Vol.2018-AVM-100 No.16
2018/3/7

エッジクラウドにおけるマルチメディアサービスファンクションチェイニングを活用した処理低遅延化に関する検討

今金健太郎[†1]　金井謙治[†1]　甲藤二郎[†1]　津田俊隆[†1]　中里秀則[†1]

概要：本稿では，マルチメディア処理の低遅延化を実現する仮想エッジクラウドシステムを紹介する．提案システムは，マルチメディアサービスを機能レベルに細分化し，それらをチェイニングすることで仮想的なマルチメディアサービスとして再定義している．本システムを OpenStack により実装し，異なる２つの研究室に展開し，低遅延化の検証を行う．

キーワード：マルチメディアサービスの機能分割，マルチメディアサービスファンクションチェイニング，エッジクラウド，エッジコンピューティング，OpenStack，マルチメディア処理

Edge Cloud System for Low-latency Processing Using Multimedia Service Function Chaining

Kentaro Imagene[†1]　Kenji Kanai[†1]　Jiro Katto[†1]
Toshitaka Tsuda[†1]　Hidenori Nakazato[†1]

Abstract: In this paper, we introduce a virtualized edge cloud system that achieves the reduction of multimedia processing latency. The system re-defines the multimedia service as a virtualized service by using multimedia service partitioning and service function chaining. We implement the proposed system by using OpenStack and deploy into the two different laboratory rooms (e.g., two regions). Evaluation results conclude that the proposed system can reduce the latency of multimedia processing compared to a conventional cloud.

Keywords: Multimedia service partitioning, Multimedia service function chaining, Edge cloud, Edge computing, OpenStack, Multimedia processing

1. はじめに

近年，アプリケーションサービスの高度・複雑化やビッグデータの台頭の影響を受け，モバイルトラフィックや通信遅延，クラウドの処理負担の増大等が懸念されている[1]. これに対し，ネットワークのエッジ部に分散配置したリソースを利用してアプリケーション処理を実行するエッジコンピューティング[2]に関する議論が行われている．エッジコンピューティングのメリットとして，アプリケーション処理要求を行う端末とエッジ間の物理的な距離が，同端末と従来のクラウドコンピューティングで利用されるクラウドサーバ間の距離に比べて大幅に短縮されることが挙げられる．さらに，複数エリアに分散配置されているエッジサーバで分散的にアプリケーション処理を実行できるため，サーバ 1 台あたりの処理負担も軽減される．一方，デメリットとして，それぞれのリソースが小規模なデータセンタのような構造で分散的に存在するため，分散した各エリアで見ると計算リソース量が少なくなること，リソースの配置や選択が複雑になり，管理コストが増大することが挙げられる．そのため，エッジコンピューティングのユースケースとして考えられているマルチメディア処理のような処理コストのかかるアプリケーションを実行するためには，構築が容易で冗長性のあるエッジコンピューティング環境および各アプリケーション特性に適した計算リソース選択が必要と言える．

これを踏まえ，筆者らはこれまで[7]にて，エッジコンピューティングにおける分散処理による低遅延処理を実現してきた．本稿では，これまで提案してきたエッジクラウドシステムの紹介をするとともに，その性能検証として，対象とするマルチメディアアプリケーションを追加し，複数のアプリケーションを複数ユーザで共有シナリオ化といった，より現実的な環境下で性能評価を行う．また，比較対象として，現実的なクラウド環境を導入し性能比較を行う．

提案しているエッジクラウドシステムは，オープンソースのクラウド管理ソフトウェア群である OpenStack[3]を活用している．複数エリアにまたがる仮想クラウド基盤上のエッジクラウドを OpenStack により構築し，マルチメディア処理に対して，マルチメディア処理を単独で動作可能な機能レベルに分割する「マルチメディアサービスの機能分割」と，それらを最適な順番・場所で実行する「マルチメディアサービスファンクションチェイニング」を実行することで，マルチメディア処理の低遅延化を図っている．

†1 早稲田大学
Waseda University

ⓒ2018 Information Processing Society of Japan

2. 関連技術

2.1 ネットワーク仮想化

1 章で述べた背景を受け，ネットワークの効率的な利用を図るために，Software Defined Network (SDN)[4]や Network Function Virtualization (NFV)[5]といったネットワーク仮想化技術に関する議論が行われている．SDN では，従来各ネットワーク機器内にハードウェア実装されていたネットワークの経路制御機能とデータ転送機能を分離し，SDN コントローラと呼ばれるソフトウェアによってこれらを一か所で集中的に制御することが可能となる．一方，NFV では，ロードバランサやファイアウォールといった，従来では各専用のネットワーク機器上で仮想していたネットワーク機能を，論理的に統合，分割された汎用サーバ上で実現することが可能となる．

上記技術に関連して，Service Function Chaining (SFC)[6]では, NFV におけるサービス機能である Virtualized Network Function に対して，適切な順序でパケットを転送するサービスチェイニングを実現している．動作手順例を図 1 に示す．サービスを利用する各ユーザに対して適切なサービスを柔軟に提供するために，まず，転送するパケットにサービスを識別するためのタグをそれぞれ付与する．次に，付与されたタグに基づいてサービス機能を連結したサービスチェインを定義し，最終的にそのサービスチェインに従ってパケット転送を行う．

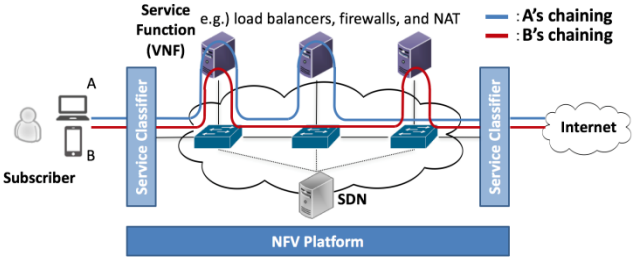


図 1. SFC 動作手順例

2.2 サーバ仮想化

2.1 章同様，サーバの効率的な利用を図るために，サーバ仮想化技術に関する議論が行われている．サーバ仮想化とは，CPU やメモリ，ストレージといったサーバのリソースを物理的な構成にとらわれずに論理的に統合・分割することで，1 台の物理サーバ上で複数の仮想サーバを利用し，リソースの有効活用を可能とする仕組みである．これにより，物理サーバ 1 台あたりの稼働率向上と物理サーバ台数の削減が可能となり，余剰リソースや保守費用，設置スペースの削減を実現できる．また，仮想サーバの台数やスペック．稼働箇所を動的に変化させることで，システムの負荷軽減や処理能力向上，災害時のバックアップのためのスケールアウトを容易にし，移行前と同じ環境で仮想サーバを利用することができる．本稿では，提案するシステムにおいて，オープンソースのクラウド管理ソフトウェア群である OpenStack を用いて Infrastructure as a Service (IaaS)環境を構築している．コントローラノードとコンピュートノードの連携により，仮想サーバやネットワーク，ストレージなどの各機能が実行され，システム利用者の要求にあわせて，仮想サーバであるインスタンスのメモリ拡大やインスタンス自体の複製といったリソーススケーリングを柔軟に行うことができるようになっている．

3. エッジクラウドシステム

本章では，筆者らが[7]にて提案しているエッジクラウドシステムを紹介する．本システムは，仮想クラウドシステムという観点から効率的なリソース利用を実現するために，OpenStack を活用している．

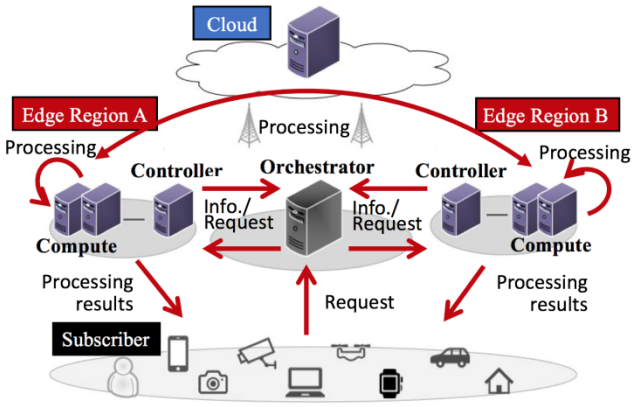


図 2. エッジクラウドシステムの全体構成[7]

3.1 システム概要

提案しているエッジクラウドシステムの全体構成を図 2 に示す．従来のクラウドコンピューティング環境(Cloud)ではユーザから見て遠隔地に計算リソースが配置されている．一方，エッジコンピューティング環境(Edge)では，マルチメディア処理に必要な計算リソースをユーザの近隣に配置することで従来のクラウドコンピューティング環境利用時に比べて通信遅延の削減が期待できる．エッジコンピューティング環境は OpenStack Ocata で構築された複数リージョン(エリア)のクラウド環境となっており，以後，エッジコンピューティング環境をエッジクラウドとして定義する．なお，各リージョンは 1 台のコントローラノードと複数台のコンピュートノードから構成されており，図 2 では例として 2 リージョン構成(Region A, B)の場合を示している．スマートフォンやタブレット，IoT 端末のユーザ(Subscriber)は，これら従来のクラウドコンピューティング環境もしくはエッジクラウドに対して，マルチメディア処理を要求する．さらに，本システムではエッジネットワーク内にオーケストレータを配置し，ユーザの要求情報や各リージョンのエッジクラウドリソースを把握し，エッジク

ⓒ2018 Information Processing Society of Japan

ラウドへのリソース操作指示やマルチメディア処理実行のスケジューリングを担う．オーケストレータの詳細は 3.2 節に示す．

最後に，エッジクラウドシステムにおけるマルチメディア処理手順を以下に示す．

1. オーケストレータおよび各リージョンのコントローラノードが定期的に各リージョンのリソース情報を収集する．
2. ユーザがオーケストレータに対してマルチメディア処理実行リクエストを送信する．
3. オーケストレータが処理 1，2 で得た情報を基にマルチメディア処理の実行スケジューリングを決定する．
4. オーケストレータが該当のリージョンのコントローラノードにスケジューリング情報を送信し，適切なエッジクラウドでマルチメディア処理やリソース操作を実行する．
5. 実行結果をユーザに送信する．

3.2 オーケストレータ

オーケストレータはエッジネットワーク内に配置され，OpenStack コントローラと合わせてエッジクラウドの各リージョンのノード・リンク情報を定期的に収集している．また，アプリケーション情報をあらかじめ与えられているものとし，ユーザの要求情報はユーザのマルチメディア処理要求発生時に随時受け取るものとする．また，それらの情報に基づいて，マルチメディア処理手順のスケジューリングを行う．

本システムでは，マルチメディア処理機能を分割し，エッジクラウド内に各処理機能を搭載したインスタンスを生成し，「マルチメディアサービスの処理機能分割」を行っている（詳細は 3.3 節にて説明する)．そのため，アプリケーションサービスをユーザに提供するためには，それら複数の処理機能を連結する「マルチメディアサービスファンクションチェイニング」を行う必要がある（詳細は 3.4 節にて説明する)．また，既存の処理機能の連結だけでなく，使用予定のエッジクラウドの計算リソースが少ない場合や，対象リージョンが混み合っている場合などに，OpenStack の機能により，リソーススケーリングやリソース複製といった，インスタンスへのリソース割り当ても各リージョンのエッジクラウド配下のコントローラノードへ指令する機能も持つ．

3.3 マルチメディアサービスの機能分割

マルチメディアサービススライシングでは，マルチメディア処理を単独で動作可能な機能レベルに細分化し「処理機能」として定義する．さらに，エッジクラウド内にそれらの処理機能を搭載した専用のインスタンスを立ち上げ，処理結果やデータをサービス間で共有，再利用する．ここで，本技術を人物検出処理に適用した場合の動作例を図 3 に示す．人物検出処理は，大きく「映像の取得」「エンコード」「検出処理」という 3 つの処理機能に分けることができる．マルチメディア処理のサービスは近年多様化しているものの，処理を分割した際の典型的な大枠はいずれも類似している．そのため，オーケストレータがこれらの共通した大枠を認識することで，複数のサービス間で処理機能を共有することが可能で，効率的なリソース利用につながる．

なお，本稿では単純化のため，各インスタンスを一つの処理機能の専用マシンとし，処理機能の分割はアプリケーション実行前に行っておくものとする．マルチメディアサービススライシングを動的に行うためのアルゴリズムの検討と評価については，今後の課題とする．

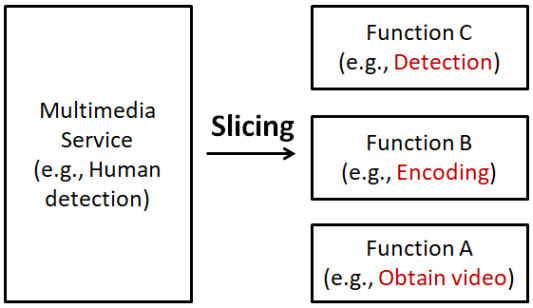


図 3. マルチメディアサービススライシング動作例

3.4 マルチメディアサービスファンクションチェイニング

エッジクラウドにおいて低遅延なマルチメディア処理を達成するためには，分割されたサービスの処理機能を適切な順序，場所で提供する必要がある．そこで，マルチメディアサービスファンクションチェイニングでは，ユーザの要求とリソース情報に基づき，分割された処理機能を適切な順序，場所で提供し，連結することで，一つのマルチメディア処理という形で提供する．

ここで，3.3 節同様，本技術を人物検出処理に適用した場合の動作例を図 4 に示す．まず，ユーザがオーケストレータに対して人物検出処理を要求し，オーケストレータは人物検出処理をマルチメディアサービススライシングによって「映像の取得」「エンコード」「検出処理」という 3 つの処理に分ける．その上で，オーケストレータおよびコントローラノードで取得しているリソース情報に基づき，オーケストレータもしくはコントローラノード上で処理手順のスケジューリングを行う．この場合は，スケジューリングの結果，「映像の取得」を第一の処理としてリージョン A のインスタンスで実行，取得した映像を同じリージョン内の別のインスタンスへ転送した後，そのインスタンスで「エンコード」を第二の処理として実行，さらにそのエンコードした映像をリージョン B のインスタンスへ転送し，そのインスタンスで「検出処理」を第三の処理として実行，最後に処理結果をユーザに転送する，という手順で行うこととなる．

ⓒ2018 Information Processing Society of Japan

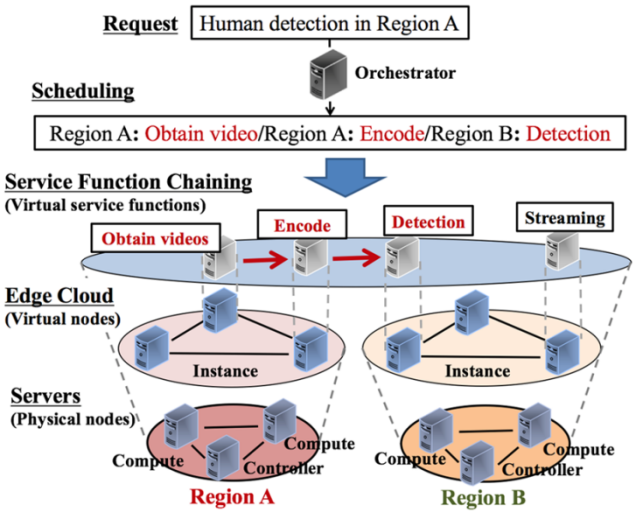


図 4. マルチメディアサービスファンクションチェイニング動作例[7]

最後に，本システムの主な特徴をまとめる．

1) OpenStack を用いたエッジコンピューティング環境

物理的な構成にとらわれない，論理的なリソースの統合管理・運用を実現するために，IaaS 環境を構築するオープンソースのクラウド管理ソフトウェア群である OpenStack を用いてエッジコンピューティング環境を構築する．これにより，必要なリソースのスケーリングを動的に行うことが可能となる．

2) 複数エリア間でのマルチメディア処理機能の共有やデータ再利用

エッジクラウドの複数リージョンの情報を収集するオーケストレータを設ける．情報収集と各処理のスケジューリングを行い，単一エリアのエッジサーバ内および複数エリアのエッジサーバ間で処理機能の共有や処理に必要なデータの再利用により，システム内のリソース利用効率を向上させる．

3) マルチメディア処理のための機能分割とサービスファンクションチェイニング

マルチメディア処理に必要な処理機能を細分化し，専用の仮想マシン(=インスタンス)として生成する（＝サービスの機能分割）．さらに適切なスケジューリングにより，必要な機能を動的かつ適切に選択し，連携させる(=サービスファンクションチェイニング)．これによってマルチメディアサービスを仮想的に再定義する．

4. 遅延特性評価

本章では，3 章で説明したエッジクラウドシステムにおいて，マルチメディア処理を実行した際の遅延特性を評価し，従来のクラウド環境と比較評価する．

4.1 実験環境

本実験の実験環境を図 5 に示す．2 リージョンのエッジクラウド(Region A，B)を大学の 2 研究室内に OpenStack で構築し，コントローラノードを 1 台，コンピュートノードを 2 台ずつ設置する．また，オーケストレータを Region B に 1 台設置する．ここまでの各物理サーバの概要は表 1 の通りである．また，ネットワークカメラをリージョン A に配置して，リージョン A 内にいる人物の様子を撮影しているものとする．各コンピュートノード上には，処理機能別に以下の 4 つの機能を実装したインスタンスを生成した．

1) “Camera”：ネットワークカメラからの映像の取得機能を実装

2) “FFmpeg”：FFmpeg [8]により動画像圧縮機能を実装

3) “YOLOv2”：YOLOv2 [9]により人物検知処理機能を実装

4) “DASH”: MPEG-DASH [10]により適応レート制御による映像配信機能を実装

また，エッジクラウドとの遅延特性比較用に，クラウドコンピューティング環境として，東京都内にデータセンタをもつクラウド事業者が提供するクラウドサーバ 1 台(Cloud)，およびエッジコンピューティング環境としてリージョン A 内にエッジサーバ 1 台(Edge)を生成し，エッジクラウドにある機能を全て持っているものとする．各仮想環境の概要は表 2 の通りである．

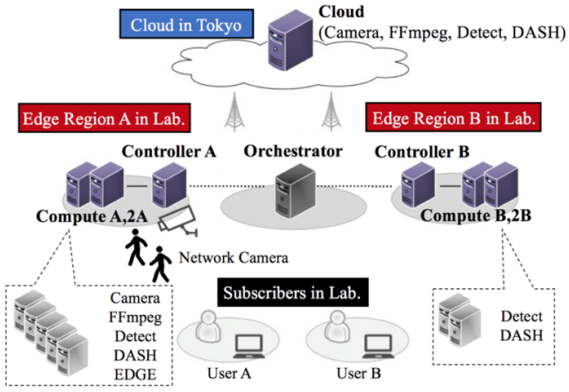


図 5. 実験環境

表 1. 物理サーバの概要

Server	CPU	Memory	OS
Compute A	Intel® Core™ i5-2400@3.10GHz	8GB	Ubuntu 16.04
Compute 2A	Intel® Core™ i7-6700@3.40GHz	16GB	Ubuntu 16.04
Compute B	Intel® Core™ i5-2400@3.10GHz	16GB	Ubuntu 16.04
Compute 2B	Intel® Core™ i5-3330@2.70GHz	4GB	Ubuntu 16.04
Orchestrator	Intel® Core™ i7-4770@2.50GHz	16GB	Ubuntu 16.04

ⓒ2018 Information Processing Society of Japan

表 2. インスタンスの概要

Instance	Region	CPU	Memory	Function
Cloud	Tokyo	20	224GB	ALL
Edge	A(lab)	2	4GB	ALL
Camera	A(lab)	2	4GB	CAMERA
FFmpeg	A(lab)	1	2GB	FFMPEG
Detect	A/B(lab)	2	4/8GB	DETECT
DASH	A/B(lab)	1	2GB	DASH

4.2 マルチメディアアプリケーション

本実験では，実際のエッジコンピューティングサービスのユースケースをもとに,「映像監視システムによる人物検出」「MPEG-DASH による映像ストリーミング配信」の 2 つのアプリケーションを実行する．概要を図 6，7 に示す．

Application 1. 映像監視システムによる人物検出

マルチメディアサービスの機能分割により，「Camera」「FFmpeg」「Detect」の 3 つの処理機能に分割される．ユーザがアプリケーション実行を要求すると，一番目のインスタンス(Camera)がネットワークカメラから Region A 内の様子を映した映像を取得する．次に，二番目のインスタンス(FFmpeg)が取得した映像をエンコードする．最後に，三番目のインスタンス(Detect)がエンコードされた映像に対して人物検出処理を行う．

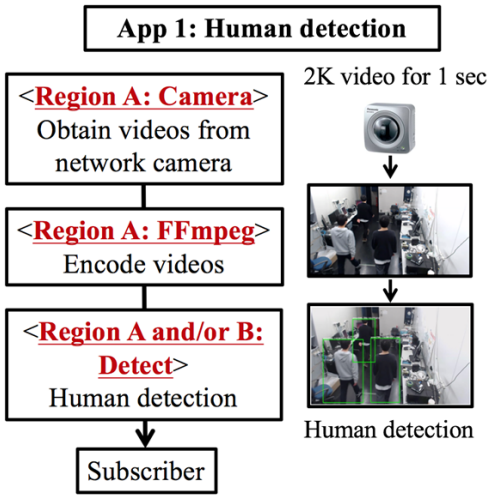


図 6. アプリケーション 1 （人物検知）[7]

Application 2. MPEG-DASH による映像ストリーミング配信

マルチメディアサービスの機能分割により，「Camera」「FFmpeg」「DASH」の 3 つの処理機能に分割される．ユーザがアプリケーション実行を要求すると，一番目のインスタンス(Camera)がネットワークカメラから Region A 内の様子を映した映像を取得する．次に，二番目のインスタンス(FFmpeg)が取得した映像をエンコードする．なお，ここまではアプリケーション 1 と共通している．最後に，三番目のインスタンス(Detect)がエンコードされた映像を 4 つのビットレート(5Mbps，3Mbps，1Mbps，0.5Mbps)にトランスコードし，DASH ストリーミング用の Media Presentation Description (MPD)ファイルを作成する．

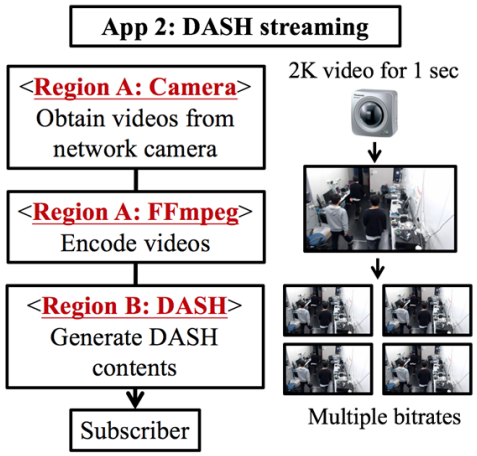


図 7. アプリケーション 2 （DASH 配信）

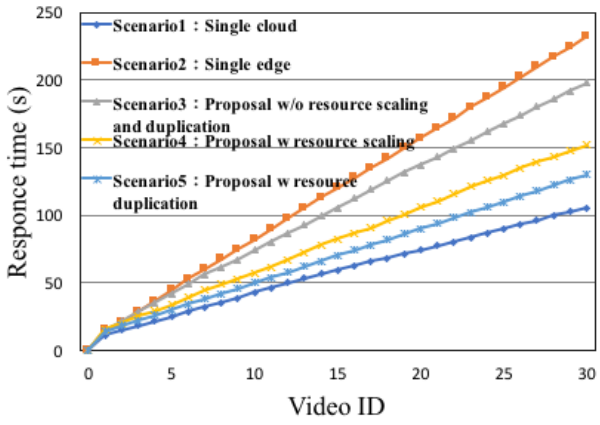
4.3 実験シナリオ

本実験では，エッジクラウド内でのリソース操作やアプリケーション要求の組み合わせによる遅延特性の変化を表 3 のシナリオで評価する．評価する遅延は，1 秒のセグメントの処理結果をユーザが 30 秒分要求する際の要求開始からセグメント到着までの時間とする．シナリオ 1 では，クラウドサーバ(Cloud) 1 台でアプリケーション処理を実行する．シナリオ 2 では，エッジクラウドと同じリージョン内に存在するエッジサーバ(Edge) 1 台でアプリケーション処理を実行する．シナリオ 3 では，エッジクラウド内の 3 台のインスタンスで機能分割された 3 つの処理機能(Camera, FFmpeg，Detect/DASH)をそれぞれ実行する．ここで，オーケストレータの機能により，アプリケーション 1 に関して，Detect 処理が遅延のボトルネックとなっていることを検知し，スケジューリングにより，シナリオ 4 ではメモリ量 2 倍のインスタンス(4GB→8GB)で Detect 処理を実行(リソーススケーリング)，シナリオ 5 ではこれまで Detect 処理を実行していたインスタンスを複製し，2 台のインスタンスで Detect 処理を実行(リソース複製)する．

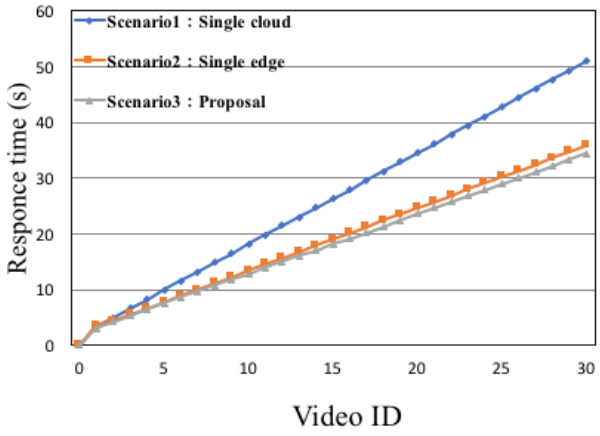
表 3. 評価シナリオ

Scenario	Instance	Resource mgmt.
1	Cloud	–
2	Edge	–
3	Camera->FFmpeg ->Detect/DASH	–
4	Camera->FFmpeg ->Detect*	メモリ増設 (Detect)
5	Camera->FFmpeg ->Detect*	複製 (Detect)

ⓒ2018 Information Processing Society of Japan



(a) アプリケーション 1 要求時



(b) アプリケーション 2 要求時

図 8. 実験結果

4.4 実験結果

図 8(a)より，アプリケーション 1 を要求した場合，遅延が短い順に「クラウド 1 台 ＜ 提案手法 ＜ エッジ 1 台」となることがわかる．これは，アプリケーション 1 では Detect 処理に高度な計算資源を要するため，今回の評価環境でもっとも高度な計算資源を保有しているクラウドで最も低遅延処理が可能となっている．しかし，提案手法ではクラウドに比べて計算資源が少ないにも関わらず，シナリオ 5 のようにリソースマネジメントを施すことで，クラウドの遅延に近い時間で処理が可能となっている．一方，図 8(b)より，アプリケーション 2 を要求した場合，遅延が短い順に「提案手法 ＜ エッジ 1 台 ＜ クラウド 1 台」であることがわかる．これは，アプリケーション 2 では各処理に高度な計算資源を必要とせず，提案手法およびエッジ 1 台でエッジコンピューティングによる通信遅延の削減効果の影響により，クラウドに比べて低遅延処理を実現できているためである．

5. まとめと今後の課題

本稿では，エッジクラウドを活用したマルチメディア処理システム上で複数のマルチメディアアプリケーションを実行した際の遅延特性を評価し，実際のクラウド環境との比較を行った．エッジクラウドを OpenStack により構築し，オーケストレータと連携して，マルチメディアサービスの機能分割，マルチメディアサービスファンクションチェイニングといった技術を導入した．これらにより，ユーザの要求・実行環境に応じたリソース操作に加え，マルチメディア処理の機能共存やデータ再利用，並列分散などといったリソース操作を可能とするマルチメディア処理システムを実現した．その結果，本システム上においてアプリケーション実行遅延の削減が可能であることを示した．

今後は，ユーザからの要求振り分けや処理インスタンスの割り当てを動的に行うためのアルゴリズムの検討を行い，他の技術と合わせてより大規模かつ複雑な実環境における提案システムの遅延評価を行う予定である．

謝　辞

本研究成果は，平成 29 年度総務省委託研究 研究課題 VI「IoT 機器増大に対応した有無線最適制御型電波有効利用基盤技術の研究開発」技術課題ア「有無線ネットワーク仮想化の自動制御技術」，および JSPS 科研費 15H01684, 17K12681 の支援を受けている．

参考文献

[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2016-2021, [online]: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html
[2] ETSI Multi-access Edge Computing [online]: http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing
[3] OpenStack [online]: https://www.openstack.org/
[4] H. Kim, N. Feamster, “Improving Network Management with Software Defined Networking,” IEEE Communications Magazine, vol.51, no.2, pp.114-119, Feb. 2013.
[5] ETSI GS NFV 001: “Network Functions Virtualization (NFV); Use Cases,” [online]: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf
[6] B. Han, V. Gopalakrishnan, L. Ji and S. Lee, B, “Network Function Virtualization: Challenges and opportunities for innovations,” IEEE Communications Magazine, vol.53, no.2, pp.90-97, Feb.2015.
[7] K. Imagane, et al., “Performance Evaluations of Multimedia Service Function Chaining in Edge Clouds,” IEEE CCNC2018, Jan.2018.
[8] FFmpeg, [online]: https://ffmpeg.org/
[9] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779-788, Jun.2016.
[10] I.Sodager, “The MPEG-DASH Standard for Multimedia Streaming Over the Internet,” IEEE Computer Society, vol.18, Issue4, pp.62-67, Apr.2011.

ⓒ2018 Information Processing Society of Japan