universität innsbruck

DBIS
DATABASES AND INFORMATION SYSTEMS

Proseminar Distributed Systems

Distributed Databases

(based on slides by Dr. Eva Zangerle)

January 12, 2021

Big Data

- Estimates of International Data Corporation (IDC)
  - 2006: 0.18 zettabytes data (1 zettabyte = 1 billion terabytes)
  - 2011: 1.8 zettabytes
  - 2012: 2.7 zettabytes
  - 2018 $\rightarrow$ 2025: 175 zettabytes (49% in publilc cloud)
- Data from
  - Internet Archive, social networks (photos, posts)
  - LHC, NYSE, MRI, sensor networks, logs, personal footprint
- Problem
  - Access speeds have not kept up
  - New data needs, can't be squeezed into Relational Database Management Systems efficiently (e.g., graph data and traversal)

Big Data (cont.)

internetlivestats.com 12.01.2021 at around 8:00

- Twitter: 9,300 tweets
- Instagram: 1,043 photos
- Tumblr: 1,820 posts
- Skype: 5„284 calls
- Internet traffic: 108,670 GB
- Google: 88,774 search queries
- Youtube: 87,682 videos
- Emails: 2,986,499 sent

Big Data (cont.)

internetlivestats.com 12.01.2021 at around 8:00

- Twitter: 9,300 tweets
- Instagram: 1,043 photos
- Tumblr: 1,820 posts
- Skype: 5„284 calls
- Internet traffic: 108,670 GB
- Google: 88,774 search queries
- Youtube: 87,682 videos
- Emails: 2,986,499 sent
- Unit: per second

NoSQL Introduction

- Aim of NoSQL
  - Develop databases to target data amounts in terabyte / petabyte scale (web 2.0 age)
  - Hard to scale relational systems with commodity hardware
- Characteristics (mostly applicable)
  - Non-relational, Schema-free
  - Scale horizontally (scale-out)
  - Open source (of course not strict: Amazon SimplDB, Neo4j)
  - Easy replication
  - Consistency model BASE (often not ACID)
  - Simple API
    - Complex queries often not possible
    - CRUD operatos (create, read, update, delete)

Motivation

- Avoid unneeded complexity
  - In some cases, no need for consistency, locking and logging
  - For Facebook, Twitter, etc. do we really need ACID?
  - Is ”eventually consistency” enough?
- High throughput
- Horizontal scalability
- Cheap commodity hardware, Virtualization
- Trade off performance for reliability

Early forms of NoSQL

- Key/value database
  - Database manager DBM (1979): stores data by use of a key in buckets (hashing)
- Document-oriented systems
  - IBM Lotus Notes (1984): stores user documents, groupware system
- Storage as Key/value pairs
  - Berkeley DB (1991)
- Column oriented systems
  - Sysbase IQ (1996)
- Distribution of these systems limited compared to RDBMS

Key/Value Stores

- Databases from teh 70ies
  - Why popular now?
- Requirements changed
  - Semi-structure data in web age
  - Data volume increased
    - Amazon, google, Twitter, Facebook (MySQL, HBase)
  - Scalability
    - Independent data
    - Hard with distributed RDMS, expertise needed
    - ACID
  - Lots of different systems

Some Systems

chordless

Redis

- **RE**mote **DI**ctionary **S**erver
  - Key/value store
  - In-memory (no data sets larger than memory)
  - Can be persisted on disc
  - Value include different data type
- Use cases
  - Caches in front of RDBMS
  - github, flickr, twitter, stackoverflow

Sample Session (CLI)


```
SET england 44
EXISTS england // returns: 1
SET france 39
KEYS * // returns: england, france
DEL france
SET france 33
RANDOMKEY // returns: france
SET water cold
EXPIRE water 10 // expires in 10 seconds
GETSET water warm // returns: cold; set to warm
```

Expire


```
SET mykey "Hello"
OK
EXPIRE mykey 10
(integer) 1
TTL mykey
(integer) 10
SET mykey "Hello World"
OK
TTL mykey
(integer) -1
```

Remaining Homeworks

- Set up and a single Redis server on Amazon EC2 (via AWS SDK)
- Answer questions about fault tolerance in general and in Redis cluster
- Set up a Redis cluster on EC2 (via AWS SDK)
- Use Redis cluster to perform MapReduce