



Technical Report

Deploying an IBM DB2 Multipartition Database on a NetApp Storage System and Linux

Bobby Oommen, Jawahar Lal, Ashwath Narayan, NetApp
January 2012 | TR-3988

SUMMARY

This technical report focuses on configuring, designing, and implementing a multipartition DB2[®] database in a NetApp[®] storage environment. Deploying the DB2 Database Partitioning Feature (DPF) with NetApp storage can deliver additional cost-saving benefits provided by backup and restore based on NetApp Snapshot[™] technology, cloning, and Flash Cache. The purpose of this document is to integrate IBM[®] and NetApp best practices for planning, configuring, designing, implementing, and administering IBM DB2 9.7 DPF for Linux[®] running on System x3650 and a NetApp storage environment.

TABLE OF CONTENTS

| | | |
|----------|---|-----------|
| 1 | INTRODUCTION | 4 |
| 1.1 | ASSUMPTIONS | 4 |
| 1.2 | BUSINESS NEEDS | 4 |
| 2 | IBM DB2 OVERVIEW | 5 |
| 2.1 | SHARED-NOTHING PARALLELISM | 5 |
| 2.2 | IBM DB2 DATA PARTITIONING OVERVIEW | 6 |
| 3 | NETAPP STORAGE OVERVIEW | 6 |
| 3.1 | NETAPP STORAGE LAYERS | 6 |
| 3.2 | NETAPP OVERVIEW | 8 |
| 3.3 | NETAPP SNAP CREATOR FRAMEWORK | 8 |
| 3.4 | WHY NETAPP STORAGE FOR DB2? | 9 |
| 4 | ARCHITECTURE | 11 |
| 4.1 | SOFTWARE AND HARDWARE COMPONENTS | 11 |
| 4.2 | NETWORK SPECIFICATIONS | 12 |
| 5 | CONFIGURATION OVERVIEW AND CONSIDERATIONS | 13 |
| 5.1 | DATABASE DESIGN | 13 |
| 5.2 | STORAGE SYSTEM CONFIGURATION | 17 |
| 5.3 | VOLUME CONSIDERATION | 21 |
| 6 | INSTALLATION AND CONFIGURATION | 22 |
| 6.1 | INSTALLING DB2 | 22 |
| 6.2 | CREATING A DB2 INSTANCE | 22 |
| 6.3 | UPDATING THE DB2NODE.CONFIG FILE | 23 |
| 6.4 | CREATING THE DATABASE | 24 |
| 6.5 | CREATING DATABASE PARTITION GROUPS | 24 |
| 6.6 | CREATING A BUFFER POOL | 25 |
| 6.7 | PLACING THE DATABASE TRANSACTION LOG IN THE NEW DIRECTORY | 26 |
| 6.8 | CREATING TABLE SPACES | 26 |
| 6.9 | CREATING TABLES AND POPULATING DATA | 26 |
| 7 | DPF DATABASE BACKUP AND RESTORE | 27 |
| 7.1 | BASIC CONFIGURATION | 27 |
| 7.2 | DATABASE BACKUP USING SNAP CREATOR FRAMEWORK | 28 |
| 7.3 | DATABASE RECOVERY USING SNAP CREATOR | 29 |
| 8 | APPENDIXES | 31 |
| 8.1 | DATABASE MANAGER CONFIGURATION PARAMETERS | 31 |
| 8.2 | DATABASE CONFIGURATION PARAMETER SETTINGS | 32 |

| | | |
|----------|--------------------------------|-----------|
| 8.3 | KERNEL PARAMETER SETTINGS..... | 35 |
| 9 | REFERENCES | 35 |
| 9.1 | NETAPP TECHNICAL REPORTS | 35 |
| 9.2 | NETAPP WEB SITES | 35 |
| 9.3 | IBM WEB SITES..... | 35 |

LIST OF TABLES

| | | |
|-----------|--|----|
| Table 1) | Hardware components. | 12 |
| Table 2) | Software components. | 12 |
| Table 3) | Network specifications. | 13 |
| Table 4) | Table space considerations. | 14 |
| Table 5) | DB2 registry variable settings. | 15 |
| Table 6) | Database manager configuration parameters. | 16 |
| Table 7) | Database configuration parameters. | 16 |
| Table 8) | Memory allocation on administrative node and data node. | 17 |
| Table 9) | FAS3270 storage controller specifications. | 18 |
| Table 10) | Physical node-1 specifications. | 18 |
| Table 11) | Physical node-2 specifications. | 20 |
| Table 12) | File system name and purpose on the DB2 physical nodes. | 21 |
| Table 13) | Database partitions..... | 23 |
| Table 14) | Snap Creator basic configuration. | 27 |
| Table 15) | Kernel parameter settings. | 35 |

LIST OF FIGURES

| | | |
|-----------|---|----|
| Figure 1) | Logical relationship between aggregate, volume, and physical disks..... | 7 |
| Figure 2) | Snap Creator Framework architecture and components. | 9 |
| Figure 3) | DB2 DPF Linux setup on NetApp storage..... | 11 |
| Figure 4) | DPF network diagram..... | 13 |
| Figure 5) | Database partition group layout. | 25 |
| Figure 6) | Snap Creator backup process flow. | 29 |
| Figure 7) | Snap Creator database restore process flow..... | 30 |

1 INTRODUCTION

With explosive data growth, database size and performance requirements have grown proportionally. It is common to deploy a large database across multiple servers to achieve the scalability and performance needed for a database environment. In DB2 environments, scaling to multiple servers requires the Database Partitioning Feature (DPF) of DB2. DPF offers scalability to distribute a large database over multiple partitions (logical or physical) using a shared-nothing architecture. DPF can be beneficial to both standalone symmetric multiprocessing (SMP) servers and environments consisting of more than one server. The divide-and-conquer processing of DPF can enhance scalability within a single server (by scaling up) or across a cluster of servers (by scaling out). One of the most obvious reasons to use DPF is to increase the performance of query workloads and insert/update/delete operations.

This technical report focuses on configuring, designing, and implementing a multipartition DB2 database in a NetApp storage environment. Deploying DB2 DPF with NetApp storage can deliver additional cost-saving benefits provided by backup and restore based on NetApp Snapshot technology, cloning, and Flash Cache. The purpose of this document is to integrate IBM and NetApp best practices for planning, configuring, designing, implementing, and administering IBM DB2 9.7 DPF for Linux running on System x3650 and a NetApp storage environment. This document offers guidance to customers who want to integrate the following components:

- IBM DB2 9.7 Enterprise Server Edition
- NetApp FAS or IBM N series storage
- IBM System x3650 M3 Express server

Note: The IBM x3650 can be replaced with any other server with the equivalent configuration and CPU architecture.

1.1 ASSUMPTIONS

This document describes the implementation of the DPF feature of IBM InfoSphere[®] Warehouse Enterprise Edition V9.7 in a Linux and NetApp storage system environment. It assumes prior knowledge of DB2, Linux administration, and the operation of NetApp storage systems. All examples in this technical report were tested using DB2 Enterprise Server Edition for Warehouse, version 9.7 with FixPak 2 or later running under RHEL 6.0; and NetApp FAS3270 storage systems with the Data ONTAP[®] 8.0.1 operating system.

NetApp also assumes that documentation is available for IBM DB2 9.7 for the relevant operating system (OS) version. Reference documents can be found at the [IBM InfoSphere Warehouse Information Center](#) Web site, and manuals can be downloaded from the [IBM support](#) Web site.

The deployment in the NetApp test laboratory was based on a modular approach to offer greater scalability so that physical servers with partitions could be added to an existing setup at any time.

1.2 BUSINESS NEEDS

Since the last decade, the size of both structured and unstructured data has grown exponentially. In addition, database use has evolved from the storage, modification, and retrieval of information to the complex analysis and diagnosis of business problems.

Continuous data growth emphasizes complex analysis and diagnosis, faster reaction to events, and deeper analysis of options, resulting in complex and demanding technical requirements. Some of the key business requirements for modern databases are:

- **Support for many concurrent users.** Instead of a few users, databases must now often support large communities of employees, partners, or customers concurrently accessing and updating data.

- **Maximum data availability.** Since enterprises rely on databases to support operational business processes, data availability requirements rise and often attain mission-critical status. Therefore, databases must often operate on a continuous or near-continuous schedule.
- **Performance and scalability.** As the size and complexity of data grow, the ability to store and retrieve data at a speed that meets business requirements becomes increasingly important. In addition, a database should have the ability to scale to accommodate future data growth.
- **Data security.** Unauthorized data access is a perpetual threat that can result in huge losses of money and business, and even of business reputation. A database must offer a comprehensive suite of security features that effectively and decisively minimize this threat.
- **Data source for intensive analytics.** Large DPF databases are commonly used directly or indirectly as a source for analyzing data. As business strategies have become more sophisticated and data volumes have continued to increase, the demand for analyzing the data has also increased substantially. As tolerance for downtime decreases, the tolerance for delay in analysis also decreases because the ability to access data is required at all times.
- **Virtualization support.** Virtualization is useful to businesses with multiple database servers that do not use their full hardware capacity. The cost-saving benefits of virtualization are clear: server consolidation, space savings, and reduced power and cooling.
- **Easy management and maintenance.** Too often, database administrators (DBAs) must spend their time managing and maintaining the database systems rather than offering value additions to the business, such as focusing on and planning for growth, cost savings, and so forth. A modern database is expected to automate and simplify both management and maintenance tasks such as memory tuning, workload management, system health monitoring, and storage features.

2 IBM DB2 OVERVIEW

2.1 SHARED-NOTHING PARALLELISM

One of the key goals of the parallel database architecture is to allow large and complex tasks to be performed quickly by operating on many database records concurrently. In parallel database architecture, the query is broken into pieces and the pieces are farmed out to many independently operating units, often called database worker processes. Because all of the workers operate simultaneously, the task is completed rapidly.

However, not all types of parallel architecture are created equal. In particular, as databases increase in scale and complexity and as the intensity of data analysis increases, some parallel architectures tend to perform better than others.

One of the fundamental strengths of DB2, in comparison with databases from competitors, is its shared-nothing parallel architecture. Such an architecture involves many database worker processes that operate on data independently and simultaneously. These processes neither share data or resources nor require coordination to complete the work. Hence, these database processes do not contend with one another for accessing data or resources, and this separation contributes to linear scalability for the most common database operations. Scalability becomes very critical to meeting business needs as data volumes and workloads continue to grow. In shared-nothing architecture, the database is divided into pieces called partitions, and the data in each partition is accessed by only one database worker process.

In all parallel database architectures, there is some coordination of the database worker processes. Execution of the many separate pieces of work must be coordinated. Results of the subqueries must be recombined to deliver a result to the user. All of this happens automatically and is invisible to the user. In DB2, this coordination is the responsibility of a coordinator process. DB2 has been designed so that a substantial amount of work is completed in parallel by the worker processes, thus reducing the work for the coordinator process. In addition, the DB2 design aims to provide efficient coordination and data

movement between the worker processes. This contributes to performance and optimizes the system to handle larger databases and workloads.

In a strong shared-nothing architecture, there is no inherent limit on system capacity. Systems may be configured as clusters of servers, and the capacity of the system may be expanded incrementally by adding servers. Thus, a data warehouse can start as a small unit by deploying only a few servers and grow in data volume, usage, or other dimensions in response to the growth in requirements. Shared-nothing architecture generally enhances scalability by minimizing the coordination required among the servers running the database worker processes. The DPF feature of InfoSphere Warehouse Enterprise Edition, described in the subsequent section, is an ideal implementation of a shared-nothing architectural concept.

2.2 IBM DB2 DATA PARTITIONING OVERVIEW

InfoSphere Warehouse Enterprise Edition offers a robust performance and scalability feature known as DPF. The DPF feature allows customers to create a database with multiple partitions. A database partition is part of a database that consists of its own data, indexes, configuration files, and transaction logs. Tables can then be located in one or more database partitions. Processors associated with each database partition are used to satisfy table requests. Data retrieval and update requests are automatically divided into subrequests and executed in parallel among the applicable database partitions. Starting with DB2 version 9.5, the DB2 DPF is available only in IBM InfoSphere Warehouse product editions.

Some of the foundation features of IBM DB2 9.7 that make it ideal for OLTP and warehouse environments are:

- A shared-nothing parallel database architecture that provides performance, scalability, and high data availability
- A long-term focus on system efficiency, based on extensive capabilities for physical database design, I/O performance, efficient cross-partition network utilization, and cost-based query optimization
- Mixed workload management capable of satisfying multiple service-level objectives on a single system managing a single logical copy of the data
- The efficient Data Compression feature with proven performance improvement

In general, most DPF deployments are for the purpose of data warehousing and business intelligence. NetApp highly recommends the DPF for large, complex query environments because its shared-nothing architecture allows outstanding scalability.

A typical partitioned database system usually has one processor core per data partition. For example, a system with n processor cores might have the catalog on partition 0 and have n additional data partitions. If the catalog partition is used heavily (for example, to hold single-partition dimension tables), a processor core might be allocated to it. If the system supports many concurrent active users, two cores per partition might be required. As a general guideline, NetApp recommends planning on about 250GB of active raw data per partition.

For more information about partitioned database configuration best practices, refer to the [IBM InfoSphere Warehouse Information Center](#).

3 NETAPP STORAGE OVERVIEW

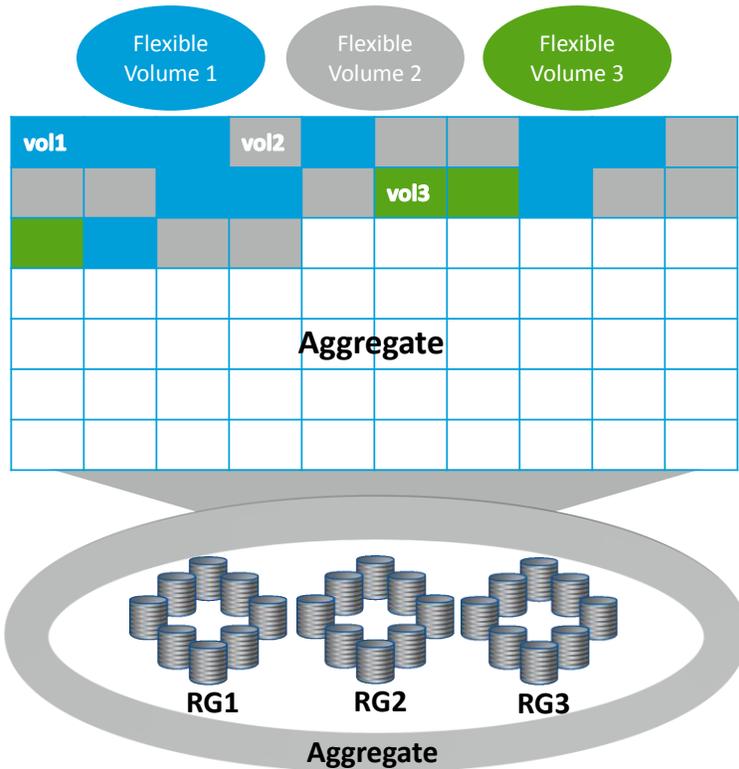
3.1 NETAPP STORAGE LAYERS

Data ONTAP 7 and higher offer a storage virtualization approach that decouples the volume from the physical storage disk. The physical storage disks are tied to a storage layer named the aggregate. The volumes that are virtualized are known as flexible volumes or NetApp FlexVol[®] volumes.

NetApp supports the following logical storage layers:

- **Aggregate.** An aggregate is a pool of physical disks that adds a layer of abstraction between the physical disks and the volumes. It possesses its own RAID configuration, plex structure, and set of disks. One or more flexible volumes can be created in an aggregate.
- **Flexible volume (FlexVol volume).** The NetApp FlexVol approach to storage virtualization allows the creation of one or more FlexVol volumes (virtual volumes) in an aggregate. The volume can grow or shrink according to the application's needs. All FlexVol volumes share the physical storage resources, such as disks RAID configuration and the plex structure of the common containing aggregate. FlexVol volumes improve the utilization of storage and save the expense of acquiring more disk space. They also improve I/O performance and reduce bottlenecks by distributing volumes across all available disk drives. NetApp Snapshot technology is available at the FlexVol level. Each volume can have up to 255 Snapshot copies.
- **A FlexVol feature called FlexClone®.** A FlexClone volume can be created from the FlexVol volume itself, or an existing Snapshot copy can be used. A FlexClone volume is similar to the source volume (its parent FlexVol volume or another FlexClone volume), but it uses no additional physical storage. Figure 1 illustrates the logical relationship between aggregates, flexible volumes, and physical disks.
- **LUN (logical unit number).** SAN or iSAN (FCoE or iSCSI) uses block allocations, and block allocations are managed by using LUNs. To create a LUN, it is necessary to first create a volume and then make it accessible by mapping it to an initiator group. Initiator groups contain one or more Fibre Channel Protocol (FCP) worldwide names (WWNs), or iSCSI qualifying names (IQNs) that are mapped to a LUN on the NetApp storage system. The mapping process is also known as LUN masking.

Figure 1) Logical relationship between aggregate, volume, and physical disks.



3.2 NETAPP OVERVIEW

Snapshot is a feature of the WAFL[®] (Write Anywhere File Layout) storage virtualization technology, which is a feature of Data ONTAP, the microkernel that is shipped with every NetApp storage system. A Snapshot copy is a locally retained, point-in-time frozen image of a WAFL volume that provides easy access to old versions of files, directory hierarchies, and LUNs. The high performance of Snapshot copies makes them highly scalable. A Snapshot copy takes only a few seconds to create, regardless of the size of the volume or the level of activity on the NetApp storage system. After a Snapshot copy has been created, changes to data objects are reflected in updates to the current version of the objects, as if the Snapshot copy did not exist. Meanwhile, the Snapshot versions of the data remain frozen. The Snapshot copy does not move data blocks; therefore, the time needed to create a Snapshot copy is independent of the size of the database. Snapshot copies incur no performance overhead; users can store up to 255 Snapshot copies per volume on the storage system, all of which can be accessible as read only. Online versions of the data Snapshot copy can be integrated into the DB2 database backup policy.

NetApp Snap Creator[™] Framework is a free tool that enables the integration of backup policies and the automation of backup and restore processes for the whole environment. This tool offers many features and functions in addition to backup and restore. Snapshot copies vastly improve the frequency and reliability of backups because they incur virtually no performance overhead and can be safely created while a database is up and running. Similarly, restore operations can be automated with Snap Creator. For more information on Snap Creator, refer to section 3.3, “NetApp Snap Creator Framework.”

NetApp SnapRestore[®] software uses stored Snapshot backups to recover entire file systems or data volumes in seconds, regardless of capacity or number of files. SnapRestore software uses the Snapshot feature of Data ONTAP to help recover anything from a single file to a multiterabyte volume within seconds. The Snapshot process is automatic and almost instantaneous, and it takes up very little storage space. The easy, one-command restoration that SnapRestore provides allows users to select data to recover from any Snapshot copy on the system. Traditional data recovery requires that all of the data be copied from the backup to the source, which requires much more time and disk capacity.

SnapRestore helps to:

- Restore databases quickly.
- Test changes with easy restores to the baseline copy.
- Recover quickly from virus attacks or after user or application error.

For more information on using Snapshot or SnapRestore technology and other NetApp products for data protection, refer to the [Data Protection Best Practices Index](#) Web page on the NetApp [Support](#) (formerly NOW[®]) site.

3.3 NETAPP SNAP CREATOR FRAMEWORK

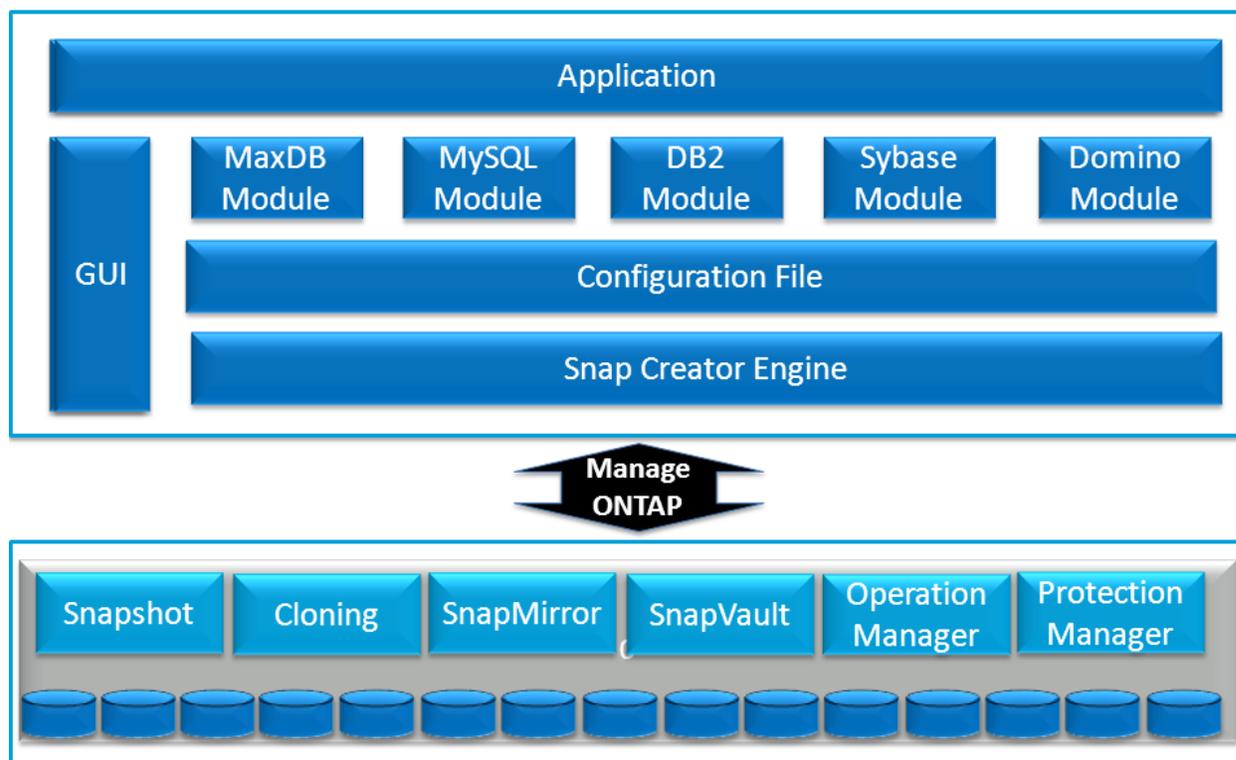
NetApp Snap Creator Framework offers rich features and allows customers to integrate DB2 and other applications and components to leverage Snapshot technology. It is a platform-independent and flexible framework that is fully supported and freely available to all NetApp customers. It can help customers to automate backup, restore, replication, cloning, and data protection processes. It eliminates the need to create and manage custom scripts and provides a flexible and reliable solution. Snap Creator comes with default plug-ins, including a plug-in for DB2 and a set of application programming interfaces (APIs) that can be used to develop custom plug-ins. Plug-ins may be needed for application consistency and recovery of the environment. Snap Creator allows policy-based Snapshot management and role-based access, and it is integrated with a number of other NetApp products. It has a robust error-handling process, and it can send alerts to an e-mail distribution list or communicate with external monitoring systems. As shown in Figure 2, the Snap Creator architecture consists of three components:

- **Snap Creator engine.** The engine is the main component of Snap Creator; it can be installed on a central backup server or on the DB2 server where the database is installed. Snap Creator can also be

integrated with an existing backup architecture. Snap Creator communicates with the NetApp storage system by using Manage ONTAP® calls.

- **Snap Creator configuration file.** The configuration file is the brain behind Snap Creator, and it can be customized as needed. Snap Creator comes with a default configuration file that can be used as a template to create a configuration file for the database. Although it supports multiple configuration files per database, only one file can be active at a time.
- **Application modules (DB2 in this case).** The database modules are used for application consistency for the Snapshot copy. In the case of DB2, the application-consistent Snapshot copies are created by setting the database in I/O suspend mode during the Snapshot creation process. After the Snapshot copy is created, the DB2 module reverts the database to normal mode. Figure 2 provides an overview of the Snap Creator architecture and components. For more information, refer to the Snap Creator Framework documentation on the NetApp [Support](#) site.

Figure 2) Snap Creator Framework architecture and components.



3.4 WHY NETAPP STORAGE FOR DB2?

This section describes the key benefits of using IBM DB2 DPF on NetApp storage.

- **Unified storage.** With a unified multiprotocol architecture, NetApp storage solutions take advantage of the benefits of Ethernet and work as a unification engine, supporting Network File System (NFS), Common Internet File System (CIFS), iSCSI, and Fibre Channel over Ethernet (FCoE) in the same system. They also leverage traditional Fibre Channel storage investments.
- **Fast backup.** Database backup windows continue to shrink or are becoming nonexistent as businesses demand 24/7 data availability. NetApp offers excellent value for database customers for DB2 backup and recovery. The storage-efficient NetApp Snapshot technology can be used to perform backups with no downtime or performance penalty. Snapshot backups are completed in a few seconds, regardless of the size of the database or the level of activity on the NetApp storage system.

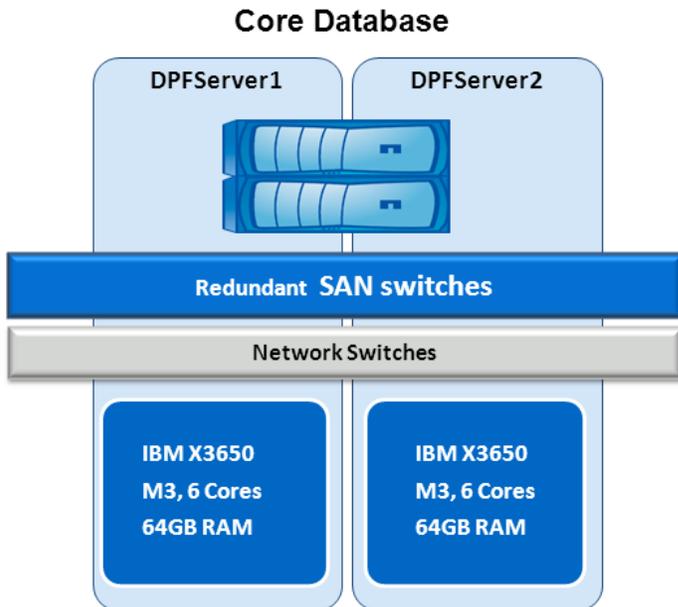
This innovative approach dramatically reduces backup window requirements and allows DBAs to schedule frequent database backups.

- **Quick recovery.** Using the Data ONTAP SnapRestore technology, an entire database can be restored in a few seconds from a Snapshot backup. Because no data copying is involved, a large amount of time is saved when a database's current state is restored as it was when the Snapshot copy was created. In addition, the Snapshot creation process is time efficient and allows the retention of a large number of Snapshot copies. The SnapRestore process can also reduce the amount of time needed to perform a roll-forward recovery operation against a database.
- **High availability.** Organizations of all sizes now require 24/7 availability. Companies cannot tolerate scheduled downtime or the extended periods of slow system response that are often caused by traditional database backup methods. Snapshot copies, on the other hand, can be created in a short time with no impact on system response time, affording high availability and uninterrupted system response.
- **High reliability.** The RAID architecture used for NetApp storage systems offers greater reliability than direct-attached storage (DAS). If a RAID member disk fails, it is automatically reconstructed (using parity disk data) without user intervention. Double-parity RAID (RAID-DP[®] technology) is considered approximately 10,000 times more reliable than traditional RAID. For more information, refer to [NetApp Technical Report 3298, RAID-DP: NetApp Implementation of Double-Parity RAID for Data Protection](#).
- **Storage virtualization.** Data ONTAP decouples physical storage from the volumes and allows the creation of virtual volumes, called NetApp FlexVol volumes. A FlexVol volume can be resized as the application requires and can be cloned in negligible time without additional storage requirements.
- **Performance boost with Flash Cache.** NetApp Flash Cache optimizes the performance of random read-intensive workloads and delivers performance comparable to that of solid state disks, without the complexity of an additional storage tier. This intelligent read cache speeds access to data, reducing latency by a factor of 10 or more compared that of to disk drives. Faster response times can translate into higher throughput for random I/O workloads. It is not necessary to move data from tier to tier to optimize performance and cost. Active data flows automatically into Flash Cache because every volume and every LUN behind the storage controller is subject to caching.
- **Workload prioritization with FlexShare.** The most important volumes and LUNs can be given caching priority when the load is at its peak through the use of NetApp FlexShare[®] software in combination with Flash Cache cards. Flash Cache can also be tuned to match a specific workload with software settings that make it possible to cache only metadata or to cache new data when it is written to disk.
- **Performance improvement with FlexCache.** NetApp FlexCache[®] software creates a caching layer in the storage infrastructure that automatically adapts to changing usage patterns, eliminating performance bottlenecks. In addition, FlexCache automatically replicates and serves hot datasets anywhere in the infrastructure by using local caching volumes. Key benefits of FlexCache include the automatic elimination of storage bottlenecks, without tedious administration and with improved read performance in distributed application and testing environments and simplified tiered storage layers. For more details, refer to [FlexCache Software](#).
- **NetApp Unified Connect.** NetApp offers the industry's first end-to-end network convergence—NetApp Unified Connect concurrent support for FCoE and IP storage traffic (iSCSI, NFS, and CIFS) over a shared 10GbE wire, from server to storage. NetApp Unified Connect makes it possible to build a flexible and efficient shared infrastructure today as the foundation for future-ready IT.
- **Automation with Snap Creator Framework.** Every environment is different and may require significant customization and automation to meet business needs. Creating scripts or using third-party tools to automate database management functions such as backup, restoration, cloning, and replication can be expensive. In addition, implementing these functions can be time consuming. NetApp Snap Creator is an automation solution that eliminates the need to create scripts and also offers the flexibility to integrate with any application environment. This solution is fully supported by NetApp and is freely available to customers, who can download it from the software download page on the NetApp [Support](#) site.

4 ARCHITECTURE

The key feature of DB2 DPF is its ability to scale to any requirements. Scaling of the system is achieved through the addition of a new node (server) to the existing architecture, followed by the necessary configurations. The DB2 DPF distributes the data across two or more database partitions. A database partition (also called a database node) is essentially a small part of a much larger database that behaves similarly to an independent database. Each node is hosted on a separate physical server and has its own CPU and memory resources. Communication between partitions is managed through the Fast Communication Manager (FCM) configured through either a 1GB or a 10GB network switch. Each partition contains its own data, indexes, configuration files, and transaction logs and is part of the DPF cluster. The data retrieval and update requests are automatically split into subrequests and are executed in parallel among all applicable database partitions. This splitting of the database across multiple database partitions is transparent to the client application. The data is distributed on DPF databases by the creation of partition groups. A database partition group can contain either one partition or multiple database partitions. Figure 3 shows a DB2 DPF Linux setup on NetApp storage.

Figure 3) DB2 DPF Linux setup on NetApp storage.



4.1 SOFTWARE AND HARDWARE COMPONENTS

Table 1 and Table 2 describe hardware and software components used in this solution. Our test environment uses a NetApp FAS3270A storage system and two IBM System x3650 M3 servers. The DB2 instance used is `db2inst1`.

Table 1) Hardware components.

| Server Type | Quantity | Specification | Name | Purpose |
|---|----------|---|--------------------|--|
| x3650 M3 | 2 | One 6-core 3.33GHz Intel® Xeon® X5680 processor with 12MB cache and 64GB of memory (8 x 8GB DDR3 RDIMM) | DPFServer1 | Server that hosts catalog partition and four data partitions |
| | | | DPFServer2 | Server that hosts four data partitions |
| Dual-port 8Gbps Fibre Channel GbE Peripheral Component Interconnect Express (PCIe) host bus adapters (HBAs) | 2 | | | Connectivity to the SAN switches |
| GbE ports on the x3650 M3 | 4 | | | Two ports are bonded and used for the GbE internal application network. One port is used for the corporate network, and the remaining port is left available for other uses. |
| Storage system | 2 | FAS3270A | netapp01, netapp02 | Storage to host catalog partition and four data partitions |
| Disk shelves | 4 | DS4243 with 24 disks | | Disks attached to each storage controller, netapp01 and netapp02 |

Table 2) Software components.

| Software Name | Description |
|---|---|
| OS software stack | Red Hat Linux 6.0 NFS mount for the DB2 home file system |
| IBM InfoSphere Warehouse Enterprise Edition 9.7.2 | IBM DB2 9.7 Enterprise Server Edition (ESE) FixPak 2 |
| Java® SDK | IBM Java SDK 1.6 SR7 and IBM Java SDK 1.6 SR6 (32-bit) |
| NetApp software | Data ONTAP 8.0.1 operating in 7-Mode FC Host Utilities for Linux version 5.1 |

4.2 NETWORK SPECIFICATIONS

The network infrastructure is supported by GbE. A separate VLAN was created to isolate the network traffic to the DPF setup. The same GbE network is used for the FCM communication between the DB2

partitions. Depending on the network throughput requirement, the network layer can be upgraded with a 10GbE switch. Figure 4 shows the DPF network, and Table 3 lists the networks used for testing in the laboratory.

Figure 4) DPF network diagram.



Table 3) Network specifications.

| Network Name | Purpose | Implementation |
|--------------------|---|---|
| Interpartition FCM | Inter-partition communications between database partitions on different physical servers (DB2 FCM). | GbE PCIe connects to a GbE switch, link aggregated for ports on servers. NetApp recommends that this network be on a private network with dedicated switches. |
| Corporate network | Provides connectivity between each database node and the corporate network. LAN-based database backups. | GbE PCIe on server connects to the corporate network fabric. |

5 CONFIGURATION OVERVIEW AND CONSIDERATIONS

In setting up a DB2 DPF environment, many options must be considered and planned for in order to achieve optimal performance, high availability, scalability, and manageability. The database system is the primary application in the system; therefore, database configuration, underlying storage configuration, and IBM System x3650 system configuration must be incorporated into the planning and design phase of setting up the environment.

5.1 DATABASE DESIGN

SYSTEM CATALOG

For this technical report, the system catalog was created on a separate LUN on the NetApp storage volume under `datavol`. If a customer prefers to keep the system catalog partition on a separate volume, it is necessary to create a LUN under a new volume on the NetApp storage.

TRANSACTION LOGGING

Database transaction logs play a critical role in database recovery. Therefore, the logging strategy for a database must be closely aligned with the backup and restore strategy selected. IBM DB2 database uses two strategies for managing log files:

- Circular logging
- Archival logging

When circular logging is used, a finite set of log files is constantly reused. Because the size and number of log files used are configurable, storage space requirements can be calculated and/or controlled.

When archival logging is used, log files are retained after they have been populated, and they can be used to perform roll-forward recovery operations on the database. Because the number of log files used is determined by the volume of database activity, the amount of storage needed for archival logging cannot be predetermined. For OLTP environments, NetApp and IBM strongly recommend enabling archive logging.

Best Practice

For an archive logging-enabled database, transaction logs must be placed on a volume separate from the data and system catalog volume. NetApp and IBM also recommend placing archive logs on their own separate volume.

BUFFER POOL DESIGN

NetApp recommends minimizing the number of buffer pools in a data warehouse, which simplifies tuning the finite amount of buffer pool memory on a node. If an environment is OLTP or the environment must support multiple buffer pools, it is necessary to determine which buffer pools require more pages and to make sure that each buffer pool is performing to its maximum efficiency.

TABLE SPACE

IBM DB2 table space is a logical layer that is used to organize and place database object data (table, index, and so on) on physical storage disks. The number of table spaces needed depends on the size and performance requirements of the warehouse. Table 4 describes some important guidelines for database object placement.

Table 4) Table space considerations.

| Object Type | Table Space Recommendation |
|----------------------------|--|
| Large tables | Place in a separate table space across the data nodes. |
| Medium-sized tables | Logically group five or more sets of tables and give them their own table space. These table spaces are placed across the data nodes. |
| Medium-sized table indexes | Place within the data table space or separate into another shared table space for indexes. |
| Small tables | Place many tables into one single-partition table space on an administration node. |
| Small table indexes | Place into the same single-partition data table space on an administration node. |
| Replicated tables | These tables are used for joins with tables in multiple partitions and are replicated across the partition. NetApp and IBM recommend placing these tables in a separate table space. |

| Object Type | Table Space Recommendation |
|---------------------------|---|
| Materialized query tables | Place in separate table spaces that are placed across the data nodes. |
| Large table index | Place in a separate, dedicated table space. |
| Range-partitioned tables | In large tables, use one table space per range. |

DATABASE PARAMETERS

IBM DB2 offers a large number of registry, database manager, and database parameters that allow DBAs to tweak the environment based on business requirements. Performance is a key requirement for all database environments.

| Best Practice |
|--|
| Based on customer experience and laboratory tests, NetApp and IBM recommend changing the default setting for some performance-related parameters. Although the recommended parameter settings are suitable for most of the large database environments, it is important to monitor the database regularly so that the recommended setting can be adjusted in response to workload and application changes. |

Table 5, Table 6, and Table 7 describe the recommended settings for DB2 registry, database manager, and database configuration parameters. The DB2 registry variables can be updated using the `db2set` commands with the recommended values from Table 5. The database and database manager configuration can be updated using the `update DB/DBM cfg` command with the recommended values from Table 6 and Table 7, respectively.

Table 5) DB2 registry variable settings.

| Parameter Name | Default Value | Recommended Value | Purpose |
|---------------------------|---------------|-------------------|---|
| DB2_EXTENDED_OPTIMIZATION | OFF | ON | Specifies whether the query optimizer will use optimization extensions to improve query performance. |
| DB2_ANTIJOIN | NO | EXTEND | Causes the optimizer to search for opportunities to transform both NOT IN and NOT EXISTS subqueries into antijoins. |
| DB2COMM | | TCPIP | Sets communication protocols for the current DB2 instance to TCP/IP. |
| DB2_PARALLEL_IO | | *:5 | Indicates that all table spaces have parallel I/O set to ON. |
| DB2RSHCMD | | /usr/bin/ssh | Uses the ssh remote shell program for all remote commands instead of the default rsh remote shell program. |

Table 6) Database manager configuration parameters.

| Parameter Name | Default Value | Recommended Value | Purpose |
|-----------------|---------------|------------------------------|---|
| CPUSPEED | -1 | 2.36e-07 | Specifies the CPU speed, in milliseconds per instruction. Used by the query optimizer to estimate the cost of performing certain operations. |
| COMM_BANDWIDTH | -1 | 100 | Specifies the value for the communications bandwidth, in MB/sec. Used by the optimizer to estimate the cost of performing certain operations. |
| NUMDB | 8 | 1 | Specifies the number of local databases that can be active concurrently. |
| DIAGPATH | | /db2home/bculin x/db2dump | Provides an alternate path for diagnostic files. The default path is on the NFS-mounted db2home instance directory. |
| SHEAPTHRES | 0 | 20000 | |
| FCM_NUM_BUFFERS | AUTOMATIC | 131072 AUTOMATIC | Specifies memory allocation for FCM. FCM is used for efficient communication between database partitions. |
| HEALTH_MON | ON | OFF | Specifies health monitor status. We turned this off in our test. When this switch is turned off, the regular database operation is not affected by automatic stats updates and other monitoring jobs. |

Table 7) Database configuration parameters.

| Parameter Name | Default Value | Recommended Value | Purpose |
|----------------|---------------|-------------------|--|
| LOCKLIST | AUTOMATIC | 16384 | Defines the amount of storage allocated for the lock list. |
| MAXLOCKS | AUTOMATIC | 10 | Defines the percentage of the lock list held by an application that must be filled before the DB manager performs lock escalation. |
| PCKCACHESZ | AUTOMATIC | -1 | Specifies the memory allocation for caching. |
| SORTHEAP | AUTOMATIC | 12000 | Defines the maximum number of pages to be used for sorts. |
| LOGBUFSZ | 8 | 2048 | Specifies the amount of the database heap to use for buffering for log records. |
| UTIL_HEAP_SZ | 5000 | 65536 | Specifies the amount of memory that can be used by utilities such as backup, restore, and load. |
| STMTHEAP | AUTOMATIC | 10000 | Specifies the size of the statement heap. |
| LOGFILSIZ | 1000 | 12800 | Defines the size of the primary and secondary log files. |
| LOGPRIMARY | 3 | 50 | Defines the number of primary log files. |

| Parameter Name | Default Value | Recommended Value | Purpose |
|-----------------|---------------|-----------------------|---|
| LOGSECOND | 2 | 0 | Defines the number of secondary log files. |
| NEWLOGPATH | | | Allows changing the location of log files. |
| MIRRORLOGPATH | | | Specifies the location for the mirror log path. |
| NUM_IOSEEVERS | AUTOMATIC | 12 | Specifies the number of I/O servers for the DB. |
| DFT_PREFETCH_SZ | AUTOMATIC | 384 | Determines the default prefetch size of table spaces. |
| CHNGPGS_THRESH | 60 | 60; 30 if SSD is used | Specifies the level (%) of changed pages at which the asynchronous page cleaners are started. |
| WLM_COLLECT_INT | 0 | 20 | Specifies the collect and reset interval, in minutes, for workload management. |

DB2 MEMORY CONFIGURATION

Two main areas of memory are used in DB2 environments:

- Linux Kernel and file system caching
- DB2 memory, which includes DB/DBM configuration parameters and FCM buffers

In a large DPF environment, one objective is to minimize file system caching of data that the database manager is expected to cache in DB2 buffer pools. This objective is supported by using the direct I/O (DIO) feature of the kernel on `ext3` to remove file system caching for DB2 regular and temporary table space containers. DIO is enabled by default when new table spaces are created. This new default behavior means that it is no longer necessary to specify the `NO FILE SYSTEM CACHING` clause when creating a table space. Other common file system activities, such as `scp`, `ftp`, `cp`, and `mv`, can cause the Linux file-system cache to grow at the expense of the DB2 instance. The file-system cache-related configuration parameters can be found in the `/proc/sys/vm` file.

Table 8 shows the recommended memory allocation for each node based on the server configuration described in this document.

Table 8) Memory allocation on administrative node and data node.

| Total Memory | Linux Kernel and DB2 Functions (15%) | Instance and 4 Partitions on Each Node (70%) | Agent Memory and Memory for Additional Software (15%) |
|--------------|--------------------------------------|--|---|
| 64GB | 10GB | 44GB | 10GB |

5.2 STORAGE SYSTEM CONFIGURATION

Storage design for a DB2 DPF environment is based on the principle that every data module has a set of dedicated physical disks for data storage, and that all database objects in the database are distributed equally across these physical disks. This modular storage design means that all database objects within each data node have equal access to I/O transaction performance available within each data node.

NetApp FAS storage systems are designed to meet scalability, high-availability, and flexibility requirements for demanding workload environments. IBM DB2 is thoroughly tested and verified on NetApp FAS systems in IBM and NetApp laboratories. The NetApp RAID-DP architecture offers a high degree of resilience and also leverages a large spindle count to deliver better performance. NetApp

laboratory tests have proven that large aggregates deliver better performance in comparison with multiple smaller aggregates on the same number of physical storage disks.

The storage layout for the test environment was designed considering resilience, workload balance, and performance. We created a single large aggregate on each storage controller. To balance the workload, we created FlexVol volumes and LUNs for each database partition on both controllers of the storage system cluster. At the server level, we created volume groups using LUNs from each controller; that is, each volume group was spread across both controllers.

The laboratory environment was targeted to meet a 2GB/sec throughput requirement for a 3 to 5TB-size warehouse. Table 9 summarizes the technical specifications of the storage controllers used in our test laboratory.

Table 9) FAS3270 storage controller specifications.

| Components | Version |
|------------------------------|---------------------------------------|
| RAM | 16GB |
| On-board 4Gb FC | 2 |
| On-board GbE | 2 |
| Storage networking supported | FC, FCoE, iSCSI, NFS, CIFS, FTP, HTTP |
| OS versions | Data ONTAP 8 |

For detailed technical specifications for FAS3270, refer to the [NetApp Data Storage Systems](#) Web site.

DB2 PHYSICAL NODE

Table 10 and Table 11 describe FlexVol volumes, volume groups, and file system settings for the tested configuration on the DB2 physical nodes. Each physical DB2 partition was installed on an IBM x3650 M3 with 6 cores and 64GB RAM. For storage, both of the physical nodes were connected to a FAS3270 active-active HA pair, and each storage controller was attached to four DS4243 disk shelves. Each disk shelf had 24 300GB 15K RPM disks. To take advantage of the large spindle count, we created a large 64-bit aggregate on each controller with 88 disks and RAID group size 22 disks using RAID-DP.

Table 10) Physical node-1 specifications.

| Storage System | Aggregate Name | Volume Name | LUN | Volume Group | Logical Volume | File System |
|----------------|----------------|-------------|------------------------|--------------|-----------------------|--------------------|
| netapp01 | db2aggr | admvol | /vol/admvol/db2home | db2homevg | /dev/homevg/db2home1v | /db2home/ |
| netapp02 | db2aggr | admvol | /vol/admvol/stage | stagevg | /dev/stagevg/stage | /stage |
| netapp01 | db2aggr | datavol | /vol/datavol/db2sys | db2vg | /dev/db2vg/db21v1 | /db2data |
| netapp01 | db2aggr | datavol | /vol/datavol/node01un1 | datavg | /dev/datavg/datan01v1 | /db2data/NO DE0000 |
| netapp02 | db2aggr | datavol | /vol/datavol/node01un1 | datavg | | |
| netapp01 | db2aggr | datavol | /vol/datavol/node11un1 | datavg | /dev/datavg/datan11v1 | /db2data/NO DE0001 |

| Storage System | Aggregate Name | Volume Name | LUN | Volume Group | Logical Volume | File System |
|----------------|----------------|-------------|------------------------|--------------|-----------------------|-------------------|
| netapp02 | db2aggr | datavol | /vol/datavol/node1lun1 | datavg | | |
| netapp01 | db2aggr | datavol | /vol/datavol/node2lun1 | datavg | /dev/datavg/datan2lv1 | /db2data/NODE0002 |
| netapp02 | db2aggr | datavol | /vol/datavol/node2lun1 | datavg | | |
| netapp01 | db2aggr | datavol | /vol/datavol/node3lun1 | datavg | /dev/datavg/datan3lv1 | /db2data/NODE0003 |
| netapp02 | db2aggr | datavol | /vol/datavol/node3lun1 | datavg | | |
| netapp01 | db2aggr | datavol | /vol/datavol/node4lun1 | datavg | /dev/datavg/datan4lv1 | /db2data/NODE0004 |
| netapp02 | db2aggr | datavol | /vol/datavol/node4lun1 | datavg | | |
| netapp01 | db2aggr | logvol | /vol/logvol/node0lun1 | logvg | /dev/logvg/logn0lv1 | /db2log/NODE0000 |
| netapp02 | db2aggr | logvol | /vol/logvol/node0lun1 | logvg | | |
| netapp01 | db2aggr | logvol | /vol/logvol/node1lun1 | logvg | /dev/logvg/logn1lv1 | /db2log/NODE0001 |
| netapp02 | db2aggr | logvol | /vol/logvol/node1lun1 | logvg | | |
| netapp01 | db2aggr | logvol | /vol/logvol/node2lun1 | logvg | /dev/logvg/logn2lv1 | /db2log/NODE0002 |
| netapp02 | db2aggr | logvol | /vol/logvol/node2lun1 | logvg | | |
| netapp01 | db2aggr | logvol | /vol/logvol/node3lun1 | logvg | /dev/logvg/logn3lv1 | /db2log/NODE0003 |
| netapp02 | db2aggr | logvol | /vol/logvol/node3lun1 | logvg | | |
| netapp01 | db2aggr | logvol | /vol/logvol/node4lun1 | logvg | /dev/logvg/logn4lv1 | /db2log/NODE0004 |
| netapp02 | db2aggr | logvol | /vol/logvol/node4lun1 | logvg | | |

Table 11) Physical node-2 specifications.

| Storage System | Aggregate Name | Volume Name | LUN | Volume Group | Logical Volume | File System |
|----------------|----------------|-------------|-----------------------------|--------------|---------------------------|-------------------|
| netapp02 | db2aggr | datavol | /vol/datavol/ /db2sys | db2vg | /dev/db2vg/ db21v1 | /db2data |
| netapp02 | db2aggr | datavol | /vol/datavol/ /db2sys | db2vg | /dev/db2vg/ db21v1 | /db2data |
| netapp02 | db2aggr | datavol | /vol/datavol/ /node51un1 | datavg | | |
| netapp01 | db2aggr | datavol | /vol/datavol/ /node61un1 | datavg | /dev/datavg/ datan61v1 | /db2data/NODE0006 |
| netapp02 | db2aggr | datavol | /vol/datavol/ /node61un1 | datavg | | |
| netapp01 | db2aggr | datavol | /vol/datavol/ /node71un1 | datavg | /dev/datavg/ datan71v1 | /db2data/NODE0007 |
| netapp02 | db2aggr | datavol | /vol/datavol/ /node71un1 | datavg | | |
| netapp01 | db2aggr | datavol | /vol/datavol/ /node81un1 | datavg | /dev/datavg/ datan81v1 | /db2data/NODE0008 |
| netapp02 | db2aggr | datavol | /vol/datavol/ /node81un1 | datavg | | |
| netapp01 | db2aggr | logvol | /vol/logvol/ node51un1 | logvg | /dev/logvg/ logn51v1 | /db2log/NODE0005 |
| netapp02 | db2aggr | logvol | /vol/logvol/ node51un1 | logvg | | |
| netapp01 | db2aggr | logvol | /vol/logvol/ node61un1 | logvg | /dev/logvg/ logn61v1 | /db2log/NODE0006 |
| netapp02 | db2aggr | logvol | /vol/logvol/ node61un1 | logvg | | |
| netapp01 | db2aggr | logvol | /vol/logvol/ node71un1 | logvg | /dev/logvg/ logn71v1 | /db2log/NODE0007 |
| netapp02 | db2aggr | logvol | /vol/logvol/ node71un1 | logvg | | |
| netapp01 | db2aggr | logvol | /vol/logvol/ node81un1 | logvg | /dev/logvg/ logn81v1 | /db2log/NODE0008 |
| netapp02 | db2aggr | logvol | /vol/logvol/ node81un1 | logvg | | |

To organize data, file systems were created for db2home, data, and logs. Table 12 describes the purpose of file systems on the DB2 physical nodes.

Table 12) File system name and purpose on the DB2 physical nodes.

| File System Name | Purpose | NFS Shared Across All Nodes |
|------------------|--|-----------------------------|
| /db2home | Used for the DB2 instance home directory | Yes |
| /home | Local file system on each node | No |
| /stage | Used for staging tables, flat files, and other purposes | Yes |
| /db2data/ | Used for storing all permanent table space containers for catalog partition or node 0 | No |
| /db2data/N* | Used for storing all permanent table space containers for data partitions; also, some of the system's temporary table spaces can be stored under /db2data/NODE0000 | No |
| /db2logs/ | Holds the primary logs for the catalog partition or node 0 | No |

The storage layouts described in Table 10 and Table 11 were designed considering Snapshot copy–based backup and recovery. This layout allows customers to back up their data warehouse database by using Snapshot technology, and it also leverages Snapshot technology for replication and cloning.

SECURITY AND ACCESS

Security and access settings are designed according to the security policies of each organization. We used the simple approach and created DB2 user accounts on the NetApp storage system. With this approach, users can be added through NetApp System Manager or the command line interface (CLI) on the NetApp storage system. System Manager can be downloaded from the NetApp [Support](#) (formerly NOW) site. For more information on how to use this tool, refer to the appropriate software documentation page, which is also available on the documentation page on the NetApp [Support](#) site.

In addition to adding users, it is necessary to apply licenses and configure rsh/ssh access. For more information, refer to [NetApp Technical Report 3272, IBM DB2 on NetApp: Deployment and Best Practices](#).

5.3 VOLUME CONSIDERATION

The basic logical data container in DB2 for table data is the table space, which defines the physical relationship of the database objects and the storage on which they reside. A table space consists of one or more containers. A container can be a directory, a file, or a raw device. If a single table space spans more than one container, the DB2 database manager uses parallelism to balance the data load across all containers.

NetApp FlexVol volumes make it easy to manage the placement of database objects on storage system volumes. If table space containers are placed in a FlexVol volume, a single Snapshot copy of the storage system volume can be used to back up multiple DB2 objects. For more information on creating and managing FlexVol volumes, refer to the `vol` command in the Data ONTAP Commands manual, which is available on the NetApp [Support](#) site.

When placing a DB2 database on NetApp storage, consider these important design principles:

- The root volume should be its own volume. The root volume usually contains data that does not change much over time; therefore, it does not require frequent backups. If a data volume fails, having a separate, still-functioning root volume saves valuable time in the recovery process.
- NetApp and IBM strongly recommend storing all database object files on a separate volume on the NetApp storage system that is not the root volume.

- If NetApp SnapMirror[®] software is used in the backup-and-recovery scenario, it is important to note that SnapMirror is available only to nonroot volumes.
- The transaction logs should be kept on a separate storage system volume from the database objects, not on the root volume. This practice decreases the loss of data after a database failure because of the availability of more current logs. If the transaction logs and the database resided on the same volume, the recovered volume would contain the active logs at the time the Snapshot copy was taken. No log data later than the Snapshot copy would be available, causing a loss of data because of inadequate logs for roll-forward operations.

6 INSTALLATION AND CONFIGURATION

Before beginning to install DB2, make sure that the servers on which each of the components will be installed have the required hardware and software and that the user authorization is correct for installing the product. The installation requirements for data server components, application server components, and client server components are described on the [Installing InfoSphere Warehouse](#) Web site.

6.1 INSTALLING DB2

DB2 ADMINISTRATOR AND USER ACCOUNTS

If DB2 is already installed on the UNIX[®] server, it is possible that the appropriate DB2 user accounts also exist. In that case, the home directories for these accounts should be moved to a storage system volume so that all remaining servers can access them if one server goes down. If DB2 is being installed for the first time on the Linux server, a DB2 administrator and one or more user accounts can be created as part of the DB2 installation process. For instructions on setting up the appropriate accounts during the installation process, refer to the [IBM DB2 documentation](#).

INSTALLATION METHODS

IBM DB2 offers multiple installation methods, including setup wizard, response file, CLI (`db2_install`), and payload file deployment. The DB2 Setup wizard is an easy-to-use interface for installing DB2 products and for performing initial setup and configuration tasks. This method also creates DB2 instances and response files that can be used to duplicate the installation on other machines. For the test environment described in this document, we used the `db2_install` command line installation method option. Although the `db2_install` command installs all components for the DB2 product specified, it does not perform user and group creation, instance creation, or configuration. The instance and database are created and configured after the DB2 software is installed.

By default, the installation location is `/opt/ibm/db2/V9.7` for Linux platforms. The default location can be changed by specifying a fully qualified path.

6.2 CREATING A DB2 INSTANCE

Creating a DB2 instance in a NetApp storage system environment is a straightforward process that uses standard DB2 commands. This section describes the steps to create a new DB2 instance in a NetApp storage environment.

1. Log in to the administration node that acts as the NFS server for the DB2 instance home directory and create an instance using the `db2icrt` command:

```
/opt/ibm/db2/V9.5/instance/db2icrt -u fenced_id instance_name
```

On the Linux platform, the instance name must be the same as the name of the user owning the instance. For example, to create an instance named `db2inst1`, run the following command:

```
/opt/ibm/db2/V9.7/instance/db2icrt -u db2inst1 db2inst1
```

Make sure the NFS home directory has been set up to allow root to write to the home directory, at least for instance creation.

2. Update the database manager configuration parameter using the `update dbm` configuration command. This command confirms the communication of the client application with the DB2 database server. For example, to update the `svcename` parameter using the value listed in the `/etc/services` file, run the following command:

```
db2 update dbm config using svcename db2inst1
```

Where:

- Instance name is `db2inst1`.
 - Service name is `db2inst1`.
3. The fault monitor daemon that is controlled by the DB2 Fault Monitor Controller Utility (FMCU) is not required for IBM InfoSphere Warehouse. NetApp recommends turning it off. To stop the fault monitor and prevent the DB2 FMCU from launching on any node, run the following command:

```
db2fm -i db2inst1 -D  
db2fmcu -d
```

The `db2fmcu` command must be run as `root` because it accesses the system's `inittab` file. Run the `db2fmcu` command on all servers.

If a DB2 FixPak is applied, the `inittab` file is again configured to include the fault monitor coordinator. To prevent the fault monitor coordinator from being launched after a FixPak has been applied, reissue the `db2fmcu -d` command as described in step 3.

6.3 UPDATING THE DB2NODE.CONFIG FILE

The database partition and server mapping is managed in the `db2nodes.cfg` file, which is found in the instance home directory (`~/sql1lib/db2nodes.cfg`). Table 13 describes the design for database partition numbering for the instance named `db2inst1`.

Table 13) Database partitions.

| Partition Number | Description | Location |
|-------------------------|--|------------|
| Partition 0 | Functions as catalog partition, coordinator, and single-partition data | DPFServer1 |
| Partition 1–Partition 4 | Database partitions with partitioned data | DPFServer1 |
| Partition 5–Partition 8 | Database partitions with partitioned data | DPFServer2 |

The catalog partition is assigned number 0 because there is only one catalog for the database. The database partitions that contain the vast majority of data are the data balanced partition units (BPUs). Because most database administration and performance tuning involves the data BPUs, the numbering of these partitions starts at 1 and continues sequentially.

The database partition and server mapping specified in the `db2nodes.cfg` file has the following format:

```
nodenum hostname logical port netname
```

Where:

- `Nodenum` is a unique number between 0 and 999 that identifies a database partition in a partitioned database system. For additional database partition servers, the `nodenum` values must be in ascending order.

- `Hostname` is the TCP/IP host name of the database partition server for use by database partitions for non-FCM communication. This network is also used for FCM communication if no `netname` is specified.
- `Logical port` specifies the logical port number for the database partition. This field is used to specify each database BPU within a single server (BCU). These logical ports correspond to the ports specified in the `/etc/services` file for interpartition communications.
- `Netname` (optional) specifies the host name or the IP address of the high-speed interconnect for FCM communication.

Do not modify the `db2nodes.cfg` file while the instance is running.

For the test laboratory Warehouse environment, the configuration file was similar to the following:

```
0 DPFServer1 0
1 DPFServer1 1
2 DPFServer1 2
3 DPFServer1 3
4 DPFServer1 4
5 DPFServer2 0
6 DPFServer2 1
7 DPFServer2 2
8 DPFServer2 3
```

6.4 CREATING THE DATABASE

After the instance is created, the DB2 DPF database can be created. The database manager creates the database catalog on the database partition where the `CREATE DATABASE` statement is executed. For this reason, make sure that the `CREATE DATABASE` command is executed on the database partition that will serve as the catalog partition. This catalog partition is located on the administration node and has partition number 0 in the `db2nodes.cfg` file.

Create a DB2 database by using the `CREATE DATABASE` command and specifying the database path and database location. For example, to create a database named `bcudb` using `/db2data` for the database and `/db2home` for `db2path`, run a `CREATE DATABASE` command similar to the following:

```
CREATE DATABASE bcudb ON /db2data USING CODESET UTF-8 TERRITORY US
COLLATE USING SYSTEM_819_US PAGESIZE 16384 DFT_EXTENT_SZ 16 AUTOCONFIGURE APPLY NONE;
```

When the database is created using the preceding statement:

- Automatic storage management is enabled for the database, and the default page size for buffer pools and table spaces is 16KB.
- File systems in `/db2data/NODENNNN` are used for all automatic storage table space containers. Sufficient zeroes are added to the partition number to produce a four-digit suffix represented by `NNNN`.
- The `/db2data/` file system is used for the database directory.
- The default extent size is 16 pages, which is equivalent to 256KB when the page size is 16KB.
- The catalog partition is configured on the `DPFServer1` node.

6.5 CREATING DATABASE PARTITION GROUPS

When a database is created, the following database partition groups are created automatically:

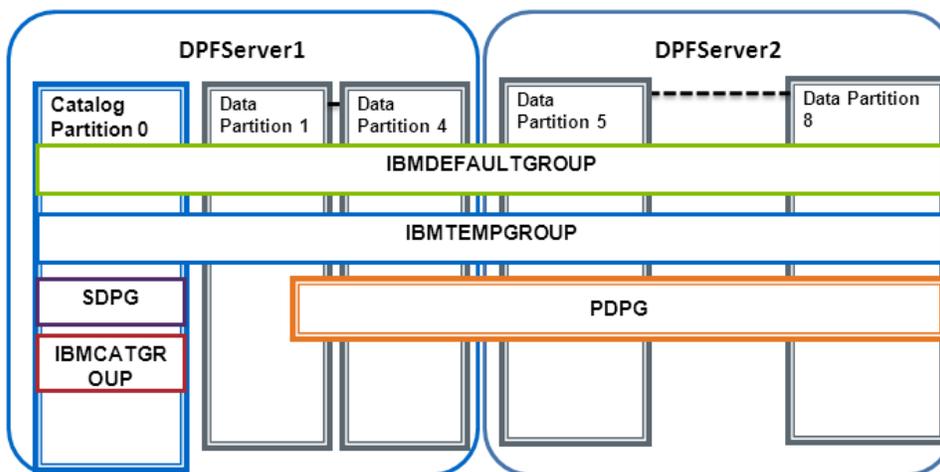
- **IBMCATGROUP.** This group consists of one database partition, database partition 0, and it contains the catalog tables.

- **IBMTEMPGROUP.** This group contains all of the database partitions. It is the database partition group in which the default TEMPSPACE1 system-managed temporary table space is created. The page size for TEMPSPACE1 is determined by the database page size, which is 16KB.
- **IBMDEFAULTGROUP.** This group contains all of the database partitions. It becomes the default partition group for table spaces that do not have explicit partition group assignments.
- **PDPG (partitioned data partition group).** This group contains only the data partition on the data nodes. It excludes the administration or catalog partition. The tables in this group are normally medium sized to quite large. For our test environment, we had two data nodes with four partitions on each. Run the following command to create the PDGP:

```
CREATE DATABASE PARTITION GROUP PDPG ON DBPARTITIONNUMS (1-8)
```

After creation of the data partition groups, the logical view of the setup resembles Figure 5.

Figure 5) Database partition group layout.



The current layout configures all of the DB2 temporary table spaces under the `datavol` on the NetApp storage where the user data resides. There is also the option of creating a separate volume on the NetApp storage to separate the DB2 temporary table spaces from the user data.

6.6 CREATING A BUFFER POOL

For this configuration, a total buffer pool size of 2.5GB per database partition was used. The first 16MB is allocated to the default buffer pool, which is used for the database catalog and for monitoring tables. The remaining buffer pool memory is allocated to a single buffer pool for both the permanent and the temporary table spaces.

Contrary to OLTP, data warehouse reads are sequential. To improve sequential read and table scan performance, IBM recommends creating a block-based buffer pool, which consists of both a block area and a page area. The block area consists of blocks, and each block contains a specified number of contiguous pages. The block area is used for sequential prefetching workloads only. The page area is used for all other I/O workloads.

When a block-based buffer pool is configured, the DB2 prefetchers perform read requests of contiguous blocks of pages on disk into contiguous pages in memory. Prefetching using block I/O is more efficient than the page-based vectored I/O that the DB2 software uses by default. The prefetchers can also increase the efficiency of buffer pool reuse by reducing buffer pool thrashing when large volumes of data are read. Use a `CREATE BUFFERPOOL` statement to create buffer pools. The following statement creates a 2.5GB buffer pool per database partition with a 1.5GB block area:

```
CREATE BUFFERPOOL bp_16k ALL DBPARTITIONNUMS SIZE 150000 PAGESIZE 16K NUMBLOCKPAGES
50000 BLOCKSIZE 16;
```

6.7 PLACING THE DATABASE TRANSACTION LOG IN THE NEW DIRECTORY

By default, DB2 database logs are initially created in a directory called `SQLLOGDIR`, a subdirectory of the database directory. To change the location where active logs and future archive logs are placed, change the value for this configuration parameter to point to a different directory or device. To update the transaction log path for the entire catalog node, run the following command:

```
db2 "UPDATE DB CFG FOR bcudb USING NEWLOGPATH /db2logs/NODE0000"
```

Repeat the preceding command for all of the partitions. The new setting does not become the value of `logpath` until both of the following conditions occur:

- The database is in a consistent state, as indicated by the `database_consistent` parameter.
- All applications are disconnected from the database.

If the database is configured for roll-forward recovery, archive logs that are currently stored in the database log path directory are not moved to the new location. The size of the `LOGPRIMARY`, `LOGSECOND`, and the `LOGFILSIZ` configuration parameters is decided based on the requirement. Because archival logging is enabled, separate volumes for data and transaction logs on storage are created to help database recovery from Snapshot backup images and subsequent point-in-time `ROLLFORWARD` recovery.

6.8 CREATING TABLE SPACES

Before table spaces are created for an environment, a table space design layout must be developed. IBM recommends using automatic storage management for all user table spaces. Use the following command to create a table space:

```
CREATE TABLESPACE <tbs_name> IN DATABASE PARTITION GROUP <pg_name> BUFFERPOOL
<bp_name> OVERHEAD 4.0 TRANSFERRATE 0.04;
```

Where:

- `tbs_name` is the table space name
- `pg_name` is the partition group name for the table space
- `bp_name` is the buffer pool name for the table space

For example, to create a table space named `ts_big` in a partition group `pdpq` that uses buffer pool `bp_16k`, run the following command:

```
CREATE TABLESPACE ts_big IN DATABASE PARTITION GROUP pdpq BUFFERPOOL bp_16k
OVERHEAD 4.0
TRANSFERRATE 0.04;
```

If the database partition group is not specified, the default database partition group (`IBMDEFAULTGROUP`) is used for `REGULAR`, `LARGE`, and `USER TEMPORARY` table spaces. For `SYSTEM TEMPORARY` table spaces, the default database partition group `IBMTMPGROUP` is used.

6.9 CREATING TABLES AND POPULATING DATA

After you have completed the steps described in sections 6.2 through 6.8, the database is ready for tables to be created and populated with data. The logical database design can be transformed to a physical database and scripts can be written to create tables, or an existing database can be reverse

engineered to create table creation scripts. After the tables are created, existing data can be imported into the environment, and the warehouse is ready to use.

7 DPF DATABASE BACKUP AND RESTORE

Normally, DB2 DPF environments tend to be large, and backing up massive amounts of data poses a big challenge for DBAs. In addition to data volumes that must be backed up, other factors such as system performance, shrinking backup windows, explosive data growth, and decreasing mean time to recover make traditional backup and recovery methods outdated and unfit. On the other hand, NetApp Snapshot technology has been proven effective to perform database backup-and-restore operations. Snapshot and SnapRestore technology are Data ONTAP features that can be used to meet database backup and recovery challenges. NetApp Snap Creator eliminates the need for scripting and automates the backup and restore process.

As described in section 3.3, NetApp Snap Creator Framework has three components:

- An engine
- A configuration file
- Application plug-ins

Snap Creator is delivered with a default configuration file. In order for the DB2 database to be configured for backup and recovery using NetApp Snap Creator, the configuration file must be customized for the DB2 environment. The following section describes the changes that must be made to the configuration file settings. For more information, refer to the [NetApp Snap Creator Framework Documentation](#) Web site.

7.1 BASIC CONFIGURATION

Snap Creator uses a configuration file to perform its operations. The purpose of each parameter is documented in the documentation section of the configuration file. Table 14 describes the recommended settings for Snap Creator parameters.

Table 14) Snap Creator basic configuration.

| Parameter Name | Purpose | Required | Test Environment Setting |
|---------------------|--|----------|---|
| SNAME | Assigns a meaningful name to a Snapshot copy. It prefixes the supplied value to the Snapshot name. NetApp and IBM recommend setting this parameter to the same as the database name. | Yes | bcudb |
| SNAP_TIMESTAMP_ONLY | Appends the timestamp (YYYYMMDDHHMMSS) to the Snapshot name. | Yes | Y |
| VOLUMES | Specifies the names of the values used for database storage. Multiple volumes are allowed. The format used for this parameter is controller1:volume1, volume2; controller2:volume1. | Yes | netapp01:datavol,logvol;netapp02:datavol,logvol |
| NTAP_USERS | Specifies user credentials to perform Snapshot operation on the NetApp storage. | Yes | netapp01:db2inst1/db2inst1;netapp02:db2inst1/db2inst1 |
| TRANSPORT | Specifies the use of HTTP or HTTPS protocol for API communications. | Yes | HTTP |

| Parameter Name | Purpose | Required | Test Environment Setting |
|-----------------------------|--|----------|-----------------------------|
| PORT | Specifies the port used to communicate with the NetApp storage. Normally, the value is 80 or 443. | Yes | 80 |
| NTAP_SNAPSHOT_RETENTIONS | Specifies the number of Snapshot copies to retain for a given policy. | Yes | daily:7,weekly:4,monthly:1. |
| NTAP_SNAPSHOT_RETENTION_AGE | Specifies a retention age (in days) for Snapshot copies. When configured, Snapshot copies are deleted only if they are older than the value defined by this number. NetApp recommends setting this field, especially if there is a recovery point objective (RPO) requirement in a production environment. | Yes | 7 |
| DATABASES | Specifies the database name and instance name. | Yes | BCUDB:db2inst1 |
| DB2_CMD | Specifies the CLI command. | Yes | db2 |
| APP_NAME | Specifies the call to the default plug-in. | No | |

Snap Creator uses application plug-ins for application consistency. The default plug-ins for DB2 are good for a nonpartitioned database backup and recovery. The partition database backup and recovery are handled by using a custom plug-in developed specifically for multipartition databases. This custom plug-in can be downloaded free of charge from the [NetApp Community](#) Web site.

To use the custom plug-in, make the following changes to the configuration file:

1. Download the `db2_DPF_plug-in.zip` file and copy it to the server where Snap Creator Framework is installed.
2. To perform application-consistent backup after write suspend on the database, update the following Snap Creator configuration file parameter:

```
APP_QUIESCE_CMD01=<Plug-in path>/writeSuspendDPF.sh <DBNAME>
```

The updated parameter in the test laboratory configuration file was similar to the following:

```
APP_QUIESCE_CMD01=/scServer/writeSuspendDPF.sh bcudb
```

3. Update the following configuration parameter to allow write resume using the custom plug-in:

```
APP_UNQUIESCE_CMD01=<Path to Plug-in>/writeResumeDPF.sh <DBNAME>
```

The updated parameter in the test laboratory configuration file was similar to the following:

```
APP_UNQUIESCE_CMD01=/scServer/writeResumeDPF.sh bcudb
```

Note: For a partition database, `APP_NAME` must be left blank.

The Snap Creator configuration file used for the test laboratory environment is shown in section 8.1, “Database Manager Configuration Parameters.”

7.2 DATABASE BACKUP USING SNAP CREATOR FRAMEWORK

After a configuration file is created for the database, use Snap Creator to create the Snapshot backup on the NetApp storage system. Snap Creator is invoked by using the `snap creator` command. Specify configuration (profile) and action options as `snap`.

To back up a database using Snap Creator, run the following command:

```
./snapcreator -profile <profile name> --action snap --policy <bkup policy> --verbose
```

Where:

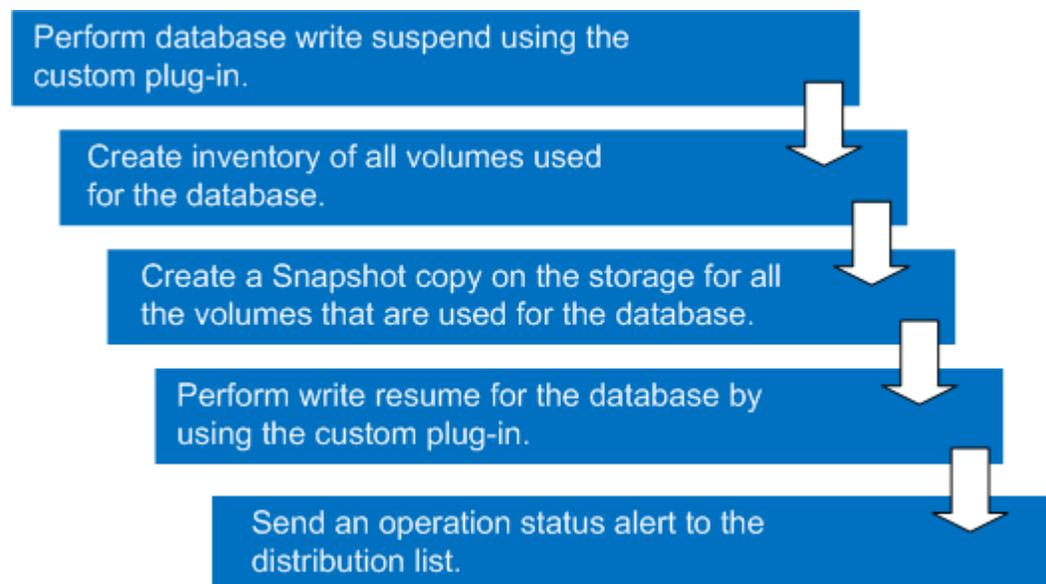
- `Profile` is the profile or configuration file name. NetApp and IBM recommend that the profile name be the same as the database name.
- `Action` is the operation being performed. Valid values are `snap`, `snap list`, `restore_alt`, `clone_vol`, and so on.
- `bkup_policy` is the backup policy type. Possible policy types are `hourly`, `daily`, `weekly`, and `monthly`.
- `Verbose` (optional) shows the Snap Creator operation on the console. Snap Creator saves the output to the logs, regardless of whether this option is specified.

The following example of a `snap creator` command creates a database backup based on a Snapshot copy of a database named BCUDB that has a backup policy of `daily`:

```
./snapcreator --profile BCUDB --action snap --policy daily --verbose
```

When the `snap creator` command is executed, Snap Creator initiates and completes background processes, as shown in Figure 6.

Figure 6) Snap Creator backup process flow.



When the Snapshot backup process is complete and the Snapshot backup has been created, use Snap Creator to list or view the backup images. To list the Snapshot-based backups for a database named BCUDB, run the following command:

```
./snapcreator --profile BCUDB --action snaplist --policy daily
```

7.3 DATABASE RECOVERY USING SNAP CREATOR

Any production database may be subjected to data corruption, either by a malicious process or by user error. To go back to a point in time before the data corruption, restore the database by using the latest available good database backup image. If the database is using NetApp storage, the restore process can

be automated by using Snap Creator. Snap Creator uses an interactive process that requires user input to restore the database from a backup image based on a Snapshot copy.

To restore a database from a Snapshot backup copy, run the following command:

```
./snapcreator --<profile name> --action restore --policy <bkup policy>
```

Where:

- `Profile` is the profile or configuration file name.
- `Action` is the operation being performed (in this case, the action is restore).
- `bkup_policy` is the backup policy type. Possible policy types are hourly, daily, weekly, and monthly.

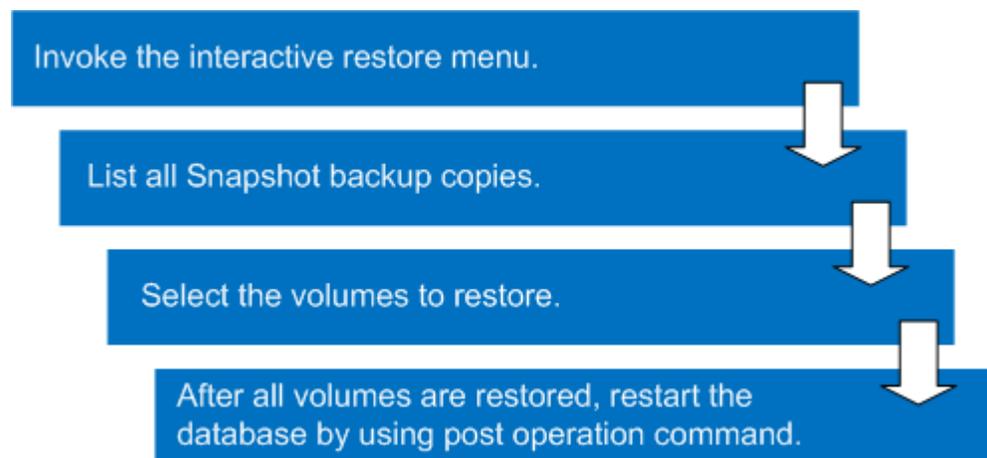
For example, to restore a database named BCUDB, run the following command:

```
./snapcreator --profile BCUDB --action restore_alt --policy daily
```

Once Snap Creator is executed with the action as restore, a snap restore operation is done on the storage system, followed by the postrestore operation on the host database server.

Figure 7 shows the background process flow.

Figure 7) Snap Creator database restore process flow.



After the restore is complete, restart the database by either using the Snap Creator postprocess operation or manually running the DB2 commands. To restart the database manually, use the following commands to initialize the database:

```
db2inidb [DBName] as mirror  
db2 restart db [DBName] write resume
```

Where:

- `DBName` identifies the name of the database to be initialized.

For example, to initialize a database named BCUDB for all of the nodes, run the following commands on the administration node database server:

```
export DB2NODE=0 && db2inidb bcudb as mirror && db2 restart db bcudb write resume &&  
db2 terminate  
export DB2NODE=1 && db2inidb bcudb as mirror && db2 restart db bcudb write resume &&  
db2 terminate  
export DB2NODE=2 && db2inidb bcudb as mirror && db2 restart db bcudb write resume &&  
db2 terminate
```

```

export DB2NODE=3 && db2inidb bcudb as mirror && db2 restart db bcudb write resume &&
db2 terminate
export DB2NODE=4 && db2inidb bcudb as mirror && db2 restart db bcudb write resume &&
db2 terminate
export DB2NODE=5 && db2inidb bcudb as mirror && db2 restart db bcudb write resume &&
db2 terminate
export DB2NODE=6 && db2inidb bcudb as mirror && db2 restart db bcudb write resume &&
db2 terminate
export DB2NODE=7 && db2inidb bcudb as mirror && db2 restart db bcudb write resume &&
db2 terminate
export DB2NODE=8 && db2inidb bcudb as mirror && db2 restart db bcudb write resume &&
db2 terminate

```

To create scripts for these commands, use a scripting language such as shell or Perl script.

8 APPENDIXES

8.1 DATABASE MANAGER CONFIGURATION PARAMETERS

The following output shows the database manager configuration from the setup that was created for this technical report.

```

Database manager configuration release level          = 0x0d00
CPU speed (millisec/instruction)                    (CPUSPEED) = 2.360000e-07
Communications bandwidth (MB/sec)                   (COMM_BANDWIDTH) = 1.000000e+02
Max number of concurrently active databases          (NUMDB) = 1
Federated Database System Support                   (FEDERATED) = NO
Transaction processor monitor name                   (TP_MON_NAME) =
Default charge-back account                         (DFT_ACCOUNT_STR) =
Java Development Kit installation path               (JDK_PATH) =
/db2home/db2inst1/sqllib/java/jdk64
Diagnostic error capture level                       (DIAGLEVEL) = 3
Notify Level                                         (NOTIFYLEVEL) = 3
Diagnostic data directory path                       (DIAGPATH) = /db2home/sqllib/db2dump
Size of rotating db2diag & notify logs (MB)         (DIAGSIZE) = 0
Buffer pool                                          (DFT_MON_BUFPOOL) = OFF
Lock                                                 (DFT_MON_LOCK) = OFF
Sort                                                 (DFT_MON_SORT) = OFF
Statement                                            (DFT_MON_STMT) = OFF
Table                                                (DFT_MON_TABLE) = OFF
Timestamp                                           (DFT_MON_TIMESTAMP) = ON
Unit of work                                         (DFT_MON_UOW) = OFF
Monitor health of instance and databases             (HEALTH_MON) = OFF
SYSADM group name                                   (SYSADM_GROUP) = BCUIGRP
SYSCTRL group name                                  (SYSCTRL_GROUP) =
SYSMAINT group name                                 (SYSMAINT_GROUP) =
SYSMON group name                                   (SYSMON_GROUP) =
Client Userid-Password Plugin                       (CLNT_PW_PLUGIN) =
Client Kerberos Plugin                              (CLNT_KRB_PLUGIN) =
Group Plugin                                         (GROUP_PLUGIN) =
GSS Plugin for Local Authorization                  (LOCAL_GSSPLUGIN) =
Server Plugin Mode                                   (SRV_PLUGIN_MODE) = UNFENCED
Server List of GSS Plugins                           (SRVCON_GSSPLUGIN_LIST) =
Server Userid-Password Plugin                       (SRVCON_PW_PLUGIN) =
Server Connection Authentication                    (SRVCON_AUTH) = NOT_SPECIFIED
Cluster manager                                     (CLUSTER_MGR) = TSA
Database manager authentication                     (AUTHENTICATION) = SERVER
Alternate authentication                            (ALTERNATE_AUTH_ENC) = NOT_SPECIFIED
Cataloging allowed without authority                 (CATALOG_NOAUTH) = NO
Trust all clients                                    (TRUST_ALLCLNTS) = YES

```

| | | |
|---|--------------------|------------------------------|
| Trusted client authentication | (TRUST_CLNTAUTH) | = CLIENT |
| Bypass federated authentication | (FED_NOAUTH) | = NO |
| Default database path | (DFTDBPATH) | = /db2data/NODE0000 |
| Database monitor heap size (4KB) | (MON_HEAP_SZ) | = AUTOMATIC(90) |
| Java Virtual Machine heap size (4KB) | (JAVA_HEAP_SZ) | = 2048 |
| Audit buffer size (4KB) | (AUDIT_BUF_SZ) | = 0 |
| Size of instance shared memory (4KB) | (INSTANCE_MEMORY) | = AUTOMATIC(14838775) |
| Agent stack size | (AGENT_STACK_SZ) | = 1024 |
| Sort heap threshold (4KB) | (SHEAPTHRES) | = 600000 |
| Directory cache support | (DIR_CACHE) | = YES |
| Application support layer heap size (4KB) | (ASLHEAPSZ) | = 15 |
| Max requester I/O block size (bytes) | (RQRIOBLK) | = 32767 |
| Workload impact by throttled utilities | (UTIL_IMPACT_LIM) | = 10 |
| Priority of agents | (AGENTPRI) | = SYSTEM |
| Agent pool size | (NUM_POOLAGENTS) | = AUTOMATIC(100) |
| Initial number of agents in pool | (NUM_INITAGENTS) | = 0 |
| Max number of coordinating agents | (MAX_COORDAGENTS) | = AUTOMATIC(200) |
| Max number of client connections | (MAX_CONNECTIONS) | = AUTOMATIC(MAX_COORDAGENTS) |
| Keep fenced process | (KEEPFENCED) | = YES |
| Number of pooled fenced processes | (FENCED_POOL) | = ATOMATIC(MAX_COORDAGENTS) |
| Initial number of fenced processes | (NUM_INITFENCED) | = 0 |
| Index re-creation time and redo index build | (INDEXREC) | = RESTART |
| Transaction manager database name | (TM_DATABASE) | = 1ST_CONN |
| Transaction resync interval (sec) | (RESYNC_INTERVAL) | = 180 |
| SPM name | (SPM_NAME) | = |
| SPM log size | (SPM_LOG_FILE_SZ) | = 256 |
| SPM resync agent limit | (SPM_MAX_RESYNC) | = 20 |
| SPM log path | (SPM_LOG_PATH) | = |
| TCP/IP Service name | (SVCENAME) | = db2inst1 |
| Discovery mode | (DISCOVER) | = SEARCH |
| Discover server instance | (DISCOVER_INST) | = ENABLE |
| SSL server keydb file | (SSL_SVR_KEYDB) | = |
| SSL server stash file | (SSL_SVR_STASH) | = |
| SSL server certificate label | (SSL_SVR_LABEL) | = |
| SSL service name | (SSL_SVCENAME) | = |
| SSL cipher specs | (SSL_CIPHERSPECS) | = |
| SSL versions | (SSL_VERSIONS) | = |
| SSL client keydb file | (SSL_CLNT_KEYDB) | = |
| SSL client stash file | (SSL_CLNT_STASH) | = |
| Maximum query degree of parallelism | (MAX_QUERYDEGREE) | = ANY |
| Enable intra-partition parallelism | (INTRA_PARALLEL) | = NO |
| Maximum Asynchronous TQs per query | (FEDERATED_ASYNC) | = 0 |
| No. of int. communication buffers(4KB) | (FCM_NUM_BUFFERS) | = AUTOMATIC(131072) |
| No. of int. communication channels | (FCM_NUM_CHANNELS) | = AUTOMATIC(2048) |
| Node connection elapse time (sec) | (CONN_ELAPSE) | = 10 |
| Max number of node connection retries | (MAX_CONNRETRIES) | = 5 |
| Max time difference between nodes(min) | (MAX_TIME_DIFF) | = 60 |
| db2start/db2stop timeout (min) | (START_STOP_TIME) | = 10 |

8.2 DATABASE CONFIGURATION PARAMETER SETTINGS

The following output shows the database configuration from the setup that was created for this technical report.

| | |
|--------------------------------------|------------|
| Database configuration release level | = 0x0d00 |
| Database release level | = 0x0d00 |
| Database territory | = US |
| Database code page | = 1208 |
| Database code set | = UTF-8 |
| Database country/region code | = 1 |
| Database collating sequence | = IDENTITY |

```

Alternate collating sequence          (ALT_COLLATE) =
Number compatibility                  = OFF
Varchar2 compatibiliy                = OFF
Date compatibility                    = OFF
Database page size                    = 16384
Dynamic SQL Query management          (DYN_QUERY_MGMT) = DISABLE
Statement concentrator                (STMT_CONC) = OFF
Discovery support for this database    (DISCOVER_DB) = ENABLE
Restrict access                       = NO
Default query optimization class      (DFT_QUERYOPT) = 5
Degree of parallelism                 (DFT_DEGREE) = 1
Continue upon arithmetic exceptions   (DFT_SQLMATHWARN) = NO
Default refresh age                   (DFT_REFRESH_AGE) = 0
Default maintained table types for opt (DFT_MTTB_TYPES) = SYSTEM
Number of frequent values retained    (NUM_FREQVALUES) = 10
Number of quantiles retained          (NUM_QUANTILES) = 20
Decimal floating point rounding mode (DECFLT_ROUNDING) = ROUND_HALF_EVEN
Backup pending                        = NO
All committed transactions have been written to disk = YES
Rollforward pending                   = NO
Restore pending                       = NO
Multi-page file allocation enabled    = YES
Log retain for recovery status        = NO
User exit for logging status          = NO
Self tuning memory (SELF_TUNING_MEM) = OFF
Size of database shared memory (4KB) (DATABASE_MEMORY) = AUTOMATIC (1397264)
Database memory threshold             (DB_MEM_THRESH) = 10
Max storage for lock list (4KB)       (LOCKLIST) = 16384
Percent. of lock lists per application (MAXLOCKS) = 10
Package cache size (4KB)              (PCKCACHESZ) = (MAXAPPLS*8)
Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 5000
Sort list heap (4KB)                  (SORTHEAP) = 12000
Database heap (4KB)                   (DBHEAP) = AUTOMATIC(1200)
Catalog cache size (4KB)              (CATALOGCACHE_SZ) = (MAXAPPLS*5)
Log buffer size (4KB)                 (LOGBUFSZ) = 2048
Utilities heap size (4KB)             (UTIL_HEAP_SZ) = 65536
Buffer pool size (pages)              (BUFFPAGE) = 1000
SQL statement heap (4KB)              (STMTHEAP) = 10000
Default application heap               (4KB) (APPLHEAPSZ) = AUTOMATIC(128)
Application Memory Size (4KB)         (APPL_MEMORY) = AUTOMATIC(40016)
Statistics heap size (4KB)            (STAT_HEAP_SZ) = AUTOMATIC(4384)
Interval for checking deadlock (ms)    (DLCHKTIME) = 10000
Lock timeout (sec)                    (LOCKTIMEOUT) = -1
Changed pages threshold                (CHNGPGS_THRESH) = 60
Number of asynchronous page cleaners    (NUM_IOCLEANERS) = AUTOMATIC
Number of I/O servers                  (NUM_IOSERVERS) = AUTOMATIC
Index sort flag                        (INDEXSORT) = YES
Sequential detect flag                 (SEQDETECT) = YES
Default prefetch size (pages)          (DFT_PREFETCH_SZ) = AUTOMATIC
Track modified pages                   (TRACKMOD) = NO
Default number of containers           = 1
Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 16
Max number of active applications      (MAXAPPLS) = AUTOMATIC(1038)
Average number of active applications (AVG_APPLS) = AUTOMATIC(1)
Max DB files open per application      (MAXFILOP) = 61440
Log file size (4KB)                   (LOGFILSIZ) = 12800
Number of primary log files            (LOGPRIMARY) = 50
Number of secondary log files          (LOGSECOND) = 0
Changed path to log files              (NEWLOGPATH) =
Path to log files                      = /db2logs/NODE0000/
Overflow log path                      (OVERFLOWLOGPATH) =
Mirror log path                        (MIRRORLOGPATH) =
First active log file                  =

```

```

Block log on disk full (BLK_LOG_DSK_FUL) = NO
Block non logged operations (BLOCKNONLOGGED) = NO
Percent max primary log space by transaction (MAX_LOG) = 0
Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0
Group commit count (MINCOMMIT) = 1
Percent log file reclaimed before soft chkpt(SOFTMAX) = 100
Log retain for recovery enabled (LOGRETAIN) = OFF
User exit for logging enabled (USEREXIT) = OFF
HADR database role = STANDARD
HADR local host name (HADR_LOCAL_HOST) =
HADR local service name (HADR_LOCAL_SVC) =
HADR remote host name (HADR_REMOTE_HOST) =
HADR remote service name (HADR_REMOTE_SVC) =
HADR instance name of remote server (HADR_REMOTE_INST) =
HADR timeout value (HADR_TIMEOUT) = 120
HADR log write synchronization mode (HADR_SYNCMODE) = NEARSYNC
HADR peer window duration (seconds) (HADR_PEER_WINDOW) = 0
First log archive method (LOGARCHMETH1) = OFF
Options for logarchmeth1 (LOGARCHOPT1) =
Second log archive method (LOGARCHMETH2) = OFF
Options for logarchmeth2 (LOGARCHOPT2) =
Failover log archive path (FAILARCHPATH) =
Number of log archive retries on error (NUMARCHRETRY) = 5
Log archive retry Delay (secs) (ARCHRETRYDELAY) = 20
Vendor options (VENDOROPT) =
Auto restart enabled (AUTORESTART) = ON
Index re-creation time and redo index build (INDEXREC) = SYSTEM
Log pages during index build (LOGINDEXBUILD) = OFF
Default number of loadrec sessions (DFT_LOADREC_SES) = 1
Number of database backups to retain (NUM_DB_BACKUPS) = 12
Recovery history retention (days) (REC_HIS_RETENTN) = 366
Auto deletion of recovery objects (AUTO_DEL_REC_OBJ) = OFF
TSM management class (TSM_MGMTCLASS) =
TSM node name (TSM_NODENAME) =
TSM owner (TSM_OWNER) =
TSM password (TSM_PASSWORD) =
Automatic maintenance (AUTO_MAINT) = ON
Automatic database backup (AUTO_DB_BACKUP) = OFF
Automatic table maintenance (AUTO_TBL_MAINT) = ON
Automatic runstats (AUTO_RUNSTATS) = ON
Automatic statement statistics (AUTO_STMT_STATS) = ON
Automatic statistics profiling (AUTO_STATS_PROF) = OFF
Automatic profile updates (AUTO_PROF_UPD) = OFF
Automatic reorganization (AUTO_REORG) = OFF
Auto-Revalidation (AUTO_REVAL) = DEFERRED
Currently Committed (CUR_COMMIT) = ON
CHAR output with DECIMAL input (DEC_TO_CHAR_FMT) = NEW
Enable XML Character operations (ENABLE_XMLCHAR) = YES
WLM Collection Interval (minutes) (WLM_COLLECT_INT) = 20
Monitor Collect Settings
Request metrics (MON_REQ_METRICS) = BASE
Activity metrics (MON_ACT_METRICS) = BASE
Object metrics (MON_OBJ_METRICS) = BASE
Unit of work events (MON_UOW_DATA) = NONE
Lock timeout events (MON_LOCKTIMEOUT) = NONE
Deadlock events (MON_DEADLOCK) = WITHOUT_HIST
Lock wait events (MON_LOCKWAIT) = NONE
Lock wait event threshold (MON_LW_THRESH) = 5000000
Number of package list entries (MON_PKGLIST_SZ) = 32
Lock event notification level (MON_LCK_MSG_LVL) = 1
SMTP Server (SMTP_SERVER) =
SQL conditional compilation flags (SQL_CCFLAGS) =

```

8.3 KERNEL PARAMETER SETTINGS

The kernel parameter settings listed in Table 15 are configured on all nodes in the environment during deployment. Do not modify these parameters unless instructed to do so by IBM Smart Analytics System Support.

Table 15) Kernel parameter settings.

| Kernel Parameter | Value |
|---------------------------|---------------------|
| kernel.msgmni | 131072 |
| kernel.randomize_va_space | 0 |
| kernel.sem | 250 256000 32 32768 |
| kernel.shmmni | 32768 |
| kernel.suid_dumpable | 1 |
| vm.dirty_background_ratio | 5 |
| vm.dirty_background_ratio | 10 |
| vm.swappiness | 0 |

9 REFERENCES

9.1 NETAPP TECHNICAL REPORTS

- [NetApp Technical Report 3272, IBM DB2 on NetApp: Deployment and Best Practices](#)
- [NetApp Technical Report 3298, RAID-DP: NetApp Implementation of Double-Parity RAID for Data Protection](#)

9.2 NETAPP WEB SITES

- [Data Protection Best Practices Index](#)
- [FlexCache Software](#)
- [NetApp Community](#)
- [NetApp Support](#) (formerly NOW)
- [NetApp Data Storage Systems](#)
- [NetApp Snap Creator Framework Documentation](#)

9.3 IBM WEB SITES

- [IBM InfoSphere Warehouse Information Center](#)
- [IBM Support](#)
- [Installing InfoSphere Warehouse](#)
- [IBM DB2 Documentation](#)

NetApp provides no representations or warranties regarding the accuracy, reliability, or serviceability of any information or recommendations provided in this publication, or with respect to any results that may be obtained by the use of the information or observance of any recommendations provided herein. The information in this document is distributed AS IS, and the use of this information or the implementation of any recommendations or techniques herein is a customer's responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. This document and the information contained herein may be used solely in connection with the NetApp products discussed in this document.

Go further, faster®



www.netapp.com

© 2012 NetApp, Inc. All rights reserved. No portions of this document may be reproduced without prior written consent of NetApp, Inc. Specifications are subject to change without notice. NetApp, the NetApp logo, Go further, faster, Data ONTAP, FlexCache, FlexClone, FlexShare, FlexVol, Manage ONTAP, NOW, RAID-DP, Snap Creator, SnapMirror, SnapRestore, Snapshot, SnapVault, and WAFL are trademarks or registered trademarks of NetApp, Inc. in the United States and/or other countries. IBM, DB2, and InfoSphere are registered trademarks of IBM Corporation. Linux is a registered trademark of Linus Torvalds. Java is a registered trademark of Oracle Corporation. UNIX is a registered trademark of The Open Group. Intel and Xeon are registered trademarks of Intel Corporation. All other brands or products are trademarks or registered trademarks of their respective holders and should be treated as such. TR-3988-0112