2021年4月　年4回季刊発行

NTT DOCOMO

テクニカル・ジャーナル

Technical Journal

Vol.29 No.1 | Apr. 2021

NTT docomo

今後のドコモR&Dの活動

R&D戦略部　部長
岡川（おかがわ） 隆俊（たかとし）

現代の社会は，少子高齢化による労働力減少，自然災害の発生，産業競争力の低下や，新型コロナウイルス感染拡大防止のためのリモート型社会への対応など，社会課題が多様化・深刻化しています．これらの社会課題解決に向けて，世の中のデジタルトランスフォーメーション（DX：Digital Transformation）*1は急速に進んでおり，これに対応してNTTもIOWN構想［1］を提唱しています．DXの推進，IOWN構想の実現に向けて，ドコモR&Dイノベーション本部では「サイバー・フィジカル融合［2］」というフレームワークを軸に研究開発を推進しています．サイバー・フィジカル融合は，現実世界（フィジカル空間）のヒト・モノ・コトの情報化，サイバー空間へのデータ蓄積，そのデータの分析による未来予測・知の発見などの価値化，フィジカル空間への価値のフィードバックというループを回していくものですが，その実現のためには，コア技術となる①AI，②ネットワーク，③デバイスをそれぞれ進化させるとともに，これらの技術の密な連携が必須となります．

①AIは，獲得・蓄積したデータをサイバー空間上で分析して未来予測・知の発見を実現するものであり，ドコモではこれまでに，携帯電話ネットワークの運用データから作成されるモバイル空間統計にAIを適用して，交通渋滞の発生やその規模・時間帯などを予測する「AI渋滞予知」や，バイクシェアの自転車集配再配置最適化などの取組みを進めてきました．

②ネットワークは，サイバー空間とフィジカル空間のコネクティビティを実現するものであり，ドコモでは第5世代移動通信システム（5G）の発展（5G Evolution）およびその先の第6世代移動通信システム（6G）へと続く移動通信ネットワークの進化に加えて，これまで独立発展してきた固定・移動のネットワークを融合させた「移動固定融合ネットワーク」の実現を加速させ，IOWN構想を含む次世代ネットワークの実現を目指したいと考えています．

③デバイスは，現実世界の顧客との接点となるものであり，ドコモではXR技術を用いたグラス型デバイスに，特に注力しています．これまでさまざまな要素技術の研究開発に取り組み，HMD（Head Mounted Display）を利用した，8KVR（Virtual Reality）（全天球8K映像）やVolumetric Video*2といった今までにない技術を積極的に生み出してきました．これらを実装し，イベントへの活用など新しい体験をお客様に提供できるようデバイスを発展させていきます．

技術を別の切り口で分類すると，基盤としてのネットワーク技術と，その上でお客様にサービスを提供するためのサービス技術に分けられますが，これらの密な連携が重要となります．また技術開発だけでなく，他社との競争激化や不確実性の高い社会への柔軟な対応のため，今後より一層重要となるサービス創出力・開発力も強化していきます．さらには，5Gネットワークとサービスプラットフォームを繋ぐ分散・集約型コンピュータ基盤であるドコモオープンイノベーションクラウドを進化させ，実社会フィールドでの実装を進めます．

①～③のコア技術をそれぞれ進化させてこれらを融合し，サイバー・フィジカル融合のループをまわすには，社内の他部門やNTTグループ間の連携およびパートナーとの連携強化が重要になります．私たちR&D部門は，法人部門およびスマートライフ事業部門と連携して，サービス創出力・開発力を強化します．また，将来の移動固定融合ネットワークの具現化や5Gのさらなる高度化と6Gに向けた研究開発を，NTTグループで連携して推進するとともに，オープンな無線アクセスネットワーク（O-RAN（Open Radio Access Network）*3/vRAN（virtualized Radio Access Network）*4）の海外展開を目的として立ち上げた「5Gオープンエコシステム［3］」の推進など国内外のベンダなどとの連携拡大，さらなる国際標準化活動の強化などにより，世界規模の研究開発を推進し続けます．

文　献

［1］ 澤田 純，井伊 基之，川添 雄彦：“IOWN構想－インターネットの先へ，”NTT出版，2019.
［2］ 谷 直樹：“新たな事業価値を生み出し続けるR&D，”本誌，Vol.28，No.4，p.1，Jan. 2021.
［3］ NTTドコモ報道発表資料：“（お知らせ）海外通信キャリアに最適なオープンRANを提供する「5Gオープンエコシステム」を協創，”Feb. 2021.

＊1 デジタルトランスフォーメーション（DX）：IT技術を活用してサービスやビジネスモデルを変革させ，事業を促進するとともに人々の生活をあらゆる面で良い方向に変化させること．
＊2 Volumetric Video：専用装置で撮影された3D立体映像で，自由視点視聴やインタラクティブな映像表現が可能である．
＊3 O-RAN：機能拡張性向上のためのオープン化インタフェースで構成されるO-RAN Alliance準拠の無線アクセスシステム．
＊4 vRAN：無線アクセスネットワークに対して，汎用プロセッサやアクセラレータなどを用いた仮想化技術を活用することで，よりオープンで柔軟性が高い形で実装する無線アクセスシステム．

NTT DOCOMO
テクニカル・ジャーナル
Technical Journal
Vol.29 No.1 Apr. 2021

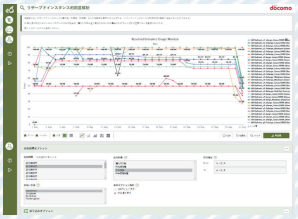
[Contents]

DOCOMO Today

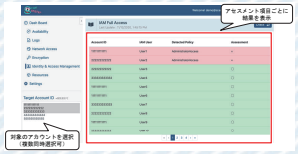
特別寄稿

パブリッククラウド活用特集
Technology Reports（特集）

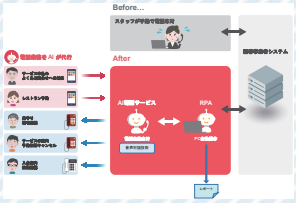
(P.13)



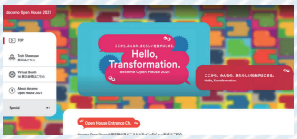
(P.23)



(P.32)



(P.60)

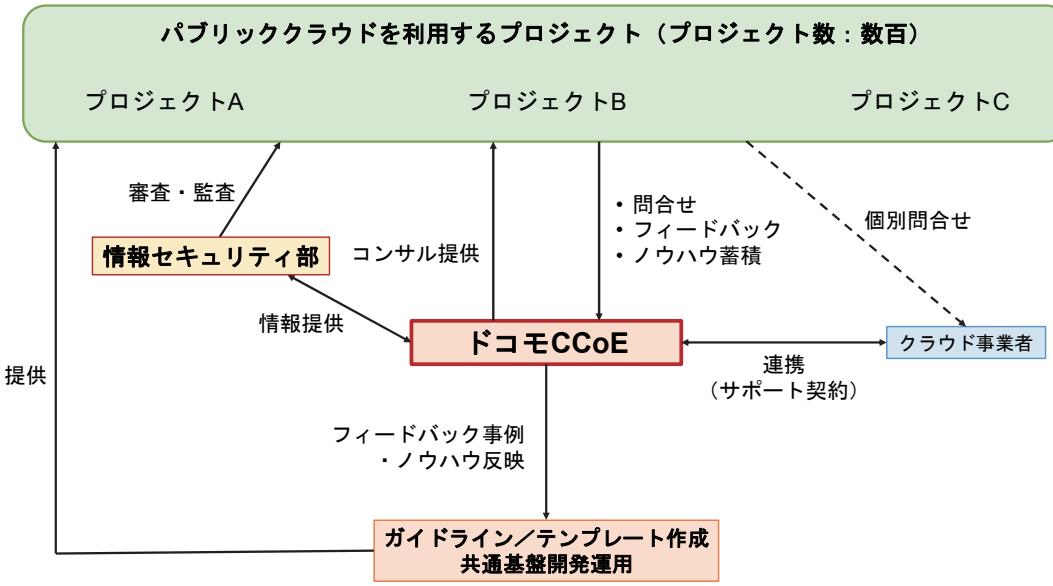


(P.67)

Technology Reports

Event Reports

News



Technology Reports（特集）　ドコモのパブリッククラウド活用とCCoEの果たす役割（P.6）
ドコモのパブリッククラウド利用体制

特別寄稿

空間伝送型ワイヤレス給電

京都大学　生存圏研究所　教授　篠原（しのはら）　真毅（なおき）さん

第5世代移動通信システム（5G）は，単なる電話とネット接続の装置ではなく，高速大容量・低遅延・多数端末同時接続という特長を活かしたさまざまな新しい応用を生み出すものとして生まれ変わろうとしている．遠隔での医療・教育，車の自動運転やAR（Augmented Reality）・VR（Virtual Reality），そしてすべてを無線でつなぐIoT（Internet of Things）のプラットフォームとして5Gは発展を期待されている．さらにBeyond 5G[*1]向けの取組みがすでに始まっており，「情報の伝送」というこれまでの無線技術に加え，情報機器（のみならずすべての最先端機器）には必須の電力すら無線で供給する「ワイヤレス給電」技術にも近年注目が集まっている．これまでも携帯電話にはワイヤレス給電が応用されており，Appleも参加する非接触給電のQi規格[*2]が，携帯電話の「置くだけ充電器」として現在世界で広く普及している．しかし，置くだけ充電器は電力を非接触（＝無線）で送り，携帯電話に給電線用の穴をあける必要がなくなるという利点はあるものの，結局充電台に携帯電話を置かなければならない（＝有線充電との差がつきにくい）という課題があった．そこで無線通信と同じように離れた距離でも電波で携帯電話を充電できるほか，IoTセンサへのワイヤレス給電などにも応用できる「空間伝送型ワイヤレス給電」が特に注目されており，研究開発とともに法制化／標準化がさかんとなっている．

空間伝送型ワイヤレス給電は古くはマイクロ波送電と呼ばれ，1960年代より米国で研究開発が始まり，日本でも1980年代より大学を中心に研究開発が行われていた．しかし当時の空間伝送型ワイヤレス給電はナロービーム型と呼ばれる，1つの受電器にビーム状のマイクロ波電力を集中させて用いるシステムがほとんどであり，ビジネス的に発展できなかった．しかし近年，RF-ID（Radio Frequency IDentification）[*3]を始めとする，複数の低消費電力デバイスへ広くワイヤレス給電を行うワイドビーム型のニーズが高まり，米国では2000年代ごろからPowerCAST社，Ossia社などのスタートアップ企業が設立され，近年さらにその数を増やし，Energous社，AETERLINK社，Wi-Charge社（レーザー送電）なども空間伝送型ワイヤレス給電の商品開発に取り組んでいる．PowerCAST社やOssia社，Energous社は2017年ごろに米国連邦通信委員会（FCC：Federal Communications Commission）から920MHz帯や2.45GHz帯の周波数の使用許可を取り，米国内での商品販売を始めている．例えば，PowerCAST社の商品の一部であるNintendo Switchのワイヤレス充電製品は，米国Amazon.comで購入できる．

一方我が国では，ワイヤレス給電の製品化において米国に後れを取っているが，2010年に電子情報通信学会通信ソサイエティの第2種研究会として無線電力伝送研究会を国内で立ち上げ，2014年に第1種研究会に改組し，学術的な研究を加速させてきた．また筆者が，米国IEEE（Institute of Electrical and Electronics Engineers）マイクロ波ソサイエティ（MTT-S：Microwave Theory and Techniques Society）の国際学会としてIEEE WPTC（Wireless Power Transfer Conference）を2011年に立ち上げ，その後の運営の中心的役割を果たし，現在も日本がIEEEのWPT関連学会活動をリードしている．筆者はまたIEEE MTT-SのDistinguish Microwave Lecturerとして，2016～2018年の3年間で55回以上の海外での講演も行ってきた．これらの日本発の学会活動がワイヤレス給電の標準化活動を活発化させ，日本の総務省とワイヤレス給電の標準化団体ブロードバンドワイヤレスフォーラム（BWF：Broadband

Profile

1991年京都大学工学部電子工学科卒業．1996年同大学大学院工学研究科博士課程修了．同年・同大学超高層電波研究センター助手を経て，2010年同大学教授となり現在に至る．IEEE MTTS TC-26前Chairman，URSI Commission D Vice Chairman，電子情報通信学会WPT研究会初代委員長，宇宙太陽発電学会副理事長，内閣府宇宙政策委員会臨時委員，（独）日本学術振興会第24期URSI分科会特任連携会員，ワイヤレス電力伝送実用化コンソーシアム代表他．

Wireless Forum）とが中心となって，2013年ごろから国際電気通信連合無線通信部門（ITU-R：International Telecommunication Union-Radio communication sector）でのワイヤレス給電に関する寄与文書を提出し続け，世界の議論を牽引してきた．2016年には空間伝送型ワイヤレス給電として初めて，ITU-Rのレポート（Report SM.2392）が採択，出版され，その後このレポートを引用する形で，特にワイドビーム型ワイヤレス給電の議論が活発に行われている．

空間伝送型ワイヤレス給電の法制化の議論はITU-Rが先行していたが，国内での法制化も始まっている．総務省は，2018年12月に「空間伝送型ワイヤレス電力伝送システムの技術的条件」について情報通信審議会へ諮問し，法制化に向けた検討が始まった．約2年の議論ののち，2020年7月14日に情報通信審議会から「構内における空間伝送型ワイヤレス電力伝送システムの技術的条件」に関する一部答申があり，本答申を踏まえ，総務省は速やかに制度整備などを行う予定となったのである．答申では920MHz帯，2.4GHz帯，5.7GHz帯の3つの周波数帯での空間伝送型ワイヤレス給電の技術的条件が示された．この動きを受け，日本初の空間伝送型ワイヤレス給電のスタートアップ企業であるスペースパワーテクノロジー社や，パナソニック，オムロン，東芝などの企業が商品化を目指し研究開発を行っている．

これら法制化の議論や企業の商品開発のサポートは，筆者が代表を務めるワイヤレス電力伝送実用化コンソーシアム（WiPoT：Wireless Power Transfer Consortium for Practical Applications）が大きな役割を果たしている．WiPoTは2013年に設立されて，2021年1月現在で法人会員43社，学識会員58名，4研究機関で活動を行っている．主に，海外企業や国内企業の紹介やビジネスマッチング，国内外の学会との連携，法制化／標準化の議論への参加（BWFと協力）などを行い，空間伝送型ワイヤレス給電の実用化の中心的役割を担ってきたと自負している．

最後に気になる中国であるが，中国ではワイヤレス給電に関する目立ったスタートアップ企業はないようである．しかし，2011～2020年3月の約10年間での空間伝送型ワイヤレス給電に関する特許を調べたところ，世界で約3,000件あり，その割合は中国が約36％，米国が約31％，日本が約25％，韓国が約7％，その他1％前後となっており，他の技術同様，中国の存在感が突出している．また，ナロービーム型ワイヤレス給電の究極の応用として，1970年代より研究されている宇宙太陽発電に関する研究が中国では非常に活発となっている，西安，重慶，四川，武漢などの大学，研究機関に集中的に研究資金が投資され，ナロービーム型ワイヤレス給電実証実験もさかんに行われている．2021年1月にはXiaomi社よりミリ波ビーム型ワイヤレス給電を用いた携帯電話の無線充電システムの発表があった．

2021年は，空間伝送型ワイヤレス給電元年になる可能性が非常に高い．日本はまだこの分野において世界の研究開発や標準化の議論をリードしているが，企業活動ではすでに米国に出遅れており，中国の膨大な研究費の集中投下に見劣りし始めている．2021年が日本敗戦の始まりとならぬよう，先手を打ってワイヤレス給電技術を伸ばし，世界をリードし続けねばならない．

＊1　Beyond 5G：「5Gの次に」「5Gの先に」という意味．
＊2　Qi規格：Wireless Power Consortiumが定める電磁誘導型ワイヤレス充電の国際規格．
＊3　RF-ID：ICタグ，無線タグ，RFIDタグなどとも呼ばれる電波を用いてRFタグのデータを非接触で読み書きするシステム．近年アパレルなどの商品タグに広く用いられるようになった．

Technology Reports（特集）　パブリッククラウド　CCoE　サーバレス
パブリッククラウド活用特集

ドコモのパブリッククラウド活用とCCoEの果たす役割

イノベーション統括部　守屋 裕樹（もりや ひろき）　森 健史（もり たけし）
DOCOMO Innovations, Inc.　直井 康広（なおい やすひろ）

ドコモでは，10年以上前からパブリッククラウドを用いた数多くのサービスを提供している．パブリッククラウドの活用においては，CCoEが中心となってより効率的かつ適切に行われるように活動を続けてきている．本稿では，それらの活動やドコモにおけるパブリッククラウド利用体制を，昨今のクラウドにおける開発トレンドと合わせて解説する．

1. まえがき

企業や組織におけるパブリッククラウド*1の利用が急速に増加しており，「クラウドファースト」という言葉で代表されるように，パブリッククラウドを活用した事業展開が一般的になってきている．同時に多くの企業や組織にとって，オンプレミス*2にはないパブリッククラウドの迅速性や柔軟性をうまく活用して自社のビジネスを展開，あるいは変革していく必要が出てきている．しかし，パブリッククラウドの活用においては，従来からあるオンプレミス型のITシステムの構築や運用とは違った考え方が求められることも多く，スムーズな導入や活用ができない企業や組織も少なくない．

一般的にパブリッククラウドの利用にあたっては，導入時のサービス検証，セキュリティ対策，利用ポリシーの策定，社内における利用体制の確立，スキルの獲得，人材育成，変化の早いクラウドの情報収集やノウハウの展開といった課題に対応していくこととなる．パブリッククラウドをうまく活用するためには，従来からあるオンプレミス型のITシステムの考え方との違いを認識し，変化の早いパブリッククラウドに積極的に追従していくことが重要となる．

ドコモにおいても，10年以上前から数多くのサービスでパブリッククラウドを活用している．それを通じて，前述したような課題にも向き合い，より効

＊1　パブリッククラウド：インターネットを介して誰でも利用ができるクラウドコンピューティングサービス．
＊2　オンプレミス：企業がシステムを構成するハードウェアを自社で保有し，自社で保守運用すること．

率的かつ適切にパブリッククラウドを活用するための活動を続けてきた．本稿では，ドコモで実施してきたそれらの活動やドコモにおけるパブリッククラウド利用体制を解説する．

2. ドコモにおけるパブリッククラウド利用

2020年12月時点での，ドコモにおけるパブリッククラウドの利用状況を解説する．

2.1 パブリッククラウドの利用規模

ドコモでは，2009年ごろに研究開発・検証目的でパブリッククラウドの利用を開始した．その後，利用範囲を拡大し，2012年から大規模商用サービスで，パブリッククラウドを採用した．利用開始時から毎年利用量は増加しており，2020年12月時点では，900以上のAWS（Amazon Web Services）*3のアカウント，250以上のGCP（Google Cloud Platform）*4のプロジェクト，50以上のMicrosoft Azure*5（以下，Azure）のサブスクリプションをそれぞれ運用している．

2.2 パブリッククラウドの活用範囲

ドコモにおけるパブリッククラウドの活用範囲は，多岐にわたっている．例えば，Webサービス，モバイルアプリケーションのバックエンドシステム*6，データ分析基盤，機械学習，社内向けシステムなどの多くの分野でパブリッククラウドを活用している．

2.3 パブリッククラウドの利用体制

ドコモでは，パブリッククラウドを効率的に活用できるように考慮した利用体制を整えている（**図1**）．ドコモにおけるパブリッククラウド利用で最も特徴

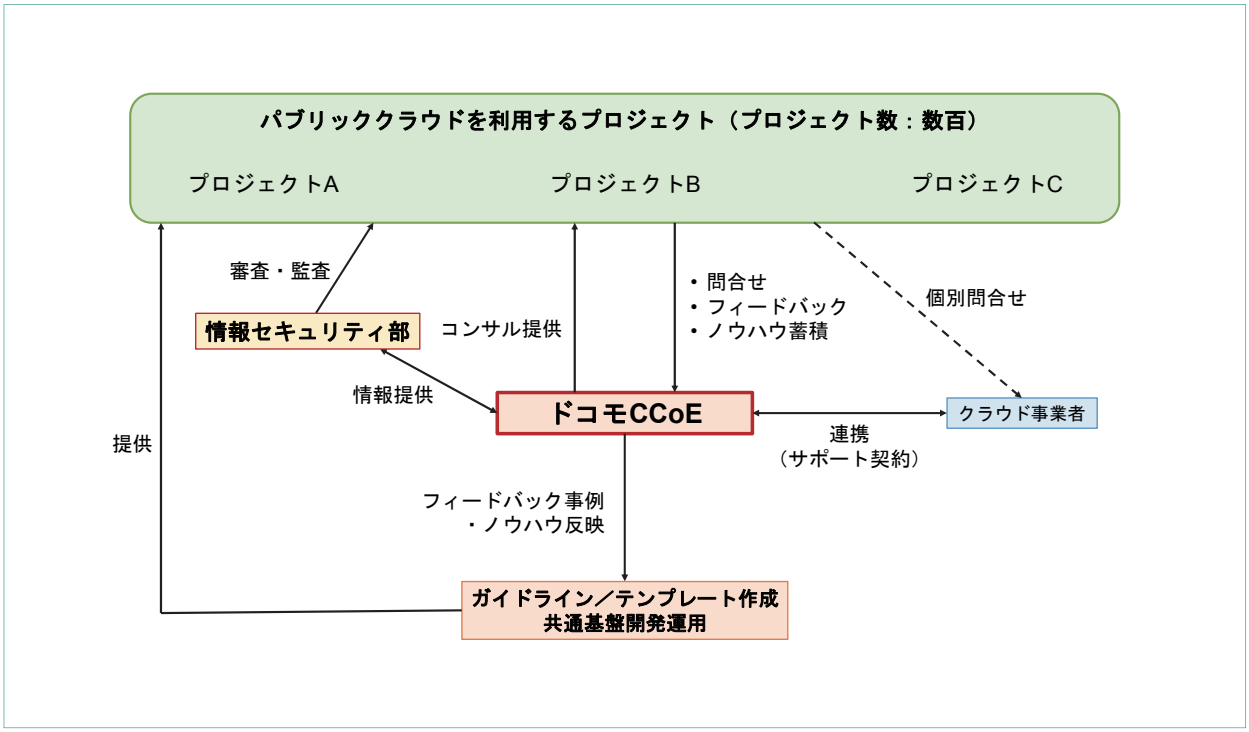


図1　ドコモのパブリッククラウド利用体制

＊3　AWS：Amazon Web Services社が提供するクラウドコンピューティングサービス．
＊4　GCP：Google社が提供するクラウドコンピューティングサービス．
＊5　Microsoft Azure：Microsoft社が提供するクラウドコンピューティングサービス．
＊6　バックエンドシステム：利用者のモバイル端末やコンピュータで動作するのではなく，サーバなどで集中的に動作させるシステム．

的な点は，明確なCCoE（Cloud Center of Excellence）[*7]が存在することである．CCoEとは，パブリッククラウドに関する多用な専門知識をもったチームであり，ドコモではこのCCoEを中心にパブリッククラウドの活用を進めてきている．

3. ドコモCCoEの活動

ドコモのパブリッククラウド利用体制において中心的な役割を担っているCCoEの活動の詳細について解説する．

3.1 コンサルティングの提供

パブリッククラウド利用時の支援はCCoEの重要な役割である．社内のプロジェクトにおいて，CCoEはパブリッククラウド上でシステムを構築する際のシステム設計支援やレビュー，セキュリティ要件を満たすための効率的な対策方法の提示などを行う．特にパブリッククラウドにおいては，最初の設計が重要で，後のコストやシステム運用業務に大きな影響を与えるため，システム設計時に支援を行うことが多い．

また，システムの運用を開始して，システムの利用者が増えてくると必然的にコストも増大していく．その際にコスト最適化を行うために，コスト要因の把握や設計見直しなどの支援も実施している．

3.2 クラウドコストの取りまとめと最適化

一般的に，パブリッククラウドは柔軟な利用ができる一方で，サービス数や購入オプションの拡大に伴い，費用支払いや会計処理などが煩雑になりやすく，プロジェクトが増えた際に，稼働が増大してしまう場合がある．そこでドコモでは，CCoEが，一元的にクラウド事業者との契約を行い，一括で各プロジェクトの支払いを取りまとめて処理をすることで，各プロジェクトの負担を軽減させている．

また，一括支払いには，事務処理負担の軽減のほかにもメリットがある．パブリッククラウド事業者によっては，利用量が多くなるほど，価格が安くなるボリュームディスカウントオプションがある．支払いをまとめることで，ボリュームが確保され，社内全体で利用料を抑えることが可能になる．

3.3 最新情報の収集と展開

パブリッククラウドの進化は早く，最新の情報を常に追いかける活動は，プロジェクト業務を抱える各プロジェクトのメンバでは難しい場合がある．そこで，最新情報の取得もCCoEにて率先して実施している．例えば，パブリッククラウドに関する技術的なイベントには積極的に参加して，情報収集や発表を行っている．また取得した情報は，ノウハウとしてガイドラインへ反映したり，自ら検証を行い，社内プロジェクトメンバへ展開したりするようにしている．

3.4 クラウド業務支援ツールの作成と展開

各プロジェクトがパブリッククラウドを効率的に活用できるように，さまざまな支援ツールを作成して，社内に展開している．

クラウドでは事業展開の速度を向上させる多様な機能が提供されているが，一方で使い方を誤るとクラウドの障害に伴うサービス停止や，設定誤りによるセキュリティ事故を引き起こす可能性がある．そのため，利用者には正しくクラウドを利用するための技術知識が必要となるが，利用者の知識レベルはまちまちであり，会社としては，社員全体の技術知識の底上げが課題となる．

このためCCoEは，ドコモ視点でのクラウド利用方法の記載を特徴とするガイドラインを作成している．本ガイドラインにより，仮に知識の乏しい利用者であっても，押さえておくべき最低限の知識が短時間で習得可能となる．ガイドラインはAWS，GCP，

＊7 CCoE：企業においてクラウドの活用を成功させるために，ベストプラクティスの確立や必要な制度，ガバナンスなどを作成し，社内に展開していく専属のチーム．

Azureに対応している．ここではAWSを対象として作成したガイドライン一覧を**表1**に，ガイドラインの一部抜粋例を**図2**に示す．

また，ガイドラインの1つであるセキュリティデザインパターン*8では，クラウドを利用してシステム構築する際のセキュリティ考慮漏れを抑制できるよう，情報セキュリティ部のセキュリティチェック項目に沿った要件（ISO/IEC27017やJISQ27002など）と，クラウド環境における要件，AWSを用いてシステム構築を行う場合の要求仕様とAWSの提供機能・サービスを並記することで，セキュリティチェック項目への準拠性を高めている．システム構築担当者はセキュリティチェック項目を満たすために一点ずつ検証を行う必要はなくなり，利用に応じ

表1　ガイドライン一覧（AWS）

No	種　類	内　容
1	クラウド開発ガイドライン	クラウドを使う場合の考え方や作法，開発フローにおける各フェーズで考慮・実施すべき指針を記載，特に設計・セキュリティなどは重点的に網羅し，間違った使い方を抑止
2	セキュリティデザインパターン	クラウドを利用し，システムを構築するにあたり，セキュリティの考慮漏れを抑制．ISO/IEC27017に沿った要件をあらかじめ用意することで，ISO管理策に対する準拠性を高めている．AWSを利用したシステムを構築するにあたり，必要となるセキュリティ要件を列記
3	セキュリティテンプレート	セキュリティデザインパターンを考慮したネットワーク構成やネットワークフィルタリング機能，基本機能を提供するインスタンス群などを生成するAWS CloudFormationテンプレートを記載，セキュリティ対策を容易に実施
4	IAMデザインパターン	AWSアカウント利用パターンに合わせて，ドコモ社内のIAMポリシー設計のベストプラクティスを記載
5	インシデント対応ガイドライン	AWSを利用する，インターネットに公開されているサービス提供システムや，社内システムなどでサイバー攻撃などのインシデントが発生した場合の対応について，実際に発生した事例を含めて記載
6	コスト最適化ガイドライン	AWSコストを管理する者向けに，コストの把握・分析方法，コスト削減・最適化の手法を記載
7	システム移行ガイドライン	オンプレミスからAWSへの移行案件をスムーズに行うために，これまでの移行事例で得た知見などを基に移行時にポイントとなる点や注意すべき点を記載
8	共通基盤化ガイドライン	複数アカウント／複数システムを運用し始めると，運用を統一化したり，運用システムを共通化することで効率化を図ると有効な場合があり，クラウドの特性を利用することで，運用の効率化をスムーズに進める方法を記載
9	コンテナガイドライン	コンテナをサービス開発／運用に取り入れようとしている人向けに，各プロジェクトで効果的，安全にコンテナ活用を実践してもらうことを目的に，どういったツールをどのように利用するのがよいか記載
10	サーバレスガイドライン	AWS上でサーバレスなシステムを開発・運用する際に，どのように導入を進めていけばよいのか，どういったサービスをどのように利用するのがよいのか，各プロジェクトで効率的かつ効果的にサーバレスを活用する方法を記載
11	DevOpsガイドライン	DevOpsの考え方をサービス開発／運用に取り入れようとしている者向けに，どういったツールをどのように利用するのがよいのか，各プロジェクトで効率的かつ効果的にDevOpsを実践してもらうことを目的としたガイドライン

IAM：Identity and Access Management
IEC：International Electrotechnical Commission
ISO：International Organization for Standardization

＊8　セキュリティデザインパターン：セキュリティ要件に対して，パブリッククラウドを利用した際の実現方法を記載したガイドラインの1種．

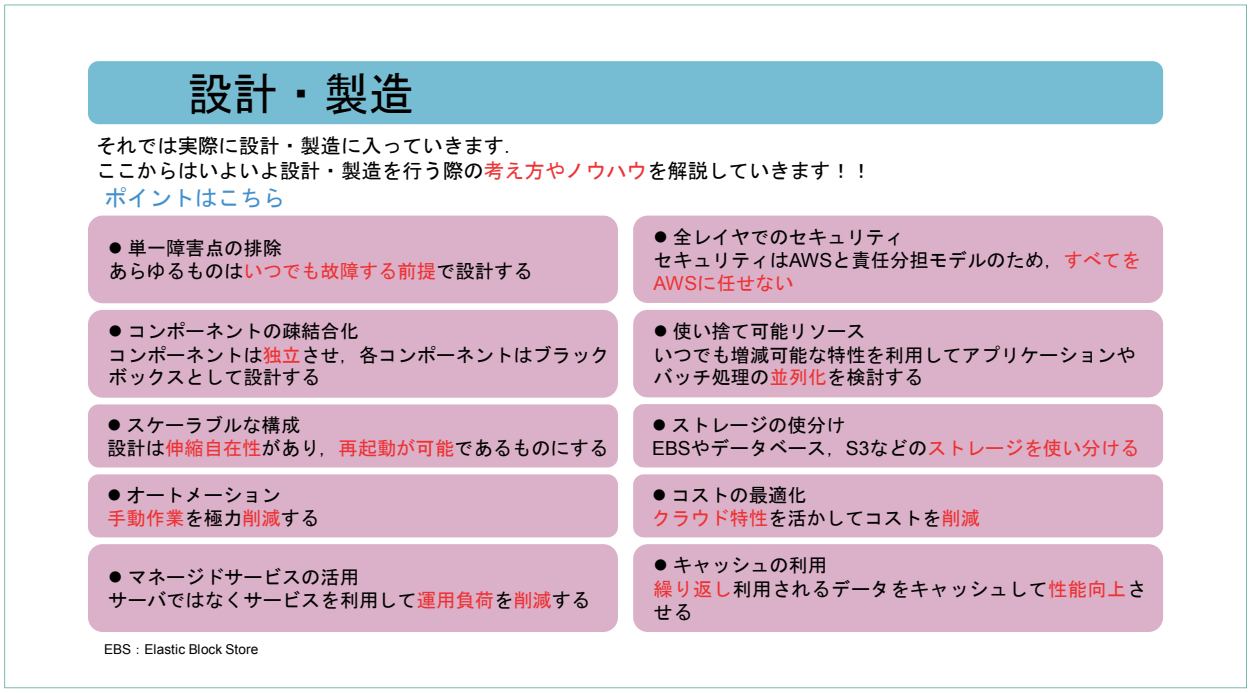


図2　ガイドライン例（クラウド開発ガイドラインの内容抜粋）

てセキュリティデザインパターンを参照しながらシステム構築を実施すれば，おのずとセキュリティチェック項目を満たせるようになる．セキュリティデザインパターンの一部抜粋例を**図3**に示す．

パブリッククラウドの機能追加やアップデートのスピードは早いため，これらの支援ツールも頻繁にアップデートをする必要がある．そのため，CCoEでは，これらのツール開発においてもアジャイル開発[*9]を取り入れ，頻繁な機能追加やアップデートに追従していく活動も行っている．

4. ドコモにおける最新の開発トレンド

ここでは，ドコモの最新の開発現場から，サーバレス技術を適用した最新の開発事例を解説する．

4.1　サーバレス技術の登場

クラウドの世界ではこれまで別々だった開発とオペレーションを分離せず，それらを統合して生産性を高めるいわゆるDevOpsの流れとともに，従来の仮想マシン[*10]からコンテナ[*11]の利用を前提とした方式設計が普及した．コンテナ化においては開発やリリースサイクルを加速させることに成功したものの，依然として従来の仮想マシンと同様に運用監視やセキュリティのための保守運用が必要である．コンテナ化の流れの一方で，AWSをはじめとしたパブリッククラウド事業者は，ミドルウェア[*12]からサーバの運用管理までを含む一連のマネージドサービス[*13]を「サーバレス」技術としてすでに提供し始めている．これはコンテナ化とは別の軸で発展しているクラウドインフラ利用技術であり，開発者や運用管理者はサーバの存在を意識することなくインフラの運用管理をすべてクラウド事業者に任せ，より多くのリソースをそのビジネスロジックの開発に集中することが可能である．

＊9　アジャイル開発：迅速かつ適応的にソフトウェア開発を行う軽量なアプリケーション開発手法群の総称．
＊10　仮想マシン：ソフトウェアによって仮想的に構築されたサーバなどのコンピュータ．
＊11　コンテナ：コンピュータ仮想化技術の一種で，1つのホストOSの上にコンテナと呼ばれる専用領域を作り，その中で必要なアプリケーションソフトを動かす方式のこと．
＊12　ミドルウェア：複数のアプリケーションから共通に利用される機能を提供するソフトウェア．

ISO/IEC27017またはJISQ27002		AWSセキュリティデザインパターン		
章番号	管理策	クラウド環境における要件	AWSを用いてシステム構築を行う場合の要求仕様とAWSの提供機能・サービス	
12.1.4	開発環境，試験環境および運用環境は，運用環境への認可されていないアクセスまたは変更によるリスクを低減するために，分離することが望ましい．	・クラウド環境に構築するシステムは，商用，開発，試験，バックアップ環境ごとに分離した構成とすること -可能であれば契約を分離する -ネットワークを分離する など ・クラウドはインターネットにさらされた構成となるが，ファイアウォールやルーティングにより論理的にネットワークを分離し，役割に応じたネットワークセグメントを構成できること ・マルチテナント環境のサービスの場合，他の利用者や環境（商用，開発，など）に影響を受けず利用を分離可能であることを確認する -物理的に分離可能 -論理的に分離可能	**【要求仕様】** ・商用，開発，試験，バックアップなどの環境を分離する際は，AWSのアカウントを別にする．難しい場合はVPCで分離する． ・役割（DMZ，リモート接続ネットワークセグメント，内部サーバネットワークセグメント，など）に応じたネットワークセグメントをVPC機能やサブネット機能で分離する．	**【AWSの提供機能・サービス】** ・AWSアカウントの用途ごとの割当てにより他のテナントと環境を論理的に分け，VPC機能によりネットワーク環境を論理的に分けることができる． さらに物理的に環境を分ける要件がある場合は，以下のどちらかを利用する． -ハードウェア専有インスタンス -Dedicated Hosts ・サブネット機能により，VPC内に複数の仮想サブネットを作成できる．それにより役割に応じたサブネットを構成することができる． ・VPCピアリングを利用することで2つのVPCを同じネットワーク内に存在するかのように扱うことができる． **【備考】** ・Multi-AZ構成にすることで可用性を向上できる．
13.1.3	情報サービス，利用者および情報システムは，ネットワーク上で，グループごとに分離することが望ましい．			
14.1.2	公衆ネットワークを経由するアプリケーションサービスに含まれる情報は，不正行為，契約紛争，並びに認可されていない開示および変更から保護されることが望ましい．	機能要件		
14.1.3	アプリケーションサービスのトランザクションに含まれる情報は，次の事項を未然に防止するために，保護されることが望ましい． －不完全な通信 －誤った通信経路設定 －認可されていないメッセージの変更 －認可されていない開示 －認可されていないメッセージの複製または再生	機能要件		

DMZ：DeMilitarized Zone
JIS： Japanese Industrial Standards
VPC：Virtual Private Cloud

図3　セキュリティデザインパターン例（社外向けVersionではISO27017規格をベースに記載）

4.2　ドコモでのサーバレス適用開発事例

ドコモが2020年6月にリリースしたドコモオープンイノベーションクラウドのポータルサイトは，以下のようなReact＊14ベースの一般的なウェブアプリケーションシステムである（**図4**(a)）．

ドコモオープンイノベーションクラウドの開発にあたって最初のリリースまでの期間は4カ月，そして運用・保守要員は最小限に抑えるという制約があった．そこで可能な限りサーバ管理をAWSにオフロード＊15するため，AWS Lambda＊16などを利用したサーバレスアーキテクチャを全面的に採用して開発を行った（図4(b)）．その結果，開発期間を大幅に短縮し，短期間でのリリースに成功した．

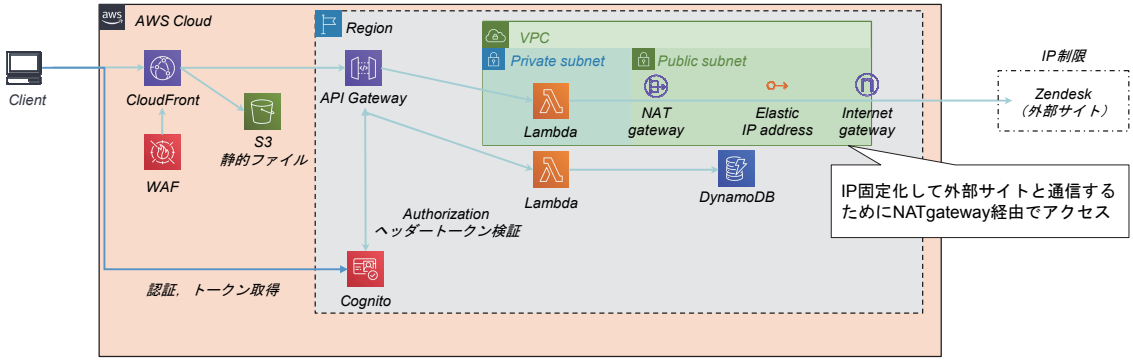
本稿執筆時点で運用開始から1年．ネットワークやサーバなどインフラを起因とした障害はゼロである．可用性においてサーバレスでは，冗長性やフォールトトレランス＊17があらかじめビルトインされている恩恵を受けている．運用・保守要員は2〜3名と当初の目標を達成した．今後，ユーザアクセス増加によるスケーラビリティの確保が必要な局面でもサーバレスであれば自動で対応可能である．料金も使用した分だけ支払う従量課金であるため，本プロジェクトでは大幅なコスト削減を実現した．

5. あとがき

本稿では，ドコモにおけるパブリッククラウドの活用とCCoEの果たす役割について解説した．パブリッククラウドの利用は今後も増えることが予想される中で，それらをうまく利用し，ビジネスを成功に導くことが重要である．そのためにCCoEは今後もそれぞれのプロジェクトがパブリッククラウドを

＊13 マネージドサービス：クラウドサービスのうち，リソースのプロビジョニングや運用の大半をクラウド事業者の責任で実施しているサービス．クラウドコンピューティングサービスのうち，特にPaaSやSaaSを指す．
＊14 React：ユーザインタフェース構築のためのJavaScriptライブラリ．
＊15 オフロード：システムやサービス，ネットワークの処理を別の同様のサービスに振り分け本来のサービスの処理を軽減すること．
＊16 AWS Lambda：AWSが提供するFaaSの1つ．アプリケーションコードの実行環境が提供されており，利用者は作成したソースコードを登録してアプリケーションが実行できる．

(a)各ウェブページのイメージ



①Client からのアクセスにおいて，ユーザ認証時に Cognito で認証トークンを発行．以降，Client は Cognito で発行された認証トークンを用いてアクセスを行う．
②Client からの要求は WAF と連携した CloudFront で受け付ける．CloudFront は静的コンテンツ要求を S3，API 要求を API Gateway に振り分ける．
③静的コンテンツの要求は S3 をウェブサーバとしてそのまま Client に配信する．
④API 要求は API Gateway 経由でバックエンドの Lambda で処理する．
⑤Lambda はユーザデータの読み書きに DynamoDB を永続ストレージとして利用，API の処理結果を Client に返す．

API Gateway：AWS 管理による API プロキシーサービス
CloudFront：AWS 管理による CDN サービス
Cognito：AWS 管理によるユーザ管理サービス
DynamoDB：キーバリューデータストアのマネージドサービス
Lambda：関数が処理単位のマネージドサービス
S3：AWS 管理によるオブジェクトストレージサービス
WAF：AWS 管理による ウェブアプリケーションファイアーウォール

(b)Web周りのシステム構成図

図4　ドコモオープンイノベーションクラウド

最大限に活用できるように支援の範囲を広げていきたい．

また，サーバレスといった最新技術を商用サービスにいち早く導入・実践することは次世代のイノベーションへの挑戦でもある．そのマインドを育てながら，今後もドコモ社内で開発の効率化を促進，成功事例を増やしていきたいと考えている．

＊17 フォールトトレランス：システムに障害が発生した場合にも正常に機能し続けること．

Technology Reports（特集）
Kubernetes AWS コンテナ
パブリッククラウド活用特集

クラウドオーケストレータを活用した複数Kubernetes管理

DOCOMO Innovations, Inc. 高田 雅人（たかだ まさと） 直井 康広（なおい やすひろ）

Kubernetesは元Googleの開発者によって開発され，現在，CNCFによりオープンソースとして公開されている．Kubernetesにより，ユーザはコンテナ化されたアプリケーションを，環境を問わず効率的に運用することが可能となり，昨今，先進的な企業をはじめとする多くの企業で活用されている．2018年ごろには1つのプロジェクトで複数のKubernetesクラスタを利用する企業も増えているが，一方で複数のKubernetesクラスタを管理するツールが存在していなかった．そこで，DOCOMO Innovations, Inc.では，複数Kubernetesクラスタの一元管理を可能とするクラウドオーケストレータをオープンソースとして開発した．本ソフトウェアは，現在，ドコモ社内の商用システム上ですでに活用されており，本稿ではその事例を解説する．

1. まえがき

2013年のDocker*1の登場は，コンテナ仮想化技術*2を活用したOSレベルでの仮想化がなされることで，アプリケーションとシステム環境が論理的に分離された．これによりワークロード*3におけるコンテナの素早いデプロイ*4やアップデートを可能とした．一方，Dockerはコンテナ管理，スケーラビリティ，自動復旧などに関する問題を抱えていた．利用者は各環境で使用する前に，これら課題を解決する必要があるが，多大な稼働が掛かっていた．

昨今，注目されているKubernetes［1］はこの課題を解決するために登場し，オープンソースソフトウェアとして提供されている．Kubernetesとは，ギリシャ語で操舵手という意味であり，その名が示すとおり，コンテナの運用管理や自動化を目的としたソフトウェアである．開発者は自前でコンテナ向けのフレームワーク*5を用意することなく，複数コン

＊1 Docker：コンテナ型仮想化ソフトウェア．Docker Inc.の登録商標．
＊2 コンテナ仮想化技術：コンピュータ仮想化技術の一種で，1つのホストOSの上にコンテナと呼ばれる専用領域を作り，その中で必要なアプリケーションソフトを動かす方式のこと．
＊3 ワークロード：CPU使用率などのシステムの負荷の大きさを表す指標．特にパブリッククラウドの分野では，クラウド上で実行されるOSやアプリケーションコードなどを含めたシステム自体を表すこともある．本稿では後者の意味で用いる．

テナの管理やオートスケール*6，自動復旧の機能など多大な恩恵を受けることができる．Kubernetesは，すでにクラウドの分野においてはデファクトスタンダードとなっており，AWS（Amazon Web Services）*7においてはEKS（Elastic Kubernetes Service）*8，Microsoft Azure*9においてはAKS（Azure Kubernetes Service）*10，GCP（Google Cloud Platform）*11においてはGKE（Google Kubernetes Engine）*12というサービス名でマネージドサービス*13が提供されており，さらにプライベートクラウド*14ベンダであるVMware，Red Hatなど多くの企業からサービスが提供されている．

ドコモ内でも2017年ごろから複数Kubernetesクラスタを利用するプロジェクトが出現し始めた．一方で複数クラスタ*15の利用に伴い，運用者は運用コストの増加やクラスタ間でのリソース使用率のばらつきの問題に直面していた．そのため，複数Kubernetesクラスタの管理を目的としたオーケストレーションツールとして，クラウドオーケストレータ（以下，CO）をオープンソースソフトウェアとしてDrupal*16上で開発した．本稿では，COにおける特長や社内の活用事例について解説する．

2. Kubernetes

2.1 概　要

Dockerは単一サーバ上で稼働するには便利なツールである一方，複数サーバで構成される大規模環境では多くの問題を抱えていた．このような大規模環境向けのコンテナ管理を目的としたコンテナオーケストレーションツールであるKubernetesが，世界的にデファクトスタンダードとして利用されている．Kubernetesは，元Googleエンジニアにより開発されたBorgが起源とされている．BorgはGoogle社内で活用されたコンテナオーケストレーションのノウハウを蓄積後，その中の主要機能がKubernetesから提供されている．Kubernetesは，その後2014年にKubernetesプロジェクトにおいてオープンソース化され，2016年にCNCF（Cloud Native Computing Foundation）*17に移管されて，コミュニティベースの開発が行われている．

2.2 アーキテクチャ

図1のとおり，Kubernetesはマスターノードとワーカーノードの2つから構成される．クラスタ内に存在するアプリケーションやその各種設定情報はリソースオブジェクトという単位で管理される．リソースオブジェクトはマニフェストファイル*18により定義され，ユーザはその新規作成や更新時に，このファイルにより操作する．Kubernetesを構成する要素は，以下のとおりである．

(1)マスターノード（Master node）

クラスタ全体を管理するノードで，ワーカーノード管理やポッド管理の役割を担う．クラスタのデプロイ時にデフォルトとして，コントロールプレーンが設定される．コントロールプレーンとは，クラスタを制御するコンポーネントを制御し，クラスタ内の状態や構成を管理するコンポーネントである．利用者は後述のkubectlを利用し，コントロールプレーンが提供するAPI（Application Programming Interface）*19サーバにアクセスし，クラスタ全体を操作する．

(2)ワーカーノード（Worker node）

アプリケーションコンテナを格納するポッドをホストするノードである．後述のKubelet，Kube-proxyなどのコンポーネントを有する．

(3)ポッド（Pod）

Kubernetesアプリケーションにおける実行単位

＊4　デプロイ：アプリケーションをそれらの実行環境に配置して展開すること．
＊5　フレームワーク：ある領域のソフトウェアに必要とされる汎用的な機能や基本的な制御構造をまとめたもの．ライブラリでは，開発者が個別の機能を呼び出す形となるが，フレームワークでは，全体を制御するのはフレームワーク側のコードで，そこから開発者が個別に追加した機能を呼び出す形となる．
＊6　オートスケール：ネットワークトラフィックやCPUの利用量など，その時々の処理に必要なリソース量に応じてオンデマンドで自動的に仮想サーバを増減する仕組み．
＊7　AWS：Amazon Web Services社が提供するクラウドコンピューティングサービス．
＊8　EKS：AWSにおけるマネージドKubernetesサービス．
＊9　Microsoft Azure：Microsoft社が提供するクラウドコンピューティングサービス．
＊10　AKS：Microsoft Azureにおけるマネージドkubernetesサービス．
＊11　GCP：Google社が提供するクラウドコンピューティングサービス．
＊12　GKE：GCPにおけるマネージドKubernetesサービス．

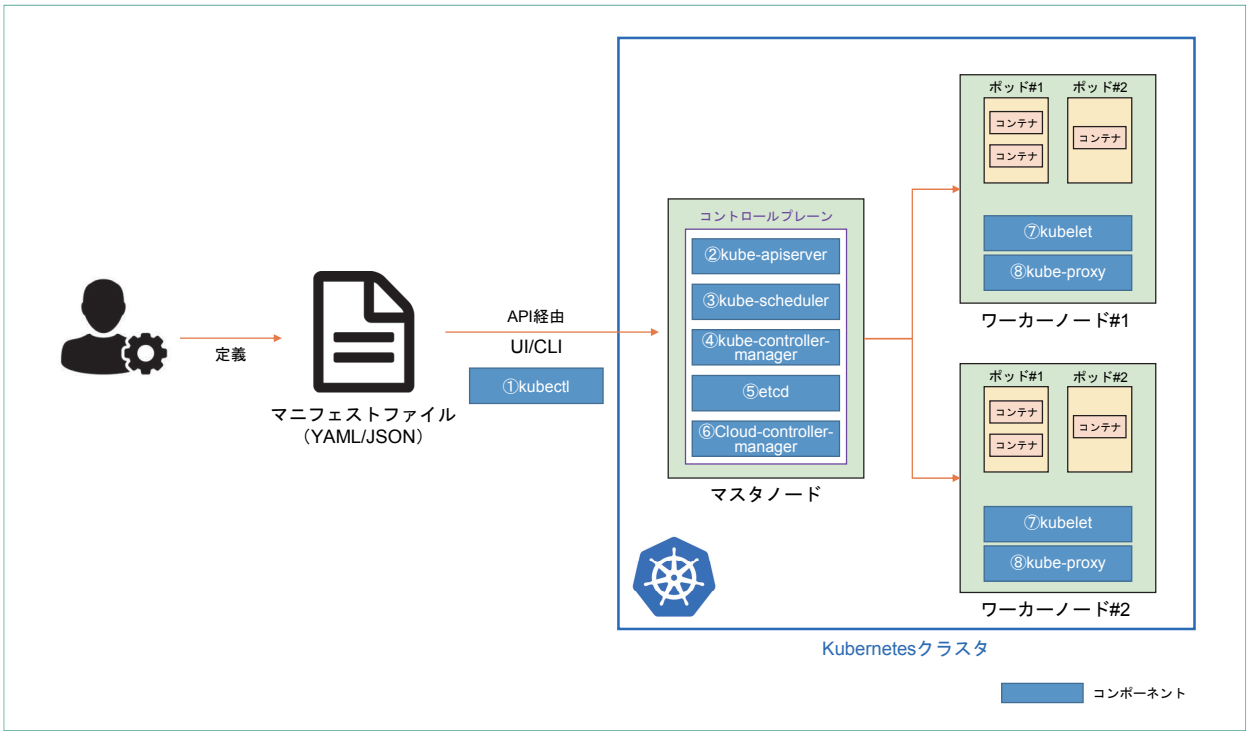


図1　Kubernetesのアーキテクチャ

で，アプリケーションのコンテナ，ストレージ，ネットワークIDやIPアドレスなどのネットワーク情報，実行方法を管理するオプションなどをカプセル化したものである．複数コンテナを格納することも可能である．

⑷マニフェストファイル（Manifest file）

リソースの構成が記載されたファイルで，ファイル形式はJSON（JavaScript Object Notation）[*20]やYAML（YAML Ain't Markup Language）[*21]を採用している．このファイルをAPI経由で宣言することで，クラスタ内のリソースの操作を可能とする．

2.3　コンポーネント

図1にあるコンポーネントについて以下に解説する．

①kubectl

ユーザが利用するコマンドで，kube-apiserverに対してリクエストを送り，リソースの作成／更新／削除などを行う．

②kube-apiserver

Kubernetesクラスタにおける APIを外部に提供するフロントエンドの役割を担う．

③kube-scheduler

新規ポッドがワーカーノードに割り当てられているかどうかを監視し，もし割り当てられていない場合，ポッドを実行する役割をもつ．スケジューリングの決定は，リソース使用率，ハードウェア／ソフトウェア／ポリシーによる制約など，いくつかの点を考慮して行われる．

＊13 マネージドサービス：クラウドサービスのうち，リソースのプロビジョニングや運用の大半をクラウド事業者の責任で実施しているサービス．クラウドコンピューティングサービスのうち，特にPaaSやSaaSを指す．

＊14 プライベートクラウド：企業・組織が社内でクラウド環境を構築し，社内の各部署やグループ会社に提供するクラウド形態を指す．一方利用・提供する人の範囲が限定されず，オープンに提供されるクラウドサービスはパブリッククラウドと呼ばれる．

＊15 クラスタ：複数のサーバを1つのサーバ群としてグループ化したもの．

＊16 Drupal：WordpressやJoomlaと同様のオープンソースのコンテンツ管理システム．

＊17 CNCF：CNCFはLinux Foundationのプロジェクトで，コンテナ技術の発展と，その進化に関連するテクノロジ業界の連携を支援するために2015年に創設された．

＊18 マニフェストファイル：アプリケーションが利用する機能などを宣言した設定ファイル．すべてのKubernetesのリソースはそれぞれの用途に合わせたマニフェストファイルを作成する必要がある．

④kube-controller-manager

ワーカーノードやポッドの状態をkube-apiserver経由で管理するコンポーネントである．ノードがダウンした場合の対応や，ポッドのレプリケーション*22管理などを行う．

⑤etcd

Kubernetesのすべてのクラスタ情報を保管するキーバリューストア*23である．また，Kubernetesのデータストア*24として使用することも可能である．

⑥cloud-controller-manager

ノード，ルーティング，ストレージなどクラウド事業者固有のオブジェクトを管理する．

⑦kubelet

ワーカーノード内で動くエージェントで，各ポッドの動作を保証する．また，マニフェストファイルで定義された内容とコンテナの設定情報が合致していることの監視，ノードやコンテナの実行環境*25を管理する．

⑧kube-proxy

コンテナ間の通信制御（ルーティング）を行う．

2.4 提供機能

Kubernetesは，複数サーバでコンテナ管理するだけではなく，コンテナのオートスケーリングや障害時の自動復旧*26など，運用における骨の折れる点の数々をサポートする．いくつか重要な点を解説する．

⑴ネットワークの負荷分散

一部のコンテナへのアクセスが集中している場合，その他のコンテナへトラフィックを分散することでコンテナの状態を安定化させる．

⑵ローリングアップデート／ロールバック

ユーザは，マニフェストファイル上で，コンテナの状態を変更するだけで，Kubernetesがその状態に変更してくれる（ローリングアップデート）．仮にアプリケーションの更新に失敗しても，マニフェストファイルを以前の状態に戻すだけで簡単に以前の状態に戻すことも可能である（ロールバック）．

⑶自動ピッキング

タスクごとに実行するノードやリソース使用量，優先度を定義できる．

⑷自動修復

障害などでコンテナが停止しても，自動でその状態を検出し，再起動を行う．

2.5 ドコモのKubernetesにおける課題

ドコモのあるプロジェクトでは，すでに2017年ごろから複数の大規模Kubernetesクラスタを商用システム上で利用していた．しかし，Kubernetesは自身のクラスタのみ管理する機能しかなく，運用者は複数のKubernetesクラスタを個別に管理しなければならなかった．例えば，複数クラスタにまたがるコンテナの状況を知りたいとき，それぞれのクラスタにアクセスし，状況をチェックする必要があった．また，機能別にクラスタを分割していたことから，クラスタ間のリソース使用率に大きな偏りが発生していた．例えば，あるクラスタではリソースのひっ迫から一時的にインスタンス*27を追加しなければならない状態である一方，他のクラスタではリソース使用に空きが見られるといったことがあった．仮に，Kubernetesクラスタ間でタスクを分散するロードバランシング*28やジョブスケジュール機能が存在すれば，このような課題が発生することはなかった．

この課題が顕在化した2018年ごろには，これらの課題を解決するオープンソースソフトウェアや外部

---

*19 API：ソフトウェアコンポーネント同士が互いに情報をやりとりするのに使用するインタフェースの仕様．
*20 JSON：JavaScriptのオブジェクト記述に基づくデータ記述言語．
*21 YAML：XMLやJSONと同様，データ構造を記述する記法，処理フォーマット．
*22 レプリケーション：データベースにおいて，データを他のサーバに複製して冗長性やバックアップを実現する仕組み．
*23 キーバリューストア：従来のリレーショナルデータベースとは異なり，レコード（データ）をキーと値の組み合わせで管理するストレージシステム．
*24 データストア：データを保存するシステム．
*25 実行環境：ソフトウェア開発用のライブラリやテスト環境など，開発に必要となるシステムを含まず，実行処理だけを提供するシステム環境．
*26 自動復旧：システムに障害が発生した際に自動的に冗長化された待機システムに切り替える仕組み．

ツールは存在しなかったため，我々でCOを開発した．なお，2019年ごろから，Kubernetesをさかんに使用していた先進的な企業（UberやRancherなど）から類似製品が提供されている．

3. COとは

3.1 開発経緯

ドコモのCOプロジェクトは，Drupalというオープンソース上で2018年初頭に開始された．当初の目的は，主にAWSを中心とした複数アカウントの管理であった．ドコモ内には，多数のアカウントが乱立し，1プロジェクトでも開発，QA（Quality Assurance）[*29]，商用と複数アカウントをもつことは珍しくない．AWSのサイトは，基本的にリージョン[*30]ごとにサービスが分割されているため，同一サービスであっても，複数リージョンにまたがった場合，それらを同一ページにてチェックすることができない．そのため，管理者の知らないところで開発者がさまざまなリージョンにインスタンスを起動させ，それらが停止／削除されないまま，管理者が請求書情報[*31]を見て，その存在に気付くということが多々発生していた．COは，こういった問題を未然に防ぐため，登録されたすべてのアカウント情報配下にあるリソース一覧を同一ページにて可視化する機能，クラスタのリソース使用率／時間ベースでの自動停止機能，シングルサインオン[*32]の機能などを提供する．ドコモ社内でKubernetesの問題が発生した際，これらのAWS向けに使用していた機能を基にCOを拡張し，Kubernetesクラスタに適用し始めたのが2018年末ごろである．

3.2 アーキテクチャ＆コンポーネント

COでは，ポータル機能などのユーザインタフェース部，ユーザ管理やクラスタ管理などの各種機能部，クラウドやKubernetesクラスタのコネクタ部の大きく3つから構成される．

図2に示すとおり，COではポータル機能を提供し

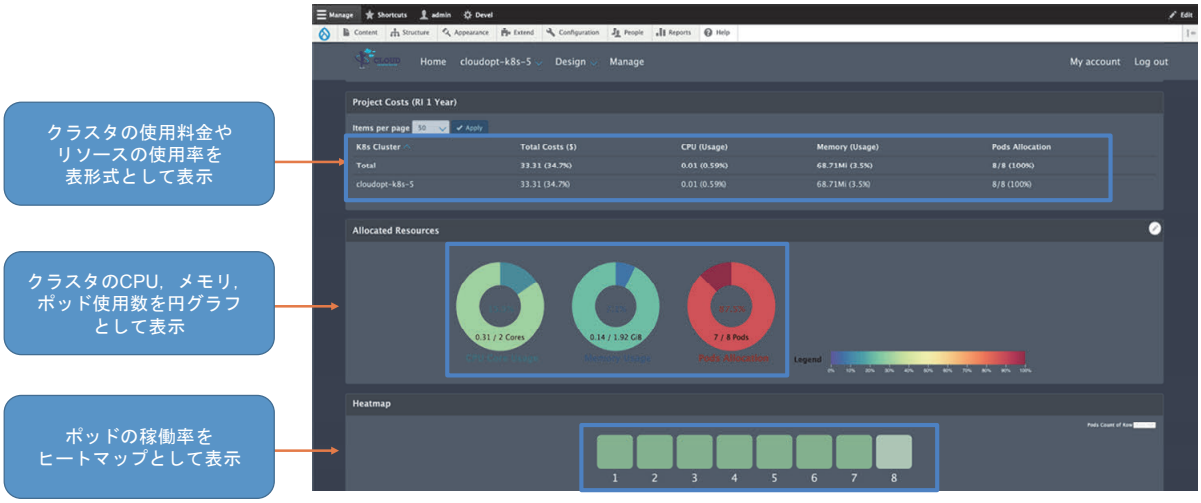


図2　COのポータル画面

＊27 インスタンス：クラウドコンピューティングにおけるオンデマンドで提供される仮想サーバ．ある処理が発生したときのみ仮想サーバが起動し終了するなど，仮想サーバの起動から終了までのライフサイクルは散発的である．
＊28 ロードバランシング：高負荷の処理を複数のサーバで分散して処理する仕組み．
＊29 QA：ソフトウェア開発において成果物の品質確保をする行為または品質保証をすること．
＊30 リージョン：クラウドサービスを提供するためのデータセンタが配置されている地域．
＊31 請求書情報：クラウド使用量を基にした請求書情報．
＊32 シングルサインオン：1つのアカウントで複数サービスにログインできること．あるユーザがシステムへの認証を1度行うことで，そのユーザに権限が与えられているすべての機能が利用できるようになること．

ている．画面上から，現在利用されているクラウド情報やKubernetesクラスタを一目で理解できる．また，クラスタ情報に行くと，現在の情報の状況を表，円グラフで可視化することができる．表示内容については，管理画面からカスタマイズが可能であり，ユーザが自由に変更可能である．また，可視化機能に加え，クラスタ上で動かしたいタスクのデプロイやデプロイ先のクラスタの選択など多くの機能を，このポータル機能を通して提供している．なお，DrupalでREST API（REpresentational State Transfer）*33機能を提供しているため，ポータル上の機能をAPI経由で実行することも可能である．

図3に示すとおり，ユーザ管理，クラスタ管理，ジョブ管理，ジョブスケジューリングなどCOでは多くの機能を提供している．Drupalが提供するロール*34管理機能は，任意の機能やサービスに対して，基本動作であるCRUD（Create，Read，Update and Delete）*35機能を定義することができる．この機能を拡張し，COではKubernetesクラスタにあるすべてのリソースオブジェクトに対するCRUD操作を規定することができる．このロール管理により，同一クラスタ上に異なるユーザがそれぞれのタスクをデプロイしても，各ユーザに自身のタスクのみ操作可能なロールを付与，つまり必要最低限のロールを付与していれば，クラスタ上のユーザ間の干渉を防ぐことができる．この考えは，後述するコスト最適化を支える基本的な機能である．

コネクタ部では各種クラウドやKubernetesクラ

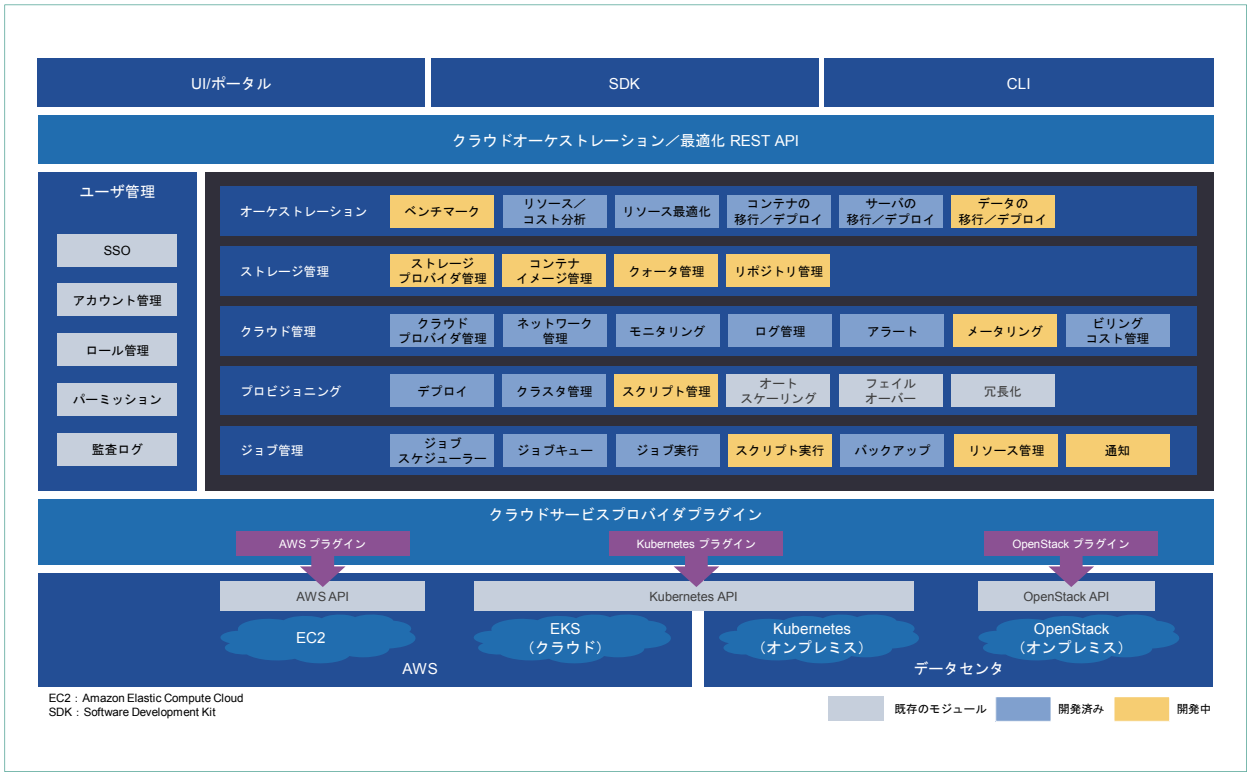


図3　クラウドオーケストレーション・最適化のコンポーネント

＊33 REST API：ウェブで用いられるソフトウェアアーキテクチャのスタイルの1つ．

＊34 ロール：ある特定の権限をユーザに付与するときのグループ（役割）のこと．

＊35 CRUD：システムやサービスを提供するソフトウェアにおいて，基本的な操作であるCreate／Read／Update／Delete（新規作成／読込み／書込み・更新／削除）の頭文字をとったもの．

スタが提供するAPIを使用しており，我々はそれらをモジュール化し，提供している．ユーザは，このモジュール機能をONにし，定期的に情報の取得を開始する．

3.3 主要機能

COにおける主要機能を解説する．

⑴シングルサインオン

Drupalのシングルサインオン機能を活用し，COは各種クラウドと連携ができる．また，Kubernetesクラスタはユーザ認証機能がないため，CO自体がユーザ認証機能を提供している．

⑵リソース最適化

COが複数のKubernetesクラスタを管理している場合，クラスタ間のロードバランシングを自動的に行う機能である．例えば，任意のコンテナのデプロイ時に，その時のクラスタのリソース使用率を基に，最も空いているクラスタを自動的に選択し，デプロイを行う．あらかじめ，ユーザ側でデプロイ先のクラスタを選択することも可能である．

⑶スケジューリング

時間とリソースベースのスケジューリング機能を有する．例えば，利用者が深夜帯などの特定時間帯にバッチ処理[*36]を行いたい場合に，タスクのデプロイ時間と終了時間をあらかじめ設定することができる．また，リソースが空いている時にのみ，優先度の低いタスクをデプロイすることもできる．

⑷コスト算出

マスターノードとワーカーノードのインスタンス費用を合計値とし，リソース使用率（メモリ，CPU，ポッド数）から算出するロジックを独自で規定している．これにより，Namespace[*37]単位でのコスト算出が可能となる．

⑸マルチクラウド管理

AWS，OpenStack[*38]，VMwareなどのパブリック／プライベートクラウドと同様に，Kubernetesクラスタも統一的に管理することが可能である．利用者は異なるアカウントやクラスタであっても，統一してリソース利用状況を確認することができる．

3.4 COの利用方法

COは，Drupalでオープンソースとして公開されているため，誰でも利用することができる［2］．

4. ドコモ事例

4.1 複数クラスタの管理と空きリソースの効率化

⑴複数クラスタの管理

COはドコモの商用システムで活用されている．このシステムの特長は，複数EKSを利用しており，1つのクラスタは100台以上のノードをもっている．**図4**に示すとおり，CO導入前，管理者は各クラスタの管理画面からクラスタやポッドの状態のチェックやCLI（Command Line Interface）[*39]経由でのコンテナのデプロイの実行など，その運用が煩雑化していた．

一方，COの導入後，COがすべてのクラスタからメトリック[*40]を取得するため，管理者は各クラスタへの確認が不要となった．また，管理画面上にてデプロイしたいコンテナとデプロイ先のKubernetesクラスタの情報を登録することで，COが自動的にデプロイを実施してくれる．仮にデプロイ先が複数Kubernetesクラスタであっても，クラスタを複数選択しておけば，自動的に選択されたクラスタ群にコンテナのデプロイを実施し，それぞれからメトリックを取得する．なお前述したように，クラスタ

＊36 バッチ処理：ユーザとの対話なしにデータをまとめて自動処理すること．
＊37 Namespace：Kubernetesにおいて，複数の異なるリソースを1つにまとめるための仮想的なグループの単位．
＊38 OpenStack：サーバ仮想化技術を用いて，一台の物理サーバを仮想的に複数のサーバのように動作させ，仮想サーバをユーザが利用するクラウドサービスごとに割り当てるクラウド基盤のソフトウェア．オープンソースソフトウェアとして提供されている．
＊39 CLI：コンピュータやソフトウェアへの指示をすべて文字によって行う方式．
＊40 メトリック：ある期間におけるCPUの利用率，メモリ使用量，ポッドの数といった定量的なデータ．

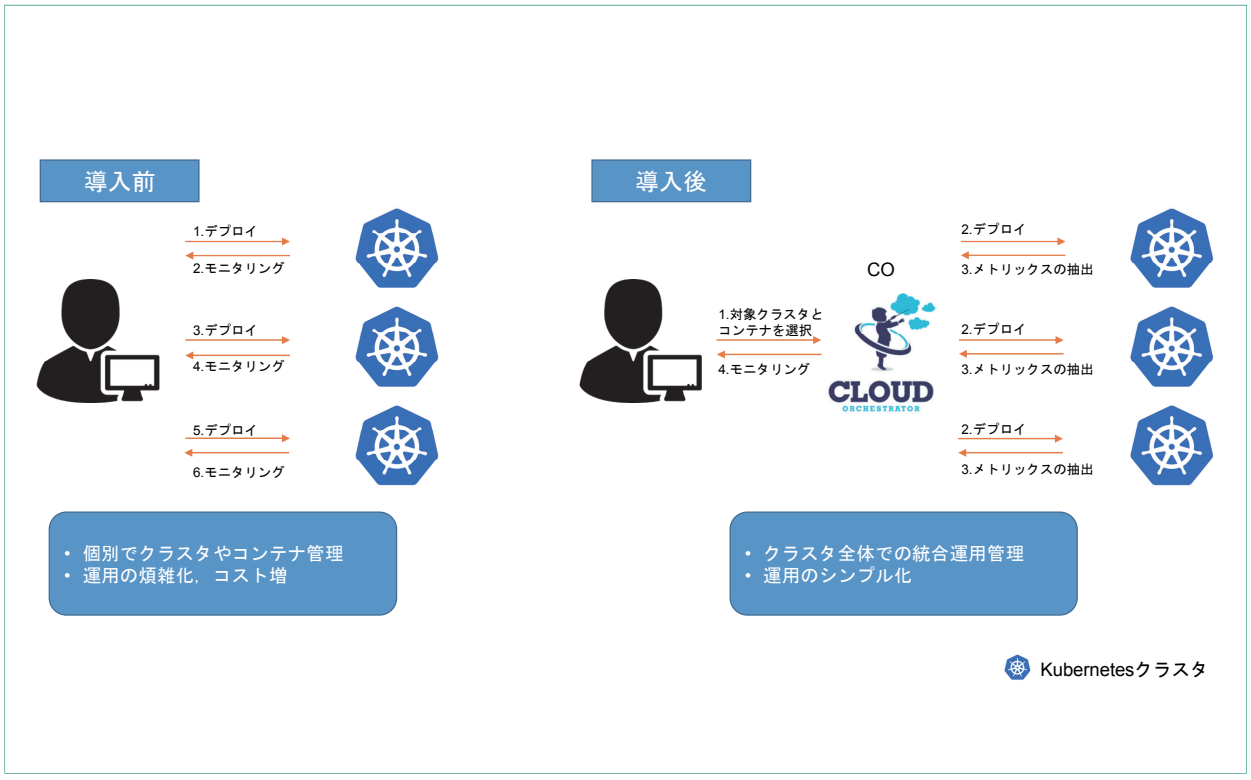


図4　CO導入における変化

選択において，ユーザは特定のクラスタを選択せず，COが自動的に空いているクラスタを選択するリソース最適化機能を使用することも可能である．

⑵空きリソースの効率化

図5に示すとおり，COは時間とリソースベースのスケジューリング機能を有している．ドコモの商用システムでは，システム全体のリソース使用率がユーザの利用に比例する傾向があった．具体的には，ユーザが良く利用する日中帯は80％以上と高い使用率になる一方で，深夜帯になるとその使用率は20％近くまでに落ち込むといった具合である．空いたリソースを利用するため，管理者はCO上で，デプロイしたいコンテナとスケジューリング方法と必要なパラメータを登録しておけば，COは登録されたスケジューリング方法に従って，コンテナのデプロイを実施する．このケースにおけるコンテナは，リアルタイム性を要求せず，他のコンテナより優先度が低いことが多く，クラスタのリソース使用率のバースト＊41時には，優先的にこれらのコンテナが削除される仕組みとなっている．この商用システムでは，時間ベースのスケジューリング機能により，深夜帯においてシステムログの集計や機械学習モデルの更新を実施している．またリソースベースのスケジューリング機能により，あるクラスタのリソースがひっ迫した場合，一時的にリソースを追加するのではなく，そのタスクの一部を他のクラスタに自動的に分散することで，システム全体のリソースの均等化を図っている．

＊41　バースト：ネットワークトラフィックが一時的に増大するなどして，瞬時的に複数の信号が，装置内の一定箇所に集中すること．

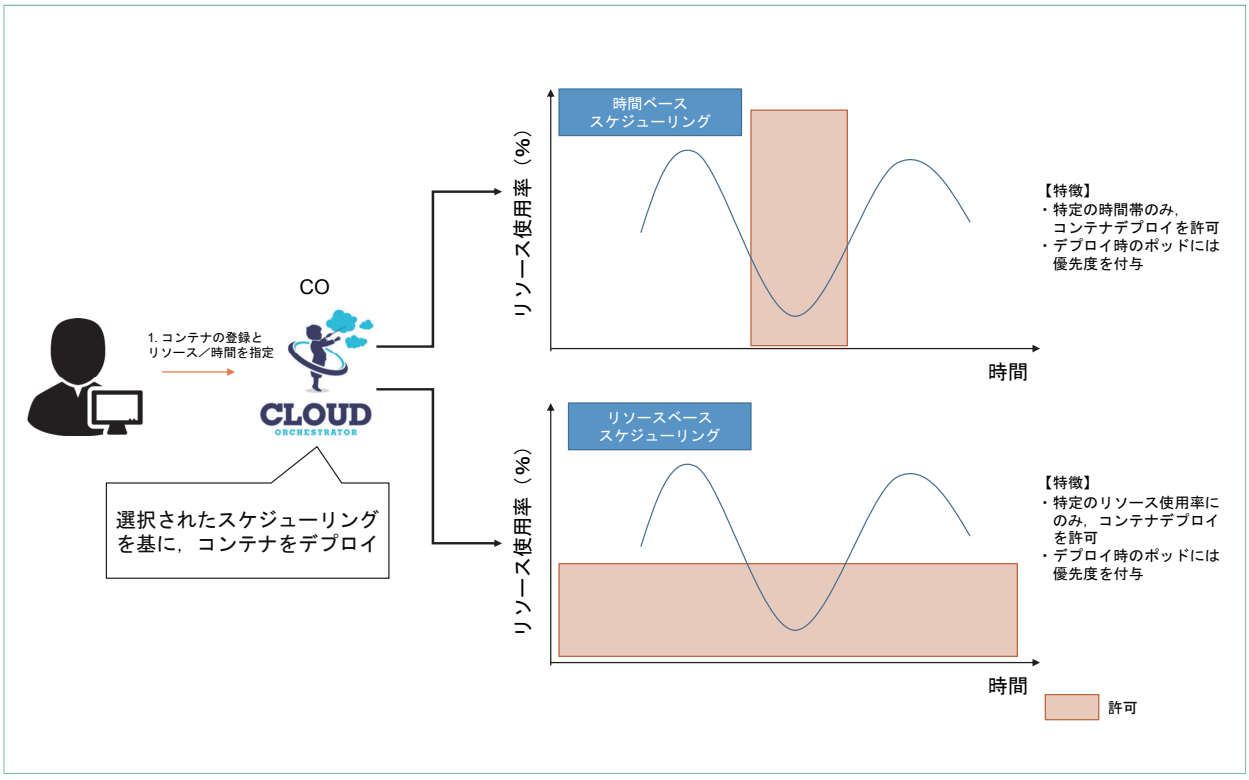


図5　スケジューリング機能

4.2　アプリケーションとシステムレイヤの分離

図6(a)に示すとおり，ドコモ内の各プロジェクトはそれぞれがAWSアカウントを作成し，その中でアプリケーション開発やシステム構築を行っている．この形式では全プロジェクトでAWSの知識の獲得，アカウントやシステム環境の管理，セキュリティポリシーの理解が必要で，サービス展開までに膨大な稼働が掛かってしまう．さらに，セキュリティ専門家の不在によるセキュリティリスクの増大，全体最適ではないことによるクラウドコスト増といったさまざまな問題が発生する．

この課題を解決するために，COを境にアプリケーションとシステムレイヤで分離することを考えている（図6(b)）．現在は構想段階であるが，この方式にすれば，アプリケーション開発者はサービスのコンテナ化を条件に，COより下のレイヤの管理や面倒なセキュリティ管理が不要になり，サービス開発に注力することが可能となる．一方で，システム管理者は，コンテナ化によりアプリケーションの中身を意識する必要がなくなる．システム管理者がクラスタを集中管理することにより，全体リソースの効率化やセキュリティリスクの低減が図られ，最終的には全社的なクラウドコストの削減が期待される．

ただし，この方式では，複数のサービスが同一のクラスタで動くことになるため，重い処理を行うコンテナは他のサービスに影響を及ぼす可能性がある．また，アプリケーション開発者はクラウド費用を意

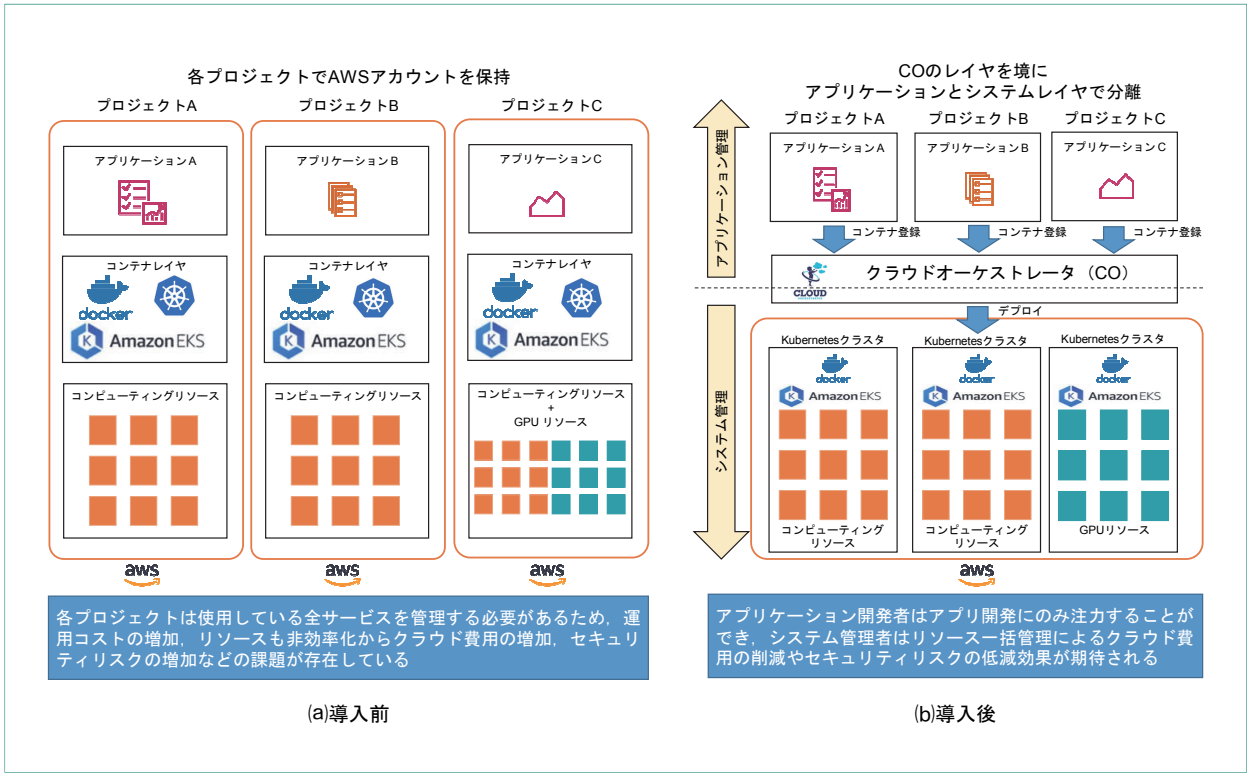


図6　COによるアプリとシステムレイヤの分離

識することがなくなるので，非効率な処理を実行する可能性もある．この点は，KubernetesのResource quota[*42]やLimit range[*43]の機能を使用することに加えて，COではKubernetesクラスタから得られるCPU，メモリ，実行ポッド数などのリソース使用率を基にしたコスト算出機能を利用して，アプリケーション開発者に利用料を通知する．これにより，アプリケーション開発者にコスト意識をもたせることができる．

現在，サービスイノベーション部と連携して，上記方式の検証を進めている段階である．この方式を満たすためには，CO側の機能が不足しているため，引き続き，これらの更新をしていく予定である．

5. あとがき

本稿では，複数Kubernetesを管理するCOについて解説した．ドコモ社内で蓄積された知見を活かして，クラウドの利用をさらに効率化できるよう，今後もCOを改善したいと考えている．

文　献

[1] Kubernetesホームページ.
https://kubernetes.io/
[2] Drupal：“Cloud.”
https://www.drupal.org/project/cloud

＊42 Resource quota：コンテナへのリソースの割当て量の設定値.
＊43 Limit range：コンテナからのリソースリクエストを制限する最小値と最大値の範囲を決める設定値.

Technology Reports（特集）　パブリッククラウド　コスト　AWS
パブリッククラウド活用特集

クラウドコスト最適化の取組み

イノベーション統括部　住谷（すみや）　哲夫（てつお）

現在，国内外の企業においてパブリッククラウドが非常に多く活用されているが，主要なパブリッククラウドの課金体系は従量制であり，仮想サーバなどのリソース管理が疎かだと無駄にリソースを起動させてしまい，想定外のコストが発生する．そのため，初期の設計段階からリソース管理や，コストの最適化について検討しておく必要がある．本稿では，ドコモがこれまで培ってきたクラウドコスト最適化のノウハウ・取組みについて解説する．

1. まえがき

近年，パブリッククラウドを利用したサービスを提供する企業が増えている．パブリッククラウドは従来のようなデータセンタでのシステム構築とは異なり，利用者が管理コンソールから数クリックするだけで簡単に仮想サーバなどのリソースを立ち上げることができるため，システム構築のスピードを圧倒的に速くすることができ，企業の競争力強化に貢献している．一方で，簡単に利用ができる分，多くの企業はいまだにパブリッククラウドのコストの管理や抑制に苦慮している．主要なパブリッククラウドの課金体系は基本的に従量制であり，仮想サーバなどのリソース管理が疎かだと無駄にリソースを保持してしまい，想定外のコストが発生する．そのため，初期の設計段階からリソース管理や，コストの最適化について検討しておく必要がある．

コストの最適化には，まずコストの可視化を行い，コスト管理の効果を見えるようにし，コスト削減施策と効果検証を繰り返していくことが重要となる．コストを可視化したリソースは継続して利用状況を定期的にチェックし，状況に応じて，構成変更，利用している仮想サーバの能力（AWS：Amazon Web Services[*1]ではEC2（Elastic Compute Cloud）[*2]インスタンス[*3]タイプなど）の見直し，料金プラン活用などの検討を行うことが必要である（**図1**）．

本稿では，クラウドコスト最適化の取組みについて，そのポイントとなる考え方を述べ，具体的なノウハウを解説する．

＊1　AWS：Amazon Web Services社が提供するクラウドコンピューティングサービス．
＊2　EC2：AWSが提供するIaaSの1つ．仮想マシン（＊15参照）を提供するサービス．
＊3　EC2インスタンス：AWSで提供される仮想サーバ．

2. コスト最適化

コストの最適化は以下の優先度で検討をする（**図2**）．

STEP①：構築したアーキテクチャの見直し

コスト削減に最も有効なのは，アーキテクチャ全体を見直し極力マネージドサービス（自分で構築するのではなく，すでにクラウド事業者が提供しているサービス）の活用を考慮することである．運用中のシステムの変更は難しいため，初期設計時やシステム更改時にしっかりとコスト観点での設計を行う必要がある．

STEP②：不要なリソースの削除，利用リソースの最適化

運用期間が長くなっているアカウントでは，使われなくなったリソースが保持され続けているケースや，本来であれば必要のない過剰なリソースを利用しているケースが散見される．無駄が無いよう定期的なリソース見直しを運用に取り入れることで，コスト削減につなげることができる．

STEP③：各種料金プランの活用

主要なパブリッククラウド事業者は，リソース利用のコミット（定められた期間の利用量をあらかじめ宣言）をすることで利用料を削減することができる料金プランを用意している．例えば，AWSではRI（Reserved Instance），SP（Saving Plans）といった，1年もしくは3年の利用をコミットした上で料金を下げることができる料金プランがある．Google Cloud Platform[*4]では確定利用割引，Microsoft Azure[*5]でも

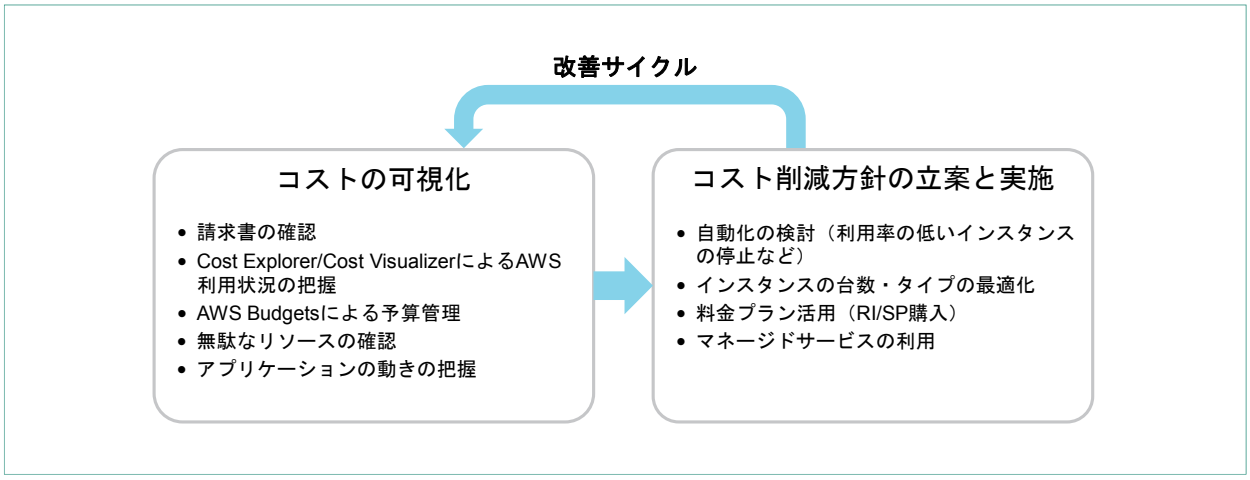


図1　コスト最適化の改善サイクル

図2　コスト最適化の検討プロセス

＊4　Google Cloud Platform：Google社が提供するクラウドコンピューティングサービス．

＊5　Microsoft Azure：Microsoft社が提供するクラウドコンピューティングサービス．

Reserved VM（Virtual Machine）*6 Instancesという形で同様の料金プランがある．リソースを最適化した上でどうしても一定期間継続的に必要となる場合には，このような料金プランを活用することでコストを削減することができる．ただし，リソース利用のコミットをする場合は，コミット期間内のシステム構成の大幅な変更や，利用リソースの見直しの実施が難しくなるため，STEP①，②の検討を行った上で活用を考えるべきである．

3. コストの可視化

上記のポイントを考慮した上で，システムのどの部分にコストがかかっているのか，現状のコストを可視化して継続的にコスト構造を把握することが重要である．

主要なパブリッククラウド事業者はコストの可視化ツールを提供している．ここでは，AWSが提供するCost Explorerを解説する．また，Cost Explorerの提供開始前からドコモで内製開発し，全社的に利用しているCost Visualizerについても解説する．

3.1 Cost Explorerによる利用状況の把握

Cost ExplorerはAWS標準のツールで，請求金額をグラフで表示することができる（**図3**）．そのため，サービス別，メンバアカウント*7別など，さまざまな切り口でコストの細分化が可能である．また，デフォルトでRIやSPの利用率，カバー率など複数のレポートを出力することができる．

注意点として，Cost Explorerの利用に当たっては，必要最低限の権限のみ（例えばコストの閲覧権限のみ）を付与したIAM（Identity and Access Management）ユーザ*8を作成し，アクセスするべきである．

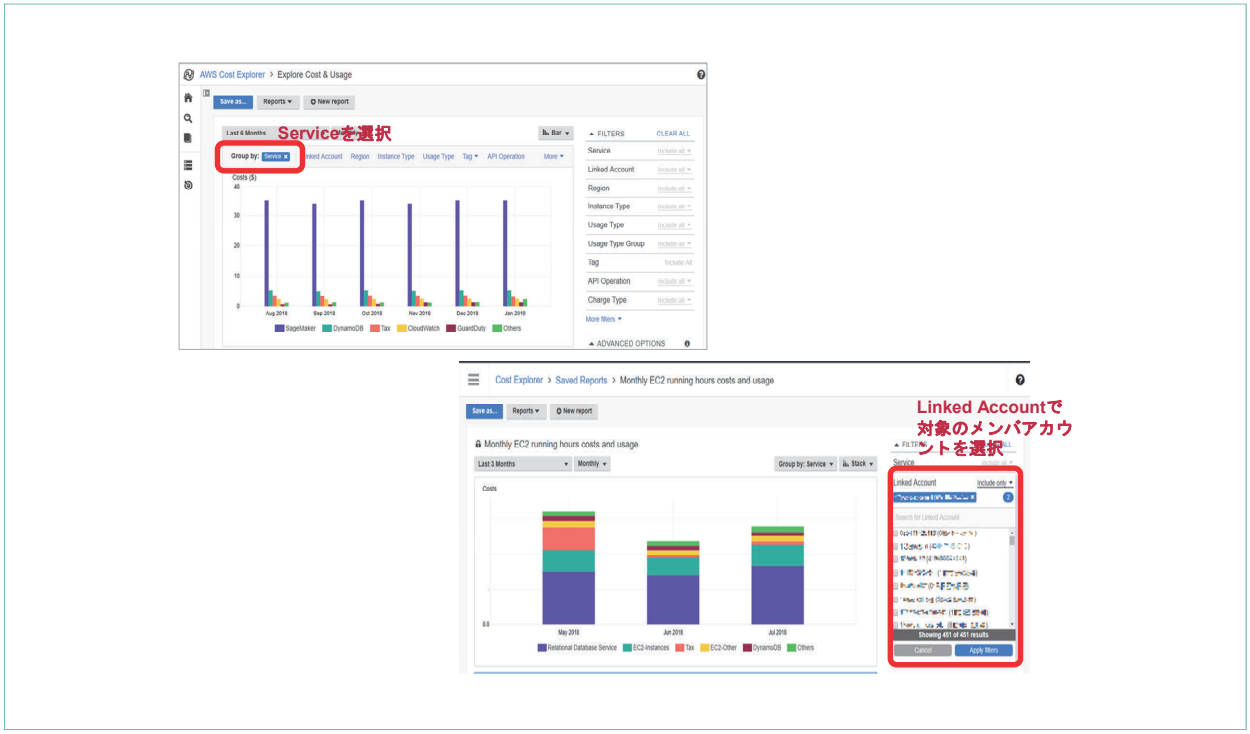


図3　Cost Explorer操作画面イメージ

*6　VM：仮想マシン（*15参照）．

*7　メンバアカウント：複数のAWSアカウントを統合する組織に所属する，管理アカウントでないアカウント．

*8　IAMユーザ：IAMサービスで作成した，AWS環境へのアクセス権をもつユーザ．

3.2 Cost Visualizerによる利用状況の把握

Cost Visualizerは，ドコモが開発・提供をしているコスト分析ツールである．ドコモが大規模にAWSを使い始めた2012年時点では前述のCost Explorerが提供されておらず，コスト管理上の必要性から本ツールが内製開発された．

Cost Visualizerは，AWSとは権限設定が分離しており，アカウント管理が別になっているため，AWSのサポートレベルや請求閲覧権限によらず利用することが可能で，Cost Explorerよりも細かい権限管理や，円グラフ表示，グルーピング表示など，Cost Explorerと異なる分析が可能である（**図4**）．

Cost Visualizerのシステム・アーキテクチャを**図5**に示す．AWSから提供されるコストと使用状況レポート（CUR：Cost and Usage Report）がAmazon S3（Simple Storage Service）*9に自動的に格納され，そのデータをETL（Extract，Transform，Load）サービス（データベースにデータをロードして利用可能な状態にするサービス）であるAWS Glue*10で抽出，変換を行い，Cost Visualizerが動作する仮想サーバ内のデータベースにロードする．マネージドサービスであるデータベースサービス（AWSではRDS：Relational Database Service）を使わずに敢えて仮想サーバ内にデータベースを起動させるのは，

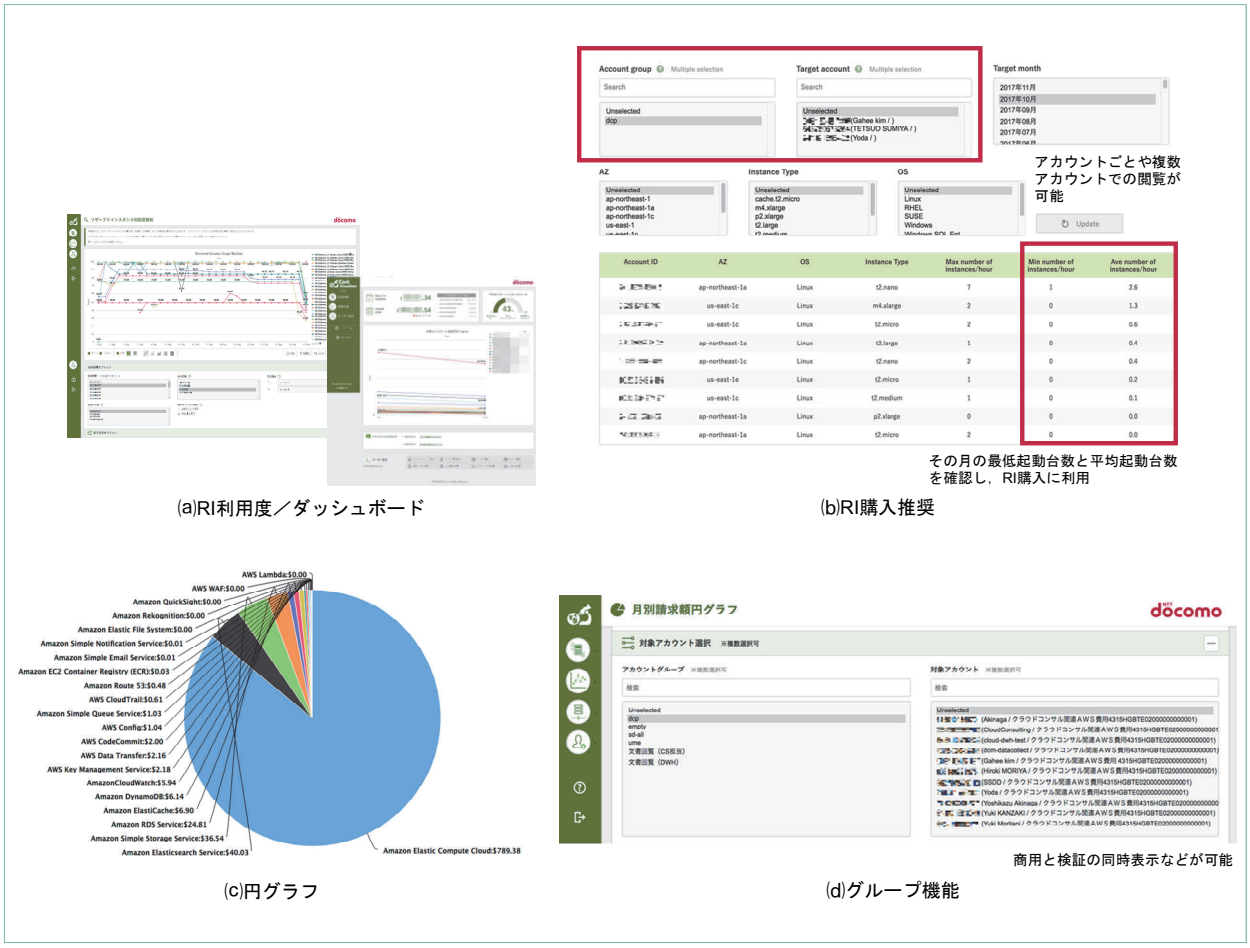


図4　Cost Visualizer操作画面イメージ

*9　Amazon S3：AWSが提供するストレージサービス．
*10　AWS Glue：AWSが提供するPaaSの1つ．データの分類，加工のための処理を実行できるサービス．

大量のデータに対してクエリ*11を連続的にかけるため，グラフ描画までの遅延を最小限にする工夫を内部的にもっているためである．また，仮想サーバ外ではAWS Glueのほかにも，AWS Lambda*12（サーバレスサービス）やAmazon Dynamo*13（Key-Valueストア*14サービス）などのマネージドサービスを活用することでコストを極力低減できる構成としている．

Cost Visualizerを使ったコストの可視化の例を**図6**に示す．マネージドサービスを使わず，仮想マシン*15中心のアーキテクチャの場合はこのようにEC2のコストが支配的になるため，EC2のコストを削減する対策が必要となることが分かる．

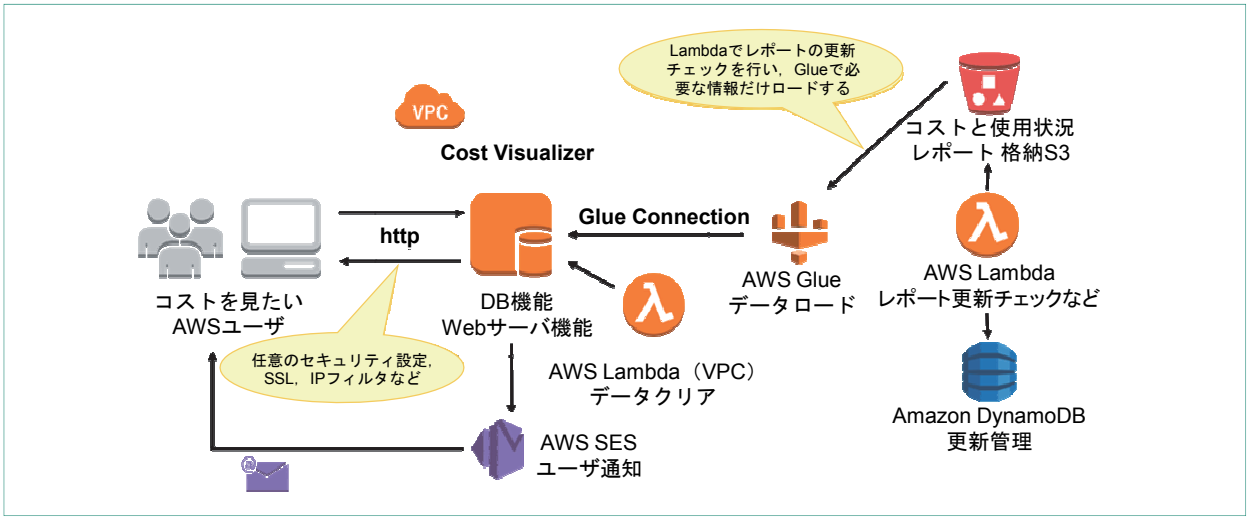


図5　Cost Visualizerシステム・アーキテクチャ

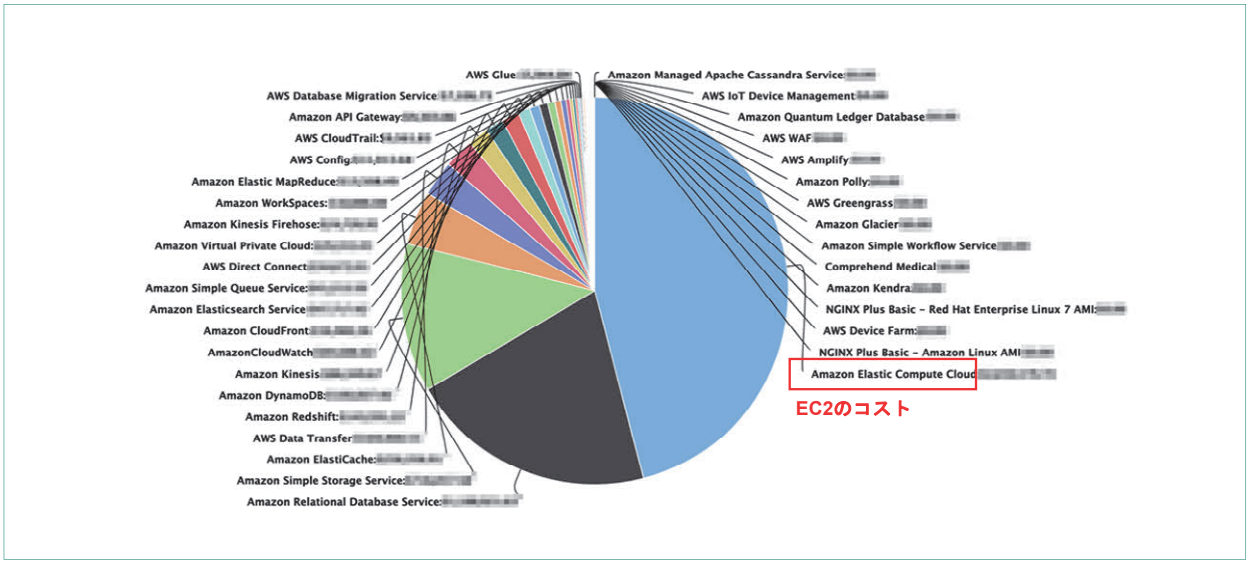


図6　Cost Visualizerによるコスト可視化の例

*11 クエリ：データベースに対する問合せ（処理要求）．
*12 AWS Lambda：AWSが提供するFaaSの1つ．アプリケーションコードの実行環境が提供されており，利用者は作成したソースコードを登録してアプリケーションが実行できる．
*13 Amazon Dynamo：AWSが提供するPaaSの1つ．高い信頼性や性能をもつ非リレーショナルデータベースのサービス．
*14 Key-Valueストア：キーと値を組み合わせる単純な構造からなるデータストア．
*15 仮想マシン：ソフトウェアによって仮想的に構築されたサーバなどのコンピュータ．

3.3 予算管理サービス

主要なパブリッククラウド事業者は予算管理のためのサービスをもち，コストまたは使用量が設定値を超過したり，超過する可能性を検知したりするとアラートをメールなどで通知している．AWS Budgetsは，コストだけではなくAWSリソースの利用量やRIの使用率なども設定することができ，コスト可視化ツールと合わせて利用することで普段のコスト意識を高めることができる．

4. コスト削減方針の立案と実施

4.1 マネージドサービスの利用

主要なパブリッククラウド事業者は，仮想サーバのようなコンピューティングリソースだけではなく，処理の特性に合わせたマネージドサービスを提供している．こういったマネージドサービスは，リソースを活用して処理を行った時間分の課金であるケースが多く，プロビジョニング*16しているだけでコストがかかる仮想サーバでシステムを構築するよりもコストを削減することができる．

例えば，AWSであれば以下のような対応をすることで大幅なコスト削減を見込むことができる．

- 長時間稼働させるサービスでリクエストが少ないものについては，EC2からLambdaへ置き換える．
- バッチ処理*17を流したい場合はEC2ではなく，AWS Batch*18の利用を検討する．AWS Batchを使用すると，バッチジョブのボリュームと必要なリソースに応じて，コンピューティングリソースが動的にスケーリング*19されるためコストを削減できる．
- 図7のようなCognito，API Gateway，Lambda，DynamoDBなどによるサーバレスアーキテク

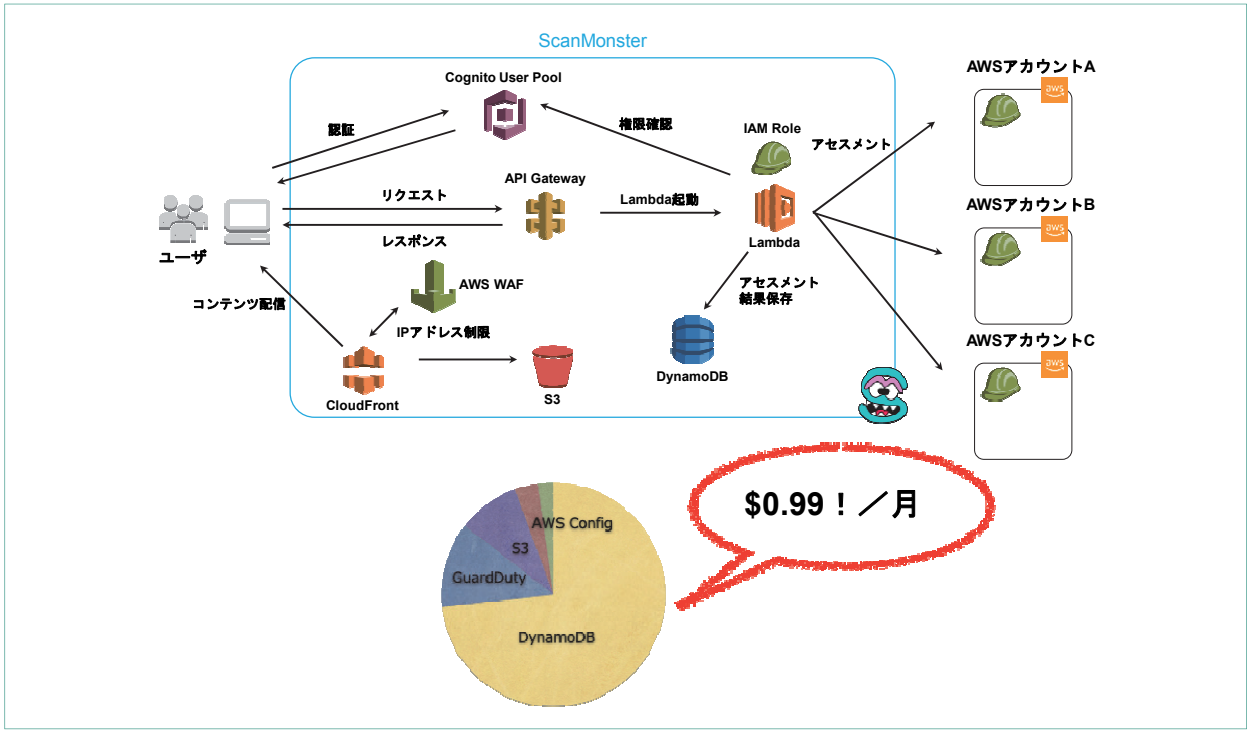


図7　サーバレスアーキテクチャの一例（ScanMonster）

*16 プロビジョニング：アプリケーションを実行するために必要となるサーバやネットワークなどのリソースの確保やそれらを動作させるための各種設定作業．
*17 バッチ処理：一定量，一定期間のデータを集め一括処理をする処理方法．
*18 AWS Batch：AWSが提供するPaaSの1つ．大規模なバッチ処理を簡単かつ効率的に実行できるサービス．
*19 スケーリング：ハードウェアやVMの負荷状況に応じて処理能力が不足，あるいは余剰になった際に，VMを増減することにより処理能力を最適化すること．

チャに変更する．

4.2 無駄なリソースの確認

クラウドのリソースは一度プロビジョニングすると実際に利用をしなくてもコストが発生してしまうため，定期的に無駄なリソースを保持していないかチェックする必要がある．例えば，起動中のEC2インスタンスに関連付けられていないAmazon EBS（Elastic Block Store）ボリューム[*20]，タグも付けられていない古いEBSスナップショット[*21]などがそういったリソースに該当する．

これらの無駄なリソースはクラウドサービスのコンソールから確認することができるが，多くのリソースを扱う場合，確認が難しくなってしまう．こういった無駄なリソースの確認の手段として，AWS Trusted Advisorなどを活用することが効率的である．Trusted Advisorでは，以下の項目を確認することができる（**表1**）．

実際にコスト削減に繋がった例として，あるプロジェクトでTrusted Advisorを利用してリソース状

表1　AWS Trusted Advisorで確認できるコスト最適化ポイント

確認項目	概　要
使用率の低いEC2インスタンス	CPU使用率，ネットワークI/Oの利用量から利用状況を判断
アイドル状態のロードバランサー	ロードバランサーへのリクエスト数や，関連付けられているEC2インスタンスの数から利用状況を判断
アイドル状態のRDS DBインスタンス	RDS DBインスタンスへの接続頻度によって利用状況を判断
利用頻度の低いAmazon EBSボリューム	EBSボリュームがEC2インスタンスにアタッチされていない，もしくは書込み頻度によって利用状況を判断
Route 53レイテンシリソースレコードセット	非効率的に設定されたレイテンシーレコードセットを確認
使用率の低いRedshiftクラスタ	Redshiftへのクラスタ接続頻度，CPU使用率から利用状況を判断
関連付けられていないElastic IP Address	実行中のEC2インスタンスに関連付けられていないElastic IP Addressを確認
RI有効期限切れ	前後30日の間に有効期限が切れたRIを確認
Amazon EC2 RIの最適化	EC2 RIの推奨購入数を表示
Amazon RedShiftリザーブドノードの最適化	Red Shift RIの推奨購入数を表示
Amazon RDS RIの最適化	RDS RIの推奨購入数を表示
SPの推奨事項	SPの推奨購入数を表示
Amazon ElastiCacheリザーブドノードの最適化	ElastiCache RIの推奨購入数を表示
Amazon Elasticsearch RIの最適化	Elasticsearch RIの推奨購入数を表示

Amazon ElastiCache：AWSが提供する，完全マネージドのインメモリデータストアサービス．
Amazon Elasticsearch：AWSが提供する，オープンソースである検索エンジンElasticsearchのマネージド・サービス．
Elastic IP Address：AWSが提供する，固定IPアドレスのサービス．
Redshiftクラスタ：AWSが提供するデータウェアハウスサービスのクラスタ．
Route 53レイテンシリソースレコードセット：AWSが提供するドメインネームサービスであるRoute 53に登録できる，エンドユーザからのレイテンシを最小にできるドメインとEC2インスタンスなどの組合せ．
ロードバランサー：サーバにかかる負荷を平等に割り振るための装置．AWSはロードバランサーをサービスとして提供している．

＊20 Amazon EBSボリューム：AWSが提供する高性能で可用性に優れたブロックストレージのサービス．ブロックストレージとは，記録領域をボリュームという単位に分割し，ボリュームの内部をさらに固定長のブロックという単位に分割して管理するストレージを指す．
＊21 EBSスナップショット：Amazon EBSボリュームのバックアップデータ．

況を確認したところ42個のブロックストレージのうち，22個がアタッチされておらず，これらを削除することでコンピューティングコストを約1割削減することができたケースがあった．また，他のプロジェクトではタグが付いていない1,000を超えるスナップショットがあり，これらを削除することでコスト削減に繋がった．

4.3 アプリケーションの動きの把握

アプリケーションをデプロイ*22する際，当初の見込みよりも実際はトラフィックが無く，過剰なプロビジョニングをしているケースがある．オンプレミス*23ではリソースを縮退することは難しいが，クラウドであれば適切なリソースに縮退や拡張をすることができる．リソースの使用状況については，AWSであればCloudwatch*24，Google Cloud Platformは Cloud Monitoring*25，Microsoft Azureは Azure Monitor*26といったサービスで確認が可能である．クラウド事業者提供のサービス以外にも，New Relic社やDatadog社などが提供する監視サービスもあり，そういったサービスを活用することで適切なリソースへの変更が可能である．AWSでは Compute Optimizerという機能があり，アイドル状態のインスタンスや使用率の低いインスタンスを特定しコストを削減できるレコメンドを行うことができる（**図8**）．

また，インスタンスタイプの価格は，最新のものが安くなるので，最新のインスタンスタイプへの変更を常に意識する必要がある．

4.4 自動化の検討

常時起動の必要がない検証環境については，深夜・休日を停止することでかなりコストを抑えることができる．例えば，平日深夜5時間停止に加えて土日に停止をさせることで，コストを約6割以下に削減することができる．ここまで停止することができれば，1年程度のコミット利用による価格低減よりも安くなる可能性が高い．多くのリソースを扱う場合は手動で毎日停止することは難しいため，自動化することで漏れ無く停止することができる．また，

図8　AWS Compute Optimizerの操作画面イメージ

＊22 デプロイ：アプリケーションをそれらの実行環境に配置して展開すること．
＊23 オンプレミス：企業がシステムを構成するハードウェアを自社で保有し，自社で保守運用すること．
＊24 Cloudwatch：AWSが提供する，AWSリソースやAWSで実行されるアプリケーションなどの監視サービス．
＊25 Cloud Monitoring：Google Cloud Platformが提供する，Google Cloud Platformのリソースや Google Cloud Platformで実行されるアプリケーションなどの監視サービス．
＊26 Azure Monitor：Microsoft Azureが提供する，Azureのリソースや Azureで実行されるアプリケーションなどの監視サービス．

バックアップの定期実行を設定し，古いバージョンについては自動的に削除する世代管理をすることでコストを低減することができる．

4.5 料金モデルの検討

これまでに記載したコスト削減の取組みを行った上で，どうしても必要になるリソースに対して，料金プランを活用することでさらにコストを削減することができる．AWSでは，コンピューティングリソースに対して，RIやSPという仕組みがある．RIは，OSの種別，リージョン[*27]単位かAZ（Availability Zone）[*28]単位，インスタンスファミリー[*29]などを指定して利用期間をコミットすることで料金を安くできる仕組みである．これに対して，SPは前述の条件（OSの種別，リージョン単位かAZ単位，インスタンスファミリーなど）の指定を緩めて純粋に利用料金のコミットをするため，柔軟に購入ができる分，割引率は低くなる．しかし，新しいインスタンスファミリーが随時発表されることを考えるとSPをベースにコミットを行うことで柔軟な運用が可能になる．

5. あとがき

本稿では，クラウドコスト最適化の取組みについて，そのポイントとなる考え方を述べ，具体的なノウハウを解説した．クラウドが従量制であること，そのためリソースや利用状況の管理を適切に行うことがコスト最適化の観点で重要となり，管理のためには可視化と効果検証を繰り返す必要があると述べた．コスト最適化には初期のシステム構成からコスト対効果を意識した設計が必要であり，運用フェーズでも継続して利用状況を定期的にチェックし，状況に応じて構成変更，インスタンスタイプの見直し，料金プラン活用などの検討を行うことが必要である．今後は，RIおよびSPの適用率に応じてSPを代表アカウントで購入するなど，ドコモ社全体としてのクラウドコスト削減の検討を行う．

＊27 リージョン：クラウドサービスを提供するためのデータセンタが配置されている地域．
＊28 AZ：物理的，ソフトウェア的に自律しているデータセンタの集合単位．
＊29 インスタンスファミリー：インスタンスの種別を用途別に分類したもので「汎用」「コンピューティング最適化」「メモリ最適化」などに分かれる．

Technology Reports（特集）
セキュリティ　クラウド　ガバナンス
パブリッククラウド活用特集

パブリッククラウドの利用に特化したセキュリティチェックツールの開発

イノベーション統括部　中村（なかむら）　拓哉（たくや）

世界中でパブリッククラウドの利用が加速する中，多くの企業がパブリッククラウドを利用して自社サービスや基幹システムのワークロードを実行している．パブリッククラウド上では，機密性の高い情報も扱われている場合があり，その際にセキュリティが非常に重要となる．

本稿では，パブリッククラウドを利用する上でのセキュリティに関する基本的な考え方について解説するとともに，ドコモにおける対策や取組みについて述べる．

1. まえがき

2021年現在，世界中の多くの企業がAWS（Amazon Web Services）[*1]やMicrosoft Azure[*2]（以下，Azure），GCP（Google Cloud Platform）[*3]などのパブリッククラウド[*4]を利用して，自社サービスや基幹システムのワークロード[*5]を実行している．パブリッククラウド上では，機密性の高い情報も扱われている場合があり，その際にセキュリティが非常に重要となる．

実際に，パブリッククラウドを利用して構築したシステムにおいて大規模な情報流出やサービスの停止などの情報事故が発生している．特に大規模な事例として，米国Capital One社がパブリッククラウド上に構築したシステムから1億人以上の個人情報が流出した事故［1］が注目を集めた．これらの事故の中には，パブリッククラウドを利用していたからこそ発生したものも多い．しかしながら，ビジネスを大きく加速させることができるなど，パブリッククラウドを利用することのメリットも数多く，もはや2021年現在において，セキュリティだけを理由にしてパブリッククラウドを利用しないという選択肢を取ることは，企業経営において適切ではないと考えられる．

ドコモでも，2009年ごろからパブリッククラウドを活用しており，2021年現在では数多くのワークロードをパブリッククラウド上で実行し，その間セキュリティに関するノウハウを蓄積してきた．本稿

＊1　AWS：Amazon Web Services社が提供するクラウドコンピューティングサービス．
＊2　Microsoft Azure：Microsoft社が提供するクラウドコンピューティングサービス．
＊3　GCP：Google社が提供するクラウドコンピューティングサービス．

では，パブリッククラウドを利用する上でのセキュリティに関する基本的な考え方について解説する．さらに，セキュリティ事故を起こさないためのドコモにおける対策や取組み事例について述べるとともに，取組みの中で開発したセキュリティチェックツール『ScanMonster』の特徴や構成について解説する．

2. クラウドセキュリティの考え方

パブリッククラウドが普及する以前は，自社でデータセンタやサーバを用意することが多かった．現在この方法はオンプレミスと呼ばれる．オンプレミスとパブリッククラウドの一番の違いは，オンプレミスでは目の前で見えるサーバを管理できていたことに対し，パブリッククラウドではデータセンタの存在すら我々から隠匿されてしまう点であろう．直接目に見えず，直接管理することもできないサーバ上に構築されたパブリッククラウドをうまく活用していく上で，非常に重要かつ基本的な考え方が，次に述べる責任共有モデルである（AWSの場合，文献［2］参照）．

2.1 責任共有モデル

パブリッククラウドでは，クラウド事業者がホストOSや仮想化レイヤからサービスが運用されている施設の物理的なセキュリティまで，さまざまなコンポーネントを運用，管理，統制している．これにより，利用者の運用上の負担が軽減するというメリットを提供しつつ，クラウド事業者はその提供の責任を負っている．

一方で，クラウド事業者がいくら対策を実施していたとしても，利用者側の設定次第では非常に危険な状態を作り出してしまう可能性がある．そのため，クラウド事業者とその利用者がそれぞれ責任を負う範囲を明確化して分担し，協力して対策を行っていく必要がある．このような考え方が責任共有モデルである．

パブリッククラウドにおける責任共有モデルでは，サービスの種類によって利用者およびクラウド事業者の責任範囲が異なる（**図1**）．IaaS（Infrastructure as a Service）[*6]と呼ばれる仮想マシン[*7]を提供するサービスであれば，クラウド事業者は主にデータセンタなどの物理的なファシリティから，物理マシンやネットワークなどのハードウェア，ホストOSや仮想化レイヤの管理までを実施する一方で，ゲストOSやその上で動作するアプリケーションは利用者側が責任をもって管理を行う必要がある．PaaS（Platform as a Service）[*8]やFaaS（Function as a Service）[*9]のようにマネージドサービス[*10]と呼ばれているサービスでは，利用者側が責任を負う範囲がより狭くなっており，多くの管理をクラウド事業者側に任せる事が可能である．このようなサービスを適宜利用していくことで，付加価値の高いアプリケーションやビジネス領域に集中して企業のリソースを投下することができる．

2.2 クラウド事業者評価

前述した責任共有モデルにおいて，利用者はクラウド事業者と責任を共有することになるため，逆に言うと責任の一部を任せなければならず，利用者はクラウド事業者が責任を全うできると信頼をする必要がある．しかし，本当にそのクラウド事業者が責任を共有する相手にふさわしいのであろうか．例えば，クラウド事業者が本当に安定的にサービスを提供できるのか，どのようにしてセキュリティを担保しているのか，あるいは利用者がパブリッククラウドへアップロードしたデータが内部で不正にアクセスされることがないだろうか，といった点が懸念される．では，信頼の置ける事業者かどうかをどのようにして評価すればよいのだろうか．

パブリッククラウドやクラウド事業者の評価方法

＊4 パブリッククラウド：インターネットを介して誰でも利用ができるクラウドコンピューティングサービス．
＊5 ワークロード：CPU使用率などのシステムの負荷の大きさを表す指標．特にパブリッククラウドの分野では，クラウド上で実行されるOSやアプリケーションコードなどを含めたシステム自体を表すこともある．本稿では後者の意味で用いる．
＊6 IaaS：サーバ，ネットワークなどのハードウェアを仮想的に貸し出すサービス．利用者は借りたサーバやネットワーク上にOSやアプリケーションソフトウェアを設定して利用する．
＊7 仮想マシン：ソフトウェアによって仮想的に構築されたサーバなどのコンピュータ．

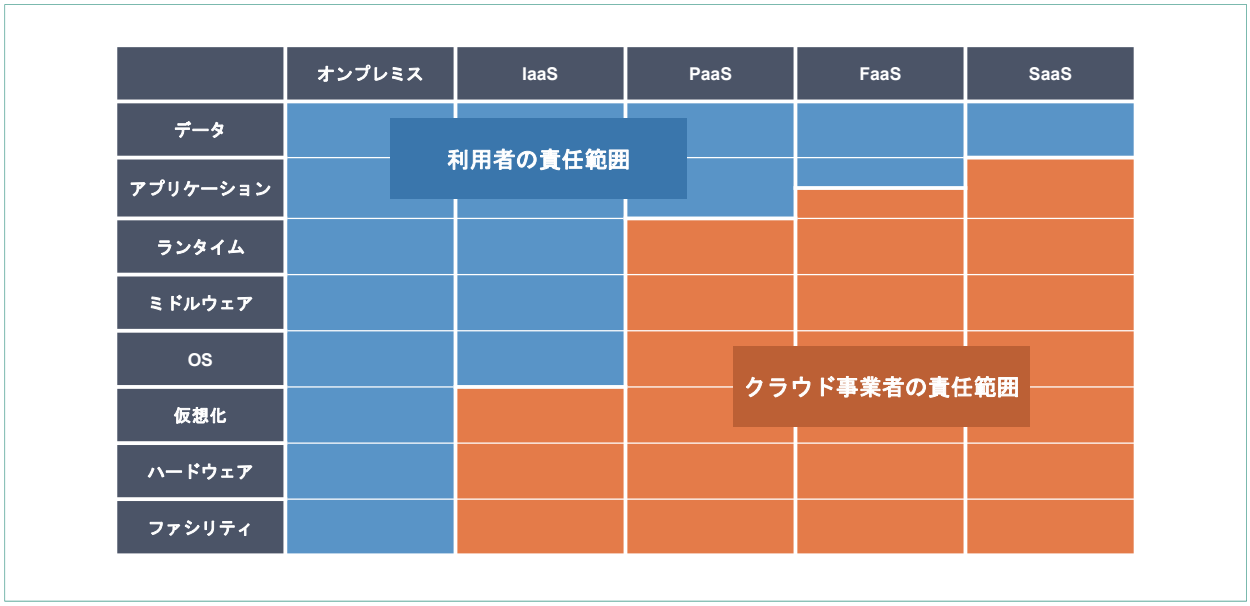


図1　パブリッククラウドのサービス種別ごとの責任範囲の違い

については，機能要件と非機能要件に大きく分けて考える必要がある．

機能要件については，クラウド事業者が提供するドキュメントやホワイトペーパー，サービスレベルアグリーメント（SLA：Service Level Agreement）[*11]を事前に確認することが非常に重要である．特にサービスの可用性やパフォーマンスといった項目は，ドキュメントに数値として明示的に記載されていることが多いため，具体的なシステム要件と照らし合わせて評価を行うことが可能である．

次に，非機能要件について評価する上で重要な判断材料として，公的機関による認定証や監査法人によるコンプライアンスレポートを活用することが有効である．これらの各種文書は，クラウド事業者が一般公開しているものもあれば，個別に秘密保持契約を結んで取得しなければならないものもある．多くのクラウド事業者は，下記のような国際機関による監査を受けており，その認証やレポートを入手することが可能である．

・ISO（International Organization for Standardization）[*12] 27017：2015 Certification
・PCI DSS（Payment Card Industry Data Security Standard）[*13] AOC（Attestation of Compliance）[*14] and Responsibility Summary
・SOC（Service Organization Controls）[*15] 2 Report

また，これらのレポートは，当該クラウド事業者が各種業界でのセキュリティ基準を満たしているかを判断するのにも役立つ．例えば，金融業界では公益財団法人金融情報システムセンター（FISC：The Center for Financial Industry Information Systems）によって安全対策基準が示されており，多くのクラウド事業者はこういった基準にどのように対応しているかという情報を公開している．

また，近年ではEUにおいて一般データ保護規則（GDPR：General Data Protection Regulation）が制定されたように，利用者のプライバシーを守るための法令の制定が活発である．これらの各地域や国で定められた法令を遵守したサービスをパブリック

＊8　PaaS：アプリケーションを実行するためのOSやミドルウェアを含むプラットフォームをクラウド上で貸し出すサービス．利用者は借りたプラットフォームの上でアプリケーションソフトウェアを作成して利用する．
＊9　FaaS：イベント駆動型のアプリケーション実行サービス．リソースの管理が不要のため，利用者はコードの記述に専念できる．一般的には実行時間に応じた課金モデルが採用されている．
＊10　マネージドサービス：クラウドサービスのうち，リソースのプロビジョニングや運用の大半をクラウド事業者の責任で実施しているサービス．クラウドコンピューティングサービスのうち，特にPaaSやSaaSを指す．

クラウド上で実装するために，パブリッククラウドでは物理的なデータセンタを，一般的にリージョンと呼ばれる国や地域ごとに分離し，それぞれのデータセンタにおいてサービスを提供している．これにより，データレジデンシー*16を明確化できるとともに，各国や地域において求められる法令遵守要件に応じてサービスをカスタマイズして提供することが可能となっている．

2.3 利用者側での対策

パブリッククラウドにおける利用者側の対策は，基本的にはオンプレミスで行っていた対策と同様のものが求められる．例えば，ソフトウェアに内在する脆弱性の管理や通信の暗号化を行うなどの対策はもちろんのこと，IaaSを利用する上ではゲストOSのセキュリティパッチを管理することや，ネットワークレベルでの攻撃を防ぐためにファイアウォールを設定することなどが必要である．

ところで，パブリッククラウドのメリットは，物理的なデータセンタやサーバの保守運用をクラウド事業者に任せられることだけなのであろうか．パブリッククラウドがここまで広く使われるようになった大きな要因に，PaaSと呼ばれるプラットフォームを提供する形態が挙げられる．PaaSであれば，IaaSで必要だった利用者側でのOSの管理も不要となる．さらに，PaaSの中でも特にFaaSと呼ばれるコードのみを管理する形態では，利用者側はミドルウェア*17の管理からも開放される．このように，オンプレミス（やIaaS）と比較して利用者が管理する部分が減少することは，セキュリティ対策として利用者が分担しなければならない責任が減少したことを意味する．つまり，PaaSやFaaSを活用すること自体が利用者側のセキュリティリスクを軽減させる効果がある．

逆に，クラウドならではのインシデント事例もある．クラウドではすべてのインフラストラクチャ*18はソフトウェア化されている．これまではサーバルームへ入室し，LANケーブルを抜き差ししなければ外部ネットワークと繋がらなかったものが，ボタンの押下1つで簡単に公開できてしまうようになったのである．このように，保守用の端末からボタン1つでインフラストラクチャを素早く大胆に変更できるようになったことはビジネス上の優位点となるが，一方でセキュリティにおいては大きな懸念となる．

クラウド事業者は，利用者の責任領域におけるセキュリティ対策に有用なサービスをいくつも提供している．例えば，AWSはAWS Well Architected フレームワーク*19というベストプラクティス集を公開している．また，AWS Trusted AdvisorやAWS Security Hubのように，利用者の責任領域において，ベストプラクティスに則ったかたちでシステムが運用されているかどうかをチェックするツールを活用することが非常に重要となっている．

3. ドコモにおけるセキュリティ統制

ここでは，ドコモにおけるセキュリティ統制の考え方や体制について述べる．

3.1 セキュリティ統制の体制

クラウドを利用する上でのセキュリティ統制の体制は**図2**のようなものとなっている．本図の情報セキュリティ部とは，セキュリティポリシーを作成・管理し，また，各システムがそのセキュリティポリシーを満たしたものかどうかを審査し，助言を行う全社的な組織である．

ドコモでは，通信ネットワークを構成する設備や通信サービスを提供する設備を自社で多く保有しており，また，これら以外にも依然として多くのワークロードをオンプレミス上で実行している．これらワークロードの実行に対しては，セキュリティ事故を避けるため厳格なセキュリティポリシーが適用さ

＊11 サービスレベルアグリーメント（SLA）：提供するサービスの品質保証．
＊12 ISO：国際標準化機構．情報技術分野の標準化を行う組織であり，電気および電気通信分野を除く全産業分野に関する国際規格を作成する．
＊13 PCI DSS：クレジットカード利用者の会員情報や取引情報を保護するために策定されたセキュリティ基準．
＊14 AOC：準拠証明書．PCI-DSSに準拠していることを示す証明書．
＊15 SOC：米国公認会計士協会により策定されたセキュリティ基準．System＆Organization Controlとも言う．
＊16 データレジデンシー：データの保管場所．

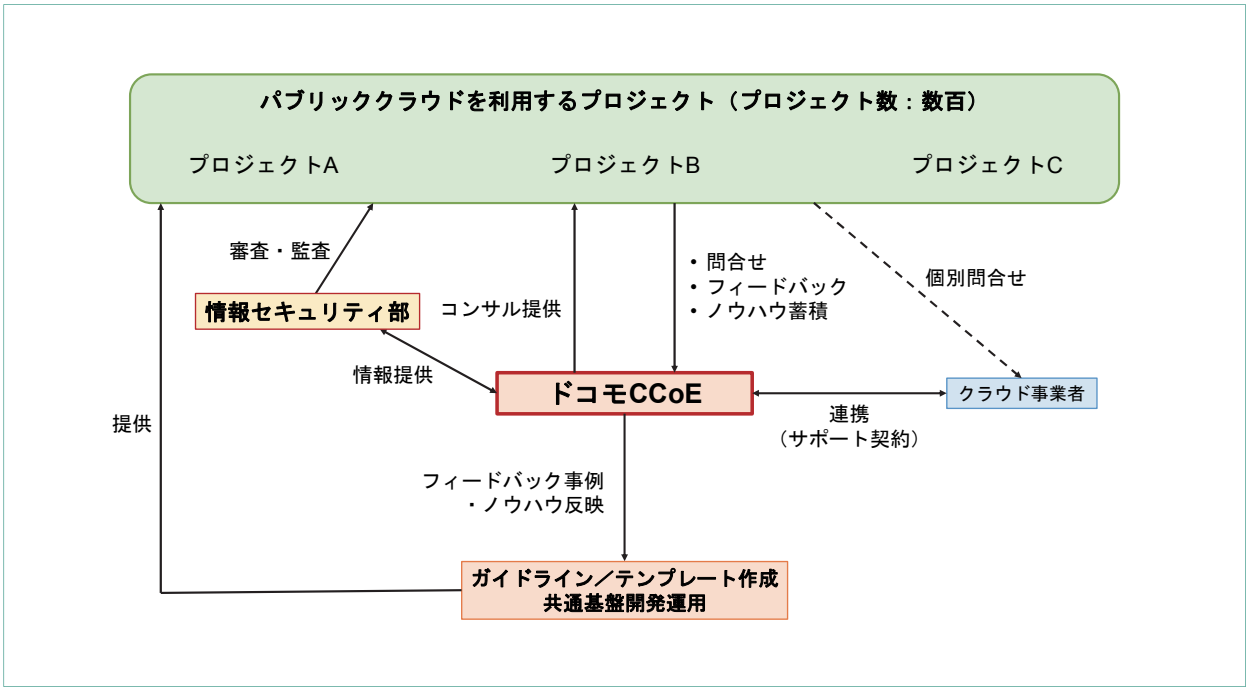


図2　ドコモ社内のクラウド統制の体制

れている．ただし，セキュリティポリシーはパブリッククラウドを利用したことにより緩和されるものではなく，パブリッククラウドを利用するかどうかに関係なく社内で構築されるすべてのシステムを対象として制定された一般的なものとなっている．そこで，パブリッククラウドの利用にあたっても，責任共有モデルなど特有の仕組みを理解し，その特性に応じたセキュリティ対策が必要とされるようになっている．

3.2　CCoEの役割と課題

前述のように，パブリッククラウドを利用する上で特有の考え方や対策を実施する必要性がある．そこでドコモでは，CCoE（Cloud Center of Excellence）というクラウド利用の支援に特化した部隊を発足させ，情報セキュリティ部へ情報提供を行うとともに，各プロジェクトに対してノウハウを提供したり，アーキテクチャやセキュリティに関するコンサルティングを行ったりしている．

しかしながら，パブリッククラウドを利用したシステムやプロジェクトの数は近年急速に拡大してきており，CCoEがすべてのシステムの構成を把握することが困難となってきた．また，パブリッククラウドを利用することの目的の1つとして，ビジネスを加速させることが挙げられるが，情報セキュリティ部やCCoEの統制強化が，各プロジェクトがサービスをいち早く市場に投入し，短いサイクルで機能を追加・改修していくことを阻害してしまう．したがって，ビジネス要件とシステム構成を各プロジェクトメンバが自身で把握し，セキュリティのリスクを自主的に判断する必要が出てきた．

そこで，CCoEから提供しているのが，ノウハウ集であるドコモ・クラウドパッケージとセキュリティチェックツールのScanMonsterである．

なお，CCoEの取組みについては本特集別記事を参照されたい［3］．

＊17 ミドルウェア：複数のアプリケーションから共通に利用される機能を提供するソフトウェア．
＊18 インフラストラクチャ：アプリケーションを実行するのに必要な物理的もしくは仮想的なデータセンタやサーバ，ネットワークなどの総称．
＊19 AWS Well Architected フレームワーク：AWSが公開している設計や運用に関するベストプラクティス．

4. 可視化とチェックツールの提供

各プロジェクトの開発者は，プロジェクトの開始にあたって，まずはドコモ・クラウドパッケージを用いてパブリッククラウド特有のノウハウを習得し，それを活用してシステムを構築する．しかし，各プロジェクトに対してヒアリングすると，「どのノウハウを適用すればよいのか分からない」「構築後にポリシーを満たせているかどうかがわからない」という声が多く聞かれた．このような課題に対して，CCoEではAWS環境を自動アセスメントするツール「ScanMonster」を開発した．

4.1 ScanMonsterの機能

ScanMonsterでは，AWS環境に対する70項目以上のアセスメントを実行することができる．アセスメント項目は，ドコモ・クラウドパッケージに記載されている内容や，CIS（Center for Internet Security）Benchmark*20，AWS Well Architected フレームワークなどいくつかの指標を参考に作成されている．例えば，「Root Account MFA」というアセスメント項目では，ルートユーザ*21に対してMFA（Multi-Factor Authentication）*22が設定されていないAWSアカウントをチェックすることができる．チェック結果を利用して，利用者はMFAが設定されていないルートユーザに対して設定を促すといった取組みが可能となる．また，「ACM Validation Method」は，AWS Certificate Manager*23において証明書発行時のドメイン検証に，DNS（Domain Name System）*24が利用されているかどうかを監査するためのアセスメント項目である．これはドコモ・クラウドパッケージへ記載している内容が基となっており，ドコモでのパブリッククラウド活用ノウハウを活かしたオリジナルの項目である．

ScanMonsterのアセスメント実行画面を**図3**に示す．利用者は，アセスメント項目を任意に選択し，ボタンを押すことでアセスメントを実行することができる．結果は「○」または「×」の2値で示され，利用者はひと目でアセスメント結果を判別できる．複数のアセスメント項目，または，すべてのアセス

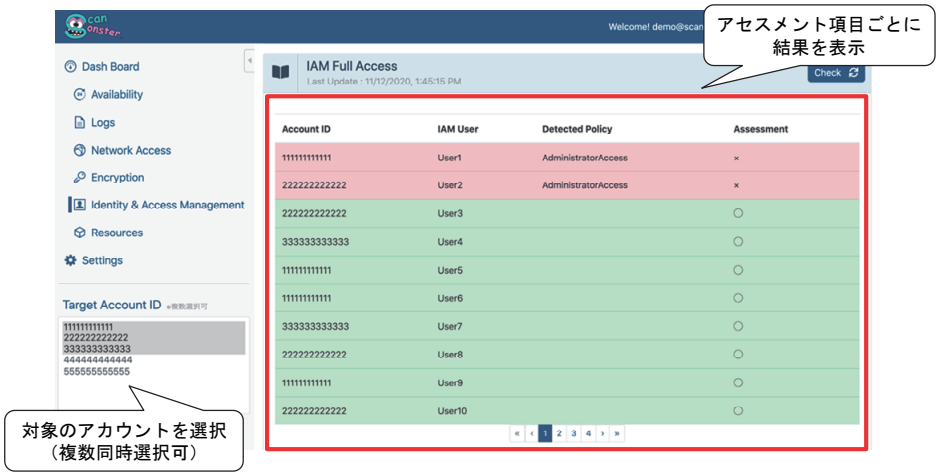


図3　ScanMonsterのアセスメント実行画面イメージ

*20 CIS Benchmark：米CISが策定しているセキュリティ基準．
*21 ルートユーザ：AWSアカウントへの完全なアクセスをもつアイデンティティ．MFAを設定して厳重に保護することがベストプラクティスとされている．
*22 MFA：多要素認証．複数の種類の要素により本人確認を行う認証方式．
*23 AWS Certificate Manager：AWSサービスで利用するSSL/TLS証明書を簡易に発行し管理できるサービス．
*24 DNS：インターネット上でドメイン名とIPアドレスの対応付けを管理し，相互に変換するサービスを提供する仕組み．

メント項目を同時に実行することも可能である．また，ユーザごとにあらかじめ設定されたアセスメント可能なAWSアカウントのうちから，複数のアカウントを選択して同時にアセスメントを実行することも可能である．

多くのアセスメント項目には，それが何を目的としたものか，アセスメントがOK/NGとなる条件，アセスメントをOKとするための対応手順などを記載したチュートリアルを用意している．ScanMonsterにおけるチュートリアルの表示画面を**図4**に示す．AWSに不慣れな利用者であっても，チュートリアルを読むことでアセスメント結果の理解と，ビジネス上のリスクの見積もり，対応を実施するかの判断が容易に行える．

4.2 ScanMonsterの構成

(1)サーバレスアーキテクチャの採用

ScanMonsterは，**図5**に示すようにそれ自体がAWS上で構築されている．また，構成要素にはAmazon EC2などのIaaS製品を使用せず，AWS Lambda[*25]やAmazon S3[*26]，Amazon DynamoDB[*27]，Amazon Cognito[*28]，Amazon CloudFront[*29]，AWS WAF（Web Application Firewall）[*30]といったサービスを採用してサーバレスの構成とした．

サーバレス構成としたことにより，2つの大きな特長が生まれた．

1つ目は，サーバレスの名前が示すとおりで，インフラストラクチャの管理が不要であるという点である．すべてのサービスがマネージドサービスであるため，サーバの死活監視や負荷に応じたプロビジョニング[*31]が不要である．

2つ目は，運用コストが非常に安価という点である．利用時間による料金が発生するのは，フロントエンドのデータを格納したAmazon S3とエンドポイントのアクセス制御のためのAWS WAF，ユーザの管理とアクセス許可のためのAmazon Cognitoのみである．AWS LambdaとAmazon DynamoDBについては，アセスメントを実行した際にのみ利用料が課金されるため，ScanMonsterを利用中でもアセスメントを行っていない期間，あるいは夜間など誰

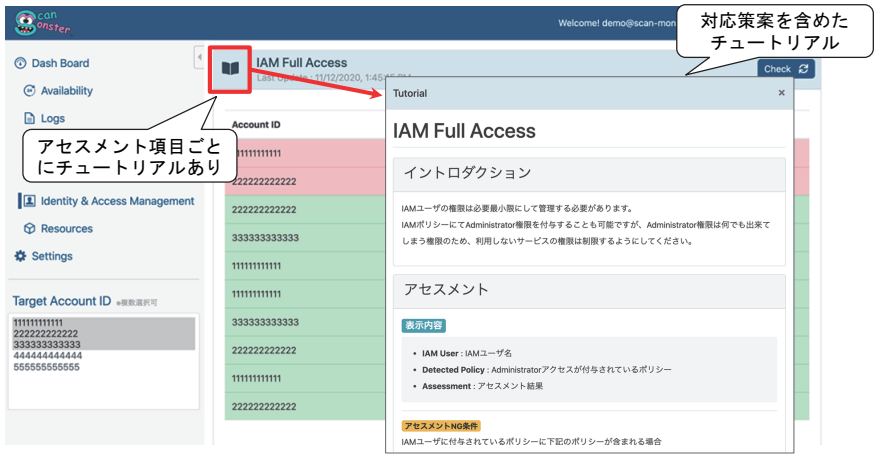


図4　ScanMonsterのチュートリアル表示画面イメージ

＊25 AWS Lambda：AWSが提供するPaaSの1つ．アプリケーションコードの実行環境が提供されており，利用者は作成したソースコードを登録してアプリケーションが実行できる．

＊26 Amazon S3：AWSが提供するオブジェクトストレージサービス．99.999999999％のデータ耐久性を実現するように設計されている．

＊27 Amazon DynamoDB：AWSが提供するNoSQLデータベースサービスの1つ．多量のリクエストを低遅延で処理可能なように設計されている．

＊28 Amazon Cognito：AWSが提供するPaaSの1つ．ウェブアプリケーションやモバイルアプリケーションに対して認証や認可，ユーザ管理機能を提供する．

もScanMonsterへアクセスしていない時間帯などは一切利用料がかからない．ドコモでの月当りのコストの実績は，数十円から数百円となっている（**図6**）．

(2)クロスアカウントアクセスによる複数AWSアカウントへのアセスメント

複数のAWSアカウントへの同時アセスメントは，IAM（Identity and Access Management）ロール*32

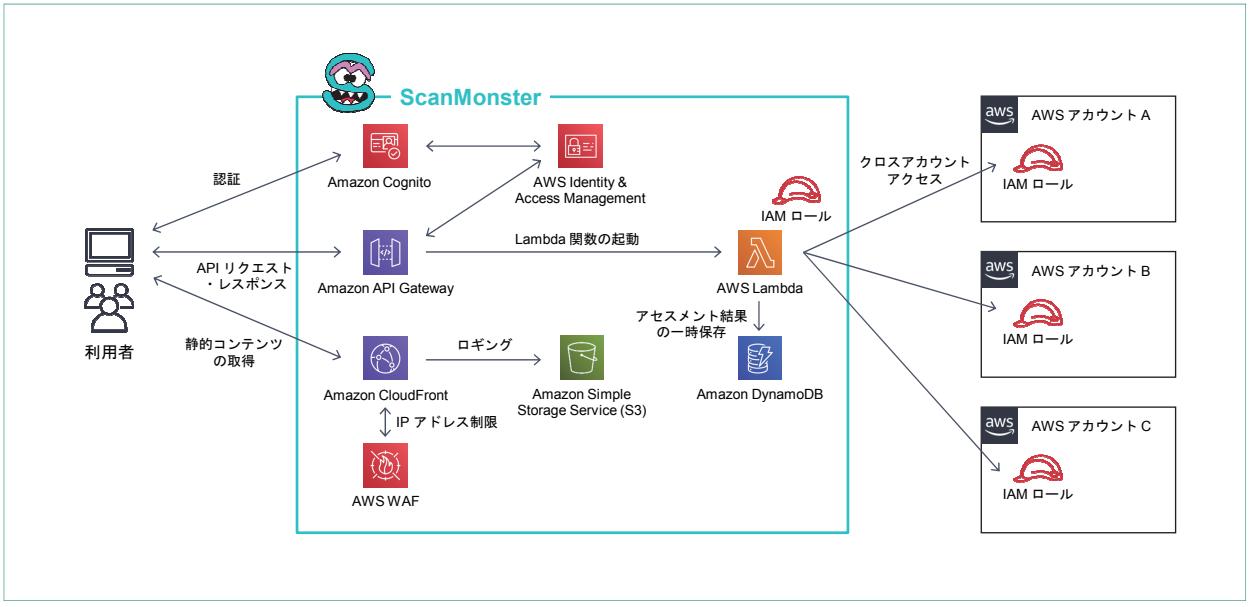


図5　ScanMonsterのアーキテクチャ図

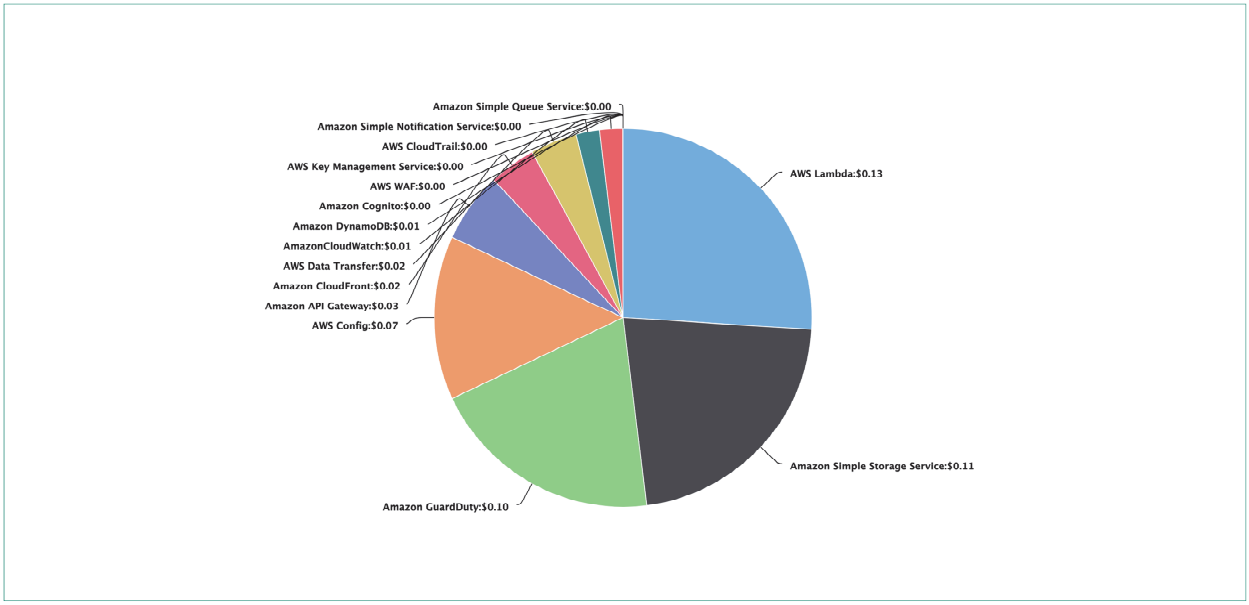


図6　ScanMonsterのコスト実績図

＊29 Amazon CloudFront：AWSが提供するCDN（Content Delivery Network，コンテンツ配信ネットワーク）サービス．

＊30 AWS WAF：AWSが提供するWebアプリケーション向けのファイアウォールサービス．

＊31 プロビジョニング：アプリケーションを実行するために必要となるサーバやネットワークなどのリソースの確保やそれらを動作させるための各種設定作業．

＊32 IAMロール：AWSにおけるアイデンティティリソースの1つ．任意の許可されたユーザ，アプリケーション，サービスなどにAWSリソースに対するアクセス権限を委任するために利用する．

の引受けによるクロスアカウントアクセスを利用している．アセスメント対象のAWSアカウント内に，ScanMonsterをデプロイしたAWSアカウントを信頼するIAMロールを設定することにより，ScanMonsterへのアクセス許可を行う．これにより，明示的にアクセス許可を与えたAWSアカウントのみがアセスメントを受け入れることを担保している．IAMロールの権限はアセスメントの実行に必要最低限の読取り権限のみとし，ScanMonsterをデプロイしたAWSアカウントからの不正な攻撃を防止している．さらに，ScanMonster内でもAmazon Cognitoのユーザを管理することで，利用者ごとにアセスメントが可能なAWSアカウントを設定することが可能である．

⑶IaC（Infrastructure as Code）[*33]による簡易なScanMonsterデプロイ

また，ScanMonsterは構成されるすべてのAWSリソースをAWS CloudFormation[*34]のテンプレートで記述しており，瞬時にScanMonsterをデプロイすることが可能である．これにより，アプリケーション開発時のテストや品質管理に，ScanMonsterの新しい環境を都度自動的に作成することが可能となっている．また，ScanMonsterは社外のお客様向けにも提供している製品であるが，この特長によりお客様自身の保有するAWSアカウント内にScanMonsterを簡単にデプロイすることができる．これにより，自社のAWS環境のアセスメント結果という非常に機密性の高いセキュリティ情報を，ドコモに開示することなく保管・利用ができる．

5. あとがき

本稿では，クラウドセキュリティの基本的な考え方と，ドコモでのセキュリティ統制の取組み内容について解説した．パブリッククラウドは今もなお進化を続けており，クラウドセキュリティの考え方や世の中の攻撃手法も刻一刻と変化をしている．大事なのは，パブリッククラウド上で一度システムを作ったらそれで終わりではなく，日々セキュリティアセスメントを行うとともに，そのアセスメントの内容も時代に合わせて日々アップデートすることである．アップデートを行うにあたって，クラウド事業者それ自身が提供するサービスを利活用するのはもちろんのこと，CCoEのようにクラウド活用の促進や，セキュリティ統制を行う専任の組織をつくって運用していくことは，進化のスピードが早いパブリッククラウドをうまく利用していく上で非常に重要である．

ドコモでは，数多くの社内プロジェクトで自律的にセキュリティアセスメントができるツールとしてScanMonsterを開発した．ガイドラインと併せてこういったツールを活用することで，ビジネスの速度を緩めることなく安全にパブリッククラウドを利用していける体制を確立した．今後は，ScanMonster自体の機能拡張としてアセスメント内容のカスタマイズ機能やガイドラインとの紐付け，現在のAWSに加えてGCPやAzureなどのマルチクラウド対応を検討している．また，アセスメント結果の情報セキュリティ部への集約や，定期的にアセスメントを実行するような仕組みの検討など，社内の体制やポリシーの改善にも取り組んでいきたい．

文　献

[1] Bloomberg：“Capital One Says Breach Hit 100 Million Individuals in U.S.,” Jul. 2019. https://www.bloomberg.com/news/articles/2019-07-29/capital-one-data-systems-breached-by-seattle-woman-u-s-says

[2] AWS：“責任共有モデル．” https://aws.amazon.com/jp/compliance/shared-responsibility-model/

[3] 守屋，ほか：“ドコモのパブリッククラウド活用とCCoEの果たす役割，”本誌，Vol.29，No.1，pp.6-12，Apr. 2021.

＊33 IaC：サーバやネットワーク，ストレージなどのインフラストラクチャの構成をコードとして記述して管理するという考え方．設定やプロビジョニングの作業を自動化できる．

＊34 AWS CloudFormation：AWSサービスの1つ．IaCを実現するために，テンプレートに記述されたAWSリソースのプロビジョニングや管理機能を提供する．

Technology Reports（特集）　パブリッククラウド　運　用　障　害
パブリッククラウド活用特集

大規模障害に備えるパブリッククラウド上でのシステム運用

イノベーション統括部　守屋（もりや）　裕樹（ひろき）

AWSをはじめとするパブリッククラウドが普及し，企業だけでなく官公庁，教育機関など多くの組織や団体にパブリッククラウドが利用されている．このように社会インフラとなりつつあるパブリッククラウドは，ひとたび停止すると社会全体に大きな影響を与えかねず，クラウドを利用したシステム運用の重要性が増している．ドコモでは長年大規模にパブリッククラウドを利用してきた経験から，大規模障害を想定したシステムの設計や運用ノウハウを蓄積してきた．また大規模障害発生時に会社内の情報連携をスムーズに行えるツールなどを開発してきた．これにより，会社全体としてパブリッククラウドの大規模障害を想定したシステム運用を行うことが可能になった．

1. まえがき

AWS（Amazon Web Services）*1をはじめとするパブリッククラウド*2が普及し始めてすでに10年以上経っており，企業だけでなく官公庁や教育機関などさまざまな組織や団体で，パブリッククラウドを利用したシステムの構築や運用がなされるようになってきた．

パブリッククラウドは，もはや単なるクラウドサービスではなく，社会全体を支えるインフラとして機能するようになってきており，ひとたびパブリッククラウドで大規模な障害が発生すると社会全体に影響を与えかねない状況となっている．

ドコモでは長年大規模にパブリッククラウドを利

＊1　AWS：Amazon Web Services社が提供するクラウドコンピューティングサービス．
＊2　パブリッククラウド：インターネットを介して誰でも利用ができるクラウドコンピューティングサービス．

用してきた経験から，パブリッククラウドの特性を最大限に活かしたシステム運用のノウハウを蓄積し，運用を支援するツールなどを開発しながら，会社全体としてパブリッククラウドをより活用できるように取り組んできた．本稿では，ドコモ以外の組織でもパブリッククラウド利用時の参考としていただくことを目的として，これらの取組みについて解説する．

2. システム障害に対する考え方

2.1 可用性の高いシステム構成

システムには障害がつきものである．壊れないシステムや障害が発生しない完璧なシステムは存在しない．この原則はオンプレミス*3でシステムをつくる場合でも，パブリッククラウドでつくる場合でも同じである．そのため，システムを構築する場合は，構成する各要素で障害が発生した場合でも，システム全体として少しでも可用性が上がるような構成をとることが重要になってくる．例えば，単一のサーバなどの装置が機能しなくなった場合に，バックアップ装置に切り替えてシステムの機能を維持するといった対応は，古くから取られてきた手法である．また，アプリケーション領域においても，システム全体の耐障害性を上げるための工夫として，非同期処理によるシステムの疎結合を図り障害影響範囲を限定する実装や，障害発生前提でのリトライ処理などの実装が多い．さらに，近年ではシステムを1つのモノリシック*4なサービスとして構成するのではなく，複数のマイクロサービス*5に分割して構成する動きが多くなってきており，障害発生時の影響範囲を限定することでシステム全体の可用性を上げるような取組みがなされている．

2.2 パブリッククラウドにおける責任共有モデル

パブリッククラウドを利用する場合，クラウド利用者とパブリッククラウド事業者の間でそれぞれの責任範囲を明確にし，双方でシステム全体の可用性を確保していく「責任共有モデル［1］」と呼ばれるモデルが採用されていることが多い．例えば，仮想マシン*6であれば，一般的にデータセンタや物理サーバ，ネットワーク，仮想化レイヤまでの領域はパブリッククラウド事業者の責任範囲であり，事業者によってこれらのインフラストラクチャ*7や物理装置，ネットワークなどの冗長化といった可用性を保つような対策がされている．一方で，仮想マシンOSやその上で動作するアプリケーションなどは，クラウド利用者側の責任で可用性を保つような実装をする必要がある．例として仮想マシンを挙げたが，近年クラウドサービスにおいてはIaaS（Infrastructure as a Service）*8だけでなく，PaaS（Platform as a Service）*9やSaaS（Software as a Service）*10が活発に活用されている．しかし，利用者とパブリッククラウド事業者間の責任範囲が異なるだけで「責任共有モデル」の考え方は同様に適用される．

多くのパブリッククラウド事業者は，利用者によるシステムの可用性を保つ設計を支援するための機能を提供しており，よくある構成パターンについては，ベストプラクティスとしてドキュメントやホワイトペーパーを積極的に情報配信している．そのため，利用者はこれらの機能を利用して，効率的に高可用性を実現するためのシステムを構築することが可能になってきている．また，PaaSやSaaSを利用する場合には，パブリッククラウド事業者側で，すでに高可用性を実現する構成がとられていることが多いため，利用者側は積極的にそれらの機能を活用してシステム構築を行うようになってきている．

＊3 オンプレミス：企業がシステムを構成するハードウェアを自社で保有し，自社で保守運用すること．
＊4 モノリシック：複数の機能を1つの大きなソフトウェアとして組み上げる構成．
＊5 マイクロサービス：複数の小さなサービスを組み合わせて1つのサービスを作り上げるシステムの開発技法のこと．
＊6 仮想マシン：ソフトウェアによって仮想的に構築されたサーバなどのコンピュータ．
＊7 インフラストラクチャ：アプリケーションを実行するのに必要な物理的もしくは仮想的なデータセンタやサーバ，ネットワークなどの総称．
＊8 IaaS：サーバ，ネットワークなどのハードウェアを仮想的に貸し出すサービス．利用者は借りたサーバやネットワーク上にOSやアプリケーションソフトウェアを設定して利用する．
＊9 PaaS：アプリケーションを実行するためのOSやミドルウェアを含むプラットフォームをクラウド上で貸し出すサービス．利用者は借りたプラットフォームの上でアプリケーションソフトウェアを作成して利用する．
＊10 SaaS：アプリケーションをクラウド上で貸し出すサービス．利用者はアプリケーションを直ちに利用できる．

3. 近年発生したクラウド事業者側での大規模障害

近年発生したパブリッククラウドの大規模障害の事例として以下のものがある．

⑴AWSの障害事例

AWSでは，2019年8月23日に空調設備の管理システム障害が原因で冷却システムが正常に稼働せずサーバがオーバーヒートを起こしてしまい，単一Availability Zone[*11]のEC2（Amazon Elastic Compute Cloud）[*12]やEBS（Amazon Elastic Block Store）[*13]に障害が発生した．また，EC2やEBSは，その他のサービスを構成するための基盤サービスとなっていることから，RDS（Amazon Relational Database Service）[*14]，Amazon Redshift[*15]，Amazon ElastiCache[*16]およびAmazon WorkSpaces[*17]などのマネージドサービス[*18]にも影響があったことが確認されている．単一Availability Zoneでの障害だったため，クラウド利用者によっては，複数Availability Zoneの冗長化構成をとることで影響を最小化したところもあったが，それでも一部設定によっては影響が確認されていたことが報告されている．

また，2020年4月20日にも東京リージョンでSQS（Amazon Simple Queue Service）[*19]やAWS Lambda[*20]などのマネージドサービスにおいて処理エラーの増加や遅延の発生などの障害発生が確認されており，これらを利用していた利用者への影響が確認されている［2］．

⑵Microsoft Azureの障害事例

Microsoft Azure[*21]では，2020年9月29日にAzure AD（Azure Active Directory）の認証エラーが世界中で観測される障害が発生した．Azure ADは開発者やユーザ，サービス間などの多くのサービスの認証認可を行うMicrosoft Azureのコアサービスであるため，これにより利用者がMicrosoft AzureやOffice 365が使えなくなるといった影響が確認されている．本来何重にもテストされてデプロイ[*22]されるはずの内部検証段階のアップデートが，SDP（Safe Deployment Process）システム[*23]の潜在的なバグにより，テストをバイパスして本番環境にデプロイされたことが原因と発表されている［3］．

⑶GCPの障害事例

GCP（Google Cloud Platform）[*24]では，2020年3月27日にアクセス制御・管理サービスであるCloud IAM（Cloud Identity and Access Management）に障害が発生した．Cloud IAMは多くのGCPサービスへのアクセス制御に利用されている共通サービスのため，他のサービスへも影響を及ぼす大規模障害となり，公式発表では障害の影響は14時間にも及んだ．原因はCloud IAMへの想定外の変更リクエストが発生したことで，Cloud IAM内部のキャッシュサーバのメモリが枯渇した結果，Cloud IAMへのリクエストのタイムアウトが発生していたことと発表されている［4］．

このように世界中ですでに多くの実績があるクラウド事業者のサービスでも，思わぬ形で障害が発生し，利用者が影響を受けることはあり，今後もゼロになることはない．

4. 利用者が考慮しておくべきこと

パブリッククラウドにおいてシステム障害が発生した際，利用者が考慮しておくべきことをいくつか挙げる．

4.1 障害発生を前提とする

全世界で多くの利用実績があるパブリッククラウ

＊11 Availability Zone：1つまたは複数のデータセンタ群．各Availability Zone間は物理的に独立している．
＊12 EC2：AWSが提供するIaaSの1つ．仮想マシンを提供するサービス．
＊13 EBS：AWSが提供するIaaSの1つ．ブロックストレージを提供するサービス．
＊14 RDS：AWSが提供するPaaSの1つ．Relational Databaseの機能をサービスとして提供する．
＊15 Amazon Redshift：AWSが提供するPaaSの1つ．データウェアハウスをサービスとして提供する．
＊16 Amazon ElastiCache：AWSが提供するPaaSの1つ．インメモリキャッシュをサービスとして提供する．
＊17 Amazon WorkSpaces：AWSが提供するSaaSの1つ．WindowsもしくはLinuxのデスクトップ環境をサービスとして提供する．
＊18 マネージドサービス：クラウドサービスのうち，リソースのプロビジョニングや運用の大半をクラウド事業者の責任で実施しているサービス．クラウドコンピューティングサービスのうち，特にPaaSやSaaSを指す．
＊19 SQS：AWSが提供するPaaSの1つ．メッセージキューの機能をサービスとして提供する．

ドにおいても障害が発生し，パブリッククラウドが進化しても障害をゼロにすることはできない．利用者がこの事実を十分に把握しておくことは非常に重要である．仮にこの事実を把握せずにパブリッククラウドを利用してしまうと，障害発生前提での設計や運用が十分にできていないことが要因で，大きな損失を出してしまうことも考えられる．まずは開発や運用担当者だけでなく，ビジネスオーナーや経営層を含めてこの事実を理解しておくことが重要である．

4.2 ベストプラクティスを活用する

前述のように障害をゼロにすることはできないが，障害による影響を低減することは可能である．多くのパブリッククラウド事業者は，可用性や信頼性を維持するためのシステム設計を利用者に推奨すると同時に，それらを利用者がより実現しやすいようにさまざまな機能やオプションを提供している．利用者はこれらの機能を利用して，パブリッククラウドに障害が発生した場合でも，システムをできるだけ継続して稼働させるような設計をすることが重要になってくる．幸いにもパブリッククラウドは世界中にすでに多くの利用者が存在しており，あらゆるユースケースで採用されてきた設計パターンがベストプラクティスとして公開されている．そのため，利用者は一から設計を考える必要はなく，それらのベストプラクティスを採用して設計や運用を行っていくことで，リスクの低減を図ることが可能である．

また，近年はPaaSやSaaS機能が充実しており，あらかじめパブリッククラウド事業者がベストプラクティスを踏まえて設計したサービスやプラットフォームを利用者が積極的に利用することで，耐障害性の高いシステムを構築するという流れが活発になっている．「サーバレス」と呼ばれる設計パターンはこの代表例である．サーバレスは，物理サーバや，IaaSに代表される仮想サーバを利用者が意識することなく，システムの設計や運用を行う手法であるが，本稿ではサーバレスの詳細は割愛する．これらの設計手法やサービスを活用することは，耐障害性の面だけでなく利用者の運用面でも負荷が低減されるというメリットがあるため，今後さらにこの流れが加速することは間違いない．

4.3 障害に備える運用設計をする

どのように対策してリスクを低減することはできても，やはり障害自体や障害の影響を完全にゼロにすることはできない．また，あらかじめ想定できる障害の場合は対策も十分に可能だが，想定外のケースや事象で発生する障害も多い．そのため，実際に障害が発生した際に迅速に原因を特定し，リカバリできるように運用を設計しておくことも重要である．具体的なポイントをいくつか挙げる．

(1)システムの可観測性をあげる

システム監視はパブリッククラウドにおいても非常に重要である．また特にパブリッククラウドを利用する場合，インフラストラクチャ構成がIaaS/PaaS/SaaSにまたがることも多く，アプリケーション構成においてもコンテナ*25やマイクロサービスの採用が一般化してきており，多くの環境に分散してシステムが配置されるようになっている．そのため，これまで以上にシステムの可観測性を上げる必要があり，それは単なる死活監視だけでなく，アプリケーションで処理される個々のリクエスト*26やメソッド*27レベルで処理を追跡する分散トレーシングや，ログや追跡結果を統合的に集約して可視化や分析を行える基盤サービスなどの採用が考えられる．これらの仕組みはクラウド事業者が提供するもの，3rd Party*28のサービスとして提供されているもの，

*20 AWS Lambda：AWSが提供するFaaSの1つ．アプリケーションコードの実行環境が提供されており，利用者は作成したソースコードを登録してアプリケーションが実行できる．
*21 Microsoft Azure：Microsoft社が提供するクラウドコンピューティングサービス．
*22 デプロイ：アプリケーションをそれらの実行環境に配置して展開すること．
*23 SDPシステム：Microsoftにて採用されている，ソフトウェアを安全にデプロイするためにその過程を管理するシステム．
*24 GCP：Google社が提供するクラウドコンピューティングサービス．
*25 コンテナ：コンピュータ仮想化技術の一種で，1つのホストOSの上にコンテナと呼ばれる専用領域を作り，その中で必要なアプリケーションソフトを動かす方式のこと．
*26 リクエスト：アプリケーションへの動作要求のこと．
*27 メソッド：GET，POST，PUT，DELETEといったHTTPメソッドのこと．
*28 3rd Party：第三者メーカのこと．

OSS（Open Source Software）[*29]として公開されているものなどさまざまであるため，目的にあったものを選択して活用していく必要がある．

⑵想定外の障害リスクを下げる

想定外の障害リスクをできるだけ小さくすることも，運用を改善していくためには重要である．クラウド上のシステムにおけるフェイルオーバー[*30]試験や，特定のクラウドサービスが停止した場合のリカバリ試験などを普段から実践しておくことは，想定外の障害リスクを下げるために効果がある．

ここで，パブリッククラウドならではの障害リスク低減手法としてChaos Engineeringを紹介する．Chaos Engineeringは，システムの本番環境で意図的に障害を発生させ，システム全体への影響を観測するという手法である．これにより，障害発生時にシステム全体に想定していない影響がないかを観測し，システムの耐障害性を向上させることができる．これは，インフラストラクチャをAPIで管理でき，作り直しがいくらでもできるクラウドならではの手法であり，米Netflix社は，日常的にこのような運用を実施することで，想定外の障害影響のリスクを下げる取組みを実践している［5］．なお，Chaos Engineeringは本番環境（実運用環境）で実施する手法のため，やみくもに実施するのではなく，すでに記載したシステム設計を適切に実践し，システムの耐障害性に十分に自信をもった上で実施する必要がある点に注意が必要である．

⑶クラウドサービスの各サービスの構成を理解する

パブリッククラウドを利用する場合に意外にも重要となるのが，クラウドサービスの各サービス自体の構成を理解することである．もちろんクラウドサービスは，セキュリティやコンプライアンスなどの関係で詳細な内部構造は公開されておらず，クラウドサービスによってはデータセンタの場所自体も非公開となっている．一方で，システム設計や運用時に利用者に考慮してもらうために，一部論理構成などが公開されているサービスなどがあり，それらを理解しておくことで障害発生時の原因特定やリカバリをスムーズに行える場合がある．

例えば，AWSの場合，Availability Zoneと呼ばれるデータセンタ群があり，各Availablity Zoneは物理的に独立している．この概念を理解すること自体は高可用性設計において必須である．またPaaS/SaaSについては，AWS自身がユーザと同様の視点でこれらのAvailability Zoneにまたがってサービスを構成した上で，利用者に展開しているものも多い．このことを理解しておくことで，特定のAvailability Zoneで障害が発生した際に，それぞれのクラウドサービスに影響が出るのか，そうでないのかの把握が可能となり，また利用者側で障害発生時に特定のAvailability Zoneへのトラフィックを切り離すオプションがあれば，それにより影響を最小化できる可能性がある，といった判断が可能となる．また仮想サーバのEC2は，多くのPaaS/SaaSサービスの基盤となっているサービスである，ということを理解しておくことで，仮にEC2に障害が発生した場合に「他のサービスにも影響が出る可能性がある」ということが予測でき，少なくとも身構えることができる．そのほかにも各パブリッククラウドサービスの構成が公開されている場合があるため，それらの構成が理解できているほど，障害発生時の対応はスムーズに実践できる．

5. Support Visualizerの開発と提供

最後に，ドコモ内で最も利用されているクラウドサービスの1つであるAWSを利用する場合において，ドコモが実践している障害発生に対する取組みにつ

＊29 OSS：ソースコードが無償で公開されており，誰でも再利用や改変が行えるソフトウェア．

＊30 フェイルオーバー：システムに障害が発生した際に自動的に冗長化された待機システムに切り替える仕組み．

いて解説する．ドコモでは，900以上（2020年12月時点）のAWSアカウントを運用しており，さまざまなワークロード*31でAWSを活用しているが，2019年ごろAWS東京リージョンで発生した障害の影響を少なからず受け，これによりいくつかの課題が判明した．これらの課題を解決するための取組みとして，AWSのサポートケース情報（後述）を集約するSupport Visualizerというシステムを構築している．

5.1 AWS大規模障害で判明した課題

前述したとおり，ドコモではさまざまなワークロードでAWSを活用している．当然ながらワークロードによりシステム要件やアプリ利用者数，データ量が異なるのでシステム設計や運用がシステム単位で独立していることが多く，利用するクラウドサービスも異なってくる．そのため，ひとたびクラウドサービスで障害が発生した場合に，それらの影響がどの程度発生するのかはシステムによって異なる．実際にこれまで大規模障害が発生した際に，全く影響がなかったシステムもあれば，クラウドサービス利用者やアプリ利用者への影響が出てしまったシステムも存在する．これらの障害発生時に，個別のシステムや会社全体の運営に対して次のような課題が判明した．

⑴会社全体での情報収集

障害による個別システムへの影響は必ずしも同じではないため，会社としては個別のシステムへの影響度合いをいち早く認識し，適切なアナウンスなどを実施する必要がある．しかし，実際には各システムの運用は独立しており，また障害発生時の現場は原因特定と復旧作業に追われているため，すぐに情報を集約することは難しい．もちろん最優先すべきは，システムを迅速に復旧させ，利用者への影響を最小化することではあるが，そのような中でも会社は社会的責務として適切に情報の配信を行っていくことが重要である．これまで発生した大規模障害発生時はこれらの両立が非常に難しく，会社全体での情報収集に課題が残った．

⑵影響範囲の特定と対処

個別のシステムでの課題も浮彫りになった．システム障害自体はパブリッククラウド利用時に限らず常に発生し得るため，大規模障害時にも通常のシステム障害発生時と同様に対処を行っていくことになるが，その際クラウド自体の障害に起因するものなのか，個別のシステム固有の問題で発生したものなのかが分からず，原因特定と対処に時間がかかってしまう事象が発生した．

パブリッククラウド事業者の多くは，自身のサービスステータスを公開しており，障害発生時にはそれらのステータスを見ることでおおよその影響範囲が確認可能である．ただし，これまでのクラウド障害の経験上，これらのステータスでは必ずしも発生しているすべての障害を示しているわけではなく，実際にはクラウド利用者からの申告で障害発生が判明するケースや，障害がクラウド事業者から事後報告されるケースもある．

5.2 Support Visualizer

AWSの場合，AWS自体が提供するサービスで障害が発生し，利用者システムに影響が出た場合は，サポートケースと呼ばれる問合せチケット*32にてAWSに報告をするような運用が可能である．これにより，AWS側には障害の影響範囲に関する情報が集まるため，影響範囲の特定が可能になる．

一方で利用者であるドコモは，各アカウントが独立して運用されているため，サポートケース上では別プロジェクトの障害影響を把握することができず，プロジェクト同士の直接的な情報交換のやりとりが

＊31 ワークロード：CPU使用率などのシステムの負荷の大きさを表す指標．特にパブリッククラウドの分野では，クラウド上で実行されるOSやアプリケーションコードなどを含めたシステム自体を表すこともある．本稿では後者の意味で用いる．

＊32 問合せチケット：個々の問合せやそれに対する返答などを管理する単位．

必要となる．ただし，すでに述べたように障害発生中は現場レベルでは自身のシステムの復旧対応などで精一杯のため，そのような情報交換を実施する余裕がないのは事実である．これらは会社全体として情報を集約する場合でも同じことがいえる．そこで，社内で起票されたサポートケース情報を集約し，社内の誰でも閲覧できるシステムとして，Support Visualizerを開発した．

⑴アーキテクチャ

Support Visualizerのアーキテクチャを**図1**に示す．

サポートケース情報はAWSアカウントごとに独立しているが，サポートケース情報を取得できるAPIがAWSより提供されているため，本システムはそれを経由してサポートケース情報を集約する．集約された情報は，Elasticsearchによって検索・分析できるようにインデックス化してデータベースに格納される．集約された情報はKibana[*33]のダッシュボード[*34]を利用してCCoE（Cloud Center of Excellence）[*35]による閲覧・分析が可能で，合わせて緊急度の高いサポートケースが起票された場合にはSlack[*36]などでCCoEに通知するように設定されている．

また社内利用者向けには，集約されたサポートケース情報の閲覧用にポータルを展開しており，キーワードやサービス，緊急度，サポートケース情報更新時間などでフィルタリングをして，社内のサポートケース情報が閲覧できるようにしている（**図2**）．

⑵解決が期待できる課題

Support Visualizerの展開により，クラウド上のシステム障害発生時に個別のシステムからサポート

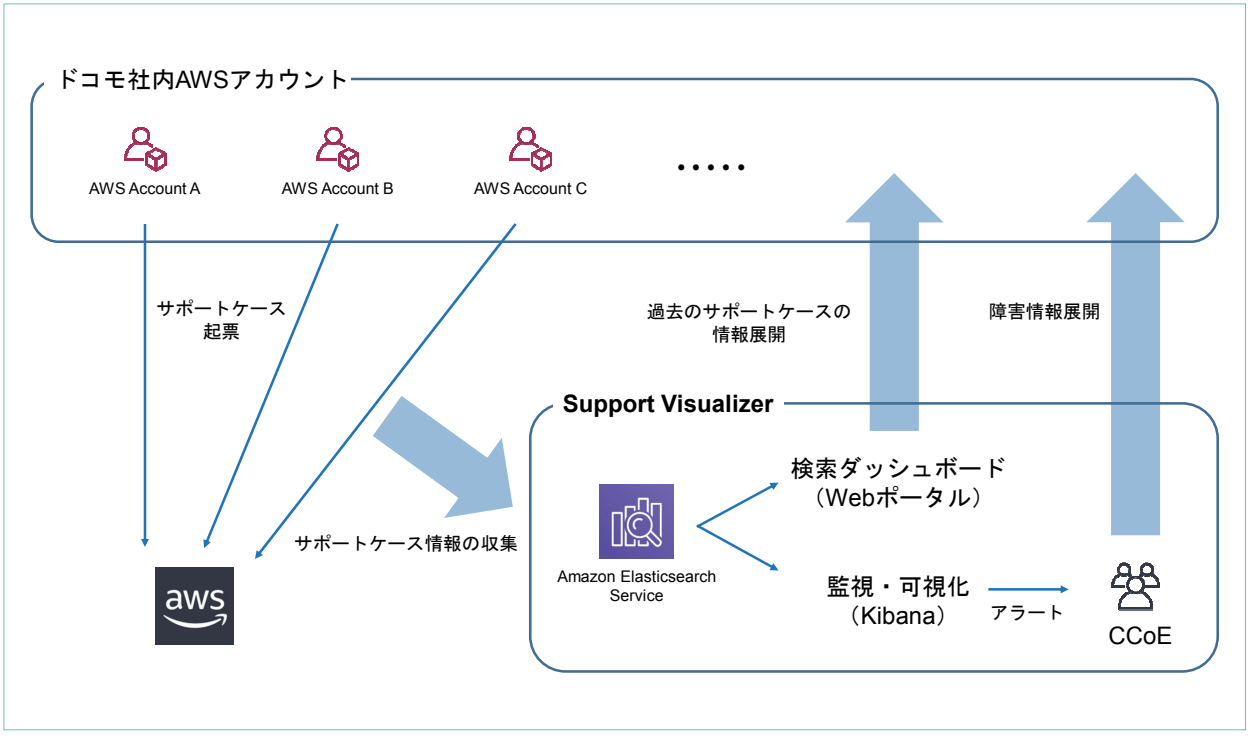


図1　Support Visualizerのアーキテクチャ

＊33 Kibana：Elastic社により開発されているオープンソースのデータ可視化ツール．
＊34 ダッシュボード：情報を集約する画面．
＊35 CCoE：企業においてクラウドの活用を成功させるために，ベストプラクティスの確立や必要な制度，ガバナンスなどを作成し，社内に展開していく専属のチーム．
＊36 Slack：Slack Technologies社が提供するビジネスチャットツール．

図2　Support VisualizerのWebポータル用のダッシュボードイメージ

ケースが起票された場合，社内のほかの利用者も含めてその内容が確認できるようになる．これは特にクラウドの大規模障害発生時に有効で，各システム運用者が個別システムの障害対応に追われている中でも，それらのシステムから起票されたサポートケースが自然と集約される．これにより，会社がシステムへの影響範囲の全体像をスムーズに把握することができると同時に，個別システムの運用者が自身のシステムで受けている影響が個別の事象なのか，クラウド全体で発生している事象なのかを特定することができる．また，すでに同様の事象が他システムでも発生している場合，それらの解決策についても確認できる可能性がある．ドコモでは，さまざまなワークロードで900以上のAWSアカウントを利用しているため，クラウド事業者から発出されるステータス情報だけでは取得しきれない，実際の現場で発生している影響も把握することが期待できる．

⑶副次効果

Support Visualizerの副次効果として，普段のクラウド利用時の開発や運用での技術的な問合せなどが集約されるため，結果として社内のクラウド利用時の技術ノウハウが集約できるという点が挙げられる．今後は集約したサポートケースの傾向分析やそこからのノウハウ集約を能動的に行う仕組みなどを取り入れて，各システムにフィードバックし，より効率的なクラウド利用を促進していきたい．

6. あとがき

本稿では，パブリッククラウドの大規模障害を見

据えたシステム運用について解説した．クラウドやコンテナ，マイクロサービスといったように技術やアーキテクチャが進化しても，障害をゼロにするシステムをつくることは不可能である．そのため，少しでも障害の影響を少なくする取組みを継続し，それらのベストプラクティスを醸成して会社全体でパブリッククラウドをうまく活用していけるように促進していきたい．

文　献

［1］ 総務省：“クラウドの特性とセキュリティ．”
https://www.soumu.go.jp/ict_skill/pdf/ict_skill_2_3.pdf

［2］ AWS：“東京リージョン（AP-NORTHEAST-1）で発生したAmazon EC2とAmazon EBSの事象概要，” Aug. 2019.
https://aws.amazon.com/jp/message/56489/

［3］ Microsoft Azure：“RCA - Authentication errors across multiple Microsoft services and Azure Active Directory integrated applications（Tracking ID SM79-F88），” Sep. 2020.
https://status.azure.com/status/history/

［4］ Google Cloud：“Google Cloud Infrastructure Components Incident #20003.”
https://status.cloud.google.com/incident/zall/20003

［5］ Netflix：“The Netflix Simian Army，” Midium, Jul. 2011.
https://netflixtechblog.com/the-netflix-simian-army-16e57fbab116

Technology Reports

クラウド　MEC　アプリケーションデザインパターン

MECを活用したアプリケーションデザインパターン

イノベーション統括部　秋永（あきなが） 和計（よしかず）

近年，5Gネットワークの整備が進み，ドコモでも2020年3月より5Gサービスを開始している．5Gには高速大容量，低遅延，多数端末同時接続という3つの大きな特長があり，その中でも高速大容量，低遅延に関しては，従来モバイルでは難しかったサービスを実現できると大きく期待されている．そのためMECによって，高速大容量，低遅延を活かすシステムを構築しようとする取組みが注目されている．ドコモでも「ドコモオープンイノベーションクラウド」というクラウドサーバと，それらを「クラウドダイレクト」というモバイルネットワーク内に直接接続するサービスが2020年6月から提供されている．これらのサービスを活かすために，アプリケーション構築時のアーキテクチャを考察する際に，必要な機能に対して用いられるアーキテクチャのテンプレート「MECアプリケーションデザインパターン」を提案する．本稿では，代表的なパターンとその効果について解説する．

1. まえがき

近年，第5世代移動通信システム（5G）ネットワークの整備が進み，ドコモでも2020年3月より5Gサービスを開始している．5Gには高速大容量（eMBB：enhanced Mobile BroadBand），低遅延（URLLC：Ultra Reliable and Low Latency Communications），多数端末同時接続（mMTC：massive Machine Type Communications）という3つの大きな特長があり，これらを活用することでさまざまなアプリケーション，ソリューション，産業にインパクトを与えると言われている．その中でも，高速大容量，低遅延に関しては，従来モバイルでは難しかったサービスを実現できると大きく期待されている．

そのためMEC（Multi-access Edge Computing）と呼ばれているネットワーク内に配置するクラウドによって，高速大容量，低遅延を活かすシステムを構築しようとする取組みが注目されている．ドコモ

でも「ドコモオープンイノベーションクラウド」というクラウドサーバと，それらを「クラウドダイレクト」というモバイルネットワーク内に直接接続するサービスが2020年6月から提供されてきた．これらのサービスを用いると，5Gの特長を活かしたさまざまなアプリケーションやソリューションの提供が可能になると期待されている．

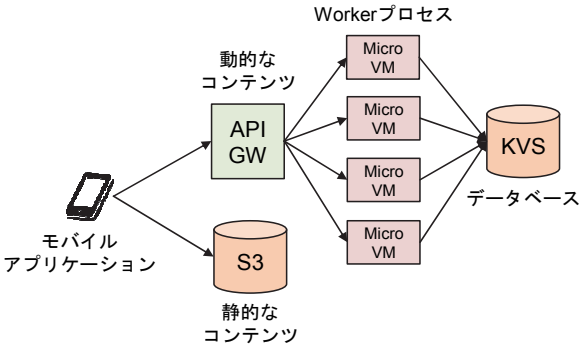
これらのサービスを活かすために，「MECアプリケーションデザインパターン」を提案する．これらを参照することで，開発者はアプリケーションを構築する上で，機能をクラウドおよびMECで最適配置した設計ができるようになると考えられる．本稿では，アプリケーションデザインパターンとMECについて述べた後，代表的なパターンとその効果について解説する．

2. アプリケーションデザインパターンとは？

アプリケーションデザインパターンとは，アプリケーション構築にかかわるアーキテクチャを考察する際，必要な機能に対して用いられるアーキテクチャのテンプレートに相当するものであり，代表的なユースケースを基に一般化されたアイデア集であり，アプリケーションアーキテクチャとして用いることができる．

本デザインパターンでは，一般的なデザインパターンと同様に個別のアプリケーション全体像を議論するのではなく，部品として議論されることから再利用性が高く，アプリケーションのアーキテクトが直ぐに新しい機能を取り込みやすく，考察しやすくしている．

例として，Webサービスを構築する上でのパブリッククラウド*1上でのアプリケーションデザインパターンを**図1**に示す．図1では，パブリッククラウドでよく用いられる典型的なサーバレスアプリケーションのアプリケーションデザインパターンを示している．動的なコンテンツと静的なコンテンツを分けてストアすることや，スケールアウトするためのWorkerプロセス*2を細かく配置すること，そ



MicroVM：非常に小さな単位で高速に起動できる仮想マシン

図1　アプリケーションデザインパターン（サーバレス）の例

＊1　パブリッククラウド：インターネットを介して誰でも利用ができるクラウドコンピューティングサービス．
＊2　Workerプロセス：ある一定の役割をもって起動され，その役割を終えると終了する単一のプログラム．

れらからのアクセスにKVS（Key Value Store）*3を用いることでデータベースのボトルネックを回避していることなどが一目で分かるようになっており，このデザインパターンを踏襲して自らのアプリケーションを設計することで，スケーラビリティのあるモバイルアプリケーションを実装することができる．このように設計上のノウハウのパターンとして整理しているのがアプリケーションデザインパターンである．

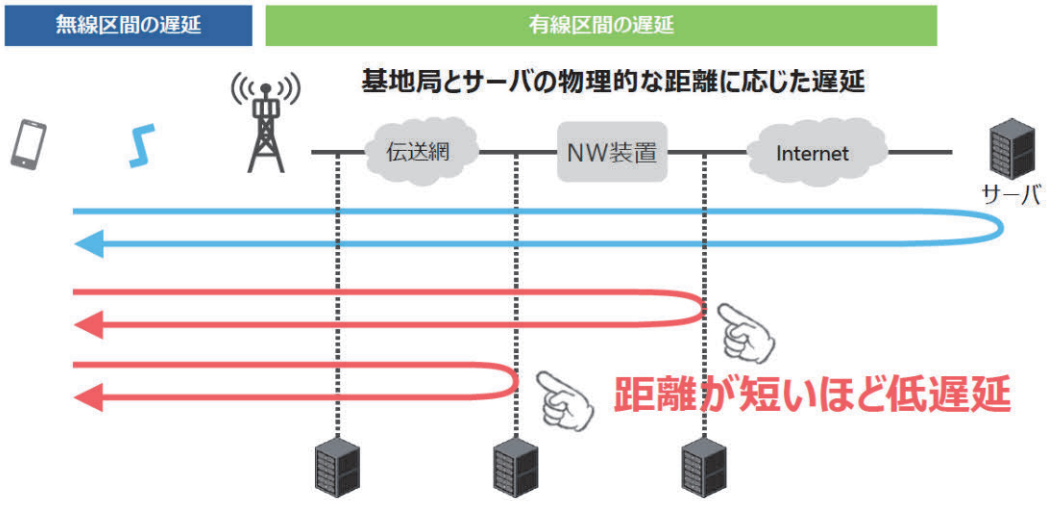
3. MECとは？

5Gネットワークにおいて，その低遅延性を利用したいという機運が高まっているが，無線区間の5G化だけでは低遅延を実現できない．システムとして低遅延を実現するためには，ネットワーク全体を見た上で，往復の伝達距離を短くし，経由する装置の数を少なくする必要がある．そのため，5Gネットワーク内部にコンピューティングリソースをもつことでその解決を図ろうとしている．これがMECである．MECは無線区間からできるだけ近い場所にコンピューティングリソース（仮想マシン）を置くことで，実現される（**図2**）．一方で，無線に近い（基地局に近い）場所になればなるほど，その遅延時間を短くすることができるが，基地局数は膨大であり，それらに対して必要な仮想マシンを設置しようとすると，必然的に数が多くなり，経済的に非現実的となる．逆にインターネットに近い場所になれば集約が図られるが，遅延は大きくなるため，需要と供給のバランスをみて適切なリソース配置が重要となる．

3.1 MECの4つの特長

MECでは，主に以下の4つの特長が利用者の観点で期待されている．

①トラフィック最適化，ネットワークでの折返し
端末同士のP2P（Peer to Peer）通信時に，地理的に近いサーバにアクセス，もしくはトラ



出典：https://www.nttdocomo.co.jp/corporate/ir/binary/pdf/library/presentation/200825/200825_qa.pdf（P.21）

図2　MECのネットワーク内の位置

*3　KVS：Key（鍵）とValue（値）の2つをセットにしてデータを簡易的に管理することに特化した，高速化されたデータベース．さらに複数サーバでの分散処理にも対応していることが多く，膨大な入出力を一度に処理することができるため，処理のボトルネック回避によく用いられる．

フィックを経由させることで，非常に低い遅延での通信が可能になり，かつサーバからは大容量のデータを流すことができる．ゲームなどで応答時間の要求条件が厳しいものに対して利用が期待できる．

②低遅延，ゆらぎの抑制

地理的に近いサーバにアクセスさせることで，5Gの特長を活かした低遅延での接続が可能になる．また，インターネットとは異なり閉域網での提供になるために，遅延におけるゆらぎの要因が少なくなり，ゆらぎに弱いストリーミングなどで利用が期待できる．

③セキュア・プライベートネットワーク構築

インターネットなどの外部ネットワークに出ることがないため，安全な接続を提供できる．SIM認証でのプライベート接続なども可能になるため，秘匿性の高い情報を扱うネットワークとしての利用が期待できる．

④コンピューティングパワーの提供，端末機能補完

端末では処理ができないような高度なAI処理や高精細な3D画像の生成など，負荷が大きな処理をエッジコンピューティング[*4]側で実施することができる．ゲームや3D CAD（Computer Aided Design），XR[*5]などでの利用が期待できる．

アプリケーションデザインパターンでは，このようなMECの特長を活かした具体的なアプリケーションへの組込み方法が議論されている．

3.2 アプリケーションデザインパターンでのMECの前提条件

MECは，地域に分散してサーバを構築するという性質上，対障害性や堅牢性の面で，集約型データセンタで運用されるシステムとは異なる．そこで，MECではDesign for failureと呼ばれる，パブリッククラウドコンピューティングにおける，故障することを前提とするシステムの構築法を踏襲する．例えば，MEC上でのデータ永続化は保証されない．これは，データの永続化のためには相当数の冗長性をもつ必要があるが，この冗長性が分散化されたシステムでは取りにくいためである．そのため，データ永続化はMECレイヤではなく，上位のパブリッククラウドや他のストレージサービスで実施するのが好ましい．

ほかにも，MECにおける前提条件は下記のものがある．

- 堅牢性，可用性は共に低い．
- IasS（Infrastructure as a Service）[*6]を提供する．
- データの永続化が期待されるようなPaaS（Platform as a Service）[*7]を極力もたない．
- 一方で，コンテナ[*8]管理サービスのようなポータビリティを向上させる機能は具備する．
- DNS（Domain Name System）[*9]サービスが提供される．

4. MECアプリケーションデザインパターン

以下では，具体的なアプリケーションデザインパターンと利用方法について，代表的なものをいくつか解説する．

4.1 高速大容量アップロードパターン

5Gの高速大容量を活用したアプリケーションが具備する機能の1つに，従来モバイルネットワークが苦手としていたアップロードがある．アップロードの速度が格段に上がったことにより，その活用を考えることができるが，インターネットへの通信は遅延の増大や，遅延時間の揺れ，それによるTCP（Transmission Control Protocol）[*10]のACK（ACKnowledgement）[*11]遅延によるスループットの低下などが発生し，十分

---

*4 エッジコンピューティング：ユーザの近くにエッジサーバを分散させ，距離を短縮することで通信遅延を短縮する技術．

*5 XR：VR，AR，MRといった仮想空間と現実空間との融合で新たな体験を提供する技術の総称．

*6 IaaS：サーバ，ネットワークなどのハードウェアを仮想的に貸し出すサービス．利用者は借りたサーバやネットワーク上にOSやアプリケーションソフトウェアを設定して利用する．

*7 PaaS：アプリケーションを実行するためのOSやミドルウェアを含むプラットフォームをクラウド上で貸し出すサービス．利用者は借りたプラットフォームの上でアプリケーションソフトウェアを作成して利用する．

な速度を出すことができない．そこで，MECを用いて，高速大容量のアップロードを実現させる．

⑴課　題

高速5Gのネットワークを利用して，大容量のアップロードを高速化したい．また，アップロードのTCPのACKパケットの遅延や，ゆらぎ，上位サーバの影響などからアップロード作業のみの独立性を高め，安定させたい．さらに，大容量アップロードを同時に多数受け付けたいという課題がある．

⑵デザインパターン

5Gのアップロード能力を最大限に活かすため，MECローカルサーバでアップロードを終端し，一時ストレージに格納する．そして，格納内容をキューイング*12し，Workerプロセスが別途一時アップロード先から回収し，パブリッククラウドのストレージで永続化する．一時ストレージに格納している段階で，別途プロセスを走らせることも可能（例：AIによる画像・映像処理）である．最も近いアップロードサーバとの接続には，DNSでの近傍サーバ検索サービスを利用する．また，ストレージへの一時的な格納は負荷分散にも効果があるため，同じ箇所からの大量アップロード対応（例えばスタジアムからの一斉アップロード）などで上位の回線の帯域が十分にない場合も対応可能と考えられる．アップロードのプロセスだけを分離して実装できるので，既存システムへの組込みが容易である．なお，可用性を担保するためにMEC上にはLB（Load Balancer）*13を配置し，アップロードサーバーを二重化しておくことが望ましい（**図3**）．

4.2　超低遅延メッセージング，ステータス管理パターン

5Gの低遅延性を活かし，複数端末での同期を行った複数人数でのゲームや，XRの同時体験，メッセージの交換などができる．この同期を実現する上でMEC上にKVSのPub/Sub機能*14を実装することで，

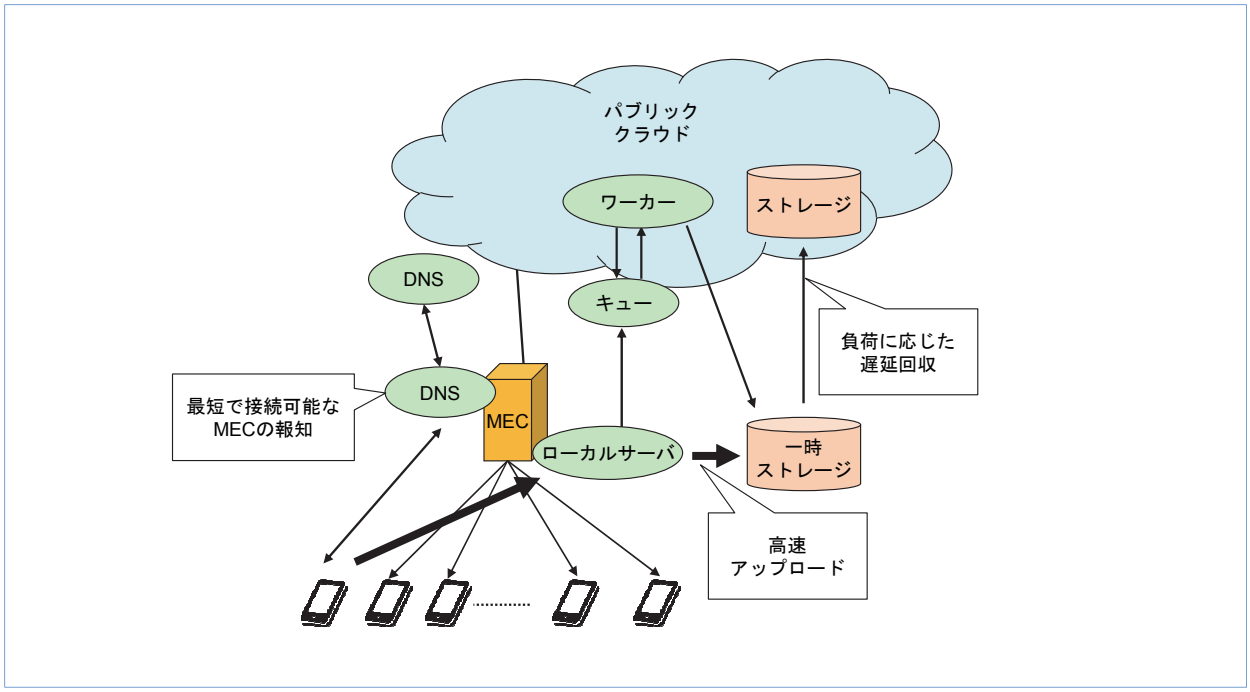


図3　高速大容量アップロードのデザインパターン

*8　コンテナ：コンピュータ仮想化技術の一種で，1つのホストOSの上にコンテナと呼ばれる専用領域を作り，その中で必要なアプリケーションソフトを動かす方式のこと．

*9　DNS：IPネットワーク上のホスト名とIPアドレスの対応付けを行うシステム．

*10　TCP：インターネットで標準的に利用されるIPの上位プロトコル．接続相手やデータ到着の確認・フロー制御・データの重複や抜けの検出などを行うことでIPの補完の役割を果たし，信頼性の高い通信を実現する．

*11　ACK：データの受信ノードが正常に受信（復号）できたことを送信ノードに通知する受信確認信号．

非常に汎用性の高いメッセージングが行えるようになる．

⑴課　題

超低遅延のKVSやそのPub/Sub機能をMECに置くことで，低遅延の同期を実現したい．また，対戦型ゲームやオープンワールド系ゲームの端末同士の状態同期，IoTデバイスの状態同期などを超低遅延で実現したい．他にも，ゲームにおけるワープ*15などのゲーム性を損なう挙動は排除したいという理由から，一定以下の同期遅延でサービスを提供したいという課題がある．

⑵デザインパターン

メッセージングや一時的なステータス管理などを超高速で行うため，MEC上にRedisなどのインメモリDB*16を配置する．これにより，端末間のメッセージングや，アプリケーションの一時的な状態管理・同期などを超高速で実装することができる．これらのインメモリDBはnsオーダで情報を配布，参照することができる．これに関しても近傍インメモリDBとの接続には，DNSでの近傍インメモリDB検索サービスを利用する．Pub/Sub機能を使った場合は近傍端末同士の超低遅延でのメッセージの交換が可能であり，また同様にKVSを使ったアプリケーションの状態管理などが可能である．これらのインメモリDBを使うことで，自分がプレーしている地域のゲームスコアの超高速集計や，ゲームランキングの集計（リアルタイムリーダーボード*17）などが可能である．また，複数システムの連携のためのサービスバス*18や，ネットワークゲームのプレイヤー同士の状態同期，リモートロボットの同期などに利用可能である（**図4**）．

⑶注意点

インメモリDBは認証が存在しないケースがあるので，実装時にはメッセージの暗号化やネットワークの分離などが必要である．

4.3　Webサービスキャッシュパターン

Webでのリアルタイム更新の方法として，Webソケット*19を用いた方法がある．こういったサービスを実装する場合は，その遅延やトラフィックの負荷分散などが大きな課題となる．この課題解決にもMECが有効である．

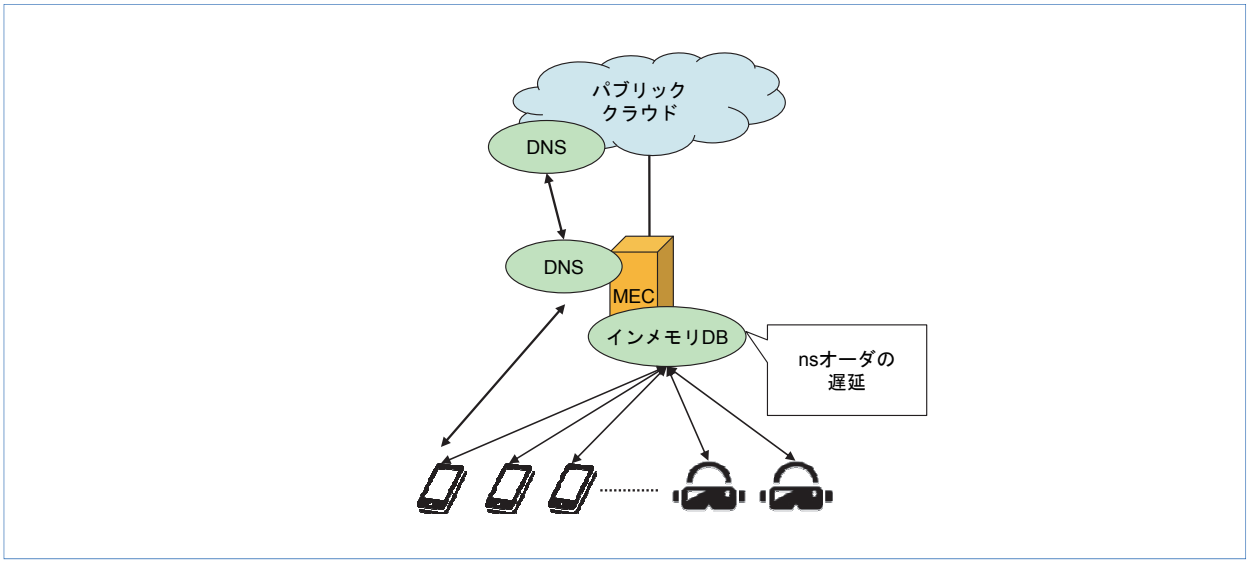


図4　超低遅延メッセージング，ステータス管理のデザインパターン

＊12 キューイング：待ち行列を作成し，処理する順番や処理する内容を一時的に保存しておくこと．

＊13 LB：通信トラフィックの負荷分散を行う装置．

＊14 Pub/Sub機能：Publish（出版）とSubscribe（購読）の機能で非同期のメッセージのやり取りをする機能．Subscribe側にはPublish側で送信されたものが全員に配布されるため，1：Nのメッセージの配布に適している．

＊15 ワープ：2者以上が参加しているゲームなどで，お互いの位置情報の同期がずれることにより，プレイヤーが突然消え，別の所から現れる現象．ゲーム性を損ねる場合が多く，ゲーム提供者側としては避けられるなら避けたい現象．

⑴課　題

動的なコンテンツの更新をできるだけ低遅延で行う（Webソケットなど）ことや，高いレスポンスのサービスを実現したいという課題がある．また，Web上での静的なコンテンツの読出し負荷の軽減を行いたいなどの課題もある．

⑵デザインパターン

Webサービスにおけるキャッシュ機構をMEC上にもつことにより，Web三層構造[*20]のうち，プレゼンテーション層とアプリケーション層をMEC側に置いて高速化を実現できる．また，データベース層についてもリードレプリカ[*21]やキャッシュ機構をMEC上にもつことで高速化を実現できる．注意したいのは，データベース層をパブリッククラウド側にもって来ないと，永続化が難しくなり，また広域での一貫性（Consistency）が保てなくなるため，キャッシュをアプリケーション層の近くに置いて高速化を実現する．

データベース層に関しては，MySQL（My Structured Query Language）[*22]のリードレプリカや，マルチマスタ[*23]なども負荷軽減という目的には有効である．しかし，低遅延を期待する場合は，データベース層の遅延はネットワークの遅延よりも大幅に大きいので，そのバランスに注意したい（**図5**）．

⑶注意点

アプリケーション層は顧客の近くに自動配置したいため，ポータビリティを考慮してコンテナ化などの工夫をするのが望ましい．また，Webのキャッシュにはインメモリキャッシュ（KVS）を用いることで，超高速レスポンスが可能になる．その場合の書込みについては，永続化の問題を考慮する必要がある．

このパターンは，実装は容易だが，MECサーバ数が増えると管理コストが増える可能性があるため，注意が必要である．また，コンテナ管理ソリューションなどを利用し，実装をパッケージ化しつつ，必要最低限のアプリケーションを配置することが，システムを効率よくかつコストを抑えるための注意点となる．

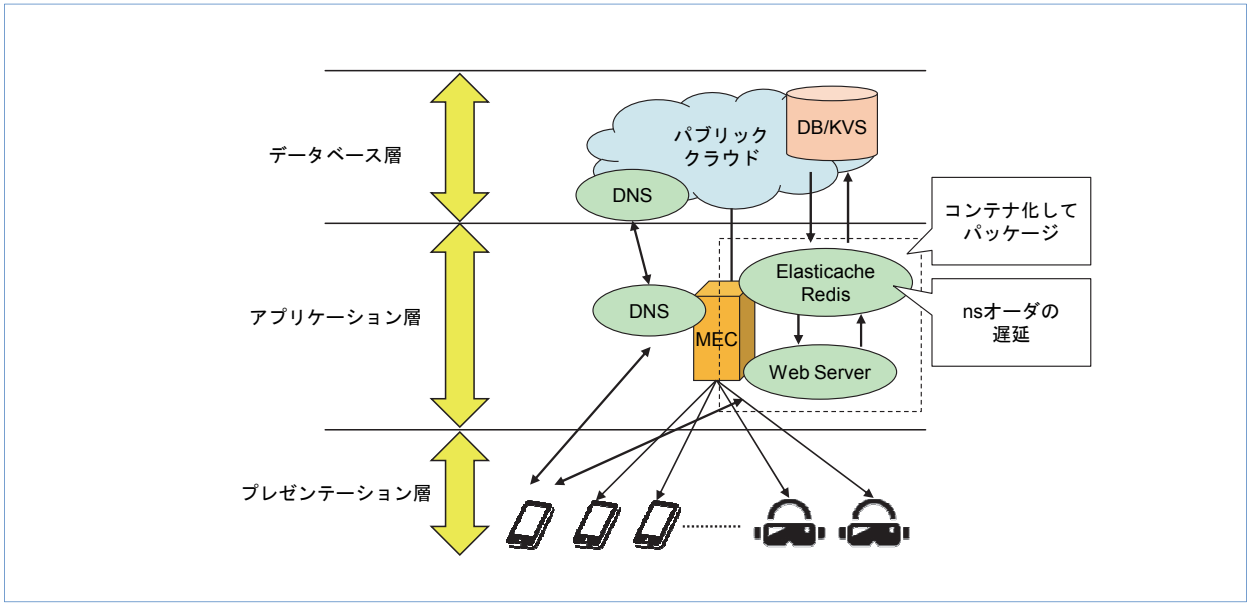


図5　Webサービスキャッシュのデザインパターン

＊16 インメモリDB：データをメモリ上に展開保持して処理することで，情報の取得に対して高速な応答を可能にしたデータベース．

＊17 リーダーボード：ゲームなどでランキングの上位者や，自分の順序が記載されているボード．

＊18 サービスバス：複数システムを連携させる場合に，お互いの状態の連携のためや，次のシステムに処理を引き継ぐためのメッセージのやり取りをする機構．

＊19 Webソケット：Web上で双方向のメッセージをやりとりするために，定められている技術規格．RFC（Request For Comments）6455として定められている．

4.4 IoTデバイス用軽量プロトコルの実装パターン

IoT機器におけるセキュリティの問題は，インターネット網からのアクセスを遮断し，キャリア網内に低負荷・低セキュリティのプロトコルを留めることで解決する．インターネットへの直接的な接続を分断でき，かつ多数のデバイスからの処理を低遅延で処理できるので，利便性が高い．

(1)課　題

IoTデバイスには，SSL（Secure Sockets Layer）*24などを実装できない小型のものなどもあるため，軽量プロトコルを採用したいが，同時にセキュリティも実現したいという課題がある．MQTT（Message Queuing Telemetry Transport）*25をTLS（Transport Layer Security）*26で実装する例などもあるが，IoT機器の電池消費量を抑えたい場合，通信の暗号化は負荷が大きいため，より認証や暗号化の少ない軽量なプロトコルを利用したいという課題がある．

(2)デザインパターン

IoT機器向け軽量プロトコルMQTTなどを安全にモバイルネットワーク内で終端し，MEC外に出る際に加工や暗号化を行い，安全にIoT機器を管理できるようにすることができる．これにより，インターネットからの攻撃の回避や安全なデバイスの管理システムを構築することができる（**図6**）．

4.5 その他のデザインパターン

そのほか考案されているデザインパターンを**表1**に示す．今後MECを有効活用する具体的な方法として，デザインパターンを増やしていきたい．

5. MECアプリケーションデザインパターンとソリューションの関係

MECアプリケーションデザインパターンは，MECを有効活用する上での具体的な機能を実現するための便利な利用パターン（ベストプラクティス）のみを示している．従って，アプリケーションもしくはソリューションを構築する際に，これらのデザインパターンを組み合わせることで実現する．その包含関係を**図7**に示す．

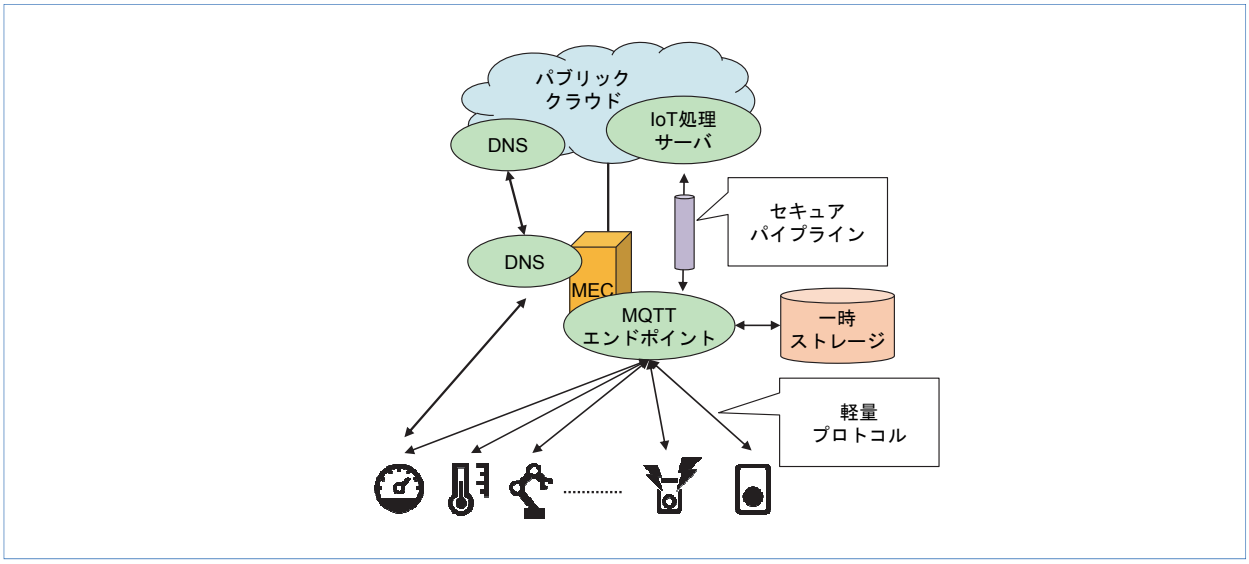


図6　IoTデバイス用軽量プロトコル実装のデザインパターン

*20 **Web三層構造**：Webシステムの構成要素をプレゼンテーション層，アプリケーション層，データ層の3層に分割し，独立したモジュールとして設計する手法．

*21 リードレプリカ：データベースの読込み専用のコピー．読込みや検索負荷の軽減のために本体のデータベースからのコピーを常に保持する．

*22 MySQL：最もよく使われているオープンソースのリレーショナルデータベース管理システム（RDBMS：Relational Data-Base Management System）のうちの1つ．

*23 マルチマスタ：データベースの高信頼化や高性能化の方式の1つで，複数の同一の機能をもつ接続先をもつことができるシステムを指す．複数の接続先があっても，参照されるデータは同じであり，利用できる機能に制約がない．

表1　その他の代表的なMECアプリケーションデザインパターン

パターン名	課　題	デザインパターン
CDNパターン	動画やストリーミングサービスにおいて，オリジンサーバ（配信サーバ）への負荷軽減をしたい． オリジンサーバへのネットワークのゆらぎなどにより品質の劣化を防ぎたい． 超高速・超大容量でのサービス提供は上位ネットワークほど負荷が集中し，品質が劣化しやすい． できるだけ映像をリアルタイムで提供したい．	通常のCDNシステムと同様にDNS側で最も近い場所を返すようにすることで最も近いサーバを判断することができる． CDN側では最も近い場所のキャッシュを取りに行くことができる．各CDNサーバは変更がある場合に，オリジンサーバ（例：パブリッククラウド上のWebサーバや，ストリーミングサーバ）から最新情報を取得する． ストリーミングでHLSなどを使っている場合も同じロジックで適用可能なため，汎用性が高い．
折返しによるP2Pトラフィックの最適化パターン	ローカルエッジネットワークにおける折返し通信により，P2Pトラフィックを最適化し，安定した高品質のサービスを提供したい． 例えばビデオ通話などをローカル限定で折り返すことにより，最適化を図りたい．	5G/4GのローカルNW上で折返しトラフィックを網内で最短ルーティングすることで低遅延での端末間通信を可能にする． MEC上にシグナリングサーバを配置することでWebRTCやVoIP（SIP）などを実装することが可能になる．シグナリングサーバを広域に展開する場合はMEC上でなくても良い．
ローカルセキュアネットワークパターン	ローカルネットワークを端末間で実現し，VPNなどを使わずにセキュアなネットワークを構築したい（ファイルサーバなど）．	VPNサーバをMEC上に配置することで，好きなNWを作ることができる．また，高速大容量を実現できるため，ファイルサーバなどへのアクセスも安全かつ容易に構築できる．
地理的制約を利用したセキュリティ実現パターン	特定の地域外からのアクセスを防止し，よりサービスを安全に提供したい．	MEC上に特定の地域のみにファイルサーバを配置し，そのシステムには特定地域のシステムからのみサーバのアクセスを許可する． DNSで地域のMECサーバを特定し，そのサーバにアクセスする． MEC上にはその地域でしかアクセスできない情報，もしくは認証方式を用いてアクセスする． パブリッククラウド上にストレージ，もしくはDBを配置し，堅牢性や可用性はそちらで担保する． 可用性を担保するためにMEC上ではLBを配置し，二重化しておく．

CDN：Content Delivery Network　　VoIP：Voice over Internet Protocol
HLS：HTTP Live Streaming　　VPN：Virtual Private Network
SIP：Session Initiation Protocol　　WebRTC：Web Real-Time Communication

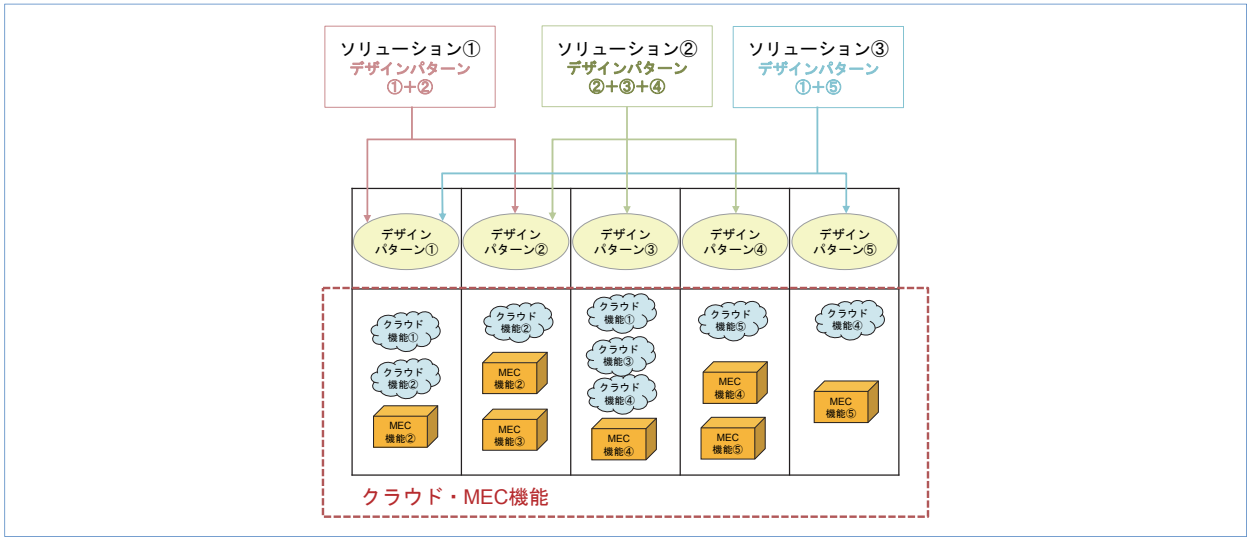


図7　アプリケーションデザインパターンとソリューションの関係

＊24 SSL：ネットワーク上のアプリケーション間，主にWWWブラウザとWWWサーバとの間で通信の暗号化，データ改ざんの発見を行うためのプロトコル．
＊25 MQTT：Pub/Sub型の軽量なメッセージキュープロトコル．IoTの各種デバイスとサーバ間でのメッセージ交換に使用される．
＊26 TLS：SSLをインターネット標準技術として規定し，拡張されたセキュリティを確保するためのプロトコル．SSLと比べ，暗号アルゴリズムやエラーメッセージ規定などが拡張されている．

6. あとがき

本稿では，MECをより便利に使っていくためのアプリケーションデザインパターンの重要性について解説した．MECを有効利用するためにはこれらのベストプラクティスを集積し，その知見を再利用することで，5G時代の新しい付加価値サービスの創造につながっていくと考えている．今後の機能提供方針として，デザインパターンそのもののバリエーションを増やすことで，より広範なアプリケーションに適用できるように努めていくとともに，よく利用されるデザインパターンをより活用しやすくするために，PaaS機能（デザインパターンを実現するための機能を自ら実装せずに簡単に使えるようにする機能）を検討し，機能具備していきたい．

Technology Reports

AI　音声認識　対話システム

電話での受付や見守りを自動化する「AI電話サービス」

サービスイノベーション部　川瀬 智子
5G・IoTビジネス部　小栗 伸
サービスデザイン部　斉藤 優樹

クラウド型コンタクトセンタシステムが普及し，コンタクトセンタを導入する事業者が増加しているが，オペレータの人材不足が問題となっている．そこで，定型の電話応対および応対後の事務処理の自動化を目的に，音声認識による本人確認を特徴とする電話応対自動化サービス「AI電話サービス」を開発した．これにより，予約や申込みの受付，高齢者の見守りといった業務の自動化が可能となる．

1. まえがき

顧客満足度向上を目指す上で，コンタクトセンタなどの電話窓口は，ITリテラシーを要さない顧客接点として意義が大きい．近年，クラウド型コンタクトセンタ*1が普及し，コンタクトセンタを導入する事業者が増加している．しかし，多様化する顧客に対して分かりやすく迅速に応対し，顧客満足度を向上させるためには，業務知識に加え，コミュニケーションスキルや通話後業務を遂行するためのITスキルがオペレータに要求されるが，それを満足するだけの人材が不足している．また人員配置の観点からは，繁忙期・閑散期に柔軟に人員を調整する必要がある．それに加え，新型コロナウイルス感染への対策として，コンタクトセンタのオフィスの人数低減も求められている．このような中，人工知能（AI）を活用した電話応対業務の自動化への需要が高まっている．海外では2018年ごろからAIで電話業務を支援するサービスの提供が進み，国内でも2020年には電話での申込み受付にAIを導入する事例が相次いで発表された．従来オペレータが担っている応対のうち，定型の応対をAIが代行することで，オペレータ稼働を非定型の応対に集中させることができる．また，通話後業務をAIが代行することで，オペレータに求められるITスキルを限定できるため，ITスキルを持ったオペレータの人材

＊1　クラウド型コンタクトセンタ：顧客応対のためのシステムで，自社でサーバを保有するのではなく，ネットワーク上のサーバを利用して運用するもの．

不足の軽減に繋がる．

ドコモでは，総合お問合せの電話窓口（総合センタ）の自動音声応答装置（IVR：Interactive Voice Response）*2に，「音声認識IVR」[1]の機能をすでに実装している．これは，用件を話すと，適切な専門センタへ自動接続するものであり，AIエージェントサービス「しゃべってコンシェル*3」および「my daiz*4」[2]で蓄積された音声対話サービスのノウハウが活用されている．音声認識IVRの導入により，オペレータに接続されるまでの待ち時間削減や，総合センタでの電話応対時間削減，専門センタへの転送業務削減の効果が出ている．

音声対話技術を，さらに電話応対業務を効率化するソリューションとし，かつ顧客事業者に提供するために，ドコモは「AI電話サービス」というクラウドサービスを新たに開発した．本稿では，AI電話サービスのサービス概要とその仕組み，多様なユースケースに対応するための音声認識技術に関する取組みについて解説する．

2. サービス概要

AI電話サービスは，自治体や小売店，飲食店，コールセンタをもつ企業といった事業者での利用を想定したサービスである．**図1**に示すように，受電でのユースケースとしては，サービス申込み／変更や，よくある問合せへの対応，飲食店や配車などの予約受付などがある．受電だけでなく架電にも対応しており，高齢者の在宅確認や体調確認による見守り，サービスの案内，予約や入金案内の確認・リマインドなど，コンタクトセンタの業務にとどまらない広い用途に適用できる．また，これまでオペレータが応対していた内容のうち，定型の応対業務を自動化するのみならず，音声対話技術とRPA（Robotic Process Automation）*5の連携により，通話後の業務も自動化する．例えば，対話内容のログに基づい

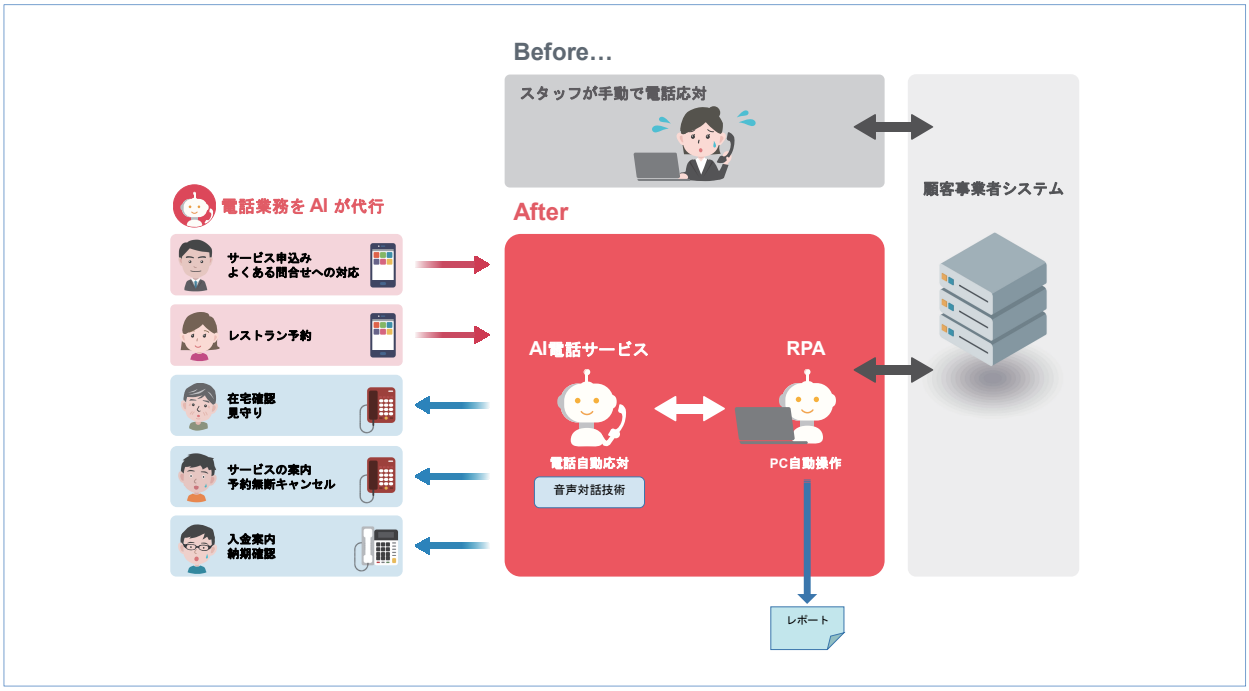


図1　AI電話サービスの概要とユースケース

*2　自動音声応答装置（IVR）：電話で「○○の場合は○番を押してください」のように音声で案内するシステム．
*3　しゃべってコンシェル：キャラクターとの会話や，会話を通じた電話発信，アラーム設定，乗換検索，占いを提供する，スマートフォンやタブレット上で動作する音声対話エージェント．
*4　my daiz：ユーザに合わせた幅広い情報を提供する，スマートフォンやタブレット上で動作する音声対話エージェント．
*5　RPA：定型業務を自動化する仕組み．

たレポートの自動作成や，顧客事業者のシステムとの連携が可能である．AI電話サービスの導入には，人材確保の問題の軽減に加え，24時間365日対応可能といった効果もある．

3. システム構成

AI電話サービスのシステム構成を**図2**に示す．「AI電話コアアプリケーション」が，ドコモの対話技術を提供する「ドコモAIエージェントAPI（Application Programming Interface）*6」[2]とクラウド型のコールセンタサービス「Amazon Connect」を連携させることにより，電話応対自動化機能を実現している．また，電話を利用するため，音声インタフェースの機能を音声認識エンジンにより提供している．

ドコモAIエージェントAPIは，あらかじめ決められた対話シナリオに沿ってAIが受け答えする機能を提供しており，顧客事業者自身で柔軟に対話シナリオを作成できる点が特長の1つである．

Amazon Connectは呼制御機能を提供しており，これによりAI電話サービスを利用する顧客事業者は，呼制御サーバの管理が不要になり，拡張も容易というメリットが得られる．オペレータへの転送機能もあるため，AIによる自動応対が難しいケースが発生した場合にはオペレータによる対応に切り替えられる．一方，Amazon Connectを採用する制約として，使用できる音声合成*7エンジンは「Amazon Polly」のみとなる．Amazon Pollyでは，AIの音声として特定の人物の合成音声を再現することはできず，日本語の場合，提供されている話者は男女1名ずつであるため，選択肢はそのどちらかのみとなる．ただし，選択した話者の声の範囲で話速や間，音量のチューニングにより，例えば重要な言葉をAIにゆっくり大きく話させるといったことは可能である．

AI電話コアアプリケーションは，Amazon Connectから取得したユーザ音声を音声認識エンジン*8に逐次送り，認識結果として発話テキストを逐次受け取る．音声認識エンジンでユーザの発話の終端を

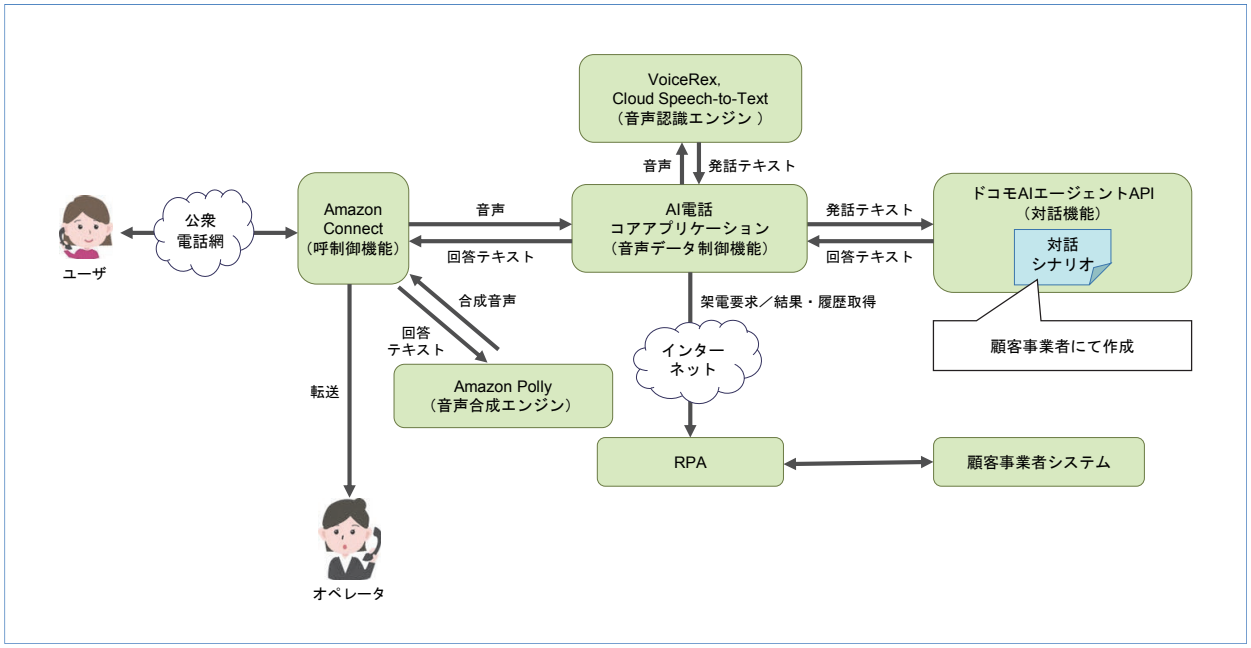


図2　AI電話サービスのシステム構成

*6　API：ソフトウェアの機能を他のプログラムから利用できるように切り出したインタフェース．

*7　音声合成：テキストから人工的に音声データを作り出し，テキストを読上げできるようにする技術．

*8　音声認識エンジン：音声データを入力し，発話内容をテキスト化する装置．

検出すると，AI電話コアアプリケーションがドコモAIエージェントAPIやRPAと連携してその後の処理を実行する．

音声認識エンジンには主にNTTの「VoiceRex[*9]」とGoogleのCloud Speech-to-Textを使用しており，ユーザ発話ターン単位で選択できる．エンジンごとに強みの領域が異なるため，聞き取る内容に応じて，あらかじめシナリオでエンジンを指定できるようにしている．例えば，電話口で本人確認が生じる場合，氏名の聞取りが必要となるが，VoiceRexでは音声認識結果を漢字と読み仮名の両方で出力でき，単語の分類が「姓」や「名」であるという情報も出力できるため，日本人の氏名を正確に認識するために有用である．

4. 多様なユースケースに対応するための音声認識技術に関する取組み

AI電話サービスでは，VoiceRexに，しゃべってコンシェルやmy daiz，音声認識IVRで実績のあるドコモ独自の言語モデル[*10]を適用している．その上で，VoiceRexを多様なユースケースで電話応対自動化に適用するために，次の3つのことに取り組んだ．

4.1 氏名の音声認識

サービス申込み／変更や予約の受付といったユースケースでは，本人確認のシナリオを完遂させることが重要であり，そのためには，氏名を正確に音声認識させる必要がある．そこで，氏名を学習データとして前述の言語モデルに追加するチューニングを実施した．個人情報の制約により，実在の氏名は利用できないため，架空の日本人の氏名を生成して学習データとした．チューニング前後の氏名認識性能を評価したところ，誤り率は，チューニング前の誤り率の7割以下まで減少した．

4.2 シナリオ特有の単語の音声認識

対話シナリオは顧客事業者ごとに異なり，シナリオ中で想定されるユーザ発話も対話シナリオごとに異なる．あるシナリオで特有の頻出単語が，一般的なAIエージェントとの対話ではまれな場合，誤認識することも珍しくない．例えば，飲食店予約のユースケースでは，ユーザが「個室」という単語を発話することが頻繁にある．しかし，前述の言語モデル内部では「個室」の出現頻度がそれほど高くないため，「保湿」「皇室」といった単語に誤認識されることがある．また，言語モデルに含まれていないサービス名などは原理上，認識できない．そのため，一般的には前述したように，事前に言語モデルをチューニングする必要がある．しかし，対話シナリオを追加するたびに言語モデルをチューニングして音声認識エンジンに実装することは，計算量の観点からも運用の観点からも現実的ではない．そこで，VoiceRexでは，音声認識リクエストごとに想定単語のリストを指定することで，言語モデルを変更することなく，指定された単語が出力されやすくなる機能を備えている．この機能を活用することで，音声認識性能を向上させることができた．

4.3 発話内容に応じた適切なタイミングの応答

AIエージェントによる音声対話では，応答の早さ，すなわち，ユーザ発話が終端してから短時間でAIによる発話を再生することが，高いユーザ体験に繋がる一要素である．一方で，住所や連続番号，自由回答など，ユーザがポーズ（呼吸や間）をおいて発話する場合，そのポーズを発話の終端と見なしてしまうと，AIがポーズより後の発話を聞き取れなかったり，ユーザ発話を遮って応答し始めてしまったりすることになる．つまり，ポーズを含む発話の聞取りの成功率と応答の早さがトレードオフの関係となる．

＊9 VoiceRex：NTTメディアインテリジェンス研究所が開発した音声認識エンジン．

＊10 言語モデル：単語の並び方の頻度を表現したモデル．

そこでAI電話サービスでは，シナリオ内のユーザの発話ターンごとに，想定される発話内容に応じて許容ポーズ長を設定し，AI電話コアアプリケーションが音声認識エンジンに対して動的に指定している．例えば，「ご氏名はドコモ太郎様でよろしいでしょうか」というAIの発話後は，「はい」または「いいえ」といった短いユーザ発話が想定されるため，許容ポーズ長を数百ミリ秒と短く設定する．一方，「健康のために気を付けていることはありますか」というAIの発話後は，ユーザは考えながら長く発話することが想定されるため，許容ポーズ長を1秒以上と長く設定する．これにより，前述のトレードオフを解消し，短いユーザ発話にはテンポ良く応答しつつ，ポーズを含むユーザ発話も最後まで聞き取れるという効果が得られる．

5. 実証実験

ドコモは，AI電話サービスを商用提供する前に，試験用環境を構築し，2つのユースケースで実証実験を実施した．

5.1 本人確認を伴う申込み受付

月額サービスを提供する事業者における，電話応対業務削減の効果を確認するために，受電での申込み受付のユースケースで実証実験を実施した．ここでの申込み受付は顧客事業者システム内のユーザデータベースと連携した本人確認を伴うため，**図3**に示すようなシナリオを設計，適用した．名義をユーザデータベースで検索するだけでユーザが一意に特定できれば，受付完了となる．名義確認，お客様番号確認，住所確認までではユーザが一意に特定できなくても，料金支払いに関する確認と合わせることで一意に特定できる場合は受付完了となる．本実証実験では，受付対話完結率77％という結果が得られた．なお，実証実験では音声入力のみで対話を進めたが，商用システムはダイヤルキー入力にも対応しており，ダイヤルキー入力と併用した場合の対

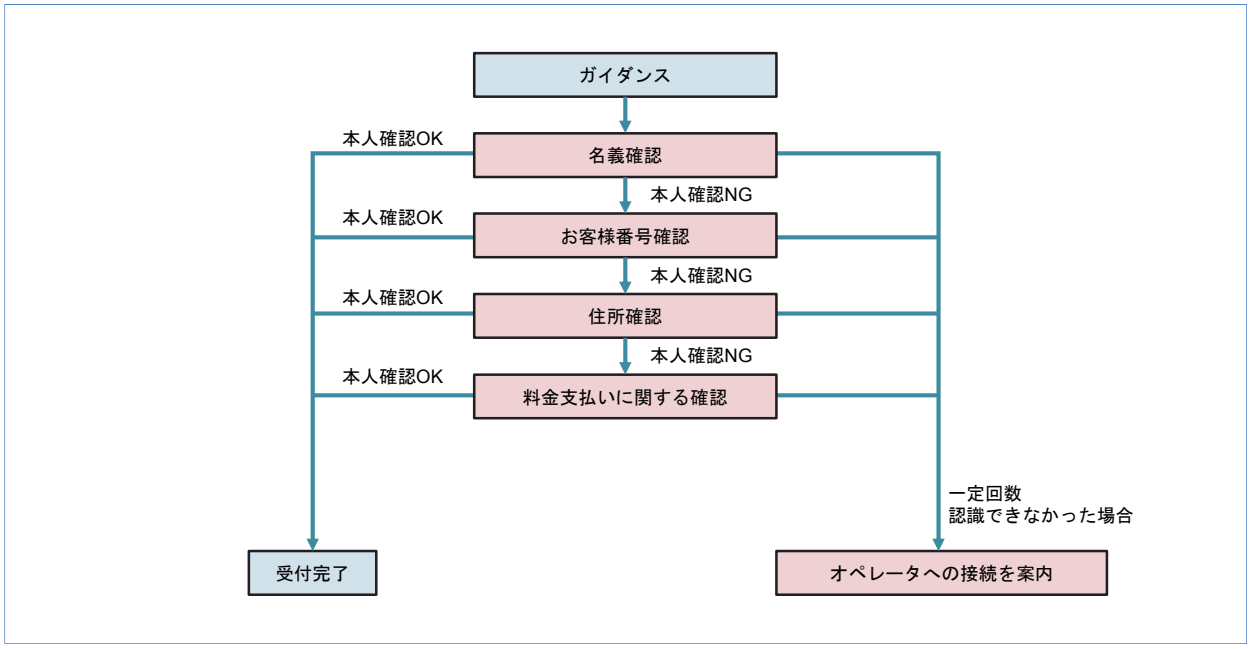


図3　本人確認を伴う申込み受付のシナリオイメージ

話完結率は88%となる見込みである．

5.2 高齢者見守り

独居高齢者は他者とのコミュニケーションが少なくなりやすく，毎日声をかけてもらうなどの支援を必要とするが，地域の支援機関も高齢者1人ひとりをケアするだけの人手が足りない，という問題が起きている．そこで，AI電話サポートが独居高齢者や支援機関の課題解決に繋がるかを検証するために，独居高齢者に定時に自動で架電し，体調や安否を確認する実証実験を実施した．電話では，AIから**表1**のような質問をし，高齢者と会話する．高齢者見守りの効果は定量的な測定が難しいため，対象者へのヒアリングを実施した．ヒアリング結果を基に，

表1　高齢者見守りでの質問

項　目	質　問
睡眠について	昨夜はよく眠れましたか.
	昨日は何時ごろに寝ましたか.
	寝ている途中に目が覚めることはありましたか.
	何かほかに睡眠で気になることはありますか.
食事について	昨日は三食食べられましたか.
	今週お肉やお魚，卵などのタンパク質は食べましたか.
	食欲はありますか.
	何かほかに食事で気になることがありますか.
活動について	昨日は外出されましたか.
	今日外出される予定はありますか.
	家族や友人とお話しされましたか.
	来週やってみたいと思っていることがありますか.
	どんなことをやってみたいですか.
体調について	今日の体調はどうですか.
	お通じはありますか.
	最近，病院は受診されましたか.
	健康のために気を付けていることはありますか.
	どんなことに気を付けていますか.
身の回りのことについて	昨日は入浴されましたか.
	湯船には浸かっていますか.
	歯のお手入れは毎日できていますか.
	何かほかに身の回りのことで気になることはありますか.

2021年2月から2次実証実験を実施している．

6. あとがき

本稿では，電話応対を自動化するAI電話サービスについて解説した．本サービスでは，電話応対業務に従事する人材不足の問題を軽減する効果が期待できる．2020年12月からAI電話サービスの商用提供が開始されており，申込みや予約の受付，高齢者の見守りといったユースケースで，実証実験の継続や正式なサービス開始が計画されている．今後は，音声認識性能のさらなる向上という技術的課題に継続的に取り組む．

文　献

[1] 橋本，ほか：“AIによるコールセンタお客様満足度向上とオペレータ業務効率―音声認識IVRの開発―，”本誌，Vol.25，No.4，pp.6–11，Jan. 2018.
[2] 大庭，ほか：“ドコモAIエージェント・オープンパートナーイニシアティブ，”本誌，Vol.26，No.3，pp.6–11，Nov. 2018.

Event Reports　5G　Open House　展示会レポート

docomo Open House 2021 ―ここから，みんなの，あたらしい社会がはじまる．Hello, Transformation.―

R＆D戦略部　玉置（たまおき） 真大（まさひろ）†

2021年2月4日から7日の4日間にわたり，オンライン上で「docomo Open House 2021 ―ここから、みんなの、あたらしい社会がはじまる。Hello, Transformation.―」を開催した．本稿では，本イベントの開催模様を紹介し，主だった展示の詳細について解説する．

1. まえがき

ドコモは，2021年2月4日から7日の4日間にわたり，オンライン上で「docomo Open House 2021 ―ここから、みんなの、あたらしい社会がはじまる。Hello, Transformation.―」を開催した．

本稿では，本イベントにおける主だった展示の詳細について解説する．

2. Webページ上でのオンライン展示会

昨今の社会情勢を踏まえ本イベントはオンライン上での開催とし，Tech Showcaseと題してドコモおよびパートナーの皆様より233展示を公開した．

展示に合わせてdocomo Open House TVと題した各種講演・セミナーも同時開催し，ドコモ幹部だけではなく社外の著名人の方々にもご参加いただき，今後の社会課題解決や新たな価値創造に関する講演など，第5世代移動通信システム（5G）やAIなどの技術に馴染みがない者にも楽しんでもらえるようバラエティに富んだ74コンテンツを公開した．4日間の会期の中で閲覧者総数は90,000人を超え，盛況であった（**図1**）．

Tech Showcaseでは，それぞれ個別にテンプレート化したWebページの中で技術の中身や，それを用いる事によるユーザメリットを紹介し，展示によってはWeb会議システムを通したライブデモの閲覧やチャットボット，閲覧者からのコメント記

† 現在，移動機開発部

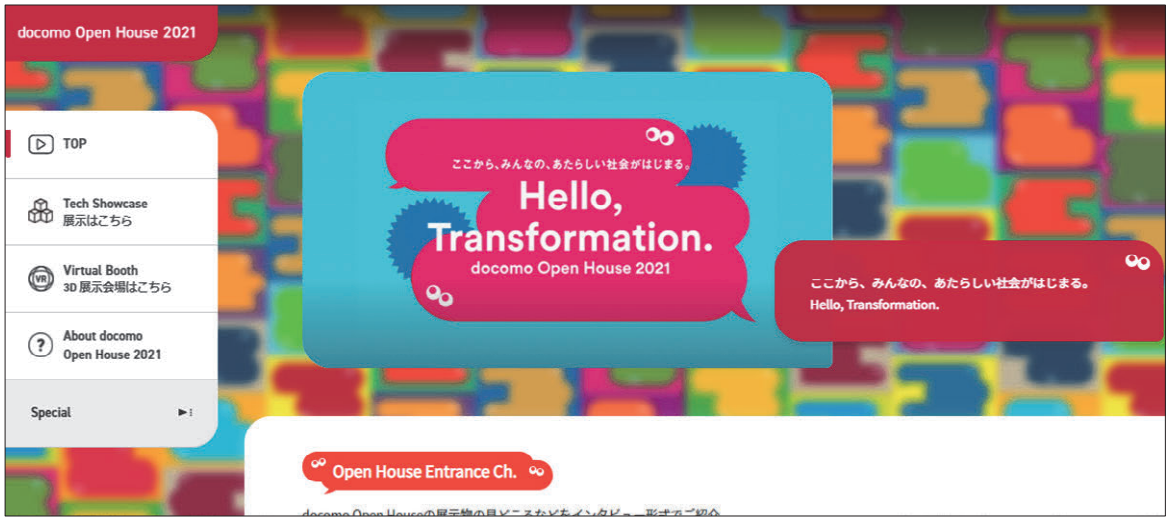


図1　docomo Open House 2021エントランスページ

載，登録情報を用いた名刺情報の交換などのインタラクティブな機能を提供した．また，通常のリアルな会場を用いた展示会と比較し，ディスプレイや展示台の設置などの制約に縛られる事無く，多くの動画によるコンテンツ訴求が可能であった（**図2**）．

講演・セミナーも同様に，会場の広さや時間の制約を受ける事なく提供が可能であり，同時に7つの動画を配信する事によりユーザがザッピング感覚で各種講演・セミナーを視聴可能とした．またすでに終了した講演・セミナーの配信についても，いわゆる「見逃し配信」としてユーザの閲覧時間に縛られない提供が可能であった．また，オンラインでの配信だからこそ可能な演出もあり，ウェルカムスピーチでは井伊 基之 代表取締役社長がVolumetric Capture[*1]による演出を行った（**図3**）．谷 直樹 常務執行役員R&Dイノベーション本部長は「デジタルがつくりだす新たな社会」と題した講演で，安全・あんしんに仕事や生活が送れる環境がニューノーマル[*2]となる社会の実現に向け，先進テクノロジやエコシステムの進化で貢献するドコモの取組みを紹介した．その他，ドコモの取組みをより楽しく知っていただけるよう，著名人による技術解説番組「なにこのテック?!」など，パートナーの枠を超えた多彩なテーマに沿った講演も開催した．

3. スマートフォンアプリケーション上でのVR展示会

前述のWebページ上での展示と同時に，Virtual Boothと題してドコモが独自開発したVirtual Event Platformを用いて，ドコモのもつ技術やソリューションを活用したコンテンツに没入体験するVR展

＊1　Volumetric Capture：カメラなどで撮影した映像を3次元のデジタルデータへ変換し，3D空間上に再現する技術．

＊2　ニューノーマル：社会の環境や情勢の変化に伴い，不可逆的に新たな常識が定着した状態．

図2　docomo Open House 2021 Tech Showcase

図3　ウェルカムスピーチ

示も実施した（**図4**）．

本Virtual BoothではVolumetric Video技術を用いて，著名人やバドミントン選手の映像に3Dのオブジェクトやモデルなどの演出を加えたリッチで臨場感のある限定コンテンツをスマートフォンアプリにより配信した（**図5**）．本技術は，専用の機材を用いてあらゆる角度から撮影した人や物の3Dモデルを VR空間にそのまま再現するだけでなく，被写

図4　Virtual Booth

図5　Virtual Booth内基本ブースの模様

体の動きも高精度にデジタル化できるもので，臨場感のあるVRコンテンツを360度自由な視点で視聴することを可能にした．

4. あとがき

本稿では，2021年2月4日から7日の4日間にわたり行われた，「docomo Open House 2021　―ここから、みんなの、あたらしい社会がはじまる。Hello, Transformation.―」の開催模様を紹介し，展示についての解説を行った．

ドコモでは，未来の新たな社会に向け，ユーザの生活スタイルやコミュニケーションを革新する，楽しさ，驚きのあるサービスを創り出していく．また，日本の成長と豊かな社会の実現を目指して，社会課題の解決に取り組みたい．

NEC＆D-Wave量子コンピュータチャレンジDays 最優秀賞を獲得

2020年12月8～10日に開催されたNEC，D-Wave Systems Inc. が主催する量子アニーリング人材を養成するためのオンラインイベント「NEC＆D-Wave量子コンピュータチャレンジDays」にて，クロステック開発部の片山 源太郎が最優秀賞を獲得しました．

本イベントにおいて，3日間のプログラムを通して量子コンピュータを使ったプログラミングを体験し，主催者から出された配送問題やスケジューリング問題の課題に取り組み，提出したモデルは，実務でモデルを使う上で，重み付けを扱いやすいように正規化するなどの配慮があった点が評価され，最高の評価点を獲得しました．

片山は今回の成績に関して以下のようにコメントしています．

「量子コンピュータを活用するのは初めての試みだったが，1日目のドリル問題を通して，理解を深めることができた．2日目のチャレンジ問題では，試行錯誤してさまざまな工夫を施した結果，それなりに良い解を出せたようで安心した．工夫としては，各量子ビットにかかる係数の範囲を一定にしたり，定式化を工夫して量子ビット数を減らしたりするなどを行った．」

ドコモでは，昨今の量子コンピュータ技術の進展をかんがみ，量子コンピュータの活用検討を行っています．量子コンピュータは，既存のコンピュータと異なるアーキテクチャで，これまで現実的な時間で解くことができなかったような問題を解くことが期待されています．NEC＆D-Wave量子コンピュータチャレンジDaysもその一環で，それ以外にも広告配信やサプライチェーンの最適化などにおける活用を検討する取組みを行っています．今後，NEC＆D-Wave量子コンピュータチャレンジDaysやその他の取組みを通して得た知見を活かして，新たなサービスを創出していきたいと考えています．

NTT DOCOMO
テクニカル・ジャーナル　Vol.29 No.1

2021年４月発行

企画編集　株式会社NTTドコモ　R&D戦略部
〒100-6150
東京都千代田区永田町 2-11-1
山王パークタワー39階

発　行　一般社団法人 電気通信協会
〒101-0003
東京都千代田区一ツ橋 2-1-1
如水会ビルディング6階

本誌掲載内容についてのご意見は
e-mail: dtj@nttdocomo.com宛