i-DHSM

Dynamic Hierarchical Storage Manager

Media Management for audiovisual digital Archiving

By (*) A. D’Alessio, A. Bertini, F. Ciferri, G. Ferrari, M. Strambini

Description of an integrated solution for the management and migration of heterogeneous data (HSM - Hierarchical Storage Management), by presenting an innovative software/hardware architecture, specifically developed for the digital archiving of audiovisual mass content; i-DHSM – Dynamic Hierarchical Storage Management - is a highly scalable and configurable solution for a wide range of applications, as it provides full performances on both SMP (Symmetric Multiprocessor) and Cluster systems.

Easily managing audiovisual files and moving clips over high-speed connections to high-performance devices is what today Broadcasters, DOT com companies and TLC operators most recommend and request whenever they plan to set up a digital archive. Audio/video clips for large digital storage nevertheless require specific management of ingestion and migration procedures when moving files from server to backup devices (i.e. automated tape libraries, etc.) and vice versa, no matter which quality or application is required (Media Asset Management and Archive Management, for example).

HSM systems (Hierarchical Storage Manager)

Any HSM system (Hierarchical Storage Manager) - i.e. systems engineered to provide integrated classification of audiovisual clips and automated migration of files from server to backup devices and vice versa – has a pretty limited space on local hard disk for content storage if compared to the space available on dedicated devices for backup operations; for this reason, HSM software application automatically moves less used files on tapes and keeps most requested clips locally, thus saving space and time.

As HSM applications are usually very I/O demanding, the server machine has a very powerful bus on board and is connected to a local RAID array, which shall be able to respond to the requested throughput. Thus typically an SMP (Symmetric Multi Processing) machine will be used as server, while SAN solution take care of fast connection required for the I/O operations.

Using this configuration as starting point, we will try to investigate new solutions regarding both the software that the architecture of HSM systems.

From here we will use the term “DFS Manager” (Data File System Manager) referring to a particular device (or logical block) dedicated to the data ingestion/retrieving inside an HSM system. The term “DFS Storage” (Data File System Storage) will be used referred
to the device (or logical block) that takes care of the data storage. Following we show a simple diagram of a traditional HSM system.

Cluster approach

We want to start analyzing a new approach from the architectural point of view. In the last years many enforces were done to move from the expensive SMP system to more flexible and cheap approach. The clusters seems to be the right choice, also with some limitations, we can say that they can satisfy all the requirements that big SMP systems resolves until today.

Typically, in a last generation cluster architecture, are used some machines as nodes and one or more front-end unit(s). Nodes and front-end unit(s) exchange data on a private network – which is usually a high-speed connection – while the front-end only is committed to exchange data with external world.

Generally, each node, realized with pc class machine, has a local storage memory, which can be shared among all nodes connected. A cluster might also be composed of heterogeneous nodes, for example by dedicating $N$ nodes to intensive operations and $M$ nodes to simple processing. In the HSM application this will be recommended whenever there are $N$ data flows which require non specific treatment prior to migration and $M$ data flows (i.e. pictures from satellite) which instead require intensive CPU processing prior to migration.

We have to underline the intrinsic fail-safe capacity of the cluster architecture, in fact every time that a node fails, the work of the failed node could be redirected to another working node inside the system. Furthermore we can build a system that contains more nodes of which we need for the
normal operation, in this way we obtain a redundancy that grows linearly with the number of redundant nodes that we add in the system. It could be simple to add more power to the system without interrupting the system’s work because we can reconfigure “on the way” the cluster to use the new added nodes.

So these are all the advantages of the cluster toward the SMP system, we could summarize these in:

- **Fail-Safe architecture.**
- **Progressive redundancy.**
- **Simply expandable and scalable.**
- **Investments saving.**
- **Dynamically reconfigurable.**
- **Accomplishes heterogeneous tasks.**

At this point we can imagine to transform the architecture of the (Fig.1) applying the rules of a cluster system, in this case we obtain a diagram like what follows in (Fig. 2)

In the simplified diagram of Fig.2 we can see that every node is attached to a local storage that we could use as a buffer for I/O operations, in this way we can guarantee also a redundancy over the disk, because they are a part of the cluster nodes management.

During the normal workflow we can assign, every time it’s needed, a node to a well determined couple composed by a DFS I/O module and a DFS storage module freely selected from those connected to the SAN switch; the data that come from/to the DFS I/O are processed by the...
assigned node, then are passed/retrieved to the DFS storage or saved, in the meantime, on to the local node’s disks. In this way these disks could substitute the RAID that we found in the SMP approach (see Fig. 1).

The limitations of this approach could resides in the I/O capacity of the single node, infact, as we indicate, generally a pc class machine it’s used as nodes. In the typical use of a cluster (i.e. computational intensive) the single cpu power of a pc is comparable to that of the SMP machine’s, otherwise the bus of a pc machine it’s absolutely not comparable with the SMP one’s. For the pc we consider a maximum throughput of 150 MB/sec as good result for the typical bus, so we have to pay attention to this parameter and remember that a node, with intensive I/O use, must support a connection to the cluster (private network), a connection to the disk and to the fiber channel that connect it to the SAN switches.

**i-DHSM**

For satisfying both new architectural schemes and more complex user requests we try to describe a software solution that rise from traditional HSM systems to latest cluster configurations. The product that follow it’s named “i-DHSM” (i- Dynamic Hierarchical Storage Manager) and is a software toolset fully implemented and engineered for an integrated media asset management.

The base idea beyond the product was to build a software that could be scalable also on different operating systems, so one way to achieve this purpose was to build a simple and, possible, little kernel with all the base functions implemented inside it. The kernel use a custom protocol to exchanges data outside, in this way it’s possible to implement all the high level functions in different modules and uses this protocol as a way to comunicate with the kernel and every modules that needs to became a part of the application.

Another advantage of this approach it’s on the capacity to analyze or change a specific data that normally resides in memory, but in this case becomes avaliable to every external module also if it’s not necessary included in the main application. This feature simplify, for example, the construction of external monitoring programs as it’s becomes really simple to access the application data over a distributed network.

Both in the SMP approach that in the cluster one’s, it’s important to underline the needs of high parallelism, for this sake the kernel takes care of syncronizing the access to internal/external data. So every module that it’s attached to the kernel take the capacity to be run concurrently with the other present in the application. In this way we can easily swap the construction of a module written for different hardware architectures.

Following the same rules it’s possible to write software drivers dedicated to interface specific devices with the system, this ability it’s fundamental because of the high number of different devices that are typically used in HSM systems.

Besides, the scalability of the architecture allows an easy integration in the system of particular modules for the processing of data: for example a specific application tool has been developed for the fast access to a portion of a clip requested by the user. In case of content stored in Mpeg-2 format, for instance, compressed data are processed and the selected sequence is extracted preserving the consistency and the correctness of data and in a transparent way for the user.

We want to spend also one word over a particular module that manages all the data flow of the application and translate it into database format. RTDBE (Real Time DataBase Engine) it’s builded to be linked at the same time to all the modules composing the application, in this way we can maintain the data avaliable even if a fault occurs. We can achieve this goal mantaining the access to the data with a very low delay using direct access to the device that contains the data (typically a system disk) and an optimized structure based on dynamic page (a page contains a specific piece of data group) reallocation algorithm.
A simple feature that we added to the RTDBE module, it’s the capacity to import/export data from/to the application and external database format (STDBE), using this module becomes simple to interface the entire application with the already exists user data environment.

We illustrate in Fig. 3 the block diagram of the main i-DHSM module’s structure and linking, where we have grouped some modules together regarding their functions.

(*) SHS Multimedia S.p.A. Italy
The module, indicated as “Storage Content Engine”, is the kernel module and it watches over the whole i-DHSM flow, by controlling:

- **DeviceManager** modules and relevant drivers for storage/backup devices. This module interfaces all devices connected to an i-DHSM system and controls the Device drivers developed according to specifications allocated by each device.
- **DFS Manager** modules and relevant drivers for ingestion/retrieval devices. Monitors system data during I/O procedures, such as file consistency and hierarchical structure; if requested, data can be accessed and changed according to specific algorithms.
- **DAM** (Distributed Access Module) This module is used for data sharing over distributed geographical area, such as by connecting two separate i-DHSM systems, each one having its specific HW configuration and installed in different sites.
- **QueryManager** module for the management of incoming queries.
- **API** (Application Programming Interface). Main functionalities of both system core and modules are available with API. Users can access and change any system event occurring during daily operational flow.
- **WatchDog** module which monitors the status of each process and, in case of fatal errors due to hardware failure, adjusts/changes settings, in order to avoid system shutdown.
- **Disk Manager**, controls the management of Data Files on local storage and preserves the system from any overloading on disks, by runtime analyzing statistical data, such as data access, profile of users, latency, etc, previously sampled.

---

**Fig. 3 i-DHSM block**
FailSafe Manager module monitors the status of all elements and devices; in case of failure, this module re-allocates resources over the whole system, thus preventing from any overloading.

One example of real i-DHSM installation is the project “TecaFast” realized for RAI (Radio Televisione Italiana) installed in Rome (Italy). The goal of this project is to build a high quality audio/video digital archive of all the material that RAI produced in almost 45 years (~400,000 hours) plus the actual transmissions. All the multimedia materials, Mpeg2 4:2:2 files digitized at 10 Mbit/sec, must be accessed from different locations connected together by fiber channel network. Every location access to the data using some encoding/decoding machines as DFS I/O devices and pc workstation for database information browsing. The main system is composed by 2 SMP server each with 8 cpu and 2 GB of memory connected to a 2.5 TB raid disk. The DFS Storage is composed by some automated library each containing 6000 cartridges and 10 tape drives. Actually there are 40 client workstations distributed over the city, in the main RAI offices, that can access the data in the same time. In the near future they becomes 100 and could be distributed over the entire country, in this case far location could be accessible by satellite link or by little regional installations.

Conclusion

The i-DHSM’s modules are built on the base of today’s HSM user requirements, but in our experience we could see that different customer’s needs bring us to add more modules to the main application. For this work we use the i-DHSM API itself, cause this feature is been able to introduce new functions at different levels, starting from the high level interface down to the kernel. In this way the system becomes more complex but maintains a good flexibility, for this reason we think that our architectural choice could be one of the possible solution to HSM system approach.