■ ASPLOS **五日目**

本会議三日目.

■ Machine Learning I

PUMA: A Programmable Ultra-efficient Memristor-based Accelerator for Machine Learning Inference

- memristive crossbar
  - 2-6 bits per cell vs 1-bit or CMOS(SRAM) = 6x
  - cell area is 4F^2 vs 120F^2 for CMS(SRAM) = 30x
  - Analog MVM 1.34pJ/op
- ref. RENO DAC15, PRIME ISCA16
- Domain-specific ISA
  - large register address space to support memoristive crossbar
  - vector width keeps instruction memory low in spatial architecture
- Hybrid core
  - hybrid memrisitive and CMOS
- compiler optimization
  - graph partitioning
  - MVM instruction consume high latency
- inference energy: skylake, Pascal と比べて削減.
- PUMA compiler https://github.com/illinois-impact/puma-compiler
- PUMA simulator https://github.com/Aayush-Ankit/dpe_emulate

FPSA: A Full System Stack Solution for Reconfigurable ReRAM-based NN Accelerator Architecture

- isues
  - ReRAM なシステムでは DAC と ADC がでかい．(Logical View だと ReRAM でかいけど)
  - communication bound
  - reliability
  - flexibity
    - ReRAM-based VMM(fast), Digital-based others(relatively slow)
- refs. bridge tha gap between neural netwoks and ..., ASPLOS'18
- FPSA; ReRAM-based processing element
  - reduce digital circuit, spiking schema
  - fully parallel
- routing = iland-style, like FPGA (ref. mrFPGA)
- system stack: neural synthesizer -> spatial-to-temporal mapper -> place & route

Bit-Tactical: A Software/Hardware Approach to Exploiting Value and Bit Sparsity in Neural Networks

- MAC 演算ユニットにスパースな演算データを無駄を省いて供給したい
- → オンデマンドであいてる演算器にデータをつっこめるようにする
- 演算器へのデータパスに MUX をいれてデータ供給を制御している，のかな.
- うまくつくれれば便利そう

■ Machine Learning II

TANGRAM: Optimized Coarse-Grained Dataflow for Scalable NN Accelerators

- scaling NN perf.
  - use more PEs & more on-chip buffers
    - monolithic engine <- low resource utilization, long array busses, far from SRAM
    - -> tiled architecture - mostly local data transfers, easy to scale up/down
    - - <- dataflow scheduling ?
- inter-layer parallel
  - buffer sharing dataflow - タイルでデータを共有 → 最初に分割して配って，あとで交換する
- inter-layer pipeline
  - pipeline multiple layers, pros: save DRAM B/W, cons: utilize resources less efficiently(long delay, large SRAM)
  - -> fine-grained data forwarding
    - forward each subset of data to the next layer as soon as ready
    - require matched access patterns between adjacent layers
    - データフローツールでパイプラインスケジューリングする

Packing Sparse Convolutional Neural Networks for Efficient Systolic Array Implementations: Column Combining Under Joint Optimization

- スパース行列をデンスな行列に変換する話
- zero weights in systolic arrays are wasteful
  - -> column combining. 9 タイルを 3 タイルに.
    - 保存された重さとの積の方だけ選択して計算する.
- ref. Full-stack Optimization for Accelerating CNNs with FPGA Validation, ICS 2019 ???

Split-CNN: Splitting Window-based Operations in Convolutional Neural Networks for Memory System Optimization

- DL faces a memory problem, HBM meomry is expensive
  - Accelerator(eg. GPU): 16GB/32GB/..., Host: 512GB/1TB/...
- opportunities enabled by NV-LINK
  - ref. vDNN(Rhu, MICRO 49)
- memory profile of training DNN
- Split-CNN
  - accuracy drops slowly as we splite deeper and into more patches
  - batch 毎に split の感じをかえる
- HMMS is a static memory planner that detemines the timing of memory allocation, deallocation, prefetching and offloading

学習時のメモリボトルネックを解決するために, データを分割する Split-CNN と, メモリ管理 / プリフェッチの管理システム HMMS を提案. IBM Power System S822LC で評価.

■ Storage

LightStore: Software-defined Network-attached Key-value Drives

組み込みクラスのプロセッサと数 TB の NAND FLASH を使った NW 接続な KVS LightStore を提案. FTL は HW 上に実装. Xeon サーバ上の RocksDB と比べて, Random Set の速度は Xen サーバを凌駕，ノード数に対してスケール，省電力.

- one ssd per network port, KV interface,
- optimization
    - system optimization
        - mmemcopy, thread
- LSM-tree spec. opt
    - decoupled keys from KV paris, bloom filter
- FTL in HW

SOML Read: Rethinking the read operation granularity of 3D NAND SSDs

3D NAND で密度あがったので同じ容量の SSD はチップ数減って，チップ間並列性がへって読み出しが遅くなった．なので, Partial-page 読み出しを 1 つの read 命令にパックできるように SW と HW を工夫した，と．

- fewer number of NAND chips -> lower multi-chip parallelism
- ← sigle-operation-multiple-location
    - Partial-page read を 1 READ 命令にまぜる

FlatFlash: Exploiting the Byte-Accessibility of SSDs within A Unified Memory-Storage Hierarchy

SSD(PCIe 接続なフラッシュストレージ) に DRAM と同じようにバイトアクセスできるようにするために
SSD->DRAM への promotion メカニズムを実装した，と．

- FlatFlash, byte addressable interface
    - avoid paging
    - reduce i/o traffic
    - reduces dram latency
- dram in ssd + pcie mmio + opencapi
- ref. FlashMap, ISCA'15 - unifying the memory and storage <- FlatFlash は 1.6 倍速い.
- DRAM への promote がおそい -> background 実行したい -> consistency 問題

■ Quantum Computing

A Case for Variability-Aware Policies for NISQ-Era Quantum Computers

- ref. qubit の swap を最適化する問題
- not all qubis are created equal
    - exploit variation in error rates to improve reliability
        - assign more operations on reliable qubits/link
        - <- SWAP カウントじゃなくて

Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices

- qubit connection limitation
- mapping with SWAP
    - heuristic - Zulehner et al., DATE'18, Siraichi et al., CGO'18
- reduce search complexity

- swap-based search
    - Prev.: mapping-based search, high complexity - O(exp(N))
    - Proposed: search a SWAP sequence - only consider high-priority qubits - O(N^2.5)
- reverse traversal for init. mapping
    - Prev.: random initial mapping
    - Proposed: Inspired by the reversibility
- control the parallelism

Noise-Adaptive Compiler Mappings for Noisy Intermediate-Scale Quantum Computers

- Q algorithm と実機にはギャップがある
- NISQ Resource constraints
    - Low qubits: 5-72
    - high gate error rates: 1-10%
    - Qubts hold state for 100us
- cur.
    - compile onece per input: more optimization opportunities
    - reduce program execution time to avoid decoherence
    - communication/SWAP optimization
    - Used in IBM, Rigetti, Google compilers
    - -> NISQ system have ~10x spatial and temporal noise variation!
- proposed: noise-adaptive compilation
- noise variation impacts successes rate
    - noise data is measured twice daly by IBM - https://quantumexperience.ng.bluemix.net/qx/devices
- #1: choose a good initial mapping
- #2: coherene-aware sheduling
    - influences mapping: choose qubits with good coherence time
- #3: reduce SWAPs, use low-error rate routes
- -> implement as a constrained optimization
- Scaffold Program -> LLVM IR ScaffCC -> Optization using z3 SMT Solver* -> OpenQASM
- * にノイズデータいれる

https://github.com/prakashmurali/TriQ

Optimized Compilation of Aggregated Instructions for Realistic Quantum Computers

ロジカルな量子操作と物理的な操作の乖離が大きい．効率的な物理制御をするために 1-, 2-qubit 操作じゃなくて，最大 10qubits まで同時に操作するようなユニットにまとめるよ．という話なのかな？

- layered approach to quantum compilation
- GRAPE - GRadient Ascent Pulse Engineering
- how to maximally utilize optimal control? - physical gate decomposition, phisical gate optimization