



Hyper Historian

Data Retrieval Optimization



APPLICATION NOTE

October 2015

Description: Guide to optimizing data retrieval through Hyper Historian as well as general troubleshooting tips for disk performance.

General Requirements: Hyper Historian, TrendWorX64 Viewer, Disk I/O, Performance Monitor, Resource Monitor.

Introduction

Hyper Historian offers persistent high speed and robust data historian solution for mission critical process at field. Users who have collected years' worth of historical data require their data retrieval to be consistent and responsive. ICONICS provides a set of easy-to-use data visualization technologies but they need to be properly configured and running on a well-tuned platform to max out productivity.

The purpose of this application note is to help the user troubleshoot poor disk logging or retrieval performance, fine tune their system to yield better disk performance, and eventually boost I/O throughput.

1. Understand disk I/O and why it's important for overall Hyper Historian performance
2. Understand the importance of CPU and memory to Hyper Historian
3. General tips for configuring the TrendWorX Viewer, SQL Query Engine and Hyper Historian to yield better performance
4. Guide to using native Windows utilities to monitor disk performance

NOTE: This application note assumes you have the basic knowledge of using Windows Performance Monitor. If you are not familiar with this tool please refer to the designated sections of the MSDN website for more information.

Latency

This application note frequently refers to "latency". In this case, "latency" means the tendency for the logger to fall behind the actual data. You are most likely to notice latency in a "real time" client like the TrendWorX64 Viewer. When latency is occurring, the latest logged sample will be a few minutes or hours behind the actual time. This gap may grow over time. Latency indicates that the logger is not able to keep up with the incoming data or calculations. This application note will help you fix issues that can lead to latency.

Disk I/O and Its Importance to Hyper Historian

Disk I/O encompasses the input and output operations on a physical disk. Regardless if you are writing/reading data into/ from a file (or database) on a disk, the processor needs to wait for the file to be written or read.

Often, access time is the killer when working with a disk. It is the time the computer needs to process a data request from the processor and then either commit/retrieve data to/ from the storage device. Conventional hard disk drives (HDD) are mechanical. Machines need to wait for the disk to rotate to the desired disk sector before it can perform further operations - depending on the disk quality and speed - which is why caching data in the memory is so important for performance.

Hyper Historian is the ICONICS solution for high speed reliable data logging and reading for various industry automation. It utilizes various cutting edge technologies to yield best data performance, including the technique of data caching. However, at the end of the day, whether writing or reading data from Hyper Historian ties back to the fundamental disk performance.

This is why it's so important to create a healthy system for your Hyper Historian to begin with. For instance, physical machines are better than virtual machines. Dedicated disks are better than shared disks. The newer solid state drives (SSD) are better than the conventional hard disk drives (HDD), etc. That being said, in reality, system setup is dictated by various factors: design, environment, budget, etc. You may not be able to obtain the most ideal platform, but you do have numbers of options to optimize the setup to get close to an ideal environment. We will talk about these options later in this application note.

CPU and RAM and Their Importance to Hyper Historian

As mentioned in the first section, data caching is a very important technique to produce better IO performance, which is what Hyper Historian leverages. Plus, when your project expands, Hyper Historian will also require more memory to host the configuration. It is important to spare enough resource for Hyper Historian, especially if you're looking for high speed data logging and retrieval.

On the other hand, it is also necessary to keep at least 20% of free memory on the machine all the time. The reason is that once memory is depleted or approaching to depletion, the operating system will start using the hard disk as a virtual memory extension



Hyper Historian

Data Retrieval Optimization



APPLICATION NOTE

October 2015

of a computer's real physical memory (called space swapping). This will slow down the access time even further because now paging has to be done on disk as well.

Sufficient CPU resource will ensure that Hyper Historian has enough horsepower to process all data requests as well as online calculations. Furthermore, if by any space swapping happens to occur, you need more CPU power for your applications to deal with the swapping.

Tips for Optimizing Data Retrieval

There are a few things we can look into on the Hyper Historian side to improve the way it utilizes memory as well as to reduce I/O operations. There are also numbers of areas we need to be aware of for the two major data retrieval methods: trend viewer and SQL query engine, to optimize data output performance as well as alleviate server side load.

Hyper Historian

When you started deploying Hyper Historian for your project, there were a number of factors you needed to take into consideration. These factors also will determine your data writing and reading performance moving forward and their influence could be even larger as your project expands. They will require more engineering time and even re-design later on if you fail to make a good decision at the beginning. These factors are:

- File creation frequency
- Multiple logger groups
- Evenly distributed calculation trigger
- System settings

File creation frequency: The number of HHD files and how often they are created doesn't have much of an effect when writing the data to disk because it is a continuous process to a single destination. (Performance calculations complicate this a bit, but that will be discussed later.)

However, you may see a performance difference when reading data out of one or many HHD files. When Hyper Historian reads data it may need to access one or hundreds of HHD files, depending on your query parameters and how often you created a new HHD file when logging. Each HHD file will require a separate connection and separate I/O operations. The more HHD files you have, the more overhead you will incur when reading your data.

Performance calculations incur even more overhead per HHD file. When doing a calculation, Hyper Historian first needs to read the data from each HHD file (incurring the same overhead as mentioned above) then it will need to write the data back into those HHD files, essentially doubling the amount of overhead per file.

Because of these factors, it is generally preferable to have fewer large HHD files than many small ones. If you are experiencing poor performance with your data retrieval or performance calculations you may want to slow down your file creation frequency. If you already have a large number of small files that you'd like to combine into larger files, contact ICONICS Technical Support for assistance.

Multiple loggers: The logger here represents the internal "logging" process running inside Hyper Historian Logger Service. Each logger process accesses its designated HHD files to read and write data. With multiple loggers, you gain the ability of concurrent I/O operation, but it will put more load onto your disk. Sometimes the performance of concurrent I/O operations will make up for the extra disk load, but if your disk is heavily used you may find that multiple loggers work best if each logger is configured to store its HHD files to a different physical hard disk.

Evenly distributed calculation trigger: Calculation tags can be triggered on a specific time interval or based on the incoming data. When setting up time interval triggers its good practice to distribute them so that the total work that Hyper Historian is doing is spread evenly.

For example, if you have two calculations that you want to happen once per minute, rather than having both of them happen exactly on the 00 second mark you can create two triggers, one to occur at every minute exactly and one to occur 15 seconds after the minute. That way, rather than doing two calculations at once and then being idle for almost a whole minute it'll do one calculation at 00 seconds and another at 15 seconds.

System settings: The default settings for Hyper Historian are a good starting point for most implementations. However, if you are seeing some undesirable behavior (such as the data replay lagging) there are a few things we can look into to improve performance from Hyper Historian side.

1. Maximum Auto-Recalculation Time Extent

This can be found under System Settings → Options → Performance Calculations. The idea of this setting is to protect data integrity of the calculations in case unexpected process



Hyper Historian

Data Retrieval Optimization



APPLICATION NOTE

October 2015

outrage occurs. By default it will go back 24 hours. If a system is undergoing some degree of data replay lag it is recommended to turn this feature off by setting it to 0 hrs. With this feature disabled Hyper Historian will not bother recalculating all the calculations, which frees up more resources for standard logging and calculations.

2. Maximum Unused Data Blocks Lifetime (sec)

This setting is located under System Settings → File Logger → Cache. When a client or performance calculation is finished with a specific set of data Hyper Historian will keep this data in memory a bit longer in case it is needed again. This setting determines how long that unused data is kept. Increasing this duration can improve data read performance for clients that switch back and forth between sets of data, or if there are calculations which require frequent access to a specific set of historical data. However, increasing this duration will require more memory resources from the system.

3. Maximum Operation Queue Length

This setting is under System Settings → File Logger → Logger which can be increased to load more read/write operations into queue for each HDD file. It improves the logger performance at the cost of system memory so be careful with the number you enter. The default value should be suffice for most cases. Consider increasing it only if the logger is unable to process all incoming requests.

4. Maximum Online Loggers Count

This setting could be found under System Settings → File Logger → Master Logger. The default is a sufficiently large number, so in most cases it won't need to be changed. However, if you have many loggers and plenty of free memory you may consider increasing the number.

In addition to these settings, the hard disk performance is a critical factor to Hyper Historian performance. If you are experiencing sub-optimal performance from your historical reads or writes you will want to investigate your disk performance to see if either Hyper Historian is not configured to utilize the disk properly or if there are other activities going on that are hogging the disk performance and effecting Hyper Historian. We will discuss general disk performance troubleshooting techniques in a later section.

TrendWorX64 Viewer

The TrendWorX64 Viewer is a frequently used client for data replay. While setting up a trend viewer can be as easy as dropping it into a client such as GraphWorX64, it may require some fine tuning to achieve maximum performance without harming the backend server.

Below are a list of settings we can examine to get the best performance. For more detailed information on these and other settings, please refer to the TrendWorX64 Viewer related application notes.

Sampling Method

In versions prior to 10.9 the resolution of data on the screen was determined by one setting called "Data Collection Rate". In version 10.9 or later there are now three different settings collectively called "Sampling Methods" that determine how the points are sampled.

- **(Sampling) Interval:** This setting applies only to aggregated data (client defined aggregates) and has no effect on raw data. Note that client side aggregates take time and resource to visualize data so if you're after performance and precision, you should consider implementing your aggregates on the server side, however server side aggregates are less dynamic (they cannot be adjusted in runtime). Note, this setting does not apply when the time-aligned is used. Auto (Sampling) is used instead.
- **(Static) Count:** This setting is useful for analog data or data analysis using free zooming or dynamic ranges. It helps reduce the amount of data transferred over the network.
- **Auto (Sampling):** This setting only applies when the time-aligned axis is used, otherwise the Interval setting applies. It shows N samples per major tick mark, where N is configurable.

Animation Rate

The Animation Rate was called "Display Refresh Rate" in versions prior to 10.9. It is the speed of the chart animation in real-time mode (unfrozen). It has no direct effect on data read performance, however slowing down the animation speed can improve the performance of the client itself and free up other system resources.

History Read Rate

History Read Rate was called "History Refresh Rate" in versions earlier than 10.9. This defines how often you read historical data from server in real-time mode (unfrozen). This is the parameter that has the biggest impact on both server and client performance. The faster your history rate is, the more often it requests data from backend server thus adds more load.



Hyper Historian

Data Retrieval Optimization



APPLICATION NOTE

October 2015

By default, to help protect the viewer from being overwhelmed by a very fast History Read Rate there is a safeguard option enabled by default called "Optimize History Read". When this option is enabled the History Read Rate only defines the fastest allowed rate, and the actual read rate is calculated based on the time range.

Session Buffer

The session buffer is the total amount of data samples displayed on the viewer. There are three related settings:

- **SessionBufferSize** is defaulted to 1000. All samples above this limit will be dropped. This setting does not apply when **IsSessionBufferSizeAutomatic** is enabled.
- **IsSessionBufferSizeAutomatic** is enabled by default. When this is enabled, the buffer size will be automatically adjusted to hold all samples returned by the server up to the maximum.
- **MaxAutomaticSessionBufferSize** is defaulted to 10000. It's the maximum number of samples which will be displayed on screen. This setting only applies when **IsSessionBufferSizeAutomatic** is enabled.

Max Automatic Session Buffer Size is now available under the Global Pen Settings. If you see "Not all data visible" in the legend for your historical pen you may want to increase the Max Automatic Session Buffer Size.

SQL Query Engine

The SQL Query Engine is a very popular option for building your reporting solution. Because it involves SQL there is potentially a large number of optimizations that could be done within the SQL query itself, too many to go over in this document. However, we will mention one setting which could improve query throughput. That setting is **MAX_ROWS_PER_CALL**.

MAX_ROWS_PER_CALL is one of the attributes contained in the **QuerySettings** table. By increasing this number you can increase the total throughput of query engine. You can read your current **QuerySettings** by running the following query:

```
select * from openquery(HH2, 'select * from querysettings')
```

To update the **MAX_ROWS_PER_CALL** setting, run the following query:

```
select * from openquery(HH2, 'update querysettings  
set MAX_ROWS_TO_RETURN = ''5000''')
```

To find out more about the SQL Query Engine, please refer to the application note entitled, *Hyper Historian - SQL Query Engine Quick Start*.

Monitoring Disk Performance

If you have ever wondered how to monitor your hard disk performance to be able to tell in advance whether problems like data replay latency will occur, this section will describe how to use some built-in indicators and Windows tools to help.

MonitorWorX Viewer

This is an ICONICS utility that ships with standard installations. Often people use it to check their license status or monitor a redundant node pair, but you can also use it to monitor latency between data logging and committing. Under the redundancy tab, you will see a Hyper Historian node. This should be visible even if you do not have redundancy configured:

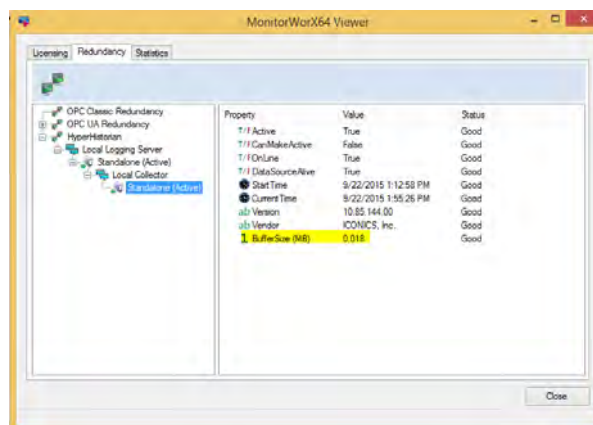


Figure 1 - MonitorWorX Viewer

If you drill down in Hyper Historian all the way to the collector level you will see a buffer size (highlighted above). This tells you how much data has been logged in memory but hasn't yet been committed to disk. Under normal circumstances you should see a buffer size running below 2 MB. It's okay for the buffer size to build up to couple of MB in a short period of time as long as it drops back less than 2 MB thereafter. A consistent high level of buffer size may indicate latency is going to happen or is already happening, and needs to be addressed.

Resource Monitor

The Windows Resource Monitor provides more information than the Task Manager alone. Most notably for our purposes, it provides real-time disk activity associated with each process and disk. It provides a quick glance of what's going on the disk and what process is contributing to your I/O load.

