1A-04

分散データリポジトリ Raptor の実装と評価

水野　拓†　　古谷　文弥†　　田胡　和哉‡

東京工科大学大学院バイオ・情報メディア研究科コンピュータサイエンス専攻†　東京工科大学コンピュータサイエンス学部‡

1 はじめに

近年のアプリケーション開発では，文字だけでなく動画像や音楽，ソースコードなど容量の大きいコンテンツを扱うことが多い．このことから，容量の大きなコンテンツでもサーバやネットワークの負荷を適切に分散し，少ない負荷で扱えるような機構が求められている．本論文では，コンテンツを用いたアプリケーション開発を支援するオブジェクトストレージである「Raptor」の構造および，物理的に離れたハードウェア同士をネットワークでつなげ，1 つとして動作させることでスケーラビリティを向上させる手法ついて述べる．

2 Raptor

Raptor とは，クラウドコンピューティングを使用したウェブアプリケーションを容易かつ迅速に構築できるプラットフォームである[1]．あらゆるデータをオブジェクトとして扱い，同一に保存・管理するオブジェクトストレージとして動作する．オブジェクトそのものや，オブジェクトを構成するコンテンツ本体であるペイロード，アプリケーション，サムネイル，メタデータ等に URL がそれぞれ意識せずに紐付けられるため，HTTP による利用が容易になっている．そのため Raptor をアプリケーションのプラットフォームとして利用することでコンテンツを用いたアプリケーション開発が容易になる．

アプリケーション側からタグ等の付加情報を容易に追加できる機構を提供し，その情報を元に検索できる機構を提供している．そのため，「あるプロジェクトに関するコンテンツ」，「あるタグがついたコンテンツ」のような柔軟な検索が可能となっている．これにより，プロジェクト等で作成したコンテンツを Raptor で一元管理することで管理が容易となり，そのコンテンツを利用するアプリケーション間での連携も容易になる．

現在 Raptor を利用して開発されているアプリケーションは以下の様なものがある．

- 用途に応じた CMS エディタ
  - 外部公開用
  - ミーティング資料用
  - リアルタイム共有用
- プロジェクト支援ツール群
  - 画面キャプチャツール
  - 動画編集ツール
- その他コンテンツビュアー
  - 動画
  - 画像
  - PDF
  - Office ファイル 等

現在 Raptor は東京工科大学のプライベートクラウド環境であるクラウドサービスセンター[2]において運用されている．さらに，Raptor 上で動作するアプリケーションを利用し，グループでサービスを企画から提案までを行う「プロジェクト実習」[3]も行った．この実習では，プロジェクト進行で発生するミーティング資料やプレゼン資料，プレゼン動画などの成果物を Raptor 上のアプリケーションを用いて管理・共有しながらプロジェクトを半年間実施した．これらの運用により，Raptor がコンテンツを用いたアプリケーションのプラットフォームとして非常に有用であることがわかった．

3 Raptor の内部構造

Raptor のアーキテクチャを図 1 に示す．Raptor は Raptor 中心であるリポジトリ部分の Raptor Core，HTTP サーバ部分の Raptor Servlet，分散ストレージ機構部分の Storage System の 3 つで構成される．

Storage System はペイロードを保存・管理する Storage Server とペイロードを取得・キャッシュ・操作する Storage Client の 2 つで構成さ

Implementation and evaluation of a scalable data repository Raptor

† Taku MIZUNO，Furuya FUMIYA
Bionics, Computer and media science, Entrepreneurship program, Tokyo University of Technology Graduate School.
‡ Kazuya TAGO
School of Computer Science, Tokyo University of Technology.

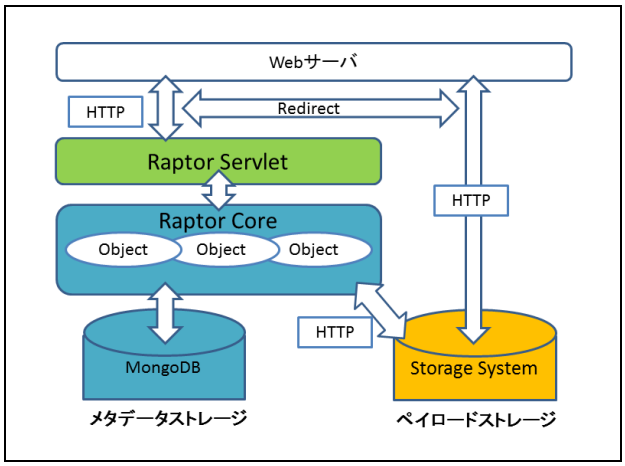

図 1 Raptor のアーキテクチャ

れる．Storage Server と Storage Client 間は HTTP でやりとりする．Storage Server では，ペイロードにバージョンをつけ，不変(Immutable)のデータとして保存・管理をしている．ペイロード名＋バージョン番号で URL を構成するため，Storage Client 側でキャッシュをしても毎回必ず正しいペイロードを取得することができる．また，ペイロードは同一バージョンにおいて不変であるため半永続的にキャッシュしておくことができる．

Raptor からのペイロードの取得の経路は 2 種類ある．1 つは Raptor Core 側の Storage Client から直接取得する経路である．この経路は主にテキストコンテンツの一部のみを取り出したいなどアプリケーション側で細かく操作するためものである．もう 1 つは Raptor Core 側からはペイロード名とバージョン番号を返し，Web サーバ内で Storage Client に対してリダイレクトすることでペイロードを取得する経路である．この経路は Web サーバが直接ペイロードを取得するため動画等の容量の大きいコンテンツをキャッシュする効果がある．

4 分散ストレージによるスケーラビリティ向上の検討

Raptor は東京工科大学全学約 8000 人に向けたサービス提供を想定している．大規模利用を想定した場合，スケーラビリティを考える必要がある．現行の Raptor は Storage System に NFS を使用しているため，ペイロードが不変であるメリットが活かしきれていない．そこで，Storage System をペイロード共有レイヤーとして独立させ，Storage Server と Storage Client とをつなぐ分散ストレージとして，Raptor に外付けでつなぐようにすることで，スケーラビリティの向上をはかる．Storage System ではペイロードが不変であるため，ペイロード名＋バージョン番号で必ず同一のペイロードを取得できる．そのため，複数環境で動作する Raptor のペイロードを共有する際は，必要なバージョンのペイロードを指定して取得することで共有可能となる．また，ペイロードは同一バージョンにおいて不変であるため，各環境の Raptor 内で半永続的にペイロードをキャッシュしても必ず正しいペイロードで共有することができる．この性質を活かすことで，物理的に離れていてもネットワークを介してペイロードを正しく共有することができる．ペイロード共有のインタフェースを HTTP にすることで，共有部分の実装も容易となる．独立させた Storage System を分散ストレージ機構とし，ペイロードストレージにすることで Raptor のスケーラビリティは飛躍的に向上する．

5 実装と運用

Raptor Core，Raptor Servlet は現在 Java で開発を行っており，Storage System は Go 言語で開発している．

ソースコードの規模は現在総行数約 55000 行，ソースコード数約 430 ファイルとなっている．

分散ストレージをつないだ Raptor は東京工科大学クラウドサービスセンターにて運用を行う．現在は一部による利用にとどまっていたが，今後は全学 8000 人に向けてサービスを提供する．

6 おわりに

本論文では，Raptor の Storage System をペイロード共有レイヤーとして独立させ，分散ストレージとすることで，ペイロードの共有が容易となり，スケーラビリティの向上に効果があると考えられる．また，この分散ストレージ機構は Raptor だけにとどまらず，多くのサービスの共有ストレージとして適用可能である．

参考文献

[1] 羽鳥 孝広，井原 雄太郎，田胡 和哉: “次世代クラウドコンピューティング基盤 Raptor の実装と評価”，情報処理学会全国大会第 73 回全国大会 (5X-8，2011).
[2] 東京工科大学: “クラウドサービスセンター” http://www.teu.ac.jp/cloud/ (2015/01/09).
[3] 東京工科大学：“プロジェクト実習” http://www.teu.ac.jp/gakubu/cs/project/index.html (2015/01/09).