Research Statement
Hyperscale Data Processing with Network-centric Designs

Qizhen Zhang

One of the most fundamental tasks in computer science is to process data in a timely manner. It is critical for nearly every computing workload and affects aspects of our lives as diverse as health (e.g., tracking outbreaks in a pandemic), finance (e.g., fast trading and quantitative analysis), education (e.g., large-scale resource sharing), and entertainment (e.g., massive online services and recommendations). Unfortunately, unprecedented data growth has meant that achieving good performance is increasingly challenging for all of the components involved: the infrastructure must provide massive amounts of resources, data processing systems must utilize the resources efficiently, and applications sometimes need different algorithm designs.

My long-term goal is to make the processing of data in any form efficient at any scale. To that end, I have researched and published across the data processing stack, from novel data science applications [1] to large-scale data processing systems [2, 3, 4, 5] and their underlying infrastructure [6, 7, 8, 9]. These papers have appeared at database, networking, and systems conferences. My dissertation research focuses on cloud data centers, where today’s largest data processing workloads are hosted. Over the years, these workloads have ballooned to hyperscale level, encompassing billions to trillions of data items and hundreds to thousands of machines per query. Enabling and expanding with these workloads are highly scalable data center networks that connect up to hundreds of thousands of networked servers.

At hyperscale, many traditional design principles break down. One example is the classic principle of layering, in which different components of a network, from applications to the transport and hardware, are built independently as layers and connected by well-specified protocols. Layering has allowed people working on different components to focus on their own systems with clear optimization goals. However, layered designs are no longer sustainable at hyperscale. For instance, without knowing how their data is transferred in the network, applications can make egregious decisions in their execution models; the cloud infrastructure also performs poorly without rethinking the interfaces and services exposed to its applications. In fact, most cloud providers already break layering in their current architectures, e.g., creating custom networking stacks and hardware that are tailored to their most common applications. The research community has not kept up, due to lack of systematic investigations on applying cross-layer designs to address hyperscale challenges in data processing. My research aims to fix this by bridging data processing systems and data center networks. My approach is principled and concerns three questions and associated challenges as follows.

How do data processing systems perform in current networks? Developers of data processing systems treat the network as a black box that simply delivers messages from point A to point B. This assumption greatly simplifies the design of distributed data processing: the execution engine can ignore details like the physical placement of distributed workers. The network has also traditionally tried to support this assumption with a “one big switch” abstraction that it provides to applications. It incurs a great cost to maintain the assumption at scale because a data processing job that requires hundreds of machines must necessarily span multiple racks or even clusters. Instead, today’s data center networks are commonly oversubscribed due to cost considerations, meaning that the cross-rack and cross-cluster network performance is significantly worse than that within racks and clusters. Such data center network characteristics are important for large-scale data processing but rarely considered in prior systems.

In an effort to address these challenges, I built GraphRex [3], the first system that processes hyperscale data by systematically adopting network awareness. These ideas demonstrated orders of magnitude better performance than the state of the art.

How will they perform in future networks? Disaggregated data centers (DDCs) are a promising new cloud proposal that decouples different types of resources from monolithic servers into resource pools, e.g., compute/CPU pool, memory pool, and storage pool, and connects these pools by a high-speed network. Compared to traditional server architectures, disaggregation offers vast operational benefits. It solves the traditional bin-packing problem when packing virtual machines (VMs) to physical machines by making resource allocation independent, potentially saving billion-dollar resource waste. DDCs also make data center expansion easier as different resources can be added and managed independently. In addition, DDCs achieve better failure isolation and provide easy elasticity for applications. Despite these benefits, this new cloud architecture can potentially
disrupt data processing systems, particularly due to memory disaggregation that completely separates compute and data.

My work pioneered the rethinking of data processing system designs in DDCs [4] and, for the first time, investigated the effect of DDCs on production systems [5], which opened a broad space of research opportunities. I proposed TELEPORT [7], a new DDC feature for efficient data processing. I also worked with industry on realizing DDC benefits in today’s clouds [8].

How do networks evolve? Finally, to enable the network innovations introduced above, I note that an essential part is to evaluate the performance of the proposals. Unfortunately, evaluating hyperscale networks is intractable because of their size and complexity. This is true for testbeds, where few, if any, can afford a dedicated, full-scale replica of a data center. It is also true for simulations, which while originally designed for at-scale network evaluation, have struggled to cope with today’s hyperscale infrastructure. For example, simulating the TCP protocol for 60 seconds in a data center of a thousand machines takes 36 days to finish! Evaluation is becoming the main roadblock for data center network innovations.

To support my work and that of others, I developed MimicNet [6], the first scalable approach to accurate performance estimation for data center networks. It evaluates recent data center proposals orders of magnitude faster than current approaches.

Summary. Overall, my dissertation seeks to address fundamental systems challenges in hyperscale data processing in the context of cloud data centers. I work across traditionally siloed layers to design novel solutions to scale their performance. My general finding is that for large-scale data processing, the network is often either the performance bottleneck or the leverage we can use to solve scale problems. Hence, I call my approach network-centric designs, which have been shown to be successful by the systems I build and the experience I have with industry.

Dissertation Work

Figure 1 maps out the components of my dissertation, following the three directions introduced above. All systems are designed for more efficient data processing at hyperscale. I now present them in detail.

Network-aware Data Processing

The first component is about applying current data center domain knowledge to data processing, for which I have developed systems to improve both graph processing and database analytics.

Graphs. Large-scale graph analytics is a popular example of hyperscale data processing as real-world graphs now scale up to billion vertices and trillion edges. My study on massive workloads showed that network communication dominated the times in processing large graph queries. However, state-of-the-art systems were incapable of capturing network characteristics and thus suffered from substantial communication cost.

This motivated GraphRex [3], a system I developed for processing graph queries at data center scale. GraphRex has three goals: ease of programming, querying efficiency, and robustness to network dynamics. These goals are required in practice but achieving all three together is difficult. We must carefully design the query language, execution, and optimizations with domain-specific knowledge.

GraphRex achieves the first goal with a declarative and easy-to-use query interface. The second and third goals require it to reduce overall network traffic, especially over bottleneck links. Our solution introduces new operators to the execution engine of GraphRex, called global operators, which consider data center network characteristics. For instance, as one of the global operators, our shuffle operator exchanges messages between workers in a topology-aware fashion, which consolidates messages when network cost is low and compresses
them to minimize the amounts of data sent through oversubscribed links. Combined, these operators substantially level up the performance and robustness of GraphRex in data centers.

We evaluated GraphRex with large real-world graphs in a data center testbed that has several terabytes of memory and thousands of CPU cores, a scale never tested in prior declarative graph systems. GraphRex proved to be two orders of magnitude faster than state-of-the-art systems under various network conditions. Databases. The network cost models in distributed database management systems (DBMSs) have not kept up with network advances. E.g., they treat network cost the same as disk I/O cost but today’s data center networks both outperform and outpace SSDs. We proposed more accurate models for popular distributed DBMSs [10].

Data Processing with Resource Disaggregation

The second component focuses on future networks in disaggregated data centers (DDCs). My research team and I have been pioneering the work on adopting this new architecture for data processing.

Rethinking data processing systems in DDCs. We opened the topic of data processing in DDCs with microbenchmarks on hash operations [4]. We showed that the separation of compute and memory, i.e., memory disaggregation, causes significant overhead for data center applications. It happens because every data access to the main memory now translates to a network communication. This overhead is particularly felt by data processing systems, which hold large working sets in memory. We proposed a set of novel data processing operators for reducing the overhead.

Understanding production systems in DDCs. The largest public clouds are already in the transition to disaggregated architectures, including memory disaggregation. Fully understanding DDC implications on data processing is hence both urgent and important. We took the first step to investigate DDC effect on production systems [5]. By studying DBMSs, which execute memory-intensive queries, we found that both the benefits and overhead of DDCs are substantial. On one hand, a large disaggregated memory pool can prevent the processing of memory-intensive queries from being spilled to secondary storage. On the other hand, network communications for remote memory accesses are expensive for large queries.

TELEPORT. To overcome the overhead of DDCs and unlock all their benefits, we introduced TELEPORT [7], a new feature of DDCs for achieving optimal data processing performance. TELEPORT is a compute pushdown framework that enables data processing systems to offload expensive operations close to data. It is based on disaggregated operating systems (OSes) that emulate traditional OS interfaces to provide backward compatibility in DDCs, so that current applications can directly run to harvest the benefits. With TELEPORT, applications are capable of executing light-weight but memory-intensive operations in the memory pool. In doing so, they eliminate costly remote memory accesses and hence achieve better performance.

TELEPORT is unique in its generality and efficiency. With a new system call, it allows applications to offload arbitrary pieces of computation by wrapping them as functions. Pushing a function down is as simple as providing the pointers of the function and its arguments to the memory pool. This is possible because applications’ stack, heap, and code pages all live in the memory pool as a byproduct of disaggregated OSes.

Data synchronization is critical for TELEPORT: the compute pool caches part of the main memory, so data copies in different pools can diverge before, during, and after pushdown. Without proper synchronization, concurrent threads in two pools may access the same memory pages without observing each other’s updates. TELEPORT employs specialized synchronization primitives that guarantee memory coherence. It only synchronizes data on applications’ demands, which outperforms application-agnostic alternatives.

We applied TELEPORT to three popular data processing workloads: in-memory database, graph processing, and MapReduce. Compared to baseline DDCs, these workloads execute up to an order of magnitude faster with TELEPORT. In addition, TELEPORT requires little effort from its users.

Redy. With TELEPORT, DDCs provide a radical solution to the limitations of current cloud infrastructure. Can we preharvest some of the DDC benefits in today’s clouds? To seek the answer, I collaborated with Microsoft researchers. Microsoft Azure data centers have massive amounts of unused memory, much of which is stranded because all CPU cores on the machines are allocated to VMs. Nevertheless, stranded memory can be productively employed by accessing it via Remote Direct Memory Access (RDMA). RDMA can access remote memory without involving remote CPUs and bypass OS kernels for low latency. Based on these insights, we developed Redy [8], a new cloud service that uses stranded memory as remote caches. It offers a lower-latency alternative to SSDs, using disaggregated memory resources that would otherwise go to waste.

Our use of RDMA leads to two challenges. The first is performance. Tuning RDMA requires complex, low-level optimizations to trade off network latency, throughput, and resource cost. There is no one-size-fits-
all configuration for RDMA. Second, stranded memory resources are highly dynamic. They come and go depending on VM allocations. Their availability can be as short as a few minutes. Providing reliable caches using dynamic memory is hard.

Redy addresses the first challenge with SLO-based configuration. It takes as input a user-defined performance service-level objective (SLO) and uses a dynamic optimization process to automatically find the configuration that satisfies the SLO with minimal resource cost. To solve the dynamic challenge, we developed a dynamic memory manager that migrates a cache to new stranded memory when the old memory is reclaimed by the cloud VM allocator. The migration occurs in a way that minimizes the impact on cache performance.

We integrated Redy with a production key-value store to demonstrate its ease of use and performance benefits. Results show that Redy caches are 20× and 8× faster than an SSD and an RDMA baseline respectively.

Facilitating Network Innovation

The final component of my dissertation concerns network innovation. We focused on evaluating network proposals with packet-level simulation, which was originally designed with three goals: providing performance results for arbitrary scale with arbitrary network extensions and arbitrary user instrumentation. Unfortunately, simulating data centers is prohibitive: parallelization barely works as the complexity of the network forces simulators to serialize all events; approximations such as flow-level approaches lack accuracy and generality.

**MimicNet.** My coauthors and I built MimicNet [6], a fast and scalable evaluation framework for data center networks based on packet-level simulation. MimicNet assumes the popular FatTree topology, where racks of servers are connected by a cluster network, and clusters are connected by a set of core switches. Cluster is the unit for scaling—real-world data centers can have thousands of clusters. Based on this topology, MimicNet works as follows. It first runs a small simulation of two clusters in full fidelity. Using the simulation results, it trains machine learning (ML) models for approximating intra-cluster and inter-cluster behavior. Finally, it runs an N-cluster simulation by composing (1) a single ‘observable’ cluster for user instrumentation, regardless of the total number of clusters in the data center, and (2) N − 1 ‘un-observed’ clusters. All components in the observable cluster and all of the remote components with which it communicates are simulated in full fidelity. All other behavior that is not directly observed by the user is approximated by the trained models. In essence, MimicNet predicts the performance of a large network by observing only small subsets of it. By removing the simulation of most clusters, it decreases the simulation time by orders of magnitude.

Network complexity makes achieving high accuracy in MimicNet approximations challenging. We address this by baking data center domain knowledge into the designs of the ML models such as their learning features and loss functions. Additionally, we allow users to trade accuracy off for higher simulation speed by training smaller models instead of more accurate larger ones.

Our experiments with real-world traces showed that MimicNet simulates data center networks orders of magnitude faster than full-fidelity packet-level simulation, and its results closely mimic the ground-truth. For example, while it takes more than a month to simulate recent network innovations in a data center of a thousand servers in full fidelity, MimicNet finishes in a few hours and its predictions are within 5% of the true results.

**Generalization.** We developed an ML platform to present network modeling as a challenge problem [11]. We offer both a formal problem definition and an experimentation framework based on packet-level simulation. By inviting more researchers, we hope network modeling can be tackled more broadly and generally with ML.

Other Work

In addition to the systems that constitute my dissertation, I have also worked on several exciting projects that focus on other aspects of data processing such as applications, cost efficiency, and fault tolerance.

**Data science application with machine learning.** Crowdfunding is a new mechanism for connecting entrepreneurs with thousands of online investors. However, no previous studies have investigated its effectiveness and what strategies startups should adopt to maximize their chances. With a 7-month data collection, we tracked the activities of over 4000 startups on AngelList, a popular crowdfunding website, Twitter, and Facebook. With the dataset, we predicted whether startups succeed or fail in crowdfunding based on novel machine learning techniques and features that describe startup social engagement.

**Performance and cost analysis of graph processing.** Many graph processing systems have adopted distributed, shared-nothing architectures to run in a cluster of machines. Others argued that a single machine is
often enough. There has been no consensus on which approach is better. From user perspective, it is difficult to select the best system given a workload. We performed the first study of the performance and cost of state-of-the-art graph processing systems [2]. Our analysis revealed that the systems that achieve the highest performance are often different than those with the lowest cost. Optimality depends on the input graph, the query, and targeted metric. Our detailed analysis provides useful insights for the selection and development of graph systems, including GraphRex [3].

**Faster and cheaper remote caching.** Remote caching systems like Redy [8], Redis, and memcached let applications offload large states to remote servers. However, even with fast caches, performance degradation is still significant due to the difference between local and remote memory access latency. We proposed CompuCache [9], a new service that supports offloading *both data and computation over the data* to remote caches. CompuCache achieves higher performance by single-round-trip offloading with server-side pointer-chasing. It also uses spot VMs as cache servers for lowering the cost. Since spot VMs are unreliable, CompuCache reacts quickly to failures. Our experiments showed that the throughput of CompuCache is 200× higher than that of Redis, achieving 126 million offloading invocations per second with a single server.

**Fault-tolerant data processing with disaggregated memory.** Processing, memory, and storage are separated from each other in DDCs. So are their failures. This isolation provides a new perspective on fault tolerance: with a reliable and disaggregated memory backend, the failures of compute workers can be recovered fast. Our system, Flushing [12], decides when and how compute workers should checkpoint their state to the memory server. It guarantees data consistency and optimizes overall system performance.

### Future Work

My plans for future work center around my long-term goal—to make the processing of data in any form efficient at any scale. The network-centric systems I have built sketch out the broad picture of hyperscale data processing and show the difficulties of achieving good performance. Many challenges remain as both applications and infrastructure evolve. I would like to move forward by applying principled approaches to process data in more diverse forms and in next-generation networks.

**Diverse workloads as data processing.** My research so far has been scoped in the processing of a few representative cloud workloads (relational databases [4, 5, 7], key-values [8, 9], and graphs [3, 2, 7]). Other types of workloads are also increasingly important and can just be processed as data. One example is processing streams from IoT (Internet of Things) devices, machine-generated logs, and cameras, where states in the processing are unbounded. Another popular example is machine learning (ML) applications, which are now powered by massive models, datasets, and computing infrastructure. Distributed ML can experience expensive network overheads in some of its critical components such as aggregating and broadcasting parameters between workers and parameter servers in data parallel training. ML models are becoming extremely large—up to hundreds of billions of parameters. I am interested in training and executing such massive models as special hyperscale data processing tasks. I plan to extend the designs in my dissertation and explore the space for these workloads by involving both network and application-specific expertise.

Emerging cloud computing paradigms like blockchains and serverless can also suffer from network bottlenecks in data centers, e.g., transaction ordering in permissioned blockchains and data shuffling between serverless functions. Although specific optimizations have been proposed recently for these platforms, it is appealing to apply network-centric designs to address the bottlenecks in systematic and scalable ways.

**New cloud trends.** An important trend in data center design is that cloud providers are augmenting their data centers with more domain and application-specific accelerators such as SmartNICs, FPGAs and ASICs to meet the rapidly increasing computation demand. Another trend is that data center networks are becoming more programmable. An example is programmable network switches, which can do more than just forwarding packets. Users can execute a wide range of operations over packets in network at line rate. I am interested in *automatically* optimizing data processing by *universally* leveraging these accelerators and programmable switches, e.g., offloading components that are costly for end hosts. My experience of building TELEPORT and CompuCache will be useful.

**Next-generation Internet.** Like data center networks, the Internet is becoming ultra fast. In the coming 5G networks, edge devices can communicate with cloud servers with multi-Gbps bandwidth. 6G is expected to be orders of magnitude faster. There are two implications of high-speed Internet drawing my attention. First, more data will likely be uploaded to clouds for processing with stricter timing requirements. This would add
higher pressure to both cloud infrastructure and data processing systems. I plan to investigate the new challenges introduced by massive load increase from the Internet. The other implication is that data processing jobs can scale beyond data centers—operations can be efficiently placed on and moved between edge devices and cloud servers. This will likely create new computing paradigms that are much larger than today’s distributed processing. The whole Internet would work like a giant data center, in which we can utilize global resources for data processing at any scale. Many techniques I mentioned may still be useful, e.g., network awareness, caching, and automatic compute offloading, but there will be new challenges. In particular, wide-area networks can experience great performance variation due to the mobility of edge devices. Proposing new architectures to improve network robustness and codesigning data processing systems to provide end-to-end guarantees are promising directions that I would like to explore.

In general, I am interested in data management, computer systems, and networking. In the course of my Ph.D. study, I am fortunate to have closely collaborated with 7 professors, 6 researchers, and 15 students across multiple institutes in academia and industry on a variety of topics. I am looking forward to embarking on exciting adventures with new colleagues and students.

References