TRZ What's Needed to Run BBj?

Any system purchased 'off-the-shelf' today is powerful enough to run BBj® and its features. That dusty Windows XP in the corner of your office likely has enough horsepower to run any thin client application developed in BBj. It is probably even powerful enough to develop apps in the BASIS IDE or the Barista® Application Framework. But ask our sales or support teams what the system requirements are for a full application deployment and they will tell you the same thing: "*It depends*."

The fact is, there are simply too many variables involved in a BBj application deployment for our sales or support teams to provide such specific information. This is true of any enterprise level Java- or .NET-based application: Only the application developer and end user truly know the system requirements of a deployment.

The good news is that all the tools necessary to assess the requirements for your deployment are at your fingertips.

Q. *What factors should I consider in a BBj deployment?*

A. You need to consider these important hardware and software factors:

Topology – The ability to distribute BBj database management, interpreter, and client functionality across multiple systems (or 'tiers') means vast varieties of application topologies. You will need to consider the system requirements for all of the tiers involved (see **Figure 1**).

Will your deployment be a **single-tier** CUI application on a UNIX server with terminal emulator clients or will it be a **two-tier** GUI deployment on a Windows Terminal Server farm with RDP clients? The former single-tier UNIX deployment actually requires less memory than a similar PRO/5® deployment, averaging only 2 MB for each additional user. The latter 2-tier Terminal Server deployment could require considerable system resources, depending upon the memory requirements of the application itself and the number of concurrent users. **Three-tier** deployments, where the 'Presentation Layer,' 'Application Layer,' and Data Layer' all reside on separate systems – clients, servers, or clusters of servers can have widely-varying memory, CPU, and disk I/O requirements at each layer. Disk I/O will be more of a factor on a Database Management Server (DBMS) than an Application server; an Application server will have greater memory requirements than a DBMS server.

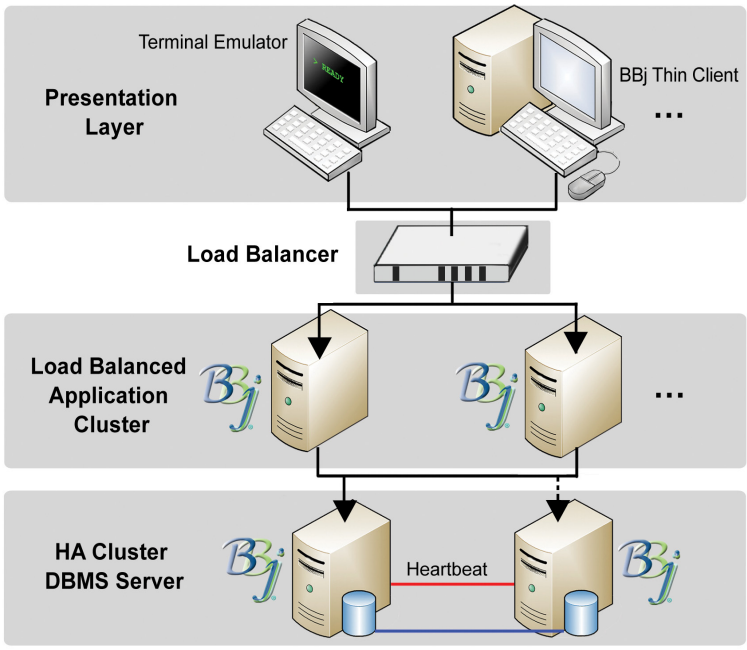

Figure 1. Typical 3-tier BBj deployment in which layer interaction occurs over the network

For an in-depth discussion of different BBj deployment strategies, see *Choices, Choices, Choices* at links.basis.com/05choices.

Application – No two applications use the same amount of resources. A CUI data-entry program would have very low memory overhead. Does your GUI application have a lot of forms or controls? If so, memory requirements on the client side could be a factor.

Number of concurrent users – How many users do you anticipate will log in to your application in the peak hours on a given site? Again, if your deployment is a single-tier CUI application, each additional user will have a minimal impact on system resources.

However, a Terminal Services deployment will be a bit more complex. Here, the 'N+1 Rule' is in effect, where the number of JVMs invoked on the system will equal the number of logged-in users plus one extra JVM for BBj Services powering the DBMS. In Terminal Server/Citrix deployments, there is greater overhead involved in the first invocation of the JVM housing the DBMS, but marginally less memory used for each subsequent invocation. One can only deduce a true picture of memory consumption after adding five or more users. There is no sharing possible in the application server JVM because each user/display must run in its own JVM. Of course, it is important to remember that the maximum memory used by all concurrent users plus BBj Services should not exceed the physical memory of the system.

By Bruce Gardner
Technical Support Supervisor

Virtualization – While it has the potential to reduce cost, you must handle virtualized deployments carefully. Multiple guests competing for resources on a single host server will have a negative impact on disk I/O and introduce network latency.

Network latency – As discussed in the **Topology** paragraph, multi-tier application deployments can place very different demands on systems in the separate layers, depending on their function. However, the distribution of presentation, application, and data functions to different systems necessarily means that the interaction between the layers will be 'over the wire.' Network latency can be the most important factor in multi-tier deployments, and you should pay attention to the latency effects between the layers.

Q. How do I determine the system specifications for my deployment?

A. Match your Test environment to the target Production environment and run it through its paces.

In an ideal world, all testing would occur in a test environment that exactly matches the target production environment. While some developers have done a spectacular job of this, it can be an unrealistic and impractical goal for most. Where it is not possible to match a test environment to a production environment, BBj, system, and third party tools can be used in the test environment to extrapolate a relative and predictive baseline. The closer you match the target production environment operating systems, topology, and workload, the more accurate your assessment. Use a meaningful approximation of the application workload. A tight loop that executes thousands of reads and writes in a minute does NOT approximate user interactive workload!

You will find that the data you collect in this process will inform your n-tier deployment decisions. For example: An interactive data entry application will not be sensitive to network latency in 3-tier deployments, but it might make sense to combine the 'application layer' and 'data layer' into one layer for batch jobs and reports, taking network latency between the application and DBMS servers out of the picture.

Q. What tools should I use?

A. The more, the better! Here are a few examples:

- **BBj tools** – BBj's Enterprise Manager can provide a great deal of information about the server deployment.
  - **Memory Usage module** – Provides real-time graph of current memory usage. You may also view your customer's memory usage simply by pointing the module to a copy of their BBj logs.
  - **System Logs module** – Provides detailed, minute-by-minute, information on current memory usage and Garbage Collection durations. This information can help 'tune' memory-related Java arguments for BBj Services.
  - **BBj Processes module** – Provides a detailed view of the BBj processes currently running.
  - **BBj File System module** – Provides a detailed view of all files currently open on the server.
- **System tools** – Know your target operating system and the tools that come with it. Utilities, such as 'top' on UNIX and 'Task Manager' on Windows, can provide a great deal of information about CPU, memory usage, and disk I/O. Once your test environment is in-place, incrementally add users and watch the resource utilization on the server. After a number of users have been added with a simulated workload, you should be able extrapolate how much CPU, memory, disk I/O, etc. will be utilized during peak production hours.
- **Third party tools** – Third party tools, such as Microsoft's free Sysinternals, can provide detailed information for just about any metric you care to measure on a Windows system. Other third party tools can help simulate a production environment by adding a non-BBj workload or simulating network latency.
- **Test programs** – It is not always possible to approximate a realistic workload on a server but you can always write programs that will: Simulate an interactive workload by spawning BBj processes, or write a program that performs thousands of file operations to simulate a batch process.

Q. Should I re-evaluate system requirements as newer versions of BBj and Java are released?

A. Yes. Java and BBj are constantly changing. New Java/BBj features might require more or less resources. Optimizations are always forthcoming as well; Java introduced dynamic memory allocation in Java 1.6.0_18, taking much of the guesswork out of memory-related Java arguments for BBj Services. Recent filesystem refactoring in BBj 12.10 have resulted in much faster concurrent file access.

Summary

There is no substitute for a real-world test that simulates the actual behavior of your BBj deployment. Only you, the developer, can truly determine the specific needs of your application. A full understanding of the dynamic factors involved in all deployments, coupled with accurate testing and use of tools, will go a long way towards determining those requirements. ■

- Read the BASIS Advantage article *Choices, Choices, Choices* at links.basis.com/05choices
- Check out Microsoft's free third party Sysinternals tools at links.basis.com/iydfu