Engineer

# キュー

ほかの処理が取り出せるよう、仕事を順番に待たせる仕組み

## どのようなものか

キューは、二つのプログラムの間にある待機場所です。一方がメッセージやジョブを入れ、もう一方が、準備できたときに取り出します。通常は、古いものが先です。多くのシステムには、仕事の種類ごとに一つずつ、複数のキューがあります。仕事を追加するプログラムをプロデューサー、取り出すプログラムをコンシューマーと呼びます。間でメッセージを保持するものがブローカーで、専用のサーバーの場合もあれば、システムが既に使うデータベースのテーブルの場合もあります。

この構成によって、双方が互いを待たずに済みます。メール送信が必要なWebリクエストは、メッセージを追加し、すぐに利用者へ応答します。少し後にワーカーが取り出し、百件が一度に届いても、自分の速度で処理します。突発的な集中は未処理の蓄積になり、依存先を待つ関係は、処理の遅延になります。

配信についてキューが何を約束するかは、二つの考え方で説明できます。一つ目は確認応答です。コンシューマーは、メッセージの処理が終わるとブローカーに知らせ、その通知を受けて初めて、ブローカーがメッセージを削除します。RabbitMQが実装するAMQP 0-9-1のモデルでは、確認応答前にコンシューマーが停止すると、そのメッセージは別のコンシューマーへ渡ります。二つ目は永続性です。永続化されたメッセージを入れた永続キューは、ブローカーが再起動しても残りますが、メモリーだけに保持するキューは残りません。

近い三つの用語は、区別する必要があります。メッセージキューは、サービス間でデータを運びます。ジョブキューは、それぞれ引数、状態、試行回数を持つ仕事の単位を、バックグラウンドのワーカーへ渡します。Apache Kafkaが使うログのモデルは、外からはキューのように見えますが、読み取り後も、トピックごとに決めた保存期間の間はイベントを保持するため、複数の読み手が同じ履歴を再生できます。キューから読むとメッセージが消えますが、ログから読んでも、読み手の位置が進むだけです。

キューを別製品にする必要はありません。PostgreSQLでも、ジョブのテーブルと、ほかのワーカーが既にロックした行を飛ばしながら次の行を確保するクエリーで、その役割を果たせます。マニュアルは、キューのように使うテーブルを、`SKIP LOCKED` 句が存在する理由として挙げています。

```
WITH next AS (
  SELECT id
    FROM jobs
   WHERE state = 'available'
   ORDER BY id
   LIMIT 1
     FOR UPDATE SKIP LOCKED
)
UPDATE jobs
   SET state = 'running',
       attempt = attempt + 1
  FROM next
 WHERE jobs.id = next.id
RETURNING jobs.id, jobs.args;
```

概念を判断に生かす

「キューとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 月末に誰かが表計算ファイルをコピーしなくても、決まった時刻に必要な場所へ数値が届く仕組みをつくります。

**サービスを見る** お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 14部門を支える共通インフラ計画 — 14 共通インフラ計画の対象となった研究部門

**14部門を支える共通インフラ計画** お問い合わせ →

## なぜ重要なのか

キューは、利用者がどの仕事の完了を待つかを決めます。キューへ入れた仕事は、応答を返した後に行うため、メール送信、PDF生成、検索索引の再構築、再計算を起こすページも、四行を書き込む時間で返答し、四つの遅い処理は、その後で終えられます。購入者にとっては、数百ミリ秒で応答する決済と、メールサーバーの準備ができるまで応答しない決済の違いです。

障害後に何が残るかも決めます。バックグラウンドのスレッドで始めた仕事は、そのプロセスのメモリーにあり、デプロイでプロセスを再起動すると、仕事も消え、行うはずだったという記録すら残りません。永続キューに置いた仕事は、記録として残り、再試行、調査、集計ができます。

二つの速度も切り離せます。プロデューサーは、入ってくる通信の速さで動き、コンシューマーは、呼び出す中で最も遅いサービスの速さで動きます。間のキューが差を吸収し、キューにたまったメッセージ数は、システムが報告できる数値の中でも、特に実態をよく示すものになります。一時間にわたり増え続けるキューは、プロセッサーのグラフがどう見えても、コンシューマーが遅れを広げていることを示します。

利点とともに、代償も生まれます。送信してから完了するまで、仕事が見えない場所に置かれます。配信の保証は、ほぼ常に「少なくとも一回」であるため、コンシューマーは、同じメッセージが二度届いても対応する必要があります。順序の保証も、見た目ほど強くありません。専用のブローカーを使えば、運用し、安全を保ち、監視し、更新するサーバーが、もう一つ増えます。

## 実際の活用例

あるプラットフォームでは、メール送信、検索索引の再構築、文書生成、順位の再計算を、すべてリクエストの処理経路から外しました。goroutineへ渡した仕事は、デプロイの途中でプロセスが再起動すると消えるため、PostgreSQLに保持するRiverのジョブキューへ移しました。データベースは、既にシステムの正本だったため、ジョブの追加を、そのジョブが扱うデータと同じトランザクションに含められます。得られたのは、再試行と、人が開いて読める履歴です。限界も明記されています。データベース内のキューが、際限なく拡張できるわけではありません。

Cライブラリーを基に作ったmacOSアプリケーションでは、キューは順序を保つためにありました。ライブラリーのブロッキング方式のイベントループが出すイベントには、順番が重要です。編集は、編集対象のメッセージより後に届く必要があるからです。イベントごとにタスクを起動すると、ランタイムが自由な順序で実行でき、編集が、参照するメッセージを追い越す場合があります。実際にリリースしたアプリケーションの設計は、一つのプロデューサースレッド、一つのコンシューマー、両者の間の一つのキューと、その後のメインアクターへの一回の受け渡しです。通り道が一つしかないため、順序が保たれます。代償は並列性で、すべてのイベントが、その一つのコンシューマーを通ります。

キューが害をもたらすこともあります。あるmacOSアプリケーションでは、アプリケーションのコアが出すログイベントの集中が、上限のないメインのストリームに保持され、そのイベントを使わないコンシューマーを持つ購読先へも複製されていました。診断レポートには、圧縮されたページが111 GBに達し、搭載メモリーが36 GBだったことが記録されました。別のメモリー使用量のサンプルでは、接続再試行のループ中に、毎秒約41 MBの増加が記録されました。イベント

の経路に、明示的な上限を設け、購読をイベントの種類で絞り込み、あふれたときの方針を定めました。無制限に保存する代わりに、繰り返す診断イベントを捨て、影響する状態を再取得することもできます。アーキテクチャーの検査と、アプリケーションのリソース予算のテストが、この設計を守ります。ただし、この記録は、修正後のメモリー使用量の実測値や、本番環境でのベンチマークを示すものではありません。

## 問題が起きるところ

キューの問題は、遅れて、別の場所で表面化します。設計上、仕事が見えない場所に置かれるためです。

**未処理の仕事が増え、メモリーかディスクを使い切ります。** 上限のないキューは、処理できるより速く仕事を受け入れ、差をため込みます。どのキューにも最大の長さを設け、到達したときに、プロデューサーを遅くする、新しい仕事を拒否する、最も古いものを捨てる、のどれを行うか、先に決めます。その上で、待っているメッセージ数と最も古いメッセージの経過時間という二つの数値を見ます。非常に古いメッセージが少しだけあるキューも、長いキューと同じくらい問題だからです。

**同じメッセージを、二度処理します。** コンシューマーが仕事を終えても、確認応答前に停止すれば、ブローカーは、正しく、同じメッセージを再配信します。RabbitMQのドキュメントは、コンシューマーに再配信を想定し、冪等性を考慮して実装するよう求めています。各メッセージに識別子を付け、処理済みの識別子を、仕事そのものと同じトランザクションで記録します。このサイトの限られた在庫を売り越さずに販売するノートも、まさにその前提で、書き込み側を設計しています。

**送った順番と違う順番で、メッセージが届きます。** 並列のコンシューマー、先読み、失敗したメッセージを新しいものの後ろへ戻す再試行は、どれも仕事の順序を変えます。順序が重要なら、一つのアカウントや一つの注文に関するものを、すべて同じパーティションか同じコンシューマーへ送ります。Kafkaは、キーで振り分けますが、負荷の高い一つのキーを処理できる速度は、一つのコンシューマーの速度が上限になります。

**一つの問題のあるメッセージで、処理が止まるか、再配信が無限に繰り返されます。** ペイロードによって毎回コンシューマーをクラッシュさせるメッセージをキューへ戻すと、再配信され、またクラッシュさせます。試行回数に上限を設

け、使い切ったメッセージは、人が調べられる別の場所へ移します。RabbitMQは、これをデッドレタリングと呼び、拒否されたメッセージ、期限切れのメッセージ、キューの長さの上限を超えるメッセージに適用します。

**再起動で、仕事が消えます。** チャネル、メモリー内のリスト、バックグラウンドのスレッドは、再起動後には何も残らないキューです。AMQPでも、永続キューだけでは不十分で、メッセージも永続化するものとして発行する必要があります。消えてもよいかどうかを、仕事の種類ごとに決めます。

**データベースとキューが食い違います。** サービスが行を書き、その後にメッセージを発行する構成で、二つの間で停止すると、片方だけが存在します。Riverのように、キューを同じデータベースに保持し、トランザクションがコミットしたときにだけ、ワーカーからジョブが見えるようにします。あるいは、そのトランザクションでメッセージをoutboxテーブルへ書き、そこから発行します。

## 参考資料


- AMQP 0-9-1モデルの解説 — RabbitMQが説明する、エクスチェンジ、キュー、バインディング、確認応答です。
- コンシューマーの確認応答とパブリッシャー確認 — ブローカーが配信済みと判断する時点と、再配信が起きる理由です。
- デッドレターエクスチェンジ — メッセージを別へ移す条件と、移動先です。
- SELECTのロック句 — `FOR UPDATE` と `SKIP LOCKED` 、および適したキュー形式のテーブルです。
- River入門 — アプリケーション自身のデータベースにジョブを保存する、ジョブキューです。
- Apache Kafka入門 — ログのモデルにおけるトピック、パーティション、保存期間です。


用語集


実際の制約に合わせて選ぶ

代表的なデータセット、そこから答えを得たい問い、現在のボトルネックをお知らせください。品質、処理量、復旧の測り方を合意できます。

**料金を見る** お問い合わせ →

# 用語を実務に結び付ける

「キューとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/queue/