RJ 10337 (A0501-006) January 13, 2005
Computer Science

IBM Research Report

SANFS Maestro: A SAN File System Planner

Aameek Singh, Kaladhar Voruganti, Sandeep Gopisetty, Aki Fleshler*, Ramani Routray, Chung-hao Tan
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

*IBM Beaverton

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

SANFS Maestro: A SAN File System Planner

Aameek Singh* Kaladhar Voruganti‡ Sandeep Gopisetty‡ Aki Fleshler§
Ramani Routray‡ Chung-hao Tan‡

Abstract

Manual planning of storage infrastructures that are large and that utilize heterogeneous devices has become a time consuming and error prone process. This problem becomes even more complex when one is designing storage area network file system (SAN FS) based storage solutions. SAN FS systems that combine the benefits of both SAN and network attached (NAS) systems have been proposed as a mechanism for designing scalable storage infrastructures. This paper proposes a SAN FS capacity planning tool that takes application level requirements, best practices, and other types of policy input to design the necessary SAN FS logical and physical constructs. The planning tool can be used to design both new SAN FS deployments or extend existing SAN FS deployments. By automating the design process, this tool helps to cut down on the amount of time and the number of designers required to design large scale storage infrastructures. Finally, there currently do not exist any planning tools for SAN file systems, and thus, this is the first paper in this regard.

1 Introduction

The storage needs of most organizations are growing at a rapid pace driving them to scalable storage infrastructure solutions. This, in turn, is prompting many organizations to move away from direct attached storage (DAS) solutions towards storage area network (SAN) and network attached storage (NAS) solutions. Organizations are making this move primarily due to a) the distance and connectivity limitations of direct-attached storage transport protocols such as parallel SCSI b) the tight coupling between servers and storage (i.e. one needs to purchase more servers in order to add more storage, even if there is no need for the extra computation power) c) ease of management and sharing.

However, in order to make the transition from DAS environments to NAS/SAN environments one has to deal with the following issues:

1. **NAS versus SAN Debate**: Both NAS and SAN solutions have their respective strengths and weaknesses, and so most organizations need to perform a detailed analysis to choose the solution that best satisfies their needs.
2. **Design Complexity**: Manual design of scalable, application aware and heterogeneous solutions is a long and in many cases error-prone process. Most organizations are finding it difficult to design a scalable NAS or SAN solution due to the following reasons:
   - They need to consider the application requirements with respect to performance, availability, disaster recovery, security, and future growth for numerous existing applications and new to-be-deployed applications.
   - They have to utilize the devices present in their existing DAS environments and integrate them in the newly proposed NAS or SAN solutions. This, in turn, can potentially raise numerous device interoperability issues.
   - They need to ensure that they adhere to the numerous known best design practices to have an optimum design.
   - They have to examine numerous design choices to come up with an optimized design plan. For example, the output design plans can be optimized based on cost, or physical space or power utilization etc.
3. **Organization Structure**: Even when migrating from DAS solutions to NAS/SAN solutions, some organizations fundamentally still maintain the notion of information technology (IT) department islands. That is, each large department performs a separate transition from their current DAS environment to a new NAS/SAN environment. Thus, in essence, the separate departments have their own private NAS/SANs, and hence, do not fully realize the benefits of storage consolidation.

SAN file systems (FS) that combine both NAS and SAN technologies have been proposed in order to realize the combined benefits of both these technologies [5, 8, 7, 4]. Hence, SAN FS technology is trying to address the SAN versus NAS debate. It is important to note that the focus of this paper is not on evaluating SAN FS technology, but instead is on developing a SAN FS capacity planning tool. Briefly, SAN FS, like NAS systems, provides a single file namespace across multiple hosts and, like SANs, it allows hosts to directly obtain storage from the storage controllers.

Many storage management vendors are introducing storage infrastructure capacity planning tools to specifically address the second problem. The storage capacity planning tools, are for designing SANs, and there do not exist any tools for designing SAN FS solutions. Finally, it is necessary for organizations to restructure their IT departments in order to address the third problem. Organizations are beginning to realize the importance of their organization structure and its impact on the overall utilization of their IT resources.

*Work done while visiting IBM Almaden. Current affiliation - Georgia Tech, aameek@cc.gatech.edu

‡IBM Almaden Research Center, sandeep@almaden.ibm.com, {kaladhar, routrayr, chungtan}@us.ibm.com

§IBM Beaverton, akif@us.ibm.com

This paper specifically addresses the problem of how to automate the design of storage solutions that utilize SAN file systems. Hence, it is addressing a combination of the first two problems listed above. We have implemented the SAN FS capacity planning tool, and this paper is describing the architecture details and the unique design aspects of our implementation. Some novel aspects of our design are:

- The introduction of the necessary planning constructs to allow for the layering of the SAN FS planner on top of a SAN planner. The existing SAN planning tools do not have the necessary notions to facilitate this layering.
- The introduction of the notion of application level templates, and the mapping of these templates to logical SAN FS constructs, physical SAN FS constructs, and the underlying SAN constructs. Existing SAN planning tools do not have the notion of application templates.
- The use of policy engine to guide and validate the SAN FS deployment plans.

It is important to note that the goal of this tool is not to replace human designers, but instead to aid them with the design process. Thus, the generated output plans can be modified by the system designers.

2 Background Information

Many of the performance and cost related differences between NAS and SAN solutions are disappearing due to the following reasons:

- With the emergence of IP SANs the cost disadvantage of SANs is not an issue.
- With the emergence of TCP offload engines, the performance disadvantage of NAS solutions are less of an issue.
- With the emergence of NFSv4, many of the network storage protocol issues related to the presence of multiple sub-protocols, lack of maintenance of state, and the absence of compound operations that coalesce multiple commands have been resolved.

However, the routing of NAS requests via an intermediate NAS server versus the direct access of storage by hosts in SANs is still considered as a key disadvantage of NAS. Similarly, the ability of multiple hosts to share a file system name space is still considered an advantage of NAS in comparison to hosts running a local file system, and using a SAN protocol to communicate with the storage controllers. Hence, the notion of SAN FS has been proposed to realize the benefits of both NAS and SAN approaches .

In SAN file systems, like in SANs, the hosts directly access storage from multiple storage controllers using block protocol access. Moreover, like in a NAS system, multiple hosts still share a single file system namespace. In SAN file systems, the hosts use a network file system protocol to retrieve and cache file meta data from a file system meta data server. Subsequently the hosts directly access storage using a block level protocol from the storage controllers.

The files in the file name space are grouped into file sets. Each file set is managed by a primary metadata server. The notion of filesets is also useful for making file level replication operations. The files in the file set obtain their storage from storage pools. Different files in a file set can obtain their storage from different storage pools. Each storage pool can be thought of as contiguous logical piece of storage with certain performance, availability, security etc properties. For replication purposes, multiple file sets can be placed in a replication consistency group. A replication consistency group preserves the write operation order across file sets (useful in database environments with write-ahead logging feature).

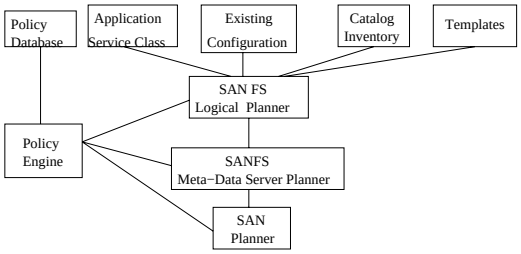

Figure 1: SAN FS planner architecture

Currently, not much work has been done with respect to SAN FS planning tools. However, a lot of work has been done with respect to SAN planning tools. SAN Architect from EMC [1], SAN Designer from Computer Associates[6], Appia [9] and Ergastulum [3] research prototypes from HP are all trying to automate the SAN design process. The SAN FS planning tool being proposed in this paper performs SAN FS specific planning tasks, and it delegates the SAN planning activities to an underlying SAN planning tool. This paper also proposes how the interfaces to the existing SAN planning tools need to be modified in order to build a SAN FS planner on top of them.

3 SAN File System Planner Architecture

This section describes the SAN FS planning architecture illustrated in Figure 1. In summary, the SAN FS planner gets various types of requirements as its input, it performs the SAN FS construct specific planning, and it then passes on control to the SAN planner to perform SAN related planning. This section is organized into three subsections describing a) planner input b) planner output c) planner engine details.

3.1 Planner Input

The SAN FS planner gets the following types of input:

- **Application Service Class Templates:** These predefined customized templates allow users to select the desired level of service for an application type. For example, *Gold* class of service for an application implies a certain level of performance, availability, security, and future growth for that application. The application template also indicates the number of data containers and their corresponding service class, that are associated with a particular application template. Data container is a storage construct that corresponds to contiguous logical storage of a particular type. For example, a database application could have a log data container, a user table space data container, and a data container for temporary data. Thus, *Gold* level OLTP DB2 application

could have *Fast* level log data container, *Express* level user table space data container and *Average* level data container for temporary data. There are concrete set of attributes associated with performance, availability, security, future growth, and backup/recovery. The details of the attributes will be provided in the long version of the paper.

- **Catalog and Device Inventory:** Every organization has a preferred list of hardware/software (e.g. switches, storage arrays, HBAs etc) that they want to use. In addition, there can be a inventory of new and old devices that need to be used as part of the design process. Both the catalog and the inventory databases need to be kept up to date in order to be useful.
- **Policy Specification:** The term *policy* is an over-loaded term, and it can be used to refer to best practices, interoperability constraints, scheduling information, security information etc. The different types of policies are specified as a tuple with the following fields: 1) if condition 2) then clause 3) scope of the policy 4) priority of the policy (to resolve policy conflicts). The policy information is stored in XML format in a local policy repository, and this local repository can be periodically updated from a centralized policy store.
- **Discovered Resources:** In cases where one is not designing new storage infrastructures but instead is extending existing infrastructures one needs to discover the existing resources, and also be aware of how the resources are getting utilized. For example, one discovers that the existing infrastructure has three storage arrays from company X that capacity wise 50 percent utilized, and only two of these arrays have the ability to support iSCSI SANs.
- **Optimization Criteria:** System designers can use different types of optimization criteria during the design process. For example, one could want a SAN design that uses the least amount of physical space, or power or money. System designers can also specify the number of alternative plans that they would like to be generated by the planner.
- **Physical and Middleware Templates:** System designers can decide to use replication boxes, virtualization boxes, security boxes etc as part of their design process. In this paper, these different types of boxes are referred to as middleware templates. Physical templates refer to a combination of resources that are known to successfully interoperate with each other. For example, HBA from vendor A, switch from vendor B, and a storage array from vendor C are known to successfully interoperate with each other.

3.2 Planner Output

The planner generates an output data structure that can be thought of as a graph of logical and physical constructs. For example, the graph nodes correspond to physical constructs such as switches, storage arrays, hosts, HBAs, disks, etc and logical constructs such as volumes, zones, filesets, consistency groups etc. Moreover, the graph edges correspond to fabric interconnectivity, and the device container relationships. For example, HBA x belongs to Host y, or volume x has been created out of storage pool y, which resides on storage array z. We have taken care to ensure that the graph is SNIA SMI-S standard compliant for those notions which currently are present in the SMI-S model. The base SMI-S model has been extended to accommodate SAN FS notions. The output plan graph can be used to provide topology viewing capability, generate bill of materials, generate storage provisioning workflows, or generate SAN FS installation scripts.

3.3 Planner Engine

We have divided the entire SAN FS planning process into three phases. In phase 1, the logical SAN FS construct planning related to filesets, consistency groups, storage pools is performed. In phase 2, the physical SAN FS metadata server related planning is performed. Finally, in phase 3, the physical SAN related planning is performed.

3.3.1 Phase-1: Logical Planning

In this phase, we determine the file system consistency groups, filesets and storage pools. This phase is the most interactive phase of the planner. The templates based design is augmented with intermediate steps to allow modifications to the planner recommendations. The flow of this phase is as follows:

1. The user describes **locations** at which the file system instance is to be deployed - this facilitates multi-site planning*.

2. The user describes the **application workload**. As mentioned earlier, this is achieved through application templates, embedded within the planner. In addition, we use the *application service class* attributes, which define the criticality of the application and thus help in determining the recommendations for logical constructs. For example, for a mailserver application with *platinum* service class, the planner advocates using two consistency groups - one for the mail data and another for attachments, with 4 filesets, containing various user mail files and *FAST* storage pool for mail data and *Express* for attachments and the *FAST* storage pool for metadata storage. These templates are obtained by running a number of benchmarks and are pre-packaged with the planner.
It is important to describe the features that "define" a certain workload. For a SAN FS, we look at operations that generate metadata transactions. For example, the fraction of *read*, *lookup*, *write*, *commit*, *map/unmap*, *dirwalk* etc. operations will determine the amount of communication between hosts and metadata servers. This will impact the number of metadata servers and memory requirements at metadata servers.
While a certain amount of knowledge can be determined from the template, the "size" of the workload will differ from deployment to deployment. Thus, we require certain inputs from users that characterize this metric. For example, application data capacity, the working set size - the number of objects being accessed simultaneously, thus requiring state information to be maintained at the metadata servers; estimates for backups - to determine the amount to extra storage to be provisioned.
For a completely customized application, we require the user to specify more information in order to determine the meta-

*Current prototype implementation only does a single-site planning

data workloads. We plan to automate this process using a monitoring engine, as part of our future work. Currently, we do include the automation of determining appropriate filesets, given the application data. For example, a user can only specify the data and let the planner figure out filesets from that data. There are a number of benefits for automating this process:

- Filesets are the granularity of metadata workloads. A wrong fileset design can significantly impact system performance.
- Fileset creation can be a tedious process, requiring many different parameters to be filled like quota, quota-type etc. An automated planner, integrated with the best-practices policy engine can significantly reduce user involvement in the process.

In our prototype implementation, we use the fileset quota size as the limiting factor. Based on the application future growth requirements, captured within service class attributes, we design filesets occupying appropriate fractions of the quota. This can be easily configured to operate on the number of objects (in a fileset) limits as well. Also, the design of the filesets is validated against best practices and other defined policies - for example, a usual good practice is to avoid nested filesets of depth more than three.
Note that at every intermediate stage of this step, the user can modify the recommendations by deleting certain consistency groups, moving contents of one group to another, modifying filesets and storage pools. This fits the desired goal of the planner to *aid* a human operator instead of replacing one.

3. Finally, the user describes its client **hosts workload**, which details the number and kind of hosts running various applications. This step is separated from the application workloads step, in order to allow a more generic workloads description, in which certain hosts can run more than one applications. This hosts information is used to determine the metadata workloads (objects simultaneously accessed by different hosts require distributed locking state to be maintained at the metadata server) and zoning/LUN masking requirements (hosts of one applications are zoned together and LUN masking is also done to prevent hosts from accessing unauthorized storage).

3.3.2 Phase-2: Metadata Server Planning

Based on various Phase-1 inputs, Phase-2 determines all metadata server related information. Specifically:

- *Size of Metadata Server Cluster*: Based on application workload, we first determine the size of the metadata servers. We first calculate the rate of file operations for that workload - based on the selection of a template[†]. It depends on both the kind of workload and the "size" of the workload. Next, we calculate the rate of metadata operations for the workload. For each application workload, the template contains the mapping of the rate of file operations to metadata operations. This mapping is based on the caching ability of metadata at clients. For example, for a read-majority workload, subsequent read operations will not result into metadata operations due to caching at the client. Thus, the template mapping takes this into account. Finally, the rate of metadata operations divided by the transaction rate of the metadata servers (available from device catalogs) gives us the required number of metadata servers.
- *Filesets to Metadata Servers Mapping*: In this phase, we also distribute filesets amongst metadata servers. In SAN FS, metadata servers are responsible for holding metadata information and other state information like locking for data in the filesets assigned to them. The planner can distribute the filesets based on general best practices (retrieved from the best-practices policy engine) - eg round robin in a homogeneous metadata cluster, or appropriate load sharing for a heterogeneous one.
- *Metadata Server Memory Configuration*: There are two main components to the memory utilization at metadata servers - caching disk structures and caching distributed state of memory. For IBM SAN FS, assuming an average file name length to be 15 ASCII characters, it typically takes (560+3*15=) 605 bytes to maintain the disk structure for each object. In addition, to maintain the distributed memory state, it requires 676 bytes for a single non-shared object. This distributed state is multiplied by the number of users sharing the object due to distributed lock state information that needs to be maintained. These numbers are multiplied by the number of objects in the file system to obtain total memory requirements. Finally, a constant factor is added to each of these numbers for the file system requirements, maintenance operations and other requirements like CIM clients. These constants are different for master and subordinate metadata servers.
- *Metadata Storage Planning*: We also need to plan for extra storage required to maintain the system pool of file metadata. Empirically it has been found that 5% of the total current data capacity (including metadata) suffices for the metadata storage in a SAN File System (as opposed to 3% in a local file system). This also takes into consideration the need for fileset level point-in-time copies (eg. IBM SAN FS FlashCopy).

3.3.3 Phase-3: Physical Planning

In this phase, the SAN FS planner leverages underlying SAN planning tools. There are existing tools which can design the complete fabric - hosts, HBAs, switches, storage arrays etc. Thus the primary task of the SAN planner drops down to mapping SAN FS requirements to the SAN planner tools inputs. This is achieved in the following ways:

- Based on storage pool requirements, the SAN planning tools directly provision appropriate storage. The metadata storage is also provisioned in this manner. In our implementation, as shown in Figure-2, the notion of SAN FS storage pool maps to the notion of a SAN planner data container. The SAN planner data container, in turn, gets mapped to storage controller pool(s). The notion of a logical flow is associated with each data container and it captures the data flow requirements of

[†]For a customized application, its operations are mapped to the closest template available. In future, the monitoring engine can evaluate it precisely

each data container. The notion of storage service class (that captures performance, availability, security, future growth requirements) is associated with each data container. Each logical flow, in turn, can get mapped to multiple flow requirements, and multiple flow requirements get mapped to a physical flow.

- The metadata servers are treated as regular hosts, SMI-S ComputerSystem, with *dedicated* field set to "MetadataServer". This allows for SAN planning tools to appropriately configure HBAs for the metadata servers.
- A new application is created with metadata servers as hosts and metadata storage as its underlying storage. This allows for application based zoning (done by SAN planners), thus zoning out all other hosts from the metadata storage, preventing unauthorized access to metadata storage by any client.
- For the IP based connectivity of hosts to metadata servers, existing networking tools are used to obtain full connectivity. The Fibre Channel (or IP) connectivity between hosts and storage arrays is directly obtained using the SAN planner. The remaining metadata server to storage connectivity is obtained by adding metadata servers to all applications as hosts, thus directing underlying SAN planners to connect and zone/LUN mask the metadata servers to all storage.

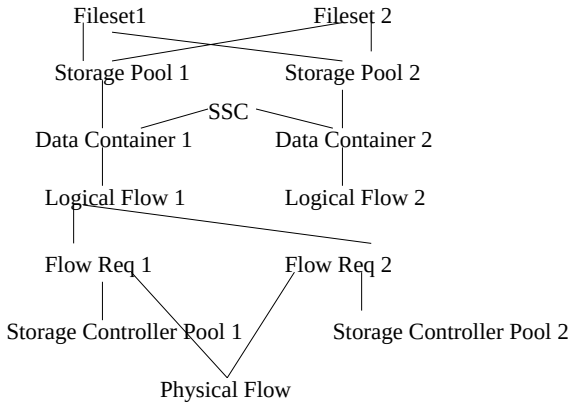

Figure 2: Mapping SAN FS planner to Block Planner

3.4 Policy Processing

System designers can specify policies that are relevant for each of the above planning phases. For example, they can specify policies to describe files to storage pools mapping, to enforce compatible metadata server hardware and software configuration, to decide which filesets and storage pools belong to a consistency group, to ensure that the proper multi-pathing software exists at the appropriate hosts etc. The detailed listing of the different types of SAN FS policies will be provided in the detailed version of the paper. There are numerous known SAN design policies in addition to SAN FS layer specific policies [2]. The capturing of policy semantics in the design process can be performed either as a validation mechanism post the design process, or as a guidance mechanism during the design process [10]. That is, most policy types can be specified and tested either in guidance mode or in the validation mode. In the guidance mode, at various points in the planner implementation, calls are made to the policy repository to retrieve the necessary guidance inputs and the planner then tries to ensure that those policies are satisfied. In the validation mode, the planner sends the generated plan to the policy engine, and the engine, after retrieving the relevant policies from the policy repository, checks to see whether any of the pre-defined policies have been violated. If the plan violates any of the specified policies, then currently in our implementation, we log the error and expect the designer to rectify the input and re-plan. In future we want to establish a feedback loop to automate this process. Currently, for a specific implemented version of the planner, all known policy types are specified and tested in the guidance mode in order to ensure that the plans do not violate policies. All new policy types (post a specific planner implementation) are implemented in the validation mode.

4 Conclusion and Future Work

Manual planning of large scale and heterogeneous storage infrastructures has become a time consuming and error-prone design process. With the advent of SAN file systems this planning process has become even more complex. Hence, currently system designers usually take in the order of months to design and deploy a SAN FS system that is not a pre-packaged one size fits all type of solution. This paper has proposed a SAN FS planning tool that uses the notions of application templates and a planning engine to assist a system designer with the design process. SAN file systems have also been advocated as one approach that allows multiple NAS servers to share a single underlying SAN. In future, we want to extend our planning tool to provide planning support for this hybrid NAS and SAN FS environment.

References

[1] SAN Architect. *EMC*.

[2] D. Aggrawal, J. Giles, K. Lee, K. Voruganti, and K. Filali. Policy-Based Validation of SAN Configuration. In *POLICY*, 2004.

[3] E. Anderson, M. Kallahalla, S. Spence, R. Swaminathan, and Q. Wang. Ergastulum: quickly finding near-optimal storage system designs. *HPL-SSP-2001-05*, 2001.

[4] C. Brooks, A. Ansari, M. Rosichini, L. Stehlik, and E. Wong. Introducing the SAN File System. In *IBM Redbook SG24-7057-01*, 2004.

[5] R. Burns. Data Management in Distributed File System for storage area networks. *PhD Thesis, UCSC*, 2000.

[6] T. Cromelin. BrightStor SAN Designer: Storage Life Cycle Management. In *White Paper, Computer Associates*, 2003.

[7] G. Gibson, B. Halevy, and B. Welch. NFS Extensions for Parallel Storage. In *Panasas Position Paper*, 2003.

[8] J. Menon, D. Pease, B. Rees, L. Duyanovich, and B. Hillsberg. IBM Storage Tank–A heterogeneous scalable SAN file system. In *IBM Systems Journal, Vol. 42. No. 2*, 2003.

[9] J. Ward, M. O'Sullivan, T. Shahoumian, and J. Wilkes. Appia: Automatic Storage Area Network Fabric Design. In *FAST*, 2002.

[10] Kang won Lee. . *Personal Communication*, 2004.