Enhancing Programmable Accelerators for Sparsity

Vidushi Dadu*, Jian Weng*,
Sihao Liu*, Tony Nowatzki*

*UCLA

Yarch, HPCA 2019, Feb 17
Irregular workloads are ubiquitous

Kernel-SVM on High-dim Data
Irregular workloads are ubiquitous

Kernel-SVM on High-dim Data

Decision Tree training
Irregular workloads are ubiquitous

Kernel-SVM on High-dim Data

Decision Tree training

Pruned Deep Neural Networks
Irregular workloads are ubiquitous

Kernel-SVM on High-dim Data

Decision Tree training

Pruned Deep Neural Networks

Genomics
Irregular workloads are ubiquitous

Kernel-SVM on High-dim Data

Decision Tree training

Pruned Deep Neural Networks

Genomics

Graph Processing
Sparsity in Workloads Requires Complex Architecture Mechanisms

**Workloads/Kernels**
- Sparse Factorization
- GBDT Training
- Dynamic Sparsification
- Graph Traversal
- Shortest Path

**Challenging Properties**
- Indirect Memory Access
- Control-dependent Memory
- Atomic Updates
- Dynamic Parallelism
- Load Balancing
- Conditional Computation
- Heterogeneous Datatypes
Sparsity in Workloads Requires Complex Architecture Mechanisms

**Workloads/Kernels**
- Sparse Factorization
- GBDT Training
- Dynamic Sparsification
- Graph Traversal
- Shortest Path

**Challenging Properties**
- Indirect Memory Access
- Control-dependent Memory
- Atomic Updates
- Dynamic Parallelism
- Load Balancing
- Conditional Computation
- Heterogeneous Datatypes

**Memory Control Data type**
Sparsity in Workloads Requires Complex Architecture Mechanisms

**Workloads/Kernels**
- Sparse Factorization
- GBDT Training
- Dynamic Sparsification
- Graph Traversal
- Shortest Path

**Challenging Properties**
- Indirect Memory Access
- Control-dependent Memory
- Atomic Updates
- Dynamic Parallelism
- Load Balancing
- Conditional Computation
- Heterogeneous Datatypes

**Memory Control Data type**
Sparsity in Workloads Requires Complex Architecture Mechanisms

**Workloads/Kernels**
- Sparse Factorization
- GBDT Training
- Dynamic Sparsification
- Graph Traversal
- Shortest Path

**Challenging Properties**
- Indirect Memory Access
- Control-dependent Memory
- Atomic Updates
- Dynamic Parallelism
- Load Balancing
- Conditional Computation
- Heterogeneous Datatypes
Sparsity in Workloads Requires Complex Architecture Mechanisms

**Workloads/Kernels**
- Sparse Factorization
- GBDT Training
- Dynamic Sparsification
- Graph Traversal
- Shortest Path

**Challenging Properties**
- Indirect Memory Access
- Control-dependent Memory
- Atomic Updates
- Dynamic Parallelism
- Load Balancing
- Conditional Computation
- Heterogeneous Datatypes
GPUs are insufficient for irregularity

Memory Irregularity (Wide mem)
Accesses to random memory location causes inefficient cache line utilization.
GPUs are insufficient for irregularity

**Memory Irregularity (Wide mem)**
Accesses to random memory location causes inefficient cache line utilization.

**Control Irregularity (SIMD)**
Data-dependent compute leads to masking in vector architectures.
GPUs are insufficient for irregularity

Memory Irregularity (Wide mem)
Accesses to random memory location causes inefficient cache line utilization.

Control Irregularity (SIMD)
Data-dependent compute leads to masking in vector architectures.

Datatype Irregularity (Fixed vector width)
Irregular algorithms doesn’t allow efficient packing of lower datatypes.
Is Massive Scalar Processor Sufficient?
Is Massive Scalar Processor Sufficient?

There are a couple of problems:

• **General purpose overheads:** Maintaining the program counter and precise state limits limits performance.
Is Massive Scalar Processor Sufficient?

There are a couple of problems:

• **General purpose overheads:** Maintaining the program counter and precise state limits performance.

• **Programmability:** Such architecture hurts the performance of regular algorithms which have high acceleration potential.
Hope: Sparse Accelerators Have Been Successful

1. **SCNN**: for sparse conv. layer
2. **EIE**: for sparse FC layers
3. **Graphicionado**: graph processor
4. **HATS**: Locality-aware scheduling for graph processing
5. **XMem**: Programmer hints to prefetchers, caching policies
Hope: Sparse Accelerators Have Been Successful

1. **SCNN**: for sparse conv. layer
2. **EIE**: for sparse FC layers
3. **Graphicionado**: graph processor
4. **HATS**: Locality-aware scheduling for graph processing
5. **XMem**: Programmer hints to prefetchers, caching policies

It will be useful if we have an architecture which gives high performance on the set of workloads we care about most.
Goal: Is there an accelerator paradigm which performs well on most irregular workloads?
Goal: Is there an accelerator paradigm which performs well on most irregular workloads?

In this talk, I will focus on **Irregular Machine Leaning**.

In machine learning domain, both dense and sparse scenarios are pretty common.

**Sub-goal**: Design a programmable sparse accelerator while maintaining efficiency for the dense workloads.
Our Approach: Start with a Dense Programmable Accelerator
Our Approach: Start with a Dense Programmable Accelerator
Our Approach: Start with a Dense Programmable Accelerator

Google TPU v2

PuDianNao (ASPLOS’15)

Tabla (HPCA’16)

Wide Scratchpad

Router

Systolic Array

Stereotypical Dense Accelerator Core

Systolic Array

Wide Scratchpad

Control

Stereotypical Dense Accelerator Core
Overview of Sparsity-Enabled SPU core

(a) Stereotypical Dense Accelerator Core

(b) Sparsity Enabled Accelerator (SPU Core)
Overview of Sparsity-Enabled SPU core

(a) Stereotypical Dense Accelerator Core

(b) Sparsity Enabled Accelerator (SPU Core)

Systolic array with novel meta-reuse control flow
Overview of Sparsity-Enabled SPU core

(a) Stereotypical Dense Accelerator Core

(b) Sparsity Enabled Accelerator (SPU Core)

Compute-Enabled High-bandwidth Indirect Scratchpad

Systolic array with novel meta-reuse control flow
Overview of Sparsity-Enabled SPU core

(a) Stereotypical Dense Accelerator Core

(b) Sparsity Enabled Accelerator (SPU Core)

Compute-Enabled High-bandwidth Indirect Scratchpad
Decomposable Memory/Network/Compute
Systolic array with novel meta-reuse control flow
Evaluation Methodology

• gem5 cycle level simulator
• Dataflow compiler
• Benchmarks (top-5 ML algorithms using by Facebook in 2018*):
  • KSVM, Conv layer, FC layer, GBDT, Arithmetic Circuits
• Datasets: Open-source practical datasets
• Baselines:
  • CPU: 4-core CPU (reference only)
  • GPU: Nvidia P4000
  • Massive In-order: SPU but with 512 inorder cores
  • SPU: Sparse Proc. Unit (64 Core)

SPU achieves 2.2-6.8x performance over GPU
Ongoing/Future Work

• Compiler for streaming-dataflow architecture.
• Specialized cache hierarchy
• Hardware-software (ISA) primitives for other irregular domains:
  • Graph processing
  • Genomics
  • Compression/Decompression