Simulation and Modeling Techniques - Final Project Fall 2016

Mohamed Elmoudni, Shazia Khan and Sekhar Mekala

Abstract

This project is about simulating a Data Center Relocation from location A to location or destination B. A data center relocation (DCR) is not just about moving servers and plugging them in at their new locale. In reality, DCR can be one of a company's most complex and challenging endeavors. With mission critical information and high-stakes money on the line, the failure of any key steps in the process can have potentially devastating repercussions. In this project we propose a new methodology to transfer the data in a logical fashion with minimal outage and hence minimal cost.

Key words and Phrases: Data center relocation, Data center migration, Data and server consolidation, Application based data center migration

Introduction

Data Center Relocation (DCR) is regarded as one of the complex projects of the Information Technology industry. The need to DCR can occur due to merging (or acquisations) of companies, or to consolidate the existing distributed data repositories into a common data repository to minimize costs or it can be due to government regulation (such as maintenance of Disaster Recovery site). In this paper we review two of the common methodologies currently existing in the industry, and propose a new methodology that could minimize the costs of DCR.

Literature review

Organizations and businesses are contantly evolving by acquiring and merging. As such, IT organizations today are under constant business pressure to transform their infrastructure to reduce costs, increase availability, improve business continutity, provide information security, deliver greater availability to meet business expectations in a cloud era, comply with federal and industry regulations and remain competitive. Whatever the reason, Consolidation and migration undertakings are complex and rare endeavors with little room for error and as such according to Hewlett packard [1], 88% of enterprises are currently in the early phases of either consolidating servers or executing a major data center relocation.

Server central's [2] survey results show that when asked by customers, partners, prospects, and colleagues about data center migrations, they all respond that migrations are a pain.

- 50% of migrations exceed budget, exceed migration windows, and result in disruption.
- 60% of companies have delayed a migration after planning.
- 50% cited concerns about downtime associated with the migration.
- 40% cited concerns about downtime associated with the migration.
- 20% cited a lack of resources

A "business-friendly" migration is required for a shifting of focus from infrastructure management to the application level, along with early participation and buy-in from all involved stakeholders. Once the understanding of business environment and its supporting applications and systems is established, it is important to understand and analyze the interdependencies between the applications and the timing of critical data transfers.

Absolute care must be taken to package the migration of the applications and systems into complimentary "migration bundles"-that is, components that must move together to ensure that operating in separate data centers won't cause any breakage or performance delays. Dell [3] uses advanced automation tool called TransitionManager to rapidly document and analyze interdependencies between applications, servers, and storage in real-time. By analyzing the data in real-time usingTransitionManager, Dell is able to deliver data center initiatives in the shortest amount of time and with the lowest amount of associated risk.

Data center migration projects may seem like a daunting task, but with the correct approach, automated tools and simulation, we can greatly reduce human errors and outage periods and much of the risk can be mitigated. The automated tools utilized throughout Dell EMC's application-centric approach both accelerate overall project time, shortening it by as much as 50%, and eliminate nearly 98% of human error. Through the combination of seasoned expertise, robust methodology, and rapid discovery tools, Dell EMC executes data center consolidation, modernization, and migration initiatives in the shortest amount of time, with the lowest risk.

In this project we show how a piece-meal migration strategy can achieve minimal outage and maximal cost savings. Our strategy is driven by migrating the logical entities (such as RDBMS tables) in an ordered fashion so that the overall downtime of the migration is minimized.

Methodology

We will be generating artificial data based on the assumptions given in Figure-1, and use the simulated data to estimate the costs associated with the different migration strategies.

Figure-1: Assumptions of the system for data generation.

Component	Assumption
Data size	~1TB – 3TB
Source Location	A
Target Location	B
Physical distance between A and B	500 miles
Time to unplug, and load the servers to the truck at A	Exponential with 6hours as the average
Time to unload and set up the servers at B	Exponential with 6hours as the average
Speed of truck	Normal distribution with a mean of 45 MPH and standard deviation of 5MPH
WAN Speed	1GB/sec
Number of Programs	200
Number of tables accessed by a program	Poisson distribution with a mean of 3/program
Table Size	Std. Normal distribution with a mean size of 3GB, and std. dev of 1GB
Program Downtime cost	Poisson distribution with a mean of $50/second
Time to access remote table by a program	3seconds X Table size in GB
Program slowness cost	Poisson distribution with a mean of $3/second
Referential constraints cost	Parent Table size in GB X Child table size in GB X 0.01
Referential constraints	Poisson distribution with 3 tables as the average
Active time of program	Assumed to be up and running all the time, except when the dependent table is in transit
Program Downtime	Program will be down when the dependent table is in transit

In every migration strategy discussed in this paper, we quantify the cost by estimating the over all downtime of the business, and thus compute the overall dollar amount loss by the business. In the real world, this data can be obtained by colsely examining the IT infrastructure and applications, and the impact of these systems

on the business. For instance to estimate the loss due to the downtime of an application, we may examine the daily server traffic, and fit a distribution model (or) fit an emperical distribution to assess the business impact (quantifying the loss due to the server unavailability).

We used R programs (given in Appendix-A, B and C) to generate the data for our simulation purpose. Using the simulated data, we estimate the costs of migration of three DCR methodologies discussed below:

Method-1 In this method, we shut down all the source systems (at location A), load them into truck(s) and drive them to the destination (in location B).

The cost associated with this method can be expressed as:

$$Method1_{cost} = L.(UP + TT + RP)$$

Where,

L = Loss per unit time due to down-time

UP= Time to un-plug servers and load to truck at the source

TT = Transit Time (Time taken by the truck to reach the destination)

RP = Time to re-plug the servers at the destination

This method is suitable if the business loss (L) is very low or the downtime $UP + TT + RP$ is low or both L and $UP + TT + RP$ are low. Some of the disadvantages of this method are given below:

- The risks are high as systems can be damaged while in transit.
- In addition, this method will have a major impact on business as systems will be unavailable through load, transit, and unload times.

Method-2 The second method is called *asset swap method*, in which we lease server (cpu/memory) equipment at the destination side, transfer data across the wide area network WAN, and then promote the destination as production site. This method carries a cost as we need to lease server assets in the destination site. Also there would be a high outage when the data is transferred between locations via WAN.

The cost associated with this method can be expressed as:

$$Method2_{cost} = L.(VL/WAN)$$

Where,

L = Loss per unit time due to down-time (example: \$50/sec)

WAN = Speed of WAN expressed as volume transferred per unit time (example: 1GB/sec)

VL = Volume of data to be transferred between the servers (example: 500 TB)

As the speed of WAN increases, the cost decreases, and as the volume of data increases the cost increases.

Method-3 The third method is based on software table replication method, where we divide the data logically into independent chunks (for instance RDBMS tables) and transfer them one at a time based on their access time and relationships (referential integrity constraints), while all other data is still available for access (Read/write). The cost associated with this method can be expressed as:

$$Method3_{cost} = \sum_{i=1}^{N} (L_i.(VL_i/WAN) + (RL_i. + \sum_{j=1}^{M} PGR_j).TR_i)$$

where

N = Number of tables

M = Number of dependent programs on the i^{th} Table

VL_i = Volume or size of i^{th} table

L_i = Loss per unit time due to the unavailability of i^{th} table = $\sum_{j=1}^{M} PGD_j$, where PGD_j is the loss / unit time due to the unavailability of j^{th} program, which is dependent on the i^{th} table

WAN = Speed of WAN per unit time

RL_i = Loss / unit time to maintain RI relation between local tables related to i^{th} table and the remote table-i

PGR_j = Loss / unit time due to slowness of j^{th} program

TR_i = Time that the i^{th} table remain remote

The cost associated with the first component (given below) in $Method-3_{cost}$ cannot be reduced.

$$Method3_{C1} = \sum_{i=1}^{N} L_i.(VL_i/WAN)$$

The cost associated with the second component (given below) in $Method3_{cost}$ can be reduced by moving the tables in a specific order, so that the loss incurred due to a table being remote is the least.

$$Method3_{C2} = \sum_{i=1}^{N}(RL_i + \sum_{j=1}^{M} PGR_j).TR_i$$

We will use linear programming technique to reduce the cost due to the $Method3_{C2}$

The *Method-3* data migration process is given in Figure-2 below:

Figure-2: Method-3 data migration process

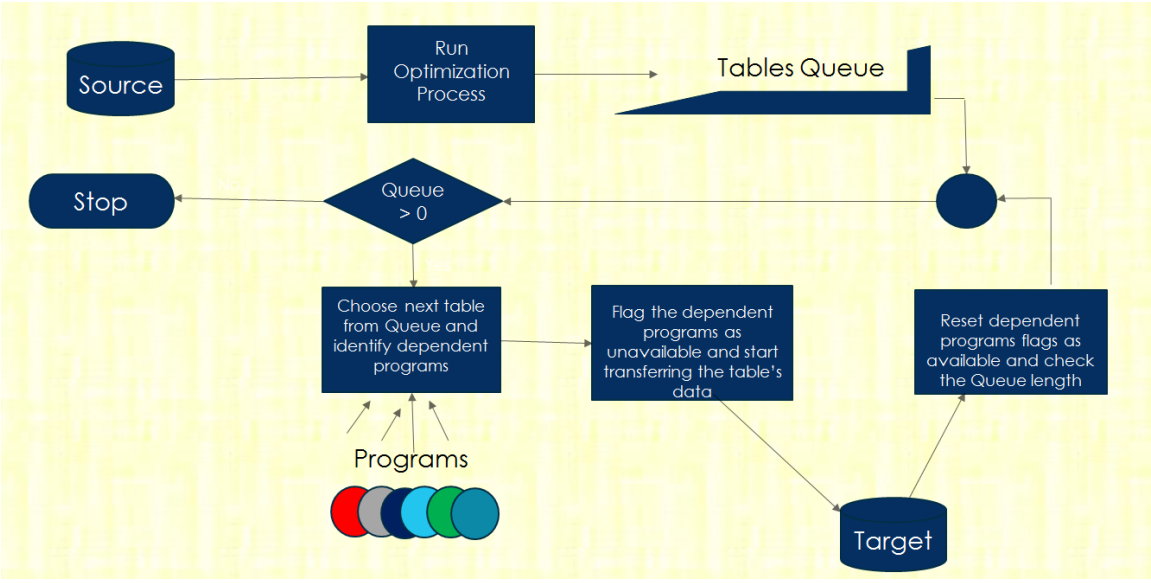


Figure 2 shows the proposed data migration process using *Method-3*. The list of tables from the source database are assigned a cost, which is based on the cost associated with the downtime of the dependent

program(s) on the table (the table will be unavailable while it is being copied from source to the destination, and hence the dependent programs on the table are also down), and the cost due to slowness in the programs since the table is remote and the programs accessing the remote table will have delayed processing.

Based on the cost associated with each table, and the time needed to copy the table from source to destination, the tables will be copied in an optimal order, so that the over all cost of migration is minimized.

Example of a small system to explain the costs of 3 methods

Let us assume a system with 7 tables, and 3 programs that have to be migrated from location A to location B. The physical distance between A and B is 500 miles, and the truck has an average speed of 45MPH with 5MPH as the standard deviation (normal distribution). The servers load time to truck and unload time from truck is having exponential distribution with 6 hours as the average. The speed of WAN between A and B is 1GB/sec, and is assumed to be constant. See Figure-1 for all the assumptions. We used the data generation program (see Appendix-A), to generate the data for this example.

The mapping between Tables and programs, along with the unavailability and remote table access costs are given in Figure-3, below:

Figure-3: Program-Tble dependency, access costs and down time costs

Program ID	Table ID	Remote Access Time (sec)	Loss due to program slowness	Remote Access Loss/sec	Program Downtime Loss Per sec
1	3	16.21396	5	81.0698	61
1	6	5.557029	5	27.78514	61
2	3	16.21396	2	32.42792	52
2	7	8.131615	2	16.26323	52
3	5	6.602972	5	33.01486	44
3	3	16.21396	5	81.0698	44
3	1	8.115839	5	40.57919	44

The figure-3 shows that program 1 is accessing the tables 3 and 6. The program 1 will need 16.21 seconds to access table 3 and 5.55 seconds to access table 6, when the table 3 and table 6 are remote, respectively. The loss encountered by the program-1 (per second) due to slowness is $5/sec. Hence, the program-1 will incur a loss of $81 and $27.78 to access 3 and 6 tables during its execution (or cost incurred to access remote table). We assume that the program is executed every second, and hence the remote access cost for program 1 to access table 3 is $81/sec and Table 6 is $27.78/sec. The program 1 down time cost is $61/sec (the program 1 will be down when table 3 or table 6 is in transit). Figure-3 also shows the costs associated with other programs.

The parent child relationship between the tables and the RI Costs are displayed in Figure-4:

Figure-4: RI (Referential Integrity) costs associated with tables

Parent Table	Child Table	Parent Table Size (GB)	Child Table Size (GB)	Parent Remote Time (sec)	Child Remote Time (sec)	RI Cost ($/sec)
1	4	2.70528	3.763593	3.763593	2.70528	0.10181572
1	6	2.70528	1.852343	1.852343	2.70528	0.05011106
1	7	2.70528	2.710538	2.710538	2.70528	0.07332764
1	3	2.70528	5.404653	5.404653	2.70528	0.14621098
1	5	2.70528	2.200991	2.200991	2.70528	0.05954295
2	5	2.994233	2.200991	2.200991	2.994233	0.06590279

The Figure 4 shows the relationships between the tables, along with their sizes and the cost of maintaining RI constraints between the tables. For instance, table 1 is the parent table for 3,4,5,6,7 tables. If table 1 is moved to remote location, but 3,4,5,6,7 stay local, then there will be a cost involved to maintain referential integrity between table 1 and tables 3,4,5,6,7. This cost is $0.10181572/sec, $0.05011106/sec, $0.07332764/sec, $0.14621098/sec, $0.05954295/sec (for the tables 4,6,7,3,5 respectively). But if 3 is moved to remote locaton, the the cost of maintaining RI will be $0.14621098/sec, since there is only one table (table 1) related to table 3.

Using the information from figures 3 and 4, we can get the cost of moving each table and the transfer time as given below:

Figure-5: Costs associated with each table in the system

Table ID	Loss due to programs accessing remote table ($/sec)	Loss to maintain RI due to the table being remote ($/sec)	Total remote access cost ($/sec)	Loss due to programs downtime, when the table is unavailable ($/sec)	Table size in GB	Time to transfer table (sec)	Total Downtime Cost ($)	Min Time the table has to stay remotely (sec)	Best possible remote time (sec)
1	40.57919	0.4310084	41.0102017	44	2.70528	2.70528	119.0323	26.531996	X1
2	0	0.3641965	0.3641965	0	2.994233	2.994233	0	28.095386	X2
3	194.56752	0.5546468	195.1221688	157	5.404653	5.404653	848.53058	22.725778	X3
4	0	0.1126908	0.1126908	0	3.763593	3.763593	0	8.410051	X4
5	33.01486	0.3040604	33.3189216	44	2.200991	2.200991	96.84359	22.224755	X5
6	27.78514	0.1555762	27.940721	61	1.852343	1.852343	112.99292	22.21359	X5
7	16.26323	0.4395359	16.7027664	52	2.710538	2.710538	140.948	26.526737	X7
				$358/sec	21.631631GB	**21.631631sec**	$1,318.35		

Let us calculate the Method-1 cost. We simulated the transfer via truck, which travels at the speed of 45MPH, with a std. dev. of 5MPH, and the speed is assumed to be normally distributed. We also assumed the

servers load time and unload time to be 6 hours on average, exponentially distributed. The distance between location A and location B is assumed to be 500 miles. With these assumptions, we simulated the Method-1 (using Simio), for 1000 times, and obtained the average total down time as 17.322 hours. This downtime will translate to a total business loss of

$$Method1_c = (358)(17.322)(60)(60) = \$22324594$$

If we use Method-2, with a WAN speed of 1GB/sec (constant speed), then we will have a business loss given below:

$$Method2_c = (358)(21.631631) = \$7744.124$$

Let us find the cost associated with Method-3. The column "Total Downtime Cost sec" in Figure-5, has calculated the component-1 in the method 3 cost formula. This cost of $1318.35 cannot be reduced any further. The column "Time the table has to stay remotely" in figure-5 talks about the min possible time the table needs to stay remotely, while its related tables are not moved to the remote location. This column (Time the table has to stay remotely) and the "Total Remote Access Cost Per sec", will decide the overall cost of migration, and sometimes it could be optimal for a table to stay remotely for a long time than listed in "Time the table has to stay remotely", since other important tables might need to be moved to remote location on a higher priority. Let X1, X2, X3, X4, X5, X6, X7 be the optimal times the tables 1,2,3,4,5,6,7 need to stay remotely so that the over all cost ("Total Remote Access Cost Per sec" X "Min Time the table has to stay remotely") is minimized. In other words, we can express this problem as a linear programming problem as given below:

Minimize:

$$41.0102017X1+0.3641965X2+195.1221688X3+0.1126908X4+33.3189216X5+27.940721X5+16.7027664X7$$

Subject To:

$$(X1, X2, X3, X4, X5, X6, X7) \geq (26.531996, 28.095386, 22.725778, 8.410051, 22.224755, 22.21359, 26.526737)$$

Using this logic, we obtained the optimal solution as:

$$(X1, X2, X3, X4, X5, X6, X7) = (26.531996, 28.095386, 22.725778, 8.410051, 22.224755, 22.213590, 26.526737)$$

Sorting the obtained solution in decreasing order, will give us the optimal transfer order of the tables:

Figure 6: Tables transfer order (as per the overall cost minimization logic)

	Table_ID	Best_Remote_Time	Remote_Access_Cost
2	2	28.095386	0.3641965
1	1	26.531996	41.0102017
7	7	26.526737	16.7027664
3	3	22.725778	195.1221688
5	5	22.224755	33.3189216
6	6	22.213590	27.9407210
4	4	8.410051	0.1126908

The minimum cost (component 2 of Method-3) is

$$41.0102017X1+0.3641965X2+195.1221688X3+0.1126908X4+33.3189216X5+27.940721X5+16.7027664X7 = 7337.804$$

where

$$[X1, X2, X3, X4, X5, X6, X7] = [26.531996, 28.095386, 22.725778, 8.410051, 22.224755, 22.213590, 26.526737]$$

The total cost of Method-3 is

$$Method3_{c1} + Method3_{c2} = \$1318.35 + \$7337.804 = \$8656.154$$

Hence, for this example, we obtained the following costs for the three methods:

Figure 7: Method-1, Method-2, and Method-3 costs for transferring a system with 3 programs and 7 tables

Method	Cost
1	22324594.000
2	7744.124
3	8656.154

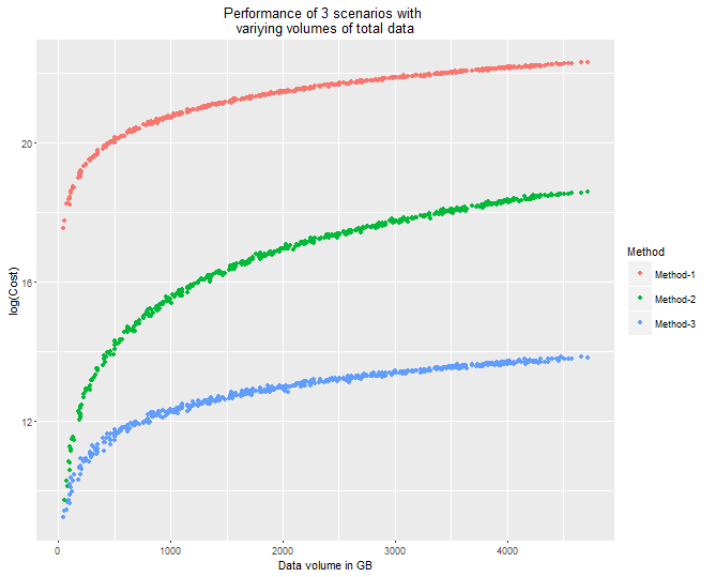
Figure-7 shows that the cost of Methos-2 is the minimum (for this example). But as the size of the data increases, the cost of Method-2 increases more rapidly than Method-3. This is discussed further in the *Experimentation and Results* section.

Experimentation and Results

Experiment 1 (Costs of methods as the size of the total data increases, with average table size of 3GB)

Using the R program given in Appendix-B, we generated the data (representing a system with different number of tables/sizes), and computed the cost of three methods. The cost of Method-1 is always high, while the cost of Method-2 and Method-3 are almost the same initially (or till the volume of data is less than 20 GB approximately), and as the size of the data increases, the cost of Method-2 increases drastically than Method-3.

Figure 8: Cost comparison of three methods, with varying volume of data (average size of each table being 3GB with a std dev of 1GB, normally distributed)



Experiment-2 (Finding the Confidence Intervals for the costs, to transfer an amount of 50TB data (approximately))

In this experiment, we will estimate the costs of all the three methods, assuming that the average cost of each table to be 3GB, while the total volume of the data remaining cosntant (approximately 50TB). The experiment has been run for 500 times to gather the data.

Figure 9: Costs of three methods to transfer 50TB of data, with the average table size of 3GB

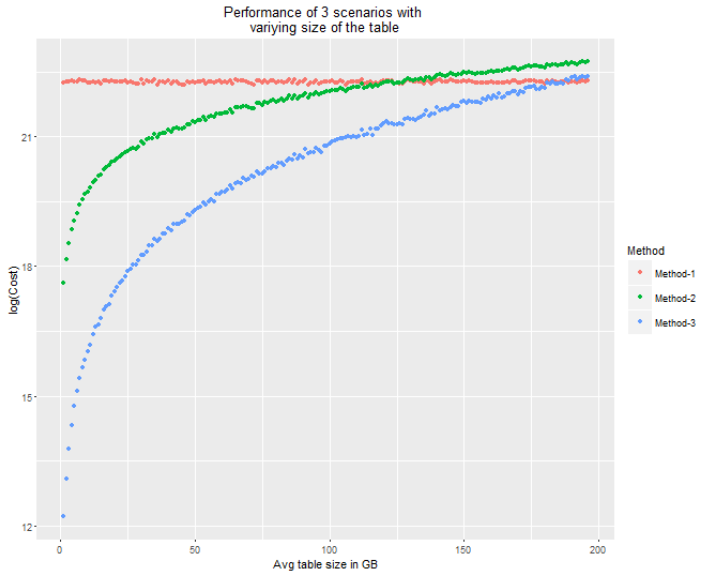
Method	Avg_Cost	SD_Cost	Low_95_CI	High_95_CI
Method-1	4672455543.7	129504782.43	4418626170.2	4926284917
Method-2	112448965.9	3215078.04	106147413.0	118750519
Method-3	961639.4	41149.39	880986.6	1042292

For an average data size of 50TB, the Method-3 has the lowest cost.

Experiment 3 (Finding the cost of the three methods with different average sizes of table (1GB to 200 GB))

Figure-10 shows how the costs of three methods increase as the average size of the tables increase from 1GB to 200GB. We can clearly see that the cost of Method-2 exceeds Method-1 around 150GB of the average table size, while the cost of Method-3 also surpasses Method-1, around 200GB of average table size. Hence the average table size plays a very important role in determining the cost of the methods.

Figure 10: Cost comparison of three methods, with varying average table size (from 1GB to 200GB)



Also, from Figure-10, we can infer that the cost of Method-1 is slightly decreasing, as the average size of the table increases, and this decrease is predominant around 150GB of average table size.

Summary and future work

In this project, we generated artificial data using the assumptions listed in Figure-1. Using these assumptions, and the data generated, we tested the costs of three methods given below:

- The Method-1, in which the servers are transfered physically on a truck
- The Method-2, in which the data is transferred on a WAN by making the servers unavailable until the transfer is complete and
- The Method-3, in which the data was transferred in a logical fashion, by dividing the data logically at the RDBMS table level.

For each of these three methods, we tested 3 scenarios:

1. The effect of the data size on the costs. The total data volume is increased (but the average size of the table is made as 3GB), and the costs of the 3 methods are studied. Figure-8 shows that the Method-3 performs the best in this scenario.

2. The total volume of data is made as 50TB approximately, and the costs are computed for 500 iterations of the three methods. The thr 95% confidence intervals along with the average and standard deviations of the costs are given in Figure-9. In this scenario also, the cost of Method-3 is the least.

3. The average size of the table has been increased gradually from 1GB to 200GB, and in this scenario, the costs of Method-2 and Method-3 increased more than Method-1. The Method-2 cost has surpassed Method-1 cost around 150 tables, and the Method-3 cost has surpassed Method-1 cost around 200 tables. Hence based on the assumptions listed in Figure-1, as the average size of the table increases, the cost of Method-1 may be more optimal than Method-2 and 3. However Method-1 also has a risk of loosing the data in the transit.

As a part of future work, we would like to perform the following:

- We assumed that the programs are constantly up and running all the time. We would like to further experiment assuming a distribution for the program's execution and the duration of the execution.

Some of the existing RDBMS products do not support referential integrity constraints between remote tables. We will research the best way to mitigate such problems (perhaps a simple application maintaining the RI constraints may be a viable solution). But there will be a cost associated with the changes to the existing application programs.

References

[1] 'https://www.hpe.com/h20195/V2/GetPDF.aspx/4AA5-1861ENW.pdf'

[2] 'https://go.servercentral.com/hubfs/PDFs/The_Successful_Data_Center_Migration-A_ServerCentral_White_Paper.pdf?t=1475680065154'

[3] 'https://www.emc.com/collateral/white-papers/h6151-ep-application-centric-approach-to-data-center-migration.pdf'

Appendix A:

Code to run experiment-1

```
##Appendix-A

rm(list=ls())
library(dplyr)
library(lpSolve)

set.seed(1)

Scenario_1_cost <- vector()
Scenario_2_cost <- vector()
Scenario_3_cost <- vector()
Total_Tables_size <- vector()
Number_of_tables <- vector()


Transfer_Speed <- 1

cnt1=1
for(cnt in seq(from=5,to=500,by=1))
{
  #Generate the list of programs
  n=cnt
  Program_ID <- 1:n

  #Program_depth <- rpois(lambda=1,n)

  #Generate the number of tables accessed by each of the program
  Total_Tables_Accessed <- rpois(lambda=3,n)

  #Generate the program slowness cost due to remote table access
  Slowness_Cost_per_unit_time <- rpois(lambda=3,n)

  #Generate the downtime cost of the program
  Down_time_cost_per_unit_time <- rpois(lambda=50,n)

  #Prepare the program, the number of tables it accesses,
  #slowness cost per unit time, downtime cost per unit time
  Programs_df = data.frame(Program_ID,
                           Total_Tables_Accessed,
                           Slowness_Cost_per_unit_time,
                           Down_time_cost_per_unit_time)


  #Generate the required number of tables
  Total_Tables = sum(Total_Tables_Accessed)


  Table_ID <- 1:Total_Tables
```

```r
#Generate the size of tables

Size <- abs(rnorm(Total_Tables,mean=3,sd=1))

Table_df <- data.frame(Table_ID, Size)

#Generate the table's local access and remote access time
Table_df$Local_Access_Time <- Table_df$Size * 0
Table_df$Remote_Access_Time <- Table_df$Size * 3

#Now prepare the remote table access cost
Atleast_One_Tab <- Programs_df[Programs_df$Total_Tables_Accessed>0,]

Pgm_Tab_df <- data.frame()

for(i in 1:nrow(Atleast_One_Tab))
{

  x = rep(Atleast_One_Tab$Program_ID[i],
          Atleast_One_Tab$Total_Tables_Accessed[i])

  y = sample(1:Total_Tables,
             Atleast_One_Tab$Total_Tables_Accessed[i])

  z = Table_df$Remote_Access_Time[y]
  Remote_Access_Loss = z * Atleast_One_Tab$Slowness_Cost_per_unit_time[i]
  Program_Downtime_Loss_Per_Unit_Time = Atleast_One_Tab$Down_time_cost_per_unit_time[i]
  Pgm_Tab_df <- rbind(Pgm_Tab_df,data.frame(
            Program_ID=x,Table_ID=y,Remote_Access_Time=z,
            Remote_Access_Loss,Program_Downtime_Loss_Per_Unit_Time))

}


#Display the tables list and programs data frames
#Table_df
#Programs_df

#Display the programs-tables mapping
#Pgm_Tab_df

#The above data frame has the following columns:
#Program_ID - Program ID
#Table_ID - Unique table accessed by the program
#Remote_Access_Time - Time to access the remote table
#Remote_Access_Loss - Loss incurred by the program while accessing the remote table once
#Program_Downtime_Loss_Per_Unit_Time - Loss incurred by the program due to downtime/sec



Table_Remote_Access_Cost <- Pgm_Tab_df %>%
  group_by(Table_ID) %>%
```

```
  summarise(Remote_Access_Loss=sum(Remote_Access_Loss))

temp <- Table_df[-which(Table_df$Table_ID %in% Table_Remote_Access_Cost$Table_ID),]

if(nrow(temp) > 0)
{
  Table_Remote_Access_Cost <- rbind(data.frame(Table_ID=temp$Table_ID,
                                               Remote_Access_Loss=0),
                                    Table_Remote_Access_Cost)
}

Table_Remote_Access_Cost <- Table_Remote_Access_Cost[
                              order(Table_Remote_Access_Cost$Table_ID),
                              ]

#The Table_Remote_Access_Cost data frame has the following columns:
#Table_ID - Unique identifier of the table
#Remote_Access_Loss - Loss incurred to access the table remotely by all the programs for one time

#-----------------------------------------------------
##Now generating the cost for maintaining the RI
#-----------------------------------------------------

Table_df$Number_of_Related_Tables = rpois(lambda=3,nrow(Table_df))

Table_Down_Time_Cost <- Pgm_Tab_df %>%
  group_by(Table_ID) %>%
  summarise(Pgms_Down_Time_Loss_Per_Unit_Time=
              sum(Program_Downtime_Loss_Per_Unit_Time))

temp <- Table_df[-which(Table_df$Table_ID %in% Table_Down_Time_Cost$Table_ID),]

if(nrow(temp) > 0)
{
  Table_Down_Time_Cost <- rbind(data.frame(Table_ID=temp$Table_ID,
                                           Pgms_Down_Time_Loss_Per_Unit_Time=0),
                                Table_Down_Time_Cost)
}

Table_Down_Time_Cost <- Table_Down_Time_Cost[order(Table_Down_Time_Cost$Table_ID),]

#The Table_Down_Time_Cost data frame has the following columns. This df has the overall cost
#associated with a table unavailability.

#Table_ID - Table identifier
#Pgms_Down_Time_Loss_Per_Unit_Time - Loss to business due to program unavailability

Tables = nrow(Table_df)
df <- data.frame()
i=2
for(i in 1:Tables)
{
  x  <-  sample(1:Tables)
```

```
  x <- x[-which(x %in% i)]

  if(Table_df$Number_of_Related_Tables[i] > 0)
  {
    y <- sample(x,Table_df$Number_of_Related_Tables[i])
    dep_tab_sz <- Table_df$Size[y]

  }
  else{
    next
  }


  z <- rep(i,length(y))
  parent_tab_sz <- rep(Table_df$Size[i],length(y))
  df <- rbind(df,
              data.frame(Parent_Table=z,Child_Table=y,
                         Parent_Table_Size=parent_tab_sz,
                         Child_Table_Size=dep_tab_sz))

}

df$Parent_Table_Size = ifelse(
  df$Parent_Table_Size == 0, 1,
  df$Parent_Table_Size)

df$Parent_Remote_Time = df$Child_Table_Size / Transfer_Speed
df$Child_Remote_Time = df$Parent_Table_Size / Transfer_Speed

df$RI_Cost = df$Parent_Table_Size * df$Child_Table_Size * .01

Table_RI_Cost <- df

Child_Table_Remote_Time <- Table_RI_Cost %>%
  group_by(Child_Table) %>%
  summarise(Remote_Time=sum(Child_Remote_Time))

names(Child_Table_Remote_Time)[1] <- c("Table")

Parent_Table_Remote_Time <- Table_RI_Cost %>%
  group_by(Parent_Table) %>%
  summarise(Remote_Time=sum(Parent_Remote_Time))
names(Parent_Table_Remote_Time)[1] <- c("Table")

df <- rbind(Child_Table_Remote_Time,Parent_Table_Remote_Time)

df <- df %>%
  group_by(Table) %>%
  summarise(Remote_Time=sum(Remote_Time))

x = Table_df$Table_ID[-which(df$Table %in% Table_df$Table_ID)]
if(length(x) > 0){
  df = rbind(df,data.frame(Table=x,Remote_Time=0))
```

```
}

df=df[order(df$Table),]
Table_Remote_Time = df

Table_RI_Cost_Cumulative <- Table_RI_Cost %>%
  group_by(Parent_Table) %>%
  summarise(Total_RI_Cost=sum(RI_Cost),
            Table_Size=max(Parent_Table_Size))


temp <- Table_df[-which(Table_df$Table_ID %in% Table_RI_Cost_Cumulative$Parent_Table),]

if(nrow(temp) > 0)
{
  Table_RI_Cost_Cumulative <- rbind(data.frame(Parent_Table=temp$Table_ID,
                                               Total_RI_Cost=0,
                                               Table_Size=temp$Size),
                                    Table_RI_Cost_Cumulative)
}

Table_RI_Cost_Cumulative <- Table_RI_Cost_Cumulative[
  order(Table_RI_Cost_Cumulative$Parent_Table),]

#The data frame Table_RI_Cost_Cumulative has Parent table, the total RI cost for the table
#and table size. The RI cost is in dollar/sec
#Parent_Table - Parent table ID
#Total_RI_Cost - RI Cost / sec
#Table_Size - Parent table size


#Final data sets:
#head(Table_RI_Cost_Cumulative)
#head(Table_Remote_Access_Cost)
#head(Table_Down_Time_Cost)

#head(Table_Remote_Time)




Cost_df <- data.frame(Table_ID=Table_Remote_Access_Cost$Table_ID,
  Loss_Due_To_Program=Table_Remote_Access_Cost$Remote_Access_Loss,
  Loss_Due_To_RI = Table_RI_Cost_Cumulative$Total_RI_Cost,
  Remote_Access_Cost_Per_Unit_Time = Table_Remote_Access_Cost$Remote_Access_Loss +
    Table_RI_Cost_Cumulative$Total_RI_Cost,
  Programs_Downtime_Cost_Per_Unit_Time=Table_Down_Time_Cost$Pgms_Down_Time_Loss_Per_Unit_Time,
  Table_Size=Table_RI_Cost_Cumulative$Table_Size
)



Cost_df$Time_To_Transfer <- Cost_df$Table_Size/Transfer_Speed
```

```
  head(Cost_df)
  Cost_df$Total_Downtime_Cost = Cost_df$Time_To_Transfer *
                      Cost_df$Programs_Downtime_Cost_Per_Unit_Time

  Cost_df$Min_Remote_Time <- Table_Remote_Time$Remote_Time


  Constraint = Cost_df$Min_Remote_Time

  Scenario_1_cost[cnt1] = sum(Cost_df$Programs_Downtime_Cost_Per_Unit_Time) * 17.322*60*60
  Scenario_2_cost[cnt1] = sum(Table_df$Size)* sum(Programs_df$Down_time_cost_per_unit_time)
  Scenario_3_cost[cnt1] = sum(Cost_df$Programs_Downtime_Cost_Per_Unit_Time *
                                Cost_df$Time_To_Transfer)+
                           sum(Cost_df$Remote_Access_Cost_Per_Unit_Time *
                               Cost_df$Min_Remote_Time)
  Number_of_tables[cnt1] = Total_Tables
  Total_Tables_size[cnt1] = sum(Cost_df$Table_Size)

  cnt1 = cnt1+1
}


library(ggplot2)
df <- data.frame(Method=c("Method-1"),
                 Cost=Scenario_1_cost, Data_Size = Total_Tables_size)
df = rbind(df,data.frame(Method=c("Method-2"),
                         Cost=Scenario_2_cost, Data_Size = Total_Tables_size))
df = rbind(df,data.frame(Method=c("Method-3"),
                         Cost=Scenario_3_cost, Data_Size = Total_Tables_size))

df$Cost = log(df$Cost)

ggplot(data=df,aes(x=Data_Size,y=Cost,color=Method))+
  geom_point()+
  labs(title="Performance of 3 scenarios with \nvariying volumes of total data",x="Data volume in GB", y

df <- data.frame(Method=c("Method-1"),
                 Cost=Scenario_1_cost, Data_Size = Total_Tables_size)
df = rbind(df,data.frame(Method=c("Method-2"),
                         Cost=Scenario_2_cost, Data_Size = Total_Tables_size))
df = rbind(df,data.frame(Method=c("Method-3"),
                         Cost=Scenario_3_cost, Data_Size = Total_Tables_size))
df$log_cost = log(df$Cost)

#write.csv(df,file="C:/Users/Sekhar/Documents/R Programs/Simulation/Final Proj/Writeup/Experiment_1.csv
```

Appendix-B

Code to generate experiment-2

```
##Appendix-B

rm(list=ls())
library(dplyr)
library(lpSolve)
set.seed(1)

Scenario_1_cost <- vector()
Scenario_2_cost <- vector()
Scenario_3_cost <- vector()
Total_Tables_size <- vector()
Number_of_tables <- vector()


Transfer_Speed <- 1

cnt1=1
for(cnt in seq(from=5,to=500,by=1))
{
  #Generate the list of programs
  n=500
  Program_ID <- 1:n

  #Generate the number of tables accessed by each of the program
  Total_Tables_Accessed <- rpois(lambda=3,n)

  #Generate the program slowness cost due to remote table access
  Slowness_Cost_per_unit_time <- rpois(lambda=3,n)

  #Generate the downtime cost of the program
  Down_time_cost_per_unit_time <- rpois(lambda=50,n)

  #Prepare the program, the number of tables
  #it accesses, slowness cost per unit time, downtime cost per unit time
  Programs_df = data.frame(Program_ID,
                           Total_Tables_Accessed,
                           Slowness_Cost_per_unit_time,
                           Down_time_cost_per_unit_time)


  #Generate the required number of tables
  Total_Tables = sum(Total_Tables_Accessed)


  Table_ID <- 1:Total_Tables


  #Generate the size of tables
  Size <- abs(rnorm(Total_Tables,mean=3,sd=1))
```

```
Table_df <- data.frame(Table_ID, Size)

#Generate the table's local access and remote access time
Table_df$Local_Access_Time <- Table_df$Size * 0
Table_df$Remote_Access_Time <- Table_df$Size * 3

#Now prepare the remote table access cost
Atleast_One_Tab <- Programs_df[Programs_df$Total_Tables_Accessed>0,]

Pgm_Tab_df <- data.frame()

for(i in 1:nrow(Atleast_One_Tab))
{

  x = rep(Atleast_One_Tab$Program_ID[i],Atleast_One_Tab$Total_Tables_Accessed[i])

  y = sample(1:Total_Tables,Atleast_One_Tab$Total_Tables_Accessed[i])

  z = Table_df$Remote_Access_Time[y]
  Remote_Access_Loss = z * Atleast_One_Tab$Slowness_Cost_per_unit_time[i]
  Program_Downtime_Loss_Per_Unit_Time = Atleast_One_Tab$Down_time_cost_per_unit_time[i]
  Pgm_Tab_df <- rbind(Pgm_Tab_df,data.frame(Program_ID=x,Table_ID=y,Remote_Access_Time=z,
                                            Remote_Access_Loss,Program_Downtime_Loss_Per_Unit_Time))

}

Table_Remote_Access_Cost <- Pgm_Tab_df %>%
  group_by(Table_ID) %>%
  summarise(Remote_Access_Loss=sum(Remote_Access_Loss))

temp <- Table_df[-which(Table_df$Table_ID %in%
                          Table_Remote_Access_Cost$Table_ID),]

if(nrow(temp) > 0)
{
  Table_Remote_Access_Cost <- rbind(data.frame(Table_ID=temp$Table_ID,
                                               Remote_Access_Loss=0),
                                    Table_Remote_Access_Cost)
}

Table_Remote_Access_Cost <- Table_Remote_Access_Cost[order(Table_Remote_Access_Cost$Table_ID),]

#The Table_Remote_Access_Cost data frame has the following columns:
#Table_ID - Unique identifier of the table
#Remote_Access_Loss - Loss incurred to access the table remotely by all the programs for one time

#-----------------------------------------------------
##Now generating the cost for maintaining the RI
#-----------------------------------------------------

Table_df$Number_of_Related_Tables = rpois(lambda=3,nrow(Table_df))

Table_Down_Time_Cost <- Pgm_Tab_df %>%
```

```
  group_by(Table_ID) %>%
  summarise(Pgms_Down_Time_Loss_Per_Unit_Time=
              sum(Program_Downtime_Loss_Per_Unit_Time))

temp <- Table_df[-which(Table_df$Table_ID %in%
                          Table_Down_Time_Cost$Table_ID),]

if(nrow(temp) > 0)
{
  Table_Down_Time_Cost <- rbind(data.frame(Table_ID=temp$Table_ID,
                                           Pgms_Down_Time_Loss_Per_Unit_Time=0),
                                Table_Down_Time_Cost)
}

Table_Down_Time_Cost <- Table_Down_Time_Cost[order(Table_Down_Time_Cost$Table_ID),]

#The Table_Down_Time_Cost data frame has the following columns. This df has the overall cost
#associated with a table unavailability.

#Table_ID - Table identifier
#Pgms_Down_Time_Loss_Per_Unit_Time - Loss to business due to program unavailability

Tables = nrow(Table_df)
df <- data.frame()
i=2
for(i in 1:Tables)
{
  x  <-  sample(1:Tables)
  x <- x[-which(x %in% i)]

  if(Table_df$Number_of_Related_Tables[i] > 0)
  {
    y <- sample(x,Table_df$Number_of_Related_Tables[i])
    dep_tab_sz <- Table_df$Size[y]

  }
  else{
    next
  }

  z <- rep(i,length(y))
  parent_tab_sz <- rep(Table_df$Size[i],length(y))
  df <- rbind(df,data.frame(Parent_Table=z,
                            Child_Table=y,
                            Parent_Table_Size=parent_tab_sz,
                            Child_Table_Size=dep_tab_sz))

}

df$Parent_Table_Size = ifelse(df$Parent_Table_Size == 0, 1,
                              df$Parent_Table_Size)

df$Parent_Remote_Time = df$Child_Table_Size / Transfer_Speed
```

```
df$Child_Remote_Time = df$Parent_Table_Size / Transfer_Speed

df$RI_Cost = df$Parent_Table_Size * df$Child_Table_Size * .01

Table_RI_Cost <- df

Child_Table_Remote_Time <- Table_RI_Cost %>%
  group_by(Child_Table) %>%
  summarise(Remote_Time=sum(Child_Remote_Time))

names(Child_Table_Remote_Time)[1] <- c("Table")

Parent_Table_Remote_Time <- Table_RI_Cost %>%
  group_by(Parent_Table) %>%
  summarise(Remote_Time=sum(Parent_Remote_Time))
names(Parent_Table_Remote_Time)[1] <- c("Table")

df <- rbind(Child_Table_Remote_Time,Parent_Table_Remote_Time)

df <- df %>%
  group_by(Table) %>%
  summarise(Remote_Time=sum(Remote_Time))

x = Table_df$Table_ID[-which(df$Table %in% Table_df$Table_ID)]
if(length(x) > 0){
  df = rbind(df,data.frame(Table=x,Remote_Time=0))
}

df=df[order(df$Table),]
Table_Remote_Time = df

Table_RI_Cost_Cumulative <- Table_RI_Cost %>%
  group_by(Parent_Table) %>%
  summarise(Total_RI_Cost=sum(RI_Cost),Table_Size=max(Parent_Table_Size))


temp <- Table_df[-which(Table_df$Table_ID %in% Table_RI_Cost_Cumulative$Parent_Table),]

if(nrow(temp) > 0)
{
  Table_RI_Cost_Cumulative <- rbind(data.frame(Parent_Table=temp$Table_ID,
                                               Total_RI_Cost=0,
                                               Table_Size=temp$Size),
                                    Table_RI_Cost_Cumulative)
}

Table_RI_Cost_Cumulative <- Table_RI_Cost_Cumulative[order(Table_RI_Cost_Cumulative$Parent_Table),]

#The data frame Table_RI_Cost_Cumulative has Parent table, the total RI cost for the table
#and table size. The RI cost is in dollar/sec
#Parent_Table - Parent table ID
#Total_RI_Cost - RI Cost / sec
#Table_Size - Parent table size
```

```
  #Final data sets:
  #head(Table_RI_Cost_Cumulative)
  #head(Table_Remote_Access_Cost)
  #head(Table_Down_Time_Cost)

  #head(Table_Remote_Time)



  Cost_df <- data.frame(Table_ID=Table_Remote_Access_Cost$Table_ID,
  Loss_Due_To_Program=Table_Remote_Access_Cost$Remote_Access_Loss,
  Loss_Due_To_RI = Table_RI_Cost_Cumulative$Total_RI_Cost,
  Remote_Access_Cost_Per_Unit_Time = Table_Remote_Access_Cost$Remote_Access_Loss +
    Table_RI_Cost_Cumulative$Total_RI_Cost,
  Programs_Downtime_Cost_Per_Unit_Time=Table_Down_Time_Cost$Pgms_Down_Time_Loss_Per_Unit_Time,
  Table_Size=Table_RI_Cost_Cumulative$Table_Size
  )



  Cost_df$Time_To_Transfer <- Cost_df$Table_Size/Transfer_Speed
  head(Cost_df)
  Cost_df$Total_Downtime_Cost = Cost_df$Time_To_Transfer *
    Cost_df$Programs_Downtime_Cost_Per_Unit_Time

  Cost_df$Min_Remote_Time <- Table_Remote_Time$Remote_Time

  #View(Cost_df)

  #Cost_df$Programs_Downtime_Cost_Per_Unit_Time * Cost_df$Time_To_Transfer



  Constraint = Cost_df$Min_Remote_Time

  Scenario_1_cost[cnt1] = sum(Cost_df$Programs_Downtime_Cost_Per_Unit_Time) * 17.322*60*60
  Scenario_2_cost[cnt1] = sum(Table_df$Size)* sum(Programs_df$Down_time_cost_per_unit_time)
  Scenario_3_cost[cnt1] = sum(Cost_df$Programs_Downtime_Cost_Per_Unit_Time * Cost_df$Time_To_Transfer)+
    sum(Cost_df$Remote_Access_Cost_Per_Unit_Time * Cost_df$Min_Remote_Time)
  Number_of_tables[cnt1] = Total_Tables
  Total_Tables_size[cnt1] = sum(Cost_df$Table_Size)

  cnt1 = cnt1+1
}

df = data.frame(Method_1_Cost=Scenario_1_cost,
                Method_2_Cost=Scenario_2_cost,
                Method_3_Cost=Scenario_3_cost)
write.csv(df,file="C:/Users/Sekhar/Documents/R Programs/Simulation/Final Proj/Writeup/Experiment_2.csv"

Method = c("Method-1","Method-2","Method-3")
```

```
Avg_Cost = c(mean(Scenario_1_cost),mean(Scenario_2_cost),mean(Scenario_3_cost))
SD_Cost = c(sd(Scenario_1_cost),sd(Scenario_2_cost),sd(Scenario_3_cost))
Low_95_CI = Avg_Cost - 1.96 * SD_Cost
High_95_CI = Avg_Cost + 1.96 * SD_Cost
df <- data.frame(Method,Avg_Cost,SD_Cost,Low_95_CI, High_95_CI)
```

Appendix-C

R code to run experiment 3

```
##Appendix-C

rm(list=ls())
library(dplyr)
library(lpSolve)

set.seed(1)

Scenario_1_cost <- vector()
Scenario_2_cost <- vector()
Scenario_3_cost <- vector()
Total_Tables_size <- vector()
Number_of_tables <- vector()


Transfer_Speed <- 1

cnt1=1
tab_size_temp=1
for(cnt in seq(from=5,to=200,by=1))
{

  #Generate the list of programs
  n=500
  Program_ID <- 1:n

  #Program_depth <- rpois(lambda=1,n)

  #Generate the number of tables accessed by each of the program
  Total_Tables_Accessed <- rpois(lambda=3,n)

  #Generate the program slowness cost due to remote table access
  Slowness_Cost_per_unit_time <- rpois(lambda=3,n)

  #Generate the downtime cost of the program
  Down_time_cost_per_unit_time <- rpois(lambda=50,n)

  #Prepare the program, the number of tables it accesses,
  #slowness cost per unit time, downtime cost per unit time
  Programs_df = data.frame(Program_ID,Total_Tables_Accessed,
                           Slowness_Cost_per_unit_time,Down_time_cost_per_unit_time)


  #Generate the required number of tables
  Total_Tables = sum(Total_Tables_Accessed)


  Table_ID <- 1:Total_Tables
  #Size <- rpois(lambda=1.5,Total_Tables)+0.5*abs(rnorm(Total_Tables))
```

```
#Generate the size of tables
#Size <- rpois(lambda=1.5,Total_Tables)
Size <- abs(rnorm(Total_Tables,mean=tab_size_temp,sd=1))
tab_size_temp = tab_size_temp + 1

Table_df <- data.frame(Table_ID, Size)

#Generate the table's local access and remote access time
Table_df$Local_Access_Time <- Table_df$Size * 0
Table_df$Remote_Access_Time <- Table_df$Size * 3

#Now prepare the remote table access cost
Atleast_One_Tab <- Programs_df[Programs_df$Total_Tables_Accessed>0,]

Pgm_Tab_df <- data.frame()

for(i in 1:nrow(Atleast_One_Tab))
{

  x = rep(Atleast_One_Tab$Program_ID[i],Atleast_One_Tab$Total_Tables_Accessed[i])
  #Local_Cost = rep(0,Atleast_One_Tab$Total_Tables_Accessed[i])
  #Remote_Cost = rep(3,Atleast_One_Tab$Total_Tables_Accessed[i])
  #In_Transit_Cost = rep(10,Atleast_One_Tab$Total_Tables_Accessed[i])

  y = sample(1:Total_Tables,Atleast_One_Tab$Total_Tables_Accessed[i])

  z = Table_df$Remote_Access_Time[y]
  Remote_Access_Loss = z * Atleast_One_Tab$Slowness_Cost_per_unit_time[i]
  Program_Downtime_Loss_Per_Unit_Time = Atleast_One_Tab$Down_time_cost_per_unit_time[i]
  Pgm_Tab_df <- rbind(Pgm_Tab_df,data.frame(
    Program_ID=x,Table_ID=y,Remote_Access_Time=z,
    Remote_Access_Loss,Program_Downtime_Loss_Per_Unit_Time))

}

#Display the tables list and programs data frames
#Table_df
#Programs_df

#Display the programs-tables mapping
#Pgm_Tab_df

#The above data frame has the following columns:
#Program_ID - Program ID
#Table_ID - Unique table accessed by the program
#Remote_Access_Time - Time to access the remote table
#Remote_Access_Loss - Loss incurred by the program while accessing the remote table once
#Program_Downtime_Loss_Per_Unit_Time - Loss incurred by the program due to downtime/sec



Table_Remote_Access_Cost <- Pgm_Tab_df %>%
```

```
  group_by(Table_ID) %>%
  summarise(Remote_Access_Loss=sum(Remote_Access_Loss))

temp <- Table_df[-which(Table_df$Table_ID %in%
                          Table_Remote_Access_Cost$Table_ID),]

if(nrow(temp) > 0)
{
  Table_Remote_Access_Cost <-
    rbind(data.frame(Table_ID=temp$Table_ID,
                     Remote_Access_Loss=0),
          Table_Remote_Access_Cost)
}

Table_Remote_Access_Cost <-
  Table_Remote_Access_Cost[order(Table_Remote_Access_Cost$Table_ID),]

#The Table_Remote_Access_Cost data frame has the following columns:
#Table_ID - Unique identifier of the table
#Remote_Access_Loss - Loss incurred to access the table remotely by all the programs for one time

#-----------------------------------------------------
##Now generating the cost for maintaining the RI
#-----------------------------------------------------

Table_df$Number_of_Related_Tables = rpois(lambda=3,nrow(Table_df))

Table_Down_Time_Cost <- Pgm_Tab_df %>%
  group_by(Table_ID) %>%
  summarise(Pgms_Down_Time_Loss_Per_Unit_Time=
              sum(Program_Downtime_Loss_Per_Unit_Time))

temp <- Table_df[-which(Table_df$Table_ID %in%
                          Table_Down_Time_Cost$Table_ID),]

if(nrow(temp) > 0)
{
  Table_Down_Time_Cost <- rbind(data.frame(Table_ID=temp$Table_ID,
                                           Pgms_Down_Time_Loss_Per_Unit_Time=0),
                                Table_Down_Time_Cost)
}

Table_Down_Time_Cost <- Table_Down_Time_Cost[
  order(Table_Down_Time_Cost$Table_ID),
  ]

#The Table_Down_Time_Cost data frame has the following columns. This df has the overall cost
#associated with a table unavailability.

#Table_ID - Table identifier
#Pgms_Down_Time_Loss_Per_Unit_Time - Loss to business due to program unavailability

Tables = nrow(Table_df)
```

```
df <- data.frame()
i=2
for(i in 1:Tables)
{
  x  <-  sample(1:Tables)
  x <- x[-which(x %in% i)]

  if(Table_df$Number_of_Related_Tables[i] > 0)
  {
    y <- sample(x,Table_df$Number_of_Related_Tables[i])
    dep_tab_sz <- Table_df$Size[y]

  }
  else{
    next
  }


  z <- rep(i,length(y))
  parent_tab_sz <- rep(Table_df$Size[i],length(y))
  df <- rbind(df,
              data.frame(
                Parent_Table=z,
                Child_Table=y,
                Parent_Table_Size=parent_tab_sz,
                Child_Table_Size=dep_tab_sz))

}

df$Parent_Table_Size = ifelse(df$Parent_Table_Size == 0, 1,
                              df$Parent_Table_Size)

df$Parent_Remote_Time = df$Child_Table_Size / Transfer_Speed
df$Child_Remote_Time = df$Parent_Table_Size / Transfer_Speed

df$RI_Cost = df$Parent_Table_Size * df$Child_Table_Size * .01

Table_RI_Cost <- df

Child_Table_Remote_Time <- Table_RI_Cost %>%
  group_by(Child_Table) %>%
  summarise(Remote_Time=sum(Child_Remote_Time))

names(Child_Table_Remote_Time)[1] <- c("Table")

Parent_Table_Remote_Time <- Table_RI_Cost %>%
  group_by(Parent_Table) %>%
  summarise(Remote_Time=sum(Parent_Remote_Time))
names(Parent_Table_Remote_Time)[1] <- c("Table")

df <- rbind(Child_Table_Remote_Time,Parent_Table_Remote_Time)

df <- df %>%
```

```
  group_by(Table) %>%
  summarise(Remote_Time=sum(Remote_Time))

x = Table_df$Table_ID[-which(df$Table %in% Table_df$Table_ID)]
if(length(x) > 0){
  df = rbind(df,data.frame(Table=x,Remote_Time=0))
}

df=df[order(df$Table),]
Table_Remote_Time = df

Table_RI_Cost_Cumulative <- Table_RI_Cost %>%
  group_by(Parent_Table) %>%
  summarise(Total_RI_Cost=sum(RI_Cost),Table_Size=max(Parent_Table_Size))


temp <- Table_df[-which(Table_df$Table_ID %in%
                          Table_RI_Cost_Cumulative$Parent_Table),]

if(nrow(temp) > 0)
{
  Table_RI_Cost_Cumulative <- rbind(data.frame(Parent_Table=temp$Table_ID,
                                               Total_RI_Cost=0,
                                               Table_Size=temp$Size),
                                    Table_RI_Cost_Cumulative)
}

Table_RI_Cost_Cumulative <- Table_RI_Cost_Cumulative[
  order(Table_RI_Cost_Cumulative$Parent_Table),
  ]

#The data frame Table_RI_Cost_Cumulative has Parent table, the total RI cost for the table
#and table size. The RI cost is in dollar/sec
#Parent_Table - Parent table ID
#Total_RI_Cost - RI Cost / sec
#Table_Size - Parent table size


#Final data sets:
#head(Table_RI_Cost_Cumulative)
#head(Table_Remote_Access_Cost)
#head(Table_Down_Time_Cost)

#head(Table_Remote_Time)




Cost_df <- data.frame(Table_ID=Table_Remote_Access_Cost$Table_ID,
Loss_Due_To_Program=Table_Remote_Access_Cost$Remote_Access_Loss,
Loss_Due_To_RI = Table_RI_Cost_Cumulative$Total_RI_Cost,
Remote_Access_Cost_Per_Unit_Time = Table_Remote_Access_Cost$Remote_Access_Loss +
  Table_RI_Cost_Cumulative$Total_RI_Cost,
```

```
  Programs_Downtime_Cost_Per_Unit_Time=Table_Down_Time_Cost$Pgms_Down_Time_Loss_Per_Unit_Time,
  Table_Size=Table_RI_Cost_Cumulative$Table_Size
  )



  Cost_df$Time_To_Transfer <- Cost_df$Table_Size/Transfer_Speed
  head(Cost_df)
  Cost_df$Total_Downtime_Cost = Cost_df$Time_To_Transfer *
    Cost_df$Programs_Downtime_Cost_Per_Unit_Time

  Cost_df$Min_Remote_Time <- Table_Remote_Time$Remote_Time

  #View(Cost_df)

  #Cost_df$Programs_Downtime_Cost_Per_Unit_Time * Cost_df$Time_To_Transfer



  Constraint = Cost_df$Min_Remote_Time
  #dir <- rep(">=",length(Constraint))

  #Optimization = lp(direction="min",
  #objective.in=Cost_df$Remote_Access_Cost_Per_Unit_Time,const.mat=diag(rep(1,length(Constraint))),
  #                  const.dir=dir,const.rhs=Constraint)

  #df = data.frame(Table_ID = 1:length(Constraint)
  #                , Remote_Time=Optimization$solution)
  #,Constraint)
  #df <- df[order(df$Remote_Time,decreasing=TRUE),]
  #head(df)

  Scenario_1_cost[cnt1] = sum(Cost_df$Programs_Downtime_Cost_Per_Unit_Time) * 17.322*60*60
  Scenario_2_cost[cnt1] = sum(Table_df$Size)* sum(Programs_df$Down_time_cost_per_unit_time)
  Scenario_3_cost[cnt1] = sum(Cost_df$Programs_Downtime_Cost_Per_Unit_Time *
                                Cost_df$Time_To_Transfer)+
    sum(Cost_df$Remote_Access_Cost_Per_Unit_Time * Cost_df$Min_Remote_Time)
  Number_of_tables[cnt1] = Total_Tables
  Total_Tables_size[cnt1] = sum(Cost_df$Table_Size)

  cnt1 = cnt1+1
}

library(ggplot2)
df <- data.frame(Method=c("Method-1"), Cost=Scenario_1_cost, Avg_TB_Sz = 1:196)
df = rbind(df,data.frame(Method=c("Method-2"), Cost=Scenario_2_cost, Avg_TB_Sz = 1:196))
df = rbind(df,data.frame(Method=c("Method-3"), Cost=Scenario_3_cost, Avg_TB_Sz = 1:196))

df$Cost = log(df$Cost)

ggplot(data=df,aes(x=Avg_TB_Sz,y=Cost,color=Method))+
  geom_point()+
  labs(title="Performance of 3 scenarios with \nvariying size of the table",x="Avg table size in GB", y=
```