EVALUATING SELECTED JAVA BEST PRACTICES FOR SAP BUSINESSOBJECTS BUSINESS INTELLIGENCE 4 ON VSPHERE

June 2012

White Paper
SAP Co-Innovation Lab

vmware®

SAP Co-Innovation Lab

Acknowledgements

This document is the work of a virtual project team at SAP Co-innovation Lab, whose members included Ashish C. Morzaria (SAP), Jay Thoden van Velzen (SAP), Kevin Liu (SAP), Roehl Obaldo (SAP), Sivagopal Modadugula (SAP), Vas Mitra (VMware), and Justin Murray (VMware).

The project team would like to thank many colleagues for their support with the setup of the test environment, the execution of the tests, and the review of this paper. To mention a few: Michael Hesse and Emad Benjamin at VMware; and Peter Aeschlimann, Corey Wilkie, Jacques Buchholz, David Cruickshank, Abhay Kale, Irakli Natsvlishvili, David Pascuzzi, Veronique L'Helguen Smahi and Andrew Valega at SAP

Content

1 Introduction

This paper focuses on evaluating a subset of Java-specific best practices by applying them to an SAP BusinessObjects BI 4 deployment and analyzing their effects. The content of the document is based on a series of tests performed in the SAP Co-Innovation Lab in Palo Alto, California in 2012. All tests were completed using the SAP BusinessObjects BI 4 suite and VMware vSphere 5. The resulting recommendations in this document only provide general guidelines related to Java and do not target any specific size or type of BI deployment.

1.1 Scope

This document makes specific recommendations for SAP BusinessObjects BI 4 systems based on the findings from our study and is not intended to be a replacement for general Java best practices from VMware's original document [Ref. 6]. VMware has also created additional best practice documents on a wider range of topics such as the "Performance Best Practices for VMware vSphere™ 5.0" guide [Ref. 5]. These contain more general best practices that are not Java or business intelligence specific. Information in those documents will not be repeated here unless specifically relevant or markedly different when deploying virtualized business intelligence systems.

1.2 Methodology

SAP and VMware technical staff worked together to create a test BI architecture using a set of virtual machines and then drove a simulated user load through the system to establish the optimal ways to configure the Java-related parts of the system. **It should be noted at the outset that this was not a performance tuning or optimization exercise.** We used an out-of-the-box configuration of SAP BusinessObjects BI 4 and VMware vSphere 5 as much as possible.

2 Products

2.1 VMware vSphere 5

VMware vSphere is the key product family that contains the core ESXi hypervisor, the vCenter management suite of tools and added-value features such as Distributed Resource Scheduling (DRS) and High Availability (HA) among others. VMware vSphere aggregates the underlying physical hardware resources across multiple systems and provides pools of virtual resources to the datacenter. Virtual machines are then executed on the new virtualized hardware and these virtual machines live within the boundaries of the resource pools. Customers can achieve tangible savings from this consolidation, and realize operational cost savings from reduced datacenter floor space, power, and cooling.

2.2 SAP BusinessObjects Business Intelligence 4

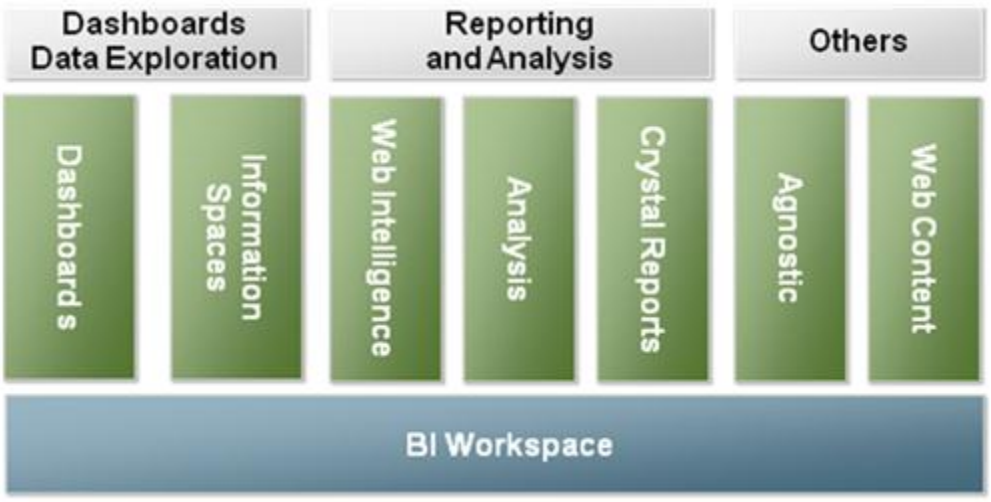
SAP BusinessObjects business intelligence (BI) is a flexible and scalable solution that provides the full spectrum of BI functionality including reporting and analysis, data exploration, dashboards, and predictive analysis. It also gives IT departments a flexible means to share BI content throughout the entire organization, empowering business users to make effective, informed decisions via self-service access to information. Deployable on physical, virtual, and cloud environments, SAP BusinessObjects BI solutions are flexible enough to best fit the unique needs of each organization

while providing a complete enterprise BI solution. They can decide better, perform better, and achieve better results throughout all areas of their business.

2.2.1 Architecture

The SAP BusinessObjects BI 4 platform is a multi-tier, server-based product that is comprised of a number of logical servers. Each report format has its own server modules that are controlled through a single management interface. These servers run as separate processes and they can all be installed on one machine or distributed across multiple machines, with multiple instances of individual servers able to run on each host. This allows for a very flexible and expandable architecture, with servers dedicated to specific processes and tasks, and of virtually any size, depending on how many concurrent users the environment needs to support.

A high level view of these logical servers is depicted below in green:



As the SAP BusinessObjects BI 4 suite has many products within it, performing tests across all of them was simply not feasible. To reduce the testing scope, we focused on the SAP BusinessObjects Web Intelligence and Tomcat application servers as they are Java-based and enabled us to isolate them to properly observe the effects of applying best practices.

The SAP BusinessObjects BI 4 platform technical architecture is composed of a set of tiers optimized for specific tasks and operations. The three tiers discussed here are:

- **Application tier:** Java web application servers and tools
- **Intelligence tier:** System server processes, security, and management
- **Processing tier:** Data analysis and report generation

The Application Tier contains Tomcat and hosts BI Launchpad application in addition to the BI clients such as Web Intelligence, Crystal Reports, and Explorer. As these applications are Java based, this study will focus on applying selected Java best practices to this tier.

The Intelligence Tier contains system server processes, including the Central Management Server (CMS). The CMS manages sessions, security rights, meta-data for the entire BOE system including the application tier (Tomcat)

and Web Intelligence report server in the context of this whitepaper. Refer to the "SAP BusinessObjects Business Intelligence Platform Administration Guide" [Ref. 4] for more details on the CMS.

The Processing Tier contains the processing servers for each of the business intelligence clients and is responsible for generating or processing the actual report content. This tier contains the Web Intelligence processing server and the Adaptive Processing Server.

Storage of actual reports, dashboard, universes, and all other content is typically placed on NAS or SAN network storage infrastructure to take advantage of improved performance, reliability, and multipath I/O capabilities. Reporting databases and even the CMS database (CMS DB) are also typically on separate machines to increase the performance headroom of the BI system.

3 Test Overview

3.1 Objectives

The main objective was to test a selected subset of best practice recommendations specifically when applied to a virtualized SAP BusinessObjects BI 4 deployment. The tests were devised to better understand the effects of these best practices and to educate the reader on the benefits of specific Java-related optimizations. Implementing best practices may negatively impact performance but can yield more important benefits by enabling the system to better survive peak loads and demonstrate consistent performance across varying conditions.

The challenge in executing tests that do not focus solely on performance is that measuring the positive effects of specific configuration changes on overall system stability and consistency is very difficult to prove definitively.

Proving a system can be more stable with specific configuration changes inherently requires the ability to create a baseline metric where a given load consistently fails at the same point. However, noticing significant deterioration or erratic behavior during the tests will be clear signs of a problem.

It is important to note that, just as on dedicated hardware, improvements within one virtual machine (or machine in a cluster) can be offset by degradation in another. Therefore, it is important to look holistically at the resource utilization of every node in your entire physical or virtual landscape and resolve any bottlenecks – regardless of which BI client or reporting format you are using.

3.1.1 Special Note On Performance Optimization

Optimizing enterprise business intelligence systems is an involved process where poor design choices can dramatically affect the efficiency of the system. You should focus on creating a good system architecture first, taking into account stack components, I/O bottlenecks, and processing tier distribution before you attempt any optimization at the virtualization level.

The goal of optimizing and applying best practices is to reduce virtual machine overhead. Realize, though, that no amount of virtualization optimization can fix an inefficient database query or a poorly performing report. It cannot be stressed enough that the performance gains through optimizing the VMware environment will be inconsequential if the system architecture, such as a poorly designed database, is hampering overall throughput of the entire system.

3.2 Test Environment

The test suite was created with a sample load of 40 simulated users against a 3-tier environment. This ensured the system would have "room to breathe" and help demonstrate that changes in the test results were not be impacted by system bottlenecks or external factors.

Multiple usage scenarios were simulated and configuration changes were implemented one by one to understand their effects individually. Multiple test runs were executed and standard deviations were calculated to separate statistically relevant findings from "noise" or extraordinary events occurring in the system. These tests were performed under relatively light loads. The effect of the parameters tested may be different under higher loads. However, we recommend that users of SAP BusinessObjects Business Intelligence consult with their SAP representative or partner before deviating from the recommended practices discussed in Section 5. The results of these tests are specific to the SAP BusinessObjects application-tier.

Tests were focused on the Tomcat server, which functions as the Web/application server for the BI Launchpad and Web Intelligence client, and the SAP BusinessObjects Web Intelligence reporting stack, which includes the Adaptive Processing Server (APS). This approach still had its challenges as we found that tests involving a "live refresh" from the reporting database demonstrated changing response times at the database layer due to caching and prefetching, as well as increased query times as load increases. However, the focus of this exercise was on the Tomcat server and the SAP BusinessObjects Web Intelligence report handler to better isolate the behavior within the SAP BusinessObjects BI 4 environment.

The host environment consisted of two identical blade servers with the following configuration:

- VMware ESXi v5
- 2 x Quad core Intel Xeon X5570 processors
- 2 x 10Gbps network cards
- 96 GB RAM

The reference environment separated the deployment into three tiers, each running in their own virtual machine on a single ESXi host as follows:

1. Tomcat/Web Application Tier
2. Central Management Server (CMS)
3. Web Intelligence
    a. Adaptive Processing Server (APS) running on the Web Intelligence machine
    b. All services not required for the test were shutdown to eliminate external factors

Each of the above virtual machines was configured identically:

- 2 virtual CPUs and 16 GB of memory
- Windows Server 2008 R2, with VMware Tools installed
- VMXNET3 and PVSCSI drivers installed

The CMS database and the reporting database were individual virtual machines on a separate ESXi host to isolate the processes under investigation and eliminate the impact from other systems. Each virtual machine was configured as above, but included additional RAM (24 and 48 GB respectively) to ensure database performance would not be a bottleneck in the tests. As Section 6.1(Methodology) describes, the test patterns were devised in such a way that the reporting database doesn't play much of a factor once the target reports have been refreshed the first time with data.

NOTE:

This is not a typical or even a recommended production BI environment. Each component here functions as a single point of failure, and was designed primarily to investigate and isolate the effect of particular settings changes on the environment as a whole, and on each individual virtual machine. By isolating the BI Platform

components this way and separating them from each other, we could observe any direct effect our parameters had over a baseline test in order to determine whether the effect was indeed meaningful.

All client systems were also provisioned as virtual machines on a separate ESXi host (with the same configuration as the other two hosts) to ensure there was no performance impact on the BI system by clients generating the testing load.

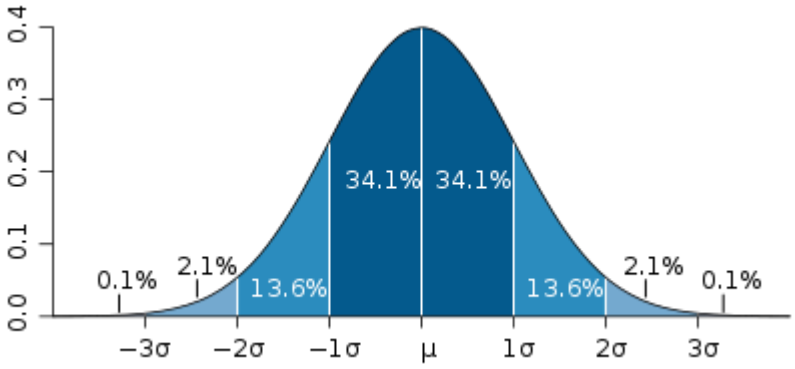
3.3Test Procedure

The tests were structured in the following way:

1. Implement one of the selected Java best practice recommendations individually across the various components in the environment. After each test, the environment was restored to its original configuration - no tests were done with cumulative changes except where noted.
2. Perform a known test scenario for a specific time period against the environment.
3. Repeat the identical test 10 times each, for the same number of users, and for the same time period.
4. Determine the mean and standard deviation for the following metrics in order to determine throughput:
    a. Messages sent
    b. Transactions completed
    c. Bytes Sent
    d. Bytes Received

Between tests, the only parameters changed were memory allocations and Java parameters.

We wanted to determine whether a particular change lead to a significant and meaningful improvement or deterioration. For this we decided to use the normal distribution in order to see whether a change really had a measureable effect.



By running a baseline to find the mean as well as standard deviation for the default, out-of-the-box configuration, we could use the results of 10 test runs for a particular configuration and determine whether it had an effect that exceeds 1σ or even 2σ.

Conclusions and recommendations of this paper are derived from statistically relevant results. It is important to remember that each configuration and environment is unique and the parameters discussed here may have more or less impact on a real system than on the test bed here.

A detailed description of the test steps conducted is detailed in Section 6.1, “Methodology” on page 14.

4 Analysis of Findings

A number of conclusions can be derived from these tests, but it is important to remember that the tests were done on an isolated system under a specific load. Systems that have much higher utilization or are experiencing more user activity may behave differently. Therefore, these conclusions should be regarded as recommendations that illustrate concepts rather than hard requirements.

For detailed test data regarding each conclusion, please refer to Section 6.2, “Detailed Test Results” on page 15.

4.1 Java Heap Size

Tests conducted with 1 GB and 4 GB Java heap sizes did not have statistically relevant performance differences under relatively light loads, except in the case where large page support is enabled but the Windows system is not rebooted. As the best practice is always to reboot Windows operating systems when changing something as fundamental as memory management, we can safely exclude this outlier.

SAP’s position is that deviating from out-of-the-box configurations of SAP BusinessObjects BI 4 solely because it is virtualized will not result in a significant and predictable performance improvement. Therefore, no virtualization specific optimizations are recommended for the Java heap, and you should follow the “SAP BusinessObjects BI 4 Sizing Companion Guide” [Ref. 3] just as you would in a physical deployment.

4.2 Use of Large Pages

Large page support enables server applications to establish large-page memory regions, which has been shown in tests on some systems to improve performance. Memory address translations use a Translation-Lookaside Buffer (TLB) which is a page translation cache that holds the most-recently used virtual-to-physical address translations. TLB is a scarce resource and a TLB miss can be costly as the CPU must read from the hierarchical page table which requires multiple memory accesses. By using a bigger page size, a single TLB entry can represent a larger memory range. Therefore, when using large pages, memory intensive applications could have better performance.

However, the use of large pages can also negatively impact system performance. If the working set of an application is scattered over a wide range of address space, the application is likely to experience thrashing of a relatively small number of large TLB entries and result in worse overall performance due to higher TLB miss rates. Additionally, applications that pin large amounts of memory may create a shortage of contiguous memory for other processes and negatively impact system performance due to memory fragmentation and eventual swapping.

It is important to note that when large pages are being set up for a virtual machine, a change is needed to be made to the guest operating system as well as to the JVM command line. These are documented in [Ref 6] and [Ref 7] respectively.

Tests 1-4 show that there is no statistically relevant evidence that enabling large pages positively or negatively impacts system performance under this specific light load. This situation may be different with a system that has higher memory utilization. For example, when looking at tests 6-8 where a 4GB heap is used, we would have expected a higher likelihood of seeing a performance improvement. However, in our tests, we observed that memory utilization did not exceed 2.3 GB which may explain the lack of an improvement.

The conclusion is that while enabling large page support for virtual machines containing Java applications can potentially provide a benefit, this isn’t consistent for all workloads and can possibly have the opposite effect under

certain circumstances. We also see in tests 7 and 8 that if not configured correctly, enabling large pages can significantly degrade performance.

4.2.1 Importance of Rebooting After Changes

Our tests also proved that applying Windows OS account level system settings such as “Lock Large Pages into Memory” require a reboot to take effect. In fact, neglecting to restart the system after such a fundamental change will result in a statistically relevant degradation of performance compared to making no changes at all.

4.3 Memory Reservations

Memory reservations are typically recommended to protect virtual machines during times of high RAM utilization in the ESXi host. This ensures a virtual machine will always have the memory resources it expects. Production Java applications require that their “heap” memory space be held in physical memory throughout their lifetime, to ensure that they perform according to their SLAs. If that memory is not physically available when the JVM needs it, then the performance of the application may be affected.

Tests 1-3 show there is no statistically relevant evidence that applying a memory reservation either positively or negatively impacts performance for a system under our light load. Note that in these tests 5-7, a 4GB heap was not fully consumed by the application, which means the virtual machine was actually using less memory than it asked for and this is likely the reason for not observing a measurable impact. All tests in this study were conducted without any memory overcommitment on the host. This means each virtual machine received all the memory it asked for without requiring any memory sharing or reclamation. In a more populated host where memory is being severely overcommitted, the lack of a memory reservation could cause either the guest OS in a virtual machine or the ESXi host to swap to disk. Our tests were executed with no memory overcommitment situations at the ESXi host level – thus not showing the benefit of setting a memory reservation.

VMware’s recommendation that virtual machines containing production Java applications that have a strict performance SLA should use memory reservation holds. We found that this does not have a negative consequence on system performance but has a potentially significant upside. Therefore the cost savings of over-allocating memory needs to be carefully weighed against the chance that system performance could seriously degrade through swapping either in the guest VMs or the host itself.

5 SAP BusinessObjects BI Platform Java Recommendations

The recommendations in this document provide guidance only and do not represent strict design requirements because each SAP BusinessObjects BI configuration can vary from one implementation to another. You should consult your SAP representative or partner to determine which recommendations are applicable to your deployment.

5.1 Follow VMware and SAP Virtualization Best Practices

Multiple studies, including this one, have shown that virtualizing SAP BusinessObjects BI does not require specific “tuning” when inside a virtual machine. VMware vSphere does an excellent job of abstracting the details of virtualization from the application so that best practices to deploy BI on a physical system are no different than in a virtual machine. However the performance of a virtualized BI deployment can be severely impacted by a poorly architected or implemented virtualization environment. It is critical to ensure you have a proper understanding of both good virtualization and BI systems design.

5.1.1 SAP's Standardized Virtualization Support Policy

"SAP Note: 1492000: General Support Statement for Virtual Environments" [Ref. 1] states that your virtual configuration is supported in the same manner as a physical one, assuming you have followed the requirements within that document. Therefore, performing virtualization-specific optimizations at the virtual machine or application level can impact your ability to be fully supported by SAP Support if it is deemed that the system in question has deviated from how it would be configured on a physical machine.

It is also possible that applying feature packs, service packs, or patches on top of such a system could result in unpredictable behavior or suboptimal performance that will be difficult to diagnose correctly. Unless you have a very clear understanding of the consequences of such a change, it is recommended that you do not deviate from a similar configuration in a physical deployment.

5.1.2 VMware's Best Practices Guide

Proper design of your virtualization environment is critical for good performance and system stability. This is especially true for BI systems as they are very I/O intensive in addition to being CPU and RAM intensive. You should follow VMware's general best practices and architecture guidelines as detailed in "Performance Best Practices for VMware vSphere™ 5.0" [Ref. 5].

5.1.3 Architecting Your VMs Appropriately

Virtualized environments can impose a minimal amount of overhead that impacts the performance of each virtual machine. The amount of virtualization overhead is a combination of actual hypervisor processing costs and other factors such as CPU, RAM, and I/O latency. Mitigating these effects is beyond the scope of this document, but you should pay special consideration to the following recommendations:

- Consider scale out instead of scale up: When designing virtual machines for your deployment, use an appropriate number of virtual CPUs as recommended in the "SAP BusinessObjects BI 4 Sizing Estimator" [Ref. 2] and the "SAP BusinessObjects BI 4 Sizing Companion Guide" [Ref. 3]. It is a best practice to scale out the deployment across multiple nodes instead of scaling up by allocating all required virtual CPUs in a single virtual machine.
- Size the VM properly: Allocating more virtual CPUs than required can increase virtualization management overhead without necessarily providing a corresponding increase in performance. This choice is really workload dependent and requires testing for your own system. Monitor the amount of virtual CPU "Used" in the vCenter management console and consider adding virtual CPUs to your virtual machine if the current consumption is high (above 75%).
- Allocate enough RAM: Ensure the virtual machine has enough RAM allocated to it so that the guest operating system can handle all processes that it needs to. A lack of RAM when the JVM requires it can result in excessive swapping in the guest operating system even when the host has plenty of memory and the virtual machine has a memory reservation.

Please refer to "Performance Best Practices for VMware vSphere™ 5.0" [Ref. 5] document for more recommendations.

5.1.4 Use Memory Reservations on Production Systems

For production systems, consider not overcommitting the physical memory of the ESXi server. Enterprise Java applications inherently allocate large areas of memory due to the application's requirements. The likelihood of a Java application within a virtual machine quickly utilizing allocated RAM is therefore higher than other types of applications. Correspondingly, the consequences of overcommitted memory are even more severe than in non-Java applications.

It is somewhat common in production environments to have memory over-allocation (where the sum of memory allocations for all virtual machines on a host is greater than the physical memory of the host). While this provides some cost savings on the assumption that not all virtual machines will use all their allocated memory at the same time, when the over-allocation becomes over-commitment, system performance can be severely impacted without warning. When operating an SAP BusinessObjects BI deployment in a virtualized environment, there is an obligation to either use explicit memory reservations or continuously monitor the ESXi host or cluster to ensure performance is not being significantly impacted as a result of overcommitting resources.

It is also important to note that it may not be possible to file a performance related support incident with SAP Support until the system in question has proven to be deficient on a system where there is no resource over-commitment. Performance issues that cannot be isolated to the application level may require monitoring products such as SAP Solution Manager so SAP Support personnel can view the VMware counters affecting your system.

For test, development, and other non-production systems, memory over-commitment can be more acceptable as these systems do not fall under SAP's support and SLA terms.

5.2 Follow SAP's Best Practices and Sizing Guides

The general sizing principles for SAP BusinessObjects BI on VMware vSphere remain the same as for physical systems, with some adjustments for virtual machine overhead. Virtualization introduces additional architectural considerations and general VMware best practices should be considered as additive and also fully relevant in your architecture decisions.

Actual sizing and architecture design of SAP BusinessObjects BI solutions at the beginning of projects should be conducted in conjunction with SAP Professional Services or an SAP implementation partner. The following are also resources that you should use when determining the rough sizing for your system:

- SAP BusinessObjects BI 4 Sizing Estimator [Ref. 2]
- SAP BusinessObjects BI 4 Sizing Companion Guide [Ref. 3]

5.2.1 Do Not Enable Large Page Support

Large page support may improve system performance under some conditions, but there is no conclusive evidence either through the tests performed here with Windows Server 2008 in a virtual machine or independent tests performed by SAP engineering on physical systems to prove that enabling large page support for SAP BusinessObjects BI 4 provides a significant or consistent improvement. SAP does not specifically recommend enabling large page support on physical machines and therefore you should not treat this differently in a virtual environment.

Enabling this support on either physical or virtual machines can result in unpredictable system performance and an inconsistent performance profile as the usage of the SAP BusinessObjects BI system changes over time. As the above section describes, you should follow the recommendations from the "SAP BusinessObjects BI 4 Sizing Companion Guide" [Ref. 3] fully and you should not deviate from this by making virtualization specific optimizations.

It is also notable that if you plan on over-committing memory for systems in production or non-production environments, using large pages will reduce the ability for VMware vSphere's to use Transparent Page Sharing (TPS) effectively unless the system is under high memory pressure.

5.2.2 Java Heap size

The SAP BusinessObjects BI platform leverages Java and in some cases changing the Java heap size for physical systems expecting heavy user traffic can improve performance. This was also observed in our tests using virtual machines. Tests from this study indicate that memory management both at the virtual machine and JVM levels are

operating as they would on physical hardware. When considering changes to Java heap allocation, you should follow the guidelines in the “SAP BusinessObjects BI 4 Sizing Companion Guide” [Ref. 3] as they fully apply in physical and virtual deployments equally.

6 Appendix - Test Methodology and Results

6.1 Methodology

The steps for the test scenarios were:

1. Performance sequence for “Logon user”
2. Navigate to the Public Folder list
3. Open a Web Intelligence report with 2,500 rows of data
    a. Note: “Refresh on open” was not enabled to exclude performance effects/impacts from database caching and optimizations
4. Move to the next page
5. Close the report
6. Open a Web Intelligence report with 5,000 rows of data
7. Move to the next page
8. Close the report
9. Open a Web Intelligence report with 10,000 rows of data
10. Move to the next page
11. Close the report
12. Open a Web Intelligence report with 25,000 rows of data
13. Move to the next page
14. Close the report
15. Logoff user

An SAP BusinessObjects BI 4 business layer was developed for the reports (UNX), the new semantic layer format in the SAP BusinessObjects BI Platform. This was connected through the Sybase IQ ODBC driver to the Sybase IQ database. Once the reports were refreshed and contained data (of the data volume specified), they were saved and were ready for testing. Note that no interactive refresh to the Sybase IQ database occurs during the tests documented here.

This test scenario is run for all of our tests, for 40 simulated users, with 10 seconds think time between individual actions, and for a duration of 20 minutes, with a 5 minute ramp up time. For each configuration, the test was run multiple times (between 7 and 15, typically around 10-12) in order to get a sense of the standard deviation of the test results within a particular configuration.

We could then take our various test results and compare them to each other with a degree of confidence. If we got a result that fell outside the 2σ range, we can state with ~95% confidence that the result is significant, and due to the change in configuration.

6.2 Detailed Test Results

6.2.1 Test 1 – First Baseline with 1 GB Heap Size

The first baseline test was a completely out-of-the-box configuration (using BI 4.0 SP2) with default settings and no modifications of any kind. This uses a Java maximum heap size of 1GB for the Tomcat JVM. It should be noted that as of BI 4.0 FP3, the default maximum has increased to 2 GB.

	Messages sent	Total Transactions	Bytes Sent	Bytes Received
Average:	**82,371**	**4,472**	**209,083,081**	**988,360,144**
Std Dev:	**621**	**23**	**1,550,972**	**7,515,089**

6.2.2 Test 2 – Large Memory Pages for Tomcat, Memory Reservation for Web Intelligence

The second series of tests employed the "Use Large Pages" setting for the Tomcat virtual machine and for the Java command line as well as introducing an 8 GB memory reservation for the virtual machine hosting Web Intelligence.

- Java heap: 1 GB
- Large pages enabled for Tomcat virtual machine
- 8 GB memory reservation for Web Intelligence VM

	Messages sent	Total Transactions	Bytes Sent	Bytes Received
Average:	**82,727**	**4,483**	**209,987,845**	**993,030,062**
Std Dev:	**418**	**19**	**1,105,252**	**5,237,612**

In this test, there is not a statistical change by using large pages and a memory reservation in this case.

6.2.3 Test 3 – Large Pages With Memory Reservation For Tomcat and Web Intelligence

The third series of tests introduced a 4 GB memory reservation for virtual machine containing Tomcat:

- Java heap: 1 GB
- Large pages enabled for Tomcat virtual machine
- 8 GB memory reservation for Web Intelligence virtual machine
- 4 GB memory reservation for Tomcat virtual machine

	Messages sent	Total Transactions	Bytes Sent	Bytes Received
Average:	**82,726**	**4,488**	**210,010,982**	**992,883,739**
Std Dev:	**417**	**18**	**1,050,208**	**5,261,200**

Introducing a memory reservation for Tomcat also does not appear to impact system performance.

6.2.4 Test 4 – Large Pages for Tomcat with No Memory Reservation

The fourth series removed memory reservations for both virtual machines:

- Java heap: 1 GB
- Large pages enabled for Tomcat virtual machine
- No memory reservations for either the Web Intelligence or Tomcat virtual machines

	Messages sent	Total Transactions	Bytes Sent	Bytes Received
Average:	**82,360**	**4,472**	**209,086,527**	**988,580,042**
Std Dev:	**865**	**27**	**2,043,044**	**9,630,279**

It is interesting that in this test the results are about the same as for our baseline in test 1 where large pages were not used. UseLargePages appears to do no harm in this case, but doesn't have any particular throughput effect either.

Using the -XX:+UseLargePages parameter for the JVM in combination with memory reservations may have a slight positive effect, but the improvement is less than 1σ, and therefore cannot be considered meaningful or significant.

6.2.5 Test 5 – Second Baseline with 4GB Heap Size

Test 5 creates a new baseline to demonstrate the impact of increasing the Java heap size from 1 GB to 4 GB. Large pages were not used and memory reservations where not enabled.

4GB baseline (Note: Tomcat memory usage doesn't exceed ~2.3GB during testing)

	Messages sent	Total Transactions	Bytes Sent	Bytes Received
Average:	**82,506**	**4,474**	**209,431,636**	**989,818,277**
Std Dev:	**674**	**20**	**1,635,022**	**7,976,144**

In this test, we noted that the Tomcat memory usage did not rise above ~2.3 GB. This is not to say it will never use more, but in this test the 4 GB ceiling was simply not reached given the load that was applied.

6.2.6 Test 6 – 4Gb Heap Size with Starting Heap and Maximum Heap Set Equal

Test 6 set the starting Java heap equal to the maximum heap size to determine if allocating all heap memory at once would have an impact. Under light load, an expanded Java heap appears to help with throughput, though not significantly so. This will, however, become more important under higher loads, and as the memory heap is actually used.

# of Errors	Messages sent	Total Transactions	Bytes Sent	Bytes Received
Average:	**82,409**	**4,479**	**209,301,542**	**989,654,916**
Std Dev:	**662**	**26**	**1,742,314**	**8,450,433**

We also tested scenarios where only the maximum heap size was set. This did not have a significant effect, but in this test, the throughput was a bit lower.

6.2.7 Test 7 – 4Gb Heap with Use of Large Pages After Logoff/Logon (No Reboot)

In test 7, we enabled Large Page support but only performed a logoff and logon instead of a reboot for the associated user account.

	Messages sent	Total Transactions	Bytes Sent	Bytes Received
Average:	**80,689**	**4,390**	**204,808,706**	**968,277,833**
StdDev:	**2,445**	**140**	**6,206,163**	**28,410,399**

6.2.8 Test 8 – 4Gb Heap Size with Use of Large Pages After Reboot

In Test 8, we rebooted the virtual machine after enabling large page support instead of just performing a logoff and logon.

	Messages sent	Total Transactions	Bytes Sent	Bytes Received
Average	**82,000**	**4,461**	**208,357,657**	**985,003,512**
StdDev	**743**	**32**	**1,835,828**	**8,160,845**

This is, in fact, a meaningful difference. The table below compares the pre-reboot results (Test 7) with the post-reboot results (test 8):

Improvement over pre-reboot			
1.63%	1.61%	1.73%	1.73%
pre-reboot against post-reboot median (in σ) σ=743			
-1.766	-2.176	-1.933	-2.050

This is near or more than 2σ, so we can be around ~90-95% confident that this is the result of the configuration change, and shows that it is really important to ensure a reboot of the VM host takes place after setting the XX:+UseLargePages option and the associated “Lock Large Pages” Windows user right assigned to the user running the process. Without it, the effect shows meaningful deterioration of throughput, as well as an increased variation of throughput (as indicated by the 3-4 times larger standard deviations) suggesting inefficiency and potentially even instability under higher loads.

When we compare the results of tests 7 and 8 with the 4GB results without XX:+UseLargePages set (test 5), the conclusions are even more glaring:

Comparing post-reboot to 4GB Xmx:

Improvement over 4GB Xmx			
-0.61%	-0.29%	-0.51%	-0.49%
post-reboot against 4GB Xmx median (in σ) σ=662			
-0.750	-0.630	-0.657	-0.604

Comparing test 8 (Large Page support enabled) with test 5 (no Large Page support), we see a minimal deterioration under light load with the 4GB, no Large Page support baseline. However, this is less than 1σ, so the impact is minimal (~0.5%), suggesting this is not a meaningful difference. The difference we previously observed when comparing the results from test 7 and 8 (showing the importance of a reboot) is even more pronounced when we compare the pre-reboot results of test 7 with the 4GB baseline of test 5:

Comparing pre-reboot to 4GB Xmx:

Improvement over 4GB Xmx			
-2.20%	-1.87%	-2.21%	-2.18%
pre-reboot against 4GB Xmx median (in σ) σ=674			
-2.697	-4.077	-2.827	-2.701

There is a significant and meaningful deterioration in throughput, larger than 2σ (95%+ confidence). This shows that even though we are only seeing a 2% degradation, it is in fact meaningful and results in a significant deterioration in throughput that we can conclude is a result of the changed configuration. Setting the -XX:+UseLargePages flag without performing a reboot of the Windows guest operating system within the virtual machine significantly reduced the throughput when compared to a configuration with a similar Java Heap Size of 4GB RAM and otherwise completely default settings.

7 References

1. SAP Note: 1492000: General Support Statement for Virtual Environments
https://service.sap.com/sap/support/notes/1492000

2. SAP BusinessObjects BI 4 Sizing Estimator
http://www.sdn.sap.com/irj/scn/index?rid=/library/uuid/1055c550-ce45-2f10-22ad-a6050fff97f1

3. SAP BusinessObjects BI 4 Sizing Companion Guide
https://websmp101.sap-ag.de/~sapdownload/011000358700000307202011E/SBO_BI_4_0_Companion_V4.pdf

4. SAP BusinessObjects Business Intelligence Platform Administration Guide
https://help.sap.com/businessobject/product_guides/boexir4/en/xi4_bip_admin_en.pdf

5. Performance Best Practices for VMware vSphere™ 5.0
http://www.vmware.com/pdf/Perf_Best_Practices_vSphere5.0.pdf

6. Enterprise Java Applications on VMware - Best Practices Guide
http://www.vmware.com/resources/techresources/1087

7. VMware Technical Paper : Large Pages Performance
http://www.vmware.com/resources/techresources/1039