Distributed Systems

Seminar 2 – Distributed System Architectures

Dr. Thomas Springer
The Client/Server model assigns roles to the processes which are communicating with each other.

a) Sketch out the connections and roles between 3 clients $C_1 \ldots C_3$ and 2 servers $S_1$ and $S_2$, whereby the clients refer to $S_1$ to request services and the server $S_1$ uses the services of server $S_2$ under a subcontract relationship.

b) Consider a chain of servers $S_i$ (with $i = 1 \ldots n$) where $S_i$ uses the server $S_{i+1}$ under subcontract relationship. What are the main problems with this organization with respect to the response time observed by a client sending a request to $S_1$ with an increasing number of servers and the availability of the service provided by $S_1$?
Solution E2.1a

- Roles (service usage)
  - Client: Requesting/Invoking an operation/service
  - Server: Implementing an operation/service and providing a result

- Roles (machine/node level, derived according to the main purpose of a node)
  - Client: includes interaction with user, mainly for client role - Laptop, Desktop PC, Smartphone
  - Server: no direct interaction with user, mainly for processing in server role - server computers, VMs in datacenter
Response time:
- can be expected to be bad for large n. The problem is that each communication between two successive layers is, in principle, between two different machines. Consequently, the performance between Client and S1 may also be determined by n – 1 request-reply interactions between the servers.

Availability:
- Another problem is that if one machine in the chain performs badly or is even temporarily unreachable, then this will immediately degrade the performance at the highest level.
- chain can be broken by each of the servers, services of S1 become unavailable independent of S1's availability

Intransparency:
- Even if S1 is local to client, S1 might be far away and produce high delay implying bad response time for the client
E2.1 - Conclusions

- Distributed System Architecture ist determined by involved nodes and their interconnection

- It matters how business logic and data is distributed across the different nodes

- Impact on:
  - Fault tolerance
  - Scalability
  - Performance
  - Development effort
• A peer in a Peer-to-Peer system offers services.
  A second peer would like to use these services.
  a) Sketch out the message exchange involved in the search for and use of services in a centralized, a pure, and a hybrid P2P architecture.
  b) Discuss the advantages and disadvantages of centralized and decentralized architectures based on the three variants in terms of communication overhead, scalability, fault tolerance, and development effort.
- **Centralized:**
  - all nodes are connected to a central node, which acts as a directory

- **Pure P2P – Decentralized, Flat:**
  - no central node; each node is connected to some neighbors; discovery is performed by flooding a request message until the target node is found or the Time to live (TTL) is reached.

- **Hybrid P2P – Decentralized, Hierarchical:**
  - devices are connected to some supernodes which maintain partial directories; discovery is performed by flooding a request message among the supernodes
0: register, 1: search, 2: acquaint, 3: use
Decentralized:

- *A decentralized system is one which requires multiple parties to make their own independent decisions*

Example ant colony

- all ants act upon local information and local interactions to collectively create complex, global behaviour.
## Solution 2.2b

<table>
<thead>
<tr>
<th>Topology:</th>
<th>Centralized</th>
<th>Decentralized, flat</th>
<th>Decentralized, hierarchical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comm. overhead</td>
<td>Low, one message to known server</td>
<td>Very high due to flooding</td>
<td>Medium, grows with no. of superpeers</td>
</tr>
<tr>
<td>Scalability</td>
<td>Limited, due to the limited resources of central node</td>
<td>Limited, due to flooding of network</td>
<td>High, since superpeer structure can be extended with additional superpeers</td>
</tr>
<tr>
<td>Fault tolerance</td>
<td>Low, if central node fails, complete system fails</td>
<td>High, rate of unanswered lookups due to limit on no. Of hops</td>
<td>Medium to low, if superpeer fails, only subset of nodes not findable</td>
</tr>
<tr>
<td>Complexity</td>
<td>Two node types, Low, coordination is centralized, other nodes are simple</td>
<td>Single node type, Low to medium, simple behavior of all peers</td>
<td>Two node types, High for self-management/coordination behavior of superpeer coord. hierarchy</td>
</tr>
</tbody>
</table>
For complex distributed systems, multi-tier architectures are predominately used.

a) How many and which tiers would you propose for an online trading platform which enables customers to shop via the WWW? Sketch out your solution!

b) Assign the following system functions to an architecture tier:
   1. The functions of a shopping basket
   2. Input forms for the editing of customer data by the customer
   3. Discount calculation
   4. Preparation of the contents of the shopping basket for presentation to the user
   5. Checking of access rights
   6. Saving of customer data

c) Discuss the terms horizontal and vertical distribution and assign multi-tier architectures to one of the two approaches.
Solution E2.3a

**Tier 1**
User interface in browser

**Tier 2**
Web server

**Tier 3**
Application logic

**Tier 4**
Data retention systems
2. Input forms for the editing of customer data by the customer

1. The functions of a shopping basket
2. Input forms for the editing of customer data by the customer
3. Discount calculation
4. Preparation of shopping basket content for presentation
5. Checking of access rights
6. Saving of customer data

Tier 1: User interface in browser
Tier 2: Web server
Tier 3: Application logic
Tier 4: Data retention systems
- **Vertical distribution** refers to the distribution of the different layers in a multitiered architectures across multiple machines. In principle, each layer is implemented on a different machine.

- **Horizontal distribution** deals with the distribution of a single layer across multiple machines, such as distributing a single database.

- Examples of layers:
  - Server-side of Client/Server applications can be broken down into three distinct layers:
    - User interface layer
    - Processing layer
    - Data layer

- Multi-tier architecture is vertical distribution
Imagine an e-commerce system for a company of global scale. Warehouses are spread across the globe and contain millions of products. Discuss architectural solutions for a store management system that ensure scalability, availability and acceptable performance/response times for clients with focus on partitioning and replication strategies for business logic and data. Discuss pros and cons of your solution.
Example: Customer Database

Partitioning: split up of logic or data

- E.g., customer surname A-D, E-H, I-L, ...
- Customer DB per country
- Split up of e-commerce logic into customer management, store management, order preprocessing, ...

Replication: Full copies of logic or data

Combined: Partial Replication
Replication of data
- Load balancing and parallel processing supported
- Fault tolerance achievable
  - Need for synchronization
  - Maybe issues with consistency or limitations on parallelism

Partitioning
- Partitioning of data per country
- Scalability supported
  - Response times could be low if customer is far away from home
  - Performance issues operations across several partitions are required
  - Reorganisation of partitions can be expensive
Imagine you are a consultant working for a startup company that intents to sell an innovative product via an e-commerce platform. What advice would you give to the company according to the decision either to provide the e-commerce system on-premises or to deploy it in a cloud infrastructure? Discuss the consequences of the advised decision.
<table>
<thead>
<tr>
<th></th>
<th><strong>On-premises</strong></th>
<th><strong>Cloud</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scalability</strong></td>
<td>Possible but expensive</td>
<td>Elasticity on-demand</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td>Intranet, data protection can be ensured, but experts needed</td>
<td>Trust to cloud provider required, location of data centers matter</td>
</tr>
<tr>
<td><strong>Complexity</strong></td>
<td>Simple for small no. of clients, gets complex for increasing client no. (scale out)</td>
<td>Higher complexity at beginning, than moderate increase to support scale out</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>Initially moderate, but high cost with increasing system size</td>
<td>Scalable cost, pay-per-use</td>
</tr>
<tr>
<td><strong>Maintenance</strong></td>
<td>Local admin needed (has to be paid)</td>
<td>No admins for system administration and maintenance but cloud expert needed</td>
</tr>
<tr>
<td><strong>System access</strong></td>
<td>Local access to physical machines, Full control</td>
<td>Remote access via admin interfaces, limited control</td>
</tr>
</tbody>
</table>