Dongyu Cao

dongyuchiao@gmail.com · +(86) 1800-1212-101 · cdeusyu·

Education

Chongqing University of Technology, Master's Degree 09/2020 – 06/2023

- Major: Computer Technology
- Mainly doing research on NLP (Text2SQL) and publishing papers in EI

Chongqing University of Technology, Bachelor's Degree 09/2016 – 06/2020

- Major: Software Engineering

Work Experience

Kwai Inc., Beijing, China 03/2022 – now

(Resource Management Platform) *Backend Development Engineer* 2022/03 –

- Responsible for the research and development of the supply chain and logistics for Douyin's e-commerce platform. During my tenure, I have been involved in various aspects of the inbound logistics process, including supporting pre-shipment quality inspections for our worry-free service, intercepting express deliveries in our cloud warehouse, and monitoring delivery times to proactively prevent package delays. By providing solutions that empower our merchants, I have been able to enhance their supply chain and logistics performance and improve their overall fulfillment experience.

  : Django REST framework, Go, Python, Redis, MySQL

  : ksboot, Redis, kconf, xxl-job, Mybatis-Plus

1. Participated in the design and iteration of the inbound link of the cloud warehouse, which supports both ERP and non-ERP merchants in creating, printing replenishment orders, scheduling warehouse appointments, green channel appointment scheduling (subject to approval), capacity occupancy, and pre-inspection. We incubated the cloud warehouse inbound service, consolidated the industry logic (special appointment scheduling, expiration date clear out rule verification, notify merchants of quality inspection results, etc.), decoupled the architecture code of the cloud warehouse industry and the inventory control platform, clarified responsibilities, and accelerated business iteration; implemented the strategy and factory pattern to support multiple rule verifications such as expiration date and returns, allowing for future scalability and extensibility. For the pre-shipment quality inspection project, multiple resources such as product photos, nameplate images, and PDF manuals need to be uploaded prior to warehousing. A universal mobile phone scanning and image uploading SDK was developed to support mobile scanning and uploading of images, which are then displayed on the PC side. This ensures the privacy and security of the uploaded images by limiting access to them and setting expiration times for the image links to prevent them from being used maliciously as image hosting sites. This SDK has been implemented in other areas of the supply chain, such as for the uploading of business licenses by cargo owners.
2. Participated in the design and implementation of the cloud warehouse express interception system as the overall project owner. I communicated and coordinated with downstream order fulfillment centers, logistics operations, and logistics work order domains for development and joint testing to ensure the smooth and risk-free launch of the project. Currently, the cloud warehouse intercepts x orders per day with an x success rate, resulting in x hours of reduced manpower. To handle the increasing volume of business and the need for multi-condition queries, we used MySQL and ES to store data. The system performs multi-condition joint queries by first querying ES, and then fetching and concatenating non-condition queries from MySQL. We used Faas to consume MySQL's binlog, concatenate data, and stream it to ES for synchronization.
3. Participated in the design and implementation of the QIC monitoring system, responsible for designing an abstract monitoring model, receiving messages from different data sources upstream, converting them to configured events as triggers for starting/ending monitoring cycles, and using the events to find corresponding timeout rules and generate monitoring orders for execution. The monitoring orders require filling in data information and calling multiple RPC interfaces, and to reduce repeated requests, a memory cache is constructed to store the request results in the context. Timer is used to obtain cached data from MySQL and store it in memory to reduce frequent DB queries, especially for infrequently changing rule and event configurations. This system is applied to monitoring processes in various domains, such as cross-border e-commerce and cloud warehousing.
   - aaa
   - aaa
4. aaa

(Resource Management Platform) *Backend Development Intern* 2022/03 – 2023.06

- aaa
  : springboot, mysql, clickhouse, redis, easyexcel, grafana
  1. aaa
  2. aaa
  3. aaa
- aaa
  : SpringBoot, MyBatis, MySQL, Redis, Kafka, XxlJob, Kubernetes, Grafana
  1. aaa
  2. aaa
  3. aaa
  4. aaa

Dell Inc., Shanghai, China 10/2021 – 02/2022

(VxRail VCF&Netword Team) *Backend Development Engineer*

- 1. Participated in the Java migration of the basketball schedule list, responsible for the transformation of schedule data and the design and implementation of the schedule list API. In response to the high access rate of the schedule list, the schedule data needs to be preheated to the local cache before service registration and startup. To address the issue of excessively long time for batch query of match times from MySQL, CompletableFuture and ThreadPoolExecutor were used to perform concurrent and asynchronous queries of MatchModel from MySQL and store the results in MatchList. To prevent the issue of cache avalanche, MatchList was partitioned and the nodes were randomly assigned expiration values before being stored in Redis cache.
  2. Participated in the Java migration of the live broadcast scoreboard and was responsible for the design and implementation of the scoreboard API. In response to the high hit rate of scoreboard data access, the scoreboard data was stored in Redis cache with an expiration time set. To ensure the timeliness of scoreboard information updates during matches, an xxl-job was used to query real-time data from the live broadcast room and compare it with the latest snapshot version of the scoreboard stored in the cache. If there was any data change, a new snapshot version was added. Considering the immaturity of the previous and current solutions, the previous Redis publish-subscribe mechanism was combined with the current MQTT double push to provide updates to the frontend. To address the issue of instability and frequent errors in the data source for the scoreboard, a Groovy script was written to compare data from multiple sources. Any discrepancies triggered a DingTalk alert to ensure timely identification and resolution of data issues.
  3. Participated in the Java migration of the live broadcast scoreboard and was responsible for the design and implementation of the scoreboard API. In response to the high hit rate of scoreboard data access, the scoreboard data was stored in Redis cache with an expiration time set. To ensure the timeliness of scoreboard information updates during matches, an xxl-job was used to query real-time data from the live broadcast room and compare it with the latest snapshot version of the scoreboard stored in the cache. If there was any data change, a new snapshot version was added. Considering the immaturity of the previous and current solutions, the previous Redis publish-subscribe mechanism was combined with the current MQTT double push to provide updates to the frontend. To address the issue of instability and frequent errors in the data source for the scoreboard, a Groovy script was written to compare data from multiple sources. Any discrepancies triggered a DingTalk alert to ensure timely identification and resolution of data issues.
  4. Participated in the Java migration of the live broadcast scoreboard and was responsible for the design and implementation of the scoreboard API. In response to the high hit rate of scoreboard data access, the scoreboard data was stored in Redis cache with an expiration time set. To ensure the timeliness of scoreboard information updates during matches, an xxl-job was used to query real-time data from the live broadcast room and compare it with the latest snapshot version of the scoreboard stored in the cache. If there was any data change, a new snapshot version was added. Considering the immaturity of the previous and current solutions, the previous Redis publish-subscribe mechanism was combined with the current MQTT double push to provide updates to the frontend. To address the issue of instability and frequent errors in the data source for the scoreboard, a Groovy script was written to compare data from multiple sources. Any discrepancies triggered a DingTalk alert to ensure timely identification and resolution of data issues.

Skills

- Familiar with Go programming, understanding of object-oriented thinking, proficient in Go encapsulation, composition, and interface features.
- Familiar with Java programming, understanding of object-oriented thinking, proficient in Java encapsulation, inheritance, and polymorphism features.
- Familiar with common data structures (Array, List, Stack, Queue, Map, Set, BinTree, BST), understand AVL, RBtree, B/B+ tree, skip list).
- Familiar in commonly used sorting algorithms including bubble sort, insertion sort, selection sort, merge sort, quicksort, heapsort, bucket sort, and counting sort.

- Familiar in Go concurrency programming, including goroutines and channels, waitGroup usage, and locking control (Mutex, RWMutex, Once).
- Understanding of Go's garbage collection (three-color marking, mixed write barrier) and memory allocation.
- Familiar with Java multi-threading, including thread creation, locking control (Synchronized, ReentrantLock), and understanding of CAS.
- Understanding of Java Virtual Machine, including runtime data areas, GC collection and recovery, and class loading mechanisms.
- Familiar with the OSI seven-layer model and the TCP/IP four-layer hierarchical structure, proficient in common network protocols including HTTP, TCP, UDP, ARP, ICMP, DNS, and DHCP.
- Familiar with the TCP three-way handshake and four-way handshake, knowledgeable about TCP flow control and congestion control, and familiar with the working principles of the security mechanisms of HTTPS.
- Familiar with database transaction properties and transaction isolation levels, knowledgeable about MVCC, Next-Key Locks, snapshot read, and current read. Proficient in MySQL index principles and query performance optimization.
- Familiar with storage engines (InnoDB, MyISAM), and master-slave replication.

Thanks

- Thank you for taking the time to review my resume. I look forward to the opportunity to work with you.