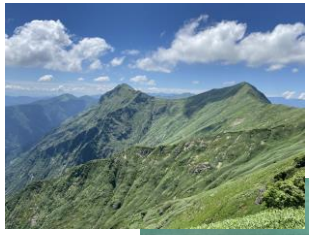
CTC Engineer ‘s Insight #5
現場エンジニアの構成判断は、なぜつまずく？
― クラウド・アーキテクチャ改善と設計思考のリアル

Exadataのクラウドシフトにおけるアプリ＆環境構成見直しの事例紹介

伊藤忠テクノソリューションズ株式会社
金融開発第1部
重田　栄二

無限の未来と、幾千のテクノロジーをつなぐ。
CTC Financial Services Group

自己紹介



氏名： 重田　栄二(しげた えいじ)

1971年3月22日 生まれ(54歳)A型

出身地： 東京都葛飾区(現在は千葉県船橋市)

趣　味： 登山、麻雀

職　歴：

1993年 某IT会社へ新卒入社

(バブル崩壊後で辞令は自宅待機だけど毎日出勤で給料6割..)

1996年 転職しCTCに中途入社

銀行/生保　アプリ開発・保守の担当⇒リーダ⇒PM

2006年 銀行/カード　インフラリーダ/PM

2019年 銀行　アーキテクチャ担当(この頃初めてAWSをさわる)

以降、クラウドシフト案件が続きアーキテクチャ担当として従事

本日お伝えしたいこと

今まで設計当初は問題なく稼働する予定の環境で実際にアプリを動かしてみたら、全然性能が出なかったりしたことは無いでしょうか？
（そしてそれをアプリだけでどうにかしてと言われたことは. . ）

そんな苦い経験を繰り返さない為に、本日はクラウドシフトに限らずシステム更改における環境構成で考慮すべきポイントを事例を交えてご紹介いたします。

事例紹介の流れ

1. 更改システムの概要とPJの背景
2. 性能面での問題
3. 対策の検討
4. RDS移行に関するプロセス
5. まとめ

1.更改システムの概要とPJの背景(事例)

■ システムの概要
- オンプレミス環境にサーバ/ストレージ、DBで構成されたOLAP(バッチ処理)システム
- 日次処理と月次処理は別々のサーバ群(VM)、ストレージで稼働
- ワークフローは収集・変換・計算の結果をロードするETL
- アプリは主にJava・シェルとファイル高速加工ソフト、計算ミドルウェアで構成
- 処理データ規模は1日約2億件

■ PJの背景
- 基盤(サーバ/ストレージ)の保守期限到来　⇒ 日次/月次統合とクラウドシフトの検討
- ファイル高速加工ソフト、計算ミドルウェアの保守期限到来　⇒ 代替製品・別方式の検討
- アプリ・運用機能は極力そのまま移植し、改修範囲を抑えコスト削減を図る

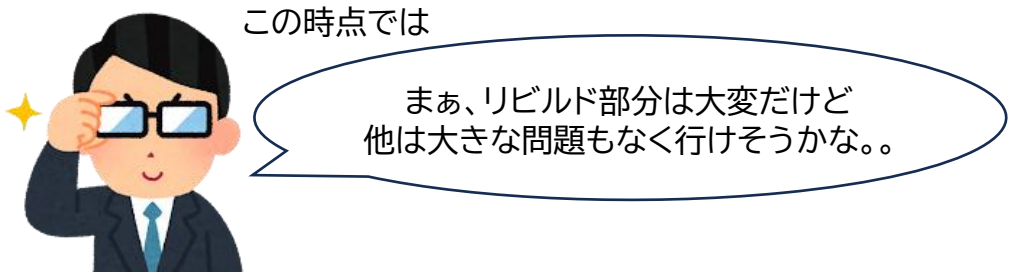
1.更改システムの概要とPJの背景(事例)

＜システム更改の方針＞

分類	構成要素	方針
インフラ	稼働環境	顧客の他システムで実績のあるAWSへリフト&シフト ※1
	サーバ/ストレージ	日次/月次を同一のEC2+EBSに統合
	DB	アプリ影響を鑑み現行と同じOracleを選定(Exadata ⇒ RDS Oracle) ※2
アプリ	Java・シェル	日次/月次統合に伴う改修以外は現行踏襲
	ファイル高速加工ソフト	現行の処理時間を維持するためPySpark+EMRでの分散処理方式にリビルド
	計算ミドルウェア	後継の同製品へ移植

※1. データ所在に関する規定およびレイテンシーを考慮し、AWSは東京リージョンに構築

※2. Oracle Database@AWSは東京リージョンでの提供はまだ未発表のため選定不可



2.性能面での問題(1)

データベースエンジンは同じOracleだし性能に関して大きな差は出ないんじゃない？
と思ったのも束の間。。性能問題が勃発！

DB参照/更新処理が20倍以上遅くなった

と、とにかくAmazon ClowdWatchメトリクスとAWRレポートを見てみよう

2.性能面での問題(1)

原因①：DBアーキテクチャ相違による処理性能の劣化

- ExadataのSmart Scan機能により高速だった参照系SQLが長走化
- ExadataのH/W(Storage Server)により高速だった更新系SQLが長走化

原因②：最適でない実行計画と統計情報収集の長走化

- Oracleの「適用計画」機能により最適でない(遅い)実行計画が採用された
- データ更新後のグローバル統計情報取得に時間が掛かり長走化

原因③：RDS OracleのI/Oスループット不足

- RDS Oracleのストレージ(EBS)のI/Oスループットが頭打ち

2.性能面での問題(2)

Oracleだけじゃなく計算ミドルウェアにも性能問題が！

日次/月次処理の同時実行時の処理時間が増加

これもとにかくAmazon ClowdWatchメトリクスを見てみよう

2.性能面での問題(2)

原因：日次/月次サーバ統合によるディスクI/Oの集中

・EC2の同一EBSボリュームへのI/O集中(スループットが頭打ち)

3.対策の検討(1)

DB参照/更新遅延への対策

原因①への対策

- RDS Oracleのインスタンスクラスの変更(スタンダード→メモリ最適化)とSGA拡張
- Smart Scan機能に依存したSQLの修正
- Domaによるバッチインサート処理の効率化

原因②への対策

- 適用計画機能の停止
- グローバル統計→増分統計に取得方法を変更
- 統計情報取得並列度の増加

3.対策の検討(1)

原因③への対策

・RDS Oracleのインスタンスクラス変更によるEBS帯域幅の増強

3.対策の検討(2)

日次/月次同時実行への対策

原因への対策

・EBSのボリュームタイプ変更(汎用SSD→プロビジョンドIOPS SSD)

3.対策の検討

何とか各性能問題への対策を施してSLAを遵守。
（変換はSLA「7時間」を分散処理で3時間に短縮）

しかし安堵したものの振り返りは大事。
（隣のPJの話だったとしても、明日は我が身ということも大いに考えられる）

■ 振り返り

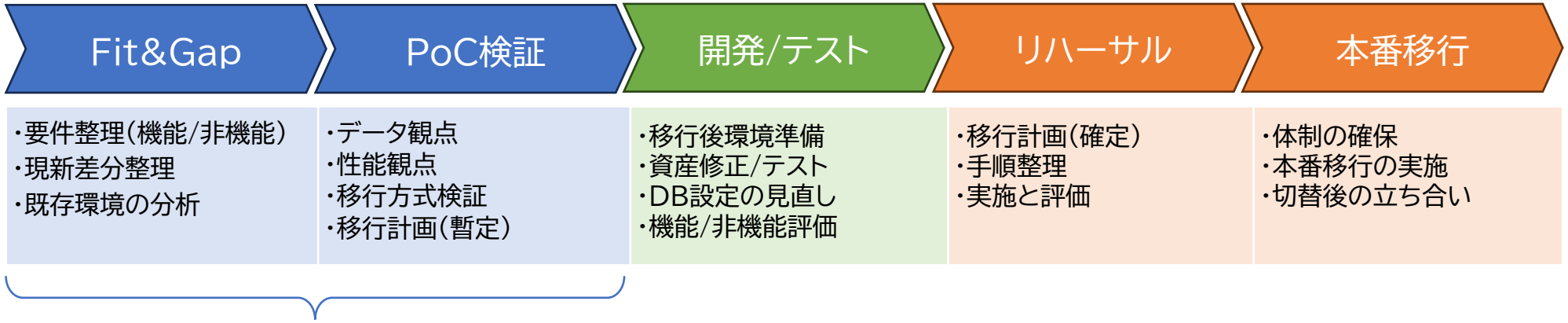
- DBに限らずアプリが製品固有機能に依存した実装を行っていないか事前の調査は念入りに
- CPU・メモリ・Diskなどリソースの上限値だけでなく、NWやI/Oなどスループットも構成検討の要素に含める
- デフォルトONの機能が知らずに悪影響を及ぼさないか確認をしておく
- クラウドシフトの場合、スペックでの課金だけでなくリクエスト数や転送データ量などの課金も把握しておく

4. RDS移行に関するプロセス

① DBのAWS移行に向けた作業フロー

データベースエンジンが同一の場合、データ型、スキーマや権限、SQL、プロシージャなどの互換性が担保されるため比較的に移行工数は少なくなりますが、今回（Exadata→RDS）の様な**Re-Platform**の場合、SQL（DDL、DML）やプロシージャーなどそのまま流用できない（動くけど性能劣化になる）場合があるため、事前に調査・検証を行います。

環境構成の検討は、机上だけでなく実機での検証結果を踏まえて確実に実施することが重要です。



この工程での検討・検証が大事！

4. RDS移行に関するプロセス

② DB移行のステップ(1/2)

データベースを移行する際には段階的なフェーズを設け、検討・検証を実施しながら進める必要があります。

移行フェーズ	検討項目	検討項目概要
Fit&Gap	要件の再確認	制限事項、機能要件、非機能要件、移行要件、運用要件を再確認 ⇒ そもそもの要件って？
	既存環境情報整理	OS情報、DB性能情報(AWR、ASH)、DB設計書、データディクショナリ情報、システム構成情報、ネットワーク構成情報、運用設計書、運用コスト情報等の収集 ⇒ 新旧対比して整理
	既存環境の性能分析	既存環境のAWR等からデータベースのワークロードの状況、最大負荷時のリソース使用量等を整理 ⇒ 余剰/適量/不足？
	利用してるフレームワーク、パッケージ整理	DBサーバ上にインストールが必要なパッケージ製品、利用しているフレームワークを整理 ⇒ インフラ/アプリどちらも
	利用機能整理	AWS上で提供されるRDSでは利用不可となる機能があるため、既存システムの利用している機能を整理し、移行に向けて代替案の検討が必要か否かを整理 ⇒ より具体的な対応方法まで整理
	ライセンスの利用方式検討	AWSへ移行するにあたり、データベースのライセンスの扱い方を整理 ※BYOLやLicense Includeのどちらにするか等 ⇒ 切替え方法、並行稼働期間の契約
	移行先アーキテクチャ検討	移行の評価を行うに辺り、移行先のアーキテクチャの概要を机上検討 ⇒ 暫定版の環境構成
	サイジング	既存環境の性能分析結果よりAWSに移行した際にどの程度のリソースが必要となるかを机上試算 ⇒ 暫定版のサイジング

4. RDS移行に関するプロセス

② DB移行のステップ(2/2)

データベースを移行する際には段階的なフェーズを設け、検討・検証を実施しながら進める必要があります。

移行フェーズ	検討項目	検討項目概要
PoC検証	移行PoC環境構築	移行先アーキテクチャを想定した移行PoC環境を構築 ⇒ なる早で構築し検証期間を確保
	移行PoC実施	移行PoC環境を用いて検証を実施(データバリエーション、データフローと処理タイミングを意識した性能検証、移行方式の妥当性/所要時間の見積り) ⇒ より実運用を意識した観点を設ける
開発/テスト	移行後環境構築	移行後の構成で開発/テスト環境を構築 ⇒ 環境情報はインフラ/アプリで共有
	資産・設定の変更/テスト	書き換えが必要な資産(SQL、プロシージャ等)の修正、製品パラメータ見直し ⇒ 機械的な置換の可否、再置換を考慮したツール作成も検討
	実績評価	機能/非機能面で実績を評価し、必要に応じて見直しを実施 ⇒ これを繰り返す
リハーサル	計画・手順整理	前提、制約、スケジュール、タイムチャート、チェックポイント、完了条件、コンチ等の計画と手順作成 ⇒ 自システム/関連他システム/ユーザ間で共有し承認を得る
	移行リハーサル実施	移行リハーサルを行い、移行方法、手順、所要時間の妥当性を検証 ⇒ リハーサルと本番時の差異整理のうえ懸念・リスクは事前に潰しておく
	実績評価	発生したインシデントの取り込み、移行計画や手順の見直し ⇒ 変更に伴う影響も確認し承認フローへ
本番移行	移行切替	移行計画に即してシステム移行/データ移行を遂行 ⇒ スクランブル発進に備えた体制確保

4. RDS移行に関するプロセス

③ Fit＆Gap、PoC検証フェーズにおける確認項目（AWS公開参考情報）

RDS移行におけるFit＆Gapや性能評価を目的としたPoC検証の際は、以下の確認と整理が重要です。

移行フェーズ	確認項目	説明
Fit＆Gap	Oracle機能への依存	パラレル処理やパーティション機能などOracleの特定の機能やライセンスオプションへの依存が無いかを確認する
	RDSの制限事項	RDSの制限事項や機能差異による影響を受けるか確認する
	現行環境の性能分析	AWR（Automatic Workload Repository）レポートより現行システムの負荷状況を確認し、CPUコアやIOPS、スループットといった性能要件を整理し、DB全体としてRDSに載せ替え可能か判断する
	機能代替の可否	Oracleの機能や性能がRDSで代替可能で移行のノックアウトファクターとならないか確認する
PoC検証	ベンチマーク指標の定義	性能評価時の基準とするSQLクエリやバッチ処理を明らかにする
	平常時の処理速度	平常時のスループット基準（クエリ秒やバッチ処理時間）を明らかにし、性能要件を満たせるか確認する
	ピーク時の処理速度	業務負荷の高い時間帯や期間などから、ピーク時の基準を明らかにし、性能要件を満たせるか確認する
	並列処理数	どの程度の並列度をもって捌けるか、単体クライアントからの並列処理や複数台での負荷を試験する
	データ連携と運用	他システムとの連携や定期運用の処理が性能面で問題ないか確認する
	障害時のレスポンス	フェイルオーバーや再起動時の影響を確認する（AZ跨ぎの影響も合わせて確認）

参考： AWS規範的ガイダンス - Oracle Exadata から AWSへの移行を成功させるための設計図
https://docs.aws.amazon.com/ja_jp/prescriptive-guidance/latest/oracle-exadata-blueprint/feature-summary.html

4. RDS移行に関するプロセス

④ ExadataとRDS for Oracleに関する主要な機能差異（AWS公開参考情報）

RDS for OracleではExadata固有の主要機能（以下 表）が利用できないため、移行に際しては既存システムにおけるExadata固有機能の利用状況や代替案の検討が必要となります。

カテゴリ	機能名	機能概要
Exadata	Smart Scan	ストレージ層でDBサーバの処理に必要なブロックを絞って転送する技術であり、特にFull Scan時のI/Oの最適化機能
	Storage Index	ストレージサーバ上で透過的に作成される索引を用いてSQLの絞り込み条件を満たすデータのみをスキャンする機能
	Hybrid Columnar Compression（HCC）	Exadata環境でのみ使用できる. 強力なデータ圧縮機能
Oracle Database	Automatic Storage Management（ASM）	複数のストレージを束ねることによって処理効率の向上等を行うストレージ管理ソリューション
	Database Vault	管理アクセス権を持つユーザーを含め、あらゆるユーザーからのOracleデータベースの特定領域へのアクセスを制限する機能
	Recovery Manager （RMAN）	データベースでバックアップおよびリカバリ・タスクを実行し、バックアップ計画の管理を自動化する機能
	データベース冗長接続プーリング （DRCP）	多数のクライアント間で共有して利用されるサーバーの接続プールを提供する機能
	統合監査（Pure モード）	様々なOracleの監査証跡を統合的に管理し証跡を取得する機能
	Multitenant	複数のデータベースを1つのデータベースに統合し、管理性、データベースの分離、およびセキュリティを向上させる仕組み
	Real Application Clusters（RAC）	高可用性および高スケーラビリティを提供する仕組み

参考： RDS for Oracle でサポートされていない機能
https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/UserGuide/Oracle.Concepts.FeatureSupport.html#Oracle.Concepts.FeatureSupport.unsupported

4. RDS移行に関するプロセス

⑤ Exadata機能の代替案（AWS公開参考情報）

RDSへの移行検討に際して、Exadata 固有の機能を考慮する一般的な代替案は以下となります。

Exadataで利用していた機能	機能ギャップに対処するための代替案
Smart Scan	メモリ最適化インスタンスを使用する
	SGA/PGA 設定を最適化する
	optimizer_index_cost_adj などのオプティマイザパラメータを調整する
	追加のスキーマインデックスを作成する
	SQL を最適化して I/O フットプリントを減らす
Storage Index	適切なスキーマインデックスを作成する
Smart Flash Cache	メモリ最適化インスタンスを使用する
	SGA設定を最適化する
	ローカル SSD ストレージを使用して、Amazon RDS for Oracle でデータベースフラッシュキャッシュ機能を設定する
	Amazon ElastiCache などの外部キャッシュソリューションを使用する
	Amazon RDS for Oracle のストレージレイヤーとして io2 Block Express EBS ボリュームを検討する

参考： AWS規範的ガイダンス - Exadataの機能とAWS代替方法の概要
https://docs.aws.amazon.com/ja_jp/prescriptive-guidance/latest/oracle-exadata-blueprint/feature-summary.html

5. まとめ

POINT 01

性能改善はアプリの改修だけではなく、基盤の構成見直しも視野に入れて解決策を検討する。
（無理ゲーをいくら頑張っても限界がある。疲弊して全体の品質を落とす危険も. . ）

POINT 02

構成の検討前にはアプリの処理方式やデータフロー、タイミングなど詳細な仕様を見える化
（ポンチ絵など具体的にイメージを共有）してアプリとインフラで協議しておくことは大事。
（言われてない、聞かれてない、多分こうで突き進むと、どこかに落とし穴が出来てしまう）

POINT 03

風通しの良いチームビルディングと適切な会議体の設定はPJの成否に大きく関わる。
（縦割りではなく皆で忌憚なくアイデアを出し合える場と人間関係の構築は大事）

ご清聴頂きありがとうございました。

無限の未来と、
幾千のテクノロジーをつなぐ。

CTC Financial Services Group

CTC
Challenging Tomorrow's Changes