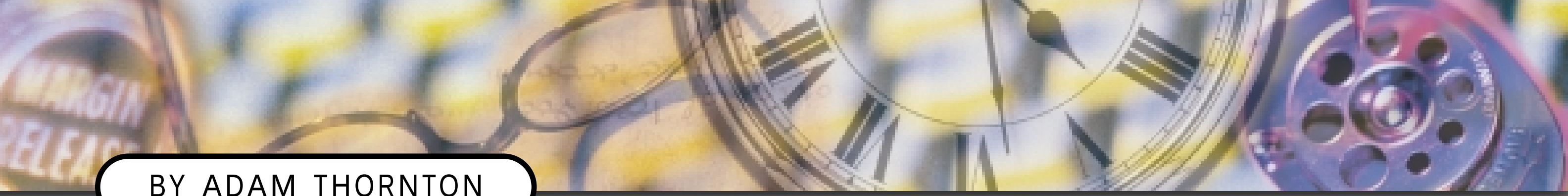


BY ADAM THORNTON

Linux Scalability Testing: Part I — The "Test Plan Charlie" Scenario

This article describes the content and construction of a test plan for implementing a World Wide Web (WWW) virtual server farm and supporting Linux for S/390 infrastructure for a large telecommunications company.

THE release of Linux for System/390, its capabilities, and the subsequent development of tools for enterprise Linux appear to have not only captured the attention of the mainframe community, but also spurred the imagination of a large number of individuals from less traditional large-scale environments. Internet service providers (ISPs) and the fledgling development of pay-per-use application vendors (ASPs) seem to be dead-set on reviving the model of centralized computing and timesharing that was prevalent in the early days of computing. One of the more interesting recent developments entailed a large-scale test case that combined the new Linux technology with tried-and-true techniques from the traditional IBM operating system toolbox in order to develop a large-scale server farm. This server farm was capable of containing tens of thousands of Linux instances: 41,000 individual instances within a single System/390 CPU complex, to be precise.

This article is the first of a two-part series describing the technology and the economics of the concept of virtual servers. Both articles will demonstrate how the mix of traditional IBM technology and cutting-edge Internet services produce an "Internet data center in three racks or less."

The test, designed to be executed in several stages, was carefully constructed to demonstrate several specific goals: application compatibility, system reliability, and ultimately, scalability of the environment.

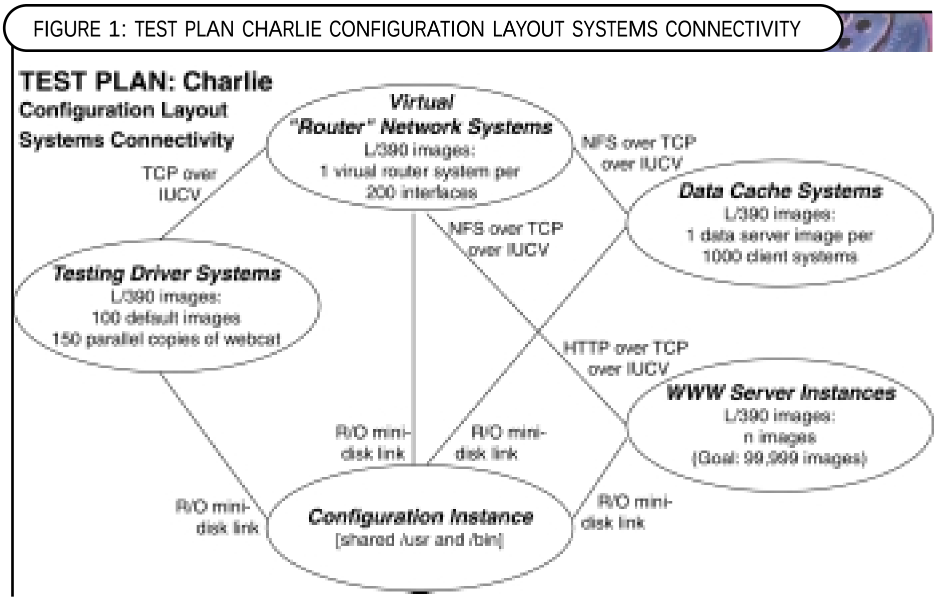
ISP INDUSTRY INFORMATION

To illustrate the solution, as well as the parameters of the problems addressed and resolved, some background information is necessary. The customer was a large telecommunications company who wanted to offer a managed Internet router service to business customers. The company envisioned a soup-to-nuts service providing both connectivity and support services so that its customers, the end users, would quickly be able to get a business connected and functioning. The initial end-user base was estimated at 250 customers, and the initial basic support services offered were domain name to IP address translation (the Internet domain name service, or DNS), along with some additional value-added services (representing business collaboration and information sharing based on standard Internet applications).

In the ISP industry, these services are deployed and managed by utilizing standard software tools, regardless of the platform on which the services are to be delivered. Whether on Sun, Hewlett-Packard (HP), IBM or Intel-based systems, the software applications originated in university settings but are now managed by the open source community. In the test case outlined here, the UC Berkeley DNS server and the community-developed Usenet News tool INN were adopted. Nearly all major ISPs use these open source applications as tools to ensure compatibility and interoperability with customer systems.

While most ISPs begin with a similar suite of Open Source network tools, thereby giving them similar starting conditions, their actual implementation needs may vary significantly. To successfully compete within the ISP industry, fast deployment is crucial — the phrase "living on Internet time" is especially significant here. Time to market for a service constitutes the primary differentiator in this industry. The second major dif-

FIGURE 1: TEST PLAN CHARLIE CONFIGURATION LAYOUT SYSTEMS CONNECTIVITY



ferentiator, however, is the cost of providing and provisioning the services. Together, these two factors separate a successful ISP from yet another “has-been” attempt. (Part II will examine the economics of this process in more detail and makes a compelling in-depth business case for the advantages of Linux for System/390 for large and very large virtual server farms.)

DEVELOPMENT OF THE PLAN

After having contacted a Big Three consulting firm for a design and infrastructure quotation, the telecommunications company in question was staggered by the sheer size — and price tag — of the proposed solution. The original design proposal can be summarized as follows: To provide guaranteed quality of service for each customer, the consulting firm had recommended a solution entailing three physical servers per customer, with associated racks, floor space, deployment and management systems to control an environment of more than 750 separate systems to support the initial deployment. Consequently, the price tag exceeded $30,000 USD per customer for server equipment alone! While the telecommunications company was prepared to build this service, it asked for a second opinion. David Boyes, a Virginia telecommunications and network engineer, had been working with IBM and others on the Linux for System/390 development. When consulted, he performed the usual review and development of a discrete system plan and offered the System/390 solution as an interesting alternative.

The telecommunications company expressed an interest in Boyes’ alternative plan, but was concerned with customer perception, since the Linux for System/390 port was relatively new. Another major concern was scalability of the solution to support large numbers of users. Still another hesitation centered around issues of appropriate context; although the telecommunications company representatives were familiar with System/390 as a billing and transaction server, they were unsure about using it as an Internet-based server infrastructure. Thus, Boyes proposed a test intended both to dispel any doubts and to establish the boundaries of the system solution. After having received permission to utilize the company’s existing System/390, Boyes began to implement his test case.

GOALS AND CONSTRAINTS

The test, designed to be executed in several stages, was carefully constructed to demonstrate several specific goals: application compatibility, system reliability, and ultimately, scalability of the environment. The company, however, had set the following constraints on the testing: First, the test must represent a real-world application environment; second, rather than idling systems, the test must constitute a real end-to-end application demonstrating both working programs and manageability of the solution.

The diagram in Figure 1 describes the relationship and construction of the test. Boyes selected a simple application — presenting a static page via the Apache web server — as a good test case that, for documentation purposes, could be quickly constructed and instrumented. The LPAR available to Boyes consisted of two CPUs from a G5-class System/390 along with 128MB of central storage and a recently acquired EMC disk unit that had not yet been placed into service and was available to be dedicated to the LPAR for testing purposes. Boyes installed VM/ESA in the LPAR, IBM’s directory maintenance utility, and VM’s TCP/IP communications stack to provide external network connectivity as well as the fabric for the internal communication network. He then commenced installation and testing of the Linux environment.

TEST PLANS ABEL, BAKER, AND CHARLIE

The tests were constructed in stages in order to verify that the original Marist distribution of Linux could be configured efficiently for this use. A VM userid was defined with separate minidisks each containing a major filesystem of a complete Linux operating system. This allowed the common utilities and operating system code to be shared between Linux instances using a VM read-only minidisk link for each virtual system. Next, Boyes created four duplicates of this master system and began optimizing and porting the applications required for the test.

All systems (as shown in Figure 1) shared the same common operating system, yet each system served a specific function, and thus was configured and optimized for one of four different purposes. In the test, Boyes needed four classes of systems: network routers to broker connections for the WWW servers, a common location to store files and data used by all the servers, the actual WWW servers themselves, and some method for generating a test load on the environment. The labels in Figure 1 indicate the function of the instances: virtual routers to provide IP connectivity and routing functions; the WWW servers; an NFS-based file server; and test load generators (each running a script making 150 parallel requests to the web server farm per minute and recording the number of seconds to retrieve the sample WWW page as the measure of performance).

Each class of systems was created from the same core Linux distribution, and each class used Open Source tools and applications to accomplish its job. All application source code was downloaded from the standard

Internet locations, and no modifications were made to the source — the porting process was, literally, the process of compiling and installing the applications in the same manner as other platforms. The telecommunications company's Unix staff performed this portion of the project to ensure the compatibility and manageability of the Unix applications.

During this phase, Boyes also developed some short REXX execs to duplicate and customize a Linux instance. He discovered that creating and configuring a new Linux instance from one of the master copies involved no more than two commands and about 90 seconds to duplicate and configure the system. This code was later used as the core of the production solution.

Once the master configurations were set, Boyes designed three sequential plans to test scalability: Test Plan Abel managed the original 250 users, while Baker, starting with 2,750 users, satisfactorily served as many as 10,000 users. These tests were performed over a few evenings, and both Abel and Baker performed smoothly for simulated loads of more than 6,000 hits per minute for the server farm. A critical feature of the test plans was their automation. That is, a few simple scripts sufficed to create each set of machines, and scaling the testing was simply a matter of adjusting the bounds governing how many resources the plans could allocate.

Finally, Test Plan Charlie, the "let's push it until it falls apart" test, was created to gauge the upper limits of the solution. Charlie began at 5 p.m. on a Friday; by midnight Saturday, it had reached 41,400 servers and it had run out of resources on the System/390 LPAR. While the system did not crash, it was unable to create new servers due to lack of resources.

Note: While the test described here was focused on the ISP environment, this type of configuration is very similar to a typical enterprise file and print server, or other distributed service configurations. In future articles, additional possibilities of this configuration in the enterprise environment will be discussed and demonstrated.

WHAT WAS LEARNED

As one might imagine, testing of this magnitude exposes a number of things that do not appear in smaller environments. In the process of Test Plan Charlie, Boyes drew a number of interesting conclusions:

- There is a significant benefit from running this environment under VM/ESA as a set of guest systems. The Linux instances inherit the management tools developed for other VM guest systems such as VSE, and, using the standard VM tools, the Linux instances can be directly instrumented and measured. This allows for detailed capacity planning and system performance measurements impossible with standalone Unix systems.

Among VM's design goals is the ability to support 100,000 simultaneous virtual machines. Is it possible to reach that number, or an even higher one, with Linux as the guest operating system?

- SLA management and resource utilization information can be not only extracted easily but also billed directly to a specific customer. This is a very attractive capability for ISPs, as most IP infrastructures do not offer an opportunity to collect detailed usage information without significant expense or performance impact.
- Console management and system control can be performed from the VM programmable operator function; this is crucial for controlling the speed of deployment (facilitating an extremely short time-to-market) as well as the management costs of the complex (lowering total-cost-of-ownership).
- Applications written for most popular Unix systems and Linux for Intel systems are directly source compatible. Therefore, these applications can be configured and managed by staff trained to manage systems on other platforms.

VISIONS AND DREAMS

Where do Linux on the System/390 and the Virtual Server Farm go from here? Several avenues appear to open up. Test Plan Charlie tested the limits of Linux scalability on an LPAR that represented a portion of a fairly large machine. How many instances could be created on a ZZ7 or similarly large machine fully available for Linux use? Among VM's design goals is the ability to support 100,000 simultaneous virtual machines. Is it possible to reach that number, or an even higher one, with Linux as the guest operating system?

The critical factor for this limit is the System/390's 31-bit address space. All control blocks for all virtual machines must fit into the bottom 2GB of system memory. However, on October 3, IBM announced its new eServer zSeries, which will be available on December 18 of this year. With a 64-bit address space and 64-bit VM, how many virtual machines will it be possible to create?

How about running virtual machines that are not based on the System/390 architecture? Linux supports a wide range of emulators: everything from the 68000-based family of Macintoshes to Amigas, Apple IIs, and Nintendo Game Boys. All of these emulators will run under Linux/390. Most fascinating, however, is Bochs or Plex86: an Intel x86 emulator. Bochs emulates a 486-based Intel system well enough to run NT Server. Believe it or not, it is possible to run NT and Microsoft Exchange on your System/390 right now. It runs very slowly, but it works. This is obviously not a viable large-scale solution at this time (Intel machines are much cheaper than emulating an Intel system on a System/390 machine). However, if a requirement for a few critical NT applications (in a mostly Linux workload) did in fact exist and performance was not paramount, the inefficiency of emulation might be tolerable in order to avoid the requirement for a separate physical installation.

Finally, VM contains strong clustering support that Linux can exploit when running in a virtual machine. An idea that has been around for a while but has recently been revived and invigorated in the Linux context is that of global clustering. There is no *a priori* reason that clustering could not take place among geographically separated systems. Imagine a mechanism to freeze a virtual machine's state and control information, transmit that information to another CPU (potentially to another continent), thaw the image, page it back into main storage, and schedule it normally. If this mechanism can be developed, it could be used to provide unparalleled failover and fast recovery techniques. A catastrophic data center failure typically means a lengthy and expensive outage as the systems are restored from offsite backups. With global clustering, however, the only sign of failure visible to the customer would be a brief hiccup in network connectivity. Less dramatically, but perhaps even more importantly from an economic perspective, global clustering would benefit the ASP market by allowing automatic provisioning of applications from the machine the shortest distance from the client system, as measured by latency, hop count, or some similar network metric.

PREVIEW

Testing a facility of this size is a significant technical achievement. Part II will continue and shed light onto the business reasons behind the test. This business case will demonstrate the economic advantages of the solution tested in Charlie by offering a detailed account of the argument that convinced the telecommunications company mentioned to go ahead and take the Linux for System/390 solution to a production environment. The strength of this solution, as it can be successfully implemented in large enterprises, will emerge with a clarity that enthralls engineers and managers alike.

Adam Thornton was first exposed to the System/390 as David Boyes' junior systems programmer at Rice University in the early 1990s. Since then he has spent nearly a decade in the penguin's clutches, having used Linux since version 0.09. Subsequent to graduate studies at Princeton, he volunteered as system administrator for penguinvm.princeton.edu, Princeton University's Linux/390 virtual machine. Recently, he became a founder and principal engineer of Sine Nomine Associates. Adam can be contacted via email at adam@sinenomine.net.