Scalability Considerations

Yogesh Deshpande

GS Lab

Agenda

- Scalability
  - Overview
  - Relevance Of Java Platform
  - Vertical Scaling
  - Horizontal Scaling
  - Cloud ==Scalability ? true: false
- Horizontal Scaling: Case Study
  - Problem Statement
  - Basic Principles
  - Architecture Highlights
- References

Scalability: Overview

- Allowing more users/mediums to use your application with consistent performance
- System should be capable of handling increasing load dynamically
- May come with cost of additional processing overhead
- Its more about architecture and not about the language/frameworks

Relevance Of Java Platform

- Components/libraries/tools are widely available with choice
- Famous GC pause?
- Specification like OSGI
- Vertical scaling of JVM
- Cloud Platforms
- New languages like Scala, Clojure on JVM

Vertical Scaling

- Adding resource within the same logical unit to increase capacity
  - Add CPU/Storage/memory
- Some Solutions
  - AzulSystem's Zinc: JVM with huge heap size
  - Terracotta's BigMemory: JVM-level clustering

Horizontal Scaling

- Adding multiple logical units of resources and making them work as a single unit
  - Think everything in terms of services
  - Most clustering solutions, distributed file systems, load-balancers help you with horizontal scalability
  - More processing overhead

Cloud == Scalability ? true:false

- Not 100% true or false
- I-a-a-S: If system is not scaling on data center it will not scale on Cloud
- P-a-a-S: Has some meat
  - Google App Engine
  - Amazon Elastic Beanstalk
  - Oracle cloud and many more

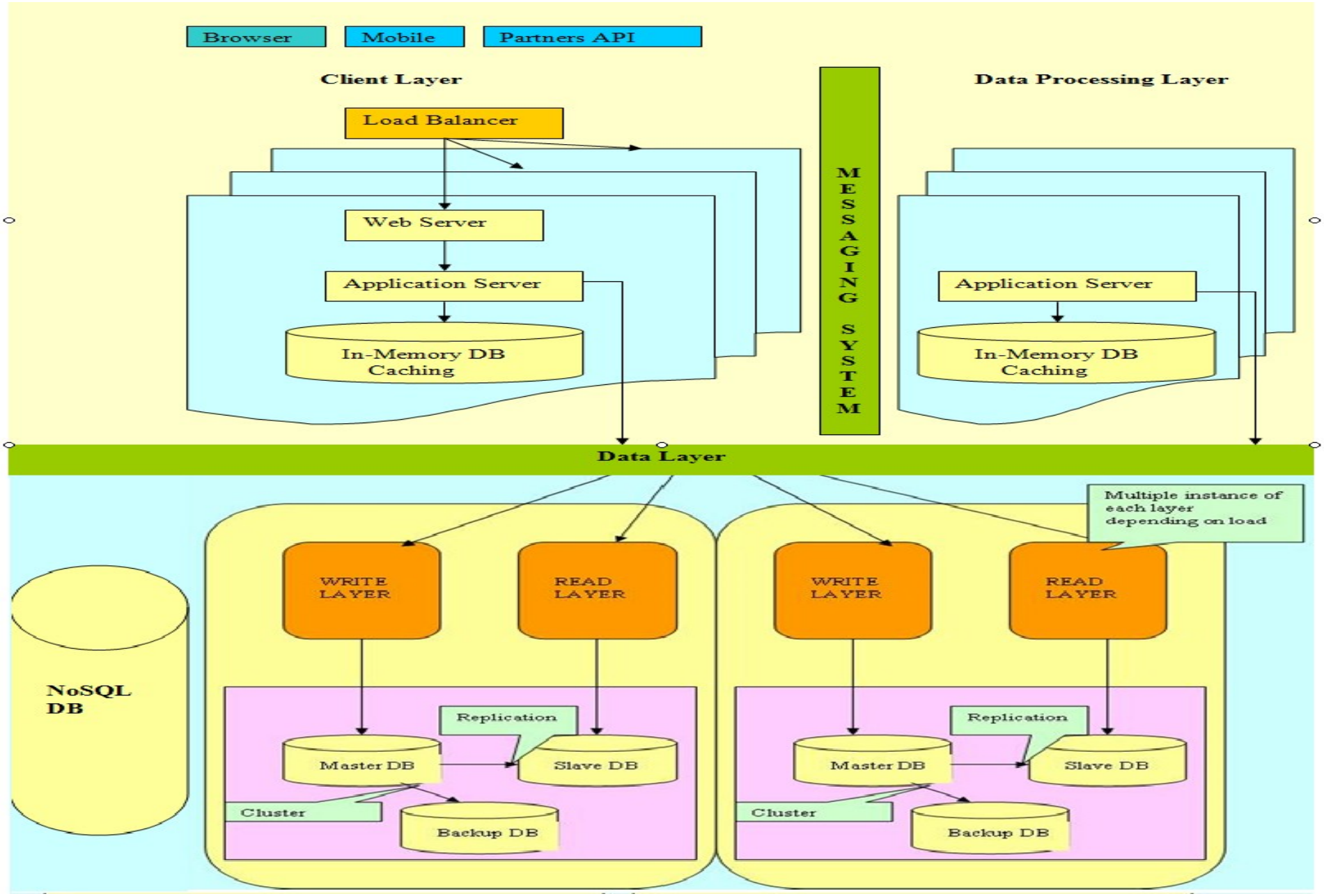
Horizontal Scaling: Case Study

- This particular product required to support one million users and processing packets coming from each user's device at frequent time interval
- Our aim was to build horizontal scalable system using java/j2ee technologies

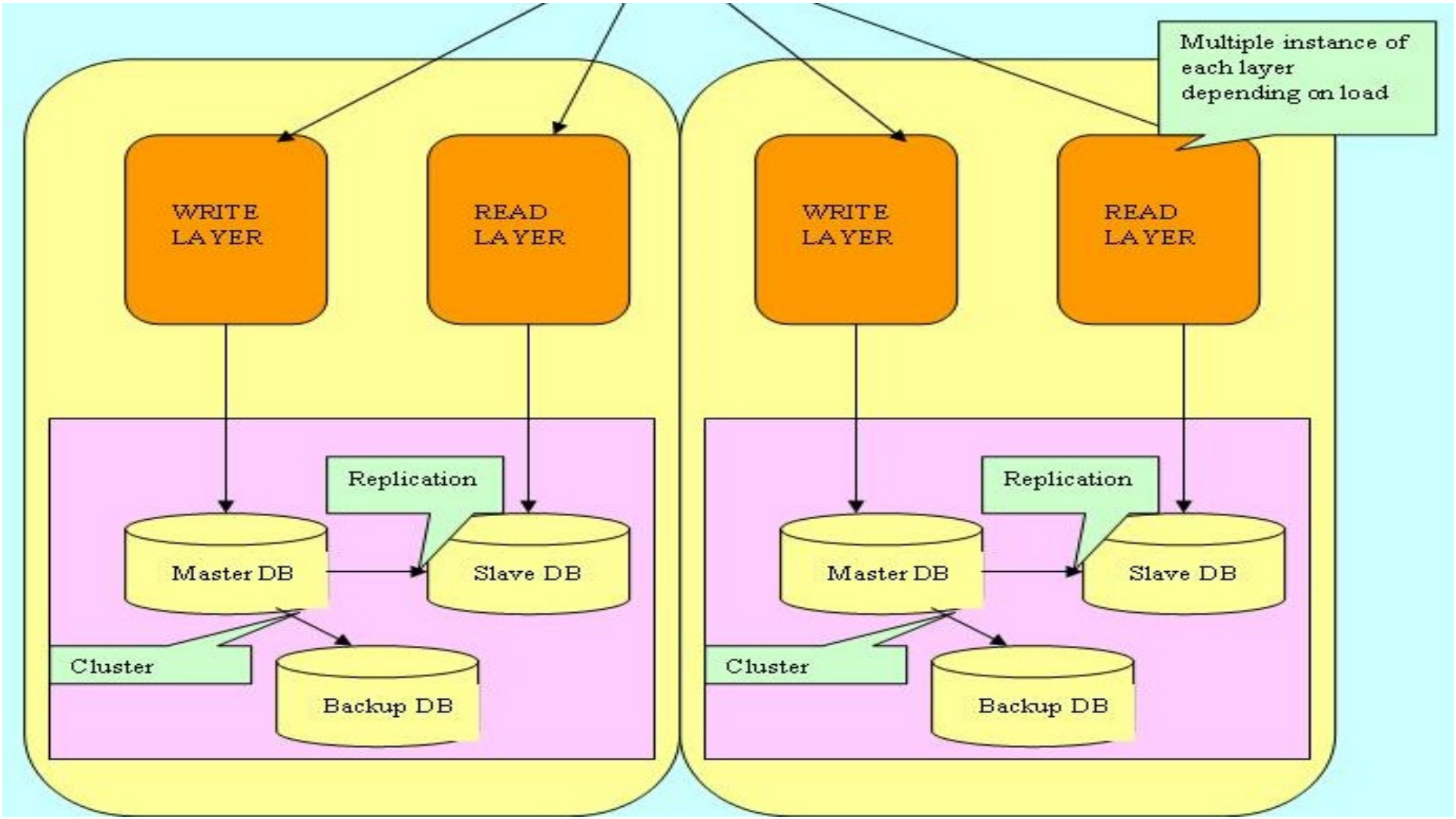
Horizontal Scaling: Principles

- Statelessness:
  - Each service should be stateless
- Caching:
  - Use as much as you can
- Parallelizing:
  - Distribute workload
- Sharding:
  - Divide and Rule

Architecture Diagram of Case Study



Magnified View of Data Layer



Architecture Highlights of Case Study

- RESTful Design: CRUD operations as Web service calls
  - **Technology**: Restlet.(Spring3.0 has it now)
- Stateless UI: No Session on Server
  - **Technology**: Struts2.0
- READ and WRITE layer: HTTP POST/PUT to WRITE and GET to READ
  - **Technology**: Hibernate for WRITE and Spring-jdbc for read.

Architecture Highlights

- MySQL: Master/Slave replication
- Workload Distribution:Scheduled job to run at multiple nodes
  - Simple logic to divide

```
for (Long userId : allUserIds) {
    if ((userId % noOfNodes) == nodeId) {
            processUserData(userId);
     }
 }
```

  - Technology: Quartz library

Failover Consideration

What can fail

- Application Server
- Database Server
- Scheduled Jobs

References

- Various articles on
  - http://highscalability.com
  - http://www.oracle.com
  - http://www.ibm.com

Thanks

For further questions contact

yogesh@gslab.com