Modeling the Batch Training Mechanic in 0AD

by Micfild

Disclaimer:

This document is designed to complement the information available on the original post and so it will not explain much about some assumptions and definitions stated in said post.

1. Batch Formula

Let's start by revisiting the BatchTime formula and casting it in an easier way to work with:

$$\boldsymbol{BatchTime}(t, k, mod) = t * (k^{mod})$$

Where:

t = BaseTime

k = BatchSize

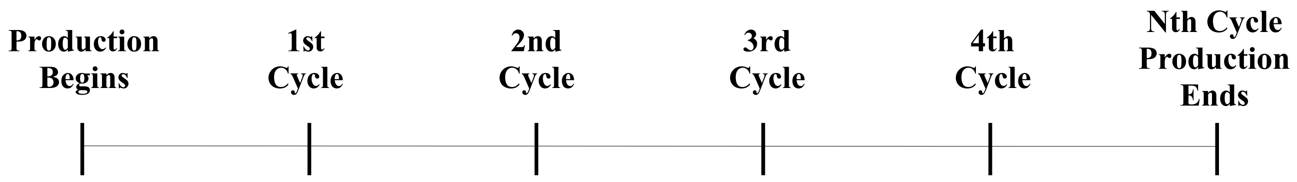
mod = Batch Modifier

This formula gives the amount of time it takes for one Batch of units to get produced. One thing to note is that if the BatchSize (k) = 1 ,as in the 1by1 method of training, BatchTime will be equal to BaseTime. This means that training times (whether Batching or building units 1by1) can always be obtained by this formula.

2. ActiveTime

Next, we'll be modeling **ActiveTime.** As previously established, each second (or fraction of a second) a unit is alive on the map, it is generating the same amount of ActiveTime, e.g. A unit alive for 10 seconds generates 10 seconds of ActiveTime, and 2 units alive for 10 seconds generate 20 seconds of ActiveTime and so on.

In this model, we are interested in the amount of ActiveTime generated overall from the moment a building starts producing units until the time it stops. The time it takes for a unit (or a batch of units) to be produced will be called **ProductionCycle** (or **Cycle** for short).



2.1. ActiveTime in 1by1

Let's model the ActiveTime for a batch of size 1 (1by1 training) since its easier. After the **first Cycle** is completed we'll have out first unit out and it will start generating ActiveTime. However, since it just got out, the total amount of ActiveTime generated thus far is 0 (**$a_0 = 0$**). By the end of the **second Cycle** we'll get one more unit and the amount of ActiveTime generated by the first unit is **the same amount of time X that it takes for a Cycle to be completed** ($a_1 = X$). But now we have 2 units out, so by the end of the **third Cycle**, those 2 units will have generated 2X ActiveTime. So, the total amount generated so far is X + 2X ($a_2 = X + 2X$). If we continue this for a few more steps we get a better view of our situation.

1st Cycle ----- $a_0 = 0$

2nd Cycle ----- $a_1 = X$

3rd Cycle ----- $a_2 = X + 2X$

4th Cycle ----- $a_3 = X + 2X + 3X$

5th Cycle ----- $a_4 = X + 2X + 3X + 4X$

....

(n+1)th Cycle ----- $a_n = X + 2X + 3X + 4X + ... + nX$.

Since X is in every term we can put it in evidence, ending up with:

$a_n = X * (1+2+3+4+...+n)$

That term in parenthesis resembles an Arithmetic Progression (A.P.) with Ratio = 1. So we resolve the sum by using the formula for the sum of an A.P. with **n terms**, and we end up with this:

$a_n = X * (n * (1+n) / 2) = X * (n^2+n)/2$

If we look carefully, we'll notice that our index m and the number of Cycles differ by 1. That means that for any **n ProductionsCycles** we'll have that the total amount of ActiveTime generated in a batch of size 1 is:

n^th Cycle ----- $a_{n-1} = X * ((n-1)^2 + (n-1)) / 2$

Since X is the amount of time it takes to complete a Cycle, which in this case is the amount of time it takes to build 1 unit (it's BaseTime), then for the 1by1 case we have:

$$\boldsymbol{ActiveTime}(n) = \left(\frac{(n-1)^2 + (n-1)}{2} \right) * \boldsymbol{BaseTime}$$

Where **n** is the **# of ProductionCycles.**

2.2. ActiveTime in Batches

Let's start by modeling this problem similarly to the previous one. Let **X** be the amount of time it takes for a single Batch of **K** units to be produced (i.e. X = BatchTime(K)). The same as before, when our **first Cycle** is complete we have now **K units**, but the Amount ActiveTime produced thus far is 0 ($a_0 = 0$). By the time we finish our **second Cycle**, we'll have 2K units and our first K units will have generated a total of K*X seconds of ActiveTime ($a_1 = K*X$). By the end of our third Cycle, we'll have 3K units, and the 2K units we had, managed to generate an aditional 2K*X seconds of ActiveTime, bringing the grand total to K*X + 2K*X ($a_2 = K*X + 2K*X$). Continuing this proccess we end up with:

1st Cycle ----- $a_0 = 0$

2nd Cycle ----- $a_1 = KX$

3rd Cycle ----- $a_2 = KX + 2KX$

4th Cycle ----- $a_3 = KX + 2KX + 3KX$

5th Cycle ----- $a_4 = KX + 2KX + 3KX + 4KX$

....

(n+1)th Cycle ----- $a_n = KX + 2KX + 3KX + 4KX + \ldots\ldots + nKX$.

This is the same behaviour we had in the 1by1 case, so if we put the KX in evidence and solve the resulting A.P with Ratio = 1 we get:

$a_n = KX * (n * (1+n) / 2) = KX * (n^2+n)/2$

And in the exact same way, our index number differs from our number of Cycles by 1 so:

nth Cycle ------ $a_{n-1} = KX * ((n-1)^2 + (n-1)) / 2$

However, in this case X is no longer the BaseTime of a unit but the BatchTime a Batch of units of size K (BatchTime(K)). So the resulting formula is:

$$\boldsymbol{ActiveTime}(n,k)=\left(\frac{(n-1)^2+(n-1)}{2}\right)*k*BatchTime(k)$$

2.3. Universal Formula:

As we've seen before, a Batch Size of 1 (K = 1) yields a BatchTime(K) = Basetime. So the formula immediately above not only works for Batch Training, but it also works for 1by1 as well, since if K =1 it will colapse into the 1by1 formula.

3. Efficiency Comparison

To produce X units, which method generates more ActiveTime?

So we have our parameter of comparison (ActiveTime) and our constraint (X units produced). We can calculate the Efficiency (or ineficiency) of Batching over 1by1 by the following equation:

$$\boldsymbol{Efficency} = \frac{TotalActiveTime\left(Batching\right)}{TotalActiveTime\left(1\,by\,1\right)}$$

But, before we can continue, we have to address something. Although our constraint is the number of units produced, the variables to which we have access in the game are BatchSize (k) and #Cycles (n). #Units comes as a result of the interaction between those two which can be described as: X(n.k) = n*k. With that thought in mind we can continue.

First, let's deal with the denominator, since it's easier to solve. All we have to do is apply the our ActiveTime formula, with **k = 1** and **n = X/k = X.**

$$\boldsymbol{Efficency} = \frac{TotalActiveTime\left(Batching\right)}{ActiveTime\left(X,1\right)}$$

Now the numerator is a bit tricky, because the Batching method will always finish its last cycle way before 1by1 finishes its. That means that our formula will only give us part of the Total amount of ActiveTime generated. The rest will be obtained by multiplying the total #of Unit and the rest of the time it takes for 1by1 to finish production, so:

$$\boldsymbol{TotalActiveTime}\left(Batching\right) = ActiveTime\left(n,k\right) + X\left[\left(X * Basetime\right) - \left(n * BatchTime\left(k\right)\right)\right]$$

where X(n,k) = n*k.

So we have our full formula:

$$\textbf{\textit{Efficency}}=\frac{ActiveTime(n,k)+X[(X*Basetime)-(n*BatchTime(k))]}{ActiveTime(X,1)}$$

If we apply it to a unit with **BaseTime of 10** seconds and **Batch Modifier of 0.8** we we'll get the following table of results.

Table. 1 – Efficiency comparison between Batch Training and 1by1 Training

		BatchSize			
		2	**3**	**4**	**5**
Batch Cycle	**1**	52%	59%	65%	69%
	2	93%	96%	99%	**101%**
	3	**101%**	105%	108%	111%
	4	104%	109%	112%	115%
	5	106%	111%	115%	118%
	6	107%	113%	116%	119%
	7	108%	114%	118%	121%
	8	109%	114%	118%	121%
	9	109%	115%	119%	122%
	10	110%	116%	120%	123%

In **Red** are the first instances where the Batched method of training breaks even with the 1by1 method, in regards to ActiveTime generated. These results show **that Batching gets more Efficient the longer you can mantain it and the bigger the Batch size is.**

4. ProgressTimeout

All units have an integer BaseTime, but BatchTimes are decimal numbers. This can become a problem since in 0AD, unit production only effectively happens on integer values. This means that if a ProductionCycle ends at 5.6 seconds, the unit will only spawn at the 6 second mark. To compensate for this, the game takes the 0,4 seconds of time lost and applies it as a discount to the next queued ProductionCycle.

This compensation does not happen if AutoQueue is on. This will artificialy increase training time in batches when using AutoQueue. Although the effect is negligible in 1 or 2 ProductionCycles it can accumulate over time, making AutoQueue potentialy less efficient than manually Batching units.

This is an interesting way to reward people that don't rely on AutoQueue and have the ability to micromanage their own economy and unit production.

To model this phenomenon is simple. If we'll always be spawning units on the next integer value of time and not applying any corrections, this effectively means that we'll be rounding up (↑) our BatchTime:

$$\boldsymbol{AutoQBatchTime} = \uparrow BatchTime(k)$$

Where the upwards arrow is used to represent the rounding up operation.

So, the only change we need to do to our orginal formula to account for AutoQueue is to just round up the value of the BatchTimes.

If we apply the correction and redo our calculations, for a **BaseTime of 10** seconds and a **Batch Modifier of 0.8** we have the following table of results.

Table. 2 – Efficiency comparison between AutoQBatch Training and 1by1 Training

		BatchSize			
		2	**3**	**4**	**5**
Batch Cycle	**1**	40%	50%	60%	65%
	2	87%	90%	96%	99%
	3	96%	**100%**	105%	109%
	4	**100%**	105%	110%	113%
	5	102%	107%	113%	116%
	6	104%	109%	114%	118%
	7	105%	110%	116%	119%
	8	105%	111%	116%	120%
	9	106%	112%	117%	120%
	10	106%	112%	118%	121%

Although we can see that there was a loss of Efficiency when compared to Manual Batching, to quantify it all we need to to is divide the results of one table with the other, after all:

$$\boldsymbol{Efficiency} = \left(\frac{Batch}{1\,by\,1}\right) \div \left(\frac{AutoQBatch}{1\,by\,1}\right) = \left(\frac{Batch}{1\,by\,1}\right) \times \left(\frac{1\,by\,1}{AutoQBatch}\right) = \frac{Batch}{AutoQBatch}$$

and that will lead us to the third and final table:

Table. 3 – Efficiency comparison between Batch Training and AutoQBatch Training

		BatchSize			
		2	**3**	**4**	**5**
Batch Cycle	**1**	129%	118%	108%	106%
	2	107%	106%	103%	103%
	3	105%	105%	102%	102%
	4	104%	104%	102%	102%
	5	104%	104%	102%	102%
	6	104%	103%	102%	102%
	7	103%	103%	102%	102%
	8	103%	103%	102%	101%
	9	103%	103%	102%	101%
	10	103%	103%	102%	101%

5. Discussion

From the results we obtained, we can conclude that Batch Training is, at first, less efficient than 1by1, but the addition of more ProductionsCycles changes that result quickly, with Batching overtaking 1by1 around the 2nd or 3rd ProductionsCycles. Batching also increases in efficiency the bigger its size (although that also increases the upfront resource cost and it's harder to maintain).

We have also seen that AutoQueue Batching is slightly less efficient that Manual Batching, due to the interaction of decimal Batching times and ProgressTimeout. Since the loss of efficiency is quite small, it can be argued that the benefits of Autoqueuing (namely its ease of management) still outweighs this small downside, specially outside high level matches.