

CTC Engineer's Insight # 5

K a f k aを用いた構成改善による 性能・スケーラビリティ改善

伊藤忠テクノソリューションズ株式会社

無限の未来と、幾千のテクノロジーをつなぐ。

CTC Financial Services Group



1. P J 背景と課題



2. 対策検討



3. Apache Kafkaとは



4. Kafkaを利用した新処理概要

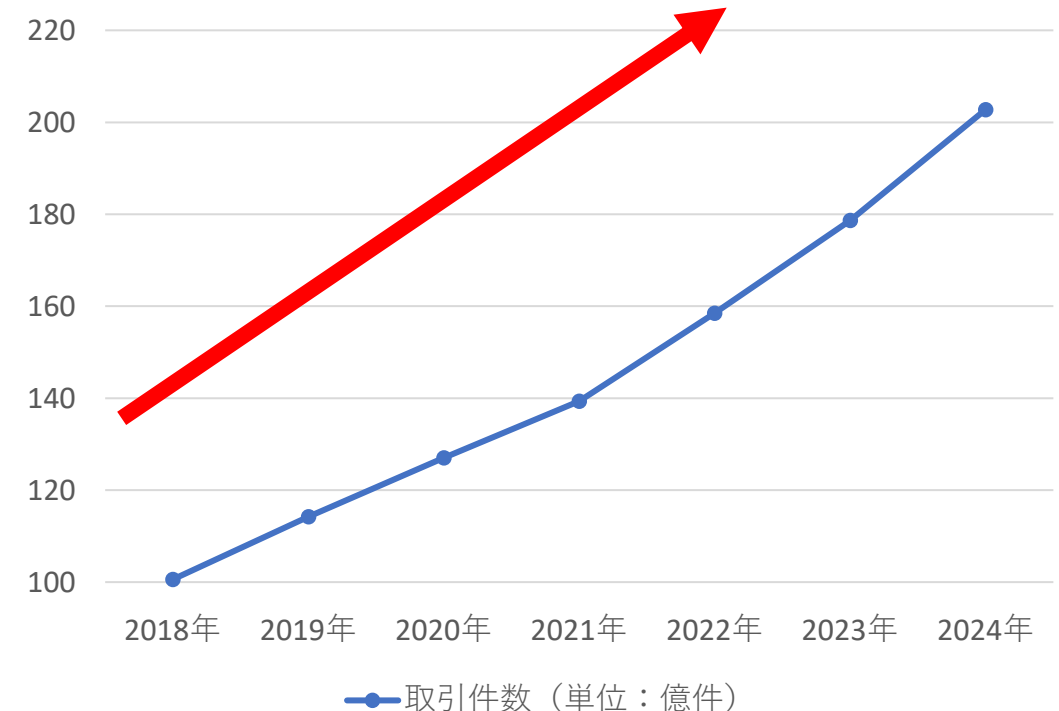


5. まとめ

1. PJ 背景と課題

- ネットショッピングの利用増加や非接触決済の普及により、クレジットカードの利用は年々増加傾向にあり、取引件数（売上件数）は6年間で2倍以上の件数増となっている。
- 取引量増加により各カード会社は1日で数百万、数千万件の取引データを処理する必要があり、処理性能が追い付かなくなっている。

クレジット年間取引件数
(単位：億件)



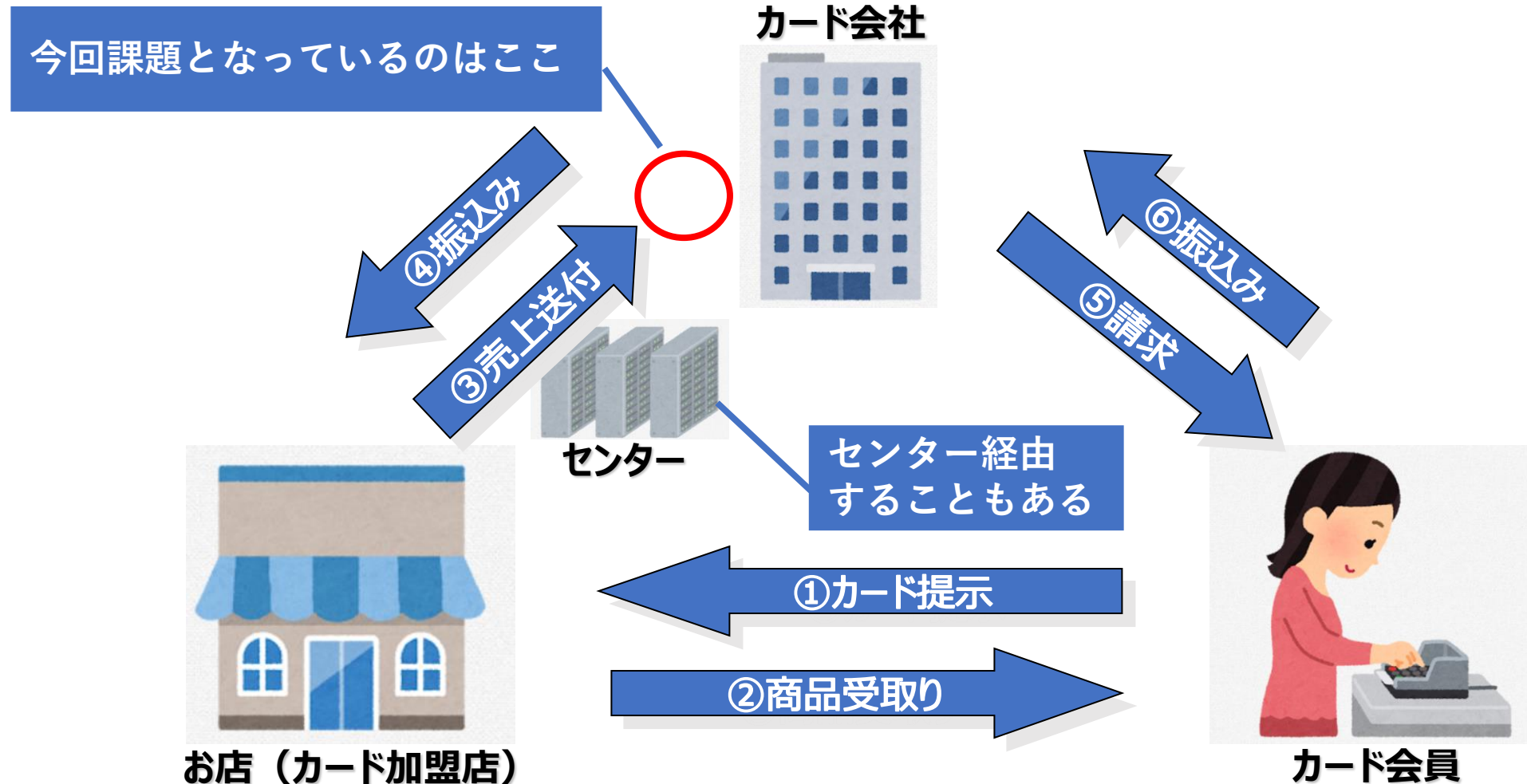
出典：日本クレジット協会 日本のクレジット統計 より
<https://www.j-credit.or.jp/information/statistics/>

1. PJ 背景と課題

- 某カード会社様でも取引件数増加により、各加盟店からの売上データに対する基幹システムへの登録処理の時間が年々増加してきており、このままいくと数年後には時限を超過する見込みであった。
- アプリ・SQLチューニングなどの小規模なパフォーマンス改善を実施しているが、この対応では限界があるため大幅なシステム改修を行い性能改善したいとのご要望があった。
- 本件について、データアクセスの最適化、不要ロジックの削除、処理構成変更等の対応を実施したが、本日は処理構成変更（Kafka導入）について、ご紹介する。

1. PJ 背景と課題

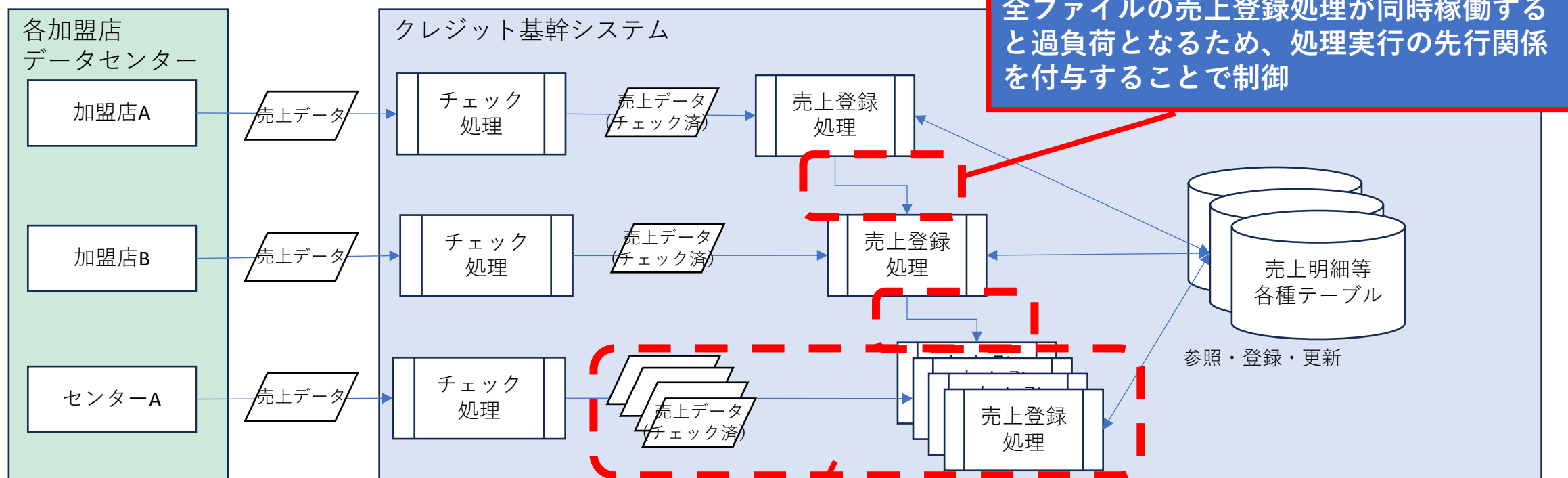
前提知識：カード利用の流れ



1. PJ 背景と課題

性能改善対象の処理概要は下図の通り、各加盟店からファイルで受領した売上データをファイルごとにチェックし、チェックが全件完了したら、売上登録を実施する。

※下図は3ファイル分のみ記載しているが、実際はもっと多くの加盟店・データセンター等から売上データを受領し、同様の仕組みで後続処理として存在



データ件数が多いファイルはチェック時にファイル分割。
売上登録処理を多重化して、処理時間短縮。

2. 対策検討

システムのあるべき姿も検討し、対策として、売上機能を別システムとして切り出し、基幹システムとは別のリソースを使用することで、性能改善を図ることを検討。
(基幹システムの全体負荷軽減や、マイクロサービス化にも繋がる)

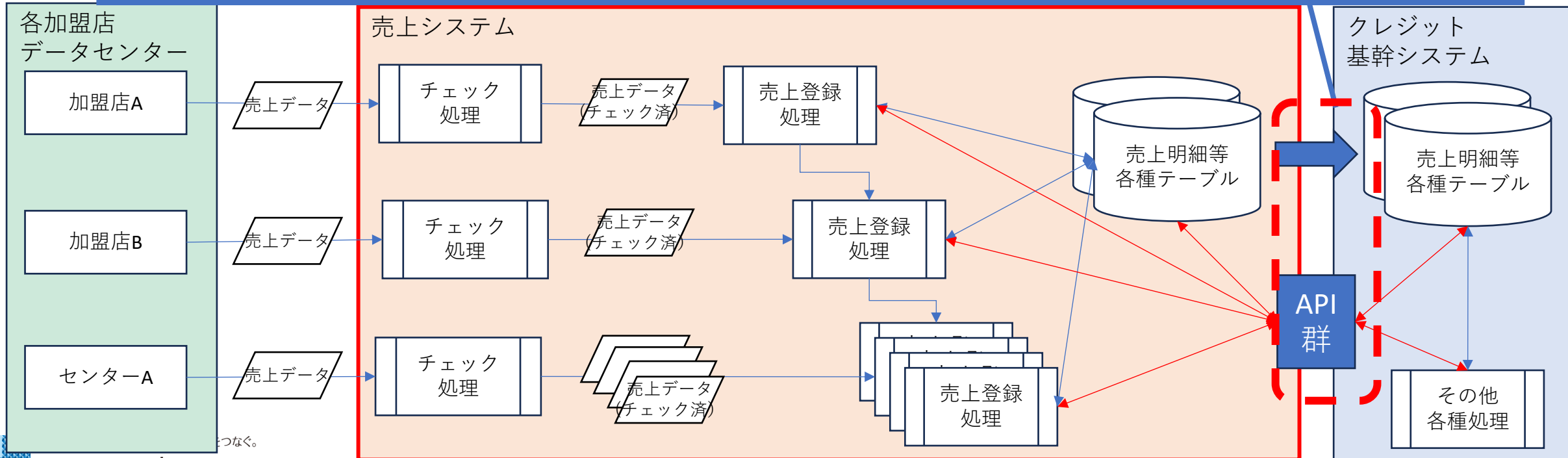
<ポイント>

各種DBテーブルについては整理の上、

①新規システム上で登録した内容を基幹システム側へ

レプリケーションや一括ロード等の高速な方法によるデータ同期を実施

②単純同期が難しいものはAPIを作成し、API経由で別システムのDBテーブルを照会・更新

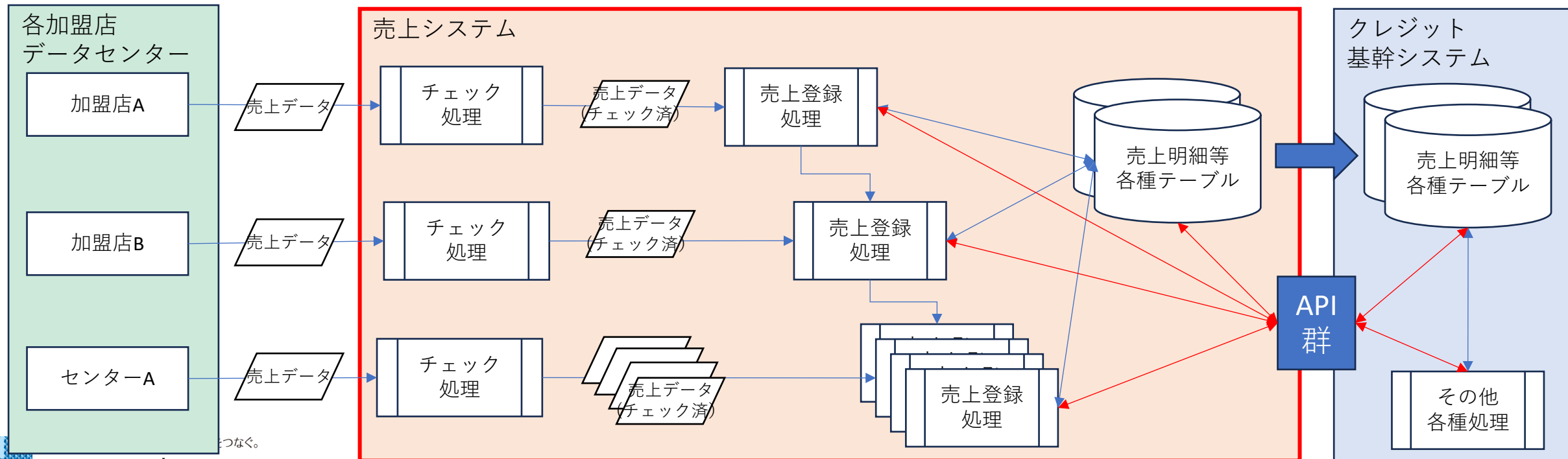


2. 対策検討

前ページの方針で検討したが、DBテーブルへのアクセスに課題が多く、断念。

- 他処理からの参照・更新等も膨大で、レプリケーションなどの単純同期で対応するにはテーブル構成の見直しから必要。
- APIでの照会・更新とするにしても影響範囲が広く、対応が必要な処理が膨大となり、レイテンシー等による処理時間増の懸念や、APIの大量開発が必要。

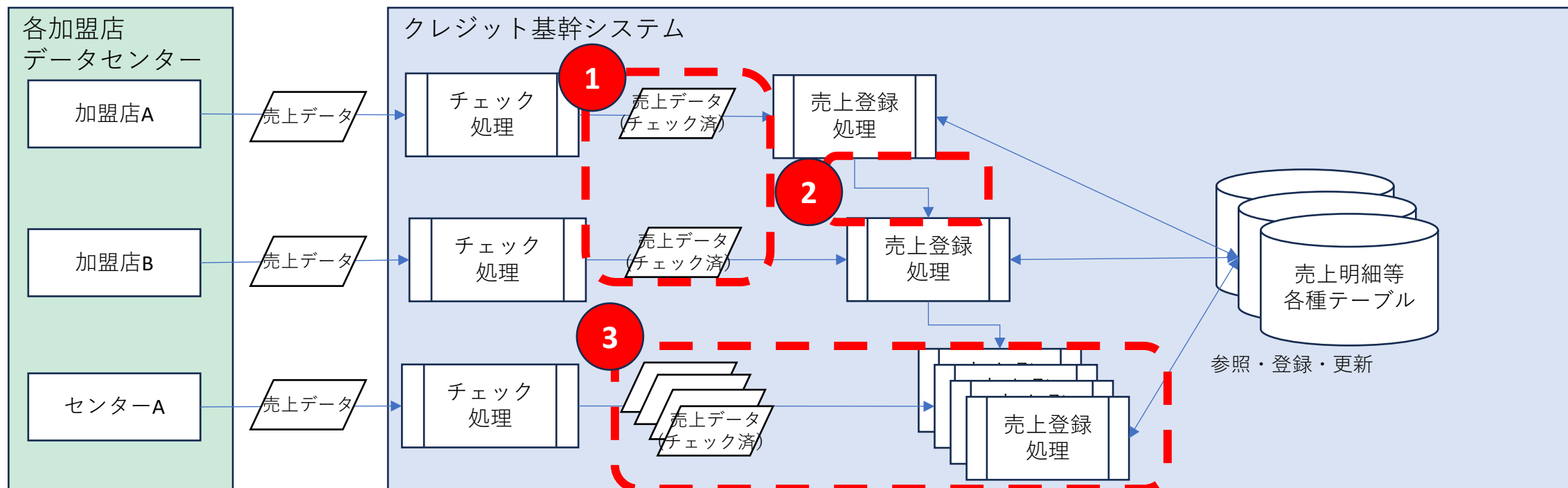
⇒コスト高、リスク高と想定される。



2. 対策検討

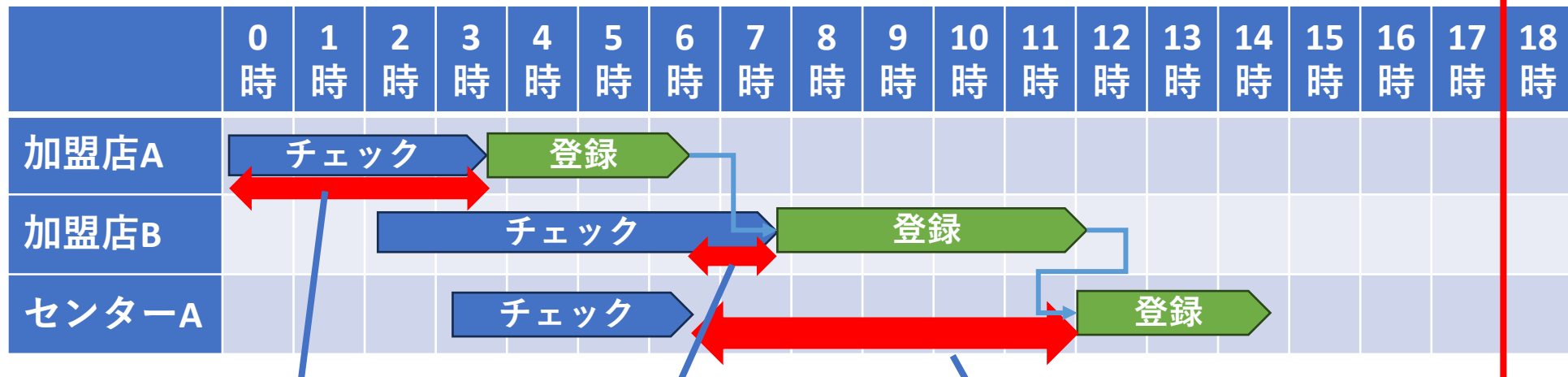
現行処理の課題を分析。以下、改善できそうなポイントあり。

- ① チェック処理が終わりきってから売上登録処理が稼働するため、チェックが完了するまで待ち時間が発生する。
- ② 前の加盟店・センター分の処理が終わってからでないと、次の加盟店・センター分が処理開始できない。
- ③ 売上登録処理を高速化するため、実際には多重化しているが、加盟店・センター毎に多重化対応を行う必要があり、また件数に変動があった場合は追加対応が必要となる。



2. 対策検討

＜補足＞ 前ページの①②について、時間軸のイメージでご説明。



①チェックが全件完了するまで、売上登録処理が稼働しない

②加盟店・センター分の処理が終わってない為、売上登録処理が稼働しない

なお、データ件数は日によって変動し、それに伴い処理時間も毎日変動するため、現状の仕組みで、待ち時間が発生しないような処理構成とするのは困難。

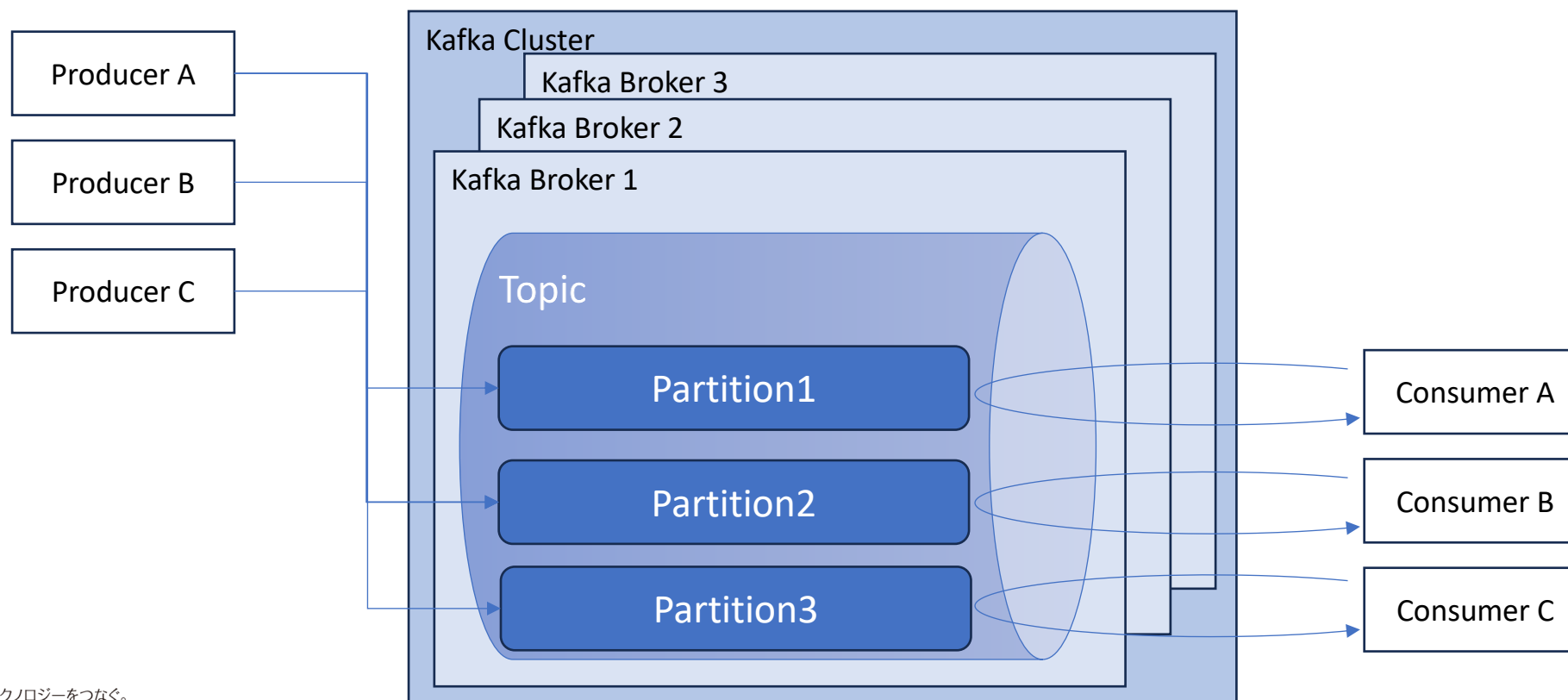
The diagram illustrates a credit-based system architecture. On the left, a green box labeled '各加盟店 データセンター' (Each Franchisee Data Center) contains three entities: '加盟店A' (Franchisee A), '加盟店B' (Franchisee B), and 'センターA' (Center A). Arrows labeled '売上データ' (Sales Data) point from each entity to a corresponding 'チェック処理' (Check Processing) block within a larger light blue box labeled 'クレジット基幹システム' (Credit Core System). A vertical dashed red line separates the 'チェック処理' blocks from the '売上登録処理' (Sales Registration Processing) blocks. Arrows labeled '売上データ (チェック済)' (Sales Data (Checked)) point from the 'チェック処理' blocks to the '売上登録処理' blocks. The '売上登録処理' blocks are arranged in a staggered fashion, with '加盟店A' and '加盟店B' having single blocks and 'センターA' having a stack of blocks. A horizontal dashed red line is positioned above the '売上登録処理' blocks. To the right of the '売上登録処理' blocks is a database cylinder labeled '売上明細等 各種テーブル' (Sales Details etc. Various Tables). Arrows point from the '売上登録処理' blocks to the database, with the text '参照・登録・更新' (Reference, Registration, Update) below the database. A large dashed red box encloses the 'クレジット基幹システム' and the database area. Above this box, the text 'Kafkaによるストリーミング処理化' (Streaming Processing by Kafka) is written.

3. Apache Kafkaとは

分散メッセージキューで、システム間のデータの受け渡しを仲介し、データを一時的に保持（キューイング）するミドルウェア。システム間の通信（処理）を非同期化することで、データ流量の急激な増加によるシステムの負荷上昇を抑制する。

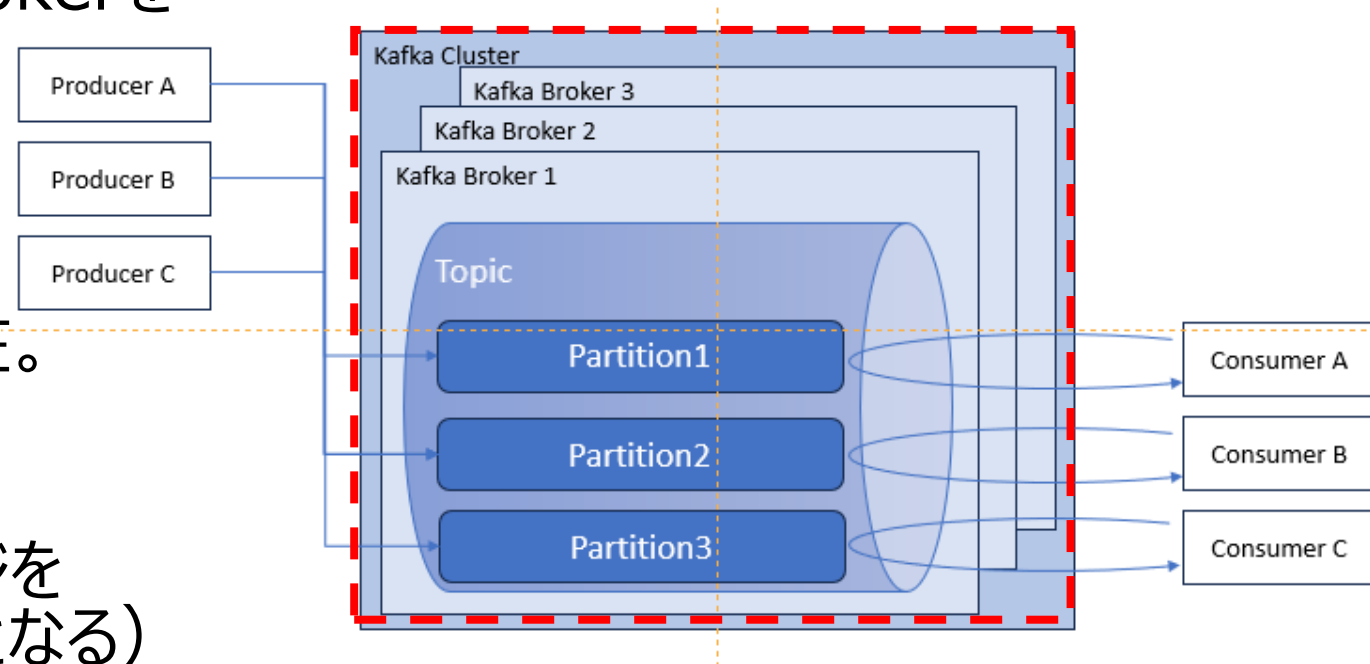
（アーキテクトではないのでKafka詳細仕様に関するご質問はご容赦ください）

<Kafka概略図>



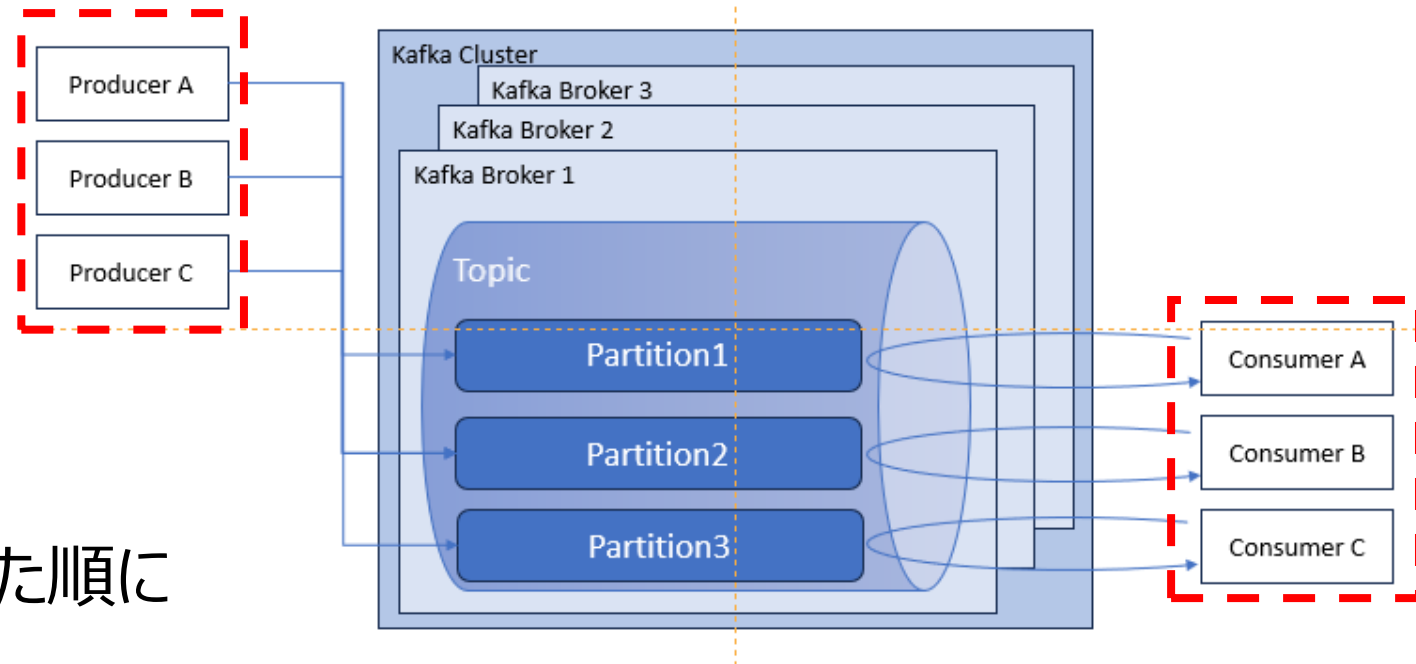
3. Apache Kafkaとは

- Kafka Broker
Kafkaが稼働するサーバ
- Kafka Cluster
可用性/信頼性向上のためにBrokerを複数台で構成したもの
- Topic
データ（メッセージ）の保存先。
データベースのテーブルに近い存在。
- Partition
Topicを複数に分割してメッセージを分散させる（並列度のファクターとなる）



3. Apache Kafkaとは

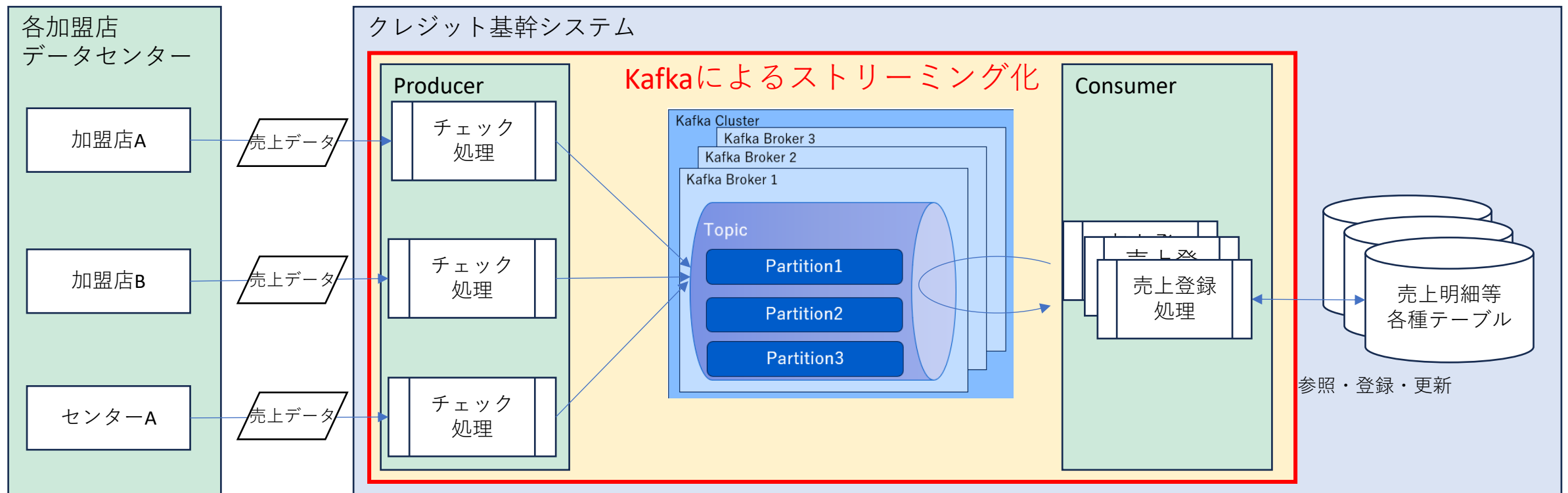
- **Producer**
メッセージを作成し、Brokerの特定のTopicを指定してメッセージを送信するアプリケーション。（Producer数とPartition数は関連性なし）
- **Consumer**
Brokerの特定のTopicを指定してメッセージを取得し、処理を行うアプリケーション。基本的にはメッセージを一定間隔で取得しつづける常駐型処理となる。
PartitionとConsumerは1:1またはN:1の関係となる。
これによりPartitionに登録された順にメッセージが取得される。



4. Kafkaを利用した新処理概要

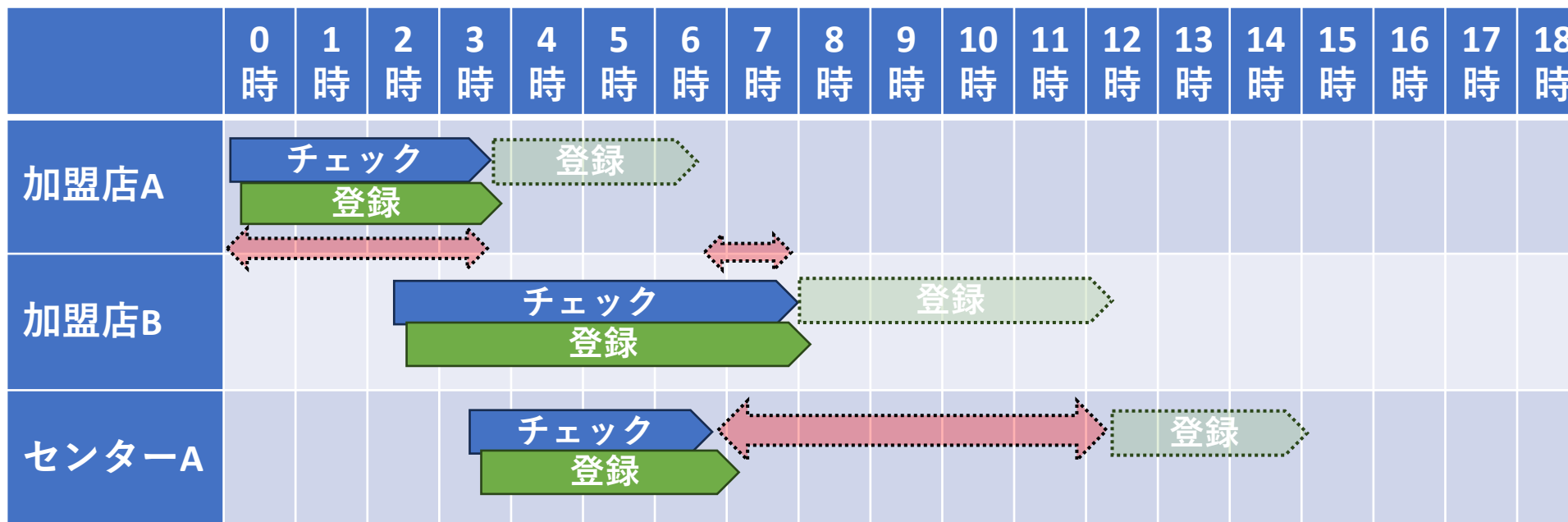
Kafka導入により以下の構成にて構築。

- ・consumerとパーティションを増やせば多重度を変更でき、スケールアウトしやすい。
- ・加盟店・センター毎に多重化対応を行う必要がなく、今後の保守開発工数削減。
- ・加盟店・センターの日単位での件数変動も、リソースの無駄なく処理可能。

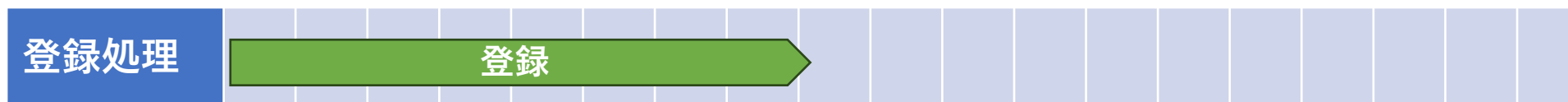


4. Kafkaを利用した新処理概要

待ち時間が発生していた⇔部分の無駄な待ち時間が解消。
全体としての処理時間が大幅削減。



なお、実際は売上登録処理は、以下のように加盟店・センターにかかわらず、共通の常駐処理として稼働。



5. まとめ

- あるべき姿を意識して、解決策を考える
- とはいえ、現実的ではない案に固執せず、コストが高い、リスクが高い等の場合は、早い段階で軌道修正する。
- 課題がどこにあるのかきちんと分析することが大切。
- アプリケーションチューニングだけでは限界がある。新アーキテクチャの導入による解決策も検討する。

無限の未来と、
幾千のテクノロジーをつなぐ。

CTC Financial Services Group

