# 軽量コンテナに基づく柔軟なホスティング・クラウド基盤の研究開発と大規模・高負荷テスト環境の構築

笠原義晃[†] 松本亮介[‡] 近藤宇智朗[‡] 小田知央[‡]
嶋吉隆夫[†] 金子晃介[†] 岡村耕二[†]

[†]九州大学 [‡]ペパボ研究所

九州大学

GMOペパボ株式会社

# 本日の内容

- FastContainerアーキテクチャについて
- テスト環境の構築
- FastContainer研究開発における今後の課題
- まとめ

# ペパボ研究所×九大情報基盤研究開発センター

九州大学　GMOペパボ株式会社

本状は GMO ペパボ株式会社と九州大学 情報基盤研究開発センターの共同プレスリリースです。
各組織より重複して配信される場合がございますが、あらかじめご了承ください。

2017 年 10 月 20 日

報道関係者各位

GMO ペパボ株式会社
九州大学 情報基盤研究開発センター

**ペパボ研究所 × 九州大学 情報基盤研究開発センター**
**コンテナ型仮想化技術によるクラウドホスティングの共同研究開発を開始**
**～より頑健で柔軟なホスティングサービスの実現を目指す～**

GMO インターネットグループの GMO ペパボ株式会社（代表取締役社長：佐藤 健太郎　以下、GMO ペパボ）で、インターネットに関する新技術の創造と実践に取り組む研究開発組織「ペパボ研究所」と、国立大学法人九州大学 情報基盤研究開発センター（センター長：谷口 倫一郎　以下、九州大学 情報基盤研究開発センター）は、2017 年 10 月 1 日より、コンテナ型仮想化技術 (※1) を基盤に用いたクラウドホスティングに関する共同研究開発を開始しました。

両者は、大規模インフラ上にコンテナ型仮想化技術を用いたクラウドホスティング環境を構築し、高負荷下での実証実験等を行うことで、より頑健で柔軟なクラウドホスティング (※2) の実現に向けて取り組んでまいります。

（※1）ユーザーごとに独立したアプリケーション実行環境（＝コンテナ）を、1 つの OS 上に複数構築することで、より少ないコンピューターリソースで仮想的な動作環境を実現する技術。

（※2）システムへの要求に応じてリソース量が自動的に最適化されるクラウドホスティング。

▲共同研究者（左から）小田、近藤、松本、笠原、岡村、嶋吉、金子

# FastContainerで解決したい課題

- コンテナ型仮想化による高集積ホスティングで
- 一般的なウェブアプリ（WordPress とか PHPとか）をそのまま普通に使ってもらいつつ
- 難しい事は利用者に意識させず基盤でやる
  - オートスケーリング
  - コンテナ内のライブラリ更新や脆弱性対応
  - 収容サーバのメンテナンスのためのコンテナ移動など

- 商用サービスで既にベータ運用中
- 実運用を通してさらに解決すべき課題

# 本研究の目的

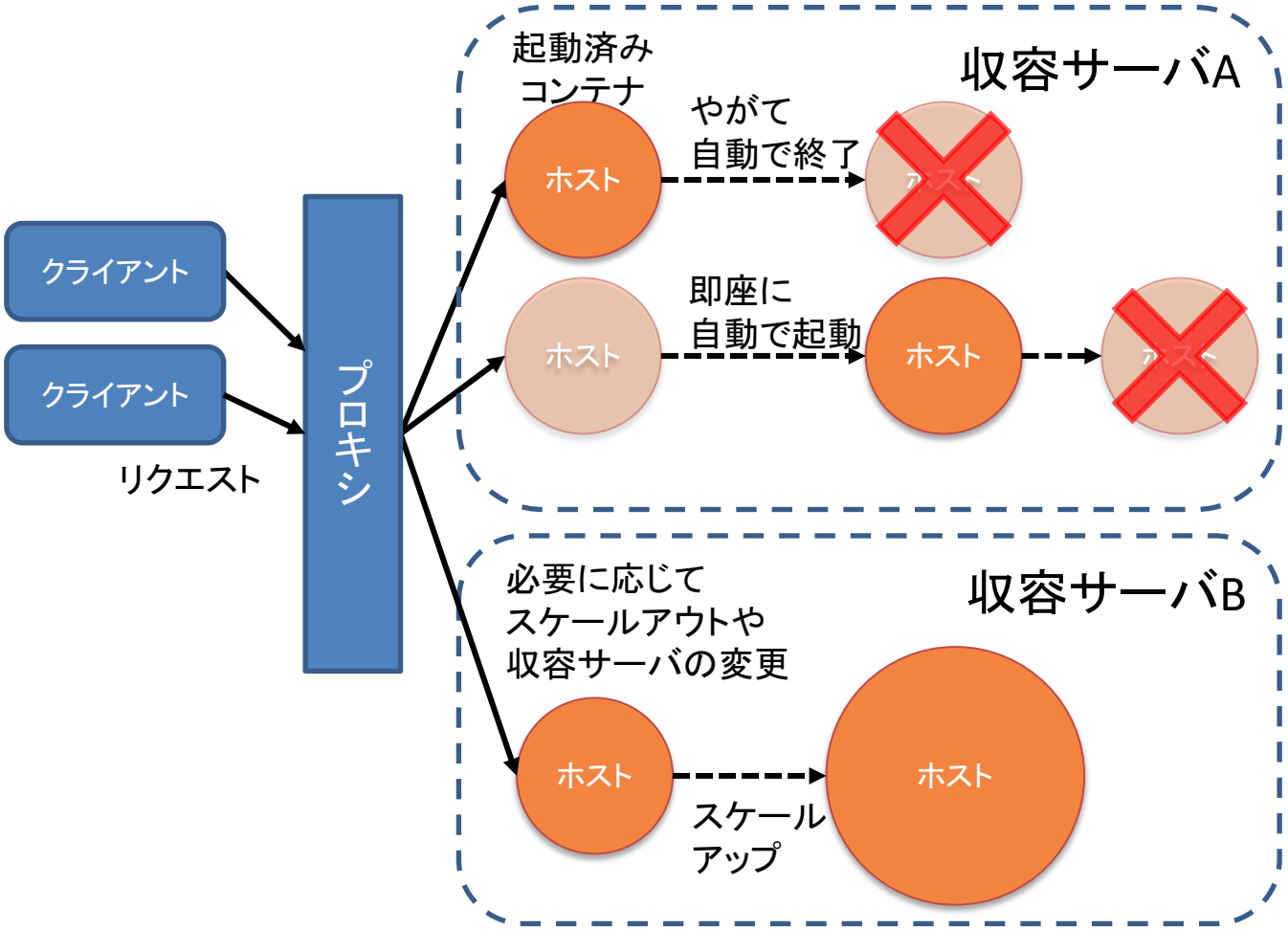
- FastContainerアーキテクチャの評価・改良のためのテスト環境の構築と活用
  - 実運用環境では試しにくいような試験を実施できるテスト環境を実運用環境とは別に構築
  - FastContainerをさらに改良

- 本発表ではFastContainerで解決すべき今後の課題について述べる

# FastContainerアーキテクチャの特徴

- 軽量コンテナに基づくホスティング・クラウド基盤
  - ペパボ研究所の松本らが開発・実装
- 各ユーザの環境は個別のコンテナで隔離
  - 独自実装の軽量コンテナ基盤を利用
- コンテンツ等の永続的な情報はコンテナ自体とは分離
  - データベースや共有ストレージに保存
- コンテナの起動や終了が高速に可能
  - コンテナは外部からのリクエストの粒度で起動され、一定時間で自動停止する
- コンテナ数の増減や割当リソースの変更も随時可能

# FastContainerの概念図



# コンテナの循環

- 一般的なサービス基盤はサービスを止めないのが前提（永続性）
- FastContainerではコンテナを使い捨てられる
  - 不要なコンテナは停止しリソースを解放する
  - スケールアウトやスケールアップが柔軟
  - ライブラリやミドルウェアは自然に更新される
  - システムの安定性も向上すると期待できる
    - （メモリリークなどの解消）
  - コンテナ収容ホスト間の移動も容易
    - メンテナンスコストの低減
- 外部から見るとサービスは継続して見える

# Serverlessアーキテクチャとの違い

- Serverlessアーキテクチャ
  - アプリケーションの運用は基盤側では行わない
  - Amazon Lambda等は利用者によるコーディングが前提で専門家向け

- FastContainerアーキテクチャ
  - 汎用的なウェブアプリケーションを基盤側で提供
  - 専門的な知識なしに利用できることを指向

# テスト環境

- FastContainerを利用したウェブホスティング環境をパブリッククラウド資源上に再構築
  - 実運用システムでは実ユーザへの影響があるような高負荷試験等のテストを実施するため

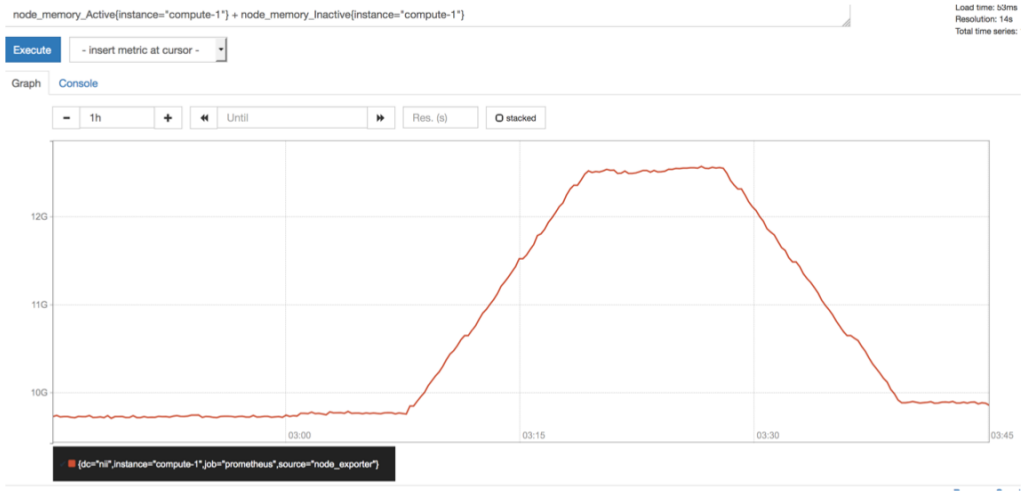
- 平成29年度国立情報学研究所クラウド利活用実証実験において提供されたクラウド資源を利用
  - 仮想マシン30台程度の構成

# 構築の自動化

- 自動環境構築ツールを積極的に利用
  - NIIの実証実験終了後も再利用可能とするため
  - プライベートなFastContainer環境構築への応用
- Terraform: 仮想環境全体の構築
  - 特定のクラウド環境に依存せず利用可能
- Chef: インスタンス毎の環境構築
  - 小規模～中規模の環境に向いている
  - 将来の大規模化にも耐えられる
  - 周辺のエコシステムが整っている

# 現状

- サービス基盤部分はほぼ構築完了
  - 商用サービスにはあるGUIは割愛
  - 共有ストレージも一部割愛
- スクリプトでプロジェクト自動作成可能
  - 残りの期間で負荷試験等実施



# FastContainer研究開発における今後の課題

- リソース効率化の実証実験
- サービスレベルの適切な導出方法
- コンテナ起動時間の影響低減
- 要スケーリング状況の迅速な検知と誤検知対策

# リソース効率化の実証実験

- コンテナの停止・循環によるリソース利用効率の改善を定量的に評価
- 本テスト環境を利用し、1000～1万コンテナ程度の規模で実験を予定
  - コンテナを永続的に稼働した場合
  - コンテナを自動停止し循環させた場合

# サービスレベルの適切な導出

- 利用するミドルウェアの種類によってはコンテナ起動に数秒程度かかってしまう
  - コンテナの初回起動に当たったサービス利用者はその分応答を待たされる
- → サービスレベル（ユーザ体験）の低下
- 初回起動に当たる確率は、コンテナの起動速度、持続時間、アクセス頻度によって決まる

- サービスレベルとの関係を定量的に導出できるようにしたい

# コンテナ起動時間の影響低減

- コンテナの起動時間は短いほどよい
- 短かくできない場合でも影響を低減したい
- 対策方法の例
  - 最低1つのコンテナを常時起動
  - コンテナプロセスのダンプ・リストアによる高速化
  - 次回起動コンテナの事前起動による切り替え
  - アクセス予測による投機起動
- いずれもリソース効率の低下を伴うがサービスレベルの向上に寄与する
  - 追加料金で使えるオプションみたいな?

# 要スケーリング状態の検知と誤検知対応

- 現在は試験的な実装
  - 15秒単位でアクセス状況等を収集
  - 3分毎に集計して要スケーリング状態の検知
- 即時性を上げるため集計時間を短縮すると誤検知による無駄なスケーリングが増加？
- FastContainerはコンテナの起動・停止が高速なのだから誤検知を許容する、という考えも

# まとめ

- FastContainerの概要
- テスト環境
- FastContainerの今後の課題

- テスト環境での基盤構築はほぼ完了したが実験がまだまだ
  - 残り期間は短いが活用したい

# 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ました
- 本研究は，以下の研究助成を受けています
  - 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.