arm Research

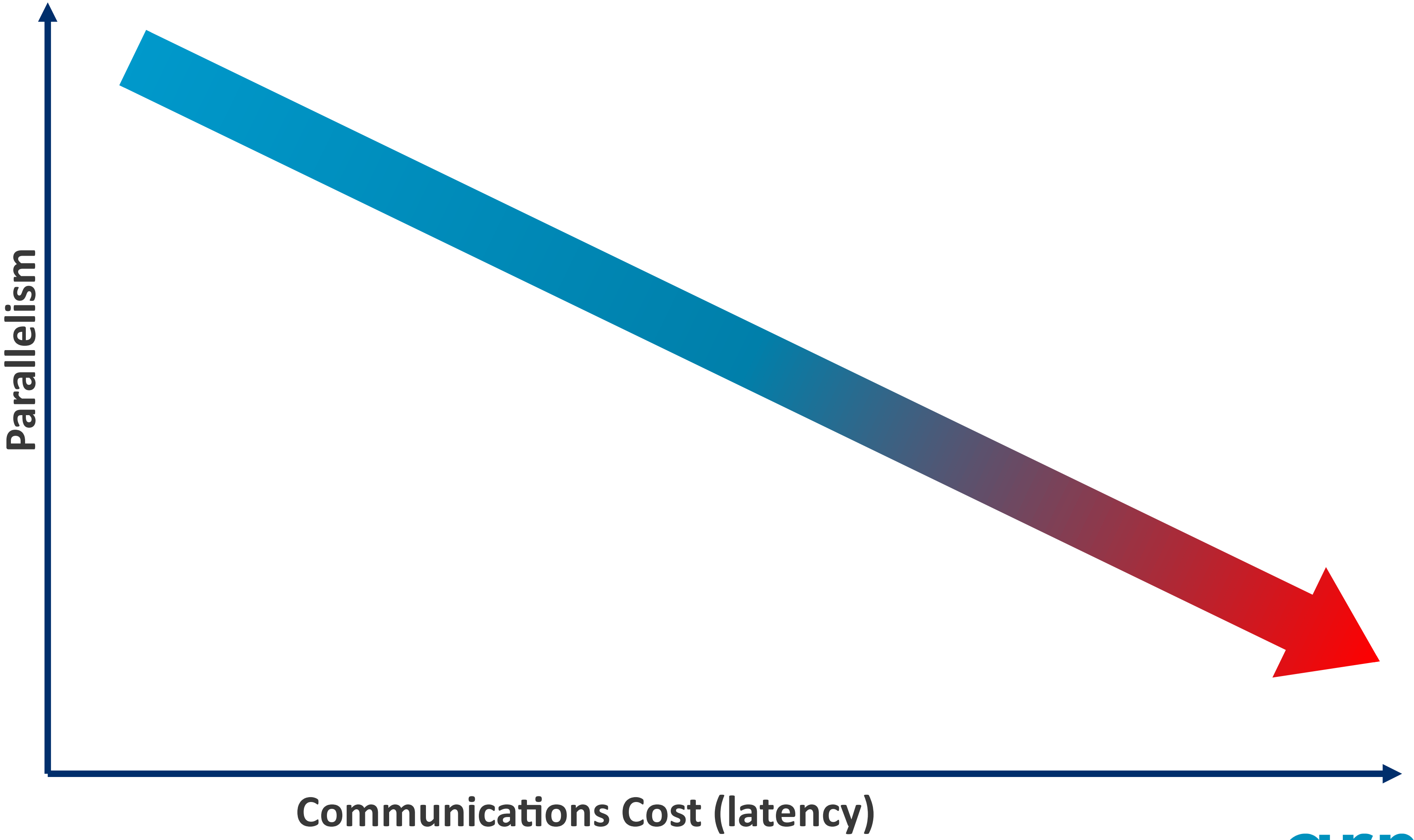
New and cool memory ~~technologies~~ SYSTEMS

Jonathan Beard

2 October 2019

Compute efficiency post-Moore

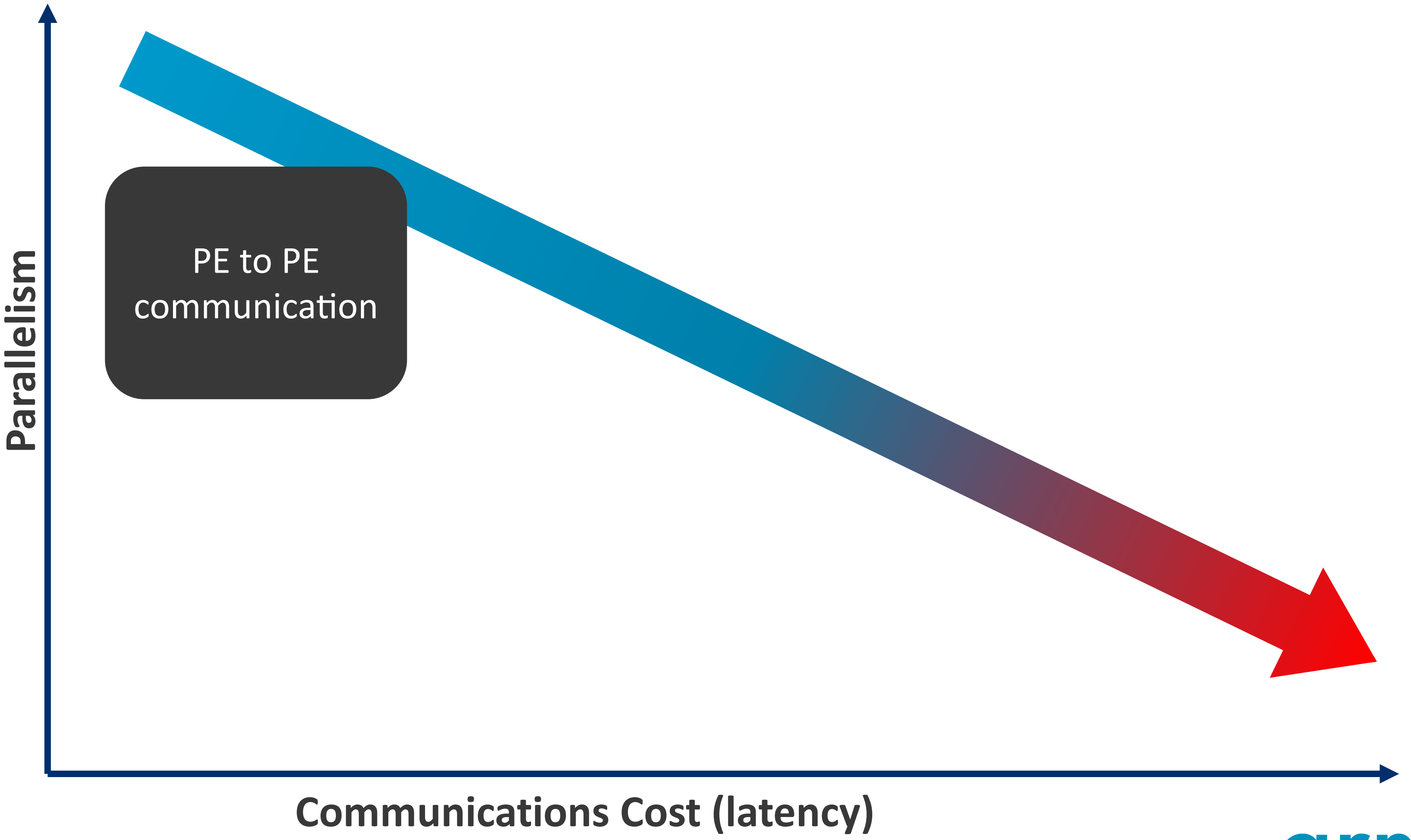
Data movement dominates, parallelism is critical



arm Research

Compute efficiency post-Moore

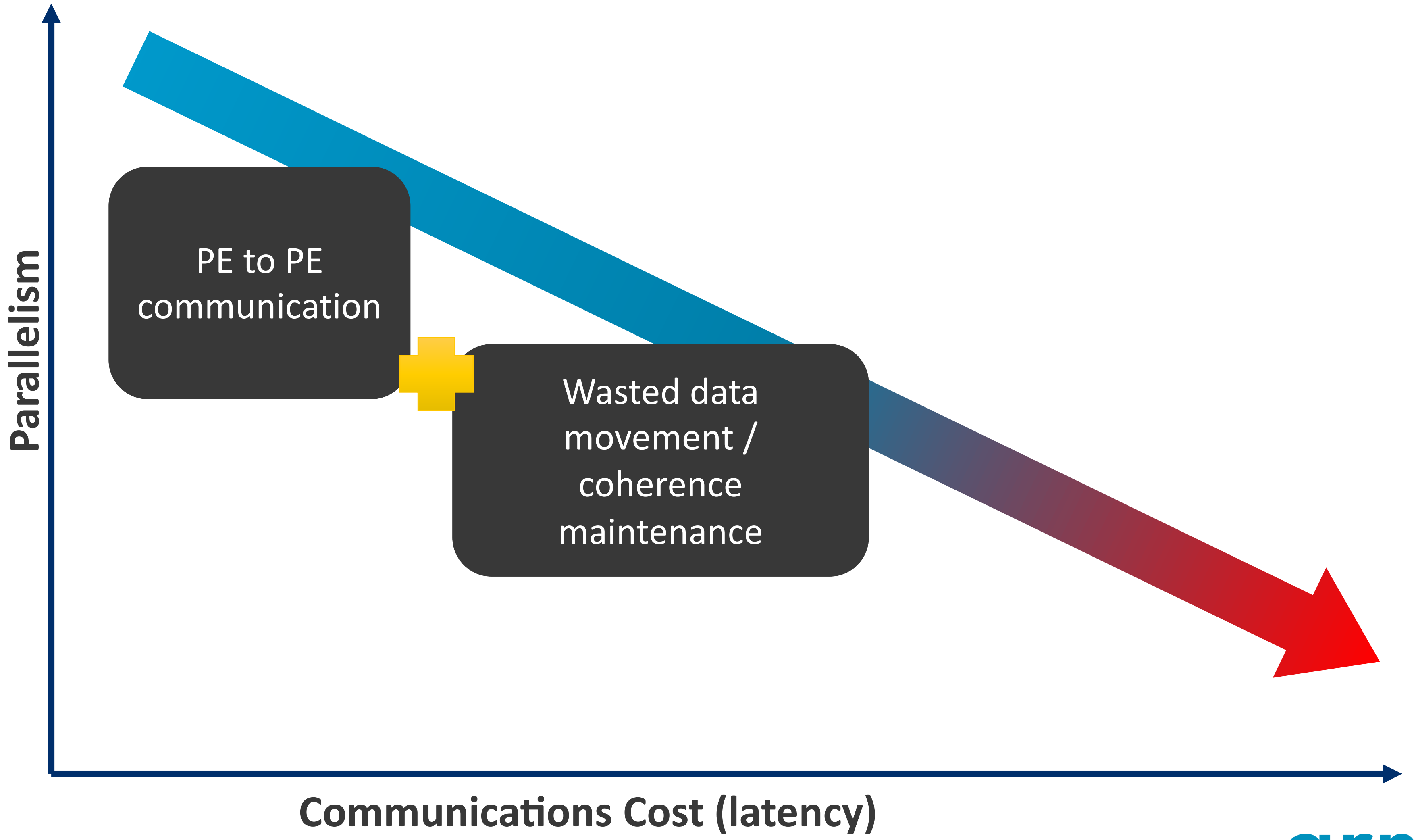
Data movement dominates, parallelism is critical



PE to PE communication

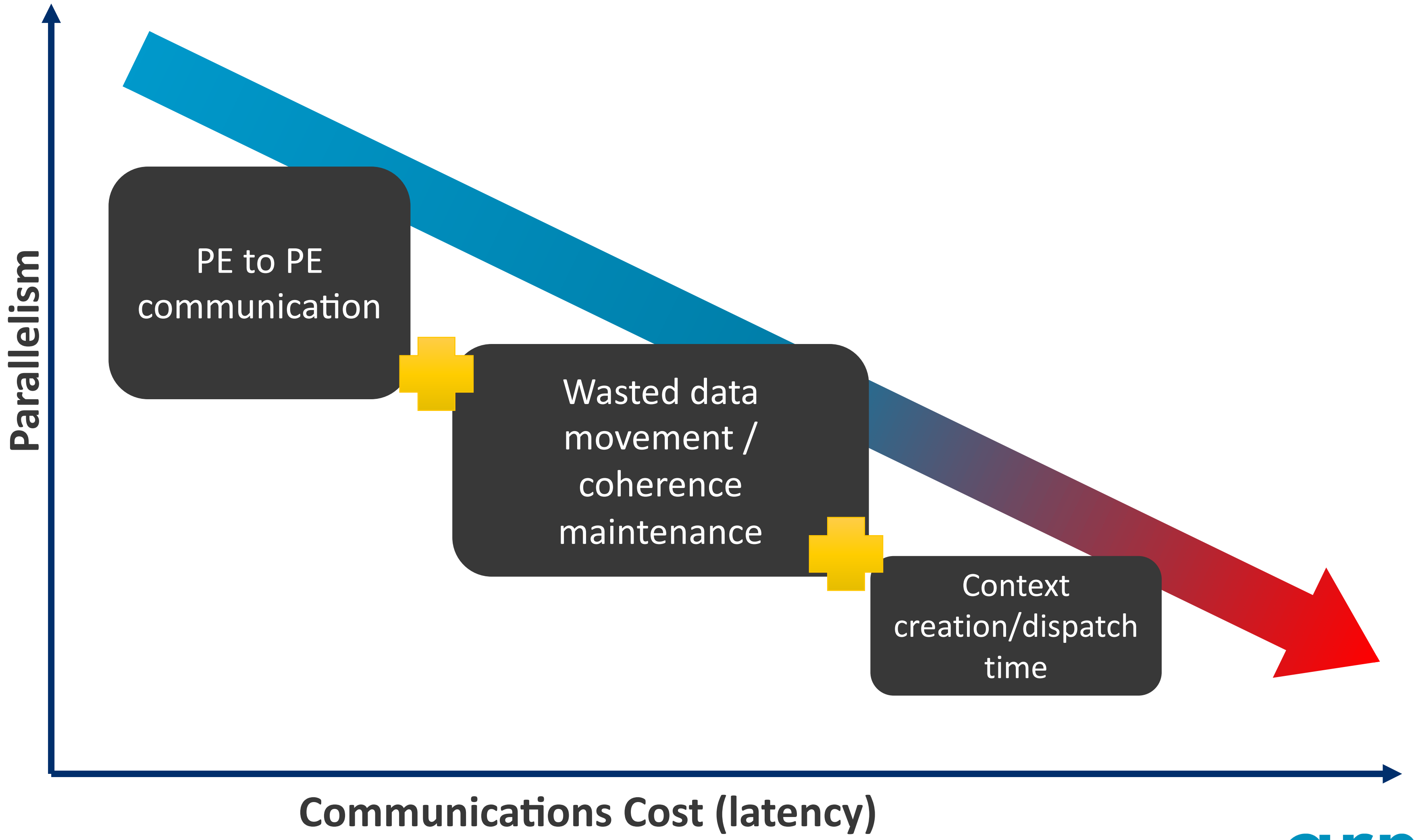
Compute efficiency post-Moore

Data movement dominates, parallelism is critical


Parallelism
PE to PE communication
Wasted data movement / coherence maintenance
Communications Cost (latency)

Compute efficiency post-Moore

Data movement dominates, parallelism is critical



Parallelism

PE to PE communication

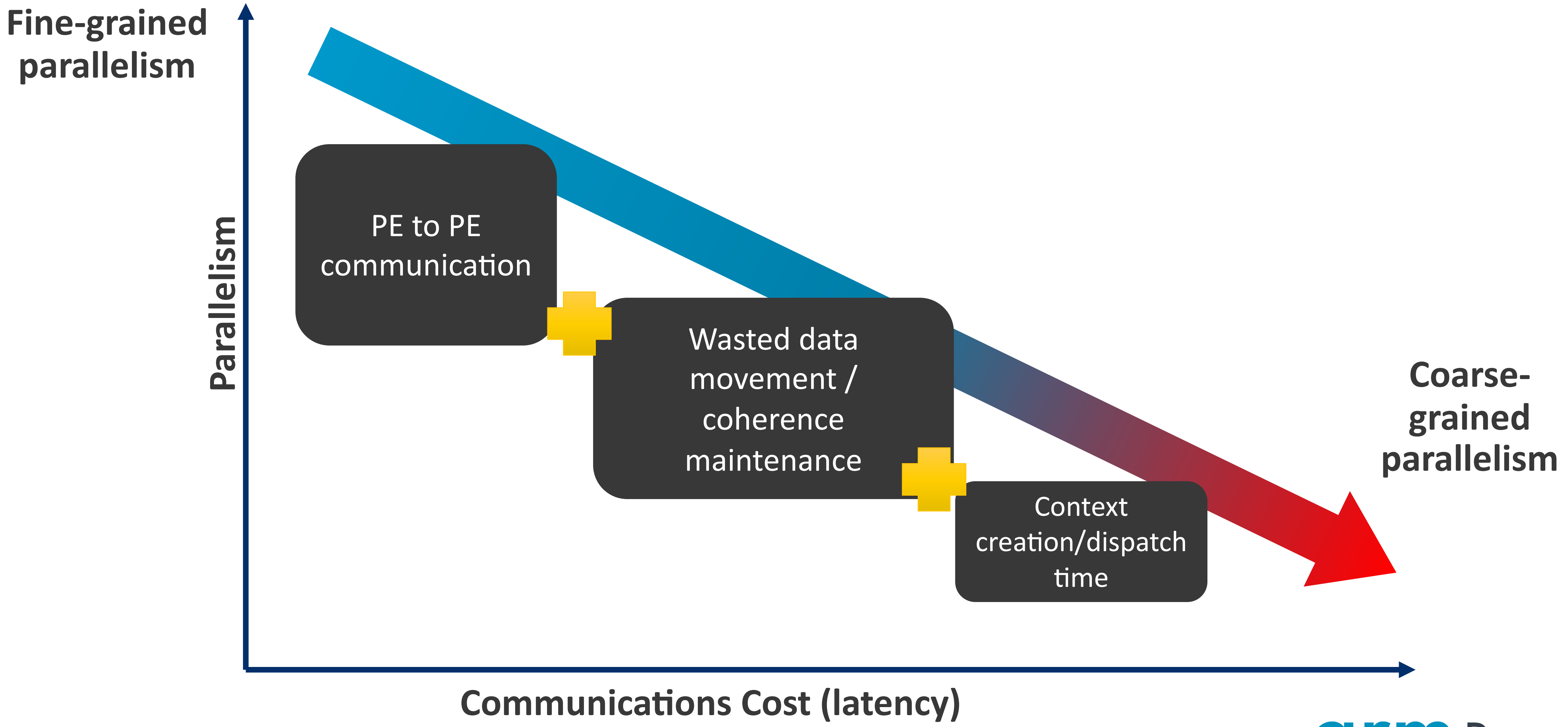
Wasted data movement / coherence maintenance

Context creation/dispatch time

Communications Cost (latency)

Compute efficiency post-Moore

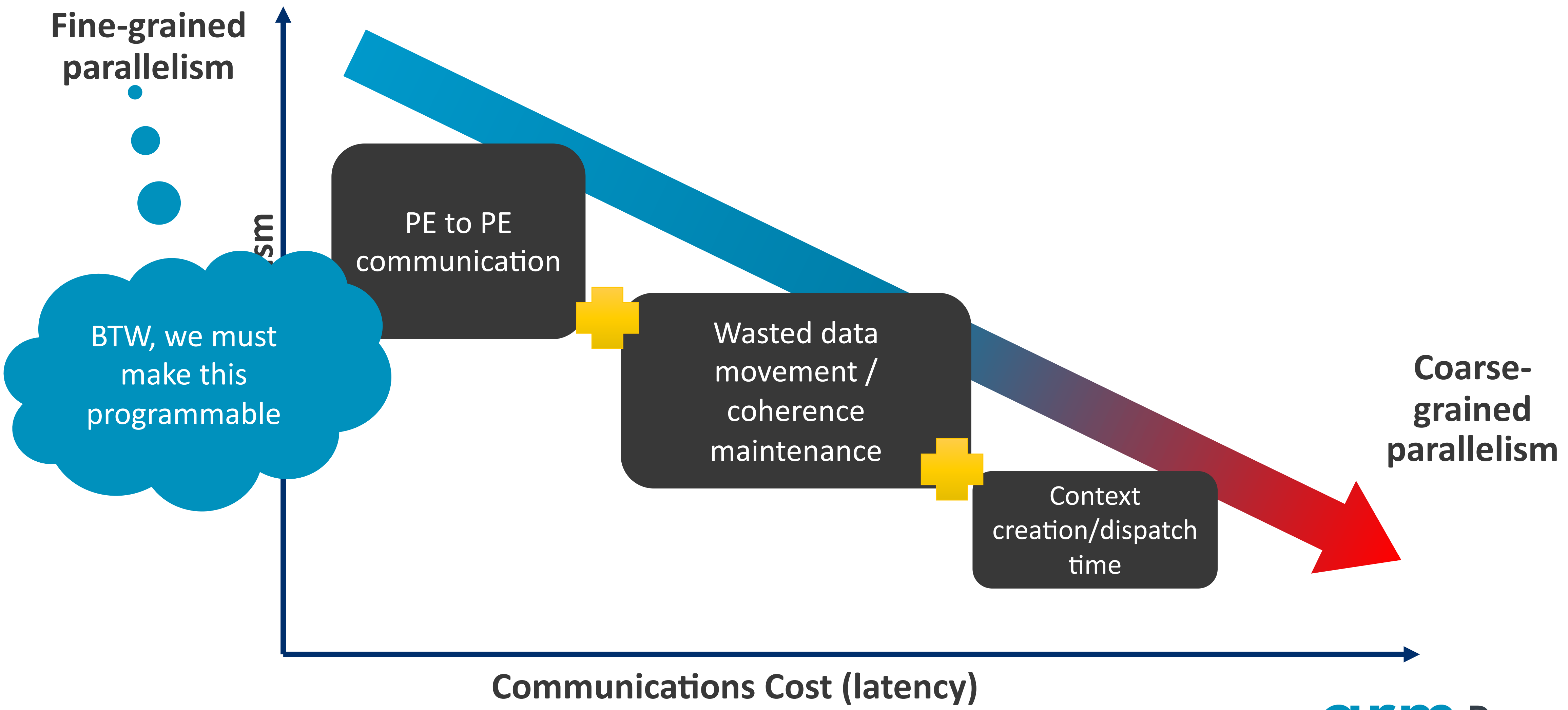
Data movement dominates, parallelism is critical



Fine-grained parallelism
Parallelism
PE to PE communication
Wasted data movement / coherence maintenance
Context creation/dispatch time
Coarse-grained parallelism
Communications Cost (latency)

Compute efficiency post-Moore

Data movement dominates, parallelism is critical



Fine-grained parallelism

sm

PE to PE communication

Wasted data movement / coherence maintenance

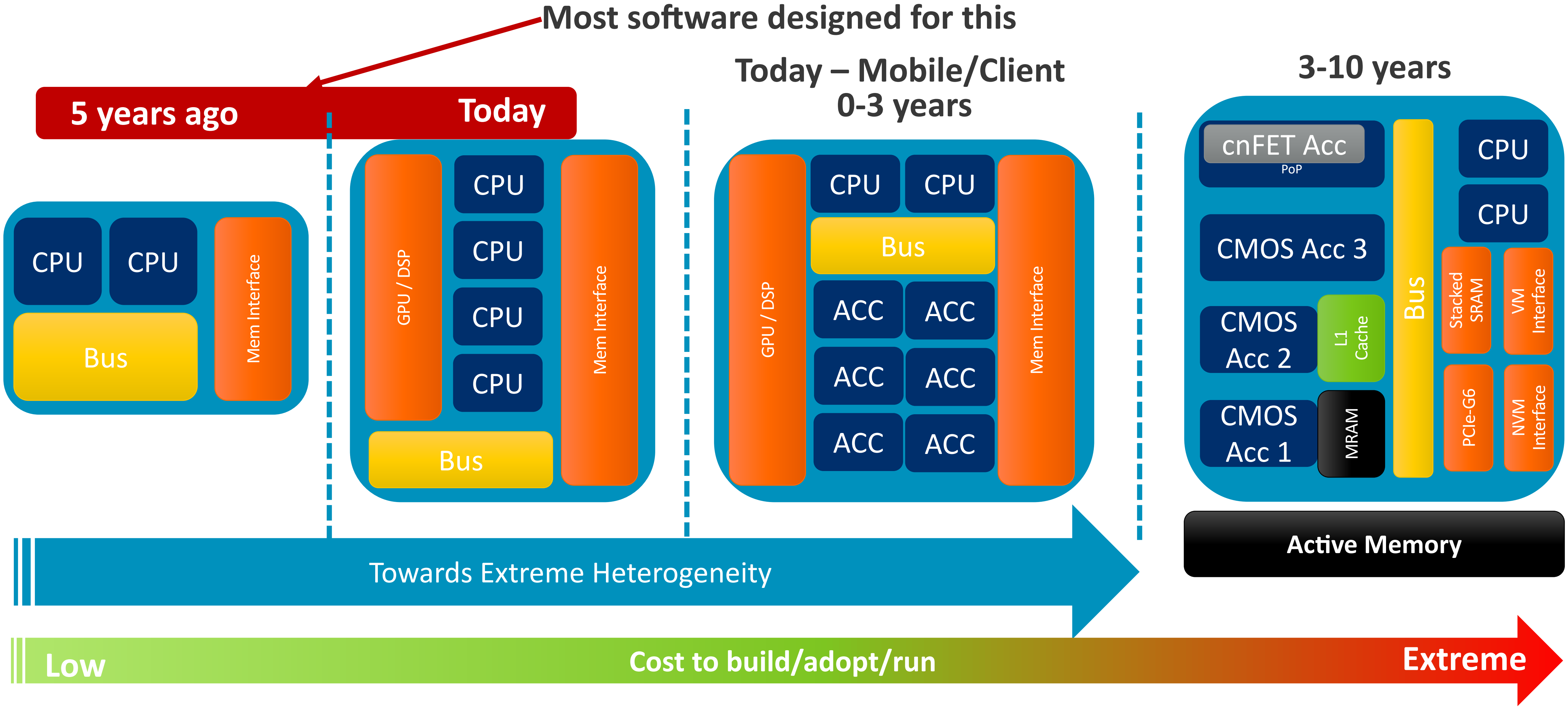
Context creation/dispatch time

BTW, we must make this programmable

Coarse-grained parallelism

Communications Cost (latency)

Adapted/modified from original figure courtesy of Dilip Vasudevan



Most software designed for this

5 years ago

Today

Today – Mobile/Client 0-3 years

3-10 years

CPU
CPU
Bus
Mem Interface

GPU / DSP
CPU
CPU
CPU
CPU
Bus
Mem Interface

GPU / DSP
CPU
CPU
Bus
ACC
ACC
ACC
ACC
ACC
ACC
Mem Interface

cnFET Acc
PoP
CMOS Acc 3
CMOS Acc 2
L1 Cache
CMOS Acc 1
MRAM
Bus
CPU
CPU
Stacked SRAM
VM Interface
PCIe-G6
NVM Interface

Active Memory

Towards Extreme Heterogeneity

Low

Cost to build/adopt/run

Extreme

Bottom line up front

Just in case I get kicked off the stage before I finish…..

- Problem we're really trying to solve is always data movement
  - Context to PE
  - PE communicating results to PE
  - Creating new context from parent PE
  - PE storing results
  - Aligning instruction/command data with input data
- PINM is just another accelerator, but not one we should tackle first.
- We have to face our inconvenient truths……

Bottom line up front

Just in case I get kicked off the stage before I finish…..

- Most PINM solutions often have issues with
  - VA -> PA Translation / interleaving (bank/channel/etc.)
  - Programmability
  - Cache maintenance operations? Where?
  - In-NVM compute, what happens when cells die? Interaction with wear-leveling??
  - Exceptions, error handling?
  - Synchronization: between PINM units and with host cores
  - Working set size vs. device size….thread migration is needed (some have solutions, do others?)
- There are no magic memories
  - If it sounds too good to be true, it usually is.

Bottom line up front

Just in case I get kicked off the stage before I finish…..

- Previous slide is a tad depressing…
- Let’s talk about some easier opportunities….

What is memory?

“the faculty by which the mind stores and remembers information”

- Apple Dictionary

“the faculty by which the mind stores and remembers information”

- Apple Dictionary

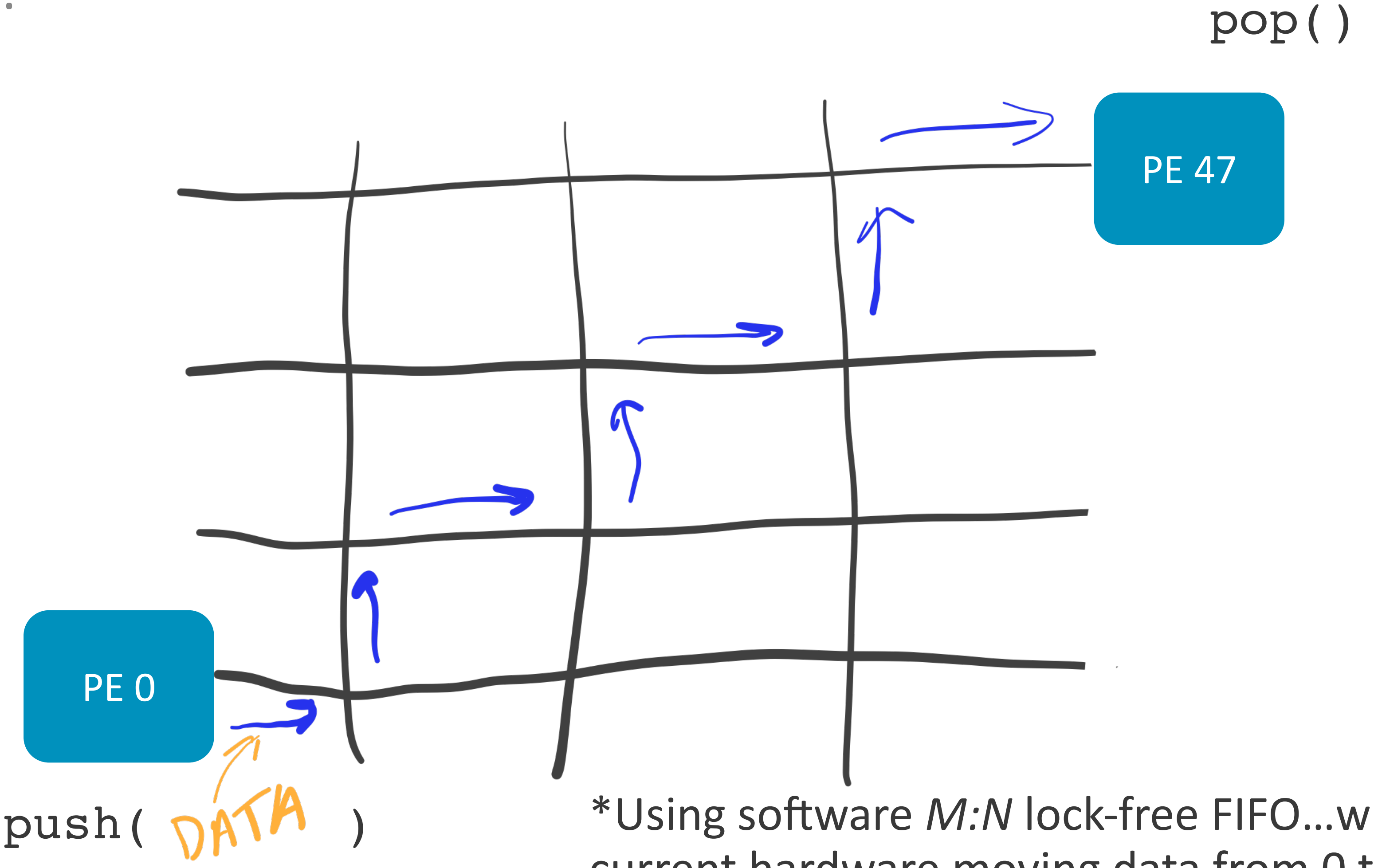
“the faculty by which the application ~~mind~~ stores and remembers information”

- Apple Dictionary, edited ☺

Why don’t we consider the interconnect as memory too?

Maybe…it should be, it is right?

- It stores memory right? Even if only a few cycles at a time.
- Interconnect is way cheaper than DRAM (energy/latency)
- Keep data within interconnect when possible.
- Why is this so hard to do? Some networking cores have, why not general purpose?

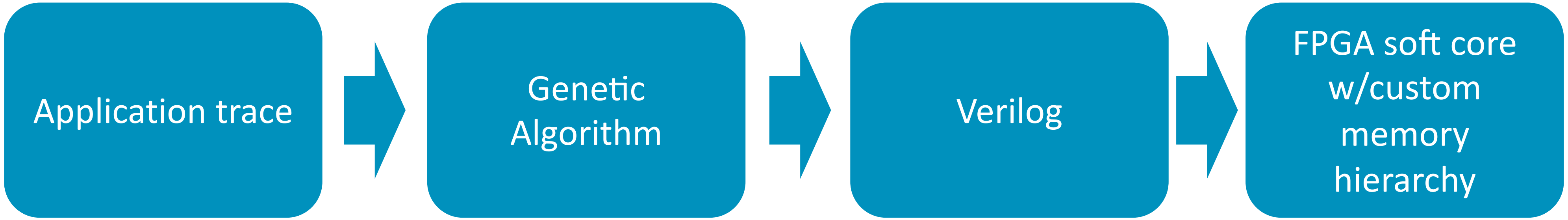


*Using software *M:N* lock-free FIFO…with current hardware moving data from 0 to 47 takes about **500** cycles ☹. Could take as few as **10-100**.

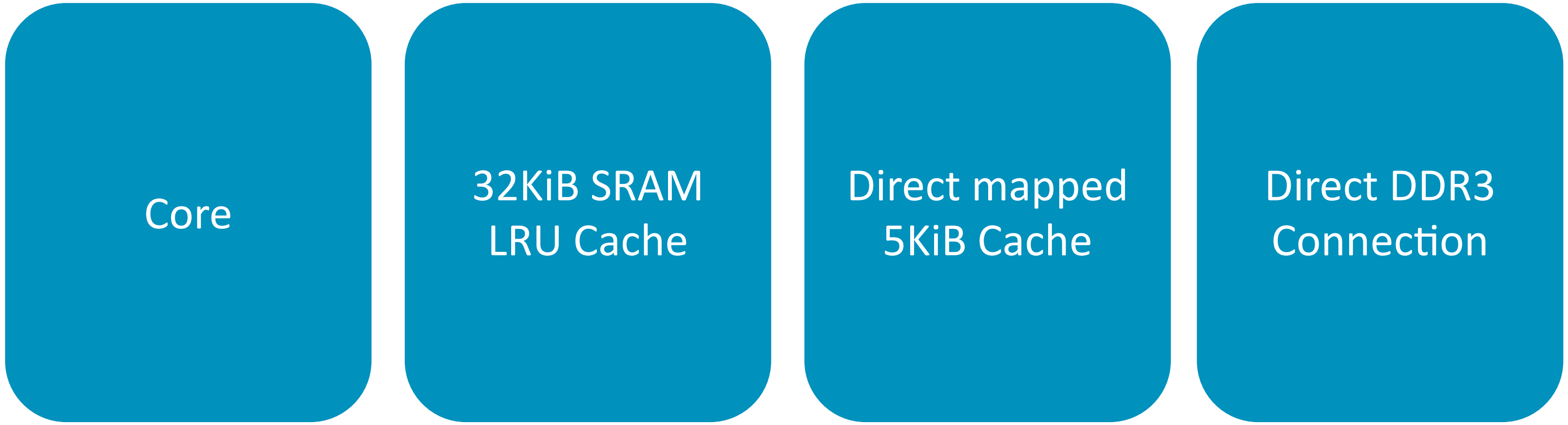
Accelerator != Just Logic

**Concept by Joe Wingbermuehle – "Application Specific Memory Subsystems", PhD Thesis, 2015*

Customized memory hierarchy

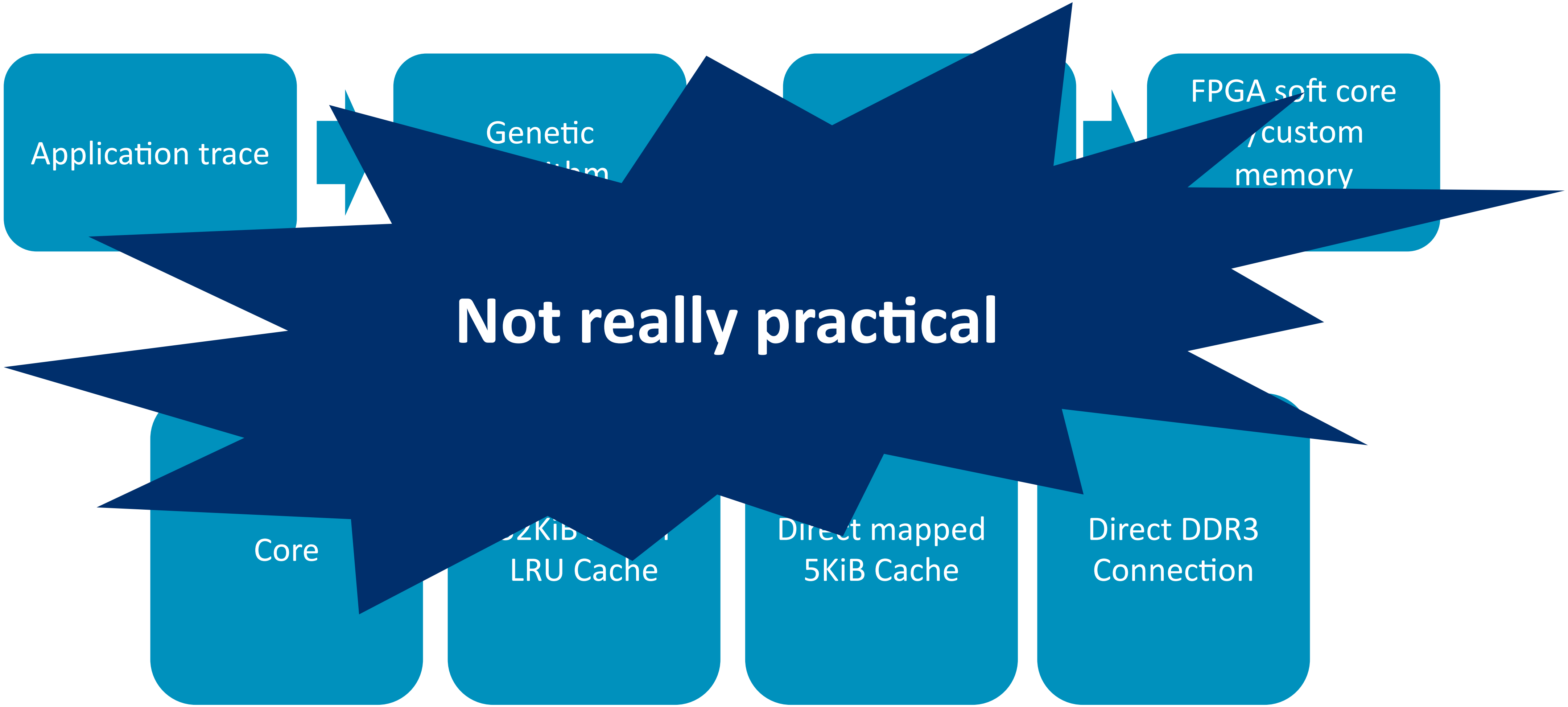


There are some strange combinations



But…2-10x speedup on FPGA, up to 100x when built as ASIC (higher clock rates)

Customized memory hierarchy

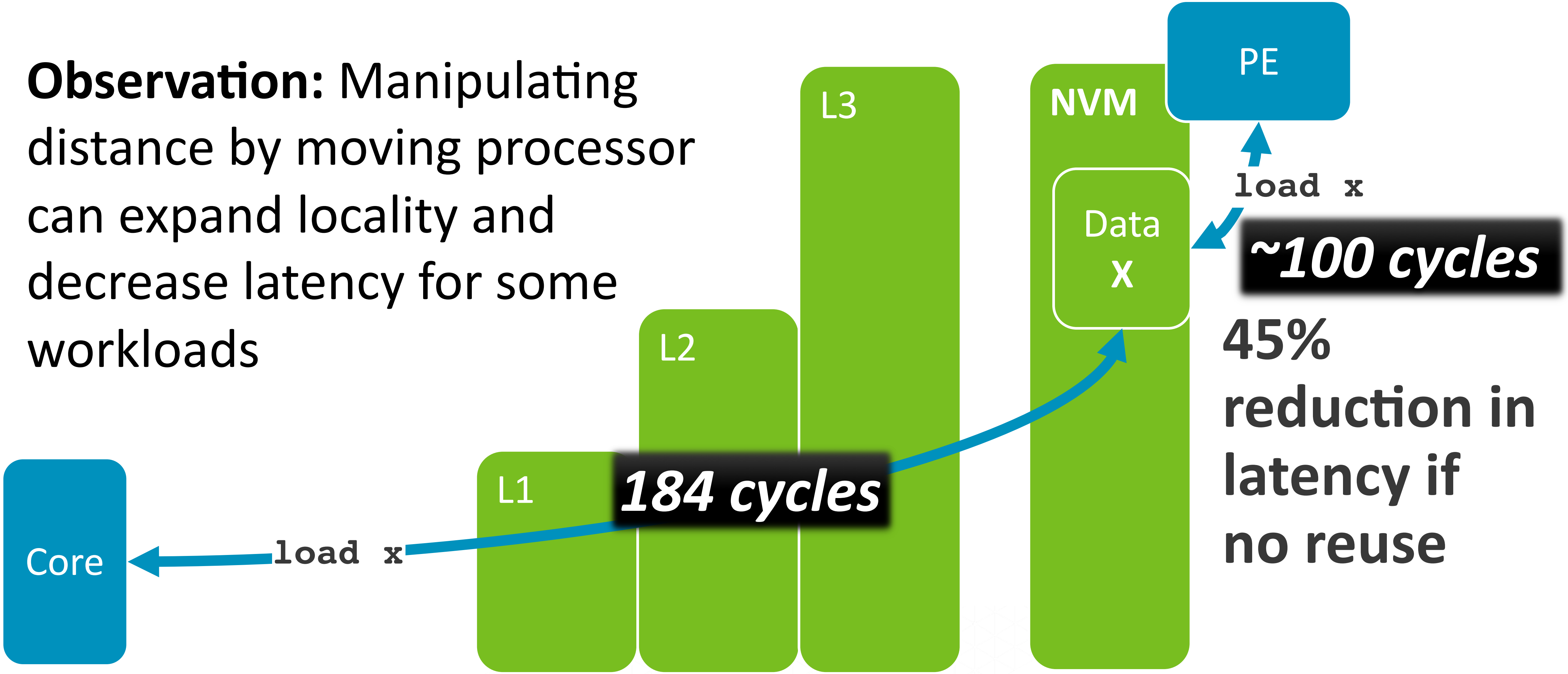


Not really practical

Relativity

Locality is, from a certain point of view.

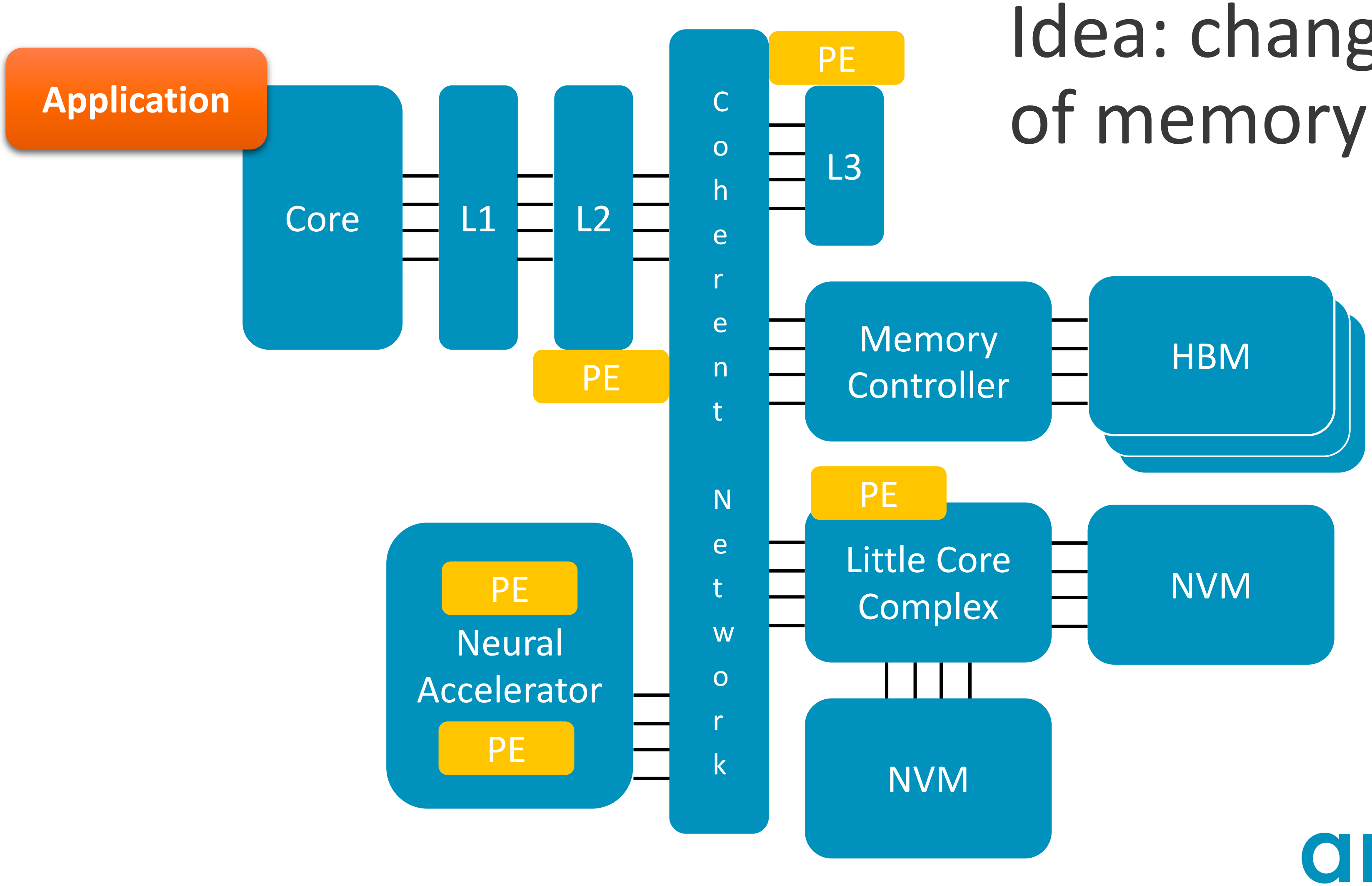
Observation: Manipulating distance by moving processor can expand locality and decrease latency for some workloads



45% reduction in latency if no reuse

Flip the script

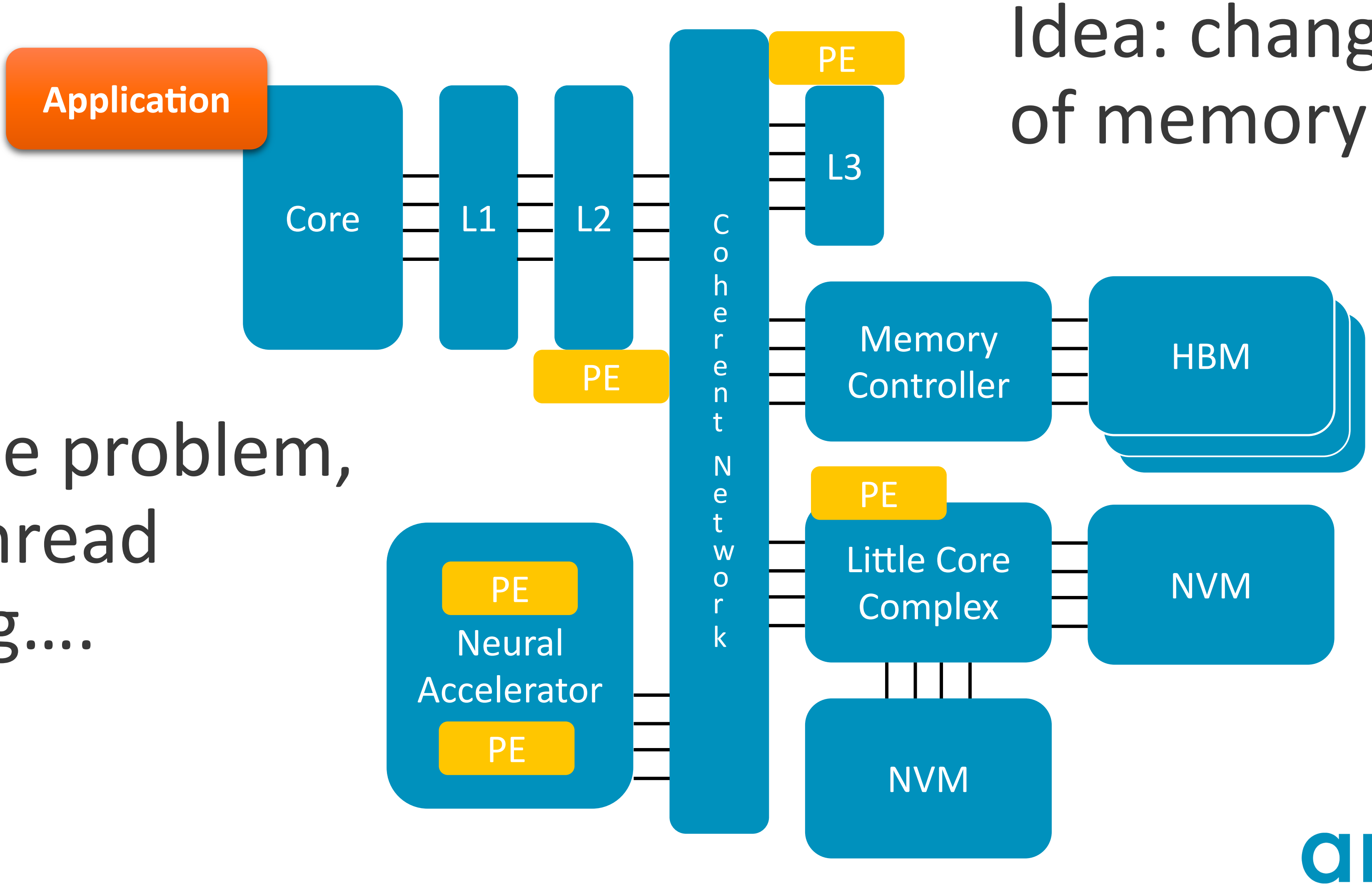
Instead of changing memory hierarchy, add processors (PEs) everywhere



Idea: change “view” of memory hierarchy

Flip the script

Instead of changing memory hierarchy, add processors (PEs) everywhere



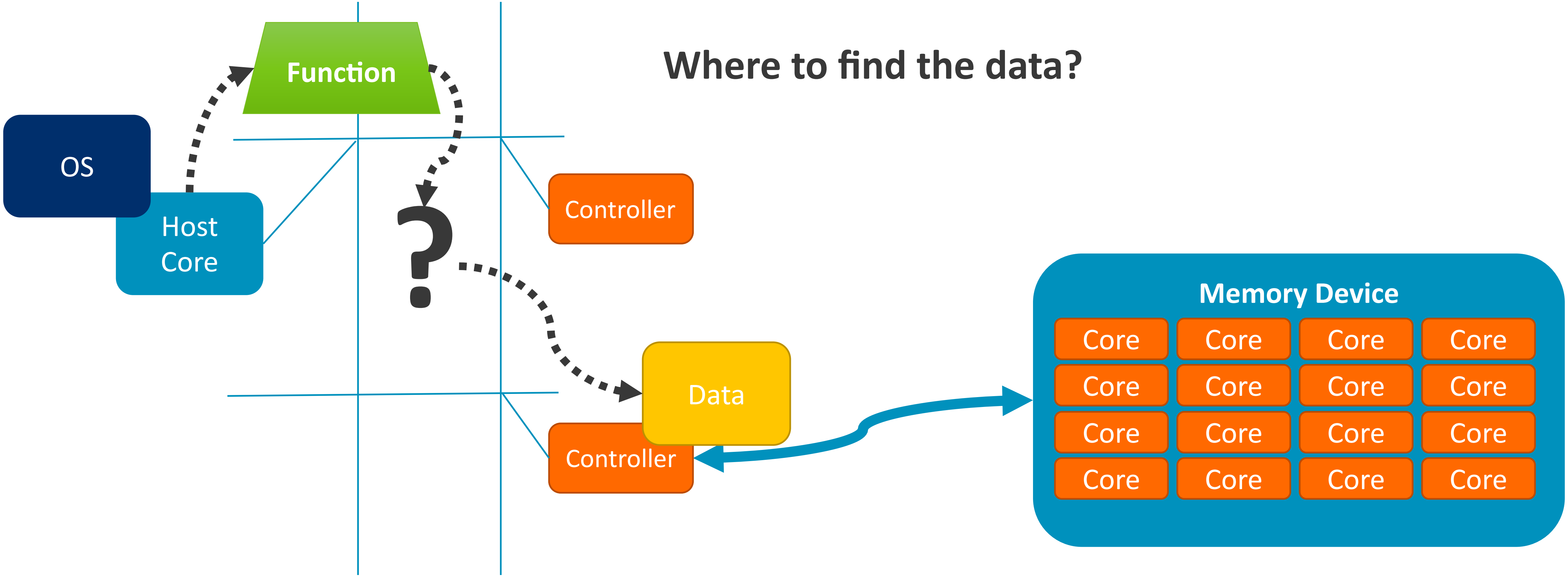
Idea: change "view" of memory hierarchy

Change the problem, it's now thread scheduling….

Do you really want to program that??

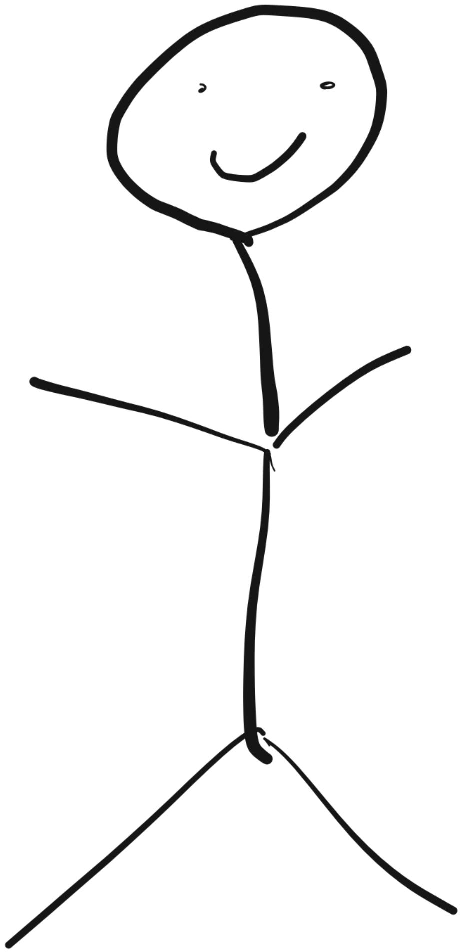
Just one problem among many..

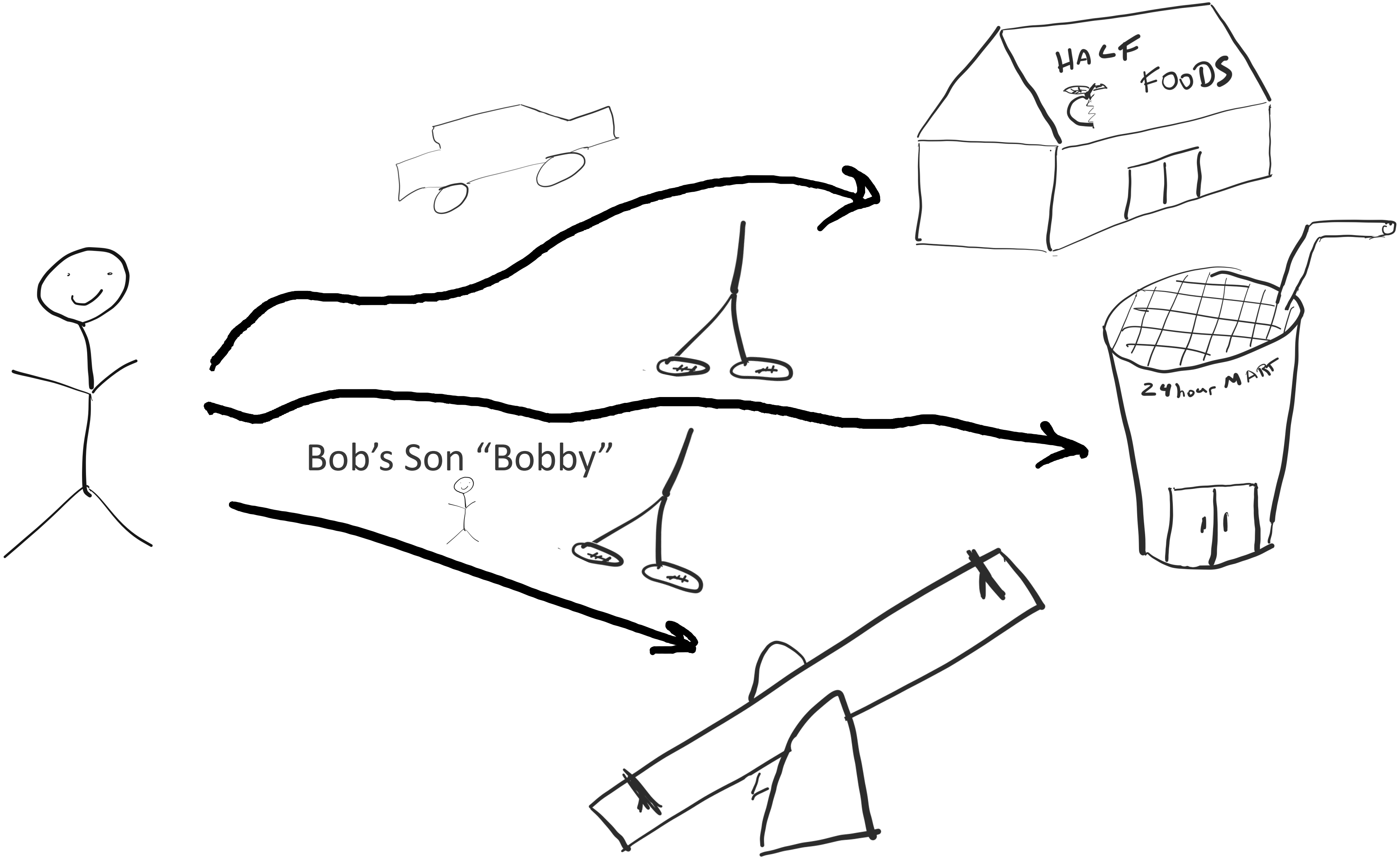
Where to find the data?



The interface

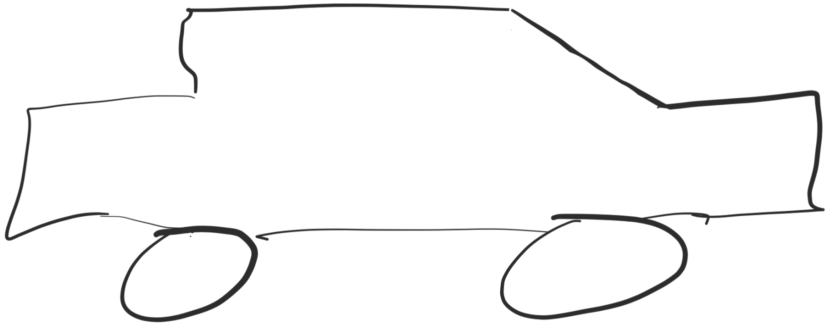
Meet “Bob”….



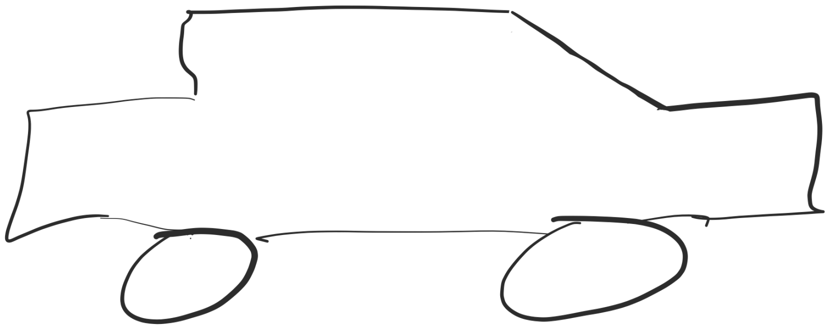


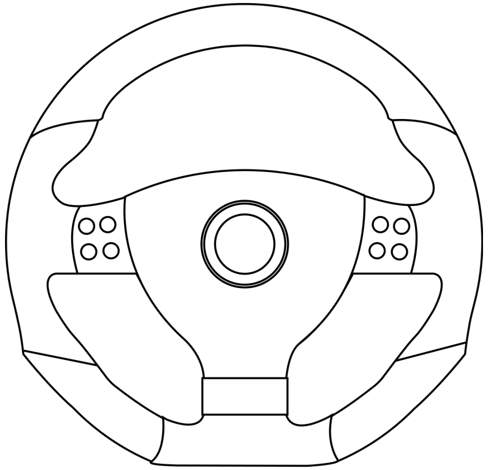
Bob’s Son “Bobby”

Interfaces

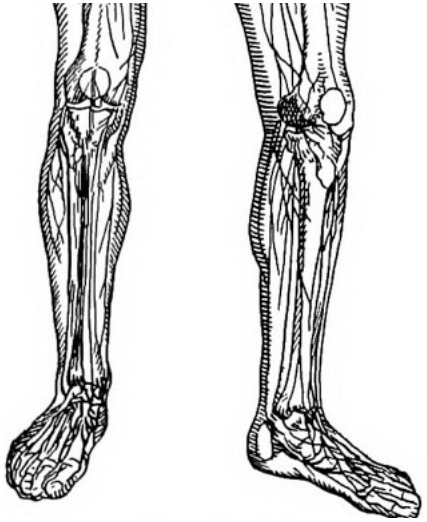


Interfaces

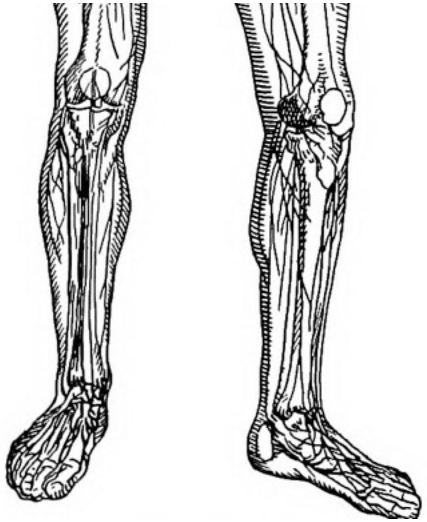


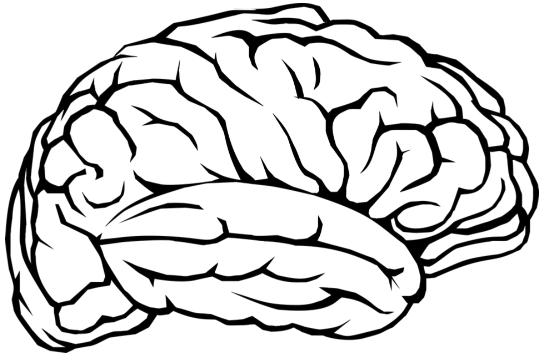


Interfaces



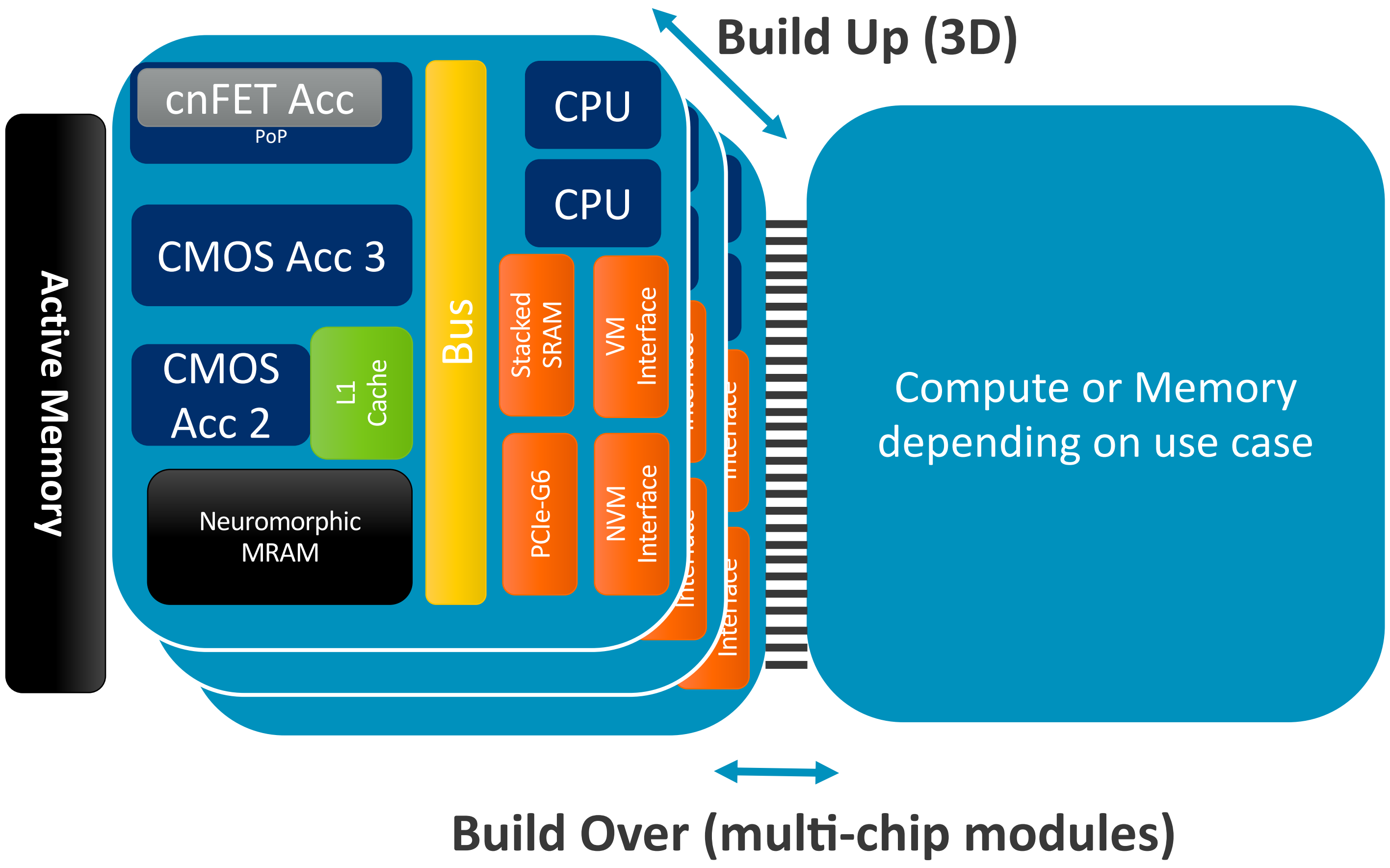
Interfaces



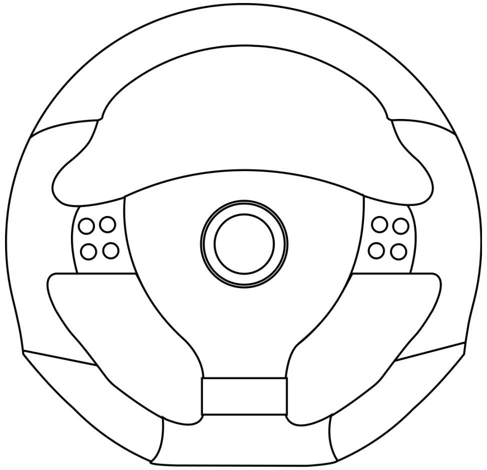


Adapted/modified from original figure courtesy of Dilip Vasudevan (LBL)

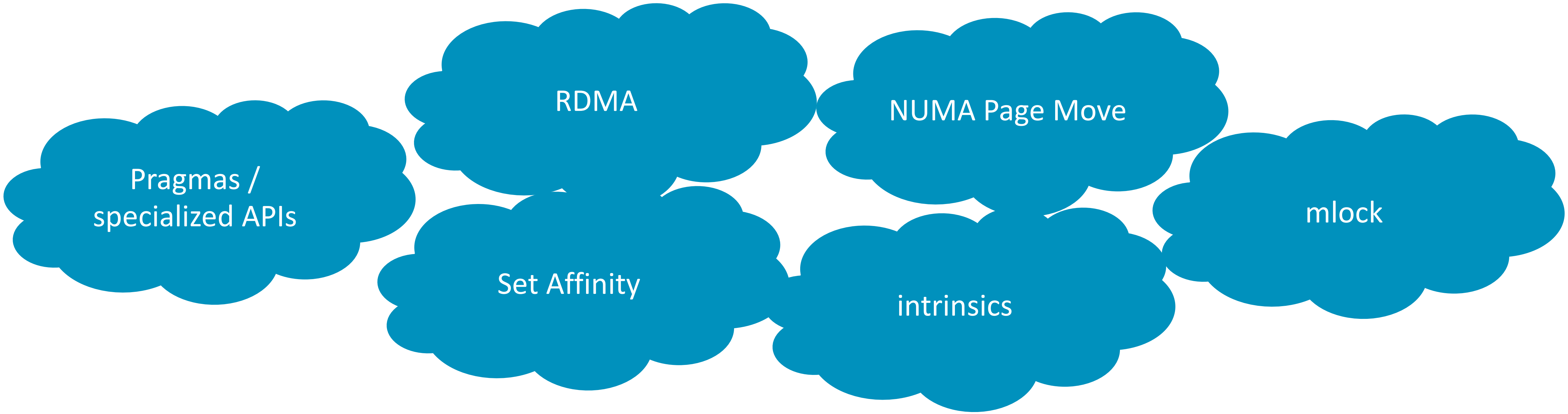
But for programming computer hardware….



We need intuitive, productive interfaces….



But….



Pragmas / specialized APIs

RDMA

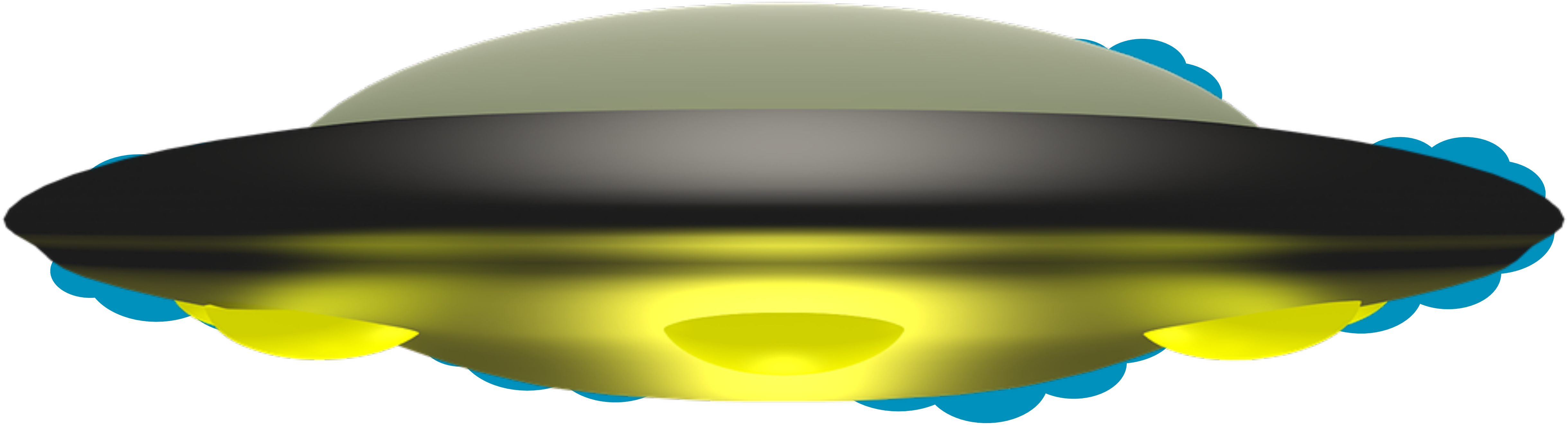
Set Affinity

NUMA Page Move

intrinsics

mlock

But….

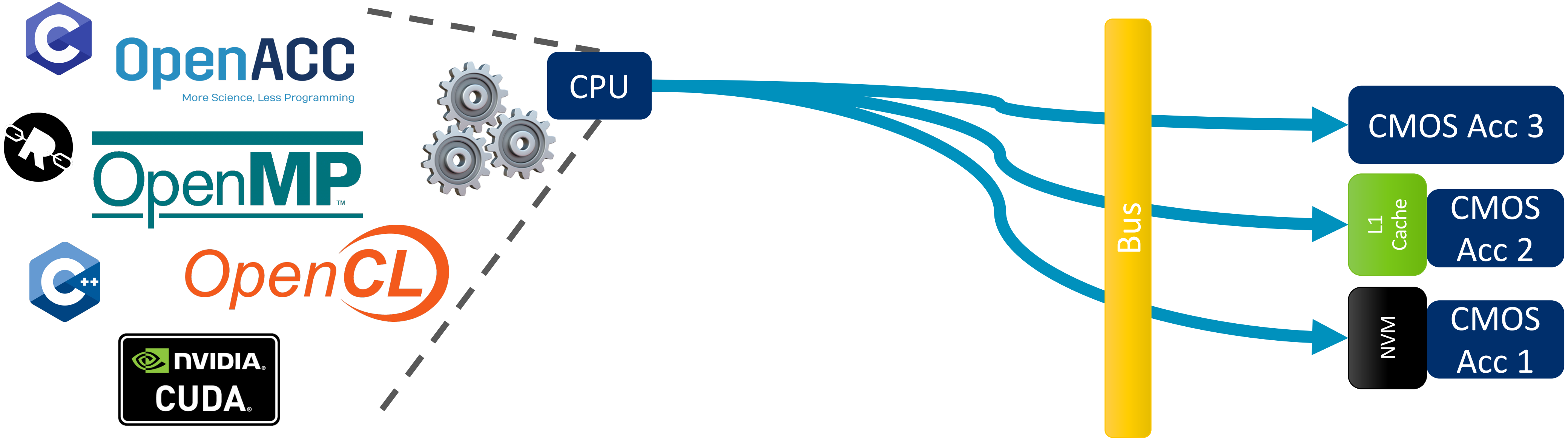


Do we need super advanced aliens to help us build programs??

(after all, they did help build the pyramids right?? ;))

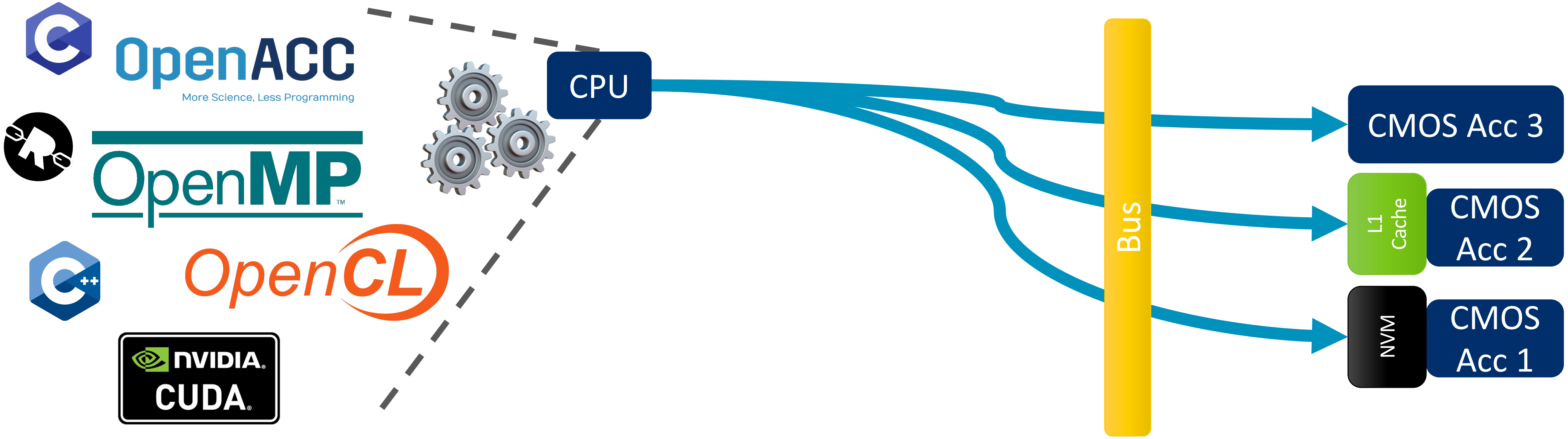
(disclaimer: aliens didn't actually build the pyramids…)

Most “start-up” hardware vendors fail…because of software, or lack of.



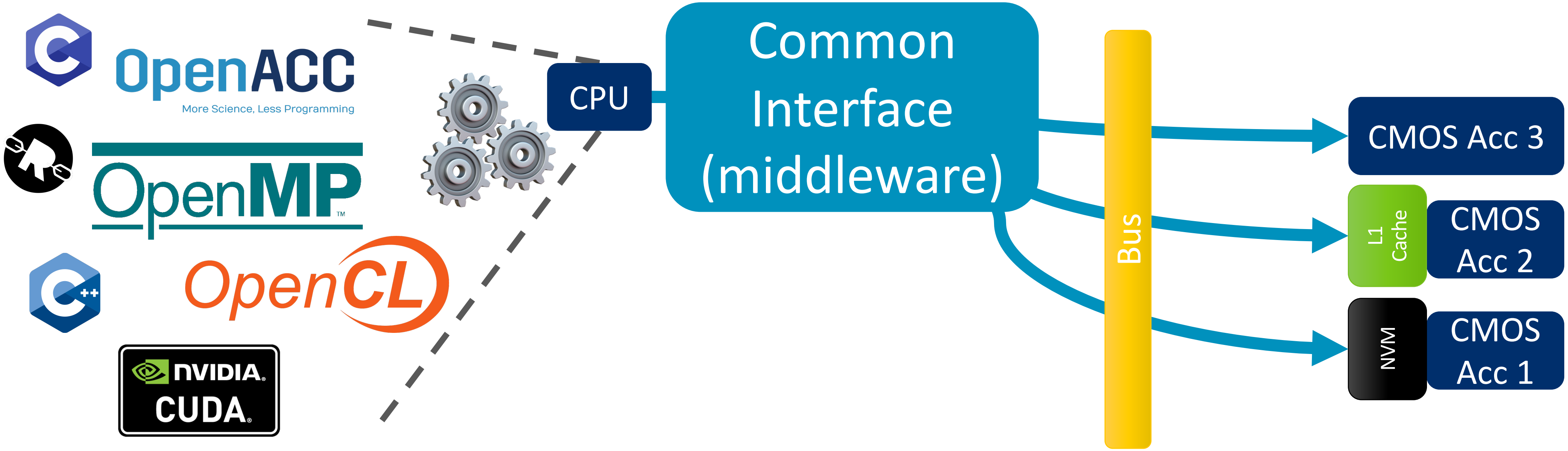
Mature Software Ecosystem

Market success isn't based on having the "best" hardware, it's in having a broad software user-base....more simply, it's the software!!

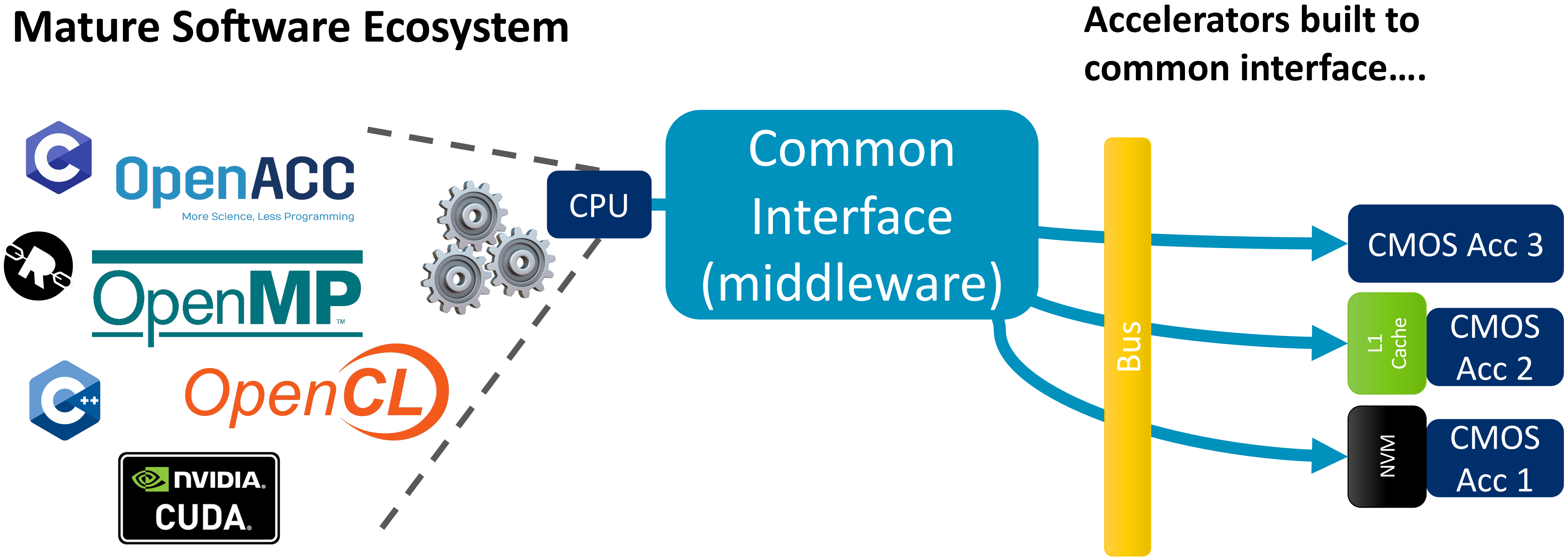


What if….

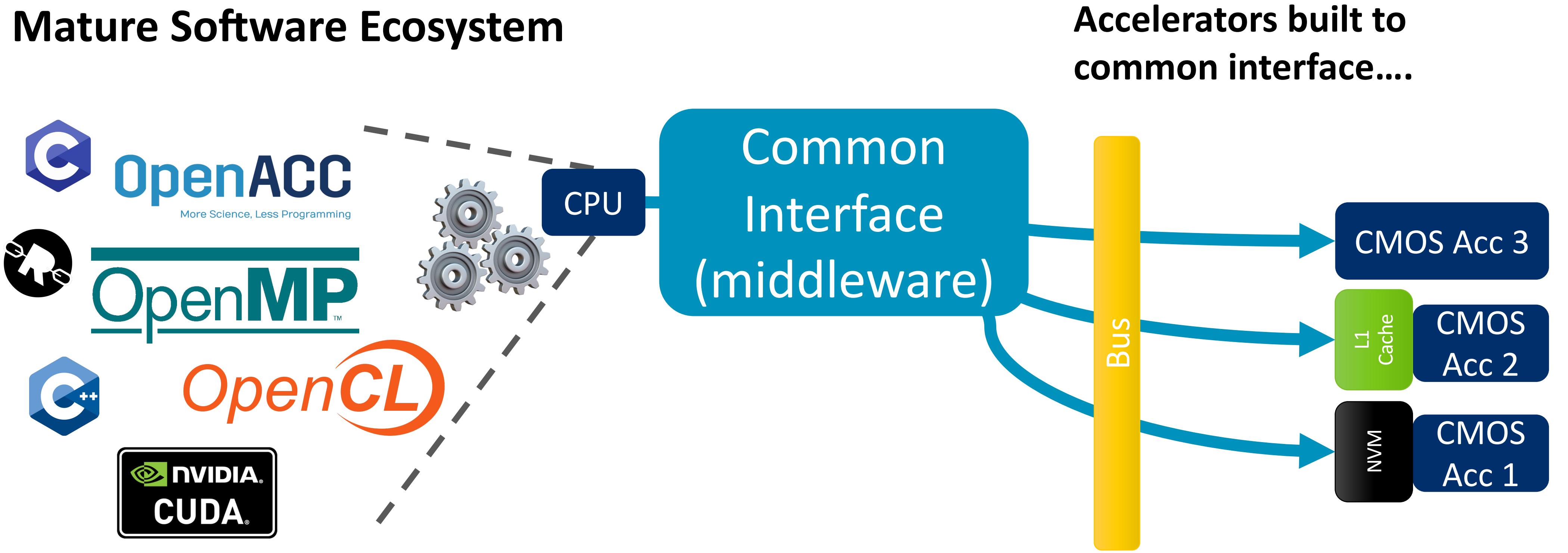
Accelerators built to common interface….



What if….

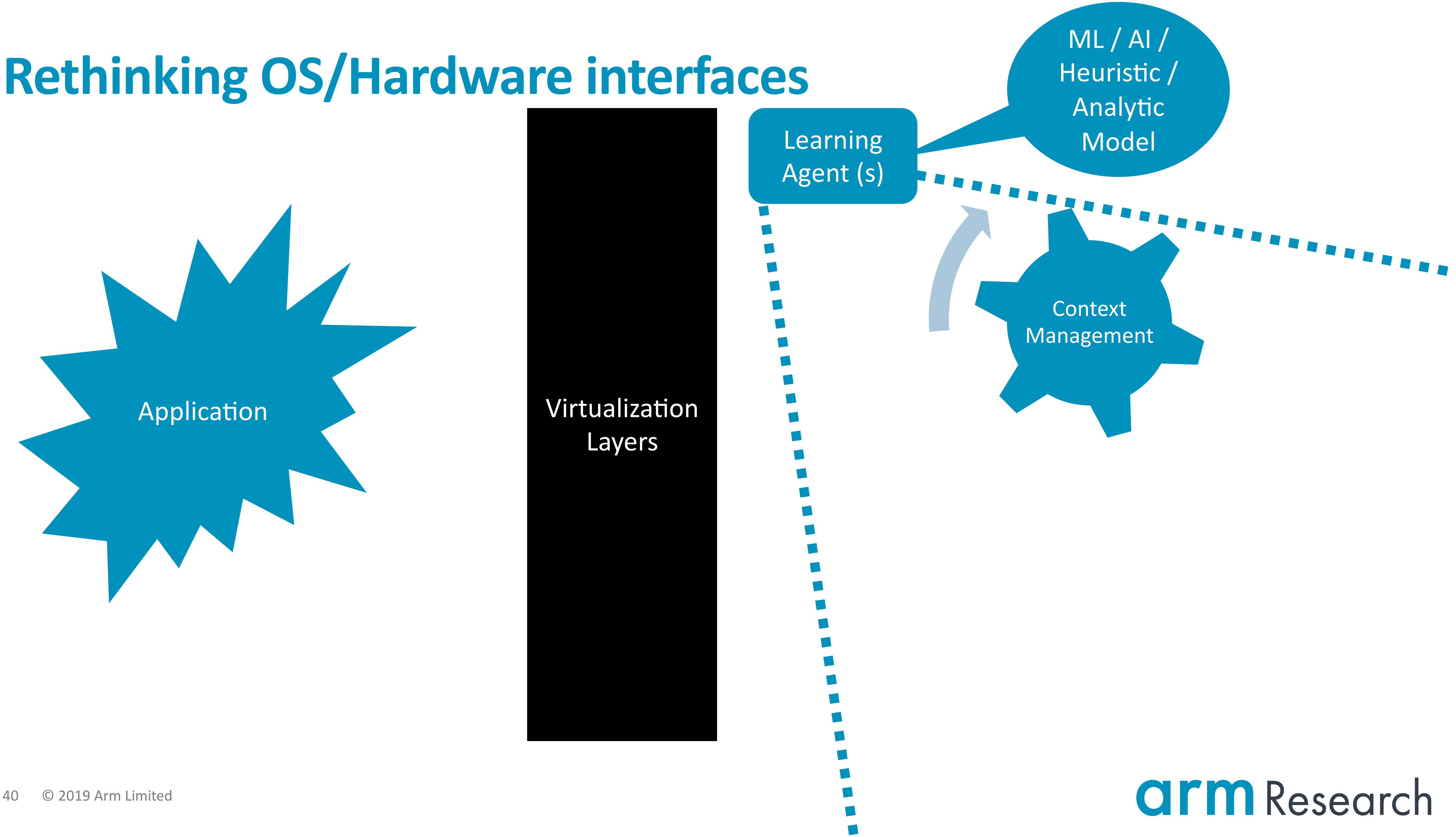

Mature Software Ecosystem
OpenACC
More Science, Less Programming
OpenMP
OpenCL
NVIDIA
CUDA
CPU
Common Interface (middleware)
Accelerators built to common interface….
Bus
CMOS Acc 3
L1 Cache
CMOS Acc 2
NVM
CMOS Acc 1

What if….

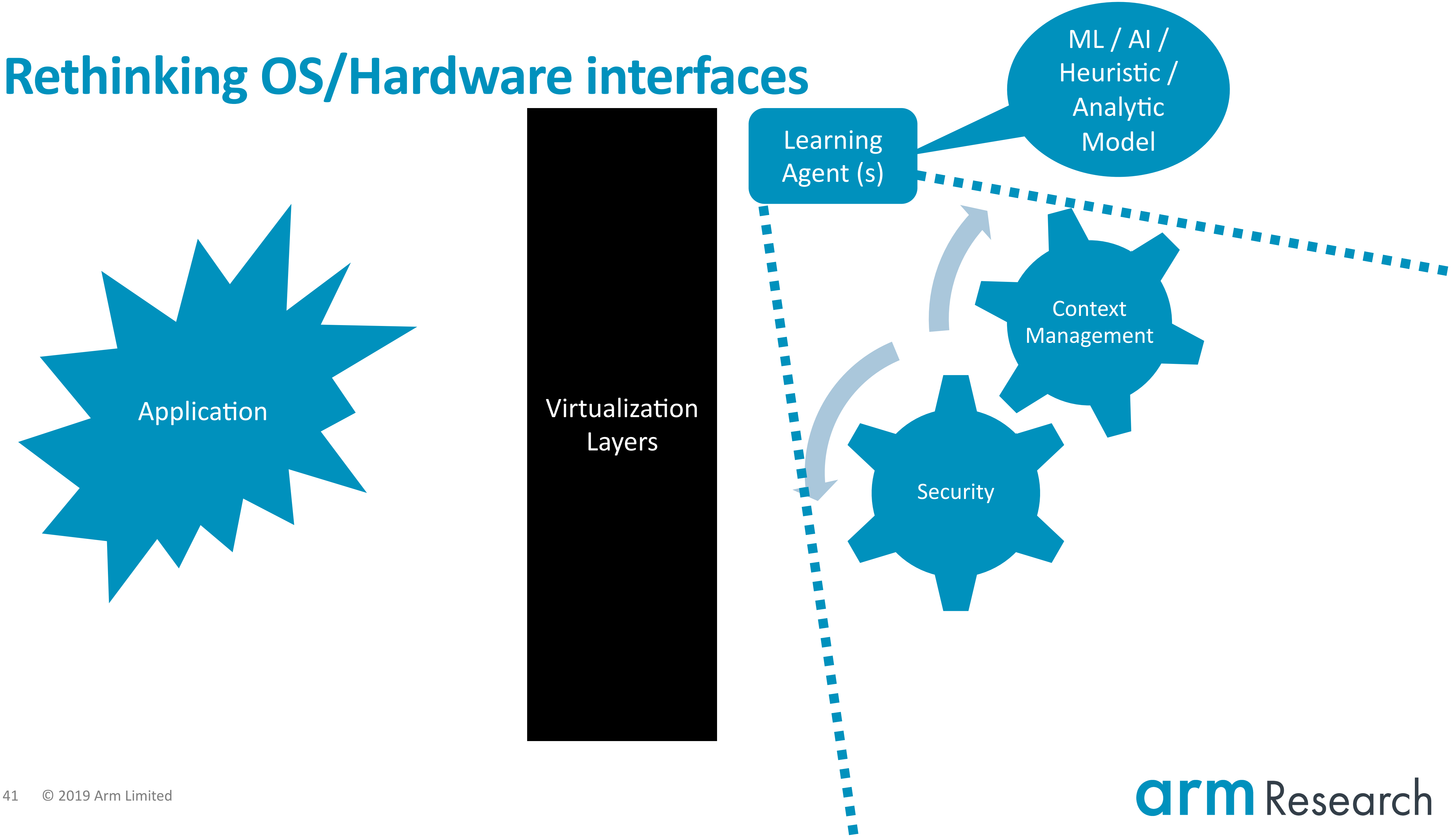

Mature Software Ecosystem
OpenACC
More Science, Less Programming
OpenMP
OpenCL
NVIDIA
CUDA
CPU
Common Interface (middleware)
Accelerators built to common interface….
Bus
CMOS Acc 3
L1 Cache
CMOS Acc 2
NVM
CMOS Acc 1

- Build interface to make it easier to determine data locality, do fast dispatch, etc.!
- Lower offload latency == more tasks / unit time

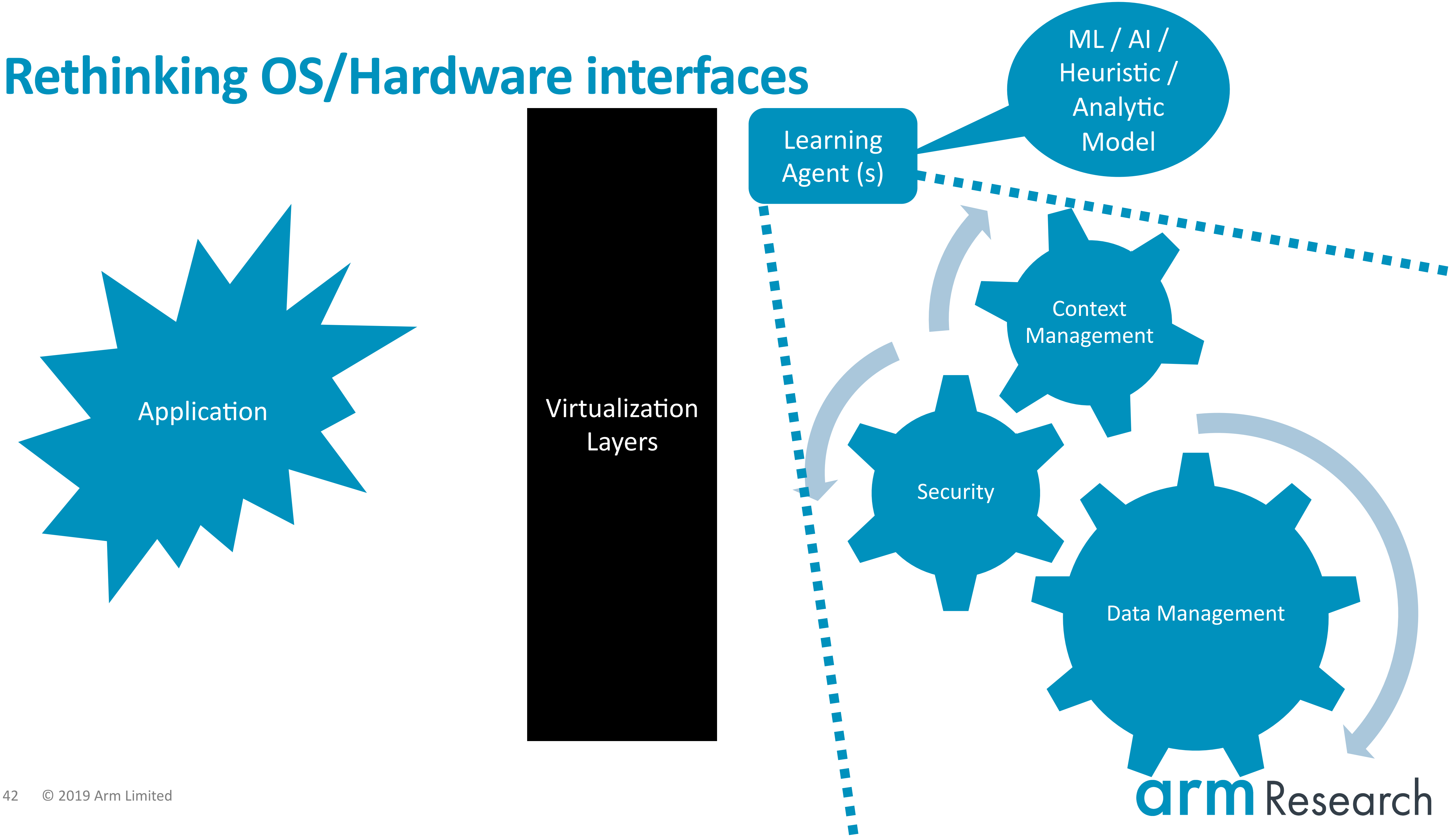
Rethinking OS/Hardware interfaces



Rethinking OS/Hardware interfaces

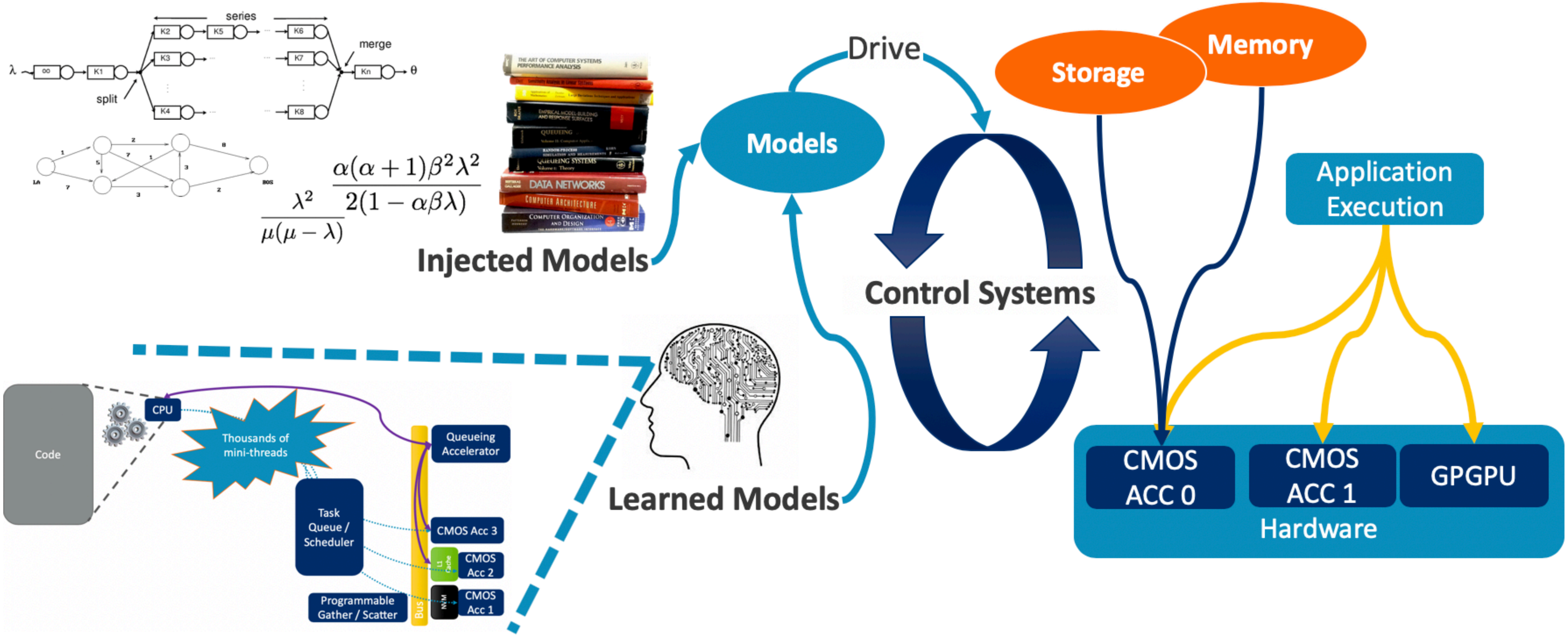


Rethinking OS/Hardware interfaces



The AI/ML Assisted System

(shh, it's not really aliens)



$$\frac{\lambda^2}{\mu(\mu-\lambda)} \qquad \frac{\alpha(\alpha+1)\beta^2\lambda^2}{2(1-\alpha\beta\lambda)}$$

Injected Models

Learned Models

Models

Drive

Control Systems

Storage

Memory

Application Execution

Hardware

Parting thoughts…

- Application performance is all about the data, and how fast you can access it
- Can we build better communications primitives to reduce main memory access?
- Memory is an accelerator, memory is compute, it is not an accessory.
- We shouldn’t need advanced aliens to come down to help us program our systems… ;)

Parting thoughts…

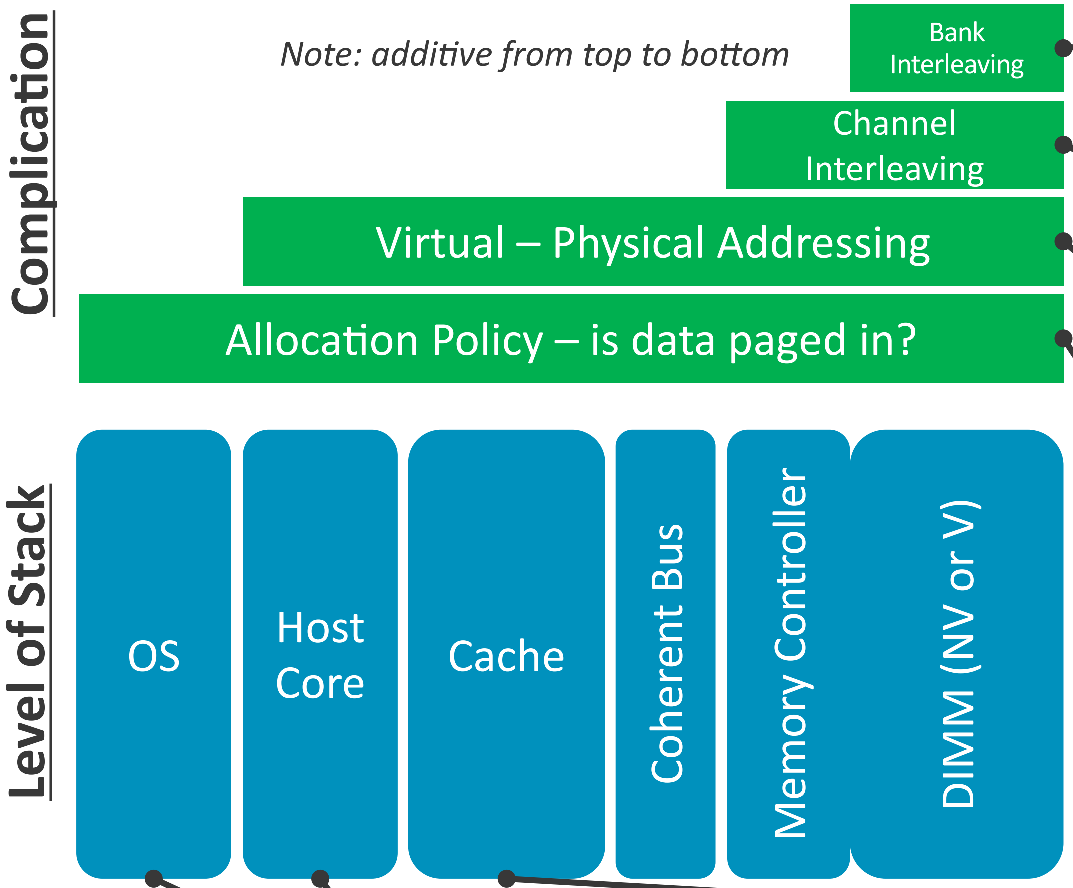
- Application performance is all about the data, and how fast you can access it
- Can we build better communications primitives to reduce memory access?
- Memory is an accelerator, it is compute, it is not an accessory.
- We shouldn't need advanced aliens to come down to help us program our systems… ;)

Thanks!

arm Research

Issues with PINM

Systems integration – translation / interleaving



Device specific, impossible to be generic or "architectural" in general, but we can be generic with X-bit ops or "atomic" ops based on some "feature" register

We must keep channel interleaving for BW, best to be able to continue practice. Best hope is to work with interleaving when present.

IOMMU can suffice for basic uses, however, address is best means of targeting data locality, partitioned schemes are best w.r.t. accelerators / PINM

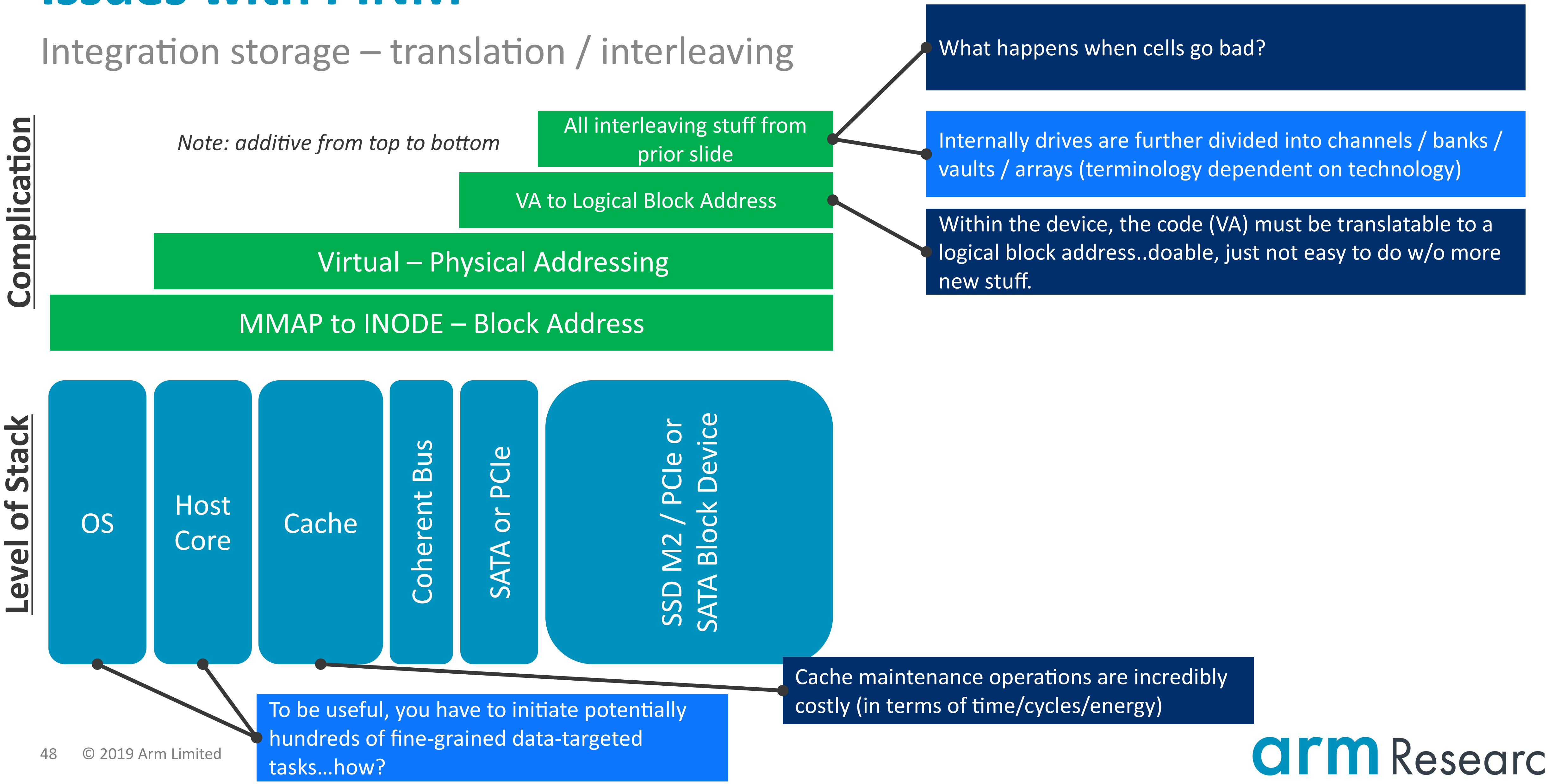
The OS shouldn't migrate or page-out data while in use – system software must either be changed to be PIM aware, or the memory fabric system architecture should be modified

Cache maintenance operations are incredibly costly (in terms of time/cycles/energy)

To be useful, you have to initiate potentially hundreds of fine-grained data-targeted tasks…how?

Issues with PINM

Integration storage – translation / interleaving



Complication

Note: additive from top to bottom

All interleaving stuff from prior slide

VA to Logical Block Address

Virtual – Physical Addressing

MMAP to INODE – Block Address

Level of Stack

OS

Host Core

Cache

Coherent Bus

SATA or PCIe

SSD M2 / PCIe or SATA Block Device

What happens when cells go bad?

Internally drives are further divided into channels / banks / vaults / arrays (terminology dependent on technology)

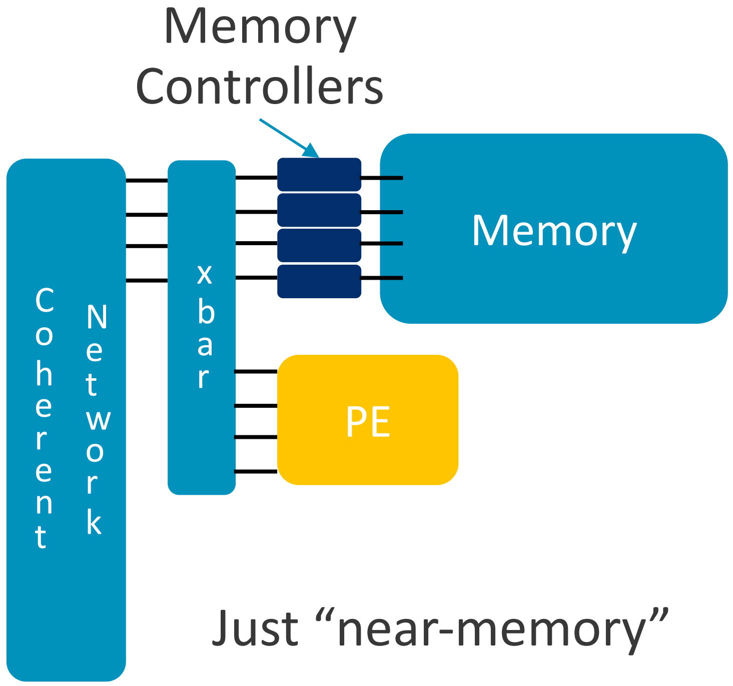
Within the device, the code (VA) must be translatable to a logical block address..doable, just not easy to do w/o more new stuff.

To be useful, you have to initiate potentially hundreds of fine-grained data-targeted tasks…how?

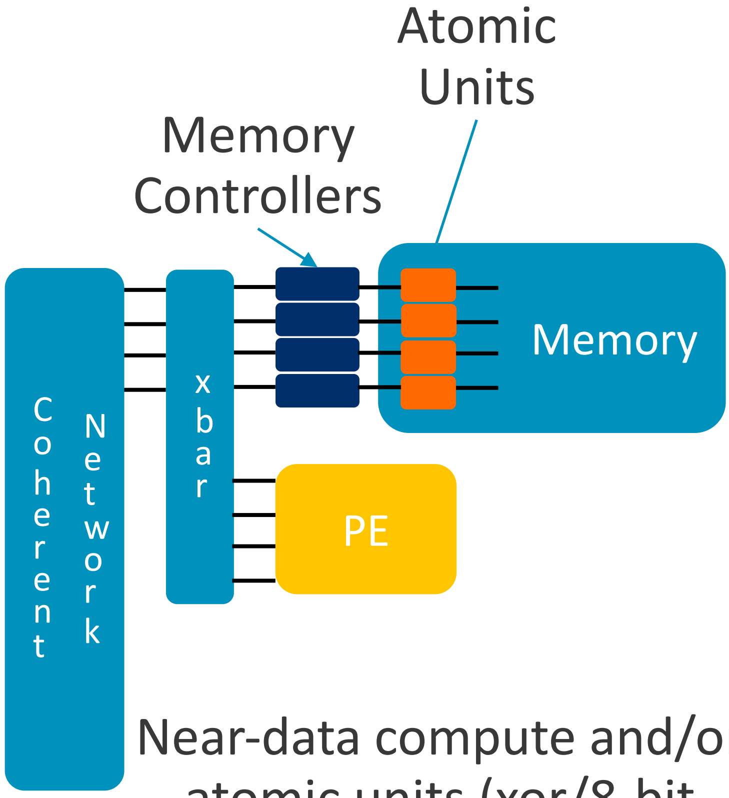
Cache maintenance operations are incredibly costly (in terms of time/cycles/energy)

Types of PINM Accelerators

Near-memory / In-memory / In-NVM



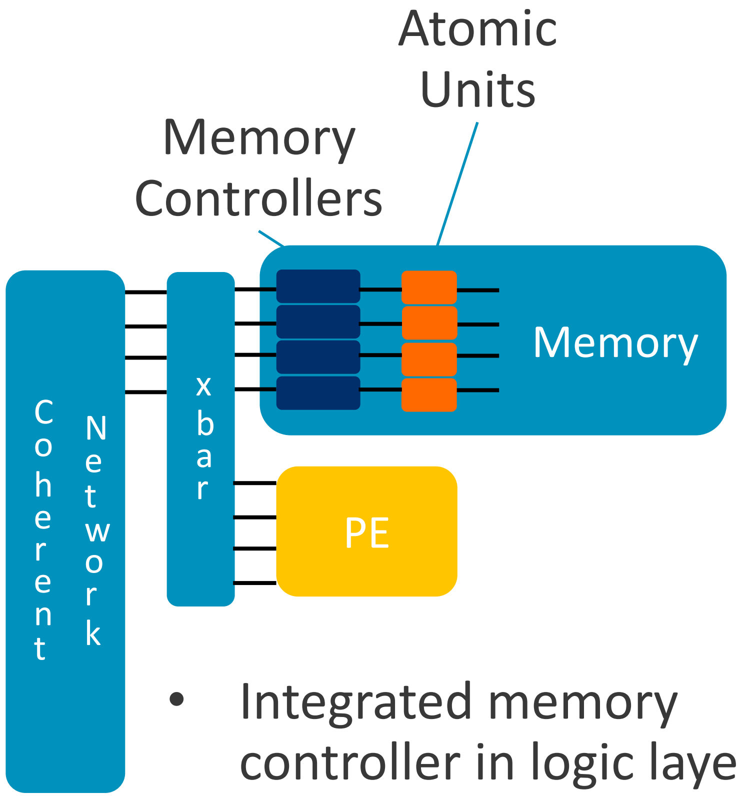
Just “near-memory”



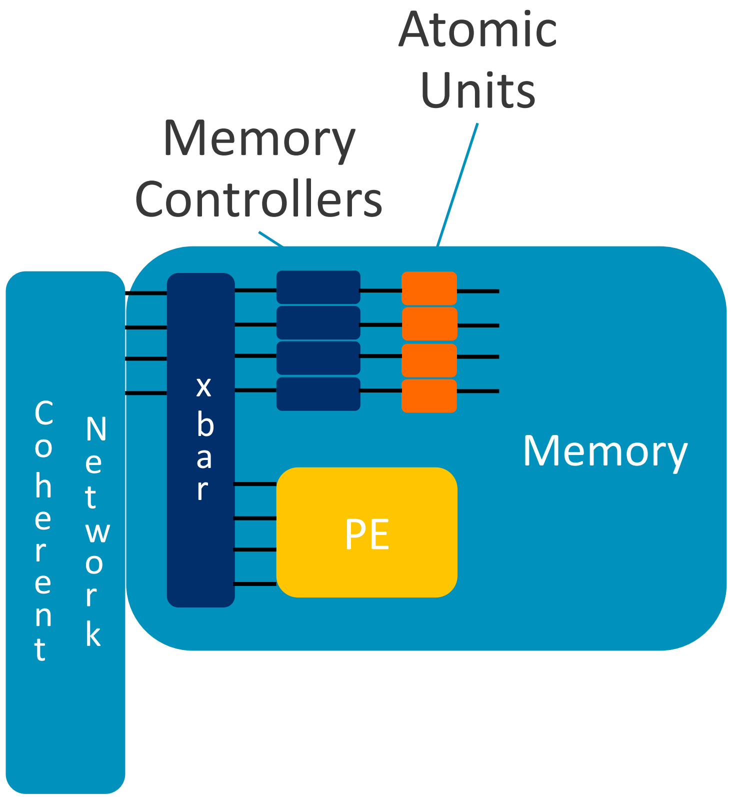
Near-data compute and/or atomic units (xor/8-bit adder) with each channel

Types of PINM Accelerators

Near-memory / In-memory / In-NVM



- Integrated memory controller in logic layer…
- It doesn’t make sense to move controller for this config.



- Integrated xbar + full cores near-memory in logic layer…