The HammerBlade: An ML-Optimized Supercomputer for ML and Graphs

Prof. Michael B. Taylor (PI)
University of Washington

Prof. Adrian Sampson
Cornell University

Prof. Luis Ceze
University of Washington

Prof. Chris Batten
Cornell University

Prof. Mark Oskin
University of Washington

Prof. Zhiru Zhang
Cornell University

Dec 2018

Dr. Dustin Richmond (Postdoc)
University of Washington

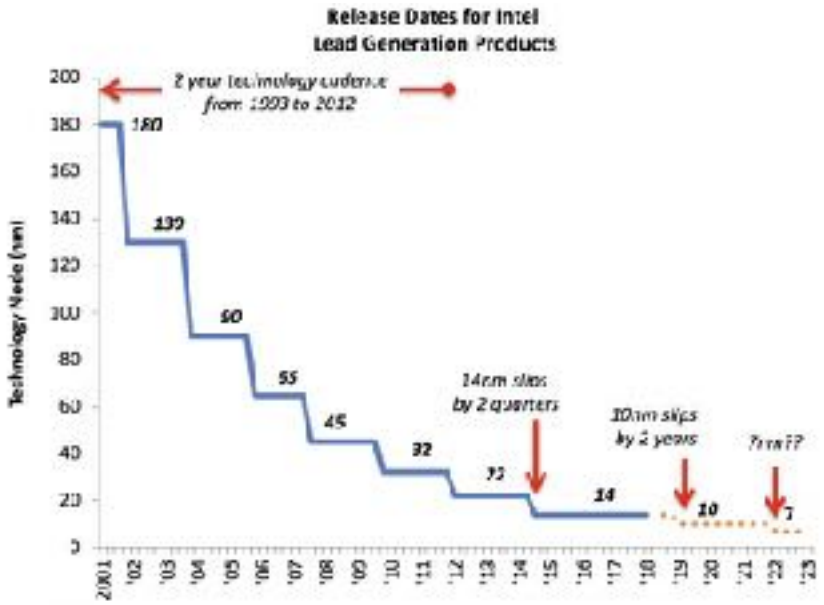
Fast Intro to Today's HW Landscape

The End of Moore's Law Approaches

Dennard Scaling Ended a Decade Ago

Energy is a fundamental limiter of all compute

Specialization Is the Solution



HammerBlade: Key Insights

Key Intellectual Thrusts of The HammerBlade

How do we solve HW & SW Specialization Complexity?

Move from Human-Centric Computation Abstraction Hierarchy

To a ML-Centric Computation Abstraction Hierarchy

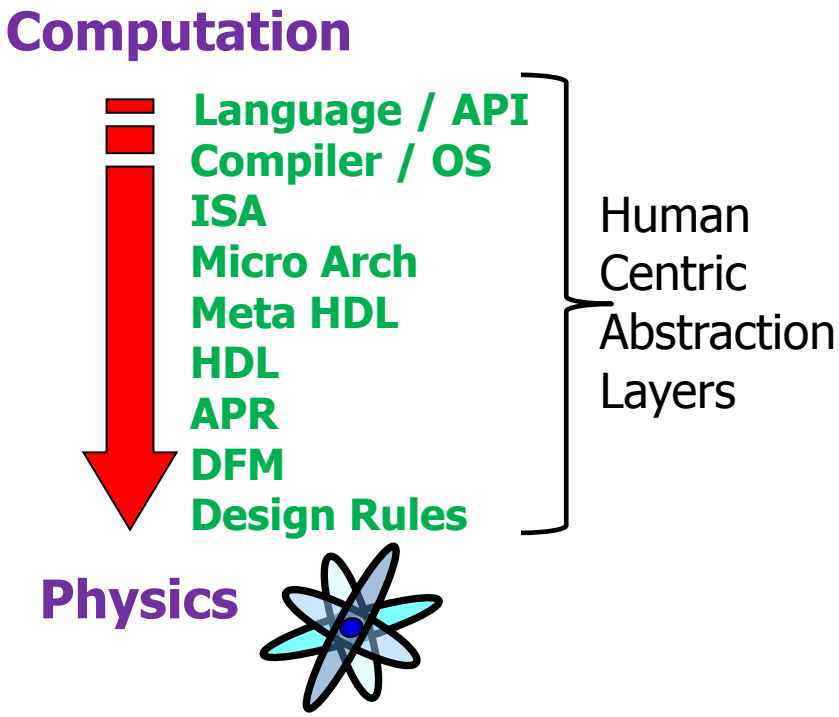
HammerBlade: Key Insights

Key Intellectual Thrusts of The HammerBlade

How do we solve HW & SW Specialization Complexity?

Move from Human-Centric Computation Abstraction Hierarchy

To a ML-Centric Computation Abstraction Hierarchy



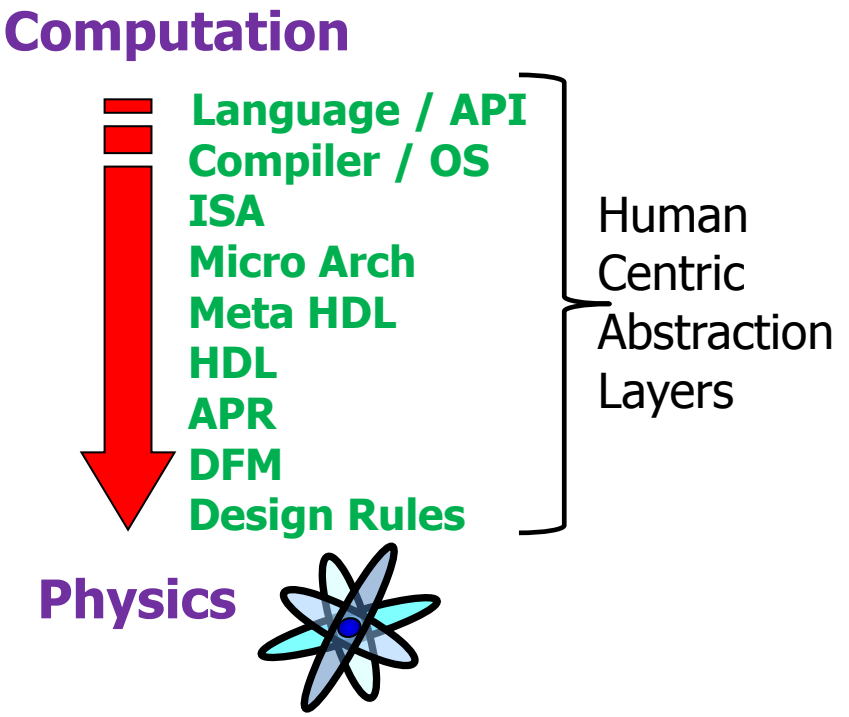
HammerBlade: Key Insights

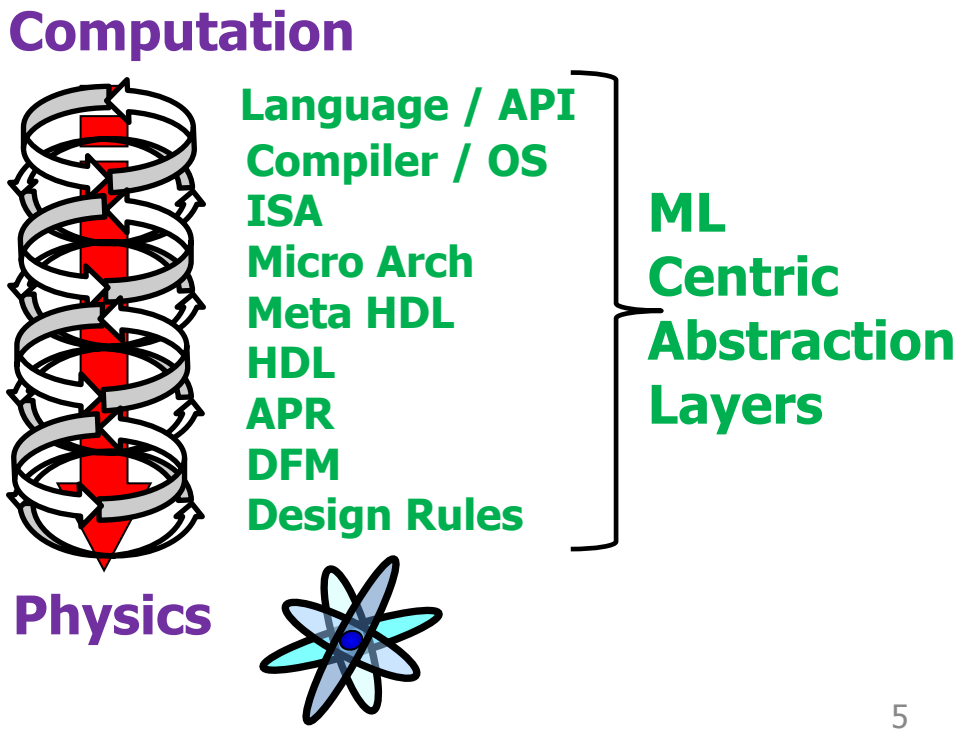
Key Intellectual Thrusts of The HammerBlade

How do we solve HW & SW Specialization Complexity?

Move from Human-Centric Computation Abstraction Hierarchy

To a ML-Centric Computation Abstraction Hierarchy





HammerBlade: Key Insights

Key Intellectual Thrusts of The HammerBlade

How do we solve HW & SW Specialization Complexity?

Move from Human-Centric Computation Abstraction Hierarchy

To a ML-Centric Computation Abstraction Hierarchy

→ Redesign the compute stack knowing that Machine Learning Will Drive How Computation is Realized in HW & SW

HammerBlade: Key Insights

Key Intellectual Thrusts of The HammerBlade

How do we solve HW & SW Specialization Complexity?

Move from Human-Centric Computation Abstraction Hierarchy

To a ML-Centric Computation Abstraction Hierarchy

→ Redesign the compute stack knowing that Machine Learning Will Drive How Computation is Realized in HW & SW

ML Co-designing HW/SW .. for ML

ML Co-designing HW/SW .. for Graphs

ML Co-designing HW/SW .. for Graphs & ML

Key Intellectual Thrusts of The HammerBlade

How do we solve HW Specialization's Inflexibility?

Seamless blend of specialization at multiple levels, with a focus on tight interoperability...

- CGRA
- FPGA
- ASIC Hard Blocks
- RISC-V CPUs
- Memory System
- Interconnect

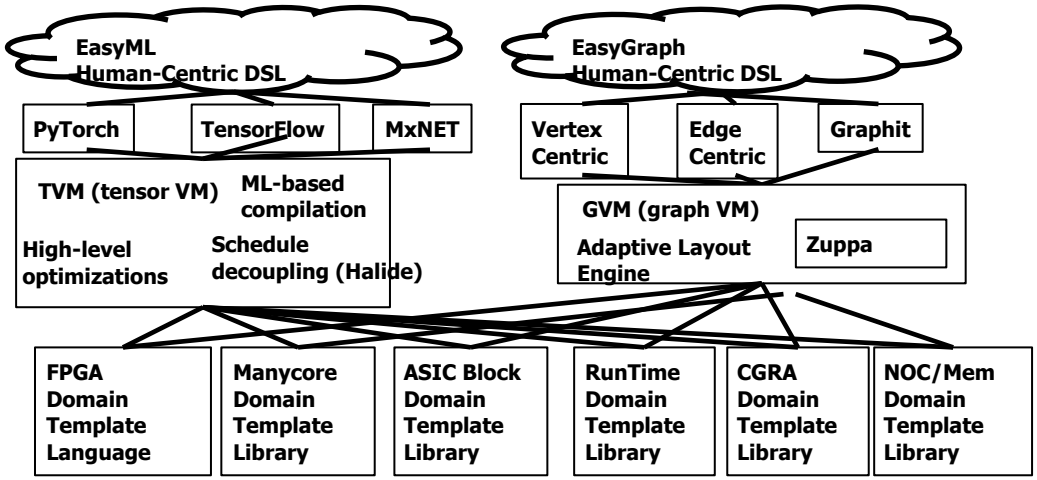
Dark Silicon: Not everything is on; Target metric is energy-efficiency not utilization

Key Intellectual Ideas of The HammerBlade

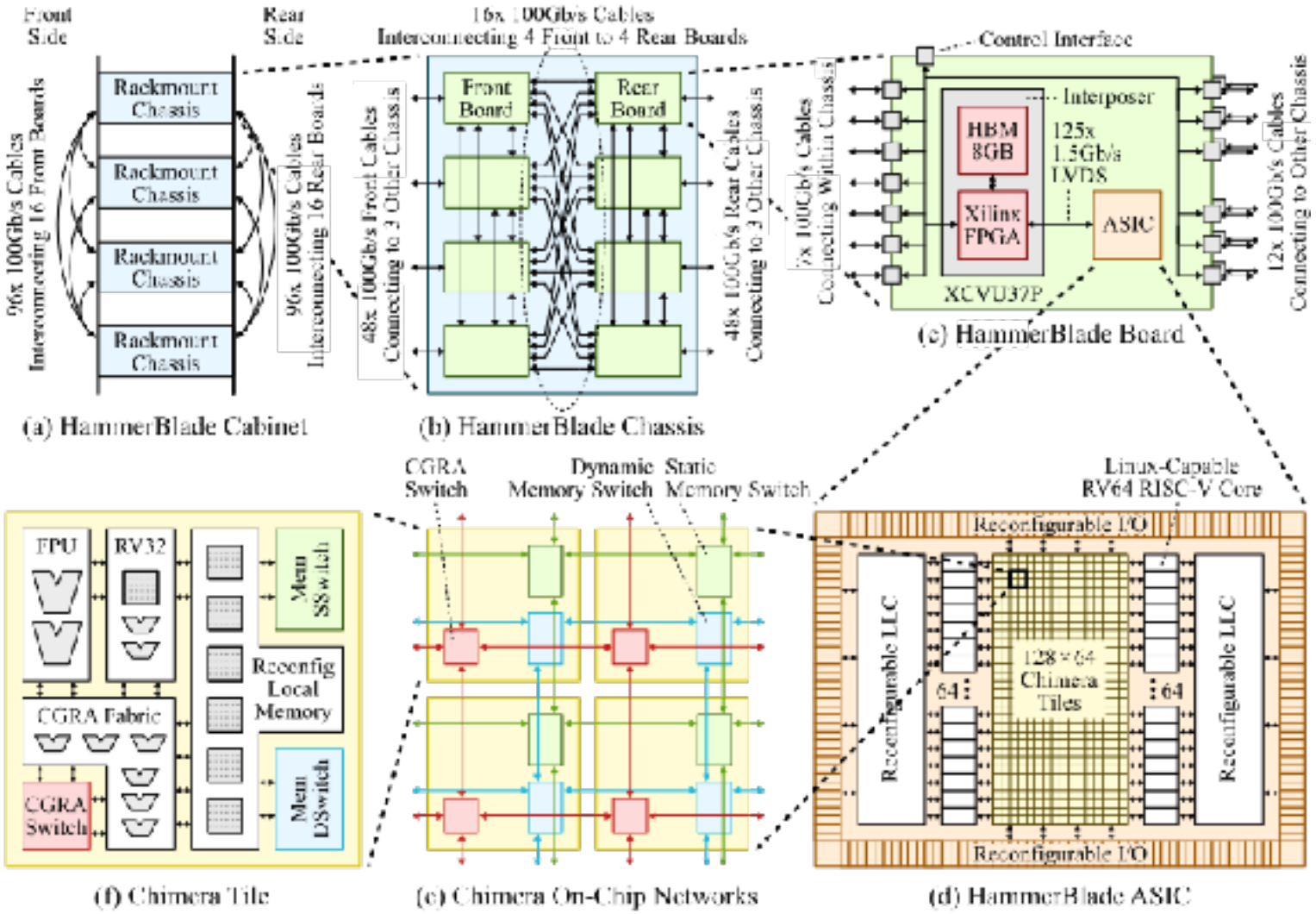
How do we address the long binding times of specialization in response to changing datasets?

Move from year/month/day specialization times to minutes/secs/microsec?

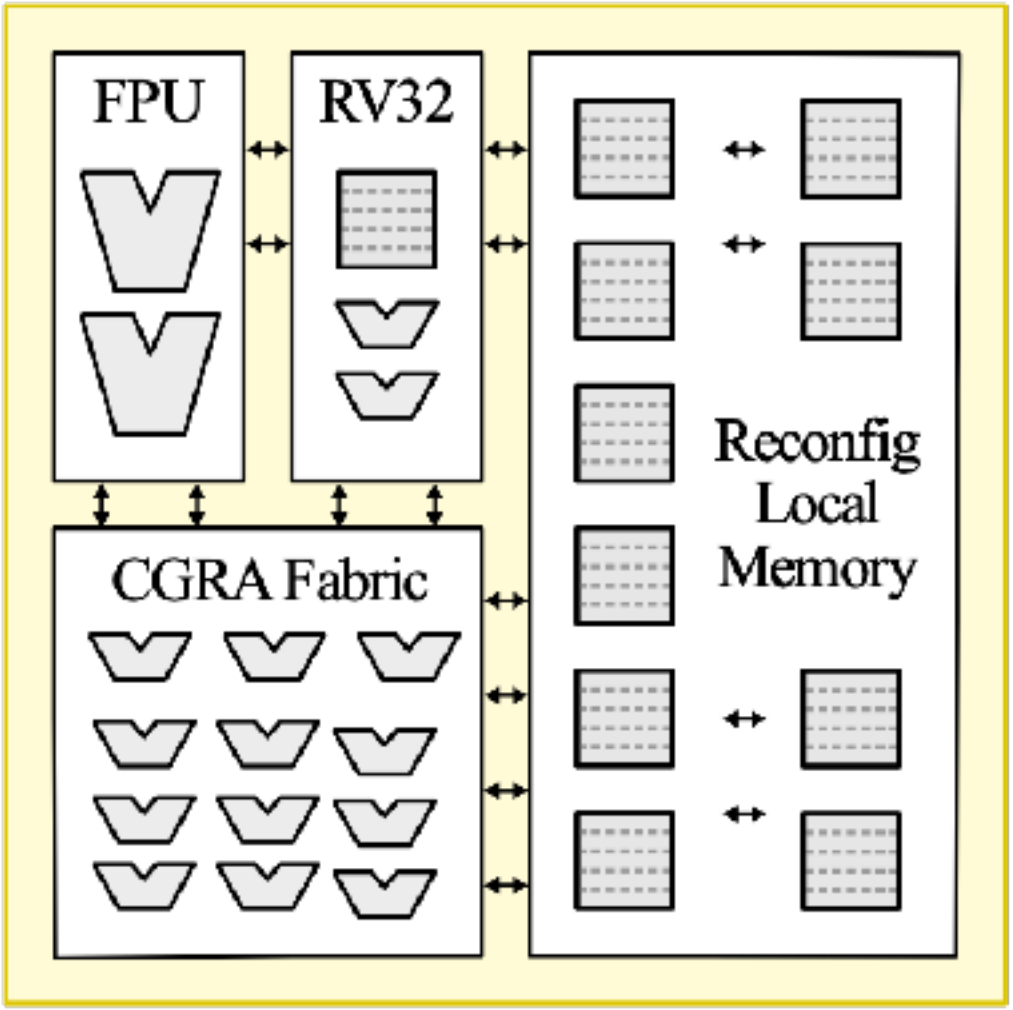
ML-stitching of ML-predesigned Fabric & Domain Specific Templates



The HammerBlade Hardware Architecture



HammerBlade Chimera Tile



ML-Designed CGRA Fabric (Incl. ASIC Hard Blocks)

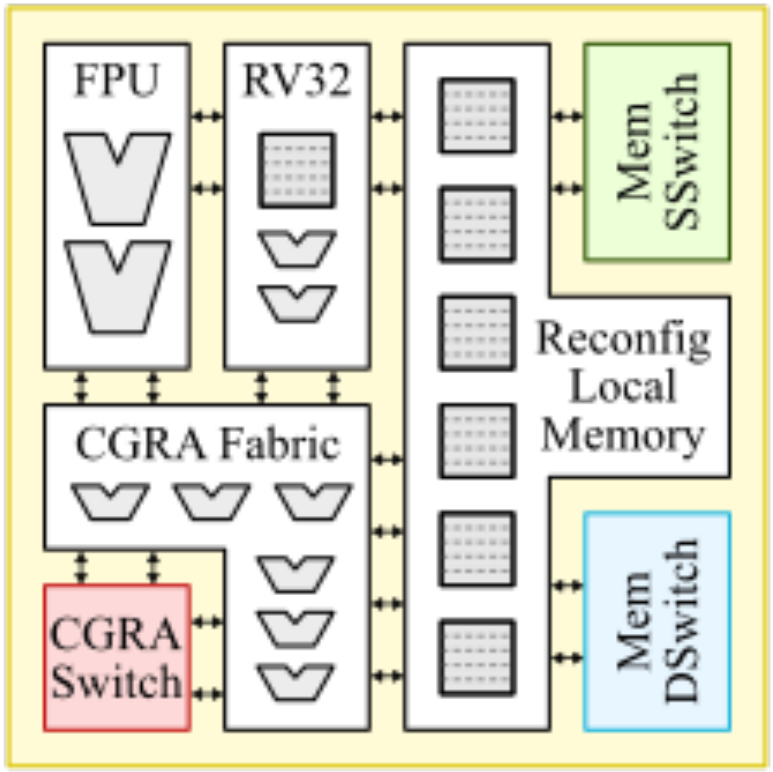
RISC-V RV32 Cores

ML-Tuned FPUs

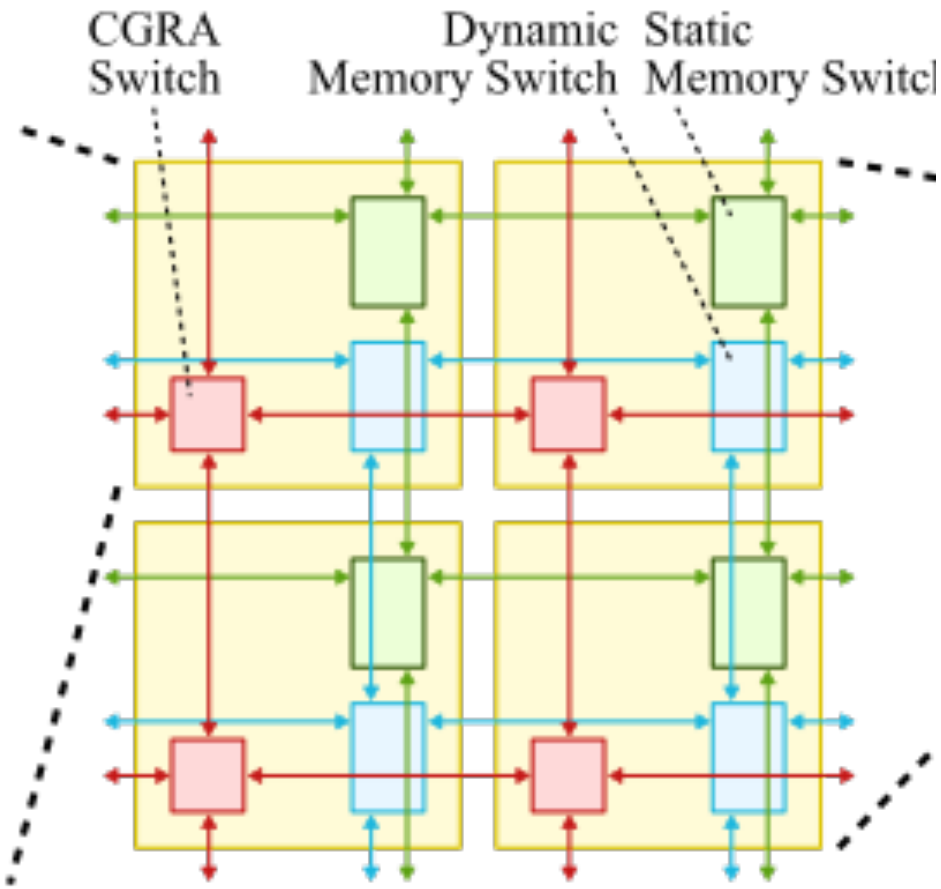
ML-Configured Reconfigurable Local Memory

ML-Programmed Interconnections

Specialized Intertile Network Fabrics



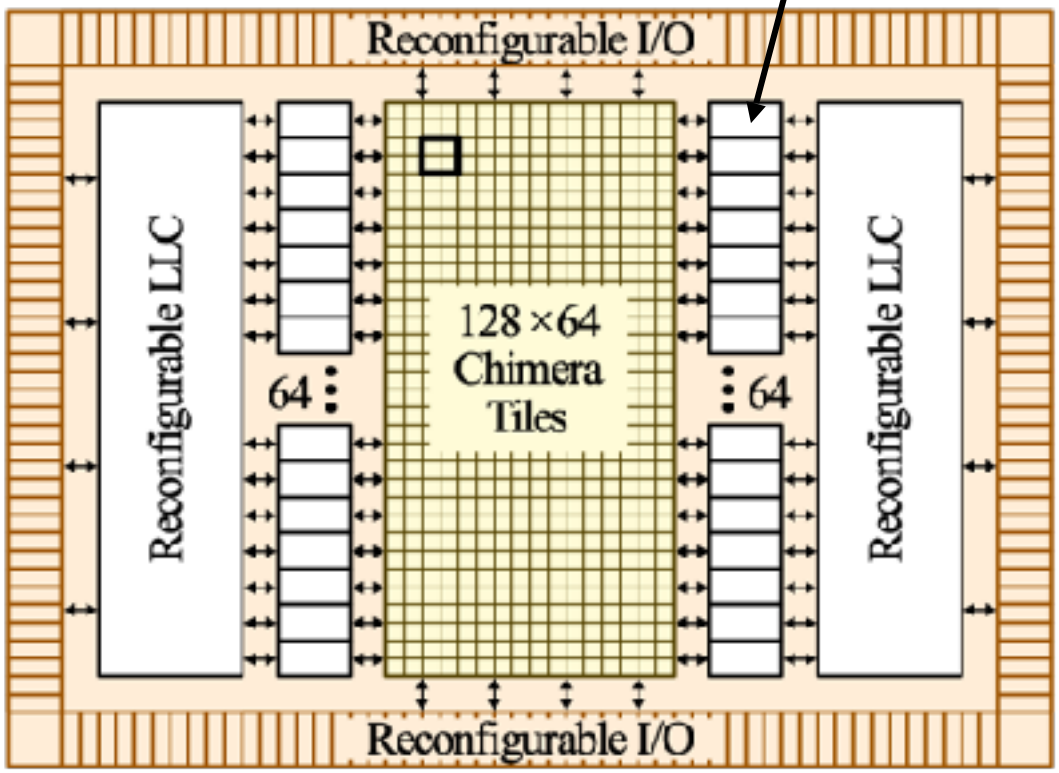
(f) Chimera Tile



(e) Chimera On-Chip Networks

HammerBlade ASIC

Linux-Capable RISC-V RV64G Core – UW/BU Black Parrot; funded by DARPA POSH



8K Chimera Tiles

128 RISC-V 64bit Linux Capable Cores

Reconfigurable LLC

Reconfigurable I/O

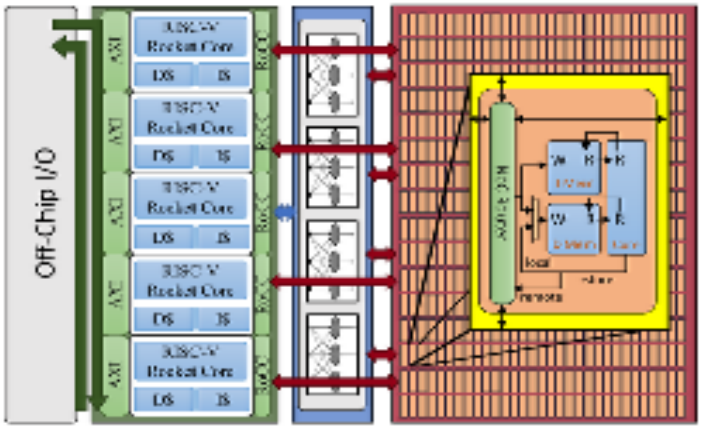
14 & 7nm, large die

HammerBlade Manycore

Leveraging Celerity's Manycore into HammerBlade Manycore/CGRA Hybrid

Celerity (opencelerity.org, IEEE Micro '18 Paper):

Broke RISC-V performance record by 100X
(500B RISC-V ops per sec)
Silicon proven in 16nm. Open Source.
50 processors per mm^2
DARPA CRAFT

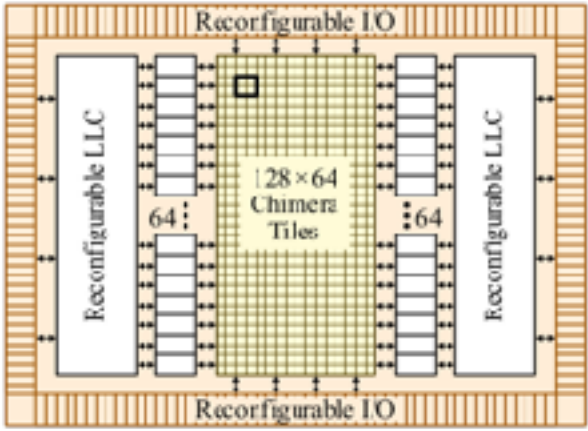


HammerBlade:

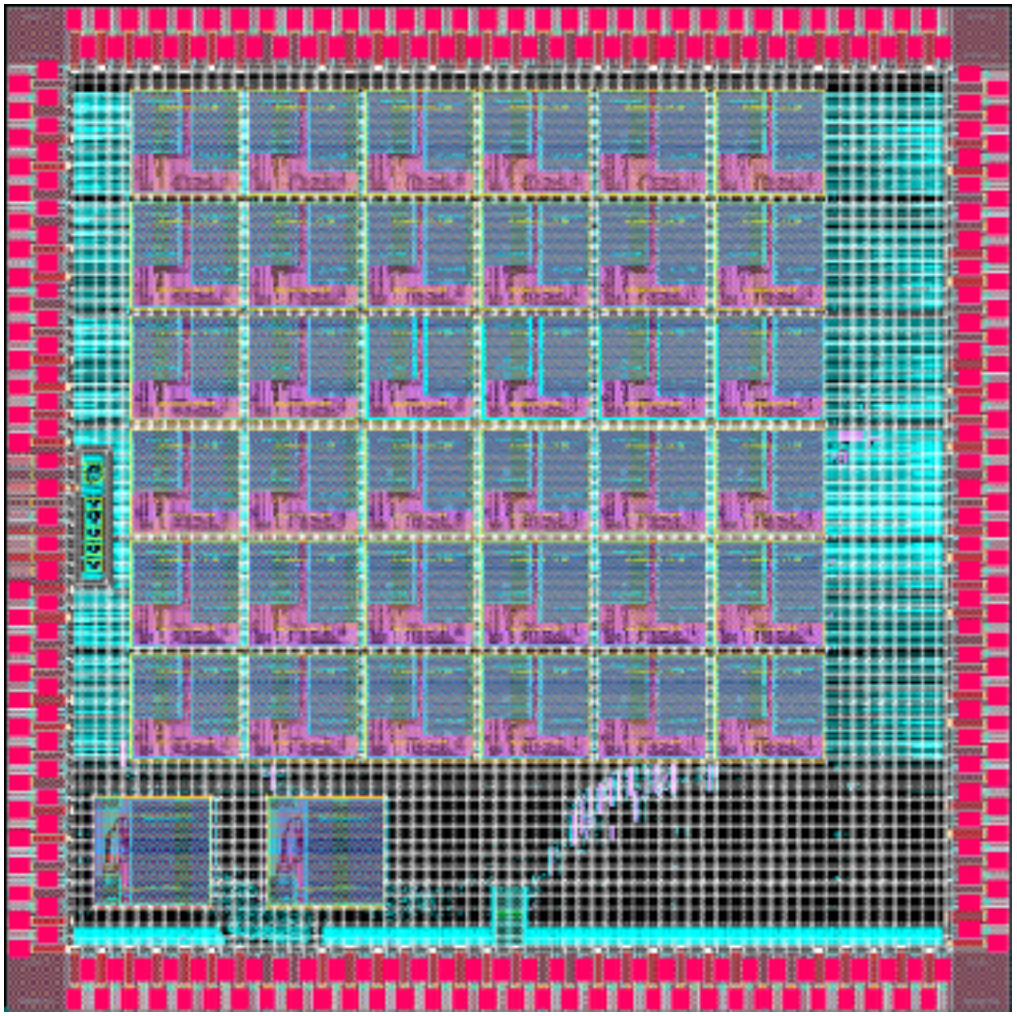
Exponentially better programmability & perf. robustness
I-caches in Chimera Tiles (CTs), initial version
Memory hierarchy, initial version
Latency Hiding in CTs (non-blocking loads & stores)
Unified Physical Address Space, initial version

Implemented?

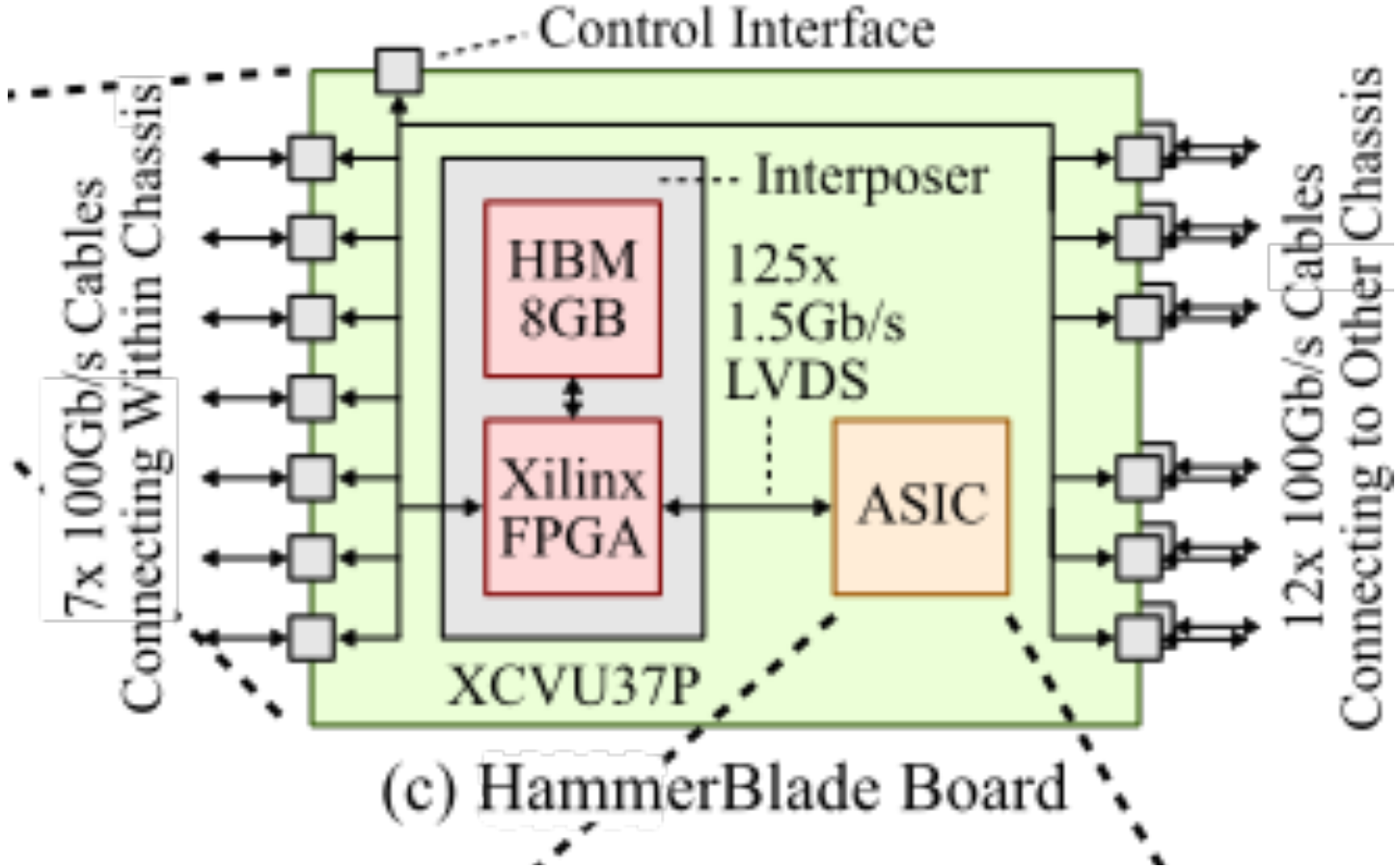
erve amazing compute density and efficiency
: for fully pipelined processor and high performance mesh router
takes less space than 4K of SRAM (!)
Integrate CGRA functionality without tile size explosion



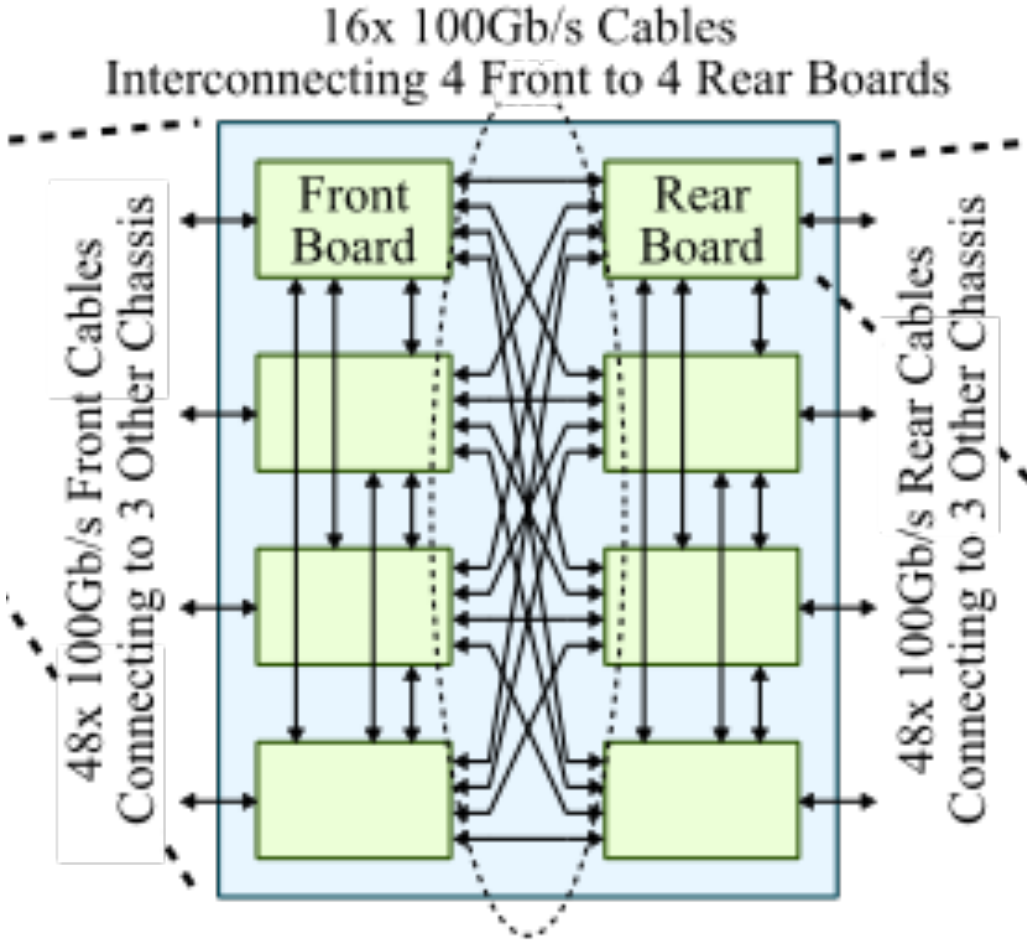
36-tile HammerBlade Proto in TSMC 40nm; enroute to 16nm



HammerBlade PCB

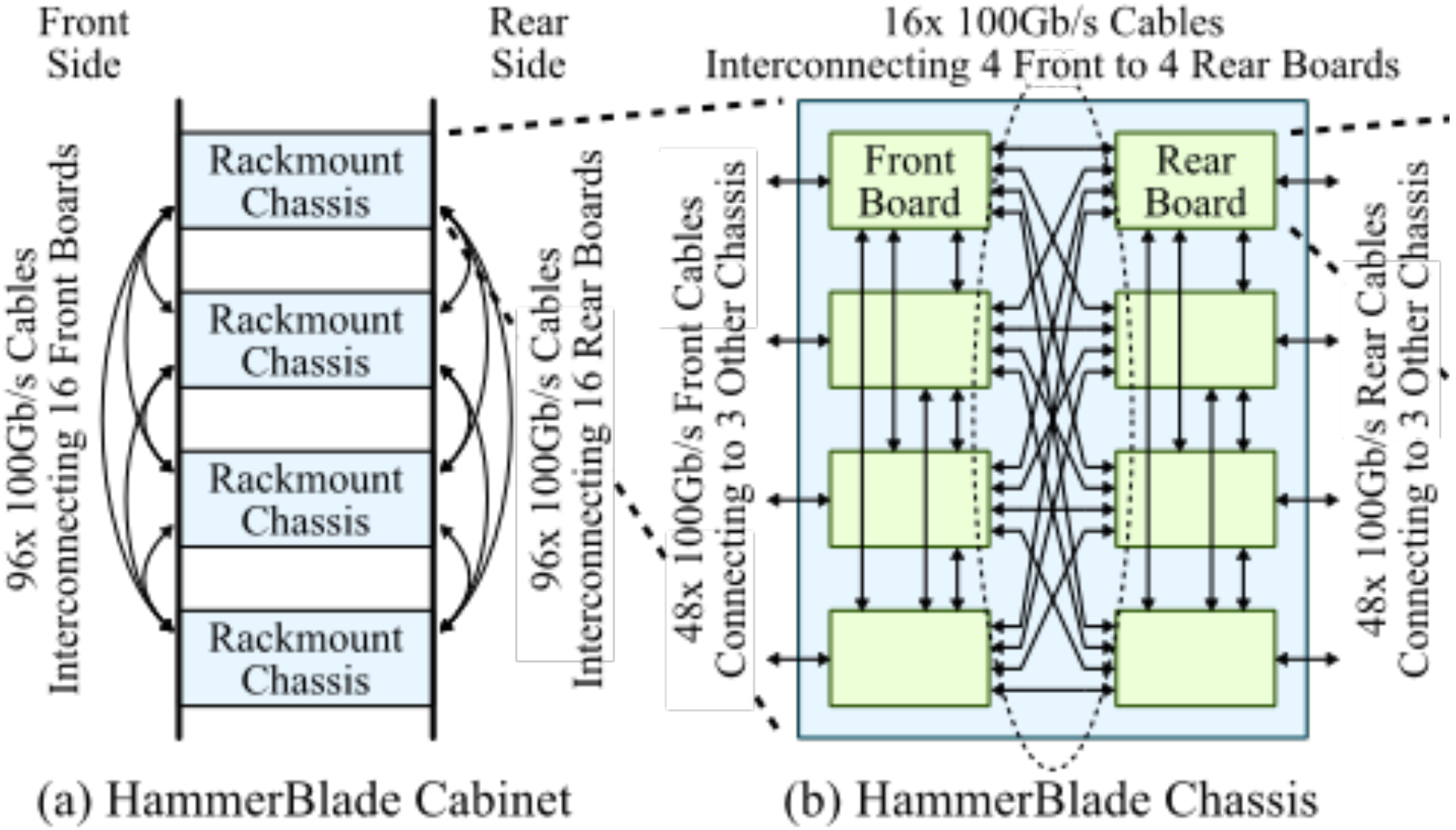


HammerBlade Chassis

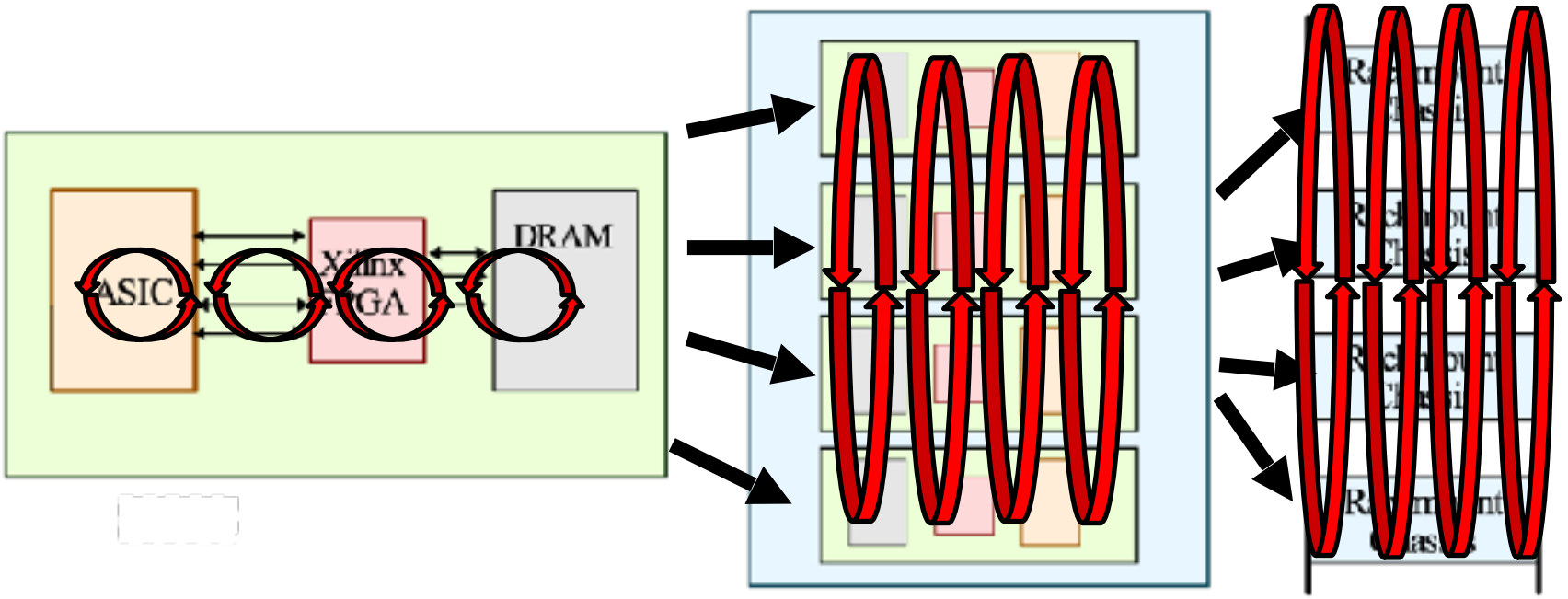


(b) HammerBlade Chassis

HammerBlade System



Memory Transmutation Layer

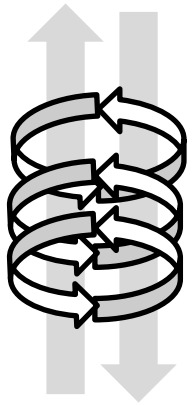


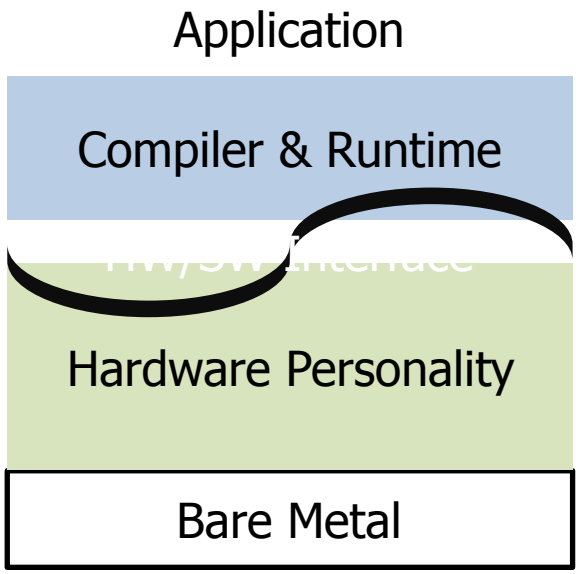
Dynamically Optimizing Data Movement Across the Full Machine Hierarchy

The HammerBlade

A Supercomputer Appliance for ML & Graphs (TA1)
with a Dynamically Evolving Software Stack (TA2)

Continuous Synthesis: learning-based empirical co-design, from design to execution.





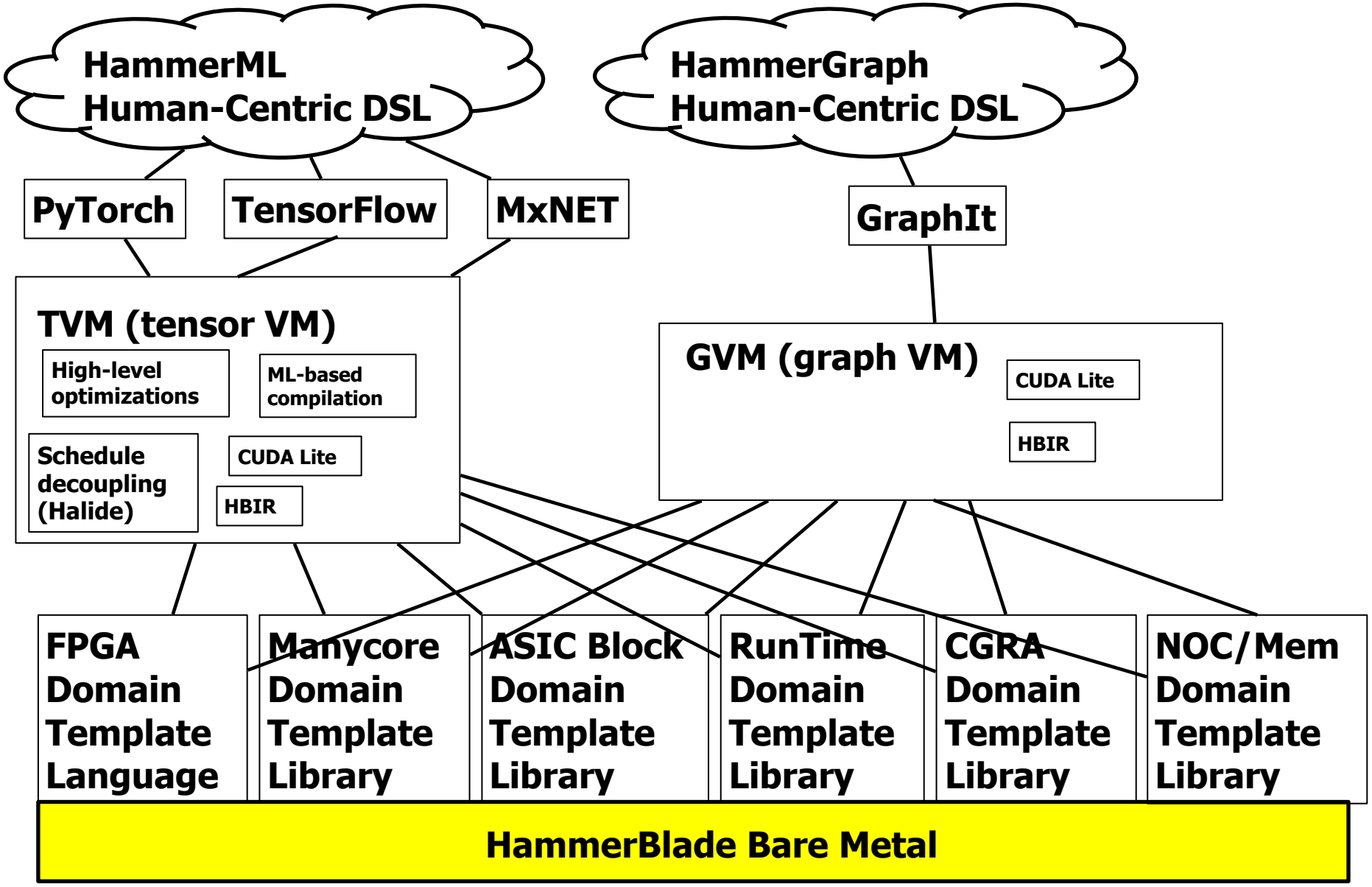
Q1 Reporting/Updates: TA-2

tvm

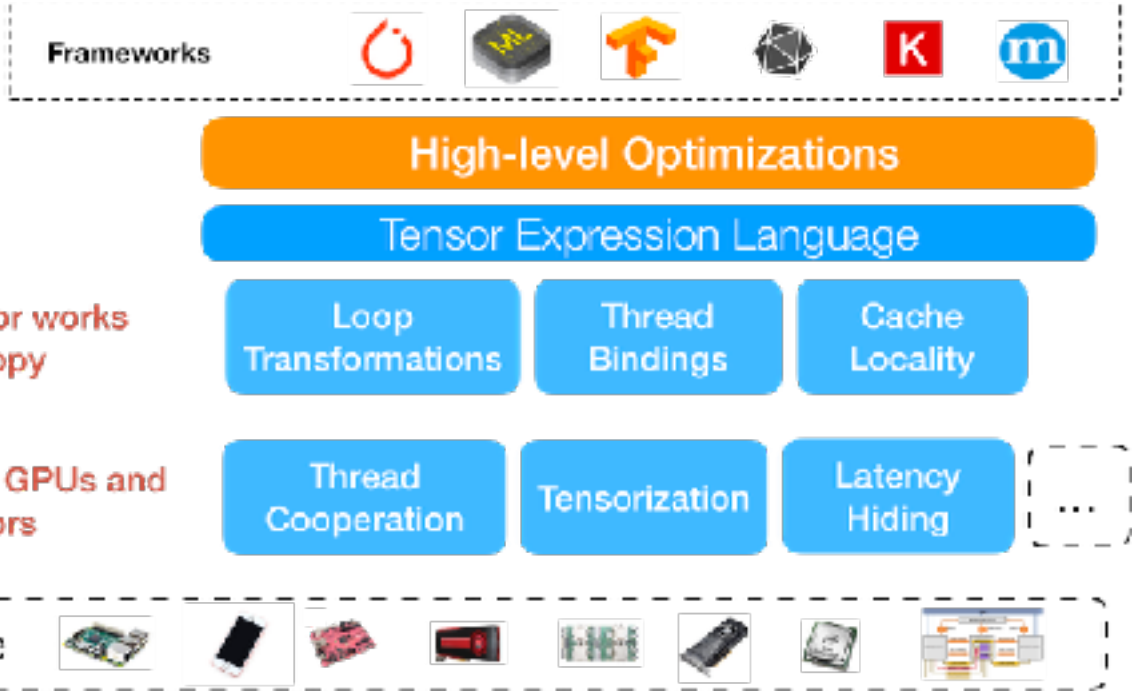
GraphIt

HBIR

Technical Approach: Software Abstraction Layers



TVM: Extensible, End-to-end Compilation for Deep Learning

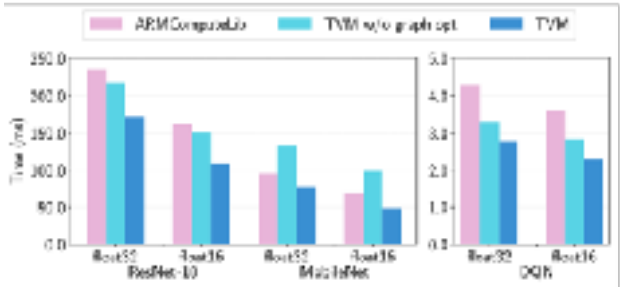


tvm.ai

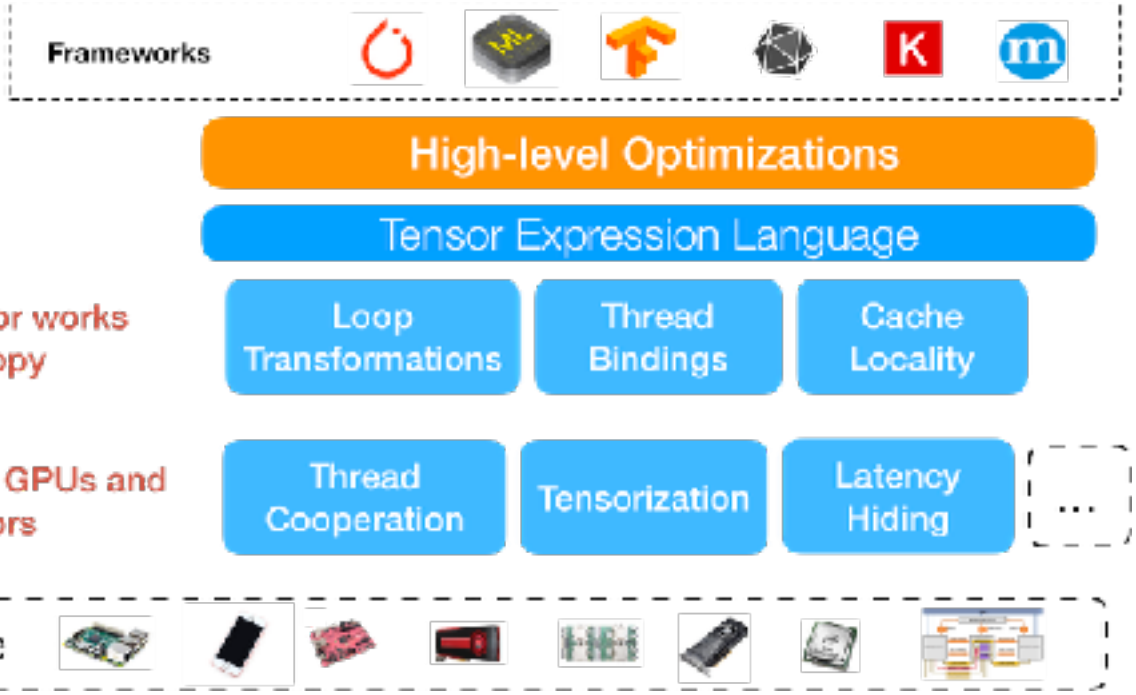
160+ contributors, several industrial users.

Try it out!

End to End Performance(ARM GPU)



TVM: Extensible, End-to-end Compilation for Deep Learning



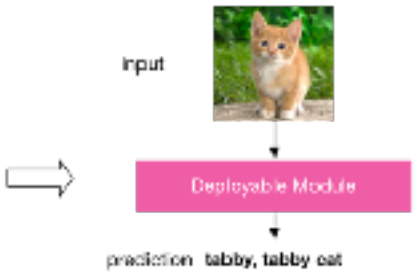
tvm.ai

160+ contributors, several industrial users.

Try it out!

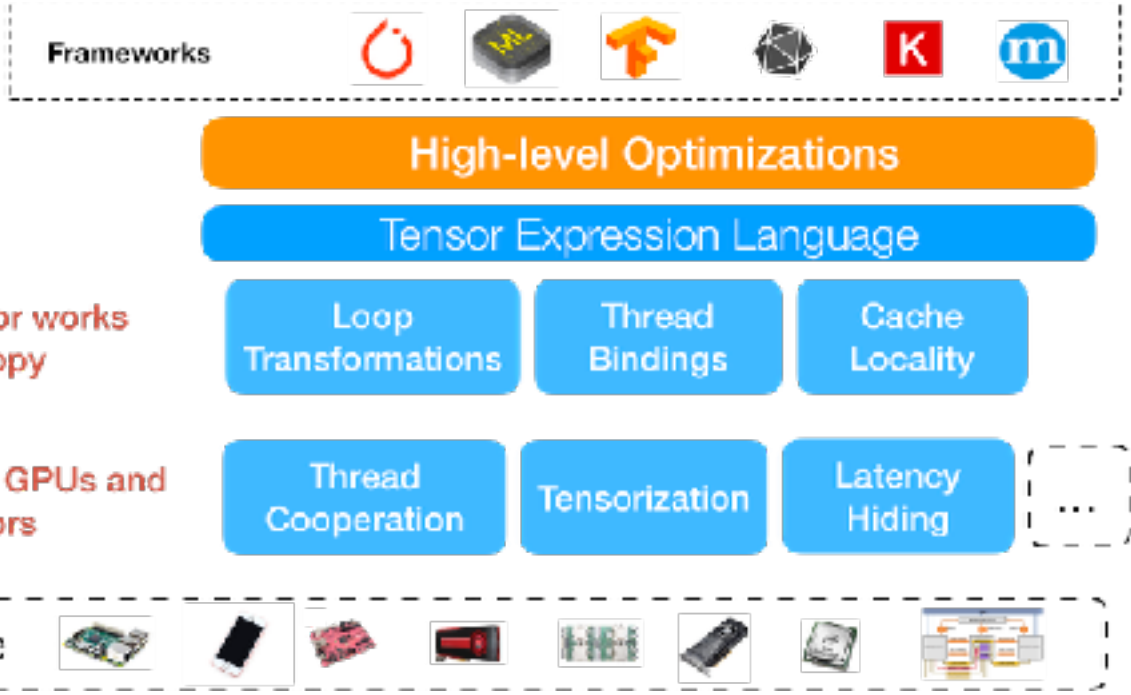
```
import tvm
import nnvm.frontend
import nnvm.compiler

graph, params =
nnvm.frontend.from_keras(keras_resnet50)
graph, lib, params =
     nnvm.compiler.build(graph, target)
```

```
module = runtime.create(graph, lib, tvm.gpu(0))
module.set_input(**params)
module.run(data=data_array)
output = tvm.nd.empty(out_shape, ctx=tvm.gpu(0))
module.get_output(0, output)
```



End to End Performance(ARM GPU)

TVM: Extensible, End-to-end Compilation for Deep Learning



tvm.ai

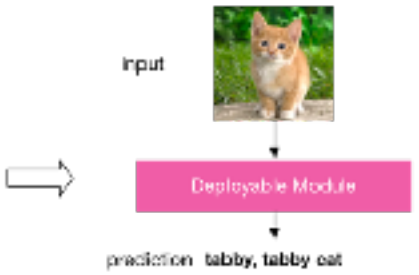
160+ contributors, several industrial users.

Try it out!

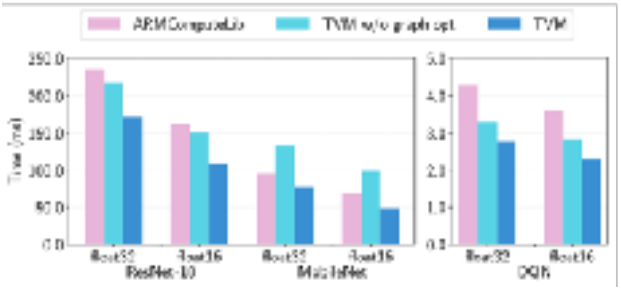
Significant engineering cost to optimize this mapping. Billions of possibilities.

```
import tvm
import nnvm.frontend
import nnvm.compiler

graph, params =
nnvm.frontend.from_keras(keras_resnet50)
graph, lib, params =
     nnvm.compiler.build(graph, target)
```

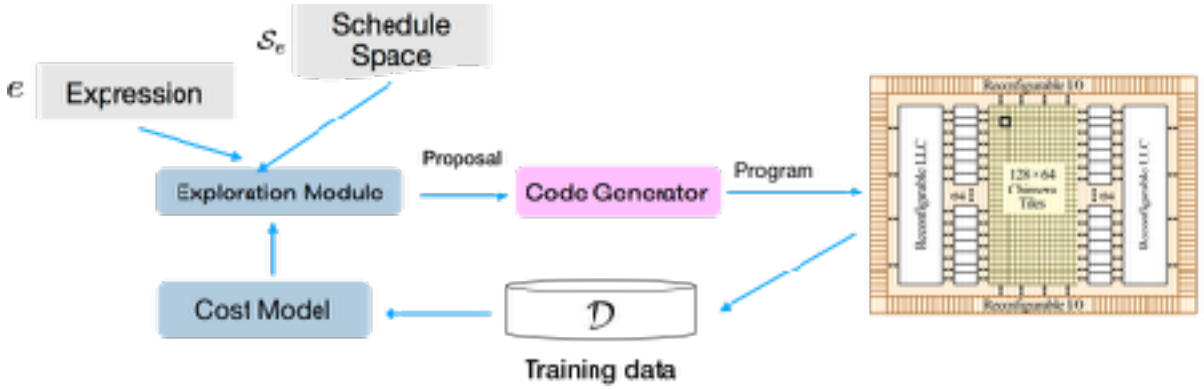
```
module = runtime.create(graph, lib, tvm.gpu(0))
module.set_input(**params)
module.run(data=data_array)
output = tvm.nd.empty(out_shape, ctx=tvm.gpu(0))
module.get_output(0, output)
```



End to End Performance(ARM GPU)

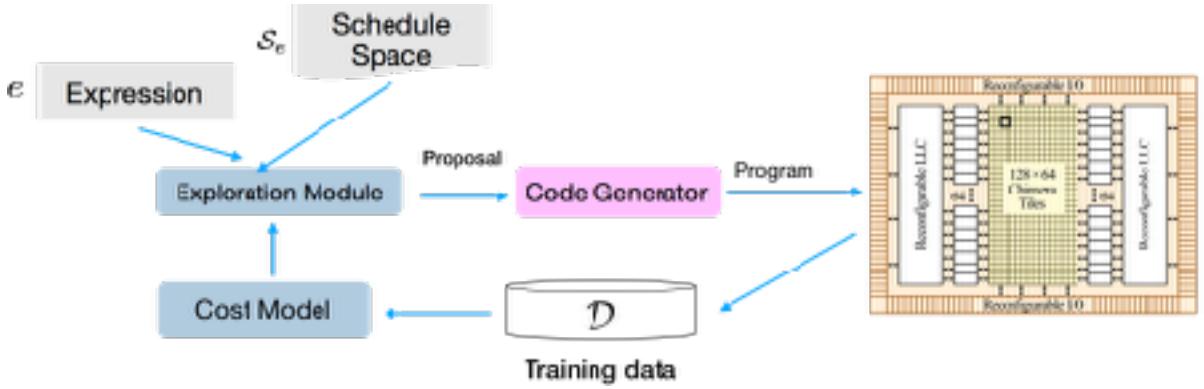


AutoTVM: Automating Code Optimizations using ML

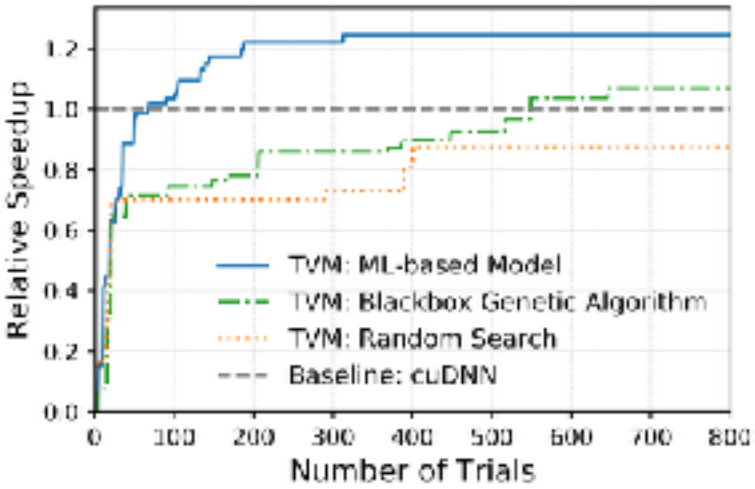


- Works very well for low-level ML code optimization [NIPS'18 spotlight]
  - e.g., beats hand-tuned-by-nVIDIA TitanX CUDA code
- Now applying to HW design exploration
  - Produce HW design variants, evaluate with compiler-in-the-loop
  - Learn HW design parameters->performance (timing, power)
  - Next: from code->HW variant->performance

AutoTVM: Automating Code Optimizations using ML



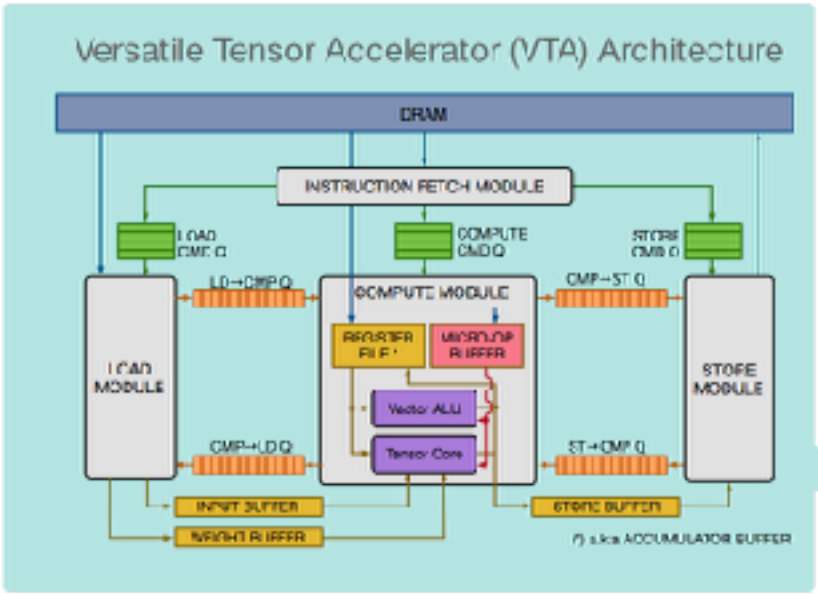
- Works very well for low-level ML code optimization [NIPS'18 spotlight]
  - e.g., beats hand-tuned-by-nVIDIA TitanX CUDA code
- Now applying to HW design exploration
  - Produce HW design variants, evaluate with compiler-in-the-loop
  - Learn HW design parameters->performance (timing, power)
  - Next: from code->HW variant->performance



AutoTVM Conv2d example on TitanX

Decoupled Access-Execute Deep Learning Accelerator Templates

Caffe2 CNTK MXNet CoreML Keras

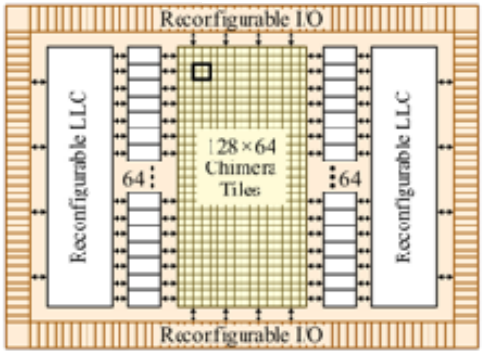


NNVM Graph Optimizer

Auto-TVM to Generate High-Performance Libraries

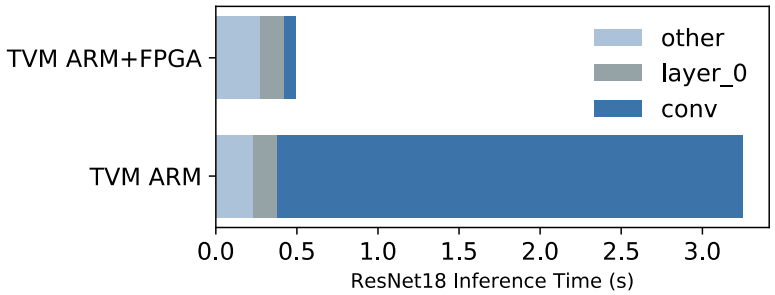
Customizable Deep Learning Accelerator Architecture

Bespoke

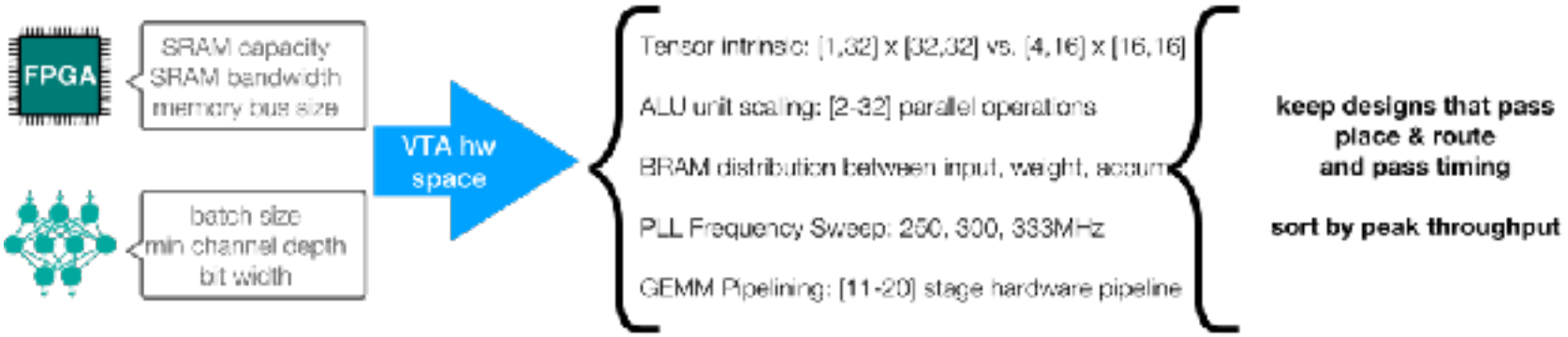


Commodity

Current open implementation @ tvm.ai/vta

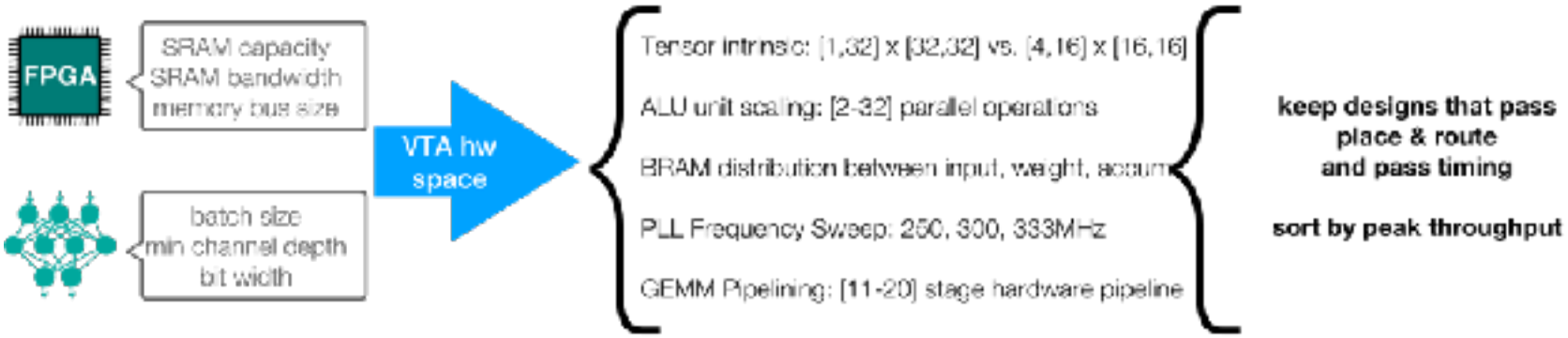


AutoVTA: Automatic Exploration of HW-SW Co-design w/ Compiler-in-the-loop (P1-TA2.4)



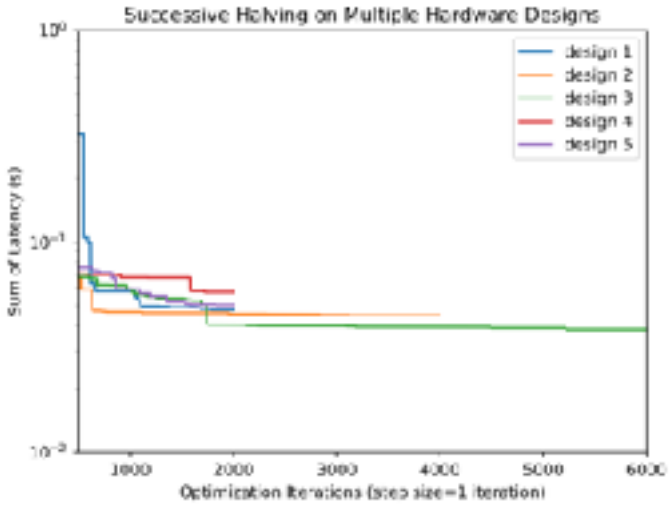
VTA variants: 1000s 10s

AutoVTA: Automatic Exploration of HW-SW Co-design w/ Compiler-in-the-loop (P1-TA2.4)



VTA variants: 1000s 10s

Apply AutoTVM



Selected designs with best End-to-End performance.

"CUDA Lite" – A Near Term IR for HB Manycore

Short term benefit to having an existing IR for architects to program the manycore.

- CUDA can express independent computation and locality and it is widely used.
- Inability to support CUDA constructs efficiently can identify issues in HB design
- TVM already lowers to CUDA
- Easy to port pre-existing CUDA code over for architectural testing.
  - High Levels of Interest from Industry for RISC-V Manycore programmable w/ CUDA

CUDA

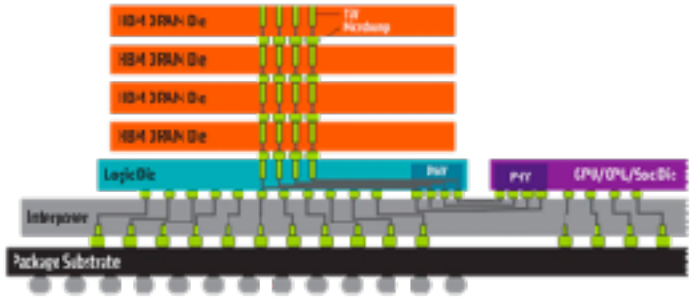
```
__global__ void add (int* a, int* b, int* c) {
    int tid = threadIdx.x ;
    if (tid < N)  // out-of-bound checks
        c[tid] = a[tid] + b[tid];
}
```

Manycore Translation

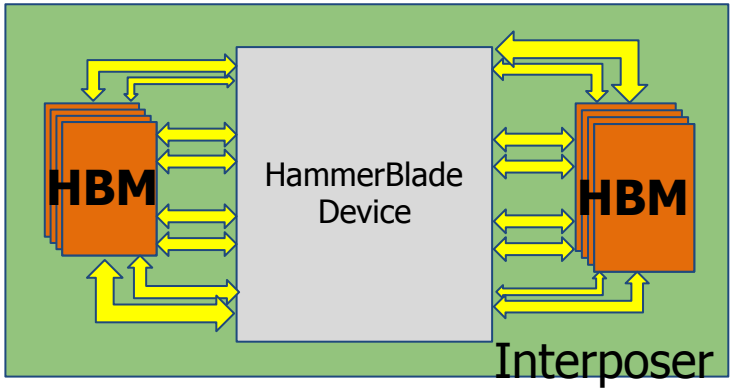
```
hb_tile void add (int* a, int* b, int* c) {
    // thread loop
    #pragma unroll
    for ( int x=hb_gangIndex; x < blockDim.x; x+= hb_gangSize){
    c[x] = a[x] + b[x];
    }
}
```

High Bandwidth Memory (HBM2) Analysis

HBM2 Device (Stack)



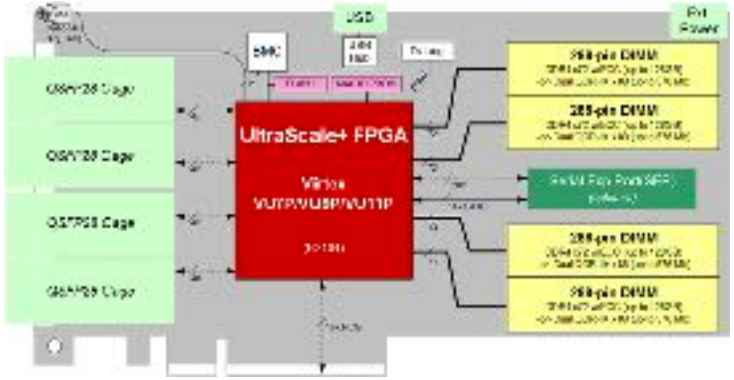
- Stacked DRAM dies connected by TSVs
  - 4 dies per HBM device
  - 2 devices per HammerBlade package
- Interposer connects SoC and HBM Dies
  - Shorter wires, higher density vs DDR4 PCB (Low Power)
- Each die has 4 "semi-independent" channels
  - 16x bandwidth vs DDR4 (4 dies * 4 channels)
- Interface is wide/slow (Low Power)
  - DDR4 is narrow/fast (High Power)



More bandwidth, more parallelism and **lower energy per bit**

Emulation on Amazon F1 FPGA Cloud

- HammerBlade Emulation on Amazon F1 FPGA Cloud
  - Architecture
    - Synthesize HB RTL to FPGA
    - HBM2 is emulated with DRAM on board
    - High Bandwidth Trace goes out of board to Intel Xeon Disk (~16 GB/sec)
    - Lower bandwidth I/O, controlled by C on Xeon
      - Write code and initialization data
      - Limited interactive debug
  - Software users can run our hardware at OS-capable speeds
    - No need for them to buy FPGA boards and tools, or license HW tools
  - Challenges
    - $1.65/hr is cheap unless you forget it's running for a week!
    - Compile times mean that simulation is a faster iteration methodology.
    - Main benefits are big workloads, scalability, and usability for non CAD-tool savvy users



Thank You

Prof. Michael B. Taylor (PI)
University of Washington

Prof. Adrian Sampson
Cornell University

Prof. Luis Ceze
University of Washington

Prof. Chris Batten
Cornell University

Prof. Mark Oskin
University of Washington

Prof. Zhiru Zhang
Cornell University

Dec 2018

Dr. Dustin Richmond (Postdoc)
University of Washington