Block Size Optimization for VBASBS Queueing Model for Blockchain

Class Project
dept. of Computer Science
Oklahoma State University
Stillwater, OK, USA
student@okstate.edu

***Abstract*—TODO: Redo this later**
***Index Terms*—Blockchain, Optimization, Queueing**

I. Introduction

TODO: Redo this later

II. Related Work

TODO: Redo this later

III. Methedology

A. Performance Model

Our algorithm will assume the performance model proposed by Seol, Kancharla, Ke, Kim and Park - A Variable Bulk Arrival and Static Bulk Service Queuing Model for Blockchain [1]. The arrival rate of the individual blockchain transactions is variable, but not unpredictable. The arrival rate is assumed to vary with the size of the incoming transaction and the rate is denoted by the symbol λ.

The variable μ is defined to be the processing time of a block to be posted and purged as noted in Seol [1], and is assumed to be fairly static as stated in the model. Since this is assumed to be true, we will focus mainly on the performance measures of L_Q, W_Q and γ as described below.

The variable n is defined to be the total number of slots in the block. This will be used to define when a block (or batch of transactions) will be processed to minimize wait time while maximizing throughput.

P_i denotes the states with P_n denoting the maximum state before the block is processed. Given from the paper, the performance measure L_Q, which is the average number of customers in the queue, can be written as:

$$L_Q = \sum_{i=0}^{n} iP_i \tag{1}$$

Where the equation of the sum of i times state P at i is given as:

$$\sum_{i=0}^{n} iP_i = \sum_{i=0}^{n} i(q_i P_0(\sum_{j=1}^{i} j(\sum_{k=1}^{i-1}(\prod_{l=1}^{k-1} q_l)k) + i)) \tag{2}$$

But as can be noted from the code in the appendix of the paper which is responsible for plotting the associated graphs, this is equivalent to:

$$\sum_{i=0}^{n} i(q_i P_0(\sum_{j=1}^{i} j(\sum_{k=1}^{i-1}(\prod_{l=1}^{k-1} q_l)k) + i)) = nP_n \tag{3}$$

Where the state P at time n is written as:

$$P_n = \frac{\lambda}{\mu}\frac{n(n+1)}{2}P_0 \tag{4}$$

Therefore, L_Q can be written as:

$$L_Q = n * \frac{\lambda}{\mu}\frac{n(n+1)}{2}P_0 \tag{5}$$

The performance measure W_Q, which is the average amount of time a customer is in the queue, can be written as:

$$W_Q = \frac{L_Q}{\lambda} \tag{6}$$

Both of these performance measures will vary somewhat constantly by n times the ratio of $\frac{\lambda}{\mu}$ past some constant. And it can be noted that W_Q is simply a scaled measure of L_Q. However, the opposite end of this dynamic is the throughput, or how many transactions will be processed in a given amount of time. This is given in the model paper as γ, which is the throughput of the model, and is written as:

$$\gamma = \mu P_n = \mu\frac{\lambda}{\mu}\frac{n(n+1)}{2}P_0 = \lambda\frac{n(n+1)}{2}P_0 \tag{7}$$

These performance metrics based off of the model put forth in Seol [1] will form the basis of our algorithm. We will then try to minimize the waiting time while maximizing throughput based on the expectation of λ to adjust n to an optimum level. In doing so, we will compare our non-synchronous model to a synchronous model with a set value of block size n.

B. New Model

Due to the ethereum network switching from Proof of Work to Proof of Stake validation, as discussed in Sec. II, we have decided to update the base model in Seol [1] to reflect this new regime. This new Proof of Stake methodology essentially flips the previous base model on its head, where the time between blocks being pushed out to validators is set at twelve seconds, and because of this, the block size (in terms of gas) can then

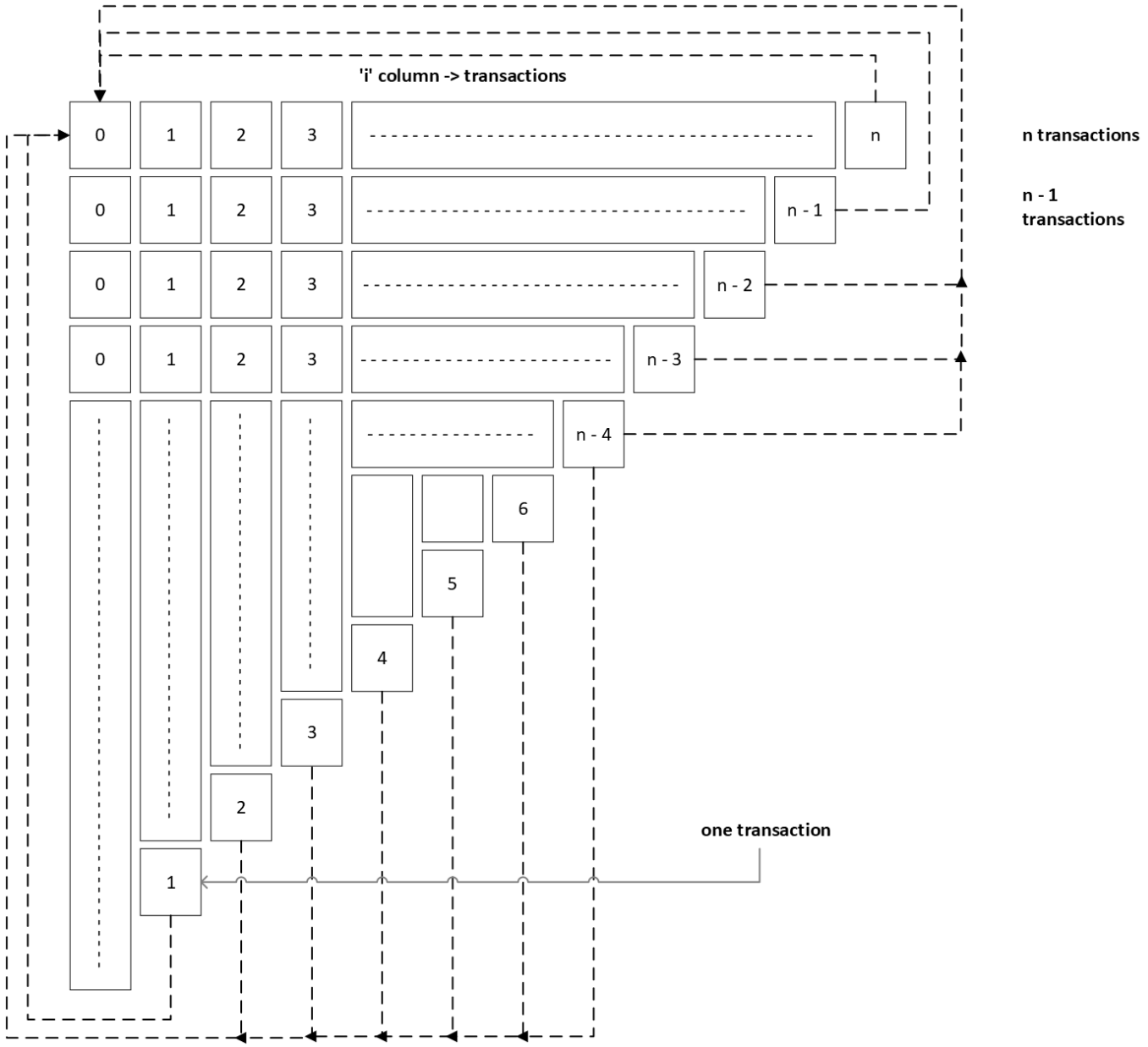

Fig. 1. Flow Diagram of new model.

fluctuate based on the number of transactions that have arrived in the latest block window of time.

The subsequently updated variables and assumptions are given below.

$P_{0,0}$: The state at which no time has passed in the block window of time.

$P_{n,0}$: The state at all of the time has passed in the block window of time, and the block is ready to be sent to the validator community, with no slow transactions.

$P_{n-j,j}$: The state at all of the time has passed in the block window of time, and the block is ready to be sent to the validator community, with j slow transactions.

$P_{i,0}$: The state at which i fast transactions have arrived in the block window of time with 0 slow transactions.

$P_{i,j}$: The state at which i fast transactions have arrived in the block window of time with j slow transactions.

τ: The number of minimum time increments that have occurred between the start of the block window of time and the current moment. This can be thought of as $1/\lambda$ where $lambda$ is defined in Seol [1].

Ω: The rate for the entire block to be posted and purged. This is assumed to balance the new form of equations as μ did in the original base model.

1) $P_{0,0}$:

Incoming from $P_{n,0}$, $P_{n-1,1}$, $P_{n-2,2}$, ... , $P_{2,n-2}$, $P_{1,n-1}$

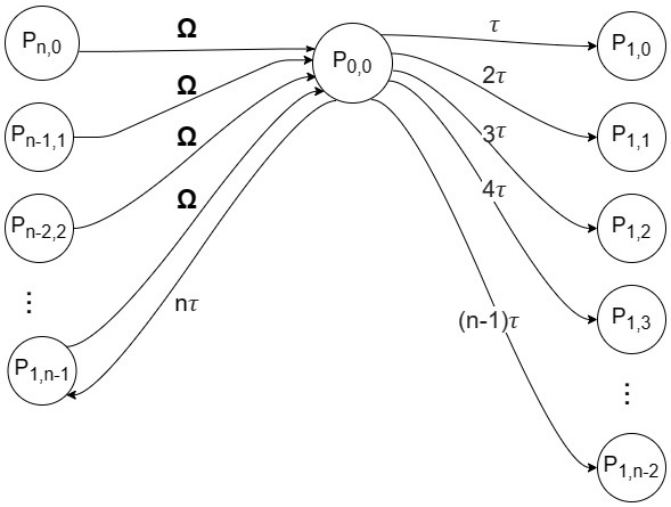

Fig. 2. State Transition Diagram of $P_{0,0}$

Outgoing to $P_{1,0}$, $P_{1,1}$, $P_{1,2}$, ... , $P_{1,n-2}$, $P_{1,n-1}$

$$\Omega(P_{n,0} + P_{n-1,1} + P_{n-2,2} + ... + P_{2,n-2} + P_{1,n-1}) = \tau(1 + 2 + 3 + ... + n - 1 + n)P_{0,0} \quad (8)$$

Here we have the summation from 1 to $n-1$ which can be rewritten as $\frac{n(n+1)}{2}$ and the other summation is all of the possible ending states before we return to $P_{0,0}$. This can then be rewritten in Eq. 9.

$$\Omega \sum_{j=0}^{n-1} P_{n-j,j} = \tau \frac{n(n+1)}{2} P_{0,0} \quad (9)$$

As can be seen here, our added parameter j for the slow transactions has the effect of splitting the final state, P_n from the base model in Seol [1] into various states from 0 slow transactions up to $n-1$ slow transactions.

It is with this addition, along chopping a fixed amount of time to minimum time increments, τ, that we can approximate the proof of stake workflow in blockchain networks, like ethereum.

2) $P_{1,j}$:

Incoming from $P_{0,0}$ only (with time $(j+1)\tau$)
Outgoing to $P_{2,j}$, $P_{2,j+1}$, $P_{2,j+2}$, ... , $P_{2,n-3}$, $P_{2,n-2}$

$$\cancel{2}(j+1)P_{0,0} = \cancel{2}(1 + 2 + 3 + ... + (n-j-2) + (n-j-1))P_{1,j} \quad (10)$$

$$P_{0,0} = \frac{(n-j-1)(n-j)}{2(j+1)} P_{1,j} \quad (11)$$

3) Combining Eq. 9 and Eq. 11:

$$\frac{\cancel{2}\Omega}{\tau(n(n+1))}(P_{n,0} + P_{n-1,1} + P_{n-2,2} + ... + P_{2,n-2} + P_{1,n-1}) = \frac{(n-j-1)(t-j)}{\cancel{2}(j+1)} P_{1,j} \quad (12)$$

$$P_{1,j} = \frac{(j+1)\Omega}{\tau(n(n+1))((n-j-1)(n-j))} (P_{n,0} + P_{n-1,1} + ... + P_{1,n-1}) \quad (13)$$

4) $P_{n-j,j}$:

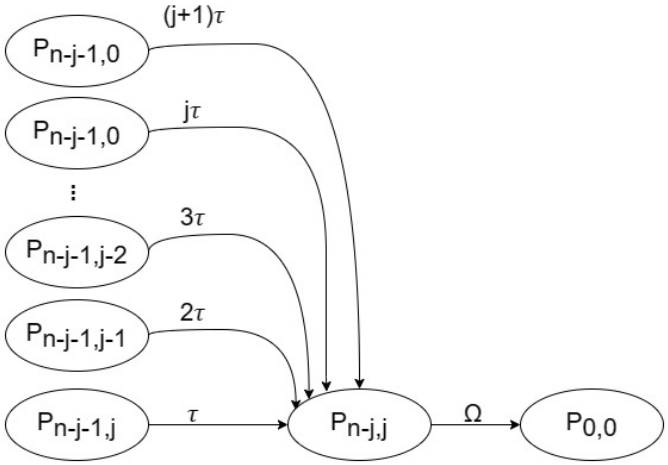

Fig. 3. State Transition Diagram of $P_{n-j,j}$

Incoming from $P_{n-j-1,j}$,$P_{n-j-1,j-1}$,...,$P_{n-j-1,1}$,$P_{n-j-1,0}$
Outgoing to $P_{0,0}$

$$\tau(P_{n-j-1,j} + 2P_{n-j-1,j-1} + 3P_{n-j-1,j-2} + ... + jP_{n-j-1,1} + (j+1)P_{n-j-1,0} = \Omega P_{n-j,j} \quad (14)$$

$$P_{n-j,j} = \frac{\tau}{\Omega}(P_{n-j-1,j} + 2P_{n-j-1,j-1} + ... + (j+1)P_{n-j-1,0} \quad (15)$$

5) $P_{i,j}$:

Incoming from $P_{i-1,j}$, $P_{i-1,j-1}$, ... , $P_{i-1,0}$
Outgoing to $P_{i+1,j}$, $P_{i+1,j+1}$, $P_{i+1,j+2}$, ... , $P_{i+1,n-i-1}$

$$\cancel{2}(P_{i-1,j} + 2P_{i-1,j-1} + 3P_{i-1,j-2} + ... + jP_{i-1,1} + (j+1)P_{i-1,0}) = \cancel{2}(1 + 2 + 3 + ... + (n-i-j-1) + (n-i-j))P_{i,j} \quad (16)$$

$$P_{i,j} = \frac{2}{(n-i-j)(n-i-j+1)}(P_{i-1,j} + 2P_{i-1,j-1} + ... + (j+1)P_{i-1,0}) \quad (17)$$

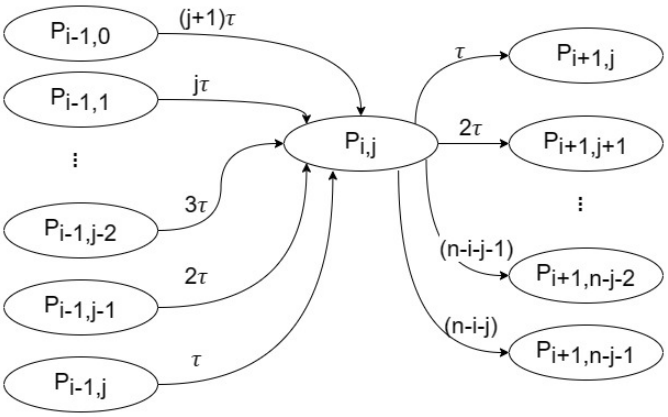

Fig. 4. State Transition Diagram of $P_{i,j}$

6) Constraint: Markovian Queueing Assumption:

$$P_{0,0} + \sum_{i=1}^{n} \sum_{j=0}^{n-i} P_{i,j} = 1.0 \tag{18}$$

7) Remarks:

• Each $P_{i,j}$ depends on the previous $(i-1)$ column and its preceding $(0...j)$ rows. This $P_{i,j}$ in turn, affects $P_{i+1,j}$ through $P_{i+1,n-i-1}$.

• The $i = 0$ column has only $P_{0,0}$ (no arrivals and no time elapsed.

C. Optimization

The main point left to discuss is the optimization problem to set n to an optimum level given what the new expectation is for the coming λ. The new expectation for the coming λ will be the average of all λ that we have seen up until that point, which is an unbiased estimator of the random variable λ since our model will not have any prior knowledge of the range of values to be passed in.

The derivatives of the γ and W_Q are difficult to calculate for these purposes, so to approximate the derivatives we will use the first differences. This calculation is given by:

$$\Delta\gamma = \gamma[2:n] - \gamma[1:n-1] \tag{19}$$

$$\Delta W_Q = W_Q[2:n] - W_Q[1:n-1] \tag{20}$$

Since the functions are evaluated at small, discrete, and uniformly spaced time steps the approximation should be accurate for our purposes.

$$optN = argmin f(n) := \{-nlg(n) \cdot \frac{\Delta\gamma}{\Delta W_Q}\} \tag{21}$$

D. Closed Form Optimization

In this semester, we can start by removing the iterative, numerical process described above with a closed form solution to get the optimal n. The first thing then is that we have to look at the metric. In the iterative process we had the ratio of throughput over waiting time multiplied by a plug of $nlg(n)$.

Let's then look at the equations of throughput and waiting time to see how the ratio looks when reducing it down. W_Q is given in eq. 6 as simply L_Q divided by λ, so lets write that out and show that λ cancels out in that equation.

$$W_Q = \frac{n * \frac{\lambda}{\mu} \frac{n(n+1)}{2} P_0}{\lambda} = n * \frac{\cancel{\lambda}}{\mu} \frac{n(n+1)}{2} P_0 * \frac{1}{\cancel{\lambda}} \tag{22}$$

Eq. 7 gives the equation for γ, so if we look at the equation for the ratio of throughput over waiting time we get eq. 23.

$$\frac{\gamma}{W_Q} = \frac{\lambda \frac{n(n+1)}{2} \cancel{P_0}}{\frac{n}{\mu} \frac{n(n+1)}{2} \cancel{P_0}} = \frac{\lambda}{\frac{n}{\mu}} = \frac{\mu}{n} * \lambda \tag{23}$$

Now, this ratio when reduced shows that throughput over waiting time is a montonically increasing function, with μ being fixed this means that n is directly dependent on λ. This would not lead to an optimal level of n, but rather as λ became smaller (or quicker), n would should become smaller. In practice, batching was necessary to process transactions more efficiently, so there should be some penalty for having blocks that are too small. The particular network or application would probably have a direct impact on this number, but for our case in an arbitrary blockchain, let's set this to $\frac{1}{3n^3}$. In this way, when n gets very small, the penalty gets larger. A maximum can also be determined based on network processing speeds as a cap to n. So, now we have eq. 24 which we can denote M for metric.

$$M = \frac{\gamma}{W_Q} - penalty = \frac{\mu\lambda}{n} - \frac{1}{3n^3} \tag{24}$$

To find the maximum/minimum of this equation we need to take the first derivative with respect to n and set it equal to zero.

$$\frac{\delta M}{\delta n} = -\frac{\mu\lambda}{n^2} + \frac{1}{n^4} = 0 \tag{25}$$

Then solving for n we have eqs. 26, 27, 28.

$$-\frac{\mu\lambda}{n^2} = -\frac{1}{n^4} \tag{26}$$

$$\frac{n^4}{n^2} = \frac{1}{\mu\lambda} \tag{27}$$

$$n = \frac{1}{\sqrt{\mu\lambda}} \tag{28}$$

We now have the closed form solution that the optimal block size (n) is inversely related to the square root of the

cost of creating a block (μ) times the speed with which the transactions are coming into the network(λ).

Using this same equation, λ can also be projected based on the cyclical nature of how transaction speeds progress throughout the day and the optimal n can be calculated before each block is created to predetermine the next block size to be created.

IV. Results

TODO: Redo this later

V. Conclusion

TODO: Redo this later

References

[1] J. Seol, A. Kancharla, Z. Ke, H. Kim, and N. Park, "A variable bulk arrival and static bulk service queueing model for blockchain," in *Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure*, 2020, pp. 63–72.