2005 年度上期

オープンソースソフトウェア活用基盤整備事業

「OSS 性能・信頼性評価 / 障害解析ツール開発」

Java アプリケーション層

JBoss クラスタ再ベンチマーク

作成

OSS 技術開発・評価コンソーシアム

商標表記

- Linux は、Linus Torvalds の米国およびその他の国における登録商標あるいは商標です。
- MIRACLE LINUX は、ミラクル・リナックス株式会社が使用許諾を受けている登録商標です。
- その他記載の会社名、製品名は、それぞれの会社の商号、商標もしくは登録商標です。

目次

1 はじめに

2005 年度上期 OSS 活用基盤整備事業「OSS 性能・信頼性評価/障害解析ツールの開発」Java アプリケーション層において J2EE マイクロベンチマーク JBento の開発、Tomcat, JBoss の性能評価を実施した。報告書公開後、JBoss の開発者と共同で JBoss の再ベンチマークとチューニングを実施し、大幅な性能向上を確認した。本書ではその結果を報告する。

2005 年度上期 OSS 活用基盤整備事業「OSS 性能・信頼性評価/障害解析ツールの開発」Java アプリケーション層の報告書(以下、2005 年度上期報告書)は下記のサイトからダウンロードできる。

http://www.ipa.go.jp/software/open/forum/development/index.html

また、再ベンチマークの結果は、OSCJ.net の JBento Project サイトでも公開中である。

http://jbento.oscj.net/

1.1 経緯

JBoss 再ベンチマークを実施することになった経緯を下記に示す。

(1) 2005 年度上期 OSS 活用基盤整備事業にて J2EE マイクロベンチマーク JBetno を開発。JBento を利用し Tomcat と JBoss の性能測定を実施

(2) OSCJ.net の JBento Project のサイトにて結果を公開
→HttpSession レプリケーションが遅いことを明らかにし、JBoss Forum に結果を投稿

(3) JBoss Inc. から共同チューニングの提案

1.2 再ベンチマークとチューニング作業について

- 期間　2005 年 11/21(月)～11/25(金)
- 実施内容

JBoss クラスタリング機構のリード開発者である Ben Wang 氏と共同作業を実施。最新安定版リリースである JBoss 4.0.3SP1 をベースに以下のチューニングを実施し、大幅な性能の向上を確認した。

- HttpSession レプリケーション機構を実現するために使われているライブラリ JBossCache をバージョン 1.2.3 から 1.2.4 へアップデート
- レプリケーション時の内部通信に利用されているライブラリ JGroups をバージョン 2.2.7 から 2.2.9RC1 へアップデート
- tc5-cluster-service.xml のプロトコルスタック部分のパラメータを変更

2 測定結果概要

JBento の Counter 処理モデルを用い、下記(4), (5)のケースについて測定を実施した。なお、(1)〜(3)は 2005 年度上期の測定結果である。ハードウェア、ネットワーク構成、OS などの測定環境は 2005 年度上期の環境と同一である。処理モデル、2005 年度上期の測定結果、測定環境については 2005 年度上期報告書を参照のこと。

(1) Tomcat 5.5.9 クラスタ HttpSession レプリケーションあり
(2) JBoss 4.0.2 クラスタ HttpSession レプリケーションあり(同期レプリケーション)
(3) JBoss 4.0.2 クラスタ HttpSession レプリケーションあり(非同期レプリケーション)
(4) JBoss 4.0.3SP1, JBossCache 1.2.4, JGroups 2.2.9RC1 クラスタ HttpSession レプリケーションあり(同期レプリケーション)
(5) JBoss 4.0.3SP1, JBossCache 1.2.4, JGroups 2.2.9RC1 クラスタ HttpSession レプリケーションあり(非同期レプリケーション)

以降、JBoss 4.0.3SP1, JBossCache 1.2.4, JGroups 2.2.9RC1 クラスタを JBoss 4.0.3SP1+α クラスタと記述する。

(1)〜(5)の測定結果について概要を図 2-1 に示す。それぞれ AP サーバ 4 台を利用した場合の結果である。横軸がクライアント数、縦軸がスループットである。

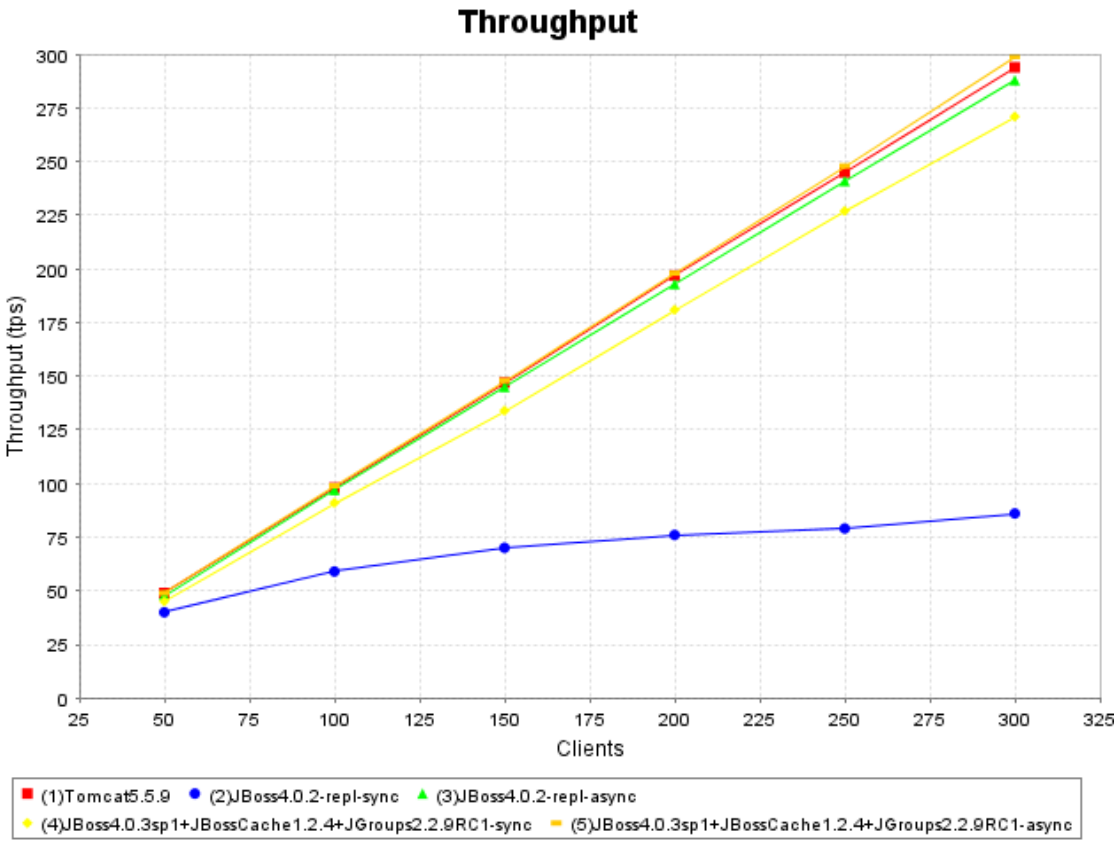

図 2-1 スループット

JBoss クラスタの性能が同期、非同期ともに大幅に向上した。300 クライアント時、同期では 86tps が 271tps に向上している(凡例(2), (4) 比較)。非同期では 288tps が 299tps に性能向上し(凡例(3), (5)比較)、またこのとき CPU 使用率が大幅に減少していることを確認した。詳細は 3.3.3 項を参照のこと。

また JBoss4 台時の同期レプリケーション測定における処理時間の内訳を図 2-2, 図 2-3 に示す。横軸がクライアント数、縦軸がそれぞれの処理に要した時間である。凡例中の x1 はリクエストが Client Emulator からサーバの最初の記録ポイントに到着するまでの時間、x2 はサーバ内での処理時間、x3 はレスポンスがサーバの最後の記録ポイントから Client Emulator に到着するまでの時間である。

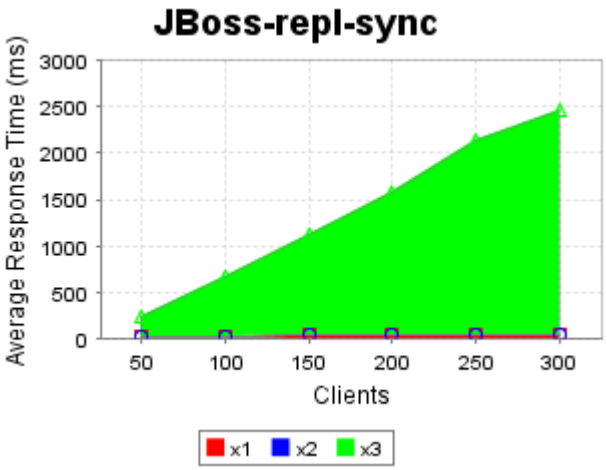

図 2-2 処理時間 (JBoss4.0.2 repl-sync)

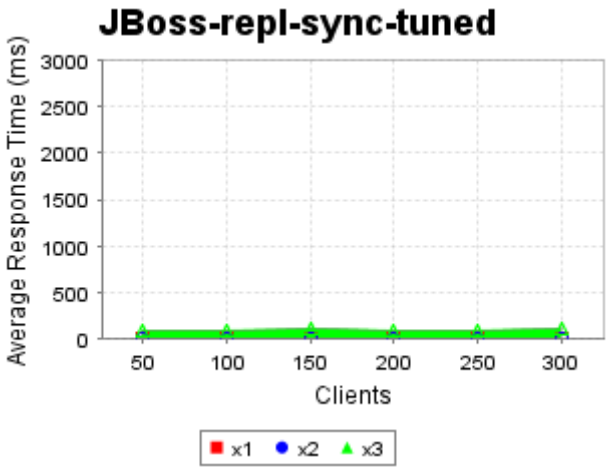

図 2-3 処理時間 (JBoss4.0.3SP1+α repl-sync)

図 2-2, 図 2-3 を比較すると、内部処理時間が大幅に減少していることが分かる。JBoss の HttpSession レプリケーションの処理は x3 の中で処理されるが、その処理時間が大幅に減少している。JBoss4.0.3SP1+α の方は、300 クライアントまでは一定の処理時間を保っている。

3 測定結果詳細

2 章で述べた測定結果の詳細を本章に記す。JBoss クラスタの HttpSession レプリケーション機能を再評価するために、下記(3),(4)のケースについて評価を行った。なお、(1), (2)は 2005 年度上期で評価を実施したものである。本章では、ケース(1)と(3), ケース(2)と(4)の比較を行う。

サーバの処理モデルには Counter を利用し、クライアント数を 50 から 300、AP Server 台数を 1 台から 4 台に変化させながら処理性能、応答時間、サーバリソースを測定することでスケーラビリティを評価した。

(1) JBoss 4.0.2 クラスタ HttpSession レプリケーションあり(同期レプリケーション)
(2) JBoss 4.0.2 クラスタ HttpSession レプリケーションあり(非同期レプリケーション)
(3) JBoss 4.0.3SP1, JBossCache 1.2.4, JGroups 2.2.9RC1 クラスタ HttpSession レプリケーションあり(同期レプリケーション)
(4) JBoss 4.0.3SP1, JBossCache 1.2.4, JGroups 2.2.9RC1 クラスタ HttpSession レプリケーションあり(非同期レプリケーション)

以降、JBoss 4.0.3SP1, JBossCache 1.2.4, JGroups 2.2.9RC1 クラスタを JBoss 4.0.3SP1+α クラスタと記述する。

台数を変化させた場合の使用サーバは以下の通り。

- AP Server1 台 - kuma06
- AP Server2 台 - kuma06, kuma05
- AP Server3 台 - kuma06, kuma05, kuma04
- AP Server4 台 - kuma06, kuma05, kuma04, kuma03

クライアントの JMeter 設定ファイルは JBento の HttpSession モジュールに含まれるデフォルトのものをそのまま利用している。デフォルトの設定ファイルではサーバへのリクエスト送信、レスポンス受信後に、1 秒スリープしてから再度リクエストの送信を行う設定となっている。そのため、スループットの理想値はクライアント数とほぼ同一となる。

3.1 JBoss 4.0.3SP1 の設定チューニング

3.1.1 HttpSession のレプリケーション粒度の変更

JBoss 4.0.2 に対する測定では HttpSession レプリケーションの粒度に ATTRIBUTE を設定していたが、今回の JBoss 4.0.3SP1 に対する測定では、SESSION を設定している。変更理由については 4.1 節を参照のこと。

3.1.2 JGroups の設定変更

JBoss 4.0.3SP1 に対する測定では、tc5-cluster-service.xml 内に記述されている JGroups の設定を変更している。JBoss 4.0.2 に対する測定ではこの設定はデフォルトのまま利用している。

変更内容は、tc5-cluster-service.xml の<attribute name="ClusterConfig">要素内の設定を、JGroups 2.2.9RC1 に含まれている設定サンプルの一つである fc-fast-minimalthreads.xml の内容に置き換えている。以下に完全な<attribute name="ClusterConfig">要素の設定を示す。

```
<attribute name="ClusterConfig">
  <config>
    <UDP mcast_addr="${jboss.partition.udpGroup:230.1.2.7}"
         mcast_port="45577"
         tos="16"
         ucast_recv_buf_size="20000000"
         ucast_send_buf_size="640000"
         mcast_recv_buf_size="25000000"
         mcast_send_buf_size="640000"
         loopback="false"
         discard_incompatible_packets="true"
         max_bundle_size="64000"
         max_bundle_timeout="30"
         use_incoming_packet_handler="true"
         use_outgoing_packet_handler="false"
         ip_ttl="2"
         down_thread="false" up_thread="false"
         enable_bundling="true"/>
    <PING timeout="2000"
          down_thread="false" up_thread="false" num_initial_members="3"/>
    <FD_SOCK down_thread="false" up_thread="false"/>
    <pbcast.NAKACK max_xmit_size="60000"
                   use_mcast_xmit="false" gc_lag="0"
                   retransmit_timeout="100,200,300,600,1200,2400,4800"
                   down_thread="false" up_thread="false"
                   discard_delivered_msgs="true"/>
    <UNICAST timeout="300,600,1200,2400,3600"
             down_thread="false" up_thread="false"/>
    <pbcast.STABLE stability_delay="1000" desired_avg_gossip="50000"
                   down_thread="false" up_thread="false"
```

```
                  max_bytes="400000"/>
    <pbcast.GMS print_local_addr="true" join_timeout="3000"
                down_thread="false" up_thread="false"
                join_retry_timeout="2000" shun="true"/>
    <FC max_credits="2000000" down_thread="false" up_thread="false"
        min_threshold="0.10"/>
  </config>
</attribute>
```

図 3-1 変更後の tc5-cluster-service.xml の<attribute name="ClusterConfig">要素

3.2 JBoss4.0.2 クラスタ / JBoss4.0.3SP1+ α クラスタ比較

HttpSession レプリケーションあり，同期レプリケーション(repl-sync)

本項では、JBoss4.0.2, JBoss4.0.3SP1+αの各クラスタ構成において、HttpSession レプリケーションあり、HttpSession レプリケーションを同期(CacheMode=REPL_SYNC)に設定した場合の評価結果を比較する。

3.2.1 クライアント数とスループット，レスポンスタイムの関係に関する測定結果

クライアント数とスループット，レスポンスタイムの関係に関する測定結果をそれぞれ図 3-2～図 3-5 に示す。横軸がクライアント数、縦軸がそれぞれスループット，レスポンスタイムである。

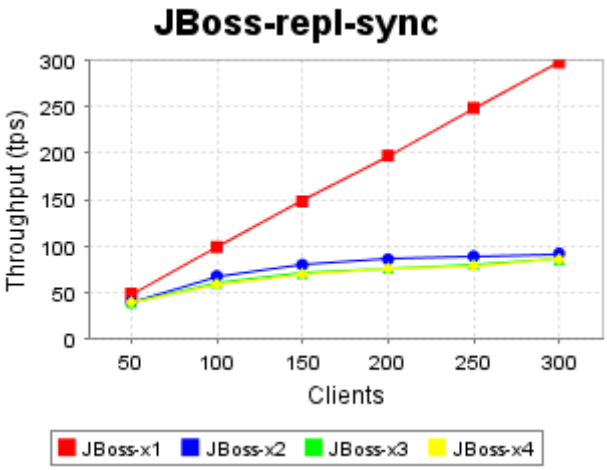

図 3-2 スループット(JBoss4.0.2 repl-sync)

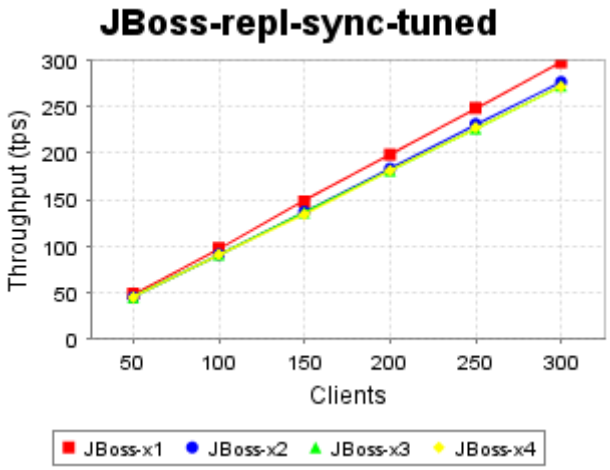

図 3-3 スループット(JBoss4.0.3SP1+α repl-sync)

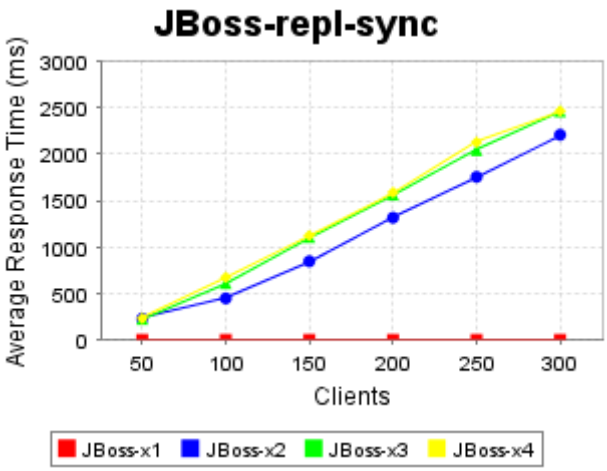

図 3-4 レスポンスタイム(JBoss4.0.2 repl-sync)

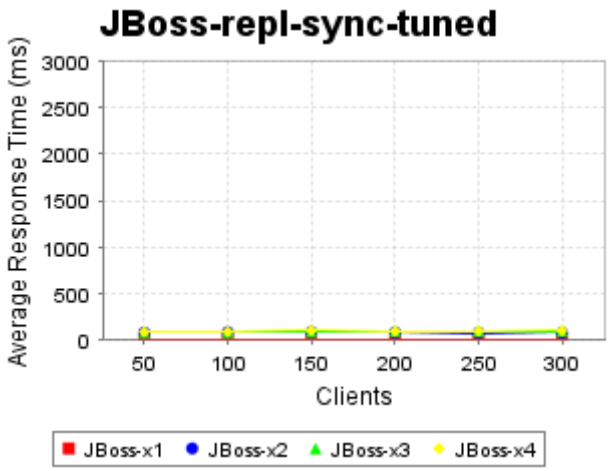

図 3-5 レスポンスタイム(JBoss4.0.3SP1+α repl-sync)

図 3-2～図 3-5 のグラフから分かる事は以下の通り。

- JBoss4.0.2 クラスタに比べ、JBoss4.0.3SP1+α クラスタでは大幅に性能が向上している。AP Server4 台、300 クライアント時でスループットは 86tps から 271tps に向上し、レスポンスタイムは 2459ms から 104ms と激減している。
- JBoss4.0.3SP1+α クラスタでは、AP Server1 台使用時よりも複数台使用時の場合では少々性能が劣化している。しかし、JBoss4.0.2 クラスタの複数台使用時の性能劣化に比べると劣化度合いは大幅に小さい。

3.2.2 クライアント数と処理時間の関係に関する測定結果

AP Server4 台の場合の、クライアント数と処理時間の関係に関する測定結果を図 3-6, 図 3-7 に示す。横軸がクライアント数、縦軸がそれぞれ処理時間である。なお、x1 はリ

クエストが Client Emulator からサーバの最初の記録ポイントに到着するまでの時間、x2 はサーバ内での処理時間、x3 はレスポンスがサーバの最後の記録ポイントから Client Emulator に到着するまでの時間である。

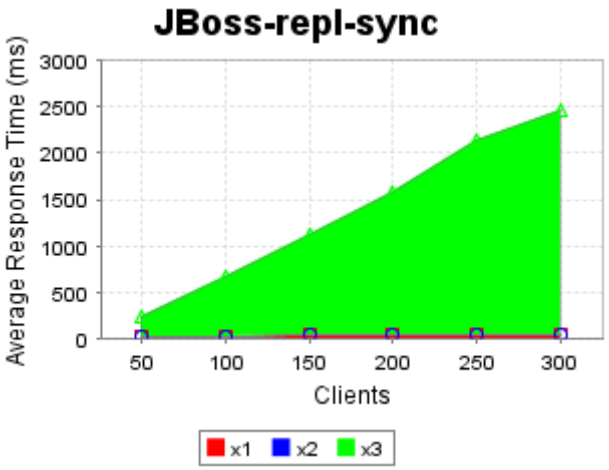

図 3-6 処理時間(JBoss4.0.2 repl-sync)

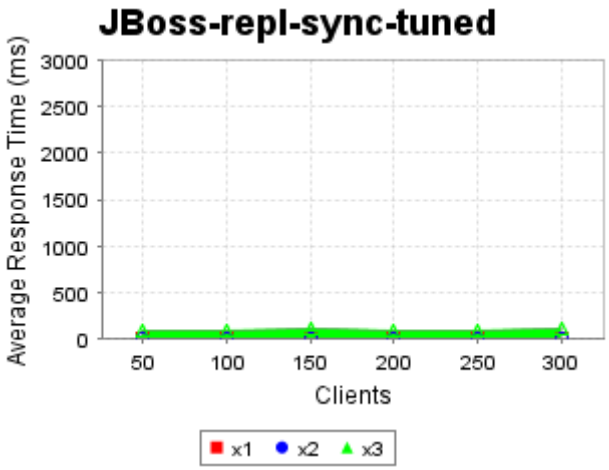

図 3-7 処理時間(JBoss4.0.3SP1+α repl-sync)

図 3-6, 図 3-7 のグラフから分かる事は以下の通り。

- 図 3-6, 図 3-7 を比較すると、内部処理時間が大幅に減少していることが分かる。JBoss の HttpSession レプリケーションの処理は x3 の中で処理されるが、その処理時間が大幅に減少している。JBoss4.0.3SP1+α の方は、300 クライアントまでは一定の処理時間を保っている。

3.2.3 サーバリソースに関する測定結果

サーバリソースとして CPU、メモリ、ディスク、ネットワークの負荷状況を測定した。それらのうち CPU に関する物だけを以下に示す。その他のサーバリソースについては特に目立った傾向は見られなかったため割愛する。

AP Server4 台の場合の各サーバの CPU 利用率の平均を図 3-8, 図 3-9 に示す。横軸がクライアント数、縦軸が CPU 利用率の平均値である。

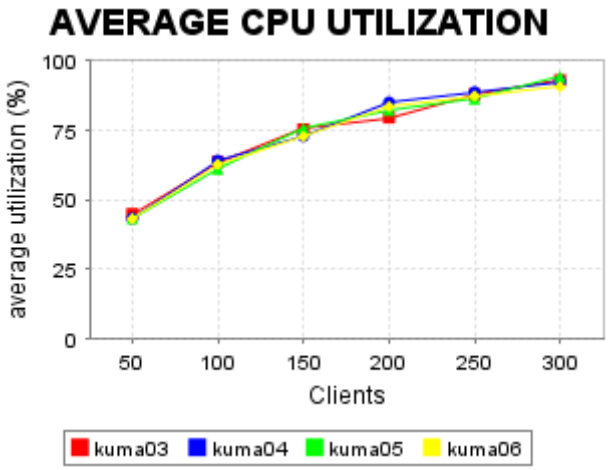

図 3-8 CPU 利用率 JBoss4.0.2 4 台

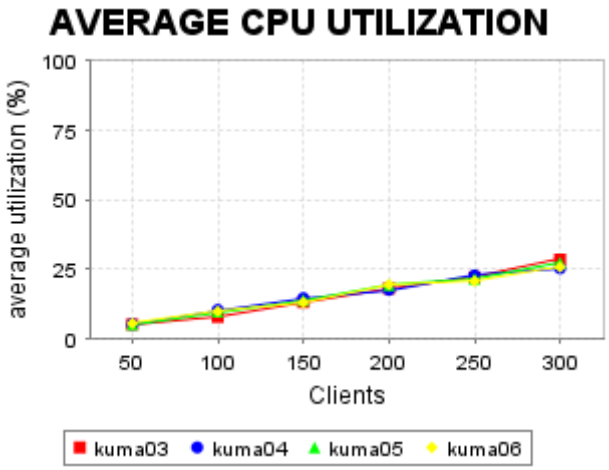

図 3-9 CPU 利用率 JBoss4.0.3SP1+α 4 台

図 3-8, 図 3-9 のグラフから分かる事は以下の通り。

- CPU 利用率が大幅に少なくなったことがわかる。JBoss4.0.3SP1+α クラスタでは、300 クライアント時に CPU 利用率は 27%程と余裕があるので、さらに負荷をかけてもスループットは上昇すると考えられる。

3.3 JBoss4.0.2 クラスタ / JBoss4.0.3SP1+ α クラスタ比較 HttpSession レプリケーションあり，非同期レプリケーション (repl-async)

本項では、JBoss4.0.2, JBoss4.0.3+α の各クラスタ構成において、HttpSession レプリケーションあり、HttpSession レプリケーションを非同期(CacheMode=REPL_ASYNC)に設定した場合の評価結果を比較する。

3.3.1 クライアント数とスループット，レスポンスタイムの関係に関する測定結果

クライアント数とスループット，レスポンスタイムの関係に関する測定結果をそれぞれ図 3-10～図 3-13 に示す。横軸がクライアント数、縦軸がそれぞれスループット，レスポンスタイムである。

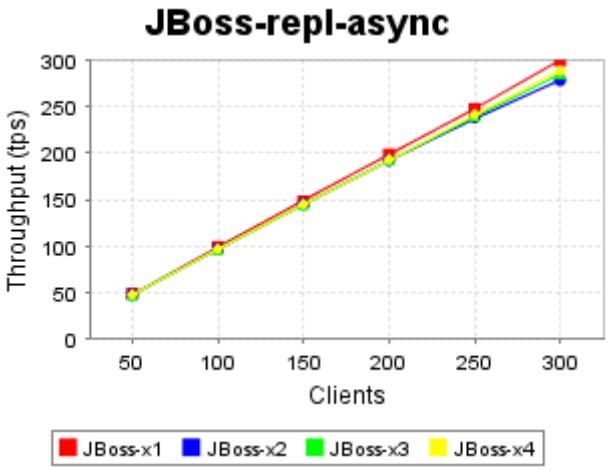

図 3-10 スループット(JBoss4.0.2 repl-async)

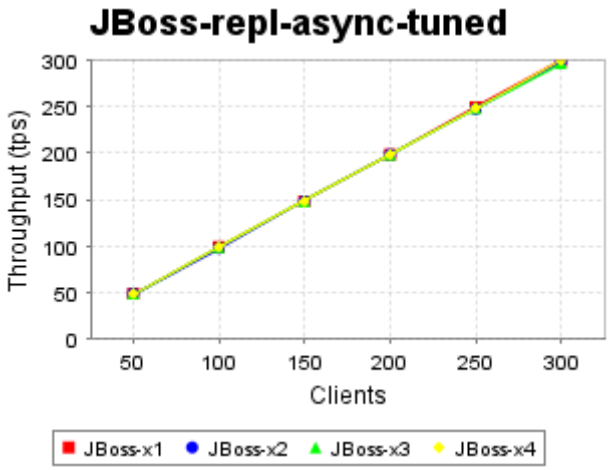

図 3-11 スループット(JBoss4.0.3SP1+α repl-async)

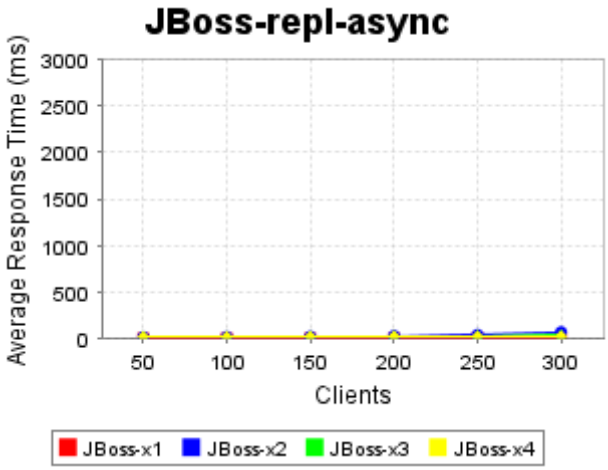

図 3-12 レスポンスタイム(JBoss4.0.2 repl-async)

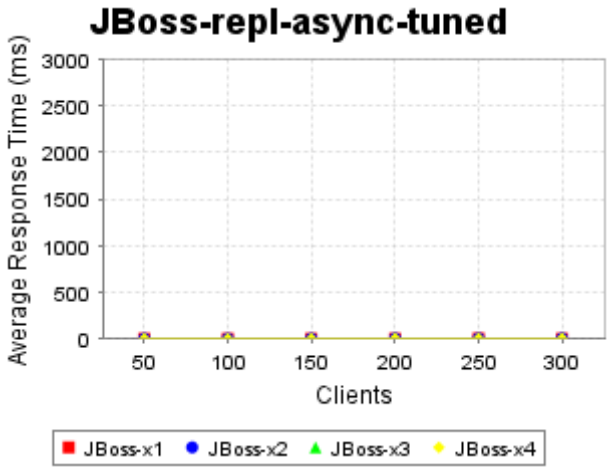

図 3-13 レスポンスタイム(JBoss4.0.3SP1+α repl-async)

図 3-10～図 3-13 のグラフから分かる事は以下の通り。

- JBoss4.0.2 クラスタでは、200 クライアント時から少々性能劣化が確認されたが、JBoss4.0.3SP1+α クラスタでは 300 クライアントまで性能劣化は起こっていない。AP Server4 台、300 クライアント時でスループットは 288tps から 299tps に向上し、レスポンスタイムは 35ms から 6ms と減少している。

3.3.2 クライアント数と処理時間の関係に関する測定結果

AP Server4 台の場合の、クライアント数と処理時間の関係に関する測定結果を図 3-14, 図 3-15 に示す。横軸がクライアント数、縦軸がそれぞれ処理時間である。なお、x1 はリクエストが Client Emulator からサーバの最初の記録ポイントに到着するまでの時間、

x2 はサーバ内での処理時間、x3 はレスポンスがサーバの最後の記録ポイントから Client Emulator に到着するまでの時間である。

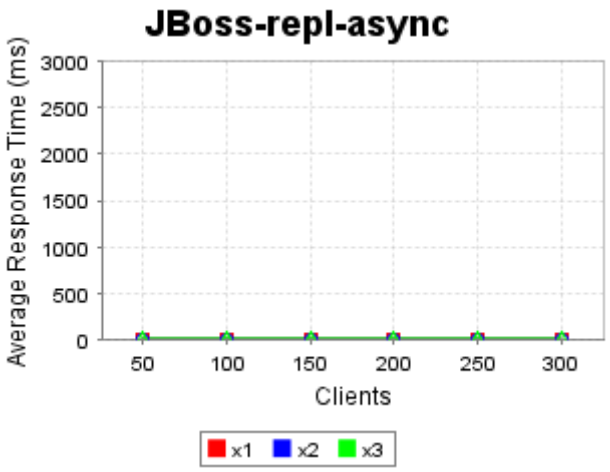

図 3-14 処理時間(JBoss4.0.2 repl-async)

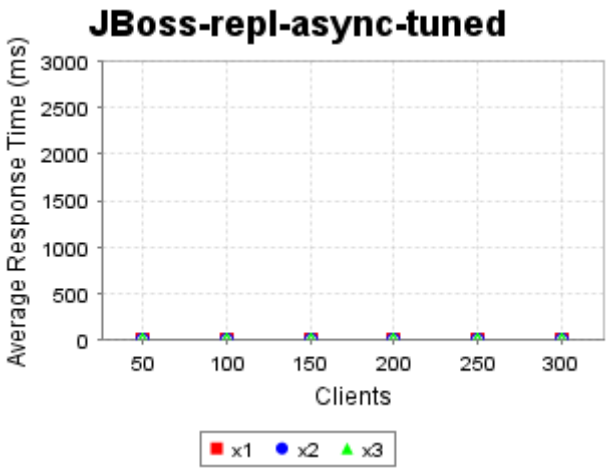

図 3-15 処理時間(JBoss4.0.3SP1+α repl-async)

図 3-14, 図 3-15 のグラフから分かる事は以下の通り。

- JBoss4.0.2 クラスタでもクライアント数 300 まで各処理時間はほとんどかかっていないが、JBoss4.0.3SP1+α クラスタではさらに処理時間が短くなっている。
- HttpSession レプリケーションを発行するタイミングは repl-sync と同様 x3 であるが、レプリケーション処理はバックグラウンドで非同期に行われるため、レスポンスタイムへの影響はほとんどなくなっている。

3.3.3 サーバリソースに関する測定結果

サーバリソースとして CPU、メモリ、ディスク、ネットワークの負荷状況を測定した。それらのうち CPU に関する物だけを以下に示す。その他のサーバリソースについては特に目立った傾向は見られなかったため割愛する。

AP Server4 台の場合の各サーバの CPU 利用率の平均を図 3-16, 図 3-17 に示す。横軸がクライアント数、縦軸が CPU 利用率の平均値である。

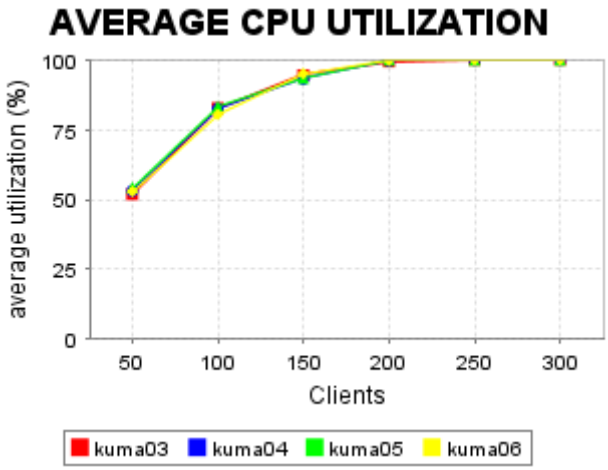

図 3-16 CPU 利用率 JBoss4.0.2　4 台

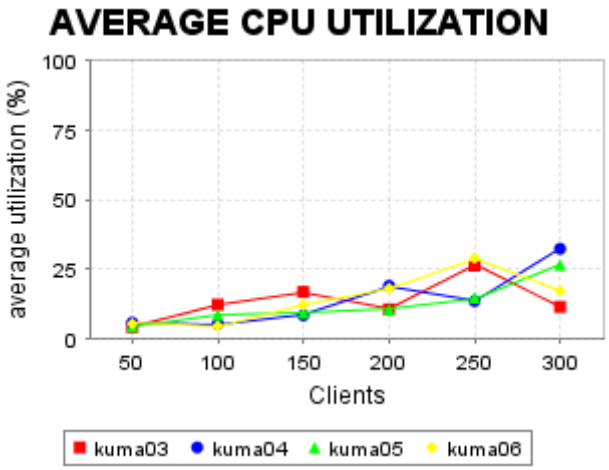

図 3-17 CPU 利用率 JBoss4.0.3SP1+α　4 台

図 3-16, 図 3-17 のグラフから分かる事は以下の通り。

- JBoss4.0.3SP1+α クラスタを利用することで、大幅に CPU 使用率が下がることが確認できた。JBoss4.0.2 クラスタの場合は、クライアント数が増えるにつれて CPU 利用率が高くなり、クライアント数 200 以降、CPU 利用率はほぼ 100%になっている。しかし、JBoss4.0.3SP1+α クラスタでは多くて 32%程度である。
- JBoss4.0.2 クラスタの場合は CPU を使いきっている 200 クライアント以上でもスループットが伸びていることから、レプリケーションが追いついていないと想定される。しかし JBoss4.0.3SP1+α クラスタでは、CPU 使用率からはそのような問題が無いことが伺える。

4 まとめ

4.1 考察

JBoss 4.0.2 の HttpSession レプリケーションの性能については、残念ながら期待する結果を大幅に下回るものであった。しかし、JBoss4.0.3SP1 をベースに JBossCache1.2.4, JGroups2.2.9RC1 を用いて再ベンチマークを実施した結果、大幅な性能向上が確認できた。

これは JBossCache のバージョン 1.2.3 から 1.2.4 の間に行われた同期処理の最適化の影響が大きい。バージョン 1.2.4 では、IdentityLock クラスと LockMap クラスにおける同期の削減、LockInterceptor クラスでの無駄な createLock()の削除などの変更が適用されている。また、JGroups の最新である JGroups 2.2.9RC1 を適用することで、CPU 使用率が大幅に低下することも確認した。詳細な変更点については、各プロダクトのリリースノートを参照されたい。

- JBossCache 1.2.4 リリースノート
  - http://jira.jboss.com/jira/secure/ConfigureReport.jspa?versions=12310457&versions=12310238&sections=.5.2.1.3.8.4.7.6&style=html&reportKey=pl.net.mamut%3Areleasenotes&Next=Next
- JGroups 2.2.9RC1 リリースノート
  - http://sourceforge.net/project/shownotes.php?release_id=370799&group_id=6081

最適化が行われた JBossCache 1.2.4 は、さらにいくつかのバグフィックスが取り込まれた後、1.2.4SP1 としてリリースされ、JBoss 4.0.3SP2 に同梱される予定である。JBoss クラスタ環境を利用する場合は、パフォーマンスや安定性の観点から、この JBoss 4.0.3SP2 以降のバージョンを利用することが望ましい。JGroups についてはバージョン 2.2.7 が安定版であるため、最新であるバージョン 2.2.9 は JBoss 4.0.3SP2 には含まれない予定である。もし、JBoss ノードの CPU がボトルネックになる可能性が高い場合は、JGroups を最新版に置き換えると良いだろう。

また、今回の測定では JBoss の HttpSession レプリケーションの粒度については SESSION を利用した。これは、JBoss Inc.の開発者の「ATTRIBUTE はまだあまり調整されていないため SESSION を使うべきだ」という判断に従ったためである。今回の処理モデルは Counter を用いたため、理論上 SESSION, ATTRIBUTE で HttpSession のレプリケーションに違いは生じない。実際に ATTRIBUTE に設定し測定を行っても性能劣化は生じなかった。今後、HttpSession レプリケーションの粒度による性能の違いが出る処理モデルを用いて、検証を実施する必要がある。

JBoss で実装予定とされている機能のうち、HttpSession レプリケーションに関係するものが 3 つある。TreeCacheAop と Buddy replication、そして Optimistic locking と呼ばれるものだ。既存の機能についても、継続的にチューニングが施される予定であり、これらの実装後にはさらに性能が向上することが期待できる。

4.2 コミュニティへの貢献

(1) J2EE マイクロベンチマークの開発

2005 年度上期の測定結果を web に公開したことで JBoss Inc.と共同でベンチマークとチューニングを実施する運びとなった。JBoss のクラスタ機能の開発者である Ben Wang 氏と一週間の作業の結果、2 章, 3 章に示すような性能改善が確認できた。

今後も JBoss に限らず、OSS コミュニティへの情報発信を継続的に行い、JBento の認知度を上げてより多くの OSS コミュニティのための性能改善の為のツールとして役立ててもらえるようにしたい。

(2) クラスタシステムの信頼性評価

2005 年度上期報告書「8. クラスタシステムの信頼性評価」の JBoss 4.0.2 を用いた実機評価の実施過程で、JBoss クラスタの運用に支障をきたす 2 件の問題点を発見した。本報告書執筆時点での進展を下記に示す。

1 件目の問題は、フェイルオーバ時に経路再設定が行われない問題である。この問題は特に 3 ノード以上のクラスタ構成で顕著に現れ、障害が発生した場合にフェイルオーバ先のノードへの経路の切り替えが正しく行われず、不安定な状態でクラスタの運用が継続されてしまう。Web で情報を調査したところ、発見した前日に JBoss のバグ管理システムに報告されていた。ワーキンググループから JBoss 開発者に対して障害報告・テスト・パッチ提供を行ったところ、提供したパッチが JBoss のソースツリーにマージされ、JBoss 4.0.2 用の修正モジュールも提供された。JBoss 4.0.2 でクラスタを構成する場合は、下記の URL で提供されるパッチモジュールを適用することにより、この問題は解消される(2005 年度上期報告書執筆時)。なお、修正内容は JBoss4.0.3 Final から適用されている。

http://jira.jboss.com/jira/browse/JBAS-2034

2 件目の問題は、JBoss クラスタの運用中に 1 ノードが障害停止するとクラスタ全体がスローダウンし処理を続行できなくなるという問題である。 JBoss クラスタはレプリケーションの同期方式(CacheMode)に同期(REPL_SYNC)と非同期(REPL_ASYNC)の 2 方式を提供しているが、この現象は同期(REPL_SYNC)のときに顕著に現れる。非同期(REPL_ASYNC)では、同期(REPL_SYNC)ほど顕著なスローダウンは発生しないが JBoss のログには同期(REPL_SYNC)と同様のエラーが出力されており、方式の変更は根本的な問題解決にはならない。この問題についても共同ベンチマーク中に再現試験を行い、バグの詳細を調査、解決に至る糸口をつかんだ。なお、修正内容は JBoss 4.0.3SP2 で適用される予定である。

http://jira.jboss.com/jira/browse/JBAS-2170

4.3 今後の予定

JBoss クラスタの検証については、HttpSession に積み込むオブジェクトのサイズを変更する、より高負荷をかけるなどの検証を実施し、情報発信を継続的に行っていく予定である。また、JBoss 以外の AP サーバの検証も実施し、OSS AP サーバの適用範囲を見極めると共に、コミュニティと連携しつつ適用範囲の拡大を図っていきたい。