In this lecture, we will discuss…

- ✧ Rationale behind NoSQL
- ✧ Scaling Issues in RDBMS
- ✧ NoSQL: What is it?

Why RDBMS

- Relational Databases – popular and commonly used
- Initially designed for non distributed
- Low Cost RDBMS alternatives (PostgreSQL, MySQL, SQLLite)
- Very Transactional - across tables and commands, and can even be transactional across distributed resources (XA) -- at a cost
- Supports Joins -- across multiple tables allowing for normalized forms of data to be stored once

Why NoSQL

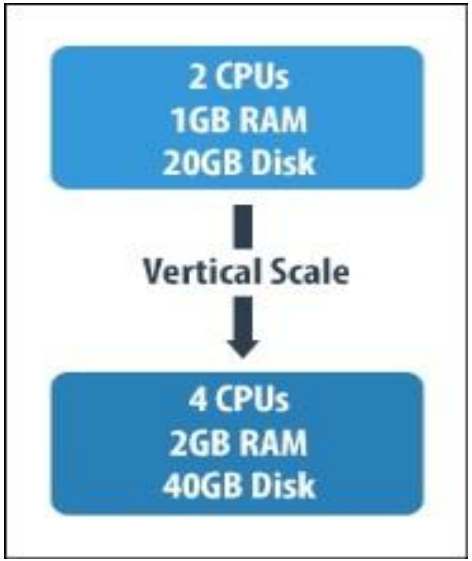
- Explosion in data
- Object/Relational Impedance mismatch
  - Objects are constantly being moved in/out of tables/rows
- RDBMS normalization and joins are powerful, but add up in cost
  - Complex objects stored across many tables and rows can be expensive to handle

Why NoSQL

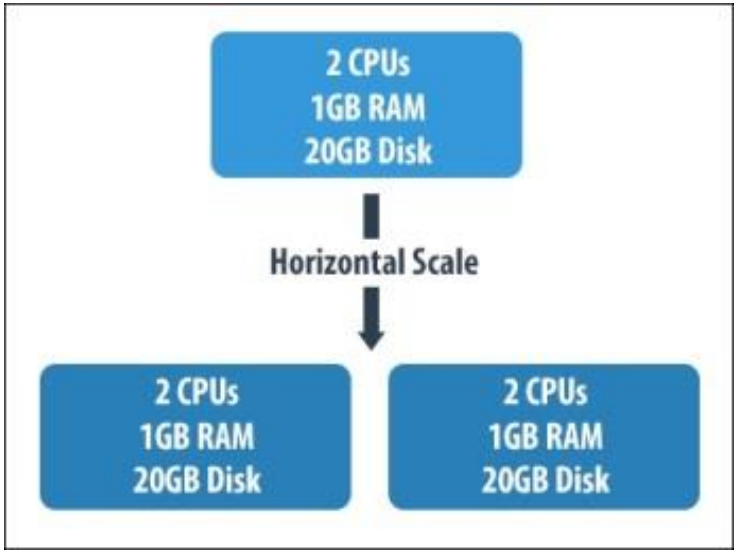
- “Big” data handling with better performance
- Supports unstructured data
  - Unique data type extensions can be easily integrated into existing collections
- Operational issues (scale, performance and availability)

Scaling Out

Vertical Scaling



Horizontal Scaling



What is NoSQL

- Stands for “Not Only SQL”
- No Fixed Schema
- Non-relational data storage systems

Summary

- NoSQL – very popular and major companies especially social networking sites such as Twitter, Facebook, LinkedIn, and Digg use NoSQL DB
- Excellent performance and stability, fast and scalable and fairly simple model
- Supports unstructured format, which makes it very agile
- NoSQL is mostly gained when access patterns to complex objects are understood and modeled correctly up front

What’s Next?

Categories of NoSQL