AARON CHEN

(216) 801-3618 | aaronbychen@gmail.com | linkedin.com/in/aaronbychen | github.com/aaronbychen | aaronbychen.com

EDUCATION

Washington University in St. Louis **Aug 2023 - May 2027**

B.S., Double Major in Computer Science and Applied Mathematics (GPA: 3.95/4.00) *St. Louis, MO*

- **Achievements:** AIME Qualifier (5% International), Llama Stack Finalist (Meta/8VC), ARML National Bronze, ACSL State 1st

EXPERIENCE

Amazon **May 2026 - Aug 2026**

Software Development Engineer Intern *Seattle, WA*

- Built a generalized launch-criteria engine for Weblab (Amazon's internal A/B testing platform) in Java, turning one team's hardcoded **decision tree** into a configurable **DAG** that lets **25 experience teams** self-serve custom launch criteria in **under 1h**
- Prototyped an **AI agent** side panel on Weblab's internal **tool-calling** agent, letting experimenters fill questionnaires and admins edit DAGs in **natural language**, with **actions constrained** to valid DAG edges and all **writes gated** by validation and user confirmation
- Modeled DAGs in **DynamoDB** as node maps and edge lists with **append-only versioning** and pre-persistence **validation** (cycle and orphan detection), exposing **delta-based** validate/publish **RESTful APIs** that cut publish payloads by **up to 87%** vs. resending full 100+ node / 200+ edge graphs
- Implemented Java graph traversal services that resolve the next question and final launch criteria on every click, **caching hot** DAG config with **Guava** to cut lookups from **millisecond** DynamoDB reads to **sub-millisecond** memory hits
- Unified preview and experiment creation on one evaluation path so previewed criteria always match persisted ones, and built **React/TypeScript** experimenter and admin UIs for questionnaires, criteria preview, and visual DAG editing
- Released to beta with unit, integration, UI, and end-to-end tests at **92% line coverage**, deployed on Amazon EC2 and S3

Portal **May 2025 - Aug 2025**

Software Engineering Intern *Houston, TX*

- Built **Elasticsearch** indexing with tuned mappings/analyzers and **Redis caching** with TTL invalidation for hot search, auth, and config paths, cutting p95 search latency **45% (320 to 175ms)** with **80-90% cache hit rates** and **40-60%** less DB load
- Built **Kafka**-backed async processing for resume parsing and LLM jobs, fanning out to event-driven **AWS Lambda** workers with bounded retries/backoff and idempotency keys, absorbing bursts of **200-400 jobs/min** with p95 queue lag **under 3s**
- Scaled **distributed Node** services horizontally behind **AWS API Gateway** and load balancing into a fault-tolerant setup for **5.1k concurrent sessions**
- Optimized **PostgreSQL/Prisma** with JSONB/GIN and composite indexes and offloaded large resume blobs to **S3** via pre-signed URLs
- Improved cross-service reliability with strict timeouts, fail-fast fallbacks, structured logs/metrics/alerts, and **90% line coverage**, shipping via **GitHub Actions** CI/CD and **Docker/Kubernetes** across dev, staging, and production

CN Business Herald **May 2024 - Aug 2024**

AI Engineering Intern *Beijing, CN*

- Owned the full RAG platform for offline news analysis, from ingestion and chunking to **Azure Cognitive Search** retrieval and **LangChain** guardrails
- Productized citation-backed research workflows with timelines and Q&A banks, cutting research from **0.5-2 days** to **10-20 min** for **200+ users**
- Routed inference to local models with **RouteLLM** behind an OpenAI-compatible gateway, cutting cloud-served inference **38%**, and built a self-serve dashboard for retrieval tuning and evaluation that cut normalized run and ops cost **27%**

PROJECTS

NL Video Scanner (Llama Stack Finalist) | Llama-Vision, LlamaIndex, NoSQL, OpenCV, FastAPI, Docker

- Built an on-device, privacy-preserving natural-language video scanner at 1 FPS with **Llama-Vision** + **OpenCV**, at **85%**+ **Hit@5** on **300+** clips
- Designed a fully offline **LlamaIndex** retrieval layer over local **NoSQL** caption and frame/time indexes for fast top-k queries on consumer GPUs
- Improved low-light and high-contrast retrieval by **40%+** with **OpenCV** preprocessing (denoising, CLAHE, color normalization), and served inference via **FastAPI** with **async streaming and batching**, cutting end-to-end latency **~50%** in load tests

Distributed RPC Framework | Distributed Systems, HPC, Spring Boot, Etcd, Vert.x, Linux

- Built an **RPC** core in **Spring Boot** + **Vert.x** with async I/O, connection pooling, and backpressure, hitting **sub-5ms p90** latency at **5k+ RPS**
- Implemented **etcd**-based service discovery with fault-tolerant registration and failover, validated through quorum and failure tests
- Built a custom **Vert.x async load generator** to benchmark discovery/failover, throughput, and **p90** latency on a **multi-node Linux cluster**

SKILLS

- **Languages**: Java, Python, TypeScript, JavaScript, C++, C, SQL, Swift
- **Technologies**: AWS (EC2, S3, Lambda, API Gateway, DynamoDB), Azure, Spring Boot, Vert.x, Node.js, FastAPI, React, PostgreSQL, MongoDB, NoSQL, Redis, Guava, Kafka, Elasticsearch, etcd, Docker, Kubernetes, GitHub Actions, Linux, Git
- **AI/ML**: PyTorch, LangChain, LlamaIndex, RAG, RouteLLM, OpenCV, scikit-learn, NumPy, Pandas