Chukwa:
A Large-Scale Monitoring System of Yahoo!

조영일

목차

소개

- "Chukwa"
  - 대규모 분산 시스템을 모니터링하고 분석하기 위한 데이터 수집 시스템
- Hadoop 기반 시스템
- Yahoo!와 UC Berkeley의 공동 연구
- 현재 Apache Project (v0.6, Nov 14)

소개

- Yahoo!의 Hadoop 클러스터 규모
  - 당시 2천 대 규모표
  - 0.5 TB / day 정도의 로그 데이터 생성
    - 디버깅, 성능 측정, 운영 모니터링 용도의 로그 데이터

요구사항

- 초 거대 규모로 배포될 수 있게 확장성 확보
- 장애를 우아하게 핸들링
- Hadoop에 쌓이는 대량의 데이터 중 기계적으로 산출되는 일부 지표를 제외하고는 대부분을 활용하지 못하고 있음
- ad-hoc 스크립트로 후처리 분석하는 게 현실 —> 당분간은 이게 정답
- 오픈소스 툴셋을 최대한 활용해보자!

목표

- Hadoop 기반, 기존 도구를 레버리징
- 일간 수 TB를 발생하는 규모의 클러스터를 감당
- 초 단위가 아닌 분 단위 모니터링
    - 실시간 장애 감지 용도는 포기 (대안: Ganglia, Nagios)

목표

- 대상 사용자 – 각각 관심이나 요구사항이 다름
  - Hadoop 사용자
    - '내 job이 얼마나 걸릴까, 어떤 자원을 사용할 수 있을까?’
  - 클러스터 운영자
    - HW 장애와 성능 이상, 자원 부족을 통보받는 게 주된 관심
- 클러스터 관리자
  - 공급과 배분에 대한 안내가 필요함. 비용 예측을 위한 도구가 필요함
- Hadoop 개발자
  - 운영 성능, 병목, 공통적인 장애 패턴 등의 정보를 확인하고자 함

구조

- 초기 아키텍쳐

Figure 1: The Chukwa Pipeline, showing how long data is retained at each stage.

구조

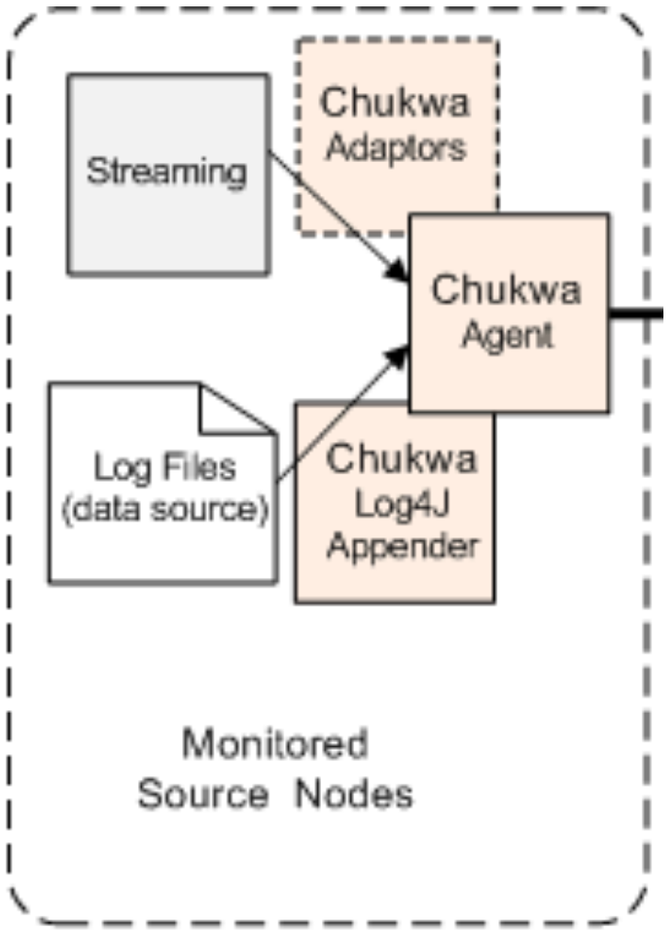
- 구성요소
  - adaptor
  - agent
  - collector
  - storage
  - demux & archive

구조

- Adaptor
  - 수집되는 데이터는 시간에 따라 변하고 머신마다 다르기 때문에 데이터 원본을 동적으로 조정할 수 있어야 함
    - 파일이나 CLI 도구 형태로서 데이터 원본을 wrapping
- 기본 adaptors
  - Hadoop Log, App 지표, …
  - disk read error count, tracer log, OS 모니터링, JVM 상태, …

구조

- Adaptor

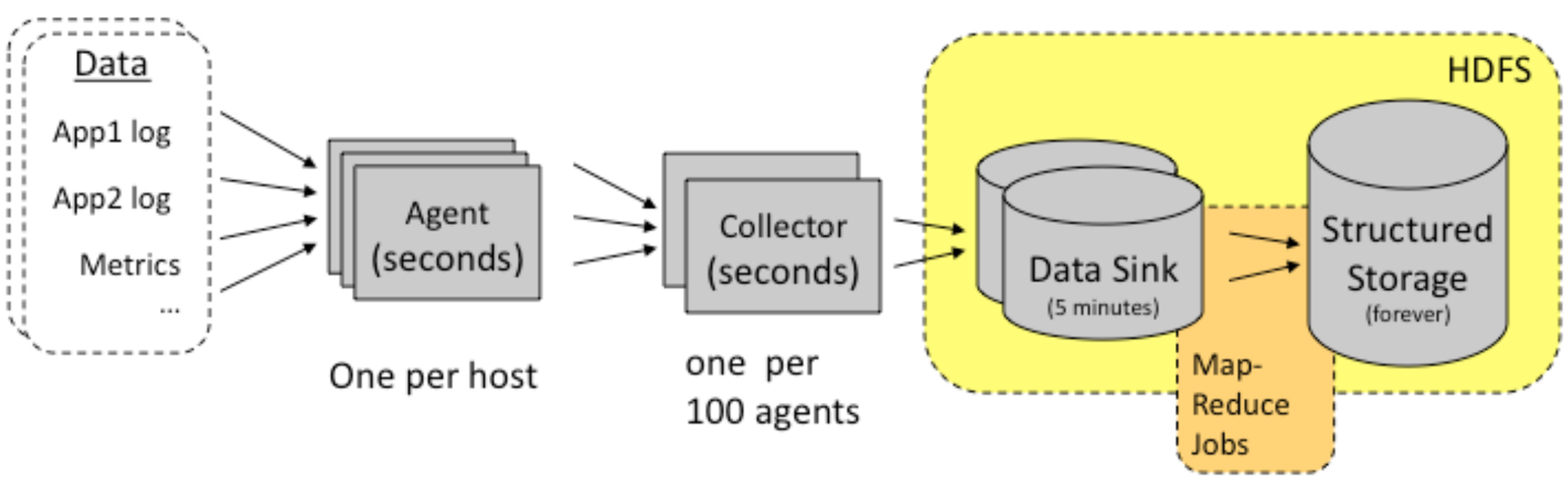

Streaming
Chukwa
Adaptors
Chukwa
Agent
Log Files
(data source)
Chukwa
Log4J
Appender
Monitored
Source Nodes

구조

- Storage
  - 확장성을 결정짓는 요소
    - HDFS를 사용하기로 결정
      - PB size + GB/s throughput
    - 데이터 분석할 때 MapReduce 사용이 용이함
- 산발적인 데이터 쓰기와 HDFS는 궁합이 좋지 못함

구조

- Agent & Collector & Storage



구조

- Collector
  - HDFS write를 담당
  - agent로부터 받은 데이터 chunk 여러 개와 chunk에 대한 메타데이터를 결합하여 하나의 sink file로 만들어 저장
- “close” - 웹서버 로그 rotation과 비슷한 작업 수행
- Java servlet으로 구현(HTTP 수신)
- HDFS의 파일 개수를 조절하는 역할
- Hadoop 시스템의 형상 변경을 감춤, or any other storage?

구조

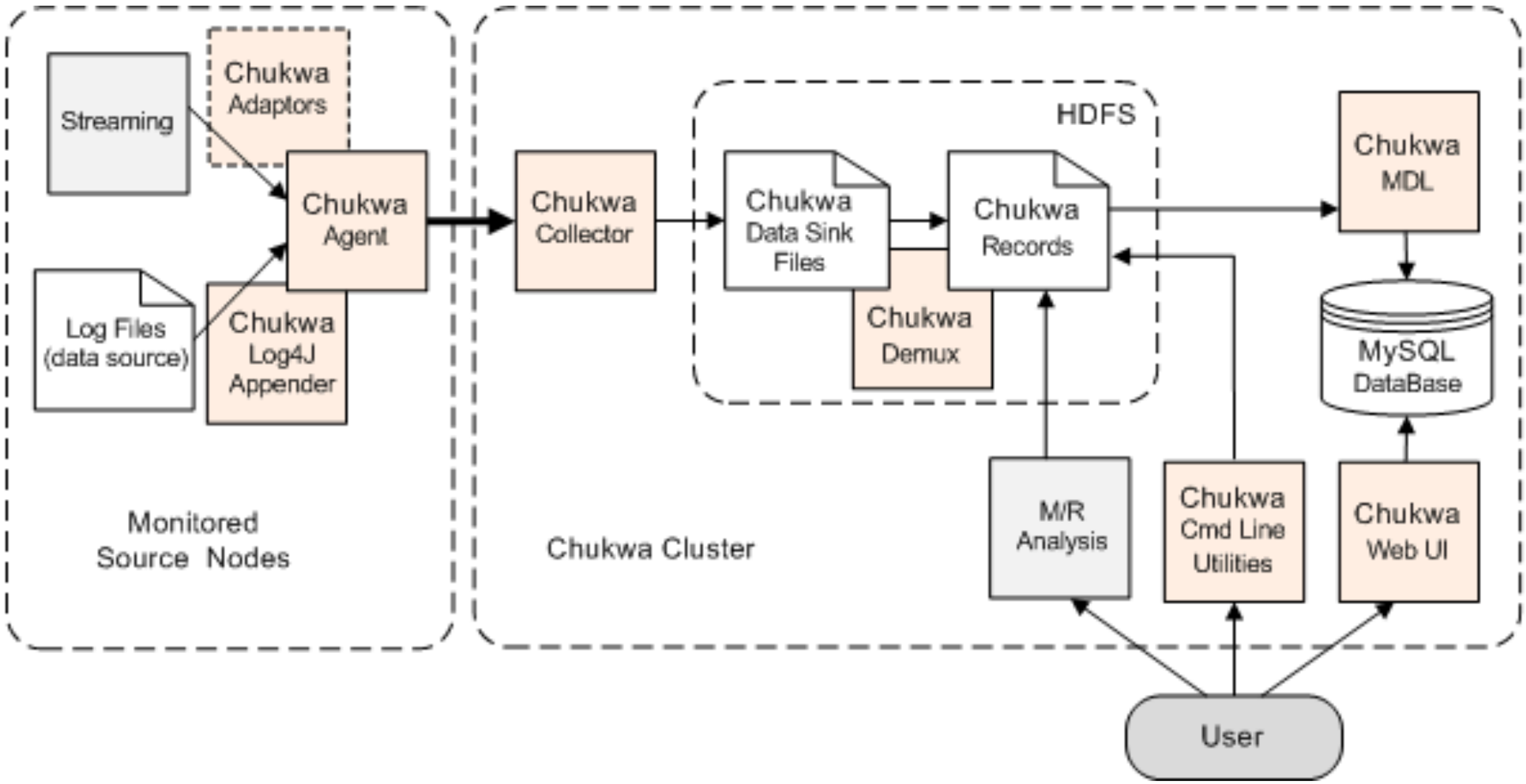
- Agent
  - long-running process
- adaptor 개발을 용이하게 해주는 다양한 서비스 제공
  - 외부로부터 명령을 받아 adaptor의 start/stop 제어
  - collector에게 HTTP로 로그 데이터를 전송
  - adaptor의 상태를 정기적으로 체크포인팅하고 장애 시 restart

구조

- Demux & Archive
  - 사용가능한(closed) sink file을 입력으로 해서 MapReduce job을 두 개가 실행됨
    - archive: 가공없이 그대로 저장
    - demux: 로그에서 구조화된 데이터를 추출하여 datastore에 저장
      - MySQL, HDFS, Hive, HBase, Hypertable, ...
      - 일부 데이터만 분석하여 장애 상황, 성능 저하 이벤트, 취약한 시스템 탐지에 활용

구조

- Detailed View

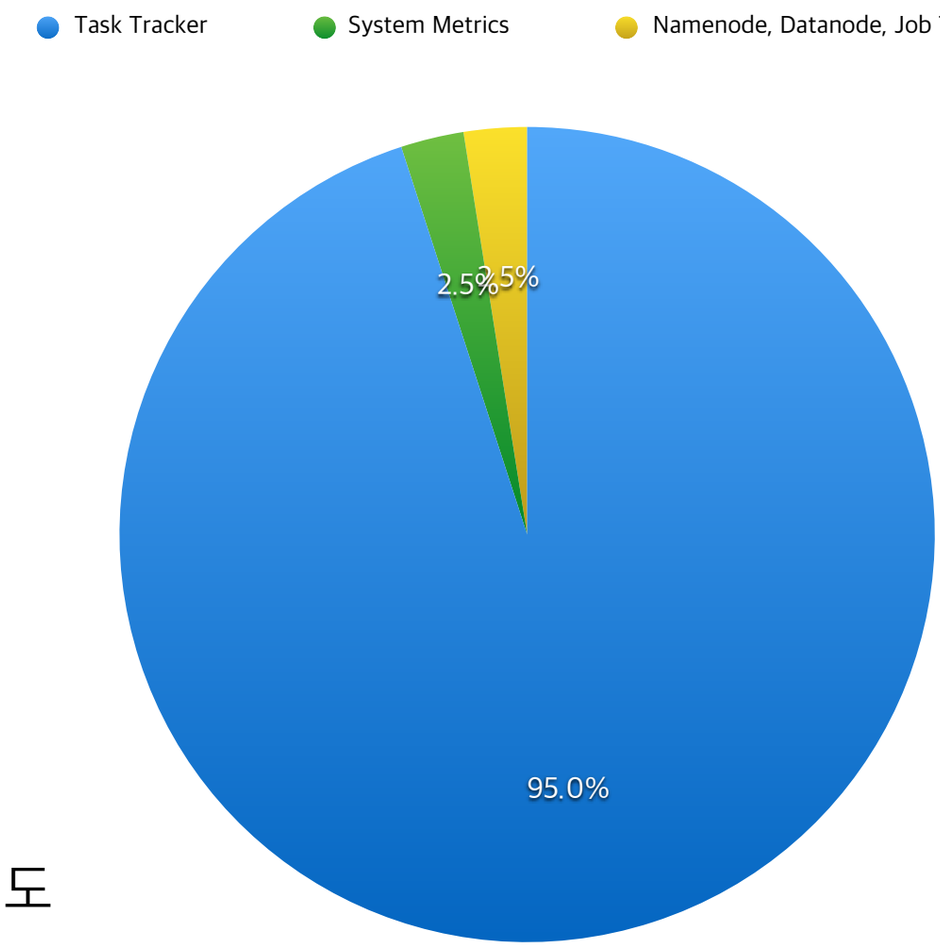


데이터 분석 및 표시

- Web UI 제공이 필요함
- 사용자마다 관심이 달라서 configurable UI가 필요함
- query의 widget화하여 조립 가능하도록
- 미리 준비된 job을 실행하거나 on-the-fly로 query 실행

평가

- Yahoo! 현황
  - production cluster에서 로그를 모아보니 2k 서버, 5.5MB/s 데이터 생성 중
- 성능
  - Linux 2.6.9, 2.8GHz Xeon CPU x 4, IDE HDD x 4, 3GB RAM
  - collector와 demux job이 bottleneck
  - collector가 최대 14MB/s까지 감당 (50% 미만 utilization)
  - 5 worker node를 가지고 2GB 처리에 3.5분 소요. collector 입장에서는 6분 분량이므로 합쳐서 10분 정도의 지연이 발생



개선할 점

- 기존 NMS 시스템의 data display 도구를 결합
- syslog보다는 유용함
- Facebook Scribe는 공개 안 하냐?
- Nagios와 Tivoli는 centalized, Ganglia는 decentralized
  - Nagios와 Ganglia가 수집하는 데이터는 일부일 뿐
  - Ganglia는 시계열 처리에는 뛰어나지만 복잡한 처리/분석에는 부적합

결론

- 기존 분산 데이터 수집 프레임웍(Hadoop)에 기반한 고성능 분산 모니터링 시스템

- Latency 이슈는 Hadoop 시스템 개선에 따라 차차 해결될 것으로 전망

Thank You