Virtual Production Environments for Software Development and Testing

There exists a significant gap between production and development environments for IT applications. This gap leads to a false sense of quality during development. Even with detailed tests planned out, quality of service in production is far from acceptable. A bug found in production (due to environment differences) is substantially costlier to fix as opposed to finding it during development. Worse still are design changes that need to be done post production. This paper proposes the concept of a Virtual Production Environment (VPE) which is meant to make developers and testers ‘feel’ the ‘pain’ of production. More specifically, it should be able to reproduce the production environment as well as provide controls for what if analysis. The paper also provides a review of state of the art technologies that are suitable for components of the VPE and the need to integrate these in to a holistic emulated environment.
About the Authors

Rajesh Mansharamani
Rajesh Mansharamani is a Vice President at Tata Consultancy Services. He is currently heading the Performance Engineering Research Centre, which focuses on tools to automate performance engineering in the software delivery lifecycle. He has been involved in architecture, capacity, and performance management of several nationwide systems. Rajesh holds a Ph.D. from University of Wisconsin, Madison.

Manoj Nambiar
Manoj Nambiar is currently with the Performance Engineering Research Center, TCS. His expertise is in Systems & Network Performance, where he has created a number of tools for network performance emulation and profiling. He has executed a number of consulting assignments for TCS customers worldwide. Manoj holds a Bachelors Degree in Computer Engineering from the University of Mumbai.

Dattatraya Tendulkar
Dattatraya Tendulkar is currently with the Performance Engineering Research Centre, TCS. His expertise is in Database Systems Performance, where he is creating several assets for database production emulation and volume data generation. He has been involved in database architecture and design for several large systems worldwide. Dattatraya has a Bachelors in Computer Engineering from Mumbai University.

Amol Khanapurkar
Amol Khanapurkar is currently with the Performance Engineering Research Centre, TCS. His expertise is in performance of J2EE and document management systems. Amol has created several assets for Java profiling and fast storage and retrieval of images for large systems. He has led performance centres of excellence at several TCS customers. Amol has a Master of Computer Applications degree from Nagpur University and is a recipient of the TCS Young Innovator Award.
Table of Contents

Introduction 3
Virtual Production Environment 3
VPE Components 5
State of the art in VPEs 7
Benefits of using VPEs 9
Conclusions 10
Acknowledgments 10
References 11
Introduction
Today there exists a significant gap between production and development environments for IT applications. Production environments are used to service the needs of thousands of concurrent users spread across multiple geographies. Production environments contain data for millions of users and billions of transactions. Applications in production environments need to interface with many external systems.

On the other hand, development and, to a large extent, test environments are accessed by a small number of concurrent users, typically over the local area network. Development and test environments often contain small databases that are sufficiently populated to handle a small number of test cases. Applications run in a stand-alone mode in such environments, and it is not typical for them to interface with many external systems.

This gap leads to a false sense of quality during development. To mitigate this risk, mature organizations embark on a lot of testing including functional testing, system testing, integration testing, user acceptance testing, load testing, volume testing, stress testing, availability, and reliability testing. Even with detailed tests planned out, quality of service in production is far from acceptable. This is due to two reasons. First, there is a significant gap between production and test environments with either less hardware or no wide area network or insufficient data volumes. Second, the pressure of business commitments leads to less time for rigorous testing.

A bug found in production is substantially costlier to fix as opposed to finding it during development. Worse still are design changes that need to be done post production. Either projects fail miserably or usage growth is limited for weeks and months. Even today it is commonplace that production defects are not reproducible in test environments.

This leads us to a fundamental proposition. With virtualisation becoming increasingly popular thanks to the proliferation of virtual machines, why not have a Virtual Production Environment (VPE) for development and testing. Fundamentally, this environment should make developers and testers ‘feel’ the ‘pain’ of production. Section 2 provides more details of what we mean by a VPE. Section 3 details the technical architecture required to create a VPE. Section 4 surveys state of the art technology that can be used to fill in the pieces required for a VPE, as well as the missing technologies that need to be present to solve the puzzle. Section 5 lists the benefits that a VPE can bring to IT organisations. Section 6 summarises the proposition and provides concluding remarks.

Virtual Production Environment
This section provides the characteristics of a VPE. As discussed above, a VPE should make developers and testers ‘see’ a ‘mirror’ of production. Essentially, we would like to bridge the gap between production and development environments. Our focus is on non-functional characteristics such as performance, availability, and reliability.

A virtual machine provides an application a virtual environment for execution, where the virtual environment behaves like a regular Operating System. The application does not have to undergo any change to run on a virtual machine. This should be the underlying theme for a VPE.

A VPE should provide for the following characteristics:
• **Reproducibility**: a VPE should reproduce the production environment for end application users. In other words:
  
  ○ Responsiveness. Even if an application is being accessed by a single user in development or test, the VPE should make it appear as if there are a large number of concurrent users. Even if an application is being accessed over the local area network, the VPE should make it appear as if the application is being accessed over a wide area network (WAN). Even if an application is accessing a small database in development or production, the VPE should make it appear as if there are billions of rows.
  
  ○ Availability. Components of production systems may fail at random and this can cause repercussive effects on applications. The VPE should mimic random failures in production.
  
  ○ Reliability. Environment behaviour is not always reliable. For example, a WAN may reorder packets or corrupt them. The VPE should provide for unreliability in the environment.
  
  ○ Completeness. Applications need to interface with multiple systems in production. The VPE should provide for these external interfaces.
  
  ○ Accuracy. Ideally, the application’s behaviour in the VPE should be identical to what it would be in production. In practice, the VPE should at least come within the closest order of magnitude. For example, if end-to-end transactions take 20 seconds in production, or database requests take 10 seconds in production, then in the VPE the timings should be reflective of what would occur in production, at least for the transactions that are bottlenecks.
  
Note that these properties are desirable for a VPE. Whether they can be achieved in reality is a challenge that we need to address.

• **Flexibility**: Production environments and workloads may change over time. WAN bandwidths may get upgraded. Network delays will be different for different geographies. Number of concurrent users may change during the day or may increase due to business growth. Production database volumes may change over months. A VPE must provide controls for environment and workload characteristics such as WAN delay, availability, bandwidth, concurrent users, database volumes, and background loads, simulators or emulators of interfaces. VPEs can be configured by infrastructure administrators much the same way as they configure production environments.

• **Transparency**: A VPE must integrate transparently with a development and test environment. The transparency should be so high that users of development and test environments must not even know about the presence of such an environment. They may at most feel the characteristics of such environments, but the VPE must not be intrusive. The presence of the VPE should not change the development and test process. The transparency must also extend to production system monitoring tools, such that tools provide similar monitoring measures such as resource consumption and latencies, as they would provide in a real production environment. Again, the monitoring tools need not know of the VPE and need not be changed for the VPE.

• **Measurability**: A VPE must provide measurements of resource consumption, throughputs and latencies, much the same way as an external monitor tool like HP BAC/Sitescope [1] or Nagios [2] would provide.

• **Configurability**: If one needs to just reproduce a slice of production, or say 10% of production capacity, a VPE must provide for these options, which will be useful for what if scenarios as well as provide analysis for infrastructure planning and deployment.

• **Simplicity**: A VPE must be easy to install and simple to configure. Since the intended usage is to augment a development or test environment, the VPE setup and usage should not intrude in to application development and test methodology and schedules.
VPE Components

Developers and testers typically have their applications hosted on web/app servers and database servers. As shown in Figure 1, a VPE should surround this development and test environment with a broader view.

A VPE must create a production like view for networks, database volumes, workload, interfaces, and monitoring and configurability, as shown in Figure 2.

Note that a VPE is meant to replicate production with respect to a given set of applications that need to be engineered for non-functional requirements. The intent of the VPE is not to have a blind copy of production. There may be hundreds or thousands of applications in production, which may have nothing to do with the applications being engineered.
The components required for a VPE are:

- **Workload Emulator:** Every component of the VPE should behave as if it is operating under production workloads. More specifically, concurrent user access and background processing. Note that it may not be possible to always provide controls for an exact characterisation of production workloads. However, simple controls for the most typical workload cases would be desirable to use early on in the development lifecycle.

- **WAN Emulator:** Developers and testers must feel as if they are accessing a WAN as opposed to a LAN. WAN characteristics such as latency, bandwidth, jitter, packet drop, should be provided as controls. The WAN emulator should allow for different developers and testers to set different values of WAN characteristics.

- **Database Volume Emulator:** Database volumes in development and production are just about sufficient to manage a few test cases. The volume emulator must provide controls such that billions of rows across multiple tables can be emulated. This means that database calls such as inserts, updates and selects must get delayed through the volume emulator as if they would in production.

- **Interface Emulator:** Applications will talk to a variety of other IT applications and components in a production environment. For example, LDAP servers will be called for authentication, legacy systems for information or processing, ERP/CRM systems, or MDM systems. The setup may be very vast and it is in the interest of the VPE to allow for emulating the interfaces.

- **Monitoring Framework:** The VPE needs to provide measurability of the environment it is emulating in terms of CPU, memory, disk, network utilisation, latencies across various system and application components hosted on it.

- **Capacity Provisioner:** One must get the control to provision only a percentage of production capacity or much more than production capacity, and evaluate what impact it has on application performance and scalability.

Note that we have used the term emulator as opposed to simulator, since we would like no change in process to development and testing. Developers and testers will continue to access their application and database servers. The VPE makes no difference to the functional use of the application and database servers. Thus it can be inserted and removed at will with no change in process or functional aspects of development and testing.

As such a production environment is very vast and complex, since changes are typically additive and over a period of time components grow out of bounds. The emulators, mentioned above, are by no means a complete set to characterise a VPE. For example, network devices and security concerns have been conveniently left out. However, in interest of performance, availability, and reliability these are the most important components of a VPE. These are necessary components but may not be sufficient for all types of production environments.
State of the Art in VPEs

As of today there is no integrated VPE product in the market. As for the components listed in Section 3, there are commercial, open source, and custom built approaches followed by IT organisations.

Workload Emulators

The workload is emulated during load testing phases of projects. Popular commercial tools include HP LoadRunner [3], IBM Rational Performance Tester [4], and Borland SilkPerformer [5]. These scale to thousands of concurrent users. Open source tools such as OpenSTA [6], WebLOAD [7], and Jmeter [8] scale to hundreds of concurrent users. These can be used only when application front ends are ready, since the approach they use record and replay techniques. The recording of user actions on GUIs (typically web pages) is converted into scripts which may need modification for replay. The replay can be configured to have multiple users, even though recording was done for only one user, by means of parameterising data inputs.

Workload emulation early during development or component testing is typically absent, though custom built scripts are used in mission critical projects.

One has to keep in mind that such type of load injection requires production like servers to sustain the load. On the other hand, use of modelling techniques can provide estimation of response times or concurrent requests under a workload profile. While modelling techniques and tools are available, none of them has been used to create a performance deceleration tool. By a performance deceleration tool we mean one that decelerates a single user response time by the amount the request would take under concurrent user load (on a larger number of computing resources).

WAN Emulators

WAN emulators replicate WAN characteristics in a LAN environment. The most popular commercial WAN emulator is from Shunra [9]. It is used in test environments and has to be physically coupled as a ‘man-in-the-middle’ between the load injector server(s)/desktop(s) and the web server(s)/load balancer. A network cable goes from the load injector server/desktop network port to one of Shunra’s input ports. Shunra’s output port is connected to the load balancer/web server by means of another network cable. Shunra provides controls specifying latency, bandwidth, packet drop, packet corruption, packet reordering, and jitter. It also provides characterisation of network links to estimate the input values for these controls.

NIST Net [10] has been a popular open source WAN emulator. It provides controls for latency and bandwidth but none for availability and reliability.

TCS has released its WAN emulator WANem [11] to open source in July 2007. This tool has become popular in the open source community due to its ease of installation and the variety of features it supports, which include latency, bandwidth, packet drop, packet corruption, packet reordering, jitter, and connection drop. Like Shunra, WANem also provides link characterisation. WANem is available as a Knoppix bootable CD. You can follow these three simple steps to use it:

1. Boot your PC with the WANem CD. It now boots as a WANem PC.
2. Configure your network routes between client and server, so that network packets flow through the WANem PC.
   This is done by simple ‘route add’ commands in DOS, Linux, or Unix.
3. Access WANem controls through your browser and set the values appropriately.
**Database Volume Emulators**
There is no product in the market to do database volume emulation. The technique followed by IBM DB2 experts or Oracle experts is to manipulate database statistics to reflect production volumes and distributions. This influences the query optimiser to work as if the development or test database has production volumes, and it generates query plans that would show up in production. Developers then optimise their queries by looking at such query plans through ‘EXPLAIN PLAN’ commands.

This approach requires significant expertise by developers and does not provide any estimate of time that the query would take in production. The EXPLAIN PLAN commands do however provide query cost estimates that developers try to optimise.

TCS has built a database volume emulator called DBProdem. It provides for:
- controls to set the database statistics reflective of production
- query execution time estimation in production by empiric models
- interception of queries and DMLs to the database, and delaying them in proportion to the time estimated to take in production

DBProdem is currently available on Oracle 10g.

**Interface Emulators**
There is no product in the market for interface emulation. This is the hardest of the emulators listed above, since it is hard to commoditise interfaces. Building interface models is not easy when one needs to scale to tens to hundreds of interfaces that may keep changing in production. Sniffing production calls to interfaces and characterising them is very difficult since the data content in responses can be hard to model without knowing enough details of the interfaced systems. The typical approaches during integration testing to test interface behaviour are as follows:
- Have a test environment as close to production as possible, so that for every interface in production there is an equivalent interface in test. This is easier said than done, and often the deployment in test environment is for a handful of critical interfaces.
- Create stubs for interface systems. These are custom built and need to be maintained for changes to the interface systems.
- Hook off production for testing. This is done in case of some legacy environments, especially for query workloads, but is often avoided because of inconsistencies that can be introduced in to production.

**Monitoring Framework**
There are a large number of systems management and monitoring products in the market today such as IBM Tivoli [12], HP OpenView [13], HP Mercury BAC [1], BMC Performance Manager [14], and CA Unicenter [15]. However, these are meant for true production environments. Since there is no notion of a VPE in the market today the question of an integrated monitoring framework for it does not arise. Products the individual components, such as WAN emulators and load injectors, have of their own built-in monitoring for the metrics they capture such as bandwidth, link utilisation, and response times.

**Capacity Provisioning Framework**
Today, Virtual Machines such as VMWare [16] can be configured for a given number of CPUs and a given amount of RAM. Hardware vendors such as IBM, HP, and SUN allow for static (and sometimes dynamic) partitioning of their servers, with some of them even allowing for micro partitioning of the CPUs. While these features are no doubt very useful in production environments, we would not like users of VPEs to spend significant time in environment
configuration. The accuracy of the VPE, as mentioned in Section 3, needs to be within the order of magnitude of production, so that fragility can be detected early enough.

We therefore need to use aspects of performance modelling to predict the behaviour of applications on different capacities. The advantage of the modelling approach is that not only can we judge behaviour under constrained capacity but also under excess capacity. The modelling methodology needs to be transparent to the end user, who needs to be provided the controls only. There are a number of performance modelling tools in the market such as SPEED [17], OPNET [18], SHARPE [19], as well as simulation frameworks such as Simpack [20] and SIMSCRIPT [21]. These need to be integrated with the execution of a VPE so that modelling is embedded in to the VPE.

Summary
As can be seen, there is a significant gap in the market between what is desired in a VPE and what exists. While there is no VPE product, components are available for some pieces and custom-built approaches used for other pieces. WAN emulation is the only piece that has reached maturity in the market today.

TCS plans to integrate its WANem and DBProdem products along with its transaction and application profiling and modelling tools, to create a single VPE for development and testing. As of today, deployments are done by using one PC for WANem and one for DBProdem, with separate GUIs for each emulator. This has proven quite useful to end customers as will be shown in the following section.

Benefits of Using VPEs
As discussed in the introduction, the benefits of using VPEs can be enormous both during development and during testing. The benefits extend to infrastructure management and also for capacity planning or troubleshooting or proactively tuning for performance and availability. We present two case studies in this section to prove the point.

• **Large Health Service**
  A very large nationwide health service needs to store patient data for 500 million prescriptions. Understanding the magnitude of the task, the development database had 5 million prescriptions and after significant amount of tuning, the development team was comfortable with the database performance. The focus of the team shifted to acceptable performance for the web service layer. When TCS DBProdem was used to emulate query performance under 50 million prescriptions, it was seen that the database performance was a major issue, much more than the web services tier. Several queries took more than 15 seconds to execute on the emulated environment though they completed in sub seconds in the development environment.

  As a result, design changes and query tuning was done to bring performance under control in the emulated environment, and now this process has become mandatory for all large database development. Had the design changes for performance not been detected they would be impossible to make during the test cycle or post production release since they would have caused change to a large number of queries. DBProdem not only provided the estimated query response times but also provided the query execution plans as they would be seen in production.

• **Large Insurance**
  A large motor insurance organisation needed to size network bandwidth for hundreds of locations all over the country, for a new core insurance product that they had selected. The network bandwidth budgets had to be
decided very quickly and there was no time to get network specialists to do the estimation. The scientific way to estimate network bandwidth would be to collect network data sent and received for the most frequent transactions such as policy and claim creation, create a network sizing model with the data, and estimate the bandwidth by executing the model. Getting such specialists and having them conduct the exercise and submit a report can take over a month.

Using WANem for WAN emulation, all that was needed instead was an insurance user to access the core insurance product and fill out a policy and a claim. Unknown to the insurance user, the browser traffic to and from the web server was routed through WANem. Using the WANem controls the insurance expert kept reducing bandwidth dynamically until response time was not acceptable and thus estimated the minimum bandwidth required per user. The rest was easy for the business experts to extrapolate since they had data on the number of users per branch across the country. This entire exercise was completed in less than an hour.

**Conclusions**

This document has highlighted the need for VPEs. A VPE seamlessly integrates with a development or test environment and provides a ‘feel’ of production to developers and testers. The reliability of a VPE is in its ability to reproduce production characteristics as much as possible in terms of background load, WAN characteristics, impact of volumes, and large number of interfaces. To integrate seamlessly into development and test environments, VPEs need to emulate production environments rather than simulate production.

There is no VPE product in the market, though there exist WAN emulators, load injectors, and custom approaches for database volume emulation and interface emulation.

TCS has embarked on creating VPEs that will integrate into development and test environments. This is currently done through its two products, WANem which is WAN emulator released to open source, and DBProdem which is a tool to emulate database volumes. These are currently being integrated in to one tool which will also feature profiling to quantify the performance, availability, and reliability of the applications using the VPE.

**Acknowledgements**

The authors would like to thank B. Gopal, Kishor Gujarathi, and Jayanti Murty from Tata Consultancy Services for their valuable feedback.
Virtual Production Environments for Software Development and Testing

References
1. https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-10_4000_100__
3. https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-126-17%5E8_4000_100__
14. http://www.bmc.com/products/products_services_detail/0,0,0,0_2001,00.html
About Performance Engineering Research Centre
TCS Performance Engineering Research Centre (PERC) creates tools to enable systematic performance engineering during the software development and system management life cycles. While the current research is towards development of state of the art emulators and analyzers, the plan ahead is to build performance automation engines for software optimization and capacity management in large systems.

About Tata Consultancy Services (TCS)
Tata Consultancy Services Limited is an IT services, business solutions and outsourcing organization that delivers real results to global businesses, ensuring a level of certainty no other firm can match. TCS offers a consulting-led, integrated portfolio of IT and IT-enabled services delivered through its unique Global Network Delivery Model™, recognized as the benchmark of excellence in software development.

A part of the Tata Group, India’s largest industrial conglomerate, TCS has over 100,000 of the world’s best trained IT consultants in 47 countries. The company generated consolidated revenues of US $4.3 billion for fiscal year ended 31 March 2007 and is listed on the National Stock Exchange and Bombay Stock Exchange in India. For more information, visit us at www.tcs.com

Contact us at
performance.engineering@tcs.com

All content / information present here is the exclusive property of Tata Consultancy Services Limited (TCS). The content / information contained here is correct at the time of publishing. No material from here may be copied, modified, reproduced, republished, uploaded, transmitted, posted or distributed in any form without prior written permission from TCS. Unauthorized use of the content / information appearing here may violate copyright, trademark and other applicable laws, and could result in criminal or civil penalties.

Copyright © 2008 Tata Consultancy Services Limited