

Running SQL Server with Hyper-V Dynamic Memory

Best Practices and Considerations

SQL Server Technical Article

**Authors:** Mike Ruthruff, Xin Jin, Serdar Sutay

**Contributors:** Tom Brown (DBBest Technologies), Guy Bowerman, Lindsey Allen, Prem Mehra

**Technical Reviewers:** Madhan Arumugam, Rohan Kumar, Vinay Kulkarni (IBM), Mike Lawson (IBM), Eduardo Freitas (Hitachi Data Systems)

**Published:** July 2011

**Applies to:** SQL Server versions and editions that support dynamic memory. Details about this support are discussed in detail within this document and references specific to support can be found in the appendix section. **Note:** All tests were performed on SQL Server 2008 R2.

**Summary:** Memory is a critical resource to Microsoft SQL Server workloads, especially in a virtualized environment where resources are shared and contention for shared resources can lead to negative impact on the workload. Windows Server 2008 R2 SP1 introduced Hyper-V Dynamic Memory, which enables virtual machines to make more efficient use of physical memory resources. Hyper-V Dynamic Memory treats memory as a shared resource that can be reallocated automatically among running virtual machines. There are unique considerations that apply to virtual machines that run SQL Server workloads in such environments. This document provides insight into considerations and best practices for running SQL Server 2008 R2 in Hyper-V Dynamic Memory configurations on Windows Server 2008 R2 SP1.

Copyright

This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2011 Microsoft. All rights reserved.

Contents

[Introduction 5](#_Toc299447680)

[Basics of Hyper-V Dynamic Memory and SQL Server Memory Management 5](#_Toc299447681)

[Hyper-V Dynamic Memory 5](#_Toc299447682)

[Memory Distribution Across the Virtual Machines 6](#_Toc299447683)

[Memory Techniques 6](#_Toc299447684)

[Host Memory Reserve 6](#_Toc299447685)

[NUMA Spanning 6](#_Toc299447686)

[SQL Server Buffer Management 8](#_Toc299447687)

[SQL Server Buffer Pool Management 8](#_Toc299447688)

[Windows Memory Thresholds and SQL Server Memory Management 8](#_Toc299447689)

[Using the Lock pages in memory User Right with SQL Server 9](#_Toc299447690)

[Core Test Scenarios and Workloads 10](#_Toc299447691)

[Test Environment / Configuration (Operating System and SQL Server) 10](#_Toc299447692)

[Core Test Scenarios 11](#_Toc299447693)

[Test Workloads 13](#_Toc299447694)

[When Will Dynamic Memory Help My SQL Server Workload? 13](#_Toc299447695)

[Best Practices for Hyper-V Dynamic Memory Configuration with SQL Server 16](#_Toc299447696)

[Enabling Hyper-V Dynamic Memory 16](#_Toc299447697)

[Determining Virtual Machine Startup and Maximum Memory 16](#_Toc299447698)

[Configuring SQL Server Buffer Pool Memory (sp\_configure ‘max server memory’) 16](#_Toc299447699)

[Using Lock Pages in Memory Option with SQL Server 17](#_Toc299447700)

[Impact of Virtual Machine Memory Removal on SQL Server Workloads 17](#_Toc299447701)

[Effect of Removing Virtual Machine Memory on the SQL Server Workload When Not Using Lock pages in memory 18](#_Toc299447702)

[Impact of Removing Virtual Machine Memory on SQL Server Workload When Using Lock pages in memory 19](#_Toc299447703)

[Considerations for Planned and Unplanned Failover 20](#_Toc299447704)

[Unplanned Failover Considerations and Best Practices 20](#_Toc299447705)

[Guaranteeing Virtual Machines Will Be Able to Start After Unplanned Failover 22](#_Toc299447706)

[Memory Rebalancing Behavior, Impact on SQL Server Workload Performance, and Impact on the Host 22](#_Toc299447707)

[Memory Rebalancing 22](#_Toc299447708)

[Live Migration Considerations and Best Practices 25](#_Toc299447709)

[Time to Perform Live Migration When *Not* Using sp\_configure ‘max server memory’ 27](#_Toc299447710)

[Time to Perform Live Migration When Using sp\_configure ‘max server memory’ 29](#_Toc299447711)

[Time to Perform Live Migration When *Not* Using sp\_configure ‘max server memory’ and Not Running SQL Server with Lock pages in memory Enabled 29](#_Toc299447712)

[Impact of Increased Workload on Memory Consumption 30](#_Toc299447713)

[Summary 31](#_Toc299447714)

[Appendix 1 – IBM System x3850 X5 Server 32](#_Toc299447715)

[Appendix 2 – Storage Configuration of the Hitachi Data Systems (HDS) AMS2500 Storage Array 33](#_Toc299447716)

[Appendix 3 – Monitoring and Measure Dynamic Memory for SQL Server Workloads 36](#_Toc299447717)

[Using Performance Monitor 36](#_Toc299447718)

[Other Monitoring Tools 37](#_Toc299447719)

[SQL Server Error Log 37](#_Toc299447720)

[Ring Buffer DMV 37](#_Toc299447721)

[Hyper-V Manager Memory Pressure Reporting 38](#_Toc299447722)

[Windows Event Logs 38](#_Toc299447723)

[Appendix 4 – Additional References 39](#_Toc299447724)

# Introduction

Being able to utilize hardware resources in the most efficient manner is becoming ever more important and is especially true in virtualized environments. In the context of SQL Server, lack of memory resources for a database engine will result in increased I/O that is orders of magnitude slower than accessing memory. In addition, in today’s shared storage environments reducing I/O has a benefit not only to a single workload but across all users of the shared resources.

Hyper-V Dynamic Memory can be leveraged to increase overall virtual machine density and make the most of hardware resources by providing the ability to better leverage memory resources. The ability to more fully leverage memory resources can reduce the I/O capacity needed to support workloads.

This white paper provides an overview of how Hyper-V Dynamic Memory functions specifically in the context of Microsoft SQL Server workloads. It uses results from specific test scenarios and provides best practice guidance for using Hyper-V Dynamic Memory and SQL Server together. The audience for this paper is Windows Server system administrators and database administrators (DBAs) who manage SQL Server in Hyper-V virtual machine environments. The first part of this paper can serve as a short primer on some of the key concepts; however, it is assumed that you are familiar with them; a more in-depth discussion of those topics is beyond the scope of the paper.

For more information about SQL Server workloads within Hyper-V, see the following articles on sqlcat.com: [Running SQL Server 2008 in a Hyper-V Environment – Best Practices and Performance Recommendations](http://sqlcat.com/whitepapers/archive/2008/10/03/running-sql-server-2008-in-a-hyper-v-environment-best-practices-and-performance-recommendations.aspx) (http://sqlcat.com/whitepapers/archive/2008/10/03/running-sql-server-2008-in-a-hyper-v-environment-best-practices-and-performance-recommendations.aspx) and [High Performance SQL Server Workloads on Hyper-V](http://sqlcat.com/whitepapers/archive/2010/05/27/high-performance-sql-server-workloads-on-hyper-v.aspx) (http://sqlcat.com/whitepapers/archive/2010/05/27/high-performance-sql-server-workloads-on-hyper-v.aspx).

# Basics of Hyper-V Dynamic Memory and SQL Server Memory Management

## Hyper-V Dynamic Memory

Hyper-V Dynamic Memory improves the memory efficiency of a Hyper-V host by allocating memory to the virtual machines based on their needs. In order to understand Hyper-V Dynamic Memory, it may be helpful to look at some new concepts introduced by it:

* **Startup Memory:** The starting amount of memory when Dynamic Memory is enabled for a virtual machine. Dynamic Memory ensures that this amount of memory is always assigned to the virtual machines by default. After a virtual machine boots, its memory increases beyond the value of **Startup Memory** as applications inside the virtual machine consume more memory.
* **Memory Demand:** The amount determined by Dynamic Memory to be the memory needed by the applications in the virtual machine. In Windows Server 2008 R2 SP1, this is equal to the total amount of committed memory inside the virtual machine.
* **Memory Buffer:** The amount of memory assigned to the virtual machines in addition to the value of **Memory Demand** to satisfy immediate memory requirements and file cache needs. In Windows Server 2008 R2 SP1, you can configure a percentage value for **Memory Buffer**.
* **Maximum Memory: Maximum Memory** specifies the maximum amount of memory that a virtual machine can grow to with Dynamic Memory.

### Memory Distribution Across the Virtual Machines

If there is enough memory on a host, Hyper-V Dynamic Memory calculates the amount of memory to be assigned to each virtual machine using following formula:

*Virtual Machine Memory = Memory Demand + Memory Demand \* Memory Buffer Percentage*

If the host memory is not sufficient to satisfy the memory needs of all virtual machines, Hyper-V Dynamic Memory uses **Memory Weight** to determine how the memory will be distributed across the virtual machines. **Memory Weight** defines the importance of memory for a virtual machine. Hyper-V Dynamic Memory removes more memory from virtual machines that are configured with lower **Memory Weight** compared to other virtual machines.

### Memory Techniques

Hyper-V Dynamic Memory uses Enlightened Memory Addition functionality of Windows when it adds memory to a virtual machine. This technique is the virtual equivalent of the Hot Add Memory functionality of Windows Server. But compared to it; this technique is lighter weight and faster because it does not require any hardware emulation inside the virtual machine.

When removing memory from a virtual machine, Hyper-V Dynamic Memory employs a technique called Memory Ballooning. In this technique, Dynamic Memory components inside the virtual machine allocate memory in the guest operating system. After the allocation, Dynamic Memory components inside the virtual machine make sure that this memory is not paged out or accessible by any other application inside the virtual machine. After this step, Dynamic Memory guest components coordinate with the Hyper-V host to free the physical memory being used by the virtual machine at the host side. This memory can then be used by other virtual machines in the system.

### Host Memory Reserve

In order to ensure that the host operating system has enough resources to execute virtualization operations, Hyper-V reserves memory to be used exclusively by the host operating system. The amount of memory needed by the host operating system depends on the physical memory size and the hardware capabilities of host.

For systems with *Second Level Address Translation (SLAT)* functionality, you can estimate the host reserve by using the following formula:

*Host Reserve = 384 MB + 30 MB per GB memory of the host*

### NUMA Spanning

Modern systems are partitioned into different nodes in order to satisfy performance needs of applications at the large scale. This architecture is called Non-Uniform Memory Architecture (NUMA). The following figure shows an example of a NUMA system (VM stands for virtual machine).

Computer

CPU

CPU

CPU

CPU

BUS

Memory

VM

VM

VM

VM

VM

VM

VM

VM

Back Channel

Memory

BUS

Node 1

Node 2

In NUMA systems, every node has its own local memory. Memory accesses to local nodes are faster than the memory access to remote nodes. That is, if a virtual machine’s virtual processors are running on Node 1, the system will have better performance if the virtual machine’s memory is also allocated on the physical memory assigned to Node 1.

By default Hyper-V Dynamic Memory tries to ensure that virtual machines are always assigned memory that is local to the nodes they are running on. Also by default, Hyper-V allows virtual machines to allocate memory from other NUMA node in the system.

In order to guarantee that virtual machines do not have remote NUMA node memory access overhead, Hyper-V has a configuration for the host called *NUMA spanning*. If you set this configuration to FALSE, virtual machines can only be assigned memory from the node they are running on. Although this setting ensures that virtual machines do not have remote node memory access overhead, disabling NUMA spanning can also affect the maximum virtual machine density because memory for a virtual machine must reside on a single node.

## SQL Server Buffer Management

In SQL Server, buffer management is a key way to achieve high I/O efficiency. It consists of two components: the buffer manager, which accesses and updates database pages, and the buffer pool, which reduces database file I/O.

### SQL Server Buffer Pool Management

In order to understand how SQL Server responds to memory being removed from the virtual machine by Hyper-V Dynamic Memory and memory pressure in general, it is important to understand how the SQL Server buffer pool works.

The buffer pool within SQL Server is the primary storage mechanism for data and other caches used by the engine (for example, stored procedure cache, lock cache, or connection cache). The amount of memory (in megabytes) in the buffer pool used by an instance of SQL Server can be reconfigured by using the two server memory options, **min server memory** and **max server memory.**

When SQL Server starts, it computes the size of virtual address space for the buffer pool. The upper limit of the virtual address space SQL Server is able to reserve is bound by the maximum physical memory on the machine. SQL Server reserves this computed amount of its process virtual address space (called the *memory target*) for the buffer pool, but it acquires (commits) only the required amount of physical memory for the current load. You can query the bpool\_commit\_target and bpool\_committed columns in the sys.dm\_os\_sys\_info catalog view to return the number of pages reserved as the memory target and the number of pages currently committed in the buffer pool, respectively.

Because the size of virtual address space reserved for the buffer cache is computed at startup time, it is static and does not grow in response to memory being added to the system, for example, through [Hot Add Memory](http://msdn.microsoft.com/en-us/library/ms175490.aspx). If at startup time, SQL Server detects that Hot Add Memory is supported, it increases the size reserved virtual address space for the buffer pool to 16 times the value of **Startup Memory**.

### Windows Memory Thresholds and SQL Server Memory Management

Windows has a notification mechanism, provided by [QueryMemoryResourceNotification](http://msdn.microsoft.com/en-us/library/aa366799(v=vs.85).aspx) in the Windows API, that reports the state of the available memory on the Windows instance. For more information about this function, see [QueryMemoryResourceNotification](http://msdn.microsoft.com/en-us/library/aa366541(v=vs.85).aspx) (http://msdn.microsoft.com/en-us/library/aa366541(v=vs.85).aspx) in the MSDN Library. Low or high memory notifications are raised when the following thresholds are hit:

* Low memory threshold: 32 MB for every 4 GB and capped at 64 MB, so for most systems a notification is sent when 64 MB is available.
* High memory threshold: three times the low memory threshold.

SQL Server only grows buffer pool memory if high memory thresholds are exceeded, and it keeps buffer pool memory constant so long as system memory is between the high and low memory thresholds, unless you explicitly change the configuration.

SQL Server uses the memory thresholds mechanisms to periodically calculate a **Target Memory** value, which is the number of 8-KB pages it can commit without causing paging. Based on this information, SQL Server responds to the current memory situation on a specific system:

* SQL Server memory grows until the system memory exceeds the high memory threshold (that is, while the target value is greater than the actual system memory value).
* If SQL Server determines that there is a low memory condition (that is, that the system memory is below the low memory threshold) it reduces the commit target of the buffer pool and starts reducing the commit target by trimming internal caches. This effect is applied for 5 seconds and then paused for 1 minute. If low memory conditions still exist, the effect is applied again. The goal of this resource monitoring is to avoid growth when SQL Server memory has been paged out.

In addition to reacting to system memory thresholds, a Resource Monitor thread also monitors memory utilization to determine whether the working set for SQL Server needs to be reduced. For example, there are cases in which the working set for SQL Server can be paged out by Windows. The Resource Monitor thread monitors for this, and if it determines that SQL Server process memory has been paged out, it reduces the target server memory for the buffer pool regardless of the current system memory state.

### Using the Lock pages in memory User Right with SQL Server

**Lock pages in memory** is a Windows policy that determines which account can use a process to keep memory allocations pinned in physical memory, preventing the system from paging the data to virtual memory on disk. When the SQL Server service account is granted this user right, buffer pool memory cannot be paged out by Windows.

After you grant the **Lock pages in memory** user right to the SQL Server service account and restart the SQL Server service, the following conditions apply:

* The Windows operating system can no longer page out buffer pool memory within the SQL Server process. However, the Windows operating system can still page out the non-buffer pool memory within the SQL Server process.
* The buffer pool of the SQL Server process still responds to memory resource notification events and it dynamically increases or decreases the target buffer pool memory in response to these events. However, you cannot see memory allocations for the buffer pool that are locked in memory in the following performance counters:
  + The **Private Bytes** counter and the **Working Set** counter in Performance Monitor
  + The **Mem Usage** column on the **Processes** tab in Task Manager

The **Total Server Memory(KB)** counter of the **SQL Server:Memory Manager** performance object accurately represents the memory that is allocated for the buffer pool.

To determine if **Lock pages in memory** is enabled, type the following at a command prompt:

exec ('xp\_readerrorlog 0, 1, ''Using locked pages''')

You can also do this through the Windows Group Policy tool. For more information about enabling the **Lock pages in memory** user right, see [How to: Enable the Lock Pages in Memory Option (Windows)](http://msdn.microsoft.com/en-us/library/ms190730.aspx) (http://msdn.microsoft.com/en-us/library/ms190730.aspx).

You can verify that the user right is granted to the service account and SQL Server is using locked pages for allocations. If the right is enabled and SQL Server is using locked pages output similar to the following is returned.



# Core Test Scenarios and Workloads

We ran a number of different test scenarios to illustrate the concepts and behavior of SQL Server workloads. This section describes our test environment and scenarios.

## Test Environment / Configuration (Operating System and SQL Server)

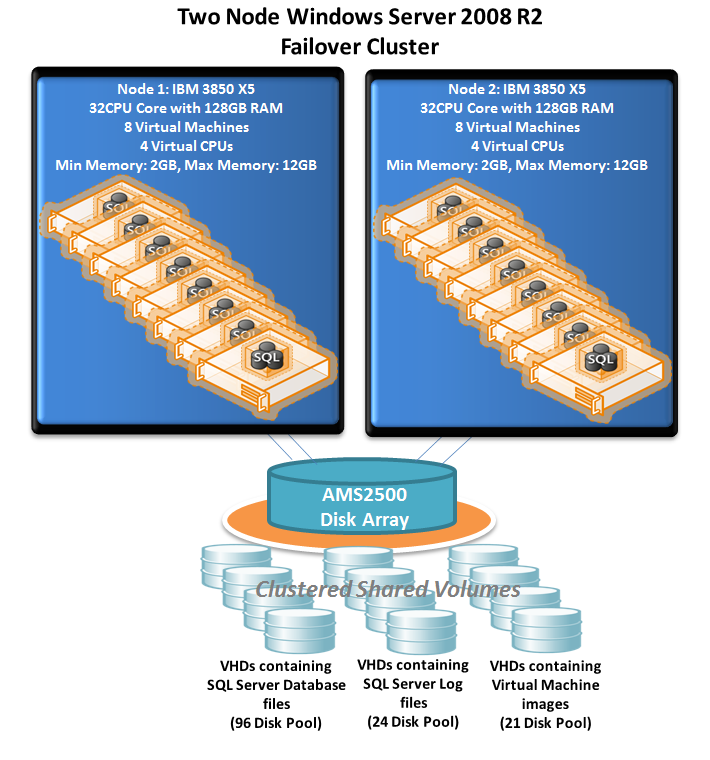
The environment used for the tests described in this paper was as follows.

A two-node Windows Server 2008 R2 failover cluster was configured to host all of the virtual machines. Each cluster node was an IBM System x3850 X5 server with 32 physical processor cores and 128 GB of RAM. As tested in this configuration, each x3850 X5 server was equipped with four 2.26 GHz eight-core processors from the Intel® Xeon® Processor 7500 series. With Intel Hyper-Threading enabled, the x3850 X5 servers each provided 64 logical processors. A dedicated Hitachi Data Systems (HDS) AMS2500 storage system was used to host all virtual machines and SQL Server data. This storage system was configured with three disk pools: one for Virtual Hard Disks (VHDs) that contained the virtual machine images, one for VHDs that contained SQL Server data files, and one for VHDs that contained SQL Server log files.

We used Clustered Shared Volumes (CSVs) to make the virtual machines highly available, enabling live and quick migration scenarios. A total of twelve clustered shared volumes were configured, consisting of:

* Four volumes (LUNs) to store the VHDs that contained the virtual machine images.
* Four volumes to store the VHDs that contained the SQL Server data files for both **tempdb** and the user databases.
* Four volumes to store the VHDs that contained the SQL Server log files.

The following diagram illustrates this configuration (SQL stands for SQL Server).



## Core Test Scenarios

For the purposes of the testing for this information, we ran four core scenarios. These are as follows:

1. **General Characterization of Test Workload:** The goal of this scenario was to characterize the test workload as it relates to increased virtual machine memory to determine what benefit Hyper-V Dynamic Memory has for this workload. For more information, see “When Will Dynamic Memory Help My SQL Server Workload?” later in this document.
2. **Unplanned Failover Scenario:** The goal of this scenario was to test unplanned failovers, including both simulated and hard failures. We ran these tests to determine the considerations in cluster failover scenarios where virtual machines that are starting will be competing for resources with existing SQL Server virtual machine workloads.

Before we simulated the unplanned failover, we ran all 16 virtual machines to steady state of the SQL Server workload (that is, eight virtual machines per host). The core purpose of this scenario was to determine the following:

* What is required to guarantee that all virtual machines will successfully restart on the available cluster node after an unexpected failure has taken the other node of the two-node cluster offline?
* How does Hyper-V rebalance memory as the eight additional virtual machines are started on Node 1 (which hosts eight virtual machines running active workloads)?
* What effect does Hyper-V Dynamic Memory rebalancing have on the SQL Server workloads that are running and on those that are being brought online?

1. **Planned Maintenance Scenario**: In this scenario, we simulated planned maintenance using live migration to move virtual machines. We included this scenario because it required memory to be removed from virtual machines with active SQL Server workloads in order for the eight virtual machines being migrated to come online.

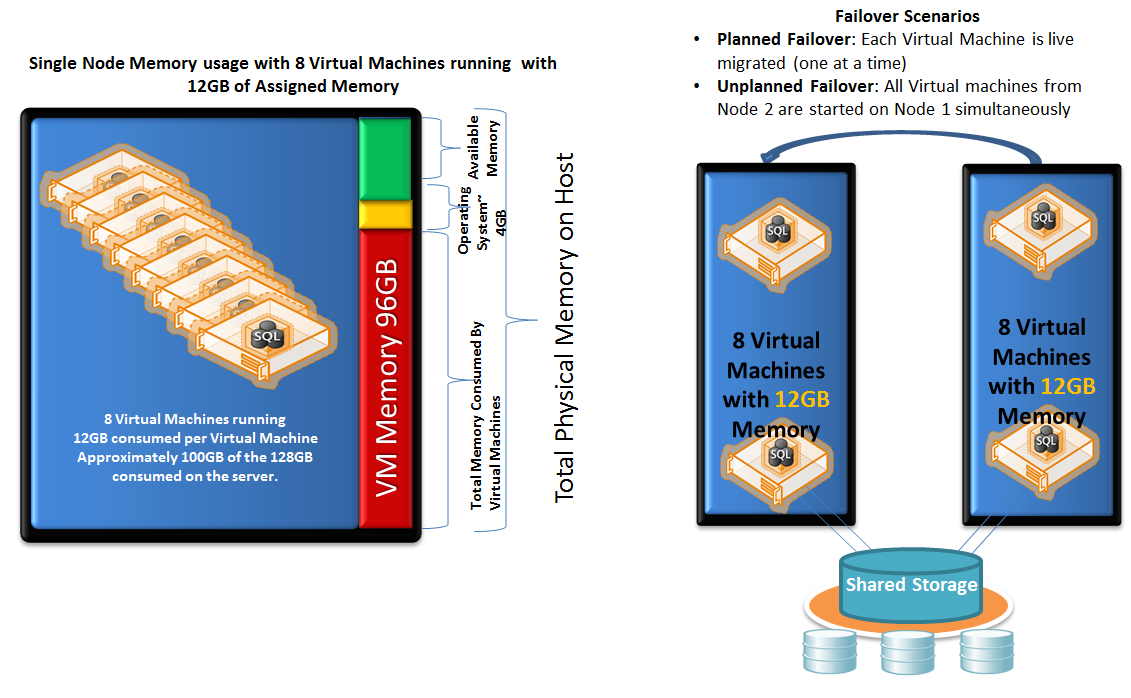
Similar to test #2, for this test the general process was to run load against both nodes until the SQL Server workload had reached steady state and all virtual machines had consumed their maximum configured memory (that is, 12 GB). At that point virtual machines were migrated in a serial fashion from Node 2 to Node 1 using Hyper-V live migration.

The purpose of this scenario was to determine:

* How Hyper-V rebalances the memory when memory must be removed from running virtual machine(s) to free up memory needed to support the live migration of a virtual machine.
* What special considerations there might be for SQL Server scenarios in which these types of live migration are attempted under active workload.
* What effect any rebalancing the Hyper-V Dynamic Memory performs has on the SQL Server workloads that are running and on those that are being migrated.

1. **Simulated “spikes” in the SQL Server Workloads**: The last test scenario tested the behavior of workload “spikes” by increasing workload for certain virtual machines when all virtual machines were running on a single node.

The following illustrations show the memory utilization of a single node prior to scenarios 2 and 3 as well as the failover scenarios from Node 2 to Node 1. In the case of the planned failover, workload is active and at steady state on both nodes when live migration is performed. In the case of the unplanned failover, workload is active and at steady when all virtual machines are brought online simultaneously.



## Test Workloads

The primary workload used for this testing was an online transaction processing (OLTP) workload. This workload simulates a customer-facing brokerage application and has characteristics that are typical of an OLTP workload:

* Test dataset of about 30 GB simulating data for 2,000 customers
* Steady transaction log activity with a write transaction rate of approximately 200 per second and batch request rate of approximately 800 per second
* Steady uniform small block read I/Os (8K) supporting selective lookup type operation
* Periodic flushes of dirty pages through the SQL Server checkpoint mechanism
* Read to write ratio of about 80/20

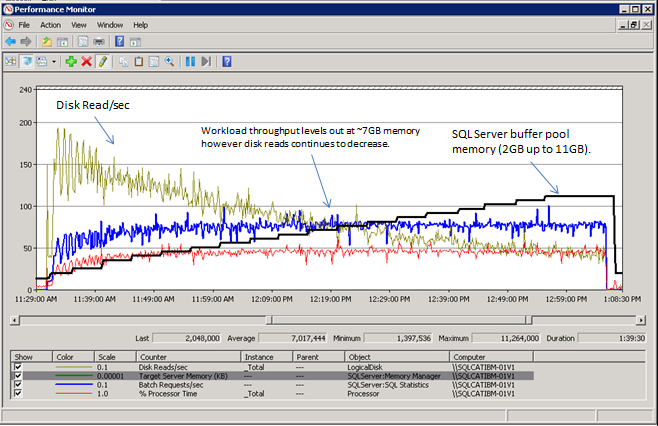
# When Will Dynamic Memory Help My SQL Server Workload?

As mentioned earlier in this paper, one of the most important system resources for SQL Server is memory. Lack of memory resources for the database engine results in increased I/O that is orders of magnitude slower than accessing memory. In addition, in today’s shared storage environments reducing I/O has a benefit not only to a single workload but across all users of the shared resources. There is a balance when additional memory will benefit any given SQL Server workload and determining any benefit usually requires experimentation or testing. One of the key benefits of leveraging dynamic memory is the flexibility to respond to the needs of a particular workload that would benefit from additional memory resources and make the most use out of all physical memory resources on a system.

The benefit of additional memory depends on your workload. Virtual machines in our tests (for more information, see “Core Test Scenarios” earlier in this document) were configured with 2 GB for **Startup Memory** and 12 GB for **Maximum Memory**. For our workload (for more information, see “Test Workloads”), we found workload throughput increased as memory increased from 2 GB to 7 GB-8 GB. Beyond 8 GB, up to 12 GB, there was no measurable gain in workload throughput. However, there was a significant reduction in the I/O needed to support the workload as memory grew from 2 GB to 11 GB.

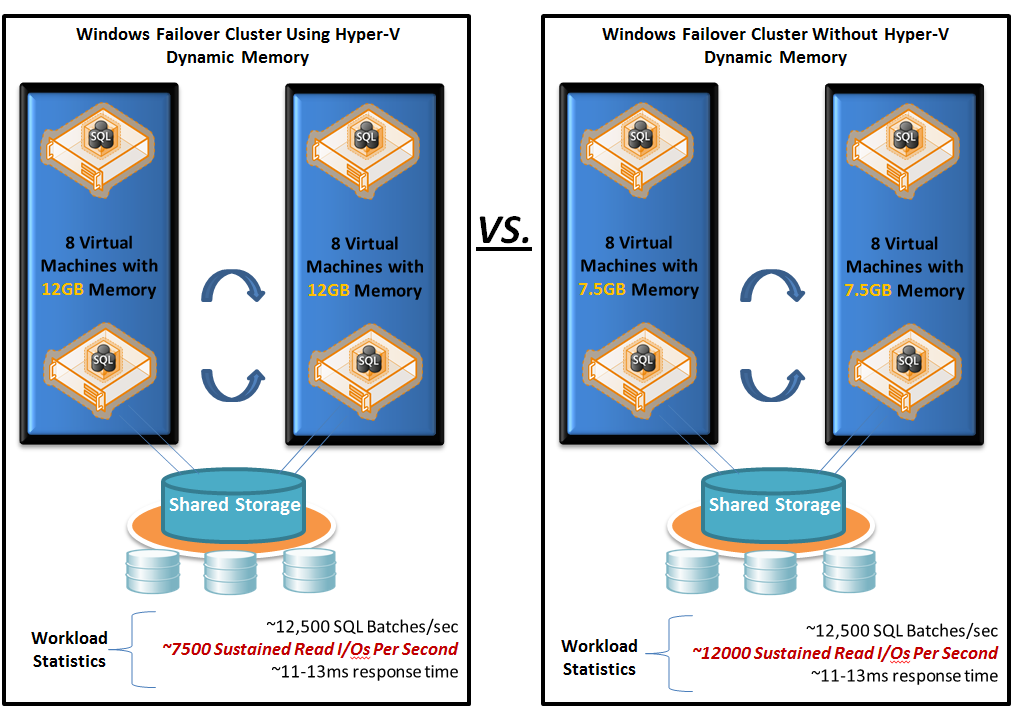
The following illustrates the benefit of additional memory provided to the workload. In this test, we used the **max server memory** configuration option of **sp\_configure** to limit the amount of memory for the SQL Server buffer pool. This setting was increased by 500 MB every 5 minutes up to a maximum of 11 GB (that is, 1 GB lower than the maximum memory allocated to the virtual machine).

Notice that there is a point when **Batch Requests/sec** levels off; however, **Disk Reads/sec** continues to fall, eventually dropping by nearly 50 percent.



So while there was not a continued increase in throughput of the workload, there was a significant decrease in the I/O needed to support the workload. This provides benefit to all users of the shared storage infrastructure. The reason there is not a continued increase in throughput (as shown in the **Batch requests/sec** counter) as the I/O is reduced is that there is a point at which the throughput becomes bound by the application driving database. In our case the batch request throughput represents the highest workload level of the application using the driver configuration that we chose for the testing. The reason we did not configure a higher workload level was that we wanted to utilize a workload that was sized appropriately for the hardware configuration we had in place.

The following figure illustrates the benefits of being able to leverage additional memory to reduce total I/O consumption at the aggregate level.



The main highlights are:

* Without Hyper-V Dynamic Memory the virtual machines would have to be sized with only 7.5 GB of static memory to ensure that all virtual machines could run on a single node in the case of a failover.
* The additional memory provides significant reduction in the number of I/O operations needed to support the same workload throughput.

It should be noted that the benefit depends on your workload. We will expand on our specific test scenarios and workload characteristics throughout the rest of this document.

# Best Practices for Hyper-V Dynamic Memory Configuration with SQL Server

## Enabling Hyper-V Dynamic Memory

* For information about guest operating systems supported and steps to enable Hyper-V Dynamic Memory, see [Hyper-V Dynamic Memory Configuration Guide](http://technet.microsoft.com/en-us/library/ff817651(WS.10).aspx) (http://technet.microsoft.com/en-us/library/ff817651(WS.10).aspx).
* Hyper-V Dynamic Memory is fully supported with SQL Server Enterprise and SQL Server Datacenter. For more information, see the SQL Server Support policy (<http://support.microsoft.com/kb/956893> ).

## Determining Virtual Machine Startup and Maximum Memory

Determining **Startup Memory** configuration across all virtual machines can be summarized as follows.

Setting **Startup RAM** to a lower value allows more virtual machines to start, therefore achieving better consolidation density overall. In SQL Server 2008 R2 and earlier the buffer pool can only grow to a maximum of 16 times the starting memory of the Windows instance. For example, SQL Server running inside a virtual machine configured with 1 GB for **Startup RAM** can never utilize more than 16 GB of memory for the buffer pool. Take this into consideration when you configure **Startup Memory** for a virtual machine. In all of our tests, **Startup Memory** for the virtual machines was configured at 1 GB.

In the unplanned failover scenario, to allow all the virtual machines to be failed over from one host to the other, the total amount of **Startup Memory** combined (both local to the host and virtual machines to be failed over) should be lower than the total physical memory minus the host reserve memory (for more information about host reserve memory, see “Hyper-V Dynamic Memory Basics” earlier in this document).

## Configuring SQL Server Buffer Pool Memory (sp\_configure ‘max server memory’)

The following are the recommended configuration practices for setting **max server memory** when you run SQL Server within a virtual machine using Hyper-V Dynamic Memory:

* Use **sp\_configure** to reduce **max server memory** prior to live migrations when possible. This may not be possible in all cases depending on your rights to SQL Server. Adjusting **max server memory** prior to migrations provides:
  + The highest success rate for migrations.
  + The most predictable impact to SQL Server workload.
  + The most even rebalancing of physical memory assigned to the virtual machines before and after migrations.
* If it is not an option to reduce **max server memory** prior to live migrations, you can use the **Memory Weight** option in Hyper-V Dynamic Memory to provide a high success rate for migrations.

Keep the following points in mind when you configure **max server memory**:

* In general, we recommend that you leave **max server memory** at its default setting, which allows SQL Server to manage memory dynamically. Doing so enables SQL Server to take advantage of any memory added to the virtual machine without needing to be reconfigured.
* If you need to reduce the memory demands of a virtual machine, you can adjust **max server memory** by reducing the memory being consumed by SQL Server. One application of this technique is to guarantee high success rate for live migrations.

## Using Lock Pages in Memory Option with SQL Server

We recommend granting the **Lock pages in memory** user right to the SQL Server service account to provide better stability of the virtual machine workload during periods of time when virtual machine memory is reduced by Hyper-V Dynamic Memory. This memory model is useful and has a positive performance impact because it prevents Windows from paging a significant amount of buffer pool memory out of the process, which enables SQL Server to manage the process of reducing its own working set (through the **Target Server Memory** setting).

Using **Lock pages in memory** is not unique to virtualization scenarios, and it can have an impact on system behavior any time Windows is trimming working sets of running processes. However, for the purposes of this paper we will focus on those scenarios specific to dynamic memory in which memory is removed from a running virtual machine. Any time **Lock pages in memory** is used it is important to understand that because SQL Server memory is locked, and cannot be paged out, your system may experience negative impact to other applications running on the system. In our scenario, as is the case with many SQL Server deployments, the Windows instance was dedicated to running SQL Server. In these cases, regardless of whether you are running within a virtual machine, you may want to protect the SQL Server process memory at the potential expense of other processes on the system.

This paper compares running with and without this privilege in more detail later.

For more information about using **Lock pages in memory** to reduce paging of SQL Server buffer pool memory, see [How to reduce paging of buffer pool memory in the 64-bit version of SQL Server](http://support.microsoft.com/kb/918483) (http://support.microsoft.com/kb/918483).

# Impact of Virtual Machine Memory Removal on SQL Server Workloads

In addition to understanding when additional memory will benefit a SQL Server workload, it is also important to understand the impact when memory is removed from virtual machine on which there are active SQL Server workloads.

When memory is removed from a running virtual machine, available megabytes on the virtual machine are reduced, which can cause low memory thresholds within Windows to be reached. SQL Server subscribes to notifications that monitor these conditions (such as QueryMemoryResourceNotification), and it begins to lower the target server memory for the buffer pool in order to avoid available memory falling below the low memory threshold. The mechanisms for this were discussed earlier in this paper.

One important consideration is to be aware that about the same time the low memory notifications are received, Windows begins to page out the working set from certain processes running within the virtual machine, one of which may be SQL Server. Paging out the working set of SQL Server can have significant performance impact on SQL Server workload because of the overhead of writing and retrieving memory pages from the page file within Windows. This overhead hinders the SQL Server engine’s ability to efficiently reduce and release buffer pool memory back to the operating system during these events. In addition to other techniques described within this paper, another way to mitigate this impact is to place the Windows page file on high performance storage.

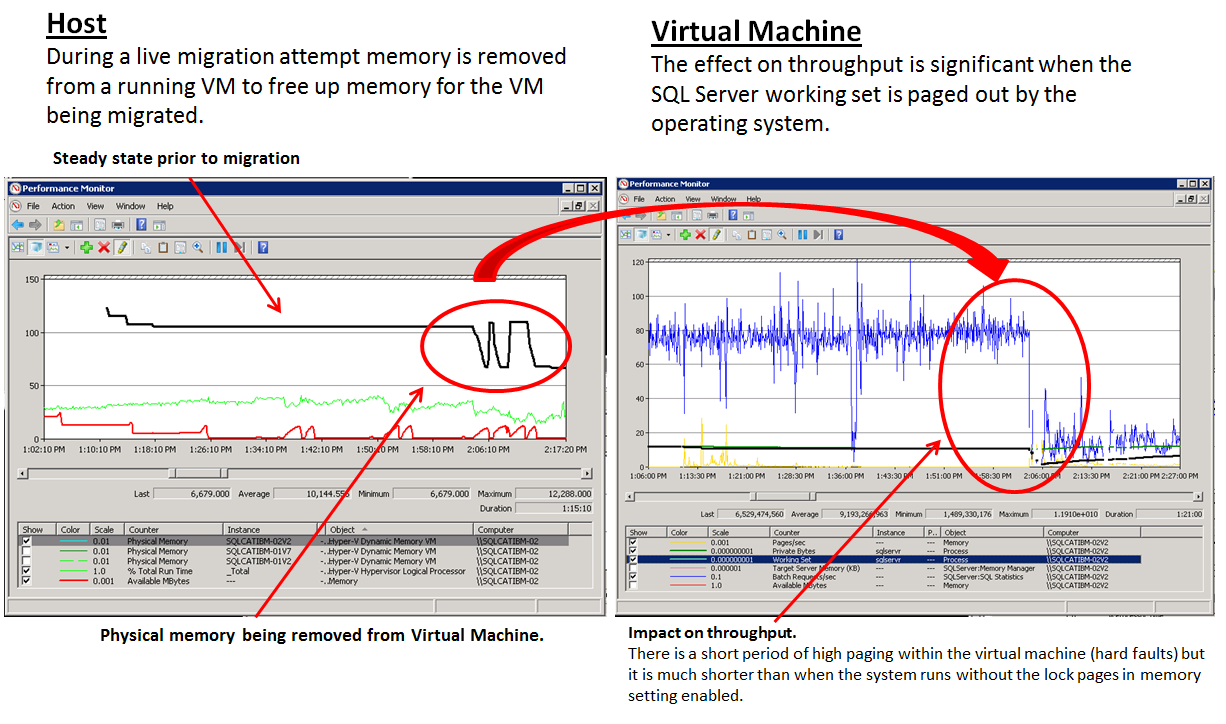
As explained in the previous section, when the SQL Server service account is granted the **Lock pages in memory** user right, Windows cannot page out all buffer pool memory allocations. This results in SQL Server being more efficient in releasing memory from the buffer pool as buffer pool memory is reduced to maintain sufficient free memory within the virtual machine. This is more efficient because SQL Server does not have to wait for pages to be paged back into main memory from the page file when trying to free those buffers.

The example in the following section compares running SQL Server with **Lock pages in memory** enabled to SQL Server running without this setting when virtual machine memory is removed by the Hyper-V Dynamic Memory.

## Effect of Removing Virtual Machine Memory on the SQL Server Workload When Not Using Lock pages in memory

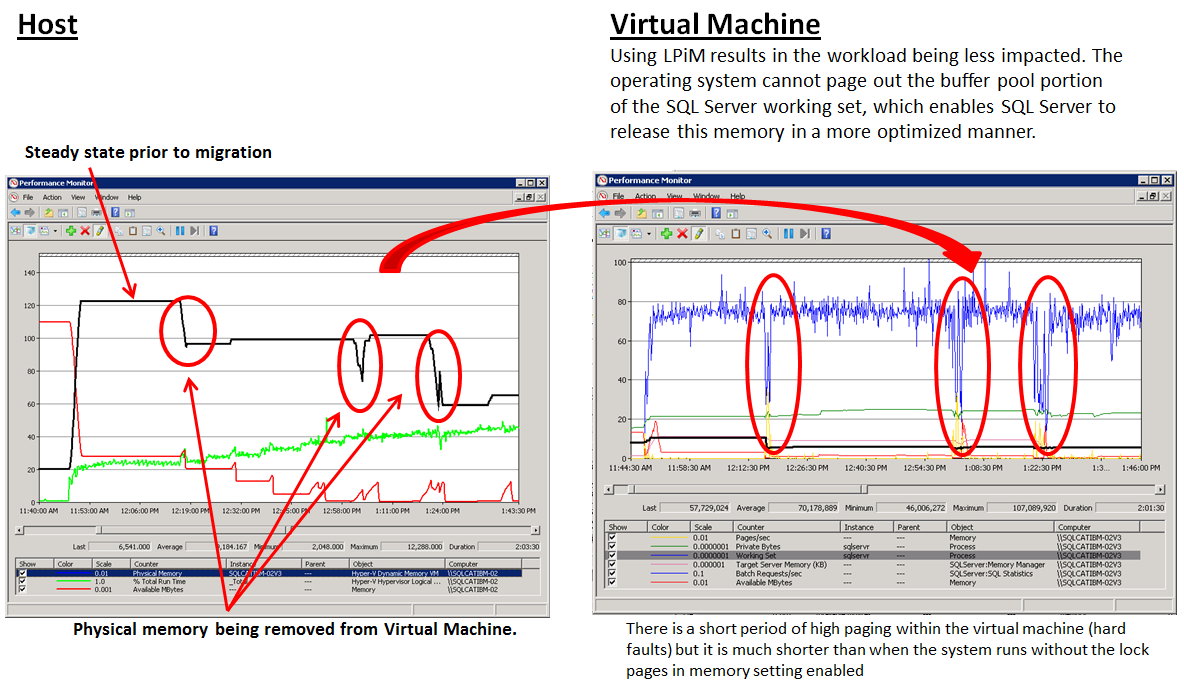
In the following figure, the chart the left side shows the host view of memory removal operations as they are performed against a virtual machine. During these operations there is a decrease in the physical memory allocated to a virtual machine. The chart on the right side shows the impact of the memory removal on the SQL Server workload inside of the virtual machine.

As illustrated, removing memory from the running virtual machine has a significant impact on the throughput of SQL Server (as seen in **Batch Requests/sec**, shown in blue). This impact lasts for a long period of time. In our tests, it took between 1 and 2.5 hours for the workload to return to the previous level of throughput. The reason for the long recovery time is that a significant portion of the working set memory was paged out by Windows because the memory was removed from the virtual machine. This paging out resulted in a significant increase in physical I/O against the Windows page file, which impacted not only the efficiency of workload activity processing in SQL Server but also the efficiency of releasing memory as the SQL Server buffer manager responded to the low memory notifications.



## Impact of Removing Virtual Machine Memory on SQL Server Workload When Using Lock pages in memory

The following charts show the same type of event. However, in this case the SQL Server service is running with the **Lock pages in memory** (LPiM in the figure) user right granted to the service account. As before, removing the memory has a significant impact on the SQL Server workload throughput (as seen in **Batch Requests/sec**, shown in blue). However, this is for a much shorter duration of time compared to the scenario without the **Lock pages in memory** option enabled. Although Windows can reduce the working set for other processes, as well as non-buffer pool memory within SQL Server, Windows cannot page out the working set of the SQL Server buffer pool. As a result SQL Server is more efficient in releasing buffer pool memory as it responds to the low memory notifications generated.



It is important to note that this behavior is not unique to Hyper-V scenarios in which SQL Server is running within a virtual machine. This behavior can occur in any situation in which available memory on the system is low enough to trigger low memory notifications and the working set trimming performed by Windows when these occur. In these types of scenarios granting the **Lock pages in memory** user right to the SQL Server service account will be beneficial.

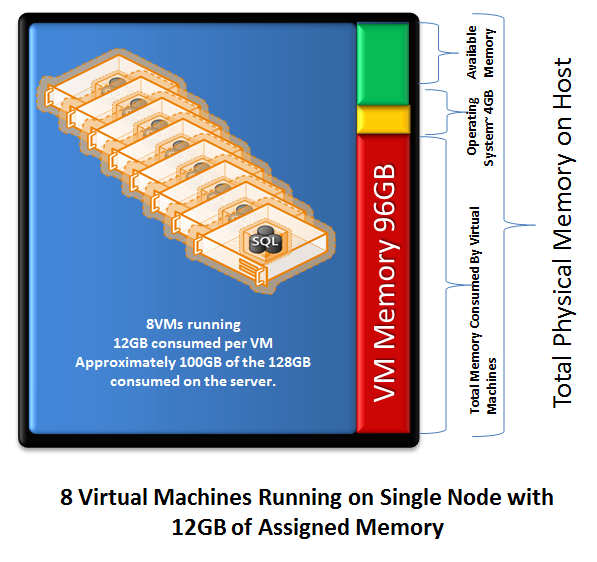
# Considerations for Planned and Unplanned Failover

## Unplanned Failover Considerations and Best Practices

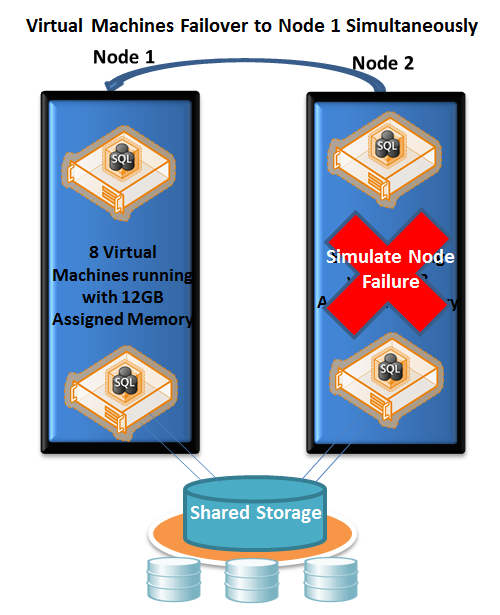
One of the most important considerations for any system is high availability. It is critical that in the event of an unplanned failover all virtual machines that failed over from a failed node are able to successfully come online. In addition, it is helpful to understand what to expect when a set of virtual machines coming online must contend with existing virtual machines for memory resources, which requires memory removal/add operations across the virtual machines.

We performed tests in which all virtual machines were run to steady state across both nodes of the two-node cluster (eight virtual machines per cluster node) and a hard failure of a single node was simulated, which resulted in eight virtual machines being started simultaneously on the node that was still running. The sequence of events was as follows:

1. SQL Server workload is run to steady state within each virtual machine on both host cluster nodes (eight virtual machines per node) and all virtual machines consume their maximum configured memory (12 GB). At this point the memory consumption on the host of a single cluster node is as follows.



1. A hard failure of Node 2 is simulated by powering off the physical host server.
2. When this occurs, all eight virtual machines that were running on this node must now be restarted on Node 1 of the cluster. Each virtual machine is configured to use a minimum starting memory of 2 GB, which means that to start all eight virtual machines, the host needs to have 16 GB of free memory.



Some of the key considerations in this scenario are as follows.

### Guaranteeing Virtual Machines Will Be Able to Start After Unplanned Failover

In order to guarantee that all virtual machines will be able to start successfully after a failover event, the cluster node on which these are being started must have enough available memory to meet the **Startup Memory** demand of the virtual machines being started. In our scenario (the two-node cluster), Node 1 needed to have 16 GB (eight virtual machines \* 2 GB per virtual machine). In clusters with more than two nodes this calculation gets a little more complex. In fact in larger clusters it may not be feasible, or desirable, to have all virtual machines ever start on a single node. In any case enough memory should be left available on a cluster node to fulfill the total amount of **Startup Memory** required by the number of virtual machines expected to be started on that node upon a failure.

It is important to note that this is not a hard requirement. If for some reason there is not enough free memory at the time of virtual machine startup, the Hyper-V manager will provide a “best effort” attempt to start the virtual machine by attempting memory removal operations from the other running virtual machines. It will continue attempting to start a virtual machine based on the number of retry attempts configured within the Failover Cluster Manager. So although this is not a hard limitation, it is considered a best practice to configure **Startup Memory** and maximum virtual machine memory settings so that there will always be enough physical memory to support the memory needed to start virtual machines in the case of an unplanned failover.

## Memory Rebalancing Behavior, Impact on SQL Server Workload Performance, and Impact on the Host

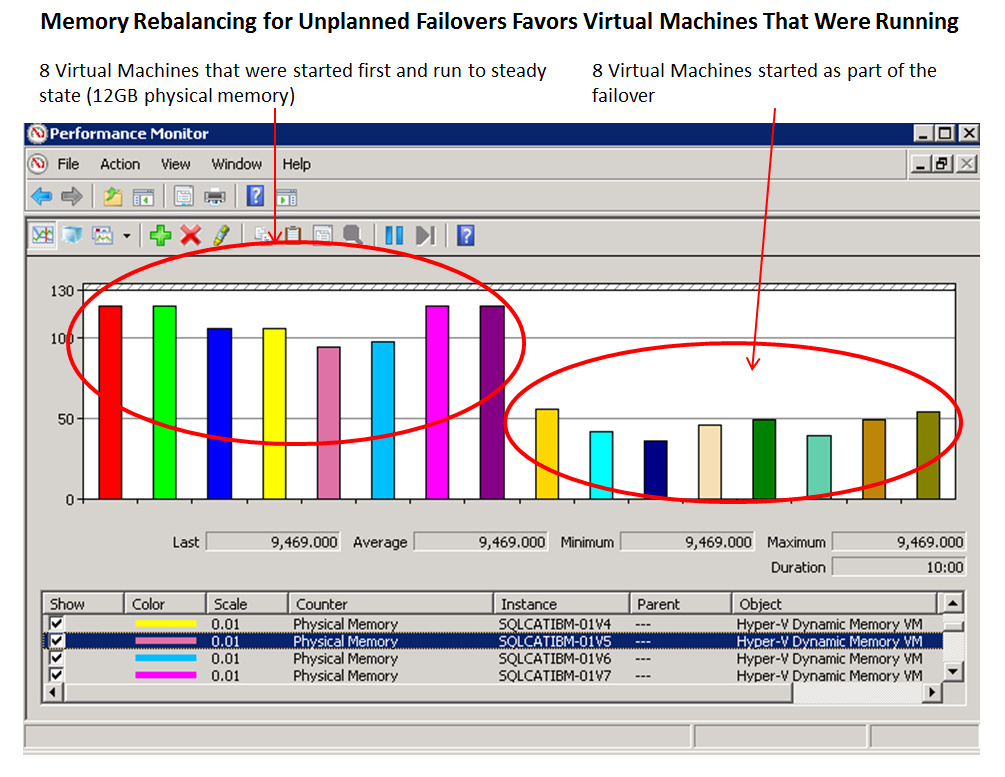
The most important consideration is to ensure that virtual machines will be able to come online so that high availability can be maintained. A natural second question would be, “what can I expect as far as performance knowing that there will be contention for memory resources after the failover?” In our testing we observed the following.

### Memory Rebalancing

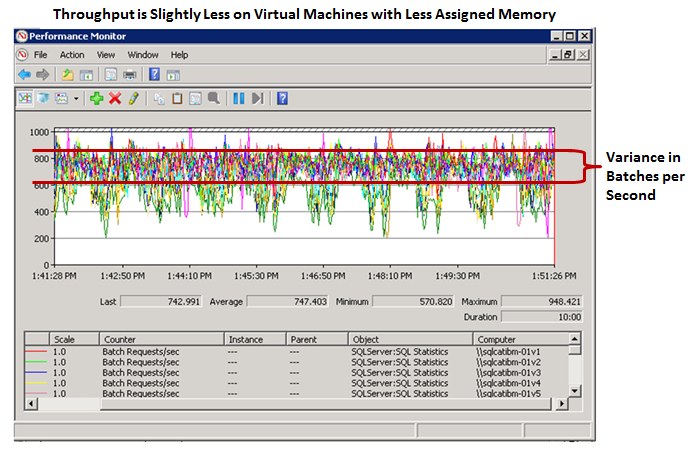
In the unplanned failover scenarios we found that as the virtual machines being started ramped up to workload-steady state, the balance in assigned memory across the virtual machines favored the virtual machines that were already running on the first node (Node 1 in our scenario). However there are several important items to note:

1. The difference in assigned memory between the virtual machines after they reach steady state is nondeterministic. This depends on the nature of the SQL Server workload as well as the nature of the memory removal and add operations performed by the Hyper-V Dynamic Memory. We observed significant differences between those virtual machines with the highest assigned memory and lowest assigned memory (up to 8 GB). This is expected behavior.
2. Even with the imbalance in memory, SQL Server workloads within the virtual machines that failed over were able to sustain workload throughput at a reasonable level, with the impact being somewhere between 10 and 25 percent lower throughput. Proper I/O sizing was the key to our achieving this.
3. The biggest impact on performance was the increase in I/O at the aggregate level (host) due to less overall memory for the 16 virtual machines, because they were running on only a single cluster node.
4. Related to #1, the nature of the memory removal and add operations is influenced by the memory demand of the virtual machines, the memory buffer setting being used, and the memory weight of the virtual machines. This means:
   1. A larger memory buffer setting (20 percent versus 5 percent) resulted in slightly closer steady state of assigned memory after rebalancing; however, the change was not significant. Also, while an increased memory buffer might be desirable in this scenario it may not be in general. In addition, the behavior of a larger memory buffer depends somewhat on workload.
   2. When we adjusted the memory weight of virtual machines so those that failed over had a higher weight, the Hyper-V Dynamic Memory assigned more memory to the virtual machines that were being brought online. The increased memory needs, of course, required the other virtual machines to release memory, but you can use this technique to selectively increase assigned memory to certain virtual machines.

Our general observation related to rebalancing was that after steady state was reached, this pattern of assigned memory was very consistent when an equal memory weight was assigned to all virtual machines.

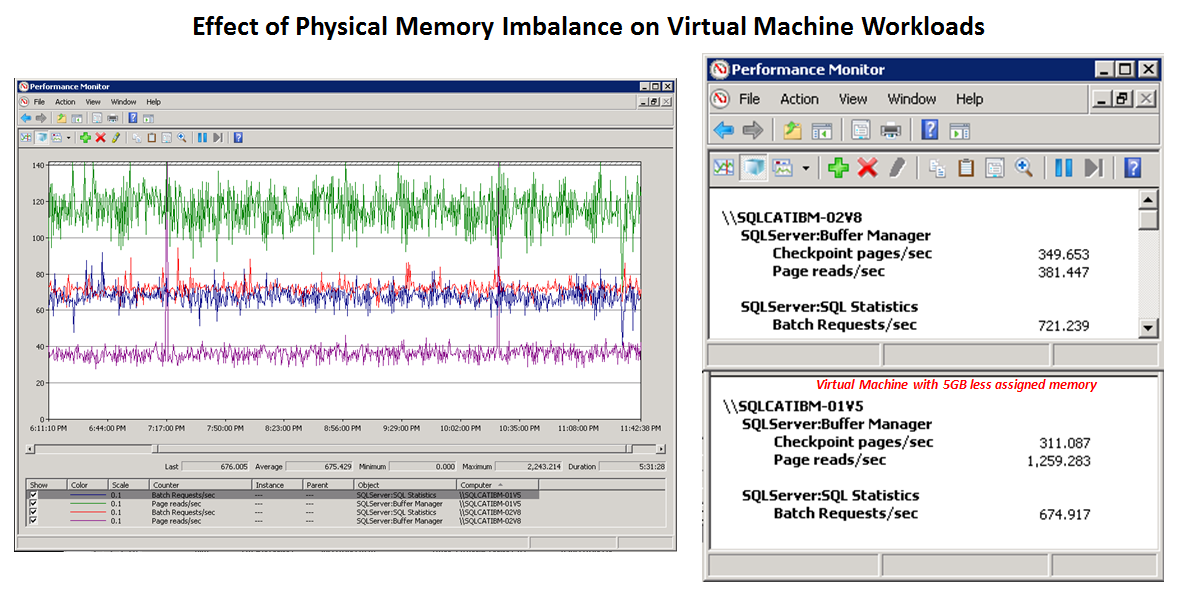


The following diagram shows the throughput of **Batch Requests/sec** across all of the virtual machines after a steady state of assigned memory had been reached as illustrated in the preceding figure. The periodic drops in throughput are correlated with checkpoint activity within the virtual machines and are expected.



At times when there was an imbalance between the virtual machines, the impact was mainly on the amount of physical I/O being issued by the virtual machines. In these cases throughput was within a 10-20 percent variance.

In the following illustration, the physical I/O operations (**Page Reads/sec**) and throughput (**Batch Requests/sec**) are represented for two virtual machines, one with 5 GB less assigned memory than the other.



As illustrated, the virtual machine with less memory has slightly less throughput. However, this drop in throughput is in the range of only 10 percent, even when it has to issue nearly three times as many I/Os, because the storage is still able to provide healthy responses times. This is one of the reasons that proper sizing of storage is so important for SQL Server workloads.

## Live Migration Considerations and Best Practices

Live migrations are relatively common operations that are performed in order to move virtual machines from one host node to another with no downtime to the virtual machine. These can be done for any number of reasons, the most common of which is to perform maintenance on a host node or to rebalance resource utilization.

There are several considerations when performing live migrations on systems with active SQL Server workloads. As described previously, in order for a virtual machine to be migrated from one node to another within the cluster, there must be enough memory available so that the virtual machine can be migrated with the current amount of assigned memory. For scenarios in which there are active SQL Server workloads, there may be times in which Hyper-V Dynamic Memory is not able to secure enough physical memory (by removing memory from running virtual machines) to provide the memory needed for virtual machine being migrated. In order for this to succeed, Hyper-V Dynamic Memory has to secure memory needed to support the migration of the virtual machine within a two-minute timeout interval.

The following illustrates the migration scenario performed.



Some practices that provided a positive impact on the ability to perform these migrations successfully included:

* **Using sp\_configure ‘max server memory’ to reduce the memory utilization of SQL Server prior to migrating the virtual machine.** Using this setting to lower the memory utilization of SQL Server results in SQL Server immediately releasing buffer pool memory so that total memory does not exceed this threshold. If you run this stored procedure before you perform a live migration, SQL Server does not attempt to increase target memory consumption as memory from the virtual machine fluctuates during memory removal operations. This results in the most orderly rebalancing of memory across virtual machines and also reduces the impact of these operations on SQL Server workloads for the virtual machines that have memory removed. When possible, we recommend that you use this approach.
* **Increasing the memory weight of the virtual machine being migrated prior to initiating the migration.** If it was not possible to adjust **max server memory**, another technique that worked well in our testing was to increase the memory weight of the virtual machine being migrated prior to the migration attempt. If you increase the memory weight of this virtual machine, the virtual machine is favored when memory removal and adding operations are performed on the running virtual machines.
* **Running with NUMA spanning disabled increases the success rate for live migrations when under heavy workloads.** When NUMA spanning is disabled, a migrated virtual machine is assigned to a specific NUMA node on the target cluster node to which it is being migrated. This means that any memory removal operations that are needed have to be performed only on the virtual machines that reside on the same NUMA node. We found that not having to perform as many memory removal operations increased efficiency, which made the live migrations more successful.

An additional consideration is that when there are many memory removal or addition operations being performed on a single system, using the **max server memory** technique generally results in a shorter time to perform the migrations as well as a shorter time period for the workloads to return to a steady state (meaning there are no more memory removal or add operations).

The examples contained in the following subsections illustrate the time it takes to perform migrations and the time to reach a steady state of virtual machine memory consumption after the migrations are performed.

### Time to Perform Live Migration When *Not* Using sp\_configure ‘max server memory’

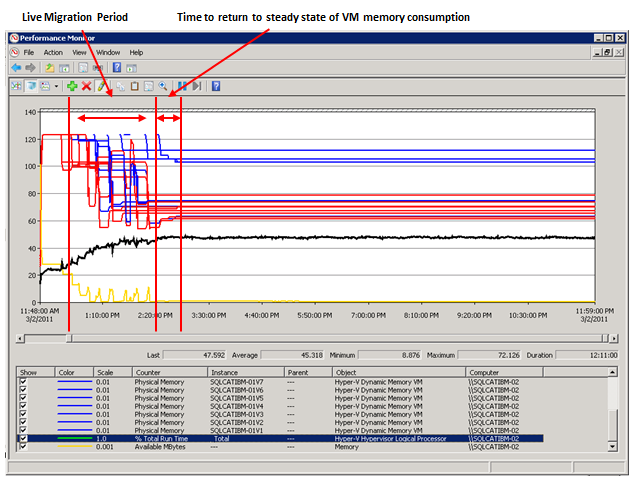
The following illustrates the time to perform a live migration of eight virtual machines from Node 2 to Node 1 while there is active full workload against all 16 virtual machines across both nodes.

There are two phases represented in the diagram:

1. Phase 1 is the period of time it takes to perform the migration of all eight virtual machines from Node 2 to Node 1 of the Windows failover cluster (host).
2. Phase 2 is the time it takes for SQL Server workloads and assigned memory across all 16 of the virtual machines to reach a steady state.

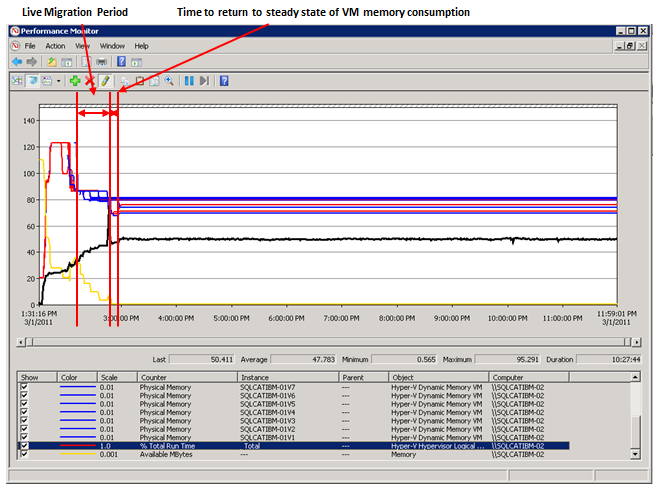
The following charts illustrate these measurements:

* **Red lines**: The assigned memory to the virtual machines running on Node 1 of the cluster.
* **Blue lines**: The assigned memory to the virtual machines that are migrating from Node 2 to Node 1. Notice that these begin to be rendered during the migration period at the point in time the virtual machine being migrated is online and running on Node 1.
* **Yellow line**: The available memory on Node 1 of the cluster (this is the host partition).
* **Black line**: Total CPU consumption on Node 1 of the cluster.



Notice that after steady state is reached, several virtual machines have significantly more assigned memory. These virtual machines were migrated from Node 2 to Node 1. This is consistent and expected behavior, because the virtual machines that are last to be migrated onto Node 1 are less impacted by memory removal operations and maintain nearly the total assigned memory at the time of the migration.

### Time to Perform Live Migration When Using sp\_configure ‘max server memory’



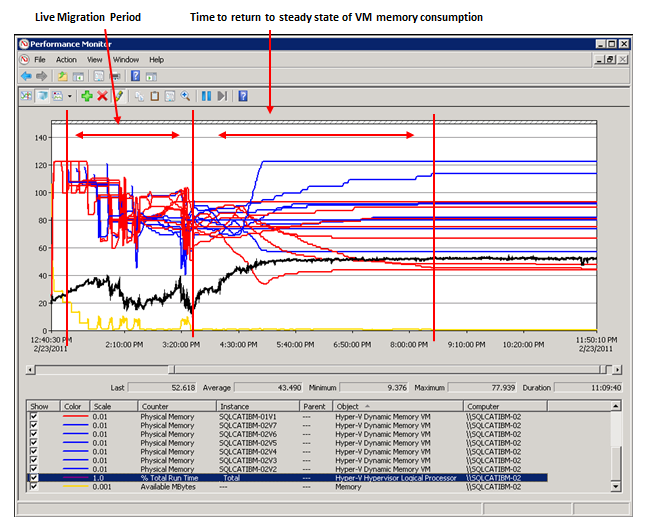
In this case there is a clear difference; not only in the time it took to perform migrations and reach a steady state, but also in how balanced the assigned memory across the virtual machines was after the migration. Using **sp\_configure** to reduce **max server memory** prior to the migrations results in SQL Server releasing memory resources before the migration and, more importantly, during the migrations, Hyper-V Dynamic Memory is not under pressure from the SQL Server workloads as memory removal operations are performed. The result is a much smoother overall migration. It should also be noted that one of the reasons there is such a difference between this and the previous example is that in this case the SQL Server workloads are active at high throughput levels during migrations. This may not be the case on all systems; the previous scenario can be considered a near worst-case scenario.

### Time to Perform Live Migration When *Not* Using sp\_configure ‘max server memory’ and Not Running SQL Server with Lock pages in memory Enabled

In the previous section where we discussed the impact of memory removal operations on a SQL Server workload running within a virtual machine, we illustrated the impact of running without **Lock pages in memory** compared to running with **Lock pages in memory**. It is important to note here that in both scenarios the SQL Server services running within the virtual machine are running with the **Lock pages in memory** user right granted to the SQL Server service account.

The following illustration shows the scenario of not using **sp\_configure** to reduce **max server memory** when the SQL Server processes within the virtual machines are not using **Lock pages in memory**. In this case the increased impact of the memory removal operations on the performance within the virtual machine not only impacts the workload running within the virtual machines but also negatively impacts the time it takes to perform the migrations and the time it takes to return to steady state of memory utilization across all virtual machines.

This illustration shows this behavior.



# Impact of Increased Workload on Memory Consumption

We tested one last scenario to determine whether certain virtual machines would begin acquiring more memory when workload levels against these virtual machines were increased. This was run to loosely simulate “spikes” in workload activity. Contrary to what you might expect, we did not observe an increase in memory consumption in these scenarios. None of the tests resulted in memory being removed by certain virtual machines so that it could be added to the virtual machine running the increased workload. The explanation for this is provided in the previous section, which described the buffer manager behavior with respect to increasing the target memory for the buffer pool. SQL Server increases the target memory for the buffer pool only if it receives high memory notifications from Windows, there is enough workload demand, and no explicit limit on **max server memory** has been reached. Even with an increase in workload SQL Server does not raise this target, because it risks putting the overall instance into a memory pressure situation.

If you want to provide more memory to a particular virtual machine for periods of increased workload, you can use the memory weight setting in Hyper-V to provide a particular virtual machine higher priority to memory resources.

# Summary

This paper provides some insight into the considerations and practices that should be followed when running SQL Server workloads in virtual machines that use Hyper-V Dynamic Memory. There is a clear benefit provided by the ability to leverage Hyper-V Dynamic Memory, and this provides a great mechanism to enable better overall performance as well as more efficient use of storage resources for SQL Server scenarios.

Here are our observations and recommendations for such deployments:

* Run SQL Server with the **Lock pages in memory** user right to provide better stability to the SQL Server workload during memory removal operations.
* Ensure that the total of **Startup Memory** settings for the virtual machines is configured to be lower than host’s physical memory so that all virtual machines can start in the event of an unplanned failover.
* When possible, use SQL Server memory management tools (**sp\_configure ‘max server memory’**) to enable more orderly live migrations.
* Disabling NUMA spanning has some advantages to live migration scenarios and may offer some performance advantage due to locality of memory allocations (this is not illustrated or measured as part of this paper, however).
* Use memory weight to selectively adjust assigned memory to certain virtual machines dynamically.
* A recommended starting point for the memory buffer setting is 5 percent; you can adjust it based on your memory rebalancing needs.
* Many of the optimal settings and behaviors depend on your workload. You may need to perform further experiments and tests to determine what works best for any given deployment.

# Appendix 1 – IBM System x3850 X5 Server

The IBM System x3850 X5 server is the fifth generation of the Enterprise X- Architecture, delivering innovation with enhanced reliability and availability features to enable optimal performance for databases, enterprise applications, and virtualized environments.

Features of the x3850 X5 include:

* Increased performance with more processors and memory than ever before on x86.
* Intel Xeon E7 processors with up to 10-cores and Hyper-Threading.
* Intel Xeon X7500 series processors with up to eight cores and Hyper-Threading.
* Up to eight sockets and 128 DIMMs with QPI scaling for larger databases, enterprise, and mission-critical workloads.
* Up to 64 DIMM slots per chassis delivering up to 1,024 GB of high speed PC3-10600 double data rate (DDR3) memory.
* An additional 32 DIMMs per chassis with MAX5 memory drawer.
* Memory reliability and availability with Memory ProteXion with Chipkill, memory mirroring, memory sparing, Intel SMI Lane Failover, SMI packet retry, and SMI Clock failover.
* Low-power cost-effective memory with Advanced Buffer eXecution chip.
* High performing databases and fast time to value for database workloads.
* Advanced networking capabilities with Emulex 10Gb Ethernet Adapter, including support for Fibre Channel over Ethernet (FCoE) as a future feature entitlement upgrade.
* Integrated Management Module (IMM) for enhanced systems management capabilities.
* Serial Attach SCSI (SAS) plus RAID to maximize throughput and ease installation of a RAID card.

IBM System x3850 X5 at-a-glance guide:

<http://www.redbooks.ibm.com/abstracts/tips0817.html?Open>

# Appendix 2 – Storage Configuration of the Hitachi Data Systems (HDS) AMS2500 Storage System

The following section describes the high level storage configuration used for this testing in more detail.

As noted earlier in the paper, a dedicated Hitachi Data Systems (HDS) AMS2500 storage system was used to host all virtual machines and SQL Server data. This storage system was configured with three disk pools (referred to in this appendix as Data Pools): one for Virtual Hard Disks (VHDs) that contained the virtual machine images, one for VHDs that contained SQL Server data files, and one for VHDs that contained SQL Server log files.

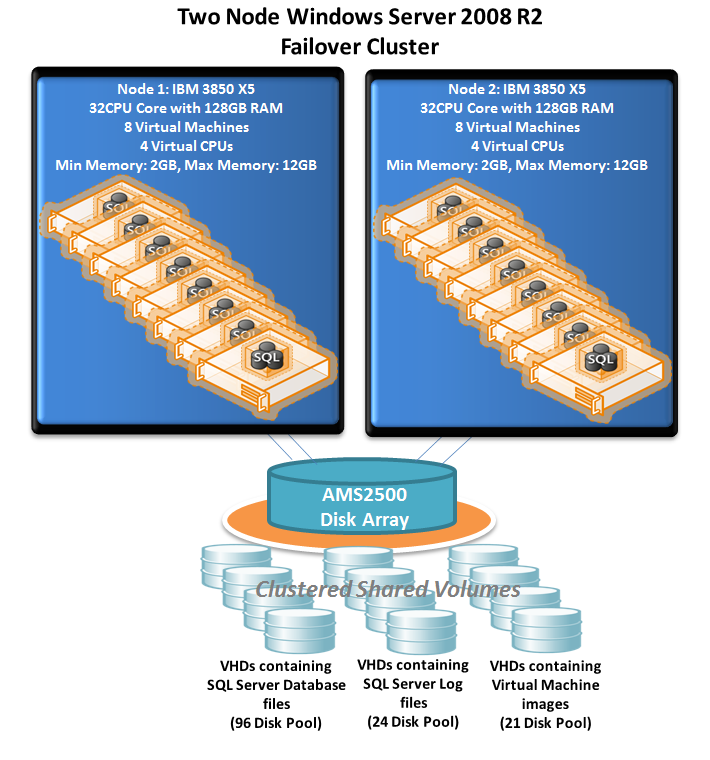
Clustered Shared Volumes (CSVs) were used to make the virtual machines highly available, enabling live and quick migration scenarios. A total of twelve clustered shared volumes were configured, consisting of:

* Four volumes (LUNs) to store the VHDs that contained the virtual machine images.
* Four volumes to store the VHDs that contained the SQL Server data files for both **tempdb** and the user databases.
* Four volumes to store the VHDs that contained the SQL Server log files.

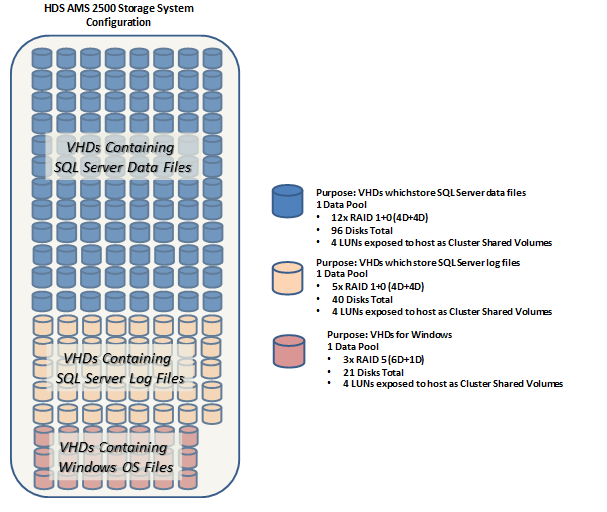
Other details about the configuration are as follows:

* Each physical node was configured with four 4Gb/s Host Bus Adapters (HBAs) directly to 4Gb/s fiber channel ports on the storage system.
* The AMS2500 was configured with 32 GB of cache (16 GB usable).
* The AMS2500 was configured with 146 GB 15K Rpm Disks.
* Thin provisioning is used; however, full database restores and an initial test run result in fully committing capacity at the physical level.

Logical configuration of the cluster and storage was as follows.



The physical and logical disk configuration of the array is shown in the following illustration.



# Appendix 3 – Monitoring and Measure Dynamic Memory for SQL Server Workloads

This appendix focuses on the data that you collect to monitor performance of the virtual machine workloads, performance of the host partition, and the activity of the Hyper-V Dynamic Memory Manager. You may find this information useful as a background for the discussion of the testing results.

The sections here provide a high-level overview of the data that you can use to monitor and measure these types of scenarios. These sections are not intended to be exhaustive explanations of the tools but rather to provide a starting point. We provide pointers to more in-depth references when possible.

## Using Performance Monitor

Performance Monitor, a tool in Windows, includes the counters we watched to gather information about SQL Server in our virtualized scenarios. These are the primary counters you use to measure workload performance, characteristics of memory utilization within a virtual machine, and resource consumption and performance on the host. These are a small subset of the available counters. They are the counters we used most frequently in the test scenarios to measure performance and observe memory rebalancing behavior.

|  |  |  |
| --- | --- | --- |
| **Counters measured from** | **Counter or Counter Set** | **Used for** |
| **Guest Partition** | **Batches/sec** | Measure throughput of the SQL Server workload. |
|  | **Pages/sec** | Provides an idea of the amount of hard page faults to/from the Windows page file. It indicates how significant the impact of memory removal operations is on SQL Server workload within a virtual machine. |
|  | **SQL Server: Memory Manager: Target Server Memory** | Represents the current target memory of the SQL Server buffer pool manger. Changes in this value are the best indication of when SQL Server is reacting to memory notifications being raised by Windows. |
|  | **SQL Server: Memory Manager: Total Server Memory** | Measures the current size of the SQL Server buffer pool. |
|  | **Process: Working Set: SQL Server process** | Measures the current size of the working set for SQL Server. This has limited use because when **Lock pages in memory** is in use, the counter does not reflect the actual working set. |
|  | **Memory: Available MB** | Measures the available memory as measured within the virtual machine. It is useful for correlating with changes in SQL Server **Target Server Memory**. |
|  |  |  |
| **Root partition** | **Hyper-V Dynamic Memory Manager VM: Guest Visible Physical Memory** | This counter represents the amount of memory visible in the virtual machine. |
|  | **Hyper-V Dynamic Memory Balancer: Available Memory** | This counter represents the amount of memory left on the node that can be assigned to virtual machines. This counter is shown as **System Balancer** if NUMA spanning is enabled. It displays memory values for each NUMA node when NUMA spanning is disabled. |
|  | **Logical Disk Counters** | Used to measure I/O being issued at the host level (aggregated across all virtual machines). |
|  | **Hyper-V Hypervisor Logical Processor: % Total Run Time** | Measures the total CPU consumption on the host. |
|  | **Hyper-V Hypervisor Logical Processor: % Guest Run Time** | Measures the total CPU consumption of guest virtual machines running on the host. |
|  | **Hyper-V Hypervisor Virtual Processor: % Total Run Time** | Measures the CPU utilization for each virtual CPU within a guest virtual machine. |
|  | **Memory: Available MB** | Same measure as the above but at the host level. Hyper-V Dynamic Memory attempts to maintain a certain amount of free memory on the host for other processing. |

## Other Monitoring Tools

In addition to Performance Monitor, there are a number of other tools available that provide insight into system performance and events that are related to Hyper-V Dynamic Memory. They include DMVs, SQL Server error logs, and Hyper-V Manager memory weight metrics that are exposed in the user interface (UI). This is not a comprehensive list of the data that is available; it is provided as a starting point to help you find the best information for your organization’s needs.

### SQL Server Error Log

The SQL Server error log is one of the best places to start when you are investigating any issue related to SQL Server. When you are looking at memory, you can use it to determine whether events have occurred that have caused the working set of the SQL Server process to be paged out. When SQL Server detects paging out, an event is entered into the error log. Here is an example of this type of message.

**LogDate        ProcessInfo        Text**

2011-03-15 09:03:45.490        spid1s        A significant part of sql server process memory has been paged out. This may result in a performance degradation. Duration: 0 seconds. Working set (KB): 96212, committed (KB): 11538048, memory utilization: 0%.

2011-03-15 09:09:14.040        spid1s        A significant part of sql server process memory has been paged out. This may result in a performance degradation. Duration: 328 seconds. Working set (KB): 3998528, committed (KB): 11569400, memory utilization: 34%.

2011-03-15 09:34:30.550        spid1s        A significant part of sql server process memory has been paged out. This may result in a performance degradation. Duration: 0 seconds. Working set (KB): 39556, committed (KB): 11442552, memory utilization: 0%.

2011-03-15 09:40:01.020        spid1s        A significant part of sql server process memory has been paged out. This may result in a performance degradation. Duration: 331 seconds. Working set (KB): 2027596, committed (KB): 11577064, memory utilization: 17%.

In addition to messages like this one, the error log can tell you whether **Lock pages in memory** is enabled for SQL Server.

**Example:**

exec ('xp\_readerrorlog 0, 1, ''Using locked pages''')

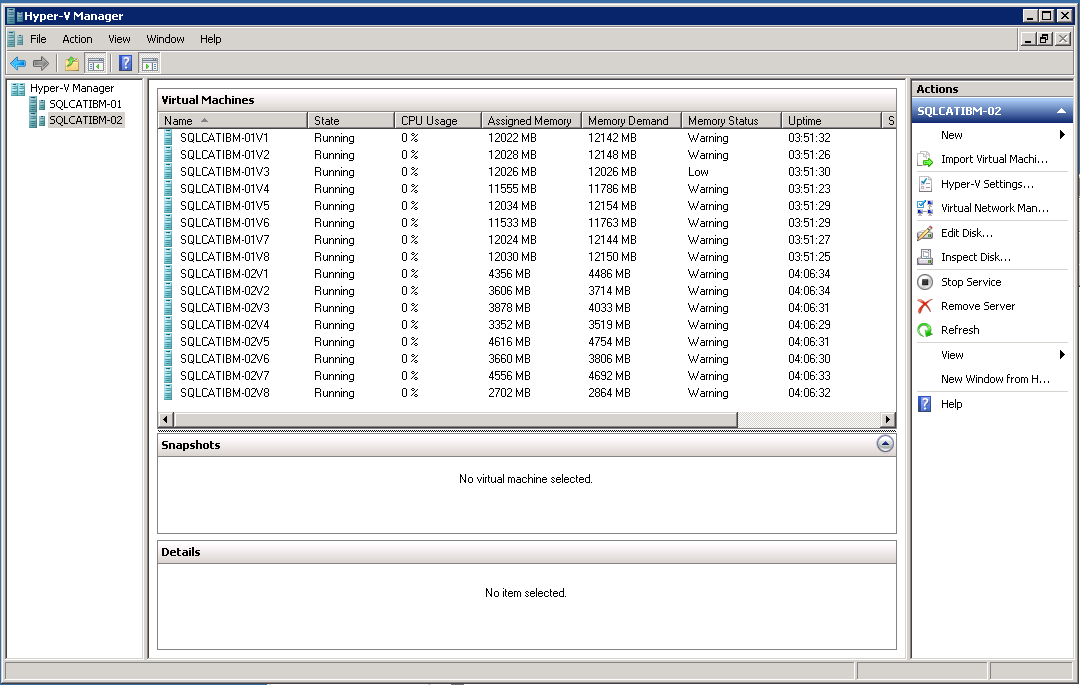
### Ring Buffer DMV

The SQL Server engine also exposes information about memory events through a Dynamic Management View (DMV) named sys.dm\_os\_ring\_buffers. This DMV is not specific to memory events; however, in the context of monitoring for memory events that influence the **Target Server Memory** SQL Server sets, there is a useful type of ring buffer named RING\_BUFFER\_RESOURCE\_MONITOR.

An in-depth discussion of this topic is beyond the scope of this paper. For more information, including the methods for gathering and interpreting this data, see the “Detecting Memory Pressures” section of [Troubleshooting Performance Problems in SQL Server 2008](http://msdn.microsoft.com/en-us/library/dd672789.aspx) (http://msdn.microsoft.com/en-us/library/dd672789.aspx).

### Hyper-V Manager Memory Pressure Reporting

When you use Hyper-V Dynamic Memory, Hyper-V Manager reports information related to dynamic memory, including Assigned Memory, Memory Demand, and Memory Status. The following image shows the information provided through the UI.



### Windows Event Logs

In addition to the System and Application event logs, Windows exposes event logs that are specific to Hyper-V. You can find them in Event Viewer under the Applications and Services Logs\Microsoft\Windows node of the tree view. The two of most interest for these scenarios are the following:

* **Hyper-V VMMS**: Contains information related to Hyper-V Dynamic Memory Manager related events.
* **Hyper-V High Availability**: Contains information about events related to failovers, including both Live and Quick migration types.

# Appendix 4 – Additional References

**Hyper-V Dynamic Memory Configuration Guidance**

<http://technet.microsoft.com/en-us/library/ff817651(WS.10).aspx>

**SQL Server Customer Advisory Team**

<http://sqlcat.com/>

**Windows Virtualization Team Blog**

<http://blogs.technet.com/b/virtualization/>

**SQL Server SQLOS Team Blog**

<http://blogs.msdn.com/b/sqlosteam/>

**Support policy for Microsoft SQL Server products that are running in a hardware virtualization environment**

<http://support.microsoft.com/kb/956893>

**Hot Add Memory Support in SQL Server**

<http://technet.microsoft.com/en-us/library/ms175490.aspx>

**How to reduce paging of buffer pool memory in the 64-bit version of SQL Server**

<http://support.microsoft.com/kb/918483>