Understanding Application Servers

Author: Ajay Srivastava & Anant Bhargava
TCS, Jan'03

Background

Application servers, whatever their function, occupies a large chunk of computing territory between database servers and the end user. This paper will provide an overview of application server and their role in n-tier applications.

Intended Audience

The paper is intended for anyone interested in knowing application server architecture and it's role in web applications. The readers of this paper are not require to have an in-depth knowledge of web applications and related technologies but they are require to have an overview of n-tire web application architecture.

History of Application Server

In the early days of application servers, it was realised that applications (the programs people were using to get work done) themselves, were becoming bigger and more complex -- both to write and to maintain. At the same time, pressure was increasing for applications to share more of their data and sometimes functionality with other applications. More applications were either located on a network or used networks extensively. It seemed logical to have some kind of program residing on the network that would help share application capabilities in an organised and efficient way -- making it easier to write, manage, and maintain the applications.

The end result of this thinking was what is now called an application server. However, these servers first appeared in client/server computing and on LANs. At first, they were often associated with "tiered" applications, when people described the functionality of applications as two-tiered (database and client program), three-tiered (database, client program, and application server), or n-tiered (all the above plus whatever). This was (and still is) a complex model of application development, and it resisted wide-scale implementation.

The Word Application Server Means

Before going further, it will be useful to explore the world of the Application Server and to know the answers of some of the most frequently asked questions like:

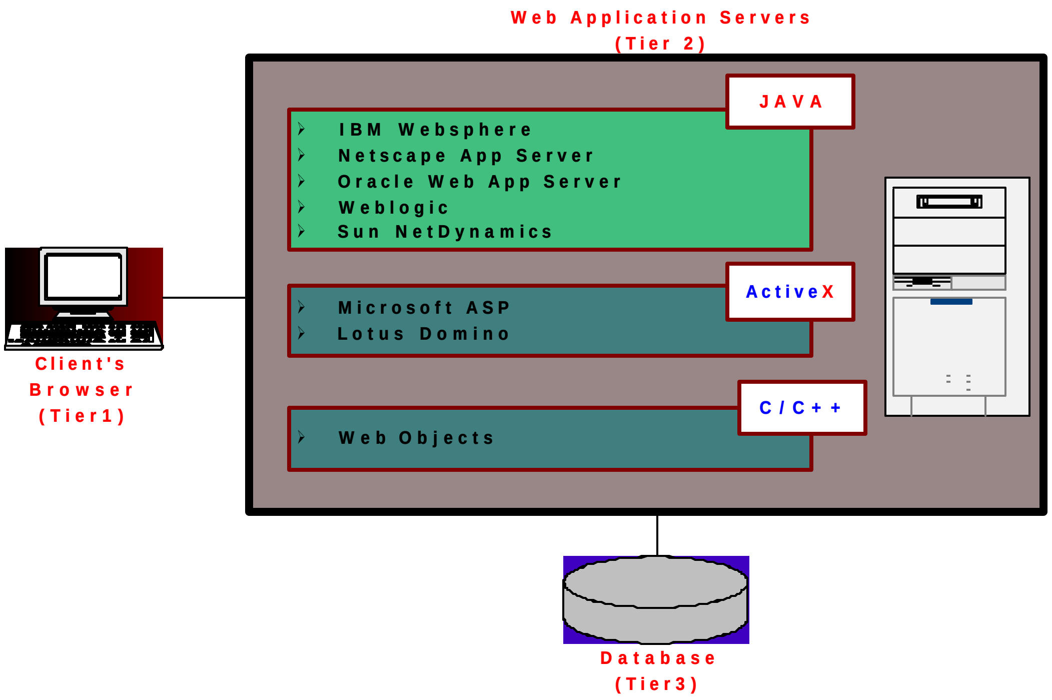
What is an Application server?

An Application server is a server program that resides in the server (computer) and provides the business logic for the application programme. The server can be a part of the network, more precisely the part of the distributed network. The server program is a program that provides its services to the client program that resides either in the same computer or on another computer connected through the network.

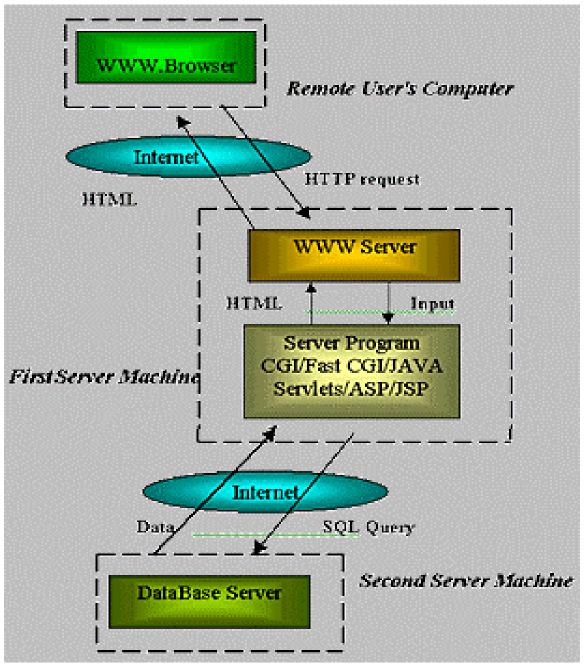
Where exactly the application server fits in?

Application servers are mainly used in web based applications that have 3-Tier architecture.
1st Tier: Front end - Browser (thin client), a GUI interface lying at the client/workstation.
2nd Tier: Middle tier - Application Server - set of application program.
3rd Tier: Back end - Database Server.

The application server is a second / middle Tier of the 3- tier Architecture. Please refer the diagram below:



How Information flows via application server?



The Application server syncs / combines with the web server for processing the request made by the client.

If we look at the request-response flow between client, web server and application server then we come to know that the client's request first goes to the web server, which sends the required information to the application server. The application server then sends back the response to the web server after taking an appropriate action. The web server further sends the processed information back to the client. Web server use different approaches or technology for forwarding or receiving back processed information. Some of the most common approaches are CGI (Common Gateway Interface), ASP (Active Server Pages), JSP(Java Server pages), Java Servlets, Java Script etc.

What Application Server do

First and foremost, application servers connect database information (usually coming from a database server) and the end-user or client program (often running in a Web browser). There are many reasons for having an intermediate player in this connection:

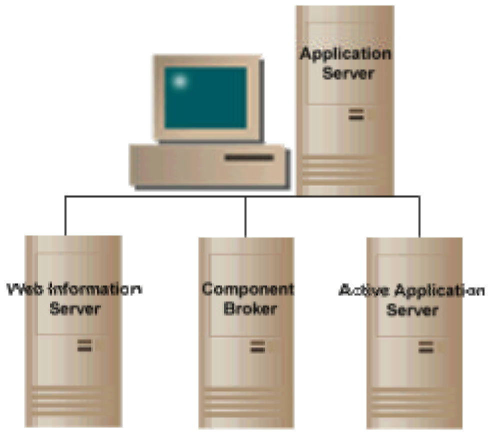
- a desire to decrease the size and complexity of client programs
- the need to cache and control the data flow for better performance
- a requirement to provide security for both data and user traffic.

Application servers have different roles, but not every implementation requires the same functionality. Someone might want an application server that simply helps organise their applications for the Web, give them better control over the business logic they contain, and make it easier to monitor and secure the data. They don't need thousands of servers. Other companies, especially big ones, do need to manage thousands of servers. For them, the scalability of an application server is crucial. So some application servers feature scalability, others feature other things, and some try to do everything. What's most important: security, scalability, business logic management, or database connectivity?

Features of Application Servers

Component Management: Provides the manager for handling all the components and run time services like session management, synchronous/asynchronous client notifications and executes server business logic.
Fault Tolerance: Ability of the Application server with no single point of failure, defining policies for recovery and fail-over recovery in case of failure of one object or group of objects.
Load Balancing: Ability to send the request to the different servers depending upon the load and availability of the server.
Transaction Management
Management Console: Single point graphical management console for remotely monitoring clients and server clusters
Security: Security features for applications security

Types of Application Servers



Application servers are categorised mainly in 3 types:

Web Information Servers

This type of server employs HTML templates and scripts to generate pages incorporating values from the database in them. These types of servers are stateless servers. Some of these types of servers are Netscape Server, HAHT, Allaire, Sybase, SilverStream etc.

Component Servers

The main purpose of these servers is to provide database access and transaction-processing services to software components including DLL's, CORBA, and Java Bean. Firstly, they provide environment for server-side components. Secondly, they provide access to database and other services to the component. These types of servers are stateless servers. Some of these servers are MTS (provides interface for DLL), Sybase Jaguar, IBM Component broker.

Active Application Server

This type of server supports and provides a rich environment for server-side logic expressed as objects, rules and components. These types of servers are stateful servers and best suitable for dealing with e-Commerce environments.

NOTE: Stateful servers are those servers which play the role of transaction co-ordinator and manage the data state, while Stateless servers doesn't do these things completely on their own, but have to rely on database or transaction monitors for the completion of the transaction.

Selection of Application Server

Choosing the correct application server to run the J2EE application can play an integral part in the application's success. To decide about an Application Server, the project requirements should first be nailed down. The decision you make will play a role in many aspects of the application, not the least of which is the budget. Different application servers are targeted for different types of applications. The first step should be to identify and prioritise the needs of the particular environment. The criteria might include:

- Performance
- Cost
- Development
- Support
- Compliance
- Administration
- Scalability/Reliability

After you've prioritised, you can begin looking for the application server that best meets the needs. If you run in a large-scale environment (Intranet or Internet) and expect to have lots of traffic to the site, performance should be at the top of the list. Not all app servers perform alike. In fact, a wide range of issues can contribute to how well (or how poorly) the server performs under load. Some of the answers are easy to find, while others might take some investigation or evaluation of the app server by using a similar application and taking some performance metrics.

Here is some performance factors to consider:

- Use of connection pooling and the types of drivers provided for JDBC support
- Use of caching features, and how configurable the cache is
- Support of Web servers for request-handling performance
- If you plan on using CMP, how is it implemented for the EJB support?

If performance isn't at the top of the list, it is still important to understand some of the issues surrounding app-server performance. Sooner or later you'll need it.

Cost

Cost is usually another factor that will play a role in the decision. Make sure the server you are considering is affordable. When evaluating costs look at:

- Pricing structure
- Support and maintenance costs
- Hardware requirements
- Development licenses

Development and support

Development and Support are two factors that can make or break a project, depending on how experienced the development team is. There are few questions, which you should ask yourself:

- Is training available if my team needs it?
- How quick is the support turnaround?
- What IDEs are integrated with the app server?
- What platforms are supported, and can development take place on multiple platforms?
- Is runtime server debugging supported or possible?

If you have a development team working on the project, make sure the application server supports the creation of multiple application spaces. Nothing will slow a team down more than if they're stepping all over each other while trying to debug and deploy an application. As far as compliance goes, make sure to determine ahead of time what version of specifications the project will be using. Then check to make sure that the app server supports those versions. Not every application server supports the latest version of JSP, EJB, JMS, or Servlet specs.

Administration

Administration is a key issue, but sometimes it's overlooked. You can waste a huge amount of time in configuring app servers. You need remote admin capabilities, as well as command-line access. The ease of deployment during development is also a large hidden cost. If it takes each engineer five minutes to deploy every time they need to test, the cost quickly adds up in terms of lost time. Being able to monitor the server is also important for making corrections to deployment parameters.

Scalability and Reliability

You also need to consider scalability and reliability. Lack of reliability means lost customers—it's as simple as that. If the application server can't scale with the application, it will cost you in the long run. Features that might be important for the application include fail over support, clustering capabilities, and load balancing. As you add more scalability features, you'll find the cost of the product starts going up.

In a project some time should be definitely devoted for evaluating the application server that best fits the needs. The time spent up front will be worth it!

Conclusion

The paper gives an overview about application server and rules, which need to be considered while selection of application server. By following these rules, there are better chances to understand and select an application server that best suits to you.