■ FPGAX

Google@ 六本木にて.

https://fpgax.connpass.com/event/115446/

■ FPGAX メモ

「TPU の最近の話」Google 佐藤さん

- TPU Pod - HPC-powered scalable all reduce distributed training
  - Cloud TPU - https://cloud.google.com/tpu/
- 実はいろんな HW がある．BigQuery architecture とかもある
  - https://cloud.google.com/solutions/architecture/complex-event-processing
  - https://cloud.google.com/blog/products/gcp/implementing-an-event-driven-architecture-on-serverless-the-smart-p
  - https://panoply.io/data-warehouse-guide/bigquery-architecture/
- TPU v1, v2, v3 - 2018
  - 2016 年時点では MLP が 6 割くらいだった
- brain floating point format
  - exp 8bit, mantissa 7bit - FP16 と違って，FP32 と同じだけ指数部を多めにとってる
- TPU v2, v3
  - v2: 180TFLOPS, 64GB HBM - $4.5/h($1.35/h preembitble) @us
  - v3: 420TFLOPS, 120GB HBM
- DAWNBench でコスト評価 https://dawn.cs.stanford.edu/benchmark/
  - GPU の 1/5 で学習できますよ，と.
- TPU3.0 Pod : > 100PFLOPS (8x faster than v2)
- All reduce with 2-D toroidal mesh network by Google's HPC hardware
- Coud TPU v2(64 units) vs. NVIDIA V100(8 units)
  - 27x faster training at 35% lower cost
- ebay
  - 55M training image
  - accuracy boost +10%, training time speedup 100x
- data parallel, model parallel
  - これからは model parallel
    - 参考 https://research.preferred.jp/2018/12/model-parallelism-in-dnn/
  - model parallel training with biggan
    - https://colab.research.google.com/github/tensorflow/hub/blob/master/examples/colab/biggan_generation_wi
- Cloud TPU APIs
  - estimator
  - keras
- Edge TPU

「AI チップ最新レビュー」北海道大学 百瀬啓さん

- ニューロモフィックな商用チップもある
  - https://www.mythic-ai.com/
  - Eta tensai - https://etacompute.com/news/press-releases/684/
- 写真 penne の ~/Pictures/fpgax_20190202 に
  - ニューロチップの世界の動向 - IMG_20190202_131619613.png
  - (Server) - Scaling Trend - IMG_20190202_132028592.png
  - DaDianNao: 中国 CAS 学習 / 推論用 2014 年 - IMG_20190202_132138781.png
  - 量子化・圧縮の適用 - IMG_20190202_132429613.png
  - (edge) - Quantization Trend - IMG_20190202_132843358.png

- 量子化 : CNN+RNN・(DNPU) ISSCC '17/14.2 KAIST D.Shin - IMG_20190202_133013971.png
- Log 量子化 / ビットシリアル ... QUEST '18 北大 - IMG_20190202_133250029.png
- (edge) in-memory processing - IMG_20190202_133538324.png
- BRein Memory : Binary/Ternary in-memory ('17/6 月) - IMG_20190202_133712446.png
- BWN/TWN/BNN(精華大) '18 VLSIC - IMG_20190202_133831586.png
- Mixed Signal Binary CNN (プリンストン大 '18 VLSIC) - IMG_20190202_134035448.png
- ReRAM(RAND) パナソニク VLSI Tech. 2018, 16-4 - IMG_20190202_134149363.png
- 8bit Analog Chip with P-PCM (IBM) '18 IEDM - IMG_20190202_134452078.png
- エネルギー効率改善 (量子化と低電力回路技術) - IMG_20190202_134945142.png
- Technology Trend - IMG_20190202_135046754.png

「 LUT-Network ～本物のリアルタイムコンピューティングを目指して～」 渕上竜司さん

- 論より Run
- バイナリより LUT のテーブルを活用できる方がいい，というアプローチ
- https://github.com/ryuz/BinaryBrain
- https://www.slideshare.net/ryuz88/lut-network-fpgx201902/1
- 発表スライド : https://www.slideshare.net/ryuz88/lut-network-fpgx201902/1

「（仮）DNN コンパイラの歩みと最近の動向」ぼのたけ /NII 今井健男さん

- Deep Learning コンパイラ
  - nGraph Intel Nervana - https://github.com/NervanaSystems/ngraph
  - TensorFlow XLA Google - https://www.tensorflow.org/xla/
  - TVM Washington Univ. - https://tvm.ai/
    - HalideIR - https://github.com/dmlc/HalideIR
  - PlaidML Vertex.ai - https://github.com/plaidml/plaidml
  - DLVM Illinoi Univ. - http://dlvm.org/
  - Tensor Comprehensions Facebook - https://github.com/facebookresearch/TensorComprehensions
  - TIRAMISU MIT - https://www.csail.mit.edu/research/tiramisu-compiler
  - GLOW Facebook - https://facebook.ai/developers/tools/glow
  - ONNC Skymizer - https://github.com/ONNC/onnc
- Deep Learning コンパイラ 中間表現
  - グラフレベル
  - オペレータレベル
- TVM の場合
  - Operator fusion - 複数のオペレータの融合
  - Constant folding - 定数伝播，簡約
  - Static memory planning - 中間テンソルのためのメモリの確保
  - Data layout transformation - 内部のテンソル計算効率化のためにデータのレイアウトを変換
- TVM Conference - https://sampl.cs.washington.edu/tvmconf/
- TVM の方向性
  - AutoTVM - 機械学習を用いたオペレータ自動最適化
    - GRU, XGBoost
    - Halide では，Mullapudi et al. '16 がある - https://dl.acm.org/citation.cfm?id=2925952
  - VTA(ヴィータ) - TVM 専用の AI チップ
    - http://sampl.cs.washington.edu/tvmconf/slides/Thierry-Moreau-VTA.pdf
    - 2 階層の ISA(CISC ベースマルチサイクル : DENSE, ALU, LOAD, STORE + RISC ベースマイクロサイクル)-
  - TVM と連動した latency hiding - DNN コンパイル時に命令列内の依存関係を解析
  - Relay - グラフレベル中間言語

        - 参考 https://www.slideshare.net/bonotake/tvmir-relay
        - グラフ最適化からプログラム最適化へ
        - shape のチェックを型検査で
        - 自動微分可能な高階関数を採用
        - - Differential programming language - https://popl18.sigplan.org/event/popl-2018-papers-keynote-some-principles-of-differential-programming-languages
- DNN コンパイラの「正しさ」とは？
    - 1. DNN の正しい振舞いとは？ - そもそも精度劣化とかあるし
    - 2. DNN とコンパイル結果の間の等価性とは？
    - 3. 等価性をどうチェックすればいいか？
    - 4. 等価性に従えばどの最適化が適用可能なのか？

「RISC-V の現況と Esperanto Technologies のアプローチ」京都産業大学 情報理工学部 安田豊さん

- WesternDigital の RISC-V https://github.com/westerndigitalcorporation/omnixtend
- Nvidia + RISC-V - https://riscv.org/2018/08/sifive-announces-first-open-source-risc-v-based-soc-platform-with-nvidia-deep-learning-accelerator-technology/
- Esperant の David Ditze(RISC-V Tokyo の話) ... David Ditze - Crusoe の人
    - bulding the highest TeraFLOPS per Watt Machine Learning computing system
    - 要は RISC-V で Big.Little やるという話
        - (Big) Maxion: 64k L1, 4M L2, Deep Pipelines, OoO, Branch Prediciton
        - (Little) Minion: 4096 コアのせる，In-Order, Vector extension of floating tensor instruction(F16, F32, F64 F128)
- tool chain もいろいろある
    - gcc, gdb, Qemu などなど
    - 商用シミュレータもある http://www.imperas.com/imperas-riscv-solutions

「HBM-FPGA をさわってみた」長瀬産業 西沢正登さん

- Q: 自然言語のベクトルとかに使えるといいのでは？ by 佐藤さん

「Deep Learning 推論を高速化するソフトウェア技術」Idein 中村晃一さん

- https://github.com/nineties
- ハードウェアに依存したくない場合がある - スマホとか.
- モデルアーキテクチャ - よいモデル重要
    - モデルの精度と計算量 https://github.com/tensorflow/models/blob/master/research/slim/nets/mobilenet_v1.md
        - 1% 稼ぐのは大変 (特に 100% に近い場合). 正しい目を持ちましょう.
        - 参考 https://www.slideshare.net/ren4yu/deep-neural-network-79382352
- 例 1: アルゴリズム，ループ融合，メモリレイアウトで 4 倍くらいはかわる
- 例 2: MobileNet/STM32H7
    - Tightly Coupled Memory) 使う，SIMD 使う，アライメントきをつける 9 秒 ->3.1 秒
    - float32->float16 で 1.1 秒
- レイヤー融合
    - Conv. Batch -> Conv.
    - Conv. ReLU -> Conv. + Relu (Relu を最内ループでインライン化)
- アルゴリズム
    - 2D Conv. のアルゴリズム Direct, im2col, Winograd, FFT
    - 参考 : Convolution の数理とアルゴリズム - https://speakerdeck.com/nineties/convolutionfalseshu-li-toarugorizumu
- テンソルの shape によって最適実装は変わる
    - 入力の方とか出力の方とか

    - フィルタサイズ (3x3 とか 1x1 とか) でも変わる
- VidroCore IV アーキテクチャ
    - 参考 : Using Raspberry Pi GPU for DNN - https://www.slideshare.net/notogawa/using-raspberry-pi-gpu-for-dnn
    - 書き戻しが細い (キャッシュとおらない DMA) のがネック
    - py-videocore https://github.com/nineties/py-videocore
    - QMKL https://github.com/Idein/qmkl
- Actcast というサービスを開発中 - https://actcast.io/
- FPGA 使いたいのはレイテンシつめたいところ．制御系.
- 発表スライド - https://speakerdeck.com/nineties/deep-learningtui-lun-wogao-su-hua-surusohutoueaji-shu

「TensorFlow XLA：XLA とは、から、最近の利用事例について」@vengineer（ソースコード解析職人）さん

- 参考 XLA.jl を試してみた - https://qiita.com/antimon2/items/ccfb5c2353d99fcb1976
- 参考 Julia から Cloud TPU を使う論文の、ざっくりまとめ - https://kiszk.github.io/2018/12/19/TensorFlow-Julia-TPU-XLA/
- 参考 Introducing PyTorch across Google Cloud https://cloud.google.com/blog/products/ai-machine-learning/introducing-pytorch-across-google-cloud
    - How To Build And Run PyTorch For TPU - https://github.com/pytorch/xla
- 参考 JAX: Autograd and XLA - https://github.com/google/jax
- 参考 LeFlow: XLA - https://github.com/danielholanda/LeFlow
    - LeFlow: Enabling Flexible FPGA High-Level Synthesis of Tensorflow Deep Neural Networks https://arxiv.org/pdf/1807.05317.pdf
    - XLA->LLVM->(LegUp)->Verilog HDL

「MN-Core について」PFN 金子紘也さん

- PFN としては Training の需要が大きい
- Training における演算
    - 誤差逆伝搬．途中の値も覚えておかないとダメ.
    - 大きな GEMM として書き下せる．学習時は基本，密行列.
    - V100 のピーク性能向上は fp16 GEMM 専用エンジン (Tenssor Core)
    - 図の参考は http://cs231n.stanford.edu/
- データ並列による分散学習
    - バッチサイズ増やして → GPU にばらまいて → All-reduce
    - 単純にバッチサイズ増やすと精度劣化．テクニックが必要.
- MN-1a = Tesla P100 1024, MN-1b = Tesla V100 512．インフィニバンド接続.
- Deep Learning の研究動向
    - SoTA ではモデルサイズは増える
        - 画像から動画 / 立体，巨大化
        - - 時間方向 / 空間方向への Conv.HD
        - MoE(Mixture of Experts)
        - NAS(Netwrok Architecture Search)
        - - ネットワークアーキテクチャ自体を自動探索する試み
- MN-Core - 深層学習用プロセッサ
    - 階層メモリ型 SIMD アーキテクチャ．512MAB を 1chip に集積 (MAB -> L1B -> L2B -> Chip)
        - 倍 / 単 / 半精度演算はロジックは共有している．命令で切り替え.
        - 1 TFLOPS/W(半精度)，500W
    - 空冷
    - 2020 年に運用予定
- 計算能力は競争力の源泉

    - NIPS の論文提出締切で大手クラウドの GPU が枯渇 https://www.theregister.co.uk/2017/05/22/cloud_providers_ai_researchers/?mt=1495474350040
- HW をささえる SW
    - ChainerX https://docs.chainer.org/en/latest/chainerx/index.html
        - 高速な自動微分の実装，選択可能な backend
        - 参考 : ChainerX とりあえず入れてみよう https://qiita.com/SatoshiTerasaki/items/defbb1ea49b88c452118
    - Chainer-compiler https://github.com/pfnet-research/chainer-compiler
        - Python から拡張 ONNX フォーマットへの convert
        - 拡張 ONNX 上における計算グラフの最適化，自動微分

「私の MNIST の FPS は 530000 です。ですがもちろんフルパワーで（以下略」 なかはらさん

- 参考 : Can FPGAs beat GPUs in Accelerating Next-Generation Deep Neural Networks? - http://isfpga.org/fpga2017/slides/D1_S1_InvitedTalk.pdf
- CNN の結果可視化してみる → 情報分類 → XOR 使ってエントロピーを最大化する問題
    - クラス分類はいい
    - 回帰にはあまりよくない
- 参考 : Optuna: A hyperparameter optimization framework - https://github.com/pfnet/optuna
    - Chainer + Optuna の例 - https://github.com/pfnet/optuna/blob/master/examples/chainer_simple.py
- 関数分解法で回路分割
    - エンコーダとみなす (どれだけ違うか，で小さくできる)
- 参考 : Analysis and synthesis of weighted-sum functions https://ieeexplore.ieee.org/document/1624513
- 参考 : Logic synthesis for a single large look-up table https://ieeexplore.ieee.org/document/528842

■あとでよむ

- RAPIDNN: In-Memory Deep Neural Network Acceleration Framework https://arxiv.org/pdf/1806.05794.pdf
- SDA: Software-Defined Accelerator for LargeScale DNN Systems https://www.hotchips.org/wp-content/uploads/hc_archives/hc26/HC26-12-day2-epub/HC26.12-5-FPGAs-epub/HC26.12.545-Soft-Def-Acc-Ouyang-baidu-v3--baidu-v4.pdf
- 3D rendering in fpga - http://www.cs.columbia.edu/~sedwards/classes/2014/4840/reports/BallBalance-presentation.pdf, http://www.cs.columbia.edu/~sedwards/classes/2014/4840/reports/BallBalance.pdf
- Learning Transferable Architectures for Scalable Image Recognition https://arxiv.org/abs/1707.07012