

# Server-Side Design Principles for Scalable Internet Systems

A Paper By Colleen Roe and Sergio Gonik

*Reviewed By David Deschenes*

*October 8, 2002*

# What is scalability?

Ability of a software system to respond to ramping demand. Computational resources can become scarce as the number of concurrent users of a system increases. A system designed with scalability in mind should allow for the addition of resources so that capacity can meet demand.

# How do you achieve scalability?

- » Focus on the software architecture
- » Optimize use of the following resources
  - CPU
  - Memory
  - File-system Bandwidth
  - Network Bandwidth

# Software Architecture

- » Emphasize functional decomposition
- » Resulting modules should
  - Have encapsulated responsibilities
  - Have clearly defined interfaces
  - Use appropriate communication protocols
  - Be independently scalable

# Optimization Problems

- » Data proliferation
  - Increases network bandwidth usage
- » Desire for real-time synchronization
  - Can increase network load
  - Can reduce interactive responsiveness
- » Need for centralized resources
- » Many others...

# Optimization Techniques

## » Parsimony

- Just-enough Data Distribution
- Resource Pooling

## » Asynchrony

- Queuing
- Near Real-Time Synchronization

## » Concurrency

# Just-Enough Data Distribution

- » Simplify the data that is communicated between systems
  - Decreases CPU utilization required to prepare the data for transmission
  - Reduces network bandwidth needed to transmit

# Resource Pooling

- » Effective way of sharing memory
  - Object graph
- » Effective way of reusing overhead
  - Object instantiation
  - Remote object activation
  - Database connections

# Queuing

- » Enhances scalability by spreading system usage across time
- » Can make the user interface more responsive
- » May permit work prioritization
- » Tasks may be run in batches during non-peak operation

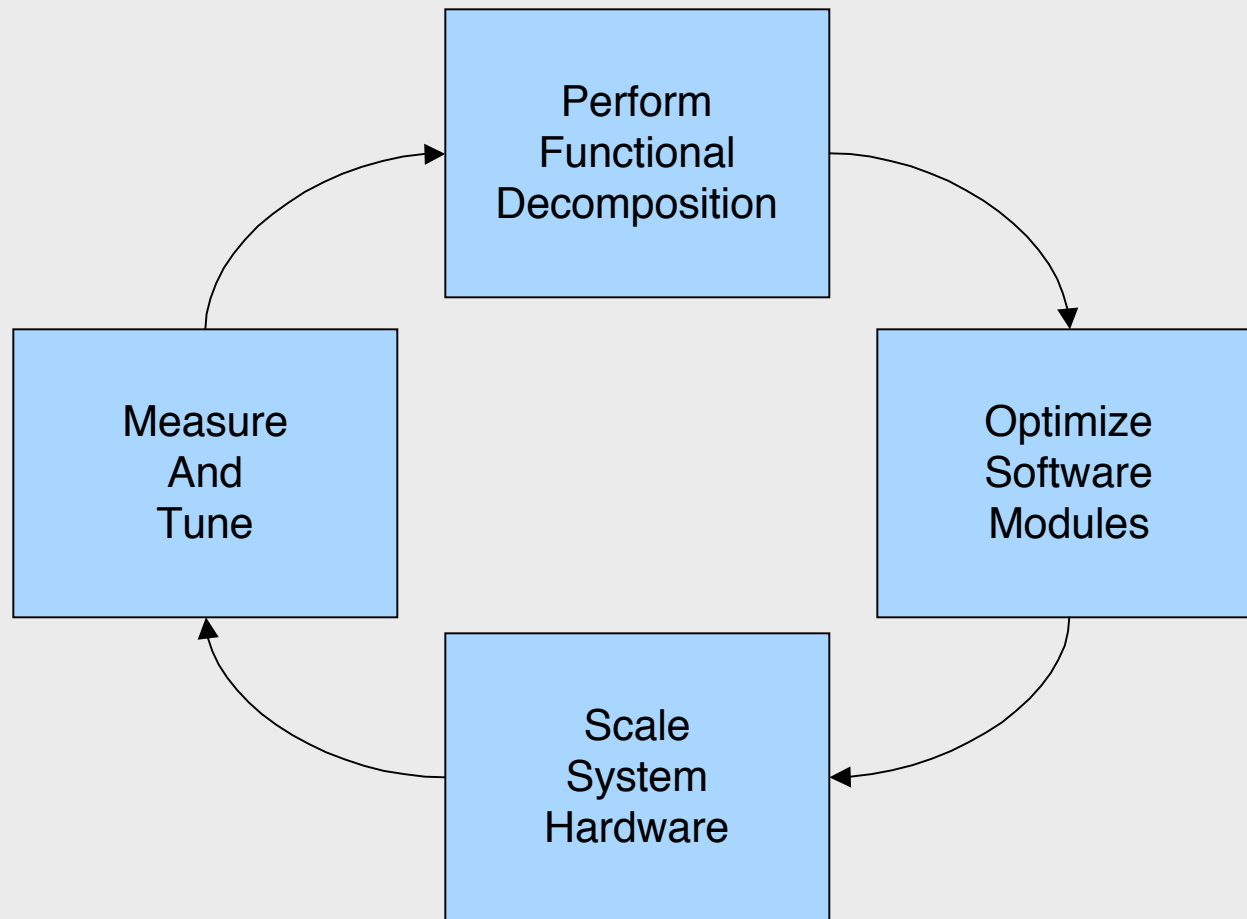
# Near Real-Time Synchronization

- » Assists scalability by spreading system transaction load across time
- » Improves responsiveness of the user interface
- » May reduce network load

# Concurrency

- » Aids scalability by ensuring that the maximum possible work is active at all times
- » Activities are spread across hardware, processes and threads
- » Can exploit modern symmetric multiprocessors

# Putting It All Together



# Questions

- » Does any of this apply to a search engine?
- » If so, what design principles (or patterns) do you see?