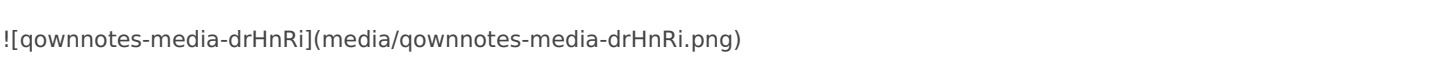
관측 가능성 (Obserability) 요약

> ⚠ 개인적으로 공부했던 부분이라 사실과 다르게 자의적인 해석이 있을수 있음



왜 옵저빌리티(observability)를 도입해야 하는가?

1. 서비스 장애시 기존에는 담당자의 경험으로 처리했으나, 컨테이너화 되면서 내부시스템은 복잡해졌고, 단순 모니터링으로는 이슈대응이 매우 어렵게 되었음
2. 목표는 내부시스템에 대한 자세한 정보를 기반으로 미래에 발생할 이벤트를 예측하고, 이런한 예측을 바탕으로 IT운영을 자동화 하는 것.
   (ef. 장애가 발생할 위험이 있으면 미리 예측하고 운영자에게 통보, 서비스에 필요한 리소스 증감을 미리 예측하는 것)
3. 단순 모니터링이 아닌 서비스간의 상관관계를 파악하고 상태 데이터를 기사화 함으로써 개발자, 운영자에게 빠른 의사결정을 할 수 있도록 도와주기 때문에 필요하다.
4. 옵저빌리티와 AI를 결합한 AIOps로 나아가기 위한 발판마련

옵저빌리티 원론

1. 옵저빌리티의 한줄요약

   > “ 오직 시스템의 외부 출력만을 이요해 시스템의 형재 상태를 이해할 수 있는 능력

2. 옵저빌리티를 구성하기 위한 조건
   - 어플리케이션의 내부 동작 이해
   - 모든 시스템 상태를 이해함
   - 외부 툴을 이요한 관측데이터로 내부 동작과 시스템 상태 이해
   - 시스템 상태의 상태와 상관없이 이해할수 있어야 함
3. MELT(Metric / Event / Log / Trace)기능을 포함하여 흩어진 데이터를 한곳에 모아 지속적인 관측을 통해 서비스 이슈 발생시 MELT를 이용해 분석, 해결할 수 있는 방법을 제공
4. 옵저빌리티는 특정 Tool의 집합체가 아닌 개념원리로 접근이 필요할듯

옵저빌리티 사용 용어

1. 카디널리티(cardinality) - 유니크한 값의 수 (수치가 낮다 - 데이터의 중복도가 높다, 수치가 높다 - 데이터의 중복도가 낮다.) 주민등록번호 (유니크한값이 있고 중복데이터가 없기 때문에 카디널리티가 높다라고 표현
2. 화이트박스 모니터링
   - 로그나 프로파링일 인터페이스를 제공하는 핸들러를 이용해 내부 측정 기준에 따라 모니터링 의미. http 응답코드 500 에러 발생횟수, 레이턴시 등. 현상이 발생한 근거를 확인하기 위한 방식
3. 블랙박스 모니터링
   - 사용자가 보게되는 기능을 외부에서 테스트 하는 방법 인프라 수준의 모니터링에 유용, 클러스터 정상작동 여부같은 것들 모니터링하기 때문에 디버깅 정보로 활용은 어려움.
   - 서버상에 나타나는 증상을 기본으로 하며 발생하는 현재 상황에 대해 모니터링
4. 옵저빌리티의 영역에서는 화이트 박스를 기반으로 예측하여, 모니터링 영역에서는 블랙박스 기반으로 진행한다.
5. 모니터링의 기본개념은 모든것을 수집하고 모니터링하기 보다는 적은 데이터라도 의미있고 예측 가능한 결과를 도출해 내는것이 좋은 방법
6. 전통적인 모니터링은 블랙박스를 기본으로 핵심적인 App과 시스템 메트릭을 중점으로 보는 반면, 옵저빌리티는 이벤트, 대그, 로그등을 결합해

서 서비스에 대한 정보를 제공하는 것이 목표. 때문에 세부적인 시스템 내용을 포함하고 있으므로, 디버깅에 적합하다.

구성요소

1. 메트릭(Metric)
   - 일정시간동안 수집된 데이터를 집계하고 수치화하는것.
   - 전체적인 시스템의 상태를 확인할때 유용
2. 로그(Log)
   - 로그관리의 목적은 시스템에 분산된 로그를 한곳에 저장해서 다루기 쉽게 하는것.
   - 로그가 표준화되어 있지 않은 경우 쉽게 검색이 불가능하고 후속처리를 위한 데이터 가공을 어렵게 함. 때문에 로그 표준화가 미리 되어 있어야 하고, 언제 발생하게 될지 모르는 장애를 대비에 지속적으로 로그 수집 / 관리가 되어야 한다
   - 로그관리 시스템은 동적인 시스템으로 구성되어야 급격이 증가하는 로그를 받을수 있다
   - 서로 다른 데이터가 정확하게 결합이 될때 데이터의 활용가치를 높일수 있다. (로그/메트리/추적/트래픽 데이터는 이기종 데이터가 아니며 결합되지 않은 데이터일뿐이다)
3. 이벤트(Event), Newrelic / Redhat에서 제안 ** - 이벤트는 로그와 다르다 **
   - 중요한 분석지점의 세부적인 기록이며, 정확한 문맥 제공을 위해 첨부되는 메타데이터에서 발생한 중요사건을 기록한것
4. 추적(Trace)
   - 서비스가 시스템을 경유하며 트랜잭션을 처리하는 과정에서 발생하는 세부적인 정보
   - 대기시간, 지연, 병목이나 에러를 일으키는 원인을 문맥과 로그, 태그등의 메타에 출력한다.

기존 모니터링 솔루션과의 비교

역활의 차이점

1. 모니터링은 정해진 템플릿에 대해서 보여주는거 것. 무엇이 언제 일어났는지에 대해 초점이 맞춰져 있다면, 옵저빌리티는 왜 어떻게 일어났는가까지 파악할 수 있도록 도와준다.
2. 옵버저빌리티를 통해 평균해결시간(MTTR)을 가속화 한다
3. 옵버저빌리티는 모니터링을 대신하는것이 아님. (옵저빌리티가 모니터링보다 더 큰 영역의 업무범위)
4. 시스템에서 외부로 출력되는 값을 사용하여 시스템 내부 상태를 이해하고 예측하는 행위
5. 레거시 모니터링 솔루션의 정적 데이터를 대체하여 인프라의 전체 스펙트럼을 분석

기능의 비교

1. 모니터링
   - 이슈 발생시 수집된 데이터를 가지고 분석
   - 대시보드에 표현된 항목을 기반으로 한 해결방법 접근
   - 큰 변화가 없는 환경에서 시스템의 이해도가 높은 시스템 개발자가 주로 사용
2. 옵저빌리티
   - 이벤트 발생시 단순 메트릭외에 여러가지 데이터를 제공하여 원인 분석
   - 가설을 기반으로 한 해결방법 접근
   - 가변데이터가 많은 복잡한 인프라 환경에서 에서 시스템 이해도가 낮은 사람도 분석 가능

APM/DPM과 차이점

1. 옵저빌리티는 APM/DPM지표가 포함되지만, 로그, 이벤트, 추적과 같은 유형으로 확대하며 시스템 동작에 대해 전반적인 이해를 높이는데 목적이 있다
2. APM/DPM은 어플리케이션 성능 모니터링 및 관리 (응답시간, 오류율, 트랜잭션 추적 등)하며, UX과정에서 발생할수 있는 문제를 찾아내어 해

결하도록 설계
3. APM/DPM은 특정 세션(어플리케이션 성능)에 초첨을 맞춘 옵저빌리티의 하위집합 개념이며, 옵저빌리티는 전체시스템 동작에 중점을 맞추었음.

Tool

구축업체

옵저빌리티를 활용한 솔루션 구축업체는 생각보다 굉장히 많음, Whatap, New Relic, Elastic, IBM(Redhat), ServiceNow, 등등등

Opentelemtry

- ovservability의 통합 표준 OSS, 보편화된 가능성이 높음
- 리시버 : 데이터 수집(prometheus)
- 프로세서 : 데이터 정제(배치/필터 실행)
- 익스포터 : 백엔드로 데이터 전달(데이터독, E/S)
- 2022년 GA발표이후 벤더(데이터독, 뉴렐릭 등)에서도 자사 API보다 오픈텔레메트리API를 더 권장하고 있음

Appendix#1. 장애조치 관련 용어

1. MTBF(Mean Time Between Failures) : 평균 장애 간격, 예정된 중지시간은 계산하지 않음. 예상치 못한 서비스 중단이슈만 계산
   계산기간의 총 가동시간을 장애 발생 횟수로 나눔. ef. 24시간 동안 2시간의 가동중지의 경우 MTBF는 11시간 (24-2)/2(횟수)
2. MTTR(Mean Time To Repair) : 해결에 걸리는 평균시간, 수리하는데 걸리는 평균시간, 수리및 테스트 시간모두 포함하며, 완전시 작동할때까지의 시간을 측정
   수리에 소요된 총 시간을 합산 / 수리횟수 10회의 중단, 4시간 동안 수리했다면 240(분)/10(중단횟수) = 24분 , MTTR는 24분
3. MTTA(Mean Time To Acknowledge) : 평균확인시간, 알람이 트리거 된 시점부터 이슈작업이 시작되는 시점까지 평균시간
   알람과 확인사이의 시가늘 합산 인시던트 수로 나눔. 10개의 인시던트 발생, 10개의 알람과 대응시작 사이의 시간이 총 40분인 경우 40/10=4, 평균 4분 소요
4. MTTD(Mean Time to Detect) : 평균 탐지 시간, 문제 발생후 식별까지 소요되는 시간
5. MTTF(Mean Time To Failure) : 평균 장애시간, 수리 불가능한 장애사이의 평균시간을 의미하며 총 작동시간 합산 후 장치수로 나누기.

- MTTA와 MTTD의 차이는 MTTD는 이슈 발생 후 알람 수신까지 소요되는 시간이고, MTTA는 이슈 발생시 인지 후 작업 시작까지 소요되는 시간을 측정하는 차이 존재

Appendix #2. 프로메테우스 데이터

1. 데이터 관리방법
   - 프로메테우스는 TSDB(Time Series DataBase)유형의 데이터베이스이며, LRU알고리즘을 사용하는데, 가장 오랫동안 참조하지 않은 페이지를 교체
   - 메모리 페이징을 사용하는데, 일정크기의 페이지로 분한해서 메모리에 적재, 데이터를 수집하고 블록형태로 만들어 디스크에 저장함
   - 데이터대신 청크(Chunk)라는 용어를 사용, 다수의 청크를 포함해서 인덱스와 기타 데이터로 수정된것이 블록
   - 데이터셋은 다수의 데이터 그룹을 의미
   - 데이터 포인트는 대시보드에서 시계열로 출력되는 개별데이터
2. 데이터 경로 hierarchical

```
./data
 -abc
 -def                # 블록정보
  - chunks          # 청크경로
    - 1
```

```
  - tombstones  # 삭제 여부 표시파일
  - index       # 색인을 위한 레이블과 시간정보 인덱스 파일
  - meta.json   # 블록의 메타데이터
- checks_head   # 청크 헤드
  - 1
- wal
  - 001
    - checkpoint.001   # wal 파일
      - 000            # 복구를 위한 체크포인트 wal파일
```

- WAL(로그 선행기록, Write Ahead Logging)은 장애 복구를 위해 사용, 프로메테우스는 메모리에 우선 저장하고 주기적으로 flush하면서 디스크에 청크를 생성하는데, 청크를 생성하기전 메모리에 있는 데이터를 백업하는 용도로 WAL생성, 프로메테우스 장애시 메모리에 있는 데이터에 문제가 생기며 WAL을 사용헤 메모리에서 관리하는 데이터를 복구 수행

1. 관련옵션
   - storage.tsdb.min-block-duration : 하나의 블록에 저장된 데이터의 시간, 2h로 선언하는 경우 하나의 블록디렉토리에는 두시간의 데이터 저장
   - storage.tsdb.max-block-duration : 하나의 블록에 최대로 저장할 수 있는 시간, 기본값은 유지율을 설정하는 옵션인 storage.tsdb.retention.time의 10%값이다
2. 데이터 저장단계
   - 메모리에는 최대 두시간 데이터가 보관

Appendix#3. 기타 정보

1. 가트너에서 정의한 3가지
   - 모니터링 : 임계값 감시, 알람 발송
   - 옵저빌리티 : 어플리케이션의 혁신을 가속화 하는 프로세스로 모니터링이 발전한것
2. 옵버질리티 툴 : 개발자가 엔지니어가 예사치 못한 시스템 동작을 보다 효과적으로 설명이 가능. 이중 opentelemetry가 데이터를 수집하는 역활을 수행하는것으로 보임. 다만 로그 수집을 하는게 아니라서 로그수집은 ekf stack으로 분할하거나 이벤트 데이터를 efk stack으로 모으는 방법도 가능할듯

옵저빌리티를 완성하기 위한 여정

1. 관련된 Tool 도입전 협업
2. 로그의 표준화 설정 필요

Reference

- 에쓰씨케이
- Elastic
- New Relic
- 이글루코퍼레이션
- IBM
- ServiceNow
- Redhat
- 와탭블로그

Revision #1
Created 21 May 2024 12:49:16 by artop0420
Updated 21 May 2024 12:51:30 by artop0420