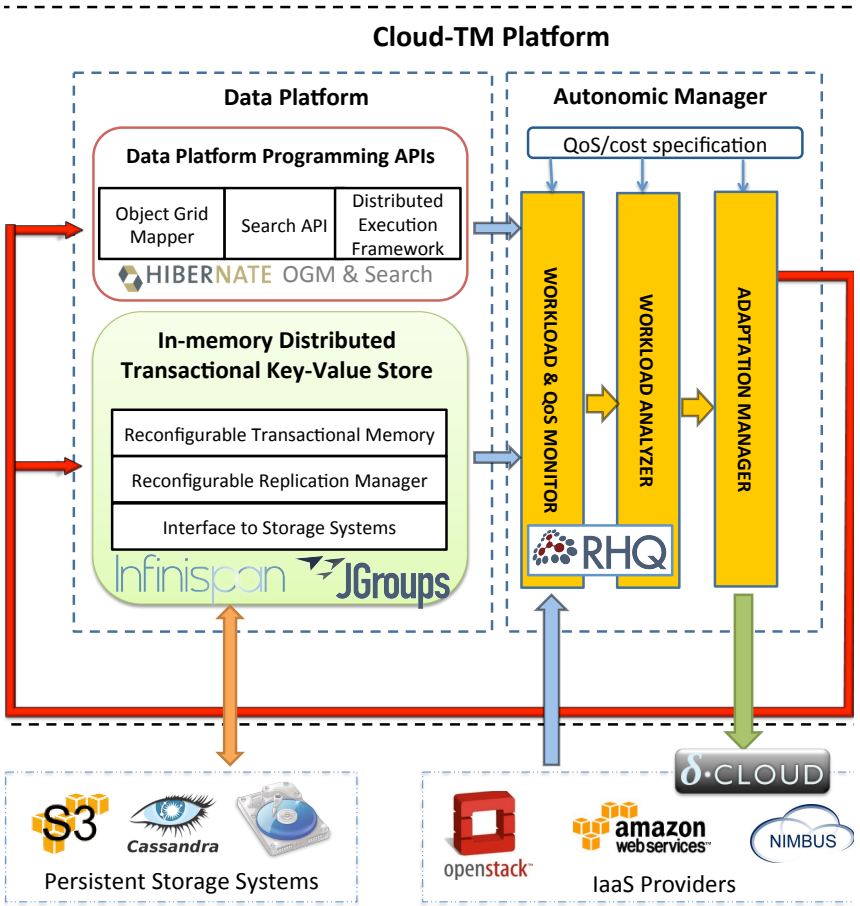
- Research results integrated in highly visible Red Hat projects:

Infinispan

HIBERNATE Search

HIBERNATE OGM

RHQ JGroups



# Conclusions


- Distributed programs are hard to develop
  - even harder in dynamic, large-scale clouds
- API should hide complexity, whenever possible
  - not limit expressiveness, whenever needed
- No-one-size-fits-all solution
  - elastic computing urges for self-tuning solutions

# The Cloud-TM Approach


- Strong transactional consistency
- Object-Oriented programming & query support
- QoS-oriented resource provisioning
- Scalability & efficiency achieved thanks to:
  - innovative data consistency schemes
  - abstractions to maximize locality & reduce conflicts
  - pervasive, multi-objective self-optimization


# Do not miss the Cloud-TM website:

## http://www.cloudtm.eu

## You will find:
## ready-to-go VM images, source code, demos, tutorials, docs...

# References


[DSN13] M. Couceiro, P. Ruivo, Paolo Romano, L. Rodrigues, Chasing the Optimum in Replicated In-memory Transactional Platforms via Protocol Adaptation, The 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2013)

[ICAC12] D. Didona, Paolo Romano, S. Peluso, F. Quaglia, Transactional Auto Scaler: Elastic Scaling of In-Memory Transactional Data Grids, The 9th International Conference on Autonomic Computing (ICAC 2012), San Jose, CA, USA, 17-21 Sept. 2012

[ICAC13] Joao Paiva, Pedro Ruivo, Paolo Romano and Luis Rodrigues, AutoPlacer: scalable self-tuning data placement in distributed key-value stores, The 10th International Conference on Autonomic Computing (ICAC 2013), San Jose, CA, USA, 26-28 June 2013

[ICDCS12] Sebastiano Peluso, Pedro Ruivo, Paolo Romano, Francesco Quaglia, and Luis Rodrigues, When Scalability Meets Consistency: Genuine Multiversion Update Serializable Partial Data Replication, 32nd International Conference on Distributed Computing Systems (ICDCS 2012)

[Middleware11] M. Couceiro, Paolo Romano and L. Rodrigues, PolyCert: Polymorphic Self-Optimizing Replication for In-Memory Transactional Grids, ACM/IFIP/USENIX 12th International Middleware Conference (Middleware 2011)

[Middleware12] S. Peluso, Paolo Romano, F. Quaglia, SCORe: a Scalable One-Copy Serializable Partial Replication Protocol, ACM/IFIP/USENIX 13th International Middleware Conference (Middleware 2012)

[PRDC 11] P. Ruivo, M. Couceiro, Paolo Romano and L. Rodrigues, Exploiting Total Order Multicast in Weakly Consistent Transactional Caches, Proc. IEEE 17th Pacific Rim International Symposium on Dependable Computing (PRDC'11), Pasadena, California, Dec. 2011

[SRDS13] Nuno Diegues and Paolo Romano, Bumper: Sheltering Transactions from Conflicts, The 32th IEEE Symposium on Reliable Distributed Systems (SRDS 2013), Braga, Portugal, Oct. 2013

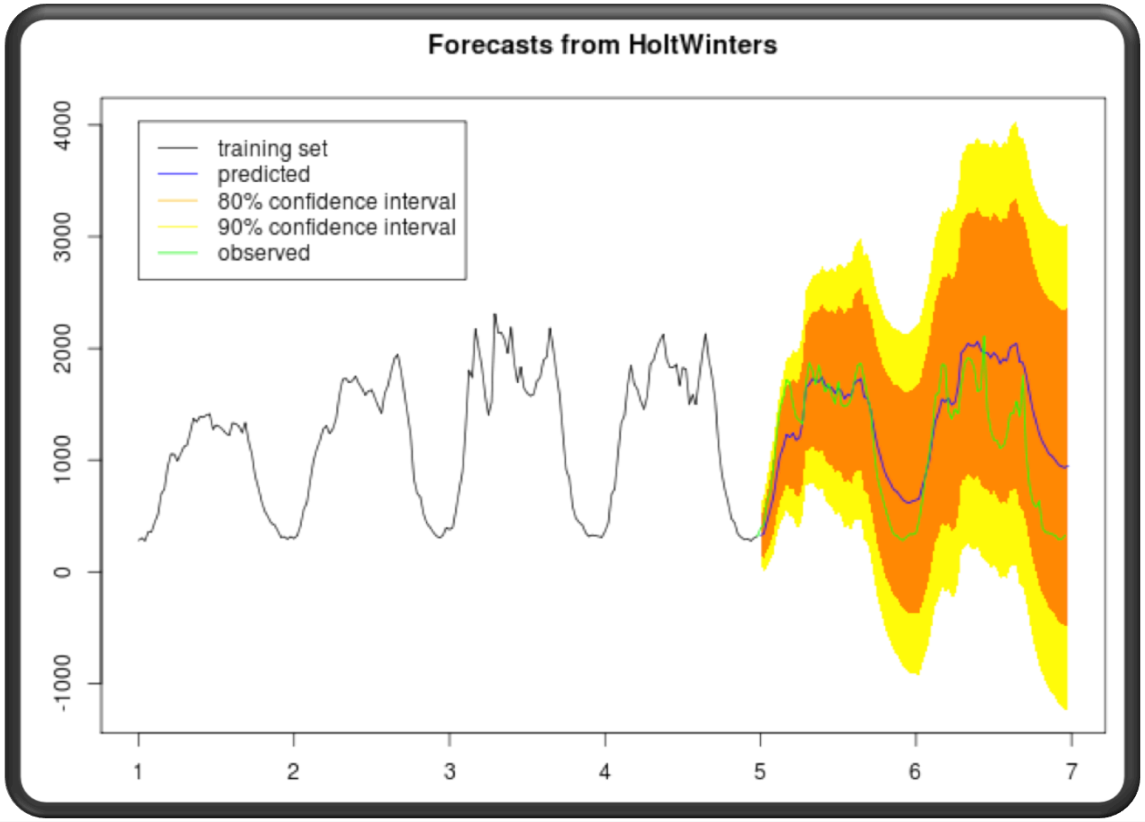

Full list of relevant publications available here:

http://www.cloudtm.eu/home/Publications


# BACKUP SLIDES

# Self-optimization

- Pervasive workload and performance monitoring
  - state of the art workload forecasting algorithms



# Self-optimization

- Pervasive workload and performance monitoring
  - novel metrics to characterize application scalability

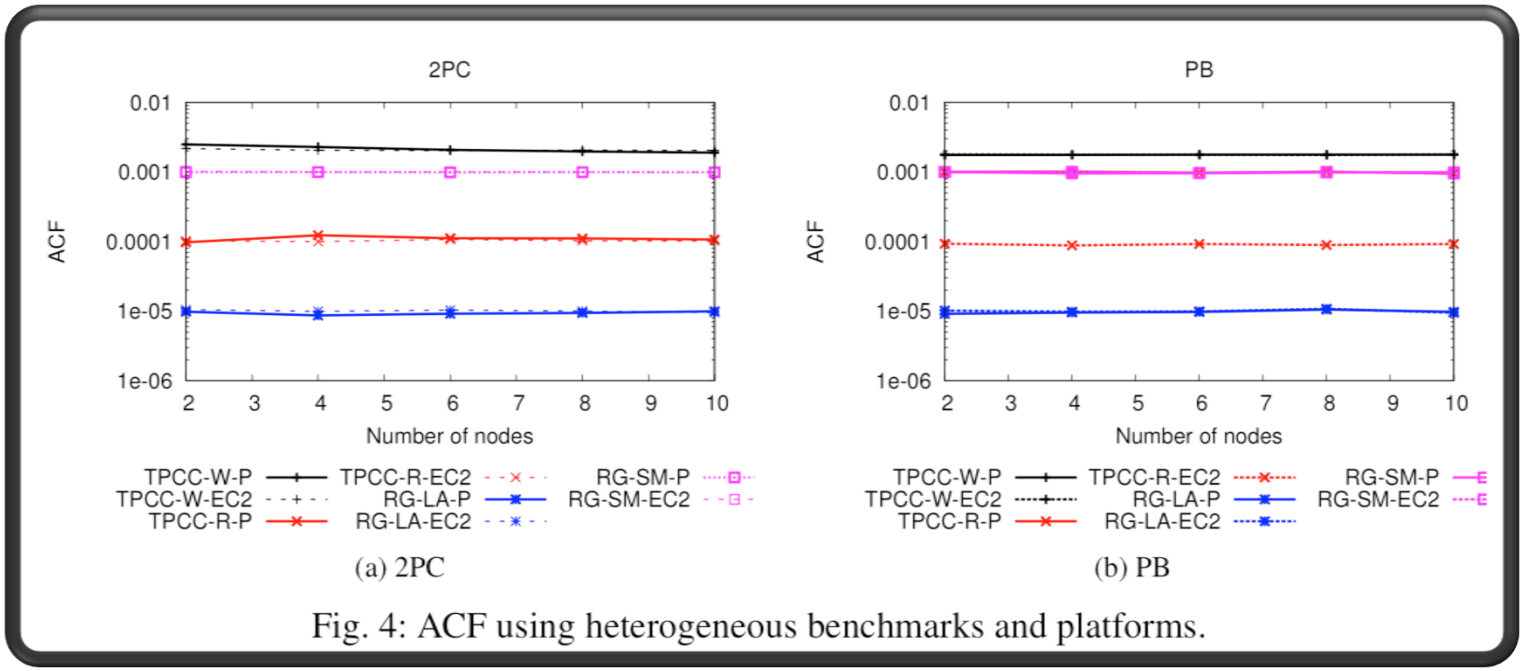
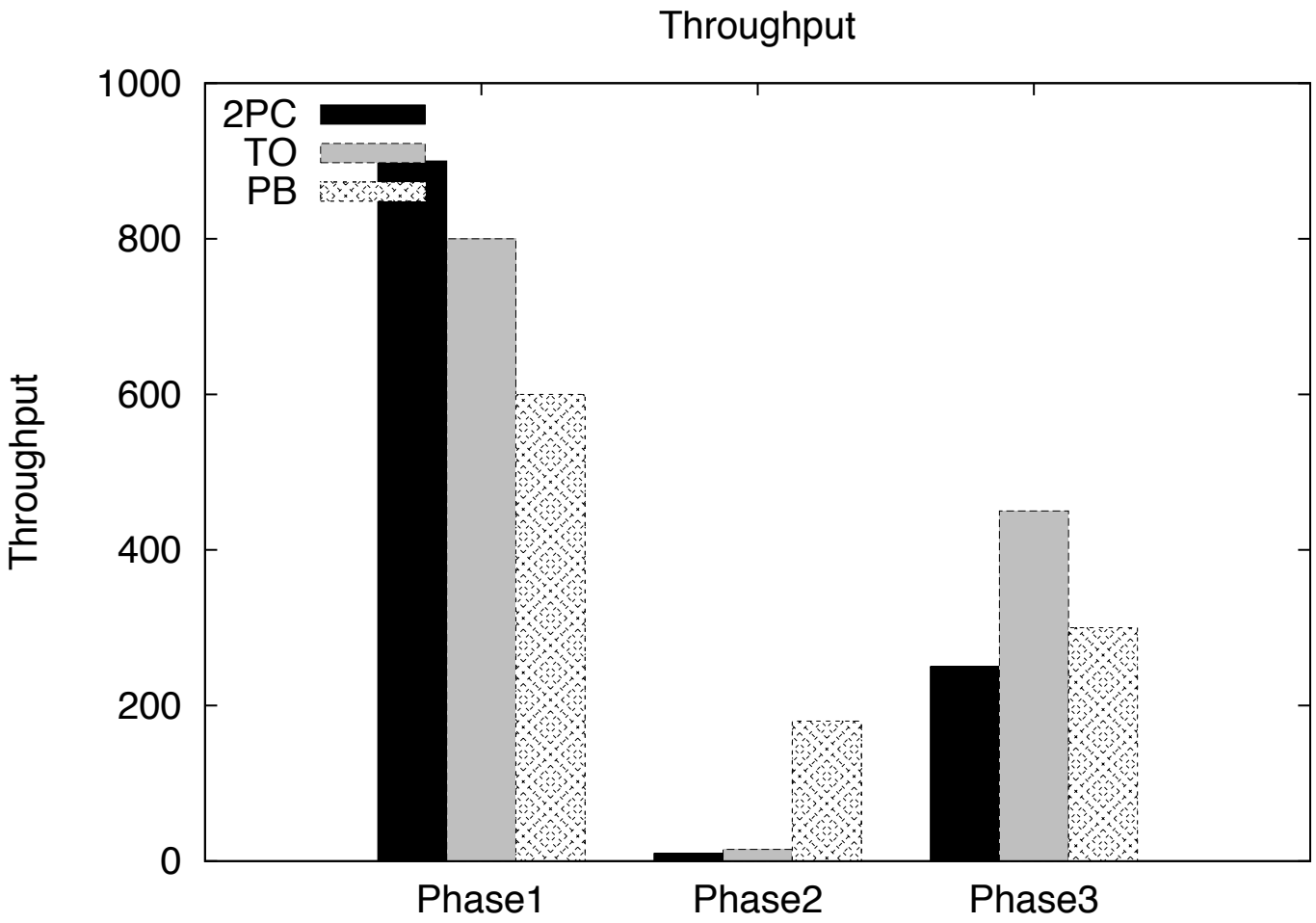


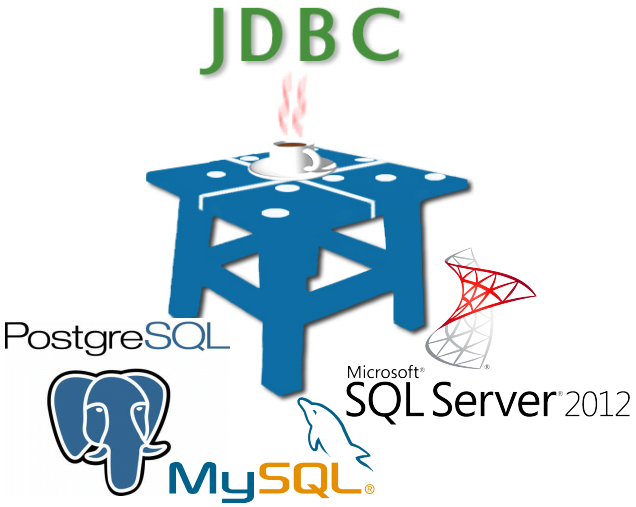

Fig. 4: ACF using heterogeneous benchmarks and platforms.

# Geograph – static configuration



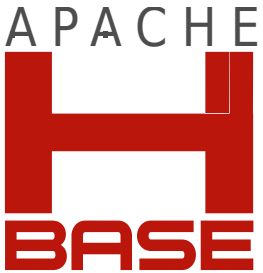
# Data distribution, replication, persistence


- Interoperability with diverse persistent storages







# The problem of data locality


- In partially replicated system, transactions may have to fetch remote data duri
- If this h
  - longer and more
  - conflicts
  - **…shortly: performance will be poor**

**Poor Data Placement Can Cripple Performance!**

# AutoPlacer


- Self-tuning system for adapting data placement in a distributed key/value store

- Key challenges:
  - which data should be moved?
    - big data ➔ large monitoring overheads
  - where to move the data
    - distributed optimization problem
  - how to encode and maintain efficiently the mapping?
    - big data ➔ large directory Key → Node(s)

# Hot Spots Detection


- Online algorithm, round based:
  - fully distributed hot-spot detection
    - most frequently accessed remote keys (per node)
  - lightweight probabilistic top-k algorithm
    - sub-linear space (stream analysis)
    - bounded error
  - in each round, nodes send access frequencies of their hot spots to their node owners

# AutoPlacer


- Self-tuning system for adapting data placement in a distributed key/value store

- Key challenges:
  - which data should be moved?
    - big data ➔ large monitoring overheads
  - where to move the data
    - distributed optimization problem
  - how to encode and maintain efficiently the mapping?
    - big data ➔ large directory Key → Node(s)

# Who gets the data?

- Distributed optimization formulated as an Integer Linear Programming (ILP) problem

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

subject to:

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

Based on approx. stats from probabilistic Top-K

Relaxed to LP problem and computed in parallel as independent subproblems

- Place d copies of each object so to:
  - Minimize global number of remote accesses
  - Ensure storage capacity constraints

# AutoPlacer


- Self-tuning system for adapting data placement in a distributed key/value store

- Key challenges:
  - which data should be moved?
    - big data ➔ large monitoring overheads
  - where to move the data
    - distributed optimization problem
  - how to encode and maintain efficiently the mapping?
    - big data ➔ large directory Key → Node(s)

# How to store where data is?

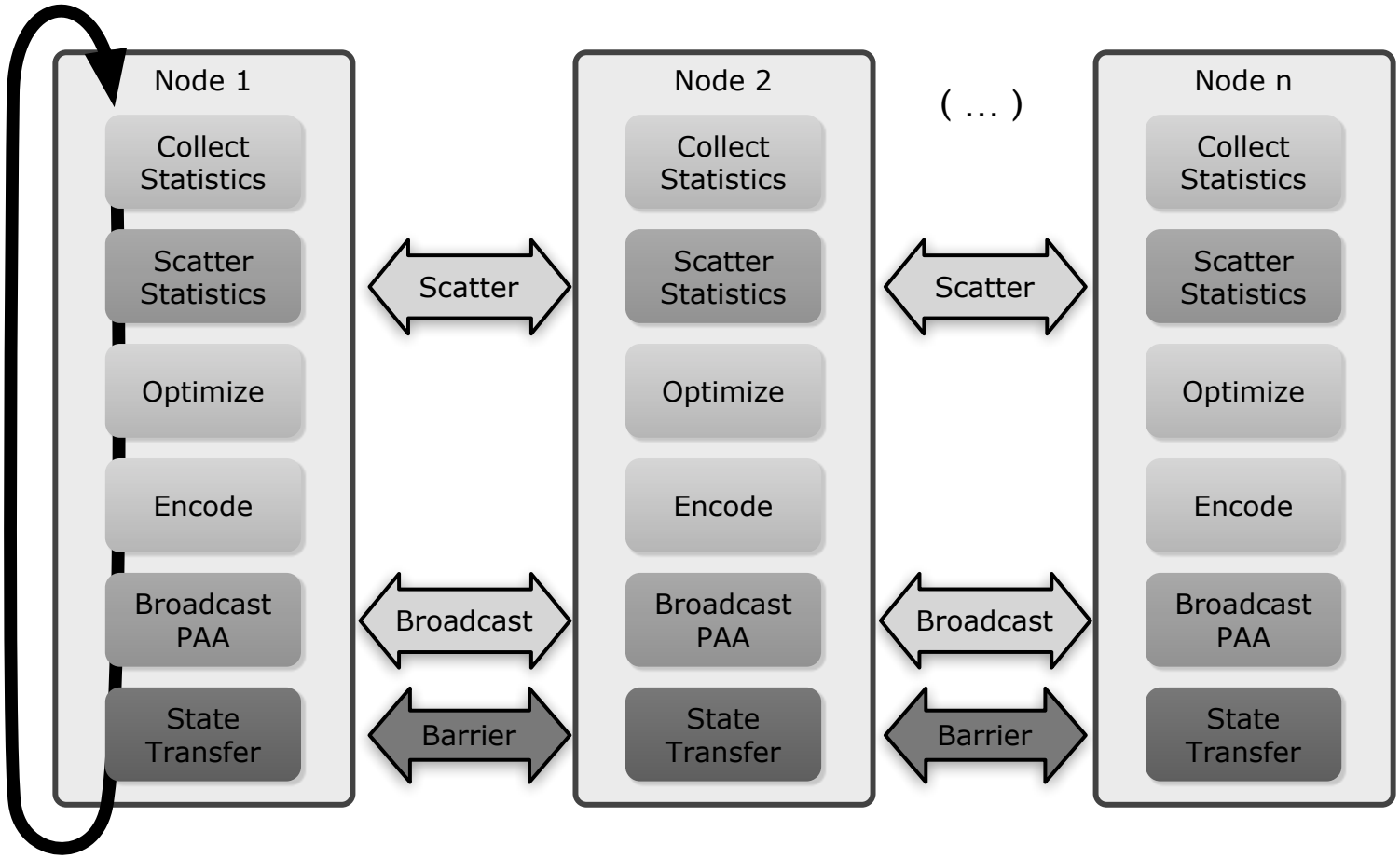
*State of the art solutions*

- directory based:
  - PRO: maximum flexibility
  - CON: non-scalable, large overhead (remote lookup)
- random hashing:
  - PRO: lightweight and scalable
  - CON: no control on data placement

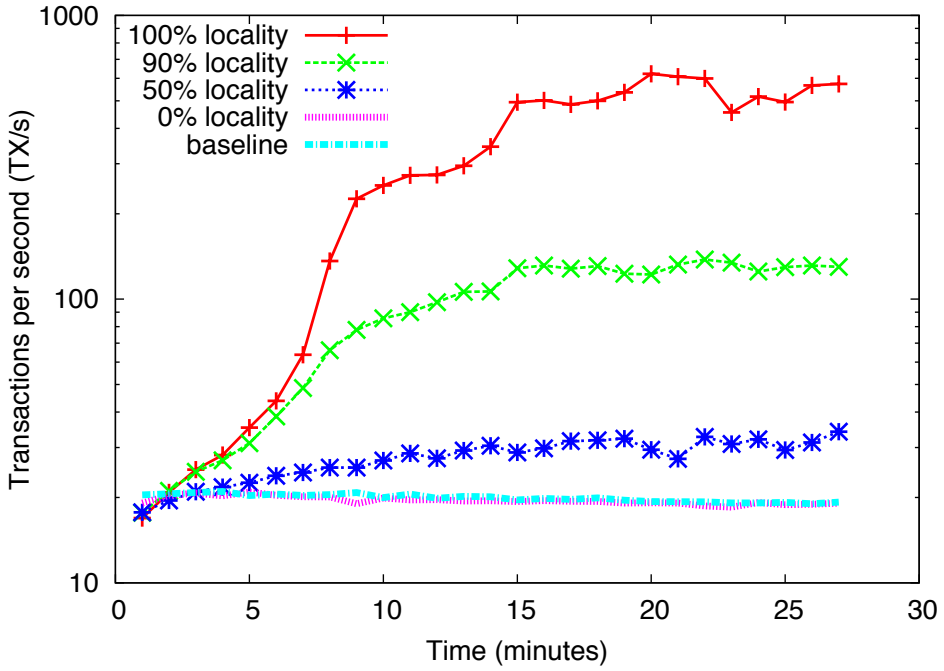
encode <data/node> association via **decision tree classifiers**:

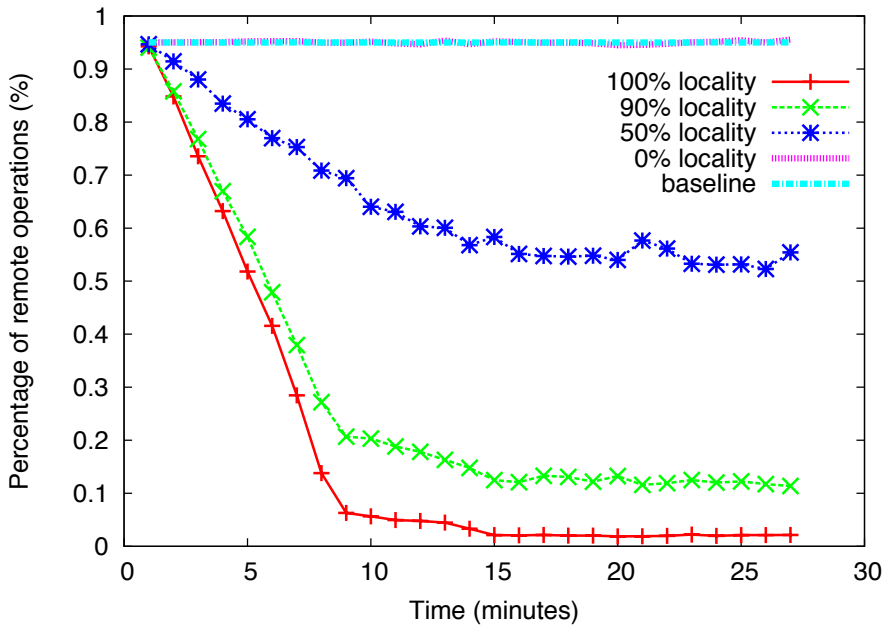
- very compact and highly efficient
- small possibility of misclassifications / misplacement

# Overview

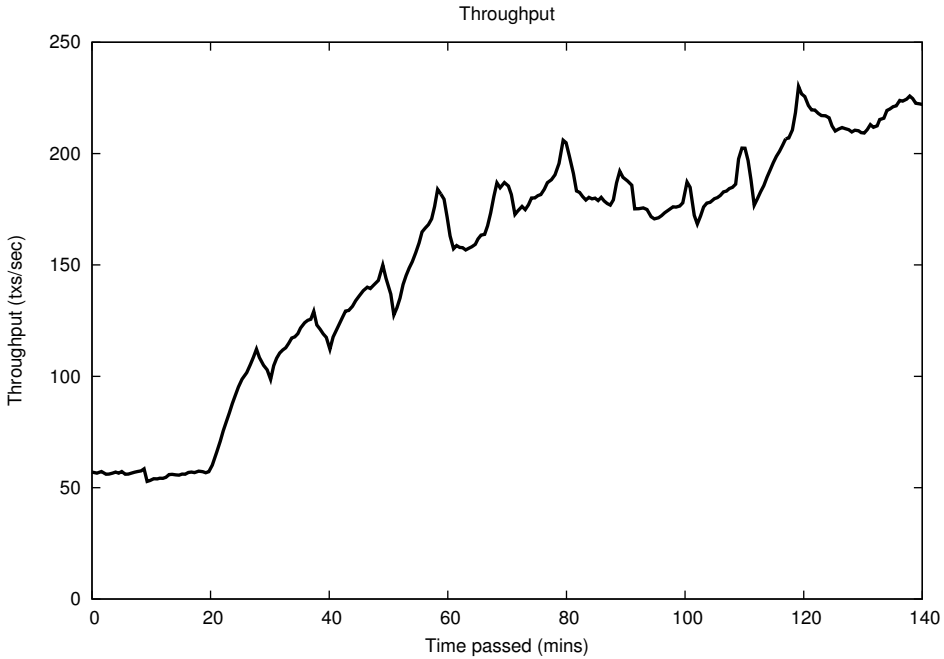

Node 1
Collect Statistics
Scatter Statistics
Optimize
Encode
Broadcast PAA
State Transfer
Scatter
Broadcast
Barrier
Node 2
Collect Statistics
Scatter Statistics
Optimize
Encode
Broadcast PAA
State Transfer
( ... )
Scatter
Broadcast
Barrier
Node n
Collect Statistics
Scatter Statistics
Optimize
Encode
Broadcast PAA
State Transfer

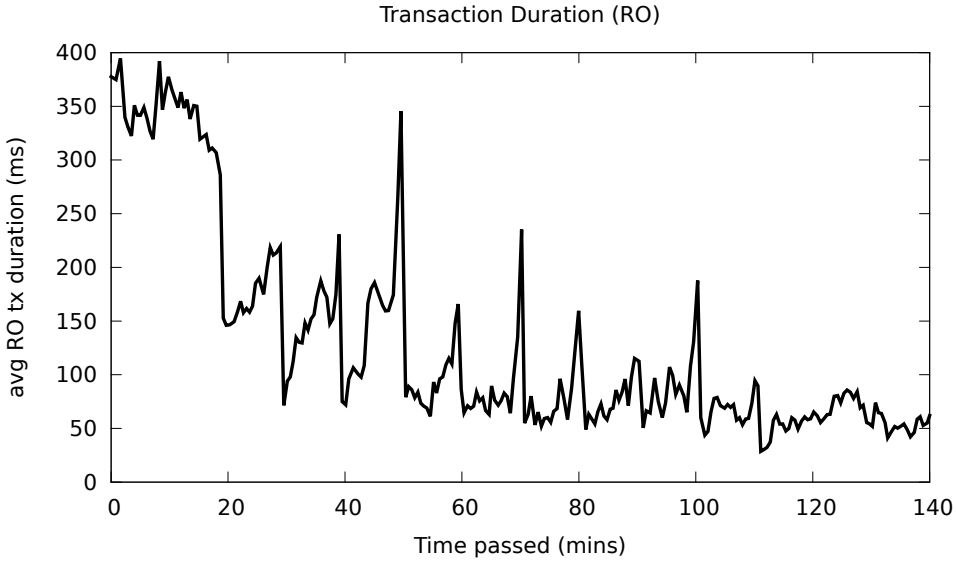
# TPC-C


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


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Percentage of remote operations (%)
100% locality
90% locality
50% locality
0% locality
baseline
0
5
10
15
20
25
30
Time (minutes)

# GeoGraph


Throughput
Throughput (txs/sec)
Time passed (mins)
0
50
100
150
200
250
0
20
40
60
80
100
120
140


Transaction Duration (RO)
avg RO tx duration (ms)
Time passed (mins)
0
50
100
150
200
250
300
350
400
0
20
40
60
80
100
120
140