DjiNN and Tonic: DNN as a Service and Its Implications for Future Warehouse Scale Computers

Johann Hauswald Yiping Kang Michael A. Laurenzano Quan Chen Cheng Li
Trevor Mudge Ronald G. Dreslinski Jason Mars Lingjia Tang
Clarity Lab
University of Michigan - Ann Arbor, MI, USA
{jahausw, ypkang, mlaurenz, quanchen, elfchris, tnm, rdreslin, profmars, lingjia}@umich.edu

Abstract

As applications such as Apple Siri, Google Now, Microsoft Cortana, and Amazon Echo continue to gain traction, web-service companies are adopting large deep neural networks (DNN) for machine learning challenges such as image processing, speech recognition, natural language processing, among others. A number of open questions arise as to the design of a server platform specialized for DNN and how modern warehouse scale computers (WSCs) should be outfitted to provide DNN as a service for these applications.

*In this paper, we present **DjiNN**, an open infrastructure for DNN as a service in WSCs, and **Tonic Suite**, a suite of 7 end-to-end applications that span image, speech, and language processing. We use DjiNN to design a high throughput DNN system based on massive GPU server designs and provide insights as to the varying characteristics across applications. After studying the throughput, bandwidth, and power properties of DjiNN and Tonic Suite, we investigate several design points for future WSC architectures. We investigate the total cost of ownership implications of having a WSC with a disaggregated GPU pool versus a WSC composed of homogeneous integrated GPU servers. We improve DNN throughput by over 120× for all but one application (40× for Facial Recognition) on an NVIDIA K40 GPU. On a GPU server composed of 8 NVIDIA K40s, we achieve near-linear scaling (around 1000× throughput improvement) for 3 of the 7 applications. Through our analysis, we also find that GPU-enabled WSCs improve total cost of ownership over CPU-only designs by 4-20×, depending on the composition of the workload.*

ISCA'15, June 13-17, 2015, Portland, OR USA

http://dx.doi.org/10.1145/2749469.2749472

1. Introduction

Apple Siri, Google Now, Microsoft Cortana and Amazon Echo represent an important class of emerging internet services known as *intelligent personal assistants* (IPAs). This class of workloads allows users to use natural language and images to interact with digital assistants through a mobile device. IPAs process this information as input, reason about the user's query, and respond in natural language. Such systems are increasingly prevalent in today's devices, and this growth is expected to further increase with the introduction of IPA-equipped wearable devices. ABI Research predicts there will be 485 million annual wearable device shipments by 2018 [40], while the primary human-computer interface for these devices are speech and vision [4, 40].

In today's systems, the computation required to process IPA queries is housed almost entirely in cloud platforms [10]. Upon a voice (or vision) query to an IPA, only the voice recording (or image) is sent from the device to the cloud. That input is transcribed to text, analyzed for semantic meaning, searched against a database, and a response is sent back to the device. Significant machine learning problems must be solved across various query types in this application domain, including classifying images, recognizing faces, decoding speech, and analyzing text. These are challenging machine learning problems that require powerful algorithms to provide a satisfactory experience for users. One such machine learning algorithm, deep neural networks (DNNs), has recently gained popularity in solving this wide range of challenges.

Using a DNN model trained on a large corpus of data has been shown in the last few years to significantly outperform traditional machine learning techniques in a number of domains [26]. This has made DNNs particularly compelling in industry due to the large and rapidly increasing quantities of available data. Numerous internet service companies (Apple, Google, Microsoft, Facebook) have been reported to use DNN as their core machine learning algorithm for a wide range of applications [3, 5, 7]. In addition, the number of products and services leveraging DNN continues to grow. For example, Google has reported that it already uses DNN across numerous product teams [5].

With the industry-wide convergence on DNN algorithms across many application domains and the observation that a sin-

gle DNN implementation can generalize across applications, webservice providers can employ an approach of providing deep learning as a general datacenter service to be used by disparate products. This approach is appealing to webservice companies because development overhead is reduced by not having each team maintain their own DNN implementations, and having a centralized service provides a single point of optimization for DNN. An example of this trend is the Google Brain project [5], a deep-learning service that is used in over 30 different product teams across the company.

Considering the amount of computation dedicated to DNN inference at the query level and the opportunity to accelerate a centralized DNN service, there has been significant research interest in accelerator platforms for DNN in datacenters. Prior work has focused on custom ASICs for accelerating DNNs [14, 15, 32]. However, a number of open questions remain for the design of an accelerator server platform based on commodity GPU hardware. In particular, a number of challenges must be addressed to understand how future WSCs should be designed to leverage GPUs to accelerate DNN as a service in WSCs, including:

1. Quantifying and optimizing the throughput and latency advantage of the GPU by investigating the bottlenecks that inhibit full utilization of GPU resources and limit DNN throughput.
2. Investigating the scalability of GPU resources in a single server, and how this scalability differs across applications and neural network architectures.
3. Understanding the performance, energy, and cost tradeoffs between various server designs and WSC organizations such as an integrated homogeneous WSC design with GPU(s) in every server, as well as disaggregated systems.

However, the lack of a publicly available DNN webservice infrastructure and a representative suite of end-to-end applications that use this service is an obstacle in addressing these challenges. In this paper, we first present the design of **DjiNN**, an open general DNN service that supports a spectrum of emerging IPA applications, and **Tonic Suite**, a set of 7 end-to-end applications built on the DjiNN service that span the domains of image processing, speech recognition, and natural language processing. Using DjiNN and Tonic, we investigate and design GPU servers for an online DNN service in WSCs. We examine strategies to maximize system throughput for the DNN service while managing query latency. We also evaluate system architecture design questions including: 1) the performance scalability with an increased number of GPUs; 2) the memory, PCIe and network bandwidth requirements; and 3) total cost of ownership tradeoffs for integrated homogeneous servers versus disaggregated servers. The detailed contributions of this paper are as follows:

- **DjiNN, a DNN service infrastructure –** This paper presents the complete design and implementation of DjiNN, a general DNN service infrastructure for an open "Brain" [5] that supports a spectrum of applications and neural network architectures (Section 3.1).
- **Tonic Suite, a set of DNN applications –** We introduce 7 end-to-end applications that use the DjiNN service. They are representative of emerging workloads spanning the areas of computer vision, speech recognition, and language processing systems (Section 3.2).
- **GPU accelerator platform design –** We identify performance bottlenecks in the DNN service and evaluate strategies to mitigate them, achieving high throughput and GPU scalability without diminishing query latency on a GPU accelerator platform. With our optimizations, we observe a 120× throughput improvement using a single GPU compared to a CPU for most applications. Scaling to 8 GPUs provides additional throughput benefit for the majority of Tonic Suite applications (Sections 4 and 5).
- **WSC designs for a DNN service –** We evaluate various configurations including the number of GPUs, PCIe and network configurations, as well as disaggregated and integrated server design options. We identify the cost-efficient server design and system architecture that achieves maximal throughput and maximal throughput per dollar while satisfying the latency constraints based on the above investigations (Section 6).

Our optimizations are able to improve the throughput of DNN inference by over 120× for all but one Tonic Suite application (40× for facial recognition) on an NVIDIA K40 GPU over an Intel Xeon core. On a GPU server composed of 8 NVIDIA K40s, we achieve near-linear scaling (close to 1000× throughput improvement) for 3 of the 7 applications. We identify natural language processing workloads as being bandwidth constrained, leaving the GPUs starved for data and operating below their full potential. We address this bandwidth bottleneck by leveraging improved network and interconnect technologies, showing performance improvements of up to 4.5× over bandwidth-constrained designs. Through our TCO analysis, we find that GPU-enabled WSCs improve TCO over CPU-only designs by 4-20×, depending on the composition of the workload.

2. Background

A neural network is a directed graph of *neurons*, where each neuron is a processing element that applies a function to its input(s) to generate an output. The structure of the network is defined by a set of connections between different groups of neurons that perform the same function, known as the *layers* of the network. As illustrated in Figure 1, a layer can be of type input, hidden, or output. A neural network can have multiple hidden layers, where the number of such layers defines the *depth* of the network. Common to all neural networks is a classifier layer that produces the final output(s) of the network. This layer has as many outputs as there are classes to predict by the network.

Deep Neural Network (DNN) A DNN is a neural network

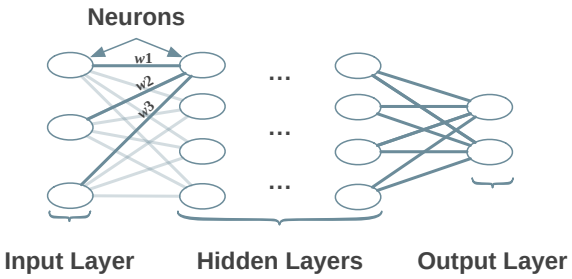


Figure 1: Deep neural network (DNN) architecture

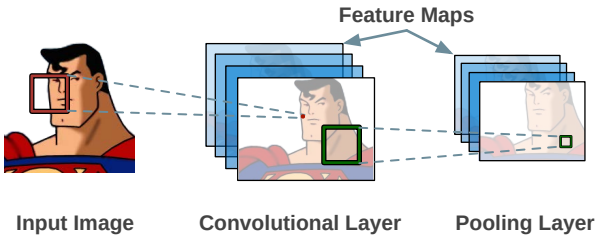


Figure 2: Convolutional neural network (CNN) architecture

with many hidden layers. Typically, each neuron is exhaustively connected to the neurons of the subsequent layer, in a configuration also known as a fully connected network. Each neuron computes a weighted sum of its inputs to form an output that is sent to the next layer. The weights applied to the inputs are learned during training and stored in a pretrained model describing the entire network. The structure of a DNN is depicted in Figure 1, where the weights ($w1$, $w2$, $w3$) are applied to the neuron's inputs to produce the output; this process is analogous for all the network's neurons.

Convolutional Neural Network (CNN) CNNs, a special case of DNNs, have a similar structure to DNNs except they are specialized for image-related tasks. Two important types of layers in CNNs include convolutional and pooling layers, used to extract features from input images. In these layers, each neuron is mapped to a region of the image to which the neuron applies a convolution or pooling operation. Because of this segmentation into regions, the network is not fully connected. These are also called sparsely connected networks. In convolutional layers, the learned weights are *kernels* that are convolved with the image to extract features. Figure 2 shows the kernel (red box) applied to the image generating a feature map. At each layer, there are multiple learned kernels each producing a distinct feature map (shaded feature maps in the figure). The pooling layers downsample each feature map to retain only "interesting" features (green boxes). This convolution-and-downsample process is repeated multiple times in a CNN to produce high quality features describing the input image. These features are used in the fully connected classifier layer to predict the content of the image.

3. DjiNN and Tonic

To investigate the design of a server platform specialized for DNN and how modern warehouse scale computers (WSCs) should be outfitted to provide DNN as a service, we design **DjiNN** and **Tonic Suite**. DjiNN is a centralized DNN service infrastructure that supports a diverse set of DNN-based applications in WSCs. Tonic Suite is a suite of 7 DNN-based applications from a wide range of domains including image classification, facial recognition, speech recognition and natural language processing. We extract the underlying DNN computation from each individual Tonic application. We create the generalized and configurable DjiNN service with a common interface to process the DNN computation for each application. In this paper, we focus on using DjiNN to process inference (forward pass) queries for these applications.

Figure 3 presents an overview of our system. Tonic Suite applications, representing inputs from mobile devices, makes requests to the DjiNN Service. DjiNN houses the trained DNN network architecture and configuration in-memory for each Tonic Suite application. To process each application's requests, DjiNN executes the DNN inference pass, which generates a prediction using the pre-processed input from the application, and returns the prediction result to the application. The design objectives and architecture of DjiNN, and Tonic Suite are described in Sections 3.1 and Sections 3.2, respectively.

3.1. DjiNN Service

The goal of the DjiNN service is to provide a unified service that executes the DNN portion of the Tonic Suite applications. With this goal in sight, we target the following objectives:

- **Decoupled Architecture –** DjiNN needs to be a standalone service accepting and processing external requests. We design the DjiNN service to accept requests using a custom socket protocol over TCP/IP. For DNN computation, we use Caffe [27], an open-source actively developed DNN library widely used in both academia and industry. For each incoming request, DjiNN spawns a worker thread, executes the DNN computation, and sends the prediction back to the application.
- **Diverse Applications –** A general DNN service must be capable of processing requests from a wide range of applications. To accomplish this, we rely on Caffe's general framework that supports various types of neural network layers. This enables flexible neural network configurations using Caffe. We extend Caffe to support DNN architectures from various applications representative of emerging WSC workloads including image processing, speech recognition, and natural language processing. Figure 3 shows our design, where DjiNN receives image, speech, and text based requests. DjiNN currently supports 7 DNN based Tonic applications. Supporting more applications simply requires providing DjiNN a pretrained neural network model.

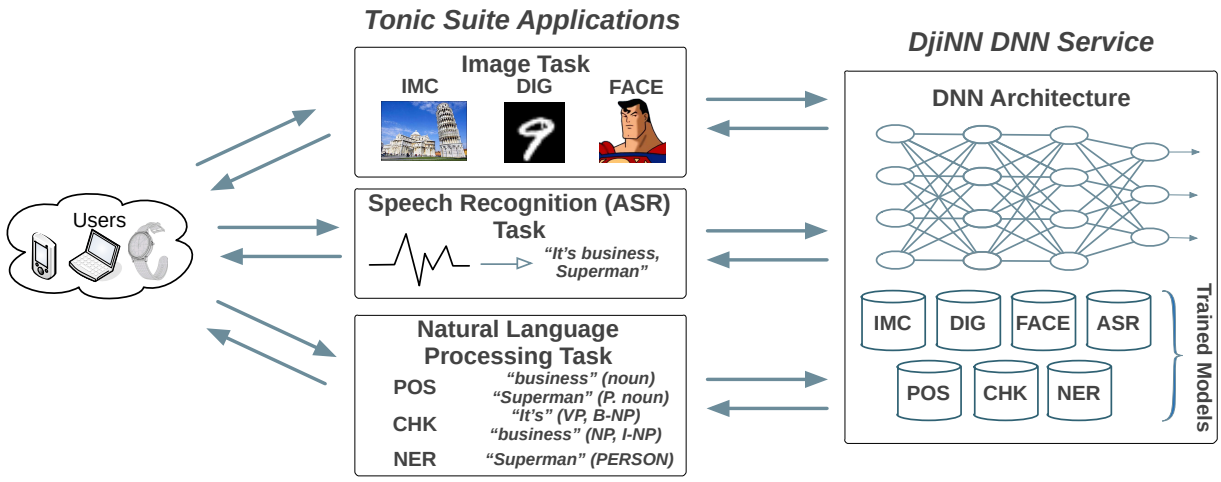


Figure 3: DjiNN architecture

- **Request Processing –** DjiNN must be able to process multiple incoming requests with limited overhead. At initialization, DjiNN loads the pre-trained model associated with each application into memory, giving all worker threads read-only access to this data. Consequently, incoming requests using the same model are accepted without needing to load their own copy of the model into memory.

The latest implementation of DjiNN is available online [2].

3.2. Tonic Suite

The DNNs used in Tonic Suite are based on recently published neural networks that achieve state-of-the-art accuracy in their target domains, which are summarized in Table 1. The suite of applications bundled with the neural network configurations, the trained models, and the server infrastructure to run the end-to-end applications have been released [2].

3.2.1. Image Task Tonic Suite's image tasks encompass three applications: image classification, digit recognition, and facial recognition. The image tasks do not have pre or postprocessing steps; the service sends the most likely prediction about the image back to the application. Each of the three image applications is described below.

Image Classification (IMC) Image classification sends an image to the DjiNN service and a prediction of what the image contains is sent to the application. This prediction is made by a model trained on 1.4M images from ImageNet [20], which can predict 1000 unique classes. We use AlexNet, a neural network architecture developed by Krizhevsky et al. [28], which achieves very high accuracy and outperforms other methods in large scale image classification competitions [41].

Digit Recognition (DIG) Digit recognition sends an image of a hand-written digit to the service and a prediction of the most likely digit (between 0-9) is returned to the application. The network architecture is based on MNIST [31], a widely used neural network for this task that achieves over 98% accuracy. A sample image is included in Figure 3.

Facial Recognition (FACE) The facial recognition application predicts the identity of faces using the DjiNN webservice. Facebook recently published DeepFace [43], a facial recognition network that achieves near human-accuracy in recognizing faces. We replicate this network into Tonic Suite and train it on a publicly available dataset of celebrity faces from PubFig83+LFW [13]. Using this dataset, DjiNN service classifies the input from 83 candidate celebrity faces.

3.2.2. Automatic Speech Recognition (ASR) Task In Tonic Suite, we include a DNN based speech-to-text decoder adapted from Kaldi [37], a state-of-the-art speech recognition toolbox actively developed by researchers from Microsoft and academia. Kaldi's speech processing techniques have been demonstrated to achieve very low word error rates (WER) on standard decoding benchmarks. The speech recognition application requires preprocessing to generate feature vectors describing the speech input that are sent to the DjiNN webservice. The service returns predictions for each feature vector that are postprocessed to find the most likely sequence of text to produce the final result.

3.2.3. Natural Language Processing (NLP) Task Included in Tonic Suite are NLP tasks designed to glean semantic information from input text. These tasks include part-of-speech (POS) tagging, word chunking (CHK), and name entity recognition (NER). For these applications, the text is preprocessed into word vector representations before being sent to DjiNN. After receiving the word predictions from the DNN service, the postprocessing step searches for the most likely sequence of tagged words. Our networks are based on Senna [19], a natural language processing toolbox developed by NEC Labs. The pretrained models are trained on Wikipedia for over 2 months and achieve over 89% accuracy for these applications.

Part-of-Speech Tagging (POS) Part-of-speech tagging assigns each word with a part of speech, for example if it is a noun or a verb.

Table 1: Tonic Suite neural network architectures

Type	Application	Network	Network Type	Layers	Parameters
Image Service	Image Classification (IMC)	AlexNet [20]	CNN	22	60M
	Digit Recognition (DIG)	MNIST [31]	CNN	7	60K
	Facial Recognition (FACE)	DeepFace [43]	CNN	7	120M
Speech Service	Automatic Speech Recognition (ASR)	Kaldi [37]	DNN	13	30M
NLP Service	Part-of-Speech Tagging (POS)	SENNA [19]	DNN	3	180K
	Chunking (CHK)	SENNA [19]	DNN	3	180K
	Name Entity Recognition (NER)	SENNA [19]	DNN	3	180K

Word Chunking (CHK) Word chunking tags each segment of a sentence as a noun or verb phrase where each word is labeled as a begin-chunk (B-NP) or an inside-chunk (I-NP). First, this application internally makes a POS service request, updates the tags for its input, and then makes it own DNN service request.

Name Entity Recognition (NER) Name entity recognition labels each word in the sentence with a category, for example whether it is a location or a person.

4. Identifying Bottlenecks for a DNN Service

In this section, we present a real-system analysis of the DNN service. We evaluate our baseline DNN service performance on a state-of-the-art GPU. We compare the GPU performance with the performance achieved on an Intel Xeon processor. We then conduct a performance analysis to identify bottlenecks to guide further throughput optimizations in the following sections. The configuration of the experimental platform is summarized in Table 2. We use 1 GPU for all the experiments in this section.

Table 2: Platform Specifications

Hardware	Specifications	Quantity
System	4U Intel Dual CPU Chassis, 8× PCIe 3.0 × 16 slots	1
CPU	Intel Xeon E5-2620 V2, 6C, 2.10 GHz	2
HDD	1TB 2.5" HDD	1
Memory	16GB DDR3 1866 MHz ECC/Server Memory	16
GPU	NVIDIA Tesla K40 M-Class 12 GB PCIe	8

DNN vs. non-DNN Components We first profile each DNN application on the Intel Xeon to characterize the amount of computation the back-end DNN service constitutes for each application. Figure 4 presents the average execution cycle breakdown for each application between its DNN portion and the rest of the computation (made up of query pre- and post-processing). For IMC, DIG, and FACE, the input images are directly fed into the DNN. Consequently, almost all of the cycles for the image services are spent on DNN computation. ASR requires substantial pre- and postprocessing to translate a voice recording into the final text. Nevertheless, the DNN service still consumes almost half of the execution cycles for ASR. For the NLP tasks, which also have pre- and postprocessing, more than two thirds of the total execution time is

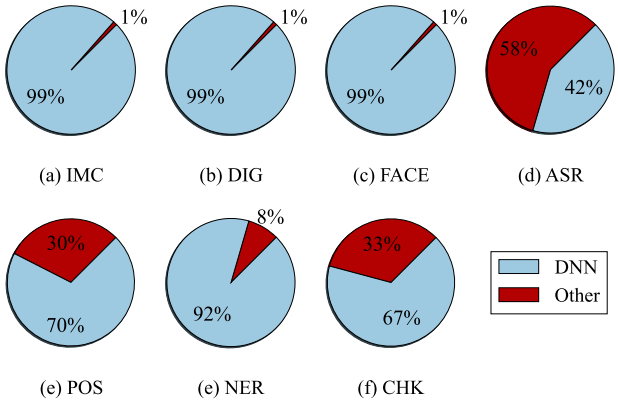


Figure 4: Cycle breakdown for each DNN application

DNN computation. This result demonstrates that DNN computation consumes a high percentage of the total execution time for almost all applications, motivating the need to design a common efficient DNN service in datacenters.

GPU vs. CPU performance Figure 5 presents the throughput improvement achieved on the K40 GPU over a Xeon CPU core. In this experiment, we use the off-the-shelf CPU version of Caffe linked to ATLAS (Automatically Tuned Linear Algebra Software) [46], a highly optimized linear algebra library widely used in commercial applications [6], for the matrix multiplication computation required by the DNN inference. For the GPU baseline, we use Caffe's GPU implementations as well as NVIDIA's recently released cuDNN library [16], which optimizes a subset of Caffe's key layers.

As shown in the figure, networks with more than 30M parameters achieve above 20× improvement from computing large matrix multiplications on the GPU. For example, ASR achieves significant improvement, 120× speedup, over the CPU baseline. On the other hand, NLP applications, which achieve only around 7× improvement, have small networks and thus the size of the matrix multiplications in the neural network forward pass is relatively small. This limits the resulting improvement achieved by the GPU.

Performance Bottlenecks To guide our throughput optimizations, we profile each DNN service using the NVIDIA

Table 3: DjiNN service applications

Application	Input to service	Input data size (KB)	Output from service	Batch size
IMC	1 image	604	1 classification	16
DIG	100 images	307	100 classifications	16
FACE	1 image	271	1 classification	2
ASR	548 speech feature vectors	4594	548 probability vectors	2
POS	28 word sentence	38	28 probability vectors	64
CHK	28 word sentence	75	28 probability vectors	64
NER	28 word sentence	43	28 probability vectors	64

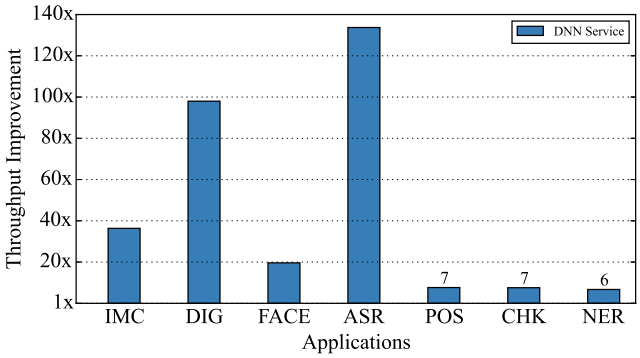


Figure 5: Throughput improvement achieved by a GPU over a single-thread CPU

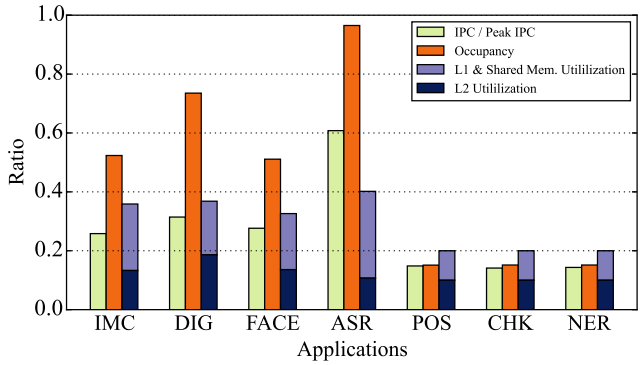


Figure 6: Performance bottleneck analysis

Profiler [1] and the NVIDIA Visual Profiler [9] to conduct performance analysis. Figure 6 presents the profiling information of several hardware performance counters for each application. The metrics are collected at the kernel level for each application, and are weighted by each kernel's execution time to calculate the average performance of the entire application. As shown in the figure, the ratio of the IPC to the peak IPC (IPC / Peak IPC) is relatively low for NLP tasks. All applications exhibit low memory bandwidth utilization (low L1, shared memory, and L2 bandwidth utilization) relative to the peak bandwidth utilization, indicating that the low IPC is not caused by a memory bandwidth limit. On the other hand, we observe that the IPC is roughly correlated to the GPU *occupancy*, the ratio of the number of active warps to the theoretical peak number of active warps. All three NLP tasks have under 20% occupancy, while ASR achieves above 90% occupancy. Low occupancy indicates that the GPU is not fully utilized for the NLP tasks. The kernels of these applications do not have enough thread blocks to hide the operation latency.

5. Designing a High Throughput System

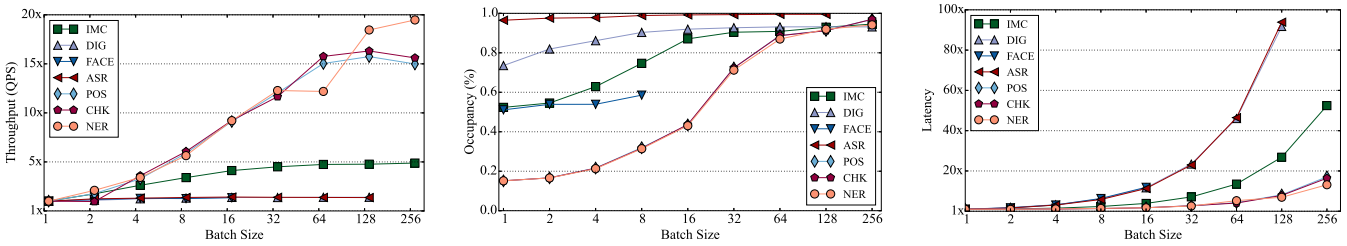
As observed in the previous section, the throughput improvement achieved by a GPU is substantially different across the DNN service component of all applications. This is due to the different neural network architectures of each application and the resulting varying degrees of GPU occupancy.

In this section, we investigate and design techniques aiming to achieve the maximal throughput for the DNN service on GPUs. We investigate three throughput improving techniques: 1) batching multiple queries into a combined query to increase occupancy on the GPU; 2) executing concurrent kernels to achieve better GPU resource efficiency; and 3) scaling the number of GPUs in a server. In addition to designing and evaluating techniques for throughput improvement, this investigation also allows us to gain insights on the throughput capability of state-of-the-art GPUs for the DNN service.

5.1. Batching DNN Inputs to Improve Throughput

We first investigate techniques to increase GPU occupancy and DNN service throughput by batching multiple DNN inputs into a single query. The type and size of the input and output data for the DNN service for each application are summarized in the first 4 columns of Table 3. To batch multiple inputs into a larger query, we increase the query input size by stacking multiple inputs into a larger matrix. Consequently, this increases the dimensions of the matrix multiplication executed in the DNN's forward pass on the GPU. The increased computation achieved by batching increases the occupancy on the GPU and the system throughput.

For each application, we vary the batch size and study the impact on the throughput and latency achieved by the GPU. Figure 7a presents how throughput is affected with varying input batch sizes. As shown in the figure, all applications exhibit a similar trend: the throughput first increases then plateaus as the batch size continues to increase. The throughput saturation point for each application is at a different batch size. In



(a) Throughput (b) GPU occupancy (c) Latency

Figure 7: Throughput, GPU occupancy and latency with varying batch sizes

addition, the throughput benefits from batching are different across applications. Speech recognition (ASR), which already achieves a considerable (120×) throughput improvement over a Xeon core (Figure 5) and near 100% GPU occupancy without batching (Figure 6), has a small throughput gain with larger batch sizes. On the other hand, some applications achieve very high throughput improvement from batching. For example, NLP tasks achieve over a 15× throughput improvement.

This throughput improvement is from improving the GPU occupancy by batching queries, as shown in Figure 7b. For NLP tasks, the baseline (batch size of 1) involves too little computation to fully occupy the GPU's resources, achieving only 20% occupancy. Increasing the batch size increases the amount of computation required. Consequently, the neural network computation uses more resources and the GPU occupancy significantly increases, achieving above 80% occupancy at a batch size of 64. We do not have data for FACE beyond a batch size of 8 in this figure because of the large size of the neural network and the high profiling overhead incurred.

Figure 7c presents the query latency for each DNN service. All inputs in a batched query are combined in an aggregated large matrix computation and thus share the same query latency across inputs within a batch. As shown in the figure, the query latency for each DNN service increases slightly at first. As the throughput plateaus, the latency starts to increase sharply. At this point, the GPU is saturated and the queuing delay starts to dominate the latency.

Based on Figures 7a and 7c, we identify the batch size for each application to achieve the high throughput while limiting query latency impact. These final values are summarized in the last column of Table 3. Overall, with the selected batch size, we achieve 15× throughput improvement for NLP tasks and 5× for IMC with limited latency increases in both cases.

5.2. Supporting Multiple DNN Services on a GPU

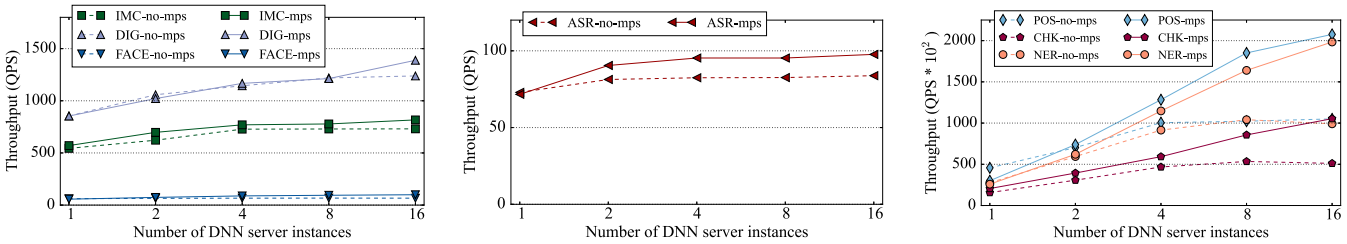
Next, we leverage NVIDIA's Multi-Process Service (MPS) [8], which allows kernels from different processes to execute concurrently on the GPU. Without MPS, each CUDA process allocates separate scheduling and storage resources on the GPU. Each time a different process executes, the GPU must context switch before resuming execution; all processes must timeshare the GPU. MPS allocates a shared pool of scheduling and storage resources for independent processes. As a result, the GPU can schedule multiple kernels concurrently from the same pool of resources without the need to context switch.

Figure 8 presents the throughput improvement as the number of concurrent DNN services on the GPU increases from 1 to 16 (the maximum number of simultaneous processes that MPS supports). Throughput is measured as queries per second (QPS). For MPS, DNN service instances can concurrently execute on the GPU. For comparison, we also present the non-MPS cases, where queries from multiple DNN service instances are time sharing the GPU. We use the batch sizes summarized in the last column of Table 3 for each application.

As shown in the figure, the achieved throughput increases as the number of DNN services on the GPU increases. With MPS, increasing concurrent kernels further improves throughput beyond what batching achieves (shown when the number of DNN instances is equal to 1). As previously described, without MPS, CUDA kernels launched by different processes timeshare the GPU's resources and have limited concurrency. With MPS, CUDA kernels launched by different processes can be executed concurrently. Because of this concurrency, the server queuing time for the next available time slice on the GPU is reduced and throughput increases. The throughput plateaus as we further increase the number of concurrent DNN services on the GPU. Overall, the DNN service achieves up to a 6× throughput improvement with concurrent service execution on the GPU.

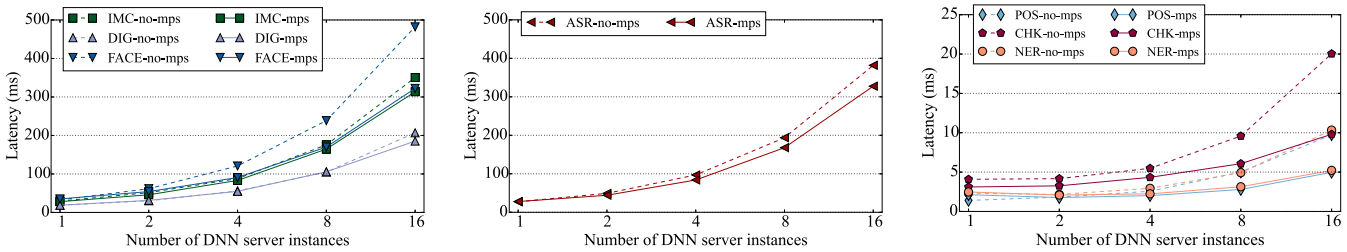
Figure 9 presents the query latency as the number of concurrent DNN services on the GPU increases. The query latency is relatively small when the number of concurrent DNN services is under 4 but increases sharply as the number of DNN services grows. MPS successfully limits the latency increase when compared to experiments without. As discussed earlier, MPS reduces the queuing and thus, as shown in Figure 9, reduces the query latency up to 3×, compared to the non-MPS configuration. Compared to the baseline configuration of executing a single service at a time on the GPU, the DNN service applications benefit from concurrent DNN services, which improves both throughput and latency.

Combining Figures 8 and 9, four MPS concurrent DNN



(a) Image service (b) ASR service (c) NLP service

Figure 8: Throughput as the number of DNN server instances per GPU increases (higher is better)



(a) Image service (b) ASR service (c) NLP service

Figure 9: Service latency as the number of DNN server instances per GPU increases (lower is better)

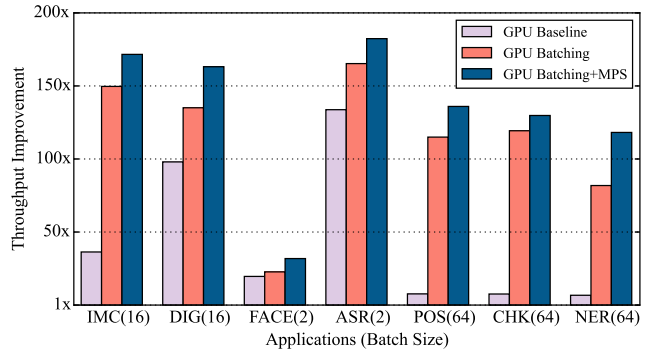


Figure 10: Throughput improvement using our optimizations achieved on a single GPU over single-thread CPU

servers on one GPU achieves high throughput gain with limited latency impact. For the DNN portion of most applications, more than 4 concurrent DNN services would have to trade high latency increase for low throughput improvement. Note the latency achieved using 4 concurrent DNN services on the GPU is smaller than the single query service time on the CPU.

Figure 10 summarizes our final throughput improvements on a K40 GPU after applying input batching (with the best batch size next to each application in the figure) and MPS. We achieve significant throughput benefits across the applications through these two optimizations. For NLP applications, batching and MPS together improve the GPU throughput gain from 7× to over 120×. The DNN service components achieve over 100× throughput improvement on the GPU for all but the FACE application, which achieves a 40× improvement.

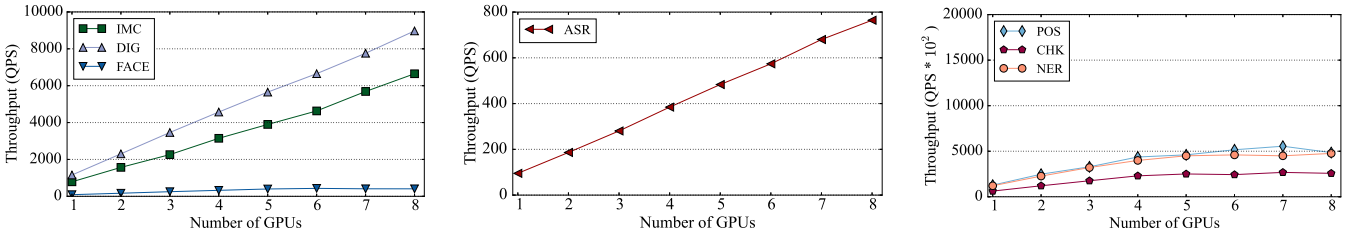
5.3. GPU Scalability

To further improve the system throughput, we scale the number of GPUs in a server and measure the system throughput of our optimized DNN service. The results are presented in Figure 11, with each application configured to use the optimal batch size and 4 MPS processes per GPU. As shown in the figure, both image services and the speech recognition service achieve near-linear scaling as the number of GPUs increases. There is no communication between GPUs and the PCIe bandwidth between the CPU and each GPU is sufficient for these services. However, we noticed that for NLP tasks, which have relatively small neural networks, the throughput plateaus as the number of GPUs reaches 4. For NLP tasks, each query requires less computation and the throughput (QPS) is several orders of magnitude higher than the other two services. As we will demonstrate in the next section, the throughput plateau is due to the PCIe bandwidth limitation.

In conclusion, the GPU scalability is dependent on the DNN characteristics for each application. For 3 out of 7 applications, by combining our optimizations and scaling the number of GPUs, we achieve 1000× throughput improvement on our 8 GPU system over a CPU core.

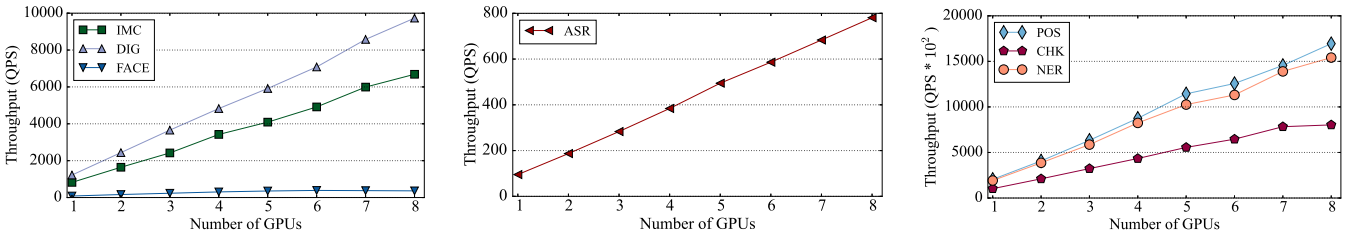
6. Implications for Future WSC Designs

Based on the insights gained from our throughput investigations in prior sections, we discuss the design of cost-efficient servers and the WSC systems necessary to provide a centralized DNN service for a wide range of applications. Section 6.1 characterizes the bandwidth requirements of the DNN ser-



(a) Image service (b) ASR service (c) NLP service

Figure 11: Throughput as the number of GPUs increases



(a) Image service (b) ASR service (c) NLP service

Figure 12: Throughput as the number of GPUs increases (no PCIe bandwidth limits)

vice, identifying bandwidth to the GPUs as the performance bottleneck for NLP applications. Section 6.2 then considers three WSC design strategies for housing the DNN service. Section 6.3 develops a TCO model to investigate the tradeoffs between the three designs, identifying the bandwidth constraint as a limiting factor for the TCO improvement of certain classes of DNN-based services. Finally, Section 6.4 describes and evaluates several network and interconnect architectures that can address the bandwidth limitation.

6.1. Bandwidth Requirements for Peak Throughput

To design the network configurations for the DNN servers in datacenters, we examine the bandwidth requirements of the DNN service. We first measure the peak throughput gain we can achieve without bandwidth constraints. To do so, we avoid communication by pinning the input of the DNN service to the GPU memory, which eliminates any data transfer (including transferring the final result). We then stress-test our system to measure the throughput of a system with no PCIe bandwidth limit. Repeating the experiment of scaling out the number of GPUs using this PCIe-bypassing setup, we measure the theoretical throughput improvement, presented in Figure 12. Without the PCIe bandwidth limit, all applications exhibit near-linear throughput improvement as we scale out the number of GPUs. This is expected because we are increasing the computational capability without any bandwidth contention.

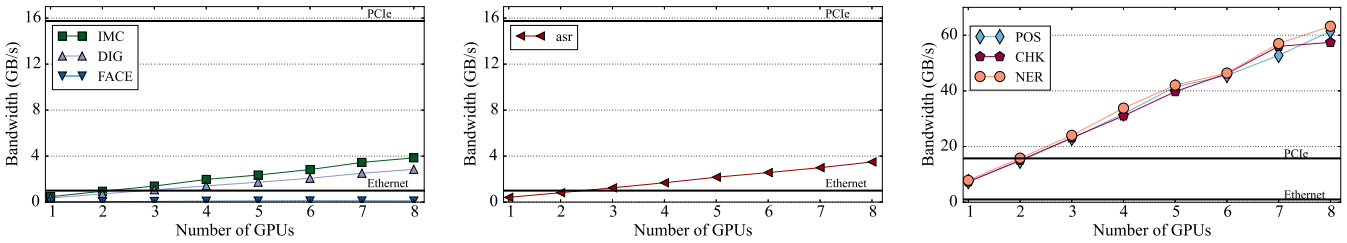
Based on the throughput improvement without the bandwidth constraint, we calculate the network bandwidth requirement for each application to achieve the maximum throughput. Figure 13 presents the network bandwidth requirements as the number of GPUs increases. As a point of reference, the peak bandwidth of several existing technologies, PCIe v3 and 10Gb ethernet (10GbE), are shown on the graph. For the computation-heavy tasks (IMC, DIG, FACE, ASR), our system is not bound by the PCIe bandwidth and the theoretical throughput can be achieved by a network with a bandwidth of at least 4GB/s (Figure 13). On the other hand, the light-computation tasks (NLP) require far higher bandwidth to sustain the near-linear throughput scaling. Later, we will use these bandwidth requirements as a guide to designing WSCs that are provisioned with sufficient bandwidth to overcome these bottlenecks.

6.2. WSC Architectures for a DNN Service

We next describe three design points for WSCs that can be used to house the DjiNN service as illustrated in Figure 14.

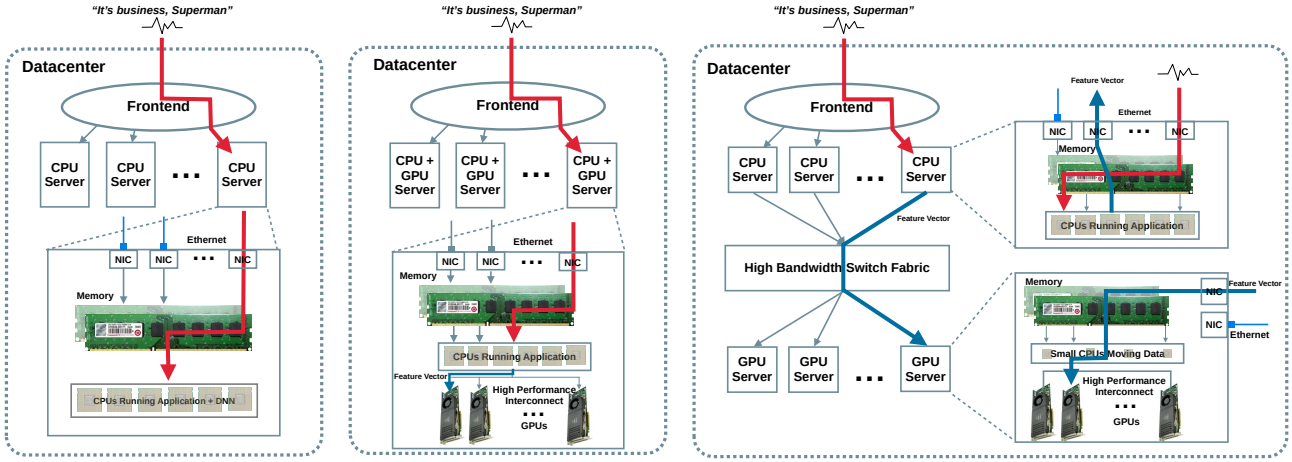
CPU Only Design As a baseline, we describe a CPU only datacenter that has no GPU capability. This design, presented in Figure 14a, includes homogeneous servers and contain beefy *CPU servers* that service all of the workloads in the datacenter, including non-DNN applications, DNN applications, and the DjiNN service. Each DNN query that hits the datacenter passes through a front-end (e.g., a load balancer) to one of the CPU servers. The path taken by each query is illustrated by a red arrow in Figure 14a. After the query hits the NIC, it is placed in memory for the CPU to process in full.

Integrated GPU Design Second, in Figure 14b we present the design of a datacenter with *Integrated GPUs*, containing a single server type of beefy CPUs and GPUs. In this design, the work of processing a query is handled within one server. However, unlike the *CPU Only* design, the work of processing the query is split between the CPU and the GPU. The path of



(a) Image service (b) ASR service (c) NLP service

Figure 13: Bandwidth requirement as the number of GPUs increases



(a) CPU only (b) Integrated GPU (c) Disaggregated GPU

Figure 14: Three WSC designs considered in this work. Red arrows show the path of a query through the WSC before preprocessing, while blue arrows show the path of preprocessed data moving from CPU to GPU

the query to the CPU is shown as a red arrow in Figure 14b and upon receiving the query, the CPU performs (if necessary) preprocessing on the query. The result of the preprocessing is passed to the GPU hosting the DjiNN service via the PCIe bus (blue arrow in Figure 14b), where the GPU processes the request. By offloading DNN inference to the GPU, this model offers substantially higher throughput over the *CPU Only* model. However, by joining the GPU and CPU within the same box, along with the overwhelming preference in WSC design for homogeneous server configurations [11], GPUs have to be apportioned to servers to accommodate the homogeneous case. In this study, we assume 12 GPUs per server based on the latest available number of PCIe $\times 16$ slots available today on commodity high performance motherboards.

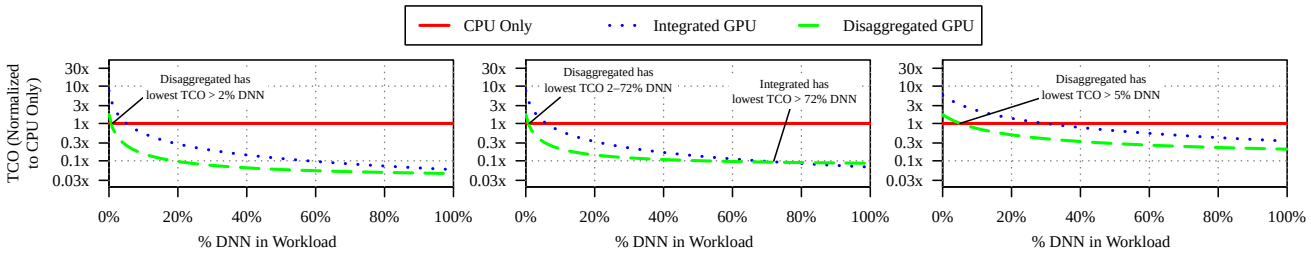
Disaggregated GPU Design To address the lack of flexibility of the integrated design, we consider a design that has *Disaggregated GPUs*. In this design, two types of servers coexist in the datacenter. Beefy CPU servers, resembling those described for the *CPU Only* model, handle all non-DNN workloads as well as pre- and postprocessing for DNN queries. In this design, illustrated in Figure 14c, each DNN-based query is first preprocessed on the CPU server, then the result is sent over the network to a GPU server hosting the DjiNN service. The GPU server is designed as a multicore system with wimpy CPU cores whose purpose is to pass query data to the GPUs.

The advantage of this approach over the *Integrated GPU* is it decouples the GPUs and beefy CPUs. Such a decoupling can be critical in WSCs where designers are motivated to use a limited number of server configurations to simplify hardware and software maintenance and insure against overspecializing servers in the presence of ever-evolving workloads. By decoupling CPUs and GPUs, the amount of GPU compute can be provisioned to handle the amount of GPU work available in the datacenter without adding GPUs to each server. However, a major challenge in this model is to provision sufficient bandwidth between the CPU and GPU servers. To provide the necessary bandwidth between the two, we aggregate 16 dedicated 10GbE NICs[1] on each device and employ a high performance network fabric to sustain sufficient bandwidth.

6.3. Total Cost of Ownership

To assess the tradeoffs between these three designs, we compute the Total Cost of Ownership (TCO) for WSCs constructed

[1] PCIe $\times 16$ supports up to 15.875GB/s. 10GbE can theoretically sustain 1.25GB/s, but may have significant protocol overheads. Assuming 80% of theoretical peak can be obtained, $16\times$ 1.25GB/s connection yields 16GB/s.



(a) MIXED workload **(b) IMAGE workload** **(c) NLP workload**

Figure 15: TCO of WSCs designed to house proportions of DNN and non-DNN webservices, normalized to CPU Only design (lower is better). Workload composition impacts the relative cost effectiveness of each design approach

Table 4: TCO parameters

Component	Cost Factor
300W GPU-capable server	$6864
High-end 240W GPU	$3314
75W wimpy server	$1716
Networking equipment	$750/10GbE NIC
WSC capital expenditures	$10/Watt
Operational expenditures	$0.04/Watt/month
Power Usage Efficiency (PUE)	1.1
Electricity	$0.067 per kWh
Interest rate on capital expenditures	8%
Server lifetime	3 years
Loan amortization period	3 years
Server maintenance/operations	5%/month

Table 5: DNN service workloads

Type	Description
MIXED	Mix (IMC, DIG, FACE, ASR, POS, CHK, NER)
IMAGE	Image processing (IMC, DIG, FACE)
NLP	Natural language processing (POS, CHK, NER)

to house DNN-based webservices using a methodology inspired by Barroso et al. [11]. Our methodology for computing TCO includes upfront hardware capital expenditures (e.g., purchasing servers, CPUs, memory, GPUs, networking equipment, facilities, etc.), operating costs (operations, maintenance and power), as well as financing costs. We do not explicitly model the GPU and CPU failure rate differences. We use the assumptions for cost factors as summarized in Table 4. We measure power on our GPU-enabled system to supply power draw estimates. In characterizing the price of the servers and GPUs, we use competitive market prices for the components at the time of this writing. For the GPU-capable server and GPU parts, the configurations we price are reflective of the high-end server used throughout this paper. To characterize the costs of networks in our approach, we assume 500 server leaf nodes connected to a hierarchical 10GbE network containing a mix of core and edge switches. We then average out the cost of those switches across the 10GbE NICs installed in the servers to arrive at a cost estimate of $750 per NIC.

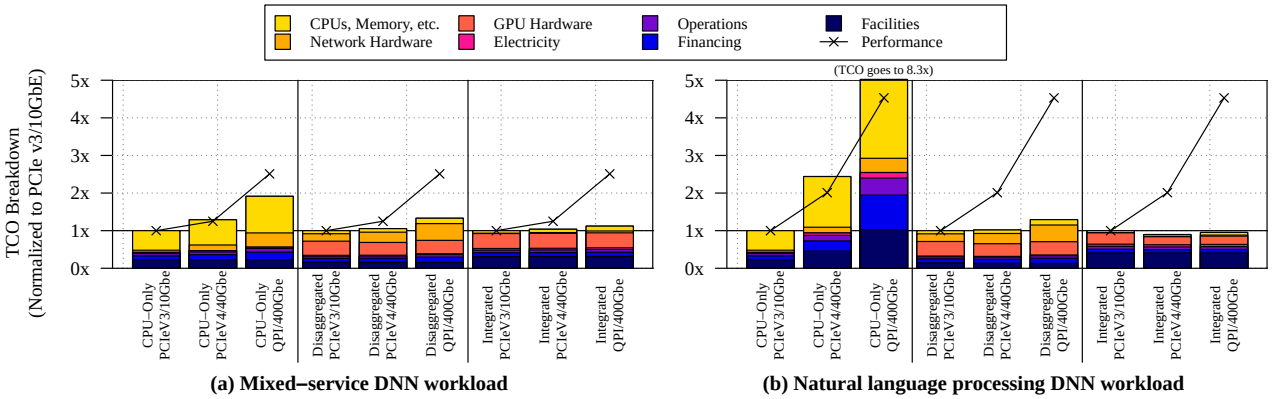
To characterize each WSC design, we first assume a workload composed in part by one of the DNN service mixes described in Table 5 and in part by non-DNN webservices. For this mix of webservices, we provision enough compute for the *CPU Only* design point to characterize its TCO and obtain a series of performance targets for each service. For example, given a workload composed of 70% from the MIXED DNN workload along with 30% non-DNN services, we would provision 30% of the servers to non-DNN services and 10% to each of the DNN services (the MIXED workload is composed of 7 services). We then build out the *Integrated GPU* and *Disaggregated GPU* designs, each matching the throughput obtained by the *CPU Only* design, finally applying the model described above to characterize their TCO.

DNN's Implications for WSC Design The results of our TCO analysis are presented in Figure 15 for (a) the MIXED workload, (b) the IMAGE workload and (c) the NLP workload. Each plot presents the TCO of the three WSC designs across a range of assumptions about the mix of DNN and non-DNN services (x-axis), where the presented TCO is normalized to the *CPU Only* case and presented on a log scale (y-axis).

For the MIXED workload presented in Figure 15a, both GPU-based designs show substantial improvements over the *CPU Only* design, except when the workload is composed almost entirely of non-DNN services. This demonstrates that there are potentially sizable cost savings available by accelerating DNN-based services (up to 20× for *Disaggregated GPU* design) as these services consume an increasing volume of cycles in WSCs. The *Disaggregated GPU* design also improves upon the *Integrated GPU* design by between 10% and 2×, which we can attribute to the relatively inefficient use of GPUs by some of the DNN services in the *Integrated GPU* design. In particular, each server in the *Integrated GPU* design utilizes the same number of GPUs, while the NLP services can saturate only a subset of those available GPUs because they are bandwidth-limited by the PCIe interface. This inefficiency is alleviated by the *Disaggregated GPU* design, which decouples CPUs and GPUs and allows for fewer GPUs to be

Table 6: Interconnect and network configurations. We design the networks to use bonded ethernet connections numerous enough to saturate the CPU/GPU interconnect, assuming an additional protocol overhead of 20% on ethernet. Prices are phrased as the purchase cost over the PCIeV3/10GbE design point

	Interconnect			Ethernet	
	Architecture	Bandwidth (GB/s)	Price ($)	Bandwidth (GB/s)	Price ($/NIC)
PCIeV3/10GbE	1× PCIe v3 bus shared by GPUs	15.87	+$0	1.25 per NIC, up to 16 NICs	+$0
PCIeV4/40GbE	1× PCIe v4 bus shared by GPUs	31.75	+$2000	5 per NIC, up to 9 NICs	+$1250
QPI/400GbE	1 QPI link between GPU and CPU socket 6 links/GPUs per socket	307.2 (25.6 per link)	+$4000	50 per NIC, up to 8 NICs	+$4250



(a) Mixed–service DNN workload **(b) Natural language processing DNN workload**

Figure 16: TCO breakdown showing the impact of future networking technologies on GPU-enabled WSCs housing DNN services. Improved bandwidth is key to unleashing the capabilities of GPUs for several DNN applications

employed in the WSC.

The IMAGE workload, presented in Figure 15b, behaves similar to the MIXED workload, except there is a crossover point when the number of DNN services exceeds 72% of the workload. After this point, the *Integrated GPU* design has lower TCO than the *Disaggregated GPU* design. Because the TCO benefits in the *Disaggregated GPU* design over the *Integrated GPU* design arise from over-provisioning GPUs in the *Integrated GPU* design, those benefits slowly disappear as the workload running in the WSC is comprised of more DNN-based services that utilize all of the GPUs in the server (i.e. IMC, FACE and DIG).

The NLP case, presented in Figure 15c has a similar trend to 15a: the *Disaggregated GPU* model has the lowest TCO over most of the workload mixes and is a modest improvement over the *Integrated GPU* design over that entire range. However, the TCO for the NLP case is much closer to the TCO of the *CPU Only* design, showing a maximum improvement of 4×, as opposed to the 20× for the MIXED case. This difference occurs because, instead of being partially composed of NLP services as in the MIXED workload, the NLP workload is composed entirely of NLP services. Because the performance of the NLP applications is bound by the bandwidth of the PCIe, the available GPUs cannot be fully utilized.

6.4. Addressing the Bandwidth Bottleneck

To address this bandwidth limitation, we consider two alternative designs to the typical configuration comprised of GPUs supplied by PCIe v3 and a 10GbE network. First, representative of cutting edge technology available today, we describe a design that connects the GPUs with PCIe v4, which doubles the bandwidth of PCIe v3 to 31.75GB/s. Accordingly, we provision the network to also have more bandwidth by using a 40GbE network with teamed connections at the server level. Assuming a 20% protocol overhead for ethernet, the PCIe v4 bus can be saturated by 9 teamed 40GbE connections. Second, representative of a more aggressively designed system that uses near-future technology, we consider a design that employs Quick Path Interconnect (QPI) [29] to connect CPUs to GPUs inside the server. Assuming 12 GPUs inside a 2-socket server, 6 point-to-point QPI links would be needed in each socket. Standard QPI links available at the time of this writing yield 25.6 GB/s, which is a total of 307.2 GB/s across all 12 links. To provision enough bandwidth in the network to feed the GPUs, and again assuming a 20% protocol overhead for ethernet, 8 teamed 400GbE connections are sufficient to saturate the QPI links.

We summarize these alternative design points in Table 6. Included in the table are our assumptions about the cost of these alternative designs, which are developed using a similar methodology described for the PCIe v3/10GbE design point, along with projections of the unit costs for PCIe v4, QPI, 40GbE NICs/switches and 400GbE NICs/switches.

Network Impact on Performance and TCO We characterize the impact of these design points with improved bandwidth by scaling up the networking equipment in the *Disaggregated GPU* model. We also make the assumption that bandwidth-constrained DNN services (NLP) bypass the bandwidth lim-

itations demonstrated in Figure 13 and continue to scale up in throughput beyond the throughput measured on our GPU-enabled server. In the *Disaggregated GPU* design, we model this performance improvement due to scaling up the network then introduce designs for the *CPU Only* and *Integrated GPU* cases that match the performance improvement. Note that we model the *CPU Only* designs as having PCIe v3 and 10GbE, as improving the network does little to improve performance of the *CPU Only* design.

We present the results of this exercise in Figure 16, applying it to workloads comprised entirely of either the MIXED DNN service (a) or of the NLP DNN service (b) in Figures 16 (the IMAGE workload is not bandwidth constrained, so it is not considered here). The figure shows the performance improvement achieved by introducing the improved network into the *Disaggregated GPU* design as black lines with "x" marks. Each group of bars shows the growth in various components of TCO that are associated with growing the WSC to improve performance.

We can draw several interesting conclusions from these experiments. First, improving the bandwidth provisioning in the network is an essential step to unlocking the full potential of GPUs for bandwidth-heavy NLP services. Large performance improvements can be realized while minimally impacting TCO in GPU-enabled WSCs. As the figure shows, the growth in TCO for the *Disaggregated GPU* design stems primarily from increased networking costs because the approach relies heavily on the network to pass large amounts of data from CPU-based compute servers to GPU-centric servers. In the *Integrated GPU* design the cost increases are slight, showing up primarily in the MIXED workload as increases in the server cost (PCIe and QPI costs appear as part of the server costs). For the NLP workload, improving the bandwidth actually reduces TCO slightly for both improved network designs. This occurs because the increased utilization of GPUs allows the design to use fewer GPUs while still improving performance significantly. Second, scaling up the performance of DNN-based services is extremely difficult to do without accelerating them. For both the MIXED and NLP workloads, scaling up throughput requires scaling up the number of servers in the *CPU Only* design roughly in proportion to that increase. Given current CPU and GPU designs, this identifies GPUs as being the more promising direction for scaling up DNN-based webservices.

7. Related Work

Deep learning techniques are outperforming state-of-the-art traditional machine learning methods in speech and image tasks [22, 28]. There is growing interest both in implementing software for deep learning methods within open source libraries [12, 21, 27] and in improving hardware designs for DNNs via CPU optimizations [45] and ASICs [14, 15, 32, 39, 44]. In this work, we focus on leveraging commodity GPU accelerators to optimize the throughput of DNN and on relieving bandwidth bottlenecks in the network and interconnect to sustain high throughput across DNN-based services.

The Catapult project [38] at Microsoft Research ported key components of Bing's page ranking to FPGAs, showing the ongoing need for specialized hardware for datacenter applications. Researchers also investigated how future datacenters must evolve by accelerating intelligent personal assistant workloads housed in WSCs [24].

In addition to prior work investigating datacenter efficiency and utilization [23, 25, 30, 34–36, 42, 47, 48], Microsoft studied reducing the total amount of machines needed in a datacenter to train an image classification network [17] increasing its efficiency. DistBelief [33] investigated distributing deep learning tasks across large systems efficiently, and Coates et al. [18] investigated designing a large network of GPUs connected with high-speed interconnects specialized for deep learning and show they are able to effectively distribute computation in a WSC. These systems investigate training deep learning networks while this work focuses on the inference task of DNNs in online applications.

8. Conclusion

This work introduces **DjiNN**, an open source deep neural network service and **Tonic Suite**, a suite consisting of 7 end-to-end DNN-based applications in the vision, speech, and natural language processing domains. Using DjiNN, we design a high-throughput DNN system based on massive GPU server designs. In most cases, our final server design achieves over a 100× throughput gain on a single GPU compared to the CPU baseline, and achieves almost linear scaling with the number of GPUs. We study the total cost of ownership to provide insights into designing future warehouse scale computer architectures for DNN services. In terms of total cost of ownership, GPU-enabled datacenters show an improvement over CPU-only designs by 4-20×. In the case of bandwidth-heavy NLP applications, we show that leveraging improved GPU interconnect and network components to alleviate bandwidth constraints is one of the keys to achieving the aforementioned improvements.

9. Acknowledgment

We thank our anonymous reviewers for their feedback and suggestions. This work was partially sponsored by Google, ARM, and by the National Science Foundation (NSF) under grants CCF-SHF-1302682 and CNS-CSR-1321047.

References

[1] "Cuda toolkit documentation," http://docs.nvidia.com/cuda/profiler-users-guide/.

[2] "DjiNN and Tonic: DNN as a Service," http://djinn.clarity-lab.org.

[3] "Facebook's quest to build an artificial brain depends on this guy," www.wired.com/2014/08/deep-learning-yann-lecun.

[4] "Google Glass," www.google.com/glass/start.

[5] "Inside the artificial brain that's remaking the google empire," www.wired.com/2014/07/google_brain/.

[6] "Maple 2015. maplesoft, a division of waterloo maple inc., waterloo, ontario." http://www.maplesoft.com/.

[7] "Microsoft corp to challenge apple inc with siri alternative: More intelligent and fast enough!" www.dazeinfo.com/2013/06/18/microsoft-corp-to-challenge-apple-inc-with-siri-alternative/-more-intelligent-and-fast-enough.

[8] "Multi-process service," https://docs.nvidia.com/deploy/pdf/CUDA_Multi_Process_Service_Overview.pdf.

[9] "Nvidia visual profiler," https://developer.nvidia.com/NVIDIA-visual-profiler.

[10] "Apple's Massive New Data Center Set To Host Nuance Tech," http://techcrunch.com/2011/05/09/apple-nuance-data-center-deal/, 2011.

[11] L. A. Barroso, J. Clidaras, and U. Hölzle, "The datacenter as a computer: an introduction to the design of warehouse-scale machines," *Synthesis Lectures on Computer Architecture*, 2013.

[12] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio, "Theano: new features and speed improvements," *Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop*, 2012.

[13] B. C. Becker and E. G. Ortiz, "Evaluating open-universe face identification on the web," in *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2013.

[14] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *Architectual Support for Programming Languages and Operating Systems(ASPLOS)*, 2014.

[15] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun *et al.*, "Dadiannao: A machine-learning supercomputer," in *Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on*. IEEE, 2014, pp. 609–622.

[16] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," *arXiv preprint arXiv:1410.0759*, 2014.

[17] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, "Project adam: building an efficient and scalable deep learning training system," in *Operating Systems Design and Implementation(OSDI)*, 2014.

[18] A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, and N. Andrew, "Deep learning with cots hpc systems," in *International Conference on Machine Learning(ICML)*, 2013.

[19] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *The Journal of Machine Learning Research*, 2011.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[21] I. J. Goodfellow, D. Warde-Farley, P. Lamblin, V. Dumoulin, M. Mirza, R. Pascanu, J. Bergstra, F. Bastien, and Y. Bengio, "Pylearn2: a machine learning research library," *arXiv preprint arXiv:1308.4214*, 2013.

[22] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *International Conference on Acoustics, Speech and Signal Processing(ICASSp)*, 2013.

[23] J. Hauswald, T. Manville, Q. Zheng, R. Dreslinski, C. Chakrabarti, and T. Mudge, "A hybrid approach to offloading mobile image classification," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[24] J. Hauswald, M. A. Laurenzano, Y. Zhang, C. Li, A. Rovinski, A. Khurana, R. G. Dreslinski, T. Mudge, V. Petrucci, L. Tang, and J. Mars, "Sirius: An open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers," in *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2015.

[25] C.-H. Hsu, Y. Zhang, M. A. Laurenzano, D. Meisner, T. Wenisch, L. Tang, J. Mars, and R. Dreslinski, "Adrenaline: Pinpointing and reigning in tail queries with quick voltage boosting," in *International Symposium on High Performance Computer Architecture (HPCA)*, 2015.

[26] X. Huang, J. Baker, and R. Reddy, "A historical perspective of speech recognition," *Commun. ACM*, 2014.

[27] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012.

[29] N. Kurd, J. Douglas, P. Mosalikanti, and R. Kumar, "Next generation intel® micro-architecture (nehalem) clocking architecture," in *VLSI Circuits, 2008 IEEE Symposium on*. IEEE, 2008, pp. 62–63.

[30] M. Laurenzano, Y. Zhang, L. Tang, and J. Mars, "Protean code: Achieving near-free online code transformations for warehouse scale computers," in *International Symposium on Microarchitecture (MICRO)*, 2014.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

[32] D. Liu, T. Chen, S. Liu, J. Zhou, S. Zhou, O. Teman, X. Feng, X. Zhou, and Y. Chen, "Pudiannao: A polyvalent machine learning accelerator," in *International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS)*, 2015.

[33] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab: a framework for machine learning and data mining in the cloud," *Proceedings of the VLDB Endowment(PVLDB)*, 2012.

[34] J. Mars and L. Tang, "Whare-map: Heterogeneity in "homogeneous" warehouse-scale computers," in *International Symposium on Computer Architecture (ISCA)*, 2013.

[35] J. Mars, L. Tang, R. Hundt, K. Skadron, and M. L. Soffa, "Bubble-up: Increasing utilization in modern warehouse scale computers via sensible co-locations," in *International Symposium on Microarchitecture (MICRO)*, 2011.

[36] V. Petrucci, M. A. Laurenzano, Y. Zhang, J. Doherty, D. Mosse, J. Mars, and L. Tang, "Octopus-man: Qos-driven task management for heterogeneous multicore in warehouse scale computers," in *International Symposium on High Performance Computer Architecture (HPCA)*, 2015.

[37] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.

[38] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray *et al.*, "A reconfigurable fabric for accelerating large-scale datacenter services," in *International Symposium on Computer Architecture (ISCA)*, 2014.

[39] W. Qadeer, R. Hameed, O. Shacham, P. Venkatesan, C. Kozyrakis, and M. A. Horowitz, "Convolution engine: balancing efficiency & flexibility in specialized computing," in *International Symposium on Computer Architecture(ISCA)*, 2013.

[40] A. Research, "Wearable Computing Devices, Like Apple iWatch, Will Exceed 485 Million Annual Shipments by 2018," 2013.

[41] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge(ILSVRC)," 2014.

[42] M. Skach, M. Arora, C.-H. Hsu, Q. Li, D. Tullsen, L. Tang, and J. Mars, "Thermal time shifting: Leveraging phase change materials to reduce cooling costs in warehouse-scale computers," in *Proceedings of the 42nd Annual International Symposium on Computer Architecture (ISCA)*, ser. ISCA '15, 2015.

[43] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Computer Vision and Pattern Recognition (CVPR)*, 2014.

[44] O. Temam, "A defect-tolerant accelerator for emerging high-performance applications," in *ACM SIGARCH Computer Architecture News*, 2012.

[45] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on cpus," in *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, 2011.

[46] R. C. Whaley and J. Dongarra, "Automatically tuned linear algebra software," in *SuperComputing: High Performance Networking and Computing*, 1998.

[47] H. Yang, A. Breslow, J. Mars, and L. Tang, "Bubble-flux: Precise online qos management for increased utilization in warehouse scale computers," in *International Symposium on Computer Architecture (ISCA)*, 2013.

[48] Y. Zhang, M. Laurenzano, J. Mars, and L. Tang, "Smite: Precise qos prediction on real system smt processors to improve utilization in warehouse scale computers," in *International Symposium on Microarchitecture (MICRO)*, 2014.