Vol.2010-OS-115 No.8
2010/8/3

IaaS 環境における仮想ディスクの効率的な同期手法

殿崎 俊太郎[†1] 山田 浩史[†1,†2]
吉田 哲也[†1] 河野 健二[†1,†2]

近年，Infrastructure as a Service (IaaS) を活用したサーバ運用が注目を浴びている．IaaS 環境では，負荷状況に応じて仮想マシン (VM) の数を自由に増減させることが可能であり，コストパフォーマンスの高い運用が可能であると言われている．しかし，VM のソフトウェア構成やコンテンツを変更する場合には，新しい VM イメージを作成し，そのイメージから新たに VM を起動する必要がある．これは変更が反映されるまでに長い時間を要し，また多くリソースを消費するため，小規模・頻繁な変更には適さない．

本論文では，差分転送と VM イメージの非同期作成により，短時間かつ省リソースで VM を更新する手法を提案する．本手法は，開発者が開発用 VM において更新したファイルを，稼働中の全公開用 VM に転送する．公開用 VM では，更新するファイルの種類に応じ，プロセス再起動等の必要な処理を判断し自動的に実行する．また，今後新たに VM を起動するためのイメージファイルは，仮想ディスクイメージレポジトリと呼ぶ別のサーバ上で非同期的に生成する．これにより，比較的長い時間を要するイメージファイルの生成が完了するのを待つ必要がなく，新たに VM を起動する必要も無いため，短時間で変更を反映させることが可能となる．また，ネットワーク帯域や計算リソースの消費も従来手法より減らすことが可能である．

Efficient Synchronization of Virtual Disks in IaaS Environments

Shuntaro Tonosaki,[†1] Hiroshi Yamada,[†1,†2]
Tetsuya Yoshida[†1] and Kenji Kono [†1,†2]

Infrastructure as a Service (IaaS), a form of cloud computing, is gaining attention for its capability of enabling efficient server administration in dynamic workload environments. The number of virtual machines (VMs) may be increased or decreased over time to adopt with the changing size of workload. It is therefore possible to maintain high cost-performance, mainly by the unnecessity of overprovisioning. In such environments, however, it consumes relatively long time and large amount of resource to update the software stack or content files of VMs. The administrator must create a new VM image, and launch new VMs from it. Small, frequent updates are not suitable in IaaS environments.

In this paper, we introduce a technique for synchronizing virtual disks by transferring only the updated files and by creating the new virtual disk image asynchronously. Files updated on the development machine are recorded and transferred to all public virtual machines via the *virtual disk image repository*. On the public VMs, necessary procedures such as restarting server processes are taken automatically according to the types of files that are updated. In the meantime, a new virtual disk image will be created asynchronously on the virtual disk image repository and will be used for subsequent launch of additional public VMs. Using our technique, the content files and the software stack of all public VMs can be updated dramatically faster. Updates become effective without waiting for time-consuming operations, such as VM image creation and VM startup, to complete. In addition, consumption of resources, such as network bandwidth and CPU time, can be held down using our technique.

1. はじめに

近年，クラウドコンピューティングが注目を浴びている．クラウドコンピューティングとは，ソフトウェアやサーバ資源などをインターネットを通じてサービスとして提供する仕組み全般を指す．提供されるサービスの内容により，Infrastructure as a Service (IaaS) や Software as a Service (SaaS) などに分類される．IaaS は CPU，ストレージ，ネットワーク帯域などのサーバ資源がインターネットを介して提供されるものを指し，代表的な例として Amazon Elastic Compute Cloud (EC2)[1)] が挙げられる．SaaS はソフトウェアがインターネットを介して提供されるものを指し，代表的な例として Windows Live Mail[2)] や Google Calendar[3)] などが挙げられる．本論文では， IaaS に着目する．

IaaS 環境の最大の特徴は，必要な時に必要なだけ資源を使用できる点である．資源の量を自在に調整できる性質を伸縮性 (elasticity) と呼ぶ[4)]．従来のデータセンタではピークデマンド時の負荷に対応できるだけのサーバ資源を用意して運用していたため，初期費用，維持費用共に高かった．IaaS 環境では必要な時に必要なだけ資源を使用することができ，常に必要最小限の資源を用いる事が可能である．例えば Amazon 社のクラウドサービスでは，

†1 慶應義塾大学
Keio University
†2 科学技術振興機構 戦略的創造研究推進事業
JST CREST

Vol.2010-OS-115 No.8
2010/8/3

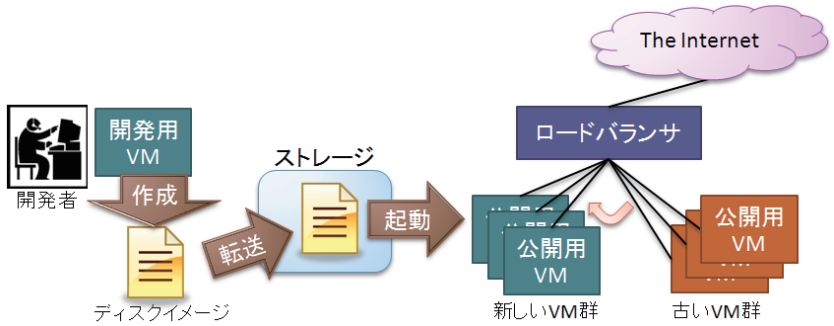

図 1 従来の VM 更新手法

負荷状況に応じて自動的に VM の数を増減させることが可能である．負荷が増大した場合にはユーザが予め作成しておいたディスクイメージから VM が起動され，負荷が減少した場合には動作中の VM のいずれかが停止される．サービス使用料は使った資源の量と時間に応じて計算されるため，負荷に見合った維持費を保つことが可能となる．また，予想外の大きな負荷が発生した際にもすぐに資源を増やして対応することが可能である．

IaaS 環境において， VM を立ち上げた後に各 VM の構成を変更する必要性が生じる場合がある．例えば HTTP サーバを運用している場合は，コンテンツ (HTML ファイル等) の更新，HTTP サーバの設定変更，HTTP サーバのアップグレードなどの変更が考えられる．

一般的に， VM の構成を変更する場合，開発者はまず開発用の VM 上で構成の変更を行い，動作を確認した上で同じ構成の VM を公開用 VM として立ち上げる．また，新しく VM が起動される時のために，新しいディスクイメージも準備する必要がある．一般に，サービスのダウンタイムの発生を避けるために，従来では以下の手順によりこれを実現していた (図 1)．

(1) 開発用の VM をもとにディスクイメージを作成し，クラウド上のストレージに保存する
(2) 新しいディスクイメージから公開用の VM 群を新たに起動する
(3) ロードバランサの設定を変更し，外部からのトラフィックを新しい VM に向ける
(4) 古い VM 群をシャットダウンする

この手法はネットワークファイルシステムを用いず，同期後は各 VM が必要なファイルを持つためスケーラビリティを確保することが可能である．

しかし，従来の VM 更新手法は時間効率やリソース効率が優れず，小規模・頻繁な変更には特に適さない．数ギガバイトの仮想ディスクからディスクイメージを作成し，ストレージにアップロードするのには最低でも数分の時間を要する．更にディスクイメージから VM を起動するのにも数十秒の時間がかかる．そのため，小規模の設定変更であっても，更新の反映に時間が掛かり，最新のサービスを提供するのが遅れてしまう．また，大きなディスクイメージを送るためには多くのネットワーク帯域を消費し，新しい VM 群を起動してから古い VM 群をシャットダウンするため一時的に倍の計算リソースが必要となる．そのため，頻繁に VM の更新を行う場合，サービスの使用料金が高くなってしまう．

本論文では，開発用 VM に対する変更を公開用 VM に効率的に反映する手法を提案する．提案手法では，更新されたファイル (差分) のみを稼働中の各公開用 VM に送信し，適用するため，従来よりも短時間かつ少ないリソースで変更を反映することが可能である．よって，提案手法は小規模かつ頻繁な変更にも適している．

提案手法では，変更の反映に要する時間を短縮するための仕組みを 3 つ兼ね備えている．第一に，更新されたファイルのみを転送することにより，転送に要する時間を削減する．第二に，稼働中の VM に差分を適用することにより，新たに VM を起動するための時間を省く．第三に，ディスクイメージの作成を非同期的に行うことにより，ディスクイメージの作成完了を待たずに稼働中の VM に変更の反映を行う．

提案手法は次の特徴を持つ．更新されたファイルのみを転送するため，ネットワーク帯域の消費量が少なくなる．また，新たに VM を立ち上げるのではなく稼働中の VM に変更を適用するため，計算リソースも節約しつつ，短時間で最新のサービスを提供することができる．さらに，提案手法は従来手法と同様に，同期後は各公開用 VM は独立して稼働することができるため，従来手法と同等のスケーラビリティを保つことが可能である．

本論文の構成は以下の通りである．2 章では差分転送によるディスク同期の効率性を示す実験について述べる．3 章では提案手法について説明する．4 章では提案機構の実装について説明する．5 章では関連研究について説明する．最後に 6 章で本論文をまとめる．

2. 予 備 実 験

本章では，従来の手法による VM の同期と，更新されたファイルのみ転送して反映する方法それぞれに要する時間を比較する．

表 1 実験に使用した VM の仕様

メモリ	1.7GB
CPU	1 EC2 Compute Unit (1.7 GHz Xeon 相当) 32 bit
ストレージ	15GiB EBS
OS	Linux 2.6.21.7

2.1 既存手法による仮想ディスク同期に要する時間

前述の通り，クラウド環境において VM のソフトウェアやコンテンツに変更を加える場合は，開発用 VM からディスクイメージを作成し，同一構成の VM を公開用 VM として新たに立ち上げるのが一般的である．ここでは，最も普及している IaaS 環境である Amazon EC2 においてこの一連の作業を行うのに掛かる時間を測定する．

Amazon EC2 では， VM を立ち上げる際に，割り当てられるメモリや CPU のコア数などが異なる数種類の VM 仕様 (instance type) から一つ選ぶ仕組みになっている．今回の実験に用いる VM は， 開発用 VM ，公開用 VM ともに m1.small と呼ばれるデフォルトの instance type であり，メモリや CPU の仕様は表 1 に示す通りである．尚，ディスクは EC2 のインスタンスから利用可能な仮想ストレージである Amazon Elastic Block Store (EBS) を用いた．EBS はディスクイメージ (EBS snapshot) の作成が可能な仮想ディスクであり，ディスクイメージの作成の際には前回作成したディスクイメージから更新されたブロックのみが保存される．よって，仮想ディスクへの書き込みが多ければ多いほど，ディスクイメージの作成は時間がかかる．

実験では，予め EC2 で用意されているディスクイメージから開発用 VM を立ち上げた状態で，以下の手順を実施した．

(1) 開発用 VM を停止する．(Stop)
(2) 開発用 VM にマウントした仮想ディスクを基にディスクイメージ (EBS Snapshot) を作成する． (Bundle)
(3) 作成したディスクイメージから，開発用 VM を起動する． (Boot)

尚，Amazon EBS では，VM を停止しなくてもディスクイメージを作成する事が可能である．但し，その場合はスナップショットの作成だけで 20 分以上の時間が掛かるため，今回はまず開発用 VM を停止してからディスクイメージを作成した．

上述の各手順に要した時間を測定し，10 回繰り返した．結果を表 2 に示す．平均で 250 秒以上の時間が掛かっていることが分かる．

表 2 Amazon EC2 におけるクローン VM の作成に要する時間 (秒)

Trial	Stop	Bundle	Boot	合計
1	45.31924	169.64303	41.24922	256.21149
2	51.24586	163.36667	32.71933	247.33185
3	57.82928	172.90110	24.33702	255.06739
4	44.80970	171.05308	30.57223	246.43501
5	43.36735	170.94332	22.65147	236.96215
6	48.54613	174.87581	33.24064	256.66259
7	50.37317	171.70274	31.08614	253.16204
8	47.51697	170.48652	31.58749	249.59098
9	48.00844	174.03765	48.19642	270.24251
10	48.69110	155.91714	30.47444	235.08267
平均	48.57072	169.49271	32.61144	250.67487
最小	43.36735	155.91714	22.65147	235.08267
最大	57.82928	174.87581	48.19642	270.24251
分散	16.59558	32.61217	55.29753	104.04028

今回はディスクイメージから VM 起動し，直後に VM を停止しただけであり，ディスク内容に明示的に変更を加えてはいない．よって，今回の実験結果は VM の同期にかかる時間のベースラインを示すものであると言える．ソフトウェアの変更やコンテンツの更新を行った場合には，今回よりも多くの時間が掛かると考えられる．

2.2 更新されたファイルのみの転送による同期

この実験では， Apache の設定ファイル (/etc/httpd/conf/httpd.conf) 及び MySQL サーバの実行ファイル (/usr/libexec/mysqld) が更新されたというシナリオで，稼働中の公開マシンへのファイルの転送と変更の反映のための処理によってかかる時間を調べる．仮装マシンの仕様は前回の実験と同様であるが，新たに Apache 及び MySQL をインストール・稼働させた．

実験は，予め開発用の VM と公開用の VM を 1 台ずつ立ち上げた状態で行った．実験手順は以下の通りである．

(1) 変更を加えたファイルを開発用 VM から公開用 VM の一時ディレクトリへ SCP で転送する．(Transfer) 尚，転送するファイルの総容量は 6772 KB である．
(2) 公開用 VM において変更を反映させるために，以下の操作を行う (Enhancement)
  (a) MySQL サーバを停止する
  (b) ファイルを一時ディレクトリから本ディレクトリに移動する

表 3　ファイルの転送及びプロセス操作による変更の反映 (秒)

Trial	Transfer	Enhancement	合計
1	1.113	4.692	5.805
2	1.045	4.773	5.818
3	0.938	5.663	6.601
4	0.998	5.665	6.663
5	0.999	4.733	5.732
6	0.895	4.717	5.612
7	1.022	4.636	5.658
8	0.923	4.712	5.635
9	0.955	4.618	5.573
10	0.964	4.720	5.684
平均	0.985	4.893	5.878
最小	0.895	4.618	5.573
最大	1.113	5.665	6.663
分散	0.004	0.167	0.164

(c)　MySQL サーバを起動する
(d)　Apache を再起動する

実験の結果を表 3 に示す．平均で 6 秒弱で転送及び反映処理が完了出来ることが分かる．変更するファイルの数や容量，再起動等の操作が必要となるプロセスの数と種類によって所要時間は大きく異なるが，今回のケースではディスクイメージを作成する方法に比べて約 2.3% の時間しか掛からなかった．転送する必要のあるファイルと，公開用 VM において必要となるプロセス操作を自動的に判断することができれば，従来手法と同様に一元的な操作で開発用 VM における変更を公開用 VM に反映させることができ，かつ劇的に時間が短縮できる見込みがある．

3. 提 案 手 法

本論文の提案手法では，更新されたファイル (差分) のみを転送し，稼働中の VM に差分を適用する事により，従来よりも短時間かつ少ないリソースで変更を反映できるようにする．設定変更に要するコストを劇的に削減することで，小規模・頻繁な設定変更を促し，VM を常に最新の状態に保ちサービスを提供することができる．

3.1 提案機構の全体像

提案機構の動作概要を図 2 に示す．提案機構は開発用 VM におけるディスクへの書き込みを監視し，前回の同期から更新されたファイルを記録する．開発者の指示によって同期が開始されると，前回の同期から更新されたファイルを仮想ディスクイメージレポジトリに送信する．

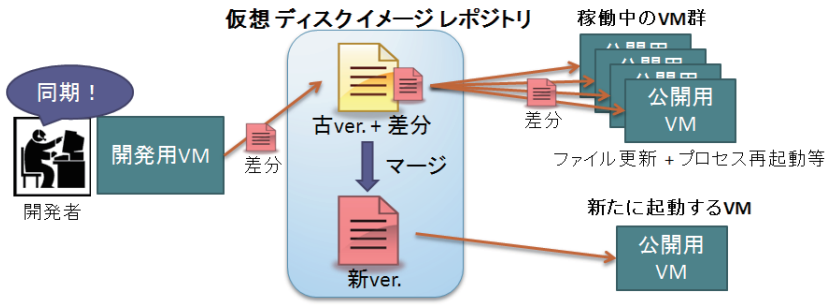

図 2　提案手法

仮想ディスクイメージレポジトリは更新されたファイルを各公開用 VM に転送し，並行して新しい仮想ディスクイメージファイルの生成を行う．

稼働中の各公開用 VM では，受信したファイルをそれぞれの仮想ディスクに適用する．その際，変更の反映に必要なプロセスの再起動などの処理を自動的に行う．

3.2 公開用 VM におけるファイル更新

公開用 VM でファイルを更新する際に，変更の適用のための処理が必要なファイルは以下の 2 種類に分類される．

1 つめは，プロセスを停止してからファイルを更新する必要があるものである．プログラムの実行ファイルなどがこれに該当する．提案機構は，更新するファイルが稼働中のプロセスの実行ファイルであるか，もしくは実行中のプロセスのアドレス空間にマッピングされている場合はファイル更新前にプロセスを停止し，ファイル更新後に再度プロセスを起動する．

2 つめは，ファイル更新後にプロセスの再起動を必要とするものである．プログラムの設定ファイルなどがこれに該当する．更新するファイルが稼働中のプロセスによって過去に開かれたファイルである場合は，ファイル更新後にプロセスの再起動を行う．

これらの動作は，必ずしも全てのファイルに適したものであるとは限らない．例えば，HTML ファイルなどの Web コンテンツは HTTP サーバによって読み込まれるものであるが，HTML ファイルを更新した後に HTTP サーバプロセスを再起動する必要はない．このような例外的な動作は，あらかじめ設定ファイルに記述しておくことにより対応する．

4. 実　　装

提案機構の実装を Linux 2.6.27.7 上で行った．大半はユーザモードプログラムとして実現されるが，一部の機能の実現にはカーネルの改変が伴う．

4.1 開発用 VM

開発用 VM では，前回の同期からどのファイルが更新されたかを記録するために，Linux のファイルシステムイベント監視機能である inotify を用いる．inotify はファイルやディレクトリを監視する事に使える．ディレクトリを関しした場合，そのディレクトリ自身及びディレクトリ内のファイルのイベントを返す．提案機構では，ファイルシステム上の全てのディレクトリを監視する．

inotify を用いて監視を行うと，アクセスされたファイルのパス名とアクセスの種類を取得する事ができる．アクセスの種類とは，オープン，読み込み，書き込み，削除，新規作成などである．提案機構では，これらのアクセスのうち，ディスクに変更を加えるものだけを記録する．具体的には，作成，削除，移動，書き込み，属性変更が該当する．提案機構では，更新されたファイルのパスと，更新内容をローカルのデータベースに保存する．

同期は開発者の指示によって開始される．提案機構は，前回更新時から更新されたファイルのうち，新たに作成されたファイルや書き込みのあったファイルのパスと内容，削除されたファイルのパスを仮想ディスクイメージレポジトリに送信する．

4.2 仮想ディスクイメージレポジトリ

ディスクイメージレポジトリでは，開発用 VM から更新情報を受信すると，それらを各公開用 VM へと転送する．並行して，新しいディスクイメージの作成を行う．仮想ディスクイメージレポジトリ VM では，一つ前のバージョンの仮想ディスクをマウントした状態であり，そこにファイルの更新を行う．ファイルの更新が完了すると，仮想ディスクの新しいイメージファイルを作成する．

4.3 公開用 VM

公開用 VM では，ファイルの更新に際して必要となる処理を，表 4 に示す通り判定する．判定を行うために，現在実行中のプロセスの実行ファイル，mmap() されているファイル，開かれているファイルを procfs から取得する．プロセスが過去に開いたファイルの一覧に関しては，標準の Linux カーネルは保持しておらず取得できないため，カーネルに改変を加える．具体的には，各プロセスが過去に開いたファイルを記録するためのリストをプロセス管理構造体に追加し，その内容を procfs 上で確認できるようにする．プロセス

表 4　ファイルの種類の判別方法及び必要な処理

ファイルの種類	判定情報源	ファイル更新前の処理	ファイル更新後の処理
稼働中のプロセスの実行ファイル	/proc/[PID]/exe	プロセス停止	プロセス起動
稼働中のプロセスに mmap されているファイル	/proc/[PID]/maps	プロセス停止	プロセス起動
稼働中のプロセスが現在開いているファイル	/proc/[PID]/fd	プロセス停止	プロセス起動
稼働中のプロセスが過去に開いたファイル	/proc/[PID]/openlog (新規実装)	-	プロセス再起動

が open() システムコールによってファイルを開く都度，このリストにファイルの i ノード番号を追加する．fork() が呼ばれた場合は親プロセスから子プロセスへリストの内容をコピーし，execve() が呼ばれた際には，リストの内容を空にする．但し，execve() が呼ばれた時点でプロセスが開いているファイルのうち，close-on-exec フラグがセットされていないファイルに関しては，リストに残す．

終了または再起動するべきプロセスを全て調べてから，実際にプロセスの終了等の操作を行う．プロセスの終了は次の通り行う．まず，終了する全てのプロセスの情報を取得して保管する．取得する情報を表 5 に示す．次に，終了する各プロセスの親子関係を調査し，子プロセスよりも親プロセスが先に終了させるように順序の並び替えを行った上で，一定間隔ごとに順々に SIGTERM シグナルを送信する．SIGTERM シグナルを送る時点ですでに終了していたプロセスに関しては，親プロセスによって終了させられたと判断し，後のプロセス起動時にも起動処理を行わない．全プロセスに SIGTERM シグナルを送った後，一定時間経過しても終了していないプロセスに関しては，SIGKILL シグナルによって強制的に終了する．

ファイルの更新が終了すると，終了したプロセスを改めて起動する．その際，表 5 に示したプロセスの各主状態を復元する．

プロセスの再起動は，SIGHUP シグナルを送った上で当該プロセスが更新されたファイルを再度読み込むか否かを inotify によって監視する．尚，inotify ではどのファイルにどのようなアクセスが発生したかを検知する事はできるものの，どのプロセスによるアクセスかは分からない．そのため，提案手法では，ファイルにアクセスしたプロセスの PID を取得できるように inotify に改良を加える．一定時間内に更新されたファイルの読み込みを行わなかった場合は，SIGHUP シグナルが適切に処理されなかったと見なし，プロセスの終了

表 5 プロセスの属性情報の取得と復元

プロセスの属性情報	取得方法	復元方法
起動時のワーキングディレクトリ	/proc/[PID]/launchDir (新規実装)	chdir()
起動コマンドライン	/proc/[PID]/cmdline	execve()
実行ユーザ / グループ	/proc/[PID]/status	setresuid(), setresgid()
環境変数	/proc/[PID]/environ	execve()
ルートディレクトリ	/proc/[PID]/root	chroot()

と起動を続けて行う．

5. 関 連 研 究

仮想化環境用のファイルシステムとして Ventata[5] が挙げられる．Ventana では複数の VM が同じディレクトリをマウントすることができるため，ファイルの更新を各 VM に反映させることは短時間で実現できる．Parallax[6] は仮想化されたクラスタ環境を想定した仮想ディスクである．仮想ディスクのクローンを短時間に作成することが可能である．Ventana, Parallax のどちらもファイルの更新に際してディスクイメージの転送や新たな VM の起動を行う必要がないため，反映時間を短縮し，リソースの消費を抑えることが可能である．しかし，両者とも全 VM が同一の中央サーバにアクセスする設計であるため，スケーラビリティに欠ける．また，クラウド環境では各 VM が地理的に離散している可能性があるため，サーバと通信するためのレイテンシが大きくなり，サービスの低下を招く可能性もある．

SnowFlock[7] は 1 秒以下の短時間で VM のクローンを作成するものである．VM に変更を加える度にクローンを生成し直す必要があるためリソース効率が良いとは言い難い．また，SnowFlock では専用のライブラリを用いてアプリケーションを作成する必要があり，既存のアプリケーションをそのまま用いることはできない．

6. お わ り に

本論文では，差分転送と VM イメージの非同期作成により，短時間かつ省リソースで VM を更新する手法を提案した．従来は仮想ディスク全体のイメージを転送していたのに対し，提案手法は，開発用 VM において更新されたファイルのみを，稼働中の各公開用 VM に転送する．これにより，転送に要する時間と消費するネットワーク帯域を削減することができる．また，新たに公開用 VM を起動するのではなく，稼働中の VM に変更を適用することにより，新たに VM を起動する時間とコストを回避することが可能だる．さらに，時間の掛かるディスクイメージの生成を非同期的に行うことにより，更新の反映に掛かる時間を極限まで短縮し，最新のサービスを遅滞なく提供することが可能である．

今後の課題としては，提案機構をオープンソースの IaaS プラットフォームである Eucalyptus[8] 上で評価すること，VM に対する変更の大小や影響を受けるプロセスの種類による提案機構の効率の違いを実験による確かめること，さらなる効率化のために rsync 等の差分符号化の導入を検討する事などが挙げられる．

参 考 文 献

1) Amazon.com, Inc.: Amazon Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2/.
2) Microsoft Corporation: Windows Live Hotmail. http://mail.live.com/.
3) Google Inc.: Google Calendar. http://www.google.com/calendar/.
4) Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. and Zaharia, M.: Above the Clouds: A Berkeley View of Cloud Computing, *UC Berkeley Reliable Adaptive Distributed Systems Laboratory* (2009).
5) Pfaff, B., Garfinkel, T. and Rosenblum, M.: Virtualization Aware File Systems: Getting Beyond the Limitations of Virtual Disks, *Proceedings of Symposium on Networked Systems Design & Implementation* (2006).
6) Meyer, D., Aggarwal, G., Cully, B., Lefebvre, G., Feeley, M., Hutchinson, N. and Warfield, A.: Parallax: Virtual Disks for Virtual Machines, *Proceedings of ACM European conference on Computer systems* (2008).
7) Lagar-Cavilla, A., Witney, J., Scannell, A., Patchin, P., Rumble, S., Lara, E., Brudno, M. and Satyanarayanan, M.: SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing, *Proceedings of ACM European conference on Computer systems* (2009).
8) Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L. and Zagorodnov, D.: The Eucalyptus Open-source Cloud-computing System, *Proceedings of IEEE International Symposium on Cluster Computing and the Grid* (2009).