2018 IEEE International Symposium on High Performance Computer Architecture

Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective

Kim Hazelwood, Sarah Bird, David Brooks, Soumith Chintala, Utku Diril, Dmytro Dzhulgakov, Mohamed Fawzy, Bill Jia, Yangqing Jia, Aditya Kalro, James Law, Kevin Lee, Jason Lu, Pieter Noordhuis, Misha Smelyanskiy, Liang Xiong, Xiaodong Wang

Facebook, Inc.

***Abstract*—Machine learning sits at the core of many essential products and services at Facebook. This paper describes the hardware and software infrastructure that supports machine learning at global scale. Facebook's machine learning workloads are extremely diverse: services require many different types of models in practice. This diversity has implications at all layers in the system stack. In addition, a sizable fraction of all data stored at Facebook flows through machine learning pipelines, presenting significant challenges in delivering data to high-performance distributed training flows. Computational requirements are also intense, leveraging both GPU and CPU platforms for training and abundant CPU capacity for real-time inference. Addressing these and other emerging challenges continues to require diverse efforts that span machine learning algorithms, software, and hardware design.**

I. Introduction

Facebook's mission is to "Give people the power to build community and bring the world closer together." In support of that mission, Facebook connects more than two billion people as of December 2017. Meanwhile, the past several years have seen a revolution in the application of machine learning to real problems at this scale, building upon the virtuous cycle of machine learning algorithmic innovations, enormous amounts of training data for models, and advances in high-performance computer architectures [1]. At Facebook, machine learning provides key capabilities in driving nearly all aspects of user experience including services like ranking posts for News Feed, speech and text translations, and photo and real-time video classification [2], [3].

Facebook leverages a wide variety of machine learning algorithms in these services including support vector machines, gradient boosted decision trees, and many styles of neural networks. This paper describes several important aspects of datacenter infrastructure that supports machine learning at Facebook. The infrastructure includes internal "ML-as-a-Service" flows, open-source machine learning frameworks, and distributed training algorithms. From a hardware point of view, Facebook leverages a large fleet of CPU and GPU platforms for training models in order to support the necessary training frequencies at the required service latency. For machine learning inference, Facebook primarily relies on CPUs for all major services with neural network ranking services like News Feed dominating the total compute load.

Facebook funnels a large fraction of all stored data through machine learning pipelines, and this fraction is increasing over time to improve model quality. The massive amount of data required by machine learning services presents challenges at the global scale of Facebook's datacenters. Several techniques are used to efficiently feed data to the models including decoupling of data feed and training, data/compute co-location, and networking optimizations. At the same time, Facebook's scale provides unique opportunities. Diurnal load cycles leave a significant number of CPUs available for distributed training algorithms during off-peak periods. With Facebook's compute fleet spread over ten datacenter locations, scale also provides disaster recovery capability. Disaster recovery planning is essential as timely delivery of new machine learning models is important to Facebook's operations.

Looking forward, Facebook expects rapid growth in machine learning across existing and new services [4]. This growth will lead to growing scalability challenges for teams deploying the infrastructure for these services. While significant opportunities exist to optimize infrastructure on existing platforms, we continue to actively evaluate and prototype new hardware solutions while remaining cognizant of game-changing algorithmic innovations.

The key contributions of this paper include the following major insights about machine learning at Facebook:

- Machine learning is applied pervasively across nearly all services, and **computer vision represents only a small fraction** of the resource requirements.
- Facebook relies upon an incredibly **diverse set of machine learning approaches** including, but not limited to, neural networks.
- **Tremendous amounts of data** are funneled through our machine learning pipelines, and this creates engineering and efficiency challenges far beyond the compute nodes.
- Facebook currently relies heavily on **CPUs for inference**, and **both CPUs and GPUs for training**, but constantly prototypes and evaluates new hardware solutions from a performance-per-watt perspective.
- The worldwide scale of people on Facebook and corresponding diurnal activity patterns result in a huge number of machines that can be harnessed for machine learning tasks such as **distributed training at scale**.

2378-203X/18/$31.00 ©2018 IEEE
DOI 10.1109/HPCA.2018.00059

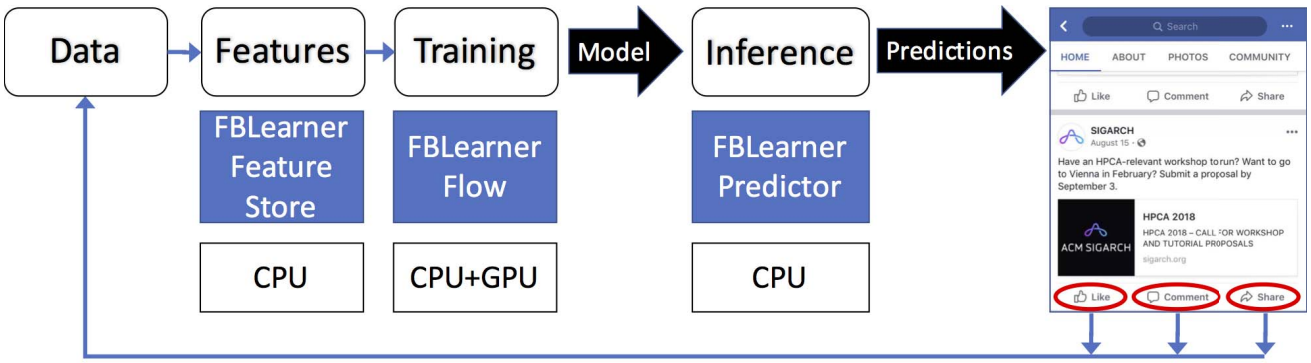

Fig. 1. Example of Facebook's Machine Learning Flow and Infrastructure.

II. Machine Learning at Facebook

Machine Learning, or ML, refers to any instance where a product leverages a series of inputs to build a tuned model, and leverages that model to create a representation, a prediction, or other forms of useful signals.

Figure 1 illustrates this process which consists of the following steps, executed in turn:

1) A **training phase** to build the model. This phase is generally performed *offline*.
2) An **inference phase** to run the trained model in production and make a (set of) real-time predictions. This phase is performed *online*.

Training the models is done much less frequently than inference – the time scale varies, but it is generally on the order of days. Training also takes a relatively long time to complete – typically hours or days. Meanwhile, depending on the product, the online inference phase may be run tens-of-trillions of times per day, and generally needs to be performed in real time. In some cases, particularly for recommendation systems, additional training is also performed online in a continuous manner [5].

One salient feature of machine learning at Facebook is the impact of the massive amounts of data that is potentially available to train the models. The scale of this data has many implications that span the entire infrastructure stack.

A. Major Services Leveraging Machine Learning

Most major Facebook products and services leverage machine learning. We discuss these services from a resource standpoint later in this document, but by way of introduction, we provide a bird's eye view of how several of the major services leverage machine learning.

- **News Feed** ranking algorithms help people see the stories that matter most to them first, every time they visit Facebook. General models are trained to determine various user and environmental factors that should ultimately determine the rank order of content. Later, when a person visits Facebook, the model is used to generate a personalized set of the best posts, images, and other content to display from thousands of candidates, as well as the best ordering of the chosen content.
- **Ads** leverages ML to determine which ads to display to a given user. Ads models are trained to learn how user traits, user context, previous interactions, and advertisement attributes can be most predictive of the likelihood of clicking on an ad, visiting a website, and/or purchasing a product [5]. Later, when a user visits Facebook, inputs are run through a trained model to immediately determine which ads to display.
- **Search** launches a series of distinct and specialized sub-searches to the various verticals, e.g., videos, photos, people, events, etc. A classifier layer is run atop the various search verticals to predict which of the many verticals to search, as searching all possible verticals would be inefficient. Both the classifier itself and various search verticals consist of an offline stage to train the models, and an online stage to run the models and perform the classification and search.
- **Sigma** is the general classification and anomaly detection framework that is used for a variety of internal applications including site integrity, spam detection, payments, registration, unauthorized employee access, and event recommendations. Sigma includes hundreds of distinct models running in production everyday, and each model is trained to detect anomalies or more generally classify content.
- **Lumos** extracts high-level attributes and embeddings from an image and its content, enabling algorithms to automatically understand it. That data can be used as input to other products and services, for example as though it were text.
- **Facer** is Facebook's face detection and recognition framework. Given an image, it first finds all of the faces in that image. Then, it runs a user-specific facial-recognition algorithm to determine the likelihood of that face belonging to one of your top-N friends who have enabled face recognition. This allows Facebook to suggest which of your friends you might want to tag within the photos you upload.

Models	Services
Support Vector Machines (SVM)	Facer (User Matching)
Gradient Boosted Decision Trees (GBDT)	Sigma
Multi-Layer Perceptron (MLP)	Ads, News Feed, Search, Sigma
Convolutional Neural Networks (CNN)	Lumos, Facer (Feature Extraction)
Recurrent Neural Networks (RNN)	Text Understanding, Translation, Speech Recognition

TABLE I
MACHINE LEARNING ALGORITHMS LEVERAGED BY PRODUCT/SERVICE.

- **Language Translation** is the service that manages internationalization of Facebook content. We support translations for more than 45 languages as the source or target language, meaning we support more than 2000 translation directions, e.g. English-to-Spanish or Arabic-to-English. With these 2K+ systems, we serve 4.5B translated post impressions every day, lowering language barriers for 600 million people who see translated posts in their News Feed every day. Currently, each language pair direction has its own model, although multi-language models are being considered [6].
- **Speech Recognition** is the service that converts audio streams into text. This provides automated captioning for video. Currently, most streams are English language, but other languages will be available in future. Additionally, non-language audio events are also detected with a similar system (simpler model).

In addition to the major products mentioned above, many more long-tail services also leverage machine learning in various forms. The count of the long tail of products and services is in the hundreds.

B. Machine Learning Models

All machine learning based services use "features" (or inputs) to produce quantified outputs. Machine learning algorithms used at Facebook include Logistic Regression (LR), Support Vector Machines (SVM), Gradient Boosted Decision Trees (GBDT), and Deep Neural Networks (DNN). LR and SVM are efficient to train and use for prediction. GBDT can improve accuracy at the expense of additional computing resources [7]. DNNs are the most expressive, potentially providing the most accuracy, but utilizing the most resources (with at least an order of magnitude compute over linear models like LR and SVM). All three types correspond to models with increasing numbers of free parameters, which must be trained by optimizing predictive accuracy against labeled input examples.

Among deep neural networks, there are three general classes in use: Multi-Layer Perceptrons (MLP), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN/LSTM). MLP networks usually operate on structured input features (often ranking), CNNs work as spatial processors (often image processing), and RNN/LSTM networks are sequence processors (often language processing). Table I shows the mapping between these ML model types and products/services.

C. ML-as-a-Service Inside Facebook

Several internal platforms and toolkits exist that aim to simplify the task of leveraging machine learning within Facebook products. The primary examples include FBLearner, Caffe2, and PyTorch. FBLearner is a suite of three tools, each of which focuses on different parts of the machine learning pipeline. FBLearner leverages an internal job scheduler to allocate resources and schedule jobs on a shared pool of GPUs and CPUs, as shown in Figure 1. Most of the ML training at Facebook is run through the FBLearner platform. Working together, these tools and platforms are designed to make ML engineers more productive and help them focus on algorithmic innovation.

FBLearner Feature Store. The starting point for any ML modeling task is to gather and generate features. The Feature Store is essentially a catalog of several feature generators that can be used both for training and real-time prediction, and it serves as a marketplace that multiple teams can use to share and discover features. Having this list of features is a good starting point for teams starting to use ML and also to help augment existing models with new features.

FBLearner Flow is Facebook's machine learning platform for model training [8]. Flow is a pipeline management system that executes a *workflow* describing the steps to train and/or evaluate a model and the resources required to do so. Workflows are built out of discrete units, or *operators*, each of which have inputs and outputs. The connections between the operators are automatically inferred by tracing the flow of data from one operator to the next and Flow handles the scheduling and resource management to execute the workflow. Flow also has tooling for experiment management and a simple user interface which keeps track of all of the artifacts and metrics generated by each workflow execution or experiment. The user interface makes it simple to compare and manage these experiments.

FBLearner Predictor is Facebook's internal inference engine that uses the models trained in Flow to provide predictions in real time. The Predictor can be used as a multi tenancy service or as a library that can be integrated in product-specific backend services. The Predictor is used by multiple product teams at Facebook, many of which require low latency solutions.

The direct integration between Flow and Predictor also helps with running online experiments and managing multiple versions of models in productions.

D. Deep Learning Frameworks

We leverage two distinct but synergistic frameworks for deep learning at Facebook: PyTorch, which is optimized for research, and Caffe2, which is optimized for production.

Caffe2 is Facebook's in-house production framework for training and deploying large-scale machine learning models. In particular, Caffe2 focuses on several key features required by products: performance, cross-platform support, and coverage for fundamental machine learning algorithms such as convolutional neural networks (CNNs), recurrent networks (RNNs), and multi-layer perceptrons (MLPs) with sparse or dense connections and up to tens of billions of parameters. The design involves a modular approach, where a unified graph representation is shared among all backend implementations (CPUs, GPUs, and accelerators). Separate execution engines serve different graph execution needs, and the Caffe2 abstraction pulls in third-party libraries (e.g., cuDNN, MKL, and Metal) for optimal runtime on different platforms.

PyTorch is the framework of choice for AI research at Facebook. It has a frontend that focuses on flexibility, debugging, and dynamic neural networks which enables rapid experimentation. With its dependence on Python for execution, it is not optimized for production and mobile deployments. When research projects produce valuable results, the models need to be transferred to production. Traditionally, this is accomplished via rewriting the training pipeline in a product environment with other frameworks. Recently we started building the ONNX toolchain to simplify the transfer process. As an example, dynamic neural networks are used in cutting-edge AI research, but it takes longer for the models to mature enough to be used in production. By decoupling frameworks, we've avoided the need of designing more complex execution engines needed for performance (such as those in Caffe2). Furthermore, researchers may prioritize flexibility over speed. In an exploration phase, performance degradations of 30%, for instance, may be tolerable, especially if it comes with the benefit of inspectability and visualization of models. However, the same degradation is not appropriate for production. This dichotomy shows up in the respective frameworks, where PyTorch provides good defaults and reasonable performance, while Caffe2 has the option to use features such as asynchronous graph execution, quantized weights, and multiple specialized backends to achieve maximum performance.

The FBLearner platform is agnostic of the framework in use, be it Caffe2, TensorFlow, PyTorch, or other alternatives, but the AI Software Platform team provides specific functionality to allow FBLearner to integrate well with Caffe2. Overall, decoupling research and production frameworks (PyTorch and Caffe2, respectively) has given us the ability to move fast on each side, reducing the number of constraints while adding new features.

ONNX. The deep learning tools ecosystem is still in its early days throughout industry. Different tools are better for different subset of problems and have varying tradeoffs on flexibility, performance, and supported platforms - similar to the tradeoffs described earlier for PyTorch and Caffe2. As a result, theres significant desire to exchange trained models between different frameworks or platforms. To fill this gap, in late 2017, we partnered with several stakeholders to introduce Open Neural Network Exchange (ONNX) [9], which is a format to represent deep learning models in a standard way to enable interoperability across different frameworks and vendor-optimized libraries. ONNX is designed as an open specification, allowing framework authors and hardware vendors to contribute to the design and to own the various converters between frameworks and libraries. We are working with these partners to make ONNX more of a living collaboration between all these tools than as an official standard.

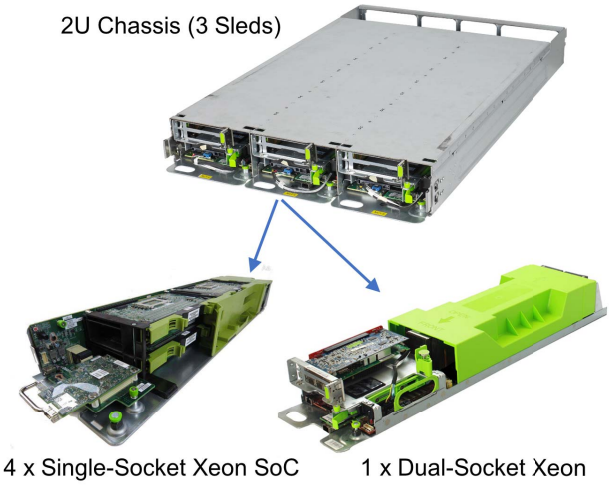

Fig. 2. CPU-based compute servers. The single-socket server sled contains 4 Monolake server cards, resulting in 12 total servers in a 2U form factor. The dual-socket server sled contains one dual-socket server, resulting in three dual-socket servers in a 2U chassis.

Within Facebook, we're using ONNX as primary means of transferring research models from the PyTorch environment to high-performance production environment in Caffe2. ONNX provides the ability to automatically capture and translate static parts of the models. We have an additional toolchain that facilitates transfer of dynamic graph parts from Python by either mapping them to control-flow primitives in Caffe2 or reimplementing them in C++ as custom operators.

III. Resource Implications of Machine Learning

Given that the two stages of machine learning - training and inference - have distinct resource requirements, frequency, and duration, we discuss the details and resource implications of these two distinct phases separately, and in turn.

A. Summary of Hardware Resources at Facebook

Facebook Infrastructure has a long history of producing efficient platforms for the major software services, including custom-designed servers, storage, and networking support for the resource requirements of each major workload [10]. We

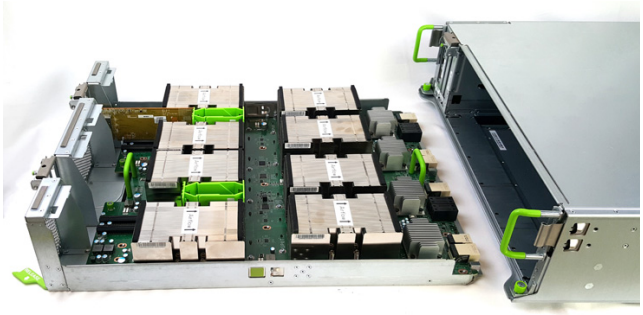

Fig. 3. The Big Basin GPU server design includes 8 GPUs in a 3U chassis.

currently support roughly eight major compute and storage rack types that map to the same number of major services. New services tend to get mapped to existing rack types until they rise to the level of warranting their own design. These major rack types were designed to meet the resource requirements of major services. For example, Figure 2 shows a 2U chassis that accommodates three compute sleds supporting two alternative server types. One sled option is a single socket CPU server (1xCPU) supported for the web tier, which is a throughput-oriented stateless workload, and therefore can be well served by a more power-efficient CPU (Broadwell-D processor) with a relatively small amount of DRAM (32GB) and minimal on-board disk or flash storage [11]. Another sled option is a larger dual socket CPU server (2x high power Broadwell-EP or Skylake SP CPU) with large amounts of DRAM that is used for compute- and memory-intensive services.

To accelerate our progress as we train larger and deeper neural networks, we also created Big Basin, our latest-generation GPU server, in 2017, shown in Figure 3. The initial Big Basin design included eight NVIDIA Tesla P100 GPU accelerators connected using NVIDIA NVLink to form an eight-GPU hybrid cube mesh [12]. The design has since been upgraded to support V100 GPUs as well.

Big Basin is the successor to our earlier Big Sur GPU server, which was the first widely deployed, high-performance AI compute platform in our data centers, designed to support NVIDIA M40 GPUs, which was developed in 2015 and released via the Open Compute Project. Compared with Big Sur, the newer V100 Big Basin platform enables much better gains on performance per watt, benefiting from single-precision floating-point arithmetic per GPU increasing from 7 teraflops to 15.7 teraflops, and high-bandwidth memory (HBM2) providing 900 GB/s bandwidth (3.1x of Big Sur). Half-precision was also doubled with this new architecture to further improve throughput. Big Basin can train models that are 30 percent larger because of the availability of greater arithmetic throughput and a memory increase from 12 GB to 16 GB. Distributed training is also enhanced with the high-bandwidth NVLink inter-GPU communication. In tests with the ResNet-50 image classification model, we were able to reach almost 300 percent improvement in throughput compared to Big Sur, allowing us to experiment faster and work with more complex models than before.

Each of these compute server designs, as well as several storage platforms, has been publicly released through the Open Compute Project. Meanwhile, internally, we are always refreshing our hardware designs and thoroughly evaluating all promising alternatives and new technologies.

B. Resource Implications of Offline Training

Today, different products leverage different compute resources to perform their offline training step. Some products, such as Lumos, do all of their training on GPUs. Other products, such as Sigma, do all of their training on dual-socket high-memory CPU compute servers. Finally, products like Facer have a two-stage training process, where they train a general face detection and recognition model infrequently (many months) on GPUs, and then train user-specific models much more regularly on thousands of 1xCPU servers.

In this section, we present high-level details about various services with respect to machine learning training platforms, frequency, and duration, summarized in Table II. We also discuss the data set trends and implications for our compute, memory, storage, and network infrastructure.

Compute Type and Locality. Offline training may be performed on CPUs and/or GPUs, depending on the service. While in most cases, training on GPUs tends to outperform training on CPUs, the abundance of readily-available CPU capacity makes it a useful platform. This is especially true during the off-peak portions of the diurnal cycle where CPU resources would otherwise sit idle, as we later show in Figure 4. Below we identify which services currently train their models on each compute resource:

- Training on GPUs: Lumos, Speech Recognition, Language Translation
- Training on CPUs: News Feed, Sigma
- Training on Both: Facer (general model trained on GPUs every few years as the model is stable; user-specific model trained on 1xCPUs in response to a threshold of new image data), Search (leverages multiple independent search verticals, and applies a predictive classifier to launch the most appropriate verticals).

Currently, the primary use case of GPU machines is offline training, rather than serving real-time data to users. This flows logically given that most GPU architectures are optimized for throughput over latency. Meanwhile, the training process does heavily leverage data from large production stores, therefore for performance and bandwidth reasons, the GPUs need to be in production near the data accessed. The data leveraged by each model is growing quickly, so this locality to the data source (many of which are regional) is becoming more important over time.

Memory, Storage, and Network. From a memory capacity standpoint, both CPU platforms as well as the GPU platform provide sufficient capacity for training. This was even true for applications like Facer, which trains user-specific SVM models

Service	Resource	Training Frequency	Training Duration
News Feed	Dual-Socket CPUs	Daily	Many Hours
Facer	GPUs + Single-Socket CPUs	Every N Photos	Few Seconds
Lumos	GPUs	Multi-Monthly	Many Hours
Search	Vertical Dependent	Hourly	Few Hours
Language Translation	GPUs	Weekly	Days
Sigma	Dual-Socket CPUs	Sub-Daily	Few Hours
Speech Recognition	GPUs	Weekly	Many Hours

TABLE II
FREQUENCY, DURATION, AND RESOURCES USED BY OFFLINE TRAINING FOR VARIOUS WORKLOADS.

on our 1xCPU servers with 32 GB RAM. Leveraging efficient platforms and spare capacity whenever possible results in significant overall efficiency wins.

Machine learning systems rely on training against example data. Meanwhile, Facebook leverages a large fraction of all stored data in machine learning pipelines. This creates regional preferences for the placement of compute resources near data stores. Over time, most services indicate a trend toward leveraging increased amounts of user data, so this will result in an increased dependence on other Facebook services, and increased network bandwidth for data access as well. So, significant local/nearby storage is required to allow off-line bulk data transfers from distant regions to avoid stalling the training pipelines waiting for additional example data. This has implications on training machine region placement to avoid having training clusters apply excessive pressure on nearby storage resources.

The amount of training data leveraged during offline training varies widely by service. In nearly all cases, the training data sets are trending toward continued and sometimes dramatic growth. For instance, some services leverage millions of rows of data before the ROI degrades, while others leverage billions of rows (100s of TB) and are bounded only by resources.

Scaling Considerations and Distributed Training. Training a neural network involves optimization of parameter weights through Stochastic Gradient Descent (SGD). This technique, used for fitting neural nets, involves iterative weight updates through assessments of small subsets (i.e., a "batch" or "mini-batch") of labeled examples. Data parallelism involves spawning model replicas (parallel instances) to process multiple batches in parallel.

Traditionally, models were trained on a single machine. Larger or deeper models can be more expressive and provide higher accuracy, although training these models may require processing more examples. Within a single machine, training performance can be maximized by increasing model replicas and employing data parallelism across GPUs. Given that the data needed for training is increasing over time, hardware limitations can result in an unacceptable increase in overall training latency and time to convergence. Distributed training is one solution for overcoming these hardware limitations and reducing latency. This is an active research area not only at Facebook, but also in the general AI research community.

A common assumption is that for data parallelism across machines, a specialized interconnect is required. However, during our work on distributed training, we have found Ethernet-based networking to be sufficient, providing near-linear scaling capability. The ability to scale close to linearly is closely related to both model size and network bandwidth. If networking bandwidth is too low such that performing parameter synchronization takes more time than performing gradient computations, the benefits of data parallelism across machines diminishes. With its 50G Ethernet NIC, our Big Basin server has allowed us to scale out training of vision models without inter-machine synchronization being a bottleneck.

In all cases, the updates need to be shared with the other replicas using techniques that provide trade-offs on synchronization (every replica sees the same state), consistency (every replica generates correct updates), and performance (which scales sub-linearly), which may impact training quality. For example, the Translation service cannot currently train on large mini-batches without degrading model quality. As a counter-example, using certain hyperparameter settings, we can train our image classification models to very large mini-batches, scaling to 256+ GPUs [13]. For one of our larger workloads, data parallelism has been demonstrated to provide 4x the throughput using 5x the machine count (e.g., for a family of models that trains over 4 days, a pool of machines training 100 different models could now train 20 models per day, so training throughput drops by 20%, but the wait time for potential engineering advancement improves from four days to one day).

If models become exceptionally large, model parallelism training can be employed, where the model layers are grouped and distributed to optimize for throughput with activations pipelined between machines. The optimizations might be associated with network bandwidth or latency, or balancing internal machine limitations. This increases end-to-end latency of the model, so the raw performance gain in step time is often associated with a degradation in step quality. This may further degrade model accuracy per step. The combined degradation of step accuracy may lead to an optimal amount of parallel processing.

In many cases, during inference, the DNN models themselves are designed to be run on a single machine, as partitioning the model graph among the machines can result in large amount of communication. But major services are consistently weighing the cost/benefits of scaling their models. These considerations may dictate changes in network capacity needs.

Services	Relative Capacity	Compute	Memory
News Feed	100X	Dual-Socket CPU	High
Facer	10X	Single-Socket CPU	Low
Lumos	10X	Single-Socket CPU	Low
Search	10X	Dual-Socket CPU	High
Language Translation	1X	Dual-Socket CPU	High
Sigma	1X	Dual-Socket CPU	High
Speech Recognition	1X	Dual-Socket CPU	High

TABLE III
RESOURCE REQUIREMENTS OF ONLINE INFERENCE WORKLOADS.

C. Resource Implications of Online Inference

After offline training, the online inference step involves loading a model onto a machine and running that model with real-time inputs to produce real-time results for web traffic. Table III summarizes the relative compute capacity and type of compute used for several services.

To provide an example of an online inference model in operation, we will walk through the Ads ranking model. The Ads ranking model screens tens of thousands of ads down to display the top 1 to 5 ads in News Feed. This is accomplished through progressively sophisticated passes of ranking calculations performed against successively smaller subsets of the ads. Each pass consists of a MLP-like model that contains sparse embedding layers, with each pass narrowing down the ad candidates count. The sparse embedding layer is memory intensive, so for later passes where the models have higher number of parameters, it is run on a separate server from the MLP passes.

From a compute standpoint, the vast majority of online inference runs on the abundant 1xCPU (single-socket) or 2xCPU (dual-socket) production machines. Since 1xCPU machines are significantly more power and cost-efficient for Facebook, there is an emphasis toward migrating models from 2xCPU servers to 1xCPU servers whenever possible. With the rise of high-performance mobile hardware, it is even possible to run some models directly on the user's mobile device to improve latency and reduce communication cost. However, some compute and memory intensive services still require 2xCPU servers for the best performance.

Finally, various products have varying latency requirements for the results of online inference. In some cases, the resulting data can be considered "nice to have" or can return after an initial quick estimate is returned to the user. For instance, it may be acceptable in some cases to initially classify content as acceptable, while more complex models are run that can later override the initial classification. Meanwhile, models like Ads and News Feed have firm SLAs for delivering the proper content to users. These SLAs are driving the model complexity and dependencies, and thus more advanced compute power can result in more advanced models.

IV. MACHINE LEARNING AT DATACENTER SCALE

Aside from resource requirements, there are major considerations when deploying machine learning at the datacenter scale including the significant data requirements as well as reliability in the face of natural disasters.

A. Getting Data to the Models

For many machine learning models at Facebook, success is predicated on the availability of extensive, high-quality data. The ability to rapidly process and feed these data to the training machines is important for ensuring that we have fast and efficient offline training.

For sophisticated ML applications such as Ads and Feed Ranking, the amount of data to ingest for each training task is more than hundreds of terabytes. Moreover, complex pre-processing logic is applied to ensure that data is cleaned and normalized to allow efficient transfer and easy learning. These impose very high resource requirement especially on storage, network, and CPU.

As a general solution, we want to decouple the data workload from the training workload. These two workloads have very different characteristics. The data workload is very complex, ad-hoc, business dependent, and changing fast. The training workload on the other hand is usually regular (e.g. GEMM), stable (there are relatively few core operations), highly optimized, and much prefers a "clean" environment (e.g., exclusive cache usage and minimal thread contention). To optimize for both, we physically isolate the different workloads to different machines. The data processing machines, aka "readers", read the data from storage, process and condense them, and then send to the training machines aka "trainers". The trainers, on the other hand, solely focus on executing the training options rapidly and efficiently. Both readers and trainers can be distributed to provide great flexibility and scalability. We also optimize the machine configurations for different workloads.

Another important optimization metric is network usage. The data traffic generated by training can be significant and sometimes bursty. If not handled intelligently, this can easily saturate network devices and even disrupt other services. To address these concerns, we employ optimization in compression, scheduling algorithms, data/compute placement, etc.

B. Leveraging Scale

As a company serving users across the world, Facebook must maintain a large fleet of servers designed to handle the peak load at any given time. As seen in Figure 4, due to variations in user activity due to diurnal load and peaks during spe-

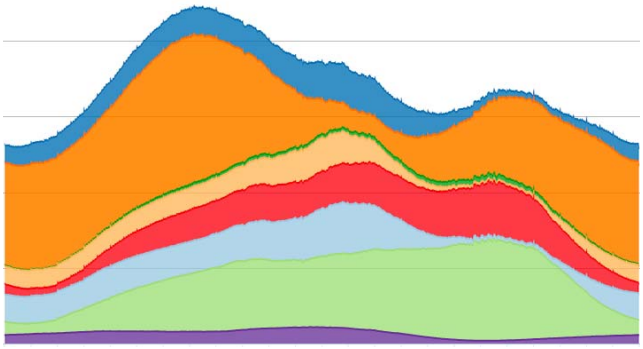

Fig. 4. Diurnal load across Facebook's fleet over a 24-hour period on 19 September 2017.

cial events (e.g. regional holidays), a large pool of servers are often idle at certain periods in time. This effectively provides an enormous pool of compute resources available during the off-peak hours. A major ongoing effort explores opportunities to take advantage of these heterogeneous resources that can be allocated to various tasks in an elastic manner. For machine learning applications, this provides a prime opportunity to take advantage of distributed training mechanisms that can scale to a large number of heterogeneous resources (e.g. different CPU and GPU platforms with differing RAM allocations). The sheer scale of the number of compute resources available during these low utilization periods leads to fundamentally different distributed training approaches, imposing a few challenges. The scheduler must first balance the load properly across heterogeneous hardware, so that hosts do not have to wait for each other for synchronization. The scheduler must also consider the network topology and synchronization cost when training spans multiple hosts. If not handled properly, the heavy intra- or inter-rack synchronization traffic could significantly deteriorate the training speed and quality. For example, in the 1xCPU design, the four 1xCPU hosts share a 50G NIC [11]. If all four hosts attempt to synchronize their gradients with other hosts at the same time, the shared NIC will soon become the bottleneck, resulting in dropped packets and timeouts. Therefore, a co-design between network topology and scheduler is needed to efficiently utilize the spare servers during off-peak hours. In addition, such algorithms must also have the capability to provide check-pointing to stop and restart training as loads change.

C. Disaster Recovery

The ability to seamlessly handle the loss of a portion of Facebook's global compute, storage, and network footprint has been a long-standing goal of Facebook Infrastructure [14]. Internally, our disaster recovery team regularly performs drills to identify and remedy the weakest links in our global infrastructure and software stacks. Disruptive actions include taking an entire data center offline with little to no notice in order to confirm that the loss of any of our global data centers results in minimal disruption to the business.

For both the training and inference portions of machine learning, the importance of disaster-readiness cannot be underestimated. While the importance of inference to drive several key projects is unsurprising, there is a potentially surprising dependency on frequent training before noticing a measurable degradation in several key products.

We discuss the importance of frequent ML training for three key products, and discuss the infrastructure support needed to accommodate that frequent training, and how this all relates to disaster recovery compliance.

What Happens If We Don't Train Our Models? We analyzed three key services that leverage ML training, to ascertain the impact of being unable to perform frequent updates to the models through training, including Ads, News Feed, and Community Integrity. Our goal was to understand the implications of losing the ability to train their models for one week, one month, and six months.

The first obvious impact was engineer efficiency, as machine learning progress is often tied to frequent experimentation cycles. While many models can be trained on CPUs, training on GPUs often enables notable performance improvement over CPUs for certain use cases. These speedups offer faster iteration times, and the ability to explore more ideas. Therefore, the loss of GPUs would result in a net productivity loss for these engineers.

Furthermore, we identified a substantial impact to Facebook products, particularly for products that rely heavily on frequent refreshes of their models. We summarize the problems that arise when these products use stale models.

Community Integrity: Creating a safe place for people to share and connect is the core of Facebook's mission; swiftly and accurately detecting offensive content is core to this mission. Our Community Integrity team heavily leverages machine learning techniques to detect offensive content in text, images, and videos. Offensive content detection is a specialized form of spam detection. Adversaries are constantly searching for new and innovative ways to bypass our identifiers in order to display objectionable content to our users. To defend against these efforts, we frequently train models to learn those new patterns. Each training iteration takes on the order of days to generate a refined model for objectionable image detection. We are continuing to push the boundaries to train models faster using distributed training techniques, but the inability to train entirely would result in degradation of content.

News Feed: Less surprising was our finding that products like News Feed have a heavy dependence on machine learning and frequent model training. Identifying the most relevant content for every user on every visit to our site results in a significant dependence on state-of-the art machine learning to properly find and rank that content. Unlike some other products, the learning side of Feed Ranking happens in two steps: an offline step to train the best model, which runs on both CPUs/GPUs, followed by continuous online training that currently runs on CPUs.

Fig. 5. Facebook global data center locations as of December 2017.

Stale News Feed models have a measurable impact on quality. The News Feed team continuously pushes the boundaries to innovate on their ranking models, and models themselves take on the order of hours to train. The loss of training compute for even one week can hinder the team's ability to explore new models and new parameters.

Ads: Least surprising is the importance of frequent training for the Ads Ranking models. Finding and displaying the very best ads involves a significant dependence on and innovation in machine learning. To underscore the importance of that dependence, we learned that the impact of leveraging a stale ML model is measured in hours. In other words, using a one-day-old model is measurably worse than using a one-hour-old model.

Overall, our investigation served to underscore the importance of machine learning training for many Facebook products and services. Disaster readiness of that large and growing workload should not be underestimated.

Infrastructure Support for Disaster Recovery Figure 5 shows the world-wide distribution of Facebook's datacenter infrastructure. If we focus on the availability of CPU resources used during training and inference, we have ample compute servers in nearly every region to accommodate the potential loss of our largest region. The importance of providing equal redundancy for GPU resources had initially been underestimated, however.

The initial workloads that leveraged GPUs for training were primarily computer vision applications, and the data required to train these models was globally replicated. When GPUs were new to Facebook Infrastructure, rolling them out in a single region seemed to be a smart option for manageability until the designs matured and we could build internal expertise on their service and maintenance requirements. These two factors led to the decision to physically isolate all production GPUs to one datacenter region.

However, several key changes occurred after that time. Due to the increased adoption of Deep Learning across multiple products, including ranking, recommendation, and content understanding, locality between the GPU compute and big data increased in importance. And complicating that need for compute-data colocation was a strategic pivot toward a mega-region approach for storage. The notion of a mega-region means that a small number of data center regions will house the bulk of Facebook's data. Incidentally, the region housing the entire GPU fleet did not reside in the storage mega-region.

Thus, aside from the importance of co-locating compute with data, it quickly became important to consider what might happen if we were to ever lose the region housing the GPUs entirely. And the outcome of that consideration drove the need to diversify the physical locations of the GPUs used for ML training.

V. Future Directions in Co-design: Hardware, Software, and Algorithms

As model complexity and dataset sizes grow, computational requirements of ML also increase. ML workloads exhibit a number of algorithmic and numerical properties which impact the hardware choices.

It is well known that convolution and medium size matrix-matrix multiplication are the key compute kernels of the forward and backward passes of deep learning. With larger batch sizes, each parameter weight is reused more often, so these kernels would exhibit improvements in arithmetic intensity (the number of compute operations per byte of accessed memory). Increasing arithmetic intensity generally improves the efficiency of the underlying hardware, so within the limits of latency, running with higher batch sizes is desirable. Compute bound ML workloads would benefit from wider SIMD units, specialized convolution or matrix multiplication engines, and specialized co-processors.

In some cases, small batch sizes per node are a requirement, both in real-time inference, when concurrent queries are low, and during training, when scaling to large numbers of nodes. Smaller batch sizes often result in lower arithmetic intensity (e.g., matrix-vector multiplication operations on fully-connected layers, which is inherently bandwidth-bound). This could potentially degrade the performance of several common use cases, where the full model does not fit into on-die SRAM or last-level cache. This could be mitigated through model compression, quantization, and high-bandwidth memory. Model compression can be achieved through sparsification and/or quantization [15]. Sparsification prunes connections during training, resulting in a smaller model. Quantization compresses the model using fixed-point integers or narrower floating-point formats instead of FP32 (single precision floating point) for weights and activations. Comparable accuracy has been demonstrated for several popular networks using 8 or 16 bits. There is also ongoing work to use 1 or 2 bits for weights [16], [17]. In addition to reducing the memory footprint, pruning and quantization can speed up the underlying hardware by reducing the bandwidth and also by allowing hardware architectures to have higher compute rates when operating with fixed point numbers, which is much more efficient than operating on FP32 values.

Reducing training time and expediting model delivery requires distributed training. As discussed in Section IV-B, distributed training requires a careful co-design of network topology and scheduling to efficiently utilize hardware and achieve good training speed and quality. The most widely-used

form of parallelism in distributed training is data parallelism, described in Section III-B, which requires synchronizing the gradient descent across all the nodes, either synchronously or asynchronously. Synchronous SGD requires an all-reduce operation. An interesting property of all-reduce, when performed using recursive doubling (and halving), is that bandwidth requirements decrease exponentially with the recursion levels. This encourages hierarchical system design where nodes at the bottom of the hierarchy form super-nodes with high connectivity (e.g., connected via high-bandwidth point to point connections, or high-radix switch); at the top of the hierarchy, super-nodes are connected via slower network (e.g., Ethernet). Alternately, asynchronous SGD (processing batches without waiting for other nodes) is harder and is typically done via a shared parameter server; nodes send their updates to a parameter server which aggregates and distributes them back to the nodes. To reduce staleness of updates and reduce pressure on parameter servers, a hybrid design could be beneficial. In such a design, asynchronous updates happen within super-nodes with high bandwidth and low latency connectivity between local nodes, while synchronous updates happen across super-nodes. Further increases to scalability require increasing the batch size without sacrificing convergence. This is an active area of algorithmic research both within and outside Facebook.

At Facebook, our mission is to is build high-performance, energy-efficient systems for machine learning that meet the demands of our abundant ML-based applications, described in Section II. We continuously evaluate and prototype novel hardware solutions, while simultaneously keeping an eye on the upcoming, near and longer-term algorithm changes, and their potential impact on system-level design.

VI. Conclusion

The increasing importance of machine learning-based workloads has implications that span all parts of the systems stack. In response, there has been a growing interest within the computer architecture community on how best to respond to the resulting challenges that have emerged. While prior efforts have revolved around efficiently handling the necessary compute for ML training and inference, the landscape changes when considering the additional challenges that arise when the solutions are considered at scale.

At Facebook, we discovered several key factors that emerge at scale and drive decisions in the design of our datacenter infrastructure: the importance of co-locating data with compute, the importance of handling a variety of ML workloads, not just computer vision, and the opportunities that arise from spare capacity from diurnal compute cycles. We considered each of these factors when designing end-to-end solutions that incorporate custom-designed, readily-available, open-source hardware, as well as an open-source software ecosystem that balances performance and usability. These solutions are what power the large-scale machine learning workloads that serve over 2.1 billion people today, and reflect the interdisciplinary efforts of experts in machine learning algorithm and system design.

References

[1] B. Reagen, R. Adolf, P. N. Whatmough, G. Wei, and D. M. Brooks, *Deep Learning for Computer Architects*, ser. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2017.

[2] J. Quiñonero Candela, "Powering Facebook experiences with AI," April 2016, https://fb.me/candela_2016.

[3] M. Kabiljo and A. Ilic, "Recommending items to more than a billion people," June 2015, https://fb.me/kabiljo_2015.

[4] M. Schroepfer, "Accelerating innovation and powering new experiences with AI," November 2016, https://fb.me/schroepfer_2016.

[5] X. He, J. Pan, O. Jun, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, and J. Quiñonero Candela, "Practical lessons from predicting clicks on ads at facebook," in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, ser. ADKDD'14. New York, NY, USA: ACM, 2014, pp. 5:1–5:9. [Online]. Available: http://doi.acm.org/10.1145/2648584.2648589

[6] J. M. Pino, A. Sidorov, and N. F. Ayan, "Transitioning entirely to neural machine translation," August 2017, https://fb.me/pino_2017.

[7] A. Ilic and O. Kuvshynov, "Evaluating boosted decision trees for billions of users," March 2017, https://fb.me/ilic_2017.

[8] J. Dunn, "Introducing FBLearner flow: Facebook's AI backbone," May 2016, https://fb.me/dunn_2016.

[9] J. Quiñonero Candela, "Facebook and Microsoft introduce new open ecosystem for interchangeable AI frameworks," September 2017, https://fb.me/candela_2017.

[10] A. G. Murillo, "The end-to-end refresh of our server hardware fleet," March 2017, https://fb.me/murillo_2017.

[11] V. Rao and E. Smith, "Facebook's new front-end server design delivers on performance without sucking up power," March 2016, https://fb.me/rao_2016.

[12] K. Lee, "Introducing Big Basin: Our next-generation AI hardware," March 2017, https://fb.me/lee_2017.

[13] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch SGD: Training ImageNet in 1 hour," *CoRR*, vol. abs/1706.02677, 2017. [Online]. Available: http://arxiv.org/abs/1706.02677

[14] J. Parikh, "Keynote address at the @Scale Conference," August 2016, https://fb.me/parikh_2016.

[15] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *International Conference on Learning Representations (ICLR)*, 2016.

[16] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," *CoRR*, vol. abs/1602.02830, 2016. [Online]. Available: http://arxiv.org/abs/1602.02830

[17] H. Alemdar, N. Caldwell, V. Leroy, A. Prost-Boucle, and F. Pétrot, "Ternary neural networks for resource-efficient AI applications," *CoRR*, vol. abs/1609.00222, 2016.