オブジェクト・ストレージとは（概説）

履歴

日付	版	内容
2018/2/16	初版	新規作成

目次

１．まえがき

コンピュータプラットフォームの進化に伴い、ストレージシステム及びデータを管理する手法はその用途と共に変化してきました。
データベースを代表とする構造化されたデータを、ブロック単位で管理する「ブロックアクセス」手法/ブロックアクセスストレージから、データが含まれる論理ボリュームをディレクトリやフォルダで階層制御する「ファイルシステム」手法/ファイルアクセスストレージが出現しました。そして画像や動画ファイルを代表とする、大量の非構造化データを「オブジェクト」と言う単位で扱う方法/オブジェクト・ストレージが普及してきている事はご承知のとおりです。
この「オブジェクト」としてデータを扱うストレージは、クラウドサービスにおいて広く利用・普及してきており(Facebook/Google/Amazon などが代表例)、今後そのニーズは様々な用途に広まってゆくと考えられます。

本書では、ストレージ及びストレージ・プロトコルの変遷から「オブジェクト・ストレージ」出現の背景、及び「オブジェクト・ストレージ」と他のストレージの違い、及び「オブジェクト・ストレージ」の利用方法も実例に基づきわかり易く説明することで、利用者から見た「オブジェクト・ストレージ」の長所・短所を解説していきます。

これらの解説により、本書が今後のオブジェクト・ストレージの選定・利用・運用に対するリファレンスとして役立つことを期待します。

２．これまでのストレージの歴史とクライアントから利用の変化

２．１．ストレージの歴史

・１９７０年代　　：メインフレーム接続の外部記憶装置として大型ディスクサブシステムの製品化（１４型交換可能ディスクパック装置など）
・１９８０年代　　：コントローラの高性能化(ディスクキャッシュのサポート)
・１９９０年代前半　　：データ２重書機能による RAID(Redundant Array of Inexpensive Disks)機能（ミラーリング)サポート
主要コンポーネントの冗長化による高信頼化
・１９９０年代後半　　:エラー訂正コード(パリティコード)を活用した RAID 技術普及
・１９９０年代後半　　：オープンシステムへの接続拡大
ストレージ機能の実装(レプリケーション技術など)
・１９９０年代後半　　：SAN(Storage Area Network)の実用化開始
・２０００年台前半　　：Gigabit Ethernet の普及に伴う NAS(Network Attached Storage)需要の拡大
・２０００年台後半　　：仮想化基盤の広がりに伴うストレージの多様化
・２０１０年前半　　：ストレージを含めたクラウドサービスの広がり

現在は、分散型ストレージ等の物理的なストレージ装置を意識しないストレージサービス（クラウド）などが普及し、ストレージにおける用途の変革期となっています。

２．２．クライアントからのストレージ利用の変化

昔（１９９０年前半迄）：
メインフレームなど基幹システムでのストレージ利用が中心
システム専用のイアンターフェースや SCSI や Fibre Channel などのインターフェースを利用したダイレクトアタッチの利用
少し前（２０００年以降）：
オープンシステムによるストレージ利用(オンプレ中心)が中心
ダイレクトアタッチに加えネットワークプロトコル利用による NAS 利用の増加
現在～将来：
モバイル、SNS、IoT、ビッグデータなどのデータを、クラウドサービス等を中心とした新たなプラットフォームでのストレージ利用が増加（オンプレ/オフプレ/ハイブリッドにおける利用）
ダイレクト接続・ネットワーク接続に加え、Web Service による利用が増加

３．なぜ新しいストレージが出てきたのか（課題と解決）

３．１．従来型ストレージの課題

３．１．１．RAID (Redundant Arrays of Independent Disks) の限界

従来型ストレージは、データ保護の手法として RAID を採用しています。RAID はその概念の提唱から既に 30 年近くも経っており、ストレージ技術としてはごくありふれたものとなっています。また、RAID 技術の詳しい解説は本資料の趣旨から外れるので省きますが、RAID レベルとしては「5」または「6」が一般的で、それぞれデータ保護のためのパリティを「1」または「2」含める事によってデータの冗長化を図っています。提唱された当初はライト時のデータ分割とパリティの付加処理、およびリード時のデータ統合処理の負荷が、ストレージコントローラに重く圧し掛かっていましたが、マイクロ・プロセッサの処理能力の向上に伴いその負荷は全く問題にならなくなり、ありふれたものになっていきました。しかしながら、昨今の HDD (Hard Disk Drive) の大容量化に伴い、新たな課題がクローズアップされる事となりました。

その課題とは「リビルド問題」です。HDD はある一定の確率で障害を起こし読めなくなる事が起こり得る宿命を負っており、そもそもその障害からデータを保護するために提唱された技術が RAID です。HDD の障害時には、付加されたパリティによって読めなくなった HDD 上のデータを復元する事によってデータの保全性を保つ事ができるのが RAID 技術のポイントです。また、ストレージ装置内に未使用のスペア HDD が搭載されていれば、そのスペア HDD 上にデータを復元する事もでき、これが「リビルド」です。HDD の大容量化に伴い、この「リビルド」に要する時間が問題としてクローズアップされる事となりました。

一般的に、今から三世代前の 3TB HDD をリビルドするのには概ね 24 時間を要するといわれています。これが最新世代の 8TB HDD になると、単純計算では 64 時間を要する事になるので、パリティを「1」しか保有しない RAID レベル「5」では、その間全くデータが保護されていない極めてリスクの高い状態にデータが晒される事になります。もし、不幸にしてリビルド中にもう一つ HDD が障害を起こすと、もはやデータを復元する事は不可能となりデータロストが発生します。必然的にバックアップからデータを復元しなければならない事になり、かなりの時間システムの停止を余儀なくされる事になりかねません。従って、昨今では RAID レベル「5」ではもはやその脆弱性は許容範囲外であり、レベル「6」は必須とまで言われています。

因みに、リビルドにこれだけ時間を要するのは、失われた HDD の内容を復旧する際に、ストレージコントローラは HDD のどの部分に意味のあるデータが書込まれていたかを判別できないため、常に HDD の全てを復元しようとするためです。

３．１．２．ファイルシステムの抱える課題

従来のストレージ装置においては、データを効率よく格納するための仕組としてファイルシステムが必要となります。ファイルシステムには様々なものがありますが、共通して以下の二つの課題を抱えている事が知られています。

先ず、ファイルシステムでは、データの論理的格納単位として「ボリューム」という概念があり、データを格納するためには先ずボリュームを定義しなければなりません。また、ファイルシステムでは同じ属性を持ったデータの論理的集合を「ファイル」という単位で取扱いますが、同一ファイルはボリュームを跨って格納する事ができないため、アプリケーションにボリュームを割当てる際に一定の非効率性を伴わざるを得ない事が挙げられます。

また、アプリケーションからファイルをアクセスする際に、その利便性のためボリュームを更に「フォルダ」と呼ばれる格納単位に分割し、同じ属性を持つファイルをフォルダごとに分類して取扱う事が一般的ですが、ボリュームに格納されるファイル数が増加するに従ってボリューム内のフォルダ構造が複雑化し、管理面での課題を抱える事が少なくありません。

３．２．課題解決へのアプローチ

これらの課題を解決するために、これまでに様々な工夫がなされて来ました。幾つかの例を挙げます。

部分的リビルド

リビルドの際に、失われた HDD 上の全てを復旧するのではなく、意味のあるデータのみを復旧する事によってリビルドの短縮化を図ります。

固定的な RAID グループの排除

予め HDD を固定的 RAID グループに分けて LUN を構築するのではなく、ファイル単位で分割されたデータとパリティの格納先を変える事によってデータの格納効率の向上を図ると共に、ファイル単位でのデータ復旧によりリビルドの短縮化を図ります。

シン・プロビジョニング

ボリュームの拡張や縮小を動的に行えるようにし、ファイルシステムの持つ非効率性も排除を図ります。

４．オブジェクト・ストレージとは

４．１．オブジェクトとは

未だ業界全体として完全にコンセンサスのとれた定義がある訳ではありませんが、従来ファイルシステムにより論理的にも物理的にも分割して管理されていた、データ（ファイル）の本体とその属性情報であるメタデータ（ディレクトリー構造体）を論理的に一つのセットとしてアプリケーションからアクセスできる様にしたデータの単位を「オブジェクト」と定義するのが一般的となりつつあります。

４．２．オブジェクト・ストレージとは

オブジェクトを直接取扱い格納する事ができるストレージの総称です。その論理的構成から、ディレクトリ構造体を持たないフラットなストレージとなり、データの位置情報を埋め込んだ単一キー「オブジェクト ID」のみで管理される事になります。
そのデータ格納方式からボリュームという概念がなく、ストレージとしての格納容量制限はありませんが、「オブジェクト ID」生成アルゴリズムに依存した、格納できるオブジェクト数の制限はあり、またデータの一部分のみを変更（更新）する事はできません。

４．３．オブジェクト・ストレージの歴史的背景

歴史的にはオブジェクト・ストレージの起源は、データの効率的な保管と保存の実現を目指した Content Addressing Storage（CAS）にありますが、より効率的なオブジェクトへのアクセスを実現するため、様々なオブジェクト ID 生成のアルゴリズムとアーキテクチャが提唱されて来ました。また、データ保護方式としては当初はレプリケーション（複製）が適用されましたが、後により効率の良い Erasure Coding（消失訂正符号）が適用される事になります。

４．４．オブジェクト・ストレージの利点

オブジェクト・ストレージは、歴史的には前述した従来型ストレージの課題を解決する事を目的に提唱されたものではありませんが、そのデータ可能方式の特性から、結果として従来型ストレージの課題を解決する事となりました。
また、オブジェクト・ストレージのデータ格納方式はハードウェアへの依存度が低く、はーウェア更新時のデータ・マイグレーションが不要である事は、システム運用上の極めて大きな利点です。従って、長期保存が求められるデータの格納先や、所謂「ビッグデータ」としてクローズアップされている「非構造化データ」の格納先として注目されています。

４．５．オブジェクト・ストレージの課題

従来型ストレージの課題を解決しているオブジェクト・ストレージですが、そのアーキテクチャに由りアプリケーションからのアクセス方式（API-Application Program Interface）は、http ベースの REST を使用しており、従来型ストレージとは全く互換性がありません。

従来型ストレージの典型である NAS（Network Attached Storage）との互換性を実現するゲートウェイが製品化されてはいますが、この部分がボトルネックとなり、オブジェクト・ストレージ本来の利点を生かせなくなる場合が少なくなく、未だ課題は多いです。

また、昨今の潮流から SDS（Software Defined Storage）としてインプリされているものが主流で、ストレージ要件に基づくハードウェア・プラットフォームのサイジングも、各ベンダーから明確なガイドラインが提示されているとは必ずしも限らず課題の一つです。

４．６．クラウドストレージとの相違点

そもそもクラウドストレージとは、ネットワーク（インターネット）を経由し、複数のユーザまたはアプリケーションが共用する IaaS としてのストレージ・インフラの総称でありますが、オブジェクト・ストレージはファシリティなのでそもそもレイヤが異なる概念です。但し、クラウドストレージを構築するためにはファシリティとしてのストレージが必要であり、当初は従来型ストレージの典型である NAS が適用されて来たものの、昨今そのコスト・パフォーマンス、シン・プロビジョニング性、およびスケールアウト性の故に、オブジェクト・ストレージが有力視され注目されて来ています。

そのため、オブジェクト・ストレージはクラウドストレージと同一視される事が少なくなく、また意図的に同一視するベンダーもありますが、あくまでレイヤの異なる概念なので明確に区別をして欲しいです。

４．７．オブジェクト・ストレージの分類と類型化

実際に実現されているオブジェクト・ストレージは、幾つかの軸で分類と類型化が可能です。以下、それを見て行きます。

提供方法

IA サーバ上で稼働するソフトウェア（SDS）が主流であるが、中にはアプライアンス、または「S3」に代表される、クラウドサービスの一環としてストレージ・サービスとして提供されているものもあります。

アーキテクチャ

並列スケールアウト型とコントローラ・ノードとストレージノードで役割分担を行う階層型に大別されます。

データ保護方式

前述した様に、レプリケーション（複製）とイレージャ・コーディング（消失訂正符号）がありますが、何れかしかサポートしないものと両方サポートするものがあります。両方サポートするものは、オブジェクト毎にレプリケーション数も含めて選択できます。

API

http をベースとした REST が基本ですが、S3 もサポートするものが多いです。また、中にはゲートウェイを介さず、ネイティブで NFS/CIFS/iSCSI をサポートするものもあります。オブジェクト・ストレージが普及しない最大の要因として、従来型ストレージとの API 非互換があるので、今後はネイティブでレガシーAPI（NFS/CIFS）サポートするものが増えてくる可能性が高いです。

地理的分散

WAN 越しにクラスターが組める（オブジェクトを地理的に分散可能な）ものとそうではないものに大別されます。

プラットフォームまたはファイルシステム

OS プラットフォームとして Linux を前提としているもの（Linux 上のアプリケーション)が多いが､中には OS を必要とせずベアメタルで稼働するものもあります。見方を変えるとファイルシステムの要否と考える事もできます。

その他

データとメタデータのセットが論理的であるか物理的であるかとか、名前付オブジェクトサポートの可否とか、オブジェクト探索アルゴリズムによる分類とかも考えられます。

想定用途

ベンダーの想定用途としては、クラウドストレージ、大規模汎用ストレージ、およびアーカイブ・ストレージに大別されます、従来ストレージとの API 非互換性から、国内では大規模汎用ストレージとしての用途は全くといって良い程注目されていません。前述した特性から、アーカイブ・ストレージとしては徐々に注目されつつあります。

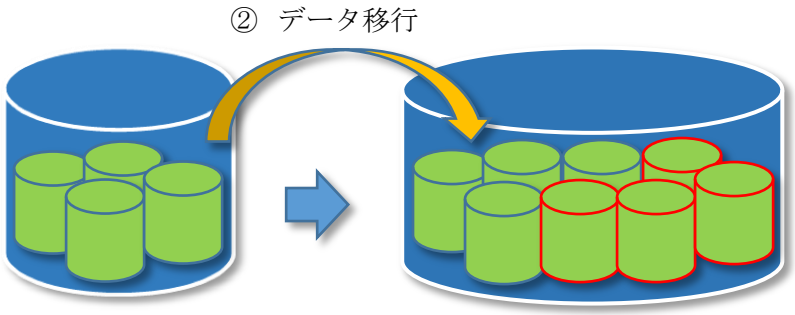
国内におけるオブジェクト・ストレージの実績としては、クラウドストレージ以外ではアーカイブ・ストレージとしての用途が主流で、医療画像保存システムの PACS における 2 次ストレージとしての用途は複数あります。

５．オブジェクト・ストレージの運用者/利用者からみた利点

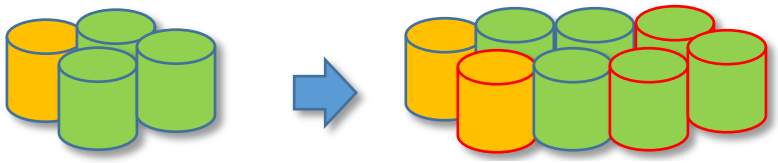
５．１．オブジェクト・ストレージの運用者/利用者から見たメリット

容量拡張・縮小の負担軽減

一般的なストレージシステムは HDD を束ねた RAID グループにより冗長性を確保しています。増加し続ける非構造化データは RAID グループを圧迫し運用管理者は容量を拡張する業務に追われます。容量拡張する為には必要な容量の RAID グループを作成し、既存 RAID グループ内にあるデータを新しい RAID グループにデータ移行する必要があります。大量なデータを移行する為には事前移行から差分移行を数回繰り返し、最終的な静止点確保の為システム停止を行う必要があります。運用管理者は容量不足となった RAID グループの容量拡張の外にシステム停止に伴う他システムとの調整が必要となります。



オブジェクト・ストレージでは RAID グループを使用していませんのでオンデマンドでの拡張が可能となっており、必要な時に必要な容量を自由に拡張できます。



またオブジェクト・ストレージでは容量を必要としなくなった場合、RAID グループの縛りが無い為、構成ディスクを除外するだけで簡単に容量を縮小する事が可能です。

需要予測の困難からの開放

一般的なストレージシステムは HDD をコントロールするストレージコントローラと実際のデータを格納する HDD の 2 つの要素から構成されています。また、ストレージシステムでは 1 つのストレージコントローラで管理できる HDD の個数に上限がある為、購入時に使

用容量の需要予測を行いストレージコントローラの種類と個数を選択する必要があります。多くのストレージベンダーは保守期間を 5 年としており、ストレージシステムを 5 年間使用する需要予測を行う必要があります。

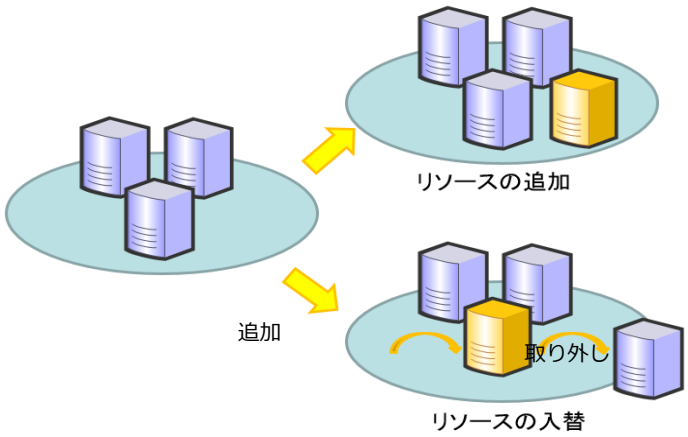
昨今、ストレージに格納されるデータ量は爆発的に増加しており需要予測が難しい状況となっております。また、新事業の立上げ、不採算事業の廃止と企業におけるビジネススピードが加速されております。需要予測が少ないと 5 年を待たずして新しいストレージを購入する必要が出てきてしまい、需要予測が多すぎると使用しないストレージシステムに過剰投資することになります。運用管理者は保守期限前に新しいストレージを購入する事を避けるため余裕を持った設計をしがちです。

これに比べオブジェクト・ストレージは HDD1 本から柔軟な追加／削除が可能であり、運用管理者は複雑な需要予測から解放され、またストレージシステムへ過剰投資を行わずに済む様になります。

スタートアップの容易さ

一般的なストレージシステムは新規導入時に数千万円から高いものでは数億円するものまであります。この様な高価なシステムへの投資の為に社内調整、稟議を実施し予算を確保する必要があります。

オブジェクトストレージシステムでは PC サーバベースにソフトウェアで管理するものも多くあり、性能が要求されるシステムには使用できないが、減価償却が終了していない PC サーバに HDD とメモリを追加して構築することも可能です。一旦構築したシステムに関しても減価償却が完了したものより逐次入れ替えを行い、性能を維持しながら容量 UP をすることも可能です。



社内に眠っているサーバリソースを活用するスモールスタートが可能であり、これからオブジェクト・ストレージの導入を検討している企業にはスタートアップへの敷居を下げる事が可能です。

大量データの格納

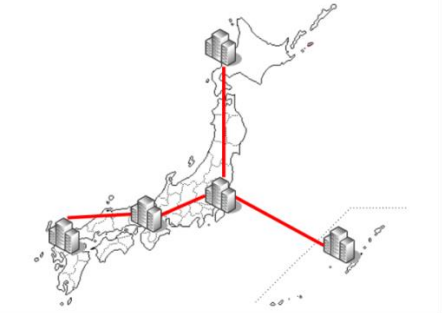
オブジェクト・ストレージは大量のデータ格納を得意としております。ファイルサーバ等ではディレクトリによる階層化により組織、用途別にデータの格納を行います。システムによっては作成できるディレクトリ数、階層数に制限があり大量にデータの格納を行った場合目的のデータにたどり着くまで大変です。

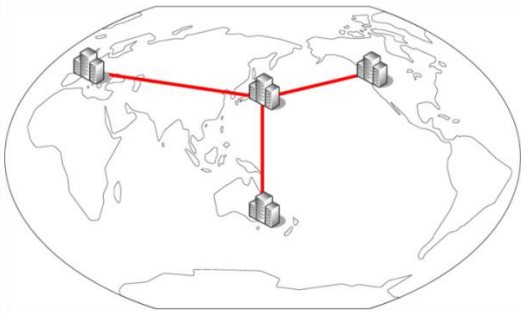
オブジェクト・ストレージにはディレクトリという概念が無いフラット構造である為、1システムで数億オブジェクトの格納も可能です。

冗長構成のカスタマイズ

オブジェクト・ストレージは、同一オブジェクトのコピーを複数ノードに保存することによって、冗長性と高可用性を実現しています。コピーされるノードの数はそれぞれの環境ごとのポリシーによって決めることがかのうであり、ノードの配置も同一データセンター内、複数データセンター内さらには複数の国のデータセンターに分散する事も可能です。

分散するノードの配置をカスタマイズする事により、冗長化レベルを複数提供する事も可能であり、運用管理者はサービスレベルの差別化を提供する事が可能であり、利用者は複数のポリシーからサービスを選択する事が可能です。





パブリッククラウド型オブジェクトストレージサービス利用

オブジェクト・ストレージを使用したクラウド事業を提供する事業者も増えてきており、GB 単価も低価格化しております。非定型で長期保存を要するが、アクセス頻度の少ないデータをクラウド事業者が提供するサービスを利用する事により、

・バックアップ管理からの解放

サービス事業者がバックアップを実施してくれる為、煩わしいバックアップ設計運用から解放される

・リソース管理からの解放

必要な容量を必要な時に利用でき急な需要拡大にも対応可能

・システムメンテナンスからの解放

煩雑なシステムメンテナンスはクラウド事業者が実施してくれます。

が実現できます。

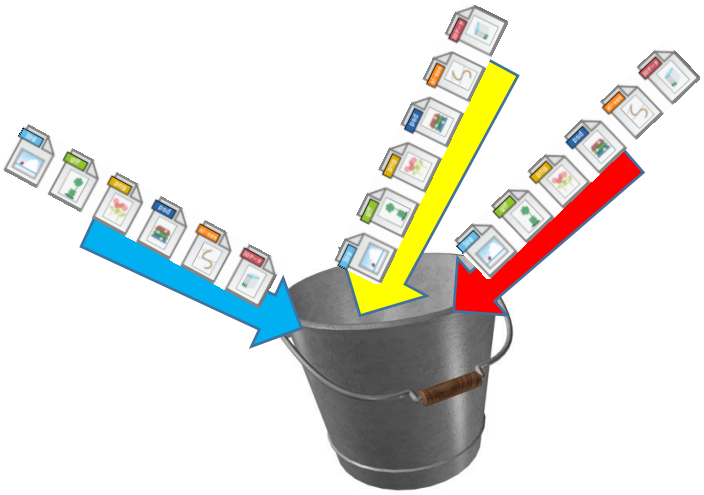
６．オブジェクト・ストレージの実装について

６．１．オブジェクト・ストレージの用途

オブジェクト・ストレージはアクセス方法が http ベースであることからファイルシステムに依存せず非構造化データを手軽に保管できるという便利さがある半面、データベースのような構造型データの保存には向きません。ファイルサーバのように保存したデータを直接、閲覧もしくは変更はできず、閲覧、変更するには、その都度ファイルをダウンロードする必要があるため、パブリックの有償サービスを利用する場合には、ダウンロード/アップロードが課金の対象となる可能性もあります。

初期のハードウェア導入費用がほとんどかからないメリットは大きいのですが、運用（使い方）によっては思いもよらぬ費用が必要になる可能性があります。このような使いにくい面もあることをからオブジェクト・ストレージの有効な利用方法としてストレージ階層化におけるコールドデータの長期保存にもっとも有効であると考えられます。

頻繁に使用するストレージに堆積している使用頻度が低い、更新をほとんど行わないデータの移行先（バケツ的にデータを溜め込む）や、更新を前提としない種類のコールドデータなどを保存するアーカイブ先として CAS のような使い方が適していると考えられます。



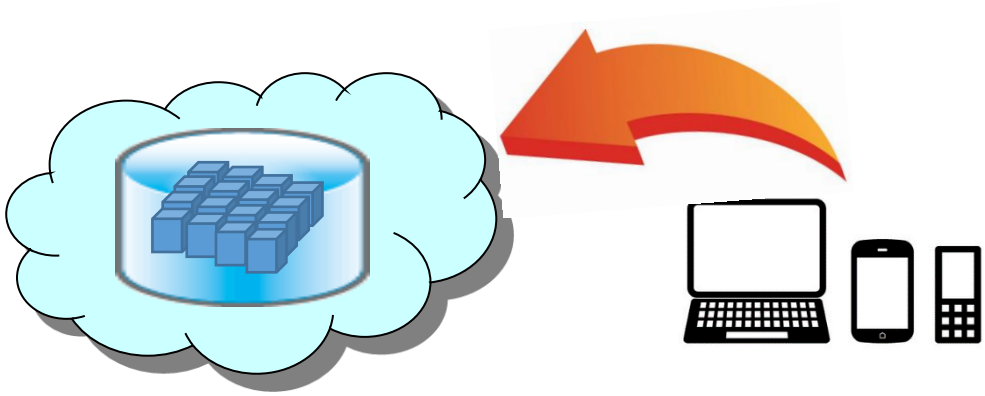
６．２．利用者としてのオブジェクト・ストレージ

実際、オブジェクト・ストレージはどのように活用されているのでしょうか？オブジェクト・ストレージの最大のメリットは容量制限へのストレスから開放されることです。数多く展開されているクラウド・ストレージサービスの基盤としてオブジェクト・ストレージは存在しており、利用者はオブジェクト・ストレージを意識することなく広く利用されています。

また、もはやスマートフォンの電話帳や写真、動画データなど携帯電話会社にて用意され

たクラウドサービスを利用しバックアップ/リストアをする事が当たり前ともなっており、個人が携帯するデータがどんどん肥大化していく中でクラウドサービスはなくてはならないものにもなっていることから、オブジェクト・ストレージとして意識される事はないとしても、多くのユーザに利用されているといえるでしょう。

利用する側として OS やファイルシステムをまったく気にすることなくソフトウェアの指示に従って気軽にデータをバックアップ/リストアができることが利用者にとって非常に使いやすい点となります。もともと、オブジェクト・ストレージは、不特定多数のユーザへのコンテンツの配信や保存など、サーバへの多量のアクセスをサービス提供サーバから切り離すことが主な用途でした。



６．３．運用者としてのオブジェクト・ストレージ

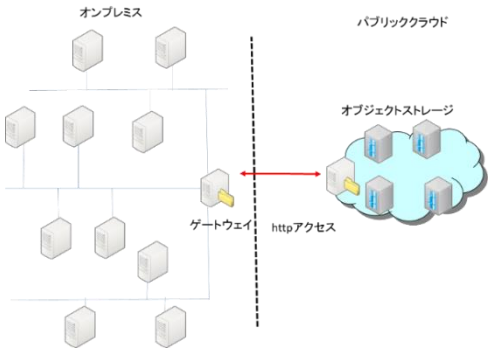
オブジェクト・ストレージは構築時には小規模でのスタートが可能で都度スケールアウトを容易に行えることから物理ストレージのように増設にともなう設計や構築などの労力はほとんどなく、物理的な仕様に縛られることがなくスケールアウトできます。

オブジェクト・ストレージの構築は、オブジェクト・ストレージ構成ソフトウェアをサーバへインストールすることによって構築が可能であり初期のハードウェア費用をほとんどかけない事も可能です。もしくはベンダーから発売されているアプライアンスサーバを利用する方法もあります。サーバにマウントされているストレージをそのままオブジェクト・ストレージとして活用することになり、対象となる容量は、サーバに搭載（マウント）されたストレージ（DAS、SAN、NAS）なら利用できるが、安価な構成を組む事やスケールアウト時に増設するサーバのスペックのばらつきを抑え管理を容易にする面でも一般的にはサーバに搭載（内蔵）されているストレージ（Disk）を利用します。

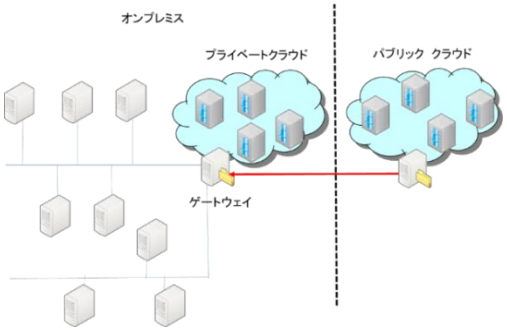
６．４．公開とリソース管理

オブジェクト・ストレージを直接利用するのはなく、通常のファイルサーバのように使う場合やバックアップ先として使う場合、つまり管理する場合は SCSI プロトコル（ブロックストレージ）や NFS/CIFS（ネットワークファイル共有プロトコル）ではオブジェクト・ストレージには直接アクセスできないので、どこかで変換が必要となります。一般的にはゲートウェイサーバを設けて、プロトコル変換を行います。ユーザからはゲートウェイサーバがいわゆるファイルサーバとして機能し、ゲートウェイサーバが公開したリソースを使うことでリソース管理ができます。

パブリッククラウド利用



オンプレミ・スクラウド利用

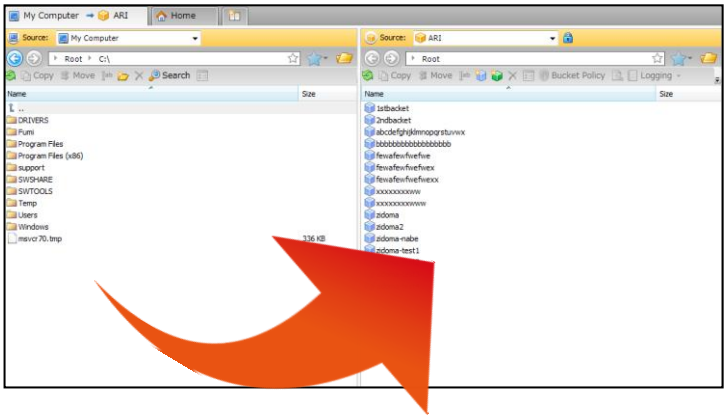


６．５．バックアップ（アーカイブ先）としての利用例

既存のブロックストレージやファイルサーバ（NAS）に保存されているデータのバックアップ先としてオブジェクト・ストレージを利用するにはバックアップサーバにゲートウェイ機能を持たせるか、バックアップ先のオブジェクト・ストレージと既存環境の狭間にゲートウェイサーバを設けて、既存環境へバックアップ先となるオブジェクト・ストレージをファイルサーバやブロックストレージとして公開する必要があります。ゲートウェイサーバから接続するオブジェクト・ストレージはパブリックでもオンプレミスでも構いません。

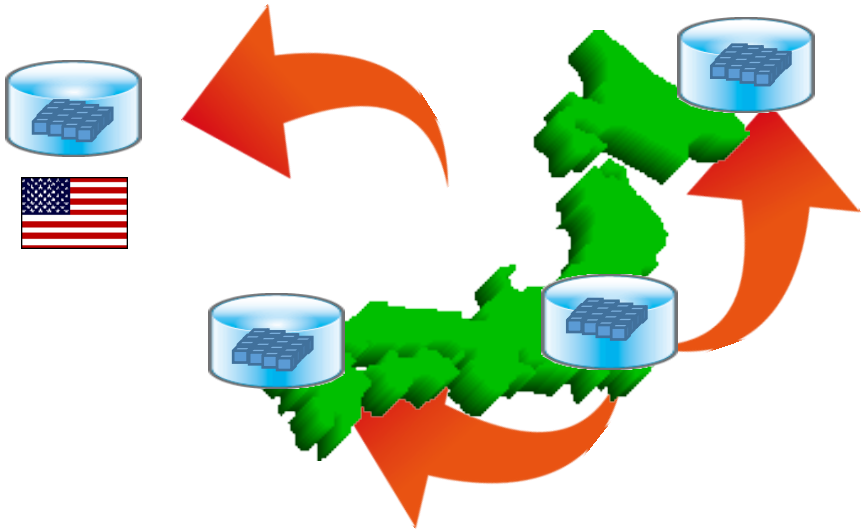
６．６．直接利用する場合

ユーザが直接オブジェクト・ストレージを利用する場合、それぞれのアカウントで Web ブラウザを利用してオブジェクト・ストレージにログインして利用します。また、エクスプローラ風な使い勝手の良い GUI ツールでドラッグ＆ドロップでオブジェクト・ストレージへファイルのアップロード/ダウンロードをすることもできます。



６．７．災害対策

オブジェクト・ストレージ自体はバックアップが容易に構成できるということも利点になります。オブジェクト・ストレージを実現しているベンダーによって冗長化の仕様は様々ですが、オブジェクト・ストレージはフラット構造で簡単であるためレプリケーション構成を容易に組むことができ、遠隔地に複数のレプリカを作ることが可能であり、耐障害性に関しては優れているといえます。



６．８．注意点

運用者として注意すべきこととしては、ファイル管理（世代管理とプロファイルの管理）とバックアップウインドウがあります。

オブジェクト・ストレージは、ファイルを直接更新するということができないのでファイルの更新はアップロードとなります。ポリシーの設定にもよるが、日付更新で上書きしない場合には、ファイルはどんどん増えていくため、世代管理が必要となります。複数ユーザでアクセスしファイルを共有している場合には一定ルールで削除していくなど世代管理や重複除外機能が必要となります。

オブジェクト・ストレージをゲートウェイ経由で利用する場合には、オブジェクト・ストレージに保存したデータはゲートウェアからログインしたアカウントのプロファイル（所有者、パーミッションなど）で扱われます。これを念頭において構成の設計を行う必要があります。

ファイルサーバのようにユーザごとにアクセス権を変える場合には、バケットと呼ばれるフォルダ（ディレクトリ）のような単位でアクセスを管理する必要があり、オブジェクト・ストレージのアカウントとユーザのアカウントの紐付けなど設計には戸惑うことになるかもしれません。

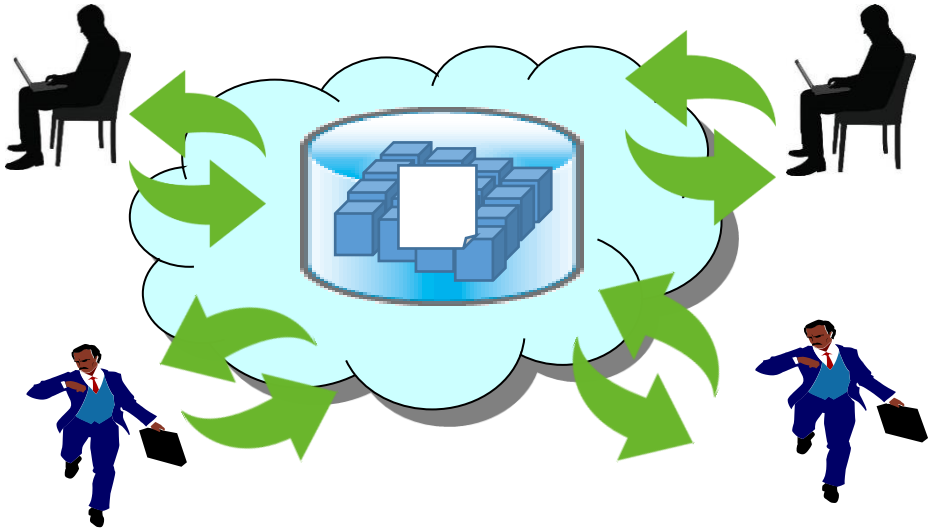
バックアップウィンドウ（処理速度）には注意が必要です。DAS や SAN や NAS などの単一筐体のストレージやテープ装置とは違ってオブジェクト・ストレージはそれを構成しているサーバの処理能力（CPU 速度、ストレージ能力、回線速度、プロトコルの変換速度）の違い、またスケールアウトを重ねた場合にはユニット間の性能差によるばらつきが発生します。

この辺を注意してバックアップを設計しないと予期せぬトラブルに陥る可能性もあります。

６．９．オブジェクト・ストレージ　利用例および事例

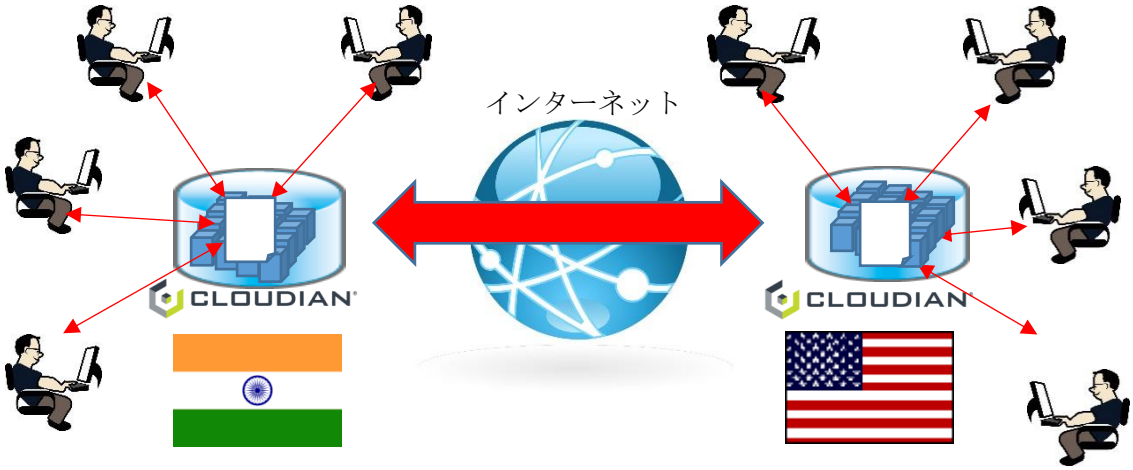
６．９．１．外出先、社内での用意な情報共有

外出の多い営業マンが社内のスタッフと資料などを手軽に共有するために、パブリッククラウドを利用するなどの使い方をしている実例もあります。この場合、モバイル端末でも簡単にアクセスしファイルをダウンロード/アップデートできるクラウドストレージは非常に利便性が良いといえます。主に外出先からは閲覧が多いので低コストで運用できます。



６．９．２．拠点間でのファイル共有事例

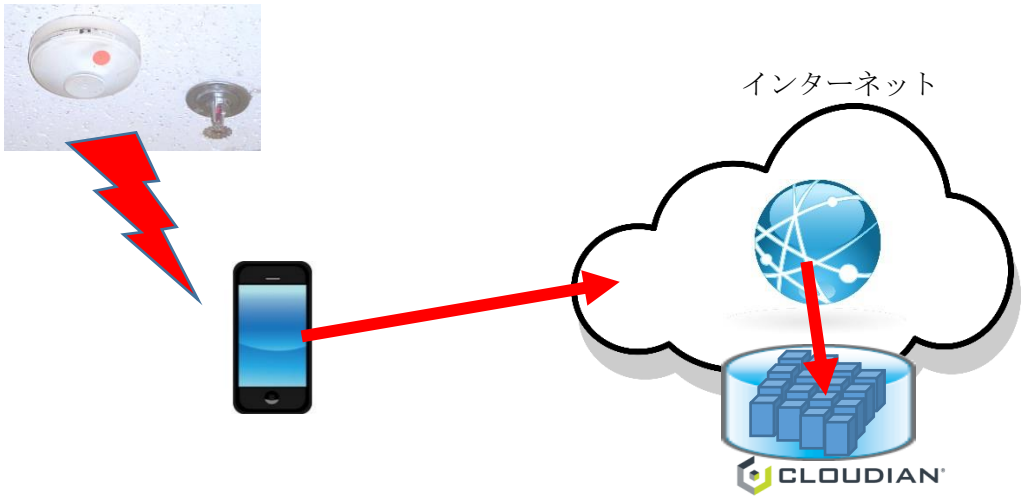
フトウェアの共同開発など、異なる拠点でデータを共有するためファイルの更新管理が非常に煩雑となっていたシステムを、オンプレミスにオブジェクト・ストレージを構築し、拠点間を同期し世代管理することにより、異なる拠点での煩雑なファイル更新管理から開放されます。



引用　CLOUDIAN

６．９．３．IOT サービスとしての活用事例

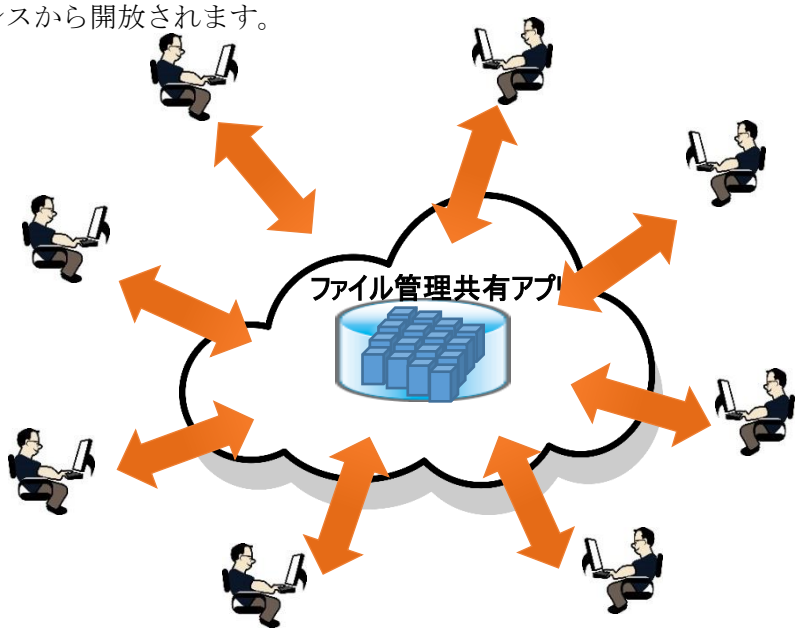
終端装置（スプリンクラー）からの大量のログを、インターネットを介して階層を気にせずにオブジェクト・ストレージに蓄積します。



引用 CLOUDIAN

６．９．４．オンプレにてファイル共有アプリを使用しファイルサーバとして運用事例

ファイルサーバの容量が限られており社員から不満が出ていました。オンプレでオブジェクト・ストレージをファイル共有アプリを使用することにより、オンプレにてクラウドストレージとして使用します。運用側として容易に拡張できることから、ファイルサーバの容量制限のストレスから開放されます。



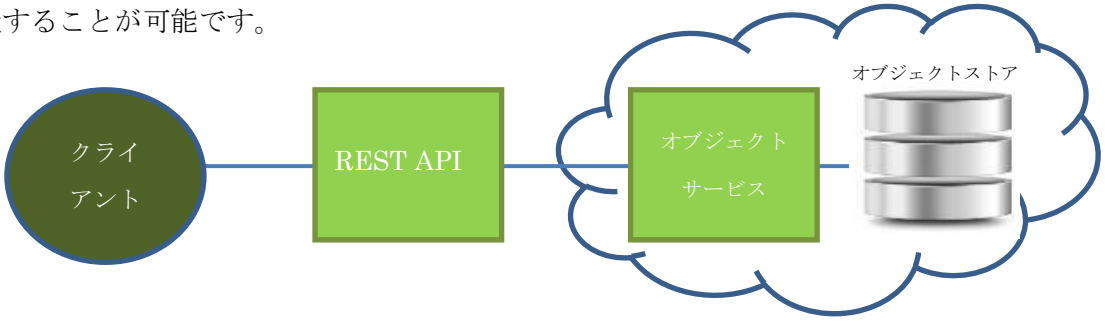
引用 CLOUDIAN

７．サイジングの考え方

７．１．オブジェクト・ストレージの性能検証について

一般にオブジェクト・ストレージはスケールアウトが容易な技術であり、スモールスタートかつオンデマンドで容量を拡大していくことが可能です。将来のストレージ使用量の見積が難しい昨今、これは大きなアドバンテージですが、言い換えれば構築時には容量よりも性能が重視されることになります。ここでは性能 SLA を満たす製品、ノード数を適切に選択するためのひとつの手法を紹介します。運用開始前に実施しておくにより、性能問題を未然に防ぐことが期待できます。

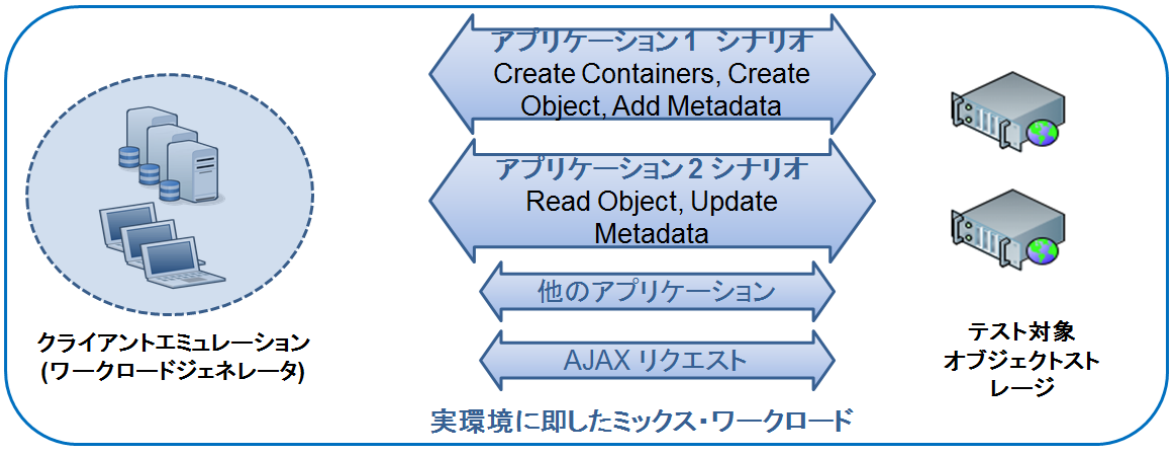
オブジェクト・ストレージは分散化されており、全体の整合性を取るためのアップデートに時間を要し、高い性能を要求するアプリケーションには不向きでした。しかしながら現在、多くのオブジェクトストレージベンダーは、性能制限の問題が改善されクラウドやビッグデータなど様々なワークロードに耐えうるようになったことを述べています。
これまでのブロック、ファイルストレージと異なり、オブジェクト・ストレージには性能を検証するナレッジ、経験則がありませんが、検証手法を流用することで正確な性能を把握することが可能です。



７．２．性能を見極める手法

ワークロードの定義

ストレージの性能を正確に見極めるには、本番環境のワークロードを再現し、ストレージに掛ける方法が最も有効です。Read/Write/メタデータアクセス比、バケット数、オブジェクト数、収容クライアント数などにより、ストレージの性能が大きく変わってしまうからです。特にメタデータアクセスは多くの割合を占めることが多く、これを出来る限り現実に忠実に再現することが重要になります。これまでファイルストレージにおいてはベンダーが提供するトレースし統計情報を得て、試験する方法がありました。この方法はオブジェクト・ストレージにも適用可能です。しかしながら、オブジェクト・ストレージが初めて導入される場合がほとんどである状況では、どのようなアプリケーションで使用するか、クライアント数がどの程度かを含めた等の情報を元に、ワークロードを想定しなくてはなりません。オブジェクト・ストレージであっても、リプレースや、ノード追加の場合は、現環境のワークロードを参照し、ワークロードシナリオを作成することができます。



実際のワークロードは、上記のように様々なアプリケーションによるワークロードが混在します。これらを再現することが必要です。

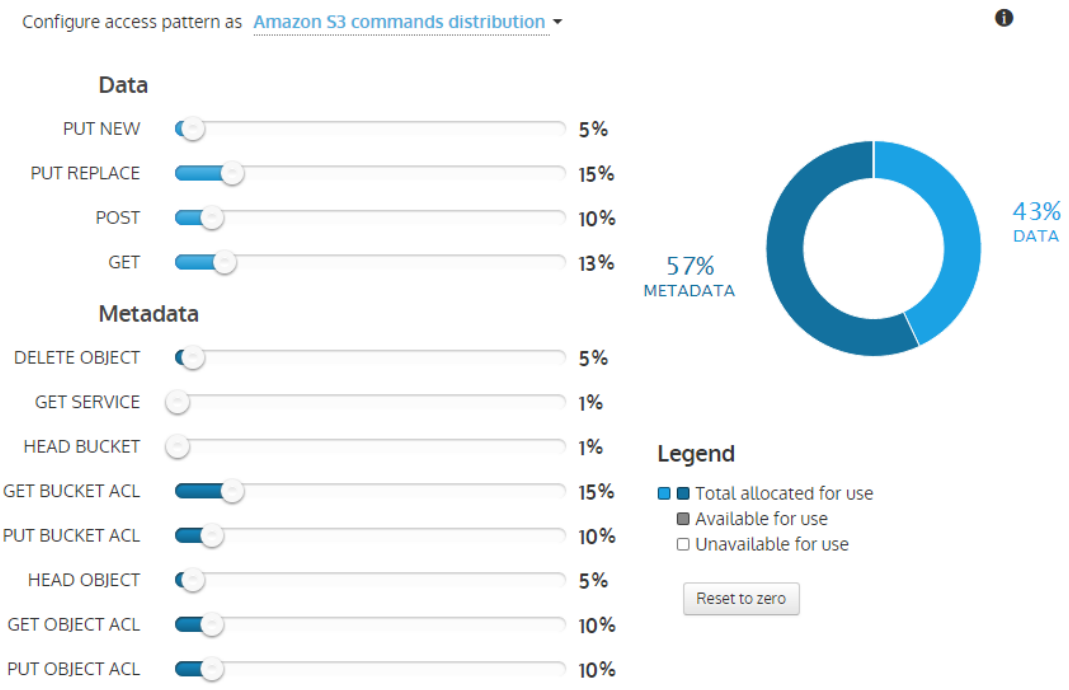


図　Amazon S3 のコマンド分布例

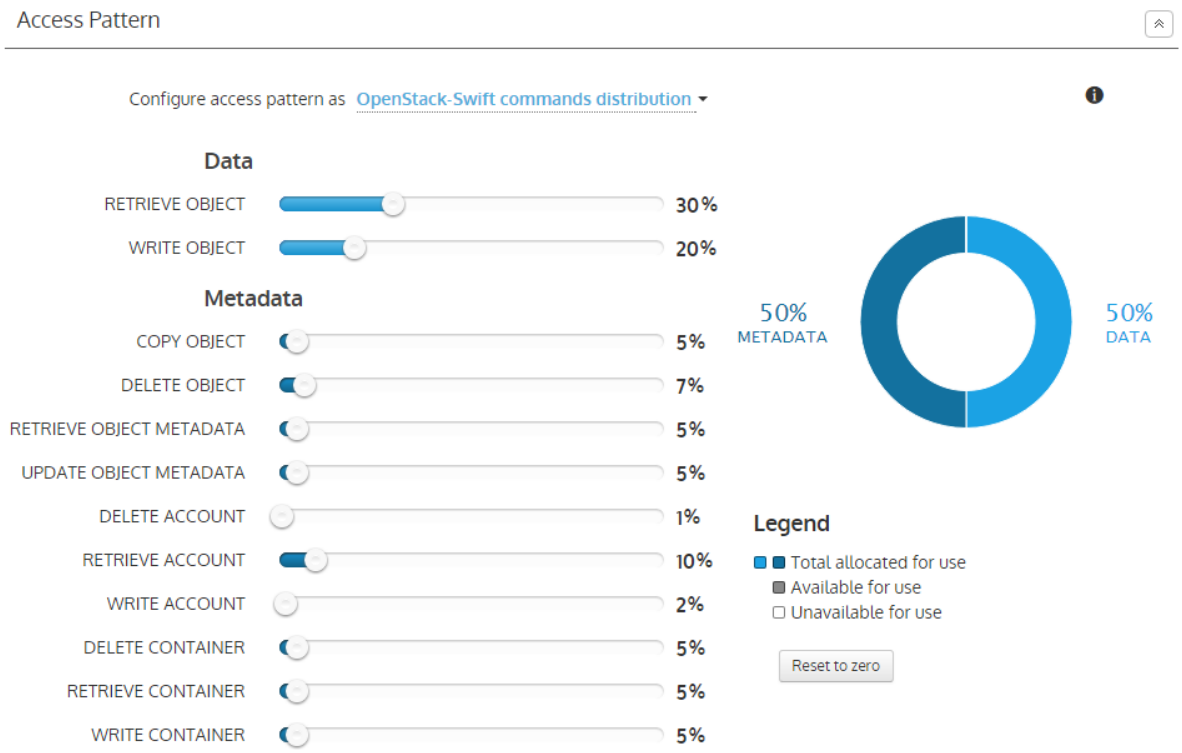


図　OpenStack Swift のコマンド分布例

上記の 2 つは、AmazonS3,OpenStack Swift それぞれの場合のコマンドミックスを再現した一例となります。アプリケーションごとにこれらのコマンド分布を作成し、複数を同時に実行します。

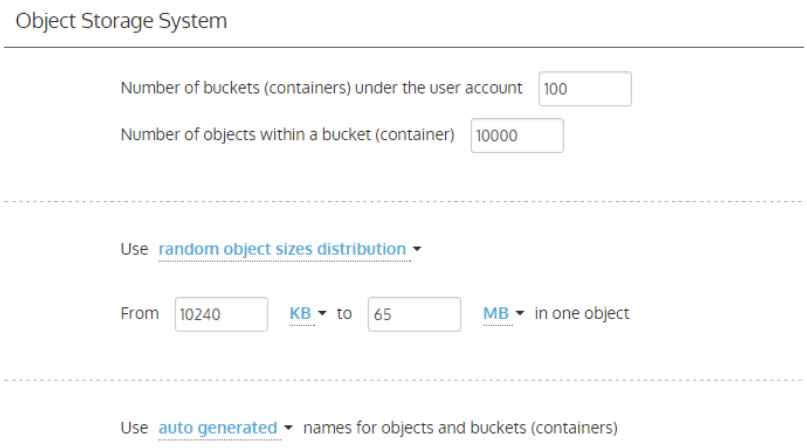


図　バケット、オブジェクトの定義

比較参照のためにファイルストレージ(NFSv3)の場合のコマンド分布を記述します。一般にNFS では Read/Write を大きく上回る割合のメタデータアクセスが発生します。

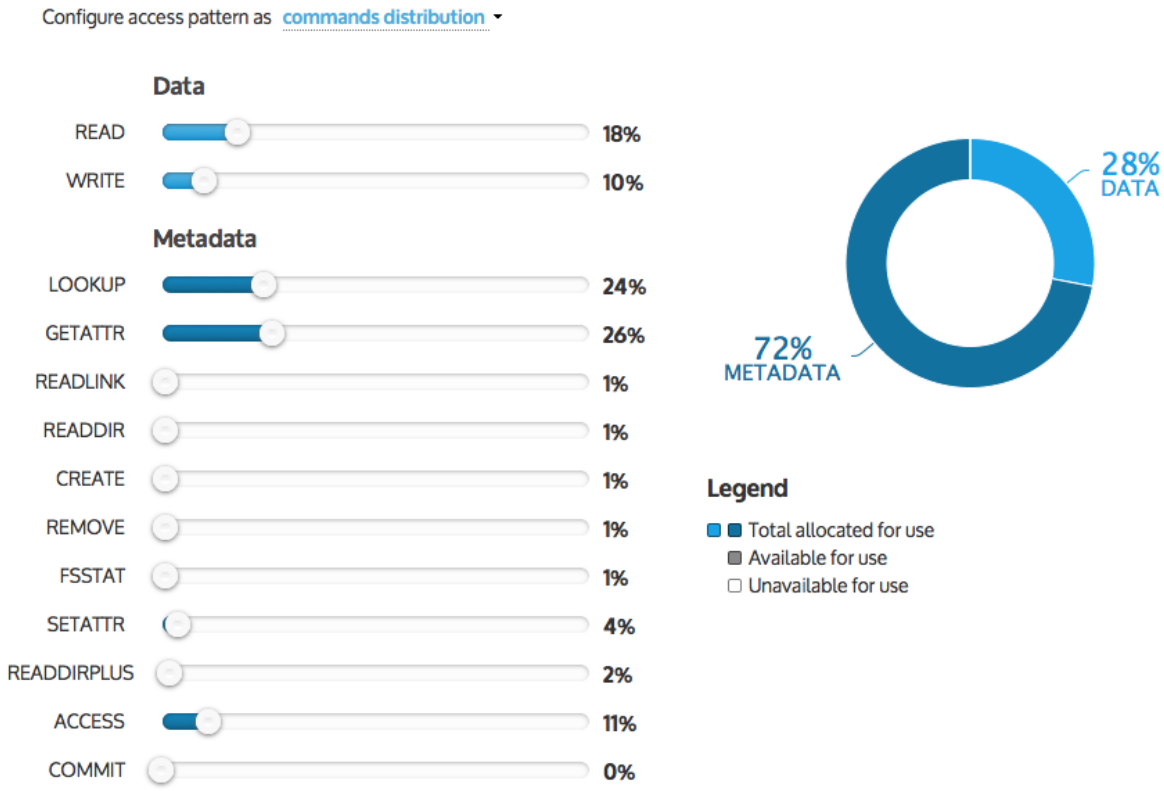


図　NFSv3 のコマンド分布例

最大性能試験

上記で定義したワークロードをもとに、オブジェクト・ストレージの最大性能を検証します。ワークロードジェネレータを使用します。

・最大 IOPS　最大スループット、オーバーオールレスポンスタイム
　　ノード数での変化をみる
　　どれだけのノード数を用意すべきかを確認
・クライアント数を増やした時のオーバーオールレスポンスタイム遷移を確認
　　クライアントをどれだけ収容できるかを確認

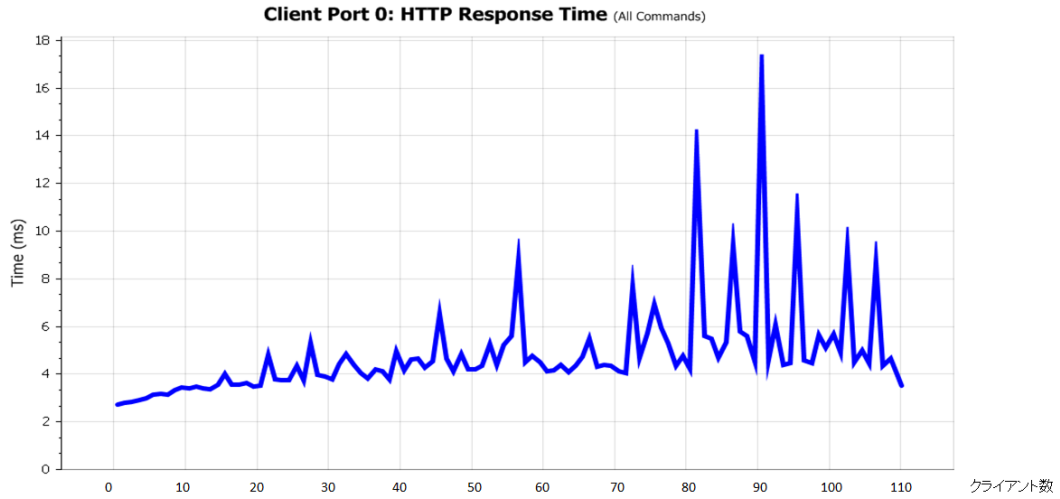


図　オブジェクト・ストレージ(OpenStack Swift)のレスポンスタイム遷移
クライアント数 70 以上から顕著に劣化

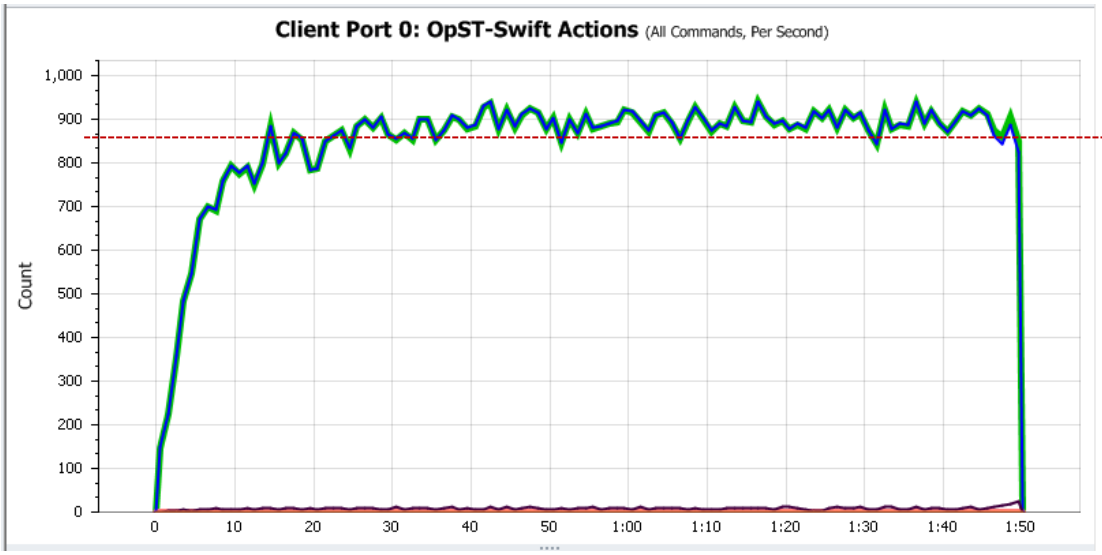


図　オブジェクト・ストレージ 1 ノード時の最大 IOPS　(約 890)

オブジェクト・ストレージの性能を見極めるためには、

・適切なワークロードで試験をすること

Read/Write/メタデータ比

バケット数、オブジェクト数/サイズ分布

複数のアプリケーション/クライントのワークロードを並行させた試験

・SLA を満たすかどうか確認する

オブジェクト・ストレージに限らずストレージは、事前想定通りの性能を発揮することが稀です。適切な測定手法で事前に検証をすることが望ましいです。

８．オブジェクト・ストレージの得意/不得意分野

８．１．オブジェクト・ストレージが適している分野・データ

従来、商業活動や自然現象などを数値化した意味のあるデータ（情報）として活用してきました。そのような数値データを、より効率的に、より生産的に活用するために、計算機（コンピュータ）が生まれ、活用が高度化していく中で、主にリレーショナルモデルをベースとしたデータベースに構造化して格納されて活用したため、そのようなデータをのちに構造化データと呼ぶようになりました。さらに、IT は、組織や人間の様々な活動をより効率的で生産的にするために、数値、文書、画像、音声、動画など、人のコミュニケーションをより表現するデータを活用するようになりました。

このようなデータは、構造化データという分類を超えて、完全な構造定義を持たない半構造化データ、あるいは、構造定義を持たない非構造化データとして分類されるようになりました。

今までのストレージを分類してみると、

・元々のストレージは構造化データを格納・利用するために作られてきた

＝＞ブロックストレージの利用

・ファイルシステムが誕生し、ファイルレベルが使えるようになる

＝＞非構造化データ増加の要因になった

・管理者からみた現状のオブジェクト・ストレージは Software Defined Storage(SDS)と同様の定義（≒SDS）

＝＞RAID の課題を超えたアーキテクチャ

RAID/LUN/パーティション/従来のシン・プロビジョニングでも複雑

＝＞ファイルシステムを意識しなくて良いアーキテクチャ

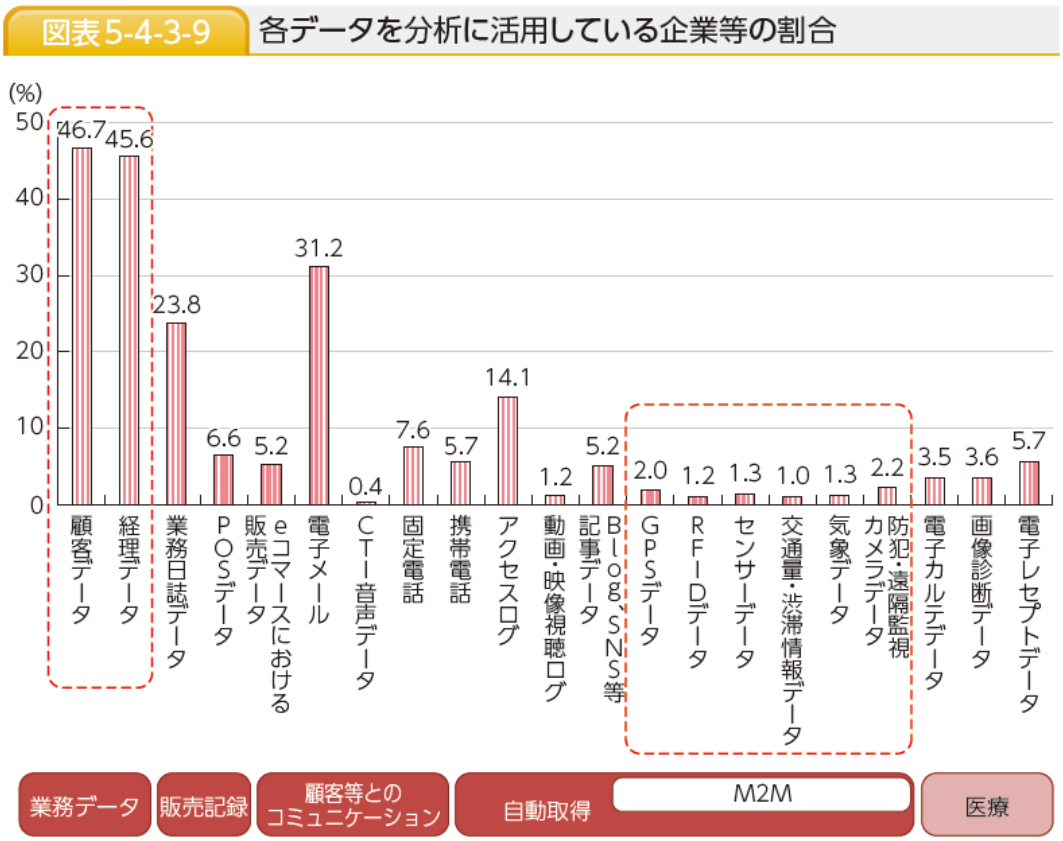
階層構造を持たない

と分類されます。

オブジェクト・ストレージが注目される理由

分析可能なデータは整理された構造化データに限られていたが、データ解析技術の進展により、一見無秩序な非構造データから法則性を抽出することが可能となった。その結果、従来は見過ごされていた様々なデータの潜在的価値が発見され、データの取得・蓄積に要するコストの低下とも相まって、企業がデータを収集するインセンティブが著しく高まっている。このことも、データ流通量が急速に拡大する要因となっている。

代表的なデータの種類で考えてみる



（出典）総務省「ビッグデータの流通量の推計及びビッグデータの活用実態に関する調査研究」（平成27年）

ここにあるようにブロック／ファイルストレージが適している部分もあるが、以下の部分はオブジェクト・ストレージが適している。

理由

① ストレージ・データの管理が容易

今後はバックアップを取ることが困難になる（しかし、レプリケーションは残る）

② データ保護手法が通常のストレージ RAID よりも良い

③ いまいま、オブジェクト・ストレージに適したデータは

適している

- ✓ ブロックストレージを念頭に作られていないアプリ
- ✓ 非構造化データ
- ✓ バックアップがとれないぐらい巨大なデータ
- ✓ 動画、画像、文書、テキスト、大量の OFFICE 系ファイル
- ✓ 半構造化データ（非構造に近いもの）
- ✓ XML
- ✓ データの保全性（高信頼性）が必要な場合

除外されるものは

- ブロックストレージを念頭に作られたアプリ
- 構造化データ：データベース系、半構造化データ系
- 半構造化データ（構造化に近いもの）：メール、グループウェア

８．２．オブジェクト・ストレージの将来像

将来 2050 年頃には、ストレージという概念がオブジェクト・ストレージと同じ（ストレージ＝オブジェクト）になる可能性が考えられます。あるいは、オブジェクト・ストレージのようにストレージアーキテクチャをユーザに意識させないで使えるようになることが予想されます。

９．おわりに

本内容はJapan Data Storage Forum(JDSF)に加入している各社の知識及び経験に基づく内容から、オブジェクト・ストレージの概要とその利用方法や将来に向けての課題などをまとめたものです。技術や顧客要件等の変化に応じて、オブジェクト・ストレージの機能や使い易さなど変化すると考えられますが、現時点における理解を助ける内容として理解してもらえる内容だと考えています。

なお、本内容作成において協力いただきました下記関係者に感謝いたします。

（五十音順）

アライドテレシス（株）
ＡＲアドバンストテクノロジ（株）
シーティーシー・エスピー（株）
東陽テクニカ（株）
日本電気（株）
富士通（株）
日立（株）

２０１８年２月
Japan Data Storage Forum
Storage Elemental Technology(SET)部会