Development of an Innovative Storage Manager for a Distributed Control System

M. Mara INFN-AC, Frascati, Italy
C. Bisegni, G. Di Pirro, L.G. Foggetta, G. Mazzitelli, A. Stecchi, INFN-LNF, Frascati, Italy
L. Catani, INFN-Roma2, Rome, Italy

ABSTRACT
ICAOS is an INFN project aimed at the definition of a new control system standard for large experimental apparatus and particle accelerators based on innovative communication framework and control services concepts. ICHAOS has been developed to address the challenging requirements in terms of data throughput of the new accelerators under study at INFN. One of the main components of the ICHAOS framework is the historical engine (HST Engine), a cloud-like environment optimized for the fast storage of large amount of data produced by the control system’s devices and services (I/O channels, alerts, commands, events, etc.), each with its own storage and aging rule. The HST subsystem is designed to be highly customizable, such to adapt to any desirable data storage technologies, database architecture, or indexing strategy and fully scalable in each part. The architecture of HST Engine and the results of preliminary tests for the evaluation of its performance are presented in this paper.

THE !CHAOS FRAMEWORK
The ICHAOS framework has been designed after an in-depth evaluation of the new software technologies for data transfer and data storage emerging from the development of high-performance Internet services, such as the non-relational databases (NDB) and the distributed caching system (DCS). Both are designed for a high degree of horizontal scaling that allows the insertion and retrieval of data at the highest possible throughput, limited only by the saturation of either the available bandwidth or the network connections of the subsystem. While the NDB logsics and techniques are used to implement the indexes management and the data fast retrieval the DCS is used to provide the “live data sharing”, a scalable service for sharing the real-time device data. This software provides in-memory key/value storage and permits fast accesses to the same key/value by many concurrent clients. This caching layer avoids overloading the front-end controller with multiple reading accesses from clients that need to fetch data of a device. These two software technologies represent the core components in the design of the new control system named ICHAOS [1, 2, 3].

THE !CHAOS STORAGE ENGINE
In ICHAOS the data storage is provided by the service called History (HST Engine). This conceptual design will allow an innovative storage system for a Distributed Control System, giving ICHAOS an important technology advantage against other equivalent most popular standard for controls. The main ideas at the base of the data acquisition process are the following: a distributed file system is used to store data produced by machine operations while a KVDB manages the indexes structure (nowadays candidates are Hadoop [7] and MongoDB [8] respectively). These tools have been chosen to avoid the diffusion in the scientific community for solving similar problems and the abundance of use cases to which learn from. The functionalities of the ICHAOS HST Engine are allocated to three dedicated components, or nodes, namely the ICHAOS Query Language (CQL) Proxy, the Indexer and the Storage Manager.

The Data Flow
By checking the medium data-pack size of the devices, and its pushing rate, the registered CQL Proxies are “allocated” to the devices in order to load balance the network infrastructure and the total computing power.

For each CQL Proxy a logical path is created in the distributed FS. To improve the performance of the system each proxy can allocate a pool of threads with the only task of getting the packets received by the proxy and start the allocation inside the file system.

After the “Moving” phase (Figure 3), every chunk of the logical file will be ordered in time (every data packet has a timestamp >= than the previous one). Anyway two or more LF chunks can be overlapped in time (a side effect of the fast moving phase).

TEST RESULTS
The components described so far, related to the staging and moving mechanism, have been tested off-line by using a software simulation of these two phases. The fusion phase has been removed from the numeric tests because is not fundamental for the data acquisition process: it is used once the data is already safe on the file system. The tests have been run on a mid level Mac Pro with two 2,6GHz Quad-Core Intel Xeon, 16GB of DDR2 RAM, and a SATA T2 SSD hard disk. The graph shown here (fig. 4) is obtained by using two Producer processes simulating ten devices running with 50 threads each, and a single consumer process running on five threads. The average data produced by the simulated devices is 3,5 MB/s simulating 515 channels pushing data packets of 68 B at 100Hz. The test environment is like a worst-case scenario for this algorithm, because it cannot gain performance by a distributed file system and a multitude of proxy machines. In fact the data rates obtained can grow almost linearly by increasing the number of proxy machines and using a more appropriate file system. The graph in figure 4 shows the three fundamentals measurement in the caching system: the data produced by the devices (in red), the data actually in the cache files (in blue) and the data actually stored inside the device logical files (in green). More intensive tests will be run in the next months on the other parts of the storage system now under development.

REFERENCES

http://chaos.infn.it

Common Integration System: http://cvs.infn.it:8080
Institutional Git Repository (Public Read Access) https://cvs.infn.it:8443/chaosframework.git
Public Git Repositories: https://chaosframework.atlassian.net

Take Part!!