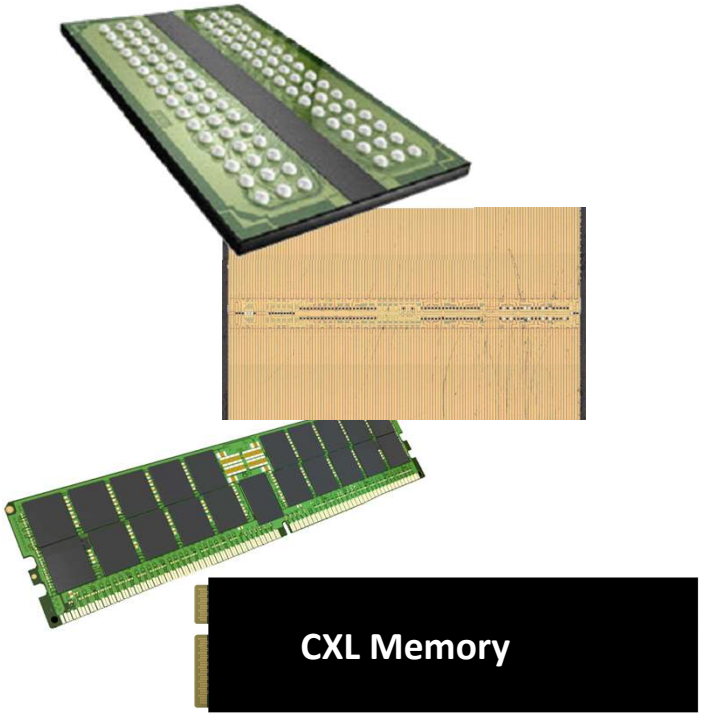
DRAM in an Increasingly Diverse Platform



Bill Gervasi, Principal Systems Architect

Wolley Inc.

bilge@wolleytech.com

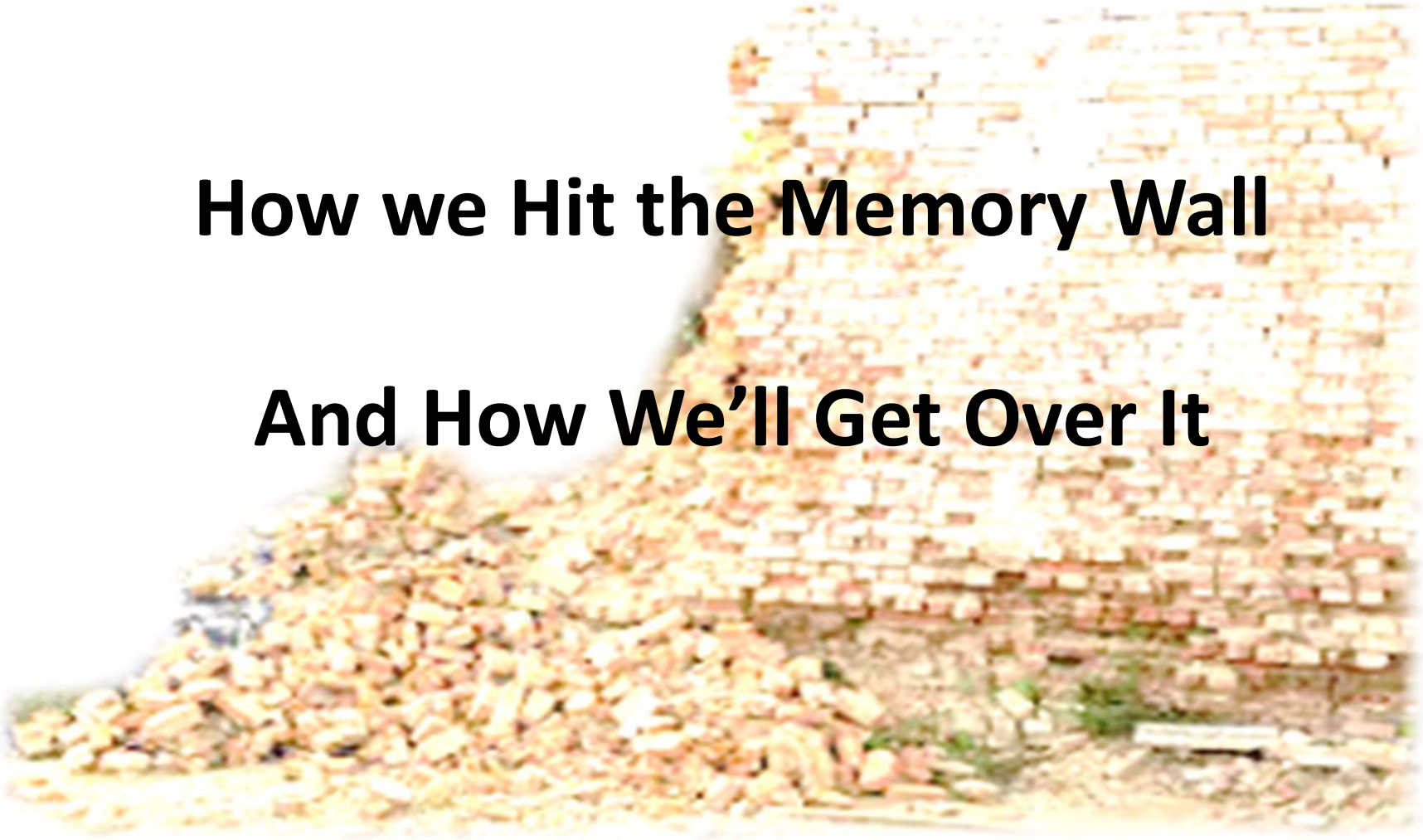
FMS the Future of Memory and Storage

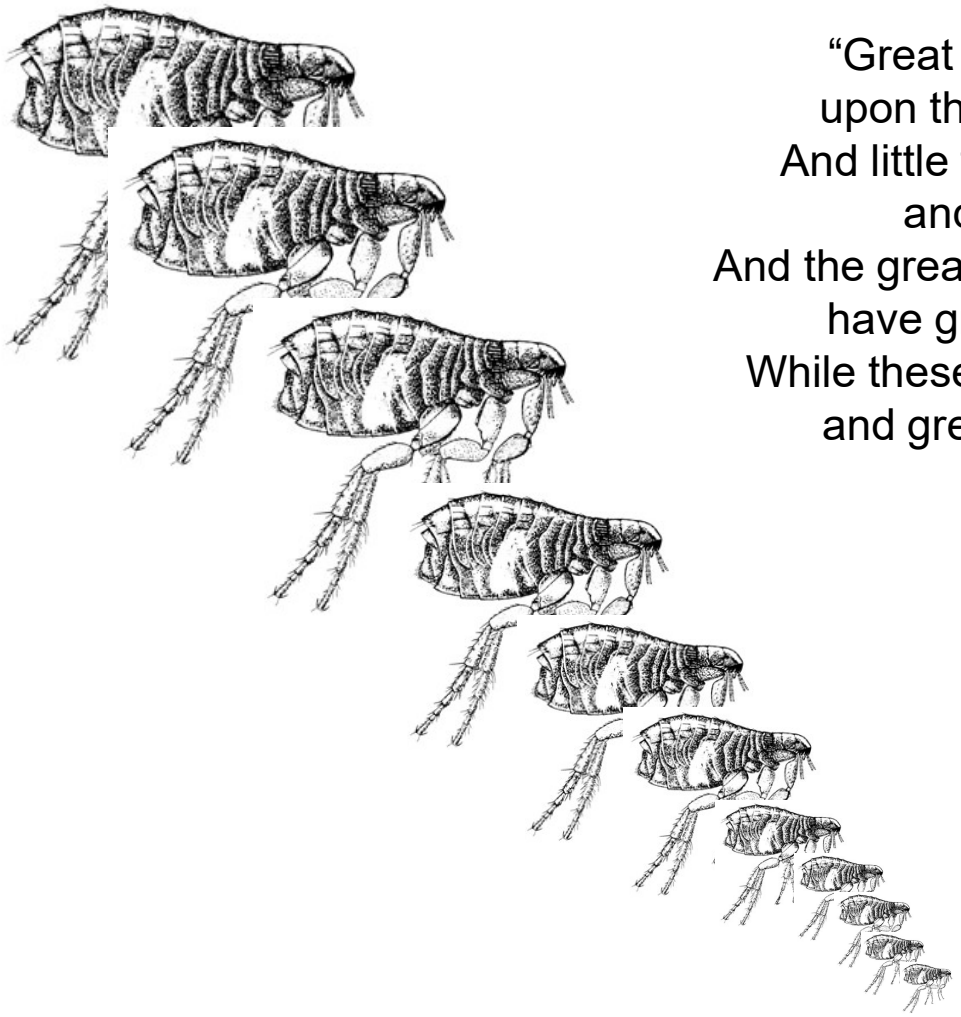
Today's Agenda

- DRAM Basics
- DRAM Challenges
- CXL: Emerging System Fabric
- Artificial Intelligence and Mobile Use Cases
- Power Challenges

How we Hit the Memory Wall

And How We'll Get Over It





"Great fleas have little fleas
upon their backs to bite 'em,
And little fleas have lesser fleas,
and so ad infinitum.
And the great fleas themselves, in turn,
have greater fleas to go on;
While these again have greater still,
and greater still, and so on."

Jonathan Swift

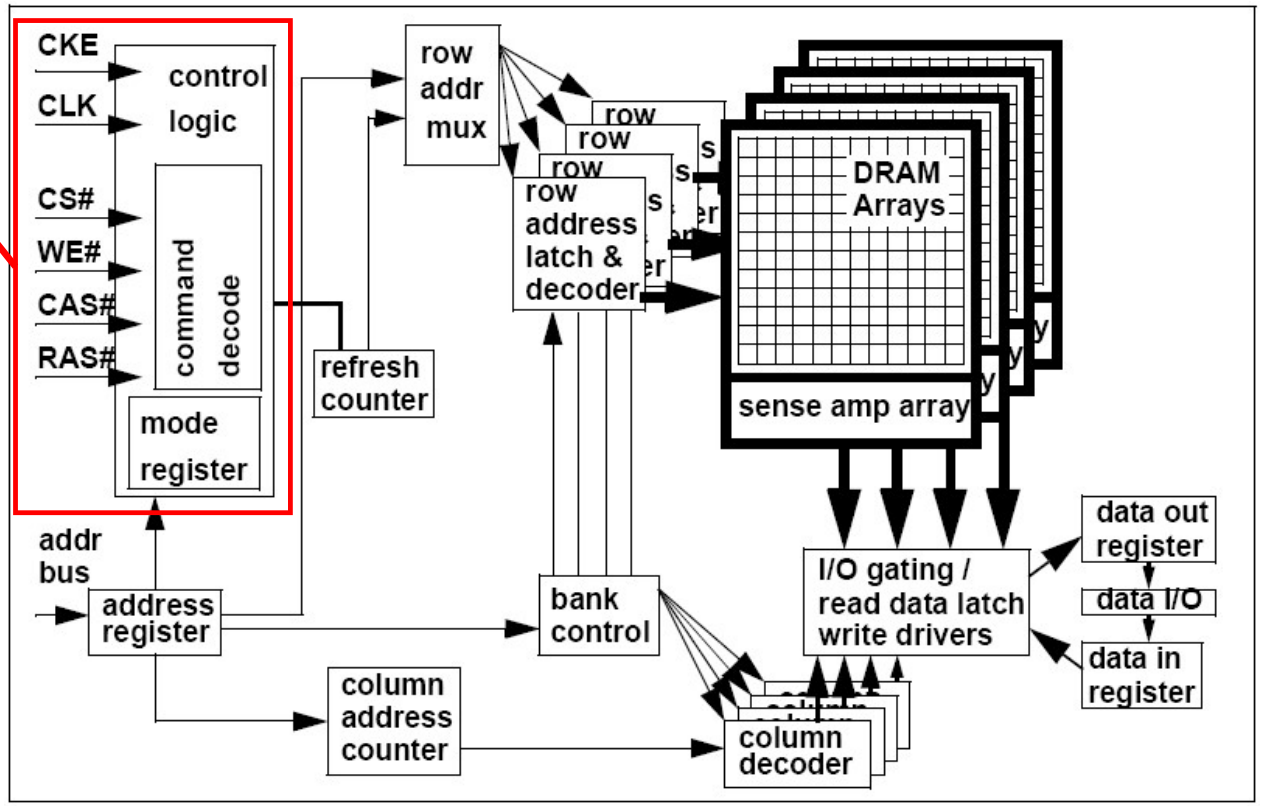
DRAM so far has resisted revolution

Just a number of evolutionary changes

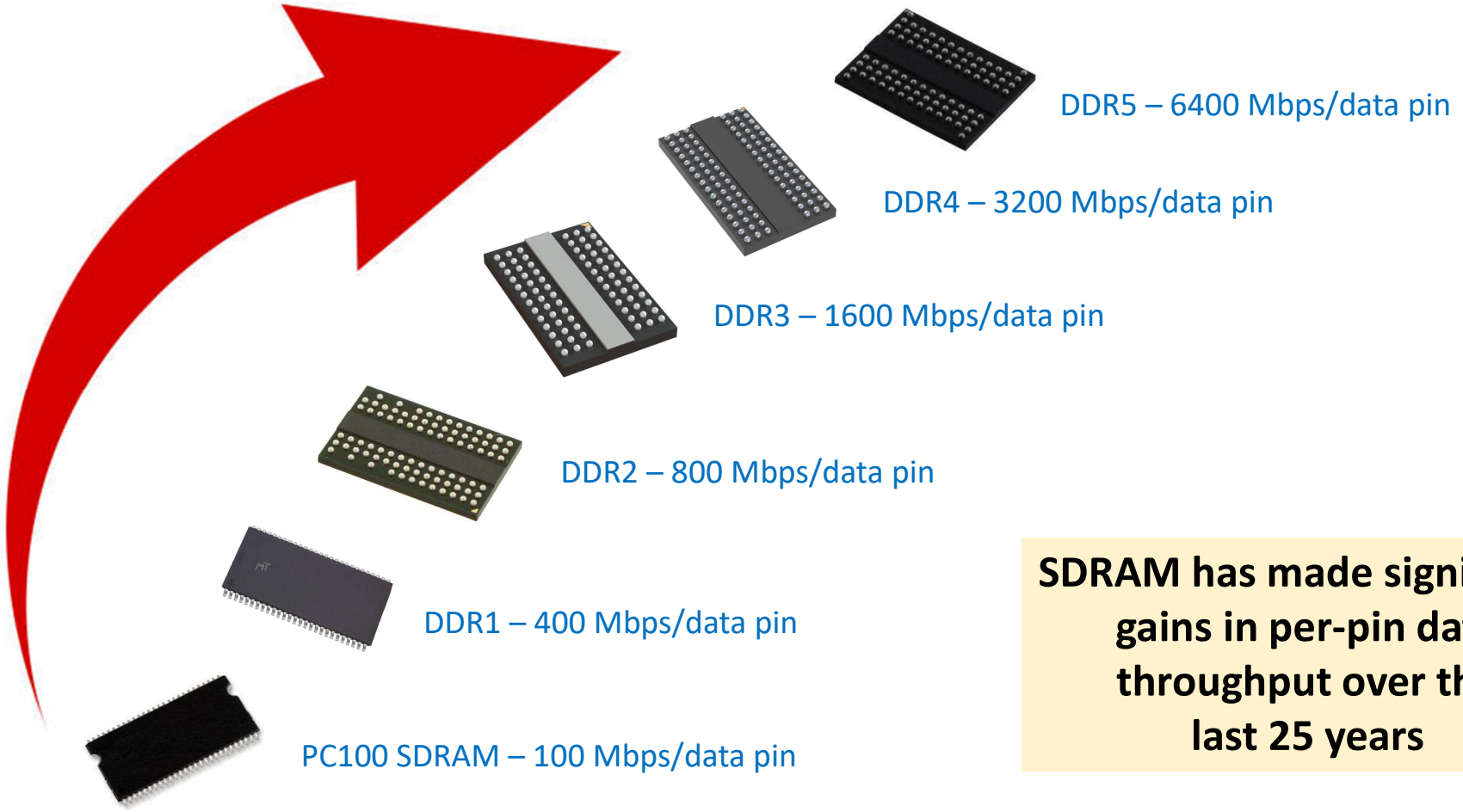
We are still using a core design >300 years old

DRAM core hasn't changed.

The vast majority of improvements have been faster, fancier I/Os

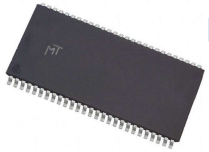


DRAM design ©1266 BCE

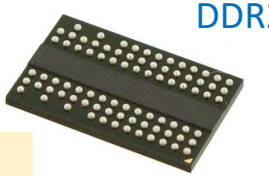


SDRAM has made significant gains in per-pin data throughput over the last 25 years

PC100 SDRAM – reference synchronous main memory



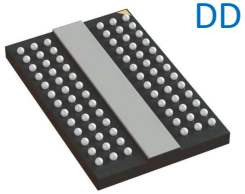
DDR1 – prefetch 2 bits, first main memory with a data strobe



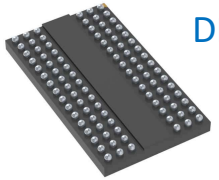
DDR2 – prefetch 4 bits, differential strobes, on-die termination

However, random access time has only improved 28%

‘cuz I/O is cheaper than core

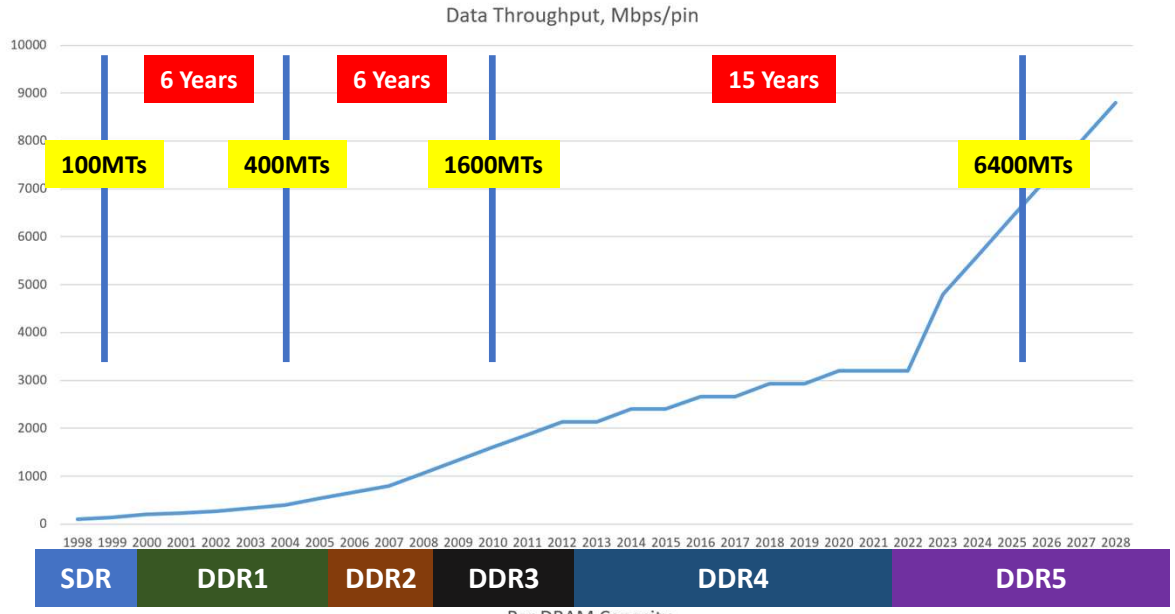


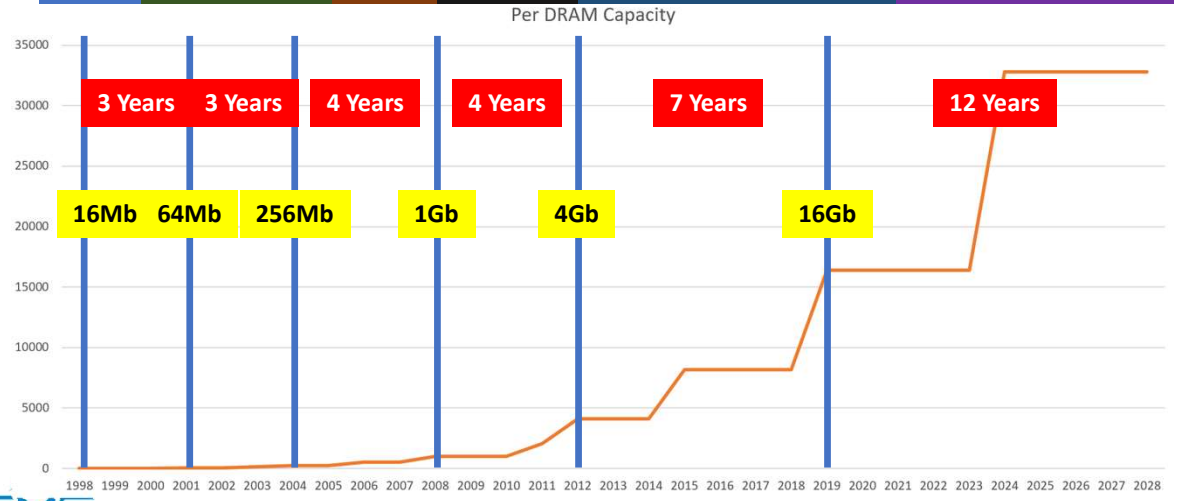
DDR3 – prefetch 8 bits, improved calibration, command-dependent ODT



DDR4 – improved calibration, ODT

DDR5 – Prefetch 16, improved calibration, PMIC





The good news:

Data throughput has had healthy increases

DDR5 was planned for 6400 Mbps max, now extended to 9200 Mbps

The bad news:

Speed improvements slowing

DRAM per-die capacity is taking longer with each generation

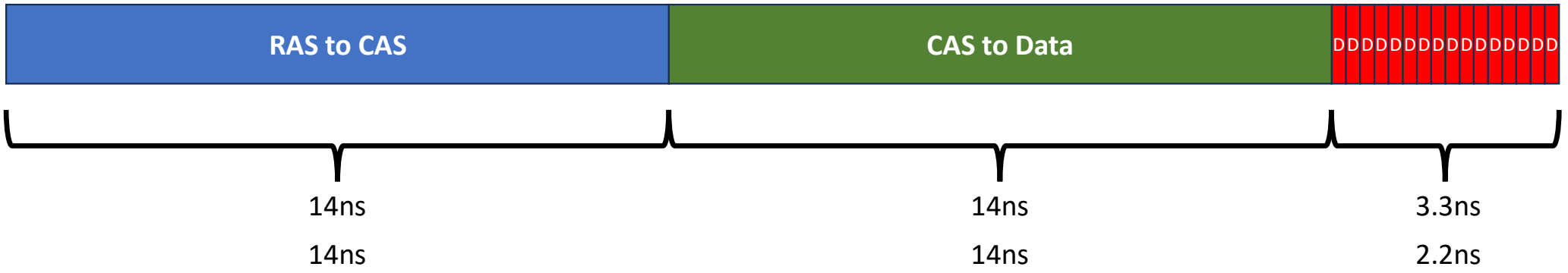
Was: quadrupling every 3 years
Is: quadrupling every 12 years

DDR5-4800: one clock = 208ps
Burst length 16 = data packet in 3.3ns

RAS-to-CAS ~ 14ns
CAS-to-Data ~ 14ns

DDR5-6400: one clock = 312ps
Burst length 16 = data packet in 2.2ns

RAS-to-CAS ~ 14ns
CAS-to-Data ~ 14ns



Transition from DDR5-4800 (BOL) to DDR5-6400 (EOL)

31.3 ns → 30.2 ns = 3.5% improvement
Random access burst

Why?

Customers pay for GB and not much else matters

How do these trends affect my system design?

How do I make the most of what we have?

Remember when this simple picture described our data tiers?

It helped us visualize the relative aspects of each tier

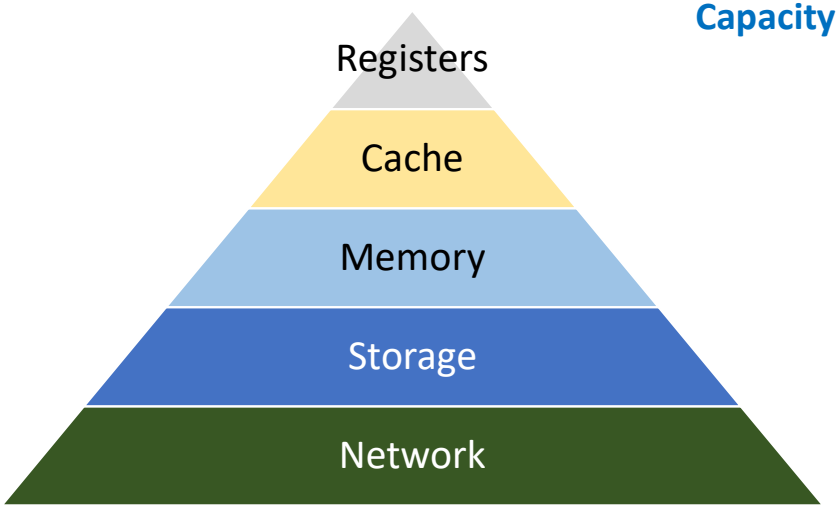
Performance

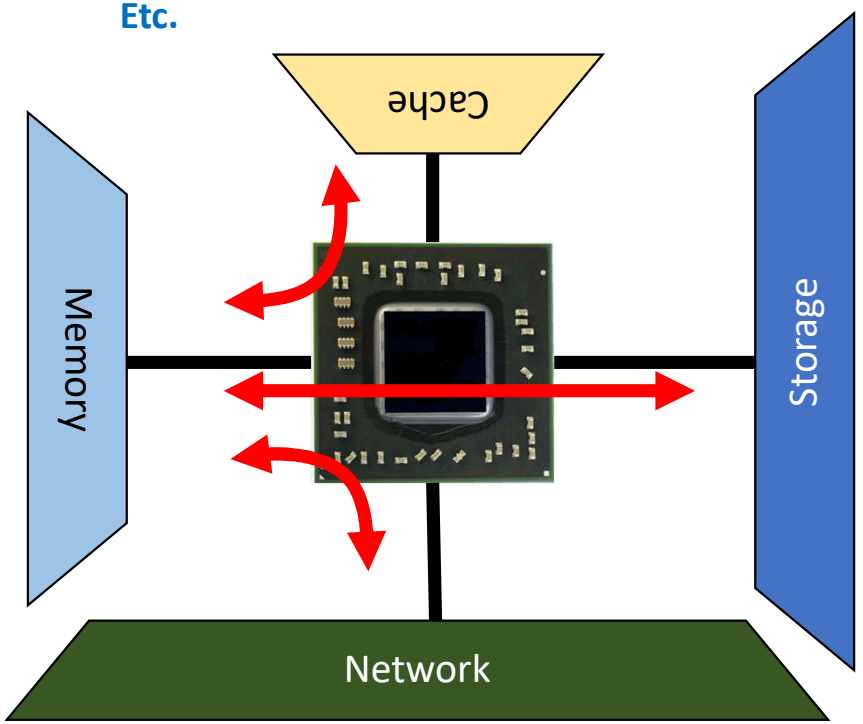
Capacity

Cost

Latency

Etc.

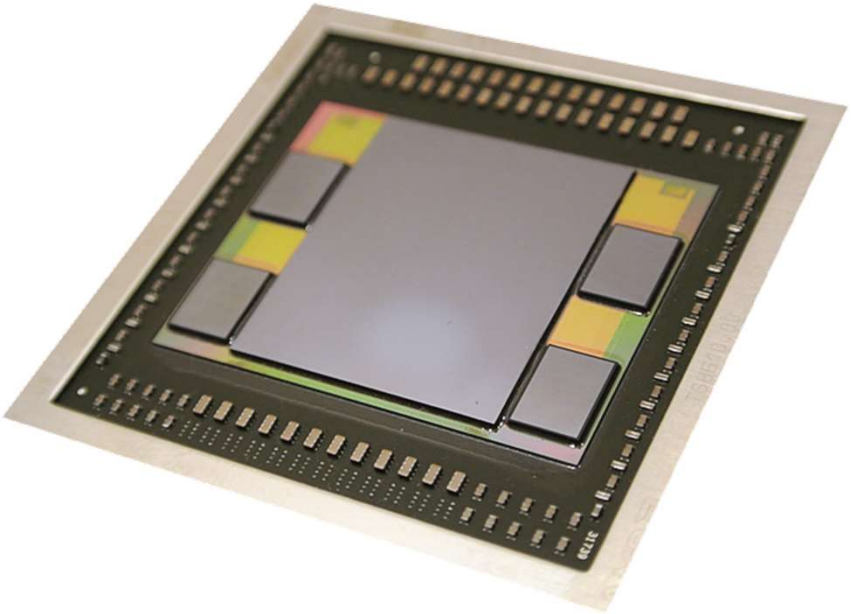


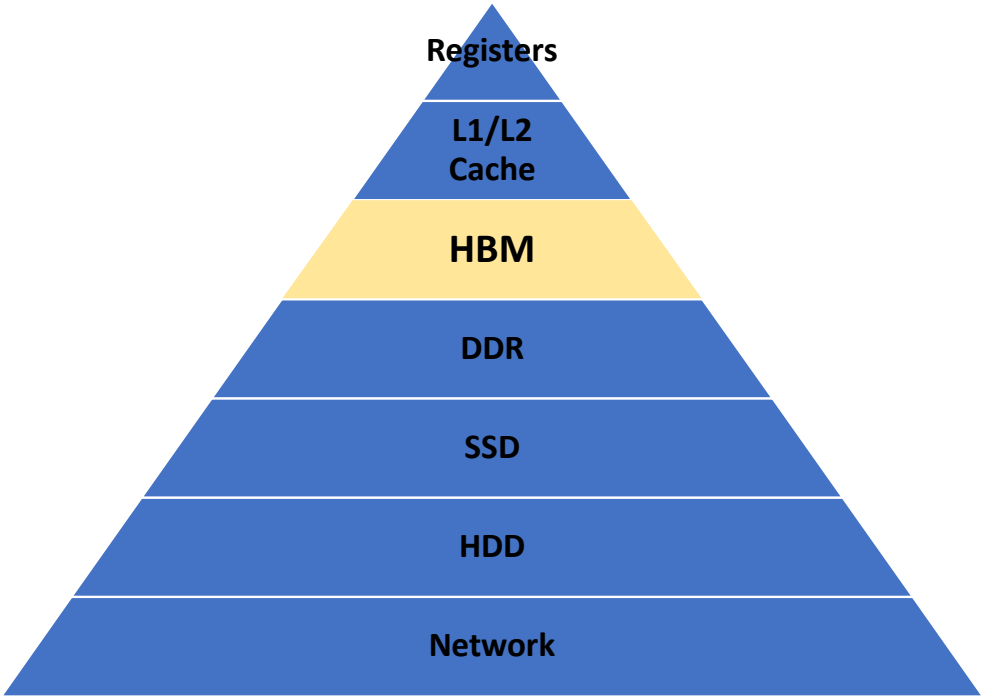


It has one significant weakness in that it doesn't show redundancy

Every tier being on its own interface means there is a LOT of data traffic between tiers

High Bandwidth Memory (HBM)





Significant addition to the memory pyramid

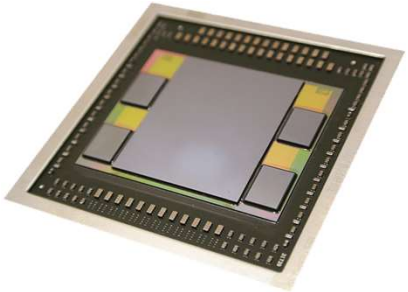
- High performance
- Low power per bit
- Mid-level capacity ~ 80GB
- Heavily deployed for AI

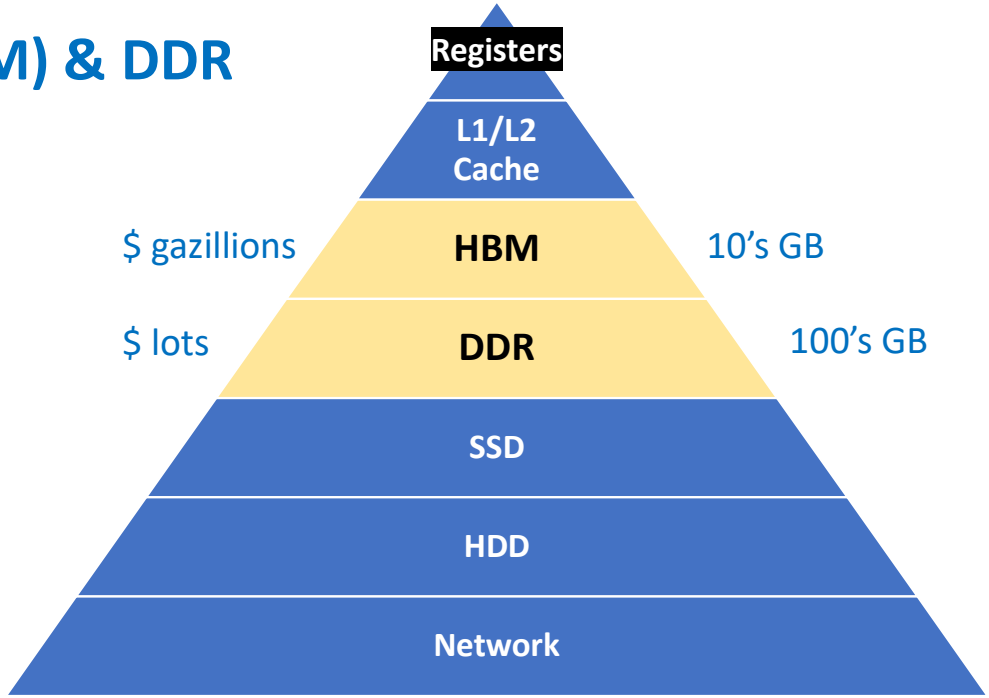
Some limitations

- Silicon substrate interconnect
- Low mm distance between processor and HBM
- Very $$$expensive
- Capacity cannot hold many modern data sets


Stay around for the session on HBM for details

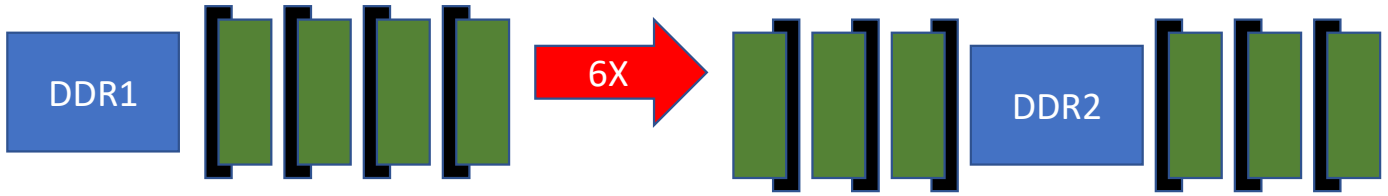
High Bandwidth Memory (HBM) & DDR





It's not either/or

It's in addition to

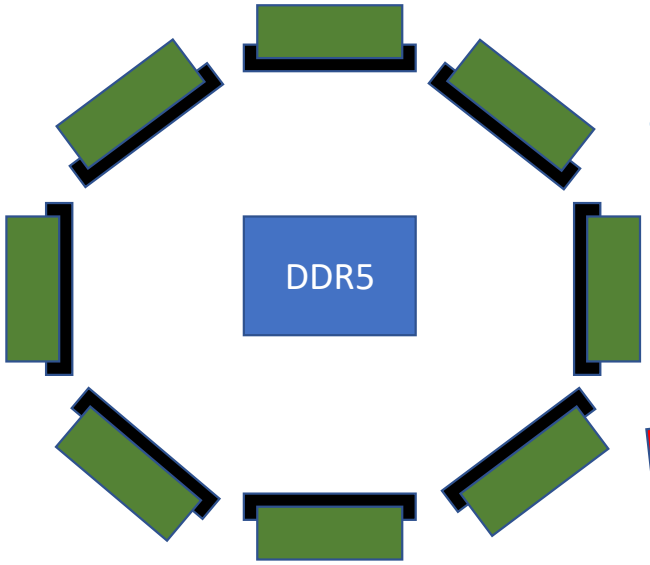


1 CH, 4 DPC, 2R/D, 256Mb = 4 GB

2 CH, 3 DPC, 2R/D, 1 Gb = 24 GB

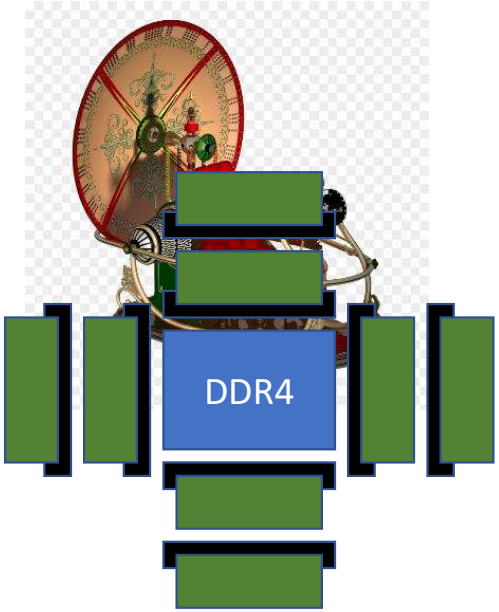
4X

Increasing frequency is slowing DIMM improvements



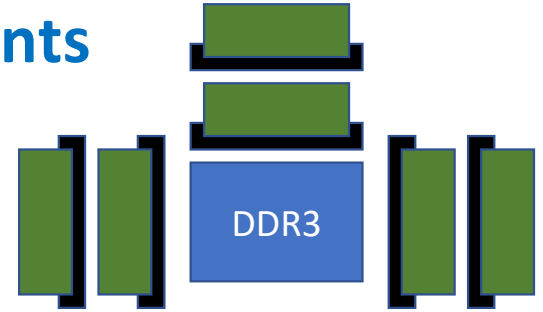
8 CH, 1 DPC, 2R/D, 16 Gb = 512 GB

1X



4 CH, 2 DPC, 2R/D, 16 Gb = 512 GB

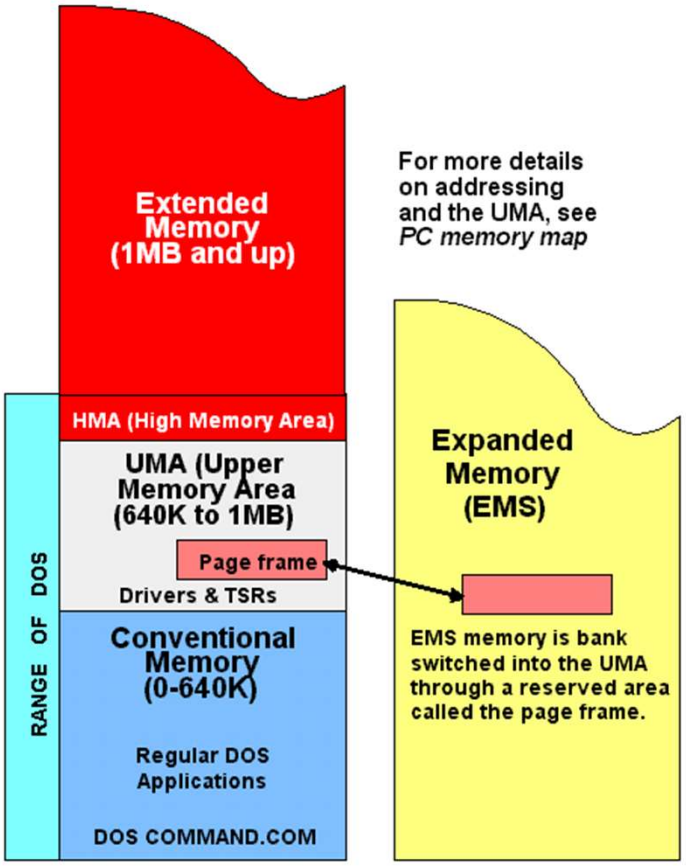
5X



3 CH, 2 DPC, 2R/D, 4 Gb = 96 GB

CH = channel
DPC = DIMMs per channel
R/D = ranks per DIMM

Assumes no 3DS



Memory Expansion is Not New

In the 1980s, Expanded and Extended Memory were common methods to grow the memory footprint of a PC beyond the CPU limits

Real time operating systems running on such systems had to comprehend the differences in access times

Memory Pooling is Also Not New

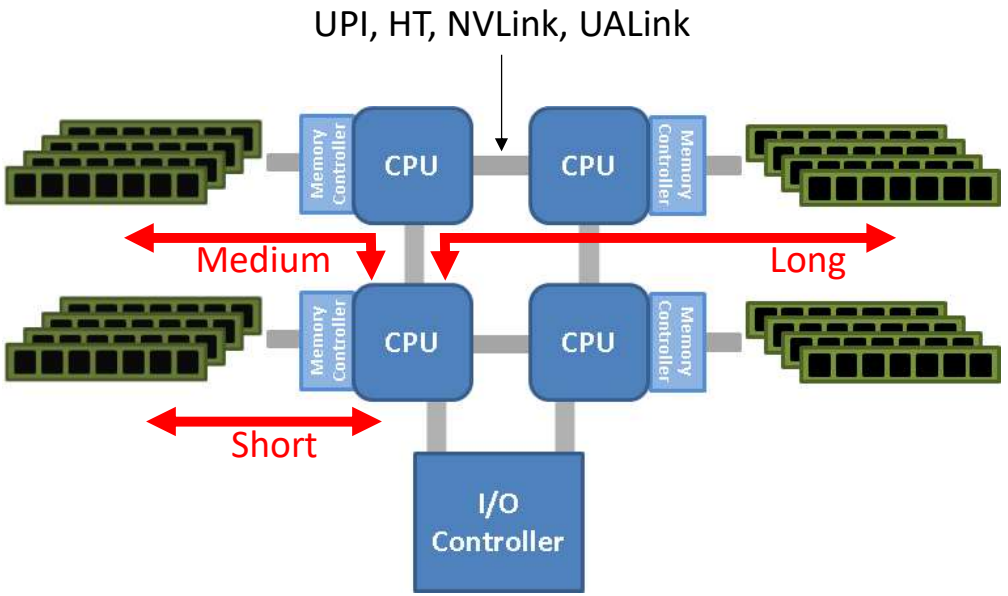
Non-Uniform Memory Architectures (NUMA) have been common ways to pool memory resources

Buses such as HyperTransport and Ultra Path Interconnect have been around for decades

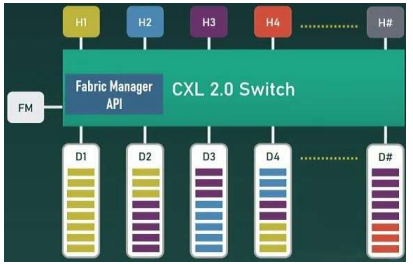
These NUMAs created a tier of resources

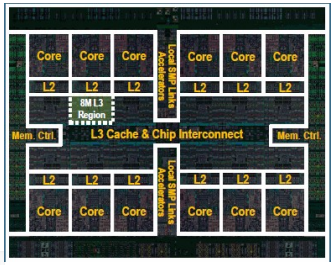
- Fastest memory attached to CPU
- Slower memory one hop away
- Slowest memory two hops away

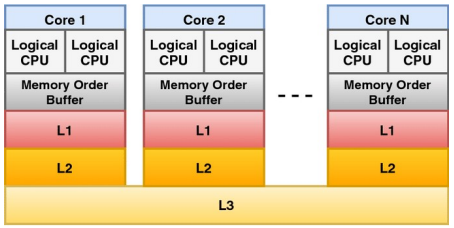
Smart software adjusted data location based on access latency

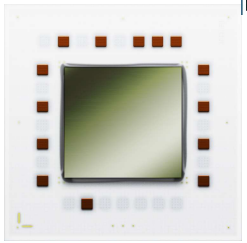


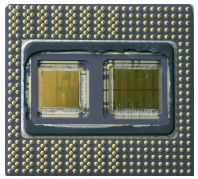
As CPUs grew hungrier

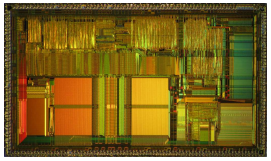




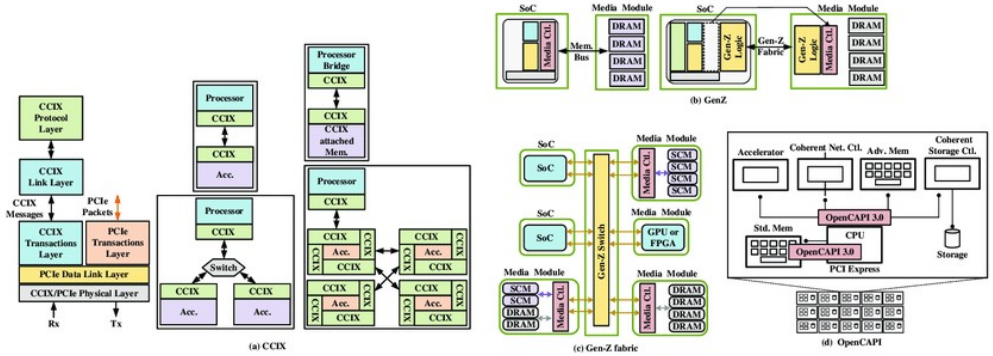








Memory solutions grew deeper and more complex

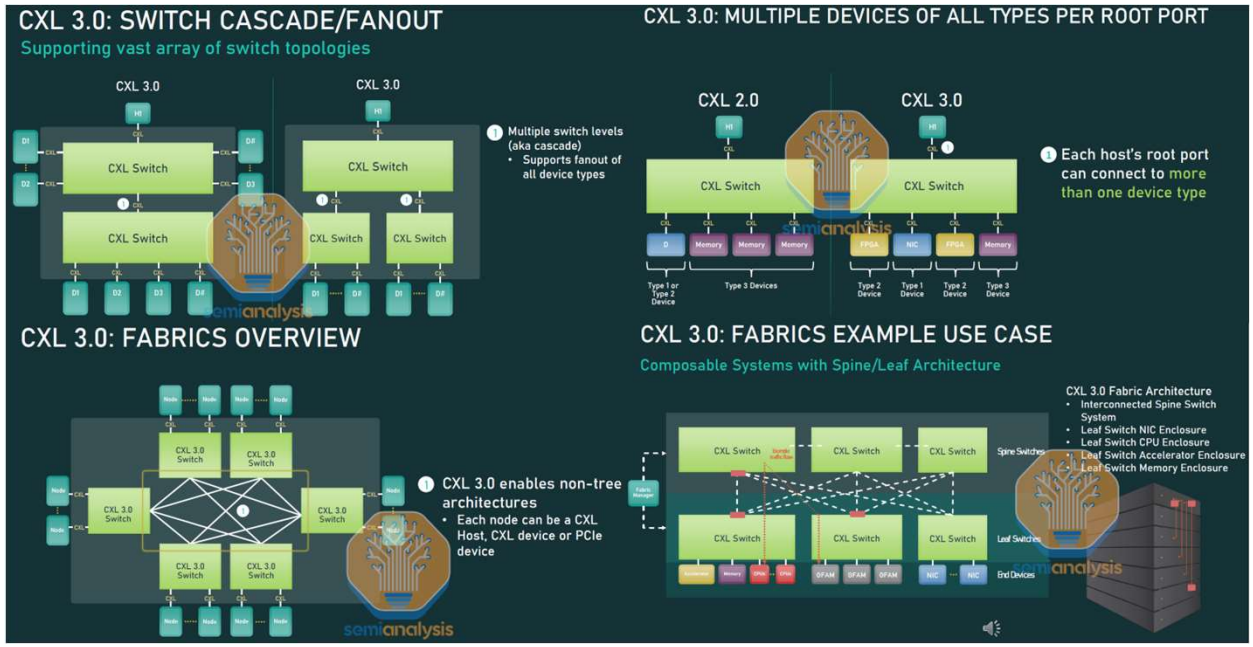


Fabric Wars

Proprietary fabrics emerged for resource sharing, however lack of standardization limited the audience

CXL Big Bang

Wide adoption of CXL allows for standardization and commoditization of expansion resources and sharing



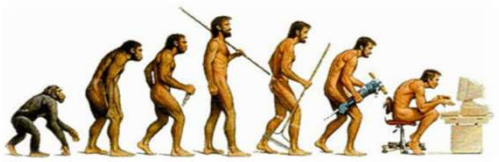
Nvidia's NVLink Vs. UALink

How NVIDIA's Hype United Tech Giants in the AI Arena

What impact will NVLink & UALink have on CXL?

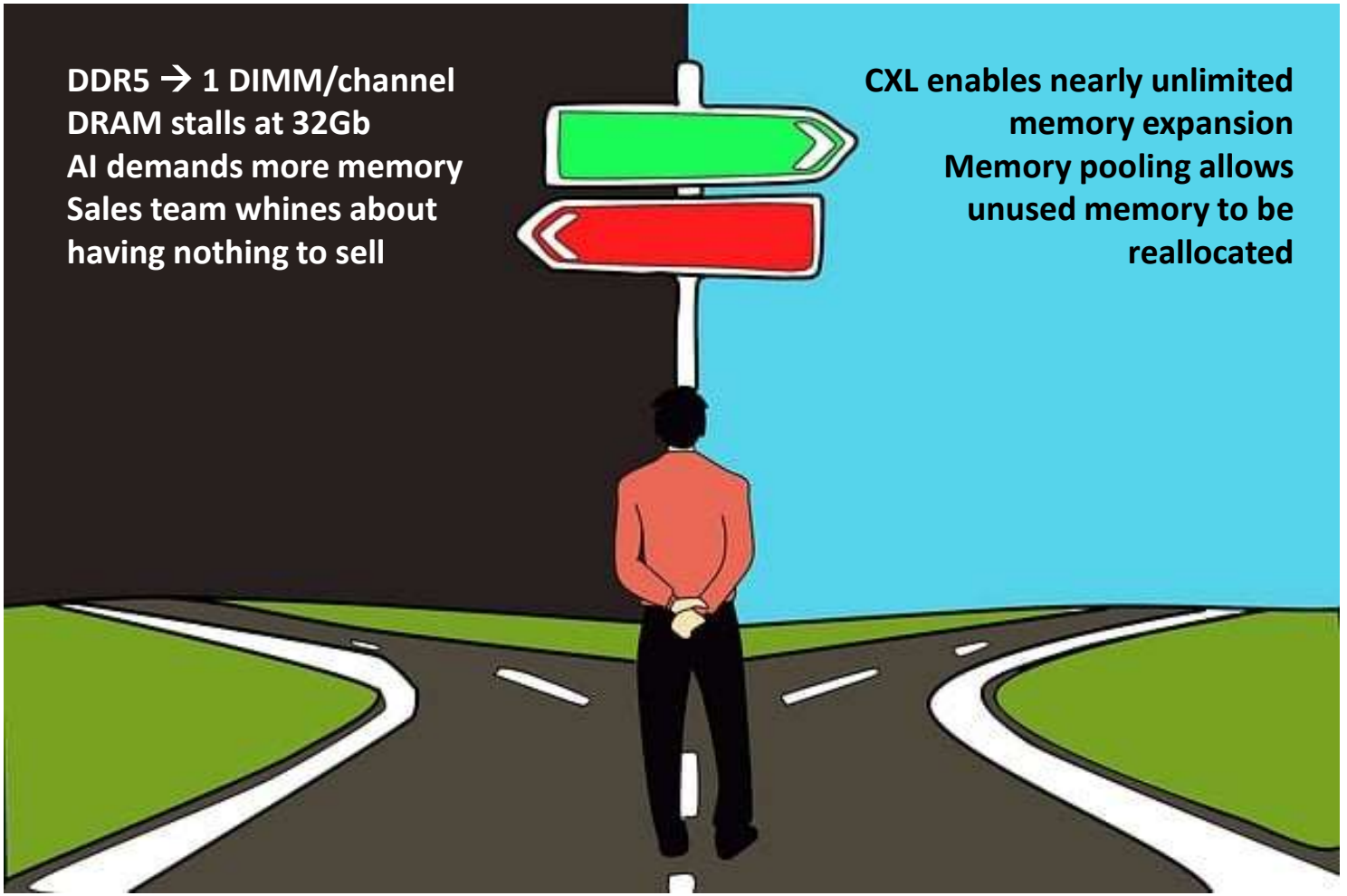
- These links are for xPU to xPU
- Not for memory expansion except NUMA
- CXL type 2 may go away
- CXL type 3 still needed

Evolution of CXL since introduction



Features	CXL 1.0 / 1.1	CXL 2.0	CXL 3.0	CXL 3.1
Release date	2019	2020	August 2022	November 2023
Max link rate	32GT/s	32GT/s	64GT/s	64GT/s
Flit 68 byte (up to 32 GT/s)	✓	✓	✓	✓
Type 1, Type 2 and Type 3 Devices	✓	✓	✓	✓
Memory Pooling w/ MLDs		✓	✓	✓
Global Persistent Flush		✓	✓	✓
CXL IDE		✓	✓	✓
Switching (Single-level)		✓	✓	✓
Switching (Multi-level)			✓	✓
Multiple Type 1/Type 2 devices per root port			✓	✓
Direct memory access for peer-to-peer			✓	✓
256-byte Flit (up to 64 GT/s PAM4)			✓	✓
256-byte Flit (Enhanced coherency)			✓	✓
256-byte Flit (Memory sharing)			✓	✓
256-byte Flit (Fabric capabilities)			✓	✓
Fabric Manager API definition for PBR Switch				✓
Host-to-Host communication with Global Integrated Memory (GIM) concept				✓
Trusted-Execution-Environment (TEE) Security Protocol				✓
Memory expander enhancements (up to 34-bit of meta data, RAS capability enhancements)				✓
			Not Supported	✓ Supported

Why Put DRAM on CXL?



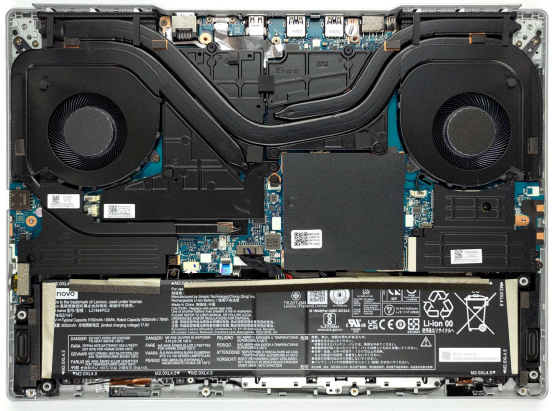
Not to be rude, but what choice do you really have?

That being said…

CXL Compute Express Link®

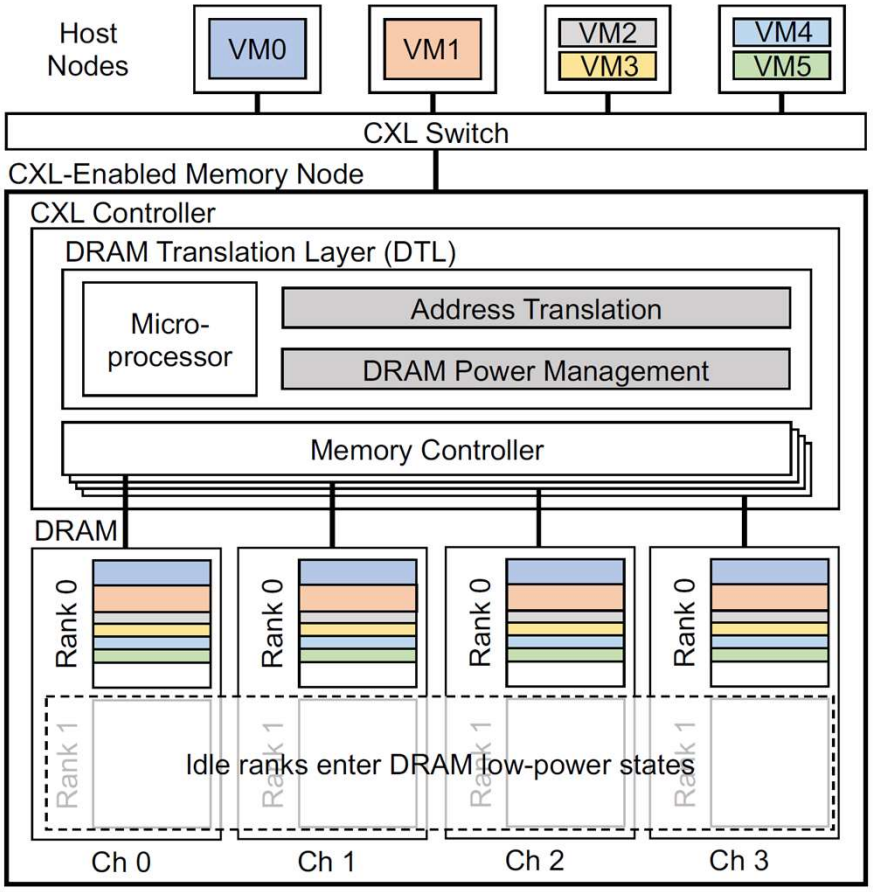
…needs to expand beyond the data center to bring the cost down…

…and stimulate innovation



FMS

WOLLEY



https://dl.acm.org/doi/abs/10.1145/3579371.3589051

CXL Unifies the Fabric

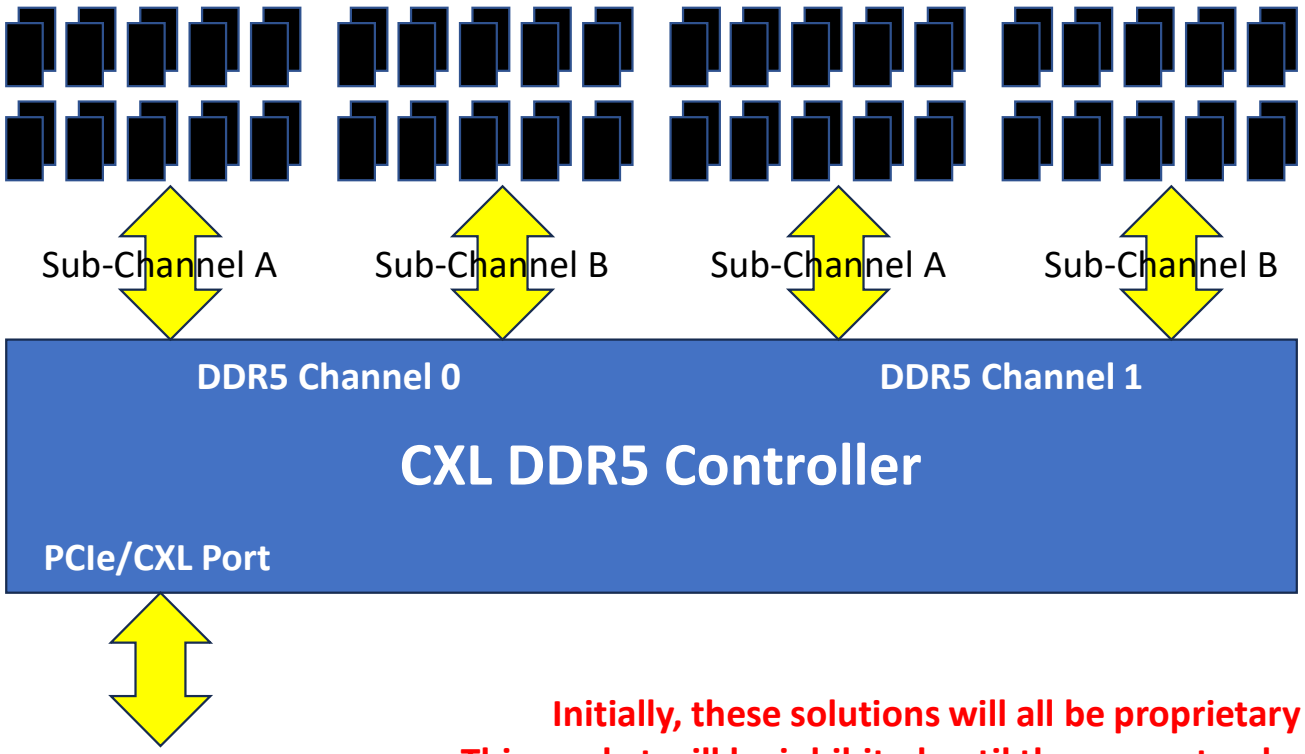
CXL is PCIe based and therefore inherits some of the features and limitations of a protocol that supports I/O or memory expansion

Legacy software only had filesystems to implement virtualization – DAX is assisting movement towards a unified addressing structure, but…

…is DAX stalled with the death of Optane?

…will CXL semantics breathe new life into a unified memory model?

Anatomy of a CXL to DRAM Bridge



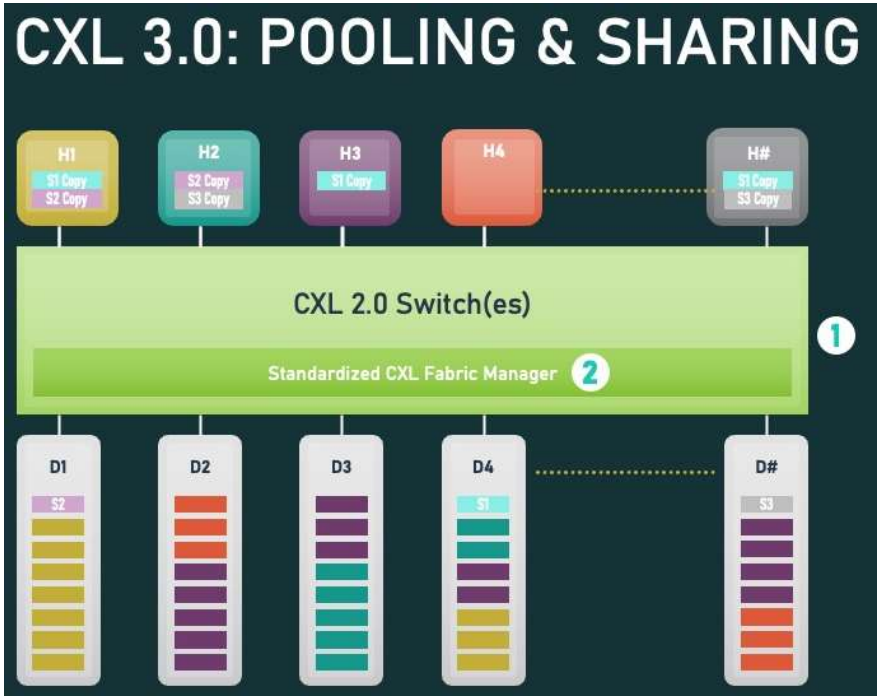
KISS: Just Do Writes and Reads

CXL is a non-deterministic protocol which allows the CXL module to operate independently

- Refresh
- Error check scrub
- Post-package repair

CXL 3+ incorporates some additional functions such as coherency

Initially, these solutions will all be proprietary
This market will be inhibited until these are standardized
Plug and play memory on CXL will be a hard requirement



It's a Brave New World with CXL Memory

CXL memory modules may be dedicated to a single processor

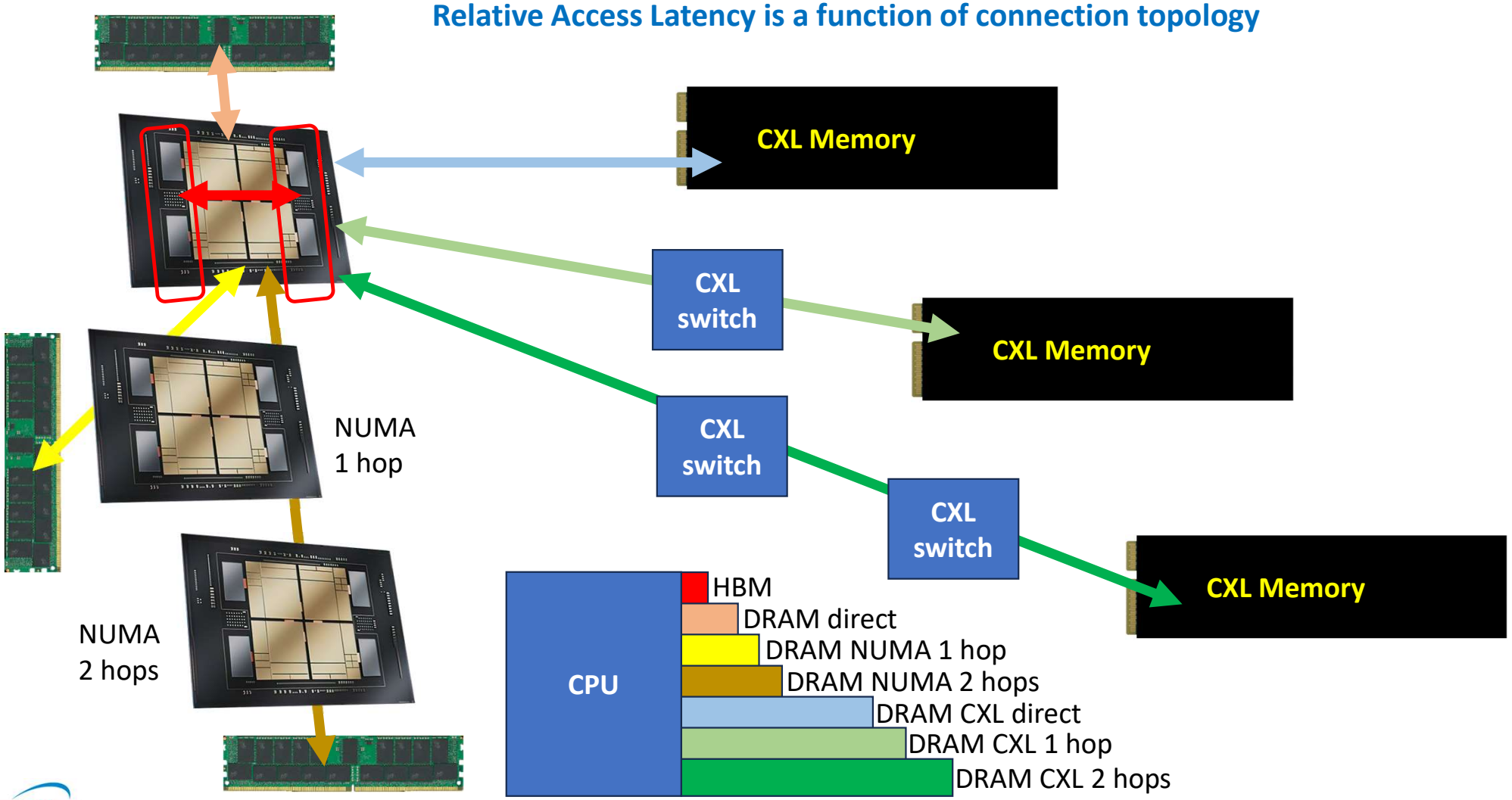
CXL memory modules may be allocated in chunks to different processors

CXL memory modules may be shared by multiple processors

Randomness of accesses made worse by pooling

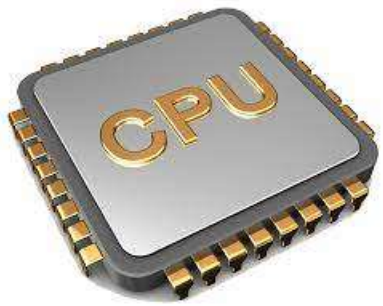
Matrix of CPUs X Cores/CPU will make access randomness the norm

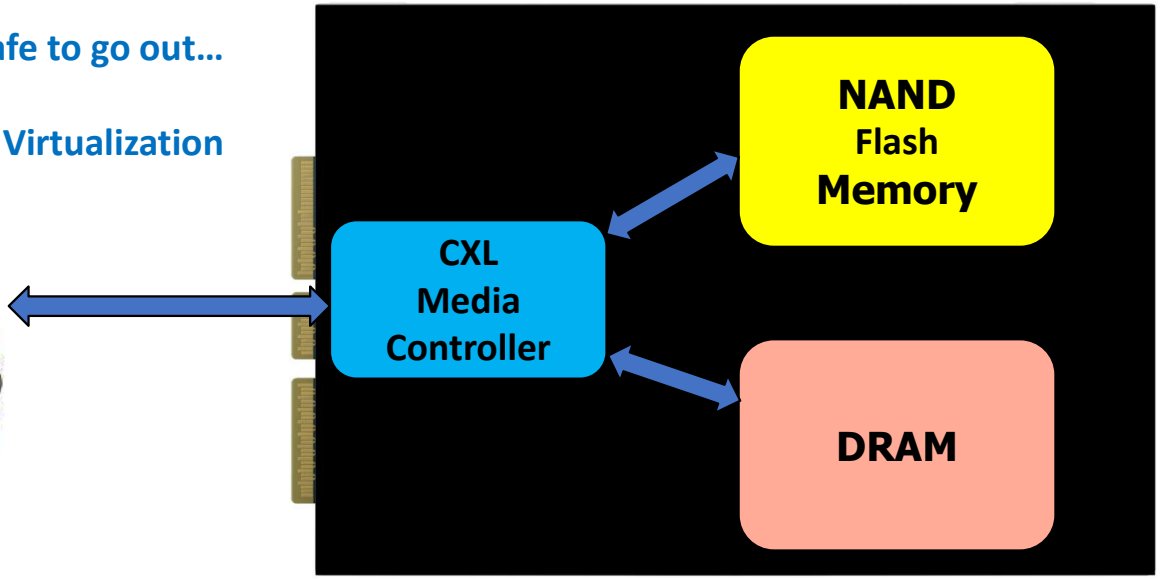
Relative Access Latency is a function of connection topology



Just when you thought it was safe to go out…

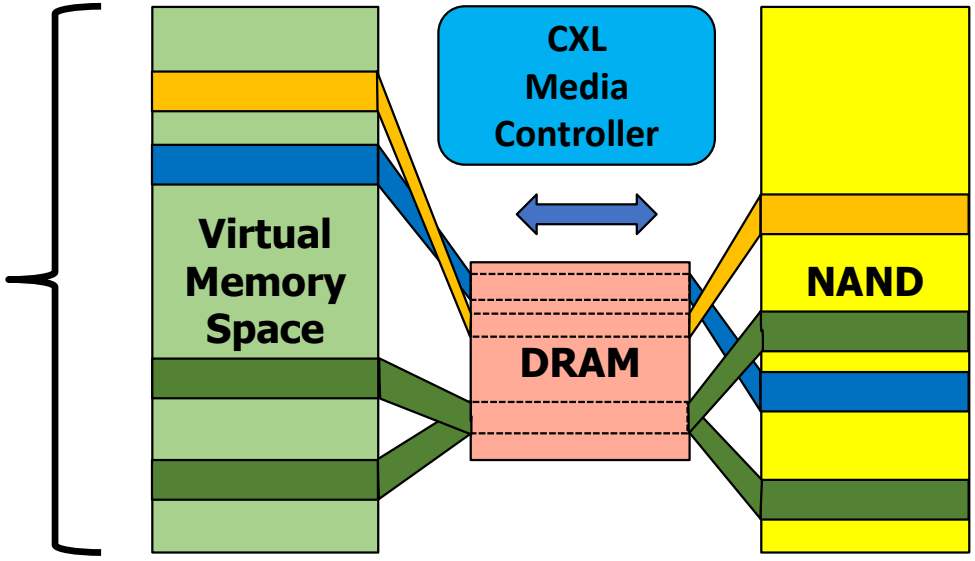
Hybrid Memory: A Virtualization





CPU sees a hybrid DRAM + NAND module as a linear RAM with the larger capacity of the NAND

As the DRAM resource is depleted, can be flushed and another block can replace it

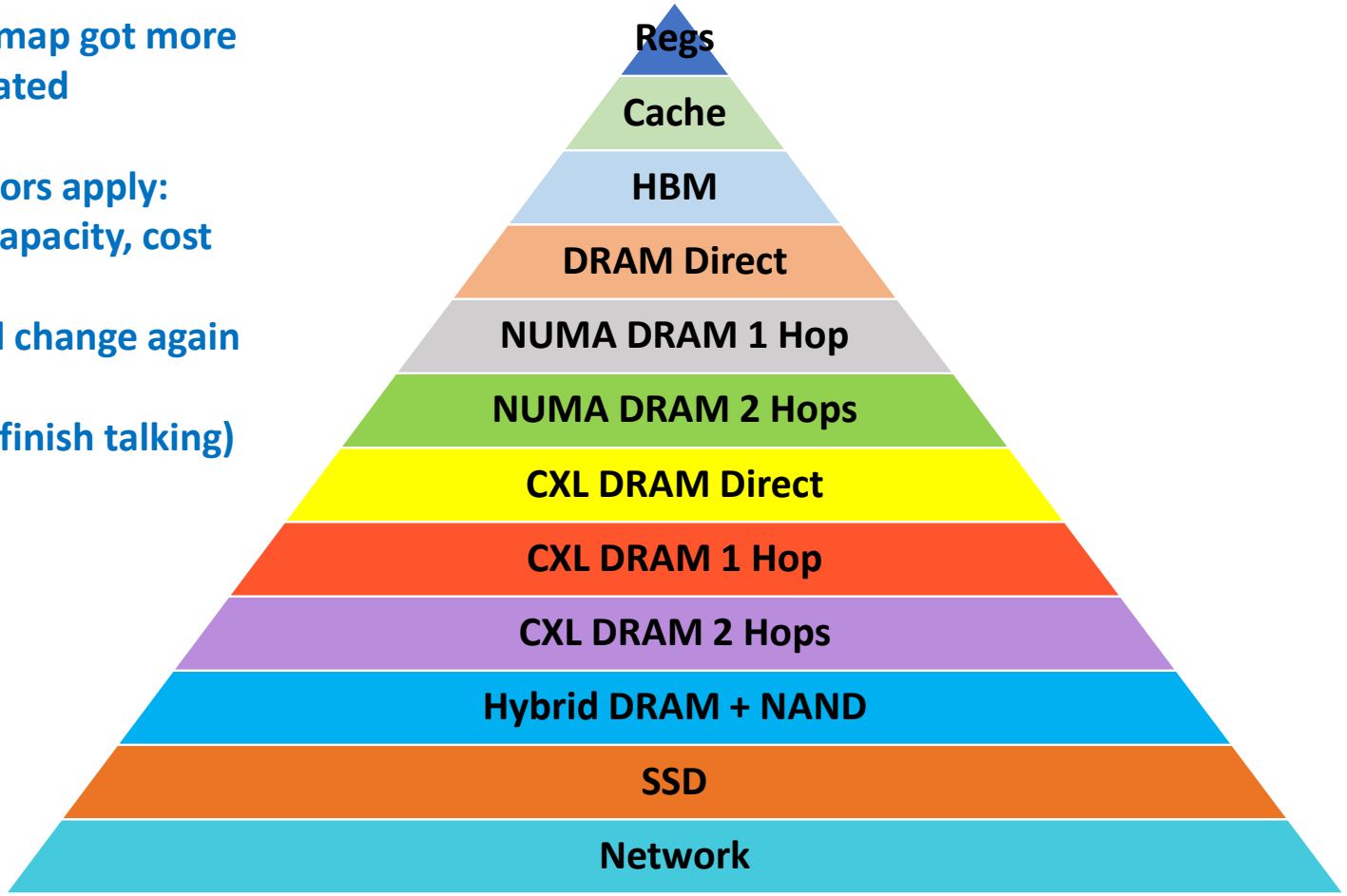


The resource tier map got more complicated

The same factors apply:
speed, latency, capacity, cost

Don’t blink. It will change again

(Possibly before I finish talking)

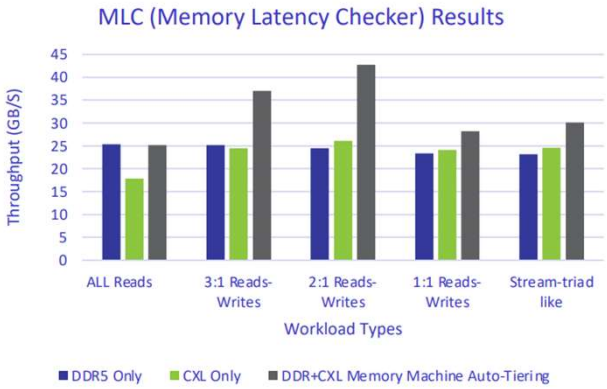


Latency Aware Software

Drivers, e.g., Memory Latency Checker

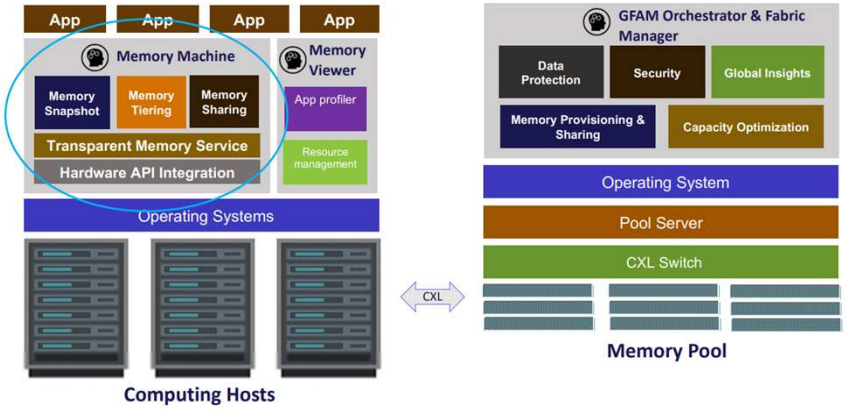
Operating systems measure the access latency of the various memory regions, categorize them

```
Measuring idle latencies (in ns)...
          Memory node
Socket         0         1
     0      67.5     125.2
     1     126.5      68.5
```



Hypervisors, e.g., MemVerge

Runtime monitoring of system resource utilization and characterization of hot/warm/cold data



Operating System Support

Linux kernel support memory hotplug & hotremove today

Need Dynamic Capacity Driver in Linux kernel

Policy should be implemented in userspace

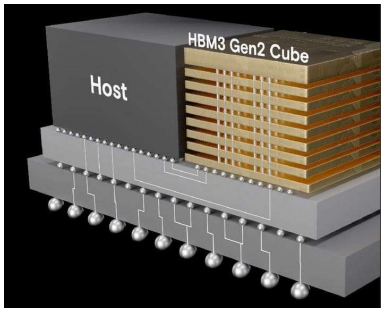
- When to request memory (hotplug)
- When to release memory (hotremove)

OS improvement to make hotplug & hotremove faster (keep region map, …)

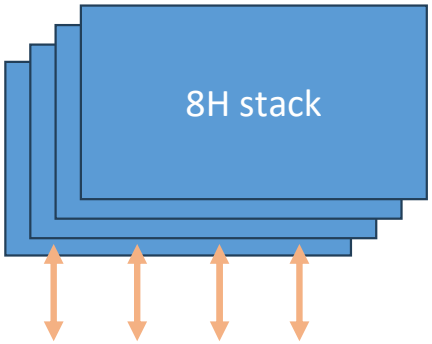
OS improvement to avoid memory pinning (which block hotremove)

- Linux kernel already have some of that (zone movable comes with tradeoff)

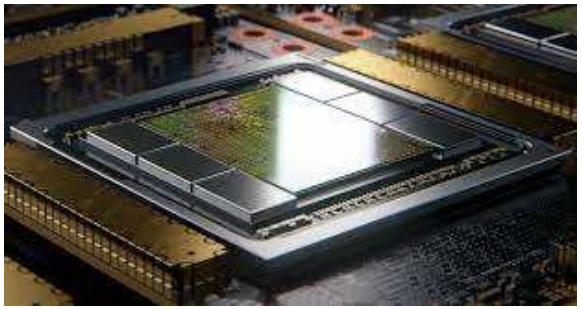
Focus Application: Artificial Intelligence

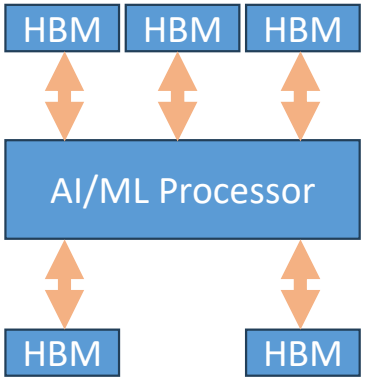


16Gb-based HBM stack

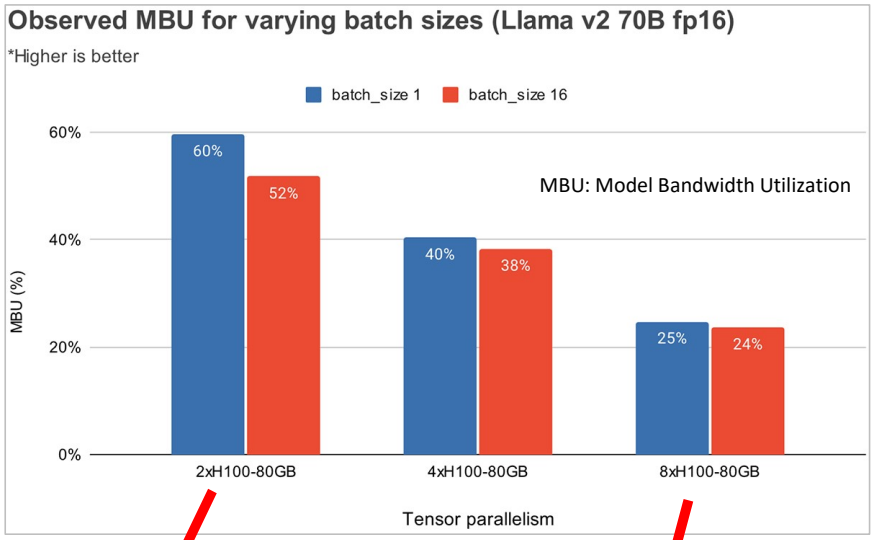


16 GB capacity
666 GB/s
1920 signals





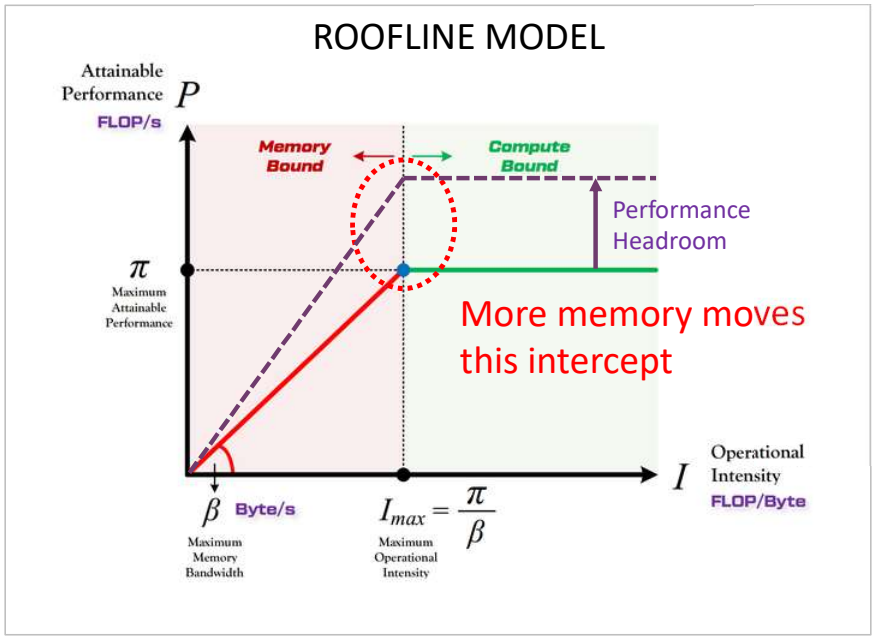
80 GB total capacity
3.35 TB/s
9600 signals

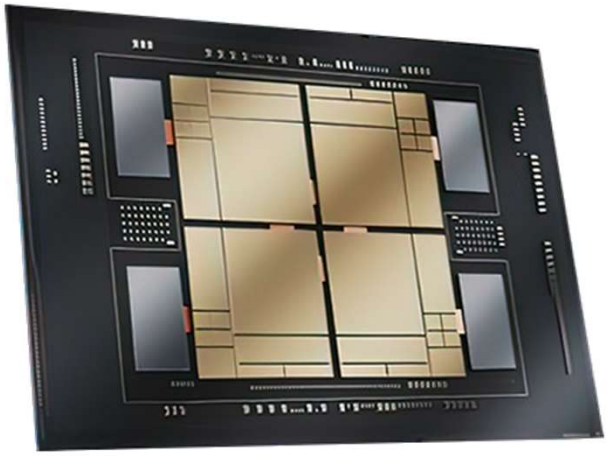


70B parameter with 16-bit precision → time per output token ≈ 35ms

Adding bandwidth helps performance BUT at the cost and complexity of more H100s

How does AI deal with memory requirements?





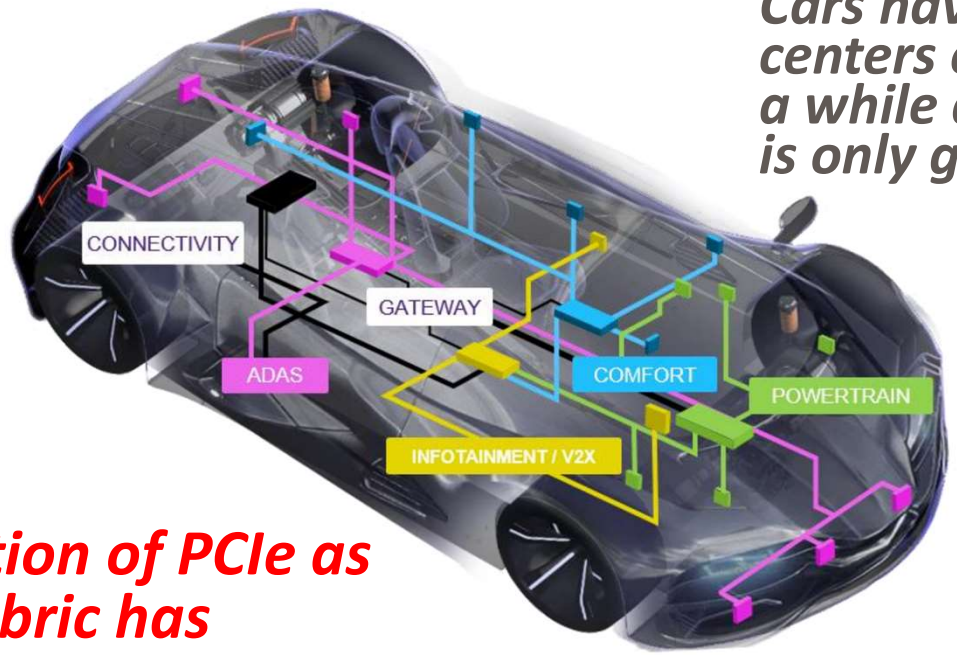
+

+

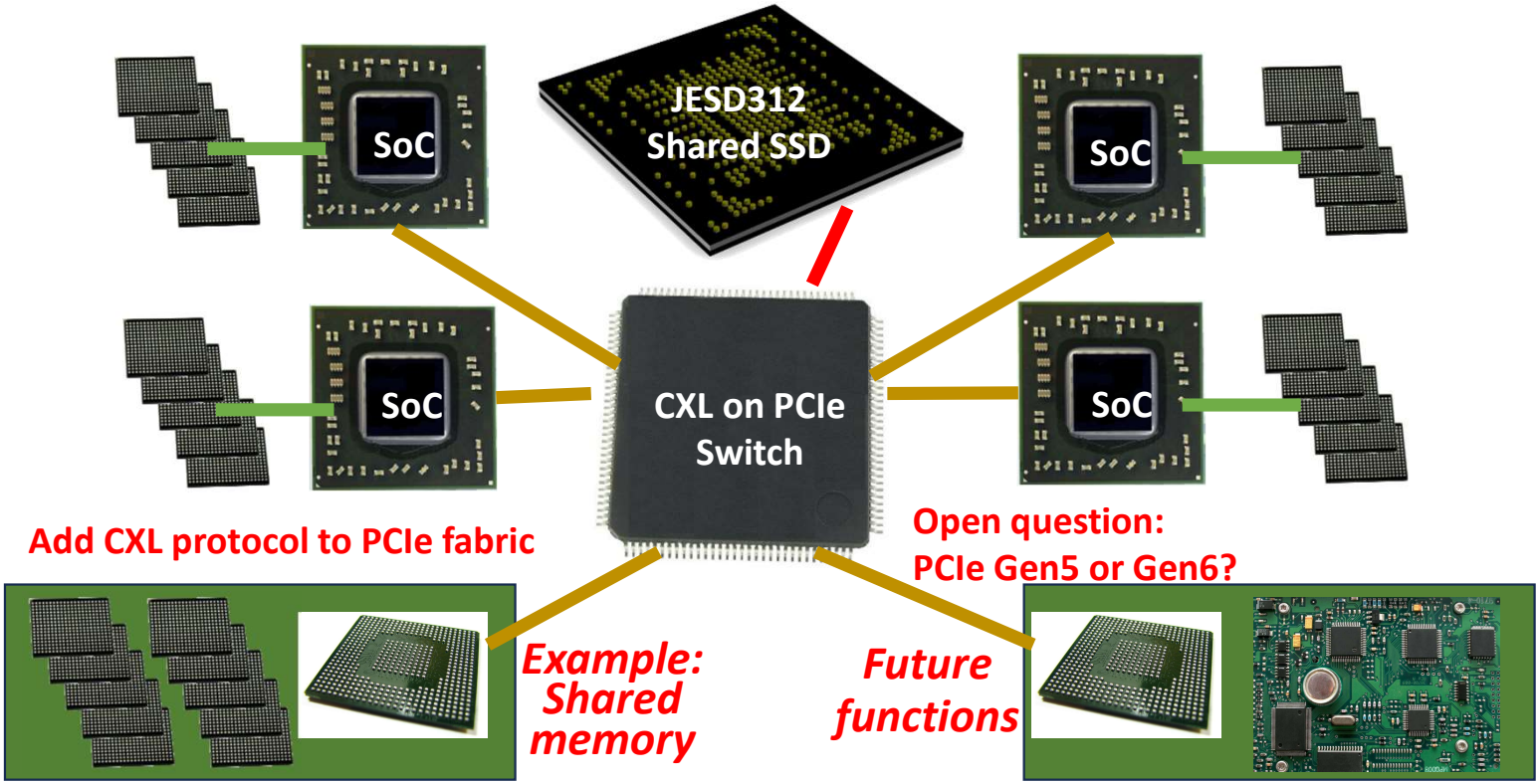
The industry is going through phenomenal growth in AI

Large Language Models grow from 80GB to 240GB to 1.8TB
No end in sight to the hunger for more memory
Tiered memory allows expansion to allow for this growth

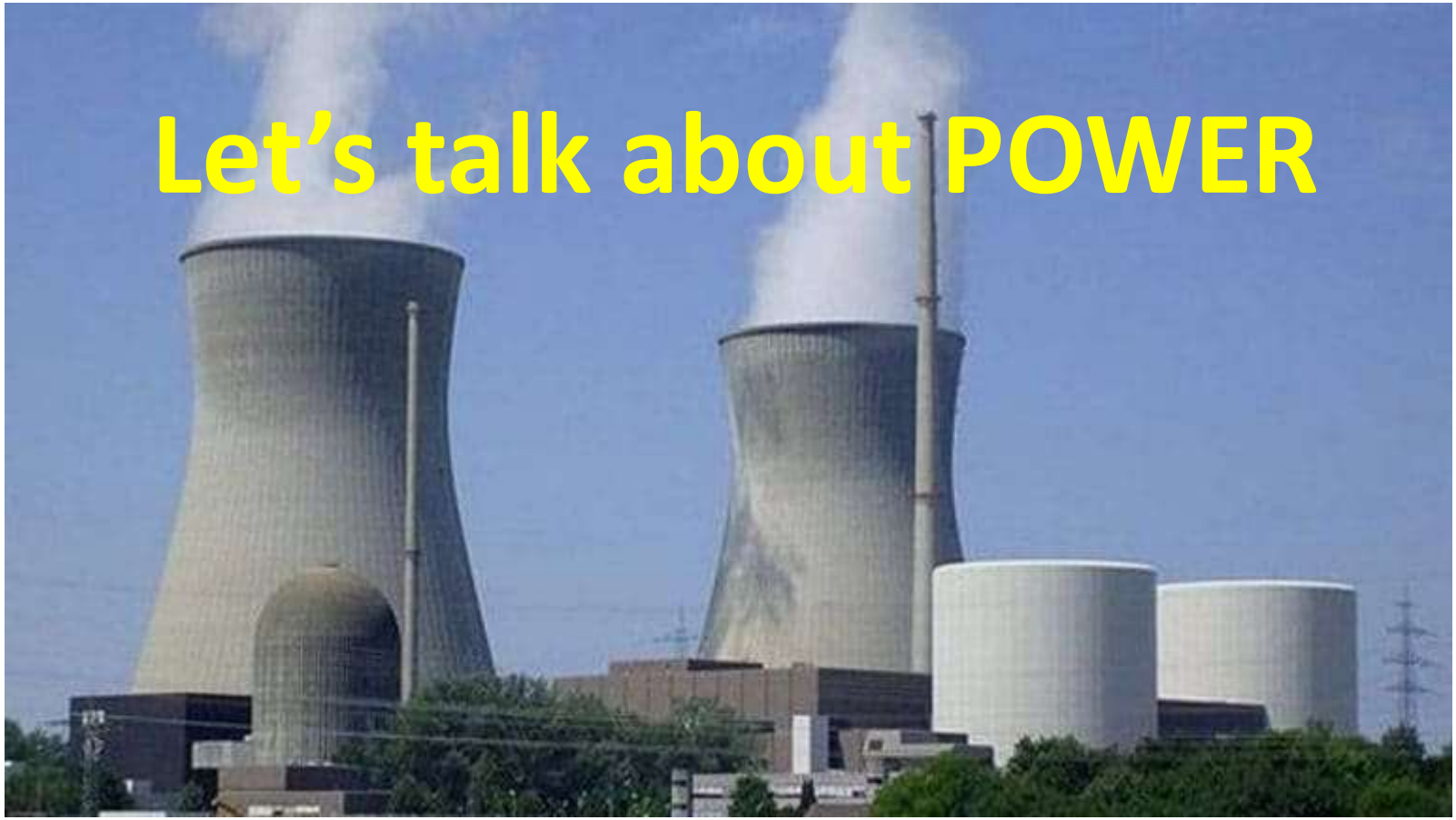
Cars have been data centers on wheels for a while and this trend is only growing



Adoption of PCIe as the fabric has already begun



Projected growth: CXL for Automotive
Solves many of the same needs as data center
Allows innovative new features
Allows for rapid growth of AI features


Let’s talk about POWER

Today's data centers are highly optimized, finely tuned and waste almost no power

Psych!

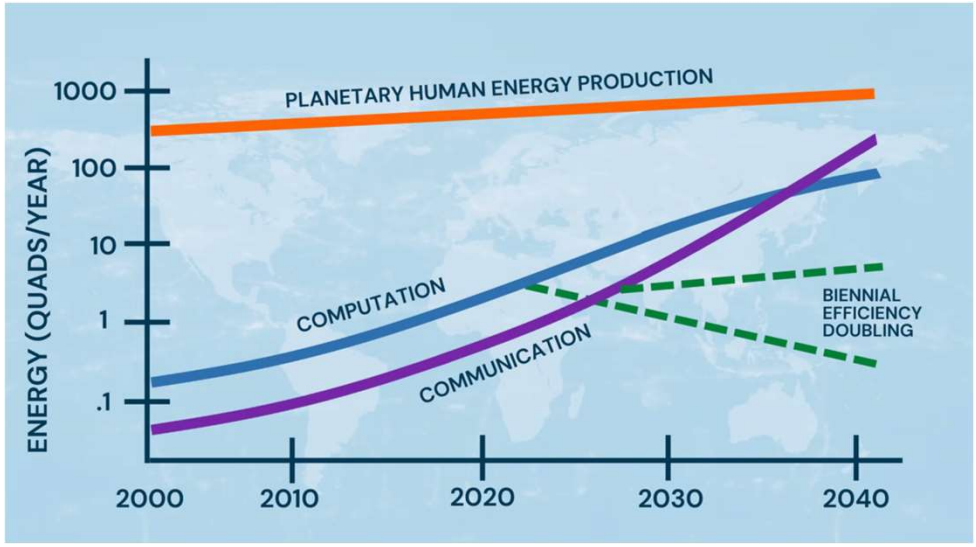
Data centers use <u>nearly no data</u> moved around

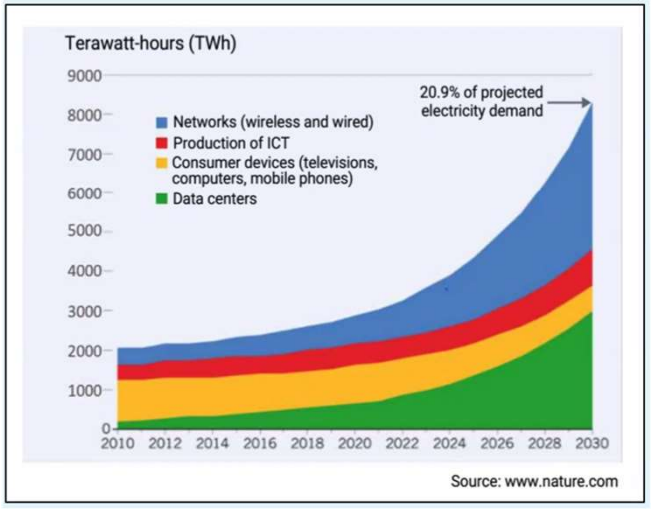
Generously estimated as 0.00004%

EES2
ENERGY
EFFICIENCY
SCALING

Designing for energy efficiency is a growing concern

"Total Cost of Ownership" partially encompasses this





Operation	Energy per bit
Wireless data	10 – 30μJ
Internet: access	40 – 80nJ
Internet: routing	20nJ
Internet: optical WDM links	3nJ
Reading DRAM	5pJ
Communicating off chip	1 – 20 pJ
Data link multiplexing and timing circuits	~ 2 pJ
Communicating across chip	600 fJ
Floating point operation	100fJ
Energy in DRAM cell	10fJ
Switching CMOS gate	~50aJ – 3fJ
1 electron at 1V, or 1 photon @1eV	0.16aJ (160zJ)

most energy is used for communications, not logic

On the current trajectory of energy use versus energy production,

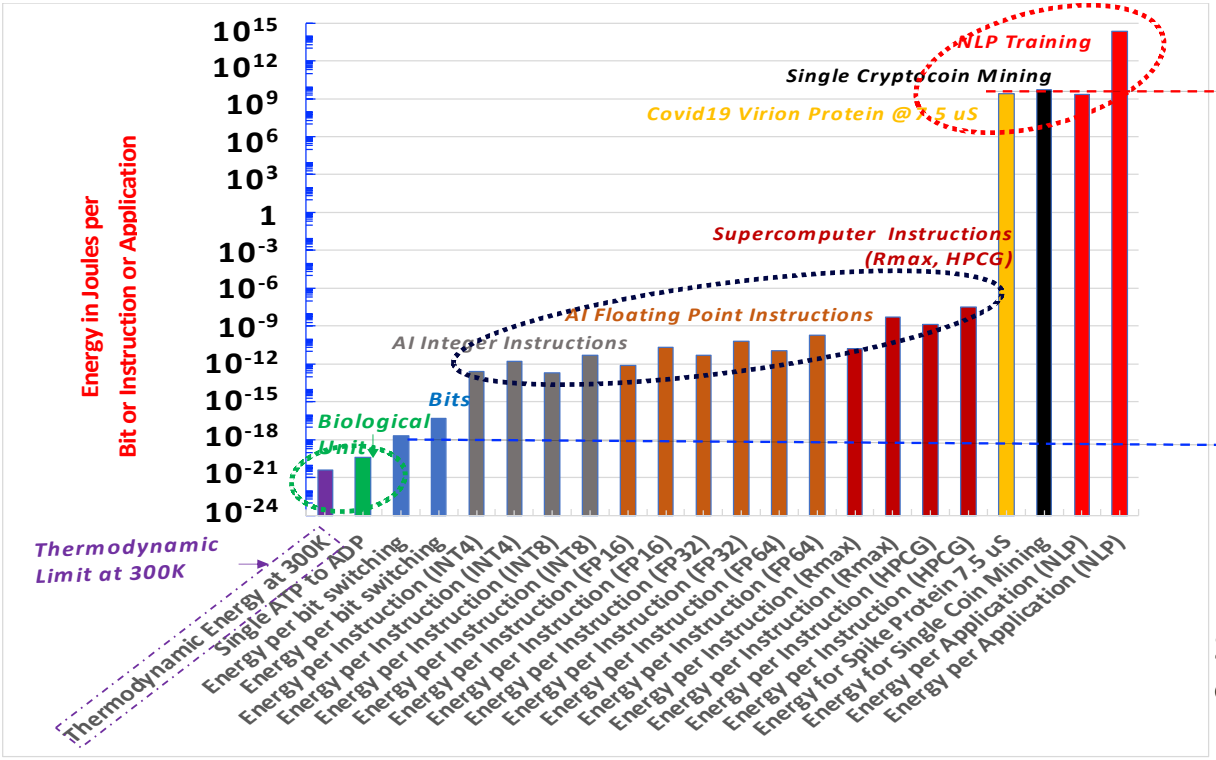
THESE CROSS OVER IN 2055

EES2 program goal is 1000X improvement in energy efficiency over the next 20 years

This program is not US-centric All countries are invited to participate

This program is tied into the US CHIPS Act funding

Where are we wasting power and what can we do about it?



10^{9} to 10^{15}

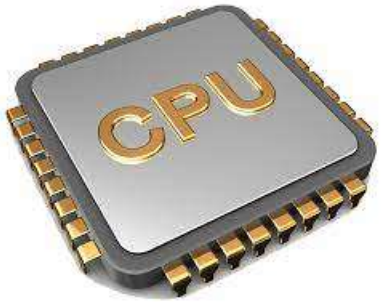
10^{-18} to 10^{-21}

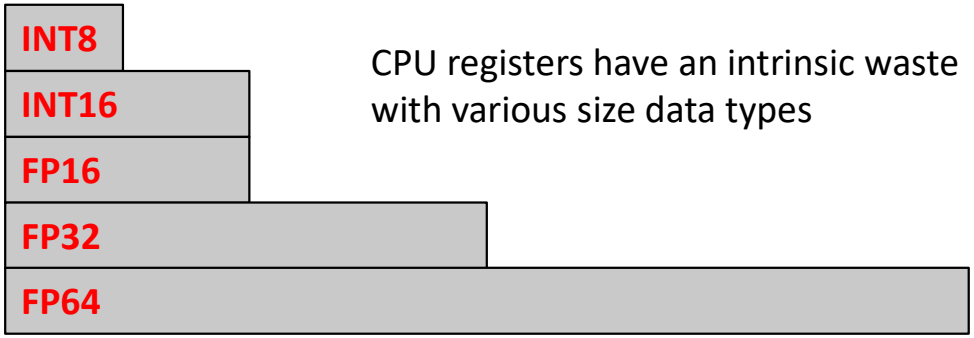
27 to 36 orders of magnitude
(From Algorithms to Atoms)

Part of the looming energy crisis is fundamental inefficiencies of applications and programming languages

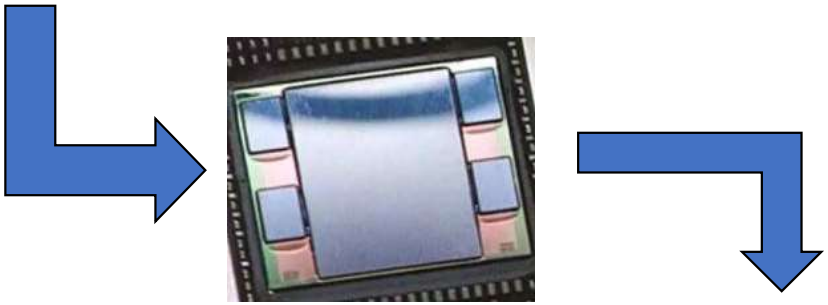
Python programming is many orders of magnitude less energy efficient than C programming (ChatGPT is Python-based)

Cryptocurrency in particular consumes ≥0.8% of world energy resources already





CPU registers have an intrinsic waste with various size data types



CPUs caches recently accessed data

Industry standard is 64 bytes per cache line

64 Byte Cache Line

If an application needs a yes or no answer (1 bit)

But accesses a cache line (64 bytes)

Waste = 99.8%

FMS

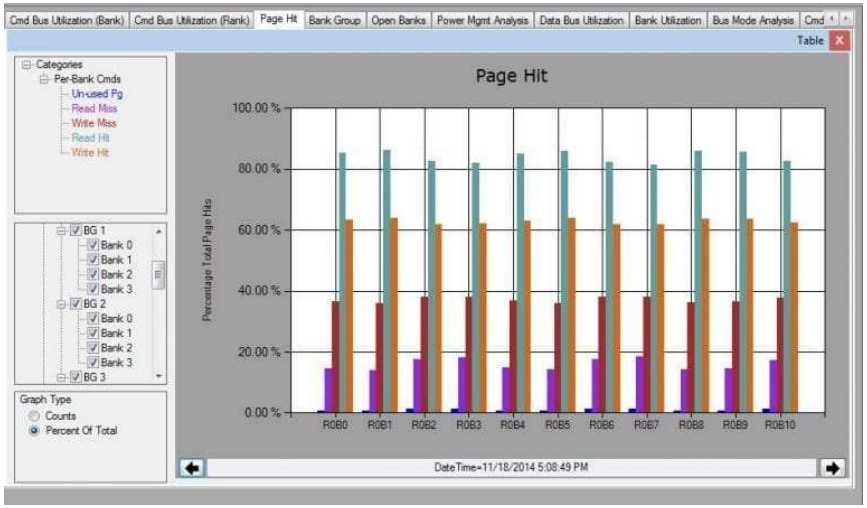
VOLLEY

L1: 96% hit rate, 1 cycle access
L2: 95% hit rate, 25 cycles access
L3: 98% hit rate, 80 cycles access

The good news: near-CPU caches do have high hit rates (reduces waste from unnecessary accesses)

By the time an access gets to the local DRAM, though, hit rates start to drop dramatically

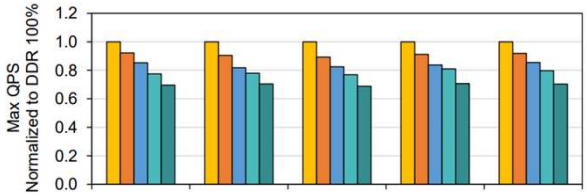
Read hit ~82%
Write hit ~62%



A question I have posed that CPU guys refuse to answer:

How much performance gain are we getting for each watt expended?

ESPECIALLY when it comes to speculative operations



Access to remote memory drops even further, especially with increased thread count

Hit rate ~65%

…and this is before memory pooling…

https://www.futureplus.com/blog/critical-memory-performance-metrics-for-ddr4-systems-page-hit-analysis

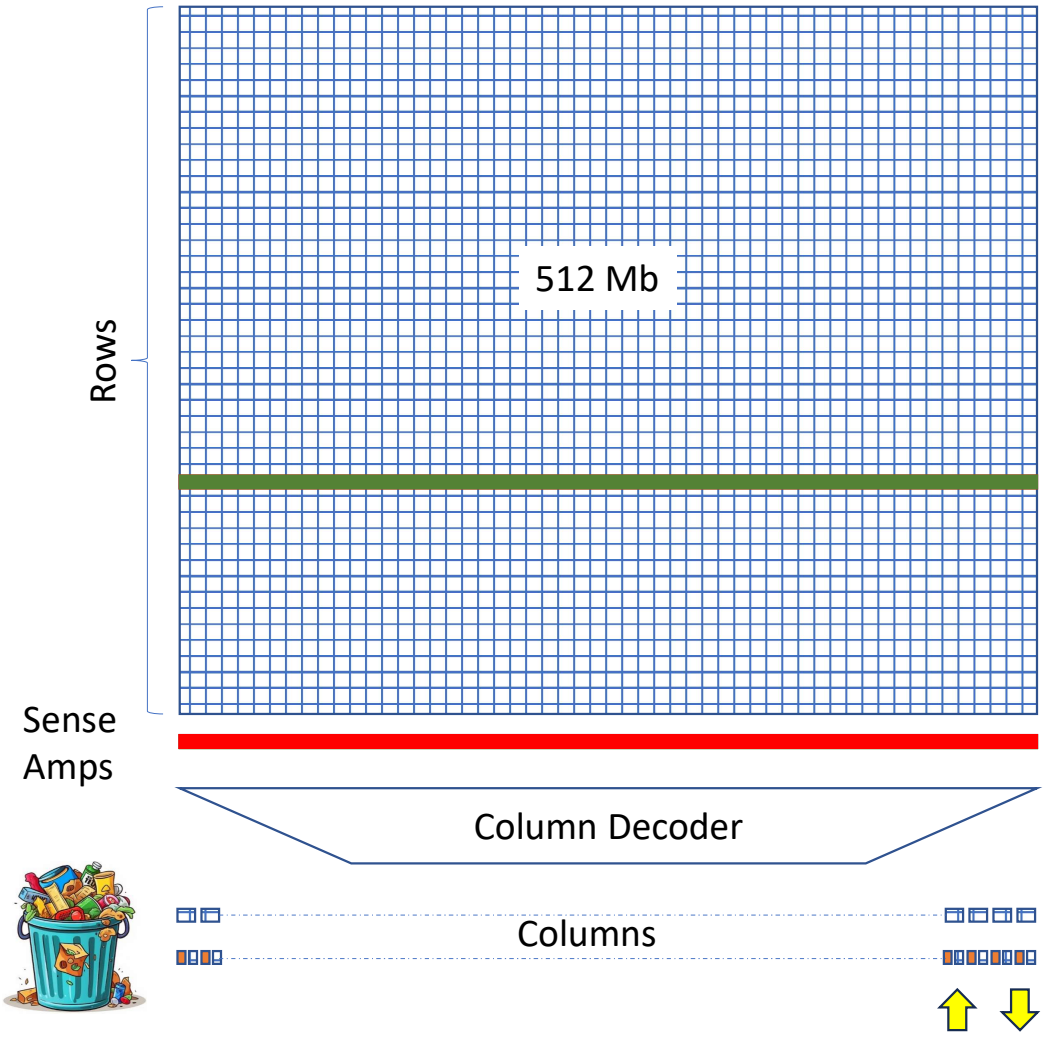
https://arxiv.org/pdf/2303.15375#:~:text=Meanwhile%2C%20as%20the%20block%20size%20increases%20beyond,latency%20begins%20to%20dominate%20the%20p99%20latency.

Power	Definition	DDR4 mA	Norm
IDDO	Active precharge	31	1.9
IDD1	Active read precharge	44	2.8
IDD2P	Precharge power-down	16	1.0
IDD3P	Active power-down	21	1.3
IDD2N	Precharge standby	22	1.4
IDD3N	Active standby	36	2.3
IDD4R	Read current	101	6.3
IDD4W	Write current	84	5.3
IDD5	Refresh	199	12.4
IDD6	Self-refresh	23	1.4
IDD7	Bank interleave read	142	8.9

Where are we spending our power?

Some simplified looks:

Refresh burns >10X idle power
Activate uses 11%
Precharge uses 21%



DRAM access procedure:

ACTIVATE reads 8192 from core to sense amps, destroying the contents of the core bits

READ operations transfer 128 bits (x8) or 64 bits (x4) from sense amps to the I/O

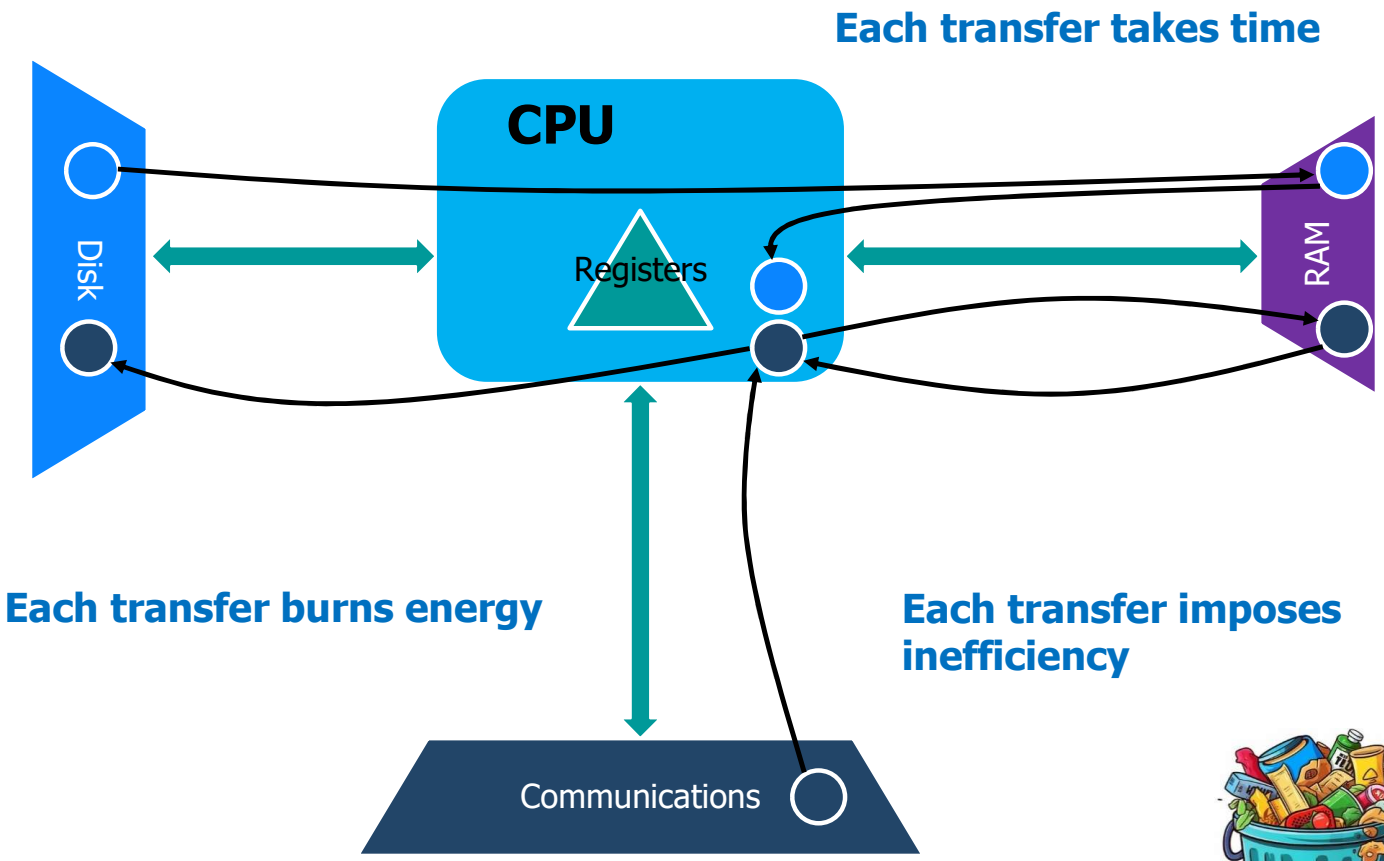
Write operations transfer 128 or 64 bits from I/O to sense amps

PRECHARGE rewrites 8192 bits back to the core

	%Of array	%Of buffer	%Of array	
8192 b	1.5%			
128 b		1.5%	0.025%	All bits used for x8 DRAMs
64 b		0.8%	0.012%	ECC half-word needed for x4

Conclusion: Open Page Mode access is grossly inefficient

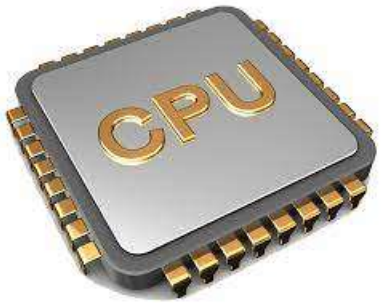
Simplified but realistic case of program execution and data movement



Typical application flow

1. App read from disk through CPU to RAM
2. App read from RAM to CPU for execution
3. Info read from I/O through CPU and written to RAM
4. App reads RAM to process
5. App writes results to disk

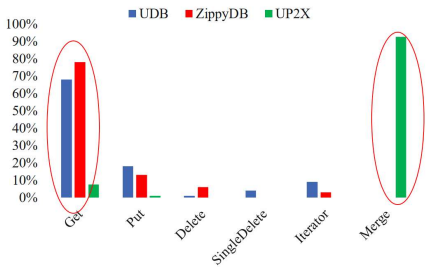
4KB block



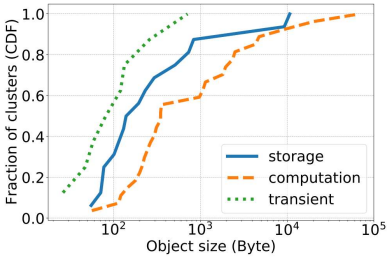
Disk/SSD access

64B cache line

Facebook RocksDB



X (Twitter) Twemcache



The average key size (AVG-K), the standard deviation of key size (SD-K), the average value size (AVG-V), and the standard deviation of value size (SD-V) of UDB, ZippyDB, and UP2X (in bytes)

	AVG-K	SD-K	AVG-V	SD-V
UDB	27.1	2.6	126.7	22.1
ZippyDB	47.9	3.7	42.9	26.1
UP2X	10.45	1.4	46.8	11.6

Typical disk block transfer size is 4KB

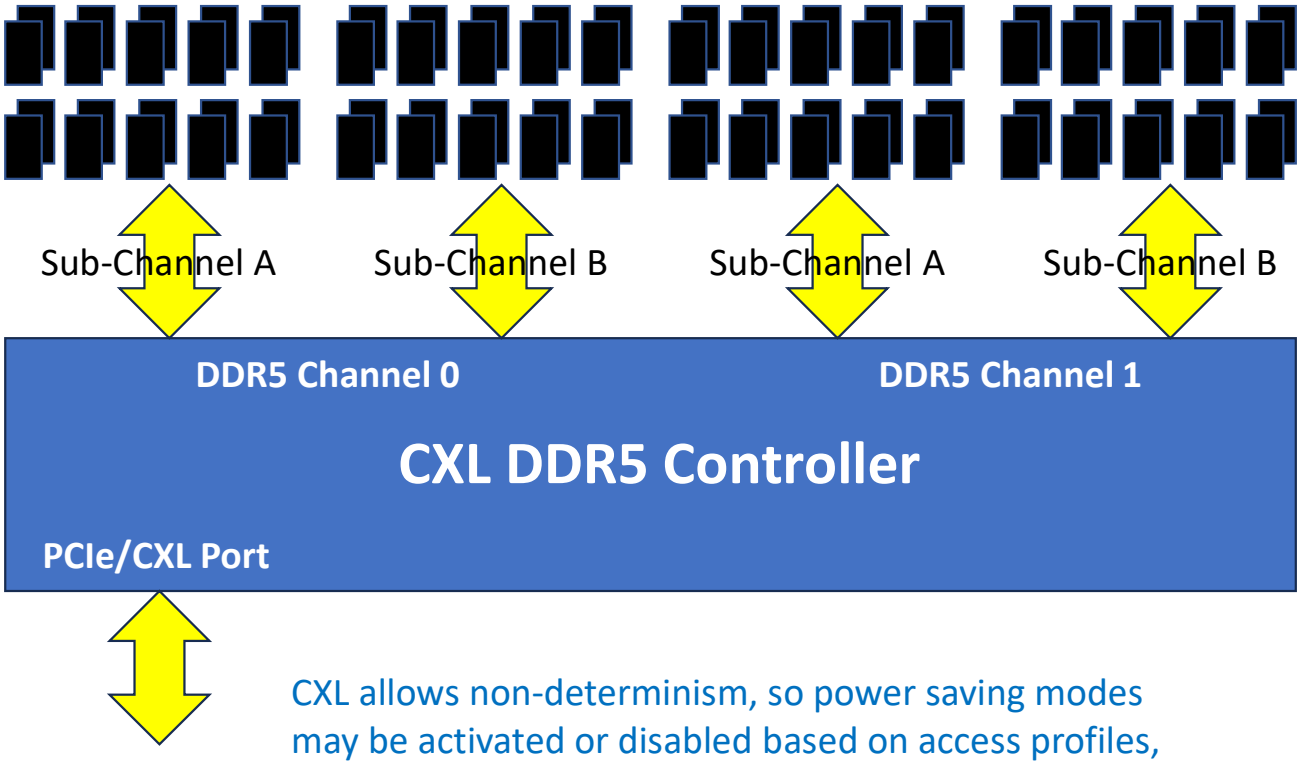
Average number of bytes actually used is 100

Waste = 97.5%*

* More if remote memory is used

FMS

VOLLEY



CXL allows non-determinism, so power saving modes may be activated or disabled based on access profiles, user configuration settings, etc.

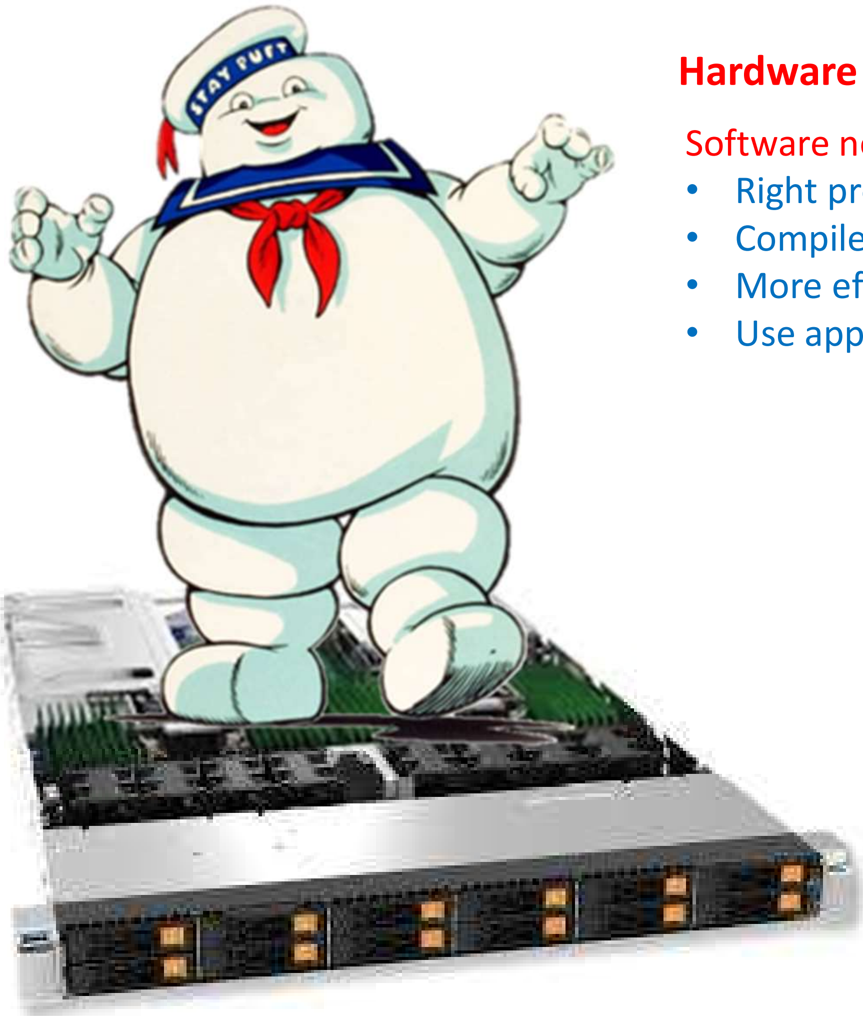
Mode switching latency penalty need only be taken once – what's a microsecond when a region has not been accessed for an hour?

Optimizing DRAM power

Use closed page mode to avoid active standby power penalty

Use CKE & self-refresh for memory regions not used often

Use Maximum Power Saving Mode for DRAM not yet allocated



Hardware can't be the only solution to optimizing power

Software needs to be part of the solution:

- Right programming language for the problem
- Compilers, not interpreters
- More efficient access mechanisms, e.g., DAX
- Use appropriate data types: not every variable needs to be FP64

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \bullet \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \bullet \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \bullet \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \bullet \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

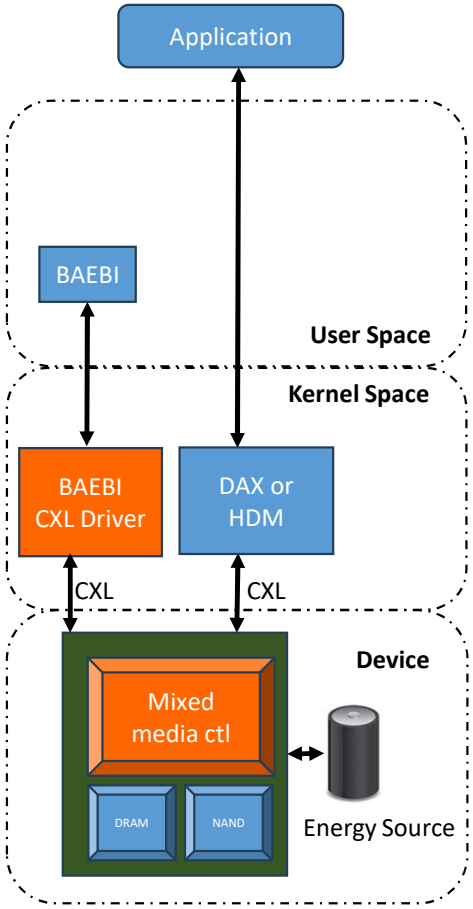
Avoid unnecessary variables in matrix calculations
The effects on performance can be exponentially bad

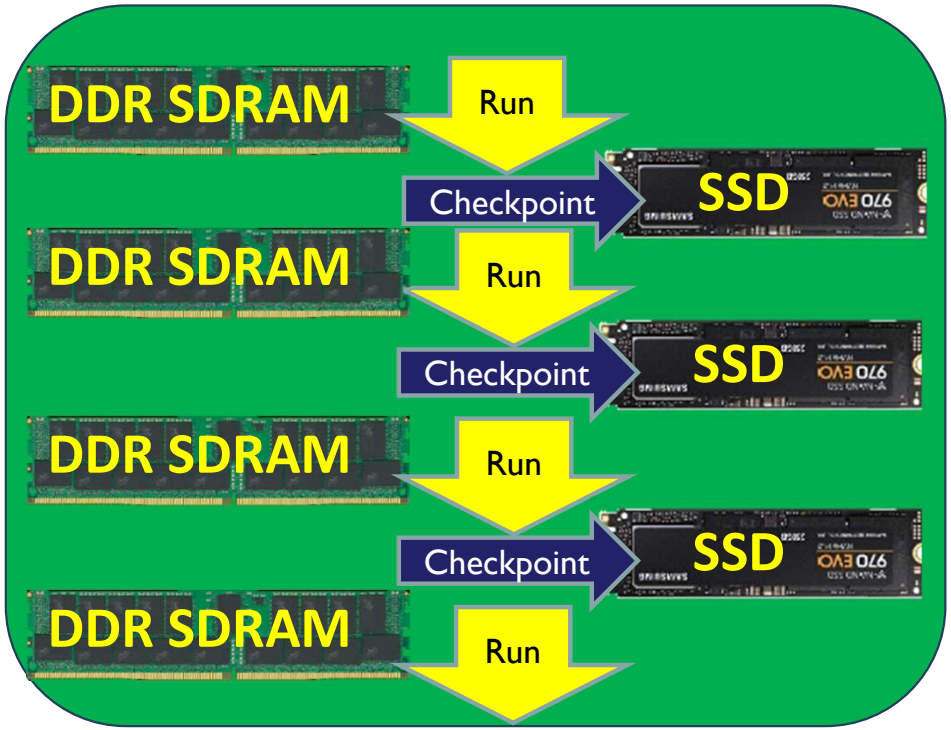
-1	A lot of rows and
0	columns are one of
+1	three values

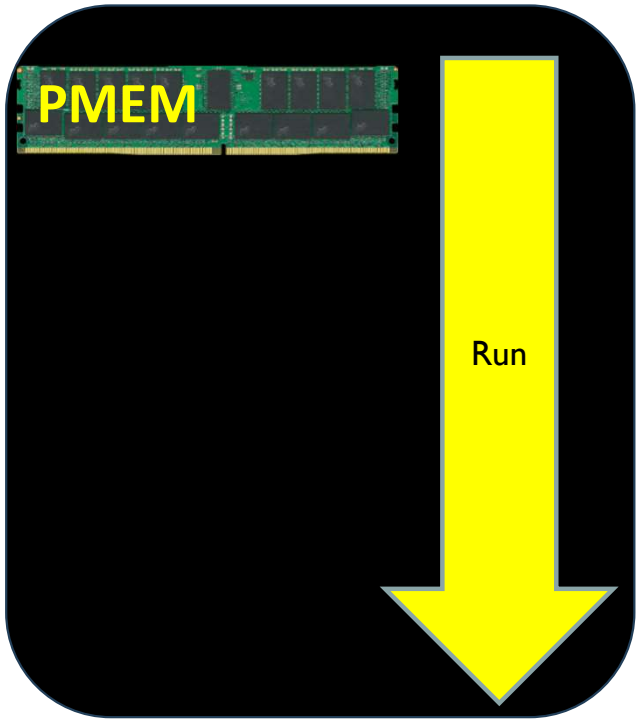
Consider memory compression to reduce the overhead

Persistent memory is not just about data integrity

Applications are forced to checkpoint contents periodically because of volatile DRAM



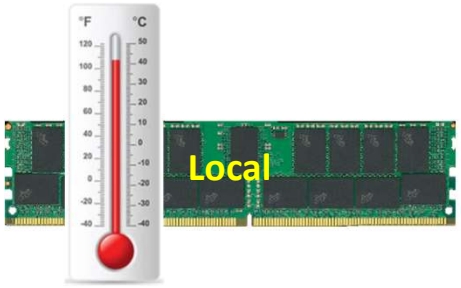


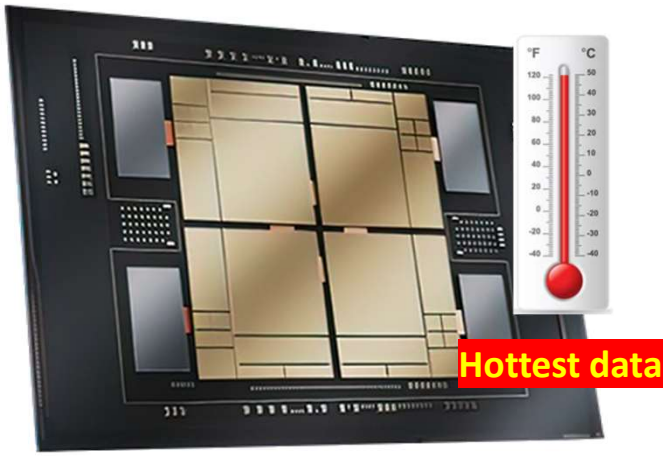


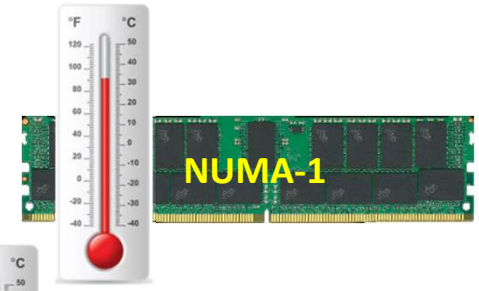
Checkpointing consumes ~8% of system throughput and power on average

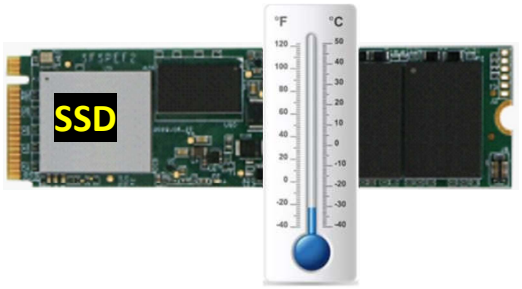
Consider the **temperature** of your data

Coldest data

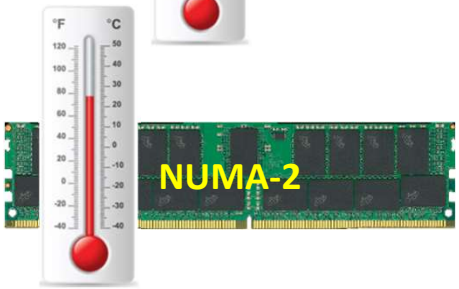


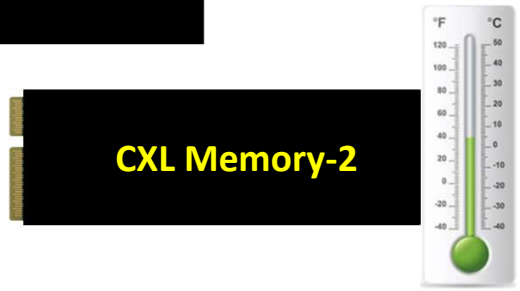






Map data into the appropriate memory tier by its temperature rating





Half of data center power is in the electronics

Half is in the cooling

Any improvements made in managing power is effectively doubled by reducing cooling requirements

Summary

DRAM evolution is slowing down

Demand for more memory is accelerating

CXL resolves the long standing fabric war

DRAM design is an evolution of 1990s SDRAM

NUMA forces software to be more latency aware

Solutions need to engage hardware and software

CXL enables new ways of virtualizing resources

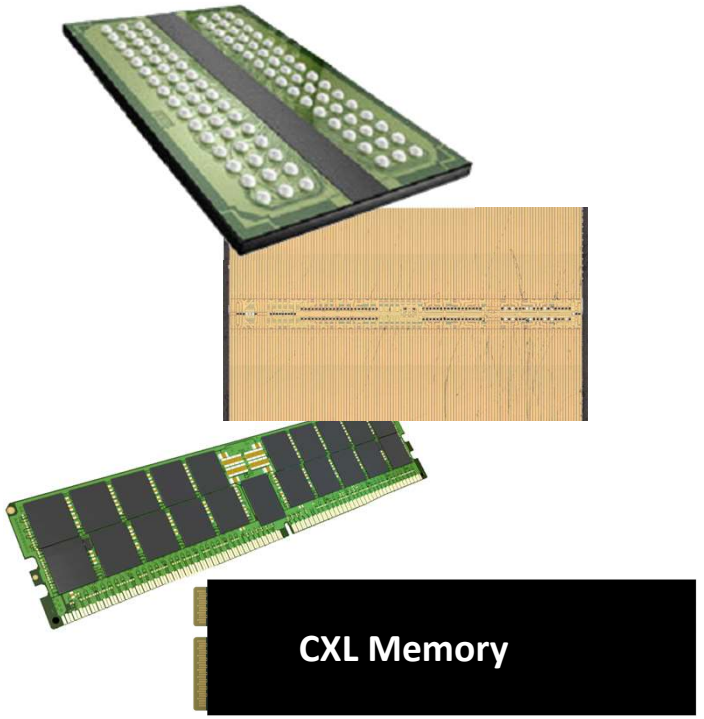
Data centers suck at actually using data

Earth is approaching a power crisis

AI and Automotive likely CXL adopters

Thank you for your time

Any more questions?



Bill Gervasi, Principal Systems Architect
Wolley Inc.
bilge@wolleytech.com