Understanding Simultaneous Multithreading on z Systems

(post-announcement)

Bob Rogers

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

Abstract

Simultaneous Multithreading (SMT) is a technique for increasing the efficiency of CPUs to deliver more throughput per processor core. It has been exploited on IBM POWER Systems and Intel processors for several years. Now, IBM System z has delivered the z13 mainframe processor that supports for SMT for specialty engines. So, it's time for mainframers to learn about SMT - what it is, how it's supported on the z13 and what it means to mainframe installations. In this webcast, Bob Rogers will provide the historical background and provide some insight into this technology as it is initially supported by z/OS. Bob gave a webcast back in April which speculated on the SMT support. This webcast will reflect what has been announced.

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

Understanding SMT

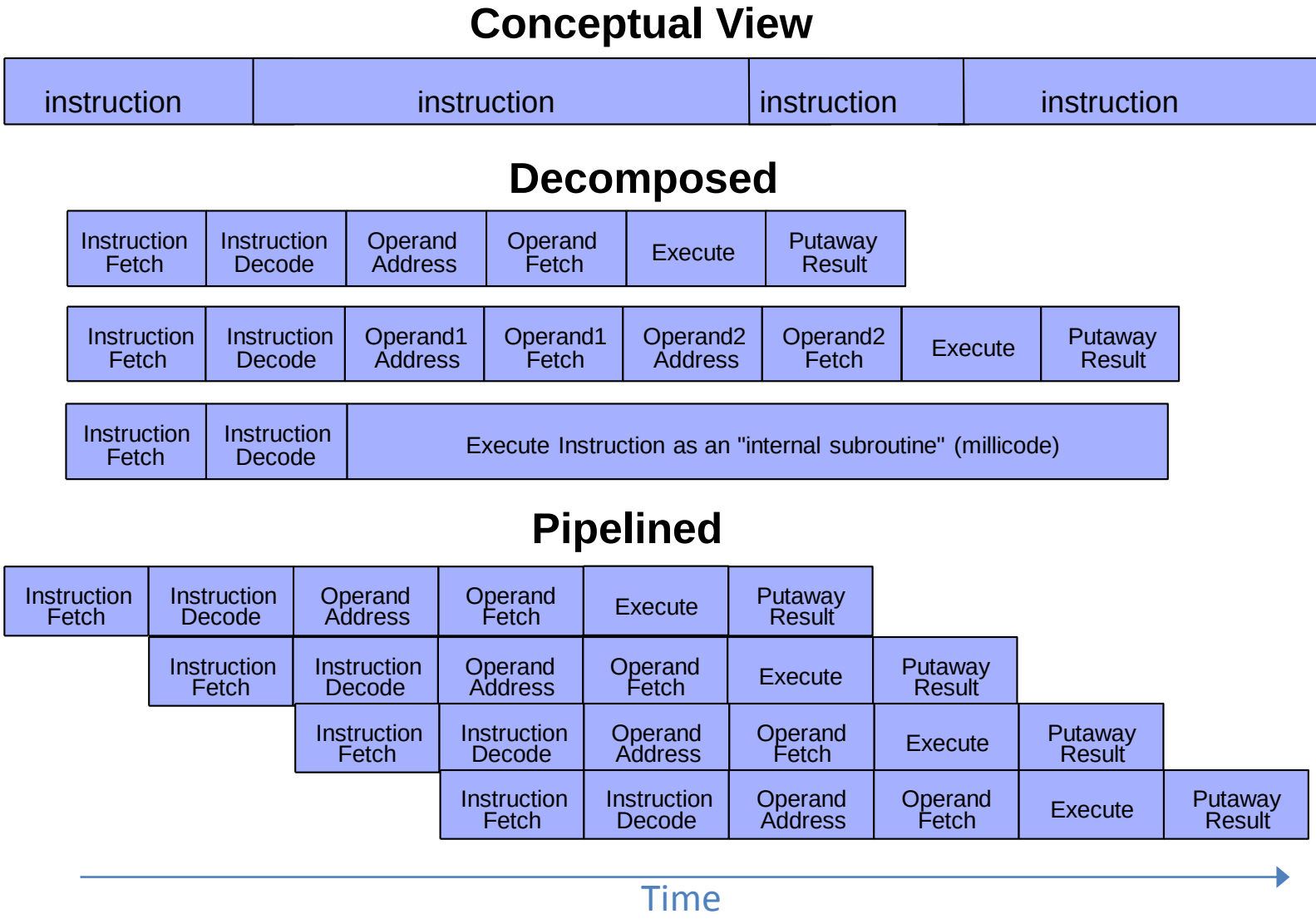
Simultaneous Multithreading is a refinement in processor design that is the end of a sequence of ideas that have increased the complexity of the processor to deliver greater throughput.

In order to understand how SMT works and what it means for performance management and capacity planning, it is best to start with how modern computers process instructions.

We can start with the evolution of the instruction pipeline.

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

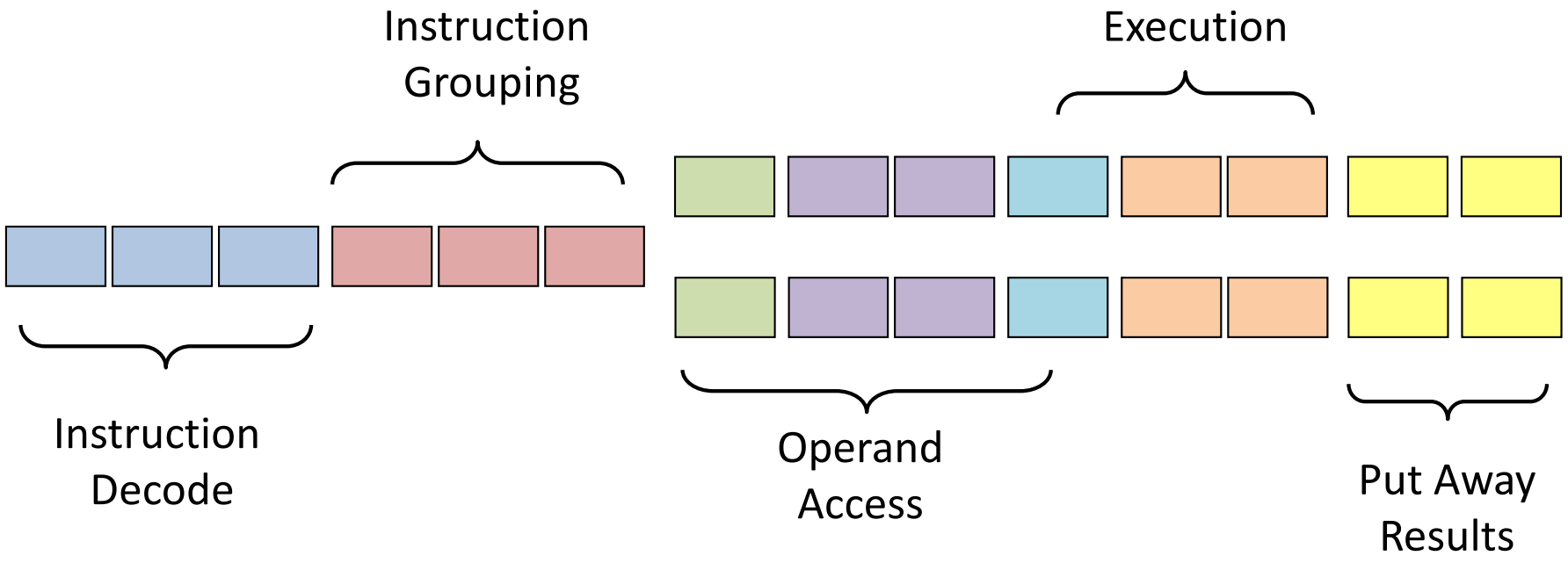
Conceptual View of Execution vs Reality



© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

Schematic of Superscalar Pipes

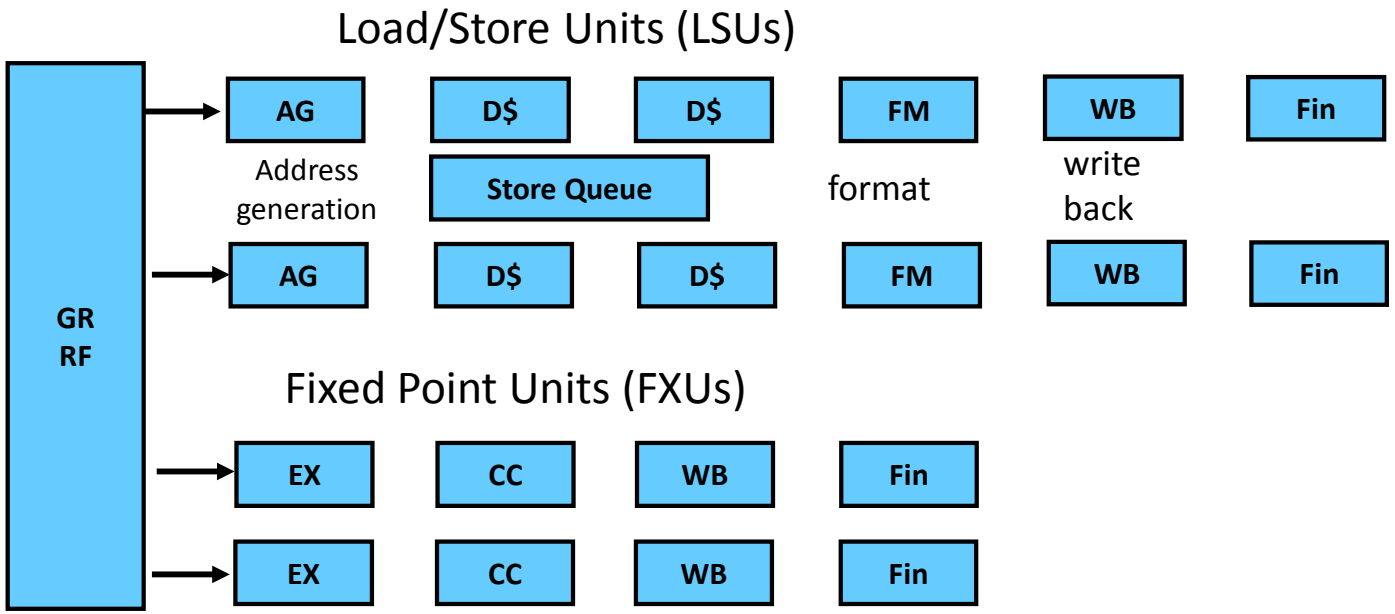
A Superscalar processor can process multiple instructions simultaneously because it has multiple units for each stage of the pipeline.



Each box from left to right represents a pipeline stage which takes one cycle. After instructions have been decoded and put into superscalar groups, they are issued down the two pipes one or two instructions at a time. In this example, the apparent order of execution is still maintained.

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

RISC-like Superscalar Pipes



Depiction of part of the z196 pipeline showing 2 load/store units and 2 fixed point units. The stages for instruction fetch, decode and grouping and putaway are not shown. Also not shown are the floating point units for binary and decimal arithmetic. Under ideal conditions, the z196 can execute 5 instructions simultaneously.

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

Out-of-Order Execution

- A processor that can execute instructions Out-of-Order (OOO) uses detailed bookkeeping and some tricks to appear to execute the program as it was written.
- To do the bookkeeping, the processor maintains what is called a global completion table to track the status of all in-flight instructions.
- The results of an instruction cannot be stored until all older instructions have previously completed.
- For example, if an interrupt occurs, all instructions that have not already stored results must be “forgotten”, and re-executed later. The interruption PSW reflects the newest instruction that stored results (i.e. the last completed instruction such that all preceding instructions had also completed).

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

Out-of-Order Execution Example

(On a 2-way superscalar processor)

Original Instruction Sequence

7 instruction groups and 10 cycles AGI delay

```
 seq     instruction text              | seq   instruction text
 01  LLGT  @04,XFORNP31                |
 02  L     @04,FW(,@04)                | 03  ST    @04,XFORS
 04  LG    @05,TOPPTR                  |
 05  LG    @09,RTTOP(,@05)             |
 06  ST    @04,RSISIZE(,@09)           | 07  SLR   @02,@02
 08  ST    @02,RSIPREV(,@09)           | 09  LG    @02,RDIPTR64
 10  LH    @08,RDITYPE(,@02)           |
```

Reordered Instruction Sequence (done by a compiler)

5 instruction groups and 6 cycles AGI delay

```
 seq     instruction text              | seq   instruction text
 01  LLGT  @04,XFORNP31                | 04  LG    @05,TOPPTR
 05  LG    @09,RTTOP(,@05)             | 07  SLR   @02,@02
 02  L     @04,FW(,@04)                | 06  ST    @04,RSISIZE(,@09)
 08  ST    @02,RSIPREV(,@09)           | 09  LG    @02,RDIPTR64
 03  ST    @04,XFORS                   | 10  LH    @08,RDITYPE(,@02)
```


© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

Register Renaming for OOO

- An OOO z/Architecture processor does not have just 16 GPRs.
  - For example, the zEC12 has 80 physical GPRs.
- The reason there are so many physical GPRs is so that the processor can effectively execute instructions out-of-order.
  - The extra physical GPRs allow the processor to reload an architected GPR while its previous value is still needed.
  - Using extra physical registers eliminates GPR interlocks
- Consider this instruction sequence:

```
L  4,0(,1)  Load address of first parameter
L  2,0(,4)  Load first parameter into register 2
L  4,4(,1)  Load address of second parameter
L  3,0(,4)  Load second parameter into register 3
```

- An OOO processor with register renaming can perform both loads of GPR 4 at the same time, and both loads using GPR 4 at the same time.

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

SMT Terminology

Processor core

- Hardware capable of executing instructions. Typically, multiple processor cores are fabricated on a processor chip.
- An SMT-capable core can execute multiple instruction streams simultaneously.

Software thread

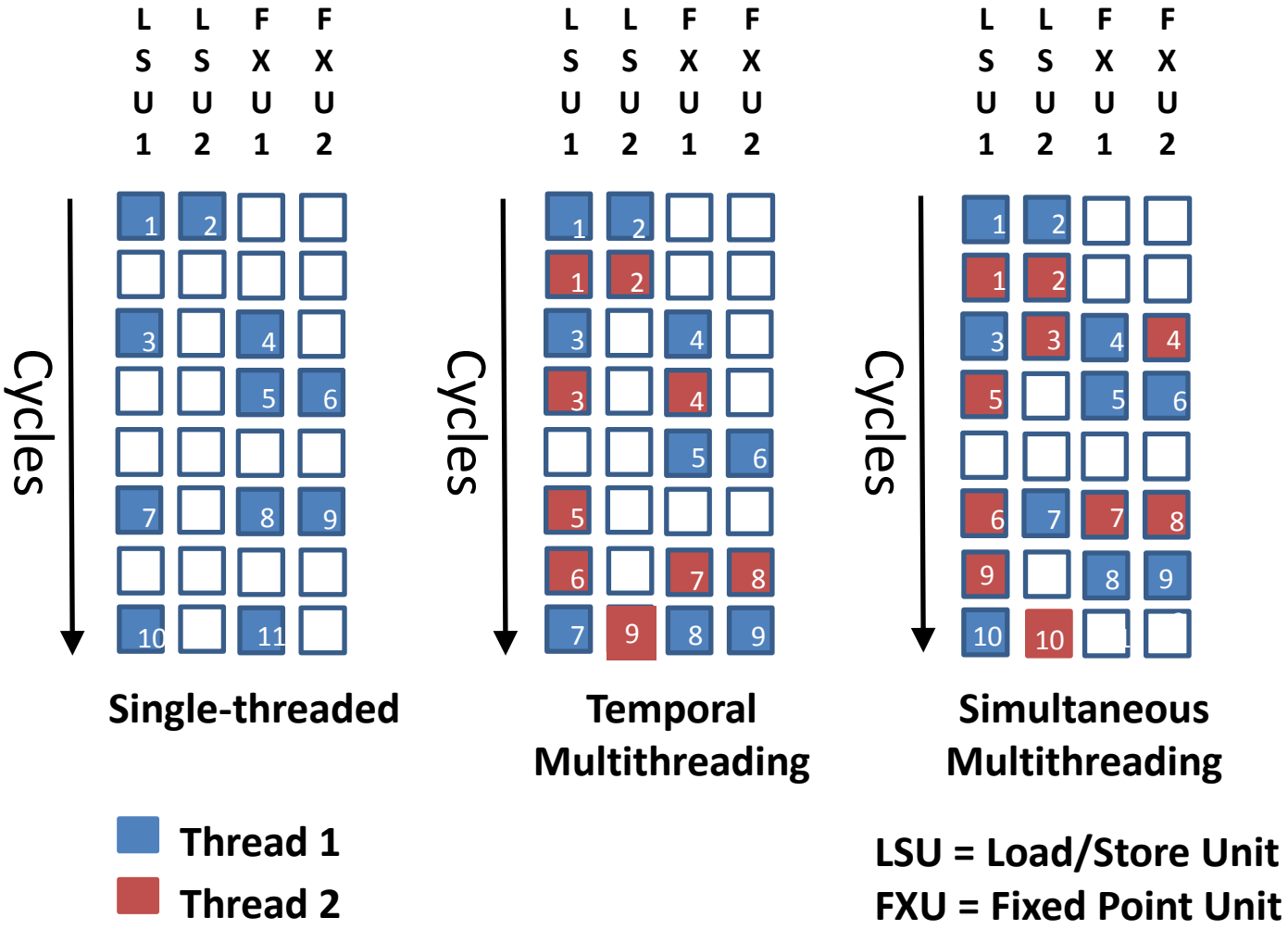
- A linear sequence of instructions executed for one program. On z/OS these is typically the instructions of a task or SRB.

Hardware thread

- The set of processor resources used to execute a single software thread. When a processor core entails more than one hardware thread, it is call Multi-threaded.

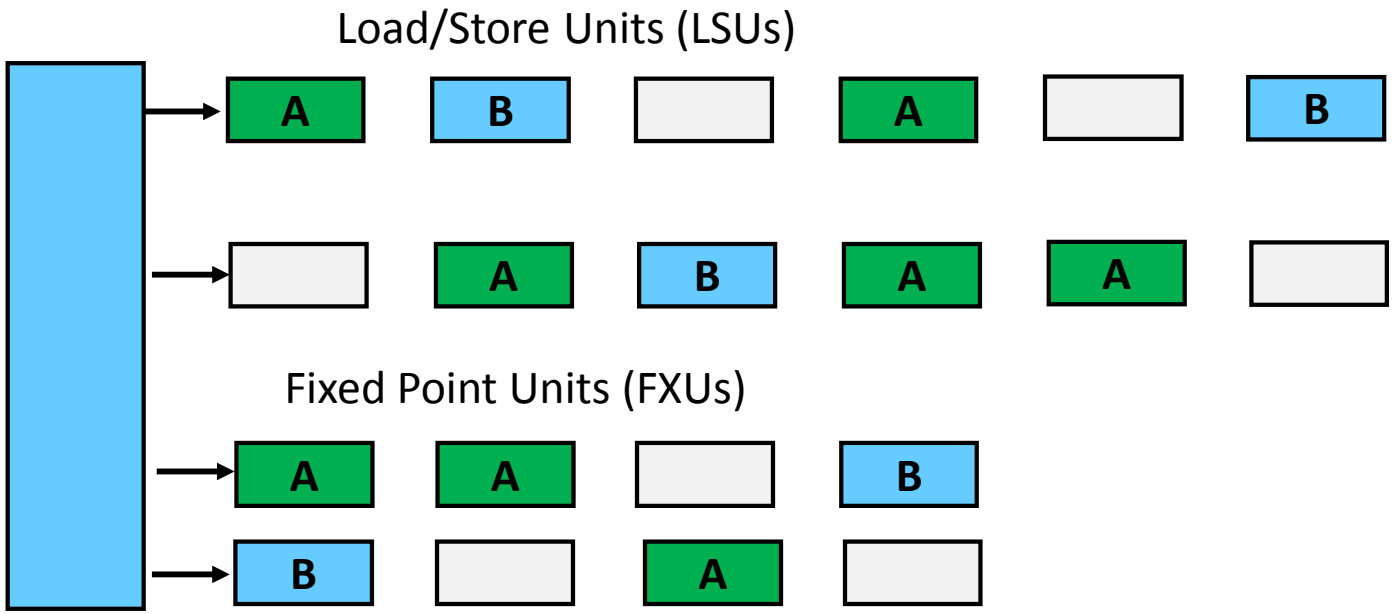
© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

Single-threaded vs Multithreading



© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

Pipeline executing 2 threads



Each stage of the pipeline can process an instruction from thread A or an instruction from thread B or no instruction. Pipeline bubbles for one thread can be filled with instructions from the other thread.

However, there can also be interference between thread for the use of shared resources of the core.

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

SMT Effectiveness

- Based on some early papers, a core design with two simultaneous threads (SMT2) can deliver an additional 40% throughput.
  - This is a gross approximation and actual results can vary considerably.
  - And, this is only when both threads are busy.
- Unfortunately, the additional throughput from SMT does not scale with the number of threads. This is because all the threads on a core share some limited resources.
- In a case where two threads (nominally) add 40% throughput, a core with four threads (SMT4) can deliver (nominally) only 60% more than single-threaded.
  - That's not even 15% more than SMT2.

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

Evaluating SMT

- On the positive side, SMT delivers more throughput per core.
  - More capacity for a given footprint size
  - Less power, cooling, etc. required per unit of capacity
- But, there are negatives as well.
- The first is that an individual thread in multithread mode is slower than a single thread would be. The speed drops quite rapidly with the number of threads.
  - If an SMT2 core provides 140% of the capacity of a single thread, then the two threads will (on average) each run at 70% of the single-thread speed when both threads are active.
  - For SMT4, if all four threads are active, they would run at only 40% of the single-thread speed.
- The second negative is an increase in variability.
  - Increased sharing of low-level resources by threads makes the amount of work that a thread can do dependent on what else the core is doing.

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

What Causes the Slowdown?

- A major cause of less than linear speed-up is the sharing of processor cache.
  - On recent System z processors, there are two levels of cache that are private to the core . They are called L1 and L2.
  - If a core has more than one thread, these caches will be shared across all the threads.
  - Each thread is forces to get by with a smaller footprint in these caches and so takes more L1 and L2 misses than if the caches were not shared.
- Other resources must also be shared:
  - The pipes,
  - The translation lookaside buffer (TLB), and,
  - Physical General Purpose Registers
  - Store Buffers and other resources on the core.

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

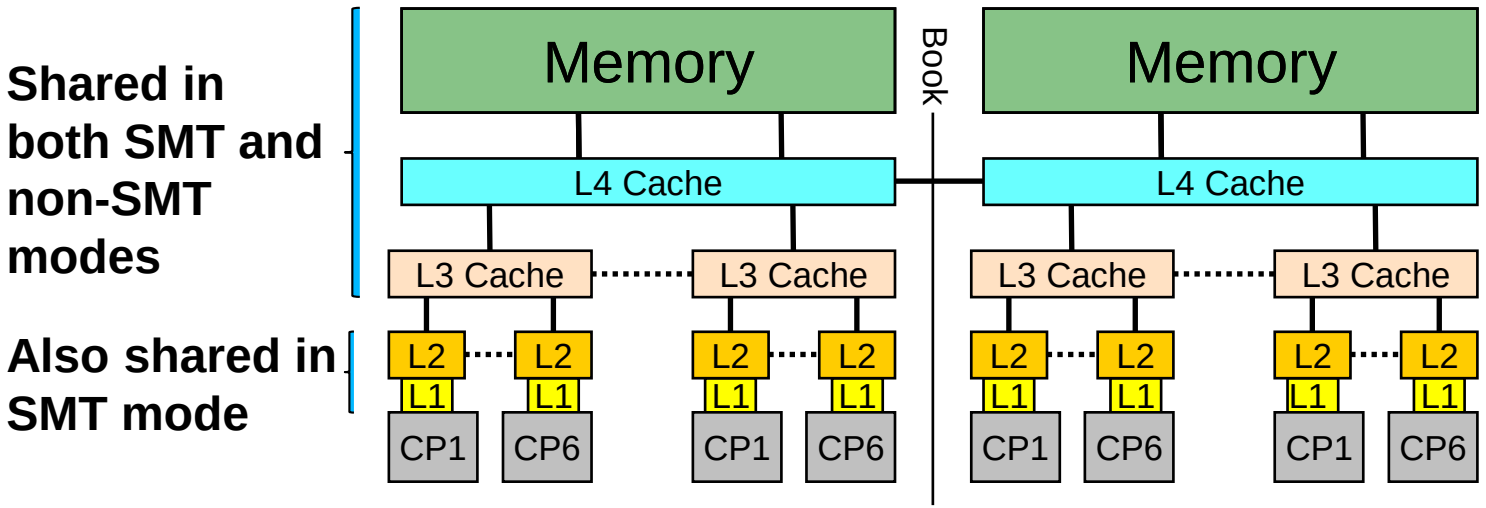
SMT creates a new dimension of variability

There is always CPU capacity variability from one processor design to the next.

And there is variability because threads share higher level caches.

Sharing of resources among the threads make the “SMT boost” further variable because with SMT, they also share the private caches, the execution pipelines and other core resources.

SMT-2 threads can run as slow as 50% or as fast as 100% single-thread speed, depending on the nature of the two instruction streams.



Cache Structure

Why the Variability?

- Citing numbers like 140% throughput for SMT2 or 160% throughput for SMT4 are gross simplifications.
- Actual throughput for SMT2 can range from less than 100% (yes, SMT2 can be worse than single-threaded) to close to 200%, depending upon the usage of the shared resources.
- For example, if programs running on the same core stress the same resources, they will run slower than average.
- Alternately, if the programs resource usage is complimentary, they can run close to the ideal maximum speed.
- Running the same application multiple times shows less repeatable CPU usage because it may run in differing environments.

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

Some Remediation for Charge-back

- For internal charge-back, IBM has paved the way for more stable charge-back with an enhancement made in z/OS 2.1.
- Even on systems exhibiting wild variations in CPU time for jobs, the number of instructions executed by a job would be expected to be fairly stable.
- To exploit this, z/OS 2.1 can include instruction count data in the SMF type 30 record, provided that the installation has turned on hardware instrumentation (HIS) and instructed SMF to include the data.
- A charge-back scheme based on instruction counts rather than CPU time might be expected to exhibit more repeatability from one run to the next than one based on CPU time.

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

Preparing for CPU Slow-down

- z/OS provides fields in the SMF type 30 record that identify the name of the program running on the task that has consumed the most CPU in the time covered by the record and give the percentage of a CPU that that task consumed.
- IBM provides a tool to identify potential problems in a batch flow, called Batch Network Analyzer for z (zBNA).
  - It can be used to explore what-if scenarios by simulating how a batch flow would run on a different speed processor.
  - Alternate CPU zIIP SMT support provided 3/31/15 in zBNAV1.6.3
- Transaction processing workloads, in general, should not have much problem with the SMT-induced slowdown.
  - The CPU-using portion of a transaction is typically pretty small
  - Furthermore, much of the CPU delay portion of a transaction will be eliminated because the transaction manager will have more threads on which to run the transactions (few large vs many small).

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

What's Supported with z13

- The z13 supports 2-way SMT (SMT2).
  - But, only for specialty engine cores (zIIPs and IFLs).
  - CP cores always have only one active thread.
- LPAR dispatches cores.
  - If the guest z/OS is running with PROCVIEW=CORE then LPAR dispatches both threads of a logical core onto the hardware threads of a physical core.
  - If there is only one thread active then the other hardware thread is dispatched in a non-running state.

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

How it is Enabled

LOADxx PARMLIB controls the PROCESSOR VIEW of CORE vs CPU

- **The specification is for the life of the IPL.**
- **PROCVIEW CORE on z13 enables MT support**
  - Allocates 2 threads per core (with 2 CPU addresses)
  - Forces HiperDispatch to be active
  - Requires ConFig Core, Display Matrix=Core commands
  - PROCVIEW CORE,CPU_OK allows use of “CPU” as an acceptable alias for “CORE” in commands
- **PROCVIEW CPU can be specified on all servers but results in the existing function and management controls**
  - Establishes existing (Pre-MT, MT-1) environment, uses Pre-MT command controls
    - Retains MT-1 configuration, allocates 1 CPU per core
    - Requires ConFig CPU, Display Matrix=CPU commands
- **To Fallback from PROCVIEW CORE to PROCVIEW CPU requires an IPL.**

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

How it is Activated

New IEAOPTxx parameter to control zIIP MT Mode

- Without an IPL you can change the zIIP processor class MT Mode (the number of active threads per online zIIP) using IEAOPTxx.
  - MT_ZIIP_MODE=2 for 2 active threads, MT_ZIIP_MODE=1 for 1. MT_ZIIP_MODE=1 is the default.
  - When PROCVIEW CPU is specified the processor class MT Mode must remain 1.
  - PROCVIEW CPU, and PROCVIEW CORE MT Mode=1 receive the same performance
  - CP cores always use MT Mode=1

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.

Compatibility

SMT support is available with z/OS V2.1 after applying the following APARs:

- OA43366 (BCP), OA43622 (WLM), OA44439 (XCF)

The new LOADxx and IEAOPTxx controls ONLY available on z/OS V2.

- LOADxx and IEAOPTxx members with SMT controls cannot be shared with (that is, used by) z/OS V1.12 or z/OS V1.13 systems.

© Copyright NewEra Software and Robert Rogers, 2015, All rights reserved.