z/OS SMT: Understanding the Measurements and Applicability

Scott Chapman
Enterprise Performance Strategies

November 2018
Session LL

Contact, Copyright, and Trademark Notices

Questions?

Send email to Scott at scott.chapman@EPStrategies.com, or visit our website at http://www.epstrategies.com or http://www.pivotor.com.

Copyright Notice:

Trademarks:

EPS presentations this week

What	Who	When	Where
Mainframe Performance Metrics and Observations in the Real World	Scott Chapman	Tue 10:30	Woodcote
z/OS SMT: Understanding the Measurements and Applicability	Scott Chapman	Wed 12:00	Woodcote

Like what you see?

- The z/OS Performance Graphs you see here come from Pivotor™ but should be in most of the major reporting products

- If not, or you just want a free cursory review of your environment, let us know!
  - We're always happy to process a day's worth of data and show you the results
  - See also: http://pivotor.com/cursoryReview.html

Agenda

- Why and What is Simultaneous Multi-Threading
- Terminology (new and re-named)
- Measurements
  - Names
  - Meanings
  - Sources
- When to investigate SMT
  - Existing measurements to evaluate
  - Decision tree
  - Recommendations

Red words are key points to pay attention to

We can’t work faster…

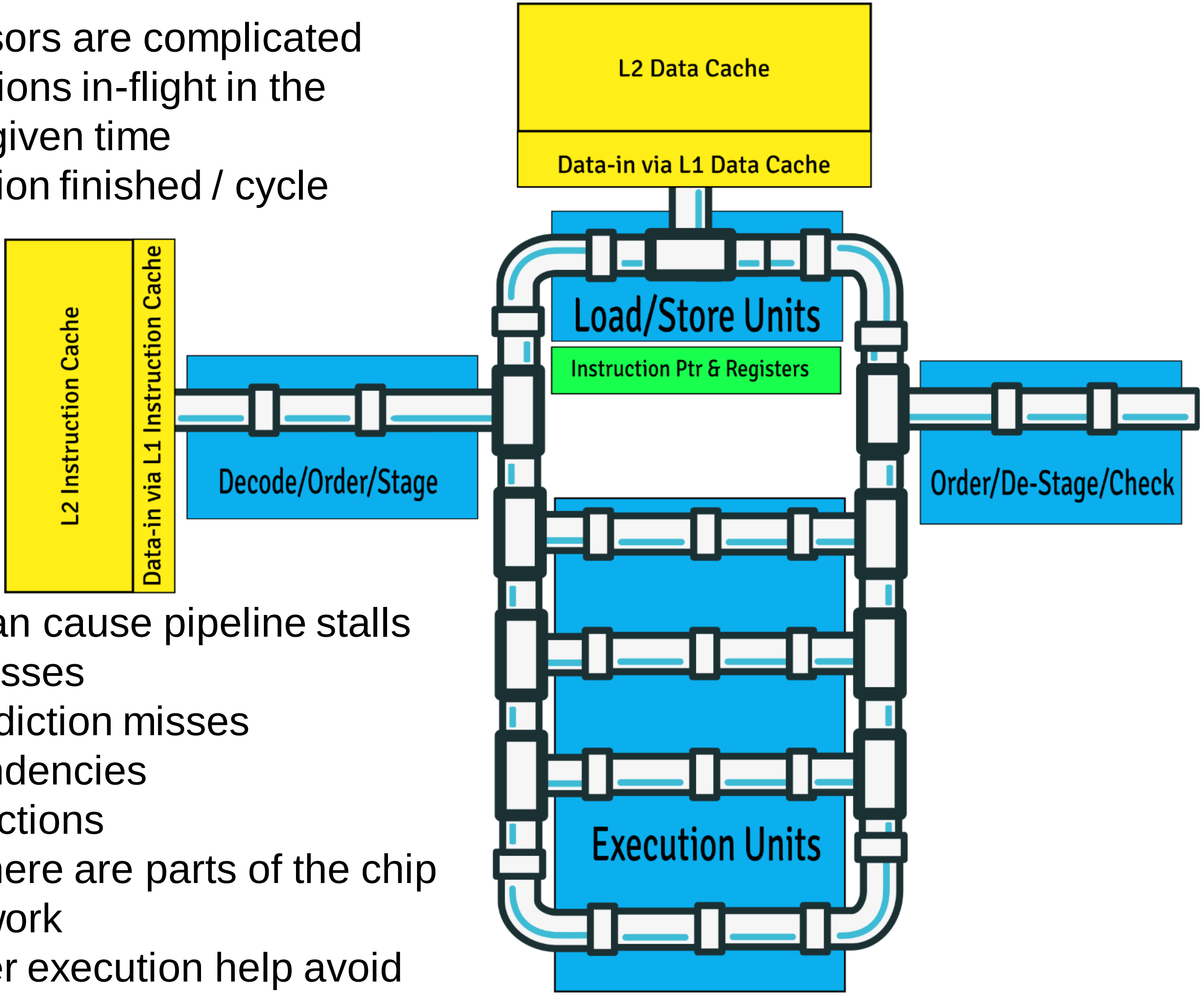
- Processor clock speeds aren’t going to continue increasing dramatically
  - Higher speeds generally mean higher voltages & heat
  - Higher voltage & heat = more risk of damaging the chip

- Performance has to come from somewhere else

- Don’t work harder, work smarter
  - Every new generation of processor adds new instructions
  - Increasing cache sizes improve throughput
  - But… you can’t spell “SMARTER” without SMT!

Modern Superscalar Processors

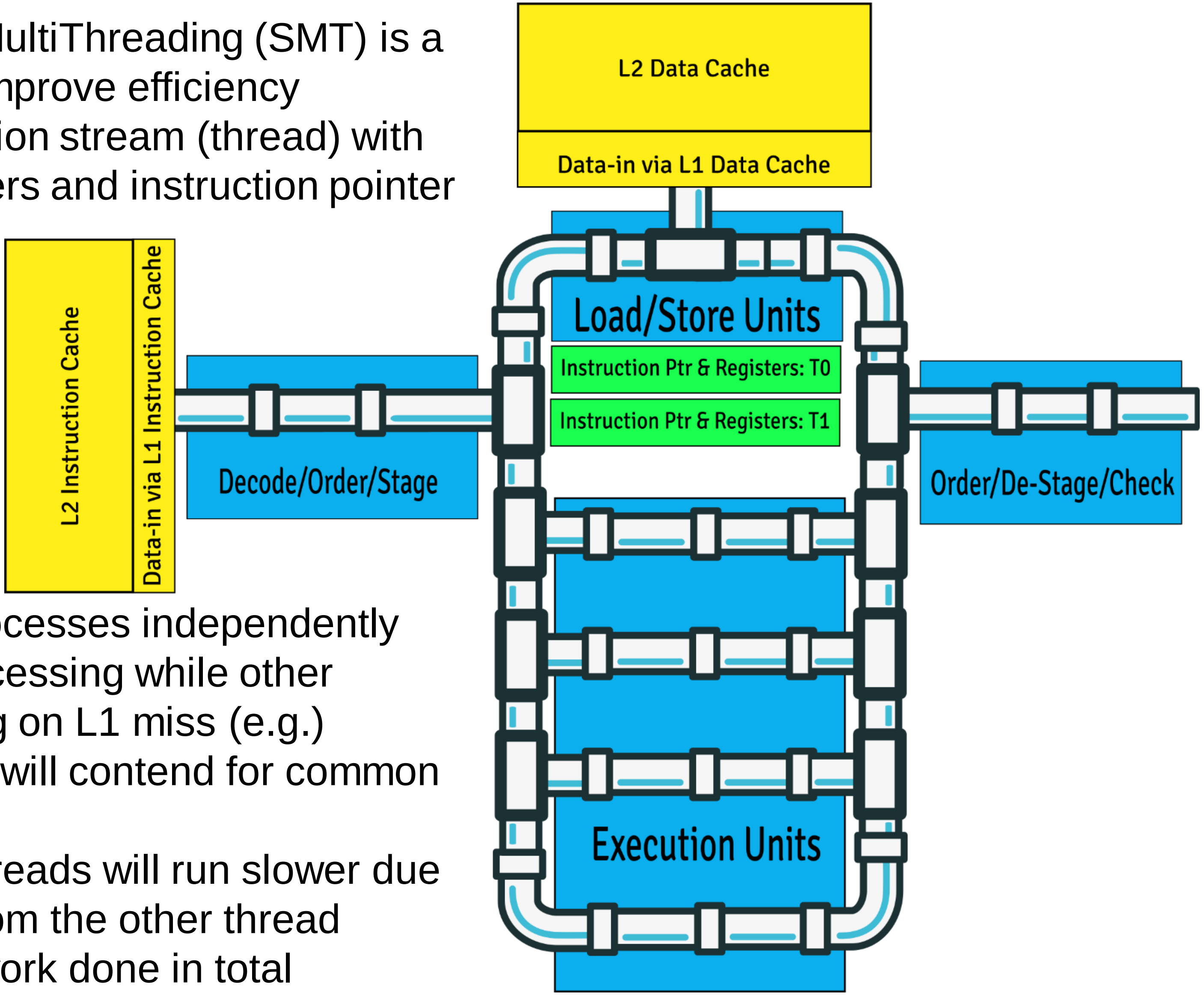
- Modern processors are complicated
- Multiple instructions in-flight in the pipeline at any given time
- Aim: 1+ instruction finished / cycle



- Lots of things can cause pipeline stalls
  - L1 / TLB misses
  - Branch prediction misses
  - Data dependencies
  - Long instructions
- When stalled, there are parts of the chip idle, not doing work
  - Out-of-order execution help avoid pipeline stalls

Modern Superscalar Processors w/ SMT

- Simultaneous MultiThreading (SMT) is a way to further improve efficiency
- Second instruction stream (thread) with separate registers and instruction pointer



L2 Data Cache
Data-in via L1 Data Cache
L2 Instruction Cache
Data-in via L1 Instruction Cache
Decode/Order/Stage
Load/Store Units
Instruction Ptr & Registers: T0
Instruction Ptr & Registers: T1
Order/De-Stage/Check
Execution Units

- Each thread processes independently and can be processing while other thread is waiting on L1 miss (e.g.)
- But the threads will contend for common resources
- So individual threads will run slower due to contention from the other thread
    - But more work done in total (hopefully)

SMT Industry History

- Sun patented the idea in 1994
  - Although much research pre-dated this, going back to IBM in the late 1960s
- Intel’s HyperThreading introduced 2002
- Power5 introduced SMT2 in 2004
- Power7 (2010) has SMT4
- Power8 (2013) has SMT8
- z13 (2015) & z14 (2017) are SMT2
- Regardless of platform: SMT has the potential to increase total system throughput, but at the possible expense of individual thread throughput
  - SMT is a more/slower vs. fewer/faster type of consideration

SMT Enablement Requirements

- LOADxx: PROCVIEW CORE{,CPU_OK}
  - Enables multithreading mode for life of IPL (but doesn’t activate it)
  - Must IPL to set this
  - With “,CPU_OK” output of D M=CPU changed to be core-centric
  - Without “,CPU_OK” have to change to D M=CORE

- IEAOPTxx: MT_ZIIP_MODE=1|2
  - Indicates how many threads to use for zIIP
  - SET OPT=xx to switch MT on/off dynamically (without IPL)

What changes with SMT-2 activation?

- HIPERDISPATCH=YES forced
  - Best choice for majority of use cases anyway
- WAITCOMPLETION=YES will disallow MT2 activation
  - Almost certainly shouldn't have this set anyway
- D M=CPU vs CORE
- CF CPU vs CORE
- SMF CPU ID
  - In some records, GCPs now are 0,2,4,6… while zIIPs are 0,1,2,3…
- Achieved Velocity will likely change for workloads using zIIP
  - Effectively, SMT2 = more/slower zIIPs vs. fewer/faster
  - Re-evaluate your WLM goals after implementation
- Reported zIIP time is MT1ET (more on this later)

But before you explore…

- Understand the measurements

- But the measurements are not well documented
  - Or in some cases, at all

- So this is my effort to make sense of the measurements

SMT Terminology and Measurements

SMT Terminology (Part 1, easy)

- CPUs -> Threads
  - z/OS dispatches work on threads / CPUs
  - With SMT: two threads per core

- Cores -> Cores
  - PR/SM dispatches logical core on physical core

- Core Busy
  - Time core is executing instructions
    - I.E. CPU Busy
    - Regardless of the number of potential or active threads

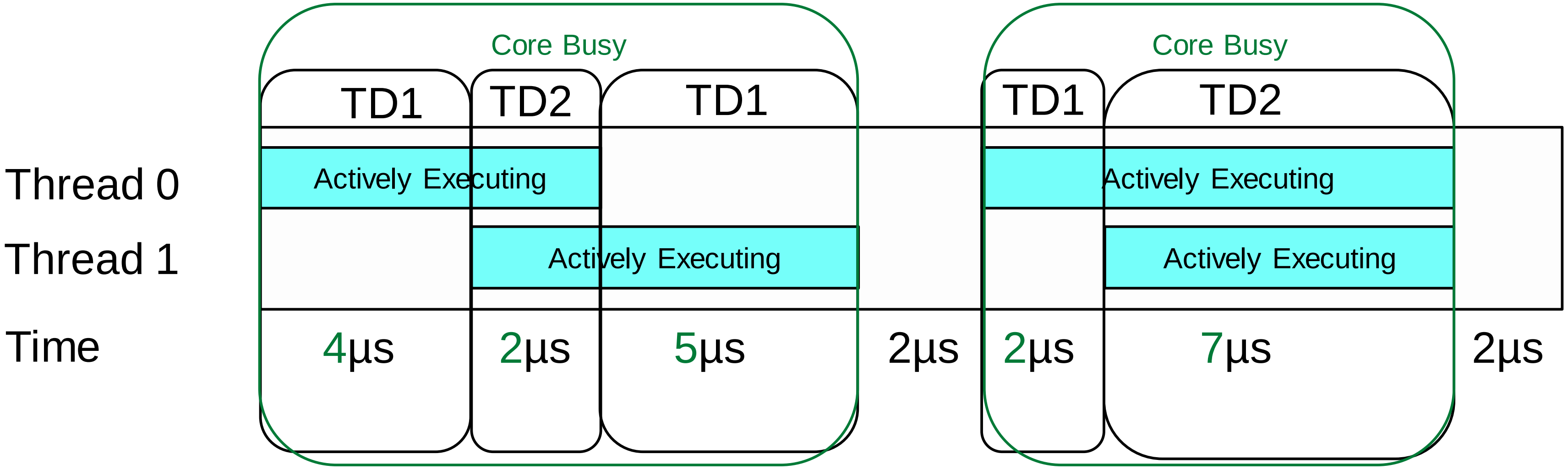
Note on following example scenario

- Example scenarios made up for illustrative purposes only
- Example shows a small slice of time (24 microseconds)
  - Done to show that things change quickly and constantly
  - In reality, samples and reporting are over much greater time periods
- In reality, the calculations deal with cycle counts instead of time
  - But cycle time is constant (absent hardware failure) so a cycle count is a measure of time
- Some accounting and calculation details are inferred from IBM sources but are apparently not documented in detail
  - Some informally verified by some IBMers
  - I’d greatly appreciate it if somebody could point me to the formal documentation!

SMT Terminology (Part 2, slightly harder)

- Average Thread Density (TD)
  - Average number of executing threads, when at least one thread is executing (i.e. the core is busy) = $\frac{TD1\ time + (TD2\ time*2)}{TD1\ time + TD2\ time}$
    - 1 = Never had more than a single thread executing
    - 2 = Always had two threads executing when one was executing

Note: cycle counts equate to time



Average TD = (4*1 + 2*2 + 5*1 + 2*1 + 7*2) / (4+2+5+2+7)

TD1 = 4+5+2 = 11µs TD2 = 2+7 = 9µs

Average TD = (11+9*2) / (11+9) = 29 / 20 = 1.45

Core Busy = (4+2+5 + 2+7)/(4+2+5 + 2 + 2+7 +2) = 20 / 24 = 83.3%

Values Externalized by RMF

SMT Terminology (Part 2, bonus)

- Given average thread density and core busy time, we can calculate the time at TD1 & TD2
- Core busy time (CBtm) = $Core\ Busy\ \% \ *\ Interval\ Time$
- TD1tm = $CBtm\ *\ 2 - CBtm\ *\ TD$
- TD2tm = $CBtm - TD1tm$
  - Note that these are total CPU times for the interval in question
- Given: Interval time = 24, CB% = 83.3%, TD = 1.45
  - $CBtm\ =\ 24\ *\ .833\ =\ 19.992$
  - $TD1tm\ =\ 19.992 * 2 - 19.992 * 1.45\ =\ 39.984 - 28.9884\ =\ 10.9956 \sim 11$
  - $TD2tm\ =\ 19.992 - 10.9956\ =\ 8.9964 \sim 9$

Example from previous slide

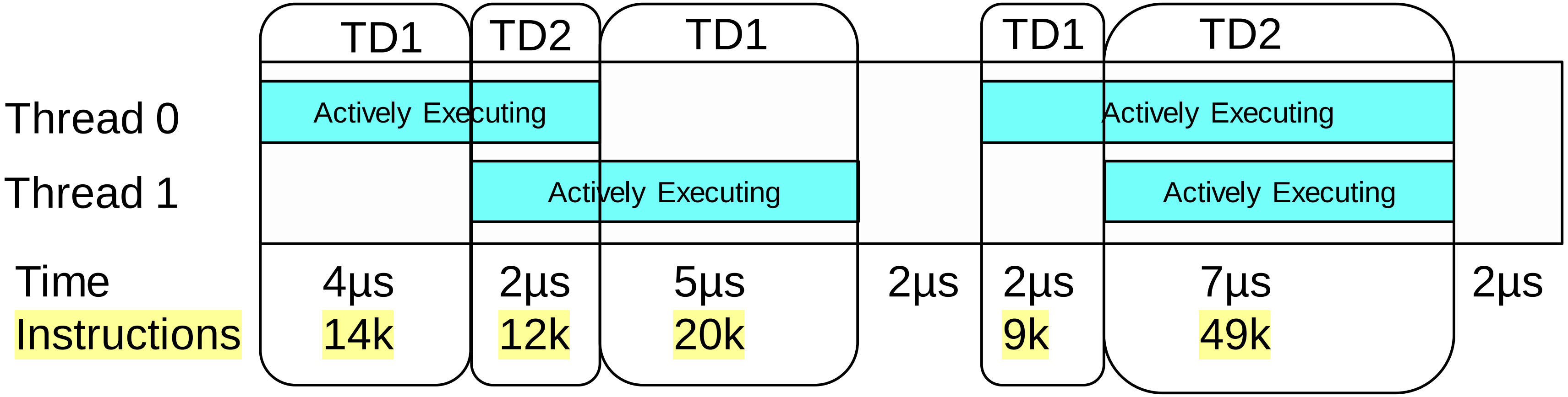
SMT Terminology (Part 3, confusing)

- Capacity Factor (CF)
  - How much work completes in total relative to TD1
  - That is, how much benefit are you getting from SMT
  - $CF = \frac{Total\ Work\ Rate}{TD1\ Work\ Rate}$
    - "Work Rate" is really Instructions Per Cycle
      - Instruction count and cycle count at TD2 & TD1 captured by HIS
- Maximum Capacity Factor (mCF)
  - Estimate of maximum work a core could complete (using both threads)
  - $mCF = \frac{TD2\ Work\ Rate}{TD1\ Work\ Rate}$
- Productivity
  - Estimate of completed work vs work that could be completed at TD2
  - $Productivity = \frac{CF}{mCF}$

Note: CF & mCF limited to 2

Capacity Factor Example

- Revisiting previous example



	TD1	TD2	TD1		TD1	TD2	
Thread 0	Actively Executing	Actively Executing			Actively Executing	Actively Executing	
Thread 1		Actively Executing	Actively Executing			Actively Executing	
Time	4µs	2µs	5µs	2µs	2µs	7µs	2µs
Instructions	14k	12k	20k		9k	49k	

TD1 Work Rate = (14k+20k+9k) / (4+5+2) = 43k / 11µs = 3,909 inst/µs
TD2 Work Rate = (12k+49k) / (2+7) = 61k / 9µs = 6,778 inst/µs
Total Work Rate = (43k+61k)/(11µs+9µs) = 104k / 20µs = 5,200 inst/µs

(Above possible internal calculations not documented)

Capacity Factor = 5,200 iµs / 3,909 iµs = 1.33
Maximum Capacity Factor = 6,778 iµs / 3,909 iµs = 1.73
Productivity = 1.33 / 1.73 = 76.9%

Values Externalized by RMF

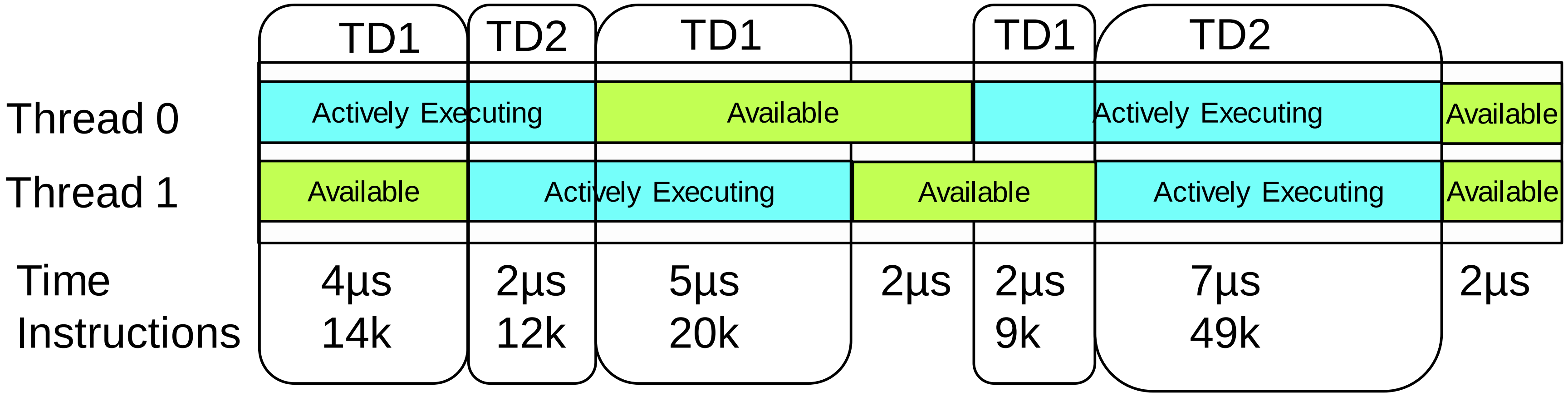
Core Utilization: how busy is the CPU??

- Understanding utilization is more complicated with SMT

- Remember:
  - Core Busy = time core is executing instructions in either TD1 or TD2
    - When not in SMT, busy % = utilization %
    - If executing instructions at TD1, even though the core is “busy” it could still do more work
  - Productivity = Estimate of completed work vs work that could be completed at TD2

- Core Utilization % = Core Busy % * Productivity %

Core Utilization Example



Core Busy = (4+2+5 + 2+7)/(4+2+5 + 2 + 2+7 +2) = 20 / 24 = 83.3%
Capacity Factor = 5,200 iµs / 3,909 iµs = 1.33
Maximum Capacity Factor = 6,778 iµs / 3,909 iµs = 1.73
Productivity = 1.33 / 1.73 = 76.9%

Core Utilization = 83.3% * 76.9% = 64.1%

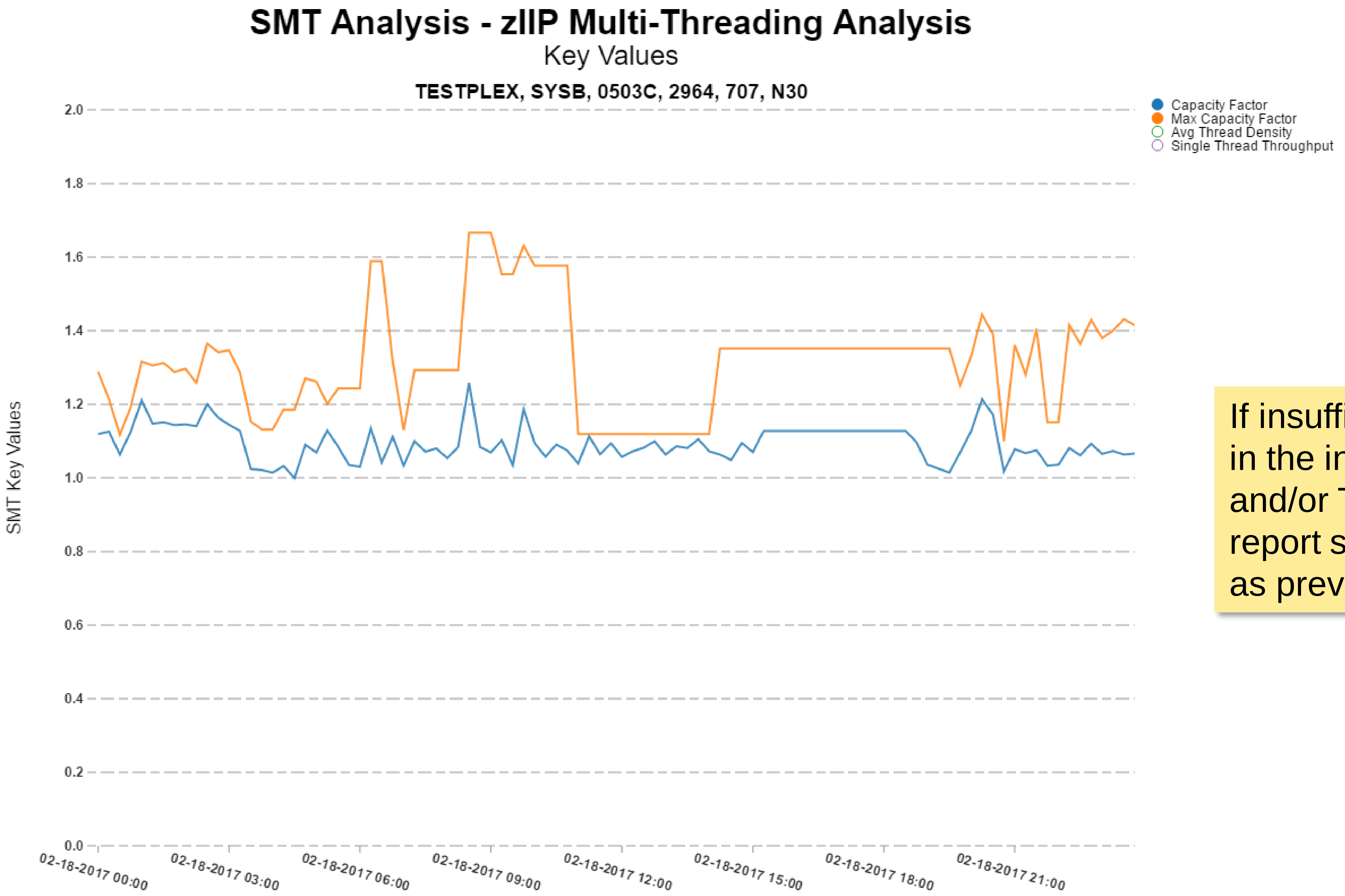
Values Externalized by RMF

Capacity Factor Issues

- Capacity Factor and mCF are key metrics to evaluate, and are the basis for additional measurements, but…
- Exact calculations and methodology undocumented
  - Public presentations and US Patent Application US 20150277985 A1 suggest the previous example calculations are correct (but possibly more complicated in at least some cases)
- Measurements are really estimates
- Calculation is based on instructions completed per unit of time
  - CPI can be variable based on a variety of factors:
    - Instruction mix
    - Instruction arrival pattern
    - Cache contention with other workloads
    - Memory access patterns

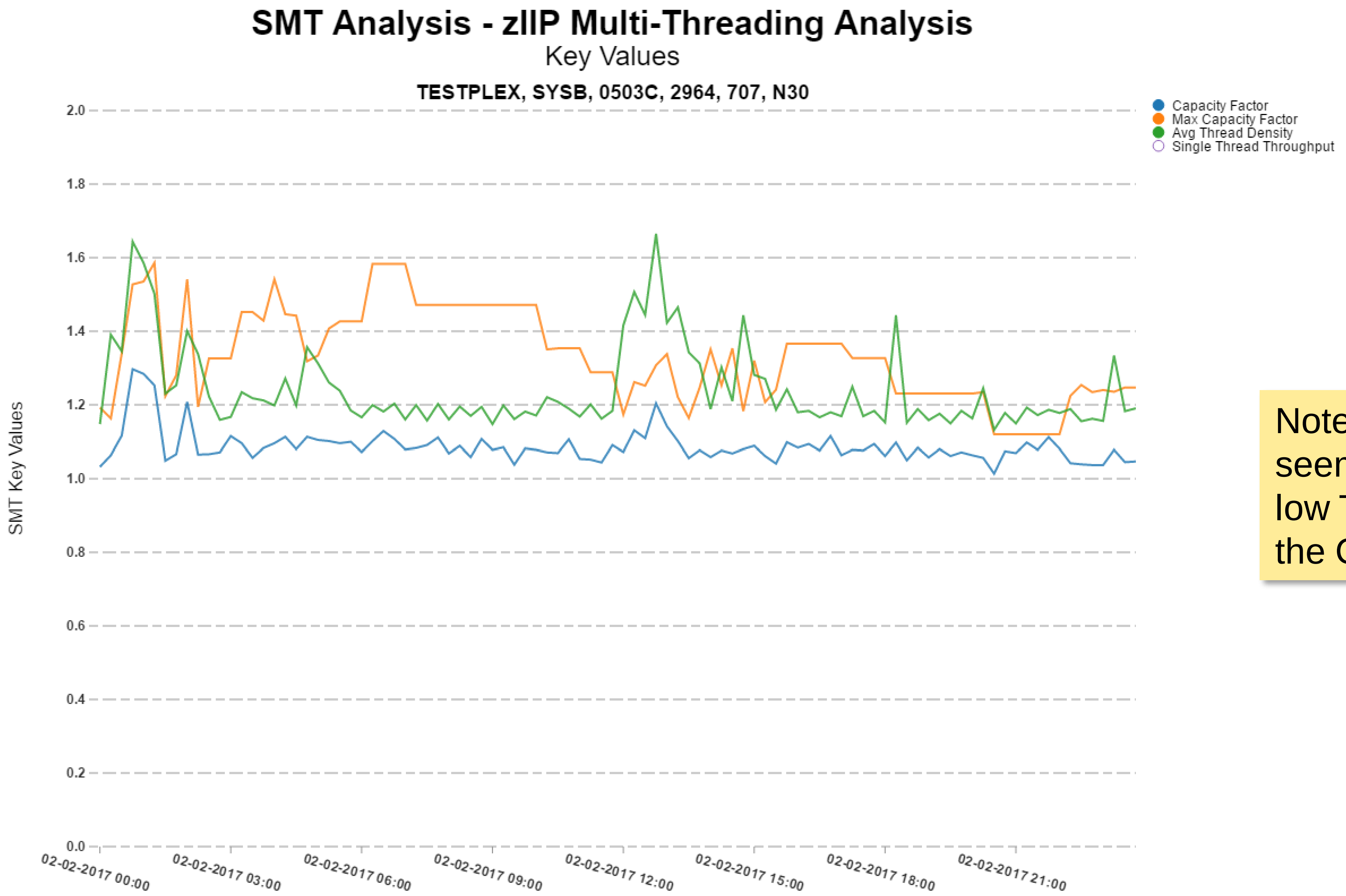
But it's what we have, and it's hard to conceive of a better solution

Result of insufficient work samples



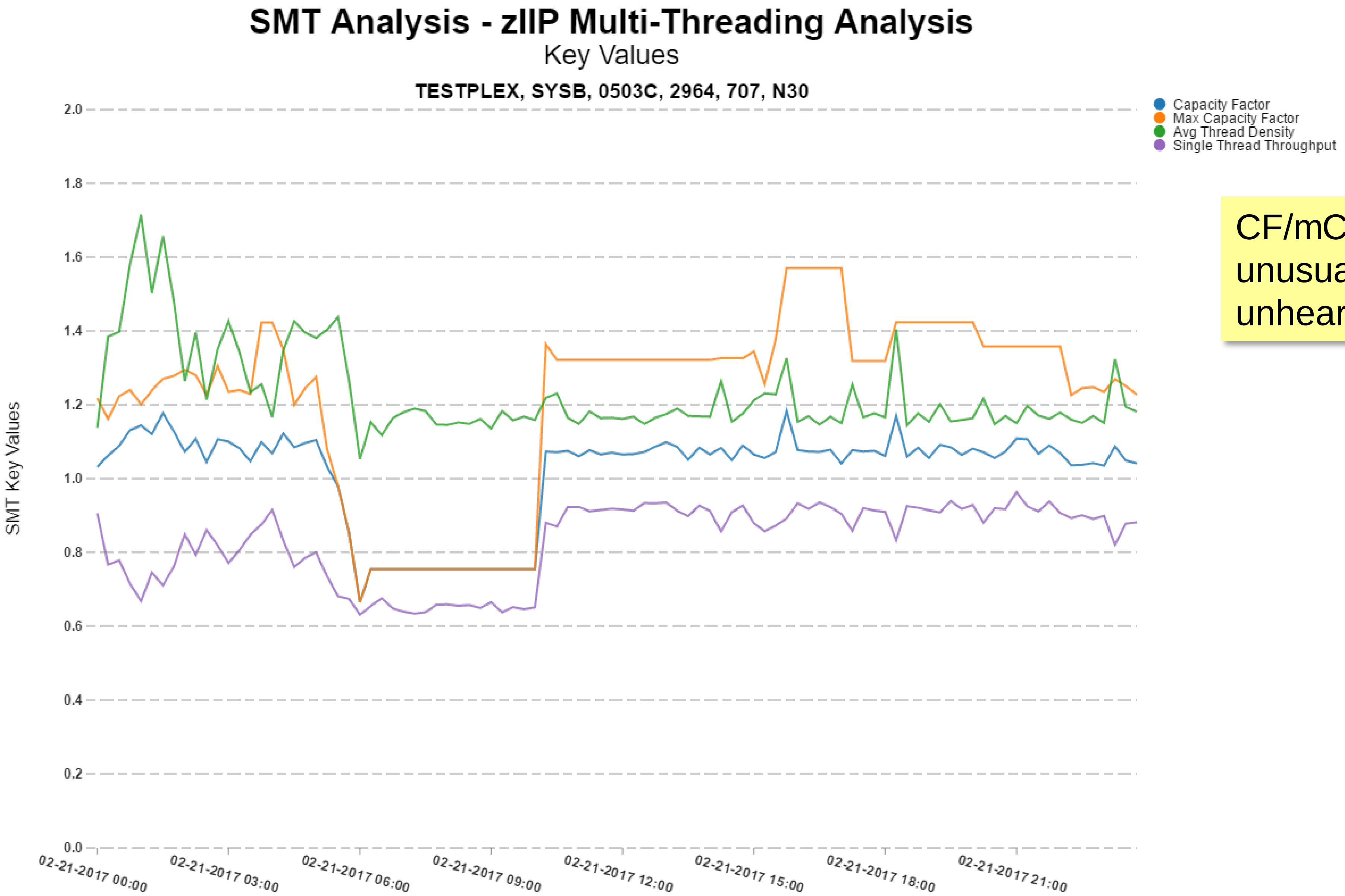
If insufficient samples in the interval at TD1 and/or TD2, RMF will report same CF, mCF as previous interval

Another common example



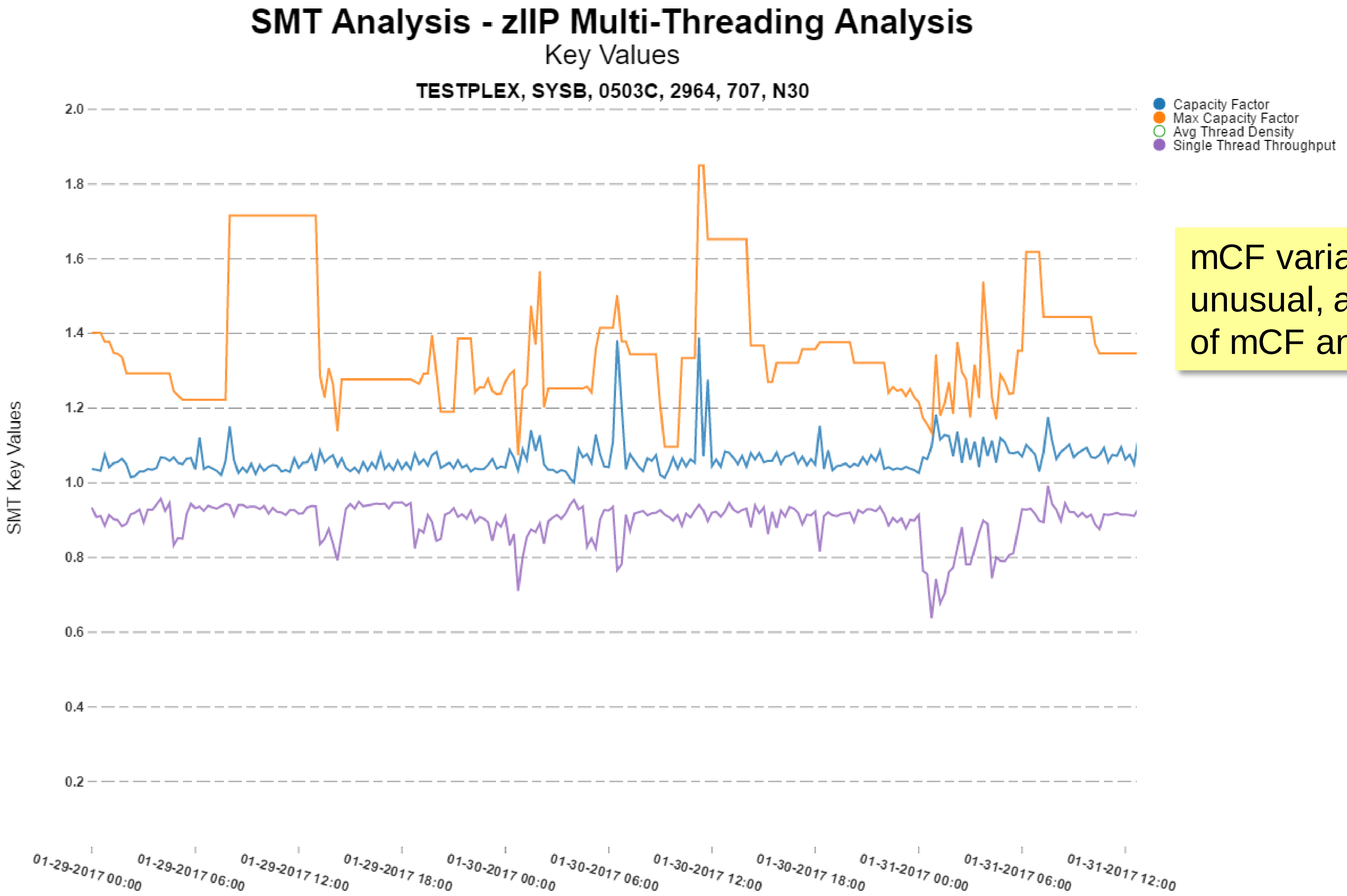
Note that the mCF seems to flatline at low TD even when the CF is variable

A less common example



CF/mCF < 1 somewhat unusual but not unheard of

mCF tends to be more variable than CF



mCF variability not unusual, as is flat-lining of mCF and/or CF

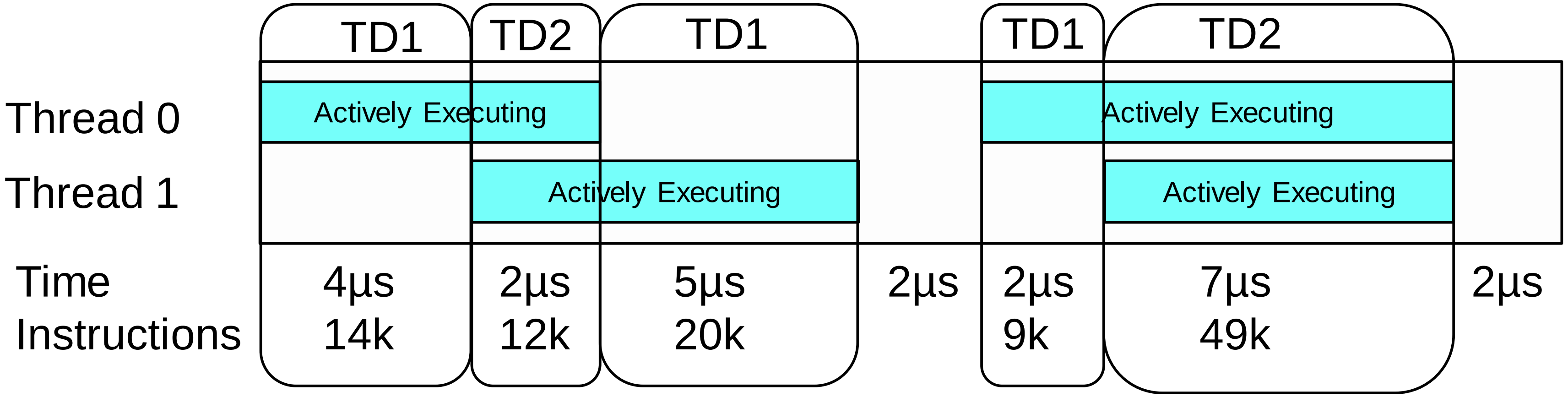
SMT Terminology (Part 4, what the…??)

- Multi-Threading 1 Equivalent Time (MT1ET)
  - CPU time work would have consumed, had the work run without SMT involved
  - All SMF/RMF CPU time measurements are normalized to MT1ET
    - e.g. on the Workload activity report
    - What "normalized" means is unclear
      - "scaled by CF, distributed to threads based on TD"
      - "RMF divides retrieved time values through the mCF"
      - Patent Application US 20150277984 A1 shows $chargeback\ factor\ = \frac{CF}{TD}$
  - Note that MT1ET should generally be less than the raw value
    - No SMT = "fewer/faster" = generally lower CPU time

- $zIIP\ Appl\% = \frac{zIIP\ MT1ET}{Interval\ Duration * mCF}$

MT1ET Example (maybe, probably; CBF based)

- Revisiting previous example
  - (assume two threads = two work units)



Capacity Factor = 5,200 iµs / 3,909 iµs = 1.33

Maximum Capacity Factor = 6,778 iµs / 3,909 iµs = 1.73

Average TD = (11+9*2) / (11+9) = 29 / 20 = 1.45

CBF = 1.33 / 1.45 = .9172

T0 CPU = 4+2+2+7=15µs raw CPU=15 * 0.9172 = 13.76 MT1ET CPU

T1 CPU = 2+5+7 = 14µs raw CPU=14 * 0.9172 = 12.84 MT1ET CPU

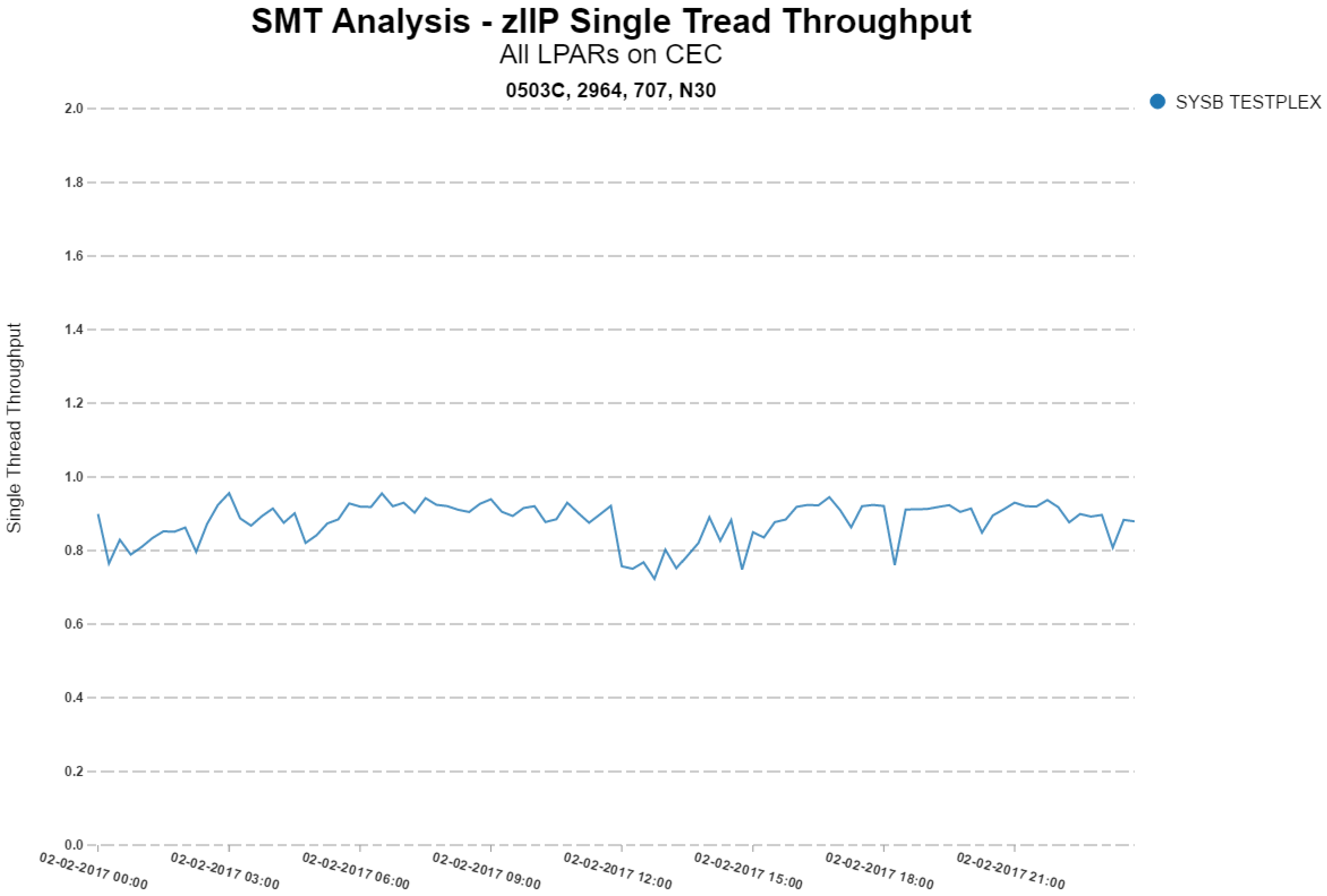
IBM patent application for "chargeback factor" (CBF)

I don't know if these calculations / results are correct!! (probably close)

So how is MT1ET calculated??

- I don’t know for sure
- I’m 99% certain that it’s close to what was shown here
  - Patent application for the chargeback factor supports this
  - HISMT service returns a Chargeback Factor (data area HISYMT)
  - But we don’t know if that CBF is calculated the same as indicated in the patent
  - Prior to investigating MT1ET and discovering CBF, I was discussing the average single thread impact with a customer and we came to the realization that CF/TD would be indicative of the average single thread throughput (ASTT)
    - And it seems reasonable that ASTT * CPU would approximate MT1ET
  - Backing into a total Raw CPU time from TD, CF, and busy, then converting to total MT1ET and comparing to total of all SMF 72 MT1ET comes out reasonably close (>97% capture ratio for some sample data I processed)
  - But it’s also possible that the CBF calculation is more “nuanced”

Average Single Thread Throughput (CBF?)



Wait a minute…

- You may be thinking, this CPU and capacity measurement is all a bit imprecise with SMT enabled
- You'd be correct
  - Reported CPU time now MT1ET
    - MT1ET is apparently based on CF
  - % Utilization based on Core Busy and Productivity
    - $Productivity\ = \frac{CF}{mCF}$
  - zIIP Appl% based on MT1ET and mCF
  - mCF is an estimate
  - CF & mCF are based on "work completed" at TD1 & TD2
    - "work completed" not defined (but is understood to be instructions / cycle)
    - likely variable with workload mix and many other factors
- To those of us used to precise measurements, this feels wrong

When does it make sense to investigate SMT?

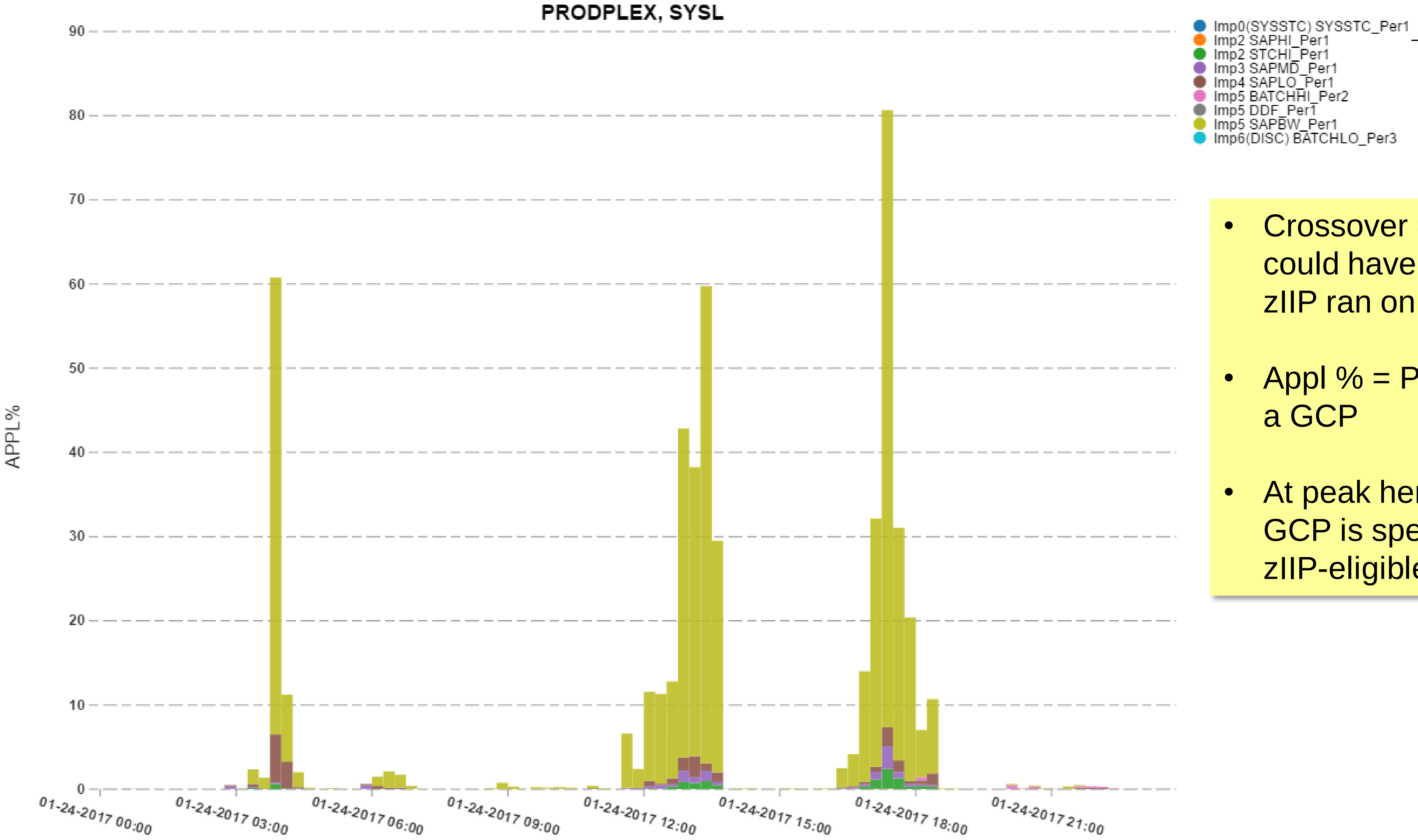
When does it make sense to investigate SMT?

- We can’t predict SMT impact
- But there situations where more/slower CPUs is beneficial vs. fewer/faster
  - Especially if more/slower = more total capacity
- In some situations, SMT might:
  - Reduce performance
  - Increase performance
  - Lower GCP utilization (and maybe lower R4HA, and maybe lower MLC costs)
  - Avoid need for an upgrade
- We can’t really avoid the first, so let’s try to find situations where one of the last three scenarios might apply

First metric to evaluate: zIIP Crossover

zIIP APPL% Crossover CPU - Service Class Period

(normalized to CP CPU speed)



- Crossover = work that could have run on a zIIP ran on a GCP
- Appl % = Percentage of a GCP
- At peak here, 80% of a GCP is spent doing zIIP-eligible work

Why is crossover so important??

- GCP utilization drives R4HA and MLC costs are based on peak R4HA

- If crossover is significant during your peak R4HA, your MLC costs are higher than they would be if you had more zIIP capacity
  - All usual MLC savings caveats apply:
    - How many peaks do you have
    - How large is the peak vs. next highest interval
    - Where are you on the MLC price curve (10% R4HA reduction means <10% savings)

- If your GCPs are slower than your zIIPS, zIIP eligible work is not performing as well as if it was on the faster zIIPs

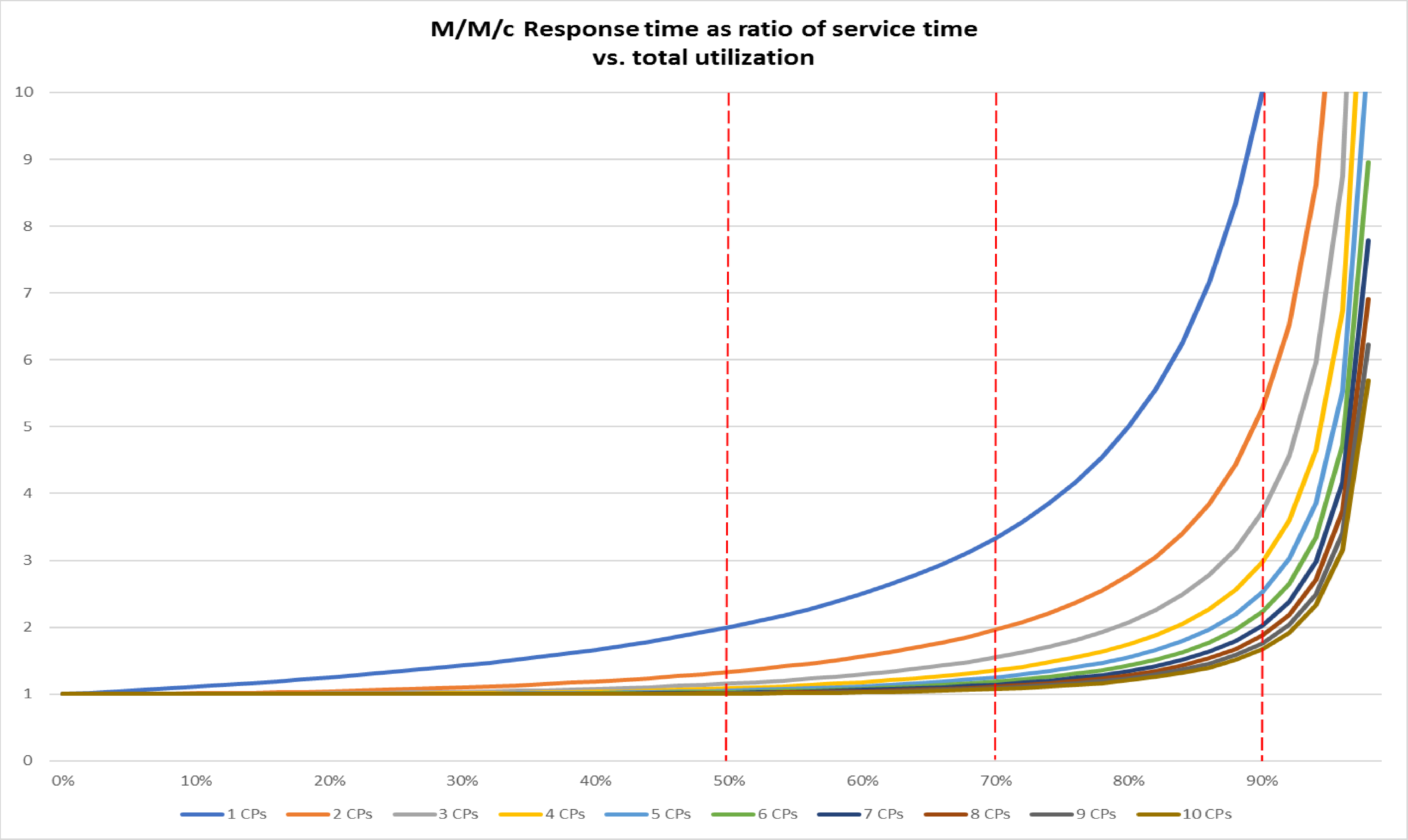
Avoid significant crossover

Potential Crossover solutions

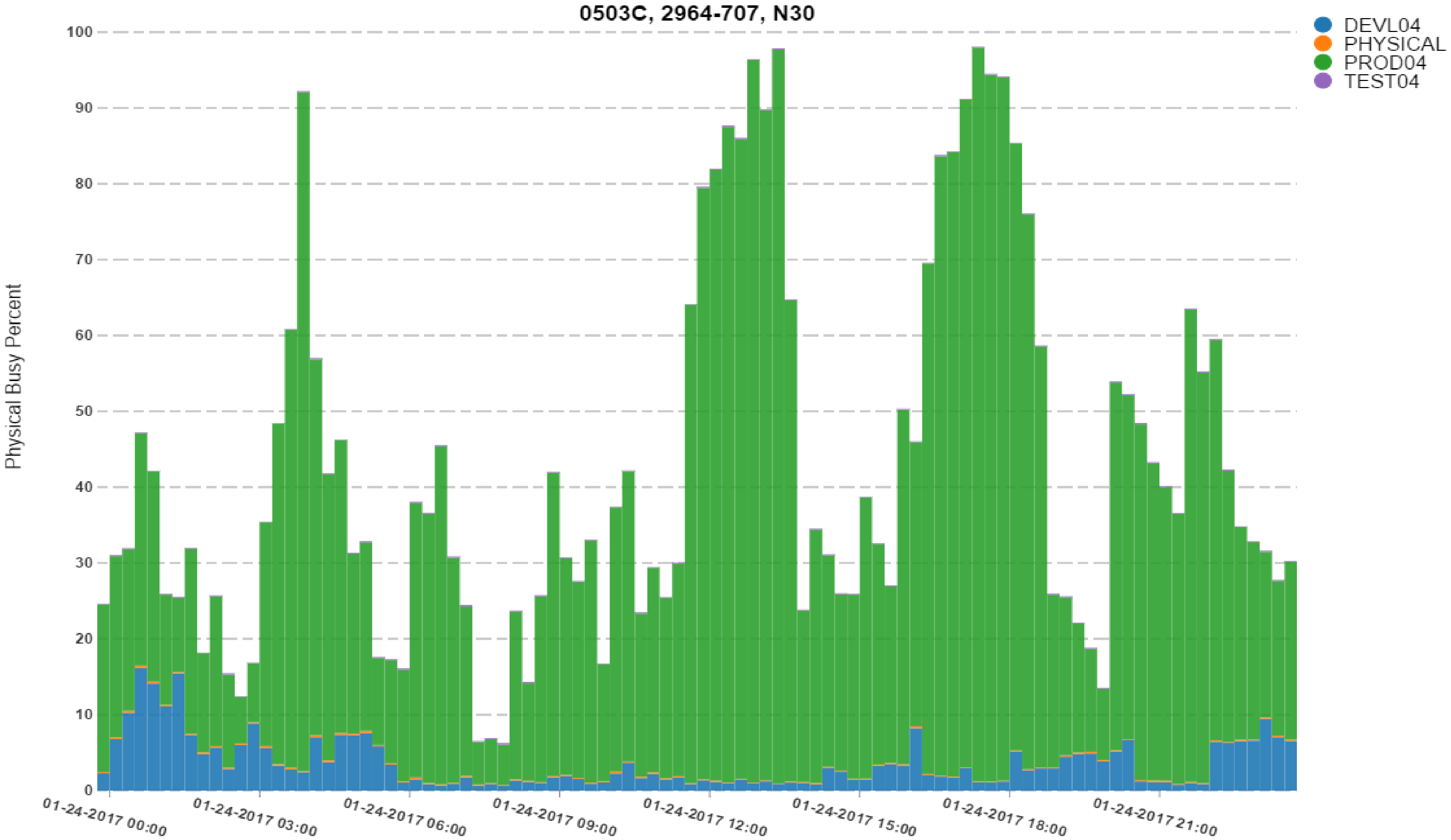
- Buy more zIIP(s)
  - Always best from a performance perspective
  - They may be cheap relative to the software costs they offset
- Use HONOR PRIORITY=NO on specific service classes
  - Prevents zIIP-eligible work in those service classes from crossing over to the GCPs
- Increasing ZIIPAWMT may be useful in some limited cases
  - Usually not recommended but we have seen situations where it was useful
- Run less zIIP work
  - This seems counter-productive: we’re generally heading towards more zIIP work, not less
- Enable SMT
  - More capacity, although likely impact to individual thread performance

What about zIIP Busy?

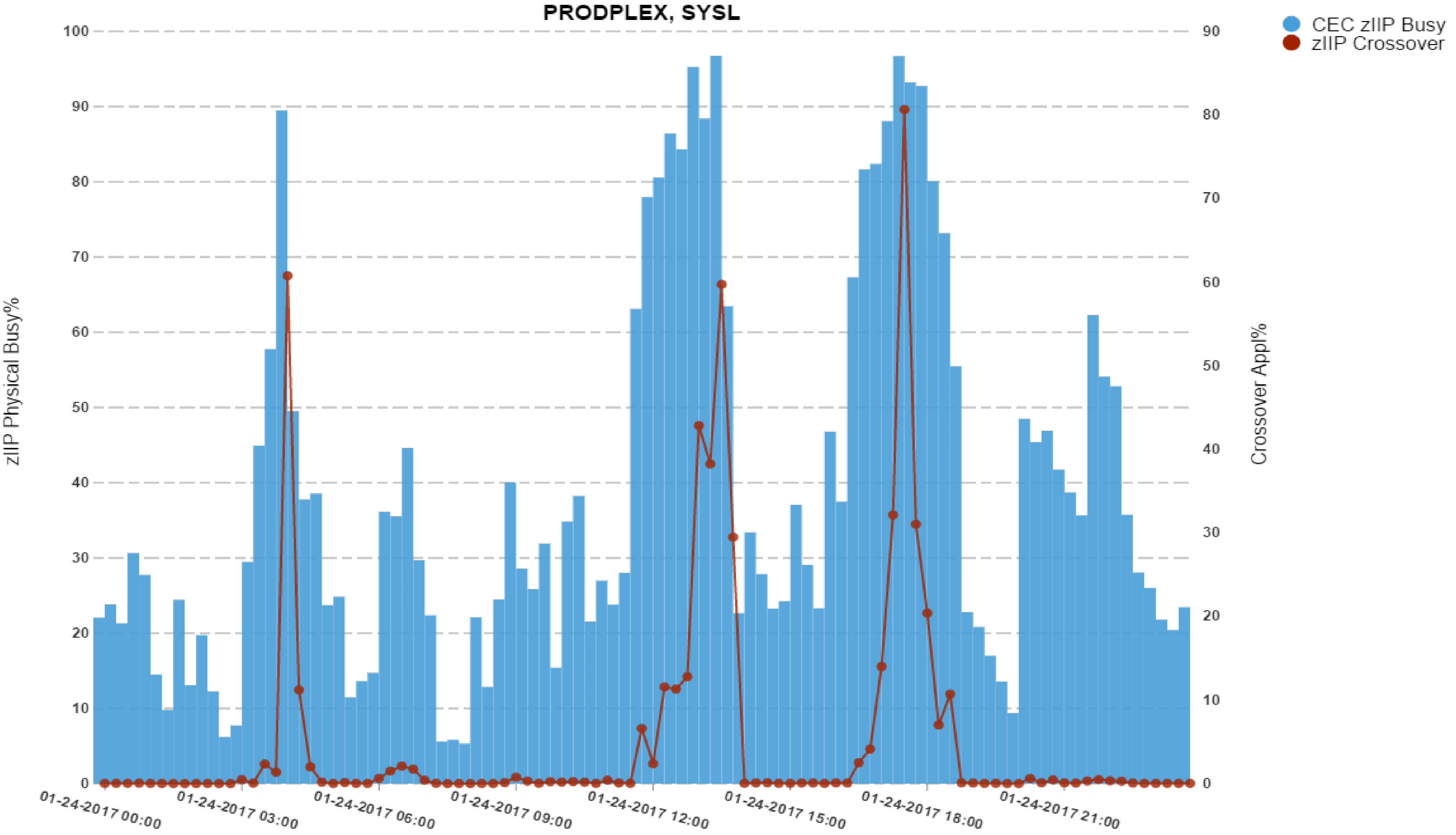
- You may hear something like “don’t run your zIIPs over 50% busy”
- Should you add capacity (more, or SMT) when they’re over 50%?
- From a performance perspective, less busy is always good
- From a financial perspective we want to make sure our resources are well-utilized
  - While one can generally afford to run zIIPs less busy than GCPs, they still aren’t free
- “Best possible performance” is usually not the financially prudent answer
  - But there will be some threshold at which the pain outweighs the financial cost
- The “too busy” threshold is dependent on the number of zIIPs
  - Queueing theory: For single “server” (CP) 50% busy means response time = 2x service time
  - But as server (CP) count goes up, that 2x pushes farther out


M/M/c Response time as ratio of service time
vs. total utilization
10
9
8
7
6
5
4
3
2
1
0
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
1 CPs
2 CPs
3 CPs
4 CPs
5 CPs
6 CPs
7 CPs
8 CPs
9 CPs
10 CPs

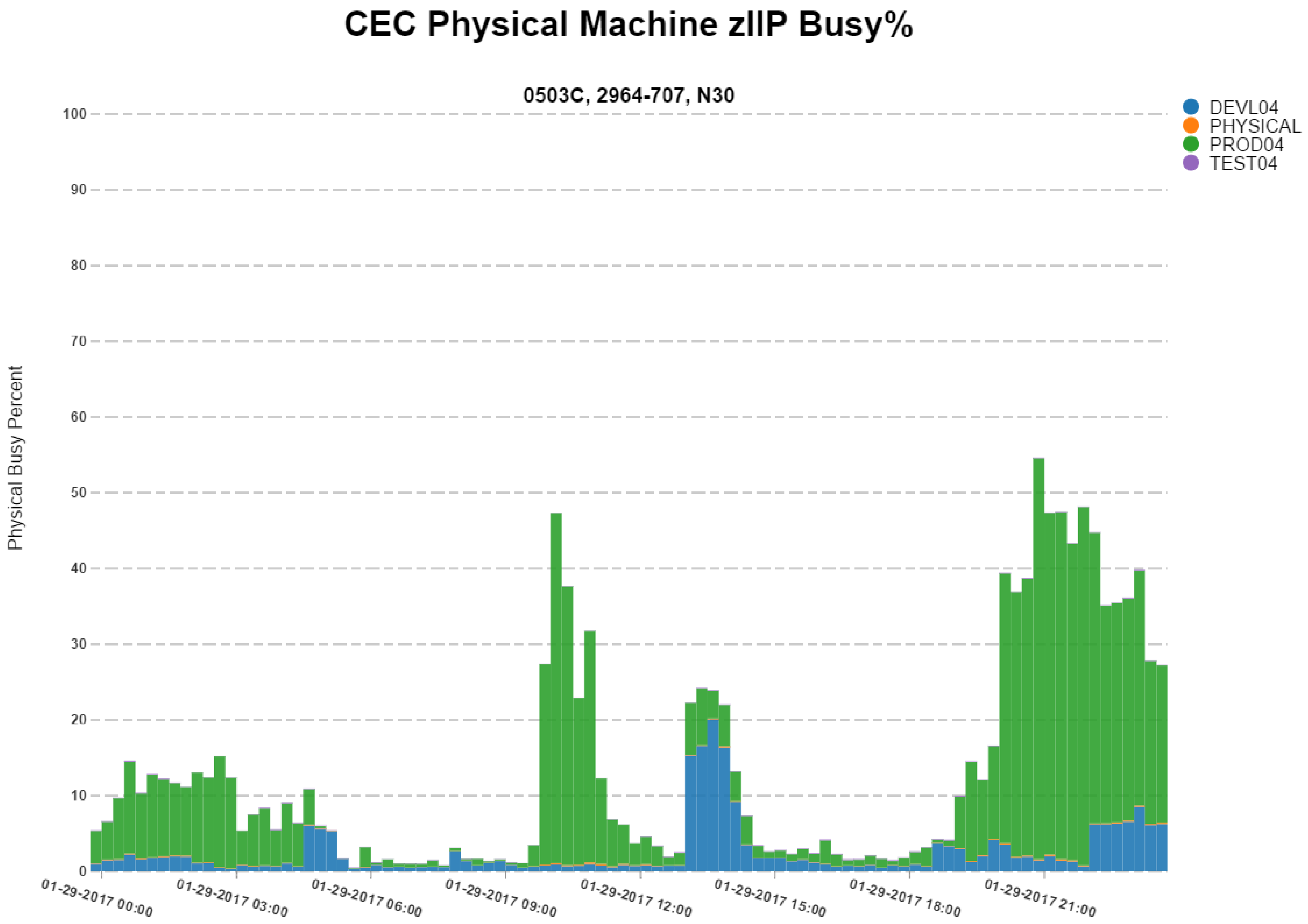
CEC Physical Machine zIIP Busy%



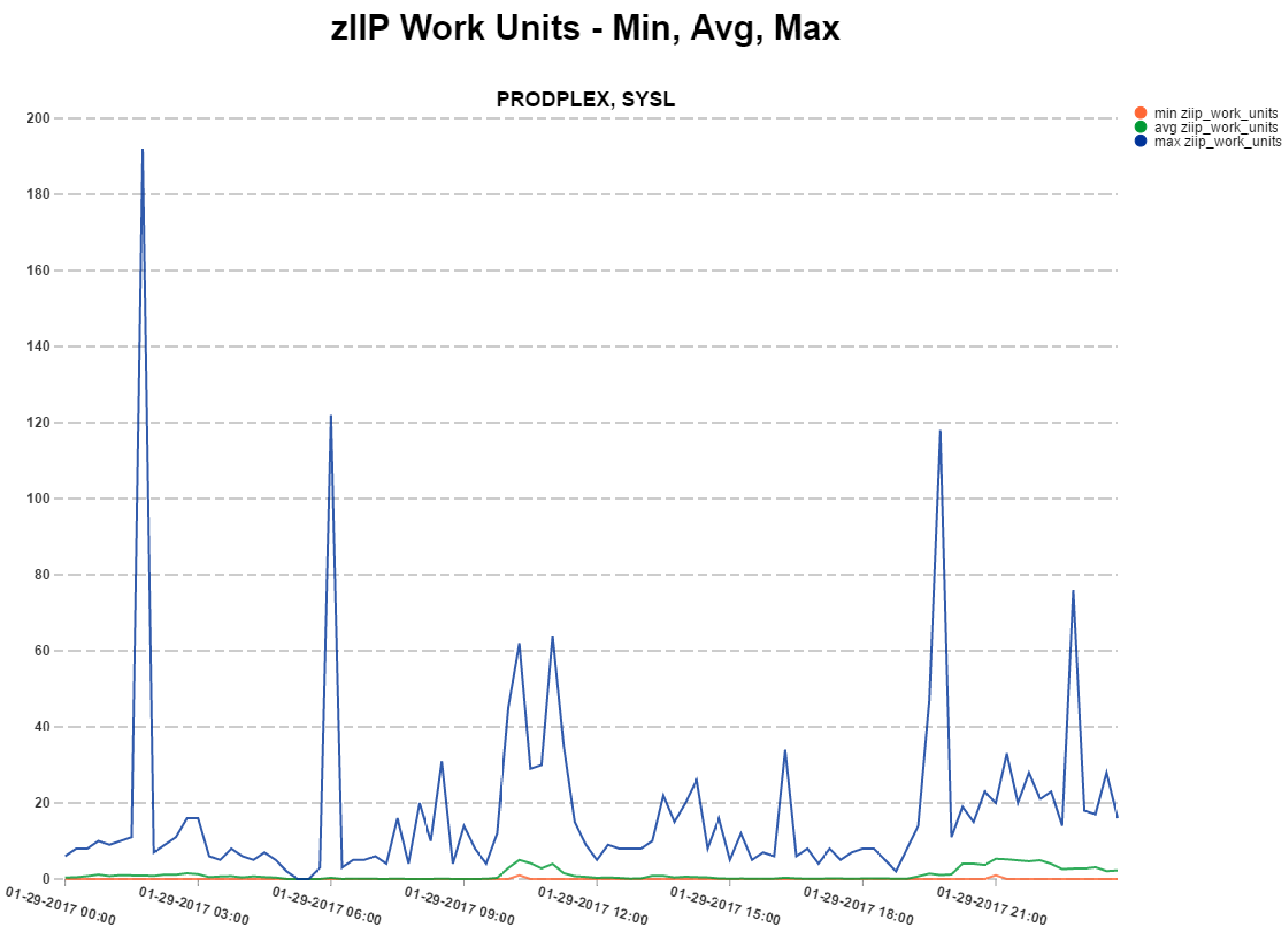
zIIP APPL% Crossover CPU vs. Physical Busy



zIIP Busy can indicate problems are unlikely



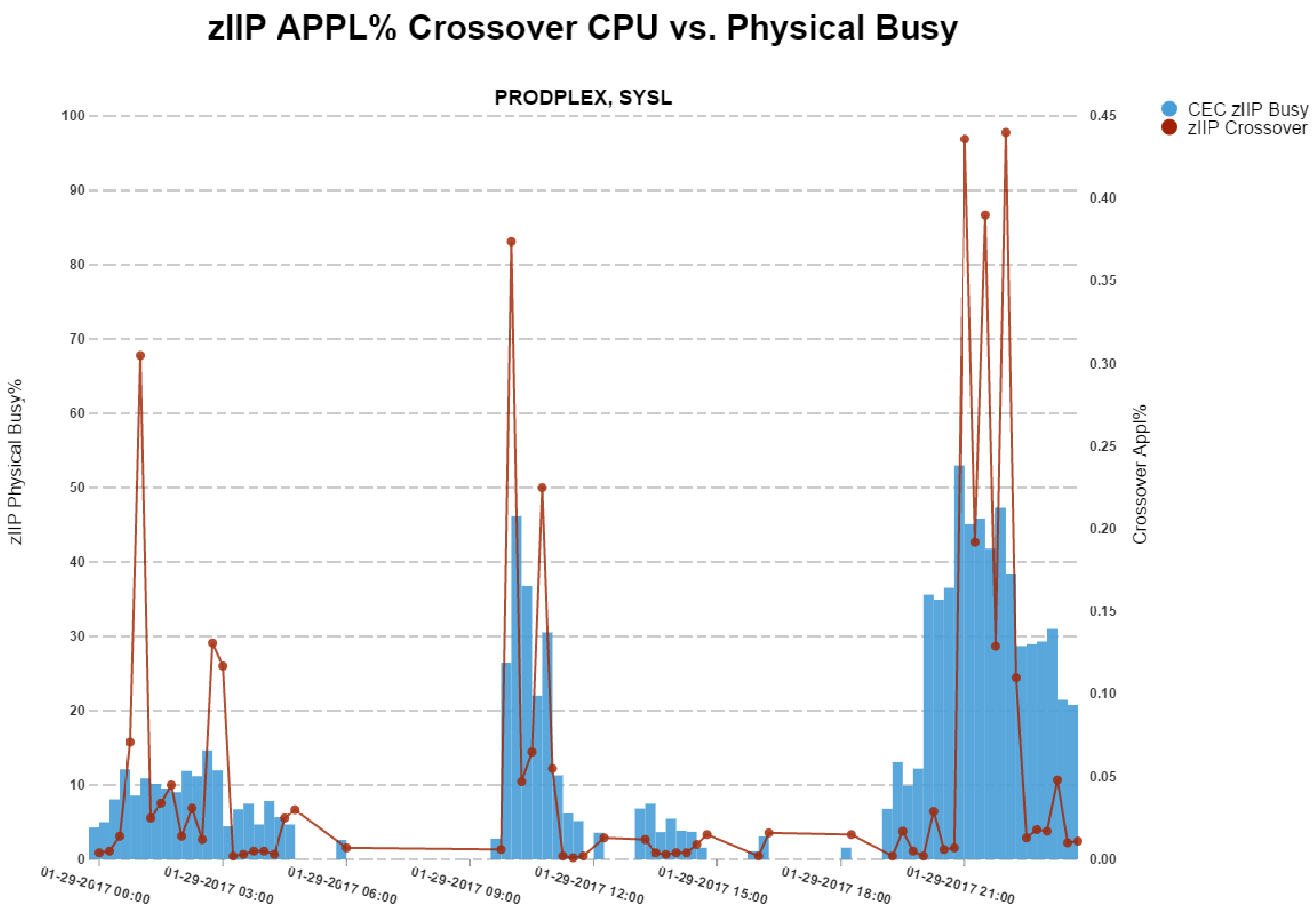
Unlikely is not impossible



zIIP Work Units

- Number of work units active on a zIIP or waiting for a zIIP
- Comes from SRM sampling
- Indication of instantaneous stress
- So on previous chart there was an instance when there were over 180 work units waiting to use a zIIP
  - But 15 minute average utilization was low
  - So quite likely the queue was very quickly drawn down

Crossover < 1% of GCP engine for that day



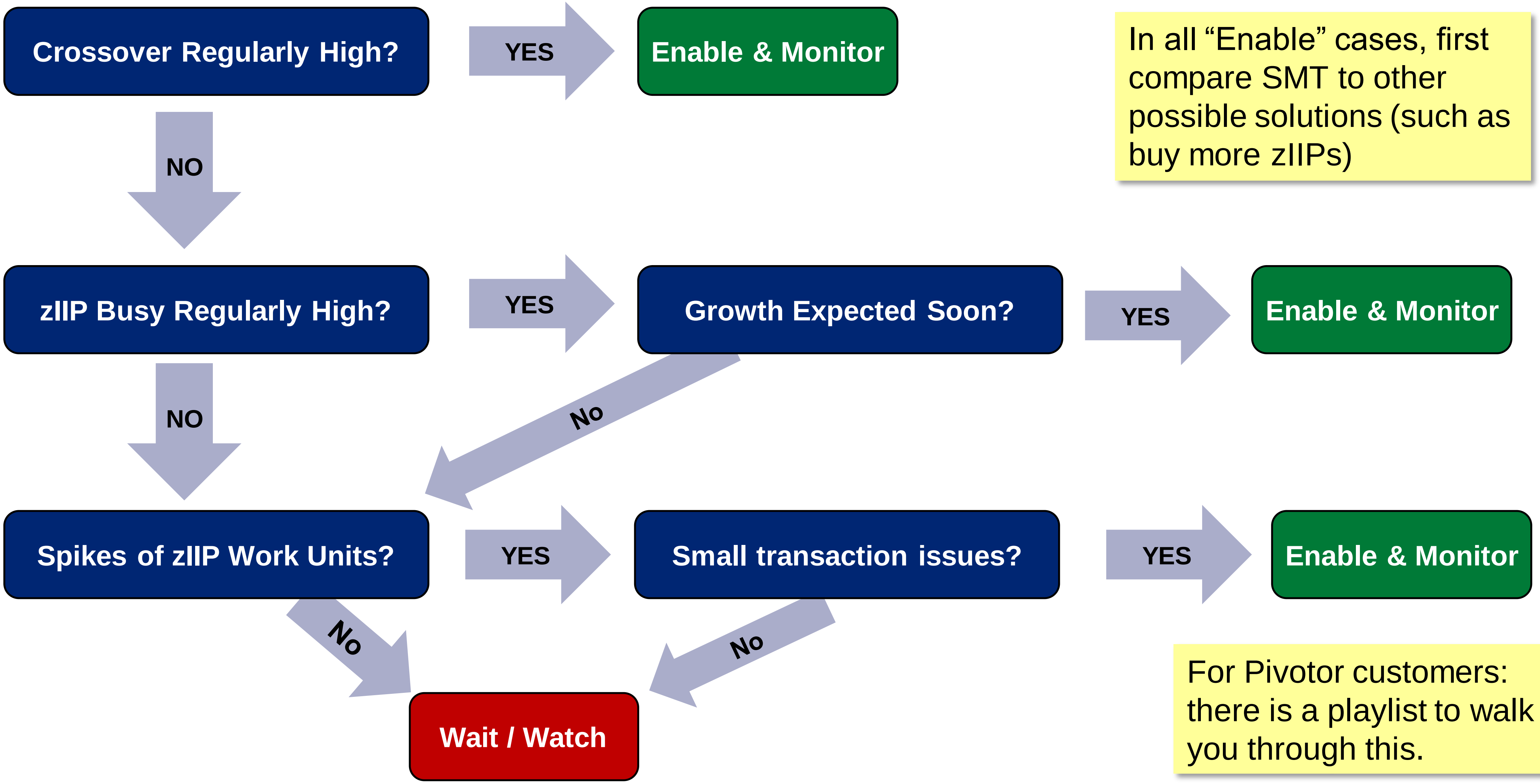
Do you care about spikes in work unit queues?

- Likely not, as long as the queue is handled quickly

- In some rare cases, where fractions of a second matter, maybe you do

- Reducing this impact means adding more engines so more work can be done in parallel
  - Buy more zIIPs
  - Allow crossover to GCPs sooner
    - Adjust ZIIPAWMT down (not generally recommended)
  - Enable SMT

SMT Enablement Flowchart



In all "Enable" cases, first compare SMT to other possible solutions (such as buy more zIIPs)

For Pivotor customers: there is a playlist to walk you through this.

SMT vs. More zIIPs

- Easy answer: buy more zIIPs if you can!

- SMT may be a good stop-gap if you can't because:
  - Financial constraints may delay purchasing more
  - After some period of time IBM will disallow microcode upgrades on z13 machine
    - z13 & z13s LIC WDFM: June 30, 2020
    - See: https://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD105503

- When buying a new machine: don't buy fewer zIIPs and hope to make it up with SMT
  - You don't know how effective SMT will be in your environment
  - You likely won't save that much money, relatively speaking
  - You eliminate the future possibility of enabling SMT to save the day

If you enable SMT…

- Verify application performance!

- Verify SMT measurements are indicating a positive change

- Reevaluate WLM Goals

- If effectiveness is too variable, consider turning SMT on/off based type of work running

- zIIP Utilization is now less than zIIP Busy
  - But consider planning by zIIP Busy since confidence around zIIP Utilization is limited

EPS Recommendations

- Understand the SMT measurements, benefits, and potential issues
- Set PROCVIEW=CORE to be prepared
- Experiment with ZIIP_MT_MODE=2 if you have a situation that may benefit from it
  - Results for dev/test and production very well may not be the same
  - Measure and understand the measurements
  - Measure effect on applications as well
- Remember that SMT will not be a good fit for all situations
  - Could turn on/off at different times of day/week/month
- With SMT enabled, be more conservative on how you think about zIIP utilization
  - zIIP utilization is an estimate with SMT
  - Compare zIIP busy vs. zIIP utilization
  - Understand that SMT makes zIIP capacity planning more difficult

Questions / Comments ?!?

Scott.Chapman@epstrategies.com
http://www.pivotor.com

SMT Measurement Summary

- Thread Density (TD) $= \frac{(TD1\ time + (TD2\ time * 2))}{(TD1\ time + TD2\ time)}$
- Capacity Factor (CF) $= \frac{Total\ Work\ Rate}{TD1\ Work\ Rate}$
- Maximum Capacity Factor (mCF) $= \frac{TD2\ Work\ Rate}{TD1\ Work\ Rate}$
- Productivity $= \frac{CF}{mCF}$
- Core Utilization % = Core Busy % * Productivity %
- zIIP CPU time is reported as Multi-Threading 1 Equivalent Time (MT1ET)
- MT1ET = raw CPU Time * Chargeback Factor (CBF) (maybe)
- CBF $= \frac{CF}{TD}$ (maybe)
- zIIP Appl% $= \frac{zIIP\ MT1ET}{Interval\ Duration * mCF}$

SMT Derived Measurement Summary

- Core busy time (CBtm) = $Core\ Busy\,\%\ *\ Interval\ Time$
- TD1tm = $CBtm\ *\ 2 - CBtm\ *\ TD$
- TD2tm = $CBtm - TD1tm$
- CBtm = MT1ET / CF (!?)