# Some considerations for using TPF Low Priority(LP)

Robert Blackburn, Mark Gambino, Mike Shershin, John Tarby

January 11, 2022
v1.0

## 1 Preamble

The following has calculations and predictions based on mathematical theory. Variations in the customer environment from the models and parameters used could result in deviations from the predicted behavior.

## 2 Executive Summary

Key points:

- Utilities are growing in their run completion time. With world wide customers there are fewer dormant times to run utilities.
- With LP utilities can complete much faster more easily meeting SLAs. In addition there will be minimal impact to TW work. LP consumes perishable MIPS and adds business value.
- Not all utilities are good candidates for LP. They must have low mean and low variance of CPU consumed between IO events. In practice most utilities would be eligible.
- Less care and administration required to run utilities. LP is much more robust than traditional utilities and balancing act between utility completion time and interference with TW work.
- IO limits can be integrated into LP so the customer can govern how much IO is acceptable in their environment.

• LP can provide significant value even without Fenced I-streams. Recoup will leverage Fenced I-streams to greatly decrease its runtime.

## 3 Introduction

This discussion will use results from queueing theory and probability. There are many good texts for reference. . e.g. D. Gross and C. Harris Fundamentals of Queueing Theory and K.L. Chung Elementary Probability Theory with Stochastic Processes.

In addition to the Executive Summary above, we will include various summary sections throughout the paper to aide in the understanding of the mathematical details.

We will use Recoup as an example of a utility since it is best understood and widely used. However the analysis will apply to all TPF utilities.

For the queueing system we are interested in let $s$ be a random variable describing the service time for a customer at the server. Here customer is the entity moving around the queueing system and is not to be thought of as a person. First some notation where we mainly focus on CPU queueing and response time.

Define $\lambda$ = arrival rate
E(s) = mean service time
E(q) = mean queueing time(does not include service time)

$$Var(s) = E(s - E(s))^2 = E(s^2) - E(s)^2 \tag{1}$$

So $\rho = \lambda$ E(s) = server utilization

For a description of how TPF implemented LP see the following:
https://www.ibm.com/docs/en/ztpf/2020?topic=programs-low-priority-ecbs

## 4 Issues with running CPU near $\rho = 1$

Over many years of experience TPF customers have known it is very difficult to run at CPU utilizations near 1. Both the mean response time and its variance get very large. While TPF runs very well at large MP levels the inherent randomness of the message arrival and service times bring queueing into play. Since typically the number of customers is very large we can model this as an open queueing system where the message arrival rate is unaffected by the number in the TPF system.

The TPF scheduler can be roughly modeled as separate M/G/1 queues with some queue balancing at the limit, where the M refers to random arrival(exponential holding times) and the G is a general service time distribution. Despite the MP load balancing in the scheduler the M/G/1 equations can show the differences in priority and FCFS disciplines. We see the effect of high utilization in (2). Even for $\rho = .95$ there is a scaling factor of 20(1/(1-.95)) as compared to very low utilizations.

$$W_q = \frac{\lambda E(s^2)}{2(1-\rho)} \tag{2}$$

Thus a valid question is why is the lab telling customers it is fine to run at $\rho = 1$ where the low priority work makes up the difference from the transaction work(TW) utilization? This will be discussed below.

## 5 Compromises with running Recoup at same priority as transactional work

From (2) we have the need to stay below $\rho = 1$. Assume we want to have $\rho < .9$ for combined TW and Recoup. This is necessary since TW and Recoup both compete at the ready list level.

Typically Recoup is run at off hours which are harder to come by in the global economy and limited in the number of child ECBs. Assume for a period of 2 hours mostly TW utilization stays below .5. Then we might set the Recoup ECBs to target an additional .4. The problem is the TW work has time based mean cpu utilization changes and say the mean jumped to .65 for 20 minutes. Then there would be CPU overload during this period and soon the input list would not be serviced. The response might be to stop Recoup. This requires constant monitoring.

In addition Recoup itself is a source of variation in CPU consumption as it goes through its phases. Thus even a fixed number of child ECBs that kept Recoup below .4 CPU could have times when Recoup only used ,say .2, of the CPU.

A solution is to be very conservative with the Recoup ECB allocation so that max TW utilization + Recoup utilization $< .9$. To ensure this requirement is satisfied it may be that the total CPU is far less than .9. The worst case would be when TW work is at a random low point and Recoup is in a relatively dormant CPU as well. This period could last minutes not seconds.

Thus we have a compromise between Recoup speed and TW impact. This is a delicate balancing act to optimize Recoup throughput and inflict minimal response time increase to TW work. In addition there is a time component: what worked for 5 months may not this month and seasonal effects e.g Thanksgiving.

So the customer programming staff will be making constant adjustments to parameters to optimize the process.

## 6 Alternate proposals to run Recoup keeping CPU near $\rho = 1$ (without LP)

There have been several ideas on how to balance Recoup with TW. Most promising would be to detect a large queue IPMT list and then stop creation of more child ECBs. Essentially at this point the system is overloaded and thus newly arriving TW will have very large response times for short periods.

This is a workable design with frequent detection and back off to minimize the length of time where $\rho > 1$ and the queue is linearly increasing with time. If LP was unavailable the lab would probably implement something like this.

But LP is superior as it never gets into the forced TW delays and yet is able to keep the CPU at $\rho = 1$ with a sufficient supply of ECBs.

## 7 Discussion on Recoup taking longer with LP

### 7.1 work conserving by Recoup

One common concern with converting Recoup to run LP is elongation of runtime since the Recoup ECBs will have less priority than regular ECBs. This is true if the Recoup ECB count was kept constant since each ECB would have the CPU queueing time increased since it comes off a lower priority list.

Recoup is a work conserving task. By this we mean that no work is created or destroyed in the system. Destruction of work would be if customer left before work completion and creation if the CPU spun while work was waiting. Thus we see that if we consume more CPU in a fixed time the associated IO will go at a faster rate and the job will finish faster.(Assuming the CU IO limit has not been reached)

People have been concerned that while there are more Recoup ECBs each one is queueing longer so what is the net effect? The work conserving argument makes this clear since the CPU sees higher mean utilization with associated higher IO rates.

Thus we need to permit Recoup to run with more ECBs. We define the Utility Run ECB Count(UREC) as the number of child ECBs the parent ECB will permit to exist; in this case Recoup but each utility will have such a number.

The UREC will be driven mainly by being large enough to drive the associated CPUs to utilization $\sim 1$. This might be attached fenced CPs or consuming extra capacity of the in use CPs. Customers should not fear running at utilization of $\sim 1$ and permitting more ECBs. Note both of these are problematic with traditional Recoup but not LP. Exactly how to calculate the UREC for each utility will be discussed in a later section.

But making the UREC unnecessarily large in the sense that far fewer ECBs would have driven the utilization to 1 is not optimal. We want reasonably tight upper bounds for the UREC for each utility. Then when multiple utilities are running at the same time the change to the number of system allocated ECBs can be kept to a minimum. This has advantages in not altering the shutdown of tiered services too greatly as opposed to just raising the ECB count radically from 4000 to ,say, 10000 with no justification.

Clearly each customer will be different but we are hopeful that something around 400 additional system defined ECBs should suffice to handle all the cases when multiple utilities are running concurently.

## 8 Equations for mean queue time in 2 priority system

We will use a combination of two models. The first is a head of the line(HOL) M/G/1 nonpreemptive priority with Poisson arrivals, general service time and one server. In this nonpreemptive system a newly arrived customer waits until the in service customer completes before getting access to the server. The infinite population just means that the number of customers in the system has no effect on the arrival rate.

The second is the cyclic queue model M/M/1/K/K with Poisson arrivals , exponential service times , one server , K allowed customers and a finite population of K identical devices. Here the arrival rate will depend on the number of customers n in service. It will be exponential rate scaled by a factor of K-n since n customers are removed from the finite supply of K and the remaining K-n population can still supply input.

So the TW work will have an infinite source of arrival and the LP work will be generated by a fixed supply of ECBs(the K) so a finite source. We expect in almost all cases K will be reasonably large, say > 30. Thus the finite source effect will not be too strong.

In addition when many independent counting processes (not necessarily Poisson) are added together, the sum process often tends to be approximately Poisson if the individual processes have small rates compared to the sum.

The following equations apply with $n$ priority classes but for our purposes we will use only two. This is because we are mainly concerned with the effect on transactional work by utilities such as Recoup and others. We care about the LP ECBs only in that there is a sufficient supply to take the remaining CPU left over by TW work. Note priority 1 is the higher priority below.

$$\begin{aligned}
\lambda &= \lambda_1 + \lambda_2 && (3)\\
E(s) &= \frac{\lambda_1}{\lambda}E(s_1) + \frac{\lambda_2}{\lambda}E(s_2) && (4)\\
E(s^2) &= \frac{\lambda_1}{\lambda}E(s_1^2) + \frac{\lambda_2}{\lambda}E(s_2^2) && (5)\\
u_j &= \lambda_1 E(s_1) + ... + \lambda_j E(s_j) \quad j = 1, 2 && (6)\\
W_{q_1} &= \frac{\lambda E(s^2)}{2(1-u_1)} && (7)
\end{aligned}$$

$$W_{q_2} \quad = \quad \frac{\lambda E(s^2)}{2(1-u_1)(1-u_2)} \tag{8}$$

## 9 Characteristics of a utility that makes it a good candidate for running LP

As seen by (7) the only way the LP work can affect TW work is through $E(s^2) = Var(s) + E(s)^2$. Thus if either the mean or the variance of CPU service time is large.

This makes sense intuitively since the only way TW work can be delayed by LP work is at TW arrival there is LP work in progress. Is is clearer that if this mean is large the ready list will wait a long time to restart since there is no preemption. But even if the mean is small but rarely a very large service time arrives on the defer list this can cause serious delays to the ready list.

Of course TW jobs will compete with other TW jobs and that is seen in the denominator of (7) in the component of TW utilization.

With Recoup which typically has $E(s) = 10\mu$ and very low variance this is not a problem and thus is well suited for LP. ($\mu$ = mics)

We bring this up since there are many TPF utilities. It will take significant domain knowledge to understand the cpu consumption variations of each utility. The lab can help with customer efforts. How ever we feel most utilities will have nice behaviors and are readily adapted to LP use.

A large exception to nice behavior is the following.

## 10 Extreme risk in using LP for infinite source random arrival (the usual TPF arrival method)

As can be seen by (7) and (8) we can have some priority classes with finite response time and others with infinite times. That is why we fix the number of ECBs in the LP population so as to bound the number waiting for service even if the higher priority classes consume all the CPU.

However if in a typical TPF open arrival(infinite source) system some ECBs were put into the LP class then we could have an unbounded increase of ECBs. An open system has the arrival rate unchanged by the number of items queued in the system. We would have a combination of open and closed queueing networks with the LP utilities being closed. If the combination of TW and LP work exceeded utilization of 1 then TPF would eventually go into ECB shutdown as more unfinished work is added to the defer list.

At this point TPF must pull ECBs off the defer list to avoid deadlock. It is likely these LP items would iterate many times between IO and back on defer list.TW work would be delayed while LP ECBs were processed. This would then be a sort of reversal of priorities as some LP utility work would go ahead of TW work waiting on the input list. Depending on the utility these LP ECBs could live a long time so the effect could exist for some time.

It is difficult to over emphasize how bad it is to use LP for open arrival ECBs.

## 11 Using ZMOWN and Data Collection to get mean service times for CPU and DASD

Let E(s) be the mean CPU service time and E(d) = mean IO service time: both without queueing.

Recoup does one DASD IO and then one trip to CPU to get the next record and repeats. Other utilities collect 16 IO requests and then issue them and wait for all to return before repetition.

ZMOWN will collect the total CPU(TC) time consumed in a run and the total number(TI) of IO completed. Then E(s) = TC / TI.

In the second case with the 16 IO just scale by 16 to get mean CPU service time.

For E(d) we need to use the file report in data collection. It shows the mean time from SSCH to IO interrupt for each device, i.e. the service time.

## 12 Calculate number of Recoup ECBs to drive single Fenced CP to $\rho$ near 1

Let E(s) be the mean CPU service time and E(d) = mean IO service time: both without queueing.

For D/D/1/K/K we have

$$\rho = min\left(\frac{K}{1+E(d)/E(s))}, 1\right) \tag{9}$$

So for constant cpu and DASD service times of 10 $\mu$ and 300 $\mu$ respectively we have for K=31

$$\rho = min\left(\frac{31}{1+300/10}, 1\right) = 1$$

For M/M/1/K/K we have

$$p_0 = \frac{1}{\sum_{n=0}^{N} \frac{N!}{(N-n)!}\left(\frac{E(s)}{E(d)}\right)^n} \tag{10}$$

and then we use

$$\rho = 1 - \rho_0$$

Here since the cpu and DASD service times are exponentially distributed the K to get near utilization of 1 increases. Roughly it will be 300/10 *1.5 = 45 as this summation is more involved to calculate.

We have that (9) and (10) effectively bracket the required ECBs (the K) for almost all utilities with the bias closer to the constant service times since utilities tend to operate that way.

Each utility will potentially have a different $E(s)$ and so more or less ECBs would be required to get CPU utilization $\sim 1$.

## 12.1 Driving Inuse CPs to full utilization

If TW work is consuming , say, .6 of the CPU less LP ECBs will be needed than in the Fenced case to reach $\rho = 1$. A proportional approach could be used: $1 - .6 = .4$ of the CP would need to consumed so take 40% of the Fenced value for a like utility.

As was discussed previously we want fairly tight bounds on the UREC to get each utility to complete quickly.

## 13 Multiple utilities running concurrently

Currently customers do not overlap utilities in time. With use of LP , the time for each utility will be reduced so there is even less of a problem in scheduling them.

Each utility will have by definition a REC(run ECB count), which is the maximum ECBs allowed to be created for that specific utility and controlled by the parent.

The REC will vary by utility and will involve $E(s)$ and $E(d)$ as well as how fast the customer needs the utility to run. Customers should establish reasonably tight upper bounds for the REC. e.g. do not just define REC=100 when it is clear REC=20 would be fine.

This is because we want to keep the extra allocated ECBs configured to run LP as a relatively small ratio in order to create new shutdown limits that preserve system behavior. We do not want 4000 ECBs to grow unnecessarily to 8000 ECBs due to poorly thought out RECs.

Going forward as LP use increases customers will have to detect and factor out LP utilization and ECBs in their decisions to shunt non mission critical work or tiered services(services that use Lodic to stop them when TW gets large) . We recommend having a large supply of ECBs such that even if by chance a few utilities ran concurrently their added ECB usage would not put the system in tiered service shutdown.

To estimate this incremental LP number of ECBs we establish a non tight upper bound but with high probability of success as follows. List all utilities in decreasing order by their LP number of ECBs: $U_1, U_2, ...$ The customer will declare probability of ,say, more than 3 utilities running together is $< .001$ and then just use $U_1 + U_2 + U3$. Now any sample 3 utilities running together is unlikely to be $U_1, U_2, U_3$ but their sum is an upper bound.

## 14 Shutdown consideration with LP

How to calculate the maximum LP ECBs needed over all posible utilities is discussed in a previous section. Assume this ECB number is 400 with 4000 ECBs currently defined in total.

Further assume the previous shutdown limits were 1500 , 1000 and 500 . This is 1500 ECB remaining for job1 and 1000 for job2 and finally 500 safety shutdown where the input list is blocked.

For the new configuration define 4000 + 400 = 4400 ECBs and adjust the percentages so that the 1500,1000 and 500 are all preserved. If LP ECBs are using 400 ECBs then job1 and job2 will still get to consume the same number of ECBs before they Lodic throttle themselves. If not the can get up to 400 more but this is fine since these numbers are quite robust so there will be small system effects with the potential change.

Note that LP work will adhere to the final input list shutdown of 500.

## 15 Discussion of the TPF IO limit for utilities

A critical piece of data needed is the response time curve as a function of CU IO/sec. IBM supplies these for its CUs and we believe the other vendors do as well. Since Fenced CPs can generate over 100K IO /sec per CP it became important for the lab to create this parameter.

Customers have typically run high IO load utilities in off hours so then the TW workload had a small IO load itself. Now we expect customers will run Recoup in prime time using LP with minimal impact to CPU so we wanted to extend this feature to IO, i.e. alllow explicit IO/sec controls.

First the customer must decide what is an acceptable IO response time increment to the TW. This gets interesting as Recoup IO has almost zero CU hit ratio being random access to the database. So the response time for IO will be above what is normal for TW work alone. Either running new CU performance reports or just proceeding cautiously an IO limit will be determined. The lab can assist with this effort. It is advisable to be quite conservative as the slope of response time is large as the CU approaches it IO throughput limit. It will be approximately an M/M/c queueing disipline and this shape can be found in the literature.

Assuming we are not at the CU limit there is no magic IO/sec limit value. Mean IO response is an increasing function of IO rate and each customer must consider it and their own SLAs etc.

## 16 Comparison of Recoup using LP and conventional approach

Assume CPU service of .25 mills for TW CPU(TWs) and 10 mics(LPs) for Recoup.

As a method for comparing the performance of different TW and Recoup combinations we will use

$$\frac{W_q}{E(s)}$$

This is intuitively reasonable as the TW work has to pay $E(s)$ each time it does an IO and we want to know how long the TW must wait before getting into service, i.e. the $W_q$.

### 16.1 TW at .5 and traditional Recoup using .4

Set TW utilization = .5 and LP at .4 for total of .9. So to find TW rate

$$.5 = .25n10^{-3} => n = 2000$$

and for LP rate

$$.4 = n10^{-5} => n = 40000$$

proportion TW =2000/40000 = .0476 so LP = .9524

Now we can calculate

$$E(s) = .0476TWs + .9524LPs = .021410^{-3}$$

Checking utilization 42000 x $.021410^{-3}$ = .8999

Assuming CPU service times are constant (0 variance) we have

$$E(s^2) = .0476(TWs)^2 + .9524(LPs)^2 = 3.0710^{-9}$$

$$W_q = \frac{\lambda E(s^2)}{2(1-\rho)} = \frac{(42000)3.0710^{-19}}{.2} = 6.44710^{-4}$$

and so

$$\frac{W_q}{E(s)} = .6447/.25 = 2.58$$

## 16.2 Transaction work(TW) at .5 plus LP Recoup using .4

$$W_{q1} = \frac{\lambda E(s^2)}{2(1-u_1)} = \frac{(42000)3.0710^{-19}}{1} = (1.289)10^{-4}$$

$$\frac{W_q}{E(s)} = .1289/.25 = .516$$

## 16.3 TW at .5

$$W_q = \frac{\lambda E(s^2)}{2(1-\rho)} = \frac{2000((.25)10^{-3})^2}{2(1-.5)} = (1.25)10^{-4}$$

Thus

$$\frac{W_q}{E(s)} = .125/.25 = .5$$

## 16.4 TW at .5 and large second moment CPU service time LP utility using .4

Assume LP work has exponential distribution with mean of 3 mills.

$$E(x) = 1/\lambda$$

$$\sigma(X) = \frac{1}{\lambda^2}$$

So $\lambda = 333(1/3\text{mills})$

$$E(X^2) = var(X) + \mu^2 = 2/\lambda^2 = 1.80(10^{-5})$$

$.4 = n(3(10^{-3})$ and so $n = 133$. TW proportion is $2000/2133 = .938$ and LP 133/2133 =.062 Finally

$$E(s^2) = .938(.25(10^{-3}))^2 + .062(1.8(10^{-5}))^2 = 1.174(10^{-6})$$

$$W_q = \frac{\lambda E(s^2)}{2(1-\rho)} = 2133(1.174(10^{-6}) = 2.5(10^{-3})$$

$$\frac{W_q}{E(s)} = 2.5(10^{-3}/.25(10^{-3} = 10$$

## 16.5 TW at .5 and LP utility with fixed 1 mill service time using .4

$$E(s^2) = .833(.25(10^{-3}))^2 + .167((10^{-3}))^2 = 2.19(10^{-7})$$

$$W_q = \frac{\lambda E(s^2)}{2(1-\rho)} = 2400(2.19(10^{-7}) = .525(10^{-3})$$

$$\frac{W_q}{E(s)} = 525(10^{-3}/.25(10^{-3} = 2.1$$