Madhav Jivrajani

Bangalore, India | Email: madhav.jiv@gmail.com
GitHub | **LinkedIn** | **Website**

Education

PES University — Bangalore, India
Bachelor of Technology *in Computer Science and Engineering (CGPA: 9.36/10)* — *Aug. 2018 – May. 2022*
With Specialization In Systems And Core Computing

Relevant Coursework: Performance Engineering, Cloud Computing, Adv. Database Technologies, Compiler Design, Heterogeneous Parallelism, Generic Programming, Big Data, Operating Systems, Computer Architecture, Design and Analysis of Algorithms

Awards: 6 time recipient of Prof. M. R. Doreswamy Merit Scholarship Award (awarded to the top 20% in the department)

Skills

Programming Languages: Golang, Python, C, C++
Technologies: Kubernetes, Containerd, Prometheus, eBPF, etcd, gRPC
Technical Skills: Distributed Systems, CRDTs, Systems Modelling and Verification (Queuing Theory and TLA+), Performance Debugging Go Systems, git
Non Technical Skills: Open Source Governance, Community Management and Sustainability

Work Experience

Member of Technical Staff - {1, 2} — Aug. 2021 – Present
VMware — *Bangalore, India*

Kubernetes and Etcd Maintainer

- Improved Scalability of The Kubernetes Storage Layer
  - Successfully understood and identified bottlenecks in the *Kubernetes API Server watchCache*.
  - Implemented a solution to reduce lock contention in the *watchCache* and reduce the API Server's CPU and memory footprint by over 70% in large clusters.
  - Implemented a BTree based caching layer to evolve the *watchCache* to serve paginated LIST calls.
- Helped Develop and Maintain Simulation Testing for The *Etcd* Distributed KV Store
  - Successfully identified the API interactions between *Kubernetes* and the *etcd* client.
  - Developed simulation tests to verify the API guarantees provided by *etcd* to *Kubernetes*.
  - Reasoned about consistency guarantees provided by *etcd* with and without caching enabled.
- Shepherded Kubernetes Releases
  - Ensured Kubernetes is released securely, reliably and on time, by tracking releasing blocking failures/flakes and regressions.
  - Worked across cross-cutting technical areas within the Kubernetes project (api-machinery, scalability, release, testing) to resolve reslease blockers.
  - Collaborated across open-source communities that Kubernetes depends on (most notably: *the Go project*) to fix releasing blocking bugs.

Software Development Engineering Intern — May. 2021 – Aug. 2021
Akamai Technologies India Pvt Ltd. — *Bangalore, India*

- Redesigned a legacy *Python* microservice to be extensible and scalable.
- Migrated microservice application to *Golang* and implemented a pluggable communication media to switch between *WebSocket* and *gRPC* protocols when needed.
- Added end-to-end testing for the microservice, exposing a *REST API*, covering interactions with *PostgreSQL*.
- Added resiliency measures to the microservice in the form of exponential back-off and jitters.

Research Experience

Center for Cloud Computing and Big Data Feb. 2019 – Dec. 2021
PES University *Bangalore, India*

- **Bacherlor's Thesis**: Designed A Graph-Based Autoscaling Algorithm for Microservices
  - Predicted the flow of bottleneck in a microservice application by using *queueing theory*.
  - Developed a proportional controller to interact with Kubernetes and making scaling decisions.
  - Implemented microservice service graph retrieval using *Istio*.
  - Performed comparative analysis between our autoscaler and that of Kubernetes illustrating our algorithm achieving a better Quality of Service for end-to-end throughput.
- **Research Project**: Designed and Implemented Maximal LFSR Counters On FPGAs
  - Designed maximal *LFSR* counters achieving a 50% throughput gain over conventional implementations.
  - Automated *iVerilog* code generation of counters using Haskell.
  - Used circuit optimization techniques like register retiming to improve communication latencies.
  - Designed *FIFO buffers* using maximal LFSRs for matrix multiplication on FPGAs.
- **Research Mini-Project**: Developed Performance Debugging Tools for Databases
  - Automated the benchmarking of *MongoDB* and *Cassandra* with YCSB.
  - Understood how and why *core migrations* take place for OS threads executing database queries.
  - Identified metrics for *dTLB/iTLB* misses experienced by threads.
  - Developed tools to capture the frequency of core migrations and dTLB/iTLB misses during database query execution.

Selected Talks and Interviews

- **Kubernetes Stale Reads**: *The Kubernetes Podcast by Google* [recording]
- **The Kubernetes Storage Layer: Peeling The Onion**: *KubeCon + CloudNativeCon, Nov. 2023* [talk][slides]
- **Reliably Absorbing a Go Release**: *GopherCon 2023* [talk][slides]
- **Keep CALM and CRDT On**: *Papers We Love Bangalore, Oct. 2023* [slides]
- **Using eBPF To Debug the Performance of The Go Scheduler**: *Go Bangalore, July 2023* [talk][slides]
- **Control Theory and Concurrent Garbage Collection: The Go GC Pacer**: *GopherCon 2022* [talk][slides]
- **Queues, Fairness, and The Go Scheduler**: *GopherCon 2021* [talk][slides]

A full list of my talks can be found here.

Awards

- **Google Open Source Peer Bonus Award, 2023**: For contributions to the Kubernetes project
- **Kubernetes Contributor Award, 2021**: For contributions to SIG Architecture

Conference Volunteering

- **Chair**: *GopherCon 2024*
- **Paper Review Committee**: *GopherCon 2022, GopherCon 2023*
- **Paper Review Committee and Track Chair**: *KubeCon + CloudNativeCon NA 2022*

Selected Projects

gse: *Go Scheduler Exporter* [code][talk] Oct. 2021 – Dec. 2021
- Implemented a Prometheus exporter to export scraped and processed metrics from the *Go runtime*.
- Implemented detection of Goroutine preemption using both the *Linux Tracing Subsystem* and *eBPF*.
- Visualized work-stealing in action inside the *Go scheduler* using *Grafana*.

btree-indexer: *A BTree backed Kubernetes client-go Indexer* [code] Feb. 2022 – Aug. 2022
- Implemented a BTree based cache defined by Kubernetes client-go's *Store* interface.
- Implemented indexing on top of the BTree cache for efficient lookup.
- Used this to enable the *watchCache* to service paginated LIST requests.

locknt: *Concurrent and Lock-Free Data Structures In Go* [code] Jan. 2021 – May. 2021
- Implemented concurrent and lock-free data structures in Go.
- Wrote benchmarks to understand bottlenecks in the implementation.
- Profiled the code using *pprof* as well as *perf* to understand the effects of false sharing.