



SAP Advanced
Technology Group

**Database Layout for SAP Installations
with DB2 UDB for Unix and NT**

February 23, 2001

Contents

Contents	2
1 Disclaimer.....	3
2 Abstract.....	3
3 Version History	3
4 Introduction.....	4
5 Understanding UDB Architectural Constraints.....	4
5.1 Table Space Size Limitation	4
5.2 Balanced Containers.....	5
6 Design Aspects of the Layout.....	6
6.1 Concept	6
6.2 Example of Layout	7
6.3 Page Size.....	8
6.4 Isolating Tables from Standard Table Spaces.....	9
6.5 Container Size and Number of Containers	10
6.6 Logging	10
6.7 Filesystems: Temporary Table Spaces etc.....	11
6.8 Performance Considerations.....	12
7 Growing the Database/Log Space	12
8 Miscellaneous	13
8.1 EEE Specific.....	13
8.2 AIX Specific.....	13
8.3 IBM ESS Specific.....	14
9 Summary.....	14
10 References.....	14

1 Disclaimer

This paper is currently work in progress. No guarantee is given for any information presented here. Recommendations are generally derived from SAP's and customers' experience. They may or may not apply to specific customer installations.

2 Abstract

This paper summarizes SAP-specific aspects of the physical design of a DB2 UDB database. Primarily, the placement and layout of DB2 UDB objects (table spaces and directories) on storage devices is discussed. Recommendations for strategic decisions on configuring the table spaces are given and explained. The fundamental concept is to distribute all kinds of data across all devices available. This has been proposed recently for state of the art storage systems and for other database management systems. The advantage of this approach is that the full bandwidth of the whole I/O subsystem is available to each individual unit (table space, directory) of the database.

The examples in the current version of this paper are based on an IBM ESS (Shark) storage system. Comments regarding different hardware configurations are welcome. Feedback on anything else that is of interest for this paper is requested, too.

3 Version History

Date	Author	Comment
January 31st, 2001	Jens.Claussen@sap.com	first draft for internal review
February 15th, 2001	Jens.Claussen@sap.com	new layout, completed

4 Introduction

Customer experience has shown that if SAP installations suffer from bad database I/O performance, this problem is often caused by an inappropriate placement of database objects on disks. In some cases, although more than one hundred (logical) disks are available, most of the I/O activity is concentrated on a few disks. Carefully planning the database storage layout for today's and tomorrow's need helps to avoid these problems. This paper summarizes design aspects for SAP installations based on a DB2 UDB database.

Primarily, the placement and layout of DB2 UDB objects (table spaces and directories) on storage devices is discussed. Recommendations for strategic decisions on configuring the table spaces are given and explained. The fundamental concept is to distribute all kinds of data across all devices available. This has been proposed recently for state of the art storage systems and for other database management systems (see references [2], [3]). The advantage of this approach is that the full bandwidth of the whole I/O subsystem is available to each individual unit (table space, directory) of the database.

Former database layouts often tried to separate table spaces onto different devices, e.g., separating data and index table spaces and putting each data table space on different drives. The consequence was that the database often ended in hotspots on devices that carried a single table space. Manual intervention was necessary in each case to alleviate the problem. As time went by, changing application profiles of data growth led to other hotspots or restructuring demands. The goal of the proposed layout is to avoid these problems. Hotspots will rarely occur since each type of data is striped across multiple devices. Consequently, no manual intervention will be necessary to correct the layout. This concept must also be implemented when growing single table spaces to cope with an increasing amount of data. Another goal of the database layout is to minimize planned downtime for database reconfiguration tasks like restructuring table spaces.

The examples in the current version of this paper are based on an IBM Enterprise Storage Server (Shark) storage system [9]. Comments regarding different hardware configurations are welcome.

This paper does not try to compete with other DB2 UDB specific documentation like DB2 manuals (e.g., [6]), IBM redbooks [7] and other material about database administration and tuning. Instead, only supplemental information that is specific to SAP systems is given.

In the remainder of this paper, the terms DB2 UDB, UDB, and DB2 always refer to the Unix and NT release of DB2 UDB Enterprise Edition (EE) Version 7. Other members of the DB2 family like DB2 UDB for OS/390 (zSeries) and DB2 UDB for AS/400 (iSeries) are not covered here. Furthermore, most SAP applications built upon the 4.6 basis release are covered.

5 Understanding UDB Architectural Constraints

DB2 UDB suffers from two limitations that have important impact on the freedom to design the database layout: The size limit for table spaces and the data balancing for containers within a table space. The proposed layout takes these two limitations into account.

5.1 Table Space Size Limitation

The architecture of DB2 limits the size of each database managed (DMS) table space to 2^{24} data pages. For the standard page size of 4kB, this means that a table space cannot be larger than 64GB. This upper limit must be considered when sizing the database and designing the layout of the database.

There are several ways to alleviate the impact of this limitation:

1. Use system managed (SMS) table spaces
SMS table spaces do not restrict the size of a table space. Instead, the size of each *table* cannot exceed the number of 2^{24} pages. SAP, however, does not recommend to use SMS for regular table spaces for performance reasons. SMS is only recommended for temporary table spaces.
2. Isolate large tables from the standard SAP table spaces
By default, all tables delivered by SAP are collected into a small number of table spaces (e.g., 13 data and 13 index table spaces for R/3). If one table space reaches its maximum size, some of the largest tables within this table space can be moved to different, newly created table spaces.

Depending on the estimated table sizes, either multiple tables are moved to a single new table space, or each table is moved to a different new table space. Section 6.4 discusses the isolation of tables in more detail.

3. Use the Enterprise – Extended Edition (EEE) of DB2
 The Enterprise – Extended Edition of DB2 allows to spread a database across multiple hosts and to install multiple *partitions* on a single host. The limitation of 2^{24} pages per table space refers to a single partition of the database. Consequently, if a table space is distributed across multiple partitions (nodes) of the EEE instance either on a single host or on multiple hosts, the limit is raised to $n * 2^{24}$ pages per table space (n is the number of partitions on which the table space is stored). SAP, however, currently supports EEE only for a few mySAP components.
4. Use larger database page sizes
 The 2^{24} page limitation yields a total amount of useable space of 64GB for the standard page size of 4kB. DB2 permits to choose the page size individually for each table space. Possible page sizes are 4kB (default), 8kB, 16kB, and 32kB. By choosing larger pages for table spaces, the total size limit for a table space can be raised up to 512GB for 32kB pages without exceeding the total number of 2^{24} pages. There are, however, several restrictions in choosing the page size. Details are covered in Section 6.3.

5.2 Balanced Containers

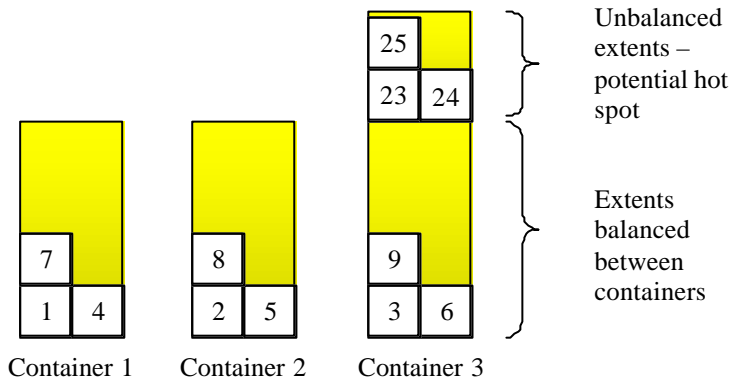


Figure 1: Table Space with Unbalanced Containers

If a DB2 table space consists of multiple containers of the same size, the DBMS tries to evenly distribute the data and thereby I/O load across all containers. This is achieved by allocating extents for database objects alternating by a round robin strategy from the different containers. If the containers do not have the same size, only parts of the data are balanced. In the example shown in Figure 1, container 3 carries more data extents and it is therefore a potential hotspot. To re-establish a balanced container layout, the table space needs to be recreated, e.g., by a redirected restore.

The recommended container layout is to start with several containers of the same size and to keep them balanced by either adding only containers of the same size or by extending all containers simultaneously by the same amount.

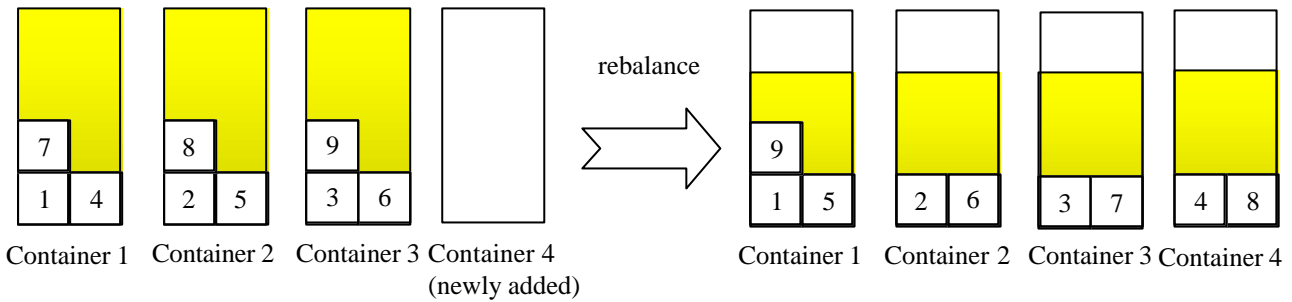


Figure 2: Rebalance after Adding a new Container

Figure 2 shows the scenario if a new container with the same size as the existing containers is added to a balanced table space (`alter table space add ...`): The complete table space is rebalanced such that afterwards all containers show the same utilization. Of course, the rebalancing process poses some additional load onto the database host and the newly added space is only available after the rebalancing process has finished. If multiple containers have to be added to a table space, it makes sense to add all containers within the same command to get along with a single rebalancing process. Adding a container of a *different* size does not trigger rebalancing and consequently immediately makes the new space available. This means, however, giving up the well-balanced container layout and accepting hotspots on the new container.

Starting with Version 7, DB2 offers an alternative way of adding space to an existing table space and keeping it balanced: The command `alter table space extend` (or `resize`) allows to enlarge the balanced containers of a table space (see Figure 3). This way all containers scale by the same amount of storage space. This extension requires, however, room for growing the already existing underlying filesystems or raw devices of the containers.

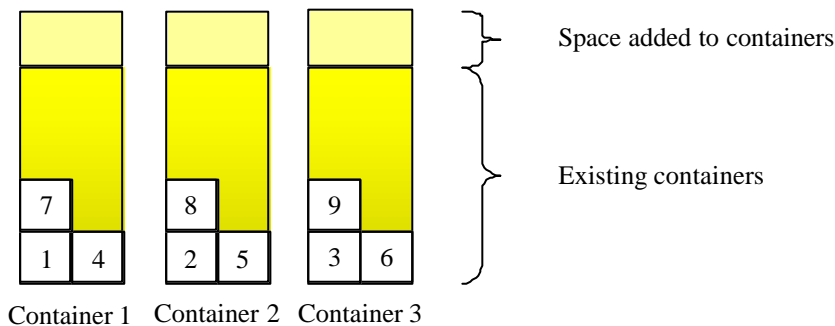


Figure 3: Extending Existing Containers

6 Design Aspects of the Layout

The new layout takes the changing hardware and software equipment at customer sites into account. Currently, most new customer installations start with a storage area network (SAN) environment based on advanced storage systems as disk subsystems. Since many UDB customers use RS/6000 AIX machines as database hosts and the IBM Enterprise Storage Server (ESS) as disk subsystem, the examples in the following are based on this scenario. They should be easily transferable to other hardware configurations. Furthermore, Section 8 covers some technical details that are special to AIX and ESS. Details for other scenarios may follow.

6.1 Concept

The central idea is to stripe all data objects across all available devices. This is the opposite of placing each table space on a set of devices of its own. There are several advantages of choosing a large number of thin

fragments of a table space instead of a few thick fragments: First, there always tend to be hotspots on a subset of the table spaces, sometimes even on a single table space. By striping these table spaces across all devices, the cumulative performance of all devices and device adapters is available to each individual table space. Second, the hotspot table space is not always known in advance or may change with changing workload on the system. If all table spaces or at least any frequently used table spaces are striped at maximum width, no care must be taken to identify the heavy-loaded table spaces. There is hardly any additional cost of striping table spaces across all devices, so all table spaces can be striped. Third, the amount of space allocated on each single volume contained within the stripe for a single table space is minimized if the number of volumes is maximized. As a matter of fact, the seek time for placing the disk head(s) on the desired track increases (although not linearly) if the zone where the covered data grows. Consequently, the seek time is minimized if the data zone on each single disk is minimized (assuming coalescing data blocks for a single table space on the disks). On the other hand, of course, the seek time for switching between different kinds of data placed on the same disk is added. In summary, the proposed layout concept should yield at least as good seek times as other approaches that concentrate each kind of data on a few disks.

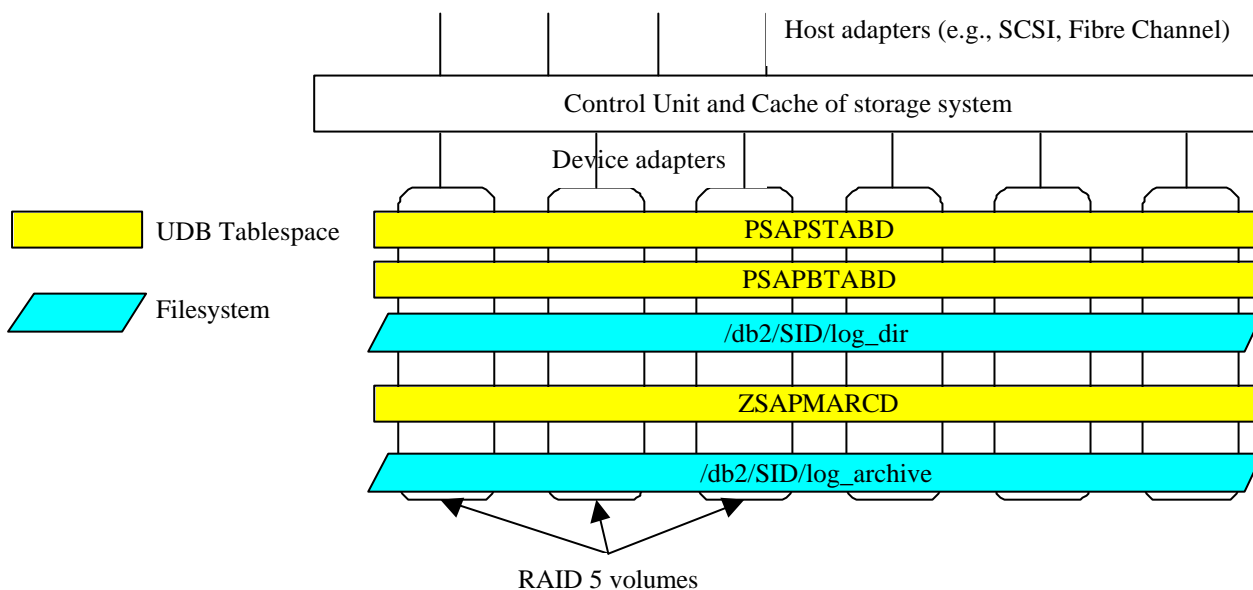


Figure 4: Concept of the Layout: Stripe Everything Across all Devices

6.2 Example of Layout

Figure 4 shows parts of an example layout on an IBM ESS storage system [9]. The vertical boxes denote RAID 5 volumes (“ranks”) consisting of several physical disks. The ranks are connected via device adapters to the control unit that contains, e.g., cache and NVRAM. The control unit attaches to host systems via SCSI or fibre channel adapters.

Each UDB table space and each filesystem is striped across all RAID 5 volumes. For each table space, the cumulative bandwidth of all device adapters (adapters between RAID volumes and the storage system central processing unit) is available to the table space. If the host adapters are configured accordingly, also the cumulative bandwidth of all hostadapters is available to the individual table space.

Certainly, the disk and adapter bandwidth needs to be shared among all table spaces. If there is substantial load on multiple table spaces, only a fraction of the bandwidth is available to each single table space. However, as long as there is no severe preference with respect to performance for any kind of data, this is still the optimum configuration since bandwidth is “allocated” automatically and dynamically on demand to the table spaces.

There is special interest in placing the online log volumes on the disks with maximum performance since minimum disk response time for online log operations is performance-critical for all database modification operations. For this reason, placing the stripe for the online log in the middle of the disk is proposed. This on

average yields minimum seek time from all head positions. Particular alternatives for log volumes are discussed in Section 6.6.

6.3 Page Size

As already mentioned in Section 5.1, there is a size limit of 64GB for table spaces with the default page size of 4kB. This limit can be raised up to 128GB, 256GB, or 512GB by means of increasing the page size to 8kB, 16kB, or 32kB, correspondingly. Furthermore, performance experiments have shown that larger pages also increase the performance for certain kinds of queries (Largest performance gains have been observed for queries involving heavy sequential I/O accesses, as they are observed, e.g., for OLAP workloads.). The side effects of using page sizes other than 4kB are discussed in the following.

First of all, there is no way to switch the whole database to larger pages. At least the catalog table space (SYSCATSPACE) has to remain at 4kB. Next, each page size used requires a distinct buffer pool with corresponding page size. Consequently, for each newly introduced page size a new buffer pool must be allocated. If temporary table spaces are used for table reorganization, then also for each page size a corresponding temporary table space is required.

DB2 limits the number of data rows on a single page to 255. For large page sizes and small rows the capacity is limited by the maximum number of rows per page and not by the number of bytes usable per page. This may lead to underfull pages and, as a consequence, to wasted storage and buffer pool space. Figure 5 shows the fillrates of different page sizes for the same row size. Assume a data page is filled with data rows of 50 byte length including all overhead. Then, a 4kB page can carry 80 data rows. An 8kB page has room for 162 rows. A four times larger page of 32kB has room for about 650 rows. Due to the upper limit for the number of rows, however, the page can only contain 255 rows. This means that because of the small rows about 60% of a 32kB page are wasted.

On the other hand, DB2 is not able to split large data rows between multiple pages. Very large data rows may therefore require larger page sizes. Table 1 lists for each page size the maximum table space size, the maximum data row length, and the minimum average data row length (excluding all overhead) that still yields a 100% filled page.

Page Size	Max. Table Space Size	Max. Row Length	Min. avg. Row Length
4 kB	64 GB	4,005 B	5 B
8 kB	128 GB	8,101 B	21 B
16 kB	256 GB	16,293 B	53 B
32 kB	512 GB	32,677 B	118 B

Table 1: Some Parameters Influenced by the Page Size

There is no unambiguous insight that larger pages generally yield better database performance. Experiments have shown that especially I/O-intensive queries perform better with larger database pages. On the other hand, more buffer pool memory is required to keep the same number of pages for random I/O workload on larger pages. That is, larger pages waste more buffer pool space with unwanted rows. Furthermore, the optimizer support for page sizes other than 4kB is suboptimal. For this reason, the probability that the optimizer chooses a suboptimal access plan (e.g., using not the most adequate join method) is a little higher.

For index table spaces, there is an additional advantage of larger pages: Since larger pages can carry more index key entries, the reduced total number of index pages may allow to save an index level, such that one I/O access is saved for an index access.

For ease of administration, it is recommended to choose only one additional page size apart from 4kB. This minimizes fragmentation of the buffer pool and simplifies the optimizer's work. Different sizes for data and index pages are possible. Only a single temporary table space is required for the database. When reorganizing tables and using temporary table spaces for the reorganization, however, a temporary table space with the same page size as the table space that carries the table to be reorganized must be available. Figure 6 shows an example configuration with 4kB and 16kB pages. Here, the table spaces PSAPSTABD, PSAPSTABI, PSAPBTABD, PSAPBTABI are moved to 16kB pages. The required 16kB buffer pool is named

SAP16BP. If reorganization of at least one of the four 16kB table spaces is supposed to use a temporary table space, the 16kB temporary table space (named ZSAPTEMP16 in the figure) is required. Since temporary table spaces should use SMS, multiple temporary could share the same filesystem in order to minimize space requirements.

More information about the use of different page sizes is given, for example, in the DB2 Administration Guide ([6], Chapter 8 "Physical Database Design").

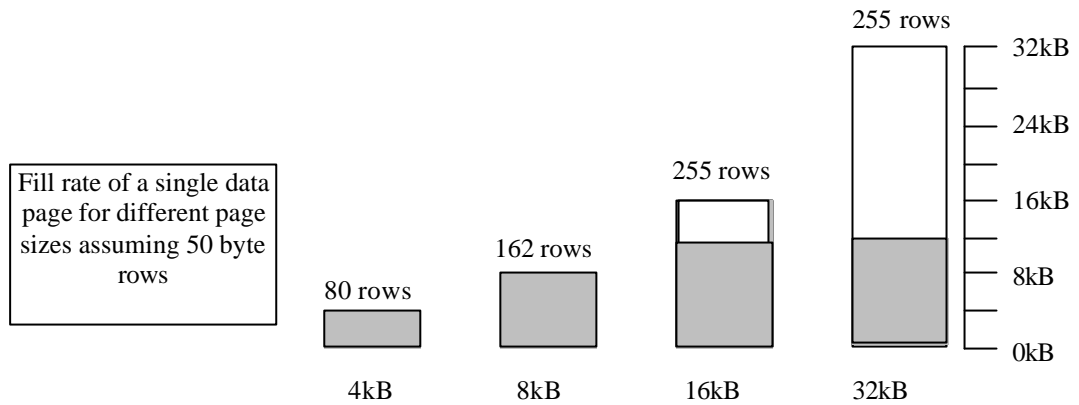


Figure 5: Effect of Different Page Sizes on the Fill Rate of a Page

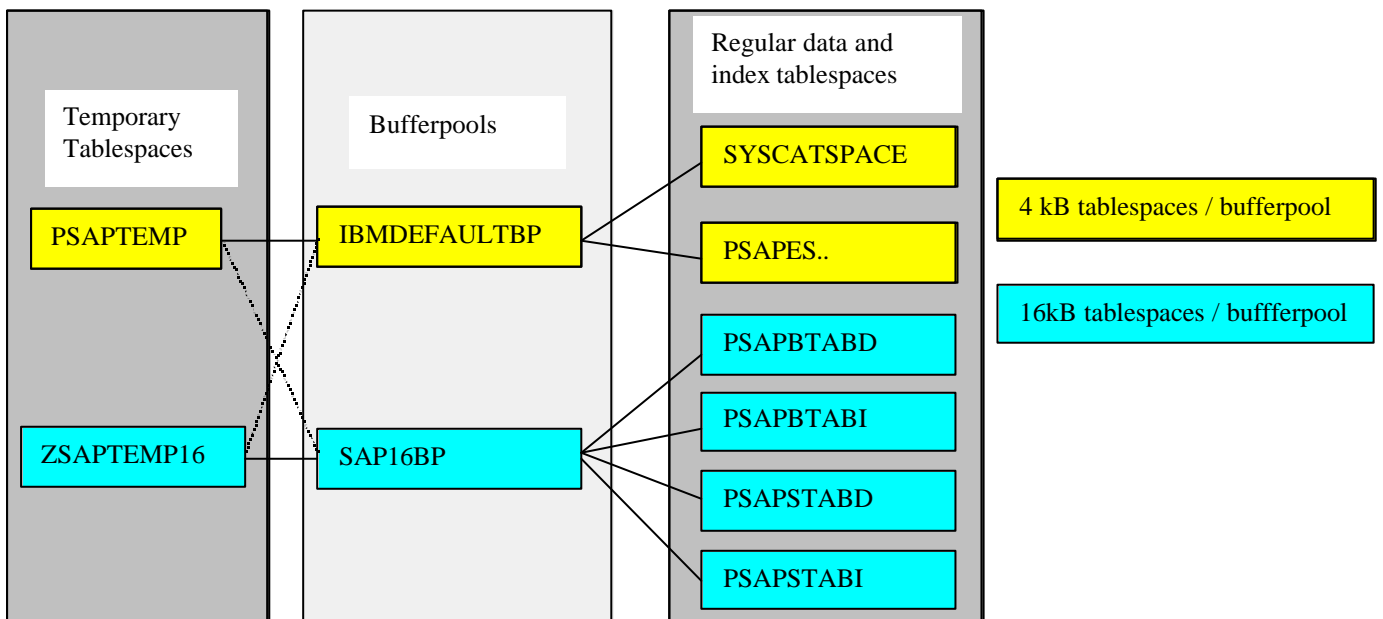


Figure 6: Table Spaces and Buffer pools with two Different Page Sizes

6.4 Isolating Tables from Standard Table Spaces

A complementary approach to overcome the 64GB limitation is the isolation of large tables. It can be applied instead of using different page sizes or in combination with larger pages. If a table space reaches its size limit, the major space consumption is commonly due to a few large tables within the table space. Moving some of these tables to a newly created table space solves the problem. Depending on the current and expected sizes of the tables, it may be beneficial to migrate either each table to its own new table space or to move a group of tables from one table space to a new one. SAP Note #136702 [4] explains the procedure of moving tables between table spaces. Note that moving tables in general requires downtime.

6.5 Container Size and Number of Containers

For each DMS table space one or more table space containers (operating system files or raw devices) must be allocated. For containers larger than 2GB, the operating system must be large file enabled. If the operating system does not pose a limit on the file size, there is a choice of setting the number of containers for each table space, selecting between raw devices and file containers, and deciding where to put the containers on the storage layout. These decisions will be discussed in the following.

Typically raw device containers yield higher I/O throughput than file containers, since the additional layer of the filesystem buffer cache is circumvented by raw devices. So from a performance point of view, raw devices should generally be preferred to files. On the other hand, file containers are sometimes considered to be easier to handle from an administration point of view. Furthermore, there are situations where the filesystem buffer cache can serve I/O requests that otherwise would go to disk. In particular, table spaces for *long* data should use file containers since long data is not cached in the buffer pool. [7] discusses the aspect in more detail.

Performance experiments with raw devices under AIX have shown that the I/O performance of DB2 is substantially higher if multiple containers are used for a table space rather than a single one, even if DB2_PARALLEL_IO is enabled for the table space. Derived from this, multiple containers for each table space with significant traffic are recommended. It may even be useful to allocate multiple containers on the same RAID5 volume. In order to keep the containers balanced, all containers should have the same size.

Since DB2 applies a striping approach on extent basis the individual containers should not be striped once again across multiple volumes. Instead, each container should be kept locally on a single volume (disk). A clear one-to-one mapping between containers and disk/RAID volumes also simplifies administration.

Figure 7 shows example layouts of two table spaces. The first table space consists of a single container that is striped across all available RAID volumes (vertical boxes). While striping the table space across all volumes is recommended, using a single container yields suboptimal performance. The preferred solution is shown for the second table space: It is also striped across all RAID volumes, but it is divided into twelve containers. Two containers are mapped to each RAID volume. (Recommendations about the number of containers to map to a RAID volume vary. But more than one may be useful for larger containers.) When using file containers, put them into multiple filesystems – at least one per RAID device.

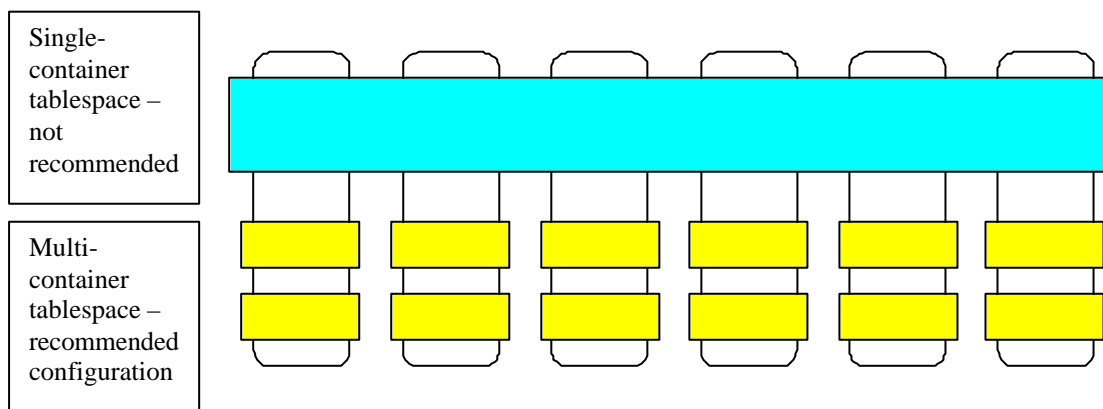


Figure 7: Layout of a Single DMS Table Space

6.6 Logging

The general recommendation is to use the fastest disks (to be more concrete: disks with minimum response time) for online log files.

One approach for the placement of the online log files has already been presented in Figure 4 (repeated in Figure 8(a)): The filesystem for the online log is striped across all devices, just as any other file system and table space. While this is a good configuration for performance, customers may require different layouts for

security reasons. Separating data volumes, online log, and archive log is recommended, for example, in the installation guide. Additional mirroring of the online log may also be desired.

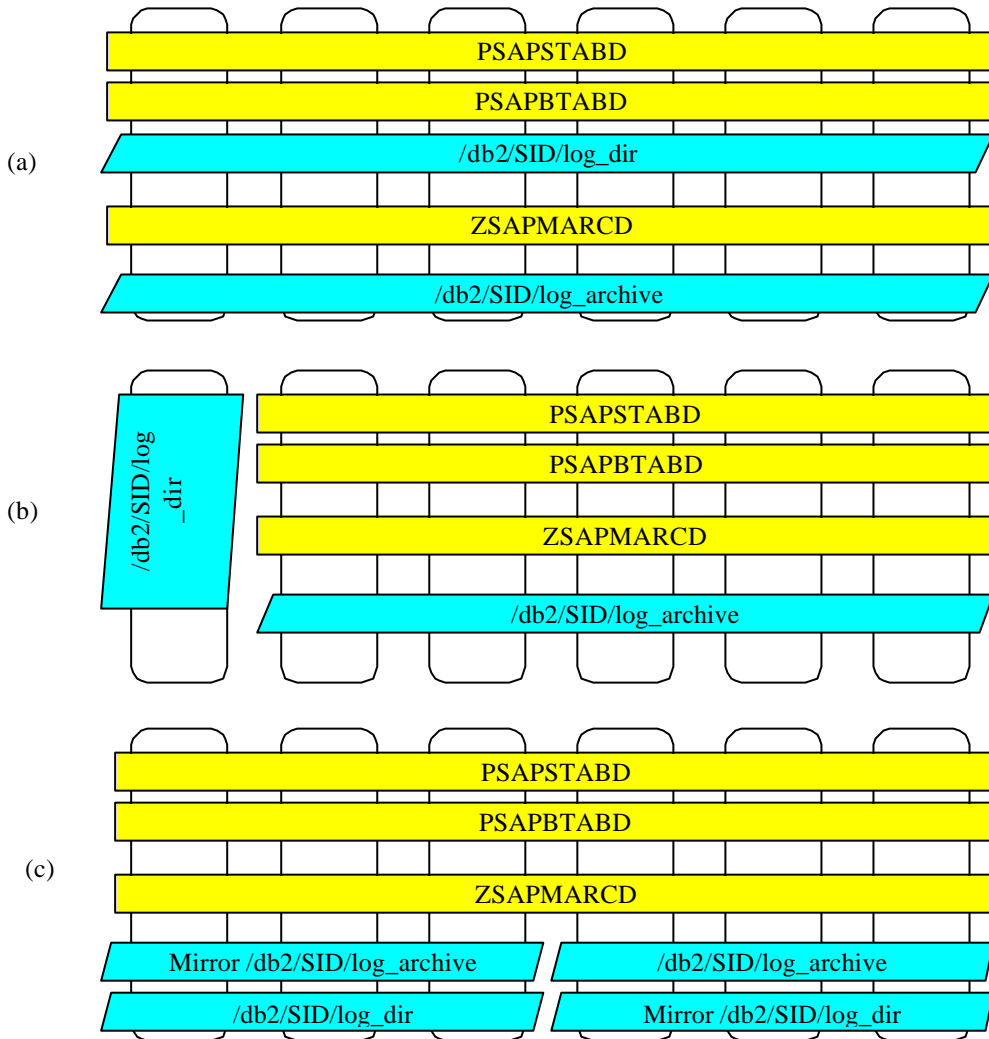


Figure 8: Three Different Approaches to Place Log Files on the System

In addition to the already known variant (a), Figure 8 shows two more feasible approaches to place online and archive log. In configuration (b), the online log is placed on a physically separate volume. Assuming that each RAID volume is driven by a distinct RAID adapter, strong isolation of the online log is achieved. With respect to performance, this layout may however be suboptimal.

Configuration (c) in Figure 8 shows an alternative using mirroring. Both online and archive log are replicated by a mirror established on logical volume manager basis. However, the logs still share the same physical volumes as the data table spaces.

6.7 Filesystems: Temporary Table Spaces etc.

The same basic recommendation that is given for DB2 table spaces also holds for filesystems. That is, stripe all filesystems that carry substantial I/O traffic across multiple devices. The two file systems for logging (online log and archive log) were already covered in the previous subsection. The most prominent filesystems not yet covered are filesystems for SMS table spaces, in particular SMS temporary table spaces. Filesystems for temporary table spaces should also be striped across multiple devices. If using multiple

containers (directories) for a SMS temporary table space, it may be beneficial to distribute these across multiple filesystems in order to avoid lock contention on a single filesystem.

6.8 Performance Considerations

This paper is not intended to cover parameter settings or other detailed DB2 tuning. Several tuning details are given, e.g., in [6] and [7]. In the following, only a few areas of configuration for performance optimization are enumerated:

The DB2 registry variables `DB2_STRIPED_CONTAINERS=ON` should be obligatorily activated before creating any table space that resides on striped devices. It enables the alignment of device stripes with DB2 extents. The extent size chosen at the time of table space creation should be chosen such that the extent size is equal to or a multiple of the RAID strip size (Strip size is the amount of data placed consecutively on a single device within the RAID stripeset.). Furthermore, the variable `DB2_PARALLEL_IO` should at least be active for table spaces that consist of containers striped across multiple devices (as this may occur, e.g., for containers residing on a RAID5 rank of an IBM ESS).

Table space parameters in general (extent size, prefetch size, transferrate, accesstime) should be considered.

The different kinds of parallelism (cf. Chapter 4 in [6]) also give room for tuning. For example, setting the number of asynchronous ioservers (prefetchers) and iocleaners to reasonable values is recommended.

Other common tasks like maintaining sufficiently current database statistics and performing reorganizations if necessary are also mandatory and cannot be replaced by choosing the right database layout.

7 Growing the Database/Log Space

Typically SAP installations reach a point where the preconfigured database space is no longer sufficient and needs to be extended. For performance and scalability reasons, it is crucial to keep the initial layout as proposed in the previous sections in mind and to continue the concept of striping across all devices.

If a DB2 table space reaches the state of being almost filled, it either has to be enlarged, or – if this is not possible – it has to be complemented by an additional table space. In the latter case, the new table space should be created according to all criteria discussed before and individual, typically quite large tables are moved from the filled table space to the newly created table space, as described in Section 5.1.

If the table space has not yet reached its 2^{24} pages size limit and it can be enlarged, then there are two approaches to add space to the table space: Adding containers or extending the existing containers. These two variants are discussed in Section 5.2. These variants will be considered under the aspect of load balancing across devices here.

The best solution for *adding* containers is to always add a full stripeset of containers to the table space. The new containers should have the same size as the existing ones to enable DB2 to evenly rebalance the data. If rebalancing is not desired, still a full stripeset should be added, but the size of the new containers must be different. If a full stripeset is not an acceptable solution, care should be taken anyway not to create hotspots. This can be achieved, e.g., by distributing the newly added space at least among a few devices. This is shown in the upper part of Figure 9: The table space initially consists of twelve containers. When enlarging the table space, three new containers are added, distributed among three devices. After adding the containers (simultaneously), the rebalancing takes place. The next extension of the table space should add another three containers, placed on the remaining three devices (indicated by hollow boxes in the figure) to complete the stripeset.

The second alternative for enlarging a table space is to *extend* all containers simultaneously. This is sketched in the bottom part of Figure 9: Each container of the table space grows by a certain amount. Extending the table space works only if all containers have the same size. The major advantage of extending the containers as opposed to adding containers is that the well-balanced placement of data is retained without rebalancing. Extending requires, however, that all volumes carrying containers have to provide enough space for the extension. If the volumes are also extended, e.g., via a logical volume manager, care must be taken to also physically keep all storage locations for each volume adjacent. It does not help to use space from a single new device to extend all existing volumes that carry containers.

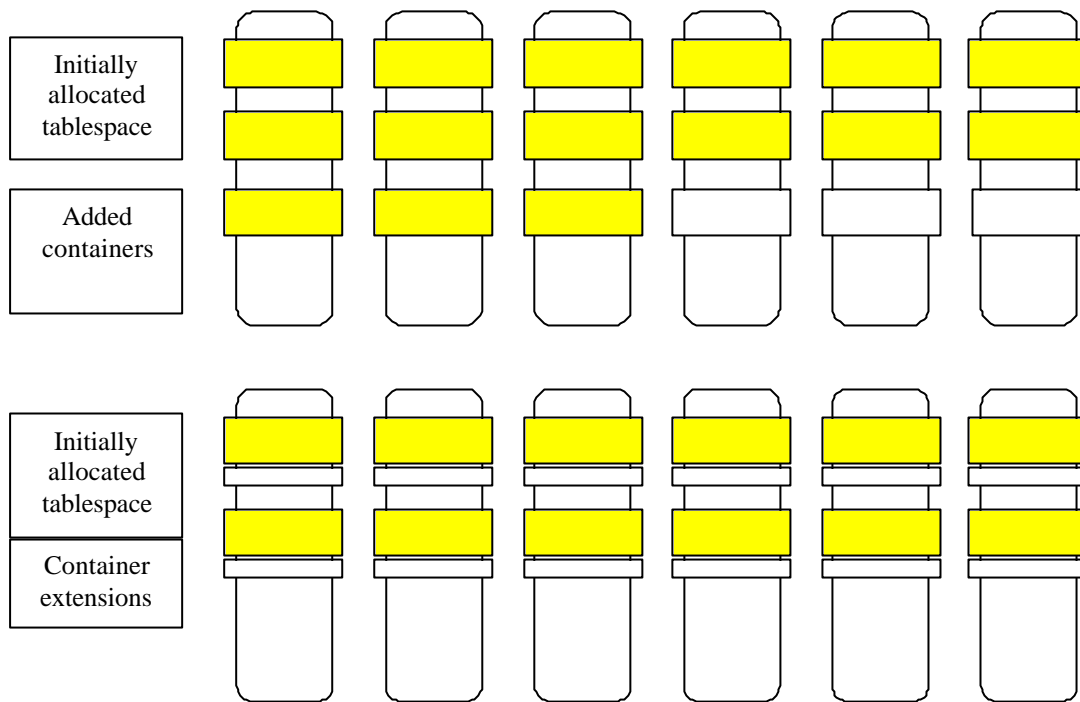


Figure 9: Several Feasible Approaches to Enlarge a Table Space

8 Miscellaneous

This section gathers detailed comments on specific operation system and storage platforms, supplemented by a few comments on EEE.

8.1 EEE Specific

DB2 UDB EEE is currently only supported for some mySAP components (see SAPnet and release notes for details). Reference [5] describes some items to consider when planning and operating a multi-partition EEE installation for BW.

8.2 AIX Specific

The AIX logical volume manager (LVM) groups physical volumes (hdisks, either physical disks attached to the host or virtual disks provided by a storage system) into volume groups. Logical volumes are allocated within a single volume group. Filesystems are placed either implicitly or explicitly (by first allocating the logical volume and then creating a filesystem on top of it) on top of a logical volume. Raw devices (e.g., for UDB table space containers) are placed directly into logical volumes.

Logical volumes cannot belong to more than one volume group. In order to allow the striping of filesystems across all available devices, the physical volumes should be collected in such a way into volume groups that physical volumes of many different disk adapters and RAID volumes meet in a single volume group. Referring to the ESS example, a volume group should cross all ranks available. In general, having as few volume groups as possible yields maximum flexibility for logical volume layout. If single logical volumes are allocated for table space containers, the volume group layout is less important.

There are multiple alternatives for allocating logical volumes in the AIX logical volume manager. First, the range of physical volumes can be chosen between minimum (try to concentrate the logical volume on as few physical volumes as possible) and maximum (try to evenly distribute the logical volume across all physical volumes). The latter alternative distributes the physical partitions (chunks of 8-32MB) by a round robin strategy between the physical volumes. This alternative is recommended at least for the IBM ESS. Derived from the unit of storage allocation (physical partitions with size of 8-32MB) this is also called "PP-striping". Alternatively, the LVM also offers striping in units of 4-128kB. This is not recommended since DB2 already performs extent-bases striping in this granularity. Another option for allocating logical volumes is to restrict

them to single physical volumes within a volume group. This is useful when allocating space for single UDB containers.

In order to get access to all LVM-options for logical volumes, file systems should be in a two-step approach: First allocate the logical volume and choose the appropriate options for the logical volume. Then create the file system on top of the existing logical volume. Letting the administration tool automatically create the logical volume hides some options for the logical volume. Furthermore, when creating filesystems care should be taken to use "large file enabled" file systems wherever needed (e.g., for temporary SMS table spaces).

8.3 IBM ESS Specific

The IBM Enterprise Storage Server (ESS) has already been mentioned multiple times in the paper and it has been used for the configuration examples. There is plenty of information on the IBM ESS web site [9]. In the following, a few additional recommendations are collected:

Try to use as many ranks as possible for each SAP system. Each rank has only a certain bandwidth capacity, so using many ranks makes the cumulative bandwidth of all ranks available to the system.

Each rank is primarily assigned to one of the two clusters. Try to balance the number of used ranks between Cluster 1 and 2. This way, the resources (e.g., cache and NVRAM) are equally exploited on both clusters. Similarly, try to use ranks from as many device adapter pairs and SSA loops as possible. This again yields the cumulative bandwidth of all resources.

The size of the volumes allocated on the ESS does not influence performance. To limit the number of physical volume to administrate on host side (and thereby to limit the number of volume groups), the ESS volumes should be chosen large enough. ESS volumes that belong to the same kind of data (e.g. a single UDB table space) should be assigned to as many SCSI/FC host adapters as possible. Once again, this offers the cumulative bandwidth of the host adapters to a single table space. The flexibility of assigning ESS volumes to hosts systems, host adapters, and finally volume groups increases with smaller ESS volumes. A reasonable volume size would be 8-25GB. Volume size starting at ≥ 16 GB should be preferred to simplify administration.

The subsystem device driver (SDD, formerly DPO) can be used for load balancing and failover purposes. It offers the option to use more than one SCSI/FC path from an ESS volume to the host system. Failure of one path (e.g., SCSI cable or host adapter failure) keeps the volume available. If the allocation of ESS volumes to host adapters is well balanced, the SDD does not yield performance gains (but still increased availability).

The ESS offers copy services for local and remote volume copy. IBM offers a split mirror backup solution employing these copy services for UDB that allows low impact backup and recovery solutions. A forthcoming whitepaper from IBM describes the solutions in detail.

9 Summary

This paper discusses multiple aspects to consider when designing the storage layout of a SAP installation based on DB2 UDB. Several alternatives are described to circumvent the table space size limitation. The most prominent are choosing large page sizes and isolating tables into different table spaces. As fundamental concept to get maximum performance from the whole system installation striping across all devices available is proposed. Details on how to implement this concept are pointed out for the different aspects of DB2 objects. In summary, this paper should help to circumvent many pitfalls of a SAP installation with DB2 UDB.

10 References

1. SAP Bluebook "Fundamentals of Database Layout".
<http://ency.wdf.sap-ag.de:1080/Bluebooks/BME048.02-FundamentalsOfDatabaseLayout.pdf>
2. "Database Layout for R/3 Installations under Oracle" <http://service.sap.com/atg> or
<https://www005.sap-ag.de/~sapidb/011000358700003877782000E.pdf>

3. "Database Layout for SAP Installations with Informix" <http://service.sap.com/atg> or <https://www005.sap-ag.de/~sapidb/011000358700005530252000E>
4. SAP Note 136702: "[Moving tables to other DB2 table spaces](#)" <http://service.sap.com/notes>
5. "DB2 UDB EEE on UNIX & Windows NT: SAP BW Administration Tasks in Multi-Partition Installations"; SAP Documentation
6. "DB2 Administration Guide: Performance Version 7" SC09-2945-00
7. "DB2 UDB V7.1 Performance Tuning Guide", IBM Redbook SG24-6012-00, <http://www.redbooks.ibm.com/abstracts/sg246012.html>
8. Split Mirror References: Forthcoming ...
9. IBM Enterprise Storage Server Home Page: <http://www.storage.ibm.com/hardsoft/products/ess/ess.htm>