Streaming e Armazenamento de Grandes Volumes de Dados

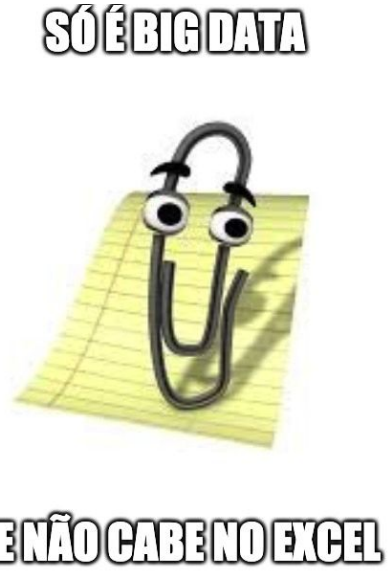
Gleicon Moraes

Roteiro

- Problema
- Fontes de dados comuns
- Lambda Architecture
- Evolução de uma arquitetura de Analytics
- Formato de dados, dimensões, dados repetidos

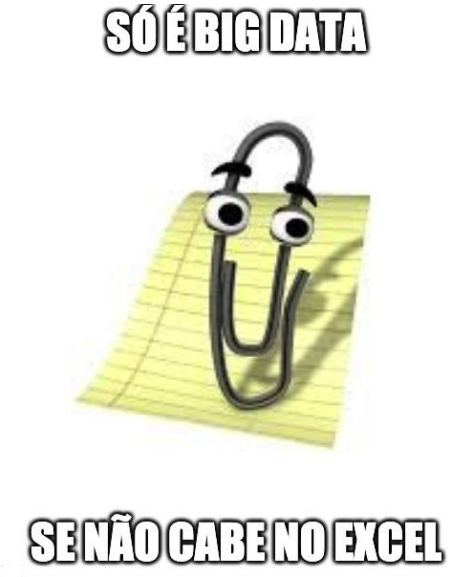
Problema

- Dados de uma ou mais aplicações não cabem em um único storage
- Novas fontes de dados são criadas constantemente
- Nem todo dado precisa estar disponível em baixa latência
- Nem todo dado é útil
- Como reagir a eventos em tempo real ?
- Como refatorar arquiteturas de dados ?



Fontes de dados comuns

- Tempo real (analytics, pixels, trackers)
- Bancos de dados
- Comunicação entre serviços
- Integração de fontes externas (DMP, importações)



Tempo real

O que parece

- Dados prontos para serem utilizados
- Volume alto mas vale a pena
- Insights frescos a cada minuto
- Magica pura

Como é

- 99% dos dados não serve para nada
- Vai ter um pico no feriado e vou gastar 12 dias migrando para outro datastore
- A razão pela qual aprendi a usar filas
- Consumidores com logica para limpar/enriquecer os dados
- Coleta em tempo real, usa daqui 2 meses.

Bancos de dados

O que parece

- O schema não é legal mas dá para usar
- Uma API na frente resolve
- XXXX é um banco de dados que podemos usar.

Como é

- Schema #%$%@
- Capacity planning
- 100000 de conexões
- API ? Mas se eu conectar direto ?
- Living la vida ETL
- A razão pela qual aprendi a usar filas

Comunicação entre serviços

O que parece

- CQRS ajuda a construir serviços robustos.
- Em vez de REST põe um KAFKA ai, se arquivar as mensagens no data lake podemos usar depois
- Vamos gastar um tempo definindo o formato das mensagens e nunca vai mudar.

Como é

- O formato da mensagem muda sempre.
- WTF de onde vem tanta mensagem ?
- As mensagens não são idempotentes
- Um KAFKA não é suficiente/Caiu o broker tudo parou.

Integração de fontes externas

O que parece

- Receber dados processados para enriquecer nosso data lake

Como é

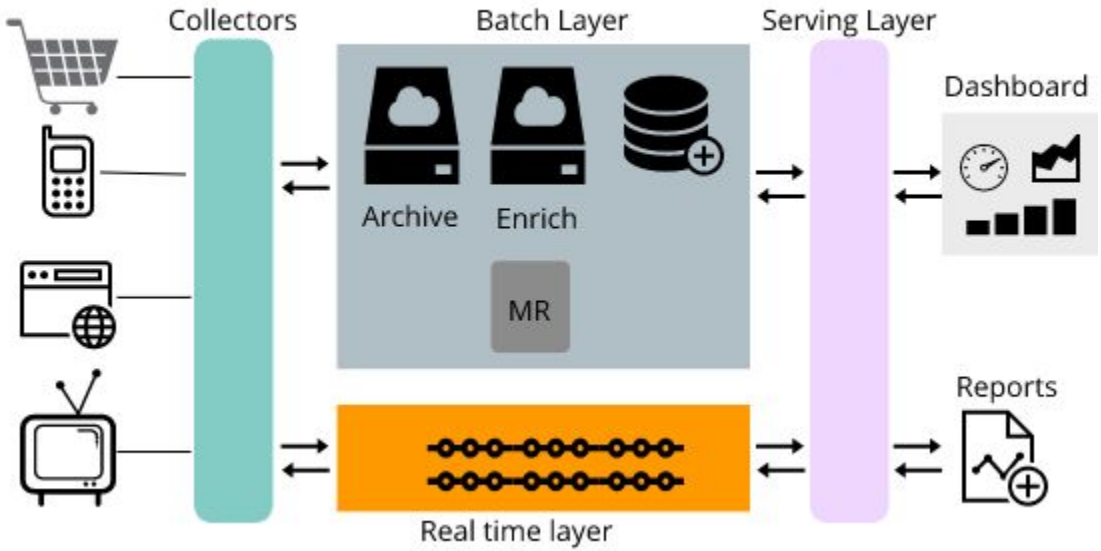
- O formato não faz sentido
- Metadados vem por email ou telegrama
- Conheça o AWS S3
- Volte aos anos 80 com FTP
- Invente seu pipeline para limpar TB de dados

Dados crescem de forma inesperada

Dados crescem de forma inesperada

- Disponibilidade
- Localidade
- Performance
- Custo
- Compatibilidade
- Single node vs Cluster

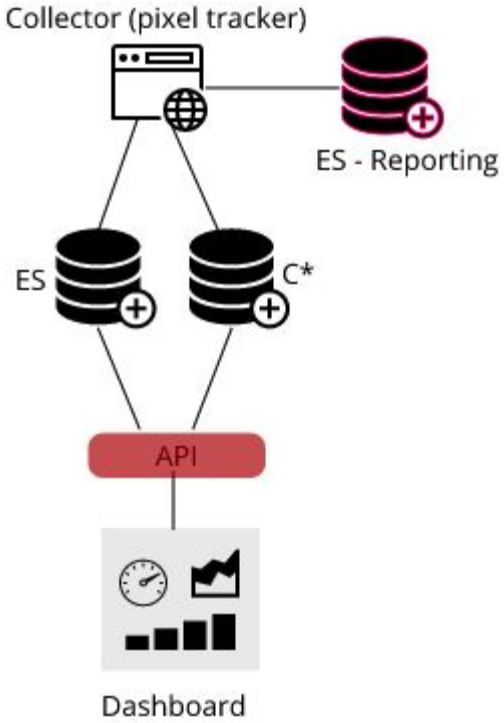
Lambda Architecture



Arquitetura mk I

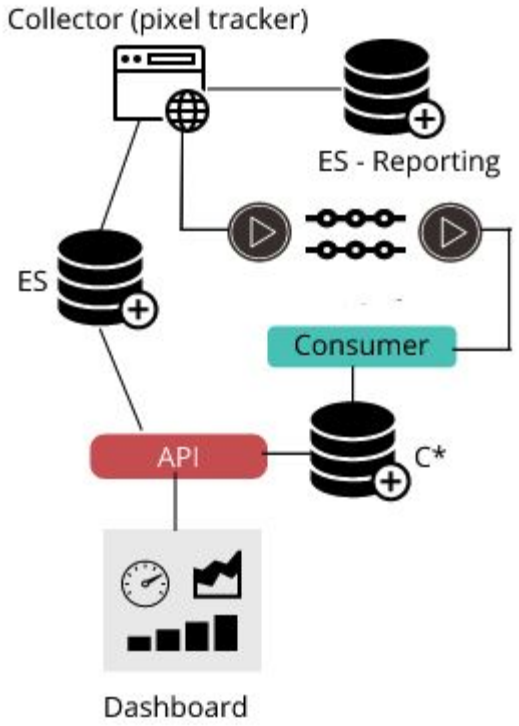
- Escritas sincronas (3)
- Raw data
- **Sensível a mudanças de workload**
- Relatórios delegados a ELK
- **Sensível ao sucesso**
- Histórico em bancos "live"

"Vamos usar o melhor DB para cada caso"



Arquitetura mk II

- Escritas sincronas (2)
- Raw data
- ~~**Sensível a mudanças de workload**~~
- Relatórios delegados a ELK
- ~~**Sensível ao sucesso**~~
- Histórico em bancos "live"
- Consumidor "sem controle"
- Usamos KAFKA para distribuir impressões

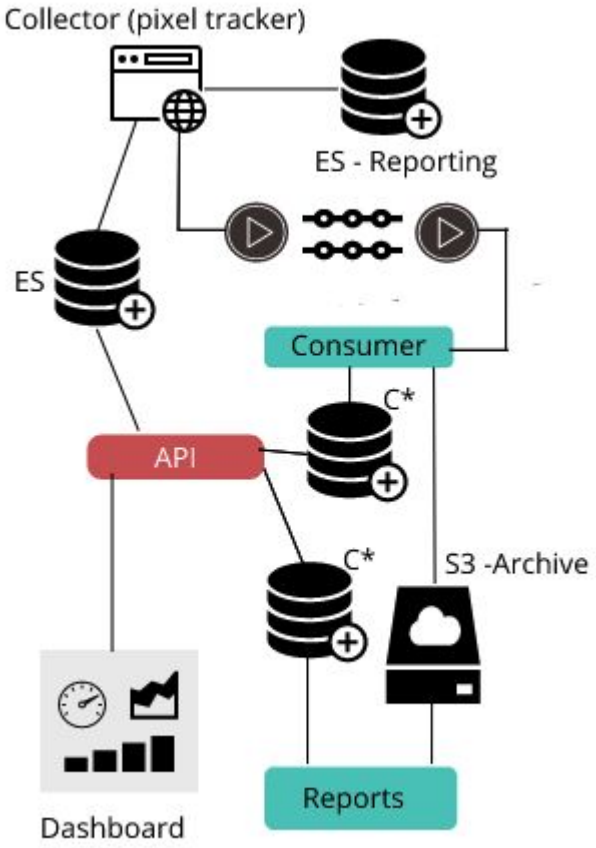


Problemas em mk I e II

- 40T documentos no ES
- 2 Cluster de 42 maquinas d2 com RAID
- Incidentes todo dia
- Cluster wide queries, timeouts
- Mesmo dado em 3 storages
- Ainda sensível à mudanças de workload
- Otimizado para tempo real em captura, mas os pipelines executam de 12 em 12h
- Pre-agregar dados era dificil e dependia de I/O de disco

Arquitetura mk III

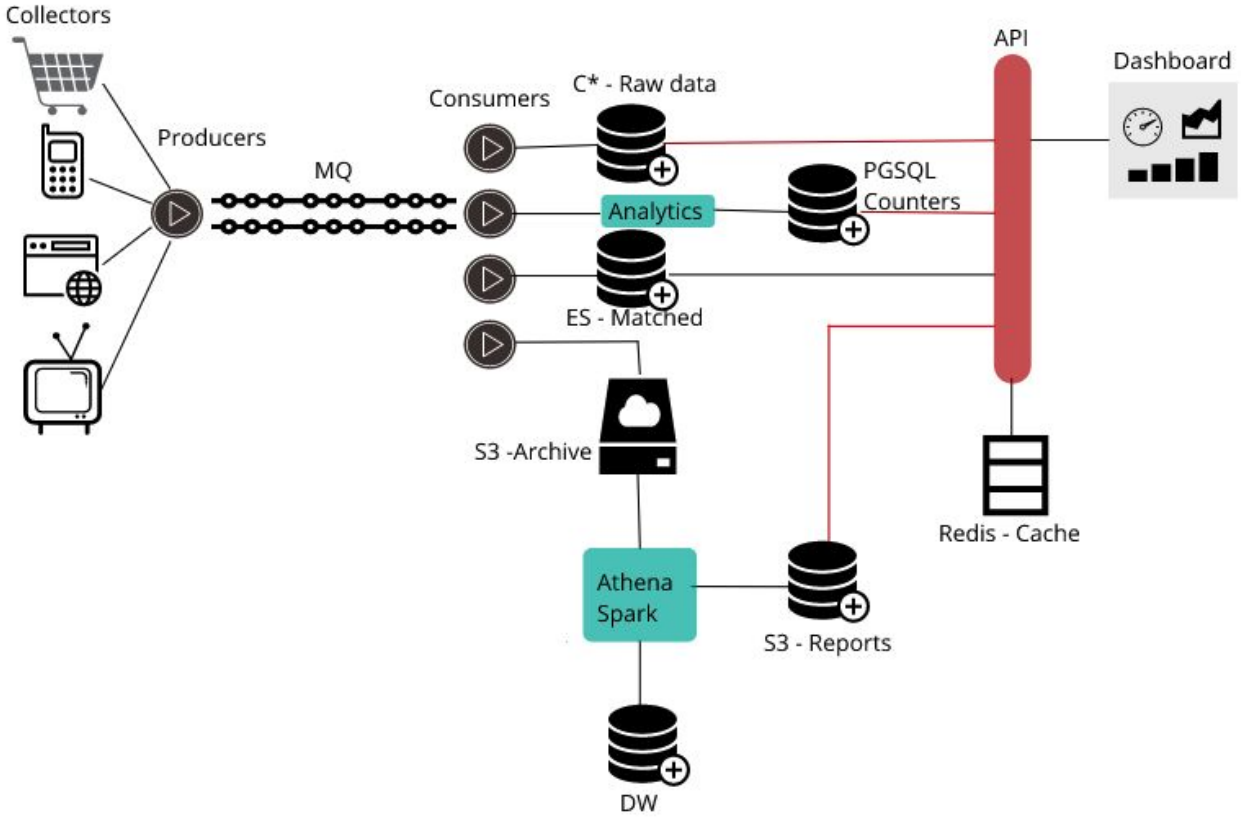
- Escritas sincronas (2)
- ~~Raw data~~ Dados arquivados em Parquet
- ~~**Sensível a mudanças de workload**~~
- Relatórios delegados a ELK
- **Sensível ao sucesso (escritas sincronas)**
- Consumidor "sem controle"
- Histórico em bancos "live"
- Histórico em S3
- Nova dependencia: Cassandra



Arquitetura mk IV

- ~~Escritas sincronas (2)~~
- ~~Raw data~~ Dados arquivados em Parquet, por data e campanha
- ~~**Sensível a mudanças de workload**~~
- ~~Relatórios delegados a ELK~~
- ~~**Sensível ao sucesso (escritas sincronas)**~~
- ~~Consumidor "sem controle"~~
- ~~Histórico em bancos "live"~~
- Histórico em S3 - Athena e Spark para relatórios
- ~~Mais uma dependência: Cassandra~~ - PGSQL para contadores
- Ruim: mantivemos Cassandra e ES, muitos bancos de dados.

Arquitetura mk IV



O que aprendemos

- Redução do uso de Elasticsearch > 60% armazenando dados em parquet no S3
- Apenas documentos "matched" (4 a 5% do total) são usados em buscas refinadas e multi-dimensionais, poderiamos sumarizar o resto dos dados.
- ES usa HyperLogLog acima de 40k docs para agregações, já estávamos com margem de erro em contadores

O que aprendemos

- Armazenar contadores no REDIS é arriscado, tempo de recuperação alto
- Armazenar contadores no PGSQL funcionou bem
- Athena funcionou bem para nossos relatórios
- Repetir os dados no S3 em diferentes dimensões é mais barato que manter todos os dados live (em média mais de 30 atributos por doc)
- O melhor formato de dados para nós foi o Parquet (tentamos CSV, JSON per line e um pouco de AVRO)

Futuro

- Todos os dados em S3, apenas dados de acesso rápido transferidos para PGSQL por streaming
- Zero ES
- Serviços menores (API e consumidores)
- Sair do KAFKA para captura de dados

Kafka é dificil ?

- Configuração de tópicos: Partições
- Client libraries
- Log Replay

Arquivamento

- /ano/mes/dia/hora (fácil de migrar entre storages)
- /campanha/
- /user_id/
- Repetir o dado é mais barato que um modelo relacional (200TB)
- Diferentes cortes, diferentes storages
- Formatos intermediarios

Conclusão

- Adotar um formato de dados no inicio é bom
- Particione seus dados
- Desconfie de soluções all-in-one
- Simplifique sua vida

Obrigado !

gleicon@gmail.com

https://github.com/gleicon

https://twitter.com/gleicon