2007－OS－105 (9)
2007／4／5

Brooklet: Intel-VTを活用したApacheのためのVMM

三木 聡 † ‡　　追川 修一 †
筑波大学 †　　株式会社フィックスターズ ‡

Virtual Machine Monitor（VMM）を利用したハードウェア仮想化技術が急速に普及している．サーバやデスクトップ分野で，ハードウェアの統合が進み，消費電力の削減や，TCO（total cost of ownership）の削減に貢献している．ところがその多くは，多機能化や汎用化をもとめ肥大化の一途を辿っている．本論文では，今後広くサポートされるであろうプロセッサの仮想化支援機能を取り入れ，産業界のニーズに照らし合わせて，Apacheサーバ統合のためのVMMを提案する．

Brooklet: VMM for Apache that makes the full use of Intel-VT processors

Satoshi Miki † ‡　　Shuichi Oikawa †
University of Tsukuba †　　Fixstars Corporation ‡

Virtual Machine Monitors (VMMs) have rapidly come into wide use in servers and desktops. They help saving power consumption and cutting down the total cost of ownership. However, many VMMs have become bigger and bigger because of the support of a number of rich software features and multiple hardware platforms. This paper describes the requirements and design principles of a VMM that meets industrial requirements by utilizing new hardware virtualization technology.

1　はじめに

1台のコンピュータを仮想的に複数台にみせるソフトウェア技術，Virtual Machine Monitor（以下VMM）がPCサーバにおいても，実用的な段階にきている．VMMは1960年代からIBM System/360 Model 67上で実用化されるなど古い歴史を持つが，昨今のPCの高性能化に伴い，PCサーバ上での仮想化技術が進歩し，サーバ用途およびデスクトップ用途として，広く利用されるようになった．IDCの調査によると，2006年には230万台の仮想サーバが稼動している．仮想化ソフトウェア最大手のVMwareの2006年の売上は7億9000万ドルを記録し，四半期ベースで前年比2倍程度の成長を持続している [1]．また，ケンブリッジ大学で開発された仮想化ソフトウェアXen [2]は，現在もオープンソースプロジェクトで活発に開発が続けられ，XenSourceによる商用化も行われている．

このようなVMMの普及に伴い，インテルやAMDのようなプロセッサメーカもソフトウェア仮想化のための支援機能をプロセッサに付加しはじめた．インテルは，32ビットプロセッサ向けIntel-VTxおよび64ビットプロセッサ向けIntel-VTiをリリースし，更にI/Oデバイスを仮想化するためのIntel-VTdを発表している [3]．同時に，仮想化支援機能をXenに組み込むためのソフトウェア開発支援も行っている [4]．AMDはSecure Virtual Machineという名前で，仮想化サポートを定義しており，さらにIOMMUというI/Oを仮想化するMemory Mapping Unitの実装を計画している [5]．

一方，ソフトウェアに目を向けると，前述の代表的なVMMである，VMwareやXenはプロセッサ・サポートのないハードウェアへの対応，多くの機能拡張やマルチプラットフォームサポートのため，プログラムコードが肥大化し，徐々にメンテナンスが困難な状況に陥っている．

本論文では，産業界で広く使われているApacheが稼動するWebサーバを統合するためのVMMに焦点を絞り，実際の稼働状況を調査し，求められる要件を検討し，Apacheサーバ統合のためのVMMの設計について検討する．

2　Webサーバ稼働状況

著者の一人が代表を勤める株式会社フィックスターズでは，関連子会社も含め，ドメインベースで77のWebサイトを運営し，81台のサーバを常時稼動させている．サーバ81台全てがプロセッサとしてインテルを採用しており，2台のWindows2000サーバを除いてオペレーティングシステム（OS）は，UNIXまたはLinuxで構成されている．これらサーバ群のうちApacheまたはIISが動作しているサーバは41サーバある．今後1〜2ヶ月の間に，9台の増設が決まっており，サーバの数は増加の一途を辿っている．サーバの増加に伴い，サーバ購入費用負担だけでなく，データセンタ利用料金や，監視や障害対応など運用人件費コストの増大が大きな問題となっている．

2.1 稼働サーバの OS

表 1 に示すように，サーバで稼働中の OS はクライアントの要求や導入時期の安定性などにより多様化している．本来なら OS のバージョンアップやセキュリティアドバイザリの勧告にあわせ，運用サーバの OS もバージョンアップするべきであるが，ビジネスの局面では予算の関係やサービス無停止の要求から，バージョンアップを見送るケースが多い．バージョンアップを見送ったシステムはハードウェアの耐久年数が近づき，ハードウェアを入れ替えるタイミングで OS のバージョンアップが行われる．

表 1: 利用されている OS

ディストリビューション	台数
Red Hat7.1	1
Redhat9	5
Fedora Core 4	3
Fedora Core 5	13
Fedora Core 6	1
CentOS 4.2	1
CentOS4.3	7
CentOS 4.4	9
Vine Linux 2.6r4	1
Vine Linux 3.1	6
FreeBSD 6.2-PRERELEASE	2
RedHatES3	18
RedHatES4	12
Windows2000 Server	2
合計	81

2.2 一般的な Web サーバの構成

Web サイトのサーバ構成は大きくアクセスが少ない Web サイト用に構成する小規模サーバ構成と，アクセスが常時大量に発生し高い可用性を求められる，大規模サーバ構成に分類できる．

2.2.1 一般的な小規模 Web サーバの構成

小規模な Web サイトを運営する場合，図 1 に示すような Firewall，Web サーバ，データベース（DB）サーバの階層で構成することが多い．この場合，複数の Web サーバから，1 台の DB サーバのデータを参照する構成となる．現在の Web サイトの多くが，DB サーバをバックエンドに持ち，多くの処理を DB サーバ側で行い，Web サーバ側ではユーザ・リクエストに答えるだけの処理を行う．そのような構成のため，安価な Web サーバをフロントに設置し，比較的高価で信頼性のある DB サーバをバックエンドに置くような構成となる．

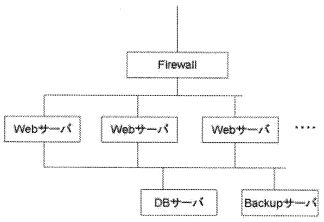

図 1: 一般的な小規模 Web サーバの構成

複数 Web サーバから 1 台の DB サーバを参照する理由は以下の通りである．

- 商用の DB サーバを利用する場合，ライセンス料が高価なため，なるべく少数構成で済ませたい．そのため，既存の DB サーバがあればそちらを利用する場合が多い．
- Web サーバ運用の際に，必要なデータをバックエンドの DB サーバに置き，Web ページを表示する際には，標準化された問い合わせ言語である SQL を使って DB サーバ上のデータを参照，更新すればよい．そのため安定運用可能であれば特に DB サーバの種類にはこだわらないケースも多い．よってバックエンドに複数 Web サイトで共用の DB サーバがあれば，新規に Web サイトを立ち上げる際にも既存の DB サーバを利用することが多い．また DB サーバの移行は，問い合わせ言語 SQL が標準化されているため，Java から PHP といったようなアプリケーションの移行に比べ容易なことが多く，もともとのシステムのサーバ環境が異なる場合の統合でも，低予算で行うことが可能である．
- 複数 Web サイトで共用の DB サーバを利用すると，複数サイトの運営を一企業が行う場合，バックアップ，メンテナンス，監視などが容易になり，運営費が抑えられる．

2.2.2 一般的な大規模 Web サーバの構成

アクセスが常時大量にあり，高可用性が要望される Web サイトの場合，図 2 に示すように最上位層に Firewall 2 台，第二層にロードバランサを 2 台置き，第三層に複数台の Web サーバを並べ，第四層に複数台の DB サーバを置くことで，No Single Point of Failure を実現する．このようなサーバ構成にすると，サーバ導入費用は先に示した小規模 Web サーバ構成に比べると 10 倍〜100 倍程度増大する．また，障害時の原因解明，監視体制，バックアップなど，運用コストも 2〜5 倍程度増える傾向にある．

このように高コストを負担しても，大量の処理を行い，高可用性を要望しているケースが多くある．CPU使用率は頻繁に90 %を超えるため，このようなサーバを統合することは考えられない．

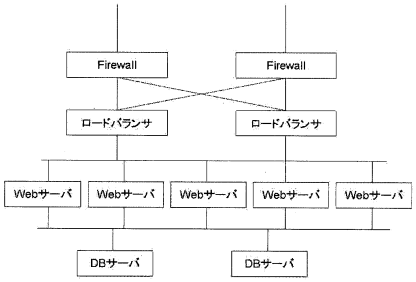

図 2: 一般的な大規模 Web サーバの構成

2.3 Web サーバの負荷状況

Web サイトの種類により，常時大量のアクセスのある大規模 Web サーバと，中小企業の会社案内のウェブサイトや，特定の製品情報を提供するウェブサイトのようにほとんどアクセスがない小規模 Web サーバに分かれる．また，図 3 に示すように，企業が公開している Web サイトの場合，公開用サーバのほかに，開発用サーバと，事前確認用サーバを必要とする場合があり，1 つの Web サイトを運営する場合でも 3 台の Web サーバが必要になることも多い．

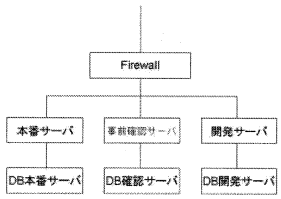

図 3: 本番サーバ，開発サーバ，事前確認サーバの構成

アクセスの少ないサイトはアイドル時間が長く，また開発用サーバと，事前確認用のサーバ，障害時のための待機系サーバは普段はまったくアクセスが無い．図 4 に比較的アクセスの少ない Web サイトのアクセス状況を示す．図 4 の 3 つの Web サイトは Apache の Virtual Host 機能を使用し，1 台の Web サーバと 1 台の DB サーバの構成で運営されている．

上位回線は 10Mbps の共有回線となっており，サーバが置かれているネットワークセグメントはすべて 1000Base-T のイーサネットで接続されている．www サイト，recruit サイトは静的 HTML が多いサイトで，cell サイトは利用頻度の高いコンテンツ・マネジメント・システムである MediaWiki を利用している．3 つのサイトを合計すると，多い日で 1 日 2 万 5 千を超えるアクセスがあるが，その場合でもサーバのピークロードアベレージは 0.8 である．図 6，図 7 に示すように，ネットワークトラフィックをみても，1Gbps 接続されている Web サーバー DB サーバ間で，平均 2 Mbps を記録しているが，Web サーバーインターネット接続間のトラフィックは，インバウンドが 50 Kb 未満，アウトバウンドでも 800 Kb を超える程度となっている．

表 2: Web サーバのハードウェアスペック

プロセッサ	Intel Pentium(R) D 930 CPU 3.00GHz x 2MB L2Cache
メモリ	2GB (4x512MB 1R) DDR2/533MHz バッファなし SDRAM DIMM ECC
ハードディスク	160GB 1 シリアル ATA ハードディスクドライブ (7200 回転) × 2 台 RAID 1 構成

一方，常時大量のアクセスがあるサーバの負荷は高く，常に CPU 使用率や，メモリ使用率を有人監視にて行っているサーバも多い．図 8 に常時大量のアクセスのある Web サイトのアクセスデータを示す．この Web サイトの例では，アクセスが少ない日でも 20 万アクセスを超え，アクセスが多い日には，140 万アクセスに近づいている．

この Web サイトの場合，システム停止が企業の売上減に直結するため，24 時間運営無停止が必須となっており，ハードウェアのバージョンアップや，システムメンテナンスの際も，サービス無停止を前提に行っている．

2.4 サーバ統合の要求

現在，フィックスターズではデータセンタ内に 6 本のラックを借り，別に社内に 2 本のラックを置き，各種サーバの運営を行っている．サーバ負荷を考慮すると，多くのサーバが資源を持て余しており，コンピュータ資源面だけを考慮すれば多くのサーバが統合可能である．

Web サーバの統合状況をみると，Apache VirtualHost 機能を使い，24 の Web サイトを 1 台の Web サーバで運営している例もある．どのような場合にサーバ統合が可能になるか考えてみると，新規に Web サイトを構築し，かつ，運用まで一式で委託する場合に，既存のサーバとの相乗り運用が可能となる場

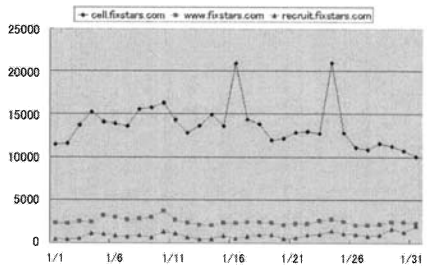

図 4: 2007 年 1 月低負荷 Web サイトのアクセス数

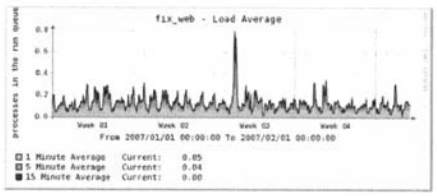

図 5: 2007 年 1 月低負荷 Web サイトの Web サーバロードアベレージ

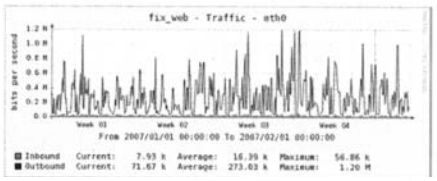

図 6: 2007 年 1 月低負荷 Web サイトのインターネット側トラフィック

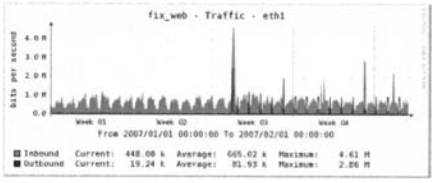

図 7: 2007 年 1 月低負荷 Web サイトの DB 側トラフィック

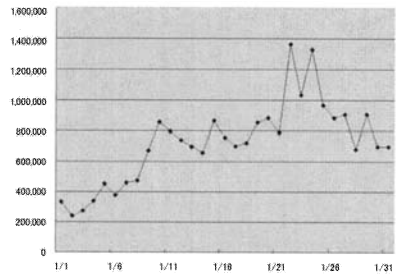

図 8: 2007 年 1 月高負荷 Web サイトのアクセス

合が多い．これは，システム動作環境をサーバ運用側で決定することが可能となり，現在稼働中のサーバ環境にあわせ，新規にシステムを構築できるためである．

サーバ統合を行うと，サーバ購入費用，データセンタ費用，運用費用の削減という経済的メリットのほかにも，消費電力の削減が可能となり地球環境に対してもプラスの影響が出る．

2.5 Apache VirtualHost 機能を利用したサーバ統合とその問題点

複数 Web サイトを 1 台のサーバで運営する手段として，Apache VirtualHost 機能が広く利用されている．この VirtualHost 機能を利用すると，1 つの OS 上で 1 つの Apache デーモンを動作させ，その上で複数 Web サイトを運営することが可能となる．フィックスターズで運営するサーバ統合はすべて，この VirtualHost 機能で実現されている．しかし，全てのサーバを統合することはできない．

複数企業の Web サイトを運営する際には，以下の理由で VirtualHost 機能だけでは，すべての Web サイトを 1 台の物理サーバに統合できない．

- 動作環境が異なる

  時代とともに，OS や Apache，PHP などのプログラム動作環境のバージョンがあがる．動作環境のバージョンがあがると，プログラムの修正が必要になったり動作が不安定になる場合がある．そのため，既存のシステムを移行してくる場合など，既存の動作環境に移行先の動作環境が適用できず，その結果，新規に Web サーバが必要になることが多い．

- 管理者権限の要求

  企業やユーザによっては，OS の管理者権限を要求してくる．Apache はユーザプロセスとして動作しているので，VirtualHost のサイト管理者ごとに，オペレーティングシステムの管理者としてのアクセスは認められない．

- 動作保証上の問題

  一つの Web サイトのプログラムミスや，集中的なアクセスにより，同一サーバで動作している全 Web サイトが機能しなくなることがある．ある企業の Web サイトのアプリケーション不具合により，別の企業の Web サイトが動作しなくなったという説明は，サーバ運営上あってはならない．フィックスターズの場合，VirtualHost 機能を使っている Web サイトは全てフィックスターズ内で，開発から運営まで任されているもので，他企業の作成した Web サイトは相乗りさせていない．

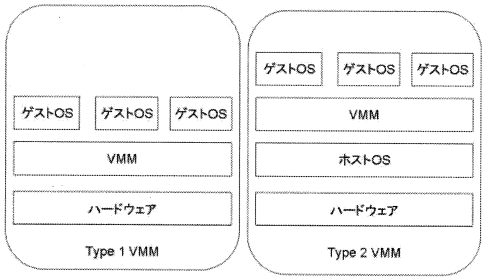

図 9: Type1 VMM と Type2 VMM

- データ保護の問題

  Web サイトのアプリケーション不具合や，サーバ設定ミスにより，ある Web サイトのデータがほかの Web サイトの管理者に漏洩したり，またはクラックした第 3 者に漏洩する可能性がある．そのため，ほとんどアクセス数がない Web サイトを運営する場合でも新規にサーバを購入する例も多い．

3 仮想化ソフトウェアを利用したサーバ統合とその問題点

上記の問題を踏まえ VMM を利用したサーバ統合がすすんでいる．

3.1 Type1 と Type2 の VMM デザイン

図 9 に示すように，VMM は大きく Type1 と Type2 の 2 種類に分けられる．Type1 VMM はハードウェア上で VMM が動作し，その上でゲスト OS が動作する．Type2 VMM はハードウェア上でホスト OS が動作し，その上で VMM が動作し，更にその上でゲスト OS が動作するものである．Type1 VMM の代表ともいえる Xen はオープンソースで開発されており，2007 年 2 月 20 日現在バージョン 3.04 がリリースされている．

Type2 VMM の場合，Linux や Windows といった大規模システムをホスト OS として動作させ，ホスト OS 上のユーザプロセスとして，ゲスト OS を動作させる．そのため，ホスト OS 側のデバイスドライバのバグなどでシステムがクラッシュするとゲスト OS 側のシステム全体がストップする可能性が高い．一方，Type1 の VMM の場合，ハードウェア上で動作しているのが VMM であるため，VMM を小さく保つことにより，バグの発生確率を押さえ，システム全体が停止する可能性を低く保つことが可能となる．

3.2 仮想化ソフトウェア・ソースコードステップ数の増加

Type1 VMM Xen3.0 の C 言語ステップ数は実効（空行，コメント行を除く）のみで 15 万 907 行となっている[1]．Xen のバージョン 1.2 から 3.0 までのステップ数の遷移を表 3，表 4 に示す．Xen1.2 ではデバイスドライバを Xen 内部に必要とするのに対し，Xen2.0 ではデバイスドライバを特殊なゲスト OS であるコントロールホスト側に移行したため，30%程度にステップ数が減っている．また，Xen2.0 から Xen3.0 へはマルチアーキテクチャへの対応や，SMP 対応，Intel-VT 対応などが行われたため，ステップ数が再膨張している．今後の開発ロードマップ [7] をみると，更に機能が盛り込まれ，複数プラットフォームがサポートされる．その結果留まることなく，ステップ数は増え続けることが予想できる．

次に Type2 VMM の例をみる．UML [8] は Linux カーネルへのパッチ形式で配布されている．uml-patch-2.4.7-1 は 20,177 行であるのに対し，uml-patch-2.4.27-1 は 46,311 行のパッチとなっている．Kernel 2.6.9-rc2 以降はカーネルに取り込まれている．カーネル 2.6 以降は，ホスト OS とゲスト OS の両方にパッチを当てる形式となっている．ホスト OS 側のパッチである skas-2.6.20-v8.2.patch は 778 行であり，ゲスト OS 側のパッチである uml-2.6.18.1-bb2.patch は 5,555 行である．Linux KVM [9] は Linux Kernel ドライバ形式で組み込むタイプの VMM で，Intel-VT および AMD SVM 対応プロセッサが必須となっている．KVM は Intel-VTx を使い実装されているため，表 5 の通り，KVM カーネル部分のソースコードは最新バージョン 1.5 で，実効 10,299 行と Xen と比べわずか 6%と非常に小さい．ただし，Linux のデバイスドライバという形式で配布されるため，Linux カーネルのバージョンアップに伴い，常に KVM カーネルの保守が必要となる．

表 3: Xen バージョンごとのステップ数の遷移

Ver.	Xen1.2	Xen2.0	Xen3.0
acm			1,502
arch	8,264	19,062	85,259
common	7,227	7,608	11,516
drivers	128,752	3,189	12,104
include	15,247	18,698	38,604
net	2,124		
tools	1,534	1,446	1,922
合計	163,148	50,003	150,907

[1]Eclipse3.22 と StepCounter1.14 を使って計測．拡張子が.c 及び.h のファイルのみを対象

表 4: Xen アーキテクチャ依存部ステップ数の遷移

Ver.	Xen1.2	Xen2.0	Xen3.0
arch/x86	8,264	16,352	40,758
arch/ia64		2,710	34,478
arch/powerpc			10,023
合計	8,264	19,062	85,259

表 5: KVM1.5 のステップ数

drivers	381
kernel	10,299
qemu	197,053
合計	207,733

3.3 肥大化の原因

Redhat6.2 のカーネルステップ数 150 万行からほぼ 1 年後にリリースされた Redhat 7.1 に含まれる Linux kernel-2.4.2 のステップ数は 2,437,470 行となっている．うち 57 %がデバイスドライバで，18 %が arch ディレクトリ内に存在する．

バージョンアップに伴い，機能拡張，対応プラットフォーム拡張，性能向上が繰り返され，結果的にステップ数が膨れ上がってきた．Xen2.0 から Xen3.0 へのバージョンアップを例にみると，PowerPC プロセッサへの対応で 1 万行，SMP 対応，VT-x 機能などの機能拡張で大幅にステップ数が増えている．

またオープンソースプロジェクトとして開発が進むと新しい機能が必然的に盛り込まれ，結果的にコードが肥大化しメンテナンス不能状態に陥るという循環が生まれる．

3.4 肥大化の問題点

システム肥大化の問題点としては，第一にソースコード中に潜むバグが増えることが挙げられる．バグの増加によりシステム障害の可能性も高まるため，ハードウェアを直接管理する VMM や OS といった基本ソフトウェアは極力小さく保たれることが好ましい．Linux カーネルも近年肥大化の一途を辿っており，LinuxTiny [10] のような小さなカーネルを目指す動きも高まっている．

第二に，機能が増大し複雑化すると，障害時の原因の切り分けが難しくなる．障害発生から，原因究明，対策に至るまでのダウンタイムを最小限にするためにも，各システムは単純に構築されるべきである．

第三に，システム運用の面から考察すると，新しいソフトウェアを導入する際には，特定機能に限定されたソフトウェアを，複雑な設定なしで簡単に利用できることが望まれる．機能を盛り込みすぎて，多くの初期設定が必要となるようなソフトウェアの導入は見送られる場合が多く，広く普及しない．運用担当者に対しても特別な知識を要求するソフトウェアも多く，運用者教育や，導入コストがあまりにも高額なため導入を見送る場合も少なくない．

4 Brooklet のシステムデザイン

以上の問題点を踏まえ，本研究では Apache を動作させるための VMM Brooklet を構築する．

4.1 システム要件

- Apache が問題なく動作すること

  今後ますます増えていく Web サイトに対応すべく，Apache の動作を優先し，Apache 以外のアプリケーションに対しての最適化は考えない．またリアルタイムに画像を生成，リサイズするなどの CPU 演算を多用するタイプの Web サイトの統合も考えない．ネットワーク I/O が主であるタイプの Web サーバを統合するための VMM を作成する．

- 問題の切り分けが簡単にできること

  ゲスト OS のメンテナンス性，単純性を優先し，多少のパフォーマンスは犠牲にしても，ゲスト OS は極力無修正のコードが動作するようにする．プロセッサ性能の劇的な向上にともない，CPU 演算能力の不足は時間が解決するため，今後のソフトウェアには，性能よりも堅固さや安全さが求められる．

- OS 間の独立性を保つ

  独立したホストマシン上の OS を利用するのとまったく同様に仮想マシン上のゲスト OS を利用できるよう，ゲスト OS 間はデータの独立性，プロセスの独立性を保たなければならない．

  また，一つの OS 上のアプリケーション負荷により，システム全体が停止することを避ける．1 台のサーバで複数ユーザが複数 OS を同時実行する際には，障害の隔離が重要要件となる．VMM のような基本ソフトウェアが普及する条件として，意思決定者への説得が不可欠となる．そのため，完全に物理サーバと OS が 1 対 1 で対応する形式の現在広く普及している方法と同条件で，VMM 上の仮想マシンが動作することを保証する必要がある．

4.2 システム設計

VMM を採用するということは，ハードウェアと OS の間に，新しくソフトウェアの層を挟むことになる．新規にソフトウェアを追加するということは，その分不具合発生率が上がることになる．本研究により，既存のシステムの安定性を損なうことは出来る限り避けなければならない．これが Brooklet のシステム設計全体を通して，全ての事項に優先される．よって，Brooklet は極力小さなものに仕上げる．全てのケースで性能よりも単純性を優先し，“Keep It Simple, Stupid”（KISS）を厳守する．ただし，多くの場合メリットとデメリットのトレードオフが発生する．Apache サーバを統合するための VMM 設計を妥協点をみつけながら進める．

4.2.1 Type1 VMM を採用

以下の理由で Type1 VMM を採用する．

- 障害時の原因切り分けが可能

  Type1 VMM を採用すると，全仮想マシンが停止するような障害発生の原因をハードウェア障害によるものか，または VMM ソフトウェアの不具合によるものかに分類可能となる．なぜなら，全仮想マシンが停止するということは全仮想マシンが依存している何かに障害が起きていることが想定でき，その何かとはハードウェアか VMM に限定できるからである．一方，Type2 VMM を採用してしまうと，仮想マシン全体が依存するのは，VMM およびホスト OS およびハードウェアになるため，ホスト OS が原因なのか VMM が原因なのか切り分けがしにくい．以上の理由から Type1 VMM を採用する．

- 資源管理を VMM が完全制御できる

  Type2 VMM の場合，仮想マシンへの資源割り当てがホスト OS に依存してしまう．一般的に OS のスケジューリングはプロセスを対象に考えられているため，必ずしもゲスト OS への資源割り当てに適さない．例えば，ホスト OS からみるとゲスト OS は単なる 1 プロセスにみえるため，ゲスト OS 中のプロセスの優先度をホスト OS は解釈できない．今後 VMM の研究が進むにつれ，仮想マシンへの資源割り当てアルゴリズムが研究されていくが，その場合 VMM 側で完全にスケジューリングが制御できなければならない．

- VMM 単体での保守が可能

  現在のハードウェア技術では，VMM が完全な形でゲスト OS に仮想マシンを提供出来るわけではないが，今後ハードウェアとソフトウェアが協調する形で仮想化技術が進歩していくと，ゲスト OS に完全な形で仮想マシンを提供することが可能になることが期待できる．ハードウェア仮想化支援と Type1 VMM の協調により，ゲスト OS に完全な形の仮想マシンが提供できると，VMM とゲスト OS が完全に切り離すことが可能となり，VMM 単体での保守が可能となる．一方，Type2 VMM はホスト OS に依存してしまう．ホスト OS の変更に伴い，VMM の変更も必要となってしまう．

4.2.2 Intel-VTx を利用した完全仮想化 VMM

安定して単純な仮想マシン環境を提供するために，ホスト OS のプログラムはなるべく変更を避ける必要がある．パフォーマンスへの影響を考えると，Linux のようにソースコードを提供している OS に対しては，一部ゲスト OS 側のカーネルソースコードを変更するパラバーチャライゼーションの採用も検討に値するが，障害時の原因切り分けが困難になる．よって，カーネル部分の修正は行わない．

このような完全仮想化による VMM を実現するために，広く利用されている Intel-VTx 対応プロセッサを動作対象とする．

これまでの VMM はプロセッサの仮想化支援がない状況で，何らかの方法を使いソフトウェアにより仮想マシンを実現してきた．本 VMM はハードウェアによる仮想化支援機能を使い，また仮想化支援のないプロセッサを動作対象から除外することにより，ソフトウェアを堅固なものにする．

そのために，インテル社のプロセッサに広くサポートされるであろう仮想化支援機能である，Intel-VTx 機能が搭載されているプロセッサを VMM 動作対象とする．

4.2.3 マルチコア CPU を前提とした非対称スケジューリング

VMM は各仮想マシンにネットワークトラフィック量に基づいた，加重ラウンド・ロビンによりコンピュータ資源を割り当てるようにスケジューリングを行う．

VMM 上で動く仮想マシンが増えるにつれ，I/O 割り込みの際に対象仮想マシンがアクティブである可能性は低くなっていく．そのため，イベントドリブンで OS を呼び出すことはしない．

このようなデザインを採用するとネットワーク I/O 性能が極端に悪くなることが広く知られている．よって，本 VMM では I/O 処理を行う専用のコントロール OS を VMM 上で動作させ，1 つの CPU コアを独占的に割り当て，全てのゲスト OS が必要とするネットワーク I/O をコントロール OS に代理処理さ

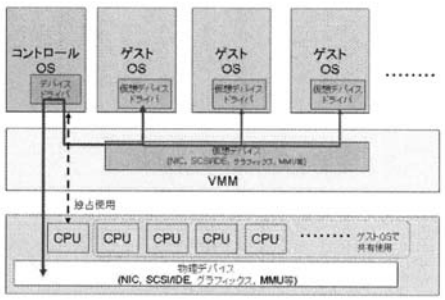

図 10: Brooklet のシステム設計概要図

せる．そのほかのゲスト OS はコントロール OS と共有メモリを使ってデータをやり取りすることにより，ネットワーク I/O 性能を向上することが可能となる．

このように，非対称な CPU スケジューリングを行うことにより，ゲスト OS 間で CPU 資源を効率的に利用することができる．その結果，1 つの OS 上のプロセスが CPU を独占することがなくなり，OS 間でのパフォーマンス独立性が高まる．

4.2.4 I/O デバイスドライバ

VMM を安定させるために，可能な限りソフトウェアは VMM の外で実行する．よって，デバイスごとに増えていくデバイスドライバは VMM 中には含めない．ゲスト OS 用に，仮想マシン用のデバイスドライバを提供し，その命令を VMM が受け取る．受け取った I/O 命令を VMM から，特殊なゲスト OS であるコントロール用 OS に再委託し，コントロール OS からデバイスに対して命令を発行する．

将来的には，次世代のインテル社製プロセッサに搭載される予定の Intel-VTd が普及し，I/O デバイスにも仮想化機能が搭載されると，ゲスト OS からデバイスに対して直接命令を発行できるようになり，仮想化による性能劣化は限られたものになる．現在では PCI-Express SPEC 2.0 に IOV（I/O Virtualization）という名称で，PCI-Express デバイスの仮想化支援仕様が正式発表され，ネットワークカードなどの PCI デバイスはデバイス自体に仮想化機能が搭載され，ゲスト OS からダイレクトに命令を発行できるようになることが期待されている．

5 関連研究

小さくセキュアな VMM の研究では，Denali [11] がある．Denali はゲスト OS を変更するパラバーチャライゼーションを採用しているが，Blooklet は完全仮想化を採用する．Intel-VT を活用した例では，前述の Linux KVM がある．Linux KVM はホスト OS にロードモジュール形式で提供される，Type2 VMM に分類されるが，Blooklet はハードウェア上で直接 VMM を動作させる Type2 VMM に分類される．Xen は Type1 VMM かつ Intel-VTx 対応だが，多機能化が進み，巨大化の一途を辿っている．Brooklet は Apache サーバの統合のみを目的とした構成にする．

6 まとめおよび今後の課題

本論文では，実稼働中の Web サーバのデータより，VMM に求められる要件をまとめ，産業界で採用される VMM の方向性を検証した．本研究をベースに，今後サポートされるであろうハードウェアの仮想化支援技術を活用した実用的で安定した VMM の構築を行う．VMM 構築後は実際の Web サイトの運営に利用し安定化を目指す．また，Xen や VMware とのパフォーマンス性能評価や安定性能評価を行う．

参考文献

[1] ヴイエムウェア株式会社 : “ヴイエムウェア，ソフトウェア分野における驚異的な成長を経て 9 周年を迎える” , http://www.vmware.com/ja/news/pdf/20070130b.pdf

[2] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield : “Xen and the Art of Virtualization” , SOSP, 2003.

[3] インテル株式会社 : “IntelR Virtualization Technology for Directed I/O” , Intel Corporation.

[4] インテル株式会社 : “Extending Xen with IntelR Virtualization Technology” , Intel Corporation.

[5] AMD : “AMD I/O Virtualization Technology (IOMMU) Specification” , AMD, 2006.

[6] Intel Corporation: “Intel 64 and IA-32 Architectures Software Developer’s Manuals” , Intel Corporation, 2006.

[7] Ian Pratt: “The Xen Roadmap” , 2006.

[8] Jeff Dike: “User-Mode Linux” , 2001.

[9] qumranet: “KVM: Kernel-based Virtualization Driver” , Qumranet inc, 2006.

[10] Matt Mackall: “Linux-tiny And Directions For Small Systems” , Linux Symposium, 2004.

[11] Andrew Whitaker, Marianne Shaw, and Steven D. Gribble: “Denali: Lightweight Virtual Machines for Distributed and Networked Applications” , USENIX, 2002.