Vol.2017-IOT-38 No.16
2017/6/24

使用済みパソコンを再利用したストレージの構築に関する検討

川戸 聡也[1,a)]　本村 真一[1,b)]　東野 正幸[1,c)]　川村 尚生[2,d)]

概要：近年，情報システムの扱うデータ量は増え続けており，データを保存するストレージには，安価や大容量，高可用性などの要件が求められている．従来は専用のストレージ製品を用いることが多かったが，費用が高くなるという問題があり，安価なストレージを適切な用途に活用するなどの様々な対策が講じられている．本研究ではそれらの対策の 1 つとして，不要となった使用済みパソコンをストレージとして再利用することを提案するとともに，使用済みパソコンの性質を活かした実用方法を検討する．検討の結果，使用済みパソコンの既存のディスクを大容量のものに換装して容量を確保した上で，使用済みパソコンを数多く利用した安価で可用性の高い分散ストレージシステムとすることを実用方法として提案した．また，分散ストレージの種類をオブジェクトストレージとし，使用済みパソコンに対してオブジェクトストレージとして利用するための環境を自動的に構築するシステムを試作し，少ない労力で使用済みパソコンを分散ストレージとして利用できることを確認した．

Construction of Storage Reusing Used Personal Computers

Toshiya Kawato[1,a)]　Shin-ichi Motomura[1,b)]　Masayuki Higashino[1,c)]　Takao Kawamura[2,d)]

1. はじめに

近年，情報システムの扱うデータ量は増え続けており，データを保存するために必要不可欠な存在であるストレージには，安価や大容量，高可用性などの要件が求められている．従来は専用のストレージ製品を用いることが多かったが，機能性や信頼性に優れるために費用が高くなるという問題があった．その対策として，一律に専用のストレージ製品を利用するのではなく，費用対効果などを踏まえた上で用途に応じた適切なストレージを選択することが挙げられる．また，その逆に，安価なストレージを利用することを前提とし，安価なストレージを有効に活用可能な実用方法を検討することも対策として挙げられる．

本研究ではそれらの対策の 1 つとして，不要となった使用済みパソコンをストレージとして再利用することを提案するとともに，使用済みパソコンの性質を活かした実用方法を検討する．使用済みパソコンとは，本来は継続して利用可能であるがリプレースなどにより用途のなくなったパソコンを指す．使用済みパソコンは既存の機器であり，それ自体の再利用において費用は掛からないため，周辺機器を含めても安価に導入することができる．また，パソコンの総台数が多く更新が頻繁に行われる大学などの環境では，使用済みパソコンも数多く存在し，それらを回収して利用することができる．更に，使用済みパソコンは利用可能な状態であっても廃棄されることが多く，分解してリサイクルすることは広く普及している [1] が，そのままの形で情報システムに組み込んで再利用する試みは少ない．そのままの形で再利用することで，更なる資源の有効活用や環境の保全にも繋げられると期待できる．

本稿では，使用済みパソコンの性質を考慮したストレー

[1] 鳥取大学 総合メディア基盤センター
Center for Information Infrastructure & Multimedia, Tottori University
[2] 鳥取大学大学院 工学研究科 情報エレクトロニクス専攻
Department of Information and Electronics, Graduate School of Engineering, Tottori University
[a)] t.kawato@tottori-u.ac.jp
[b)] motomura@tottori-u.ac.jp
[c)] higashino@tottori-u.ac.jp
[d)] kawamura@ike.tottori-u.ac.jp

ジとして，使用済みパソコンの既存のディスクを大容量のものに換装して容量を確保した上で，使用済みパソコンを数多く利用した安価で可用性の高い分散ストレージシステムとすることを提案する．また，分散ストレージの種類をオブジェクトストレージとし，使用済みパソコンに対してオブジェクトストレージとして利用するための環境（以下，「オブジェクトストレージ環境」という）を自動的に構築するシステムの試作により，少ない労力で使用済みパソコンをオブジェクトストレージとして利用できることを確認する．

2. 使用済みパソコンの性質を考慮したストレージシステム

使用済みパソコンをストレージとして再利用するにあたり，他の方法に対して利点のあるストレージシステムとする必要がある．すなわち，再利用しない場合と比べて利点よりも欠点が勝るようであれば，使用済みパソコンはストレージとしての利用には適さないということになる．また，使用済みパソコンは専用のストレージ製品やサーバに比べて性能が劣るため，用途によりストレージとして相応しくないと判断されることが考えられる．このため，使用済みパソコンをストレージとして利用する上で，利点があり適切な用途に利用可能な実用方法を検討する必要がある．ここでは，使用済みパソコンの性質を考慮した実用方法を検討する．

使用済みパソコンの性質として，まずは個人用であるために容量が小さいことが挙げられる．最近のパソコンではTB のディスクを搭載することも珍しくなくなったが，現時点で回収可能な使用済みパソコンの容量は 1 台あたり数百 GB 程度だと考えられる．このため，TB 台の大容量とするためには使用済みパソコンの台数を数多く確保する必要があるが，そのままの状態では非効率であり設置場所や消費電力の面で現実的ではない．そこで，使用済みパソコンの既存のディスクを大容量のものに換装して容量を確保した上でストレージとして利用することとする．換装するためのディスクに費用が発生することとなるが，一般的なパソコン用のディスクは専用のストレージ製品やサーバ向けのディスクに比べて安価であり，全体として安価にストレージを構築することができる．また，使用済みパソコンは利用の程度にばらつきがあり利用においては耐用年数を考慮する必要があるが，最も故障頻度の高いディスクを新品とすることで障害が発生するリスクの軽減を図ることができる．

次に，数多くの台数を利用できることも使用済みパソコンの性質として挙げられる．使用済みパソコン自体には費用が発生しないため，設置場所の許す限り数多くの台数を利用することができる．数多くの台数を利用する場合はディスクが分散して配置されることとなり，分散されたディスクに対してデータを分散配置や冗長配置することができれば高い可用性を安価に実現できる．また，使用済みパソコンはラックマウント型サーバなどに比べて設置する場所に制限が少ないため物理的にも分散して配置しやすく，電源とネットワーク環境さえあれば場所を問わず利用することが可能である．そこで，ストレージの種類はネットワーク上で分散されたディスクを連結して利用する構成である分散ストレージとする．使用済みパソコンは，数多くの台数を確保できることから運用途中のシステムへの追加や削除が容易であることが望ましいが，分散ストレージであれば柔軟に対応可能である．

更に，専用のストレージ製品やサーバに比べて性能が劣ることも使用済みパソコンの性質として挙げられる．使用済みパソコンはサーバ用途ではなくディスクもパソコン用のものであるため，高速な通信や高負荷な処理を求めることは難しい．このため，適切な用途としては，利用や更新の頻度が低いデータを長期間保存するためのバックアップ用のストレージなどとして利用することが考えられる．

加えて，耐用年数が様々であることも使用済みパソコンの性質として挙げられる．使用済みパソコンは回収されるまでの利用の形態や期間が様々であるため，回収後にいつまで利用できるのかといった耐用年数がばらばらで且つ判断しづらい．このため，使用済みパソコンの状態を監視することによる耐用年数の評価や障害発生時の自動切り離しといった障害に備えた対応を行う必要がある．

最後に，使用済みであることに限らないが，様々な性能や構成の機種が混在することも使用済みパソコンの性質として挙げられる．パソコンは様々なメーカが様々な機種を販売しているため，サーバに比べて多様性が高い．このようなパソコンをストレージとして利用するために 1 台 1 台設定すると構築に多大な労力を要することが予想される．このため，性能や構成の異なる使用済みパソコンが多数混在する場合でも，ストレージとして利用するための環境を自動的に構築可能な手法が必要である．

以上より，使用済みパソコンの性質を考慮したストレージとして，使用済みパソコンの既存のディスクを大容量のものに換装して容量を確保した，数多くの台数を利用することによる安価で可用性の高い分散ストレージシステムを提案する．主な用途としては，使用済みパソコンの低い性能でも処理可能な，利用や更新の頻度が低いデータ用のストレージを想定する．ここで，提案したストレージシステムを構築するにあたり，まずは使用済みパソコンに対して分散ストレージとして利用するための環境（以下，「分散ストレージ環境」という）を自動的に構築するシステムを検討する．

3. 使用済みパソコンに対する分散ストレージ環境の自動構築

使用済みパソコンに対して分散ストレージ環境を自動的に構築するシステムについて述べる．

まず，分散ストレージの種類としてはオブジェクトストレージ [2], [3] とする．オブジェクトストレージとは，データをオブジェクトという単位で管理するストレージであり，利用や更新の頻度が少ないデータの保存に向く．Amazon S3[4] などのオンラインストレージで広く利用されており，大学内においても教育・研究用情報システムなどで活用に取り組まれている [5]．データへのアクセスには Representational State Transfer（REST）に則った HTTP プロトコルを利用する．OS に依存することなく Web アプリケーションのように利用可能だが，ファイルシステムに比べて利用する側が未対応であることが多く，ファイルシステムを利用するファイルストレージに比べて汎用性に欠ける．この汎用性の問題については，s3ql[6] などのオブジェクトストレージをマウントして利用可能なゲートウェイと組み合わせて利用することで対応可能である．オブジェクトは，データの保存場所を示す固有の識別子と様々な情報を記録可能なメタデータを付与した上で，階層構造ではない平坦な空間に格納される．固有の識別子が実際に保存されるディスク上の場所に依存しないことでデータの配置に制約が少ない．このため，データの移動や分散配置が容易であり，遠隔地への複製保存やディスクの追加によるスケールアウトを実現しやすい．

また，使用済みパソコンへ OS やオブジェクトストレージを構築するためのソフトウェアを自動的にインストールして設定する仕組みが必要となる．可能な限り手作業を排除して自動化するためには，ネットワーク経由で一連の設定が完了し，中央サーバにて管理できることが望ましい．そこで，OS はネットワークブート，ソフトウェアについてはインフラストラクチャー構築の自動化ツールを用いることで，インストールや設定を自動化する．ネットワークブートは，パソコンなどのクライアントに対してネットワーク経由でプログラムを実行可能な仕組みである．また，インフラストラクチャー構築の自動化ツールは，あるソフトウェアが動作するための基盤部分の構築を自動化可能なツールであり，ソフトウェアが動作する環境の構築や管理に掛かる労力を軽減するために用いられ，広く普及しつつある．

4. 使用済みパソコンに対する分散ストレージ環境の自動構築システムの試作

使用済みパソコンに対してオブジェクトストレージ環境を自動的に構築するシステムを試作した．

4.1 Swift によるオブジェクトストレージ

オブジェクトストレージを構築するためのソフトウェアとして，オープンソースで利用可能なものであれば OpenStack Swift[7]（以下，「Swift」という）や Ceph[8] があり，今回は Swift を利用した．

Swift とは，クラウド基盤を構築するオープンソースソフトウェアである OpenStack のうち，オブジェクトストレージを実現するコンポーネントである．Openstack には多くのコンポーネントがあり，連携することでクラウド基盤を構築可能だが，オブジェクトストレージの構築を目的とした Swift 単体での利用も多い [9], [10]．Swift の基本構成を図 1 に示す．Swift では，オブジェクトはコンテナで一覧を管理され，コンテナはアカウントにて一覧を管理される．オブジェクトを格納する Object Server，コンテナとアカウントの役割を担う Container Server と Account Server はまとめてストレージノードと呼ばれる．ストレージノードはゾーンという単位でグループ化され，ゾーンは複数作成することができる．それぞれのゾーンには同じオブジェクト，コンテナ，アカウントが格納されるため，1 つのゾーンに障害が発生した場合でもそれ以外にアクセス可能なゾーンが存在することで，冗長化や耐障害性の向上が実現できる．実効容量を求めるのであればゾーン数を少なくし，高い可用性を求めるのであればゾーン数を多くするという対応になる．また，ゾーンへのストレージの追加やゾーン自体の追加をすることで全体の容量を容易に拡張できる．加えて，稼働しているストレージノードの情報やオブジェクトの配置方法などは ring と呼ばれるファイルに設定され，ストレージノードの全てのサーバ上に配置されることで単一障害点の排除などを図っている．ストレージノードとクライアントとの通信は Auth Server で認証が行われた後，プロキシノードと呼ばれる Proxy Server が中継する．Proxy Server はクライアントからのオブジェクトのアップロードや削除などの要求を PUT や DELETE などの HTTP プロトコルで受け取り，ストレージノードにて要求された処理を実行する．プロキシノードについても複数作成することができ，高い可用性とパフォーマンスの向上を実現できる．

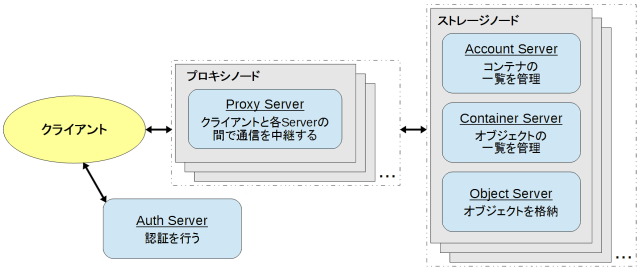

図 1　Swift の基本構成

4.2 OS の自動インストール

使用済みパソコンへの OS の自動インストールには Kickstart[11] と PXE ブート [12] を利用した．Kickstart とは，Red Hat 社が提供している Red Hat 系 OS の自動インストールツールである．OS インストール時の設定項目やインストール後に実行したいスクリプトなどを予めファイルに記述して読み込ませることで，インストール時に必須となるパーティション設定などの様々な入力や選択，OS インストール後のスクリプト実行といった作業を自動的に行うことができ，ネットワークブート時にも利用可能である．PXE ブートは，Intel 社が開発した Preboot eXecution Environment（PXE）を利用したネットワークブートの仕組みであり，PXE ブートすることでネットワーク経由での OS の起動やインストールが可能となる．Kickstart と PXE ブートを組み合わせることで，使用済みパソコンをネットワークに接続して PXE ブートさせる操作のみで，OS のインストールや初期設定を自動的に完了させることができる．

ここで，PXE ブートさせるためにはクライアントとなる使用済みパソコンに IP アドレスを払い出す必要がある．管理上，個々の使用済みパソコンを識別可能な管理方法が望ましい．今回は固定 IP アドレスを設定することとした．ここで，DHCP サーバで MAC アドレスに対応した固定 IP アドレスを払い出す設定を行う必要があるが，接続するパソコンが多くなることを考えると手動で設定するのではなく，可能な限り自動化するべきである．このため，固定 IP アドレスを払い出す設定をスクリプトにより自動化することとした．パソコンをネットワークに接続して PXE ブートを行うと，DHCP サーバのログに DHCP DISCOVER が記録されることを利用し，ログを常に監視して DHCP DISCOVER が現れると対応する MAC アドレスを抽出する．この MAC アドレスが DHCP の設定ファイルになければ設定を追記し，設定ファイルを再読み込みするという流れである．

Kickstart と PXE ブートにより，OS として CentOS 6 をインストール可能な PXE サーバを仮想マシンにて構築した．PXE サーバは，DHCP サーバ，TFTP サーバ，HTTP サーバにより構成される．OS の自動インストールの流れを図 2 に示す．クライアントからネットワークブートの要求が行われると，DHCP サーバが前述のスクリプトにより固定 IP アドレスをクライアントに払い出す．その後，クライアントが TFTP サーバからブートイメージを読み込み，OS のイメージや Kickstart ファイルが読み込まれることで OS の自動インストールが実行される．なお，Kickstart ファイルなどをクライアントから読み込むことが可能な場所に配置するため，HTTP サーバを構築した．

Kickstart ファイルでは，OS インストール時に必要となる基本的な設定を中心に記述した．主な設定項目を以下に示す．

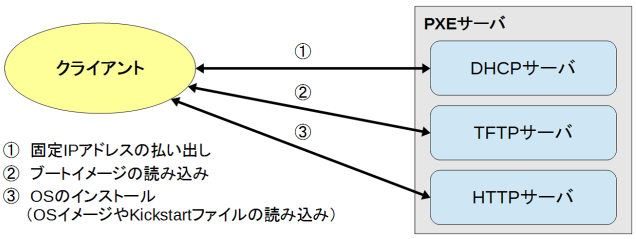

図 2 Kickstart と PXE ブートによる OS のインストール

- root パスワードの設定
- ディスクのパーティションの設定
- インストールするパッケージの選択
- 外部スクリプトの実行

外部スクリプトは，Kickstart ファイルでは対応しにくい処理をまとめて実行するように設定したものである．主な設定項目を以下に示す．

- 外部ネットワーク接続用プロキシの設定
- hosts ファイルの設定（SSH 接続の許可）
- 公開鍵認証による SSH 接続の設定

4.3 Swift の自動インストール

Swift の自動インストールに必要なインフラストラクチャー構築の自動化ツールとして，オープンソースで利用可能なものであれば Chef[13] や Puppet[14]，Ansble[15] がある．これらは，プログラミングのようにコードを記述することでインフラストラクチャーの構築が可能であり，構築手順を記述するのではなく最終的なあるべき状態を定義するため，複数回実行しても同じ結果となる冪等性を持つという特徴がある．今回は Chef を利用した．

Chef とは，Chef Software を中心に開発されているインフラストラクチャー構築の自動化ツールである． 基本的な利用方法としては，Cookbook と呼ばれるフォルダの中に Recipe と呼ばれる設定内容を記述するファイルを作成して実行する形である．また，Cookbook には Recipe に加え，各種設定ファイルを作成するための雛型となる Template や，処理対象の環境に応じて値を変更可能な変数を定義するための Attribute，処理対象に配置したいファイルを格納するための files フォルダなどが存在する．Recipe に加え，必要に応じてこれらを活用することで，処理対象の最終的なあるべき状態を定義することができる．

Chef によるインストールの流れを図 3 に示す．まず，前述の PXE サーバと同一の仮想マシン上に，hef 本体やクライアントでサーバ上の Recipe を実行可能な knife-solo をインストールした Chef サーバを構築した．次に，Swift 用の Cookbook を作成し，Swift のインストールに必要な設定を記述した Recipe や Swift の各種設定ファイルの雛型となる Template などを作成することで，knife-solo の実行

によりクライアントに Swift によるオブジェクトストレージ環境が構築されるようにした．

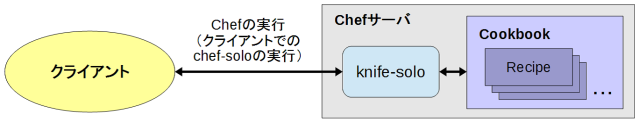

図 3　Chef によるインストール

ここで，プロキシノードは既存のサーバを利用することとしたため，ストレージノード部分のみを構築するように Recipe を作成した．knife-solo の実行も OS インストール時のスクリプト内で行うことで OS のインストールから Swift のインストールや設定まで完全に自動化することも可能だが，動作確認を自動構築の段階ごとに行うためにこの動作は手動で実行することとした．なお，実際にプロキシノードと通信を行うためには ring ファイルを作成して各クライアントに配置する必要があるが，今回は ring ファイルの自動化までは実施していない．Chef にて設定した主な項目を以下に挙げる．

- Swift に関する各種パッケージのインストール
- インストール先のディスク数に対応したディスクパーティションの作成
- Template を利用した各種設定ファイルの設定

4.4　実験

試作した使用済みパソコンに対してオブジェクトストレージ環境を自動的に構築するシステムにより，使用済みパソコンに対して実際にオブジェクトストレージ環境を構築した．利用した使用済みパソコンの台数と性能は以下の通りである．これらは，実際に機器の更新などにより廃棄する予定であったパソコンを回収したものである．なお，今回は大容量ディスクへの換装は行わず，既存のディスクのままとした．

- 小型デスクトップパソコン（NIC：1GbE，ストレージ：HDD160GB × 1，2008 年製）× 9 台
- ノートパソコン（NIC：100MbE，ストレージ：HDD320GB × 1 つ，2008 年製）× 1 台
- ノートパソコン（NIC：1GbE，ストレージ：HDD640GB × 1 つ，2012 年製）× 1 台
- デスクトップパソコン（NIC：1GbE，ストレージ：HDD160GB × 2 つ，2008 年製）× 1 台

まず，空間を有効に活用するためにメタルラックを設置し，使用済みパソコンを各段に設置した（図 4）．次に，それらを 1GbE のスイッチングハブにより同一ネットワークに接続し，PXE ブートさせた．多くの場合，起動時に特定のキーを押下することで PXE ブート可能だが，標準の設定では PXE ブートできない場合があり，使用済みパソコンの構成に応じて BIOS や UEFI の設定を変更する必要があった．PXE ブート後は，PXE サーバにより OS のインストールと初期設定が，Chef サーバにより Swift のインストールや設定が自動的に完了した．ここで，NIC の性能により処理が完了するまでの所要時間に差が見られ，100MbE のノートパソコンについては明らかに所要時間が長かった．ストレージとして利用する上である程度の通信が発生することも踏まえると，NIC は最低でも 1GbE は必要だと考えられる．

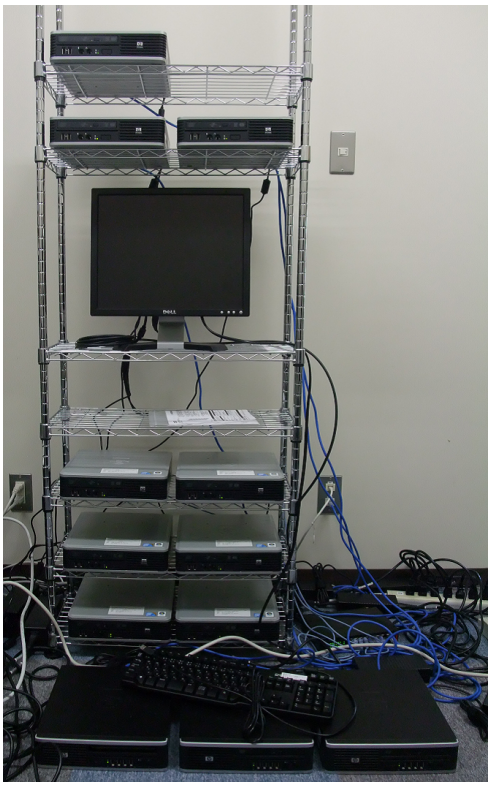

図 4　実験風景

使用済みパソコンを 12 台利用することでディスクの総容量は約 2700GB，今回は Swift のゾーン数を 3 としたために実際にデータ保存に利用可能な実効容量は約 900GB となった．データが分散した配置される可用性の高い構成ではあるが，既存のディスクを利用すると使用済みパソコンの台数に対して容量が小さ過ぎるため，やはり大容量のディスクに換装することで大容量化を図る必要がある．仮に，デスクトップパソコンの既存のディスクを 4TB に，ノートパソコンの既存のディスクを 1TB に交換した場合の実効容量は約 15TB であり，換装に必要な費用は原稿作成時点で約 15 万円である．なお，2.5 インチのディスクは 3.5 インチに比べて割高であるため，安価に大容量を実現する上では 3.5 インチのディスクを搭載可能なデスクトップパソコンが好ましい．ただし，デスクトップパソコンはノートパソコンに比べて大きいため，省スペースで設置可能という面ではノートパソコンが好ましい．用途や設置環境に応じて選択する必要がある．

自動構築システムの試作により，配線や設置などの物理的な作業や PXE ブートの実行などの手動操作はいくつか必要であるが，自動構築により少ない労力で使用済みパソコンを分散ストレージとして利用できることを確認した．

5. おわりに

使用済みパソコンをストレージとして再利用することを提案した．その実用方法として，使用済みパソコンの既存のディスクを大容量のものに換装して容量を確保した上で，数多くの台数を利用した安価で可用性の高い分散ストレージシステムとすることを提案した．また，分散ストレージの種類をオブジェクトストレージとし，使用済みパソコンに対してオブジェクトストレージ環境を自動的に構築するシステムを試作することで，少ない労力で使用済みパソコンを分散ストレージとして利用できることを確認した．

今後の課題として，実際に大容量のディスクに換装した上で使用済みパソコンに対して分散ストレージ環境を構築し実際に運用することが挙げられる．これにより，パソコン用のディスクを分散ストレージとして利用する上での，速度や故障率といった性能の評価や，具体的にどのようなデータをどのように読み書きするのに向いているかといった調査を行う．併せて，電力消費量の測定などを行い，ランニングコストについても検証する．また，実運用にあたっては，今回試作して動作を確認した使用済みパソコンへの分散ストレージ環境の構築を自動化することに加え，運用中の使用済みパソコンにおける状態の監視や障害原因の切り離しといった障害対応も自動化することが必要となる．これらを通して，使用済みパソコンをストレージとして再利用することの有効性を示し，実際に活用していく．

参考文献

[1] 使用済みパソコンの回収実績【平成 27 年度】，入手先 ⟨http://www.pc3r.jp/association/recycle_result.html⟩ (2017.05.31).
[2] M. Factor, K. Meth, D. Naor, O. Rodeh, and J. Satran: *Object storage: the future building block for storage systems*, Local to Global Data Interoperability - Challenges and Technologies, pp. 119-123 (2005).
[3] M. Mesnier, G.R. Ganger, and E. Riedel: *Object-based storage*, IEEE Communications Magazine, Vol. 41, pp. 84-90 (2005).
[4] Amazon Simple Storage Service ドキュメント，入手先 ⟨https://aws.amazon.com/jp/documentation/s3/⟩ (2017.03.29).
[5] 本村真一，川戸聡也，木本雅也：教育・研究用情報システムにおけるオブジェクトストレージの活用，学術情報処理研究，No.19, pp. 26-34 (2015).
[6] nikratio / S3QL - Bitbucket, 入手先 ⟨https://bitbucket.org/nikratio/s3ql/⟩ (2017.05.31).
[7] Welcome to Swift’s documentation!, 入手先 ⟨https://docs.openstack.org/developer/swift/⟩ (2017.05.31).
[8] Ceph Homepage, 入手先 ⟨http://ceph.com/⟩ (2017.05.31).
[9] DDN ジャパン、及び日本 IBM は、Yahoo! JAPAN が運営する日本のデータセンターに OpenStack Swift の最低限必要なキャッシュデータを置き、数十 PB 超のデータを米国データセンターに 50TB/日で保存するアクティブアーカイブシステムを構築，入手先 ⟨https://ddn.co.jp/media/2016/12/07/7⟩ (2017.05.31).
[10] 「ドコモメール」を支える新クラウドストレージを構築，入手先 ⟨http://www.nttdata.com/jp/ja/news/release/2015/011500.html⟩ (2017.05.31).
[11] 第 32 章 キックスタートを使ったインストール，入手先 ⟨https://access.redhat.com/documentation/ja-JP/Red_Hat_Enterprise_Linux/6/html/Installation_Guide/ch-kickstart2.html⟩ (2017.05.31).
[12] Preboot Execution Environment (PXE) Specification, 入手先 ⟨http://www.pix.net/software/pxeboot/archive/pxespec.pdf⟩ (2017.05.31).
[13] Chef - Automate Your Infrastructure, 入手先 ⟨https://www.chef.io/chef/⟩ (2017.05.31).
[14] より優れたソフトウェアへの最短パス — Puppet, 入手先 ⟨https://puppet.com/ja⟩ (2017.05.31).
[15] Ansible is Simple IT Automation, 入手先 ⟨https://www.ansible.com/⟩ (2017.05.31).