Vol.2018-IOT-40 No.17
2018/3/5

精緻に解析可能な恒常性のあるメール基盤の設計

松本 亮介[1,a)]　小田 知央[1]　笠原 義晃[2]　嶋吉 隆夫[2]　金子 晃介[3]　栗林 健太郎[1]　岡村 耕二[2]

概要：様々なコミュニケーションサービスが普及する中，今もなおメールは広く利用されている．メールサービスを提供する事業者では，大量のメールアカウントを管理するために，サーバに高集積にメールアカウントを収容することでリソース効率を高めている．一方で，特定のアカウントによるリソース占有により他のアカウントが影響を受け，サービスのサポート・運用コストの増大やメール遅延などの問題が後を絶たない．本論文では，高集積マルチアカウント型メールシステムを安定化するための，精緻に解析可能な恒常性のあるメール基盤の設計について述べる．

A Design of Elaborately Analyzable Homeostatic E-mail System

Ryosuke Matsumoto[1,a)]　Tomohisa Oda[1]　Yoshiaki Kasahara[2]　Takao Shimayoshi[2]
Kosuke Kaneko[3]　Kentaro Kuribayashi[1]　Koji Okamura[2]

1. はじめに

SNS やチャットサービスといったコミュニケーションサービスが普及する中，今もなおメールは広く利用されている．メールは，広く世界に普及したことと，送信側と受信側双方向での利用が前提となるため，両者の同意なくメールによるコミュニケーションを別のサービスに変更することは難しい．そのような背景の中，メールサービスには，spam メール対策 [14] や大量送受信の問題 [8] 等に対応しながらも，メールが遅延しないような安定性やセキュリティの担保が求めらている．

メールサービスを提供するホスティングサービスや ISP 事業者では，大量のメールアカウントを管理しつつハードウェアコストや運用管理コストを低減し，安価にサービスを提供するために，単一のサーバに高集積にメールアカウントを収容してリソース効率を高めている．高集積マルチアカウント型のメールシステムでは，アカウント単位のリソース使用量を効率化できるが，特定のアカウントの高負荷によって，同一サーバに収容している他のメールアカウントが影響を受けやすい課題がある．その結果，特定のアカウントの乗っ取りや大量送受信に伴うリソース占有がサーバの高負荷を引き起こし，サービスのサポート・運用コストの増大やメール遅延などの問題に繋がる．

これらの問題を解決するために，一般的には予めメールアカウント単位でメールの同時送受信数や時間単位での送信数制限等を静的に設定しておく手法 [9] がとられる．しかし，そのような手法を採用した場合，予め負荷を制限することはできるが，サーバに収容されている全てのメールアカウントがサーバリソースを適切に使いきるための制限値を決定することが難しい．また，一般的なメールサーバソフトウェアの制限手法は限定的で，現代の予測困難なメールの流量を状況に応じてリソースを制御することは困難である．

本論文では，高集積マルチアカウント型のメール送受信システムを安定化するために，精緻にセッション情報を解析でき，適応的にメールの流量制御やリソース制御を行えるメール基盤の設計について述べる．メールアドレスのドメイン単位でメールの流量を制限するために，メールに関するセッションを受信した時点で，リアクティブにドメイ

[1] GMO ペパボ株式会社 ペパボ研究所
Pepabo Research and Development Institute, GMO Pepabo, Inc., Tenjin, Chuo ku, Fukuoka 810-0001 Japan
[2] 九州大学 情報基盤研究開発センター
Research Institute for Information Technology, Kyushu University
[3] 九州大学 サイバーセキュリティセンター
Cybersecurity Center, Kyushu University
[a)] matsumotory@pepabo.com

ン単位でメールサーバプロセスを起動した Linux コンテナを起動させる．このアーキテクチャにより，予測の難しいメール流量やリソース使用量の変化に素早くシステムを変更できる．同時にセッション情報をセッション管理データベースに保存し，メールの送受信時にセッション情報を解析できるようにする．

メール送受信の流量を制御したい場合には，セッション情報の統計値を活用したり，システム管理者がセッションデータに基づいて判断したりすることにより，メールサーバソフトウェアの制限設定と Linux コンテナのリソース制御手法を組み合わせて実現する．また，Sendmail のメールフィルタープラグイン機能である Milter プロトコル [1] を活用して既存の SMTP サーバソフトウェアである Postfix[4] の制限手法を拡張したり，IMAP サーバソフトウェアの Dovecot[3] の拡張プラグインを実装することで，SMTP/IMAP コマンド単位やメールアドレス単位でメール流量やリソース使用量を精緻に制限できるようにする．

これらのシステム設計には，我々が研究開発している，実行環境の変化に素早く適応できる恒常性を持つシステムアーキテクチャの FastContainer[13] を応用する．メールの流量が増加した場合には，FastContainer によって，メールサーバプロセスが起動しているコンテナをオートスケーリングによるスケールアウト，またはスケールアップ型のリソース増加により対応する．実行環境の状態変化を素早く実現できるアーキテクチャを利用して，リソースを効率化するために，FastContainer の方式により，メールのセッションが生じていない場合はコンテナを自動的に停止させる．

本稿の構成を述べる．2 章では，従来の高集積マルチアカウント型のメールシステムの課題について整理する．3 章では，提案手法の設計と FastContainer アーキテクチャの概要ついて述べ，4 章で提案するメールシステムの実装の進捗について述べる．5 章でまとめとする．

2. 高集積マルチアカウント型のメールシステムの流量制御の課題

メールサービスを提供する企業や大学では，大量のメールアカウントを管理しつつも，ハードウェアコストやシステムの管理・運用コストを低減するために，高集積マルチアカウント型のメールシステムを採用している．ここで，高集積マルチアカウント型のメールシステムは，CPU の論理コア 24，メモリサイズ 32GB の昨今の一般的な PC サーバにおいて，メールのドメイン数を数万程度収容するシステムを想定している．

高集積マルチアカウント型のメールシステムでは，単一のサーバのハードウェアリソースを，複数のメールアカウントのドメインで共有できることはメリットにもなる．各ドメインのメール処理におけるリソース使用量に差異があったとしても，ハードウェアリソースが余っている場合においては，ドメイン間のリソース利用量の違いを許容できる．ハードウェアリソースを共有することによって複数ドメインでのリソース使用量を効率化し，各ドメインのリソース使用量の違いに対する運用・管理コストを低減できるというメリットがある．状況によっては，一般的にサービス利用者の専用のサーバや VM よりも複数ユーザの共有サーバの方がハードウェアスペックが高いため，価格以上の性能を得られることがある．

一方で，多数のドメインを高集積に収容することにより，ひとつのドメインがメールを大量に送受信したりすることによりリソースを占有し，他のドメインがメールを正常に送受信できなくなる課題がある．そういった問題によって，サービスプロバイダのサポートや運用・管理コストが増加し，コストを低減するための高集積マルチアカウント型のメールシステムが，かえってコストを増大させることにつながる．

そこで，高集積マルチアカウント型のメールシステムにおける採用のメリットとデメリットを比較しながら，流量制御の課題を整理する．

2.1 送信流量制御手法の課題

高集積マルチアカウント型のメールシステムでは，システムを介してメールを送信する場合に，メールサーバに接続している特定のクライアントの過剰な接続による影響を与えないようにするために様々なリソース制限手法 [9] がある．広く利用されている SMTP サーバソフトウェアである Postfix では，接続クライアント単位で以下の制限が可能である．

- 接続クライアント単位での同時接続数
- 接続クライアント単位での単位時間あたりの接続数
- 接続クライアント単位での単位時間あたりのメッセージ配送要求数
- 接続クライアント単位での単位時間あたりの送信可能な受信者アドレス数

上記のように，クライアント単位での柔軟な制限は可能であるが，メールアカウント単位やドメイン単位での制限はできないという課題がある．そのため，多数のクライアントから接続があった場合には，メールアカウントあるいはドメイン単位といった細かい粒度でのリソース使用量を適切に制限できない．

筆者が所属する GMO ペパボ株式会社では，一般的な SMTP サーバソフトウェアを利用した高集積マルチアカウント型のメールシステムにおいて，メールアドレスやドメイン単位での同時送信数を個別に制限できる機能を独自に実装している [5]．この制限によって，特定のメールアドレスが大量にメールを外部に送信したとしても，同時送信数制限内でしか送信できない．そのため，特定のメールアド

レスが大量送信によってリソースを占有することを防止できる．

一方で，同時送信数を静的に設定しており，全てのメールアドレスやドメインの制限が固定であるため，サーバリソースが余っていたとしてもそのリソースを適切に利用できない．また，サーバ収容設計時に事前にハードウェアやソフトウェアから適切に制限値を決定することは，メールの流量を予測できないことと相まって困難である．経験的に制限値を決定すると，ハードウェアリソースを必要以上に余らせてしまったり，高負荷状態になってから改めて制限値を決定するなど，流量の管理コストが増大しがちである．

特定のドメインのメール送信処理に対するリソース使用量が平常時よりも超過している状況で，平易にオートスケーリングのようなリソース追加を部分的に実施することも困難である．

2.2 受信流量制御手法の課題

メールシステムにおける受信処理を以下の 3 つの処理に分類する．

(1) 外部の MTA(Mail Transfer Agent) から管理対象の MTA がメールを受信する処理
(2) 管理対象の MTA にキューとして溜まったデータをメールサーバのメールボックスに配送する処理
(3) MUA(Mail User Agent) によってメールボックスからメールデータを受信する処理

(1) については，高集積マルチアカウント型メールシステムの場合，多数のドメインに対するメールを受信するため，2.1 節で言及したことと同様に，メールアドレスやドメイン単位で個別に受信数を制限できるが，静的な設定になり，制限値を経験的に決定することは困難である．また，外部からのメール受信は，送信する場合と比較して，メールシステムのサービス利用者が外部からメールが送られてくることを制御できないため，サービスとしてはできるだけメールを受信できるようにすべきである．そのため，送信処理の制限と比較して，メールシステムのゲートウェイに位置する MTA の受信処理は，制限が必要な時だけ制限するといったことが求められる．

一方で，外部からの spam メールを受信や，利用者の Gmail 向けへの転送設定によって spam メールが大量に転送されてしまうことにより，Gmail 側から拒否メールが大量に送り返されることもある．そのような現象によって，(2) の処理過程においてサーバに defer キューが大量に溜まり，システム管理者に通知が届くが，対応すべきこともないため，システム管理者の疲弊を招くことになる．本来メールキューの監視は，転送先 MTA から返信されたエラーメールが適切であるかどうかをシステム管理者が認識する場合に必要であり，明らかな spam メールに基づく defer は (1) の段階でキューに入れることなくエラーを返すべきである．

(3) においては，MUA から IMAP や POP3 によってメールを受信する際に，高集積マルチアカウント型メールシステムでは大量にアクセス集中が生じる場合がある．そのため，例えば IMAP/POP サーバソフトウェアとして広く使われている dovecot では，クライアント IP アドレスの同時接続数を制限する設定はあるが，メールアカウントやドメイン単位で個別に設定することはできない．また，特定のアカウントによる IMAP の検索や大量メールの操作処理によってリソースを占有することがある．この場合，特定のアカウント単位や，IMAP コマンド単位でのリソース使用量の制限をする必要があるが，そのような制限手法はない．

特定のドメインのメール受信処理に対するリソース使用量が平常時よりも超過している状況で，平易にオートスケーリングのようなリソース追加を部分的に実施することも困難である．サービス利用者によっては，追加料金を支払うことによって，サービスの快適性を求める状況もあり得るが，2 節で言及した高集積マルチアカウント型メールシステムのプロセスモデルのように，大量のアカウントを単一のサーバプロセスで管理している場合には，そういった要求に答えることが難しい．

3. 提案手法の設計

3.1 設計概要

2 節の課題から，高集積マルチアカウント型メールシステムにおいて，メール流量をドメイン単位で個別に制御可能であり，流量制御のきっかけとなる調査を精緻に収集したデータから精緻に解析可能な基盤が必要である．さらに，出来る限り高集積マルチアカウント型のリソース効率のメリットを両立する必要もある．

本研究では，高集積マルチアカウント型メール送受信システムを安定化するために，精緻にセッション情報を保存・解析でき，適応的にメールの流量制御やリソース制御を行うためのメール基盤の提案とその設計について述べる．メールアカウントのドメイン単位でメールの流量を制御するために，メールセッションを受信した時点で，ドメイン単位でリアクティブにメールサーバプロセスをコンテナ上に起動させる．同時にメールの送信，受信共にセッション情報をセッション管理データベースに保存し，メールの送受信時にセッション情報やその解析結果に基いて流量制御できるようにする．セッション情報の統計値を活用したり，サーバ管理者が判断したりすることにより，メールサーバプロセスの制限設定とコンテナのリソース制御手法を組み合わせて流量制御を実現する．

これらの設計には，我々が研究開発している，実行環境の変化に素早く適応できる恒常性を持つシステムアーキテ

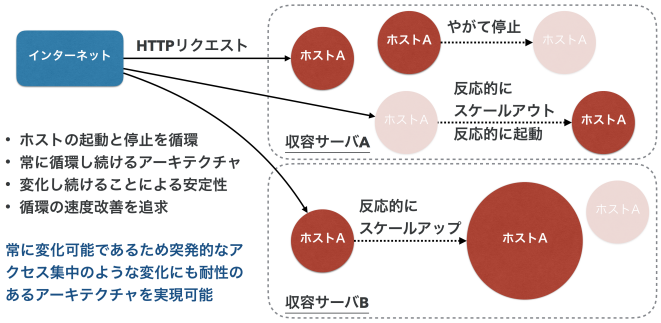

図 1　FastContainer アーキテクチャ概要図
Fig. 1　Concept Design of FastContainer Architecture

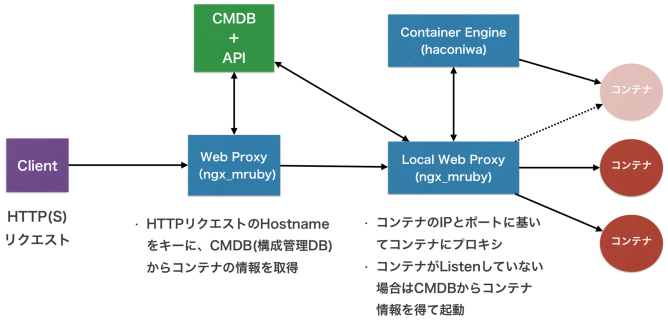

図 2　HTTP FastContainer のアーキテクチャ
Fig. 2　HTTP FastContainer Architecture

クチャの FastContainer アーキテクチャ [13] を応用する．メールの流量の増加を検知した場合には，FastContainer アーキテクチャによって，メールサーバプロセスが起動しているコンテナを自動的にスケールアウト，またはスケールアップ型のリソース増加を行う．

3.2　FastContainer アーキテクチャの概要

FastContaier アーキテクチャとは，実行環境の変化に素早く適応できる恒常性を持つシステムアーキテクチャである．従来の Web ホスティングサービスを利用できる程度の知識を持った個人が Web コンテンツを配信することを前提に，サービス利用者が負荷分散のシステム構築やライブラリの運用・管理を必要とせず，迅速にユーザ領域を複数のサーバに展開可能にするアーキテクチャである．

図 1 に示すように，Linux コンテナの起動を HTTP リクエスト単位でリアクティブに決定し，実行環境の変化に素早く適応できる恒常性を持つことを特徴としている．ここで述べる「リアクティブな決定」とは，HTTP リクエスト単位で状況に応じたコンテナの構成，例えば，起動，停止，起動数，リソース割り当て，起動継続時間 (ライフタイム)，といった状態を迅速に決定することを意味する．その特性から，コンテナの起動後は，リクエストを処理を行いながらライフタイム経過後に破棄され，プロセスがリクエストを処理していないアイドル状態のまま起動している時間を短縮することも可能であり，マルチテナント方式のリソース効率を高めている．

FastContainer アーキテクチャは，現時点では，**図 2** に示すように，HTTP リクエストに基いてリアクティブにコンテナを起動させることが前提になっている．本研究では，FastContainer を SMTP セッションでコンテナを立ち上げられるように拡張する．SMTP や IMAP も HTTP と同様に，メールアドレスのドメインに基いて通信対象のコンテナを区別することができる．

3.3　恒常性に基づくリソース効率化と精緻な流量制御の両立

FastContainer アーキテクチャによる恒常性により，メールセッションが生じない状況においてはプロセスを絶えず起動させておく必要はない．そのため，メールの処理が行われていないコンテナは一定期間で停止させることにより，リソース使用量を節約し，集積率を高める．

一方で，メールの送受信のゲートウェイ用途のコンテナと，ウィルスや spam メールのフィルタ用途のコンテナは大量のセッションが予想されるため，FastContainer の拡張として，常に 1 以上の数のコンテナが循環しながらも起動している状態を担保する ResidentContainer として動作させる．ResidentContainer は，FastContainer と違い，SMTP セッションにリアクティブにコンテナを起動せず，システムのスケジューラによって新しいコンテナの起動と振り分け先登録を行う．これは，全く起動していない状態があり得る FastContaier と違い，最低 1 つ以上のコンテナが起動していることが保証されるため，リアクティブ性がそれほど要求されないと判断したためである．

恒常性に基づくリソース効率化と精緻な流量制御と両立するために，ドメイン単位でメール送信コンテナ (SMTP) とメール受信コンテナ (IMAP) をそれぞれ起動させる．ドメイン単位でコンテナを起動することにより，Postfix のような SMTP サーバソフトウェアや Dovecot のような IMAP サーバソフトウェアの制御設定を特定のドメインに対して活用できるようになる．また，IMAP のコマンド単位での CPU 使用率の制限や，ドメインよりもより粒度の細かいメールアカウント単位での制御も，SMTP コンテナは Milter とセッションデータベースとの連携，IMAP コンテナはサーバソフトウェアを拡張してセッションデータベースと連携して行えるようにする．セッションデータベースについては 3.4 節で言及する．

さらに，コンテナは cgroup[10] の機能により，プロセスレベルでのリソース制御機構が組み込まれいてる．サーバソフトウェアのプロトコル前提の設定で対処が難しい場合は，コンテナのリソース制御により，特定のドメインの処

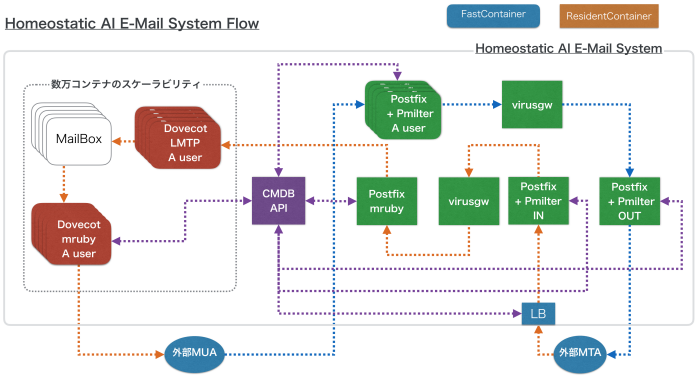

図 3　提案するメールシステムのフロー
Fig. 3　Proposed E-mail System Flow

理に使われる CPU 使用率やメモリ使用量の割り当てのみを部分的にリアルタイムで変更できる．

3.4　精緻にセッション情報を保存

各ゲートウェイやフィルタコンテナ，および，ドメイン単位での SMTP コンテナや IMAP コンテナのセッション処理時に，Milter やサーバソフトウェアの拡張機能により，セッション情報をデータベースに精緻に保存できるようにする．SMTP の場合は，Milter プロトコルを介してプロトコルのセッション処理に関わる情報を都度収集する．IMAP の場合は，Milter プロトコルのような連携プロトコルがないため，IMAP サーバソフトウェアに個別にセッション情報を収集できる拡張機能を実装する．これらの拡張機能はセッション情報の保存だけでなく，セッションデータに基いて流量制御を行える程度の柔軟性を持たせる．

収集したセッション情報は，送信，受信共にデータベース化することにより，送信と受信の関連性やセッション情報の遷移などを非同期に解析できるようにする．今後，ルールベースの解析から，統計処理によって分類を行うことにより，大量のセッション情報と流量に基づいて，大量送信や spam メールの振る舞い等を検知できないかについても検討していく．

4.　実装の進捗

4.1　提案するメールシステムの受信フロー

図 3 に提案するメールシステムの構成とメールのフローを示す．外部からメールを受信した場合の本メールシステムのフローとその実装について説明する．まず，外部 MTA から配送されたメールは，ロードバランサ (LB) で受けた後，SMTP のゲートウェイコンテナ (Postfix+Pmilter IN) に配送される．その際に，ゲートウェイコンテナは最低 1 つ以上のコンテナが起動している ResidentContainer であるため，3.3 節で述べたように，動的に配送先を決定することなく，システムのスケジューラによって決定されたコンテナにメールを配送する．また，FastContaier と ResidentContainer アーキテクチャ共に，近藤が中心となって我々が開発しているコンテナランタイムの Haconiwa[11] を利用することにより，SMTP や IMAP セッションに応じてリアクティブに状態を変化させられるようにする．

メールシステムにキューとして受信したくないメールについては，CMDB API と連携してセッション情報の解析結果を取得し，ゲートウェイの時点でエラーを返すようにする．セッション情報の保存と取得，及びエラーの判定は，SMTP の各セッションで Milter プロトコルを介して事前にフックされた Ruby スクリプトを実行できる Milter サーバソフトウェアの pmilter[*1] として実装した．この Ruby スクリプトファイルはセッション毎に都度実行されるため，SMTP コンテナ起動中であっても，再起動させることなく Ruby の実装を変更させることができる．pmilter により，ResidentContainer であっても即時流量制御の振る舞いや設定を反映させることができる．

次に，ゲートウェイコンテナからウィルスや spam メールフィルタコンテナ (virusgw) に配送する．フィルタコンテナを通過できるメールは，ドメイン単位で LMTP プロトコル [6] で起動する IMAP コンテナ群にプロキシするために，まず SMTP ルーティングコンテナ (Postfix mruby) に配送される．このルーティングコンテナは，次に配送されるべきドメイン単位での LMTP コンテナが数万個以上存在し，かつ，そのコンテナの構成情報は高頻度で更新される状況を想定して配置している．LMTP コンテナはセッションに応じて起動するため，同時に数万個以上起動するわけではない．

配送するメールアドレスのドメイン名に対応した LMTP コンテナを起動させるために，Postfix を mruby で拡張できる Postfix-mruby[*2] を利用し，SMTP ルーティングコンテナ上でドメイン名から CMDB API を通じて配送すべきコンテナ情報を動的に取得する Ruby スクリプトを実行する．その結果に基づいて，さらに配送すべき LMTP コンテナが収容されているホストサーバで，同様に起動しているローカル SMTP ルーティングコンテナに対してメールを配送する．ローカル SMTP ルーティングコンテナはドメインに対応する LMTP コンテナの構成情報を CMDB API から取得し，該当コンテナが起動されていなければ起動し，メールを配送してメールボックスに保存される．メールボックス自体は NFS 上で共有されており，複数の LMTP コンテナ (Dovecot LMTP A user) や IMAP コンテナ (Dovecot mruby A user) で共有できるようにファイルベースの dot lock を利用する．

外部 MUA からのメール参照時には，MUA が IMAP コンテナに IMAP セッションを接続した際に，CMDB API からドメインに対応するコンテナ構成情報を動的に取得

*1　https://github.com/matsumotory/pmilter
*2　https://github.com/tmtm/postfix-mruby

し，IMAP コンテナに接続する．実装には，筆者が開発している ngx_mruby[12][*3]に IMAP プロキシ機能を追加する方法を検討している．この実装によって，IMAP コンテナが起動していなければ，起動させてから IMAP セッション接続をしたり，起動済みであれば起動処理を省略して接続するといった処理が可能となる．

IMAP コマンド単位での制限や，セッションデータの保存と取得のために，Dovecot を mruby で拡張できるプラグイン，dovecot-mruby-plugin[*4]を開発した．dovecot-mruby-plugin は，IMAP コマンドの実行前後で Ruby スクリプトをフックして実行することが可能である．また，IMAP セッション情報を Ruby のメソッドで取得し，外部のデータベースに保存したり，データベースのデータに基いて Dovecot の振る舞いを拡張できる．

4.2 提案するメールシステムの送信フロー

メールを送信する際には，4.1 節で述べた IMAP プロキシと同様に，ngx_mruby に SMTP プロキシを実装し，ドメインに基いて動的にドメイン単位の SMTP コンテナに接続する．この場合も，FastContainer の特性により，SMTP プロキシによってコンテナが起動されていなければ起動させてから処理を行う．

ドメイン単位での SMTP コンテナでは，pmilter と連携して SMTP セッション情報を保存しておく．保存したセッション情報に基いてドメインやメールアカウント単位での同時送信数を制御することにより流量制限を行う．また，IMAP セッションデータと SMTP セッションデータを組み合わせつつ，統計的な解析を行うことにより，異常なセッション状態を検知できるようにする仕組みも検討中である．

SMTP コンテナでメールを送信処理した後は，受信のフローと同様にウィルスや spam メールのフィルタコンテナを介して，メール送信ゲートウェイコンテナ (Postfix+Pmilter OUT) に配送され，外部の MX サーバに送信される．

5. まとめ

本研究では，高集積マルチアカウント型のメール送受信システムを安定化するために，精緻にセッション情報を解析でき，適応的にメールの流量制御やリソース制御を行えるメール基盤の設計と本論文執筆時点での実装と検討項目の進捗を述べた．

今後，引き続き必要機能の実装を行いながら，メールの流量変化に想定通り耐えられるかの検証，リソース利用効率の定量的評価などを進めていく．また，保存したセッションデータを活用することにより，統計的手法や機械学習を使ってコンピュータが適応的に流量制御するための研究も行っていく．

本研究は，提案するメールシステムの必要なソフトウェアを実装し，NII クラウド上にシステムを構築して実証実験を行う予定である．

謝辞 本研究で使用したクラウド資源は，平成 29 年度国立情報学研究所クラウド利活用実証実験において提供された．また，本研究は，以下の研究助成を受けている．

松本亮介, 小田知央, 近藤宇智朗, 笠原義晃, 岡村耕二, 嶋吉隆夫, 金子晃介, mruby を利用した軽量コンテナクラウド基盤の研究開発を介した mruby の大規模・高負荷テスト, 2017 年度 Ruby Association 開発助成, 2017 年 10 月．

[*3] https://github.com/matsumotory/ngx_mruby
[*4] https://github.com/matsumotory/dovecot-mruby-plugin

参考文献

[1] Costales, Bryan and Flynt, Marcia, Sendmail milters: a guide for fighting spam, Addison-Wesley. 2005.
[2] Darya Gudkova, Maria Vergelis, Nadezhda Demidova, Tatyana Shcherbakova, Spam and phishing in 2016, https://securelist.com/kaspersky-security-bulletin-spam-and-phishing-in-2016/77483/.
[3] Dovecot: Secure IMAP server, https://www.dovecot.org/.
[4] Dent, K. D, Postfix: The Definitive Guide: A Secure and Easy-to-Use MTA for UNIX, O'Reilly Media, Inc., 2003.
[5] Hiroya Ito, Yapc Asia 2009 ペパボでの Perl の使い方, https://www.slideshare.net/hiboma/yapc-asia-2009-perl.
[6] John Myers, RFC2033: Local mail transfer protocol, The Internet Society, Oct 1996.
[7] Nazario, José, Using xinetd, Linux Journal, Vol.2001, No.83es, pp8, 2001.
[8] Yoshiharu Tsuzaki, Ryosuke Matsumoto, Daisuke Kotani, Shuichi Miyazaki, Yasuo Okabe, A Mail Transfer System Selectively Restricting a Huge Amoount of E-mails, Workshop on Resilient Internet based Systems (REIS 2013), pp. 896-900, Dec. 2013.
[9] Postfix Performance Tuning, http://www.postfix.org/TUNING_README.html.
[10] Rosen R, Resource Management: Linux Kernel Namespaces and cgroups, Haifux, May 2013.
[11] 近藤宇智朗，松本亮介，栗林健太郎, Haconiwa: プログラムによる，組み立て可能性と拡張性を持つ Linux コンテナ, 情報処理学会第 80 回全国大会, 2D-04, 2018 年 3 月.
[12] 松本亮介, 岡部 寿男, mod_mruby: スクリプト言語で高速かつ省メモリに拡張可能な Web サーバの機能拡張支援機構, 情報処理学会論文誌，Vol.55, No.11, pp.2451-2460, Nov 2014.
[13] 松本亮介, 近藤宇智朗, 三宅悠介, 力武健次, 栗林健太郎, FastContainer: 実行環境の変化に素早く適応できる恒常性を持つシステムアーキテクチャ, インターネットと運用技術シンポジウム 2017 論文集，No.2017，pp.89-97, 2017 年 12 月.
[14] 松井一乃, 金高一, 加来麻友美, 池部実, 吉田和幸, milter の組合せによる低配送遅延を目指した spam 対策メールサーバの設計と導入の効果について, 情報処理学会論文誌, Vol.55, No.12, pp.2498-2510, 2014 年 12 月.
[15] 山井成良, 岡山聖彦, 中村素典, 清家巧, 漣一平, 河野圭太, 宮下卓也, SMTP セッションの強制切断による spam メール対策, 情報処理学会論文誌, Vol.50, No.3. pp.940-949, 2009 年 3 月.