情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2010-DPS-145 No.5
Vol.2010-GN-77 No.5
Vol.2010-EIP-50 No.5
2010/11/25

オープンソースメッセージ基盤の評価

鈴木 芳生† 花井 知広† 佐藤 竜也†

近年，開発生産性の観点から Service-Oriented Architecture(SOA)が注目されているが，SOA では AP やサービスを非同期連携させるためのメッセージ基盤(MOM)が重要となる．今後，クラウド利用により外部サービスとの連携が進むと，相互接続性や低コスト化が重要となる．そのため，米国金融機関の主導で標準化が進められている標準プロトコル Advanced Message Queuing Protocol(AMQP)を採用した Open Source Software(OSS)実装 MOM の評価を行った．その結果，PC サーバを利用した評価環境において，AMQP の OSS 実装の一つである Qpid(C++)はメッセージサイズが 1KB の時に，秒間 45 万件の転送スループットとなり，他の OSS 実装の MOM に対して 2.6 倍の性能が得られる事を確認した．

Evaluation of Open Source Implementation of Message Oriented Middleware

Yoshio Suzuki,† Tomohiro Hanai† and Tatsuya Sato†

Message oriented middleware (MOM) for transferring application messages between systems is an important functionality in event-driven Service-Oriented Architecture (SOA). Moreover, interoperability and cost-effectiveness of MOM is becoming increasingly important in cloud computing era. In this paper, we evaluate Open Source Software (OSS) implementation of Advanced Message Queuing Protocol (AMQP) which is an open standard application layer protocol for messaging. Experimental results shows that Qpid(C++) which is an OSS implementation of AMQP has achieved message transfer rate of 45,000 mps(message per second) and outperformed another OSS implementation by 2.6 times for 1KB message size in our experimental environment using PC servers.

1. はじめに

近年，開発生産性の向上や運用の柔軟性等の観点から Service-Oriented Architecture (以下 SOA)の注目度が高まっており 1), Credit Suisse など海外の金融機関においても SOA 導入事例が報告されている 2)3). また米国調査会社の Gartner では，クライアントがサービスを明示的に呼び出す従来型の SOA に対して, イベントによってサービスが呼び出されるイベント駆動型の SOA の重要度が高まるとしている 4). イベント駆動型 SOA では，イベント，すなわちメッセージによってサービスが起動されるため，イベント発信元とサービス，あるいはサービス間をつなぐメッセージ基盤(以下，MOM)の重要性がますます高まる．金融システムなどミッションクリティカル分野向け MOM は，従来から特に高い性能や信頼性が求められてきているが，これに加えてそれぞれ独立に構築された拠点システム間，あるいは他社のシステムとの接続・統合のしやすさも重要となってきている．

MOM は，様々な分野で利用されているが，現在はプロプライエタリな製品が多く利用されている．しかし，これらの製品間には相互運用性が保証されていないという問題がある．SOA 化やクラウド化の流れを受けて多様なサービスや大量の計算機，デバイスなどの連携が必要となる次世代システムの基盤技術としては，相互運用性の重要性がますます高まると考えられる．

これらの問題点の解決のためにメッセージングの標準仕様が望まれており，米国ではユーザ企業である大手金融機関が主導して，非同期メッセージングのための標準プロトコル Advanced Message Queuing Protocol (以下, AMQP)6)の策定が進められている．また，標準プロトコルに従ったオープンソースソフトウェア(以下，OSS)実装もリリースされ始めており，相互接続性の保証されたシステムを安価に構築可能とする技術として注目を集めている．2008 年の金融危機以降，構築・運用コストを削減する技術の重要性が増しており，AMQP の OSS 実装に注目が集まっている．しかしながら，金融などのミッションクリティカル分野では秒間数千件以上のスループットが要求される自動取引への対応など，性能や信頼性に高い要件が求められるため，OSS 技術の適用にあたっては事前の検証が必須である．

そこで本研究では，標準仕様に基づいた有力な MOM に関して，機能と基本性能の評価を行なうことを目的とした．

† 株式会社日立製作所中央研究所
Central Research Laboratory, Hitachi,Ltd.

Vol.2010-DPS-145 No.5
Vol.2010-GN-77 No.5
Vol.2010-EIP-50 No.5
2010/11/25

2. メッセージ基盤の要件と標準技術の適用可能性

情報システムの SOA 化やクラウドコンピューティングの普及により，企業内/企業間でのシステム連携が加速度的に進展することが予想され，同時に標準仕様に基づいた安価なメッセージ基盤(MOM)の重要性が高まると考えられる．そこで，本章ではミッションクリティカル分野での要件を考慮に入れた上で，標準仕様に基づくメッセージ基盤と従来技術との比較を行い，詳細評価の対象とする MOM を選定する．

最初に 2.1 節で MOM に求められる要件とその要件に適したメッセージングアーキテクチャについて議論する．次に 2.2 節では MOM の各種プロトコルの利害得失を整理し，詳細評価対象の MOM を選定する．

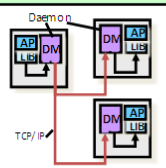
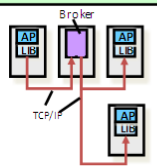
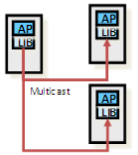
2.1 メッセージング基盤の要件と代表的なメッセージングアーキテクチャ

本節では，ミッションクリティカル分野における MOM に求められる要件に対し，最適なメッセージングアーキテクチャを検討する．表 1 にメッセージング基盤に求められる要件と代表的なメッセージングアーキテクチャを示す．メッセージングアーキテクチャとしては，Peer-to-Peer (以下，P2P)型，Broker 型，Reliable Multicast (以下，RM)型の 3 種類がある．

P2P 型は，送受信サーバにデーモンプログラムが動作する方式であり，デーモンを介して送信サーバと受信サーバが直接送受信処理を行う．これに対し，Broker 型では送受信サーバは Broker を介して通信を行う．すなわち，送信側 AP が送信したメッセージを Broker で一旦受信し，Broker から受信側 AP に配信を行う．最後に，RM 型では送信側の AP からマルチキャスト(同報通信)により受信側 AP に配信処理を行う．P2P 型/Broker 型とも TCP/IP で通信を行うのに対し，RM 型では多重構成での性能向上実現のため UDP マルチキャストを利用する．

性能面では，UDP マルチキャストを利用する RM 型がスループットおよびレイテンシの面で最も有利である．一方 P2P 型は，送信側と受信側が直接通信を行うためスループット性能は高くなるが，デーモンプログラムを介する分レイテンシが大きくなる．同様に，Broker 型は全てのメッセージが Broker を経由するためレイテンシが大きくなる．次にスケーラビリティについては，1 台の送信者(Publisher)に対して複数の受信者(Subscriber)が存在する Pub/Sub 構成時において，Subscriber 数増大時の Publisher の負荷を考慮する必要がある．RM 型では Subscriber が増加した場合にも UDP マルチキャストを利用して効率的にメッセージを配信できるのに対し，TCP/IP を利用する P2P 型および Broker 型では Subscriber 毎に送信処理を行う必要があり，Subscriber 数増大に応じて送信元や Broker の負荷が増大するため，ボトルネックが発生しやすい．

表 1　代表的なメッセージングアーキテクチャ

		Peer-to-Peer (P2P)	Broker	Reliable Multicast (RM)
概要				
性能	スループット	○ (送受信サーバが直接通信)	△ (全メッセージがBrokerを経由)	○ (送受信APが直接通信)
	レイテンシ	△ (ﾃﾞｰﾓﾝﾌﾟﾛｸﾞﾗﾑを経由)	△ (Brokerを経由)	○ (送受信APが直接通信)
スケーラビリティ		△ (受信サーバ毎に送信処理)	△ (受信サーバ毎に送信処理)	○ (UDPマルチキャスト)
非同期メッセージング		△ (ﾃﾞｰﾓﾝﾌﾟﾛｸﾞﾗﾑが動作していれば可)	○ (Publisher/Subscriberを分離)	× (送受信APが同時動作要)
運用容易性 開発容易性		× (分散管理)	○ (統合管理)	× (分散管理)

スループットとレイテンシ，およびスケーラビリティの評価においては，RM 型は非常に好ましい特性を保持していることがわかる．しかしながら，MOM をビジネス適用する場合には，これらに加えて非同期型のメッセージングのサポートが重要となる．特にイベント駆動型 SOA では，イベントの発生元とサービスを提供するシステムとが分離されており，その間を MOM で接続する形となっている．すなわち，イベント発生元とサービスを提供するシステムの非同期な動作が必要となる．金融分野への適用を考えた場合においても，例えば自動取引などではイベント駆動型，すなわち非同期型の処理が必須となる．

P2P 型および Broker 型においては，デーモンプログラムや Broker でのキューイング制御によって，非同期型メッセージングへの対応は容易であるが，UDP マルチキャストにより AP 同士が直接通信を行う RM 型では，非同期型メッセージングへの対応は困難である．その理由は，RM 型では UDP マルチキャストにより送信側 AP から受信側 AP へ直接メッセージを送信しており，メッセージの送受信においては送信側 AP と受信側 AP が同時に協調動作しなければならないためである．これに対して P2P 型や Broker 型では，たとえ一部の Subscriber が障害により一時的にダウンしていた場合でも，その Subscriber に対するメッセージをキューイングしておき Subscriber 復旧時にメッセージの再送処理を行うことで非同期型のメッセージングの実現が可能である．運用管理や AP 開発の観点からは，Broker 型では集中/統合管理が可能かつ柔軟性が高いのに対し，P2P 型や RM 型では送受信側を個別に管理/制御する必要があり，運用管理コストおよび AP 開発コストが高くなることがわかる．

Vol.2010-DPS-145 No.5
Vol.2010-GN-77 No.5
Vol.2010-EIP-50 No.5
2010/11/25

以上の検討結果をまとめると，(1)性能に関しては RM 型が優れる，(2)非同期メッセージング対応は P2P と Broker 型は比較的容易なのに対し RM 型では対応が困難，(3)運用管理，AP 開発コストは集中管理が可能な Broker 型が優れることがわかった．ビジネス適用を考えた場合には, (2)の非同期メッセージング対応, および(3)の運用管理, AP 開発コストの削減は必須であり，これら観点を踏まえて以降では Broker 型を対象とした検討を行う．

2.2 Broker 型メッセージングのプロトコル比較

2.1 節でメッセージングのアーキテクチャとしては Broker 型が有力と述べた．同様に，MOM のクラウドコンピューティングに代表される大規模なシステムへの適用を考えた場合には，相互接続性の確保，システム構築，運用管理コストの削減が重要と述べた．相互接続性の確保，システム構築，運用管理コストの削減を実現するにあたり有力な候補として，標準化されたプロトコルと，その OSS 実装が挙げられる．そこで以降では，相互接続性を保証した標準仕様，およびその OSS 実装の MOM 製品の比較を行う．なお標準仕様としては，米国金融機関の主導により策定が進められている Advanced Message Queuing Protocol (以下，AMQP) **6)**，および，Web サービスの標準仕様として策定されている WS-Reliability 8)および WS-ReliableMessaging **9)** (合わせて以下 WS-*)を，そして OSS 実装の MOM としては，ビジネス分野への適用例も報告されている ActiveMQ **10)**を選択した．

表 2 にこれらの特徴をまとめた．まず相互接続性については，AMQP や WS-*は標準仕様のため，相互接続性が保証されている．一方，ActiveMQ については OSS 実装であるため接続仕様は公開されているが，仕様は厳密には規定されていないこと，そしてその策定プロセスも不透明であることから，相互接続性の保証はない．次にシステム構築, 運用管理コストについては OSS 実装が存在する AMQP や ActiveMQ はライセンスコストが発生する商用製品に対して優位である．性能については，商用製品や ActiveMQ は実システムでの実績が多数あること，また，商用製品では 40,000 メッセージ/秒以上の処理性能結果が報告もされており 11)，ミッションクリティカルシステムへの適用に対する十分な性能が得られていると考えられる．一方，WS-*については XML を用いたプロトコルであり，メッセージ送受信時に XML 解析処理などが必要となる．例えば，金融システムでは，自動取引などにより処理の自動化や小口化が進んだ結果，取り扱うべきデータ量が増大している 12)．WS-*は開発生産性が高く，また様々な SOA 製品でサポートされているというメリットはあるが, このようにハイエンドな性能が要求される場合には適さないと考えられる．

以上をまとめると，相互接続性およびコストの観点で AMQP が有力である．しかしながら，実案件適用の事例が少ないため，性能面で十分な能力を有するか否かは不明であった．そこで本研究では，性能面においてミッションクリティカル分野へ AMQP の適用が可能かを検証するために，性能評価を実施した．

表 2　Broker 型メッセージングの特徴比較

仕様・製品	AMQP	Apache ActiveMQ	WS-* (WS-R,WS-RM 等)
相互接続性	○ 標準仕様 (策定中)	△ 仕様はオープンだが未保証	○ 標準仕様
構築，運用管理コスト	○ (OSS 実装有)	○ (OSS 実装)	○ (OSS 実装有)
性能	不明 (4 章にて評価)	不明 (4 章にて評価)	× (XML ベースのため)

3. Advanced Message Queuing Protocol (AMQP)の概要

2 章での検討より，相互接続性とコスト，および非同期メッセージングへの対応といった観点から，標準仕様である AMQP が有望であることが明らかとなった．そこで本章では，最初に AMQP の仕様策定の背景を説明した後，AMQP の仕様を説明する．最後に，AMQP を実装する複数の OSS 実装について，管理機能の有無やメンテナンスの容易性などの観点で比較を行う．

3.1 AMQP 提案の背景

自動取引など処理の自動化や小口化が進んだ結果，金融業界が扱うべきデータは増大しており 12)，金融業界ではこれらのデータをサービスやアプリケーション間の間でやりとりするための高速な MOM が必要とされている．そのためこれまでも，各金融機関においては独自の MOM の開発が行われてきた．しかしながら，独自の MOM の持続的な開発は困難かつ非効率であることなどから, JPMorgan Chase のオハラ氏は，非同期メッセージングにおいても，TCP/IP のような標準プロトコルが必要と考え，標準プロトコル AMQP の策定に着手した 13).

AMQP は，2006 年に金融機関および IT ベンダによってワーキンググループ(WG)が結成され，現在でも仕様の検討および標準化作業が行われている．2010 年 9 月時点で AMQP WG には表 3 に示す数多くの有力企業やベンダが参加している．通常，プロトコルの標準規格の策定は IT ベンダが主導して行われることが多いが, AMQP WG の特徴としては JPMorgan Chase 等のユーザ企業が積極的に主導して標準化作業が行われている点が挙げられる．また，IT ベンダとしては，従来から Red Hat，Cisco 等の大手ベンダが参加しており, 2008 年 10 月には Microsoft が参加を表明し注目を集めた. 2008 年以降も新たな金融機関やベンダの WG への参加表明が行われており，金融機関では Barclays Bank (2009 年 6 月参加)，Bank of America (同 2009 年 11 月)が，IT ベンダとし

Vol.2010-DPS-145 No.5
Vol.2010-GN-77 No.5
Vol.2010-EIP-50 No.5
2010/11/25

ては Tervela (同 2009 年 2 月), INETCO (同 2010 年 1 月), Software AG, VMware (同 2010 年 6 月)などが参加を表明している．

2010 年 3 月現在，AMQP WG では AMQP 1.0 仕様の策定作業が行われており，Recommendation 版仕様が公開されている 14)．1.0 仕様の策定プロセスとしては，上記の仕様に対して 2 種類以上の製品実装を行い，それらの実装によって仕様の有効性や相互接続性が確認された後に，最終的な AMQP 1.0 仕様(Standard 版と呼ばれる)が決定されるプロセスとなっている．AMQP の策定は金融機関が主導しているが，適用領域としては金融システムのみに限定されるわけではなく，より広く AP 間の非同期連携を実現する標準基盤を目指している．SOA の基盤としても利用可能なことから，Gartner の報告においてもイベント駆動型 SOA の注目技術の一つとして挙げられている 15)．

表 3　AMQP WG 参加団体(2010/09 時点)

銀行・証券系	IT ベンダ系
Bank of America	Cisco Systems
Barclays Bank	INETCO Systems
Credit Suisse	Informatica
Deutsche Borse Systems	Microsoft
Goldman Sachs	Novell
JPMorgan Chase Bank	Progress Software
	Rabbit Technologies
その他(業界団体)	Red Hat
TWIST Process Innovations	Software AG
	Solace Systems
	Tervela
	VMware
	WSO2
	29West

3.2 AMQP 仕様の概要と特徴

本節では AMQP 仕様の概要および特徴について述べる．以下の内容は，AMQP 1.0 Specification Draft Public Review 2 14)の内容に基づいている．

図 1 に AMQP のプロトコルスタックを示す．AMQP では，(a)管理層，(b)メッセージングモデル，(c)セッション層，および(d)トランスポート層の仕様が定義されている．AP との API は定義されていないが，JMS のモデルに合致するよう設計されているため，AMQP 実装プロダクトが JMS の API を提供することは容易である．

従来の MOM では，独自のメッセージフォーマットやプロトコルを利用しているため，JMS API を利用した AP であっても，異なる製品の Broker に接続することができなかった．すなわち，図 2(a)に示すように，A 社の MOM 向けに開発した AP は B 社の MOM に接続することはできなかった．これに対し，AMQP 実装では共通メッセージフォーマット，共通プロトコルが利用されるため，図 2(b)に示すように異なる実装の Broker にも容易に接続すること可能となる．以下では，AMQP のメッセージングモデル(図 2(b))，メッセージングの形態の概要を説明する．

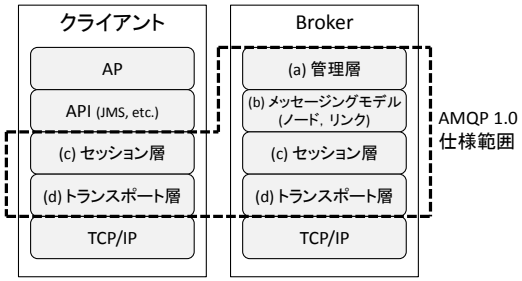


図 1　AMQP のプロトコルスタック

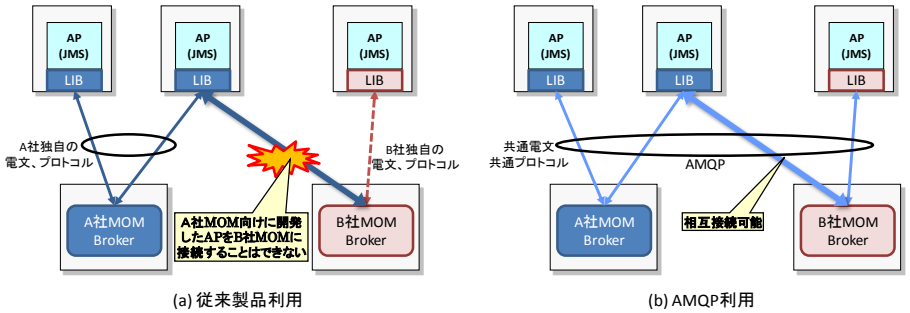


図 2　AMQP の相互接続性

図 3 に AMQP のメッセージングモデルを示す．AMQP のメッセージングモデルでは，AP へ提供するメッセージングのセマンティクスが定義される．AMQP のメッセージ配信は，ノードとリンクの概念で構成される．ノードはメッセージの中継点であり，ノード間はリンクによって接続される．メッセージはノードからノードへ，リンクをたどることによって転送される．Broker 中のノードの内部にはキューがあり，このキューを用いることで非同期メッセージングが可能となる．ノードとリンクを構成することで，メッセージの柔軟なルーティングが可能となる．

Vol.2010-DPS-145 No.5
Vol.2010-GN-77 No.5
Vol.2010-EIP-50 No.5
2010/11/25

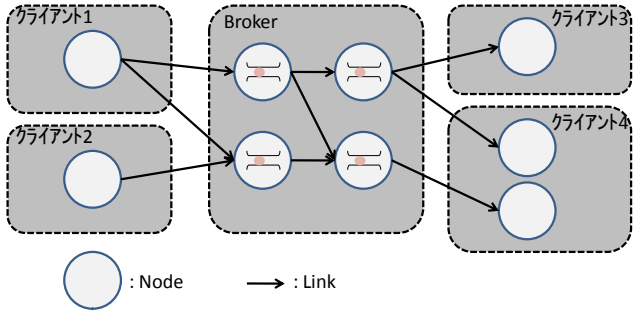


図 3　AMQP のメッセージングモデル

次に，ノードとリンクにより構成される AMQP の代表的なメッセージングの形態を図 4 に示す．図 4 の(a)(b)は送信側 AP (クライアント 1)が送信したメッセージ(図中①～④)を，いずれか 1 つの受信側 AP (クライアント 2 もしくは 3)が受信する形態である．図 4 (a)では受信 AP 毎にノード(キュー)があり，送信側 AP が受信側 AP を指定してメッセージを送信する形態である．一方，図 4 (b)では複数の AP が同一のノード(キュー)から受信し，あるメッセージは 1 つの AP にのみ配信される．この図 4 (b)は，処理要求メッセージを複数のサーバに負荷分散する場合に利用できる．次に，図 4 (c)(d)は送信側 AP が送信したメッセージを，複数の受信側 AP が受信する形態(Pub/Sub 構成)である．図 4 (c)は送信側 AP が送信したメッセージを，全ての受信側 AP が受信する．AMQP ではリンクにメッセージの転送条件を設定することができ，図 4 (d)のようにメッセージの属性が条件を満たしたメッセージのみを配信することも可能である．このように，AMQP ではノードとリンクの概念により，柔軟なメッセージルーティングが実現できる．

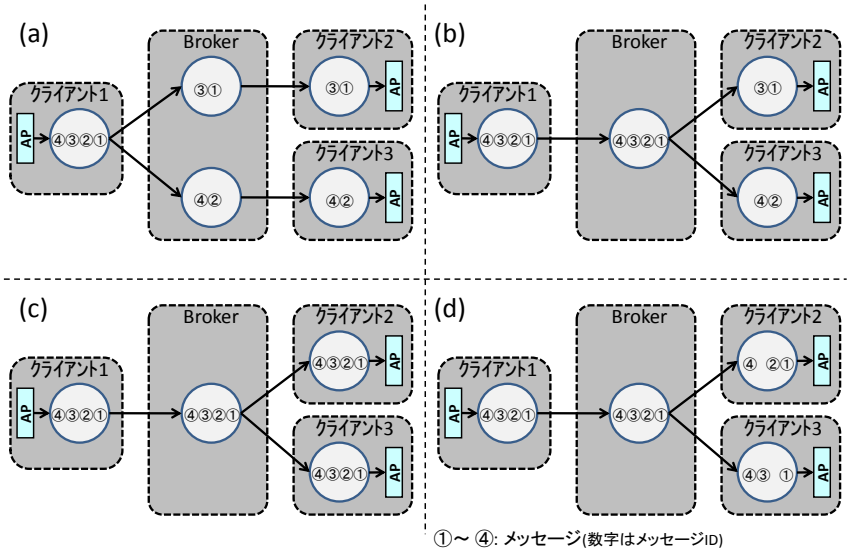


図 4　代表的なメッセージング形態

AMQP では管理層(図 1(a))で，上記のノードやリンクの生成，変更，削除の操作を提供する．また，セッション層(図 1(c))では，クライアント-Broker 間の接続手順やメッセージ送受信手順，認証や暗号化方式が定義されている．セッション層では流量制御についても仕様が定められている．さらに，メッセージ送信時にはメッセージに優先度を付与して配信することもできる．

トランスポート層((d))ではメッセージフォーマットが定義されている．メッセージフォーマットが仕様で定義されることにより，異なる実装間での相互接続が可能となる．AMQP のメッセージフォーマットはバイナリ形式であり，メッセージに含まれる(AP が格納した)データを意識せずに配信を行う．これにより，WS-*等のテキストベースプロトコルよりも可読性は劣るが，XML 解析などのオーバヘッドの大きい処理が不要となり，高速性が実現される．また，メッセージサイズも小さくすることが可能となり，ネットワーク帯域の使用量低減にもつながる．

以上のように AMQP は，相互接続性の実現，性能面への配慮，多様かつ柔軟なメッセージ送受信形態など，ミッションクリティカル分野における SOA 向けの MOM として必要な特徴を備えている．

Vol.2010-DPS-145 No.5
Vol.2010-GN-77 No.5
Vol.2010-EIP-50 No.5
2010/11/25

3.3 AMQP のオープンソース実装

AMQP はオープンな仕様であるため，複数の実装が存在する．実システムへの適用を考えると，性能に加えて，運用容易性や信頼性，および開発容易性が重要となる．そのため，本節では，AMQP の 3 つの実装 Apache Qpid**16)**，RabbitMQ**17)**，OpenAMQ**18)** を運用管理/信頼性/開発容易性の観点で比較した．さらに，AMQP 準拠ではないが実システムへの適用が報告されている OSS 実装の MOM として，ActiveMQ**10)**も比較対象に加えた．比較結果を表 4 に示す．

表 4　AMQP の OSS 実装

		AMQP			ActiveMQ
		Apache Qpid16)	RabbitMQ17)	OpenAMQ18)	
運用管理	管理 I/F	○ API,GUI,CLI	△ API,CLI	△ API,CLI	○ API,GUI,CLI
	セキュリティ	○ 暗号化,ｱｸｾｽ制御	○ 暗号化,ｱｸｾｽ制御	△ ｱｸｾｽ制御	○ 暗号化,ｱｸｾｽ制御
信頼性	永続化	○ (DBMS)	○ (file)	×	○ (DBMS, file)
	クラスタリング	○ 系切替,負荷分散	○ 系切替,負荷分散	○ 系切替,負荷分散	○ 系切替,負荷分散
開発環境	Broker 実装	C++/Java	Erlang	C/C++	Java
	Client 提供言語	C++,Java, Ruby,Python,C#	Java,Ruby,Python, .NET,PHP,Perl, Erlang,C,Lisp	C,Python, Java,Ruby	C,C++,.NET, Erlang,JavaScript, Perl, Ruby 等

管理インターフェースやセキュリティについては，Apache Qpid は ActiveMQ とほぼ同等の機能を備えていることが分かった．また，信頼性に関しても，Apache Qpid は ActiveMQ とほぼ同等の機能を備えていることが分かった．

4. 性能評価

2，3 章より，次世代 MOM として，相互接続性やコストの観点から AMQP が有望であること，AMQP の実装としては開発容易性などから Qpid が有望であることがわかった．本章では，自動取引など他分野に比べて特に高性能なメッセージ処理が要求されるミッションクリティカル分野への適用可能性を検証するために，Qpid を対象として実機を用いた性能評価を実施した．

4.1 評価環境および評価方法

評価環境と評価に用いたハードウェア/ソフトウェアの構成および仕様をそれぞれ図 5 および表 5 に示す．図 5 に示すように，評価は 2 台の PC サーバを 1GbE で接続した環境で実施した．すなわち，1 台の PC サーバ(Broker サーバ)で Broker を動作させ，もう 1 台の PC サーバ(測定サーバ)では，メッセージの送受信を行う測定プログラムを動作させた．

Broker としては，Qpid に加えて性能比較の対象として Apache ActiveMQ についても測定を行った．Qpid に関しては，C++/Java の両実装とも評価対象とした．また，測定には Apache Qpid 付属の測定プログラムである Perftests を利用した．Perftests は JMS API を利用しており，Qpid，ActiveMQ はいずれも JMS API をサポートしているため，測定プログラムを共通化できる．

評価は送信スループットを測定することで行った．測定における処理の流れは以下のとおりである．まず，測定プログラムから Broker へ連続的にメッセージを送信する(図 5①)．Broker はメッセージを受信すると，受信したメッセージを測定プログラムへそのまま配送する(図 5②)．測定プログラムは，5 分間連続してメッセージを送信する．本実験では，上記の処理の流れでメッセージサイズを変化させながら，スループット性能を測定した．

Vol.2010-DPS-145 No.5
Vol.2010-GN-77 No.5
Vol.2010-EIP-50 No.5
2010/11/25

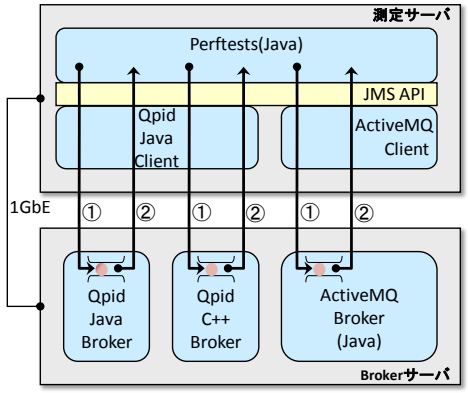


図 5　実験システムの構成

表 5　実験システムの仕様

#	項目	仕様
1	サーバ台数	2 台
2	CPU	Intel Core2 Q6700(L2: 4MB，2.66GHz, 4core)
3	メモリ	4GB
4	ネットワーク	1 Gigabit Ethernet
5	OS	Fedora 10 (Linux) カーネル: 2.6.27.5-117.fc10.x86_64
6	Java 実行環境	Sun Java Runtime Environment 1.6.0_13 Server VM
7	Qpid バージョン	0.5

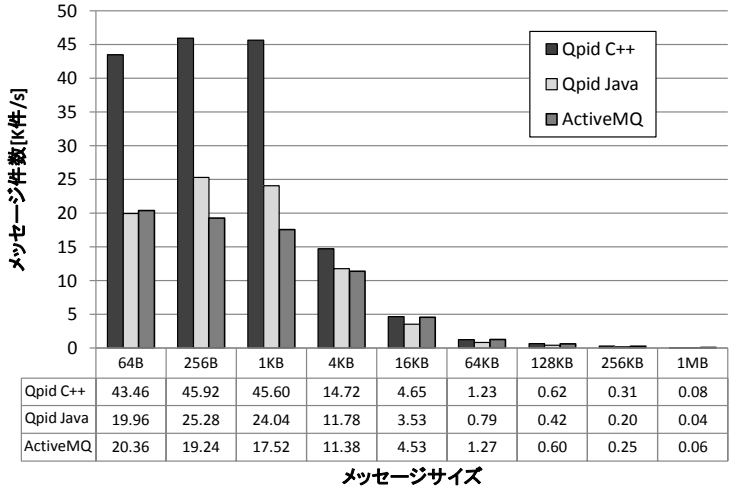
8	Qpid パラメタ	[C++版Broker]
		`--daemon --no-data-dir --tcp-nodelay --auth no --worker-threads 5 --mgmt-enable no --default-queue-limit 838860800`
		[Java版Broker]
		`-server -Xmx1024m -XX:+UseConcMarkSweepGC -Damqj.tcpNoDelay=true -Damqj.read_write_pool_size=5`
		[Perftests(測定プログラム)]
		`-server -Xmx1024m -Xms256m -XX:+UseConcMarkSweepGC`

4.2　測定結果および評価

図 6 および図 7 に測定結果を示す．図 6，図 7 とも横軸はメッセージサイズであり，図 6 はメッセージサイズに対する秒間のメッセージ送受信件数を，そして図 7 はメッセージサイズに対するメッセージ転送バイト数を示している．図 6 より，メッセージサイズが小さい場合は，Qpid C++版，Java 版とも，ActiveMQ を上回っていることがわかる．特に 1KB 以下のメッセージの送受信においては，Qpid C++版が他を大きく上回っており， 1KB の場合で 45,600 メッセージ/秒のスループットが得られており，ActiveMQ に対して 2.6 倍の性能向上となった．これは，Qpid C++では複数メッセージを一括化の上で送信を行っている事，及び，メッセージサイズが小さい場合，小さなデータのネットワーク I/O が多数発生するため，OS の I/O 関数を直接発行可能な C++での実装と，VM のオーバヘッドが発生する Java での実装の差が顕在化したためと推測される．

また図 7 より，メッセージサイズが大きい場合にでも，メッセージ転送バイト数(メッセージ転送量)において，Qpid C++版は ActiveMQ と同等もしくは上回っていることが分かる．一方 Qpid Java 版については，16KB より大きいメッセージでは ActiveMQ より性能が下回った．Qpid の C++版に対して Java 版の性能が低い理由は，AMQP 仕様そのものではなく Qpid Java 版の内部実装に起因すると考えられる．

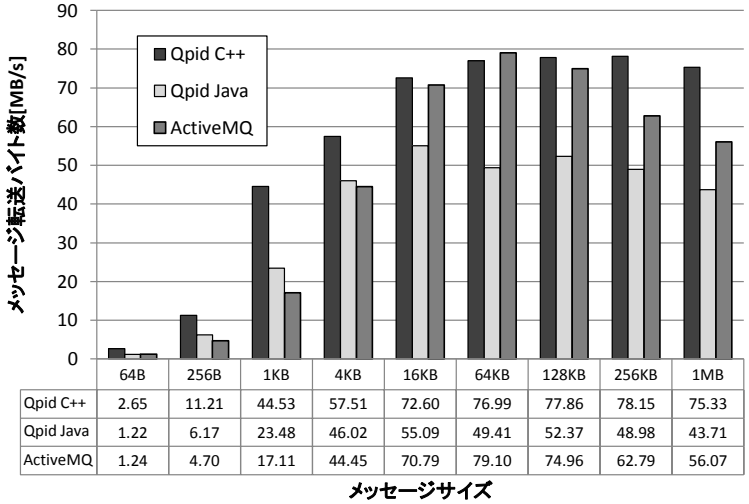
以上をまとめると，Qpid C++版についてはメッセージサイズにかかわらず，ActiveMQ と同等もしくはそれを上回る性能が得られた．



	64B	256B	1KB	4KB	16KB	64KB	128KB	256KB	1MB
Qpid C++	43.46	45.92	45.60	14.72	4.65	1.23	0.62	0.31	0.08
Qpid Java	19.96	25.28	24.04	11.78	3.53	0.79	0.42	0.20	0.04
ActiveMQ	20.36	19.24	17.52	11.38	4.53	1.27	0.60	0.25	0.06

図 6　秒あたりのメッセージ件数

Vol.2010-DPS-145 No.5
Vol.2010-GN-77 No.5
Vol.2010-EIP-50 No.5
2010/11/25



	64B	256B	1KB	4KB	16KB	64KB	128KB	256KB	1MB
Qpid C++	2.65	11.21	44.53	57.51	72.60	76.99	77.86	78.15	75.33
Qpid Java	1.22	6.17	23.48	46.02	55.09	49.41	52.37	48.98	43.71
ActiveMQ	1.24	4.70	17.11	44.45	70.79	79.10	74.96	62.79	56.07

図 7　秒あたりのメッセージ転送バイト数

5. 結言

5.1 本研究の結論

システムの SOA 化やクラウドコンピューティング環境の利用が進むにつれて,外部サービスとの相互接続性, システム構築・運用コストの削減がますます重要性となる．そのため，サービス連携の基盤となるメッセージ基盤(MOM)においても，相互接続性確保とコスト削減実現のために, 標準技術に基づいた OSS 実装の MOM が有力な選択肢となる．そこで本研究では，米国金融機関が中心に仕様策定を進めている AMQP に着目し，その OSS 実装を中心に MOM の調査および評価を行った．

その結果，AMQP は複数の実装間の相互接続性を保証していること，およびビジネス適用の重要な要素である改変可能なライセンス形態を持つ OSS 実装が存在することがわかった．また，標準的な構成の PC サーバを用いた性能評価の結果，AMQP の OSS 実装の一つである Qpid は秒間 45,000 件のスループットとなり，他の OSS 実装に対して 2.5 倍の性能が得られることがわかった．以上より，AMQP は相互接続性に加えて秒間数千件以上のスループットといった性能も重視する金融などミッションクリティカル分野にも適用可能なポテンシャルを有することを確認した．

5.2 今後の課題

今後の課題として，高信頼化や運用管理機能など，AMQP への拡張機能の検討が挙げられる．また，本報告では基本性能のみを検証したが，大規模構成における性能やスケーラビリティについても評価を進める必要がある．最後に，AMQP の仕様はまだ策定中の段階であり，今後も継続的に状況を調査していく必要がある．

参考文献

1) Dirk Krafzig, Karl Banke and Dirk Slama: Enterprise SOA, Prentice Hall (2004).
2) Massimo Pezzini: Best Practices in Advanced SOA, Gartner Application Architecture, Development and Integration Summit, pp. 12 (2009).
3) Information Week: Wachovia Puts Stock In A Service-Oriented Architecture. http://www.informationweek.com/news/software/soa/showArticle.jhtml?articleID=201806191 (2007).
4) Yefim Natis: Tutorial: Fundamental Models of SOA - Patterns of Design and Patterns of Use, Gartner Application Architecture, Development and Integration Summit, pp. 9 (2009).
5) Oracle: Java Message Service. http://www.oracle.com/technetwork/java/index-jsp-142945.html
6) AMQP Working Group: Advanced Message Queuing Protocol. http://www.amqp.org/
7) IBM: WebSphere MQ V7.0. http://www-06.ibm.com/software/jp/websphere/integration/wmq/
8) OASYS: Web Services Reliable Messaging TC WS-Reliability 1.1. http://docs.oasis-open.org/wsrm/ws-reliability/v1.1/wsrm-ws_reliability-1.1-spec-os.pdf
9) OASYS: Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.2. http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.2-spec-os.html
10) Apache Software Foundation: Apache ActiveMQ. http://activemq.apache.org/
11) IBM: MP07: WebSphere MQ - JMS V7 Performance Evaluations. http://www-01.ibm.com/support/docview.wss?rs=3163&context=SSWLGF&dc=D400&uid=swg24022778&loc=en_US&cs=UTF-8&lang=en
12) ORPA: “Updated Traffic Projections 2010 & 2011. http://www.opradata.com/specs/Updated_Traffic_Projections_2010_2011.pdf
13) John O'Hara: Toward a Commodity Enterprise Middleware, ACM Queue, Volume 5, Issue 4, pp. 48-55 (2007).
14) AMQP Working Group: AMQP 1.0 Specification DRAFT Revision: Recommendation, http://www.amqp.org/confluence/download/attachments/4489238/amqp-1-0-recommendation-draft.pdf (2010).
15) Roy Schulte: Debunking the SOA Fanatics and the Naysayers, Gartner Application Architecture, Development and Integration Summit, pp. 10 (2009).
16) Apache Software Foundation: Apache Qpid. http://qpid.apache.org/
17) Springsource: RabbitMQ. http://www.rabbitmq.com/
18) iMatix Corporation: OpenAMQ. http://www.openamq.org/