

TC-03 テクノロジートラック

AWSビッグデータソリューション

Amazon Redshift, Amazon EMR, Amazon DynamoDBのご紹介

Yifeng Jiang
Solutions Architect
Amazon Data Services Japan



アジェンダ

1. AWSのサービス全体像とビッグデータ関連サービス
2. 解剖ビッグデータ
3. Getting Started with Big Data Services
 - Amazon Redshift
 - Amazon Elastic MapReduce
 - Amazon DynamoDB
4. Practical Deep Dive

アジェンダ

1. AWSのサービス全体像とビッグデータ関連サービス
2. 解剖ビッグデータ
3. Getting Started with Big Data Services
 - Amazon Redshift
 - Amazon Elastic MapReduce
 - Amazon DynamoDB
4. Practical Deep Dive

AWSサービスの全体像 - 40以上のサービス

 **amazon web services** | Partner Network Technology Partner / Consulting Partner **Ecosystem**  **aws marketplace**

Management & Administration



自動化とデプロイメント



データ分析



コンテンツ配信



アプリケーションサービス



コンピューート処理



ストレージ



データベース



ネットワーク

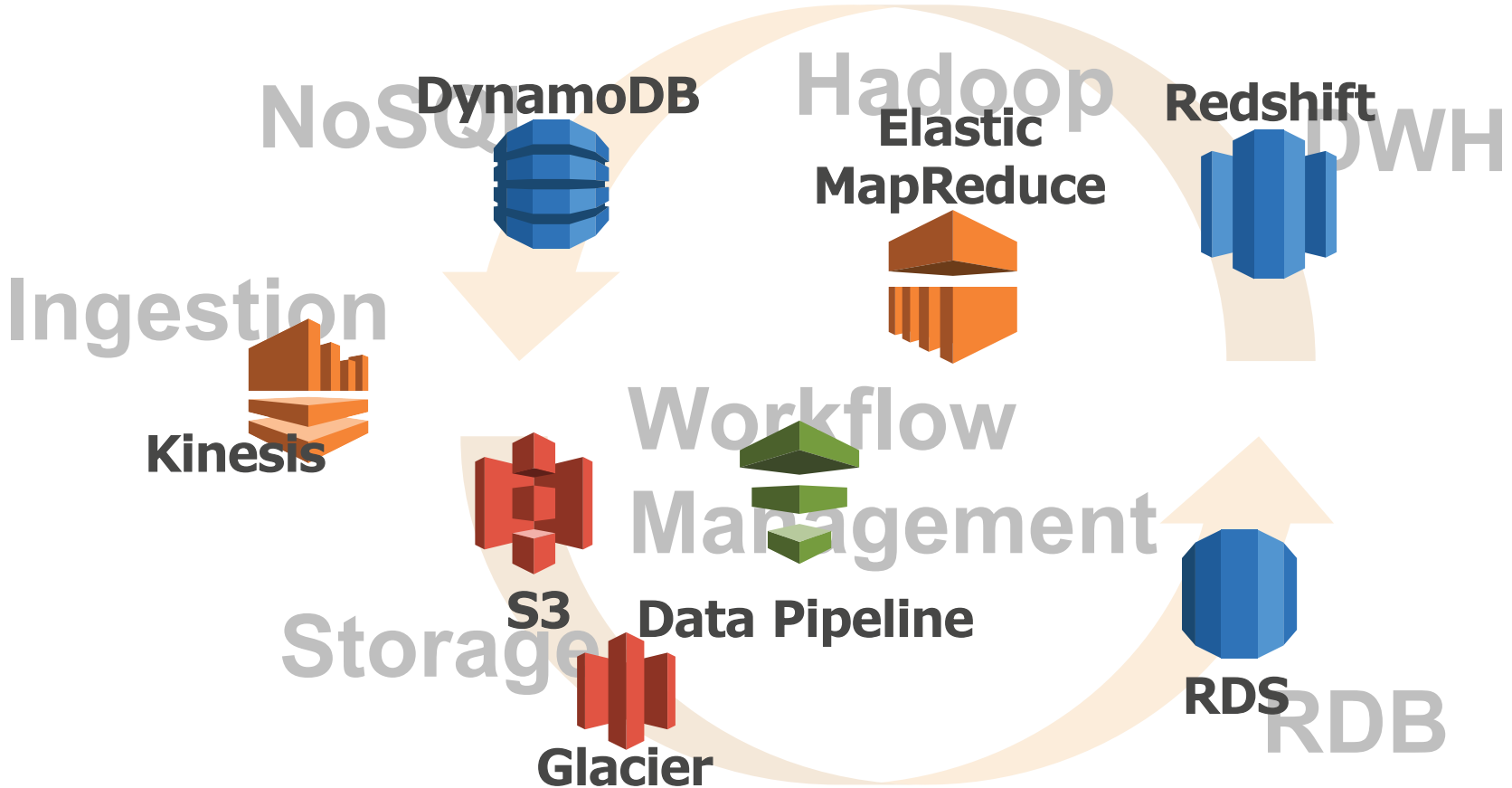


AWSグローバルインフラ



Regions / Availability Zones / Contents Delivery POPS

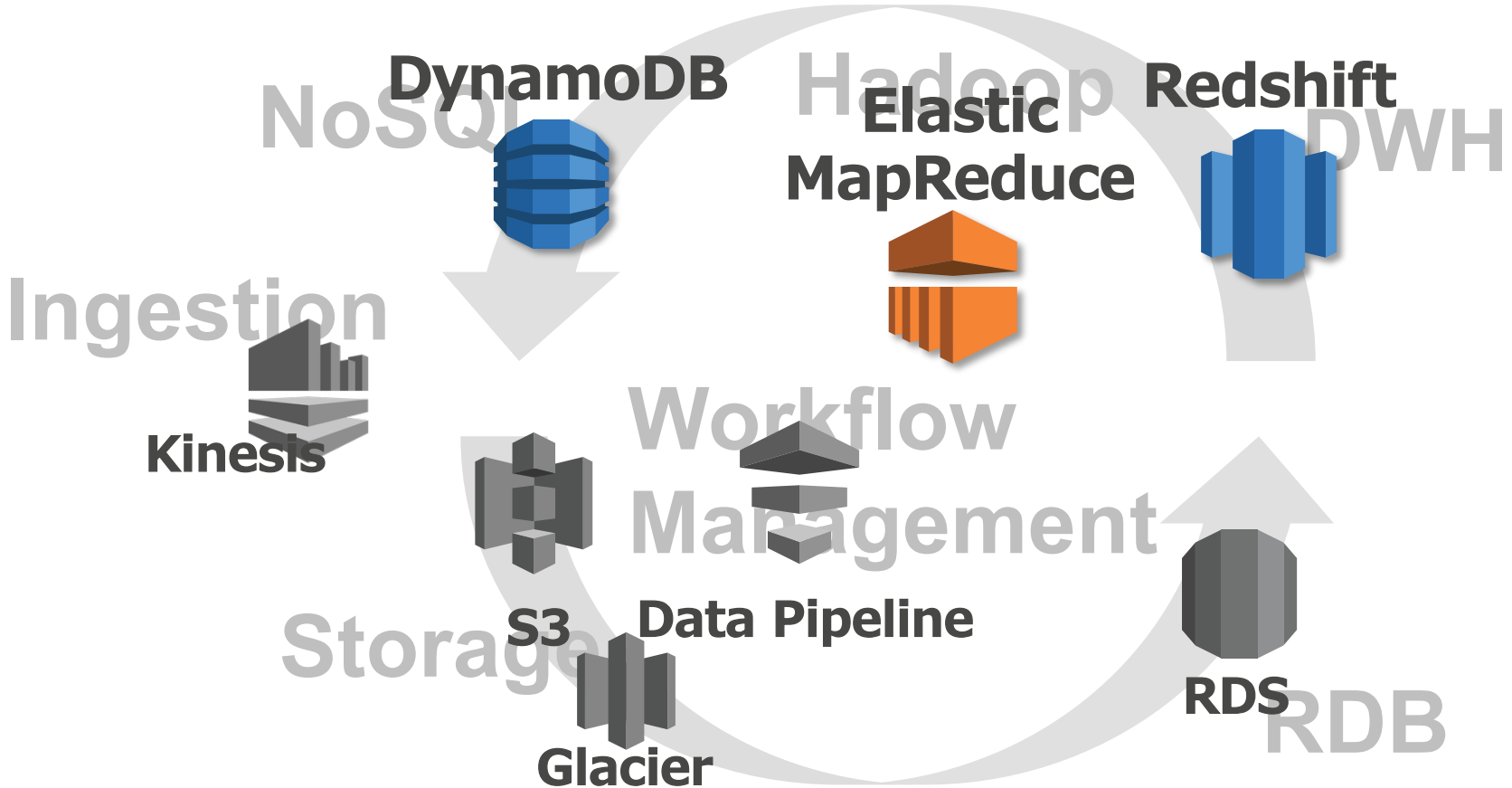
Big Data services on AWS



分析やビッグデータもAWS

～小さく素早くはじめて、どこまでもスケールできます～

Big Data services on AWS



Big Data services on AWS



今日はこれらのサービスの役割や使い分け、組み合わせ方をメインに取り上げていきます。



アジェンダ

1. AWSのサービス全体像とビッグデータ関連サービス
2. 解剖ビッグデータ
3. Getting Started with Big Data Services
 - Amazon Redshift
 - Amazon Elastic MapReduce
 - Amazon DynamoDB
4. Practical Deep Dive

データ活用の4つのステップ

1. あつめる

- 多数のアプリケーションサーバーやクライアント、デバイスからのデータ収集

2. ためる

- 安全でコスト効率よく、かつ利用しやすい形でデータを保存

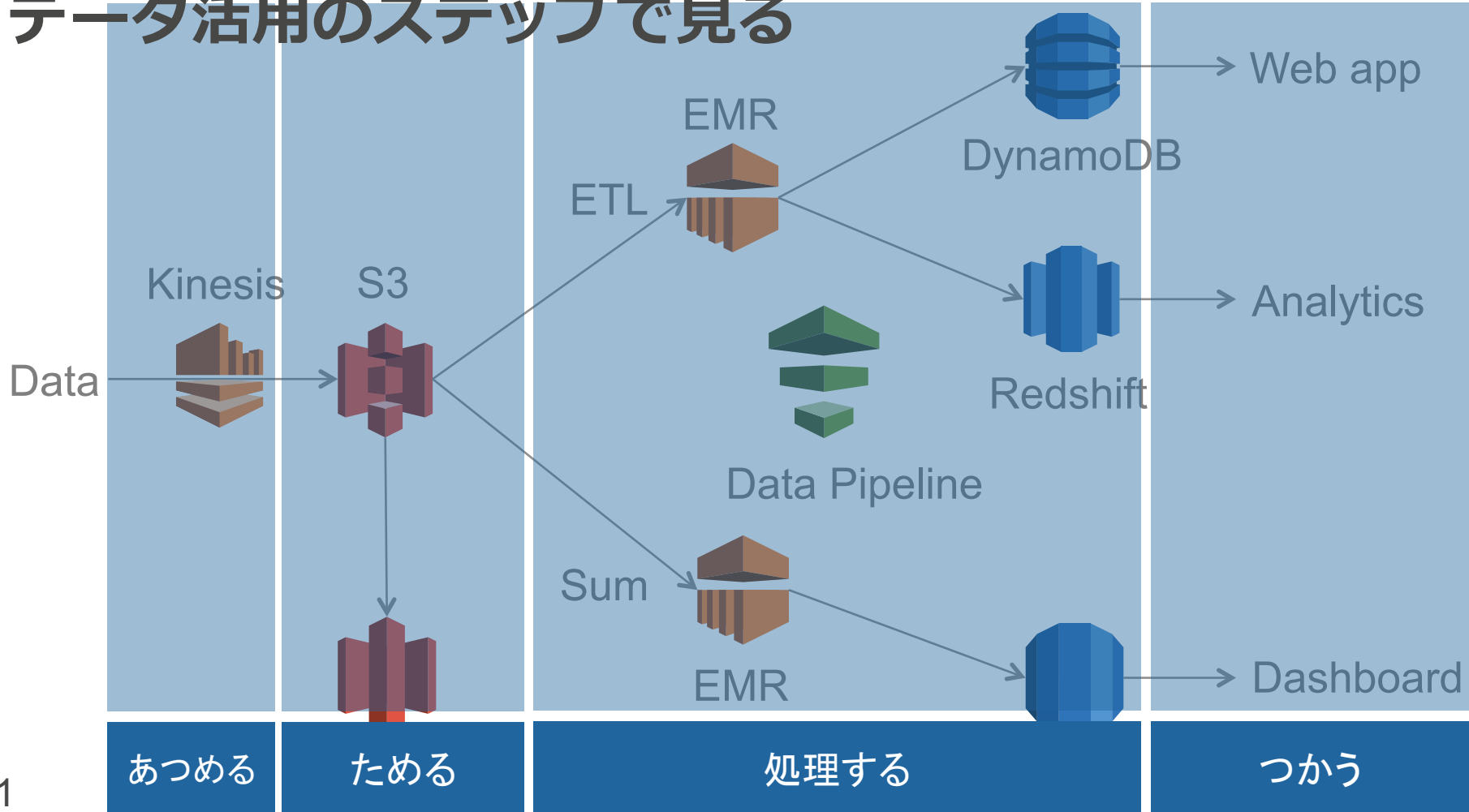
3. 処理する

- 抽出、除外、整形、いわゆる前処理
- 一次集計もここに含まれる

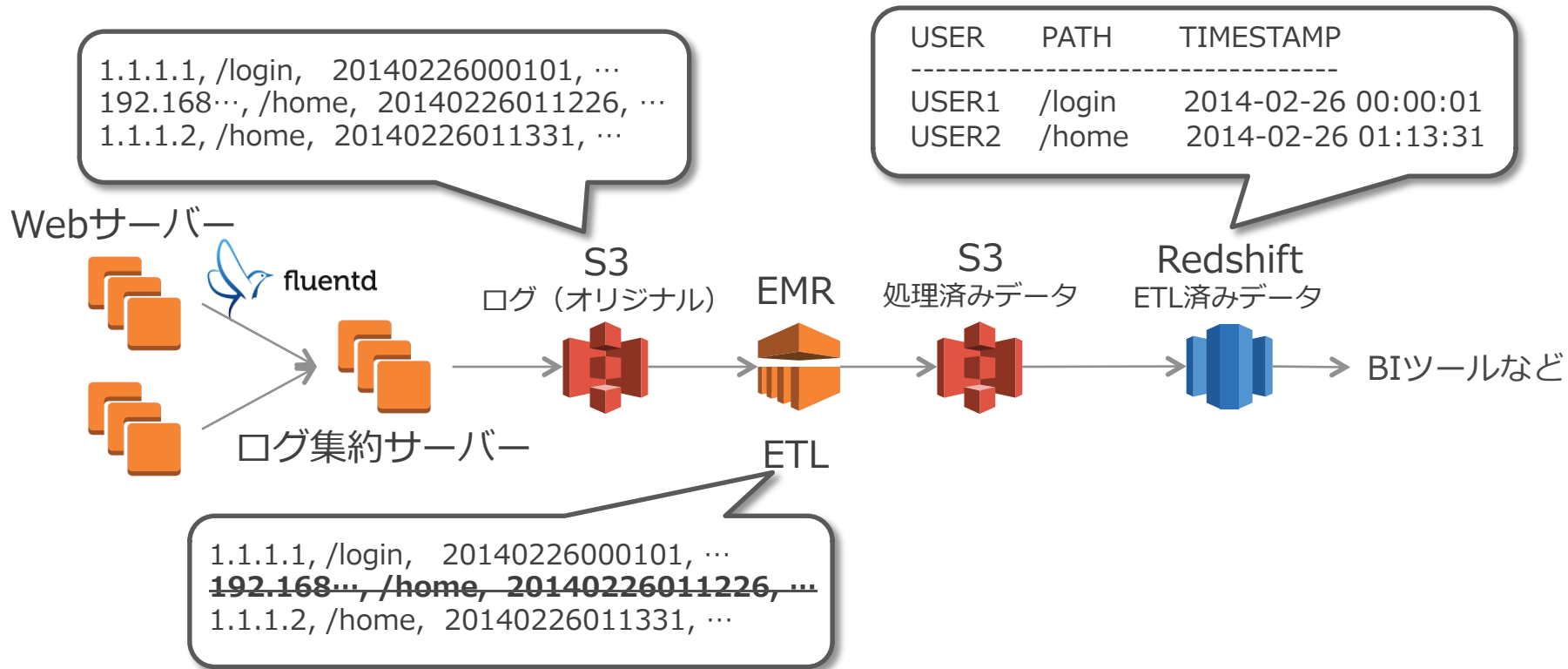
4. つかう

- BIツールで利用
- データをAPIで提供

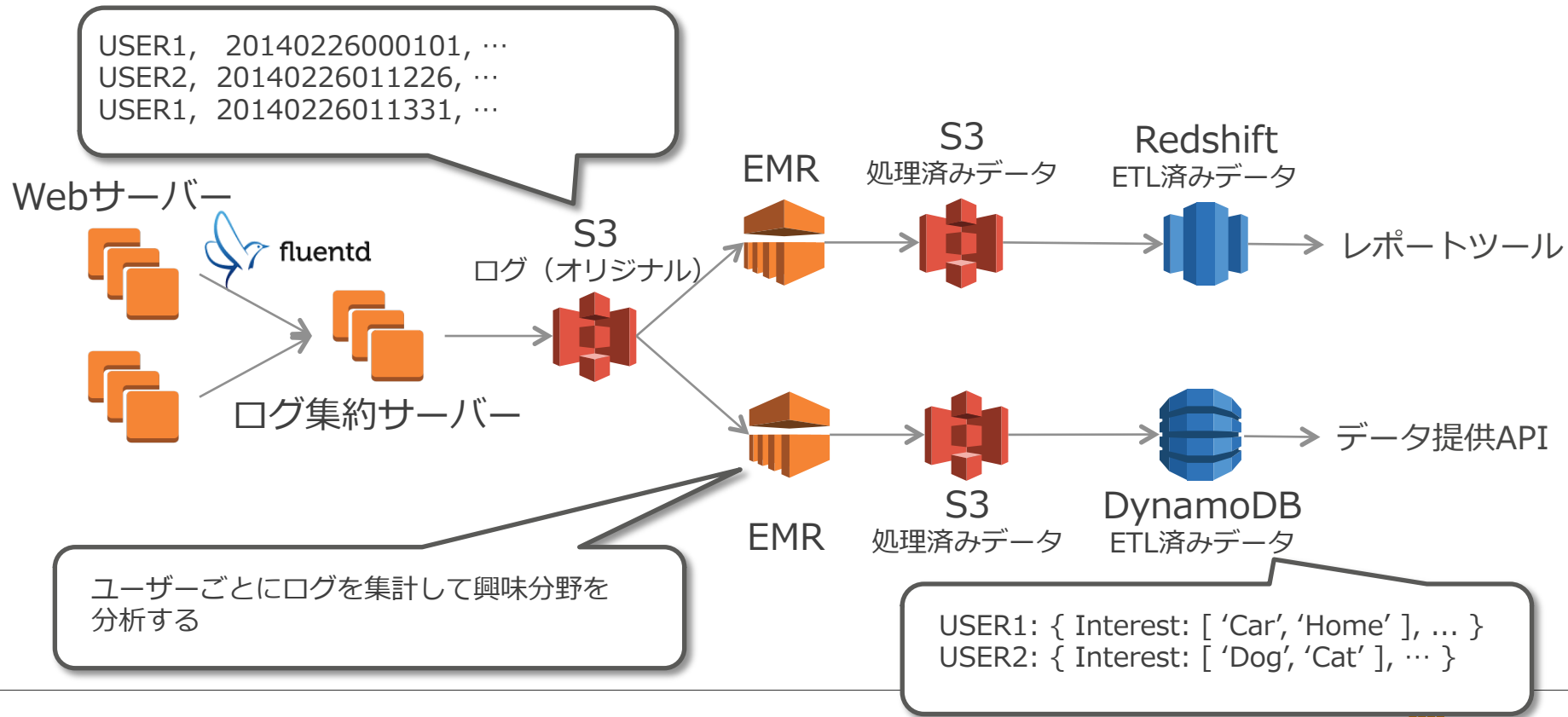
データ活用のステップで見る



例：バッチ処理によるアクセスログ集計



例：収集したデータの活用 -DMP-



データ活用の4つのステップ

1. あつめる

- 多数のアプリケーションサーバーやクライアント、デバイスからのデータ収集

2. ためる

- 安全でコスト効率よく、かつ利用しやすい形でデータを保存

3. 処理する

- 抽出、除外、整形、いわゆる前処理
- 一次集計もここに含まれる

4. つかう

- BIツールで利用
- データをAPIで提供

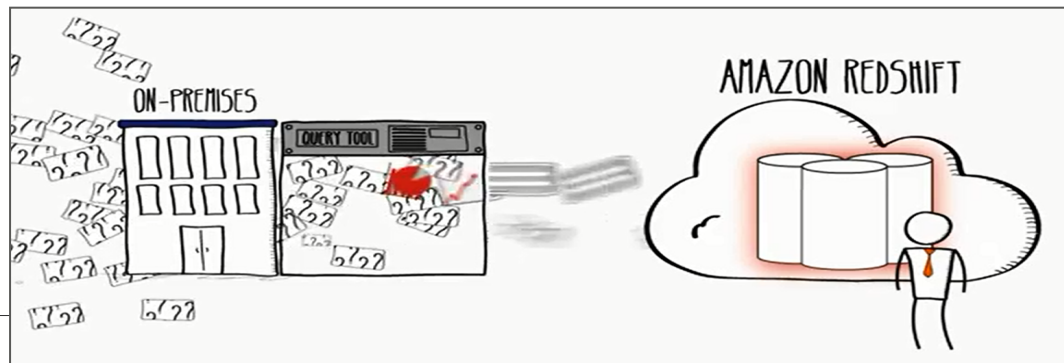
アジェンダ

1. AWSのサービス全体像と
ビッグデータ関連サービス
2. 解剖ビッグデータ: あつめる、ためる、つかう
3. Getting Started with Big Data Services
 - Amazon DynamoDB、Amazon Elastic MapReduce、
Amazon Redshift
4. Practical Deep Dive
 - 現場で見かけるアーキテクチャ

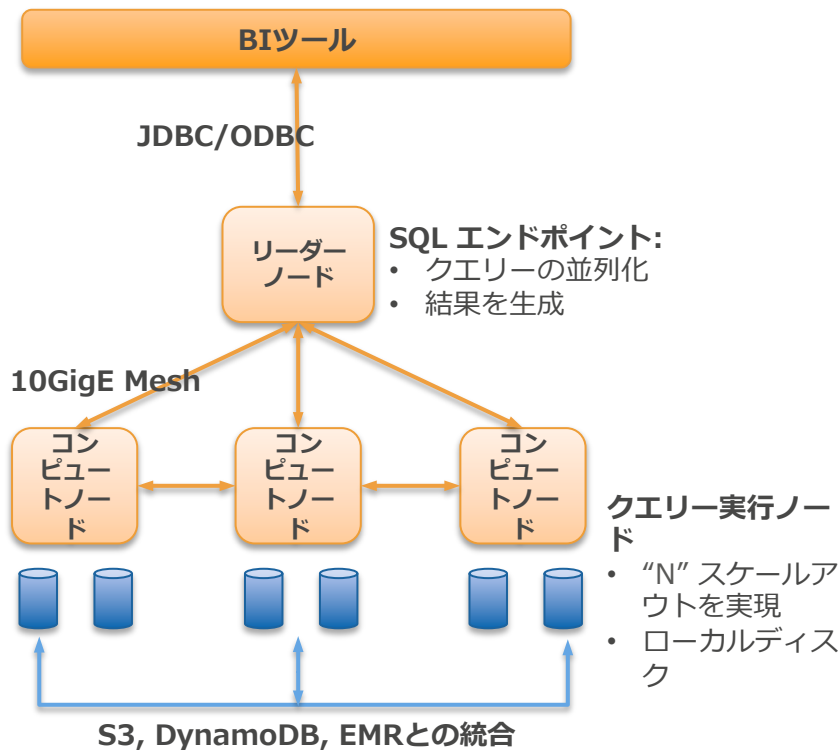
Amazon Redshift

Amazon Redshift概要

- Data Warehouse as a Service – 分析用に整理された大量の統合業務データの管理サービス（フルマネージドサービス）
- 拡張性：160GB～1.6PBまで拡張可能
- 高速：カラムナ型、超並列演算(MPP)
- 低額：インスタンスの従量課金（初期費用、ライセンス費用不要）



Amazon Redshiftのアーキテクチャ



- リーダーノードを経由してクエリーを実行
- インターフェイスはPostgreSQL互換(psqlで使えます)
- 各コンピュートノードで演算が並列実行
- 各コンピュートノードにローカルストレージを保持

アーキテクチャ：列指向

- 行指向 (RDBMS)

orderid	name	price
1	Book	100
2	Pen	50
	...	
n	Eraser	70

- 列指向 (Redshift)

orderid	name	price
1	Book	100
2	Pen	50
	...	
n	Eraser	70

Amazon Redshiftのノードタイプ

dw1.xlarge:

- CPU: 2 virtual cores
- ECU: 4.4
- Memory: 15 GiB
- Storage: 2TB(HDD)
- Network: 0.3GB/s

dw1.8xlarge

- CPU: 16 virtual cores
- ECU: 35
- Memory: 120 GiB
- Storage: 16TB(SSD)
- Network: 2.4GB/s

• dw2.large:

- CPU: 2 virtual cores
- ECU: 7
- Memory: 15 GiB
- Storage: 160GB(SSD)
- Network: 0.2GB/s

• dw2.8xlarge

- CPU: 32 virtual cores
- ECU: 104
- Memory: 244 GiB
- Storage: 2.56TB(SSD)
- Network: 3.7GB/s

Amazon S3からCSVファイルをロードしてみる

- まずはRedshiftにログイン

```
psql -d mydb -h YOUR_REDSHIFT_ENDPOINT -p  
5439 -U awsuser -W
```

- Redshiftのシェルでcopyコマンド

```
COPY customer FROM 's3://data/customer.tbl.'  
CREDENTIALS  
'aws_access_key_id=KEY;aws_secret_access_key=  
SEC' DELIMITER ',' GZIP TIME_FORMAT 'auto';
```

Amazon Redshiftにログインしてデータロードしてクエリを掛けてみる

- Redshiftのシェルでテーブルを定義

```
CREATE TABLE nginx (  
  remote_addr char(15),  
  time timestamp,  
  request varchar(255),  
  status integer,  
  bytes bigint,  
  ua varchar  
);
```

あとはいつものSQL

```
SELECT ua, request, COUNT(*)  
FROM nginx  
GROUP BY ua, request;
```

S3へデータを書き出すのも簡単

```
UNLOAD TO 's3://YOUR_BUCKET/PATH/  
SELECT * FROM nginx;
```


Tableau + Redshift

Amazon Redshift Connection

Step 1: Enter a server name:
ev.cztbcyhph88.us-east-1.redshift.amazonaws.com Port: 8912

Step 2: Enter a database on the server:
test

Step 3: Enter information to log on to the database:
Username: jsmith
Password: ●●●●●●

Step 4: Establish the connection:
Connect

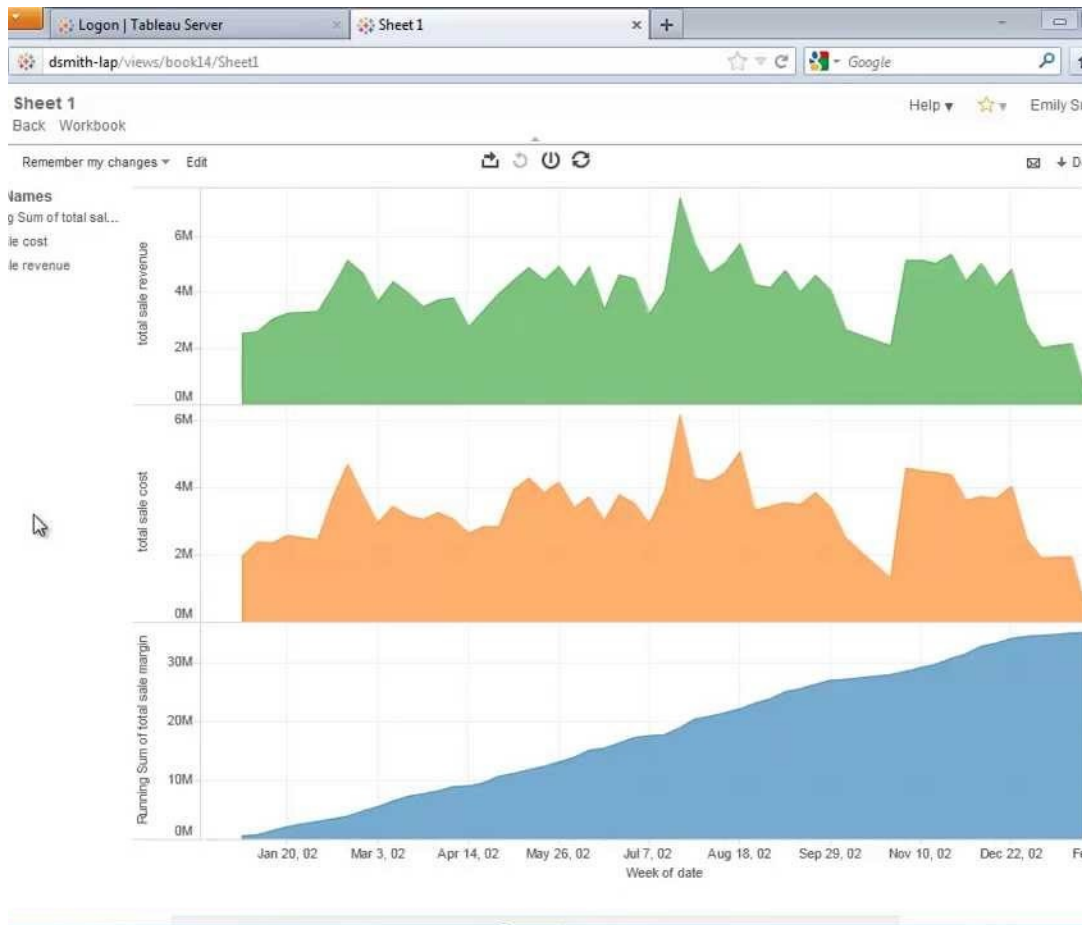
Step 5: Select a schema on the server:
public

Step 6: Select a table or view from the database:
 Single Table Multiple Tables Custom SQL

batters (public.batters)
calcs (public.calcs)
datebins (public.datebins)
datetime (public.datetime)
election (public.election)
fisheriris (public.fisheriris)
... (public. ...)

Step 7: Give the connection a name for use in Tableau:
Election Data

OK Cancel



Amazon Redshiftとは . . .

- 大量のデータを高速にSQLで処理してくれる
- RDBと違いデータ量が増えても性能が劣化しにくい
- ただし、あくまで分析用データベースである
- データはETLや正規化されている必要がある
- 集計や統計など、数値の可視化に有効

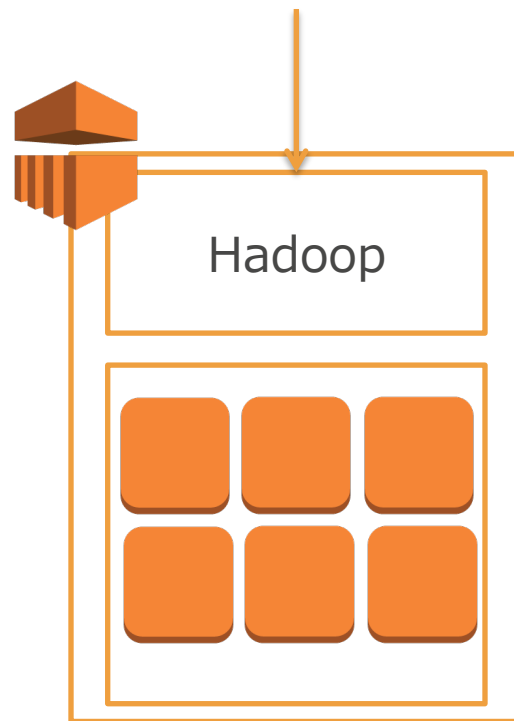
レポートツール等,データ可視化に最適

Amazon Elastic MapReduce

Elastic MapReduce

- AWSが提供するマネージドHadoop
- Hadoop1系、2系、MapRが利用可能
- マネージド？
 - クラスタの構築・監視・復旧
 - CloudWatchによるモニタリング
 - S3のデータを扱える
- 2つの大きな特徴
 - ワークフローマネジメント
 - S3、DynamoDBのデータを扱える

ユーザーは普通のHadoopとして
利用できる



ワークフローマネージメント : スクリプトでHadoopを起動

```
aws emr create-cluster \  
--name bigdata-handson \  
--ami-version 3.2.1 \  
--applications Name=Hive \  
--instance-groups  
InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m1.large  
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m1.large \  
--log-uri s3:/PATH/T0/LOG/ \  
--ec2-attributes SubnetId=subnet-a06474e6,KeyName=YOUR_KEY
```

ワークフローマネージメント：スクリプトでHadoopを起動

```
aws emr create-cluster \  
--name bigdata-handson \  
--ami-version 3.2.1 \  
--applications Name=Hive \  
--instance-groups  
InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m1.large  
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m1.large \  
--log-uri s3:/PATH/T0/LOG/ \  
--ec2-attributes SubnetId=subnet-a06474e6,KeyName=YOUR_KEY
```

名前をつける

ワークフローマネージメント : スクリプトでHadoopを起動

```
aws emr create-cluster \  
--name bigdata-handson \  
--ami-version 3.2.1 \  
--applications Name=Hive \  
--instance-groups  
InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m1.large  
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m1.large \  
--log-uri s3:/PATH/T0/LOG/ \  
--ec2-attributes SubnetId=subnet-a06474e6,KeyName=YOUR_KEY
```

AMI (Hadoop) のバージョンを指定する

ワークフローマネージメント : スクリプトでHadoopを起動

```
aws emr create-cluster \  
--name bigdata-handson \  
--ami-version 3.2.1 \  
--applications Name=Hive \  
--instance-groups  
InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m1.large  
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m1.large \  
--log-uri s3:/PATH/T0/LOG/ \  
--ec2-attributes SubnetId=subnet-a06474e6,KeyName=YOUR_KEY
```

インストールするアプリケーションを指定

ワークフローマネージメント : スクリプトでHadoopを起動

```
aws emr create-cluster \  
--name bigdata-handson \  
--ami-version 3.2.1 \  
--applications Name=Hive \  
--instance-groups  
InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m1.large  
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m1.large \  
--log-uri s3:/PATH/T0/LOG/ \  
--ec2-attributes SubnetId=subnet-a06474e6,KeyName=YOUR_KEY
```

インスタンスタイプを指定

ワークフローマネージメント : スクリプトでHadoopを起動

```
aws emr create-cluster \  
--name bigdata-handson \  
--ami-version 3.2.1 \  
--applications Name=Hive \  
--instance-groups  
InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m1.large  
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m1.large \  
--log-uri s3:/PATH/T0/LOG/ \  
--ec2-attributes SubnetId=subnet-a06474e6,KeyName=YOUR_KEY
```

ログの吐き出し先を指定

ワークフローマネージメント：スクリプトでHadoopを起動

```
aws emr create-cluster \  
--name bigdata-handson \  
--ami-version 3.2.1 \  
--applications Name=Hive \  
--instance-groups  
InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m1.large  
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m1.large \  
--log-uri s3:/PATH/T0/LOG/ \  
--ec2-attributes SubnetId=subnet-a06474e6,KeyName=YOUR_KEY
```

VPCなどの情報を指定

さらに

仕事を予め定義して起動し、終わったら自動削除

```
aws emr create-cluster \  
--name bigdata-handson \  
--ami-version 3.2.1 \  
--applications Name=Hive \  
--instance-groups  
InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m1.large  
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m1.large \  
--log-uri s3://PATH/T0/LOG/ \  
--ec2-attributes SubnetId=subnet-a06474e6,KeyName=YOUR_KEY  
--steps Type=HIVE,Name='Hive program', Args=[-f,s3://PATH/T0/  
QUERY.q] \  
--auto-terminate
```

cronやData Pipelineでワークフローを制御すればHadoopのジョブを自動化できる！

S3との連携

S3のデータを扱える

- HDFSとシームレスにS3上のデータを扱える
- INPUTやOUTPUTにs3://~を指定する
- S3からデータを取り出して結果を更にS3に吐き出す

```
hadoop jar YOUR_JAR.jar \  
--src s3://YOUR_BUCKET/logs/ \  
--dest s3://YOUR_BUCKET/output/
```

- S3からデータを取り出して結果はローカルのHDFSに吐き出す

```
hadoop jar YOUR_JAR.jar \  
--src s3://YOUR_BUCKET/logs/ \  
--desct hdfs:///output/
```


もちろんHiveでも

```
CREATE EXTERNAL TABLE s3_as_external_table(  
    user_id INT,  
    movie_id INT,  
    rating INT,  
    unixtime STRING )  
ROW FORMAT DELIMITED FIELDS  
TERMINATED BY '\t'  
STORED AS TEXTFILE  
LOCATION 's3://mybucket/tables/';
```

Hiveを使ったETL

```
INSERT INTO TABLE table2
SELECT
    column1,
    column2,
    column5,
FROM table1;
```

←例えばtable1から
column3,4を除外したい

EMR ちょっとDeep Dive ～RedshiftとEMRどちらを使う？～

Amazon Redshift

- 基本的な使い勝手はRDB
- SQLを使って解析
- BIツールのバックエンドとして
- ある程度正規化されたデータが前提条件
- 複雑なジョインも得意
- クラスタは基本的には起動しっぱなし

Amazon Elastic MapReduce

- Hadoop
- MapReduce, Hive, Pig, Hadoop StreamingなどのHadoopのエコシステムが利用できる
- hiveでSQLっぽく使うこともできるがRedshiftのほうが速いし簡単
- 正規化しづらいデータを扱うのが得意
- 立ちあげっぱなしではなく、ジョブごとにクラスタを起動して終了した破棄する使い方もできる

EMRかRedshiftか

- SQLを使った分析/解析ならRedshiftのほうが圧倒的に速い
- それ以外ならEMR

ざっくり言うと

Amazon Elastic MapReduceとは . . .

- Hadoopを便利に利用できるようにしたサービスで大量のテキストデータを整形するのが非常に得意
- ワークフローをうまく活用すれば処理の自動が用意
- S3のデータを自在に扱える

生ログのETLや一次集計に強みを持つ

Amazon DynamoDB

DynamoDBとは

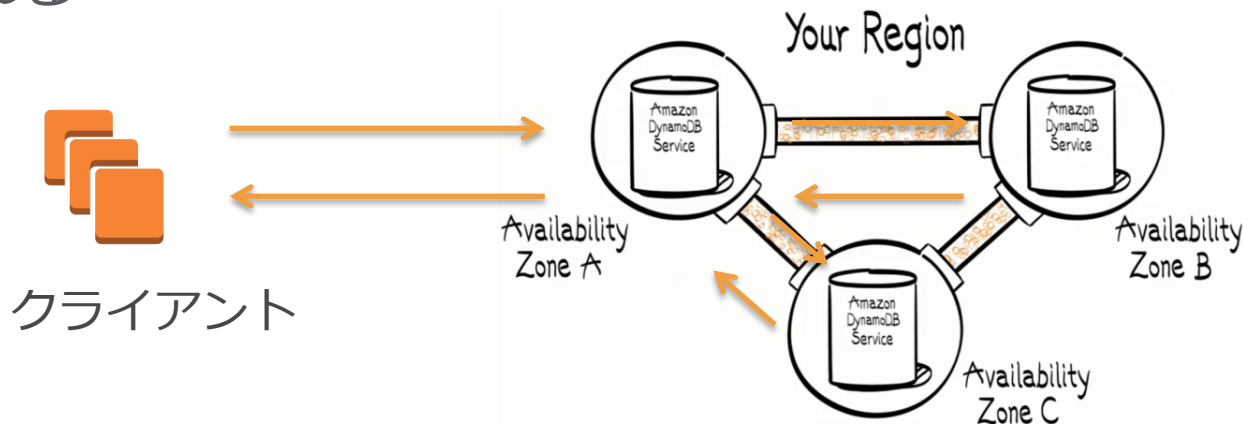
- NoSQL as a Service
- データ量が増えても性能が劣化しない
- 大規模なデータを高速に扱いたいときに真の価値を発揮

DynamoDBの特長

- 管理不要で信頼性が高い
- プロビジョンドスレーブット
- ストレージの容量制限がない

特長 1 : 管理不要で信頼性が高い

- SPOFの存在しない構成
- データは3箇所の設備に保存されるので信頼性が高い
- ストレージは必要に応じて自動的にパーティショニングされる



特長 2 : プロビジョンドスループット

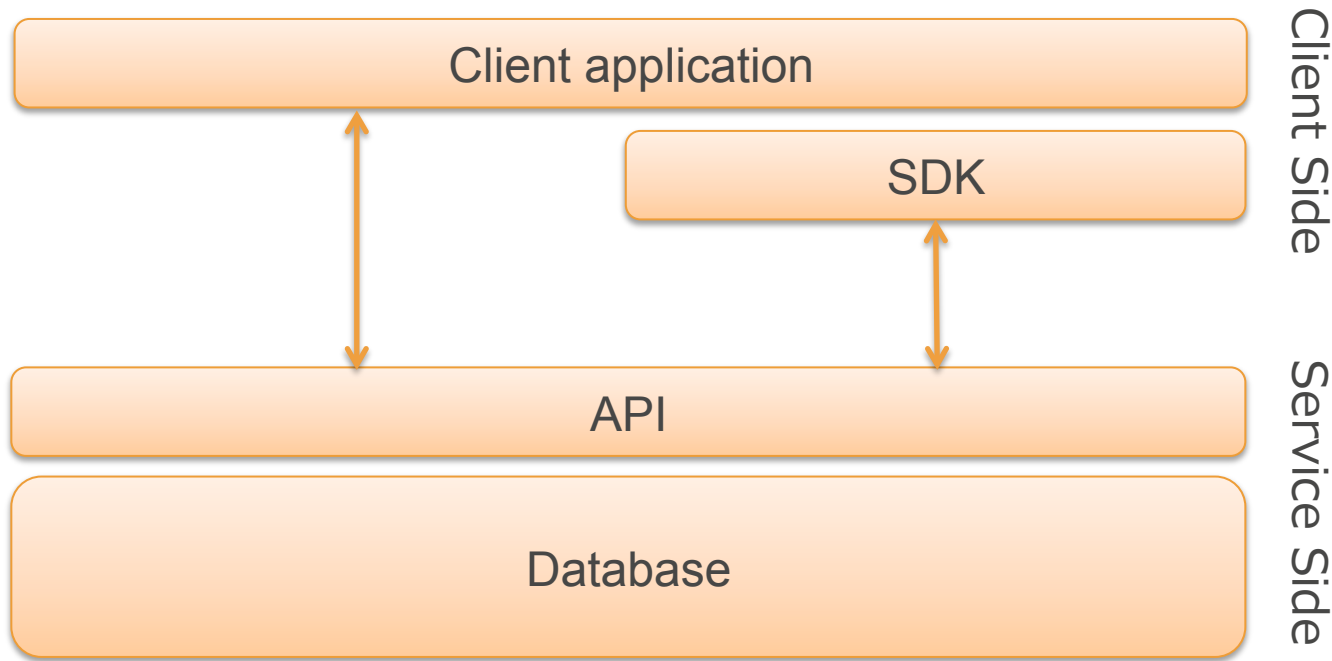
- テーブルごとにReadとWriteそれぞれに対し、必要な分だけのスループットキャパシティを割り当てる（=プロビジョンする）ことができる
- 例えば下記のようにプロビジョンする
 - Read : 1,000
 - Write : 100
- 書き込みワークロードが上がってきたら
 - Read : 500
 - Write : 1,000
- この値はDB運用中にオンラインで変更可能

特長 3 : ストレージの容量制限がない

- 使った分だけの従量課金制のストレージ
- データ容量の増加に応じたディスクやノードの増設作業は一切不要

DynamoDBの構成要素

- オペレーションはHTTPベースのAPIで提供されている
- ユーザーはコードを書くだけで利用できる



DynamoDBを使い始めるには

1. テーブルのKeyやIndexを決める
 - カラムスキーマは決めなくていい
2. Read/Writeそれぞれのスループットを決める

That's it, write your code!

DynamoDBのテーブルのプライマリキーの持ち方は2種類

- Hash key
- Hash key & Range key

プライマリキーがハッシュキーのサンプル 1 : ユーザー情報データベース

ユーザー情報データベース

ユーザーIDをプライマリキーとしたKVS的なテーブル

- UserIdで一意のItemを特定し情報の参照や更新、削除を行う



Users Table

<u>UserId (Hash)</u>	Name	Nicknames	Mail	Address	Interests
aed9d	Bob	[Rob, Bobby]	foo@example.com	some address	[Car, Motor Cycle]
edfg12	Alice	[Allie]			
a8eesd	Carol	[Caroline]			
f42aed	Dan	[Daniel, Danny]			

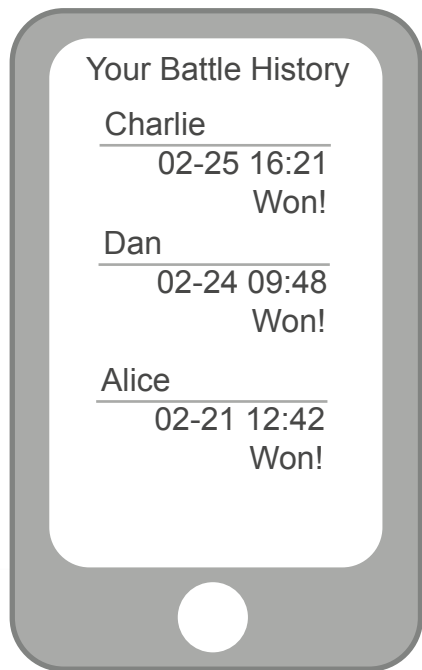
※DynamoDBにはauto_increment等でユニークIDを払い出す機能はないので注意。
UUIDなどを使ってください。

プライマリキーがハッシュ&レンジのテーブルサンプル： ゲームの行動履歴管理データベース

ゲームの行動履歴管理データベース

自分のバトル履歴を確認するケースを想定

- Userに自分(Alice)を指定し、更にTimestampが7日以内のデータをクエリしたりできる



Battle History

User (Hash)	Timestamp (Range)	Opponent	Result
Alice	2014-02-21 12:21:20	Bob	Lost
Alice	2014-02-21 12:42:01	Bob	Won
Alice	2014-02-24 09:48:00	Dan	Won
Alice	2014-02-25 16:21:11	Charlie	Won

テーブル設計のための基礎知識+1

- **Local Secondary Index**

- Range key以外に絞り込み検索のためのキーを持つことができる
- Hash keyが同一のアイテム群の中からの検索のために利用
- インデックスもテーブルにプロビジョンしたスループットを利用する

- **Global Secondary Index**

- Hash Keyをまたいで検索を行うためのインデックス
- インデックスにテーブルとは独立したスループットをプロビジョンして利用する

Amazon DynamoDBとは . . .

- 分散型のNoSQL
- 大量のデータを投入しても性能が劣化しない
- Redshiftとは違い、こちらはOLTP用データベース
- SQLのようにJOINができるわけではない

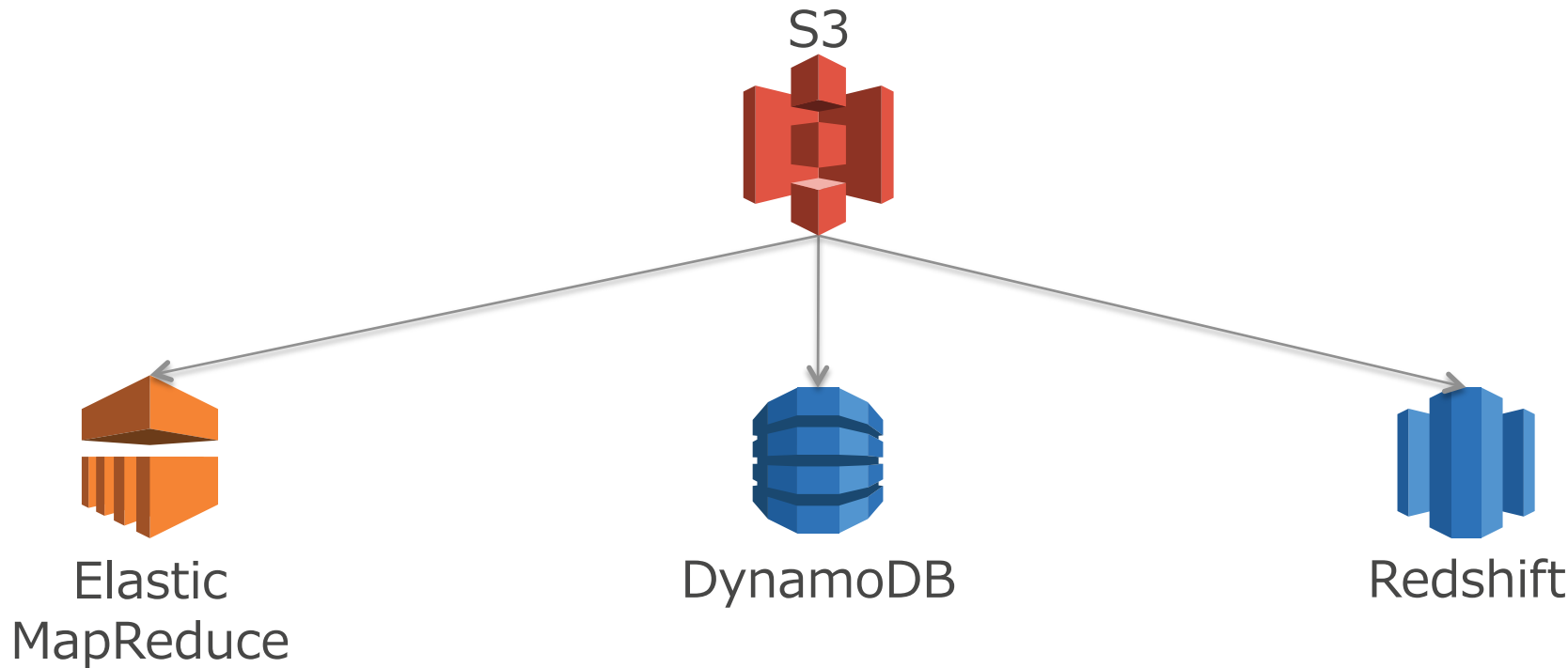
- 大量のデータを格納しておいて、必要な少数のデータを高速にやりとりするのに強みを持つ

アジェンダ

1. AWSのサービス全体像とビッグデータ関連サービス
2. 解剖ビッグデータ: あつめる、ためる、つかう
3. Getting Started with Big Data Services
 - Amazon Redshift
 - Amazon Elastic MapReduce
 - Amazon DynamoDB
4. Practical Deep Dive

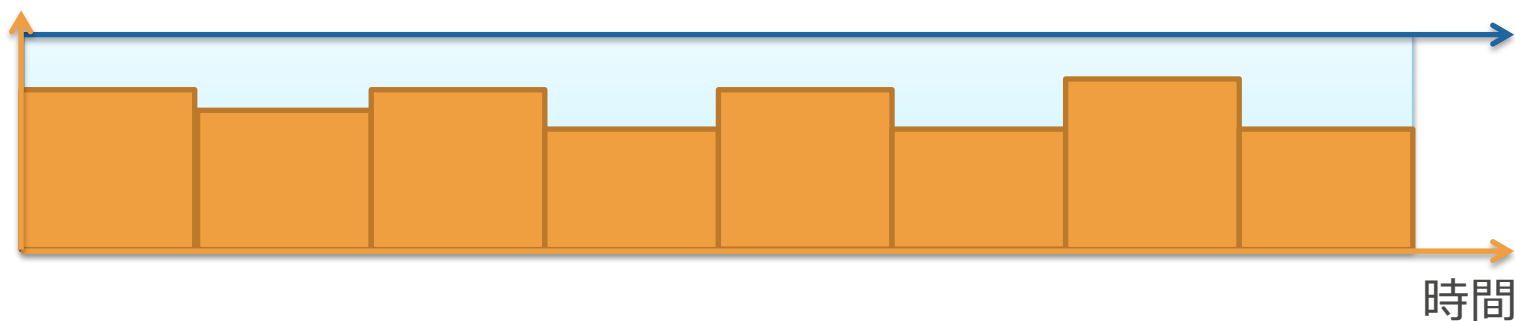
AWS上でデータ処理を行う際には
S3の利用がキー

データがS3にあれば あとは必要に応じて処理クラスタを起動して利用できる



S3とEMR

負荷



Hadoop単体ではデータ共有はできない



ひとつのデータに対する処理は
ひとつのクラスターに強く結合してしまう

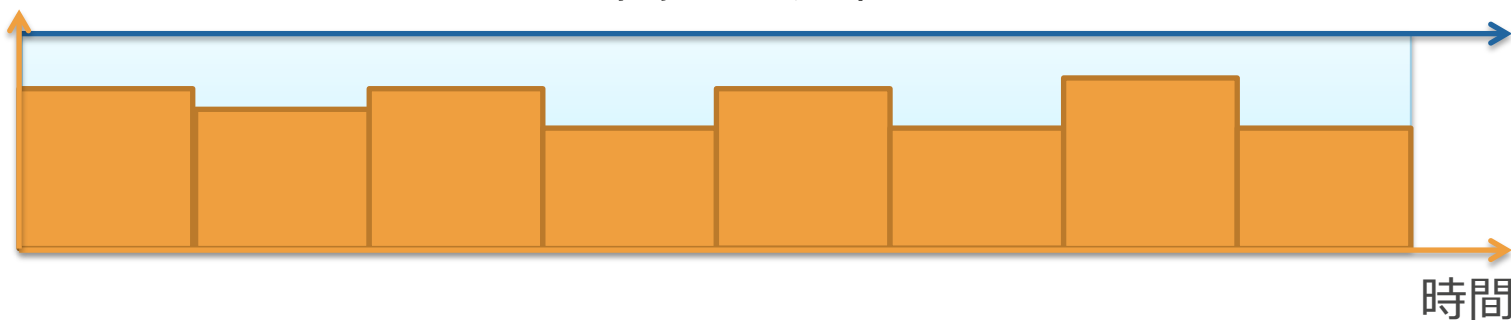


キャパシティプランニングが難しい

S3とEMR

負荷

キャパシティ



時間



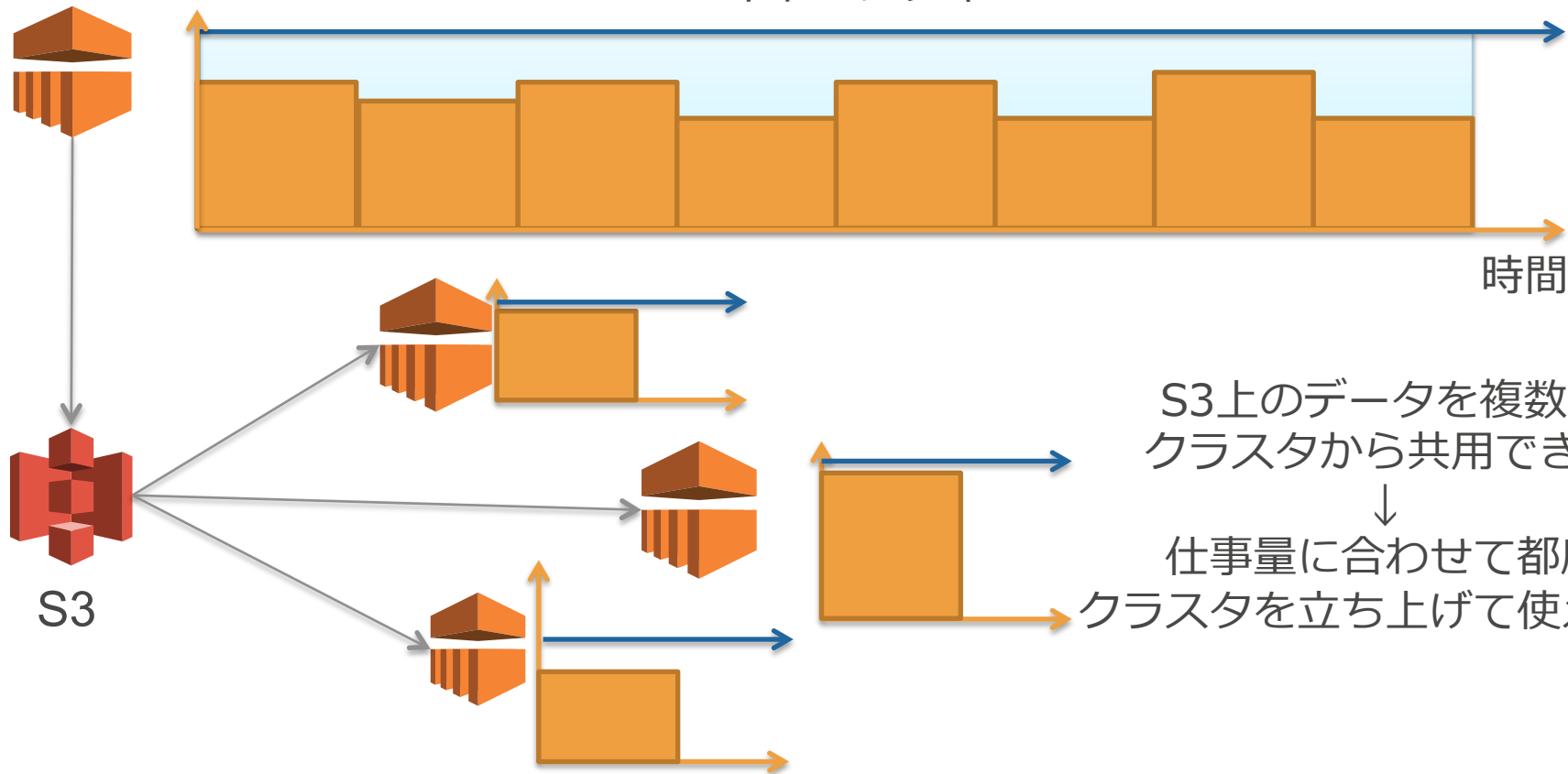
S3

データをHDFSではなく
S3に格納しておけば・・・

S3とEMR

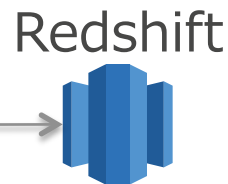
負荷

キャパシティ



S3とRedshift

```
COPY table_name FROM 's3://hoge'  
CREDENTIALS 'access_key_id:hoge...'  
DELIMITER ','
```



Redshiftへのデータロードは
S3経由が効率的

S3とRedshift

```
UNLOAD ('SELECT * FROM...')  
TO 's3://fuga/...'  
CREDENTIALS 'access_key_id:hoge...';
```

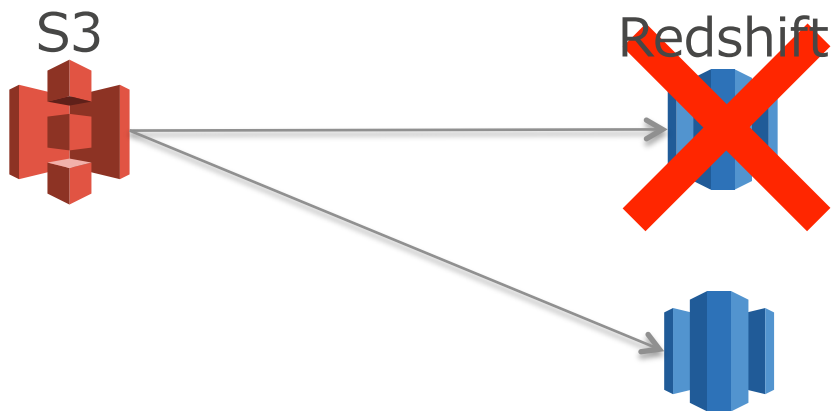
RedshiftからS3への
エクスポートや
スナップショットも容易



Redshift



S3とRedshift



クラスタに問題が起きたら、
S3からクラスタを再構築

S3とDynamoDB



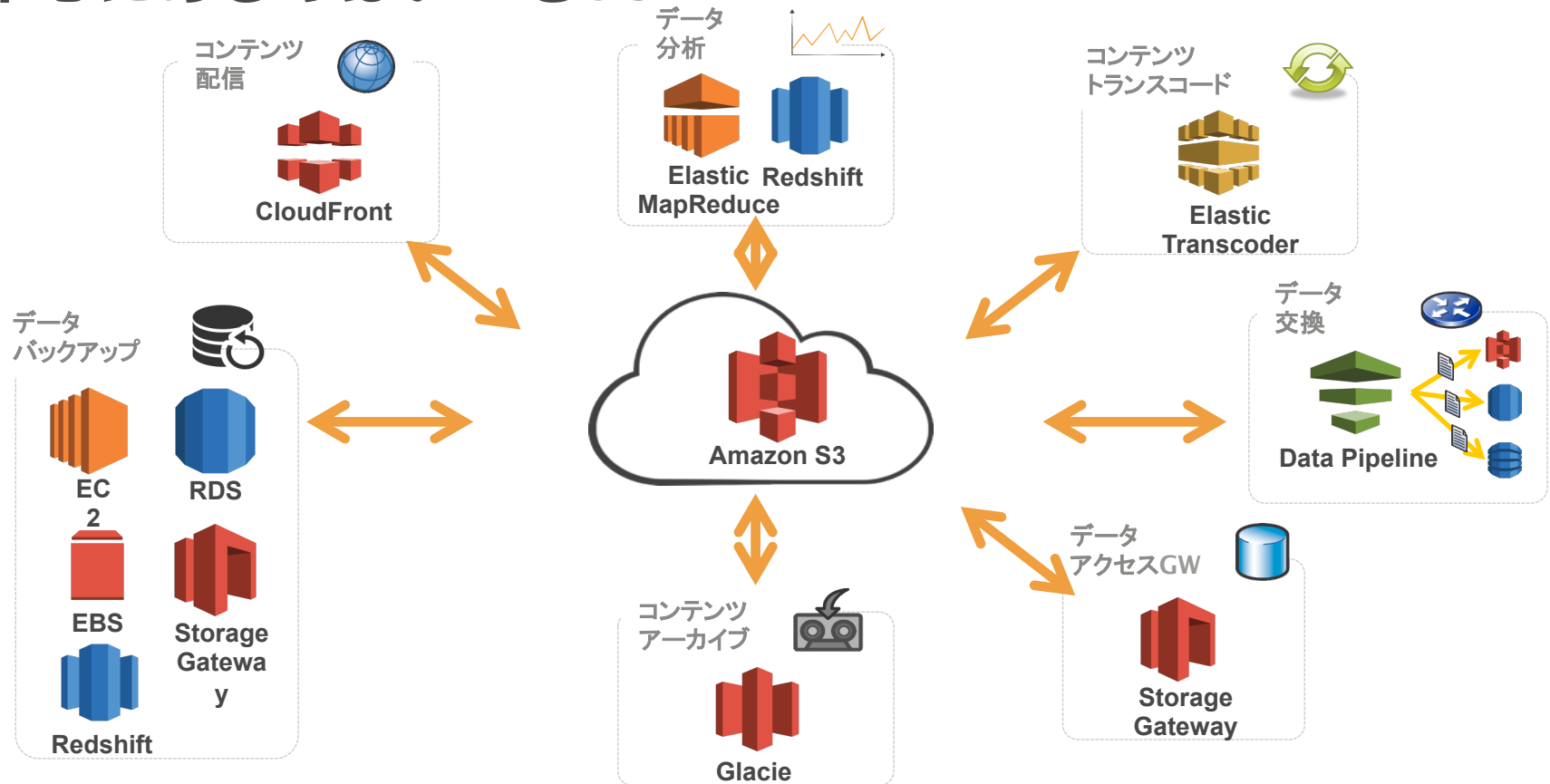
DynamoDB

S3とDynamoDB

古いデータはS3へオフロード
もしくはバックアップ



大事なことなのでもう一度。 中心にあるのはいつもS3



まとめ

各サービスの役割を理解し、うまく組み合わせる

- あつめる、ためる、処理する、つかう、それぞれのフェーズに有効なサービスは異なる。
- 自分がどのフェーズのソリューションを必要としているのかを見極めてサービスを選ぶ。

各サービスの役割を理解し、うまく組み合わせる

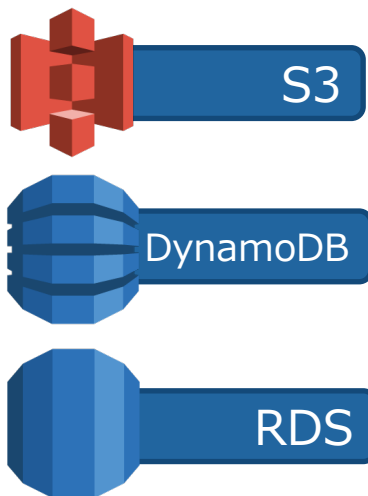
どこから？

あつめる



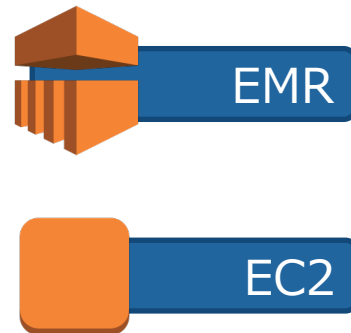
どのくらい？

ためる



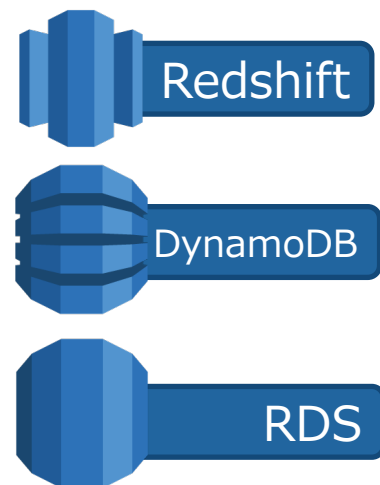
どういう？

処理する



どうやって？

つかう



ありがとうございました