メッセージキュー「Pulsar」のご紹介
～ヤフーの導入事例を交えて～

ヤフー株式会社 システム統括本部
坂本 雅宏

2017/05/27

自己紹介

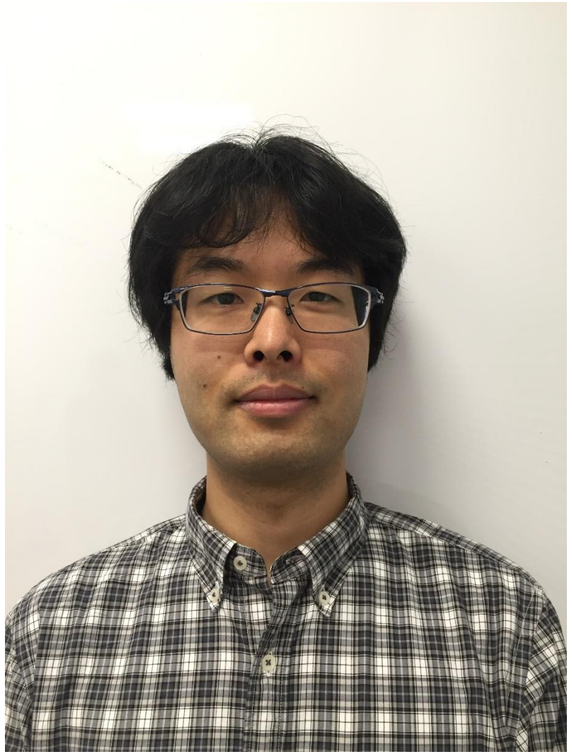
坂本 雅宏

経歴：

- 2013/04　　ヤフー入社
- 2013/07~　ユーザの属性情報を扱うシステムを担当
- 2015/10~　社内向けのWebAPI実行環境を担当
- 2016/04~　メッセージキュー「Pulsar」を担当

扱う事が多い言語：日本語、Java、PHP、Node.js

扱うのに困難が伴う言語：英語



Pulsar

Yahoo! Inc. で開発された新しいPub-Subメッセージングシステム（メッセージキュー）

- 2014秋　　Yahoo! Inc. が開発を開始
- 2015春　　Yahoo! Inc. で利用開始
- 2016/04　　Yahoo! JAPANが開発に参加
- 2016/09　　OSSとして公開
- 2017/01　　Yahoo! JAPANで利用開始

アジェンダ

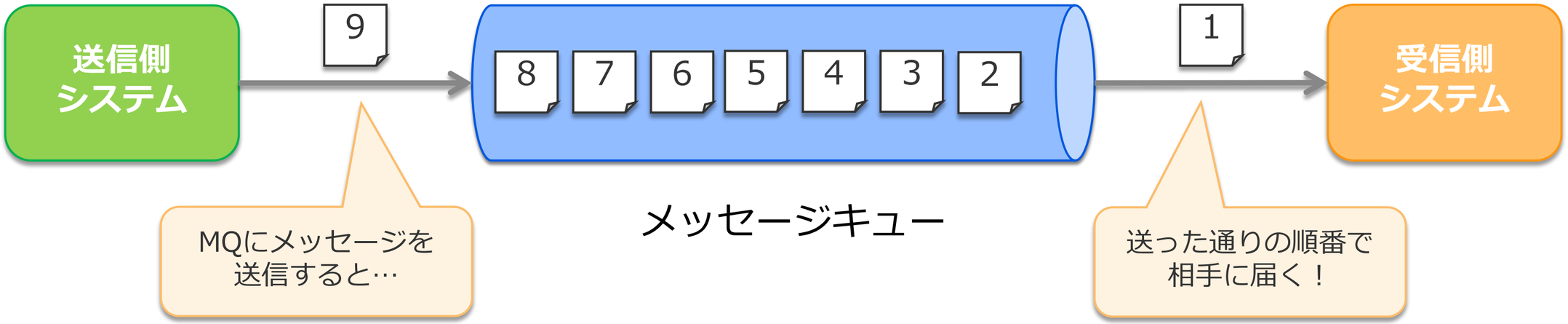
1. メッセージキューについて
2. Pulsar導入の背景
3. Pulsarの概要
4. インターフェース
5. アーキテクチャ
6. 利用事例
7. まとめ

メッセージキューについて

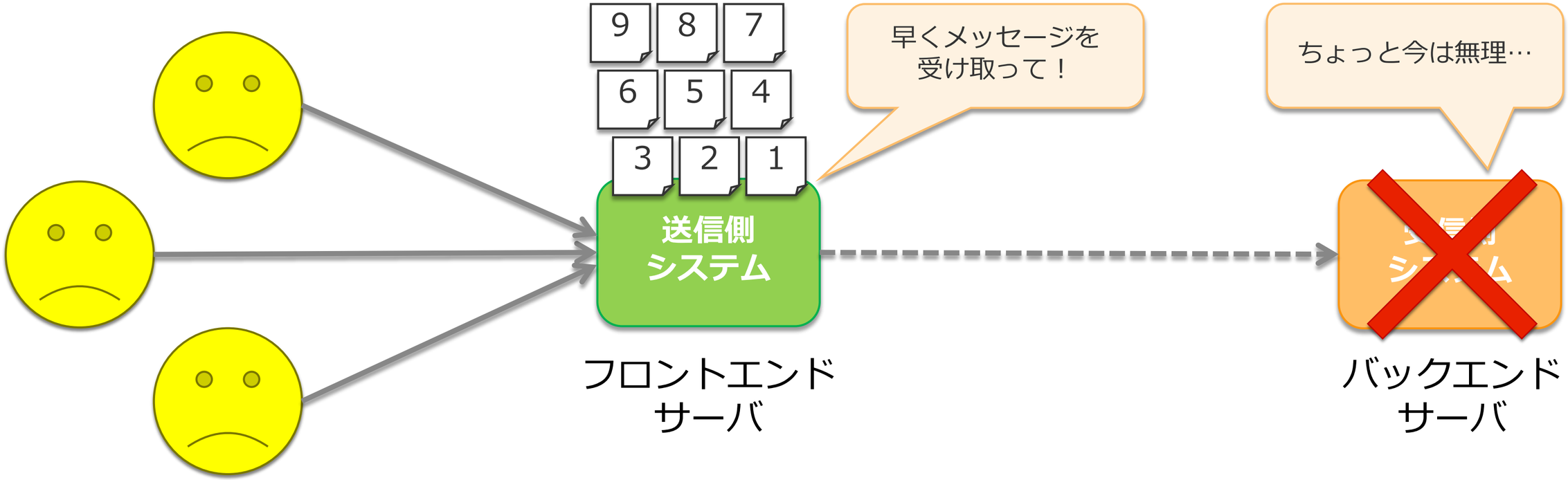
メッセージキュー（MQ）とは？

システム間のメッセージのやり取りに使用されるソフトウェアの総称

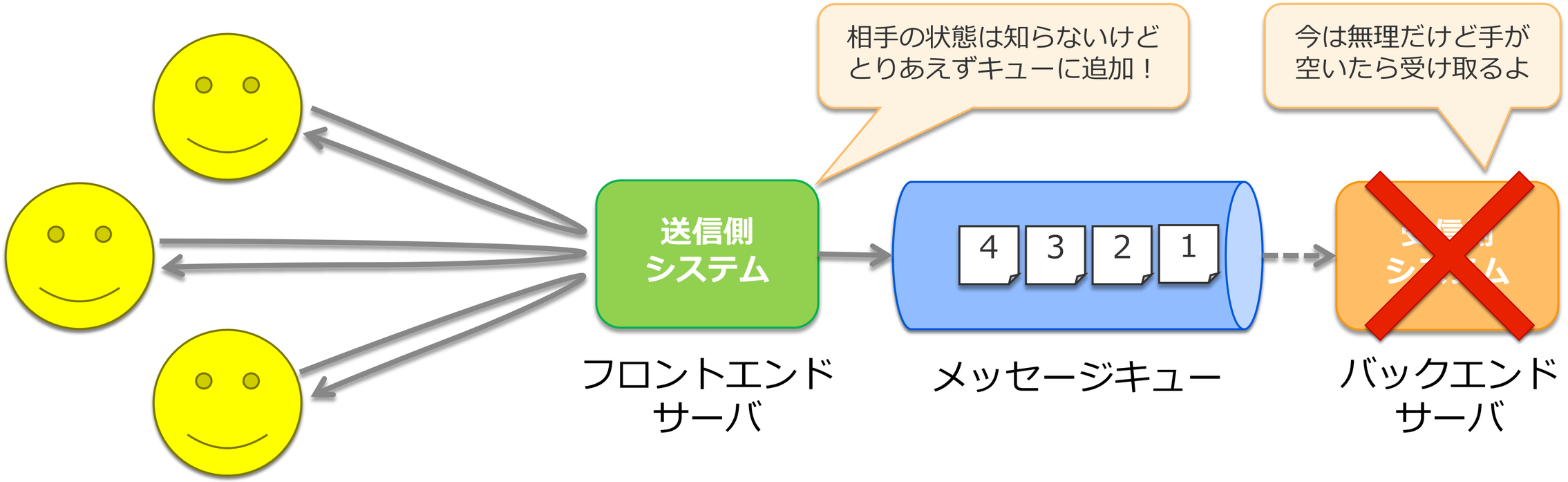
- メッセージ … 他のシステムへの通知情報、処理して欲しいデータ、ログなど
- キュー … 先に入れたものから順に出てくるデータ構造
  - メッセージの送信 = キューの最後尾にメッセージを追加する
  - メッセージの受信 = キューの先頭からメッセージを取り出す



メッセージキューを使わなかったら…



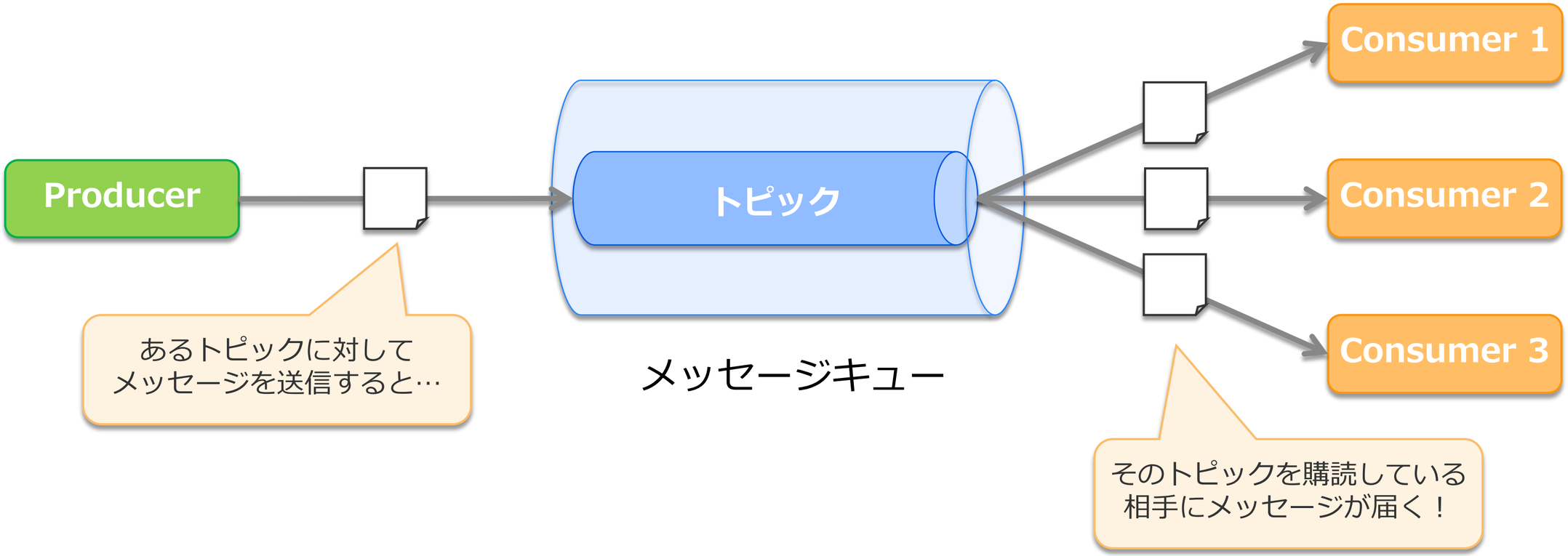
メッセージキューを使えば…



Pub-Subメッセージングとは？

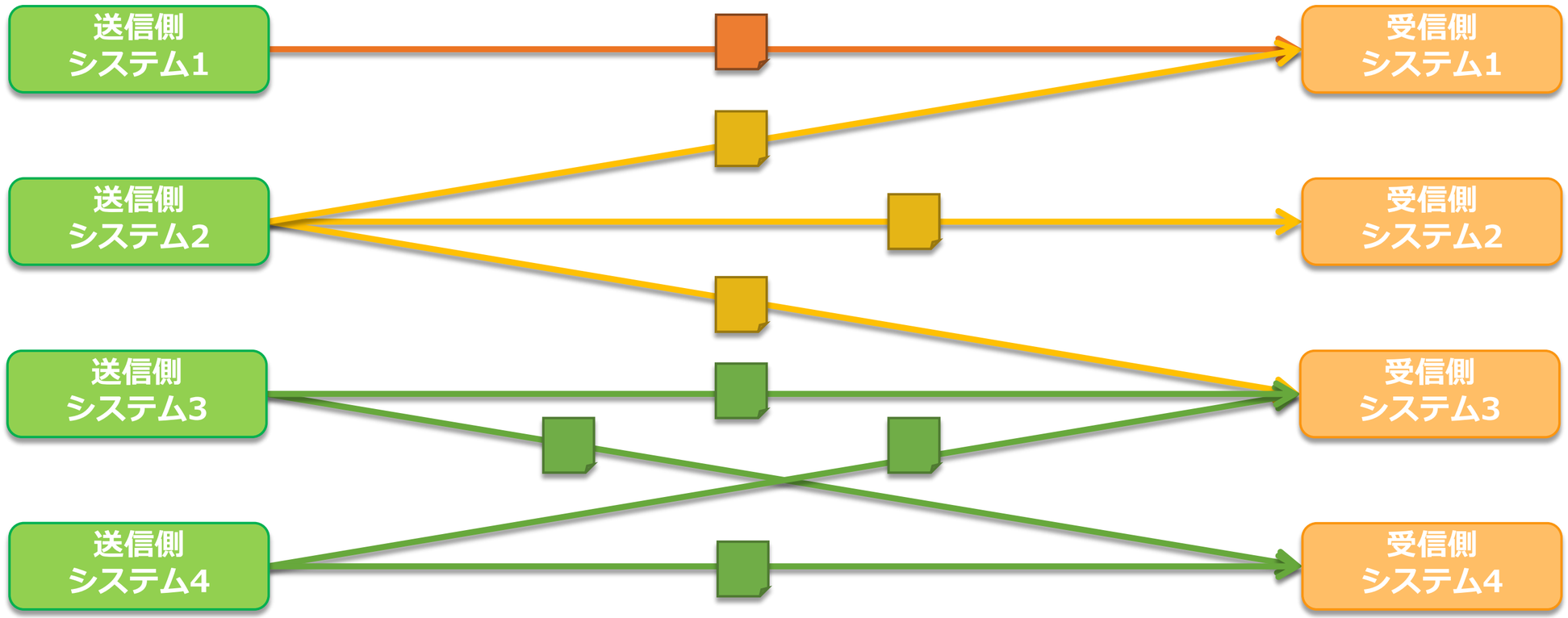
トピックと呼ばれる宛先に対してメッセージの送受信が行われる方式

- メッセージの送信側 ＝ Producer / Publisher / 出版者
- メッセージの受信側 ＝ Consumer / Subscriber / 購読者



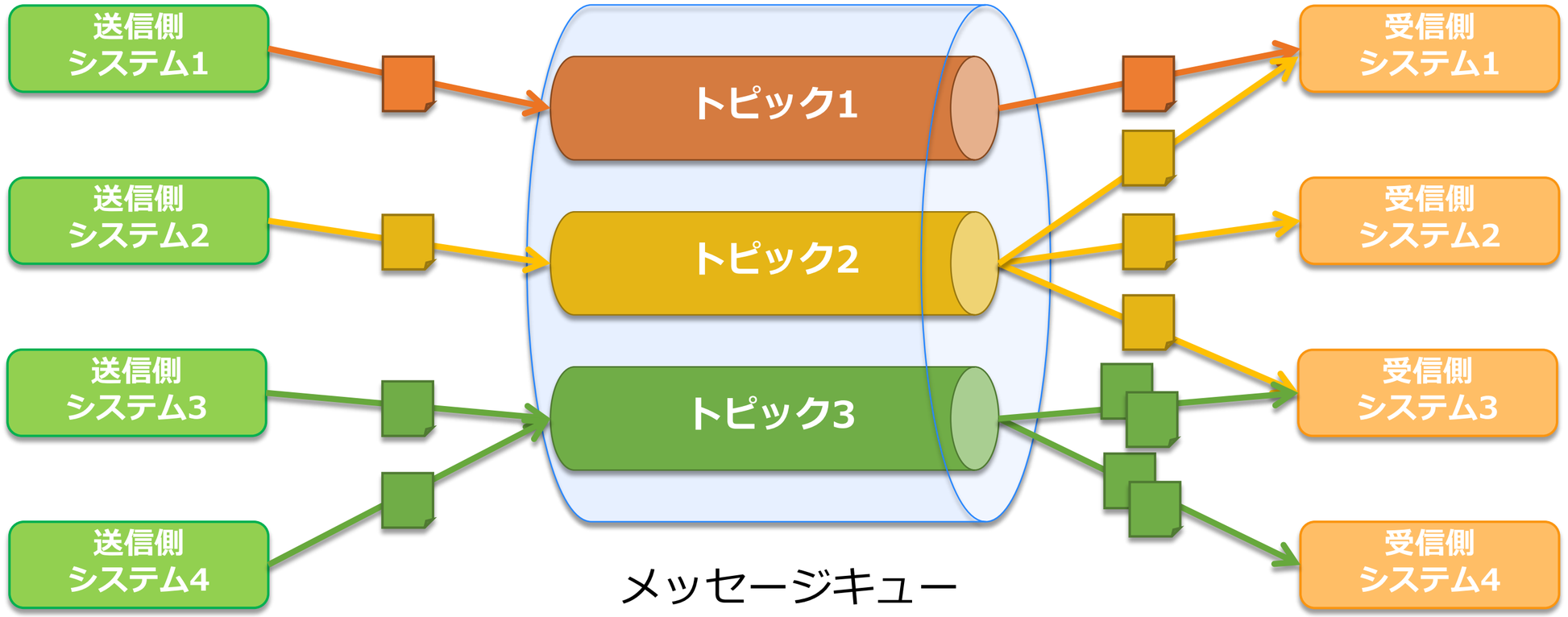
Pub-Subメッセージングをしなかったら…

1対1の通信なら問題ないが、**1対多**や**多対多**になるとても複雑で大変



Pub-Subメッセージングをすれば…

送信側も受信側もお互いを意識する必要がない → **シンプル**かつ**スケーラブル**に



代表的なメッセージキュー

- **Kafka**
  - 2011年にLinkedInから公開
  - 大量のデータを高速に処理する事を目指し設計
  - ストリームデータの処理によく使用される
- **RabbitMQ**
  - 2007年にRabbit Technologiesから公開
  - 通信プロトコルにAMQPを採用
  - どちらかと言えばパフォーマンスよりも信頼性を重視
- **ActiveMQ**
  - 2004年から使用されている歴史あるプロダクト
  - 様々なプロトコルをサポートし、多数の言語から利用可能なのが強み

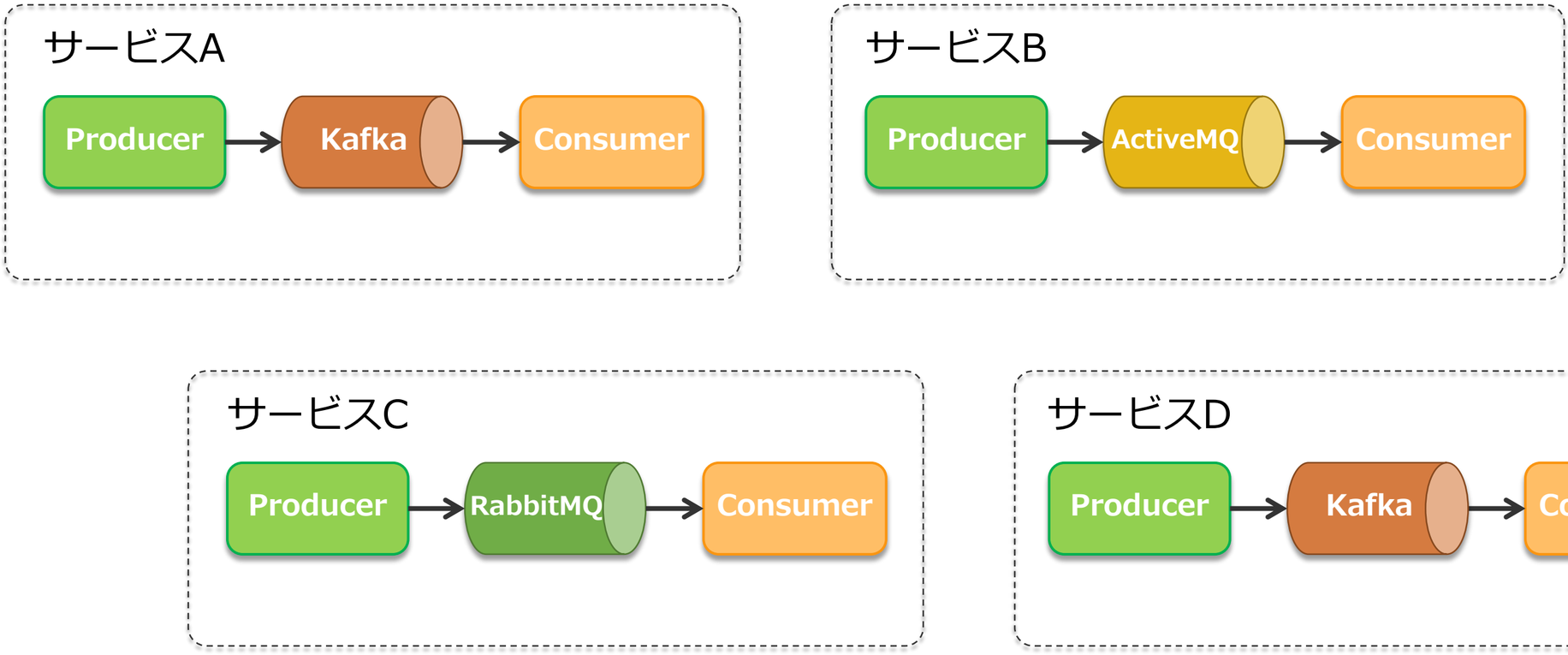
Pulsar導入の背景

課題1 – 膨大な利用者とメッセージ数

- ヤフーでは**100程度**のサービスを提供しており、それぞれの利用者数も膨大
- 例えばトップページ（PC）の1日あたりの利用者数は…
  - PV（Webページの参照回数）：**2億以上**
  - UB（ユニークなWebブラウザ数）：**3千万以上**
- それを支えるMQも大量のメッセージを高速に処理できる事が求められる
- 利用者数増大の可能性もあるためスケーラブルである事も必須

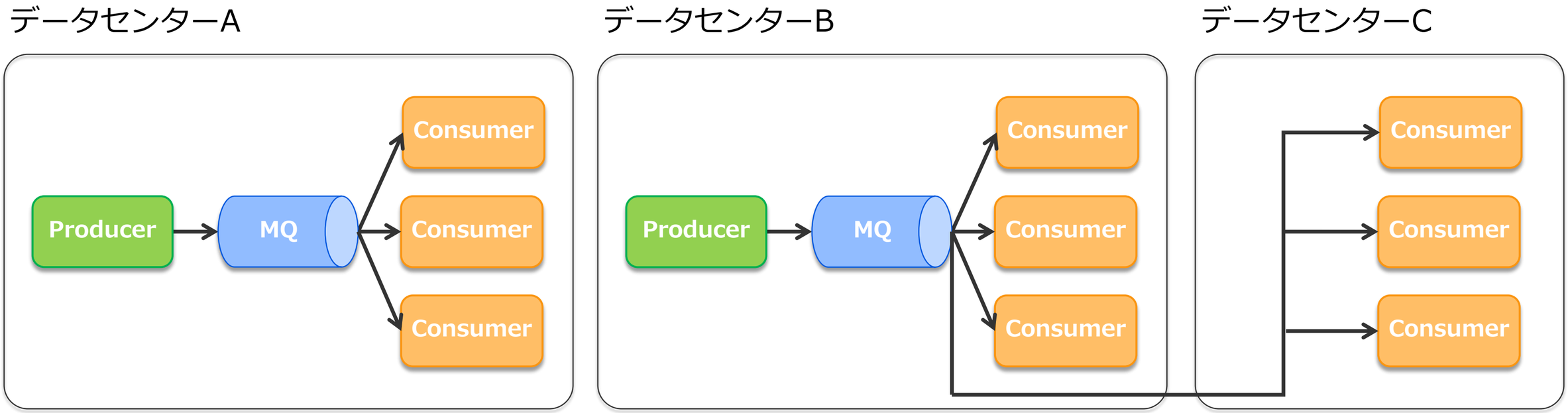
課題2 - 複数のサービスでMQが乱立

- 設備・運用コストの増大
- ナレッジの分散
- サービス開発への集中を阻害



課題3 – 地理的に離れた複数のデータセンター

- ヤフーは地理的に離れた複数のデータセンターでサービスを展開
- 全てのデータセンターのシステムがメッセージを受けて同じ処理を実行したい場合：
  - データセンターごとにMQを用意？ → 運用コストや整合性の問題
  - 特定のデータセンターのMQに全てのConsumerが接続？ → 効率が悪い

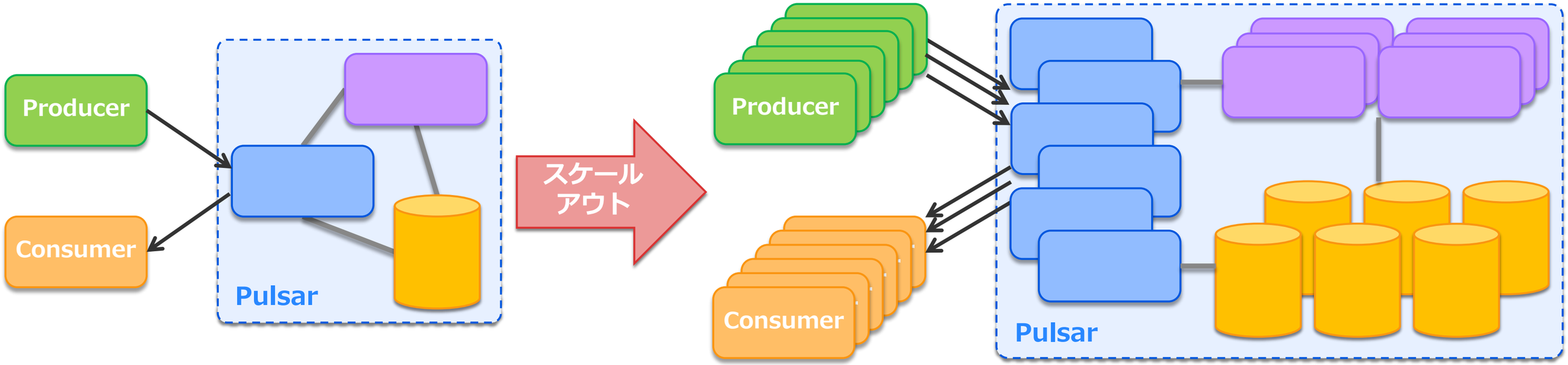


Pulsarの特徴

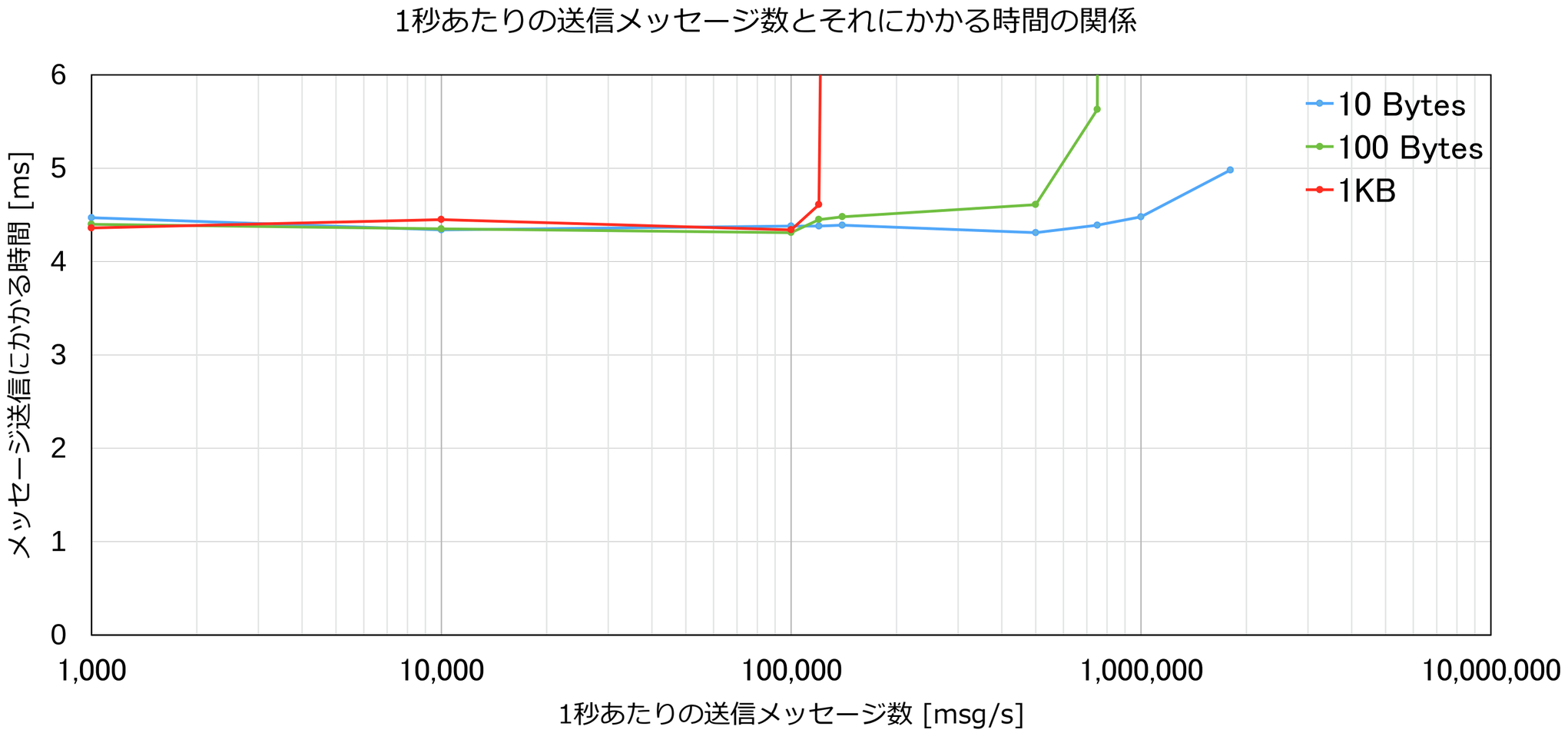
特徴1 – 圧倒的な高速性とスケーラビリティ

大量のメッセージを高速に送受信可能

- Yahoo! Inc. での実績
  - トピック数：**140万**
  - メッセージ数：**1,000億 [msg/day]**
  - メッセージの送信に要する時間：**5 [ms]**
- サーバの台数を増やせばその分だけキャパシティを増やせる → スケーラブル



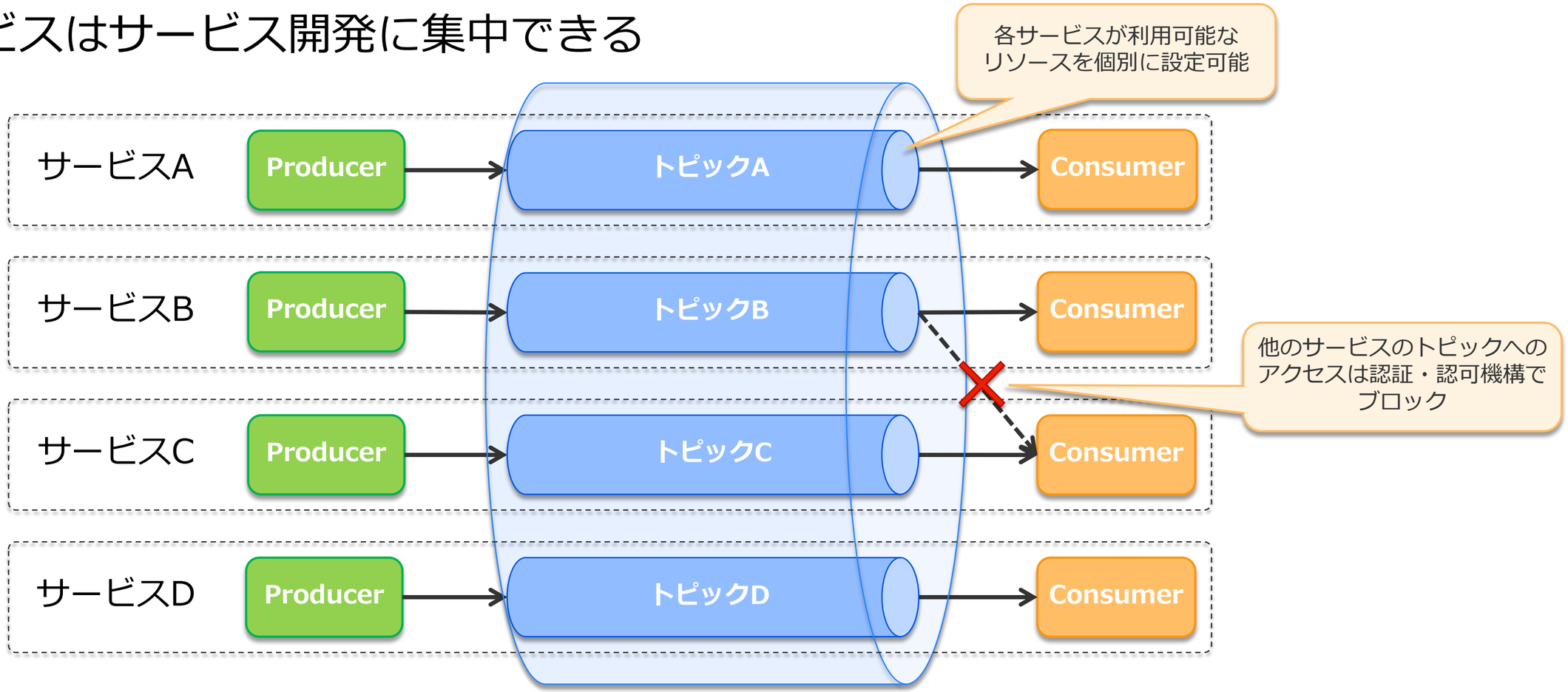
パフォーマンス



特徴2 - マルチテナント

複数のサービスが1つのPulsarインスタンスを共用可能

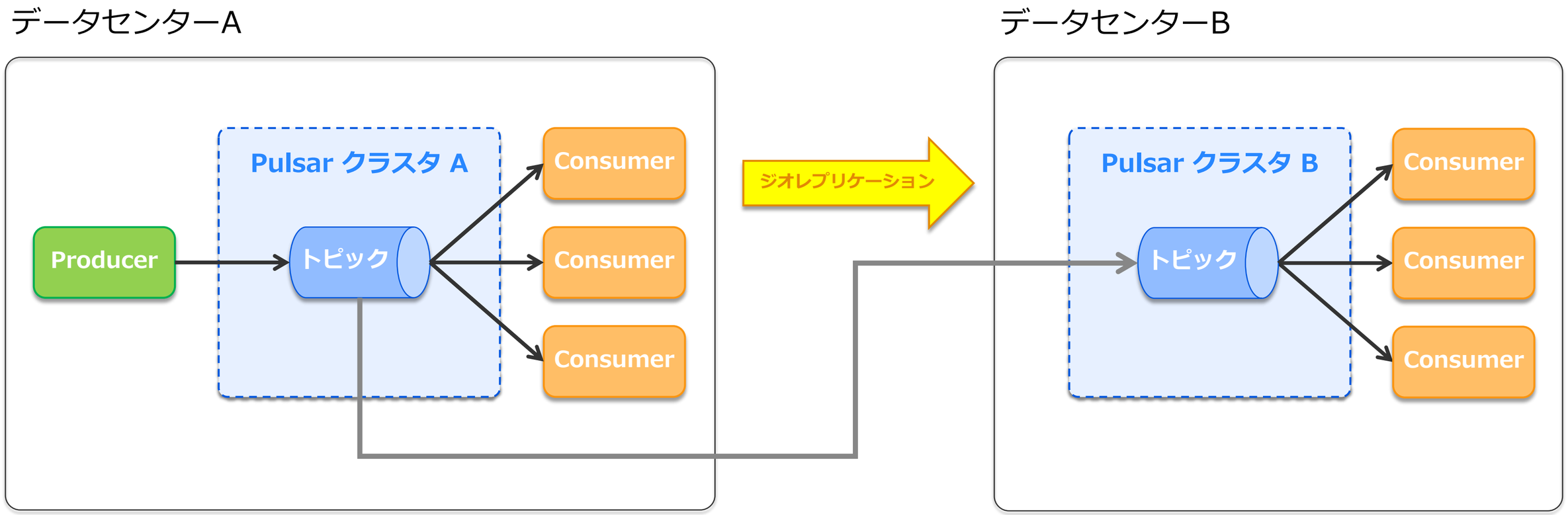
- ヤフーでは全社で共有するインスタンスを専用のチームが提供・運用
- 各サービスはサービス開発に集中できる



特徴3 – ジオレプリケーション

あるデータセンターのトピックに送信したメッセージを他のDCに複製・配信可能

- Producerは自分と同じデータセンターのPulsarにメッセージを送ればいい
- Consumerは自分と同じデータセンターのPulsarからメッセージを受け取れる



インターフェース

Pulsarのインターフェース

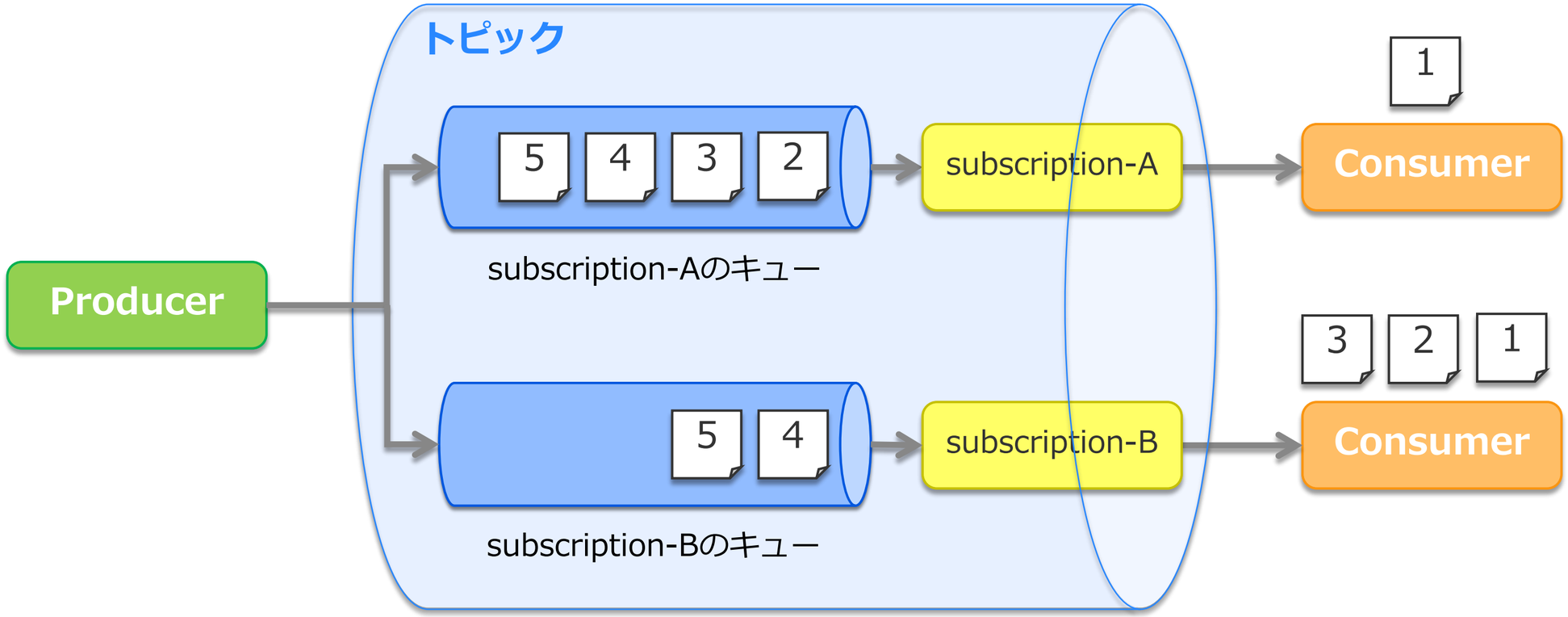
- クライアントライブラリは**Java**と**C++**を提供
- 他の言語からも**WebSocket API**を利用可能
- Producer/Consumerは次のようなトピックURIを指定してPulsarに接続

- さらにConsumerは**サブスクリプション**の名前を指定する必要がある
- 用途に応じて3種類の**サブスクリプションモード**（メッセージの配信方式）を選択可能

サブスクリプションとは？

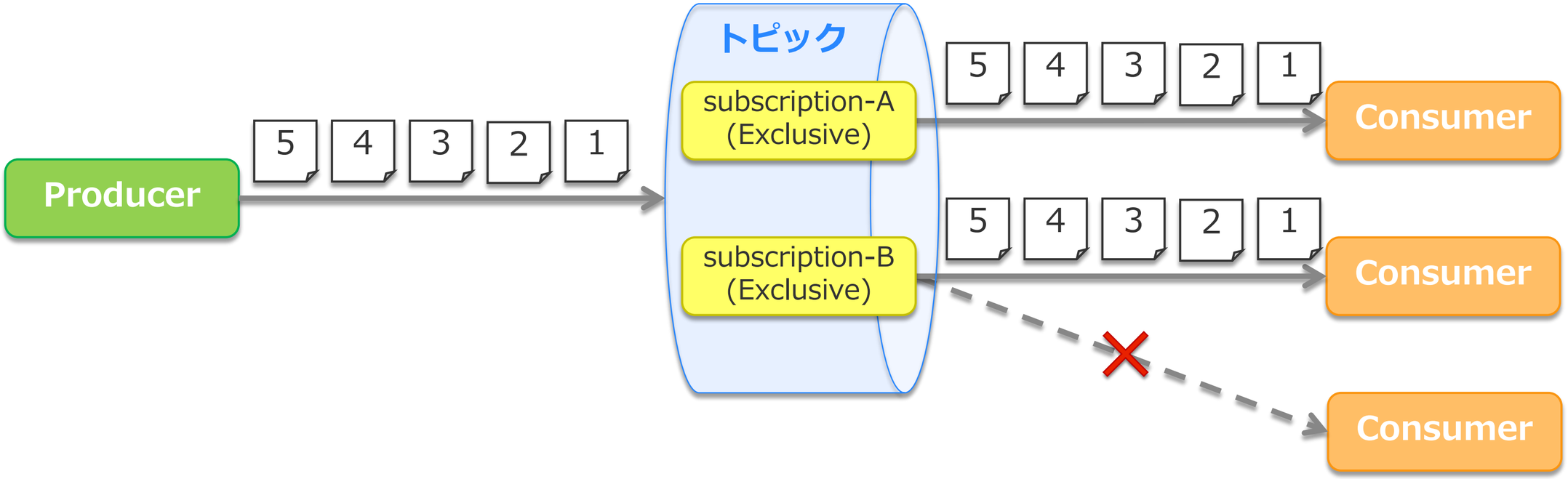
Consumerがトピックの購読を開始した時に作成される購読の管理単位

- メッセージのキューはサブスクリプションごとに用意される
- Consumerとの接続が切れても削除されず、再接続までの間メッセージを溜め続ける



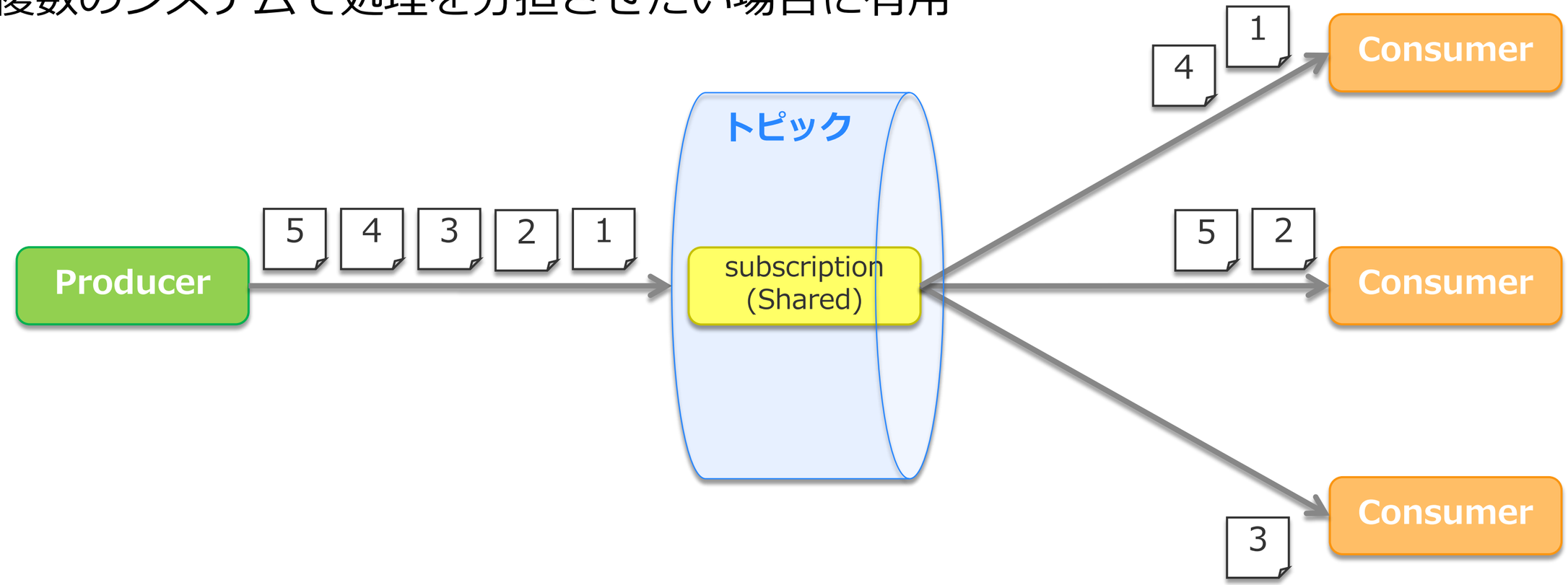
サブスクリプションモード1 - Exclusive

- 1つのサブスクリプションには1つのConsumerだけが接続可能
- 既にConsumerが存在するサブスクリプションに接続を試みるとエラーになる



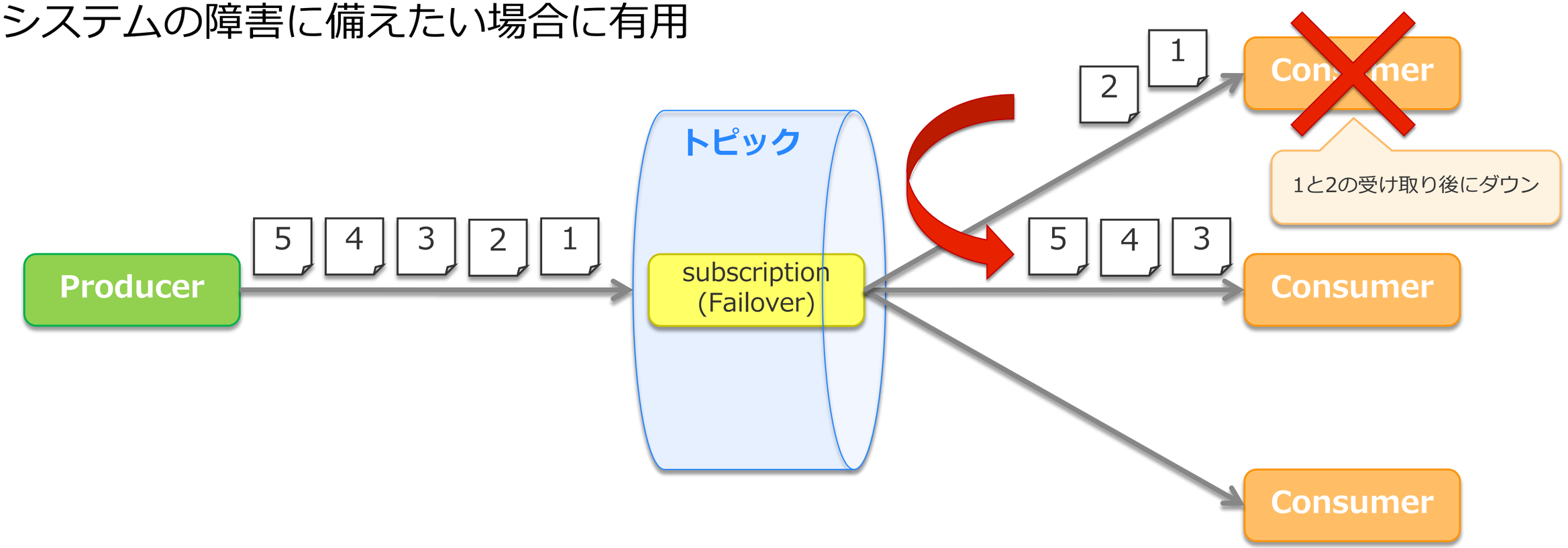
サブスクリプションモード2 - Shared

- 1つのサブスクリプションに複数のConsumerが接続可能
- メッセージは各Consumerにラウンドロビンで配信される
- 複数のシステムで処理を分担させたい場合に有用



サブスクリプションモード3 - Failover

- 複数のConsumerが接続可能だが、メッセージを受け取るのはその内1つだけ
- そのConsumerがダウンすると別のConsumerがメッセージを受け取るようになる
- システムの障害に備えたい場合に有用



Javaのサンプルコード - Producer

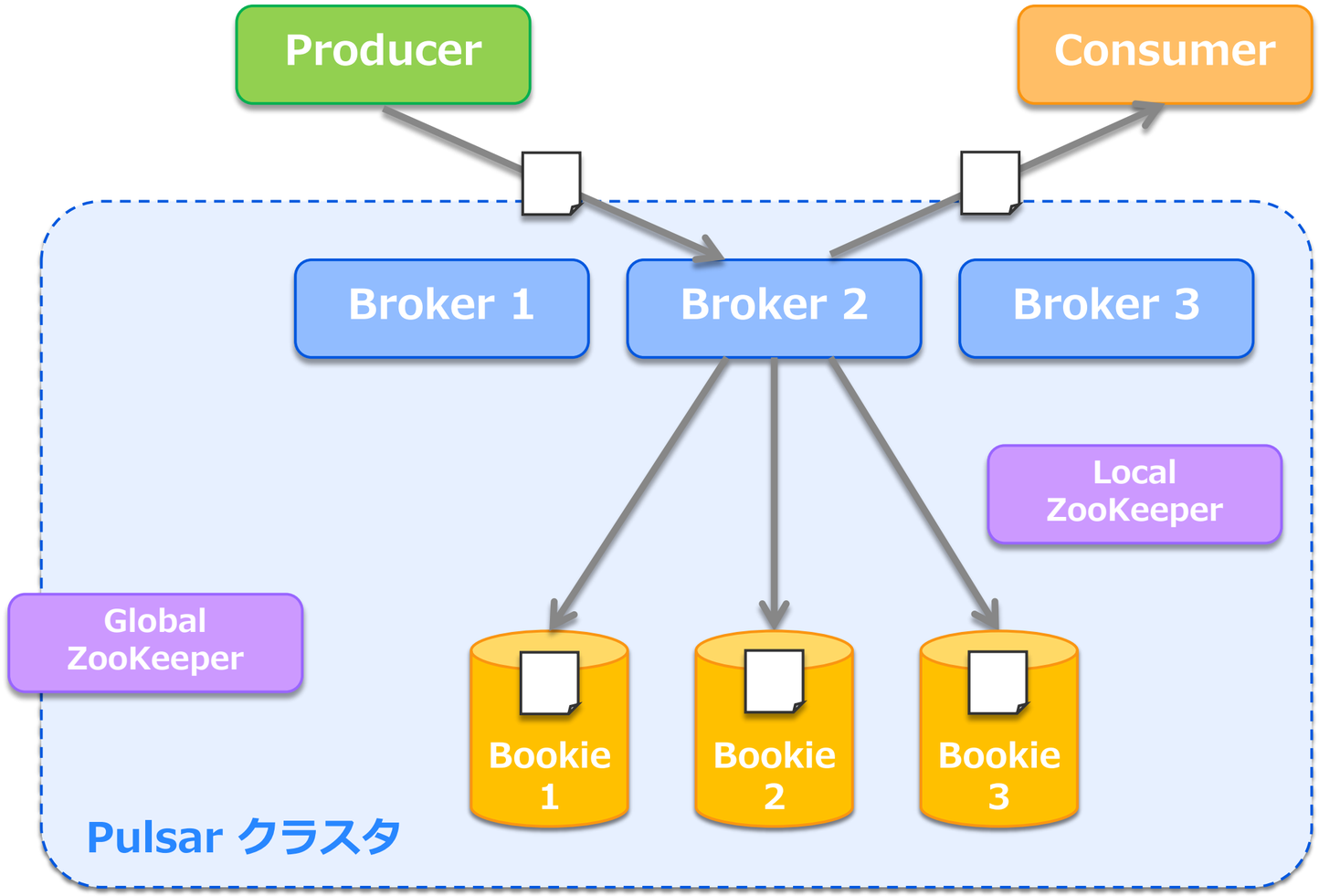
```
// PulsarのURLを指定してクライアントを作成
PulsarClient client = PulsarClient.create(
    "pulsar://broker.usw.example.com:6650");

// トピックを指定してProducerを作成
Producer producer = client.createProducer(
    "persistent://my-property/us-west/my-namespace/my-topic");

// メッセージを送信
producer.send("my-message".getBytes());
```

Javaのサンプルコード - Consumer

```
PulsarClient client = PulsarClient.create(
    "pulsar://broker.usw.example.com:6650");

// トピックとサブスクリプションを指定してConsumerを作成
Consumer consumer = client.subscribe(
    "persistent://my-property/us-west/my-namespace/my-topic",
    "my-subscription-name");

// メッセージを受信して画面に表示
Message msg = consumer.receive();
System.out.println(new String(msg.getData()));

// メッセージに対するACKを返信（キューからメッセージを削除）
consumer.acknowledge(msg);
```

アーキテクチャ

システム構成図



Broker

- クライアントとのメッセージのやり取りを担当
- 状態を持たないため増設が容易

Bookie

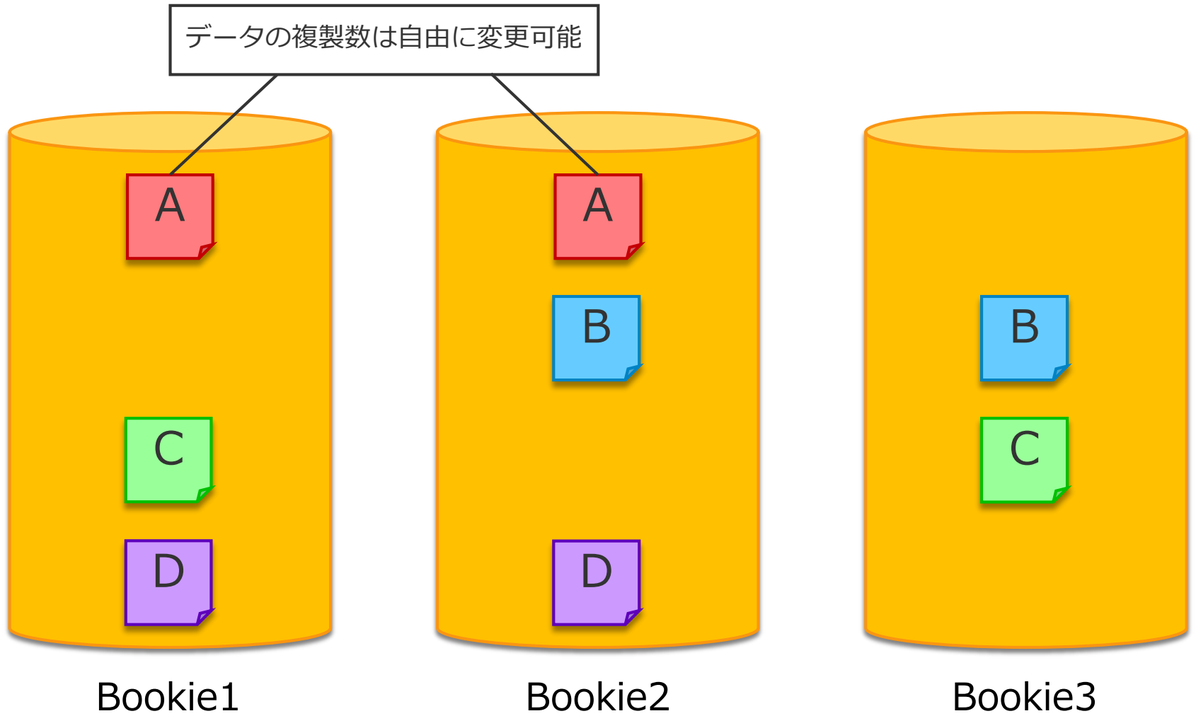
- BookKeeperのストレージノード
- トピックに送信されたメッセージやそれに関連するデータを保存

ZooKeeper

- トピックの管理に必要なメタ情報を保存
- Local ZKはクラスタ内に閉じた情報を担当
- Global ZKは全てのクラスタで共有すべき情報を担当

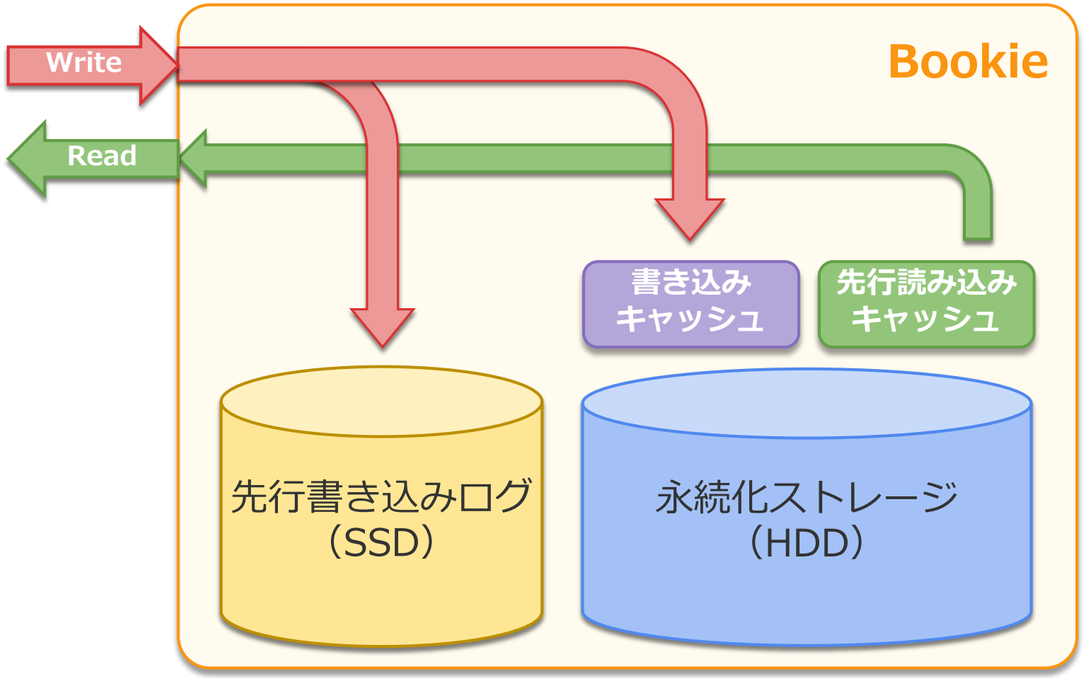
BookKeeperとは？

- オープンソースの分散型ログストレージサービス
- データを複製し複数のノード（Bookie）に分散して保存 → 強い耐障害性
- ノード数を増やせば容量と速度の向上が可能 → スケーラブル



BookKeeperのアーキテクチャ

- 先行書き込みログ
  - 高速なSSDを使用
  - ストレージに対する「操作」を先に書き込む
  - ストレージへの反映は後からバックグラウンドで
  - 途中で電源が落ちても「操作」を再開可能
- 永続化ストレージ
  - 速度はなくても大容量なHDDを使用
  - 連続するデータをまとめてキャッシュに読み込んでおく
  - 一般にメッセージキューは連続するデータを扱うため効率がよい

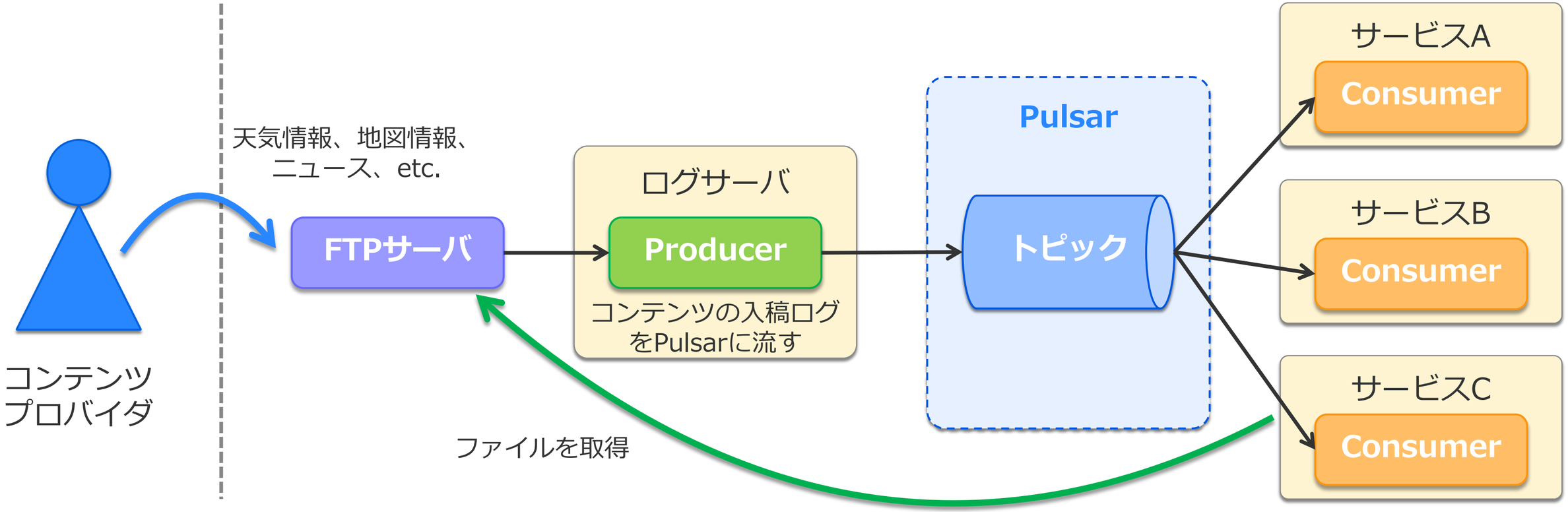


速度と永続性の両立を実現

利用事例

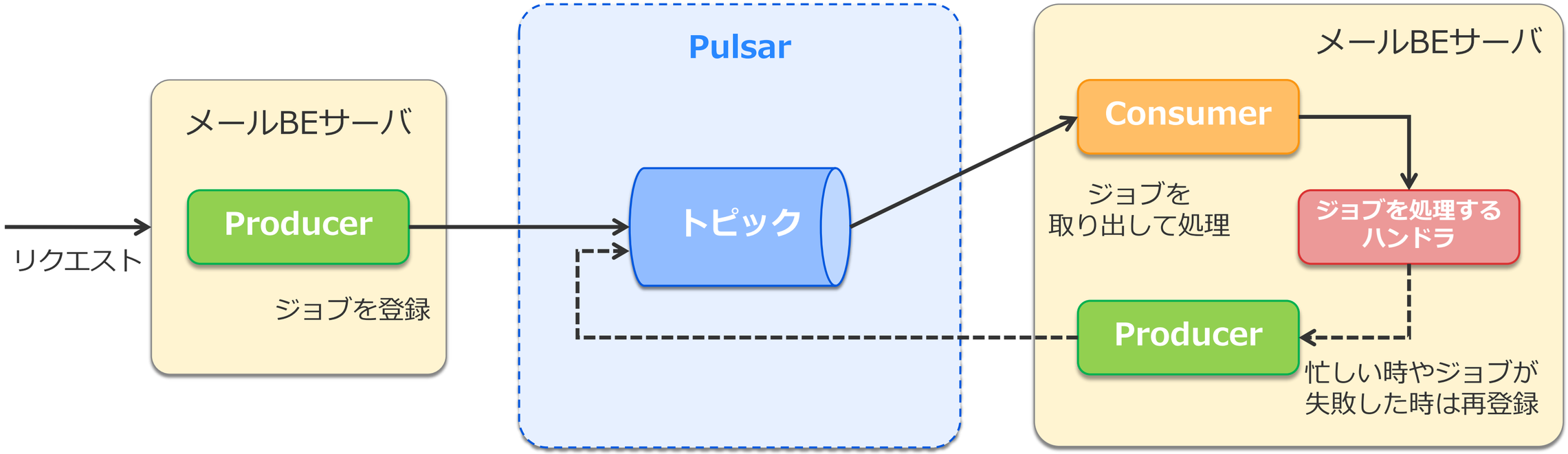
利用事例 - コンテンツ入稿プラットフォーム

- ヤフーには外部のコンテンツプロバイダから様々なファイルが入稿される
- FTPサーバへのファイル転送を検知したら入稿ログをPulsarに流す
- トピックを購読している各サービスがファイルを取得して処理する



利用事例 - Yahoo!メールのBEシステム（予定）

- メールの検索インデックスの生成/修正といった時間のかかるジョブを非同期に処理するために利用
- ProducerはPulsarにジョブをキューイング
- ConsumerはPulsarからジョブを取り出して順番に処理



Yahoo! JAPANのPulsarに対する取り組み

- C++クライアントライブラリのOSS化
- Athenz用認証プラグインのOSS化
- Spark Streamingのカスタムレシーバ作成
- ドキュメントの日本語化
- その他細かいバグ修正や機能追加

まとめ

まとめ

- Pulsar
  - 分散Pub-Subメッセージングシステム
  - 高速、スケーラブル、マルチテナント、ジオレプリケーション
  - 特に大規模なサービスでの利用に向く
- コードとドキュメントは github.com/yahoo/pulsar にあります
- 日本語ドキュメント もあります
- 質問やフィードバックはメーリングリストへ：
  - Pulsar-Users
  - Pulsar-Dev

今後の予定

- Brokerの負荷分散機能の改善
  - Pulsarでは原則として1つのトピックは1つのBrokerのみが担当
  - 複数のBrokerに担当させる事もできるが、あまり効率的に分散されない
  - 負荷分散の効率化がさらなるパフォーマンス向上に繋がる可能性も
- パブリックなトピックの実現
  - 現状は全てのトピックで認証を必須にするか、まったくしないかの2択
  - 誰でも認証なしで購読できるトピックの導入
  - 組織内データの利活用に有用

One more thing…

Apache Incubatorプロジェクト

- PulsarがApache Incubatorプロジェクトとなる事が満場一致で決定！
  - Apache Software Foundationのプロジェクトになるためのスタート地点

```
Thank you for all that participated in the VOTE.

The VOTE has ended and have this result :

9 +1 binding votes
7 +1 non-binding votes
No 0 votes
No -1 votes

With this Pulsar is officially accepted as Apache incubator project.
Congratulations!
```

- 皆さんもぜひPulsarを使ってみてください


YAHOO!
JAPAN