7M-01

メニーコアシステムにおける分散ストリーム処理システムの性能評価 – スループットに関する評価 –

徳増　直紀 † 石川　佳治 †† 杉浦　健人 ††

† 名古屋大学工学部電気電子・情報工学科 †† 名古屋大学大学院情報学研究科

1 はじめに

現在，ビッグデータに対して，よりタイムリーな解析の要求が増加している．しかし，従来のバッチ処理システムではデータの取得から処理までにラグがあり，そうした要求に答えるための低遅延な処理が難しい．それにより，分散並列ストリーム処理システムへの関心が高まり，学術・産業両面において研究開発が進められている．ストリーム処理システムは，バッチ処理システムのように有限のデータセットを想定するのではなく，無限のタプルを持つデータセットを想定し入力されるタプルを順次処理することでリアルタイムな解析を可能にするシステムである．

分散並列ストリーム処理の OSS の主なものはシェアードナッシングアーキテクチャを採用し，複数マシンを並列に動作させることで性能をスケールアウトしている．シェアードナッシングアーキテクチャとは，分散並列システムの構成法の一つで，システムの各ノードがリソースを共有しておらず，それぞれが独立するように設計されたアーキテクチャである．つまり，この方式ではノード間の共有リソースがボトルネックとならず，ノードの増加による性能のスケールアウトが円滑に行える．

一方で，近年ではメニーコア CPU を用いたマシン性能のスケールアップが見られ始めている．これは，プロセッサ単体の性能向上が限界に近づき，ムーアの法則が成り立たなくなってきているため，それを代替する方法として CPU コアのコア数を増やすというアプローチが近年活発に行われているからである．ここで，ムーアの法則とは，集積回路の進化の指標で、インテル社の創業者のひとりであるゴードン・ムーアが 1965 年に提唱した「半導体のトランジスタ集積率は 18 ケ月で 2 倍になる」というものである．

しかし，メニーコアシステムにおける既存の OSS の並列処理の性能はまだ未知数な部分が多い．そこで，本研究では既存の分散並列ストリーム処理 OSS のメニーコア CPU における処理性能を調査し，特にスループットの面からその評価を行う．

Performance Evaluation of Distributed Stream Processing Systems in a Many-Core System: Evaluation of Throughputs

Naoki Tokumasu†, Yoshiharu ishikawa††, and Kento Sugiura††

†Department of Information Engineering, School of Engineering, Nagoya University

††Graduate School of Informatics, Nagoya University

2 調査対象 OSS

本章では，本研究で調査対象とした Flink [1]，Samza [2]，Storm [3]，Spark Streaming [4] についてそれぞれの特徴を述べる．

2.1 Flink

Flink の処理は，on-at-a-time で行われる．Flink では，入力のストリームは事前に 1 つ以上のストリームパーティションに分割される．また，各オペレーターにも 1 つ以上のオペレーターサブタスクがあり、それらのオペレーターサブタスクは互いに独立していて異なるスレッドや異なるマシンで並列に実行される．オペレーターサブタスクの数は，その特定のオペレーターの並列処理数である．ストリームは 2 つのオペレーター間でデータを転送するが，それらの対応が 1 対 1 の時は要素の分割と順序を保持し，ストリームが再配布される場合はストリームの分割を変更する．

2.2 Samza

Samza の処理も Flink と同様に one-at-a-time で行われる．Samza でも入力ストリームは 1 つ以上のパーティションに分割されるが，各パーティションにはメッセージが割り当てられる．この各メッセージには，パーティションごとに一意のオフセットという識別子がある．ジョブは，複数のタスクに分割することでスケーリングされ，タスクは各入力パーティションからデータを消費する．タスクは，メッセージオフセットの順序で各入力パーティションからのメッセージを順番に処理する．タスクへのパーティションの割り当ては変更できず，タスクが失敗したマシン上にある場合，タスクは他の場所で再起動され，同じストリームパーティションを消費する．

2.3 Storm

Storm の処理は，Storm Core であるか Storm Trident であるかによって異なってくる．Storm Core では one-at-a-time で処理されるのに対し，Storm Trident では micro-batch 形式が採用されている．そして，Storm では spout（ストリームのソース）からストリームを入力として受け取り，そのストリームをグループ化によって各ボルトに割り振り，各ボルトがそれらを処理して出力ストリームを生成する．この時，ストリームは事前に分割される．

2.4 Spark Streaming

Spark Streaming の処理は，microbatch で行われる．Spark Streaming では，1 つのデータソースから読み込んだデータを 1 つの RDD（Resilient Distributed Dataset）として扱い，1 つの RDD は複数のパーティションに分割され，1 パーティショ

表 1　マシン性能

機器名	Dell PowerEdge R640
OS	Ubuntu 18.04.3 LTS
CPU	Intel(R) Xeon(R) Gold 6262V CPU @ 1.90GHz
メモリ	256GB
ストレージ	480GB

ンを 1 タスクが変換処理する．このタスクが Spark クラスタ内の各ノードに分散配置され，並列で変換処理が行われる．各タスク内ではシャッフルを伴わない変換処理が行われ，シャッフルが必要になった時点でそのタスクは終了し，シャッフル後は新しいタスクで処理が行われる．シャッフルが不要な範囲の連続した RDD の変換をまとめて「ステージ」と呼び，ステージ内の全タスクが完了しないと次のステージに進むことはできない．

3　評価手段

本章では，実際に評価を行う手段について説明する．評価を行うベンチマークとしては、既存研究をもとにデータセットを自作し，そこで並列（処理スレッド）数・キーの偏り・時間窓の幅の 3 点をパラメータとして変化させて実行時間を計測後、スループットを計算する．入力用データのタプル数は 1 億個で，並列数は実行時のコアの数を増加させていくことで変化させる．キーの偏りは入力用データのスキューの値を変化させることで発生させ，時間窓はプログラムの TimeWindow の幅の値を変化させて調整する．そして，タプル数 1 億を計測した実行時間で割ることでスループットを計算する．

本実験で使用するマシンの性能については表 1 に示す．

4　おわりに

本稿では，既存の分散並列ストリーム処理 OSS のメニーコア CPU における処理性能の調査とその比較について議論した．各既存 OSS について調査し，それらの性能を比較するための手段について検討した．今後は本稿で述べた性能調査を実際に実施してそれらを評価していく予定である．

謝辞

本研究は，JSPS 科研費（16H01722, 19K21530）の助成，および，国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の委託業務による．

参考文献

[1] P. Carbone, S. Ewen, G. Fóra, S. Haridi, S. Richter, and K. Tzoumas, “State management in Apache Flink®: consistent stateful distributed stream processing,” *PVLDB*, vol. 10, no. 12, pp. 1718–1729, 2017.

[2] S. A. Noghabi, K. Paramasivam, Y. Pan, N. Ramesh, J. Bringhurst, I. Gupta, and R. H. Campbell, “Samza: stateful scalable stream processing at LinkedIn,” *PVLDB*, vol. 10, no. 12, pp. 1634–1645, 2017.

[3] A. Toshniwal, J. Donham, N. Bhagat, S. Mittal, D. Ryaboy, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, and M. Fu, “Storm @Twitter,” in *Proc. SIGMOD*, pp. 147–156, 2014.

[4] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica, “Discretized streams: Fault-tolerant streaming computation at scale,” in *Proc. SOSP*, no. 1, pp. 423–438, 2013.