WHITE PAPER

# Top 10 Tips to Scale a Cloud App

# TABLE OF CONTENTS


Introduction 03

Top 10 Tips 04

# INTRODUCTION

When developing a comprehensive cloud application, whether a web portal or mobile apps, critical constraints are only realized once usage starts to go up. A common myth is that one can “throw hardware” at the problems of scale. When things are not working well even after adding physical and virtual resources, a technology team may arrive at the costly realization that the app itself was not built in the right way. This can be a potentially fatal error for companies attempting to exploit a high risk opportunity as is the nature of technology innovation.

Avoid investing in “throw-away” work even if your app is a Minimum Viable Product or a Proof of Concept. Applying strategic foresight from the outset drives more value for your business that can be reinvested into further development.

Here are the top 10 tips to equip your technology product with the foresight of elastic stability and scalability.

# TOP 10 TIPS

## Scaling-Up

Scaling-out means adding low cost virtual machines in a cluster that software can exploit. Scaling-up is when you utilize more or better hardware in an attempt to solve your scalability problems. When designing for the cloud era it is better to architect a distributed system that can scale-out in an elastic manner. This type of architecture is able to “breath-out” during peak periods by spinning up more Virtual Machines, and “breath-in” in during off peak periods.

## Distributed Architecture

The Service-Oriented Architecture (SOA) design paradigm offers a smart way to build cloud apps with the concept of a lightweight messaging bus that links various services together. SOA provides high performance and a maintainable way of connecting consumers to providers. Various SOA architectures can provide support for workflow (ie. stateful) and non-workflow (ie. stateless) interactions. Scaling-Out vs.

## Sharding

Sharding means partitioning a system according to various criteria. The most common use of sharing is scaling the database by users. Sharding can be used by tenants in a multi- tenancy system according to specific functions or domains. Sharding requires thoughtful information architecture and is best done by asking the right questions upfront.

## Load-Balancing

This well known architecting concept involves directing web traffic to the right web server based on the load characteristics of the server. When developing RESTful architecture, care must be taken to also develop stateless servers that avoid relying on a specific web server behind the load-balancer to complete the transaction in progress.

## System Caching

Content caching is a sure way to improve the responsiveness of user-experience by serving content fast. Users do not expect to stare at a spinning wheel for more than 3 to 5 seconds. Caching static content by utilizing a Content Delivery Network (CDN) is one way to improve responsiveness. In addition to a CDN for static data, frequently accessed or more relevant data can be retrieved faster if memchached is used. Various browsers and apps offer additional ways of caching data. Integrating all of these modes of caching is crucial in making the UI responsive. In certain apps, in-memory databases like Redis can speed up the processing for requests and searches.

## UI Caching

Caching on the UI side should be done with careful attention. iOS or Android apps should avoid caching data which is obsolete or invalid. Some mobile apps have fall-back provisions to show previously cached content when the app does not have network access. When showing cached content, the UX design should ensure that the user is informed if they are viewing older content.

## SQL vs NoSQL

When data is stored in a database, it may go through an Object-to-Relational Mapping (ORM) layer such as Spring Hibernate. A careful examination of the problem domain and representative data types will appropriate selection between SQL and NoSQL. If the data being stored is more relational in nature, then the SQL database is a natural choice. If the data is more document oriented, the use of a MongoDB type NoSQL database will make the app more responsive and scalable.

## Search

When providing search functionality within the app using options like Solr, ElasticSearch and others, it is vital to analyze the specific needs prior to implementation. For example, if the data being searched is not indexed properly through the right tokenizer, the results that are returned may not be what users expect.

## Instrumentation

One of the cheapest ways to ensure that you build a system to scale and perform well is by rigorously instrumenting the code. Developers may skip the instrumentation step only to realize later on that they have no idea why the system is performing so poorly. Developing a system of instrumentation standards and following through on profiling the code for performance is an effective way to identify and fix bottlenecks compared to the “spray and pray” technique of over provisioning hardware or virtual resources. When a mobile component is present, instrumenting only the web services is not sufficient. The mobile component should also be heavily instrumented to trace performance bottlenecks from end-to-end.

## Automation

Once you have enabled instrumentation, logging, and monitoring for your cloud app, it is important to consider investing in DevOps automation. This allows you to leverage automatic recovery from system outages, automatically spin up new Virtual Machines when you exceed trigger load and response time characteristics, and backup your database for data retention purposes. Advanced DevOps automation scripts, written in Puppet, Chef or Python keeps your app humming efficiently, even allowing you to automatically recover incase of an unexpected outage inthe middle of the night.

## ZYMR IS A SILICON VALLEY BASED CLOUD CONSULTING AND FULL-STACK SOFTWARE DEVELOPMENT SERVICES COMPANY

We work alongside our customers as an integrated software product engineering team to deliver disruptive solutions within challenging timelines. Whether you need to migrate legacy services to the cloud, sharpen your current cloud strategy, or architect new solutions spanning multiple environments, we ensure that your initiatives realize maximum value.

## DESIGN THINKING MATTERS.
## DOMAIN EXPERTISE MATTERS.

**REACH US AT:**

**ZYMR, INC.**
WWW.ZYMR.COM
SAN JOSE | AHMEDABAD | PUNE | BENGALURU

ZYMR