

Event-driven Monitoring System

イベント駆動型監視システムが変える、 アプリ向け監視オペレーションの未来

開発担当者が語る、ここだけの話

an architecture for realtime event-driven monitoring system

Distribution Version

@zembutsu 前佛雅人

at SHIBUYA MARKCITY (IBM)

Apr 5, 2012

ここからは

ゆるふわな

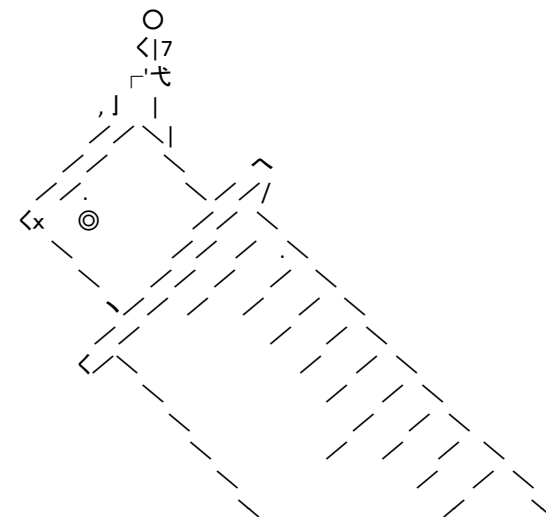
セッションです♪

今日の内容

もう何も(障害も)恐くない!!

- 割と新しい監視の取り組み。
“みんなと共有できたら、
とってもうれしいなっ…て。”

- イベント駆動型とは？
- 従来の監視との違いは何？
- 時代に対応した監視サービスのあり方とは？



CEP (Complex Event Processing) Setsuna と Muninwalk のデモを
交えながら、CEP と連携する開発中の監視システムのコンセプトを、
皆様と共有できればと考えています。

…の前に。 自己紹介

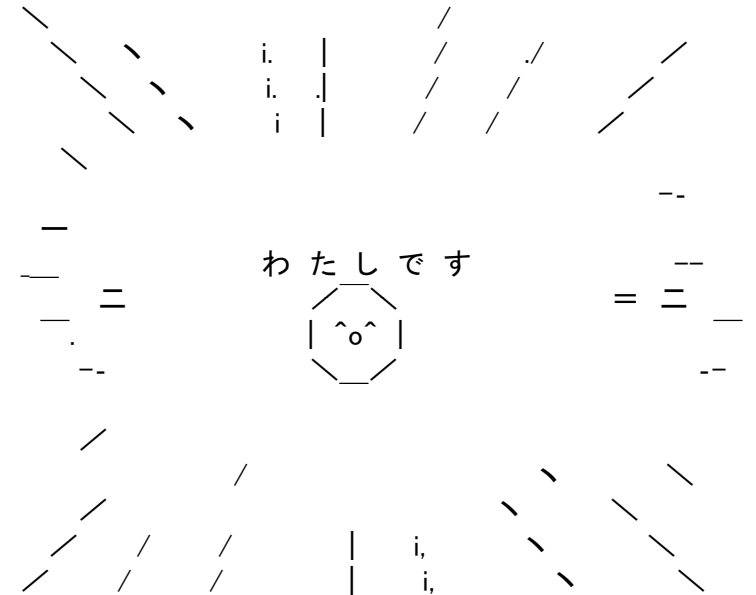


@zembutsu

<http://twitter.com/zembutsu/>

<http://facebook.com/zembutsu/>

<http://pocketstudio.jp/>



魂を重力(物理インフラ層)にひかれた、
インフラエンジニア的な何かが私です。

- ・クラウド系のコミュニティ
- ・Cloudcomputing.jp 編集見習い

…の前に。 自己紹介



@zembutsu

<http://twitter.com/zembutsu/>

<http://facebook.com/zembutsu/>

<http://pocketstudio.jp/>



株式会社リンク アプリプラットフォーム部
エンジニアをやっております。

運用・開発・設計・企画・営業・などなど

社内ネットワーク運用・ファイルサーバ管理・ISMS…

…の前に。 自己紹介



@zembutsu

<http://twitter.com/zembutsu/>

<http://facebook.com/zembutsu/>

<http://pocketstudio.jp/>

at+link

こまけえことはいらんだよ！！



鯖を捌く Ops的、何か



—Don't forget. always, somewhere,
someone is fighting for you.
—As long as you remember her.
you are not alone.

Operation

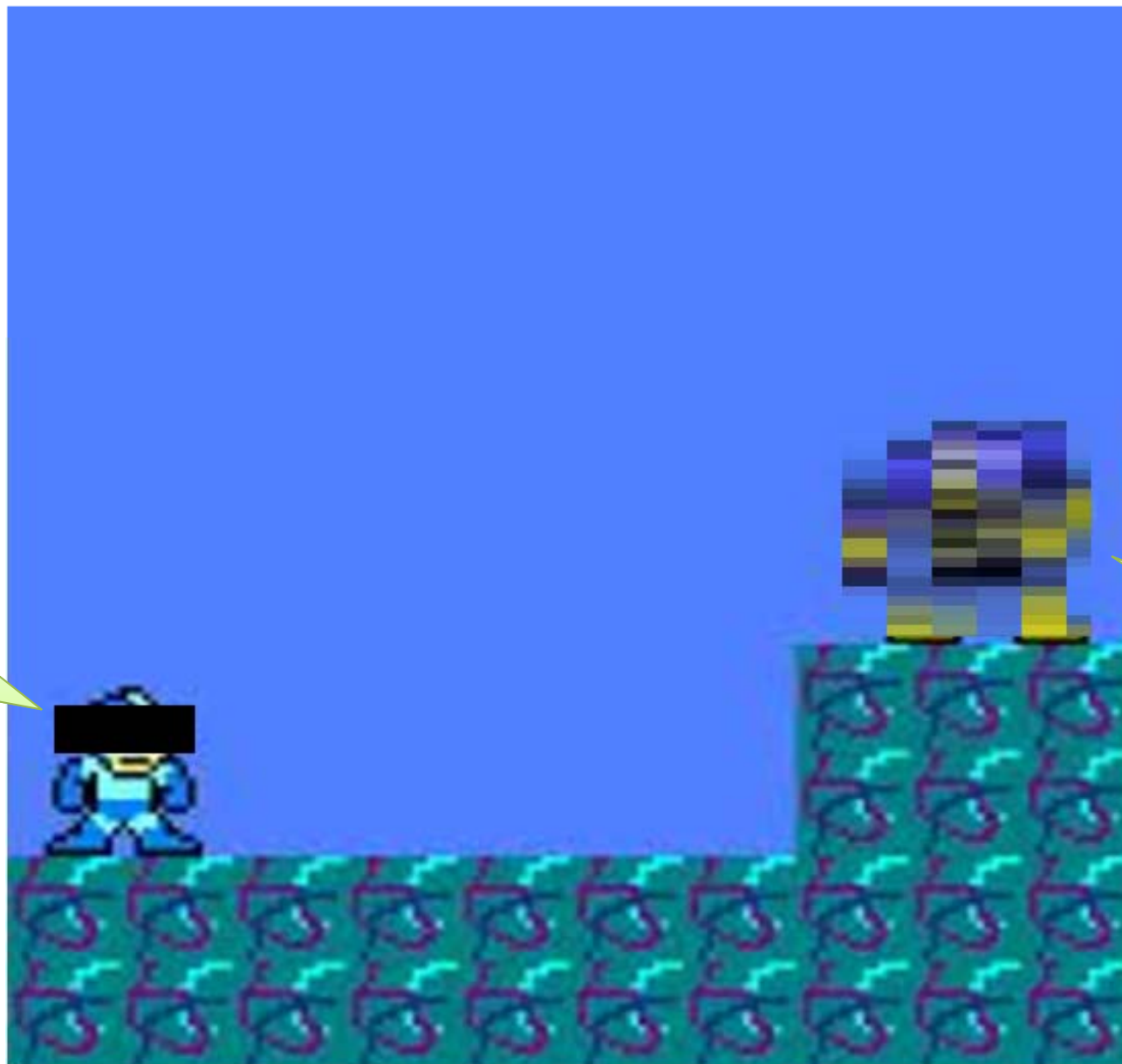
運用

Monitoring

監視

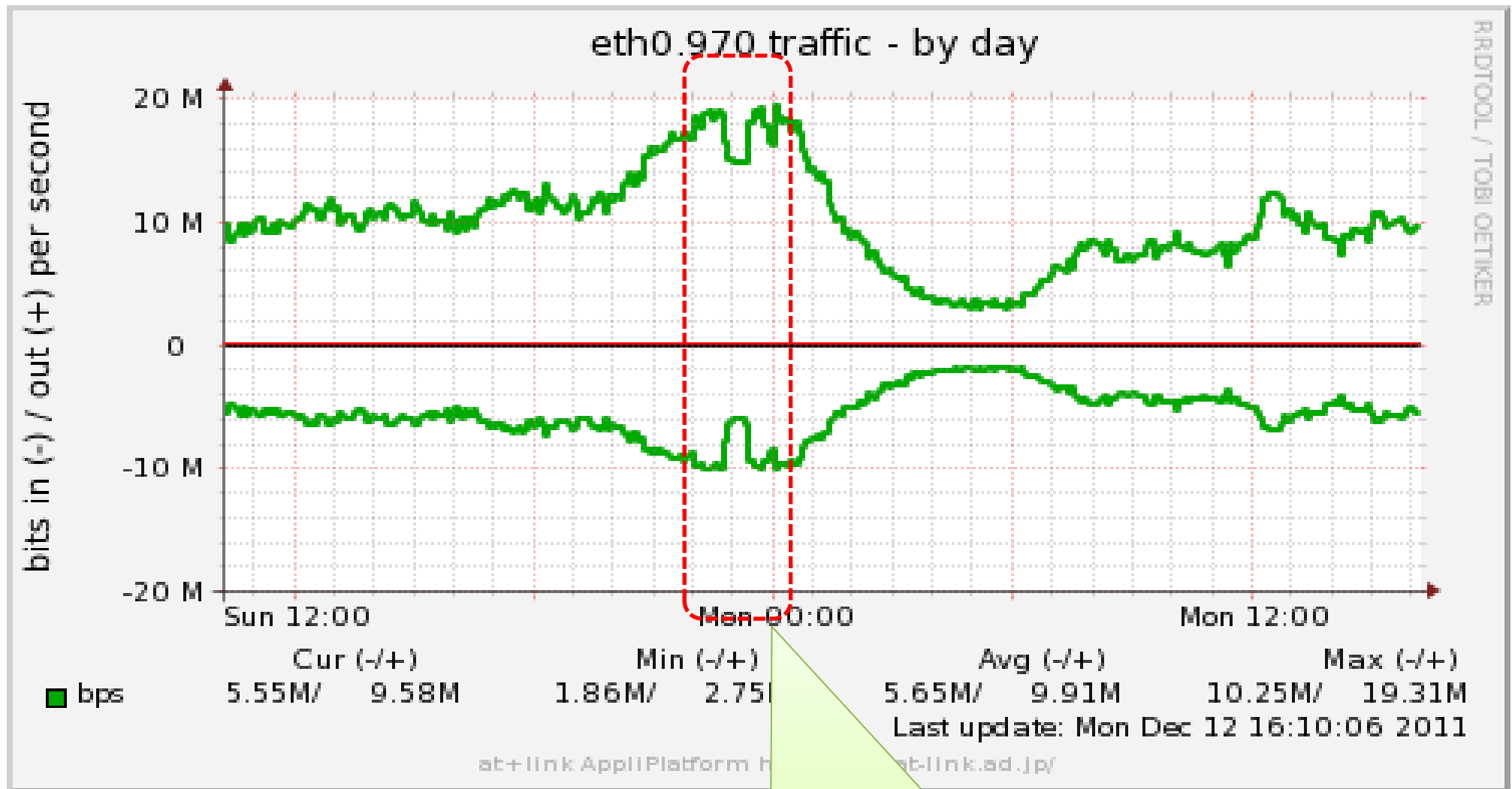
この障害を検知できない！

監視
システム



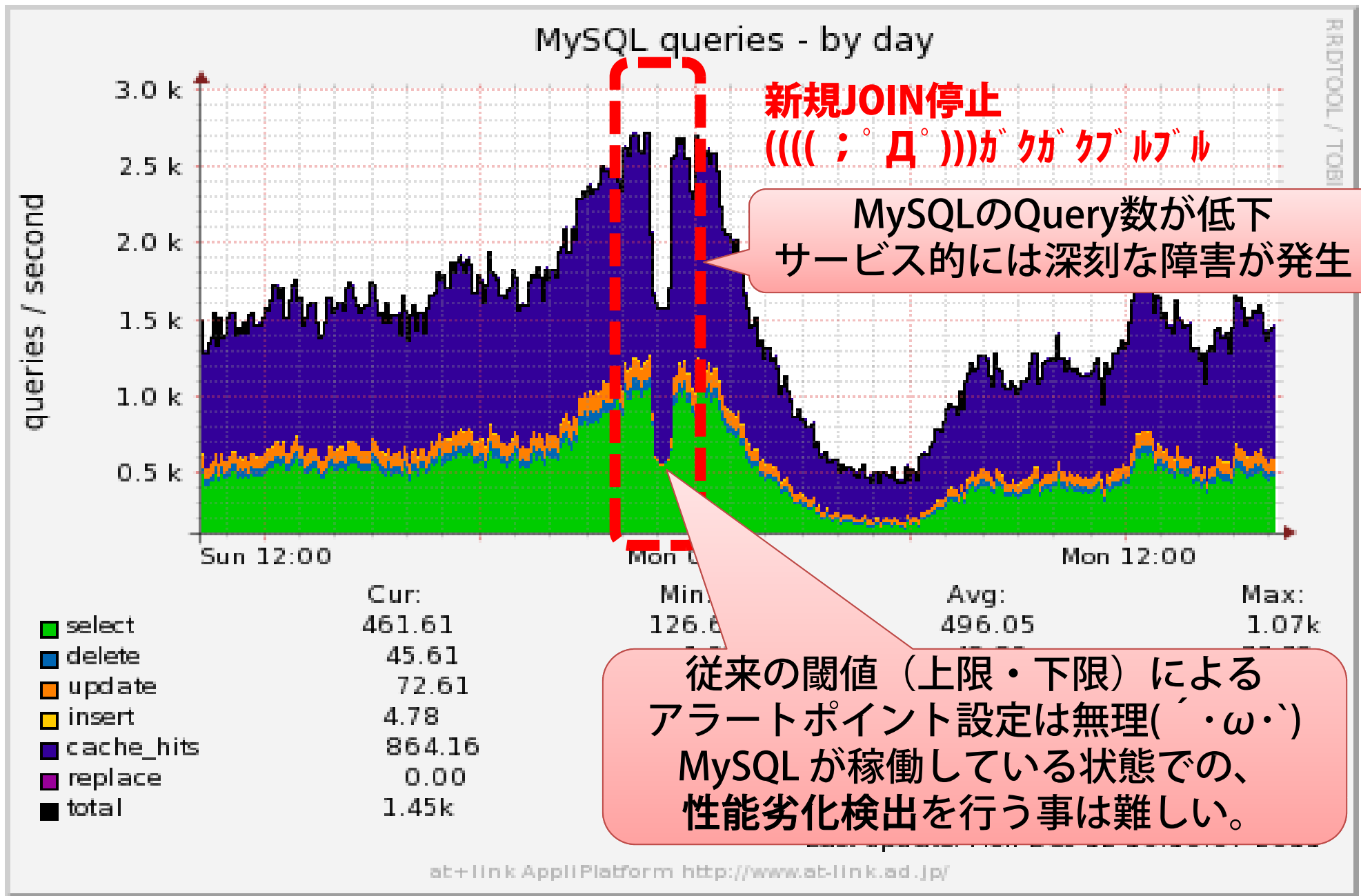
「何度発生
しても検出
できない」
みたいな。
障害の
イメージ…

この障害を検知できない！



凹んだ所、一見すると…単純な
トラフィック微減に見えますが

この障害を検知できない！



そこで考えた監視手法

パフォーマンスの劣化を検出するためには…

- [1] 人が24時間365日張り付く！

(;´Д`)

運用の現場的には辛いけれど
監視が自動化出来ない以上、
有効な手段ではないでしょうか…。

- [2] 新しい監視手法を作る！

(`・ω・´)

今手許のツールが出来ないから諦めるのではなく、
必要に迫られ、必要なアプローチで取り組むのも
正直アリでは？

at+link アプリプラットフォーム

- 監視は2段階

サービスとして提供はしていますが...

- 標準死活監視

- Nagios ベースの PING, SSH, HTTP ポート監視

一般的な監視

- HTTP 応答時間監視

- HTTP に対して複数セッション同時監視

ソーシャルアプリ特化監視

- リソースモニタリング

- Munin による視覚化

- 数百ノード、数千metrics

監視対象が増えていくと、
システムリソース的に
正直限界になってきます

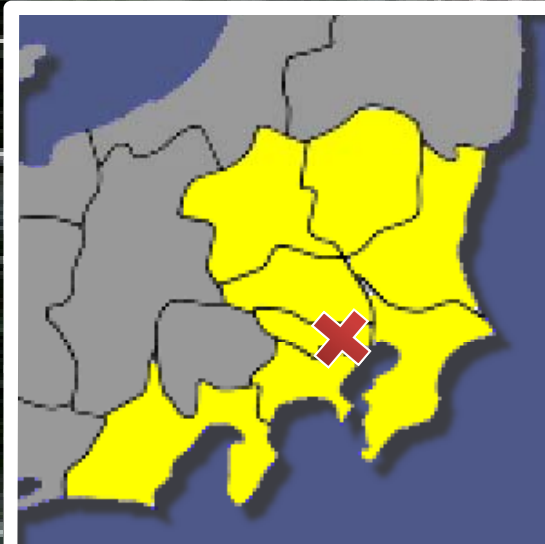
限界

EventDriven

そこで考えてみました
「イベント駆動型監視」の概念。

監視システムから、対象ノードへの
定期的なポーリングではなく、障害
発生を検出や通知を、能動的実行。

理想は「緊急地震速報」のように
障害予測ができる監視・通知システム



緊急地震速報 (at+link)

データセンタで地震 強い揺れに警戒

千葉 茨城 栃木 群馬 埼玉
東京 神奈川 静岡

なぜイベント駆動なのか？

- **従来の時系列・閾値検出は限界**

- 監視ポイントの乗数的増大
- 人間が24時間張り付くコストの問題

今はまだ、人間が対応出来る規模だから、許容されてる

- **結局、障害時は人間が対処するしか無い**

- 起こった上で、迅速に復旧するアプローチ
- イベントやキャンペーン期間中は、どうせ担当者が張り付くのが常だし…であれば、障害時に迅速に状況を把握する方法も必要。

人間が行うオペレーションの補助＝現場の運用負担軽減

イベント駆動型監視システム

①リアルタイム(秒単位)のリソース監視

1分間隔では遅い。短時間の機会損失が大きな金銭的損失に繋がる

②複数リソース(metrics)に対する並列監視

ポートの応答(0 or 1)ではなく、リソース推移も監視対象

③リスクアセスメントと通知

「障害発生から
復旧までの時間を短くしたい！」

目下サービス化に向けて開発中で御座います。
皆様から、ご意見・ご要望を頂ければ幸いですm(__)m

DEMONSTRATION

画面デモ

…の前に MUNIN

- Munin はリソース変化を把握するツール
 - RRDtoolベースでグラフ化する機能
 - アラート機能
 - Munin2.0 RC4がリリース (3/24)

詳しい資料は
こちらご覧下さい

 **MUNIN**で、
今日から始める
リソース監視

Munin User Group Japan
<http://munin.jp/>

@zembutsu

Dec 9, 2011 at IBM@SHIBUYA MARKCITY

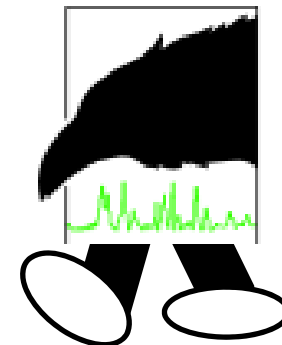
Let's begin resource monitoring with
munin 2011 1209 zem_distribution
<http://www.slideshare.net/zembutsu/lets-begin-resource-monitoring-with-munin-2011-1209-zemdistribution>



slideshare
Present Yourself

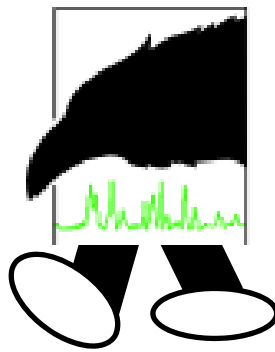
Muninet, Muninwalk 作ってみた

- snmpwalk, snmpget を
~~パクリました~~インスパイア(ｷﾘｯ
- Muninの値をコマンドラインで取得
 - Muninwalk … 特定ホストの fetch
 - Muninet … 複数ホストの fetch をループ
- Perl です
- OpenSource 予定
- Github で公開予定



<http://github.com/zembutsu/muninwalk/>

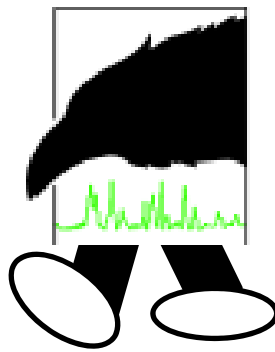
Muninwalk



- **snmpwalk** のような挙動
 - MIB (Management Information Base) や
OID (Object Identifier) のようなデータ構造で
Munin の値を取得する CLI (Command Line Interface)
 - **snmpd** (TCP Port 161) に替わり
munin-node(4949)と通信する

```
$ muninwalk 210.239.46.254 cpu
210.239.46.254::cpu.user = 101873560
210.239.46.254::cpu.nice = 40440522
210.239.46.254::cpu.system = 21888721
210.239.46.254::cpu.idle = 1969063306
210.239.46.254::cpu.iowait = 49614151
210.239.46.254::cpu irq = 261758
210.239.46.254::cpu.softirq = 1489834
210.239.46.254::cpu.steal = 0
```

Muniget

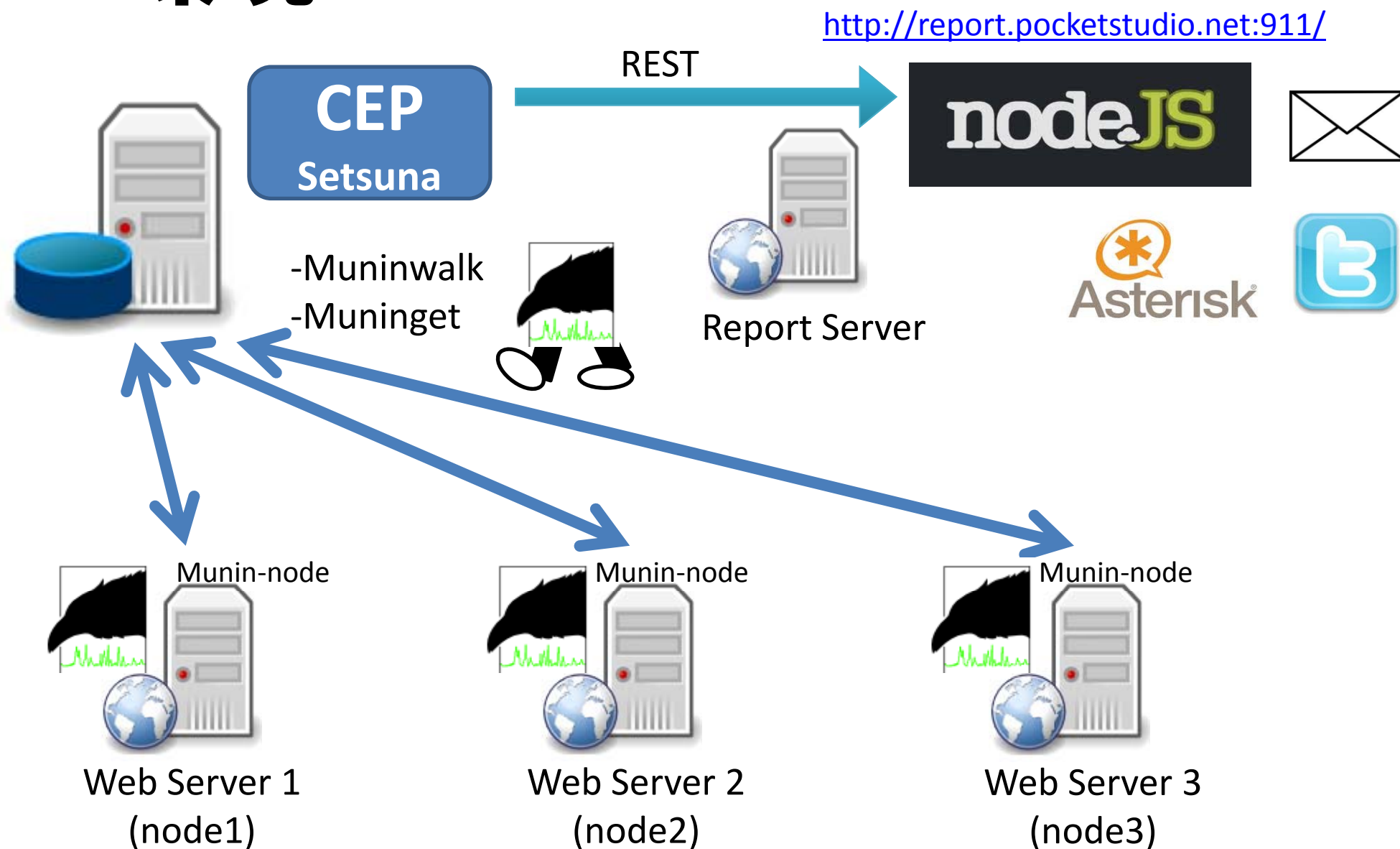


- **snmpget のような挙動**
 - Muniget の branch
 - 複数の munin-node と通信し、値を取得
 - 一定間隔(単位ミリ秒)の単純ループ

```
$ muniget node1,node2,node3 load1 -s1.0
12/04/11 10:50:44.131319::node1::load1.load1 = 0.93
12/04/11 10:50:44.181189::node2::load1.load1 = 0.43
12/04/11 10:50:44.231158::node3::load1.load1 = 0.10
12/04/11 10:50:45.285504::node1::load1.load1 = 0.86
12/04/11 10:50:45.336458::node2::load1.load1 = 0.43
12/04/11 10:50:45.386935::node3::load1.load1 = 0.10
12/04/11 10:50:46.440780::node1::load1.load1 = 0.86
12/04/11 10:50:46.491734::node2::load1.load1 = 0.43
12/04/11 10:50:46.542200::node3::load1.load1 = 0.10
12/04/11 10:50:47.595552::node1::load1.load1 = 0.86
12/04/11 10:50:47.647011::node2::load1.load1 = 0.39
12/04/11 10:50:47.697976::node3::load1.load1 = 0.09
```

今回のデモでは、この標準出力を CEP (Setsuna) に流し込みます。条件の判定を行い、トリガが発生すると、レポートを行います。

デモ環境

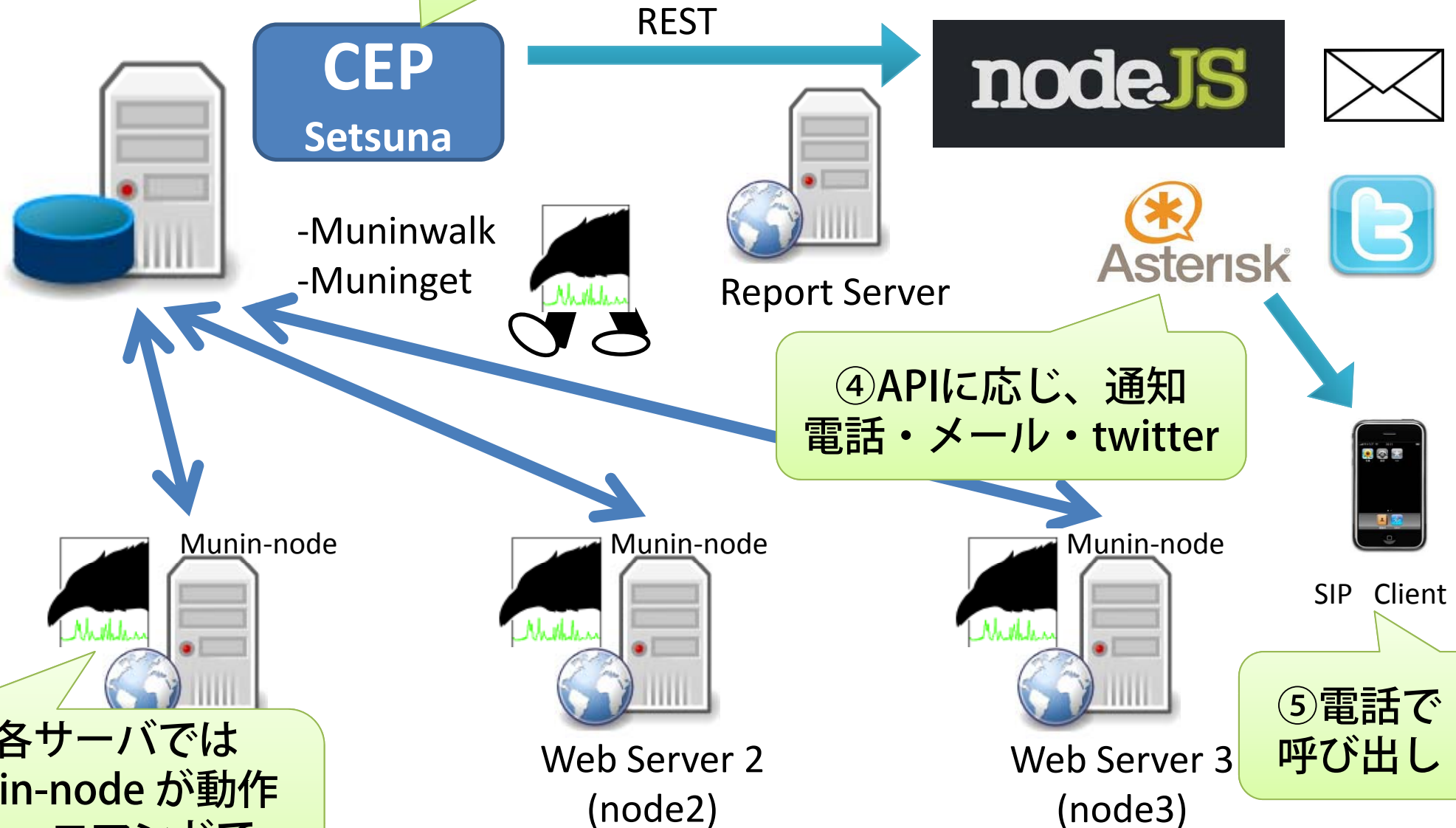


デモ環境

②Muningetの結果を CEP (Setuna) に取り込み、判定する

③Node.js で non-blocking I/O なレポートサーバに通知

<http://report.pocketstudio.net:911/>



①各サーバでは munin-node が動作 stress コマンドで "Load Average 上昇"

⑤電話で呼び出し

デモ内容

- 3台のサーバ (node1, node2, node3) を用意
- ↓
- Muninget の結果を毎秒 Setsuna (CEP)に取り込み
- ↓
- サーバで stress コマンド実行、負荷生成
- ↓
- 1台でも Load Average が上昇すると、
レポートサーバに通知(REST経由)
- ↓
- 電話呼び出し
(Asterisk 経由で、iPhone の SIP clientをcall)



Setsuna でのデータ取り込み

- 事前用意 (alias)

```
$ alias setsuna='/usr/local/jdk1.6.0_23/bin/java -jar /home/zem/develop/setuna/setsuna-0.0.2/setsuna.jar'
```

- Setsuna 単純取り込み

各ホストの Load Average を
muninget で取得し(0.1秒間隔)、
setsuna に渡す

```
$ muninget node1,node2,node3 load1 -s0.1 | setsuna
```

```
{"COLUMN0":"12/04/11","COLUMN1":"12:01:38.442694::node1::load1.load1","COLUMN2":"=", "COLUMN3": "0.64"}  
{"COLUMN0":"12/04/11","COLUMN1":"12:01:38.492657::node2::load1.load1","COLUMN2":"=", "COLUMN3": "0.17"}  
{"COLUMN0":"12/04/11","COLUMN1":"12:01:38.543632::node3::load1.load1","COLUMN2":"=", "COLUMN3": "0.05"}  
{"COLUMN0":"12/04/11","COLUMN1":"12:01:38.698019::node1::load1.load1","COLUMN2":"=", "COLUMN3": "0.64"}  
{"COLUMN0":"12/04/11","COLUMN1":"12:01:38.748482::node2::load1.load1","COLUMN2":"=", "COLUMN3": "0.17"}  
{"COLUMN0":"12/04/11","COLUMN1":"12:01:38.799450::node3::load1.load1","COLUMN2":"=", "COLUMN3": "0.05"}  
{"COLUMN0":"12/04/11","COLUMN1":"12:01:38.955386::node1::load1.load1","COLUMN2":"=", "COLUMN3": "0.64"}  
{"COLUMN0":"12/04/11","COLUMN1":"12:01:39.005834::node2::load1.load1","COLUMN2":"=", "COLUMN3": "0.17"}
```

※参考: Setsuna-0.0.2リリース~ - okuyamaooの日記

<http://d.hatena.ne.jp/okuyamaoo/20120324/1332593522>

ここが Load Average の値

Setsuna の条件判定とイベント

- 条件判定は **-trigger** オプション
 - サーバnode2で stress 実行

```
$ stress --cpu 10
```

```
stress: info: [9285] dispatching hogs: 10 cpu, 0 io, 0 vm, 0 hdd
```

- **Load Average が 2.0 を越えたときだけ画面表示**

```
$ muninget node1,node2,node3 load1 -s0.1 | setsuna -stream load1 -trigger "COLUMN3 > 2.00"
```

```
{"COLUMN0":"12/04/11","COLUMN1":"12:09:36.448781::node2::load1.load1","COLUMN2":"=", "COLUMN3":"2.34"}  
{"COLUMN0":"12/04/11","COLUMN1":"12:09:36.704623::node2::load1.load1","COLUMN2":"=", "COLUMN3":"2.34"}  
{"COLUMN0":"12/04/11","COLUMN1":"12:09:36.960959::node2::load1.load1","COLUMN2":"=", "COLUMN3":"2.34"}  
{"COLUMN0":"12/04/11","COLUMN1":"12:09:37.216300::node2::load1.load1","COLUMN2":"=", "COLUMN3":"2.34"}  
{"COLUMN0":"12/04/11","COLUMN1":"12:09:37.471146::node2::load1.load1","COLUMN2":"=", "COLUMN3":"2.34"}
```

- 条件判定時の処理は **event** オプション

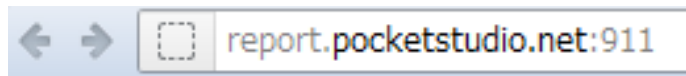
```
$ muninget node1,node2,node3 load1 -s0.1 | setsuna -stream load1 -trigger "COLUMN3 > 2.00" ¥  
event "curl http://report.pocketstudio.net:911/emergency"
```

※ repot サーバは Node.js サーバが Port 911 で待ち受けている。
リクエストを受けると、Asterisk 経由で SIP Client を call する

レポートサーバの挙動

- **Node.js (non-blocking I/O)**

/usr/local/bin/node /home/zem/develop/911/reportServer/911.js



Manual operation (DEBUG)

- 1. EMERGENCY (call+mail+tweet)
- 2. Critical (mail+tweet)
- 3. Warning (tweet)
- 4. Notice (logging)
- 5. Report (logging)

REST件 試験用 Web interface が、
リクエストに応じてレポートを行う。

内部でスクリプトを実行

- mail ... メール送信
- tweet ... Twitter CLI 呼び出し→自分宛mention
- call ... SIP Client の CLI 呼び出し

<http://www.pjsip.org/release/1.12/pjproject-1.12.tar.bz2>

CEPを用いたイベント駆動監視

• 基本コンセプト

- 障害が発生すると、大量のアラートメール
 - 何が本当に問題なのか分かりづらい
 - だから障害リスク評価システムが必要
- クリティカル障害(緊急時)は電話が確実
 - Asterisk(SIP)経由で電話がかかる仕組み
 - 将来的には…
 - 自動的に、関係者全員を電話会議室への誘導
 - システム側の状況は、text2speech が随時読み上げ
- すべては、**迅速な障害対応・復帰のため。**

今回のデモ環境で、コンセプトに応じた通知が可能である事や、
アイデアを皆様と共有できたと思っています。

…という名の、絵に描いた餅(;´Д`)

理想



イベント駆動型監視システム サービス化に向けての課題

- **ダッシュボード(GUI)開発**
- **精度向上**
 - CEP(setsuna)調整
- **複数の監視手法の組み合わせ**
 - 閾値指定,推定,時系列分析(Holt-Winters法等)
- **リスクアセスメント評価に応じた通知**
 - 評価：統計的手法を用いる
 - 通知：メール, twitter, 電話連動
(電話会議・text2speech)

汎用サービス化を目指す
と、課題や障壁が多いです。

目下サービス化に向けて開発中で御座います。
皆様から、ご意見・ご要望を頂ければ幸いですm(__)m

0xXX

超宣伝タイム!!

スーパー宣伝タイム1

- セミナーでお時間をいただきます！

第2回 『いまさら聞けない！システム運用・管理のコツ』
5/24（木） 19:00 – 22:00

- <http://everevo.com/event/1334>

データセンタが（障害で）静止する日。
～元現場運用者が語る、修羅場のくぐり方～

スーパー宣伝タイム2

- **社員募集中！**

- 株式会社リンク アプリプラットフォーム部

- **ioDriveと戯れる簡単なお仕事です**

- 興味がありましたら、中の人へDMくださいあ

- **／人・人＼**

- 「僕(の会社)と(雇用)契約して、
インフラエンジニアになってよ」

Thank you for listening!

- 最後までおつきあいいただき、ありがとうございました!!

- **Contacts**

- at+link 専用サーバ・サービス アプリプラットフォーム
http://www.at-link.ad.jp/appli_platform/
- 株式会社リンク ディベロッパーサポート部
前佛 雅人 (Masahito Zembutsu)
 - Twitter: @zembutsu
 - Facebook: <http://facebook.com/zembutsu/>
 - E-mail: zembutsu@link.co.jp
 - Tel: 03-5785-0555

at+link