ISSN: 2836-8495

Research Article

Journal of Current Trends in Computer Science Research

Increased Resource Utilization Through Sharing as Analyzed Using a Random Walk with an Absorbing Barrier

Nicholas A. Strakhov*

Lexington, University of Kentucky, United States

***Corresponding Author**
Nicholas A. Strakhov, Lexington, University of Kentucky, United States.

Submitted: 2023, Dec 26; **Accepted:** 2024, Jan 15; **Published:** 2024, Feb 05

Citation: Strakhov, N. A. (2024). Increased Resource Utilization Through Sharing as Analyzed Using a Random Walk with an Absorbing Barrier. *J Curr Trends Comp Sci Res, 3*(1), 01-10.

Abstract

Intuition supports the idea that if multiple entities use items to service their clients, then sharing the items between the entities will result in more provided services than if the items are pre-allocated to each entity. This idea is quantified by first creating a grid of two to five dimensions (one dimension for each entity). The number of items in use is represented by a single point whose projection on each axis is determined by the number of items in use by each entity. Moving from point to point on this grid occurs as each entity needs an additional item or gives one up. This movement is similar to a Random Walk. The probabilities of moving forward or backward are each independently chosen to represent customer usage. An absorbing barrier enforces the finite number of items available. The results of a computer program simulating this model is presented which shows improved resource utilization. Next, a detailed analysis is developed and the formula obtained is simple and accurate in predicting the results.over the range computed by the simulation.

Index Terms-Sharing resources, multi-dimensional random walk, trinomial expansion

1. Introduction

This paper studies the utilization of an inventory of items that is to be shared by two or more entities (or servers). A basic assumption is that the inventory is not consumed by the entity but is returned to inventory after use. There are two possible approaches to a system for implementing the
sharing process:

• The inventory is divided initially with each participating entity receiving an inventory for its exclusive use. This approach can result in some entities running out of inventory while others have an excess. This approach will be called preallocated.

• Another approach is to maintain the inventory as one common pool and allow each entity to use the next available item. This will be called shared.

The analysis to be presented will demonstrate that the approach to sharing the inventory utilizes the inventory better unless the pre-allocated approach is divided accurately in proportion to the demand.

1.1. Some Examples

Recently there was a strong demand for ventilators to treat patients seriously ill with COVID-19. At times some medical facilities were running out of ventilators while others had some excess. Clearly a central stockpile of ventilators might have helped provide ventilators to where they could be used more effectively.

Outdoor dining has become more popular in part by the need for more ventilation in the era of COVID-19 and other pandemics. In some cases, if restaurants have contiguous outdoor dining areas, it might be possible to share the dining tables. More diners could be served and fewer placed on a waiting list with shared facility dining.

An example in an industrial setting might be that as transmission network demands change, specialized equipment may be temporarily required in different parts of the network. Again, sharing may be an important consideration.

Notice that in all these cases the inventoried item is returned. In the case of ventilators, when a patient is better or dies the ventilator is available for use elsewhere. In the case of outdoor dining, when the diners are done, the table is free for use by another author. No Artificial Intelligence software was used in any phase. Nicholas Strakhov is retired from Telcordia Technologies. He was previously associated with Bell core and Bell Telephone Laboratories. He may be reached at home at nstrakho@alum.mit.edu. diners. Similarly, network equipment can be reused when no longer needed in the current installation.

1.2. Background

There are a large number of studies focusing on sharing resources. See, for example, [1-3]. One notable difference is that these studies focus on consumable resources such as raw

materials for several producers. Several interesting concepts in linear algebra theory are applied to the solution. Another approach is presented by which considers two servers sharing an input stream and applying queuing theory to the service time [4]. Again, this model does not address returning an item to a common pool.

As will be presented shortly, the concept of Random Walks (RW) will be utilized. There is an even larger body of knowledge applied to random walks. See, for example, which are introductions to the topic [5,6]. A major difference is the statistics in this study are much different as will be seen. Yet another approach that might be applied to this study is described by [7]. This analyzes how plants compete for limited air, water and soil. Again, however, the inventory is being consumed by the plants.

2. Simulation

2.1. Inventory Model

The first model of the pool will represent two entities and will be displayed on a two- dimensional graph. The X axis represents the number of items in use by server 2 and the Y axis represents the same for server 1. See Figure 1.

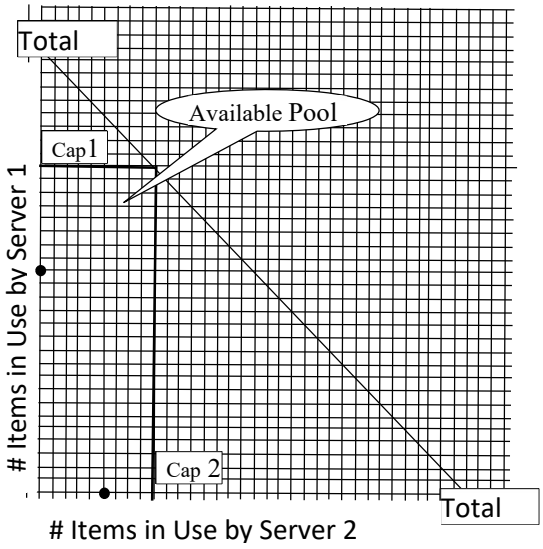


Figure 1: Pre-Allocated Inventory

This chart represents the condition where the inventory has been divided between the two servers (Cap1 for Server 1 and Cap2 for server 2). Cap1 plus Cap2 cannot exceed the 45° line which represents the total available inventory. The two black doughnuts depict actual items in use. In effect this model creates two independent RW.

The next chart shows how the points on the X and Y axes move about.

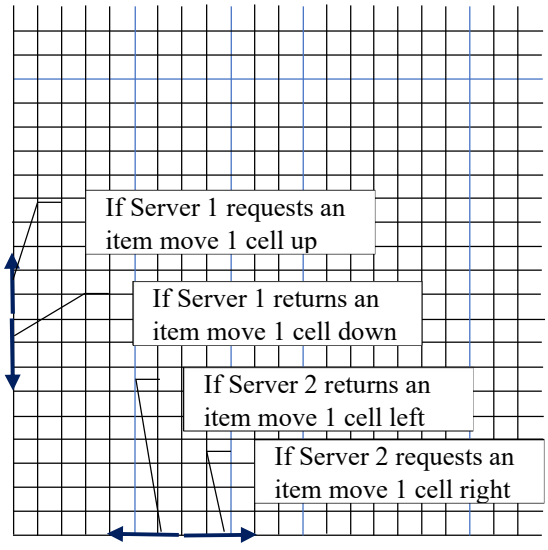


Figure 2: Two One Dimensional Random Walks

The two points (one on each axis) represent the amount of inventory each Server is using. They are not allowed to go beyond the capacity assigned to each one. As noted on Figure 2 this will be modeled with two independent random walks.

The foregoing described the basic model behind two servers that

are using pre-allocated inventory. Next, the generalization of the model to address two servers sharing the inventory is presented. The model now becomes:

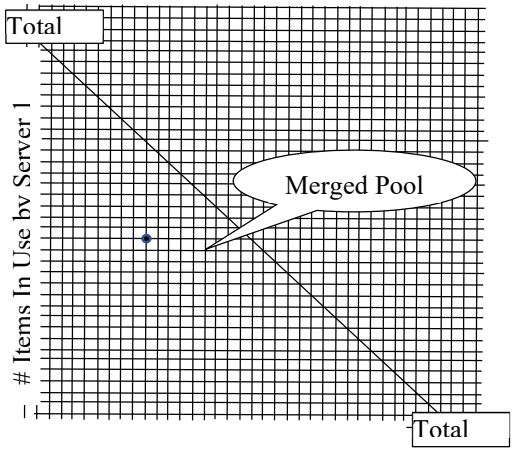
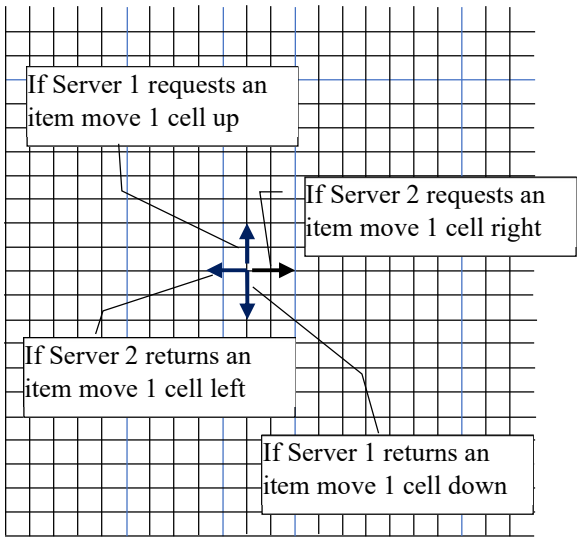


Figure 3: Two Dimensional RW

The Current Operating Point (black doughnut) represent the number of items in use by each Server. The next chart depicts how this point moves as conditions change.



Creates Two-Dimensional Random Walk

Figure 4: Walk RW Movement Rules

Extending this model to more servers is straightforward: simply add a dimension for each server in the collection. There is still only one point in the multidimensional grid that represents the number of customers being served.

2.2. Simulation Program

The probability of Server_i requiring an additional item (moving to the next higher cell) is designated pi and the probability of returning an item to its pool is
q_i. (i = 1,2...Number of servers).

In most random walk analyses, $p_i + q_i = 1$ and furthermore pi = qi = ½. This latter requirement assures that the operating point stays near the origin. Otherwise, the location of the operating point will grow without limit [5,8].

In this analysis, pi is initially greater than qi but as Server_i accumulates more items, pi decreases. Also, if the number of items held by all Servers attempts to increase above the sum of all capacities, then no additional items can be assigned to any server until the number held falls below the capacity sum.

The probabilities that will be used in this analysis are:

- The probability of moving to the next higher cell will be given by $p_i = p_{0i}e^{-(Mi/Ri)}$ (Eq 1)

where p_{0i} is a constant between 0 and 1, Mi is the current item count for server i, and Ri is a constant that represents the demand for service.

- q_i is set to q_0 where q_0 is a constant between 0 and 1. Clearly there could be other choices for p_i and q_i
- If pi is true and qi is false the cell advances. If pi is false and qi is true the cell moves back. Otherwise, the cell stays put but the

cycle is still counted.
The simulation program generates the following output:
• Each server is evaluated separately with their own pre-allocated pool of items.
• All servers are evaluated collectively with the merged pool of items.
• The evaluation consists of the average location of the items and the percent of attempts to cross the boundary
• The number of repeated cycles is a program input but is chosen to be 200,000 for the examples presented.
• The evaluation uses a range of demand for service or a range of capacity. In this study a range of demand is used.2
• Further analysis of the program's output is achieved through a download to an Excel spreadsheet. A key function is to combine the pre-allocated overflow with the formula

$$\frac{1}{TD}\sum_{i=1}^{S} Ri * OFi$$

where Ri is the demand for the i-th server, OFi is the corresponding overflow and TD is the sum of all Ri's.
Also, the number served by each preallocated server is combined by adding the individual equilibrium values. These last two steps allow for comparison of pre-allocated results with merged results.

It turns out that some simplification in the number of variables that need to be considered is achieved by dividing the Demand (R) by the Capacity and also dividing the average location by Capacity. All results will be presented in these terms.

2.3. Pre-Allocation to Merged Comparison

In this case three different values of demand and capacity with the same R/Cap ratio are presented. In addition, pre-allocated and merged results are presented with the same parameters. This uses two servers.

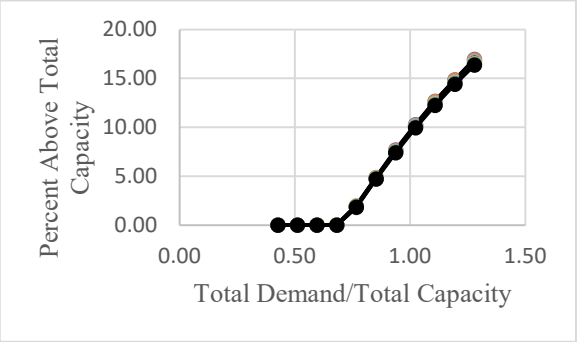


Figure 5: Percent Over - Balanced Inventory Allocation at Three Scales

This chart behaves as expected. For small values of demand, there is little to no overflow over capacity. As the demand increases, the probability for the RW to move forward increases, and more overflow is experienced.

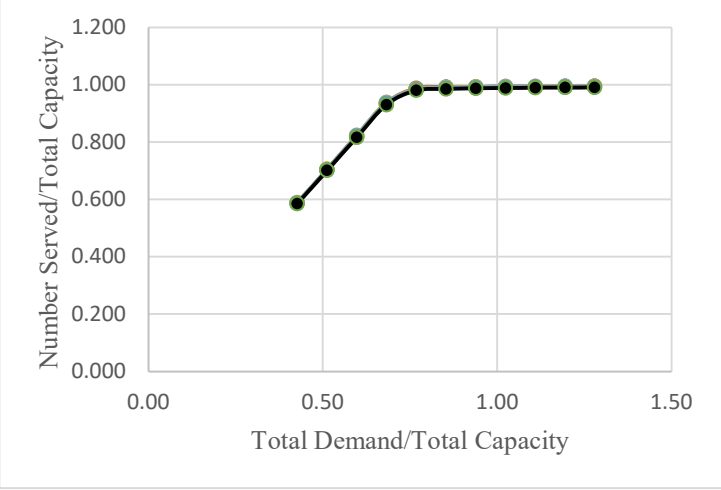


Figure 6: Number Served - Balanced Inventory Allocation at Three Scales

All six cases plot over each other in this case as well. The average position of the items used is taken to be the number served.

This chart also behaves as expected. As the demand increases from a small starting value, the number served linearly increases. As the Capacity of the inventory is reached, the number served/ Capacity approaches 1 as there is no additional inventory to offer.

Two conclusions can be drawn from these results:
• Using the ratio of demand/capacity is a useful procedure since the results are insensitive to their absolute value.
• The pre-allocated and merged inventory case provide equal results when the ratio of demand to capacity is equally proportioned and the pi and qi values are the same.
The next two charts show the effect of an imbalance. In this case the capacity was equally divided but the demand was not which

resulted in a ratio of
Demand/Capacity of 74% for Server 1 and 97% for Server 2

(starting value) - n*(starting value)/10 and five other values equal
to (starting value) + n*(starting value)/10 for n=1,2…5.

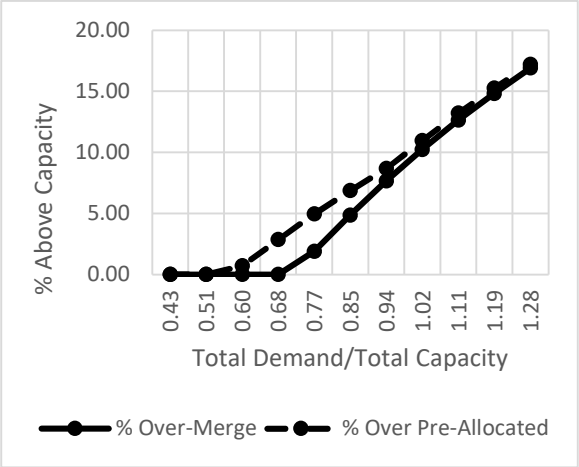


Figure 7: Unbalanced Inventory Pre-allocation

Comparing the Balanced (Figure 5) to the Unbalanced (Figure 7) percent overage at a Demand/Capacity value of 0.77:
• In both cases the overage is about 2% for the merged simulation.
• The balanced simulation (Figure 5) for the pre-allocated is also 2% overage.
• The unbalanced simulation (Figure 7) for the pre-allocated (dashed curve) is about 5% overage.

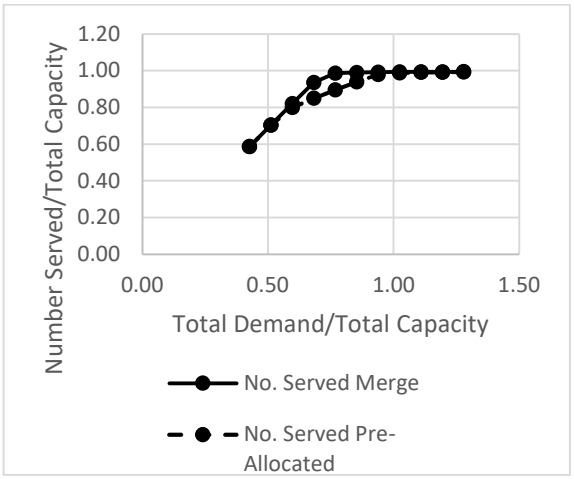


Figure 8: Number Served - Unbalanced Inventory Pre-Allocation

This chart shows the number served. Note that at a Demand/Capacity level of 0.77 the unbalanced case shows 10% fewer served (dashed curve) than the merged case (solid curve).
The next chart shows the difference in number served for this case.

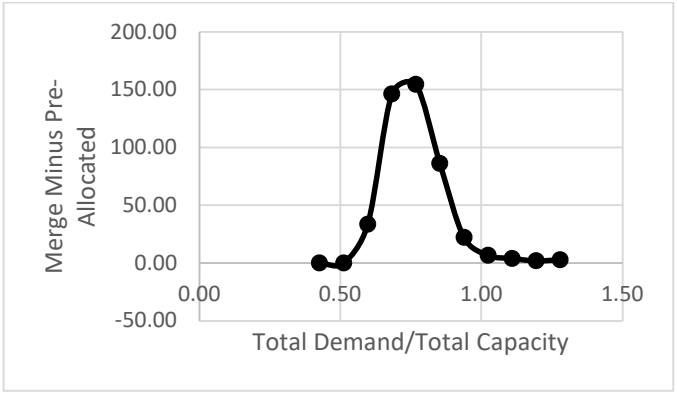


Figure 9: Merged Minus Unbalanced Inventory PreAllocation – Number Served

Note that for low levels of Total Demand/Total Capacity (< 0.6), the difference is essentially zero. This is because in both cases the numbers being served are not affected by their respective capacities. For large levels of TD/TC, the demand claims all possible inventory so the difference is small (about two extra served if merged). In between there are 50 to 150 extra served in the merged case.

Of course, there are many different ways to create an unbalance. The probabilities poi and qi can be changed for different servers. Also, the number of servers can be increased. However, all results follow the same pattern so they are not presented here.

3. Theoretical Results

3.1. Number Served

When the random walk reaches equilibrium, it seems reasonable to assume that this will occur when $p_1 = q_1$. Accordingly, since $p_1 e^{-(M1/R1)} = q_1$, $M_1 = R_1 \ln(p_1/q_1)$ as long as $M_1 \leq Cap1$ as R1 increases. For larger values of R_1, $M_1 = Cap_1$.

A spread sheet can now be created to predict the extra services provided by merging. Generalizing the expression for M1 results in:

1. $M_i = R_i \ln(p_i/q_i)$ where i = 1,2,3,.. up to number of servers
2. For the pre-allocated estimate calculate $\sum_{i=1}^{No.Severs} Mi$ where $M_i = R_1 \ln(p_1/q_1)$ but if $M_i > Cap_i$ use Cap_i for M_i.
3. For the merged estimate calculate $\sum_{i=1}^{No.Severs} Mi$ where $M_i = R_1 \ln(p_1/q_1)$ but if $\sum_{i=1}^{No.Servers} Mi > \sum_{i=1}^{No.Servers} Capi$, then set all M_i to Cap_i.
4. Subtract result obtained in step 2 from the result of step 3.

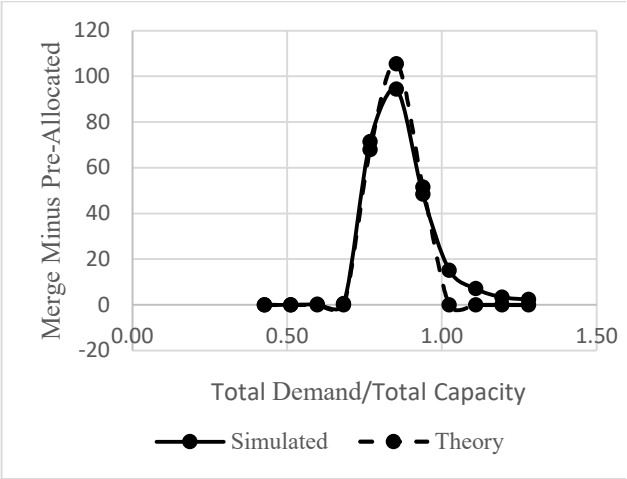


Figure 10: Comparing Number Served: Simulation vs Theory

B. Overflow

The general approach employed to derive equations for the overflow is as follows:

1. Let $p_i + q_i + ni = 1$ where ni is the probability that there will be no change in location for Server_i

2. Calculate probability of all of the paths of length N for each server (N >> 1).

3. To minimize the number of paths to consider, all paths start at capacity in the calculation. For a large number of cycles this should not be significant.

4. Note that a necessary condition for the RW to exceed capacity is for the following steps to occur within one cycle: sum of all forward steps across all servers minus the sum of all backward steps across all servers where the first step is at capacity. Whenever the RW exceeds capacity, its location is reset to be on the capacity.

5.The calculation will compute the expected value of a long string of steps along the barrier. The probability of each step is governed by the three probabilities for each RW.

6. For a single server the expected value of a Random Variable (RV) is given by

$$E(RV) = \sum_{(all\ i,j,k)} RV(x_{i,j,k}) P(x_{i,j,k})$$

7. The starting point for this analysis is the trinomial expansion given by $(p + q + n)^N = 1$

$$= \sum_{i+j+k=N} \left(\frac{N!}{i!\,j!\,k!}\right) p^i\, q^j\, n^k$$

8. Convert to expected value where (Th) stands for Threshold: E(Th)=

$$\frac{1}{N} \sum_{i+j+k=N} Th(Path_{N,i,j,k}) \left(\frac{N!}{i!\,j!\,k!}\right) p^i\, q^j\, n^k$$

where $Th(path_{N,i,j,k})$ is the value of $path_{N,i,j,k}$.

9. Calculation of Th for each path becomes thenumber of p_1's minus the number of q_1's.

For one Server this is i, (the number of p_1's), minus j, (the number of q_1's), or (i - j).

Substituting that value into the expression for E(Th) results in

$$E(Th) = \frac{1}{N} \sum_{i+j+k=N} (i-j) \left(\frac{N!}{i!\,j!\,k!}\right) p^i\, q^j\, n^k$$

Simplifying the expression yields

$$E(Th) = \frac{1}{N} \sum_{i+j+k=N} (i) \left(\frac{N!}{i!\,j!\,k!}\right) p^i\, q^j\, n^k - \frac{1}{N} \sum_{i+j+k=N} (j) \left(\frac{N!}{i!\,j!\,k!}\right) p^i\, q^j\, n^k$$

$$= p \sum_{i+j+k=N} \left(\frac{N!}{i!\,j!\,k!}\right) p^i\, q^j\, n^k - q \sum_{i+j+k=N} \left(\frac{N!}{i!\,j!\,k!}\right) p^i\, q^j\, n^k$$

$$= p - q \qquad \text{(Eq 2)}$$

This summation is a well-known result [9,5,10]. For a single server, the values for p and q are determined from prior expressions (See Eq 1). The next Figure shows the comparison between the formula and the simulation.

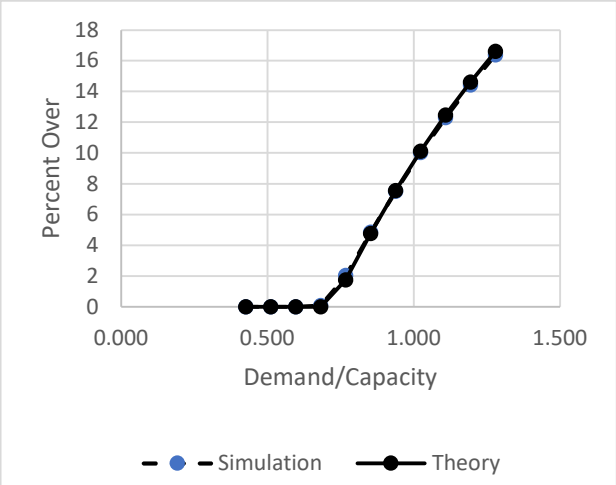


Figure 11: Comparison of Theory with Simulation – Single Server

As can be seen on this chart the calculated value for E(Th) is almost identical with the simulation result.

Extending this result to more servers is straightforward but the subscript notation becomes unwieldy. For example, for three servers:

$$E(Th)=\frac{1}{N}\sum_{l+m+n=N}\sum_{i+j+k=N}\sum_{s+t+u=N} Th(Path_{N,i,j,k,l,m,n,s,t,r})\ C(N,p_1,q_1)\ C(N,p_2,q_2)\ C(N,p_3,q_3) \quad \text{(Eq 3)}$$

Where the expression for C() is given by

$$C(N,p_g,q_g) = \left(\frac{N!}{\alpha!\,\beta!\,\chi!}\right) p_g^{\alpha}\, q_g^{\beta}\, n_g^{\chi}$$

and g = 1, 2 or 3 and $n_g = 1 - p_g - q_g$

For g = 1, (α,β,χ) = (l,m,n)

For g = 2, (α,β,χ) = (i,j,k)

For g = 3, (α,β,χ) = (s,t,u)

The expected value is now the number of p1's minus the number of q2's minus the number of q3's. To this add the number of p2's minus the number of q1's minus the number of q3's and so forth. This table details all the cases:

Server	Number of p_x's	Number of q_x's
1	l	par(j) + par(t)
2	i	par(m) + par(t)
3	s	par(m) + par(j)

Where par() stands for partial and par(x) + par(x) = x and x=j,m,t

The net positive minus negative result is
l - par(j) – par(t)

i – par(m) – par(t)

s – par(m) – par(j)

Summing these terms gives:

$$Th(Path_{N,i,j,k,l,m,n,s,t,r}) = l + i + s - j - t - m$$

And substituting this result into (Eq 3) results in

$$E(Th) = (p_1 + p_2 + p_3) - (q_1 + q_2 + q_3)$$
$$E(Th)\text{ per server} = \frac{1}{3}(p_1 + p_2 + p_3) - \frac{1}{3}(q_1 + q_2 + q_3)$$

This process can be extended to more servers with the following result:

$$E(Th)\text{ per server} = \frac{1}{S}(p_1 + p_2 + \ldots + p_S) - \frac{1}{S}(q_1 + q_2 + \ldots + q_S) \quad \text{where S = No. of Servers} \quad \text{(Eq 4)}$$

The values for pi and qi cannot be determined from the original expressions (See Eq 1). This is because the values of Mi depend on where along the capacity boundary the boundary is exceeded. Returning to Figures 3 and 4, the emphasis here is to determine the pairwise side-to-side position along the boundary by equating the probability of moving to the upper left to the probability of moving to the lower right. This results in : $p_1q_3=p_3q_1$ $p_1q_2=p_2q_1$ $p_3q_2=p_2q_3$

Substituting for p_i and q_i (See Eq 1) and solving for M2 and M3 gives:

$$M3/R3 = L_{3,1} + M1/R1$$

$$M2/R2 = L_{2,3} + M3/R3$$

where $L_{i,j} = \ln[(p_{0i}/q_i)*(q_j/p_{0j})]^3$ (Eq 5)

Also defining TC and TD as:

M1+M2+M3=TC (total capacity)

R1+R2+R3 =TD (total demand)

Placing this in matrix notation results in:

$$\begin{bmatrix} TC \\ L(3,2) \\ L(1,3) \end{bmatrix} = \begin{bmatrix} R1 & R2 & R3 \\ 0 & -1 & 1 \\ 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} M1/R1 \\ M2/R2 \\ M3/R3 \end{bmatrix}$$

Solving for the vector Mi/Ri by inverting the matrix [11] gives

$$\begin{bmatrix} M1/R1 \\ M2/R2 \\ M3/R3 \end{bmatrix} = \frac{1}{TD} * \begin{bmatrix} 1 & R2 & R2+R3 \\ 1 & -(R1+R3) & -R1 \\ 1 & R2 & -R1 \end{bmatrix} * \begin{bmatrix} TC \\ L(3,2) \\ L(1,3) \end{bmatrix}$$

After multiplying and refactoring the following formula is obtained:

$$\begin{bmatrix} \frac{M1}{R1} \\ \frac{M2}{R2} \\ \frac{M3}{R3} \end{bmatrix} = \begin{bmatrix} \frac{TC}{TD} \\ \frac{TC}{TD} \\ \frac{TC}{TD} \end{bmatrix} + \begin{bmatrix} L(1,1) & L(1,2) & L(1,3) \\ L(2,1) & L(2,2) & L(2,3) \\ L(3,1) & L(3,2) & L(3,3) \end{bmatrix} \begin{bmatrix} \frac{R1}{TD} \\ \frac{R2}{TD} \\ \frac{R3}{TD} \end{bmatrix}$$

(Eq 6)

Although this is the result for three servers, it is easily extended to more servers.

An Excel spread sheet was created to perform the following calculations in sequence:

- Expand Eq 6 to a 5 x 5 matrix
- Calculate the L(i,j) coefficients using Eq 5
- Calculate Mi/Ri from Ri/TD using Eq 6
- Determine $p_i = p_{0i}\exp(-Mi/Ri)$
- Combine them using Eq 4 (with S=5)

The largest number of servers in the simulation program was five so that is the basis of comparison.

The five servers have the following data

Server	p_{0i}	q_i	Demand	Capacity
1	0.8	0.3	550	645
2	0.8	0.3	900	1055
3	0.7	0.3	550	640
4	0.7	0.2	700	750
5	0.7	0.2	300	320

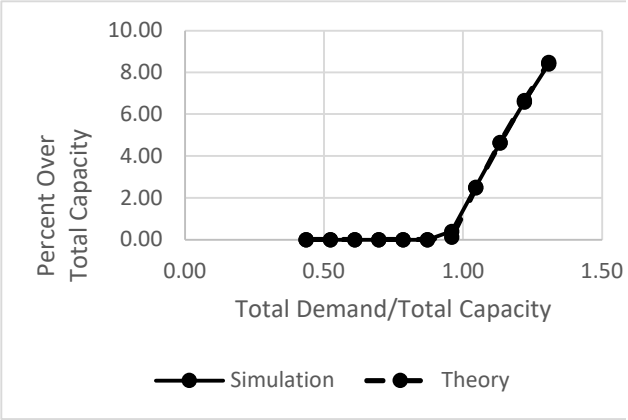


Figure 12: Comparing Theory with Simulation for Five Servers

For completeness, the extra service provided by using a merged pool is shown next:

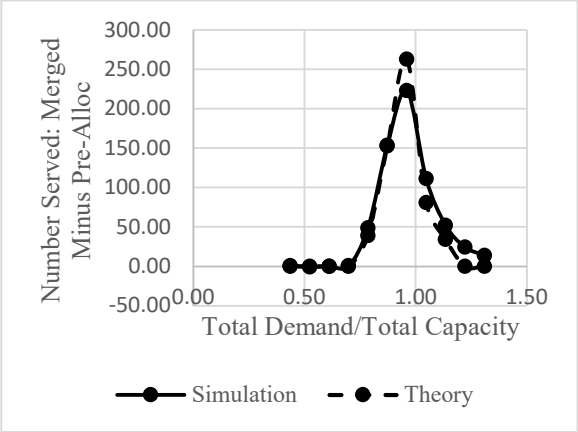


Figure 13: Increased Resource Utilization Through Sharing with Five Servers

4. Summary

After establishing the idea of comparing shared versus pre-allocated items (inventory) for multiple entities (servers) that each assign a customer temporary use of an item, a multi-dimensional grid was established to depict the items in use by each entity. Each axis of the grid is associated with an entity and the entire collection of items in use is represented by a single point on the grid. Forward and reverse probabilities were specified to control the random movement of the point. The probability of moving ahead decreases with increasing distance from the origin. Also, the probability of moving forward increases with increasing demand which makes intuitive sense. Since there is a finite number of items available for use, an absorbing boundary is provided which can be viewed as a multi-dimensional plane on the grid.

Next, a computer program was presented that simulates in one run both pre-allocated and shared entities for given input parameters. The pre-allocated cases are combined to allow comparison with the merged case. In broad terms, the main result is if each of the entities maintain the same demand/capacity ratio, then there is no advantage to sharing. But if not, sharing has a roughly 5 to 6% improvement in item utilization for the case when Total Demand/Total Capacity is approximately 0.8 to 1.0.

Finally, equations were derived for the number served and for overflow which led to simple closed form expressions. Comparing the equation to a fiveserver simulation provided an excellent match.

5. Acknowledgements

The software for the simulation was developed using Visual Studio Community 2017 Version 15.8.9. Visual Studio is a product of © 2017 Microsoft Corporation. The code was developed using the visual basic language.

The spread sheets mentioned in the article used Microsoft Excel. All of the graphs were produced with Excel. This document was created using Microsoft Word. Excel and Word are part of the Microsoft Office package.

The computer used in this study was a Dell D3630
Processor containing an Intel(R) Core(TM) i5-8500 CPU @ 3.00GHz 3.00 GHz.
The Installed RAM is 8.00 GB (7.84 GB usable).

References

1. He, L., Yabo, L., & Hong, L. (2009, October). Schedule optimization for NC resource sharing based-on greed algorithm and Tabu search. In *2009 Second International Conference on Intelligent Computation Technology and Automation* (Vol. 3, pp. 282-285). IEEE.
2. Chen, Q., Xu, Q., & Wu, C. (2019, September). Optimal sharing strategies of idle manufacturing resource considering the effect of supply-demand matching. In *2019*

International Conference on Industrial Engineering and Systems Management (IESM) (pp. 1-6). IEEE.
3. Gang, J., & Friderikos, V. (2018, April). Optimal resource sharing in multi-tenant 5G networks. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)* (pp. 1-6). IEEE.
4. Bell, S. L., & Williams, R. J. (2001). Dynamic scheduling of a system with two parallel servers in heavy traffic with resource pooling: Asymptotic optimality of a threshold policy. *The Annals of Applied Probability,* 11(3), 608-649.
5. E. W. Weinstein "Random Walk—1Dimensional" MathWorld—A Wolfram Web Resource.
6. Kalikow, S. A. (1981). Generalized random walk in a random environment. *The Annals of Probability, 9*(5), 753-768.
7. Tilman, D. (1986). Resource competition and the dynamics of plant communities. *Plant ecology,* 51-75.
8. Papoulis, A., & Unnikrishna Pillai, S. (2002). *Probability, random variables and stochastic processes.*
9. Wikipedia - https://en.wikipedia.org/wiki/Trinomial_expansion
10. Chandrasekhar, S. (1943). Stochastic problems in physics and astronomy. *Reviews of modern physics*, 15(1), 1.
11. Stover, Christopher and Weisstein, Eric W. "Matrix Inverse."