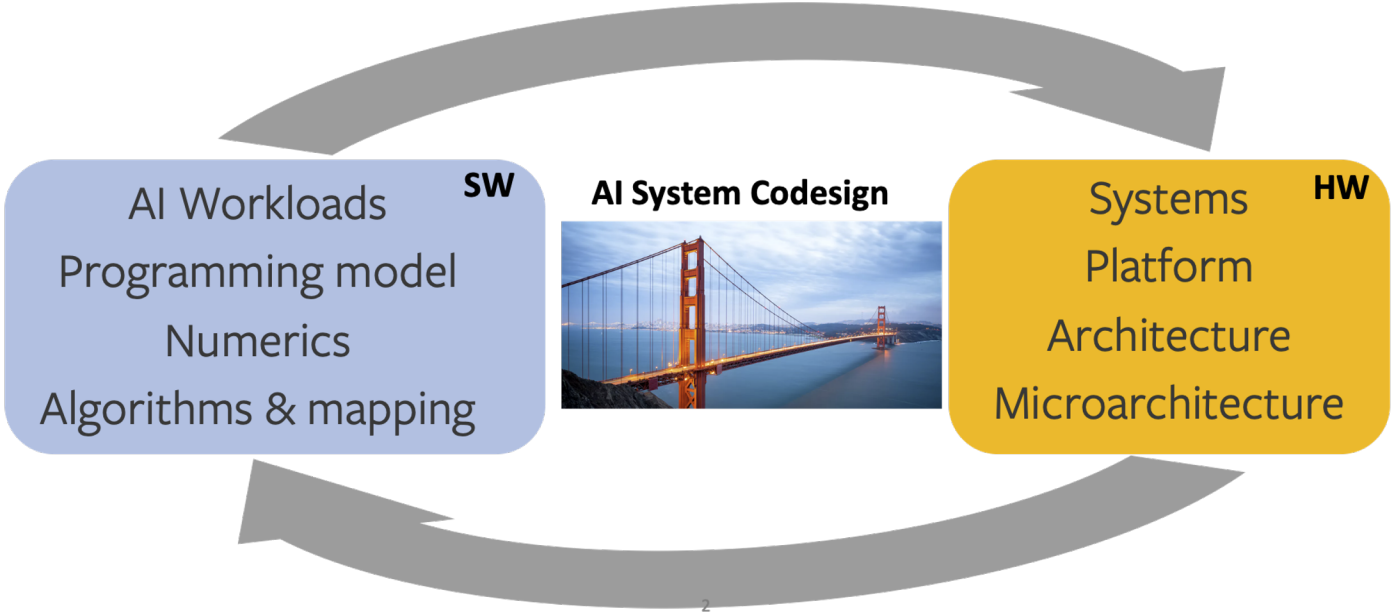
FlexFlow + Legion
- fast/flexible prototyping

Dheevatsa Mudigere
AI System Co-Design, Facebook

AI HW/SW System Co-design team

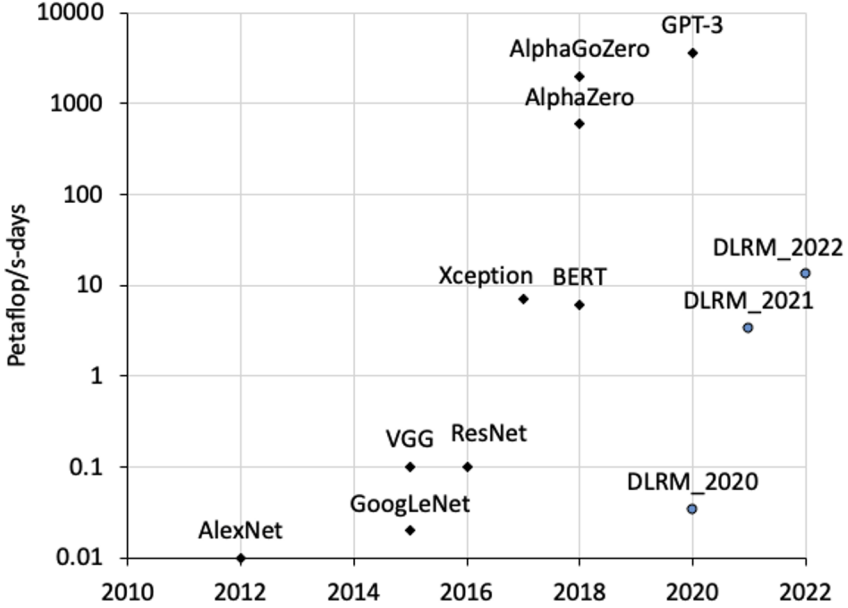


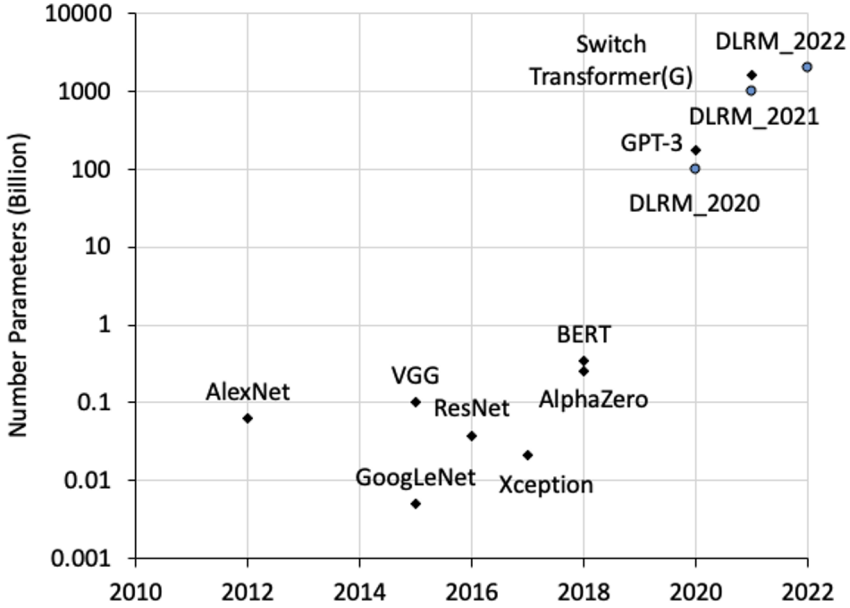
AI at FB

Across a large number of applications/services and at **very large scales**, driving a **large portion of our overall infrastructure both HW and SW**

- Ranking and recommendation - NewsFeed, Video, Ranking, Ads, Search…
- Content understanding - Computer vision, Speech, Translation, NLP, Video...

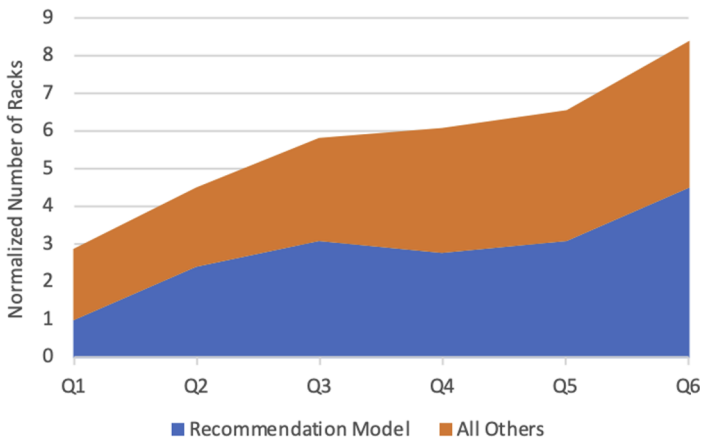
AI/DL models are diverse and are growing (fast) !



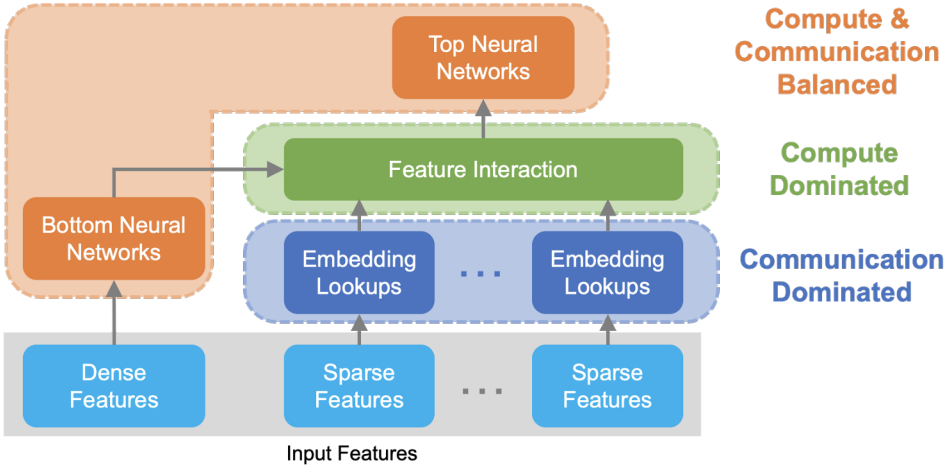


Deep learning recommendation models (DLRMs)

DLRM very important class of models and perfect use-case to highlight all of the requirements/benefits for large scale AI training



M. Naumov, J. Kim, D. Mudigere, et al. Deep Learning Training in Facebook Data Centers: Design of Scale-up and Scale-out Systems, ArXiv, 2020



Distributed training a necessity

Total compute for training at 100s PetaFlops-days, with model size trending to ~10s of TB

Practical deployments - increased frequency of such training jobs , having to almost continually train on different models/data

Training as fast as possible is a requirement and can be achieved only with scaling out training

- High performance scalable distributed training is an absolute necessity !

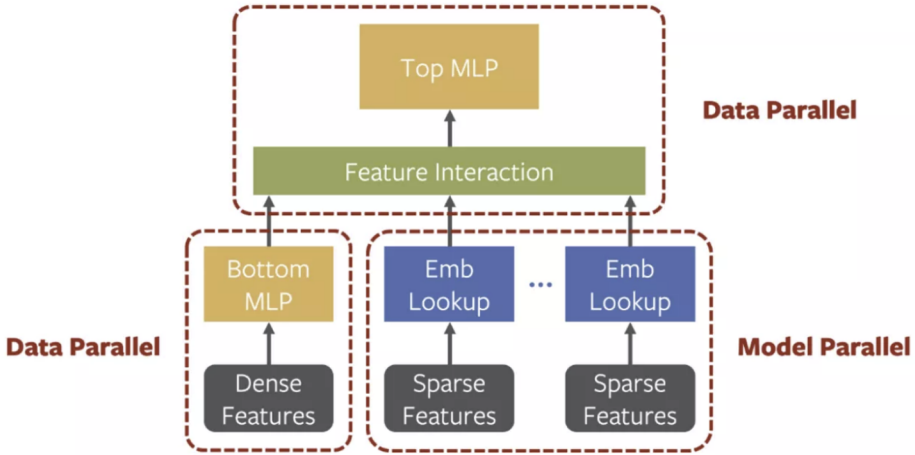
Industry compute infrastructure designed/optimized for efficiency, not only performance unlike HPC

- Additional requirement of scaling performance on large-scale datacenters - elasticity, reliability, heterogeneity, scale vs. capacity tradeoff,...

Parallelism - all (possible) ways

Large-scale DNN training necessitates a more holistic approach to parallelism !

- Large models, complex architectures and increasing training throughput cannot be supported by traditional data or model parallelism
- Parallelizing along all possible dimensions: data + model (params, sharding) + pipelining, with the optimal strategy being a non-trivial combination of all of the above



Deep learning recommendation model (DLRM)

D. Mudigere, Y. hao, J. huang et al. High-performance, Distributed Training of Large-scale Deep Learning Recommendation Models, ArXiv, 2021

Requirements for Training at scale

Maximum flexibility for expressing parallelism, without loss in performance and dev efficiency

- Requires significant experimentation to identify optimal strategy, under different system and usage constraints
- Heterogeneous systems, requires re-scaling and support for elasticity, with varying strategies at different scales

Same model with different optimal parallelization strategies depending use-case/scenario

- Online vs. offline training, resource constraints

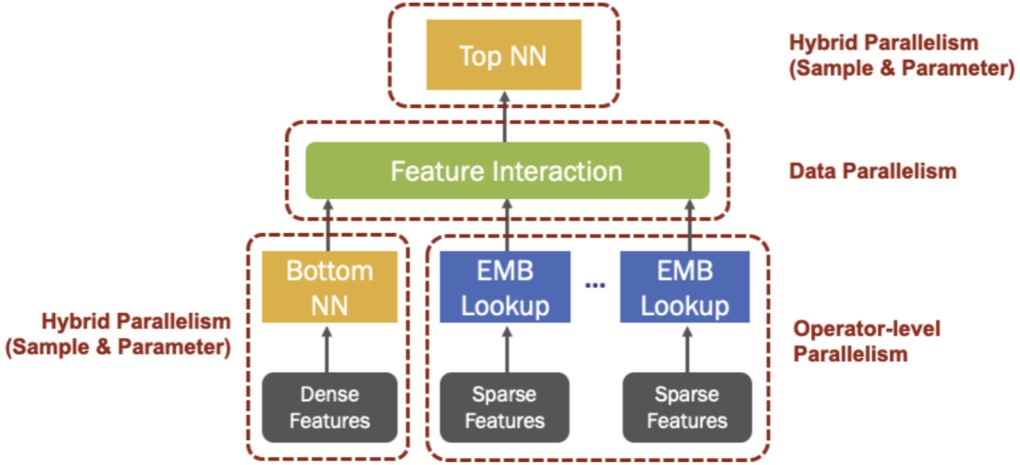
Models and their architectures constantly evolving and changing rapidly

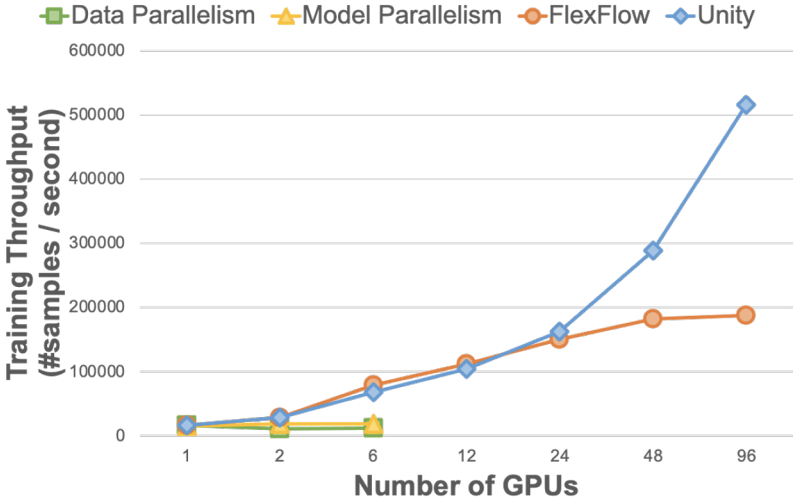
FlexFlow + Legion

Ideal as a fast + flexible tool for exploration

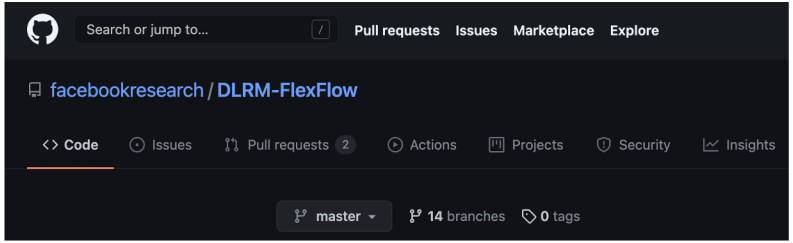
- allows for prototyping and evaluating larger number of possible parallelization strategies
- high performance, with optimized low-level primitives for both compute and comms due to the HPC roots/pedigree
- automated search to identify performant parallelization strategy, manual strategies generally time consuming and sub-optimal
- open-source and relatively easy to use
- HW/System co-design, using performance estimation and evaluation

DLRM + FlexFlow



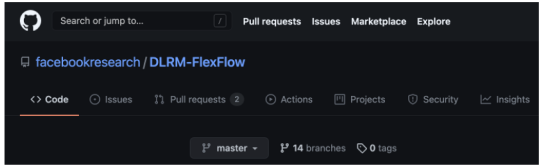


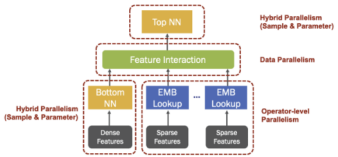
DLRM + FlexFlow



Prototype implementation of FB DLRM in FlexFlow - to evaluate and identify optimized distributed training strategy

- Adding support for specific DLRM operators - embedding_bag, sparse optimizers, interaction ops
  - Specialized NN modules, operators
- Optimized collectives for vanilla data parallel gradient allreduce
- Heterogeneous strategies GPU + CPU
  - Subset of embeddings w/ lower lookups on CPU, other embedding and compute-intensive MLPs on GPUs
- Streaming data-loader, from FB's exabyte-scale network store
- Extend for alternate network - IB, RoCE
- Unit and integration testing for current and new ops to allow for wider usage

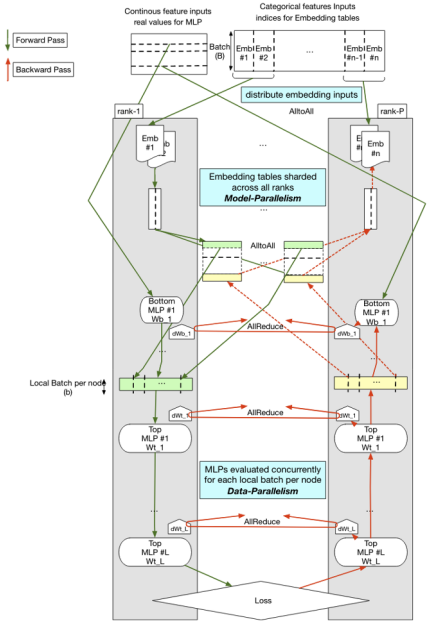




DLRM + FlexFlow

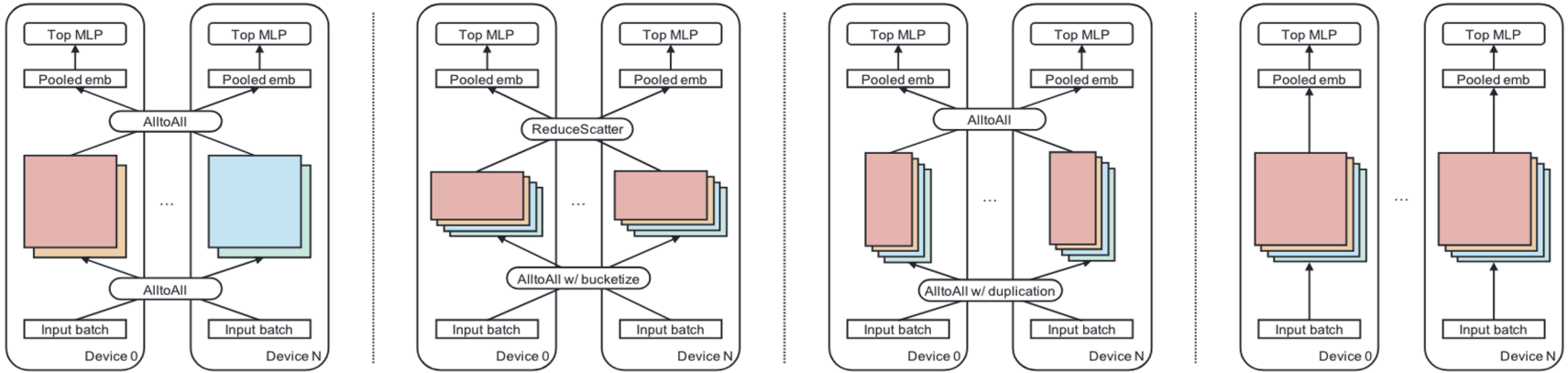
How did it help ?

- Demonstrating achievable performance on candidate platform(s) and upper bound performance targets
- Target HW/System evaluation and modification - co-designing HW/SW
- Fast prototyping new, alternate and atypical model implementations and performance
- Guiding the subsequent hardened PyTorch-based implementation now deployed in FB-prod, training model with up to 13 Trillion params



D. Mudigere, Y. hao, J. huang et al. High-performance, Distributed Training of Large-scale Deep Learning Recommendation Models, ArXiv, 2021

Parallelism - all (possible) ways



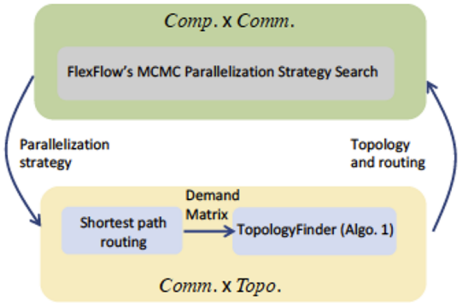
D. Mudigere, Y. hao, J. huang et al. High-performance, Distributed Training of Large-scale Deep Learning Recommendation Models, ArXiv, 2021

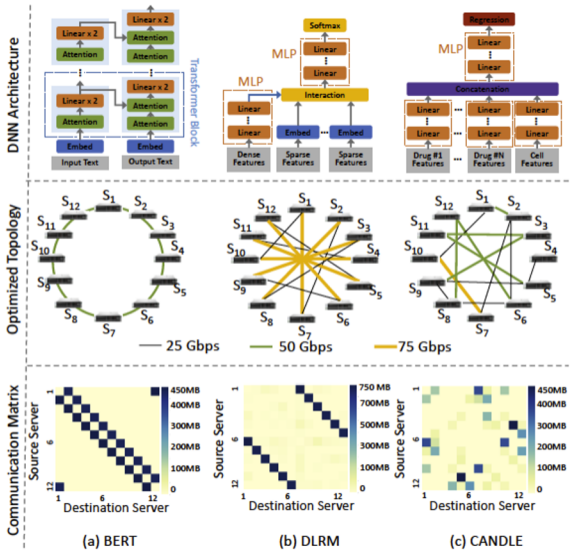
FlexFlow for System Co-design

Extending FlexFlow to also incorporate system design as part of global optimization problem

Joint optimization of network topology along with DNN parallelization strategy (with Prof. Manya Ghobadi's group at MIT)

- Identify most optimal network topology for a given ML workload (DLRM)
- Evaluate future paradigms such as reconfigurable networks, which can changed to an optimal setting as per target app/model





Some other ongoing/emerging uses

ONNX support for FlexFlow

- Extending FlexFlow to other DNN models

FlexFlow + TASO (Unity)

- More comprehensive e2e solution combining graph and parallelization optimizations for distributed training/inference
- Unified representation combining both parallelization and execution graphs and using substitutions on this graph for combined optimization of e2e distributed perf

Production requirements/challenges

What limits FlexFlow + legion from being adopted for wider prod deployment

- Exploration + Flexibility vs. Specialization for Common use-cases
  - Have to balance support, by also optimizing for the more common strategies (for e.g. fast collective operations) while retaining the improved flexibility, which comes at non-zero cost/overheads
- Supporting common model authoring frontends/frameworks
  - Reimplementing models for FF is significant barrier for entry and wider adoption, more seamless integration with existing models and FE/FW is necessary
- Increased focused on stability and reliability
  - Prod adoption prerequisite - stable code base, maintainability, support for fault tolerance/recovery
- Limited/reduced third-party dependencies
  - Additional libs, sw support increases the all of the above to those dependencies as well

Looking ahead

More complex and demanding models aren't going away soon and hence more holistic approach for distributed training is quickly becoming the norm rather than just an exception

Manual parallelization is nearing the end of the road, automated/adaptive parallelization the most promising way forward

Performance requirements continues to go up, paving the way for specialized/non-standard hardware and systems

Thank you :)

dheevatsa@fb.com