

Akamai teams with WebSphere to offer on-demand edge platform

At the Edge

BY JAY PARIKH



ABOUT THE AUTHOR

Jay Parikh is director of Engineering for Akamai's globally distributed computing service, EdgeComputing. He drives product direction and provides coordination and continuity in Engineering and across other organizations at Akamai. In addition to EdgeComputing, Jay has supported other Akamai customer-facing services, including Akamai's flagship content delivery service, EdgeSuite; Edge Side Includes (ESI), a markup language for dynamic assembly and delivery of Web applications at the edge; and FirstPoint, Akamai's global load-balancing service.

E-MAIL
jay@akamai.com

As more enterprises move their business to or enable their business on the Internet, Web applications have come into widespread use in many enterprise application infrastructures. The infrastructure to deliver these Web applications typically includes a wide range of technologies such as load balancers, HTTP Web servers, caching servers, messaging systems, transaction-processing monitors, application servers, and databases. A typical enterprise application infrastructure is shown in Figure 1.

As performance and geographic reach requirements expand, it becomes increasingly more difficult to scale the Web site infrastructure. IT managers must continually evaluate capacity plans to keep pace with the expected peak demand, and planning must consider events such as marketing promotions, news events, and other events that inevitably create more planning uncertainty. Errors in planning can result in overloads that can crash sites or cause unacceptably slow response times, leading to lost revenue.

Pre-provisioning extra capacity as insurance against overload is financially unacceptable for most enterprises. Ideally, enterprises want the needed resources when – and only when – they are needed; they do not want to buy extra resources that sit idle when they are not needed. “On-demand” computing provides better utilization of computing resources and represents a model in which they are brought into service as needed.

On-Demand Application Platforms

Today, on-demand computing technologies have been integrated

into centralized enterprise application platforms, and generally offer enterprises improved fault tolerance and better scalability, through the use of intelligent scheduling and load balancing of application workloads. However, for many of today's Web applications, end users are inherently spread across the Internet. Congestion and failures in this end-user environment are common and because of this, centralized enterprise applications can become unreliable and their performance can become unpredictable.

One solution for enterprises is to use a distributed application delivery platform at the edge of the Internet to ensure optimal application performance and reliability. Akamai is one example of an on-demand edge application platform. Deployed at the “edge” of the network – close to user access points – it consists of nearly 15,000 servers in over 1,100 networks around the world. Enterprises can deploy applications to this distributed platform in order to avoid service bottlenecks and failures, and at the same time provide on-demand scalability,

global reach, and high performance for application users. Enterprises are not limited to serving static content from edge networks; several edge networks offer enhanced services. One such capability is Edge Side Includes (ESI), which is specifically designed for dynamic content assembly.

In order to extend the enterprise J2EE programming platform to the edge network, IBM and Akamai have integrated and deployed WebSphere Application Server (WAS) version 5.0 onto the edge servers. Enterprises can execute J2EE Web tier applications in an on-demand WebSphere environment, known as Akamai EdgeComputing powered by WebSphere, and consume Internet computing resources on a pay-per-use basis.

This widely distributed application environment, shown in Figure 2, consists of an end user typically using a browser; the enterprise (origin) running business logic, legacy systems, and databases; and the edge servers running an embeddable WAS server that supports the J2EE Web application programming model.

Developing Applications for an Edge Platform

The development model remains standard J2EE for edge applications and does not require the use of any proprietary APIs; it is the deployment model that changes. If your applications generally follow J2EE component programming best practices, adapting the existing application for the edge will be easier. Akamai's edge application platform extends the WebSphere application programming platform to enable the execution of J2EE Web tier application components – JSPs, servlets, tag libraries, and JavaBeans.

Development for an edge application platform still relies on standard J2EE development tools and best practices in developing applications, but you must architect your edge-enabled application as two cooperating sub-applications: an edge-side application and an enterprise-side application.

Presentation Components on the Edge

The presentation components are the most common application components to deploy to the edge. These components access enterprise data via the Java Web services client model. Typically, the Web application will be developed using a framework based on the Model-View-Controller (MVC) architecture.

Jakarta's Struts framework, an open source framework for building Web applications based on the MVC pattern, is well suited for edge deployment. Struts provides its own Controller component via a servlet of class `ActionServlet`. This servlet is configured by defining mappings, `ActionMappings`, between the requested URI and the class name of an Action class. For the View, Struts can leverage JSPs and provides a very flexible and powerful set of JSP tag libraries that allow you to build the View portion of an MVC application without embedding Java code directly into the JSP itself. The Model component is commonly represented by JavaBeans. These Model JavaBeans may be self-contained or represent facades for other components such as JDBC or EJBs.

The View and Controller components of a Struts application are good candidates for distribution to the edge network. These components execute on the edge servers and can interact with Model components (EJBs) running at the enterprise. Depending on the functionality of your application, the extent to which these applications can move onto the edge application platform will vary. The edge View and Controller components are bundled, along with other Java classes, into a Web application archive (WAR) and deployed onto the edge server network.

Normally in an enterprise environment the Web tier will use RMI (Remote Method Invocation) to communicate with the business tier. For example, a servlet may use RMI-IIOP to call an EJB. However, with J2EE application components running in a distributed WAN environment, EJBs will remain running at the enterprise and are made accessible to the edge application via Web services inter-

faces. A session bean facade can be used to expose the necessary business logic as a Web service to the edge application. The edge application makes a Web service call back to the enterprise to invoke the appropriate logic through a session bean facade, perhaps made visible through a servlet running on the enterprise application servers. The edge application can use JAX-RPC to call the Web service at the enterprise via SOAP/HTTP(S). The edge application platform enables an edge application to cache the results of SOAP/HTTP(S) requests to optimize interactions with the enterprise.

In addition to Web services as a communication channel, other standard communication protocols are supported, including the following:

- **HTTP(S):** An edge application can make HTTP(S) requests to the central enterprise, using the `HttpURLConnection` class, for example, to request content or data objects. Any content fragments can be fetched and used to build the end-user response. Data objects such as XML files can be fetched and then transformed (XSLT) or parsed (JAXP) to create the end-user response. HTTP responses from the enterprise can be cached on the edge, so content or data objects can be persisted on the edge across user requests to further reduce the load on the enterprise.
- **JDBC:** Akamai provides a Type 3 JDBC driver that allows edge applications to tunnel JDBC queries to the enterprise via HTTP(S). If an application has already been developed using JDBC for data transaction with a database, the JDBC/HTTP mechanism will make it easier to adapt an application for the edge. The edge application can still use JDBC and benefit from the relational data form. Further, JDBC query responses can be configured to be cached on the edge servers.
- **RMI:** WebSphere applications running at the edge can use RMI-IIOP tunneled over HTTP(S) to communicate back to the enterprise. In this configuration, a servlet runs at the enterprise, intercepts the RMI requests from the edge, and translates the requests into the appropriate RMI method calls to the business tier. Since RMI calls are set to be uncacheable in the edge application environment, it is recommended that you use Web ser-

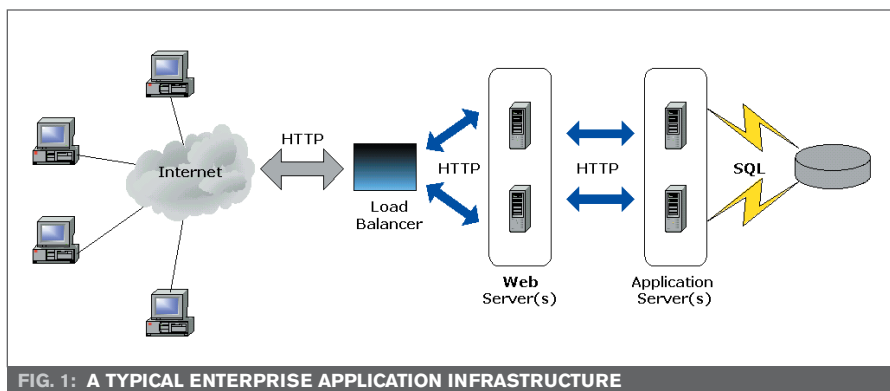


FIG. 1: A TYPICAL ENTERPRISE APPLICATION INFRASTRUCTURE

DATA ACCESS AND UPDATE FREQUENCY	Read-only access Low update frequency	Read/write access Low update frequency	Read/write access High update frequency	Write access Low/high frequency
BEST PRACTICE	Local DB on edge (Cloudscape)	Web service from edge to origin – cache at edge	If read access can be cached and preserve consistency needs, use Web service from edge to origin	Tunnel from edge to origin (Akamai edge to origin protocol optimizations)

TABLE 1: BEST PRACTICES FOR DIFFERENT DATA ACCESS REQUIREMENTS

VICES over RMI to achieve better performance.

Data Access on the Edge

Any of these HTTP-based protocols are useful interfaces that allow your applications to bridge from the edge to the enterprise, but it is still important to avoid excessive communication for edge-origin requests, as end-user latency and origin utilization will increase. Since there is an absolute cost for every round-trip from the edge to the enterprise, calls should be as effective as possible. Requests should be bundled, if possible, and edge caching should be used to store data and other objects across requests.

The EdgeComputing platform provides caching mechanisms to persist application data to minimize the interaction and load on the enterprise infrastructure on a request-by-request basis.

- **ServletContext:** Any data that can be used across users for a particular application can be stored in the application's ServletContext object.

- **HttpSession:** Akamai supports the standard HttpSession interface on the edge application platform. The platform provides replication for HttpSession objects across the edge server nodes. HttpSession objects are most commonly used to keep per-user state information, but keeping them small in size is important for performance reasons.
- **HTTP cookies:** An edge application can store some user-specific data in user cookies or URL tokens, but privacy and security concerns may prevent the use of cookies to store data in some situations.
- **Object caching:** As previously mentioned, an edge application can make requests for data or content (using HTTP, Web services, or JDBC) and these objects can be stored in the edge node's cache.

Another powerful edge data capability employs IBM's 100% Pure Java Cloudscape DBMS to host infrequently changing, read-only data in an edge database. In this model, application data is exported into a JAR file (done by the enterprise application developer), and this "database in a JAR" file is bundled into the edge WAR file under the standard WEB-INF/lib directory. The application's data source has a fixed configuration corresponding to the location of the database within the JAR file, and the application and corresponding configuration are deployed in the WAR to the edge servers.

By using Cloudscape on the edge, even the Model components of an MVC application can be distributed onto the edge application platform. Edge applications can make use of JavaBeans and JDBC as Model components with Cloudscape as the DBMS to further reduce communication to the back-end enterprise systems.

One disadvantage of this model is that any time the application data changes, the application must be redeployed to the edge network. Future integration development between IBM and Akamai will enable Cloudscape to function as a database

query cache at the edge of the network. This will enable dynamic caching of query results depending on the SQL statements issued by the edge application.

Table 1 outlines suggested best practices given the type of data access required by an application running on the edge.

Edge Application Examples

The following examples describe some applications modeled to run on EdgeComputing and illustrate the use of WebSphere Web services and Cloudscape in distributed edge applications.

- **Product catalog:** A product catalog browsing application can run almost entirely in the edge environment. Since most product catalogs consist of relatively static product data (not including inventory information), the edge application can utilize Cloudscape as the local DBMS. The data can be bundled into the edge WAR along with the catalog-browsing presentation components. Using this deployment model, it is feasible for the end user browsing interaction to be handled entirely by the edge application. When a user is ready to purchase any selected items, the edge application tunnels back to the enterprise for order processing.

- **Marketing promotional contest:** Say an enterprise wants to conduct a large-scale marketing promotion to give away a certain new product. Because of the uncertainty of the number of end users (contestants), an on-demand edge application is extremely beneficial to assuring a successful outcome. In this scenario, the application might have "random selection" logic to determine if an end user is a winner.

An application can be designed and developed to execute this logic on the edge, offloading the load from the enterprise. In addition, the corporate marketing team can implement various controls on how long the contest runs, how many products are given out, the rate at which they are disbursed, or other controls. The edge application executes the corresponding business logic entire-

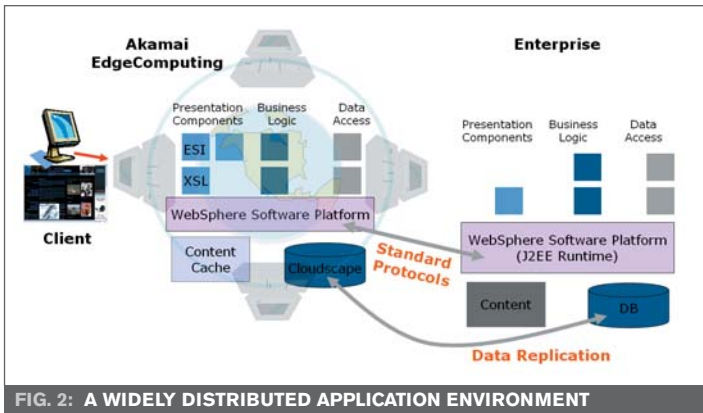


FIG. 2: A WIDELY DISTRIBUTED APPLICATION ENVIRONMENT

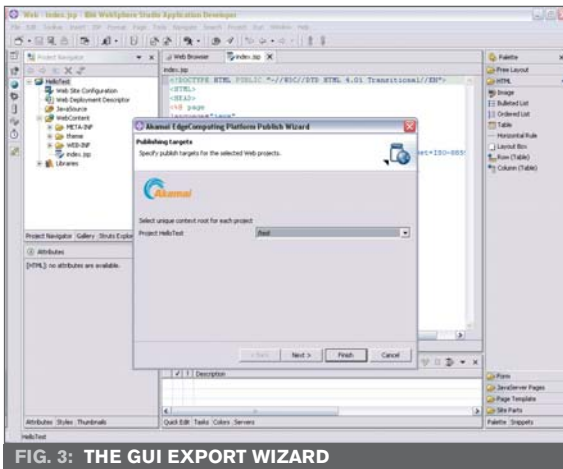


FIG. 3: THE GUI EXPORT WIZARD

ly on the edge and retrieves the control parameters from the enterprise via Web services calls.


Application Deployment to the Edge Platform

Once designed, developed, and tested, the edge subapplication is uploaded and provisioned on the edge network. Akamai has developed a provisioning mechanism that is available via two interfaces. The conventional interface involves using a browser-based GUI on the Akamai enterprise portal. A second interface involves uploading and provisioning via SOAP-based Web services that can be invoked programmatically.

IBM and Akamai have developed a plug-in for WebSphere Studio Application Developer (WSAD) in the form of a GUI export wizard, shown in Figure 3. This plug-in securely invokes the EdgeComputing provisioning Web service, and it allows developers to deploy their applications directly from the WSAD environment. In addition to the upload function, the plug-in is being enhanced to provide validation and a simulated unit edge test environment for the WSAD environment.

Summary

IBM and Akamai have developed a new deployment model for J2EE Web applications. The Akamai Edge-Computing powered by WebSphere service allows application components to migrate to the edge. By extending the WebSphere programming platform to run on the Akamai edge network, enterprises are able

to attain higher levels of performance, reliability, and global reach and ultimately achieve scalability on demand. Akamai EdgeComputing powered by WebSphere helps enterprises respond to a growing need for “on-demand” computing and will help many enterprises adapt their business functions to the edge. 

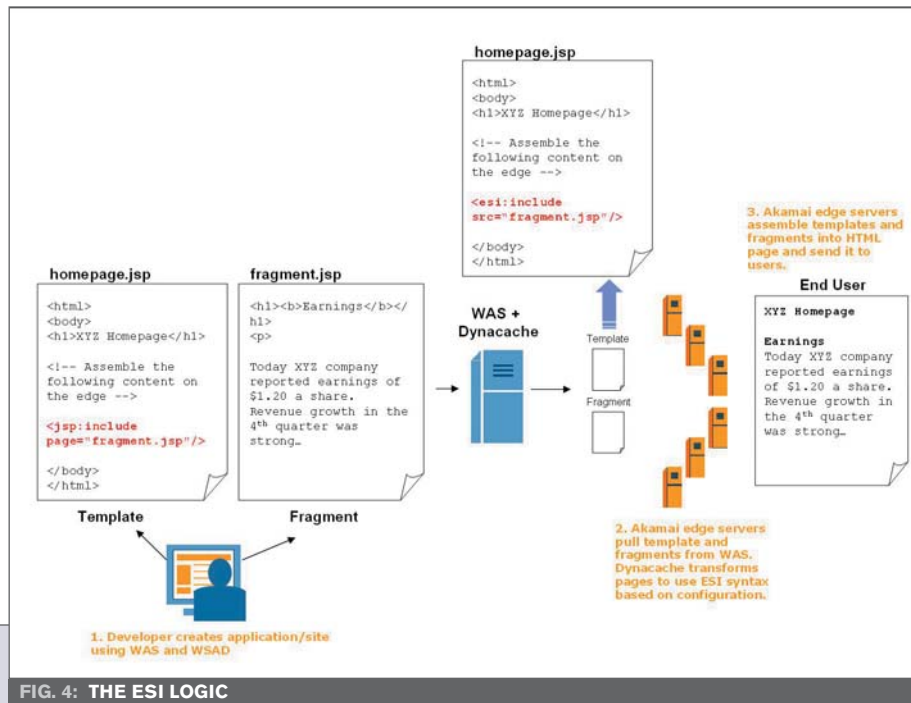


FIG. 4: THE ESI LOGIC

EDGE SIDE INCLUDES

Edge Side Includes (ESI) is a simple XML-like markup language that can be used to assemble dynamic, personalized Web pages at the edge of the Internet. Using ESI can dramatically improve the delivery of dynamic Web content. ESI content assembly is based on the fact that a dynamic Web page is composed of several smaller individual pieces, or fragments. Each of the individual fragments can have its own cacheability properties. An edge server processes the ESI logic and will fetch only noncacheable or expired fragments from the origin Web site. This model of edge page assembly can significantly reduce the load on the enterprise infrastructure, as well as the amount of data that must be transported to the edge.

IBM's WebSphere Dynamic Caching Service component, as part of WebSphere Application Server, supports the ESI specification and seamlessly integrates with Akamai's ESI processing capability. A page composed from a collection of servlets/JSPs (e.g., a portal-style page) can be assembled on the Akamai edge servers from either cached or uncacheable components. JSP/servlet results can be defined as “edgeable,” using the WebSphere assembly tool or by directly editing the WebSphere Dynamic Caching Service configuration file, cachespec.xml. The ESI logic is illustrated in Figure 4.

Upon receiving an end-user request, an Akamai server contacts the enterprise WAS server. It adds an identification HTTP request header in the form of: Surrogate-Capability: akam="ESI/1.0". Upon processing this header, WAS returns content containing esi:include tags instead of a fully built Web page. The embedded ESI tags are processed by the Akamai edge server and the appropriate fragments are fetched (from the enterprise or from cache) to build the response to the end user.

The following code snippets show how a main, or template page, homepage.jsp, could call and insert the fragment, fragment.jsp.

homepage.jsp

```

<html>
<body>
<h1>XYZ Homepage</h1>

<!-- Assemble the following content on the edge -->

<jsp:include page="frag.jsp"/>

</body>
</html>

```

fragment.jsp

```

<h1><b>Earnings</b></h1>
<p>
Today XYZ company reported earnings of $1.20 a share. Revenue growth in the 4th quarter was strong..

```

WAS can invoke an Akamai Web services interface to force an invalidation of any templates or fragments. Additionally, IBM and Akamai are developing a mechanism for WebSphere cache configuration settings defined in the enterprise's WebSphere Dynamic Caching Service cachespec.xml to be automatically set in Akamai's customer cache configuration metadata.