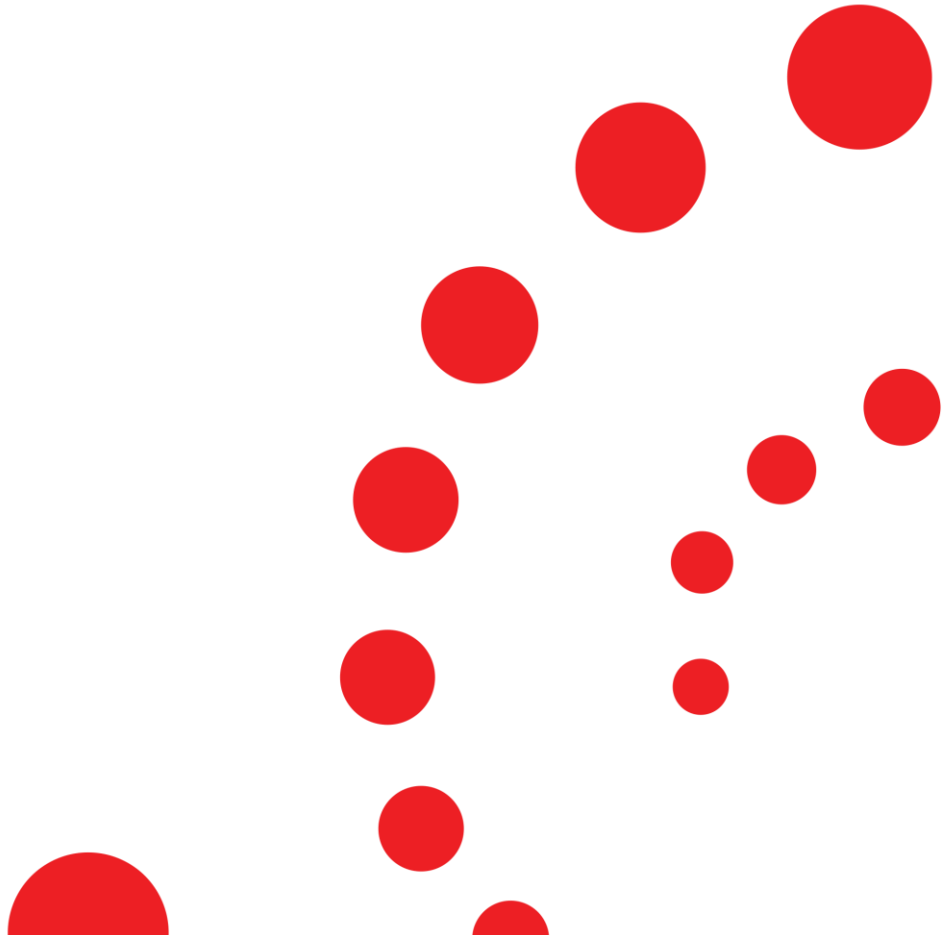
FUTUREWEI
Technologies

OSDI 2022 Trip Report

9/30/2022

OpenMindSpore Project
Silicon Valley System Software Lab



Contents

- Overview
- Best paper awards
- Selected papers of interests
- Interaction with Academia and Industry
- Comments

Contents

- Overview
- Best paper awards
- Selected papers of interests
- Interaction with Academia and Industry
- Comments

16th USENIX Symposium on Operating Systems Design and Implementation (OSDI'22)

- Date: July 11-13, 2022
- Location: Carlsbad, CA, USA
- Held virtual for the last two years (2020 and 2021)
- Held in hybrid format, both in-person and virtual
- 253 papers submitted, 49 paper accepted (19.4% acceptance rate)
  - 33 earned "Available" badge (94%), 31 earned "Functional" badge (88%), 27 earned "Result Reproduced" badge (77%)
- 52 posters accepted
  - 28 of them correspond to accepted OSDI papers, 24 independent submissions

OSDI 2022 Sponsors

- Platinum sponsor
  - Amazon
- Gold sponsor
  - Google
- Silver sponsors
  - Akamai, Apple, Futurewei Technologies, Meta, Microsoft, Moloco
- Bronze sponsors
  - Baidu, exotanium, IBM, Two Sigma, VMWare
- Open access sponsor
  - NetApp

Technical Sessions

- Distributed storage and far memory
- Bugs
- Persistent memory
- Machine learning 1
- Potpourri
- Storage
- Formal verification
- Machine learning 2
- Isolation and OS services
- Security and private messaging
- Managed languages
- Recommenders and pattern mining

Contents

- Overview
- Best paper awards
- Selected papers of interests
- Comments

Best Papers

- XRP: In-Kernel Storage Functions with EBPF (Columbia Univ)
  - A framework allowing application to execute user-defined storage functions from eBPF hook in NVMe driver, safely bypassing most of OS kernel's troage stack, resulting in significantly improving throughput and latency in key-value stores.
  - Source code available at https://github.com/xrp-project/XRP.
    - Modified Linux kernel and WiredTiger for XRP support
- MemLiner: Lining up Tracing and Application for a Far-Memory-Friendly Runtime (UCLA)
  - A runtime technique that improves the performance of far-memory systems by "lining up" memory accesses from the application and the garbage collection (GC), reducing local-memory working set and remote-memory prefetching through simplified memory access patterns.
  - Source code available at https://github.com/uclasystem/MemLiner.
    - Modified Linux kernel and GC in OpenJDK

Contents

- Overview
- Best paper awards
- Selected papers of interests
- Comments

OSDI Presentations of Interests

- SparTA: Deep-Learning Model Sparsity via Tensor-with-Sparsity-Attribute (Microsoft)
- ROLLER: Fast and Efficient Tensor Compilation for Deep Learning (Univ of Toronto, Microsoft Research)
- Walle: An End-to-End, General-Purpose, and Large-Scale Production System for Device-Cloud Collaborative Machine Learning (Alibaba)
- Unity: Accelerating DNN Training Through Joint Optimization of Algebraic Transformation and Parallelization (Stanford U)
- Orca: A Distributed Serving System for Transformer-Based Generative Models (SNU)
- Microsecond-scale Preemption for Concurrent GPU-accelerated DNN inference (SJTU)
- Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning (UC Berkeley)
- Efficient and Scalable Graph Pattern Minding on GPUs (MIT)

SparTA: Deep-Learning Model Sparsity via Tensor-with-Sparsity-Attribute (Microsoft Research)

- End-to-end approach to model sparsity via Tensor-with-Sparsity-Attribute (TeSA)
  - TeSA enables specification and propagation of the sparsity attributes and patterns, creating highly efficient, specialized operators.
- SparTA delivered 1.7x-8.4x average speed up on inference latency
  - Compared to PyTorchJIT, TensorRT, TVM, TVM sparse, Rammer, Rammer sparse, and OpenVINO (CPU).
- SparTA implemented on Rammer.
  - Source code available at https://github.com/microsoft/SparTA

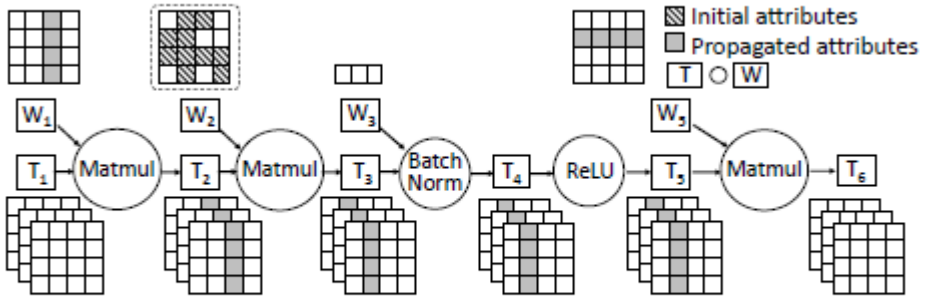


Figure 1: The sparsity attribute of one tensor can be propagated along the deep learning network.

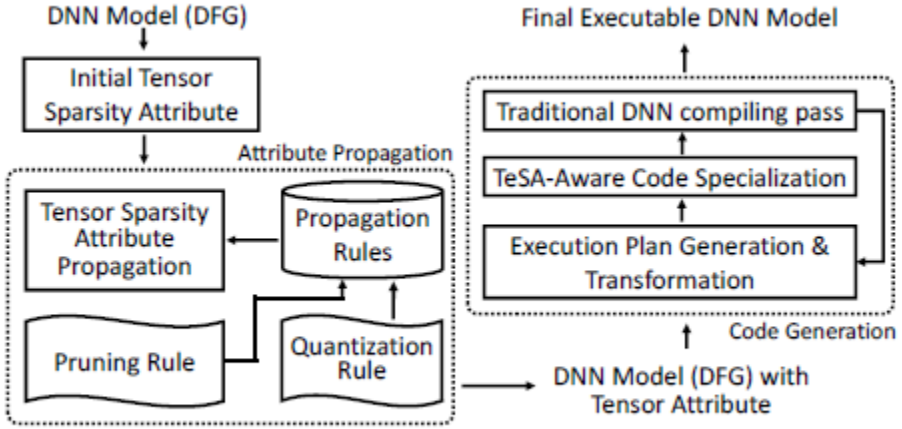


Figure 2: The system architecture of SparTA.

MindSpore Perspective

- MindSpore has an object to represent sparse tensors.
  - SparseTensor (deprecated in 1.7.0)
  - COOTensor (support from 1.7.0)
    - COO (Coordinate) format (compared to CSR/CSC)
- However, there is no notion of propagation of sparse attributes from one tensor to the other.
- Integrating TeSA into AKG would be major work.

ROLLER: Fast and Efficient Tensor Compilation for Deep Learning (Microsoft Research)

- rTile of ROLLER is a new tile abstraction that encapsulates tensor shapes aligning with the key features of the underlying accelerator, achieving faster compilation and efficient kernel code generation.
- The performance of rProgram based on recursive rTile-based construction algorithm can be evaluated efficiently with micro-perofmrnace model, without real device.
- ROLLER is implemented based on two open-source DNN compilers, TVM and Rammer.
- Evaluation covers NVIDIA CUDA GPUs, AMD ROCm GPUs, Graphcore IPUs.

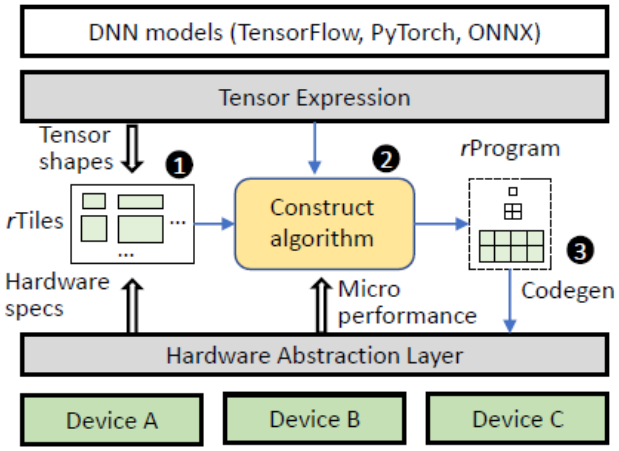


Figure 2: System overview of ROLLER.

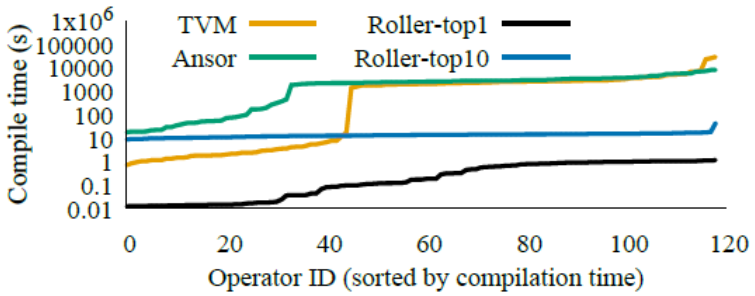


Figure 11: Compilation time for each operator.

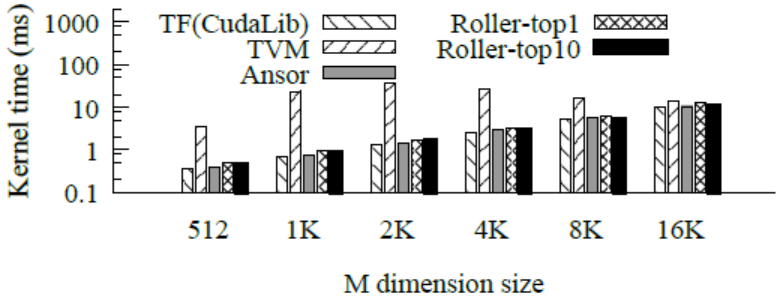


Figure 12: Kernel time for MatMul operator with different sizes of M in BERT-Large model, K=1024, N=4096.

FUTUREWEI Technologies

MindSpore Perspective

- ROLLER's rTile is the basic computing unit to compose a tensor computation and encapsulates a multi-dimensional tile shape defined along each axis of a given tensor expression expr, and rTile must align with both underlying hardware features and tensor shapes in tensor expression, which minimizes search space for DNN compilers.
- MindSpore's tensors are generic and computation on tensors are search for optimal parameter space is constrained by polyhedral constrained optimization, so potentially search space could be large.
  - Zhao et al., (Huawei), AKG: Automatic Kernel Generation for Neural Processing Units using Polyhedral Transformations, PLDI 2021.
- Recent research on MindSpore introduced the constraint tree abstraction which may be generated by a non-linear optimizer and injected to the polyhedral optimization process to build better solutions.
  - Bastoul et al. (from Huawei, France), "Optimizing GPU Deep Learning Operators with Polyhedral Scheduling Constraint Injection", CGO 2022.
- However, the approach above did not address the issue of long compilation time.

Walle: An End-to-End, General-Purpose, and Large-Scale Production System for Device-Cloud Collaborative Machine Learning (SJTU & Alibaba)

- Platform for device-cloud collaborative ML
  - Compute container based on MNN (Mobile Neural Network)
    - Geometric computing using raster operator
    - Atomic operator optimization
    - Semi-automatic search mechanism to quickly identify best backend
  - Modified Python thread-level VM, removing GIL(Global Interpreter Lock)
  - On-device stream processing
    - Trie-based event trigger mechanism

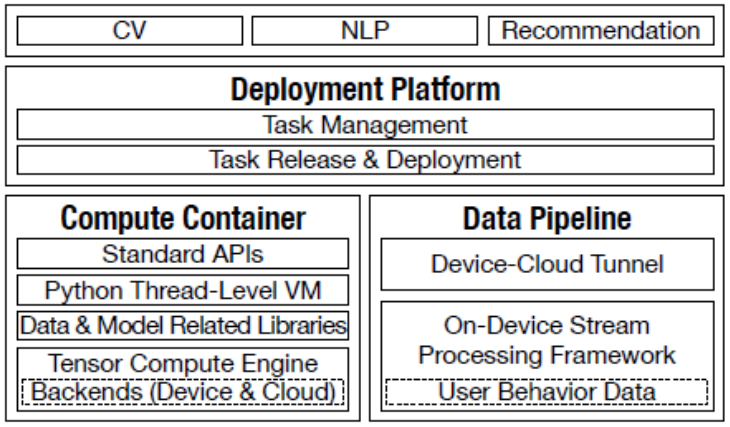


Figure 2: Architecture of Walle.

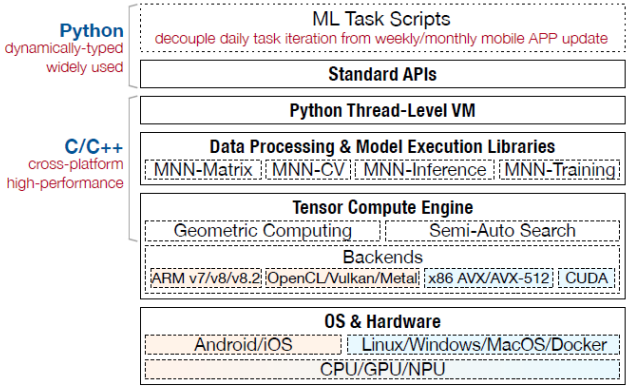


Figure 3: Architecture of compute container.

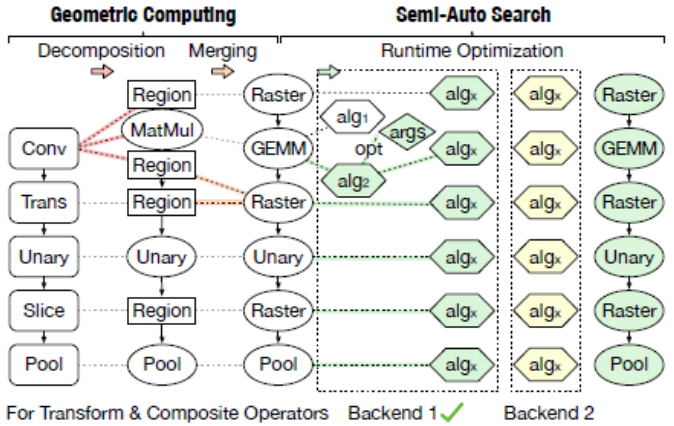
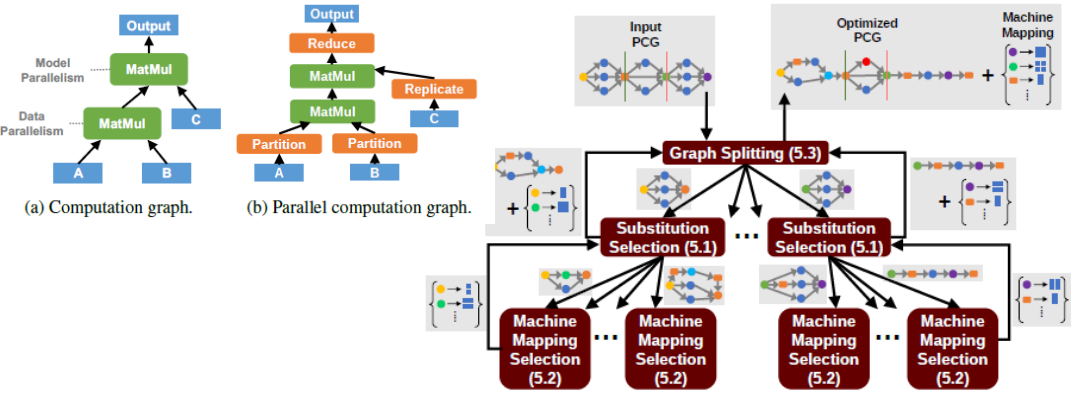


Figure 5: Geometric computing and semi-auto search.

FUTUREWEI Technologies

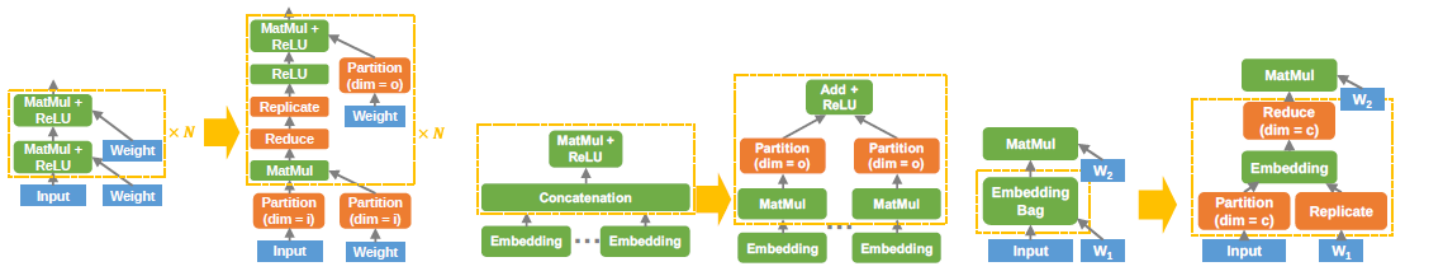
Unity: Accelerating DNN Training Through Joint Optimization of Algebraic Transformations and Parallelization (Stanford & CMU)

- Jointly optimizing algebraic transformations (e.g., operator fusion) and parallelization in distributed DNN training, while maintaining scalability
  - Unified parallel computation graph (PCG) simultaneously expresses computation, parallelization, and communication.
  - Six parallelization operators capture computation and communication costs associated with different parallelization strategies.
    - Partition and combine, replicate and reduce, pipeline and batch
  - Three-level hierarchical search algorithm
    - Fast heuristic to identify candidate substitutions (using modified TASO)
    - Followed by more expensive formal verification (using automated theorem prover Z3)
    - Cost-based backtracking search algorithm for substitution selection (from TASO)
- Implemented on top of FlexFlow, distributed multi-GPU runtime for DNN training
  - Experiments performed on Summit supercomputer (two IBM POWER9 CPUs, 512GB memory, 6 Nvidia V100 GPUs, NVLink on Mellanox EDR 00Gb Infiniband).



(a) Computation graph. (b) Parallel computation graph.

Figure 12: High-level depiction of Unity's hierarchical search.



(a) Multi-layer perceptron. (b) `Concatenation`. (c) `EmbeddingBag`.

Figure 16: Example joint optimizations of computation graph and parallelization discovered by Unity. For `Partition`, i and o indicate the input and output channel dimensions of a matrix multiplication.

FUTUREWEI Technologies

MindSpore Perspective

- Combining computation and communication cost into a single optimization problem is similar to MindSpore's TR (tensor redistribution) idea.
  - Unity can handle sequential pipeline parallelism, whereas MindSpore leaves it up to user by using semi-automatic parallelism mode.
  - MindSpore considers re-materialization, whereas Unity does not now.
  - Unity may violate memory constraints, whereas MindSpore does not.
- Unity reports that it can scale with over 300 operators and machines with 192 GPUs while keeping search times below 20 minutes.
  - Negligible compared to hours or days of training times for modern DNNs.

Orca: A Distributed Serving System for Transformer-Based Generative Models (SNU & FriendliAI)

- Serving for large-scale transformer-based models
  - Request-based scheduling preventing early return of finished requests to clients
    - Solution: iterative scheduling
  - Issue with batching from iterative scheduling
    - Solution: selective batching of requests in the same phase
- Distributed architecture
  - Intra-layer and inter-layer parallelism
  - Engine master and worker processes
  - Minimizing CPU-GPU synchronization (compared to Megatron-LM and FastTransformer) by separate NCCL and gRPC communication for GPU and CPU respectively.
- 36.9x throughput improvement at the same of latency compared to Nvidia's FastTransformer
- Cost of serving with 400 GPT3 175B instances for same target median latency and throughput
  - Baseline 190.6M/year vs. Orca $5.7M/year

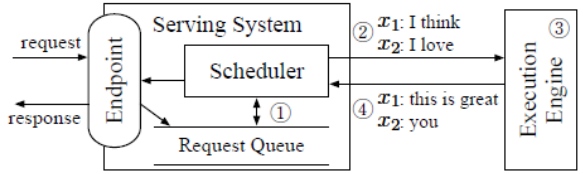


Figure 2: Overall workflow of serving a generative language model with existing systems.

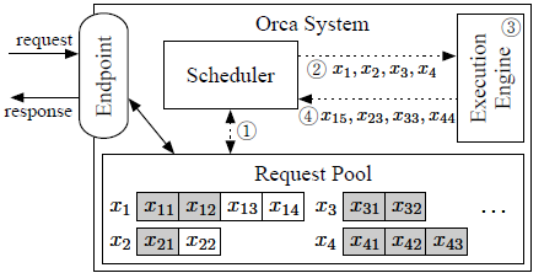


Figure 4: System overview of ORCA. Interactions between components represented as dotted lines indicate that the interaction takes place at every iteration of the execution engine.

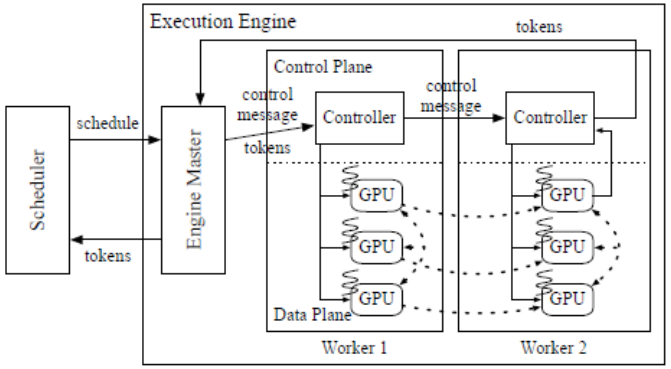


Figure 7: An illustration of the distributed architecture of ORCA's execution engine using the parallelization configura-

Microsecond-scale Preemption for Concurrent GPU-accelerated DNN inference (SJTU)

- REEF: GPU-accelerated DNN inference serving system
  - Reset-based kernel preemption on GPU
    - Supporting both AMD GPUs (open-source via approximation of preempted tasks) and Nvidia GPUs (REED-N for closed-source via lazy eviction)
  - Priority-based scheduling of real-time and best-effort tasks on GPU
    - Maintaining one real-time queue and many best-effort queues.
    - Round-robin scheduling when no task in real-time queue; real-time scheduling when there is a task in real-time queue
  - Dynamic kernel padding for controlled concurrent execution of tasks on GPU
    - Combining real-time and best-effort kernels into one at compile time
    - Model compiler from modified Apache TVM

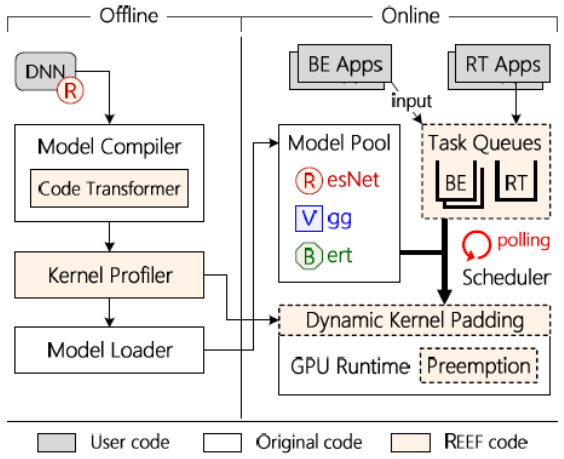


Fig. 5: *Architecture of* REEF. *Modules in boxes with dashed border are on the critical path of serving DNN inference requests. Other modules do not directly impact serving latency and throughput.*

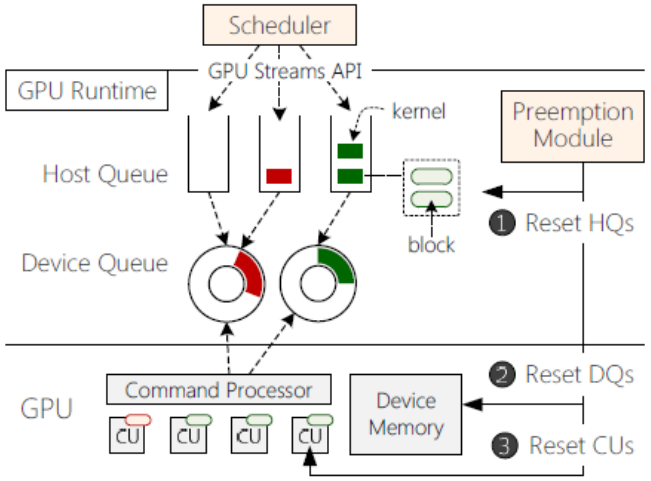


Fig. 7: *Extended GPU runtime in* REEF *for instant preemption.*

FUTUREWEI Technologies

Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning (UC Berkeley)

- Automation of unified data, operator, and pipeline parallelism for large deep learning model training on distributed compute devices
  - Hierarchical scheduling: inter- and intra-operator parallelism
  - ILP solver for intra-op parallelism (linearization from quadratic form) and DP solver for inter-op parallelism (based on TeraPipe)
  - Cross-mesh re-sharding to address many-to-many broadcast problem
- Implementation
  - Jax as frontend and XLA as backend
  - Ray actor to implement device mesh worker
  - XLA runtime for executing computation
  - NCCL for communication

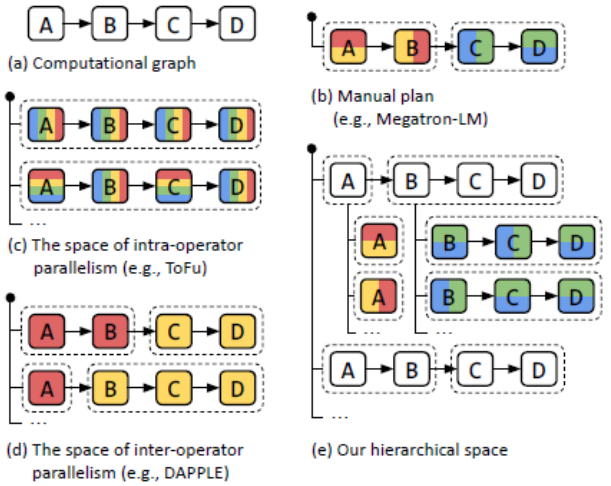


Figure 1: Generation of parallelization plans for a computational graph shown in (a). Different colors represent different

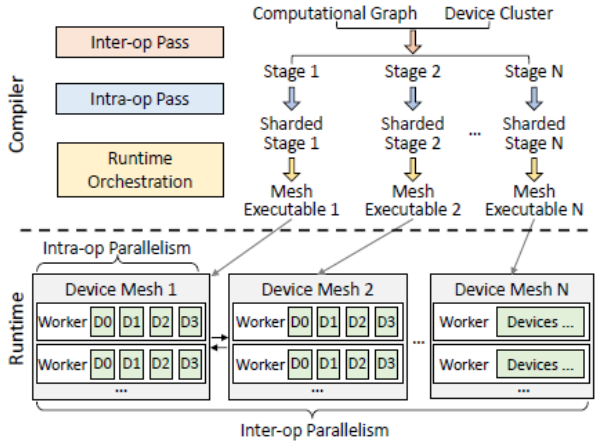


Figure 3: Compiler passes and runtime architecture. A sharded stage is a stage annotated with the sharding specs generated by intra-op pass.

FUTUREWEI Technologies

MindSpore Perspective

- Alpa and MindSpore both support automatic parallelism.
  - Alpa has 2-level intra- and inter-operator (pipelining) parallelism.
  - MindSpore supports pipelining parallelism in semi-automatic mode only.

Efficient and Scalable Graph Pattern Minding on GPUs (MIT)

- G2Miner: graph pattern mining framework on GPUs
  - Pattern-aware, input-aware, and architecture-aware search strategies
  - Code generator automatically generates pattern-aware CUDA code.
- GPU optimizations
  - Mitigating thread divergence: warp-centric data parallelism
  - Improving load balancing: by reducing task granularity via edge parallel (vs vertex parallel)
  - Reducing memory consumption: by orientation (from undirect graph to direct) and label frequency
  - Pruning search space: for counting-only problem and local graph search for hub-patterns
  - Improving efficiency based on GPU hardware features: SIMD-aware set operations

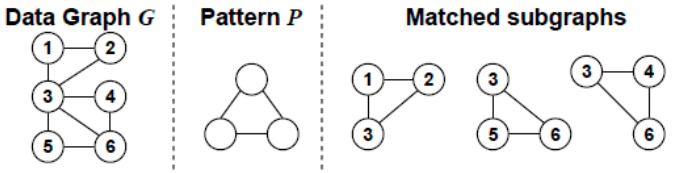


Figure 1: Graph Pattern Mining example. The pattern $\mathcal{P}$ is a triangle, and 3 triangles are found in the data graph $\mathcal{G}$.

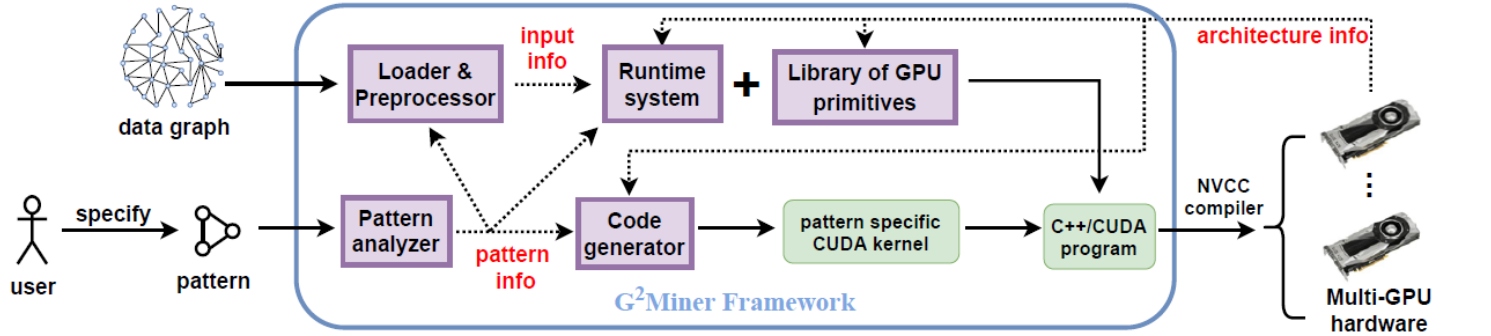


Figure 2: G^{2}Miner system overview. It contains a graph loader, a pattern analyzer, a runtime, a library of GPU primitives and a code generator.

FUTUREWEI Technologies

Contents

- Overview
- Best paper awards
- Selected papers of interests
- Comments

Comments

- OSDI is one of top computer systems conferences, which is held once every two years.
- Software Lab currently focuses on Machine learning & deep learning systems, also previously on distributed systems. OSDI is highly relevant to our interests.
- OSDI 2022 has 3 sessions (out of 12) for machine learning and recommender systems this year, indicating more interests and audience from ML/DL systems research.
- OSDI 2022 was held jointly with ATC (Annual Technical Conference) 2022 and allowed virtual attendance as well, therefore provided more flexible participation for audiences.