

# 项目简介-技术组

## 数据库设计

黄玺

厦门大学 计算机科学系

2016 年 12 月 10 日

# Outline

- 1 The Big Picture of Data Management
  - Why database management systems?
  - Why MongoDB?
- 2 Designing Database for CPP
  - Methodology
  - Facts and Challenges
  - Scale-out: From Standalone to Distributed

# Outline

- 1 The Big Picture of Data Management
  - Why database management systems?
  - Why MongoDB?
- 2 Designing Database for CPP
  - Methodology
  - Facts and Challenges
  - Scale-out: From Standalone to Distributed

# Data, DB and DBMS

**Data** is a set of values of qualitative or quantitative variables.

**Database** is an organized collection of data containing its schemas, tables, queries, reports, views, and other objects.

**DBMS** database management system, which is a computer software application that interacts with the user, other applications, and the database itself and allows the definition, creation, querying, update, and administration of databases.

\*Sometimes a DBMS is loosely referred to as a 'database'.

# Databases As a Service

表: Table 1-1 Comparing managing data with files and DBs

Spec	Files	Databases
Storage	On disk	Managed by database
Data status	Raw	Organized
Availability	Load on use	Stable
Managebility	Messy	Ready for change, backup and administrating
Lots of small files	Tricky	Taken care by the impl of storage engine
Difficulty	Straight but troublesome	Need to learn and also troublesome...for maintainers : )

# Outline

- 1 The Big Picture of Data Management
  - Why database management systems?
  - Why MongoDB?
- 2 Designing Database for CPP
  - Methodology
  - Facts and Challenges
  - Scale-out: From Standalone to Distributed

# Dealing with volume, velocity and variety

Facts:

- Data is becoming semi-structured and unstructured, e.g. business transaction and web application data
- The scale of data sets is also growing rapidly
- Need more powerful systems to support real-time applications

MongoDB provides:

- Storage based on BSON (the very basis of NoSQL)
- Fast ops
- Scalability



Fig. 1-1 MongoDB Logo

# Outline

- 1 The Big Picture of Data Management
  - Why database management systems?
  - Why MongoDB?
- 2 Designing Database for CPP
  - Methodology
  - Facts and Challenges
  - Scale-out: From Standalone to Distributed



# Methodology

- Deploy MongoDB instances
- Design data schemas
- Run benchmark to check the performance
- Merge with Scrape system
- Maintain & iterate

# Quick Demo

# Outline

- 1 The Big Picture of Data Management
  - Why database management systems?
  - Why MongoDB?
- 2 Designing Database for CPP
  - Methodology
  - **Facts and Challenges**
  - Scale-out: From Standalone to Distributed

# Facts and challenges

- The 1G mem of cheap servers cannot meet the need of MongoDB
- The disk size of our servers is also limited
- Top Level Design: How can we make things easy to use and iterate?
  - Monitoring
  - Automating
  - Transparency
  - Decoupling

# Outline

- 1 The Big Picture of Data Management
  - Why database management systems?
  - Why MongoDB?
- 2 Designing Database for CPP
  - Methodology
  - Facts and Challenges
  - Scale-out: From Standalone to Distributed

# Solution to hardware limitation: Scale-out

The following figures and texts are adapted from assets of Mining Massive Datasets by Leskovec, Rajaraman, and Ullman, Stanford University, 2016.  
Non-public and non-commercial use only.

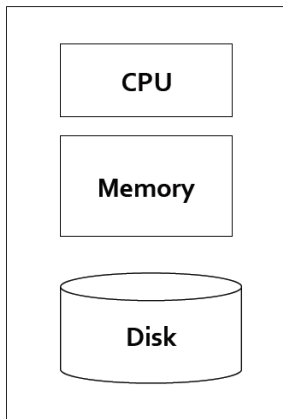


Fig. 2-1 Single Node Architecture

# Solution to hardware limitation: Scale-out

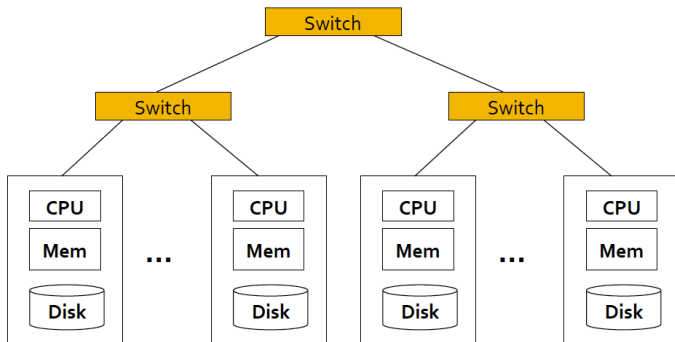


Fig. 2-2 Cluster Architecture

# Cluster Computing Challenges

- Network bottleneck
- Distributed programming
- Node Failures
  - Hardware breakdown
  - Software malfunction
  - Unstable networking

## Problems:

- How to store data persistently and keep it available if nodes can fail?
- How to deal with node failures during a long-running computation?
- How to hide most of the complexity of the distributed system?

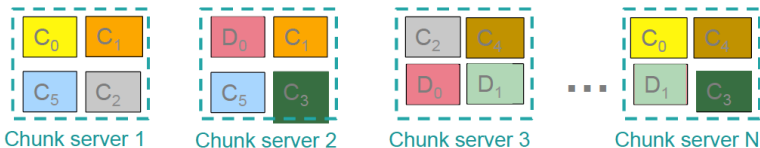


# Redundant Storage Infrastructure

- Distributed File System
  - Provides global file namespace, redundancy and availability
  - E.g. Google GFS, Hadoop HDFS
- Typical usage pattern
  - Huge files (hundreds of GB to TB and even PB)
  - Data is rarely updated
  - Reads and appends are common

# Distributed File System

- Data kept in “chunks” spread across machines
- Each chunk **replicated** on different machines
  - Ensures persistence and availability



Chunk servers also serve as compute servers

Bring computation to data!

# Distributed File System

## ■ **Chunk servers**

- File is split into contiguous chunks (16-64MB)
- Each chunk replicated (usually 2x or 3x)
- Try to keep replicas in different racks

## ■ **Master node**

- a.k.a. Name Node in Hadoop's HDFS
- Stores metadata about where files are stored
- Might be replicated

## ■ **Client library for file access**

- Talks to master to find chunk servers
- Connects directly to chunk servers to access data

# Distributed MongoDB

Problems:

- How to store data persistently and keep it available if nodes can fail?
- How to deal with node failures during a long-running computation?
- How to hide most of the complexity of the distributed system?

Todo:

- Auto-sharding
  - Replica
  - Balancing
  - Primary election
  - e.t.c.

DONE : )

# Distributed MongoDB

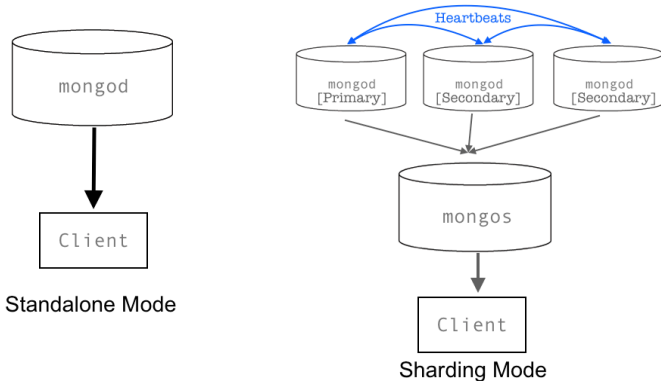


Fig. 2-3 Standalone and Sharding MongoDB Architecture

Any questions?

