Notes on N-Tier Architectures

Jeff Offutt

http://www.cs.gmu.edu/~offutt/

SWE 642

Software Engineering for the World Wide Web

N-Tier Architectures

- Distributed application : Programs run on two or more host computers
- Architecture : Where the programs run, what their responsibilities are, and how they interact
- Tiered Architecture : Programs only communicate with each other if they are on adjacent tiers
  - information flow is linear — tier 1 programs do not communicate with tier 3 programs
- Client-server : Programs run on two computers
  - They usually interact in a "master-slave" relationship (client is the master
  - This is also called "2-tier"
- 3-Tier : A third computer is used (typically a DB)

N-Tier Architectures (2)

- N-Tier : An unlimited number of tiers
- Each tier may have multiple computers
- Advantages :
  - More powerful applications
  - Many services to many clients
  - Enhanced security, scalability and availability
- Disadvantages :
  - Software is more complex (effects design, reliability, maintainability)
  - Much more complicated to design and model
  - Performance risks
  - Not sure how to achieve reliability
  - Software maintenance is very different

Six Major Quality Attributes

Effects of N-Tier architectures

- Reliability : New methods are needed
- Usability : Separation makes usability easier to achieve
- Security : Tiers provide for security "walls"
- Availability : Tiers enhance redundancy
- Scalability : Fairly easy to expand services
- Maintainability : Good design – maintenance is easy … bad design – maintenance is hard

Comparison of Architectures

Architecture	Pros	Cons
One tier	Simple Very high performance Self-contained	No networking – can't access remote services Potential for spaghetti code
Two tiers	Clean, modular design Less network traffic Secure algorithms Can separate UI from business logic	Must design/implement protocol Must design/implement reliable data storage
Three tiers	Can separate UI, logic, and storage Reliable, replicable data Concurrent data access via transactions Efficient data access	Need to buy DB product Need to hire DBA Need to learn SQL Object-relational mapping is difficult
N tiers	Support multiple applications more easily Common protocol/API	Less inefficient Must learn API (CORBA, RMI, etc.) Expensive products More complex, more faults Load balancing is hard

Challenges of N-Tier Architectures

- Communication and distribution is usually handled by third-party middleware (CORBA, EJB, DCOM, etc)
- Software becomes heterogeneous and parallel
- A lot of new technologies to learn
- Designing truly reusable objects is difficult
  – the design must be high quality
  – they may not satisfy the needs of future systems

Challenges of N-Tier Architectures (2)

- General distributed object protocols are slow
  - This is usually not important because the internet is so slow, and if it is, more speed can usually be achieved by adding more hardware
- Load balancing is quite difficult : distributing requests to computers such that each computer does approximately the same work

In small systems, everything is simple, but in large systems, the overall software design is crucial to product success

Web Clustering

- Web sites can no longer grow by adding a bigger server
- Modern web sites use groups of servers that act as a single unit or cluster
- Adding new servers to the cluster allows for scalability
- Adding new servers adds redundancy, which increases availability

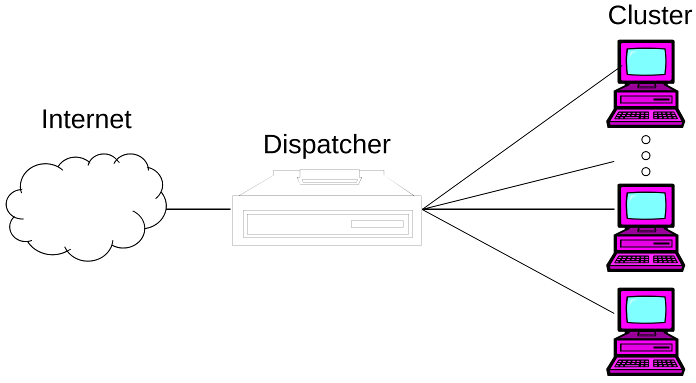
Web Clustering

Clustering introduces problems with security :

- The primary security protocol is Transport Layer Security (TLS) (Formerly SSL)
- TLS uses a large amount of processing to set up a secure session
- The fact that HTTP is sessionless means that TLS's information must be cached on a server
- But with clustering, the next request from a client may be on a different server, thus the TLS information may **not** be available

Web Clustering (2)

Most clustering techniques use a dispatcher in front that routes requests to cluster members :



Web Clustering (3)

- Three categories of clustering technologies:
  1. L4/2 : Layer 4 switching, 2 packet forwarding
  2. L4/3 : Layer 4 switching, 3 packet forwarding
  3. L7 : Layer 7 switching, 2 or 3 packet forwarding

- TCP / IP packets contain several layers of information, layers 3 (network) and 4 (transport) contain :
  – Source and destination IP addresses
  – Source and destination protocol addresses (ports)
  – Information as to whether the packet starts a session or continues a session

Web Clustering (4)

- Dispatchers choose a server based on the data in the packets and a load-sharing algorithm
  – Continuing sessions are usually sent to the server that is already assigned
- L4/3 clusters :
  – Each server has its own IP address
  – Dispatcher must do more checks on the packet header
  – The server gets packets from the dispatcher, and cannot respond directly to the client
- L4/2 clusters are <u>faster</u> :
  – All servers have the same IP address (layer 3)
  – Once given a packet, the server can respond directly to the client without going through the dispatcher

Web Clustering (4)

- L4 dispatchers are “content blind”, requiring that each server have the same file systems (replicated or shared)
  - If data is static, no problem
  - Replicating data is very difficult
  - Sharing file systems is slow
- L7 clusters are more intelligent :
  - Dispatchers look at application data (OSI layer 7)
  - Dispatchers can be more intelligent – different servers can be used to serve different types of requests
  - Each dispatch decision is more complicated – the application data is usually not structured

N-Tier / Clustering Summary

- The clustering is mostly transparent to the programmer
- It effects the high level design (architecture) of web apps, but not very much