

Quality Contracts for Real-Time Enterprises



Alexandros Labrinidis, Huiming Qu, Jie Xu

Advanced Data Management Technologies Lab

Department of Computer Science

University of Pittsburgh

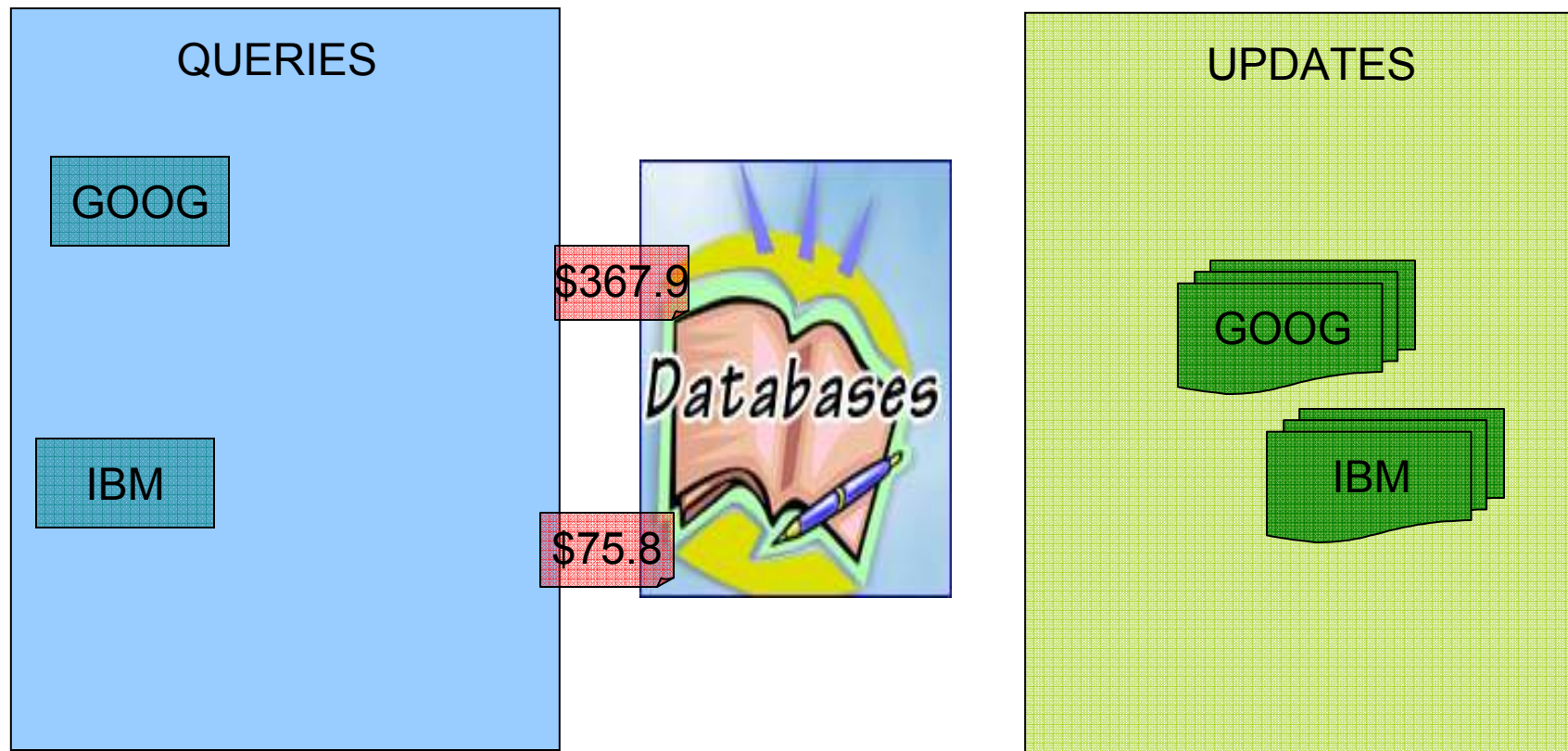
<http://db.cs.pitt.edu>

BIRTE 06 - Business Intelligence for the Real Time Enterprise
September 11, 2006 - Seoul, South Korea (in conjunction with VLDB 2006)

Quality in Real Time Enterprises

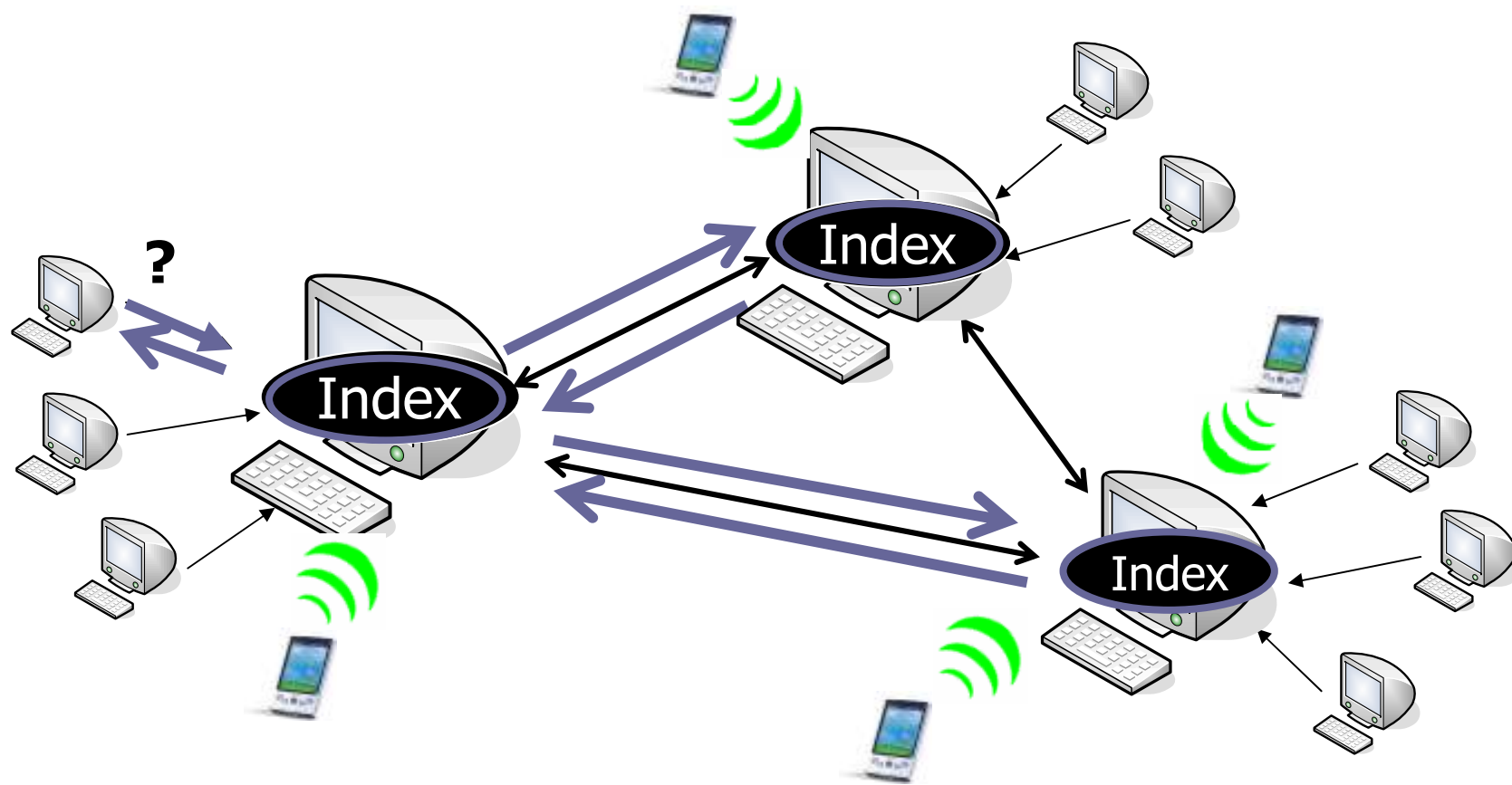
- In real time enterprises:
 - Response time is critical
 - However, it is not the only concern
- **Quality concerns** may include
 - Response time, or **Quality of Service** (QoS)
 - Freshness, or **Quality of Data** (QoD)
- Examples
 - Case 1: web database systems
 - Case 2: distributed database systems

Case 1: Update-intensive Web database system (e.g., stock info portal)



Case 2: Distributed database system

□ Hierarchical loosely-coupled distributed system



Desired properties

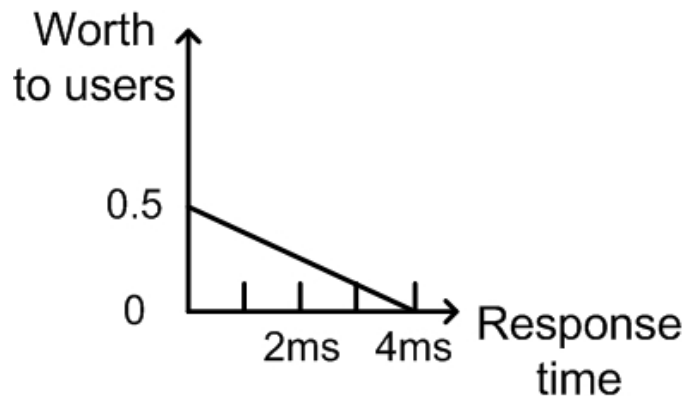
- support **user preferences**:
 - help system survive a broader spectrum of workloads when high number of tasks are competing for CPU
 - User preferences guide system decisions
 - E.g., Web-database, competing queries and updates
 - help system choose the appropriate service site when multiple service providers are competing
 - E.g., distributed database with replication
- provide a unified metric to measure system performance

Outline

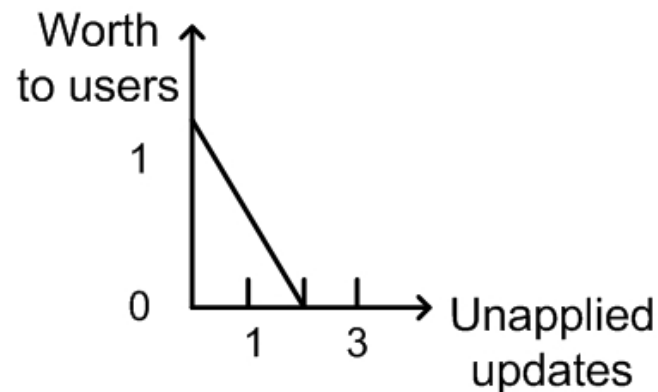
- Introduction
- Quality Contracts Framework (QC)
- Case 1 – Transaction scheduling under QC
- Case 2 – Distributed query processing under QC
- Related work
- Conclusions

Quality Contracts (QC)

- QC is a set of functions that match **performance metrics** to how much they are **worth to users**
- Performance metrics could be
 - **Real metrics** such as *response time (rt)*
 - **Virtual metrics** such as $rt - avg(rt)$



(a) QoS function



(b) QoD function

Optimization under Quality Contracts

- QCs enable the specification of user preferences
 - Among **different quality metrics**
 - Among **different queries**
- Micro-economic Paradigm
 - Servers get profit according to QC for satisfying users' preferences
- Optimization goal:
 - Maximize overall system profit P
 - $P = f(\text{QoS}) + (f(\text{QoS}) > 0 ? f(\text{QoD}) : 0)$

Applicability of Quality Contracts

- Payment for regular DBMS queries
 - e.g., \$10 for ideal execution of one query
- Payment stream for DSMS queries
 - e.g., \$10/hour for a continuous query
- QC framework is applicable for both!

Usability of Quality Contracts

- **Parameterized versions of QCs** reduce the burden of user preference specification
 - servers identify different classes of QCs for each type of user (e.g., silver, gold, platinum levels)
- **Real-life example:**
classes of **pre-configured cell phone plans**
 - a user only needs to choose
 - the **total budget**, and
 - whether his/her **preference** is
 - **QoS** (local plan with more minutes) or
 - **QoD** (national plan with fewer minutes)

Outline

- Introduction
- Quality Contracts Framework (QC)
- Case 1 – Transaction scheduling under QC
- Case 2 – Distributed query processing under QC
- Related work
- Conclusions

Case 1 - transaction scheduling under quality contracts

□ Transaction types

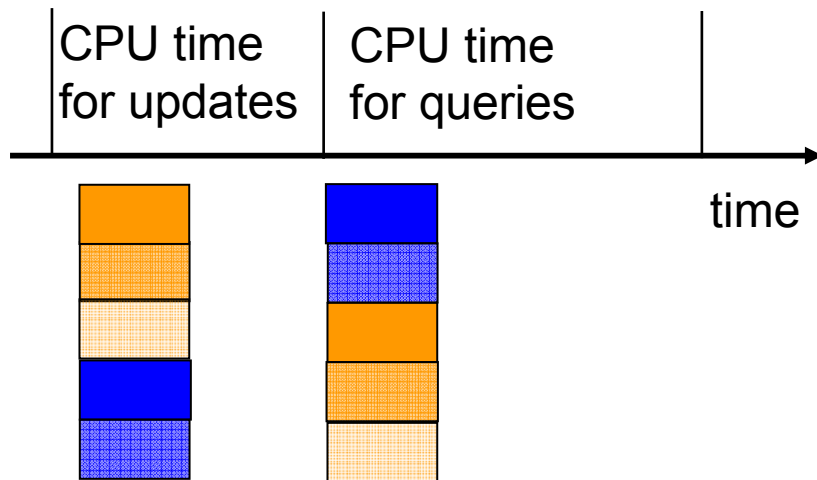
- Read-only queries, write-only updates
- Each query is associated with a QC
- Queries and updates are competing for system CPU
 - more cpu to queries, higher profit from $f(\text{QoS})$
 - more cpu to updates, higher profit from $f(\text{QoD})$

□ Optimization Goal

- Maximize overall system profit through balancing the load of query and update transactions

QUTS - overview

- Dual priority queue with dynamic concatenation

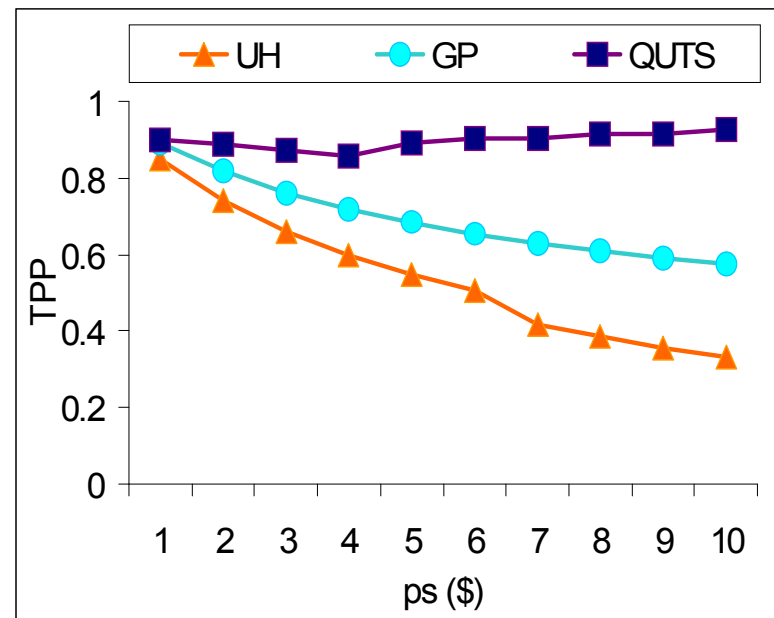
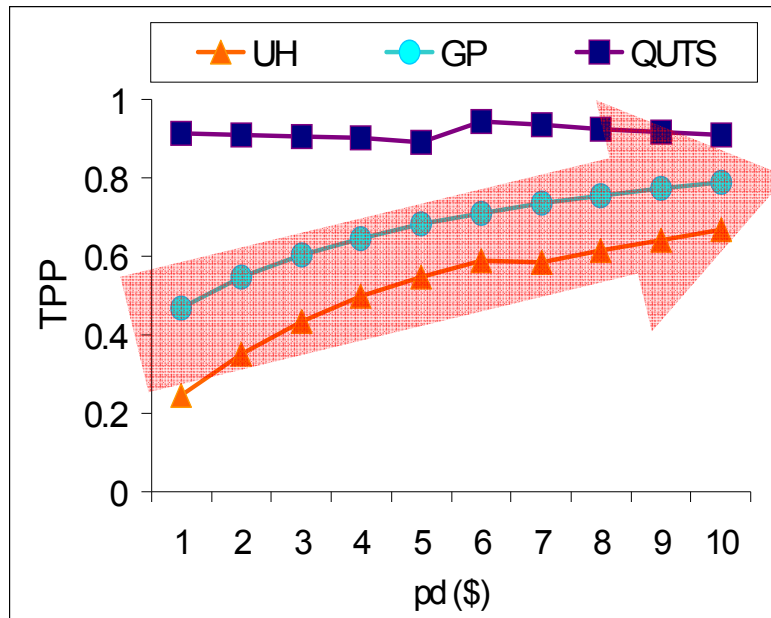


- Two-level scheduling scheme

- At the **high** level
 - resource allocation (which queue to execute) according to the QCs.
- At the **low** level
 - Query queue uses PRD
 - Update queue uses HDP

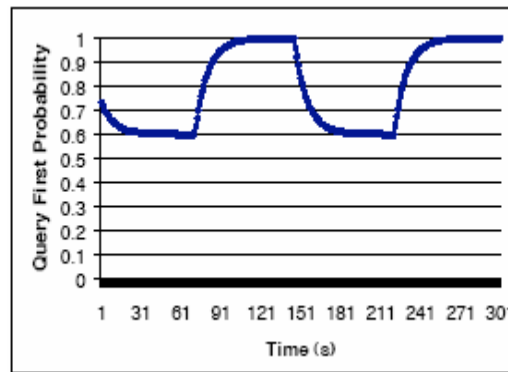
Exp 1. Performance (pd and ps)

- QUTS consistently achieves over 85% profit over the entire ranges.
- Both **UH (Update-High)** and **GP (Global-Priority)** have are biased for freshness, thus performance increases while QoD worth increases.

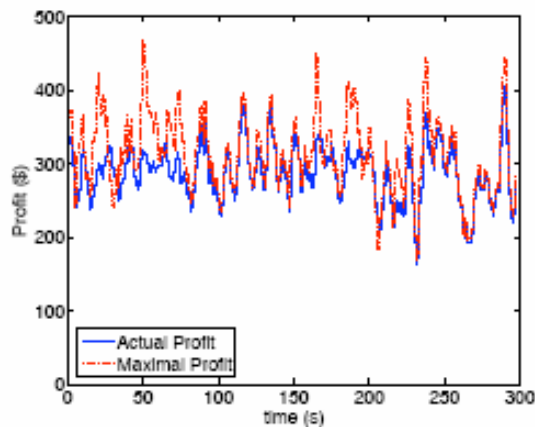


Exp 2. Adaptability

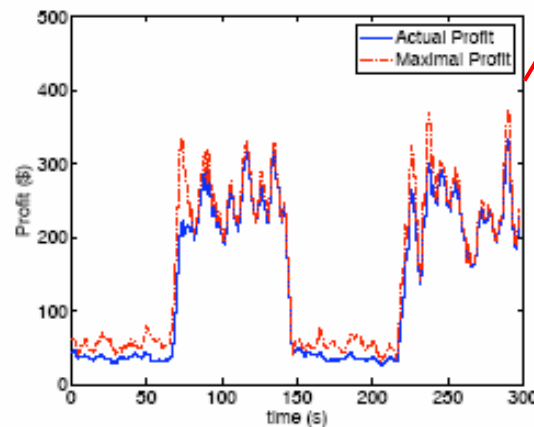
- Time is divided into four segments.
- In odd segments,
 - $pd = 5 \times ps$
- In even segments,
 - $ps = 5 \times pd$



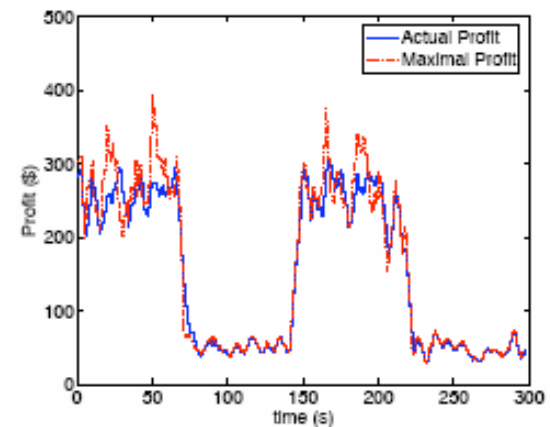
Similar pattern



(a) total profit



(b) QoS profit



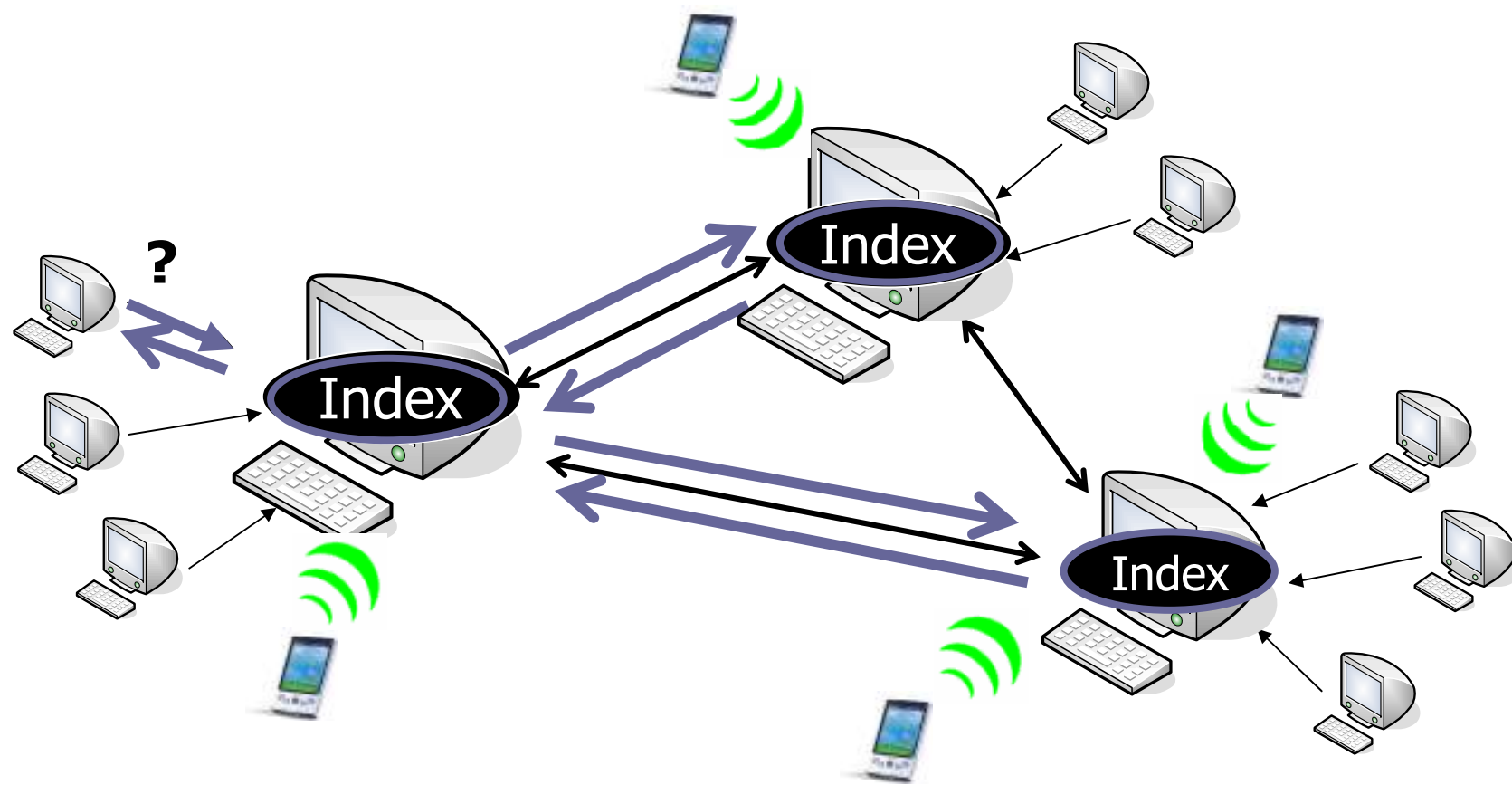
(d) QoD profit

Outline

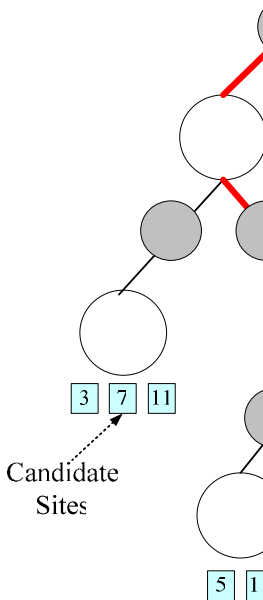
- Introduction
- Quality Contracts Framework (QC)
- Case 1 – Transaction scheduling under QC
- Case 2 – Distributed query processing under QC
- Related work
- Conclusions

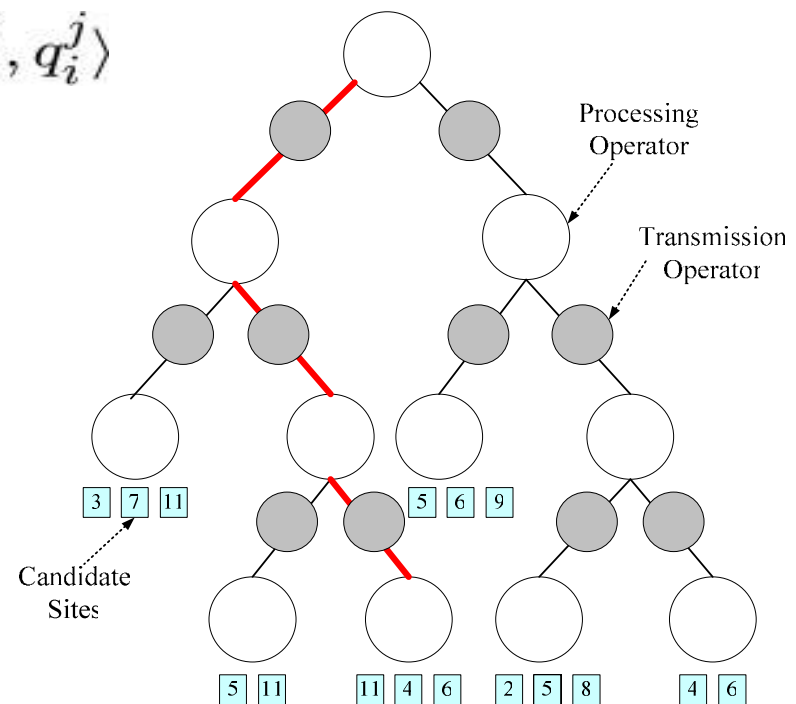
Case 2 – Distributed query processing under quality contracts

□ Hierarchical loosely-coupled distributed system



Replication Selection Problem

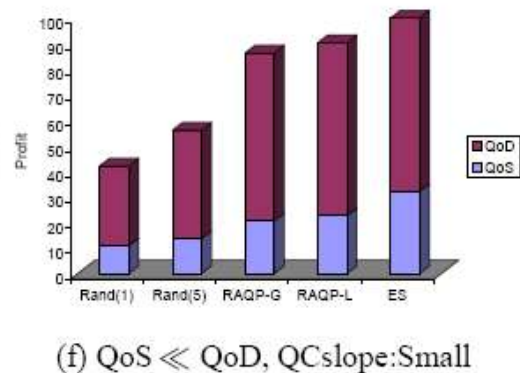
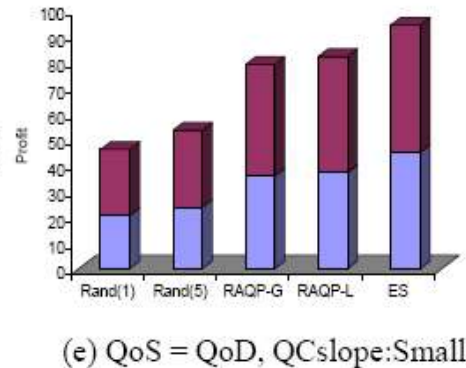
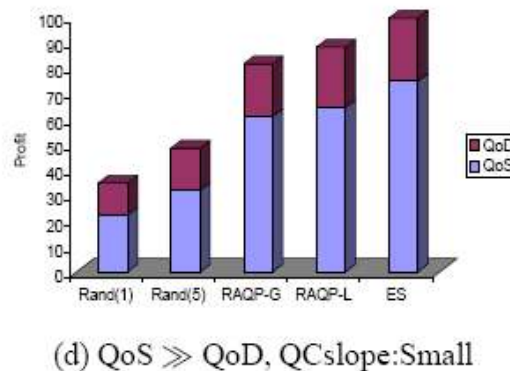
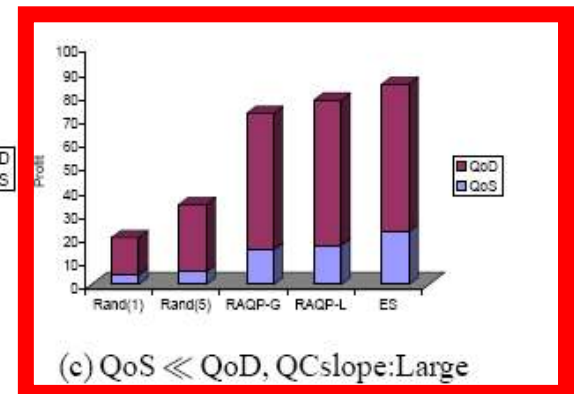
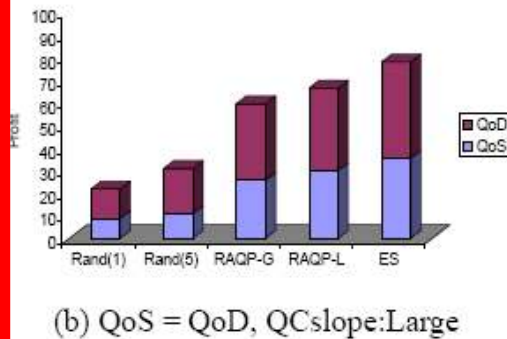
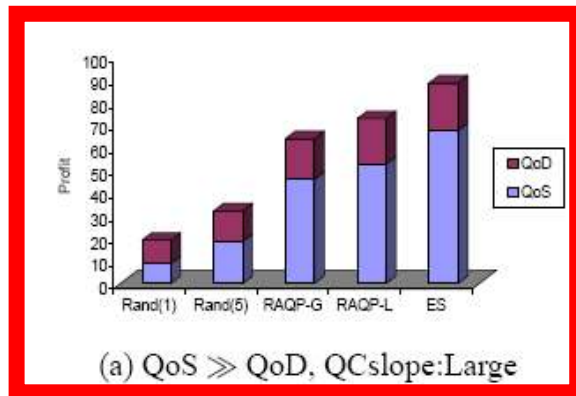
- Data $D = \{1, 2, \dots, m\}$
 - Replication $S = \{R_i^1, R_i^2, \dots, R_i^n\}$
 - Each replication $R_i^j \langle p_i^j, s_i^j, q_i^j \rangle$
 - Query execution plan
 - Processing operator
 - Transmission operator
 - Goal:
 - Label each replica as winning or losing
 - Maximize processing node's revenue
- 



Replication-Aware Query Processing (RAQP)

- Two-step query optimization
 - Generate statically-optimized logical execution plan
 - Dynamic programming algorithm [system R]
 - Lifted constraint of left-deep trees
 - Select execution site for each operator and replica to use
 - *Initial Query Unit Allocation*
 - *Iterative Improvement*
- [Xu & Labrinidis, WebDB 2006]

Experimental Results - Profit



Algorithm adapts well to user preferences

Related work

□ Economic Models

- [Ferguson et al., 1996]
- [Stonebraker et al., 1996]

□ Real time DB and Web-DB

- [Abbott et al., 1988]
- [Sha et al., 1991]
- [Haritsa et al., 1993]
- [Haritsa et al., 1993]
- [Ramamritham et al., 1994]
- [Burns et al., 2000]
- [Challenger et al. 2000]
- [Luo et al. 2002]
- [Datta et al. 2002]
- [Labrinidis et al. 2004]

□ Stream Processing

- [Carney et al., 2002]
- [Das et al., 2003]
- [Babcock et al., 2004]
- [Sharaf et al., 2005]
- [Abadi et al., 2005]

□ Query Optimization

- System R [Selinger79]
- Volcano [Graefe89, Cole94]
- Eddies [Avnur00], Rio [Babu05]
- R* [Mackert86]
- LBQP [Carey86]
- Distributed eddies [Tian03]

Conclusions

- We proposed **Quality Contracts**
 - a unified framework to specify user preferences on **multiple quality metrics**.
- We showed two application examples which utilize the Quality Contracts framework
 - **Transaction scheduling** in update-intensive Web-databases
 - **Replica selection** in distributed databases
- Future work
 - Extend QC framework to other domains

Thank you

Alex(andros) Labrinidis
labrinid@cs.pitt.edu

Questions

