Compute Cache Architecture for the Acceleration of Mission-Critical Data Analytics

H. Lam, S. Bhat, K. Rajasekaran,V. Srinivasan D. Ojika
Center for Space, High-Performance, and Resilient Computing (SHREC)
University of Florida (UF), http://nsf-shrec.org, hlam@ufl.edu

This study explores how to exploit a ***compute cache*** architecture to bring computation close to memory. Using a combination of experimental prototypes, benchmarking, and modeling & simulation, we perform architectural and application explorations of emerging/notional memory devices and compute cache architectures of the future to accelerate data analytics applications.

Analogous to a memory cache hierarchy in which different levels of memory caches (with different capacities and performance) are strategically placed in and near the CPU compute device (Fig. 1(a)), a ***complementary*** *hierarchy* of compute "caches" can be strategically placed to provide compute-in-memory (CiM) and compute-near-memory (CnM) capabilities (Fig. 1(b)). ***In-memory*** compute cache implements ***compute primitives*** (e.g., arithmetic ops, data-ordering ops) which are simple enough to be embedded in the logic layer of emerging memory (eMEM) devices. Analogous to the in-core memory caches, compute primitives provide low functionality but high performance. ***Near-memory*** compute caches can provide ***CnM capabilities*** to implement ***compute kernels***. Near-memory compute caches can be implemented on an FPGA along-side an eMEM device in the same package. CnM compute kernels make use of CiM primitives to accelerate functions (e.g., FFT, convolution, Bloom filter) or to preprocess data from the memory device (e.g., reorder, reduce) in order to support the acceleration of the host data analytics or machine-learning ***application***. This concept can be extended to near-storage compute caches.

Note that existing memory devices (HMC, HBM2) do not have or have limited CiM capabilities. Thus, for experimentation today, we implement these CiM primitives also on the FPGA of existing (FPGA + HMC/HBM2) platforms and collect benchmarking (performance) data for the modeling/simulation tasks. The benchmarking data collected from our experimental studies, along with research findings in the literatures (on emerging or notional memory devices) are used to develop models and perform simulation studies to perform *design-space exploration* of emerging/notional memory devices and compute cache architectures of the future.

In summary, the compute cache architecture provides a fundamental impact by introducing a novel approach to add a ***new dimension in parallel computing architecture***, increasing parallelism by distributing performance-critical computing near or in memory, and at different levels of granularity. A near-term impact is to demonstrate how the compute cache architecture (FPGA + eMEM) can ***amplify the acceleration capabilities of FPGAs*** (e.g., SPMV, graph analytics, DNN training) by providing low-latency, high-bandwidth memory access for FPGA acceleration.

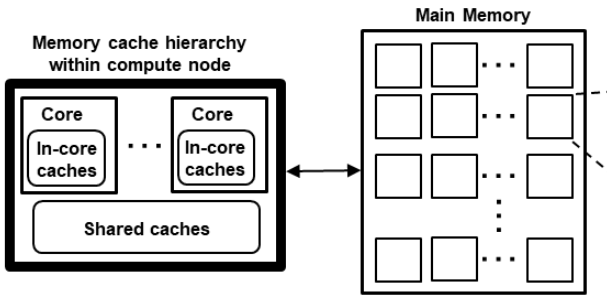


Figure 1(a) Memory cache hierarchy

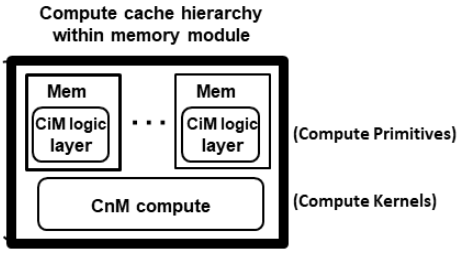


Figure 1(b) Compute cache hierarchy

* This research is funded by the NSF SHREC Center and the National Science Foundation (NSF) through its IUCRC Program under Grant No. CNS-1738420; and by NSF CISE Research Infrastructure (CRI) Program Grant No. 1405790.

Compute Cache Architecture for the Acceleration of Mission-Critical Data Analytics

Mission-Critical Computing
NSF CENTER FOR SPACE, HIGH-PERFORMANCE, AND RESILIENT COMPUTING (SHREC)

University of Pittsburgh
BYU BRIGHAM YOUNG UNIVERSITY
Virginia Tech
UF UNIVERSITY of FLORIDA

Dr. Herman Lam
UF Site Director

Dr. David Ojika
Post Doctoral Associate

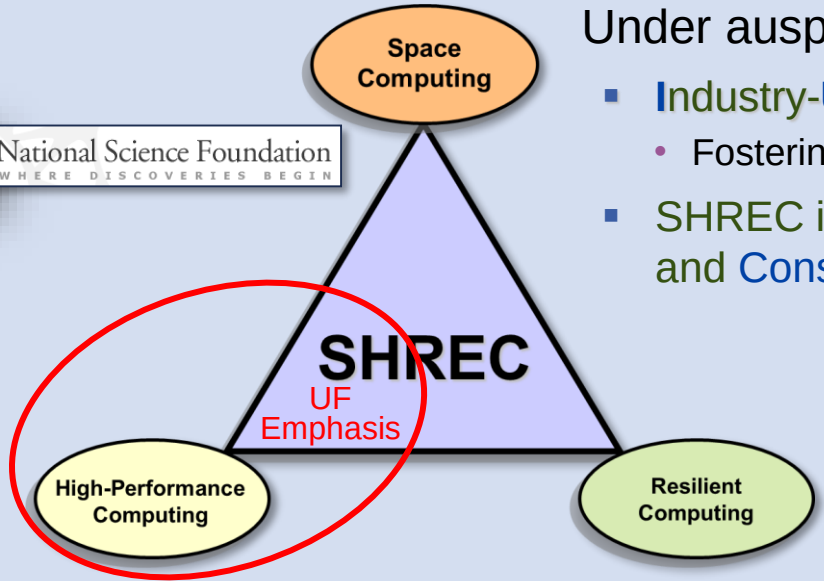
Sharath Bhat
Karthikeyan Rajasekaran
Vishnu Srinivasan
Research Assistants

NSF Center for Space, High-Performance, and Resilient Computing (SHREC)
University of Florida

Introduction to SHREC@UF*

Fermilab · Ball Aerospace · DELL · Sensors, Space Vehicles · NASA ARC, GSFC, IV&V, JSC, LaRC · National Security Agency · Lockheed Martin · OLPS The Laboratory for Physical Sciences · ONR · Harris · Los Alamos National Laboratory · Honeywell · MIT Lincoln Laboratory · Boeing · Walt Disney Animation Studios · Raytheon · U.S. Naval Research Laboratory · Satlantis · BAE Systems Inspired Work · Sandia National Laboratories · Space Micro · Draper Laboratory · intel · ARL · Collins Aerospace · L3 Space & Sensors · Emergent Space Technologies

***** NSF Center for Space, High-Performance, & Resilient Computing



National Science Foundation
WHERE DISCOVERIES BEGIN

Under auspices of **IUCRC** Program at ***NSF***

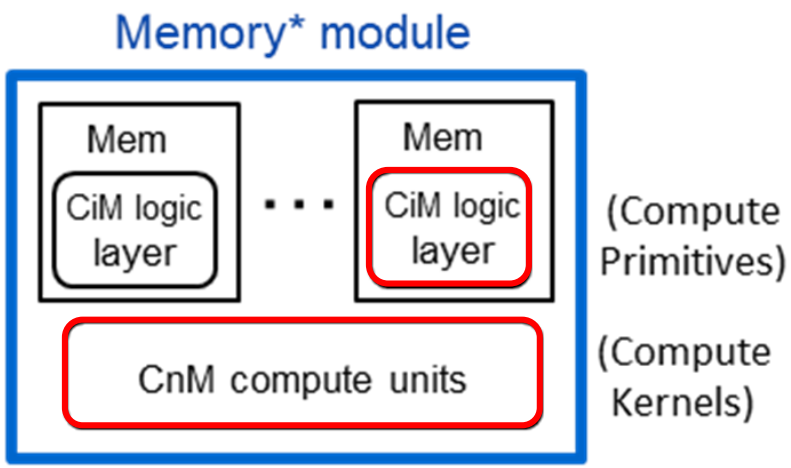
- **I**ndustry-**U**niversity **C**ooperative **R**esearch **C**enters
  - Fostering university, agency, & industry R&D collaborations
- SHREC is both National Research Center (universities) and Consortium (member organizations)
  - University of Pittsburgh (lead site)
  - Brigham Young University
  - ***** University of Florida (UF)
  - Virginia Tech

Outline

- Overview: Compute caches
  - Compute near Memory (CnM), Compute in Memory (CiM):
- *System-level* ModSim of *heterogeneous* HPC systems
  - Foundation of our approach
  - Approach: deployed, near-future, future heterogeneous HPC systems
  - Role of compute caches in SLAMS* of such systems
- Current progress
  - Experimental platform:
    - Bittware 520N-MX, Intel Stratix 10 MX FPGA
  - Case study
    - In preparation for Bittware board
    - Preliminary results

* SLAMS: System-level Application Modeling & Simulation

Introduction: CnM and CiM



* Ex: DDR/HBM memory; Storage-class memory

- CnM: Compute-near-memory
  - Ex. Intel Stratix-10 MX (HBM2 in same package)
  - CnM compute units – *compute kernels* (e.g., FFT, Bloom filter) to accel. an app
- CiM: Compute-in-Memory
  - Ex. Emerging memory* with built-in logic layer(s)
  - CiM logic – *compute primitives* (e.g., add, mult, data-ordering ops)

Approach: Combination of *experimental* and *ModSim* studies

- **R&D Thrust 1:** Amplify the acceleration capabilities of FPGAs (CnM processing)
  - *FPGA* **+** (*Mem* w/ low-latency, high-bandwidth); e.g., larger MatMult, FFTs, DNN training
- **R&D Thrust 2:** System-level ModSim of heterogeneous HPC systems
  - Ex: Emerging HPC system: Some combination (CPU, accelerators, compute cache**)

**Compute cache of CnM/CiM devices

Outline

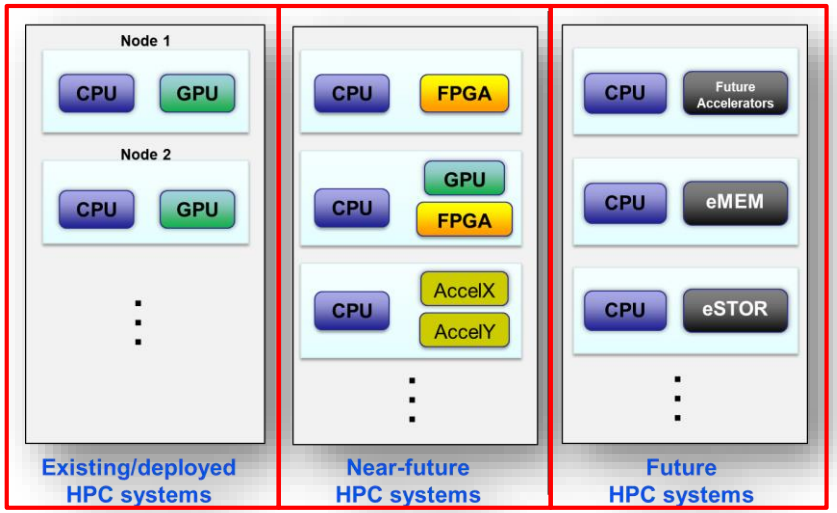
- Overview: Compute caches
  - Compute near Memory (CnM), Compute in Memory (CiM):
- *System-level* ModSim of *heterogeneous* HPC systems
  - Foundation of our approach
  - Approach: deployed, near-future, future heterogeneous HPC systems
  - Role of compute caches in SLAMS* of such systems
- Current progress
  - Experimental platform:
    - Bittware 520N-MX, Intel Stratix 10 MX FPGA
  - Case study
    - In preparation for Bittware board
    - Preliminary results

* SLAMS: System-level Application Modeling & Simulation

SLAMS[1] of Heterogeneous[2] HPC Systems

Goal: Explore methods, tools, and research directions:

- To meet the challenges of *System-Level* Application Modeling and Simulation of *emerging* and *future heterogeneous* HPC systems



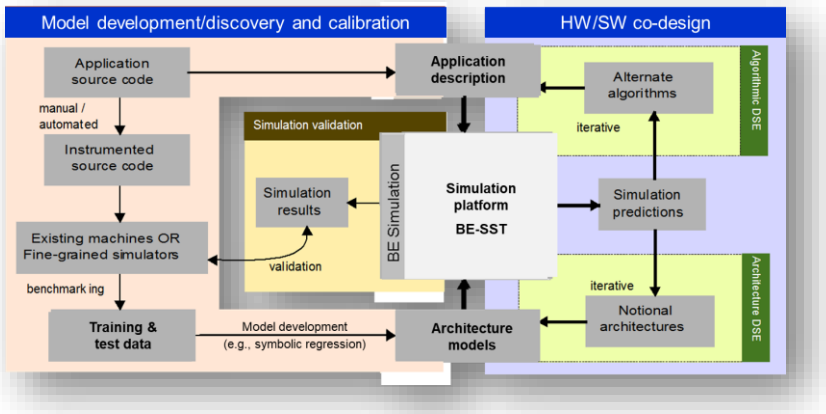
[1] SLAM: System-Level Application Modeling, IARPA-RFI-19-08, https://www.iarpa.gov/index.php/working-with-iarpa/requests-for-information/slam.

[2] 2018 Report for DOE ASCR Basic Research Needs Workshop on Extreme Heterogeneity https://orau.gov/exheterogeneity2018/2018-Extreme- Heterogeneity-BRN-report-final.pdf

- Existing: deployed complete HPC systems; available for benchmarking
- Near-future: no such deployed HPC systems exist yet
  - Containing desired accelerators (e.g., FPGAs)
  - Available nodes (e.g., servers with FPGAs) for benchmarking (ex. Stratix 10 MX)
- Future: no such deployed HPC systems exist yet
  - Device/node not available for benchmarking (eMEM*, eSTOR*, future accelerators)
  - Simulators available to collect data for modeling

* eMem, eSTOR: emerging memory and storage (with CiM capabilities)

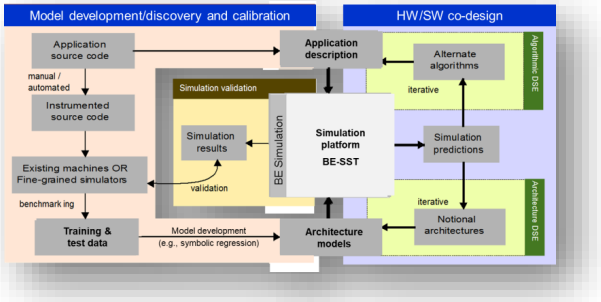
Foundation: SLAMS of Hetero. HPC Systems



Based on current work at Center of Compressible Multiphase Turbulence (CCMT) at the University of Florida (for *homogeneous (CPU-only)* HPC systems)

- **Behavioral Emulation (BE):** a coarse-grained ModSim methodology for design-space exploration of applications on existing (e.g., Vulcan, Titan, Quartz) and notional HPC systems
- **BE-SST:** a distributed parallel simulation library for Behavioral Emulation, integrated into the Structural Simulation Toolkit (SST) from Sandia National Labs

BE-SST Design Goals & Features



1. **Balance** of modeling *fidelity* and simulation *speed* (timeliness).

   Solution: *coarse-grained* ModSim approach

2. **System-level** modeling and simulation
   - Need to capture difficult-to-model app and machine-specific behaviors, without requiring deep knowledge of app and machine

   Solution: *symbolic regression* modeling methods and tools

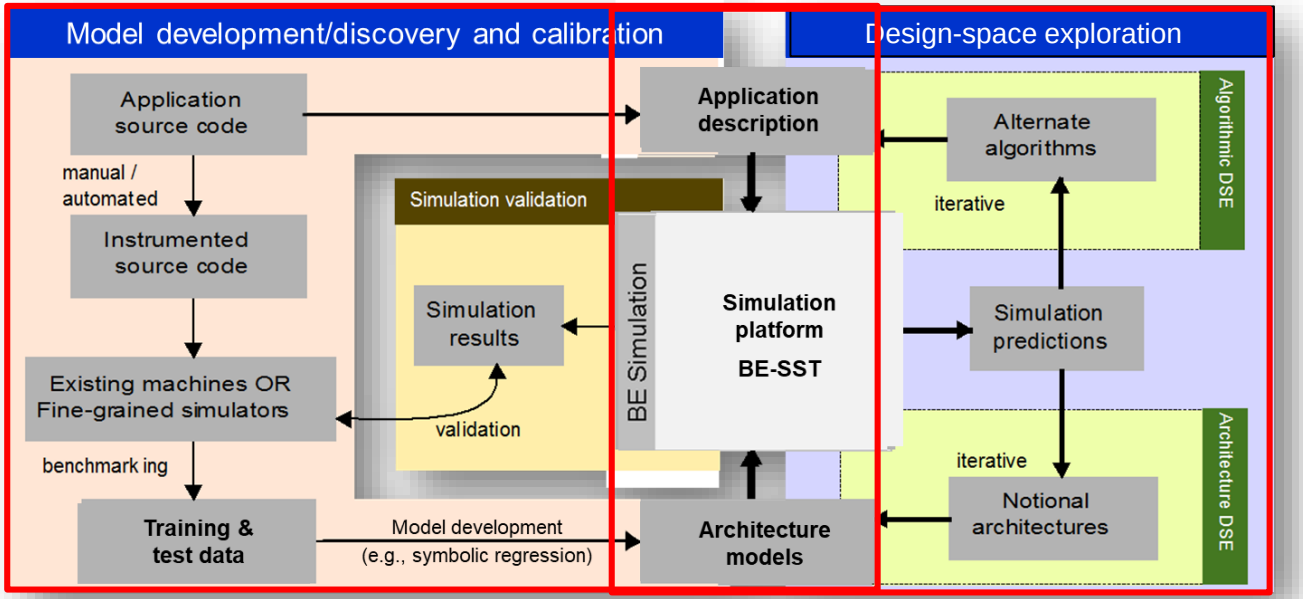
3. **ModSim** tools and system which are *scalable* and *useable* for other interested users.

   Solution: integration with *SST (BE-SST)*

BE Workflow

HW/SW co-design

Algorithmic & arch. design-space exploration (DSE)



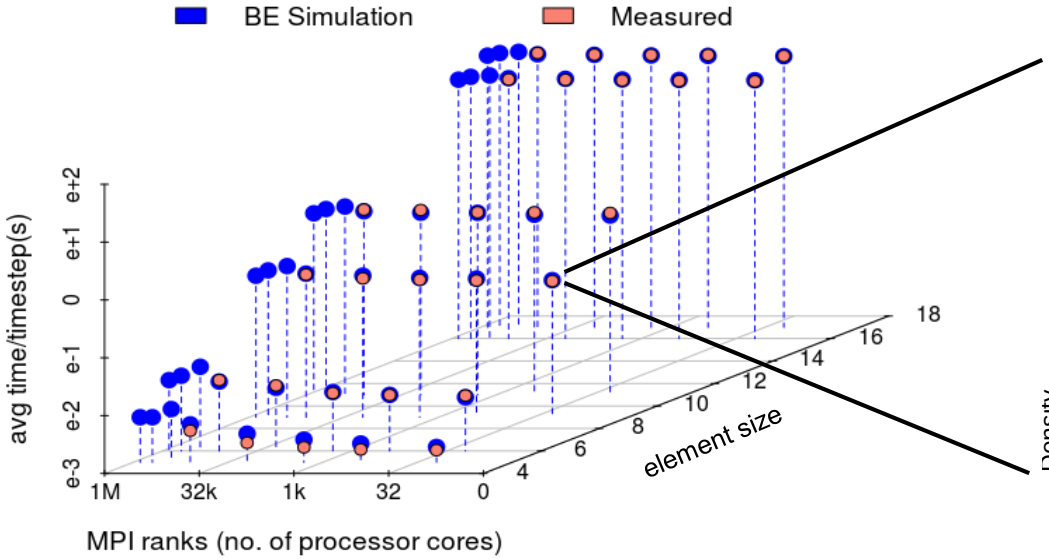
BE-

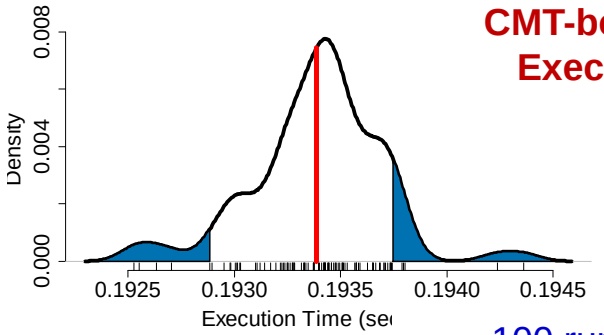
- ***Profile*** code to determine key computational and communication kernels
- ***Instrument & benchmark*** code on *HPC systems* to collect training & test data (currently only homogeneous (CPU-only) HPC systems)
- ***Develop/discover models*** using training data to (e.g., *symbolic regression*)
- ***Calibrate models*** if necessary: compare simulation results vs. test data

Design-space exploration phase:

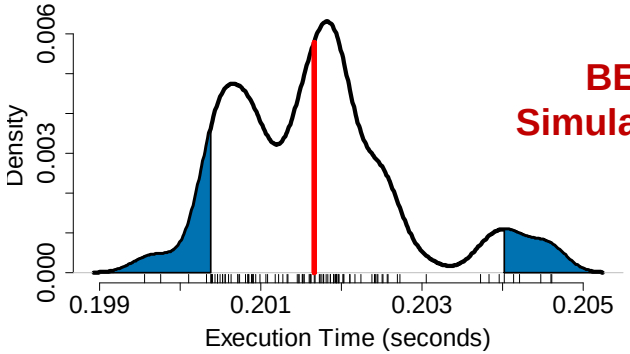
- Algorithmic & architectural design-space exploration (DSE)

BE Simulations of Titan+ *(sample output)*





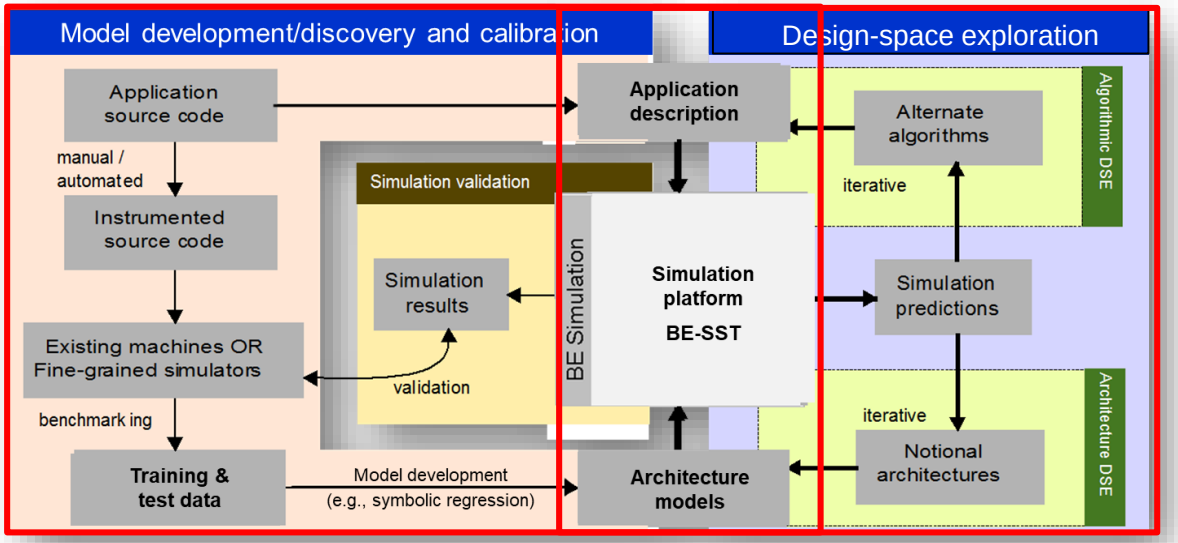
100 runs &
100 simulations



- Simulating bigger system than Titan (approaching 1 Million cores)
- Average % error between CMT-bone-BE simulation and execution time is 4%
- Maximum error is 17%

$$\%\ error = \frac{\frac{\sum_{i=1}^{N} t_{measured\ i}}{N} - \frac{\sum_{i=1}^{N} t_{simulated\ i}}{N}}{\frac{\sum_{i=1}^{N} t_{measured\ i}}{N}} \times 100$$

What BE/BE-SST are NOT



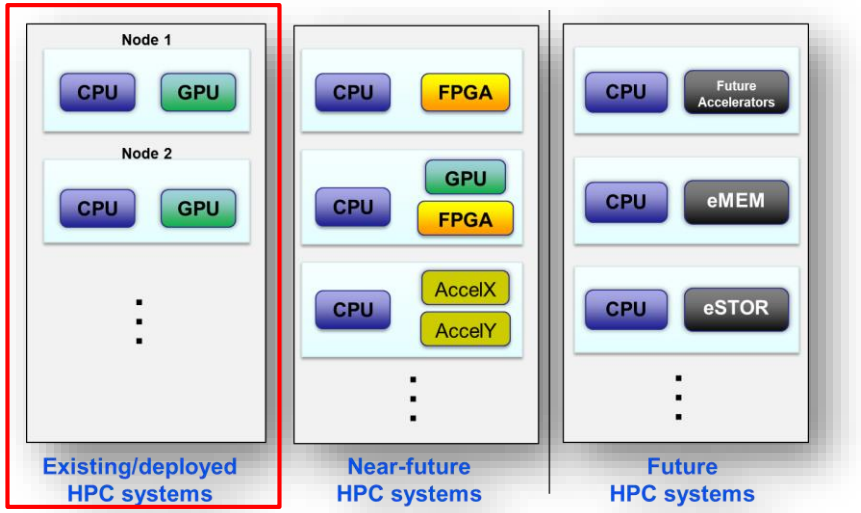
1. **Not** to "**componentize**" and model **every detail** of a complex system.
   - Unless required, difficult to model machine-specific behaviors abstracted and discovered through use of training data & *symbolic regression*
   - Components under study are modeled in detail
2. **Not** claim to solve the **general** co-design DSE problems
   - Focus: application DSE on given architecture and its notional variations

Outline

- Overview
  - Compute near Memory (CnM); Compute in Memory (CiM)
  - Research directions D1, D2
- *System-level* ModSim of *heterogeneous* HPC systems
  - Foundation of our approach
  - Approach: deployed, near-future, future heterogeneous HPC systems
  - Role of compute cache in SLAMS* of such systems
- Current progress
  - Experimental platform:
    - Bittware 520N-MX, Intel Stratix 10 MX FPGA
  - Case study
    - In preparation for Bittware board
    - Preliminary results

* SLAMS: System-level Application Modeling & Simulation

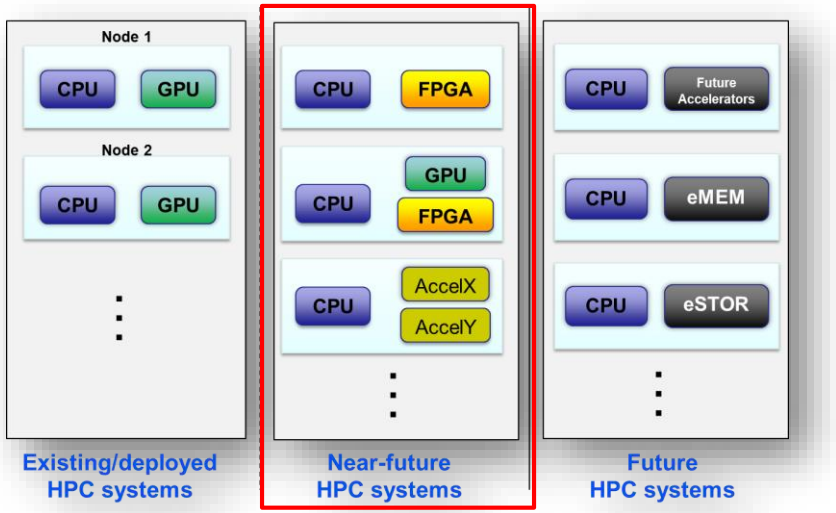
SLAMS Approach: Existing HPC Systems



Deployed *heterogeneous* HPC system

- Existing CPU+GPU HPC systems (e.g., Summit, Titan).
- Available for benchmarking
- We can use existing BE method for system-level ModSim of such systems
  - Instrument & benchmark code on nodes of HPC systems to collect training/test data
  - Develop/discover models (e.g., symbolic regression)
  - Perform design-space exploration

SLAMS Approach: Near-Future HPC Systems

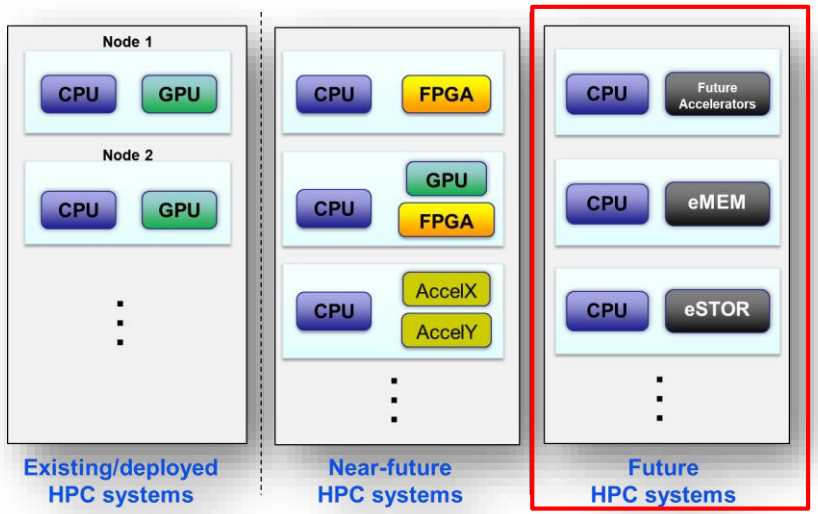


Near-future *heterogeneous* HPC system

- No such deployed heterogeneous HPC systems yet (e.g., CPU+FPGA, CPU+GPU+FPGA)
- But independent nodes for such systems do exist (e.g., servers with FPGA boards
- Approach:
  - Instrument & benchmark code on available nodes to collect training/test data
  - Develop/discover models (e.g., symbolic regression)
  - Perform design-space exploration

BittWare 520N-MX: w/ Stratix 10 MX for compute-near-memory (CnM) research

SLAMS Approach: Future HPC Systems



Future HPC systems (augmented w/eMEM*, eSTOR*, future accelerators)

- Devices and nodes for future systems do not exist
- Cannot collect benchmarking data at the node level of such systems
- Approach
  - Depend on simulators to generate training/test data
  - Develop/discover models (e.g., symbolic regression)
  - Perform design-space exploration

Compute-in-memory (CiM) research

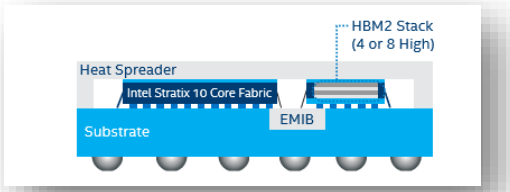
* eMem, eSTOR*: emerging memory and storage (with CiM capabilities)

Outline

- Overview
  - Compute near Memory (CnM); Compute in Memory (CiM)
  - Research directions D1, D2
- *System-level* ModSim of *heterogeneous* HPC systems
  - Foundation of our approach
  - Approach: deployed, near-future, future heterogeneous HPC systems
  - Role of compute cache in SLAMS* of such systems
- Current progress
  - Experimental platform:
    - Bittware 520N-MX, Intel Stratix 10 MX FPGA
  - Case study
    - In preparation for Bittware board
    - Preliminary results

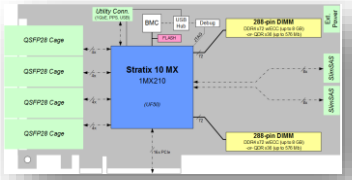
* SLAMS: System-level Application Modeling & Simulation

Stratix 10 MX Board



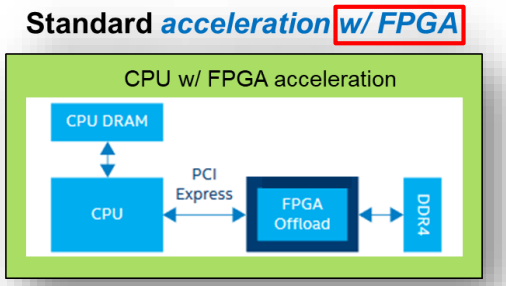
Bittware 520N-MX

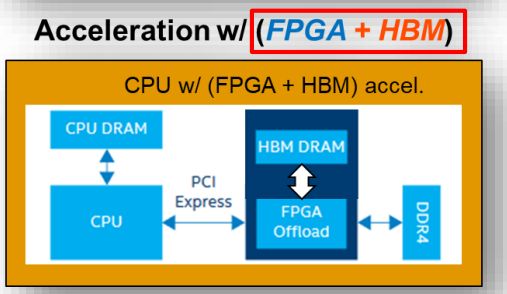
- Intel Stratix 10 MX 1MX210 FPGA
  - Up to 2 million logic elements
  - 16 GBytes HBM2 in same package
- High Memory Bandwidth 512 Gbps
- 2 PCIe hard IP blocks
- PCIe Interface - x16 PCIe Gen1, Gen2, Gen3
- Built-in Intel USB-Blaster
- UltraPort SlimSAS™ for serial expansion
- OpenCL support



"Available 2019
~~June/July~~ ~~August~~
Late Sept/early October"

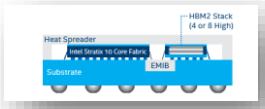
CnM Processing for Kernel and App Acceleration





- FPGAs *effective as accelerators* for many data-analytics apps & kernels
  - As long as problem size can fit into the FPGA
- **Compute**-*near-Memory* **(CnN):** *Amplify* acceleration capabilities of FPGAs by exploiting *FPGA + nMEM**
  - FPGA-accelerated computation
  - High-bandwidth, lower-latency access to HBM2 cache

Stratix 10 MX board
Stratix 10 + 16GB HBM2
(in same package)



(Available 4th quarter 2019)

Apps Under Study

High-bandwidth cache

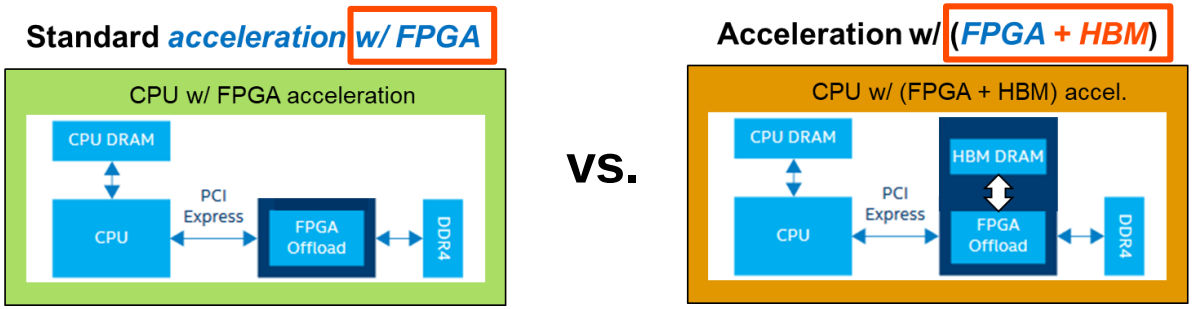
- Very Large (VL) matrix multiply: dense linear algebra
- VL FFT
- Deep neural nets

Random access

- Bitonic sort: graph traversal
- BFS: graph traversal search
- Bloom filter: large volume of small random accesses

* nMEM: near-memory e.g., Stratix 10 MX with in-package16 MB HBM2

Example Case Study CnM: FFT on 520N-MX



Compute near Memory:

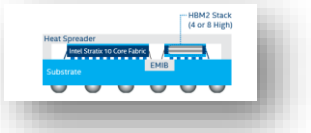
FPGA vs. FPGA + nHBM:

- Improve performance
- and/or FFT size

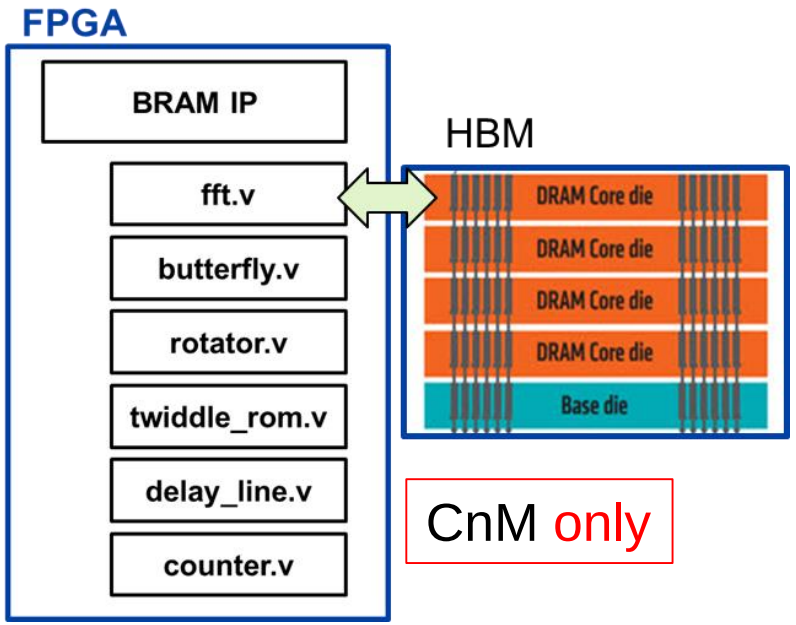
Stratix 10 MX board

Stratix 10 + 16GB HBM2

(in same package)

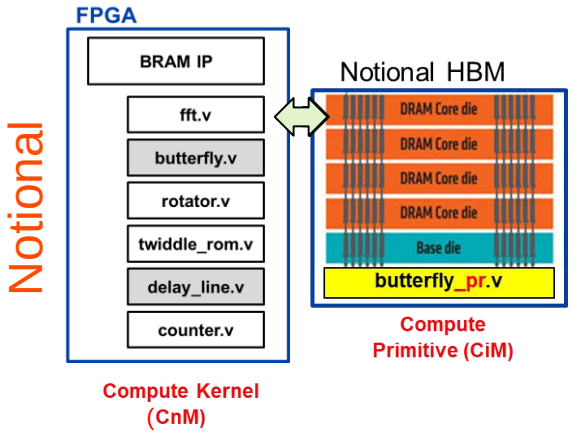


(Available 4th quarter 2019)



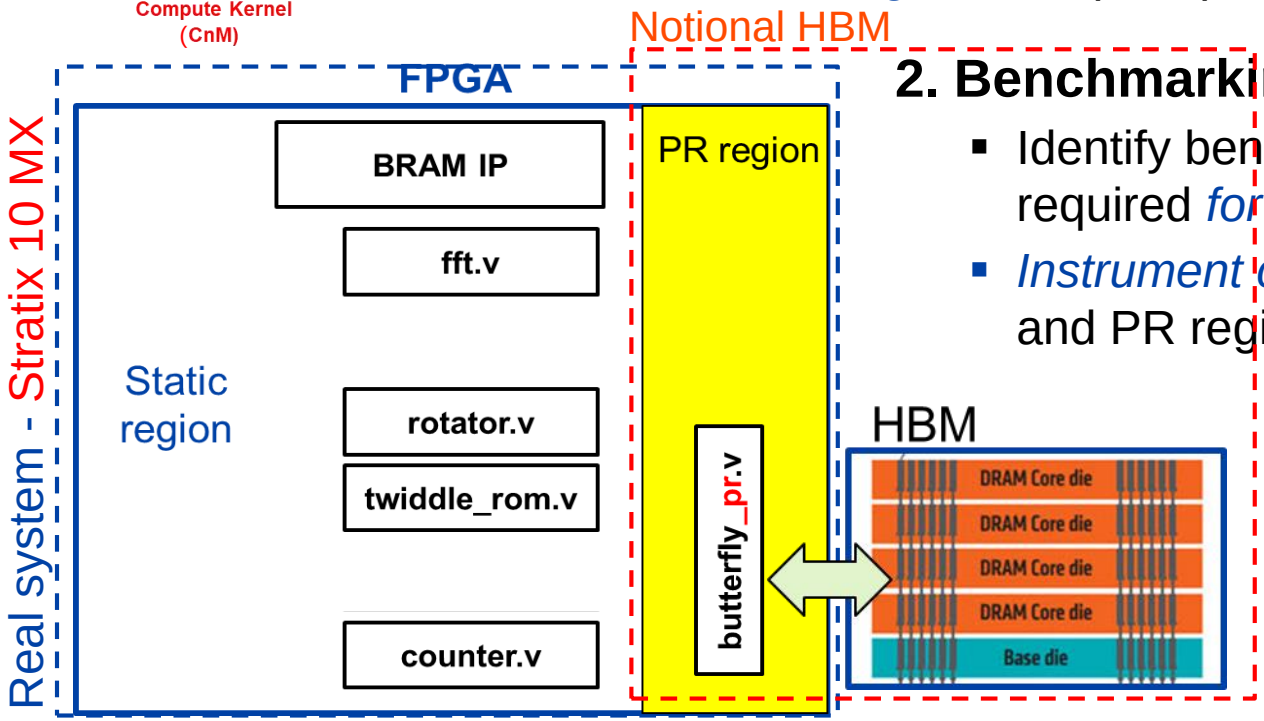
CnM only

CnM+CiM: 520N-MX Prototype & Benchmarking



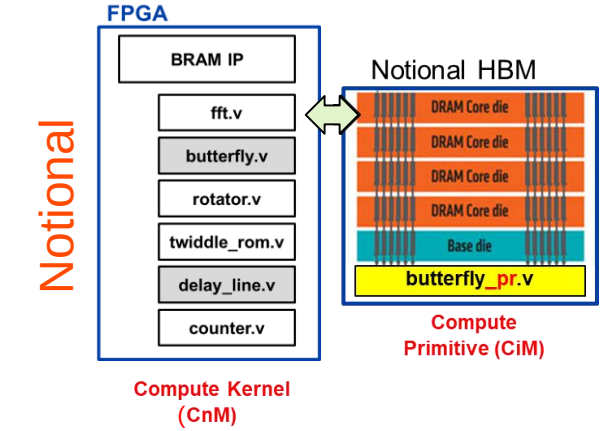
Approach:

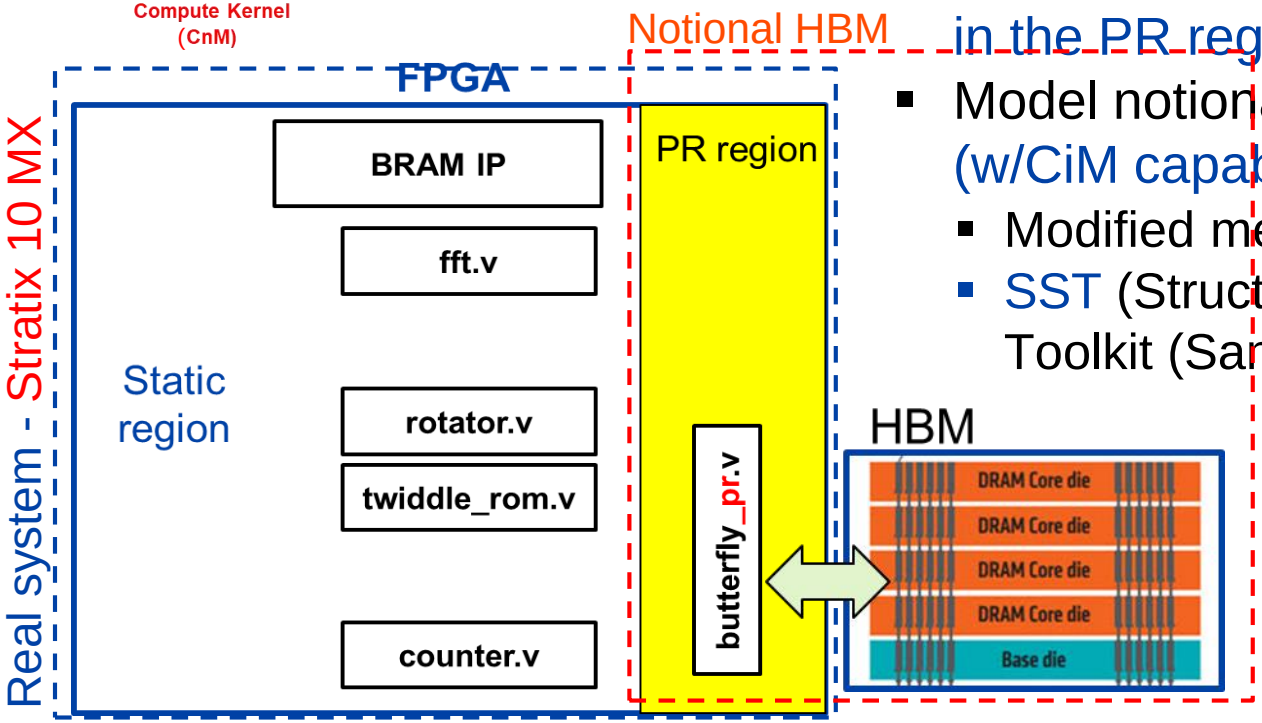
1. **Experimental prototype** (Statix10 MX)
   - Implement totally in FPGA, *using PR*
   - Static region: compute kernel
   - *PR region:* compute primitive(s)



2. **Benchmarking/data collection**
   - Identify benchmarking data required *for model/simulation*
   - *Instrument code* (both in static and PR regions) to collect data

CnM+CiM: Model & Simulation

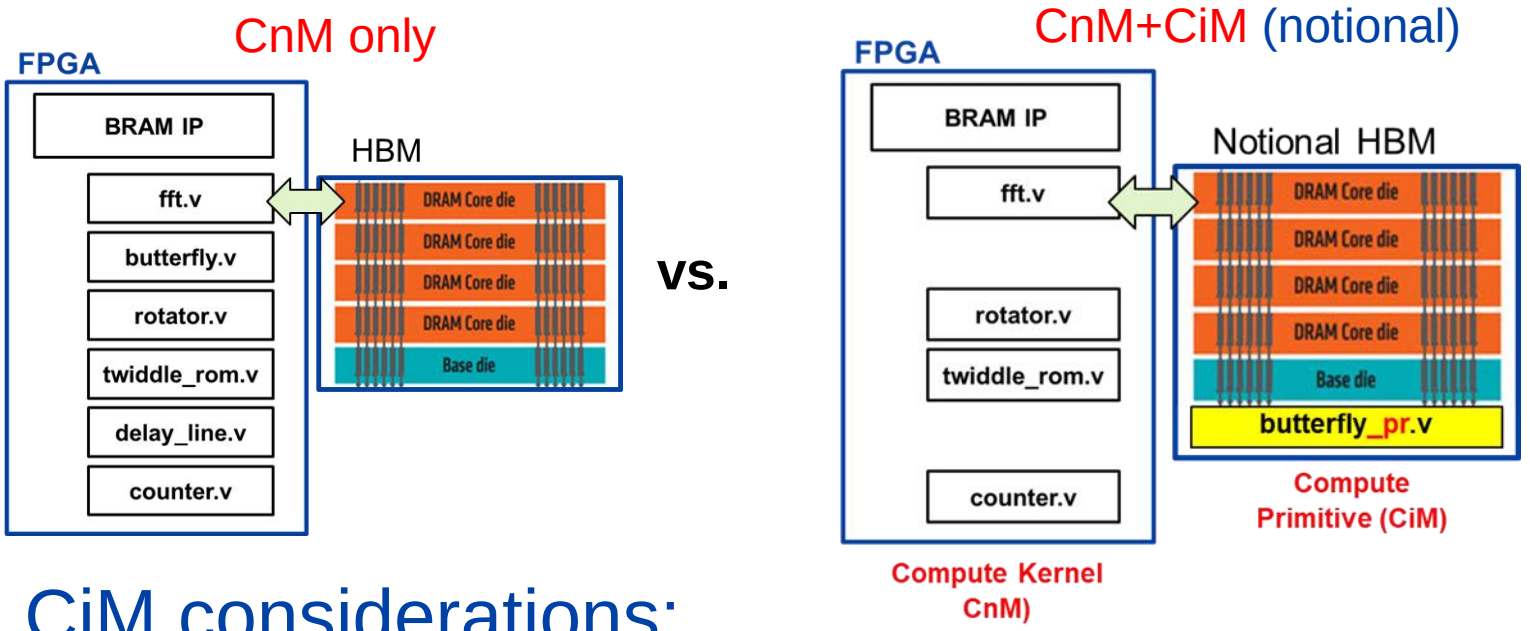




Approach:

1. **Experimental prototype**
2. **Benchmarking/data collection**
3. Model & Simulation
   - Model the data benchmarked in the PR region
   - Model notional memory (w/CiM capability)
     - Modified memory simulator?
     - SST (Structured Simulation Toolkit (Sandia)?
   - Design-space exploration: CiM algorithms & architectures

App/Kernel of Interest: CiM-amenable?



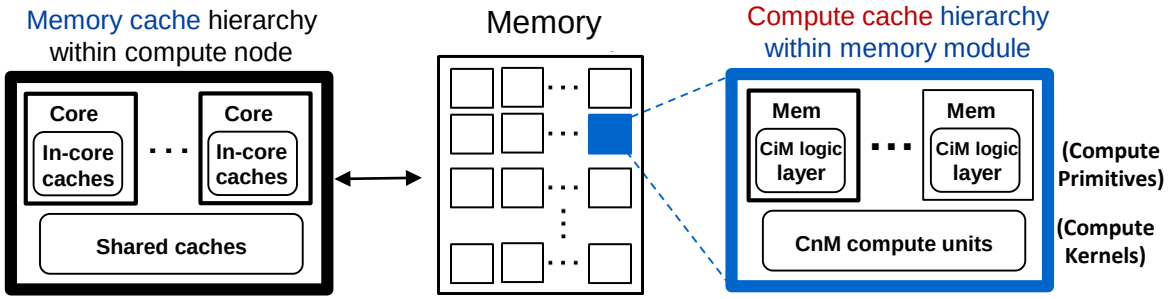
CiM considerations:

- Can algorithm be *re-factored* to take advantage of CnM+CiM arch?
- Compute primitive characteristics
  - light-weight ops; reduces data transfer; not frequent interacting with compute kernel, can be pipelined with compute kernel ops, … what else?…

Summary: Compute Cache

Compute cache architecture for *heterogeneous* HPC system

- Memory cache hierarchy - bring *memory close to compute* devices
  - In-core caches & near-core shared caches



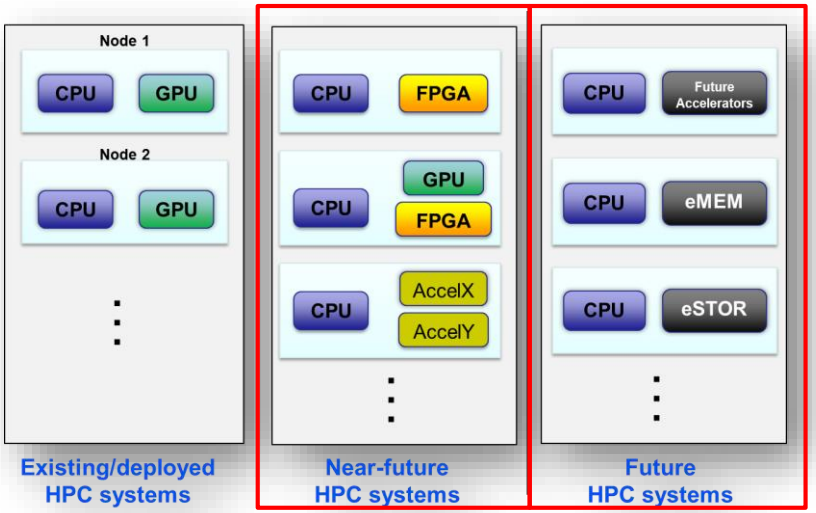
- Complementary compute cache hierarchy – bring *compute close to memory*
  - CiM logic – *compute primitives* (e.g., add, mult, data-ordering ops)
  - CnM compute – *compute kernels* (e.g., FFT, Bloom filter)

- **Goal:** *System-level* ModSim of emerging *heterogeneous* HPC systems
  - Emerging HPC system: Some combination (CPUs, accelerators, compute caches)

SLAMS[1] of Heterogeneous[2] HPC Systems

Going forward: Explore methods, tools, and research directions:

- To meet the challenges of *System-Level* Application Modeling and Simulation of *emerging* and *future heterogeneous* HPC systems



[1] SLAMS: System-Level Application Modeling and Simulation, IARPA-RFI-19-08, https://www.iarpa.gov/index.php/working-with-iarpa/requests-for-information/slam.

[2] 2018 Report for DOE ASCR Basic Research Needs Workshop on Extreme Heterogeneity https://orau.gov/exheterogeneity2018/2018-Extreme- Heterogeneity-BRN-report-final.pdf

- Existing: deployed complete HPC systems; available for benchmarking
- Near-future: deployed HPC systems
  - Without desired accelerators (e.g., FPGAs)
  - Available nodes (e.g., servers with FPGAs) for benchmarking (ex. Stratix 10 MX)
- Future: deployed HPC systems
  - Device/node not available for benchmarking (eMEM, eSTOR, future accelerators)
  - Simulators available to collect data for modeling

Outline

- Overview
  - Compute near Memory (CnM); Compute in Memory (CiM)
  - Research directions D1, D2
- *System-level* ModSim of *heterogeneous* HPC systems
  - Foundation of our approach
  - Approach: deployed, near-future, future heterogeneous HPC systems
  - Role of compute cache in SLAMS* of such systems
- Current progress
  - Experimental platform:
    - Bittware 520N-MX, Intel Stratix 10 MX FPGA
  - Case study
    - In preparation for Bittware board
    - Preliminary results

* SLAMS: System-level Application Modeling & Simulation


DISCUSSION