Vol.2017-HPC-160 No.33
2017/7/28

同期複数タスク実行フレームワークSMTEFを用いたメニータスク並列ベンチマークとその評価

岩井 厚樹[1,a)]　建部 修見[2]　田中 昌宏[2]

概要：これまで並列ベンチマークを行う際はMPIを用いた単一タスクのマイクロベンチマークの使用が一般的だった．一方で，サイエンスワークフローアプリケーションではワークフローエンジンを用いてタスクを複数ノードで多数実行し，解析やシミュレーションを行っている．しかし，ワークフローエンジンを用いて多数のタスクを並列に実行してシミュレーションや解析を行うサイエンスワークフローアプリケーションに適した並列ベンチマークはこれまでなかった．同期複数タスク実行フレームワークSMTEFでは多数のタスクを同時に起動して分散環境の性能評価を行う仕組みを提供する．本論文ではそのSMTEFを用いて分散環境におけるメニータスク並列ベンチマークを行い，その評価を通してタスクを一斉に実行した際の性能やワーカーを増やして実行した際の計算機の振る舞いについて調査する．

1. はじめに

天文学や物理学などの分野におけるデータ量は実験装置の性能向上や，スーパーコンピュータによるシミュレーションなどにより増加の一途を辿っている．例えば，気象の分野では，30秒間で200GBものデータの入出力がある[1]．そのような大規模データを解析することで結果を得る手法をデータインテンシブサイエンスと呼ぶ．データインテンシブサイエンスでは複数のノードで多数の処理がタスクという単位で実行されている．例えば，Montage[2]やCyberShake[3]などのサイエンスワークフローアプリケーションがあるが，それらはPwrake[4]のようなワークフローエンジンを用いて多数のタスクが同時に実行されている．

しかし現在，そのようなタスクが多数起動するようなアプリケーションに適したベンチマークの仕組みが存在しない．IOR[5]やmdtest[6]などのマイクロベンチマークがあるが，それらはMPIを用いた並列ベンチマークであり，ワークフローエンジンを用いてタスクを並列に実行するサイエンスワークフローアプリケーションとはベンチマークの対象アプリケーションが異なる．そのため，同時に多数のタスクを実行して分散環境における計算機のパフォーマンスを測定できる並列ベンチマークが求められている．

そこで，我々は複数ノードにおいて多数のタスクを同時に実行して，計算機のパフォーマンスを測定するメニータスク並列ベンチマークを行うための枠組みである同期複数タスク実行フレームワークSMTEF(Synchronous Multiple Task Execution Framework)の設計および開発を行った[7]．SMTEFでは実際のサイエンスワークフローアプリケーションの実験にも使用されているPwrakeを元に開発を行った．SMTEFでは単に複数ノードでタスクを実行させるだけでなく，各ベンチマークタスクの実行開始時刻を同期して揃える工夫がなされている．これにより，メタデータサーバのような管理システムに対して一定時間で集中的に負荷をかけるようなベンチマークが可能となる．

本論文では実際のワークロードで想定されるファイルのI/O，メタデータ操作をSMTEFを用いてメニータスク並列ベンチマークの実行および評価を行う．これにより，タスクを一斉に実行した時の最大性能や，ワーカーを増やした時の実験環境の振る舞いを知ることができる．

以下2章では，関連技術について述べる．3章では，同期複数タスク実行フレームワークSMTEFについて解説する．4章では，SMTEFを用いたメニータスク並列ベンチマークについて述べる．5章では，まとめと今後の課題について述べる．

2. 関連技術

複数のノードにログインしてタスクを同時実行するのであれば，SMTEFではなく，並行実行可能なSSHを用いる考えもある．しかし，並行実行可能なSSHで大規模環境下においてメニータスク並列ベンチマークをしようとした場

[1] 筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻
[2] 筑波大学計算科学研究センター
[a)] iwai@hpcs.cs.tsukuba.ac.jp

合，幾つか問題がある．ここでは並行実行可能な SSH の調査およびメニータスク並列ベンチマークを実行しようとした際の問題点について述べる．

- pssh，dsh
  dsh[8] および pssh[9] はコマンドもしくはファイルで指定した複数のノードにログインし，コマンドを複数ノードで同時に実行する．dsh および pssh は一つのコマンドが終了するたびに接続を切るため，その都度多数のノードに接続しなければならないという問題点がある．
- ClusterSSH
  ClusterSSH[10] はログインノード数分の xterm を起動し，それらに単一のコンソール画面からコマンドを送信し，複数のノードを一括で制御することができる．ログインすれば各ノードと接続を保ったままだが，全接続先の xterm を表示するためログイン先のノードが一定数を超えると使いづらいというデメリットがある．
- GXP
  GXP[11] は多数のノードでプログラムを瞬時に並行実行することを目標として開発された分散環境向けのシェルである．GXP は，GXP make[12] という GXP を拡張した Make に接続を保ったまま複数回同期処理ができるが，GXP make には個々のタスクに対してノードを指定して実行する機能がない．そのため，データ配置を考慮したタスクを実行できない．

3. SMTEF について

同期複数タスク実行フレームワーク SMTEF とは，メニータスク並列ベンチマークのためのフレームワークである．大規模環境において，多数のタスクを一斉に起動し計算機のパフォーマンスを測るベンチマークの仕組みはこれまでなかった．SMTEF はユーザが定義したタスクを複数のノードで実行することでサイエンスワークフローアプリケーションを想定したメニータスク並列ベンチマークを可能にする．SMTEF では単にタスクを複数ノードで実行するだけでなく，開始時間の同期をするようになっている．これにより，メタデータサーバのようなシステムに対して同時間帯に一斉に負荷をかけるような並列ベンチマークが可能になる．

図 1 に SMTEF の動作概要図を示す．図中において，楕円はタスクを表し，矢印は動作の流れを示す．まず，同期タスクが実行され，それが完了すると各ワーカーでタスクが実行される．各ワーカーでタスクを実行後，再び同期タスクが実行され，同期処理を行う．これを繰り返すことで，タスクの開始点を揃え，一斉に同時にタスクを開始することが可能となる．

SMTEF は Pwrake を用いて実現されている．Pwrake とは Ruby 版の Make である Rake[13] に並列分散実行機能を

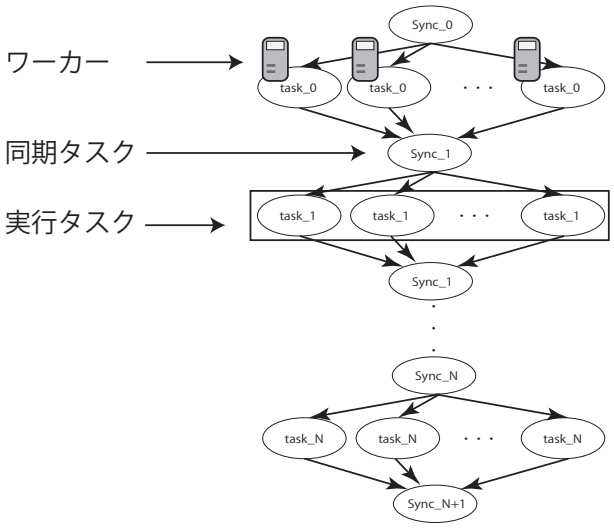

図 1 SMTEF の概要図

拡張したワークフローエンジンである．Pwrake は Rakefile の記述に沿ってワークロードを実行する．SMTEF では多数のタスクを同時に実行し，また同期処理をできるよう Rakefile を作成した．ユーザは SMTEF の設定ファイルに実行ファイルや引数，変数を与えるだけでメニータスク並列ベンチマークを実現できる．また，Pwrake は Gfarm ファイルシステム [14] と協調することで局所性の高いスケジューリングにより，高い I/O 性能を発揮する処理を可能にする．よって，そのようなパフォーマンスを要求するアプリケーションに適したメニータスク並列ベンチマークも可能である．

現時点での SMTEF は，タスクを複数のワーカーで実行し，同期処理を施して開始時間を揃える処理しか提供していない．そのため，タスクの処理時間および処理結果を得るには Pwrake のログを参照する，もしくはタスクの出力を参照しなければならない．

3.1 SMTEF の使用方法

SMTEF は幾つかの環境変数を通じてパラメータを指定することができる．パラメータは SMTEF の設定ファイルに記述するか，もしくは SMTEF の実行コマンド引数に与えることで指定可能となっている．以下にそのパラメータ一覧を示す．

- *WORKER_COUNT*
- *TASK_FILE*
- *PARAMETER_ARRAY*
- *ARGUMENTS*

WORKER_COUNT は，SMTEF がタスクを割振る対象のワーカー数を示す．割振るワーカー一覧は Pwrake の実行時に必要となるホストファイルを参照する．

TASK_FILE には，ワーカーで実行するタスクが記述されたファイルの相対パスもしくは絶対パスを指定する．注

表 1　評価環境

CPU	Intel(R) Xeon(R) E5620 CPU 2.40 GHz x2
コア数	8 (4cores/socket)
メモリ	24GB
OS	CentOS 6.8
ストレージ	Fusion-io ioDrive 160GB SLC
ネットワーク	IPoIB
Gfarm	ver. 2.7.3
Ruby	ver. 2.3.1
Pwrake	ver. 2.2.1
gfarm2fs	ver. 1.2.10

意点として，SMTEF はワーカーに Rakefile のみを送信するため，タスクファイルはあらかじめ NFS や Gfarm ファイルシステムなどで共有しておく必要がある．

PARAMETER_ARRAY には，ベンチマークで使用する変数一覧を与える．例えば，ブロックサイズを変化させながらバンド幅を測定しようとする際には，*[”4k”, ”8k”, ”16k”, . . .]* のように記述できる．

ARGUMENTS には実行タスクの引数を与える．

4. SMTEF を用いた評価

4.1 評価環境

評価に使用した環境は**表 1** の通りである．使用ノード数は 8 とし，そのうち 1 ノードをメタデータサーバ，残りの 7 ノードを計算ノード兼ファイルシステムノードとした．Pwrake はホストファイルに使用ノードとそのノードの使用コアを記述でき，今回の評価ではノード数 7，それぞれのコア数は 8 なので，最大ワーカー数は 56 とする．

並列 I/O バンド幅の評価および IOPS 評価の際のワーカー数は [2，3，4，5，6，7，14，21，. . . ，49，56] のように変化し，ワーカー数 2 から 7 ではノードあたり 1 ワーカーとし，合計ワーカー数 14 以降ではノードあたりのワーカー数は合計ワーカー数を 7 で割った数となっている．

4.1.1 同期オーバーヘッド

SMTEF を用いた評価の準備として，SMTEF の同期オーバーヘッドを計測した．同期オーバーヘッドとは，SMTEF の実行において複数のワーカーにタスクを分配した際に発生する本来のタスク処理とは関係ない処理時間である．計測方法としては，何もしない NULL タスクを各ワーカーで実行し最初に開始した NULL タスクの開始時間と最後に終了した NULL タスクの終了時間との時間差とした．NULL タスクのため，理想的には全ワーカーで実行しても 0 秒になるはずだが，タスクを他のワーカーに分配する際の遅延が影響し，オーバヘッドが発生する．同期オーバーヘッドの評価結果を**図 2** に示す．最大ワーカー数 56 でおよそ 0.05 秒という結果が得られた．よって，タスクを全ワーカーで一斉に実行した時に，最初に開始したタスクの開始時間と最後に終了したタスクの終了時間の差が 5 秒以上であれば，タスクを実行中の 99%は同時間帯に実行されているということになる．その間に実行されたタスクは同時間帯に実行されているため，タスクを並列に起動した時の計算機への負荷を測ることができる．

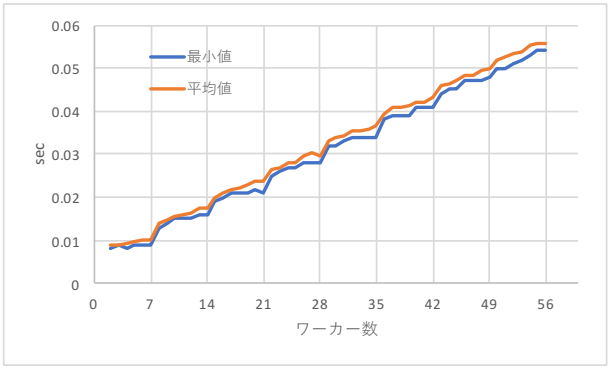

図 2　同期オーバーヘッド

4.2 並列 I/O バンド幅

SMTEF を用いてローカルディスクへの並列 I/O バンド幅を評価した．評価方法は，各ワーカーが起動しているノードに接続している ioDrive にファイルサイズ 5GB のファイルの読み書きに要した処理時間からバンド幅を求めた．読み込み性能はバッファキャッシュを有効にしてある．

評価結果を**図 3** に示す．書き込みの場合，ワーカー数 2 から 7 の時はノードあたり 1 ワーカーのため，並列バンド幅は ioDrive の理論性能 (750MB/sec) にワーカー数を乗算した性能値となっている．ワーカー数 14 以降はワーカー数を増やしても ioDrive の理論性能で頭打ちとなっていることがわかる．読み込みの場合，ワーカー数 28 を境に性能が劣化している．これは，ワーカー数 28 以下では 1 ノードあたりワーカー数が 4 以下のため読み込み容量が 20GB 以下となりメモリサイズを下回っているが，合計ワーカー数 35 を超えるとノードあたりのワーカー数が 5 以上となり，読み込み容量がメモリ容量を超えるため，キャッシュサイズに収まらず性能劣化が発生していると考えられる．

以上の結果より，実際のアプリケーションではワーカーを増やした際のメモリやストレージなどの性能限界を考慮してワークロードを構築する必要があるといえる．

4.3 IOPS 評価

SMTEF を用いて実験環境における IOPS 性能を評価した．およそ 2 万個のディレクトリを作成数は固定した状態でワーカー数を増やしながら mkdir コマンドを用いて作成した．この時，Gfarm ファイルシステムのバックエンドデータベースである PostgreSQL およびジャーナリングを有効にした状態と，それらを全て無効化に設定した状態の 2 種類について評価した．ディレクトリ作成先は Pwrake において Gfarm ファイルシステムを自動マウントするディ

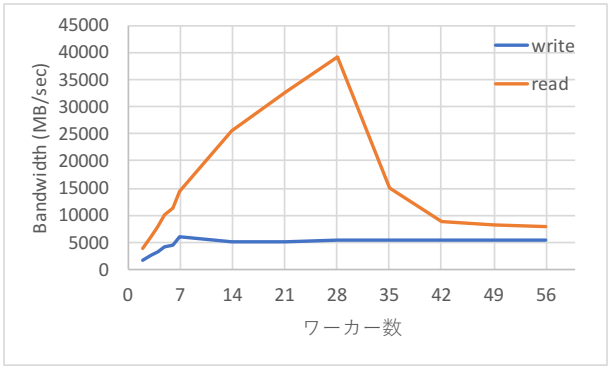

図 3　ローカル I/O バンド幅

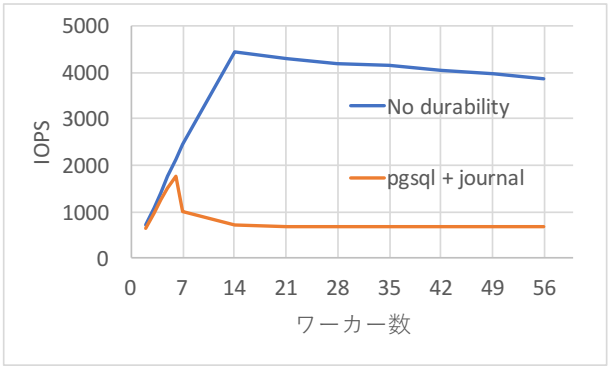

図 4　ディレクトリ生成の IOPS

レクトリとした．なお，マウントポイントは ioDrive 上とする．Pwrake はホストファイルに記述したコア数分マウントするため，ワーカー毎にマウントポイントを分けることでディレクトリを並列に作成することができる．

ディレクトリ作成数は固定して，ワーカー数を増やしながら IOPS 性能を評価した結果を図 4 に示す．青のラインが PostgreSQL およびジャーナリングを無効化した状態，オレンジのラインがそれらを有効化した状態である．また，無効化状態でのワーカー数 14 およびワーカー数 56 での並列度を図 5 および図 6 に示す．そして，無効化状態でのワーカー数 14 とワーカー数 56 において，ワーカーごとのディレクトリ作成に要した時間のグラフを図 7 に示す．

図 4 より，PostgreSQL およびジャーナリングを有効化している状態ではワーカー数 7 で性能が劣化していることがわかる．無効化した状態ではワーカー数 14 までは性能が向上しているが，ワーカー数 21 からは性能が劣化している．図 5 および図 6 より，どちらも並列にワーカーがタスクを実行していることがわかる．図 7 を見るとワーカーあたりのディレクトリ作成数が少ないワーカー数 56 の方が割り当てられたディレクトリ数の作成に時間を要してしまっており，これが図 4 の無効化状態においてワーカー数 14 以降で性能が劣化してる要因である．原因は Gfarm ファイルシステムのメタデータ性能の限界だと考えられる．それでも，無効化状態では 1 つのノードで 1 つのワーカーを起動するよりも，複数のワーカーを起動しディレクトリ作成をした方が性能が良いことがグラフより読み取れる．

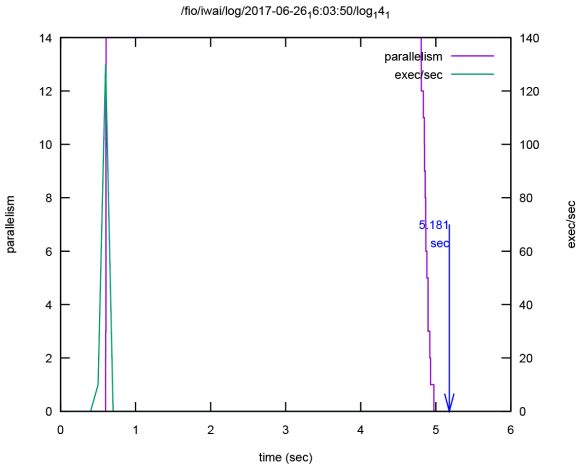

図 5　ワーカー数 14 での並列度

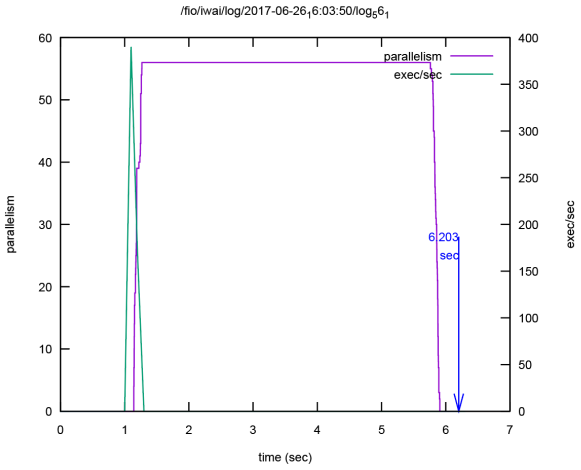

図 6　ワーカー数 56 での並列度

5. おわりに

本論文では，データインテンシブサイエンスに適したベンチマークを目標にして開発された SMTEF を用いて，メニータスク並列ベンチマークの実行およびその評価を行った．

データインテンシブサイエンスでは多数のタスクが並列に複数ノードで実行されているが，そのような環境に適した多数のタスクを起動して並列ベンチマークをするような仕組みはなかった．同期複数タスク実行フレームワーク SMTEF はメニータスク並列ベンチマークのフレームワークとなっており，これを使用することで複数ノードでタスクを多数起動して計算機のパフォーマンスを測ることが可能になる．

SMTEF を用いて評価を行ったところ，並列 I/O バンド幅測定では，書き込みの場合ストレージの特性に沿った性能を得ることができた．読み込みの場合，ワーカーを増やすとキャッシュに収まらず性能劣化するという結果を得

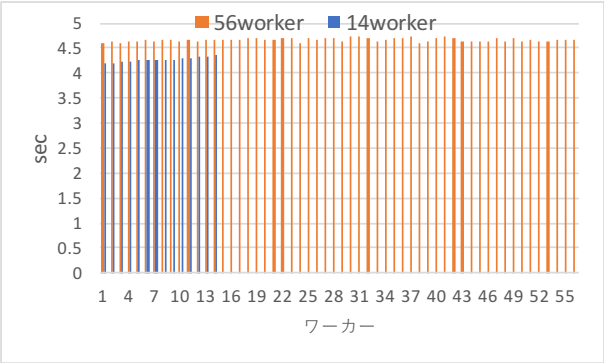

図 7 ワーカー数 14 と 56 の各ワーカーの実行時間

た．IOPS 評価では PostgreSQL およびジャーナリングの無効化状態において，ノードあたりのワーカー数が複数個の方がノードあたりのワーカー数 1 の時よりも性能が向上していることが確認できた．

今後の課題・目標としては，SMTEF にメニータスク並列ベンチマークの解析機能を拡張し，ボトルネックの調査や I/O の振る舞いを確認できるような総合的なベンチマークツールとすることを目指す．

謝辞 本研究の一部は，JST CREST「EBD：次世代の年ヨッタバイト処理に向けたエクストリームビッグデータの基盤技術」，JST CREST「広域撮像探査観測のビッグデータ分析による統計計算宇宙物理学」および JSPS 科研費 JP17H01748 の助成を受けたものです．

参考文献

[1] Matsuoka, S., Sato, H., Tatebe, O., Koibuchi, M., Fujiwara, I., Suzuki, S., Kakuta, M., Ishida, T., Akiyama, Y., Suzumura, T. et al.: Extreme Big Data (EBD): Next generation big data infrastructure technologies towards yottabyte/year, *Supercomputing frontiers and innovations*, Vol. 1, No. 2, pp. 89–107 (2014).
[2] Montage, http://montage.ipac.caltech.edu/.
[3] Graves, R., Jordan, T. H., Callaghan, S., Deelman, E., Field, E., Juve, G., Kesselman, C., Maechling, P., Mehta, G., Milner, K. et al.: CyberShake: A physics-based seismic hazard model for southern California, *Pure and Applied Geophysics*, Vol. 168, No. 3-4, pp. 367–381 (2011).
[4] Tanaka, M. and Tatebe, O.: Pwrake: a parallel and distributed flexible workflow management tool for wide-area data intensive computing, *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ACM, pp. 356–359 (2010).
[5] IOR, https://github.com/LLNL/ior.
[6] mdtest, https://sourceforge.net/projects/mdtest/.
[7] 岩井厚樹，建部修見，田中昌宏：並列ベンチマークのための同期複数タスク実行フレームワークの設計，研究報告ハイパフォーマンスコンピューティング (HPC)，Vol. 2017, No. 34, pp. 1–8 (2017).
[8] dsh, https://sourceforge.net/projects/dsh/.
[9] pssh, https://linux.die.net/man/1/pssh.
[10] ClusterSSH, https://sourceforge.net/projects/clusterssh/.
[11] 田浦健次朗：GXP: 分散環境を心地よく使う並列シェル，コンピュータ ソフトウェア，Vol. 27, No. 4, pp. 4_144–4_171 (2010).
[12] Taura, K.，Matsuzaki, T.，Miwa, M.，Kamoshida, Y.，Yokoyama, D.，Dun, N.，Shibata, T.，Jun, C. S.，Jun'ichiTsujii：Design and implementation of GXP make—A workflow system based on make，*Future Generation Computer Systems*, Vol. 29, No. 2, pp. 662–672 (2013).
[13] Rake, http://rake.rubyforge.org/.
[14] Tatebe, O., Hiraga, K. and Soda, N.: Gfarm grid file system, *New Generation Computing*, Vol. 28, No. 3, pp. 257–275 (2010).