社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2009－DPS－138 (35)
2009－CSEC－ 44 (35)
2009／3／6

追記・参照型データ管理システムにおけるPush／Pull混在方式の特性評価

長谷川 知洋 赤間 浩樹 山室 雅司
日本電信電話株式会社 NTTサイバースペース研究所

大量のセンサーデータや各種情報システムのログ情報，ブログなどユーザによって生成される情報，映像や音声などの各種ストリームデータを収集・蓄積・管理し，利用者の目的に応じた処理を行う情報統合管理サービスを目指して追記・参照型データ管理システムの研究開発を進めている．追記・参照型データ管理システムではデータの受付を行う追記部とデータを処理するフィルタ部が存在し，追記部とフィルタ部の間のデータ受け渡しを，(a)追記部からフィルタ部へのPushのみ，(b)フィルタ部による追記部からのPullのみ，(c)前記Push／Pullの混在の3方式について，各種パラメータを変化させて評価実験を行った．本論文では評価結果に基づいて各方式の特性を考察する．

An Evaluation of Push, Pull and Hybrid Data Transfer Methods for Stream Data Processing

Tomohiro Hasegawa, Hiroki Akama and Masashi Yamamuro
NTT Cyber Space Laboratories, NTT Corporation

Various stream data are generated, like sensor data, logs on various information systems, blog, video and voice data, etc. To deal with these data, we are developing stream data management systems in WRITE ONCE and READ MANY manner. This system has three function units: data receiving unit, data processing unit and data providing unit. In this paper, we describe three options of transferring data from data receiving unit to data processing unit: (a) data push, (b) pull and (c) hybrid of push and pull. We evaluate the overall processing time between these two units with the three methods.

1. はじめに

近年，気象・防災情報[1]，道路交通情報やヘルスケア情報などのセンサーデータや各種情報システムのログ情報，ブログなどユーザによって生成される情報，監視カメラ映像や音声などのストリームデータが広域分散ネットワーク上に大量に生成されてきており，これらのストリームデータを統合して利用したいという要求が高まってきている[2,3]．これらの時系列情報を追記型で収集・蓄積・管理し，各種ストリームデータを組み合わせて新たな価値を創造したり，利用者の目的に応じた処理を行う情報統合管理サービスを目指して追記・参照型データ管理システムの研究開発を進めている[4,5,6]．

我々が描く情報統合管理サービスのイメージを図 1に示す．映像やシステムログなど各種情報源から生成された時系列データが生データとして受付・蓄積されるのと同時に，ユーザの目的に応じて情報源ごとにあらかじめ登録されている様々な演算処理が呼び出され，最終的には加工データとなって各種応用サービスに出力・提供される．情報統合管理サービスの実現に向けた様々な課題のうち，我々は特に情報源の追加や処理量の増加に対する演算処理の動的スケールアウト（規模成長）や，連続して到着するストリームデータに対する処理内容の動的変更（機能成長）の実現を目指している．

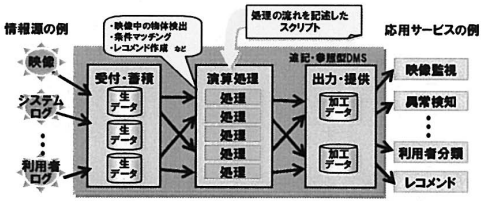

図 1 情報統合管理サービスイメージ

本論文では，2章で情報統合管理サービスを実現するための基盤システムとして，現在開発中の追記・参照型データ管理システムについて文献[6]に沿って述べる．3章では追記・参照型データ管理システムにおける 3 種類（Push／Pull／Hybrid）のデータ転送方式を実装した実験システムについて述べ，4章では本実験システムを用

いて行った各データ転送方式の評価と考察を行い，5章でまとめと今後の課題を示す．

2. 追記・参照型データ管理システム

追記・参照型データ管理システムでは，情報源の追加や処理量の増加に対する演算処理の動的スケールアウト（規模成長）の実現や，連続して到着するストリームデータに対する処理内容の動的変更（機能成長）の実現のために図 2に示すようなアーキテクチャを採用している．

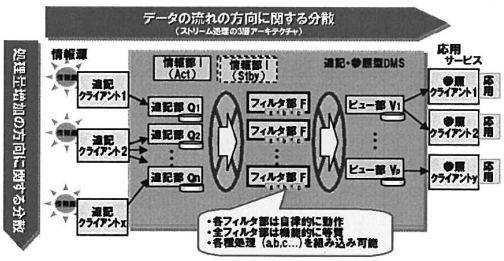

図 2 システムアーキテクチャ

2.1. アーキテクチャ概要

本システムは，各種情報源から追記型でデータの受付および蓄積を行う**追記部**，受け付けたデータに対して多様な演算処理を行う**フィルタ部**，処理結果のデータを各種応用サービスに提供する**ビュー部**の 3 つの機能から構成されており，分割された各機能は数百台規模の PC クラスタ上で動作する．情報源の追加や処理量の増加に対して追記部およびフィルタ部の台数を増加させることで対応し，マルチキャストによる緩やかな情報同期機能を活用して追記部およびフィルタ部が系に柔軟に参加・脱退できる仕組みになっている．系内に複数存在するフィルタ部は各々が等しい処理能力を持ち（機能的に等質），それぞれが独立に動作する．各フィルタ部は追記部からデータを取得すると，情報源ごとにあらかじめ登録されている演算処理を行い，処理結果をビュー部に提供する．

2.2. Push／Pull／Hybrid データ転送方式

追記・参照型データ管理システムにおける追記部とフィルタ部間のデータ転送方式として，以下の 3 方式が考えられる．

(a) Push 方式

フィルタ部にキューを持たせ，キューに空きがあるフィルタ部に対して追記部からデータを配信（Push）する方式．

(b) Pull 方式

キューを持たないフィルタ部が，自ら都合の良いタイミングで追記部に接続してデータを取得（Pull）する方式．

(c) Hybrid 方式

キューを持つフィルタ部とキューを持たないフィルタ部を混在させ，追記部からフィルタ部へのデータ配信（Push）とフィルタ部による追記部からのデータ取得（Pull）を並行に動作させる方式．

文献[6]で述べたシステムでは，(1)フィルタ部が過負荷になることを防ぎ，系としての安定性を向上させる目的と(2)フィルタ部が自らのタイミングで処理の制御（処理の実行，停止，再起動など）を可能にする目的から，方式(b)のPull方式を採用しているが，クライアントからのデータ入力頻度やフィルタ部におけるデータ 1 件あたりの処理量などの違いによっては，他の方式が有利なケースも考えられる．

3. 実験システム

3.1. 実験システム概要

追記・参照型データ管理システムにおける追記部とフィルタ部間の 3 つのデータ転送方式の特性を評価するために，図 3に示すような実験システムを作成した．本実験システムは，クライアントと追記部およびフィルタ部から構成されている．今回の実験では処理結果を標準出力に出力するだけの簡単なモデルであるため，ビュー部の実装は省略した．

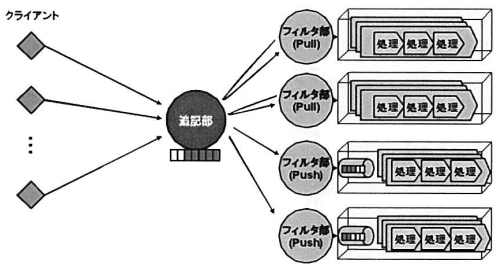

図 3 実験システム概要

3.2. クライアント

本実験システムでは，クライアント数を自由に定義することができるが，今回の実験ではすべての評価パターンでクライアント数を 5 に固定した．各クライアントは 100ms 毎に指定した件数

のデータを追記部に入力することができ，今回の実験では100ms毎に20，50，100，200件のデータを入力させた．また，5つのクライアントからのデータ入力頻度を均一にしたデータ入力パターンと各クライアントのデータ入力頻度にばらつきがある不均一なデータ入力パターンについての比較も行った．

3.3. 追記部

追記部はキューを持ち，クライアントから受け付けたデータがフィルタ部によって処理されるまでの間，一時的にデータをキューに蓄積する．本実験システムではクライアントから受け付けたデータが溢れることがないように追記部のキューを可変長としている．

追記部では個々のフィルタ部を一意に識別しており，各フィルタ部がPushでデータを取得するか，Pullでデータを取得するかを管理している．Pushでデータを取得するフィルタ部に対しては，ラウンドロビン順にデータを転送する．データ転送先のフィルタ部のキューに空きがなければ，転送順序をスキップして次のフィルタ部にデータを転送するが，すべてのフィルタ部のキューに空きがなくなるとデータは追記部のキューに滞留する．一方，Pullでデータを取得するフィルタ部に対しては，フィルタ部からのデータ取得要求に応じて，都度データを転送する．

3.4. フィルタ部

フィルタ部はPushまたはPullのいずれか一方の方式により，追記部のキューに蓄積されたデータを取得し，各データに対して，情報源ごとにあらかじめ登録されている処理を行う．Push方式に対応するために各フィルタ部が固定長のキューを持っており，評価パターンに応じてキューのサイズを変更することが可能である．今回の実験では，Push方式におけるキューのサイズを1，10，100，1000と変化させて評価を行った．Pull方式の場合は，フィルタ部がデータのキューイング機能を持たない状態にして評価を行った．

フィルタ部で実行される処理は，小さな機能部品をワークフローとして組み合わせることで，1つの意味のある処理を実現している．従来技術として，処理ノード（フィルタ部に相当）ごとに決まった役割をもたせた様々な機能部品を動作させておき，データに機能部品ノード間を往き来させることでワークフロー処理を実現するアプローチ[7,8]もあるが，本実験システムでは，分散環境における通信オーバーヘッドや信頼性および可用性を考慮し，フィルタ部にすべての機能部品をもたせ，データの情報源ごとに決められた処理を実行するのに必要な機能部品を繰り返し呼び出すことでワークフロー処理を実現するアプローチを採用した（図4）．

本実験システムにおける機能部品は**Pluglet**と呼ばれ，ワークフロー定義の中で様々な機能を持ったPlugletの処理順序を自由に設定することができる．各Plugletには処理対象のデータに加え，ワークフローの経路情報（次に実行されるべきPluglet の情報）も渡すことができるため，現在処理中のPlugletの処理結果に応じてワークフローの経路情報を書き換えて，次に実行されるPlugletを動的に変更することも可能である．ワークフローとして定義された一連のPluglet処理はフィルタ部にある**PlugletManager**によって管理される．本実験システムではフィルタ部上に任意個の PlugletManager を並行に動作させることができるため，Pluglet によるワークフロー処理を複数並行に処理させることができる．今回の実験では，1つのフィルタ部で動作させるPlugletManager数を2，5，10，20，50，100と変化させて実験を行った．

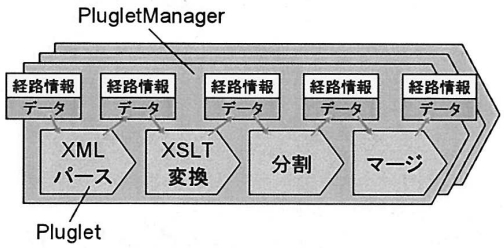

図 4 Pluglet と PlugletManager

4. 評価と考察

4.1. 実験環境

追記・参照型データ管理システムにおける追記部とフィルタ部間の3つのデータ転送方式（Push／Pull／Hybrid）の特性を評価するために，前記実験システムの各種パラメータを変更しながら，指定した入力頻度でクライアントが入力した最初のデータが追記部に到着してから最後のデータがフィルタ部で処理完了するまでの時間（処理時間）を測定した。実験環境として，DELL PowerEdge 1750（CPU:Xeon 3.06GHz,

Memory : 2GB, OS : Fedora Core）を 5 台用意し，追記部サーバ 1 台，フィルタ部サーバ 4 台の構成で実験を行った．4 台のフィルタ部サーバのうち，4 台すべてを Push 方式のフィルタ部サーバ（Push サーバ）として動作させた場合と，4 台すべてを Pull 方式のフィルタ部サーバ(Pull サーバ）として動作させた場合と，Push サーバと Pull サーバを混在させた構成で動作させた場合（Hybrid）のそれぞれについて，以下の評価パターンにおけるデータの処理時間を測定した．

4.2. 評価パターン

(A) データ入力頻度とキューサイズの関係

Push サーバが持つ固定長のキューのサイズを 1，10，100，1000 と変化させた場合のそれぞれの処理時間[ms]を，5 つのクライアントからのデータ入力頻度を 20，50，100，200[件/100ms]と変化させながら測定した．入力頻度が 20[件/100ms]の時に系の入出力が均衡するモデルとし，50，100，200[件/100ms]と増加させることでバースト的な入力を模擬した．各クライアントからはそれぞれの入力頻度において，100[ms]×60 回のデータ入力を行い，1 つのフィルタ部における PlugletManager 数は 2 に固定した．

Pushサーバのキューサイズを 1 にした場合（図 5），Push方式（4Push）は他の方式に比べて処理時間が長くなっている．これは追記部がフィルタ部にデータをPushしたくてもフィルタ部のキューに空きがない状態が続き，データが追記部のキューに滞留したことが原因と考えられる．

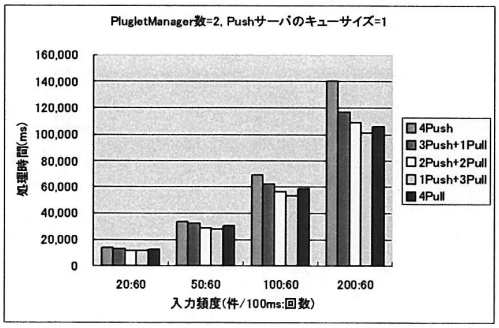

図 5 キューサイズ=1 の処理時間

一方，Pushサーバのキューサイズを 10，100，1000 にした場合（それぞれ図 6，図 7，図 8）の結果は一変して，Push 方式やHybrid 方式（3Push+1Pull，2Push+2Pull，1Push+3Pull）の処理時間が大幅に短縮されている．Push方式の大幅な処理時間短縮は追記部のキューでのデータ滞留が解消されたことと，フィルタ部のキューによるキューイング効果により，追記部とフィルタ部間の通信オーバーヘッドが隠蔽されたことによる効果であると考えられる．また，Hybrid 方式でもPushサーバが処理時間短縮の効果を受けたことと，Pushによるデータ転送の裏側でPullによるデータ転送も行っていることにより，Push方式よりもデータ転送能力が向上し，全体の処理時間が更に短縮されたと考えられる．

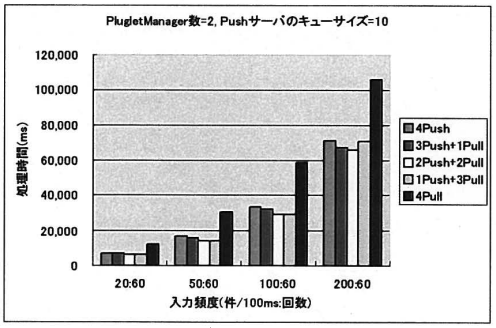

図 6 キューサイズ=10 の処理時間

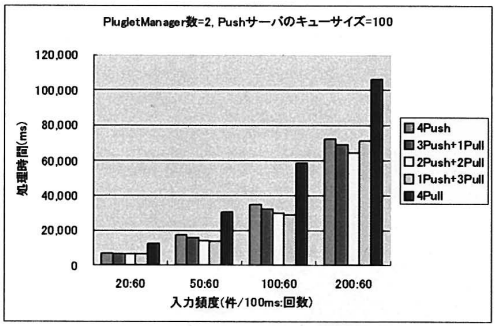

図 7 キューサイズ=100 の処理時間

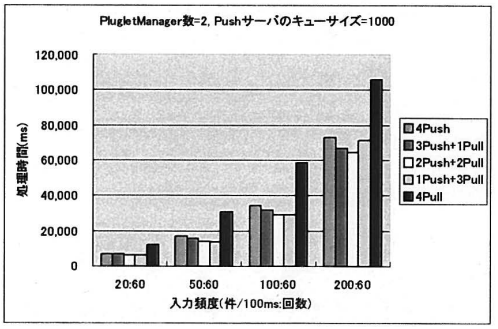

図 8 キューサイズ=1000 の処理時間

(B) 均一なデータ入力と不均一なデータ入力

フィルタ部のPlugletManager数を 2，Pushサーバのキューサイズを 100 に固定し，クライアントから均一にデータを入力した場合と，不均一にデータを入力した場合を比較した．均一なデータ入力では 5 つのクライアントが等しく 200[件/100ms]のデータを 60 回入力し，不均一なデータ入力では 5 つのクライアントがそれぞれ 500, 200, 100, 100, 100 [件/100ms]のデータを 60 回入力した（総データ件数は同じ）．測定した処理時間の比較結果を図 9に示す．

Push／Pull／Hybrid のいずれの方式もほぼ同様の傾向を示しており，均一なデータ入力と不均一なデータ入力を比較した結果，顕著な差異は見られなかった．これは追記部のキューがクライアントからの不均一な入力を吸収して，フィルタ部に対して不均一な入力を隠蔽したことが原因と考えられる．

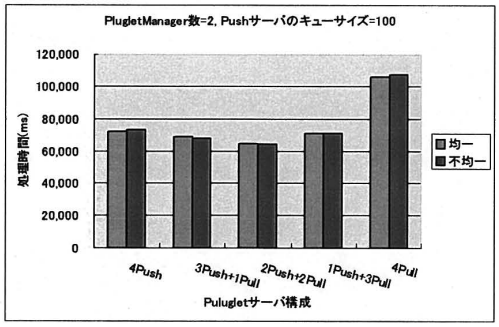

図 9 均一な入力と不均一な入力の比較

(C) 入力頻度と PlugletManager 数の関係

Pushサーバのキューサイズを 100 に固定し，フィルタ部のPlugletManager数を 2, 5, 10, 20, 50, 100 と変化させた場合の処理時間を比較した．クライアントからのデータ入力頻度を 20[件/100ms]とした場合の結果を図 10に示し，200[件/100ms]とした場合の結果を図 11に示す．

フィルタ部サーバ上の PlugletManager 数が少ない場合はフィルタ部の処理能力が不足し，特に Pull 方式（4Pull）の処理時間が長くなっている．この傾向は高頻度（200[件/100ms]）でデータを入力した場合も同様である．一方，PlugletManager 数が増加するにつれて，Push 方式の処理時間が増加する傾向が見られる．これは PlugletManager 数の増加に伴い，追記部から Push でデータを転送すべき Push サーバ数が増加し，ラウンドロビンでデータを転送する周期が長くなってしまい，データ待ち状態の Push サーバが多く発生したことが原因であると考えられる．いずれの場合も Hybrid 方式(2Push+2Pull, 1Push+3Pull)が良好な結果を示しており，この傾向は高頻度（200[件/100ms]）なデータ入力時に顕著である．

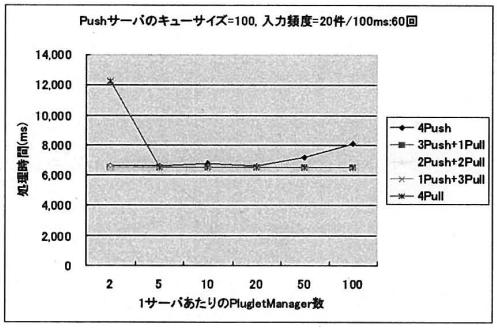

図 10 入力頻度（低）と PlugletManager 数

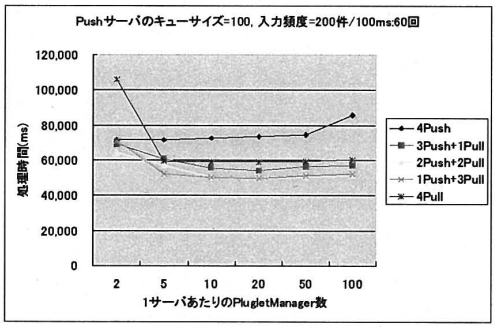

図 11 入力頻度（高）と PlugletManager 数

(D) 重い Pluglet 処理とキューサイズの関係

フィルタ部のPlugletManager数を 2，クライアントからのデータ入力頻度を 20[件/100ms]に固定し，フィルタ部のPluglet処理でデータ 1 件あたり 100～200[ms]かかるビジーループを埋め込んで擬似的に重い処理を作り出し，Pushサーバのキューサイズを変化させて処理時間の測定を行った．今回の実験では，Pushサーバのキューサイズを 100 にした場合と 1000 にした場合について，それぞれ測定した．処理時間を比較した結果を図 12に示す．

Push サーバを構成に含んだ Push 方式および Hybrid 方式は，Pull 方式に比べて処理時間が長くなっている．特に Push サーバのキューサイズが 1000 の場合は，Push 方式(4Push)，Hybrid

方式(3Push+1Pull), Hybrid 方式(2Push+2Pull), Hybrid 方式(1Push+3Pull) の順, つまり Push サーバの台数が少なくなるにつれて処理時間が増加している. これは Push サーバのキューサイズが大きいと, 追記部のキューに蓄積されているデータが早い段階で Push サーバのキューに転送され, Pull サーバが追記部のキューからデータを取得しようとしてもデータが存在しない状況が発生していると考えられる. その結果, Push サーバが自身のキューに蓄積されたデータを処理してビジー状態になっているにもかかわらず, Pull サーバは処理すべきデータが取得できずにアイドル状態となり, 全体として, Push サーバの台数が少ない構成ほど十分な処理能力が得られずに処理時間が長くなったと考えられる.

一方, Pull 方式では各 Pull サーバがアイドル状態になったタイミングで自ら追記部からデータを取得してデータ処理を行うため, 結果的に 4 台のサーバの処理能力を効果的に使うことができ, 処理時間を短くすることができたと考えられる. Push 方式では Push サーバのキューが固定長のため, すべての Push サーバのキューに空きがなくなる状況が発生し, 一時的に追記部からフィルタ部にデータを Push できなくなったことが原因で Pull 方式に及ばなかったと考えられる.

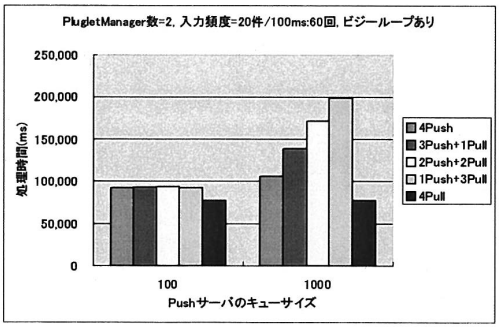

図 12 重い Pluglet 処理とキューサイズ

5. おわりに

今回の実験を通して, 追記・参照型データ管理システムにおける追記部とフィルタ部間のデータ転送方式として, フィルタ部の処理が重い場合に Pull 方式が有効であることが確認できた. また, 条件によっては Hybrid 方式が十分有効であることが確認できた. 今回の実験結果をまとめると以下のようになる.

- フィルタ部の処理が比較的重い場合は Push 方式, Hybrid 方式のいずれの場合も Push サーバのキューサイズを小さくする／もしくは Pull 方式で対処し, それ以外のケースでは Hybrid 方式を用いる
- Hybrid 方式を採用する場合は, Push サーバのキューサイズを適度に小さくし, 系内の Push サーバと Pull サーバの割合を同程度にする
- Push 方式を採用する場合は、1 フィルタ部あたりの PlugletManager 数を大きくし過ぎないようにし, Pull 方式を採用する場合は、1 フィルタ部あたりの PlugletManager 数を小さくし過ぎないようにする

今回の実験では, 4 台のフィルタ部サーバを Push サーバや Pull サーバにして, 合計 5 パターンの構成について評価を行ったが, 今後は更にサーバ台数を増やした評価や, 追記部のキューを可変長から固定長に変更した際の特性評価を行うことなどが考えられる. また, クライアントからの入力データのデータサイズと入力頻度が処理時間に及ぼす影響をより詳細に調査し, データサイズや入力頻度に応じて, Push／Pull／Hybrid 方式を自律的に変化させながら効率的な処理を行うことができる追記・参照型データ管理システムの開発を行っていきたい.

参考文献

[1] 防災情報提供センタ. http://www.bosaijoho.go.jp/.

[2] 渡辺, 北川, 内山 : 問合せ最適化機構を備えたデータストリーム統合システムの開発, 電子情報通信学会 DEWS2004 3-C-04 (2004).

[3] Coral8 Technology Overview. http://www.coral8.com/system/files/assets/pdf/5.2.0/Coral8TechnologyOverview.pdf.

[4] 赤間, 内山, 三浦, 西岡, 内藤, 谷口, 山室, 櫻井 : 追記・参照型データ管理プラットフォームアーキテクチャの提案, 情報処理学会 DPSWS 2006, pp199-204 (2006).

[5] 内山, 赤間, 西岡, 内藤, 谷口, 長谷川, 三浦, 山室, 櫻井 : 分散データストリーム処理アーキテクチャの提案, 情報処理学会研究報告 2007-DBS-143, DBWS 2007, pp327-332 (2007).

[6] 赤間, 内山, 西岡, 内藤, 谷口, 長谷川, 兵藤, 三浦, 山室, 櫻井 : 追記・参照型データ管理システムの設計と評価, 情報処理学会論文誌, Vol.49, No.2, pp749-764 (2008).

[7] 高橋, 昆, 秋山, 神宮司 : 分散データ駆動型アーキテクチャの性能評価とアプリケーション設計, 電子情報通信学会論文誌 B Vol. J88-B, No.7, pp1202-1212 (2005).

[8] Yahoo Pipes. http://pipes.yahoo.com/pipes/.