Architecting Scalable Enterprise Web Applications

2023

This presentation is based on

Chapter 1: "*Architecting Scalable Enterprise Web Applications*"

Book: "Architecting High Performing, Scalable and Available Enterprise Web Applications" by: Shailesh Kumar Shivakumar, 2015

outlook

1. Introduction
2. Scalability layers
3. Scalability patterns and best practices
4. Principles of Scalability
   - Scalability through fault tolerance and failover
   - Distributed computing
   - Service scalability
   - Database scalability
   - Storage scalability
   - Virtualization
   - Cloud Computing

1. Introduction

Scalable

- *Scalability is the capability of the enterprise application and its ecosystem (IT Infra) components to handle increased workload and demand without compromising its overall efficiency.*
  - Application
  - Ecosystem (server, hardware, network, base software)
  - Workload
    - # HTTP request per time period
    - # queries to a database
  - Efficiency
    - Good response times
    - SLA
    - TPS

Little's Law

handling capacity of the system

$L = \lambda \times W$

Where:

L = Average number of requests in a stable system

λ = Average request arrival rate

W = Average time to service the request

Little's Law

Example:

100 web request per second

0.5 s to serve each request

L = 100 * 0.5 = 50 request concurrently

Ejercicio

Según la ley de Little, la capacidad de un sistema está dado por: $L = \lambda \times W$

L -> promedio de requerimientos en un sistema estable

λ -> promedio de llegadas

W -> tiempo de atención por servicio

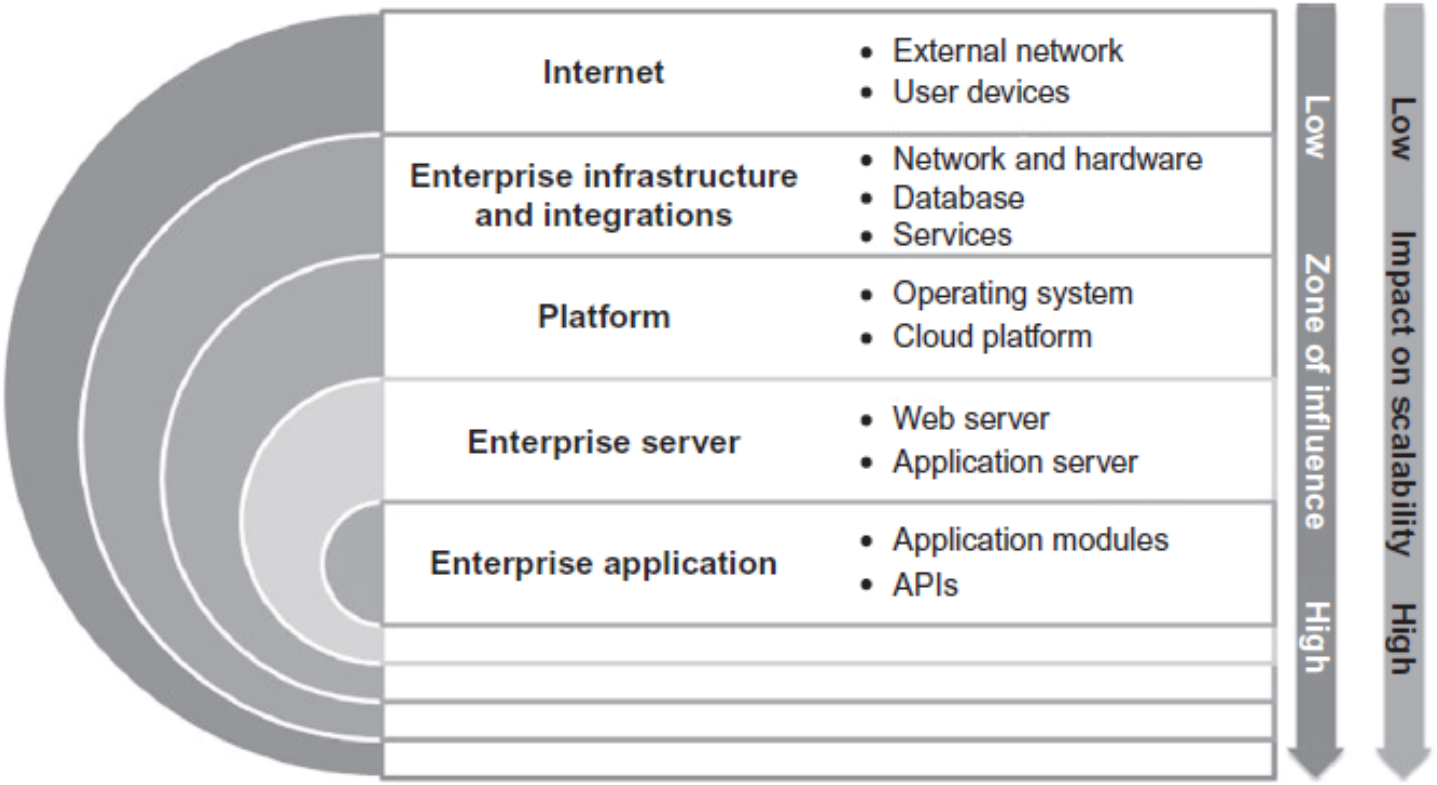
Si tenemos un sistema con L = 50 y un tiempo de servicio de 2 segundos, con un promedio de llegada de 25 peticiones por segundo.

Si se reduce el tiempo de servicio a 1 segundo, cuál de las siguientes interpretaciones en válida.

a. El sistema debe reducir el promedio de llegadas para mantener el L.
b. El sistema puede duplicar el número de llegadas promedio para mantener el mismo L.
c. El sistema reduce el valor L de estabilidad a 25 interpretándose como que se requiere menos recursos para lograrlo.
d. El sistema requiere más recursos para el nuevo valor de L.

2. Scalability layers

Scalability Layers



Dimensions of scalability

- Load scalability (size scalability)
- Functionality scalability
- Integration scalability
- Geographic scalability
- Administrative scalability

Scaling techniques

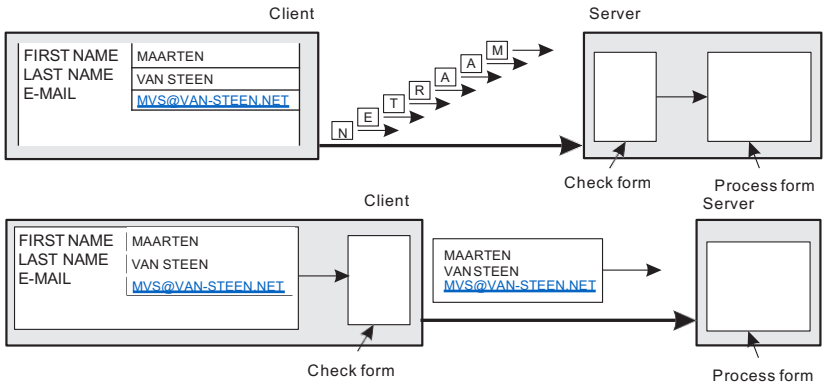
- Partition data and computations across multiple machines
- Replication and caching: Make copies of data available at different machines

Scaling techniques

- Make use of asynchronous communication
- Have separate handler for incoming response
- Problem: not every application fits this model

Scaling techniques

Facilitate solution by moving computations to client



Scaling techniques

Partition data and computations across multiple machines

- Move computations to clients (Java applets, FEF-vue-angular-react-etc)
- Decentralized naming services (DNS)
- Decentralized information systems (WWW)

Scaling techniques

Replication and caching: Make copies of data available at different machines

- Replicated file servers and databases
- Mirrored Web sites
- Web caches (in browsers and proxies)
- File caching (at server and client)

What is the problem with replication?

3. Scalability patterns and best practices

Scalability patterns - 1

Scalability Pattern	Impact on Scalability
Distributed computing pattern	Distributes the requests among all computing nodes and helps each of the individual systems and components scale better to provided optimized load scalability
Parallel Computing pattern	Helps in distributing the workload through parallel execution thereby utilizing the advantages of distributed computing
Event Driven Architecture pattern	Provides asynchronous communication which scales each of the interacting components, layers and systems
Data Push and Data Pull Model	Facilitates optimal transfer of data thereby reducing the load on individual systems and communication networks which helps in scaling

Scalability patterns - 1

Scalability Pattern	Impact on Scalability
Distributed computing pattern	Distributes the requests among all computing nodes and helps each of the individual systems and components scale better to provided optimized load scalability
Parallel Computing pattern	Helps in distributing the workload through parallel execution thereby utilizing the advantages of distributed computing
Event Driven Architecture pattern	Provides asynchronous communication which scales each of the interacting components, layers and systems
Data Push and Data Pull Model	Facilitates optimal transfer of data thereby reducing the load on individual systems and communication networks which helps in scaling

Scalability patterns - 2

Scalability Pattern	Impact on Scalability
Service Oriented Architecture	Makes each of the layers loosely coupled and enables reusability and asynchronous communication. This helps all participating layers to scale well by providing load scalability, integration scalability and functionality scalability
Workload/demand distribution	Optimal distribution of demand help in leveraging the available resources which helps in load scalability of the overall system
Database scalability patterns	Various patterns at database layer help in optimizing data retrieval and persistence
Enterprise Portals Pattern	Provides optimized integration and information aggregation which provides integration scalability
Messaging pattern	Provides asynchronous communication and helps in integration scalability

Distributed computing pattern

- Ex: Divide and conquer
- Ex: Map Reduce

Parallel Computing pattern

- This can be achieve through multiple means:
  - Publisher and Subscriber (pub-sub) model
  - Join pattern
  - Asynchronous execution
    - Ex: Ajax - JQueryJQuery

Event Driven Architecture pattern

- state changes of a system generates events which are notified to all event listeners.
- propagated in asynchronous manner.
- loose coupling
- Event oriented architecture
- Key components:
  - Events
  - Messages
  - Asynchronous communication
  - Middleware -> Enterprise Service Bus

Data Push and Data Pull Pattern

- Pull – Client demands data from server. Ex: RSS
- Push – Server sends the data to client.
- Which is better?
  - Pull scales better.

Service Oriented Architecture Pattern

- Exposes functions as a service
- Stateless
- Nothing sharing between services
- Key tech:
  - RESTful: HTTP, URI, JSON
  - SOAP, UDDI, WSDL, XML

Workload/demand distribution

- Load balancer
- Minimize load sources

Database scalability patterns

- Sharding
- Caching
- NoSQL
- Clustered
- Replication
- Data mirroring

Enterprise Portals Pattern

- Portals are self-contained and independent applications

Messaging Pattern

- asynchronous message passing
- Messaging provides point-to-point message queues model and publish-subscribe model
- Message-oriented-middleware(MoM)

Scalability best practices

Stateless session

- A transation or request independent from another.
- Web pages with cookies.
- Transport session info on URL or each request
- Use REST
- Reduce session stickness, to help load balancers
  - Support horizontal scaling
- Spit the application by services located on different servers (functionality scaling)
- Promotes use of: load balancers, caching, reduce session state sync for app servers.

Lightweight design

- Web pages size (in bytes) & time load (in seconds)
- Minimize static assests (images, js, css)
- Use AJAX to update partial web page
- Use REST rather than SOAP or API calls.
- Use JSON rather than XML
- Those reduce CPU processing and data tranfered by network.

On-demand data loading

- On-demand web pagination.
- Dinamic pre-load of list-box (ex: 1st select “Colombia”, then 2nd load Cities). AJAX can help.
- Load images and page content when the user goes to this section.
- Those reduce CPU processing and data tranfered by network.

Resource pooling

- database connection pool
- thread pool
- service pool

Using Proven technologies

- Tech that has been time-tested in huge implementations (documentation and benchmarking)
- Buy vs build
- market analysis of available open-source and commercial alternatives
- On open-source: community
- Ex: Eafit Interactive (in-house dev) vs Moodle (open-source)

Optimal Enterprise integrations

- Service-based integration
- Asynchronous integration
- Lightweight and on-demand data transfer
- Use EIA (Enterprise Integration Application)
- Use ESB (Enterprise Service Bus)

Scalability by design

- Static analysis (heap size, CPU cycles, etc)
- Proactive analysis tools categories:
  - Static analysis using code metrics
    - Number of Calls, Cumulative time, Method Time, Average Method Time.

Latency and throughput optimization

- Latency is the time taken for the first response from the server
- Throughput is the total number of transactions the system can process in a given time interval.
- Best practices:
  - Minimize data transfer over wire
  - On-demand data load
  - Asynchronous data loading (ex: Ajax)
  - Co-location
  - Cache
  - Compression

Early runtime application analysis

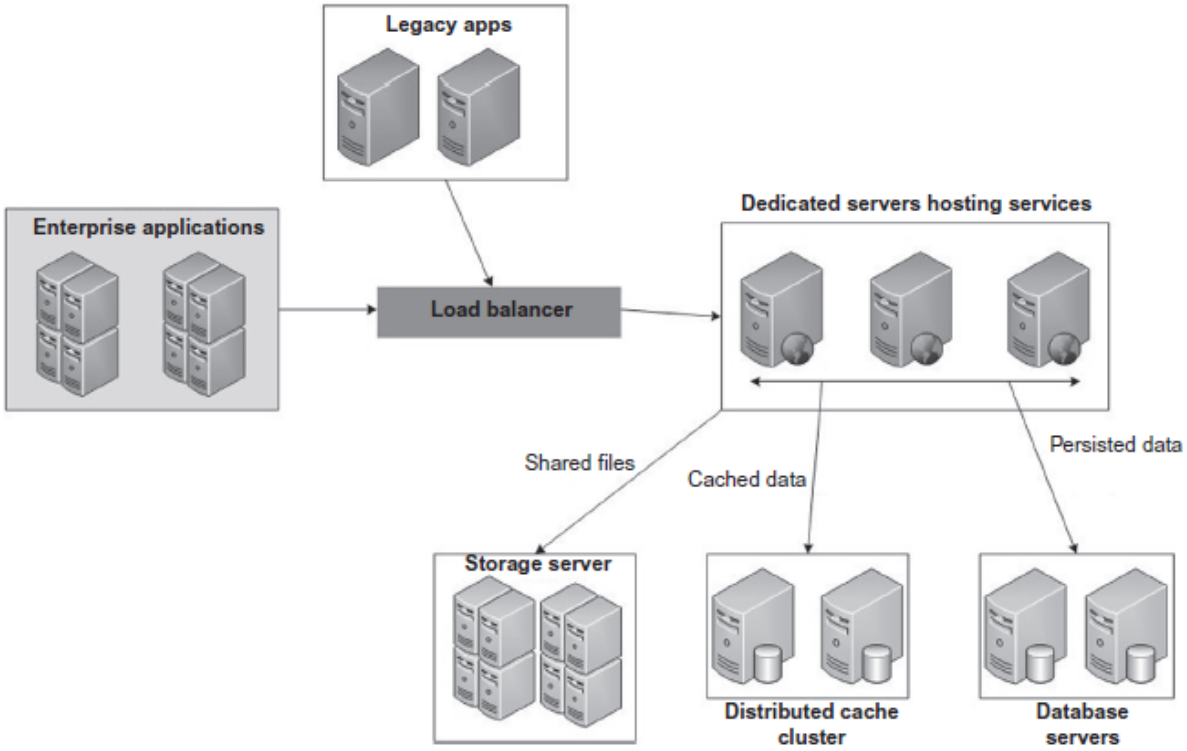
- Performances monitored:
  - CPU Usage
  - Memory Usage
  - Network usage.
  - File I/O
  - Database I/O and Query.

Avoid blocked waits

- During event handling and for services/resource calls, avoid blocked waiting for response or acknowledgment .
- Best practices:
  - Asynchronous invocation
  - Message queue

Architecting scalable services infrastructure

Clustered server configuration of web services

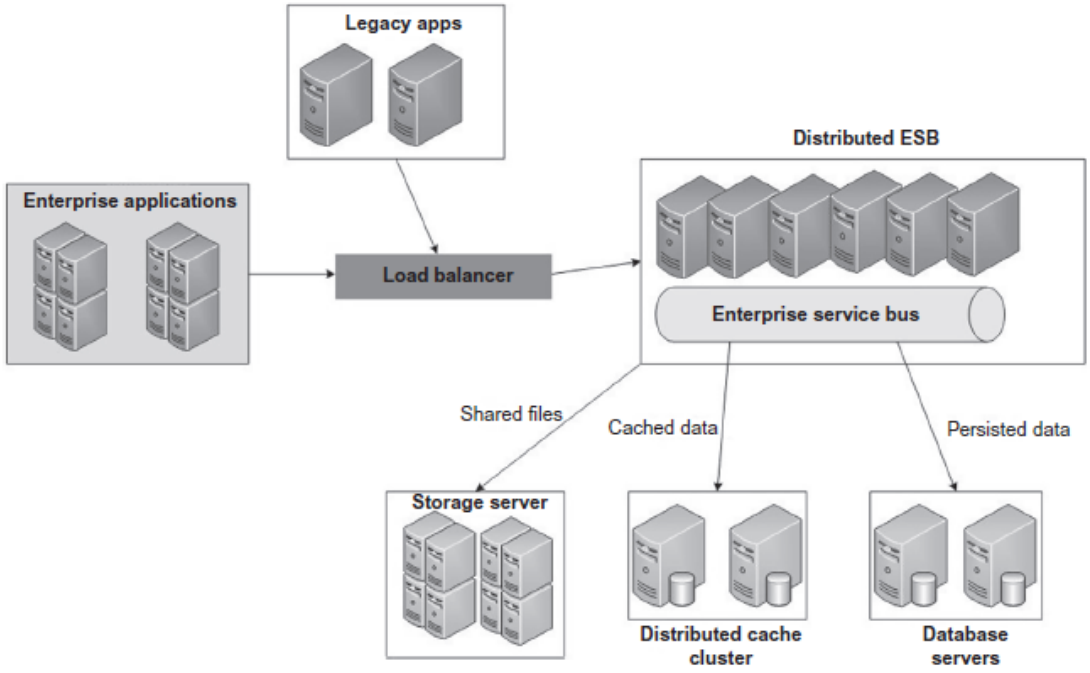


When can this configuration be used? Clustered server configuration works well when the exposed services are homogenous in nature.

Distributed clustered ESB configuration

- An ESB is a message-oriented-middleware (MoM) communication infrastructure that allows interoperability among various diversified components using SOA
- ESB provides: message exchange, protocol conversion, data conversion, scalability, distributed component handling, complexity abstraction, quality of service, and governance

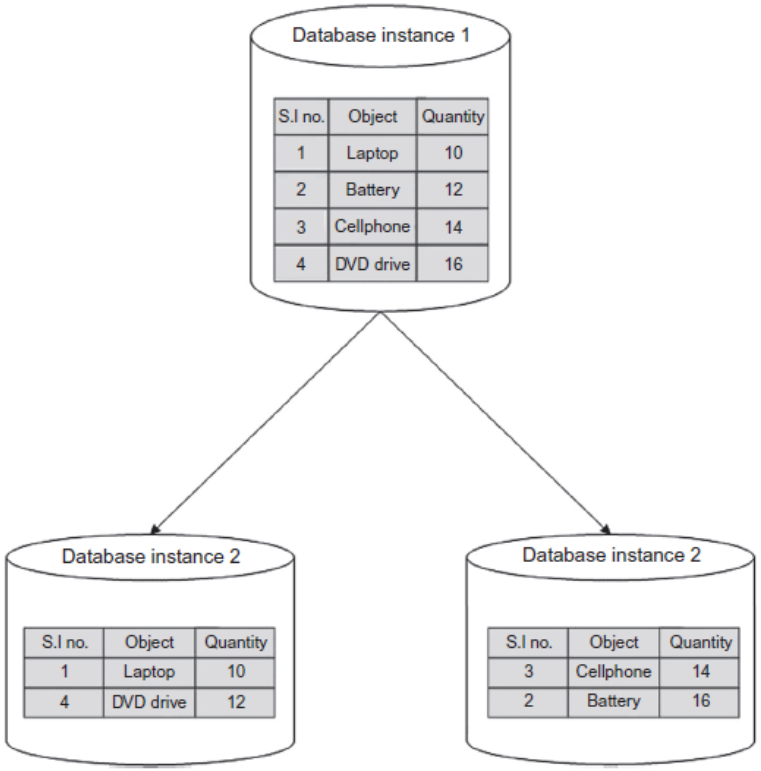
Distributed clustered ESB configuration



When can this configuration be used? As we can see from the benefits, ESB is a more reliable and robust SOA architecture. It provides many enterprise-quality and integration features out-of-box.

Database scalability

- Database storage
- Database cluster
- Indexing
- Data replication
- Partioning
- Data snapshots
- Backup and recovery



RDBMS vs NoSQL

Data characteristics	RDBMS	NoSQL database
Strict schema adherence	Yes	No
Data model containing key-value pairs	No	Yes
Data model containing document-based data	No	Yes
High level of interconnection among data elements	No	Yes
Requirement of high level of consistency	Yes	No
Transaction processing requirement	Yes	No
Large data sets that occur in sequential order such as blog posts	No	Yes

Why NoSQL databases offer a high level of scalability?

- Horizontal scaling or Auto-sharding of data
- Enhanced caching

Storage scalability

- Storage Area Network (SAN) and Network Attached Storage (NAS) systems
- Solid-state drives (SSDs)
- Content Distribution Network (CDN)

Virtualization

- Virtualization is a very efficient method to provide scalability and flexibility.
- Server virtualization (IaaS)
- OS virtualization
- Storage virtualization
- Network visualization

Cloud computing

- Several models:
  - IaaS – Infrastructure as a Service
  - PaaS – Platform as a Service
  - SaaS – Software as a Service
  - Many others
- Advantages:
  - Pay-as-you-go
  - On-demand speed and capacity
  - Elastic scalability
  - Zero infrastructure investment
  - DR env and business continuity
  - Usage and resource monitoring
- Virtualization is a previous step to Cloud Computing