

スケーラブルなデータセンター 情報処理アーキテクチャ

IIJ 技術研究所

阿部 博

2014/11/20

データセンターの情報収集

少し前、現状
(ホスティング、SI)

- ・ SLA（品質保証）、レポーティング、可視化
- ・ データを見ることが目的

クラウド時代

- ・ 課金情報、上位サービス（IaaS, PaaS）との連携
- ・ 見るだけからサービスの制御/改良に利用

次世代

- ・ ファシリティとクラウド資源管理との連携
- ・ 相関分析、フィードバック制御、予測制御

データセンターの情報収集

少し前、現状
(ホスティング、SI)

- ・ SLA (品質保証)、レポーティング、可視化
- ・ データを見ることが目的

ギャップ1

クラウド時代

- ・ 課金情報、上位サービス (IaaS, PaaS) との連携
- ・ 見るだけからサービスの制御/改良に利用

次世代

- ・ ファシリティとクラウド資源管理との連携
- ・ 相関分析、フィードバック制御、予測制御

ギャップ 1：可視化から制御へ

問題点

- ターゲットデバイスの増加
サーバ、スイッチ、ルータ、センサー、etc
- クラウド向けデータセンター
サーバ数1,000～X0,000台

収集ターゲットが多い＝扱うデータ量が多い

コンテナ型DCで収集するデータ

大まかな見積もり（エントリー / 5 sec）

	1デバイス	1ラック	1コンテナ(3ラック)
PDU	100	200(2 PDU)	600
Switch	500	1,500(3 sws)	4,500
Server	100	4,000(40 srvs)	12,000
others(UPS, iSCSI, sensors...)	1,000	1,000	3,000
Total	1,700	5,700	21,000

- コンテナ型DCの1秒毎のデータエントリー = 約4000エントリー
 - 14M エントリー / 1 hour, 345M エントリー / 1 day, 126G エントリー / 1 year
- データサイズ
 - 1 エントリー = 32 byte(time, value), 126G エントリー x 32 byte = 4TByte / year

既存データ処理システムの問題点

データ処理

- ・ 基本スケールアップ、疑似スケールアウト (master+複数slave)
- ・ 構成を意識したクエリーの投げ込み

データストア

- ・ リレーショナルデータベース (RDB)
- ・ メモリキャッシュ (memcached, Redis)



問題点

- ・ 台数を増やすと障害ポイントが増える/運用が大変
- ・ RDBの利点の更新/削除をあまり使わない (特性のミスマッチ)

問題を解決するために必要なもの

スケーラブルプロセッシングプラットフォーム

スケーラブルストレージ

ハイスピードプロセッシング

問題を解決するために必要なもの

スケーラブルプロセッシングプラットフォーム

スケーラブルアーキテクチャ

スケーラブルストレージ

- Hadoop: 台数を増やせば処理速度が上がる
- HDFS : 台数を増やせばストレージ容量が増える

問題を解決するために必要なもの

スケーラブルプロセッシングプラットフォーム

ハイパフォーマンスアーキテクチャ

ハイスピードプロセッシング

- 時系列データベース: 時間をキーとした処理に関して高性能
- Impala, BigQuery : テラバイト級データを秒のオーダーで処理

扱うデータの特徴

フォーマット

- ・ time, key, value形式
- ・ JOINは基本的に行わない

クエリの種類

- ・ insert, selectがメイン
- ・ update, deleteは基本的に行わない

計算

- ・ avg, min, maxなども求めたい
- ・ 可能であればinsertと同時に計算したい

情報の処理方法

バッチ処理

- 日時/月次/年次レポート処理
- 夜間や特定の時間処理を流し続け結果を返す

ストリーミング 処理

- 分/時など現在から近傍の時間データ処理
- ストレージに貯めずパイプライン的に計算する

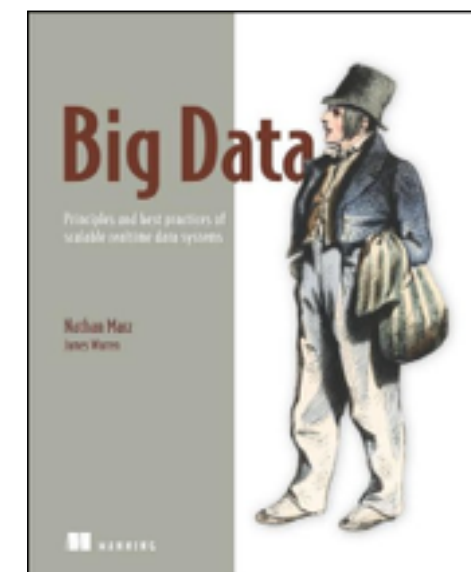
即値処理

- リアルタイム処理向けのアラートやrawデータ
- データを右から左に受け流す

Lambda Architecture

バッチ処理 + ストリーミング処理

- バッチレイヤ: バッチ処理 + データストア
- スピードレイヤ: ストリーミング処理



Nathan Marz and James Warren (2014),
Big Data : Principles and best practices of scalable realtime data systems,
Manning Publications Co.

情報の処理方法

バッチ処理

- 日時/月次/年次レポート処理など
- 夜間や特定時間の処理を流し続ける

Lambda Architecture

ストリーミング 処理

- 分/時など現在から近傍の時間データ処理
- ストレージに貯めずパイプライン的に計算する

即値処理

- アラートやrawデータ
- データを右から左に受け流す

IIJ Labが設計するアーキテクチャ

Lambda Architecture + 'a'

バッチ処理

- 日時/月次/年次レポート処理など
- 夜間や特定時間の処理を流し続ける

ストリーミング
処理

- 多
- 多

即値処理

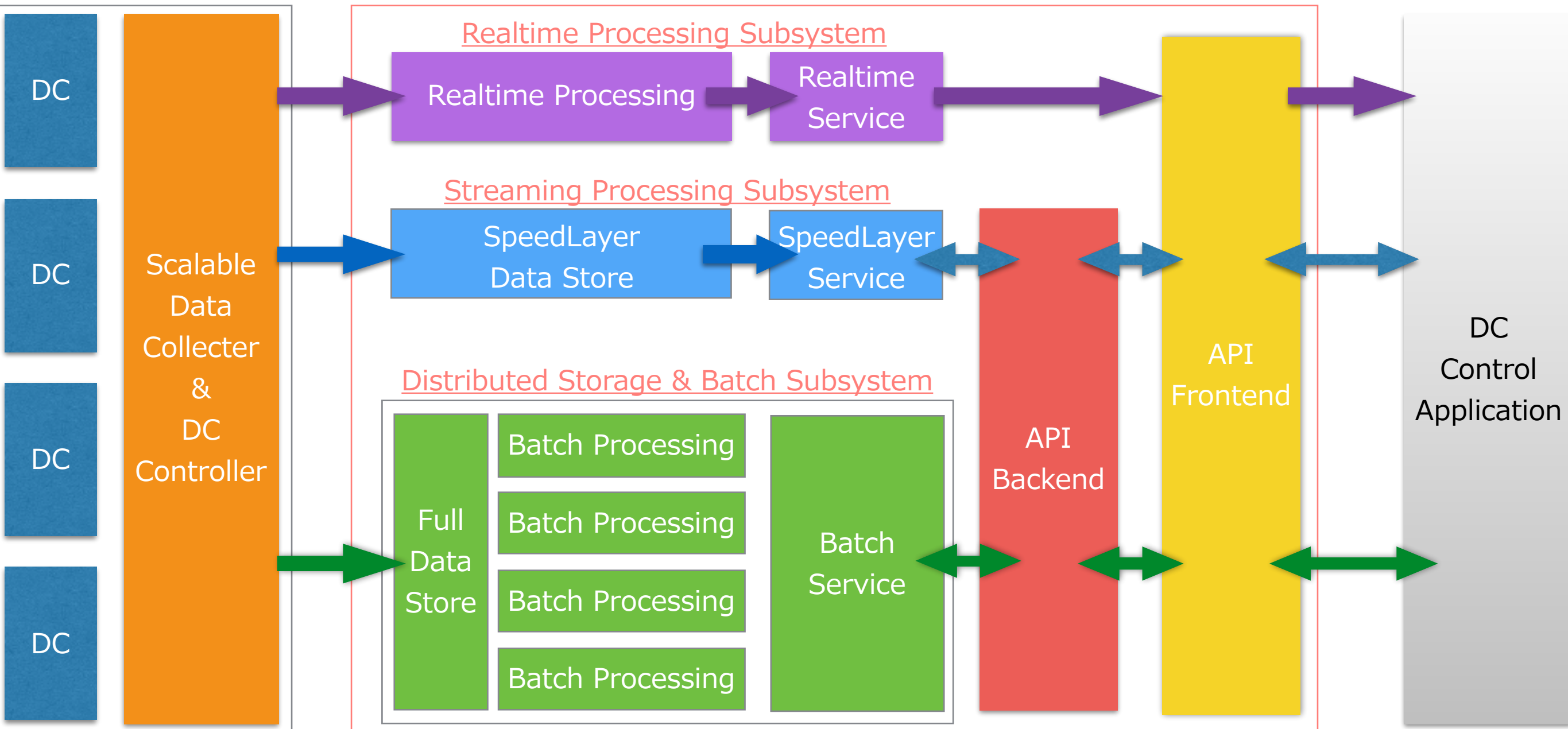
- アラートやrawデータ
- データを右から左に受け流す

Lambda Architecture + 'a'

る

IIJ Labが設計するアーキテクチャ

DataCenter Information Analysis System



IIJ Labが設計するアーキテクチャ

特徴

- Lambdaアーキテクチャでは扱えない即値を扱う
- データ処理を処理時間でレイヤリング
 - バッチ、ストリーミング、リアルタイム
- システム全体がスケールアウト可能な作り

IIJ Labが設計するアーキテクチャ

システム構成要素

- バッチレイヤ: Impala, BigQuery
- スピードレイヤ: InfluxDB, Redis
- リアルタイムレイヤ: MQTT subscribe, Fluentd
- APIリクエストバックエンド処理: RabbitMQ

ギャップ 1：まとめ

クラウド時代の「大量なデータ」に対して以下を実現

- レイヤリングしたプロセッシング/ストレージシステム
- スケールアウト可能なシステムアーキテクチャ
- フィードバック制御のヒント情報を得られる仕組み

データセンターの情報収集

少し前、現状
(ホスティング、SI)

- ・ SLA (品質保証)、レポーティング、可視化
- ・ データを見ることが目的

ギャップ1

クラウド時代

- ・ 課金情報、上位サービス (IaaS, PaaS) との連携
- ・ 見るだけからサービスの制御/改良に利用

ギャップ2

次世代

- ・ ファシリティとクラウド資源管理との連携
- ・ 相関分析、フィードバック制御、予測制御

ギャップ2：連携/予測制御

- ファシリティとクラウドサービス基盤の連携
 - フィードバック制御だけでいいのか？
- プロアクティブ、予測制御などの実現
 - ディープラーニング？
 - 機械学習などの要素が必要？

ギャップ2：連携/予測制御

- ファシリティとクラウドサービス基盤の連携
 - フィードバック制御だけでいいのか？
- プロアクティブ、予測制御などの実現
 - ディープラーニング？
 - 機械学習などの要素が必要？

- 今後の課題
- 議論ポイント

まとめ

- データセンターでの情報収集の過去/現在/未来
- 既存システムの問題点と
スケーラブルアーキテクチャによる解決方法
- 扱うデータの特徴と情報の処理方法
- LambdaアーキテクチャとIIJアーキテクチャ
- 予測制御などの今後の課題