DRBDと仮想化技術を利用した耐障害性と汎用性の高いサーバファームの構築

二川 潤† 下農 淳司†† 雪田 修一†

†法政大学大学院情報科学研究科
††京都大学理学研究科宇宙物理学教室
E-mail: †{jun@futagawa.info, yukita@hosei.ac.jp} ††shimono@kusastro.kyoto-u.ac.jp

あらまし 組織の大小に関係なく，今日では基幹業務は多様なネットワークサービスへの依存度を高めており，あらゆるシステムにおいてその規模に関係なく高い可用性が求められるようになってきた．また，インターネットサービスにおいても，不特定多数の利用者が24時間利用する可能性があるため高い可用性が求められている．本稿では，ネットワーク越しに異なるマシンが持つブロックデバイス間をミラーリングするDRBDと仮想化技術を利用した，低コストで耐障害性と汎用性の高いサーバファームを提案する．また，実際のサーバファーム構築例として，日本におけるMozillaコミュニティであるもじら組での事例を紹介する．
キーワード サーバファーム，DRBD，仮想計算機，耐障害性，高可用性，汎用性，低コスト

The implementation of a fault tolerant server farm using DRBD and virtual machine technology

Jun FUTAGAWA†, Atsushi SHIMONO††, and Shuichi YUKITA†

†Graduate School of Computer and Information Sciences, Hosei University
††Department of Astronomy, Kyoto University
E-mail: †{jun@futagawa.info, yukita@hosei.ac.jp} ††shimono@kusastro.kyoto-u.ac.jp

Abstract Mission-critical tasks in the Internet age today depend strongly on various network services regardless of the size of the organization. Consequently, high availability has come to be requested in all systems regardless of their scales. Moreover, high availability is mandatory in internet services because customers on the whole globe use the service 24 hours. In this paper, we propose an implementation of a fault tolerant server farm using a virtual machine technology and DRBD that mirrors between different block devices over the network. To prove the effectiveness of our approach, we present a case study of the server farm of MozillaGumi, a major Mozilla community in Japan.
Key words server farm, DRBD, virtual machine, fault tolerant, high availability, generality, low cost

1. はじめに

組織の大小に関係なく，今日では基幹業務は多様なネットワークサービスへの依存度を高めており，あらゆるシステムにおいてその規模に関係なく高い可用性が求められるようになってきた．また，インターネットサービスにおいても，不特定多数の利用者が24時間利用する可能性があるため高い可用性が求められている．ハードウェア面におけるシステムの多重化を考えると，ハードディスクや電源ユニット，ネットワークカードなどの多重化構成は比較的低コストに故障から保護する手段として一般的に利用されているが，CPUやマザーボード自体など多重化し難い箇所は，高価なシステムを除き多重化されることは少ない．そのため，そのような箇所が故障した場合にはシステムを停止して故障個所を修理するか，別のマシンにハードディスクを載せ替えたり，バックアップデータから復旧するまではシステムの停止時間は続いてしまう．また，多重化されたハードウェアであっても，その一部が故障した時点ではシステムが停止することはなくとも，故障個所を修理する際には，多くの場合，システムを一時停止させる必要が生じてしまう．これらの問題に対処するために，複数台のマシンで動作する個々のアプリケーションレベルで冗長構成を組む手段が考えられるが，すべてのアプリケーションにおいて冗長構成を組めるとは限らない．また，一時的に低下した冗長性を再び確保するために，故障したマシンをいち早く容易に復旧させることは常に必要である．

一方，潤沢な計算機資源を有効に使用するため，あるいは，運用する計算機全体での消費電力を低減させるために，仮想化技術を用いて1つの物理マシンで複数の仮想マシンを運用するサーバファームを構築する例が増えてきている．仮想マシンを利用す

ることで物理マシンのハードウェアに依存することがなくなることから，異なる物理マシンへ仮想マシンのディスクイメージをコピーしてもすぐにその仮想マシンを動作させることができるが，仮想マシンに割り当てるディスクサイズを大きくすると，障害時に仮想マシンのディスクイメージを別の物理マシンにコピーする時間が掛かるようになり，システムの停止時間が長引く原因となる．そこで，高価なシステムであれば，仮想マシンのディスクイメージは，Storage Area Network（SAN）に配置し共有することですぐに別の物理マシンで仮想マシンを起動できる仕組みを構築することができるが，SAN 構成は高価であることからすべてのシステムに導入することは困難である．また，SAN 本体，FC インタフェースカード，FC スイッチなど汎用性の高くない機材を使用するため，故障した際にすぐに代替機材を用意することが困難である．そのため，なるべく低コストであり，かつ，汎用的に利用できるサーバファームの仕組みが必要とされている．

そこで本稿では，ネットワーク越しに異なるマシンが持つブロックデバイス間をミラーリングすることができる Distributed Replicated Block Device（DRBD）[1]と仮想化ソフトウェア Xen[2]を組み合わせ，ハードウェア故障時であっても短時間に復旧可能な耐障害性と，汎用的なハードウェアのみで低コストに構築可能である汎用性を兼ね備えたサーバファームを提案する．また，実際のサーバファーム構築例として，日本における Mozilla[3]コミュニティであるもじら組[4]での事例を紹介する．

2. DRBD

本節では，本システムで使用するディスクミラーリングソフトウェア DRBD について述べる．

2.1 概要

DRBD は，LINBIT 社が開発した Linux 上で動作する TCP/IP 経由でネットワーク上の異なるマシンが持つブロックデバイス間のデータを同期するディスクミラーリングソフトウェアである．DRBD は，ファイルシステムより低いレイヤであるブロックデバイスとして動作し，システムからは専用の DRBD リソースとも呼ばれる DRBD ブロックデバイスを経由してアクセスする．このブロックデバイスは /dev/drbdN（N は設定に対応した任意の数字）というデバイスファイル名になる．ファイルシステムから渡されたブロックデータをそのままリアルタイムに異なるマシンの下位ブロックデバイスへミラーリングするため，DRBD リソースには様々なファイルシステムを作成することができる．DRBD にはオープンソース版 DRBD と商用版の DRBD Plus が存在するが，本稿ではミラーリング可能台数が 2 台，ブロックデバイスごとの最大作成可能サイズが 4TB のオープンソース版 DRBD を利用する．なお，商用版の DRBD Plus を利用することで，ミラーリング可能台数が 3 台，ブロックデバイスごとの最大作成可能サイズが 16TB になり，さらなる耐障害性を得ることもできる．

新しい DRBD リソースを作成する際には，ミラー対象となるディスクに作成したパーティションや LVM の論理ボリュームに対応する下位ブロックデバイスに DRBD メタデータを作成し，すべてのブロックデバイスのデータをフル同期する必要がある．この作業はミラーし合う下位ブロックデバイスのサイズが大きい程時間が掛かるが，関連する初期 DRBD メタデータを予め書き込んでおくことで初回のみフル同期を省略することができる[5]．

DRBD リソースが動作している時，DRBD ブロックデバイスには Primary と Secondary の状態があり，Primary 状態にあるマシン側のみデータを読み書きすることができ，Secondary 状態にあるマシン側はデータを読み書きすることはできない．また，障害などにより，Secondary 状態にあるマシンが停止しても，Primary 状態にあるマシンでは DRBD ブロックデバイスを読み書きし続けることができる．このことから，Primary 状態にあるマシンを Active，Secondary 状態にあるマシンを Standby ということができる．DRBD メタデータが破損せずに DRBD ブロックデバイスが復旧した場合，DRBD は自動的に停止中に動作していた Primary 側で書き換えられた部分だけを同期することで短時間に再同期する．ハードディスクを新規に交換するなどし，DRBD メタデータがない場合は，新規に DRBD メタデータ作成後，Primary 状態の DRBD リソースとフル同期する必要があるが，同期中も Primary 状態にある DRBD ブロックデバイスは読み書きできるため，Primary 側マシンのサービスを動作させながら障害復旧することができる．また，障害時の一連の動作を自動化するために，別途 Heartbeat[6]や Keepalived[7]などの HA クラスタソフトウェアと組み合わせて運用することで，Primary 側マシンに障害が起こった場合に自動的に Primary 側を切り離し，Secondary 側の状態を Primary 状態にフェイルオーバさせることもできる．その他，DRBD 単体ではファイルのロック管理機構を提供していないが，DRBD リソースを Primary/Primary 構成にし，Global File System（GFS）[8]や Oracle Cluster File System 2（OCFS2）[9]といったクラスタファイルシステムを使用することで，複数台から同時にデータを読み書きすることができる共有ディスクとして DRBD リソースを使用することもできる．

表 1　評価に用いた計算機と環境

計算機	環境
HP ProLiant ML110 G5 x2 台	Core 2 Quad Q9550 2.83GHz, 8GB memory (Dom0 1GB 割当), SATA 1TB x2 (WD1001FALS,no raid) 1000Base-T (Broadcom Tigon3), CentOS 5.2 (2.6.18-92.1.10.el5xen), DRBD 8.2.6 x86_64

表 2　Bonnie++によるベンチマーク

比較対象	連続読み込み (ブロック単位)	連続書き込み (ブロック単位)
Local Disk	107905 KB/sec	104850 KB/sec
DRBD x1	105028 KB/sec	71063 KB/sec
DRBD x2	104377 KB/sec	64833 KB/sec

2.2　DRBD の性能

DRBD の性能はネットワーク速度とディスク読み書き性能に強く依存する．DRBD は TCP/IP ネットワーク経由でデータをミラーリングするため，特にネットワーク速度は重要であるといえる．100Mbps のネットワークであれば，理論値でも 12.5MB/sec のデータ読み書き速度しか得られず大きなボトルネックとなるが，現在標準的に利用できるようになった 1000Mbps のネットワークであれば，理論値で 125MB/sec の速度を得ることができ，高いディスク I/O 性能を求められない場面では十分に実用可能である．表 1 に示す評価環境において，ハードディスクベンチマークツール Bonnie++[10] を用いたローカルハードディスクと Primary/Secondary 構成の DRBD リソースの性能を Primary 側マシンで測定した（表 2）．Bonnie++は初期設定のまま実行し，割り当てメモリの2倍になる2GB のファイルを読み書きに使用した．また，DRBD リソースの同期プロトコルには，最も高い信頼性を得られる 2 台のマシンのディスクに書き込んだ時点で書き込み完了とする方式 C を使用し，ファイルシステムはすべて ext3 を使用した．

表 2 の DRBD x1 は 1 つの DRBD リソースに対して実行した場合，DRBD x2 は 2 つの DRBD リソースを用意し，2 台のマシンでそれぞれ異なる 1 つの DRBD リソースに対して同時に実行した場合の計測結果である．結果は連続読み込み(ブロック単位) であればローカルハードディスクと遜色のない速度が得られた．これは，読み込み処理は常に自分自身が持つ Primary 状態にある DRBD ブロックデバイスの下位ブロックデバイスにのみアクセスするためである．連続書き込み(ブロック単位) ではローカルハードディスクに比べて DRBD x2 であっても 6 割程度の性能が得られている．また，今回使用したハードディスクの場合，ローカルハードディスクの性能が 1000Mbps のネットワークでの理論値 125MB/sec を下回っていることから，DRBD によるオーバヘッドがまったく無い状態においてもネットワークがボトルネックになることはない．しかし，実際の運用時には様々なサービスがネットワークを利用するため，ネットワーク帯域が十分ではない場合は，ネットワークインタフェースを複数用意し，DRBD の同期に利用するネットワークインタフェース同士を直結するなどしてトラフィックを分散する必要がある．

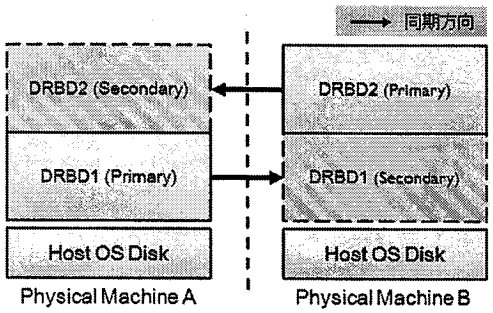

図 1　DRBD の基本構成

3.　サーバファームの基本構成

先述した DRBD と仮想化ソフトウェア Xen を組み合わせ，ハードウェア障害時であってもすぐに復旧可能な耐障害性に考慮したサーバファームを設計する．本節では最小構成である物理マシン 2 台による構成で述べる．なお，仮想化ソフトウェアは，Xen 以外の仮想化ソフトウェアでも構わない．

3.1　概要

DRBD によりミラーリングしたブロックデバイス上に仮想マシンのディスクイメージを配置することで，仮想マシンのディスクイメージがリアルタイムに物理的に異なるマシンにバックアップされ続けることになる．これにより，物理マシン A に障害が発生した場合，もう片方の物理マシン B に同期されている DRBD ブロックデバイスを Primary にし，仮想マシンを起動することで直前まで動作していた状態でシステムを復旧することができるサーバファームを実現できる．

DRBD 以外のネットワーク越しのファイル同期の仕組みとして Lsyncd[11]が挙げられる．Lsyncd はファイルシステムに関係なく，特定のファイルをリアルタイムに同期することができるが，バイナリファイルを同期する場合，変更の度にすべてのデータを転送し直すため，仮想マシンのディスクイメージファイルなどの大きなファイルの共有には適していない．また，DRBD を用いた場合では，ファイルシステムを作成せずに，直接 DRBD ブロックデバイスを指定して仮想マシンを作成することも可能である．

3.2　DRBD リソースの構成

本サーバファームにおける DRBD の構成は図 1

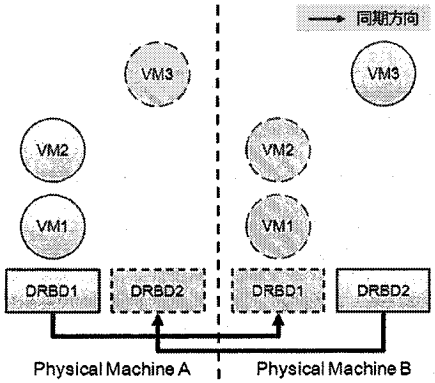

図 2　通常運用時の仮想マシンの動作構成

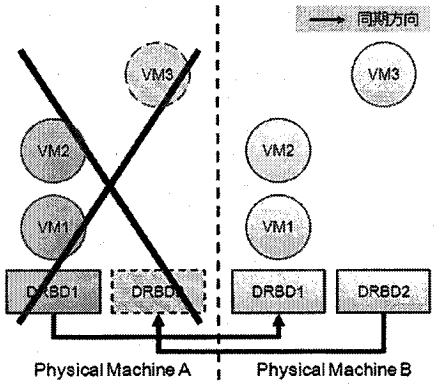

図 3　障害時の仮想マシンの動作構成

に示すように，物理マシン 2 台に対して 2 つ以上の DRBD リソースを用意するのが望ましい．DRBD リソースの同期処理は CPU 資源をあまり使用しないため，DRBD リソースが 1 つの場合，DRBD ブロックデバイスが Secondary 状態にある物理マシンの CPU 資源が無駄になってしまう．そこで 2 つ以上の DRBD リソースを 2 台の物理マシンで分散して Primary 状態にし，それぞれの物理マシンで仮想マシンを起動することでサーバファーム全体の CPU 資源を無駄なく有効に使用できる．

DRBD リソース上での仮想マシンのディスクイメージは，ファイルシステムを作成した上でファイルとして管理する方法と，DRBD ブロックデバイス上に直接作成して管理する方法がある．前者のファイルとして管理した場合，物理マシン上のファイルシステムを経由するオーバヘッドが存在し，後者の DRBD ブロックデバイス上に直接作成して管理した場合より仮想マシン上でのディスク I/O 性能は不利ではあるが，仮想マシンのディスクイメージを別のサーバファームへ移動するのが容易である．一方，後者の場合，一度ディスクイメージをダンプする必要があり，管理上の手間が発生する．また，1 つの DRBD リソースに複数の仮想マシンのディスクイメージを管理するには，DRBD ブロックデバイス上に更に LVM を作成し論理ボリュームのブロックデバイスを用意する必要がある．実際のサーバファーム構築時には，これらの必要とする性能と管理上の手間を考慮し，仮想マシンのディスクイメージの管理方法を選択する必要がある．

3.3　通常運用時と障害時の動作

図 2, 図 3 はサーバファームの通常運用時と障害時の仮想マシンの動作構成である．物理マシン 2 台でそれぞれ Primary 状態の DRBD ブロックデバイス上にある仮想マシンのディスクイメージから仮想マシンを起動する．動作している仮想マシンに書き込まれたデータは，DRBD ブロックデバイスを経由してもう片方の物理マシンの対応するディスクに書き込まれる．それぞれの物理マシンで仮想マシンを動作させることにより，CPU 資源とメモリ資源を無駄無く利用している状態である．本サーバファームを利用した際の障害時の対応を以下に示す．

1. 物理マシン A がダウン
2. 物理マシン B の DRBD1 ブロックデバイスの状態を Primary に変更し，必要に応じてファイルシステムのチェックを実行し，マウント
3. 物理マシン B で仮想マシン 1，2 を起動(図 3)

この際，DRBD ブロックデバイスのマウントポイントを揃え，DRBD リソース上に作成したファイルシステム上に仮想マシンの設定ファイルを配置しておくことで，異なる物理マシンでもすぐに仮想マシンを起動することができるようになる．また，この一連の障害時の対応は，Heartbeat や Keepalived を利用することで自動化することも可能である．

同様に，障害復旧後の対応を以下に示す．

1. 物理マシン A の障害復旧
2. DRBD リソースの再同期開始，完了
3. 物理マシン B で動作中の仮想マシン 1，2 を停止
4. 物理マシン B の DRBD1 ブロックデバイスの状態を Secondary に変更し，アンマウント
5. 物理マシン A の DRBD1 ブロックデバイスの状態を Primary に変更し，マウント
6. 物理マシン A で仮想マシン 1，2 を起動(図 2)

本サーバファーム構成であれば，SAN や FC インタフェースカードなど特別なハードウェアに依存していないため，安価なハードウェアで代替機材を用意することが可能である．また，システムの性能を向上したい場合も，より高性能な部品群で構成された物理マシン上へ仮想マシンのディスクイメージをコピーすることで，システムを再構築することなく容易に性能向上を図ることができる．

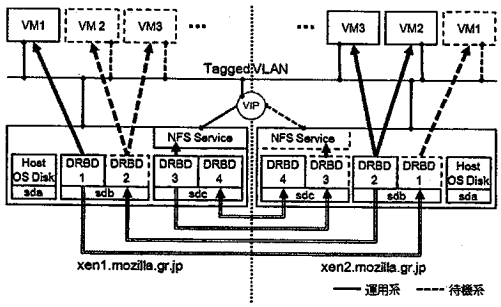

図 4　もじら組サーバファーム構成

4. 実装

前節で紹介した DRBD と Xen を組み合わせたサーバファームの構築例として，法政大学情報科学部がオープンソース支援活動の一環としてハードウェアと回線を提供している，日本における Mozilla コミュニティであるもじら組のサーバ環境を例に紹介する．もじら組とは，1998 年に開発が始まった Mozilla を支援しようと集まった日本のユーザが，2000 年 1 月に立ち上げた mozdev-j メーリングリストのメンバーを中心として 2000 年 12 月から活動している日本の Mozilla 開発者・ユーザのグループである．現在では，メーリングリストや BBS フォーラムといった日本のコミュニティ向けのサービス以外にも，Mozilla の開発者向けドキュメンテーションサイトである MDC (Mozilla Developer Center) への貢献者用にさまざまなツールを提供する MDC Japan Project [12]を稼動させるなど，日本国外にも情報発信を行っており，もじら組のサーバの可用性を高めることは，Mozilla プロジェクト全体にとっても重要である．過去のもじら組のサーバにおいても，システムファンやマザーボード自体の故障により，サーバが停止することがあったため，本サーバファームを利用することで可用性の向上を目指した．

4.1 構成

もじら組サーバファームを構成する物理マシンには，表 1 の構成にホスト OS 用として 80GB のハードディスクを追加し，物理マシン 1 台につき 3 台のハードディスクを搭載したマシンを 2 台使用した．それぞれの物理マシンを xen1，xen2 と呼び，2 台ずつ搭載された 1TB のハードディスクをそれぞれ半分のサイズに分けた領域を DRBD 用として使用し，図 4 に示す構成で DRBD リソースを計 4 つ作成した．DRBD1, 2, 3 は Primary/Secondary 構成の DRBD リソースで，ファイルシステムには ext3 を使用している．DRBD4 は実験的に用意した Primary/Primary 構成の DRBD リソースであるが，現時点では実運用はしていない．通常運用時，物理マシン xen1 では

表 3　仮想マシンの割り振りと割り当てメモリ

xen1 (dom0 + NFS)	1.0GB	xen2 (dom0)	1.0GB
lvs1	0.5GB	lvs2	0.5GB
admin1	1.0GB	admin2	1.0GB
db1	1.0GB	web1	2.0GB
web-ssl1	1.0GB	web-ssl2	1.0GB
mail1	1.0GB	mail2	1.0GB
irc1	0.5GB	ftp1	0.5GB
cron1	1.0GB		
運用	7.0GB	運用	7.0GB
待機	1.0GB	待機	1.0GB

DRBD1, 3 を Primary 状態にし，DRBD1 に物理マシン xen1 上で動作させる仮想マシンのディスクイメージファイルを配置し，DRBD3 にすべての仮想マシンで共有する/home などのデータを配置する NFS サーバ用の領域として使用している．一方，物理マシン xen2 では，DRBD2 を Primary 状態にし，物理マシン xen2 上で動作させる仮想マシンのディスクイメージファイルを配置した．動作させる仮想マシンのディスクイメージはすべて 20GB とし，最大割り当てメモリサイズは表 3 に示す配分にした．また，仮想マシンで動作させる OS はすべて Linux であることから，すべての仮想マシンは性能面で有利な準仮想化環境で動作している．

表 3 のメモリサイズ配分ではどちらかの物理マシンに障害が発生した場合，障害の起きていないもう 1 台の物理マシンで新たに仮想マシンを複数起動するためのメモリが足りない．そのため，縮退環境移行時は，障害の起きていない物理マシンで稼働中の仮想マシンの割り当てメモリサイズを使用状況に応じて Xen の Balloon 機能を使用して動的に減らし，障害により停止した仮想マシンのうち起動する必要がある仮想マシンのみを，割り当てメモリサイズを減らした状態で起動する運用形態をとっている．縮退環境時であってもすべての仮想マシンを起動する必要がある場合は，予め対となる物理マシンで稼働している仮想マシン分のメモリサイズを待機メモリとして確保しておくことが望ましいといえる．

仮想マシン上で動作する各ホストは，ホスト名に従ったサービスを主として提供しているが，すべての仮想マシンで利用する認証情報をプロバイダ・コンシューマ構成の OpenLDAP サーバ，rsyslog による TCP 経由でのサーバログ集約化を仮想マシン admin1，admin2 で一元管理している．DRBD3 領域を export する NFS サーバは，Heartbeat を用いて仮想 IP でサービスし，物理マシン xen1 が停止した場合，自動的に物理マシン xen2 の DRBD3 ブロックデバイスを Primary 状態に切り替え，NFS サーバを起動し，

NFS クライアントへサービスし続ける．仮想マシン上で稼働するサービスのディスク I/O は，NFS サーバで公開された DRBD3 領域へ集中することから，仮想マシンのディスクイメージファイルがある DRBD1，2 の物理ハードディスクと NFS サーバで export する DRBD3 の物理ハードディスクは異なる物理ハードディスクになるように構成した．また，NFS サーバはディスク I/O を重視し，仮想マシンではなく，ディスク I/O のオーバヘッドがより少ない物理マシン上のホストで動作させている．

本サーバファーム移行前に使用していたハードウェア RAID によるディスクの連続読み込み(ブロック単位) が 65647KB/sec，連続書き込み(ブロック単位) が 51250KB/sec であったが，本サーバファームで使用したハードディスク（表 1）が高速（表 2）だったこともあり，DRBD を利用して耐障害性を確保しつつも結果として I/O 速度も向上した．

4.2 ネットワークの仮想化

グローバル IP アドレスに限りがあるため，もじら組専用のローカルネットワークをネイティブ VLAN で構成し，外部へ直接公開する仮想マシンにのみ，タグ VLAN を用いてグローバルネットワークにも所属させた．この際，Xen の DomU のホストにのみタグ VLAN 設定を行っても透過的にグローバル IP アドレスを利用することができるため，サーバファームを構成する Dom0 のホストにはローカル IP アドレスしか割り当てていない．このようにタグ VLAN を利用することで，仮想マシンごとに異なるネットワークに所属させることも可能であるため，十分なネットワーク速度が得られる環境であれば，遠隔地にまたがったサーバファームの構築も可能である．

4.3 ハードウェアメンテナンスの課題

もじら組サーバファームでは，仮想マシンのディスクイメージを Primary/Secondary 構成の DRBD リソースに配置しているため，片方の物理マシンのハードウェアメンテナンスを行うには，一時的にメンテナンスを実施する物理マシン上で動作する仮想マシンをすべて停止し，もう片方の物理マシン上の対応する DRBD ブロックデバイスを Primary 状態にした後，仮想マシンを起動し直す必要がある．可能であればこのような停止時間も無くせるのが望ましい．これを実現する手段として，Primary/Primary 構成の DRBD リソースに GFS や OCFS2 といったクラスタファイルシステムを作成し，仮想マシンのディスクイメージファイルを配置することで DRBD リソースを共有ディスクとして扱い，Xen の提供するライブマイグレーション機能を利用する方法が考えられる．しかし，GFS でフォーマットした Primary/Primary 構成の DRBD リソースの性能を Bonnie++を用いて調査した際，両ホストから同時に 16GB の大きなファイルを書き込み続けると両ホストの DRBD ブロックデバイスが切り離されてしまう現象を確認した．考えうる原因の 1 つとして，片方のマシンで連続書き込み中に，もう片方のマシンからの書き込み要求に再試行回数を超えて応答できなかった結果，切り離されてしまうのではないかと考えられる．詳細な原因の追及と評価は今後の課題とする．

5. まとめ

本稿では，DRBD と仮想化技術を利用した低コストで構築可能な耐障害性を伴うサーバファームを提案し，もじら組での実装例を紹介した．本サーバファームを用いることで，単一マシンのあらゆるハードウェア箇所が故障しても，サービスを短時間に復旧することができるようになる．また，汎用的な部品で構成されたマシンが 2 台あれば実現できることから，高いディスク I/O 性能が要求されない数多くのシステムで低コストに適用可能である．サーバファーム全体のマシン資源の使用面においても，2 つ以上の DRBD リソースを利用し，それぞれの物理マシンで仮想マシンを起動することで無駄なく有効に使用できる．なお，本サーバファームは，個々のハードウェア多重化やアプリケーションレベルでの冗長構成を不要とするものではなく，それらを組み合わせることにより，より高い信頼性を兼ね備えたシステムを実現することができる．

今後の課題としては，DRBD の Primary/Primary 構成時での安定稼働の調査検討や DRBD と仮想マシンを一元的に管理可能な管理ソフトウェアの開発が挙げられる．

参考文献

[1] DRBD:What is DRBD, http://www.drbd.org/.
[2] Home of the Xen hypervisor, http://www.xen.org/.
[3] Home of the Mozilla Project, http://www.mozilla.org/.
[4] もじら組, http://www.mozilla.gr.jp/.
[5] examples:skip-initial-sync [DRBD+doku], http://drbd-plus.linbit.com/examples:skip-initial-sync.
[6] Heartbeat: Linux HA, http://www.linux-ha.org/Heartbeat.
[7] Keepalived for Linux - Linux High Availability, http://www.keepalived.org/.
[8] Red Hat Global File System, http://www.redhat.com/gfs/.
[9] Project: OCFS2, http://oss.oracle.com/projects/ocfs2/.
[10] Bonnie++, http://www.coker.com.au/bonnie++/.
[11] lsyncd - Live Syncing (Mirror) Daemon, http://www.pri.univie.ac.at/index.php?c=show&CEWebS_what=Lsyncd.
[12] A. Shimono, "Mozilla Developer Center Japan Project," Mozilla 24 US Event (Stanford University), 15 September 2007.