IDUG
Leading the DB2 User Community since 1988

IDUG DB2 North America Tech Conference
Philadelphia, Pennsylvania | May 2015

#IDUGDB2

C02 – DB2 DPF: What a single partitioned DBA needs to know in 10 points

Welcome to the presentation.
Thank you for taking your time for being here.

In this presentation, my goal is to share with you 10 practical points that a single partitioned DBA needs to know to get head start on administering and managing a DPF database.

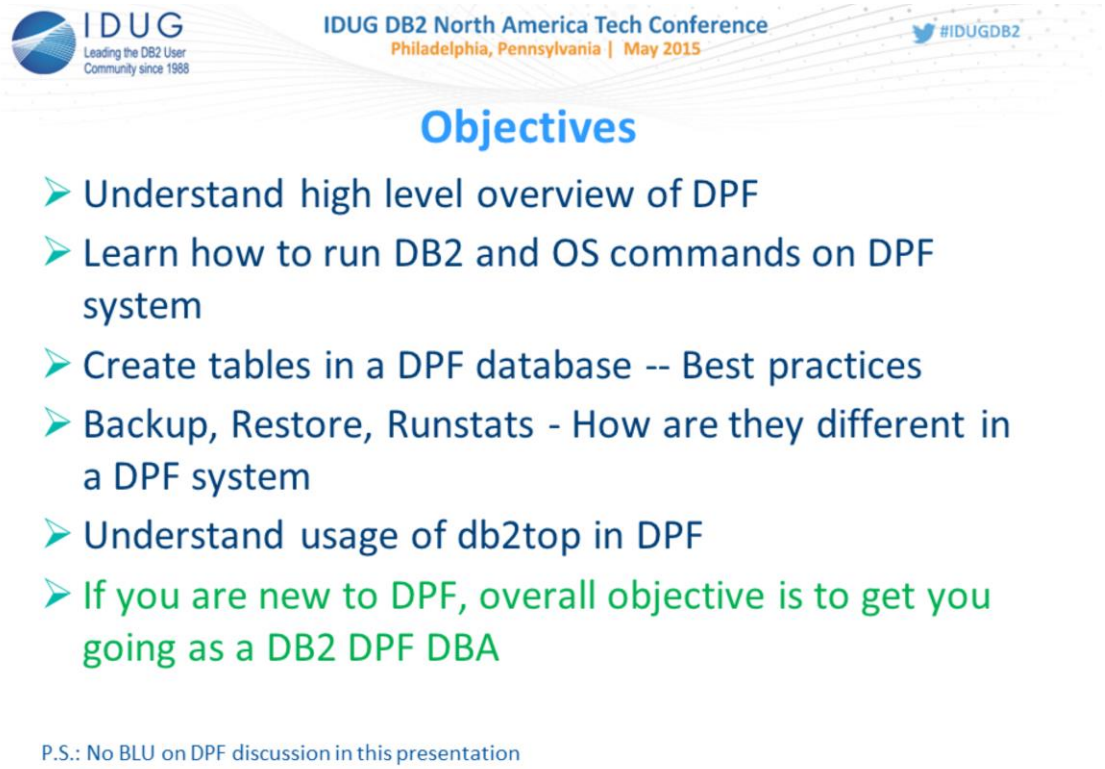
Even if you are a seasoned DB2 DPF DBA, my hope is that you would get something out of this presentation.
We will have time at the end for questions.

For the DB2Night Show today, the timing is perfect to learn about DPF or MPP because DB2 11.1 has just been released which has support for BLU on MPP. That means columnar technology is now available for DPF databases as well. This is going to be huge one and a game changer for many shops that run DPF because of multi-fold performance gains that BLU can offer and

of course we shouldn’t forget one of the most major selling points of BLU which is compression.

I am eager to see how BLU plays out in MPP myself.

- DB2 LUW 11.1 released June 2016!
- Supports BLU column organized tables on DPF (MPP)
- Compression !
  - Adaptive compression in DB2 10.5 ~70% gains
  - Imagine what BLU on DPF can do in 11.1!
- Let me show you how to become a DPF DBA



Objectives of the presentation that were included when abstract was submitted to IDUG for consideration.

High level Agenda for this presentation.
Agenda could be divided into 2 parts.
In Part 1, DPF will be introduced
In Part 2, we will discuss 10 points that a single partitioned DBA needs to know about DPF.

Pavan Kristipati

- IBM Champion 2015, 2016
- IDUG North America Conference Planning Committee
- DB2 LUW DBA since 2005
- Presented at IDUG in 2014 and 2015
- IBM Certified Advanced Database Administrator
- Technical blogging
  - www.db2talk.com – Owner
  - www.db2commerce.com – Occasional guest blogger
- https://www.linkedin.com/in/pavankristipati
- @pkristipati @db2talk

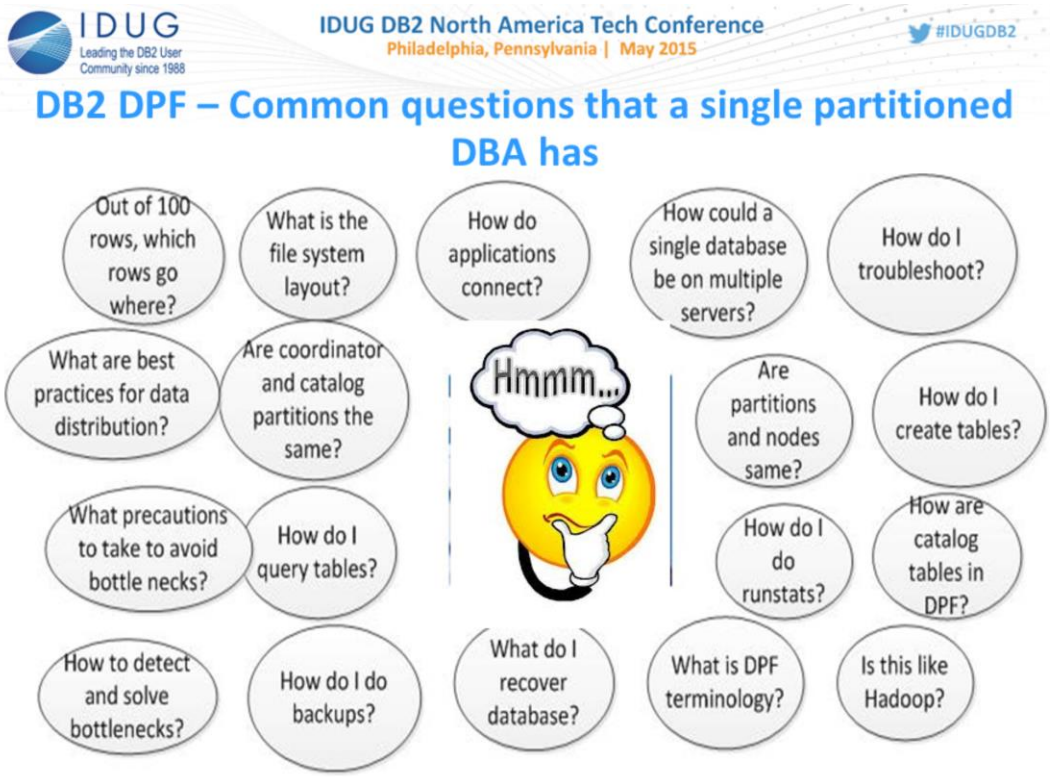
Setting the stage

- Presentation is mostly on DB2 9.7, 10.1 on UNIX (AIX / Linux)
- Assumption – DPF is installed; No discussion on installation
- Focus is on T-Shaped skills (**breadth** vs. depth)
- Best practices from experience gained managing DPF databases (I continue to work as single partitioned DBA ☺)
- Performance topics have been left out – Out of scope
- Additional reading resource on topics presented -- Pavan Kristipati's blog www.db2talk.com


db2talk
DB2 Linux, Unix and Windows Administration and Development
Home | About: Pavan Kristipati | Disclaimer
APRIL 28, 2015
Why you should specify DISTRIBUTION KEY for tables in a single database partition in a DB2 DPF database?!?
In this blog post, I will make a case for why you should specify a DISTRIBUTION KEY clause for a table that is about to be created on a single partition in a DPF database. This doesn't make sense.. Does it? For me, this idea did not make any sense at all.. However, when I started exploring more on this topic, it made some sense and when I did more research, I realized that there is a big benefit in taking this approach. Hence the blog post 🤩 Please read on ...

DB2 DPF – What, Why, Where, How?

- **What** -- DPF – Database Partitioning Feature (MPP = Massively Parallel Processing)
- **Why** -- Solution for a database that is too large on a single server
- **Where** -- Mostly Data Warehouse systems, Decision Support Systems
- **How** – Workload spread across multiple systems
- **How** -- Applications need not be aware of any parallelism; DPF presence is largely transparent to end user
- Shared nothing (a.k.a. dedicated system resources) architecture – **Each piece of database has its own data, logs, system resources etc.**
- Near-linear scalability



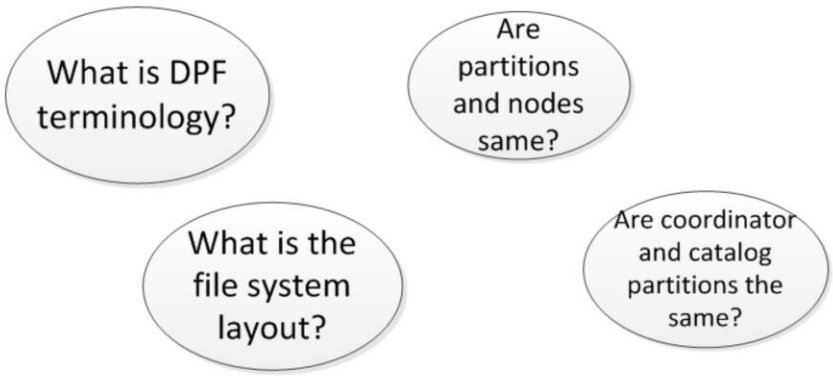
These are some of the common questions that a single partitioned DBA who is getting ready to work on DPF usually has.

While each of these questions could be a presentation by itself, we aim to address (from a high level) most of these questions in this presentation except for performance related topics.

You should have basic understanding of each of these topics by the end of this presentation.

10 POINTS

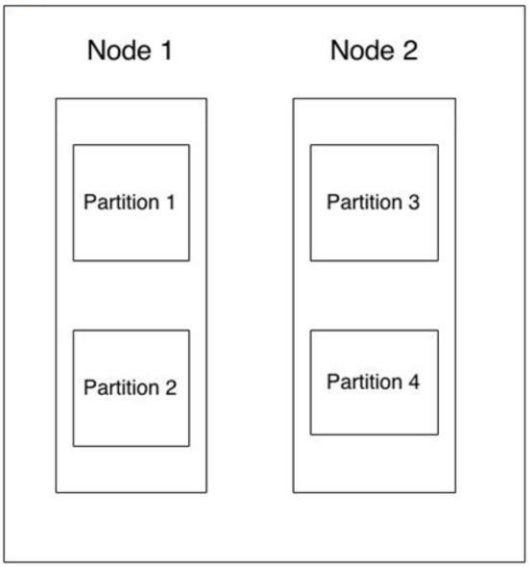
#1 OVERVIEW OF DPF



Terminology in DPF

- Node = UNIX node (Machine / Server)
- Partition = DB2 (Database Partition) → A piece of database

Example: 2 nodes and 4 partitions



A node in DPF is a UNIX server/LPAR.
A partition in DPF is a piece of database.
When there are 2 partitions on a node, those are called logical partitions

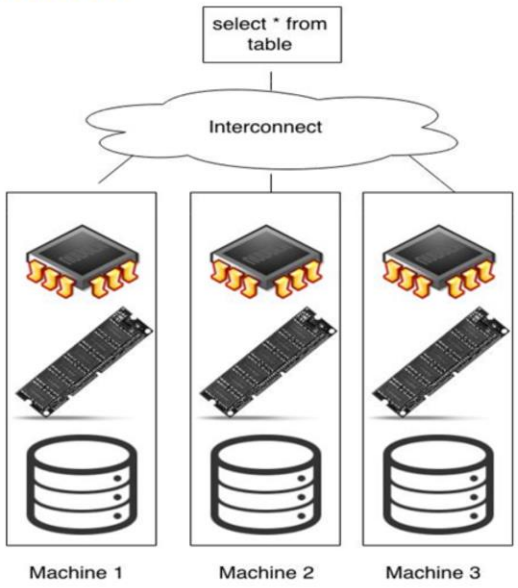
IDUG
Leading the DB2 User Community since 1988

IDUG DB2 North America Tech Conference
Philadelphia, Pennsylvania | May 2015

#IDUGDB2

Shared Nothing Architecture – The secret sauce

- Exploits parallelism
- Dedicated system resources for each node
- Each Database partition only accesses its own rows
- No lock contention among partitions
- Partitions communicate via high-speed TCP/IP network
- How is future growth handled ? Scale out – Add nodes



Each database partition has its own set of computing resources, including CPU and storage. In a DPF environment, each table row is distributed to a database partition according to the distribution key specified in the CREATE TABLE statement. When a query is processed, the request is divided so each database partition processes the rows that it is responsible for. Essentially, DPF is a scalability feature. DPF can maintain consistent query performance as the table grows by providing the capability to add more processing power in the form of additional database partitions. This capability is referred to as providing linear scalability using

DB2's shared nothing architecture.

2 ways to exploit parallelism

- Data (I/O) shipping

Vs.

- Function (SQL) 

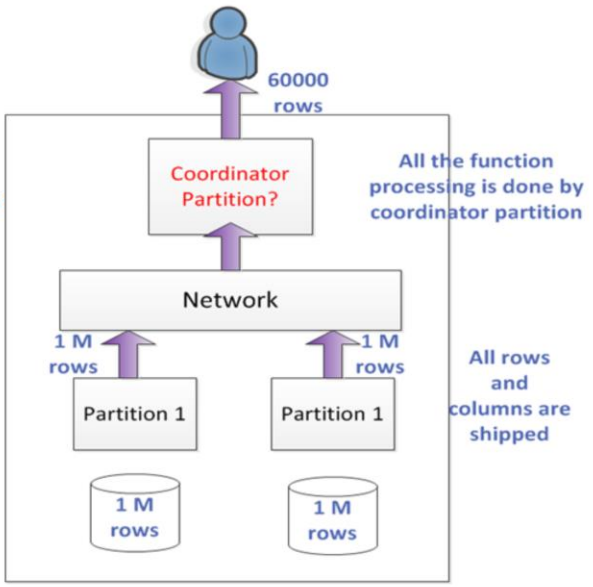
DPF relies on both these techniques but majorly uses Function shipping.

It ships SQL (function) from the coordinator node to database partitions.

Result data is shipped from data nodes to coordinator node after data processing is done on data nodes.

Data (I/O) Shipping

- Data (all rows and all columns) from each partition is sent to coordinator partition
- No SQL processing happens on individual partitions
- Coordinator node processes SQL
- Parallelism only in data-movements
- Doesn't scale

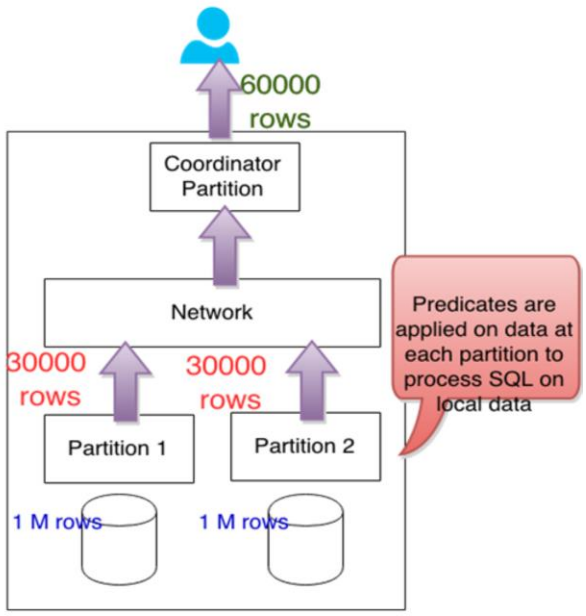


All the 1M rows from each database partition are shipped via network to the Coordinator partition which then processes the query.

If the table has 10 columns, all columns are shipped irrespective of if the SQL really needs all the columns or not.

Function Shipping a.k.a SQL Broadcasting

- SQL is sent to each partition instead of large data movements
- Process (SQL) data locally on each partition
- Reduce result set to the maximum
- Only selected columns are sent to coordinator partition
- Maximized parallelism



The select statement is broadcasted to each of the individual partitions and the predicates are applied to reduce the number of rows.

The number of columns are also reduced to the number that is required to provide the answer to the application.

The results are passed to the coordinator partition which is then passed to the end user (application).

This approach results in reduction of network traffic because the SQL function is shipped to the data instead of data being shipped to the SQL

If an ORDER BY clause was specified in the SQL, sorts would be done by each partition process and

the coordinator partition would merge the answer set.

Catalog Partition, Coordinator Partition and SSH

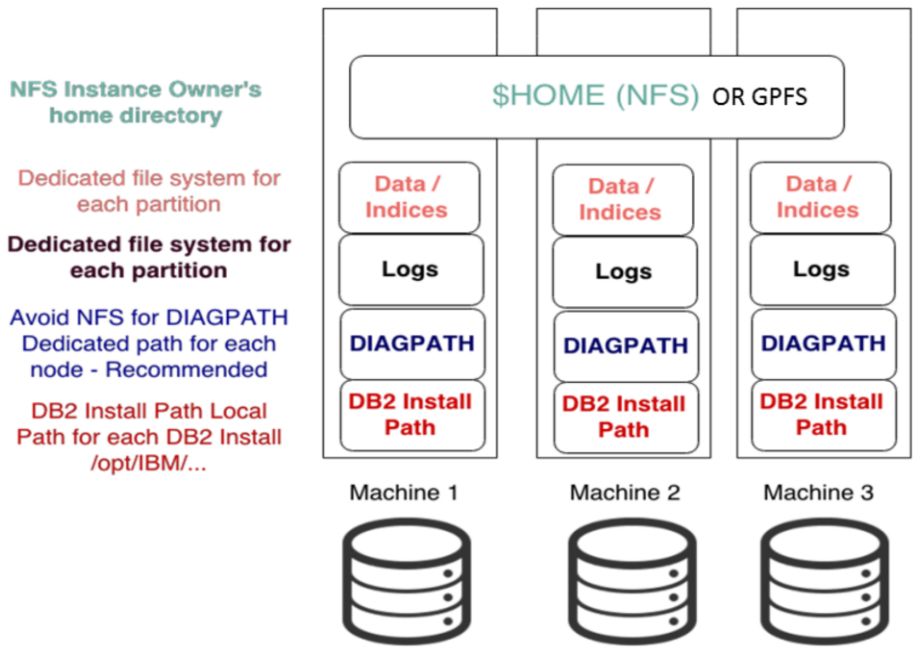
- Catalog Partition
  - Has catalog tables (Create database command was run on this partition)
- Coordinator partition
  - The one that the user connected to (could be any partition)
  - Puts together result set and presents to the user

==========

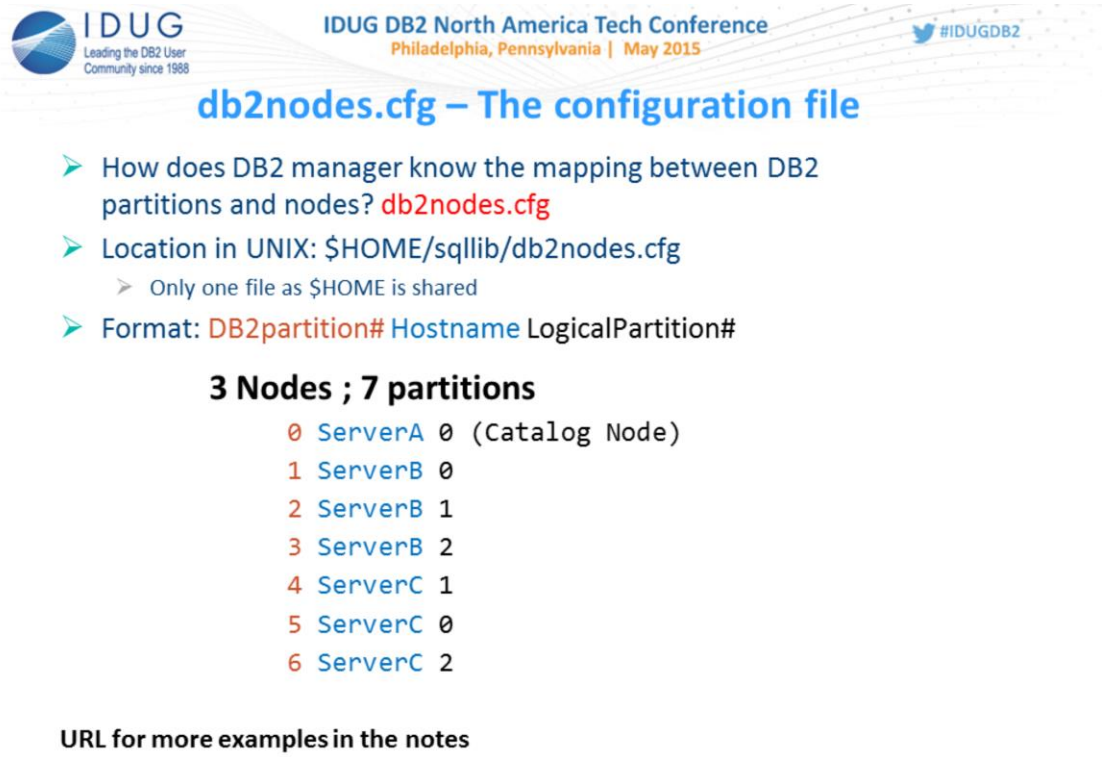
- Requirement for DPF -- Password-less 'ssh' between nodes (hosts)

```
db2inst1@ServerA:~> ssh ServerB
Last login: Mon Dec 15 16:35:47 2014 from ServerA

db2inst1@ServerB:~> ssh ServerC
Last login: Mon Dec 15 16:36:44 2014 from ServerB

db2inst1@ServerC:~> ssh ServerA
Last login: Mon Dec 15 16:37:45 2014 from ServerC
```

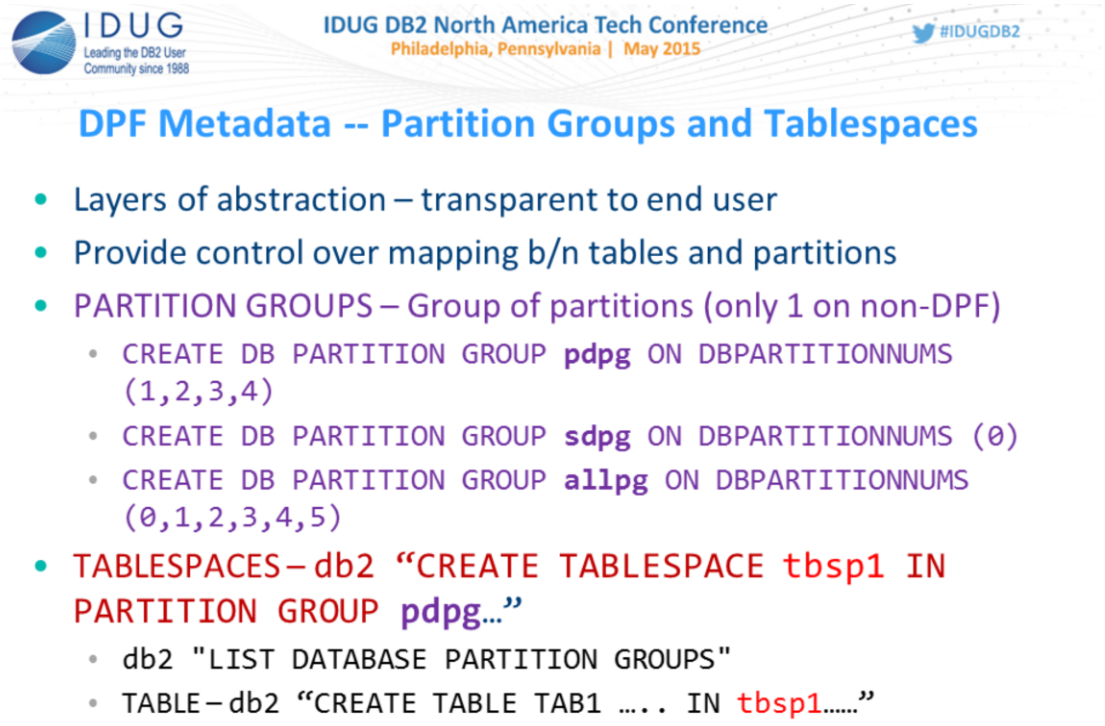
DPF – Recommended File System Layout



- Instance owner's home directory – NFS -- shared across nodes
- Local mount points for transaction and archive logs
- DIAGPATH – Diagnostic Path
  - Avoid NFS or shared mount point
  - Best Practice = Local mount point
- In a large DPF system, an NFS DIAGPATH could be IO bound – Heavy write activity from multiple hosts at the same time



Refer to http://www-01.ibm.com/support/knowledgecenter/SSEPGG_9.1.0/com.ibm.db2.udb.uprun.doc/doc/r0006351.htm for more details on these examples.



Syntax to create a partition group:

http://www-01.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.sql.ref.doc/doc/r0000921.html?cp=SSEPGG_10.5.0%2F2-12-7-62

Syntax to create Tablespace:

http://www-01.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.sql.ref.doc/doc/r0000929.html?cp=SSEPGG_10.5.0%2F2-12-7-102

#2: RUNNING COMMANDS IN DPF

How do I interact with nodes and partitions?

How is it different from running commands on single partitioned DB2 database?

Two options
db2_all vs. rah

db2_all (Run on all db2 partitions)	Rah (Run on All Hosts)
Runs command on each database partition	Runs command on each node (host)
Prefix 'db2_all' command to db2 command	Prefix 'rah' to OS command

db2_all – returns output from each db partition

1 catalog node + 2 hosts with 2 partitions on each = total 5 partitions

```
$db2_all "db2 get db cfg for edwdv | grep LOGARCHMETH1"
 First log archive method                    (LOGARCHMETH1) = TSM
dvadm01: db2 get db cfg for ... completed ok
 First log archive method                    (LOGARCHMETH1) = TSM
dvdata01: db2 get db cfg for ... completed ok
 First log archive method                    (LOGARCHMETH1) = TSM
dvdata01: db2 get db cfg for ... completed ok
 First log archive method                    (LOGARCHMETH1) = TSM
dvdata02: db2 get db cfg for ... completed ok
 First log archive method                    (LOGARCHMETH1) = TSM
dvdata02: db2 get db cfg for ... completed ok
```

Rah (Run on All Hosts) – returns output from each UNIX host (node)

Create a file alert.log on each Server in DPF cluster

```
$rah 'touch /tmp/alert.log'
dvadm01:  touch /tmp/alert.log completed ok
dvdata01: touch /tmp/alert.log completed ok
dvdata02: touch /tmp/alert.log completed ok
dvdata03: touch /tmp/alert.log completed ok
```

4 Nodes/Hosts

4 lines of output

```
$rah 'ls -ltr /tmp/alert.log'
-rw-r--r-- 1 db2inst1 bcuigrp 0 2015-01-14 16:30 /tmp/alert.log
dvadm01: ls -ltr /tmp/alert.log completed ok
-rw-r--r-- 1 db2inst1 bcuigrp 0 2015-01-14 16:30 /tmp/alert.log
dvdata01: ls -ltr /tmp/alert.log completed ok
-rw-r--r-- 1 db2inst1 bcuigrp 0 2015-01-14 16:30 /tmp/alert.log
dvdata02: ls -ltr /tmp/alert.log completed ok
-rw-r--r-- 1 db2inst1 bcuigrp 0 2015-01-14 16:30 /tmp/alert.log
dvdata03: ls -ltr /tmp/alert.log completed ok
```

Check for db2sysc processes in DPF using 'rah'

```
$rah "ps -ef | grep -i db2sysc | grep -vi grep"
db2inst1 13799 13784 62  2014 ?        94-18:43:01 db2sysc 0
dvadm01: ps -ef | grep -i db2sysc ... completed ok

db2inst1 12159 12154 19  2014 ?        29-22:36:17 db2sysc 1
db2inst1 12165 12156 15  2014 ?        23-05:56:11 db2sysc 3
db2inst1 12171 12153 14  2014 ?        22-12:38:24 db2sysc 2
db2inst1 12175 12155 15  2014 ?        22-17:23:02 db2sysc 4
dvdata01: ps -ef | grep -i db2sysc ... completed ok

db2inst1 21081 21079 15  2014 ?        22-15:12:20 db2sysc 7
db2inst1 21157 21150 15  2014 ?        22-21:02:11 db2sysc 8
db2inst1 21158 21146 15  2014 ?        22-17:31:29 db2sysc 6
db2inst1 21194 21148 15  2014 ?        23-03:19:41 db2sysc 5
dvdata02: ps -ef | grep -i db2sysc ... completed ok

db2inst1 18972 18970 17  2014 ?        26-00:12:51 db2sysc 10
db2inst1 19037 19032 15  2014 ?        22-16:09:42 db2sysc 12
db2inst1 19042 19034 15  2014 ?        23-03:03:20 db2sysc 9
db2inst1 19871 19869 14  2014 ?        22-08:13:05 db2sysc 11
dvdata03: ps -ef | grep -i db2sysc ... completed ok
```

Number of db2sysc processes ~ No. of partitions

How do I create tables?

#3 BEST PRACTICES – CREATING TABLES IN A DPF DATABASE

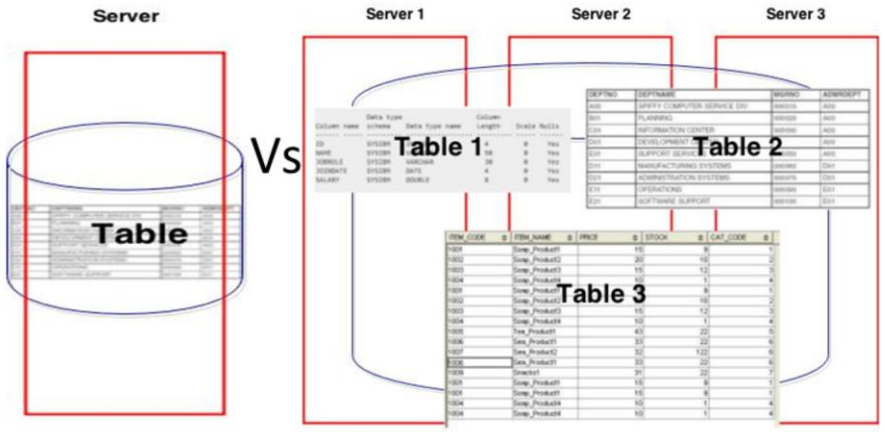
What are best practices for data distribution?

What precautions to take to avoid bottle necks?

Tables in DPF

- A single table spread across multiple servers
- How does DB2 know which partition to send a row to?

Single Partitioned Database **DPF Database**



How to create a table on more than 1 partition?

No option in 'create table' command to specify which partitions the table is to be created on

Create table in DPF in 3 steps:

- **Create a "Partition Group" (Group of partitions)**
  -- CREATE DATABASE PARTITION GROUP PDPG (Done only once)
- **Create a "Tablespace" in partition group created in step 1**
  -- CREATE LARGE TABLESPACE *TBSP1* IN DB PARTITION GROUP *PDPG*
- **Create a table in the tablespace created in step 2**
  -- CREATE TABLE EDWSDV.EMPLOYEE .... **DISTRIBUTE BY HASH (EMPLOYEE_ID)** IN TBSP1 INDEX IN TBSP1_IX;

EMPLOYEE_ID is called Distribution Key

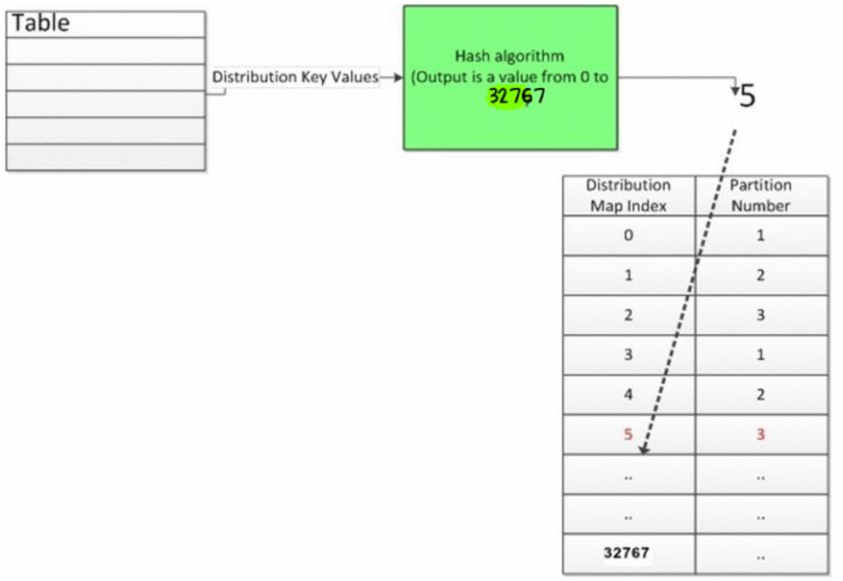
What is a Distribution Key?

- A distribution key is a column (or set of columns) that dictate the database partition on which DB2 will physically store a row in a table. Also called "Hash Key"
- Distribution key applies to each (hash partitioned) table and not to the entire database.

Example:

- Table1 → DISRIBUTION KEY (col1, col2)
- Table2 → DISRIBUTION KEY (col3)
- For a table, a distribution key is defined by using the CREATE TABLE statement with the DISTRIBUTE BY clause.

Distribution Map



When a new row is inserted into a hash partitioned table, DB2 applies the hashing algorithm on the values of distribution keys for that row. The output of the hashing algorithm is a number from 0 to 32767 (was 4095 until DB2 9.7). This number corresponds to one of the entries in the array that contains the value of the database partition number where the row is to be stored.

In the example below, as the hashing algorithm returned an output value of 5, the row would be stored on partition #3.

Here is a question you should be asking yourselves

- Is every table in DPF spread across multiple nodes?
- How to know if I should spread the table across multiple nodes or not?
- Does it matter which column I pick as the distribution key?

To Hash or Not – Best Practices

- **Goal – To exploit parallelism (divide and conquer)**
- Ask (Data Modeler) for expected cardinality and usage pattern
- Hash table with > 100k rows (My practice)
- Not a hard-and-fast rule; DBA's discretion
- Hash large tables to divide work among database partitions

Exploiting Parallelism by distributing rows across partitions

```
SELECT COUNT (*) AS ROW_COUNT,
DBPARTITIONNUM(ANY_COLUMN) AS
PARTITION_NUM
FROM SCHEMA.TABLE
GROUP BY DBPARTITIONNUM
(ANY_COLUMN)
ORDER BY 2 WITH UR;
```

```
ROW_COUNT   PARTITION_NUM
----------- ---------------
     400000               1
     400000               2
     400000               3
     400000               4
4 record(s) selected
```

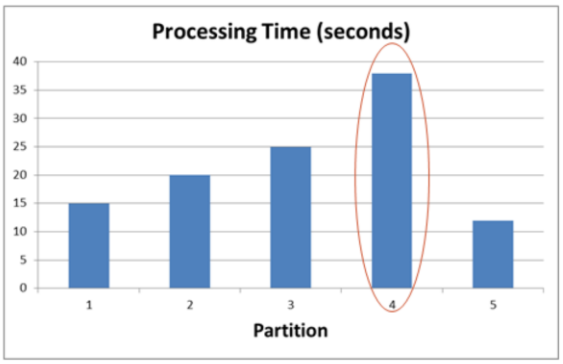
ROW_COUNT PARTITION_NUM

2000000 0

1 record(s) selected

Choosing a Distribution Key

- **Goal -- Reduce the time required to process a query**
- If the data to be processed is suitably partitioned, parallel execution is possible without excessive overhead and bottlenecks.
- Query's run time is as long as time taken by the slowest database partition.



2 Strategies
Colocation Vs. Even Data Distribution

- Co-locate frequently joined tables
- Ensure even distribution of a table's data

Let us dig into details…

Strategy #1: Co-Location

What is it?
What does it do?

- Minimizes data transfer overhead
- Ensures the matching rows between two tables to always reside on the same database partition
- A co-location join can only occur when certain criteria are met:
  - Both joining tables are in the same DB PG.
  - The distribution keys for both tables have same number of columns and compatible data types.
  - The distribution key columns are partition-compatible.

Strategy #1: Co-Location

When to use it?

- Co-locate frequently joined Fact and Dimension tables
- Can result in skew (and hence processing delays) on a subset of partitions – Fact tables are much larger than dimension tables
- Tends to be application / workload specific
- A single new query which is completely different from the existing ones could prove the distribution strategy to be non-optimal.

Strategy #2: Even distribution of data

What is it?
What does it do?

- Aims to minimize data skew by ensuring even data distribution on disks across multiple nodes
- Multiple database partitions (on which the table is spread out) work on every query
- **More parallelism** although there could be data transfer across the partitions.
- From experience -- Scales well with data growth
- Distribution Key cardinality plays a major role

Distribution Key Columns Cardinality and data skew

- Distribution key – Pick high cardinality column(s)
- Best Candidate – Primary Key
- Avoid timestamp (or type 'time') → Could result in processing skew if mostly going after a specific time period
- Hashing on:
  - Low-cardinality columns leads to data skew => processing skew (possibility)
  - High-cardinality columns minimizes data skew

Hashing on LOW cardinality column

```
$db2 "set serveroutput on"
$db2 "call estimate_existing_data_skew
('DB2INST1','TRANSACTION_HASH1',100)"
```

Estimated total number of records in the table: : 2,000,000
Estimated average number of records per partition : 166,666
Row count at partition 1 : 191,984 (Skew: 15.19%)
Row count at partition 2 : 188,357 (Skew: 13.01%)
Row count at partition 3 : 227,031 (Skew: 36.21%)
Row count at partition 4 : 182,864 (Skew: 9.71%)
Row count at partition 5 : 113,382 (Skew: 31.97%)
Row count at partition 6 : 40,560 (Skew: 75.66%)
Row count at partition 7 : 213,304 (Skew: 27.98%)
Row count at partition 8 : 201,435 (Skew: 20.86%)
Row count at partition 9 : 127,661 (Skew: 23.40%)
Row count at partition 10 : 104,527 (Skew: 37.28%)
Row count at partition 11 : 194,245 (Skew: 16.54%)
Row count at partition 12 : 214,650 (Skew: 28.79%)

Hashing on a low cardinality column leads to uneven distribution of data => possibility of bottlenecks

SET SERVEROUTPUT command

Specifies whether output from the DBMS_OUTPUT message buffer is redirected to standard output.

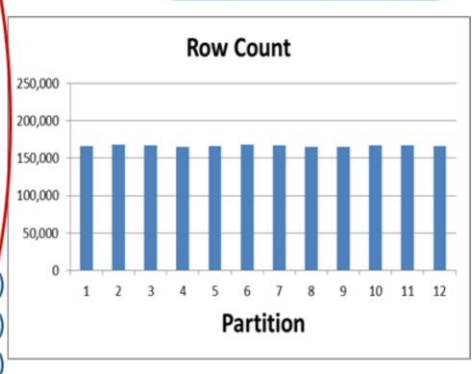
Partition # 6 has 76% more rows compared to the avg. of 166k rows.. Such a skew would be a result of distributing a key on low cardinality column.

Hashing on HIGH cardinality column

```
$db2 "set serveroutput on"
$db2 "call estimate_existing_data_skew ('DB2INST1','TRANSACTION_HASH1',100)"
```

Estimated total number of records in the table: : 2,000,000
Estimated average number of records per partition : 166,666
Row count at partition 1 : 166,075 (Skew: 0.35%)
Row count at partition 2 : 168,052 (Skew: 0.83%)
Row count at partition 3 : 167,450 (Skew: 0.47%)
Row count at partition 4 : 165,207 (Skew: 0.87%)
Row count at partition 5 : 166,456 (Skew: 0.12%)
Row count at partition 6 : 167,996 (Skew: 0.79%)
Row count at partition 7 : 167,483 (Skew: 0.49%)
Row count at partition 8 : 164,753 (Skew: 1.14%)
Row count at partition 9 : 165,649 (Skew: 0.61%)
Row count at partition 10 : 167,659 (Skew: 0.59%)
Row count at partition 11 : 167,357 (Skew: 0.41%)
Row count at partition 12 : 165,863 (Skew: 0.48%)

Note:
It is Important to do data skew analysis on a fairly large data set to avoid misleading conclusions (see notes)



The skew % would be totally different if analysis is done on a fairly small data set.

For only 120 (multiple of # of database partitions) rows and distribution key = primary key, below is skew data:

Estimated total number of records in the table: : 120
Estimated average number of records per partition : 10
Row count at partition 1 : 106 (Skew: 960.00%)
Row count at partition 2 : 0 (Skew: 100.00%)
Row count at partition 3 : 0 (Skew: 100.00%)
Row count at partition 4 : 14 (Skew: 40.00%)
Row count at partition 5 : 0 (Skew: 100.00%)
Row count at partition 6 : 0 (Skew: 100.00%)
Row count at partition 7 : 0 (Skew: 100.00%)
Row count at partition 8 : 0 (Skew: 100.00%)
Row count at partition 9 : 0 (Skew: 100.00%)

Row count at partition 10 : 0 (Skew: 100.00%)
Row count at partition 11 : 0 (Skew: 100.00%)
Row count at partition 12 : 0 (Skew: 100.00%)

#4: CATALOG DATA IN A DPF DATABASE

What is a Catalog Partition?

- Catalog Partition – CREATE DATABASE command was run
- DB2's system catalog tables exist on catalog partition ONLY
- Usually partition #0
- Resides in IBMCATGROUP partition group – Created by default
- How to find which partition is a catalog partition? Use SQL below

```
db2 "select * from SYSCAT.DBPARTITIONGROUPDEF where DBPGNAME = 'IBMCATGROUP'"

DBPGNAME     DBPARTITIONNUM     IN_USE

------------------------------------------

IBMCATGROUP            0                Y

  1 record(s) selected
```

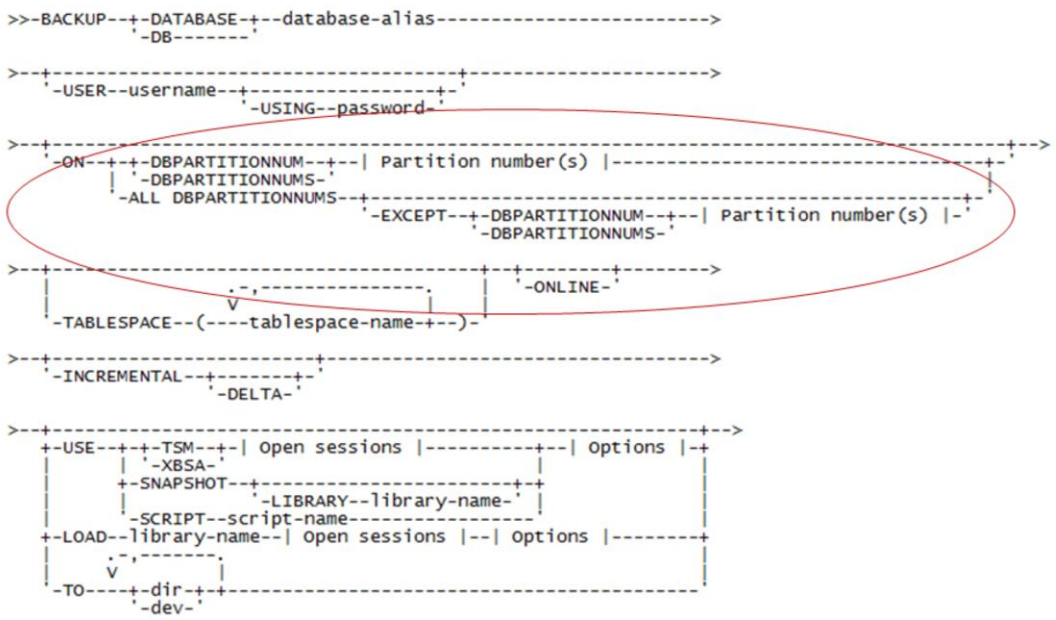
Catalog Data in a DPF database

- Catalog data could be read from non-catalog partition as well (although it physically exists on catalog partition)
- Schemas starting with SYS reside on catalog partition
- Single catalog entry for each table/view/index (even though data tables exist on multiple partitions)
  - syscat.tables / syscat.views
  - syscat.indexes
  - syscat.tablespaces
- SQLs in the bonus section

- Which database partitions is the table on?
- Which database partitions is the tablespace on?

#5: DB2 UTILITIES IN A DPF DATABASE

Utility #1: Database Backup



For a database backup command, the highlighted area is the only difference (in syntax) between DPF and non-DPF environments

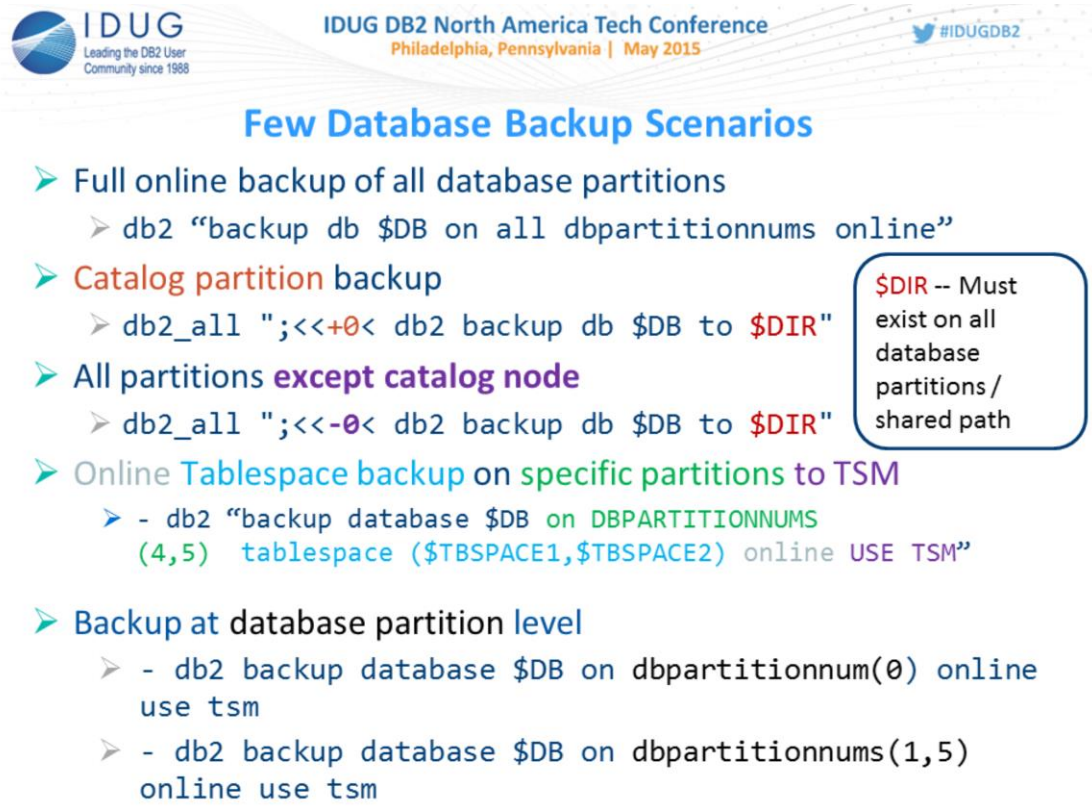
Utility #1: Database Backup

```
db2 "BACKUP DATABASE TESTDB ON ALL DBPARTITIONNUMS online"
Part  Result
----  ------------------------------------------------------------
0000  DB20000I  The BACKUP DATABASE command completed successfully.
0001  DB20000I  The BACKUP DATABASE command completed successfully.
0002  DB20000I  The BACKUP DATABASE command completed successfully.
0003  DB20000I  The BACKUP DATABASE command completed successfully.
Backup successful.The timestamp for this backup image is: 20141201131655
```

- 4 backup files generated -- single timestamp for all images
- The backup image name indicates details like database name, type of backup (full offline), Instance Name, partition being backed, catalog number, timestamp and sequence number

TESTDB.0.db2inst1.NODE0000.CATN0000.20141201131655.001
TESTDB.0.db2inst1.NODE0001.CATN0000.20141201131655.001
TESTDB.0.db2inst1.NODE0002.CATN0000.20141201131655.001
TESTDB.0.db2inst1.NODE0003.CATN0000.20141201131655.001

SSV (Single System View) backup generates multiple (as many as # of partitions) backup image files, all with the same timestamp.



Best Practices for "Building a recovery strategy for an IBM Smart Analytics System data warehouse". Refer to the link below:

https://www.ibm.com/developerworks/community/wikis/form/anonymous/api/wiki/0fc2f498-7b3e-4285-8881-2b6c0490ceb9/page/d78c9663-638f-4aeb-9135-ee54f23ec0b8/attachment/c678ff88-de64-455b-b89f-d91f82b5c05c/media/IBM-SAS_Recovery-BP.pdf

Utility #2: Database Restore

- Earlier approach was to restore catalog partition followed by non-catalog partitions – **No need to do this any more**
- **No difference in syntax** between single partitioned and DPF

Best Practices for “Building a recovery strategy for an IBM Smart Analytics System data warehouse”. Refer to the link below:

https://www.ibm.com/developerworks/community/wikis/form/anonymous/api/wiki/0fc2f498-7b3e-4285-8881-2b6c0490ceb9/page/d78c9663-638f-4aeb-9135-ee54f23ec0b8/attachment/c678ff88-de64-455b-b89f-d91f82b5c05c/media/IBM-SAS_Recovery-BP.pdf

Few Restore / Recover Scenarios

- Two approaches – Serial / Parallel restore
  - **db2_all** "db2 RESTORE DATABASE EDWDBPR USE TSM TAKEN AT 20140318143005 WITHOUT PROMPTING"
  - **db2_all** "|| db2 RESTORE DATABASE EDWDBPR USE TSM TAKEN AT 20140318143005 WITHOUT PROMPTING“ – Notice space between “ and db2.

- db2_all "<<**+3**<db2 \"restore database bcudb1 tablespace (pdts_in, pdts_ini) online use tsm taken at 20140501143616 replace existing \" “ → **Restore tablespaces on partition #3**

- db2 “**recover** database $DB to end of logs on dbpartitionnum (0)” – **Recover catalog partition**

Tablespace level restore scenarios

- **Commands could be run from any node**
- **Table spaces on non-catalog partitions**
  - *db2_all "||<<-0< db2 RESTORE DATABASE TESTDB TABLESPACE \(TBSP3, TBSP3_IX, TBSP4, TBSP4_IX\) ONLINE USE TSM TAKEN AT 20140227143006 REPLACE EXISTING WITHOUT PROMPTING"*
- **Table spaces on all partitions**
  - *db2_all "|| db2 RESTORE DATABASE TESTDB TABLESPACE \(TBSP5, TBSP5_IX, TBSP6, TBSP6_IX\) ONLINE USE TSM TAKEN AT 20140227143006 REPLACE EXISTING WITHOUT PROMPTING"*

Roll Forward – Quick Note

- Apply transactions in the database log files
- Usually run after restore – to recover activity after backup
- Roll forward could only be run from catalog partition
- Default behavior -- DB / TBSP roll forward applies to all database partitions (use 'EXCEPT' clause for specific partitions)
- *Recover from a rouge DML run immediately after backup:*
  - *db2 "ROLLFORWARD DATABASE TESTDB TO END OF BACKUP **AND STOP** TABLESPACE (TBSP5, TBSP5_IX, TBSP6, TBSP6_IX) ONLINE"*
- *Point in Time Roll Forward*
  - *db2 "ROLLFORWARD DATABASE TESTDB TO yyyy-mm-dd-hh.mm.ss **AND STOP** TABLESPACE (TBSP5, TBSP5_IX, TBSP6, TBSP6_IX) ONLINE"*

Runstats in a DPF database

- Runstats' influence on catalog info
  - `$db2 "select card from syscat.tables where tabname = 'TRANSACTION_HASH1' AND TABSCHEMA = 'DB2INST1' with ur" -- 2303808`
  - `$db2 "select count(*) from DB2INST1.TRANSACTION_HASH1" -- 2000000`
- DB2's system catalogs have the row count (card) for this table as 2.30 M, a whopping **15% off** from the **correct value**.
- **This is because, when runstats command is run, by default, DB2 does the runstats on only the 1st partition on which the table is created or the partition currently connected to.**
- DB2's catalog information in a DB2 DPF database is heavily influenced by the choice of distribution keys.

Choose High Cardinality Distribution Key to avoid this problem !

If data skew cannot be controlled, explicitly connect to the partition that has largest row count for runstats – **Moving target !**

#6: DATA SKEW

Data Skew

- Large # of skewed tables ➔ Partition level skew
- Avoid table level skew - Key is to pick high cardinality distribution keys (already discussed)
- Partition level skew – How skewed are top 100 largest tables?
  - How to find partition level skew ? **Use db2top**
- Partition level skew – Bottleneck to Backup, Restore
- db2top (option ‘p’) – Gives partition level skew
- SQL for tablespace skew in bonus section

Partition level Skew

Partition Number	Partition Status	Buffer LWM	Delta BufSent/s	Delta BufRcvd/s	Pool CurrSize	Pool HWM	Channels Free	Space Used	Total Space	Log Current	Log First	Log Last	Number of Pools
0*	Active	70478	6,034	5,409	3.6G	4.3G	2617	254.4G	310.0G	438334	438333	438382	76
1	Active	272498	646	886	4.4G	5.5G	6245	460.5G	519.7G	119169	119168	119217	37
2	Active	272498	283	249	4.4G	5.4G	6245	465.5G	531.5G	120435	120434	120483	35
3	Active	272498	261	214	4.4G	5.5G	6242	488.5G	587.4G	123444	123443	123492	38
4	Active	272498	277	242	4.4G	5.4G	6245	460.4G	521.4G	119547	119546	119595	35
5	Active	272880	596	799	4.4G	5.4G	6239	459.8G	525.0G	119100	119099	119148	38
6	Active	272880	519	666	4.4G	5.4G	6243	475.4G	555.8G	120775	120774	120823	35
7	Active	272880	537	693	4.4G	5.4G	6243	463.3G	525.7G	119338	119337	119386	35
8	Active	272880	296	275	4.4G	5.4G	6243	619.4G	679.4G	120727	120726	120775	35
9	Active	273351	276	235	4.4G	5.4G	6247	460.7G	525.0G	118177	118176	118225	38
10	Active	273351	291	266	4.4G	5.4G	6246	460.1G	541.4G	118944	118943	118992	36
11	Active	273351	298	279	4.4G	5.4G	6247	457.9G	513.2G	119192	119191	119240	35
12	Active	273351	293	268	4.4G	5.4G	6249	457.1G	536.8G	128130	128129	128178	35

#7: INDEX CARDINALITY

Index Cardinality in a DPF database

- Index Cardinality -- Number of distinct values of column(s) that make up a given index
- Index I1 defined on column C1 on a table S1.T1:
  - db2 "select count(distinct(C1)) from syscat.indexes where tabschema = 'S1' and tabname = 'T1' with ur"

- Why should you care?
- Low-cardinality indexes negatively impact performance – **DROP THEM !!!**
- Index cardinality -- straight forward (above SQL) – Single Partitioned Database
- Tricky in DPF -- the catalog data for indexes that do not have primary key columns in them seems to be incorrect.

Example to illustrate Index Cardinality in a DPF database

DB2INST1.EMPLOYEE		
Column	Data Type	
COL1	INTEGER	Primary Key
COL2	BIGINT	Unique Values
COL3	TIMESTAMP	

INDEX_NAME	FULLKEYCARD (SYSCAT.INDEXES)	TABLE_CARD (SYSCAT.INDEXES)	INDEX_CARD_PERCENT = 100*(FULLKEYCARD / TABLE_CARD)
PK_EMPLOYEE (COL1)	10361208	10361208	100
I1_EMPLOYEE (COL2, COL1) (has PK column)	10361208	10361208	100
I2_EMPLOYEE (COL3) (no PK column)	430632	10361208	4.15
I3_EMPLOYEE (COL2) (no PK column)	863434	10361208	8.33 (Expected 100)

8.33% -- Very low Index Cardinality DROP INDEX!!!

Example to illustrate Index Cardinality in a DPF database

- **Multiply INDEXCARD (from catalogs) by no. of partitions in which the table is on to get the correct FULLKEYCARD value.**

For I3_EMPLOYEE:

- Multiply 8.33 (index card from catalogs) by **12** to get corrected index cardinality percentage (8.33*12) = 99.96 ~ 100% (It was an index on unique values)
- You might drop indexes based on their (low) cardinality if you strictly go by catalog data
- How to calculate the 'corrected' Index Cardinality?
  - http://db2talk.com/2014/04/28/index-cardinality-in-a-db2-multi-partitioned-dpf-database/

#8: ANALYZING CONTENT IN DB2DIAG.LOG

db2diag.log

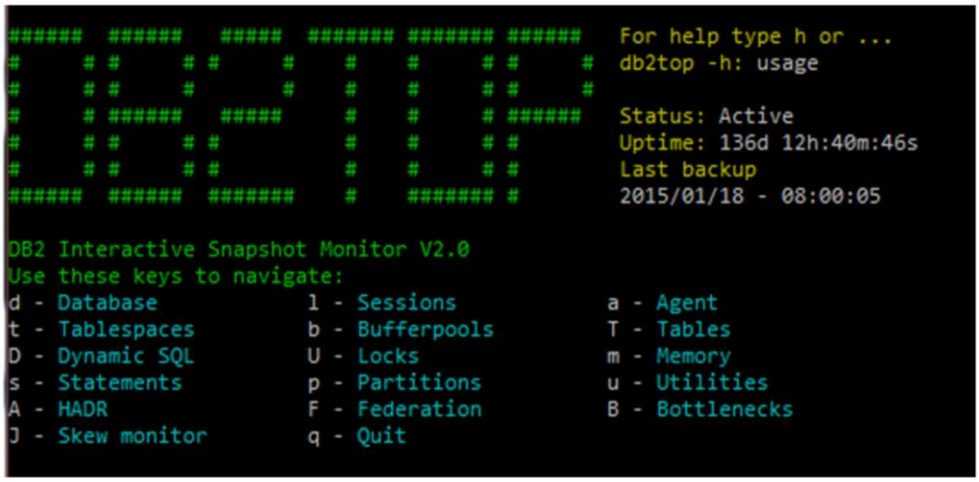
- # of db2diag.log(s) → # of servers in DPF system

```
2015-02-04-15.29.34.320410-300 E32879553E447       LEVEL: Info
PID     : 12171                TID  : 46912916941120PROC : db2sysc 2
INSTANCE: db2inst1             NODE : 002
EDUID   : 906                  EDUNAME: db2logmgr (EDW) 2
FUNCTION: DB2 UDB, data protection services, sqlpgArchiveLogFile, probe:3180
DATA #1 : <preformatted>
Completed archive for log file S0118169.LOG to TSM chain 2 from
/db2fs/db2inst1/NODE0002/SQL00001/SQLOGDIR/.
```

- **db2diag -global -merge $DIR** → Merge logs
- **db2diag -rc $HEXCODE** → reason-code for hexa-decimal code
- **db2diag -A** → Archives existing db2diag.log and creates a new one on the current node.
- **db2diag -A - global** → Archives existing db2diag.log and creates a new one on **all nodes**

#9 DB2TOP

db2top



Single view for all database partitions

SQL activity, Locks, Sessions, Utilities on all partitions

db2top

Partition Number	Partition Status	Buffer LWM	Delta BufSent/s	Delta BufRcvd/s	Pool CurrSize	Pool HWM	Channels Free	Space Used	Total Space	Log Current
0*	Active	70478	1,629	3,128	3.4G	4.3G	2628	232.4G	297.0G	422303
1	Active	272498	347	145	4.3G	5.5G	6248	412.3G	472.7G	115404
2	Active	272498	348	150	4.3G	5.4G	6248	416.6G	486.3G	116628
3	Active	272498	351	152	4.3G	5.5G	6248	436.6G	539.2G	119442
4	Active	272498	355	152	4.3G	5.4G	6248	412.3G	473.6G	115779
5	Active	272880	348	148	4.3G	5.4G	6253	411.7G	475.3G	115350
6	Active	272880	345	147	4.3G	5.4G	6253	425.1G	506.7G	116874
7	Active	272880	342	145	4.3G	5.4G	6253	414.6G	481.2G	115533
8	Active	272880	341	145	4.3G	5.4G	6253	558.6G	619.6G	116754
9	Active	273351	343	144	4.3G	5.4G	6249	412.5G	473.7G	114414
10	Active	273351	339	143	4.3G	5.4G	6249	411.9G	494.8G	115160
11	Active	273351	337	143	4.3G	5.4G	6249	410.0G	468.7G	115433
12	Active	273351	337	143	4.3G	5.4G	6249	418.0G	486.6G	124305

db2top

Utilities

Hash Value	# of entries	Utility Start Time	Utility Type	Uti Pri	Utility State	Invoker Type	Completed Work	Work Unit	Phase Prog%	Phase Start Time	Progress Description
2122241	13+	06:12:14.745364	Backup	0	Execute	User		0 Bytes	0%	06:12:14.745374	

Locks Linux,part=[25/25],DB2INST1:EDWDBDV
Cpu=1,[qp=off]

#10 HANDY SQLS

Which database partitions is the tablespace on?

```
SELECT  CHAR(tbspace, 20) TABLESPACE,
CHAR(pgs.dbpgname,20) AS partition_group,
Listagg(DBPARTITIONNUM, ',') within group (ORDER BY DBPARTITIONNUM) A
S PARTITIONS
FROM    syscat.dbpartitiongroupdef pgs ,
        syscat.tablespaces tablespaces
WHERE   tablespaces.ngname = pgs.dbpgname
AND     tablespaces.tbspace = 'TBSP1'
GROUP BY CHAR(pgs.dbpgname,20),
        tbspace WITH UR;
------------------------------------------------------------------
TBSP1           PDPG            1,2,3,4,5,6,7,8,9,10,11,12
```

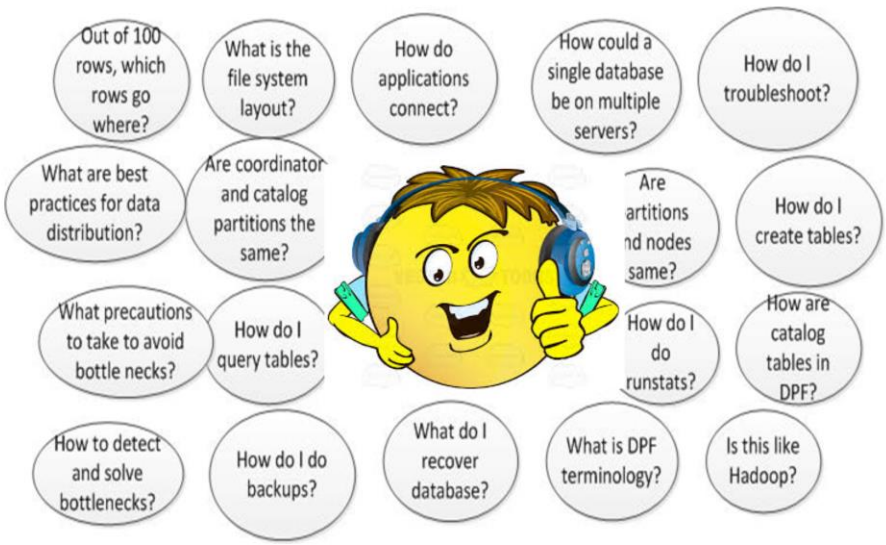
Which partitions does a table belong to?

```
SELECT   CHAR(tabschema, 20) AS SCHEMA,
         CHAR(tabname, 40)    AS table,
         CHAR(tablespaces.tbspace, 20) AS TABLESPACE,
         CHAR(pgs.dbpgname,20) AS partition_group,
         Listagg(DBPARTITIONNUM, ',') within group (ORDER BY DBPARTITIONNUM) AS
PARTITIONS
FROM     syscat.tables tables ,
         syscat.tablespaces tablespaces,
         syscat.dbpartitiongroupdef pgs
WHERE    tabschema = 'EDW'
AND      tabname = 'EMPLOYEE'
AND      tablespaces.tbspace = tables.tbspace
AND      tablespaces.dbpgname = pgs.dbpgname
GROUP BY pgs.dbpgname,
         tabname,
         tabschema,
         tablespaces.tbspace WITH UR;
```

SQL does not work for range partitioning tables

```
SCHEMA           TABLESPACE     PARTITION_GROUP  PARTITIONS
--------     ------------ ---------- -       ------------ -----------------
EDW              DV20140902114851  PDPG  1,2,3,4,5,6,7,8,9,10,11,12
```

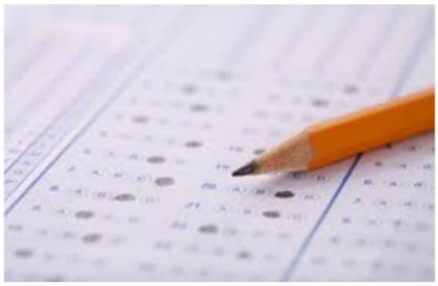
Summary


Out of 100 rows, which rows go where?
What is the file system layout?
How do applications connect?
How could a single database be on multiple servers?
How do I troubleshoot?
What are best practices for data distribution?
Are coordinator and catalog partitions the same?
Are artitions nd nodes same?
How do I create tables?
What precautions to take to avoid bottle necks?
How do I query tables?
How do I do runstats?
How are catalog tables in DPF?
How to detect and solve bottlenecks?
How do I do backups?
What do I recover database?
What is DPF terminology?
Is this like Hadoop?

Thank you for attending.

Any Questions?

Pavan Kristipati
The Huntington National Bank
pavan.kristipati@huntington.com



C02 – DB2 DPF: What a single partitioned DBA needs to know in 10 points

Bonus Slides

The DB2NODE Environment Variable

- Similar to DB2INSTANCE environment variable

DB2INSTANCE	DB2NODE
Switch between instances	Switch between partitions in DPF

- Find out the active partition
- Use 'export' to switch
- Usage scenario:

Set transaction logs path for a partition:

a) Switch to partition using 'export'

b) db2 "update db cfg for …."

```
db2 "values (current dbpartitionnum)"
1
-----------
          14
1 record(s) selected.
```

```
$DB2NODE=12
$export DB2NODE
$db2 terminate
DB20000I  The TERMINATE command completed successfully.
## Connect to database
$db2 "values (current dbpartitionnum)"
1
-----------
          12
 1 record(s) selected.
```

Refer to http://www-01.ibm.com/support/knowledgecenter/SSEPGG_9.1.0/com.ibm.db2.udb.uprun.doc/doc/r0006351.htm for more details on these examples.

db2pd

- Useful command for monitoring and troubleshooting
- No latches => faster retrieval and no engine resources are used
- Multiple options to define scope of the command
- Command output depends on the partition/node it is run on
- Expects database to be active for 'database' option
- db2pd –help for usage and options

db2pd for DPF

Option	Description
Addnode	Shows the progress of the "ADD DATABASE PARTITION SERVER" operation. It must be run on the partition Server that is being added.
dbptnmem	Shows database partition memory statistics.
-fcm	Shows information about the fast communications manager (FCM) that facilitates communication between the partitions.
alldbpartitionnums	Output for the db2pd command will be shown for all db partitions on the physical Server the command is run on.
dbpartitionnum <number>	Output for the db2pd command will be shown once for the specified partition or range of partitions specified.
-member all (-global)	Runs command on all (remote) hosts

How to save db2pd output from multiple nodes to a local file?

- Use the option '-file filename' to save output on the machine the command is run

```
$db2pd -file db2pd_osinfo –osinfo –host host1, host2
$ls -ltr db2pd_osinfo
-rw-r--r-- 1 db2inst1 bcuigrp 221 2015-01-19 14:33 db2pd_osinfo
```

db2pd - alldbpartitionnums vs. member all

- **Scenario:** In a DPF database, run 'db2pd' command to collect information on Active Statements on all database partitions.
- Usual Suspect – 'alldbpartitionnums' clause.
- db2pd -d proddb -**alldbpartitionnums** -activestatements
- Or is it –member all?
- db2pd -d proddb -**member all** -activestatements
- **What is the right option to use?**

How to display info from all db partitions
Choice #1: db2pd - alldbpartitionnums

db2inst1@hostdata1>db2pd -d PRODDB -alldbpartitionnums -activestatements

(removed repeating header for partitions #6,#7,#8)

Database Partition 5 -- Database PRODDB -- Active -- Up 136 days 13:28:10 -- Date 2015-01-19-13.21.14.752850

Active Statement List:

Address AppHandl [nod-index] UOW-ID StmtID AnchID StmtUID
EffISO EffLockTOut EffDegree EntryTime StartTime LastRefTime
0x00002AAB9C7710A0 1503 [000-01503] 2 2 910 21781
1 -1 0 Mon Jan 19 13:21:04 Mon Jan 19 13:21:04 Mon Jan 19 13:21:04

Node	Partitions
Hostadm01	0
Hostdata01	1,2,3,4
Hostdata02	5,6,7,8
Hostdata03	9,10,11,12

Database Partition 6 -- Database PRODDB -- Active -- Up 136 days 13:28:10 -- Date 2015-01-19-13.21.14.769036

0x00002AAB9E225B20 1503 [000-01503] 2 2 910 21781
1 -1 0 Mon Jan 19 13:21:04 Mon Jan 19 13:21:04 Mon Jan 19 13:21:04

Database Partition 7 -- Database PRODDB -- Active -- Up 136 days 13:28:10 -- Date 2015-01-19-13.21.14.784791

0x00002AAB9C51A500 1503 [000-01503] 2 2 910 21781
1 -1 0 Mon Jan 19 13:21:04 Mon Jan 19 13:21:04 Mon Jan 19 13:21:04

Database Partition 8 -- Database PRODDB -- Active -- Up 136 days 13:28:10 -- Date 2015-01-19-13.21.14.800534

0x00002AAB9C9CDFC0 1503 [000-01503] 2 2 910 21781
1 -1 0 Mon Jan 19 13:21:04 Mon Jan 19 13:21:04 Mon Jan 19 13:21:04

How to display info from all db partitions
Choice #2: db2pd - member all

db2inst1@hostdata1>db2pd -d PRODDB –member all –activestatements

(removed header ; replaced 'activestatements' output with '.....' for brevity)

Database Partition 9 -- Database PRODDB -- Active -- Up 136 days 13:28:45 -- Date 2015-01-19-13.21.51.192125

.....

Database Partition 10 -- Database PRODDB -- Active -- Up 136 days 13:28:45 -- Date 2015-01-19-13.21.51.202655

.....

Database Partition 11 -- Database PRODDB -- Active -- Up 136 days 13:28:45 -- Date 2015-01-19-13.21.51.213070

.....

Database Partition 12 -- Database PRODDB -- Active -- Up 136 days 13:28:45 -- Date 2015-01-19-13.21.51.223456

.....

hostdata03: db2pd -d PRODDB -member ... completed ok

Database Partition 5 -- Database PRODDB -- Active -- Up 136 days 13:28:42 -- Date 2015-01-19-13.21.46.080808

.....

Database Partition 6 -- Database PRODDB -- Active -- Up 136 days 13:28:42 -- Date 2015-01-19-13.21.46.091378

.....

Database Partition 7 -- Database PRODDB -- Active -- Up 136 days 13:28:42 -- Date 2015-01-19-13.21.46.102688

.....

Database Partition 8 -- Database PRODDB -- Active -- Up 136 days 13:28:42 -- Date 2015-01-19-13.21.46.113151

.....

hostdata02: db2pd -d PRODDB -member ... completed ok

(Output continued next slide...)

Node	Partitions
Hostadm01	0
Hostdata01	1,2,3,4
Hostdata02	5,6,7,8
Hostdata03	9,10,11,12

How to display info from all db partitions
Choice #2: db2pd - member all

db2pd output continued...

```
Database Partition 1 -- Database PRODDB -- Active -- Up 136 days 13:28:46 -- Date 2015-01-19-13.21.48.819865
.....
Database Partition 2 -- Database PRODDB -- Active -- Up 136 days 13:28:44 -- Date 2015-01-19-13.21.48.830437
.....
Database Partition 3 -- Database PRODDB -- Active -- Up 136 days 13:28:44 -- Date 2015-01-19-13.21.48.840849
.....
Database Partition 4 -- Database PRODDB -- Active -- Up 136 days 13:28:44 -- Date 2015-01-19-13.21.48.851231
.....
hostdata01: db2pd -d PRODDB -member ... completed ok

Database Partition 0 -- Database PRODDB -- Active -- Up 136 days 14:08:13 -- Date 2015-01-19-13.21.47.851729
.....
hostadm01: db2pd -d PRODDB -member ... completed ok
```

'-member all' displays db2pd output from all hosts and partitions (if applicable) and hence is the right option to use in this scenario