Vol.2013-HPC-138 No.22
2013/2/22

広域分散ファイルシステムBlobSeer-wan/HGMDSの設計と評価

鷹津 冬将[1,a)]　平賀 弘平[1]　建部 修見[2]　Gabriel Antoniu[3]

概要：広域分散ファイルシステムでは，クライアントの場所にかかわらずファイルシステムにアクセスすることができる．しかしながら，既存の広域分散ファイルシステムは特定のメタデータサーバへのアクセスが必要であり，クライアントとメタデータサーバ間の遅延の大きさが性能を左右している．本稿では，拠点ごとにメタデータサーバ等を設置し，クライアントが行うすべてのオペレーションを拠点内で完結させる広域分散ファイルシステム BlobSeer-wan/HGMDS の設計・実装を行い，評価を行った．ひとつの拠点内における評価では，ほぼすべてのオペレーションについてクライアントが 8 ノードの場合の際に BlobSeer-wan/HGMDS が Gfarm に比べ高い性能を示した．特にファイルを作成しデータを書き込み同期を行う評価ではクライアントが 8 ノードの場合に Gfarm の性能の 10.9 倍の性能を示した．また，拠点間の遅延が大きい二つの拠点における評価においても，各拠点における性能差が小さいこと，及びクライアントのノード数に比例して BlobSeer-wan/HGMDS の性能が高くなることを示した．拠点ごとのクライアント数が 4 ノードの際における BlobSeer-wan/HGMDS の各拠点の結果の和の値は Gfarm のその値に比べ，Directory Creation で 2.5 倍，Fire Creation で 1.1 倍，Directory stat で 1.5 倍，File stat で 2.5 倍，File removal で 3.7 倍の値を示した．

1. はじめに

近年計算機における性能の向上や，価格低下により，多数の計算機を複数台並べることによって大規模なデータセンターを構築したり，大量のデータを解析処理する技術が確立されてきた．一方で，天文学や生命科学などの科学技術分野におけるデータインテンシブコンピューティングや e-サイエンスの分野では，実験装置の発展やシミュレーション規模の拡大等によって年々扱うデータ量が増加の一途をたどっている．また，これらの分野においては，観測装置や計算機等が広域に散在するため，これらの装置を複数の拠点間で効率的に扱う広域データ解析の要求が高まっている．

このように地理的に広域に分散した拠点間において大量のデータを共有するために，広域分散ファイルシステムを利用することが注目されている．しかしながら，既存のファイルシステムでは広域であることからクライアントとメタデータサーバ，ストレージサーバ間の通信がネットワークにおける遅延が非常に大きいことがボトルネックとなる．

また，メタデータサーバを複数台設置した場合においても既存のファイルシステムでは特定のメタデータサーバをマスターのメタデータサーバとして設定するため，マスターのメタデータサーバが故障した場合においてスレーブをマスターに昇格させるまでの時間などサービスを利用できない時間が発生する問題もある．

本研究では，ネットワークにおける遅延が非常に高い環境においても継続的に高い性能を示す広域分散ファイルシステムの実現を目指している．本稿では，分散データストレージ Blobseer [1] を広域においても高い性能を示すよう現在も開発されている広域分散データストレージである blobseer-wan と広域分散ファイルシステムのためのメタデータサーバである HGMDS [2] を用いることによって広域分散ファイルシステム BlobSeer-wan/HGMDS の設計および実装を行い，遅延の小さな拠点内における性能の評価および，拠点間の遅延の大きな二つの拠点における性能の評価を行った．

2. 関連研究

複数の拠点間で分散してデータをストアする方法としては Dynamo [3] という高可用性の Key-Value ストアがある．Key のハッシュ値を元にデータを保持するストレージ

[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] 筑波大学システム情報系
[3] INRIA
[a)] takatsu@hpcs.cs.tsukuba.ac.jp

ノードを選択することによって，データを分散して管理している．ストレージノードの決定にハッシュ値のみを利用していることから，高遅延環境においては遅延の影響を受ける場合があるため，広域で利用することには適さない．

リモートファイルシステムとしては NFS や Luster，Gfarm [4] などさまざまなファイルシステムがあり，現在も研究，開発が行われている．この中でも Gfarm は本研究で設計・実装を行った BlobSeer-wan/HGMDS と同様に広域に対応した分散ファイルシステムである．Gfarm はメタデータサーバを複数台持つことができ，そのメタデータサーバはマスターとスレーブとしてそれぞれ動作している．オペレーションの実行にマスターのメタデータサーバが必要となるため，マスターのメタデータサーバとクライアントの間の遅延の大きさが性能を左右している．

3. 設計と実装

3.1 BlobSeer-wan について

BlobSeer-wan は，分散データストレージである BlobSeer を広域向けに現在も開発されている広域分散ストレージである．Blobseer と同様にテラバイトスケールの大きなサイズのデータをストアすることができ，高いバンド幅のアクセス性能を保持している．またバージョン管理も行っており，過去のスナップショットに対してアクセスすることもできる．

BlobSeer-wan は，実際にデータをストアする Provider，複数の Provider において状態を管理したりロードバランシングを行う Provider Manager，ストアしているデータの page location を管理する Metadata Provider，およびバージョンを管理する Version Manager によって構成されている．Provider と Metadata provider は複数ノード設置することができ，広域においては各拠点ごとに設置することで高い性能を示す．

クライアント向けに API が提供されており，この API を用いることによってクライアントを開発することができる．

3.2 HGMDS について

HGMDS は，広域分散ファイルシステム HGFS のために実装された Multi-master のメタデータサーバであり，各ファイルにおける stat 構造体の内容について管理を行う．HGMDS はクライアントである hgfuse とメタデータサーバとなる hgmds によって構成される．hgfuse は FUSE [5] (Filesystem in Userspace) によって実装されており，各リクエストを msgpack-rpc [6] を使うことによって hgmds と通信させて処理している．hgmds は hgfuse からのさまざまなリクエストを受けて，各メタデータの作成・更新等を行うと，結果をクライアントに返した後，追加・更新されたメタデータを他の hgmds と通信を行い同期をとる．

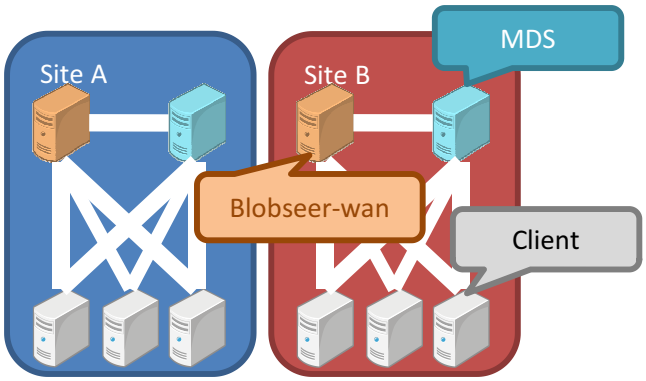


図 1 BlobSeer-wan/HGMDS の構成

拠点ごとにひとつの hgmds を設置することによって，拠点間のネットワークにおける遅延が高い環境においても高いメタデータ操作性能が示される．

メタデータは Vector Clock を用いていることでバージョンを管理しており，複数のメタデータサーバ間において衝突が起きた場合でも自動的に解決される．

3.3 BlobSeer-wan/HGMDS の設計と実装

先述の BlobSeer-wan と HGMDS を用いた広域分散ファイルシステム BlobSeer-wan/HGMDS を設計した．

ファイルシステムの構成を図 1 に示す．図に示されるように BlobSeer-wan/HGMDS においてはクライアントやメタデータサーバも複数台設置することができる．各クライアントは使用するメタデータサーバをひとつ指定し，さまざまなオペレーションに対して処理を行う．

クライアントに対して要求された各オペレーションにおける動作は以下のような設計となる．

(1) open()
open() はファイルを開くオペレーションである．クライアントは open() が要求されるとメタデータサーバに open に関するリクエストを送る．メタデータサーバはそのファイルに関するエントリがあればそのファイルのファイルサイズと blobseer-wan における ID をクライアントに返す．クライアントはメタデータサーバから結果が返ってくるとファイルの BlobSeer-wan におけるファイルの ID をファイルハンドラに書き込み，フルパスをキーとした map にファイルサイズを保持したのち，ユーザーに結果を返す．

(2) create()
create() はファイルを作成し，open() するオペレーションである．クライアントは create() が要求されるとメタデータサーバに create に関するリクエストを送る．メタデータサーバは blobseer-wan と通信を行い，blobseer-wan 上で新しいファイルを作成し，そのファイル ID を取得する．そしてファイル ID と stat 構造体などファイルのメタデータなどを含んだ新しいエン

トリを作成する．そしてその ID をクライアントに返した後，その情報を他のメタデータサーバと通信を行うことで同期する．クライアントはメタデータサーバから結果が返ってくるとファイルの BlobSeer-wan におけるファイルの ID をファイルハンドラに書き込み，フルパスをキーとした map にファイルサイズ（作成後なのでサイズは 0）を保持したのち，ユーザーに結果を返す．

(3) read()
read() は指定されたファイルにおいて読み込みを行うオペレーションである．クライアントは read() が要求されると直接 blobseer-wan と通信を行うことでファイルの中身を取得してユーザーに返す．

(4) write()
write() は指定されたファイルにおいて書き込みを行うオペレーションである．クライアントは write() が要求されると，クライアント内で保持しているファイルサイズを更新した後，書き込みに関わるページの書き換えない箇所を blobseer-wan から取得し，書き込むデータとともに blobseer-wan と通信を行うことでデータを書き込む．

(5) release()
release() は指定されたファイルを解放するオペレーションである．クライアントは release() が要求されると，そのファイルを open() してから書き換えたかどうかを判定する．書き込んでいない場合はクライアント内で保持しているファイルサイズの情報を破棄するが，書き込んだ場合はメタデータサーバと通信を行い，ファイルサイズを更新しファイルサイズの情報を破棄する．メタデータサーバはファイルサイズを更新する要求がくると，自身が保持するエントリを更新し，そのエントリをほかのメタデータサーバと通信を行うことで同期する．

(6) getattr()
getattr() は指定されたファイルの stat 構造体の値を取得するオペレーションである．クライアントは getattr() が要求されると，メタデータサーバに getattr に関するリクエストを送る．メタデータサーバは自身が保持する指定されたファイルのエントリの中から stat 構造体の中身を返す．クライアントはメタデータサーバから結果が返ってくると，自身がそのファイルのファイルサイズに関する情報を保持しているかどうかを判断し，保持している場合には返ってきた結果のファイルサイズに関する情報だけ更新して，ユーザーに返す．

各オペレーションにおいて設計したとおりの動作を行うように，クライアントに関する部分は hgfuse を，メタデータサーバに関する部分は hgmds を修正することよって実装を行った．

4. 評価

4.1 評価方法

評価には mdtest [7] を用いた．mdtest はオプションを以下の 3 種類のように変えることによって各性能を評価できる．

(1) (None)
これはオプションを指定しない方法である．つまり，ファイル（ディレクトリ）を作成し，stat 構造体を取得した後削除を行うのみである．

(2) -w 4096
オプションとして-w 4096 を指定すると，作成されたファイルに 4096 バイトのデータを書き込む．つまり，ファイル（ディレクトリ）の作成し，データを書き込んだ後 stat 構造体を取得して削除を行う．

(3) -w 4096 -y
さらにオプションとして-y を指定すると write() ごとに sync() を行う．つまり，ファイル（ディレクトリ）の作成し，データを書き込んだ後 sync をしてから stat 構造体を取得して削除を行う．

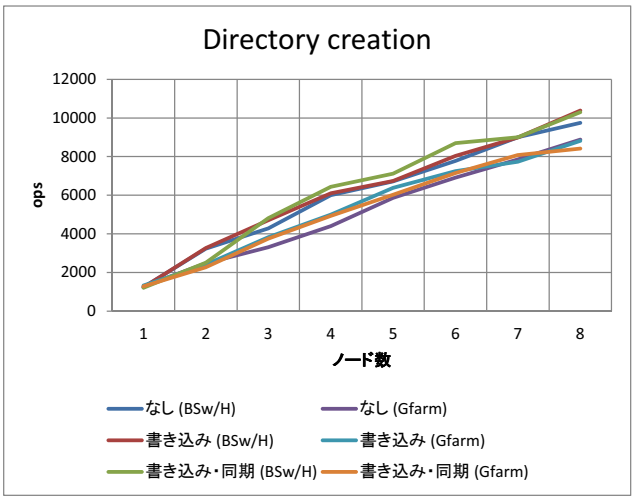
この評価ではファイルシステムに対してファイルやディレクトリをそれぞれ 1000 個ずつ作成し，作成や stat 構造体の取得，削除に伴う性能を測定し，1 秒間に処理できるオペレーション数について評価を行った．

性能の比較対象としては関連研究でも挙げた Gfarm を選択した．

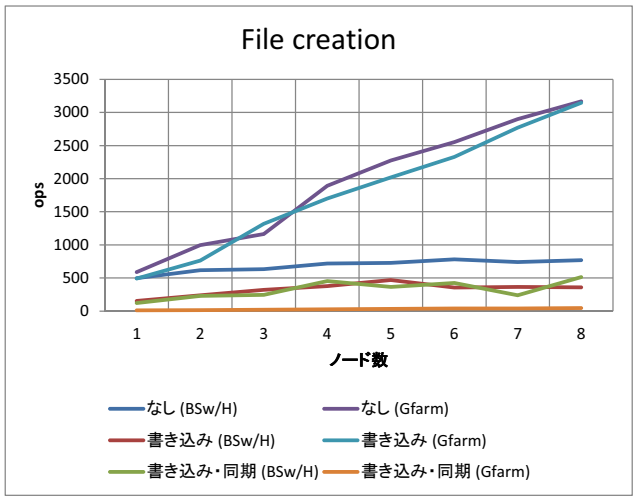
4.2 シングルサイトにおける評価

この評価では一つの拠点における BlobSeer-wan/HGMDS の性能を調査するために，InTrigger [8] の Tsukuba Site を利用し，図 2 に示す環境で評価を行った．mdtest のオプションとしては前述の 3 種類の方法を用いて評価を行った．評価に用いた各ノードの性能を表 1 に示す．図 2 に示すように BlobSeer-wan/HGMDS のメタデータサーバと BlobSeer-wan の各プロセスを 1 台のノード上で実行し，クライアントを他のノード上においてノード数を変化させつつ実行した．Gfarm についても同様にメタデータサーバのプロセスである gfmd とストレージサーバのプロセスである gfsd を 1 台のノード上で実行し，クライアントを他のノード上においてノード数を変化させつつ実行した．

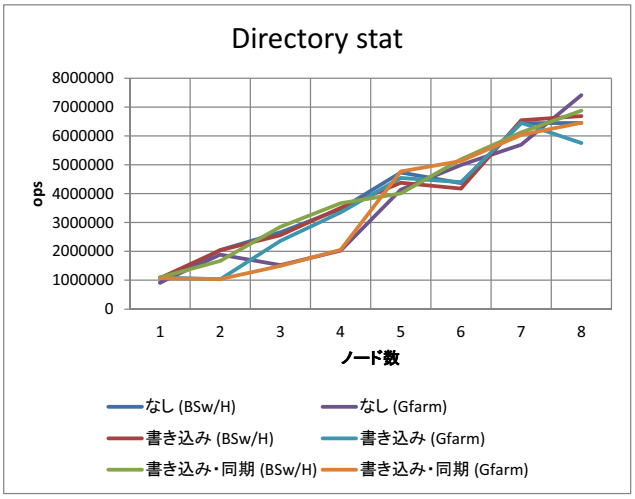
評価結果を図 3 に示す．図 3 中の各グラフでは，横軸にクライアントのノード数を示し，縦軸には 1 秒間に実行できたオペレーション数を表している．そのため，値が大きいほど性能が高いということを表している．また，各グラフ中で BSw/H と表記してあるデータが BlobSeer-wan/HGMDS のデータを表している．



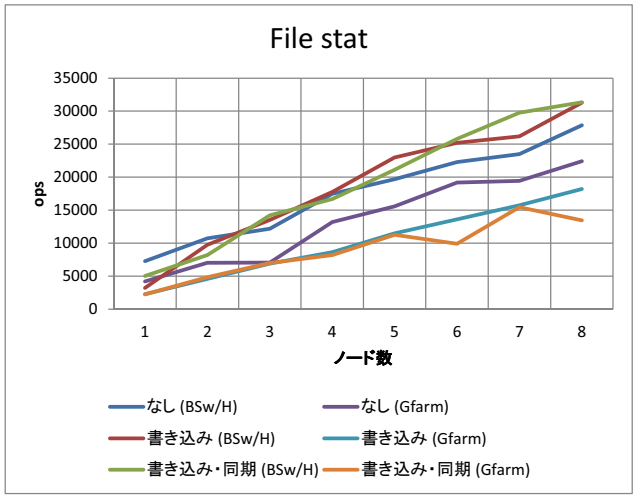
(a) Directory Creation



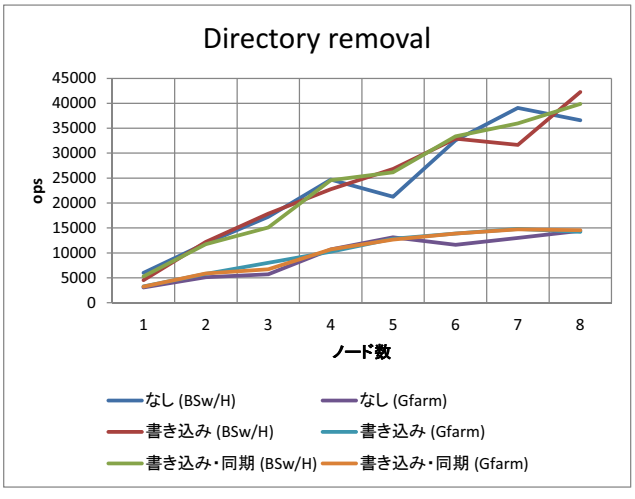
(b) File Creation



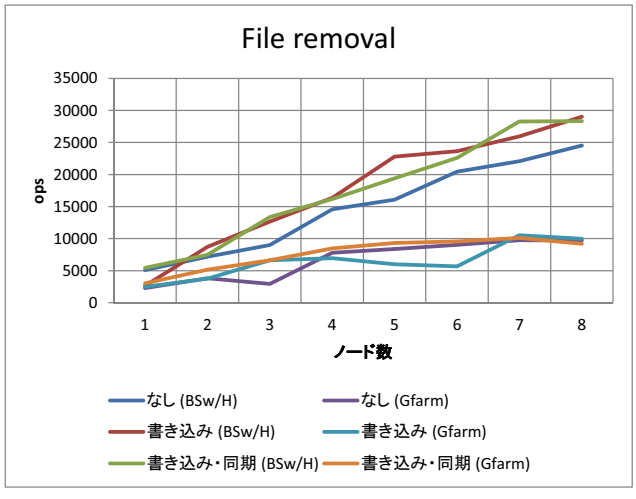
(c) Directory stat



(d) File stat



(e) Directory removal



(f) File removal

図 3 シングルサイトにおける評価結果

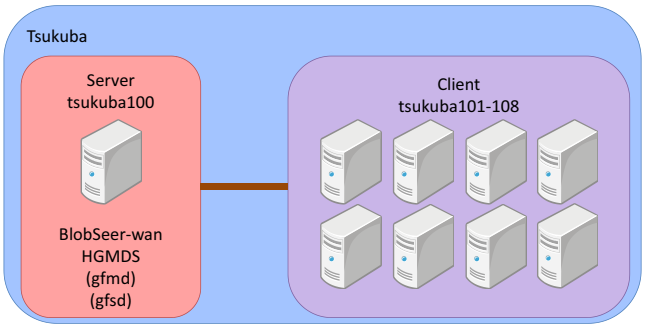


図 2 シングルサイトにおける評価の環境

表 1 InTrigger Tsukuba Site の各ノードの性能

CPU	Intel(R) Xeon(TM) E5620 CPU @ 2.40GHz (4 プロセッサ 8 スレッド) x2
メモリ	24GB
OS	Debian 5.0.7 (2.6.26-2-amd64)
NIC	GbE

File creation 以外のオペレーションの結果を表す図 5(a), 図 5(b), 図 5(c), 図 5(f) では，BlobSeer-wan/HGMDS の性能がクライアント数に比例して高くなることを表している．これらのオペレーションでは mdtest に対するオプションの付け方を変えても実行される操作は変わらないため，性能に差が示されなかった．Directory removal の結果を表す図 5(e) においても同様の結果が示されているが，BlobSeer-wan/HGMDS においてディレクトリの削除は実際には行っていないのでこの図が示す値はあくまでも参考値である．File creation の結果を表す図 5(b) においては，mdtest のオプションによる性能差が現れている．Fire Creation においてはクライアントが 8 ノードの場合に Gfarm の性能の 10.9 倍の性能を示した．

4.3 マルチサイトにおける評価

この評価では拠点間の遅延が大きい複数の拠点で BlobSeer-wan/HGMDS を運用した際における性能を測定するために InTrigger の Tsukuba Site と Kyutech Site を利用し，図 4 に示す環境で評価を行った．評価に用いた各ノードの性能を前述の表 1 に加え表 2 に示す．mdtest のオプションとしては前述の 3 種類の中からデータを書き込まない方法を用いて評価を行った．また，この評価では mdtest を拠点ごとに同時に実行した．そして，図 4 に示すように BlobSeer-wan/HGMDS のメタデータサーバと BlobSeer-wan の各プロセスを拠点ごとに 1 台のノード上で実行し，クライアントを他のノード上においてノード数を変化させつつ実行した．

Gfarm については，ストレージサーバのプロセスである gfsd を拠点ごとに 1 台のノード上で実行し，クライアントを他のノード上においてノード数を変化させつつ実行した．Gfarm のメタデータサーバは冗長構成を行った場合でもマスターのメタデータサーバのみが使用される設計になっているため，冗長構成ではなく Tsukuba Site でのみ gfmd を 1 台のノード上で実行した．

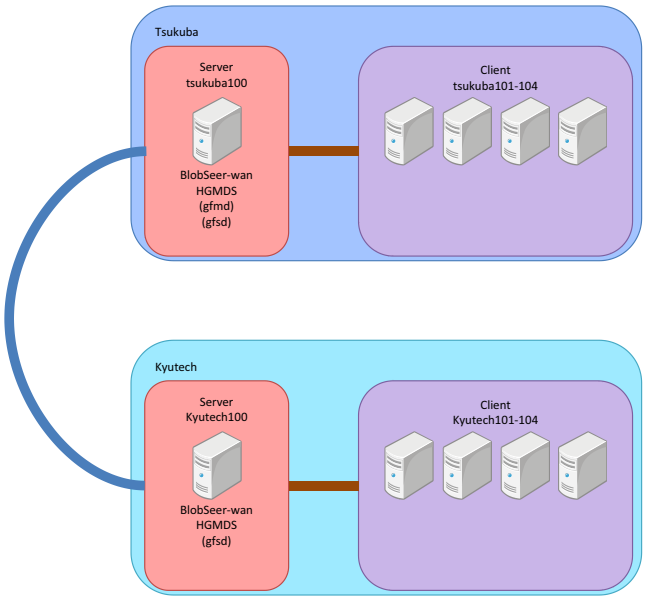


図 4 マルチサイトにおける評価の環境

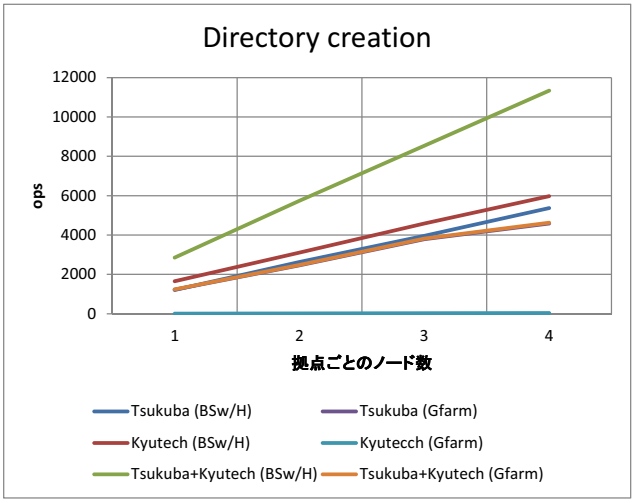
表 2 InTrigger Kyutech Site の各ノードの性能

CPU	Intel(R) Xeon(TM) E5410 CPU @ 2.33 GHz (4 プロセッサ 4 スレッド) x2
メモリ	32GB
OS	Debian 5.0.4 (2.6.26-2-amd64)
NIC	GbE x2

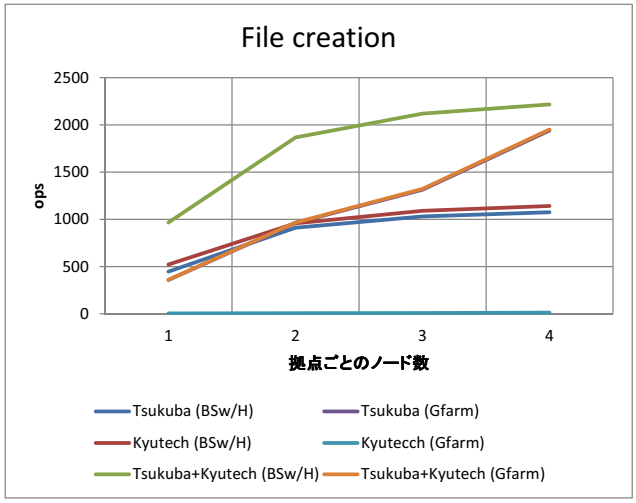
評価結果を図 5 に示す．図 5 中の各グラフでは，横軸にクライアントのノード数を示し，縦軸には 1 秒間に実行できたオペレーション数を表している．そのため，値が大きいほど性能が高いということを表している．また，この評価では拠点ごとに mdtest を実行したため，グラフでは各拠点の値とその和を表している．

この評価においても図 5(e) が示す Directory removal の結果は BlobSeer-wan/HGMDS においてディレクトリの削除は実際には行っていないのでこの図が示す値はあくまでも参考値である．

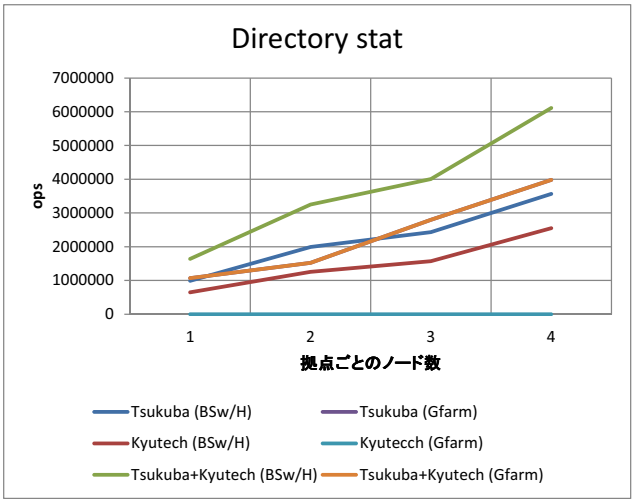
Gfarm の結果より，Tsukuba Site と Kyutech Site の結果を比較するとメタデータサーバが遅延の大きな拠点に設置されている場合に性能が低いことが示された．一方で BlobSeer-wan/HGMDS においては，Tsukuba Site と Kyutech Site の結果を比較するとふたつの拠点における性能差が少ないことから BlobSeer-wan/HGMDS が拠点間の遅延の影響を受けにくいことが示された．また，拠点ごとのクライアント数が 4 ノードの際における BlobSeer-wan/HGMDS の各拠点の結果の和の値は Gfarm のその値に比べ，Directory Creation で 2.5 倍，Fire Creation で 1.1 倍，Directory stat で 1.5 倍，File stat で 2.5 倍，File removal で 3.7 倍の値を示しており，BlobSeer-wan は広



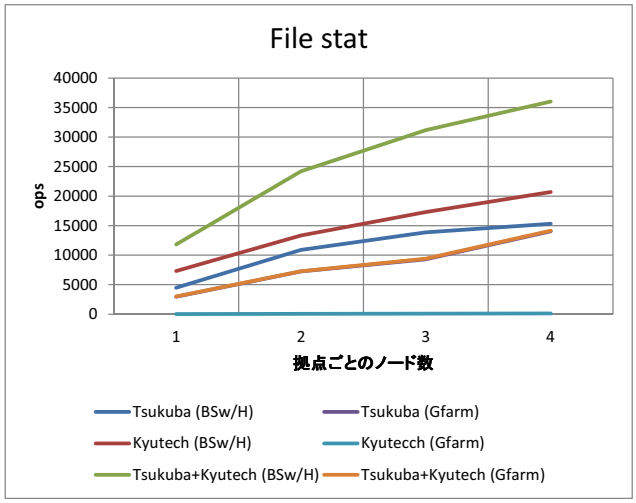
(a) Directory Creation



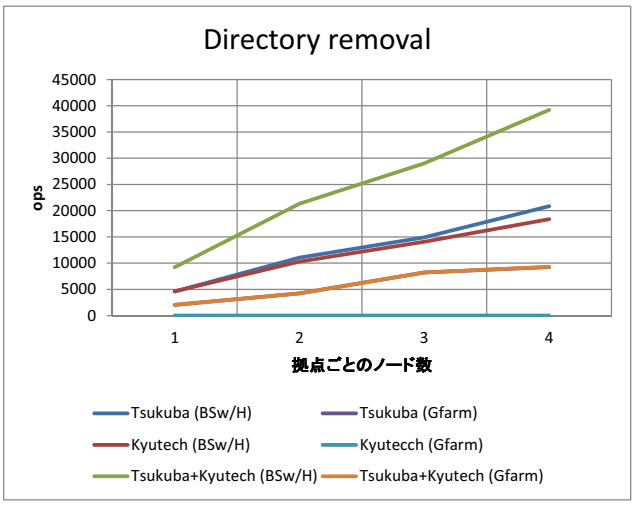
(b) File Creation



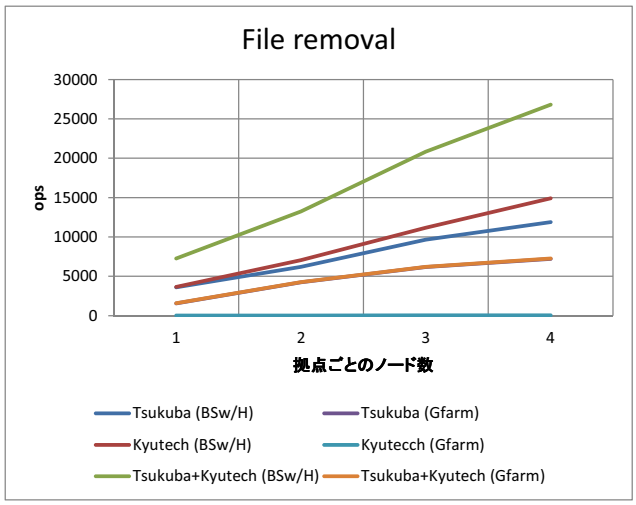
(c) Directory stat



(d) File stat



(e) Directory removal



(f) File removal

図 5 マルチサイトにおける評価結果

域においては Gfarm に比べて高い性能があることが示された．

5. まとめと今後の課題

本稿では，広域分散データストレージである blobseer-wan と広域分散ファイルシステムのためのメタデータサーバである HGMDS を用いることによって広域分散ファイルシステム BlobSeer-wan/HGMDS を設計・実装し，遅延の小さなひとつの拠点内における性能の評価と拠点間の遅延が大きい複数の拠点における性能の評価を行った．

ひとつの拠点内でクライアント数を増やした場合における評価では，BlobSeer-wan/HGMDS がクライアントのノード数に比例して性能が高くなることを示し，ディレクトリを削除するオペレーション以外のオペレーションについてクライアントが 8 ノードの場合において高い性能を示した．特にファイルを作成しデータを書き込み同期を行う評価ではクライアントが 8 ノードの場合に Gfarm の性能の 10.9 倍の性能を示した．

また，遅延の大きい複数の拠点間における性能評価では，拠点ごとのクライアント数が 4 ノードの際における BlobSeer-wan/HGMDS の各拠点の結果の和の値は Gfarm のその値と比べ，Directory Creation で 2.5 倍，Fire Creation で 1.1 倍，Directory stat で 1.5 倍，File stat で 2.5 倍，File removal で 3.7 倍の値を示しており，BlobSeer-wan は広域においては Gfarm に比べて高い性能があることが示され，各拠点における性能差が小さいことから BlobSeer-wan/HGMDS が拠点間の遅延の影響を受けにくいことが示された．

今後の課題としては BlobSeer-wan/HGMDS の安定性の向上，およびバンド幅の調査等が挙げられる．

謝辞 本研究の一部は科学技術振興機構「ポストペタスケールコンピューティングのためのフレームワークとプログラミング（FP3C）」による．

参考文献

[1] Nicolae, B., Antoniu, G., Bougé, L., Moise, D. and Carpen-Amarie, A.: BlobSeer: Next Generation Data Management for Large Scale Infrastructures, *Journal of Parallel and Distributed Computing*, Vol. 71, No. 2, pp. 168–184 (online), DOI: 10.1016/j.jpdc.2010.08.004 (2011).
[2] 平賀弘平，建部修見：広域ファイルシステム HGFS のための分散メタデータサーバの実装と性能評価，情報処理学会研究報告. [ハイパフォーマンスコンピューティング]， Vol. 2010, No. 29, pp. 1–9（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/110007995517/⟩ (2010).
[3] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P. and Vogels, W.: Dynamo: amazon's highly available key-value store, *SIGOPS Oper. Syst. Rev.*, Vol. 41, No. 6, pp. 205–220 (online), DOI: 10.1145/1323293.1294281 (2007).
[4] Tatebe, O., Hiraga, K. and Soda, N.: Gfarm Grid File System, *New Generation Comput.*, Vol. 28, No. 3, pp. 257–275 (2010).
[5] FUSE: Filesystem in Userspace, http://fuse.sourceforge.net/.
[6] msgpack-rpc, https://github.com/msgpack/msgpack-rpc.
[7] mdtest HPC Benchmark, http://mdtest.sourceforge.net/.
[8] InTrigger, http://www.intrigger.jp/.