Exploration of In-Memory Computing for Big Data Analytics using Queuing Theory

Riktesh Srivastava
Skyline University College, Sharjah
+971 6 5441155
rsrivastava@skylineuniversity.ac.ae

ABSTRACT
Assigning suitable memory chunk for Big Data analysis is posing serious problems for Business Analysts. There are plentiful solutions that came along to solve the issue of memory management. The noteworthy solutions to the problems included JVM based and Container based solutions. However, both of these solutions suffered from disk I/O bottleneck. To reduce disk, I/O bottleneck, in-memory system was introduced, which supports interactive data analytics. Present study conducts request time processing for in-memory system using three types of queue models- MG1, GM1 and GG1.

CCS Concepts
D.3.4 [Programming Languages]: Processors—Code generation, compilers, memory management, optimization, run-time environments.

D.4.2 [Operating Systems]: Storage Management—Garbage collection, main memory.

Keywords
In-Memory Computing (IMC), M/M/1 Queue, M/G/1 Queue, G/M/1 Queue, G/G/1 Queue.

1. INTRODUCTION
Moore's Law is still going strong with microprocessors development gets doubled every two years. Software developers developed algorithms collecting data (structured from database + unstructured from social media) for data analysis. The volume of data has increased so large and colossal, that was termed "big data" later. Analysts face the issue of analyzing "big data", as it takes time for server to repossess data from storage systems in cloud or hosting provider. What industry follows is termed as "distributed architecture", which stores and process "big data" in two steps- first locating the required data blocks and then loading and reading part. Data can be placed in diverse categories of memories, cache memory, main memory, and secondary memory [1], as mentioned in Figure 1.

HP3C 2018, March 15–17, 2018, Hong Kong, Hong Kong

ACM ISBN 978-1-4503-6337-2/18/03…$15.00
https://doi.org/10.1145/3195612.3195621

These "distributed architecture" algorithms use the concept of amassing the data from secondary storage and process them by provisionally storing them in main memory. The biggest problem in these algorithm was lag between secondary and main memory data shifting. This lag has forced companies to develop the application code to maximize the use of main memory and minimize the access to secondary storage [2]. Using IMC solves another problem of memory usage, as it was found that enterprise storage system uses only a small portion of available memory they buy, as Microsoft found that 85% of the memory in there data center is free all the time [2].

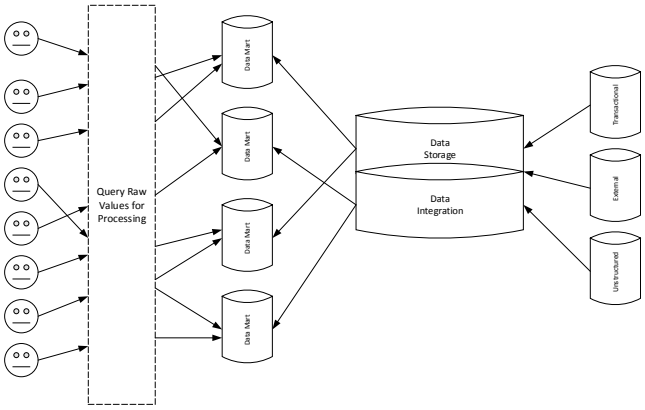

Figure 1: Traditional memory usage for "big data"

These "distributed architecture" algorithms use the concept of amassing the data from secondary storage and process them by provisionally storing them in main memory. The biggest problem in these algorithm was lag between secondary and main memory data shifting. This lag has forced companies to develop the application code to maximize the use of main memory and minimize the access to secondary storage [2]. Using IMC solves another problem of memory usage, as it was found that enterprise storage system uses only a small portion of available memory they buy, as Microsoft found that 85% of the memory in there data center is free all the time [2]. It is due to this reason following companies have started reconnoitering the usage of IMC

1) Facebook newsfeed uses IMC for data fetch [3]
2) Tagged.com, a social networking site, always retrieve data from memory tier only [2]
3) SAP HANA only access the non-volatile memory [4]
4) Oracle has also started combining DRAM and flash memories into "memory tier" [5]

Collecting and reading data from secondary memory and main memory is not an ideal approach for Big Data and other high performance applications concerning limited I/O bandwidth [6]. It is due to this reason In-Memory Computing (IMC) is used in Big Data, wherein, entire data (both structured and unstructured) is picked from secondary memory into cache memory, and the entire speed can be increased to 10x, 20x or even 100x [7]. Furthermore, study conducted by [8], [9], states that there are three key problems while accessing data through secondary and main memory, skills, integration and security. IMC resolves the three glitches by placing the entire data to be processed into cache memory. Skills are now vital to manage data sources only, problem of integration and security does not even occur, as main data sources still are in secondary memory. Also businesses dealing in Big Data benefits from cost savings, simplicity and efficient data processing and improved visibility through better business decisions [10]. The architecture adopted for IMC is given in Figure 2 below:

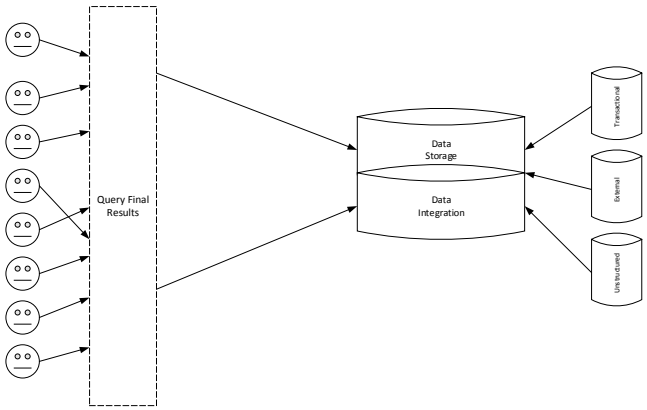

Figure 2: IMC for Big Data

The processing time of data processing is evaluated based on architecture using queuing theory. Using queuing theory to store and refer large data sets does not take large amount of memory [11]. The rest of paper is divided as follows: Section 2 depicts the concepts of IMC using Columnstore indexes for smaller datasets. Section 3 extrapolates the outcomes for higher number of records using Gauss Elimination and back substitution techniques. Section 4 computes the results. Section 5 provides conclusion.

2. IN-MEMORY COMPUTATION USING COLUMNSTORE INDEXES

For Big Data Analysis, the customary methodology is to extract the data based on row based index. When the size of data includes million rows and predictive analytics needs to be performed, row based index fails. In this study, In-Memory Computation IMC is implemented using column based index, also called "Columnstore indexes".

Arrivals of random data sets in IMC is λ and the rate at which the processed data departs from IMC is μ. Mathematically, both arrivals and departures are function of time, as mentioned in equations (1) and (2)

$$\lambda = \frac{\sum_{i=1}^{n} a_i}{T} \quad (1)$$

$$\mu = \frac{\sum_{i=1}^{n} d_i}{T} \quad (2)$$

For estimation of probability of n data sets to arrive at IMC, three assumptions are made:

1. At any given time λ and μ are number of data sets to be processed.
2. There may be no requests at IMC, depicting no requests departs from IMC.
3. Also, there may be only one large dataset at IMC, thus, there is only one response departs from IMC.

Based on above mentioned assumptions:

- Probability of one arrival = $\lambda.\Delta t$
- Probability of one departure= $\mu.\Delta t$
- Also, probability of no arrival of dataset = $1 - \lambda.\Delta t$
- Thus, the probability of no departure from IMC= $1 - \mu.\Delta t$

For processing, if we increase the time from t to $t + \Delta t$, we get

$$P_n(t+\Delta t) = \begin{cases} P_n(t)(1-\lambda\Delta t)(1-\mu\Delta t) \\ P_{n+1}(t)(\mu\Delta t) \\ P_{n-1}(t)(\lambda\Delta t) \end{cases} \quad (3)$$

Arranging the conditions in equation (3), we get

$P_n(t+\Delta t) = P_n(t)(1 - \lambda.\Delta t)(1-\mu\Delta t) + P_{n-1}(t)\lambda\Delta t + P_{n+1}(t)\mu\Delta t$ (4)

or,

$$\frac{P_0(t+\Delta t) - P_0(t)}{\Delta t} = -\lambda P_n(t) - \mu P_n(t) + \lambda P_{n-1}(t) + \mu P_{n+1}(t) \quad (5)$$

But,

$$\lim_{\Delta t \to 0}\left\{\frac{P_n(t+\Delta t) - P_n(t)}{\Delta t}\right\} = \frac{d}{dt}\{P_n(t) = 0\}\ for\ stable\ condition$$

Thus, the R.H.S. of equation (5) becomes

$P_{n-1}(t)\lambda - (\lambda+\mu)P_n(t) + P_{n+1}(t)\mu = 0$ (6)

To solve equation (6), it is assumed that there were 0 requests at time $t + \Delta t$. This can be obtained from the states as given under:

Thus, L.H.S. of equation (7) becomes

$$\lim_{\Delta t \to 0}\left\{\frac{P_n(t+\Delta t) - P_n(t)}{\Delta t}\right\} \quad (7)$$

$$= \frac{d}{dt}\{P_0(t) = 0\}\ for\ stable\ condition \quad (8)$$

Hence equation (8) becomes

$$P_1(t) = \left(\frac{\lambda}{\mu}\right)P_0(t) \quad (9)$$

From equations (6) and (8), the following can be derived as:

$$P_0(t) = \left(\frac{\lambda}{\mu}\right)^0 P_0(t)$$

$$P_2(t) = \left(\frac{\lambda}{\mu}\right)^1 P_0(t)$$

$$P_2(t) = \left(\frac{\lambda}{\mu}\right)^2 P_0(t)$$

:

:

$$P_n(t) = \left(\frac{\lambda}{\mu}\right)^n P_0(t) \quad (10)$$

Summation of all the equations:

$$\sum_{i=0}^{n} P_i(t) = \left\{\left(\frac{\lambda}{\mu}\right)^0 + \left(\frac{\lambda}{\mu}\right)^1 + \left(\frac{\lambda}{\mu}\right)^2 + \cdots + \left(\frac{\lambda}{\mu}\right)^n\right\}P_0(t)$$

(11)

Based on limiting condition, when $n \to \infty$ and $\frac{\lambda}{\mu} < 1$, L.H.S. becomes 1 and R.H.S. becomes

$\left[\frac{1}{\left(1-\frac{\lambda}{\mu}\right)}\right] P_o(t)$

Thus equation (10) becomes

$$1 = \left[\frac{1}{\left(1-\frac{\lambda}{\mu}\right)}\right] P_o(t) \quad (12)$$

Substituting equation (12) in equation (11), we get

$$P_n(t) = \left(\frac{\lambda}{\mu}\right)^n \left(1-\frac{\lambda}{\mu}\right) \quad (13)$$

From equation (13), the probability of n data at IMC any given time is evaluated.

In order to calculate the estimated size of in-memory for a given data set so that no data is left out is given as:

$$IMC(n) = \sum_{n\to\infty}^{N} n.P_n(t) = \sum_{n\to\infty}^{N} \left(\frac{\lambda}{\mu}\right)^n \left(1-\frac{\lambda}{\mu}\right) \quad (14)$$

From equation (13), the average memory size for IMC can be evaluated as mentioned in equation (15) as:

$$IMC(n) = \frac{\left(\lambda/\mu\right)}{\left(1-\lambda/\mu\right)} \quad (15)$$

Flowchart given in Figure 3 expands the broad implementation process.

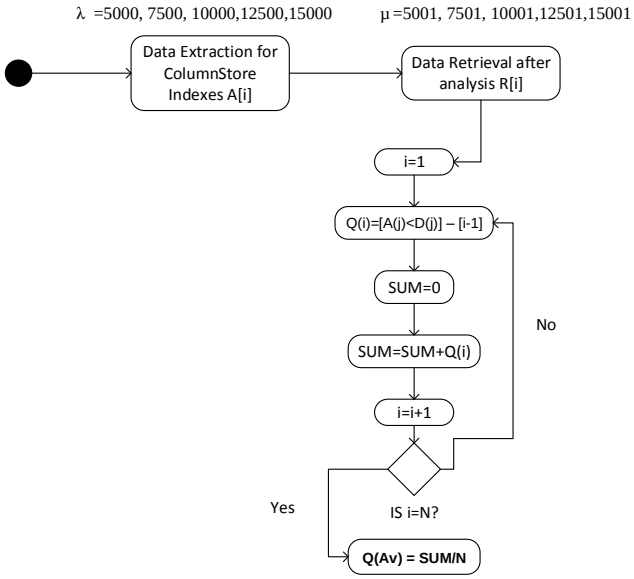

Figure. 3. Columnstore execution via Markovian Distribution

2.1 Results for smaller datasets (using Markovian Distribution)

The computational results using the equation (15) are given in table 1.

Table 1: Computation results

λ	μ	MM1	MG1	GM1	GG1
5000	5001	5000	5102	5098	5538
7500	7501	7500	7653	7647	8400
10000	10001	10000	10204	10194	11190
12500	12501	12500	12755	12740	14126
15000	15001	15000	15306	15283	17695

Notice that the memory size is calculated for lower magnitude of data sets. Also, the value of μ is evaluated as λ +1 (considering to be worst case scenario for effective data management). To find the memory size for higher values of data set, multinomial regression technique is used, which extrapolates the average $IMC(n)$. If we plot the values obtained for effective memory size for all four queue models, we get the following as shown in Figure 4:

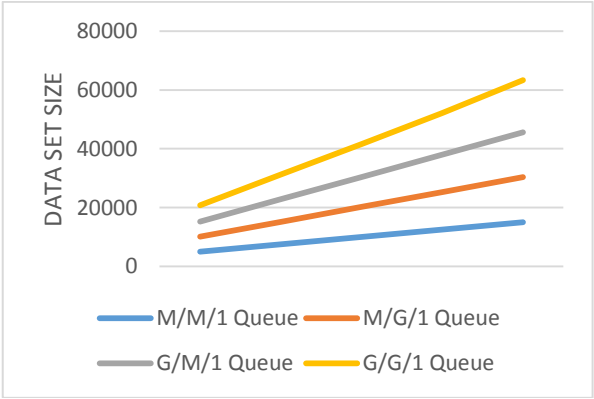

Figure. 4. Graphical Representation

Close observation from Figure 4 gives MM1 queue as a linear equation with slope of unity. However, other queue models [MG1, GM1 and GG1] have curves.

These curves are smooth and assumed to be second order polynomial. The equations of $IMC(n)$ for all the three models are denoted as:

$$y_i = a_2x_i^2 + a_1x_i + a_0$$
$$y_i = b_2x_i^2 + b_1x_i + b_0$$
$$y_i = c_2x_i^2 + c_1x_i + c_0$$

Notice $a_2, a_1, and\ a_0$ are coefficients of polynomial for MG1, $b_2, b_1, and\ b_0$ are coefficients of polynomial for GM1 and $c_2, c_1, and\ c_0$ are coefficients of polynomial for GG1 queue model. Let S represents the error in computation and real values of memory size, then S is represented as

$S = \sum\left(y_i - a_2x_i^2 + a_1x_i + a_0\right)^2$ for MG1 model

$S = \sum\left(y_i - b_2x_i^2 + b_1x_i + b_0\right)^2$ for GM1 model

$S = \sum\left(y_i - c_2x_i^2 + c_1x_i + c_0\right)^2$ for GG1 model

2.2 Mathematical Assessment of General Distribution

For general distribution, two types of distribution [12] (Bernoulli and Gaussian distribution) are used $IMC(n)$ is evaluated as average of these two distributions.

2.2.1 Bernoulli Distribution

Bernoulli Distribution equation can be given as:

$$f\left(y(i)\right) = a + by(i)\ for\ i\epsilon 1\ to\ N$$

Equiprobable Distribution $= x(i)$

Bernoulli Distribution$= y(i)$

then,

$$y(i)_{BD} = -a \pm \sqrt{a^2 + \frac{2bx(i)}{b}} \quad (16)$$

where, $b = \frac{1}{\lambda}$ for arrival of data sets

$b = \frac{1}{\mu}$ for departure

$a = 1 - b$

2.2.2 Geometric Distribution

Geometric Distribution equation can be given as:

$$\boldsymbol{f(y(i)) = \frac{a}{(1 - by(i))} \; for \; i \epsilon 1 \; to \; N}$$

Equiprobable Distribution = $x(i)$

Geometric Distribution= $y(i)$

then,

$$y(i)_{GD} = \frac{1}{b(1 - e^{-\delta x(i)})} \quad (17)$$

where,

$\delta = \frac{b}{a}$, $b = 1 - a$ and $b = \frac{\lambda}{\mu}$ for both arrival and departure

From equation (16) and (17), $IMC(n)$ is given as

$$IMC(n) = Avg\left[\left(-a \pm \sqrt{a^2 + \frac{2bx(i)}{b}}\right) + \left(\frac{1}{b(1 - e^{-\delta x(i)})}\right)\right] \quad (18)$$

Figure 5 given below mentions the step wise evaluation for General distribution arrival and distribution.

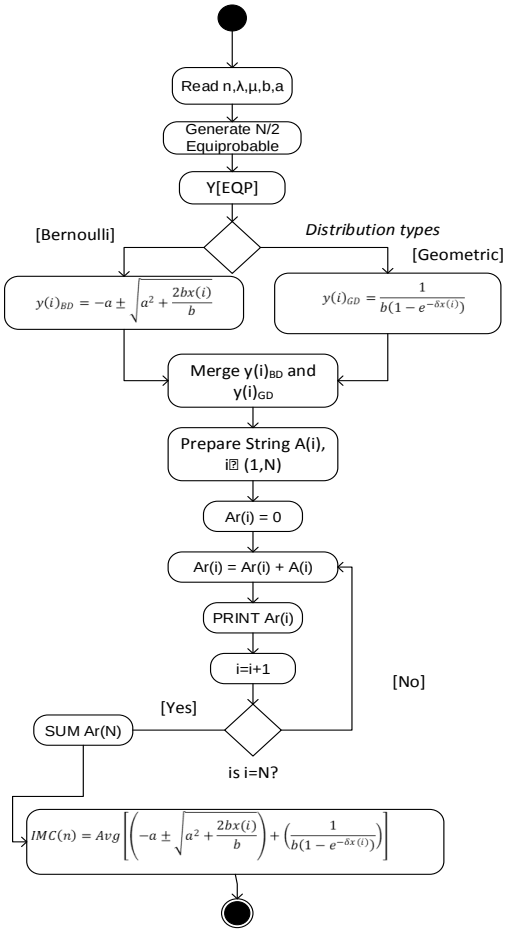

Figure 5. General Distribution Evaluation

EXTRAPOLATION OF RESULTS

In order to extrapolate the results for higher order Columnstore index, we need to look into curves from Figure 4. It is observed that these curves are smooth and assumed to be second order polynomial. The equations of $IMC(n)$ for all the three models are denoted as:

$$y_i = a_2 x_i^2 + a_1 x_i + a_0$$
$$y_i = b_2 x_i^2 + b_1 x_i + b_0$$
$$y_i = c_2 x_i^2 + c_1 x_i + c_0$$

Notice $a_2, a_1, and\ a_0$ are coefficients of polynomial for MG1, $b_2, b_1, and\ b_0$ are coefficients of polynomial for GM1 and $c_2, c_1, and\ c_0$ are coefficients of polynomial for GG1 queue model. Let S represents the error in computation and real values of memory size, then S is represented as

$S = \sum\left(y_i - a_2 x_i^2 + a_1 x_i + a_0\right)^2$ for MG1 model

$S = \sum\left(y_i - b_2 x_i^2 + b_1 x_i + b_0\right)^2$ for GM1 model

$S = \sum\left(y_i - c_2 x_i^2 + c_1 x_i + c_0\right)^2$ for GG1 model

Differentiating S w.r.t a_2, a_1, a_0 and setting each of these coefficients to 0, we get following equations for MG1 queue.

$$na_0 + a_1 \sum x_i + a_2 \sum x_i^2 = \sum y_i$$
$$a_0 \sum x_i + a_1 \sum x_i^2 + a_2 \sum x_i^3 = \sum x_i y_i$$
$$a_0 \sum x_i^2 + a_1 \sum x_i^3 + a_2 \sum x_i^4 = \sum x_i^2 y_i$$

Similarly, we get the following equations for coefficients for GM1 queue

$$nb_0 + b_1 \sum x_i + b_2 \sum x_i^2 = \sum y_i$$
$$b_0 \sum x_i + b_1 \sum x_i^2 + b_2 \sum x_i^3 = \sum x_i y_i$$
$$b_0 \sum x_i^2 + b_1 \sum x_i^3 + b_2 \sum x_i^4 = \sum x_i^2 y_i$$

And, coefficients for GG1 queue,

$$nc_0 + c_1 \sum x_i + c_2 \sum x_i^2 = \sum y_i$$
$$c_0 \sum x_i + c_1 \sum x_i^2 + c_2 \sum x_i^3 = \sum x_i y_i$$
$$c_0 \sum x_i^2 + c_1 \sum x_i^3 + c_2 \sum x_i^4 = \sum x_i^2 y_i$$

For the size of data set as mentioned in Table 1, we can compute the equations to find the value of coefficients. The results of computation are depicted in Table 2 below:

Table 2: Computation of results

COMPUTATION RESULTS PARAMETERS	MG1	GM1	GG1
$\sum x_i$	50000	50000	50000
$\sum y_i = Q_i$	50962	51020	56949
$\sum x_i^2$	562500000	562500000	562500000
$\sum x_i^3$	6875000000000	6875000000000	6875000000000
$\sum x_i^4$	88828125000000000	88828125000000000	88828125000000000
$\sum x_i y_i$	573277500	573975000	644590000
$\sum x_i^2 y_i$	7006293750000	7015250000000	7918512500000

2.3 Normal Equations

By taking the values as computed and observed from Table 2, the normal equations and values of coefficients for MG1, GM1 and GG1 Queues can be easily computed.

2.3.1 Normal equations for MG1 queue

Substituting the values from Table 2, we get the following equations for MG queue

$5a_0 + 50000a_1 + 562500000a_2 = 51020$

$50000a_0 + 562500000a_1 + 6875000000000a_2 = 573975000$

$562500000a_0 + 6875000000000a_1 + 88281250000000000a_2 = 573975000$

By Gaussian elimination and using back substitution, we obtain:

$a_0 = 1.0089\ X\ 10^{-7}$, $a_1 = 1.001$, $a_2 = 1.68\ X\ 10^{-6}$

2.3.2 Normal equations for GM1 queue

Placing the values from Table 2, we get the following equations for GM1 queue

$5b_0 + 50000b_1 + 562500000b_2 = 50962$

$50000b_0 + 562500000b_1 + 6875000000000b_2 = 573277500$

$562500000b_0 + 6875000000000b_1 + 88281250000000000b_2 = 7006293750000$

By Gaussian elimination and using back substitution, we obtain:

$b_0 = 2.001\ X\ 10^{-7}$, $b_1 = 1.02149142857$,

$b_2 = 1.885714285\ X\ 10^{-6}$

2.3.3 Normal equations for GG1 queue

Placing the values from Table 2, we get the following equations for GG1 queue

$5c_0 + 50000c_1 + 562500000c_2 = 56949$

$50000c_0 + 562500000c_1 + 6875000000000c_2 = 644590000$

$562500000c_0 + 6875000000000c_1 + 88281250000000000c_2 = 79185125000000$

Using Gaussian elimination and using back substitution, we obtain:

$c_0 = 2.009\ X\ 10^{-7}$, $c_1 = 0.845028571$

$c_2 = 1.7828571\ X\ 10^{-5}$

3. CALCULATION OF IN-MEMORY SIZE

The algorithm is executed with arrival rates in range of 5000 to 15000 for MG1, GM1 and GG1 models. The curve equations of the model which passes through the points of rate of arrival and $IMC(n)$are determined for these models. The equations have been considered polynomial of second order. The first order polynomial fits in MM1 model only. The second order polynomial usually be part of parabola. The regression technique to determine the coefficients of polynomials of second order are calculated. These equations are used to analytically calculate the size of memory at any high rate of the model. The model which offers largest Queue Length is used to decide the memory size for Big Data Analysis. Memory size is decided as sum of Average value and its Standard deviation. The entire work is summarized in this section.

Based on calculation of coefficients for three types of queues, the average $IMC(n)$are denoted as:

$y(i)_{GMI} = 1.68\ X\ 10^{-6}x_i^2 + 1.001x_i + 1.009\ X\ 10^{-7}$

$y(i)_{MGI} = 1.89\ X\ 10^{-6}x_i^2 + 1.021x_i + 2.001\ X\ 10^{-7}$

$y(i)_{GGI} = 1.79\ X\ 10^{-5}x_i^2 + 0.854x_i + 2.009\ X\ 10^{-7}$

In all the equations, $y(i)$ is in-memory size and x_i is value as derived in equations (15) and (18) respectively.

From the study, it was observed that optimal value of in-memory $IMC(n)_{optimal} = IMC(n)_{average} + S.D.of\ IMC(n)$

In case of Markovian arrival, $S.D.of\ IMC(n)$ is given as

$S.D.of\ IMC(n) = IMC(n)_{average}$

Thus, the optimal value of $IMC(n)$ is

$$IMC(n)_{optimal} = 2\ X\ IMC(n)_{average}$$

The calculated in-memory size is depicted in Table 3 below:

Table 2: in-Memory Size (in MB)

Data Set Size	Optimal in-Memory Size (in MB)		
	MG1	GM1	GG1
10^7	356020000	397572686	3582614771
10^8	33800200000	37918583986	356740425714
10^9	3362002000000	3773471554257	35658832057142
10^{10}	336020020000000	377163286828571	3565731100571420

4. CONCLUSION

The problem of deciding the size of memory for Big Data Analysis is extremely complex, especially, when attempted using row level indexing. This was the reason of selecting Columnstore indexing for evaluation of results. Presence of NA's and different format of data makes the study even more complex and model was studied using random Columnstore size. This was the reason of selecting queue models to decide the size of memory and functioning of Big Data Analysis using Columnstore indexes. There are two possibilities:

- Either dataset have some distribution.
- or, Dataset doesn't have any discipline for the distribution.

The present study has been carried out for both disciplines and indiscipline's. Disciplined data arrival or departure has been studied under MM1. Analytical and Stimulation study are possible to decide the memory size for processing. This study for the same is carried out and observed that general distribution is not possible to be studied analytically, due to presence of large degree of freedom of distribution. Hence, this has been studied combining 2 distributions and giving the name, general distributions. These distributions were merged and applied to three queue models, MG1, GG1 and GG1.

The processing and departure of results from dataset, termed as departure rate, μ. Columnstore data extraction and processing were targeted at $\mu > \lambda$, which makes the entire data analysis extremely stable. This state is also termed as "Ergodic functioning for Big Data Analysis". At any time, if it is observed, that extraction rate is greater than the processing and departing rate, the processing was stopped were stopped and restarted (though this situation happened only twice).

For the given experiment, MG1 model provided the best results as given in Table 3. MG1 memory required the least memory size for the same amount of data to be processed. GG1 model used similar amount of memory for less number of data, but the performance decreases for large datasets. The experiment will be further conducted for real-time dataset environment as an extension to the result.

5. REFERENCES

[1] T. Raynaud, R. Haque, and H. A ́-kaci, “CedCom: A High-Performance Architecture for Big Data applications,” 2014 IEEEACS 11th Int. Conf. Comput. Syst. Appl. AICCSA.

[2] E. Savitz, “IT Revolution: How In Memory Computing Changes Everything,” Forbes. [Online]. Available: https://www.forbes.com/sites/ciocentral/2013/03/08/it-revolution-how-in-memory-computing-changes-everything/. [Accessed: 10-01-2018].

[3] B. J. Says, “The Myth Of In-Memory Computing,” The Next Platform, 19-Feb-2016. [Online]. Available: https://www.nextplatform.com/2016/02/19/the-myth-of-in-memory-computing/. [Accessed: 21-Jan-2018].

[4] Mihnea Andrei et.al, "SAP HANA Adoption of Non-Volatile Memory", The Proceedings of the VLDB Endowment (PVLDB) pp: 1754 [Access on 19-01-2018]

[5] Oracle Report, “New Oracle Exadata X7 Delivers In-Memory Performance from Shared Storage.” [Online]. Available: https://www.oracle.com/corporate/pressrelease/oow17-oracle-exadata-x7-100217.html. [Accessed: 12-12-2017].

[6] B. Roussey, “How in-memory computing helps enterprises overcome Big Data woes,” How in-memory computing helps enterprises overcome big data woes, 05-May-2017. [Online]. Available: http://techgenix.com/in-memory-computing/. [Accessed: 29-Dec-2017].

[7] D. Gutierrez, “insideBIGDATA Guide to In-Memory Computing,” insideBIGDATA, 25-Sep-2014. .

[8] P. Jeffcock, “3 Key Problems To Solve If You Want A Big Data Management System,” 2014. [Online]. Available: https://blogs.oracle.com/bigdata/3-key-problems-to-solve-if-you-want-a-big-data-management-system. [Accessed: 09-Dec-2017].

[9] N. Khan et al., “Big Data: Survey, Technologies, Opportunities, and Challenges,” Sci. World J., vol. 2014, 2014.

[10] SAP AG Report, Harnessing the Power of Big Data in Real Time through In-Memory Technology and Analytics, The Global Information Technology Report 2012, World Economic Forum, 2012, pp:88-96, [Accessed on 10-12-2017]

[11] “Memory Management for Large Data Sets - LabVIEW 2017 Help - National Instruments,” Memory Management for Large Data Sets, 01-Mar-2017. [Online]. Available: http://zone.ni.com/reference/en-XX/help/371361P-01/lvconcepts/memory_management_for_large_data_sets/. [Accessed: 29-Dec-2017].

[12] R. Srivastava, “Estimation of Buffer Size of Internet Gateway Server via G/M/1 Queuing Model,” Int. J. Comput. Inf. Eng., vol. 1, no. 9, pp. 2667–2675, 2007.