Relational Cloud: the case for Databases as a service

Carlo Curino, Evan Jones,
Yang Zhang, Eugene Wu,
Sam Madden

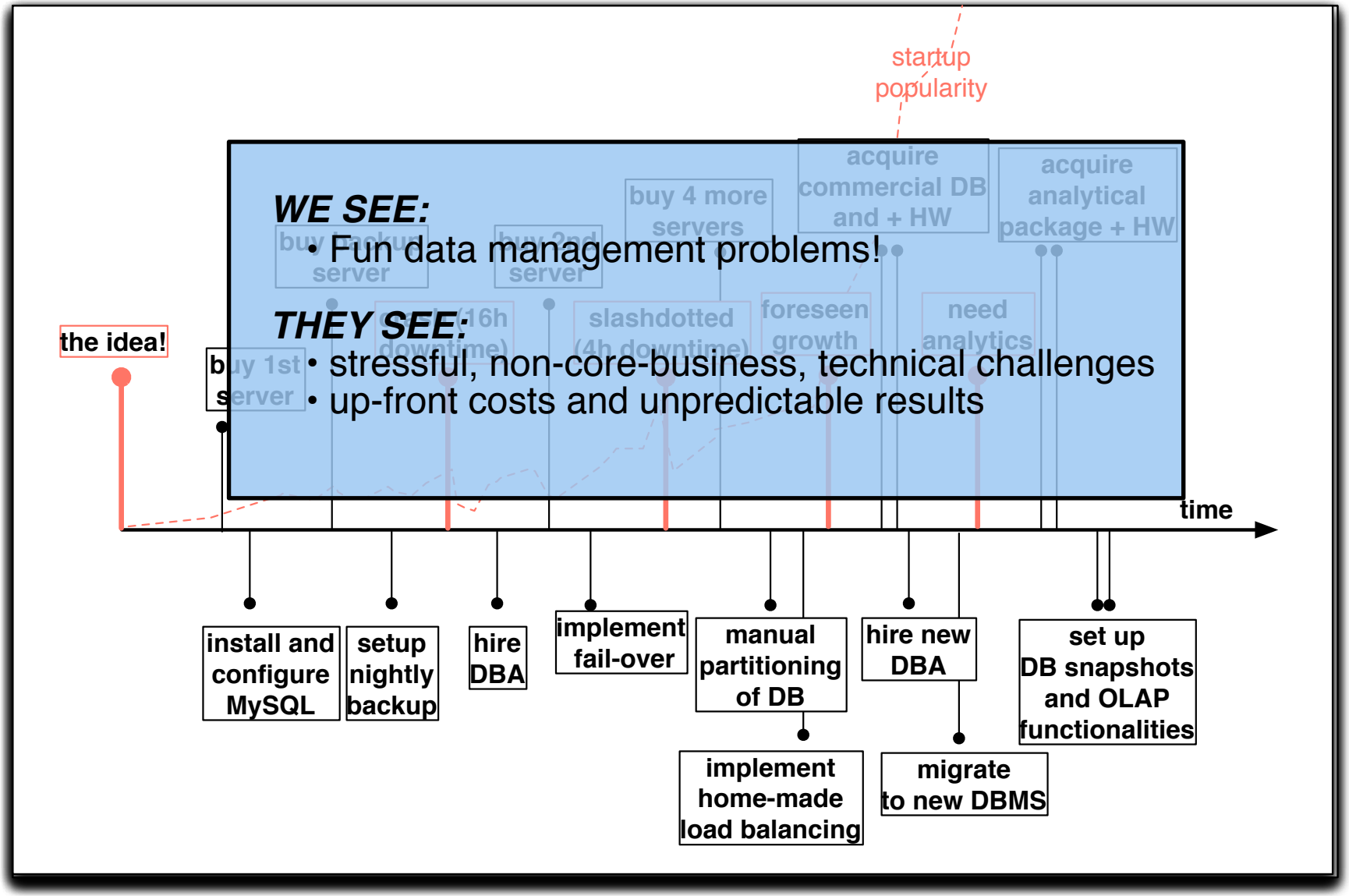
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA

Relational Database Systems

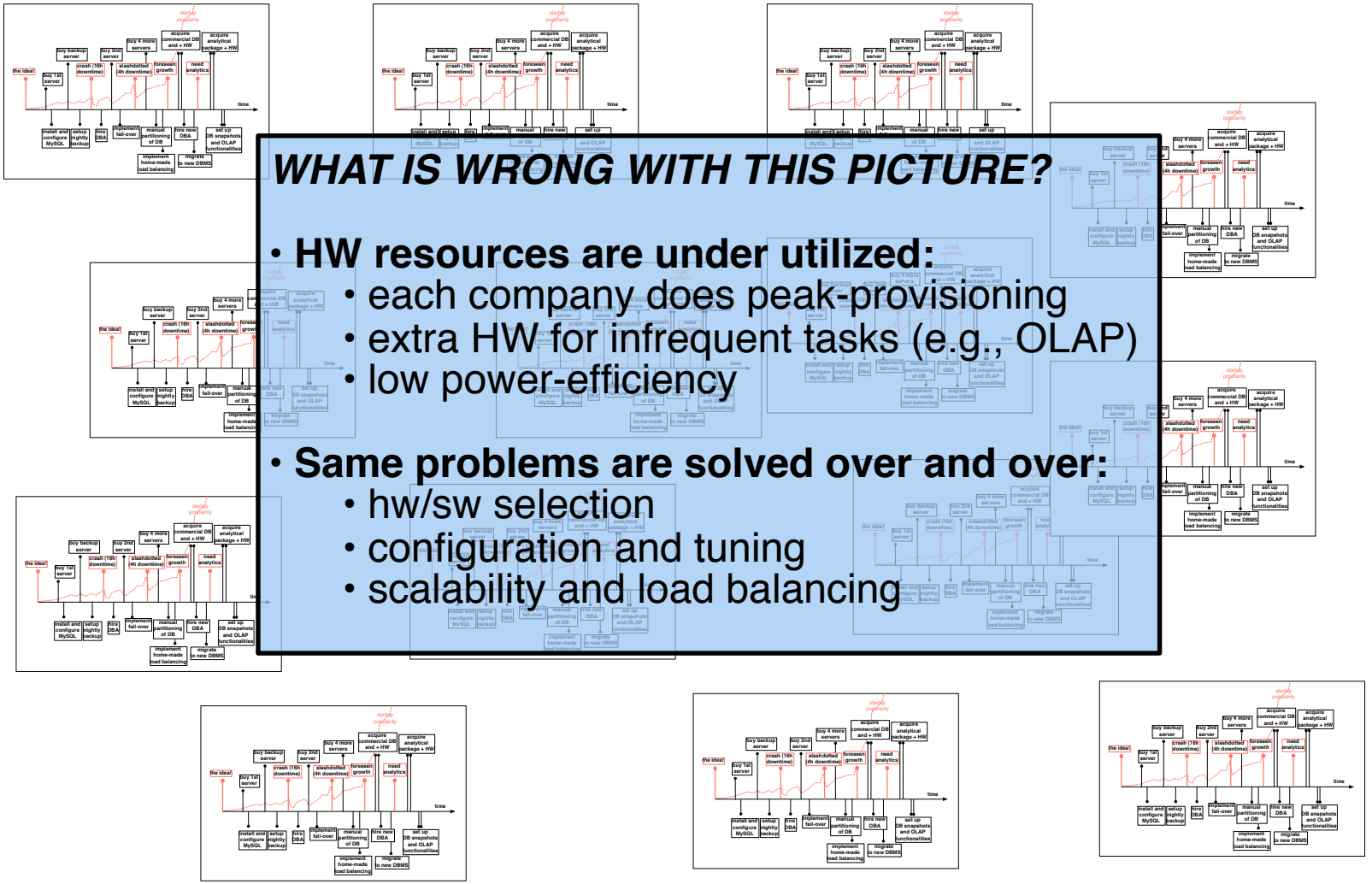
- **Extremely successful data storage and access abstraction**
  - Elegant formal framework
  - Market of over 20B$/year

We love databases, but…

Startup success: a story of DB drama



Startup success: a story of DB drama



WHAT IS WRONG WITH THIS PICTURE?

- **HW resources are under utilized:**
  - each company does peak-provisioning
  - extra HW for infrequent tasks (e.g., OLAP)
  - low power-efficiency
- **Same problems are solved over and over:**
  - hw/sw selection
  - configuration and tuning
  - scalability and load balancing

Relational Cloud Vision

- **SOLUTION:**
  - A full-fledged, transactional, relational DB service
  - Hide complexity, exploit resource pooling, increase automation
- **SCENARIOS:**
  - *Private cloud*: consolidating DB resources in large organizations
  - *Public Cloud**: outsourcing of DB functionality to a service provider

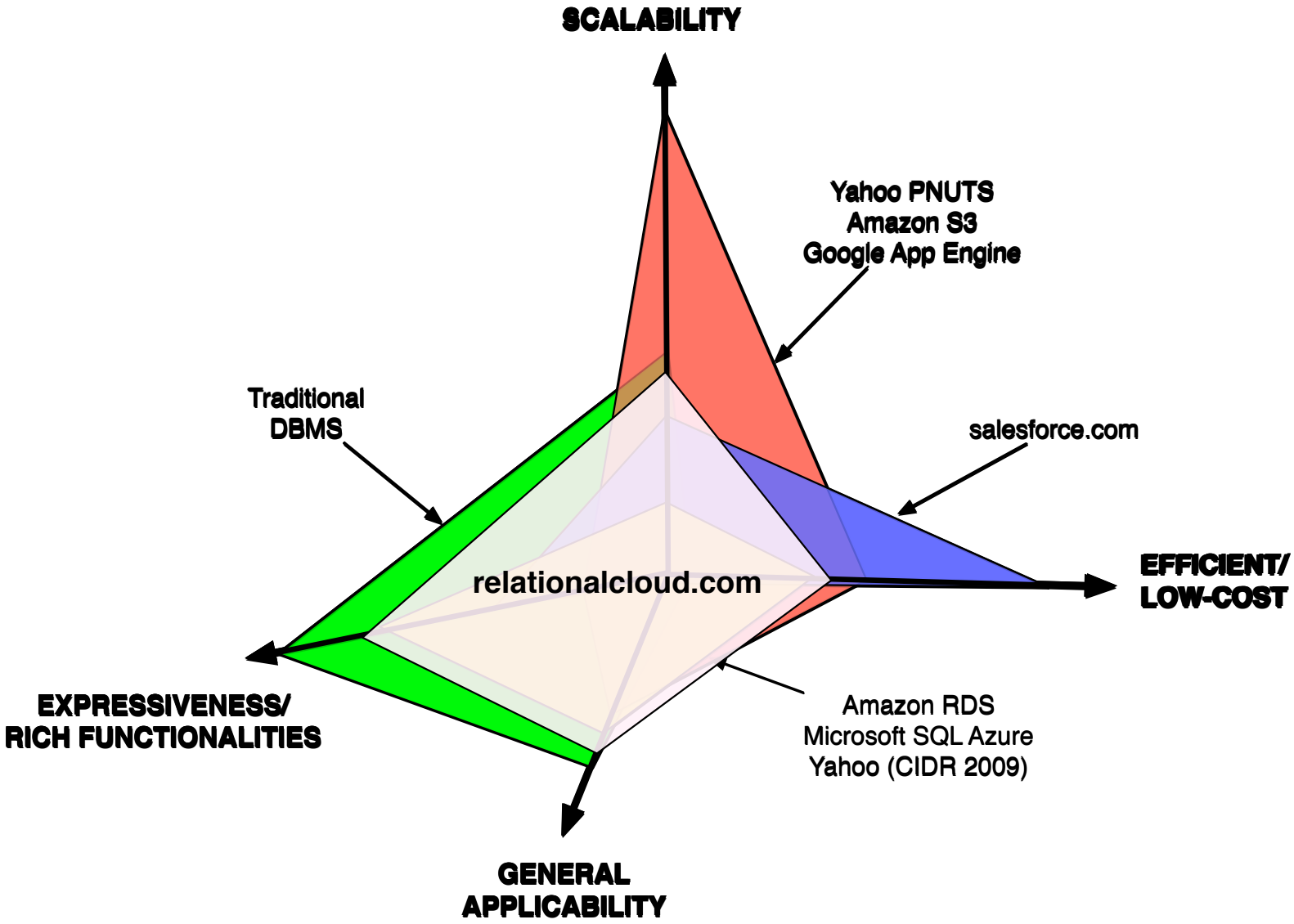
    * Extra requirements (pricing scheme, security/privacy, latency)
- **DISCLAIMER**:
  - Everything I will discuss is a Work in Progress!

Relational Cloud Requirements

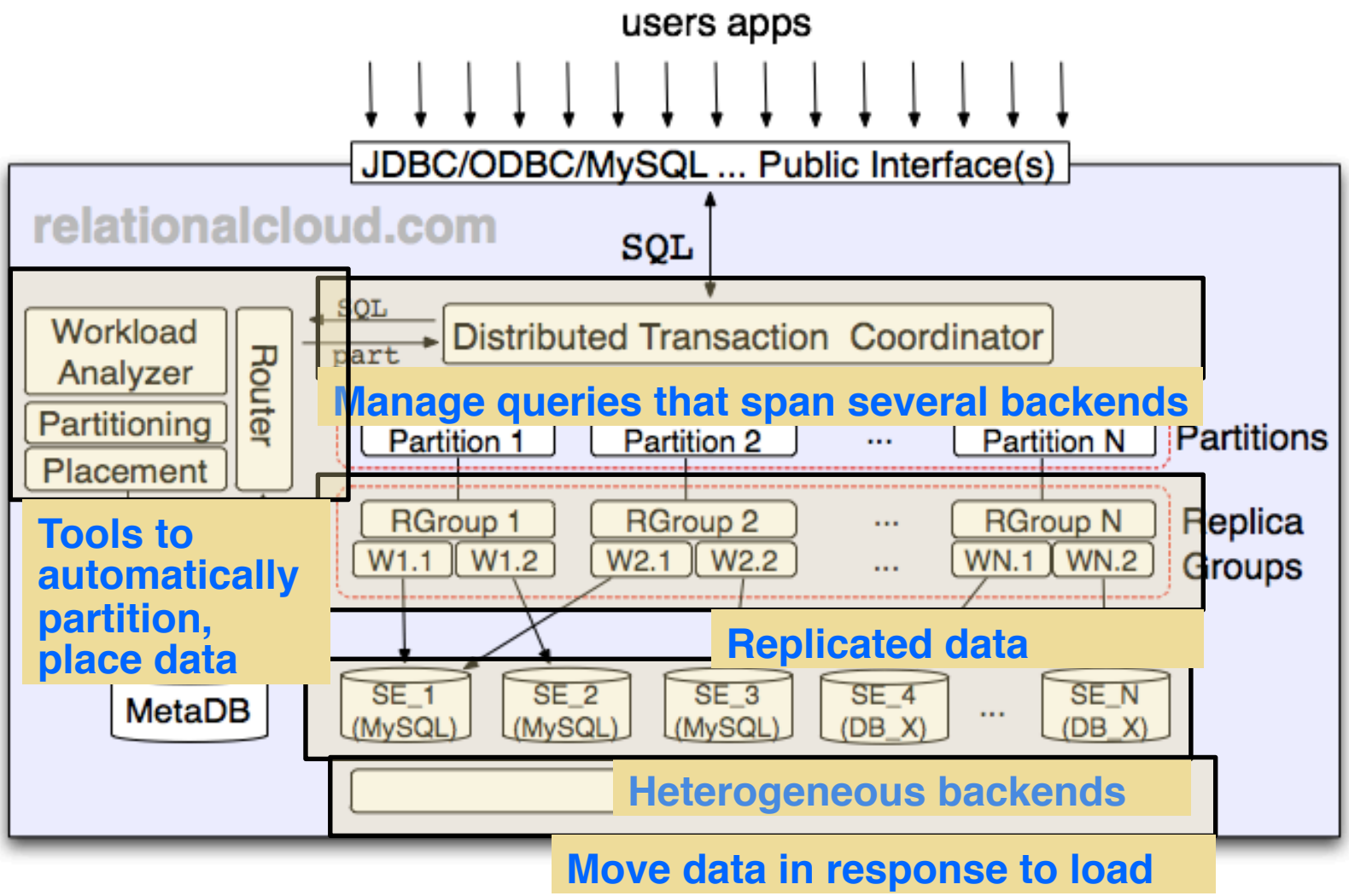
- **User's perspective**
  - Simple SQL API, with near-zero configuration and administration
  - High-performance (e.g., throughput, scalability)
  - High availability and reliability (e.g., hot standby, backup)
  - Ease of access to advanced features (e.g., time-travel, data mining)

- **DB service provider's perspective**
  - Meet user service level agreement (under dynamic workloads)
  - Reduce HW and power costs (e.g., via intense multiplexing)
  - Reduce administration costs (i.e., higher automation)

Current Storage Solutions



Relational Cloud Architecture



Key Features

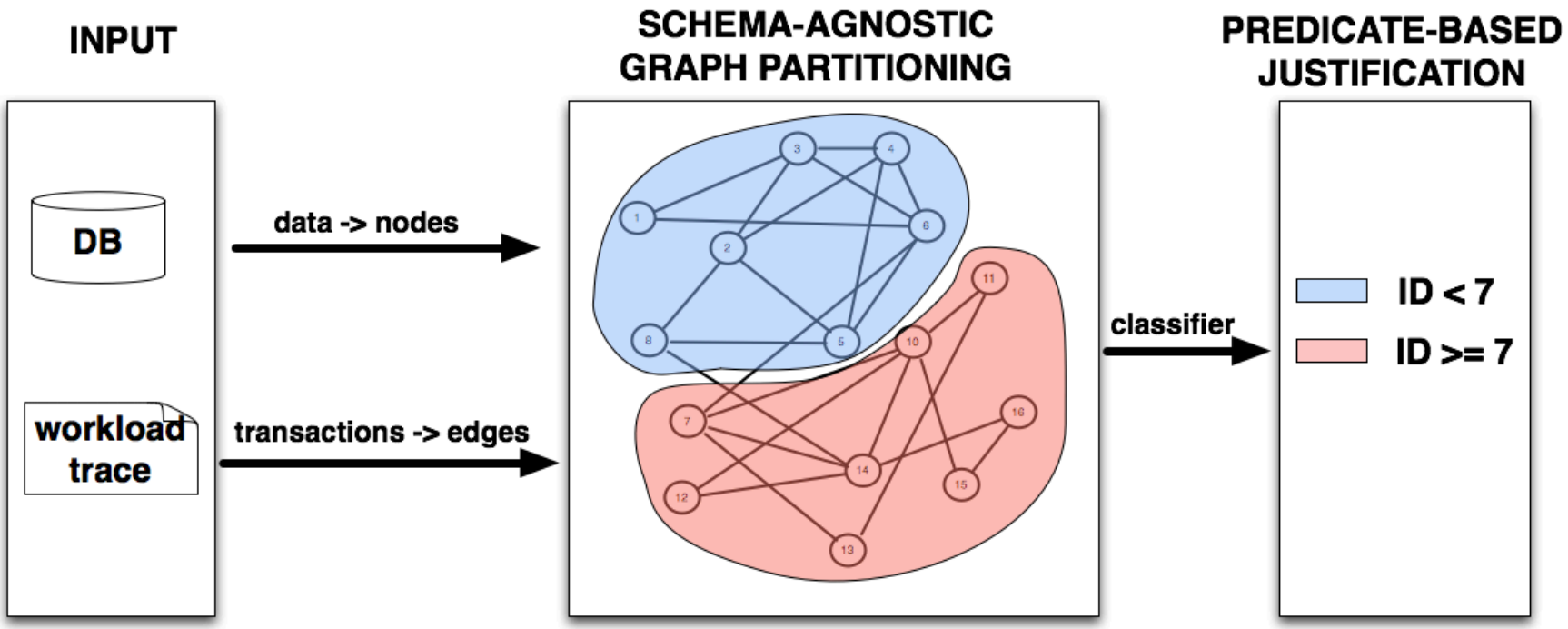
- **Multiple Storage Engines**
  - Specialized engines achieve great performance on specific workloads
  - e.g., column-store for OLAP, main-memory for OLTP
- **Intelligent Workload Analysis and Resource Allocation**
  - Choose best engine for each workload
  - Optimally allocate workloads to physical resources
  - Co-locate engines that don't interfere on same machine
- **Automatic Partitioning**
  - Split databases into partitions to maximize performance
- **Live Migration**
  - Automatically move partitions in response to load

Automatic Partitioning (for OLTP)

- **Why Partitioning:**
  - Scale beyond a single node
  - Finer grained migration/replication/allocation

- **Problem:**
  - *Partition the database into N chunks that maximize performance of the given workload*
  - e.g., minimize distributed transactions for OLTP, maximize intra-query parallelism for OLAP
  - We focus (for now) on OLTP/Web workloads

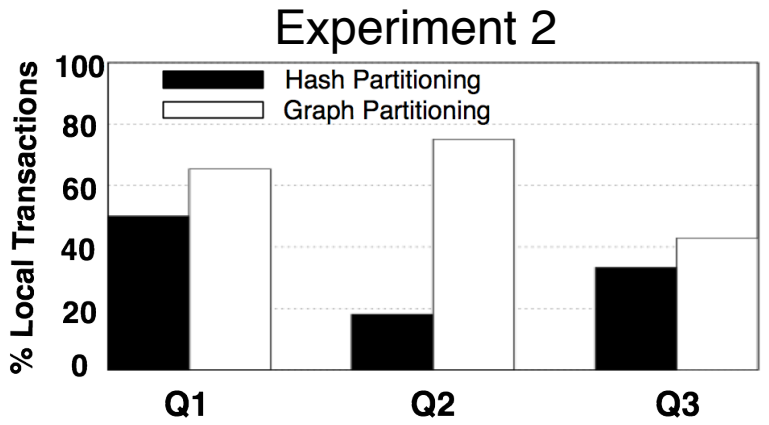
Partitioning: our approach



- **Schema-Agnostic phase:**
  - build a graph representing the DB and the workload (nodes and edges)
  - Use graph partitioning algorithms to find balanced partition
- **Justification phase:**
  - select common attributes from workload
  - build decision tree classifier to "justify" the partitioning

Partitioning Implementation/Performance

- **Main challenge:**
  - Scalability of the graph representation
- **Solution:**
  - Scalable graph-partitioning algorithms
  - Several heuristics to contain graph size



- **Experiment 1 (easy):**
  - TPC-C workload (10 warehouse scale)
  - Partitioning/replication is indistinguishable from manual selection

- **Experiment 2 (hard):**
  - *epinions.com* social-network style schema (i.e., multiple n-to-n relationships)
  - system captures intrinsic correlation between data items
  - 28-314% better locality than hash-partitioning

Status

- **Current Prototype**
  - Many pieces of the infrastructure
    * Distributed Transaction Coordinator (very low-overhead)
    * Query routing component
    * Replication mechanisms
    * Wired-in MySQL and HSQLDB backends
  - Automatic Partitioning/Replication tools

- **Investigations**
  - Cost of distributed transactions and 2PC
  - Unusual storage engines allocations (main-memory and disk-based)
  - Cache-based approaches to Live Migration

Conclusion

- **Databases as a service offer tremendous benefits**
  - Reduced administration and hardware costs
  - Elasticity and scalability

We are implementing these ideas and we are interested in your feedback!

http://relationalcloud.com