IBM MQ V9.3 機能と構成

1. 概説

2023年09月

日本アイ・ビー・エム　システムズ・エンジニアリング（株）

トピック

- IBM MQとは
  - IBM MQとは
  - MQ Familyの歩みと機能概要
  - メッセージ・キューイング型処理の特徴
  - MQの構成要素
  - メッセージ通信モデル
  - 接続形態
  - プログラミング・インターフェース
- MQの主な機能
  - メッセージの永続性
  - 同期点処理
  - メッセージの待機受信
  - メッセージの識別
  - メッセージの存続時間
  - メッセージのグループ化・セグメント化
  - ストリーミング・キュー
  - MQクラスター

(続き)

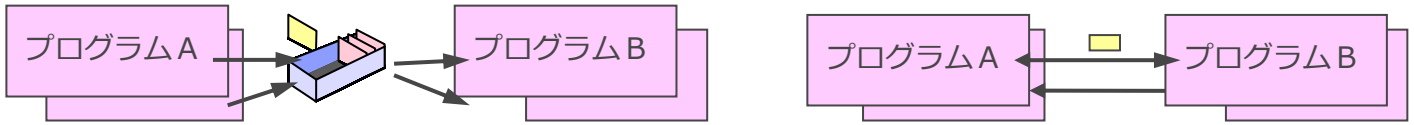
  - マルチインスタンス・キュー・マネージャー
  - RDQM
  - Native HA(参考)
  - Uniform Cluster
  - その他の主な機能
  - MQの管理機能
- 適用例
  - 長時間かかる処理
  - ディレード処理（時間差処理）
  - ファイル転送
  - ワークフロー業務
  - 既存システムのWeb化
  - 他システム連携
  - ハイブリッドクラウド統合

IBM MQとは

IBM MQとは

- 異機種システムのプログラム間でデータ通信を行うためのミドルウェア製品
  - ◆ マルチプラットフォーム サポート ：Windows、Linux、AIX、IBM i、z/OS
  - ◆ マルチ言語 サポート ：C/C++、Java、.NET、COBOL など

- メッセージ・キューイング型の非同期通信基盤を提供
  - ◆ キューを介してデータ（メッセージ）を送受信
    - プログラムはキューにメッセージを読み書きする
  - ◆ 相手とコネクションを持たない疎結合の通信



- 単一のプログラミング・インターフェースを提供
  - ◆ プログラムは、MQI(Message Queue Interface)により、キューにメッセージを読み書きする
    - JMS(Java Message Service)/Jakarta Messagingもサポート：J2EE標準のメッセージング・インターフェース

- システム連携のための基盤および、ソリューションを提供
  - ◆ 異機種間の接続や、クラウド環境をまたがったアプリケーション同士の接続を実現

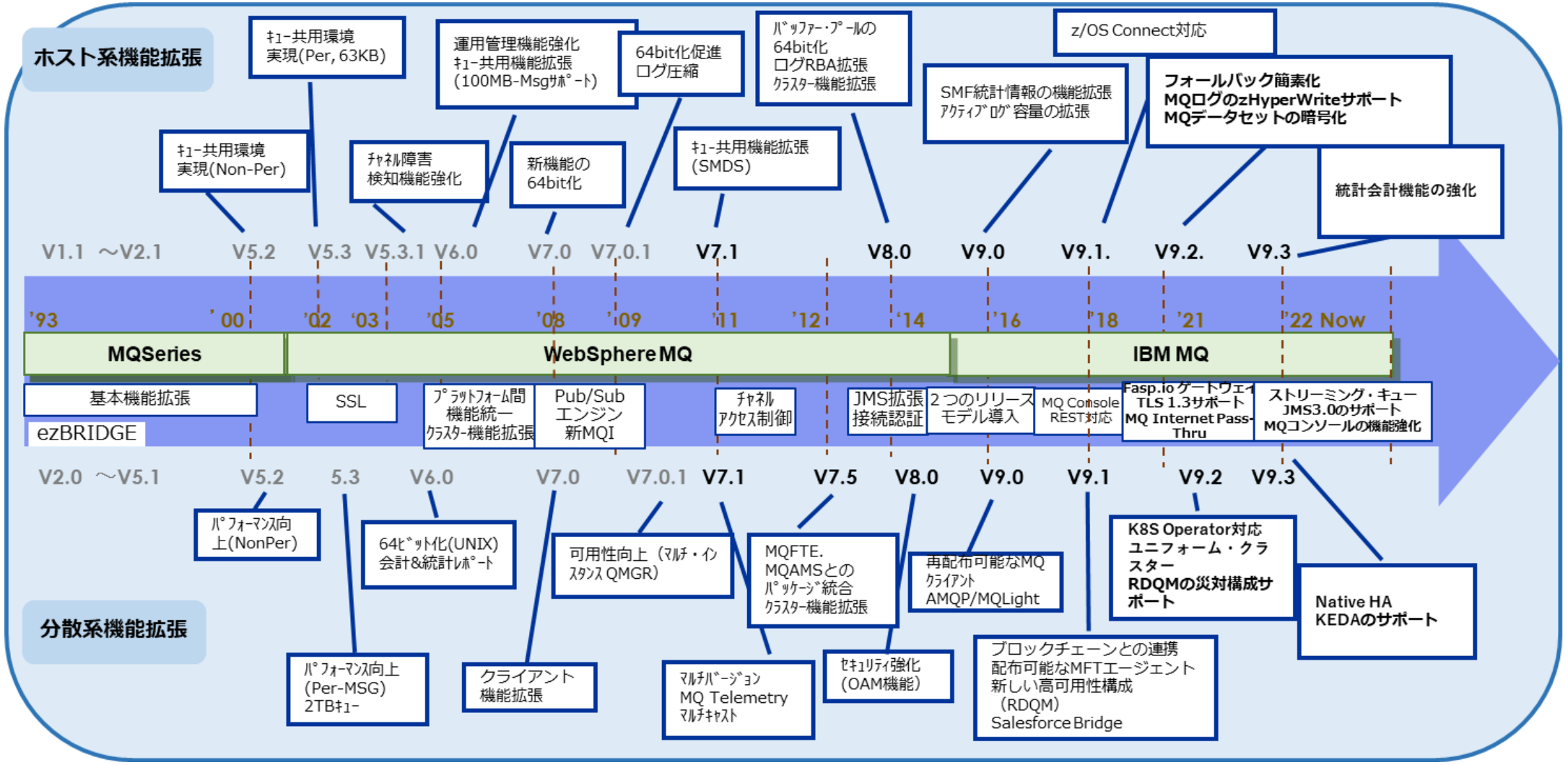
<参考> サポート･プラットフォーム一覧（MQサーバー）

OSの種類	Minimumレベル	MQ v9.0	MQ v9.1	MQ v9.2	MQ v9.3	ハードウェア
AIX	AIX 7.3	–	–	–	○	POWER System - Big Endian
	AIX 7.2 TL3	–	–	○	○	
	AIX 7.2 TL2	–	○	–	–	
	AIX 7.2	○	–	–	–	
	AIX 7.1 TL5 SP2	–	○	○	–	
	AIX 7.1 TL4	○	–	–	–	
IBM i	IBM i 7.5	–	–	–	○	
	IBM i 7.4	–	–	○	○	
	IBM i 7.3	○	○	○	○	
	IBM i 7.2	○	○	–	–	
Linux-Red Hat	RHEL 8.4	–	–	–	○	POWER System - Little Endian, System z, x86-64
	RHEL 8	–	–	○		
	RHEL Server 7.9	–	–	–	○	
	RHEL Server 7.6	–	–	○	–	
	RHEL Server 7.3	–	○	–	–	
	RHEL Server 7.2	○	–	–	–	
Linux-Suse Linux	SLES 15 SP2	–	–	–	○	
	SLES 15 SP1	–	–	○	–	POWER System - Little Endian , x86-64
	SLES 15	–	–	○	–	System z
	SLES 12 SP5	–	–	–	○	POWER System - Little Endian, System z, x86-64
	SLES 12 SP4	–	–	○	–	
	SLES 12 SP2	–	○	–	–	
	SLES 12 SP1	○	–	–	–	
Linux-Ubuntu	Ubuntu 20.04 LTS	–	–	–	○	
	Ubuntu 18.04 LTS	–	–	○	–	
	Ubuntu 16.04 LTS	–	○	–	–	
	Ubuntu 14.04 LTS	○	–	–	–	POWER System - Little Endian , x86-64
Windows	Windows 11	–	–	–	○	x86-64
	Windows 10 (20H2)	–	–	–	○	
	Windows 10	○	○	○	–	
	Windows 8.1	○	○	–	–	
	Windows 8	○	–	–	–	
	Windows 7 Service Pack 1	○	–	–	–	
	Windows Server 2022	–	–	–	○	
	Windows Server 2019	–	–	○	○	
	Windows Server 2016	–	○	○	–	
	Windows Server 2012	○	–	–	–	
	Windows Server 2012 R2	○	○	–	–	
	Windows Server 2008 R2 Service Pack 1	○	–	–	–	
Solaris	Solaris 11 Update1	–	○	–	–	x86-64, SPARC
	Solaris 11	○	–	–	–	
	Solaris 10	○	–	–	–	
HP-UX	HP-UX 11i v3	○	–	–	–	IA64
z/OS	z/OS 2.5	○	○	○	○	IBM z Systems
	z/OS 2.4	○	○	○	○	
	z/OS 2.3	○	○	○	–	
	z/OS 2.2	○	○	–	–	
	z/OS 2.1	○	–	–	–	

＊　MQの互換性製品として日立社より、Open TP1/MQを提供

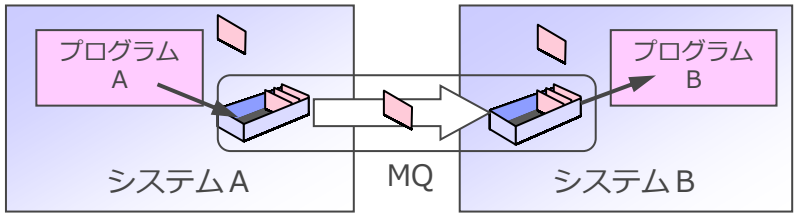
＊　OSの詳細バージョン、最新情報などは右のリンクを参照：https://www.ibm.com/support/pages/system-requirements-ibm-mq

MQ Familyの歩みと機能概要

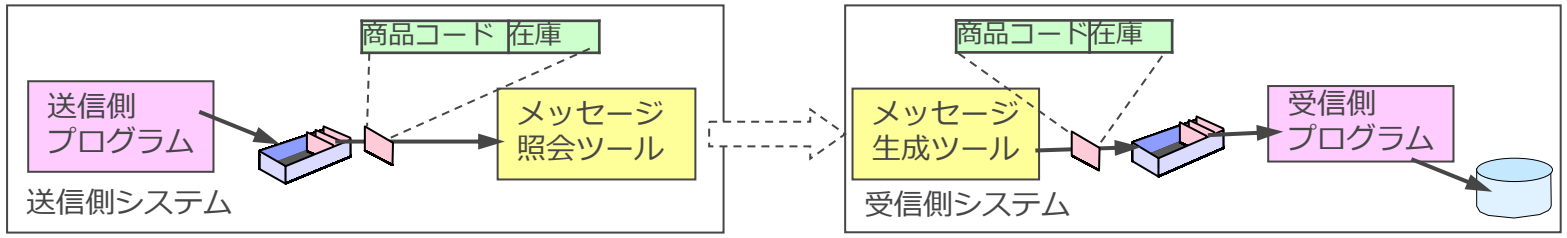


メッセージ・キューイング型処理の特徴

- キューを介したデータ（メッセージ）通信
- 相手の稼動状態に依存しない処理が可能
  - ◆ 通信相手のプログラムやシステムが稼働していなければ、メッセージをキューに一時保存
  - ◆ システム（キュー）間のメッセージの転送はMQが保証
    - 稼働を待ってメッセージを転送
  - ◆ 大量DB更新処理、ファイル/イメージデータの送信、ディレード処理など

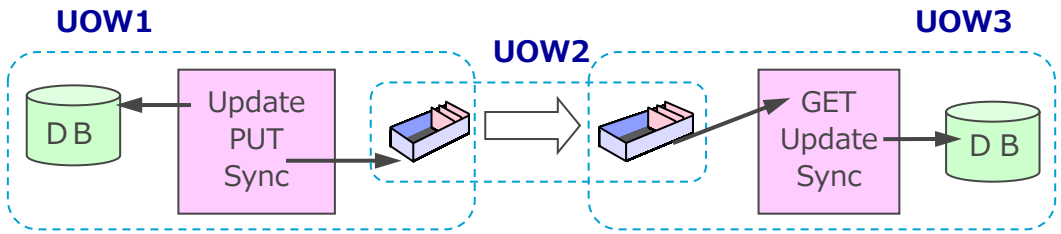


- 開発/テスト、問題判別が容易
  - ◆ 送信側/受信側のアプリケーションを独立して開発/テストが可能
  - ◆ 決められたフォーマットのメッセージを書き出し/読み込み処理することで検証が可能
    - メッセージ参照ツール、メッセージ生成ツールの作成
  - ◆ インターフェース（メッセージ・フォーマット）が明確化されることにより、システム間連携において、開発生産性の向上および、障害発生時の問題判別が容易となる

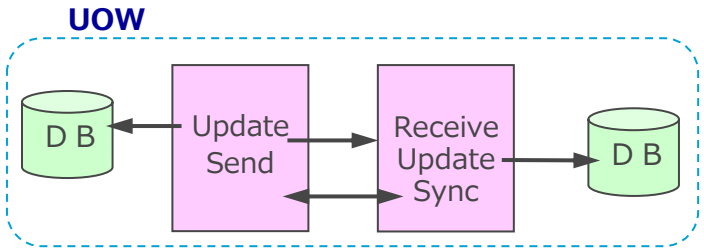


メッセージ・キューイング型処理の特徴

- リソース更新のUOWが分かれる
  - リソース更新がローカルで完了するので、データベースのロック時間を最小化できる
  - システムを跨ったリソース更新の同期はできない

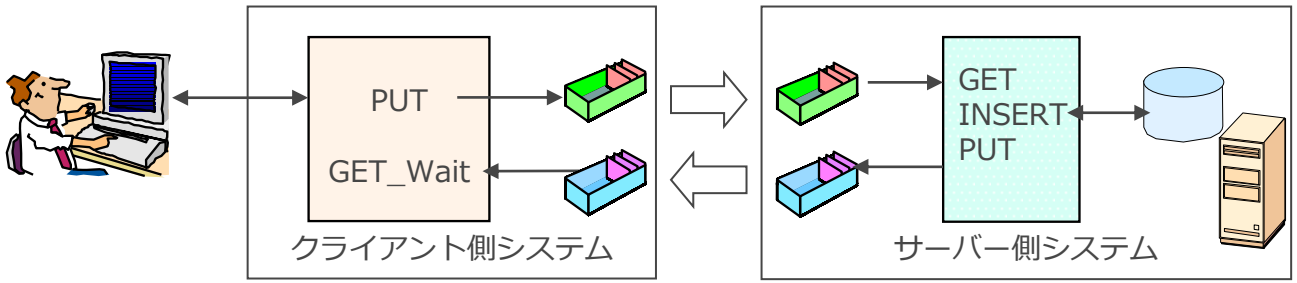


【非同期モデルのUOW】



【同期モデルのUOW】

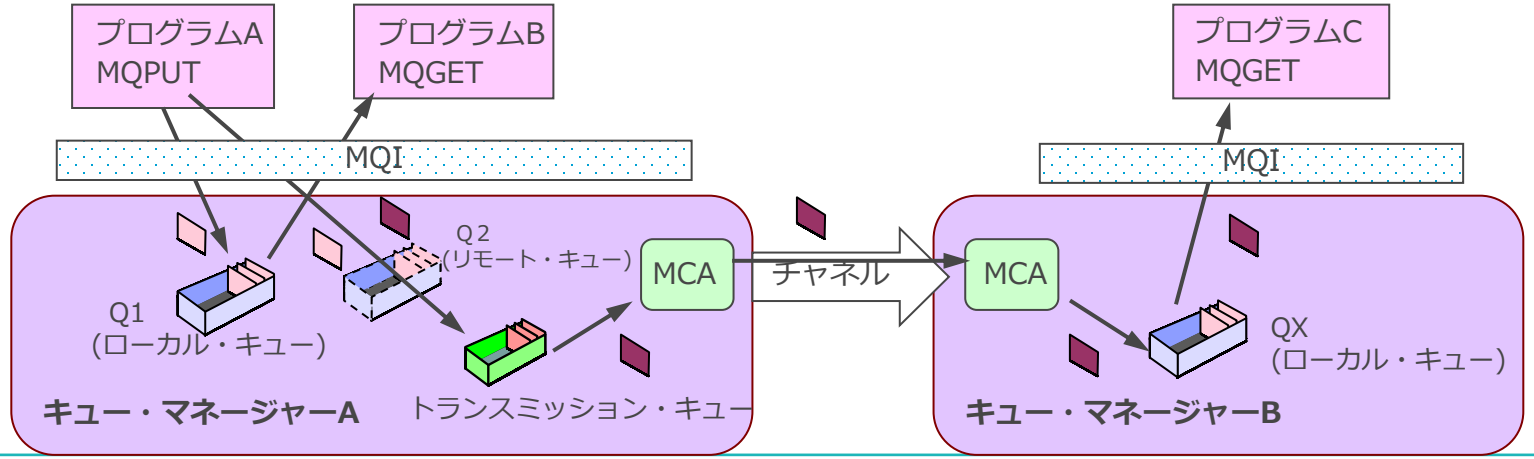
- 非同期型処理だけでなく、同期型処理も実装可能
  - 要求/応答型処理
  - 障害時のリカバリーに注意が必要
    - キューに滞留しているメッセージの扱い



MQの構成要素

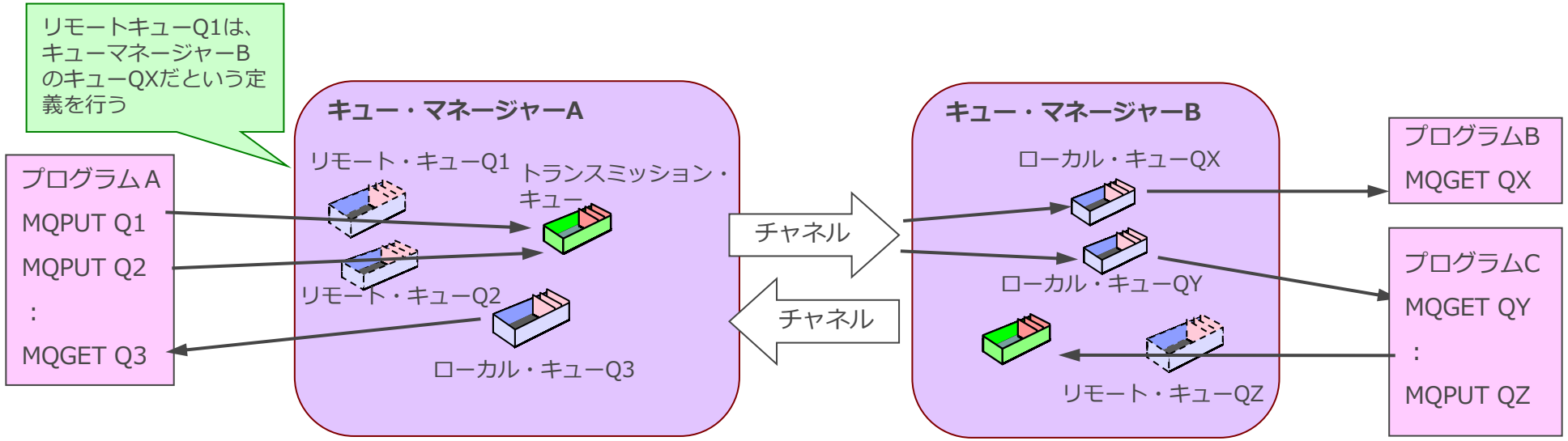
- キュー・マネージャー
  - ◆ キュー、チャネルなどを管理
    - いくつかのシステム・プロセスから構成される
- キュー
  - ◆ メッセージを保持する論理的な箱
    - 実体はメモリ、ファイルなどのストレージ
- チャネル
  - ◆ キュー・マネージャー間のコミュニケーション・リンク
  - ◆ 両端でMCA（メッセージ・チャネル・エージェント）が稼働
    - MCAはキューにメッセージを読み書きする通信プログラム

- メッセージ
  - ◆ キューを介してプログラム間で送受信されるデータ
    - MQのヘッダー（MQMD） + ユーザー・データ（最大100MB)



MQの構成要素

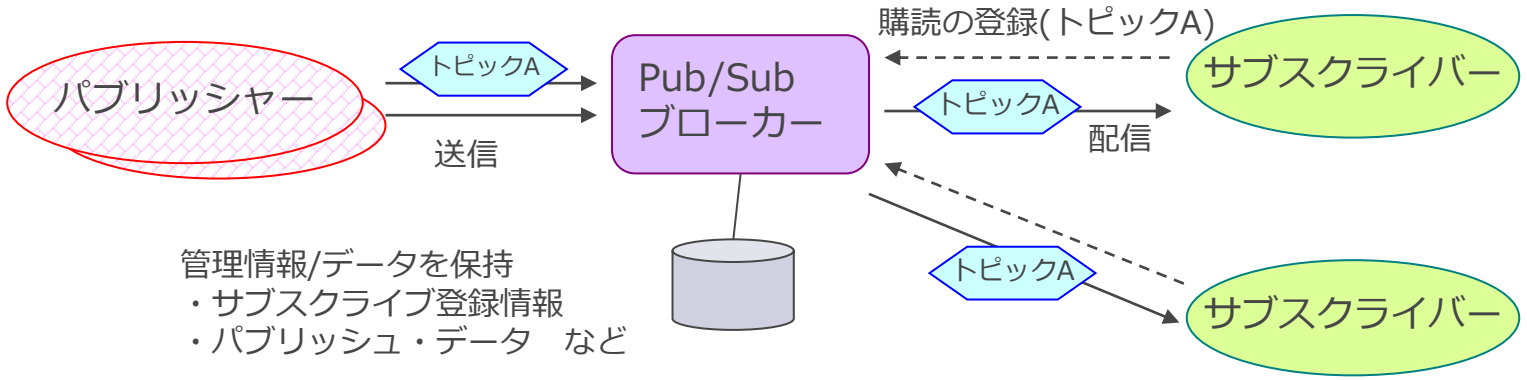
- チャネル（続き）
  - ◆ リモート・システムへのメッセージ送信の流れ
    - リモート・キューに宛先キュー・マネージャー名とキュー名を定義しておく
    - リモート・キューに書き込まれたメッセージは、転送用のキュー（トランスミッション・キュー）に書き込まれる
    - 転送用のキューからチャネルがメッセージを取り出して、宛先のローカル・キューに送信
  - ◆ チャネルは一方向通信のため、キュー・マネージャー間に送信用/受信用の1組を作成する
    - １組のチャネルで複数のキューへの転送が可能



メッセージ通信モデル

- Point-to-Point（PTP）型
  - ◆ 送信プログラムと受信プログラムがキューを介して1対1で通信するモデル
  - ◆ 最も単純な接続形態

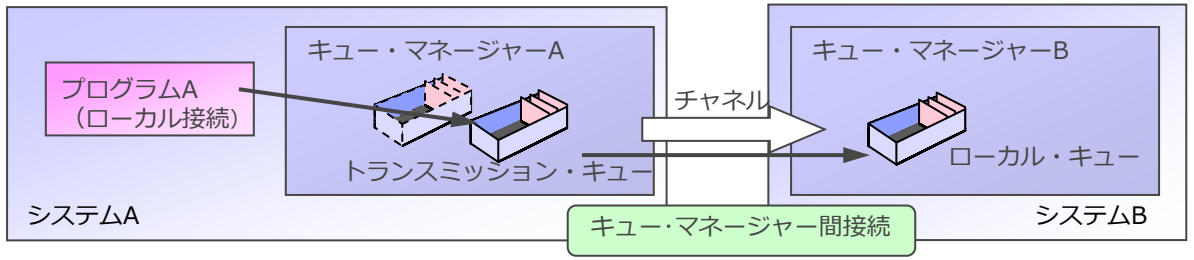
- Publish / Subscribe型
  - ◆ 送信プログラムと受信プログラムがトピックを介して１対Nで通信するモデル
    - 送信側（パブリッシャー）が受信側（サブスクライバー）を意識せずにデータ送受信を行う
    - サブスクライバーは受信したいトピックにサブスクライブ（購読）登録を行う
    - パブリッシャーは、送信したいデータをトピックに対してパブリッシュ（送信）する
    - 両者の間には必ずPub/Subブローカーが介在し、データの管理、配信を行う



接続形態

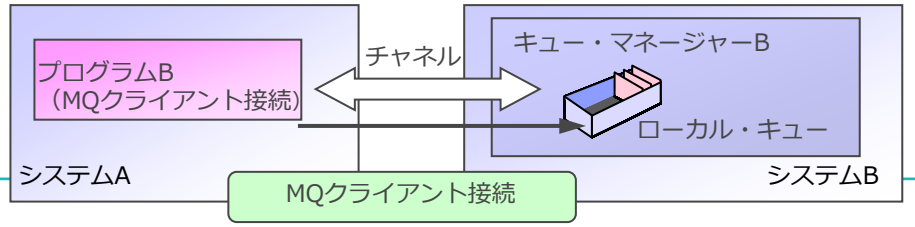
キュー・マネージャー間接続

- 双方のシステム上にキュー・マネージャーが稼動
- 非同期通信（宛先のキュー・マネージャーの稼動状況に依存しない）
- サーバー同士の接続では、MQクラスター機能の利用が可能
- 双方のシステムでキュー、チャネル、ログなどのリソースの管理、運用が必要



MQクライアント接続

- リモートのキュー・マネージャーとMQクライアント接続してキューにメッセージを読み書きする
- キュー・マネージャーの稼動していないシステムでもMQアプリケーションの稼動が可能
- クライアント側でのデータの蓄積は不可（キューが存在しないため）
- 同期通信（宛先のキュー・マネージャーの稼動状況に依存）
- クライアント側では前提H/Wスペックが緩和、システム管理 / 運用コストが小さい
- クライアント側のライセンス費用は無償



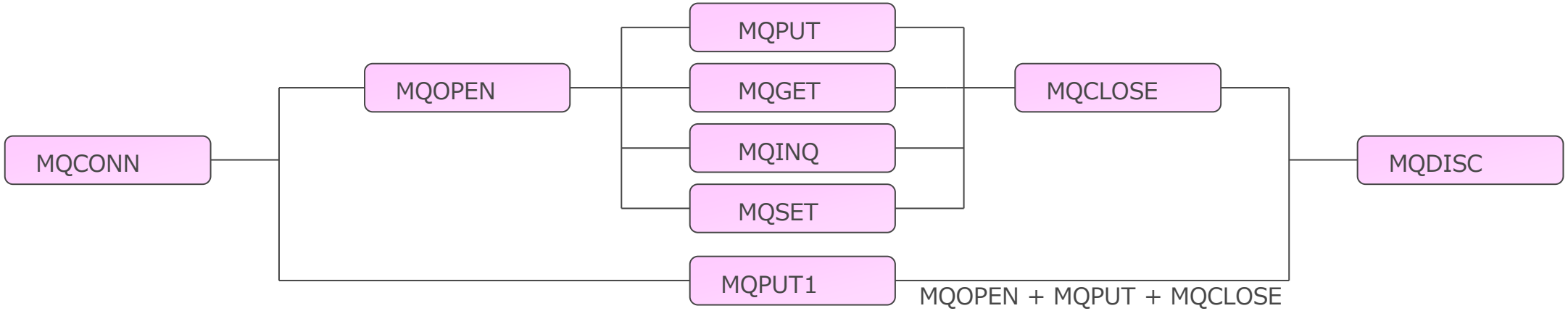
プログラミング・インターフェース

MQI(Message Queue Interface)

◆ MQの標準APIで、多数の言語から同一のインターフェースでプログラミング可能

基本のMQI

◆ メッセージの送受信やトランザクション制御用のAPI



- キュー・マネージャーとの接続 / 切断
  - MQCONN（MQCONNX） / MQDISC
- キューのオープン/クローズ
  - MQOPEN / MQCLOSE
- メッセージの読み書き
  - MQPUT（MQPUT1） / MQGET

- オブジェクト定義情報の照会 / 更新
  - MQINQ / MQSET
- 同期点コール
  - MQCMIT / MQBACK
- グローバル・トランザクションの開始
  - MQBEGIN

プログラミング・インターフェース

その他のMQI

◆ 利用する機能に応じて使用するAPI

- Pub/Sub関連
  （サブスクライブ登録）
  - MQSUB / MQSUBRQ
- メッセージ・プロパティ関連
  （メッセージ・プロパティの設定/参照）
  - MQSETMP / MQINQMP / MQDLTMP
  - MQCRTMH / MQDLTMH
  - MQBUFMH / MQMHBUF
- 非同期メッセージ受信
  （コールバック機能の利用）
  - MQCB / MQCTL / MQCB_FUNCTION
- 非同期メッセージ送信
  （MQPUTの実行結果を後から確認）
  - MQSTAT

<参考> プログラミング・インターフェース

MQIが使用できる開発言語のサポート状況

	AIX	Linux	Windows	z/OS	IBM i
C	○	○	○	○	○
C++	○	○	○	○	○
COBOL	○	–	○	○	○
PL/I	– (v5.3まで○)	–	– (v6.0まで○)	○	–
VB	–	–	○	–	–
ActiveX	–	–	– (v9.1まで○)	–	–
.NET	–	○	○	–	–
Base Java(注1)	○	○	○	○	○
JMS2.0(注2)	○	○	○	○	○
Jakarta Messaging3.0(注3)	○	○	○	○	○
RPG	–	–	–	–	○
Assembler	–	–	–	○	–

注1：Base Javaは、v8.0の機能レベルまでをサポートし、今後の新機能の追加や機能拡張は提供されない
注2：JMS2.0は、JMS2.0 standardをサポートするが、既存のJMS2.0アプリケーションの保守および機能拡張のみで、新規の機能拡張は後継のJakarta Messaging3.0で行われる
注3：Jakarta Messaging3.0はJMS2.0と機能的に同等であり、新規にJavaでメッセージング・アプリケーションを開発する際はJakarta Messaging3.0の使用が推奨される

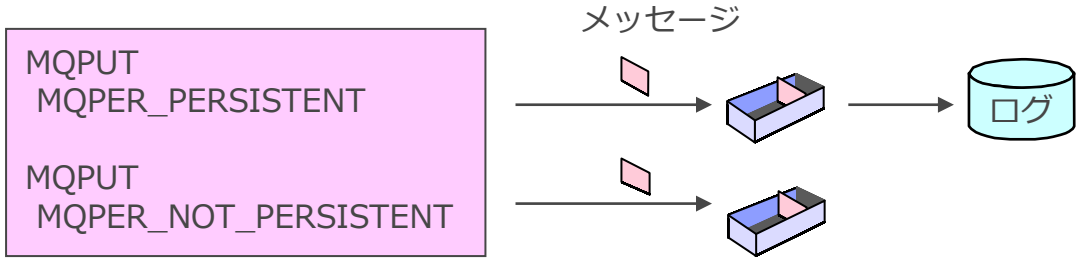
MQが提供する主な構造体

構造体	使用目的
MQOD	オープンするキュー名の指定
MQMD	メッセージヘッダー、メッセージの属性を指定（永続性、メッセージID、プライオリティ、存続時間など）
MQPMO	メッセージの送信方法に関するオプション（同期点処理の要否など）
MQGMO	メッセージの受信方法に関するオプション（同期点処理の要否、受信待機の要否、待機時間 など）

主な提供機能

メッセージの永続性

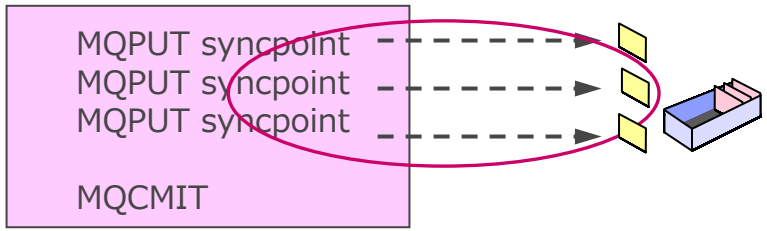
- キュー・マネージャーの再起動をまたがってメッセージを保持または、削除
  - ◆ パーシステント・メッセージ
    - メッセージを保持する
    - メッセージ読み書き時にリカバリのためのログを取得、ディスクI/Oが発生するためパフォーマンスが劣る
  - ◆ ノンパーシステント・メッセージ
    - メッセージは消失する
    - ログを取得せず、ディスクI/Oが発生しないため、パフォーマンスがよい
- 永続性はメッセージ毎に設定できる
  - ◆ メッセージ・ヘッダー(MQMD.Persistence)に設定して、MQPUT
    - MQPER_PERSISTENT　　パーシステント
    - MQPER_NOT_PERSISTENT　　ノンパーシステント
    - MQPER_PERSISTENCE_AS_Q_DEF　　キューのDEFPSIST属性に従う
  - ◆ 同一キュー内にパーシステント/ノンパーシステントの混在も可能



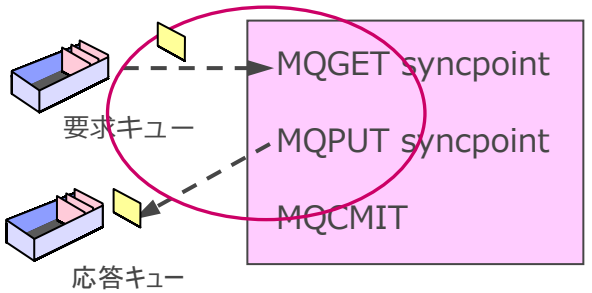
同期点処理

- 複数のメッセージの送受信を1UOWで処理
  - ◆ メッセージ毎に、同期点処理するかどうかを指定できる
  - ◆ MQPUT時に、MQPMO.Optionsに同期点処理の要否を指定
    - MQPMO_NO_SYNCPOINT　　同期点処理に参加しない
    - MQPMO_SYNCPOINT　　同期点処理に参加する
  - ◆ MQGET時に、MQGMO.Optionsに同期点処理の要否を指定
    - MQGMO_NO_SYNCPOINT　　同期点処理に参加しない
    - MQGMO_SYNCPOINT　　同期点処理に参加する
    - MQGMO_SYNCPOINT_IF_PERSISTENT パーシステント・メッセージであれば同期点処理に参加
  - ◆ MQCMIT / MQBACKで同期点をとる
  - ◆ 同期点前のメッセージは、他のアプリケーションからアクセスできない

同期点処理の例1.
関連する複数のメッセージをすべて送信するか/
すべて送信しないかで制御



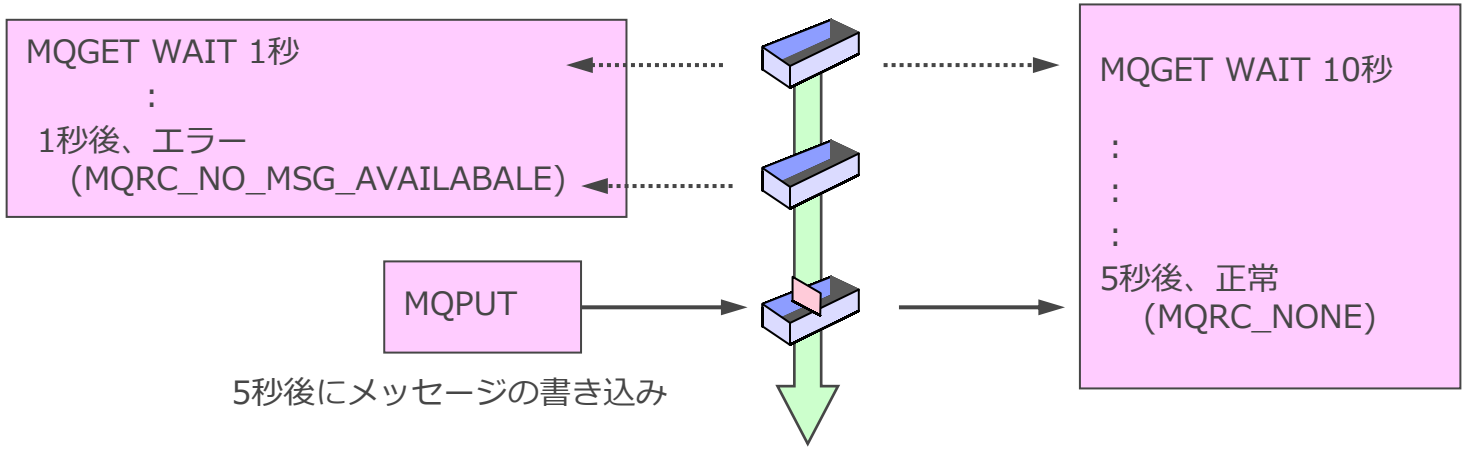
同期点処理の例2.
応答メッセージの確実な返却



メッセージの待機受信

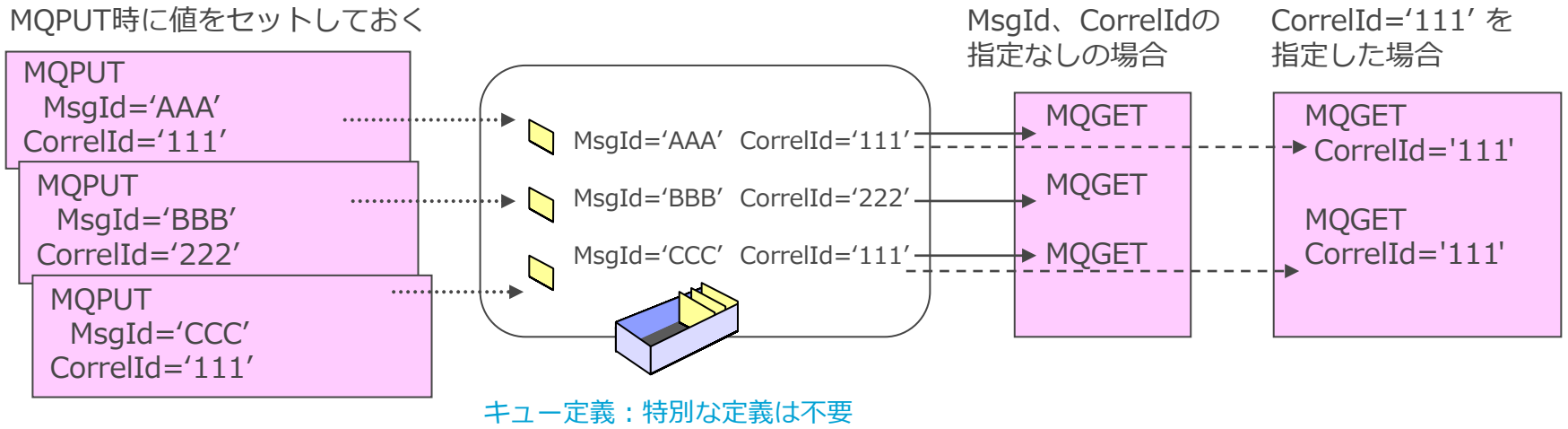
■デフォルトでは、メッセージを待機しない
- ◆MQGET時にキューにメッセージがなければエラー

■メッセージがキューに書き込まれるのを待ち受けることができる
- ◆MQGET時に、MQGMO.OptionsにMQGMO_WAITを設定
- ◆最大待機時間は、MQGMO.WaitIntervalに指定(単位ms)
  - ●MQWI_UNLIMITEDを指定して無限に待機することも可能
- ◆待機時間を経過するか、メッセージが書き込まれるまでMQGETがブロックされる



メッセージの識別

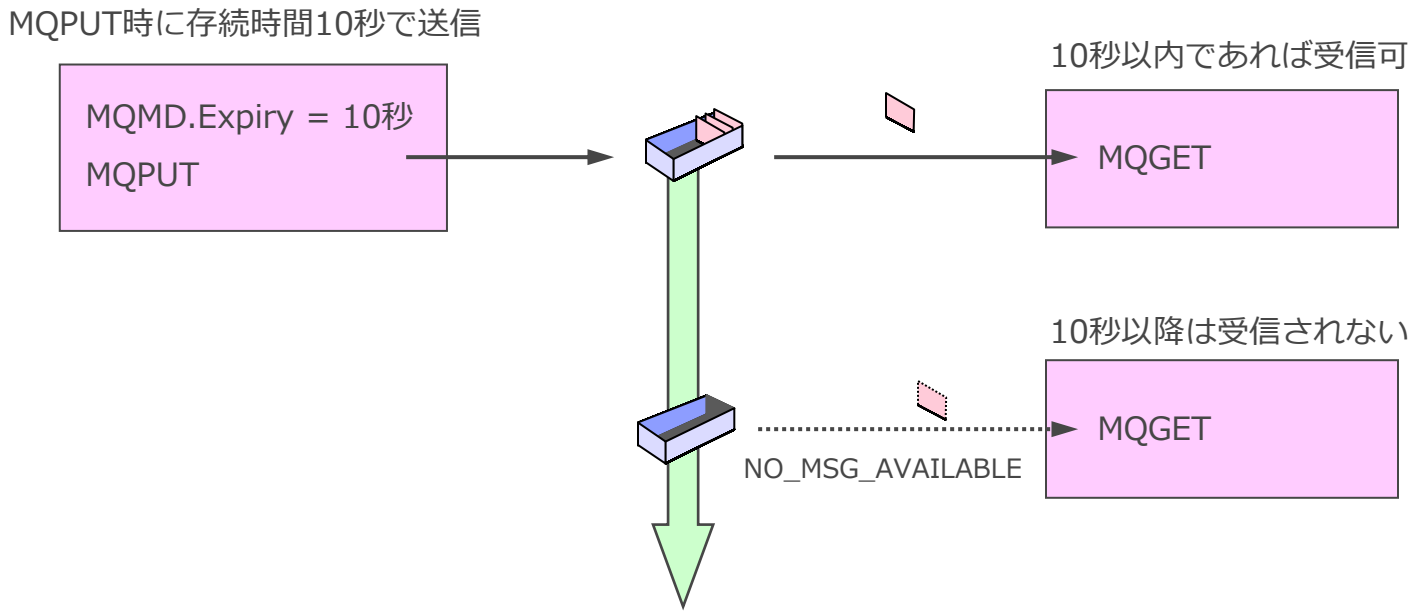
- 個々のメッセージの識別のために２種類のキーの指定が可能
  - ◆ メッセージID（MQMD.MsgId）　：キュー・マネージャーに任意の値を設定させることも可能
  - ◆ 相関ID　（MQMD.CorrelID）
  - ◆ メッセージ・ヘッダー（MQMD.MsgId、MQMD.CorrelID）に値を指定して、MQPUT
- キューからID指定によりメッセージを選択して受信できる
  - ◆ メッセージ・ヘッダー（MQMD.MsgId、MQMD.CorrelID）に値を指定して、MQGET



- リクエスト/リプライ処理で、要求メッセージと応答メッセージの紐付けに利用

メッセージの存続時間

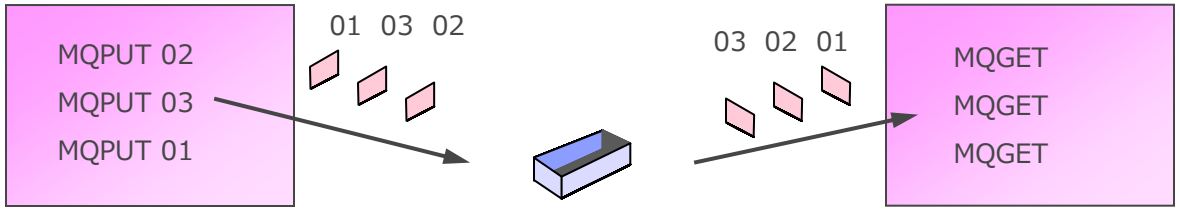
- 指定時間、受信されないメッセージを自動削除
  - メッセージ・ヘッダー（MQMD.Expiry）に存続時間を設定して、MQPUT
    - 存続時間は1/10秒単位
    - デフォルトは無限（MQEI_UNLIMITED）



メッセージのグループ化・セグメント化

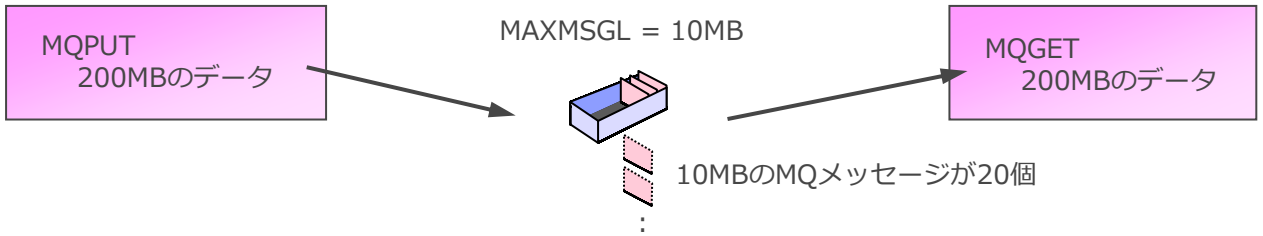
グループ・メッセージ機能

- 関連する複数の物理的なメッセージをグループ化して送信
- メッセージを物理順序でなく、論理順序で受け取ることも可能
- 同一のグループIDをもつ、複数の論理メッセージで構成する



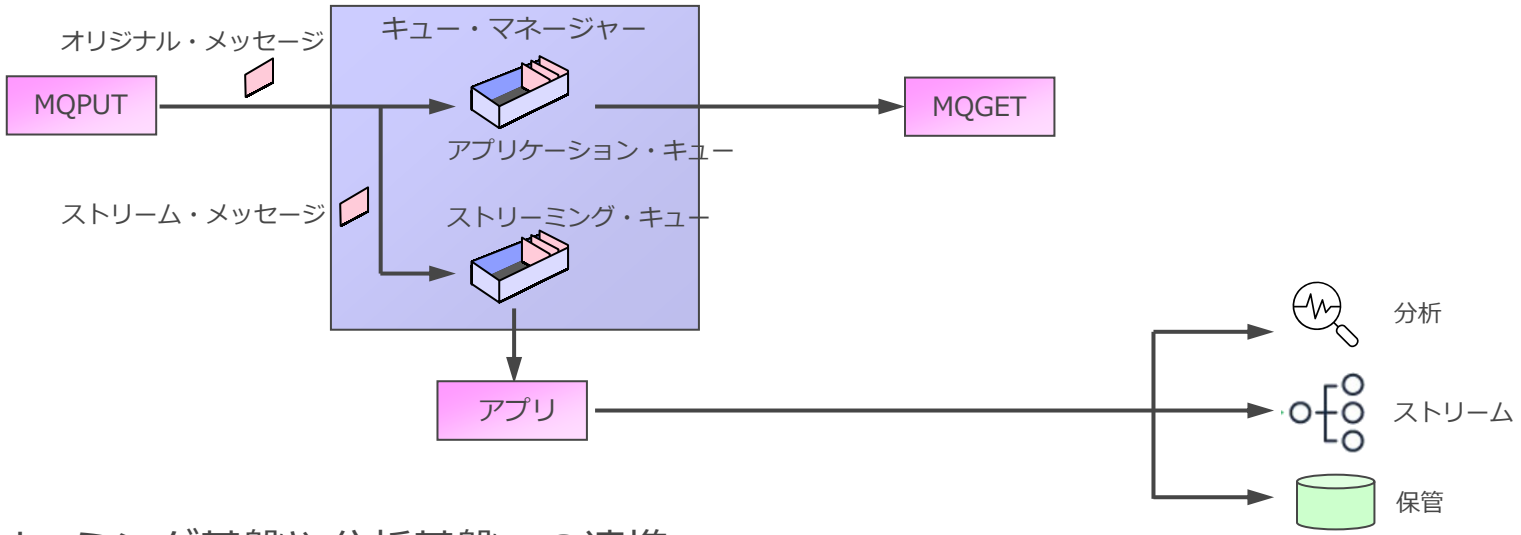
セグメント・メッセージ機能

- ラージ・メッセージを、物理的に分割（セグメント化）して複数のセグメント・メッセージで送信
- セグメント・メッセージへの分割、元のメッセージへの組み立てをMQに任せることが可能
- セグメント化はz/OS版では未サポート



ストリーミング・キュー

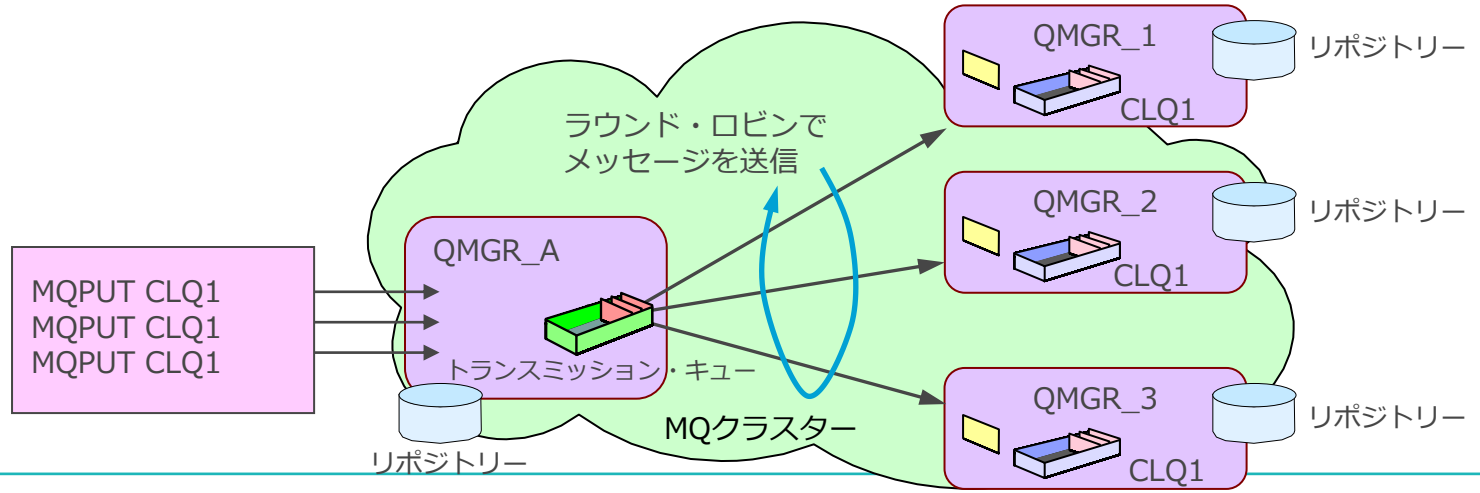
- キューに書き込まれたメッセージを、指定した別のキューにコピーする機能
  - ◆ 既存のアプリケーション設定を変更せずに、メッセージのコピーが可能
  - ◆ ローカル・キューとモデル・キューで利用可能
  - ◆ 2つのキュー属性を使用
    - STREAMQ属性：コピー先のキューを指定
    - STRMQOS属性：サービス品質(Best effort, Must duplicate)を設定



  - ◆ 使用例
    - Kafka等のストリーミング基盤や分析基盤への連携
    - リカバリーのためのメッセージ保管
    - 開発・テスト用のメッセージ・コピー

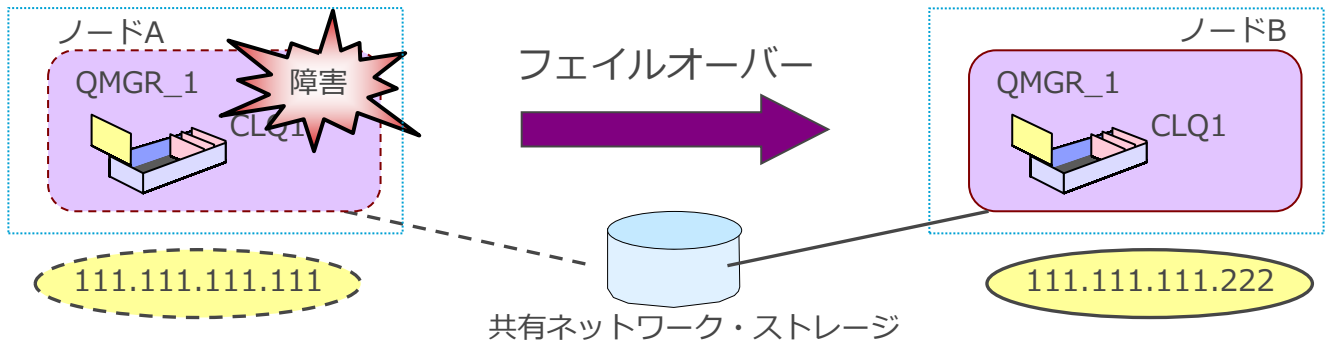
MQクラスター

- MQクラスターとは
  - ◆ キューやチャネルの定義情報を共有するキュー・マネージャーの集合
- 容易な構成管理を実現
  - ◆ リモートのキュー定義やチャネル定義を自動的に交換し、リポジトリーに保持する
- メッセージ送信時のワークロード・バランシングを実現
  - ◆ クラスター属性をもつ同一名称のキューに対して、ラウンド・ロビンで宛先振り分けを行う
    - 振分け対象のチャネルやキューに重み付けを付与することも可能
    - メッセージ単位に振り分け
    - MQOPENの単位に振り分け：関連性をもった複数メッセージを同一の宛先に送付
  - ◆ ラウンド・ロビンで送信するときは、メッセージ送信元にはキュー・マネージャーが必要



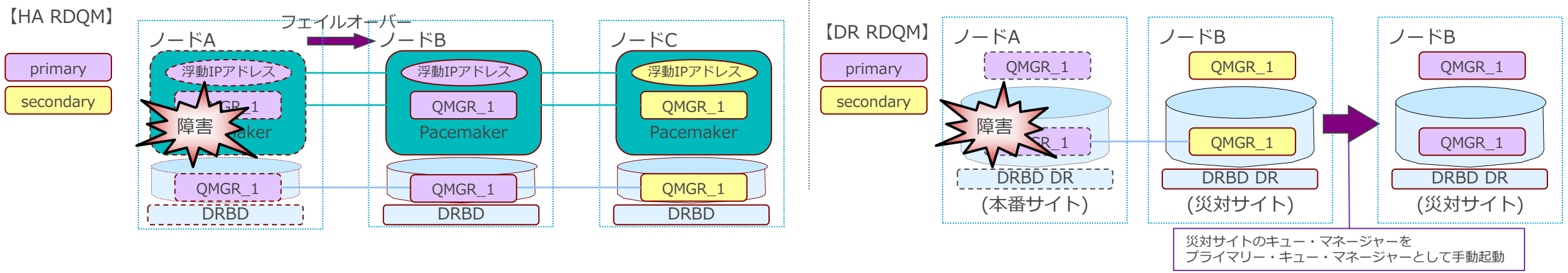
マルチインスタンス・キュー・マネージャー

- HAクラスター製品を使用せずにフェイルオーバーを実現する機能
  - ◆ HAクラスター製品を使用せずに同一キュー・マネージャーを複数マシンで稼動させることが可能
  - ◆ キュー・マネージャー実体(キュー・ファイル、トランザクション・ログ)をネットワークストレージに配置
  - ◆ 障害発生時は待機ノードでキュー・マネージャーを起動
    - ● 障害時にクラスター・ソフトウェアなしに共有ネットワーク・ストレージを待機系に引継ぎ
      - ただし、IPアドレスは待機系に引継がれない
    - ● 仕掛中のトランザクションのロールバック、パーシステント・メッセージをキューに回復
  - ◆ 自動フェイルオーバーが可能
    - ● アクティブ・インスタンスで障害が発生した場合、ネットワーク障害で共有キュー・マネージャー・データに接続されていない場合など、自動でスタンバイ・インスタンスへフェイルオーバー
  - ◆ 待機ノードでサービスが開始されるまでサービス提供が停止
  - ◆ 待機ノードでMQのライセンスがかからないため安価



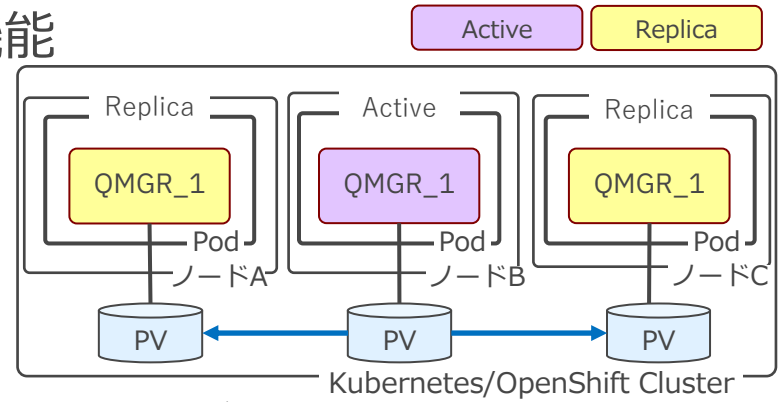
RDQM

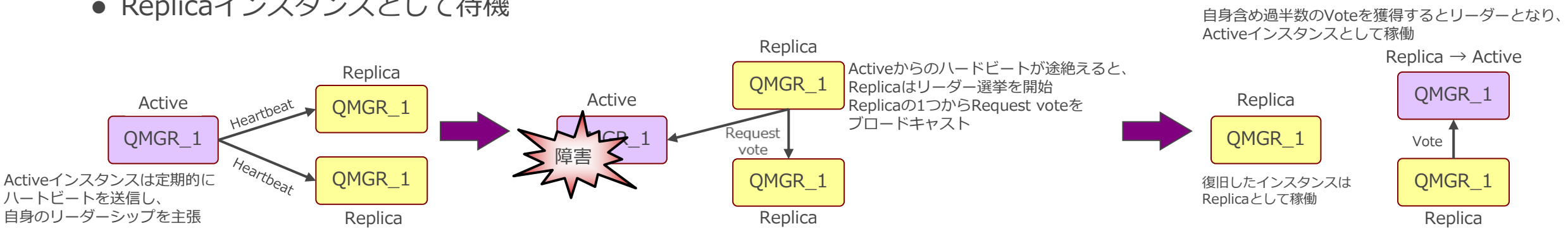
- オンプレミス環境およびクラウド環境で外部ディスクを使用せずにフェイルオーバーを実現する機能
  - ◆ Linux RHELプラットフォームで構成可能、ただしAdvancedライセンスが必要
  - ◆ 以下3種類の構成が可能
    - HA RDQM：高可用性 (HA) グループとしてクォーラム構成された3つのサーバーで構成
      - キュー・マネージャーを実行しているノードの障害発生をPacemakerが検知すると、別ノードでキュー・マネージャー・インスタンスが自動的に開始
    - DR RDQM：本番サイトと災対サイトで稼働する1ペアのサーバーで構成
      - 障害時、DRBDがデータの複製を行う
    - DR/HA RDQM：HA RDQMとDR RDQMの機能を統合
      - DR/HA RDQMが実行されているノードで障害が発生すると、自動的にそのHAグループ内の別のノードにフェイルオーバー
      - 本番サイトで障害が発生した場合には、災対サイトのプライマリー・ノードでRDQM キュー・マネージャーを手動で開始する



Native HA(参考)

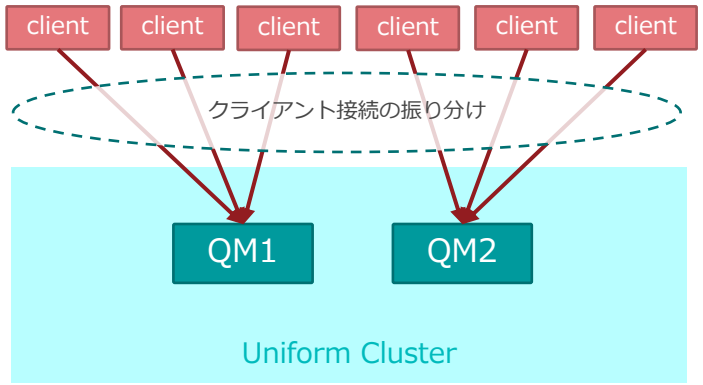
- MQ独自のログ・レプリケーションでフェイルオーバーを実現する機能
  - Kubernetes/OpenShift環境で構成可能
  - Cloud Pak for IntegrationもしくはMQ Advancedのライセンスが必要
  - キュー・マネージャーは3つのPodで稼働
    - 1つがActiveインスタンスとして稼働、残りはReplicaインスタンスとして待機
  - 各PodがPersistent Volume上にログのコピーを保有
    - キュー・マネージャーのログはレプリケートされているため、構成やメッセージは引き継がれる
  - 障害時にフェイルオーバーしたキュー・マネージャーへの接続切り替えはKubernetes/OpenShiftのServiceで解決
    - 接続元のチャネルやクライアントは1つの宛先に対して再接続するだけでよい
  - 追加のクラスタ製品やストレージの冗長構成が不要
  - 障害QMGR/Podはセルフ・ヒーリング機能で同一もしくは別ノードで再起動
    - Replicaインスタンスとして待機

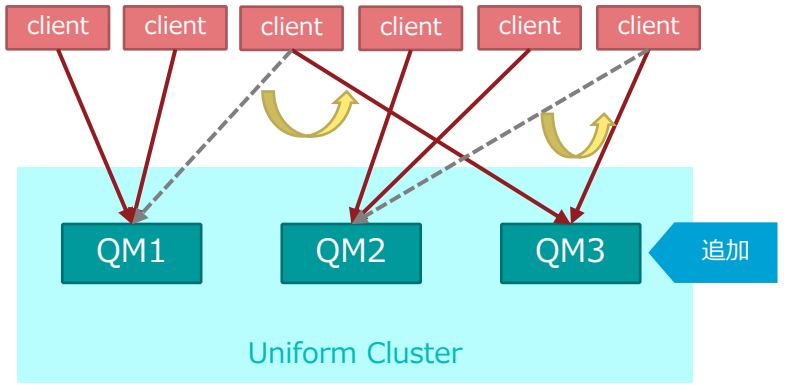




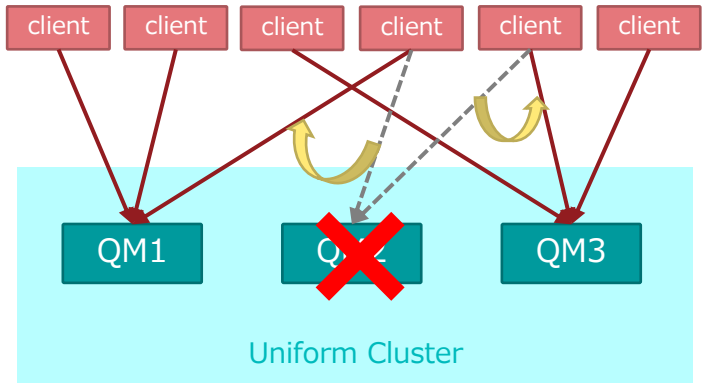
Uniform Cluster

- MQクライアント接続を均等に割り振る機能
  - ◆ 複数のキュー・マネージャーをグルーピングしてクラスターを構成
  - ◆ クラウド環境の多重化されたMQ アプリケーションに高い可用性を提供
  - ◆ MQクライアント接続の自動リバランスを実現
  - ◆ クライアント・アプリケーションからの接続をUniform Cluster内のキューマネージャーに均等に分散
  - ◆ キュー・マネージャーやクライアント・アプリケーションが増減した場合にも自動的に対応
    - ● ただし、CCDTの変更は必要
  - ◆ IBM MQ for z/OSでは使用不可





スケールアウト時も自動的に接続の振り分け



キューマネージャー障害時も自動接続振り分け

その他の主な機能

- メッセージの優先処理
  - ◆ メッセージにプライオリティをつけて優先制御することが可能．デフォルトはFIFOで処理
- トリガー機能
  - ◆ キューにメッセージがPUTされたのをトリガーに特定のプログラムを起動
- 文字コード変換
  - ◆ メッセージが全て文字型の場合
    - MQの標準機能で変換可能
  - ◆ メッセージがバイナリ、文字型/バイナリ混在の場合
    - データ変換EXITモジュールを作成すれば、MQが適切なタイミングでEXITを呼び出す
- 2フェーズ・コミット
  - ◆ XAプロトコルで、MQ資源の更新とデータベース更新を1つのUOWで処理
    - キュー・マネージャーがトランザクション・コーディネータとして稼動 … キュー・マネージャー・コーディネーション
    - キュー・マネージャーがリソース・マネージャーとして稼動 … 外部コーディネーション

他

MQの管理機能

■ コマンドインターフェース

◆ runmqscコマンド

- 対話型にMQオブジェクトの管理が可能
  - キュー、チャネルの作成/削除、属性変更など
- 管理の変更は動的に反映される
  （キューマネージャーの再起動は不要）

```
C:¥>runmqsc QMA
5724-H72 (C) Copyright IBM Corp. 1994, 2022.
キュー・マネージャー QMA に対して MQSC を始動中です。

display ql(QL01) curdepth
     1 : display ql(QL01) curdepth
AMQ8409I: キューの内容を表示します。
   QUEUE(QL01)                             TYPE(QLOCAL)
   CURDEPTH(0)
```

キュー内のメッセージ数を表示

◆ 管理REST API

- REST APIでMQオブジェクトの管理が可能
  - キュー、チャネルの作成/削除、属性変更など
- REST APIを利用するアプリケーションとの連携が容易
- データ・フォーマットはJSON

```
HTTPS GET
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QMA/q
ueue/QL01?status=status.currentDepth
{"queue": [{
  "name": "QL01",
  "type": "local",
  "status": {"currentDepth": 0}
}]}
```

キュー内のメッセージ数を表示

MQの管理機能

- IBM MQコンソール
  - ブラウザでキューマネージャー、MQオブジェクトの管理が可能なGUIツールを提供
    - IBM MQ Explorerは、Fix CentralからスタンドアロンIBM MQ Explorerをインストールすることで使用可能
  - MQオブジェクトの管理だけでなく、メッセージの読み書きも可能
  - リモートのキューマネージャーに接続し管理することも可能
  - クライアント側へのコンポーネントのインストールが不要

ホーム画面

キューの管理画面

MQの管理機能

- 会計 / 統計レポート
  - キャパシティ・プランニングなどに利用可能な会計&統計レポートの出力をサポート
  - 会計情報
    - MQアプリケーション単位にMQCONN～MQDISCの詳細な活動情報が取得可能
  - 統計情報
    - キュー・マネージャーの一定時間毎の詳細な活動情報が取得可能

種別	説明
MQI会計	アプリケーションがMQCONN～MQDISCまでに発行したMQIコールの種類/回数、他
キュー会計	アプリケーションがMQCONN～MQDISCまでに個々のキューにPUT/GETした回数/総バイト数、他
MQI統計	計測インターバル内にキュー・マネージャー全体で発行されたMQIコールの種類/回数、他
キュー統計	計測インターバル内に個々のキューで発生したメッセージ滞留数(最小/最大)、滞留時間の平均、他
チャネル統計	計測インターバル内に個々のチャネルが転送したメッセージ数/総バイト数、1バッチの平均、メッセージ数、他

MQの管理機能

■ セキュリティ機能

◆ MQオブジェクトへのアクセス制御

- OSのユーザ・グループで制御
  - MQ管理権限を持つmqmグループ（ Windowsの場合はAdministratorsも）にユーザーを所属
    - 全てのMQオブジェクトに対する操作権限を持つ
- ユーザー、またはグループ単位にMQのセキュリティ管理機能（OAM)を使用してMQオブジェクトに対する権限を付与
  - MQオブジェクト単位、API単位に権限の許可を設定することが可能
  - setmqautコマンドまたは、PCFコマンドを使用して、権限を登録
  - キュー・マネージャーへのCONNECT時、オブジェクトのOPEN時にセキュリティチェックを実施
    - 通常はチャネルのMCAUSER属性の値か、プログラムの実行ユーザーIDでチェック
    - 代替ユーザーIDを使用して、実行ユーザー以外のユーザーIDでのチェックも可能
- LDAPユーザーやOSに存在しないユーザーに対しても権限付与が可能

◆ チャネルの接続認証

- チャネル認証
  - チャネル認証レコードを使用して、チャネル・レベルで接続システムへのアクセスを制御
- 接続認証
  - MQのユーザー認証機能
  - MQ接続(ローカル/クライアント)時に、アプリケーションがユーザーID/パスワードを提供
  - キューマネージャーはリポジトリーを使用して提供された組み合わせをチェック
- MQクライアント接続の場合には併用可能

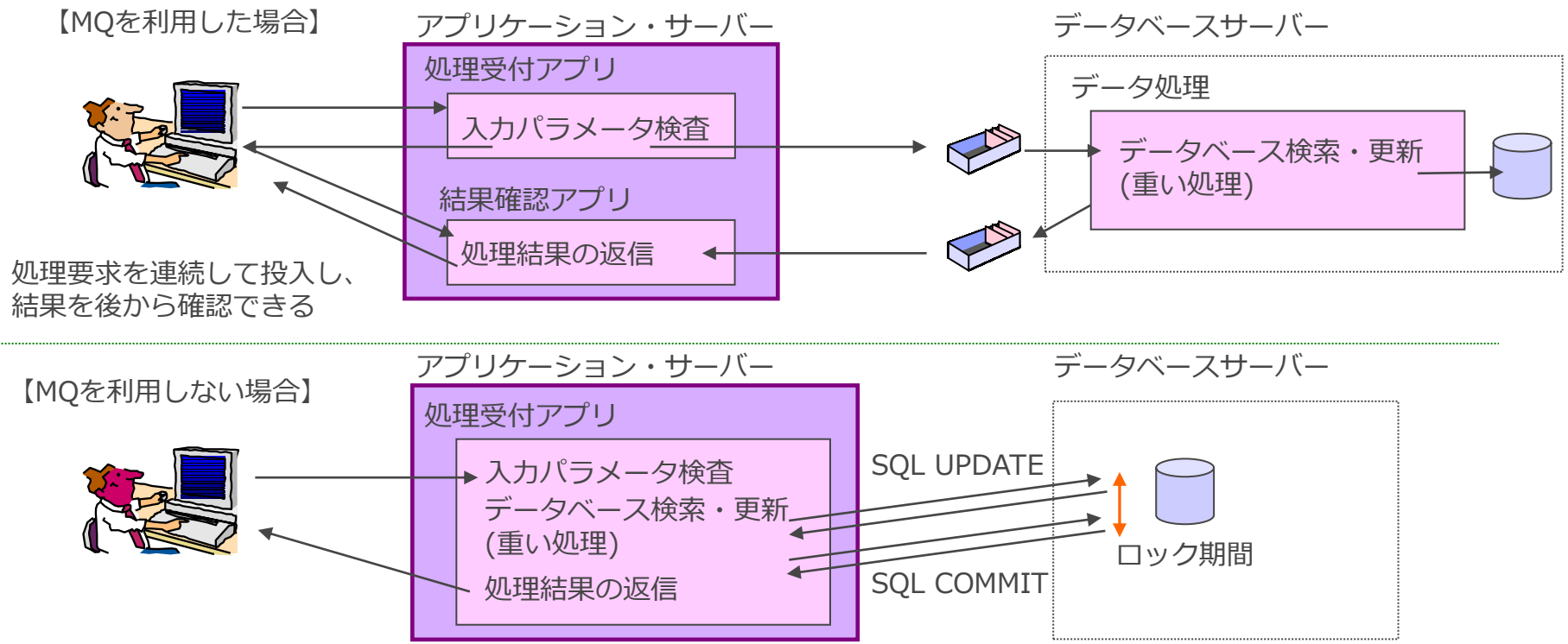
MQの管理機能

- セキュリティ機能（続き）
  - ◆ 通信データの保護（認証 / 暗号化）
    - チャネルのSSL/TLSサポート
      - すべてのタイプのチャネルで使用可能
      - 通信時の認証、暗号化を実現
    - ユーザーEXIT
      - チャネル接続時の認証（セキュリティEXIT）やメッセージの暗号化/複合化（メッセージEXIT）をEXITを実装して実現することも可能

適用例

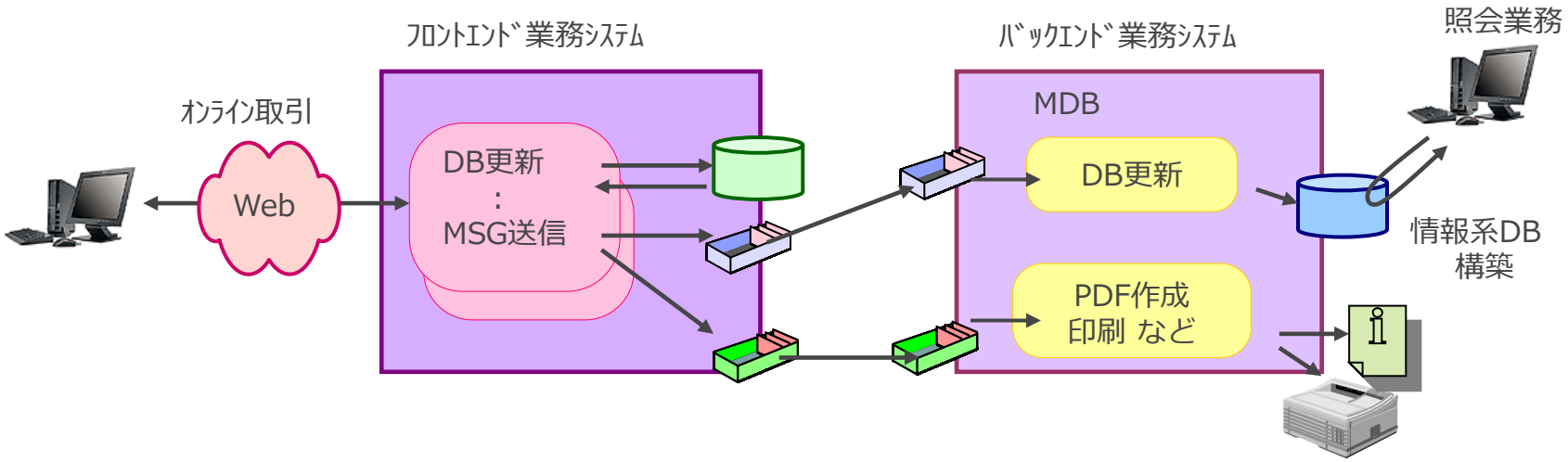
適用例（長時間かかる処理）

- 大量のデータベース検索、更新処理など
  - ◆ リアルタイムで応答を必要としない場合
    - 要求を受付けた時点でクライアントに応答を返し、後から結果を確認してもらう
  - ◆ クライアント数、トランザクション数が多い環境で有用
    - アプリケーション・サーバーへの同時接続数を抑え、少ないリソースで多数のクライアントにサービスを提供
  - ◆ データベースのロック期間を短くできるので、ロック競合の発生を抑えられる



適用例（ディレード処理）

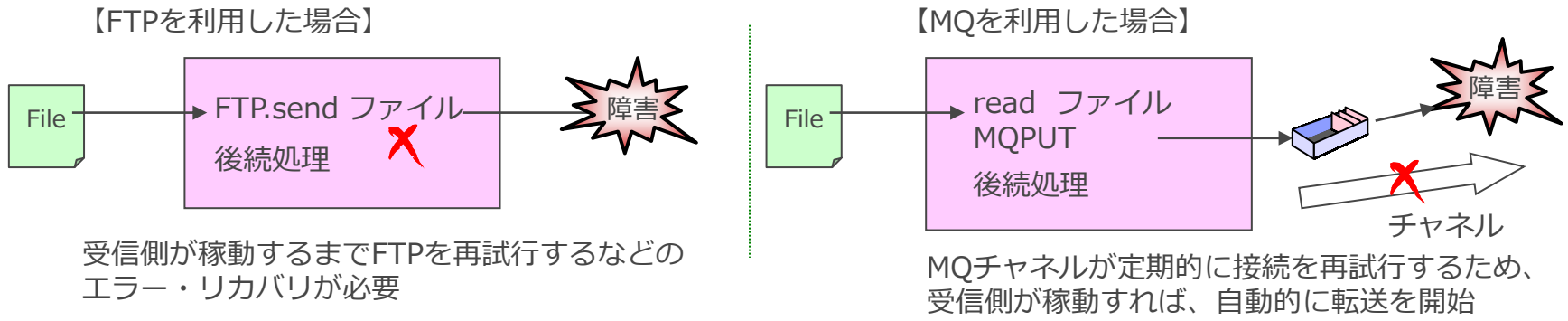
- オンライン取引情報から即時、情報系データの構築
  - ユーザ取引ログを分類してメッセージとして出力し、バックエンド業務システムに送信
  - バックエンド業務システムで、情報系DBの構築やドキュメントの作成、印刷などを実行
  - 一方向型のメッセージ送信
  - システム構成変更にも即時対応可能となる
    - キュー構成を変更するだけで、アプリケーションの変更は不要
    - 負荷状況に対応した、バックエンド業務システムの構成変更が容易
    - バックエンド・システムを再構築してもフロント・システムの変更は不要



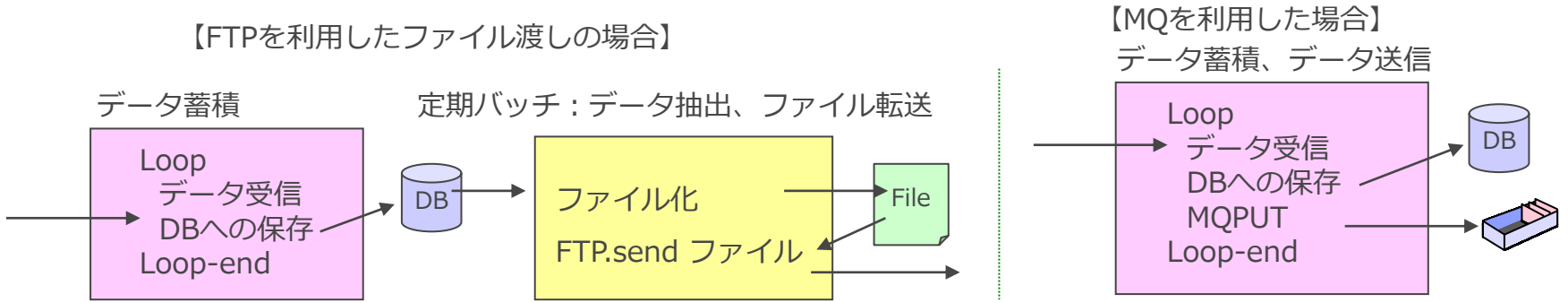
適用例（ファイル転送）

ファイル転送業務

- 一方向型、非同期通信の典型例
- 受信側システムが稼動していなくても、送信側の処理は完了し、後続処理の実行が可能
- 受信側システムの稼動を待って、MQが転送を実行
  - アプリケーションに特別なエラー・リカバリが不要



ファイル転送業務のオンライン化

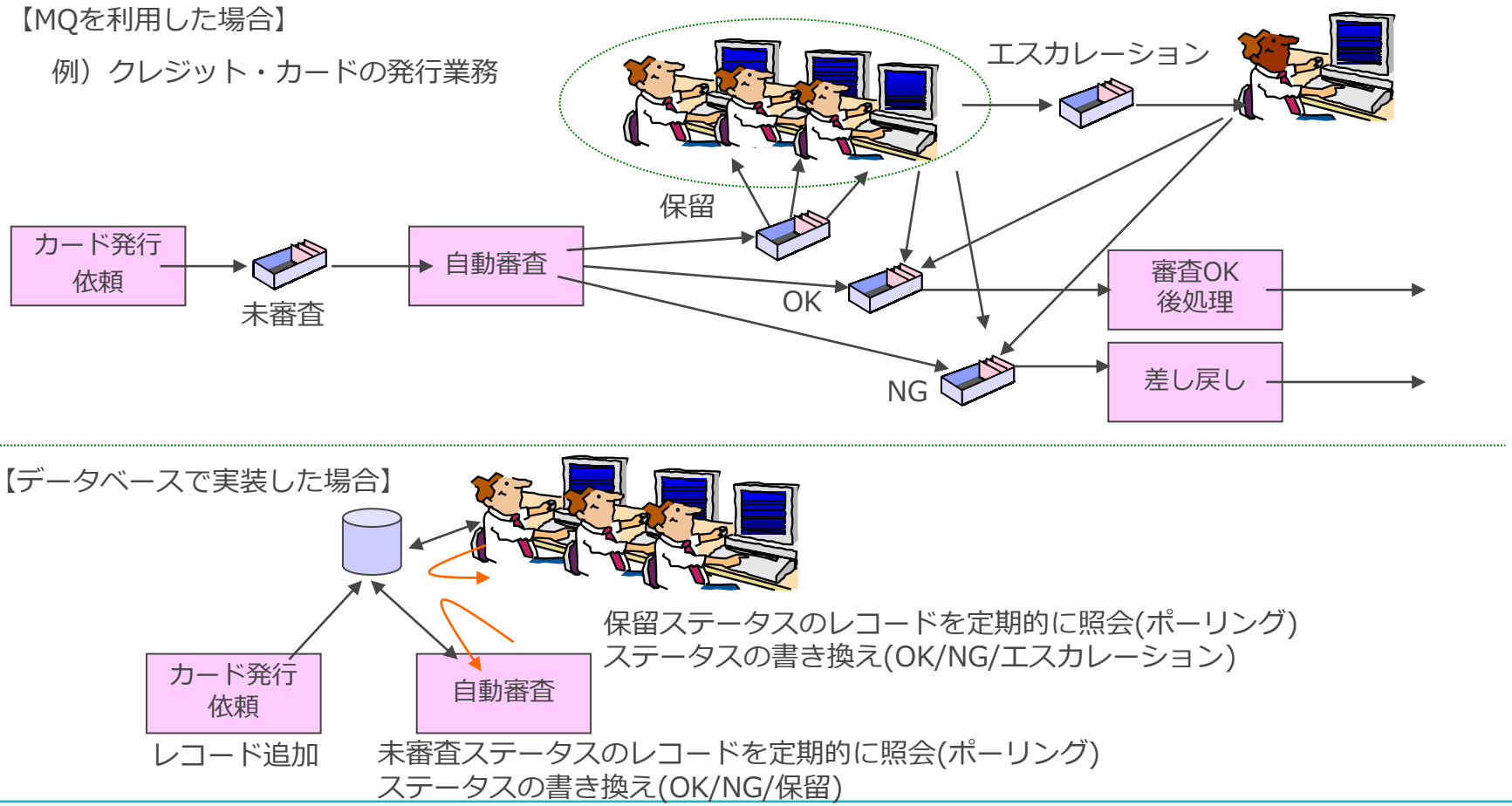


適用例（ワークフロー業務）

■ ワークフロー業務

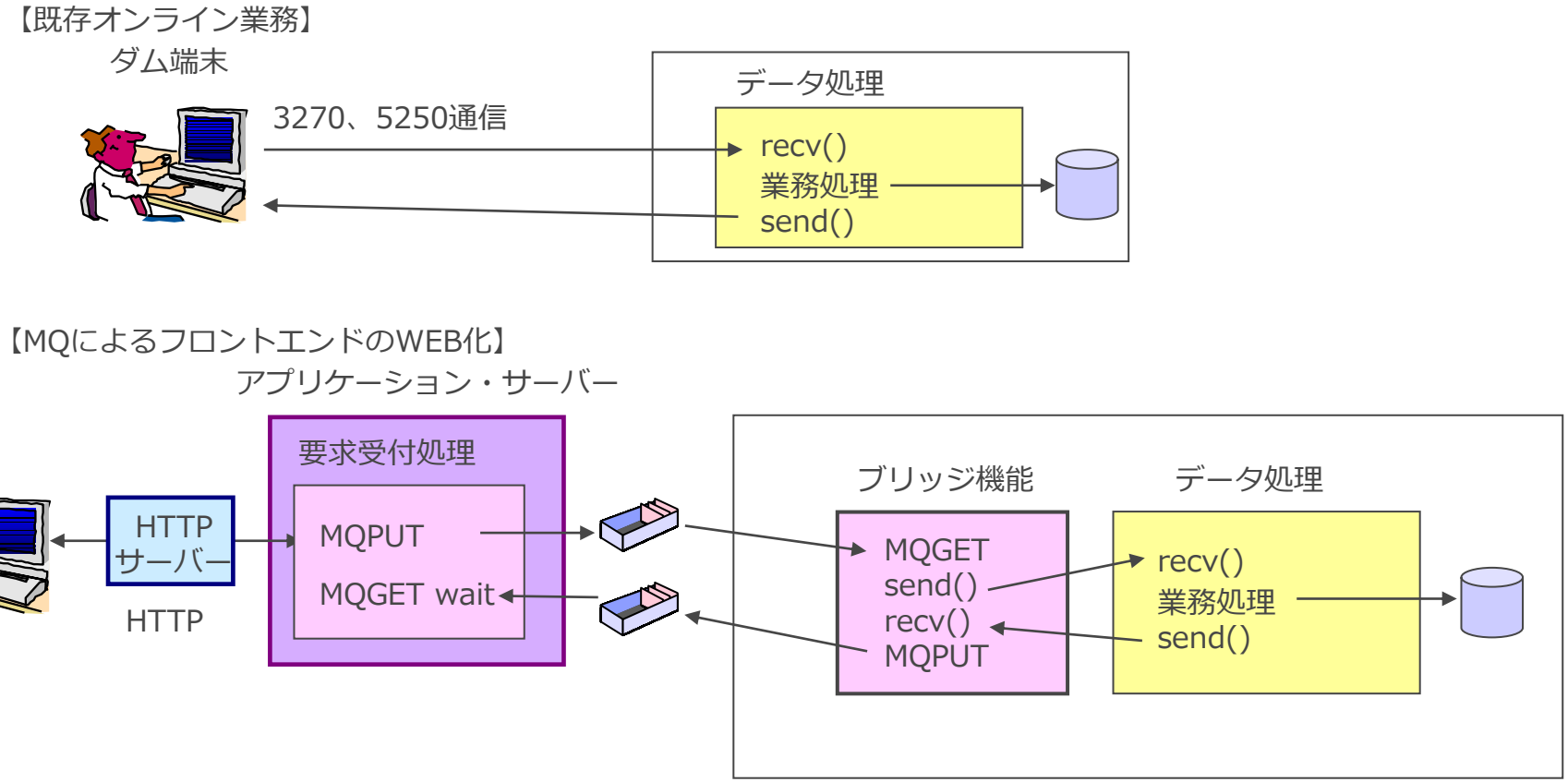
◆ 典型的な非同期処理の業務のため、非同期通信のMQでの実装が容易

- データベースでの実装も考えられるが、ポーリングによる負荷、ロック競合などが心配・・・



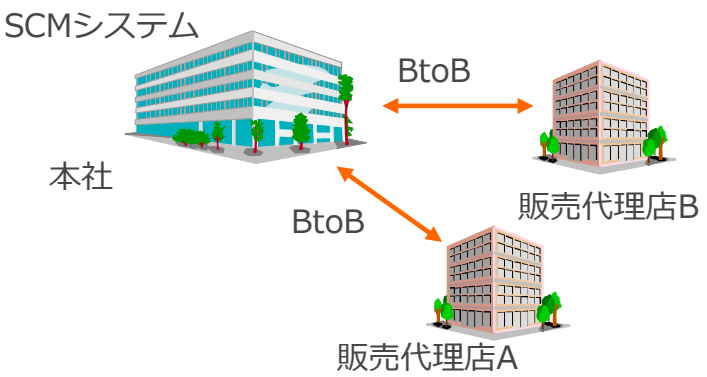
適用例（既存システムのWEB化）

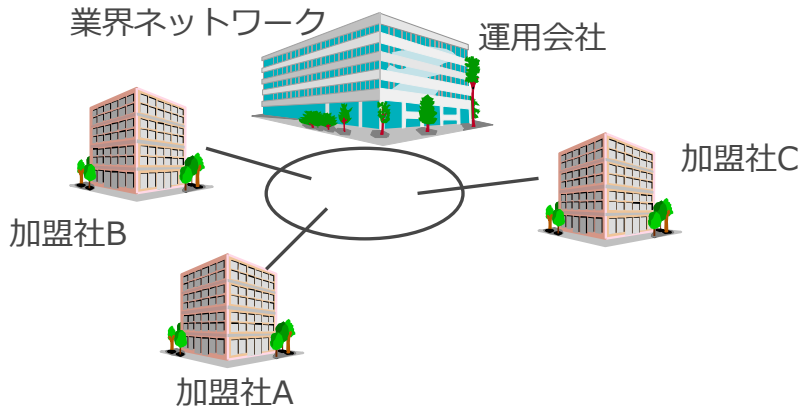
- バックエンドの既存オンラインシステムを有効利用しながら、フロントエンドをWeb化
  - ◆ 既存システムとの間をMQで接続する
  - ◆ 既存システム側で、MQ通信を行うブリッジ・アプリケーションの開発を行う



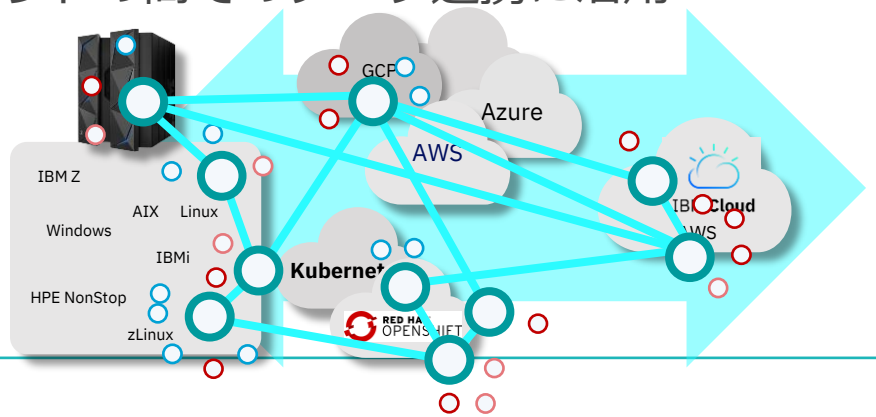
適用例（他システムとの連携）

- BtoB、業界ネットワーク
  - ◆ MQを使ってシステム間を疎結合することで、システム間の相互影響を最小化
  - ◆ App Connect EnterpriseなどのESB製品と連携して、メッセージ変換などの要件に対応することも可能





- ハイブリッド・クラウド統合
  - ◆ オンプレミスと各種クラウドの間でのデータ連携に活用



<参考> MQの役立つリンク集

- MQ v9.3 製品マニュアル
  - https://www.ibm.com/docs/en/ibm-mq/9.3
- 前提SW/HWの確認サイト
  - https://www.ibm.com/support/pages/node/318077
- サポート・パックの提供サイト
  - https://www.ibm.com/support/pages/ibm-mq-supportpacs-product
- Fix のダウンロード・サイト
  - https://www.ibm.com/support/pages/recommended-fixes-ibm-mq