TEXAS ADVANCED COMPUTING CENTER

WWW.TACC.UTEXAS.EDU

The University of Texas at Austin

Bandwidth Limits in the Intel Xeon Max (Sapphire Rapids with HBM) Processors

ISC 2023 IXPUG Workshop

John D. McCalpin
Texas Advanced Computing Center, The University of Texas at Austin

Intel “Sapphire Rapids”: DDR5 vs HBM

Intel’s new “Sapphire Rapids” processors are available in several versions:

- 4th-generation Xeon Scalable Processors (DDR5 only)
- Xeon Max Processors (HBM + optional DDR5)

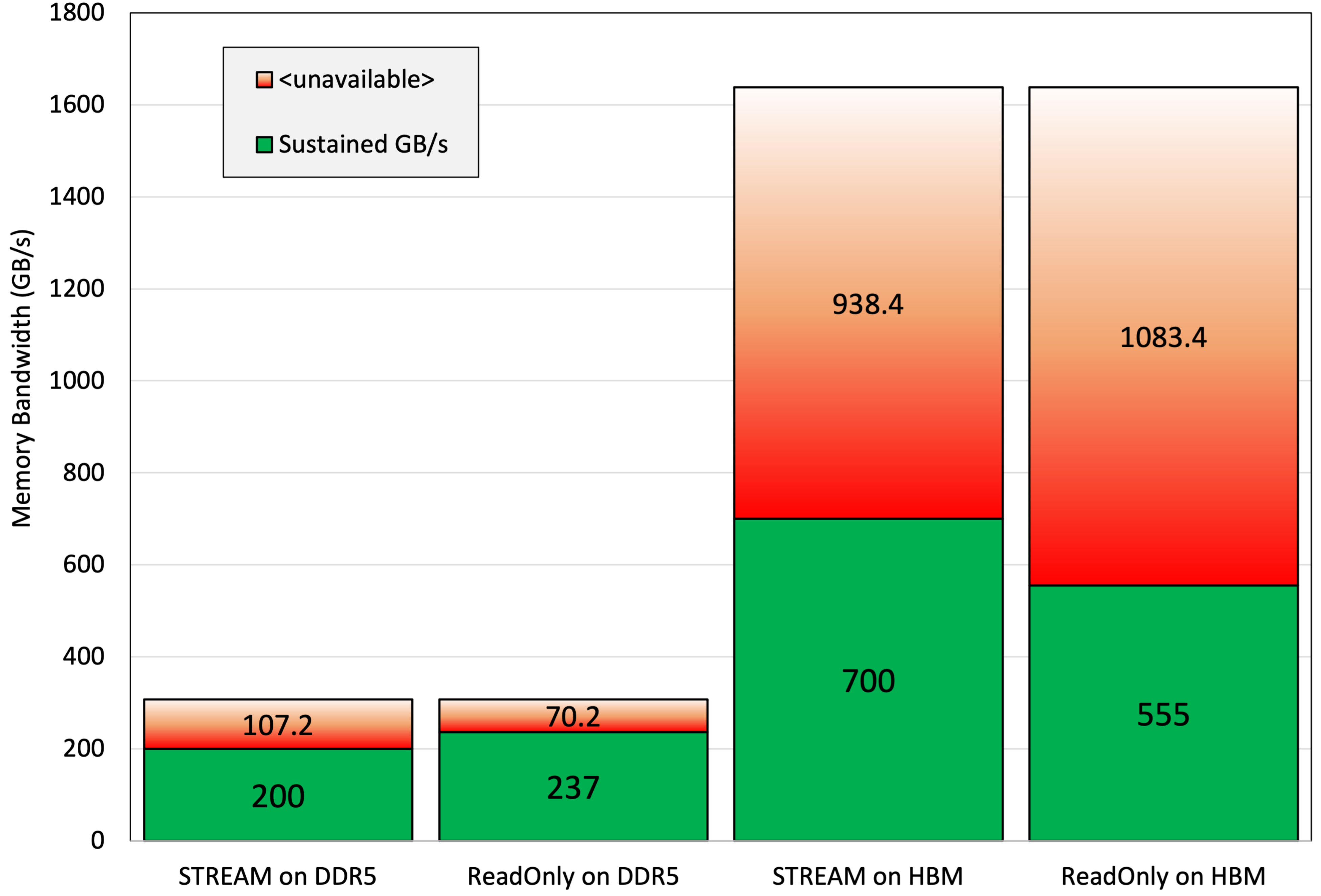
Measured bandwidth with HBM is significantly higher than with DDR5, but not as much higher as the ratio of the peak bandwidths.

This talk explains why….

Test Systems

- Xeon Max 9480 DDR5
  - 8 channels at 4.8 GT/s (1 single-rank DIMM/channel)
  - 8 * 8 * 4.8 = 307.2 GB/s (peak) DDR5 BW per socket
- Xeon Max 9480 HBM
  - 4 HBM stacks with 3.2 GT/s transfer rate
  - 4 * 128 * 3.2 = 1638.4 GB/s (peak) HBM BW per socket
- Ratio of Peak Bandwidths: 5.33x
- Most tests run in “flat, quadrant” mode
  - 1 NUMA node per socket

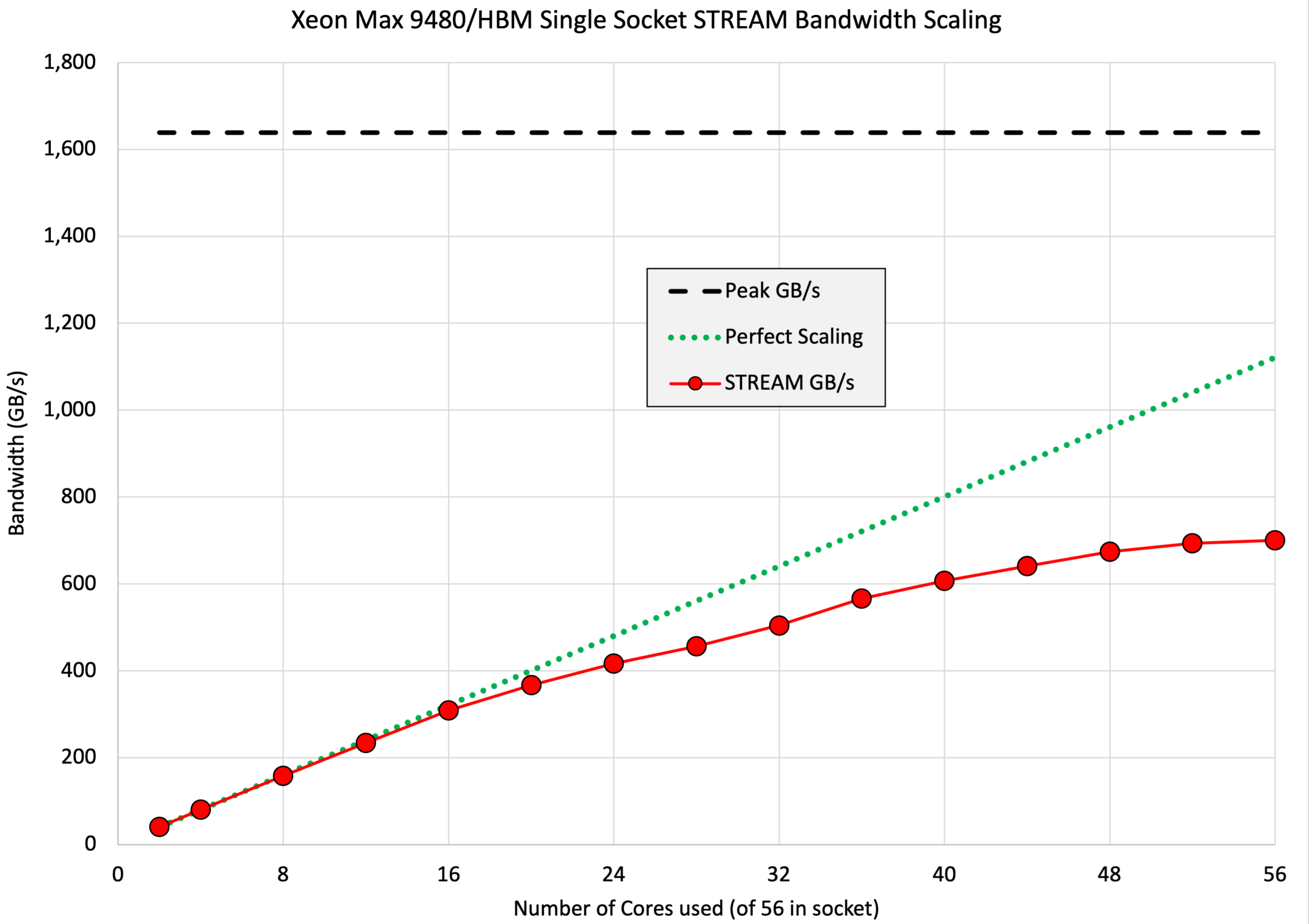
Sustained vs Peak Memory Bandwidth for 1-socket Xeon Max 9480



HBM delivers:
- 3.5x higher STREAM BW
- 2.3x higher Read BW

But much of the HBM BW remains unavailable…

Why?



Could this be a parallel scaling problem?

~68% of peak

~43% of peak

Even perfect scaling is not enough – this must be a per-core problem!

What Limits Single-Core Bandwidth?

- Inadequate ***concurrency*** relative to memory ***latency***
- The equation is the same as "Little's Law" from queueing theory

Concurrency = Latency * Bandwidth

– or –

Concurrency = Latency / Gap

- This is the fundamental "physics" of data transfer in computers!
- Little's Law can be rearranged and applied in many ways…

Understanding Concurrency with Cache Line transfers

Assume 64 Byte Cache Lines and

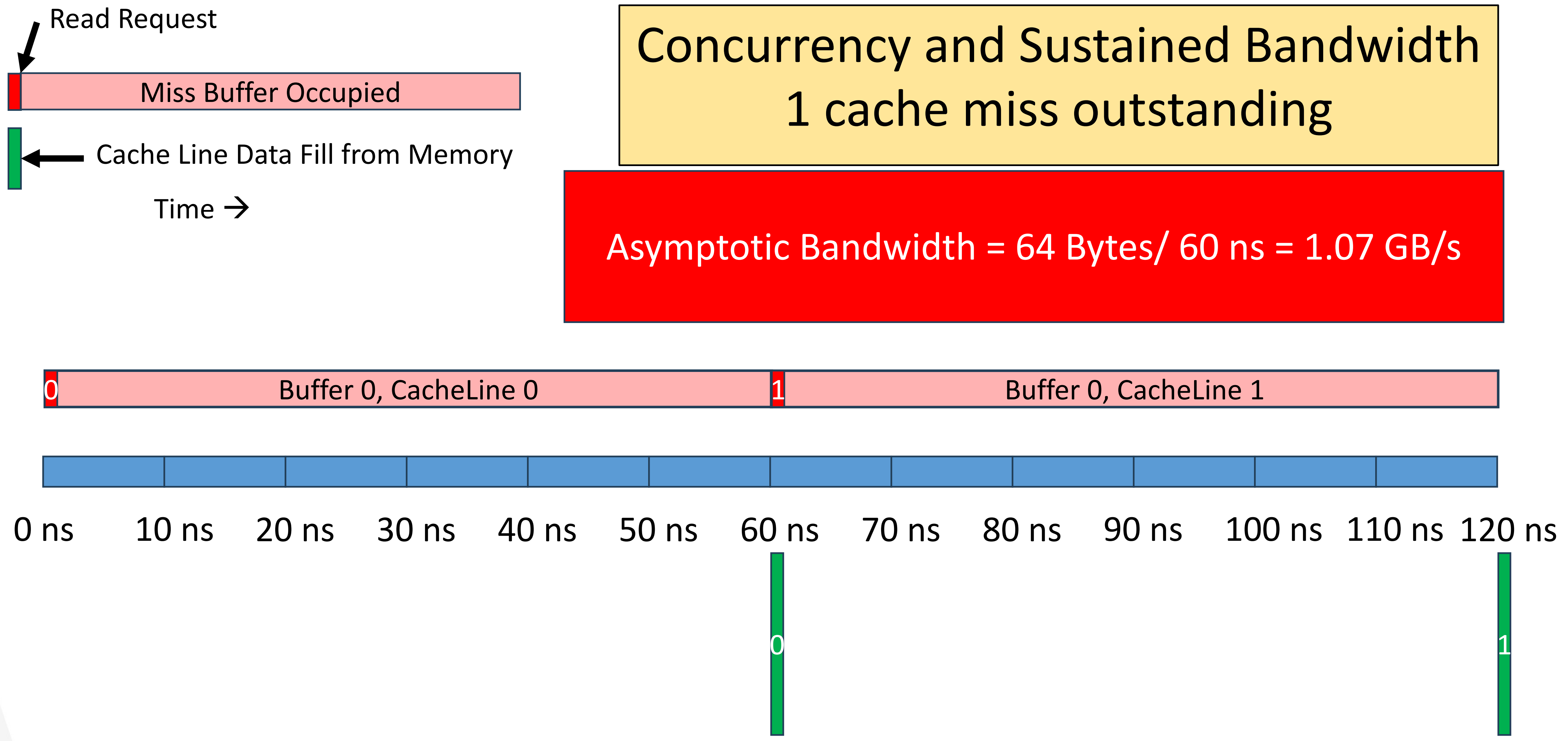
- 64 GB/s peak memory bandwidth
- 1.0 ns/line transfer time (“gap”)
- 60 ns memory latency

Then

- To receive 1 line/ns (100% BW) you must request 1 line/ns.
- 60 requests are made before the first data is returned
- I.e., 60 cache lines are “in flight” at all times
- The cache miss “concurrency” is 60 cache line misses (read requests to memory)

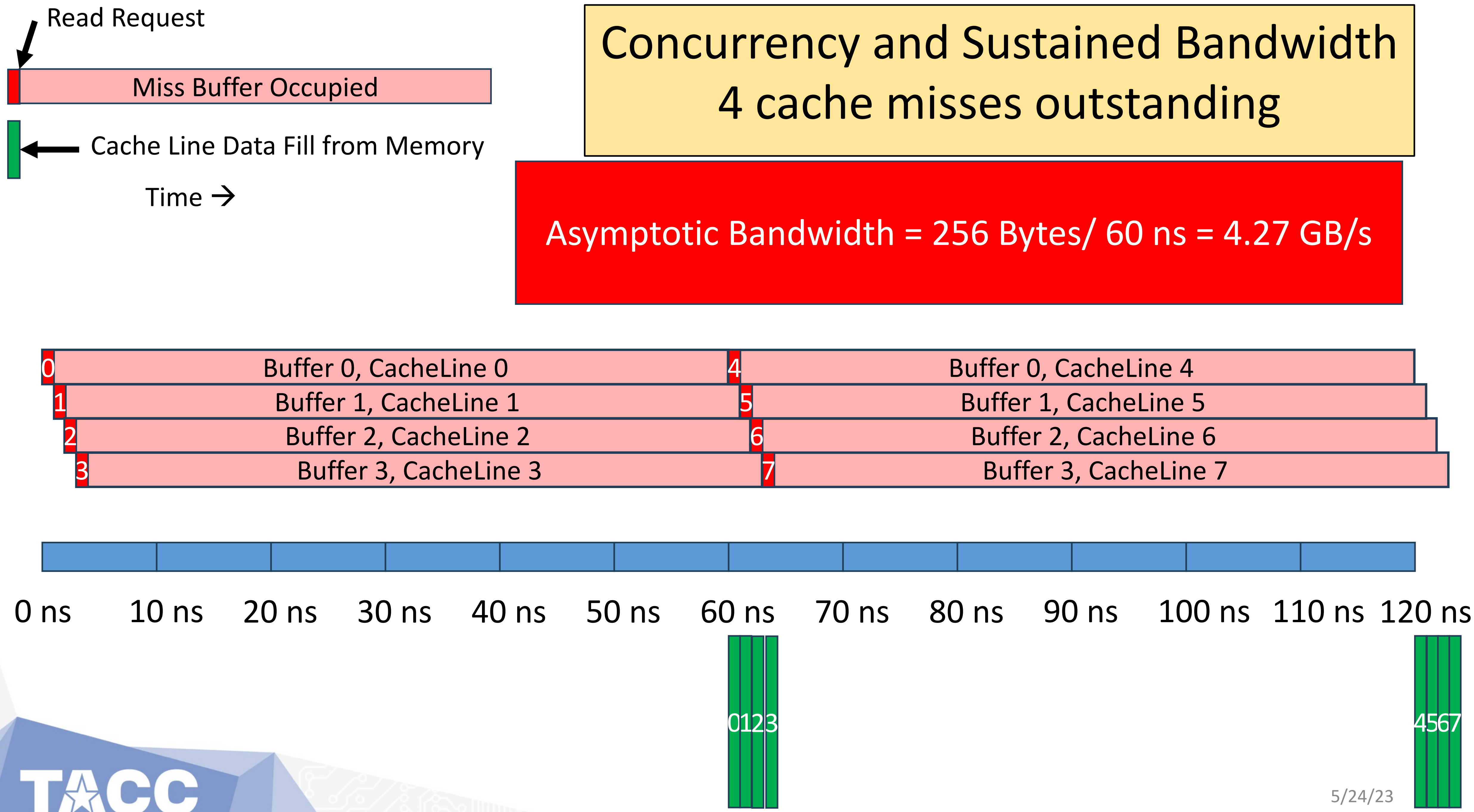
Concurrency and Sustained Bandwidth
1 cache miss outstanding

Asymptotic Bandwidth = 64 Bytes/ 60 ns = 1.07 GB/s



Concurrency and Sustained Bandwidth
4 cache misses outstanding

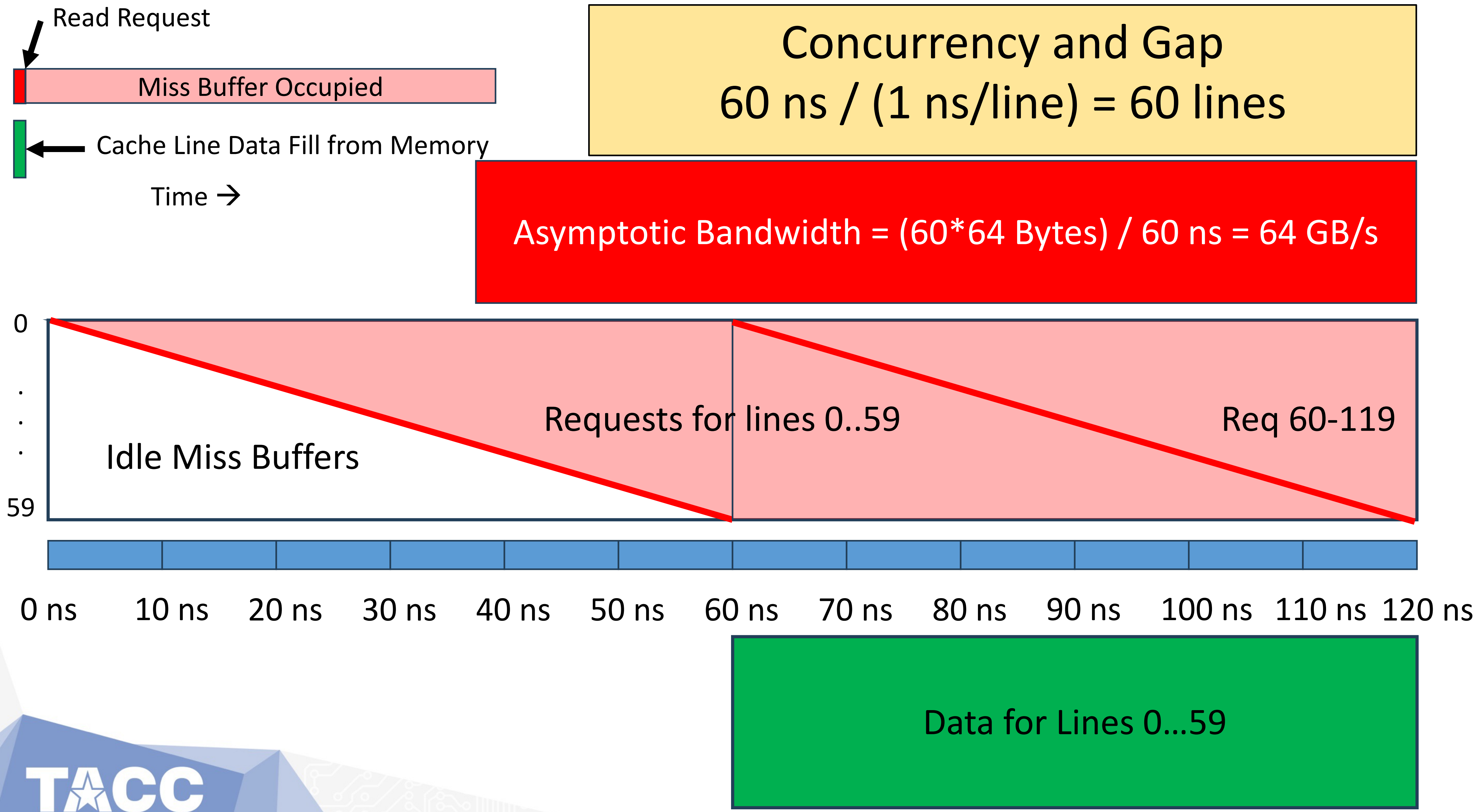
Asymptotic Bandwidth = 256 Bytes/ 60 ns = 4.27 GB/s



Concurrency and Gap

60 ns / (1 ns/line) = 60 lines

Asymptotic Bandwidth = (60*64 Bytes) / 60 ns = 64 GB/s



What about Xeon Max 9480/HBM?

Single Core Resources

- Up to 16 L1 Data Cache Misses
- Up to 48 L2 Cache Misses
  - Must exploit L2 Hardware Prefetches to exceed 16 requests
- Latency = 130 ns
- BW limit => 48*64 Bytes/130ns = 23.6 GB/s
- Consistent with observations? Yes!
  - All Read benchmark ~22 GB/s
  - STREAM benchmark ~20 GB/s

TACC

Scaling to all cores

- Required Concurrency for full BW at idle latency

  1638.4 GB/s * 130 ns = 3328 cache lines
- Maximum Available Concurrency

  56 cores * 48 L2 misses/core = 2688 L2 misses/chip
- Projected BW limit

  2688 cache lines / 130 ns = 1323 GB/s
- Maximum measured Read BW

  ~590 GB/s
- What went wrong?

Bridging theory and observation

- Intel core performance counters allow us to track the concurrency

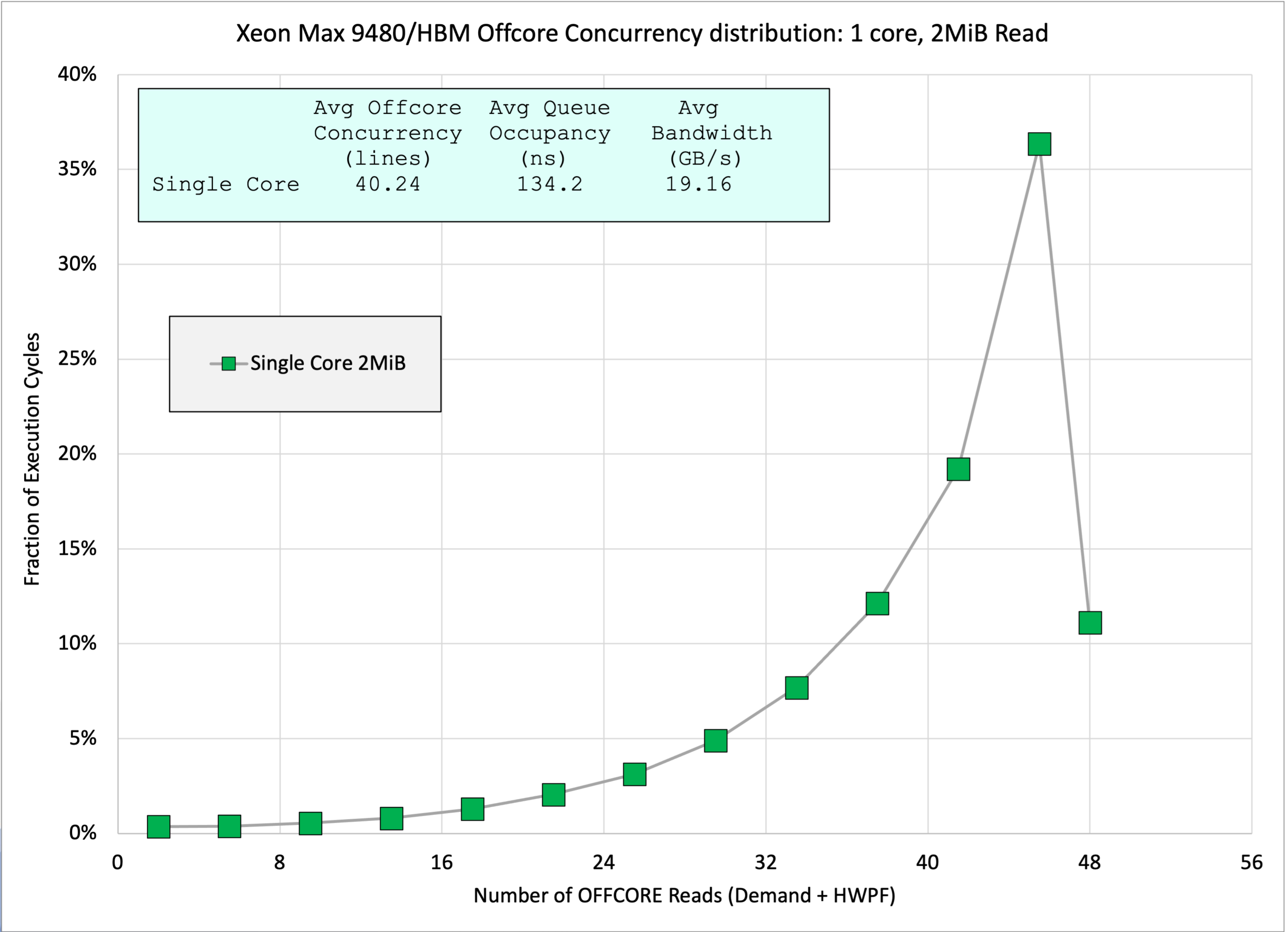
1. Sum outstanding L2 miss data reads (demand & prefetch) each cycle
   - `OFFCORE_REQUESTS_OUTSTANDING.DATA_RD`
2. Number of cycles with at least one outstanding L2 Miss Data Read
   - `OFFCORE_REQUESTS_OUTSTANDING.CYCLES_WITH_DATA_RD`
3. Number of L2 miss data reads (demand & prefetch)
   - `OFFCORE_REQUESTS.DATA_RD`

Compute

- Avg number of outstanding misses = (1) / (2)
- Avg cycles occupied by a transaction (1) / (3)

Bridging single-core results

- Single core theory:
  - 48 outstanding misses (max)
  - 130 ns latency
  - 23.6 GB/s bandwidth (max)
- Single core measurements:
  - 40.2 outstanding misses (avg)
  - 134.2 ns average queue occupancy
  - ~19 GB/s
- Occupancy looks good, but not all queues are always in use



	Avg Offcore Concurrency (lines)	Avg Queue Occupancy (ns)	Avg Bandwidth (GB/s)
Single Core	40.24	134.2	19.16

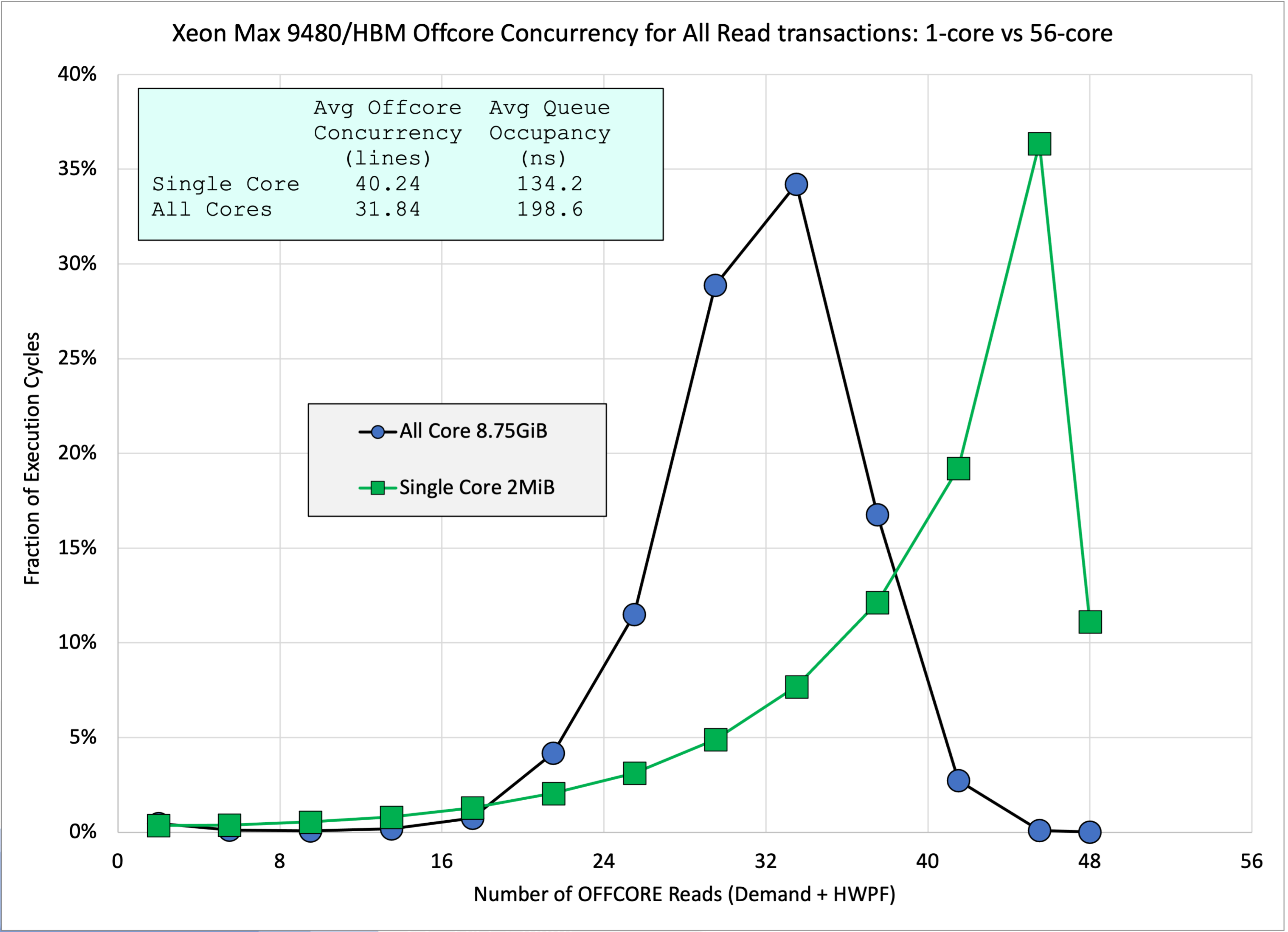
Not all L2 Miss buffers are used all the time.

Issuing 48 Read Requests to the mesh probably takes at least 48 cycles….

This suggests that the average fraction of buffers in use will likely decrease as the total number of L2 Miss buffers increases….

Bridging multi-core results

- Theory for 56 cores
    - 48 outstanding misses (max)
    - 130 ns latency
    - 1323 GB/s bandwidth (max)
- 56-core measurements:
    - 31.8 outstanding misses (avg) ← 1.51x BW reduction
    - 198.6 ns average queue occupancy ← 1.52x BW reduction
    - ~575 GB/s
- The average queue occupancy is in the expected range from measurements using the Intel Memory Latency Checker
- Unknown why average concurrency is lower



```
             Avg Offcore  Avg Queue
             Concurrency  Occupancy
               (lines)       (ns)
Single Core     40.24       134.2
All Cores       31.84       198.6
```

The L2 Hardware Prefetch engine is known to be dynamically adaptive, but with unknown inputs and unknown heuristics.

~1.26x reduction in BW due to reduced average concurrency
~1.48x reduction in BW due to increased average latency (occupancy)

The distribution of concurrency also changes for the single-thread case as a function of number of cache lines read. Prefetch becomes more aggressive for larger sizes.

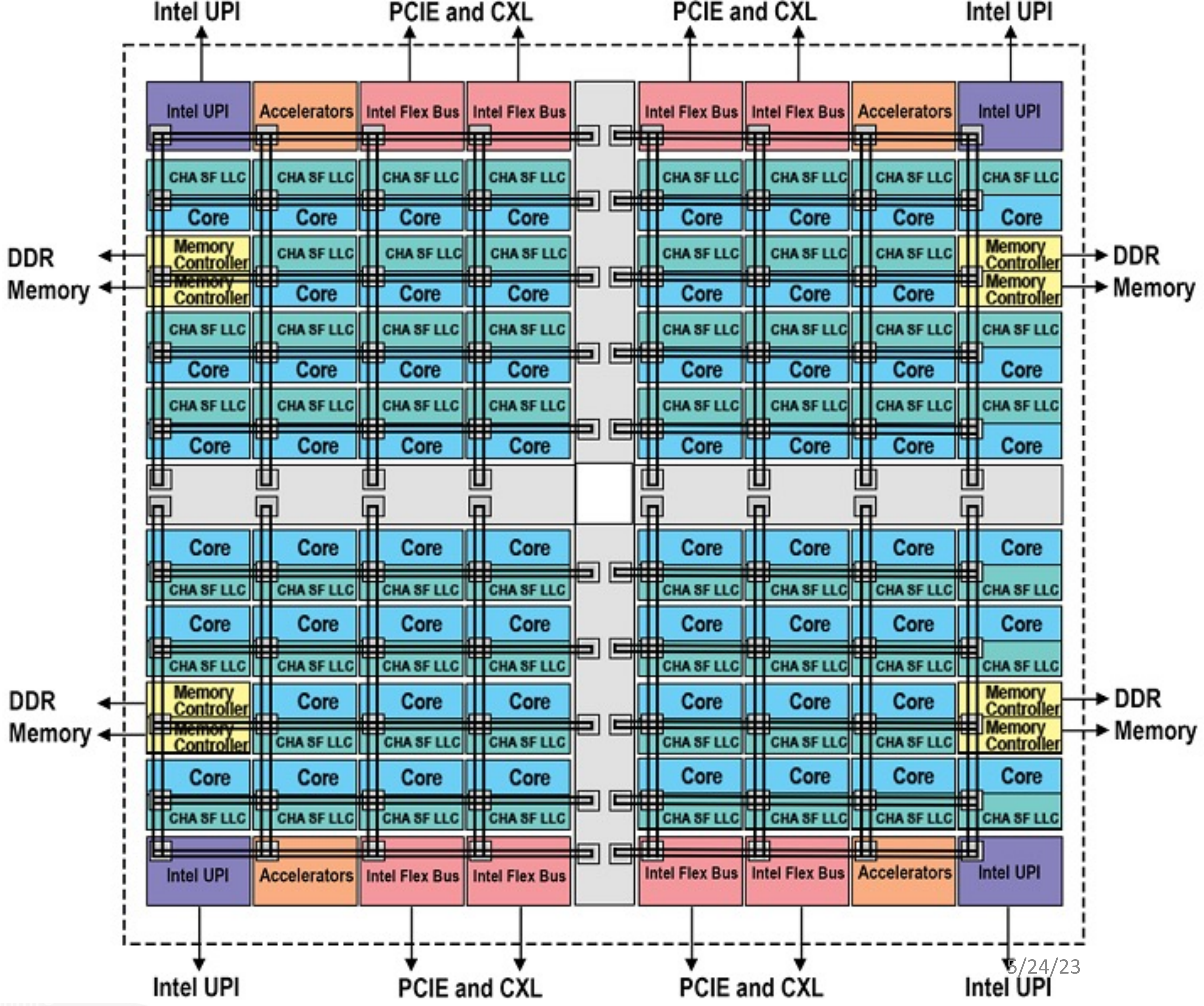
What about SNC4 mode?

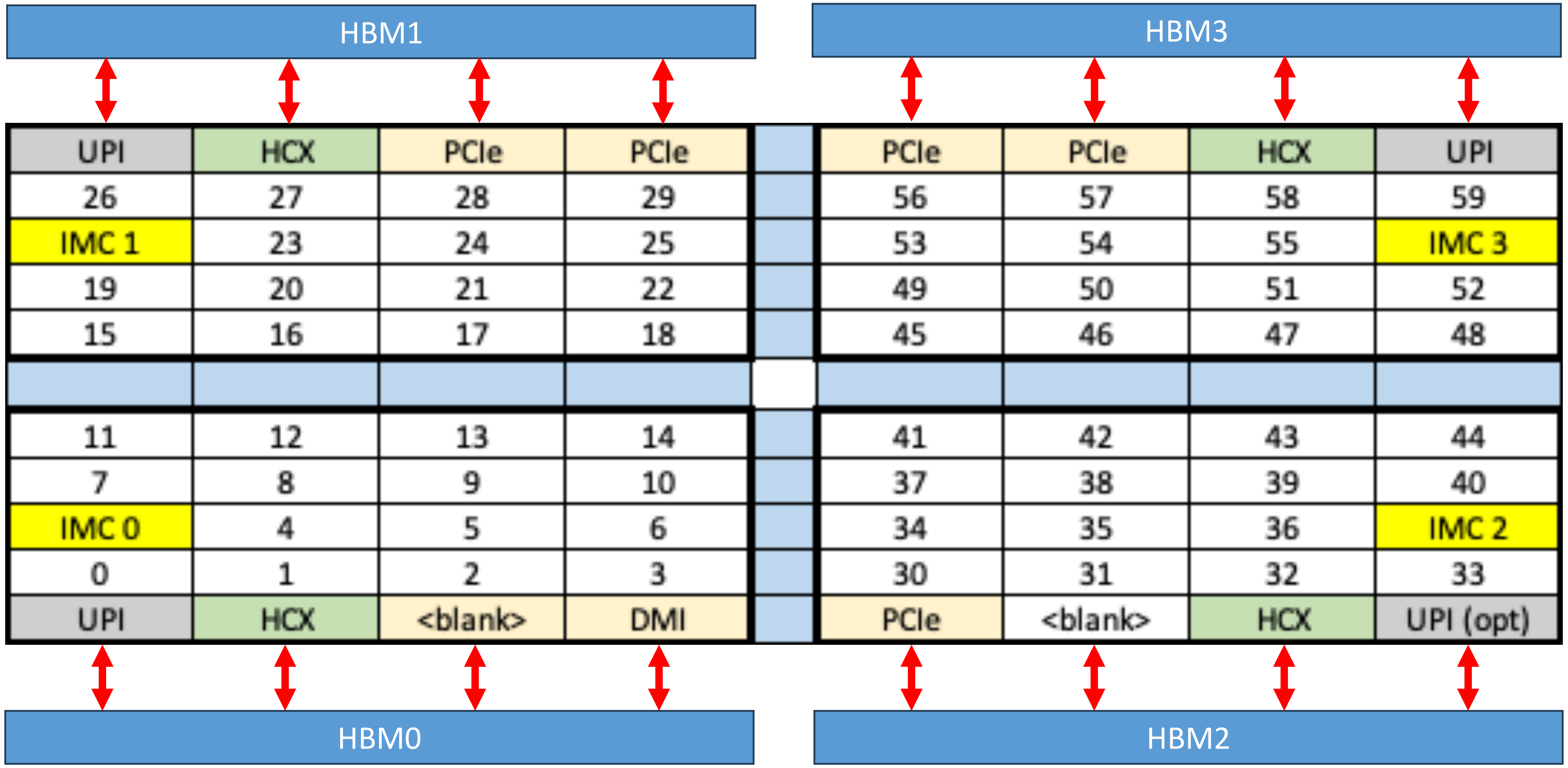
- SNC4 mode reduces average latency
  - Fewer average hops
  - No die-to-die crossings
  - No mesh traffic contention with cores in other quadrants
- SNC4 mode increases sustained bandwidth, but results are variable and somewhat confusing.
- The reduction in latency is not enough to allow full bandwidth, even with perfect core scaling
  - MLC All Read BW is ~707 GB/s per socket (43% of peak)
  - Not yet reproduced with instrumentation

What if Intel significantly increases concurrency?

- Additional bandwidth limiters lie close behind
- Mesh (all-core)
- Mesh (single-core)

Sapphire Rapids layout





UPI	HCX	PCIe	PCIe		PCIe	PCIe	HCX	UPI
26	27	28	29		56	57	58	59
IMC 1	23	24	25		53	54	55	IMC 3
19	20	21	22		49	50	51	52
15	16	17	18		45	46	47	48
11	12	13	14		41	42	43	44
7	8	9	10		37	38	39	40
IMC 0	4	5	6		34	35	36	IMC 2
0	1	2	3		30	31	32	33
UPI	HCX	<blank>	DMI		PCIe	<blank>	HCX	UPI (opt)

HBM bandwidth (409.6 GB/s per stack) is too high for a single mesh link (51.2 – 76.8 GB/s), so it is distributed uniformly across the four column inputs of the quadrant.

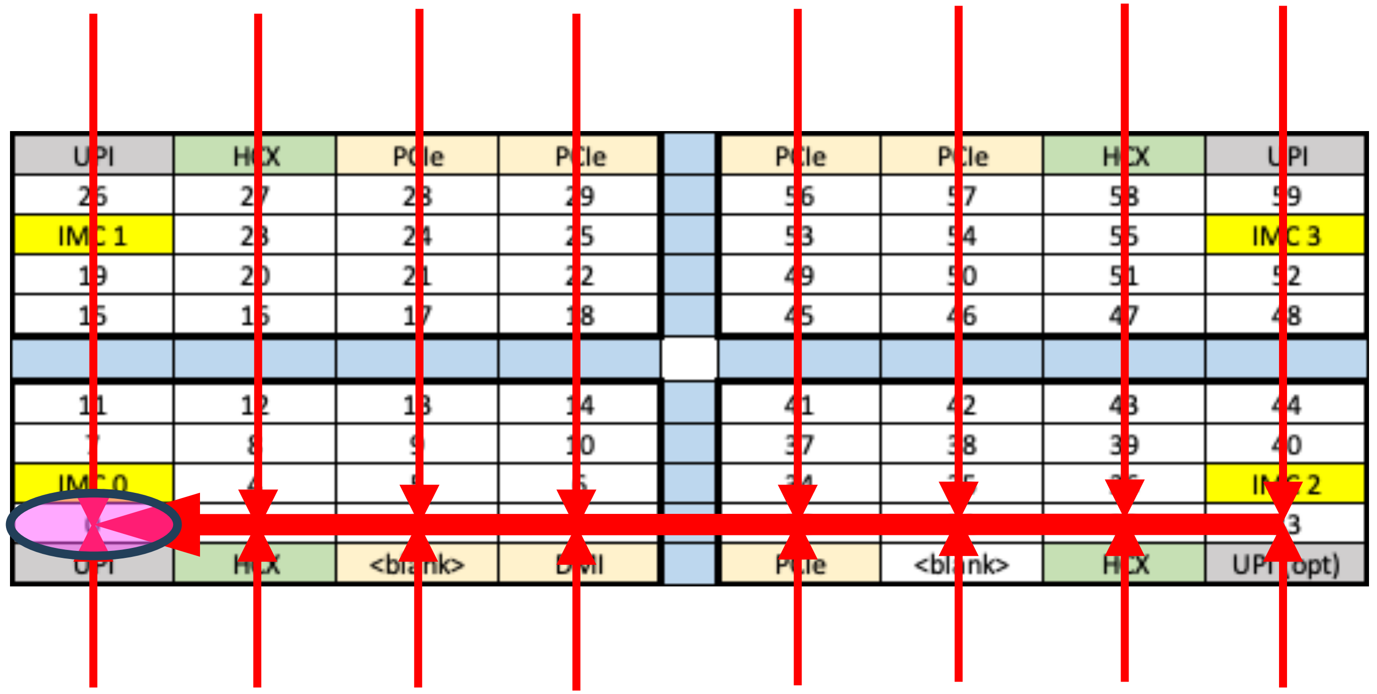
All-Core Mesh BW Limits?

- On TACC's Xeon Max 9480 processors, the Uncore frequency drops to 1.6 GHz when under load.
- Each Mesh link is 32 Bytes wide (per direction), giving a peak BW of 51.2 GB/s (per direction)
- The HBM traffic is uniformly spread across 16 vertical links (4 per quadrant), for a peak BW of 102.4 GB/s per mesh link.
- 100% Read Bandwidth from the HBMs is exactly twice the mesh capacity.
- This limit is the same for both flat mode and SNC4 mode.

Single Core Mesh BW Limits?

- Y-X routing of HBM traffic results in severe traffic imbalances for cores on the left and right edges of the mesh

Core 0, Flat mode:
 14/16 of reads arrive on the left-bound mesh link



Each core can receive data on a link every other cycle, for a throughput of 25.6 GB/s @ 1.6 GHz. With 14/16 of the traffic on one link, peak read BW is limited to 29.2 GB/s for cores in the left and right columns.

Cores in the center columns would be limited to 51.2 GB/s by the busiest input mesh link.

Summary

- Xeon Max processors provide significantly (2.5x – 3.5x) higher sustained memory bandwidth from HBM than can be obtained from DDR5 memory.
- The cores used in the Xeon Max processors do not support enough concurrency to reach full HBM bandwidth with the current core counts. 2x more concurrency or 2x more cores.
- The 2D mesh interconnect in the Xeon Max processors only provides ½ of the bandwidth required to reach full read BW from HBM.
- Routing HBM traffic on the mesh would cause visible core-to-core differences in sustained BW if the cores supported more concurrency.