Jin Dong

github.com/djdongjin | djdongjin.github.io | linkedin.com/in/jdong95 | djdongjin95@gmail.com | 332-262-8260

Education

McGill University | School of Computer Science — Montreal, Canada
M.S. in Computer Science, GPA: 4.00 / 4.00 — *2018.09 - 2020.08*

- **Relevant Courses:** Distributed System, Deep Learning, Reinforcement Learning, Natural Language Processing, Applied Machine Learning (TA).

Jilin University | School of Computer Science — Changchun, China
B.S. in Computer Science — *2013.09 - 2017.07*

- **Relevant Courses:** Algorithms, Data Structures, Database Fundamentals, Operating System, Compiler, Java Programming (TA).

Technical Skills

Languages	Go, Java, C/C++, Python, Rust, C#/.Net
Tools	Docker/containerd, Kubernetes, MicroVM, DynamoDB, MySQL, PyTorch

Work Experiences

Amazon Web Services
Software Engineer 2, Container Runtime (Fargate, containerd, snapshotter) — *2022.10 - Present*

- Maintainer and top 5 contributor of nerdctl (docker-compatible CLI for containerd), a CNCF and containerd sub-project. (link)
- Implemented cosign image signing/verification support for nerdctl compose (a docker-compose alternative). (link)
- Contributed implementations of 11 docker-compose commands to nerdctl, significantly improved nerdctl's compose support. (link)
- Developing SOCI, a containerd remote snapshotter that speeds up container launch by lazily pulling/loading images. (link)
- Decoupled the snapshotter implementation and compression algorithm (gzip), enable support to new image compression such as zstd. (link1, link2)
- Optimized SOCI's image fetch/cache component to be thread-safe and efficient by utilizing atomic operations and mutex lock. (link)
- Collaborated with AWS Fargate team and launched SOCI on Fargate platform (beta).

Microsoft
Software Engineer 2, Azure Kubernetes Service (AKS, Scalability) — *2022.01 - 2022.10*

- Integrated the next-gen Azure VM node pool (Flex) into AKS, reducing 50-node cluster creation from 9 minutes to 3 minutes.
- Contributed to scaling AKS to 5K nodes, by tuning kubelet/apiserver config, optimizing the scheduling of control plane components.
- Mentored intern project that enables customers to monitor AKS control plane metrics with their Prometheus by creating a proxy.

Software Engineer, Web Experience (Observability, Telemetry) — *2020.09 - 2021.12*

- Created an Azure AD authentication app using Java and integrated into team's OLAP service.
- Reduced the runtime of a critical MSN data pipeline from 1 hour to 20 minutes, by optimizing the data extractor and processor.

Google Summer of Code | TensorFlow
Student Developer, TensorFlow Hub Team — *2020.05 - 2020.08*

- Built command line tools for fine-tuning machine learning models based on TensorFlow Hub.
- Optimized the image classification fine-tuning tool by supporting multi-GPU training, updating to TF 2.0 Dataset pipeline.

Quebec AI Institute (Mila) & RL Lab, McGill
Research Assistant, Advisor: Prof. William Hamilton — *2019.01 - 2020.04*

- Constructed a Q&A benchmark dataset, built 4 non-structured and structured DL baseline models for the dataset using PyTorch.
- Published two research papers on EMNLP main conference about Question Answering.

Amazon | AWS AI Lab
Applied Scientist Intern, DGL Team — *2019.06 - 2019.08*

- Built a knowledge graph embedding (KGE) module for Deep Graph Library (DGL) using PyTorch.
- Implemented 8 KGE models, and matched five evaluation metrics with reported results on all models.
- Built an efficient negative sampling method and gained 20x speedup on sampling using C++.

Tencent | WeChat Group
Software Engineer Intern — *2016.12 - 2017.03*

- Built an attention-based LSTM model for joint intent detection and slot filling.
- Created a sequence-to-sequence dialog system with LSTM and trained on Weibo Dialog dataset using TensorFlow.

Project Experiences

Mini Operating System based on RISC-V and QEMU — *2021.01 - 2021.03*

- Implemented multiple system calls to the teaching operating system xv6 using C.
- Created a per-process kernal page table to speed up userspace data deference in kernal mode.
- Optimized the OS by lazy-allocating pages and copy-on-write fork.

Distributed Key-Value Storage with Fault-Tolerant Guarantee — *2020.04 - 2020.07*

- Built a distributed key-value storage and a client application using GoLang.
- Implemented the Raft consensus algorithm to guarantee fault-tolerance, passed 1000 parallel testcases.
- Partitioned and distributed the application by implementing a shard master and shard servers.