サーバシステムの性能データ収集および転送効率化に向けた検討

飯山 知香 (指導教員：小口 正人)

1 はじめに

クラウド環境をはじめとして，多数台サーバの共有利用や分散処理利用に関する需要が増えてきている．このような環境で，負荷分散およびシステムやアプリケーションのチューニングを行うには，各サーバの低レイヤを含めた性能データを低オーバヘッドで収集してリアルタイムに分析・提示する手法が必要である．

また，多数台のサーバのデータを一元的にリアルタイムで分析する際，分析対象のサーバとその分析を行うサーバは分割されることが多い．しかし，そこで用いられる Linux の性能データなどの時系列データはデータサイズが比較的大きく，扱う際のオーバヘッドが大きくなる可能性があるため，効率的に扱う手法の実現が求められている．

既存研究 [1] では，アプリケーションの実行後に，各ノードで収集した性能データをファイルベースで収集・分析する手法が提案されているが，データ収集と転送・分析を同時に行うことは困難である．

そこで我々は，CPU や OS の性能データを時系列で切り出してデータ解析用サーバに転送し，データベースに格納しながら分析する手法を検討中である．

今回，時系列データの中でもサイズが大きく処理に時間のかかる CPU のプロファイルデータに関し，時系列 DB およびその API を利用して転送するケースを想定した実機評価を行い，課題やオーバヘッドを見積もり，今後のデータ収集および転送の効率化の改善事項を抽出した．

2 実験

2.1 実験概要

性能データ収集/転送の効率化を検討するにあたり，まず予備実験として，データ通信のオーバヘッドを計測する実験を行い，サーバシステムの性能データを収集/転送/蓄積する際のリソース利用量，オーバヘッドを計測・分析した．

環境構築として，データ収集用サーバ (以下収集サーバ) とデータ解析用サーバ (以下解析サーバ) の 2 台サーバを用意した．収集サーバには，Linux Kernel の標準的なイベントデータ収集/トレース機能のフレームワークである perf[2] や，様々なベンチマークを使用して CPU やメモリなどに負荷をかけられるツールである stress[3] を使用した．解析サーバには，収集した大量の時系列データを管理する時系列 DB として InfluxDB[4] を使用した．InfluxDB については [5] で解説されている．

今回は収集サーバにて CPU 性能データ (perf record によって取得したプロファイルデータ) を 1 ミリ秒単位で 1 秒間収集し，解析サーバ上の InfluxDB に転送した．その際，収集サーバでのデータ転送量や転送時間，解析サーバ上での DB 増加量を計測した．1CPU 分のデータを増加させた際の転送時間と比較するため，収集サーバにて収集する CPU 性能データを 100 マイクロ秒単位にする実験も行った．概要を図 1 に示す．

perf record により収集したデータから，必要なデータ (時刻，プロセス ID，スレッド ID，実行アドレス) のみを抽出し，InfluxDB に転送可能なデータ形式に変換した．そのデータを InfluxDB の Python Client モジュールを利用してデータ解析サーバに転送した．この時，1CPU(core) の 1 秒分に相当する 1000 レコード分 (100 マイクロ秒単位で収集した場合は 10000 レコード分) の抽出データを，DataFrame 形式 (Python Pandas を利用) に格納した．InfluxDB Client モジュールには，Pandas の DataFrame 形式データを InfluxDB サーバに書き込む API(Influxdb.DataFrameClient) があり，今回はこれを利用している．DataFrame 形式に格納したデータのサンプルを図 2 に示す．

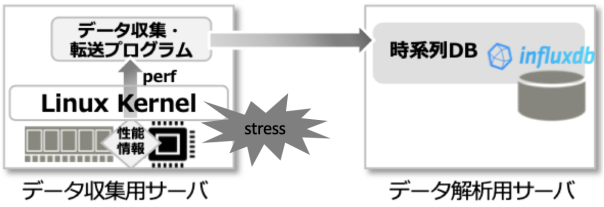

図 1: 実験概要

時刻(ns単位)	Process ID	Thread ID	EIP(実行アドレス)
2021-12-20T13:18:36.159557443	698085	698085	0x55e3e9dd1151
2021-12-20T13:18:36.160568219	698085	698085	0x7faa24bd7e02

図 2: DataFrame サンプル

2.2 結果

1CPU 分で転送を実施した際は，データ転送に約 0.051 秒かかった．2，10，20CPU 分繰り返して転送を行なった際には，それぞれ約 0.085 秒，約 0.40 秒，約 0.80 秒に増えた (転送時間はすべて 10 回の計測の平均)．1 回分の転送用データのサイズは約 32KB で変化なく，それぞれ合計約 64KB，約 320KB，約 640KB 転送された．繰り返し転送をする際には，InfluxDB の Database へは，measurement(表の名前) を CPU0，CPU1，CPU2，というようにして転送した．この際，データ転送にかかる時間は CPU 数によって変動するが，転送用データのサイズは，それぞれ 64bit の時刻，プロセス ID，スレッド ID，実行アドレスの 4 項目のデータが 1000 レコード分ある 1CPU 分の値で不変であるため，約 32KB で一定となっている．各 CPU 数ごとの，データ転送にかかった時間を図 3 に示す．

100 マイクロ秒単位で CPU 性能データを収集し，1CPU 分のデータを増加させて転送を行った際には，1CPU 分の転送には約 0.21 秒かかった．1 ミリ秒単位でデータ収集および転送を行った時と同様に，2，10，20CPU 分の転送も行った．その結果，それぞれ約 0.57 秒，約 2.0 秒，約 3.9 秒に増え，元の 4〜7 倍ほどになった．1 回分の転送用データのサイズは約 320KB と元の 10 倍になった．各 CPU 数ごとの，1 ミリ秒/100 マイクロ秒単位のデータ転送にかかった時間の比較を図 4 に示す．

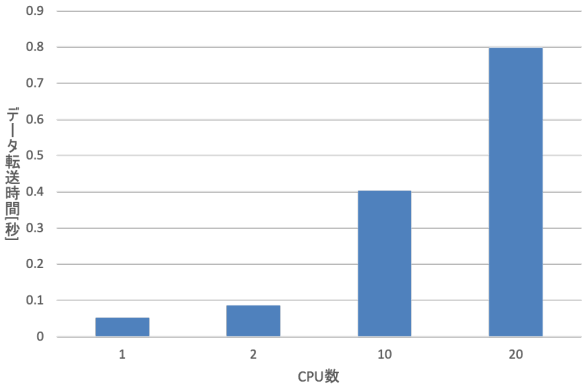

図 3: データ転送時間 (1 ミリ秒単位)

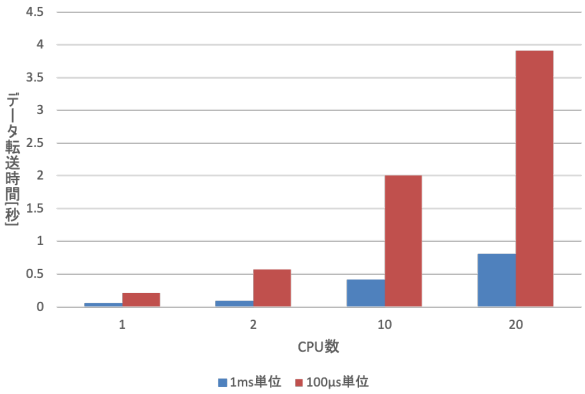

図 4: データ転送時間比較 (1 ミリ秒/100 マイクロ秒単位)

解析サーバ上での DB 増加量も計測した．InfluxDB を構築した解析サーバのデータ格納用ディレクトリ下のファイルサイズ増加量を観察した．1 ミリ秒単位で 1CPU 分の性能データ収集および転送を行った際，収集サーバからデータ自体は約 32KB 転送されているが，解析サーバでの DB 増加量は約 13.9KB となり，転送データの半分以下になった．転送前後の DB ファイルサイズを表 1 に示す．今回取得した perf のプロファイルデータは，プロセス ID やスレッド ID が同一値または近い値が連続するため，データを格納する際の圧縮効果が高くなっていると考えられる．

表 1: 転送前後の DB ファイルサイズ

転送前のファイルサイズ	58bytes
転送後のファイルサイズ	13895bytes

以上の結果から，1CPU 時の転送時間は約 0.05 秒，2CPU 以上では 1CPU あたり約 0.04 秒増加しており，25CPU(core) 以上の多数コア CPU では perf record の 1 秒間相当のデータを転送するのに 1 秒以上かかることが分かる．収集する性能データの粒度を 10 倍にした実験からは，1CPU 分のデータを 10 倍にしても転送時間は 10 倍にはならず，4〜7 倍程度にしか増加しないことが分かった．データ転送には InfluxDB の python モジュール API を使用しており，1CPU のデータ転送に対して，転送データのサイズに関わらず 1 度だけ呼び出しを行っている．転送するデータのサイズを 10 倍にしても 4〜7 倍程度にしか転送時間が伸びなかったことから，データ本体の転送以外の，InfluxDB を載せたサーバ (解析サーバ) とのネゴシエーション等のコストが大きく，時間がかかっていると推察される．

これらの結果より，多数コアの CPU 性能データを含む時系列で切り出して転送するには，より効率的な手法を検討する必要がある．またデータ収集と転送を同時に行うことを想定しているため，出来るだけ CPU 負荷などの外乱をかけない効率化手法が必要である．現時点では，有用な転送効率化手法として，データ転送用プログラム (ライブラリ，モジュール) の高速化，転送データの簡易分析，転送データの圧縮などの手法を検討している．

3 まとめと今後の課題

性能データの収集および転送の効率化手法を検討するにあたり，まずサーバシステムの性能データを収集/転送/蓄積する際のリソース利用量やオーバヘッドを計測・分析した．その結果，通常の InfluxDB の python モジュールを使用した転送方法では，多数コア CPU において 1 秒間相当のデータの転送に 1 秒以上かかってしまうため，データ収集と転送を同時に行うにはより効率化が必要であることが分かった．1CPU 分の転送データのサイズを 10 倍にした実験からは，データ本体の転送以外にかかる，InfluxDB サーバとのネゴシエーション等のコストが大きい (時間がかかる) ことも分かった．また，転送先の解析サーバでの DB 増加量は転送したデータの半分以下であったため，今回取得した perf のプロファイルデータは圧縮効果の高いデータだったと考えられる．

本研究では，転送効率化のため，データ転送用プログラム (ライブラリ，モジュール) の高速化や転送データの簡易分析，転送データの圧縮などの手法を検討した．今後は，今回検討した効率化手法を試行していくとともに，InfluxDB サーバとのネゴシエーション等の詳細や，どのようなプロファイルデータでどの程度圧縮が効くのかについても調べていく．

謝辞

本研究の一部はお茶の水女子大学と富士通研究所との共同研究契約に基づくものである．

参考文献

[1] 阿部文武, 中村 朋健, and 志田 直之. プロファイラにおける汎用的な cpu 性能情報と表示機能. 研究報告ハイパフォーマンスコンピューティング *(HPC)*, 2016(18):1–8, 2016.

[2] perf. https://perf.wiki.kernel.org/index.php/Main*page*.

[3] stress. https://linux.die.net/man/1/stress.

[4] influxdb. https://www.influxdata.com/products/influxdb/.

[5] S.N.Z. Naqvi, S. Yfantidou, and E. Zimányi. Time series databases and influxdb, université libre de bruxelles. 2017.