HDS performance tests

Description: This document describes the experiences gathered with HDS and SQL Server tests.

www.pcvuesolutions.com

FRANCE - Paris
ARC Informatique
Head Office

GERMANY - Munich
PcVue GmbH

ITALY - Milan
PcVue Srl

UK - London
Control Technology International

USA - Boston
PcVue Inc

SINGAPORE - Singapore
PcVue Sea

MALAYSIA – Kuala Lumpur
PcVue Sdn Bhd

CHINA - Shangai
PcVue China

JAPAN - Nagoya
PcVue Japan

Keywords: PcVue, HDS, SQL Server

Last Revision Date:

ARC Informatique
ISO 9001 : 2008
ISO 14001 : 2004
certified

Authorization

	Name	Stamp	Date
Written by	AD		05 September 2013
Checked by	PB		02 October 2013
Authorized by			

Revision history

Revision	Author	Action	Editing	Date	Distribution

SUMMARY

1 Introduction

The aim of this document is to provide best practices when using SQL Server with PcVue in giving a simple overview of the performance expected from PcVue HDS.
Only write performances are tested for optimization given that they might reflect the overall performance in a real project. Hard disks being a key factor, single and multiple independent disks are tested.

RAID (**redundant array of independent disks**, originally **redundant array of inexpensive disks**) is a storage technology that combines multiple disk drive components into a logical unit for the purposes of data redundancy and performance improvement. Data is distributed across the drives in one of several ways, referred to as RAID levels, depending on the specific level of redundancy and performance required.

In **RAID 10**, often referred to as **RAID 1+0** (mirroring and striping), data is written in stripes across primary disks that have been mirrored to the secondary disks.

In this document you will learn that PcVue, on this specific hardware configuration, is able to:

- write about 2400 records per second on a **single HDD** in a continuous mode,
- write about 5100 records per second on **two RAID 10** with embedded cache configuration, in a continuous mode.

Other tested parameters have a low impact on the overall performance.

Finally an estimation of what could be the performance of a real project run on a single station with a basic purge policy has been done showing that PcVue is able to:

- write about 3700 records per second on **one RAID 10** with embedded cache configuration, in a continuous mode.

1.1 Test conditions

Excepted for the real project expectation, all tests were realized without external solicitation of HDD.

- HDS makes only write operation (no read and no maintenance task)

⚠ **For these reasons, you can't retain following results as limitation for a real project.**

- Set initial database size to 30GB (29.5GB for .mdf / 0.5GB for .ldf)
- Disable HDS traces
- Delete database before each test

⚠ **The expected performances for a real project are really dependent of parameters such as number of connected stations, PLC communication, operators' activity, Database purge policy and more.**

1.2 PcVue versions

The following PcVue version has been used for performing the tests:

- PcVue 11 installed on system disk

1.3 Software configuration

- Windows Server 2008 R2 Enterprise SP1 x64
- SQL Server 2008 x64 installed on system disk

1.4 Common hardware configuration

- CPU : Intel Xeon E31220 @ 3.10GHz
- RAM : 8GB

1.5 Benchmark tools

- CrystalDiskMark: used to make quick tests.

http://crystalmark.info/download/index-e.html

- SQLIO Disk Subsystem Benchmark Tool: used to make accurate tests.

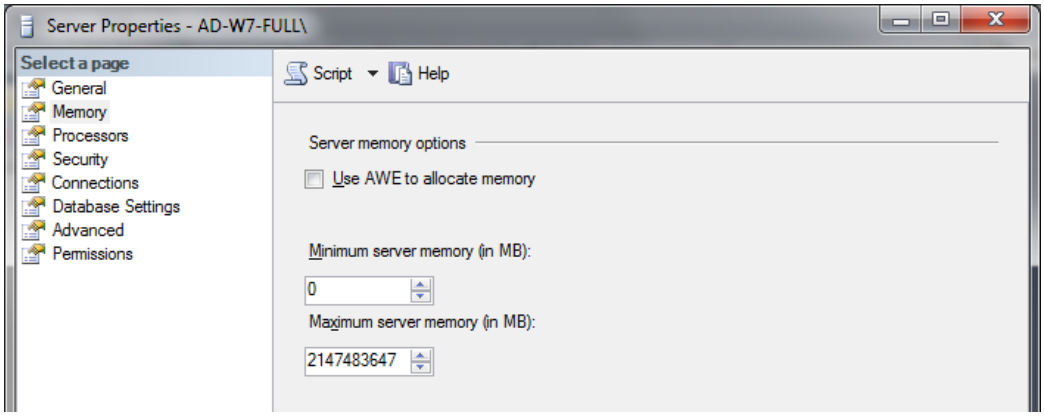
http://www.microsoft.com/en-us/download/details.aspx?id=20163

Please refer to Appendix 1: SQLIO for SQL Server to install and configure it.

1.6 Advices on SQL Server configuration

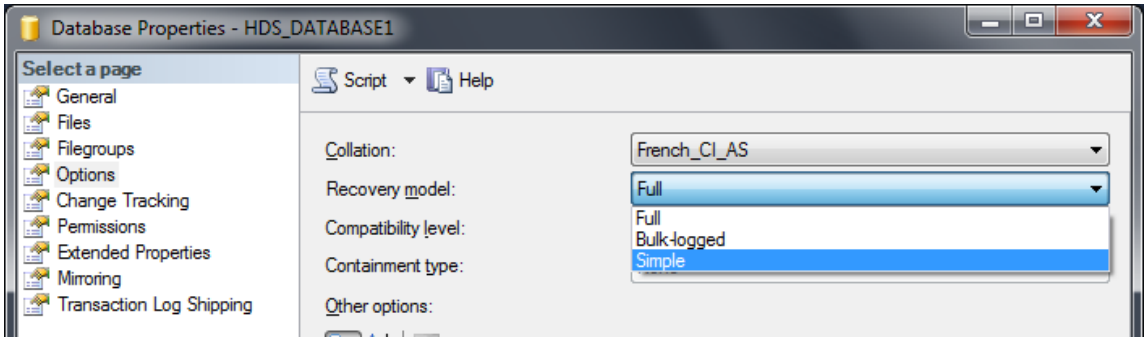
SQL Server has the property of allocating for itself all the necessary RAM available at a given time, thus creating a potential conflict with PcVue needs.

- Set maximum server memory dedicated to SQL Server. This operation is useful to configure an instance of SQL Server in conjunction with the memory requirements of other applications that run on the same computer. In PcVue case, value must be done regarding sv32.exe and HDS.exe memory used and total of memory available on your computer.



⚠ maximum RAM used by sv32.exe and HDS.exe is 4GB (2x2GB). This limitation can be exceeded if your project uses the "Large Address Aware" option.

- Take care about the database "Recovery model". SQL Server backup and restore operations occur within the context of the recovery model of the database.
  Recovery models are designed to control transaction log maintenance. A recovery model is a database property that controls how transactions are logged, whether the transaction log requires (and allows) backing up, and what kinds of restore operations are available.
  Three recovery models exist: simple, full, and bulk-logged. If you don't back up database log file, set value to simple.



⚠ default value is set by your database model.

- Initialize initial size of databases (.mdf, .ldf) to limit dynamical allocation and fragmentation. Create your backup file on a separate HDD to not disturb your database HDD.

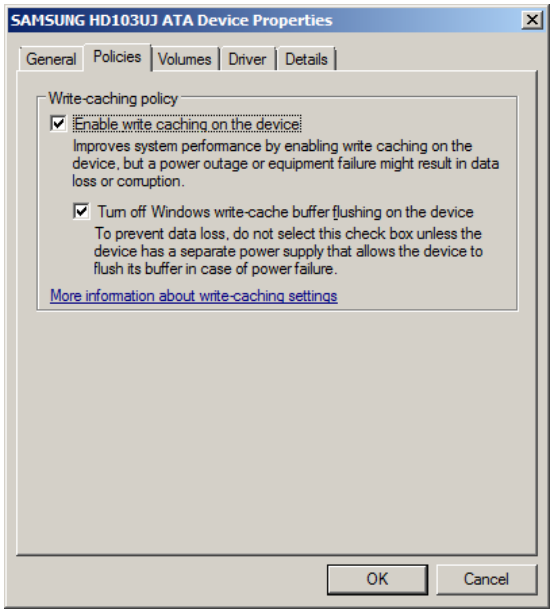
1.7 Advices on RAID configuration

This kind of architecture needs system administrator skills.

- Software RAID is not as efficient as hardware RAID solution.
- RAID card with embedded cache has better performances than RAID card without embedded cache.

- Use same Hard Drive Disk (worth disk performance decrease global performances)
- Enable embedded cache on RAID card with software utility or windows option.

⚠ If you turn off Windows write-cache buffer to increase write performances you need a separate power supply that allows the device to flush its buffer in case of power failure.



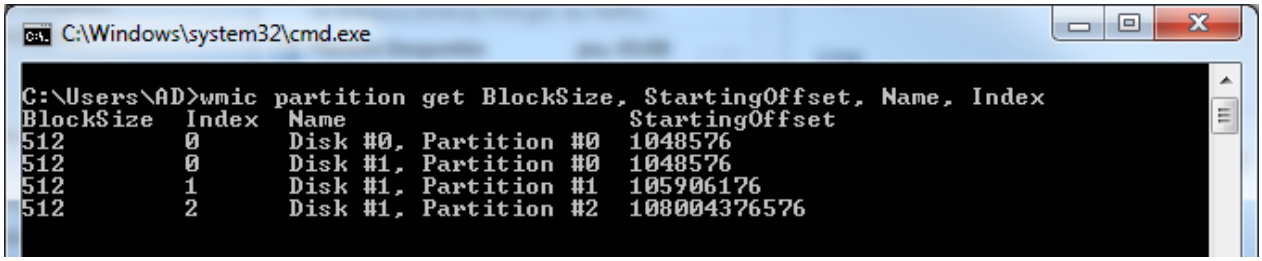
- Align your RAID partition to optimize disk latency and duration.

⚠ To get block size and starting offset, use following command line

```
> wmic partition get BlockSize, StartingOffset, Name, Index
```

⚠ To get cluster size use following command line

```
> fsutil fsinfo ntfsinfo d:
```



For more information on this subject you can take a look to MSDN:
http://msdn.microsoft.com/en-us/library/dd758814.aspx

2 Hardware performances

2.1 Make an appropriate choice for your needs

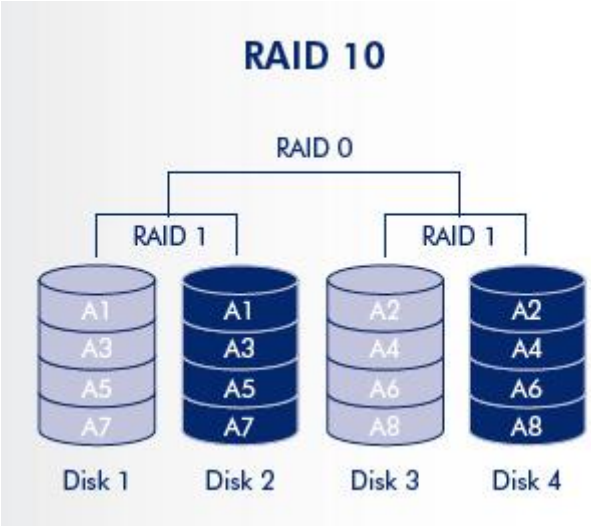
When using a dedicated server to host databases, you need to pay attention to 3 components:

- Processor
- RAM memory
- Hard drive disks

Usually, hard drive disk configuration is the bottleneck for SQL Server because it is underestimated and hard to monitor. Through several benchmarks, this chapter will help you to choose the best hardware configuration based on the expected performance for your project.

The first benchmark is done with a basic HDD configuration. In this document, results will be the reference used to compare the performances with a professional hardware.

Two others benchmarks are done to measure the gain of performance and reliability with RAID 10 configurations. Indeed, in most of the cases, RAID 10 provides better throughput and latency than all other RAID levels except RAID 0. It's the preferable RAID level for I/O intensive application such as database.



Raid10 level

In RAID 10, often referred to as RAID 1+0 (mirroring and striping), data is written in stripes across primary disks that have been mirrored to the secondary disks:

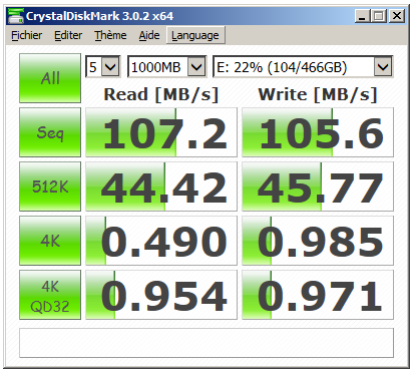
- ✔ Good availability of data: survival loss multiple disks (depending on position)
- ✔ Improved performance in reading and writing
- ⛔ Only 50% of the useful disk space is available
- ⛔ Use at least four hard drives

2.2 Determining I/O capacity with single HDD or RAID10 configuration

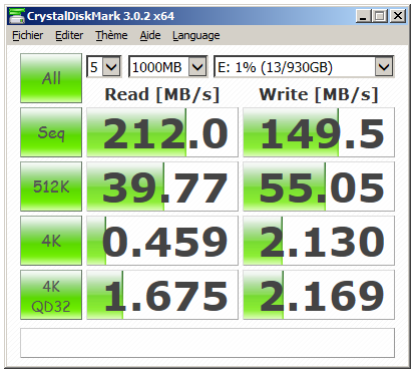
This section is necessary to determine I/O capacity of your hardware and detect any eventual dysfunction or bad configuration of it.
CrystalDiskMark tools show impact of disk configuration on bandwith.

To produce comparable results we have always used the same model of hard drive disk and separate operating system from benched devices to not disturb it by any external process.

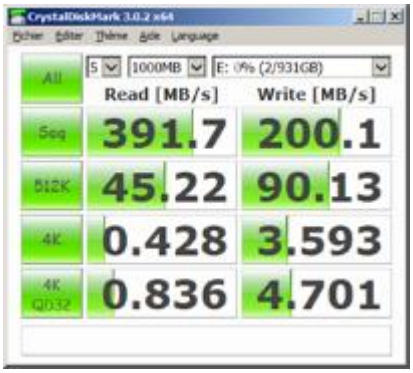
Device	Description	Used to
HDD1 **SEAGATE ST3400620AS**	400GB 7200RPM 16MB Cache	System disk, not tested OS, SQL Server, CrystalDiskMark
HDD2 HDD3 HDD4 HDD5 **TOSHIBA MK5061GS**	500GB 7200RPM 16MB Cache	Benchmark these devices (single HDD / RAID10)
RAID CARD 1 **DELL PERC H200**	SAS No Cache	Create RAID 10 array with 4 disks
RAID CARD 2 **HP Smart Array P420**	SAS Cache 1GB	Create RAID 10 array with 4 disks



Single HDD (HDD1)



RAID10 without cache



RAID10 with cache

- **Seq**, sequential read/write
- **512K**, random read/ write of 512KB block size
- **4K**, random read/ write of 4KB block size
- **4K QD32**, random write/read of 4KB block size with a queue depth of 32 Bytes

Results of these tests show that RAID10 configurations have a positive impact on read and write bandwidth. Prefer a RAID card with embedded cache memory instead of integrate RAID controller or a RAID card without embedded cache memory.

⚠ Take care of results about RAID10 card with cache because cache memory can give very good results if only cache is used. That's why we have reduced cache of RAID10 card to 100MB.

☺ **RAID10 with cache improves input-output capacity and provides data security, it's a good choice.**

2.3 Cluster size influence

To improve I/O with SQL Server, Microsoft advizes to setup the cluster size of NTFS volume to 64KB.

By default, NTFS volumes allocation unit size is set to "4KB" by the operating system but you can change it during format operation. Over 16TB, volume allocation unit size is set to "8KB" instead of "4KB"…

Although Microsoft recommends using a cluster size equal to 64 KB for SQL Server files, our tests didn't reveal any performance gain.

😐 **On Both RAID10 and NOT RAID, cluster size does not have any benefit on I/O access and transfer rate**.

See Appendix 2 for more detailed results on cluster size influence.

Determining I/O capacity of your hardware configuration is very important if you wish good performances. Be careful to constructor characteristics and make tests by yourself before validating definitive configuration.

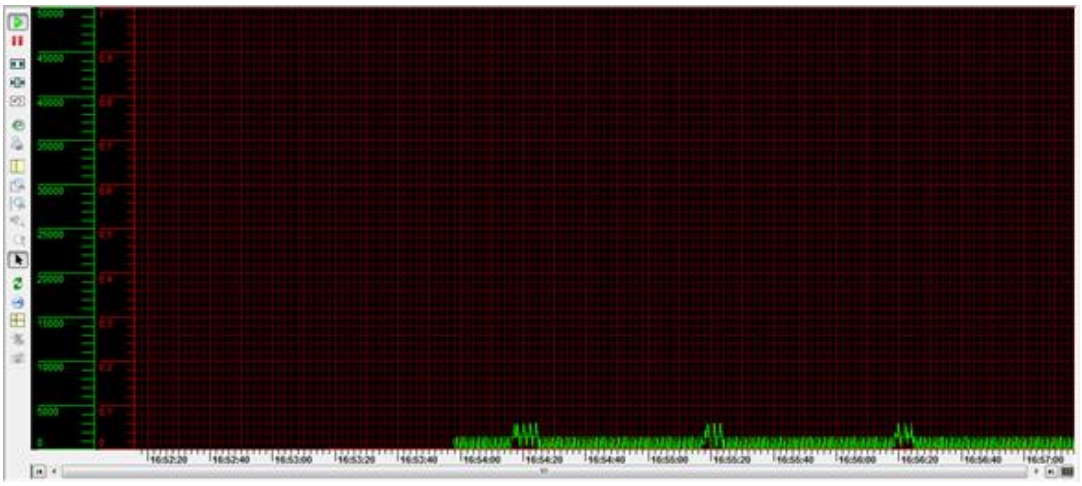
3 SQL Server optimization

Database created by HDS is defined by two files:

- Data file (.mdf)
- Log file (.ldf)

Data file is read and write randomly and log file is read and write sequentially. Consequently, an easy way to improve read and write performances of SQL Server is to split database onto separate HDD.

To found limitation of HDS write process, we used a project containing N counter variables set as trend. Each counter is incremented every second. Then we open a trend viewer to show "HDS.PendingRecords" register evolution. While value stays stable near zero and go back to 0, write process was considerated as stable.



Let's see the difference on RAID and NOT RAID architecture.

3.1 Database files on same HDD

Device	Description	Used to
HDD1 **SEAGATE ST3400620AS**	400GB 7200RPM 16MB	OS, SQL Server, PcVue
HDD2 **TOSHIBA MK5061GS**	500GB 7200RPM 16MB	Store HDS database files
SQL Server		
Data File on	HDD2	
Log File on	HDD2	

PcVue write limitation with 4KB cluster size: 2400 rec/sec
PcVue write limitation with 64KB cluster size: 2400 rec/sec

We will use these results as reference.

3.2 Database files split on two HDD

Device	Description	Used to
HDD1 **SEAGATE ST3400620AS**	400GB 7200RPM 16MB	OS, SQL Server, PcVue
HDD2 HDD3 **TOSHIBA MK5061GS**	500GB 7200RPM 16MB	Store HDS database files
SQL Server		
Data File on	HDD2	
Log File on	HDD3	

PcVue write limitation with 4KB cluster size: 2500 rec/sec
PcVue write limitation with 64KB cluster size: 2500 rec/sec

In this case, splitting database files on two disks increases write performances about 4 percent.

This gain may be biggest if you do not have the same hard drive model. With another computer and two different disks, I increased performance about 70 percent because one of the disks had a very poor IO benchmark result compared to another. In a first time, .mdf file was store on the less efficient disk and .ldf file was store on the most efficient disk. And secondly, .mdf file was store on the most efficient disk and .ldf file was store on the less efficient disk.

Always store .mdf file on your better disk.

3.3 Database files on RAID10 (without embedded cache)

Device	Description	Used to
HDD1 **SEAGATE ST3400620AS**	400GB 7200RPM 16MB	OS, SQL Server, PcVue
HDD2 HDD3 HDD4 HDD5 **TOSHIBA MK5061GS**	500GB 7200RPM 16MB	Store HDS database files
RAID CARD 1 **DELL PERC H200**	SAS No Cache	Create RAID 10 array with 4 disks
SQL Server		
Data File on	RAID10	
Log File on	RAID10	

PcVue write limitation with 4KB cluster size: 2400 rec/sec
PcVue write limitation with 64KB cluster size: 2400 rec/sec

Theoretically, RAID10 configuration should increase write performances about 100% with 4 disks. So this result is not good for a RAID card.

After a quick search on the Dell support portal, other people have had this bad experience. We have to test a better RAID card.

3.4 Database files split on HDD and RAID10 (without embedded cache)

Device	Description	Used to
HDD1 **SEAGATE ST3400620AS**	400GB 7200RPM 16MB	OS, SQL Server, PcVue
HDD2 HDD3 HDD4 HDD5 **HDD6** **TOSHIBA MK5061GS**	500GB 7200RPM 16MB	Store HDS database files
RAID CARD 2 **HP Smart Array P420**	SAS Cache 1GB	Create RAID 10 array with 4 disks
SQL Server		
Data File on	RAID10	
Log File on	HDD6	

PcVue write limitation with 4KB cluster size: 2700 rec/sec
PcVue write limitation with 64KB cluster size: 2700 rec/sec

This configuration improves performance about 12 percent.

3.5 Database files on RAID10 with embedded cache

Device	Description	Used to
HDD1 **SEAGATE ST3400620AS**	400GB 7200RPM 16MB	OS, SQL Server, PcVue
HDD2 HDD3 HDD4 HDD5 **TOSHIBA MK5061GS**	500GB 7200RPM 16MB	Store HDS database files
RAID CARD 2 **HP Smart Array P420**	SAS Cache 1GB	Create RAID 10 array with 4 disks
SQL Server		
Data File on	RAID10	
Log File on	RAID10	

PcVue write limitation with 4KB cluster size: 4700 rec/sec
PcVue write limitation with 64KB cluster size: 4700 rec/sec

Our results confirm the theory; write performances almost doubled.

This configuration improves performance about 96 percent.

3.6 Database files split on HDD and RAID10 (embedded cache)

Device	Description	Used to
HDD1 **SEAGATE ST3400620AS**	400GB 7200RPM 16MB	OS, SQL Server, PcVue
HDD2 HDD3 HDD4 HDD5 **HDD6** **TOSHIBA MK5061GS**	500GB 7200RPM 16MB	Store HDS database files
RAID CARD 2 **HP Smart Array P420**	SAS Cache 1GB	Create RAID 10 array with 4 disks
SQL Server		
Data File on	RAID10	
Log File on	HDD6	

PcVue write limitation with 4KB cluster size: 3100 rec/sec
PcVue write limitation with 64KB cluster size: 3100 rec/sec

In this configuration limitation is due to log file store on HDD6.
Two ways:

- HDD6 is too slow for log file write regarding data file on RAID 10
- Motherboard controller is not enough powerful

To define limitation origin, we will replace HDD6 (7200RPM) by a most powerful HDD (15000RPM).

Device	Description	Used to
HDD1 **SEAGATE ST3400620AS**	400GB 7200RPM 16MB	OS, SQL Server, PcVue
HDD2 HDD3 HDD4 HDD5 **TOSHIBA MK5061GS**	500GB 7200RPM 16MB	Store HDS database files
HDD7 **SEGATE ST3300657SS**	300GB 15000RPM 16MB	Store HDS database files
RAID CARD 2 **HP Smart Array P420**	SAS Cache 1GB	Create RAID 10 array with 4 disks
SQL Server		
Data File on	RAID10	
Log File on	HDD7	

PcVue write limitation with 4KB cluster size: 3100 rec/sec
PcVue write limitation with 64KB cluster size: 3100 rec/sec

Using a fastest HDD didn't change anything. Splitting log file on a HDD not managed by RAID10 is not a good option.

3.7 Database files on RAID10 (embedded cache) - 15RPM HDD

Device	Description	Used to
HDD1 **SEAGATE ST3400620AS**	400GB 7200RPM 16MB	OS, SQL Server, PcVue
HDD7 HDD8 HDD9 HDD10 **SEGATE ST3300657SS**	300GB 15000RPM 16MB	Store HDS database files
RAID CARD 2 **HP Smart Array P420**	SAS Cache 1GB	Create RAID 10 array with 4 disks
SQL Server		
Data File on	RAID10	
Log File on	RAID10	

PcVue write limitation with 4KB cluster size: 4800 rec/sec
PcVue write limitation with 64KB cluster size: 4800rec/sec

This configuration improves performance about 100 percent.

3.8 Database files on two RAID10 with embedded cache - 7/15RPM HDD

Device	Description	Used to
HDD1 **SEAGATE ST3400620AS**	400GB 7200RPM 16MB	OS, SQL Server, PcVue
HDD2 HDD3 HDD4 HDD5 **TOSHIBA MK5061GS**	500GB 7200RPM 16MB	Store HDS database files
HDD7 HDD8 HDD9 HDD10 **SEGATE ST3300657SS**	300GB 15000RPM 16MB	Store HDS database files
RAID CARD 1 **DELL PERC H200**	SAS No Cache	Create RAID 10 array with 4 disks (HDD2 to HDD5)
RAID CARD 2 **HP Smart Array P420**	SAS Cache 1GB	Create RAID 10 array with 4 disks (HDD7 to HDD10)
SQL Server		
Data File on	RAID10	Raid card 2 with 15K HDD
Log File on	RAID10	Raid card 1 with 7K HDD

PcVue write limitation with 4KB cluster size: 5100 rec/sec
PcVue write limitation with 64KB cluster size: 5100 rec/sec

This configuration improves performance about 112 percent. It's our best configuration.

3.9 To summarize

SQL Server optimization is really important to improve write performances and RAID10 have a real impact on it in addition to being more reliable for data.

Be aware, RAID architecture is not enough to secure your data. You must backup your database regularly.

	7200 RPM / 16MB						15000 RPM / 16MB				Results	
	HDD1	HDD2	HDD3	HDD4	HDD5	HDD6	HDD7	HDD8	HDD9	HDD10	Gain	Reliable
Cfg. 1	OS PCVUE SQL	.LDF .MDF									Ref.	⛔
Cfg. 2	OS PCVUE SQL	.MDF	.MDF								★ +4%	⛔
Cfg. 3	OS PCVUE SQL	RAID10, no cache memory .LDF .MDF									+0%	✔
Cfg. 4	OS PCVUE SQL	RAID10, no cache memory .MDF				.LDF					★ +12%	⛔
Cfg. 5	OS PCVUE SQL	RAID10, 1GB cache memory .LDF .MDF									★★★ +96%	✔
Cfg. 6	OS PCVUE SQL	RAID10, 1GB cache memory .MDF				.LDF					★★ +29%	⛔
Cfg. 7	OS PCVUE SQL	RAID10, 1GB cache memory .MDF					.LDF				★★ +29%	⛔
Cfg. 8	OS PCVUE SQL						RAID10, 1GB cache memory .LDF .MDF				★★★ +100%	✔
Cfg. 9	OS PCVUE SQL	RAID10, 1GB cache memory .LDF					RAID10, 1GB cache memory .MDF				★★★ +112%	✔

Good configurations of SQL Server can double your performances.

4 PcVue optimization

You can adjust some parameters in HDS configuration file to improve performances. These parameters don't have a real impact on bandwidth but they can help you to reduce pending records variations and improve global stability.

4.1 HDSConf.ini

This file is global to all PcVue projects.

[General] section		
Parameter name	Description	Default value
DefaultMaxBufferedEvents	This parameter specifies the number of incoming events buffered for treatment by the HDS. When HDS has more events it stops accepting new events from external source while its buffer list is saturated.	20 000

To configure it, you have to increase this value to 200,000 and record "HDS.PendingRecords" variable to follow it in real time.

For example: if in your project, "HDS.PendingRecords" value remains under 100,000 you can readjust "DefaultMaxBufferedEvent" parameter to 120,000 to keep a gap for security.

4.2 HDSConf.dat

This configuration file is related to a specific project.

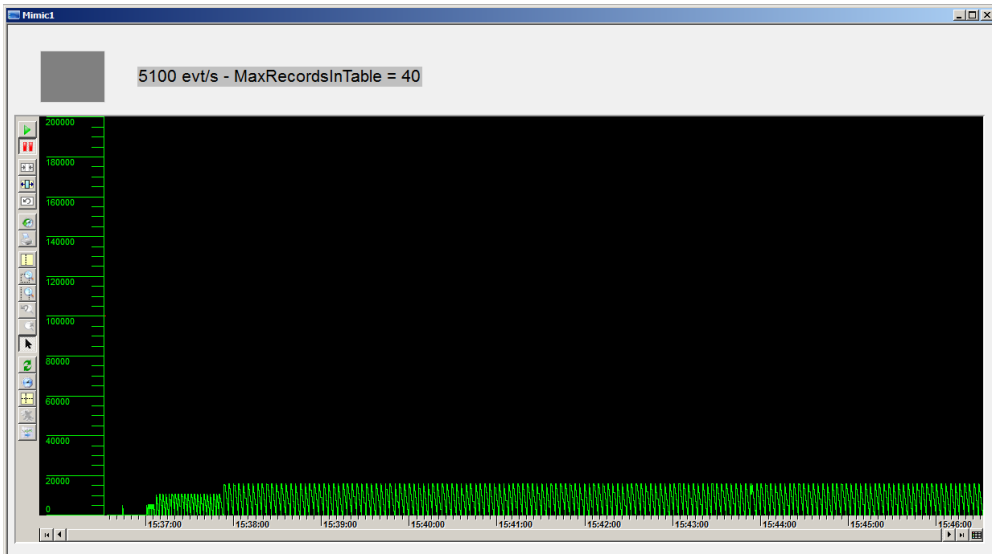
[Performance] section		
Parameter name	Description	Default value
TimeSortingPeriod	Interval of time the data will be stored in the buffer before being written in the database (should help to reduce the database fragmentation)	0
MaxBookmarkInTable	The writing buffer can contain `MaxBookmarkInTable` rows. When the buffer is full, the buffer is triggered to be written in the database.	40

The most important parameter to modify is "MaxBookmarkInTable". It can be interpreted as the maximum of records in HDS table before flushing/insert them onto SQL Server Database.

4.3 MaxBookmarkInTable

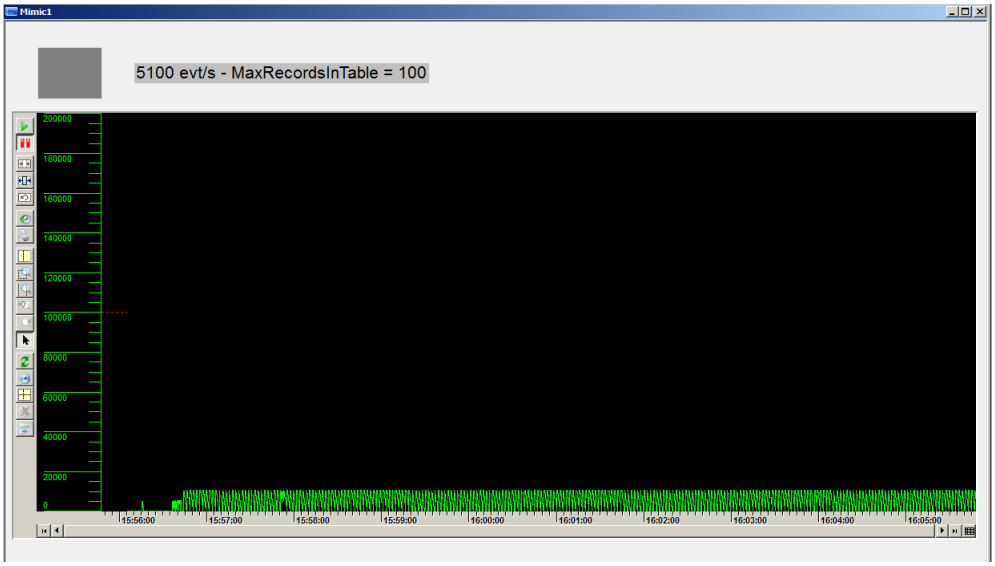
4.3.1 Set to 40 (default value)

As you can see on following picture, "HDS.PendingRecords" values oscillate between 0 and 20,000. This configuration with 5100 events written per second is stable.



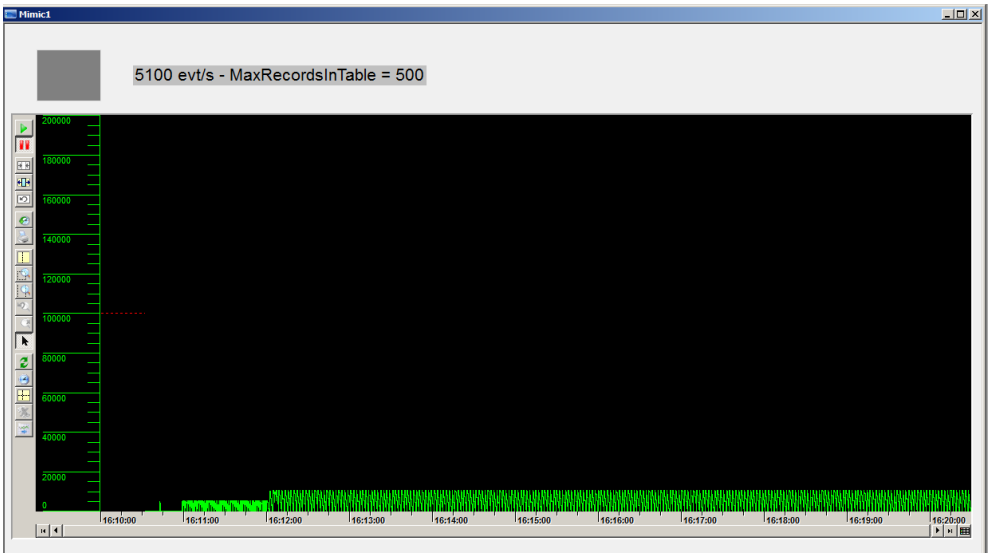
4.3.2 Set to 100

This configuration with 5100 events written per second is always stable but didn't permit us to increase events per second to 5200. "HDS.PendingRecords" values oscillate between 0 and 10,000. It's better than default value of MaxBookmarkInTable.



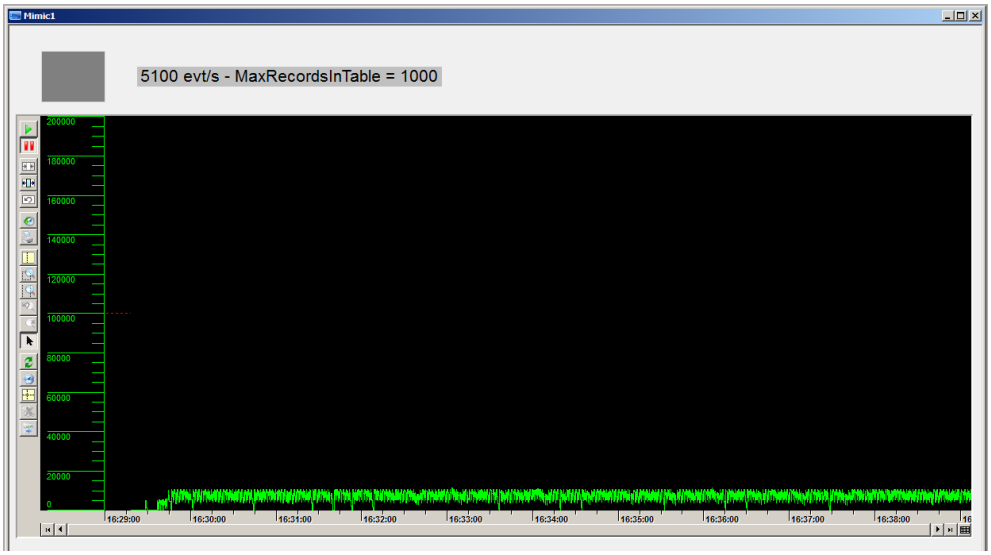
4.3.3 Set to 500

With MaxBookmarkInTable set to 500, "HDS.PendingRecords" values oscillate between 0 and 10,000 and didn't permit us to increase events per second to 5,200. There is no improvement compared to 100 records.



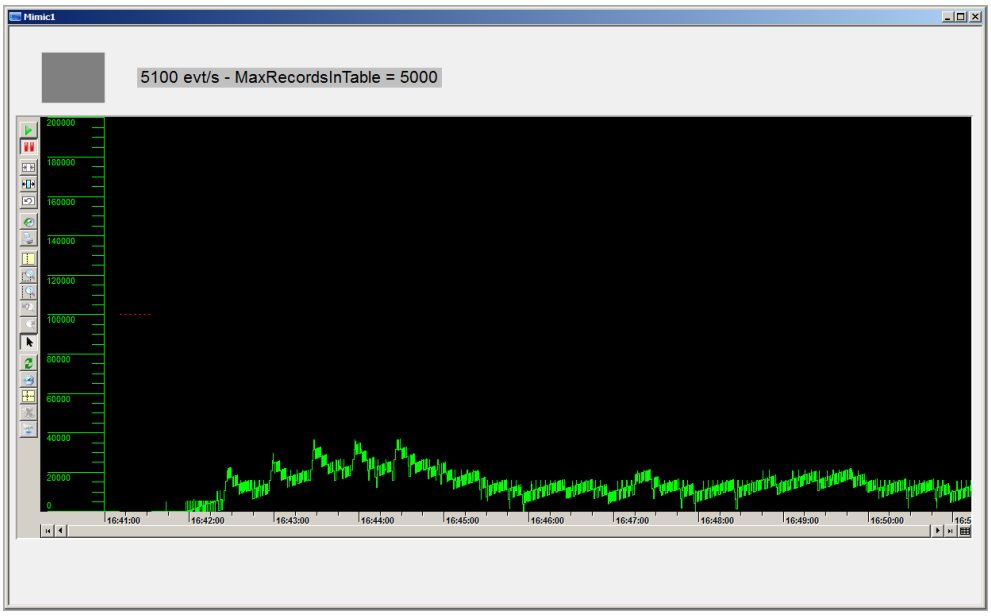
4.3.4 Set to 1000

With MaxBookmarkInTable set to 1000, "HDS.PendingRecords" values do not often falls back to zero. It becomes unstable.



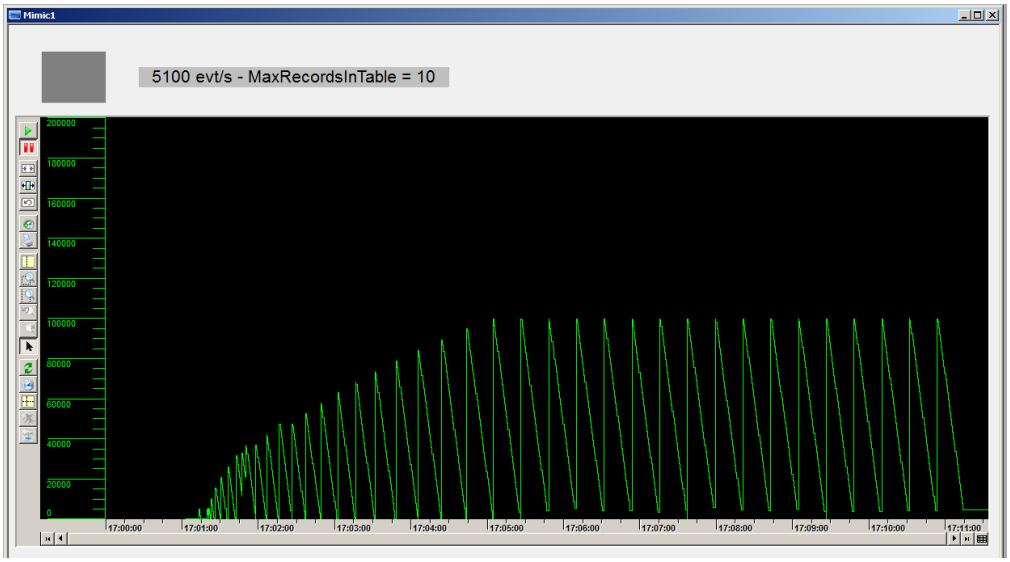
4.3.5 Set to 5000

With MaxBookmarkInTable set to 5000, "HDS.PendingRecords" values oscillates too much to be considered as stable.



4.3.1 Set to 10

With MaxBookmarkInTable set to 10, "HDS.PendingRecords" values oscillates too much to be considering as stable.



4.4 To summarize

PcVue optimisation can improve stability of write process but doesn't increase performances.

For our test project, best configuration range is between 100 and 500.

5 What to expect in a real project?

To realise our tests on a real project, we will reuse the previous configuration who gave us good performance results: Database files on RAID10 (embedded cache) - 15RPM HDD

5.1 PcVue versions

The following PcVue version has been used for performing the tests:

- PcVue 11 installed on system disk

5.2 Software configuration

- Windows Server 2008 R2 Enterprise SP1 x64
- SQL Server 2008 x64 installed on system disk

5.3 Hardware configuration

Device	Description	Used to
CPU	Intel Xeon E31220 3.10GHz	
RAM	8GB	
HDD1 **SEAGATE ST3400620AS**	400GB 7200RPM 16MB	OS, SQL Server, PcVue
HDD7 HDD8 HDD9 HDD10 **SEGATE ST3300657SS**	300GB 15000RPM 16MB	Store HDS database files
RAID CARD 2 **HP Smart Array P420**	SAS Cache 1GB	Create RAID 10 array with 4 disks
SQL Server		
Data File on	RAID10	
Log File on	RAID10	

5.4 Project configuration

Our test project run on a single computer, it makes data acquisition from a PLC and records some of them in SQL Server database. This project contains 5000 variables that change value randomly depending on what returns the PLC.
The number of write per seconds has been adjusted by adding more or less trend variables in the archive unit configuration.

To simulate a random read process we have add a SCADA basic program opening sequentially 10 different mimics each 30 seconds. Each mimic contains a "trend viewer" object displaying one hour of eight archived variables.

To avoid time distortion of the SQL Server engine, we used realistic variables name and not var01, var02, var03...

To simulate real data life in the database, we configured a maintenance plan with common tasks: purge and shrink. This maintenance plan is executed each hour and purge task is configured to keep 1 hour of data.

5.5 Project optimization

To optimize our test project, we have to increase MaxBufferedEvents parameter to 200 000 instead 20 000 and set MaxBookmarkInTable parameter to 100 instead 40.

To increase the performance of the purge at the limit of the maximum of pending records, it's possible to change another parameter configured in the HDS.ini file.

[General] section		
Parameter name	Description	Default value
DefaultMaxPurgedRecords	Specifies maximum records to delete in one time during a purge maintenance task.	30 000

5.6 Recording 3700 evt/sec

5.6.1 Purge block size of 30 000 records

Project optimisation	Value
MaxBufferedEvents	200 000
MaxBookmarkInTable	100
DefaultMaxPurgedRecords	30 000

Monitored values	Result description
Purge duration	50 minutes to purge one hour of data
Shrink duration	1 minutes
Max pending records	47 000
Average pending records	4 000
Max database size	6 491MB

This project is considered as stable. PcVue is able to delete one hour of data in less than one hour.

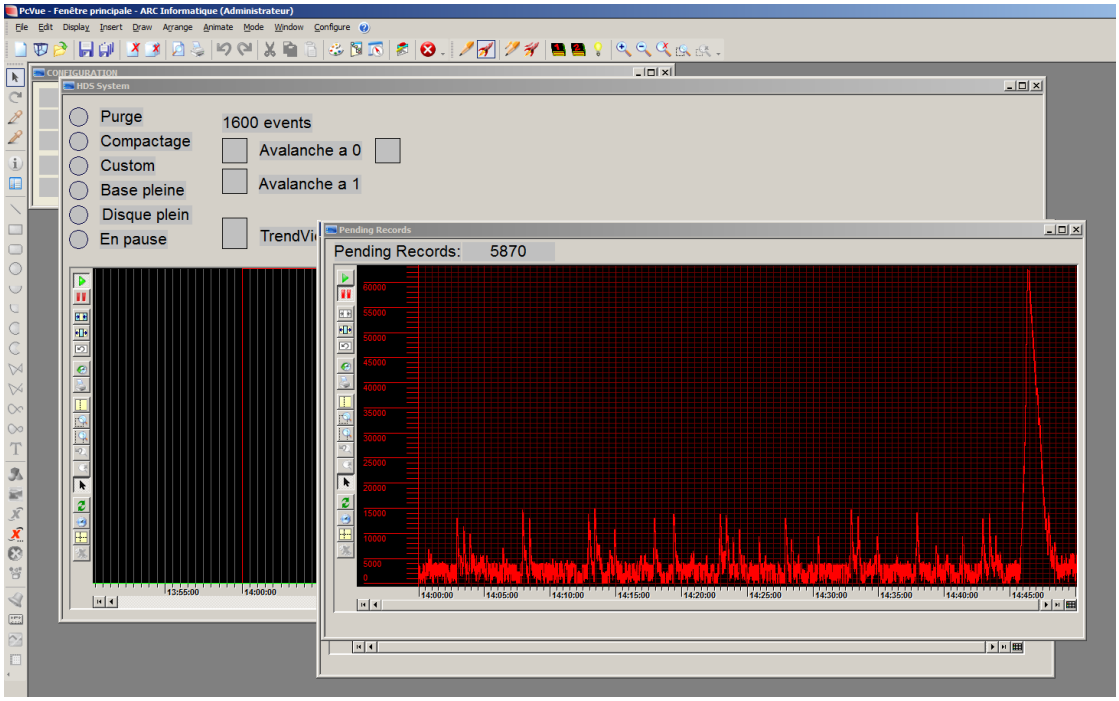
Another interesting value is the max database size. With 3700 records per second it takes 6.5GB to store them. So in 1 day you fill 156GB and in 1 week more than 1TB.

5.6.2 Purge block size of 60 000 records

Project optimisation	Value
MaxBufferedEvents	200 000
MaxBookmarkInTable	100
DefaultMaxPurgedRecords	60 000

Monitored values	Result description
Purge duration	46 minutes to purge one hour of data
Shrink duration	1 minutes
Max pending records	68 000
Average pending records	5 000
Max database size	6 491MB

This project is considered as stable. PcVue is able to delete one hour of data in less than one hour.



Pending records max value is reach during shrink tasks. So in our case, this task didn't have real interest for our configuration because we only have one database on HDD volume.

5.7 To summarize

In the previous chapter, this hardware configuration has given us a limitation of 4800rec/sec in our write only test.

With this single station “real project like” the 3700rec/sec write rate is considered as stable with a basic maintenance task without replication, backup and index rebuild.

Do not use shrink database if your project doesn’t need it.

6 Appendix 1: SQLIO for SQL Server

6.1 Prerequisite

The aim of this chapter is to help system administrator to set up benchmark SQLIO tool. SQLIO is a free software shared by Microsoft which can be used to determine the I/O capacity of a given configuration.

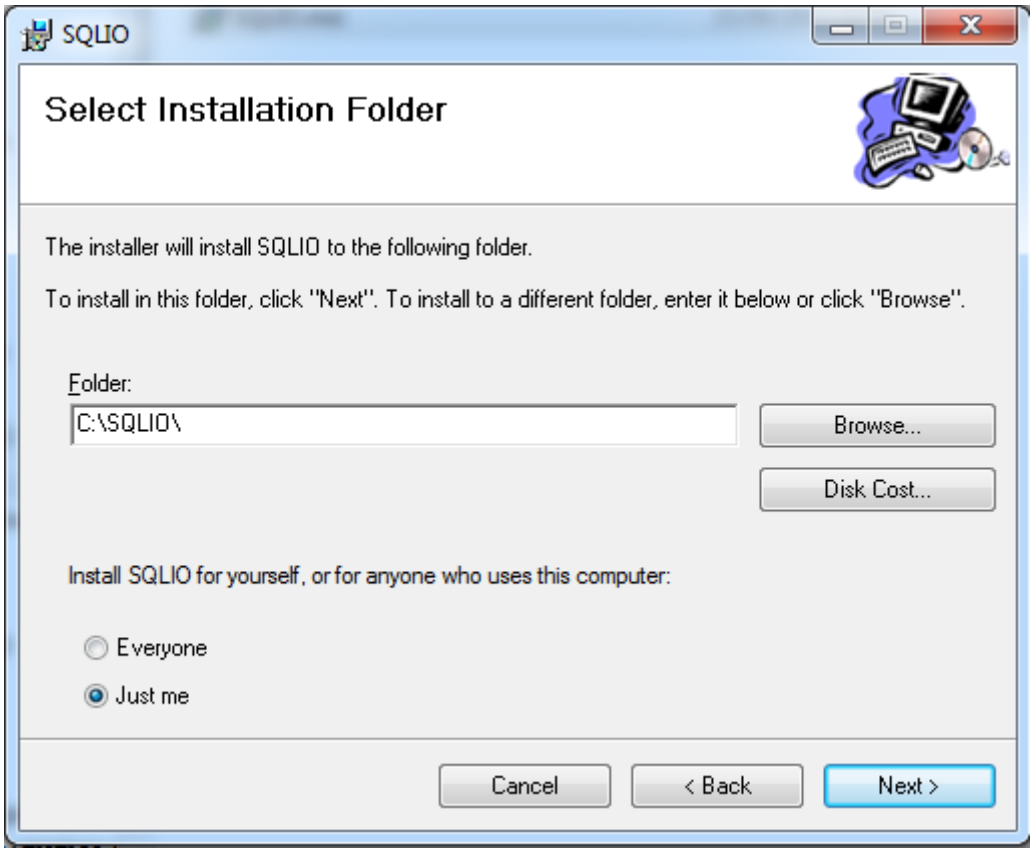
6.1.1 Download SQLIO

Official Microsoft website:
http://www.microsoft.com/en-us/download/details.aspx?id=20163

6.1.2 Install SQLIO

Execute downloaded file (.msi) and select a destination directory.
➔ C:\Program Files\SQLIO\
or
➔ C:\Program Files (x86)\SQLIO\
Or
➔ C:\SQLIO\

See following image:



6.1.3 What you have to know

- SQLIO has nothing to do with SQL Server except the name.
- In this document we will try to mimic SQL Server I/O patterns.
- Performance benchmark test results can be very helpful in taking decisions, regarding the deployment of a database.

6.2 Configuration

6.2.1 Customize file param.txt

This file is the one which serves as an option to specify where to conduce test.
By default it contains file to test, the number of threads to perform read/write operations, an affinity mask and the size of the test file in MB.

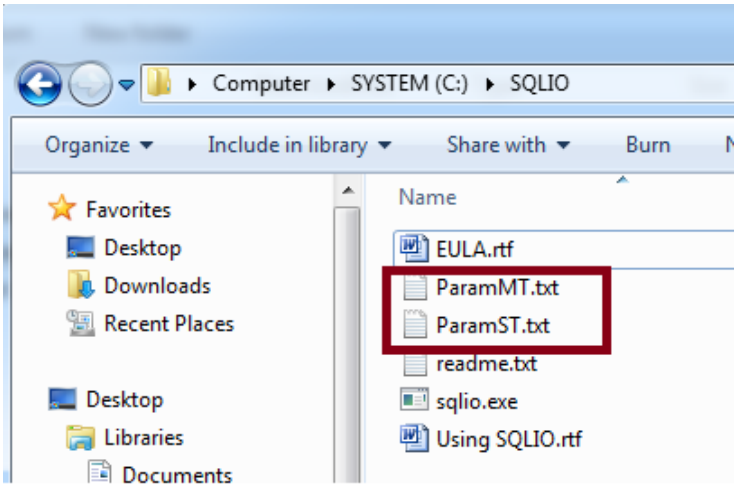
In SQLIO install directory, open "Param.txt" file.

Param.txt content

```
c:\testfile.dat 2 0x0 100
#d:\testfile.dat 2 0x0 100
```

c:\testfile.dat: tests file location
2: number of threads
0x0: affinity mask
100: test file size (MB)
#: comment line

For our test we will create two "param.txt" named "ParamST.txt" and "ParamMT.txt".
See following image:



Read test will be done with ParamMT that uses all of available CPU and write test will be done with ParamMT and ParamST that uses one thread (similar to checkpoint).

. ParamST.txt content

```
e:\testfile.dat 1 0x0 102400
```

Instead "e:\" selects logical drive you need to test.
Instead "102400" if you didn't have 100Go available on your HDD you can use a smaller value (unity of this value is MB).

. ParamMT.txt content

```
e:\testfile.dat 8 0x0 102400
```

Instead value "8", put the number of logical CPU used by your environment.
Instead "e:\" selects logical drive you need to test.
Instead "102400" if you didn't have 100Go available on your HDD you can use a smaller value. Use the same value as you use previously in ParamST.txt.

6.2.2 Create a batch file

Following batch file tries to simulate hard drive disk access as SQL Server engine do when you use HDS or make regular SQL operation as INSERT, UPADTE, SELECT…

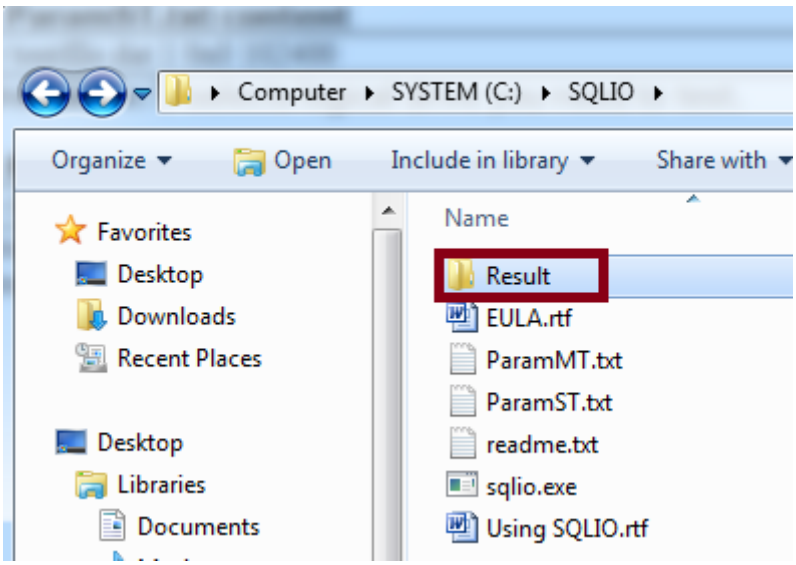
This batch simulates the following SQL Server behavior.

Read:
- Single page reads (8 KB) in SQL with 1 or 8 outstanding request / thread
- Extent reads (64 KB) in SQL with 1 or 8 outstanding request / thread
- Read ahead (512 KB) in SQL with 1 or 8 outstanding request / thread

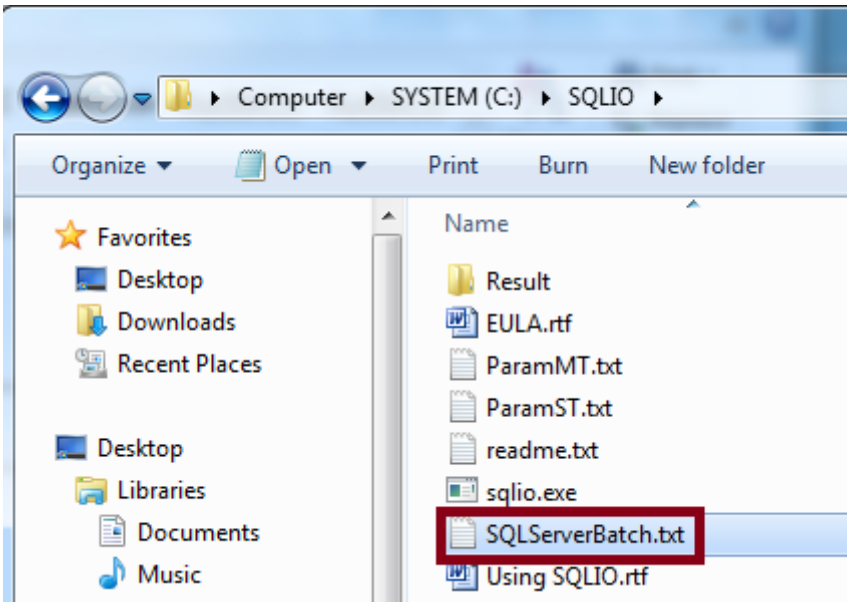
Write:
- Single page writes (8 KB) in SQL with 1 or 8 outstanding request / thread
- Extent writes (64 KB) in SQL with 1 or 8 outstanding request / thread
- Checkpoint write (256 KB) in SQL with 1 or 8 outstanding request / thread

Follow these five steps to create your own batch file:

1. Create "Result" directory at the same level of SQLIO.exe. See following image:



2. Create a text file SQLServerBatch.txt at the same level of SQLIO.exe. See following image:

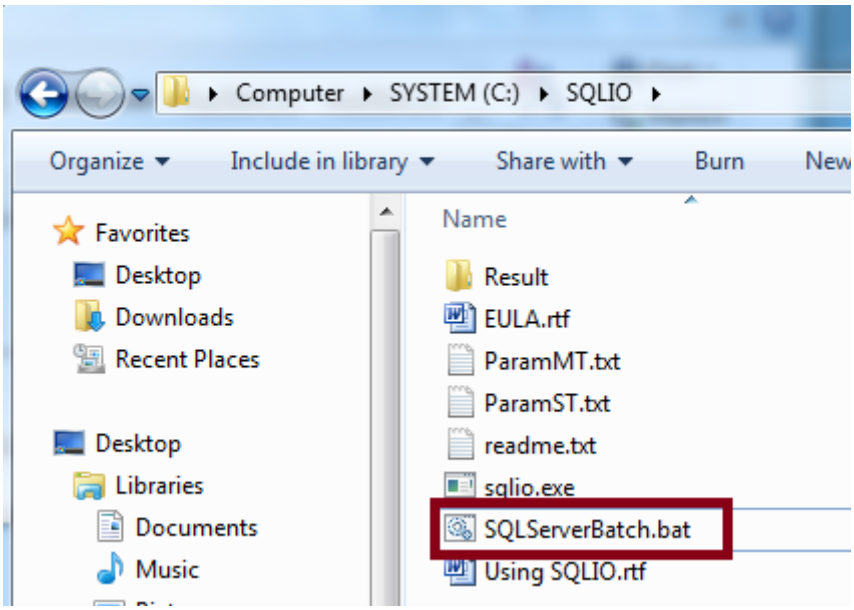


3. Edit SQLServerBatch.txt with note pad as following.

```
@echo off
echo ****** Start read tests *****

echo - Similar to single page reads (8 KB) in SQL -
sqlio -kR -s300 -frandom -o1 -b8 -LS -FparamMT.txt > .\Result\ReadsRandom8K1Oustanding.txt
timeout /T 10
sqlio -kR -s300 -frandom -o8 -b8 -LS -FparamMT.txt > .\Result\ReadsRandom8K8Oustanding.txt
timeout /T 10

echo - Similar to extent reads (64 KB) in SQL -
sqlio -kR -s300 -frandom -o1 -b64 -LS -FparamMT.txt > .\Result\ReadsRandom64K1Oustanding.txt
timeout /T 10
sqlio -kR -s300 -frandom -o8 -b64 -LS -FparamMT.txt > .\Result\ReadsRandom64K8Oustanding.txt
timeout /T 10

echo - Similar to reads Ahead (512 KB) in SQL -
sqlio -kR -s300 -frandom -o1 -b512 -LS -FparamMT.txt > .\Result\ReadsRandom512K1Oustanding.txt
timeout /T 10
sqlio -kR -s300 -frandom -o8 -b512 -LS -FparamMT.txt > .\Result\ReadsRandom512K8Oustanding.txt
timeout /T 10

echo ****** Start write tests *****

echo - Similar to single page writes (8KB) in SQL -
sqlio -kW -s300 -frandom -o1 -b8 -LS -FparamMT.txt > .\Result\WritesRandom8K1Outstanding.txt
timeout /T 10
sqlio -kW -s300 -frandom -o8 -b8 -LS -FparamMT.txt > .\Result\WritesRandom8K8Outstanding.txt
timeout /T 10
```

```
echo - Similar to extent writes (64 KB) in SQL -
sqlio -kW -s300 -frandom -o1 -b64 -LS -FparamST.txt > .\Result\WritesRandom64K1Outstanding.txt
timeout /T 10
sqlio -kW -s300 -frandom -o8 -b64 -LS -FparamST.txt > .\Result\WritesRandom64K8Outstanding.txt
timeout /T 10

echo - Similar to checkpoint write (256 KB) in SQL -
sqlio -kW -s300 -frandom -o1 -b256 -LS -FparamST.txt > .\Result\WritesRandom256K1Outstanding.txt
timeout /T 10
sqlio -kW -s300 -frandom -o8 -b256 -LS -FparamST.txt > .\Result\WritesRandom256K8Outstanding.txt
```

4. Change extension of "SQLServerBatch.txt" as ".bat". See following image:



5. Execute batch file (it will take few times to create test file define in ParamST and ParamMT).

> ⚠ **CAUTION:** do not use your computer during benchmark

Follow these four steps to test another hard drive disk:

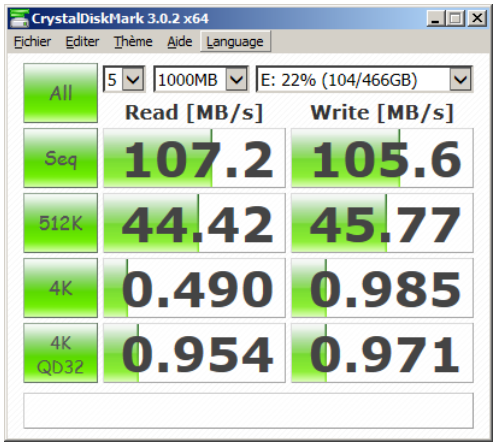
1. Save result directory
2. Clear result directory
3. Modify test file location parameter in ParamST.txt and ParamMT.txt files
4. Run batch "SQLServerBatch.bat" once again

7 Appendix 2: Cluster size detailed tests

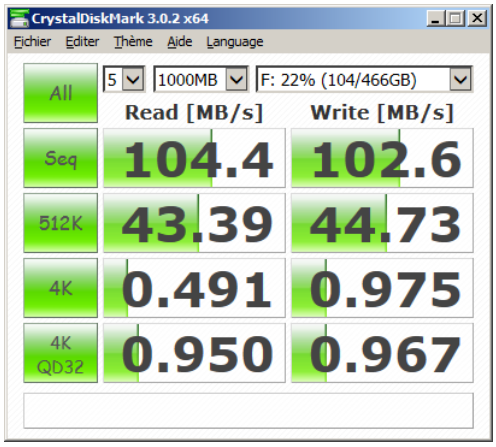
7.1.1 On single HDD

Device	Description	Used to
HDD1 **SEAGATE ST3400620AS**	400GB 7200RPM 16MB	OS, SQL Server, CrystalDiskMark, SQLIO
HDD2 **TOSHIBA MK5061GS**	500GB 7200RPM 16MB	Benchmark this device

CrystalDiskMark tools provide quick results to measure impact of cluster size on transfer rate.



4KB cluster size



64KB cluster size

In this first set of results we have similar transfer rate with the same Hard Drive Disk formatted with a cluster size of 4KB or 64KB.
To complete these tests with more accurate results we used SQLIO Disk Subsystem Benchmark Tool and SQL Server batch.

The file size used to make tests is set to 100GB for representing a realistic database size.
In the following results, the most important values are transfer rate "MB/Sec (4K / 64K)".

Random read operations

Input parameters			IO		Latency (ms)		
Block size (KB)	Thread	Outstanding	IO/Sec (4K / 64K)	MB/Sec (4K / 64K)	Min (4K / 64K)	Avg (4K / 64K)	Max (4K / 64K)
8	8	1	114,72 / 114,7	0,89 / 0,89	3 / 3	69 / 69	534 / 632
8	8	8	136,15 / 136,09	1,06 / 1,06	154 / 172	469 / 469	2687 / 2406
64	8	1	109,59 / 109,66	6,84 / 6,85	0 / 0	72 / 72	669 / 590
64	8	8	126,98 / 126,91	7,93 / 7,93	229 / 229	503 / 503	2572 / 2734
512	8	1	73,76 / 72,98	36,88 / 36,49	2 / 2	107 / 109	1088 / 1278
512	8	8	83,68 / 82,76	41,84 / 41,38	374 / 380	763 / 771	2948 / 2939

Random write operations

Input parameters			IO		Latency (ms)		
Block size (KB)	Thread	Outstanding	IO/Sec (4K / 64K)	MB/Sec (4K / 64K)	Min (4K / 64K)	Avg (4K / 64K)	Max (4K / 64K)
8	8	1	87,24 / 87,19	0,68 / 0,68	56 / 52	91 / 91	460 / 464
8	8	8	87,41 / 87,14	0,68 / 0,68	25 / 27	731 / 733	842 / 811
64	1	1	82 / 81,66	5,12 / 5,1	3 / 3	11 / 11	34 / 83
64	1	8	83,22 / 82,44	5,2 / 5,15	28 / 26	95 / 96	126 / 132
256	1	1	70,54 / 69,72	17,63 / 17,43	4 / 4	13 / 13	36 / 48
256	1	8	71,67 / 70,86	17,91 / 17,71	32 / 34	111 / 112	145 / 150

- **Block size (KB)**, the size of the IO request.
- **Thread**, number of threads to run simultaneously.
- **Outstanding**, the number of outstanding I/O requests per thread to increasing the queue depth and found disk saturation.
- **IO/Sec (4K/64K)**, number of I/O's per second with the same Hard Drive Disk formatted with a cluster size of 4KB or 64KB.
- **MB/Sec (4K/64K)**, the transfer rate with the same Hard Drive Disk formatted with a cluster size of 4KB or 64KB.
- **Latency Min/Avg/Max (4K/64K)**, the average disk latency with the same Hard Drive Disk formatted with a cluster size of 4KB or 64KB.

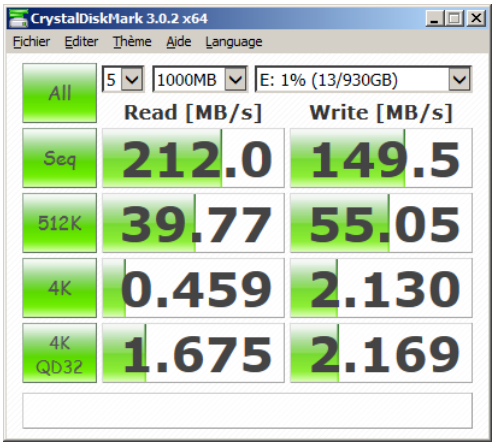
In this second set of results we can observe similar transfer rate with the same Hard Drive Disk formatted with a cluster size of 4KB or 64KB.

On a single HDD, cluster size does not have any benefit on I/O access and transfer rate.

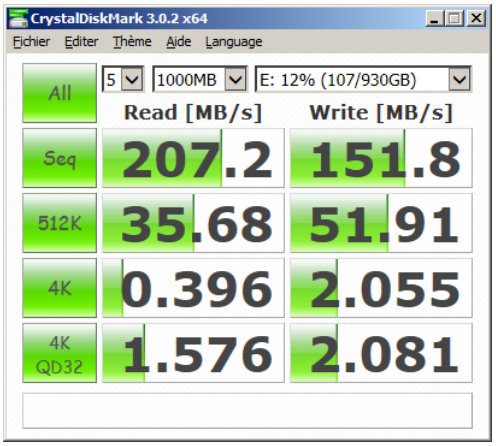
7.1.2 On RAID10 (without embedded cache) architecture

Device	Description	Used to
HDD1 **SEAGATE ST3400620AS**	400GB 7200RPM 16MB	OS, SQL Server, CrystalDiskMark, SQLIO
HDD2 HDD3 HDD4 HDD5 **TOSHIBA MK5061GS**	500GB 7200RPM 16MB	Benchmark these devices (RAID10)
RAID CARD 1 **DELL PERC H200**	SAS No Cache	Create RAID 10 array with 4 disks

In following set of results we can observe similar transfer rate with the same RAID10 volume formatted with a cluster size of 4KB or 64KB.
To complete these results with more accurate results we used SQLIO Disk Subsystem Benchmark Tool and SQL Server batch.



4KB cluster size



64KB cluster size

The file size used to make these tests is set to 100GB to represent a realistic database size.

Random read operations

Parameters			IO		Latency (ms)		
Block size (KB)	Thread	Outstanding	IO/Sec (4K / 64K)	MB/Sec (4K / 64K)	Min (4K / 64K)	Avg (4K / 64K)	Max (4K / 64K)
8	8	1	212,12/212,8	1,65/1,66	0/0	37/37	340/436
8	8	8	287,38/287,29	2,24/2,24	0/0	222/222	2488/2390
64	8	1	204,67/204,41	12,79/12,77	0/0	38/38	317/530
64	8	8	265,17/264,13	16,57/16,5	0/0	240/241	2391/2489
512	8	1	96,88/96,63	48,44/48,31	2/1	82/82	491/576
512	8	8	101,10/101,35	50,55/50,67	32/32	631/630	1380/1406

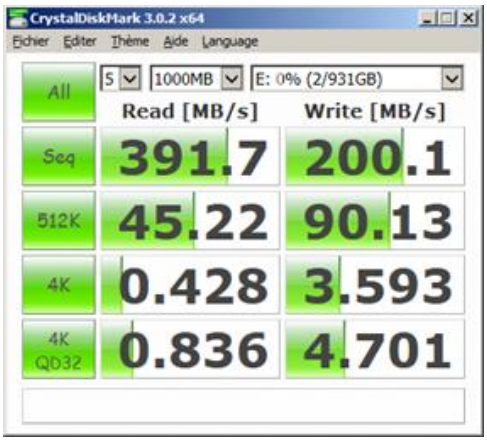
Random write operations

Parameters			IO		Latency (ms)		
Block size (KB)	Thread	Outstanding	IO/Sec (4K / 64K)	MB/Sec (4K / 64K)	Min (4K / 64K)	Avg (4K / 64K)	Max (4K / 64K)
8	8	1	306,21/304,41	2,39/2,37	0/0	25/25	931/1067
8	8	8	310,58/309,26	2,42/2,41	0/0	205/206	1286/1377
64	1	1	228,16/211,08	14,26/13,19	0/0	3/4	527/539
64	1	8	230,42/216,07	14,40/13,5	0/0	34/36	559/599
256	1	1	101,31/89,07	25,32/22,26	0/0	9/10	299/377
256	1	8	102,15/89,17	25,53/22,29	0/1	77/89	368/457

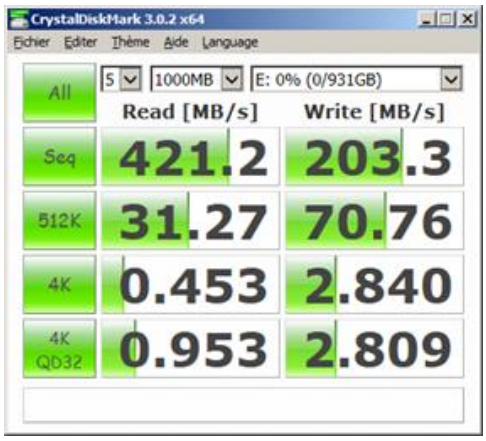
On RAID10 (without cache), cluster size does not have any benefit on I/O access and transfer rate.

7.1.3 On RAID10 (with embedded cache) architecture

Device	Description	Used to
HDD1 **SEAGATE ST3400620AS**	400GB 7200RPM 16MB	OS, SQL Server, CrystalDiskMark, SQLIO
HDD2 HDD3 HDD4 HDD5 **TOSHIBA MK5061GS**	500GB 7200RPM 16MB	Benchmark these devices (RAID10)
RAID CARD 2 **HP Smart Array P420**	SAS Cache 1GB	Create RAID 10 array with 4 disks



4KB cluster size



64KB cluster size

Results are better in 4KB cluster size than 64KB cluster size.
To complete these tests with more accurate results we used SQLIO Disk Subsystem Benchmark Tool and SQL Server batch.

The file size used to make these tests is set to 100GB to represent a realistic database size.

Random read operations

Parameters			IO		Latency (ms)		
Block size (KB)	Thread	Outstanding	IO/Sec (4K / 64K)	MB/Sec (4K / 64K)	Min (4K / 64K)	Avg (4K / 64K)	Max (4K / 64K)
8	8	1	382,16/381,77	2,98/2,98	0/0	20/20	140/157
8	8	8	542,44/541,44	4,23/4,23	0/0	117/117	2378/1812
64	8	1	366,21/366,45	22,88/22,90	0/0	21/21	183/156
64	8	8	506,36/507,31	31,64/31,70	0/0	125/125	2182/1701
512	8	1	179,19/179,37	89,59/89,68	0/0	44/44	413/350
512	8	8	175,92/179,08	87,96/89,54	31/43	363/356	895/1141

Random write operations

Parameters			IO		Latency (ms)		
Block size (KB)	Thread	Outstanding	IO/Sec (4K / 64K)	MB/Sec (4K / 64K)	Min (4K / 64K)	Avg (4K / 64K)	Max (4K / 64K)
8	8	1	440/440,99	3,43/3,44	0/0	17/17	716/738
8	8	8	438,26/438,34	3,42/3,42	0/0	145/145	831/866
64	1	1	338,40/338,97	21,15/21,18	0/0	2/2	389/320
64	1	8	241,65/341,45	21,35/21,34	0/0	22/22	345/382
256	1	1	139,79/221,71	34,94/55,42	0/0	6/4	245/141
256	1	8	139,92/221,3	34,98/55,32	0/0	56/35	323/185

Warning, the results may be slightly skewed due to embedded RAID card cache.

On RAID10 with cache, cluster size does not have any benefit on I/O access and transfer rate except for 256KB block size.