Software Defined Hardware

For data intensive computation

Wade Shen

DARPA I2O

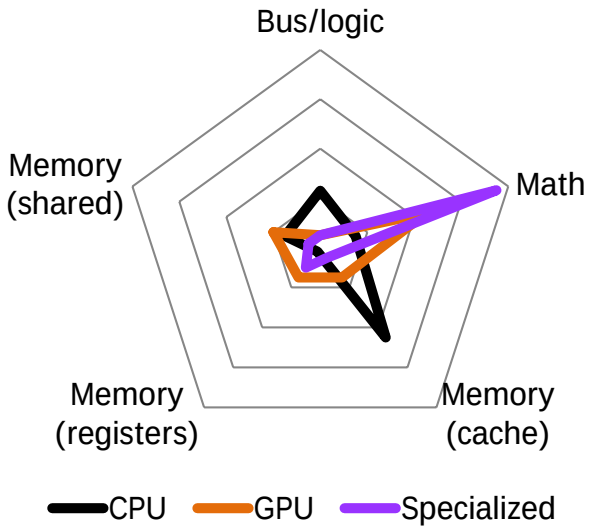
September 19, 2017

Goal Statement

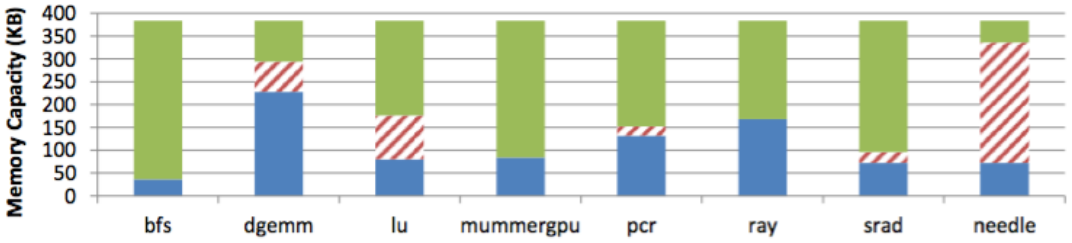
Build runtime reconfigurable hardware and software that enables near ASIC performance (within 10x) without sacrificing programmability for data-intensive algorithms.

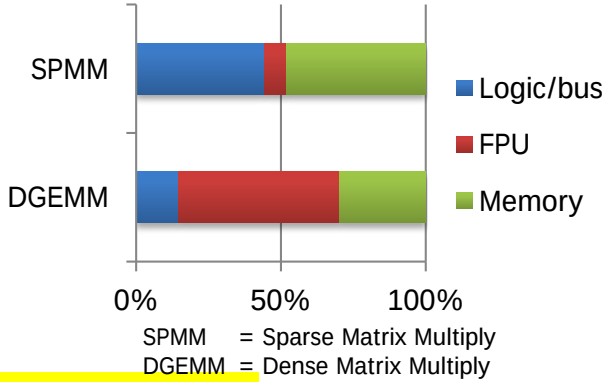
Problem

- Processor design trades
  - Math/logic resources
  - Memory (cache vs. register vs. shared)
  - Address computation
  - Data access and flow



The problem: Optimal hardware configuration differs across algorithms

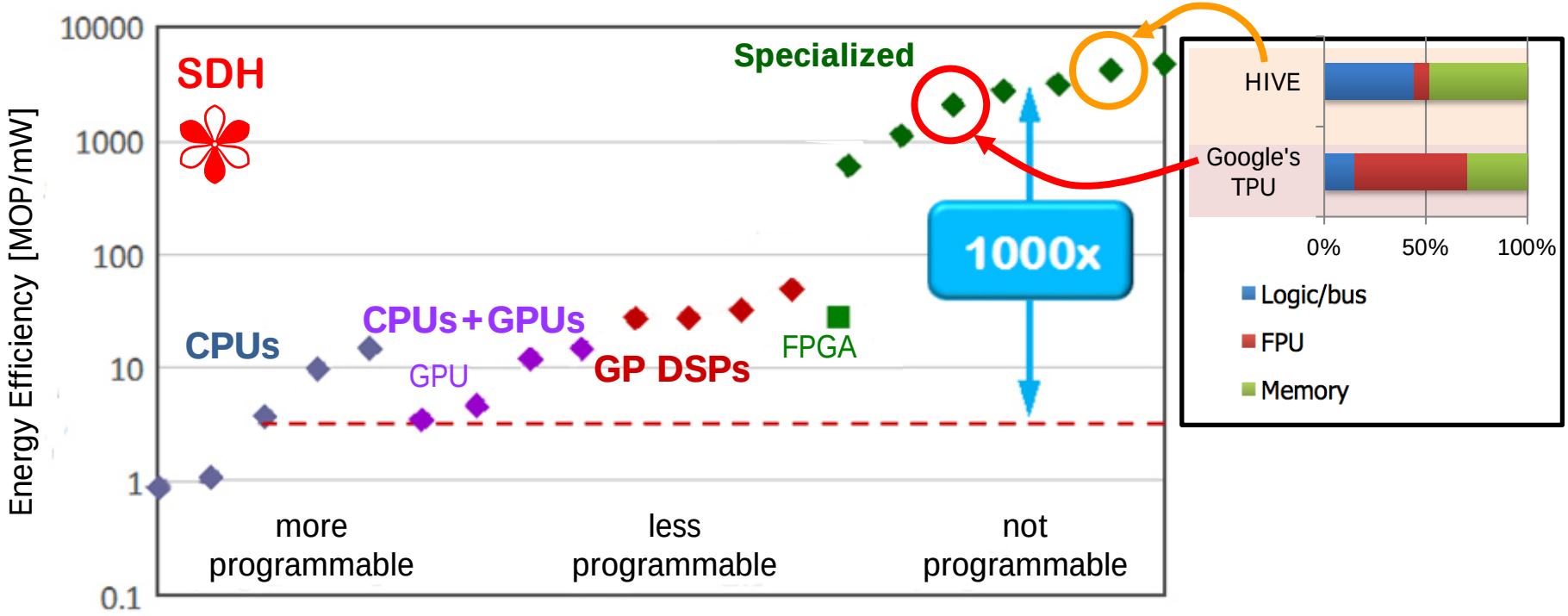




No one hardware efficiently solves all problems well

SDH: Runtime optimization of software and hardware

For data intensive computation



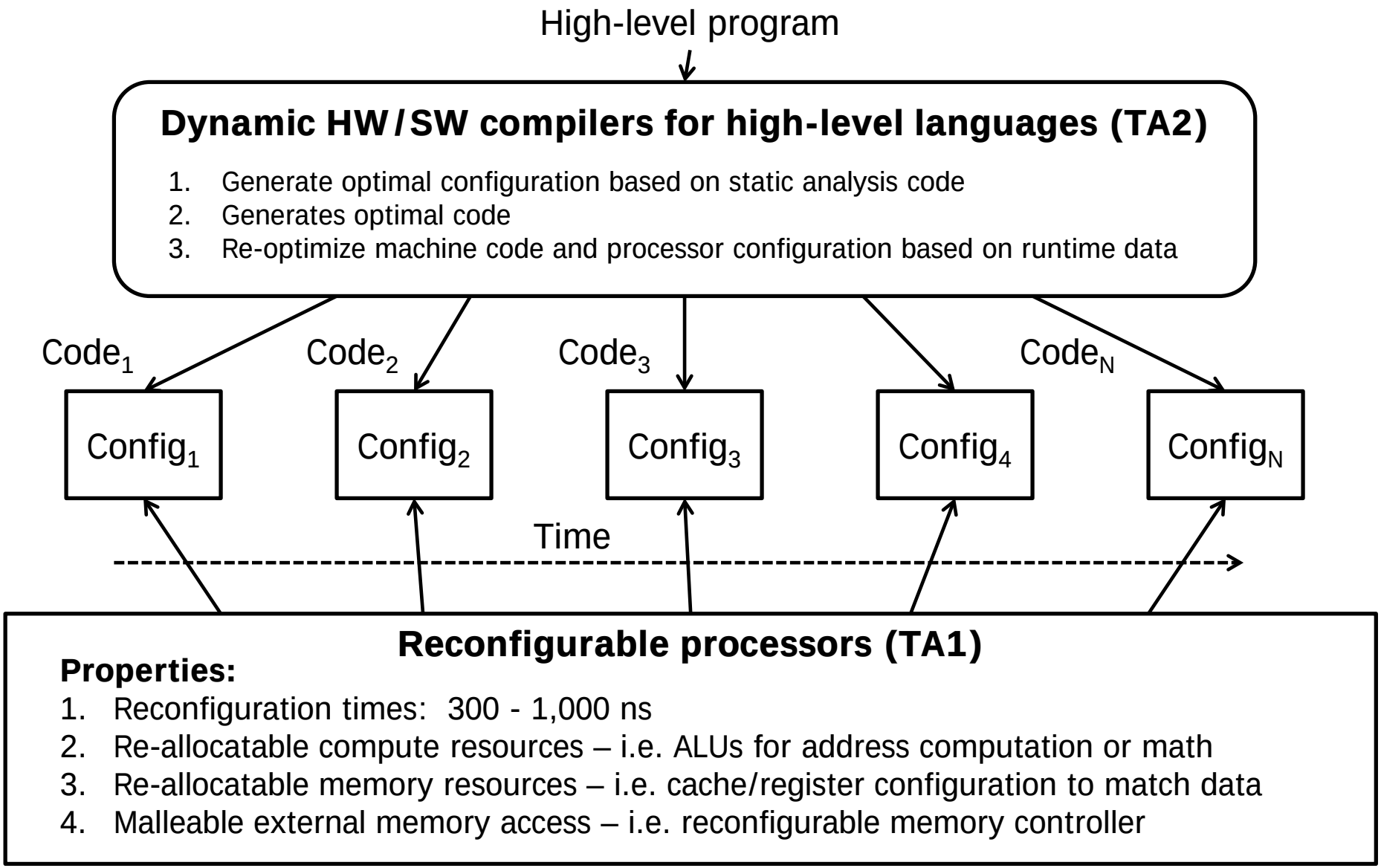
Today: HW design specialization

- One chip per algorithm
  - Chip design expensive
- Not reprogrammable
- Can't take advantage of data-dependent optimizations

Tomorrow: Runtime optimization of hardware and software

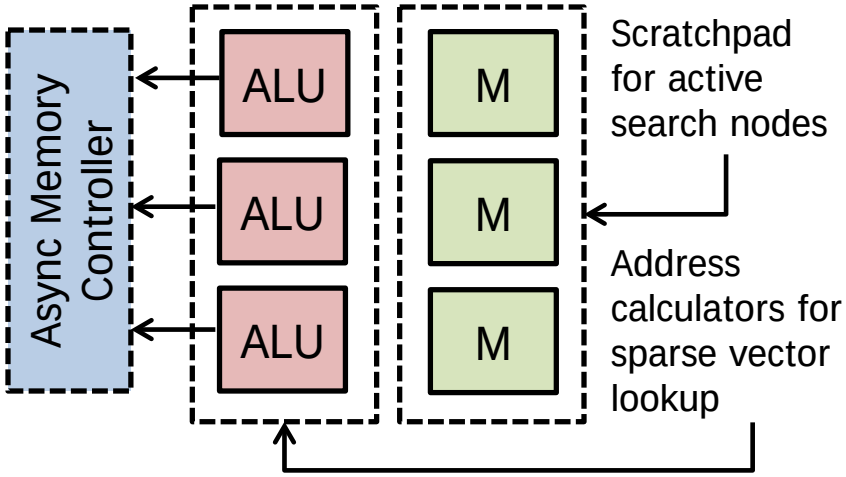
- One chip many applications
  - One time design cost
- Reprogrammable via high-level languages
- Data-dependent optimization (10-100x)

Software-defined hardware



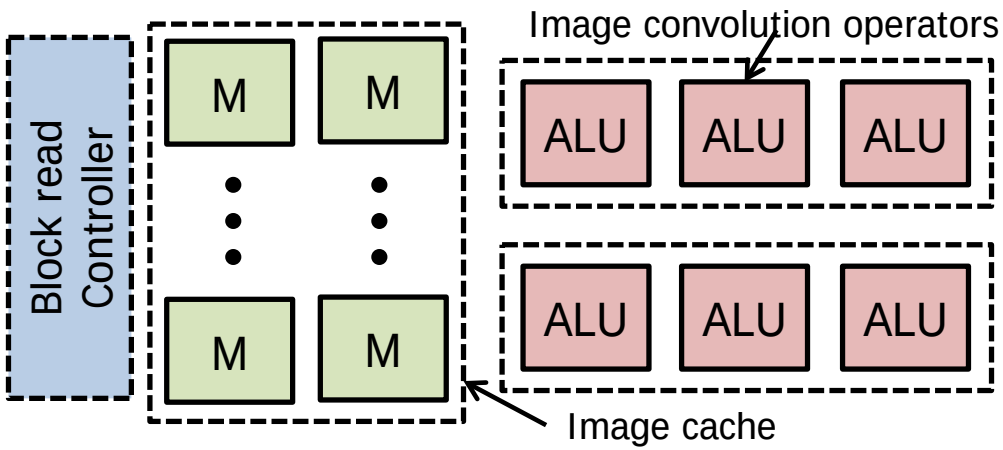
TA1: Reconfigurable processors

Graphicionado: graph search engine



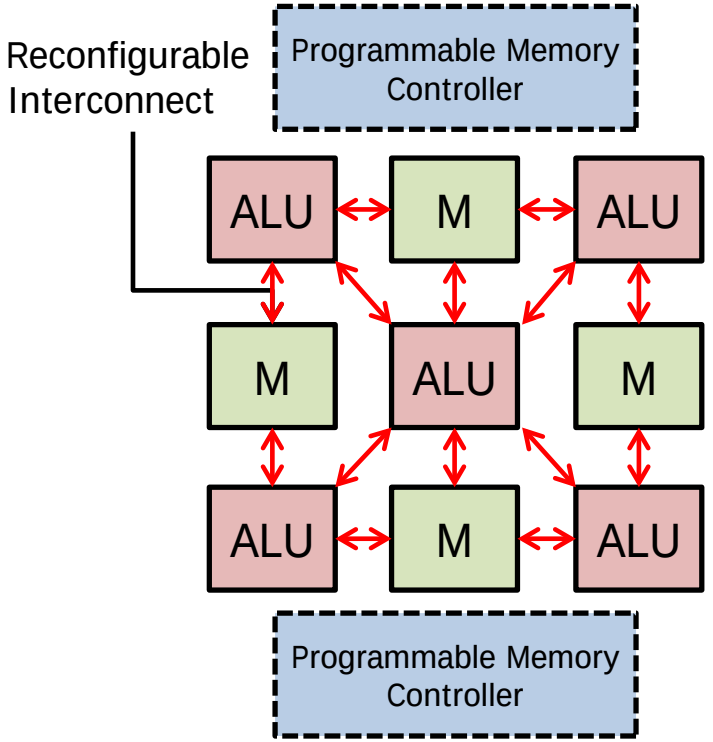
Performance: 157M edges/s/W search (BFS)

Eyeriss: Image neural net engine



Performance: 250 images/s/W (AlexNet)

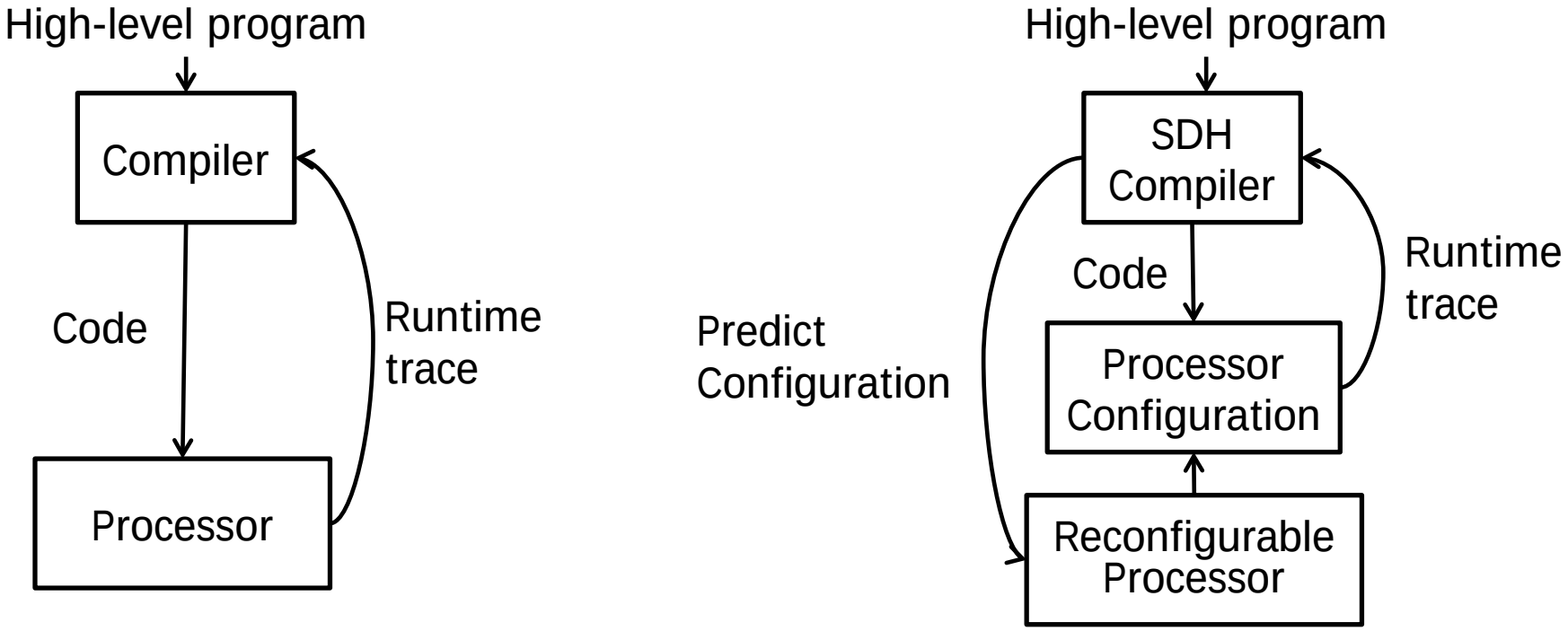
SDH



Plasticine: Stanford Seedling

- Graph search: 102M edges/s/W
- Image recognition: 130 images/s/W

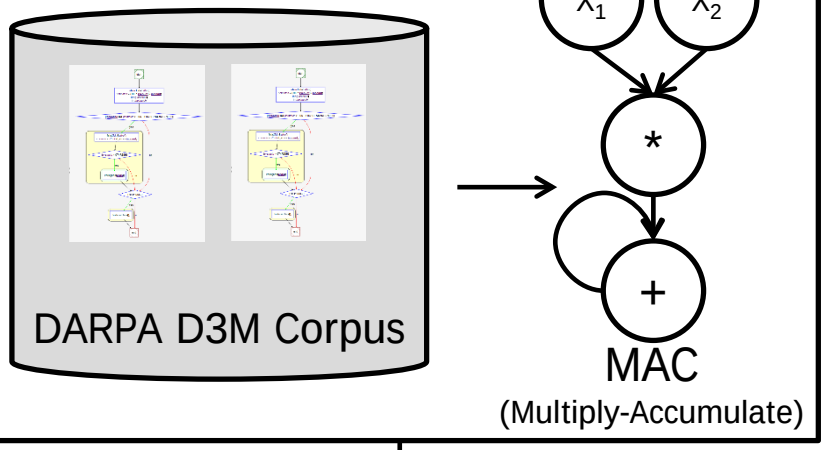
TA2: Compilers to build hardware and software



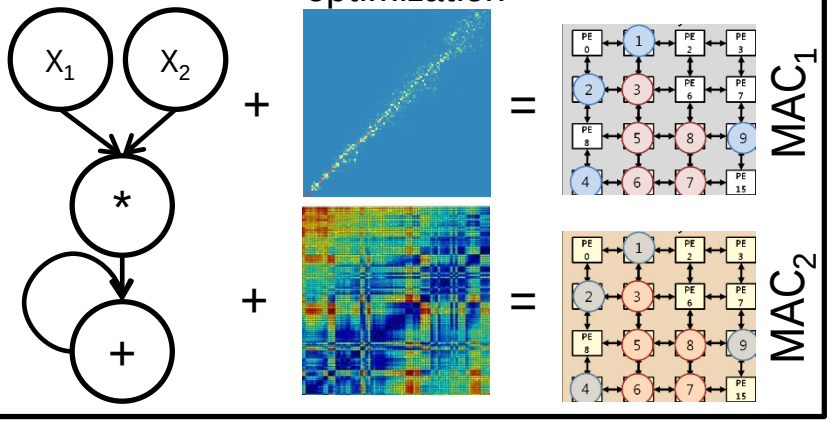
- Compilers generate optimal code via static analysis + tracing methods
  - Assume static processor configuration, compile code, run, trace, recompile
- SDH compilers don't assume a static processor configuration
  - Generates optimal configuration/code given program + data
  - Problem: Resources and architecture optimization space is large
- Solution:
  1. Configure initial processor configuration, compile code, run and trace, then
  2. Predict best configuration via reinforcement learning/stochastic optimization

How will TA2 work?

Empirical kernel mining from D3M ML corpus



IR (intermediate representation) component optimization



High-level data programs

$f(\mathbf{x}) = \text{warp}(\text{fft}(\text{window}(\mathbf{x})))$
$f(\mathbf{A}, \mathbf{x}) = \mathbf{Ax} + b$

Low-level implementations not exposed

Pre-compilation to IR (data-independent)

Ax
fft(**y**)
→ **… MAC_1 …**
… MAC_2 …

Compile time

Run time

Optimization (data-dependent)

Ax
fft(**y**)
→ **… MAC_2 …**
… MAC_1 …

Program evaluation and goals

- USG team will create a benchmark suite of machine learning, optimization, graph and numeric applications
  - 500+ programs from D3M program
  - Implementations for GPU and CPU
  - Subset of 100 optimized for ASIC (FPGA proxy)

- Metrics:
  - Speedup/power relative to ASIC and general purpose processors
  - Programmability: time to code solution for SDH languages vs. NumPy/Python

- Target outcomes:

	vs. CPU	vs. ASIC	vs. ASIC (sparse math, graphs)	Programmability
Phase 1	100-300x	within 10x	2x	within 3x
Phase 2	500-1000x	within 5x	8-10x	~1x

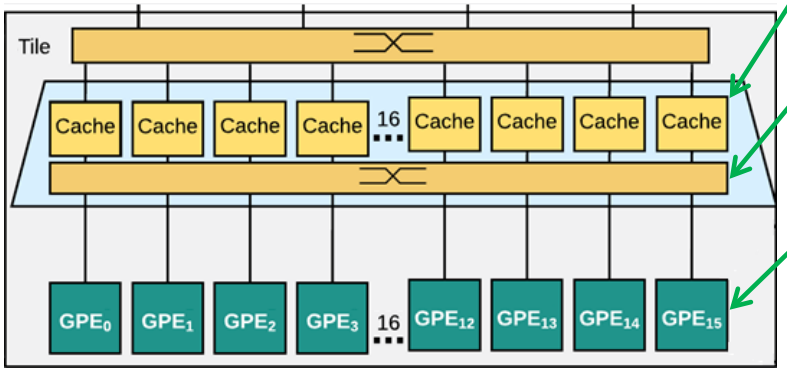
www.darpa.mil

Backup

TA1: Example architecture (U. Michigan Seedling)

- Dramatic performance and energy opportunity by tailoring architectures to applications: e.g. sparse and dense matrix multiplication and graph algorithms
- Cache hierarchy, memory bandwidth, SIMD vs. MIMD, dedicated cores

Sparse Computation

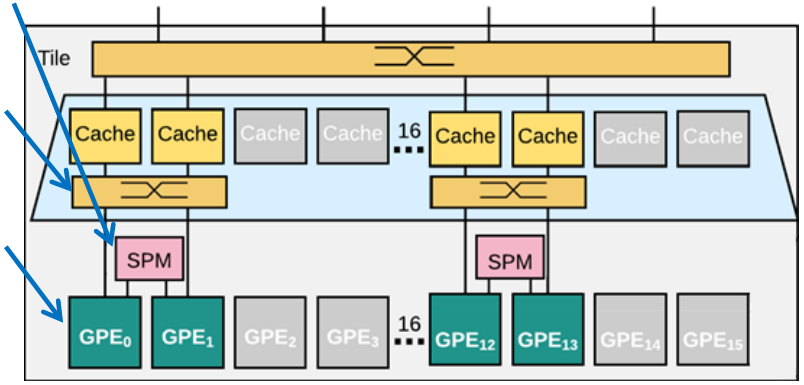


Global cache vs local scratchpad

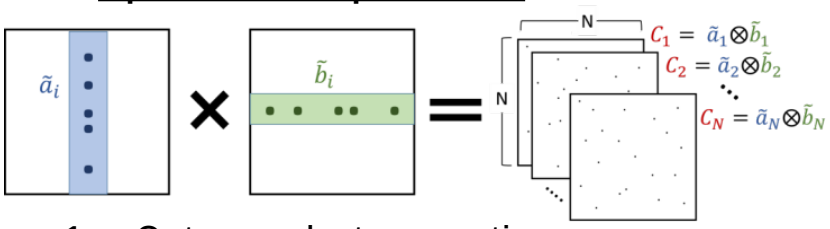
Merging crossbar vs data broadcasting

All cores for data throughput vs fewer cores for memory throughput

Dense Computation

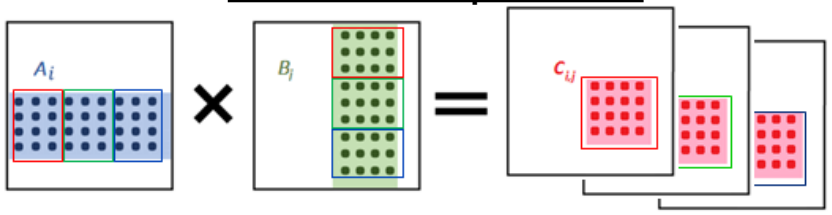


Sparse Computation



1. Outer product generation
2. Merge outer products

Dense Computation



1. Inner product on individual tiles
2. Merge tiles

- vs. CPU: 20 – 100x performance gain – data reuse, reduced memory bandwidth
- vs. GPU: 10 – 50x performance gain – data movement & placement, async execution
- vs. ASIC: within 2 – 3x of performance but full flexibility / programmability

TA1: Fast reconfigurable processors

ASICs

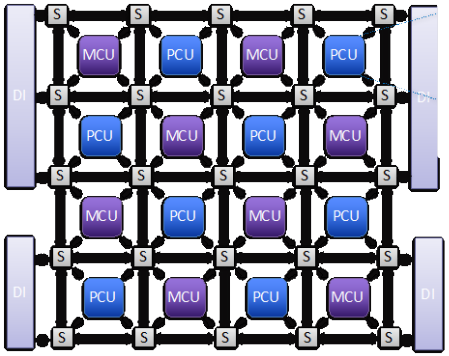
Graphicionado: ASIC for graph search

- Element-based memory access
- Specialized indirect address calculator for sparse vectors
- Specialized on-chip scratch pad
- **157K edges/s/mW search (BFS)**

Eyeriss: Specialized DNN accelerator

- 168 specialized convolution units
- Specialized implementation of neural non-linearity (ReLU)
- Large block memory access only
- **250 images/s/W on AlexNet**

SDH



SPARSE:
BFS on Twitter
102K edges/s/ mW

DENSE:
AlexNet (full app.)
130 images/s/W

SDH Opportunities (vs. CPU)			
Attribute		Dense Advantage	Sparse Advantage
64 memory control units 64 data pattern control units		30x Perf, 52x Perf/W	8.2x Perf, 55.2x Perf/W
Configurable off-chip access for dense and sparse (scatter/gather)		1x	18x
Configurable on-chip memory for high BW & coarse-grain pipes		2.1x	18.4x
Compute	Pipelined SIMD	12.6x	NA
	Variable precision	25.2x	
	Shift network	7x	

Key Challenges:

- Data flow: configurable memory control units, data patterns, data storage
- Compute flexibility: compute granularity, modular functionality, high BW malleable interconnect