
aws
SUMMIT
ONLINE

SAP BusinessObjects 向けクエリパフォーマンス向上の取り組みと、Amazon Redshift Concurrency Scaling の活用による効果

小林 裕也
情報統括部
株式会社コーセー

aws SUMMIT ONLINE

自己紹介

小林　裕也（こばやし　ゆうや）

株式会社コーセー　情報統括部

略歴：

・SIerで様々な企業のBIシステム開発を経験

・2018年に株式会社コーセー　入社

　主にデータ分析基盤のプロダクトを担当

アジェンダ

会社紹介

コーセーの分析基盤

パフォーマンス向上への取り組み

Concurrency Scaling活用による効果

会社紹介

aws SUMMIT ONLINE

コーセーの概要

- 株式会社コーセー
  - 創　業　　1946年3月2日
  - 従業員数　　7,758名 (連結)
  - 事業内容　　化粧品の製造、販売
  - 売上高　　3,329億万円 (19年3月期)
  - 営業利益　　524億円 (営業利益率 15.7%)

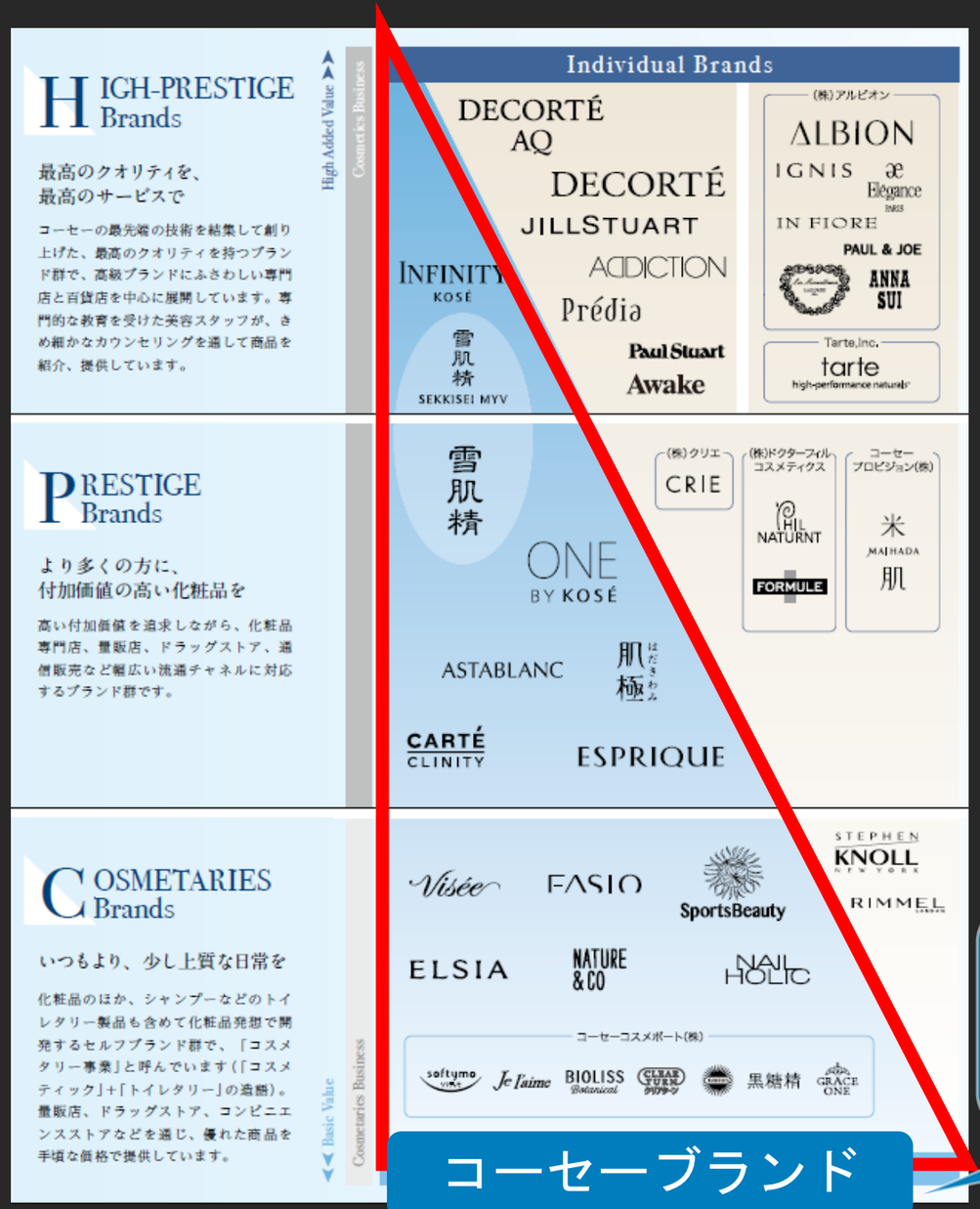
コーセー研究所 (北区王子)

コーセー本社 (日本橋)

- 本社　　東京都中央区日本橋
- 営業所　　全国39支店
- 研究所　　東京都北区・板橋区
- 工場　　埼玉県・群馬県
- 研修センター　　東京都北区王子
- 関係会社　　コーセーコスメポート
  アルビオン他

コーセー王子研修センター

コーセーのブランド戦略



コーセーブランド

コーセーの名前を出さず、ブランド独自の世界観を重視したブランド展開

コーセーの名前を全面に出したブランド展開

インディビジュアルブランド

コーセーの分析基盤

aws SUMMIT ONLINE

分析基盤が構築される前の背景・課題

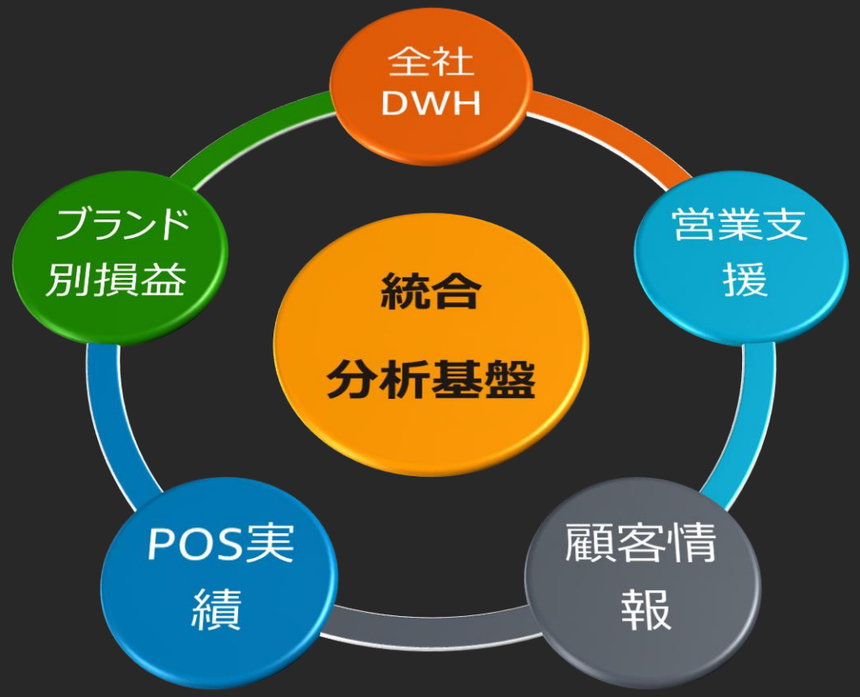
背景・課題

部門や業務ごとに個別最適化された
データウェアハウスを構築

- 複数の情報系システムが存在し管理費が増大
- システム毎に利用方法が異なるため利用者に負荷
- マーケティングデータ分析ニーズに対応できない
- 集計ロジック分散によるシステム毎の結果の差異

解決

統合分析基盤（ｻｲﾛ化ﾃﾞｰﾀの統合）
意思決定に必要な情報を一元管理



統合分析システム構築の効果

◆データベース統合によりデータ抽出効率が向上

- 情報の一元管理
- システム操作方法の統一化

◆iPADの活用

- 店頭提案力・説得力の強化
- 隙間時間の有効活用（時間短縮）

◆必要なデータの自由分析を実現

- 社内のデータ活用推進（教育啓蒙）
  - 初級者：欲しいデータが簡単に（定型）
  - 上級者：より高度な分析（応用）
- 活用スキルの統一化（ナレッジの蓄積）

◆BIツールの最新機能を利用

- 分析機能、表現力（グラフ）の強化
- 成功事例の共有化

◆クラウド（AWS）利用により負荷変動に柔軟

- 既存システムの老朽化対策
- 導入コスト、運用コストの削減

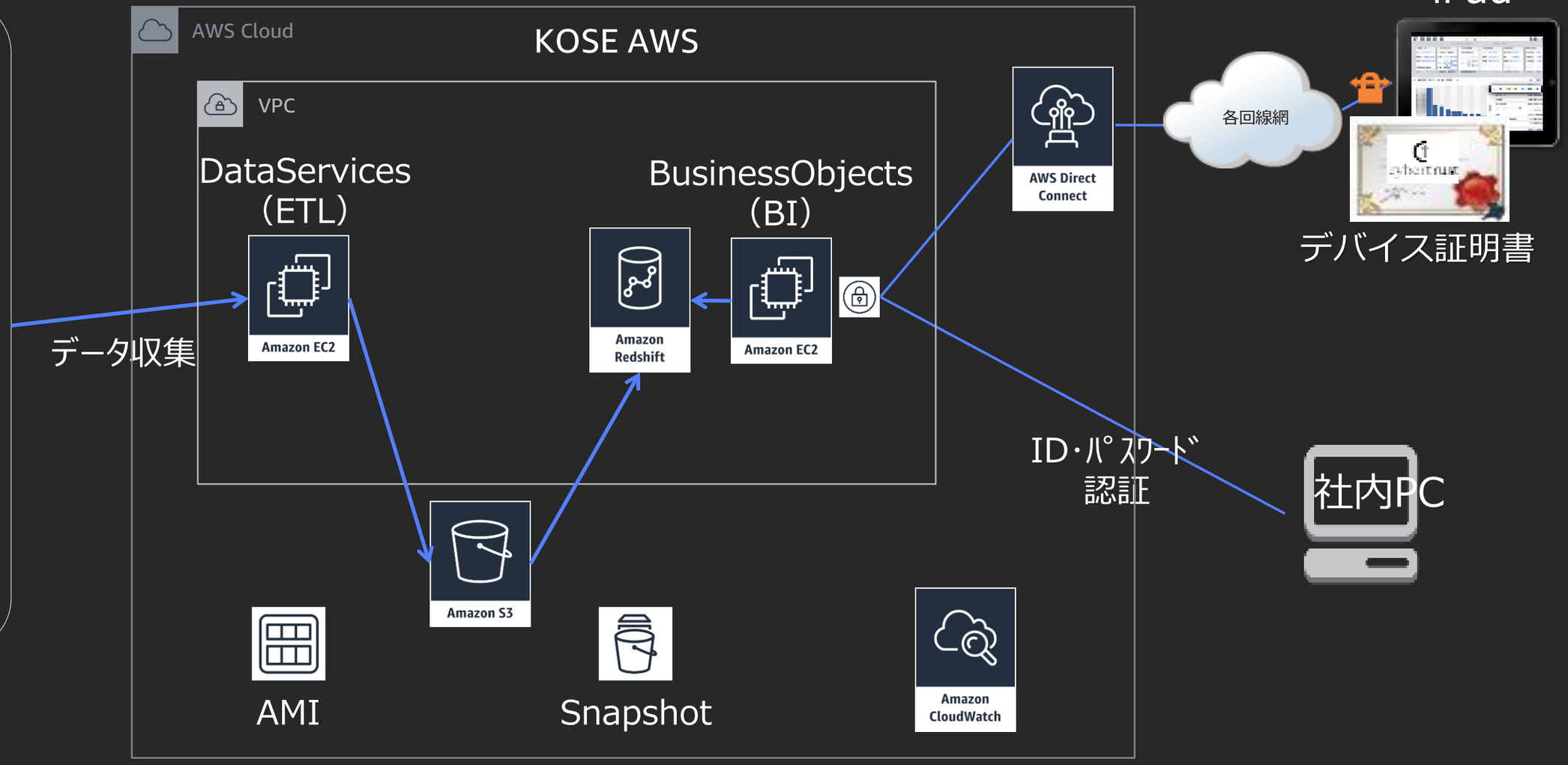
統合分析システムの構成

細かく話さなくてよい

社内システム

統合分析基盤

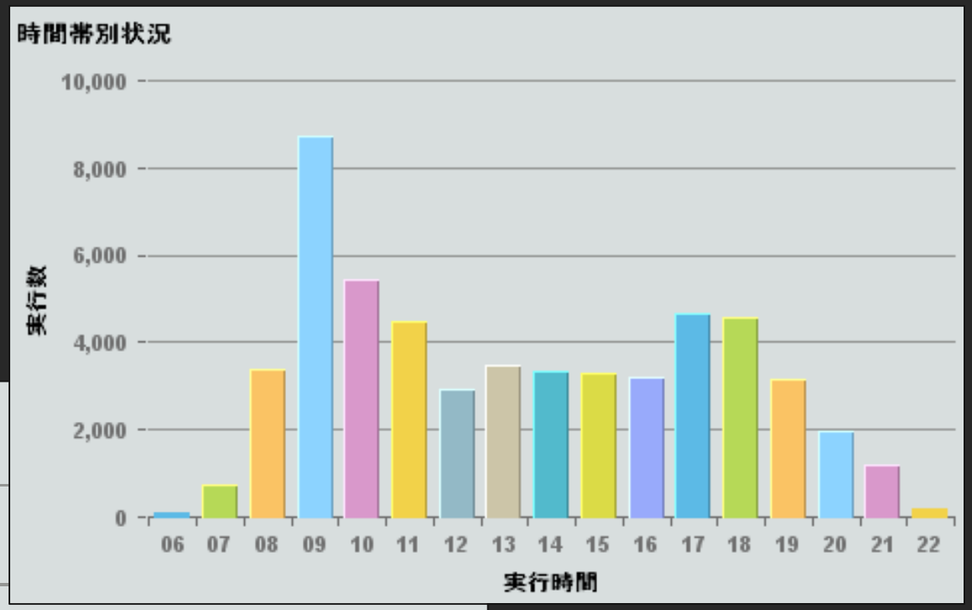
- マスタ
- 基幹システム
- 販売システム
- 店頭システム
- 損益システム
- ・
- ・
- ・

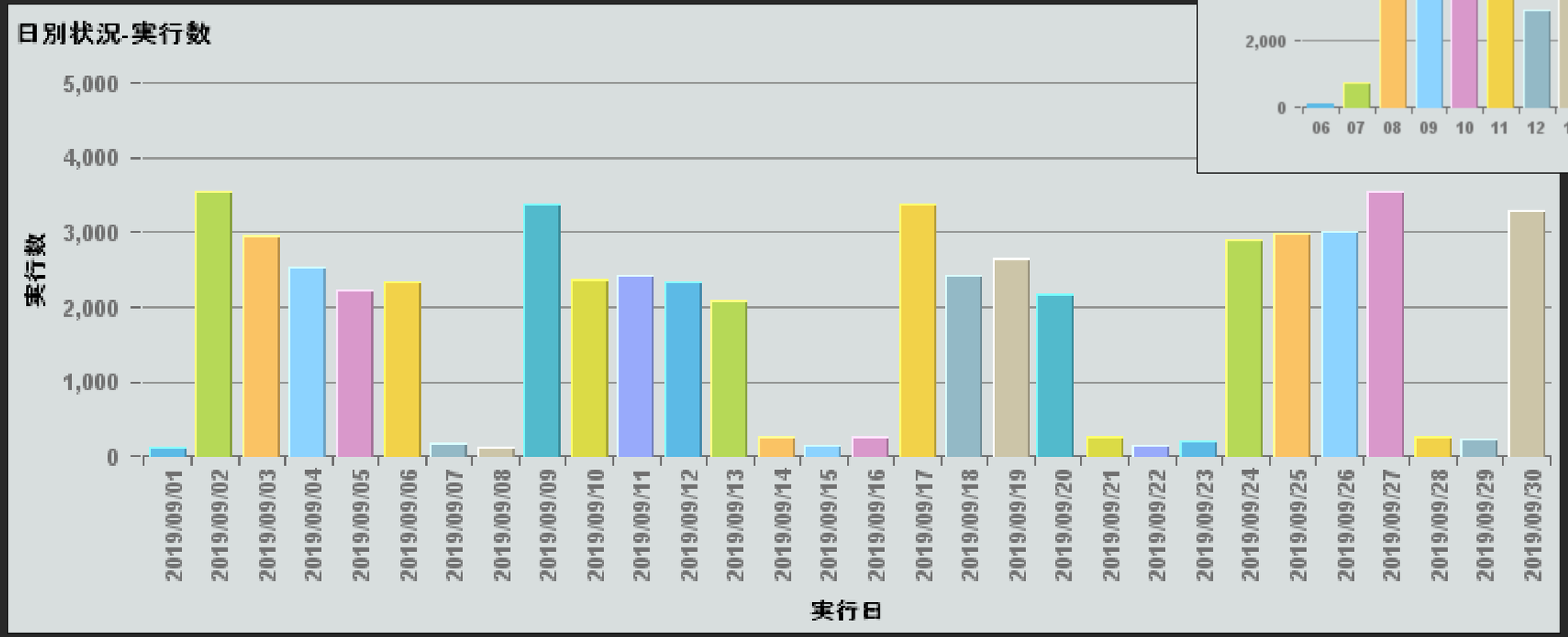


システムの利用状況

※( )は公開時

利用ユーザ数	**約1500人** (約1000人)
ピーク時の同時利用数	**平均100セッション** (平均15セッション)
1日あたりの利用者数	**平均400人** (平均50人)
1日あたりの検索数-休日除く-	**平均2800検索** (平均700検索)



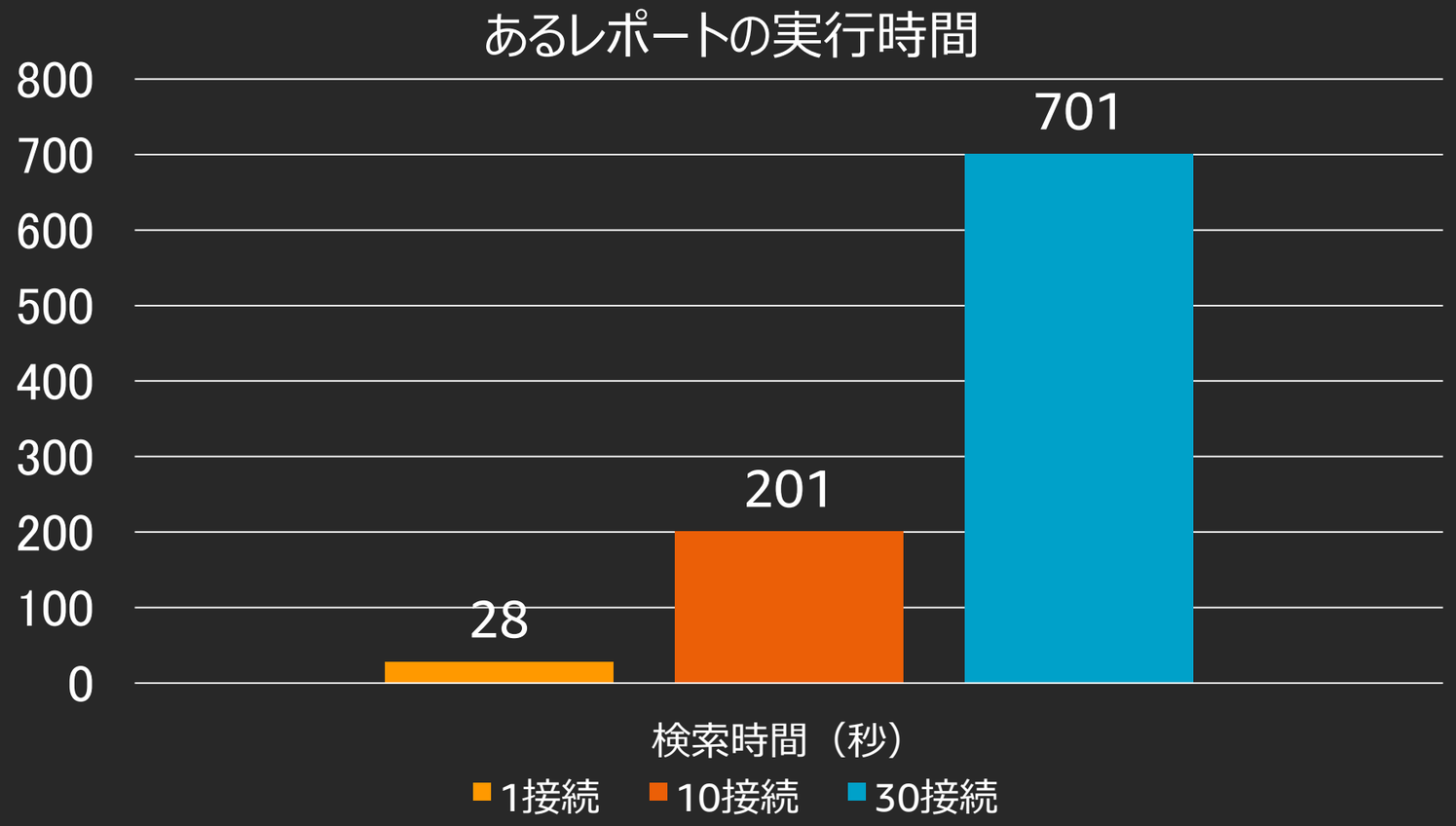


2019/9時点

利用者が増えることで抱えた問題・・・

- 想定していたパフォーマンスがでない
  - 複数人が同時実行しただけでレスポンスが悪化
  - ピーク時の検索は通常の数十倍時間がかかるケースもある

単一の検索では２８秒程度
同時検索が増えるにつれレスポンス悪化



ボトルネックとなっていた部分

ログを検証した結果、レポートが表示されるまでの時間の内、**90%がRedshiftのクエリ処理時間**であることが判明

クエリを分析すると、**20%のクエリが処理待ち状態となっていた**

通常数秒程度で結果が出力されるクエリも数分程度の時間を要していた

Redshiftは**同時実行に弱い**ことが判明

パフォーマンス向上への取り組み

aws SUMMIT ONLINE

Redshiftの設定

Redshift (dc1.large ノード数:10)　※ノードタイプは後にdc2.largeに変更

- WLMは３つのキューによって構成
  - 検索用 (メモリ割当：70%、並列数：5)
  - バッチ処理用 (メモリ割当：20%、並列数：2)
  - 管理者用(メモリ割当：10%、並列数：1)

- テーブル構造
  - 圧縮タイプはデフォルト
  - マスタはALL分散、トランザクションはEVEN分散

取り組み① - WLMの設定変更

WLM（Work Load Management)

クエリに対して割り当てるメモリの割合や並列度、タイムアウトの時間を指定する事でクエリに対してリソースの配分や長時間実行されるクエリを止めてクラスタリソースを無駄遣いしないようにすることが目的

デフォルトは５

並列数は最大で５０まで設定できるようだが、公式のベストプラクティスでは、キュー全体で１５以下が推奨

WLMの検索用キュー並列処理数を　「2」、「5」、「12」　で30同時実行し、レスポンス測定

結果（×）・・・顕著な改善は見られなかった

取り組み① - WLMの設定変更

取り組みの副産物として分かったこと

- 設定数を減らす・・・1クエリあたりのメモリ割当量が増え、単体のクエリパフォーマンスは向上
- 設定数を増やす・・・同時に実行できるクエリは増えるが、メモリ割当量は減るため、1クエリあたりのパフォーマンスは低下

改善は見られなかったものの、処理性能 < 並列処理数 を重視し、検索用キューのクエリ並列処理数は「１２」で設定

取り組み② - 圧縮タイプの変更

Redshiftは列レベルで圧縮形式を選ぶことができ、適切な形式を選択することで以下のような効果を発揮

・ストレージスペースが節約

・データのサイズが軽量化 ⇒ ディスク I/O の量が減少

⇒ クエリパフォーマンスが向上

結果（〇）・・・圧縮タイプを再設定することで、ディスクＩＯが削減、パフォーマンスが改善

取り組み③ - カラムサイズの変更

Redshift は列データを非常に効果的に圧縮するため、必要なサイズよりかなり大きな列を作成しても、データテーブルのサイズにはあまり影響は発生しない

但し、複雑なクエリの処理中に、中間クエリ結果を一時テーブルに格納される場合があり、**一時テーブルは圧縮されない**ので、不必要に大きな列がメモリ及び一時ディスク領域を過剰に消費し、**クエリパフォーマンスに影響を与える可能性がある**

全体的にカラムサイズを見直し、縮小して、クエリを実行

結果（×）・・・顕著な改善は見られなかった

取り組み④ - ソートキーの変更

頻繁に使用される特定のカラムに対して、ソートキーを設定し、ディスクアクセスの範囲を最小にする

ソートキーの注意点

- テーブルにソートキーを付けても自動でソートはされない
- 自動ソートされるのは、空のテーブルに一括ロードした場合のみ
- 追加データをロード、INSERTをした場合、VACUUMコマンドが必用

高頻度で利用される年月日をソートキーとして設定、処理時間を検証

結果（×）・・・顕著な改善は見られなかった

取り組み⑤ - RDSの導入

様々な取り組みを行ったが、同時処理能力を根本的に解決することはできなかったため、リソース増強に踏み切る

同時処理能力が求められる機能で利用するテーブルをRDSに逃がし、Redshiftの負荷を減らす方式を採用　⇒　RDSの導入

RDSはPostgreSQLを採用

（選定理由）

- 同時検索に強い
- RedshiftとSQLの親和性が高い
- RedshiftとのDBリンクが可能

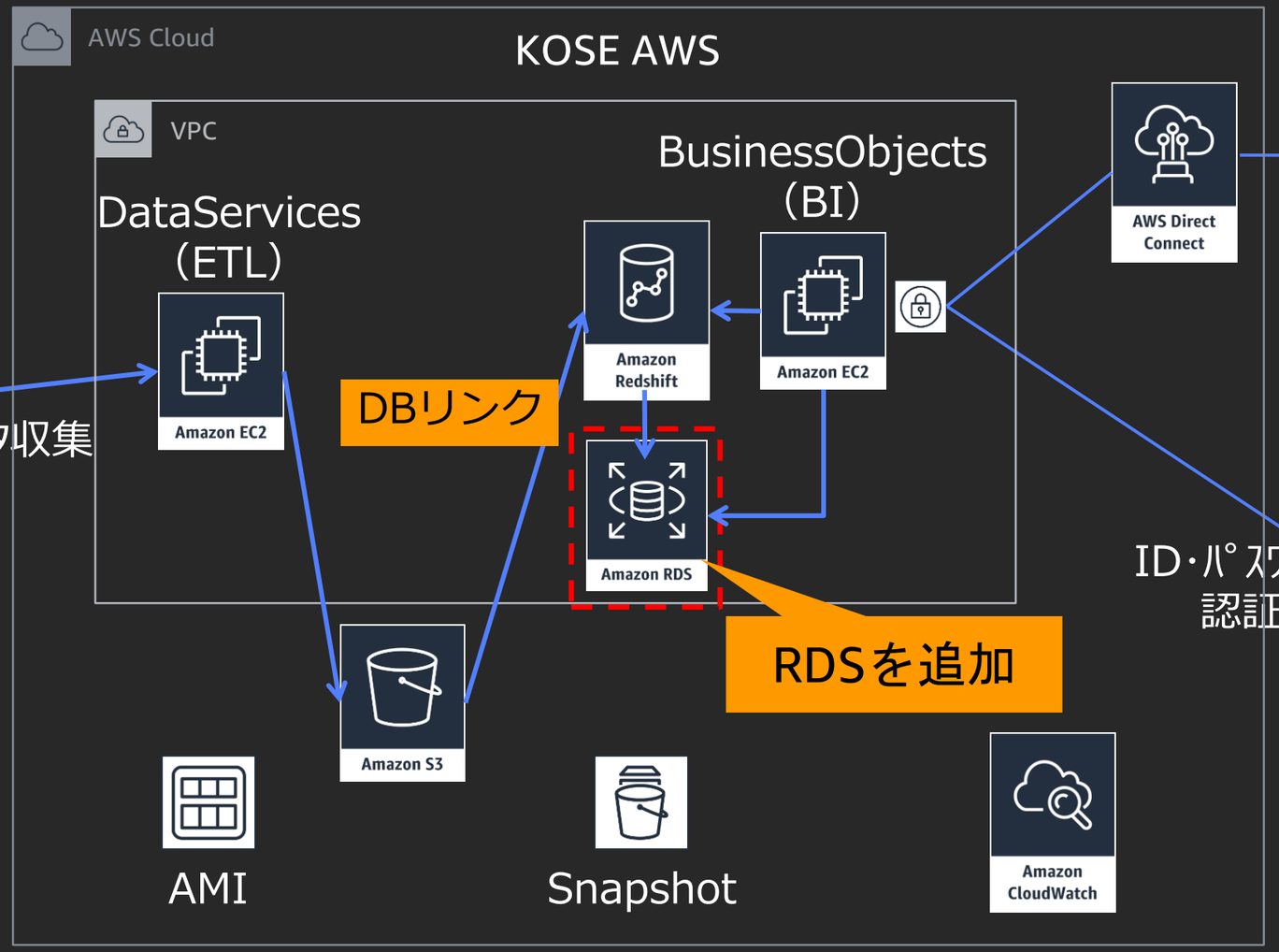
結果（◎）・・・並列処理に強いデータベースであったことから、同時検索時のパフォーマンスは劇的に改善

統合分析システムの構成　※RDS導入後

社内システム

- マスタ
- 基幹システム
- 販売システム
- 店頭システム
- 損益システム
- ・
- ・
- ・

統合分析基盤



取り組みの結果

Redshift側でのパラメータ調整ではシングル実行での改善は見込めるが、同時接続問題への根本的な解決はしなかった

RDS導入は大きな効果があり、データベース周りの負荷は分散され、システムは一定の安定稼働をむかえることができた

そして月日は流れ・・・

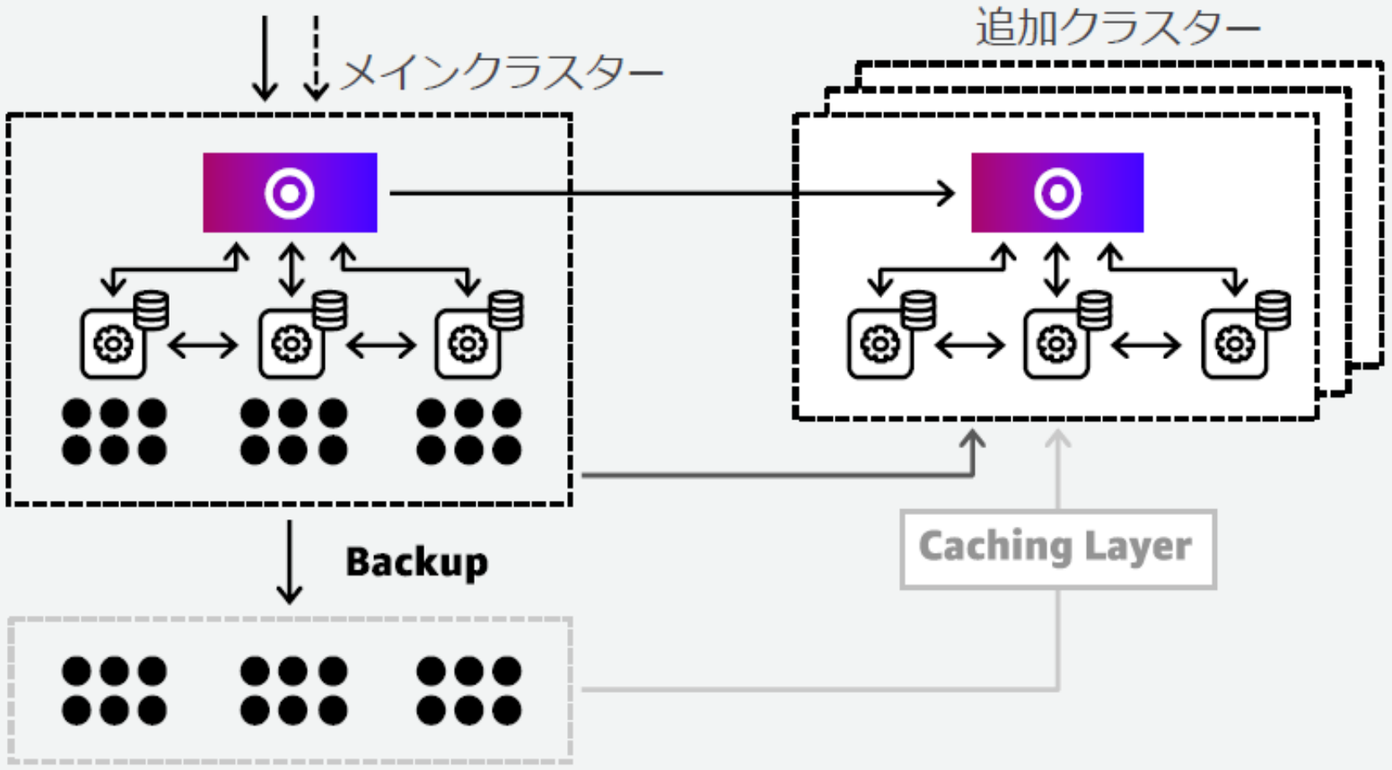
２０１９年３月３０日、「**ConcurencyScaling**」が登場

Concurrency Scaling活用による効果

aws SUMMIT ONLINE

ConcurencyScalingとは

※AWS社資料抜粋



- Redshift上で実行されるクエリの同時実行性が高まったときにバックグラウンドでクラスターを追加し、クエリ処理をオフロードすることにより、処理を待たせることなく実行できるようにする**WLM機能の拡張**
  - 追加クラスターは数秒でアタッチされて処理が可能な状態となる
- ピークワークロードに発生するクエリのキュー待ち時間を排除するシンプルかつコスト効率の高い方法
  - 例：毎朝9:00～9:30、出社直後に多くボードをリフ

要するに同じ性能を持つクラスターが複数追加され、増えた分だけ同時実行数が倍々で増える機能

ConcurencyScalingとは

※AWS社資料抜粋

Concurrency Scaling の価格

- 各追加クラスターで**クエリが実行された期間（秒数）**に対して課金（最小1分）
  - メインクラスターの**ノードタイプのオンデマンド単価×ノード数**で計算
- ただし、24時間あたり**1時間分は無料**の Concurrency Scaling クレジットが付与される
  - 無料のクレジットは、クラスターあたり最大 30 時間分を累積可能
- 例）東京リージョンの**dc2.large コンピュートノード10台**構成の場合で、無料の Concurrency Scaling クレジットを超えて、**2つの追加クラスターでそれぞれ30分間クエリが実行**されたケース
  - メインクラスターとしてのオンデマンド単価：0.314 * 10 * 1/3600 = 0.009ドル/秒
  - Concurrency Scaling にかかるコスト： 0.009ドル/秒 * 1,800秒 = **1.57ドル**

ポイントは、CSが利用された分だけ課金される体系で、１日１時間分は無料。最大で３０時間分をクレジット可能

http

aws

ConcurencyScalingを適用してみた

ワークロード管理のパラメータより、ConcurencyScaling（以降CSと表記）の最大追加クラスター数を「１」⇒「５」に変更　※最大は１０

パラメータ

次のパラメータは、このパラメータグループに関連付けられているすべてのクラスターのデータベース設定に影響を与えます。これらのパラメータ値を変更した場合、それはクラスターを再起動するま
りません。WLM 設定を表示および設定するには、[WLM] タブを選択します。 詳細はこちら。

名前	タイプ	値	使用できる値
auto_analyze New	boolean	true	true,false
datestyle	string	ISO, YMD	
enable_user_activity_logging	boolean	true	true,false
extra_float_digits	integer	0	-15-2
max_concurrency_scaling_clusters	integer	5	0-10
max_cursor_result_set_size	integer	19200	0-14400000
query_group	string	default	
require_ssl	boolean	false	true,false
search_path	string	$user, public	
statement_timeout	integer	0	0,100-2147483647

ConcurencyScalingを適用してみた

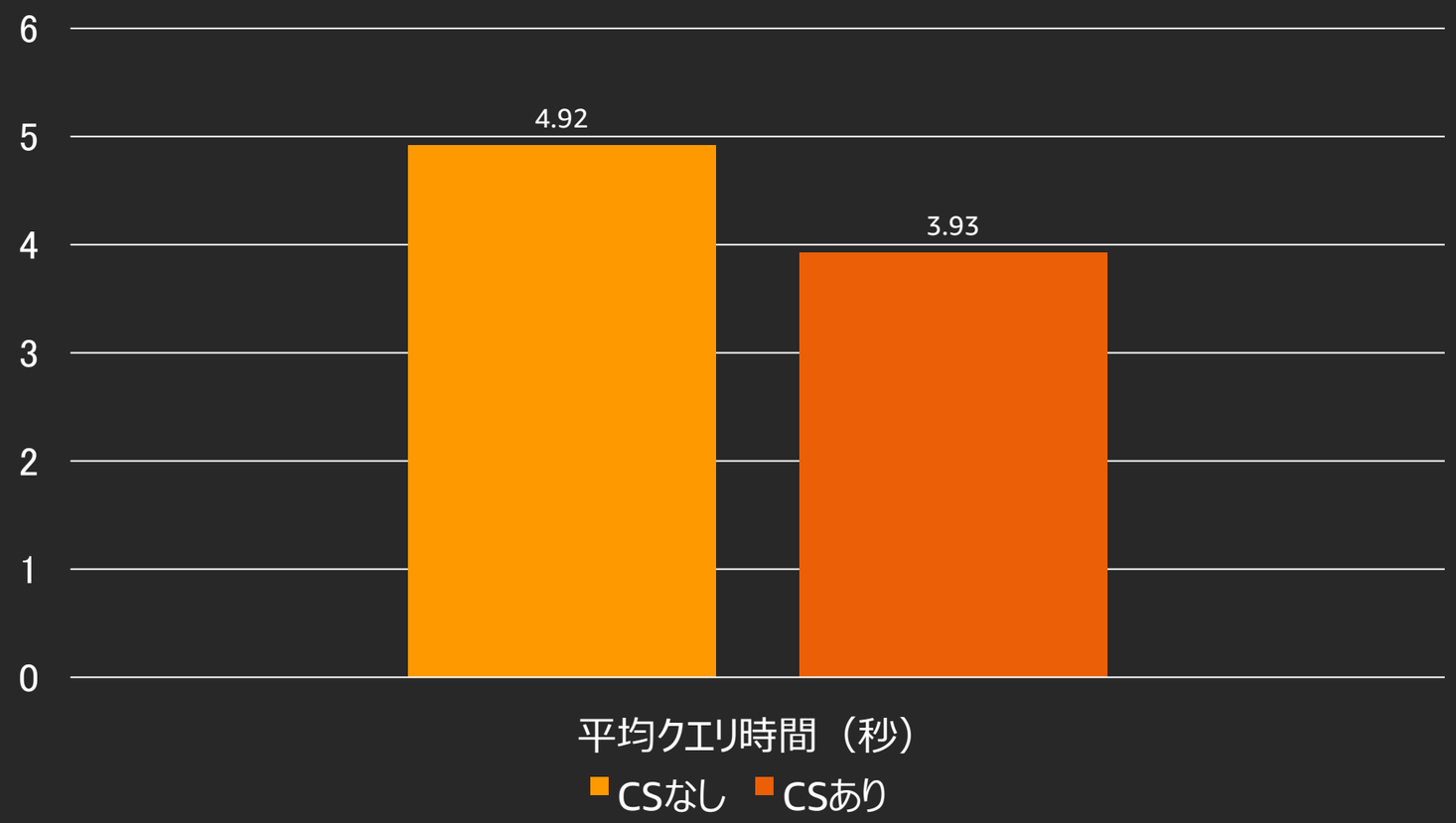
ワークロード管理から、CSを適用したいキューの”Concurency Scaling mode”を、「off」 ⇒ 「auto」 に変更

これで完了 （非常に簡単）

効果検証

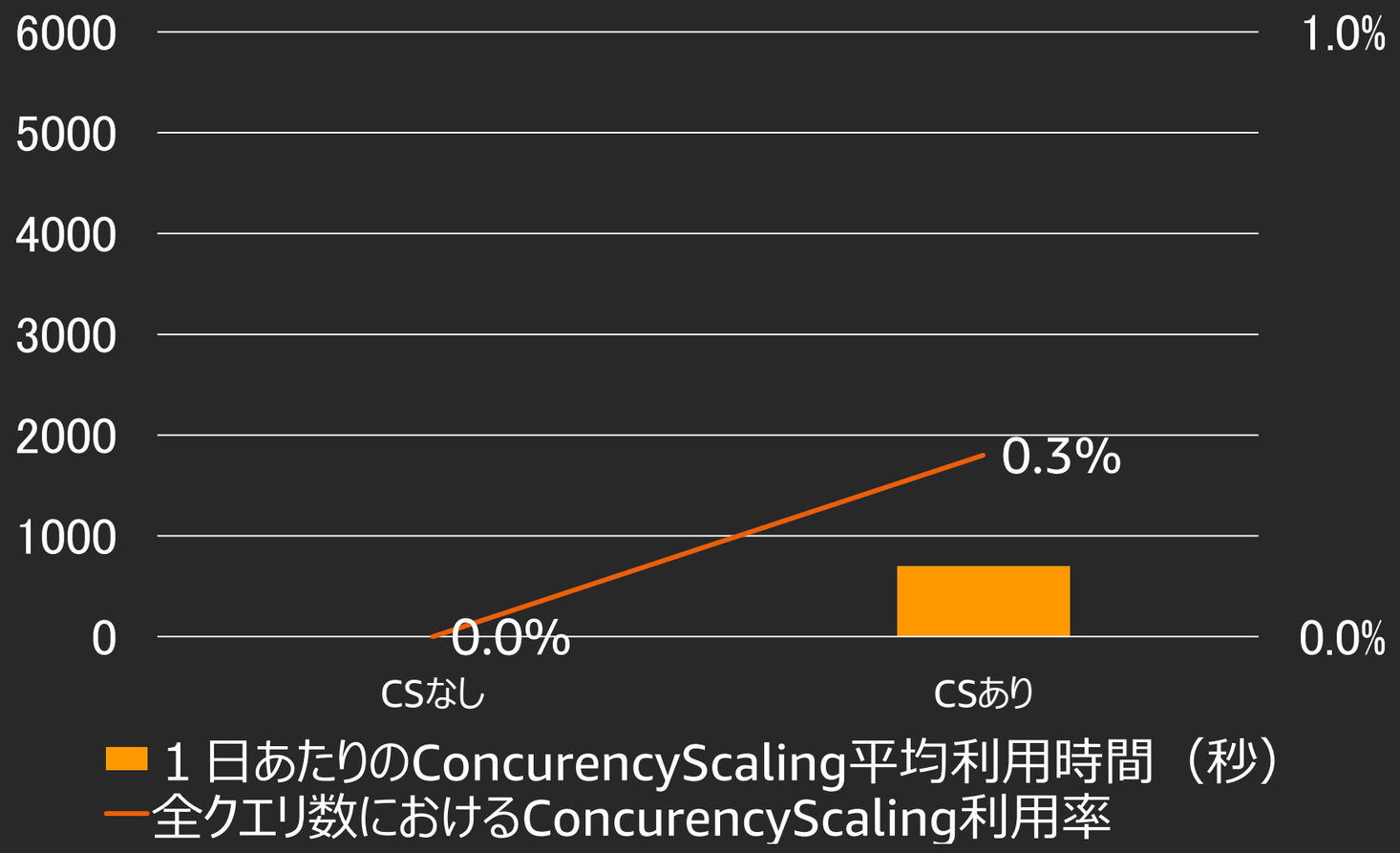
1か月間システムを通常稼働し、効果を測定

その結果・・・設定するだけで約２０%のパフォーマンス改善を確認



効果検証

効果は見えたものの、CSの利用状況を確認したところ、
実はそこまで追加クラスターがアタッチされていなかった



CSを十分に活用するためには、適切な設定を行うことが必要！

効果検証と推測

CSの利用時間とクエリを再度分析し以下の傾向を発見

・クエリ実行時間が１０秒以上かかるものが全体の２０%弱を占める

・１分を超えるロングランクエリが比較的多い

・クラスターがアタッチされる際は、６０秒以上のキュー待ちクエリが発生している

分析結果をもとに推測・・・

SLOTあたりのメモリ割当量が低いため１クエリ処理に時間がかかり、ロングランクエリが発生？

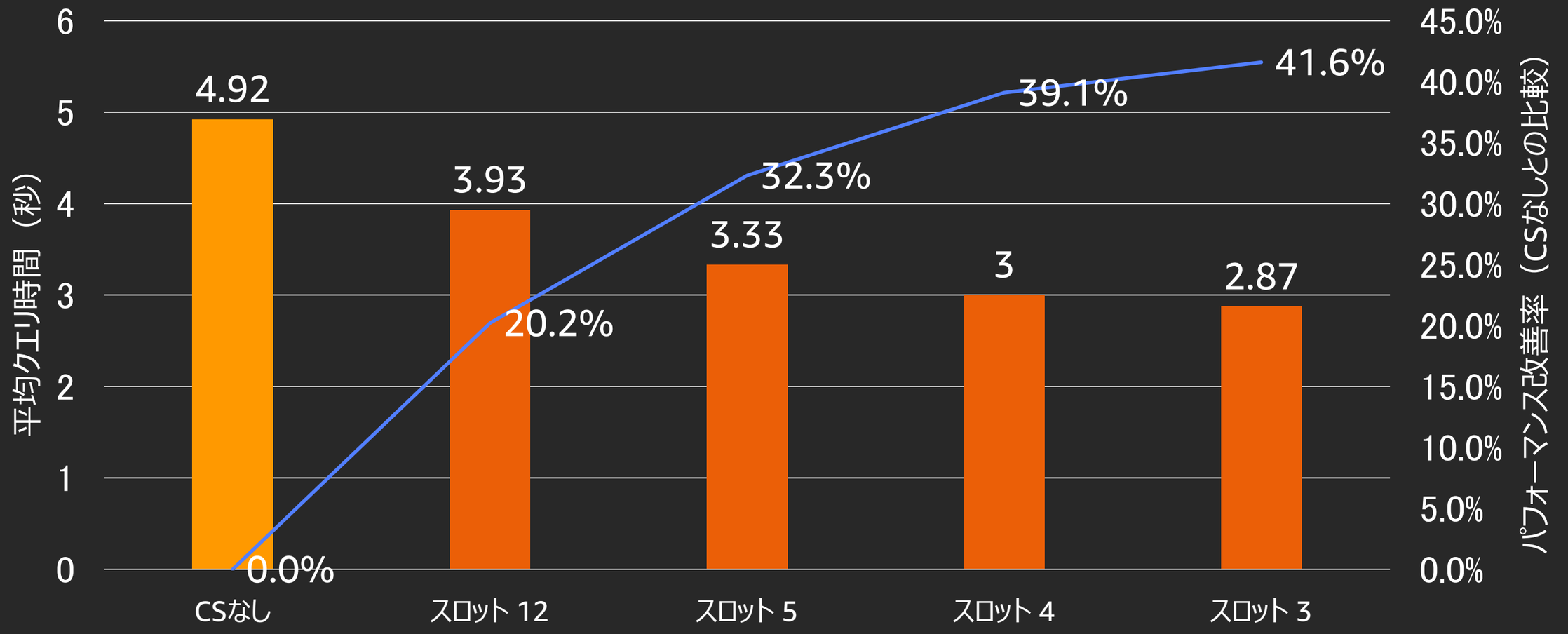
追加クラスターがアタッチされるには６０秒程度のキュー待ちが必要？

SLOT数を少なくし、クエリあたりのメモリ割当を増やすことで、頻繁にキュー待ちが発生し、CSを有効的に利用できるのでは

効果検証②

SLOT数をそれぞれ５、４、３に変更し1か月間ずつ稼働して検証

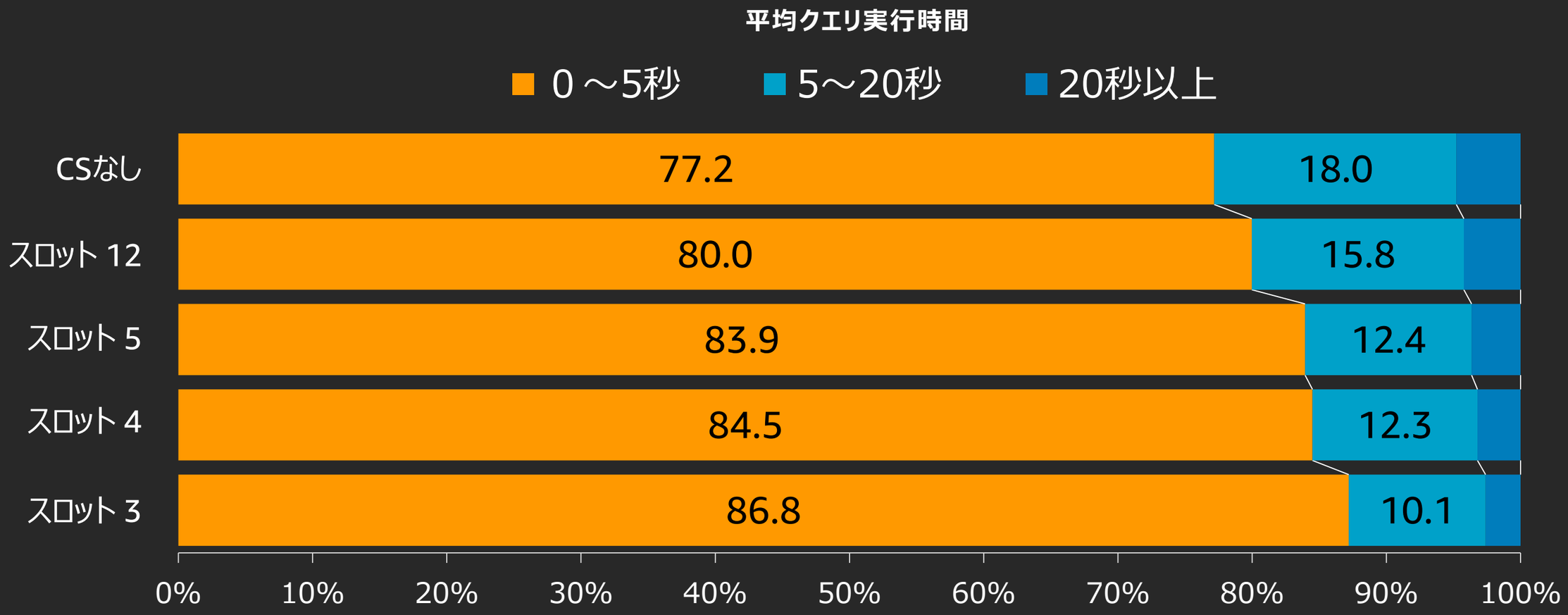
・スロット数を減らし１クエリのメモリ割当を増やすことで４０%程度同時実行パフォーマンスが改善



クエリ構成比の変化

・スロットを減らすことで、５秒以下レンジのクエリ割合が増加

・５～２０秒レンジのクエリが減少しているためパフォーマンスは向上

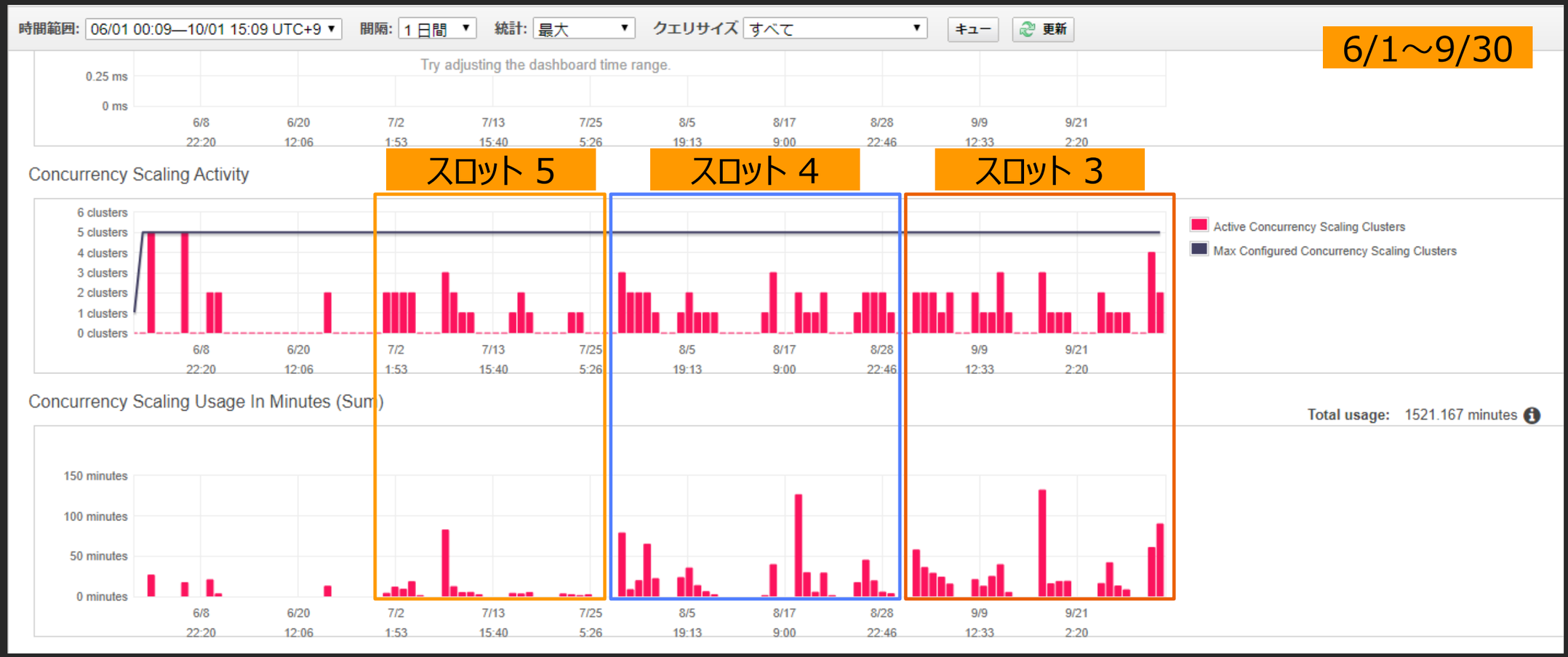


CS利用状況の確認

・スロット数を5以下に設定することで、追加クラスターはコンスタントにアタッチ

・最大で４程度　　※通常時は１～２程度

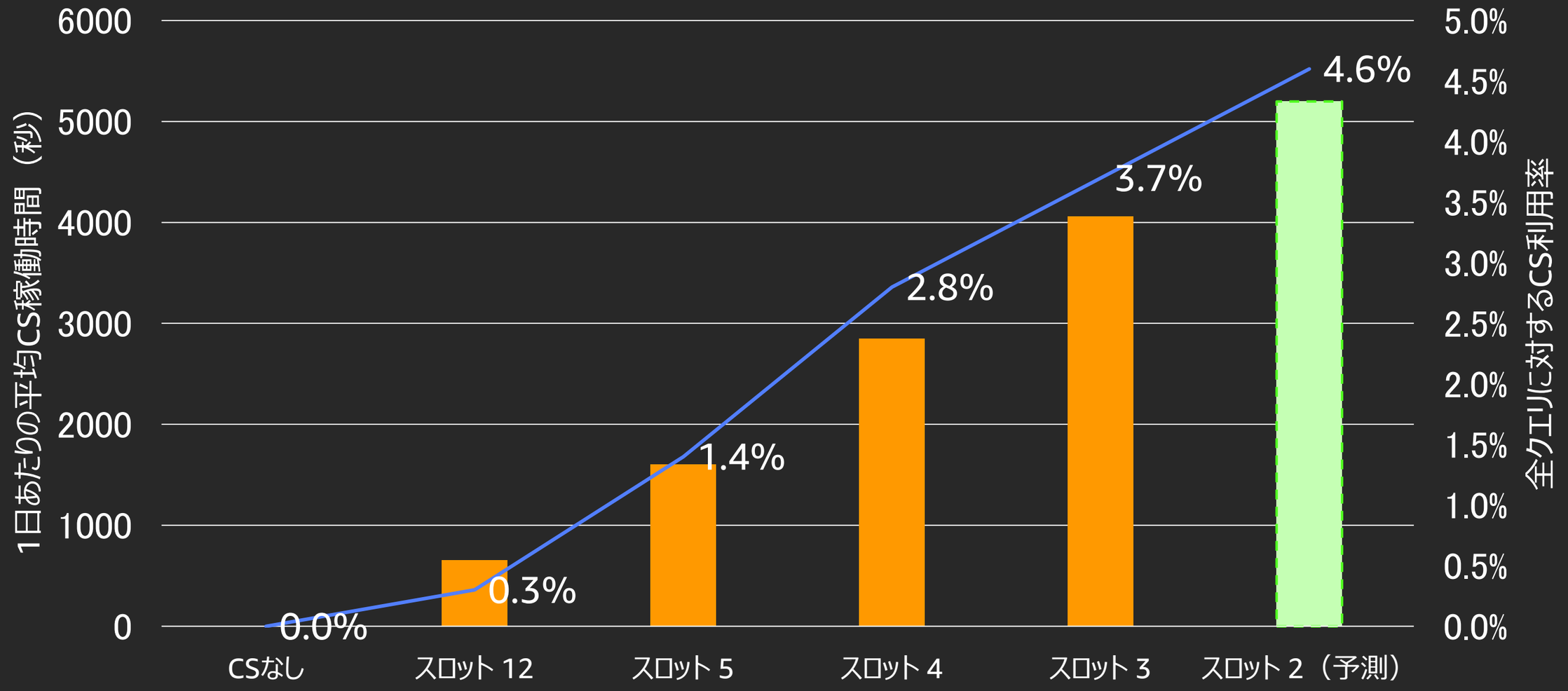
・CS利用時間はスロット数に依存（スロット数が少ないほど利用時間↑）



CSを最大限に利用する設定

・スロット3で１日あたり６７分の利用に収まり、無料で最大限活用

・スロット2では無料枠を超過する予測



結論

※当社環境

・CSを利用だけで、クエリパフォーマンスは向上

・スロット数を少なくしたほうがCSの利用率が上昇（４０％程度まで向上）

- ✓ キューイング割合が増加することで、追加クラスターがアタッチされやすくなる
- ✓ 即時利用可能になる時間が増加
- ✓ 結果としてパフォーマンスが向上

・最もCSを有効活用できる設定はスロット3

- ✓ 休日はCSが利用されることがないことを考慮しても、無料枠を最大限活用できている
- ✓ スロット２は無料クレジット枠を超過する予想

最後に

・CSを利用することで同時実行処理数が飛躍的に向上

・ロングランクエリの割合が多い場合は恩恵を感じやすい

・無謀な設定を行わない限り、無料クレジット枠で利用が可能

・CSを最大限に活用するには、自社のクエリ傾向を分析し調整を行うことが重要

Thank you!

aws SUMMIT ONLINE

アンケートへの回答を
お願いいたします。

aws SUMMIT ONLINE