Scalable Distributed Sorting System Implementation

Netty 기반 Hybrid Network(TCP/UDP)
와 External Merge Sort 구현

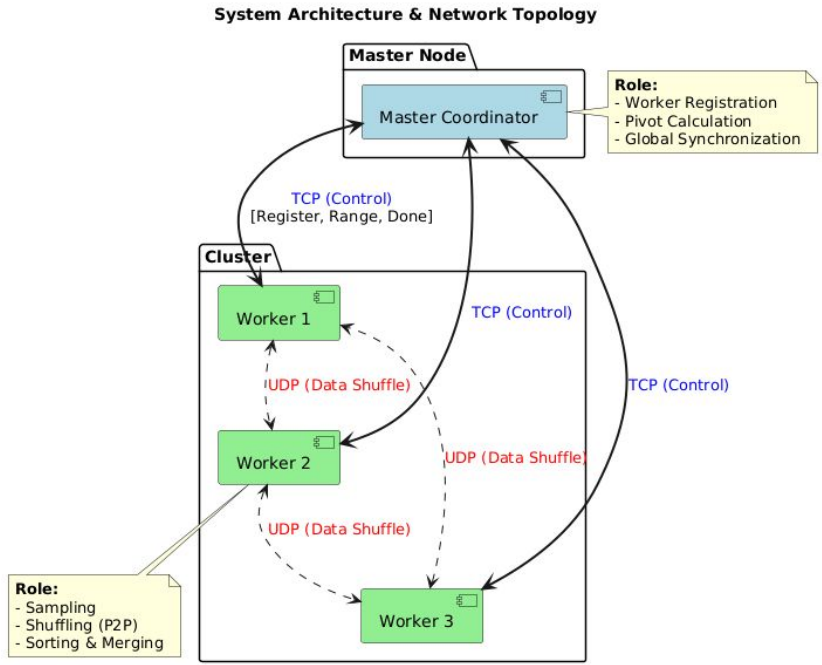
신지용 안강현 박수민

Project Overview

- 목표: 대용량 데이터를 여러 Worker 노드에 분산하여 정렬하는 시스템 구축
- 핵심 스펙:
  - 데이터: 100 Byte 레코드 (10 Byte Key + 90 Byte Value)
  - 언어/프레임워크: Scala, Netty (Non-blocking I/O)
  - 아키텍처: Master-Worker 구조
- 전체 흐름 요약:
  - 1) Worker 등록 & 샘플링
  - 2) Master의 키 범위(Partitioning) 계산
  - 3) Worker 간 데이터 셔플(Shuffle)
  - 4) 로컬 정렬 및 병합(Merge)

System Architecture

- Master-Worker 토폴로지
- Master 역할:
  - - 작업 조율, 상태 관리, 파티션 범위(Pivot) 계산
- Worker 역할:
  - - 데이터 저장, 샘플링, 셔플, 정렬, 병합 수행
- 특징: 데이터 전송은 Worker 간 P2P로 수행하여 Master 부하 감소

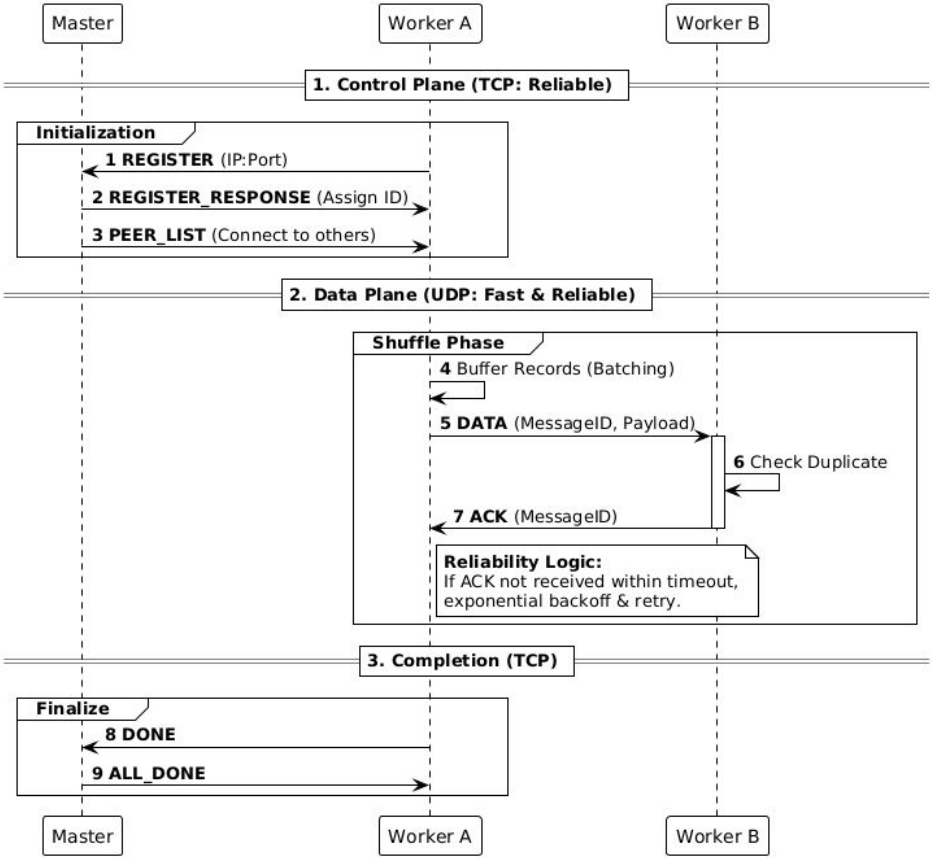


Network Design: Hybrid TCP/UDP

- 설계 의도: 제어 신뢰성과 대용량 전송 효율성 동시에 추구
- TCP (Control Plane):
  - Master ↔ Worker 통신 (등록, 피어 리스트, 완료 신호)
  - Netty NioServerSocketChannel, Length-field framing 사용
- UDP (Data Plane):
  - Worker ↔ Worker 데이터 셔플
  - Netty NioDatagramChannel 기반 구현
  - TCP 연결 오버헤드 없이 고속 전송 지향

UDP에 ACK를 도입하여 신뢰성을 부과하였으며, ACK를 즉시 반환하지 않고, 보관 성공 후 ACK를 반환하여 Worker 장애 시 복구 가능하도록 함.

Network Design: Hybrid TCP/UDP

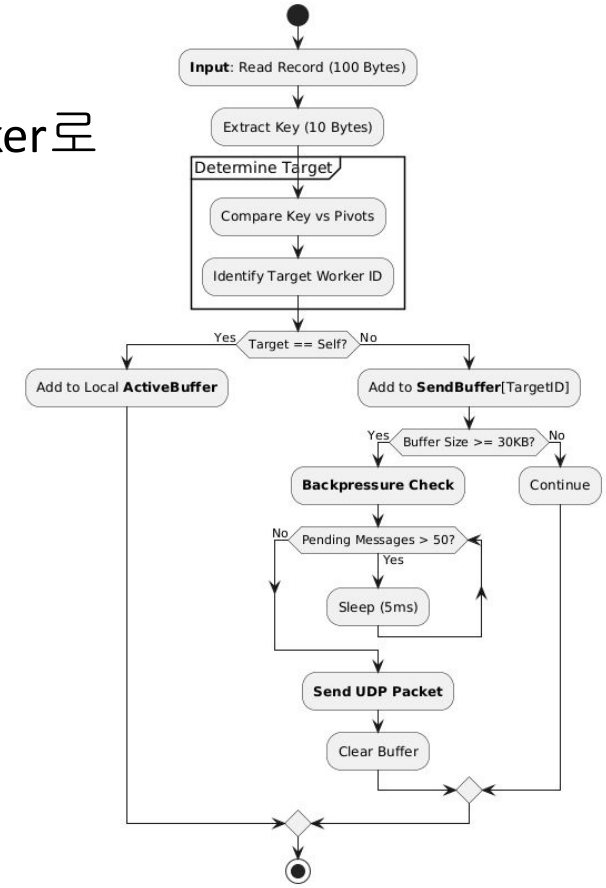


Sorting Algorithm (1): Partitioning

- 목표: 키 범위에 따라 데이터를 워커들에게 균등 분배
- Sampling & Pivots 과정:
  1) 각 Worker가 입력 데이터 일부를 샘플링 후 Master로 전송
  2) Master가 샘플을 정렬 후 Worker 수만큼 등분 Pivot 계산
  3) KeyRange 정보를 모든 Worker에게 브로드캐스트
- Unsigned Byte 비교: 0xFF 마스킹을 활용해 Signed Byte 문제 해결

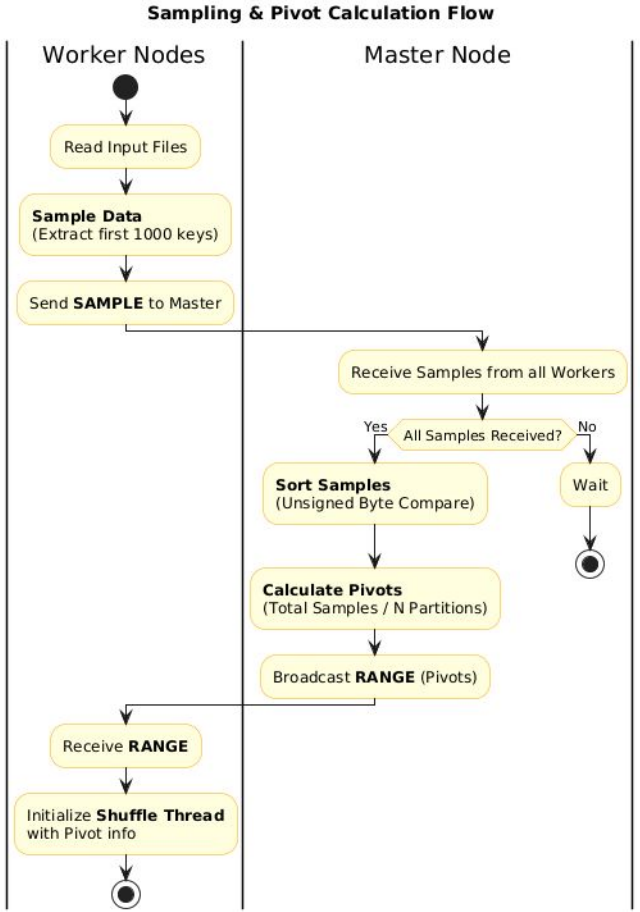
Sorting Algorithm (2): Shuffle Phase

- 데이터 이동: Worker는 레코드 Key에 따라 타겟 Worker로 전송
- 성능 최적화 (Buffering & Batching):
  - 레코드를 즉시 전송하지 않고 30KB 버퍼 단위로 모아서 전송
- 흐름 제어 (Backpressure):
  - MAX_IN_FLIGHT = 50 제한으로 ACK 대기 메시지 수 제어



Sorting Algorithm (3): External Merge Sort

- 메모리 제약 극복: Disk Spilling 방식 사용
- 1단계 – In-Memory Sort:
  - DataSorter 버퍼(MAX_RECORDS)에 데이터 저장
  - 버퍼가 차면 정렬 후 임시 파일(run_xxx.dat)로 플러시
- 2단계 – K-Way Merge:
  - 모든 데이터 수신 후 DiskMerger가 임시 파일 병합
  - PriorityQueue를 사용하여 O(K log K) 병합



Implementation Details & Troubleshooting

- 동시성 제어:
  - LinkedBlockingQueue, ConcurrentHashMap, synchronized 사용
  - Netty EventLoop와 Worker Thread 간 컨텍스트 스위칭
- 리소스 관리:
  - FileIO Iterator 패턴으로 대용량 파일 처리 시 메모리 사용 최소화
  - 작업 완료 후 임시 디렉토리(tmpDir) 정리

Validation & Demo

- 검증 도구:
  - InspectData: 레코드 수, 첫/마지막 키, Hex 포맷 확인
  - ValidateSort: 파티션 내부 정렬 여부 전수 검사
- 테스트 시나리오:
  - (Small/Big/Large) 데이터 모두 정렬 성공
  - 모든 파티션에서 [PASS] Sorted correctly 확인
  - Worker 장애 시 복구(Fault Tolerance) 기능

Conclusion & Future Work

- 결론:
  - Master-Worker 구조 분산 정렬 시스템을 바닥부터 구현
  - Netty 기반 고성능 네트워크 + External Sort로 대용량 처리
- 배운 점:
  - 비동기 네트워크 프로그래밍의 복잡성
  - 분산 환경에서의 상태 동기화 중요성
- 향후 과제:
  - 디스크 I/O·네트워크 I/O 병목 구간 프로파일링 및 최적화

Q & A