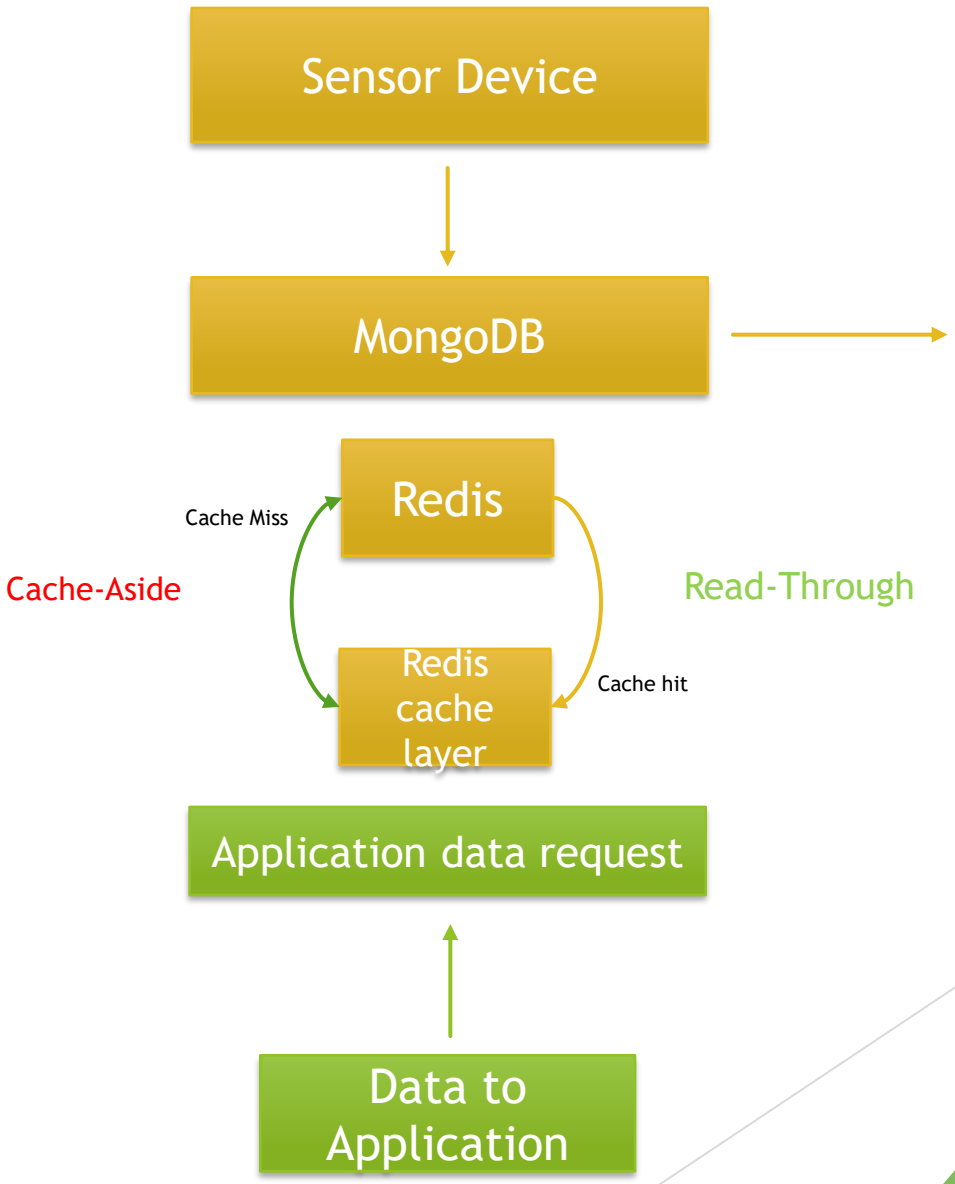
Architecture & ETL Data Flow



Lessons Learned from Cached Database Pipeline

- Redis caching dramatically reduced database read latency and improved app responsiveness.
- Cache-aside and read-through methods provided high hit rates on repeated reads.
- TTL caching maintained a good balance between data freshness and performance.
- Write-through ensured cache consistency, while write-behind enabled faster write performance.
- Performance tests showed significant improvements under warm-cache conditions compared to cold-cache queries.

Challenges, Observations & Future Improvements

- Cache invalidation and synchronization across TTL windows were challenging.
- Write-behind strategy required proper background flushing to prevent data loss.
- Monitoring Redis memory usage and ensuring optimal eviction policies were crucial.
- Ensuring MongoDB and Redis data parity under concurrent operations was complex.
- Future goals: Add LRU eviction, implement monitoring dashboards, and explore Redis Streams for real-time synchronization.