

APPLICATIONS OF THE LITTLE'S LAW IN INFORMATICS AND COMPUTER PERFORMANCE MODELLING FOR EXTRA PROCESSING IN CLOUD DATABASE SYSTEMS RESULTED FROM INTRODUCING NEW EUROPEAN BANKING REGULATIONS

Adriana-Nicoleta TALPEANU

Bucharest University of Economic Studies

Florin-Catalin ENACHE

Bucharest University of Economic Studies

ABSTRACT

This paper surveys the contributions and applications of queuing theory in the field of banking data networks.

The Little's Law mathematical definitions of formula for a specific λ arrival rate, and the average service time will be described, used and confirmed by computer simulations of real queues usually found in the banking computing systems.

The goal is to provide sufficient information to computer performance analysts who are interested in using the queueing theory to model a network of banking computer systems using the right simulation model applied in real-life scenarios, e.g. overcoming the negative impacts of the European banking regulations while moving towards green computing.

Keywords: Little's Law, queueing theory, banking system, computer network, economical informatics, banking regulation, computer simulation, Basel

JEL classification: C02, C15, C61

INTRODUCTION

With the global advancement seen in the last 20 years, especially with the increased volume, complexity, spread of exchanges in the economic and financial relations, all the computing systems, but especially the banking systems, had to adapt fast not only their banking regulations [16] [18] to the continuous changing world [12] but also their networking field, which had changed drastically over the time. Perhaps, the most fundamental change has been the rapid development of optical fiber technology. This has created limitless opportunities for new digital networks with greatly improved capabilities. The current broadband integrated service networks that provide integrated data, voice and video seem to have almost nothing in common with the

data networks of the last 20 years, but in fact, many of the underlying principles, mathematical and statistical laws are the same.

2. Little's Theorem [2]

The little's Theorem was first published in 1961 and can be stated as $L = \lambda'W$, where L stands for length or queue length, λ' stands for the arrival rate and W stands for the waiting time. Actually this is translated to the fact that the expected queue length is equal to the arrival rate multiplied by the expected waiting time. This is very neat because it connects an average of a discrete random variable (L) and an average of a continuous random variable(W). From a practical point of view, it turns out that it is often relatively easy to calculate the average queue length and from that, by dividing by λ' the average waiting times can be easily calculated. We can definitely go back and forth between the 2 averages, but it is obvious that L is usually easier to calculate because it is a discrete variable, while W is a continuous variable.

The basic idea is that for the system represented in Fig.1, we have some entities/customers who arrive and enter the system at rate λ' - entry rate (noted lambda prime in order to distinguish between the arrival rate and the entry rate). The arrival rate is the rate at which the customers seek to enter the system, but in general they do not necessarily enter. When a customer enters the system, it stays there for an average length of time called W (stands for waiting time) and while it is there waiting for its lifetime to expire, on average there is L of them present. The target is to find the relationship between the average amount of time that a customer spends in the system and the number of customers in the systems having their lives spent.

Little's original proof was quite difficult and made some assumptions that turned out to be not necessary, and over the years this has been refined and generalized over and over again and in fact there are many paper now that have to do with Little's theorem and its consequences. Although originally assumptions were made that were essentially probabilistic or stochastic about the stochastic processes involved, it turned out that the it is really a deterministic rule and that the stochastic part had nothing to do with it. This underlines the robustness of the formula, because it is going to hold regardless of whether these arrivals occur according to a Poisson process, and regardless of the service times distribution. Practically the only essential requirement is that the L, W and λ' must exist. But as long as they exist, then the formula is going to hold. An intuitive argument that quickly explains the formula. Imagining that the system is observed over a long period of time T and that as the customers enter, they bring with them their lifetime, and that these lifetimes build-up, it is obvious that these must equal to the lifetimes that are used-up.

Over a certain period of time t , the expected number of arrivals that enter the system would be $\lambda't$ and each of those customers follows a blueprint of entering the system, staying there for its lifetime (in average W), therefore the total amount of lifetime brought into the system over a certain period of time t is equal to $\lambda'tW$.

On the other hand, the amount of used-up lifetime during the interval of time of length t must be evaluated. On average, there are L people in that system and their lifetimes are expiring in parallel, at the rate of one unit of life per unit of time, so therefore, in time t , if we had 1 person present, the amount of lifetime that would expire would be t , consequently, for L people would be Lt .

On the long run, the boundaries of the initial condition regarding t will be irrelevant, and t can be cancelled out since it is a non-negative value. Mathematically, this can be written as: $\lambda'tW = Lt$ which is equivalent to $\lambda'W = L$.

The beauty of this is that the definition of system is very flexible and the particular type of the stochastic processes involved are irrelevant, making the Little's theorem very powerful and general.

3.Using Little's Theorem to forecast a database server's performance

3.1) IO Performance indicators

The Oracle database RDBMS is one of the best software example in which performance indicators are very good documented. For the purpose of this article, we will take into consideration only performance indicators related to storage subsystems

```
SELECT pname,pval1 FROM aux_stats$ WHERE sname='SYSSTATS_MAIN';
```

PNAME	PVAL1
CPUSPEED	836
MAXTHR	992766976
MBRC	53
MREADTIM	24
SLAVETHR	8995840

In the above extract, the MBRC is the effective multi block read count during the collection interval. The example shows a high-end system. The SREADTIM(single block read time in milliseconds) of 6 ms is not very fast, but the MAXTHR(maximum throughput) is about 1GB/s.

When speaking about the I/O Statistics, the number of blocks read and written is counted by the performance statistics “physical reads” and “physical writes”.

The performance statistic “physical reads” includes: physical reads cache, physical reads direct, physical reads direct temporary table space, physical reads direct (lob).

The performance statistic “physical writes” includes: physical writes from cache, physical writes direct, physical writes direct temporary table space, physical writes direct (lob).

All statistics count the number of Oracle blocks read or written and not to hard disk blocks.

The Oracle Database documents also the wait events related to I/O. the most common “System I/O” wait events are :

- ‘log file parallel write’ (LGWR writing redo records from the log buffer to the redo log files)
- ‘log file sequential read’ (ARCH reads redo log files for archiving)
- ‘db file parallel write’ (DBWR is performing a write to files)
- ‘control file parallel write’, ‘control file sequential read’ (usually performed by the DBWR, LGWR, CKPT and ARCH database processes)
- ‘io done’ (on platforms that do not support asynchronous I/O a background process waits for an I/O to complete or it waits for a slave process to become available to submit an I/O request)
- ‘RMAN backup & recovery I/O’

Computation of Peak I/O

For predicting the peak I/O, elements of queueing theory like the M/M/1 queue [4] are needed. The M/M/1 queue successfully describes an I/O subsystem e.g. HDDs, RAIDs, Logical Volumes, Volume groups. Each of these are a separate M/M/1 queue, and each one of them has different response-times and throughput. For a scalable model, variation must be taken into consideration. When speaking about computer performance, a multiple CPU queue is managed as a M/M/n queue, where n is the number of CPU cores. This can be understood as a new-style

airport queue, with multiple counters serving one queue, while any I/O subsystem queue is always a single server queue.

Fortunately, the UNIX and Linux systems have a multitude of utilities for measuring performance of I/O subsystems. In Fig.1 an overview is presented.

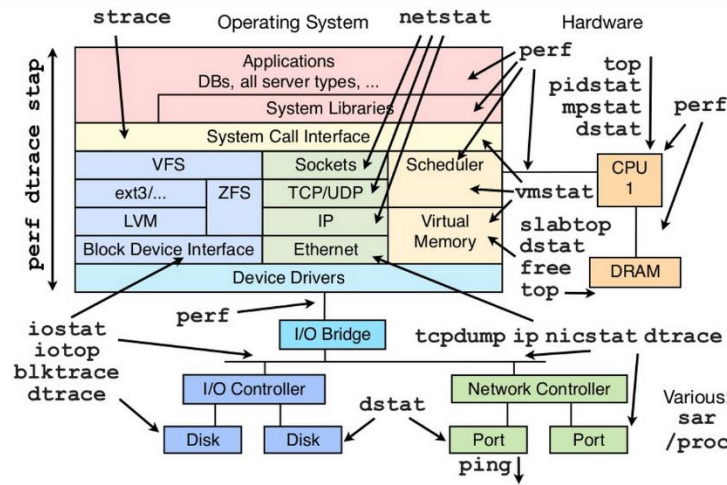


Figure 1. overview of tool for analysing IO performance

One of the best examples is given by the following output of the command `iostat -x`. The `-x` option display extended statistics. This option works with post 2.5 kernels since it needs `/proc/disk stats` file or a mounted sysfs to get the statistics.

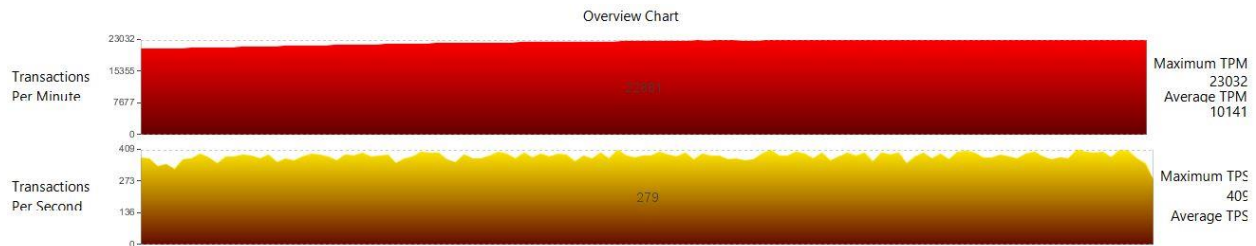
```
iostat -x
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.12    0.00    0.89    1.82    0.00   97.16

Device:            rrqm/s   wrqm/s     r/s     w/s    rsec/s   wsec/s  %util
sdc                 8.84   4721.32    3.68    38.62    30.92   4760.23   9.92
sda                 1.54     0.98   26.69     1.48   754.86    21.17   1.70
sdb                 8.87   6392.50    3.68    52.23    30.95   6445.29   9.42
```

In the above example, the `iostat` utility is used. Although the utility provides a big amount of data, extracting only the data that needs to be interpreted is the key factor for a proper and correct forecast of performance. The data will be extracted in a snapshot-based way, collecting it

at precise, repeating time intervals. For collecting the data, a database is needed, where the performance data will be stored.

The following image is a snapshot from the swingbench utility, showing a maximum of 23032 transactions per minute, respectively 409 transactions per second.



Using an oracle database, the following io_stat2 table will be created containing:

- The name of the IO device,
- The number of read requests that were issued to the device per second from the r/s column,
- The number of write requests that were issued to the device per second from the w/s column
- Percentage of CPU time during which I/O requests were issued to the device from the %utilcolumn(bandwidth utilization for the device). Device saturation occurs when this value is close to 100%.
- block changes as value from the database and
- redo writes as value from the database.

The following code illustrates the structure of the io_stat2 table and the routine that is collecting performance data during the swingbench workload every second.

```
CREATE TABLE io_stat2(snap_time DATE,  
  
io_device VARCHAR2(20),  
  
io_req_read NUMBER,  
  
io_req_write NUMBER,  
  
io_util NUMBER,  
  
block_changes NUMBER,  
  
redo_wr NUMBER);
```

Loading sequence :

```
while true; do  
  
iostat -x 1 2 |\  

```

```

sed 1,13d | sed 3d |\
awk '{ printf("%s %s %s %s\n", $1, $4, $5, $12); }' |\
while read DEVICE R_SEC W_SEC UTIL; do
$ORACLE_HOME/bin/sqlplus -s system/oracle <<EOF
VARIABLE v_bl_changes NUMBER;
VARIABLE v_redo_size NUMBER;
BEGIN
SELECT value INTO :v_bl_changes FROM v\$_sysmetric WHERE metric_name ='DB Block Changes Per Sec'
AND group_id = 3;
SELECT value INTO :v_redo_size FROM v\$_sysmetric WHERE metric_name ='Redo Writes Per Sec' AND
group_id = 3;
INSERT INTO io_stat2 VALUES (sysdate,'$DEVICE',$R_SEC,$W_SEC,$UTIL,:v_bl_changes,:v_redo_wr);
COMMIT;
END;
/
EXIT;
EOF

```

The above code sequence collect data in the io_stat2 table. Using simple SQL queries against the io_stat2 table. For example, finding the average required reads/writes and the IO utilization for every IO subsystem, the following is used:

```
SQL> select IO_DEVICE, avg(IO_REQ_READ),avg(IO_REQ_WRITE) , avg(IO_UTIL) from io_stat2 group by IO_DEVICE;
```

IO_DEVICE	AVG(IO_REQ_READ)	AVG(IO_REQ_WRITE)	AVG(IO_UTIL)
sda	53.6677117	782.275275	19.1228947
sdb	62.7722998	762.438776	19.8460984

Using the same approach, other important performance data can be found. Since the data files respectively the redo logs were placed on separate IO devices, the same workload can help in accurately forecasting the performance of both IO components.

```
SQL> select IO_DEVICE "DF Drive" ,AVG(BLOCK_CHANGES) from io_stat2 where IO_DEVICE='sdb' group by IO_DEVICE;
```

```
DF Drive          AVG(BLOCK_CHANGES)
-----
sdb              3031.32136

SQL> select IO_DEVICE "REDO Drive" ,AVG(REDO_SIZE) from io_stat2 where IO_DEVICE='sda' group by
IO_DEVICE;

REDO Drive       AVG(REDO_SIZE)
-----
sda              65.9950912
```

4. Analyzing the data

The question being answered by the Little’s Law and Queueing Theory in generalis how would a specific system behave when the workload increases by 10%, 20% or more. For the above case the performance of each of the IO devices will be calculated, but also a weighted performance value.

Weighted average service time for all devices is calculated as the arrival rate weighted average for all devices. For example,

$$T_s = \frac{\sum_{i=1}^n (\lambda_i T_{s_i})}{\sum_{i=1}^n \lambda_i} = \frac{(782.275275 * 0.02444522) + (762.438776 * 0.02602976)}{782.275275 + 762.438776} = 0.02522731$$

describe

s the weighted average service time that differs from the unweighted average service time of $= (0.02444522 + 0.02444522) / 2 = 0.02523749$ for the following dataset, but making it a more realistic values as it is closer to reality :

Device	Arrival rate	Utilization%	ServiceTime
/dev/sda	782.275275 (r/s+w/s)*	19.1228947	0.02444522
/dev/sdb	762.438776 (r/s+w/s)*	19.8460984	0.02444522

*arrival rate calculated as read + write requests.

According to the Little [4] formula, the answer time is calculated as the sum of the service time and queueing wait time ($T = T_s + T_w$), where the queueing wait time is a function of the service

time (easily measured) and utilization (reported by most of the OS utilities, e.g. iostat) $T_w = T_s / (1 - U)$.

In the following table, once the first line is obtained (corresponding to 100% of the workload) using the above mentioned formulas, it is only a matter of increasing the workload, respectively the arrival rate by 10%, 20% and so on until the breaking point of the IO device.

	A	B	C	D
1	ServiceTime	0.0252273		
2				
3	Workload	Arrival Rate	Utilization	AnswerTime
4	100%	772.35703	19.54	0.05658116
5	110%	849.59273	21.43295	0.05733658
6	120%	926.82843	23.3814	0.05815314
7	130%	1004.0641	25.32985	0.05901231
8	140%	1081.2998	27.2783	0.05991752
41	470%	3630.078	91.57713	0.32473711
42	480%	3707.3137	93.52558	0.41487338
43	490%	3784.5494	95.47403	0.58261782
44	500%	3861.7851	97.42248	1.00397188
45	510%	3939.0208	99.37093	4.03549746

Figure 2. Answer times and breaking point for the IO subsystem

In Fig. 2 it can be seen that increasing the workload by 10%, 20%, up to 490% increases the response time, but there is a clear breaking point around 505%, where the response time gets extremely high from one iteration to the next, making the IO device practically impossible to use. The graphic of this simulation will always look as in the fig. xxx, almost linearly until the spike.

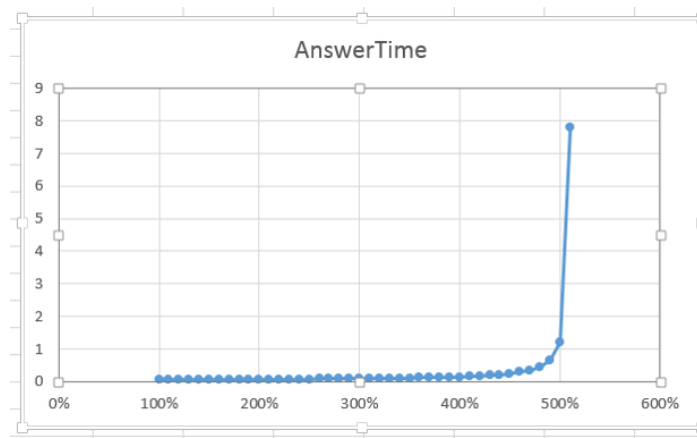


Figure 3. Graphical interpretation of the answer times as load increases

Conclusions

Such simulations help a lot in answering questions like:

- Will the system support the 20% increase in reporting that the European Bank is imposing?
- There is a trend of 5% more processing each year. What is the best moment to do a hardware refresh, without coming to a hardware failure?

There is also a clear recommendation to have a performance Data Warehouse where snapshot data is stored. This will help both in analysing performance issues and in forecasting the performance of the whole system.

Moreover, such a simulation gives insight on how such a queue would behave as a result of different arrival processes and service times. Further, we consider that it offers a methodology for looking into more complicated cases not only like getting input times from a network of banking systems trying to implement a new set of banking regulations where a mathematical approach cannot help, but also in other complex areas.

References

- [1] Campbell-Kelly, M., &Aspray, I. (2004). *Computer: A History Of The Information Machine (Sloan Technology)*. Westview Press
- [2] Cooper, R. B. (1981). *Introduction to Queueing Theory, Second Edition*. New York: North Holland, New York.
- [3] Dardac, N., Moinescu, B. (2007). *Politicimonetareșitehnicibancare*. Course notes.
- [4] Enache, F.C. (2014). *Stochastic Processes and Queueing Theory for Cloud Computer Performance Analysis*, In: Conference Proceedings of the 14th International Conference on Informatics in Economy, pp13-19.
- [5] Ghahramani, S. (2005). *Fundamentals of Probability with Stochastic Processes, Third Edition*. Upper Saddle River, Pearson Prentice Hall, New Jersey.
- [6] Goldthwaite, R. A. (1995). *Banks, Places and Entrepreneurs in Renaissance Florence: 492 (Variorum Collected Studies)*, Variorum.
- [7] Hoggson, N. F. (1926). *Banking Through the Ages*. Dodd, Mead & Company.
- [8] Lakatos, L. (2008). *A note on the Pollaczek-Khinchin Formula*. In: Annales Univ. Sci. Budapest, Sect. Comp. 29, pp. 83-91.

- [9] Sigman, K.. (2011). *Exact Simulation of the stationary distribution of the FIFO M/G/c Queue*. J. Appl. Spec. Vol. 48A, pp. 209-213, <<http://www.columbia.edu/~ks20/papers/QUESTA-KS-Exact.pdf>>. [January 20, 2015].
- [10] Sigman, K.. (2010). *Inverse Transform Method*, <<http://www.columbia.edu/~ks20/4404-Sigman/4404-Notes-ITM.pdf>>. [January 15, 2015].
- [11] Tarullo, D. K.. (2008). *Banking on Basel: The Future of International Financial Regulation*. Peterson Institute for International Economics, U.S.A.
- [12] Van Dillen, J.G. (1964). *History of the principal public banks*. Frank Cass & CO LTD.
- [13] Von Neumann, J. (1945). First Draft of a Report on the EDVAC. *Contract No. W-670-ORD-4926 between the United States Army Ordnance Department and the University of Pennsylvania*.
- [14] Wolff, R. W. (1981). *Poisson arrivals see time averages*, <<http://www2.isye.gatech.edu/~spyros/courses/IE7201/Fall-13/PASTA-proof.pdf>>. [August 14, 2015].
- [15] ***, (2015). *Bank for International Settlements- Basel Committee on Banking Supervision*, <<http://www.bis.org/bcbs/>>. [August 10, 2015].
- [16] ***, (2015). *Erlang B Calculator*, < <http://home.earthlink.net/~malcolmhamer/Erlang-B.xls>>. [August 14, 2015].
- [17]***, (2010). *The European Environment - State and Outlook 2010*. European Environmental Agency.
- [18] ***, (2015). *European Banking Authority*, <www.eba.europa.eu>. [September 02, 2015].
- [19] ***, (2015). *Kendall's notations*, <https://www.andrewferrier.com/oldpages/queueing_theory/Andy/kendall.html>. [August 10, 2015].