Database Scalabilty, Elasticity, and Autonomic Control in the Cloud

Divy Agrawal
Department of Computer Science
University of California at Santa Barbara

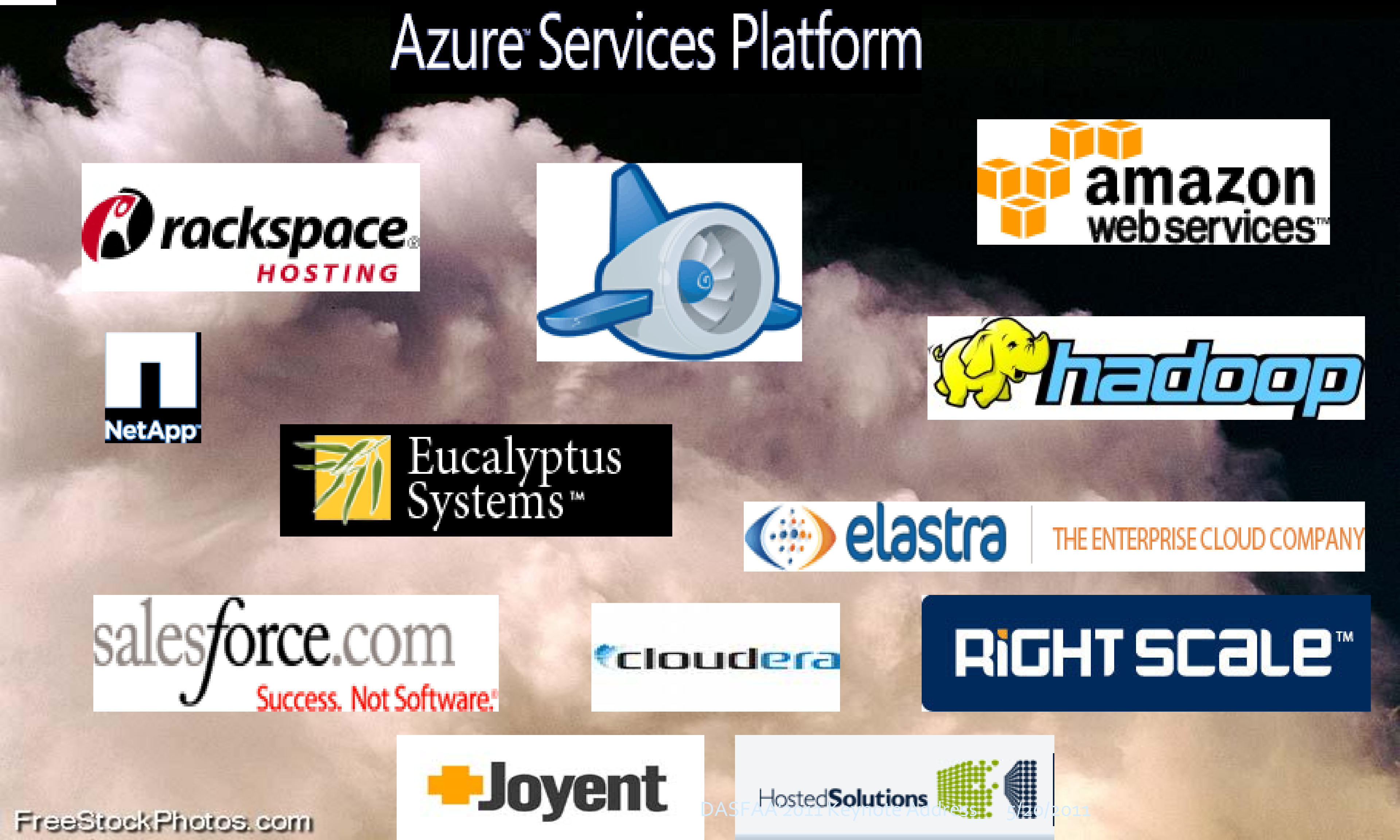
Collaborators: Amr El Abbadi, Sudipto Das, Aaron Elmore

Outline

- Infrastructure Disruption
  - Enterprise owned → Commodity shared infrastructures
  - Disruptive transformations: Software and Service Infrastructure
- Clouded Data Management
  - State of the Art lacks “cloud” features
  - Alternative data management models
  - Application development landscape
- Architecting Data Systems for the Cloud
  - Design Principles
  - Data Fusion and Fission
  - Elasticity
  - Autonomic Control

WEB is replacing the Desktop

Paradigm Shift in Computing

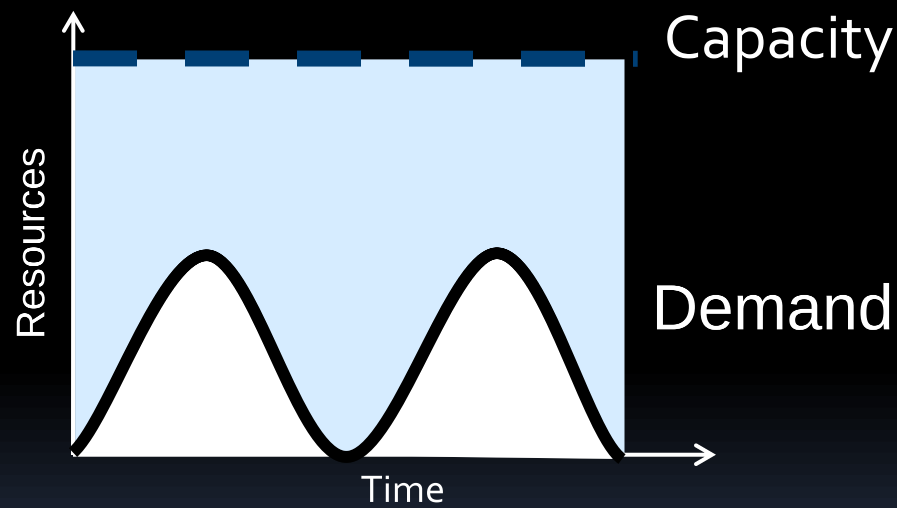


Cloud Computing: Why Now?

- Experience with very large datacenters
  - Unprecedented economies of scale
  - Transfer of risk
- Technology factors
  - Pervasive broadband Internet
  - Maturity in Virtualization Technology
- Business factors
  - Minimal capital expenditure
  - Pay-as-you-go billing model

Economics of Data Centers

- Risk of over-provisioning: underutilization



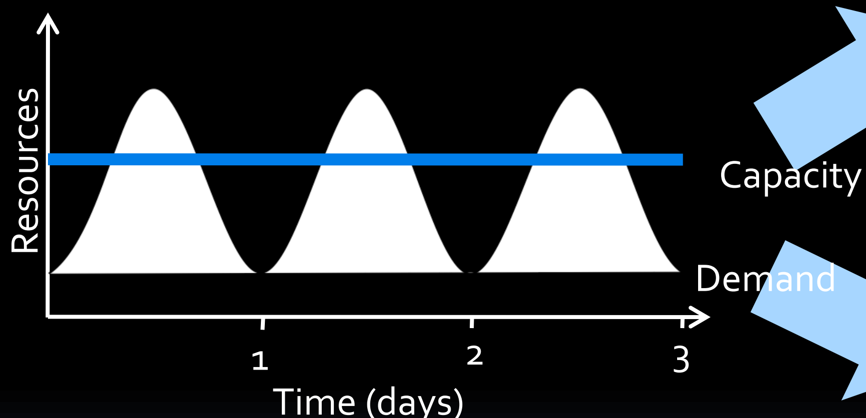
Static data center

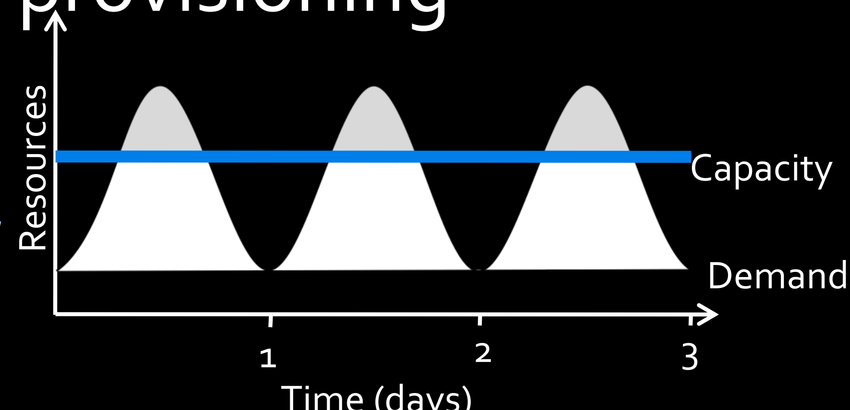
Money & Time Questions:

1. How much?
2. How Long?

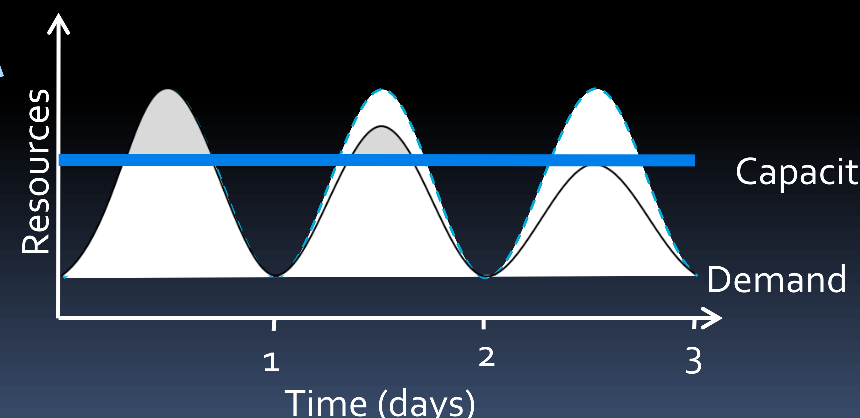
Economics of Internet Users

- Heavy penalty for under-provisioning





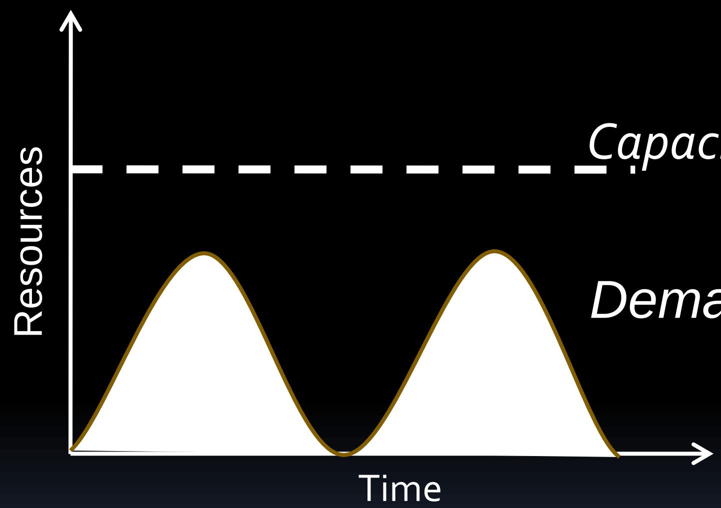
Lost revenue



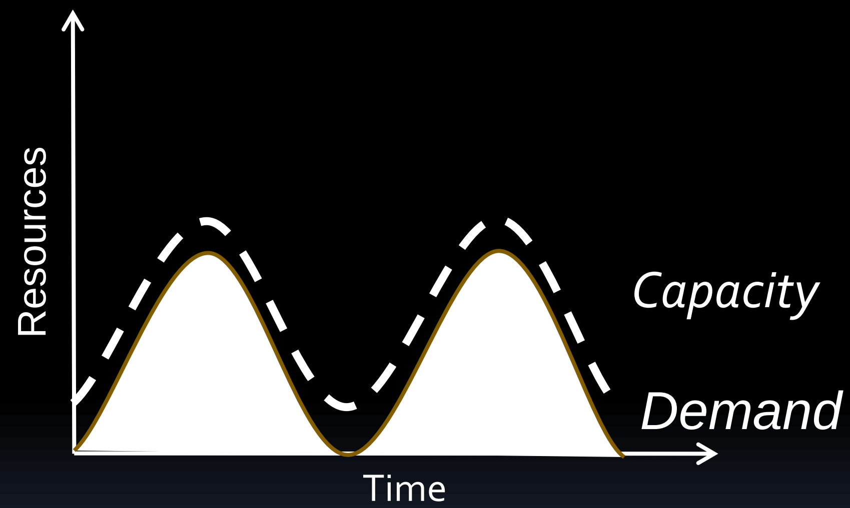
Lost users

Economics of Cloud Computing

- Pay by use instead of provisioning for peak



Static data center



Data center in the cloud

The Big Picture

- Unlike the earlier attempts:
  - Distributed Computing, Distributed Databases, Grid Computing
- Cloud Computing is REAL:
  - Organic growth: Google, Yahoo, Microsoft, and Amazon
  - IT Infrastructure Automation
  - Economies-of-scale
  - Fault-tolerance: automatically deal with failures
  - Time-to-market: no upfront investment

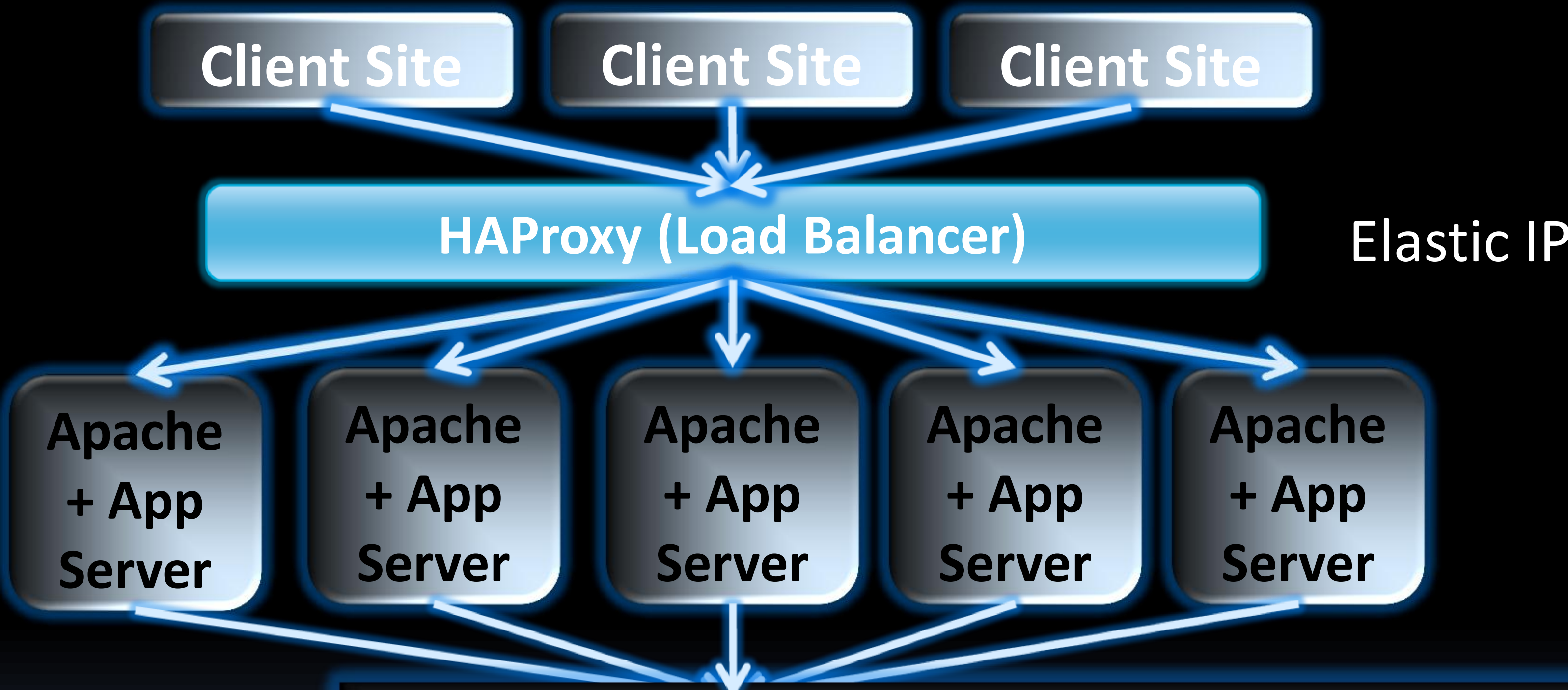
Cloud Reality

- Facebook Generation of Application Developers
- Animoto.com:
  - Started with 50 servers on Amazon EC2
  - Growth of 25,000 users/hour
  - Needed to scale to 3,500 servers in 2 days (RightScale@SantaBarbara)
- Many similar stories:
  - RightScale
  - Joyent
  - …

Outline

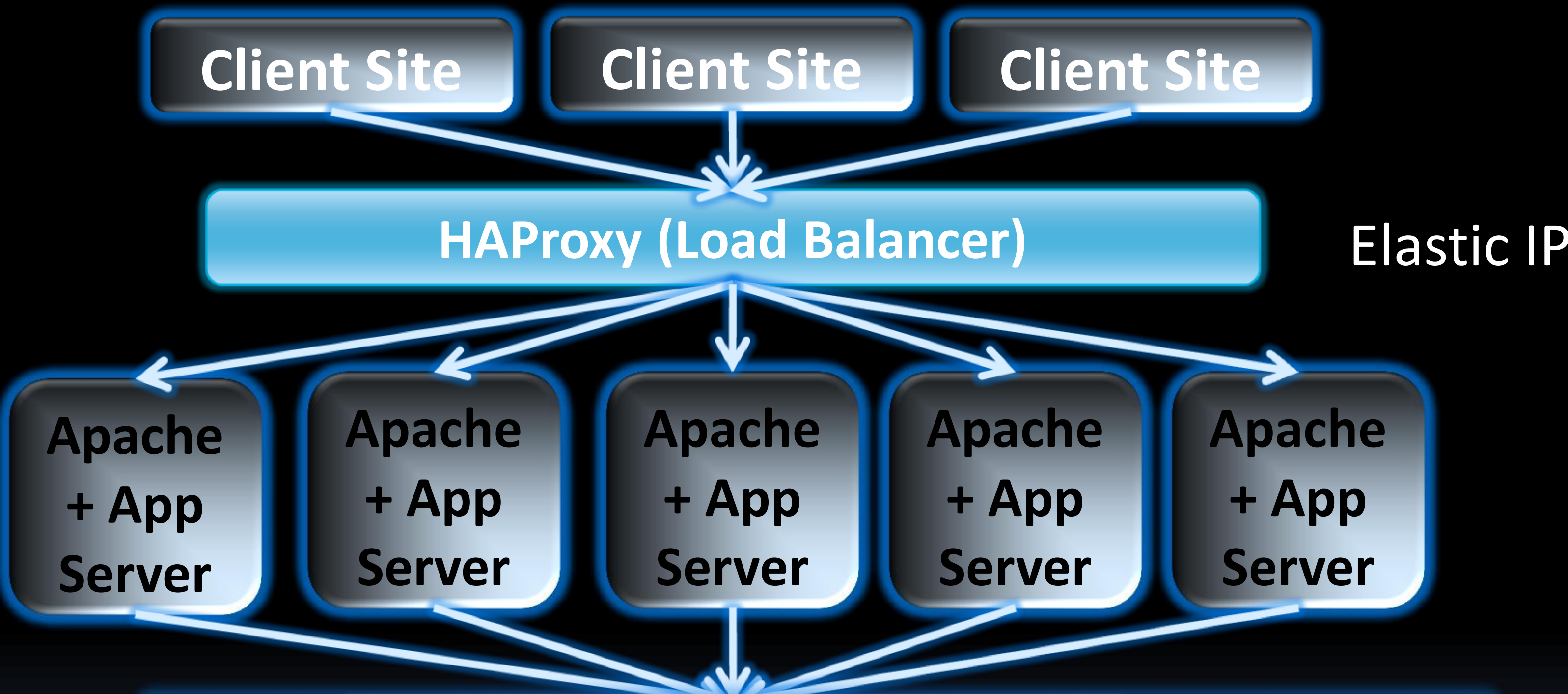
- Infrastructure Disruption
  - Enterprise owned → Commodity shared infrastructures
  - Disruptive transformations: Software and Service Infrastructure
- Clouded Data Management
  - State of the Art lacks "cloud" features
  - Alternative data management models
  - Application development landscape
- Architecting Data Systems for the Cloud
  - Design Principles
  - Data Fusion and Fission
  - Elasticity
  - Autonomic Control

Scaling in the Cloud



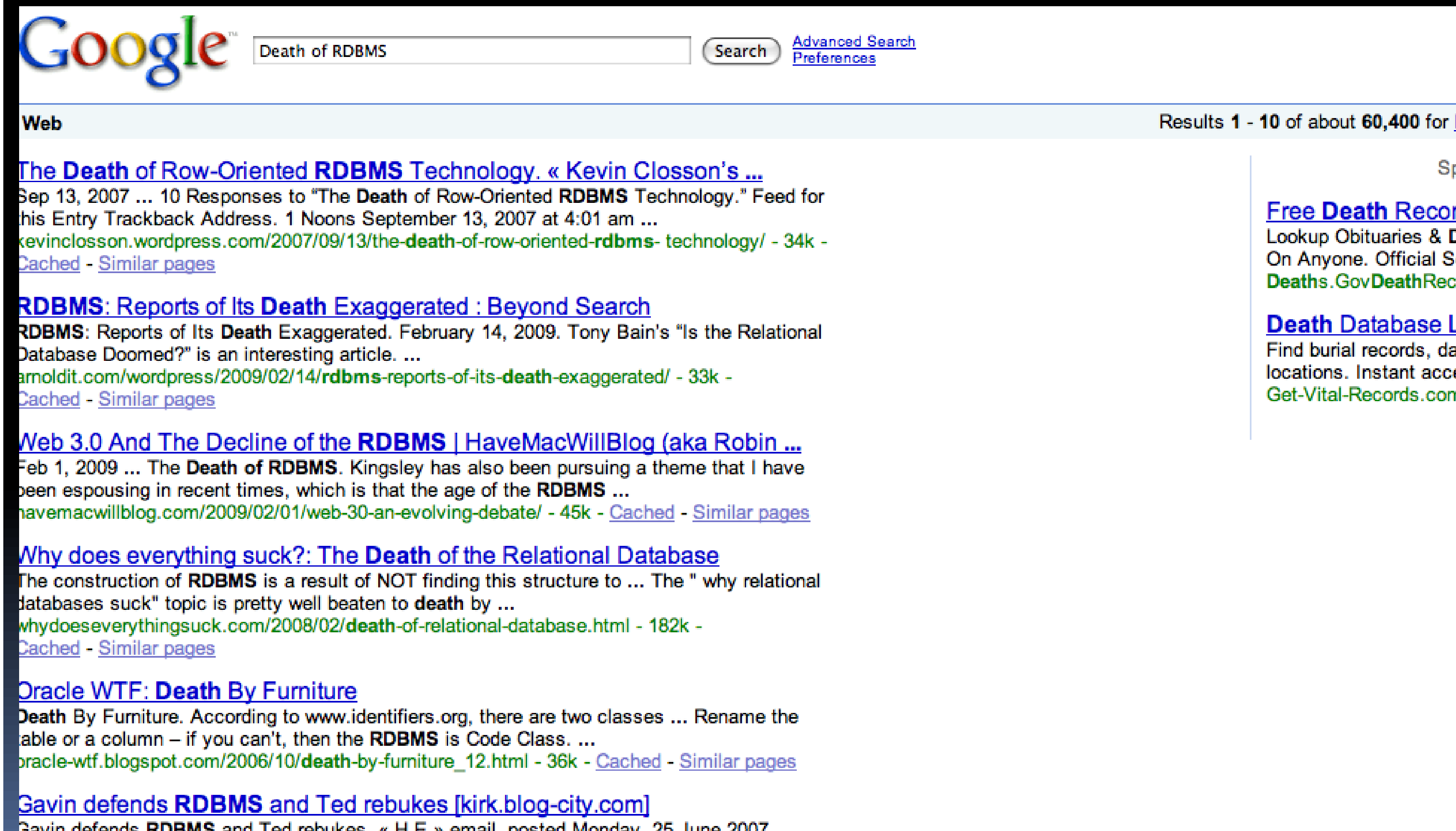
Database becomes the Scalability Bottleneck
Cannot leverage elasticity

Scaling in the Cloud



Scalable and Elastic
But limited consistency and operational flexibility

Internet Chatter



Application Simplicity using DBMS

```
public void confirm_friend_request(Alice, Bob)
{
begin_transaction();
	update_friend_list(Alice, Bob); // Tokyo
	update_friend_list(Bob, Alice); // Hong Kong
end_transaction();
}
```

Application Challenges using Key-Value Stores

```
public void confirm_friend_request(Alice, Bob){
  try
    { update_friend_list(Alice, Bob); // Tokyo }
  catch(exception e)
    { report_error(e);  return;}
  try
   { update_friend_list(Bob, Alice); // Hong Kong}
  catch(exception e)
  {   report_error(e);
      revert_friend_list(Alice, Bob);              n; }
}
```

Eventual Consistency Model

```
public void confirm_friend_request_B(Alice, Bob){
 try
 {  update_friend_list(Alice, Bob); // Tokyo}
 catch(exception e)
 {  add_to_retry_queue(<updatefriendlist, Alice, Bob>); }
 try
 { update_friend_list(Bob, Alice); // Hong Kong}   catch(exception e)
 { add_to_retry_queue(<updatefriendlist, Bob, Alice);}  }
```

Eventual Consistency Challenge

/* get_friends() method has to reconcile results r … friends() because there may be data inconsistency due to a conf… ange that was applied from the message queue is contrad… quent change by the user. In this case, status is a bitflag whe… merged and it is up to app developer to figure out what to do. * …

ted with

arantees

I love **eventual consistency** but there are some applications that are much easier to implement with strong consistency. Many like eventual consistency because it allows us to scale-out nearly without bound ***but it does come with a cost in programming model complexity***.

up

r1,

fr … el … }//foreach … return actual_friends; }

I …

Redu

February 24, 2010

Outline

- Infrastructure Disruption
  - Enterprise owned ➔ Commodity shared infrastructures
  - Disruptive transformations: Software and Service Infrastructure
- Clouded Data Management
  - State of the Art lacks “cloud” features
  - Alternative data management models
  - Application development landscape
- Architecting Data Systems for the Cloud
  - Design Principles
  - Data Fusion and Fission
  - Elasticity
  - Autonomic Control

Design Principles: Scalable Systems

- Separate System and Application State
- Limit Application interactions to a single node
- Decouple Ownership from Data Storage
- Limited distributed synchronization is feasible

Scalability of Data in the Cloud

- **Data Fusion**
  - Enrich Key Value stores [Gstore: ACM SoCC'2010, MegaStore: CIDR'2011]
- **Data Fission**
  - Cloud enabled relational databases [ElasTras: HotCloud'2009, Relational Cloud: CIDR'2011,SQL Azure: ICDE'2011]

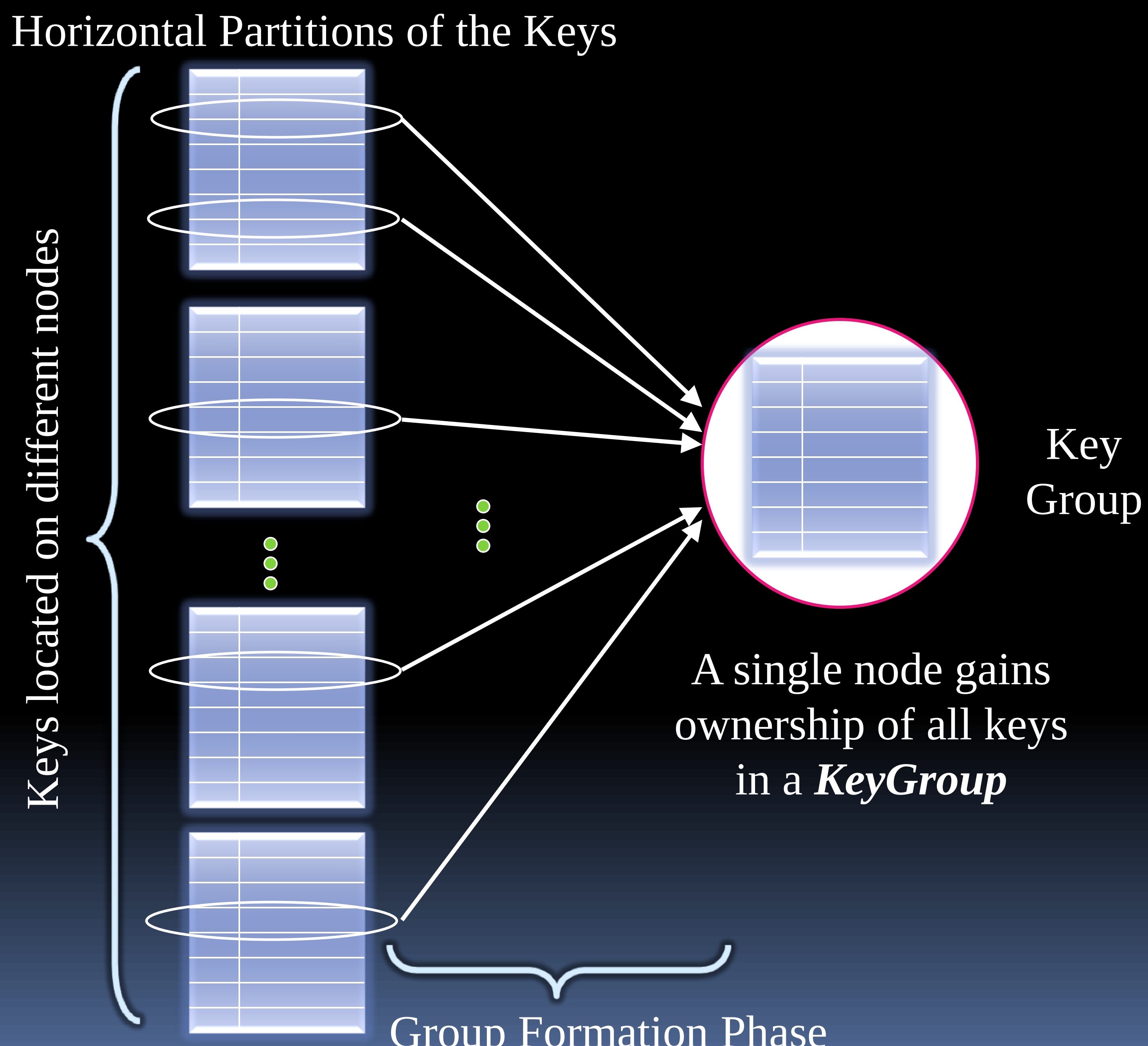
Data Fusion

Atomic Multi-key Access

- Key value stores:
  - Atomicity guarantees on single keys
  - Suitable for majority of current web applications
- Many other applications warrant multi-key accesses:
  - Online multi-player games
  - Collaborative applications
- Enrich functionality of the Key value stores [Google AppEngine & MegaStore]

GStore: Key Group Abstraction
ACM SoCC'2010

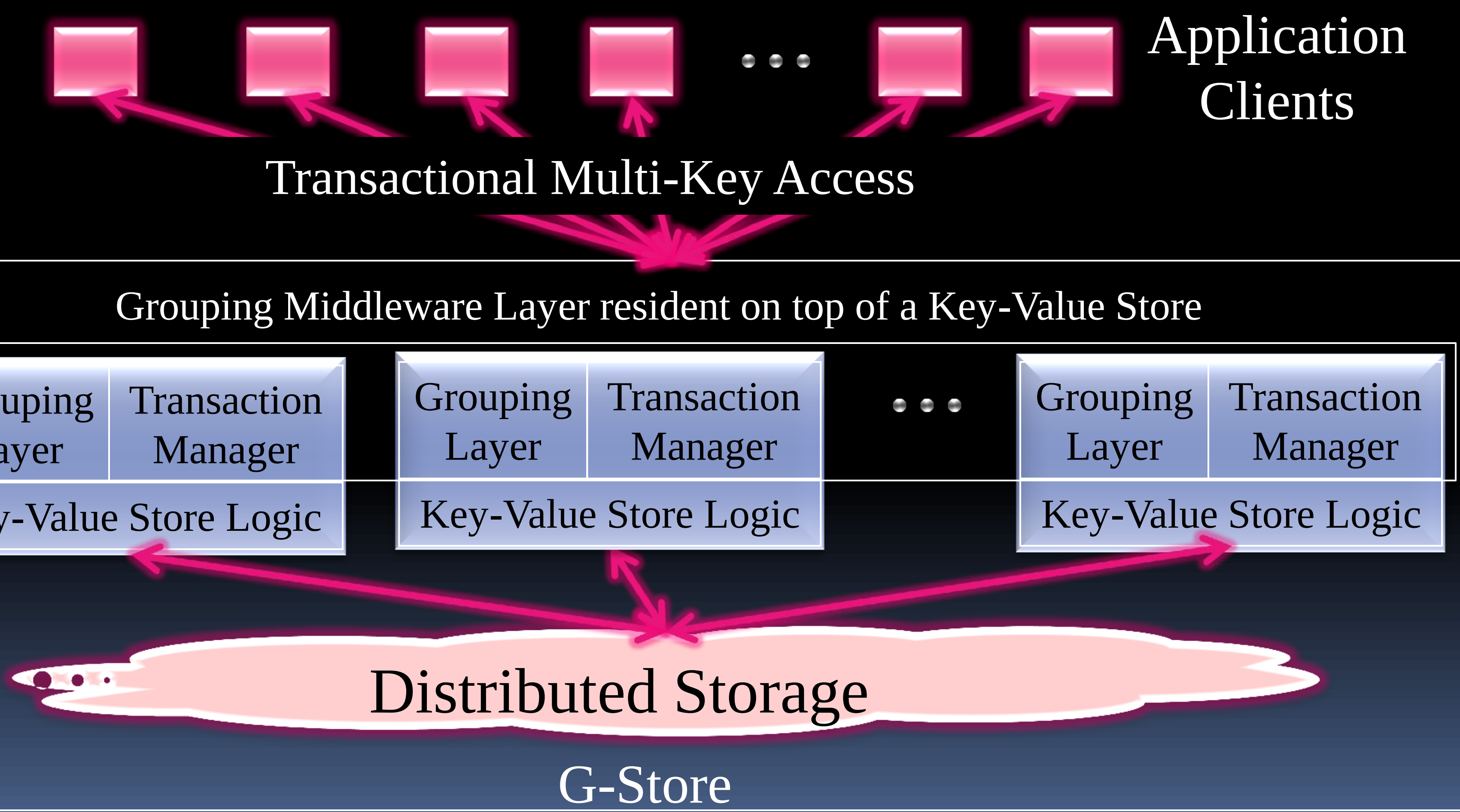
- Define a granule of on-demand transactional access
- Applications select any set of keys
- Data store provides transactional access to the group
- Non-overlapping groups


Horizontal Partitions of the Keys
Keys located on different nodes
Key
Group
A single node gains ownership of all keys in a ***KeyGroup***
Group Formation Phase

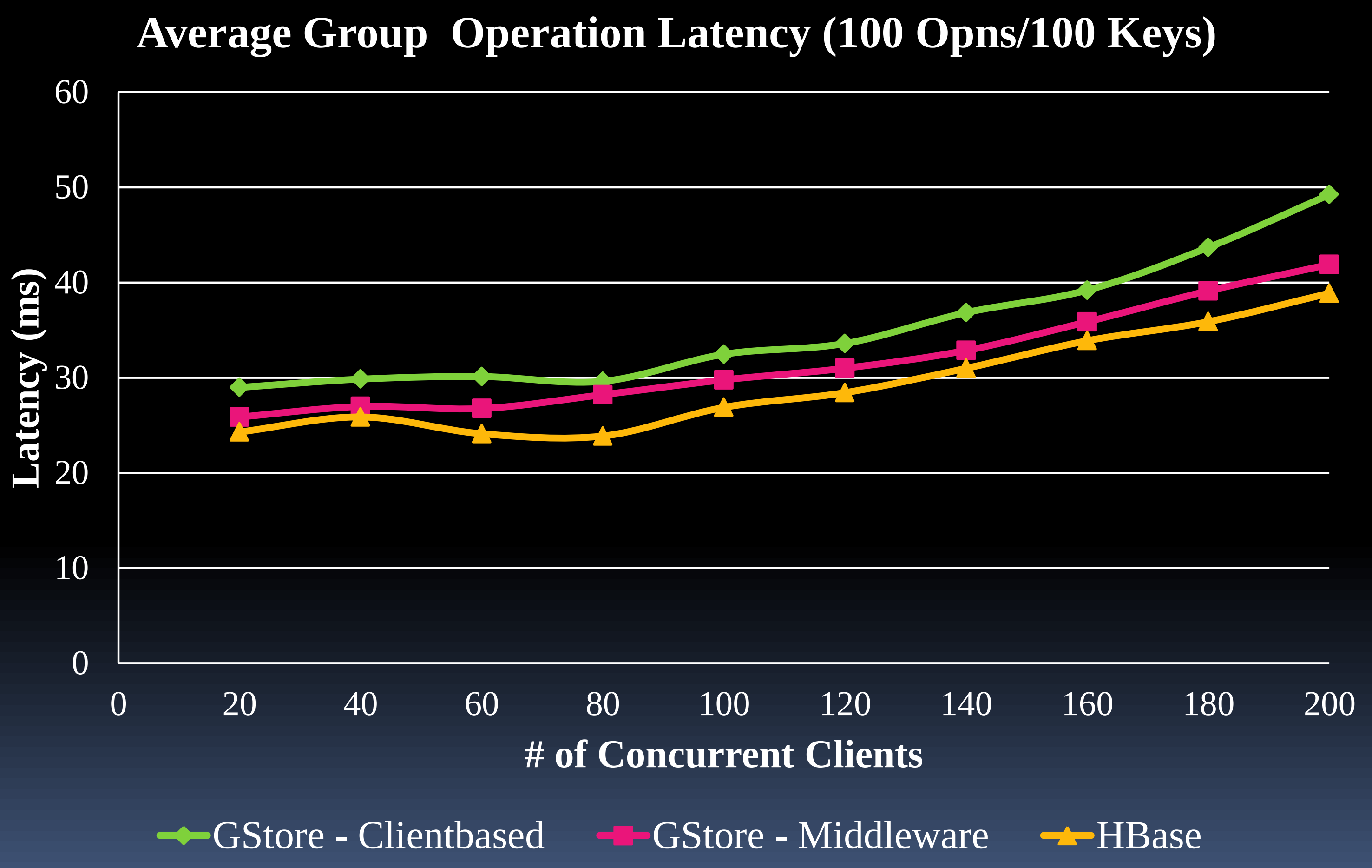
Key Grouping Protocol

- Conceptually akin to "locking"
- Allows collocation of ownership
- Transfer key ownership from "followers" to "leader"
- Guarantee "safe transfer" in the presence of system dynamics:
  - Dynamic migration of data and its control
  - Failures

Implementing GStore



Latency for Group Operations



Google MegaStore
CIDR'2011

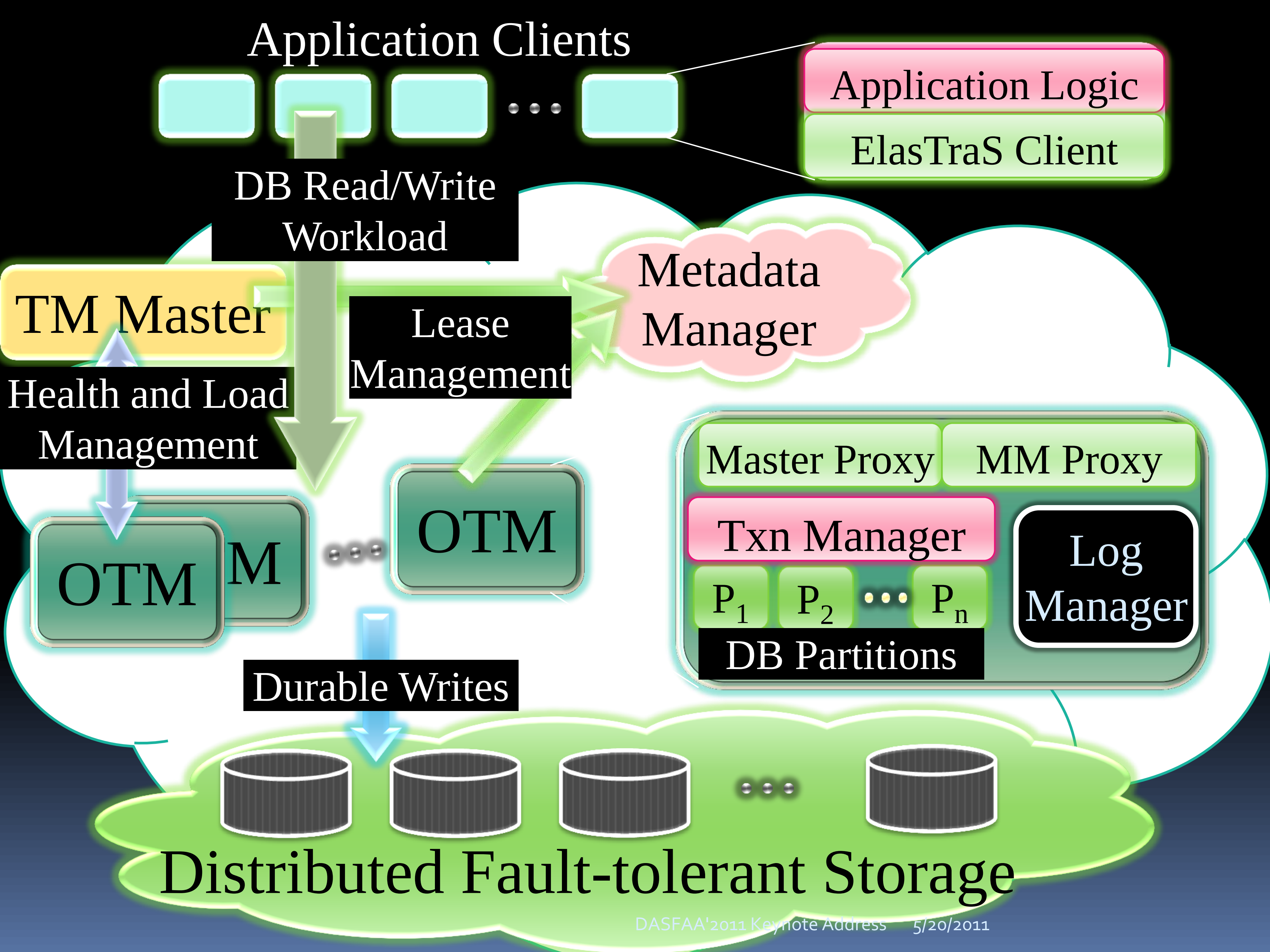
- Transactional Layer built on top of BigTable
- "Entity Groups" form the logical granule for consistent access
  - Entity group: a hierarchical organization of keys
- "Cheap" transactions within entity groups
- Expensive or loosely consistent transactions across entity groups
  - Use 2PC or Persistent Queues
- Transactions over **static** entity groups

Data Fission

Elastic Transaction Management

ElasTras: HotCloud'2009, UCSB TR'2010

- Designed to make RDBMS cloud-friendly
- Database viewed as a collection of partitions
- Suitable for:
  - Large single tenant database instance
    - Database partitioned at the schema level
  - Multi-tenancy with a large number of small DBs
    - Each partition is a self contained database

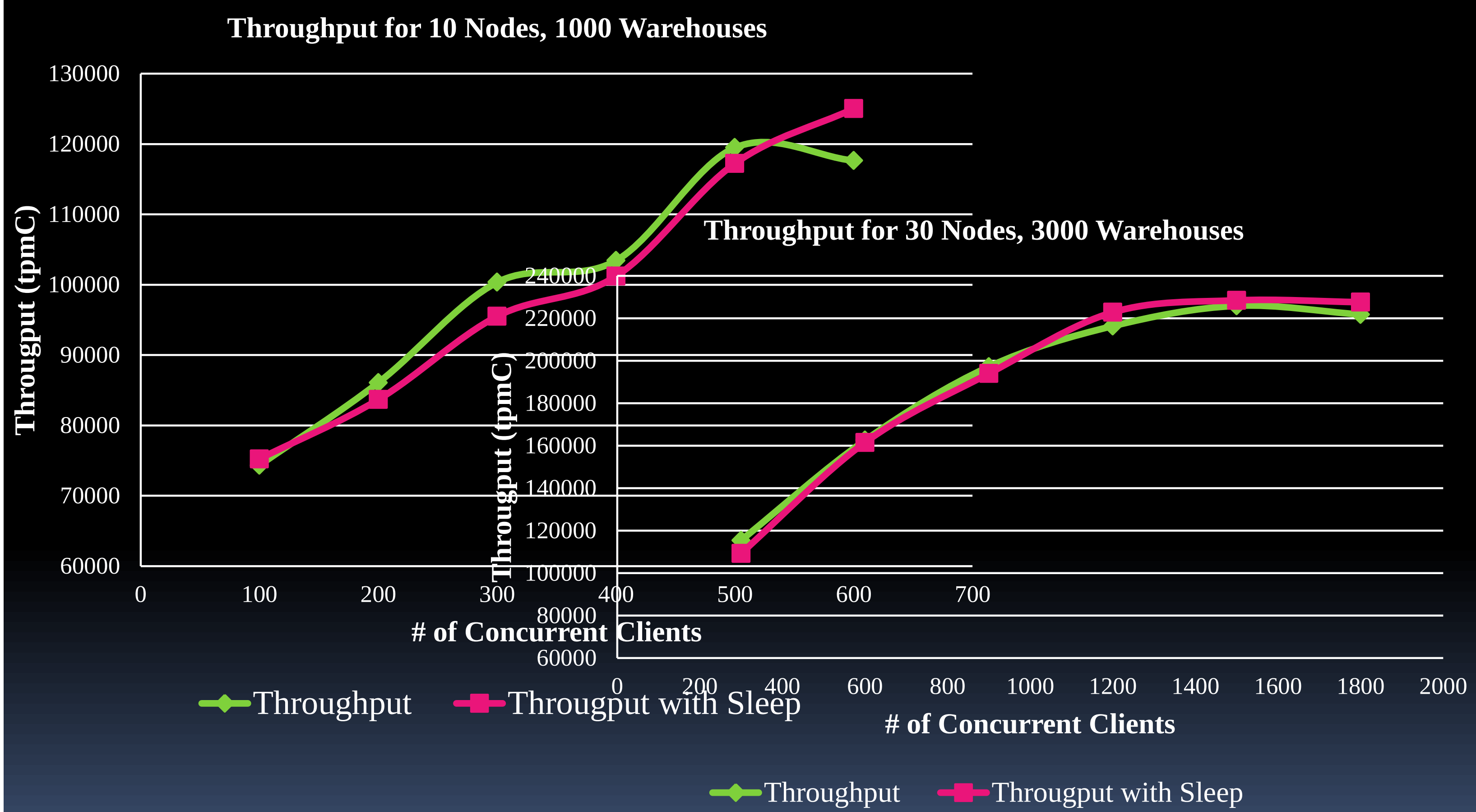


Application Clients
Application Logic
ElasTraS Client
DB Read/Write Workload
TM Master
Lease Management
Metadata Manager
Health and Load Management
OTM
M
OTM
Master Proxy
MM Proxy
Txn Manager
P_1 P_2 P_n
DB Partitions
Log Manager
Durable Writes
Distributed Fault-tolerant Storage

Elastic Transaction Management

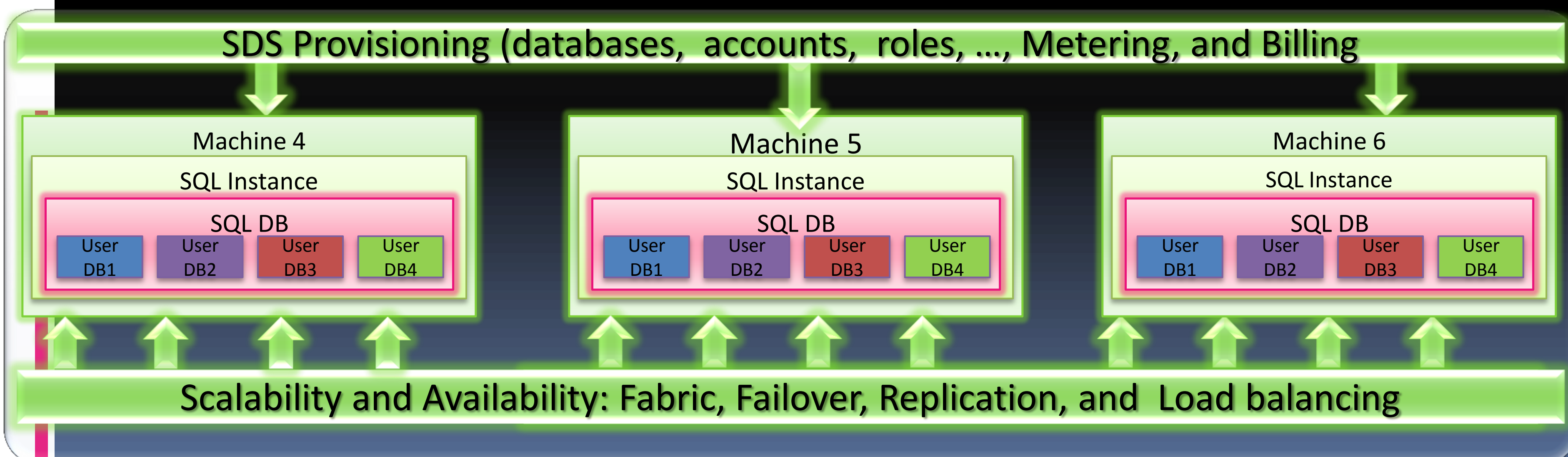
- Elastic to deal with workload changes
- Load balance partitions
- Recover from node failures
- Dynamic partition management
- Transactional access to database partitions

ElasTras: Throughput Evaluation



Microoft Product: SQL Azure
ICDE'2011

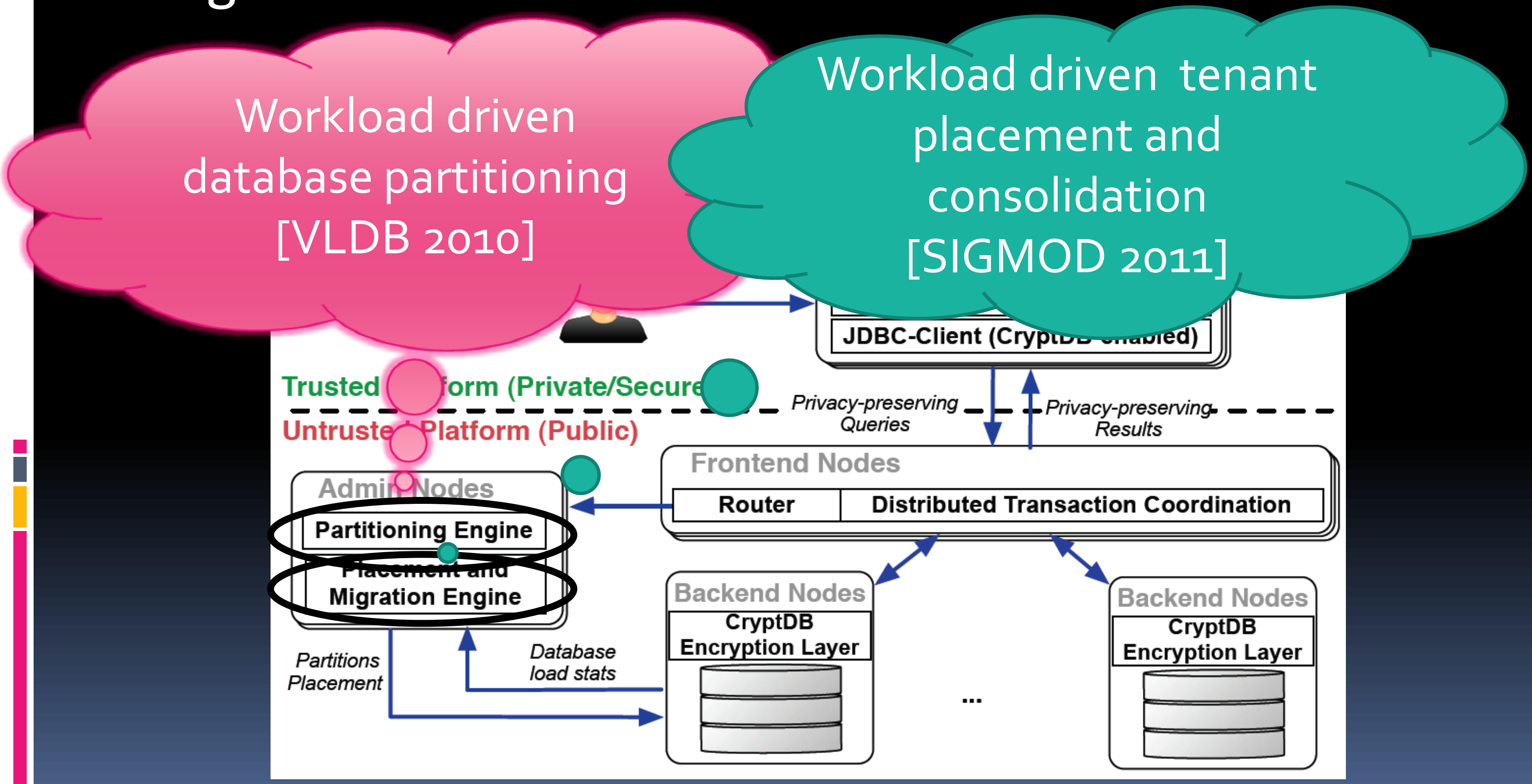
- Shared infrastructure at SQL database and below
  - Request routing, security and isolation
- Scalable HA technology provides the glue
  - Automatic replication and failover
- Provisioning, metering and billing infrastructure



Slides adapted from authors' presentation

MIT Project: Relational Cloud
CIDR'2011

- SQL Server replaced by Open-source database engines

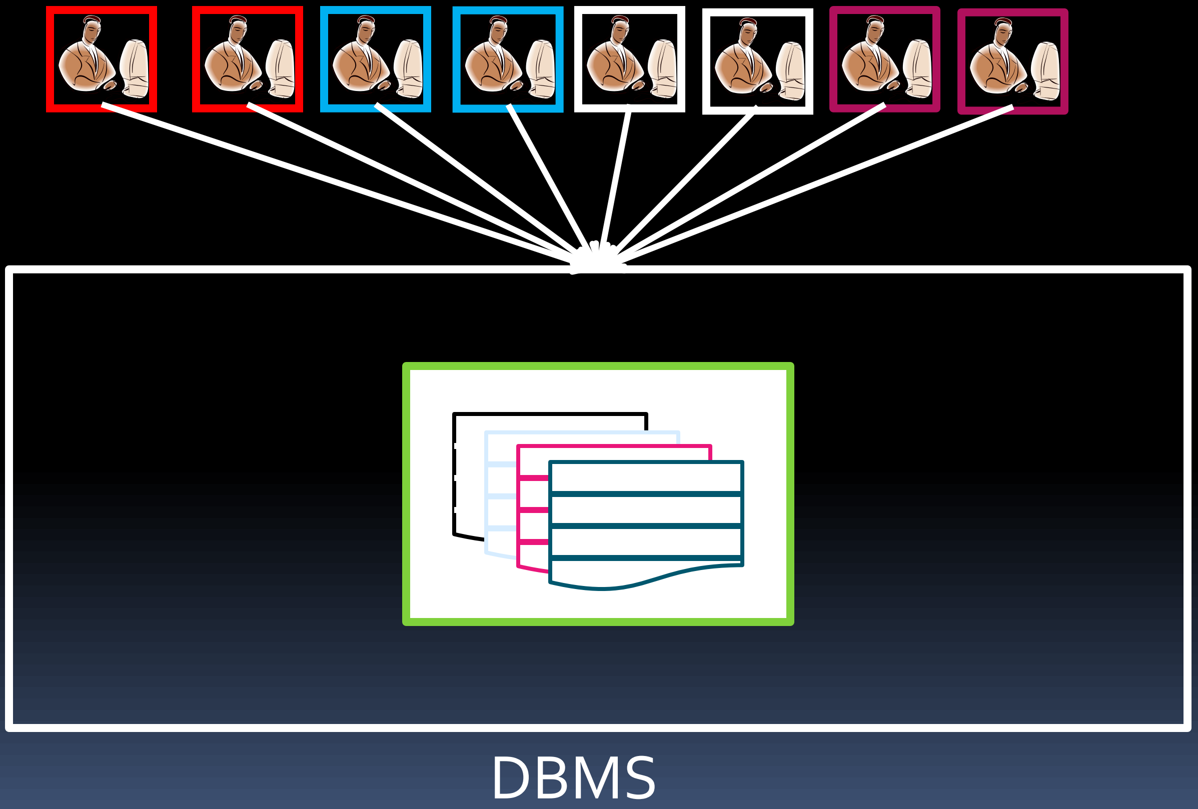


Elasticity in the Cloud: Live Data Migration

Elasticity

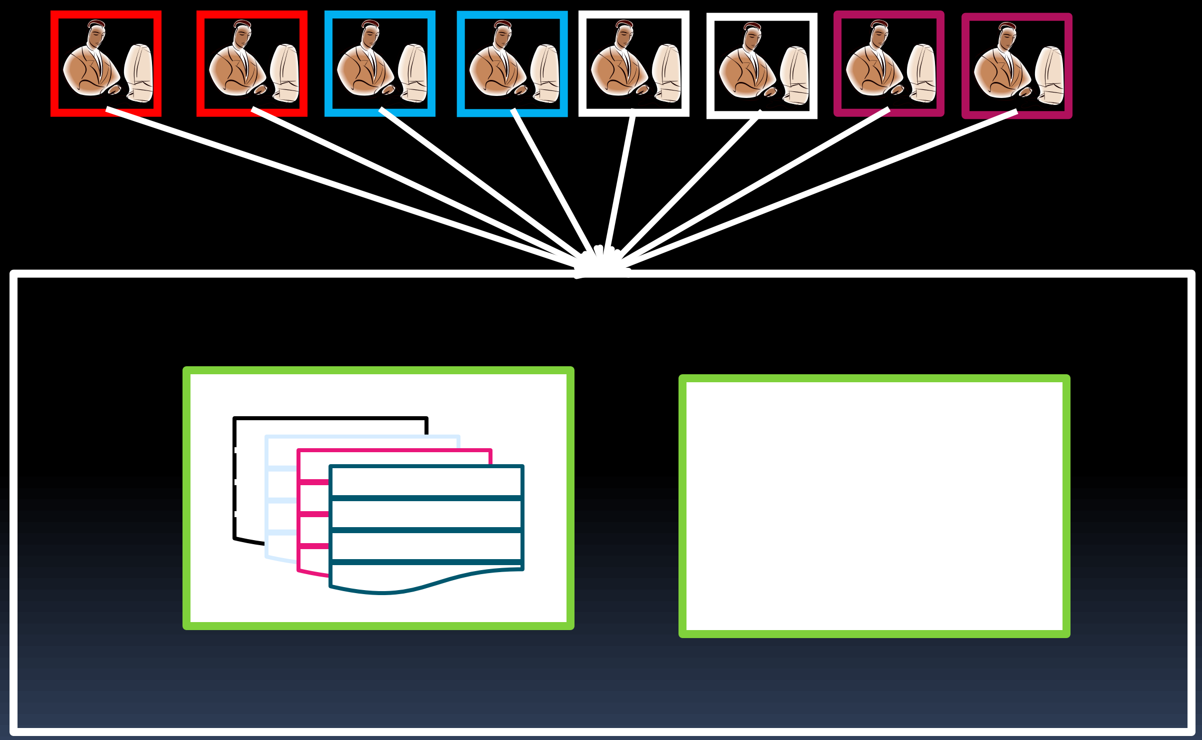
- A database system built over a pay-per-use infrastructure
  - Infrastructure as a Service for instance
- Scale up and down system size on demand
  - Utilize peaks and troughs in load
- Minimize operating cost while ensuring good performance

Elasticity in the Database Layer



Elasticity in the Database Layer

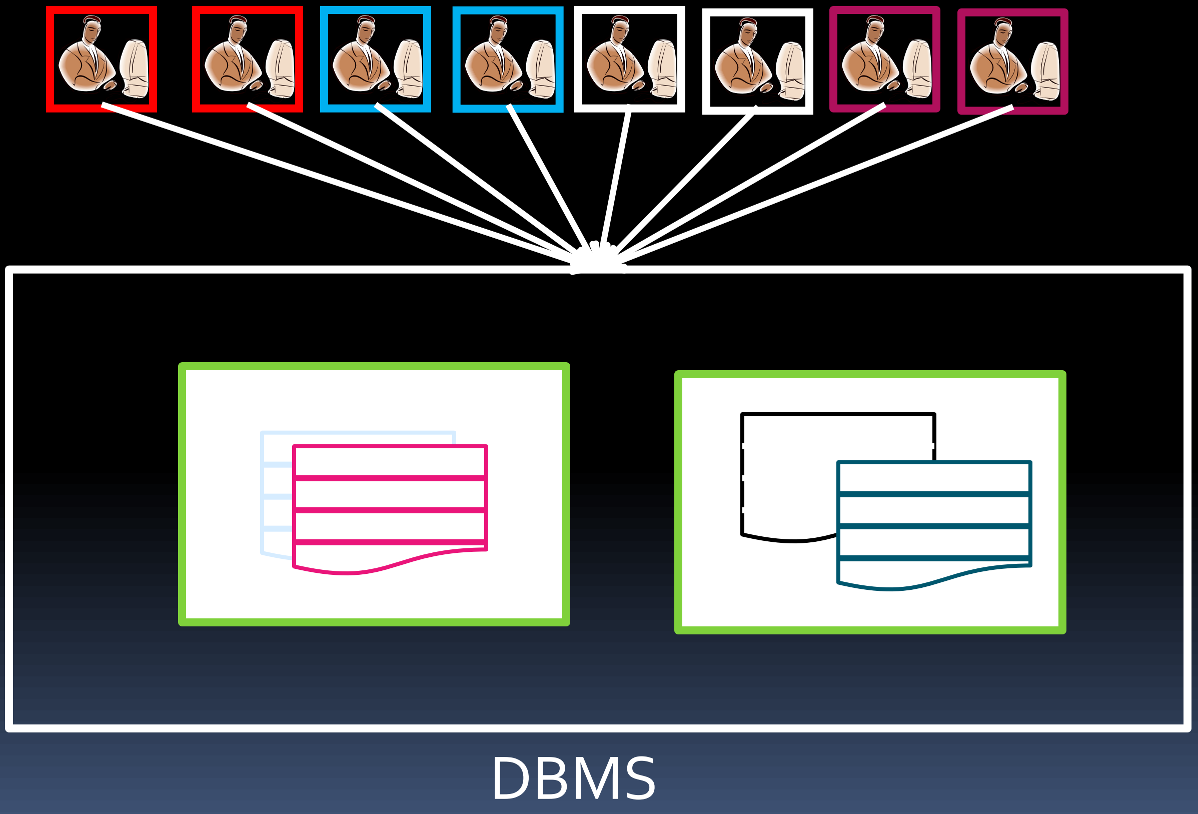
Capacity expansion to deal with high load – Guarantee good performance



DBMS

Elasticity in the Database Layer

Consolidation during periods of low load – Cost Minimization



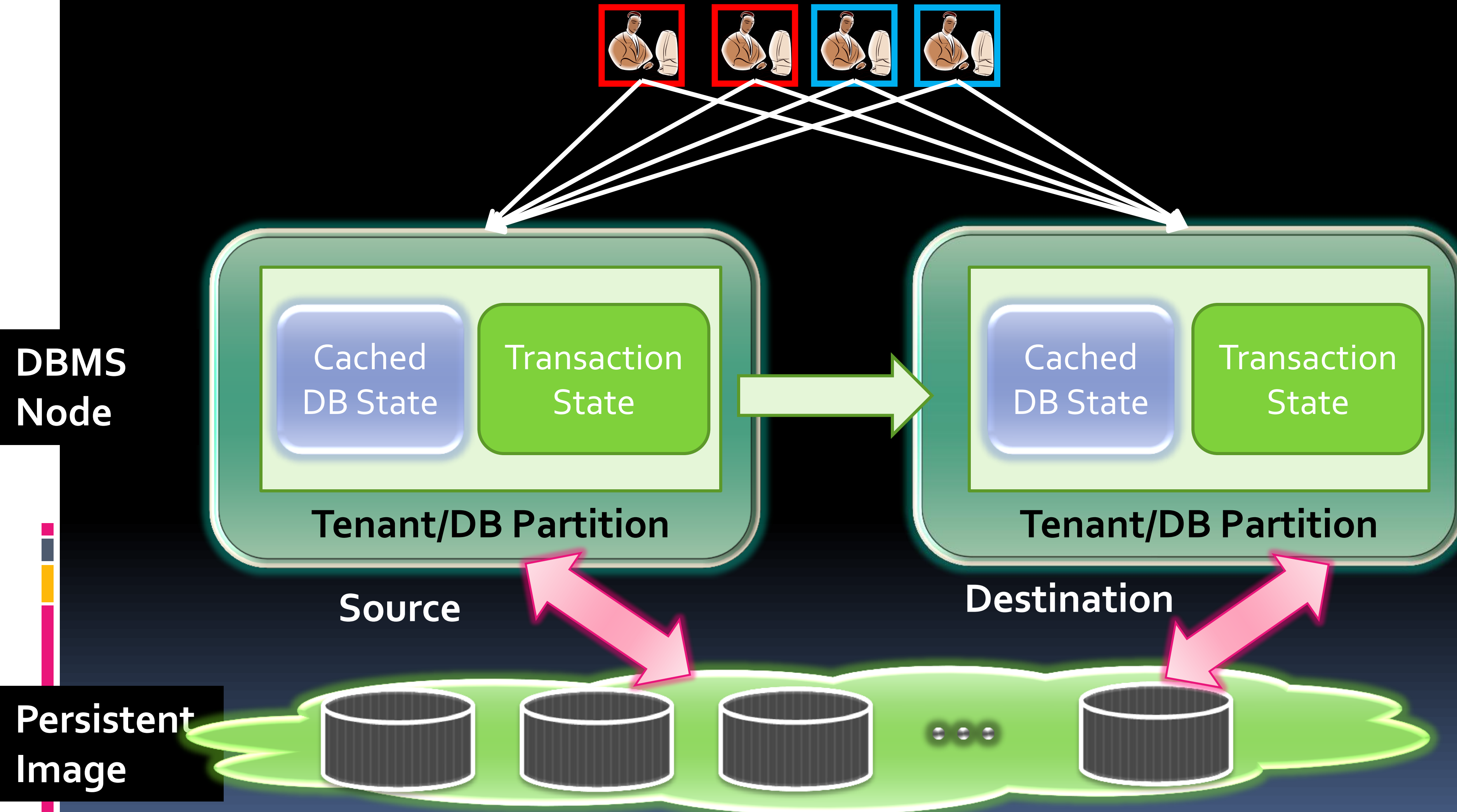
Live Database Migration
A Critical operation for effective elasticity

- Elasticity induced dynamics in a **Live** system
- **Minimal service interruption** for migrating data fragments
  - Minimize operations failing
  - Minimize unavailability window, if any
- Negligible **performance impact**
- **No overhead** during normal operation
- Guaranteed **safety** and **correctness**

Shared storage architecture
Albatross: VLDB 2011

- **Proactive** state migration
  - No need to migrate persistent data
  - Migrate **database cache** and **transaction state** proactively
  - Iteratively copy the state from source to destination
  - Ensure low impact on transaction latency and no aborted transactions
- Migrate transactions on-the-fly
  - Transactions start at source and complete at destination

Albatross



DBMS Node
Cached DB State
Transaction State
Tenant/DB Partition
Cached DB State
Transaction State
Tenant/DB Partition
Source
Destination
Persistent Image

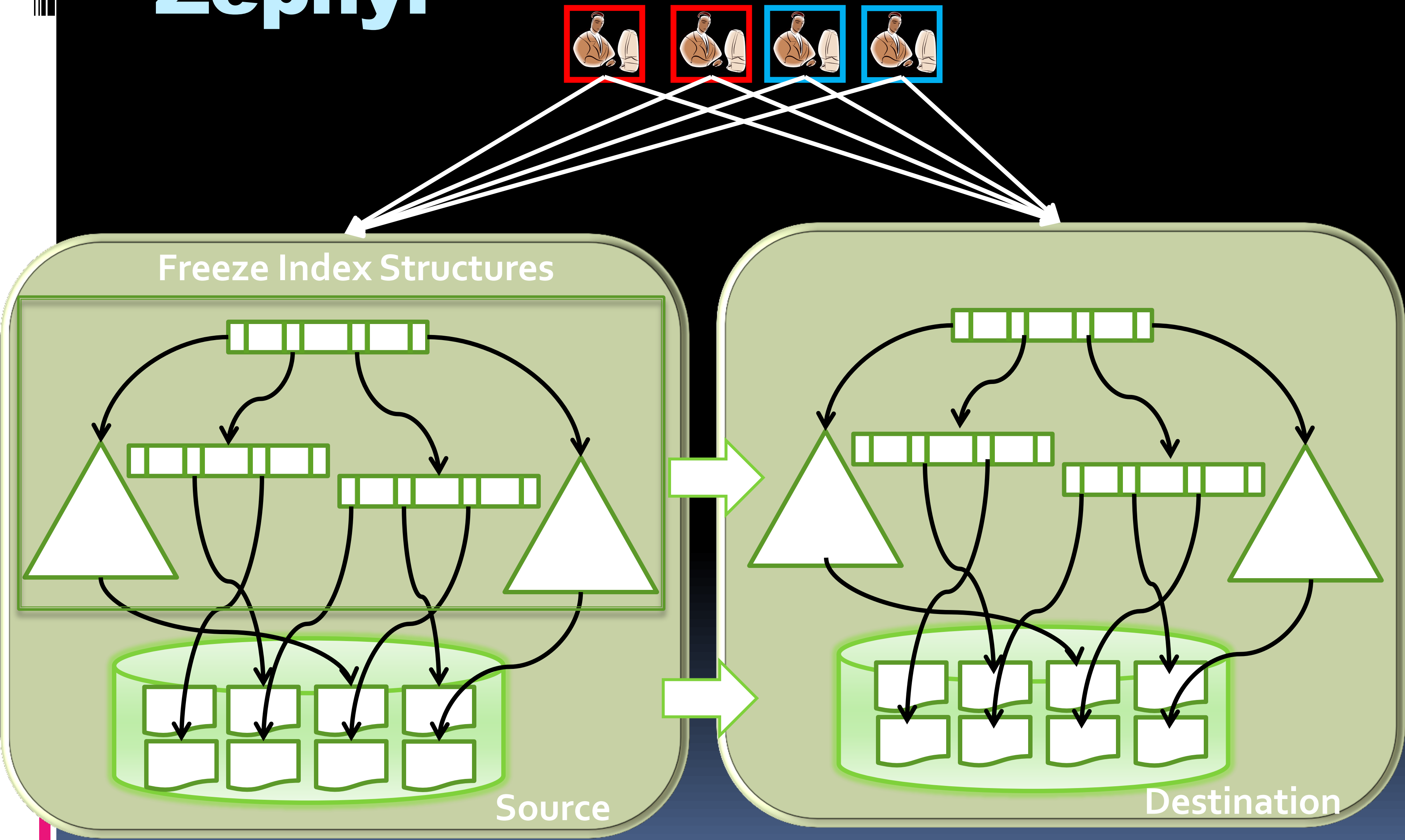
Albatross: Evaluation Summary

- Two transaction processing benchmarks
  - YCSB and TPC-C
- *Unavailability window* of **300-800ms**
  - Naïve solutions: **2-4 second** unavailability
- **No** *failed requests*
  - Naïve solutions: **hundreds** of failed requests
- **15-30%** *increase in transaction latency* after migration
  - Negligible performance impact during migration
  - Naïve solutions: **200-400%** increase in latency
- *Data transferred*: **1.3-1.6** times database cache
  - Naïve solutions: approximately the size of the cache

Shared nothing architecture
Zephyr: SIGMOD 2011

- **Reactive** state migration
  - Migrate minimal database state to the destination
  - Source and destination concurrently execute transactions
    - Synchronized DUAL mode
  - Source completes active transactions
  - Transfer ownership to the destination
  - Persistent image migrated asynchronously on demand

Zephyr



Zephyr Evaluation Summary

- Yahoo! Cloud Serving Benchmark
- **No** *unavailability window*
  - Naïve solution: **5-8 seconds**
- **50-100** *failed requests*
  - Naïve solution: ~**1000**
- **~20%** *increase* in *transaction latency* over entire workload
  - Naïve solution: **~15%** increase in latency
  - Higher latency due to on-demand remote fetch
- Data transferred: **1.02-1.04** time database size
  - Naïve solution: size of the database

Autonomic Control: DBMS Administration in the Cloud

Current State: Database Administration

Significant Operational Challenges:

- Provisioning for Peak Demand
- Resource under-utilization
- Capacity planning: too many variables
- Storage management: a massive challenge
- Software and system upgrades: extremely time-consuming
- Complex mine-field of software and hardware licensing

➔ Unproductive use of people-resources from a company's perspective

Large-scale Data-centers

- "A large distributed system is a Zoo"
  - Detecting failures and failure recovery
  - Coordination and synchronization
  - Lease Management
  - Load Management
  - System and Performance modeling

- Autonomic controllers → economies of scale

Autonomic Database Systems

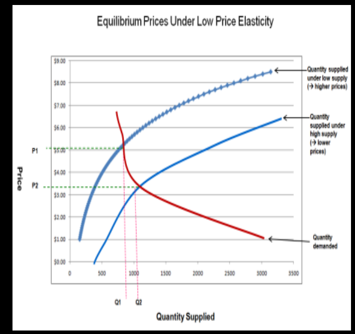
- Autonomi onomies of scale

Autonomic Control Challenges
Ongoing Work at UCSB

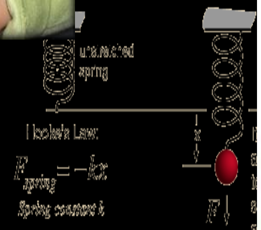
- Minimize operating cost
  - Leverage pay-per-use pricing
- Ensure Service Level Agreements
  - Maximize profits
- Static Component
  - Model behavior
  - Optimal placement to minimize number of nodes
- Dynamic Component
  - Monitor load and resource usage
  - Load balancing and elastic scaling

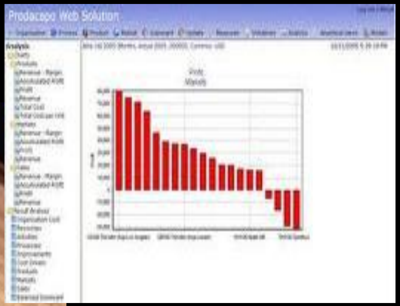
Concluding Remarks

- Data Management for Cloud Computing poses a fundamental challenges:
  - Scalability
  - **Elasticity**
  - **Autonomic Control**
  - **Payment Model Integration: future challenge**
- Cloud Computing in Emerging Markets:
  - Leveling the playing field in the context of IT
- Finally, the computing substrate will also evolve:
  - Multiple Data Centers
  - Leveraging the Network Edge (beyond content caching)



Elasticity of Utility Computing





Payment Models for Utility Computing