M-023

通信プロトコルの違いによるメッセージブローカの処理時間の比較
Comparison of Message Brokers implemented in Different Communication Protocols

中川 雄介 †
Yusuke Nakagawa
乃村 能成 †
Yoshinari Nomura

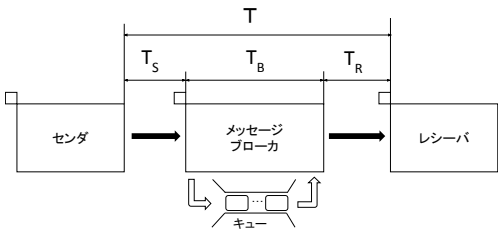

図 1: データの流れと測定のタイミング

1. はじめに

メッセージブローカとは，分散システムに用いられるメッセージキューサーバである．メッセージブローカを実装する際には，性能，信頼性，実装の容易さなどの観点から採用する実装言語や通信プロトコルにバリエーションが考えられる．

そこで本稿では，メッセージブローカの基本的な仕組みをいくつかの言語とプロトコルで実装し，それらの性能比較をした結果について報告する．具体的には，通信プロトコルとしてプレーンな TCP/IP と gRPC[1] の 2 種，言語として C 言語 (C++) と Ruby[2] の 2 種の違いで合計 4 つの実装を行い，メッセージ転送のスループットの違い，通信時間と CPU 処理時間の割合の違いを比較した．

これにより，比較した結果がメッセージブローカの実装方式を判断する材料の 1 つとなることを目的とする．

2. 評価システムの実装

通信プロトコルと実装言語による性能の差を評価するために，メッセージブローカの基本的な仕組みを実装したシステムを作成した．実装したシステム全体のデータの流れを**図 1** に示す．

センダ，ブローカ，およびレシーバを各 1 プロセスずつ用意し，センダは，メッセージパケットをブローカに送信する動作を繰り返す．ブローカは，受信したメッセージをキューに保存しつつ，レシーバに送信する．センダとレシーバは，1 スレッド，ブローカは 2 スレッドで，キューはロックフリーである．

それぞれの通信では，あらかじめ設定された数 (以降ウィンドウサイズ，WS) のメッセージを受信する毎に応答メッセージを返信することとする．

通信プロトコルとして TCP/IP と gRPC，実装言語として C 言語 (もしくは C++) と Ruby を用いて実装を行った．これら 4 実装を (C, TCP)，(C, gRPC)，(Ruby, TCP)，(Ruby, gRPC) と呼称する．

† 岡山大学大学院自然科学研究科，Graduate School of Natural Science and Technology, Okayama University

3. 比較評価

3.1 測定方法

センダ，メッセージブローカおよびレシーバを各 1 つずつ用意し，センダから 100,000 個のメッセージをブローカ経由でレシーバに送信して測定を行い，各処理にかかる時間とその割合およびメッセージブローカを介したセンダ，レシーバ間のスループットを求める．送信するメッセージのメッセージサイズが 1KB，2KB，4KB の場合に関してそれぞれ測定した．またウィンドウサイズ (以下 WS) を設定して，センダが 1 度に送信するメッセージの個数を決め，各場合について測定を行った．

測定したプログラムの組み合わせについて，以下では，実装言語が C 言語，通信プロトコルが TCP/IP，メッセージサイズが 1KB，WS が 1 の時 (C, TCP, 1KB, 1) と記述する．

測定区間については，図 1 の各区間: センダ→ブローカの通信時間 (T_S)，メッセージブローカ内のデータ保持時間 (T_B)，ブローカ→レシーバの通信時間 (T_R) とした．これらの合計 (T) をセンダの送信からレシーバの受信までにかかる時間とした．

3.2 評価結果

3.2.1 比較項目

各実装を比較する際の比較項目について以下に示す．

(1) WS がスループットに与える影響の差

(2) メッセージサイズがスループットに与える影響の差

(3) 各処理時間の全体に占める割合の差

3.2.2 WS がスループットに与える影響の差

メッセージサイズを 1KB に固定し，WS を 1 から 100,000 まで変化させて，100,000 回メッセージを送信するのにかかる時間を測定した．実装言語と通信プロトコルによるスループットの比較を**図 2** に示し，以下に説明する．

(1) WS が十分に大きいとき，(C, TCP) のスループットは，(Ruby, TCP) のそれに対し約 2 倍である．WS 100,000 でのスループットを比較すると，(C, TCP) では約 131,773 msgs/s，(Ruby, TCP) では約 68,038 msgs/s である．WS を減少させると，実装言語によるスループットの差は減少する．これはブローカ内の処理速度による影響が小さくなったためであると考えられる．

(2) (Ruby, gRPC) のスループットは (Ruby, TCP) の約 1/5 である一方で，(C, gRPC) は (C, TCP) の約 1/2 であり，(Ruby, TCP) と同等である．

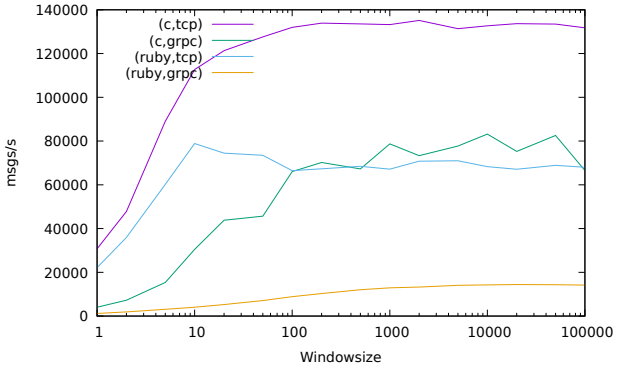

図 2: 同じメッセージサイズ (1KB) のときの
各実装の WS によるスループット

gRPC の場合，メッセージサイズを 1KB から増やすことでスループットが改善される可能性がある．

(3) gRPC は，TCP/IP に比べ，WS 増大の効果が緩やかである．

通信プロトコルに TCP/IP を用いて実装したメッセージブローカは WS が 10 から 50 の間で既にほぼ最大のスループットに到達しているが gRPC が最大のスループットに到達するのは WS が 100 を超えてからになっている．このことから gRPC は TCP/IP に比べて 1 回の通信にかかるオーバーヘッドが大きく，性能を引き出すためにはある程度 WS を大きくして通信回数を減らす必要があることが分かる．

3.2.3 メッセージサイズがスループットに与える影響の差

WS 100,000 として，メッセージサイズを 1KB，2KB，4KB と変えながら，100,000 回メッセージを送信するのにかかる時間を測定した．実装言語と通信プロトコルによるスループットの測定結果を**表 1** に示し，以下に説明する．

(1) 実装言語に C 言語，C++を用いた実装は Ruby を用いた実装に比べてメッセージサイズを増大させてもスループットが大きく減少しない．

(C, TCP) はメッセージサイズが 1KB から 4KB と増えたときスループットは約 7.2%減少しており，(C, gRPC) は約 4%減少している．また (Ruby, TCP) はスループットが約 26.2%減少しており，(Ruby, gRPC) はスループットが約 39.2%減少している．このことから C 言語，C++は Ruby に比べ，メッセージサイズを増加させてもスループットの減少幅が小さいため，メッセージサイズ増加による性能への影響がより大きいと考えられる．

3.2.4 各処理時間の全体に占める割合の差

メッセージサイズを 1KB に固定し，WS を 1，10，100 と変更して 100,000 回メッセージを送信した．(Ruby, gRPC) と (Ruby, TCP) の各処理時間を測定し，各処理時間の全体に占める割合を**図 3** に示し，以下で説明する．

表 1: 同じ WS(100,000) のときの各実装の
メッセージサイズによるスループット

	スループット (msgs/s)		
メッセージサイズ	1KB	2KB	4KB
(C, TCP)	131,773	129,008	122,312
(C, gRPC)	75,476	74,478	72,446
(Ruby, TCP)	68,038	69,582	50,226
(Ruby, gRPC)	14,186	11,510	8,618

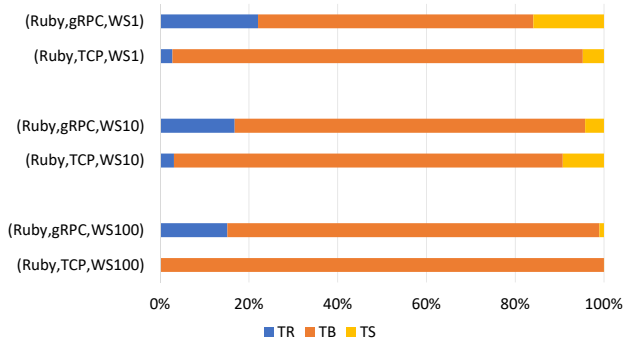

図 3: 各 WS における Ruby を用いた実装の
通信プロトコルの違いによる処理時間の割合

(1) (Ruby, gRPC) は (Ruby, TCP) に比べ WS 1，10，100 においてセンダとレシーバの通信時間の全体に占める割合が大きくなっている．

WS 1 のとき (Ruby, gRPC) はセンダとレシーバの通信時間の割合の合計が約 37%となっているが (Ruby, TCP) は約 7.5%となっており (Ruby, gRPC) の方が約 30 ポイント高くなっている．このことから実装言語が同じであるためキュー操作による内部処理の速度差はほとんどなく，通信プロトコルによる通信時間の差が全体に占める割合の差として表れていると考えられる．

4. おわりに

実装言語と通信プロトコルの視点から，メッセージブローカの基本的な仕組みを実装したプログラムを実装し，プログラムの性能について比較評価をした．今後は，各処理をさらに細かく分割して各処理の時間割合から，OS 処理の影響を含めた詳細な分析を検討する．

参考文献

[1] Google: gRPC, Google (online), available from ⟨https://grpc.io/⟩ (accessed 2021-5-25).

[2] Matsumoto, Y.: Ruby Programming Language, Ruby community (online), available from ⟨https://www.ruby-lang.org/ja/⟩ (accessed 2021-5-25).