**Research in Distributed Database System—Data failures and recovery**

Om Prakash Dhakal

Atlantis University

1442 Biscayne Bvld,Miami,FL

**Abstract**

In this article, investigating the problem of data consistency in the context of distributed database systems. My current research effort concentrates on survey results of data failure mechanisms such as hardware, software and network failure. Based on the survey results, I have been recommended different three sections. In addition, I am trying to research in data recovery techniques. The replication and concurrency control mechanisms are correct and maintain the consistency of the database, the failures of hardware and/or software at the processing site and communication network may destroy the consistency of the database. In order to cope with failures, distributed database systems must provide recovery mechanisms. The goal of check pointing is to save database states on a separate secure device so that the database can be recovered when errors and failures occur. A check pointing mechanism which does not interfere with the transaction processing in distributed environment is highly desirable for many applications, where restricting transaction activity during check pointing is not feasible.

Keywords: data failures, distributed database system (DDBS), research motivation, data recovery

**Research in Distributed Database System—Data failures and recovery**

# Introduction

A distributed database is a single logical database that is spread physically across computers in multiple locations that are connected by a data communications network. We emphasize that a distributed database is truly a database, not a loose collection of files. The distributed database is still centrally administered as a corporate resource while providing local flexibility and customization. The network must allow the users to share the data; thus a user (or program) at location A must be able to access (and perhaps update) data at location B. The sites of a distributed system may be spread over a large area (such as the United States or the world) or over a small area (such as a building or campus). The computers may range from microcomputers to large-scale computers or even supercomputers.

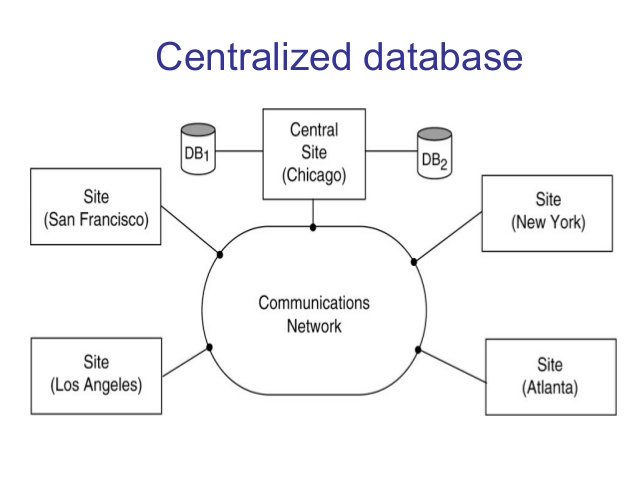
A major objective of distributed databases is to provide ease of access to data for users at many different locations. To meet this objective, the distributed database system must provide what is called location transparency, which means that a user (or user program) using data for querying or updating need not know the location of the data. Any request to retrieve or update data from any site is automatically forwarded by the system to the site or sites related to the processing request. Ideally, the user is unaware of the distribution of data, and all data in the network appear as a single logical database stored at one site. In this ideal case, a single query can join data from tables in multiple sites as if the data were all in one site.

A second objective of distributed databases is local autonomy, which is the capability to administer a local database and to operate independently when connections to other nodes have failed (Date, 1995). With local autonomy, each site has the capability to control local data, administer security, and log transactions and recover when local failures occur and to provide full access to local data to local users when any central or coordinating site cannot operate. In this case, data are locally owned and managed, even though they are accessible from remote sites. This implies that there is no reliance on a central site.

Faster response Depending on the way data are distributed, most requests for data by users at a particular site can be satisfied by data stored at that site. This speeds up query processing since communication and central computer delays are minimized. It may also be possible to split complex queries into subqueries that can be processed in parallel at several sites, providing even faster response.

**Literature Review**

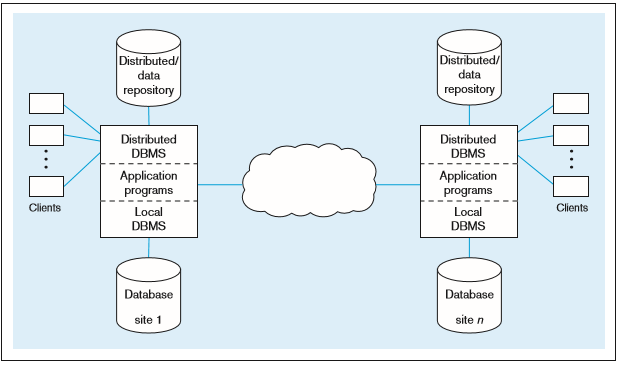
Before going on to Distributed Database Systems, let me briefly explain about Centralized Database Systems. Here all system components i.e. the database and the Database Management System (DBMS) reside at a single computer or site. Users may be able to access the centralized Database System remotely via terminals connected to the site; however all the data access and processing takes place at the central site. The following figure shows a centralized DB System.



Figure(c) Centralized Database with different sites

Distributed database management system (DDBMS) In a DDS, database applications running at any of the system's sites should be able to operate on any of the database fragments transparently i.e., as if the data come from a single database managed by one DBMS. The software that manages a distributed database in such a way is called DDBMS.

The notion of distributed database is different from that of decentralized database. The latter does not imply sharing of data by a communication network. The former implies a collection of sites connected together with some kind of network and where each site has a database in its own right, but the sites work together as if data was stored at only one site.(Potocki 2017)

 Figure (b) Distributed Database System

The communication network is used for information exchange, interaction and co-ordination among the various processes. Some systems are a library of routines intended at communication between hosts, while other systems link the various hosts tighter such that the application sees only one system. These two cases are called loosely and tightly couple distributed systems respectively.

## Problem Statement

Despite a lot of research over distributed database system, the challenge in a database failure and recovery techniques still remain unsolved.

## Research questions

1. Which collaborating components (hardware, software, and communication) played pivotal role to fail data in distributed database system?
2. How deadlocks are created in distributed system, how does it effects to manage data, and how can deadlocks be prevented in efficient ways?
3. How easy is it to perform data recovery issues? And what are the challenges of replicating data in different geo-graphical locations and how these challenges will impact in data replication process?

**Research Methodology**

The method of this study involved online Google form instrument that consists of three categorized (Data failed conditions portion of the survey, Deadlock portion of the survey, and Data recovery portion of the survey) 20 questions was administered to students who enrolled in computer science courses at one higher education. The students at the public campus consisted of undergraduate, and graduate students. The survey was published on March 19, 2017 and end on April 04, 2017.Total 22 responses had got from this survey.

**Survey questions**

**Data fail conditions portion of the survey:**

1. Which component played vital role to fail data in distributed database system?
2. Hardware,
3. Software,
4. Network
5. Which database architecture can mainly create data failure condition?
6. Homogenous architecture can fail,
7. Hybrid architecture can fail,
8. Heterogeneous architecture can fail
9. What is the common type of data failure?
10. Software failure,
11. Hard failure,
12. Network failure
13. Is network congestion creates problem for data?
14. Yes ,
15. NO,
16. May be
17. Can operating system paly role in this system?
18. Yes, it can,
19. No, it does not
20. Which of the following should you care in distributed system?
21. Connection of nodes
22. Data slicing
23. Fragmentation
24. Database schema

Deadlock portion of survey:

1. Which state mainly create deadlock?
2. Data fragmentation
3. Data slicing
4. Network congestion
5. Which transaction graph create deadlock?
6. Recall wait for graph
7. Local wait for graph
8. Global wait for graph
9. In distributed deadlock prevention approach, a transaction should acquire all the locks before starting to execute.
10. Strongly agree
11. Agree
12. Neither agree nor disagree
13. Disagree
14. Can play communication delay to create deadlock?
15. Yes
16. No
17. May be
18. I don’t know
19. How important resources in deadlock management?
20. Very important
21. Not very important
22. A system has 3 processes sharing 4 resources. If each process needs a maximum of 2 units then, deadlock:
23. Can never occur
24. May occur
25. Has to occur
26. None of these

Data recovery portion of survey:

1. What strategy is best to recover data in distributed database system?
2. Recovery Strategy includes transaction undo or rollback
3. Recovery strategies encompass restoring a past of the database from archival backup
4. Which action can take recovery manager during immediate update mode?
5. Transactions which are in active list and failed list are undone and written on the abort list.
6. Transactions which are in before-commit list are redone.
7. No action is taken for transactions in commit or abort lists.
8. In data replication process, which replication is more effective?
9. Synchronous replication—Majority approach—voting->Data item D replicated at n sites, Each copy maintains a version number

Biased protocol: read only write all, shared lock:simply request a lock on a D at one site that contains a copy of D.

1. Asynchronous replication: Primary site: Choose exactly one copy residing at a primary site

Peer to peer: More than one of the copies can be a master

1. How important of check pointing to recover data in distributed database system?
2. Extremely important
3. Important
4. Not important
5. I don’t know
6. Is redo/undo recovery method using for data recovery process?
7. Yes,
8. No,
9. May be
10. The purpose of the checkpoint process is to
11. Trigger a checkpoint
12. Reduce the workload on the log writer process
13. Write a transaction commit entry
14. Update data file and control file log sequence number
15. What is the first step to take to recover a NONARCHIEVELOG database?
16. Update the control file
17. Replace damaged or lost files
18. Make a backup
19. Which of the following backup technique is most space efficient?
20. Full backup
21. Incremental backup
22. Differential backup

**Analysis**

Results were analyzed through Google form, which is a demographic tool to can represent data in various demographic representation (charts, bars etc.).Data were also analyzed through Google spreadsheets after export from Google form. Each question response was represented through pie charts in percentage format.

**Results**

The results based on breakdown of each research question as follows:

**Research question one findings**

The first research question explored the data failure conditions in distributed database system. Different questions were asked to get this result. The first survey question was related with the component which involves to fail data in this system. The component result, which involved to fail data in distributed database indicated that 50% through software, 27.3% through network and 22.7% through hardware. Architecture of database result indicated that 54.5% failed data through heterogeneous, 36.4% failed through data through hybrid and 9.1% failed through homogenous architecture. The common data failure type result indicated that 40.9% by software, 31.8% by network and 27.3% by hardware. Network congestion can problem or not?, result indicated that 59.1% said yes, 31.8% said may be and 9.1% said no. The result about operating system role showed that 63.6% can affect to fail data rest of the percentage doesn’t affect. Connection of nodes should care in distributed system to protect data according to the result.

**Research question two findings**

The second question explored the condition of deadlock in distributed system. The survey question related with that about the state of deadlock creation result indicated that 63.6% through network congestion, 18.2% through data slicing and same percentage through data fragmentation. Transaction graph create deadlock or not?, The result indicated that 59.1% through global wait for graph , 27.3% through local wait for graph and 13.6% through recall wait for graph. In distributed deadlock prevention approach, a transaction should acquire all the locks before starting to execute, the result showed that 45.5% strongly agreed, 40.9% agreed and 13.6% neither agreed nor disagreed. The result showed about communication delay to create deadlock, 63.6% response was maybe, 18.2% response was yes, 9.1% response was I don’t know. The result of how important resources in deadlock management’s result showed that 90.9% said very important, 9.1% said not very important. A tricky question was asked in this section which was “A system has 3 processes during sharing 4 resources. If each process needs a maximum of 2 units then, deadlock”, result indicated that 54.5% may occur deadlock at that situation, 36.4% may occur deadlock and 9.1% has to occur.

**Research question three findings**

Third research question explored about the how affect the data recovery process in distributed system. The first question related about the strategy of recover data in distributed database system, the result indicated that 59.1% recover strategy includes transaction undo or rollback and 40.9% recovery strategies encompass restoring a pass of the database from archival backup. Second survey question related with this research question was what kind of action can take recovery manager during immediate update mode, the result indicated that 54.5% of transactions which are in before-commit list are redone, 27.3% of transaction which are in active list and failed list are undone and 18.2% had no action taken for transaction in commit or abort lists. The result about the type of effective replication method in recovery process indicated that 68.2% through asynchronous replication and 31.8% through synchronous replication. Check pointing is important or not during data recovery process, in this kind of scenario, the result showed that 50% said important, 31.8% extremely important and 18.2 said not important. The result of redo/undo recovery method using for data recovery process demonstrated that 63.6% said yes, 18.2% said no and same percentage said may be. The purpose of checkpoint, 45.5% to reduce the workload on the log writer process, 22.7% by update data file and control file log sequence number, 18.2% by trigger a checkpoint and 13.6% write a transaction commit entry. And the final survey question was about which backup technique is space efficient, result indicated that 45.5% through incremental backup, 31.8% through differential backup and 22.7% through full backup.

## Conclusions

Based on the results the following recommendations are:

**Recommendation for data failure condition:**

The findings from the research conclude the following: In order to detect the data failure condition in distributed database system, should be focus on software(50% results) i.e. what kind of software is using at that particular situation that does matter, heterogeneous architecture(54.5%) , network congestion, operating system and connection of nodes.

**Recommendation for deadlock management:**

Deadlock management process is not an easy to handle in distributed database system. The findings from the research conclude the following: In order to handle deadlock, need to know which state create deadlock, according to the result, focus on network congestion rather than data slicing and data fragmentation. Global wait for graph and communication delay can create deadlock. Resources are important in deadlock management, result indicated that 90.9% said very important.

**Recommendation for data recovery strategy:**

Based on the results, the best strategy to recover data in distributed database system is encompass restoring a pass of the database from archival backup(59.1%) rather than transaction undo or rollback. Replication is another recover strategy, according to the result, asynchronous replication (68.2%) is more effective rather than synchronous replication.(31.8%).Check pointing is also important to recover data in this system. Redo/undo recovery method can use to recover data. The purpose of checkpoint is to reduce workload on the log writer process and incremental backup technique is most space efficient according to the result.

I recommend to prevent data through various recovery levels are recovery to the correct state, recovery to a checkpoint current state, recovery to a possible previous state, recovery to a valid state, recovery to a consistent state and crash resistance (prevention).

**Future research consideration on distributed database system**

This research project will concentrate in different algorithms that can be contribute to detect data failure state and try to improve data performance capability. Also, detect the state of network congestion during sharing different resources in different nodes. Different algorithms, they are related with how to prevent data during sending data within the interconnected network. In this context, have to make sure that the efficient data integrity mechanism in this distributed system.

And finally, in this article, offer a vision of a possible future of the replication movement. The plan is to implement this vision via the Virtual Data Center project, which – by automating the process of finding, sharing, archiving, sub setting, and converting, analyzing, and distributing data – may greatly facilitate adherence to the replication standard.

References

1. What is distributed database? - Definition from WhatIs.com. (n.d.). Retrieved April 09, 2017, from <http://searchoracle.techtarget.com/definition/distributed-database>
2. T. (n.d.). Distributed DBMS Tutorial. Retrieved April 09, 2017, from http://www.tutorialspoint.com/distributed\_dbms/
3. Types of Distributed Database Systems. (n.d.). Retrieved April 09, 2017, from <http://berb.github.io/diploma-thesis/original/063_types.html>
4. Lampson, B., & Sturgis, H. E. (1979, January 01). Crash Recovery in a Distributed Data Storage System. Retrieved April 09, 2017, from https://www.microsoft.com/en-us/research/publication/crash-recovery-in-a-distributed-data-storage-system/