mruby を用いた Linux ロードバランサインタフェースの実装

卒業研究最終発表

中野研究室

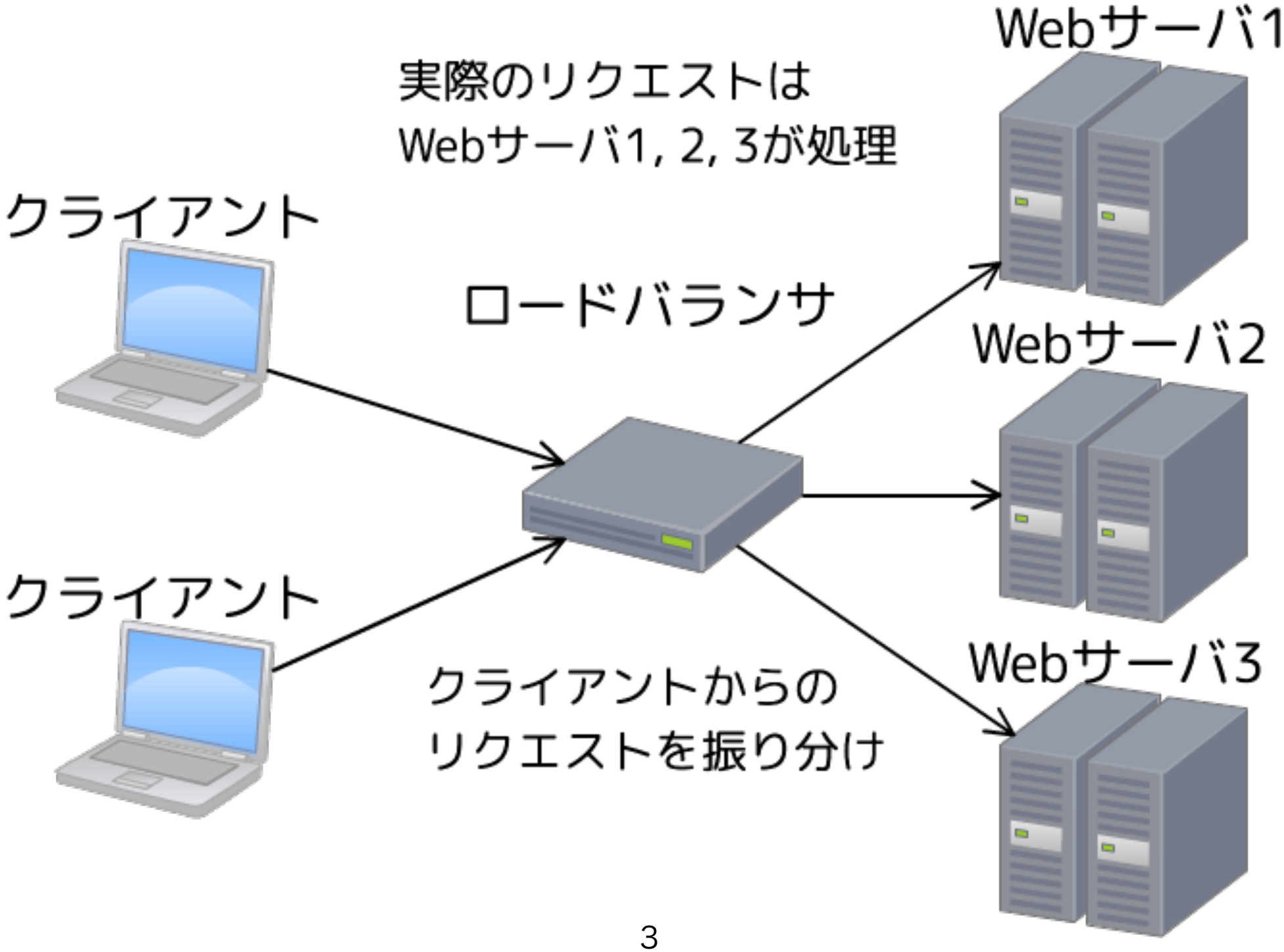
吉川竜太

2014/2/6

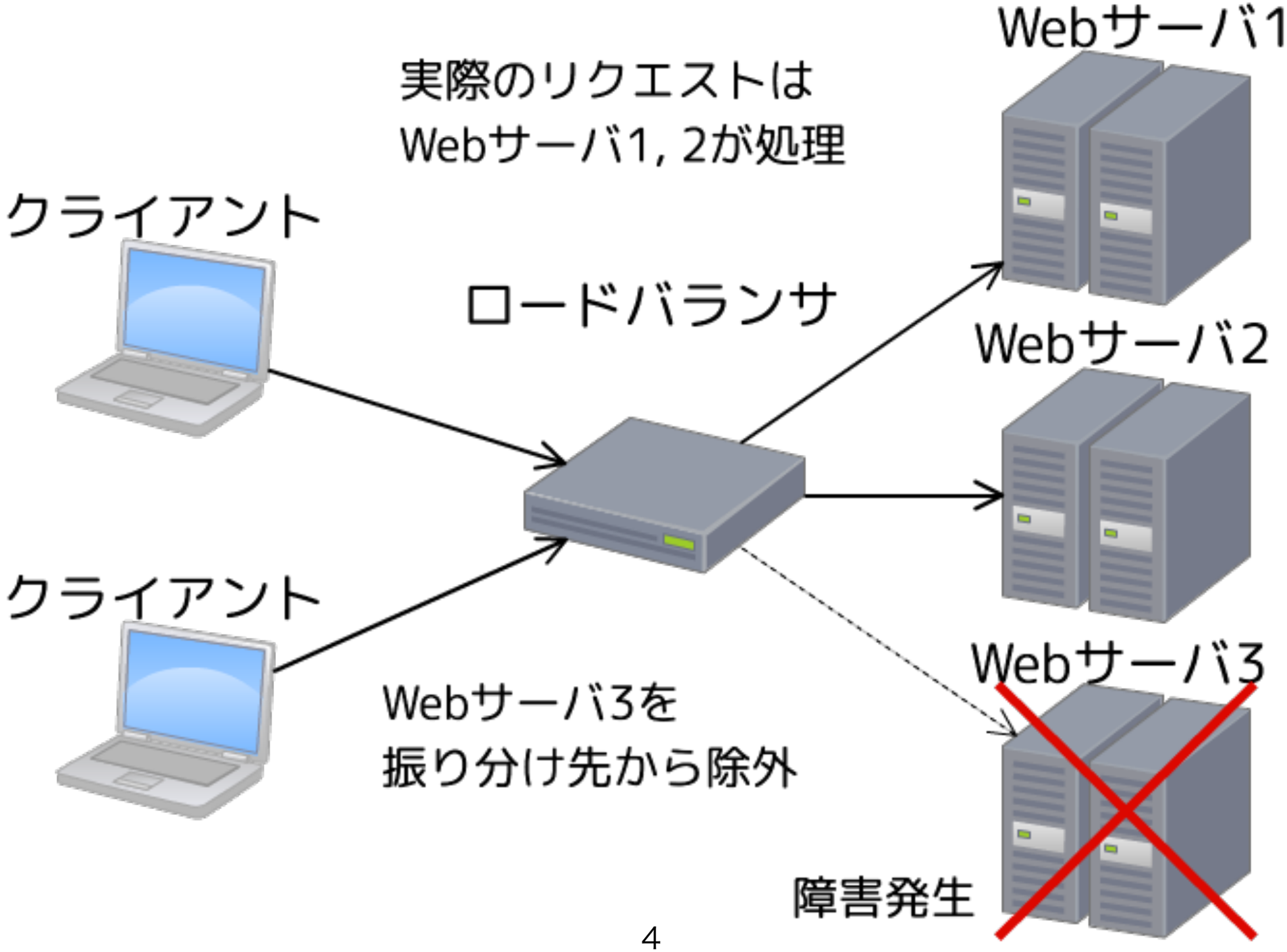
背景 - ロードバランサ

- サーバの負荷を分散させるシステム
  - クライアントからのリクエストを複数のサーバに振り分ける
  - サービスの可用性・拡張性の需要の高まりに伴い普及
- 振り分け先のサーバを定期的に確認し，異常があれば振り分けをやめる (ヘルスチェック)

ロードバランサによる振り分け



ヘルスチェック


実際のリクエストは
Webサーバ1, 2が処理
クライアント
ロードバランサ
Webサーバ1
Webサーバ2
クライアント
Webサーバ3
Webサーバ3を
振り分け先から除外
障害発生

IP Virtual Server (IPVS)

- Linux のロードバランサ
  - Linux カーネル 2.6 以降に組み込まれている
- 振り分けの設定として，次のような方法がある
  - C 言語のライブラリである libipvs を用いる方法
  - 設定用のソフトウェアである Keepalived を用いる方法
- Keepalived を用いる方法が一般的である

IPVS 設定用のインタフェース

- libipvs
  - C 言語を用いるため記述力は高い
  - メモリ管理や文字列操作が煩雑
  - ヘルスチェックの機能を記述するのが煩雑
- Keepalived
  - 設定ファイルで簡潔に記述することが可能
    - 複雑な操作は出来ず，記述力が低い
  - ヘルスチェックの機能が限定的である

インタフェースごとの比較

	記述力	ヘルスチェック	簡潔性
libipvs	○	△	×
Keepalived	×	△	○

目的と方針

- 目的
  - IPVS の記述力・簡潔性の高いインタフェースの実装
- 方針
  - 組込み向けで軽量な Ruby である mruby を採用
  - libipvs のラッパーを mruby に実装する
  - 実装したラッパーを使い，より簡単な構文で書けるモジュールを mruby を用いて実装

ラッパーの実装

- ロードバランサ・振り分け先をインスタンスとして生成
- libipvs の関数名を mruby のメソッドとして提供
  - libipvs の主な構造体の関数がほとんど利用可能

```
lb  = IPVS::Service.new({
        'addr' => '10.0.0.1',
        'port' => 80})
web1 = IPVS::Dest.new({
        'addr' => '192.168.0.1',
        'port' => 80})
lb.add_service
lb.add_dest(web1)
```

モジュールの実装

- mruby の記述力の高さを利用し，設定ファイル風の構文で書けるモジュールを実装
  - 設定ファイルの構文中に mruby のコードを挿入可能
  - 外部ライブラリを用いることで自由なヘルスチェックが可能
- 例として，Keepalived 風の構文を実装し，比較する

Keepalived の記述例

```
virtual_server 192.168.0.1 80 {
  real_server 192.168.0.2 80 {
    HTTP_GET {
      url {
        path /
      }
    }
  }
  real_server 192.168.0.3 80 {
  }
}
```

HTTPのGETを用いたヘルスチェック

192.168.0.1の80番ポートに来た接続を
192.168.0.2, 192.168.0.3の80番ポートへ
振り分ける設定

Keepalived の問題点 (1)

```
virtual_server 192.168.0.1 80 {
  real_server 192.168.0.2 80 {
    HTTP_GET {
      url {
        path /
      }
    }
  }
  real_server 192.168.0.3 80 {
  }
}
```

ヘルスチェックの方法が限定的

動的なサーバ追加等は出来ない

Keepalivedで定められたヘルスチェックしか出来ない
状況に応じた動的なサーバ追加などは出来ない

Keepalived の問題点 (2)

```
virtual_server 192.168.0.1 80 {
  real_server 192.168.0.2 80 {
    HTTP_GET {
      url {
        path /
      }
    }
  }
  real_server 192.168.0.3 80 {
  }
}
```

ブレースが足りないがエラーは出ない

意図しない設定で動作してしまう

上記設定をそのまま記述すると、
192.168.0.1を192.168.0.2に
振り分ける設定と解釈し動作

本実装による記述例 (1)

```
virtual_server("192.168.0.1:80") {
  web1 = real_server ("192.168.0.2:80") {
    healthcheck {|lvs, dests|
      if Curl::get(web1.addr).body == nil
        lvs.del_dest(web1)
      end
    }
  }
  real_server("192.168.0.3:80") {
  }
}
```

192.168.0.1の80番ポートに来た接続を
192.168.0.2, 192.168.0.3の80番ポートへ
振り分ける設定

本実装による記述例 (2)

```
virtual_server("192.168.0
  web1 = real_server("19
    healthcheck {|lvs, dests|
      if Curl::get(web1.addr).body == nil
        lvs.del_dest(web1)
      end
      if dests.length < 2
        real_server("192.168.0.4:80"){}
      end
    }
  }
  real_server("192.168.0
  }
```

外部ライブラリを使った
HTTPのGETによるヘルスチェック

振り分け先が2つ未満
になったらサーバ追加

mruby のコードが
そのまま書けるため
自由なヘルスチェック定義・
動的なサーバ追加などが可能

mruby が構文チェックを行うため、
ブレースが欠けてしまった場合は実行されない

まとめ

- Linuxのロードバランサである IPVS を
  mruby を用いて操作できるインタフェースを実装
- 記述力の高い mruby でインタフェースを実装することに
  より様々な構文を定義することが可能
  - 例としてKeepalived風の構文を実装し，問題点を解決
- 近日公開予定

関連研究

- ipvsadm
  - Linux のコマンドとして提供される IPVS のインタフェース
- ldirectord
  - Keepalived と同様，設定ファイルを用いて IPVS の操作を行うソフトウェア
- mod_mruby [松本ら '12]
  - mruby を用いて Web サーバ Apache の機能拡張を行える機構を実装

なぜmrubyを使うのか

- mruby
  - 組込み開発でも利用できる軽量 Ruby
    - 既存の C/C++ アプリケーションと共存可能
    - ロードバランサのようなリアルタイム処理に向いている
- サーバエンジニアの業界で Ruby が流行
  - Chef/Vagrant などの自動化ソフトウェアが活躍