Database Scalability, 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

Azure Services Platform

rackspace®

Eucalyptus Systems™
salesforce.com

NetApp

Hadoop

Amazon Web Services™

NetApp

Elastra

Right Scale™

Joyent

HostedSolutions™
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

Money & Time Questions:
1. How much?
2. How Long?

Static data center

Money & Time

Questions:
1. How much?
2. How Long?
Economics of Internet Users

• Heavy penalty for under-provisioning
Economics of Cloud Computing

- Pay by use instead of provisioning for peak

Static data center vs. Data center in the cloud

- Demand
- Capacity
- Resources
- Time

Static data center

Data center in the cloud

DASFAA'2011 Keynote Address 5/20/2011
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
  - ...

DASFAA'2011 Keynote Address  5/20/2011
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

The Death of Row-Oriented RDBMS Technology. « Kevin Closson's...
Sep 13, 2007 ... 10 Responses to "The Death of Row-Oriented RDBMS Technology." Feed for this Entry Trackback Address. 1 Noons September 13, 2007 at 4:01 am ...

RDBMS: Reports of Its Death Exaggerated : Beyond Search
RDBMS: Reports of Its Death Exaggerated. February 14, 2009. Tony Bain's "Is the Relational Database Doomed?" is an interesting article. ...
arndoit.com/wordpress/2009/02/14/rdbms-reports-of-its-death-exaggerated/ - 33k - Cached - Similar pages

Web 3.0 And The Decline of the RDBMS I HaveMacWillBlog (aka Robin ... Feb 1, 2009 ... The Death of RDBMS. Kingsley has also been pursuing a theme that I have been espousing in recent times, which is that the age of the RDBMS ...
havemacwillblog.com/2009/02/01/web-30-an-evolving-debate/ - 45k - Cached - Similar pages

Why does everything suck?: The Death of the Relational Database
The construction of RDBMS is a result of NOT finding this structure to ... The "why relational databases suck" topic is pretty well beaten to death by ...
www.deathofrelationaldatabase.com/2008/02/death-of-relational-database.html - 182k - Cached - Similar pages

Oracle WTF: Death By Furniture
Death By Furniture. According to www.identifiers.org, there are two classes ... Rename the table or a column -- if you can't, then the RDBMS is Code Class ...
oracle-wf.blogspot.com/2008/10/death-by-furniture_12.html - 36k - Cached - Similar pages

Gavin defends RDBMS and Ted rebukes [kirk.blog-city.com]
Gavin defends RDBMS and Ted rebukes. « H E » email, posted Monday, 25 June 2007 ...
public void confirm_friend_request(Alice, Bob)
{
    begin_transaction();
    update_friend_list(Alice, Bob); // Tokyo
    update_friend_list(Bob, Alice); // Hong Kong
    end_transaction();
}
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);
    }
}
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>);
    }
}
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.
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**
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]
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

A single node gains ownership of all keys in a KeyGroup.

Key Group

Group Formation Phase

Keys located on different nodes

DASFAA’2011 Keynote Address  5/20/2011
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

Grouping Middleware Layer resident on top of a Key-Value Store

<table>
<thead>
<tr>
<th>Grouping Layer</th>
<th>Transaction Manager</th>
<th>Grouping Layer</th>
<th>Transaction Manager</th>
<th>Grouping Layer</th>
<th>Transaction Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key-Value Store Logic</td>
<td></td>
<td>Key-Value Store Logic</td>
<td></td>
<td>Key-Value Store Logic</td>
<td></td>
</tr>
</tbody>
</table>

Distributed Storage

G-Store
Latency for Group Operations

Average Group Operation Latency (100 Opns/100 Keys)

# of Concurrent Clients

Latency (ms)

- GStore - Clientbased
- GStore - Middleware
- HBase

5/20/2011
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
Distributed Fault-tolerant Storage

Application Clients

OTM

Application Logic
ElastraS Client

OTM

TM Master

Metadata Manager

Lease Management

Health and Load Management

OTM

Durable Writes

DB Partitions

Master Proxy

Log Manager

MM Proxy

Txn Manager

P_1, P_2, \ldots, P_n

DB Read/Write Workload

DASFAA'2011 Keynote Address 5/20/2011
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

Throughput for 10 Nodes, 1000 Warehouses

Throughput for 30 Nodes, 3000 Warehouses
Microsoft Product: SQL Azure

- 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
MIT Project: Relational Cloud
CIDR’2011

- SQL Server replaced by Open-source database engines

Workload driven database partitioning [VLDB 2010]

Workload driven tenant placement and consolidation [SIGMOD 2011]
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

DBMS
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

DBMS
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: 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
Freeze Index Structures
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

- A large distributed system is a "Zoo"

- Detecting failures and recuperating

- Coordination and synchronization

- Lease Management

- Load Management

- System and Performance modeling

- Autonomic controllers

- Economies 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 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