ClearSCADA

Software for Telemetry and Remote SCADA Solutions

Performance Guidelines

March 2018

Schneider Electric

Introduction

ClearSCADA is a high performance SCADA system, capable of scaling to large databases and/or fast data processing. Each new release of ClearSCADA provides new features and higher performance to cope with the increasing demands of real-time monitoring, control, storage and analytics.

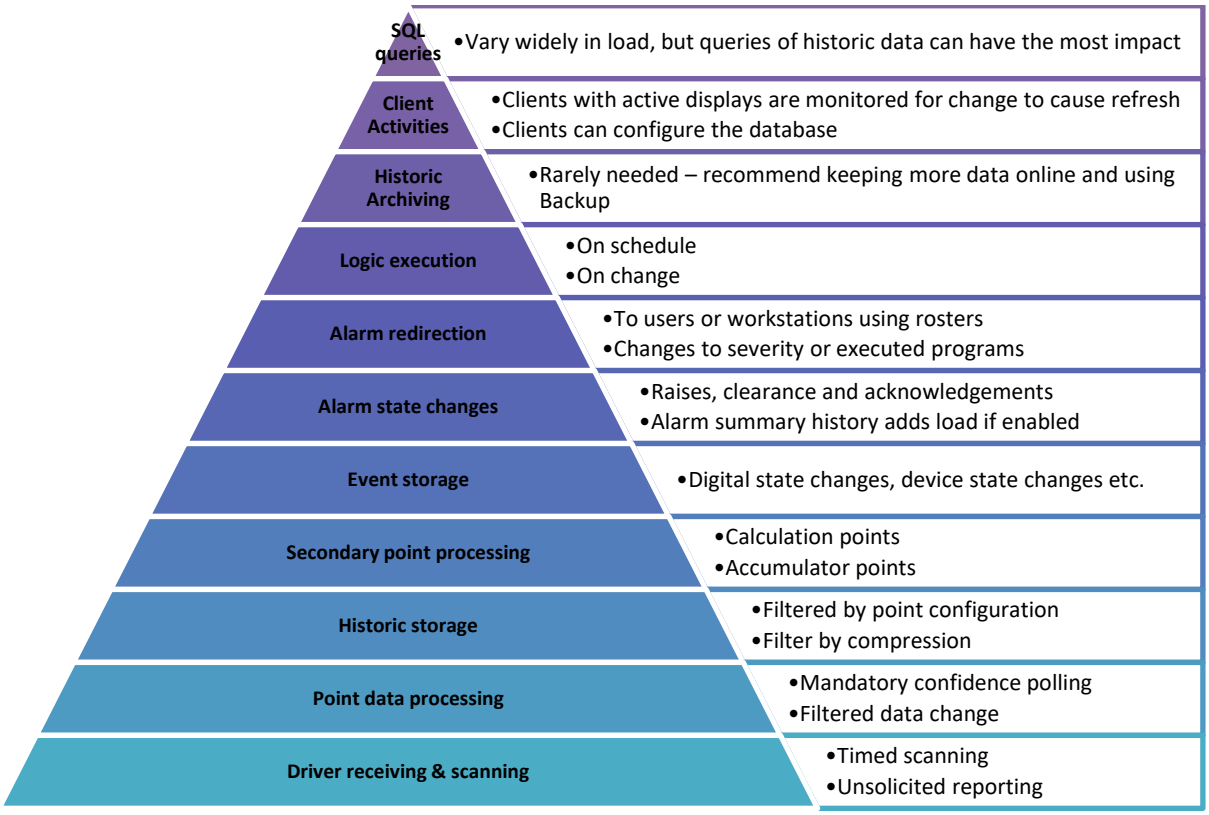
This document serves to provide some guidelines for performance when creating and maintaining larger ClearSCADA deployments.

ClearSCADA Server Activities

The limitations of a ClearSCADA system are a combination of the physical resources of the system's computers - memory size and speed, disk access time and CPU speed - and of the database architecture which co-ordinates system activity. Having one fast physical resource such as disk is important, but the architecture will always be limited by the slowest resource. In high load conditions we find that the CPU and memory speed are limiting factors. This is typically the case with SCADA systems which are scanned and updated at high rates, and is due to the need for a database to control updates using locking so that integrity can be assured.

Typical of most SCADA systems, the loading on ClearSCADA consists of many different activities, all of which are configurable to control the balance of work. A good way to visualize this is the pyramid below.

At the bottom of the pyramid is the basic receipt of data. This can be scanned using simple timed methods, or input by the devices using 'unsolicited' or 'asynchronous' data reporting. The latter methods are often better because there is no wasted activity reading unchanged data, although protocols such as Modbus only have a scanning capability. As we move up the pyramid, the data processing quantity reduces, because each layer filters redundant or irrelevant data. For example, the driver may read all points in a given time, but the unchanged (or little changed) data is not passed to the next layer. Similarly, the historic storage layer will either filter by configuration of value dead-bands or by time intervals.



As we go up the pyramid, while the number of processing steps decreases, there is often an increase in the amount of computational and storage workload. For example, the processing of historic storage requires more CPU cycles to store a new record in the database, and the processing of an alarm state change for a point will require new event log records and changes to many associated fields of a database record. The diagram is naturally not to scale, and there is significant variability in the workload of items at each level, but it is a useful guide to configuring a system. For example, historic storage data rates need to be controlled to avoid storing redundant data.

Alongside the activities listed in the pyramid, there are others such as the flushing of database information to disk and the synchronisation of data from the Main server to the number of Standby servers in the architecture. The latter loading is approximately proportional to the aggregate of all the other activities in the pyramid.

Disk access time can restrict the performance of the system, and the activity types which are principally affected are:

a. Queries for historic data - where a large amount of historic data needs to be searched and/or retrieved.
b. The flushing of historic data to disk (including events and alarm summary records).
c. Access (read or write) of large Data Table objects by SQL queries.

There is a specification of ClearSCADA limitations for some of these pyramid layers. This is detailed in the documentation under the heading and search term 'Operational Limitations - Server'.

This includes a limit to the number of historic data records stored per minute and per point, which is 4 - a maximum interval of 15 seconds. This does not prevent the field hardware from being read at a much faster rate, that could be 1 second for example, nor is this an enforced limit, because ClearSCADA points can be configured to store data at this rate and higher, but there are follow-on impacts to configuring the system in this way.

While a typical application for ClearSCADA will need some data to change rapidly, that rapid rate is never required to be sustained. For example, analogue trend data needs a high storage rate when the information content is high - when it changes more, and at a low storage rate when the process is quiescent.

Maximizing Performance

A question often asked is 'how many x can be processed in y seconds?' This is very difficult to answer, particularly because if that is the only activity, this number can be very high. But adding a spread of other activities required of the system will reduce the available capacity.

There are also some good practices concerning leaving headroom for unusually high activity if the system is exposed to unexpected conditions. Clearly it is important to design the system so that not only is there no data loss, but also that users of the system get an acceptable level of performance, such as display times and screen update times, particularly when the input changes are important to operators.

ClearSCADA is designed to avoid 'missing' any of the activities in the pyramid. For example, historic data is queued and cached at high burst rates, then stored later when there is time for the database to do this. Alarms and alarm redirections are also executed in a way to ensure that they are not missed. There are some parts of the pyramid which can be configured to skip processing at busy times. Logic is the best example here. If a Logic program is queued from a state change, then that queue size can be limited to avoid unnecessary load. Also, some timed activities for scanning and Logic will be deferred if the previous timed activity is still being processed.

There are many variables which a configuration team can use to control system load; how can it be determined what the desired load should be? Measuring the CPU and Disk activity of a server is not enough, due to the internal controls of database locking described above. While there can be no single number which guides all the activity of ClearSCADA a good measure which we have found effective is the total time the database spends in a locked state - in other words the time spent updating or reading data. It is important this is not too high so that other activities can take place and the database remains quickly available for new activity to occur.

This total lock time can be found using the ClearSCADA Server Status tool. It is the sum of two numbers in the top line of the 'General | Locks' page. This line is labelled 'Database' on the left, and the columns '% Time in Excl Lock' and '% Time in Shared Lock' should be added. Note that this will be high on system start-up and should gradually decrease during continuous operation. It can be reset by a right-click menu. Total lock time can also be read by summing two OPC tags: '#LOCK.1.% Time In Excl Lock' and '#LOCK.1.% Time In Shared Lock'. You could create a ClearSCADA Calculation Analogue Point with the expression:

```
("#LOCK.1.% Time In Shared Lock" + "#LOCK.1.% Time In Excl Lock")*100
```

We recommend that this total lock time remains below 50% on the Main server for most of the duration of ClearSCADA operation. Naturally there may be busy periods, but the long-term average should be less than our suggested figure.

Using this lock time guideline may require a configuration teams to try alternative strategies to get the most effective use of the system. In this case using the pyramid to understand loading will help to guide the team towards an effective balance.

The following table concludes this guide. It lists each element of the pyramid, how the amount of processing can be measured, and what steps could be taken to influence that amount.

Note that all items in the last column are advisory and can be exceeded, noting that the overall load is more important. The figures are based on large systems with adequate CPU, memory and disk performances.

Load Type	Description	How to Measure	How to Change	Expected order of Magnitude
SQL queries	Vary widely in load, but queries of historic data can have the most impact	Query CPerformanceStats table. Select QueryExecuteTotal from CPerformanceStats. Execute this twice with a known time interval and observe the difference. Server Status and Snapshot Log File - "Latest, Largest and Longest queries"	Ensure queries in Logic are marked NOCACHE unless necessary. Keep queries of history to short periods if possible.	Aim for fewer than 500 queries per minute, but fewer than 10 historic queries per minute.
Client Activities	Clients with active displays are monitored for change to cause refresh	Server Status and Snapshot Log File - OPC-DA Subscriptions	Review mimic complexity. Reduce or avoid indirect tag use (square bracket syntax). Reduce scripting if possible.	Design mimics with fewer than 500 tags (individual data items) in total, and avoid indirect tags, or use indirect tags only to objects which change seldom.
Client Activities	Clients can configure the database	Event log. Configuration change log (if enabled).	Configuration activity causes the configuration database to be processed and saved to disk.	
Client Activities	Logic can configure the database	See the column 'Write Config Count' in the System Query in ViewX 'Logic Execution Status'	Take care to reduce or eliminate Logic which writes to configuration fields of database objects. Not only does this cause data saves to disk and sync to Standbys, it also causes re-evaluation of validation.	Avoid using Logic which performs configuration regularly.
Historic Archiving	Rarely needed – recommend keeping more data online and using Backup	Find database objects of type 'Archive'. Check linked schedules.	Configure archive to work weekly, and ensure that it completes each week, to avoid a build-up of unarchived data. Consider not using archive, and retaining history for longer on all servers.	

Load Type	Description	How to Measure	How to Change	Expected order of Magnitude
Logic execution	On schedule and On Change	See the columns 'Overruns', 'Execution Queue Max', 'Queued Executions Lost' and 'Execution Count' in the System Query in ViewX 'Logic Execution Status'	Look for overruns and reduce scheduled frequency. Look for queued executions lost and check for inputs which change too frequently and reduce their scan or data update rate. Look for high execution counts and review why.	Aim for fewer than 200 logic executions per minute.
Alarm redirection	To users or workstations using rosters. Changes to severity or executed programs	Query CPerformanceStats table. Select AlarmRedirTotal from CPerformanceStats. Execute this twice with a known time interval and observe the difference. See server Status tool - page 'Database | Alarm Redirections'. Look for rapid changes in the list.	Review whether all states or severity levels need to cause redirection.	Aim for fewer than 20 alarm redirections per minute.
Alarm state changes	Raises, clearance and acknowledgements	Query CPerformanceStats table. Select AlarmRaisedTotal from CPerformanceStats. Execute this twice with a known time interval and observe the difference. Review SQL query of alarm raise counts using: SELECT "FullName" AS "~FullName", "Id", "RaiseCount" FROM CDBPOINT ORDER BY "RaiseCount" DESC	Use persistence to reduce fleeting alarms. Review plant behaviour to resolve faulty plant.	Aim for fewer than 50 alarms per minute. Consider the burden on operator interactions.
Alarm state changes	Alarm summary history adds load if enabled	Check Server Configuration	Alarm Summary is a feature which is essential if you require Alarm Adviser	Be careful when enabling Alarm Summary on an existing system with high load - it may be necessary to review that load first.
Event storage	Digital state changes, device state changes etc.	Query CPerformanceStats table. Select EventsProcessedTotal from CPerformanceStats. Execute this twice with a known time interval and observe the difference. To look at event counts from simple drivers, use "Select Id,FullName,PointEventCount from CPointSource". From ClearSCADA 2017 R2, see the column 'EventCountPreviousHour' of all objects.	For points which report excessive events, check and implement persistence and verify correct plant operation. If the event is not required for an object and/or state, reconfigure the Severity field to 'None'.	Aim for fewer than 500 events per minute. Also aim for fewer than 20 events per file granule per minute. The granule file stores events for a group of consecutively numbered objects, and the group size is set in the server configuration tool. Look to keep granule file sizes in line with limits in the documentation and the default granule size alarms.

Load Type	Description	How to Measure	How to Change	Expected order of Magnitude
Secondary point processing	Calculation points	Query CPerformanceStats table. Select PointsCPntCalcTotal from CPerformanceStats. Execute this twice with a known time interval and observe the difference. See and compare process counts using SQL: 'SELECT ID, FULLNAME, PROCESSCOUNT FROM CDBPOINT WHERE TYPEDESC LIKE 'Calculation %' ORDER BY "ProcessCount" DESC'	Review high frequency updates. Choose on-input or on-time processing appropriately.	Aim for fewer than 1000 calculation point updates per minute.
Secondary point processing	Accumulator points	Query CPerformanceStats table. Select AccProcTotal from CPerformanceStats. Execute this twice with a known time interval and observe the difference. Look for processing overruns using 'SELECT ID, FULLNAME, OVERRUNCOUNT FROM CRESETTINGACCUMULATOR WHERE "OverrunCount" <> 0'.	Timed report intervals may be too short. Selecting 'Continuous' will cause processing for every report time. Accumulators are updated for each process of the input point. Check whether each update type (checkboxes) are really needed.	Aim for fewer than 1000 accumulator point updates per minute.
Historic storage	Historic values can be filtered by point configuration, and by compression.	Query CPerformanceStats table. Select HistWriteRawTotal from CPerformanceStats. Execute this twice with a known time interval and observe the difference. To see individual point historic data quantities, use this query: 'SELECT P.ID, P.FULLNAME, H.UPDATECOUNT FROM CDBPOINT AS P JOIN CHISTORYBASE AS H USING (ID) ORDER BY "UpdateCount" DESC', and compare over a time period.	Review point configuration to reduce unnecessary data storage. Accumulators may need Reset values stored and not Report values. Points may need only significant change values and not report values stored. If it is necessary to update current values often, consider using historic compression, on the Historic tab of points.	Aim for fewer than 5000 historic storage items per minute.
Point data processing	Point processing includes timed scanning of values, retrieval of logged data (as supported by the protocol) and confidence polling. Note that confidence polling forces historic storage, and is intended as a 'slow' backup scan, such as daily.	Query CPerformanceStats table. Select PointsProcessedTotal from CPerformanceStats. Execute this twice with a known time interval and observe the difference. To see individual point updates, use this query: 'SELECT ID, FULLNAME, PROCESSCOUNT FROM CDBPOINT ORDER BY "ProcessCount" DESC', and compare over a time period.	Review scanning times, reduce confidence poll intervals. Widen Significant Change, increase Persistence, and widen alarm Deadbands.	Aim for fewer than 10000 point data processes per minute.

Load Type	Description	How to Measure	How to Change	Expected order of Magnitude
Driver receiving & scanning	Timed scanning, logged data and unsolicited reporting.	Driver log files. Communications logs. These will indicate what is being requested from the device, and what data is received.	The driver is the first 'filter' for data updates, and the configuration should be designed so that unrequired changes are 'seen' by the driver but not passed up to the database. Review scanning times, reduce confidence poll intervals. Widen Significant Change, increase Persistence, and widen alarm Deadbands.	No limit - but consider splitting devices across multiple sets/channels to allow separate CPU threads to distribute load.

Conclusion

The ClearSCADA team is committed to providing a high-performance system. Each new release provides more features and more performance. ClearSCADA combines unique Telemetry/Remote SCADA features, best in class driver support, and a growing scalability to meet the challenges of the 'original' Internet of Things.

Performance tuning is an important part of both system design and maintenance. In addition to this document, you can also find the ClearSCADA Design Guidelines document, search the ClearSCADA and consult the ClearSCADA Resource Center at:

http://resourcecenter.controlmicrosystems.com/display/CS/Home