2Y-04

ネットワーク環境とメッセージブローカの実装の違いによる性能の比較

中川 雄介† 乃村 能成‡ 三宅 貴義†
†岡山大学 大学院自然科学研究科 ‡岡山大学 学術研究院 自然科学学域

1 はじめに

マイクロサービスといった分散システム内において，プロセス間の結合を疎にするためにメッセージブローカがしばしば用いられ，その性能はシステム全体の性能に大きく影響する．また，実装のしやすさと実行速度はしばしばトレードオフの関係にあるため，どちらを優先するべきかは条件によって異なる．そのため今回は C 言語と Ruby を用いてメッセージブローカを実装した．本稿ではスループットと使用するネットワーク帯域を測定し，各実装において必要となるネットワーク環境を評価する．

2 メッセージブローカの実装について

2.1 本システムの構成

本システムの構成とメッセージの流れを**図 1**，使用するメッセージを**表 1** に示す．本システムは 1 つのブローカと複数のクライアントから構成されている．ブローカはクライアントが接続すると，送信キューと送信済みキューを作成する．送信キューにはセンダから受信したメッセージを保存し，送信済みキューにはレシーバに送信したあと FREE_REQ を受信するまでメッセージを保存しておく．

Ruby を用いて実装したメッセージブローカの各スレッドの処理流れについて**図 2** に示し，以下で説明する．各スレッドは epoll_wait でクライアントからのメッセージを待ち受け，各メッセージに応じて以下の様に処理を行った後に ACK を返す．HELLO_REQ に対しキューが存在しなければキューを生成し，クライアントを登録する．SEND_REQ に対し宛先の送信キューにメッセージを追加する．FREE_REQ に対し送信済みキューからメッセージを削除する．また，epoll_wait が timeout した際にはブローカからレシーバへ PUSH_REQ を送信し，送信キューのメッセージを送信済みキューへ移動させる．

3 比較評価

3.1 評価の方針

ネットワーク環境を変更し，C 言語を用いて実装したメッセージブローカ [1] と Ruby を用いて実装したメッセージブローカの性能測定を行い，ネットワーク帯域の使用率を求める．測定するネットワーク環境は 100Mbps，1Gbps，10Gbps の 3 つである．センダはブローカに SEND_REQ を送信し，ACK の受信を待って次の SEND_REQ の送信を行う．これを指定された回数行って終了する．また，複数のレシーバに送信する場合は，それぞれのレシーバに対してラウンドロビンで送信する．レシーバはブローカから PUSH_REQ を受信し，ACK を送信しメッセージに応じた処理を行う．その後，FREE_REQ を送信する．これらを指定された回数行って終了する．

3.2 測定方法

計算機を 2 台用意し**図 3** のネットワーク構成で実験

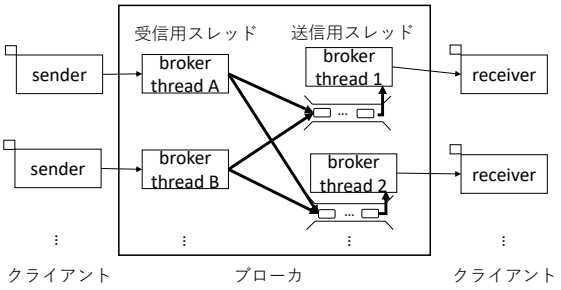

図 1 メッセージの流れ

表 1 メッセージの種類

通番	種類	説明
1	HELLO_REQ	クライアント登録
2	SEND_REQ	センダ→ブローカ送信
3	PUSH_REQ	ブローカ→レシーバ送信
4	FREE_REQ	レシーバの処理完了通知
5	ACK	メッセージに対する応答

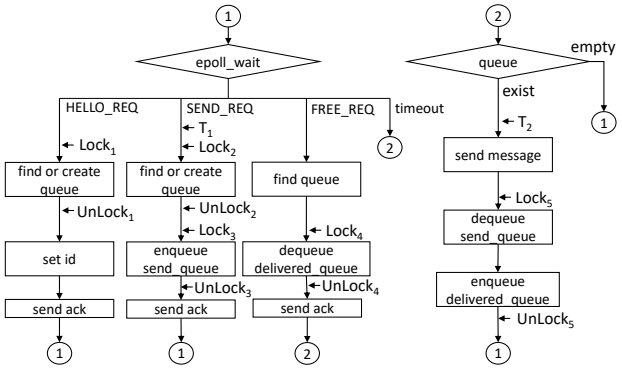

図 2 メッセージブローカの処理流れ

Performance Comparison under Different Network Environments and Message Broker Implementations
Yusuke Nakagawa†, Yoshinari Nomura‡, Takayoshi Miyake†
†Okayama University ‡Okayama University

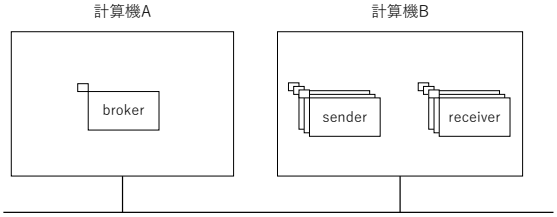

図 3　ネットワークの構成

表 2　実験環境

種類	計算機 A(ブローカ)	計算機 B(クライアント)
CPU	Ryzen 9 5950X (16C 32T@3.4GHz)	Core(TM) i5-9500 (6C 6T@3.00GHz)
メモリ	64GB	8GB
カーネル	Linux カーネル 5.15.0	Linux カーネル 5.15.0

を行う．Ruby 実装のメッセージブローカはブローカを計算機 A で起動し，計算機 B でセンダとレシーバをそれぞれ 10 プロセス起動する．ブローカを介して合計で 100,000 メッセージをセンダからレシーバへ送信し，スループットを測定する．各計算機を**表 2** に示す．ネットワーク帯域を変化させるために，計算機 A の NIC のリンク速度を 100Mbps，1Gbps，10Gbps と変えて，各ネットワーク環境においてスループットを測定した．測定区間は最初のメッセージの T_1 (図 2) から最後のメッセージの T_2 までとする．

3.3　評価結果

各ネットワーク帯域によるメッセージブローカの実装言語とスループットの関係を**図 4** に示し，以下で分かったことについて述べる．

(1) C 言語 の場合 1Gbps のネットワーク帯域はボトルネックとなる

C 言語で実装したメッセージブローカの場合，100Mbps と 1Gbps でネットワークの帯域使用率が 90%となり，スループットがネットワーク帯域の上限に達する．また図 4 にはないが 2.5Gbps のネットワーク帯域においては帯域使用率は 73%となり，ネットワーク帯域を広くした量に比例してスループットが増大した．しかし，10Gbps の帯域を用いた場合はネットワークの帯域使用率が 28%になるが，この際のスループットの増加量は比例しておらず，メッセージブローカの処理性能がボトルネックになったと考えられる．このため，C 言語を用いて実装したメッセージブローカで最大限の性能を必要とするような環境では，1Gbps のネットワーク帯域で通信を行う場合ネットワークがボトルネックとなってしまう可能性がある．

(2) Ruby の場合 1Gbps 以上のネットワーク帯域があれば十分となる

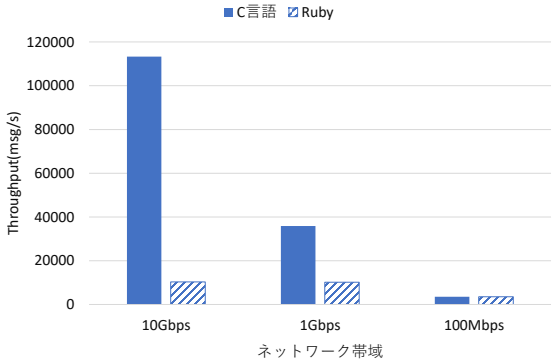

図 4　ネットワーク帯域による実装言語ごとのスループット

Ruby で実装したメッセージブローカの場合，100Mbps でネットワークの帯域使用率が 89%となり，スループットがネットワーク帯域の上限に達するため，ネットワークがボトルネックとなっている．また，1Gbps の帯域を用いた場合，ネットワークの帯域使用率は 25%と余裕があり，10Gbps にしてもスループットの増加が見られない．1Gbps と 10Gbps のネットワーク帯域においてスループットが増加しないのは Ruby で実装したメッセージブローカの性能によるスループットの限界に到達しており，メッセージブローカの処理性能がボトルネックとなっているためだと考えられる．このため，各クライアントとブローカが広域なネットワーク上に点在しているような環境においては Ruby を用いて実装したメッセージブローカで十分な性能を有する場合もあると考えられる．しかし，LAN で接続して通信を行うような環境はネットワーク帯域よりも Ruby で実装したメッセージブローカの処理性能がボトルネックになると考えられる．

4　おわりに

C 言語と Ruby を用いてスループットを測定し，使用するネットワーク帯域を求め，各実装において必要となるネットワーク環境を評価した．今回の実装と計算機の場合は，C 言語を用いたメッセージブローカの場合，1Gbps のネットワーク帯域で，ブローカが最大限の性能で動作した際にネットワーク帯域がボトルネックとなることが分かった．また，Ruby を用いたメッセージブローカの場合 1Gbps のネットワーク帯域があれば十分であり，各クライアントとブローカが広域なネットワーク上に点在しているような環境においては十分な性能を有すると考えられる．

参考文献

[1] 中川雄介，乃村能成：ネットワーク環境の違いによるメッセージブローカの性能比較，2022 年度 (第 73 回) 電気・情報関連学会中国支部連合大会 (2022).