Application Management: What is out there for the Linux Industry

Chris Lesar, Vice President Operations
Dirig Software
One Indian Head Plaza, 6th Floor
Nashua, NH 03063
603-889-2777
Fax: 603-889-2471
clesar@dirig.com

As Linux continues its breakneck pace of evolving into a globally accepted server operating system, robust ebusiness solutions with high transaction volumes are being developed on a variety of Linux distributions. To facilitate these ebusiness development efforts, leading vendors such as IBM and BEA systems have released versions of their Java-based application servers supporting Linux. The growth of content and commerce transactions online has spawned the need for the next evolution of management products addressing the most granular details of ebusiness performance metrics. New technologies now provide Linux application management down to the components of an ebusiness solution, which includes Web applications, Java server pages, servlets, enterprise Java beans and connection pools.

Component Management - Managing Application Servers

For the past 10 years, there have been drastic transformations in managing a companies' infrastructure. In the beginning, much of the focus was placed on the network. Gathering hundreds of statistics across routers, hubs and switches was commonplace. Eventually, when network stability was established, the management focus changed to servers and desktops. Basic information of server health was important to the overall performance of the network. However, this information proved inadequate, yielding way to applications management products. These products would supply detailed information about an application, including statistics from the application, outages, as well as end-to-end response time. Although these management products deliver incredible amounts of information about applications, the popularization of the Internet, specifically e-commerce, requires even more scrutiny and detail about application performance. Thus management products must evolve into managing the individual components of an application.

Along with the expansion of the Internet, a new breed of software framework has emerged. In the past, a large majority of applications were developed using database technologies from Oracle and Microsoft. As companies continue to improve their Internet presence, the challenge of developing applications that could be accessed through the Web, while continually accessing a database has forced a new technology, called Application Servers. Application Servers facilitate the merger of Web and database technologies through a series of smaller software objects called *components*. Components, in their simplest form, are small pieces of software that operate within an application server. Each component is generally responsible for a very explicit task, such as verification of a credit card transaction.

Application Servers are developed using many different types of technologies. However, they can easily be broken up into several different logical elements. At the core of the application server is the **transaction engine**. The **Web Interface** allows Web browsers to connect to the application server invoking **components** within each **Web application**. In turn, each component connects, through **connection pools**, to the backend database. The diagram below details the different parts of most Application Servers, and their relationship between them.

The core of an application server is the **transaction engine**. It is a traditional system process (NT Service or Unix Daemon). It has several responsibilities including message routing, component and database connection management and traditional transactional processing.

The **Web Interface** provides a mechanism for Web browsers to invoke different operations within each Web application, for example buying books or database searches. Web interfaces are implemented differently based on the type of Application Server. For example, Java-based Application Servers (such as BEA Web Logics or IBM WebSphere), use a combination of Java Server Pages (JSP) or servlets to access the Web applications. Microsoft uses Visual Basic Script (vbscript) or Active Server Pages to communicate with each Web application.

The most important pieces of Application Servers are the **Web applications** and their **components**. Each Web Application is comprised of a series of components that perform a variety of specific tasks. All of the application knowledge resides within each component. In Java based platforms, these components are Enterprise Java Beans (EJB). Microsoft uses their Common Object Model (COM+) objects to perform the individual tasks.

Finally, the transaction engine communicates to the database through a series of **connection pools**. Each pool consists of a series of dedicated connections, which are allocated to components when necessary. This improves overall application server performance as well as abstract the complexity of the database from each Web application and its' components.

Recently, Microsoft has been enhancing their version of the application server. Originally developers used Microsoft Transaction Server (MTS) to integrate Web Servers and databases, using COM objects as the component level interface. With the development of COM+, users will now use the more recent application server framework supplied through Microsoft Distributed Transaction Coordinator (MSDTC). However, this migration is moving slowly, and it is important for products to manage both environments.