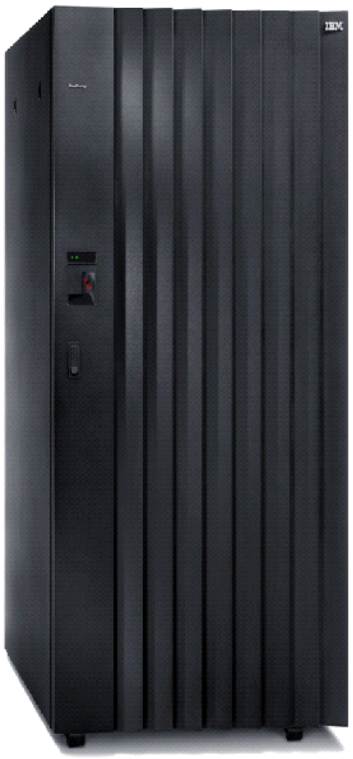
Logical configuration concepts for IBM System Storage™ DS8000 series in large data centers



This document can be found in the IBM Techdocs library.

Version 1.5 (2008-04-15)

Wilhelm Gardt (willigardt@de.ibm.com)
Peter Klee (peter.klee@de.ibm.com)
Gero Schmidt (gerosch@de.ibm.com)

IBM Systems and Technology Group (STG)
IBM System Storage - Advanced Technical Support (ATS)
European Storage Competence Center (ESCC), Mainz, Germany

Trademarks

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX, FICON, IBM, IBM (logo), System i, System z, HACMP, DS4000, DS6000, DS8000, FlashCopy, TotalStorage, System Storage, DB2, z/OS

For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml

The following are trademarks or registered trademarks of other companies:

HP, HP-UX, are trademarks of Hewlett-Packard Company in the United States, other countries, or both

Veritas, VxVM, Veritas Volume Manager, are trademarks of Symantec Corporation in the United States, other countries, or both.

Solaris, Sun, Solstice Disk Suite are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

SAP, SAP R/3 Enterprise are trademarks of SAP AG in Germany, other countries, or both.

Oracle, Oracle ASM, Oracle Cluster are trademarks of Oracle Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Disk Magic is a registered trademark of IntelliMagic, Inc. in the United States and other countries

Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.

Any other trademarks, registered trademarks, company, product or service names may be trademarks, registered trademarks or service marks of others.

Disclaimer

This paper is intended to be used as a guide to help people working with IBM System Storage DS8000 series storage systems. It discusses findings based on configurations that were created and tested under laboratory conditions. These findings may not be realized in all customer environments, and implementation in such environments may require additional steps, configurations, and performance analysis. This information does not constitute a specification or form part of the warranty for any IBM or DS8000 series products. Specific parameters or settings that are described in this document may have been set to meet the requirements of this study and do not necessarily represent “correct”, “preferred” or “recommended” settings for a production environment.

No part of this document may be reproduced or transmitted in any form without written permission from IBM Corporation. Product data is subject to change without notice. This information could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services do not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectually property rights, may be used instead. It is the user's responsibility to evaluate and verify the operation of any non-IBM product, program or service.

THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS", WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT OR INTEROPERABILITY.

IBM shall have no responsibility to update this information. IBM products are warranted according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. IBM is not responsible for the performance or interoperability of any non-IBM products discussed herein.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY, 10504-1785, U.S.A.

The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer’s ability to evaluate and integrate them into their operating environment. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Abstract

In a customer IT environment with hundreds of databases, ERP systems and data warehouses, storage allocation is comparable to assembly-line work. Vendor recommendations regarding logical volume layout are usually difficult or impossible to implement in this type of infrastructure.

This white paper describes best practices to deal with such a situation by deploying IBM System Storage™ DS8000 storage subsystems. With a so-called "*storage factory*" approach, storage capacity is allocated to applications in a strictly regulated manner, resulting in an *automation process for providing storage capacity*.

The results of this approach are as follows: applications are not assigned to dedicated storage subsystems; logical volumes are distributed among storage ranks if possible; logical volumes have fixed sizes; additional storage subsystems are ordered only if the available subsystems are filled up to a certain percentage, and so on.

At least one of IBM's customers can testify that despite this "carefree policy", 80 to 85 percent of its applications are running with good or at least satisfactory performance. This customer is making special arrangements to further improve I/O performance for very critical applications.

Authors

The major chapters have been written by individual authors. So if you have questions regarding a subject from a specific chapter, please don't hesitate to contact the author of the chapter directly:

Chapter 1 ***Storage factory approach***

Peter Klee (peter.klee@de.ibm.com)

Chapter 2 ***Balanced logical configuration approach***

Gero Schmidt (gerosch@de.ibm.com)

Chapter 3 ***Logical volume layout for databases***

Wilhelm Gardt (willigardt@de.ibm.com)

Table of Contents

Introduction

Data centers have evolved from machine rooms for large and complex computer systems in the early phase of commercial computing to multi-platform and multi-component environments that communicate via various network topologies and technologies. The boom of microcomputers in the 1980s and the development of server and networking technologies in the 1990s have resulted in the creation of huge data centers with many hundreds or even thousands of servers located at different physical sites. Additional equipment like power distribution, air conditioning, networking and storage increases this complexity further.

In order to distinguish between different kinds of data centers, it is necessary to look beyond the sheer amount of hardware and equipment deployed in them. The most useful way to differentiate data centers is in terms of the applications running in them. By analyzing the applications themselves, their communication with different components, their interactions with other applications or even business to business relations, each data center becomes a unique construct. Using this view, the following kinds of data centers can be identified:

1. Data centers with a huge number of different applications

   This kind of data center is typically operated by data center service providers that offer outsourcing, housing and hosting services. In this kind of data center, a service provider will run systems for many different customers. Each customer has a set of applications that need to interoperate with each other, but their business data must be held in isolation from other companies' systems.

2. Data centers with many applications and dependencies between them

   In these data centers, customers run environments that provide services to their clients based on common databases that hold market data and customer profiles. Examples are financial institutions like banks or insurance companies. They may provide services that combine account management, online banking, investment management, transaction services and so on. Each service may be represented by a group of different applications that use information from other service applications. In addition to these core business applications, applications for controlling, HR and customer relationship management are often also located in the same data center.

3. Data centers with chains of applications

   These data centers are run by large manufacturing companies and other customers that need to control production processes and provide tools and databases for product development and research. The interactions between the applications in this environment are restricted along the production chain, rather than across the complete range of applications.

4. Data centers with large-scale CPU and storage resources

   This kind of data center is typically operated by research centers. These data centers tend to gather and/or archive huge amounts of data. Processing and analyzing this data requires considerable CPU resources.

The categories of data centers shown above can be seen as a "base set" of data center types; combinations of these types are also possible. For example, a pharmaceutical company's data center may combine features of both a production-line-driven data center (as described in item 3) and a research-driven data center (as described in item 4).

The challenge of managing these large environments lies in the ability to deal with huge numbers of entities and the relationships between them.

For example, having hundreds of servers with two Fibre Channel Host Bus Adapters (HBAs) means

that you need twice as many zone definitions in the storage environment. Each server requires a number of storage volumes which must be generated at the storage subsystem level and assigned to individual HBAs. This requires the management of hundreds or even thousands of entities. Graphical User Interfaces can help, but even in a strong hierarchical structure, finding a single entity or defining groups and dependencies between entities can be sometimes very challenging.

A way to improve the management of such an environment is to divide the environment into pieces, or 'building blocks', that can be managed individually. Each building block has a special purpose, and each has interfaces that enable interactions with other building blocks. Once the data, the functionalities and the interfaces of each building block are defined, the processing of management tasks can be automated using scripts, batch jobs or standard management software components which can be integrated to generate a workbench for the operating staff.

1 Storage factory approach

In this chapter, we provide an overview of possible strategies for implementing, managing and maintaining a storage environment for various types of large-scale data center. It gives a set of ideas and recommendations based on several projects in major data center implementations.

The scope of this white paper is focused on providing storage to servers as a commodity – treating storage as part of the general infrastructure of the data center. This is achieved by defining a storage infrastructure that delivers connectivity, capacity and other important functionalities.

The idea behind this so-called *storage factory approach* is to organize the storage infrastructure as a production line that generates these deliverables. The factory is constructed using a set of building blocks. In the following sections we give an overview of the different kinds of building block, their purposes and interfaces, and how to create applicable functionalities for applications.

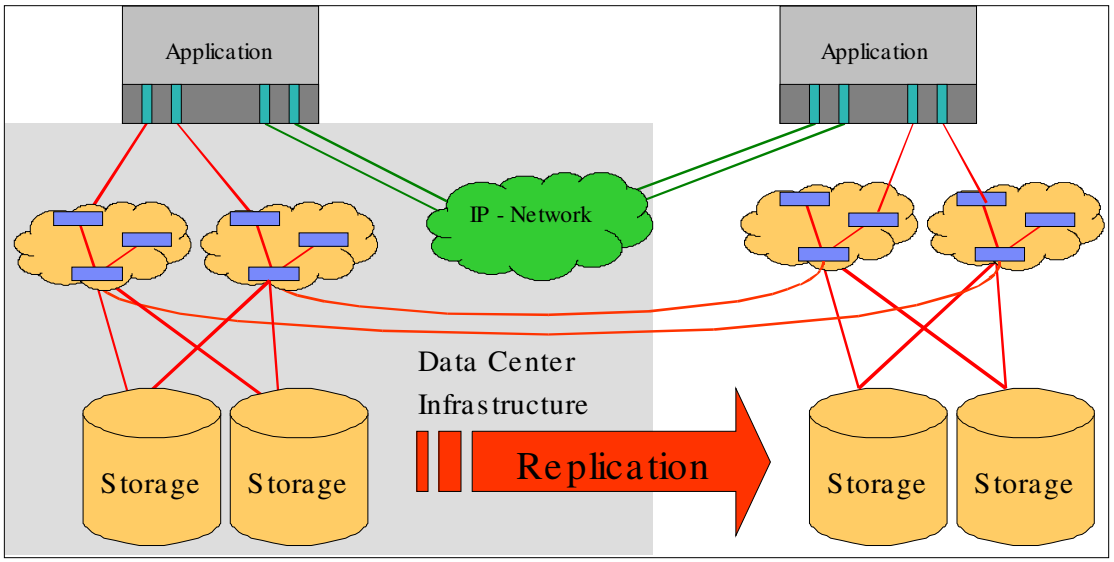


Figure 1: Example of a storage factory as a part of the data center infrastructure

1.1 Building blocks

The storage factory requires two different general kinds of building blocks, which are the storage itself and the connectivity which provides access to the storage for application server or other storage subsystems.

1.1.1 Storage subsystems

One goal is to define an allocation policy for the storage subsystem that delivers the optimum balance between scalability and performance. The effect of such a policy may be to define standard portions of storage that are provided to the server like a set of fixed-size LUN addresses. This can optimize the cut-off of storage when the ranks of the storage subsystem are getting used up.

The allocation policy must also meet criteria to provide copy services functionalities without influencing normal storage operations. For example, it should be possible to allocate FlashCopy target volumes to different ranks than the ranks on which the source volumes are stored, but they still should be managed by the same DS8000 processor complex.

The assignment of the storage host ports also has to be considered. Although distributing I/O across many host ports is usually recommended, it may be a better approach to keep a port provisioned to take care of scalability issues created by external demands – for example, serving multiple customers

or providing connectivity to other storage subsystems for remote copy functionalities.

In most data centers that run many different applications, 80% or more of the applications do not need to make extraordinary demands on the performance of the storage factory. This means that these applications can be served by a standardized configuration of the storage subsystems.

1.1.2 Storage area networks

The purpose of a storage area network (SAN) is to provide connectivity, both to the servers that use the storage factory environment and to other storage components that are part of the storage factory. In the latter case, connectivity is required for storage functionalities like virtualization and remote copy functions.

1.1.2.1 Separating fabrics

A large storage environment leads to a large SAN, which increases the complexity of SAN management. For example, in general, it is strongly recommended to introduce redundancy by implementing two independent fabrics for dual path access. One possible way to optimize the management is to implement large scale switches or directors in order to reduce the amount of devices in each fabric. The disadvantage of this approach is the fact that all applications are connected to one single configuration. This may increase the logistics effort in the customer's change management process.

When creating building blocks for storage area networks, separating the fabrics may be a possible solution. A suitable way to separate the fabrics must be identified according to conditions at the customer site - such as the customer's organization of operational staff or their business model, e.g. when multiple customer clients are using the data center storage environment.

The following example may illustrate this approach: let us assume that a customer is running a data center as a service provider. The customer uses separate management teams to manage service provision for different groups of clients. In this example, an association of dedicated fabrics for each management team may be suitable.

To avoid an increasing overhead of managing different fabrics, a set of rules, methods and standards should be defined which are applicable for all fabrics. This can be achieved by establishing a central repository for the SAN environment and a common set of scripts for operational tasks, monitoring and reporting.

1.1.2.2 Connecting server HBAs to DS8000 HAs (cabling)

The access of the servers to the logical volumes at the storage subsystem is managed by mapping server HBAs to DS8000 volume groups and by associating the DS8000 volume groups to DS8000 host ports.

The following policies for assigning ports should be considered:

- Assign dedicated storage host ports to applications or groups of applications. This approach is the most compatible with the multiple fabrics concept, as described above. A disadvantage is that it requires an accurate planning of the required number of host ports and an estimation of expected growth behavior of the application.
- Define a group of I/O ports for a group of applications – for example, all SAP systems in the data center. This group of ports should be assigned to a dedicated fabric, which is in turn assigned to the group of applications. The storage volumes are assigned with respect to load-balancing across the storage ports. With this approach the applications are kept together and can easily be monitored. The disadvantage is that an equal balancing of the I/O load across the ports requires good documentation or automated processing.

- Assign volumes to all host ports, while managing access to the storage only via zones of the fabrics. This is a more general approach to host port usage, which is easier to manage from the storage point of view. The disadvantage of this approach is that the I/O loads of different applications overlapped at storage ports.

1.2 Virtualization

Virtualization in this context implies the use of a virtualization platform like IBM SAN Volume Controller (SVC), which is an option for all the standard storage platforms. The implementation of a virtualization platform provides the following benefits:

- Integration of different storage platforms:
  Multiple storage classes can be accommodated within the SAN – from high-end storage systems like the DS8000 series, through mid-range systems like the DS4000 series to low-end storage on platforms with near-line drives in the back-end (SATA or FATA drives). Storage classes can be defined using SVC, which enables applications with different storage requirements to be assigned to volumes with different characteristics in terms of speed and capacity.
- Single set of storage functions across multiple storage platforms:
  This is typically used to provide remote copy functionality from one storage platform to another (for example, from a DS8000 to a DS4000). For copy functions within the same storage platform, it is usually more efficient to use the platform's native copy functionality.

1.3 Storage factory guidelines

In the following section, some guidelines are given, based on experience gained in many storage implementations.

1.3.1 Define a standard configuration for hardware

In the end, each building block of the storage factory is based on hardware components like storage subsystems, switches and so on. For a large environment, large numbers of these components will need to be installed. To minimize the effort of sizing and customization for each installation, it makes sense to use a single standardized configuration.

Defining a standard configuration is a trade-off between the following aspects:

Performance	The system should be able to give the best performance to the servers
Scaling	It should be easy to increase the resources assigned to each server
Utilization	Each component should be utilized to its full capacity

It is not always possible to maximize all three aspects, because they may pull in opposite directions. For example, optimizing storage performance for a certain application in a most performance optimized could mean that adding storage to this application at a later point in time will not be possible without unbalancing certain resources in the storage subsystem. On the other hand, to take advantage of investment in a storage subsystem, it is in the customer's interest to utilize the full capacity of each storage subsystem – which might produce serious logistical problems when applications request more storage or data needs to be migrated.

To create an effective compromise between the three aspects, the following consideration may help:

1. Experiences of large server environments has shown that in an average data center, more than 80% of the applications work well in a standardized configuration. The remaining 20% very often require a dedicated storage environment or solution anyway.
2. Instead of utilizing all of the given capacity, it may be better to keep 5% to 10% per system as

spare storage. Managing storage subsystems that are operating at full capacity can be difficult, and can lead to administration becoming more expensive than the spare storage would have been.

3. A careful forecast of storage demands for the next investment period is recommended. This includes defining a utilization threshold in order to be ready for timely provisioning of further storage before current capacity limits are reached.

1.3.2 Storage allocation

With the DS8000 and DS6000 series storage subsystems, it is in general possible to create and assign volumes of any size to any server. To optimize utilization and facilitate migrations and mirroring to other storage subsystems with Copy Services functions, it is usually best to operate with a set of fixed volume sizes (for example 16, 32, 64 and 128 GB volumes). This makes capacity management relatively straightforward. On the other hand, it is possible that applications may not always require standard-sized storage volumes: for example, database log files will typically be stored in volumes smaller than 16 GB.

Extent pools are containers from which volumes are composed. Each extent pool contains one or more ranks. To keep control of volume allocation, ranks can be associated with dedicated extent pools. With this approach, the flexibility of the virtualization capabilities of the DS8000/DS6000 will be reduced. A reassignment of ranks is only possible when all volumes of a rank have been deleted. See section 2.2.4 (Extent pool creation and volume allocation algorithms) on page 24 for a detailed description of how ranks and extent pools should be used.

To take full advantage of the virtualization capabilities of the DS8000/DS6000, Logical Subsystems (LSS) should be assigned to certain applications, groups of applications, servers or groups of servers. Each instance can use multiple LSSes, but should be assigned at least 2 LSSes to enable performance-balancing.

1.3.3 Storage area network

According to section 1.1.2 (Storage area networks) on page 8, the way that the topology of the fabrics should be defined depends on the construction of the building blocks. Using large port scale directors in a flat network topology enables a simple approach to connecting servers and storage. There are fewer devices to be managed in a flat network topology. This enables a simpler fabric management, because no inter-switch links have to be monitored; it also means simpler management, as servers and storage are connected to a single director.

Directors are typically designed to provide high-speed access at 4 Gbit/s. Connecting hosts to these ports will usually not utilize this bandwidth. This leads to the disadvantage that the high investment costs of SAN directors and the low utilization of the ports can lead to a higher price per SAN-port.

A well sized core/edge topology may optimize both the cost and the effectiveness of the infrastructure. All server and storage ports are connected to edge switches with a lower port count. The edge switches are connected to core switch via inter-switch links (ISLs). If the data center consists of more than one site, the core switches in each site are interconnected to a backbone.

For a two-site data center, a possible concept for deploying fabric topologies may look like the following:

Fabrics are categorized into simple, standard and extended fabrics. Each topology provides a certain capacity of SAN-ports, depending on the port count of the switch hardware used. It is possible to upgrade each category to the next level up.

Simple core topology
The simplest kind of fabric is one with a single switch in each location, where the switches are connected by two ISLs. This construct can be seen as a special case of a core/edge design, whereby

in each site the edge switch and the core switch are the same physical switch. This may be the entry level for scaling to the next level.

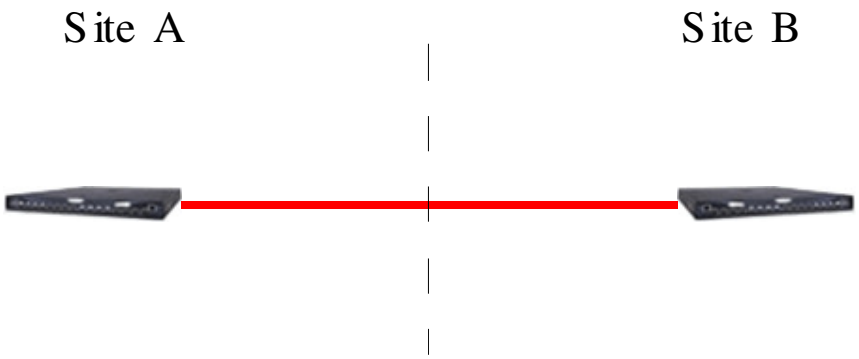


Figure 2: Simple core topology

Standard core/edge topology

With this topology it is possible to scale by the number of edge switches which can easily be connected to the core switches, without changing the topology. This approach can be maintained as long as enough ports on the core switches are available.

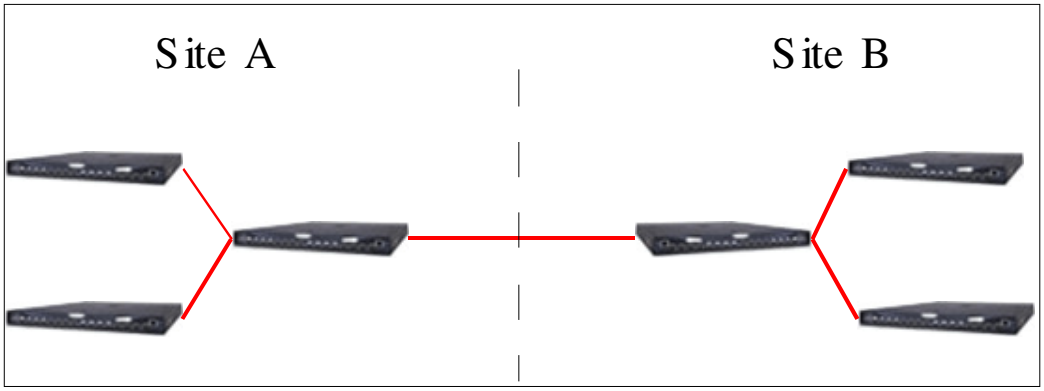


Figure 3: Standard core/edge topology

In the example above we have two core switches and two edge switches connected to each core switch. Each ISL consists of two physical links. Each core switch is therefore using six ports for ISL connections. Assuming that the core switches each have 16 ports (and disregarding a potential ISL over-subscription of the ISLs between the two core switches) this configuration offers a further ten free ports per core switch, making it possible to connect five more edge switches to each core switch.

Servers and storage are connected only to the edge switches. Edge switches with 4Gb/s high-speed ports can be used to deliver excellent I/O performance for the storage infrastructure.

A performance impact due to the ISL over-subscription between both core switches can usually be avoided if applications only access storage at their own site. Even in a high availability solution like HACMP spanned across both sites, normal operation should run on the site where the storage is allocated. Cross traffic will only occur in case of a cluster take-over to the other site.

If host-based mirroring and parallel access applications like Oracle RAC are used, the total required ISL bandwidth must be reconsidered.

Extended core/edge topology

If the fabric size exceeds the capabilities of a standard core/edge topology, the core switch in one or

both sites must be upgraded. This could be done by replacing it with a core switch with a higher port count switches, or by deploying directors.

The standard core/edge topology also has another disadvantage: an outage of one core switch will cause the outage of the entire fabric. Although it is always strongly recommended to implement two fabrics for redundancy reasons, it may be the case that a more resilient core architecture is required. A possible approach could look like the following example.

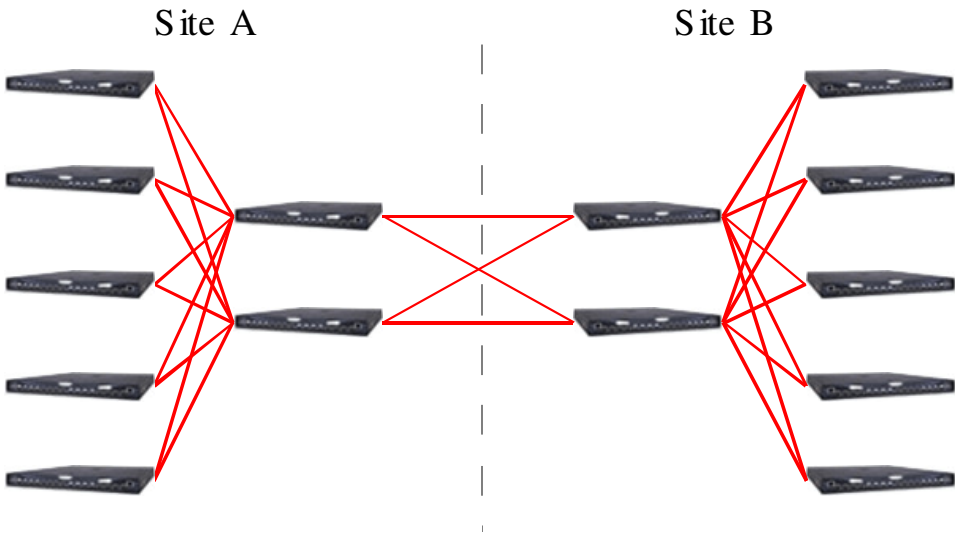


Figure 4: Resilient core/edge topology

The core of the fabrics is a mesh of four switches, with two at each site. The edge switches are connected to both of the local core switches. This ensures that the fabric will continue to work if one core switch at each site fails. This improves the availability of the fabric during maintenance operations – for example during microcode updates.

For all topologies it is recommended to define point-to-point zones, which enhance problem determination and prevent unwanted side-effects. A zone in a fabric can be seen as a 'virtual' SCSI cable. If more than one initiator and more than one target are added to the zone, this virtual SCSI cable would connect all these devices to one virtual SCSI bus. A SCSI bus reset, for example, would affect all the devices which are members of this zone.

Very large configurations with hundreds of multiple accesses to the same device ports are an exception to this rule. One example of such a configuration is a data backup environment consisting of some Tivoli Storage Manager servers with a high number of HBAs. Each HBA must access a huge tape library with a large number of tape devices. In this case, the consolidation of a certain number of tape devices into one zone with a single HBA would reduce the complexity of the zoning definitions.

1.4 Managing a storage factory

The main challenge of managing a storage factory is to transcribe the processes defined in each building block. The management functions for the storage factory should cover the following tasks:

- Adding, changing and deleting volumes
- Monitoring events and storage and network performance
- Providing resources and functions to perform migration tasks

These functions must be applied to each building block, and must be applied in a consistent way to all

affected components.

1.4.1 Automating management

During the definitions of the building blocks, certain functionalities have been defined (for example, allocating storage, assigning storage to hosts, and so on). In large environments it is challenging to find the right resources to fulfill the requested operations.

For example, in a single DS8000, many thousands of volumes can be created. A large storage factory may consist of ten, twenty or even up to a hundred DS8000 subsystems. Hosts may access more than one storage subsystem. This results in a very complex configuration, where the identification of all affected components and entities (zones, host adapters, LSSes etc.) becomes very complicated. This means that an automated way of managing the storage factory is required.

A very effective way to automate the management of the storage factory is to create scripts written in a shell language, like Perl or a similar interpreter language. It is possible to write simple helper scripts or even a comprehensive framework which enables the addition, modification and even removal of management functionalities.

Scripts may be used by different operational staff that manage the storage factory. This means that all scripts must be written a way that enables people with different skill levels to run them. For example, certain management operations can only be executed at times when there is little or no storage I/O activity. These scripted tasks may need to be initiated by operational staff working night-shifts.

Once the scripts have been developed, they need to be maintained, because the targets and objective for which the scripts were originally written will change during their lifetime – hardware may change or new functionalities may be required. In order to make the maintenance of the scripting environment robust against these changes, separating the data from the logic of the script and establishing external sources (like ASCII files, stanzas or databases) is recommended. These external sources can be seen as a repository for all necessary configuration data for the storage fabric. In this way, the data can easily be changed or queried, whereas the logic of the script will stay the same as long as the functionality itself does not need to be changed.

1.4.2 Scalability aspects

Scalability is of major importance when designing a storage infrastructure, because the demand for storage is constantly growing. The future capacity requirements expected at the beginning of a year-by-year planning cycle will usually be exceeded before the end of the planning period. For this reason, it is important to define thresholds and alert administrators before storage capacity limits are reached.

The estimation of storage resources for the next investment period is a trade-off between optimized utilization of all resources and the need to keep capacity in reserve in order to be capable of acting before all resources are used up. Despite the desire to keep initial investment as low as possible, the sizing of the storage should include enough room for growth in order to avoid a situation where further investment in new storage resources is required before the end of the current planning period.

A second point to consider is the scalability of Copy Services. Copy Services functions require additional storage and fibre channel ports. If the current storage environment already provides Copy Services for applications, the next investment estimate can be based on the current usage of Copy Services plus a growth estimate for new applications which are using Copy Services.

A third factor related to scalability is the relationship between applications and storage hardware. Applications sometimes run out of storage because the resources in the current storage box are used up. In this situation, resources from other storage subsystems must be assigned. This can be an inconvenient configuration, because the dependencies between the application and the storage infrastructure increase, which can have a negative impact on maintenance workload or in disaster recovery scenarios. It is recommended to use as few storage subsystems as possible for each application. If the storage is distributed across multiple subsystems, a consolidation process should be

planned as soon as possible.

1.4.3 Migration aspects

Migration tasks are recommended in the following situations:

- **New application releases**
  An application running in a data center may consist of a database running on a dedicated server and some application servers acting as a front-end for the users. This configuration will be periodically updated with new front-end functionalities and other enhancements. A major release change of the application could mean that the architecture of the whole configuration must be changed, and may even involve other servers that are attached to separate storage subsystems.
- **New hardware**
  If the server hardware is replaced by new models with higher capabilities, the storage assigned to the old server hardware will typically be replaced also by new storage. Very often, hardware is leased for a certain period and must be replaced when the lease expires.
- **Storage consolidation**
  As described in section 1.4.2 (Scalability), applications that exceed the limits of their current storage subsystem and have additional storage allocated to them from other subsystems should be consolidated by a migration to a single subsystem.
- **Physical move of hardware to other data center sites**
  Storage migrations to other data center sites may take place when new data centers are deployed. Alternatively, parts of the data center may need to be moved for logistical reasons.

The migration itself can be performed either via the storage using normal copy functions, or via the host. Leveraging DS8000 Copy Services can provide a very reliable method of migration. Data can be copied to the target storage subsystem without modifying the production servers. However, when the data has been copied, the host must switch over to the new storage. For Open Systems platforms, this usually requires a shutdown of the applications, a failover to the new storage, and a startup of the applications from there.

1.4.3.1 Migrations with host-based mirroring

Host-based migrations require the host to have access to the new storage. Data will be read from the current storage subsystem and copied to the new one. This usually means either that new HBAs must be installed in the server or that a performance analysis needs to be done. Without new HBAs, the utilization of the internal PCI bus rather than the fibre channel bandwidth becomes a limiting factor. Another reason that may enforce the use of new adapters might be that device drivers in the new hardware (especially if it comes from a different vendor) might not be able to cooperate with the same HBA.

The new storage subsystem should be not too far away from the host, because the latencies of the link will also directly influencing the performance of the applications. In this case a performance degradation would occur after the initial copy phase, when the Logical Volume Manager (LVM) mirror goes into the synchronous mode. This means that for larger distances or higher latencies of the links, the switchover to the new storage should be done quite quickly after the initial copy phase.

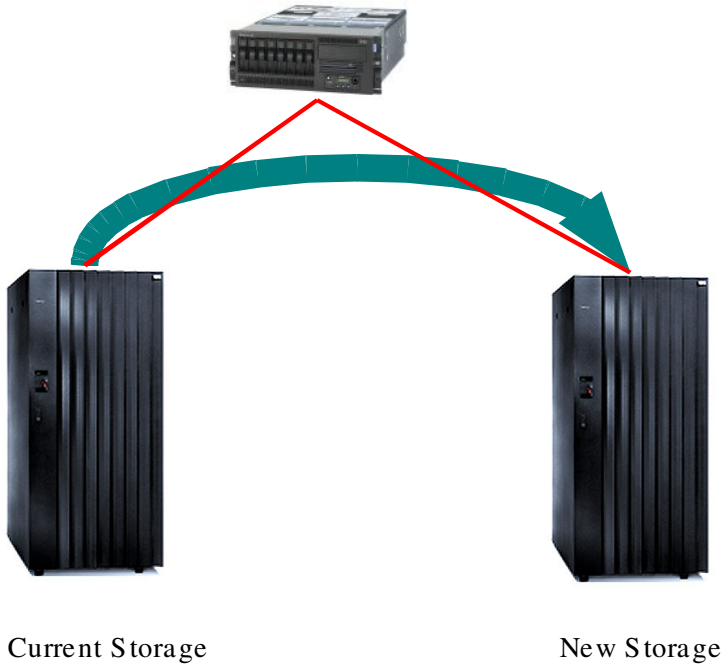


Figure 5: Example of a host based migration. Once these issues have been solved, a very elegant method for migrations is to set up a host-based mirror to the new storage. The application running on that host can continue with normal operation – although during the migration process, the performance of the application may be affected to some extent. Some operating systems allow control of the synchronization throughput, for example by starting multiple synchronization daemons. The switchover to the new storage can be done without interrupting production systems, simply by removing the mirror to the old storage.

1.4.3.2 Migrations using remote copy functions

With remote copy functions of the DS6000 / DS8000 the migration can be done with less impact of the performance and completely independent of the distance to the new storage location, but with the disadvantage, that the production takeover to the new storage implies a downtime to the application. Migrations with remote copy are chosen when the storage and the server must be migrated to new locations.

The replication to the new storage will be setup in general as a asynchronous replication. For production takeover the applications must be stopped before the storage fail over to the new storage can take place. When the new storage is assigned to the same server hardware, the host connection must be changed in that way, that the old storage must be take away from the host and the new one must be assign to the host. Now the applications can be restarted. When the production takeover includes also new server hardware, the applications can now be started at the new host.

It is also possible to migrate storage which is already in a remote copy replication like Metro Mirror (synchronous replication). In this case another asynchronous replication to the new storage location are established as a cascaded Global Copy (asynchronous replication). If the whole Metro Mirror should be migrated to the new location, a second cascaded Global Copy can be established to the new secondary storage. After the initial copy has been passed the production takeover includes that the first cascaded replication is removed and the second cascaded replication is changed to Metro Mirror.

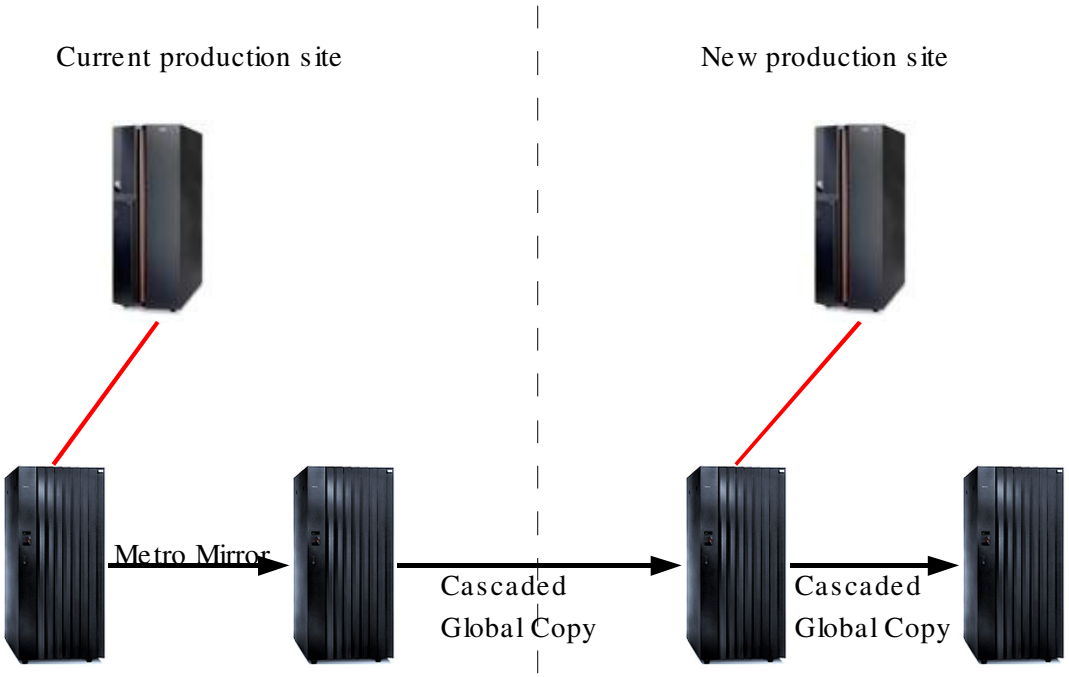


Figure 6: Example of a migration with double cascaded Global Copy

1.4.4 Performance monitoring

The purpose of monitoring the performance of the storage factory is to visualize the load profile in production and give the ability to take action in case performance is being limited by the SAN or the storage subsystem.

In an Open Systems environment, most applications tend to read more data than they write during normal office hours, and write more than they read during the night and at weekends. Very often data downloads, database imports or other cooperative application transactions are performed during specific periods. Beside these activities, data backup jobs also tend to have their preferred execution time. Performance monitoring helps to identify overlapping jobs and resources with free capacity, helping to organize workload so as to utilize the factory more efficiently.

2 Balanced logical configuration approach

This chapter provides an overview of the DS8000 architecture and some logical configuration concepts that attempt to distribute the I/O workload evenly across all DS8000 subsystem resources. Balancing workload can help to avoid hot spots and bottlenecks, which are the most common source of performance problems. The goal is to utilize all available subsystem resources evenly, up to the limits of the subsystem's capabilities.

The chapter outlines performance and layout considerations for large, fast-growing environments where multiple applications or database instances are located on each DS8000 subsystem. It is not intended to discuss the optimal layout for a single database instance. The chapter will introduce a simple and generic logical configuration approach for *sharing* all resources and thus *spreading* all workloads evenly across the whole machine – a sensible approach, especially if little or no information about the particular host system and application workloads is available in advance.

For an in-depth description of the possible architecture and layout considerations for optimal performance, please refer to the following excellent IBM Redbooks:

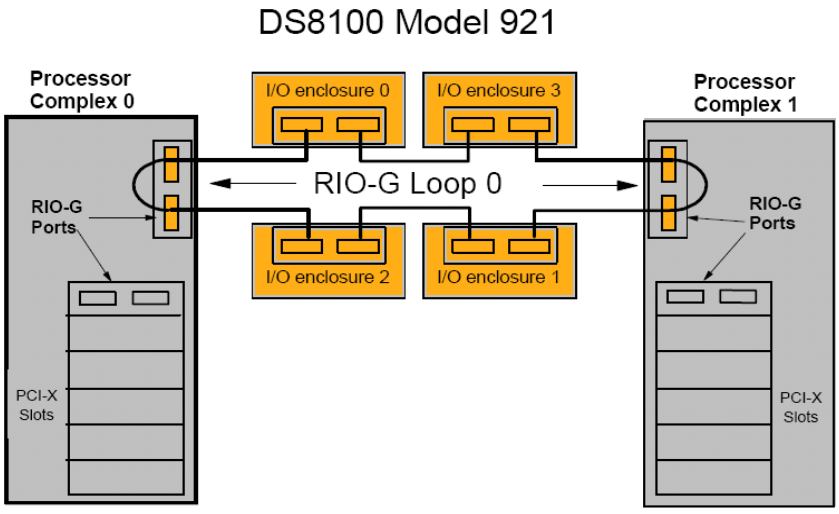
- ✓ *IBM System Storage DS8000 Series: Architecture and Implementation* (SG24-6786)
- ✓ *IBM TotalStorage™ DS8000 Series: Performance Monitoring and Tuning* (SG24-7146)

2.1 Architecture overview

To better understand the concepts for the logical configuration, a short overview of the DS8000 hardware architecture is given in this chapter.

2.1.1 DS8000 processor complex and RIO-G loop interconnect

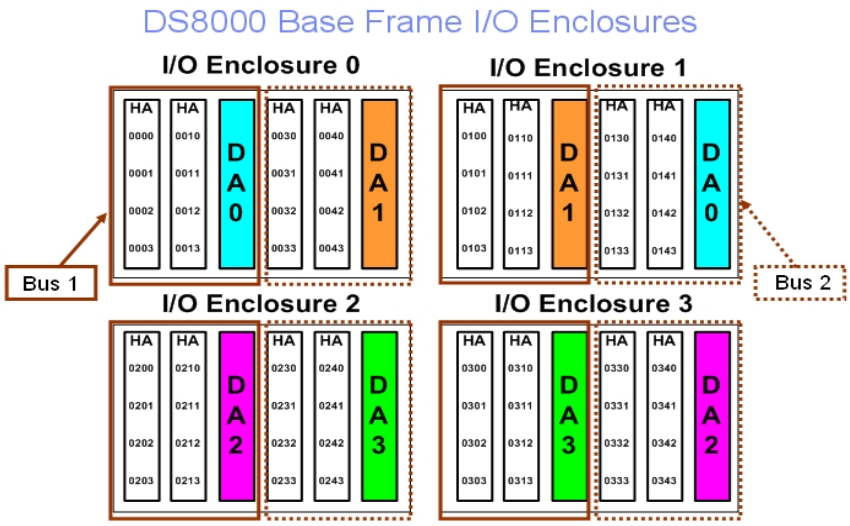
On DS8000 models, there are two *processor complexes*, referred to as *server#0* and *server#1*, which are housed in the base frame that services I/O requests. These processor complexes form a redundant pair, such that if either processor complex fails, the surviving processor complex continues to run the workload. *RIO-G loops* provide connectivity between the processor complexes and the I/O enclosures which contain the host adapter (HA) and disk adapter (DA) cards. It is called a RIO-G loop because the RIO-G connections go from one component to another in sequence, and then back to the first. Each RIO-G port can operate at 1GB/s in bidirectional mode, and is capable of passing data in either direction on each cycle of the port, creating a redundant high-speed interconnection.



The DS8100 has a single RIO-G loop with four I/O enclosures; the DS8300 has two RIO-G loops with eight I/O enclosures.

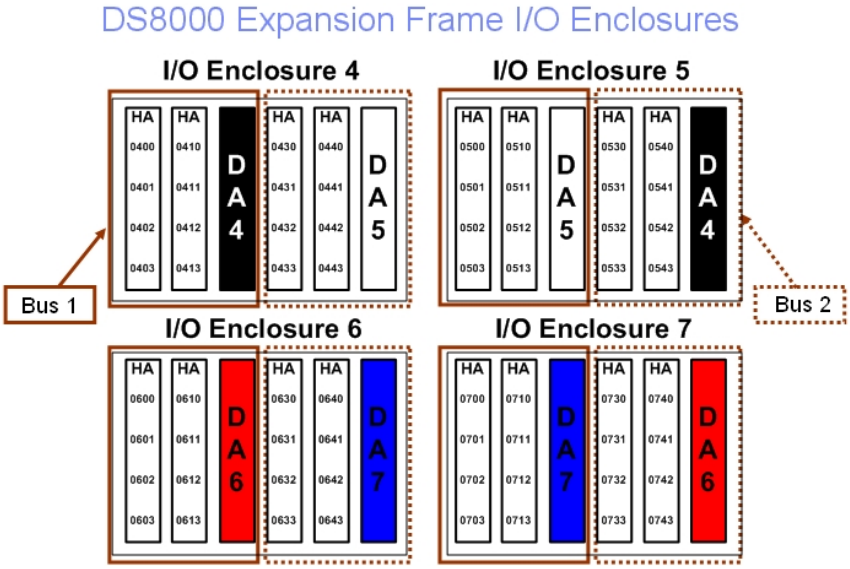
2.1.2 DS8000 I/O enclosures with host and device adapters

The *I/O enclosures* hold the *device adapters* (DAs) that service back-end I/O requests, as well as *host adapters* (HAs) that service front-end I/O requests. All I/O enclosures within the RIO-G interconnect fabric are equally served from either processor complex. Each I/O enclosure has six adapter slots, two slots for device adapters (DAs) and four slots for host adapters (HAs). The six slots are distributed across two internal buses within a single I/O enclosure, with each bus servicing three slots for two host adapters and one disk adapter.



The two disk adapter cards of a DA pair are split across two adjacent (left and right) I/O enclosures for redundancy, with each DS8000 storage server always having a closest path to one of them. Server#0 has a closest path to enclosures 0 (4) and 2 (6) (left-side enclosures) and server#1 has a closest path to enclosures 1 (5) and 3 (7) (right-side enclosures) on the RIO-G loop. The number of disk drives installed determines the number of *device adapter* pairs (DAs) required. The overall throughput of the DS8000 subsystem scales with the number of installed DA pairs.

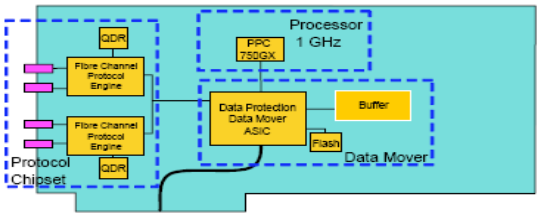
The DS8100 has a maximum of four DA pairs (DA pair install order: DA2-DA0-DA3-DA1) evenly distributed across the four I/O enclosures on the one RIO-G loop, as shown in the chart above. The DS8300 has a maximum of eight DA pairs (DA pair install order: DA2-DA0-DA6-DA4-DA7-DA5-DA3-DA1) evenly distributed across the eight I/O enclosures on the two RIO-G loops, as shown in the chart below.



In general 64 disk drive modules (eight ranks) are installed per DA pair before the next DA pair is used (although a special ordering option is available to allow only 32 disk drives per DA pair, for environments with a high throughput demand and only a low number of required disk drives). After all DA pairs are installed and equipped with 64 disk drive modules, additional disk drives are installed on DA pairs DA0 and DA2, which then will service twice as many disks as the other DA pairs in a fully equipped DS8000 machine. Therefore a DS8100 with four DA pairs and 256 DDMs or a DS8300 with eight DA pairs and 512 DDMs provides a balanced hardware configuration with regard to the disk back-end resources.

Host adapter cards (HAs) are installed as required to support host connectivity. As the full box bandwidth scales with the number of DA pairs, you also need to balance the HA card bandwidth with the available DA card bandwidth. The positions for the DA cards are fixed, while the HA cards follow a given installation order. HA cards are typically ordered in pairs for availability and independently for the base and/or the expansion frame. The first four HA cards in the base frame are 023x, 030x, 003x, 010x on the first RIO-G loop, and 063x, 070x, 043x, 050x in the expansion frame on the second RIO-G loop (see previous charts). When ordering eight HA cards for a DS8300 with two installed RIO-G loops, consider ordering four HA cards for the base frame and four HA cards for the expansion frame to balance the host I/O load across both RIO-G loops.

Each DS8000 Fibre Channel HA card provides four ports to attach to the host systems.



Each of the four ports on a DS8000 adapter can independently be configured to support either Fibre Channel protocol (FCP) or FICON. The HA card itself is PCI-X 64-bit 133MHz, and is driven by a new high-function, high-performance ASIC, as illustrated in the figure above. Each Fibre Channel port supports a maximum of 509 host node port logins. The overall bandwidth of one HA card scales well up to two ports, while the other two ports simply provide additional connectivity. For workloads with high sequential throughputs, it is recommended to use only one of the upper pair of FCP ports and one of the lower pair of FCP ports of a single HA card, and spread the workload across several HAs. However with typical transaction-driven workloads showing high numbers of random, small block-size I/O operations, all four ports can be used.

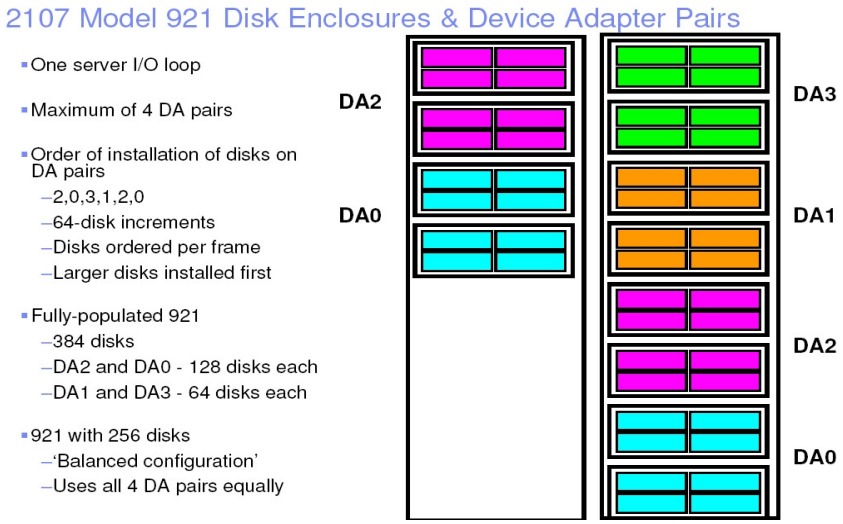
When attaching host systems that use multi-pathing device drivers, it is recommended to spread the host connections evenly across multiple (at least two) HA cards, I/O enclosures, buses and RIO-G loops (if available), in order to maximize performance and minimize the points where a hardware failure would cause outages on multiple paths. So for a host system with two FC links to a DS8100, it is sensible to consider using one HA port in a *left* I/O enclosure (e.g. #0 or #2), and one HA port in a *right* I/O enclosure (e.g. #1 or #3). For a host system with four FC links to a DS8100, consider using one HA port in each of the four I/O enclosures. If a host system with four FC links is attached to a DS8300, consider spreading two HA connections across enclosures in the first RIO-G loop and two across enclosures in the second RIO-G loop.

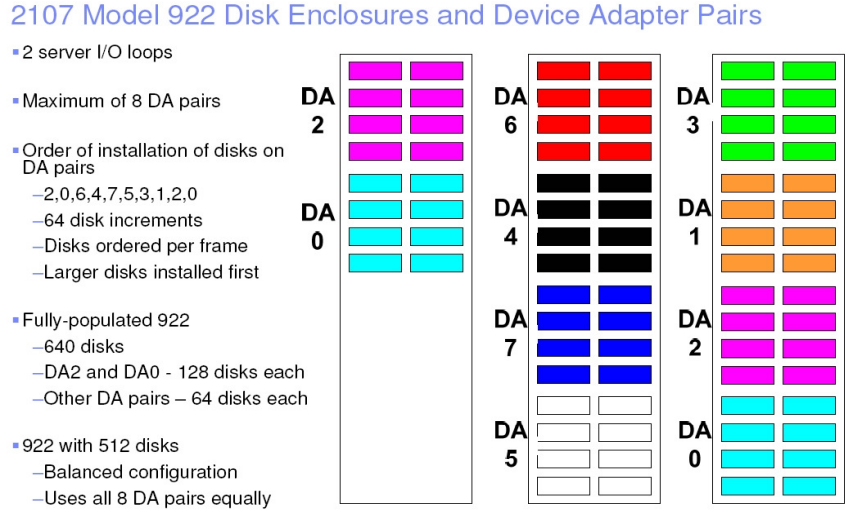
2.1.3 DS8000 physical disk drives

In the DS8000 hardware, certain physical disk locations are cabled to certain DA pairs during installation. The relationship between physical disk location and DA pairs on the DS8000 is fixed.

A group of 8 disks makes up an *array site*, and is related to a specific DA pair. Array site IDs for the DS8000 do not have a pre-determined or fixed relation to physical disk locations. Any array site ID may be used with array sites anywhere in the DS8000. This means that it is very important to check how the array sites have been assigned to DA pairs, in order to have control over the mapping of

logical volumes and the workload distribution across the available DA pairs. The best way to see the relationship between array site IDs and DA pairs is to use the DS command-line interface (CLI) **`lsarraysite`** command, or, if these have already been configured into arrays, using the **`lsarray -l`** command.





Array sites are logically configured into RAID *arrays* and finally into *ranks*. There is a one-to-one relationship between each array site (8 disk drives) and each rank. The *rank* finally provides a certain amount of logical storage *extents* of 1GB (2^30 bytes for fixed block volumes / Open Systems) in size which later are used for the creation of volumes for the attached host systems when assigned to an *extent pool*.

Note that there is no pre-determined or fixed hardware relationship between the physical disk locations or array sites and a specific DS8000 processor complex. Each processor complex or DS8000 server has full access to all array sites of a DA pair. An assignment to server#0 or server#1 only takes place by software when performing the logical configuration and finally assigning the configured rank to an *extent pool*. All ranks assigned to *even* numbered extent pools (P0, P2, P4, ...) form *rank group 0* and are managed by DS8000 server#0. All ranks assigned to *odd* numbered extent pools (P1, P3, P5, ...) form *rank group 1* and are managed by DS8000 server#1. Only in case of an unavailable DS8000 server (due to a code load or failure) will the alternate server take over the ranks of the other rank group.

2.2 Logical configuration overview

This chapter provides a brief overview of the workflow for logically configuring a DS8000 storage subsystem. Logical configuration deals with the creation of arrays, ranks, extent pools, volumes, and finally the assignment of the volumes to the attached host systems.

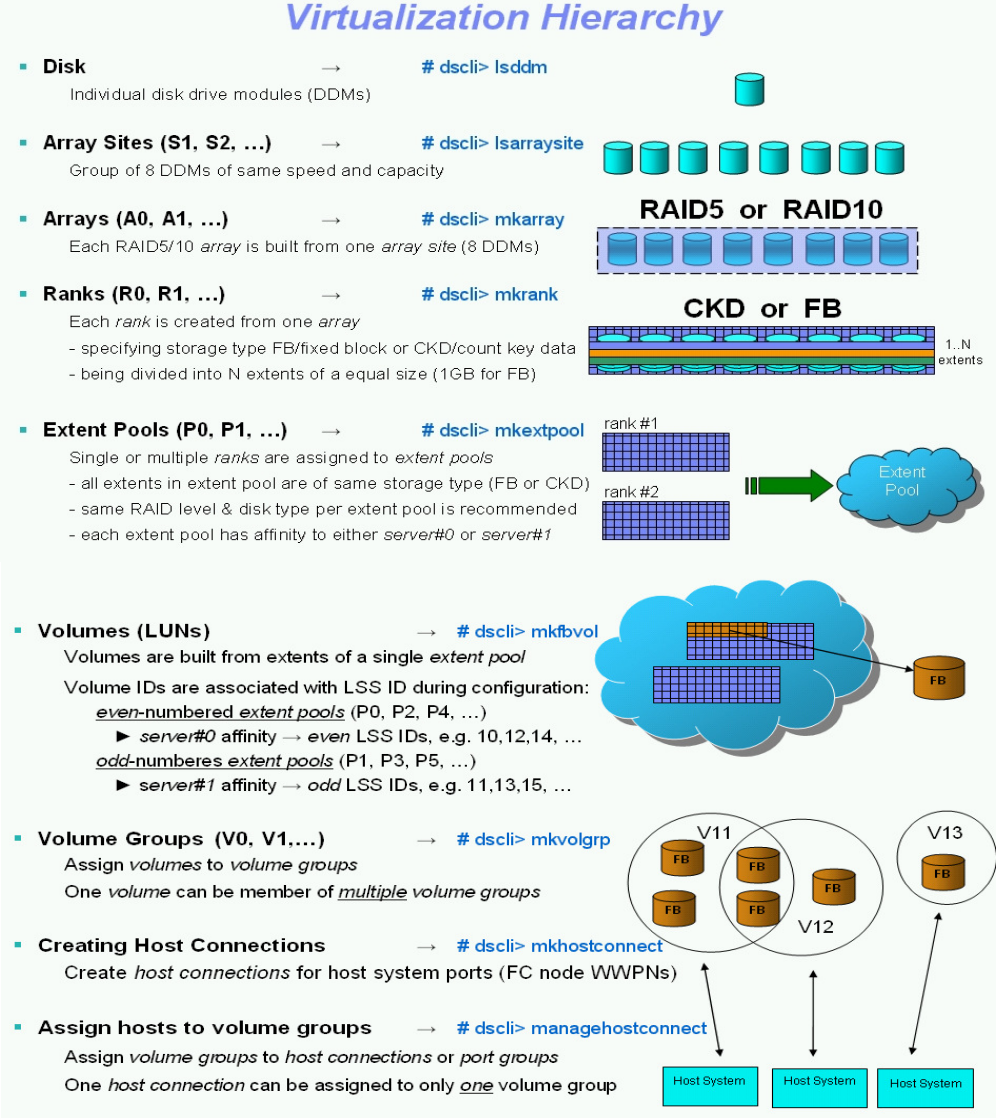
With the DS CLI (command-line interface), you can configure the storage unit using simple and well-structured commands. Using the DS CLI is the most efficient way to perform logical configuration. The basic commands are grouped into five categories for managing logical objects like, for example, *volumes* or *host connections*:

- *Make* commands starting with **`mk`** to create objects, e.g. **`mkhostconnect`**
- *Change* commands starting with **`ch`** to change object properties, e.g. **`chhostconnect`**
- *List* commands starting with **`ls`** to show a list of objects, e.g. **`lshostconnect`**
- *Remove* commands starting with **`rm`** to delete objects, e.g. **`rmhostconnect`**
- *Show* commands starting with **`show`** to show details of an object, e.g. **`showhostconnect`**

2.2.1 Logical Configuration Steps

When configuring a DS8000 storage image for attached Open Systems host systems, you need to perform the following basic steps, using either the DS8000 Storage Manager or the DS CLI:

1. Prepare the available *physical storage capacity*.
   (a) Create *arrays* from *array sites* (8 DDMs) by specifying the *RAID level* (RAID-5 or RAID-10)
   (b) Create *ranks* from the *arrays* by specifying the *storage type* (FB or CKD)
   Fixed block (FB): used for Open Systems hosts and System i hosts
   Count key data (CKD): used for System z hosts
   (c) Create *extent pools* populated with *ranks* to provide the logical storage capacity from which the volumes for the individual host systems will be created.
2. Configure the DS8000 subsystem's *I/O ports* by setting the *Fibre Channel topology* for the available host adapter FC ports that are used for the host attachments.
   (a) FC-AL: The FC-AL topology setting enables the SCSI ULP (upper layer protocol) with a FC-AL topology.
   (b) SCSI-FCP: The SCSI-FCP topology setting enables the SCSI ULP with a point-to-point or switched fabric topology. PPRC path I/O operations can only be enabled using this setting.
   (c) FICON: The FICON topology setting enables the FICON ULP with a point-to-point or switched fabric topology.
3. Create *volumes* for the attached open systems host systems (FB volumes).
   (a) Create FB *volumes* from extents of a given extent pool.
   (b) Create *volume groups* to group volumes for a common assignment to a host system.
   (c) Create *host connections* by specifying the WWPNs of the attached host system ports.
   (d) Finally assign the *volume groups* to *host connections* to enable I/O access.



2.2.2 Array creation and RAID level

When creating the arrays, you need to specify the array site and the RAID type, either RAID-5 or RAID-10. *RAID-5* optimizes cost-effective performance while emphasizing the use of available capacity through data striping. It also provides fault tolerance if one disk drive fails. Hot spots are avoided by distributing parity across all the drives in the array. *RAID-10* optimizes high performance while maintaining fault tolerance for disk drive failures. Volume data is striped across several disks and the first set of disk drives is mirrored to an identical set. RAID-10 can tolerate at least one, and in most cases, multiple disk failures.

With RAID-10, each write operation at the disk back-end initiates two disk operations to the rank. With RAID-5, an individual random small block write operation to the disk back-end typically causes a “RAID-5 write penalty”, which initiates four I/O operations to the rank by reading the old data and the old parity block before finally writing the new data and the new parity block (this is a worst-case scenario – it may take less operations dependent on the optimization of the queue of cached I/Os on a loaded system).

On modern disk systems, such as the DS8000 or DS6000, write operations are generally cached by

the storage subsystem and thus handled asynchronously, with very short write response times for the attached host systems, so that any RAID-5 write penalties are generally shielded from the users in terms of disk response time. However, with steady and heavy random write workloads, the back-end write operations to the ranks (disk drives) may still become a limiting factor in some circumstances, so that only a RAID-10 configuration will provide enough back-end disk performance at the rank level.

Consider using RAID-10 if there is a steady heavy random write workload with a write percentage larger than 35%. If this is the case, RAID-10 will provide almost twice the throughput of RAID-5 for the same number of disk drives, but will use about 40% less disk capacity. Larger drives may be used with RAID-10 to achieve the random write performance benefit, while maintaining about the same usable capacity as a RAID-5 array with the same number of disks.

RAID-5 and RAID-10 arrays basically deliver the same performance for *read* operations. However, RAID-5 outperforms RAID-10 for *sequential writes*. This is because the parity is calculated on the fly from the new data without the need to read the old parity and the old data from the back-end. RAID-10 is the better choice for workloads with a high amount of *random write* operations (more than 35% writes).

When creating arrays, the DS8000 allocates one spare for each RAID-5 array and two spares for each RAID-10 array until the following requirements are met:

- a minimum of four spares per DA pair
- a minimum of four spares of the largest capacity array site on the DA pair
- a minimum of two spares of capacity and RPM greater than or equal to the fastest array site of any given capacity on the DA pair

Depending on the distribution of the spare drives, you get different RAID array capacities. Typically the first arrays created per DA pair will have dedicated spare drives and will offer less capacity with a RAID-5 (6+P+S) or RAID-10 (2x3+2S) array configuration. If the minimum spare requirements per DA pair are met, the following arrays will have no spare drives anymore and thus will provide larger capacities with RAID-5 (7+P) or RAID-10 (2x4) array configurations.

If you create RAID-5 and RAID-10 arrays on the same DA pair you may consider starting with the configuration of the RAID-10 (2x3+2S) arrays first, as these will already reserve *two* spare drives per array - otherwise you might end up with more spare drives on the system than required, and will just waste storage capacity. However, you may also start with four RAID-5 (6+P+S) arrays first per DA pair if you want to obtain RAID-10 (2x4) arrays without spares.

When creating arrays from array sites, it may help to order them with regard to the DA pair they are attached to. The mapping of the array sites to particular DA pairs can be taken from the output of the **`lsarraysite`** command. Array sites are numbered starting with S1, S2, ... by the microcode. Arrays are numbered starting with IDs A0, A1, ... in the sequence they are created.

If you go with a homogeneous configuration (only RAID-5 or only RAID-10 arrays) you may start simply configuring the arrays in a round robin fashion across all available DA pairs by creating the first array from the first array site on the first DA pair, then the second array from the first array site on the second DA pair and so on. This sequence will also sort the arrays by array size (i.e. arrays with or without spares), creating the smaller capacity arrays with spare drives first.

Alternatively, you may also create the arrays one after another, grouped by DA pair. However, if these arrays are later configured into ranks with the same ID order, the round-robin approach across all DA pairs provides a stricter distribution of the volumes across ranks from all DA pairs within a multi-rank extent pool, as the distribution of successively created volumes across the ranks within a multi-rank extent pool also follows the ascending numerical sequence of rank IDs (with *Rotate Volumes* or *Rotate Extents* allocation methods)..

2.2.3 Rank creation

When creating *ranks* from the arrays, you simply specify the *storage type*: either FB (*fixed block* is used for Open Systems) or CKD (*count key data* is used for System z). The rank is then divided into a number of fixed sized *extents* for that storage type (FB extent = 1GB/2^30; CKD extent = 1113 cylinders). The ranks are later assigned to extent pools that provide the logical storage capacity from which the logical volumes for the attached host systems are created. Rank IDs start with R0, R1, ..., and are initially assigned in sequence. There is a one-to-one relation between a rank, an array and an array site which can be shown using the DS CLI **`lsarray -l`** command.

Each rank has an association with a DA pair based on the underlying array site from which it was created. However, a rank does not have a pre-determined or fixed relation to DS8000 server#0 or server#1 by hardware. A rank becomes associated with server#0 or server#1 only when it is assigned to an extent pool by software. Extent pools with even IDs (P0, P2, P4, ...) are primarily owned by DS8000 server#0 (rank group 0) and extent pools with odd IDs (P1, P3, P5, ...) by DS8000 server#1 (rank group 1). You should spread ranks from each DA pair equally across extent pools from both rank groups.

2.2.4 Extent pool creation and volume allocation algorithms

After creating arrays and ranks, the final step is to create *extent pools* and assign ranks to them. Each rank provides a particular number of storage *extents* of a certain storage type (fb or ckd) to an *extent pool*. An *extent pool* finally aggregates the extents from the assigned ranks and provides the logical storage capacity for the creation of logical volumes for the attached host systems. Extent pools can only contain ranks of the same storage type, either FB (fixed block - Open Systems/System i) or CKD (count key data - System z). Typically the ranks within an extent pool should have the same RAID type and the same disk drive characteristics (type, size and rpm speed), so that the storage extents in the extent pool have identical characteristics. Multiple extent pools, each with different rank characteristics, easily allow tiered storage concepts – for example, you may have extent pools with slow, large-capacity drives for backup purposes (e.g. 300GB10k) and others with high-speed, small capacity drives (e.g. 75GB15k) for performance-critical transaction applications. Furthermore, using dedicated extent pools with an appropriate number of ranks and DA pairs is a very suitable approach for isolating workloads.

You can configure *single-rank* extent pools, containing only a single rank, or *multi-rank* extent pools, containing a set of multiple ranks. Using single-rank extent pools or multi-rank extent pools in general does not have any influence on the achievable I/O performance. The performance aspect is only related to the *distribution of the volumes and I/O workloads across the available ranks* within the extent pools. In order to achieve uniform subsystem I/O performance and avoid single resources becoming bottlenecks, it is desirable to distribute volumes and workloads evenly across *all* ranks (disk spindles) and DA pairs in a balanced manner.

Single-rank extent pools provide an easy one-to-one mapping between ranks and extent pools and thus a direct association between *volumes* and *ranks* which makes performance management and control easier by *manually* distributing the volumes across the ranks. However the administrative effort increases as you have to create the volumes for each attached host system in multiple steps from each extent pool separately when distributing the volumes across ranks. Furthermore you may not only waste storage capacity if some extents remain left on each rank (because you can only create a single volume from a single extent pool, not across extent pools), but you may also be artificially restricted by this approach with regard to potential future DS8000 microcode enhancements which may exploit more of the DS8000 architecture's virtualization capabilities (like dynamic volume expansion, hot spot extent reallocation, volume striping across multiple ranks, etc.) and which may be restricted to ranks within a single extent pool only (not across extent pools).

Multi-rank extent pools not only allow the creation of large volumes that exceed the capacity of a single rank, but also still provide full control of volume placement across the ranks – using the DS CLI

command **`chrank -reserve`** to reserve all extents from a rank from being used for the creation of volumes. The DS CLI command **`chrank -release`** can be used to release a rank and make the extents available again, in case it is necessary to manually enforce a special volume allocation scheme.

However, with the latest ***Rotate Volumes*** (**`rotatevols`**) allocation algorithm or the advanced ***Rotate Extents*** (**`rotateexts`**) allocation algorithm of the DS8000, homogeneous extent pools and a reasonable concept for the volume layout, there is in most cases no need to manually select the ranks for the volumes, as the algorithm already does a good job of distributing the volumes across all ranks within an extent pool in a balanced manner. In most standard cases, manual allocation of ranks or the use of single-rank extent pools would only achieve the same result – but with much more administrative effort and a loss of flexibility with regard to potential future microcode enhancements and the ability to create volumes from extents across ranks.

Especially when using *homogeneous* extent pools (which strictly contain only *identical ranks* of the same RAID level, DDM type and capacity) together with a *standard volume size*, multi-rank extent pools offer an administrative benefit. The volumes that are created from such a multi-rank extent pool are automatically distributed across all the ranks in that extent pool in a round-robin manner by the DS8000's volume allocation algorithm, which provides an excellent balanced distribution of volumes.

Furthermore, multi-rank extent pools enable you to benefit from the flexibility that is available with the DS8000's virtualization architecture, which allows the creation of volumes across ranks from remaining extents on multiple ranks for more effective usage of the available storage capacity up to the last extents. They also ensure you that you will be ready to benefit from future DS8000 microcode enhancements, which may exploit more of the DS8000's virtualization capabilities.

With the ***Most Empty*** volume allocation algorithm, which was introduced with DS8000 code levels 6.0.500.46, each new volume was created on whichever rank in the specified extent pool happened to have the largest total number of available extents. If more than one rank in the specified extent pool had the same total number of free extents, the volume was allocated to the one with the lowest rank ID (Rx). If the required volume capacity was larger than the number of free extents on any single rank, volume allocation began on the rank with the largest total number of free extents, and continued on the next rank in ascending numerical sequence of rank IDs (Rx). All extents for a volume were on a single rank unless the volume was larger than the size of a rank or the volume started towards the end of one rank and spilled over onto another rank. If all ranks in the extent pool had the same amount of available extents, and if multiple volumes of the same capacity were created, they were allocated on different ranks in ascending rank ID sequence.

With DS8000 code level 6.2.420.21 (which was released in September 2006), the algorithm was further improved and finally replaced by the ***Rotate LUNs*** (or ***Rotate Volumes***) volume allocation algorithm, which more strictly ensures that successive LUN allocations to a multi-rank extent pool are assigned to different ranks by using an internal pointer which points to the “next rank” within the extent pool that should be used when creating the next volume. This algorithm especially improves the LUN distribution across the ranks within a multi-rank extent pool in client environments where unequal sized LUNs are allocated on a 'on demand' basis.

With the latest DS8000 code level 63.0.102.0 (released in December 2007), the new ***Rotate Extents*** volume allocation algorithm was introduced in addition to the ***Rotate LUNs*** algorithm, as further a option of the **`mkfbvol`** command (**`mkfbvol -eam rotateexts`**). This option evenly distributes the *extents* of a single volume across all the ranks within a multi-rank extent pool. The new algorithm – also known as *storage pool striping (SPS)* – provides the maximum granularity available on the DS8000 (i.e. one *extent* level = 1GB), spreading each single volume across several ranks and thus evenly balancing the workload within an extent pool. The previous volume allocation algorithms before *Rotate Volumes* and *Rotate Extents* are now referred to as ***legacy*** algorithms, as listed in the *eam* (extent allocation method) column of the output of the **`lsfbvol -l`** command.

The reason for single-rank extent pools originally arose from the initial ***Fill and Spill*** volume allocation

algorithm on the DS8000 (code levels prior to 6.0.500.46), where volumes were created on the first rank in the extent pool until all extents were used, and then volume creation continued on the next rank in the extent pool. This did *not* lead to a balanced distribution of the volumes across the ranks.

Today, multi-rank extent pools offer a good volume distribution across all ranks in a balanced manner, and deliver uniform performance. However, note that even with the latest DS8000 code level, the extents for a single volume are not spread across ranks in a multi-rank extent pool by default. You need to manually specify the **`-eam rotateexts`** option of the **`mkfbvol`** command in order to spread the extents of a volume across multiple ranks. While single-rank extent pools offer a direct relation between volume, extent pool and rank due to the one-to-one mapping of ranks to extent pools, you have to use the DS CLI commands **`showfbvol -rank`** / **`showckdvol -rank`** or **`showrank`** with multi-rank extent pools in order to determine the location of volumes on the ranks. The **`showfbvol -rank`** command lists all ranks that contribute extents to the specific volume and the **`showrank`** command reveals a list of all volumes that use extents from the specific rank.

Each extent pool is associated with an extent pool ID (P0, P1, P2, ...), and each rank can be assigned to only one extent pool. There can be as many extent pools as there are ranks. Extent pools can simply be expanded by adding more ranks to the pool. However, when assigning a rank to a specific extent pool, the affinity of this rank to a specific DS8000 server is determined. There is no predefined affinity of ranks to a storage server by hardware. All ranks assigned to *even* numbered extent pools (P0, P2, P4, ...) form *rank group 0* and are serviced (owned/managed/controlled) by DS8000 server#0. All ranks assigned to *odd* numbered extent pools (P1, P3, P5, ...) form *rank group 1* are serviced (owned/managed/controlled) by DS8000 server#1.

Although the minimum number of required extent pools is one, you should spread the available ranks and storage capacity evenly across both DS8000 servers using at least *two* extent pools (one extent pool P0 assigned to server#0 and one extent pool P1 assigned to server#1; each containing half of the available ranks and storage capacity) in order to balance workload activity across both DS8000 servers. Typically, this means assigning an equal number of ranks from each DA pair to extent pools assigned to DS8000 server#0 (rank group 0: P0, P2, P4, ...) and extent pools assigned to DS8000 server#1 (rank group 1: P1, P3, P5, ...). In environments with FB and CKD storage (Open Systems and System z) you additionally need separate extent pools for CKD and FB volumes, which would lead to a minimum of four extent pools to balance the capacity and I/O workload between the two DS8000 servers. Additional extent pools might be desirable in order to meet individual needs, such as implementing tiered storage concepts or simply separating ranks with regard to different DDM types, RAID types, clients, applications, performance or Copy Services requirements.

It is strongly recommended to spread ranks associated with a single DA pair evenly across extent pools from both DS8000 servers (i.e. extent pools with even and odd IDs), so that each DS8000 server can access the ranks via the closest DA adapter of the DA pair and utilize the full bandwidth of the DA pair. If you assign all ranks from a DA pair to extent pools managed by only one DS8000 server, you cut the maximum potential DA pair throughput by 50%, as only one DA card of the DA pair is used.

2.2.5 Volume creation and logical subsystems

The extent pools provide the storage extents that will be used for creating the volumes or LUNs for the attached host systems. A single volume can only be created from extents of the same extent pool; it cannot span multiple extent pools. However, a volume can span multiple ranks within an extent pool.

Each volume is associated with a 4-digit volume ID that has to be specified when creating the volume, for example, volume ID 1101:

Volume ID	1101	1st digit: 1	Address Group (0-F: 16 address groups on DS8000)
		1st & 2nd digits: 11	LSS ID (Logical Subsystem ID) for FB
			LCU ID (Logical Control Unit ID) for CKD

(x0-xF: 16 LSSes or LCUs per address group)

3rd & 4th digits: 01 Volume number within an LSS/LCU

(00-FF: 256 volumes per LSS/LCU)

The first digit specifies the *address group*, 0 to F, of that volume. Each *address group* can only be used by a single storage type, either FB or CKD. The first and second digit together specify the LSS ID (logical subsystem ID) for Open Systems volumes (FB) or the LCU ID (logical control unit ID) for System z volumes (CKD), providing 16 LSSs/LCUs per address group. The third and fourth digits specify the *volume number* within the LSS/LCU, 00-FF, providing 256 volumes per LSS/LCU. The volume with volume ID 1101 is the volume with *volume number* 01 of *LSS* 11 belonging to *address group* 1 (first digit). The LSS ID in the volume ID reflects the affinity of that volume to a DS8000 server. All volumes which are created from *even* numbered extent pools (P0, P2, P4, ...) have *even* LSS IDs and are managed by DS8000 server#0 (rank group 0). All volumes created from *odd* numbered extent pools (P1, P3, P5, ...) have odd LSS IDs and are managed by DS8000 server#1 (rank group 1).

Consider spreading the volumes for each attached Open Systems host or application workload evenly across multiple ranks and across both DS8000 storage servers by creating half of the volumes from *even* numbered extent pools (rank group 0 associated with server#0) and the other half of the volumes from *odd* numbered extent pools (rank group 1 associated with server#1) in order to balance I/O across both DS8000 storage servers. For a high demand of random I/O operations for a given host system or application workload, spread the volumes across a sufficient number of ranks in order to utilize a high number of disk spindles.

Before creating the volumes, the relationship between volume IDs and LSS IDs should be carefully planned. An LSS ID is related to a logical subsystem (LSS), which is a logical construct that groups 256 logical volumes. In contrast to the IBM Enterprise Storage Server (model 2105), there is no fixed binding between ranks and logical subsystems. Volumes of the same LSS can be spread across different ranks and extent pools. The LSS ID typically becomes important when using remote copy services functions such as Metro Mirror, Global Mirror or Global Copy, which operate at LSS level, especially in conjunction with establishing PPRC paths and consistency groups.

You should consider assigning volumes that belong to the same application (from a single host system or a group of host systems) to the same LSS, in order to be able to make easy use of the advanced copy services functions when required - even if you do not intend to use these functions at the moment. As application workloads should typically be spread across extent pools from both DS8000 servers, you need to use a minimum of two LSSes per application, as even numbered extent pools only allow the creation of volumes with even LSS IDs and odd numbered extent pools only allow volumes with odd LSS IDs. The LSS ID which is part of the volume ID (first two digits) also reflects the affinity to DS8000 server#0 (even LSS IDs) or DS8000 server#1 (odd LSS IDs).

Using specific numbering schemes for the volumes IDs with regard to the location of the volumes on the ranks or extent pools can further help system administrators identify independent volumes from different ranks, as the volume ID is transparent to the host system. Identifying independent volumes from different ranks on the host system might be helpful if a physical separation of certain application data structures on different physical disks is desired (e.g. separation of database table spaces and logs).

2.2.6 Volume assignment to host systems

In order to assign volumes to the attached host systems, these volumes need to be grouped in a *volume group*. A volume group can be assigned to multiple host connections, and each host connection is specified by the WWPN of the host's FC port. A set of host connections from the same host system is called a *host attachment*. Each host connection can only be assigned to a single volume group. You cannot assign the same host connection to multiple volume groups, but the same volume group can be assigned to multiple host connections. In order to share volumes between multiple host systems, the most convenient way would be to create a separate volume group for each

host system and assign the shared volumes to each of the individual volume groups as required, because a single volume can be assigned to multiple volume groups. Only if a group of host systems shares exactly the same set of volumes and there is no need to assign additional non-shared volumes independently to particular hosts of this group, can you consider using a single shared volume group for all host systems in order to simplify management.

Each host system should have a minimum of two host connections to HA cards in different enclosures of the DS8000 for availability. Try to spread the host connections across different HA cards from different I/O enclosures, buses and even RIO-G loops if available. Ideally use at least one HA card in one *left* (even numbered) and in one *right* (odd numbered) I/O enclosure so that there is a shortest path via the RIO-G loop to either DS8000 server for a good balance of the I/O requests to both rank groups. All host connections should be spread and balanced across all available I/O enclosures, enclosure buses and – if available – RIO-G loops for best performance.

The four storage unit I/O ports on each HA card that are used to connect to the host systems can be set independently to support FC-AL, SCSI-FCP and FICON protocol. Typically the SCSI-FCP setting will be used for a point-to-point or switched fabric topology. When using ports for remote copy connections (PPRC paths) between DS8000, DS6000 and ESS storage units, you also need to configure these ports for SCSI-FCP. Generally it is recommended to use dedicated ports for PPRC paths.

2.3 Basic configuration concepts

There are three major principles for achieving a logical configuration on a DS8000 subsystem with regard to optimal I/O performance for the given workloads:

- Workload *isolation*,
- Workload *resource sharing* and
- Workload *spreading*.

Here we will give a brief introduction to these basic concepts.

2.3.1 Workload isolation

Workload isolation requires dedicating a subset of hardware resources to the I/O workload of a given application. For example, a set of certain ranks with all their disk drives may be dedicated to an isolated workload. Also certain I/O ports for the host connections may be set aside to be used by an isolated workload only. Logical volumes and host connections for the workload are isolated to the dedicated resources such as ranks and DA pairs (for volumes/disk capacity) or I/O ports and HA cards (for host connections). Using DS8300 LPAR capabilities (DS8300 model 9B2), you can not only dedicate a set of DDMs, ranks, DA and HA pairs, but also DS8000 processors and cache to a certain workload within that partition.

The isolated workload may be a very important business- or performance-critical application. In that case, workload isolation is used to simply "*protect the loved ones*" and provide a consistent response time by removing resource contention with other less important workloads. However, the maximum performance possible for the workload is limited to the subset of hardware resources that is dedicated to it. It will not be able to achieve the maximum performance potential of the hardware if it is only allowed to use a subset of the hardware resources.

Conversely, the isolated workload may also be less business-critical, but might make heavy I/O demands that would cause severe contention with other, more important workloads. In this case, the workload may be "quarantined" to protect other workloads (the loved ones). Either a "loved" workload or a "badly behaving" workload may be a good candidate for isolation.

Workload isolation is recommended if a workload will tend to consume 100% of the resources it is allowed to access. For the DS8000, workload disk capacity may be isolated at the rank level and/or

the DA level. Heavy random workloads tend to overrun rank capacity and stress the disks, so rank level isolation may be appropriate for those workloads. Heavy large block-size sequential workloads tend to over-utilize the Device Adapters, so DA level isolation may be indicated for these workloads.

2.3.2 Workload resource sharing

Workload resource sharing refers to more than one workload sharing a set of hardware resources such as ranks or DAs (for disk capacity), or I/O ports or host adapters (for host connections). Logical volumes and host connections are allocated to the shared hardware resources. Workload resource sharing usually means a larger set of hardware resources is potentially available to a workload, so the potential performance is increased. If the workloads sharing the resources do not experience contention with each other, they may experience higher performance than they would achieve by using a smaller set of dedicated resources.

The ultimate example of this would be sharing all hardware resources of a given DS8000 storage unit. In this case, if a workload peaks at a time when the other workloads are not driving I/O, the 'peak' workload may be able to take advantage of all the hardware resources of the whole storage subsystem. Resource sharing is a good approach when workload information is not available, with workloads that do not try to consume all the hardware resources available, or with workloads that show peaks at different times.

2.3.3 Workload spreading

Workload spreading is the most important principle of performance optimization, and it applies to both *isolated workloads* and *resource-sharing workloads*. It simply means using *all* available resources of the storage subsystem in a balanced manner by spreading each workload evenly across all available resources that are dedicated to that workload (either to an isolated workload or resource-sharing workloads). Workload disk capacity spreading is done by allocating logical volumes evenly across ranks, DS8000 servers (i.e. both rank groups), and DA pairs, in order to achieve a balanced utilization of back-end resources. Workload host connection spreading means allocating host connections evenly across I/O ports, host adapters (HA cards), I/O enclosures, I/O enclosure buses and even RIO-G loops so as to achieve a balanced utilization of front-end resources. Using host-level striping and multi-pathing software along with workload spreading will further contribute to optimizing performance.

2.4 Simplified balanced configuration approach: share & spread

As each installation of a storage subsystem with a large amount of consolidated storage capacity needs to meet various client needs and requirements or even restrictions, there is no general cook-book-like set of step-by-step rules available about how to logically configure such a subsystem for best performance. Each logical configuration is tailored to specific requirements, not only due to the differences in the installed DS8000 hardware (DS8100 or DS8300, RIO-G loops, number of DA and HA cards, number and type of disk drives) but also due to the different application requirements (capacity and capacity growth, volume sizes, service level agreements, performance needs, use of advanced functions such as remote copy or FlashCopy, etc.) that need to be met. This can lead to logical configurations that dedicate some resources to particular applications (*workload isolation*) and share other resources for other applications (*workload sharing*). So typically the principles of *workload isolation*, *workload resource sharing* and *workload spreading* will all be applied for the logical configuration.

Here we will simply outline the idea of a *balanced logical configuration* of a DS8000 subsystem when strictly sharing *all* resources of a DS8000 subsystem evenly across all applications with a homogeneous DS8000 hardware and - ideally - also with a *standard volume size*. This means basically applying the *workload resource sharing* principle together with the *workload spreading* concept onto a whole DS8000 subsystem which then simply provides an easy-to-administer *building block* for storage capacity in a large computing environment

Generally when planning a logical configuration for a new DS8000 storage subsystem, you should evaluate the *capacity* and *I/O performance requirements* of each workload that is placed on the new subsystem very carefully. You also need to take into account the *business severity*, *service level agreements* and especially the particular *workload characteristics* of each of these workloads in order to finally decide which application workloads can *share* resources (such as ranks, DA pairs and HAs) and which need to be *isolated* from others. These initial considerations already will lead to a minimum number of required *extent pools*. Workload isolation is primarily achieved by using dedicated extent pools for these workloads with an appropriate number of ranks and DA pairs. Remaining extent pools may be used for application workloads that can share resources. After having identified *shared* and *isolated* workloads you can start with the logical configuration and *spread* these workloads across all the available resources in the DS8000 subsystem in a balanced manner. This will usually guarantee an optimal I/O performance – meeting application needs and avoiding potential hot spots such as saturated ranks, DA pairs or HAs.

However, this approach to achieving an optimal layout for best performance requires a certain amount of time and work in order to analyze and understand each of the application workloads in advance. Furthermore, steady performance monitoring of the new DS8000 subsystem should be in place, e.g. using *IBM TotalStorage Productivity Center (TPC) for Disk* to oversee the current utilization of the machine's resources as the overall I/O workloads grow. This monitoring will easily identify more and less highly utilized resources (such as ranks, DAs, HAs), and will thus help to manage a balanced workload distribution across the whole subsystem when selecting resources for new additional workloads.

In large computing environments, however, with separate departments for managing servers, applications and storage, there is often *little or no knowledge available about the real workload characteristics* when storage capacity is requested. Without knowledge about the particular workload characteristics and performance requirements, you will not have enough information to plan an optimal layout in advance. Instead, you will need to rely on a more rigid and generic approach that will help to achieve a balanced configuration and utilization of the storage subsystem resources without taking individual workload characteristics into account.

In this situation, the storage subsystem is simply taken as a homogeneous *building block* for providing a certain amount of storage capacity with uniform performance characteristics, which can scale up to the subsystem's peak capabilities by balancing the workload evenly across all internal resources and thus preventing single resources from becoming early bottlenecks and limiting the overall performance.

In this chapter we want to introduce such a concept for a logical configuration, based on a *share and spread everything* approach, and explain the ideas behind it. Without any knowledge of each application's workload characteristics, this approach will help to achieve balanced configuration that in most cases will satisfy the needs of almost 80% of the standard applications.

Of course, there are always special applications with outstanding I/O workload characteristics and/or critical service level agreements that will require a dedicated approach with an in-depth workload analysis and a careful planning of the subsystem resources required. These applications will always need special attention and additional effort, such as performance planning and monitoring, even after they have been implemented on the new subsystem, to watch the growth of the I/O workload and react in time when additional resources are needed.

Although these applications will always require special attention and effort, by following the strict approach in this chapter you will most probably expend less effort and still gain good performance results for around 80% of the typical workloads - giving you more time to focus on taking care of the critical ones that require special attention.

With a storage subsystem as a *building block* in a larger fabric, you need to achieve an overall balanced utilization of all available subsystem resources without taking the individual workload characteristics of each request for storage capacity into account. Here, without any information on the particular workload characteristics behind each request for storage capacity, the principle of strictly

sharing all subsystem resources and thus *spreading* each application's workload evenly across all of these resources (*share and spread everything*) is a reasonable approach. Still, monitoring at least the overall utilization and performance of such a storage building block subsystem (using, for example, TPC for Disk) would be a good practice in order to keep a careful eye on workload growth and react to increasing workload demands in time.

To achieve a balanced configuration and achieve sufficient performance with the building block approach, you not only need a good and homogeneous *hardware base*, but also an appropriate *logical configuration* concept. The *hardware base* should in general meet your basic capacity and performance requirements as long as you make the right choice of DS8000 model, processor power and memory/cache size, as well as the right disk and RAID type.

The *logical configuration* should guarantee a balanced workload distribution of all applications across all available hardware resources within the storage subsystem, with regard to:

- DS8000 front-end resources
  - I/O ports
  - HA cards (host adapters)
  - I/O enclosures
- DS8000 back-end resources
  - ranks (disk drive modules)
  - DA pairs (device adapters)
- DS8000 processor complexes (DS8000 storage servers: server#0 and server#1)

2.4.1 Hardware base for a storage building block concept

Choosing the right, uniformly equipped *hardware base* for the storage *building block* is crucial to the overall fabric approach that is introduced in this paper. If you choose the wrong hardware base for the subsystem as building block – for example, one with insufficient processing power and cache or simply with the wrong disk or RAID type – you may not achieve sufficient I/O performance. In this case you would probably saturate your whole disk subsystem long before you even could use the maximum available storage capacity – even with a perfectly balanced logical configuration. Depending on the average workload performance requirements, the intended storage capacity per subsystem and future capacity growth strategies, you first need to choose your hardware base for the subsystem. You must not only choose between the DS8100 and DS8300 *models* with regard to available storage capacity, capabilities and footprint – you also need to decide which *cache* size you need and, most critically, what kind of *disk drives*.

Furthermore, using a *building block* concept for a large and steadily growing storage environment requires a *uniform storage subsystem as hardware base, together with a balanced logical configuration concept* that provides the building block for the storage capacity demands. This implies that adding new storage capacity to the environment simply means adding a new building block and thus installing a new, uniformly equipped subsystem. Storage growth here is basically satisfied by adding a new uniformly configured subsystem as building block, not by adding disks to the existing storage subsystems. Simply adding disks to the existing subsystems would not follow a building block approach, which requires a uniformly configured subsystem with a balanced logical configuration. Adding capacity to an almost fully utilized subsystem would only lead to performance bottlenecks, because at this point most of the subsystem's resources would already be fully utilized and the additional capacity would be restricted to only a part of the subsystem resources – thus not allowing a strictly balanced logical configuration any more.

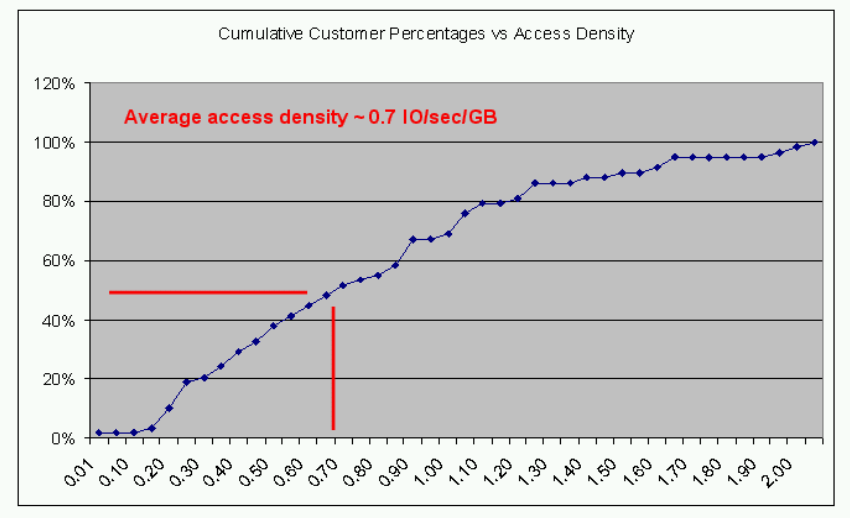
To start with the question of the adequate amount of *cache* per subsystem, there is unfortunately no quick answer because it depends on the nature of the workload. However, it is a common approach to

base the amount of cache on the amount of disk capacity (cache-to-backstore ratio). In general, as the cache-to-backstore ratio increases, so does the hit ratio: the larger the cache, the higher the chances of data hits in cache. However, it isn't possible to predict exactly what the hit ratio increase will be, as it depends on many other workload characteristics. If existing cache hit ratio information is available for the workload this information may be used in the IBM storage subsystem sizing tool *Disk Magic*. If there is no existing cache size information to work from, then the rules of thumb in the following table may be used to determine how much system memory should be considered for the DS8000:

Useable Capacity (TB)	Minimum Recommended System Memory (GB)				
	Open and System i		z/OS		Both
	High Performance	Standard Performance	High Performance	Standard Performance	Low Access or Archival
Up to 5 TB	16 GB	16 GB	32 GB	16 GB	16 GB
5 TB to 11 TB	32 GB	16 GB	64 GB	32 GB	16 GB
11 TB to 23 TB	64 GB	32 GB	128 GB	64 GB	16 GB
23 TB to 51 TB	128 GB	64 GB	256 GB	128 GB	32 GB
51 TB to 118 TB	256 GB	128 GB	-	256 GB	64 GB
118 TB to 168 TB	-	256 GB	-	-	128 GB

Note that the 16 GB system memory option is only available for the DS8100, while the 256 GB option is only available for the DS8300. Memory sizes of 32 GB, 64 GB and 128 GB are available on either model.

Even when unable to take all the *individual* workload characteristics of each application into account, you may still be able to make a general decision about the *disk type* that should be used for the storage building block in a shared environment with regard to the overall range of applications.



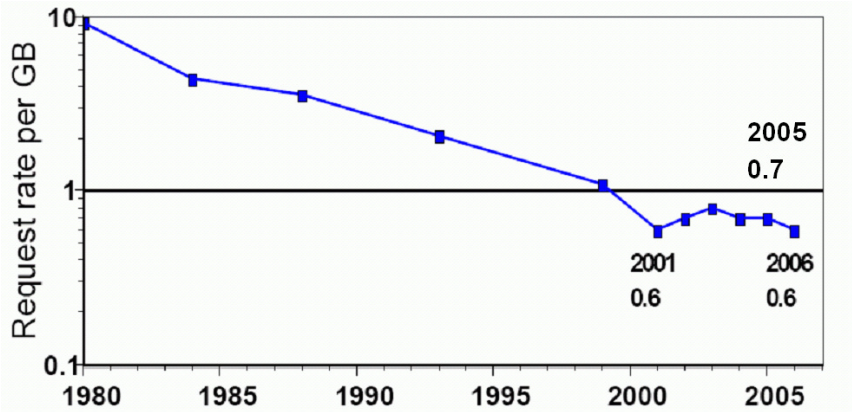
Taking a look at the *average access density* (average I/O operations per second per usable GB of storage) as seen in typical installations of storage subsystems, you can generally classify application workload demands as follows:

- *High performance* applications, which are generally characterized by access densities of about 2.0 IOps/GB and response time requirements of less than 5 ms on average,
- *Standard performance* applications, with access densities of about 0.7 IOps/GB and response time requirements of less than 10 ms
- *Low access* applications, with an average access density of 0.35 IOps/GB and no firm response time target.

Here the concept of *access density* relates the range of typical average performance needs (as seen at clients all over the world) to the amount of total usable storage in the environment.

Taking a look at the cumulative chart of the total number of clients versus the access density from 2005, you can see that an average access density of up to 0.7 IOps/GB already covers almost 50% of the clients – i.e. those that are satisfied with *standard performance* requirements. The upper limit seen for the access density in this chart is around 2.1 IOps/GB, for clients in *high performance* environments.

Still, you need to keep in mind that we are talking here about an *average* value. Variations in application workloads can be expected from day to day, and in most production environments, peak demands may occasionally result in loads of two or three times average levels. However, without detailed workload information, the goal of the building block approach is simply to offer some sort of rule-of-thumb for choosing a hardware base with a system that is likely to remain robust relative to reasonable variations in day-to-day load.



The average industry value for access density in the year 2005 was thought to be approximately 0.7 IOps/GB. Year-to-year industry data is incomplete, but the value has been decreasing as companies acquire usable storage faster than they access it.

Remember that each disk drive is only capable of processing a limited number of random I/O operations per second, so the mere *number* of disk drives used for a given amount of storage capacity finally determines the achievable random IOps performance. 15k drives offer approx. 30% more IOps performance than 10k drives (rule-of-thumb for random IOps calculation: 160 IOps per 15k FC drive, 120 IOps per 10k FC drive). Large capacity FATA disk drives are only intended for environments with fixed content, data archival, reference data, or near-line applications that require large amounts of data at low cost and do not require drive duty cycles greater than 20%. In general, FATA drives are not an option for the hardware base in a storage fabric approach with typical business applications (database, mail, web, file server) and transaction-oriented workloads.

As shown in the previous charts, access density can be used to determine whether a customer should consider fast, smaller hard drives or large, slower hard drives. In general, if the storage is highly stressed with an access density around 2.0 IOps/GB, fast, small capacity drives should be considered to deliver sufficient transaction performance. On the other hand, with access densities below 0.5 IOps/GB, large capacity drives can be considered. However, with workload characteristics available, IBM's storage sizing tool Disk Magic should be used to model the specific hardware configuration.

Taking a look at the charts, it is possible to cover almost 70% of all client application loads with an average I/O access density of no more than 1.0 IOps per usable GB. So with an average calculation of 160 random IOps per 15k rpm FC disk drive you may consider going with 146GB15k FC drives to meet these needs. 146GB10k drives may be suitable for access densities below 0.8. If you know your access density and performance requirements are considerably below this (around 0.5 IOps/GB), you

may consider going with 300GB15k drives. Lower requirements may even lead you to consider 10k drives. However, if you know you are mostly working with performance-critical applications with high access densities, you should consider 73GB15k drives which should meet even the highest performance needs.

For typical IT environments which need to meet the requirements of a wide variety of different applications (for example, online transaction databases, mail, web and file services together with other less critical applications), the 146GB15k drives may be a good choice for the hardware base of the storage subsystem building block, offering good performance for access densities up to 1.1 (which, according to the access density charts would cover almost 70-80% of the standard client needs). Typically you find a high read ratio in such environments with read:write ratios above 70:30, so that RAID-5 may be considered. Only in environments with an overall write percentage above 40% should you consider a RAID-10 configuration at the cost of available capacity per subsystem.

In order to be able to achieve a balanced logical configuration for the chosen hardware base, it is important that the subsystem itself has a balanced hardware configuration with regard to hardware resources like HA cards, DA pairs and disk drives. As the full box bandwidth scales with the number of DA pairs, you also need to balance the HA card bandwidth with the available DA card bandwidth according to your throughput requirements.

Further note that a *fully* equipped DS8000 subsystem shows an imbalance with regard to the number of ranks per DA pair. Typically the number of ranks scales with the number of DA pairs, starting with eight ranks per DA pair before the next DA pair is used (four ranks per DA pair is a separate ordering option for environments with a high throughput demand but only a minimum number of required disk drives). However, in a *fully equipped DS8000 storage subsystem*, certain DA pairs (starting with DA2 and DA0) will service twice the number of ranks than other DA pairs, which will violate any concepts for a strictly balanced logical configuration across all back-end resources by introducing a potential bottleneck at the DA pair level with regard to bandwidth limitations. Although with small-block, random IOps workloads, these DA pairs may generally not become the limiting hardware resource here, they may easily become a limiting factor with large-block, throughput-dominated workloads. So with a *fully* equipped DS8000 subsystem as building block for a storage fabric in throughput-dominated environments, additional individual concepts should be applied, in order to deal with the imbalance of ranks at the DA level - perhaps using these additional ranks for specialized workloads only.

A *balanced* hardware configuration for a DS8000 subsystem consists of four DA pairs with a total of 32 ranks (256 DDMs) for a DS8100 and eight DA pairs with a total of 64 ranks (512 DDMs) for a DS8300. Here each DA pair operates eight ranks and the overall performance of the subsystem scales uniformly with the number of installed DA pairs and disk drives.

2.4.2 Balanced logical configuration concept

When an appropriate hardware base has been defined as the building block for the storage fabric, the next goal is to provide a *logical configuration concept* that will guarantee a balanced workload distribution across all available hardware resources within the storage subsystem at any time - from the beginning, when only part of the available storage capacity is used, up to the end, when almost all capacity of the subsystem is allocated.

In this section, we will provide some basic ideas for strictly applying a *spread and share everything* approach for the logical configuration, which is quite a reasonable approach for a DS8000 subsystem used as a building block in a large storage fabric where there is no chance of detailed workload planning in advance.

To achieve a balanced utilization of all resources of the DS8000 storage subsystem, you need to distribute the I/O workloads evenly across the subsystem's *back-end resources*

- ranks (disk drive modules)
- DA pairs (device adapters)

and the *front-end resources*

- I/O ports
- HA cards (host adapters)
- I/O enclosures

as well as both DS8000 processor complexes (storage servers).

Before starting to configure the volume layout, we need to configure the disk array sites according to the *RAID type* (RAID-5 or RAID-10) and the *storage type* (fixed block or count key data). Here, in this example, we will decide to go with RAID-5 for a pure Open Systems environment (fixed block) which in general would be a good choice in such environments for a storage fabric approach with a homogeneously configured subsystem.

Although the array site IDs, array IDs and rank IDs are not essential for the volume layout, it may help to order them by *DA pair* and *array size* when performing the configuration, as the sequence of steps finally determines the numbering scheme of the arrays and ranks. Creating the arrays in a round-robin fashion from each DA pair (e.g. creating the first array from the first array site of the first DA pair and creating the second array from the first array of the second DA pair and so on) would sort the arrays easily by array size (arrays with spares are created first from each DA pair) with the array ID sequence cycling through all available DA pairs. If the ranks are created in the same sequence from the arrays, the rank ID sequence will also cycle across all DA pairs in a round-robin fashion. This may enhance a stricter distribution of the volumes across ranks from all DA pairs within multi-rank extent pools, as the creation of successive volumes within an extent pool will also follow the sequence of rank IDs.

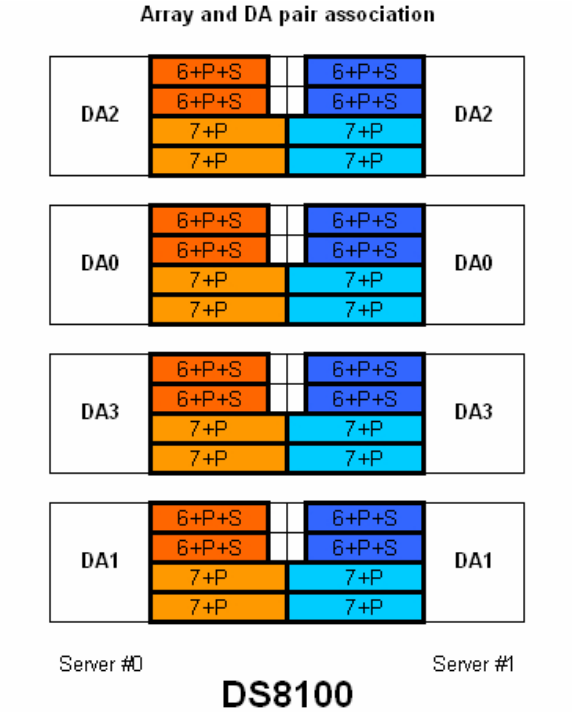
The final mapping of the arrays and ranks to the DA pairs can be taken from the **`lsarray -l`** command:

```
dscli> lsarray -l
Date/Time: May 11, 2007 8:54:58 AM CEST IBM DSCLI Version: 5.2.410.299 DS: IBM.2107-75L2321
Array State    Data   RAIDtype  arsite Rank DA Pair DDMcap (10^9B) diskclass
=============================================================================
A0    Assigned Normal 5 (6+P+S) S1     R0   0                146.0 ENT
A1    Assigned Normal 5 (6+P+S) S9     R1   1                146.0 ENT
A2    Assigned Normal 5 (6+P+S) S17    R2   2                146.0 ENT
A3    Assigned Normal 5 (6+P+S) S25    R3   3                146.0 ENT
A4    Assigned Normal 5 (6+P+S) S2     R4   0                146.0 ENT
A5    Assigned Normal 5 (6+P+S) S10    R5   1                146.0 ENT
A6    Assigned Normal 5 (6+P+S) S18    R6   2                146.0 ENT
A7    Assigned Normal 5 (6+P+S) S26    R7   3                146.0 ENT
A8    Assigned Normal 5 (6+P+S) S3     R8   0                146.0 ENT
A9    Assigned Normal 5 (6+P+S) S11    R9   1                146.0 ENT
A10   Assigned Normal 5 (6+P+S) S19    R10  2                146.0 ENT
A11   Assigned Normal 5 (6+P+S) S27    R11  3                146.0 ENT
A12   Assigned Normal 5 (6+P+S) S4     R12  0                146.0 ENT
A13   Assigned Normal 5 (6+P+S) S12    R13  1                146.0 ENT
A14   Assigned Normal 5 (6+P+S) S20    R14  2                146.0 ENT
A15   Assigned Normal 5 (6+P+S) S28    R15  3                146.0 ENT
A16   Assigned Normal 5 (7+P)   S5     R16  0                146.0 ENT
A17   Assigned Normal 5 (7+P)   S13    R17  1                146.0 ENT
A18   Assigned Normal 5 (7+P)   S21    R18  2                146.0 ENT
A19   Assigned Normal 5 (7+P)   S29    R19  3                146.0 ENT
A20   Assigned Normal 5 (7+P)   S6     R20  0                146.0 ENT
A21   Assigned Normal 5 (7+P)   S14    R21  1                146.0 ENT
A22   Assigned Normal 5 (7+P)   S22    R22  2                146.0 ENT
A23   Assigned Normal 5 (7+P)   S30    R23  3                146.0 ENT
A24   Assigned Normal 5 (7+P)   S7     R24  0                146.0 ENT
A25   Assigned Normal 5 (7+P)   S15    R25  1                146.0 ENT
A26   Assigned Normal 5 (7+P)   S23    R26  2                146.0 ENT
A27   Assigned Normal 5 (7+P)   S31    R27  3                146.0 ENT
A28   Assigned Normal 5 (7+P)   S8     R28  0                146.0 ENT
A29   Assigned Normal 5 (7+P)   S16    R29  1                146.0 ENT
A30   Assigned Normal 5 (7+P)   S24    R30  2                146.0 ENT
A31   Assigned Normal 5 (7+P)   S32    R31  3                146.0 ENT
```

Note that depending on the installed hardware resources in the DS8000 storage subsystem, you may have different numbers of DA pairs and even different numbers of ranks per DA pair.

Furthermore, note that even with a uniform hardware configuration (same disk drives, same RAID type) you will have ranks of different sizes due to the *spare distribution*. However this is generally not a problem in achieving a balanced logical configuration, as the larger 7+P arrays also contain one more *active* disk drive than the smaller 6+P+S arrays, which means that the larger arrays are also capable of a slightly higher I/O workload in relation to their higher capacity.

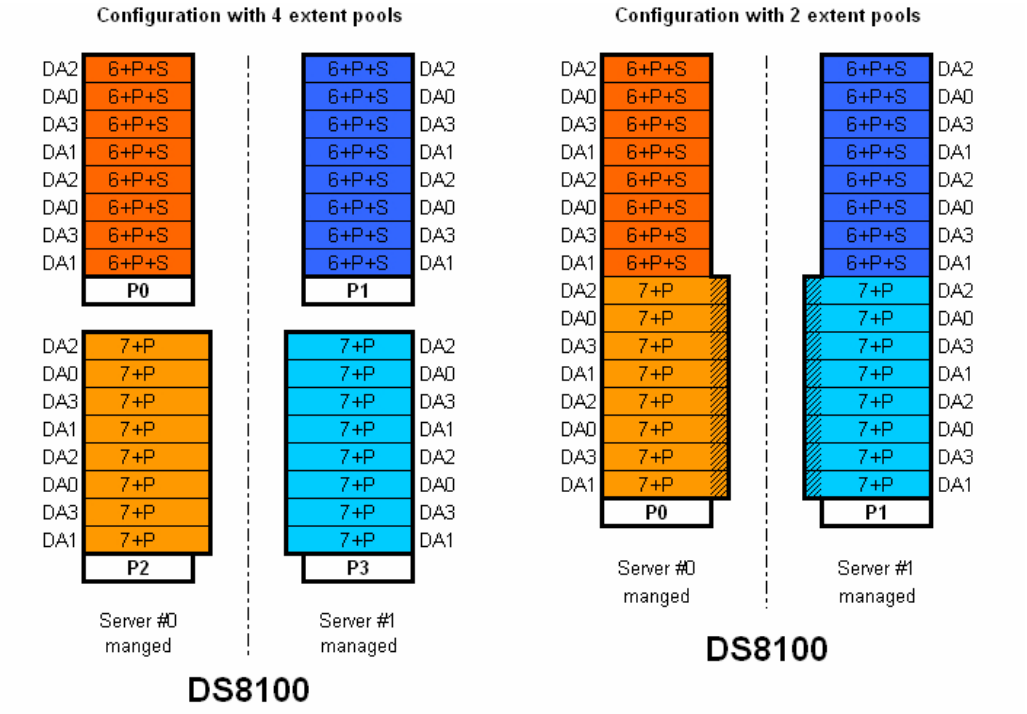
A typical DS8100 with a balanced hardware configuration would have four DA pairs and eight ranks per DA pair, as seen in the **`lsarray`** example above and the diagram below:



The key to a balanced workload distribution across the *back-end resources* (DA pairs, ranks) is a proper *volume layout*. The goal here simply is to *distribute the volumes of each workload evenly across all available ranks and DA pairs*. This may be achieved with various extent pool configurations. The extent pools themselves have no influence on the achievable performance. As the extent pools have an affinity to either storage server#0 (rank group 0 / even numbered extent pools: P0, P2, P4, ...) or storage server#1 (rank group 1 / odd numbered extent pools: P1, P3, P5, ...) care needs to be taken to assign half of the ranks to even-numbered extent pools and the other half to odd-numbered extent pools, so that the overall capacity (and therefore the workload) is balanced between both DS8000 storage servers. Ranks with or without spares should also be balanced across both DS8000 servers. As ranks have an association to a DA pair, you also need to spread the ranks from each DA pair evenly across both DS8000 servers (both rank groups) to balance the workload across both DA cards from each DA pair.

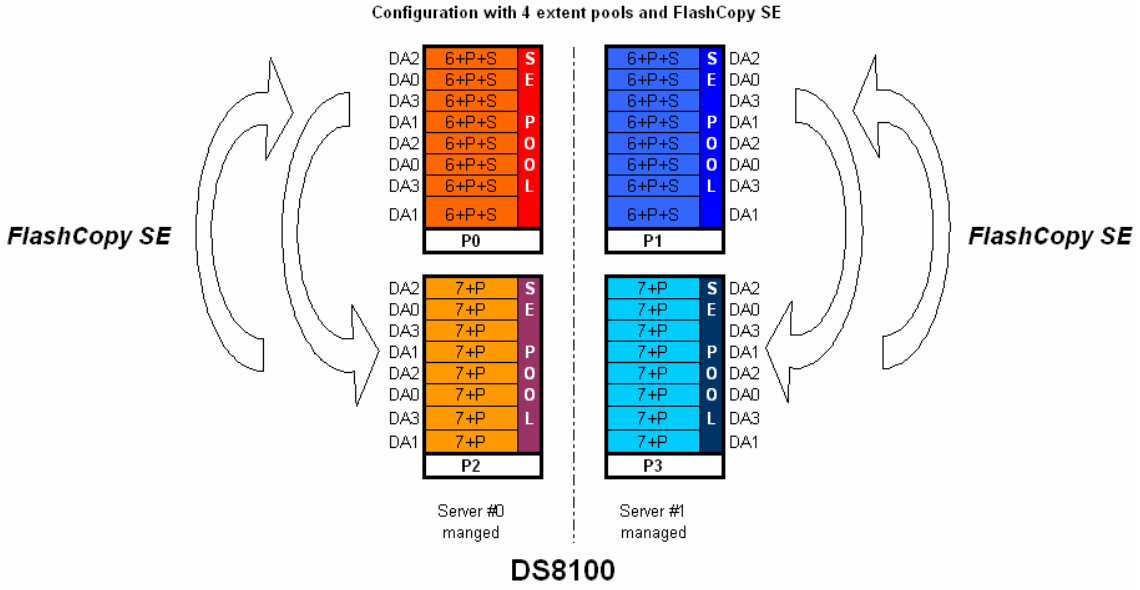
Configuring the extent pools in a balanced manner simply means distributing the available ranks evenly across both DS8000 servers. To achieve a balanced logical configuration within a storage fabric with a DS8000 as a generic building block, consider configuring only a minimum number of homogeneous multi-rank extent pools. These *homogeneous multi-rank extent pools* together with a *standard volume size* would not only offer a good volume distribution across the ranks using the DS8000 internal extent allocation algorithm (thus reducing administrative effort), but would also allow a more effective usage of all of the available storage capacity by not wasting remaining extents on isolated ranks in single-rank extent pools (volumes can not be created across extent pools). And, last but not least, this approach would also preserve all of the flexibility offered by the DS8000's advanced virtualization architecture, allowing the customer to benefit from potential future microcode features and enhancements that may exploit more of the DS8000's virtualization capabilities.

In general, with two storage servers, the minimum recommended number of extent pools would be *two* (P0, P1) on a uniformly equipped subsystem in order to spread the workload evenly across both DS8000 storage servers. However, with regard to the different array sizes due to the distribution of spare drives (i.e. 6+P+S and 7+P arrays) you might also consider using *four* strict homogeneous extent pools (P0, P1, P2, P3), where each extent pool has arrays of the same capacity. You should also spread arrays from all DA pairs evenly across all extent pools.



Homogeneous extent pools with ranks of the same capacity would offer the best basis for the DS8000's volume allocation algorithms in order to achieve a strictly balanced distribution of the volumes across all ranks within an extent pool especially with a standard volume size. However, you still need to manually distribute the volumes of each application workload evenly across all extent pools.

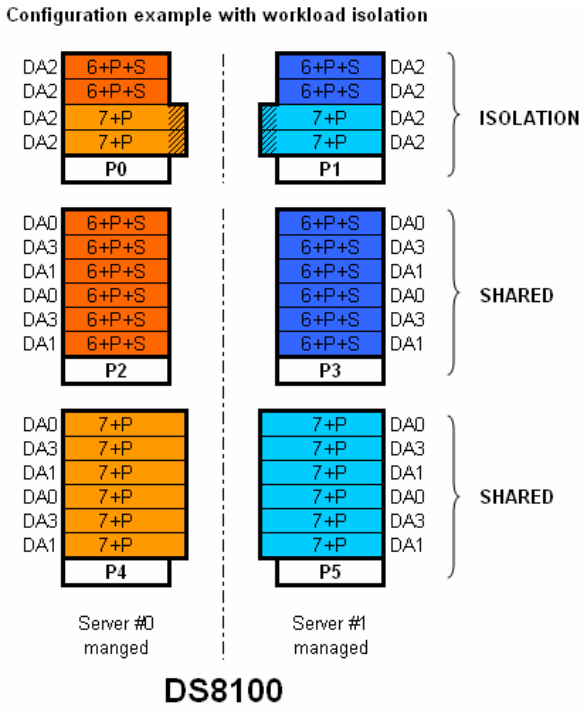
Two large extent pools might be a good choice when using the *legacy* or *Rotate Volumes* extent allocation method with dedicated volumes on each rank and a standard volume size. A minimum of *four* extent pools should be considered when using the *Rotate Extents* extent allocation method, because with *storage pool striping* and the distribution of each volume's extents across all ranks within an extent pool it is recommended *not to exceed eight ranks per multi-rank extent pool*.



Furthermore, if FlashCopy SE (space efficient FlashCopy) is used you also should consider a minimum of *four* extent pools with *two* extent pools per rank group. As the FlashCopy SE repository for

the space efficient target volumes is distributed across all available ranks within the extent pool, it is recommended to distribute the source and target volumes across *different extent pools* from the *same DS8000 storage server* (i.e. the same rank group) for best FlashCopy performance. Each extent pool should have FlashCopy source volumes as well as repository space for space efficient FlashCopy target volumes.

Remember, the concept of a balanced logical configuration that is presented in this chapter is based on the idea of sharing *all* storage subsystem resources evenly with a uniformly equipped and configured storage subsystem used as a *building block* in a storage fabric. In principle you also could combine *workload isolation* and *workload sharing* on the same storage subsystem by dedicating a number of *extent pools* to workloads that need to be isolated. An example of such a configuration is given in the figure below with workload isolation on rank and DA pair level. But note that in this example the shared extent pools would have fewer resources (DA pairs and ranks) than on subsystems without workload isolation, so it slightly violates the generic approach for a uniformly configured storage subsystem which is used as building block throughout the whole storage fabric.



Basically, when creating a set of multi-rank extent pools, you should also consider potential growth strategies if additional ranks are to be added to the subsystem later on. In general it would not be a good idea to increase the capacity of a full multi-rank extent pool by simply adding a single rank. There is no automatic reallocation of the extents that have already been used, so that all new volumes would only be placed on the newly added rank, introducing a potential performance bottleneck or 'hot spot'. For capacity growth strategies it might be convenient to work with extent pools that are built from four to eight ranks only, and add capacity by adding new *extent pools* with an appropriate number of ranks. However, with regard to a storage fabric approach that uses the storage subsystem as a building block to satisfy storage demands in a large and steadily growing environment, adding new storage capacity to the environment simply means adding a new building block and thus installing a new, uniformly equipped and configured storage subsystem.

The *two extent pool configuration* might be convenient if FlashCopy SE and storage pool striping (*Rotate Volumes*) is *not* used. Here simply distributing the ranks from each DA pair evenly across extent pools P0 and P1 offers all of the flexibility of the subsystem's virtualization architecture together with a simple and robust concept for the logical configuration, as well as ease of use. For the DS8000's volume allocation algorithms to perfectly balance the volumes across all the ranks within

two *homogeneous* extent pools, you even may consider creating a set of dummy volumes on the large arrays initially after preparing the extent pools, so that finally all ranks start with the same available capacity when the first logical volumes for the attached host systems are created. This might especially be necessary when still working with the older DS8000 *Most Empty* (legacy) volume allocation algorithm.

You can achieve a *homogeneous* two extent pool configuration by following the steps below after the initial creation of the two extent pools:

1. Identify the number of available extents on the ranks and the assignment of the ranks to the extent pools from the output of the **`lsrank -l`** command. Calculate the amount of extents that make up the difference between the small and large ranks.
2. Use the DS CLI command **`chrank -reserve`** against all smaller (6+P+S) ranks in order to reserve all extents on these ranks within each extent pool from being used for the creation of the dummy volumes in the next step.
3. Now, create a number of dummy volumes using the **`mkfbvol`** command from each extent pool according to the number of large arrays in the extent pool and the additional capacity of these ranks in comparison to the smaller arrays. For example, with 16 ranks in extent pool P0 as shown in the diagram above you have eight small capacity (6+P+S) ranks and eight (7+P) large capacity ranks. With 73GB disk drives this would give 388 extents per (6+P+S) rank and 452 extents per (7+P) rank. In this case you would need to create 8 dummy volumes of 64 extents in size (volume size = 64GB, binary) per extent pool using two **`mkfbvol`** commands

   ```
   # mkfbvol -extpool P0 -cap 64 -type ds -name dummy_vol ee00-ee07

   # mkfbvol -extpool P1 -cap 64 -type ds -name dummy_vol ef00-ef07
   ```

   Here we use LSS ee with volume IDs ee00-ee07 for P0 (even extent pool) dummy volumes and LSS ef with volume IDs ef00-ef07 for P1 dummy volumes. The volume allocation algorithm will automatically distribute the volumes across the ranks.
4. Use the DS CLI command **`chrank -release`** against all smaller (6+P+S) ranks in order to release all extents on these ranks again so that finally all ranks in the extent pools are available for the creation of volumes for the attached host systems.

Now we even have created a *homogeneous two extent pool* configuration with ranks of equal size. The dummy volumes can be removed when the last amount of storage capacity needs to be allocated. However, here you need to keep in mind that the volumes that will be created from these final extents on the large (7+P) arrays are distributed only across *half* of the ranks in the extent pool, so you should consider using this capacity only for applications with lower I/O demands.

Of course, you can also apply a similar procedure to a *four* extent pool configuration with 6+P+S and 7+P ranks mixed in each extent pool if you prefer *four* identical extent pools with exactly the same amount of storage capacity. However, simply using separate extent pools for 6+P+S and 7+P ranks reduces the administration effort considerably. But still, you need to be aware that, when the capacity of the 6+P+S pools is exceeded, all additional volumes can only be created from the 7+P extent pools and thus utilize only half of the subsystem's back-end resources. Here you should also consider using these remaining extents only for applications with lower I/O demands.

When creating a set of volumes in homogeneous extent pools such as these, the volumes will be distributed evenly across all ranks and DA pairs within the extent pool in a round-robin fashion by the DS8000's volume allocation algorithm. With the default ***Rotate Volumes*** (**`rotatevols`**) algorithm, each *volume* will be placed on a single rank, with a successive distribution of the volumes across all ranks within the multi-rank extent pool in a round robin fashion. With the optional ***Rotate Extents*** (**`rotateexts`**) algorithm, the *extents* of each single volume will be spread across all ranks within the extent pool (provided the size of the volume in extents is equal to or greater than the number of ranks in the extent pool). Nevertheless, the maximum granularity for distributing a single volume (and thus

balancing the workload evenly across the ranks within a multi-rank extent pool) can be achieved simply using the *Rotate Extents* volume allocation algorithm (**`mkfbvol -eam rotateexts`**).

After creating the extent pools and evenly distributing the back-end resources (DA pairs and ranks) across both DS8000 storage servers, you can start with the creation of *host volumes* from these extent pools. When creating the host volumes, it is important to follow a strict *volume layout* scheme that spreads the volumes of each application across *all* ranks and *all* extent pools in a balanced manner, in order to achieve a balanced I/O workload distribution across the ranks, DA pairs and DS8000 storage servers.

The DS8000's volume allocation algorithms will take sufficient care of the distribution of volumes across the ranks within an extent pool. However, with individual volume sizes and volume numbers for each storage request, the achieved volume layout as obtained through the automatic algorithm may not be optimal and may not strictly follow a round robin volume distribution across all ranks for each request. The achieved distribution may still be sufficient for many less critical environments, but within a large and growing storage fabric some additional effort should be spent on implementing a more advanced *volume layout concept* that leads to a strict volume distribution across all ranks for each storage request. The intention here is simply to avoid a situation where a sequence of volumes for a single application is located on the same ranks or only a limited number of ranks, instead of being spread across all available ranks (which might occur if a mixture of different numbers of small capacity and large capacity volumes are created from an extent pool).

One way to achieve a uniform volume distribution across the ranks would be to choose a *standard volume size* throughout the storage fabric. You can choose this standard volume size based on the required granularity for average storage requests, and align it to the available capacity on the ranks, or even to the number of ranks within the extent pools (if the *Rotate Extents* algorithm is used) in order to be able to allocate most extents on the subsystem with that volume size in a balanced way without wasting capacity.

Another way to spread the workload of each storage request or application evenly across all ranks and extent pools without using a standard volume size would be to divide the capacity of each storage request by the number of available ranks and create an appropriate number of volumes if dedicated ranks for each single volume are desired, using the *Rotate Volumes* algorithm. Alternatively, when using the *Rotate Extents* algorithm, the extents of each volume should be spread *evenly* across all ranks within an extent pool, and therefore the volume size should be aligned to the number of ranks within the extent pools by choosing a volume size in GB which is a multiple of the number of ranks within the extent pool.

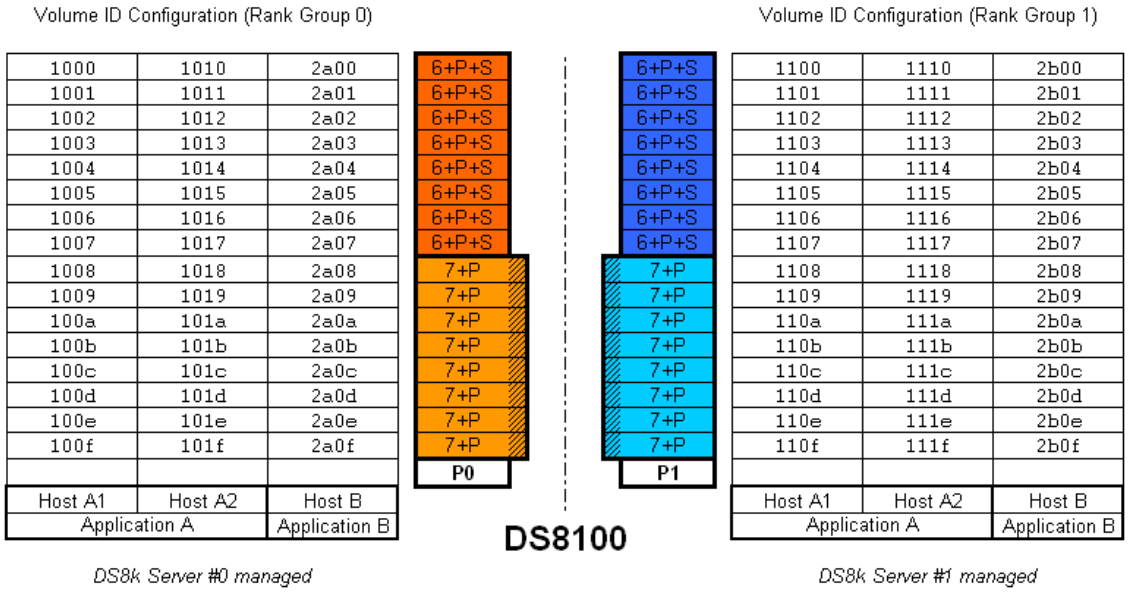
In either case, you should also consider a meaningful *numbering scheme* for the volume IDs with regard to the *specific applications* (or host systems) and the *physical volume location* on the *ranks* (when using the *Rotate Volumes* algorithm and dedicated ranks for the volumes) or *extent pools* (when using the *Rotate Extents* algorithm and spreading each volume across multiple ranks within an extent pool). Ideally, all volumes that belong to a certain application or a group of related host systems should be within the same logical subsystem (LSS). However, as the volumes should be evenly spread across both DS8000 storage servers, it would typically require at least *two* logical subsystems per application, one even LSS for the volumes managed by server#0 and one odd LSS for volumes managed by server#1 (for example, LSS 10 and LSS 11).

The assignment of LSSes to applications is essential when you plan to use advanced copy services functions, as basic parts of these functions (e.g. consistency groups, PPRC paths) are related to logical subsystems. Even if you currently do not plan to use copy services you should plan your volume layout accordingly, as this will make management easier if you need to introduce copy services at some point in the future.

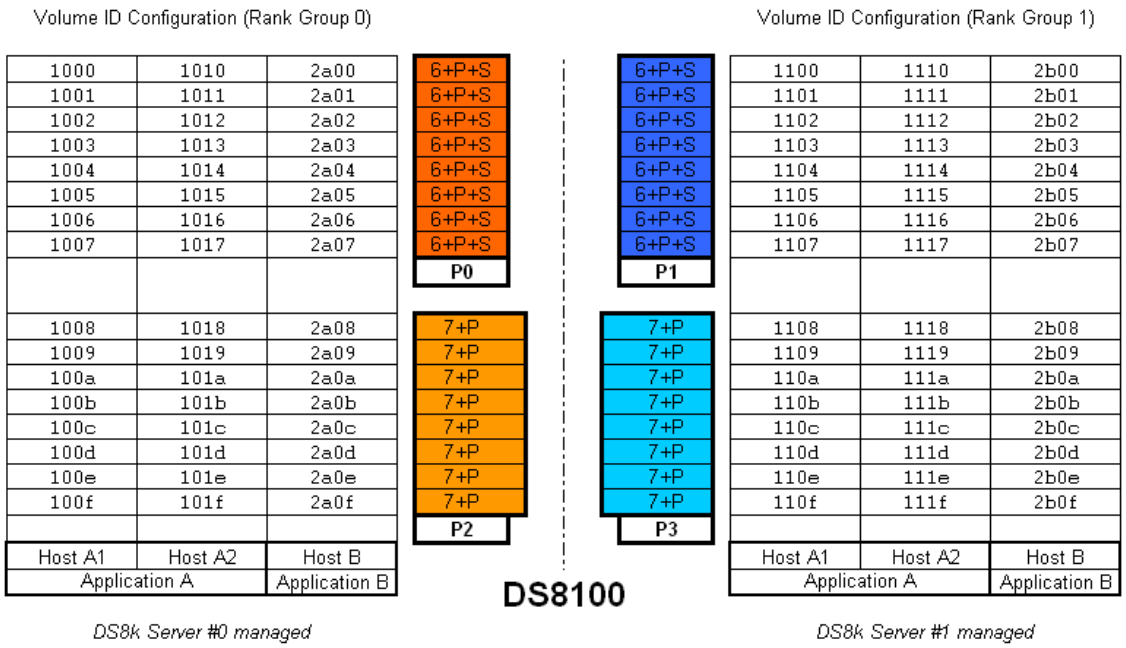
Each LSS can have 256 volumes, with volume numbers ranging from 00 to ff. You should additionally try to relate the *volume number* to the physical location of the volume on the *rank* (when using the *Rotate Volumes* algorithm) or the *extent pool* (when using the *Rotate Extents* algorithm).

Here are some suggestions for a volume ID numbering scheme when using the *Rotate Volumes* algorithm with multi-rank extent pools and volumes on dedicated ranks. For example, with 16 ranks within each extent pool P0 and P1 you might consider creating 16 equal volumes with volume IDs 1000-100f in extent pool P0 and 16 volumes with volume IDs 1100-110f in extent pool P1 for a given application that requires 32 volumes. If you need another 32 volumes for that application, assigned to a second host system of the same host group, you might consider creating another set of volumes with volume IDs 1010-101f and 1110-111f, so that the fourth digit is related to a specific rank in the appropriate extent pool. When creating the volumes in a balanced way, the volume allocation algorithm will spread the volumes across the arrays accordingly.

Volume distribution across ranks and two extent pools (using *Rotate Volumes* algorithm)



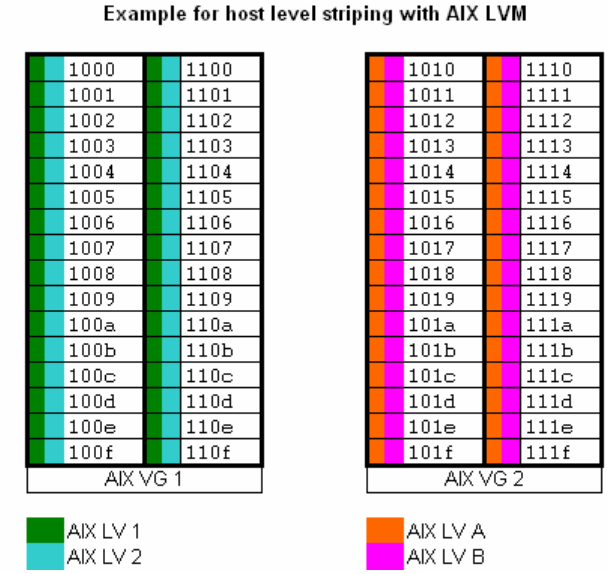
Volume distribution across ranks and four extent pools (using *Rotate Volumes* algorithm)



Relating the volume ID in some way to the physical location of a volume on a *rank* (when using the

Rotate Volumes algorithm) or *extent pool* (when using the *Rotate Extents* algorithm) may help to identify *independent* volumes on the attached host system in order to be able to tell from the volume ID whether two LUNs share the same physical disks or are located on different physical disks (ranks). This is of interest on the host system when, for example, you need to identify LUNs located on different ranks in order to separate table spaces from the database logs, or want to create a volume group that is striped across independent LUNs from different physical disks.

When the volumes of each host or application are distributed evenly across the ranks, DA pairs and storage servers using the *Rotate Volume* algorithm, you may also consider using *host level striping* on the host system in order to achieve a uniform distribution of the workload across the assigned volumes (and thus the DS8000 hardware resources). In this case, host level striping may generally be established as a basic rule for the storage fabric, so that OS logical volumes on the host systems are striped across multiple DS8000 LUNs from *different* ranks and DA pairs. For host level striping, a large granularity stripe size of at least 8MB or 16MB is generally recommended in order to span multiple full stripe sets of a DS8000 rank (which internally uses a stripe size with 256kB segments per disk drive) and not to disable the DS8000 cache algorithm's sequential read ahead detection mechanism. For example, using an AIX host system with AIX LVM would mean building an AIX LVM volume group (VG) with LUNs 1000-100f and LUNs 1100-110f (as seen in the charts above), and creating AIX logical volumes from this volume group with an INTER-POLICY of *maximum* and PP sizes of at least 8MB, spread across all LUNs in the volume group (so-called AIX PP striping). If you have another set of volumes from the same ranks, you can simply configure them in a second AIX LVM VG.

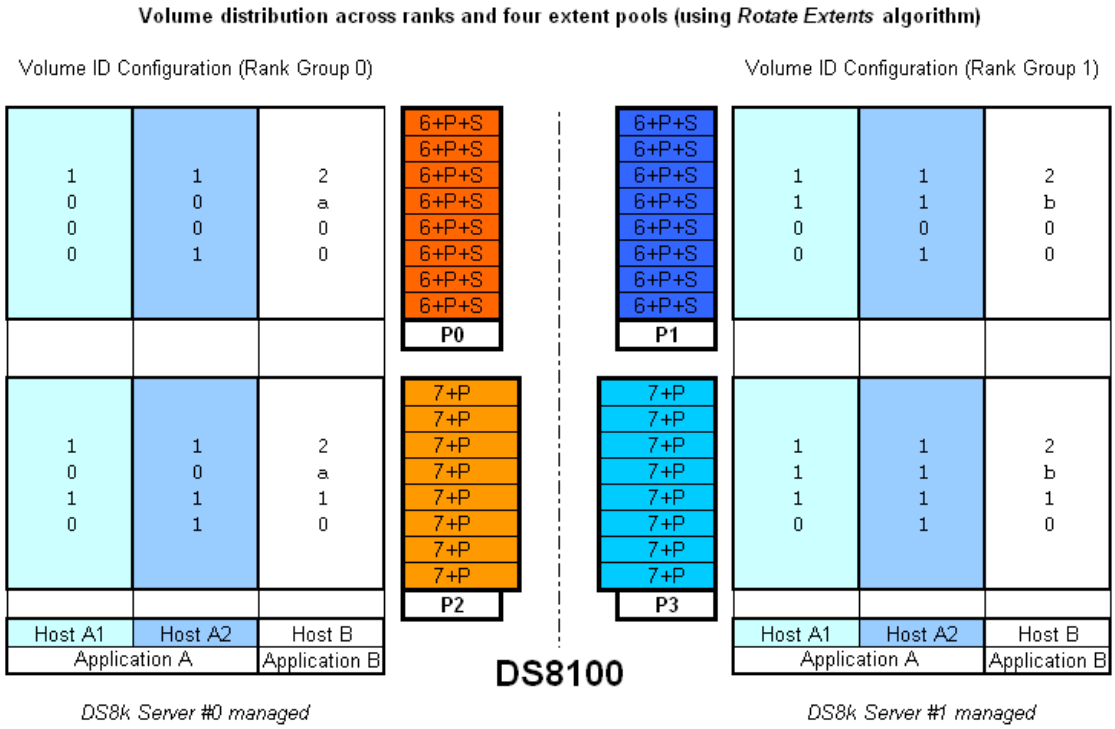


Host level striping, which is available on the AIX operating system, may offer even finer granularity when distributing chunks of each logical volume (PPs of AIX LVs) across multiple LUNs from different ranks than the *Rotate Extents* volume allocation algorithm, which operates only on 1GB (=one extent) extents as a minimum chunk size. Furthermore, using single volumes on dedicated ranks also provides a higher level of manageability with regard to performance management, as you can identify busy volumes on hot ranks and move these volumes to other less utilized ranks. Although the new *Rotate Extents* volume allocation algorithm offers in most cases sufficient granularity for spreading the volumes, you need to be aware that there is currently no way to manage performance at the extent level on the DS8000 in the event of hot extents. This is because there are no performance statistics available at the extent level, and there is no way to relocate single extents of a given volume.

The *Rotate Extents* algorithm, however, combines ease of use and good overall performance without the use of host level striping. The DS8000 simply distributes the volumes and thus balances the workload evenly across the ranks within an extent pool on extent level (1GB). This is a major benefit, especially for Microsoft Windows environments where in many cases only a few large single volumes

are requested per server, and host level striping is rarely used. With volume allocation methods prior to the *Rotate Extents* algorithm, a *single* volume was basically limited to the performance capabilities of a single rank (as an LUN was normally placed on a single rank). Now, with the new *Rotate Extents* volume allocation method, you can fully exploit the performance capabilities of multiple ranks within an extent pools with just a *single* volume – without having to manage multiple volumes on the host system..

Another numbering scheme for volume IDs could be considered when using the *Rotate Extents* volume allocation method, in order to relate the volume IDs to the *extent pools*. In addition to the LSS chosen for a given host system or application, the range of volume numbers could be used to allow a simple mapping between a range of volume IDs and an extent pool, so that you can identify the physical resources (a set of shared ranks within a certain extent pool) behind a volume simply from the volume ID. An example of such a potential volume numbering scheme, with a four extent pool configuration using the *Rotate Extents* volume allocation method, is given in the following diagram:



Finally, when attaching the host systems to the storage subsystem's host adapter ports, you also need to achieve a balanced workload distribution across the *front-end* resources. This can simply be achieved by distributing the FC connections from all attached host systems evenly across the DS8000's HA ports, HA cards, I/O enclosures, I/O enclosure buses and — if available — RIO-G loops.

For high availability, each host system should use a multi-pathing device driver such as SDD and have a minimum of *two* host connections to HA cards in different I/O enclosures on the DS8000, preferably using one left side (even numbered) I/O enclosure and one right side (odd numbered) I/O enclosure so that there is a shortest path via the RIO-G loop to either DS8000 server for a good balance of the I/O requests to both rank groups. For a host system with four FC connections to a DS8100, consider using one HA port in each of the four I/O enclosures. If a host system with four FC connections is attached to a DS8300, consider spreading two connections across two different I/O enclosures in the first RIO-G loop and two connections across different I/O enclosures in the second RIO-G loop. The number of host connections per host system is primarily determined by the required bandwidth. Use an appropriate number of HA cards in order to satisfy high throughput demands.

3 Logical volume layout for databases

The basic principle for the setup of a high-performance layout is to spread data across the DS8000 ranks. The storage fabric guidelines that have been described in the above chapters provide a set of LUNs for a server host or application. These LUNs will be distributed across the DS8000 ranks at the best possible rate, i.e. as optimal as the current utilization of the ranks allows. A customer who runs more than 100 databases and approximately 20 SAP environments on multiple DS8000 subsystems has found that these sets of LUNs provide satisfactory performance for more than 80 percent of the applications.

This chapter gives some operating system- and database-specific recommendations for LUN configuration on a server host.

3.1 Host-specific recommendations

3.1.1 Logical volume manager (LVM)

On the attached UNIX host, each DS8000 logical volume is presented as a physical disk or physical volume (PV). The volumes that were allocated for the data files are packaged to one Volume Group (VG) using the host's Logical Volume Manager (LVM).

With LVM, one or more Volume Groups can be created using these physical disks. LVM organizes Volume Group space in "physical partitions" (PP), which is an AIX term. Other operating systems use different names. As an example, HP-UX uses the term "physical extent" (PE).

We recommend a PP size between 4 MB and 64 MB whenever possible. If few hosts or applications share the DS8000 rank, use large PP sizes. If many hosts or applications share the rank, use smaller PP sizes.

Now that the Volume Group(s) have been defined, the Logical Volumes (LV) can be created on the host. Use host striping to spread the workload accesses across physical resources. Most Logical Volume Managers offer inter-disk allocation policies for Logical Volumes. We recommend physical partition allocation for the Logical Volumes in round-robin order. The first free extent is allocated from the first available physical volume. The next free extent is allocated from the next available physical volume and so on. If the physical volumes have the same size, optimal I/O load distribution among the available physical volumes will be achieved.

This method is called "inter-physical volume allocation policy" by AIX and "distributed allocation policy" by HP-UX. Other terms are "PP spreading" or "poor man's striping".

I. AIX

On AIX, we recommend "physical partition striping", for the following reasons:

1. Single-threaded I/O streams can be spread across multiple physical volumes with high granularity LVM striping. For random I/O loads, LVM striping can potentially impair the ability of storage subsystems to detect and optimize for sequential I/O workloads.
2. If physical volumes are added to a volume group, LVM stripe extension is slightly complex. Since AIX 5.3, stripes can be extended, with the restriction that PVs must be added in multiples of the LV striping width (striped columns).
3. Prior to AIX 5.3, you could enlarge the size of a striped logical volume as long as enough physical partitions were available within the group of disks which define the RAID disk array. If free space was not available in the volume group and physical volumes (=LUNs) had to be added, rebuilding the entire LV was the only way to expand a striped logical

volume.

In this case rebuilding means: backup, delete then recreate the striped logical volume with a larger stripe width followed by a restore operation of the logical volume data (see reference [1], chapter “Striped column support for logical volumes”).

4. With PP striping, any number of physical volumes can be added to the volume group. Then, the *reorgvg* command redistributes physical partitions for a logical volume.

Newer versions of AIX support larger stripe sizes (1 MB with AIX 5.2, up to 128 MB with AIX 5.3), but the advantages of PP striping remain clear, as far as maintenance is concerned.

II. HP-UX

The AIX recommendations are also valid for HP-UX LVM, which supports both (high granularity) LVM striping and poor man’s striping (distributed allocation policy). Note that the HP-UX LVM equivalent for the AIX term “physical partition” (PP) is “physical extent” (PE). HP also supports the use of Veritas Volume Manager on HP-UX. Please refer to the VxVM section in the Sun Solaris chapter below (see reference [2] for LVM and VxVM on HP-UX).

III. Sun Solaris

Symantec Veritas Volume Manager (VxVM) offers several layout methods. With “Striping” (RAID-0) a volume is spread out over two or more physical disks, which in this case correspond to DS8000 logical volumes (see reference [3]).

Solaris Volume Manager (formerly Solstice Disk Suite) also knows “Striping” (RAID-0). “Striping” forms logical storage units by interleaving equally-sized segments of data across two or more physical disks in a round-robin fashion (see reference [4]).

Above all, limiting the number of DS8000 ranks that are used by an application is recommended, unless you are forced not to by huge space or performance requirements. Space distribution to a maximum of 16 or 20 ranks is reasonable in many environments. Adding even more ranks will improve performance, but the relationship between further I/O performance growth and number of used disks will slightly get worse.

3.1.2 Host-specific recommendations — multi-pathing

If a host system is attached to storage devices, a multi-path solution provides enhanced data availability, dynamic (I/O) load balancing across multiple paths and automatic path failover protection. Multi-path solutions are specific to disk storage subsystems, operating systems or volume managers and differ in functionalities and pre-conditions.

I. AIX

On AIX, IBM offers two multi-path solutions for DS8000: IBM Subsystem Device Driver (SDD), and MPIO in combination with a storage subsystem-specific control module (Subsystem Device Driver Path Control Module, SDDPCM). The MPIO base software packets are installed with the operating system. Both SDD and SDDPCM can be downloaded for free (see reference [5]).

IBM has no preference for one of these solutions over the other. However, there are technical restrictions or advantages that can drive the choice between them. In some cases, a customer will have both options.

Examples for distinctive features are:

1. MPIO/SDDPCM is available for AIX 5.2 ML 5 (or later) and AIX 5.3 ML 1 (or later). SDD is also available for older AIX Releases. Please check the DS8000 Interoperability Matrix for current information (see reference [6]).

2. SDD offers a wider choice of path-selection algorithms than SDDPCM (e.g. load balancing sequential). SDD supports SCSI-3 persistent reserve function, while SDDPCM does currently not support HACMP with persistent reservation policies.

II. HP-UX

IBM's SDD provides load balancing for DS8000 subsystem I/O on HP-UX. In addition, HP provides multi-pathing solutions with the HP-UX operating system and the Logical Volume Manager.

Up to HP-UX 11iv2, the Logical Volume Manager "pvlinks" feature is used to manage an active I/O path and so-called "alternate links" to enable fail-over in case of an I/O path failure. "pvlinks" provides path failover in case of an error, but does not provide I/O load balancing. Nevertheless, "pvlinks" makes it possible to change the active disk I/O without interruption, which at least provides a kind of "static" load balancing (see references [2] and [5]).

With HP-UX 11iv3, HP introduced "agile addressing", which creates a single persistent "device special file" (DSF) for each mass storage device, regardless of the number of hardware paths to the disk. The "pvlinks" functionality is still supported with "legacy DSFs". In addition, HP-UX 11iv3 offers a choice of I/O distribution policies (e.g. round-robin, closest path, preferred path). (See reference [7]).

III. Solaris

There are three multi-pathing solutions for use with DS8000 and Solaris: SDD, DMP (Dynamic Multi-Pathing) and MPxIO (Multiplexed I/O).

1. Symantec Veritas Volume Manager (VxVM) includes DMP, which means that it provides multi-pathing as default. Nevertheless, Sun Cluster with VxVM requires MPxIO as an additional layer underneath DMP.
2. With Solaris Volume Manager (formerly Solstice Disk Suite), you can choose between SDD and MPxIO.
3. In either case, MPxIO is a must on Sun Cluster.

IV. OS-independent

Eventually, MPIO could be the better choice since it will handle all external disks through a common framework. MPIO solves problems for systems that need to connect to more than one type of disk subsystem.

On UNIX, Oracle's Automatic Storage Management (ASM) currently does not work with SDD. ASM requires the raw device files to be owned by the Oracle administration user. For Oracle database servers with ASM on UNIX, MPIO/SDDPCM (SDD Path Control Module) is required.

In addition, there are some OS-specifics prerequisites, recommendations or limitations regarding the number of LUNs and FC paths (see reference [8]).

3.2 Database specific recommendations

3.2.1 Oracle

In the past, Oracle recommended that several categories of RDBMS files should be isolated from each other: data files from logs, and logs from archive files. If the number of available disks was insufficient, archive and data files were usually allocated on the same disks.

Today, RAID-5 and RAID-10 architectures both protect against single disk failures within an array. If protection against double disk failures is critical, the old recommendation may continue to exist.

In contrast, two basic concepts exist when looking at performance only:

1. Allocate space for database logs and data files on separate DS8000 ranks.
2. Allocate space for the database logs and data files in the same manner, i.e. create “stripes” on the DS8000 ranks.

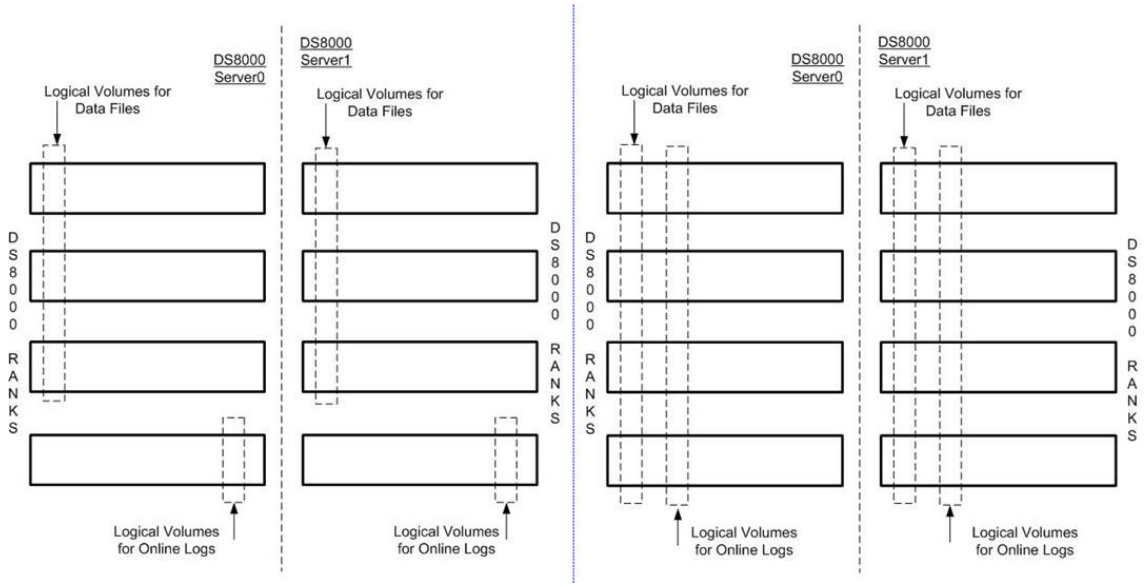


Figure 7: 2 possible DS8000 Logical Volume Layouts for Databases

There are pros and cons for both layout concepts: some may consider the first approach to be a waste of space. With the second method, I/O hotspots that are generated by database log I/O will not be visible.

A few years ago, Oracle began recommending the use of the SAME (Stripe and Mirror Everything) methodology for the layout of Oracle database files. SAME has two fundamental goals:

1. Provide protection against single disk failures.
2. Reduce I/O hotspots by balancing I/O activity across multiple physical disks.

Our approach to distributing data files across the available ranks together with the DS8000’s RAID technology is consistent with SAME.

3.2.2 Oracle ASM

Oracle Automatic Storage Management (ASM) is a database file system that provides cluster file

system and volume manager capabilities that are fully integrated into the Oracle database kernel. Oracle ASM was introduced with the Oracle 10g release and offers an alternative to OS-specific file systems.

It can be used to provision storage for database files like data files and redo logs. However, it is not used to store binaries, trace, and text files, and thus it is not the only tool needed to provision storage for Oracle database environments.

ASM introduces the term “disk group”, which is a pool of disks managed as a logical unit. A disk group divides total disk space into uniform megabyte-sized units. ASM distributes each Oracle file evenly across all disks in a disk group. The default “ASM stripe size” is 1 MB.

ASM offers the possibility of creating separate disk groups for Oracle database file types like data files, logs and recovery files.

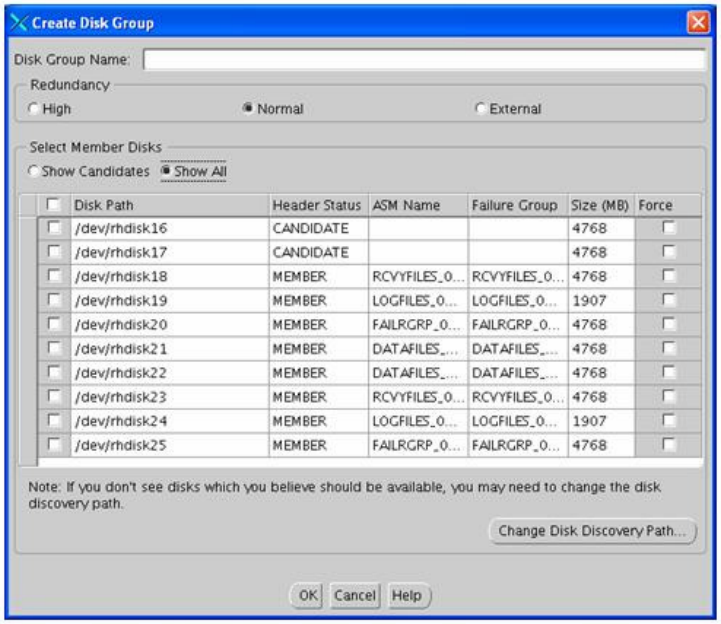


Figure 8: ASM disk group definition

According to this, the common recommendation to “spread and stripe everything” can be implemented with ASM.

It is worth mentioning that there are some restrictions or special factors relating to the use of ASM. For example, IBM’s SSD multi-pathing driver is currently not supported with ASM on AIX. You should choose MPIO instead (also see the Multi-pathing chapter above).

3.2.3 IBM DB2

Tablespaces are a logical level between the database and the tables stored in the database. DB2 offers two types of tablespaces: SMS (system managed space) and DMS (database managed space). A database can have any combination of SMS and DMS tablespaces.

In an SMS tablespace, the operating system’s file system manager allocates and manages the space

where a table is stored. The database administrator decides on the location of files, DB2 controls their names, and the file system manages them.

In a DMS tablespace, the database manager controls the storage space. The database administrator decides which files or devices to use and DB2 manages the space on these devices and files. The files and devices that DMS uses are also called “containers”. According to a rule of thumb, database I/O performance with DMS is about 10 percent faster compared to SMS.

In a container, data for an object is stored using extents. DB2 stripes the data across all available containers in the tablespace, based on the extent size. The default extent size is 32 pages. Valid page size values are 4 KB, 8 KB, 16 KB and 32 KB. The round-robin process of writing to containers offers a possibility to balance I/O workload across the containers of a tablespace.

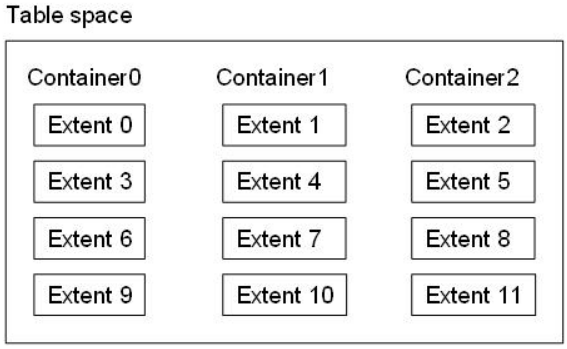


Figure 9: DB2 table space with three containers and 12 extents

You do not always have the chance to decide which type of table space to use. As an example, only DMS tablespaces are supported for SAP systems with DB2 as a database. SAP recommends that you only create SMS tablespaces as temporary tablespaces (see SAP note 543897).

3.2.4 General database-specific recommendations

We recommend always using 15 krpm disks drives for demanding, IOPS-critical database production environments. 15 krpm disk drives deliver 25% better performance than 10 krpm drives. These are some very rough rules of thumb which provide a conservative estimate:

FC 15 krpm DDM : 160 IOps

FC 10 krpm DDM : 120 IOps

SATA 7.2 krpm DDM: 50 IOps

Thus, ten drives of 15 krpm disks can deliver approximately 1,500 IOPS.

In the planning phase, IBM’s Disk Magic tool should be used to analyze drive capacity choices. Disk Magic is a Windows-based disk subsystem performance modeling tool. It is a product of IntelliMagic, licensed exclusively to IBM and IBM Business Partners. Updated versions of Disk Magic are available at regular intervals (typically one per month). Each version of Disk Magic will expire at a pre-assigned date (see reference [9]).

3.3 Recommendations for FlashCopies

The integration of the DS8000 FlashCopy function with database backups provides significant advantages for production environments:

1. The database server’s CPU is relieved by relocating the backup to a so-called “backup mover”.
2. The backup window is shortened (i.e. the time that the database stays in backup mode is

optimized).

3. Fewer redo logs will be required to roll the database forward to a state in which it can be restarted.

3.3.1 Performance

If a short backup window is guaranteed, the importance of backup performance and duration is reduced in many cases. However, some performance aspects remain:

1. If FlashCopies are initiated with the background copy option, the time that is required to synchronize the source and target volumes is important because it defines the interval for subsequent FlashCopies. The background copy option is not required for backups, but is – for instance – used by Tivoli Storage Manager for Advanced Copy Services (TSM4ACS), which leverages FlashCopy technology for the backup of databases or SAP environments (see reference [10]). TSM4ACS currently supports the DS8000 series, ESS and IBM TotalStorage SAN Volume Controller (SVC).
2. The read throughput for the backup data can influence the backup duration, and thus the utilization of hardware resources (e.g. tape libraries and drives).

If performance of the FlashCopy backup environment is an issue, consider these recommendations:

- Separate the source and target volumes of each pair to different DS8000 ranks within the same server.
- Allocate the source and target volumes of each pair to ranks that are served by different device adapter (DA) pairs.
- Run source and target volumes with the same RAID geometry.

3.3.2 FlashCopy pre- and post-processing for Oracle/DB2

If a FlashCopy of a database is created, particular attention must be paid to the consistency of the copy – i.e. it must be possible to start the database copy that is allocated to the FlashCopy target volumes. Apparently, the easiest – and most unusual – way to provide consistency is to stop the database before creating the FlashCopy pairs. If a database cannot be stopped for the FlashCopy, some pre- and post-processing actions have to be performed to create a consistent copy.

Two DS8000 feature are used to provide a consistent database copy: Logical Subsystems (LSS) and Consistency Groups (CG).

Consistency Groups are quite useful in database environments because consistency across multiple LUNs or logical volumes can be achieved in a backup copy. Some characteristics of consistency groups:

- freeze (i.e. temporarily queue) I/O activity to a LUN or volume
- create consistent point-in-time copies across multiple LUNs or volumes
- offer an alternative to quiescing host or database I/O

LSS is a logical construct to group logical volumes. This means that DS8000 logical volumes can belong to the same LSS but still reside in multiple arrays or ranks. The logical volume's LSS is defined when it is created.

Freezing of I/O to preserve data consistency across multiple copy pairs is done at the LSS level. If several applications share volumes in one LSS, I/O freeze will apply to these applications, because consistency group 'create' commands are directed to each LSS that is involved in a consistency group.

The prerequisites and the algorithm for a DB2 or Oracle database FlashCopy are described below.

Database FlashCopy prerequisites:

1. If the database files are allocated in file systems, allocate the file system log (*jfslog*) to one disk only (i.e. do not use striping for the file system log) or use FlashCopy consistency groups.
2. The FlashCopy target volume allocation should try to minimize the number of DS8000 Logical Subsystems (LSS) involved. Thus, FlashCopy handling will be slightly simplified: FlashCopy consistency groups are handled at the LSS level.

Database FlashCopy algorithm:

Use Oracle online backup mode or DB2 suspend I/O.

Remark: An I/O suspend is not required for Oracle if Oracle hot backup mode is enabled. Oracle handles the resulting inconsistencies during database recovery.

Perform file system sync before FlashCopy creation.

Optionally perform a file system freeze operation before and a thaw operation after the FlashCopy.

Remarks:

(1) If the file system freeze is omitted, file system checks will be required before mounting the file systems on the FlashCopy target volumes.

(2) Commands used on UNIX operating systems to freeze/thaw I/O include:

OS	freeze	thaw
Solaris	lockfs –w <fs>	lockfs –u <fs>
AIX	chfs –a freeze=<timeout> <fs>	chfs –a freeze=off <fs>
HP-UX		A command interface is not available. The VX_FREEZE and VX_THAW ioctl system calls are used.

Use FlashCopy consistency groups if the file system log (*jfslog*) is allocated on multiple disks

First create FlashCopies of the data files, then switch database log file, and finally create FlashCopies of the database logs.

Remark: If an online backup is performed, the backup of Online Redo Logs will be inconsistent. Therefore, you should choose another process to provide the log files for a database recovery (e.g. *brarchive* in SAP environments).

3.3.3 Oracle ASM

In an ASM environment, a so-called ASM instance and a disk group are created for managing and storing Oracle database files. Subsequently, you can create multiple disk groups for the ASM instance to manage. For example, separate disk groups for data files, redo logs and backup space are possible.

Consequently, a FlashCopy algorithm for a database in an ASM environment must handle the ASM instance. This includes copying the database control and initialization (*.ora) files as well as configuring Oracle Cluster Synchronization Services (CSS) which handle synchronization between

ASM and the database instances.

An IBM white paper describes in detail how to use FlashCopy to make point-in-time copy of the database volumes in an ASM environment (see reference [11]).

References

[1] Redbook: AIX 5L Differences Guide Version 5.3 Edition
http://www.redbooks.ibm.com/abstracts/sg247463.html?Open

[2] HP-UX 11i v3 LVM and VxVM Manuals: (1) HP-UX 11i Version 3 HP-UX System Administrator's Guide: Logical Volume Management (2007, includes information about LV striping and multi-pathing) (2) VERITAS Volume Manager 4.1 Administrator's Guide HP-UX (July 2006) http://docs.hp.com/en/oshpux11iv3.html

[3] Veritas™ Volume Manager 5.0 Administrator's Guide for Solaris

http://www.sun.com/products-n-solutions/hardware/docs/pdf/875-3890-10.pdf

[4] Solaris Volume Manager Administration Guide, June 2006

http://docs.sun.com/app/docs/doc/819-2789?q=%22Solaris+Volume+Manager%22+%22Solaris+10%22&a=load

[5] IBM DS8000 Multi-path Device Driver Download Page:

http://www-1.ibm.com/support/dlsearch.wss?rs=540&tc=ST52G7&dc=D430

[6] IBM DS8000 Interoperability Matrix:

http://www-03.ibm.com/servers/storage/disk/ds8000/pdf/ds8000-matrix.pdf

[7] HP white papers about Storage Management with HP-UX 11i v3:

http://docs.hp.com/en/netsys.html#Storage%20Area%20Management

[8] IBM System Storage: Multi-path Subsystem Device Driver User's Guide

http://www-1.ibm.com/support/docview.wss?rs=540&context=ST52G7&uid=ssg1S7000303

[9] Disk Magic download for IBMers:

http://w3-1.ibm.com/sales/systems select: Selling Hardware → Tools → Tool

Disk Magic Download for Business Partners via "PartnerWorld":

https://www-1.ibm.com/partnerworld/sales/systems/portal/.scr/Login

[10] "Tivoli Storage Manager for Advanced Copy Services" Product Information

http://www-306.ibm.com/software/tivoli/sw-atoz/indexS.html

[11] IBM Whitepaper by Pat Blaney, Erik Salander: Oracle® 10g R2 with Automatic Storage Management and IBM® System Storage FlashCopy Backup and Recovery Procedures (Draft version from Dec 13, 2006) http://w3-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100929

