CloudDB: A Data Store for all Sizes in the Cloud

Hakan Hacigumus

Data Management Research

NEC Laboratories America

http://www.nec-labs.com/dm
What I will try to cover

- Historical perspective and motivation
- (Preliminary) Technical Approach
- Current Status
- Food for Thought
Why Data Management Research?

- Many Data Management Technologies and Products have been around
- Data Centers have evolved over the time
- Data Center hosting became a business
- Database Community was successful in creating technologies and business
Why Data Management (Again)?

- **Amount of Data**: Amount of business data doubles every 12-18 months.
- **New Data Types**: Relational databases only manage 10-15% of the available data.
- **New Data Sources**: Individual user via Web2.0 applications, social sides, collaboration, mobile devices, sensors, etc.
- **New Usage Patterns**: Around the clock, around the world, highly interconnected.
- **Large Number of Users**: Unprecedented increase and fluctuations.
- **New Type of Apps**: Highly integrated, extremely data intensive.

(Good Old) Database
Cloud Computing

- A paradigm shift in how and where a workload is generated and it gets executed
  - Cloud service provider – Cloud service consumer

Market Size
- Data Management Market ~$20B
- IT Cloud Service ~$42B (by 2012) (IDC)
Cloud Computing

- A paradigm shift in how and where a workload is generated and it gets executed
  - Cloud service provider – Cloud service consumer

Market Size

- Data Management Market ~$20B
- IT Cloud Service ~$42B (by 2012) (IDC)
Animoto on Amazon EC2

- A no-infrastructure startup
- Biggest piece of hardware
  - A (fancy) espresso machine!

- Rapid growth in three days, the number of users increased from 25k to 250k
- Number of servers from 50 to 3500
- Assume $500 per machine, $1.75M!
- Instead, they used Amazon EC2

Problem: It is not trivial to distribute users’ accesses to the data by just scaling out cloud computing nodes
Database-as-a-Service?

Providing Database as a Service

Hakan Hacigumus  Bala Iyer  Sharad Mehrotra

ICDE 2002!

Reaction: Cool but…

Technology

Business Model

Psychological Acceptance

Regulations
Cloud computing model may provide a platform to address new challenges

But the problem is:

- Data Management Systems were not designed and implemented with cloud computing model in mind

So the question is:

- What are the data management challenges we need to address before the full potential of cloud computing can be realized?
Need for New Solutions

- Massive scalability to handle
  - Very large amount of data
  - Very large number of diverse users/requests

- Elasticity to
  - handle varying demand
  - optimize operating costs

- Flexibility to handle different data and processing models

- Massively multi-tenant to achieve economies of scale

- More intelligent system monitoring and management
Cloud Data Management Challenges

- **Data scalability**
  - # of records / query
  - Key challenge: scalable multi-tenant hosting

- **Multi-tenancy**
  - Large Analytic apps (OLAP)
  - Small apps
  - Large Transactional apps (OLTP)

- **Query scalability**
  - # of queries / sec
  - Key challenge: scalable scan and aggregation
  - Key challenge: scalable read/write
  - Key challenge: seamless data management

- Ultimate goal: CloudDB
Buy All Sizes?

? – NO!

OLAP

OLTP

“One Size Fits All”: An Idea Whose Time Has Come and Gone

Michael Stonebraker
Buy One Size?

“One Size Fits All”: An Idea Whose Time Has Come and Gone

Michael Stonebraker

Uğur Çetintemel
Let Someone Else Do All That

Access and Management

OLAP

OLTP
Let Someone Else Do All That

- Easier integration with applications
- Leveraging very specialized database technologies
- Easier adoption by developers (dominant force for adoption of cloud!)
- Easier and more flexible deployment options in the middleware
### Wish Lists

<table>
<thead>
<tr>
<th>Clients</th>
<th>Service Provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Standard language API (e.g., SQL)</td>
<td>- Satisfying clients’ SLAs to sustain revenue</td>
</tr>
<tr>
<td>- Identifiable and verifiable Service Level Agreements</td>
<td>- Great cost efficiency via high level of automation and resource sharing to ensure profitability</td>
</tr>
<tr>
<td>- Common DBMS maintenance tasks, (e.g. backup, versioning, patching etc.)</td>
<td>- Maintaining an extendable platform for value-add services</td>
</tr>
<tr>
<td>- Availability of value-add services, such as business analytics, information sharing, collaboration etc.</td>
<td></td>
</tr>
</tbody>
</table>
## (Some) Storage Models

<table>
<thead>
<tr>
<th>Store Type</th>
<th>Main Purpose</th>
<th>Pro</th>
<th>Con</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational</td>
<td>- Transaction processing</td>
<td>- Standardization</td>
<td>- Scalability</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Higher performance on Online Transaction Processing (OLTP)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- ACID properties</td>
<td></td>
</tr>
<tr>
<td>Key/Value</td>
<td>- Scalable data storage</td>
<td>- Scalability</td>
<td>- Standardization</td>
</tr>
<tr>
<td></td>
<td>- Read/Write intensive workload</td>
<td></td>
<td>- Performance issues</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Complex query capability</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- ACID properties(?)</td>
</tr>
<tr>
<td>Column-Oriented</td>
<td>- Analytics processing</td>
<td>- Higher performance on Online Analytical Processing (OLAP)</td>
<td>- Standardization</td>
</tr>
<tr>
<td></td>
<td>- Read optimized, throughput oriented</td>
<td>- More flexible schema evolution (?)</td>
<td>- Complex query capability</td>
</tr>
</tbody>
</table>
Application Scenario

Application v1
- Personal Profile Management
  - Address
  - Phone
  - Notes
  - Contacts
  - Calendars
  - Reminders

Application v2
- Information Portal
  - Online Shopping Catalogs
  - Product Reviews
  - Subscriptions
  - ...

Very difficult migration
- Application developers (skills, time)
- Architects (redesign)
- Company (investment)

Relational Database
- Profile Data
  - User 1 Data
  - User 2 Data

Key/Value Store
- Portal Data
  - Products
  - Reviews
  - ...

External Sources

Relational Database

External Sources
Data Model Decisions

- Problem: Users are forced to make a decision on the data model based on the current needs of the applications
  - Is it possible to make the “right” decision all the time?
- Problem: The developer (client) has to re-architect their application in order to take advantage of different data models
  - How easy is it to change the architecture and the implementation?

![Diagram showing application versions and data models](image-url)
Remember Data Independence?

1968

DESCRIPTION OF A
SET-THEORETIC DATA STRUCTURE
David L. Childs

FEASIBILITY OF A SET-THEORETIC DATA STRUCTURE
A General Structure Based on a
Reconstituted Definition of Relation
David L. Childs

1970

A Relational Model of Data for
Large Shared Data Banks
E. F. Codd
IBM Research Laboratory, San Jose, California
Data Independence

- Decouple application logic from data processing
- Let them be optimized and managed independently
- Enabled decades of innovation and improvement in databases
Data Independence

- The application should not have to be aware of the physical organization of the data (and how it can be accessed)
- All it needs is a logical (declarative) specification
- CloudDB makes decisions based on application context, workload characteristics, etc.
Language?

- New Breed Databases
  - CouchDB, Project Voldemort (Dynamo), Cassandra, BigTable, Tokyo Cabinet, MangoDB, SimpleDB, ...
  - MapReduce/Hadoop
  - ...

NoSQL
Some Reminders about SQL

- By far the most widely used data access language
- It has nothing to do with
  - How the data is stored
  - How the queries are executed
  - How the transactions are handled
- Very large number of skilled programmers
- Huge amount of existing applications and tools
SQL is actually good?

- HIVE: **SQL API** op top of MapReduce

- Google BigQuery: **SQL** over data stored in non-relational databases

- ....
CloudDB - Guiding Principals

- Embrace heterogeneity
  - One size does not fit all
  - Leverage specialized technologies

- Maintain and restore “declarative” nature of data processing

- Understand and Define dimensions of scalability
System Independence?
- The middleware would be responsible for making all the decisions regarding the choice of data stores, processing the queries, and end-to-end system optimization.
- While the middleware can abstract away the underlying storage systems, it should explicitly express certain essential aspects of the system, such as consistency levels and scalability of transactions.
CloudDB Platform – Key Points

Client SLAs

- Intelligent Cloud Database Coordinator (ICDC)
- Design Optimizer
- Multi Tenancy Manager (MTM)
- Cluster Controller
- Workload Analysis
- Capacity Planner
- System Monitor Database

(External) Applications

- SQL Queries
- Distributed Query Processing

API/Language Support (JDBC, SQL)

Distributed Query Processing

- SLA Aware Dispatcher
- Scheduler
- Scheduler
- Scheduler

One Unified, Standard API

Intelligent Analysis and Decision Making

Specialized Stores for Specific Needs

CloudDB Store

- Internal Query Processing
- Auto Sharding
- Key-Value Store
- Data Migration

- Internal Query Processing
- Auto Replication
- Auto Partitioning
Our Data Management Platform

Key Research Areas

Client SLAs

Intelligent Management

(External) Applications

SQL Queries

Results

One Unified, Standard API

Workload Management

Data Stores

Intelligent Analysis and Decision Making

Specialized Stores for Specific Needs

SQL

API/Language Support (JDBC, SQL)

Intelligent Cloud Database Coordinator (ICDC)

Workload Analysis

Design

Optimizer

System Monitor

Database Cluster Controller

Client SLAs

Auto Sharding

Relational Store

Internal Query Processing

Auto Replication

Auto Partitioning

Specialized Stores

for Specific Needs

Intelligent Analysis and Decision Making

One Unified, Standard API

Specialized Stores for Specific Needs
CloudDB System Architecture -- Microsharding is a **part** of CloudDB

**Client SLAs**
- Intelligent Cloud Database Coordinator (ICDC)
- Design Optimizer
- Multi Tenancy Manager (MTM)
- Cluster Controller

**(External) Applications**
- SQL Queries
- API/Language Support (JDBC, SQL)
- Distributed Query Processor
- SLA Aware Dispatcher
- Scheduler

**Intelligent Cloud Database**
- Workload Analysis
- Capacity Planner
- System Monitor Database

**Database Cluster**
- Database Cluster Controller
- Scheduler

**Relational Store**
- Auto Replication
- Auto Partitioning

**Key-Value Store**
- Auto Sharding

**Analytics Store**
- Auto Replication
- Auto Partitioning

**CloudDB Store**
- Data Migration
SQL over Key-Value Stores

- **Microsharding** to enable SQL over key-value stores

Key challenge: limited access capabilities (only key-based put/get)
Microsharding

- Key-Value stores are good at scaling write intensive workloads

- But, they don’t leverage a large body of technologies developed in databases over the decades such as:
  - Relationships
  - Transactions
  - Advanced query functions etc.

- These are *hand-coded* by developers

- *Microsharding aims at bringing those capabilities into key-value stores in a principled way*
Key Technical Questions Addressed

- How can we map relational schemas to key-value store data models?

- How can we map relational tuples to key-value objects?

- Once we have those mappings, how can we define transaction classes that can be supported in a scalable way in key-value stores?

- What are the system implementation issues with such a middleware?
Physical design: mapping between relational data and K/V data

**Physical Design**

**Schema (+data)**

| TABLE users (id primary key ...) |
| TABLE reviews (id: primary key user_id: foreign key to orders ...) |

**SELECT** * FROM users, reviews WHERE users.id = reviews.user_id and users.id = ?

**Transformed data (KV data)**

GET \(\rightarrow\) UNNEST

**Query plan**

“Microshard” User[Review]
A microshard is
- a logical unit of data
- a principled way to shard a database into small fragments
- a unit of transactional data access
- is accessed by its key, key of root relation
Isolation Levels

- No consistency guarantee on read/write outside of a microshard

Distributed on query execution nodes

Distributed on key-value store
Scale Independence

- **Experiment Setup**
  - RUBiS benchmark (eBay type auction application)
  - Read/Write workload (transition matrix)
  - Short think time to saturate the system
  - Voldemort (Dynamo) key-value store

Message:
Ability to automatically scale to more concurrent sessions (throughput) simply by increasing the number of key-value nodes
Directions/Questions

- **Support for Specifying Relaxed Consistency**
  - Tooling to relax consistency just to the degree that there exists a feasible solution (physical design and query plans) for the specification

- **Scalable Data Organization over heterogeneous data stores**
  - Physical design over heterogeneous stores such that the service level specifications are met
  - Scalability vs. Consistency
The Cast

- NEC Labs Researchers
  - Hakan Hacigumus
  - Yun Chi
  - Wang-Pin Hsiung
  - Hojjat Jafarpour
  - Hyun J. Moon
  - Oliver Po
  - Junichi Tatemura
  - Jagan Sankaranarayanan

- Advisors/Collaborators
  - Michael Carey (U. of California, Irvine)
  - Hector Garcia-Molina (Stanford)
  - Jeff Naughton (U. of Wisconsin, Madison)
CloudDB would be…

- A **unified data management platform** that provides capabilities to **transparently** and **efficiently** support **heterogeneous workloads** by leveraging **specialized storage models** with **SLA-conscious profit optimization** in the **cloud**.
Thank You!