



Virtual Lab Automation

A Quantum Leap in IT Cost Reduction and Application Development Process Improvement

an akimbi whitepaper



Virtual Lab Automation

CONTENTS

Executive Summary	3
The Growing Impact of Distributed Architectures on SDLC Processes and Infrastructure	4
Virtual Lab Automation	5
Virtual Lab Automation System Architecture	6
Virtual Lab Automation System Operation	7
<i>Checking Out a Configuration from the Library</i>	7
<i>Capturing a Configuration to the Library</i>	8
Virtual Lab Automation System Benefits	9
Conclusion	12
About Akimbi Systems	13

EXECUTIVE SUMMARY

Two trends in enterprise software development – the accelerating adoption of distributed application architectures (the service-oriented architecture, or SOA, being the state of the art approach) and the outsourcing of software development activities – are undermining the effectiveness and efficiency of prevailing enterprise software development lifecycle (SDLC) processes and the infrastructure supporting these processes. Across the board, enterprise software development organizations are grappling with:

- **Server Sprawl**

Organizations face an explosion in the number of machines required to develop and test enterprise applications, with some application development (AD) organizations reaching server- to-staff ratios of greater than 7:1, even though average server utilization rates are often below 10%. Servers are hoarded under desks and duplicated across underutilized labs housed in data centers that are short on space, power and cooling capacity.

- **Setup and Provisioning Overhead**

An enormous amount of time is wasted on repetitive system setup, provisioning and configuration tasks, done in preparation for software development and test activities. These tasks often account for more than 50% of the total time expended in an application development and test cycle.

- **Costly System Failures**

Difficulties reproducing, diagnosing and correcting software defects discovered in remote development facilities, or by outsourcing partners, are leading to serious system failures in production, when the cost to repair can be over 470x higher than if resolved earlier in the AD process. [Baziuk 1995]

“When testing software, a major challenge for G2000 organizations is the setup, configuration and management of systems on which the tests will be run. Departmental groups tend to hoard their configured systems, exacerbating existing testing bottlenecks. Facilitating the setup, management and coordination of testing systems can increase productivity, and cut costs and time to production release for software.”

Melinda Ballou
Program Director, IDC Research
Application Lifecycle Management

Virtual Lab Automation (VLA) substantially mitigates these problems – reducing server-to-staff ratios by more than 75%; slashing the percentage of AD cycle time spent configuring systems from 50% (or greater), to less than 5%; and ensuring software defects can be rapidly and consistently reproduced and resolved early in the development cycle. The result is higher quality software, built faster and with lower server- and data center-related capital and operating costs.

A Virtual Lab Automation System automates the setup and teardown of complex, multi-machine software configurations on a centralized pool of servers shared by the application development and QA teams in an enterprise. These operations are performed in a self-service manner by developers and QA engineers, relieving the tedious provisioning burden often shouldered by the IT organization.

Enabled by the emergence of reliable and high-performance virtual machine technology from vendors such as VMware and Microsoft, a VLA System allows development and QA professionals to suspend, then capture to a shared storage library, the complete state of a “complex configuration” – a collection of running, interdependent software systems usually spanning multiple servers. Over time, an organization builds up its configuration library, including test scenarios, configurations exhibiting software defects, historical build archives, replicated production environments and customer configurations.

When a configuration in the library is needed for development or test purposes, the VLA System can instantly deploy the entire multi-machine configuration to the best available resources in the pool of shared servers, exactly as they were captured – running and ready for use, including installed operating systems, applications and data.

VLA-managed capture and restore operations literally take seconds. What would normally be a painstaking, multi-hour or multi-day IT provisioning exercise (gathering machines, installing operating systems, installing and configuring applications, establishing inter-machine connections, booting) can now be done in seconds by a developer or QA engineer with a click of the mouse or via a single API request from any other software system, including test automation systems.

THE GROWING IMPACT OF DISTRIBUTED ENTERPRISE SOFTWARE ARCHITECTURES ON SDLC PROCESSES AND INFRASTRUCTURE

Enterprise software architecture is dramatically different today than architecture considered state of the art just a few years ago. But despite the contrast between then and now, the transition has been gradual; characterized by a progressive move away from “monolithic” software system design toward designs that are increasingly distributed in nature.



But as the benefits of this transition have accumulated, a number of challenges have emerged. These challenges threaten continued progress:

- **Server Sprawl**
Organizations face an explosion in the number of machines required to develop and test enterprise applications, with some application development (AD) organizations reaching server- to-staff ratios of greater than 7:1, even though average server utilization rates are often below 10%. Servers are hoarded under desks and duplicated across underutilized labs housed in data centers that are short on space, power and cooling capacity.
- **Setup and Provisioning Overhead**
An enormous amount of time is wasted on repetitive system setup, provisioning and configuration tasks, done in preparation for software development and test activities. These tasks often account for more than 50% of the total time expended in an application development and test cycle.
- **Costly System Failures**
Difficulties reproducing, diagnosing and correcting software defects discovered in remote development facilities, or by outsourcing partners, are leading to serious system failures in production, when the cost to repair can be over 470x higher than if resolved earlier in the AD process. [Baziuk 1995]

Ensuring the quality of a new software system requires that it be built and tested in the context in which it will (or could) eventually be deployed, thereby ensuring it will perform as expected when interacting with the systems that will surround, support and depend on it.

Consider a new software system in a typical modern enterprise computing environment. It will likely be expected to:

- render correctly in a variety of browsers, or support a range of client-side operating systems;
- run on an assortment of application servers;
- interact with database systems;
- integrate with directory servers, mail systems, single sign-on servers, firewalls, VPNs, network infrastructure components, domain controllers and enterprise management systems;
- and reliably pull from and push data into any number of line-of-business applications – including packaged and custom-built applications, and applications delivered as a service.

Notwithstanding the tremendous progress in efforts to conform interactions between software components through standardized APIs and network protocols, simply ensuring components can “talk” to each other does not mean the entire system will behave as required, or perform as expected, once deployed. Application behavior depends on how its constituent components process and respond to messages, the order of message arrival, message timing and a host of other variables. Ensuring the quality of a distributed application requires testing “in context.” And the earlier in the development process this occurs, the lower the total cost to produce and deploy high-quality software.

The traditional approach has been to “throw more servers at the problem” and to enlist the IT organization for the setup, provisioning and interconnection of those servers as needed by application development and test teams in the enterprise. Servers are requisitioned; new development test labs are built and existing labs expanded; developers, QA professionals and IT support personnel spend nights and weekends performing menial configuration tasks. This works for a while.

But as organizations move further to the right on the distributed computing continuum, the effectiveness and efficiency of these tactics rapidly break down. Unfortunately, the breakdown sneaks up on an organization and at some point becomes a crisis, whether or not it is recognized as such. Data centers are reaching capacity. And being asked to wait days, or longer, for IT to complete a provisioning request does not make for rapid application development cycles. The ultimate outcome, if these problems are not positively addressed, is an IT organization unable to rapidly, or cost-effectively, respond to the changing requirements of the enterprise through IT system enhancement.

VIRTUAL LAB AUTOMATION

Virtual Lab Automation (VLA) substantially mitigates the previously highlighted problems – reducing server-to-staff ratios by more than 75%; slashing the percentage of AD cycle time spent configuring systems from 50% (or greater), to less than 5%; and ensuring software defects can be rapidly and consistently reproduced and resolved early in the development cycle. The result is higher quality software, built faster and with lower server- and data center-related capital and operating costs.

For example: What if you need access to a configuration of five interconnected servers, in a known clean state, including a Windows XP client machine, a WebLogic Server instance running on Red Hat Linux, an Oracle9i Database, an Active Directory domain controller and a single sign on server?

Today this can require a painstaking, multi-hour IT exercise including gathering machines, installing operating systems, installing and configuring applications, establishing inter-machine connections, loading data and rebooting each server multiple times.

“For many organizations, especially large organizations, establishing realistic testing environments has become a time consuming chore. It is not done well and sometimes not done at all. The unfortunate result is that many systems end up going live that shouldn’t.”

Robin Bloor
Founder and Chief Research Officer
Bloor Research

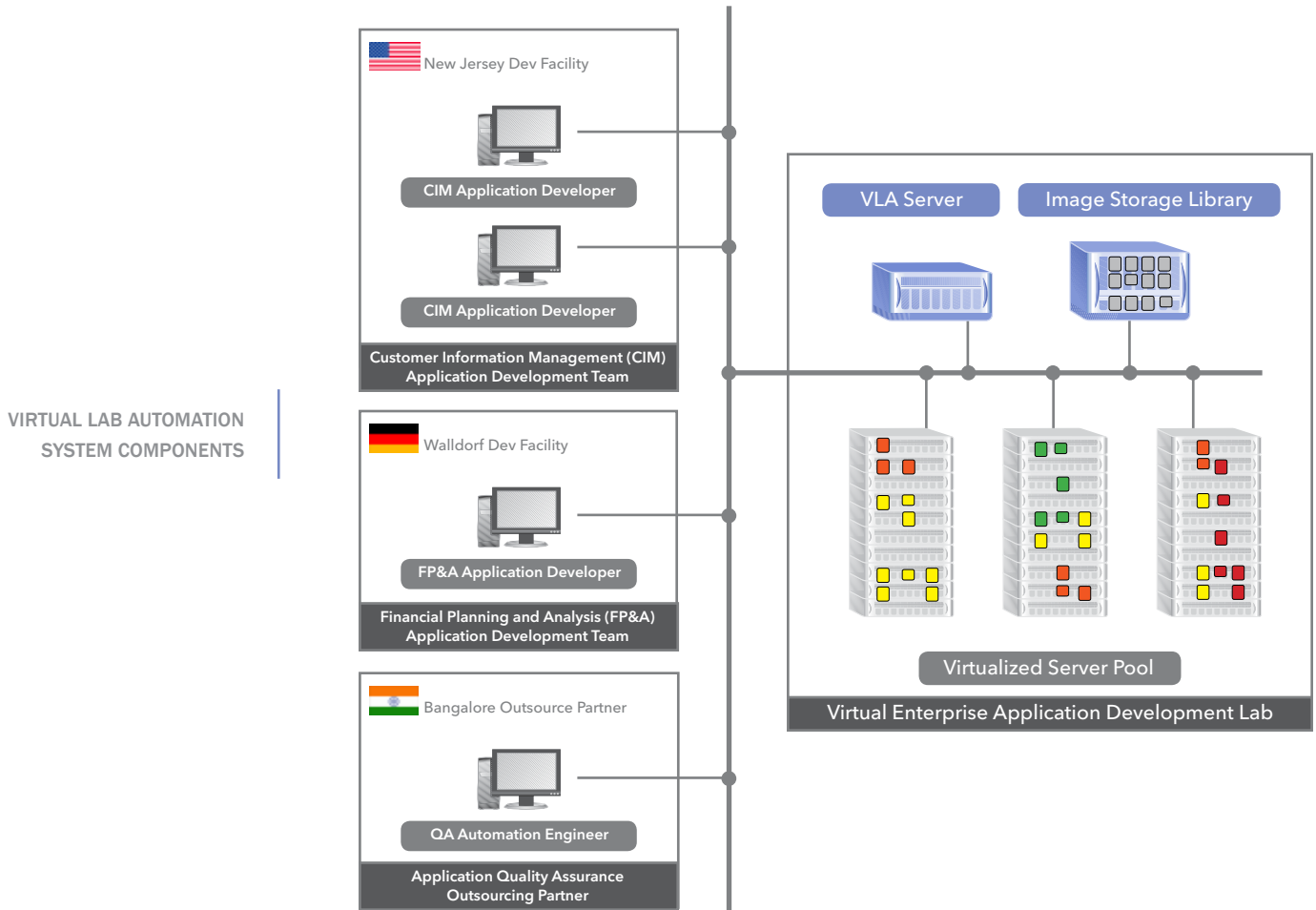
In an automated virtual lab, this same multi-machine configuration can be made available for use, in seconds, with a single click of the mouse by an AD user in a self-service user interface.

VIRTUAL LAB AUTOMATION SYSTEM ARCHITECTURE

Enabled by the emergence of reliable virtual machine technology from vendors such as VMware and Microsoft, a Virtual Lab Automation System allows application developers and QA professionals to suspend, then capture to a shared storage library, the complete state of a “complex configuration” – a collection of running, interdependent software systems that usually span multiple servers. Over time, an organization builds up its library, including test scenarios, configurations exhibiting software defects, historical build archives, replicated production environments and customer configurations.

When a configuration in the library is later needed for development or test purposes, a VLA System can instantly deploy the entire configuration to the best available resources in a pool of shared servers, exactly as captured – running and ready for use.

The following diagram highlights the components of a Virtual Lab Automation System.



Virtualized Server Pool. The most visible component of a VLA System is the central collection of virtualized servers and other supporting systems on which configurations are deployed. While resources are securely shared by users of the VLA System, users are shielded from this fact. From the perspective of a user, it appears as though there is an unlimited pool of resources available and instantly configurable on demand (though typically within the bounds of administrator-controlled limits).

VLA Server and Image Storage Library. The VLA server directs the capture, storage, movement, management and restoration of multi-machine configurations. Each machine image in a configuration contains the complete suspended state of machine CPUs, memory and disks. The software development and test environment is unique in its diversity and quantity of configurations, and VLA configuration libraries often grow to occupy many terabytes of storage. Efficient storage compaction algorithms and library management tools are a critical part of any VLA System.

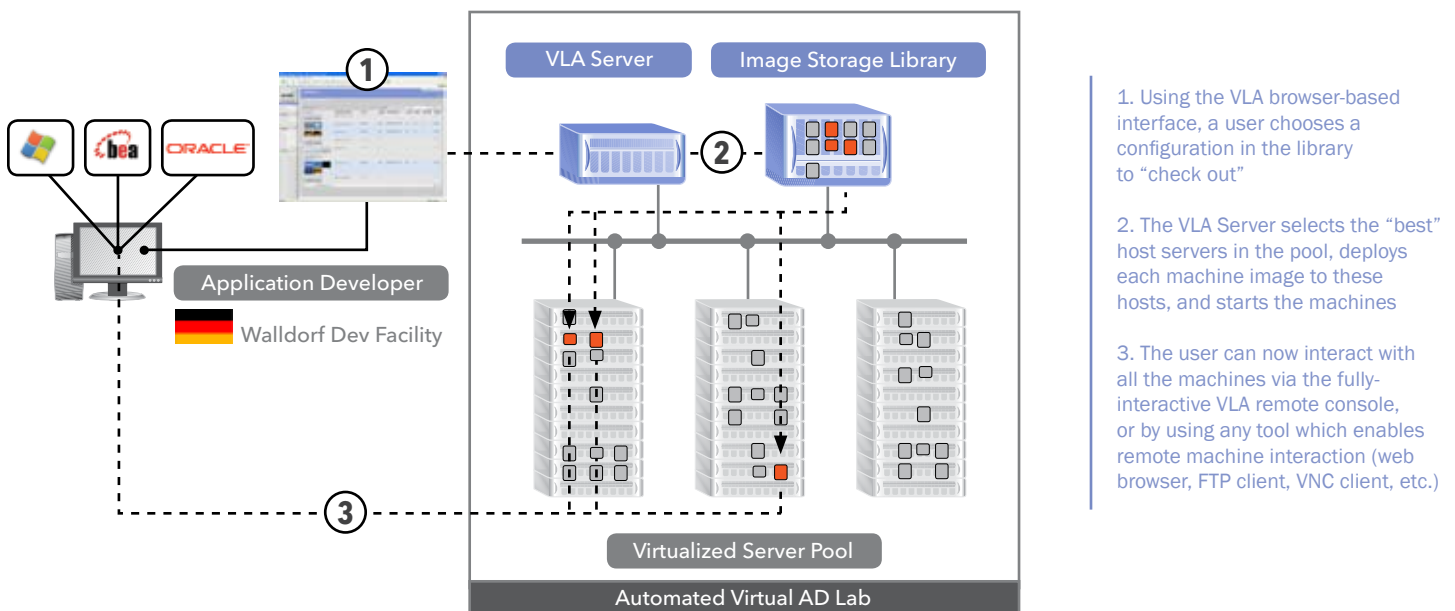
VLA System Users. Application developers and QA engineers request the deployment and capture of multi-machine configurations. Most commercial VLA Systems provide both user and programmatic (API) interfaces. API interfaces make it possible to interact with a VLA server directly from test scripts and test management systems, enabling automation of what is today often a manual or semi-manual step in an otherwise automated test process.

VIRTUAL LAB AUTOMATION SYSTEM OPERATION

From a user's perspective, the VLA System is a library of pre-configured, suspended multi-machine configurations that can be "checked out" and used whenever needed. The figures below illustrate its use.

CHECKING OUT A CONFIGURATION FROM THE LIBRARY

In this example suppose the user is a QA engineer at a financial services organization. The user needs to test an updated version of a system for automated trading of small-cap equity securities. The trading logic is contained in a Java application that executes in a WebLogic J2EE container on a Linux-based server. The application requires an Oracle database and is accessed via a Windows Forms application that executes on a Windows XP Pro client.



Using the VLA user interface, typically presented via web browser, the user examines the library of stored configurations and finds the desired configuration, labeled "ATS Production - Equity Small Cap." Highlighting this configuration, the developer selects "check out."

In response to this request, the VLA System initiates a multi-step process to match, transfer and execute the virtual machine images on the "best" hosts in the shared server pool. This operation takes place automatically and without user intervention, or even user awareness as to the details of the process.

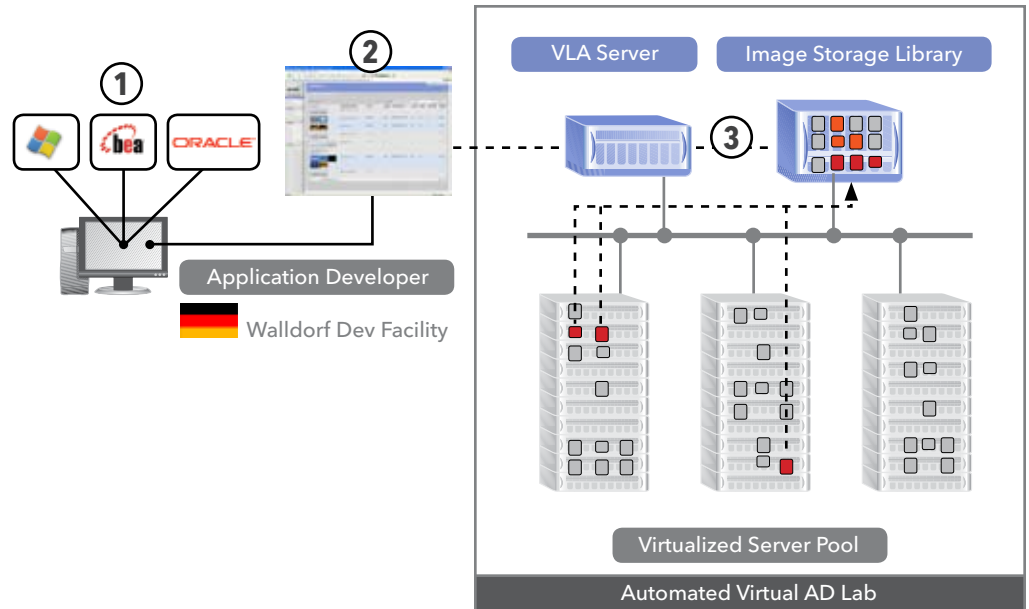
Once the machines are deployed and running (elapsed time to this point is typically less than 45 seconds), the user can copy the updated trading application and client code to the respective servers and begin testing. Interaction with the machines is done in any manner desired, such as through a remote desktop connection, secure shell, test script, debugger, browser or FTP client. Once a configuration is deployed, the VLA System has done its work and is out of the loop, though a VLA System should also provide for direct remote KVM console access from within the VLA user interface itself. Some VLA Systems can present side-by-side remote interactive consoles for all machines in a configuration, in a single browser page.

This example highlighted user-initiated deployment and use of a configuration. This process could just as easily have been performed programmatically from within an automated test script. In the automation case, a single API request from a test management system can initiate a deployment of the configuration. Once deployed, the script is notified and automated testing can begin. When testing completes, the resources are released and made available for others via another API request. A VLA System is optimized for rapid configuration deployment and capture, maximizing the number of unique test configurations through which a test script can iterate per unit of time.

CAPTURING A CONFIGURATION TO THE LIBRARY

In addition to deploying configurations from the library, a VLA System can capture new configurations to the library. Continuing the previous example, suppose the QA engineer has discovered a bug that is easy to replicate, but it requires the machines be in their current state. With a VLA System, the configuration can be suspended and added to the library. It can later be checked out, in exactly the current running state, by a developer who can then reproduce, troubleshoot and correct the defect. And this can all be done from a continent away – lowering some of the communication and collaboration costs inherent in distributed and outsourced application development and test.

1. After using and modifying the machines in the configuration – e.g., deploying and testing a Java application on the BEA Server – the user would like to capture the updated state of the configuration to the library
2. Via the VLA browser-based interface, the user selects the configuration, provides a name for the new library entry, and requests it be captured to the library
3. The configuration is captured to the library where it too is now available for check out at a later time by this user, or by other users, if the configuration is shared.



To perform the capture, the user highlights the configuration in the VLA user interface and selects “capture to library.” After the user provides a new name and description, the VLA server captures and stores the new configuration in the storage library. The configuration can either be a “net new” configuration in the library (most likely in this case) or the original configuration can be overwritten with the new state of machines, subject to security settings and permissions. Like the check out operation, the capture process can be fully automated through an API request from another AD infrastructure component.

Once the capture operation concludes (again, elapsed time should be measured in seconds) the configuration remains deployed and available on the server pool. The user can continue working with it. In this case, testing can continue from this point forward, or the machines can be rapidly “reset” to their initial state, if desired.

Configurations can be captured by a user as often as desired, within administrator-controlled quota limits. For example, a user may create a machine and install an operating system, say Microsoft Windows 2000 Advanced Server. This machine could be captured to the library as “Windows 2000 Advanced Server.” He could then apply Service Pack 1 to the server and once again capture the machine to the library, now as “Windows 2000 Advanced Server SP1.” This process could continue for SP2, SP3 and SP4. The library now contains five configurations (each with a single machine, in this example) representing the major releases of Windows 2000 Advanced Server that are deployed within the enterprise. The user could then install applications and combine servers into multi-machine configurations, capturing every step of the way. In this way, a rich library of configurations is built up for use by anyone in the organization with a VLA System account and appropriate permissions.

VIRTUAL LAB AUTOMATION SYSTEM BENEFITS

A Virtual Lab Automation System delivers measurable value to the application development teams within an enterprise and to the IT organizations that support them. Application developers, QA engineers, IT operations and support professionals all benefit from Virtual Lab Automation adoption. The table below summarizes these benefits.

VLA SYSTEM CAPABILITY	QUANTIFIABLE BENEFITS
Create a centralized pool of virtualized servers, storage and networking equipment shared across software development and test teams and team members	<ul style="list-style-type: none"> • Reduce server-to-staff ratios by over 75% • Slash equipment-related capital and operating expenditures • Accelerate software development cycles
Automatically and rapidly set up and tear down complex, multi-machine software configurations for use in development and test activities	<ul style="list-style-type: none"> • Reduce the percentage of AD cycle time spent configuring systems from 50% or greater, to less than 5% • Increase the number of certified deployment configurations supported by new software systems • Find and fix more bugs earlier in the development cycle
Give every developer or test engineer the equivalent of their own fully-equipped data center with dedicated provisioning staff	<ul style="list-style-type: none"> • Remove the repetitive AD provisioning support burden from IT • Eliminate the time and energy wasted begging for, re-configuring and protecting hoarded servers, storage and networking equipment • Enable real-world unit testing
Maintain a comprehensive library of customer and production system environments	<ul style="list-style-type: none"> • Eliminate upgrade breakage • Deliver better support to users of a software system • Rapidly troubleshoot customer production problems
Suspend and capture “live” multi-machine configurations to a shared library	<ul style="list-style-type: none"> • Reproduce bugs reliably and reduce time spent in the debug phase • Reduce the number of latent software defects that slip into production • Maintain a historical record of builds and test scenarios
Effortlessly move configurations between development and test facilities	<ul style="list-style-type: none"> • Lower the cost of communication between distributed software development teams • Enhance the efficiency and productivity of outsourced software development and test partners

Create a central pool of virtualized servers, storage and networking equipment, shared across software development and test teams and team members. Across the Global 2000, it is estimated that for every server in production, there are between 2 and 5 servers in the application development organizations which build and support the applications running the business.

A productivity metric that has emerged to address this server sprawl is the server-to-staff (STS) ratio, defined as the average number of servers required to support each developer and test engineer. Some organizations have reached STS ratios of 7:1 and higher. With a VLA system, these organizations can attain ratios of less than 2:1, reducing AD server counts by 75% and higher.

A Virtual Lab Automation System makes possible this level of consolidation primarily due to its ability to rapidly and effortlessly return servers to a desired software state, thus enabling the cost-effective sharing of resources. Without a VLA System, the expense of repetitive configuration and provisioning operations outweigh the benefits of sharing.

Automatically and rapidly set up and tear down complex development and test configurations. Regardless of configuration complexity – whether a single server or a dozen interconnected servers – a VLA System can complete the provisioning process and make the machines available in seconds, literally.

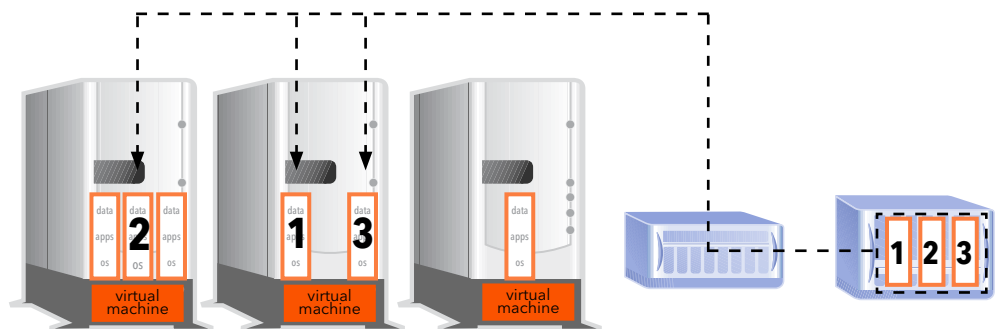
It was noted above that this capability makes efficient sharing of servers possible. But even with servers dedicated to a single individual, there is a constant need to return machines to a known “clean state.” The last build tested may have trashed the Windows Registry, or otherwise left one or more of the machines in an unknown or unstable state.

With a Virtual Lab Automation System, machine states can be instantly returned to a known good state, on demand.

Give every developer or test engineer the equivalent of their own fully-equipped data center with dedicated provisioning staff. In most organizations, economics preclude fulfilling every resource request that could be put to good use, albeit intermittently. When servers or other resources are needed, they are often difficult to come by and time consuming to configure if they can be found and borrowed. And even though equipment is usually sitting idle elsewhere in the organization, it is often unavailable to those in need for reasons summarized above.

With a shared pool of resources and the near instantaneous configuration capabilities of a VLA System, servers can be “borrowed,” configured in seconds and put to use in ways that were never before practical.

The benefits of a VLA System are amplified through the underlying leverage of virtual machine technology. Not only can a pool of servers be shared, but each individual server can be used simultaneously by multiple users, if desired. The diagram below is a slightly more detailed look at the process of “checking out” a configuration when using virtual machine technology in the shared server pool.



In this example the user is checking out a three machine configuration. The machines in this configuration are each stored in the library as virtual machine images. A virtual machine image represents the complete running state of a machine – including processor, memory and disk state. The target servers in the managed server pool are running Virtual Machine Monitor (VMM) software (such as Microsoft Virtual Server 2005 or VMware ESX Server). VMM software can accept, load and execute a virtual machine image resulting in what is, from the user's perspective, indistinguishable from a non-virtualized, fully-configured server being added to the network and made available for his use.

On check out, the VLA server selects the “best” available servers in the server pool to host the virtual machines, based on server loading trends, virtual machine requirements (type of processor needed, amount of memory, etc.) and other considerations. The VLA server communicates with the VMMs on the selected servers, requesting they load and execute the machine images.

Capture multi-system configurations exhibiting a bug or other defect, ensuring reproducibility at debug time. One of the more interesting benefits of a VLA System is its ability to eradicate from the software development lexicon the all too frequently heard words “it works fine on my machine” and “I can't reproduce the problem.”

In the testing phase of a software project, the QA organization may find an anomaly in a specific test configuration. Often the defect is related to the combination of systems in the given test configuration or the current state of the systems in that configuration. The tester files a defect report which is eventually assigned to the appropriate developer. Once the bug report arrives, the developer tries to reproduce the problem in his environment and, more often than not, the problem does not manifest.

At this point the developer can either gather all the systems and software needed to recreate the environment in which the problem was found by the QA organization, or he can challenge the QA organization to show him the problem. Of course, testing has proceeded and systems have been reconfigured and QA is now unable to reproduce the problem either. If the problem is serious enough, the team makes it its mission to reproduce the problem. The amount of time and organizational energy consumed can be very high. But it is nothing in comparison to the cost of not finding the problem now and having it appear in a production or customer scenario.

With a VLA System, an organization need never face this situation again. If a problem is discovered, the VLA System's configuration capture capability enables a QA professional to capture the entire state of the running configuration in the state in which the defect is easily reproduced. Once captured the library configuration can be referenced in the defect report. When the bug report arrives to the developer, he can check out the configuration and instantly recreate and debug the problem.

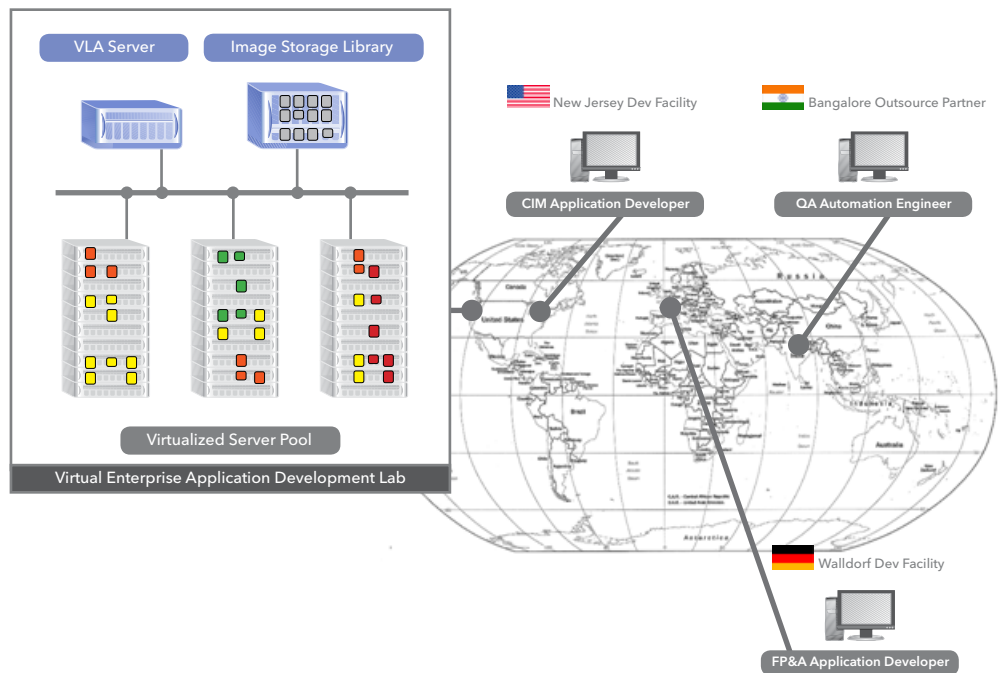
Maintain a complete library of customer and production system environments. By maintaining, in the VLA library, a collection of production and key customer deployment environments, an organization can lower the cost of customer support, increase customer satisfaction and deliver software system enhancements with confidence.

The speed with which a VLA System can deploy even very complex configurations makes real-time recreation of customer environments possible. If a customer encounters a system problem, a support engineer can instantly recreate the environment in which the system is deployed, accelerating the troubleshooting process.

In addition to solving customer or production problems more rapidly, a VLA system can help avoid problems in a planned system upgrade. Pulling a customer or production configuration out of the library to test an upgraded software system or patch in a real target deployment environment can be done in seconds. The tremendous cost savings associated with early problem detection and resolution are well documented.

Efficiently move configurations between development and test facilities. The encapsulation and insulation properties of virtual machine technology make the movement of multi-server configurations between geographically distant facilities possible. But it doesn't necessarily make movement of these servers feasible. Virtual machine images can be extremely large - containing the complete state of memory and storage associated with the server.

An effective VLA System goes beyond simply enabling a configuration copy between locations. Using intelligent compression, caching, image fragmentation and reassembly, and background processing, a VLA System can very efficiently move configurations, sync and mirror libraries across VLA System installations.



Ensuring teams of developers and test engineers are “on the same page” even though they may be distributed across continents is made effortless with a VLA System.

CONCLUSION

Responding to an explosion in the number of underutilized servers that have accumulated across software development teams and to the unsustainably increasing percentage of time spent on repetitive system setup and configuration tasks in support of AD activities, enterprise application development teams, and the software development teams in leading software companies and system manufacturers, have turned to Virtual Lab Automation for relief.

By slashing equipment-related capital and operating costs, dramatically reducing the time wasted repetitively provisioning systems, and by providing for cost-effective quality assurance even when teams are distributed around the globe, the Virtual Lab Automation System represents a new and critical “best practice” component in any software development and test environment.

ABOUT AKIMBI SYSTEMS

Akimbi Systems is the Virtual Lab Automation technology, product and market share leader. Since 2002, Akimbi technology has been a mission-critical infrastructure component within some of the world's most effective software development organizations. Over two dozen Global 2000 enterprises including Coldwater Creek, Experian, Intel, Juniper Networks, RSA Security and Sara Lee Corporation have selected the Akimbi Slingshot™ Virtual Lab Automation System to power their Virtual Lab Automation initiatives. Akimbi is a privately held company and has received funding from Hummer Winblad Venture Partners, Mayfield Fund, Partech International and Stanford University. Strategic partners include VMware, Microsoft, PlateSpin and Mercury.

For more information about Akimbi's Virtual Lab Automation solution visit www.akimbi.com, email us at sales@akimbi.com, or call us at 1-877-4-AKIMBI (+1.650.356.4400).



1400 Fashion Island Blvd.
Suite 900
San Mateo, California 94404

+1.650.356.4400 phone
+1.650.356.4450 fax
www.akimbi.com