# RITA: An Index-Tuning Advisor for Replicated Databases

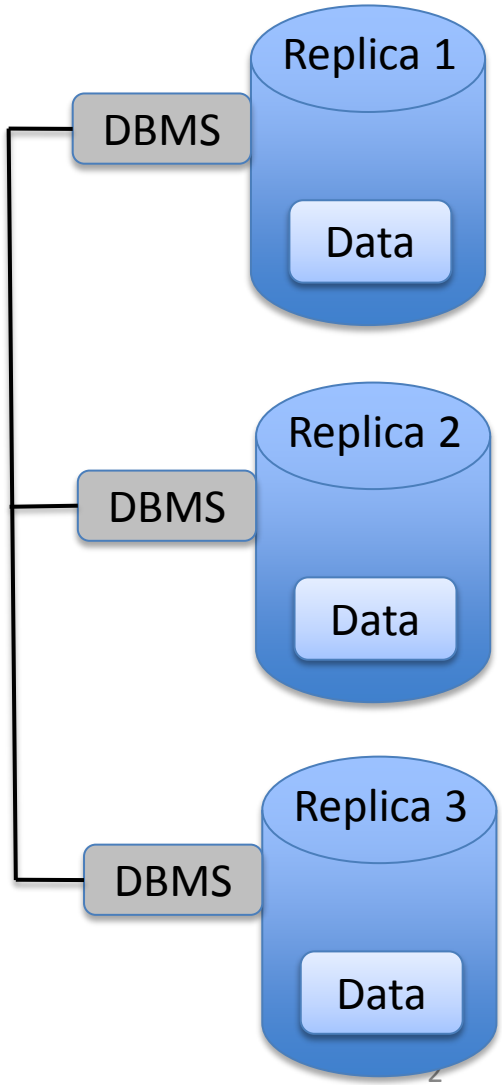
Quoc Trung Tran – Oracle USA *
Ivo Jimenez – UCSC
Rui Wang – UCSC
Neoklis Polyzotis – Google Inc. *
Anastasia Ailamaki – EPFL

* Work done while authors were affiliated with UC Santa Cruz

SSDBM 2015

# Replicated Databases

- Replication is used to provide
  - Fault tolerance
  - High availability
  - Load balancing
- A query is routed to any replica
- Updates go to all replicas
- Examples: Oracle Cloud, SQL Azure, Amazon RDS



# The Index-Tuning Problem

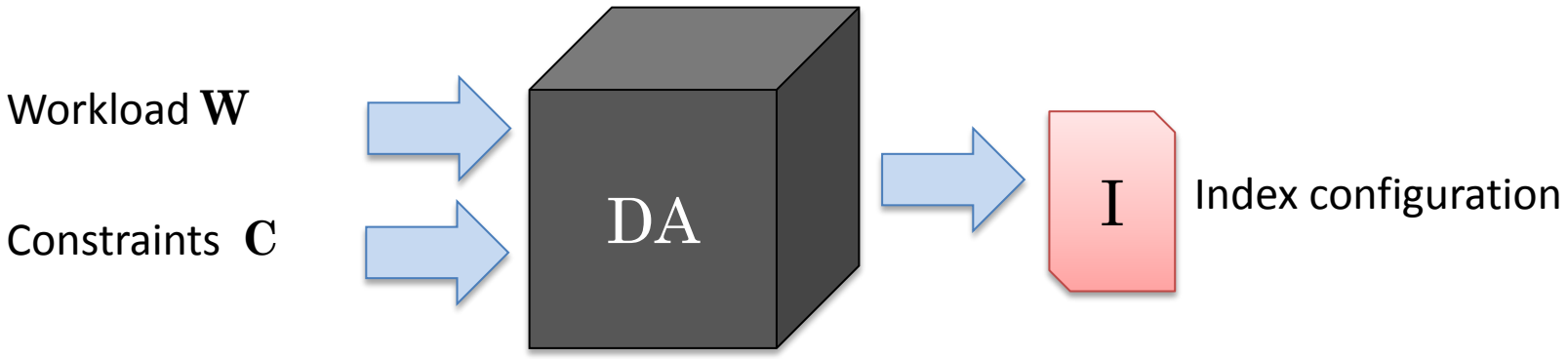
(Informally) What database indexes should be materialized to maximize performance?

```
SELECT  R.y
FROM    R
WHERE   R.x = 10
```

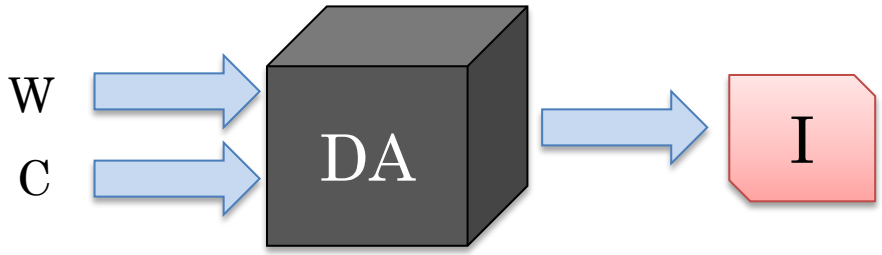
- An index **CREATE INDEX i1 ON R(x)** can help to valuate the where-clause
- An index **CREATE INDEX i2 ON R(x,y)** can help to evaluate the whole query

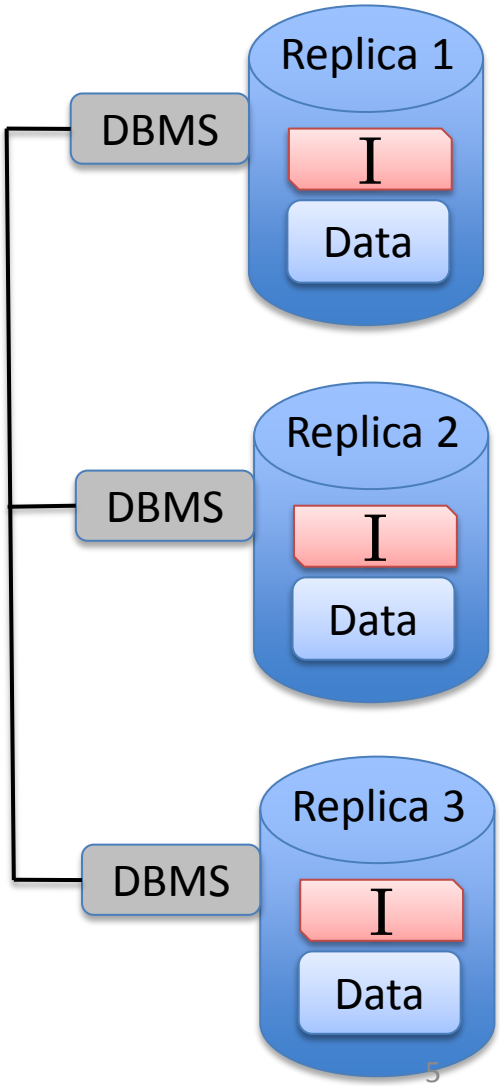
# Index-Tuning Advisor

- Tools available on commercial DB systems
- Workloads W: queries and updates
- Constraints C, e.g., space budget constraint



# Replicated Databases – Uniform Design

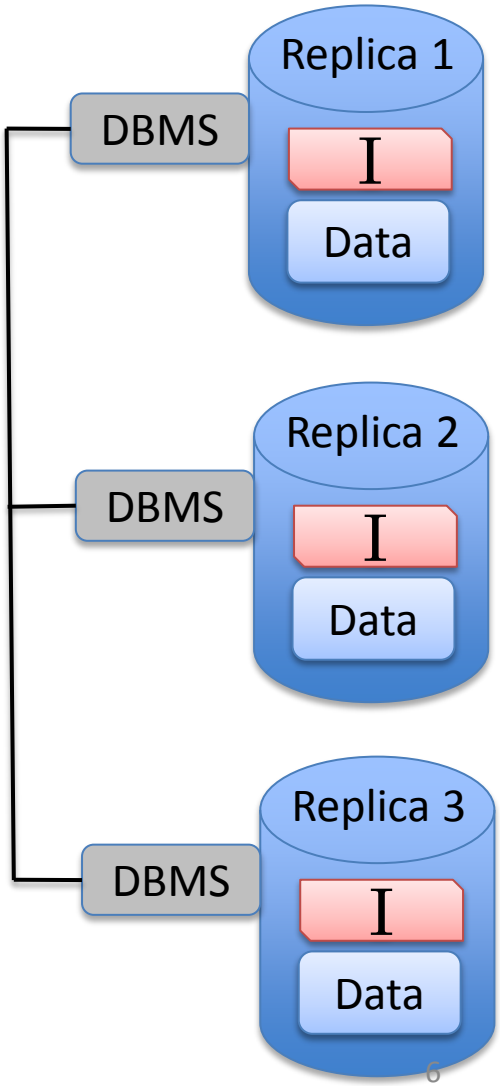


# Uniform Design – How it Works

**Query routing is straightforward:**

- For each query q in Workload W:
  route q to ***any*** replica
- Route update u to ***every*** replica
- Cost of q is the same on all replicas

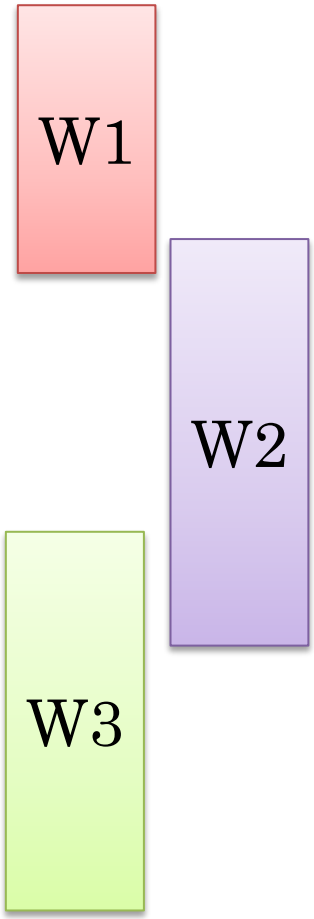


# Divergent Design

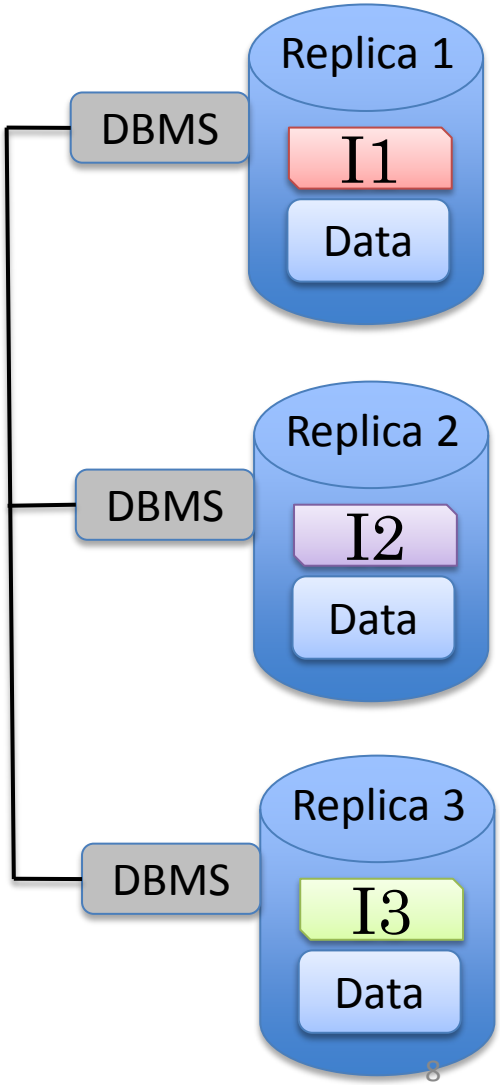
Create different physical designs for each replica

**Reference:** M. Consens, K. Ioannidou, J. LeFevre and N. Polyzotis. *Divergent Physical Design Tuning for Replicated Databases. SIGMOD 2012.*

# Replicated Databases – Divergent Design



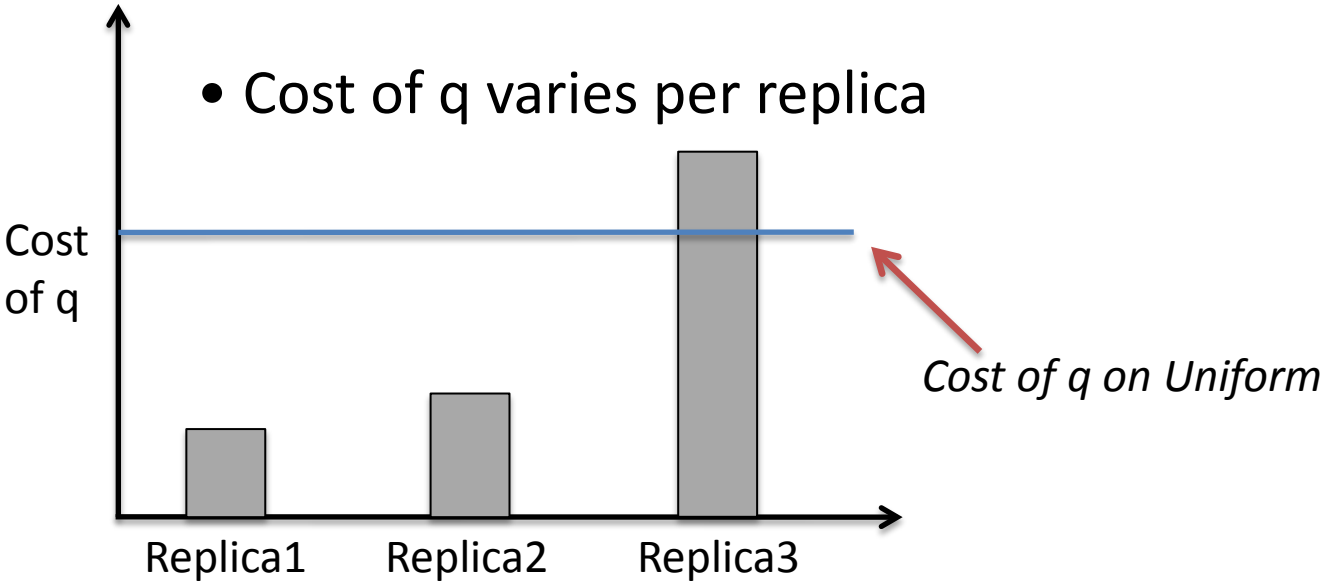

$W = W1 \cup W2 \cup W3$

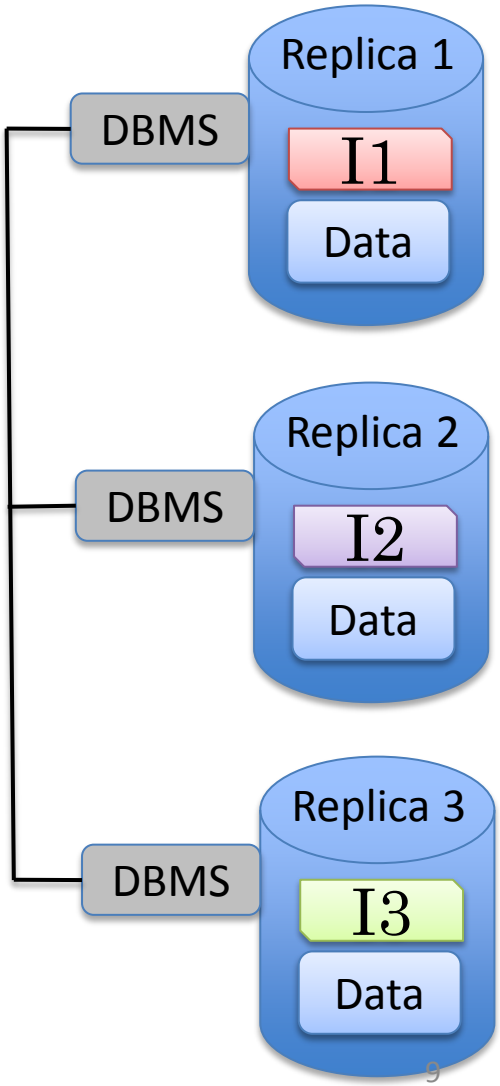


# Divergent Design – How it Works

**Queries are routed to a *subset* of replicas:**

- For each query q in Workload W:
  route q to ***any low-cost*** replica
- Route update u to ***every*** replica
- Cost of q varies per replica



# Divergent Design vs. Uniform Design

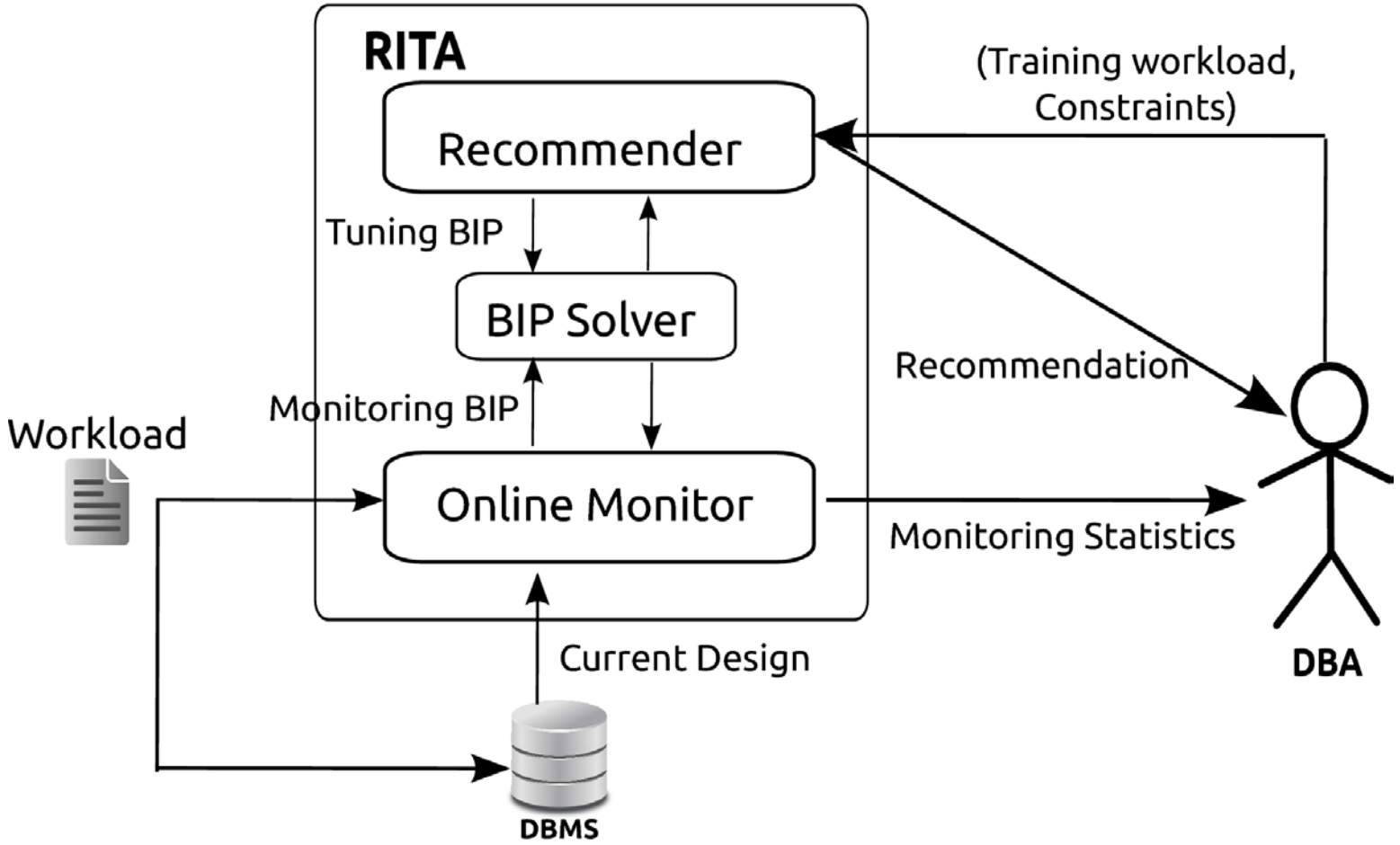
- Divergent design brings **significant perf. Improvements**
  - Queries are executed faster: replica specialization
  - Updates become significantly more efficient: fewer indexes are installed in each replica
  - Uses less storage space for indexes

# Limitation of the Previous Work

- Ignores the possibility of replica failures
- Might cause highly skewed load distribution among replicas
- Targets a static system
  - A fixed number of replicas
  - Known workload distribution

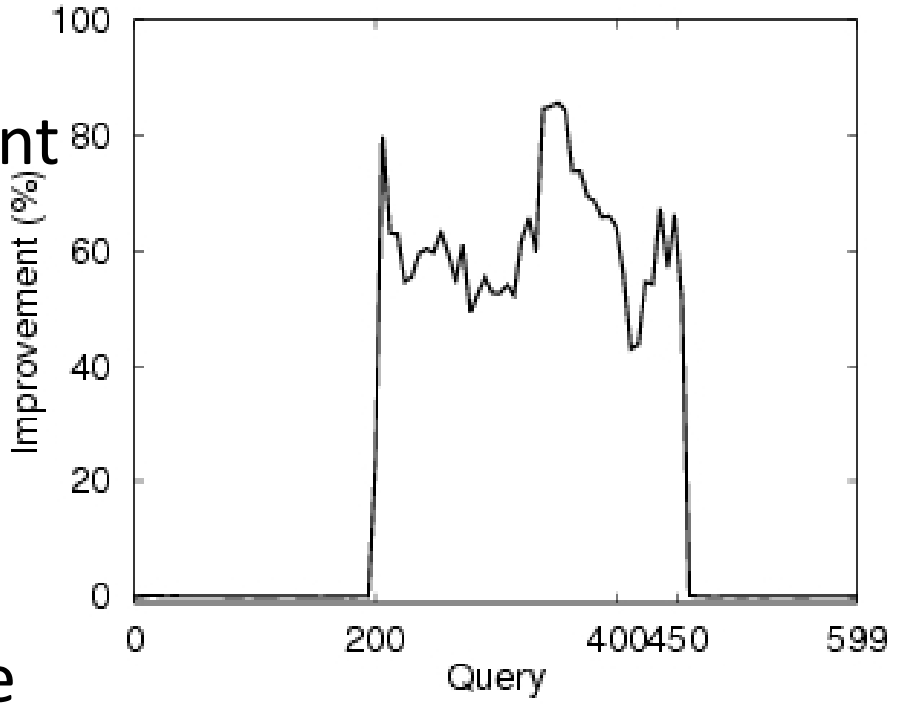
# Our Contributions

RITA: **R**eplication-Aware **I**ndex-**T**uning **A**dvisor



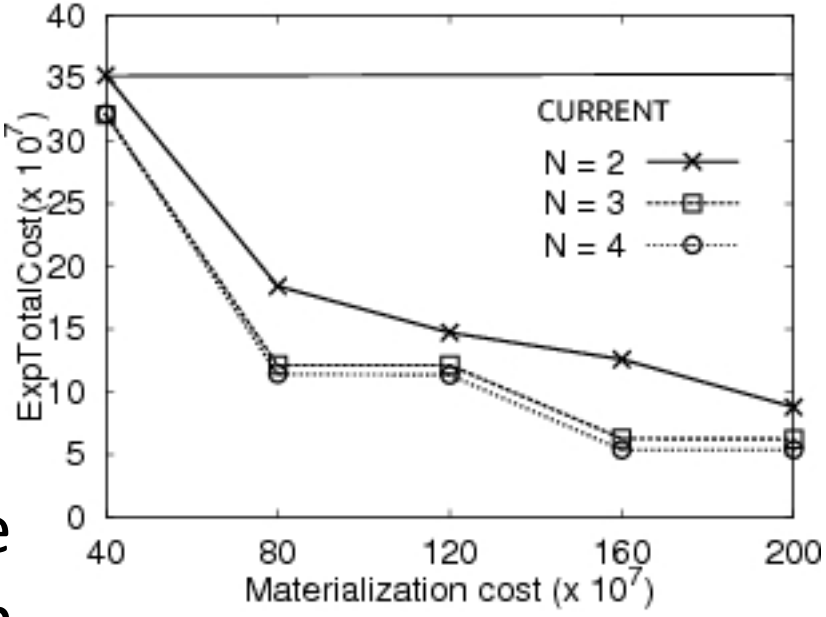
# RITA: Online Monitor

- Continuously analyzes the incoming workload
- Computes an effective divergent design for latest queries
- Compares the up-to-date design to the current design
- RITA takes 6 seconds to update the graph after each query is seen



# RITA: Advisor

- Handles replica failures gracefully
- Balance loads among replicas
- Offer suggestions to elastically reconfigure the system
- RITA takes 40 seconds to update the graph after each query is seen



# Outline of the Talk

- Divergent Design Tuning (DDT)
  - Handle replica failures
  - Balance loads among replicas
- DDT as Binary Integer Programming (BIP)
- Experimental results

# Divergent Design Tuning

Given:

- Workload $W = Q \cup U$
  - Every query q in Q has weight f(q) provided
  - Every update u in U has weight f(u) provided
- Number of replicas N
- A set of constraints C
- Probability of at most one replica fails at same time α

Compute a ***divergence design*** that minimizes **total workload cost**

# Divergent Design

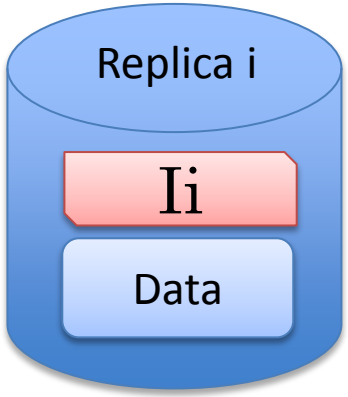
- Divergent design (I, h)

- **Index configurations** $I = \langle I_1, I_2, \ldots I_N \rangle$
  - $I_N$: index configuration of replica N

- **Routing functions** $h = \langle h_0, h_1, \ldots, h_N \rangle$
  - $h_0(q)$: how to route query q when all replicas alive
  - $h_N(q)$: how to route query q when replica N fails

# Cost of a Statement in W

- Let $I_i$ be *installed* on replica I
- Cost of a query q on replica i
  - $Cost(q, I_i)$
- Assuming a query can be routed to **m** replicas




$$Cost(q) = (1-\alpha)\frac{f(q)}{m}\sum_{r\in h_0(q)} Cost(q, I_r) + \frac{\alpha}{N}\frac{f(q)}{\max(m, N-1)}\sum_{j=1..N}\sum_{r\in h_j(q)} Cost(q, I_r)$$

# Divergent Design Tuning (Revisited)

Given:

- Workload $\mathrm{W} = \mathrm{Q} \cup \mathrm{U}$
  - Every query q in Q has weight f(q) provided
  - Every update u in U has weight f(u) provided
- Number of replicas $\mathrm{N}$
- A set of constraints C
- Probability of at most one replica fails at at time α
- **A routing multiplicity m**

Compute a ***(I,h)*** that minimizes $\sum_{q \in W} Cost(q)$

# Constraints

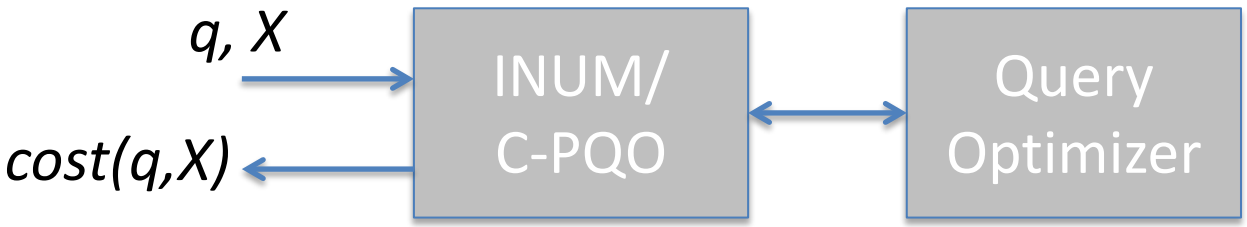
- Inter-replica constraints
  - Improve total cost by some percentage
  - The load skew among replicas is below a threshold
- Intra-replica constraints
  - The size of indexes is within storage budget
  - The cost to update indexes is below a threshold

# Outline of the Talk

- Divergent Design Tuning (DDT)
  - Handle replica failures
  - Balance loads among replicas
- DDT as Binary Integer Programming (BIP)
- Experimental results

# Fast what-if optimization

- Examples: INUM [Papadomanolakis et al. 2007], C-PQO [Nehme and Bruno 2008]




- Input: query q and an index-set X
- Output: cost of evaluating *q* given indexes in *X*
- Much faster than invoking the optimizer

# Our Approach

- **Theorem**: Any instance* of the DDT can be reduced to a *compact* Binary Integer Program

  * Assuming fast what-if optimization

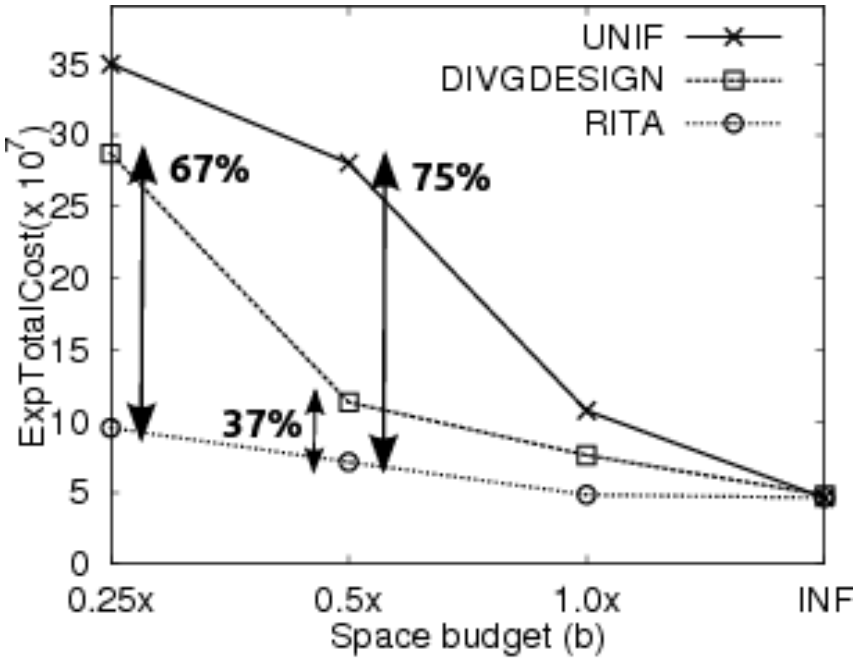
- BIP: Optimize f(**x**) such that g(**x**) ≤ 0
  - All variables in **x** are binary
  - f and g are linear functions

# Our Approach

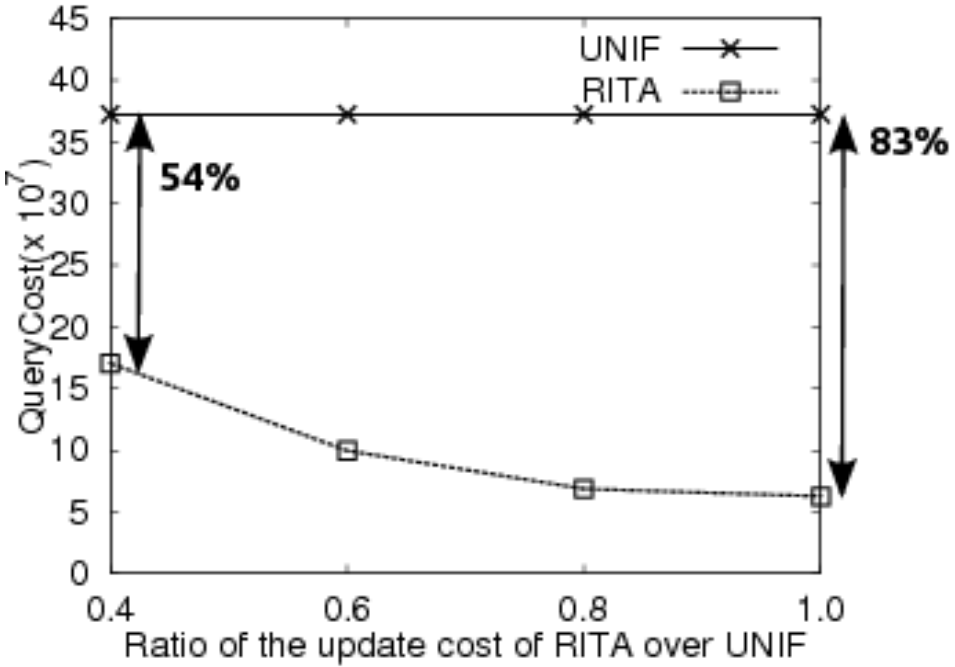
- Solution to BIP ➔ Optimal divergent designs
- We can use mature off-the-shelf BIP solvers to perform index tuning
- What about the theoretical complexity?
  - Solving a BIP is NP-Hard
  - But the average case is much much better!

# Outline of the Talk

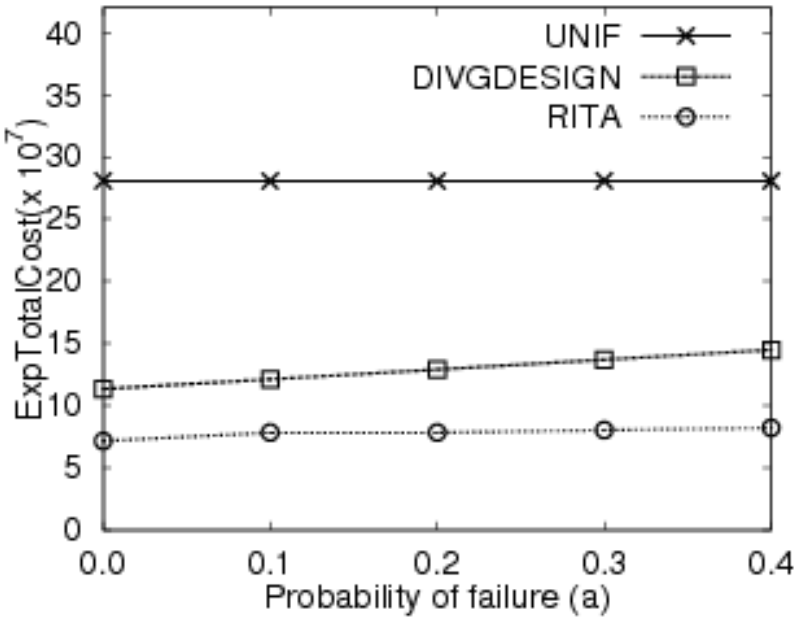
- Divergent Design Tuning (DDT)
  - Handle replica failures
  - Balance loads among replicas
- DDT as Binary Integer Programming (BIP)
- Experimental Results

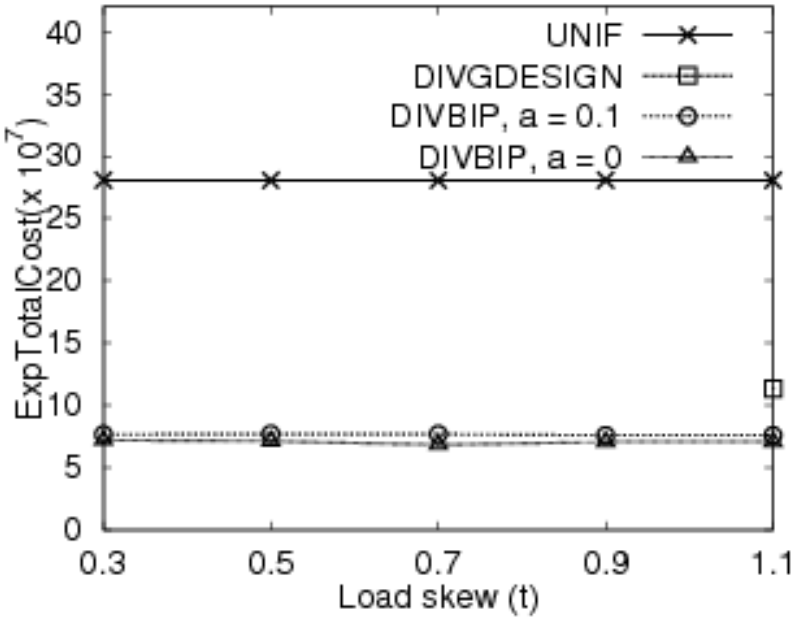



- TCPDS queries only
- Number of replicas (N): 3
- Prob. of failure (alpha): 0

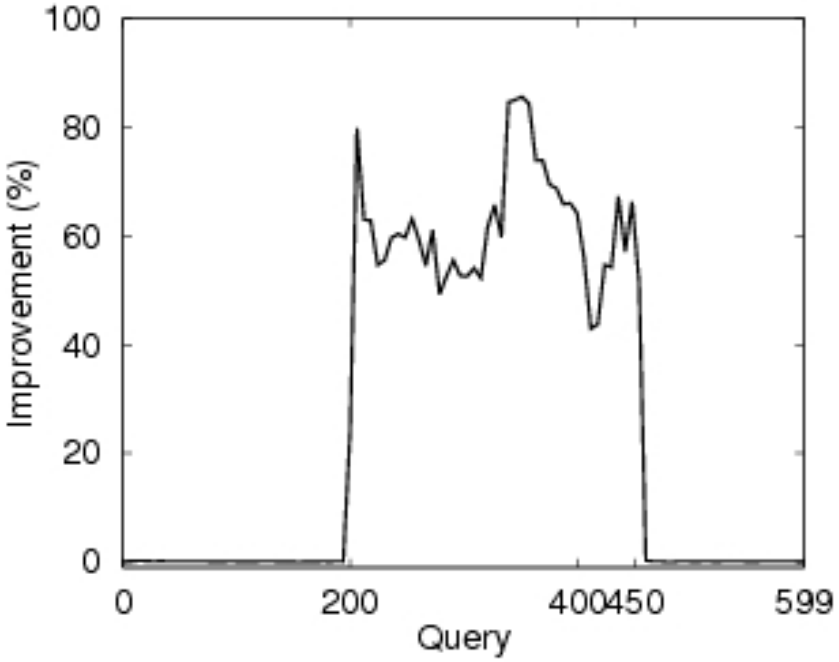


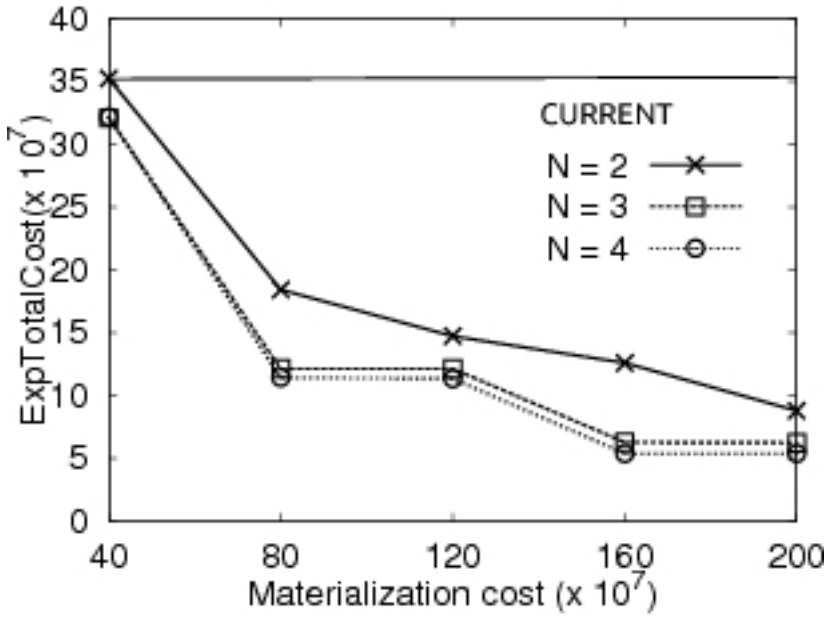

- TCPDS queries and updates
- Number of replicas (N): 3
- Prob. of failure (alpha): 0




- TCPDS queries only
- Number of replicas (N): 3




- TCPDS queries
- Shift to a different query distribution at query 200
- Shift back to original distribution at query 400
- Sliding windows: last 60 queries

# Conclusions

- *Divergent Design* represents a new paradigm for tuning replicated databases
- RITA is a powerful tool for divergent index tuning
  - Offer richer functionality
  - Compute divergent design that result in significantly better performance
- Reduce the problem to a compact BIP