クラウドってなんだろ？
クラウドを活かすアプリケーションの設計とは？

goo.gl/X54o2S

GCPUG Admin
Google Developers Expert
Mercari / Merpay Solution Team
@sinmetal

クラウドってなんだろ？

データセンター上に存在するハードウェアリソースをAPIを利用して、動的にリソースを借りることができる仕組み。

柔軟に迅速にリソースの提供を受けるために、料金はだいたい従量課金制で、使ったリソースの料金を払う。

クラウドは何がうれしいのか？

- H/Wを調達するイニシャルコストがかからない
- H/Wをリプレースする必要がない
- いつでもマシンリソースの構成を変更できる
  - 負荷に合わせて増減
  - 月末だけ追加
  - CGのレンダリングや機械学習する時だけ
  - リリースする時だけ2倍のリソースを用意

クラウドのレイヤー

- SaaS（Software as a Service）
- PaaS（Platform as a Service）
- DBaaS (Database as a Service)
- FaaS (Function as a Service)
- IaaS（Infrastracture as a Service）

言葉はどんどん増えていってる

クラウドは何がうれしいのか？

- クラウドサービスで提供してもらうレイヤーも選択できるし、組み合わせを行うこともできる
  - Web API -> App Engine
  - 動画のエンコード -> Compute Engine
  - 動画の保存 -> Cloud Storage
  - ログの解析 -> BigQuery

クラウドは壊れないのか？

- クラウドサービスが構築されているデータセンターにはもちろんH/Wが存在し、日々壊れていっている。
- SLAのAvailabilityはリージョンやゾーン全体が落ちたのが対象になっているので、1つのインスタンスがエラーを返したものは含まれない

クラウドは不死鳥のように蘇る

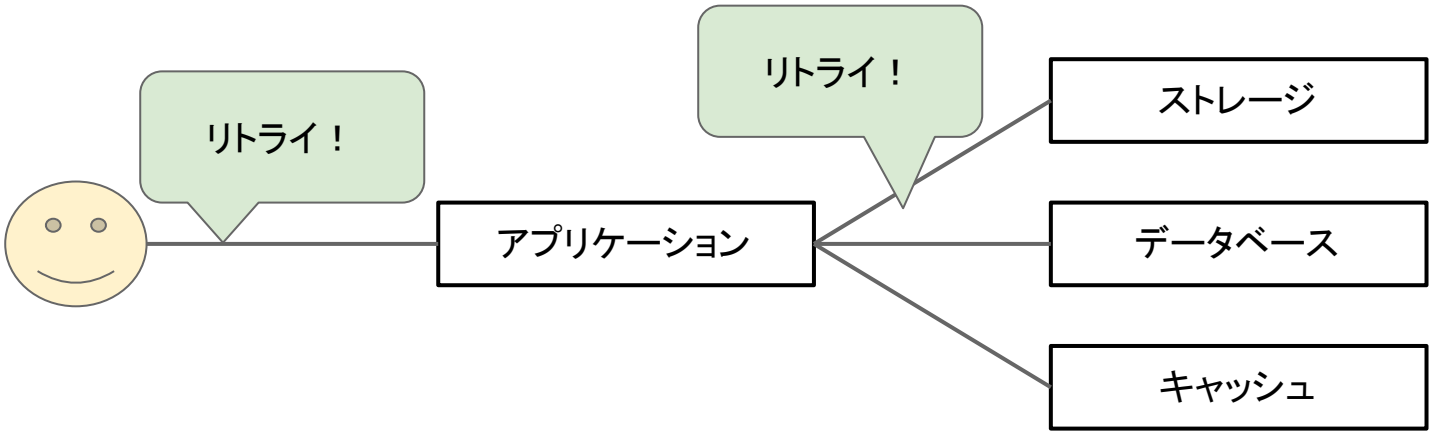
- クラウドサービスは壊れないのではなく、壊れてもすぐ蘇るもの
- H/Wが壊れても、すぐ別のH/Wでもう一度生成してくれる
  - ただ、その瞬間そのH/Wで動いていたものは失われる

クラウドを活かすには？

- アベイラビリティを上げる
  - クラウドサービスは瞬間的に壊れるのを前提に設計する
    - リトライ
    - リクリエイト

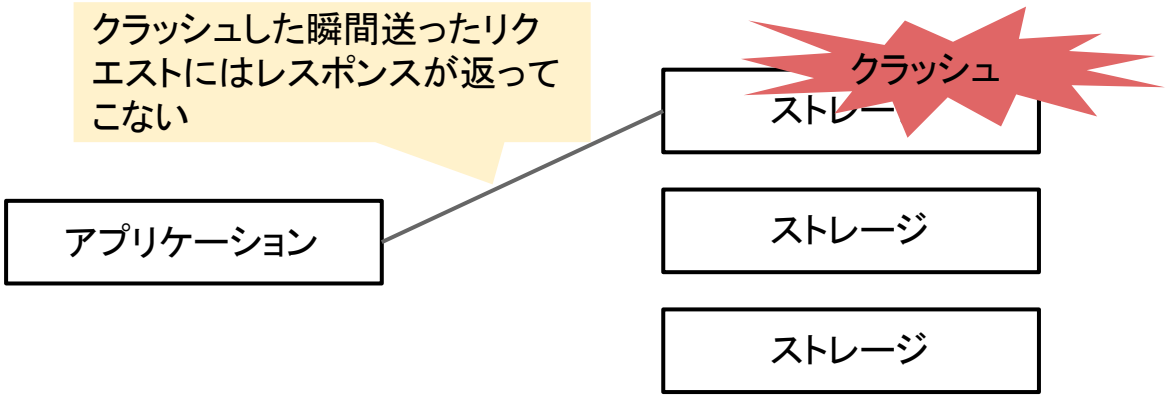
リトライ

- 間にネットワークが入れば、エラーは起こる
  - いろんなレイヤーでリトライしよう！



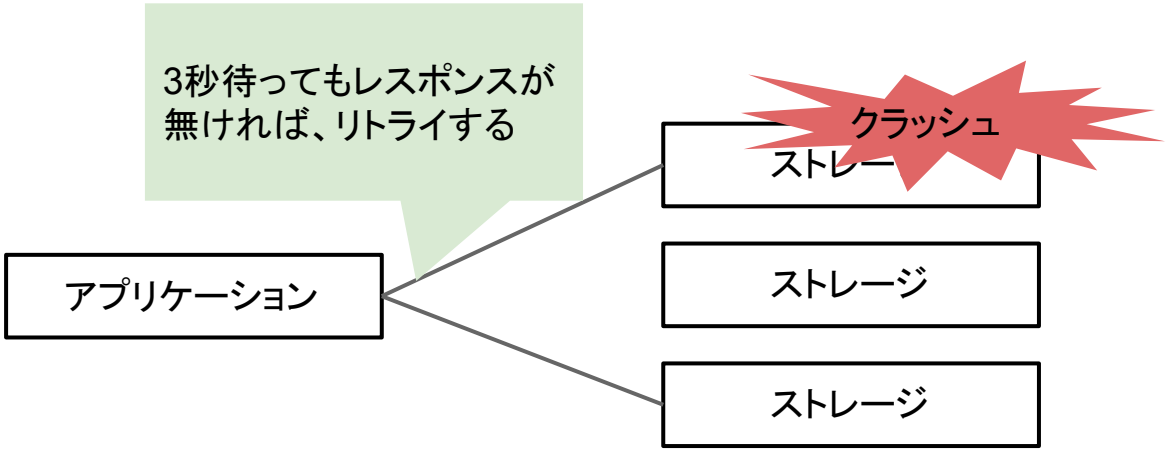
リトライ

- クラウドのサービスにはレプリカが存在するものが多い
- 1つ壊れても、他のものたちが応えてくれる
- でも、壊れる瞬間に繋いでいたものには応答が返ってこない



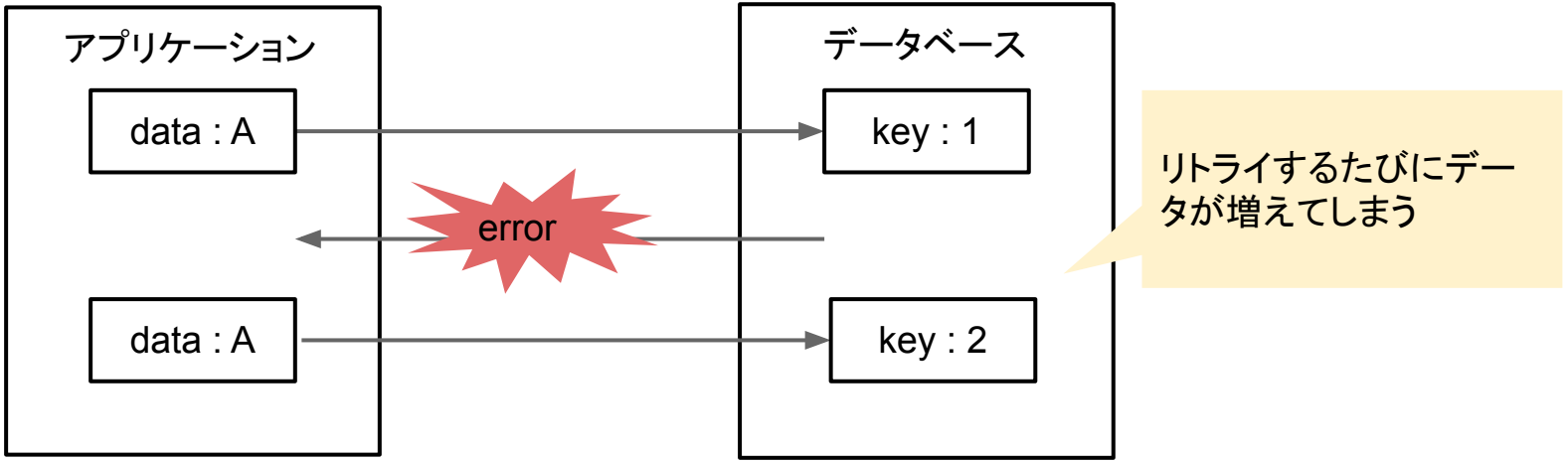
リトライ

- 適切なタイムアウトを設定して、リトライする



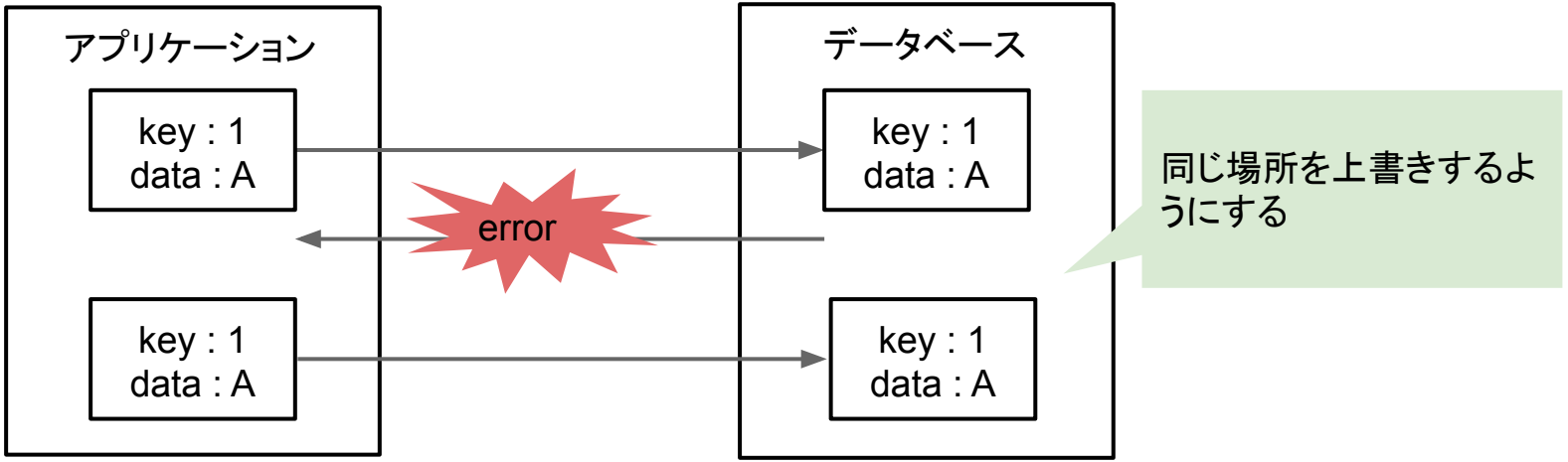
冪等性 (1)

- リトライ時に複数データを作成してしまわないように冪等性を考える



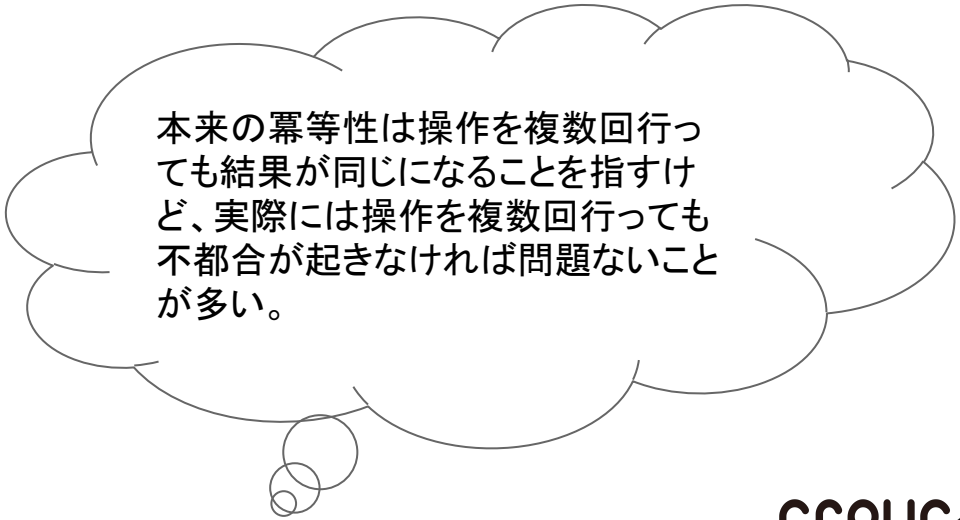
冪等性 (2)

- リトライ時に複数データを作成してしまわないように冪等性を考える



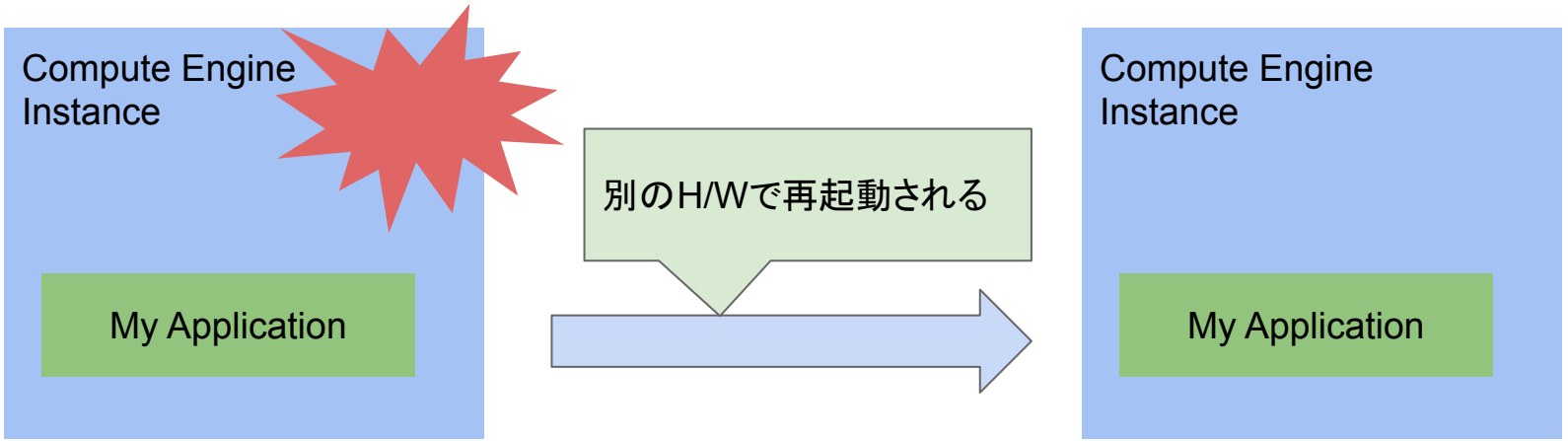
冪等性 (3)

- Keyを最初に決めて、同じ場所を上書きするようにする
- すでに更新されていたら、処理をスキップする



リクリエイト (1)

- 自分のアプリケーションが乗っているH/Wが壊れたら？



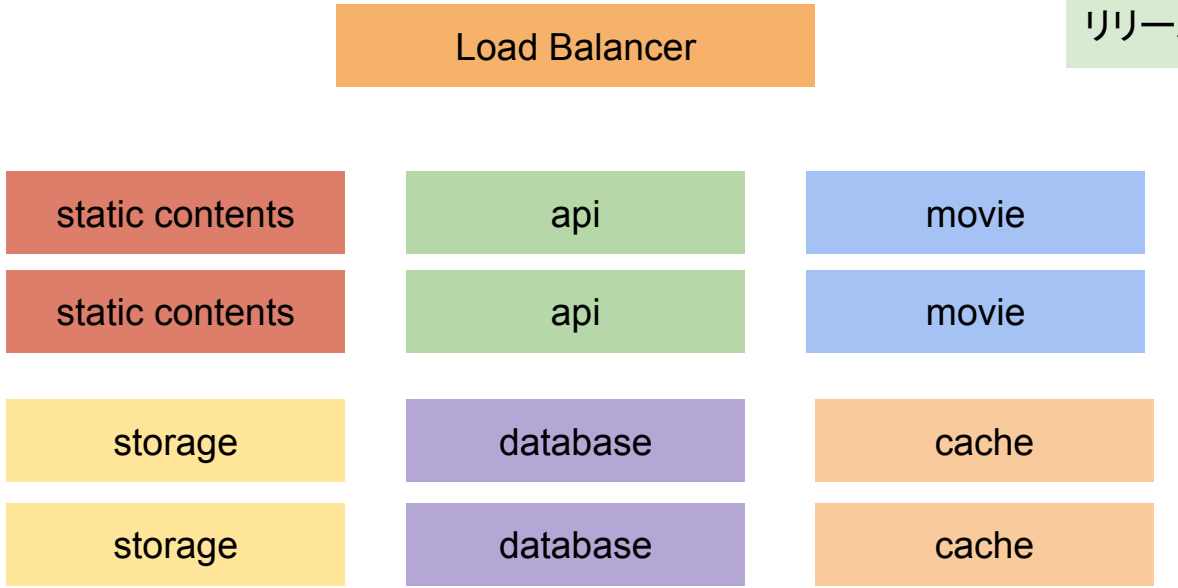
リクリエイト (2)

- 複数インスタンス用意しておく
  - 自分のアプリケーションが0台にならないように
- 起動にかかる時間を短くする
  - なるべく早く役割を果たせるように
- 状態はなるべく持たないようにする
  - メモリは消し飛ぶので、状態はなるべく外へ

リクリエイト (3)

- 役割を小さくして、初期処理を少なく
- クラッシュした時の影響範囲を限定的に

マシンスペック
アベイラビリティ
スケーラビリティ
トラフィック
リリース頻度

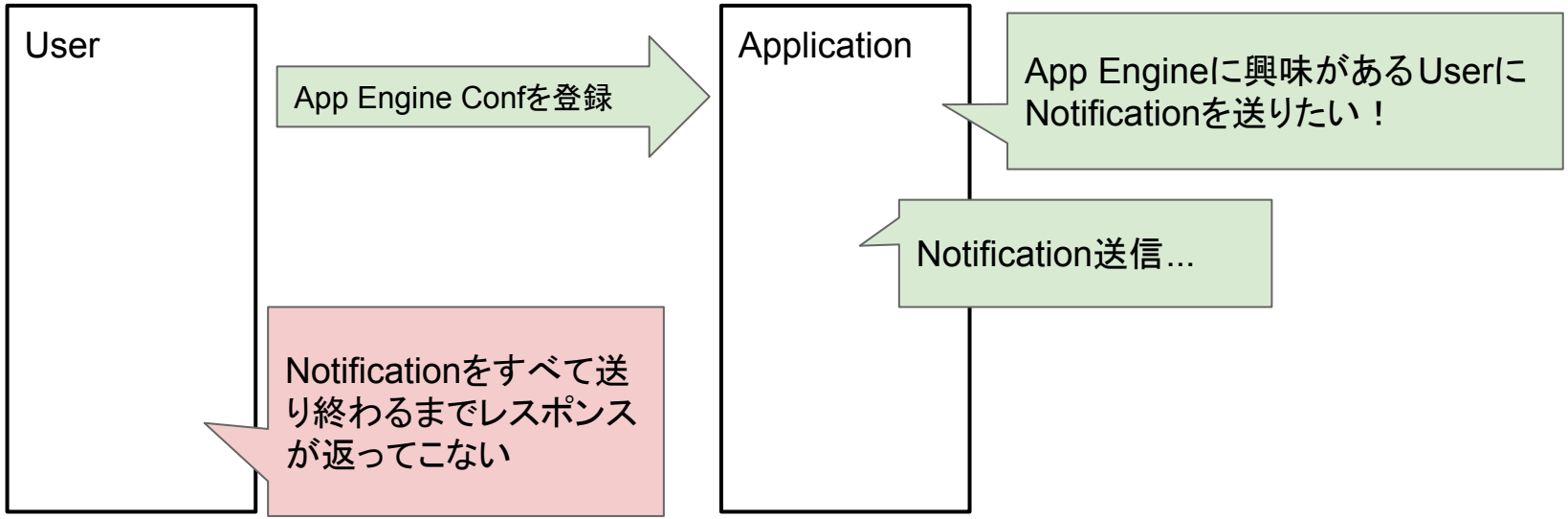


クラウドを活かすには？

- パフォーマンスを上げる
  - 非同期処理
  - 分散処理

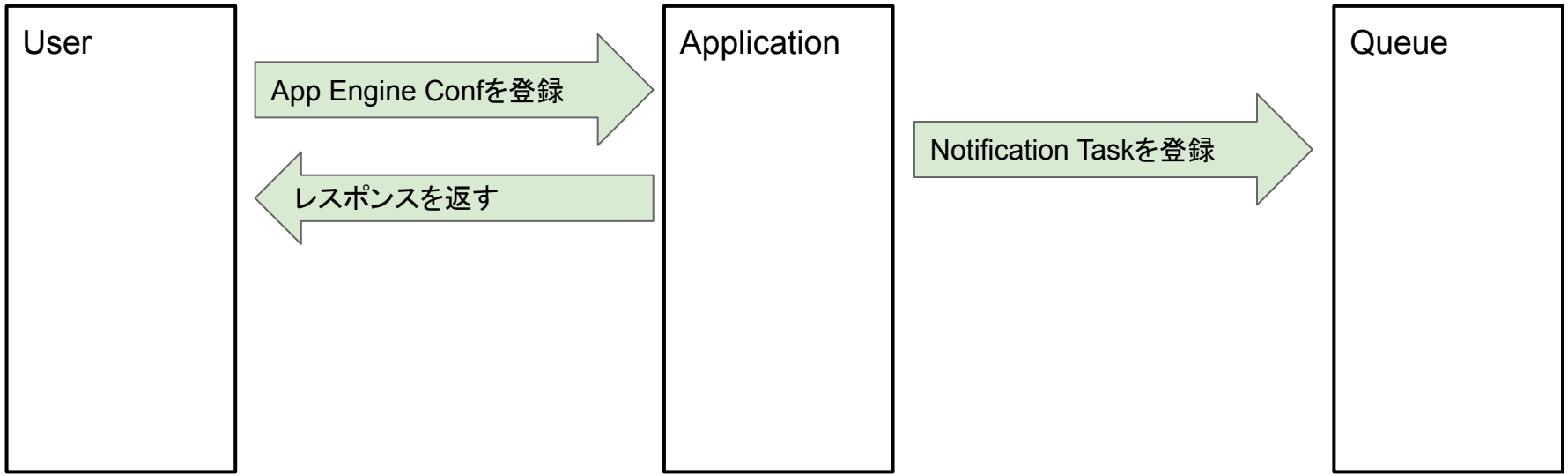
非同期処理, 分散処理 (1)

- 同期的に処理する必要がない処理は非同期に処理した方が安くパフォーマンスを上げれる



非同期処理, 分散処理 (2)

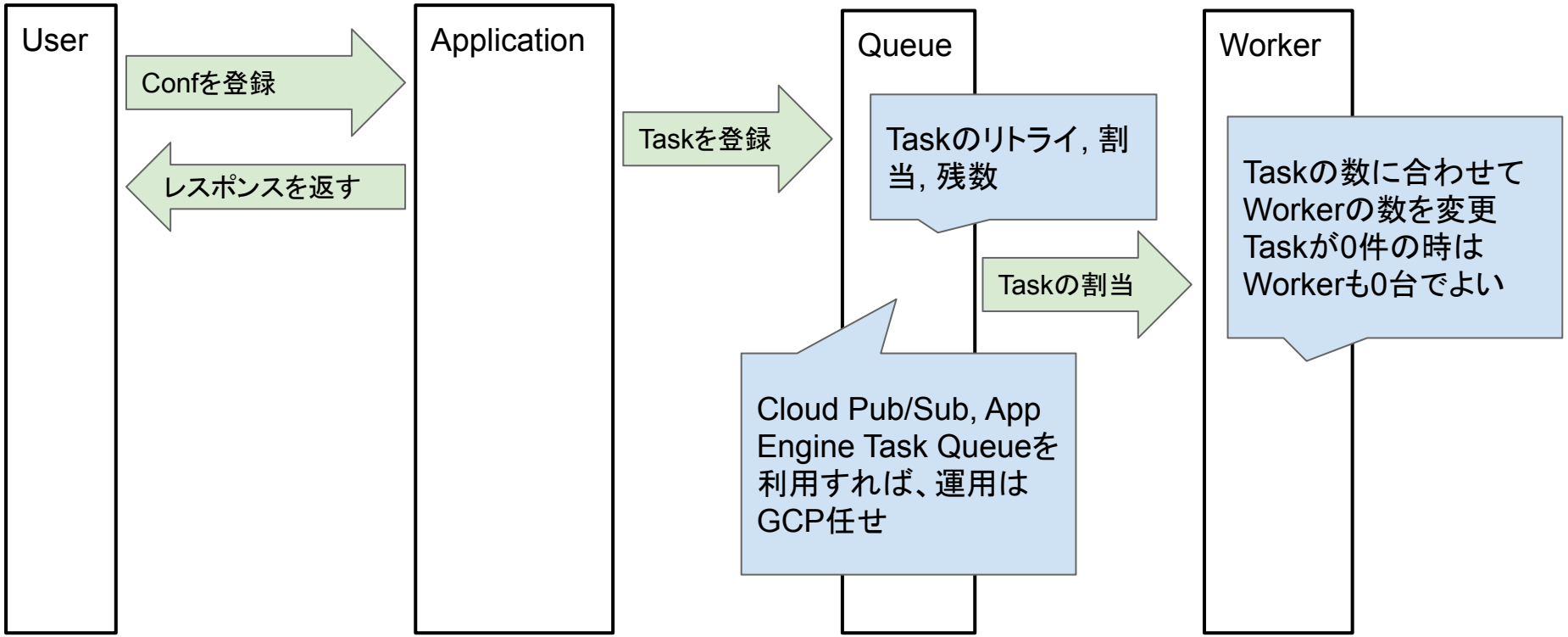
- 非同期に処理をする時に便利なのがQueueのサービス
- GCPだとCloud Pub/Sub, App Engine Task Queueがある



非同期処理, 分散処理 (3)

- Queueを挟むことで、処理の粒度でWorkerを分離できる
  - Workerのマシンスペックを変更できる
  - Workerの生成タイミングを遅延できる
  - Taskの管理をQueueがしてくれる
    - Workerへの割当
    - リトライ

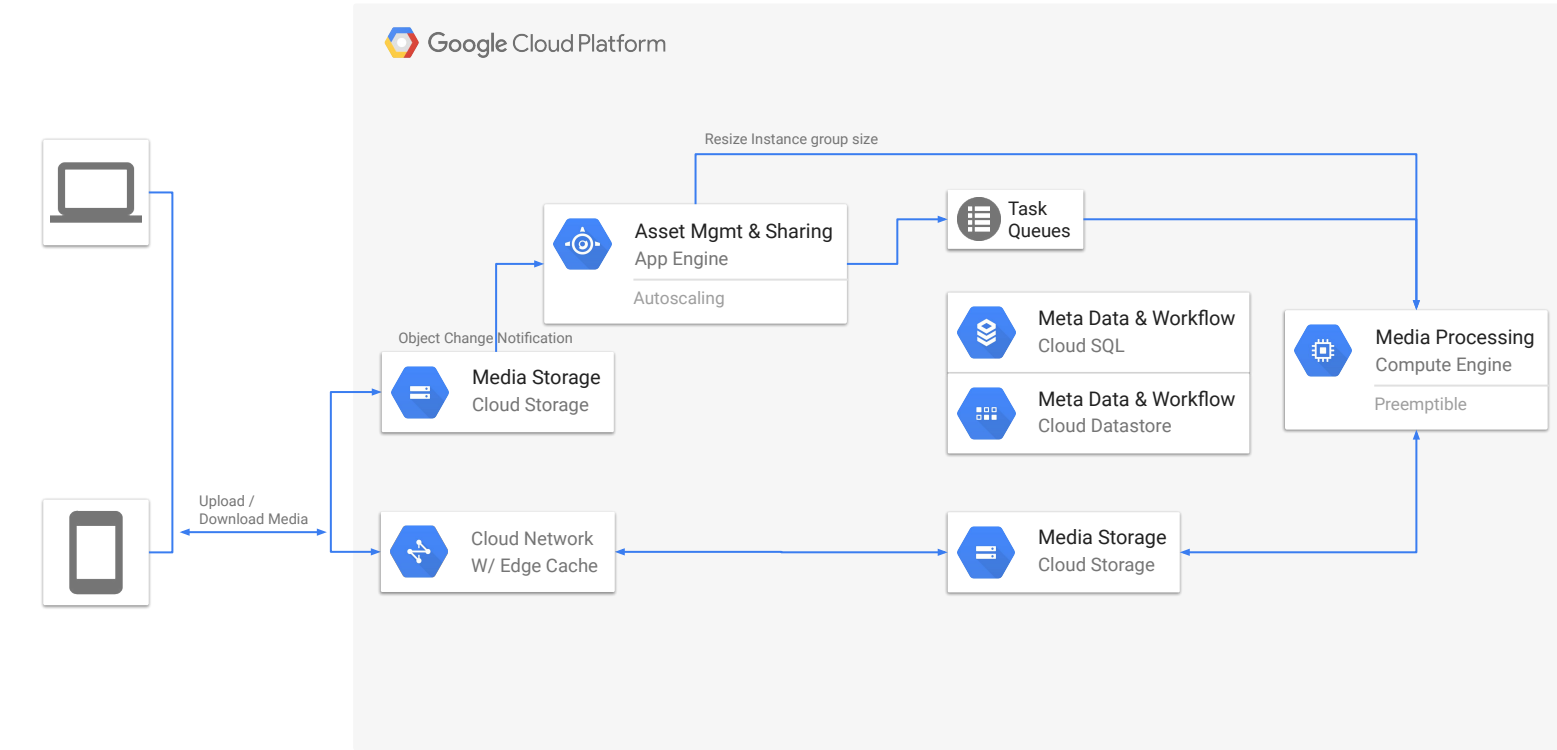
非同期処理, 分散処理 (4)


User
Application
Queue
Worker
Confを登録
レスポンスを返す
Taskを登録
Taskのリトライ, 割当, 残数
Taskの割当
Taskの数に合わせてWorkerの数を変更
Taskが0件の時はWorkerも0台でよい
Cloud Pub/Sub, App Engine Task Queueを利用すれば、運用はGCP任せ

非同期処理, 分散処理 (5)

- 非同期処理, 分散処理をする時に気を付けること
  - タスクを小さくしておく
    - リトライ範囲を小さく
    - 分割してたくさんにしないと分散できない
    - 巨大なCSV -> たくさんの小さなCSV

TaskQueueとプリエンプティブVMを利用して、動画の加工処理を行う分散アーキテクチャ

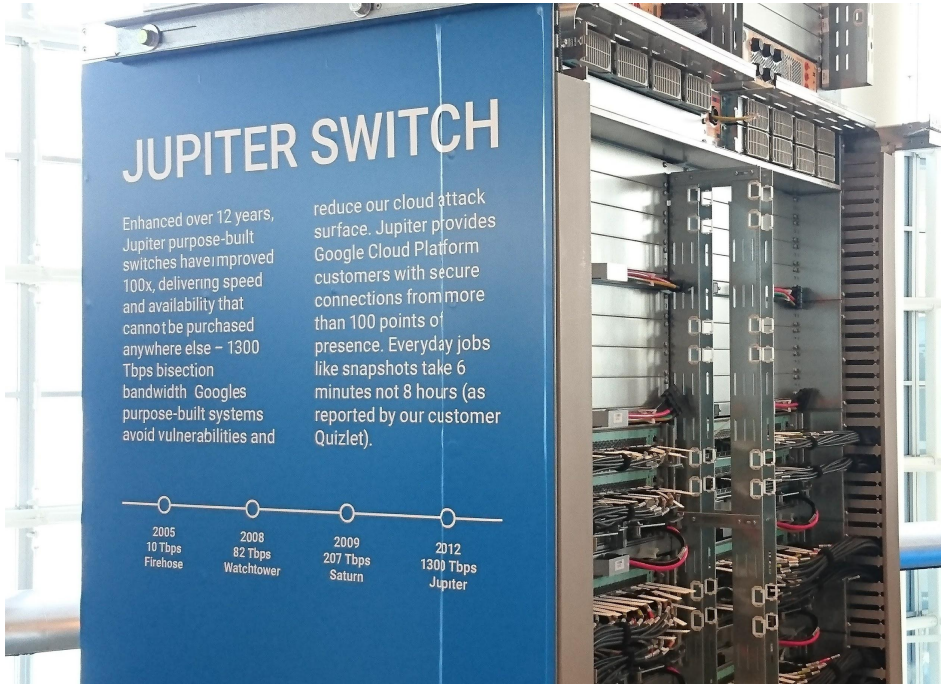


コンパクトな環境

- 必要な時に必要なだけリソースを用意するには、環境がコンパクトな方が便利
  - Container
    - Docker
    - Borg (Google内部のみ)
  - Binary
    - Go

高速なネットワーク

- 分散したタスクの分配や、コンテナの配備のために、高速なネットワークが必要



めんどうだったら

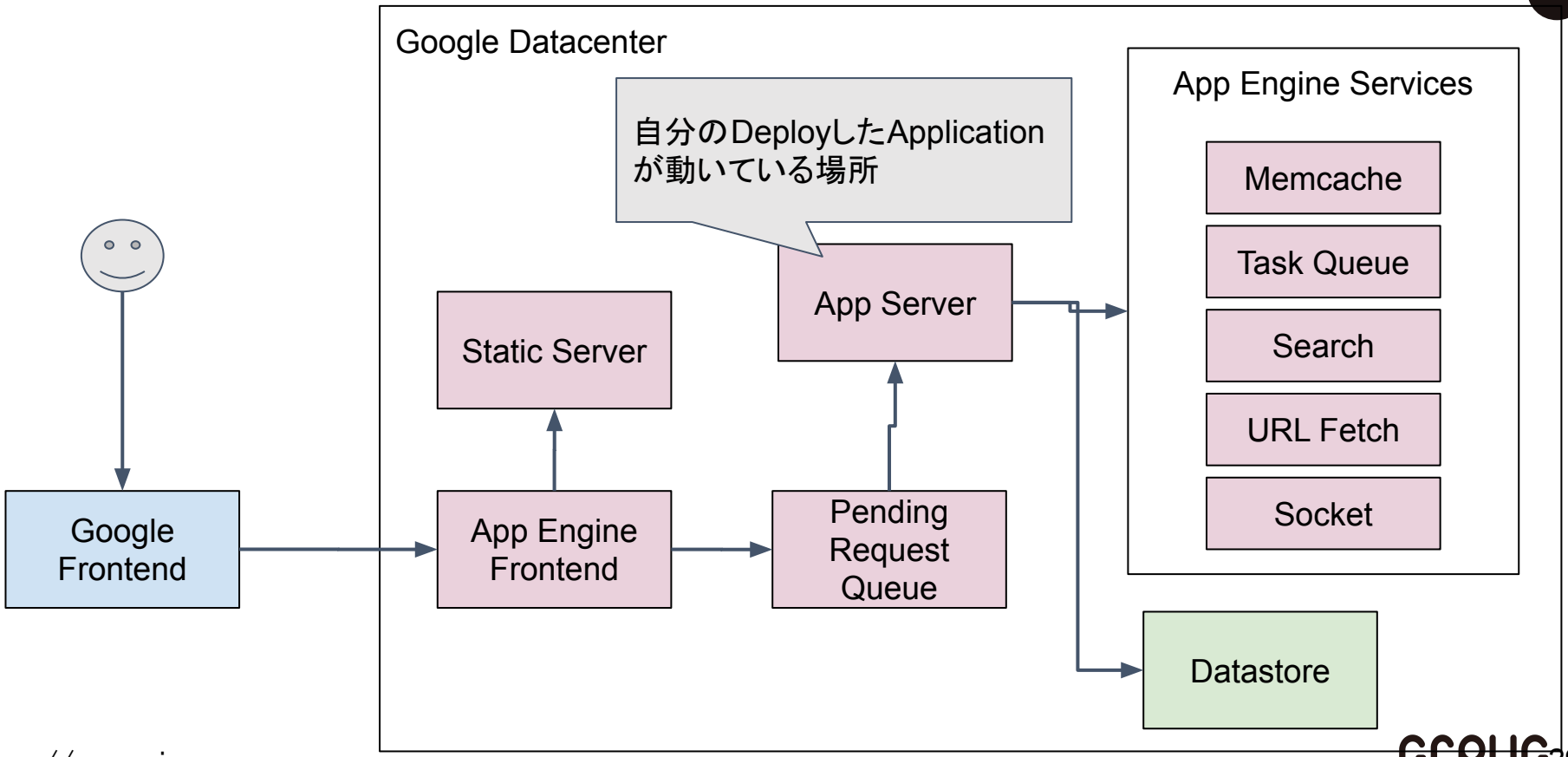
- GCPのフルマネージドサービスにある程度任せてしまおう
  - App Engine
  - BigQuery
  - Cloud Dataflow

GCPのフルマネージドサービスはクラウドに最適化されているものたちなので、アーキテクチャはとても参考になる。

Google App Engine

- Web ApplicationのためのPaaS
- 非常に高いスケーラビリティ
  - デフォルトでオートスケール
- Versioning
- Task Queue

App Engine Architecture



Google BigQuery

- データを分析するためのフルマネージドサービス
- SQLを利用してインタラクティブな分析が可能
- TB単位のデータでも数秒でフルスキャン

Google BigQueryは、なぜはやいのか？

- データを元から分割して大量のHDDに保存している
- クエリ実行時に動的に数千台のインスタンスを起動する

Colossus

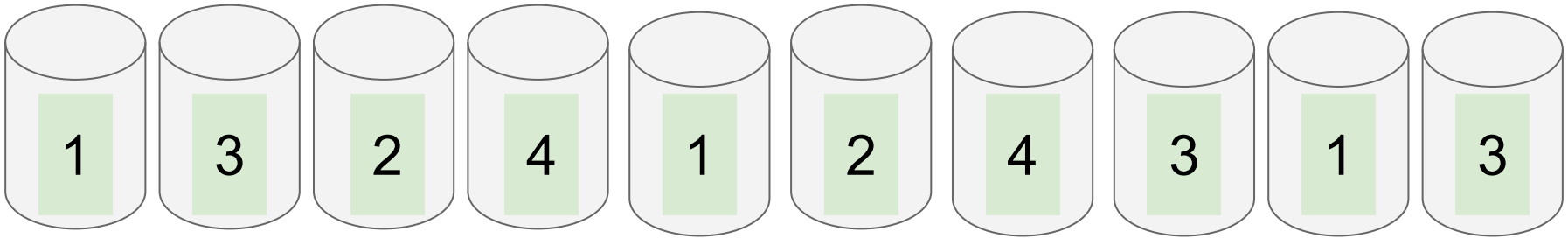
- Googleが作成した分散ファイルシステム
  - Google File Systemの後継だが、名前だけが公開されていて、中身は謎

Colossusの雰囲気

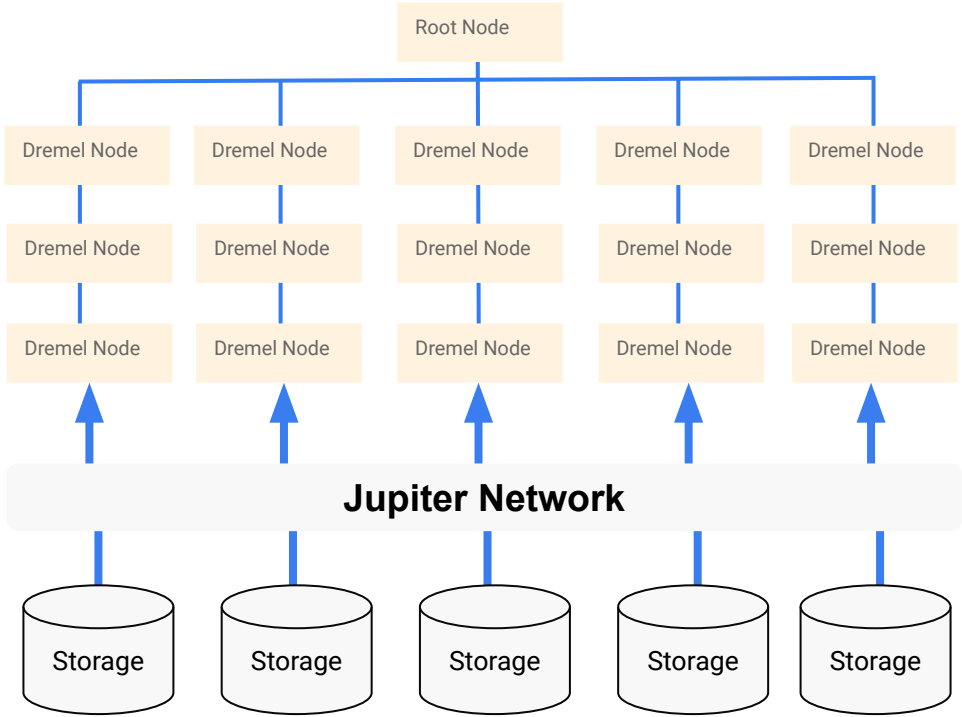
File

1 2 3 4

Fileを分割して複数のDiskに複製して保存する。



Dremel Architecture (BigQueryの裏の仕組み)



HDDから秒間読み込めるデータ容量は百数十MBなので、データを予め100MBほどに分割して、別々のHDDに保存しておく。

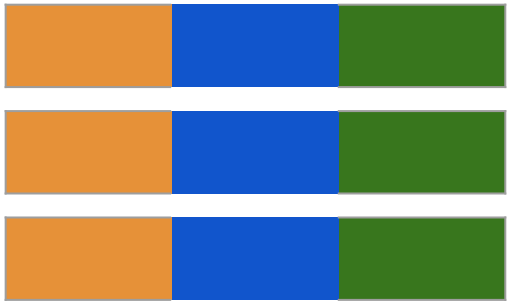
クエリ実行時に、必要なテーブルが保存されているHDDに対してDremel Nodeを割り当て、分散処理を行っていく

HDDにHotspotがある場合に備えて、同じファイルを複数のStorageに問い合わせて、最初にレスポンスが返ってきたものを採用している

列指向ストレージ

- 分析時にすべてのカラムを読み込む必要があることは少ないので、カラムごとに分けて保存して、クエリに必要なカラムのみ読み出している
  - カラムの型が決定されるので、圧縮効率も最適化できる

行指向

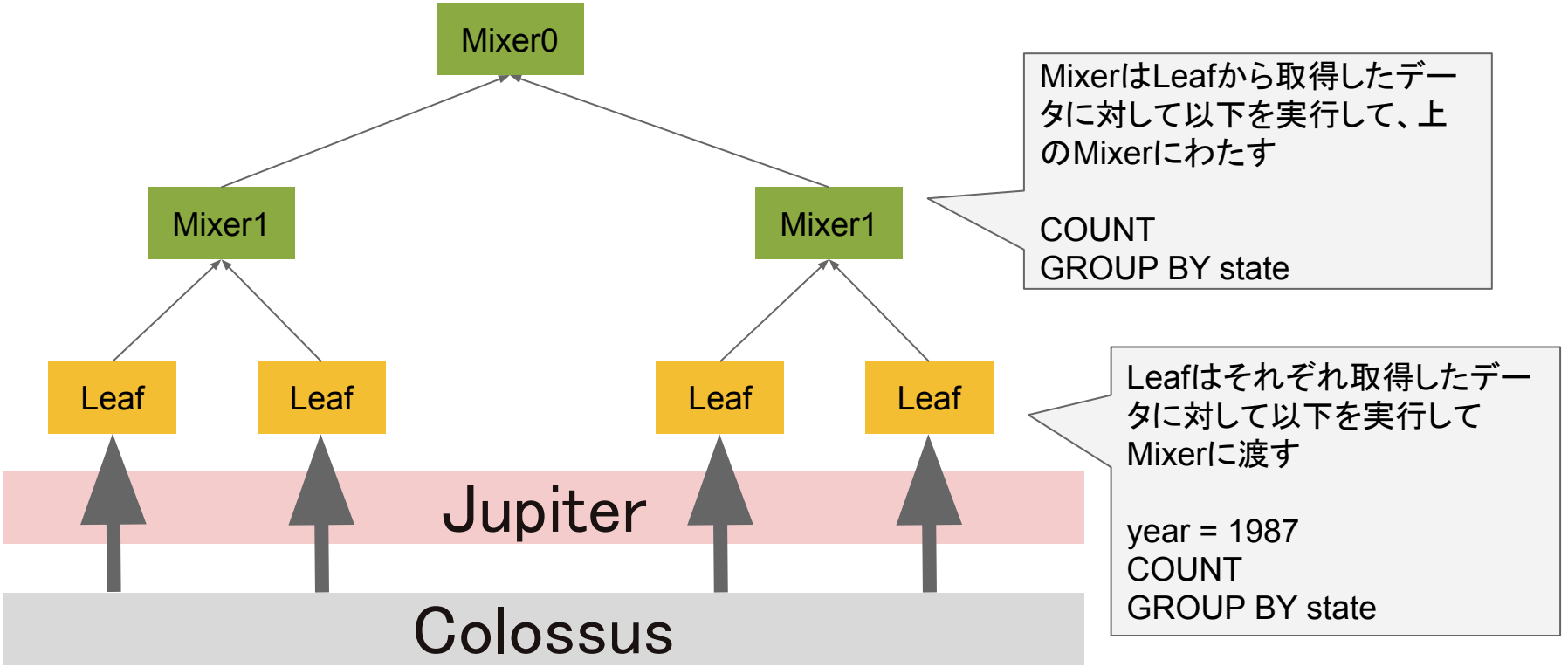


列指向

Dremel

```
SELECT
  state,
  COUNT(1) AS count
FROM
  `bigquery-public-data.samples.natality`
WHERE
  year = 1987
GROUP BY
  state
ORDER BY
  count DESC
```

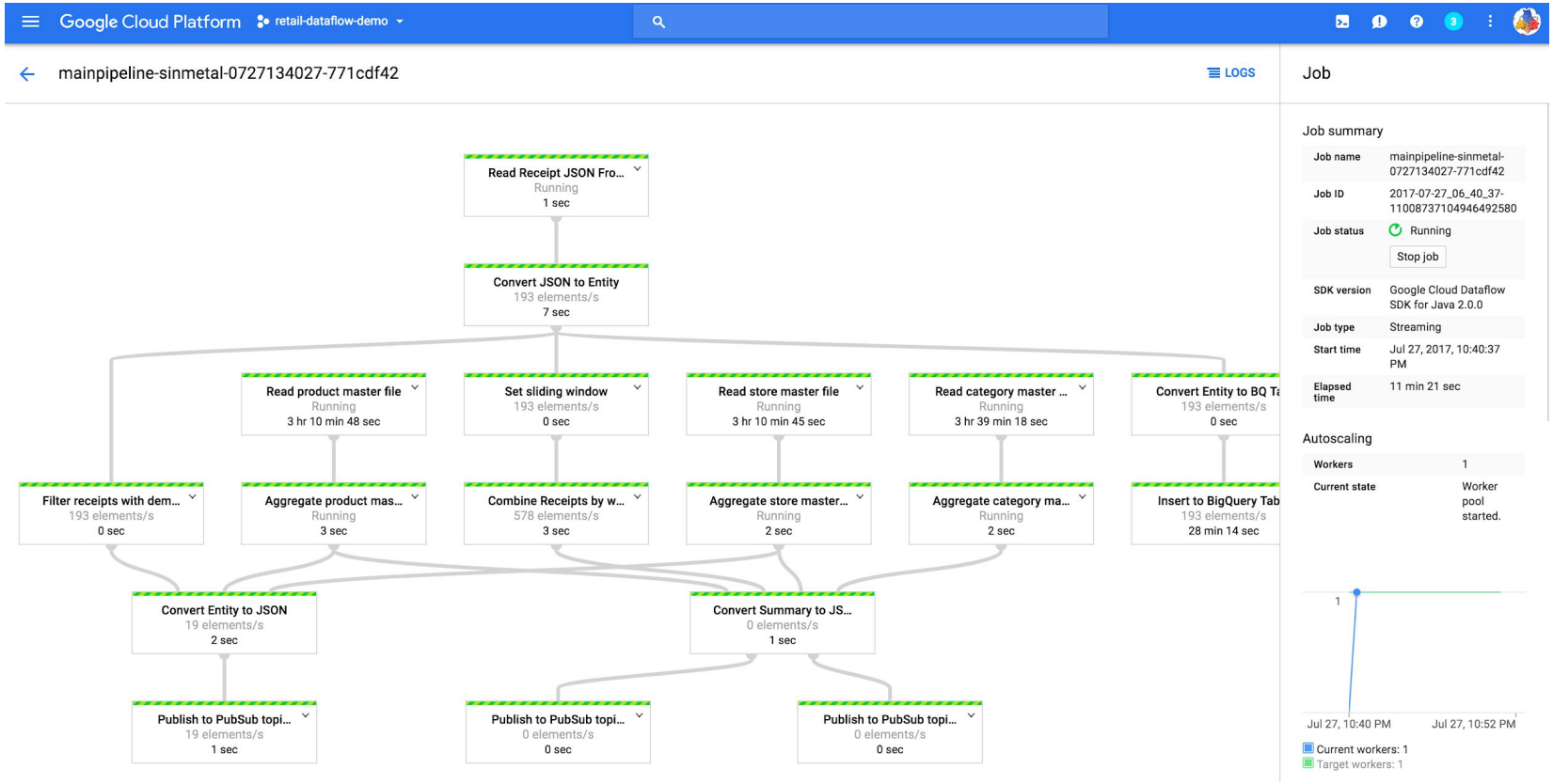
Dremel v1 Query Processing



Google Cloud Dataflow

- パイプライン処理をフルマネージドで行うサービス
- バッチ処理、ストリーミング処理、どちらも可能
- Java or Python SDK (Apache Beam)

Google Cloud Dataflow



https://gcpug.jp

GCPUG39
Google Cloud Platform User Group

Google Cloud Dataflow

- BigQueryではやりにくい処理を任せられる
  - SQLではやりづらいこと
    - 形態素解析
    - 外部のAPIの実行
    - 入力されたデータを複数の場所に出力
  - ストリーミング処理

Thanks you

Slackで待ってます！

https://slack.gcpug.jp

Resources

Google Cloud Platformの謎テクノロジーを掘り下げる

Google Cloud Storage: tips for reliability, performance and scalability (Google Cloud Next '17)

Google Cloud Dataflow(Python) Overview