Yue Cheng Research Scholarship Statement

I request consideration for promotion to Associate Professor with tenure in the School of Data Science at the University of Virginia (UVA). This statement summarizes my research contributions to the field of HPC- and cloud-scale systems and data-intensive computing during my last five years at George Mason University (GMU) and my agenda for continued scholarship productivity at UVA for years to come.

1 Background

I am an assistant professor with a primary appointment in the School of Data Science and a joint appointment in the Department of Computer Science (CS) in the School of Engineering and Applied Science at the University of Virginia. I joined UVA as a tenure-track assistant professor in August 2022. Prior to UVA, I was a tenure-track assistant professor in Computer Science at George Mason University from August 2017 to August 2022. I received my Ph.D. degree in Computer Science from Virginia Tech in 2017. My Ph.D. dissertation was adviced by Prof. Ali R. Butt and focused on improving the performance and flexibility of large-scale, distributed storage systems. I graduated from Beijing University of Posts and Telecommunications (BUPT) in Beijing, China with a Bachelor of Engineering degree in Computer Science in 2009.

2 Research Scholarship Statement

2.1 Scaling Large Computations on Serverless Cloud: Application Frameworks, State Management, and Infrastructures

A primary focus of my current research is on solving challenging, real-world problems in serverless cloud computing systems. With an end-to-end approach, my research looks into each layer of the serverless system stack, including application frameworks, FaaS (Functions-as-a-Service) platforms, and OSes. My work in this area has shown impact outside of academia. For example, FAASNET *has been incorporated in Alibaba Cloud Function Compute, accelerating millions of function cold starts on a daily basis*.

2.1.1 [Application Frameworks] Scalable Serverless Data Analytics and Parallel Computing

Executing large-scale DAG (directed acyclic graph) workflows on serverless platforms is appealing, as serverless providers promise to offer transparent auto-scaling. Existing serverless parallel computing frameworks borrow scheduling techniques used in conventional, serverful cluster computing frameworks such as MapReduce/Spark: A centralized scheduler dispatches tasks to long-running task workers running as a cluster of servers. Such designs, however, are not well suited for serverless platforms, since function invocation incurs non-trivial scheduling overhead and each function is stateless and resource constrained. This causes application performance bottleneck and high data movement cost.

WUKONG [SoCC'20, Wukong Project] redesigns serverless parallel computing frameworks by using decentralized scheduling, a highly scalable scheduling approach that naturally unleashes the power of FaaS elasticity. Decentralized scheduling resolves the mismatch between serverless parallel workflow applications and conventional cluster computing architectures. Partitioning the work of a centralized scheduler (i.e., tracking task completions, identifying and dispatching ready tasks, etc.) across a large number of serverless functions naturally exploits the auto-scaling property of FaaS platform.

Scheduling in WUKONG uses a combination of static and dynamic scheduling. WUKONG partitions a DAG into multiple, possibly overlapping, subgraphs. A static schedule is a subgraph of the DAG. Each subgraph is assigned to a serverless executor that is running inside of a serverless function. An executor uses dynamic scheduling to enforce data dependencies of the tasks in its static schedule. An executor can invoke additional functions to increase task parallelism, or collapse tasks to eliminate any communication delay between them. Executors coordinate at DAG fan-ins and fan-outs during dynamic scheduling. Evaluation shows that WUKONG achieves up to 68× higher performance compared to state-of-the-art serverless frameworks while reducing the monetary cost by 93%. *More importantly,* WUKONG *removes users' burden of framework configurations, which is notoriously challenging for users who lack distributed systems background. This, lowers the bar of parallel computing for a broader set of users and makes parallel computing effectively "server-less" and user-friendly.*

The original idea was proposed by me. My research group takes all credit for this research.

2.1.2 [Big Data Management] Cost-Effective Cloud Storage using Serverless Functions

Data-intensive applications extensively use cloud memory caching for performance. A wide range of such applications repetitively read large data objects with sizes ranging from a few MBs to several GBs. Examples include container registries and big data analytics. Large, intermittent I/Os require huge cache capacity and high read throughput, which existing cloud memory caching services are too expensive to serve: they are designed for small-I/O-intensive, latency-sensitive workloads.

INFINICACHE [FAST'20, InfiniCache Project] addresses this problem by exploiting serverless functions as a cost-effective, yet performant storage medium. A serverless application is structured as a collection of serverless functions. A function has memory that can be used to store objects that are needed during its execution. The key idea of INFINICACHE is to use this collective, distributed memory to store cached objects. Under INFINICACHE, functions are invoked by a user application to access the cached objects. FaaS provider caches invoked functions and their state so in-memory objects are retained between function invocations. Providers charge users only when a function is invoked, in our case, when a cached object is accessed. INFINICACHE significantly reduces the monetary cost of memory capacity compared to today's in-memory caching services such as AWS ElastiCache (by up to two orders of magnitude).

Conventional wisdom holds that it is impractical to support stateful storage services directly on ephemeral serverless functions. Contrary to common beliefs, we have demonstrated, for the first time, that the FaaS model has great potential of supporting data-intensive, stateful backend services. I believe that harvesting collective, ephemeral resources in form of serverless functions is a missing piece that enables new, serverless BaaS (Backend-as-a-Service). INFINICACHE provides continuous availability and fault tolerance by using an efficient primary-backup mechanism; this mechanism pairs a function and a clone instance of itself for periodic delta-sync and leverages FaaS auto-scaling to provide automatic failover in face of a function failure. INFINICACHE stores object chunks in separate functions using erasure coding to (1) enhance data availability, (2) parallelize I/Os for high aggregate throughput, (3) tolerate straggling functions with redundancy.

In general, INFINICACHE is well suited for cloud-native, data-intensive applications that need a large, intermediate storage. For example, INFINICACHE could provide a drop-in solution for caching intermediate data of big data analytics applications such as Spark. Companies including BigStream have expressed interests in adopting INFINICACHE. INFINICACHE could also leverage the processing power of serverless functions to enable in-situ computations. This will open doors to new, stateful serverless applications in a broader range of disciplines. I have received an NSF CAREER Award [NSF CAREER] to support this research and I am very excited to explore these new directions in the near term.

I came up with the whole idea of serverless storage, and the whole project was designed and developed by my Ph.D. students.

2.1.3 [Platforms] Scalable Provisioning of Custom Serverless Container Runtimes

Custom serverless container support is gaining traction as it enables better control over OSes and tooling for modernizing FaaS applications (e.g., dependency-heavy ML applications, data analytics, etc.). However, providing rapid container provisioning introduces non-trivial challenges for FaaS providers, as containers are often huge in size and real-world FaaS workloads are highly dynamic.

FAASNET [ATC'21, FaaSNet Project] eliminates the scalability bottlenecks in custom serverless container provisioning. The key idea is to decentralize container provisioning to all participating function environments—a virtual machine (VM) or a bare-metal machine—which are organized in function tree structures. A *function tree* (FT) is a logical, tree-based network overlay. A FT consists of multiple host VMs and allows provisioning of container runtimes or code packages to be decentralized in a scalable manner. FaaSNet is highly adaptive using a tree balancing algorithm that dynamically adapts the FT topology to accommodate dynamic VM joining and leaving.

The design of FaaSNet is driven by the needs of custom-container-based FaaS applications from a large cloud provider's FaaS platform, Alibaba Cloud Function Compute. However, FaaSNet is generally applicable to other open platforms as well, e.g., OpenWhisk and Kubernetes. A key design principle of FaaSNet is to separate control and data planes. This has two benefits. First, it naturally fits Alibaba Function Compute's architecture, which uses lots of resource-constrained VMs to hold function containers; a VM would have become a bottleneck if it were to run FT's control plane on it; this also rules out existing provisioning solutions, which use extra, dedicated, centralized components for data seeding, metadata management, and coordination. Second, by separating control and data paths, FaaSNet can be integrated as a plugin to an existing FaaS scheduler, thus offering a general solution to a broader set of applications including large data sharing on Kubernetes. After I published it at USENIX ATC 2021, FaaSNet *was integrated as part of the next-generation FaaS infrastructure of Alibaba Function Compute, enabling second-level function container provisioning at scale.*

I had been deeply involved from the very beginning and contributed to the design, implementation, and evaluation of this idea.

2.2 Building Scalable and High-Performance Machine Learning Systems

Another focus of my current research is on designing scalable and high-performance ML systems. My research in this area rethinks the design of federated learning (FL) systems [HPDC'20, SC'21] and distributed deep learning (DDL) systems [ICDM'20].

2.2.1 High-Performance Federated Learning Systems

Traditional, distributed machine learning requires the training dataset to be located in a common, central location accessible to trusted training parties. However, privacy concerns and legislations such as General Data Protection Regulation (GDPR) inhibit transmitting data to a central location. This makes training a high-quality machine learning model impossible. Federated learning (FL) has emerged as an alternative way to perform collaborative model training without requiring training parties to share their private data. In FL, each data party or owner, maintains its own data locally while engaging in collaborative training. Model updates are shared, aggregated, and broadcast by a central aggregator. In FL, data parties may have highly heterogeneous hardware capacity and training data. Such heterogeneity creates bottlenecks in both training time and model performance.

TiFL [HPDC'20] tackles the resource and data heterogeneity challenge via a new, tiered design. TiFL logically divides data parties into tiers based on their training time and selects parties from the same tier in each training round. This solves the straggler issues caused by imbalanced hardware capacity and training data sizes. TiFL uses a dynamic tier selection approach to make sure that parties from different tiers can all participate in training. This way, TiFL is data heterogeneity-aware and strikes a balance between training time and model accuracy.

Asynchronous FL training is straggler resilient, as the central aggregator does not need to wait for the stragglers before updating other waiting parties. However, in asynchronous FL training, all data parties can communicate with the aggregator at any time and thus introduce communication bottleneck at the aggregator. To solve this problem, I design FedAT [SC'21], a tiered FL system that supports asynchronous FL training. FedAT inherits the tiered design from TiFL [HPDC'20], but goes beyond TiFL in that FedAT combines synchronous, intra-tier training and asynchronous, cross-tier training. This way, FedAT minimizes the straggler effect with improved convergence speed and test accuracy. FedAT compresses uplink and downlink communications using an efficient, polyline-encoding-based compression algorithm to minimize the communication cost.

My research in FL systems has been integrated in IBM's Federated Learning Framework, which is used to solve critical, real-world problems such as anti-money laundering, fraud detection by banking industry, and improving diagnostics and treatment design by medical research facilities. I am excited to see applications of FL technologies in these critical domains and am enthusiastic to collaborate with domain scientists on applying FL to new application scenarios. For example, I have secured a collaborative NSF FMSG grant [NSF FMSG] to support new FL applications in distributed additive manufacturing.

This was a collaborative project and my research group (my Ph.D. student and I), as the project lead, takes a significant portion of credit.

2.2.2 Scalable Deep Learning Systems

Alternating Direction Method of Multipliers (ADMM) has recently emerged as a potential alternative optimizer to Stochastic Gradient Descent (SGD) for deep learning. This is because ADMM can solve gradient vanishing and poor conditioning problems. However, ADMM training shows poor parallelism due to a high level of layer dependencies among variables. I co-design pdADMM [ICDM'20] that addresses the scalability problem of ADMM-based deep neural network (DNN) training. pdADMM uses a novel reformulation of feed-forward problem in ADMM by splitting a DNN into independent layer partitions; this way, parameters in each layer can be updated in parallel in order to speed up the training process with proved convergence rate. This naturally enables model parallelism for ADMM. This research is supported by an NSF OAC grant [NSF OAC] and an Amazon Research Award [Amazon Research Award].

Million-node, billion-edge graphs are prevalent in today's graph-based applications. Despite the success of Graph Neural Networks (GNNs), it remains a grand challenge to train GNNs on those large graphs. I propose a new disaggregated architecture that manages the training (compute) and embedding data (storage) separately and independently. The new method partitions a large graph into subgraphs and uses historical embeddings to train subgraphs in parallel. This research has received a Meta Research Award for AI System Hardware/Software Codesign [Meta Research Award].

I am also co-leading a project where I adopt a hardware-software codesign to enable scalable and high-performance, SGD-based, data-parallel DNN training. The key idea is to leverage a novel hardware-based compression technique to design a distributed, OS-level data cache that effectively exploits the physical memory freed via program-memory compression. This research has received an NSF SPX grant [NSF SPX].

2.3 Data-intensive Systems and Data Storage Systems

My Ph.D. dissertation research had focused on solving challenges in data-intensive and storage systems targeting three main application scenarios: (1) massive-scale web applications that need efficient and scalable distributed storage systems; (2) enterprise server applications that demand intelligent caching systems; and (3) cloud-native big data analytics applications that require careful deployment planning.

MBal [EuroSys'15] improves the performance of distributed memory caching for web applications. MBal performs fast, lockless PUT and GET operations via partitioned compute and memory resources called *cachelets*. MBal quickly detects presence of hotspots (i.e., overload) in the workloads and uses an adaptive, multi-phase load balancing approach to mitigate load imbalance.

While there are many flash caching solutions, their effectiveness is difficult to judge due to lack of a "best case" baseline. Belady's MIN, the go-to offline optimal caching algorithm, considers read hit ratio but not flash endurance. I designed the first offline flash caching heuristic [ATC'16] that minimizes erasures with optimal read hit ratio. My investigation provides a useful approximate baseline for evaluating any online flash caching algorithms.

I co-designed BespoKV, the first *serverless storage framework* [SC'18], to make it easier to develop and debug distributed storage systems. BespoKV abstracts away redundant components that handle distributed system management tasks such as fault tolerance, network topology management, and consistency. Using BespoKV, developers only need to focus on the development of core storage functions (such as read and write); BespoKV will convert provided storage functions into a scalable, highly configurable, and distributed storage deployment, following a serverless programming model.

Finally, I presented the first characterization of a large-scale, production Docker registry workload [FAST'18]. This study reveals insights about optimizing cloud object storage systems and container registries and my workload trace datasets have been widely used by researchers across the globe.

2.4 Future Research Directions

Serverless computing and machine learning are still rapidly changing areas. I believe that the fundamental themes of my research will only grow in importance as the areas mature. Looking further forward, my interests extend to the following directions: designing easy-to-use serverless supercomputer; inventing new abstractions for self-managed FaaS infrastructure; exploring ML for systems and systems for ML; and enhancing security and privacy for user data.

2.4.1 Easy-to-Use Serverless Supercomputing for All

With systems like WUKONG [SoCC'20, Wukong Project], we have addressed many of the challenges of executing large-scale, asymmetric DAG workflows. However, accelerators such as GPU are still not first-class citizens in today's serverless computing. Yet, there are huge demands from domain scientists and researchers for having one-click deployment and execution of large computations on massive, elastically-scaled resources. I am currently collaborating with Adobe Research on designing the first serverless GPU infrastructure for interactive ML training workloads. This work is a step towards the broader goal of supporting serverless, heterogeneous computing. Many systems challenges remain: How can we design language-integrated programming models to make the life easier for users to program massive, disaggregated cloud resources? How can we build efficient scheduling algorithms for large-scale data science and scientific computing applications on heterogeneous CPU and GPU resources? How can we support fast data storage and network communication across cloud functions? I will explore these directions and hope to collaborate with domain experts to investigate intersections of serverless supercomputing with programming languages, frameworks, and OSes, as well as interesting application use-cases in the near term (the coming 3-4 years).

2.4.2 Data-Driven, Autonomous FaaS Infrastructure

Today's FaaS is still in its infancy. Frontend FaaS offerings require users to decide on almost every configuration option such as function resources and concurrency levels. Backend FaaS resource scheduling and allocation heavily rely on existing cloud infrastructure, e.g., the VM infrastructure, which are not designed for FaaS. The whole FaaS infrastructure badly needs new cross-cutting tools and data-driven approaches to enable holistic, self-managed, autonomous services for FaaS users and providers. Such tools would offer users optimal configuration options for their applications. Better visualization tools would help ease the infrastructure management by exposing observability and monitoring as first-class citizen. A compelling direction that I am interested in pursuing in the long run is designing data-driven approaches to automating allocation of backend resources such as function caches. I will leverage my collaborations with cloud providers such as IBM and Alibaba to achieve these goals.

2.4.3 ML/DS for Systems and Systems for ML/DS

My prior and current research on systems software has inspired me to explore effective ML techniques that help solve systems challenges. For example, my offline flash caching work [ATC'16] has demonstrated huge opportunity space for improving flash endurance; however, an optimal flash caching algorithm is in theory intractable and designing efficient, online flash caching algorithm remains a challenging, manual task. I am interested in exploring ML approaches to learn a flash caching policy that optimizes both performance and endurance. Furthermore, emerging ML applications such as Federated Learning and Graph Neural Networks (GNNs) require new systems innovations. In particular, I am interested in exploring new ways to scale these applications. One direction that I am currently working on is to exploit serverless computing to achieve automatic scaling for these applications.

3 Concluding Remarks

This statement summarizes my accomplishments in research, teaching, service, and diversity promotion as an assistant professor for the last five years. My research has made significant contributions to the field of big data processing systems, cloud computing, and data storage systems. My work has been consistently published at top-tier conferences and has over 906 citations. I have developed an independent research program with 7 Ph.D. students (4 at UVA and 3 at GMU) and 5 NSF research grants. I am the recipient of a number of highly competitive awards, including an **NSF CAREER award** (2021), an **Amazon Research Award** (2021), a **Meta Research Award** (2022), and an **IEEE CS TCHPC Early Career Researchers Award for Excellence in High Performance Computing** (2022). With a carefully planned research and teaching agenda towards the joint domains of Data Science and Computer Science, I am committed to continued excellence in scholarship development and data science and data systems research for years to come.

Selected Publications and Grants

[FAST'20] Ao Wang, Jingyuan Zhang, Xiaolong Ma, Ali Anwar, Lukas Rupprecht, Dimitrios Skourtis, Vasily Tarasov, Feng Yan, **Yue Cheng**. INFINICACHE*: Exploiting Ephemeral Serverless Functions to Build a Cost-Effective Memory Cache*, In USENIX FAST '20.

[NSF CAREER] *CNS-2045680: CAREER: Harnessing Serverless Functions to Build Highly Elastic Cloud Storage Infrastructure*, https://www.nsf.gov/awardsearch/showAward?AWD_ID=2045680, 2021.

[SoCC'20] Benjamin Carver, Jingyuan Zhang, Ao Wang, Ali Anwar, Panruo Wu, **Yue Cheng**. WUKONG*: A Scalable and Locality-Enhanced Framework for Serverless Parallel Computing*, In ACM SoCC '20.

[ATC'21] Ao Wang, Shuai Chang, Huangshi Tian, Hongqi Wang, Haoran Yang, Huiba Li, Rui Du, **Yue Cheng**. FAASNET*: Scalable and Fast Provisioning of Custom Serverless Container Runtimes at Alibaba Cloud Function Compute*, In USENIX ATC '21.

[SoCC'21] Li Liu, Haoliang Wang, An Wang, Mengbai Xiao, **Yue Cheng**, Songqing Chen. *Mind the Gap: Broken Promises of CPU Reservations in Containerized Multi-tenant Clouds*, In ACM SoCC '21.

[SC'22] Yuqi Fu, Li Liu, Haoliang Wang, **Yue Cheng**, Songqing Chen. *SFS: Smart OS Scheduling for Serverless Functions*, In SC '22.

[HPDC'20] Zheng Chai, Ahsan Ali, Syed Zawad, Stacey Truex, Ali Anwar, Nathalie Baracaldo, Yi Zhou, Heiko Ludwig, Feng Yan, **Yue Cheng**. TIFL*: A Tier-based Federated Learning System*, In ACM HPDC '20.

[SC'21] Zheng Chai, Yujing Chen, Ali Anwar, Liang Zhao, **Yue Cheng**, Huzefa Rangwala. *FedAT: A High-Performance and Communication-Efficient Federated Learning System with Asynchronous Tiers*, In SC '21.

[NSF FMSG] *CMMI-2134689: FMSG: Cyber: Federated Deep Learning for Future Ubiquitous Distributed Additive Manufacturing*, https://www.nsf.gov/awardsearch/showAward?AWD_ID=2134689, 2021.

[ICDM'20] Junxiang Wang, Zheng Chai, **Yue Cheng**, Liang Zhao. *Toward Model Parallelism for Deep Neural Network based on Gradient-free ADMM Framework*, In IEEE ICDM '20.

[NSF OAC] *OAC-2007976: OAC Core: SMALL: DeepJIMU: Model-Parallelism Infrastructure for Large-scale Deep Learning by Gradient-Free Optimization*, https://www.nsf.gov/awardsearch/showAward?AWD_ID=2007976, 2020.

[Amazon Research Award] *Amazon Research Award: Distributed Large-scale Graph Deep Learning by Gradient-free Optimization*, 2020.

[Meta Research Award] *Serverless and Scalable GNN Training with Disaggregated Compute and Storage*, https://research.facebook.com/research-awards/2022-request-for-research-proposals-for-ai-system-hardware-software-codesign/, 2022.

[NSF SPX] *CCF-1919075: SPX: Collaborative Research: Cross-stack Memory Optimizations for Boosting I/O Performance of Deep Learning HPC Applications*, https://www.nsf.gov/awardsearch/showAward?AWD_ID=1919075, 2019.

[EuroSys'15] **Yue Cheng**, Aayush Gupta, Ali R. Butt. *An In-Memory Object Caching Framework with Adaptive Load Balancing*, In ACM EuroSys '15.

[HPDC'15] **Yue Cheng**, M. Safdar Iqbal, Aayush Gupta, Ali R. Butt. *CAST: Tiering Storage for Data Analytics in the Cloud*, In ACM HPDC '15.

[ATC'16] **Yue Cheng**, Fred Douglis, Philip Shilane, Michael Trachtman, Grant Wallace, Peter Desnoyers, Kai Li. *Erasing Belady's Limitations: In Search of Flash Cache Offline Optimality*, In USENIX ATC '16.

[FAST'18] Ali Anwar, Mohamed Mohamed, Vasily Tarasov, Michael Littley, Lukas Rupprecht, **Yue Cheng**, Nannan Zhao, Dimitrios Skourtis, Amit S. Warke, Heiko Ludwig, Dean Hildebrand, Ali R. Butt. *Improving Docker Registry Design based on Production Workload Analysis*, In USENIX FAST '18.

[SC'18] Ali Anwar, **Yue Cheng**, Hai Huang, Jingoo Han, Hyogi Sim, Dongyoon Lee, Fred Douglis, Ali R. Butt. *BESPOKV: Application Tailored Scale-Out Key-Value Stores*, In SC '18.

[InfiniCache Project] `https://ds2-lab.github.io/infinicache/`.

[Wukong Project] `https://ds2-lab.github.io/Wukong/`.

[FaaSNet Project] `https://github.com/ds2-lab/FaaSNet`.

[SFS Project] `https://github.com/ds2-lab/SFS`.