Yang Zhou

Curriculum Vitae

SEC 4.429, 150 Western Ave, Allston, MA, 02134
(+1) 617 599 8532
yangzhou@g.harvard.edu
https://yangzhou1997.github.io/

Education

Ph.D. **Computer Science, Harvard University**, *Cambridge, MA, USA*, 2018–Present.
Advisor: Professor Minlan Yu and James Mickens

B.S. **Computer Science, Peking University**, *Bejing, China*, 2014–2018.
Advisor: Professor Tong Yang

Publications

2022 **Yang Zhou**, Ying Zhang, Minlan Yu, Guangyu Wang, Dexter Cao, Eric Sung, and Starsky Wong. Evolvable Network Telemetry at Facebook. NSDI 2022.

2019 **Yang Zhou**, Varun Gandhi, Mark Wilkening, James Mickens, and Minlan Yu. NFShield: Securing NIC-Accelerated Network Functions in the Cloud. Pending submission.

2018 Tong Yang, Jie Jiang, Peng Liu, Qun Huang, Junzhi Gong, **Yang Zhou**, Rui Miao, Xiaoming Li, and Steve Uhlig. Elastic Sketch: Adaptive and Fast Network-Wide Measurements. SIGCOMM 2018.

2018 **Yang Zhou**, Tong Yang, Jie Jiang, Bin Cui, Minlan Yu, Xiaoming Li, and Steve Uhlig. Cold Filter: A Meta-Framework for Faster and More Accurate Stream Processing. SIGMOD 2018.

2018 **Yang Zhou**, Hao Jin, Peng Liu, Haowei Zhang, Tong Yang, and Xiaoming Li. Accurate Per-Flow Measurement with Bloom Sketch. IEEE INFOCOM Workshops (INFOCOM WKSHPS) 2018.

2018 **Yang Zhou**, Omid Alipourfard, Minlan Yu, and Tong Yang. Accelerating Network Measurement in Software. SIGCOMM CCR 2018.

2018 Xiangyang Gou, Chenxingyu Zhao, Tong Yang, Lei Zou, **Yang Zhou**, Yibo Yan, Xiaoming Li, and Bin Cui. Single Hash: Use One Hash Function to Build Faster Hash Based Data Structures. IEEE International Conference on Big Data and Smart Computing (BigComp) 2018.

2018 Omid Alipourfard, Masoud Moshref, **Yang Zhou**, Tong Yang, and Minlan Yu. A Comparison of Performance and Accuracy of Measurement Algorithms in Software. ACM Symposium on SDN Research (SOSR) 2018.

2017 Tong Yang, **Yang Zhou**, Hao Jin, Shigang Chen, and Xiaoming Li. Pyramid Sketch: A Sketch Framework for Frequency Estimation of Data Streams. VLDB 2017.

2017 **Yang Zhou**, Peng Liu, Hao Jin, Tong Yang, Shoujiang Dang, and Xiaoming Li. One Memory Access Sketch: A More Accurate and Faster Sketch for Per-Flow Measurement. IEEE Global Communications Conference (Globecom) 2017.

2017 Junzhi Gong, Tong Yang, **Yang Zhou**, Dongsheng Yang, Shigang Chen, Bin Cui, and Xiaoming Li. ABC: A Practicable Sketch Framework for Non-Uniform Multisets. IEEE International Conference on Big Data 2017.

Work Experience

09/2021–Present **Student Researcher**, *Google*, CA, Mentor: Hassan Wassel.
- Continue working on the fault-tolerant remote memory.

06/2021–09/2021 **Research Intern**, *Google*, CA, Mentor: Hank Levy and Hassan Wassel.

- Work on fault tolerance for application-integrated far memory system.
- Propose a span-based erasure coding scheme to encode different sizes of objects.
- Design a RMA-based data-parity consistency protocol for swapping out spans.

07/2020–09/2020 **Research Intern**, *VMware Research*, CA, Mentor: Israel Cidon and Christos Karamanolis.

- Accelerate geo-distributed data analytics and save WAN traffic cost, by applying traffic redundancy elimination (TRE) technique to data analytics jobs.
- Hack Alluxio, an in-memory data cache platform, to enable TRE when sending data across different data centers.

11/2019–05/2020 **Research Collaborator**, *Facebook*, CA, Mentor: Ying Zhang.

- Work on Facebook change-aware network telemetry system.
- Layering design along the data collection pipeline to tolerate changes.
- Incorporate cross-layer dependency in monitoring system to help troubleshooting.

03/2018–05/2018 **System Operation and Maintenance Intern**, *SenseTime*, Beijing.

- Wrok on Ceph setup, testing, maintenance, monitoring, and alerting.

Research Experience

Harvard University

10/2018–09/2019 **Securing NIC-accelerated network functions in the Cloud**.

- Build SGX-like TEEs for NFs in SmartNICs under multi-tenant cloud environment.
- Pervasively virtualize hardware accelerators (similar to SR-IOV); Enforce single-owner semantics for on-NIC RAM and caches; Provide dedicated bus bandwidth for each network function.

Yale University (interned at Minlan's group)

06/2017–01/2018 **NF chain performance diagnosis; network measurement acceleration**.

Peking University

04/2016–03/2017 **Improving the accuracy and speed of approximate data stream processing**.

Teaching Experience

Spring 2021 **TA for CS145 Networking at Scale**, *Harvard University, MA*.
Professor Minlan Yu

Fall 2018 **TA for Algorithm Design and Analysis**, *Peking University, Beijing*.
Professor Tong Yang

Highlighted project

03/2019–05/2019 **Direct message passing in serverless platform**, Harvard CS260r.

- Add the feature of direct message passing to opensource serverless platform – OpenWhisk
- Use docker overlay network for socket connection; Implement a zero-overhead DNS service among serverless containers. Report and code

Making Data Centers Efficient, Secure, and Manageable

Yang Zhou
Harvard University

I work at the intersection of systems and networking, focusing on *data center* environments. A data center centralizes hundreds of thousands of machines with high-bandwidth, low-latency network fabrics. These machines run computations over huge amounts of data, and run popular services (e.g., Facebook's photo rendering, Google's search indexing) that impact millions of people's lives. However, there are many challenges to making data centers efficient, secure, and manageable. My research has focused on three aspects: fault-tolerant remote memory, secure smart NICs, and network telemetry.

Fault-tolerant remote memory (on-going project) aims to solve the challenge that memory resources on individual machine do not fit application needs, causing out-of-memory (OOM) errors. We design a new application-integrated remote memory approach that achieves efficient fault tolerance via erasure coding on a special object layout. Secure smart NICs project (pending submission) tackles the challenge that sharing memory resources in specialized smart NICs among different applications causes security issues. We propose several hardware primitives to build trusted execution environments (TEEs) for network functions on multi-core smart NICs. The network telemetry project (i.e., ColdFilter [1] and PCAT [2]) focuses on designing memory-efficient data structures for network monitoring and better telemetry systems to manage fast-evolving networks.

1 Fault-Tolerant Remote Memory

Modern data centers provision tightly-coupled memory and CPU resources in commodity servers, e.g., each CPU core is coupled with several Gigabytes of memory. However, heterogeneous applications have different requirements for memory and CPU resources. For example, in-memory databases usually need more memory than network packet processing applications. When deploying these heterogeneous applications, the job manager or the application team needs to estimate the maximum amount of memory one application would use; then the job manager needs to solve a complex bin-packing problem to find a nearly-optimal job allocation scheme [3]. A common application error is out-of-memory (OOM): when one application tries to use more memory than its estimated memory usage, the job manager will detect an OOM error and kill the application. Interestingly, the data center does not lack memory in aggregate; Google[3] and Alibaba [4] report only around 60% memory utilization on the average server, with substantial variance across machines.

The *root cause* of such complex yet non-optimal bin-packing and potential OOM errors is that memory and CPU resources are tightly coupled together since the time of server installation. When an application on one server wants to use memory, it can only use the memory resources provisioned on that specific server, even though the other servers in the same rack or data center have abundant free memory. Then the problem becomes how to break the server boundary of memory resources so that one server can use remote servers' free memory via network fabric. Memory disaggregation or remote memory attempts to efficiently expose a server's local memory to external servers.

Recent works on remote memory (e.g., FastSwap [5] and AIFM [6]) lack fault tolerance. In a data center with hundreds of thousands of machines, server failure is common. Without fault tolerance for the remote memory, applications running on disaggregated platforms will suffer from a much higher failure rate than if they run purely on local memory. The reason is that the failure domain of remote memory applications has expanded to multiple servers including the underlying network fabric that connects them. Only guaranteed with fault tolerance, remote memory can safely host business-critical applications.

There are three requirements to achieve cost-efficient fault tolerance for remote memory:

1. High-performance remote memory at network bandwidth. Thus, replicating to SSD (e.g., INFINISWAP [7]) does not work here due to its bandwidth limit.
2. Minimizing the amount of remote memory used for fault tolerance, because memory is an expensive resource in data centers. Thus, in-memory replication approaches (e.g., FaRM [8]) do not work here.
3. Efficient and low-latency data swapping-out/in. Thus, simply stripping and erasure coding **each** memory page to multiple remote nodes (e.g., Hydra [9]) do not work here, because it incurs too many network IOs, and suffers from straggler issues when swapping out/in pages.

In collaboration with Google, I have designed a new approach for exposing remote memory to local computation in a fault-tolerant manner, which fulfills all the above requirements.

Application code uses fat pointers that describe the locations and status (e.g., being present in local memory or not) of remotable memory objects. Application code calls *Dereference()* to swap in or keep the interested memory object to local memory with a lightweight RCU lock, and gets its local virtual address. Now, the application code can access the object data using the object's local virtual address. Our system has a set of background threads that swap out cold objects to remote memory, which is the key to enable remote memory. To support cold object swapping-out in the background, (1) we store a reverse pointer near the object data that points to the corresponding fat pointer, so that the background threads can trace back and modify the fat pointers during object swapping-out; (2) we maintain a hotness counter embedded in each fat pointer that is updated in *Dereference()* function, so that the background threads can identify cold objects by looking at the hotness counters. The abstraction of fat pointer and remotable object purely in user space enables applications to use remote memory, without customized OS kernel or paying page fault overhead (e.g., customized network-backed swapping device).

To achieve fault tolerance for different sizes of objects, (1) we allocate memory objects with similar sizes within the same spans (similar to slab memory allocator, e.g., TCMalloc), each of which is a run of contiguous pages; (2) we apply erasure coding on those page-aligned spans, and then swap out the spans with their parity data to multiple remote nodes. Packing similar-size objects into spans eases memory management and reduces memory fragmentation. More importantly, it produces page-aligned spans, that are suitable for erasure coding without padding or stripping. The swapping-out/in also happens in span granularity, where we trade more network bandwidth usage for easy memory management and erasure coding. In modern data centers, 100Gbps or even higher-bandwidth NIC on each server is becoming common – we think the network bandwidth won't bottleneck our system.

We further design a RMA-based (i.e., Remote Memory Access) swapping scheme that enables fast swapping-out/in (e.g., minimal network IOs, straggler avoidance) while guaranteeing data-parity consistency in erasure coding. For swapping-out, we first batch a few spans with the same size, generate parity from these spans, and swap them out to multiple data and parity nodes at the same time to achieve fault tolerance. For swapping-in, we only swap in the span that contains the interested object. This creates memory "holes" in remote nodes, which are required to be there to guarantee data-parity consistency. To fill these memory "holes" so that more spans can be swapped out, we spawn a few background threads in local compute node to orchestrates the defragmentation of remote memory. Our defragmentation procedure utilizes the partial update feature [10] of erasure coding to guarantee data-parity consistency on remote nodes. In addition, based on the profiling of Google's applications, we find the majority of their heap memory is consumed by large-size objects, thus our remote memory runtime prioritizes swapping out large-size cold objects to remote. This helps speed up the local memory reclaiming process.

Our preliminary results show that a graph processing application on our span-based fault-tolerant remote memory system achieves similar performance with a manually-tuned Google-internal implementation, and it only has 16% slowdown compared to running the application in purely local memory. In the near future, we plan to apply our fault-tolerant remote memory system to more business-critical applications, e.g., distributed database and in-memory database.

2 Secure Smart NICs

Modern data centers are moving network functions (e.g., firewalls, DPI, NATs) from VMs to smart NICs. A smart NIC has programmable cores which allow functions to access packets directly in on-NIC RAM, avoiding DMA overheads between host RAM and on-NIC RAM. Smart NICs also have hardware accelerators for common activities like checksum calculation. Recently, Microsoft reports a 97% TCO (i.e., Total Cost of Ownership) reduction when offloading their NVMe (i.e., Non-Volatile Memory Express) storage stack over the network to ARM-based smart NICs (see LeapIO [11]).

However, modern smart NICs provide little isolation between the network functions belonging to different tenants. These NICs also do not protect network functions from the data center-provided management OS which runs on the smart NIC. The reason is that on-NIC RAM has few access controls, and low-level hardware resources like checksum accelerators are not virtualized. These deficits hurt function robustness and security, making multi-tenant occupation of a single NIC unsafe. Even in single-tenant scenarios (where all functions and privileged software on a NIC belong to the data center provider), buggy or subverted code anywhere is a threat to all other software on the NIC.

The goal of this project is to show that, with minimal changes to smart NIC hardware, data centers can provide offloaded functions with strong isolation, while preserving many of the TCO reductions and performance boosts that offloading has traditionally provided. We introduce a new smart NIC design, called NFShield [12], that provides strong isolation guarantees. The core idea of NFShield is a set of novel hardware-level primitives that secure NF creating, launching, attestation, and destroying. For example, `NF_create` atomically installs a network function on a virtual smart NIC with necessary physical resources, generates cryptographic identifier of the function states used for later attestation, and then launches the function. After `NF_create`, a developer can use `NF_attest` to remotely attest if the current network function is running in an expected state. Only if the remote attestation passes, network traffic will be sent to the function for processing. Finally, the developer can use `NF_destroy` to atomically destroy an NF, erase the contents of its memory pages, and release its resources. In addition, to guarantee strong isolation between different network functions and from the data center providers, NFShield pervasively virtualizes hardware accelerators, enforces single-owner semantics for on-NIC RAM and caches, and provides dedicated bus bandwidth for each network function.

Using this design, we eliminate side channels involving shared hardware state, and give each network function the illusion of having a private smart NIC. The overall result is that NFShield enables the construction of strongly-isolated data center applications with minimal performance penalty ($\approx$1.7%); in these applications, network functions and host-level code receive hardware-guaranteed protections from other applications and the data center provider.

3 Network Telemetry

Network telemetry allows a data center operator to measure various aspects of the data center network's behavior (e.g., link congestion). Telemetry data must be stored in RAM, at least temporarily, but RAM is a precious resource. Network devices (e.g., NICs, switches) often have less than 100MB of RAM; server memory is more plentiful, but should be mostly devoted to applications. My ColdFilter project introduces memory-efficient probabilistic data structures that can be updated at line rate, have a low memory footprint, and high accuracy.

The first insight of ColdFilter is to have a filtering layer perform fast processing of a large number of small flows, while forwarding the remaining few large flows to existing measurement solutions for later processing. The filtering layer captures enough information about small flows with a small memory footprint, while the subsequent measure solutions use abundant memory to process fewer but more important large flows (compared to small flows). This filtering layer helps improve the accuracy of common telemetry tasks (e.g., flow size estimation, heavy hitter detection) by up to 51x. The second insight of ColdFilter is to aggregates the incoming packets on the same flows and sends them as a batch to the measurement solutions, reducing the number of computations and random memory accesses in these measurement solutions. The aggregating process of ColdFilter helps accelerate common telemetry tasks by up to 4.7x.

In 2019 and 2020, I helped Facebook develop their network telemetry system – PCAT, that handles various changes happening in their network: modification to monitoring intent, advance of device APIs, etc. Inspired by the database community [13], we introduce the change cube abstraction for telemetry to explicitly track the time, entities, property, and components for each change, and a set of primitives to explore changes systematically. Based on the change cube abstraction, we re-architect our telemetry system to be change-aware and evolvable.

PCAT includes an intent-based layering design that separates monitoring intents from data collection and supports change cubes across layers. PCAT enables change attribution by allowing network engineers with rich network domain knowledge to define intents while having software engineers building distributed data collection infrastructure with high reliability and scalability. PCAT then compiles intents to vendor-agnostic intermediate representation (IR) data model, and subsequently to vendor-specific collection models, and job models. Tracking these changes not only helps to troubleshoot the network when errors happen, but also enables change-driven topology derivation that is much more timely than our previous one. The overall result of PACT is a change-aware network telemetry system that supports fast-evolving data center networks at Facebook.

4 Future Work

In the near term, I am excited to continue working on my research into memory disaggregation. There are several optimizations that can help close the performance gaps between remote memory and local memory: (1) I want to leverage memory profiling to identify memory objects that are frequently accessed together; these objects can be coalesced onto the same span, and swapped out/in as a unit to reduce network IOs; (2) I want to offload network-intensive operations (eg, pointer chasing) to remote memory node, to reduce the multiple rounds of network latency.

Fault-tolerant remote memory is a software-only approach towards memory disaggregation. With the advent of hardware interconnect techniques, I believe there will be more efficient software-hardware co-design approaches. For example, the CXL 2.0 specification natively supports memory pooling to maximize memory utilization. A software-level abstraction (e.g., conventional kernel paging or remotable pointers) on top of these new hardware interconnects would help existing applications quickly harness the power of such technologies, without much code changes. I would be happy to work in this direction to push forward the envision of memory disaggregation.

5 Summary

My research has focused on the *efficiency, security, and manageability* of data centers. The remote memory project enables one node to use remote nodes' free memory in an efficient and fault-tolerant manner. The smart NIC project designs a new NIC architecture that supports running multiple network functions securely. The network telemetry project helps efficiently monitor and troubleshoot the data center networks. In these three projects, I motivate my research with realistic production needs and challenges, and collaborate closely with the industry to thoroughly test and verify my insights and approaches. With plentiful experience from both academia and industry, I believe I am in a good position to make data centers efficient, secure, and manageable.

References

[1] Y. Zhou, T. Yang, J. Jiang, B. Cui, M. Yu, X. Li, and S. Uhlig, "Cold filter: A meta-framework for faster and more accurate stream processing," in *Proceedings of ACM SIGMOD*, pp. 741–756, 2018.

[2] Y. Zhou, Y. Zhang, M. Yu, G. Wang, D. Cao, E. Sung, and S. Wong, "Evolvable network telemetry at facebook," in *Proceedings of USENIX NSDI*, 2022.

[3] M. Tirmazi, A. Barker, N. Deng, M. E. Haque, Z. G. Qin, S. Hand, M. Harchol-Balter, and J. Wilkes, "Borg: the next generation," in *Proceedings of the Fifteenth European Conference on Computer Systems*, pp. 1–14, 2020.

[4] C. Lu, K. Ye, G. Xu, C.-Z. Xu, and T. Bai, "Imbalance in the cloud: An analysis on alibaba cluster trace," in *2017 IEEE International Conference on Big Data (Big Data)*, pp. 2884–2892, IEEE, 2017.

[5] E. Amaro, C. Branner-Augmon, Z. Luo, A. Ousterhout, M. K. Aguilera, A. Panda, S. Ratnasamy, and S. Shenker, "Can far memory improve job throughput?," in *Proceedings of ACM EuroSys*, pp. 1–16, 2020.

[6] Z. Ruan, M. Schwarzkopf, M. K. Aguilera, and A. Belay, "AIFM: high-performance, application-integrated far memory," in *Proceedings of USENIX OSDI*, pp. 315–332, 2020.

[7] J. Gu, Y. Lee, Y. Zhang, M. Chowdhury, and K. G. Shin, "Efficient memory disaggregation with infiniswap," in *Proceedings of USENIX NSDI*, pp. 649–667, 2017.

[8] A. Dragojević, D. Narayanan, M. Castro, and O. Hodson, "Farm: Fast remote memory," in *Proceedings of USENIX NSDI*, pp. 401–414, 2014.

[9] Y. Lee, H. A. Maruf, M. Chowdhury, A. Cidon, and K. G. Shin, "Mitigating the performance-efficiency tradeoff in resilient memory disaggregation," *arXiv preprint arXiv:1910.09727*, 2019.

[10] H. Chen, H. Zhang, M. Dong, Z. Wang, Y. Xia, H. Guan, and B. Zang, "Efficient and available in-memory kv-store with hybrid erasure coding and replication," *ACM Transactions on Storage*

(TOS), vol. 13, no. 3, pp. 1–30, 2017.

[11] H. Li, M. Hao, S. Novakovic, V. Gogte, S. Govindan, D. R. Ports, I. Zhang, R. Bianchini, H. S. Gunawi, and A. Badam, “Leapio: Efficient and portable virtual nvme storage on arm socs,” in *Proceedings of ACM ASPLOS*, pp. 591–605, 2020.

[12] Y. Zhou, V. Gandhi, M. Wilkening, J. Mickens, and M. Yu, “Nfshield: Securing nic-accelerated network functions in the cloud,” 2019. Pending submission.

[13] T. Bleifuß, L. Bornemann, T. Johnson, D. V. Kalashnikov, F. Naumann, and D. Srivastava, “Exploring change: A new dimension of data analytics,” *Proceedings of the VLDB Endowment*, vol. 12, no. 2, pp. 85–98, 2018.