Research Statement
Sangeetha Abdu Jyothi

My research centers on the design and implementation of systems guided by theoretical principles, at the intersection of networking, cloud computing, and edge computing. The ever-growing number of cloud applications coupled with the exponential growth in real-time data they handle is intensifying resource demands in data centers (DCs). Applications such as deep learning training and inference, Augmented Reality and Virtual Reality (AR/VR), the back end for the Internet of Things (IoT) applications such as connected cars [1], connected robots [2], etc. are pushing the boundaries of cloud and edge data center capabilities. As infrastructure providers spend billions of dollars in capital and operational expenses for supporting the burgeoning number of users and applications, efficient usage of resources in cloud and edge data centers is increasingly critical. My goal is to improve performance and manageability in large-scale shared infrastructure to support diverse applications with varied performance needs. Towards this goal, I have worked on (a) quantifying the limits of performance achievable with a given set of resources [3–5], (b) identifying the right interfaces between applications and the resource manager [3, 6, 7] and (c) building automated systems for efficient resource utilization [3, 8].

In particular, my dissertation research focuses on automated resource management in large-scale networked systems. At a high level, these are closed-loop resource allocation systems that have three main modules: (i) inference of resource availability and requirements, (ii) scheduling for efficient resource allocation, and (iii) real-time monitoring and adaptation. My approach towards tackling challenges in large-scale resource management begins with the analysis of the infrastructure and workloads. Techniques used in this phase range from machine learning for understanding workload characteristics to theoretical analysis. In the next phase, I design resource allocation schemes tailored to the environment using optimization-theoretic formulations as well as fast and efficient heuristics. Finally, I evaluate the system with realistic workloads and using stress tests under extreme loads and failures.

I work closely with industry to gain insights into real-world systems, obtain realistic workloads, and to eventually have my solutions adopted in production systems. I am also keenly interested in open-sourcing my work to enable wider adoption. I have built two systems with the high-level goal of automated resource management in diverse environments. Morpheus [3] supports efficient closed-loop resource management in enterprise clusters. Morpheus [OSDI’16] was built in close collaboration with Microsoft Research and is available as open-source on Hadoop since release 2.9. Patronus [8] is designed for resource management across geo-distributed micro data centers separated by a Wide Area Network (WAN). On the theoretical front, I have explored various facets of scheduling to understand the fundamental limits of performance achievable on a given network/cluster/application framework. My work on understanding the limits of throughput performance on network topologies [SC’16] led to the design of a source-routed data center fabric [4] which supports near-optimal throughput in networks with limited switch memory. In the data center environment, my collaborators and I have also built a system for predicting network flow sizes and studied the impact of partial knowledge of flow sizes on the performance of schedulers [9]. In deep learning systems, I devised an efficient network scheduling scheme [10] that accelerates neural network training by exploiting the knowledge of the internal graph representation of the computational model. In future, I plan to (i) explore resource management for distributed edge computing beyond Patronus (§ 3.1), and (ii) address challenges in building fully automated clouds including design of interacting control loops and application-specific solutions (§ 3.2).

1 Automated Resource Management in Networked Systems

(i) Resource Management in Enterprise Clouds: Business-critical data analytics jobs constitute a significant fraction of load on multi-tenant enterprise clusters. While users expect these jobs to complete before an intended deadline, predicting the amount of resources needed to achieve this goal is cumbersome, a problem exacerbated by unpredictabilities in cluster resource availability and application behavior. Users typically mitigate this problem by over-allocation of resources, in turn resulting in poor cluster utilization for the provider. The key challenges in this environment include estimating the minimal amount of resources required to meet the user’s deadline and adapting the estimates in response to execution variance. Morpheus [3] tackles this problem using automated Service Level Objectives (SLOs) realized with a closed-loop composed of three components. First, the learning module adopts a data-driven approach for codifying implicit user requirements as SLOs through learning-based estimation of resource requirements of a job and its deadline using traces from the job’s past runs. Second, the reservation module employs a fast and efficient scheduling heuristic with provable performance guarantees to improve cluster utilization. Third, the dynamic reprovisioning module tracks real-time progress of a job and adapts allocation based on its progress to meet the job’s objectives, which also serves as input for the learning module. With tight resource provisioning and fast reprovisioning using closed-loop control, Morpheus reduced the number of deadline violations by 5x to 13x while retaining cluster utilization, and lowering cluster size requirement by 14% to 28% in Microsoft clusters.

(ii) Resource Management Across Geo-Distributed Micro Data Centers: Driven by the need for compute in close proximity to users for supporting low-latency applications, Micro Data Centers (MDCs), each with limited resources (tens or hundreds of servers), are emerging as prominent components in the Internet infrastructure. Cellular providers
are converting traditional Central Offices into MDCs with commodity servers for supporting virtualized cellular functions (cellular control plane latency limit is 50ms in 4G and 1ms in 5G). MDCs are also critical for offering support to the Internet of Thing (IoT) applications such as video processing at the edge, connected cars, AR/VR, etc. We refer to this emerging environment of MDCs interconnected by Wide Area Network as a WAND (WAN As a Network of Data Centers). In a WAND, the MDCs enjoy limited benefits of statistical multiplexing due to their small sizes. The solutions proposed for traditional clouds with high-performance network topologies cannot support bandwidth-intensive and/or ultra-low latency geo-distributed applications in this multi-resource environment composed of server resources across MDCs and network resources in WAN with varying bandwidth and latency characteristics. Hence, we need efficient resource management with a global perspective in WAND.

We designed Patronus [8], a centralized controller, for fast and efficient resource management in private WANDs. To handle the complexity of scheduling a wide range of application requirements (latency, throughput, deadline, etc.), we divided the optimization into two phases temporally — long-term and instantaneous. The slower long-term scheduler handles complex constraints such as deadlines. The faster instantaneous scheduler adapts to real-time variations and schedules a subset of tasks deemed active in an instant by the long-term scheduler. Patronus can simultaneously schedule batch and streaming jobs using a technique called hierarchical optimization, where multiple objectives in the same Linear Program (LP) with different priorities, weights, and tolerance levels enable support for diverse applications and give scheduling flexibility to the provider. We evaluated Patronus with a realistic cellular provider back-end WAND topology and workloads. This environment has a wide range of private applications — cellular service chains, batch processing on cellular logs, VPN service, connected car service, etc. — across different priority classes. Adaptive scheduling with the division of labor across two schedulers allowed Patronus to reduce peak MDC utilization by up to 47% while improving tail latency of latency-critical applications by up to 85% in a realistic cellular WAND.

2 Other Related Research

(i) Network Scheduling in Deep Learning Systems: Distributed Deep Neural networks (DNN) training is a prominent cloud application today with gigabytes of data transferred in each iteration which lasts milliseconds. For users on commercial cloud environments with limited bandwidth, we identified an opportunity [6] for accelerating distributed training through network scheduling [SysML’18]. The computational model in popular frameworks such as TensorFlow and PyTorch is a Directed Acyclic Graph (DAG), where each node represents a computation operation. The network transfers are the parameter reads and writes (represented by root nodes) in the model. Not all parameters are consumed by the model at the same instant. Today, the parameters are read randomly from a parameter server without considering the order in which they are operated on, leading to blocking on computation. However, determining the optimal network schedule across models is a non-trivial task. In TicTac [10], we designed a scheme that can determine the optimal schedule of parameter transfers for any DNN model to accelerate its training. Using TicTac on TensorFlow, we accelerated the training by up to 19% in commonly used models in realistic settings.

(ii) High-Performance Networks in the Cloud: We have explored two key directions: (a) understanding the throughput performance of DC network topologies, and (b) network flow prediction and its impact on network scheduling. (a) Understanding topology performance: To understand the worst-case throughput of an application in a dynamic environment such as multi-tenant DC, we need a systematic methodology for measuring throughput. We showed that the commonly-used metric, bisection bandwidth, can result in erroneous estimates since cut-based metrics can be off from the actual worst-case throughput by \(O(\log n)\). To tackle this problem, we built a framework to measure throughput using Linear Program (LP) formulation of the multi-commodity flow problem and developed a heuristic to measure the worst-case — a provably near-worst-case traffic matrix [SC’16]. Using this framework, we performed a head-to-head comparison of topologies [9] and found that expander topologies are more robust to traffic variations compared with structured topologies and randomization of flow placement at the top-of-rack switches can improve throughput performance in real-world Clos DC networks. Building on these insights, we proposed a flexible, high-performance data center fabric which relies on source routing [4]. (b) Predicting network flows: DC scheduling techniques [11][14] rely on precise flow size information for efficient network scheduling. However, accurate estimation of flow sizes is a complex task. We developed Flux [9], a framework which uses system-level heuristics and learning methods to predict flow sizes accurately. Using Flux, we showed that network scheduling efficiency can be improved even with partial knowledge of flow sizes.

3 Future Directions

3.1 Resource Management for Distributed Edge Computing

Patronus [8] is the first step in the uncharted territory of managing resources across geo-distributed MDCs and several challenges remain open in generic WAND resource management.

(i) Distributed Controllers in WAND: Efficiency in WAND resource management depends on the accuracy of resource requirement estimation and the rate of adaptability of the scheduling mechanism. As applications move towards a mi-
croservice architecture with a large number of loosely coupled components which may be geo-distributed, augmenting
the centralized controller in Patronus with distributed controllers is critical for ultra-low latency applications that re-
quire millisecond-scale scheduling response time. Since components of an application may be handled by multiple
controllers, stale information can lead to instabilities. The key technical challenges involve geographical and temporal
partitioning of scheduling across multiple controllers and the design of mechanisms to ensure system stability.

(ii) WAND Service Interface: Today’s cloud providers offer VMs with capabilities specified in terms of CPU, GPU,
network, storage, etc. over a few regions. In a WAND, defining the unit of service is more cumbersome. A WAND
provider will typically have macro- and micro-data centers with different compute capabilities and latency profiles
between them. The cost and performance using the same amount of resource may vary widely depending on the size
and location of the MDCs and within the same MDC across time due to limited statistical multiplexing. In this context,
we need to devise new models for WAND service interface (similar to cloud instance). This can be addressed in one
of the following ways: (i) using a game-theoretic approach to determine time-varying instance pricing across multiple
MDCs (building on spot pricing in the cloud with additional WAND-specific attributes), or (ii) developing a WAND
platform where the users only specify (and are charged for) the performance requirements of the application and the
provider adapts resources across multiple MDCs dynamically.

3.2 Towards Fully Automated Clouds

A fully automated cloud should allow a naive user to submit a job with high-level performance objectives and be
capable of understanding the job dependencies and requirements across various resources (CPU, memory, network,
etc.) to meet the user objectives. While the community has taken preliminary steps in this direction, several challenges
remain open. I identify a few key directions towards building fully automated clouds broadly spanning application-
controller interfaces, the composition of multiple closed-loops, and application-specific solutions.

(i) Composing Closed-Loop Control over Multiple Control Functions: Today, cloud resource management involves
multiple controllers — a cluster scheduler that handles server resources, a network controller responsible for traffic en-
ingineering, a cloud monitoring module dealing with failure detection, and so on. Prior work showed that this separation
of control affects the overall performance of applications as the controllers are unaware of resource inter-dependencies
(e.g., server-network inter-dependency of application load [15]). The current trend of independent automation over
each controller does not address this problem, and may even exacerbate it by introducing instabilities in the system. My
goal is to design interacting closed-loop controls that can achieve high resource efficiency while maintaining stability. I intend
to address this challenge with carefully designed feedback mechanisms between multiple control loops, each handling
a different control function (e.g., network and server control loops sharing data for learning inter-dependencies while
scheduling independently). As a first step, I will focus on bridging the network-server scheduling divide, i.e., integrating
cluster scheduler and network controller automation.

(ii) Extending Resource Management to Programmable Switches: Distributed applications can now leverage limited
compute and memory in programmable switches to improve performance [16,17]. This extension of applications
into the network switches introduces a new dimension in cloud resource management. How do we design automated
resource management over server resources, network links, and switch resources to support diverse applications with in-
network components? Challenges in this space include characterization of application requirements across multiple
resource types, trade-offs across various dimensions (e.g., an application may either use only server and network heavily
or use switch resources and drastically reduce network load), and design of schedulers that can handle these trade-offs.

(iii) Learning-Based Application-Controller Interface for the Cloud: To build a fully automated cloud, we need to
first determine the right interface between applications and the resource controller. Among a set of correlated flows,
some flows may have a higher priority than others (e.g., priorities across parameter transfers in distributed deep learn-
ing [12(i)]), the lower priority of large images in a webpage load, etc.) Current abstractions such as Coflow [13] fail to con-
vey the application intent about the network flows to the controller. As a first step towards bridging this gap, we devise
the CadentFlow network abstraction [7] for cloud applications with per-flow metrics (weights/priorities/deadlines)
and an application-level objective. In future, I plan to extend the abstraction with automated application intent ap-
praiser using learning techniques that can predict the dependencies across flows using system-level cues such as the
time when data is consumed/written from/to the disk or memory.

(iv) Accelerating Deep-Learning with Automated Scheduling: Deep learning frameworks such as TensorFlow are
currently incorporating decentralized parameter aggregation to address the incast problem. Building on insights from
TicTac [19], we plan to extend model-aware network scheduling to decentralized aggregation techniques such as all-
reduce. Scheduling is challenging with decentralized aggregation due to the synchronized nature of network transfers
across multiple workers. The automated scheduling design should determine the order of network transfers and incor-
porate additional mechanisms for tackling stragglers using techniques such as adaptive worker loads.

As the global infrastructure and the gamut of applications diversify and grow, I am excited about contributing
towards their efficiency and cost-effectiveness by building systems with provable performance guarantees.
References