Best Practices for Partitioning Data in InfoSphere Warehouse Environment

Session IDW-1820B

Adriana Carvajal, IBM

IBM Software

Information On Demand 2011

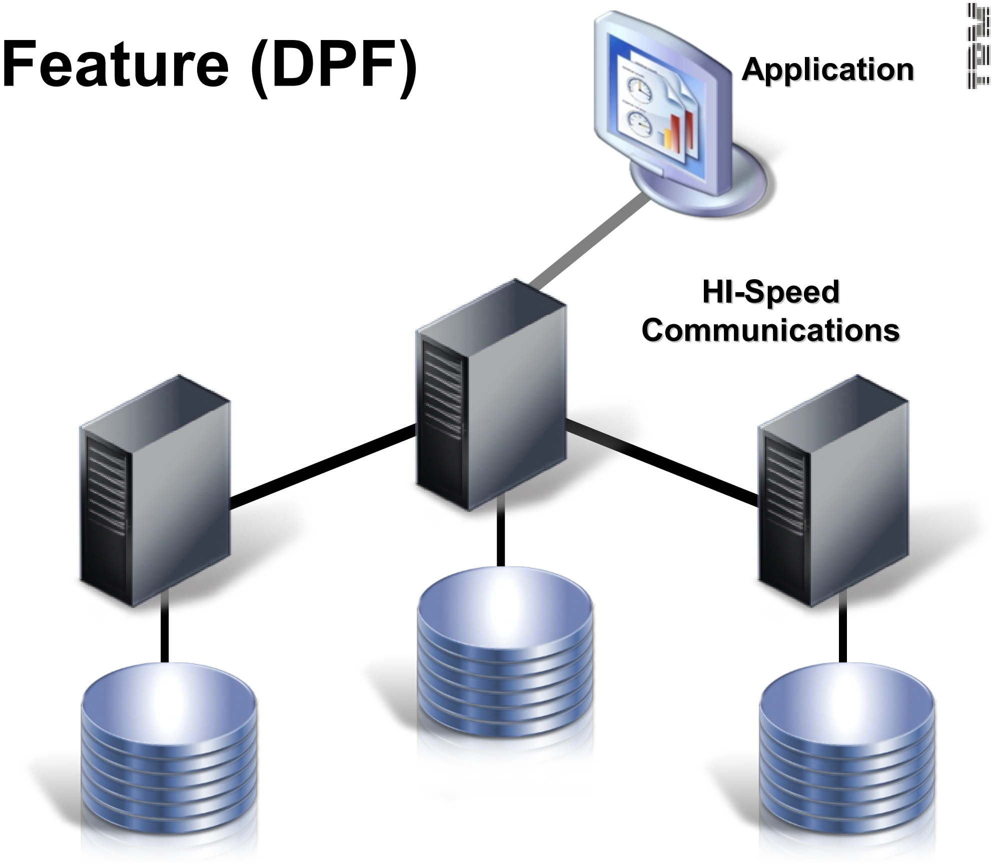
DB2 Partitioning Features

- DATABASE PARTITIONING **(DPF)**
  - **DISTRIBUTE BY HASH**
  - **Suitable for:** Large tables such as Fact tables, they often contain hundreds of millions or billions of rows.
  - **Benefit:** Hardware Parallelism for best performance and scalability
- TABLE PARTITIONING **(Sometimes called "Range Partitioning")**
  - **PARTITION BY RANGE**
  - **Suitable for:** Tables where large volumes of rows are added or removed periodically. In Fact tables, new data are often added daily and obsolete data removed usually monthly or quarterly.
  - **Benefit:** High performance during Roll in / Roll out while keeping table online.
- MULTI-DIMENSIONAL CLUSTERING **(MDC)**
  - **ORGANIZE BY DIMENSION**
  - **Suitable for:** Optimal physical clustering for prefetch queries whose result sets returns rows with similar values along multiple dimensions (BI/OLAP)
  - **Benefit:** Significant query performance over traditional INDEX based optimization. No REORG requirement.

All these partitioning features can be used simultaneously on the same table and are three significant technologies for managing warehousing databases.

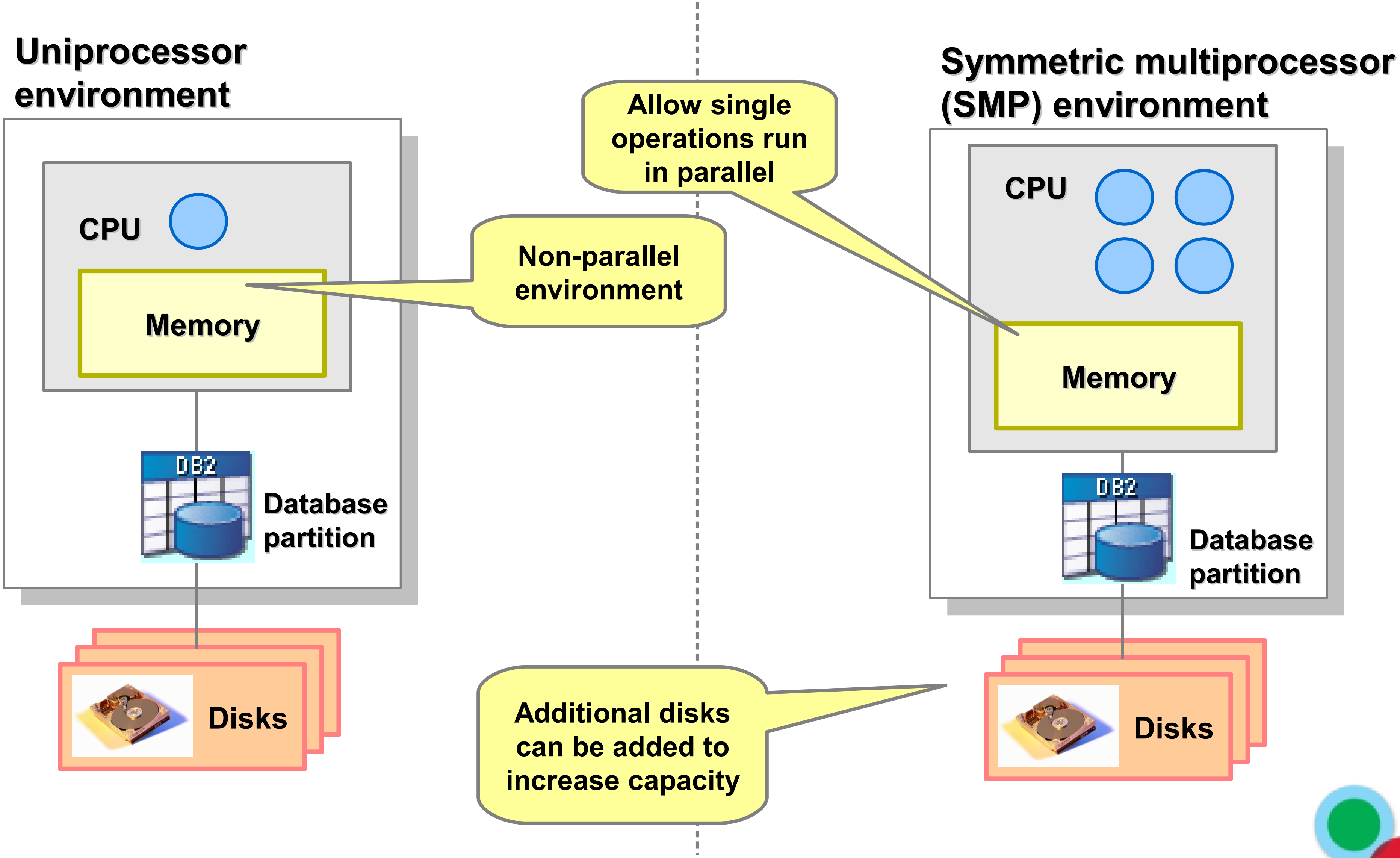
DB2 Database Partitioning Feature (DPF)

- Enables to evenly distribute rows across **database partitions (nodes).**
- Why Partition?
  - Scale Out, Performance, …
- Benefits
  - Transparent to users and applications
  - Parallelism (divide and rule)
    - Workload is divided among all nodes
    - Asynchronous I/O Parallel I/O
    - Dynamic throttling based on load
  - Near linear scalability
    - As the table grows, add more processing power in form of additional database partitions.



- DB2 core **Scale-Out** architecture based on **Parallelism** aka **Shared Nothing** architecture
  - Ability to spread all data across multiple server, **each database partition has its own set of computing resources**, including CPUs, memory, disk controllers and disks.

Single Partition (one database partition)



Uniprocessor environment

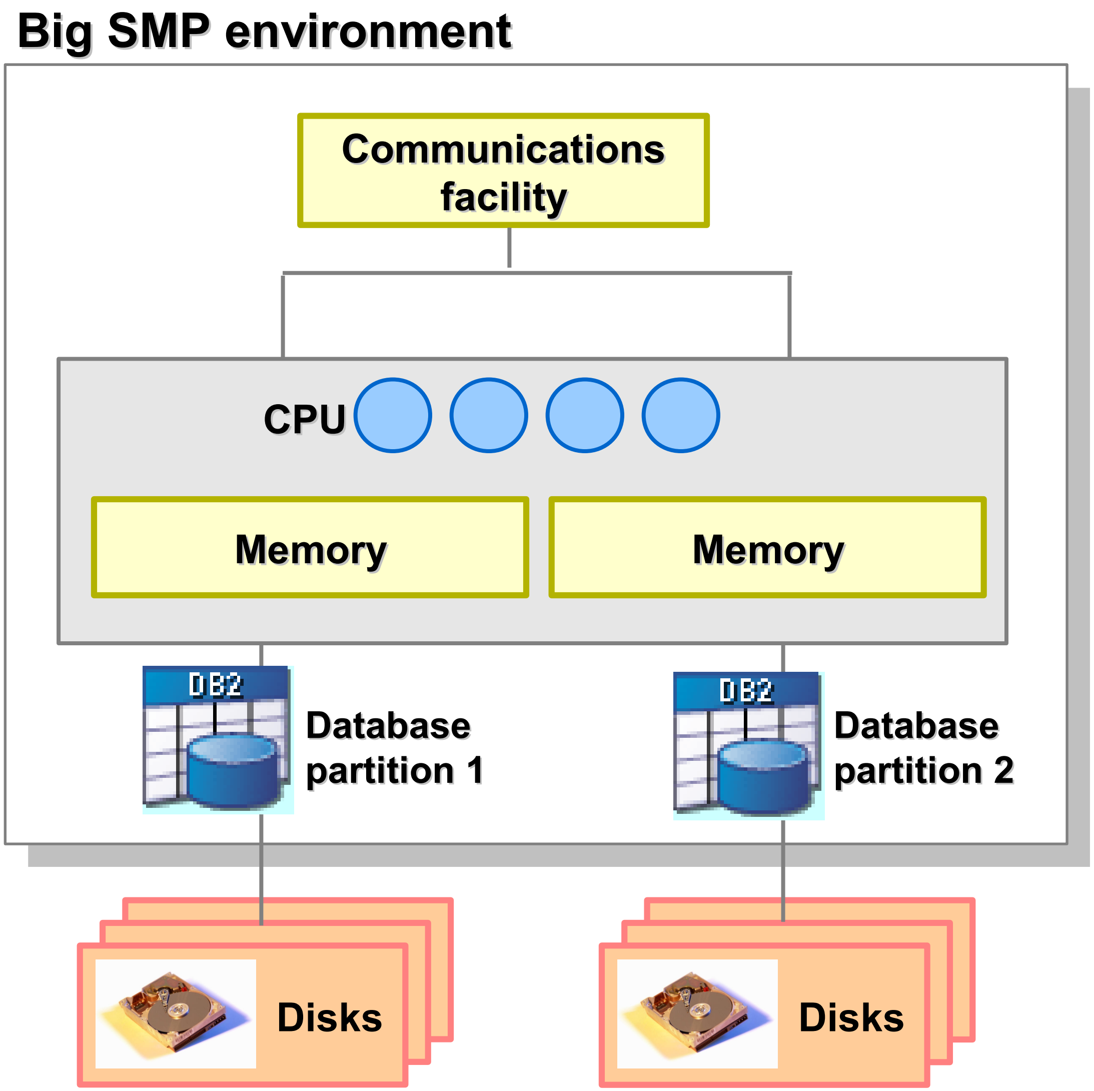
Symmetric multiprocessor (SMP) environment

Allow single operations run in parallel

Non-parallel environment

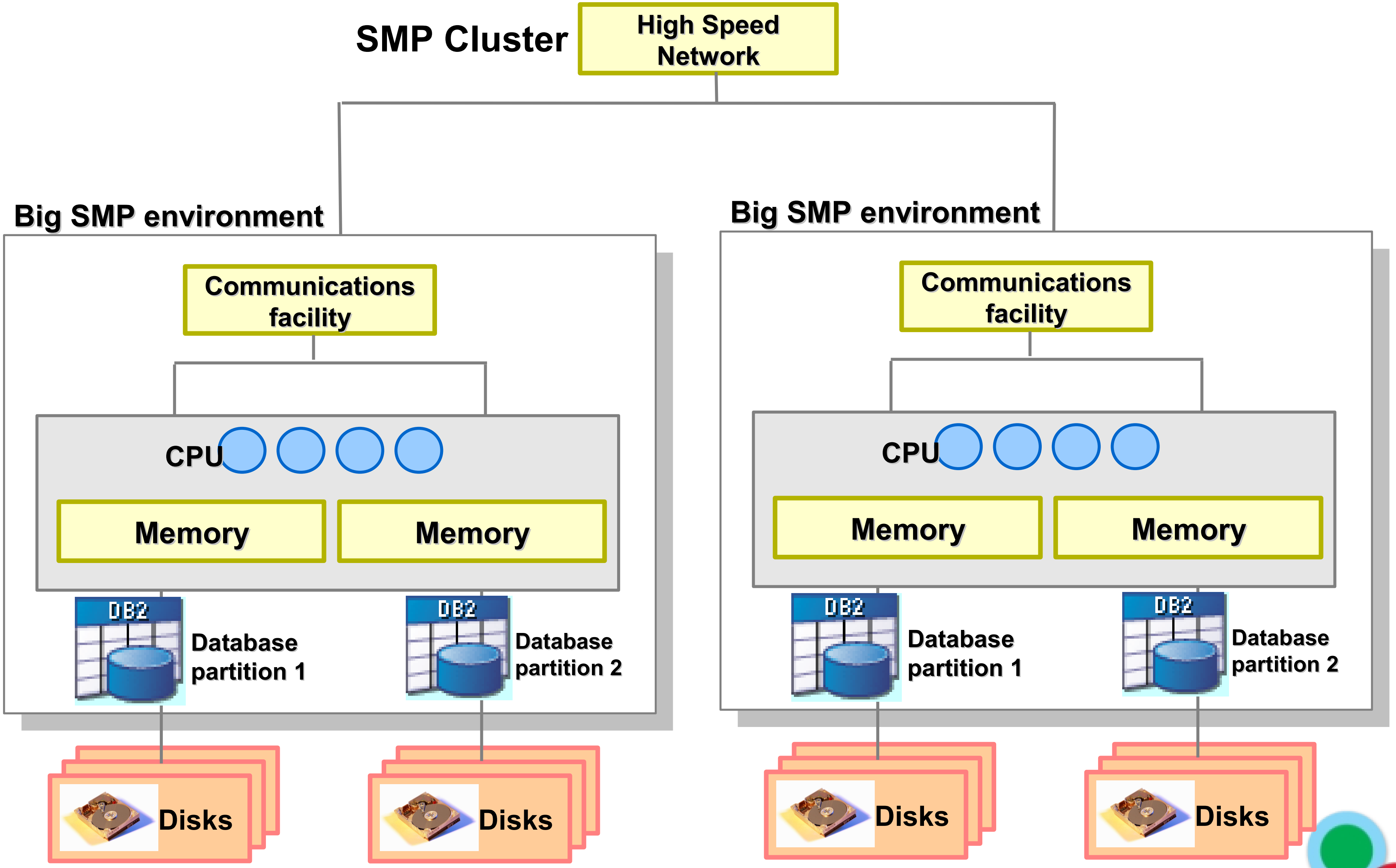
Additional disks can be added to increase capacity

Logical Partitions (multiple partitions per machine)



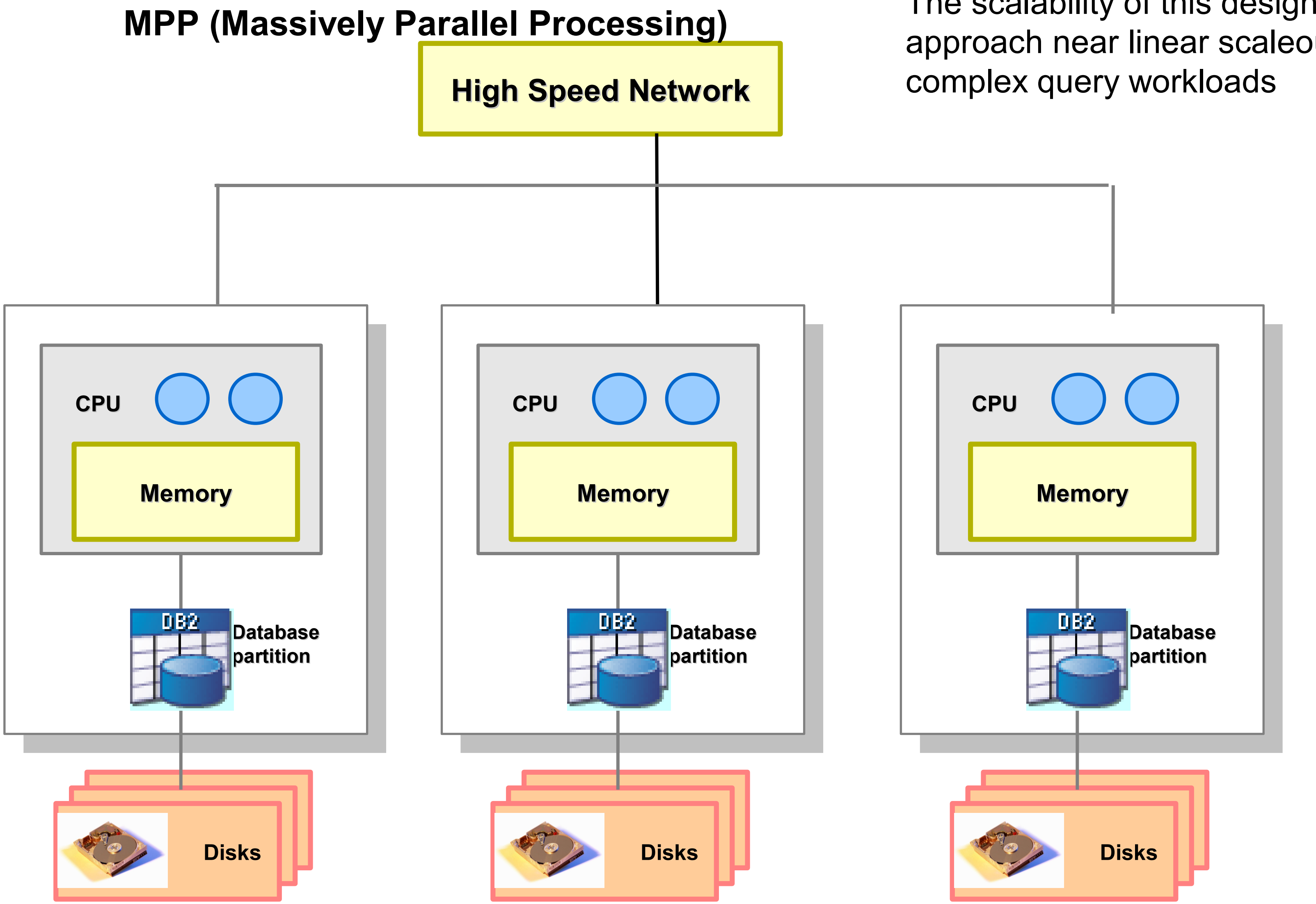
- More than one database partition on a server
- Processors are shared
- Memory and disks are not shared

Several SMPs loosely coupled using a network



Multiple DB Partition (one partition per machine)

The scalability of this design can approach near linear scaleout for many complex query workloads

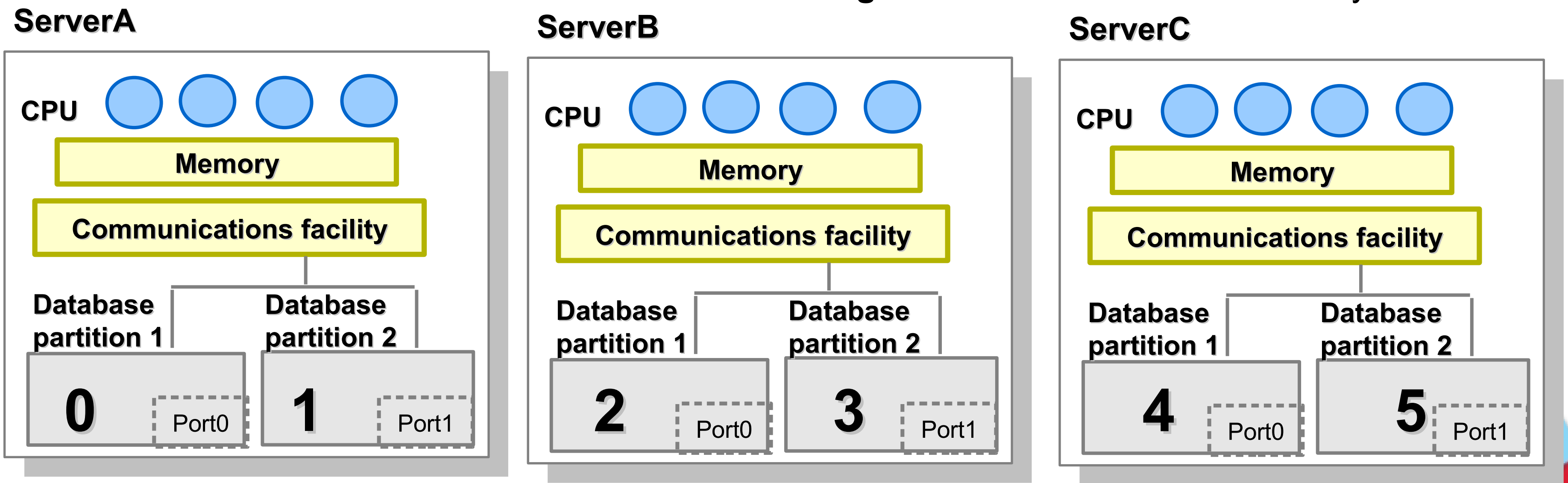


DB2 Node Configuration on a DPF Environment

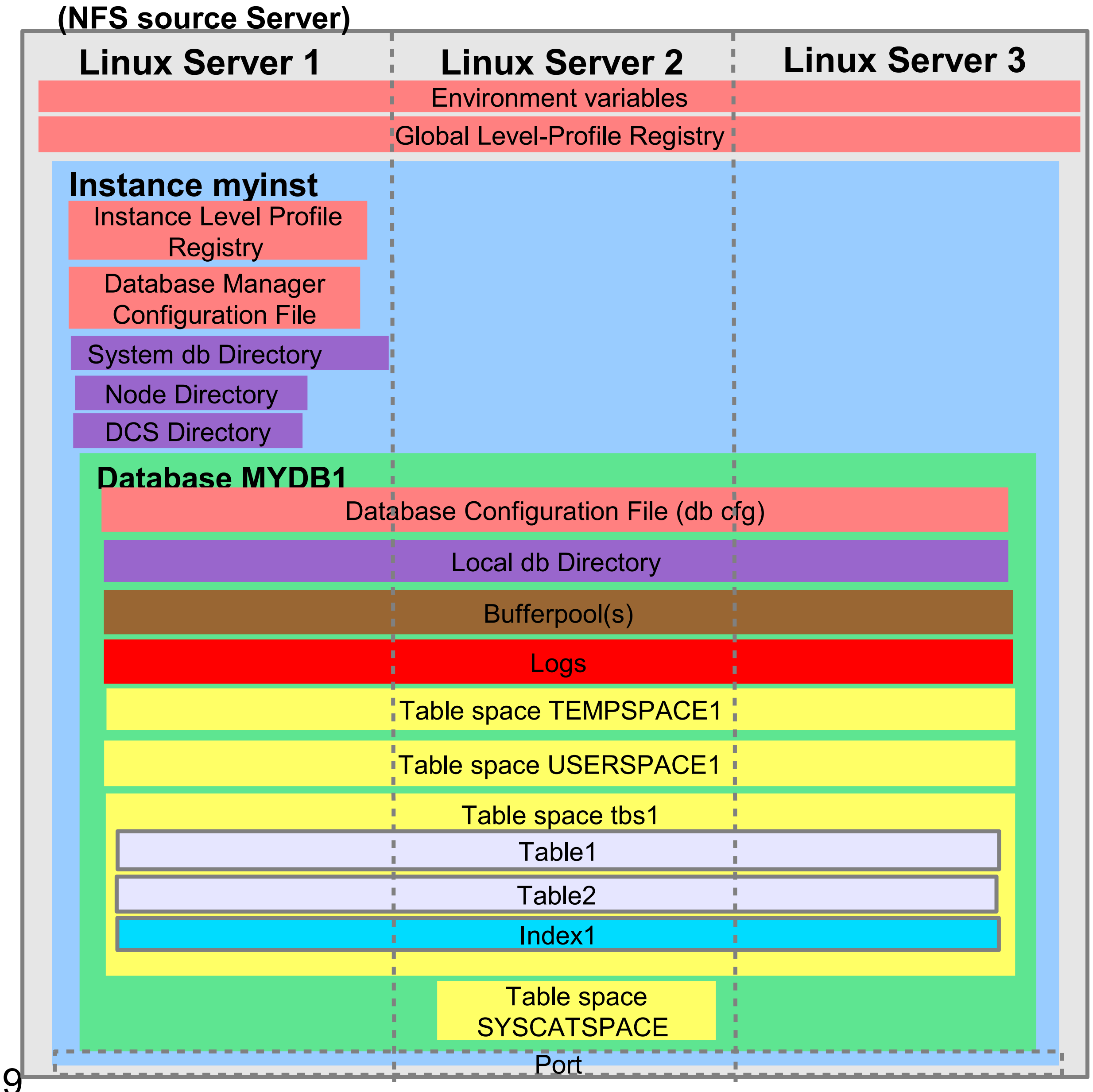
DB partition num	Host Name	Logical Port
0	ServerA	0
1	ServerA	1
2	ServerB	0
3	ServerB	1
4	ServerC	0
5	ServerC	1

Defined in the **db2nodes.cfg** file

- Necessary parameters:
  - **dbpartitionnum**: Unique database partition ID
  - **hostname**: Machine's name or IP address
  - **logical-port**: Logical partition ID within a machine
- db2nodes.cfg must be located:
  - SQLLIB directory (Linux and UNIX)
  - SQLLIB\instance_name directory (Windows)
- On Windows, only can use **db2ncrt** and **db2ndrop** commands to create and drop database partitions; the **db2nodes.cfg** file should not be edited directly.



How DB2 Environment is split on DPF



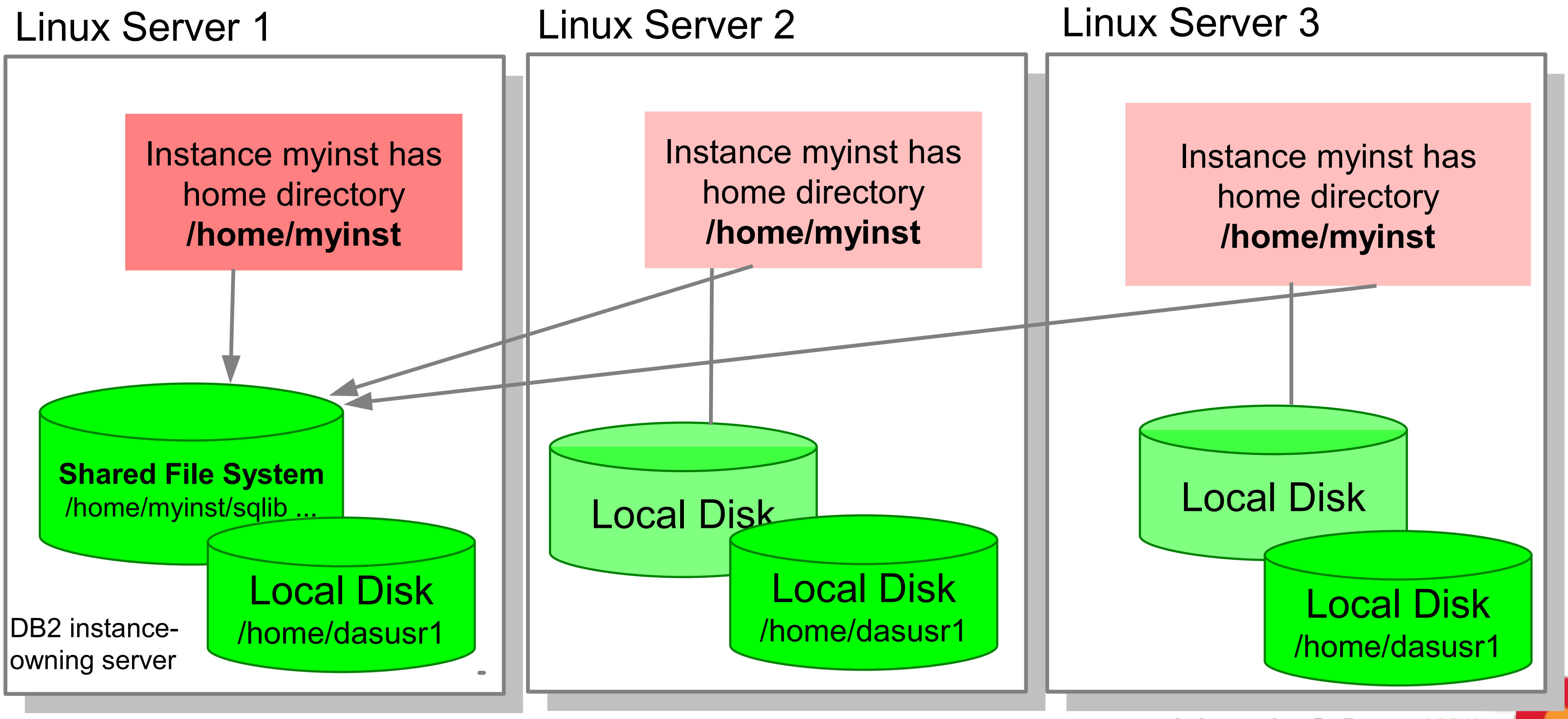
Visualize how a DB2 environment is split in a DPF system

- All partitions share
  - Instance level profile registry
  - Database manager config file (**dbm.cfg**)
  - System db directory
  - Node directory
  - DCS directory
- Each server can have its own
  - Environment variables
  - Global-level profile registry variable
  - Database configuration file
  - Local database directory
  - Log files

Instance on a DPF Environment

Partition a database, not an instance.

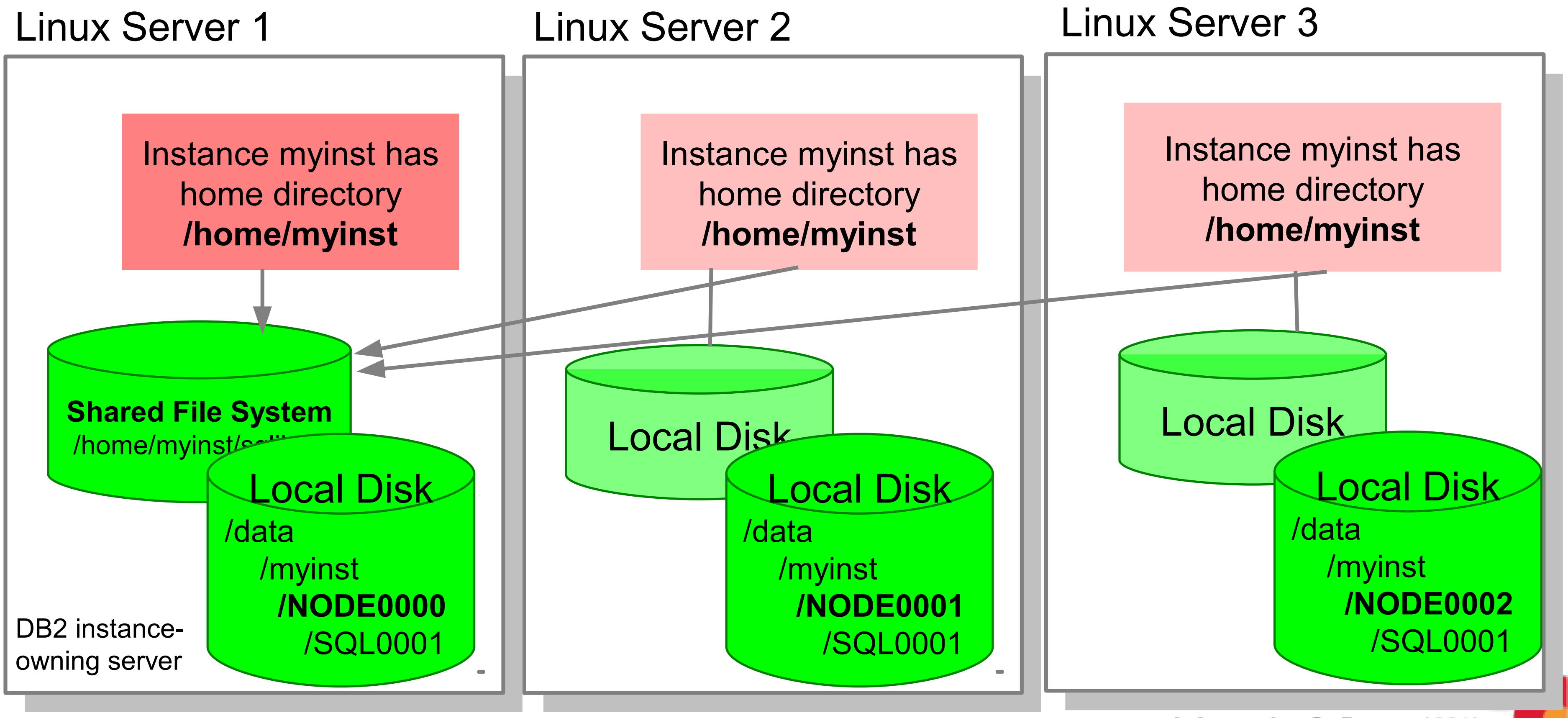
- In a DPF environment an instance is created once on an NFS source server. The instance owner's home directory is then exported to all servers where DB2 is to be run.
- Make sure the passwords for the instances are the same on each of the servers in a DPF Environment, otherwise the partitions are will not be able to communicate



A Database in a DPF Environment

To partition a database in a DPF environment, we recommend that you create a directory with the same name, locally in each of the servers.

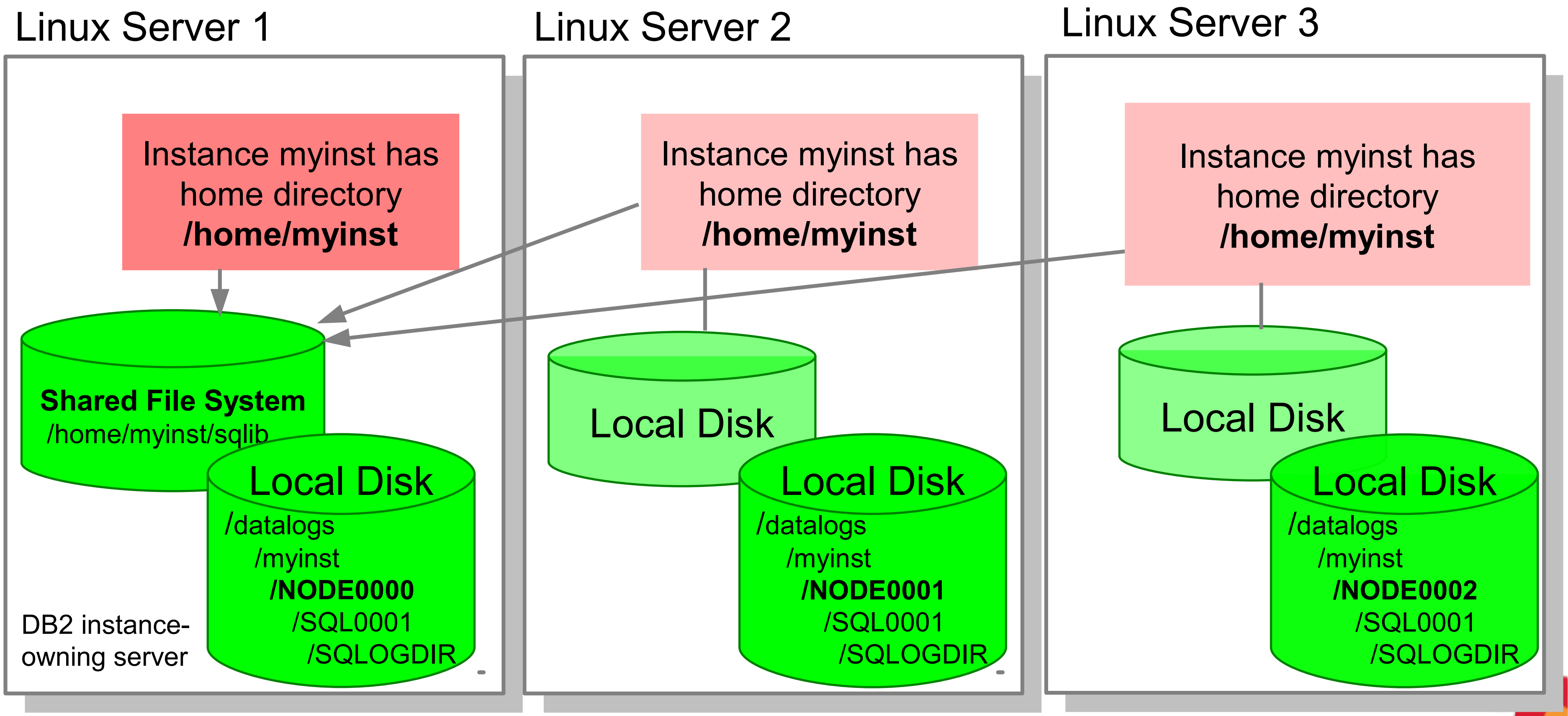
- Then make sure to include this path in your command `CREATE DATABASE mydb on /data`
- Or to simply issue the `CREATE DATABASE mydb` be sure to change the value of **dbm cfg** parameter, **DFDBPATH**, to include this path.



Logs in a DPF Environment

The Logs on each partition should be kept in a separate filesystem different than the database location.

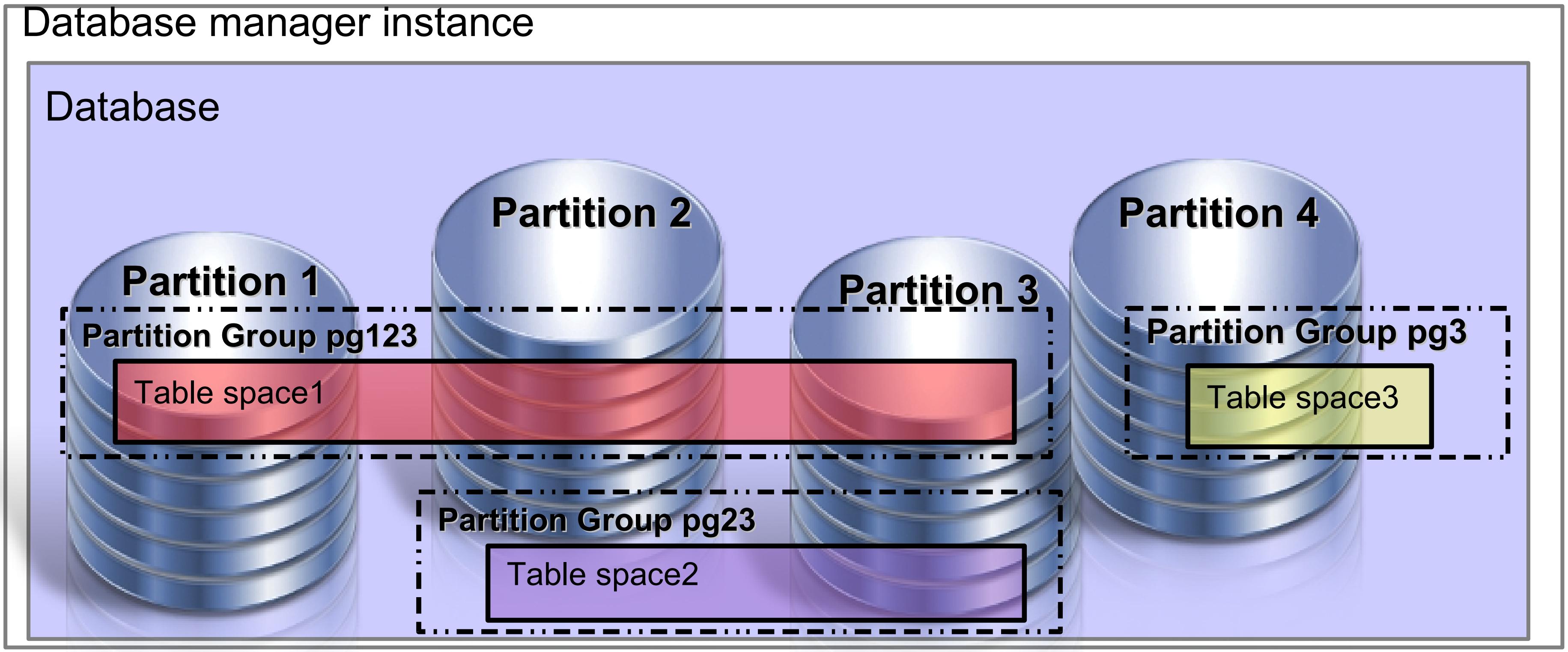
- The database configuration parameter **LOGPATH** on each partition should point to a local file system, not a shared file system.
- To change the path for the logs, update the database configuration parameter **NEWLOGPATH**



Database Partition Groups in a DPF environment

A Logical Layer that:

- Allows the grouping of one or more database partitions.
- Allows table spaces to span on different partitions



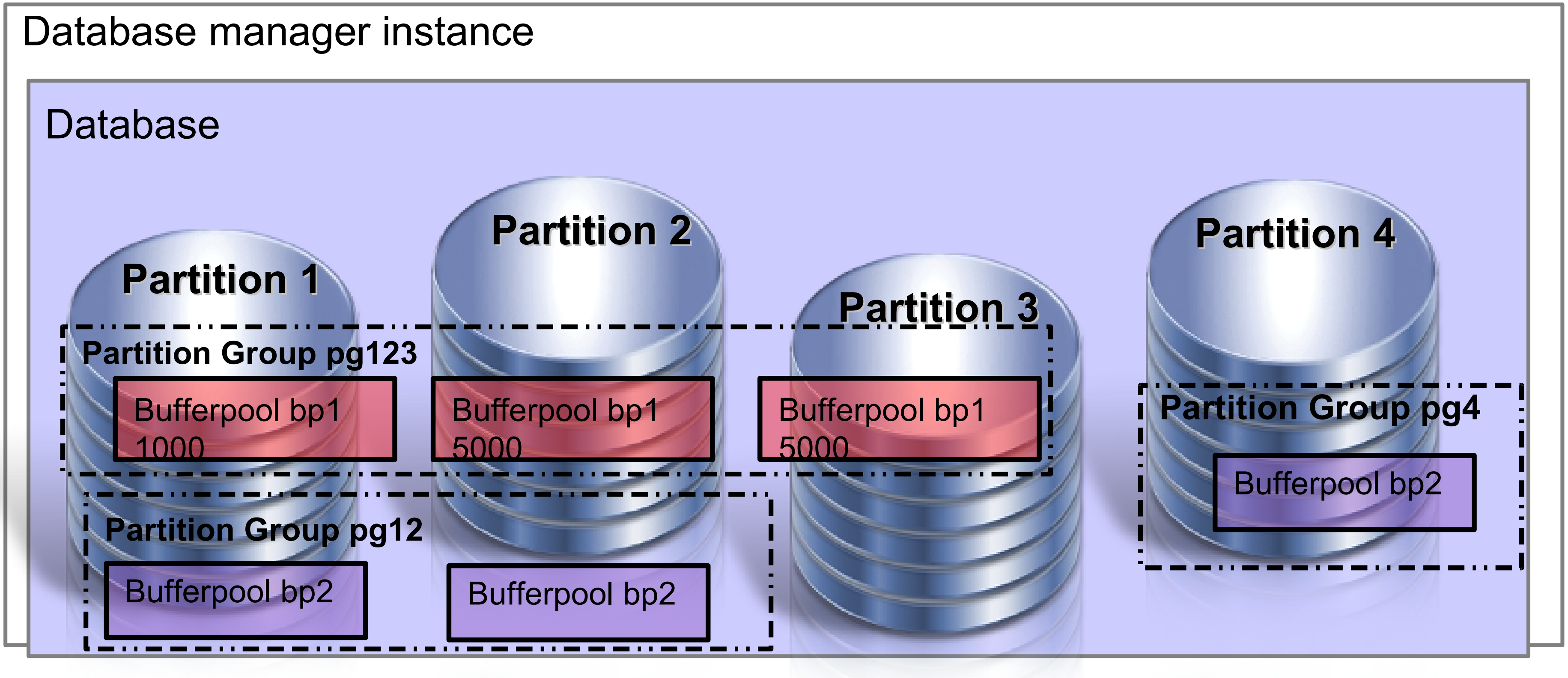
```
db2 CREATE DATABASE PARTITION GROUP pgrpall ON ALL DBPARTITIONNUMS
```

```
db2 CREATE DATABASE PARTITION GROUP pg123 ON DBPARTITIONNUMS (1,2,3)
```

Bufferpools in a DPF environment

The data cached in the bufferpools is not partitioned

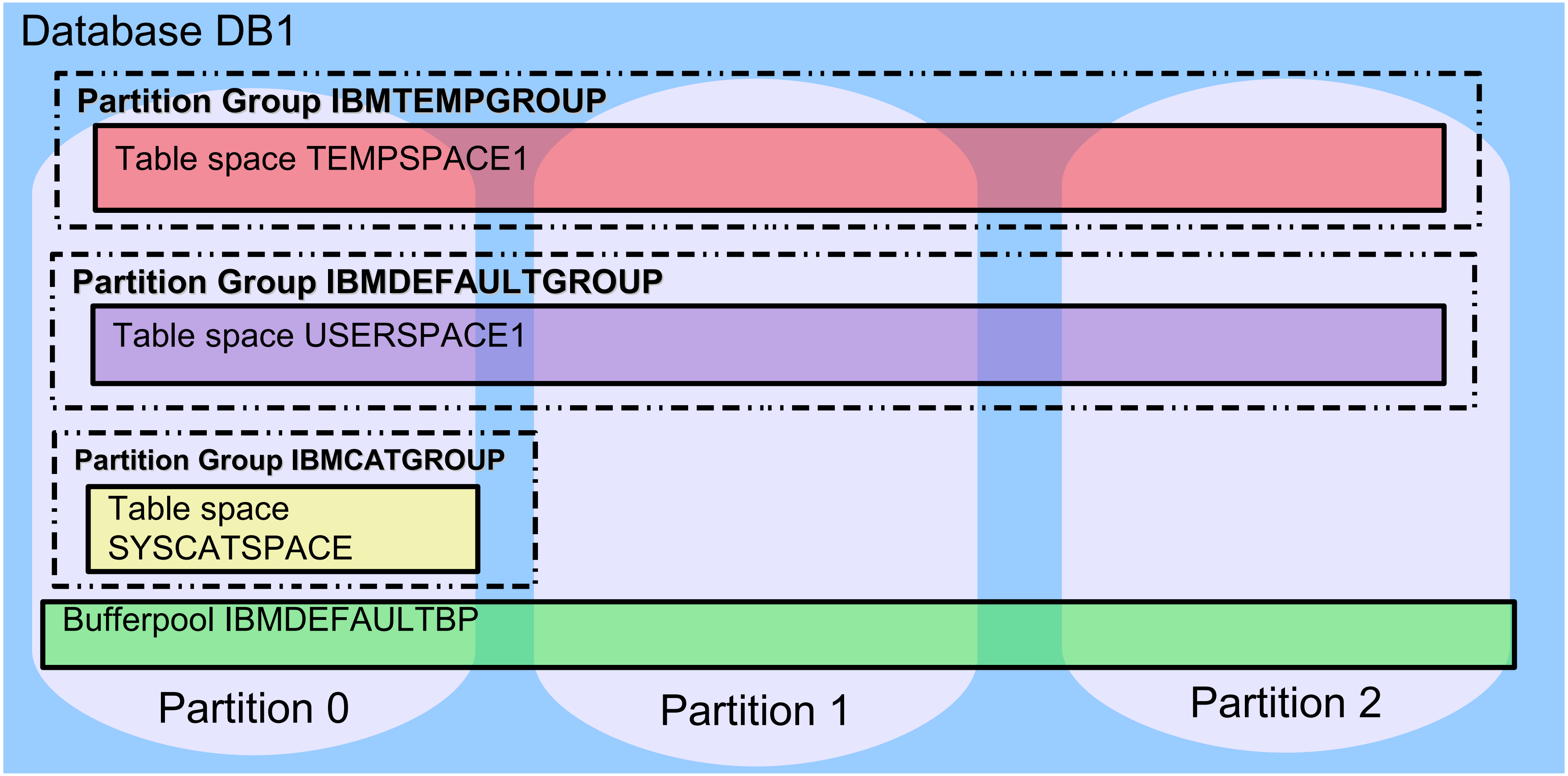
- Each bufferpool in DPF holds data only from the database partition where the bufferpool is located
- You can have the flexibility to define a buffer pool on the specific partitions defined in the partition group
- Bufferpools can also be associated to several partition groups



```
db2 CREATE BUFFERPOOL bp1 DATABASE PARTITION GROUP pg123 SIZE 1000
    EXCEPT ON DBPARTITONNUM (2 to 3) SIZE 5000
```

Creating the Database in a DPF Environment

Database Objects created by default



```
db2 CREATE DATABASE db1 ON drive(s)/path(s)
```

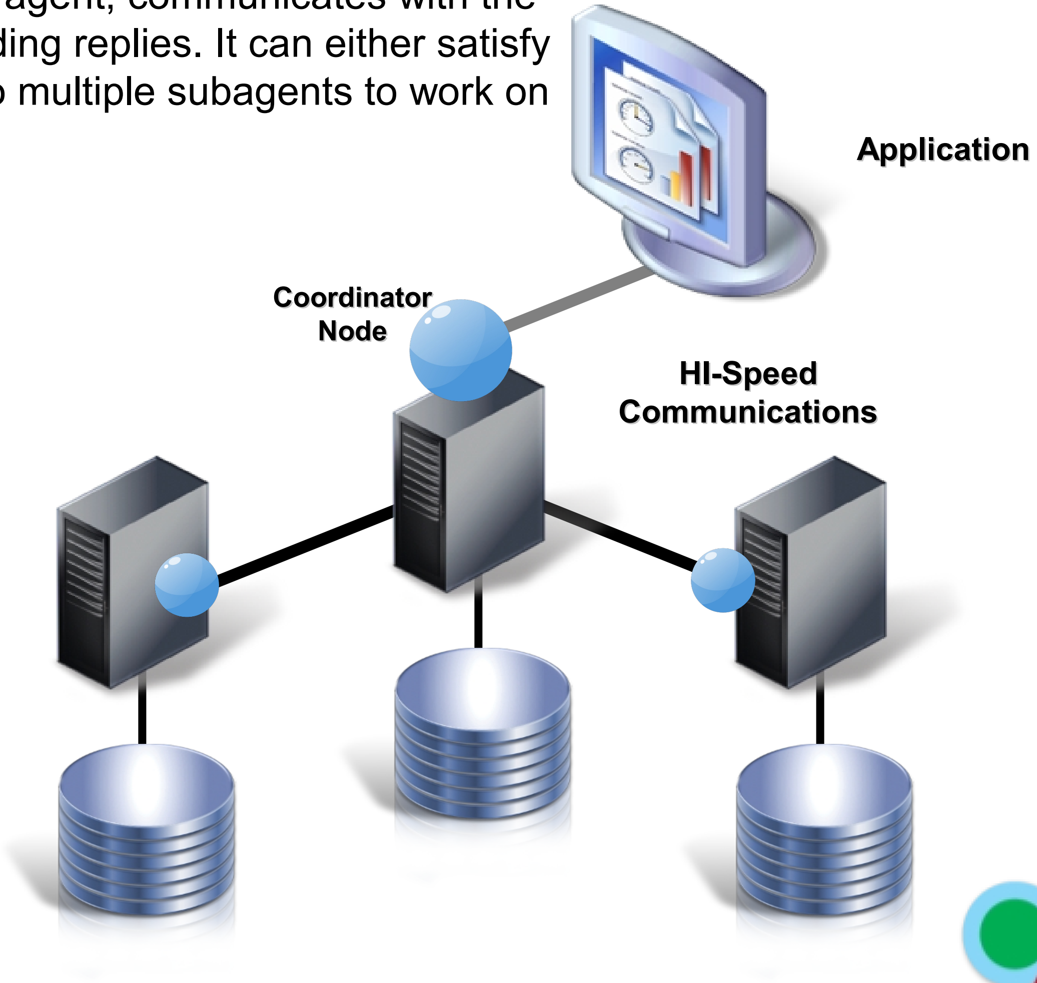
- The **CREATE DATABASE** command in a multipartition environment automatically takes the contents of the partition configuration file (**db2nodes.cfg**) into consideration.
- If you don't explicitly connect to a database partition or server, the database will be created with the system catalogs on the first partition in the **db2nodes.cfg**

The Coordinator Partition

The partition where the application connects

- Each database connection has a corresponding DB2 agent handling the application connection. The coordinator agent, communicates with the application, receiving requests and sending replies. It can either satisfy the request itself or delegate the work to multiple subagents to work on the request

- The coordinator partition of a given application is the partition where the coordinator agent exists. Any partition can potentially be a coordinator
- Use the **SET CLIENT CONNECT_NODE** command to set the partition that is to be coordinator partition
- Single system view management
  - Administrative commands and application code are transparently propagated to all partitions



How Data is Distributed on a DPF Environment

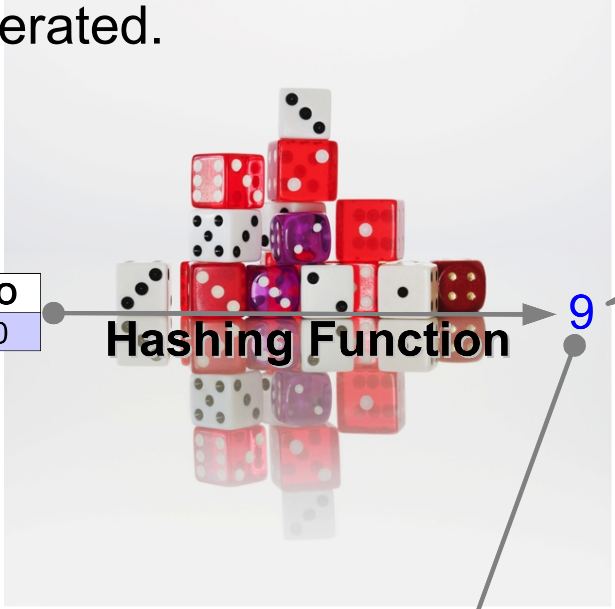
- The **Distribution Map** is an internally generated array used for deciding where data will be stored within the partitions
  - Partition numbers are specified in a round-robin fashion in the array
  - **New to DB2 9.7**: Grown from 4096 (4 KB) entries to 32 768 (32 KB) entries
- The **Distribution Key** is a column(s) that determines the partition on which a particular row of data is physically stored
  - Define key using **`CREATE TABLE`** statement with the **`DISTRIBUTE BY`** clause
  - Design Advisor can be used to suggest an optimal distribution key
- The **Hashing Algorithm** generates a value between 0 and 32 767 based on the distribution key

How does DPF distribute rows?

Distribution maps and distribution keys

- When a database partition group is created, a distribution map is generated.

EMPNO
000120



A distribution key is a column (or a group of columns) that is used to determine the partition in which a particular row of data is stored.

The distribution key value is hashed to generate the partition map index value. The hash value ranges from 0 to 32767

4096 entries (with 8 repeats) if **DB2_PMAP_COMPATIBILITY=ON**

Distribution Map

Index	0	1	2	3	4	5	6	7	8	9	10	11	…	…	…	…	…	32766	32767
Partitions	1	2	4	1	2	4	1	2	4	1	2	4	…	…	…	…	…	…	…

A distribution map is an array of 32768 entries, each of which maps to one of the database partitions associated with the database partition group

DB2 uses a round-robin algorithm to specify the partition numbers.

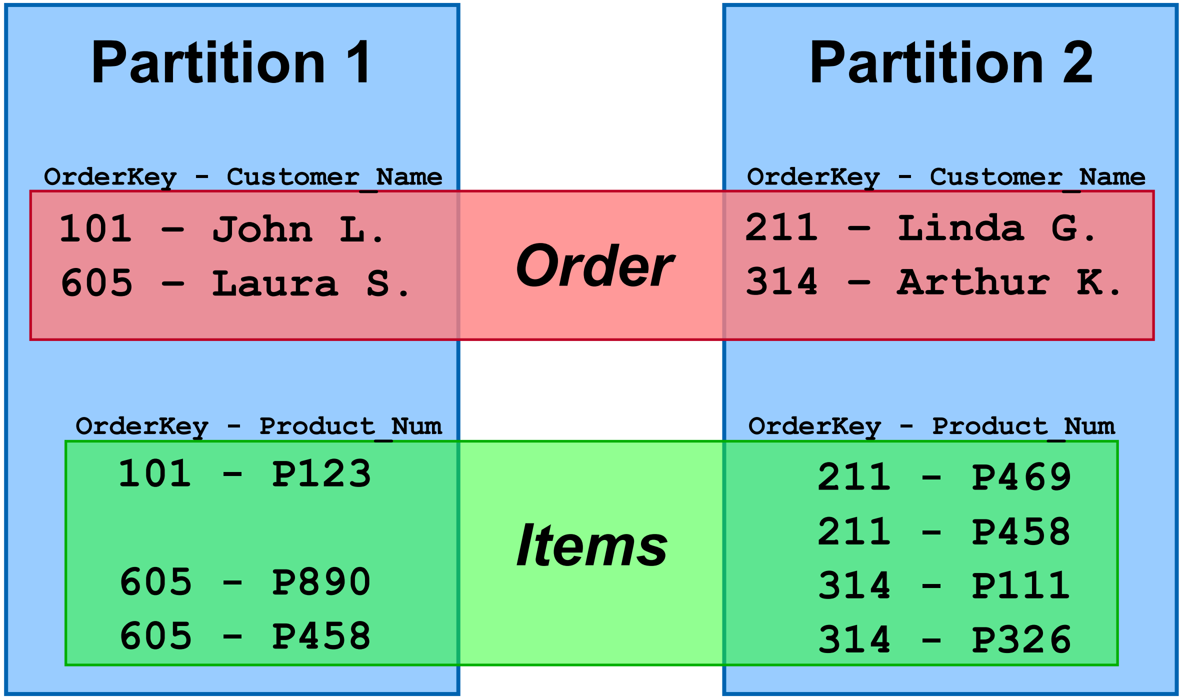
Partition1	Partition2	Partition3	Partition4

Selecting Distribution Keys

- The primary decision choice is determining which columns to use to hash partition each table. To choose a good distribution key candidate, consider the following rules:
  - Columns that have a large number of different values (high cardinality) to ensure an even distribution of rows across all database partitions in the database partition group.
  - Unique keys are good candidates
  - Integer columns are more efficient than character columns, which are more efficient than decimal.
  - Use the smallest number of columns possible.
  - No long fields or XML columns allowed
- Having an inappropriate distribution key can cause uneven data distribution. This can cause the database manager to ship large amounts of rows between partitions

Collocated JOIN

```
SELECT Customer_Name, Product_Num
  FROM Order, Items
 WHERE Order.OrderKey = Items.OrderKey
   AND Qty > 1000;
```



Partition 1

OrderKey - Customer_Name

101 – John L.
605 – Laura S.

OrderKey - Product_Num

101 - P123

605 - P890
605 - P458

Order

Items

Partition 2

OrderKey - Customer_Name

211 – Linda G.
314 – Arthur K.

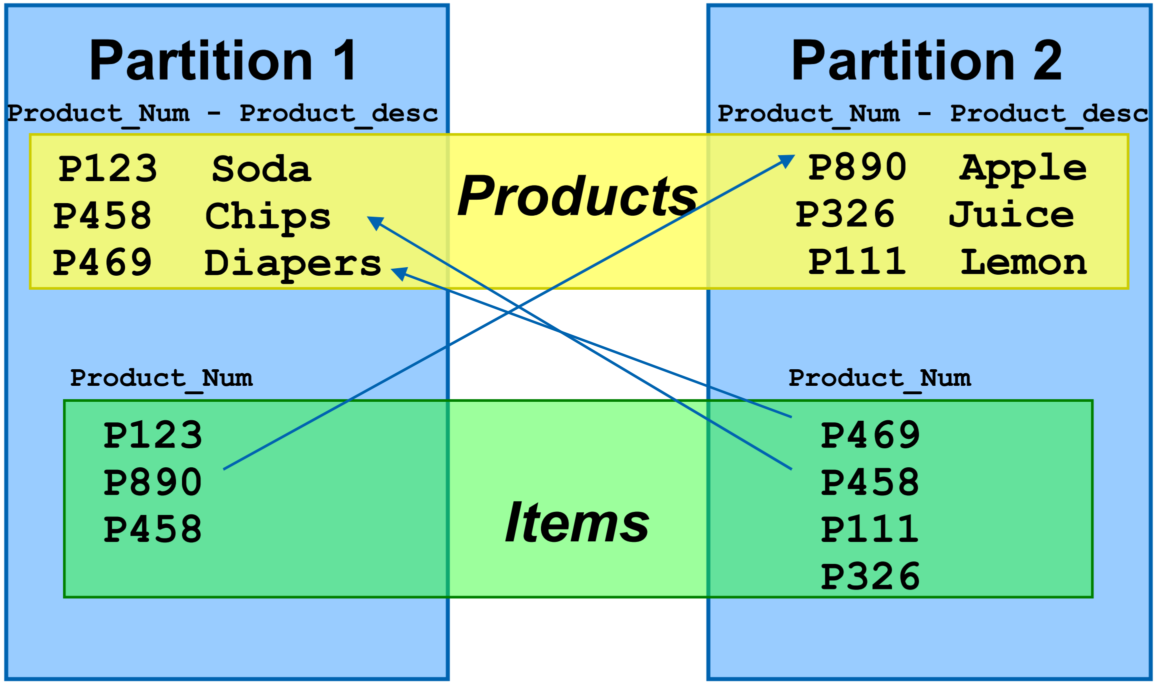
OrderKey - Product_Num

211 - P469
211 - P458
314 - P111
314 - P326

Collocated Join
Orders and Items on Same Partition
All Joins are Local

Directed OUTER (or INNER) JOIN

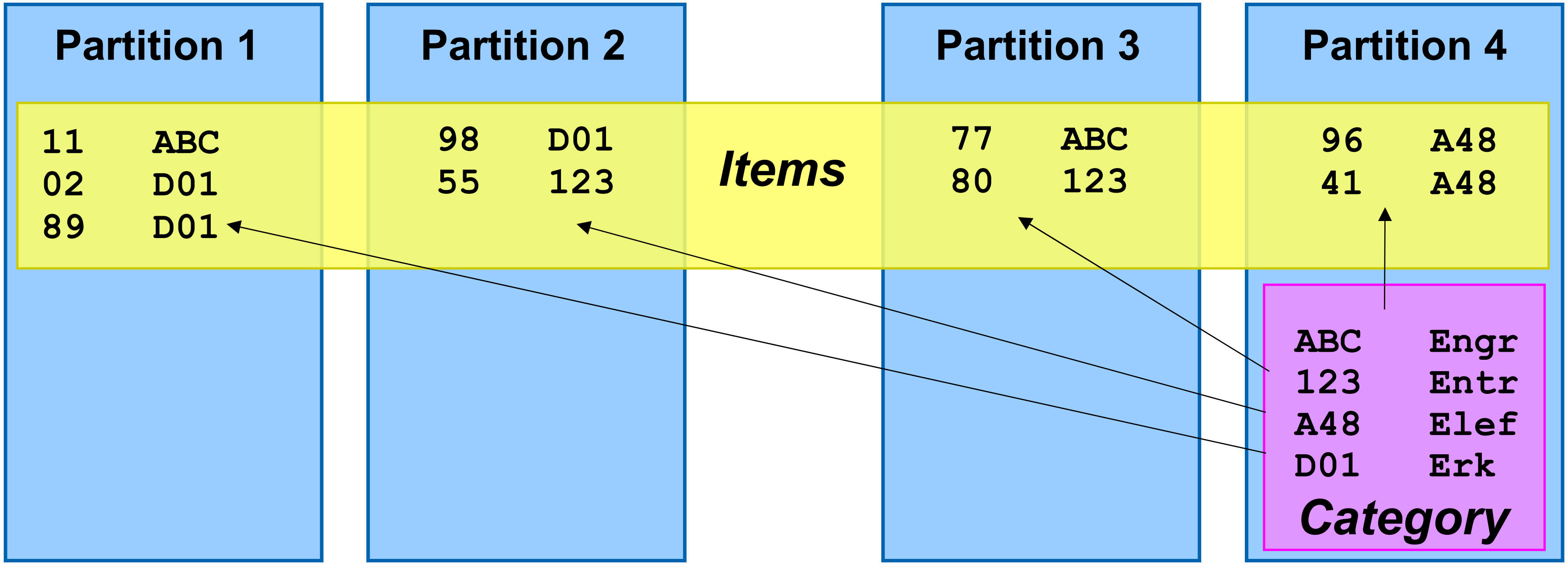
```
SELECT Product_Num, Product_desc
  FROM Items, Products
 WHERE Items.Product_Num = Products.Product_Num
   AND Items.Qty > 1000;
```



Directed Outer (or Inner) Join
Re-Hash Items to Widgets Partitions
Repartitions Items Table on
Distribution Key of Products Table

Broadcast OUTER (or INNER) JOIN

```
SELECT Item_Nbr, Qty, Category_descr
  FROM Items, Category
 WHERE Items.Category_code = Category.Category_Code
   AND Category_descr LIKE 'E%'
```



Partition 1

11 ABC
02 D01
89 D01

Partition 2

98 D01
55 123

Items

Partition 3

77 ABC
80 123

Partition 4

96 A48
41 A48

ABC Engr
123 Entr
A48 Elef
D01 Erk

Category

Broadcast Outer (or Inner) Join
All Resultant Rows of one Table sent
to all Partitions of other Table

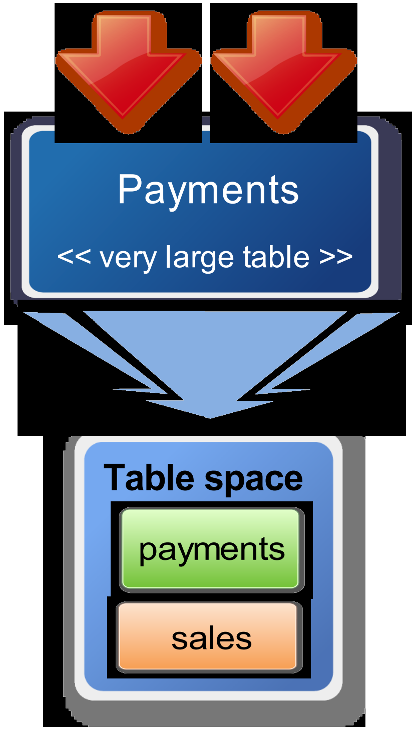
Selecting distribution keys (cont)

- Having an inappropriate distribution key can cause uneven data distribution. This can cause the database manager to ship large amounts of rows between partitions:
  - Avoid choosing a partitioning key with a column that is updated frequently; this could incur additional overhead on the update to repartition the row to another partition.
  - Frequently joined columns
  - Equijoin columns. An *equijoin* is a join operation in which the join condition has the form expression = expression.
  - Collocation of rows being joined will occur (avoiding movement) if the partitioning key is included in the WHERE clause.
  - Collocate the largest dimension-table's key as the partition key for the fact table, considering the number of distinct values and skew within the corresponding fact-table column
  - Replicate small dimensions, depending on the storage available

Table Partitioning

- Allows a single logical table to be broken up into multiple separate physical storage objects (up to 32K range partitions)
  - Each storage object corresponds to a ‘partition’ of the table
  - Ranges of value are used to specify each partition
  - A partition will only contain rows that match its range of values

Non-partitioned table



Payments

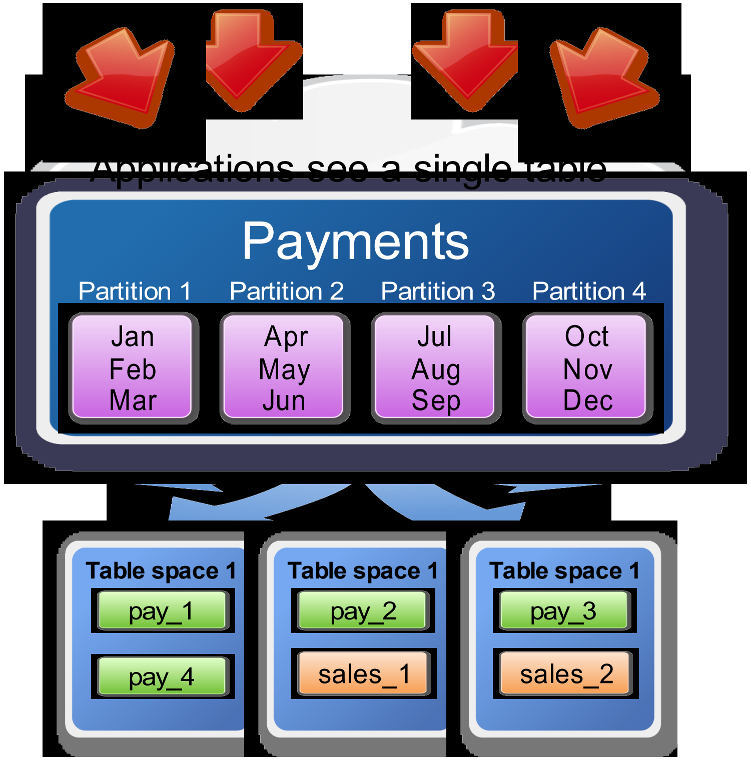
<< very large table >>

Table space

payments

sales

Partitioned table



Applications see a single table

Payments

Partition 1	Partition 2	Partition 3	Partition 4
Jan Feb Mar	Apr May Jun	Jul Aug Sep	Oct Nov Dec

Table space 1	Table space 1	Table space 1
pay_1	pay_2	pay_3
pay_4	sales_1	sales_2

Creating a Range Partitioned Table

Overview

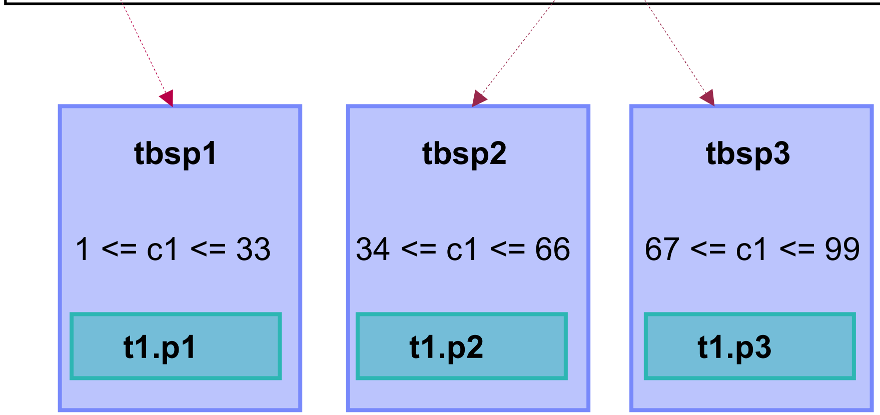
Short Form

```
CREATE TABLE t1(c1 INT) IN tbsp1, tbsp2, tbsp3
   PARTITION BY RANGE (c1)
   (STARTING (1) ENDING (99) EVERY (33))
```

- or –

Long Form

```
CREATE TABLE t1(c1 INT)
   PARTITION BY RANGE (c1)
   (PARTITION p1 STARTING(1) ENDING(33) IN
tbsp1,
      PART p2 ENDING(66)     IN tbsp2,
      PART p3 ENDING(99)     IN tbsp3)
```



- Partitioning column(s)
  - Must be base types (e.g. No LOBS, LONG VARCHARS)
  - Can specify multiple columns
  - Can specify generated columns
  - Can specify tablespace using IN clause
- SQL0327N : The row cannot be inserted because it is outside the bounds
- Special values, MINVALUE, MAXVALUE can be used to specify open ended ranges.

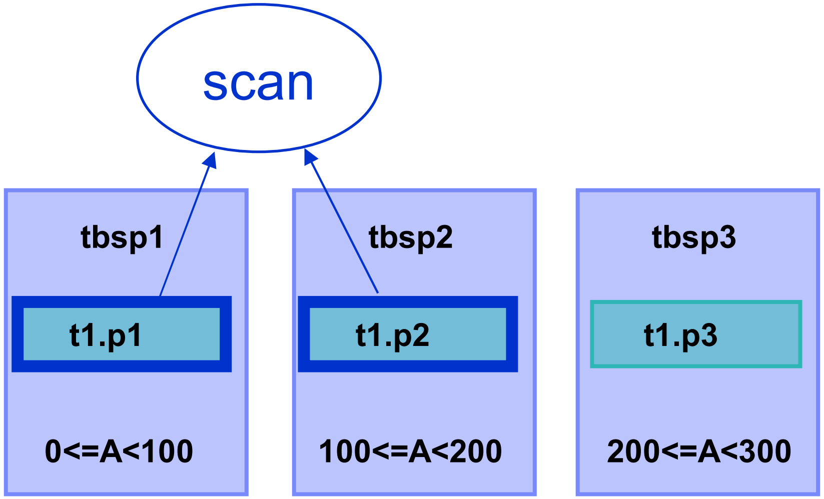
```
CREATE TABLE t1 …
( PARTITION p1 STARTING(MINVALUE)
   ENDING(MAXVALUE))
```

Data Partition Elimination

Table Scans

- Ability to determine that only a subset of the data partitions in a table are necessary to answer a query.

```
SELECT * FROM t1
WHERE A>50 AND A<150
```



- Will only access data in tbsp1 and tbsp2

Operations for Roll-Out and Roll-In

- Roll-Out: ALTER TABLE … DETACH
  - An existing range is split off as a stand alone table
  - Data instantly becomes invisible
  - Minimal interruption to other queries accessing table
- Roll-In: ALTER TABLE … ATTACH
  - Incorporates an existing table as a new range
  - Follow with SET INTEGRITY to validate data and maintain indexes
  - Data becomes visible all at once after COMMIT
  - Minimal interruption to other queries accessing table
- Key points
  - No data movement
  - Nearly instantaneous
  - SET INTEGRITY is now online

Typical Roll-Out Scenario (before)

```
CREATE TABLE sales_old …
INSERT INTO sales_old (SELECT * FROM sales WHERE …);
DELETE FROM sales WHERE ….
```

- What is wrong with this?
  - Slow, error prone
  - What will queries show while DELETE is in progress?
    - Different sets of results, possibility of deadlocks
  - Also possible to use UNION ALL views

Roll-Out Scenario (with Table Partitioning)

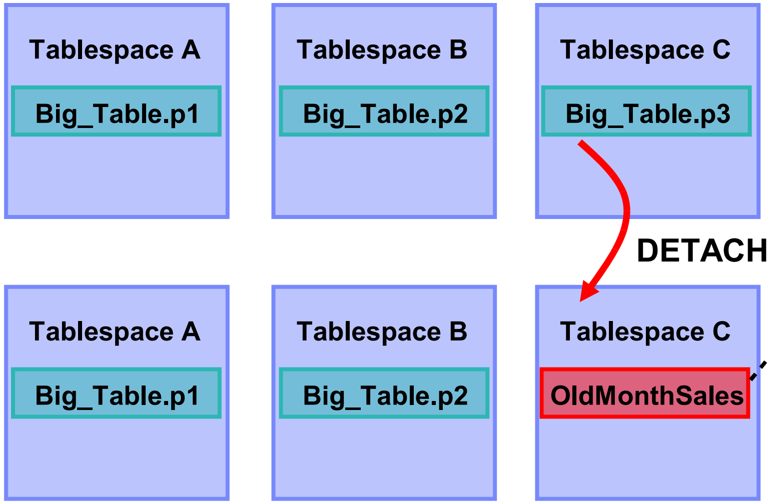
- Detached data now invisible
- Detached partition ignored in index scans
- Rest of Big_Table available
- Index maintenance is kicked off

```
ALTER TABLE Big_Table
DETACH PARTITION p3
INTO TABLE
OldMonthSales


COMMIT


SET INTEGRITY FOR
Mqt1,Mqt2 FULL ACCESS


EXPORT OldMonthSales;
DROP OldMonthSales
```

- Queries are drained and table locked
- Very fast operation
- No data movement required
- Index maintenance done later (asynchronously in background)
- Dependent MQT's go offline

- (Optional) this becomes a stand-alone table that you can do whatever you want with

- (Optional) maintains MQTs on Big_Table



- Partition 3 becomes stand-alone OldMonthSales table

Typical Roll-In Scenario (before)

- Data in a single table
  - Extract data from operational data store
  - Do data cleansing/transformation
  - Load into table
  - Use SET INTEGRITY to check RI constraints, maintain MQTs
- Using UNION ALL view
  - Extract/transform/load into a new table
  - Drop and recreate the view to incorporate new data
  - SET INTEGRITY for constraints, MQTs

Roll-In Scenario (with Table Partitioning)

```
CREATE TABLE NewMonth
LOAD/Insert into NewMonthSales


ALTER TABLE Big_Table ATTACH
PARTITION STARTING '03/01/2005
'ENDING '03/31/2005' FROM TABLE
NewMonthSales


COMMIT


SET INTEGRITY FOR Bit_Table


COMMIT
```

- Create empty staging table

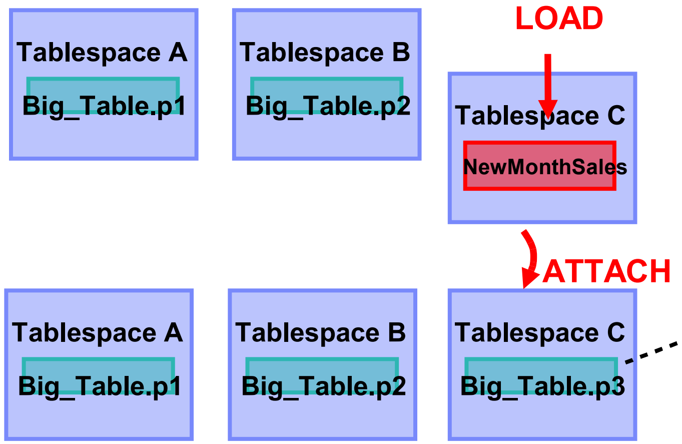
- Perform ETL on `NewMonthSales`

- Very fast operation
- No data movement required
- Index maintenance done later

- New data still not visible

- Potentially long running operation
- Validates data
- Maintains global indexes, MQTs
- Existing data available while it runs
- Used to complete the roll-in

- New data is now visible



- `NewMonthSales` becomes Partition 3 in `Big_Table`

Multidimensional Clustering Tables (MDCs)

- MDCs are a unique object in DB2 LUW that provide many advantages over other indexes
  - Particularly regular clustered indexes

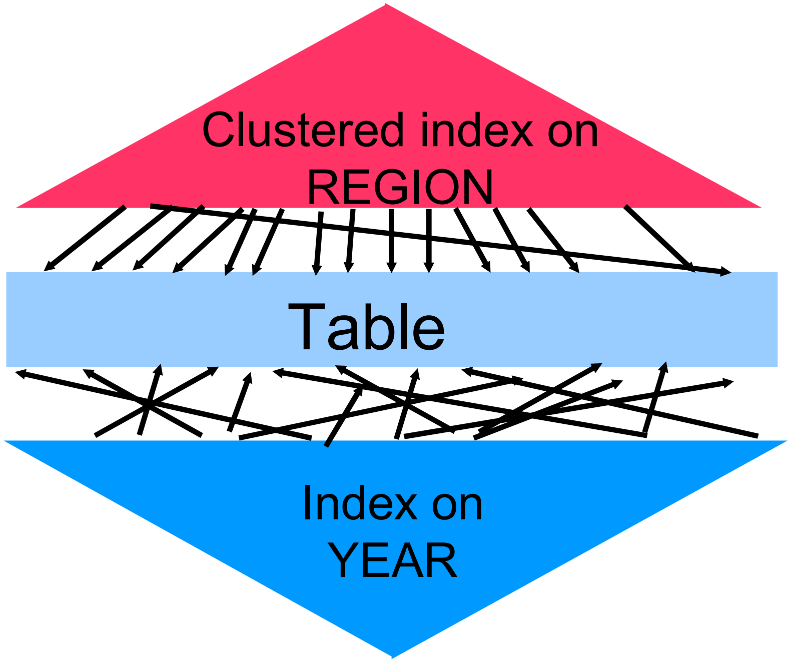
- Provide continuous, flexible and automatic clustering of data on disk

- Yield significant improvements in
  - Query performance
  - Disk space efficiency
  - Data management overhead

- “Dimensional” and great for warehousing / BI
  - Great for OLTP too

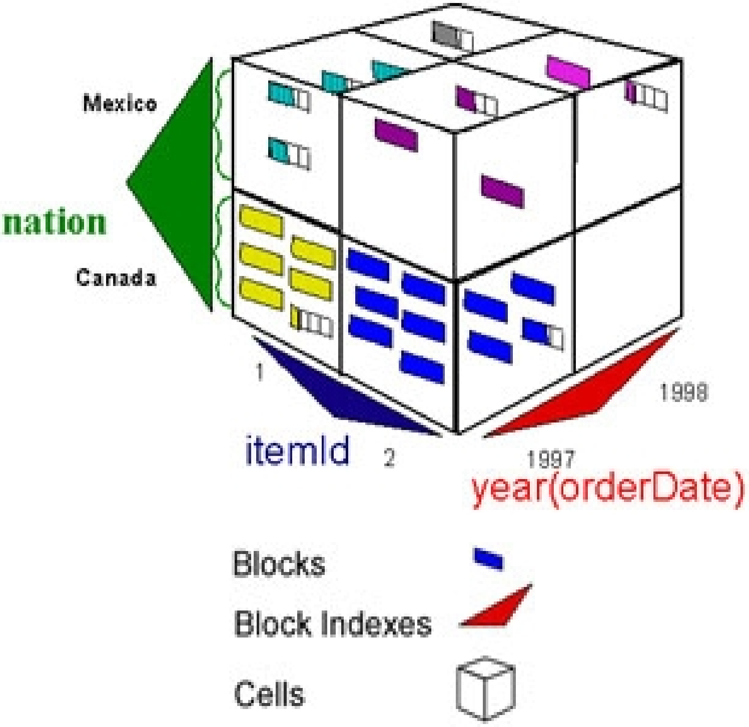
Before MDCs – Traditional Clustered Indexes

- Data physically clustered according to the cluster column
- Efficient access on one dimension, but….
- Can only cluster on one column
  - RID-based indexing on other columns doesn't benefit from ordering
- Heavy maintenance load
  - Inefficient disk clustering over time
  - Monitor and re-org to reclaim lost space
- Large RID-based index overhead
  - Excessive index space requirements



Multi-Dimensional Clustering (MDC)

- Allows for clustering of the physical data pages in multiple dimensions
- Guarantees clustering over time even if there are frequent INSERT operations performed
- Blocks
  - DB2 places records that have the same column values in physical locations that are close together
- Block Indexes
  - Indexes that point to an entire block of pages
- Cells
  - Blocks that have the same dimension values are group together



Benefits of MDCs

- Efficient I/O == **Performance**
  - 3-4X average query performance improvement, 10X+ for some queries

- **Automated** dimensional index creation & management
  - DB2 automatically creates and manages dimensional indexes

- **Never** REORG an MDC table for re-clustering
  - Only reorganize an MDC table to perform space reclamation

- Up to **64 Clustered Indexes** per table (Not just the one)

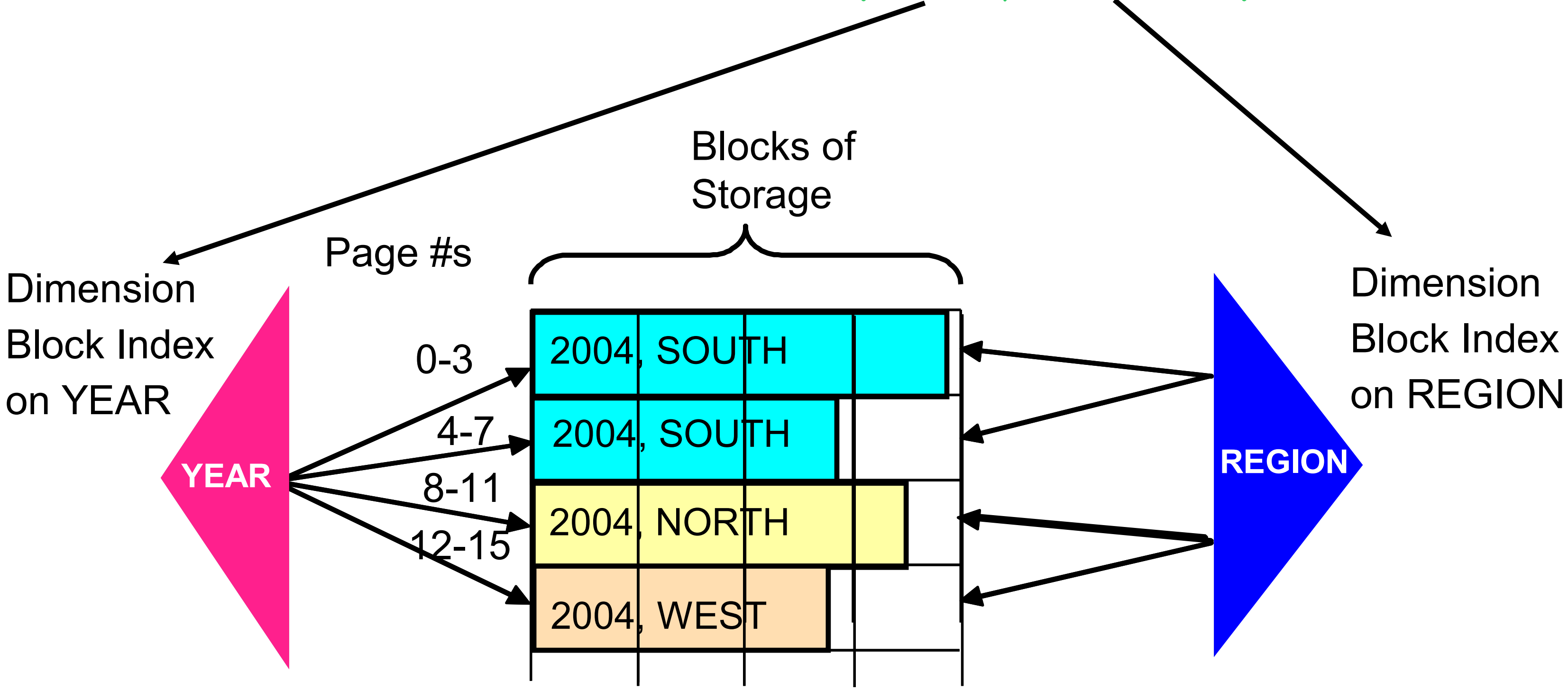
- **90+% dimension index compression** because of the on-disk nature of a MDC table and its associated block pointers
  - You can mix MDC indexes with traditional RID indexes

- **Administration-free** rolling ranges
  - No manual ATTACH or DETACH for range cycling: just load the data and MDC automatically provides the clustering

How do MDC work?

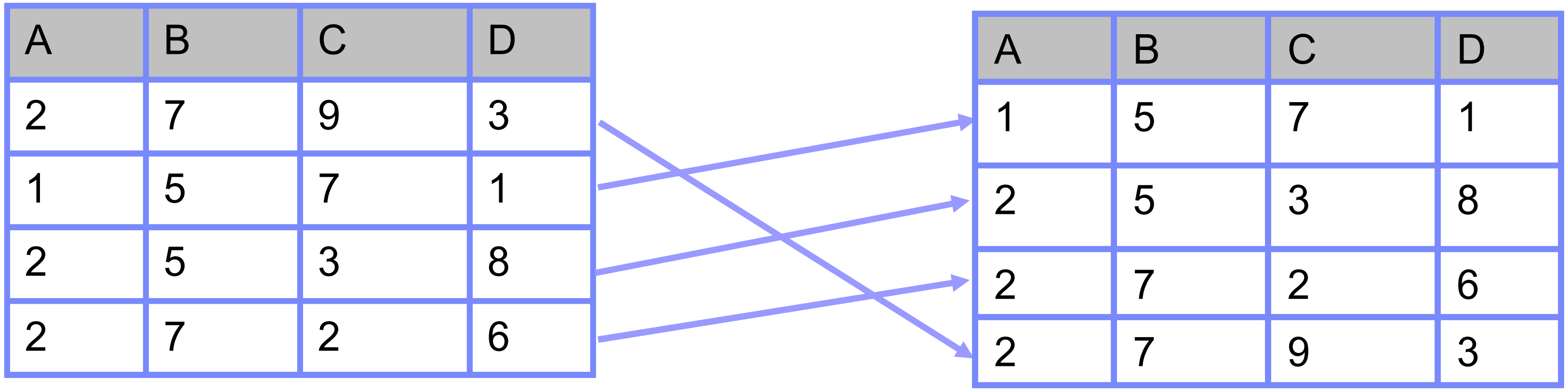
Rows clustered by Dimension values

```
CREATE TABLE MDCTABLE (
       YEAR INT,
       REGION CHAR (8),
       SALES INT,
...)
ORGANIZE BY (YEAR, REGION)
```



Multidimensional Clustering with Table Partitioning and DPF

```
CREATE TABLE my_hybrid
    (A INT, B INT, C Date, D INT …)
    IN Tablespace A, Tablespace B, Tablespace C …
    INDEX IN Tablespace B
    DISTRIBUTE BY HASH (A)
    PARTITION BY RANGE (B) (STARTING FROM (100) ENDING (300) EVERY (100))
    ORGANIZE BY DIMENSIONS (A,B,C)
```



Data blocks without MDC

A	B	C	D
2	7	9	3
1	5	7	1
2	5	3	8
2	7	2	6

Data blocks with MDC

A	B	C	D
1	5	7	1
2	5	3	8
2	7	2	6
2	7	9	3

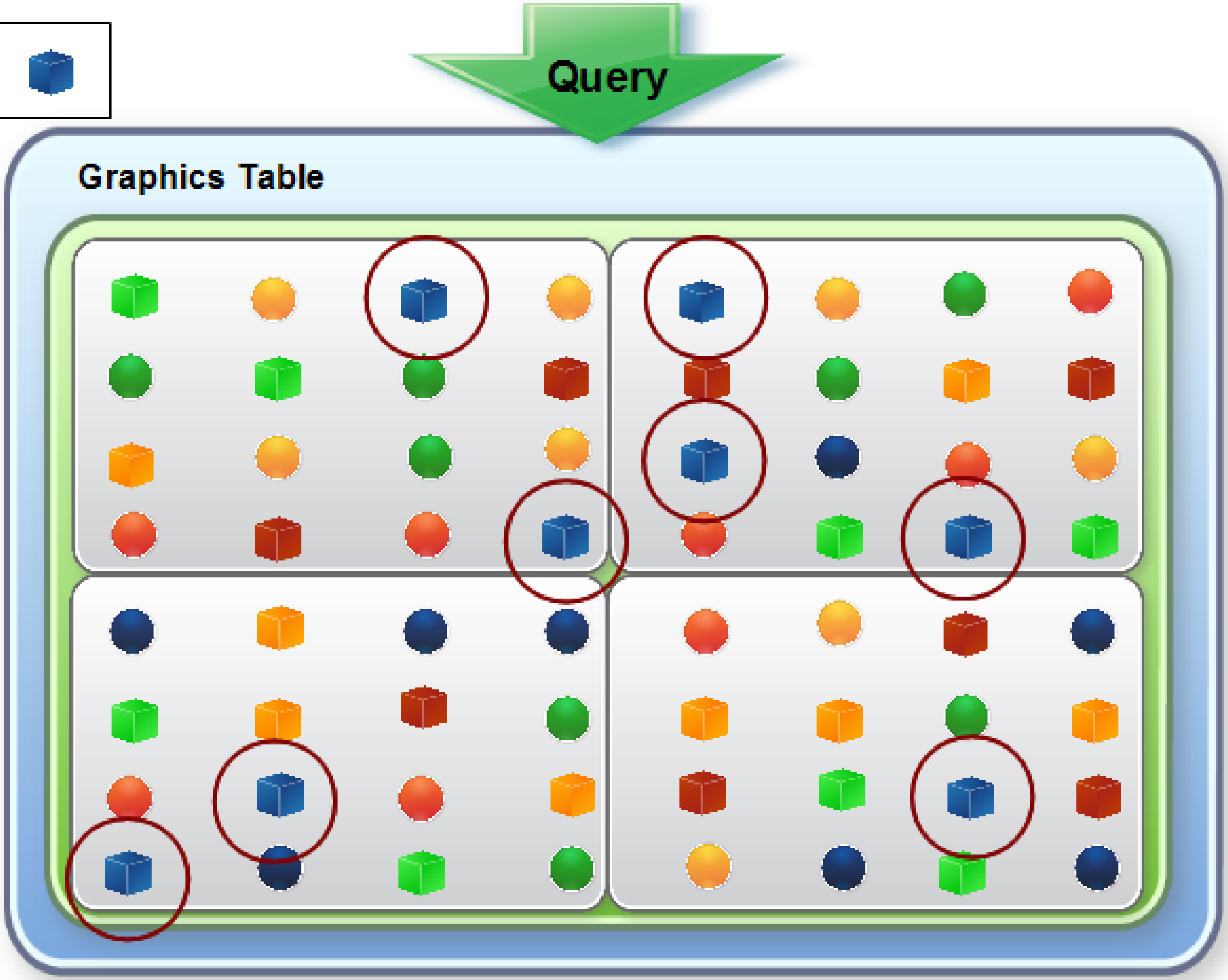
How Does DB2 Technology Help BI?

- DB2 has proven technology to break the I/O barrier
- Parallelize I/O with Database Partitioning Feature (DPF)
- Reduce I/O with Range Partitioning
- Compact I/O with Multidimensional Clustering Tables (MDC)
- The following will illustrate…..

CASE OF STUDY: Single large table without partitioning

```
SELECT * FROM GRAPHICS where 🟦
```

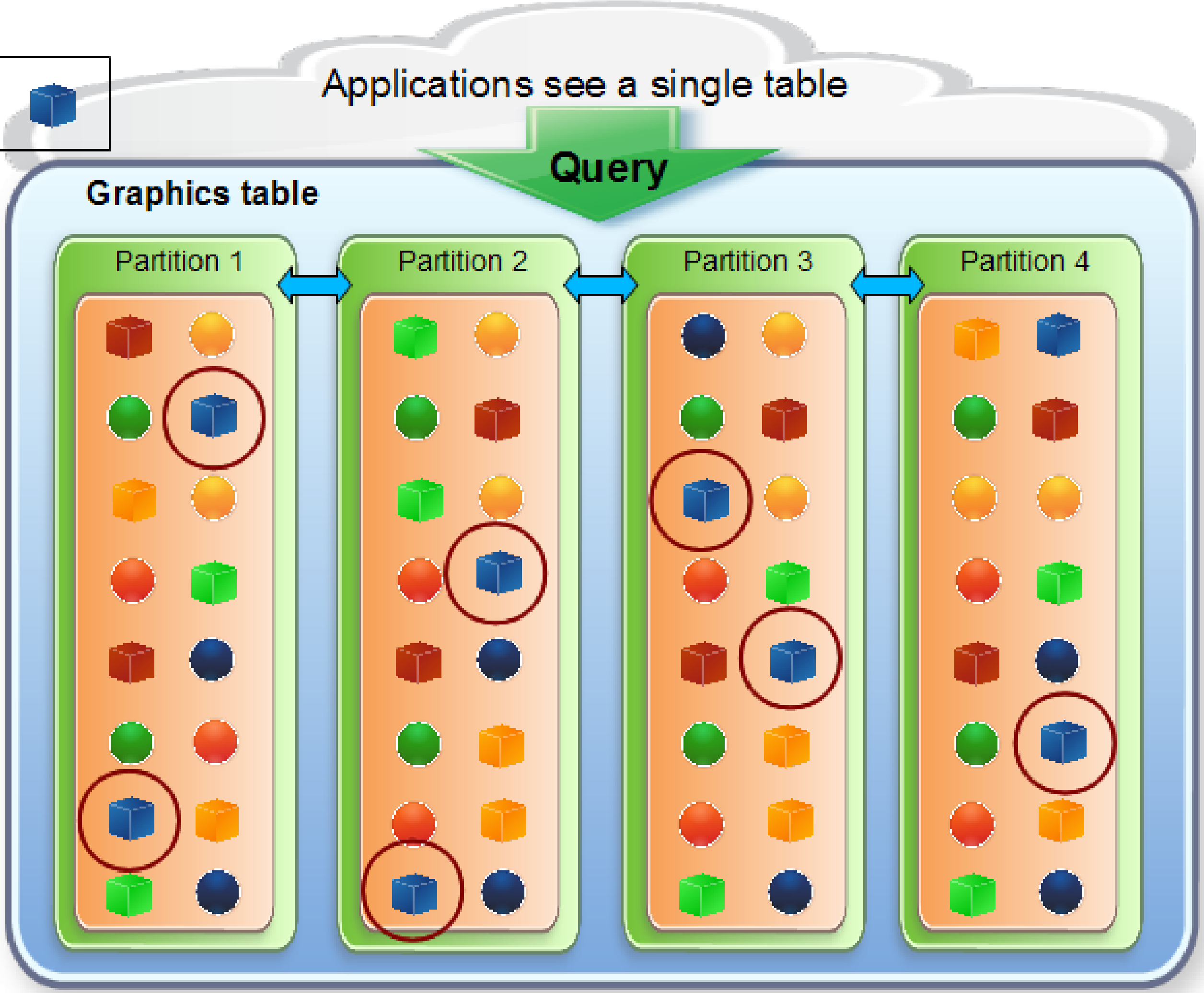
- Each I/O can pick up many unrelated records that happen to reside on the same page.
- Only one CPU is utilized for much of the processing done by a query



CASE OF STUDY: Using Database Partitioning

```
SELECT * FROM GRAPHICS where 🟦
```

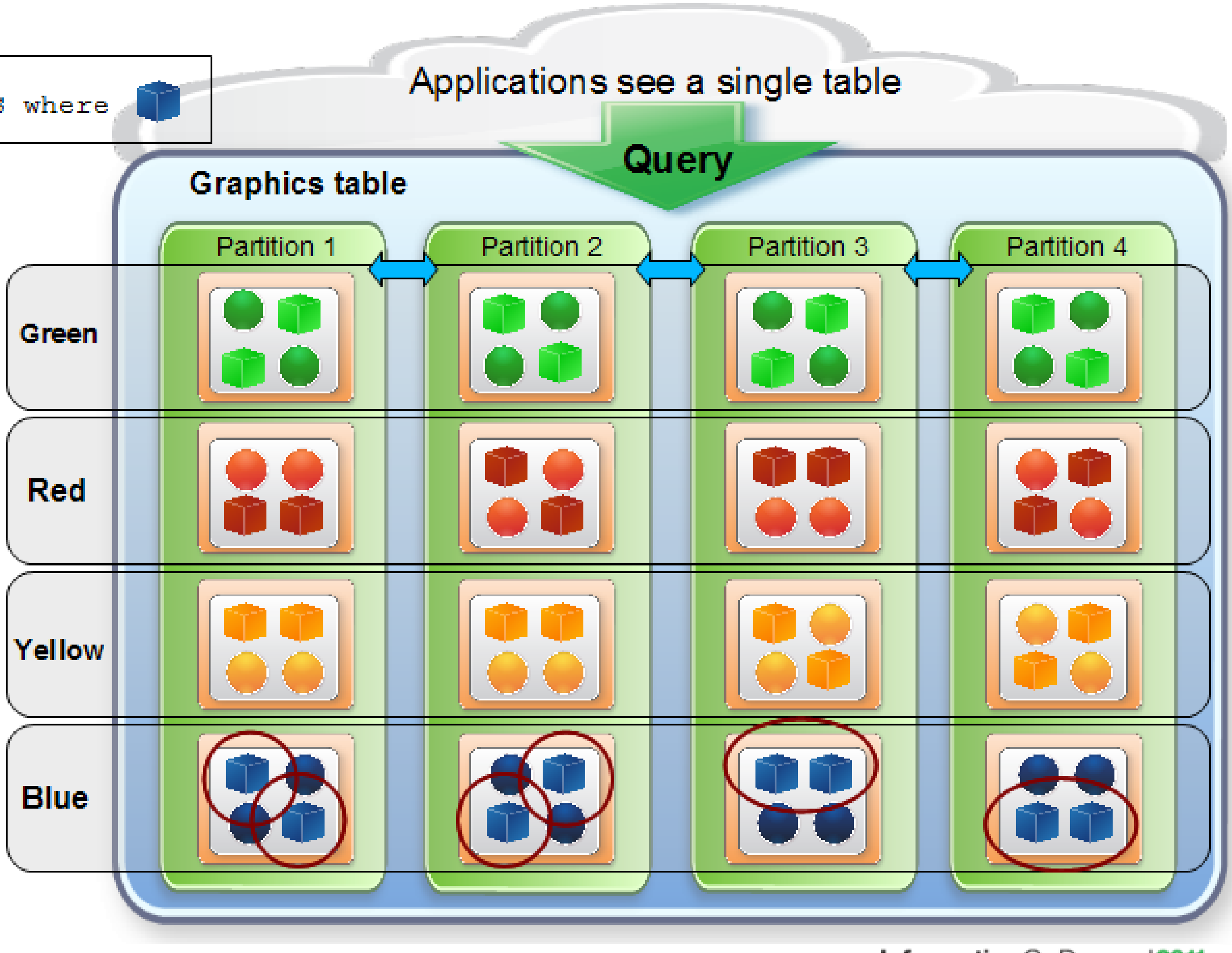
- **Take the advantage of the query parallelism provided by DB2 DPF attacking each partition in parallel.**



CASE OF STUDY: Using Database Partitioning and Table Partitioning

```
SELECT * FROM GRAPHICS where 🟦
```

- All data in the same user-defined range is consolidated in the same data partition. The database can read just the appropriate partition.
- Savings in I/O operations

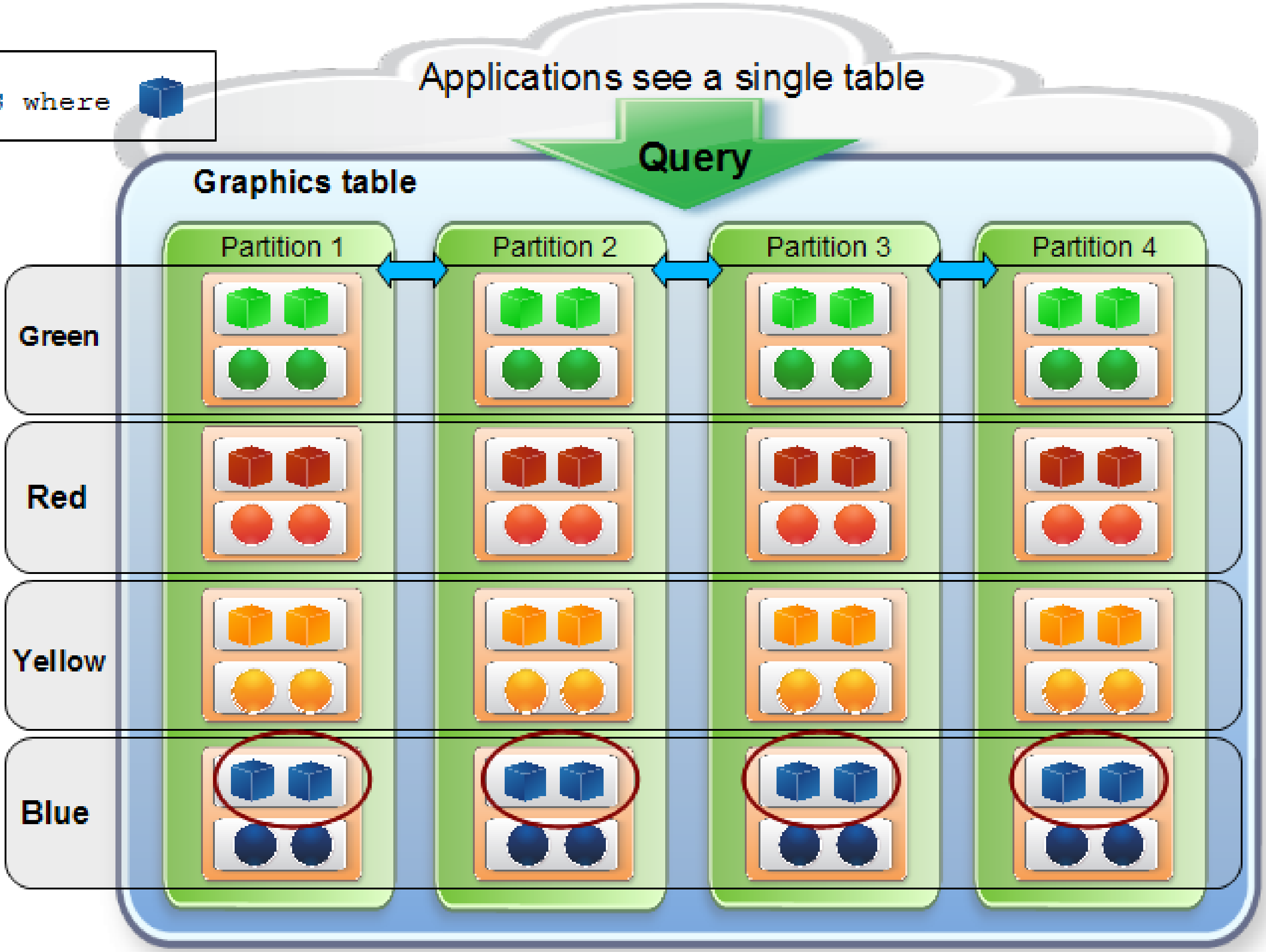


CASE OF STUDY: Using database partitioning, table partitioning, and MDC

Combine them all

```
SELECT * FROM GRAPHICS where 🟦
```

- Even less I/O is performed to retrieve the records of interest.
- The database can read just the appropriate range in the appropriate dimension



Thank You

Questions

http://www.ibm.com/software/data/infosphere/warehouse/

Acknowledgements and Disclaimers:

Availability. References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS-IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

IBM, the IBM logo, ibm.com,and DB2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml

Other company, product, or service names may be trademarks or service marks of others.

Communities

- **On-line communities, User Groups, Technical Forums, Blogs, Social networks, and more**
  - Find the community that interests you…
    - **Information Management** **ibm.com**/software/data/community
    - **Business Analytics** **ibm.com**/software/analytics/community
    - **Enterprise Content Management** **ibm.com**/software/data/content-management/usernet.html

- **IBM Champions**
  - Recognizing individuals who have made the most outstanding contributions to Information Management, Business Analytics, and Enterprise Content Management communities
    - **ibm.com**/champion

Thank You!
Your Feedback is Important to Us

- Access your personal session survey list and complete via SmartSite
  - Your smart phone or web browser at: iodsmartsite.com
  - Any SmartSite kiosk onsite
  - Each completed session survey increases your chance to win an Apple iPod Touch with daily drawing sponsored by Alliance Tech