2. Evolution of computing

2.1. Internet technology and web-enabled applications 2.2. Web application servers 2.3. Overview of Computing Paradigm: Grid Computing, Cluster Computing, Distributed Computing, Utility Computing, Cloud Computing 2.4. Internet of services 2.5. Adopting Cloud Computing in Business.

Internet technology and web-enabled applications

Internet-based applications rely fundamentally on HTTP, the HyperText Transfer Protocol, and HTML, the HyperText Markup Language; both are now standards defined by the world wide web consortium (W3C). Browsers, such as Internet Explorer, and servers, such as HTTPD (HyperText Transfer Protocol Daemon) implement these standards to enable content publishing over the internet.

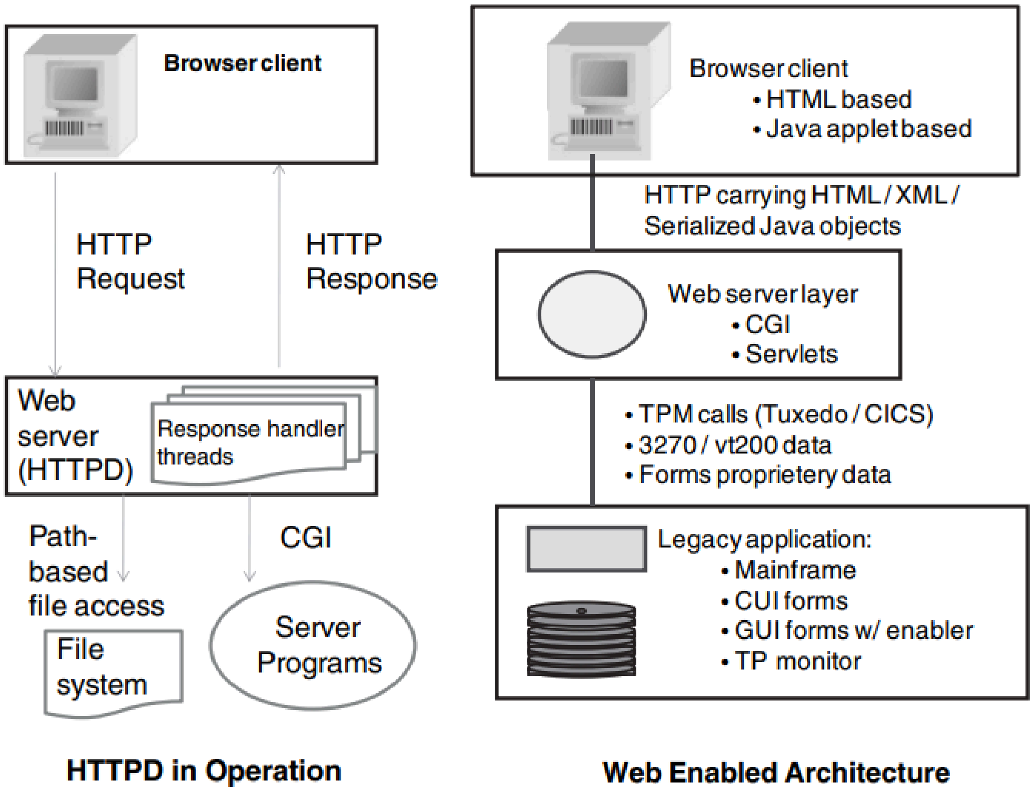

FIGURE 2.1. Internet technology and web-enabled applications

As depicted to the left in Figure above, a web server is a process, such as the Apache HTTPD daemon (see below), that receives HTTP requests from clients, typically web browsers. Requests are queued until assigned to a request handler thread within the web-server process. The server returns an HTTP response containing data, either retrieved directly from a file system path or as computed by a server program initiated to respond to the request. The CGI (common gateway interface) protocol is used by the web server to launch server programs and communicate with them, i.e. pass parameters and accept their results, such as data retrieved from a database. The browser client merely interprets HTML returned by the server and displays it to the user.

In the initial years of the web (through the late 90s), the HTTP protocol, together with features available in HTML supporting data entry forms, presented the opportunity to develop browser-based (or 'web-enabled') interfaces to legacy systems. This became especially useful for accessing mainframe applications that otherwise could be accessed only from dedicated terminals.

WEB APPLICATION SERVERS

In a web-enabled application architecture, processing logic, including database access, took place outside the web server process via scripts or programs invoked by it, using CGI for interprocess communication. Each such 'CGI-script' invocation included the costly overhead of launching the required server program as a fresh operating-system process.

The invention and proliferation of the Java language, designed to be portable across machine architectures with its interpreted yet efficient execution model made possible alternative approaches to execute application functionality inside the web-server process, leading to the birth of the 'application server' architecture.***The 'servlet' container, for example, first introduced in the 'pure Java' Apache Tomcat server, allowed Java programs to execute in a multi-threaded manner within the server process as 'servlet code.' The container would also manage load balancing across incoming requests using these threads, as well as database connection pooling, in a manner similar to TP monitors.***

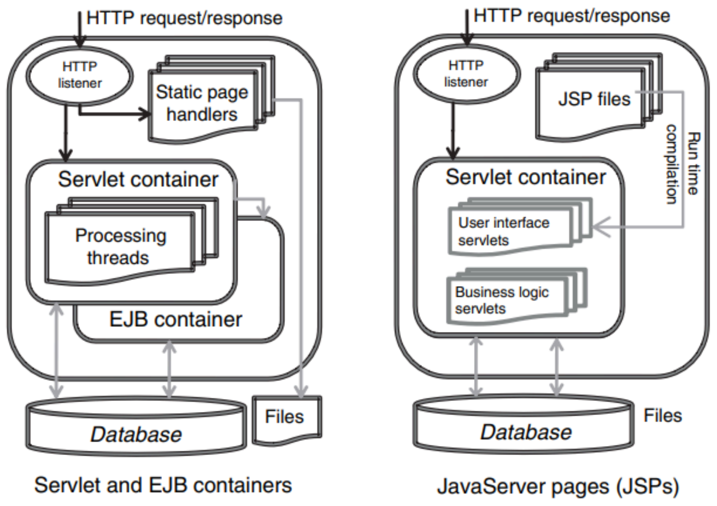

Figure 2.2. Web application server

A major drawback of servlet applications was that they mixed-up user interface and business logic code, resulting in applications that were difficult to maintain. ‘Java server pages’ (JSPs), also introduced in Tomcat, allowed user interface behavior to be encoded directly as Java code embedded within HTML. Such ‘JSP’ files are dynamically compiled into servlets, as illustrated on the right in Figure 2.2. Using JSPs enabled a clear separation of user interface and business logic to be enforced in application code, resulting in better maintainability.

While elements of the load balancing feature of TP monitors were present in servlet containers, these could not yet scale to large transaction volumes. As an attempt to bridge this gap, the ‘Java 2 Enterprise Edition’ (J2EE) specification was developed by Sun Microsystems in 1999, introducing a new application execution container called ‘Enterprise Java Beans’ (EJBs). Application code packaged as EJBs could be deployed in separate processes from the controlling web application server, thereby opening up the possibility of distributed multiprocessor execution to boost performance. The EJB container also provided a host of additional services, such as security, transactions, greater control on database connection pooling and Java-based connectors to legacy systems.

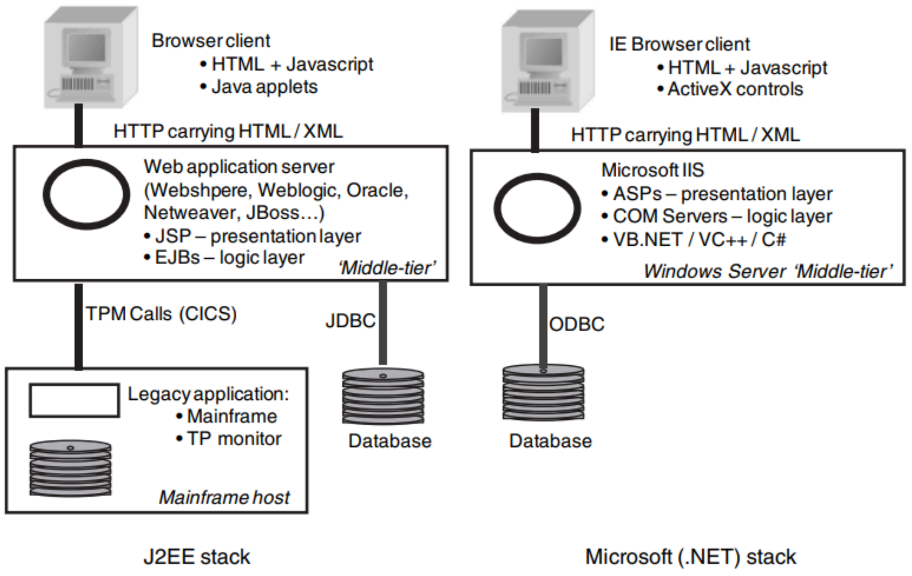

FIGURE 2.3. Web application server technology stacks

A competing family from Microsoft was being developed, as depicted alongside the J2EE stack in Figure 2.3. The Microsoft web/application server, IIS (Internet Information Server), runs only on the Windows operating system. However, unlike the J2EE stack, multiple language support was provided, including C, C++, and Microsoft specific languages such as C# (C 'sharp') and VB (visual basic). The application container in this case was simply Microsoft's COM environment on the Windows operating system that enabled multiple processes to execute and communicate.