A Real-Time Data Acquisition System for the Laguna Verde Nuclear Power Plant

ILSE LEAL AULENBACHER, JOSÉ MARÍA SUÁREZ JURADO, EFRÉN R. CORONEL FLORES
Gerencia de Supervisión de Procesos
Instituto de Investigaciones Eléctricas
Reforma 113, Col. Palmira, Cuernavaca, Morelos
MEXICO
{ileal, jmsuarez, coronel}@iie.org.mx http://www.iie.org.mx

Abstract: - This paper focuses on a Data Acquisition Real-Time System developed for the Laguna Verde Nuclear Power Plant in Veracruz, Mexico. Due to the fact that the data acquisition modules needed to be replaced, the need for a New Acquisition System arose. Replacing the former data acquisition system has several technical challenges, including the fact that it must work together with the former system. Additionally, it must remain online during the whole process, since the Data Acquisition System is one of the nuclear power plant most important monitoring systems and it is required for its operation. This paper focuses on the key software design aspects of the New Acquisition System, which were used to address the main system requirements: Integration with the former data system, multiple data source support, high-frequency data storage and high-speed data access. The New Acquisition System is composed of a vast number of modules. This paper specifically focuses on Acquisition Subsystems, the Central Acquisition Process and the Master Historical Archive Process.

Key-Words: - Data acquisition system, Real-time, Linux, Software, Design, Archive, Acquisition Subsystem

1 Introduction

The system described in this paper is a mission-critical, high-availability and high-speed real-time system capable of acquiring thousands of signals that power plant operators and engineers use to monitor the state of the nuclear power plant and its different processes.

In order to monitor the plant, operators need to be able to monitor groups of signals in real-time as well as in extended periods of time. Therefore, a very important aspect of this kind of systems is the ability to generate accurate historical data archives and being able to extract useful information that assists engineers in decision-making through statistical analysis and calculations.

During the last two decades, the Laguna Verde Nuclear Power Plant has been using a Data Acquisition System (DAS) that runs on a VAX/VMS platform together with Analogic data acquisition (DAQ) hardware modules. DAQ modules are components of a complete data acquisition system which perform the input, processing and conversion to digital format of analog signal data measured from a system or process. The resulting data is then transferred to a computer for display, storage and analysis [1]. Analogic and RTP are examples of DAQ hardware manufacturers.

Due to obsolescence factors, Analogic DAQ modules need to be gradually replaced by RTP DAQ modules.

To make the DAS replacement process more secure and because the DAS is very important and is required for power plant operation, the safest option was to perform a gradual migration. Hence, Analogic DAQ modules are to be replaced gradually for RTP DAQ modules over time. In order to achieve that without stopping the DAS operation, the New Acquisition System (NDAS) has to be capable of acquiring data not only from RTP modules, but also from Analogic Modules. Therefore, the NDAS must be flexible and reliable, since it needs to be operational during the several stages of Analogic DAQ module replacement.

In addition, it must provide fast and reliable data archiving through different types of historical archives which feature high-speed data access and efficient data extraction routines.

This paper describes the main software design aspects of data acquisition –including central processing and acquisition subsystems– and historical archive generation for the NDAS.

2 NDAS key features

As described in the introduction, the NDAS was designed taking into consideration several requirements to ensure it would be flexible and powerful enough to handle large amounts of data at a high frequency.

The following section describes the main NDAS features and how they were addressed and implemented.

2.1 Provider-independent

One important requirement for the NDAS is that it must remain in operation as long as possible. Software licensing plays an important role in this aspect. Several options were considered [2]; however the option which was deemed most appropriate was to provide our client with a custom-made system, independent from commercial providers. One of the main reasons is that since the DAS replacement is gradual, the NDAS has to be very flexible and highly customizable. Commercial systems rarely provide access to source code. In contrast, with this approach our client has access to fully documented source code, gaining the power to freely maintain the system or incorporate new software modules in the future.

Another important consideration in NDAS design is the need for the system to be able to acquire data from different DAQ modules and support changes such as the incorporation of new DAQ modules, or switching a data source from a group of signals.

The NDAS was designed not only to solve the Analogic DAQ module replacement problematic, but to provide continued support for other DAQ modules. To provide such flexibility, the NDAS acquisition process (and the system in general) was designed to be provider-independent by being prepared for future incorporation of new DAQ modules, without vendor-specific constraints.

The platform of choice was PC server under Linux, because it is a well-supported open-source operating system which is fully customizable in all its components and has a low failure rate and system maintenance time [3].The NDAS was developed using C++ with the GNU gcc compiler.

The servers in which the NDAS is currently installed have 16GB RAM memory, dual processors and three high-speed 300GB disks. However, it is important to note that the NDAS has been tested on several machines with good results. The system, in its configuration is very flexible and can be configured to acquire and process from tens to thousands of variables.

2.2 Acquisition Subsystems

Because the NDAS was designed to work together with the former DAS, one of the most important system requirements is support for multiple data sources. That means that the system must be flexible enough to support acquiring data from RTP, Analogic and other DAQ modules. In order to do this, the system was conceptually divided into various units which interact among them while maintaining their logical and functional independence.

The NDAS is a system composed of several modules. The module that deals with acquisition is called the Central Acquisition Process (CAP), which polls acquisition subsystems (AS) and requests data each second. Each acquisition module, be it RTP, Analogic or any other, needs some sort of software driver that will deliver data to the NDAS Central Acquisition Process. The software that performs this operation is called an Acquisition Subsystem.

An acquisition subsystem connects and acquires data from different data sources, such as data servers or hardware acquisition modules. Acquisition subsystems function as independent processes within the NDAS, each one different from the other depending on the characteristics of the hardware module from which data is acquired as well as the communication protocol used to obtain data.

Data can be acquired on real time or in historical mode. Real-time acquisition involves a 1Hz sampling rate. In contrast, Historical mode acquisition involves sampling rates that can range from 1 to 250Hz. Sampling rates are determined by the hardware acquisition module features.

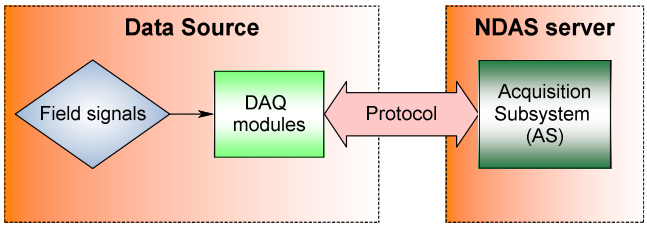

Fig. 1

Fig. 1 is a block diagram that illustrates the communication scheme between data sources and NDAS servers. Field signals are wired up to data acquisition modules. Data acquisition modules convert analogical data (temperature, pressure, etc.) into digital data (i.e. bits). An acquisition subsystem uses a protocol (such as TCP, ModBus, etc.) to connect to the DAQ module and acquire data. Afterwards, the Acquisition Subsystem will make the acquired samples available to the NDAS system.

It is important to mention that Acquisition Subsystems are so flexible that software data providers can also be incorporated to the NDAS.

Fig. 2 shows an alternative setup in which the system connects and communicates with a software data provider, through a protocol or driver.

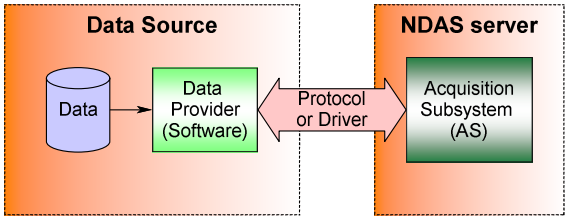

Fig. 2

Acquisition subsystems integrate to the NDAS by putting acquired samples in shared memory areas through an inter-process communication protocol, which controls the way in which the NDAS internal processes communicate and exchange information (see section 2.3.1).

Fig. 3 shows a diagram that depicts the components of an acquisition subsystem. A typical subsystem has Acquisition Subroutines which establish a connection to different sources in order to acquire data. These subroutines also acquire real-time data with sampling rates that range from 1 to 4Hz and historical data with sampling rates of up to 250Hz. Data sampling rates depend on the data source features and configuration.

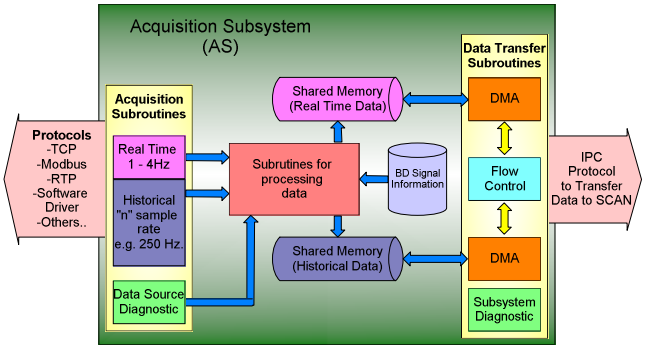

Fig. 3

Acquired data are processed and placed in shared memory areas ready for their transfer to the historical archive. Besides acquiring samples, Data Acquisition Subsystems obtain DAQ diagnostic information, which gives the NDAS the possibility to report problems with DAQ modules in a timely manner.
Diagnostic information includes DAQ modules temperature, card status, online/offline status, communication problem warnings, etc. With this information at hand, operators can easily get information on any hardware-related issue.

Acquired data are processed by other subroutines in the subsystem which verify the integrity of the obtained information by validating different aspects defined in the database such as correct conversion between raw counts and engineering units, acceptable signal ranges, etc. Once the information is validated, it is stored in a subsystem-specific internal shared memory area which serves as a buffer for both real-time and historical data.

To transfer acquired data to the SCAN process, the subsystem provides routines that perform information flow control over the information requested by the SCAN process. Data is transferred using direct memory access to the SCAN shared memory area.

Communication DAQ modules are independent from the protocol used to transfer data. Different protocols can be used; for example: TCP/IP, proprietary RTP protocol, Modbus over TCP or any other TCP embedded protocol. Moreover, software drivers can also be used, as in the case of a SQL database query.

Fig. 4 shows a schematic of the different subsystems currently installed in the Laguna Verde Power Plant. As depicted, Acquisition Subsystems obtain data from data servers or DAQ hardware directly (as in the case of RTP).

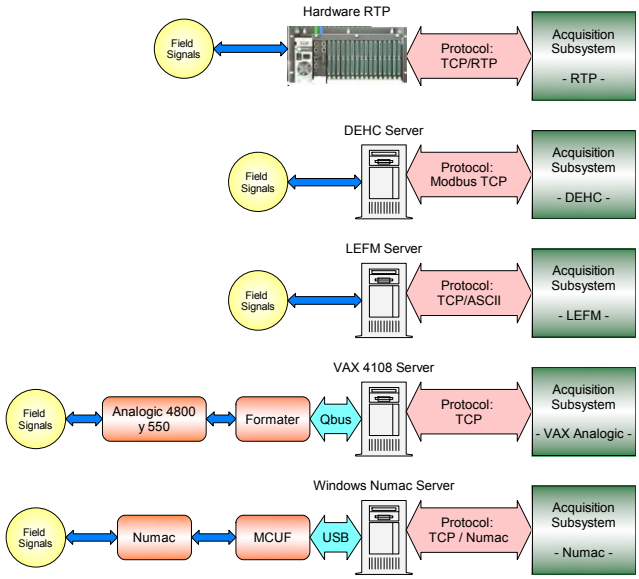

Fig. 4

Communication between data sources and Acquisition Subsystems can be performed through a variety of protocols. It is important to note that conceptually, subsystems are designed to work and communicate with the NSAD regardless of the protocol or data provider used.

The following section describes the Central Acquisition

Process, which coordinates the reception of data from the Acquisition Subsystems and integrates the received data into the SCAN historical archive. This process is critical for timely, efficient and reliable data storage.

2.3 Central Acquisition Process

Fig. 5 is a block diagram that illustrates how the elements described above interact among them. As mentioned in the previous section, DAQ modules transform field signals (temperature, pressure, etc.) into electrical signals. For each DAQ module, there is an Acquisition Subsystem which communicates with the Central Acquisition Process.

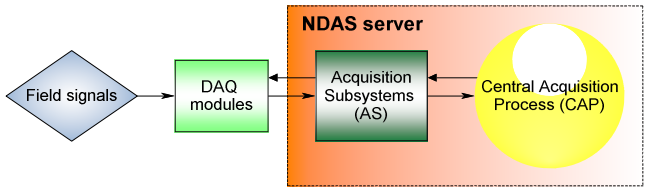

Fig. 5

In order to maintain functional independence, each Acquisition Subsystem communicates with the CAP via a protocol, which defines the way Acquisition Subsystems interact with the NDAS.

2.3.1 CAP-AS Inter-process protocol

The Central Acquisition Process – Acquisition Subsystem protocol controls the way Acquisition Subsystems incorporate into the acquisition process. This protocol is based on Linux inter-process communication (IPC) facilities.

Among IPC facilities, shared memory provides total flexibility as well as the highest speed compared with other mechanisms such as queues or sockets [4]. Although shared memory is fast, it also needs proper control mechanisms to ensure data integrity. This is achieved by using semaphores and timers which regulate when shared memory areas are accessed [5].

The protocol allows Acquisition Subsystems to communicate with the CAP. However, Acquisition Systems view the CAP as a black box that requests data periodically. Likewise, the CAP regards Acquisition Subsystems as black box data providers which respond to data requests.

The IPC protocol is based on a set of semaphores, shared memory areas and high-resolution timers which are controlled by the CAP. Through the IPC protocol, the CAP performs one-second acquisition cycles.

During an acquisition cycle, the CAP sequentially queries each AS and signals it to deposit data in designated shared memory buffers. An acquisition cycle is performed in two phases: In the first phase, Acquisition Subsystems deposit real-time data (1 Hz sample rate) for each signal. In the second phase, Acquisition Subsystems deposit historical (up to 1KHz sample rate) for each signal.

Fig. 6 is a simplified illustration of the relationship between Acquisition Subsystems and the Central Acquisition Subsystems, which communicate through the IPC protocol.

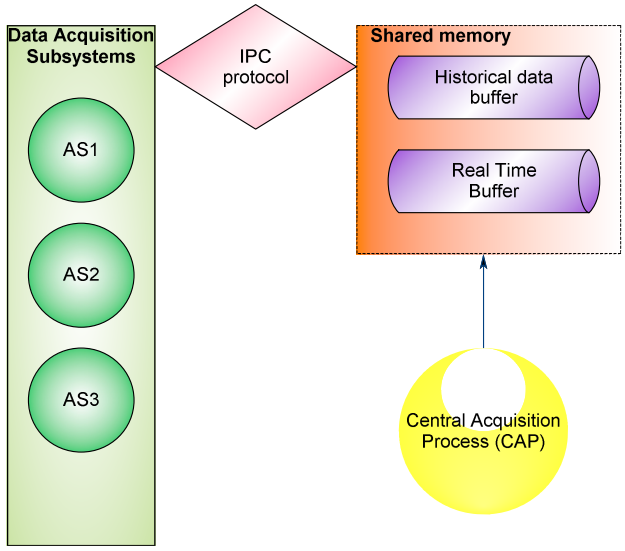

Fig. 6

The IPC protocol is embedded in a C++ object which provides Acquisition Modules with access to the necessary resources (semaphores, shared memory areas and databases) to incorporate to the acquisition cycle.

The IPC protocol is fully documented and further Acquisition Subsystems can be developed independently by third parties by instantiating the protocol object and thus accessing the semaphores and shared memory areas that will allow them to integrate as a NDAS data source.

2.3.2 Databases and data sampling

Databases play a very important role in NDAS configuration. Because databases need to be read by various processes and for the sake of high-speed access,

they are loaded directly into a shared memory area as plain ANSI text files.

The NDAS can be configured to acquire data from several Acquisition Subsystems. The Master Signals Database contains the relation of signals to acquire. Each record contains information about each signal such as the signal identifier, type of signal (analogical or digital) and data source (from which acquisition subsystem it is acquired).

The Master Sample Plan contains the frequency at which samples are acquired. In most cases, the sample rate for analogical samples is 250Hz and for digital samples 1 KHz. Both the Master Signals Database and the Master Sample Plan are accessible by the Acquisition Subsystems and the Central Acquisition Process.

The fact that each signal be assigned an Acquisition Subsystem as data source has proved really useful, since it allows switching between DAQ modules for a signal or group of signals during the DAS gradual replacement process just by modifying the databases.

2.4 Advanced historical data archives

The NDAS acquires and processes more than seven thousand signals from different acquisition modules. Considering that the NDAS supports acquisition rates up to 1 KHz for each signal, it is necessary to provide a fast and reliable way to store information so it can be available for further analysis. Historical archives are binary files that store samples for a given period of time.

2.4.1 Master Scan Archive

NDAS most important data archive is called Master SCAN archive. A SCAN archive stores all samples just as they are acquired, with no filter or compression of any kind.

The SCAN archive is the Master Historical Archive of the NDAS because it is accessed by several processes to read data. With that in mind, the SCAN Master Historical Archive resides in a shared memory area since it is the fastest method of inter-process communication [5].

One of the reasons high-speed historical data access is crucial for the NDAS is that it is necessary for creating alternative historical archives. In addition to the shared memory SCAN archive, the NDAS supports other types of historical archives that are stored on disc and feature different compression methods and structure. The master SCAN archive serves as the source of data for the processes that create these alternative archives. It is important to note that the NDAS supports several historical archives at the same time and hence the importance of fast and reliable inter-process communication.

The process that generates the Master SCAN Archive is referred to as the Scan Process in the following sections. The Scan Process is synchronized with the Central Acquisition Process and thus with the Acquisition Subsystems to seamlessly integrate all data in one single shared-memory archive, just as if all samples came from one single data source.

Fig. 7 is a simplified block-diagram that shows the overall synchronization relationship between the Acquisition Subsystems, the Central Acquisition Process, the Scan Process and the related shared memory areas.

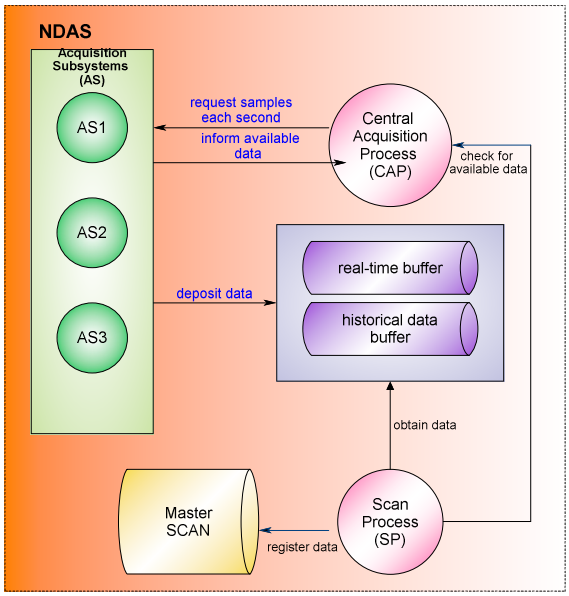

Fig. 7

The above figure shows how the Central Acquisition Process requests data each second to each Acquisition Subsystem, which in turn inform whether they have available data. If they do, after all Acquisition Subsystems finish depositing data, the Scan Process records the information in the Master Scan Archive.

2.4.2 Master Scan integrity

To maintain integrity, the Scan archive must ensure that it records samples of all the signals *with the same time stamp*. For example, suppose that signals are acquired from Acquisition Subsystem A and Acquisition Subsystem B. If data are requested for date *f*, and AS-A has f+2secs worth of data and AS-B has f+1secs, by definition the SCAN Process would register the amount

of data, available to *all* Acquisition Subsystems. Therefore, in this case, it would record f+1secs of samples.

Fig. 8 illustrates the above example. It must be noted that a single Acquisition Subsystem is usually comprised of several DAQ modules. The blue squares in Fig. 8 indicate the number of seconds available per signal. Although two signals may share the same AS, they still might come from different DAQ modules. When an AS reports the number of available seconds, it always takes the minimum as a reference. Thus, in the example below, the SCAN Process would record 1 second worth of information.

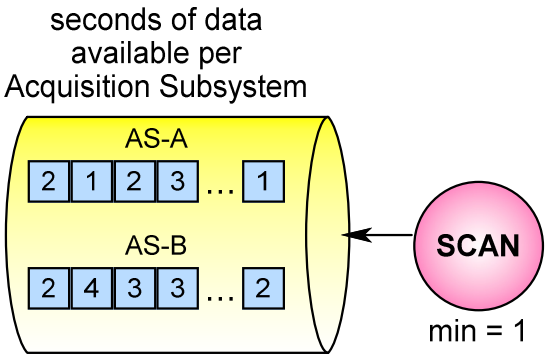

Fig. 8

In addition, the SCAN Process must not get too far behind the time-stamps of data in Acquisition Subsystems. This means that while the Scan Process tolerates lags in data delivery, it cannot wait indefinitely for data. Therefore, if a group of signals cannot be acquired, the Scan Process (after a configurable tolerance period, usually 6 seconds) registers data including the samples that could not be delivered. However, such group of samples is marked as bad data, through status values.

The integrity of values provided by this system is critical for monitoring the operation of the Power Plant. In case of an error during the acquisition process, it is preferable not to show values rather than showing incorrect values as if they were correct. For this reason, each sample has a status value assigned to it. The status value indicates whether the value is good or of some error in the acquisition has been detected.

It is important to clarify that the Scan Process deals with historical data which is different from real-time data. Real-time data is refreshed every second and it requires Acquisition Subsystems to deposit only 1 sample per second. In contrast, historical data can tolerate up to 6 seconds of lag, in order to be able to maintain the archive integrity even if one of the acquisition modules lags.

The Scan Process always seeks to be as close as possible to the system clock. As mentioned before, it has the ability to wait for lagging group of signals but also, it has the ability to acquire several seconds of information at a time. This means that if the Scan Process had to wait 4 seconds to get data, in the next cycle it will attempt to get 4 or 5 seconds of data to catch up with the system clock. The number of seconds of data that the Scan Process can register depends on whether the Acquisition Subsystems have data available.

This and most modules were designed taking into consideration that the NDAS was to be developed using a standard Linux kernel. Even though Linux provides high-resolution timers [6], the mechanisms that compose the NDAS are designed to tolerate certain lags through a coordinated and adaptive communication between processes, as well as an adequate use of intermediate buffers. This allows the Scan Process to tolerate lags and still keep delivering data. It is important to mention that buffer sizes are configurable to accommodate different operation scenarios [7]. This applies not only for the Scan Process but for most NDAS modules and has made it possible to develop the system without depending on a specific or special real-time operating system.

2.4.3 Storage considerations

The Master Scan archive was designed taking into consideration that large amounts of data would be generated; as a result, it is necessary to store data in the most efficient way possible. Moreover, since this archive is stored in the server memory, it is extremely important to ensure its efficient utilization.

Memory is a limited resource, at least when compared to the capacity of a disk. Furthermore, being a critical part of any system, it must be handled with caution. Memory misuse could have consequences that range from abnormal process termination to rendering a complete system offline. The SCAN data archive cannot be stored completely in memory because it usually amounts to several GB of memory. In order to store data for several hours, it is necessary to generate a disk SCAN archive. Therefore, the size in bytes of a SCAN archive in memory is limited by the server installed memory, the operating system limits and the memory other processes need.

Linux has turned out to be very flexible in allowing system limits to be modified. Those limits may be modified through kernel tuning. For example, the limit for a shared memory maximum size is usually small. In

order to allow the NDAS to create large shared memory areas (a typical memory SCAN archive is by the order of several Gigabytes), a kernel parameter known as `SHMMAX` can be modified thus allowing the operating system to be suited to the NDAS specific requirements.

The SCAN archive maximum size is configurable. The amount of memory assigned to the SCAN archive depends on several factors such as the number of variables to be stored and the type of values to be stored and the amount of installed memory. Normally, the main parameter taken into account is available memory. In order to allow enough memory for other processes and the operating system, the size of the memory SCAN archive is usually set to half the amount of installed memory in a server.

Variables in the NDAS are classified according to the kind of values they can store. Floating-point values such as temperature, pressure, etc. are called Analogical Variables. Variables that can take either one or zero as value are known as Digital Variables. This kind of variable is used to represent the state of elements like valves (open or closed), switches (on or off), etc. Discrete variables are very much like Digital ones except that they can store up to sixteen states. This kind of variable is used to indicate any element that can have more than two states such as handles or some types of valves.

Data in memory are arranged as an array of floats, which in most machines occupy 4 bytes [7]. However, 4 bytes are not required to represent a 1 or a 0, as in the case of digital values. Therefore, an efficient use of memory is made by organizing samples according to their type. In this way, a digital value is stored as a bit, a discrete value is stored as a byte and an analogical value is stored as a float [8].

The amount of memory a historical data archive will occupy is computed using the following equation (1).

$$T = \sum_{n}^{0} \left(\frac{f_m \times p_s}{d} \right) k + T_e \qquad (1)$$

Where: T is the size of the scan archive, n is the number of variables to store; f_m is the sampling frequency of a variable; p_s is the desired period of time for the archive in seconds; k is a constant that represents the size in bytes of a float, d is divisor applied according to the type of variable to store. For example, for digital values this d would be 32 since in a 4-byte float 32 bits can be stored.

T_e is the amount of memory used to store sample status values. Since status values rarely change, they are stored in a separate shared memory area in a double-linked list. To save memory space, instead of storing a status value for each sample, only changes in the status value are stored. Since too many status value changes is a rare case that can indicate an I/O card failure, the NDAS limits the maximum number of status changes that can be recorded within one minute. Taking into consideration this constraint, together with the number of variables and the Master SCAN archive duration, T_e is defined as follows:

$$T_e = (j \times n \times p_s) F \qquad (2)$$

Where: T_e is the size of the data status memory archive in bytes, n is the number of variables to store; p_s is the desired period of time for the archive in seconds; j is a constant that represents the maximum number of status values that can be stored per minute and F is a factor between 0.05 to 1.0 that determines the percentage of memory likely to be used. Since there is no way to predict how many status value changes will present, a worst-case scenario is computed (the amount of memory that would take to store all of the status values if they were changing all the time during the entire duration of the archive) and the factor F applied to determine the final memory size. This factor is configurable and is normally set to 0.10 (ten percent) or less.

Fig. 9 shows the SCAN shared memory area structure, which is divided in three sections: The header, variable control and information and the historical data section.

The header contains general information about the SCAN archive. For example, the number of variables recorded, the start and end dates, etc.

The variable information and control section contains information about each variable. For example, name of the variable, type of variable, data sampling frequency and memory indexes. This information is very important, because it is with the contents of this section that the SCAN process calculates the memory address range that pertains to each variable. This section of the archive is also accessed by extraction processes to perform calculations to locate the address for a variable at a certain date.

The historical data section is the biggest section which contains all of the stored data. This area is a succession of float values. In Fig. 9 memory ranges for each variable are represented by squares. That is to show that data for a given variable are stored continuously for the entire duration of the archive. Each square represents the data region for each variable. It is important to note that

not all of the squares have the same size because, although the duration of the archive is the same for every variable, the occupied memory space can vary depending on the type of variable (analogical, digital or discrete) and the sampling rate. Some variables can be stored with a 20 Hz sampling rate while other can be stored with a rate up to 1 KHz.

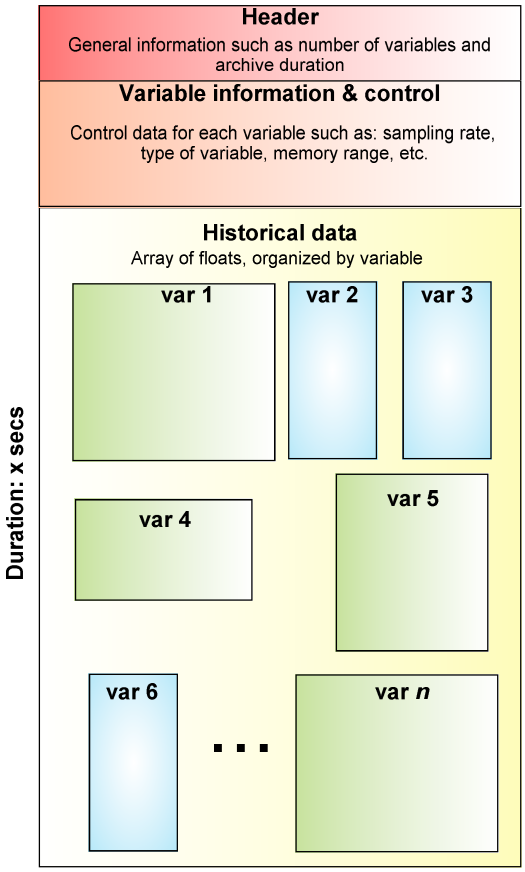

Fig. 9

Because available memory is often the limiting factor for the memory-stored Master SCAN archive, the NDAS allows the system administrator to determine how many GB of a server's memory can be used for the archive. With that parameter defined, the NDAS automatically computes the duration that the SCAN will have. This helps to have more control over the amount of memory used by the NDAS processes.

2.5 Alternative historical data archives

The Master Scan Archive is circular, which means that it registers data for a certain amount of time and when it gets full, oldest data are replaced with new data.

Since memory is a limited resource and data are recorded at a high frequency, this shared-memory archive may not last more than a couple of hours at the most. Operators need to have at least 12 hours of information. The Master SCAN archive aim is to provide the NDAS processes with high-resolution historical data. One of the most common types of processes that process Master SCAN data are alternate historical archive creation processes [9].

Fig. 10 shows the integration between different elements in the NDAS for alternate historical archives generation. The Central Acquisition Process and the Master SCAN processes are considered central processes since they are the main data processing programs within the NDAS architecture. Through these two processes, acquired data are organized and made available for the rest of the NDAS processes.

Memory management is another important part of alternative historical archive generation. Because the NDAS is a soft real-time system, the adequate use of buffers in crucial. Both the CAP and SCAN processes use internal buffers so they can adapt to different operation scenarios.

To generate alternate historical archives, other processes take data from the Master SCAN shared memory area, process the information and tore it on disk.

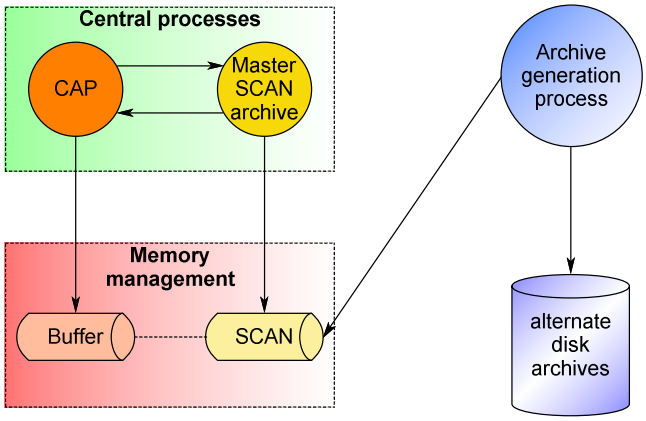

Fig. 10

One of the advantages of the NDAS regarding historical archive management is that there can be several active historical archives at the same time. By default, the system automatically writes to disc a 12-hour version of the memory-residing Master SCAN archive. However, operators can launch multiple customized archives as well. In this type of customized archives, operators can choose which signals to include in the archive, the sampling rate for each signal and the archive duration.

2.6 Data extraction

Data extraction is a very important part of the NDAS system. Storing thousands of samples would be useless if there wasn't a fast and reliable way to retrieve information based on the NDAS acquired data.

Data extraction from the SCAN historical archive is performed mainly by the processes which require historical data, be it to generate other historical archives on disk or for monitoring purposes.

Table 1 presents the average elapsed time for data extraction of samples of a variable stored in the Master SCAN archive, with different periods and sampling rates.

Table 1: Average elapsed time for data extraction

Period	Sampling rate (samples/s)	Elapsed time (s)	Extracted samples
1 MIN.	250	0.008	15,000
1 MIN.	1000	0.008	60,000
1 HR.	250	0.080	900,000
1 HR.	1000	0.139	3,600,000
3HRS.	250	0.176	2,700,000
3HRS.	1000	0.423	10,800,000

Table 1 shows some common query scenarios for data extraction. As shown in the table, response times are less than a second, which are adequate and provide the user with a responsive user experience.

Extracted data can be used in a variety of ways. For example, extracted data can be used to generate information displays.

Fig. 11 shows a typical display used by operators to assess the state of the plant through displays that provide information about the different power generation processes.

Fig. 11

As mentioned in the introduction, the NDAS coexists with the former Data Acquisition System which runs on VAX/VMS machines. One of the most significant parts of the former system still in use is the graphical interface that operators use to visualize reports. Although a new user interface has been designed for the NDAS [10], legacy reports are still widely used since official data display servers run on VAX.

Fast and reliable data extraction response times are very important because most times, data are to be transferred via TCP/IP sockets to the servers (as in the case of VAX display servers). However, it is important to mention that historical data can be extracted directly from NDAS servers.

2.7 Performance considerations

Since NDAS is a real-time system, during the development of any software module, it is not only important to consider functionality but good performance and efficient use of system resources. In a system that thousands of samples are acquired and processed, it is imperative to consider a process CPU time and memory usage.

Process efficiency is a problem that commonly arises during the development of real-time systems such as the NDAS. It is relatively simple to develop programs with an impeccable logic. However, at times these programs turn out to be impractical when executed, due to efficiency issues. The development of programs that execute rapidly through the efficient use of the CPU is an art in continuous evolution. In this case, data extraction for NDAS historical archives had to be developed to comply with efficiency criteria.

To accomplish the latter, during development the following actions were taken:

- Limit the use of string comparison operations. This kind of operation uses a lot of CPU time.

- Limit calls to methods or functions. Although NDAS is written in object-oriented C++, there are parts of the code where it was necessary to write the code directly instead of issuing a method call. Although modular programming makes programs easier to read and maintain, some critical parts of the code had to be reengineered to avoid excessive method/function calls. In these kinds of systems, performance prevails as one of the top considerations.

- Limit repetitive object instantiation. It is preferable to declare objects as attributes once rather than declaring them over and over in methods which might be called

frequently. Although this might seem obvious, in practice, it is common to have masked object declarations within structure definitions or from inherited classes.

- The use of profiling tools to identify problematic code areas. In the case of Linux, a good tool is `gprof` that generates statistics that reflect functions or methods that occupy the most CPU time as well as the parts of the code that are executed most frequently.

3 Results

The NDAS is currently installed in the Laguna Verde Nuclear Power Plant in Veracruz, Mexico. The system has been online for more than two years, supporting both Analogic and RTP DAQ modules.

It is also important to mention that several Acquisition Subsystems have already been developed and incorporated to the NDAS, allowing new signals coming from DAQ modules other than Analogic or RTP modules to be acquired and stored.

The NDAS currently works in parallel with the former DAS with good results. About 8000 signals are acquired from DAQ modules. NDAS flexibility has allowed Analogic DAQ modules to be gradually replaced by RTP modules without interfering with power plant monitoring systems.

The Master SCAN archive serves diverse processes, like the 12-hour disk SCAN archive generation process, other types of historical archive generation processes and several monitoring processes. High-speed data access allows a correct synchronization between the Central Acquisition Process, Acquisition Subsystems and the processes in general which rely on historical data.
High-speed data access has been tested during normal NDAS operation, since users are able to query the SCAN archive via reports and displays, with several historical archives being generated at the same time, without disrupting the timely synchronization between processes.

4 Conclusion

The NDAS is a flexible and powerful data acquisition system that can adapt to different operation scenarios. Because it was designed in a modular manner, further modules can be incorporated and the software is easy to maintain.

The NDAS will continue to support the data acquisition system replacement. However, the NDAS was designed not only to allow DAQ modules replacement, but to constitute a reliable and robust data acquisition system that will keep evolving and will be able to adapt to different operation circumstances. The NDAS flexible and modular design will ensure the former.

Acknowledgment:
The authors would like to thank MSI. Arturo Ramírez García from *Subgerencia de Ingeniería, Gerencia de Centrales Nucleoeléctricas* at *Comisión Federal de Electricidad,* for his support in the development of this project.

References:

[1] PARK, John, MACKAY Steve, *Practical Data Acquisition for Instrumentation and Control Systems*, Newnes, 2003, p. 4
[2] SUAREZ, Jurado, *Estrategia recomendada para el reemplazo del Sistema de Adquisición de Datos del SIIP, Instituto de Investigaciones Eléctricas*, 2004.
[3] BOVET, Daniel, CESATI, Marco, "Understanding the Linux Kernel", O'Reilly Media.
[4] GALLMEISTER, Bill, *Programming for the Real World POSIX.4*, O'Reilly & Associates Inc., pp. 110-112
[5] MÁRQUEZ, Francisco M., *UNIX Programación Avanzada*, Alfaomega Ra-Ma, España, 2004
[6] R. Gupta, *Linux 2.6 for Embedded Systems- Closing in on Real Time,* RTC Magazine, November 2003, online edition.
[7] LEAL, Ilse, SUÁREZ, J. María, *Registros históricos de tipo SCAN en memoria para un sistema de adquisición de datos en tiempo real para la Central Nucleoeléctrica de Laguna Verde*, VI Congreso Internacional en Innovación y Desarrollo Tecnológico, México, 2008.
[8] J.M. Suárez, "Diseño de Base de Datos General para Procesamiento Histórico de Señales", México (1997).
[9] LEAL, Ilse, SUÁREZ, J. María, *Registros históricos DELTA en memoria para un sistema de adquisición de datos en tiempo real para la Central Nucleoeléctrica de Laguna Verde*, VI Congreso Internacional en Innovación y Desarrollo Tecnológico, México, 2009. ISBN: 978-607-95255-1-4.
[10] VERGARA, Alfonso, *Diseño de la Interfaz Gráfica del SIIP*, internal document IIE/GSP/12979/103/2008/F, 2008.