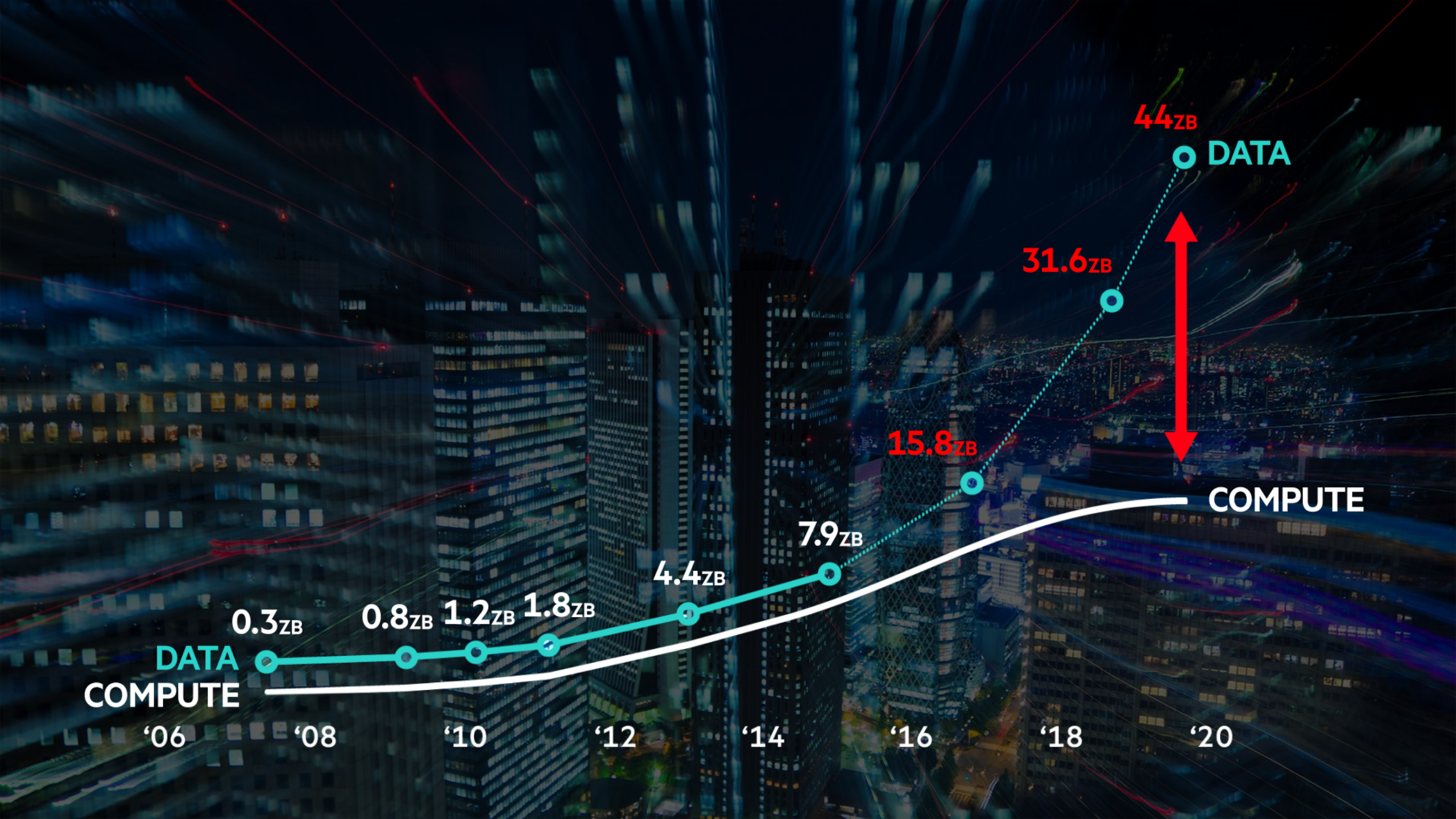
Hewlett Packard Enterprise

Accelerating Machine Learning and Neural Networks: Software for Hybrid Memristor-based Computing

Dejan Milojicic, Distinguished Technologist, Hewlett Packard Labs

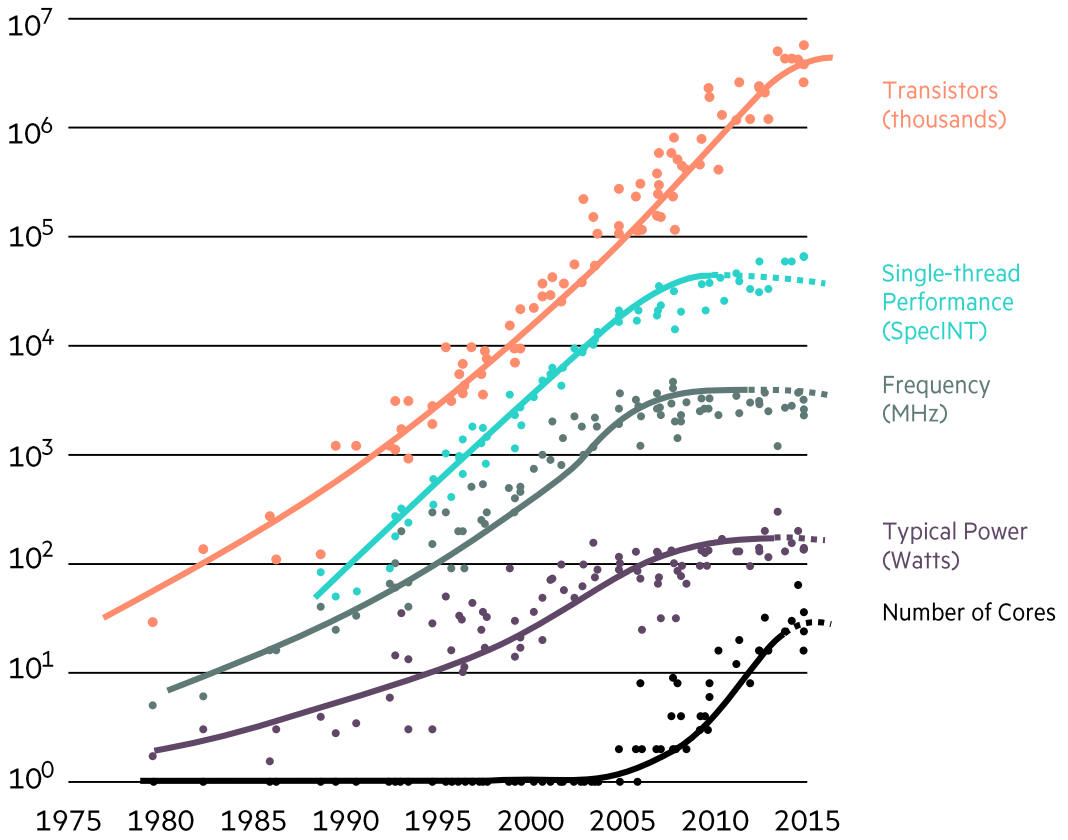
Presentation to IEEE-CNSV, May 14, 2019

Co-authors from Hewlett Packard Labs, Penn State, Purdue, UIUC, USP, GaTech, ETH, ….
Materials from the IEEE ICRC'[16, 17, 18], IEEE ICDCS'18, ACM ASPLOS [16, 19], and more

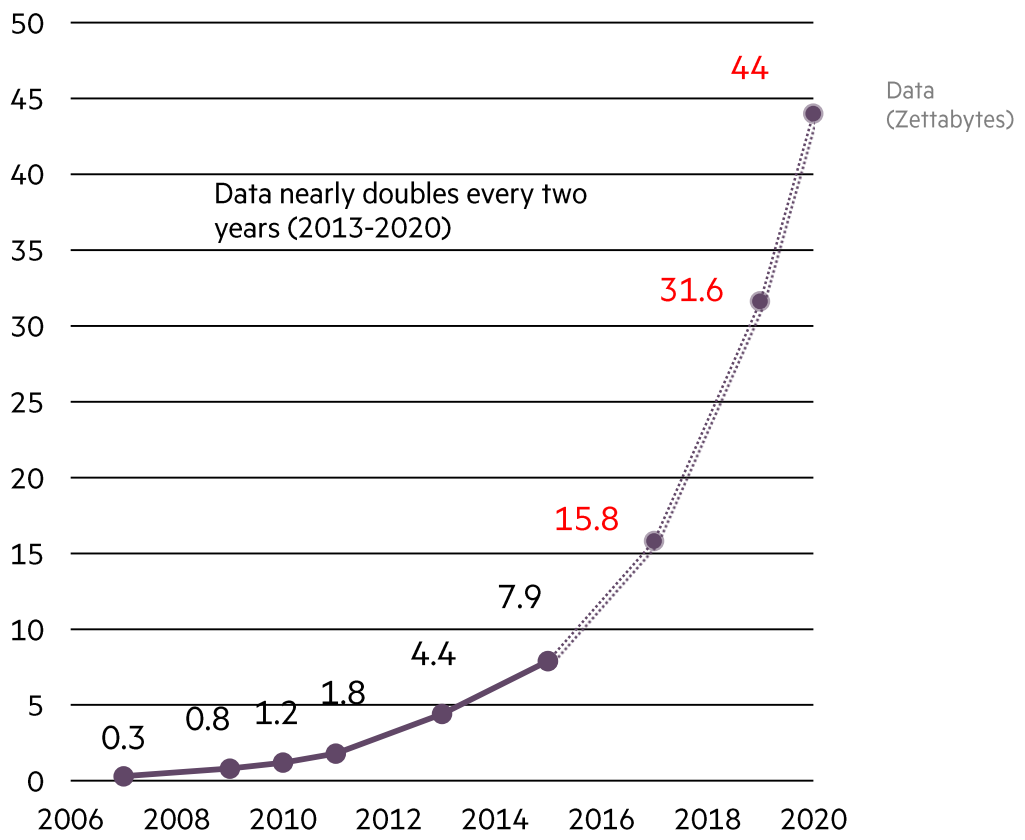

44ZB
DATA
31.6ZB
15.8ZB
COMPUTE
7.9ZB
4.4ZB
0.3ZB
0.8ZB
1.2ZB
1.8ZB
DATA
COMPUTE
‘06
‘08
‘10
‘12
‘14
‘16
‘18
‘20

The New Normal: Compute is not keeping up

Microprocessors

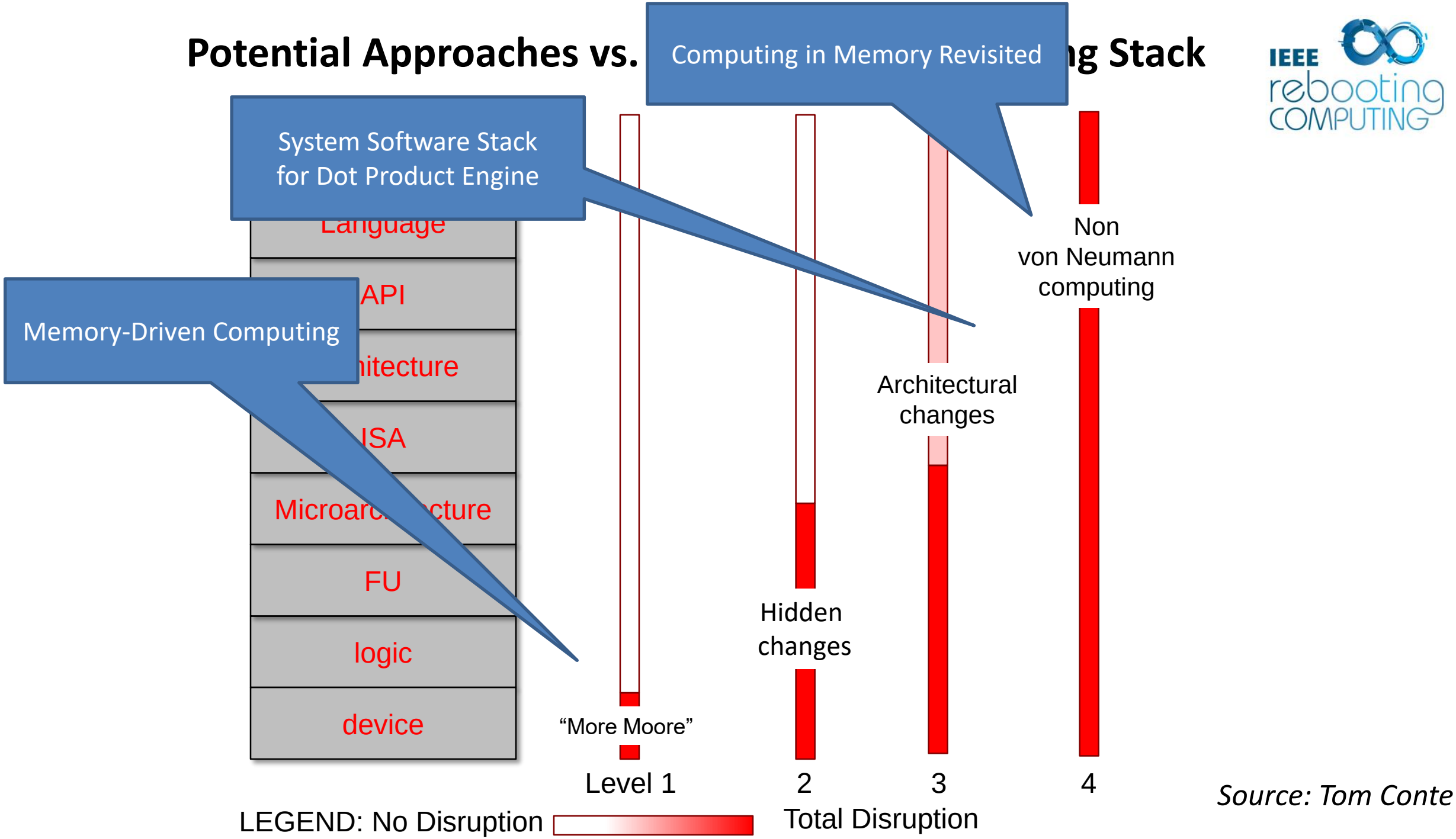


Data growth



Hewlett Packard Enterprise

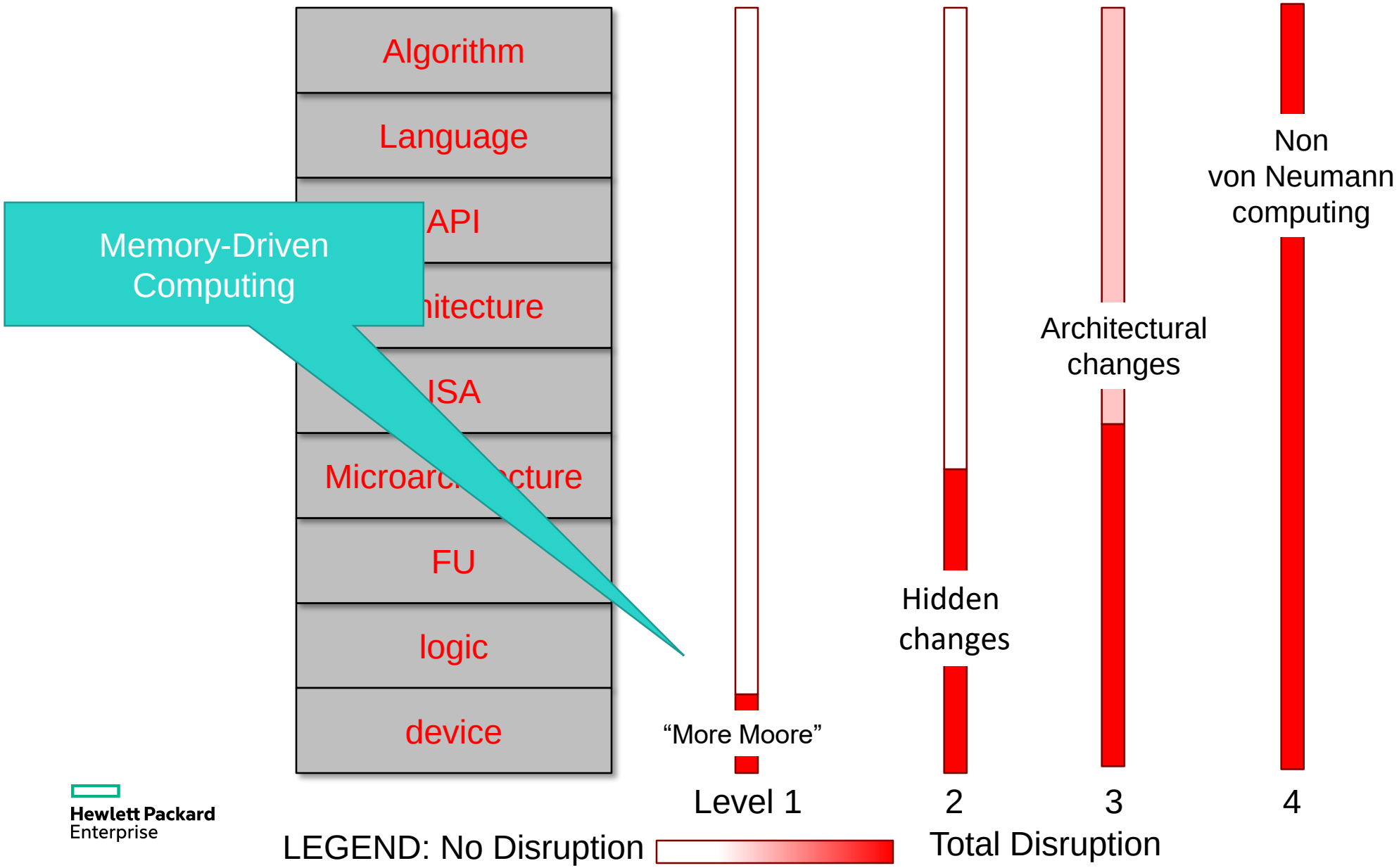
Potential Approaches vs. ng Stack



Source: Tom Conte

Memory-Driven Computing

Potential Approaches vs. Disruption in Computing Stack



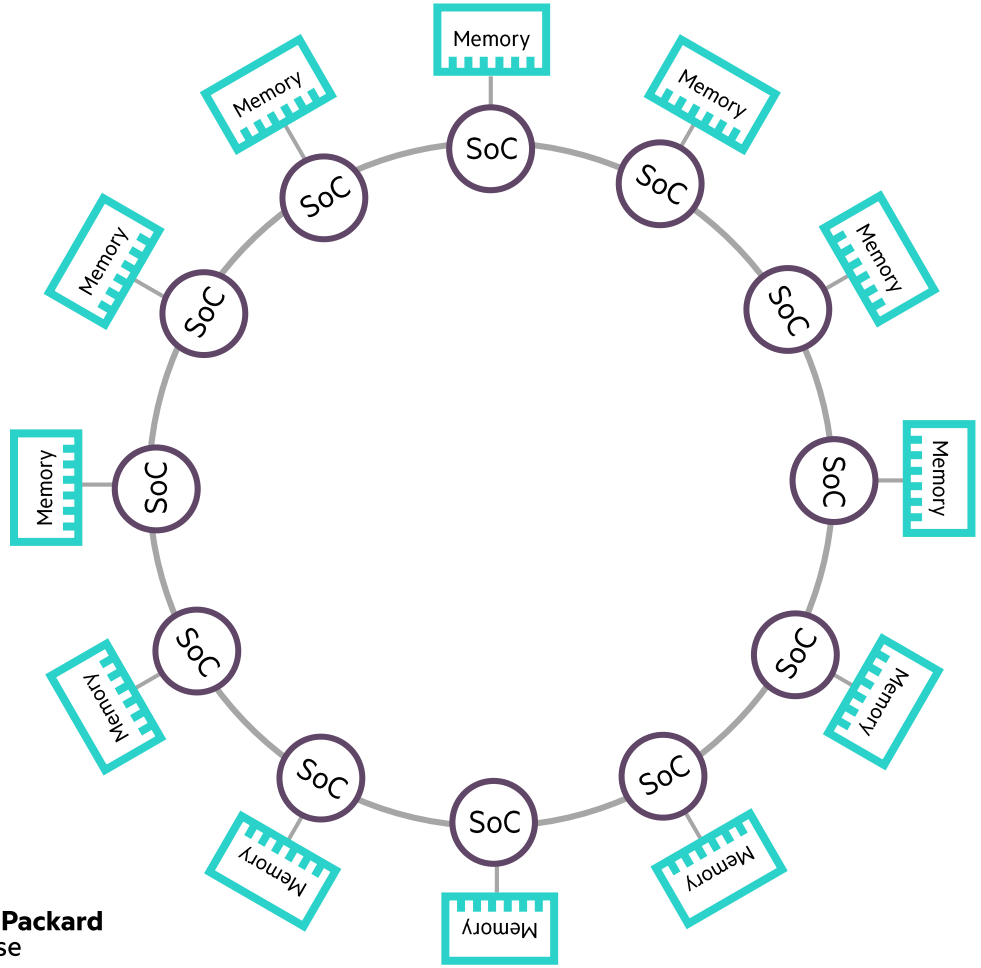
Hewlett Packard Enterprise

Source: Tom Conte

How to improve things while Moore's law has not run out?

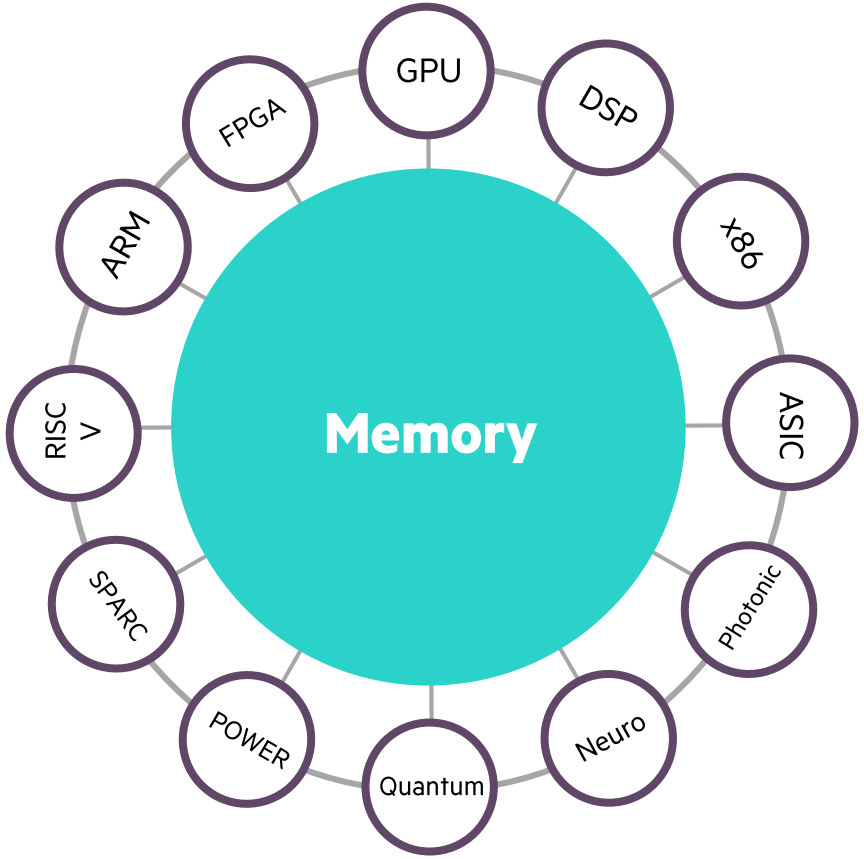
Today's architecture

From processor-centric computing



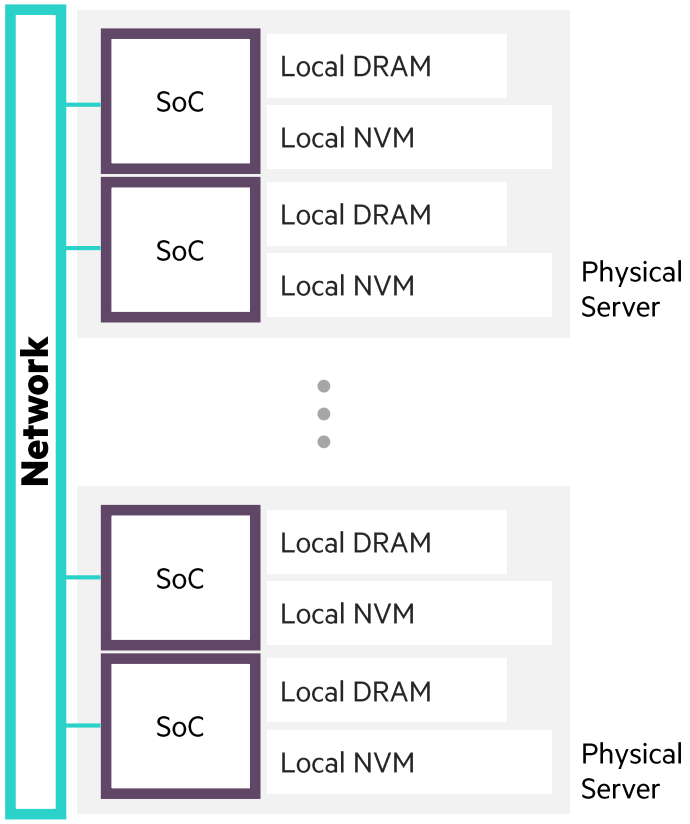
Future architecture

Memory-Driven Computing

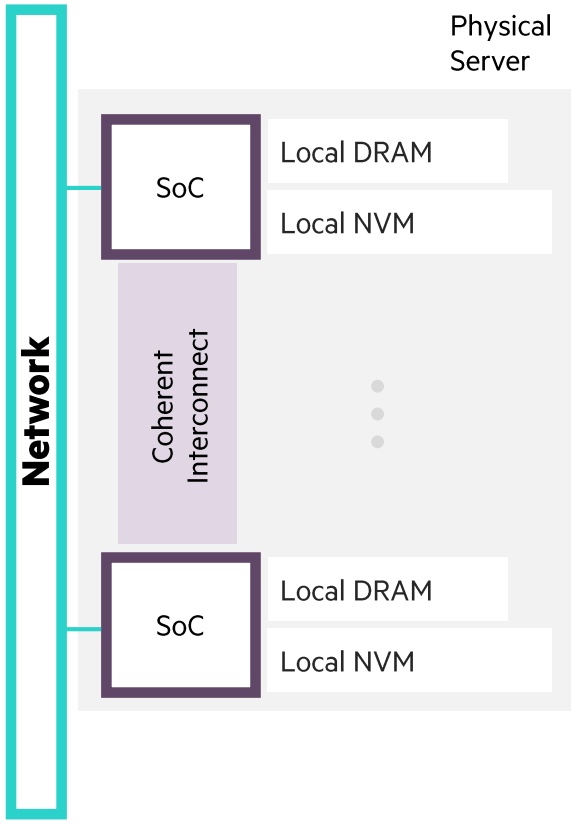


Hewlett Packard Enterprise

Memory-Driven Computing in context



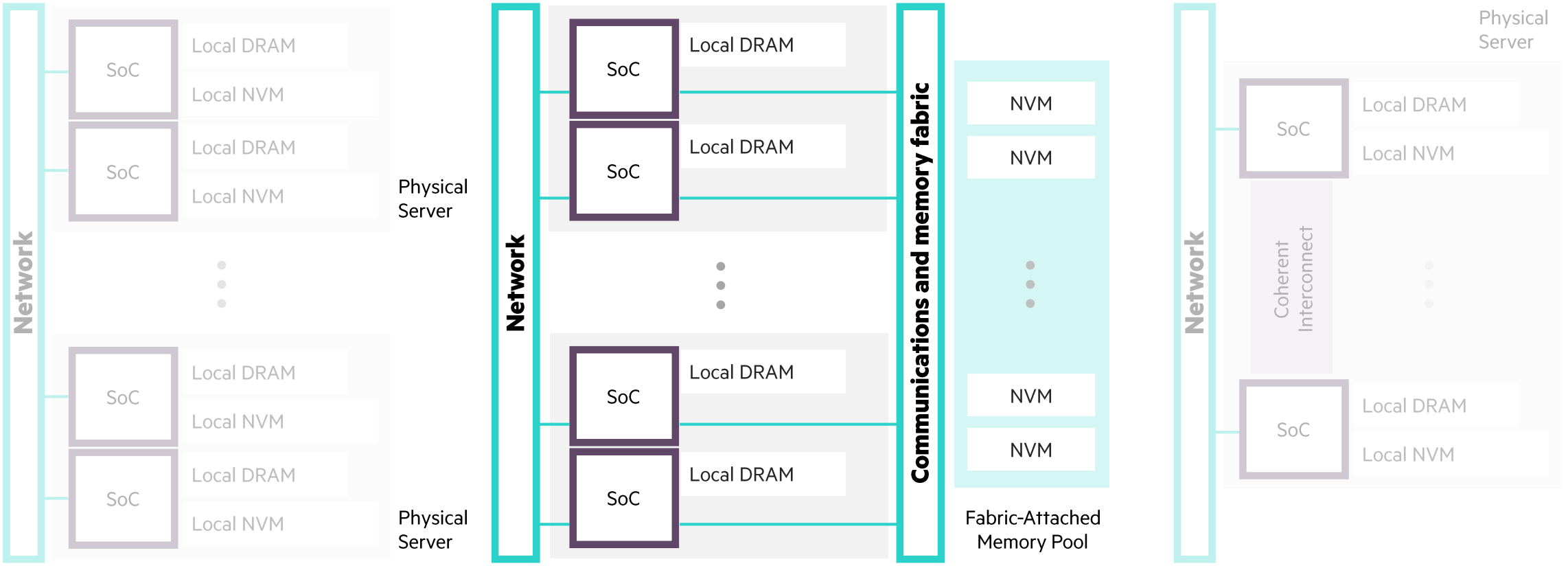
Shared nothing



Shared everything

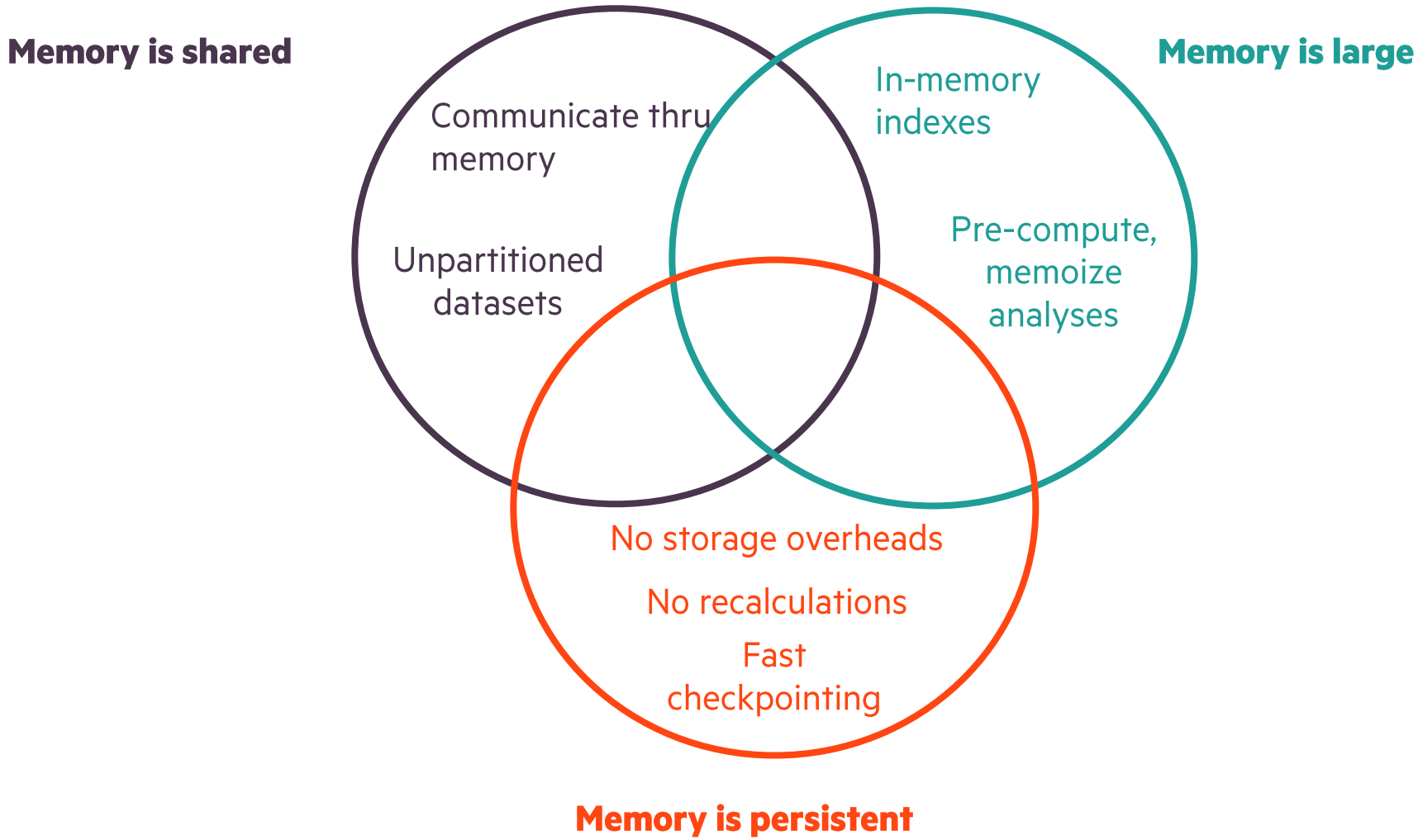
Hewlett Packard Enterprise

Memory-Driven Computing in context



Hewlett Packard Enterprise

Memory-Driven Computing benefits applications



Hewlett Packard Enterprise

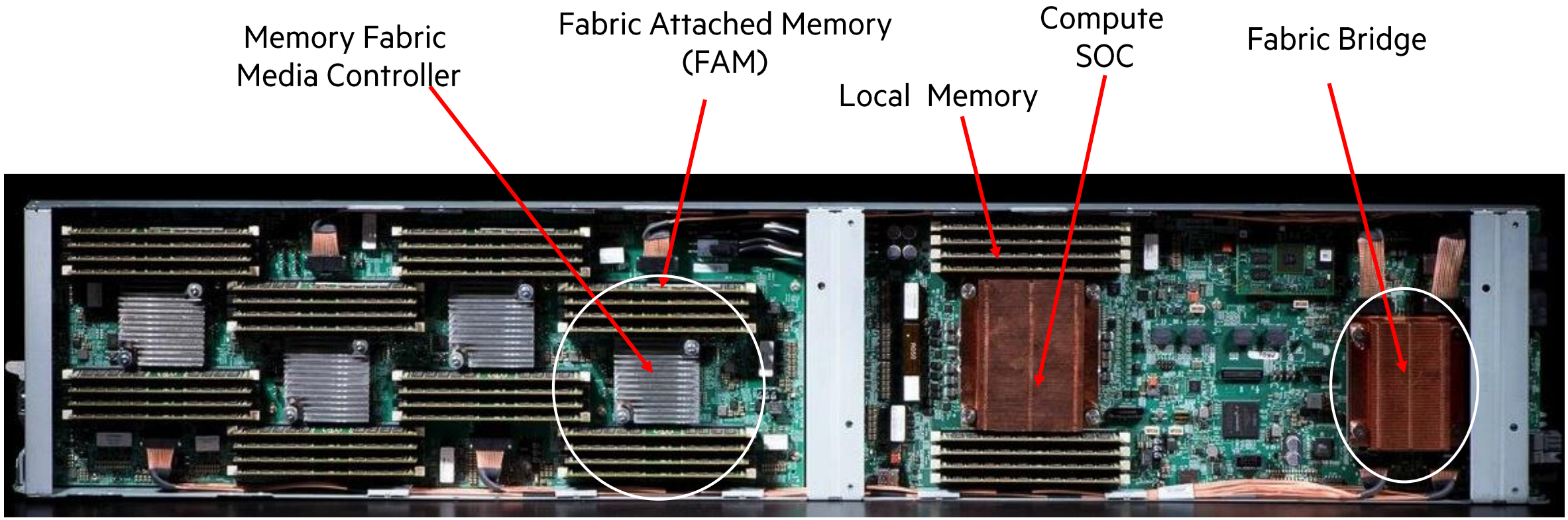
HPE introduced the world's largest single-memory computer

The prototype contains 160 terabytes of memory

- 160 TB of shared memory spread across 40 physical nodes, interconnected using a high-performance fabric protocol.
- An optimized Linux-based operating system running on ThunderX2, Cavium's flagship second generation dual socket capable ARMv8-A workload optimized System on a Chip.
- Photonics/Optical communication links, including the new X1 photonics module, are online and operational.
- Software programming tools designed to take advantage of abundant of persistent memory.

Hewlett Packard Enterprise

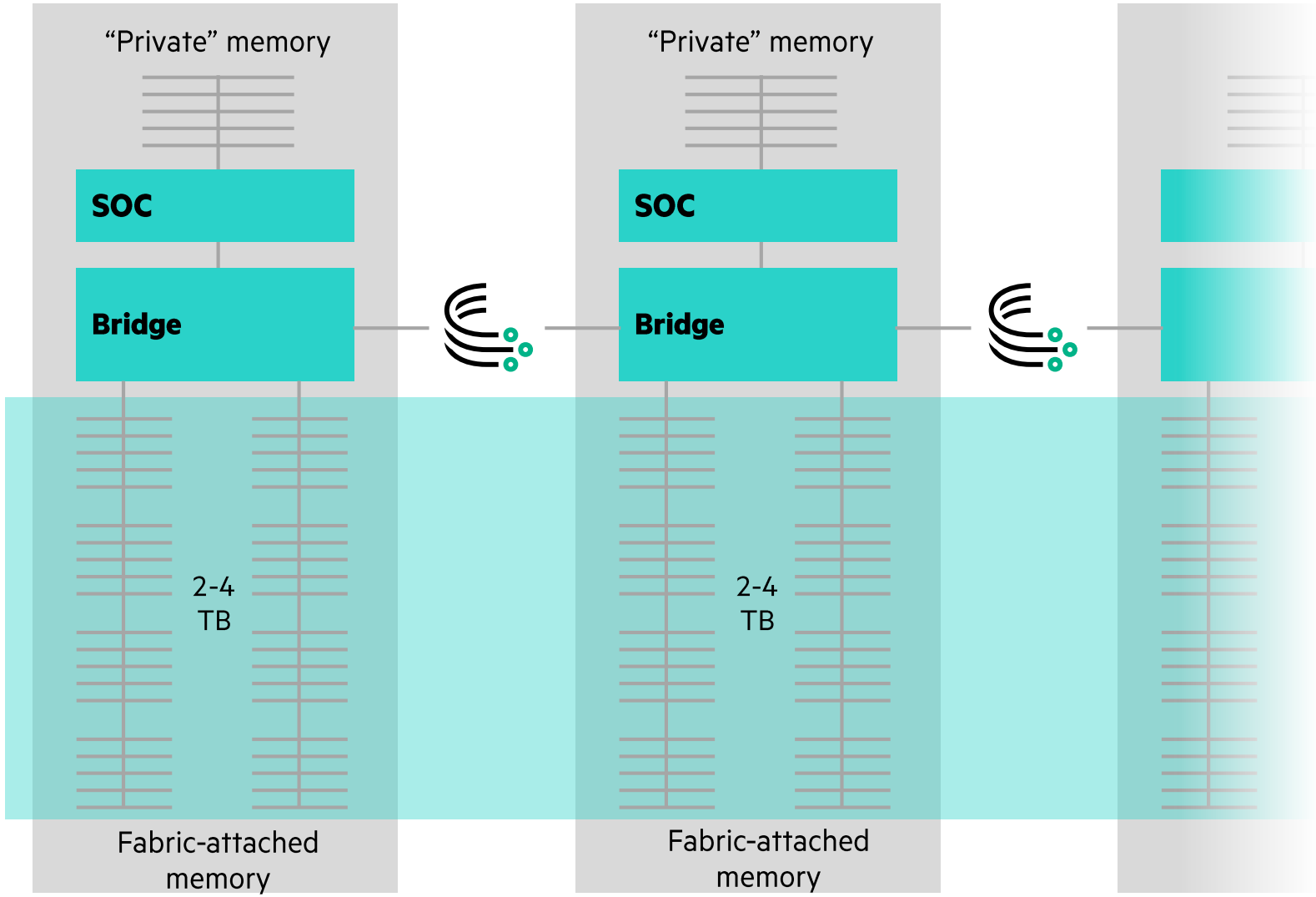
The Machine program: Memory Fabric Testbed



Hewlett Packard
Enterprise

How fabric-attached memory works

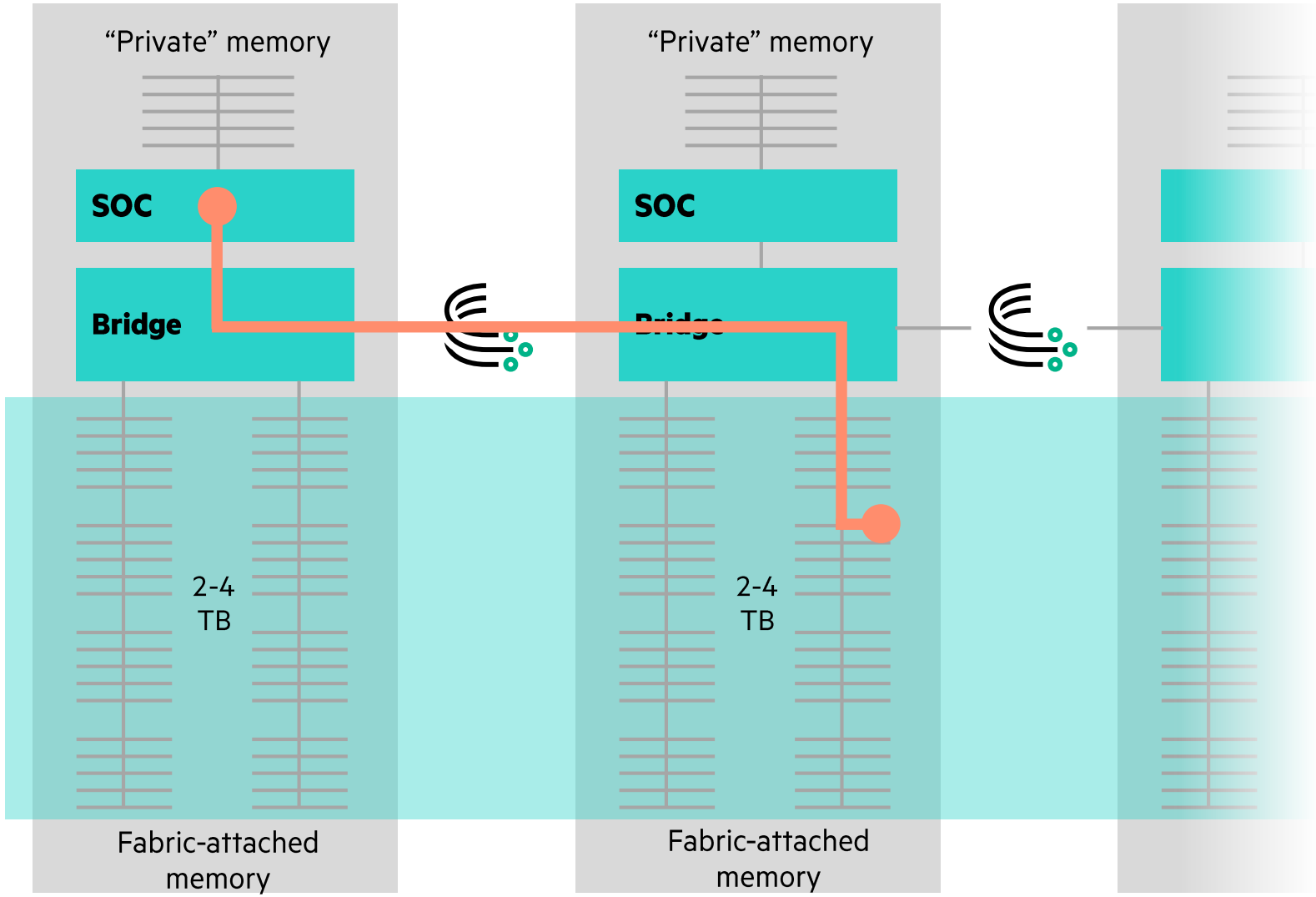
Allows a compute node to access any part of the fabric-attached memory pool



Hewlett Packard Enterprise

How fabric-attached memory works

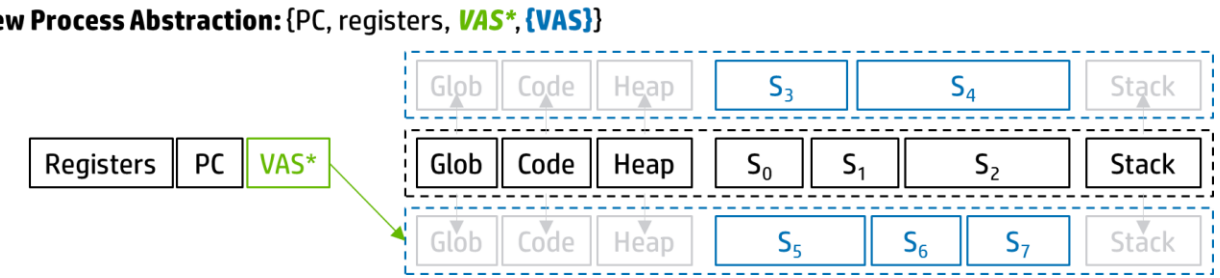
Allows a compute node to access any part of the fabric-attached memory pool



Hewlett Packard
Enterprise

SpaceJMP: Programming with Multiple Virtual Address Spaces

- Virtual address space as first-class citizen
- Process can have multiple virtual address spaces



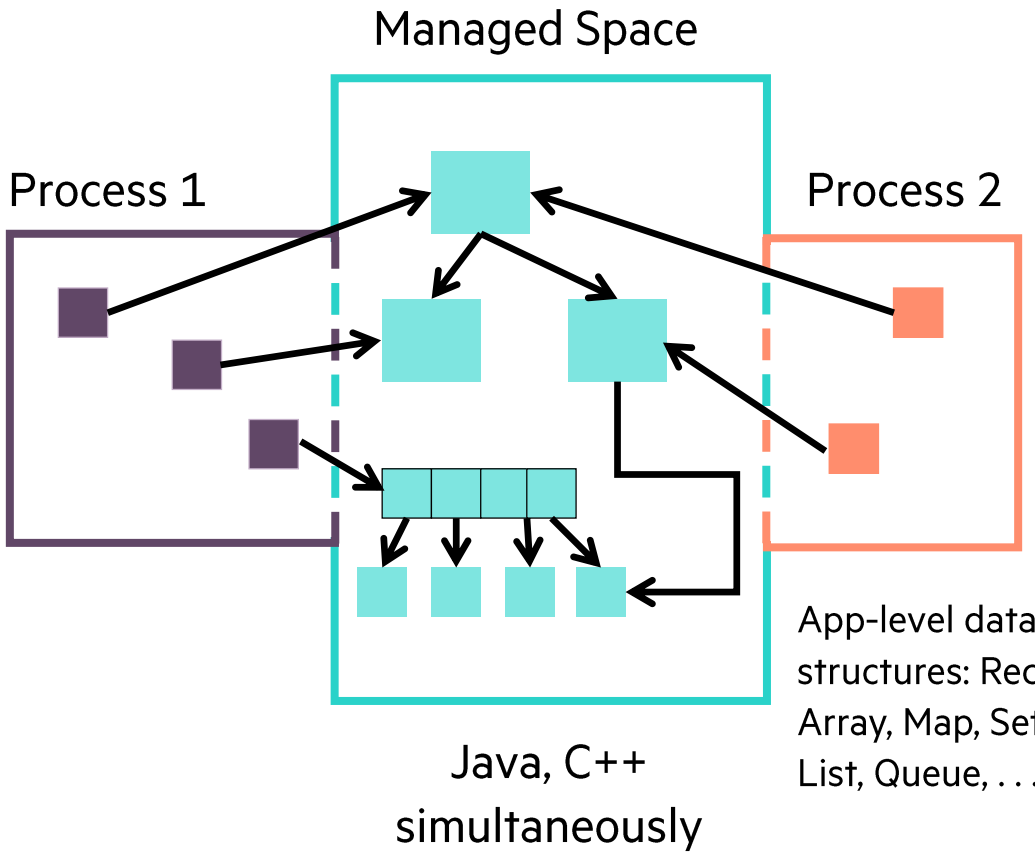
- Efficient safe programming and sharing for huge memories
- Data sharing and communication between processes
- Versioning and checkpointing
- Co-design between OS, programming languages, compilers, and runtimes
- Prototype implementations in BSD, Linux, and Barrelfish

I. El Hajj, et al. "SpaceJMP: Programming with Multiple Virtual Address Spaces," *Proc. Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2016.

Managed Data Structures (MDS)

Simplify programming on persistent in-memory data

- Ease of Programming
  - Programmer manages only application-level data structures
    - MDS data structures are automatically persisted in NVM
  - APIs in multiple programming languages: Java, C++
    - Programmer access through references to data
    - Direct reads and writes
- Ease of Data Sharing
  - Just pass a reference
    - Each program treats the data as if it was local to the program
  - High-level concurrency controls
    - Ensure consistent data in the face of data sharing by multiple threads/processes

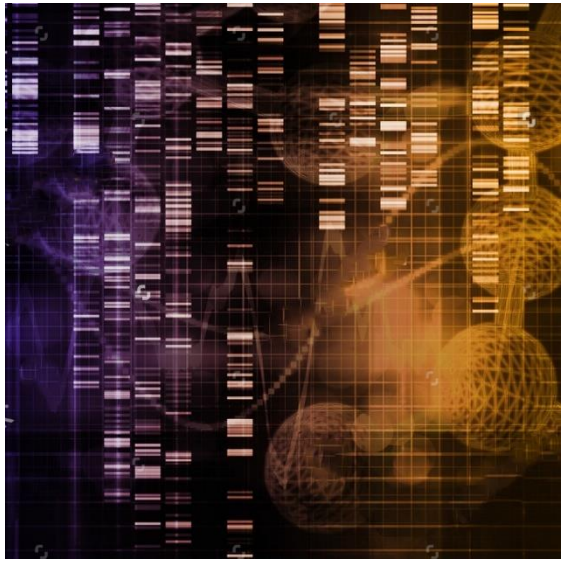


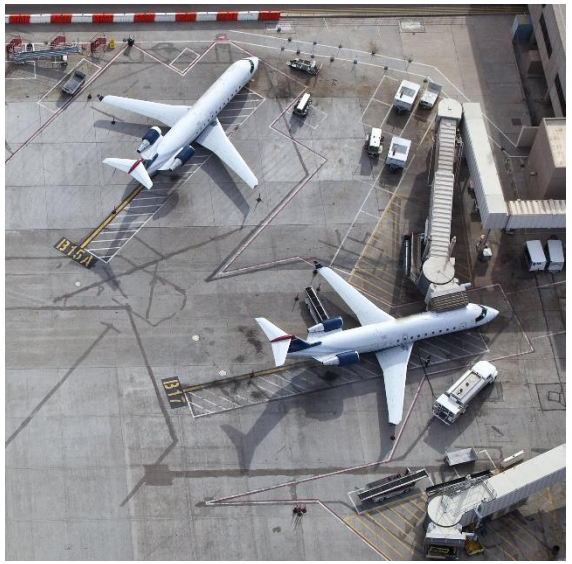
Transform performance with Memory-Driven programming

Modify existing frameworks

New algorithms

Completely rethink

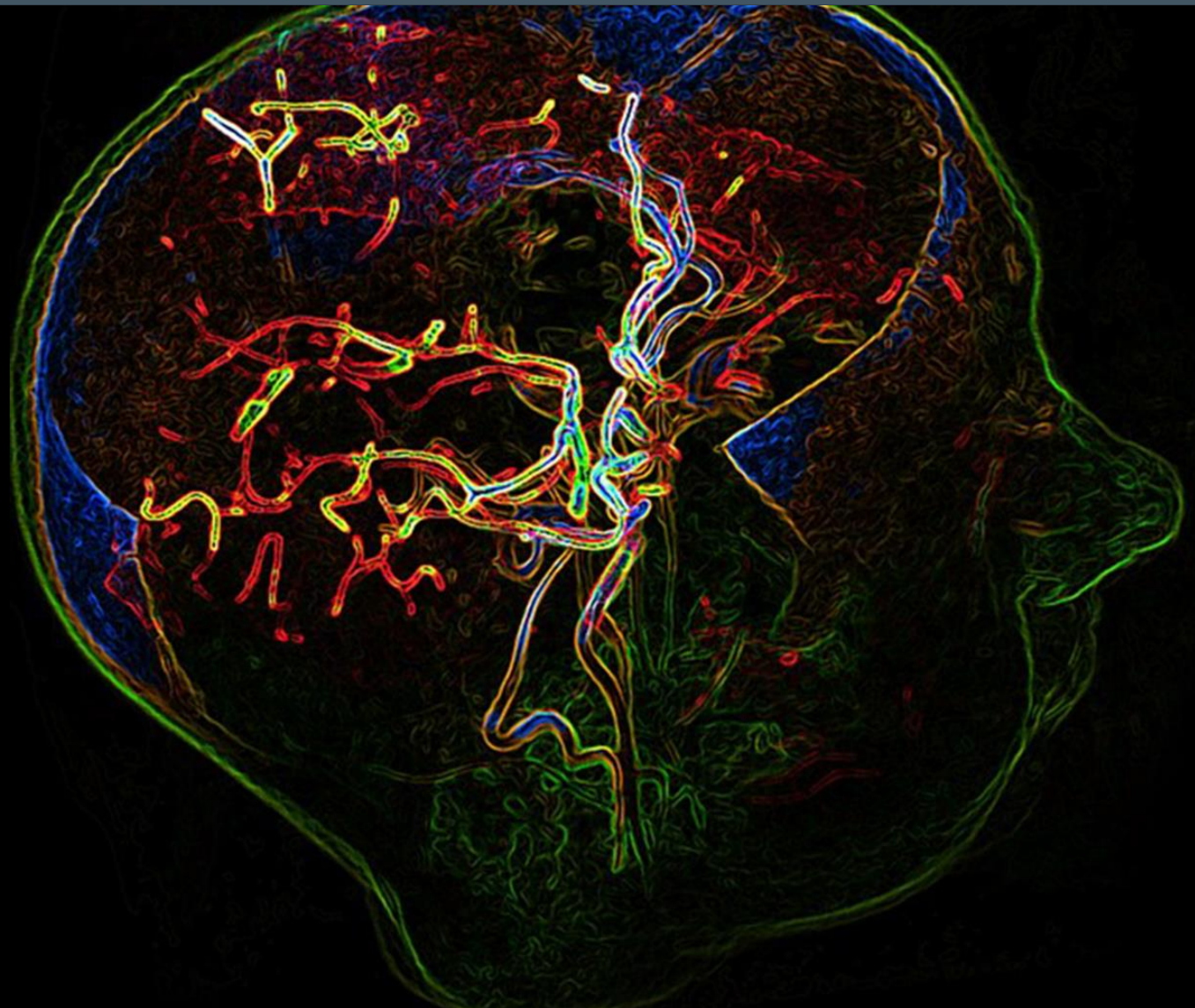




In-memory analytics	Similarity search	Large-scale graph inference	Financial models
15x faster	**40x** faster	**100x** faster	**10,000x** faster

Hewlett Packard Enterprise

Memory-Driven Computing helps outpace the global time bomb of neurodegenerative disease



DZNE
Deutsches Zentrum für
Neurodegenerative Erkrankungen
in der Helmholtz-Gemeinschaft

DZNE discovered HPE's Memory-Driven Computing — and saw unprecedented computational speed improvements that hold new promise in the race against Alzheimer's

60%
power reduction
cuts research costs

101x
increase in analytics speed
blasts research bottlenecks,
leading to shorter processing
time —
from 22 minutes to **13** seconds

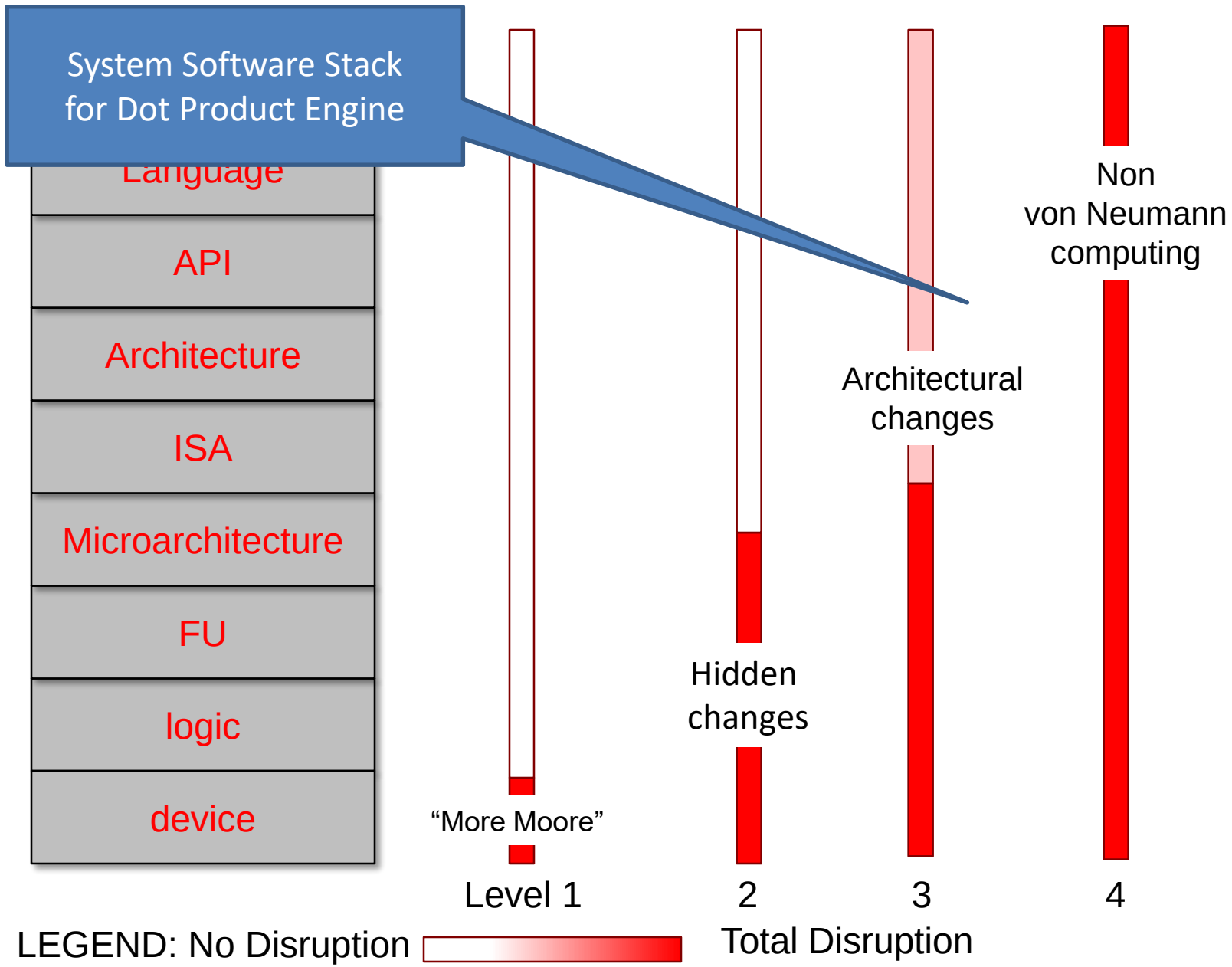
Hewlett Packard Enterprise

Generalize or Die:
ISA and Compiler for Memristor Accelerators

Aayush Ankit, Pedro Bruel, Sai Rahul Chalamalasetti, Izzat El Hajj,
Cat Graves, Dejan Milojicic, Geoffrey Ndu, John Paul Strachan

ACM ASPLOS 2019, IEEE ICRC 2018

Potential Approaches vs. Disruption in Computing Stack



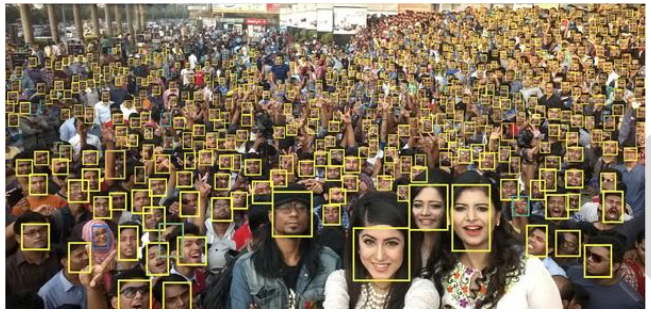
Source: Tom Conte

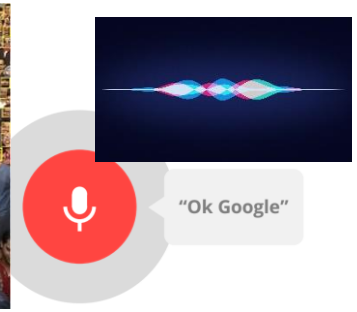
Outline

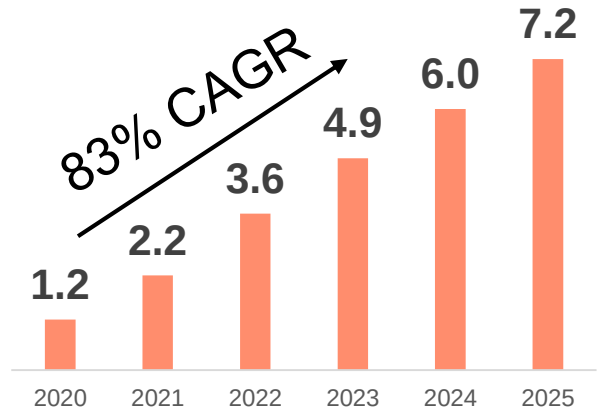
- Introduction to Dot Product Engine (DPE) Memristor Accelerator

  Architecture and laboratory demonstrations
- New Instruction Set Architecture (ISA) to facilitate development
- Microarchitecture
- Compiler
- Performance Benchmarks
- Business Opportunity and Next Steps
- Summary

Intelligent Edge Opportunities for Machine Learning / AI







Total Addressable Market – Edge Accelerator (**$Billions**)

- Machine learning is part of real products, real revenue, real growth
- Over $30 Billion invested annually
  - Reflects software and hardware investments by startups, established players (Google, Apple, Microsoft, Nvidia, etc.) and countries (China)
- HPE Labs has significant R&D history and IP in machine learning and cognitive systems
- Current and past programs supported by US government
- Task-specific accelerators a growing opportunity to enable new applications at the Edge

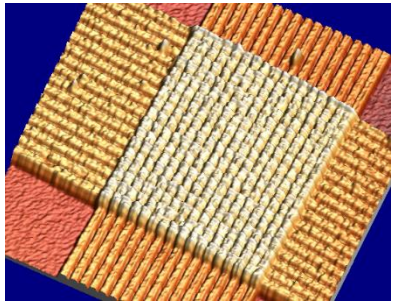
Beyond Moore's Law

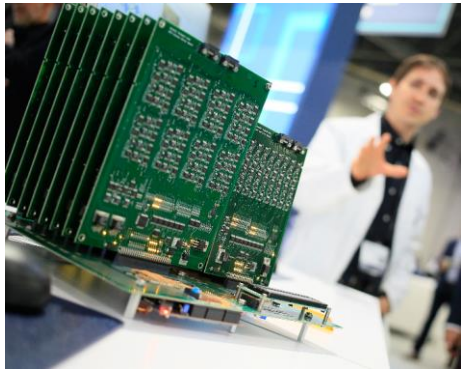
Further into the future: unconventional accelerators

Neuromorphic computing:

Dedicated hardware for brainlike computing

- Neuromorphic computing can **quickly handle tasks that take trained computers several hours**
- **Dot Product Engine is our testbed using vector-matrix multiplication** and studying which algorithms and applications benefit the most from using this speedup architecture

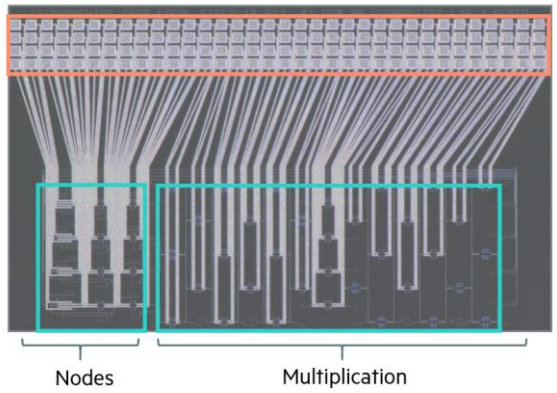


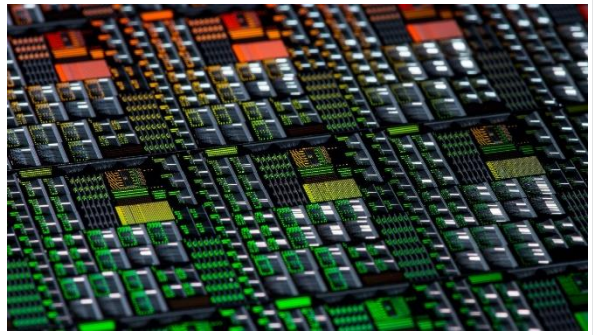


Optical Computing:

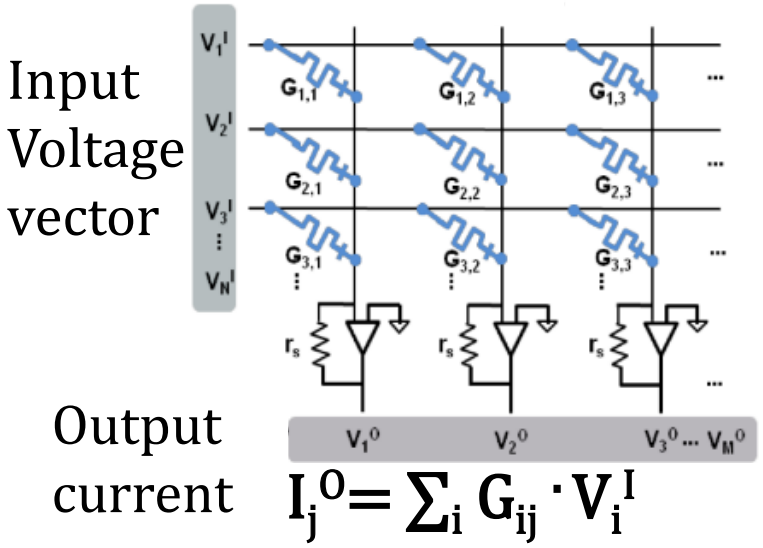
Computing at the speed of light

- Pushing limits of photonic chip design
- Pushing complex computations through light to boost speed and save energy
- Typical circuits are <10 components. **We're integrating over 1,000 optical parts in a chip** – the largest photonic components working together to compute.



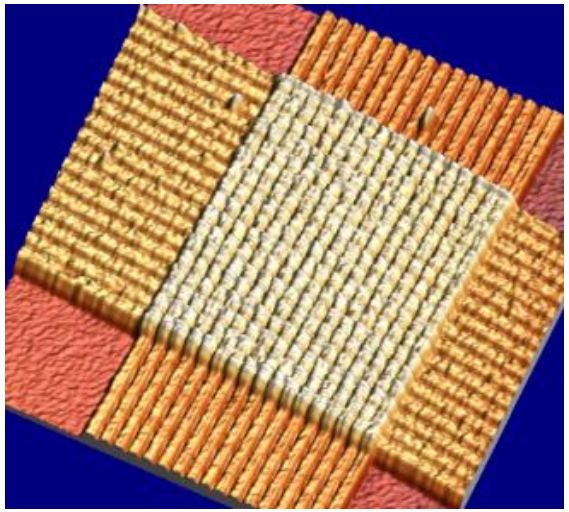


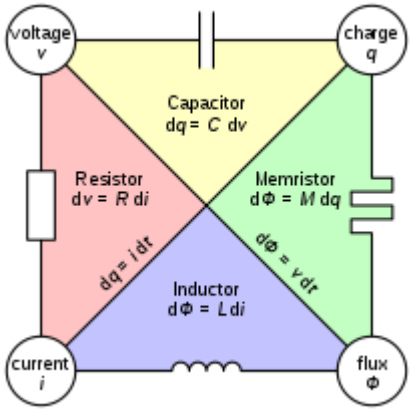
Dot Product Engine (DPE): memristor-based accelerator



- Perform vector-matrix multiplication (VMM) in a single cycle
- Memristors = nanoscale, non-volatile memory cells
- High data density; each memristor can be 1-8 bits
- Allows efficient multiplication in analog domain
- Key advantage is in-memory processing
- Many pathways for future scaling

  Array size, Memristor levels, lower current, 3D layers, DAC/ADC

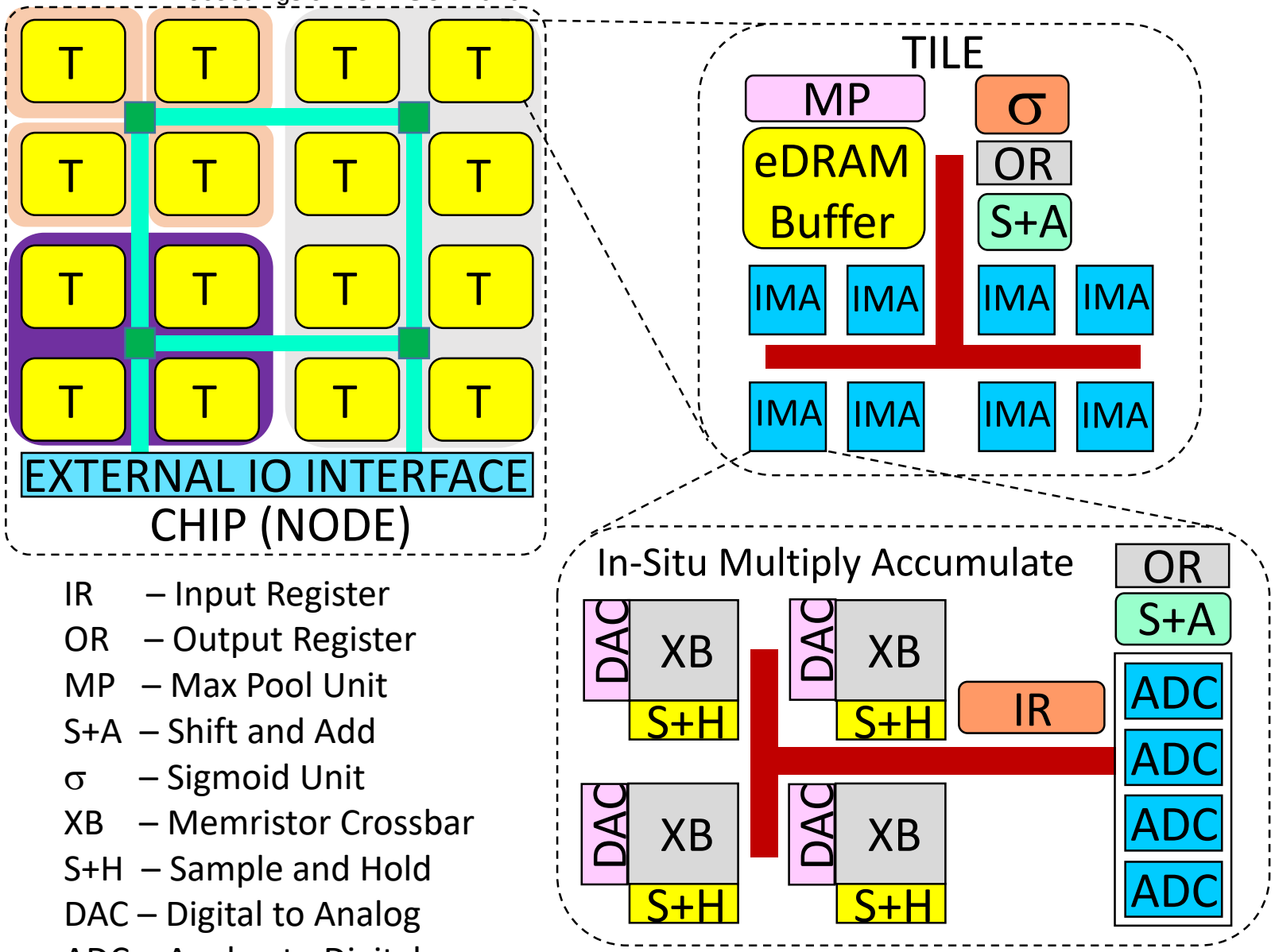




Hewlett Packard Enterprise

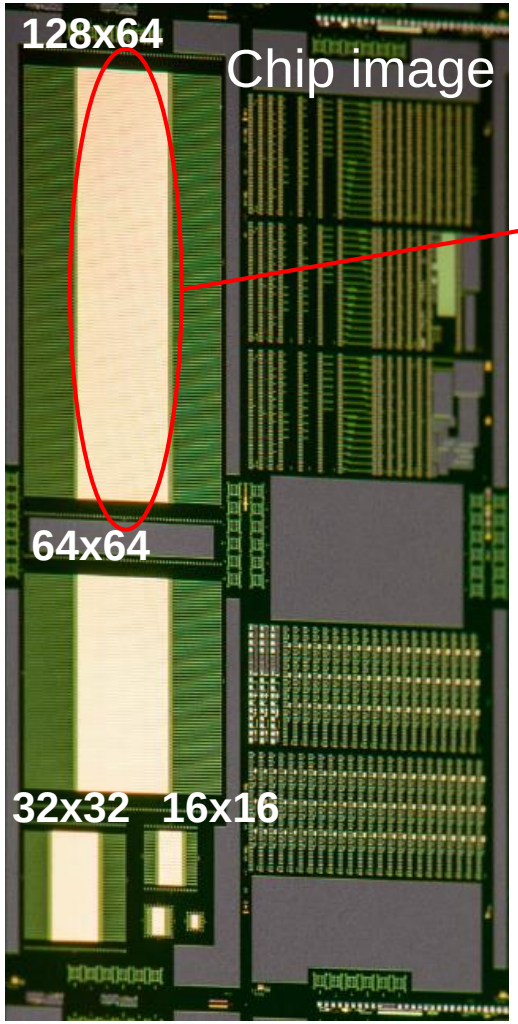
DPE, continued

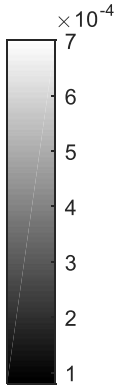
Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R. Stanley Williams, and Vivek Srikumar. 2016. ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars. Proceedings of ACM ISCA 2016



IR – Input Register
OR – Output Register
MP – Max Pool Unit
S+A – Shift and Add
σ – Sigmoid Unit
XB – Memristor Crossbar
S+H – Sample and Hold
DAC – Digital to Analog
ADC – Analog to Digital

Dot Product Engine: lab prototype



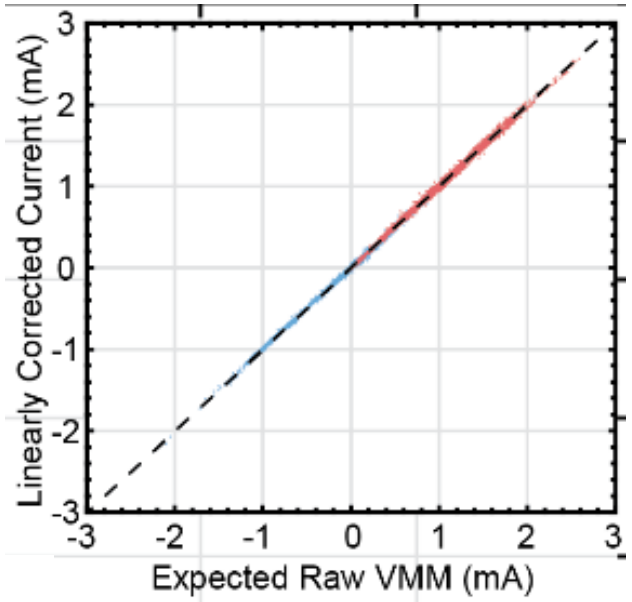


Each grayscale pixel is an analog memristor
>7 bits per memristor cell

Custom control circuitry for in-memory computing

SDL designing and taping-out integrated memristor-control system

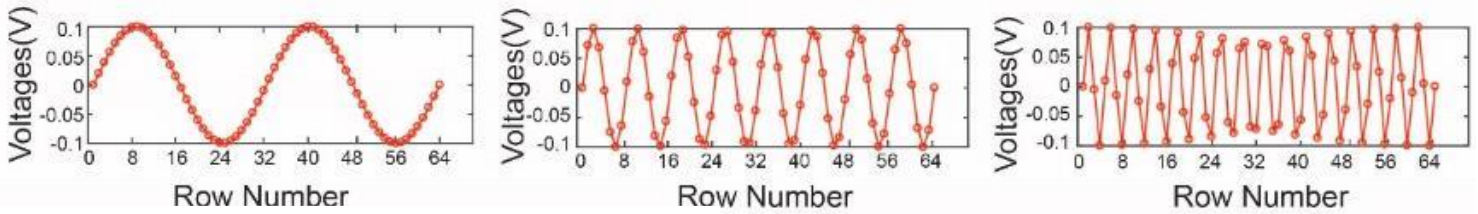
Experimental demonstrations of analog vector-matrix multiplications (VMM) in memristor chip

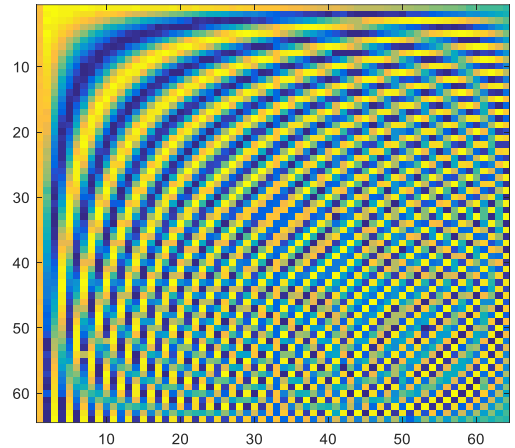


Hewlett Packard Enterprise

Dot Product Engine – signal processing

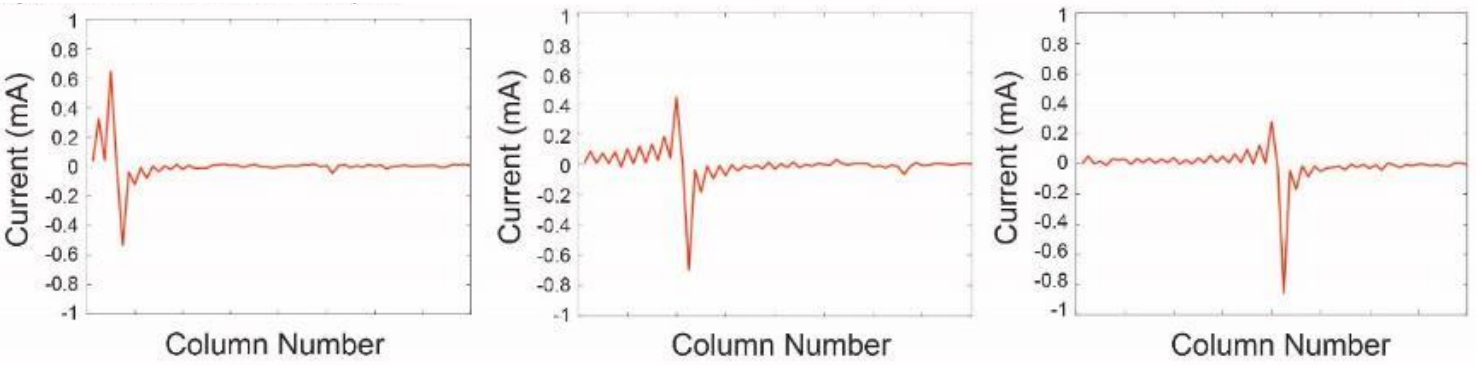
Time-domain inputs (applied to rows)



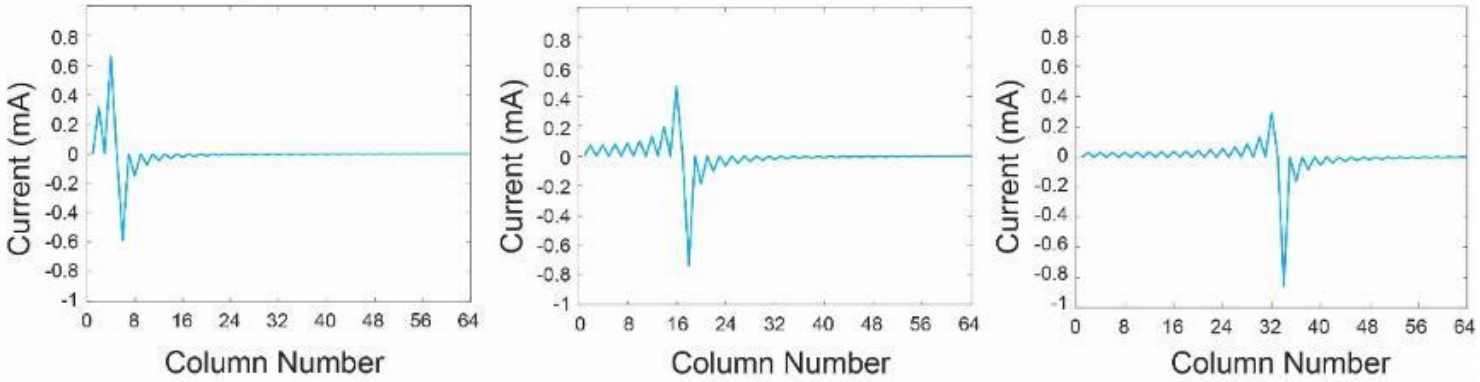


Freq-domain (DCT) outputs

Experimental



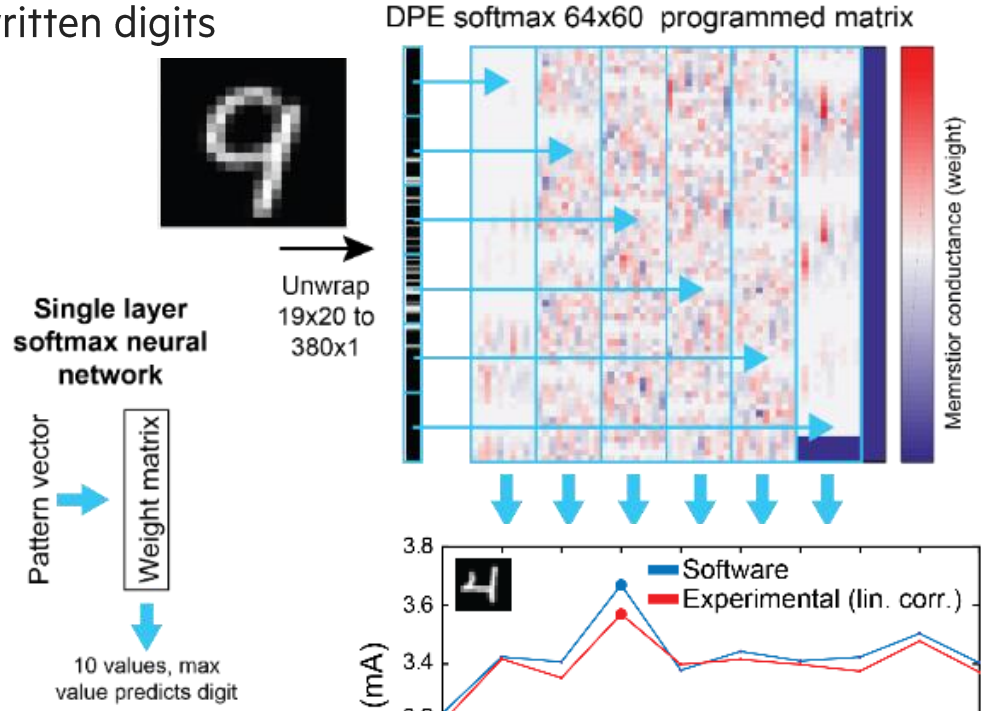
Software

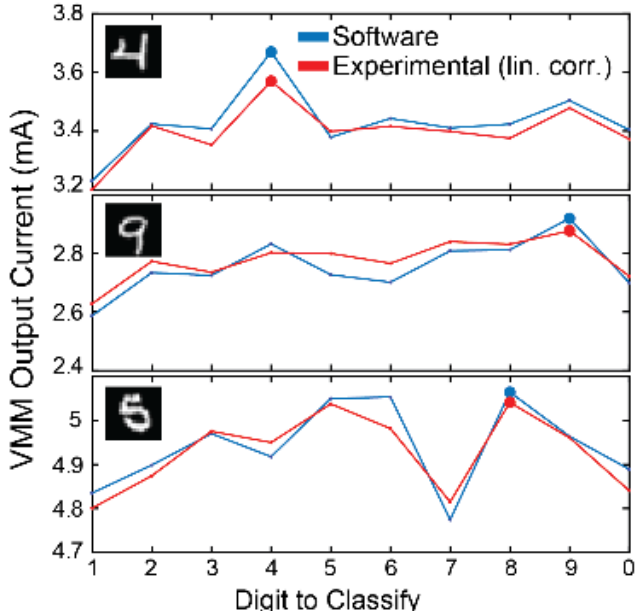


C. Li, et. al, *Nature Electronics*, (2018)

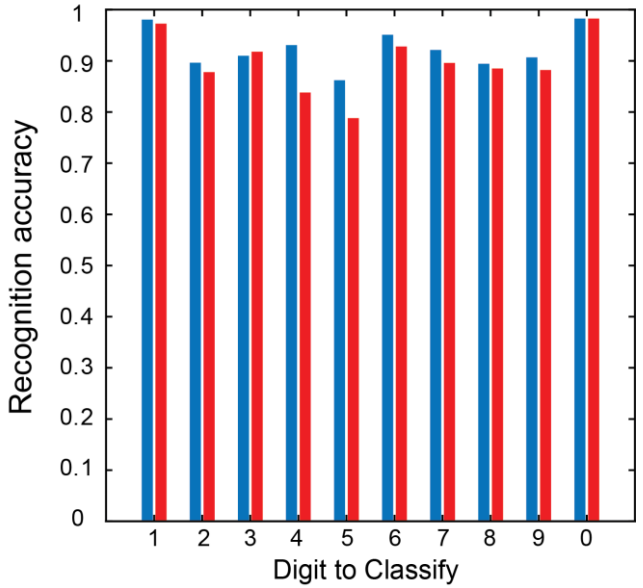
Dot Product Engine – single-layer neural network, MNIST

MNIST database of handwritten digits





Successful Neural Network inference with memristor-based analog computing



M. Hu, et. al, Adv. Mater. *2018*

Challenges: Generalize or Die

– Need to broaden application space for the DPE

– Need to reduce software development burden for customers

Solutions

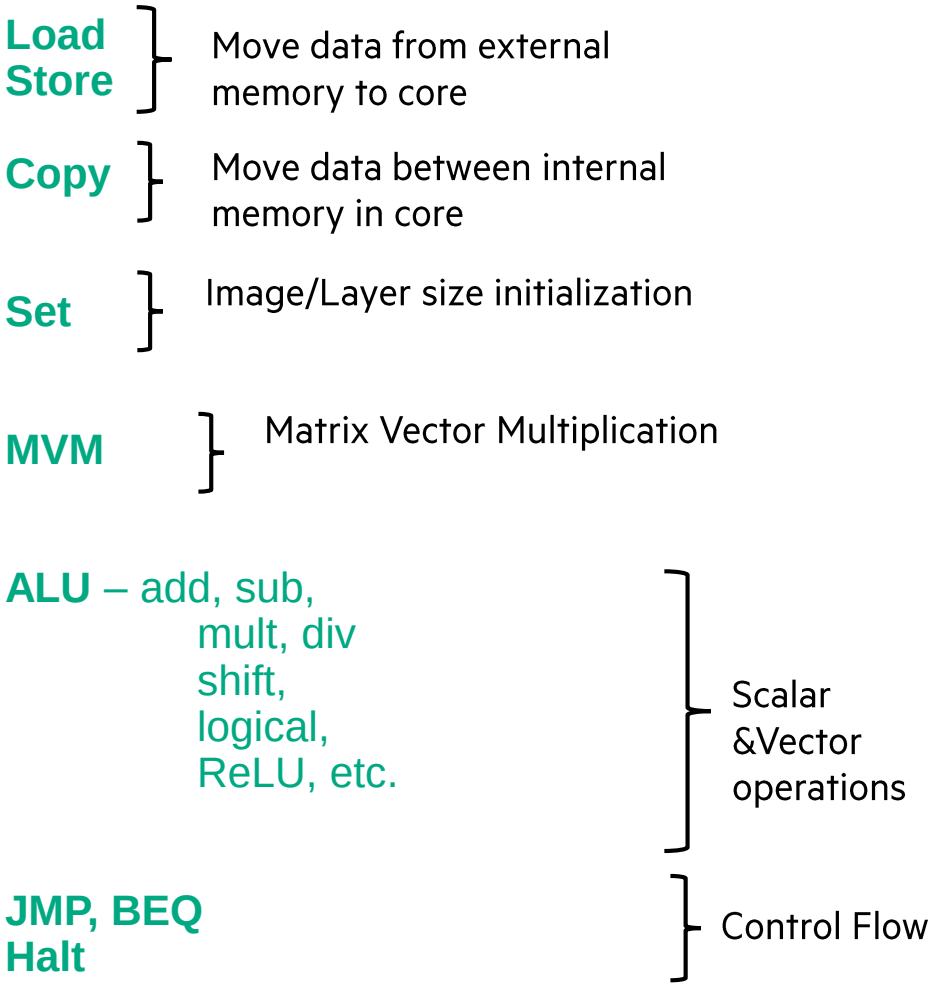
– Developed an **Instruction Set Architecture (ISA)** supporting state-of-the-art neural network types

– Developed a high performance **microarchitecture** implementing ISA

– Developed **compiler** to add programming support and increased functionality

– Benchmarked **performance** of the resulting end-to-end solution

Instruction Set Architecture (ISA)



ISA supports key intelligent edge applications:

Computer vision: Convolutional Neural Networks (CNN)

Speech Recognition: Recurrent Neural Networks (RNN) especially Long Short-Term Memory (LSTM)

Video Recognition: CNN + RNN

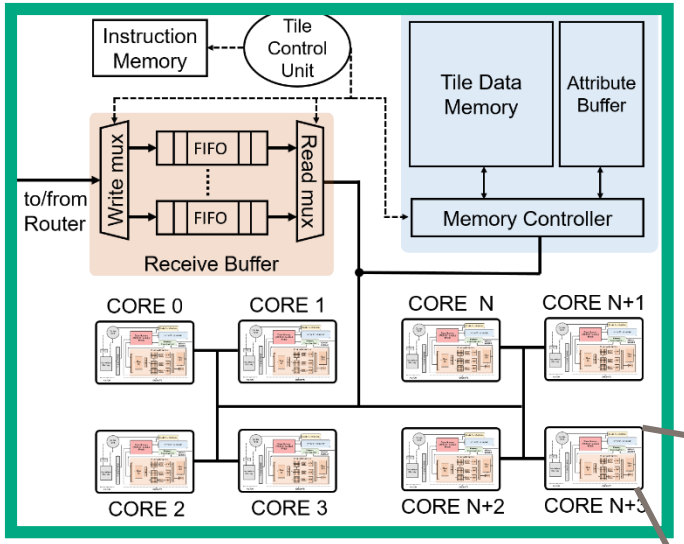
Object Detection: Multi-Layer Perceptrons (MLP)

Bio-inspired Neuromorphic systems: Spiking Neural Networks (SNN)

Additionally, Boltzmann Machines (BM) and Restricted Boltzmann Machines (RBM).

Implementing the ISA: the “PUMA” microarchitecture

PUMA = Programmable Ultra-efficient Memristor-based Accelerator



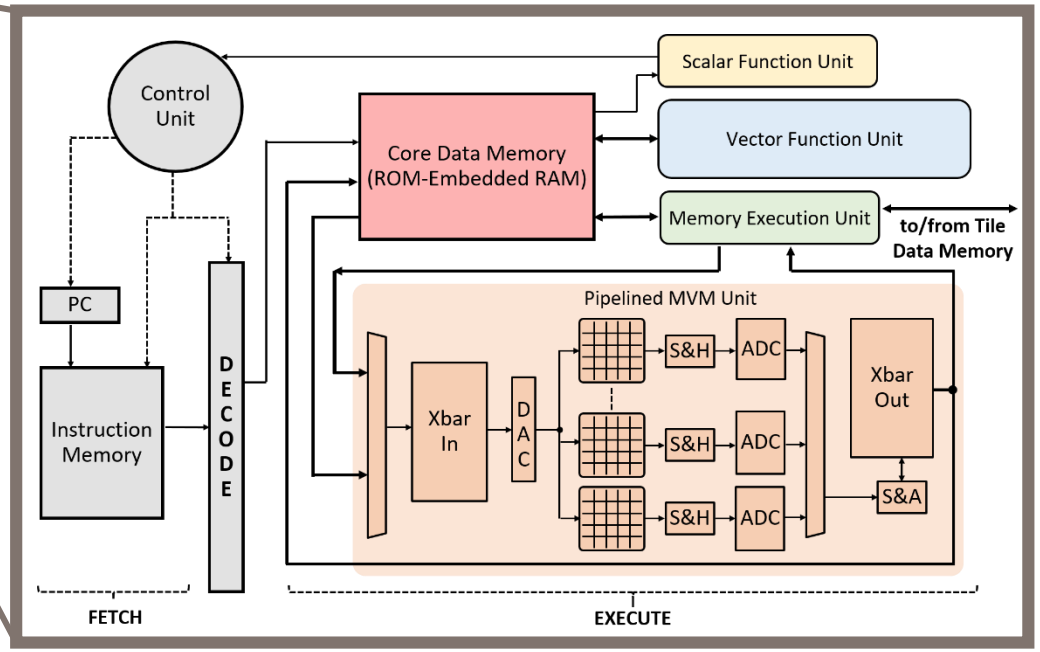
Tile

Each core contains multiple memristor xbars performing matrix vector multiplications.

Also supports simpler digital operations (logical, scalar, and vector units).

3-stage pipeline, instruction decoder, and instruction memory.

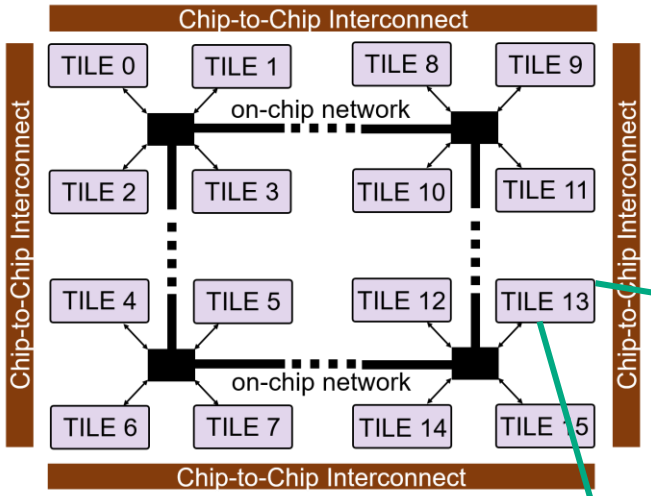
Core



Aayush Ankit, Izzat El Hajj, Sai Rahul Chalamalasetti, Geoffrey Ndu, Martin Foltin, R Stanley Williams, Paolo Faraboschi, Wen-mei W Hwu, John Paul Strachan, Kaushik Roy, Dejan S Milojicic
Proceedings of the Twenty-Fourth APLOS, 2019.

Implementing the ISA: the “PUMA” microarchitecture

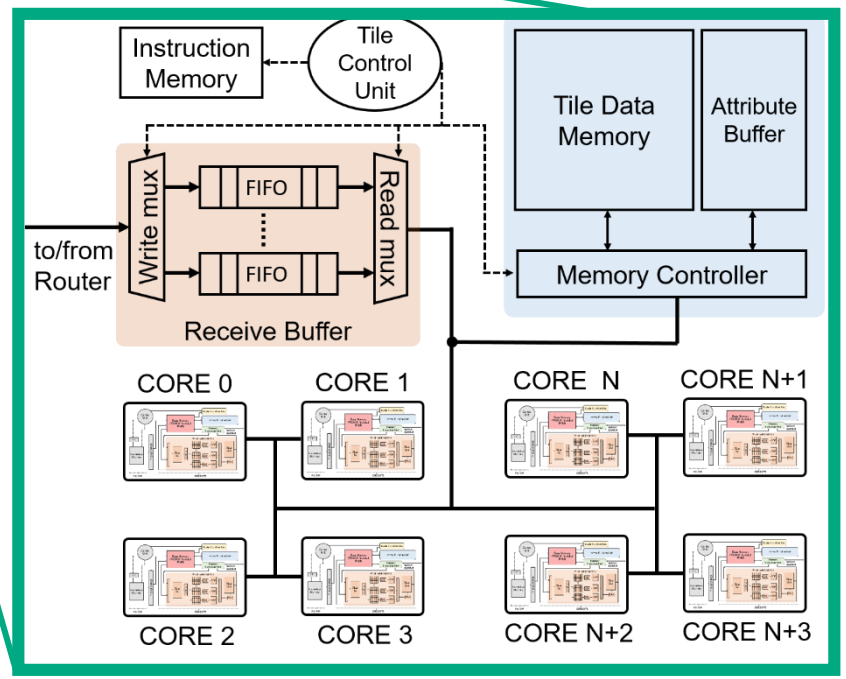
PUMA = Programmable Ultra-efficient Memristor-based Accelerator



Node

Each “node” consists of multiple tiles, with an on-chip network, implementing the NN layers

Tile



Within each “tile” are multiple “cores” with a shared memory for holding data to/from other tiles and to/from cores.

Cores perform all scalar, vector, and matrix operations.

Hewlett Packard Enterprise

Minimal area and power overhead

Architecture	Applications Supported	Tile-area (mm^2)	Tile-power (mW)
ISAAC	CNN	0.372	330
PUMA	CNN, RNN, LSTM, BM, RBM, MLP, etc	0.491 **(1.32x larger)**	377.9 **(1.14x larger)**

ISAAC = accelerator for CNN.

PUMA = accelerator for CNN, RNN, BM, RBM, MLP, etc.

- New ISA and Microarchitecture in PUMA provides **much broader application support**, with a **small area and power cost**

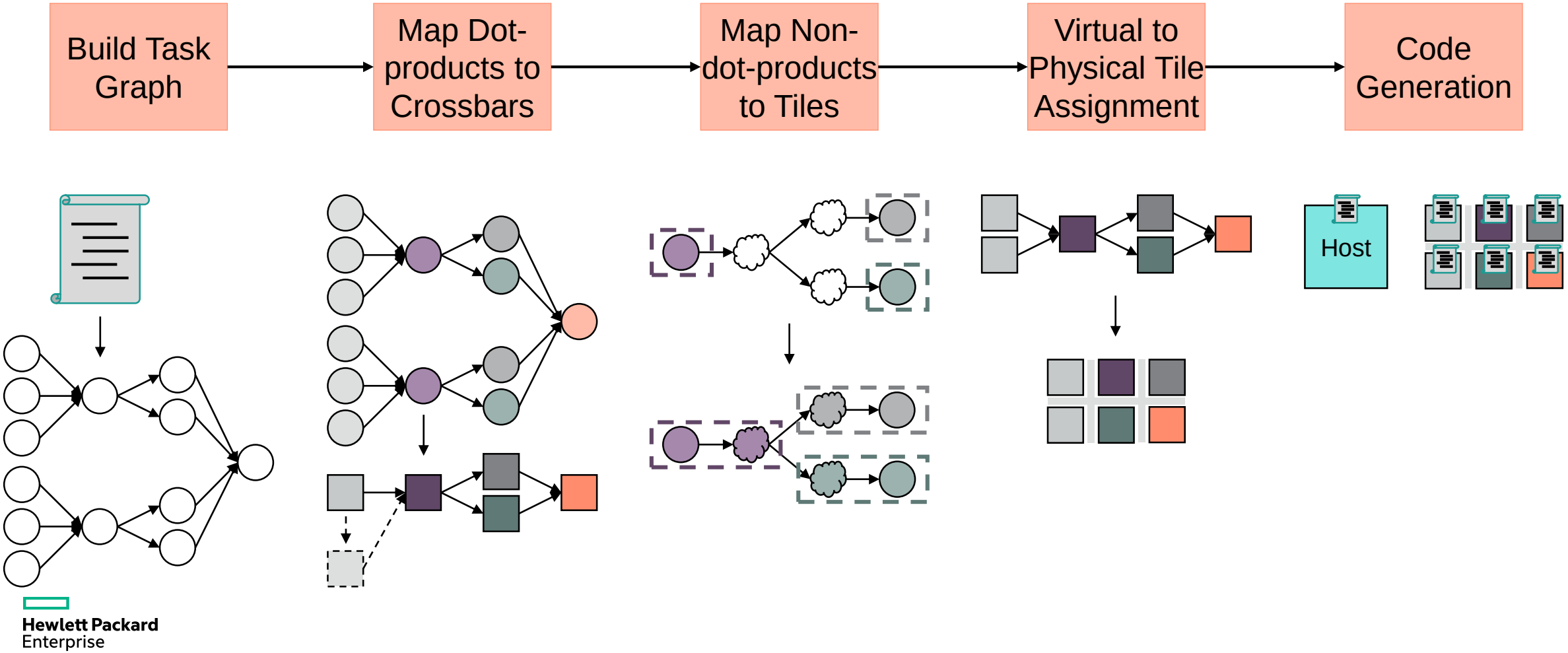
The PUMA compiler

– As applications grow in size and inter-core/tile dependencies become more complex, a compiler becomes mandatory to increase programmers' productivity

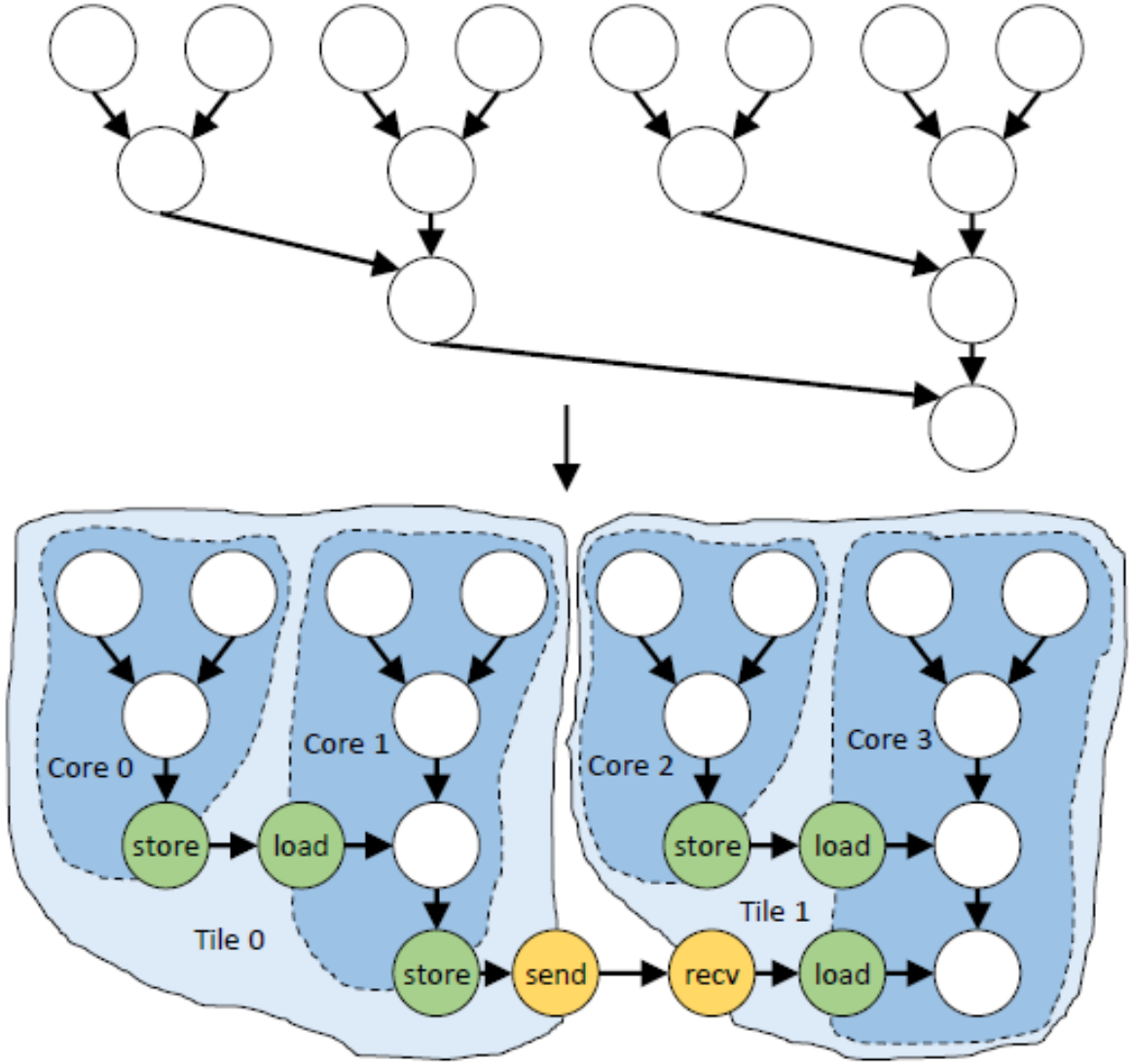
– We designed and implemented a complete compiler that translates high-level programs to PUMA assembly

– Compiler supports the following aspects
  – Programming Interface
  – Graph Partitioning
  – Data Communication
  – Tile Placement
  – MVM Coalescing
  – Linearization
  – Memory Allocation

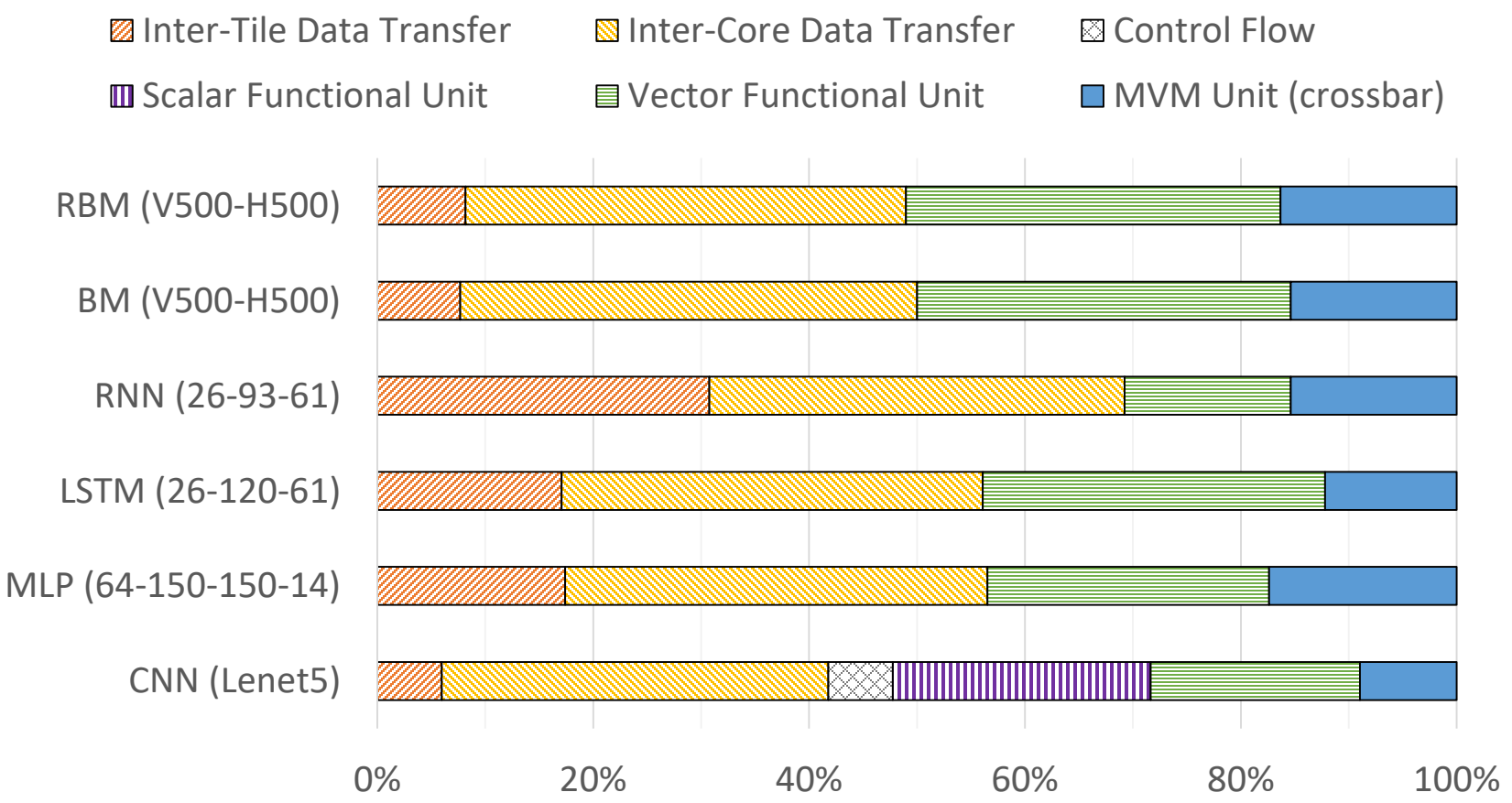
Memristor-based DPE programming



Graph partitioning and data communication



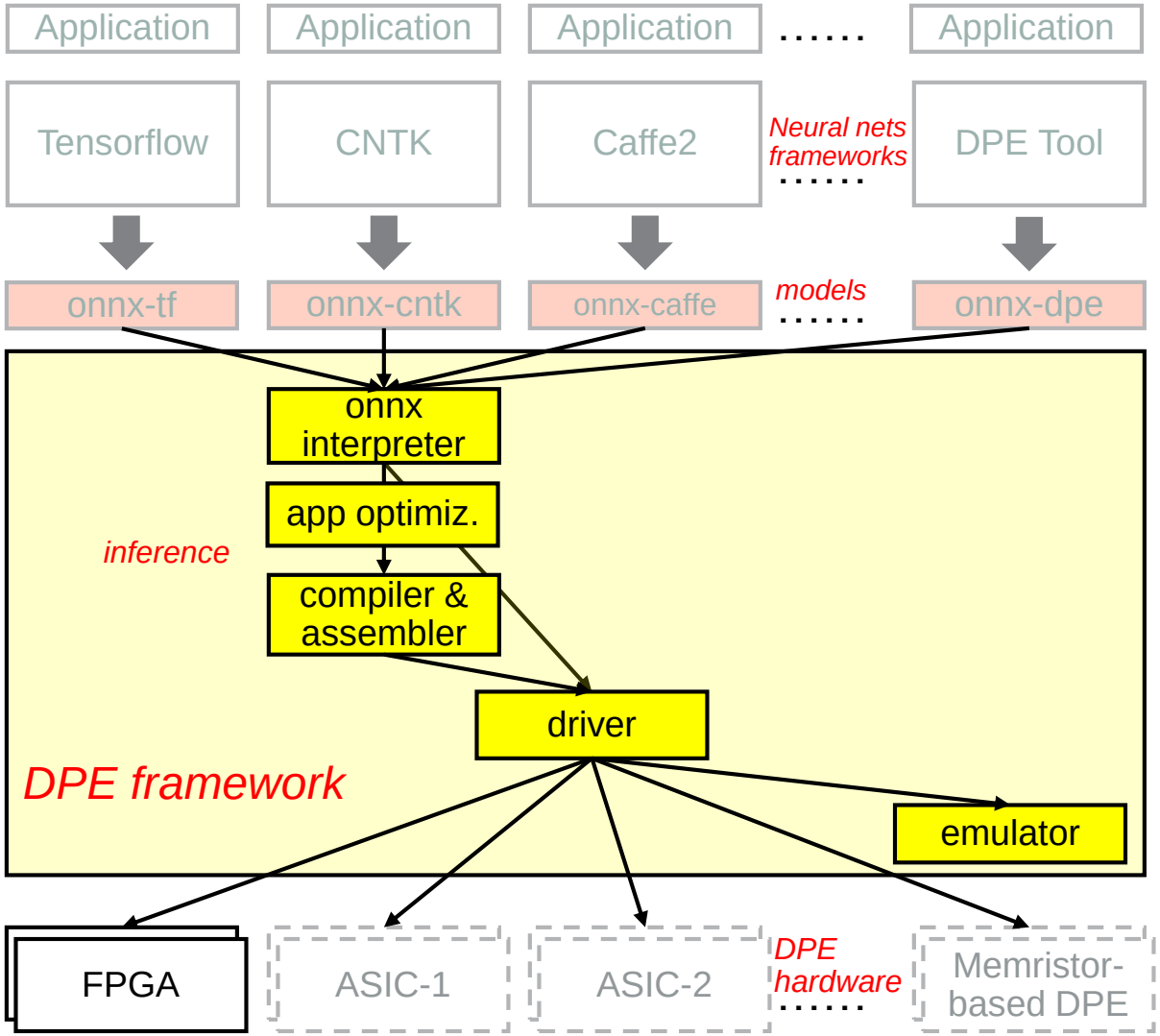
Instruction usage (importance of different execution units in PUMA core)



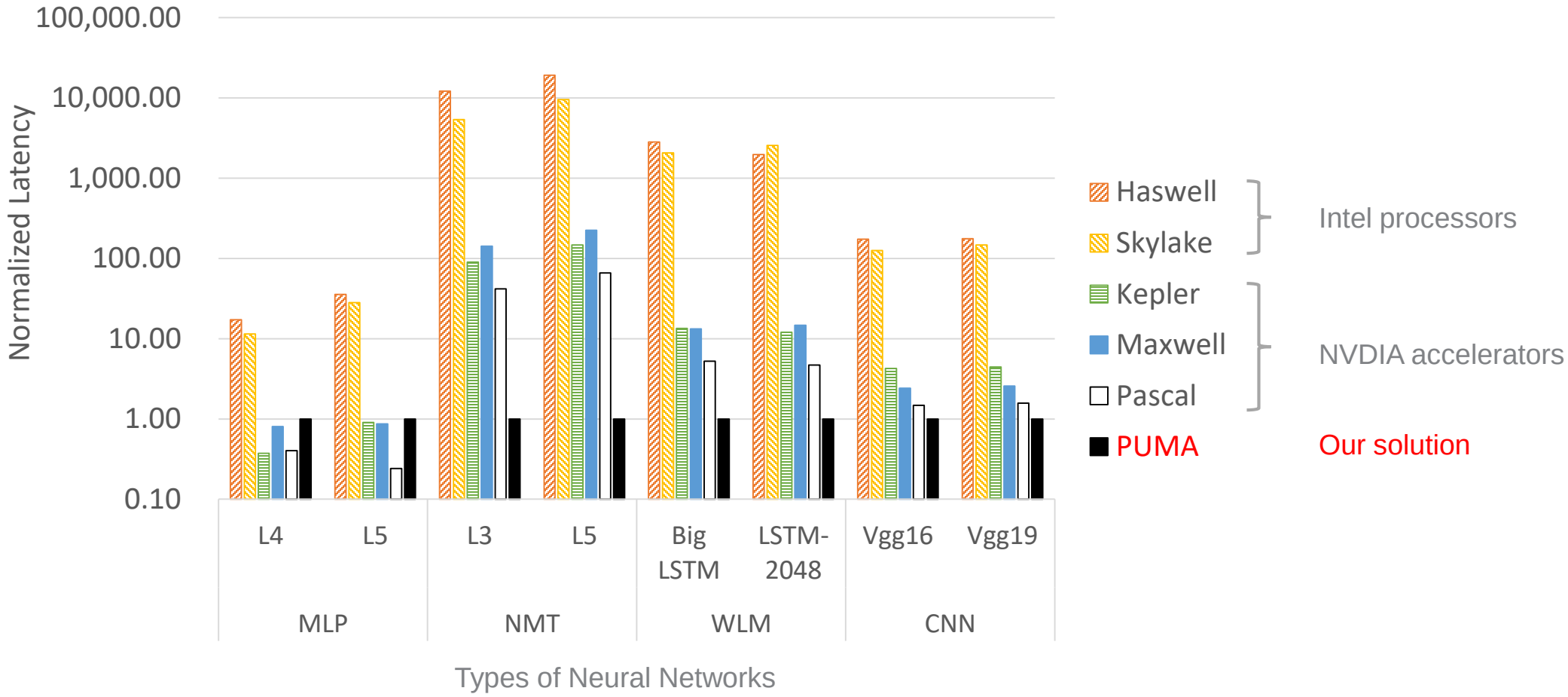
All ISA components play an important role across various neural networks

Software stack

- DPE-SW components
  - ONNX Interpreter: converts ONNX model to input to our compiler/assembler and eventually hardware
  - Applications optimization: improve performance of deployed models (quantization, weights, etc.)
  - Compiler and assembler: converts models into executables for our own hardware
  - Driver: operating system support for the hardware running DPE, e.g. for multiple PCIe boards
  - Emulator: used for hardware verification (Brazil) and as a development platform for software, QEMU
- Summary
  - Only required DPE software components being developed for users & developers' community, leverage rest!
  - Focus on inference (hardware, performance, robustness)
  - Extensive collaborations to support use cases and demos; but also driving future support, e.g. training



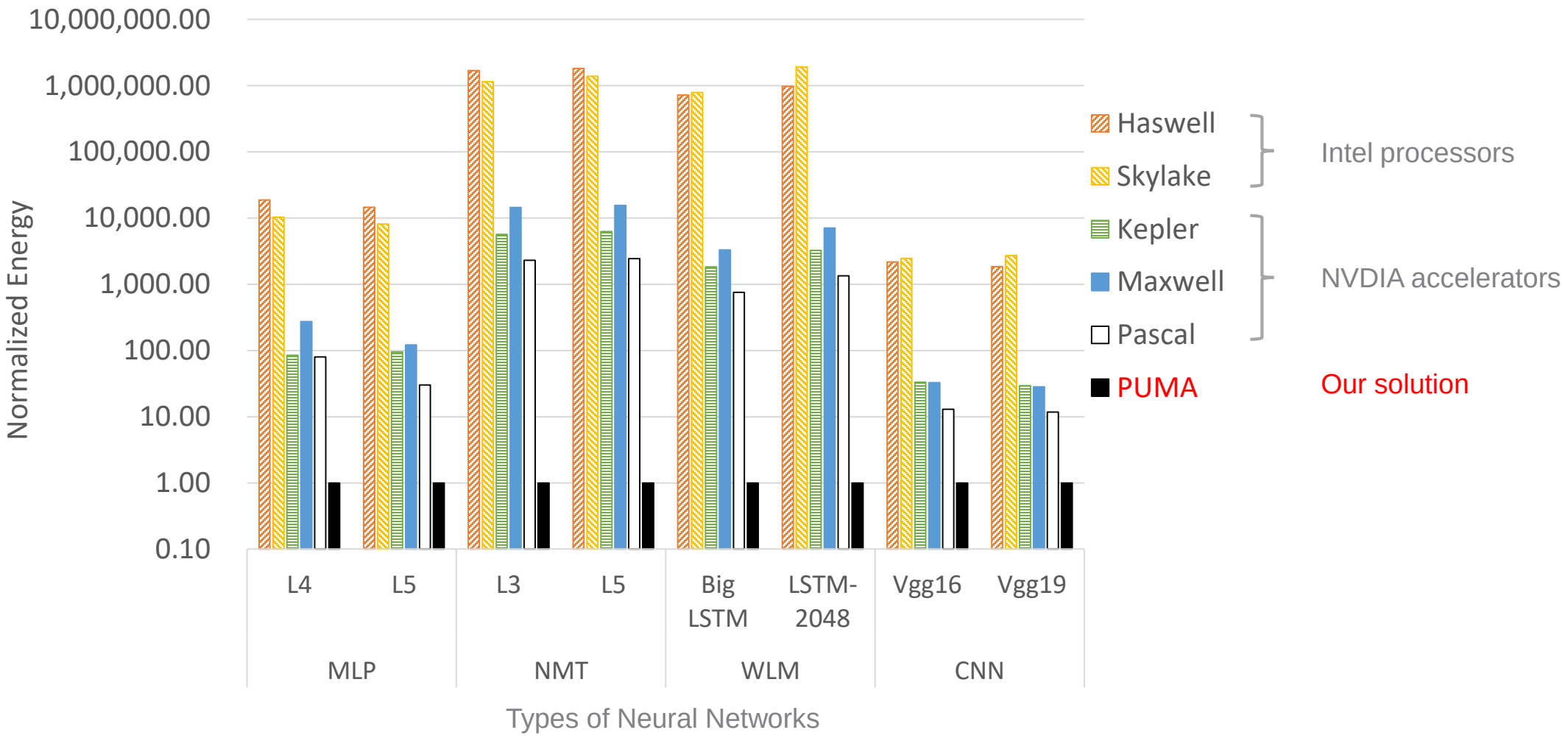
Inference latency normalized to PUMA (lower is better)



Our solution is better than Intel processors (10x-10^4x) and NVIDIA accelerators (10x-10^2x) except for one NN

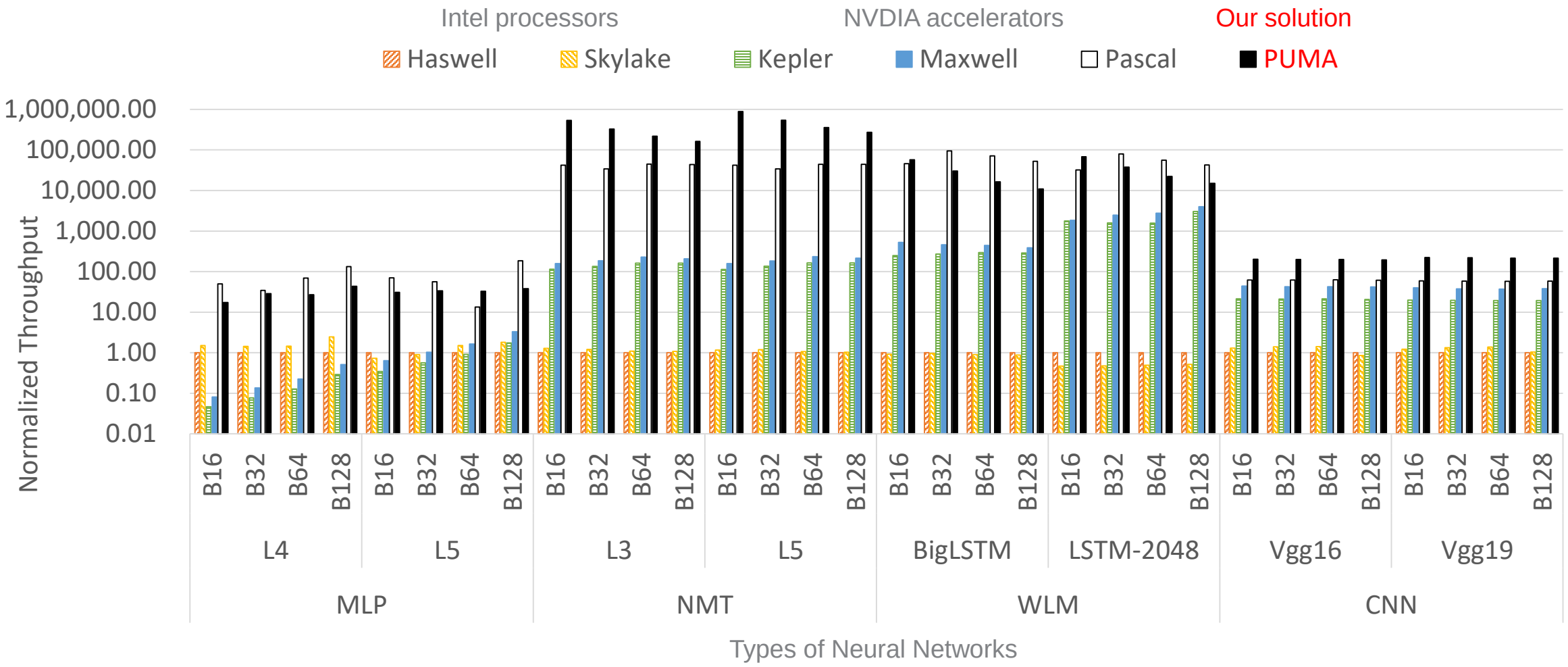
Hewlett Packard Enterprise

Inference energy normalized to PUMA (lower is better)



Our solution is substantially better than both Intel processors (10^3x-10^6x) and NVIDIA accelerators (10x-10^3x)

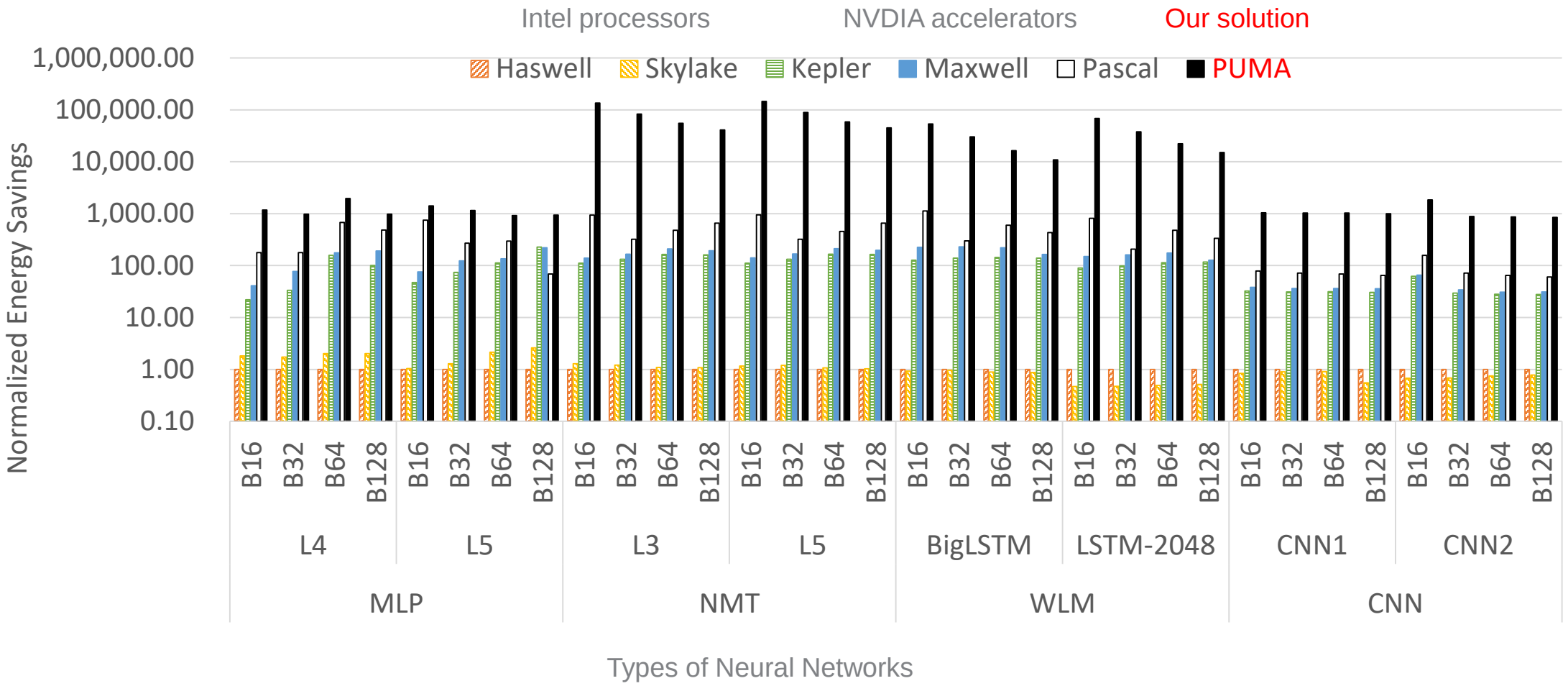
Batch throughput normalized to Haswell (higher is better)



Our solution is better than Intel processors (10^3x-10^6x) and comparable or better than NVIDIA accelerators

Hewlett Packard Enterprise

Batch energy savings compared to Haswell (higher is better)



Our solution is better than Intel processors (substantially, 10^3x-10^5x) and NVIDIA accelerators (10^2x-10^3x)

Hewlett Packard Enterprise

Opportunities and next steps

Urgency: Time to market

- Aggressive investment by competitors
- Google Tensor Processing Unit (TPU), Microsoft FPGAs, many startups
- DPE offers significant advantages to leverage

Market opportunity

- Focus on Edge; in between *sensors* and *data center*
- Leverages HPE presence
- Bulk of industry investments are at the datacenter or the far edge (sensors)

Next steps

- Standardization - interoperability with ONNX format
- Developing an FPGA prototype, followed by an ASIC for Aruba applications
- Platform for further SW development
- Proof of Concept for early Edge applications development
- Engage customers and future partners

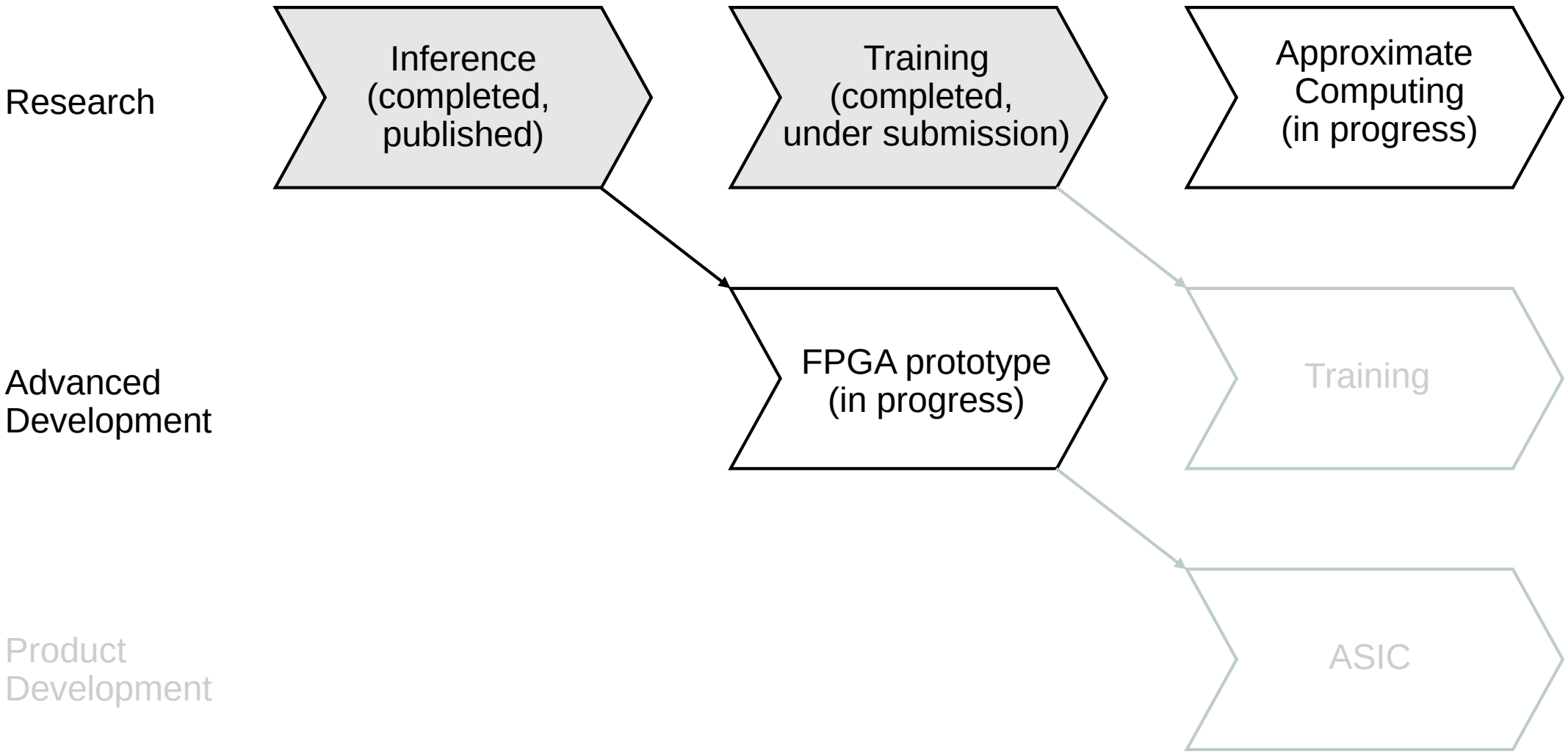
Further technology improvements

- Support NN training
- Open source (compiler/ISA, with UIUC/Purdue)

Summary

- Due to the end of Moore's law and Dennard's scaling, a need arises for new technologies
- DPE is one such promising approach that HPE has substantial IP ins (others include Quantum, adiabatic, neuromorphic, bio-inspired, ultra low power, reversible, stochastic, optical, etc., etc……)
- DPE's original architecture was specialized and optimized for CNNs (ISAAC architecture, ISCA'2015)
- PUMA increases the generality of memristor-based accelerators by adding a general purpose ISA
- We have developed an architecture simulator and a compiler to generate high-level code to PUMA assembly
- PUMA achieved improvements in
  - latency, throughput, energy consumption (significantly)
  - efficiency over traditional processors
  - coverage for a multitude of neural network workloads
- We focused on inference, as we believe this is where energy efficiency pays off first
- Inference is rapidly moving to the "edge", where small devices are space- and energy-constrained
- Because of its good programmability, PUMA can also work well for training, which we are currently exploring

Overall Approach



Hewlett Packard Enterprise

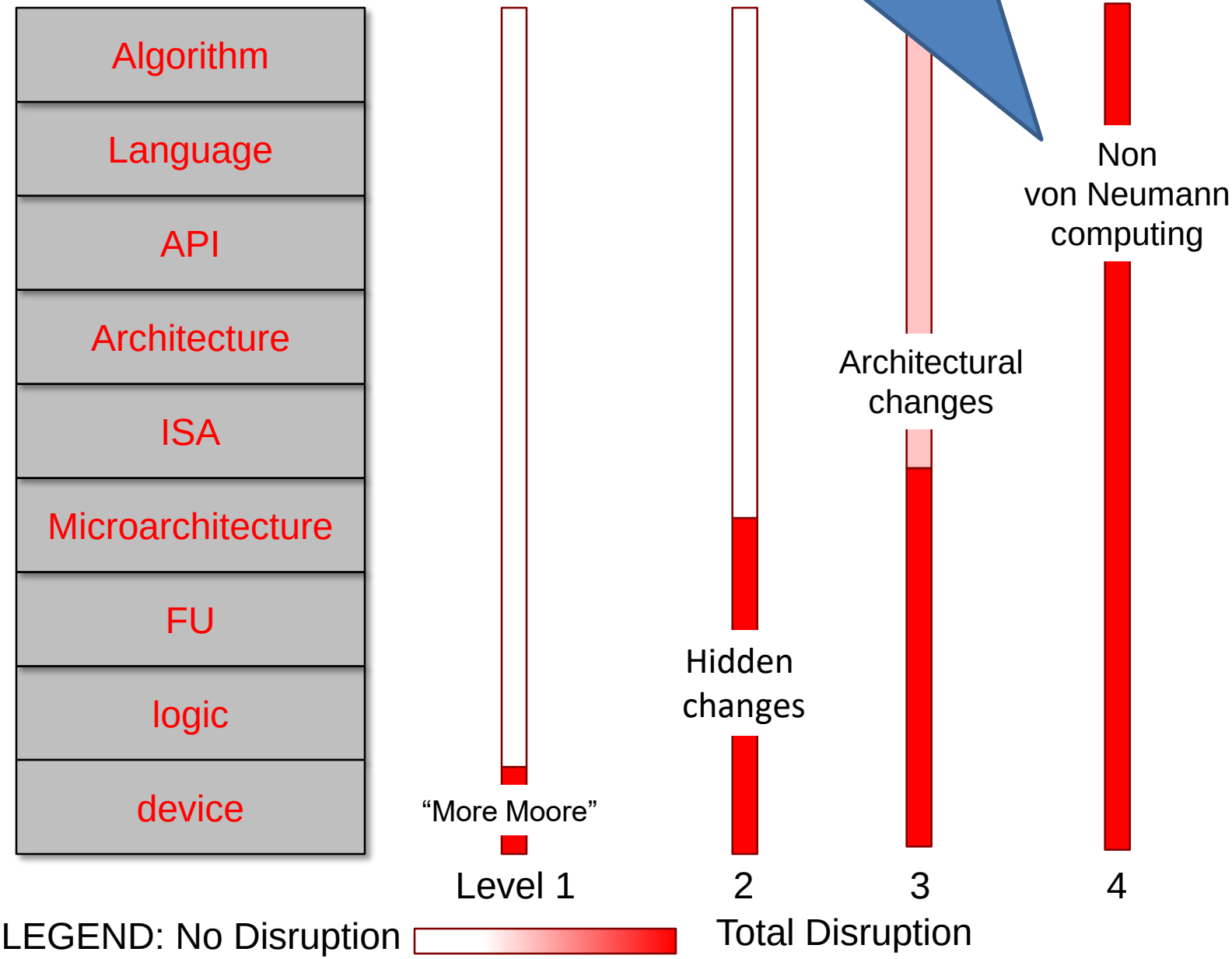
Computing In-Memory (CIM), Revisited

Dejan Milojicic, Kirk Bresniker, Gary Campbell,
Paolo Faraboschi, John Paul Strachan, Stan Williams

IEEE ICDCS, Vision Track, Vienna, Austria, July 2-5, 2018

Potential Approaches vs. ng Stack

Computing in Memory Revisited



Source: Tom Conte

Von Neumann model

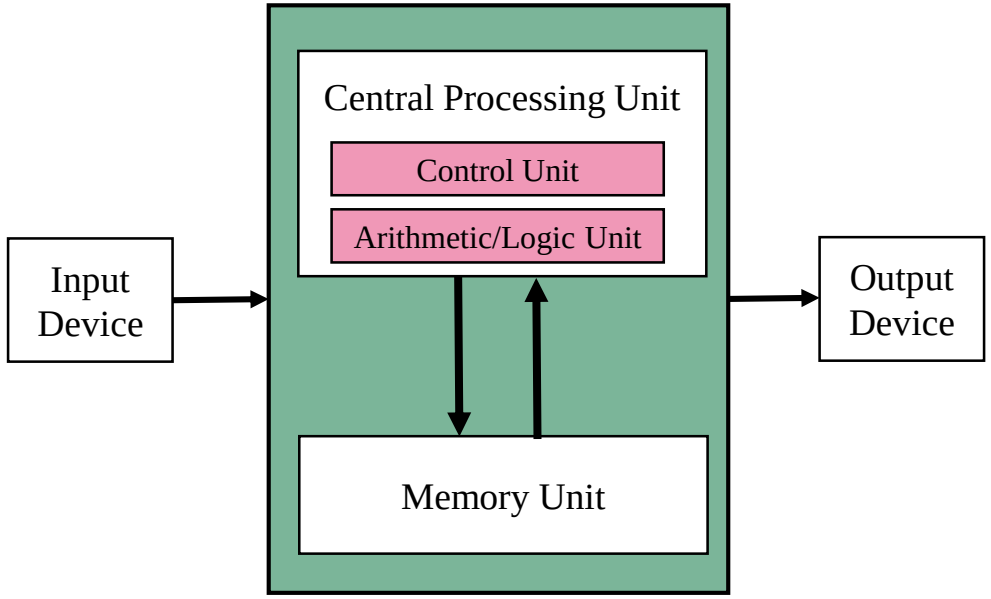
– The Von Neumann model has dominated computing systems ever since its introduction in 1945, seven decades ago

– Data & instructions stored & accessed from memory, loading them into CPU, which executes control & arithmetic/logic ops

– Over time, memory access latency started to become a problem as CPUs became faster than memory

– Cache hierarchies brought major benefits (improved memory latency), and problems (coherence complexity, security flaws)

– Alternatives, such as Processing in Memory (PIM), have been proposed, with limited success (databases, storage systems)



Challenges faced by a Von Neumann architecture

– Steady reduction of computing systems' ability to effectively operate on large data: ratio of the memory bandwidth (bytes/s) vs computing speed (flops/s)

– Steady drop over time from a byte/flop ratio of 1.0 (where all data in memory is readily available at processor speeds) to several orders of magnitude lower.

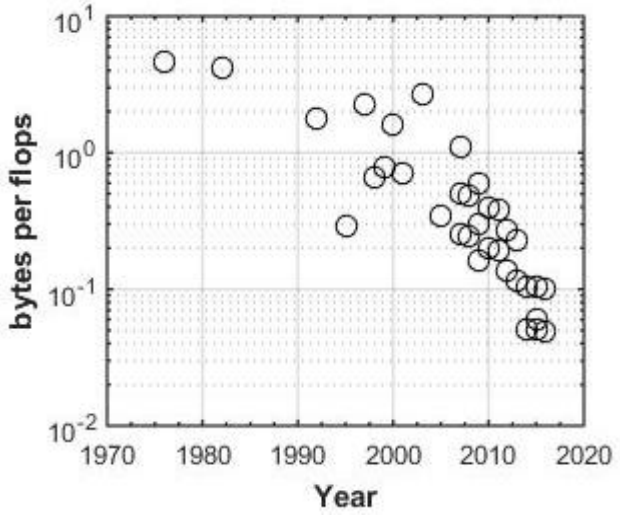
– Increased data volumes and data mining applications with limited compute intensity and locality are making this imbalance even more challenging today

– There is a strong interest to find ways to reverse the historical trend and significantly increase the bytes/flops ratio.

– The introduction of novel memory devices that can combine storage and computing in the same cell provides an opening for such a reversal

– With these devices, it makes much more sense to bring the computation to memory, also the basis of what at HPE we call "memory driven computing"

– In our CIM vision, we focus on dataflow-like architectures, where data is continuously input into device which stores some data and computation.



Memory bandwidth per processor floating point operations (FLOP)

Past work

PIM in 1990s

– Peter Kogge at the University of Notre Dame, an early proponent since EXECUBE work

– David Patterson at UC Berkeley, on mixing logic and DRAM

– Josep Torellas at UIUC, FlexRAM, and many others

Memory Side Accelerators

– Chameleon (UIUC and SNU), near-DRAM Acceleration Architecture (2016)

– Our own Labs and HPE efforts

– Micron's Hybrid Memory Cube integrated some logic with memory, but with exclusive operations (2011)

Use of Memristors and recent technologies

– Yavits, ReRAM-based Associative Processor (2015), CAM approach similar to Cat's work

– Ni, Distributed In-Memory Computing on Binary RRAM Crossbar with digital memory logic pair (2017)

– Zha, Reconfigurable In-Memory Computing with Resistive Memory Crossbar (2016), and many others

If many failed, why do we think CIM can be successful?

Earlier attempts were ahead of their time and the current attempts are timely because of the “perfect storm” effect caused by the convergence of the following trends:

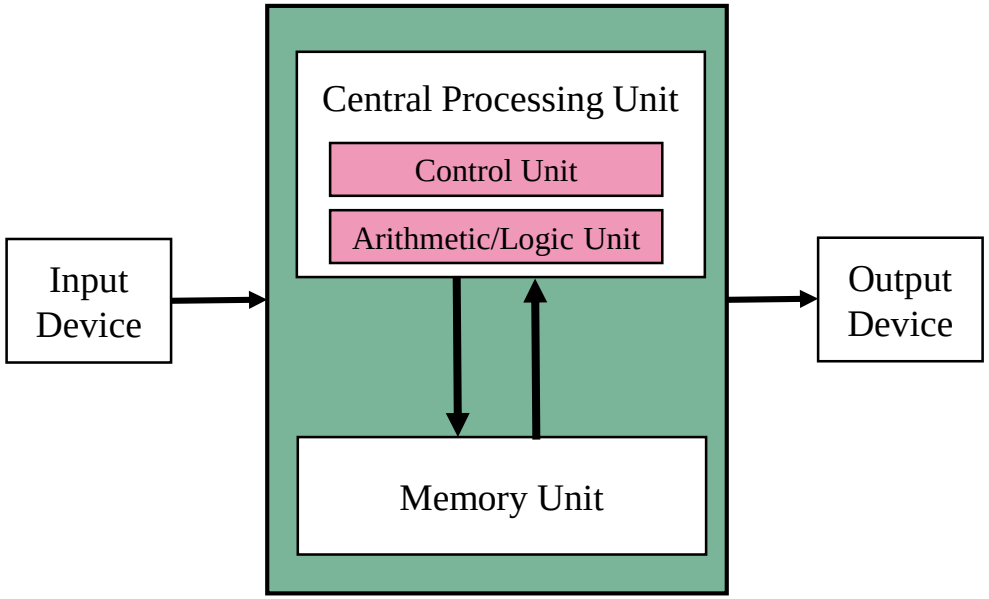
- **Emerging technologies** (neuromorphic, bio-inspired, adiabatic, reversible, approximate, quantum) are opportunity to revisit Von Neumann model and problems with caches, security, complexity, etc.
- **Application demand:** image/video/audio recognition and data analytics dominate processing compared to general purpose computing and are becoming center of attention of CPU vendors and IT companies
- **Economy of scale:** PIM lacked the economy of scale of IoT, hardware accelerators for AI/ML/DL are broadly deployed on sensors and mobile devices, turning economy of scale to their favor
- **Critical to mankind:** deep learning applications are being deployed in every facet of life (phones, sensors, autonomous vehicles, manufacturing) and mankind is increasingly dependent on IT, cybersecurity

Use cases

- **Edge computing** is close to the data generation sources (e.g. sensors), perform analytics to filter out redundant data and extract information (e.g. convert images/video into a tagged meta-data representation)
- **Memory-centric computing**: value/size of data grows higher than computation, becoming first level citizen, surrounded by computation as needed (e.g. graph-heavy apps in intelligence & social networks communities)
- **Deep Learning**: as content complexity increases, AI and Machine Learning (ML) can leverage CIM because of the dataflow nature of tensor operations, and the underlying matrix operations.

- These use cases have the following characteristics in common
  - **Data is close to computation:** there is no need to move it, resulting in power/performance optimization; earlier approaches (offloading and migration), were not as effective and break programming models.
  - **(Some) Data is persistent**: the application state is constantly captured over time and upon reboot or restart (due to failure) it will be available to continue computation (NVMs or distributed persistence)
  - **Applications employ dataflow**: data manipulation, understanding and mining matches well dataflow programming models and opens up opportunities for embarrassingly parallel computing

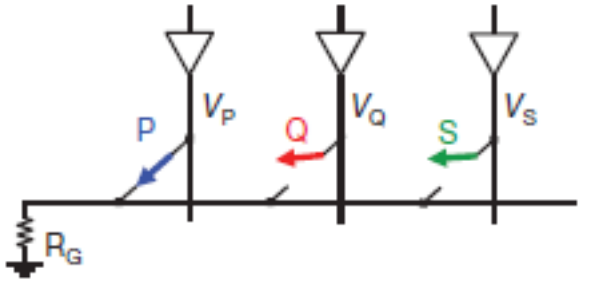
Evolving Von Neumann's architecture into CIM



Von Neumann Architecture

a

	Clocking voltages	
Step 1: $s = 0$		$V_S = V_{CLEAR}$
Step 2: pIMPs	$V_P = V_{COND}$	$V_S = V_{SET}$
Step 3: qIMPs	$V_Q = V_{COND}$	$V_S = V_{SET}$

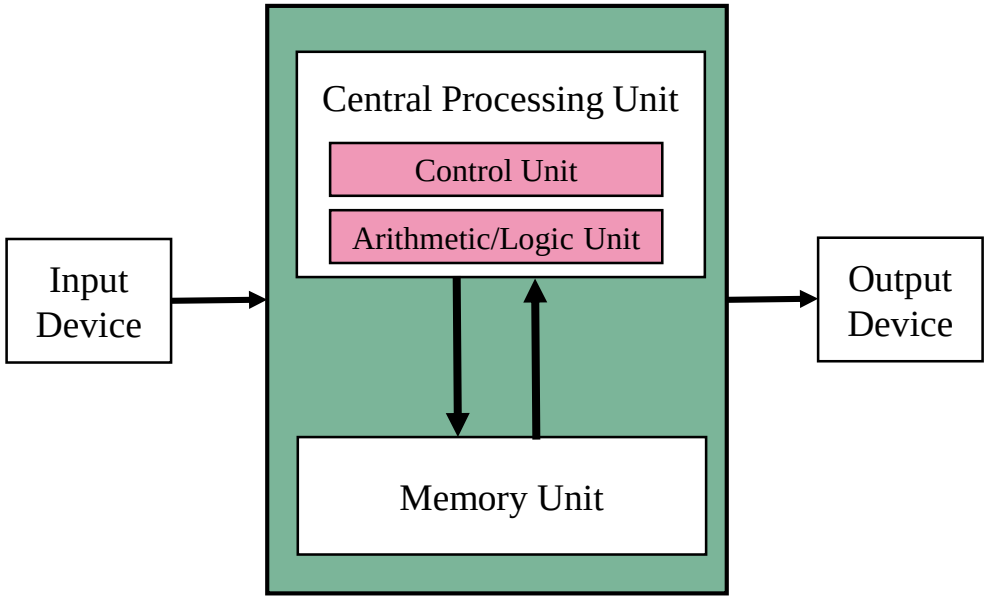


b

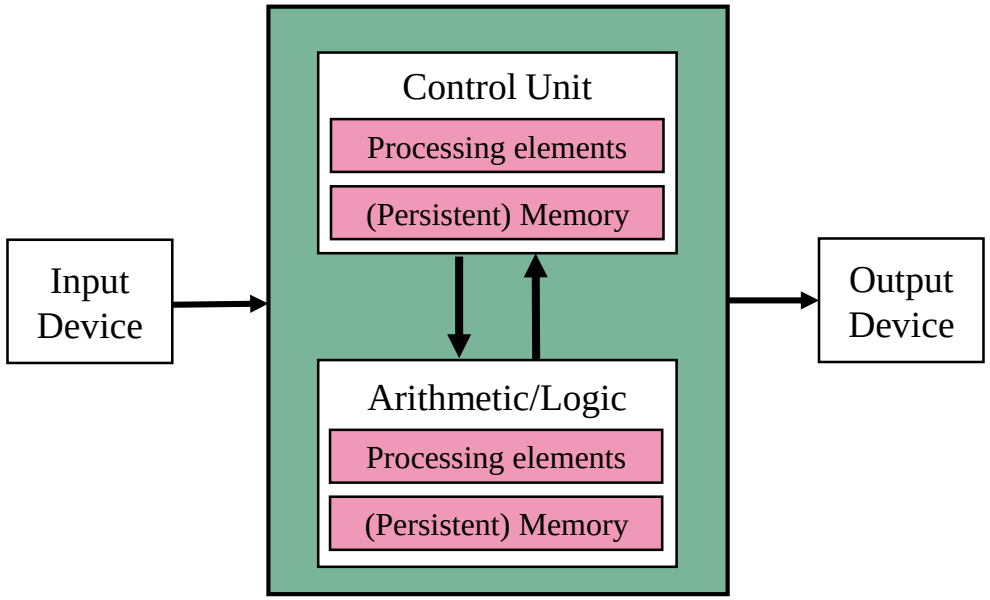
Step 1	Step 2		Step 3				Steps 1, 2, 3			
$s = 0$	$s' \leftarrow p$IMPs		$s'' \leftarrow q$IMPs'				$s'' \leftarrow p$NANDq			
s	p	s	s'	q	s'	s''		p	q	s''
0	0	0	1	0	1	1	=	0	0	1
0	0	0	1	1	1	1		0	1	1
0	1	0	0	0	0	1		1	0	1
0	1	0	0	1	0	0		1	1	0

'Memristive' switches enable 'stateful' logic operations via material implication
Julien Borghetti, Gregory S. Snider, Philip J. Kuekes, J. Joshua Yang, Duncan R. Stewart & R. Stanley Williams
Nature Letters, Vol 464|8 April 2010| doi:10.1038/nature08940

Evolving Von Neumann's architecture into CIM, cont



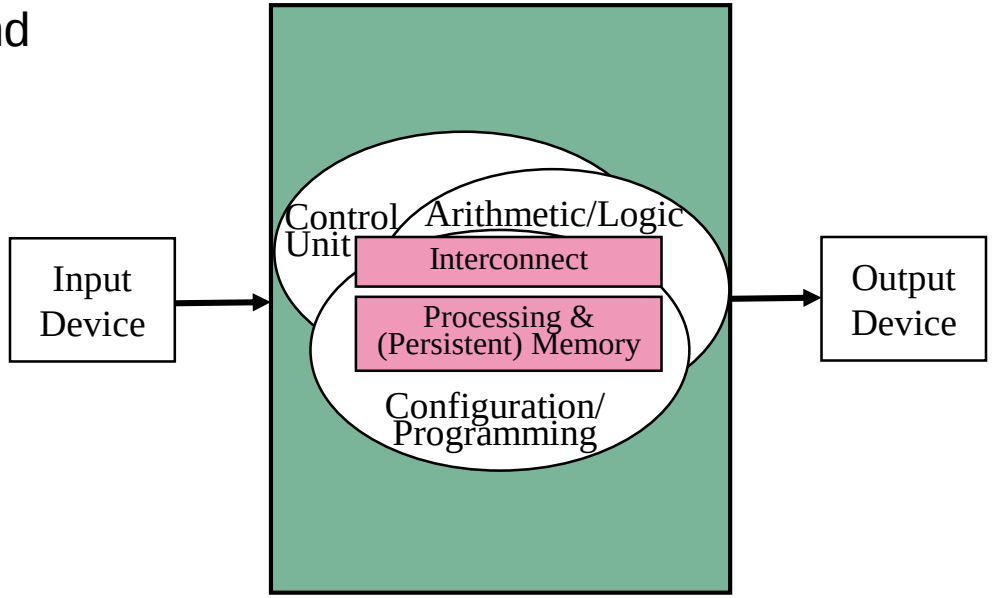
Von Neumann Architecture



Von Neumann Architecture, Evolved

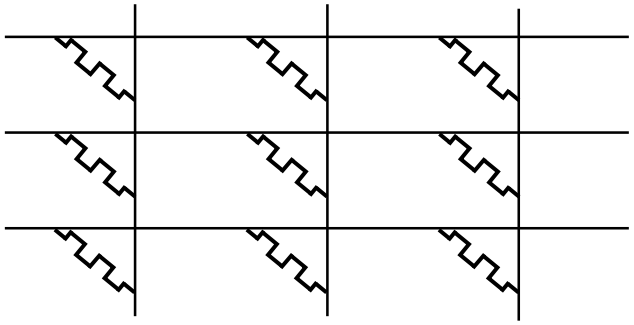
Evolving Von Neumann's architecture into CIM, cont.

- CIM technologically and architecturally collocates processing and memory, for compute (logical, arithmetic) and control functions
- Interconnects are integral to CIM, programming & configuration becomes core functionality for control and arithmetic/logic units
- Interconnects reconfigure paths for the dataflow model, and reconnect units into application-specific workflows
- Interconnect standards (CCIX, GenZ, and OpenCAPI) are the prime candidate for success in this domain

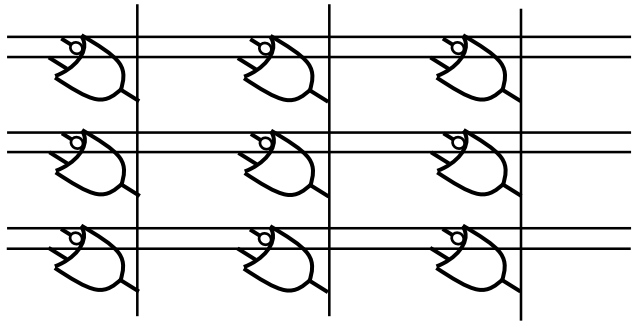


Von Neumann Architecture, Evolved

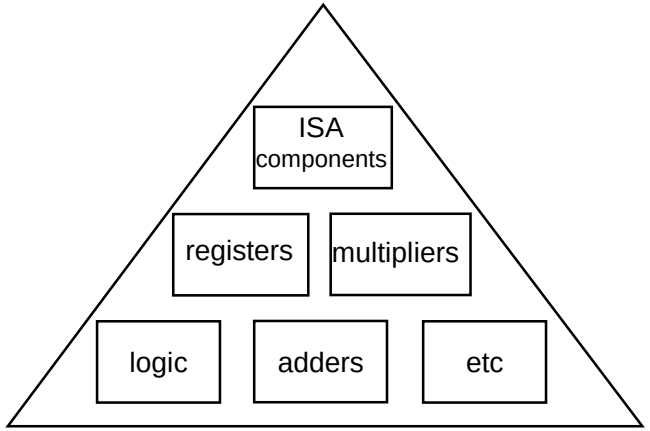
Evolving Von Neumann's architecture into CIM, cont.



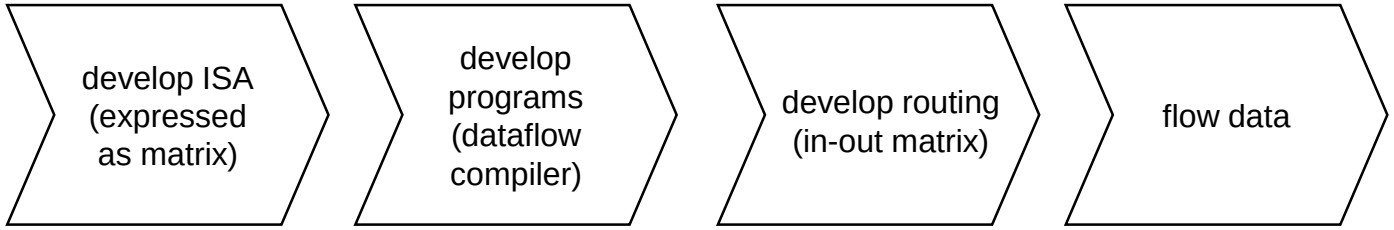
(a) physical memristors matrix



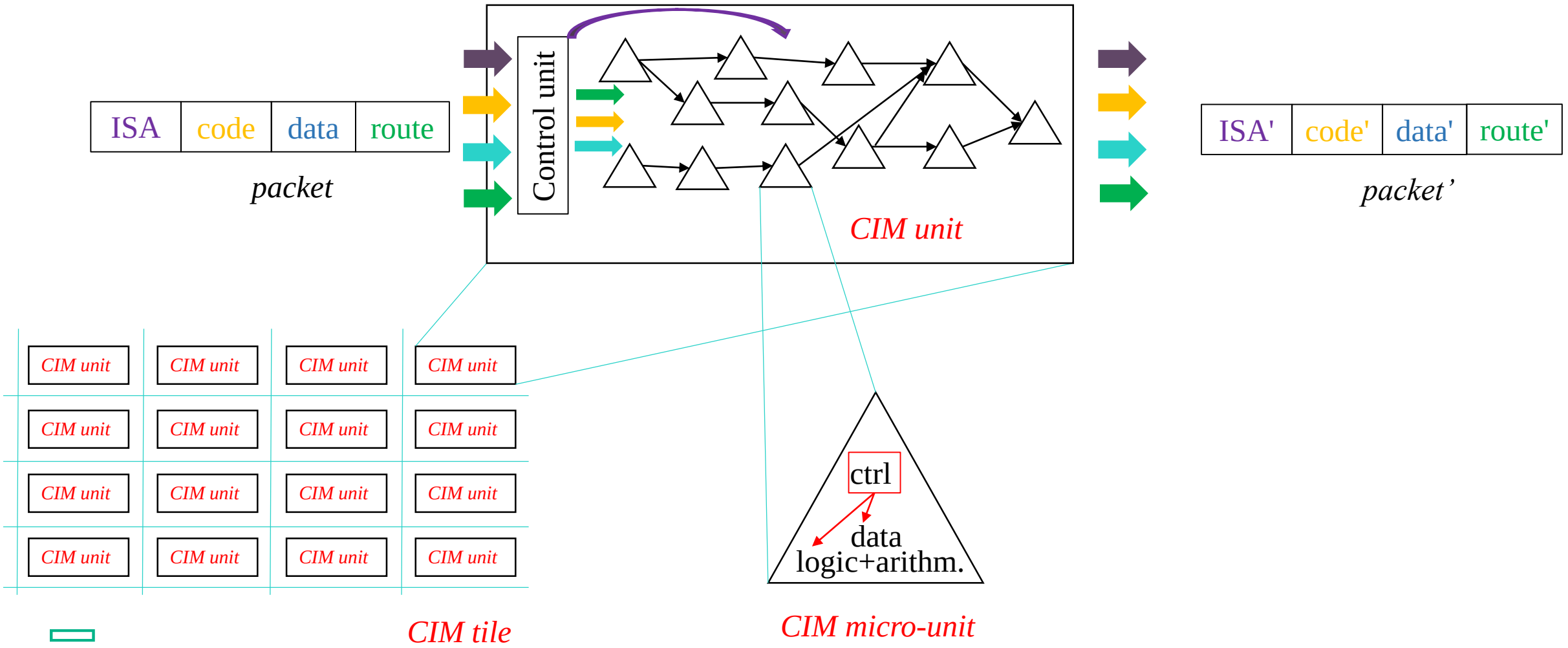
(b) logic elements matrix



(c) CIM micro-unit



Possible CIM architecture



Programming CIM

- CIM programming adopts static, dynamic, and self-reprogrammable dataflow concepts, each bringing an additional degree of flexibility achieved by reconfiguring different aspects of the CIM architecture:
  - **Static dataflow** is the natural extension of existing dataflow models, applications can program the CIM crossbars to implement a target neural network that would execute over and over again. This enables more opportunities for training, as well as feed-forward and closed loops.
  - **Dynamic dataflow** assumes that the data coming in can be dynamically routed to parts of the CIM at different granularity as a function of the state in the CIM and the input data. The routing could be expressed explicitly as a part of the incoming packet or implicitly as a function of the state in CIM, or both.
  - **Self-programmable dataflow** enables carrying code as a part of the packets to dynamically program functions as packets arrive. This allows the highest level of flexibility in programming. Past research exists on this topic, but no production-level commercial equivalent exists as yet.

Security, virtualization, resource management

- Security often considered an afterthought, with performance and reliability dominating requirements; as a consequence numerous bugs, leaks, and exploits are consistently being discovered
- A complete new architecture paradigm opens an opportunity to reconsider security as a first class requirement, along with security come the tightly related requirements for virtualization and resource management.
- **Security** can benefit from
  - **Packet based communication**, which is better understood than shared memory model, paths can be better secured by partitioning and data inspected prior and after entering and exiting CIM model
  - **A dataflow architecture** can introduce barriers as a containment mechanism to stop propagation of errors and bugs; packets in flight can be encrypted and networking key protection model applied
- **Virtualization and Partitioning** can benefit from analogy to Network Function Virtualization (NFV)
  - **Dynamic hardware isolation**: parts of the CIM components can be completely isolated from other parts for security
  - **Quality of service**: minimal influence from one stream to another achieved by provisioning enough interconnect
  - **Failover**: should streams be redirected, switching to other components would have minimal impact on performance.
- **Resource Management**: traditional load balancing techniques apply to CIM:
  - **Load information** measures latencies and bandwidth of streams, as well as usage of individual and aggregate resources.
  - **Load balancing** by redirecting streams to underutilized CIM components, some of the streams may need to be pinned
  - **Closed loops** can be used to manage performance according to given SLA agreements

Non-functional characteristics

- Non-functional characteristics of new hardware and computing-memory models are equally important for its success as the functional characteristics
- Failure tolerance
  - **Fault detection** can be accomplished at any component level, it can use extra bits on data or instruction states
  - **Fault containment:** boundaries of each component are convenient place to contain errors
  - **Fault prevention** can be accomplished through redundancy of information and components.
  - **Fault recovery** by failing over to redundant components and by retaining data in preceding components
- Scaling
  - Similar to scaling web servers, if the individual elements are stateless and only execute data streams, if stateful, requires scaling of each class of modules and then spreading the state and interactions with end-to-end application
- Configurability
  - Design points: precision and number of bits at lowest level, reconnecting components enables reconfiguration at higher levels, similar to Coarse Grained Reconfigurable Architectures (CGRA)
- Serviceability
  - Deployed equipment is hard to support, even harder at the edge, motivating the need for graceful aging and self-healing at multiple levels of CIM components. This leads to preventive support, self-healing, and closed loops

Performance

- We have pursued a form of CIM under the umbrella of the Dot Product Engine project, we implemented static data flow CIM model which enables us to program and reconfigure the CIM for classes of neural networks
- This is the follow-on work to the ISAAC architecture, which we made more programmable; detailed performance is beyond the scope of this paper, we provide orders of magnitude benefits
  - for neural network class of applications, we achieved latency between 10 and 10^4 times better than CPUs and between 10 and 10^2 better than GPUs
  - we achieved bandwidth better 10^3-10^6 times compared to modern CPUs and comparable to modern GPUs.
  - in terms of power, benefits are even larger, 10^3-10^6 better than CPUs and 10-10^3 better than GPUs.
  - performance benefits of using CIM are real. Of course they cannot be generalized to all applications, we expect challenges in terms of hiding the asymmetric latency for writing memristors based devices
- We consider acceptable scaling with multiple boards with standard and proprietary interconnects

Comparison of different approaches to computing

Comparison	Approaches to Computing		
	Von Neumann		In-Memory
	Parallel (shared memory)	Distributed	
programming model (common)	multi-threaded	message passing	dataflow
scaling (per system)	100s of cores (eg HPE Hawks)	200 racks (e.g. Exascale)	no perceived limit, higher then exascale
failure tolerance	whole partition fails	failover to another machine	stream redirection to redundant unit
security	whole partition	machine boundary	packet and stream based
robustness	OS-dependent	cluster dependent	application- specific

Application suitability to CIM model

Class of application	Characteristics						
	Compute intensive	Data intensive		Operational intensity Flop/byte (temporal locality)	Communication (iterative)	Parallelism (dependencies)	CIM
		Bandwidth	Size				
Machine learning	high	high	high	high	low	high	high
Neural Networks	high	high	high	high	low	high	high
Graph problems (FB, intel.)	low	medium	high	high	high	high	high
Bayesian inference	high	low	low	high	high	medium	low
Markov chain	high	low	low	low	high	high	low
KVSs (persistency layer)	low	high	high	low	medium	high	medium
Data Bases (analytics)	low	high	high	low	medium	high	high
Data Bases (transactions)	medium	high	low to med.	high	high	medium	medium
Search (indexing problem)	high	high	high	high	high	high	low
Optimization problem (resource allocation)	high	low	low	high	high	low	low
Scientific Computing	high	low to high	low to high	medium	high	high	low
Finite Element Modelling	high	low	medium	medium	high	high	medium
Collaborative (mail, chat,..)	low	high	medium	low	high	low	low
Signal (image) processing	high	high	high	low	high	medium	low
CIM	low	high	high	high	low	high	

Related work

– Different teams have approached the core operations differently. Chen et al. rely on AND, OR, and XOR operations upon which to build all other logic

– Borghetti et al. are using NOT and IMP (material imply) as two core logic operations, hardware architectures are based on these operations

– Various approaches to hardware architecture, classified by Khoram et al., rely on: matrix multiplication (dot products) combined with shared memory, such as in ISAAC and memristive Boltzmann machine

– Neuromorphic systems mimicking human brain, such as in FlexRAM and work by Liu

– Associative processors known as content addressable memory combined with nonvolatile memory, such as TCAM and Associative Processors

– Coarse grained reconfigurable architectures, such as nonvolatile FPGA and reconfigurable in-memory computing architecture

Summary

– We have motivated the need for adopting the new computing architecture, Computing-in-Memory; we discussed use cases where it could be beneficial, as well as applications.

– We presented the CIM model in more detail, discussed security and other non-functional characteristics and compared them against those in von Neumann architecture.

– Computing in Memory is already being demonstrated in research and adopted in product prototypes, whether it will take off is tied to future applications; however, it is important to avoid repeating mistakes from the past

– These are there most important aspects that need further in-depth exploration:
  – **exploiting inherent application parallelism**: recognizing dominant applications of the future suitable for CIM will depend on application inherent parallelism and architectural support to exploit this parallelism
  – **security**: as architects and systems and applications developers start to design solutions, security needs to be treated as first level requirement (if we do not want to be haunted by bugs and vulnerabilities of the past)
  – **software development productivity**: the largest cost in systems solutions lies in software development, whatever computing models are developed will also require conducive ways of software development

Thank you!

Questions?