z/OS SMT: Deciding Whether to Enable

Scott Chapman
Enterprise Performance Strategies, Inc
scott.chapman@epstrategies.com

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:

Enterprise Performance Strategies, Inc. presentation materials contain trademarks and registered trademarks of several companies.

The following are trademarks of Enterprise Performance Strategies, Inc.: **Health Check®, Reductions®, Pivotor®**

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries: IBM®, z/OS®, zSeries® WebSphere®, CICS®, DB2®, S390®, WebSphere Application Server®, and many others.

Other trademarks and registered trademarks may exist in this presentation.

Agenda

- Why and What is Simultaneous Multi-Threading
- Terminology (new and re-named)
- Measurements
  - Names
  - Meanings
  - Sources
- Thoughts
  - What we've seen
  - Possible applicability

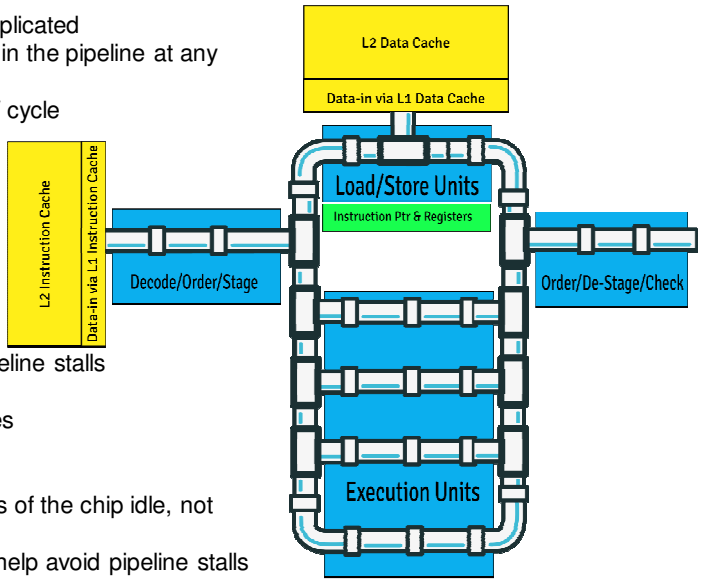
Red words are key points to pay attention to

We can't work faster…

- We can't assume clock speeds are going to continue increasing
  - 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

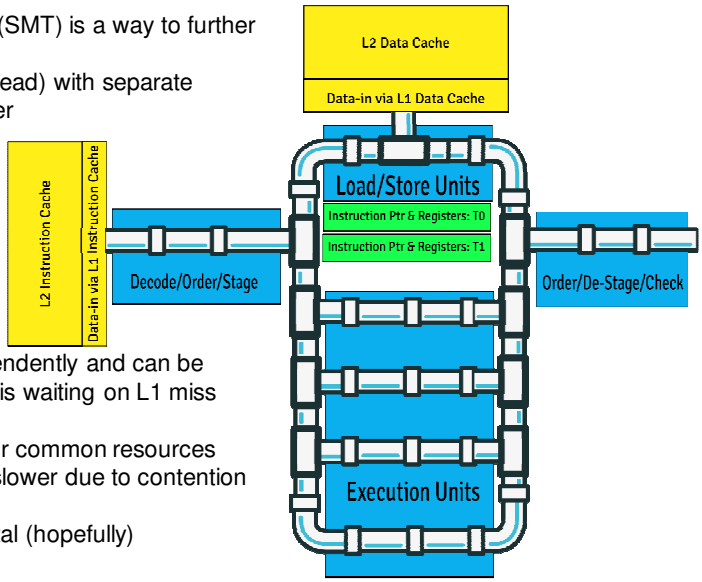
- Modern processors are complicated
- Multiple instructions in-flight in the pipeline at any given time
- Aim: 1+ instruction finished / cycle



- 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



- 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)

L2 was unified I- and D- cache prior to zEC12. Possibly should have multiple output pipes.

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) is 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 Considerations

- You don't have to pay extra to enable SMT
  - But there are operational considerations
- 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
- Some commands may change
  - 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…

SMT Enablement Follow-up

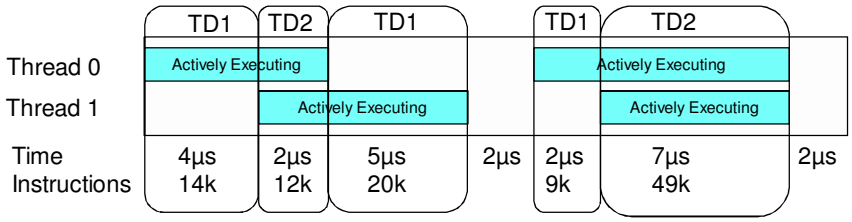
- Achieved velocity will likely change for workloads using zIIP
  - Effectively, SMT2 = more/slower zIIPs vs. fewer/faster
  - Re-evaluate your WLM goals after implementation
- Evaluate application responsiveness
  - For same reason as above: performance will likely change
- Evaluate SMT measurements
  - Understanding these measurements is a whole separate presentation
    - (It may make your head hurt)
  - Application performance numbers are more important than abstract measurements

SMT Measurement Simplified Definitions

- Thread Density (TD)
  - Average number of threads executing on a core, when at least 1 thread is running
- Capacity Factor (CF)
  - How much work is being done with SMT enabled vs. if SMT wasn't enabled
  - For example: 1.15 = 15% more throughput due to SMT
- Maximum Capacity Factor (mCF)
  - Maximum predicted throughput benefit from SMT if TD was 2
- Productivity
  - How much work is being done vs how much could be done
- Core Utilization %
  - New utilization measure factoring in SMT (no longer same as busy)
- MT1ET
  - CPU (zIIP) time that would have been consumed without SMT

You have to test in production

- Single thread is running on a core = no contention, full performance
- Two threads running on a core = contention, performance degradation



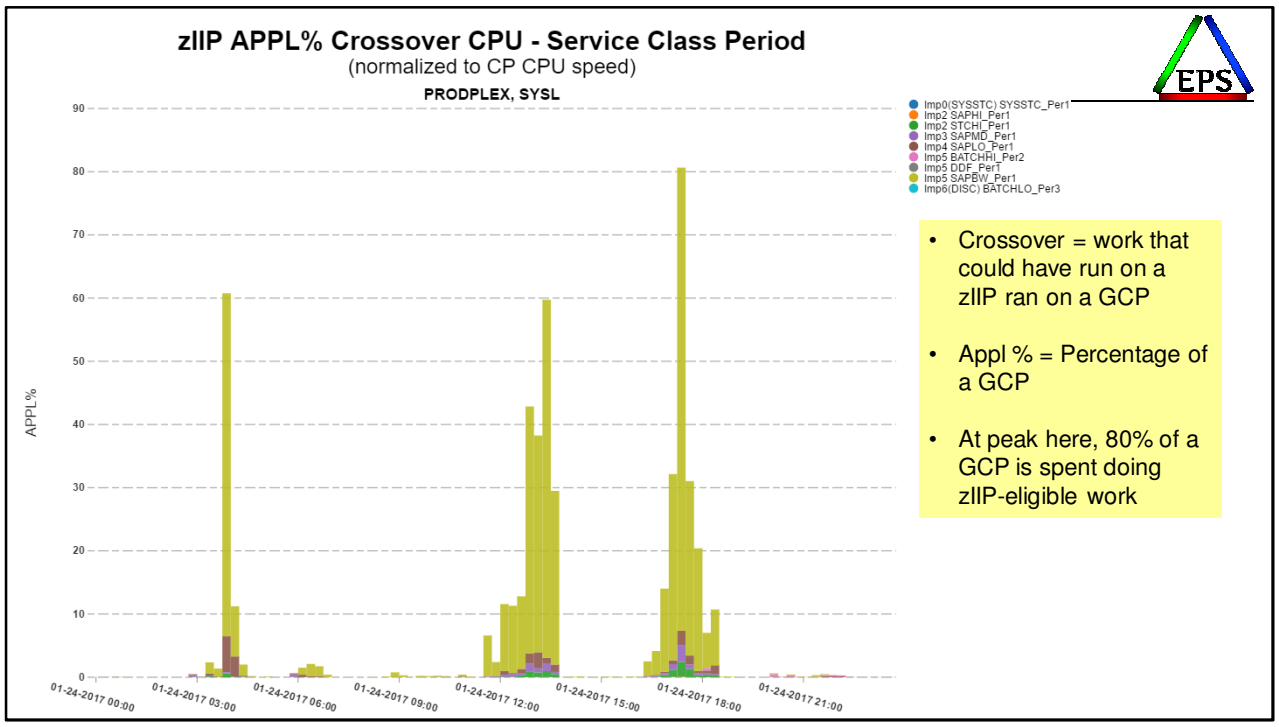
- SMT performance impact is dependent on how often two threads are concurrently executing on the same core and how much they're contending with each other
  - Very dependent on specific arrival patterns of work, potentially at microsecond scale
  - z/OS densely packs cores
- Results from test environment don't necessarily represent production results

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 how can we tell if one of the last three scenarios might apply

First metric to evaluate: zIIP Crossover



- 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

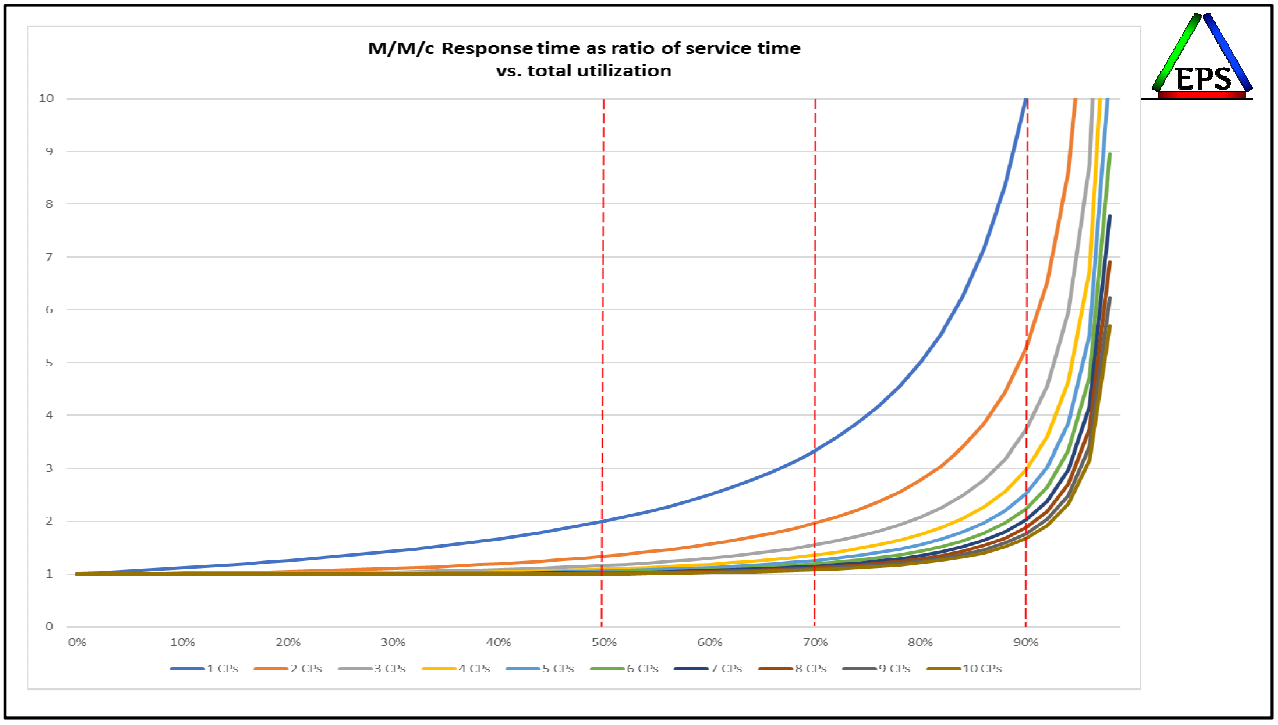
Avoid significant crossover

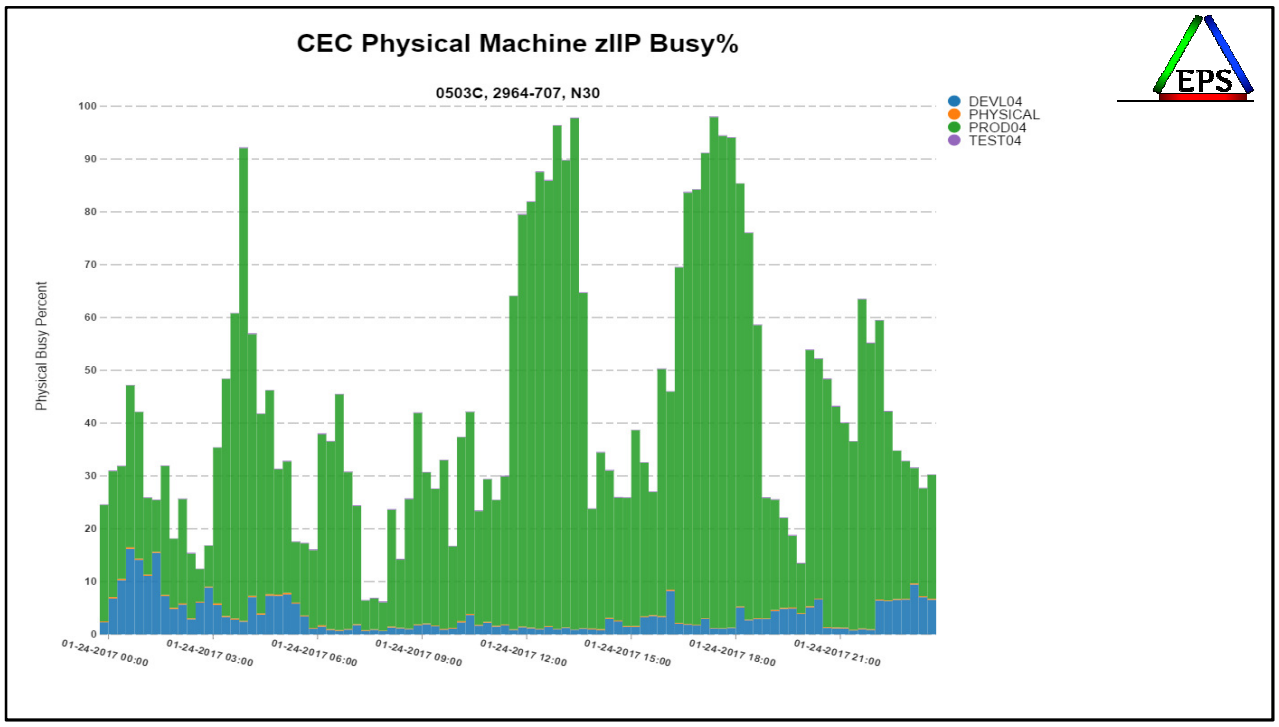
Potential Crossover solutions

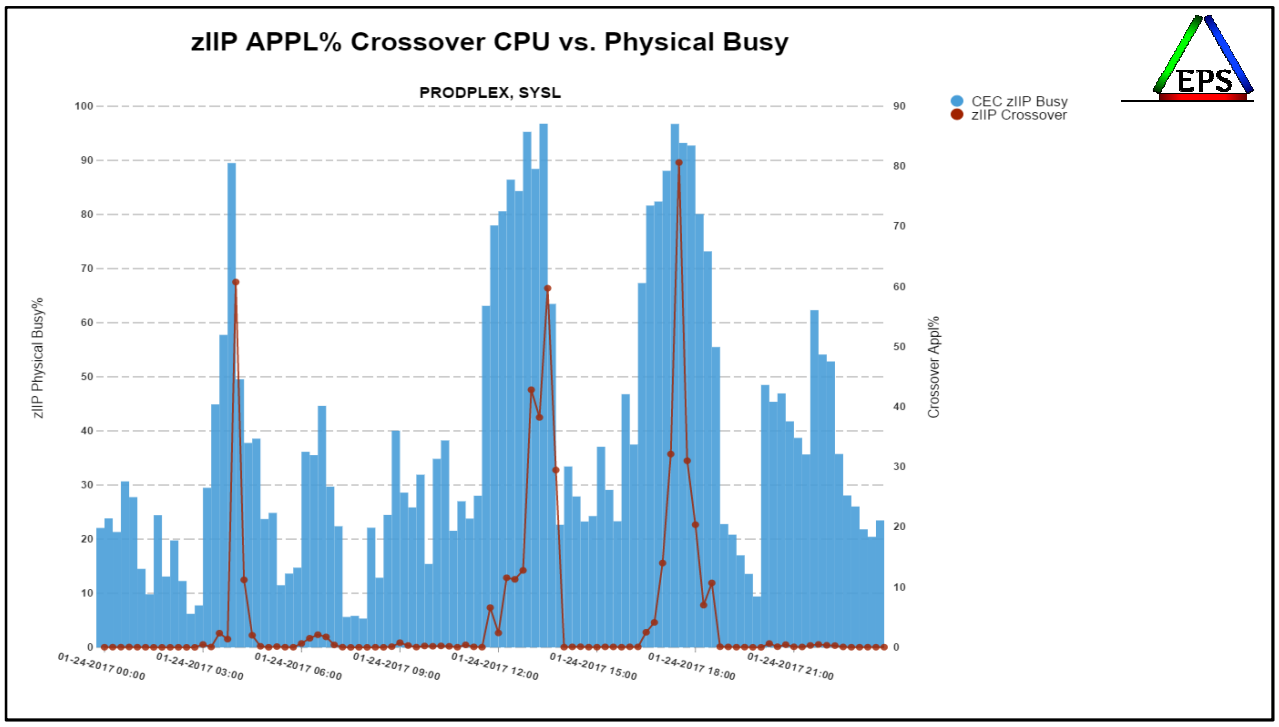
- Buy more zIIP(s)
  - Always best from a performance perspective
  - They may be cheap relative to the software costs they offset
- IIPHONORPRIORITY=NO
  - Set in IEAOPTxx
  - Complications with DB2 v11+ make this more difficult
    - Possibly: use =YES while starting DB2, switch to =NO after
    - Wait for DB2 to "fix" the issue
    - Increasing ZIIPAWMT may be useful in some limited cases
- Enable SMT
  - More capacity, although likely impact to individual thread performance
- Run less zIIP work
  - We're generally heading towards more zIIP work, not less

What about zIIP Busy?

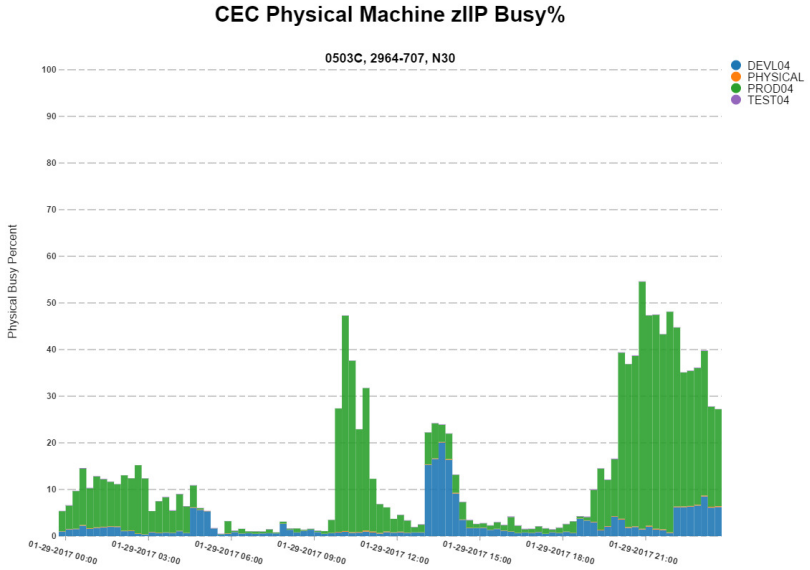
- You may hear something like “don’t run your zIIPs over 50% busy”
- Should you add capacity (more, or SMT) when they’re over 50%?
- From a performance perspective, less busy is always good
- From a financial perspective we want to make sure our resources are well-utilized
  - While one can generally 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

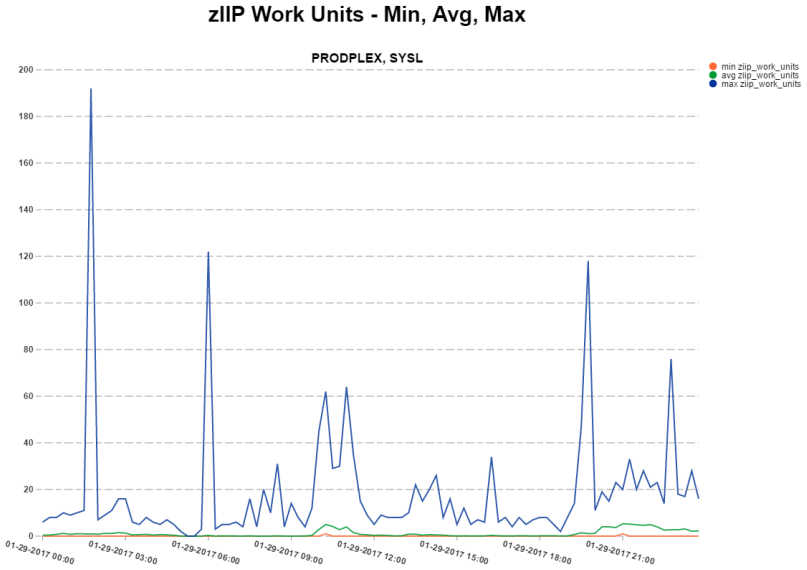

CEC Physical Machine zIIP Busy%
0503C, 2964-707, N30
Physical Busy Percent
100
90
80
70
60
50
40
30
20
10
0
01-24-2017 00:00
01-24-2017 03:00
01-24-2017 06:00
01-24-2017 09:00
01-24-2017 12:00
01-24-2017 15:00
01-24-2017 18:00
01-24-2017 21:00
DEVL04
PHYSICAL
PROD04
TEST04
EPS


zIIP APPL% Crossover CPU vs. Physical Busy
PRODPLEX, SYSL
CEC zIIP Busy
zIIP Crossover
zIIP Physical Busy%
Crossover Appl%
100
90
80
70
60
50
40
30
20
10
0
01-24-2017 00:00
01-24-2017 03:00
01-24-2017 06:00
01-24-2017 09:00
01-24-2017 12:00
01-24-2017 15:00
01-24-2017 18:00
01-24-2017 21:00

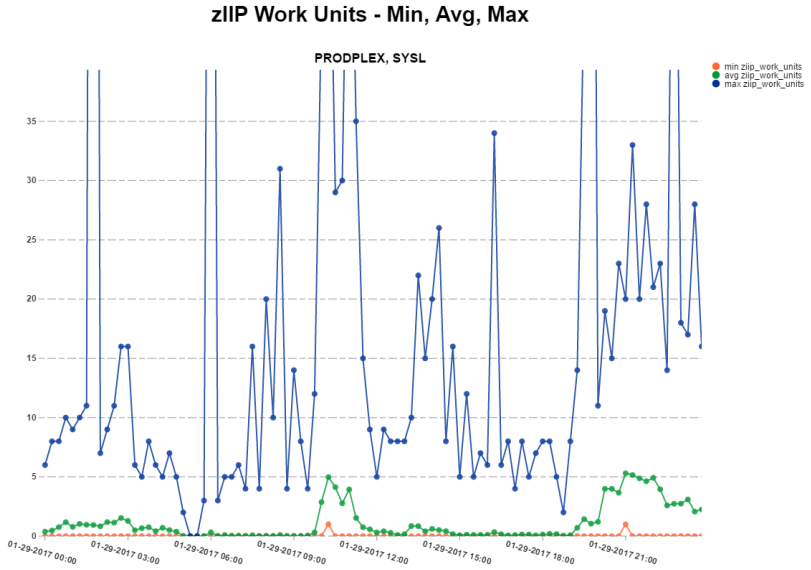
zIIP Busy can indicate problems are unlikely



Unlikely is not impossible



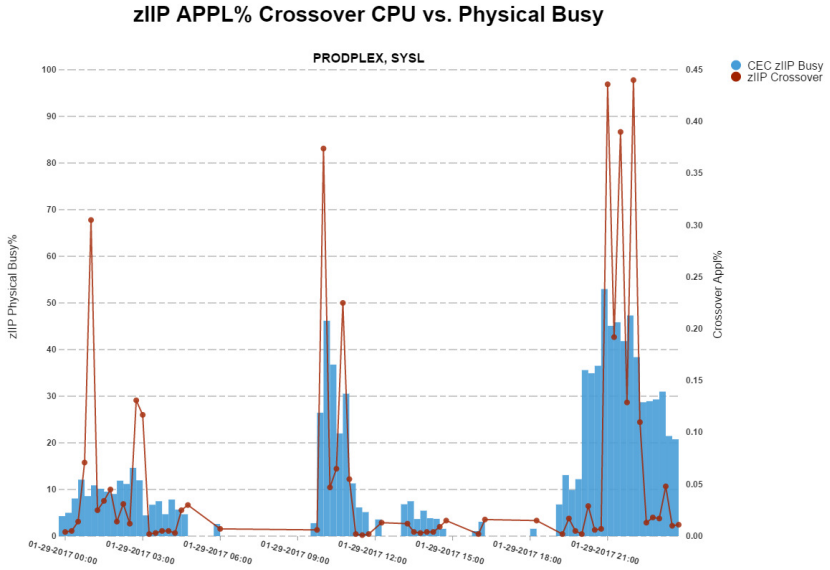
Zoomed in to see average better



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

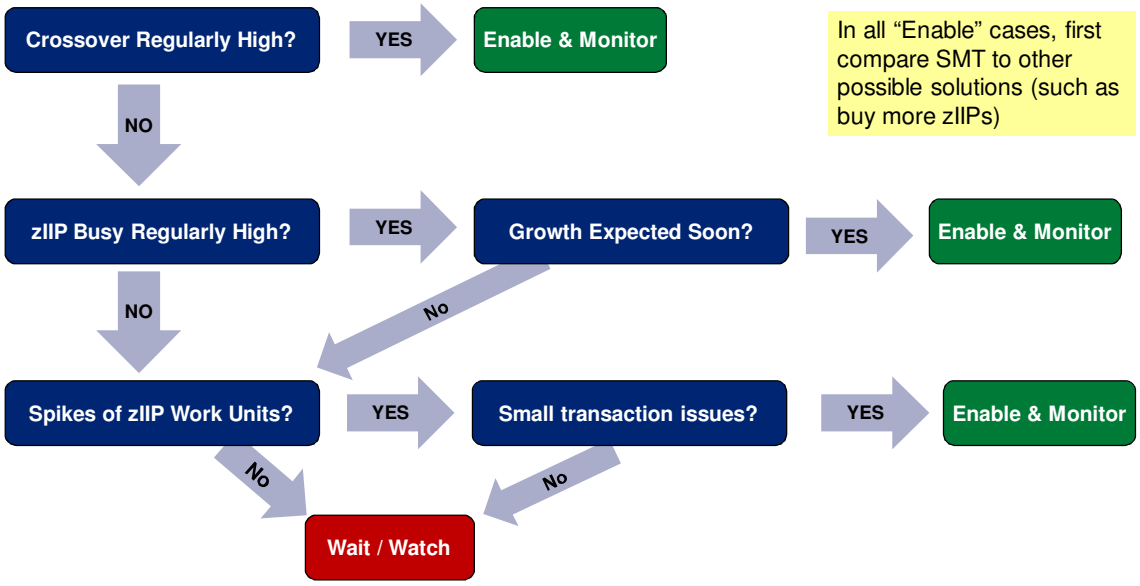
Crossover < 1% of GCP engine for that day



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)

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

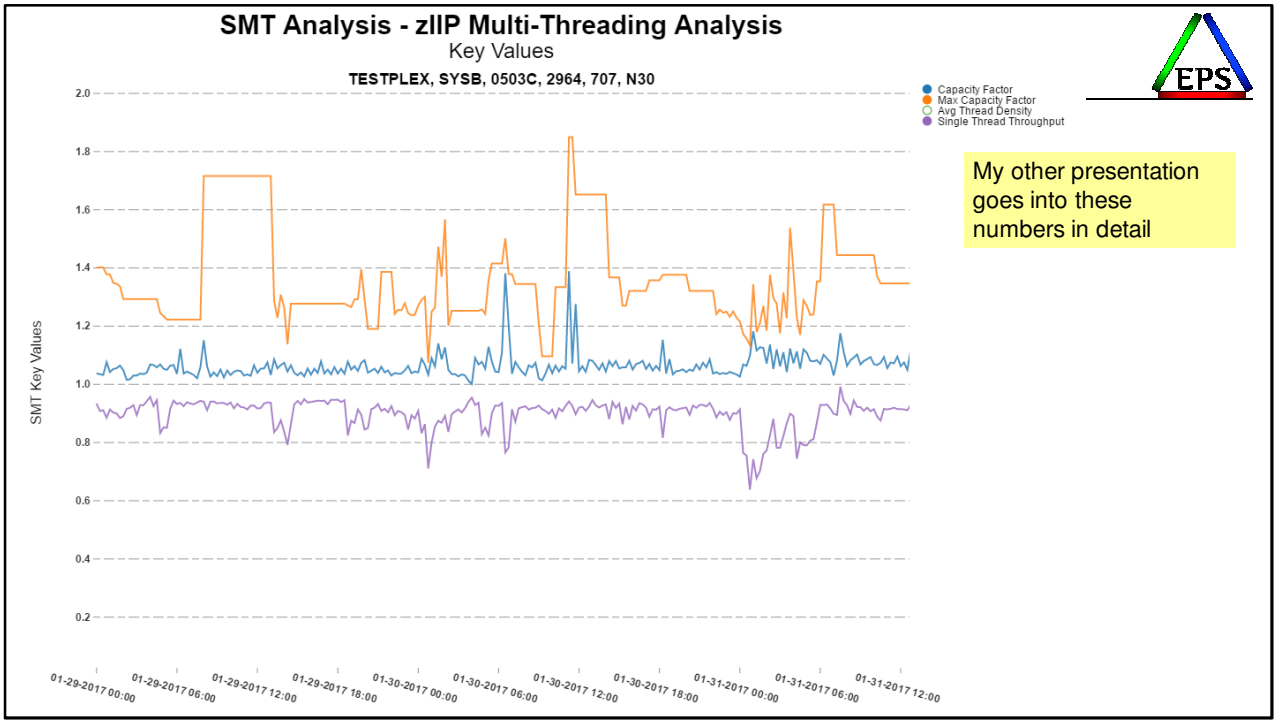
- When buying a new machine: don't buy fewer zIIPs and hope to make it up with SMT
  - You don't know how effective SMT will be in your environment
  - You likely won't save that much money, relatively speaking
  - You eliminate the future possibility of enabling SMT to save the day

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)

If you enable SMT…

- Verify application performance!
- Verify SMT measurements are indicating a positive change
  - See my other presentation for the details
- 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



My other presentation goes into these numbers in detail

Summary

- zIIP Crossover is a good indicator for needing more zIIP capacity
- If you need more zIIP capacity consider other options before SMT
- If you do enable SMT, be sure to follow up after enablement: not set and forget
- When buying a new machine, buy the zIIP capacity you need, don't count on SMT to provide some of that needed capacity
  - But maybe be less concerned about long-term needs, as SMT can be your "ace in the hole"

Questions / Comments ?!?