Research Statement

Mosharaf Chowdhury

With the rapid rise of cloud computing, scale-out applications running on large clusters are becoming the norm. Although the diversity of applications and the capacity of datacenters are continuously growing, application- and network-level goals are moving further apart. Data-parallel applications often care about *all* their parallel communications, yet the network treats each point-to-point communication independently – this fundamental mismatch has resulted in complex point solutions for developers, a myriad of configuration options for end users, and an overall loss of performance. My thesis research has focused on bridging the gap between application-level performance and network-level optimizations by revisiting and extending traditional concepts of networking to leverage the communication semantics of emerging big data applications.

The primary themes of my research are: (1) understanding requirements, constraints, and tradeoffs that actually matter to applications, (2) leveraging these insights to develop deployable solutions, and (3) distilling them to identify fundamental principles. They have borne fruit in several areas. My work on application-aware networking [1–4] has generalized point-to-point communication to data-parallel applications, creating novel opportunities for network scheduling and load balancing, and I have designed resource allocation algorithms [5–8] for private datacenters, public clouds, and virtual networks considering the unique tradeoffs in each environment. Most of my projects are open source and tested in the cloud, and several have been deployed in commercial datacenters.

Application-Aware Networking

Despite fundamental shifts in computing requirements, data-parallel applications still use the Internet-era *flow* abstraction. While flows are sufficient for client-server applications, they cannot capture the semantics of a data-parallel communication, restricting (1) how the network can further application-level goals and (2) how applications can assist in better managing the network. During my Ph.D., I have worked on mitigating challenges in both these directions.

Coflow Scheduling: Communication is crucial for analytics at scale. Yet, traditional approaches toward improving network-level metrics are often misaligned with the goals of data-parallel applications. I first observed this phenomenon in 2010 while developing the communication layer of Apache Spark [9]. Although a data-parallel application treats all the flows in each communication stage (e.g., shuffle or broadcast) as a whole, the network treats each flow independently. The workaround was to override network-level optimizations from the application layer using multiple user-tunable parameters. It quickly became clear that we need a clean abstraction that can help the network realign its objectives with that of the applications instead of manually tuning every application.

A communication stage often cannot complete until all the flows have completed – based on this observation, I proposed the *coflow* abstraction to capture such collections of flows with multiple bottlenecks and to convey their application-level semantics to the network. In order to fully utilize the additional information exposed through coflows, the network must determine how to (1) optimize individual data-parallel communication patterns, (2) arbitrate between multiple applications in a shared network, and (3) react well to application-level unpredictabilities.

Orchestra [1] focuses on the first problem, i.e., intra-coflow scheduling. We demonstrated that shunning traditional flow-level fairness in favor of data-proportional rates can improve the completion time of a shuffle. Similarly, we proposed a BitTorrent-like broadcast protocol that does the opposite of BitTorrent for the Internet – it prioritizes slow peers instead of penalizing them to ensure that the *entire* broadcast finishes as fast as possible. Our nonconventional solution results in $4.5\times$ faster broadcast completions on the same network in comparison to traditional approaches.

Coflows, however, do not exist in isolation. Once we could optimize individual coflows, we focused on coordinating coflows from coexisting data-parallel applications. The Varys [2] project reformulates network optimization as the inter-coflow scheduling problem, extending both networking and scheduling literatures. We have discovered and characterized a novel scheduling problem called *"concurrent open shop scheduling with coupled resources"* and provided the first online heuristic. Using coflows, Varys improves application-level communication performance by more than $3\times$ in comparison to flow-level fairness and prioritization schemes. It can also meet $2\times$ more deadlines for data-parallel applications with latency constraints.

Both Orchestra and Varys assume that the characteristics of a coflow, i.e., the number or sizes of its flows, are known a priori. We soon learned that perfect knowledge is impossible in datacenter environments with failures being the norm. Application-level optimizations like speculative executions introduce even more unpredictability. We have recently proposed Aalo [3], a non-clairvoyant coflow scheduler, to cope with unforeseen events. It uses *current* observations to classify coflows into discrete priority queues. By scheduling similar coflows in each queue in the FIFO order and by prioritizing dissimilar coflows across queues, Aalo still approximates Varys.

The advantages of the coflow abstraction are not limited to performance improvements. It simplifies developing data-parallel applications and saves users from the frustrations of parameter tuning. Finally, because any flow is still a coflow with just one flow, my work applies to traditional client-server applications as well.

Application-Assisted Load Balancing: In the Sinbad [4] project, I leveraged application-level knowledge for load balancing the network. Traditional traffic engineering and load balancing techniques assume that the endpoints of any flow are fixed. We observed that distributed file systems do not constrain the destinations of their network transfers – they do not care where data replicas are located as long as they are replicated into multiple failure domains. Moreover, our analysis of production traces from Facebook and Microsoft showed that data replication contributes almost *half* the traffic in data-intensive clusters. By leveraging the flexibility in replica placement, Sinbad avoids network hotspots and decreases write latencies without affecting data availability or storage balance. As a collateral benefit of a more balanced network, coflow completion times improve as well.

Many of these approaches have been adopted in production deployments. For example, Orchestra ships as the default broadcast mechanism for Apache Spark that is used by hundreds of companies, and I spent several months at Facebook in 2013-14 to merge, test, and deploy Sinbad in their HDFS clusters.

Resource Allocation in Datacenters

Application-level information, when available, plays a crucial role in resource allocation for long-running services as well. For example, in private datacenters, we can exploit the internal structure of a service to better suit its demands; whereas, in public clouds, we often have to settle for some notion of fairness between coexisting tenants. Regardless of disparate requirements, constraints, and tradeoffs, I have relied on understanding and leveraging the unique characteristics of private, public, and virtual environments to develop practical solutions.

Large-scale Internet services run on thousands of machines with strict latency and fault tolerance requirements. However, simultaneously guaranteeing both is often infeasible – the former requires collocating machines, whereas the latter calls for spreading them out across many failure domains. By studying Bing's production datacenters, I observed that Bing services are composed of many individual components, each with widely varying latency and availability requirements. By exploiting component-level variations, we designed an optimization framework to improve Bing's latency and availability metrics [5]. Our solution went into production in 2012, and it was patented in 2013.

In the absence of tenants' intents, public cloud networks are shared in a best-effort manner. FairCloud [6] identified three essential attributes for fairly sharing such networks: *minimum bandwidth guarantee*, *high network utilization*, and *payment proportionality*. However, simultaneously achieving all three is, in fact, impossible. This result highlighted the need for a minimal interface that allows a tenant to express their requirements to the operator, resulting in a sequence of followups from cloud providers.

I have also worked on allocating virtual networks for tenants who are willing to explicitly provide their requirements. ViNEYard [7, 8] is an optimization framework to allocate entire networks with simultaneous node and link constraints. In last few years, many have used ViNEYard as a building block to allocate resources in different contexts including software-defined networks.

Future Research

I look forward to building upon my experience in optimizing communication, storage, and computing to realign application- and infrastructure-level goals for data analytics and web serving. Some areas of future work include:

Communication-First Big Data Systems: The amount of data we want to process is already growing faster than Moore's Law, and as we *personalize* and *connect* everything, it will keep accelerating for foreseeable future. Design decisions made in the last decade can hardly cope with this volume or velocity. We stand at the confluence of three hardware trends that all put communication at the center stage. First, *solid-state drives* strike a good balance between memory capacity and disk throughput, but a few of them can easily saturate 10GbE network interfaces. Second, *optical networks* are being proposed to achieve higher bisection bandwidth, but applications must exploit wavelength-locality to fully utilize the broadcast and multicast capabilities. Finally, *resource disaggregation* decouples the growth of compute and storage capacities, but it increases network utilization between consolidated resource banks.

I consider exploring communication-optimized analytics to be a top-priority challenge. Instead of retrofitting communication optimizations into current applications, we must design new ones whose data ingestion, storage, and computation are all planned for better suiting communication constraints. The challenges include *decreasing* the amount of data transfers through collocation and compression, *increasing* effective bandwidth by exploiting optical networks for replication and broadcasts, *spatial load balancing* using recently discovered techniques in coding theory (e.g., Product-Matrix codes), and *temporal network scheduling* extending my current work on coflows.

During my Ph.D., I have operated from the application layer to ensure deployability in legacy environments. Going forward, I want to rethink the I/O pipeline of operating systems as well as networking protocols (e.g., routing, in-network load balancing) that can *opportunistically* take advantage of application-level parallelism. Recent advances in software-defined networking and storage provide an ideal premise to pursue this direction.

Unified Resource Sharing: Resource sharing continues to be a challenging problem. Existing schedulers either focus on a specific type of similar resources or a specific objective for similar applications. Furthermore, they consider either spatial or temporal allocation. However, data-parallel applications have many bottlenecks that can change over time between resources. I believe that the solution lies with identifying a higher-level abstraction that reflects the *end-to-end progress* of an application. Even if finding one unified model proves difficult, our explorations will lead to pragmatic resource schedulers for dynamic environments and help us better understand relevant tradeoffs.

Wide-Area Analytics: Large organizations store data in datacenters closest to their users, but they often want to analyze these datasets as a whole. Wide-area bandwidth is the most constrained resource in this setting. It calls for decreasing usage through better query planning and judicious usage through better network scheduling, even more so than inside high-capacity datacenters. Higher latencies over the Internet do not help either: coordination, failure detection, or straggler mitigation, everything would take longer. I will expand my work on coflow scheduling and in-datacenter analytics to develop decentralized solutions for this lower-bandwidth, higher-latency environment to support both bulk offline processing and continuous online analytics.

Application-Aware Serving: Modern webpages consist of tens to hundreds of objects. Page load time – a key metric to measure user experience – depends on retrieving and rendering *all* the objects in the browser. Consequently, how browsers request and retrieve individual objects play a crucial role in improving user experience. SPDY and the upcoming HTTP/2 protocols take a step in the right direction by exposing opportunities to leverage page-level information through multiplexing, prioritization, and proactive server push. I will develop mechanisms for dependency-aware prioritization in browsers and predictive content-pushing from servers that work *in tandem* to improve user experience in diverse network conditions.

Behind the scenes, page creation time depends on the tail latencies of micro-tasks and communication between hundreds of machines inside datacenters. Instead of blindly optimizing the tail, request-aware tail optimization and speculation can lead to faster page creations. Doing this successfully under very tight deadlines will require efforts in distributed optimization and machine learning-based realtime prediction mechanisms.

Overall, I am excited about theoretical and practical challenges in large-scale networked systems. I believe that leveraging application-specific insights to build practical systems, while working toward identifying their fundamental principles, is a good rule of thumb to have impact in this tightly-connected world.

References

[1] **M. Chowdhury**, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica. Managing data transfers in computer clusters with Orchestra. In *ACM SIGCOMM*, pages 98–109, 2011.

[2] **M. Chowdhury**, Y. Zhong, and I. Stoica. Efficient coflow scheduling with Varys. In *ACM SIGCOMM*, pages 443–454, 2014.

[3] **M. Chowdhury** and I. Stoica. Coflow scheduling in the dark with Aalo. In submission, September 2014.

[4] **M. Chowdhury**, S. Kandula, and I. Stoica. Leveraging endpoint flexibility in data-intensive clusters. In *ACM SIGCOMM*, pages 231–242, 2013.

[5] P. Bodik, I. Menache, **M. Chowdhury**, P. Mani, D. Maltz, and I. Stoica. Surviving failures in bandwidth-constrained datacenters. In *ACM SIGCOMM*, pages 431–442, 2012.

[6] L. Popa, G. Kumar, **M. Chowdhury**, A. Krishnamurthy, S. Ratnasamy, and I. Stoica. FairCloud: Sharing the network in cloud computing. In *ACM SIGCOMM*, pages 187–198, 2012.

[7] **N. M. M. K. Chowdhury**, M. R. Rahman, and R. Boutaba. Virtual network embedding with coordinated node and link mapping. In *IEEE INFOCOM*, pages 783–791, 2009.

[8] **M. Chowdhury**, M. R. Rahman, and R. Boutaba. ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Transactions on Networking*, 20(1):206–219, 2012.

[9] M. Zaharia, **M. Chowdhury**, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *USENIX NSDI*, pages 15–28, 2012.