Muninでリソース監視

ENOG20

2013年4月26日

株式会社グローバルネットコア

羽賀　晴彦

<haruhiko.haga@global-netcore.jp>

アジェンダ

- Muninとは
- Munin-nodeのインストール
- Muninのインストール
- プラグインの作成
- 閾値の設定
- 過去のグラフが見たい
- Nagiosとの連携
- 負荷対策

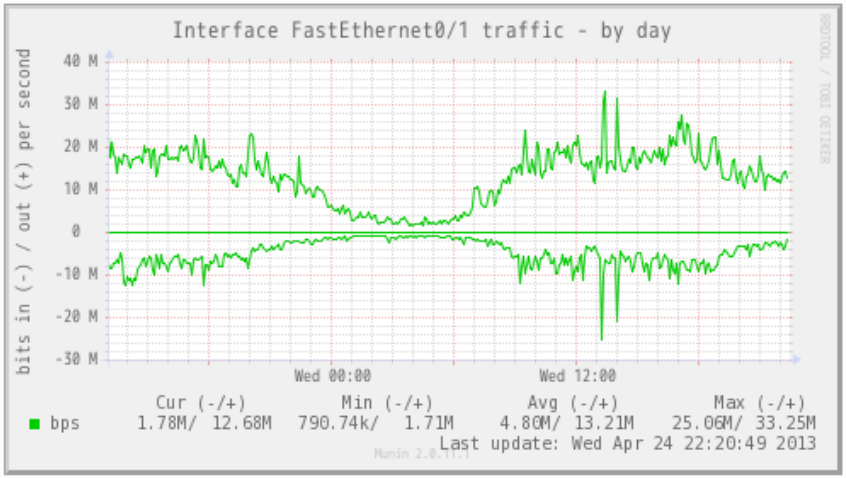
Muninとは(1)

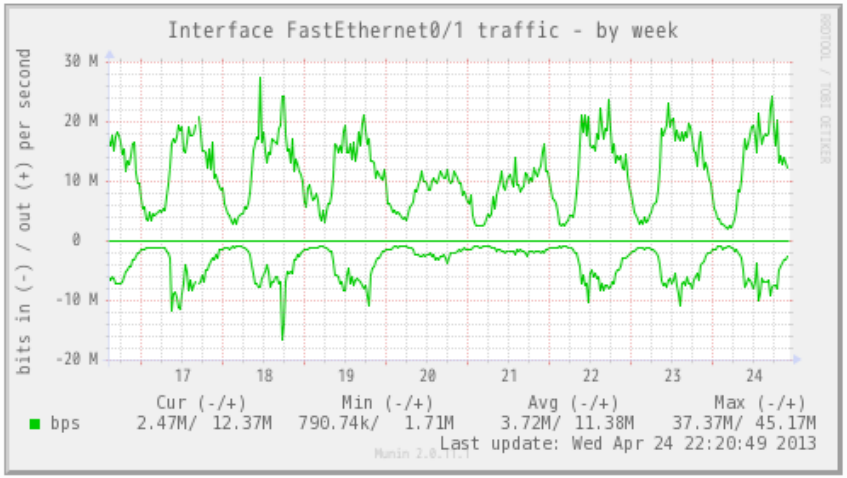
- サーバ、ネットワーク機器などのリソースをモニタリングするためのツールです。
- 代表的なツール
  - MTRG ←いままでこれを使ってました
  - cacti ←むかし検証しましたが、、、

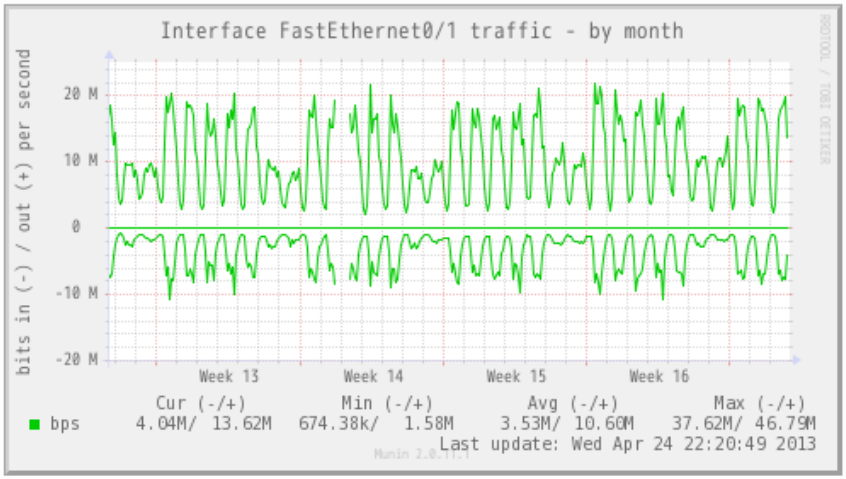
Muninとは(2)

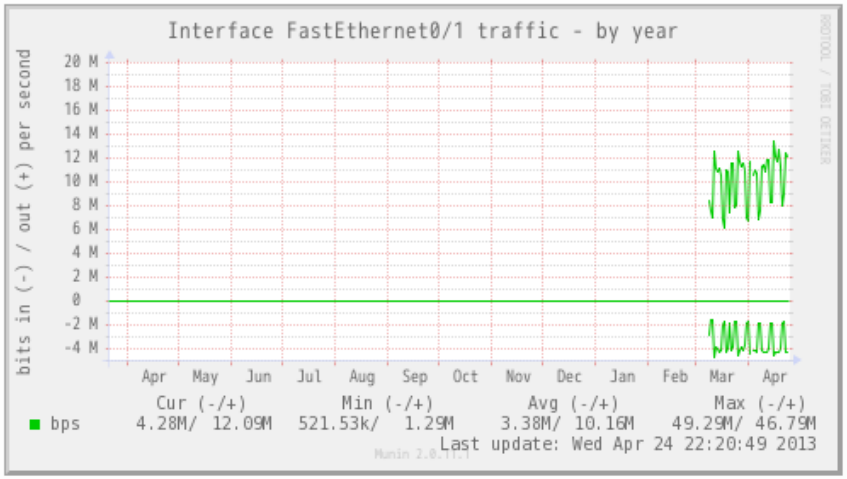
- 公式ページ
  - http://munin-monitoring.org/
- Stable
  - Munin 2.0.12
  - (2013/4/25 現在)

画像イメージ



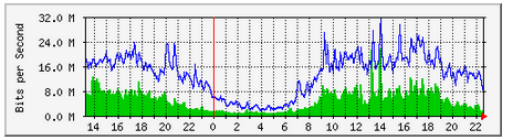






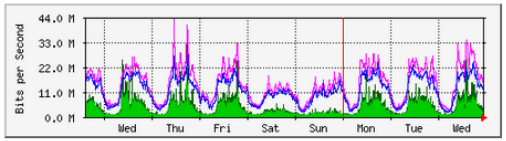
MRTGの画像イメージ

`Daily' Graph (5 Minute Average)



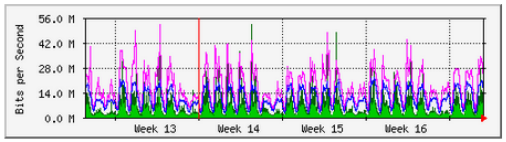
	Max	Average	Current
In	21.7 Mb/s (21.7%)	4873.0 kb/s (4.9%)	1687.3 kb/s (1.7%)
Out	31.2 Mb/s (31.2%)	13.4 Mb/s (13.4%)	10.6 Mb/s (10.6%)

`Weekly' Graph (30 Minute Average)



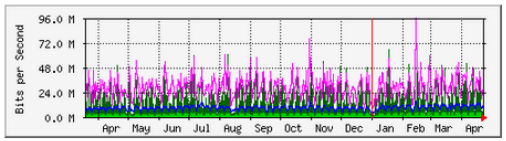
	Max	Average	Current
In	40.4 Mb/s (40.4%)	3731.1 kb/s (3.7%)	2361.8 kb/s (2.4%)
Out	43.2 Mb/s (43.2%)	11.5 Mb/s (11.5%)	12.0 Mb/s (12.0%)

`Monthly' Graph (2 Hour Average)



	Max	Average	Current
In	52.2 Mb/s (52.2%)	3539.4 kb/s (3.5%)	4038.2 kb/s (4.0%)
Out	52.0 Mb/s (52.0%)	10.6 Mb/s (10.6%)	13.7 Mb/s (13.7%)

`Yearly' Graph (1 Day Average)



	Max	Average	Current
In	65.6 Mb/s (65.6%)	2689.4 kb/s (2.7%)	4365.8 kb/s (4.4%)
Out	95.1 Mb/s (95.1%)	8560.0 kb/s (8.6%)	12.3 Mb/s (12.3%)

MRTG(グラフ作成までの流れ)

- 監視対象となる機器にsnmpを設定
  - 場合によってはMIB情報に渡すスクリプトも設置
  - 稼働サービスによって設置スクリプトが異なる
    - MySQLのクエリ数
    - SMTP接続数　　etc...
- iptablesやFWでsnmpを許可
- MRTGのサーバに対象のconfigを作成
  - 稼働サービスによってconfigの内容が異なる

MRTG(グラフ作成の悩み)

- snmpの設定をするのはけっこう面倒
- 特にスクリプトの設置
- サーバによってweb, mail, DB など、稼働サービスは様々

Muninにした背景(1)

- 近年、管理する機器（仮想サーバ含む）が倍以上に増加
- サーバ構築の機会も増え、SNMP, MRTGを設定する機会も増加
- 結果、設定にかなりの時間を割く事に・・・
  - 他にも、サーバスペック不足、機器保守切れなども要因

↓

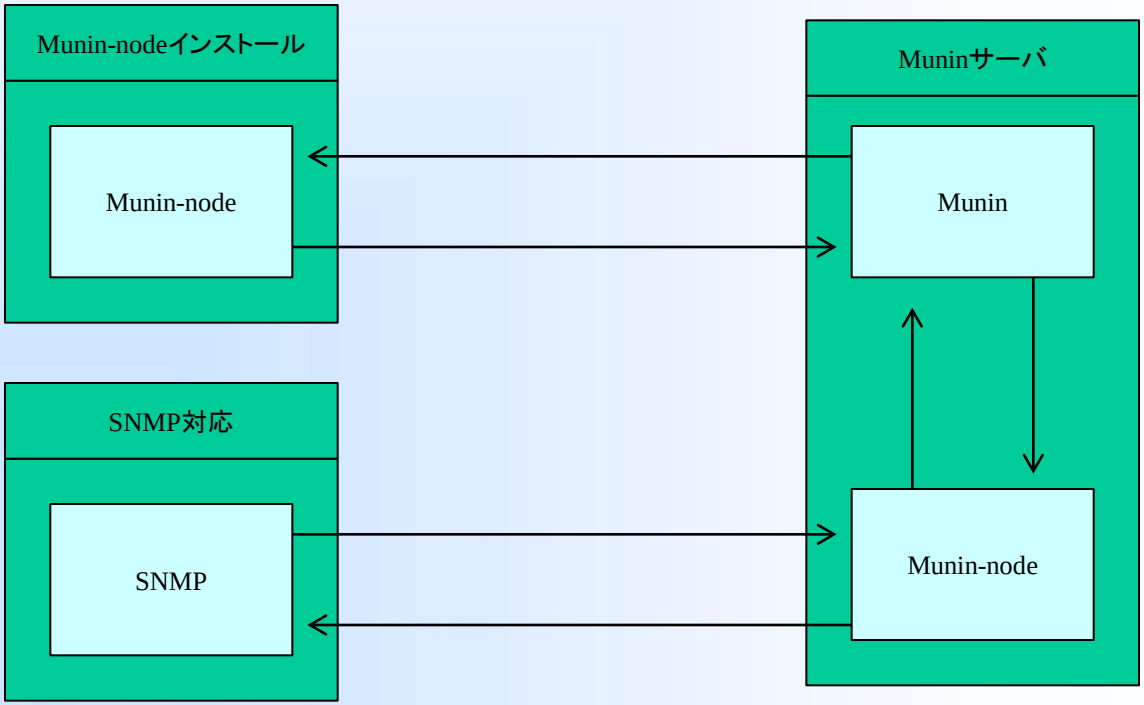
- 求める機能
  - とにかく、設定が楽なもの
  - SNMPより、エージェント型がいいな～
  - SNMPでのデータ取得にも対応
  - 閾値監視

Muninにした背景(2)

- 比較的、設定が簡単で日本語の情報が多いMuninを採用する事に。
  - 1系は動作が重い。
  - 2系では改善された、らしい。
- 過去にcactiを検証したが、動作が重く使いづらかった。
  - 今は改善されているかも
  - GUIで設定できたのは魅力的

Muninの監視イメージ

- muninはcronで定期的に実行
- munin-nodeに接続してデータを取得
- munin-nodeはデーモンとして稼働



munin-nodeのインストール

munin-nodeのインストール(1)

CentOS 6 (64bit)の場合

```
# rpm -ivh http://ftp.jaist.ac.jp/pub/Linux/Fedora/epel/6/x86_64/epel-release-6-8.noarch.rpm
# yum install munin-node
# chkconfig --add munin-node
# chkconfig munin-node on
```

```
# munin-node-configure --shell
```

利用できるプラグインの一覧が表示されるので、、、

```
ln -s '/usr/share/munin/plugins/if_err_' '/etc/munin/plugins/if_err_eth0'
ln -s '/usr/share/munin/plugins/if_err_' '/etc/munin/plugins/if_err_eth1'
ln -s '/usr/share/munin/plugins/interrupts' '/etc/munin/plugins/interrupts'
ln -s '/usr/share/munin/plugins/ntp_kernel_err' '/etc/munin/plugins/ntp_kernel_err'
```

```
# munin-node-configure --shell | sh
```

/etc/munin/plugins にシンボリックリンクを作成する事で監視できる。

設定後はサービスを再起動

```
# /etc/init.d/munin-node restart
```

munin-nodeのインストール(2)

ソースからインストールする場合

```
# groupadd munin
# useradd -d /var/lib/munin -s /sbin/nologin munin

cpan> install File::Path
cpan> install Net::SNMP
cpan> force install Net::Server
cpan> install ExtUtils::MakeMaker
cpan> install Perl::OSType
cpan> install Module::Build

# wget http://sourceforge.net/projects/munin/files/stable/2.0.6/munin-2.0.6.tar.gz/download
# tar zxfv munin-2.0.6.tar.gz
# cd munin-2.0.6
```

munin-nodeのインストール(3)

```
# vi Makefile
```

```
-CONFIG = Makefile.config
+CONFIG = dists/redhat/Makefile.config
```

```
# make
# make install-common-prime install-node-prime install-plugins-prime
# cp dists/redhat/munin-node.rc /etc/init.d/munin-node
# chmod 755 /etc/init.d/munin-node
# chkconfig --add munin-node
```

Makefileを修正する事でパッケージで入れた時と同じパスに展開されます。
ただし、dists/redhat/* ファイルが用意されているのは「2.0.6」まで

/etc/munin/plugin-conf.d/munin-node

- ここにプラグインの個別情報を設定できる。
- プラグイン名の指定は「*」も設定可。
- プラグインは通常「munin」ユーザで実行される。
- プラグイン内の環境変数を変更できる。

```
[diskstats]          ←プラグイン名
user munin           ←プラグインの実行ユーザ

[iostat_ios]
user munin

[if_*]
user root

[mysql*]
env.mysqladmin /usr/local/mysql/bin/mysqladmin
env.mysqlopts -u root -pXXXXX
```

プラグインを直接実行

```
# munin-run if_eth0
down.value 78424950503
up.value 52005046510

# munin-run df
_dev_sda3.value 1.71663176899059
_dev_shm.value 0
_dev_sda1.value 28.4778598276421
```

/etc/munin/munin-node.conf

接続元IPアドレスの許可くらいは入れておきましょう。

```
host_name sv01.example.jp

allow ^192¥.168¥.100¥.10$
allow ^192¥.168¥.100¥.11$

# cidr_allow 127.0.0.1/32
# cidr_allow 192.0.2.0/24
# cidr_deny  192.0.2.42/32

port 4949
```

Port 4949に接続してみた

```
# telnet sv01.example.jp 4949
Trying 192.168.100.10...
Connected to sv01.example.jp.
Escape character is '^]'.
# munin node at sv01.example.jp
```

muninのインストール

muninのインストール(1)

CentOS 6 (64bit)の場合

```
# rpm -ivh http://ftp.jaist.ac.jp/pub/Linux/Fedora/epel/6/x86_64/epel-release-6-8.noarch.rpm
# yum install munin
# yum install munin-cgi
# chkconfig --add munin-node
# chkconfig munin-node on
```

muninのインストール(2)

ソースからインストールする場合

```
# groupadd -g 499 munin
# useradd -d /var/lib/munin -u 498 -g 499 -s /sbin/nologin munin

cpan> install File::Path
cpan> install Net::SNMP
cpan> force install Net::Server
cpan> install ExtUtils::MakeMaker
cpan> install Perl::OSType
cpan> install Module::Build

# wget http://sourceforge.net/projects/munin/files/stable/2.0.6/munin-2.0.6.tar.gz/download
# tar zxfv munin-2.0.6.tar.gz
# cd munin-2.0.6
```

muninのインストール(3)

```
# vi Makefile
```

```
-CONFIG = Makefile.config
+CONFIG = dists/redhat/Makefile.config
```

```
# make
# make install
# cp dists/redhat/munin-node.rc /etc/init.d/munin-node
# chmod 755 /etc/init.d/munin-node
# chkconfig --add munin-node
# cp dists/redhat/munin.cron.d /etc/cron.d/munin
```

/etc/munin/munin.conf

```
htmldir /var/www/html/munin
includedir /etc/munin/conf.d
cgiurl_graph /cgi-bin/munin-cgi-graph
graph_strategy  cron
html_strategy cron
contact.mail.command       mail -s "Munin ${var:group}::${var:host}" hhaga@example.jp
contact.mail.always_send   critical

[GroupA;]
contacts        mail
[GroupB;]
contacts        mail
```

/etc/munin/conf.d/*

- 監視対象の設定ファイルはサーバ毎に別ファイルで保管できる。
- もちろん、1ファイルで管理も可
  - sv01.example.jp.conf

```
[GroupA;sv01.example.jp]
        address sv01.example.jp
        use_node_name yes
```

  - sw01.example.jp.conf

```
[GroupB;sw01.example.jp]
        address 127.0.0.1
        use_node_name no
```

SNMP経由での監視

- ネットワーク機器など、munin-nodeをインストール出来ない機器の場合には、snmp経由でデータを取得する事が可能
- 標準でいくつかのプラグインが用意されています。

snmp__cpuload snmp__df snmp__df_ram snmp__fc_if_ snmp__fc_if_err_ snmp__if_ snmp__if_err_ snmp__if_multi snmp__load snmp__memory snmp__netapp_diskusage_ snmp__netapp_inodeusage_ snmp__netstat snmp__print_pages	snmp__print_supplies snmp__processes snmp__rdp_users snmp__sensors_fsc_bx_fan snmp__sensors_fsc_bx_temp snmp__sensors_fsc_fan snmp__sensors_fsc_temp snmp__sensors_mbm_fan snmp__sensors_mbm_temp snmp__sensors_mbm_volt snmp__swap snmp__uptime snmp__users snmp__winload snmp__winmem

プラグインの設定

- Masterサーバで設定します。
  - プラグインで指定しているMIB情報が取得出来るかを確認します

```
# munin-node-configure ¥
 --snmpversion 1 ¥
 --snmpcommunity public ¥
 --snmp sw01.example.jp ¥
 --shell

ln -s '/usr/share/munin/plugins/snmp__if_' '/etc/munin/plugins/snmp_sw01.example.jp_if_1'
ln -s '/usr/share/munin/plugins/snmp__if_' '/etc/munin/plugins/snmp_sw01.example.jp_if_2'
ln -s '/usr/share/munin/plugins/snmp__if_' '/etc/munin/plugins/snmp_sw01.example.jp_if_3'
ln -s '/usr/share/munin/plugins/snmp__if_err_' '/etc/munin/plugins/snmp_sw01.example.jp_if_err_1'
ln -s '/usr/share/munin/plugins/snmp__if_err_' '/etc/munin/plugins/snmp_sw01.example.jp_if_err_2'
ln -s '/usr/share/munin/plugins/snmp__if_err_' '/etc/munin/plugins/snmp_sw01.example.jp_if_err_3'
ln -s '/usr/share/munin/plugins/snmp__if_multi' '/etc/munin/plugins/snmp_sw01.example.jp_if_multi'
ln -s '/usr/share/munin/plugins/snmp__uptime' '/etc/munin/plugins/snmp_sw01.example.jp_uptime'
```

/etc/munin/plugin-conf.d/munin-node

- snmpバージョン、コミュニティ名などの情報はここに登録しておきます。

```
[snmp_*]
env.timeout 20
env.version 1
env.community public

[snmp_sw02.example.com_*]
env.community public_test
```

/etc/munin/conf.d/*

- addressの指定に注意

```
[GroupA;sw01.example.jp]
        address 127.0.0.1
        use_node_name no
```

- 設定後は/etc/init.d/munin-node restart で反映

プラグインの作成

プラグインの作成(1)

- 標準のプラグインでは取得できない、という場合
  - 独自にMIB情報に渡していた値
  - ベンダー独自のMIB
  - qmail関連の値

  ↓
- これらを取得するプラグインを作成しました。

プラグインの作成例(1)

- プラグインを作成してみる。
  - Loadアベレージのデータを取得

```
# cat /proc/loadavg
0.96 1.54 1.52 1/210 24948
```

```
# echo -n "load.value "; cut -d' ' -f2  /proc/loadavg
```

- この1行だけでプラグインとしては成立する。

プラグインの作成例(2)

- グラフタイトル、ラベル、グラフの種類などの情報を出力

```
#!/bin/sh

if [ "$1" = "config" ]; then
    echo 'graph_title Load average '
    echo 'graph_args --base 1000 -l 0 '
    echo 'graph_vlabel load '
    echo 'graph_category system '
    echo 'load.label load '
    echo 'load.info 5 minute load average '
    echo 'load.draw LINE '
    exit 0
fi

echo -n "load.value "
cut -f2 -d' ' < /proc/loadavg
```

プラグインの作成例(3)

- munin-node-configure --shell で登録するためには

```
if [ "$1" = "autoconf" ]; then
        echo yes
        exit 0
fi
```

- 条件を付けてみる

```
LOADAVG=/proc/loadavg

if [ "$1" = "autoconf" ]; then
        if [ -f ${LOADAVG} ] ; then
                echo yes
                exit 0
        else
                echo no
                exit 0
        fi
fi
```

プラグインの作成例(4)

- あらかじめ閾値を登録しておくことも可能

```
echo  “load.warning 5 “
echo  “load.critical 10 “
```

プラグインの作成例(5)

- 実際に登録

```
# munin-node-configure --shell
ln -s '/usr/share/munin/plugins/load' '/etc/munin/plugins/load '
# munin-node-configure -shell | sh

# munin-run load
load.value 1.25

# munin-run load config
graph_title Load average
graph_args --base 1000 -l 0
graph_vlabel load
graph_scale no
graph_category system
load.label load
load.info 5 minute load average
load.draw LINE
```

閾値の設定

閾値の設定

- 監視項目に閾値を設定し、メールを送信させる事が可能です。
  - メールだけでなく、スクリプト実行も可能
  - 指定されたコマンドを実行しているだけです。
- /etc/munin/munin.conf

```
contact.mail.command       mail -s "Munin ${var:group}::${var:host}" kanshi@example.jp
contact.mail.always_send   critical

[GroupA;]
contacts        mail
[GroupB;]
contacts        mail
```

閾値の設定場所

- 標準のプラグインに既に設定されているものも多数
- Node, Master どちらでも設定可能
- 設定するのはお好みで
- ただし、snmpで取得する場合はmaster側でしか設定できない
- 当社ではMaster側で閾値設定する方法を採用

Munin-node側で閾値設定

- /etc/munin/plugin-conf.d/munin-node に設定

```
[load]
   env.warning 10
   env.critical 20
```

Munin側で閾値設定(1)

- /etc/munin/conf.d/sv01.example.jp.conf
  - パーティションごとに設定可能
  - デバイス名を調べる手間が必要

```
[GroupA;sv01.example.jp]
        address sv01.example.jp
        use_node_name yes

        df._dev_sdb1.warning                    92
        df._dev_sdb1.critical                   98
        df._dev_sda1.warning                    92
        df._dev_sda1.critical                   98
        df._dev_sda3.warning                    92
        df._dev_sda3.critical                   98
```

Munin側で閾値設定(2)

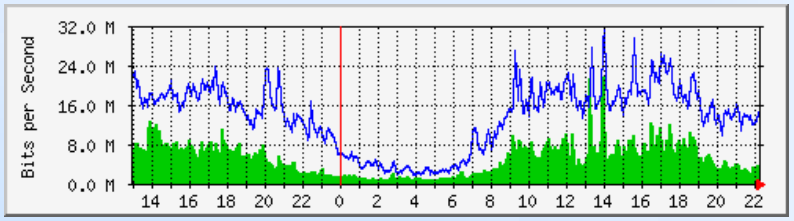
- 通知しない事も可能

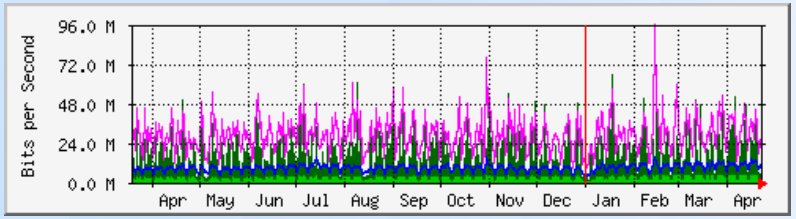
```
diskstats_latency.contacts              no
df_inode.contacts                       no
munin_stats.contacts                    no
if_err_eth0.contacts                    no
if_err_eth1.contacts                    no
open_files.contacts                     no
```

過去のグラフが見たい

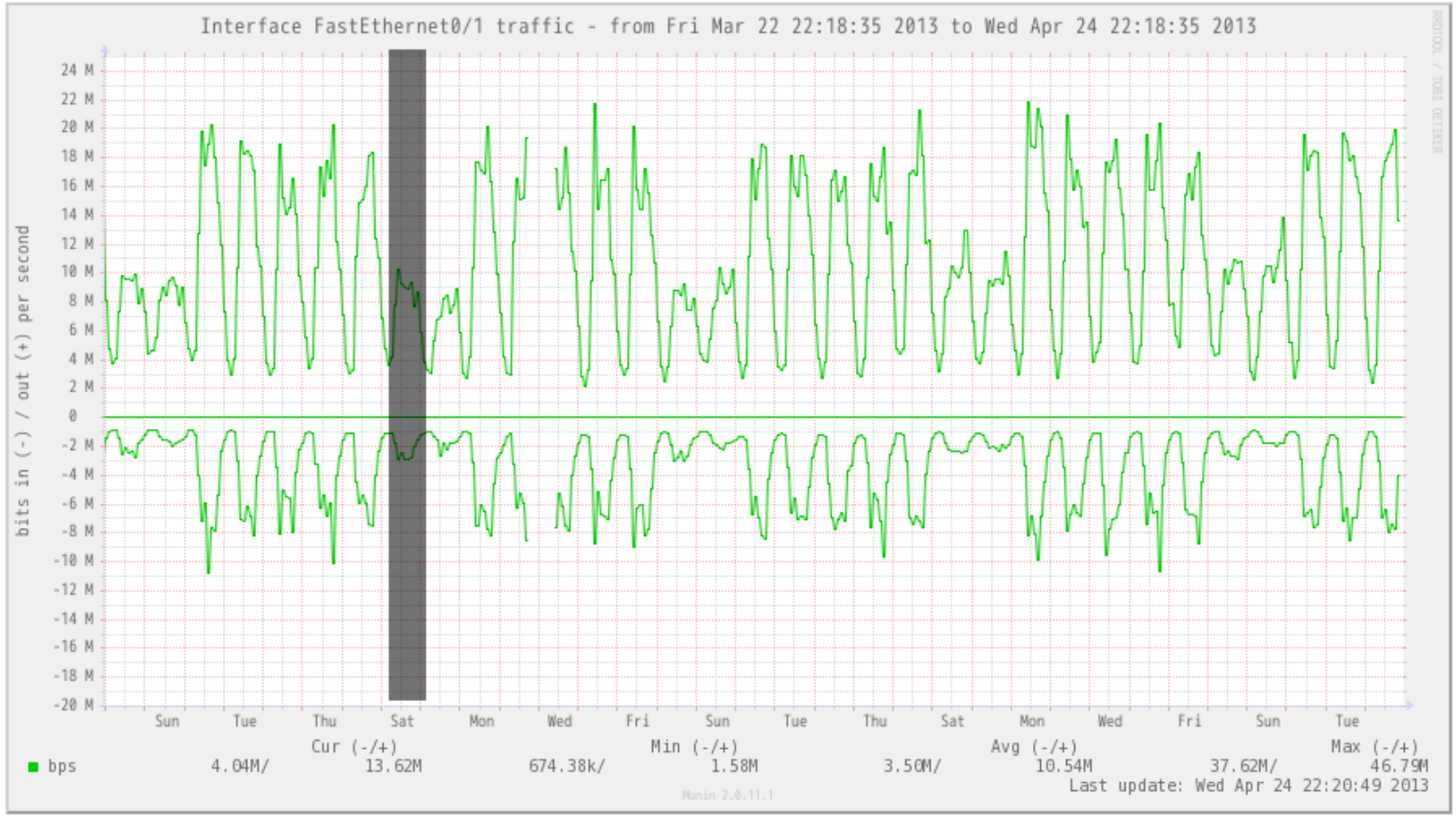
過去のグラフが見たい

- MRTGって過去のグラフは丸められていた。
  - Yearlyは1日平均のグラフなってしまう・・・
  - X月Y日のグラフを見たい、なーんて事はできなかった・・・

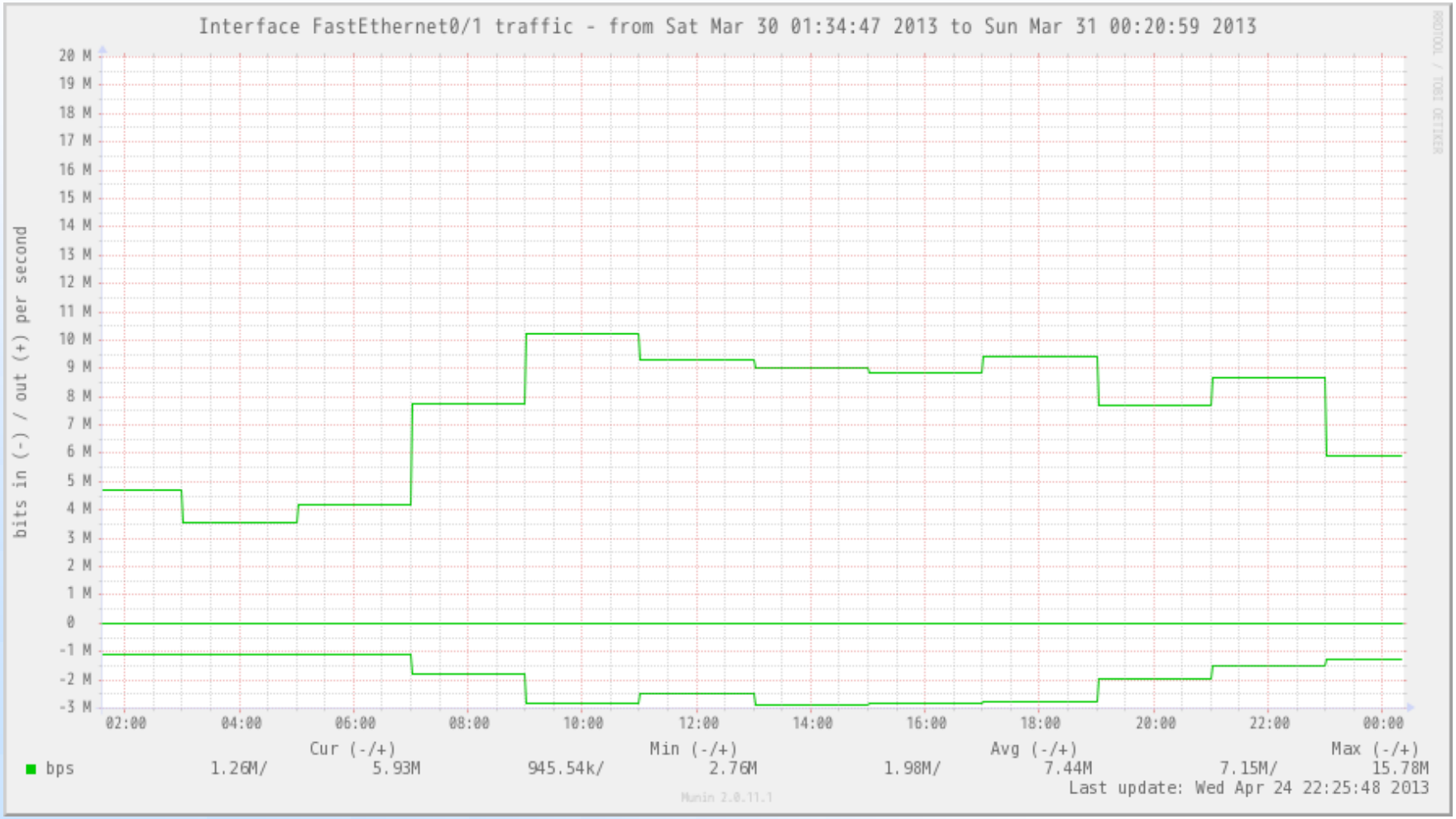




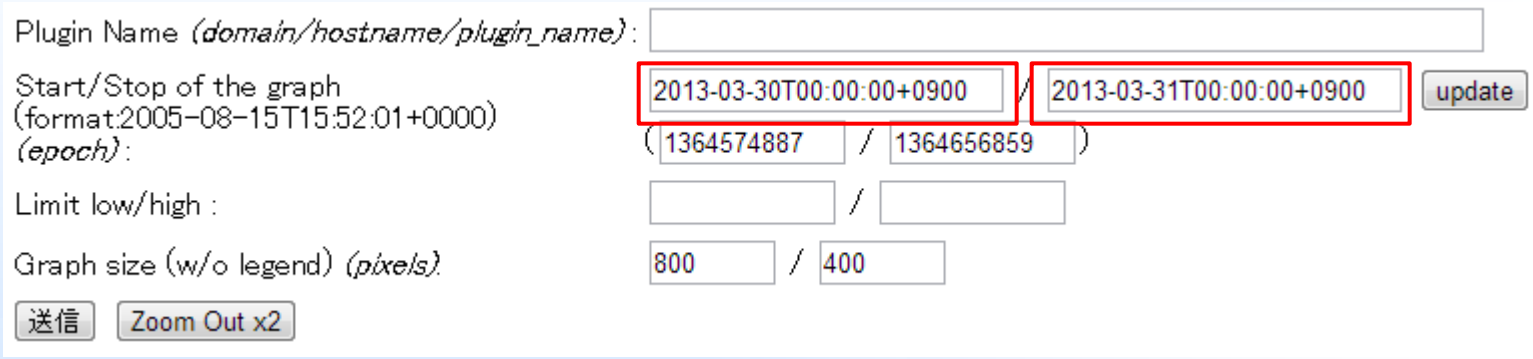
でも、Muninなら



できちゃうんです



正確に日時を指定する事も可能



過去のグラフが見たい

- データが丸められないので、過去のグラフを見返したい時に便利。
  - 例えば、障害時の負荷状況を見たい。
- 1年前までのグラフであれば拡大表示で見る事が可能。
- レポートなどを作成する時に便利
  - 3月1日から3月31日まで、期間指定

Nagiosとの連携

Nagiosとの連携

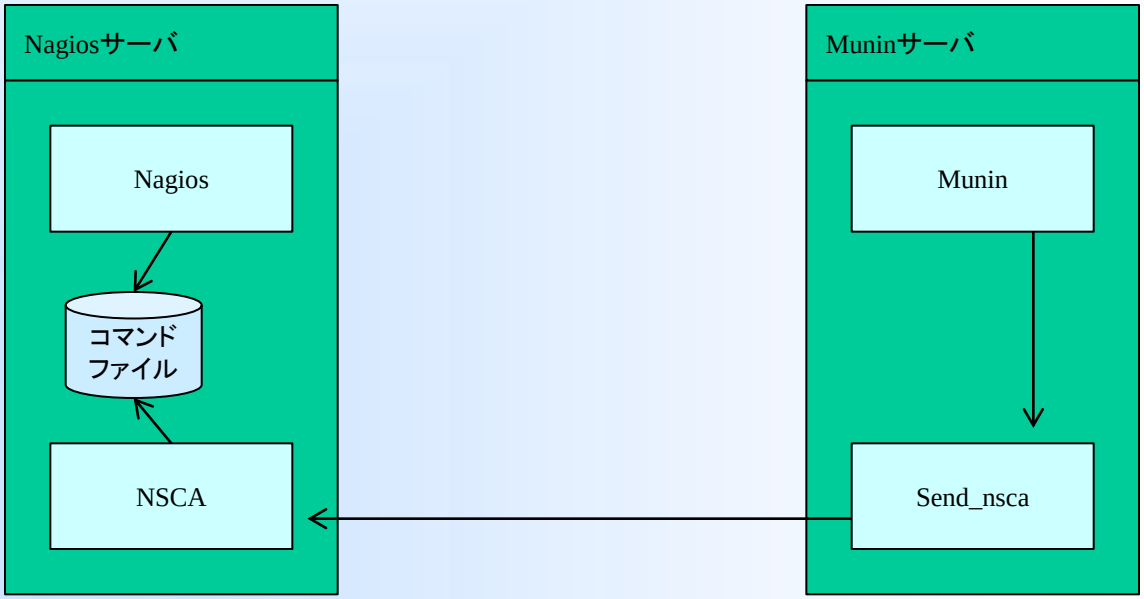
- Muninの通知は機能があまりない。
  - だって標準ではmailコマンドに渡しているだけ

```
contact.mail.command        mail -s "Munin ${var:group}::${var:host}" kanshi@example.jp
```

  - もっと細かい制御をしたいなら、プログラムに渡すべき。
    - 通知プログラムは自作してください。
- 通知機能をNagiosにおまかせする事が可能。
  - XX回連続して閾値を超えたら発報、とかって事が出来る。

Nagiosとの連携イメージ図

- 閾値を超えた場合、muninはSend_nscaを実行
- Send_nscaはNSCAに接続
- NSCAは監視結果をファイルに保存
- Nagiosはファイルを定期的にチェック



NSCA

- NSCA(Nagios Service Check Acceptor)
- Send_nsca(NSCAの監視エージェント)

Send_nscaインストール

```
# cd /usr/local/src
# wget http://prdownloads.sourceforge.net/sourceforge/nagios/nsca-2.7.2.tar.gz
# tar zxvf nsca-2.7.2.tar.gz
# cd nsca-2.7.2
# ./configure
# make send_nsca
# cp src/send_nsca /usr/bin/
# cp sample-config/send_nsca.cfg /etc/
# chown munin:munin /etc/send_nsca.cfg
# vi /etc/send_nsca.cfg
```

```
password=hogehoge
encryption_method=2
```

munin.confの修正

```
contact.nagios.command /usr/bin/send_nsca nagios.example.com -c /etc/send_nsca.cfg

[GroupA;]
contacts        nagios
[GroupB;]
contacts        nagios
```

NSCAのインストール

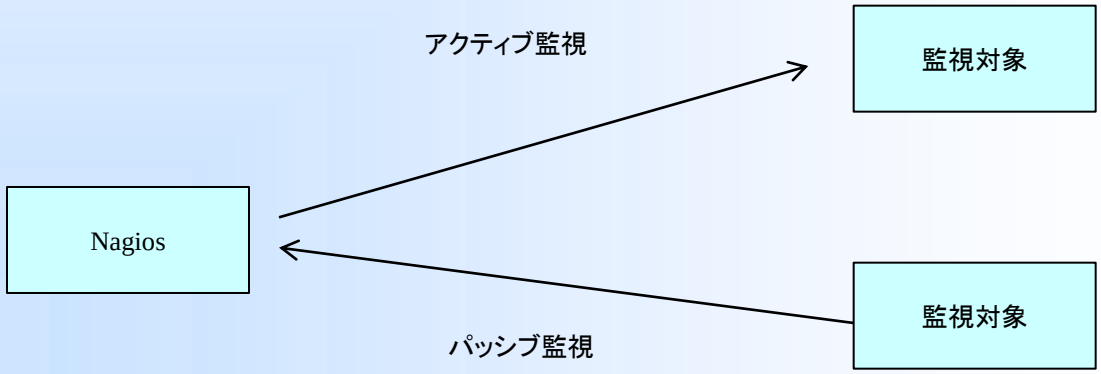
```
# wget http://prdownloads.sourceforge.net/sourceforge/nagios/nsca-2.7.2.tar.gz
# tar zxvf nsca-2.7.2.tar.gz
# cd nsca-2.7.2
# ./configure
# make all
# cp src/nsca /usr/local/nagios/bin/
# cp sample-config/nsca.cfg /usr/local/nagios/etc/
# cp init-script /etc/init.d/nsca
# chmod 755 /etc/init.d/nsca
# vi /usr/local/nagios/etc/nsca.cfg
```

```
server_port=5667
nsca_user=nagios
nsca_group=nagios
command_file=/usr/local/nagios/var/rw/nagios.cmd
password=hogehoge
encryption_method=2
```

```
# chkconfig --add nsca
# chkconfig --list nsca
# /etc/init.d/nsca start
```

Nagios側の設定

- Nagiosは本来、Nagios側から監視対象にアクセスして監視を行っている（アクティブ監視）
- Muninと連携する場合、NSCAから監視結果が送信されてくる（パッシブ監視）



command.conf

```
define command {
        command_name                    munin-check-dummy
        command_line                    $USER1$/check_dummy $ARG1$ $ARG2$
        register                        1
}
```

services.cfg

```
define service {
        host_name                       sv01.example.jp
        service_description             Filesystem_usage
        check_command                   munin-check-dummy!0!OK
        initial_state                   o
        max_check_attempts              4
        active_checks_enabled           1
        passive_checks_enabled          1
        check_period                    none
        obsess_over_service             1
        check_freshness                 1
        event_handler_enabled           1
        flap_detection_enabled          1
        flap_detection_options          o,w,u,c
        process_perf_data               1
        retain_status_information       1
        retain_nonstatus_information    1
        register                        1
}
```

/etc/munin/conf.d/*

```
[GroupA;sv01.example.jp]
        address sv01.example.jp
        use_node_name yes
        df.notify_alias Filesystem_usage
```

Nagios連携時の注意点

- 閾値を設定するサービスを、ホスト毎にnagiosに登録しなければならない。
  - Server1　apache
  - Server1　mail
  - Server2　apache
  - Server2　mysql
- Muninに登録
- Nagiosにも登録

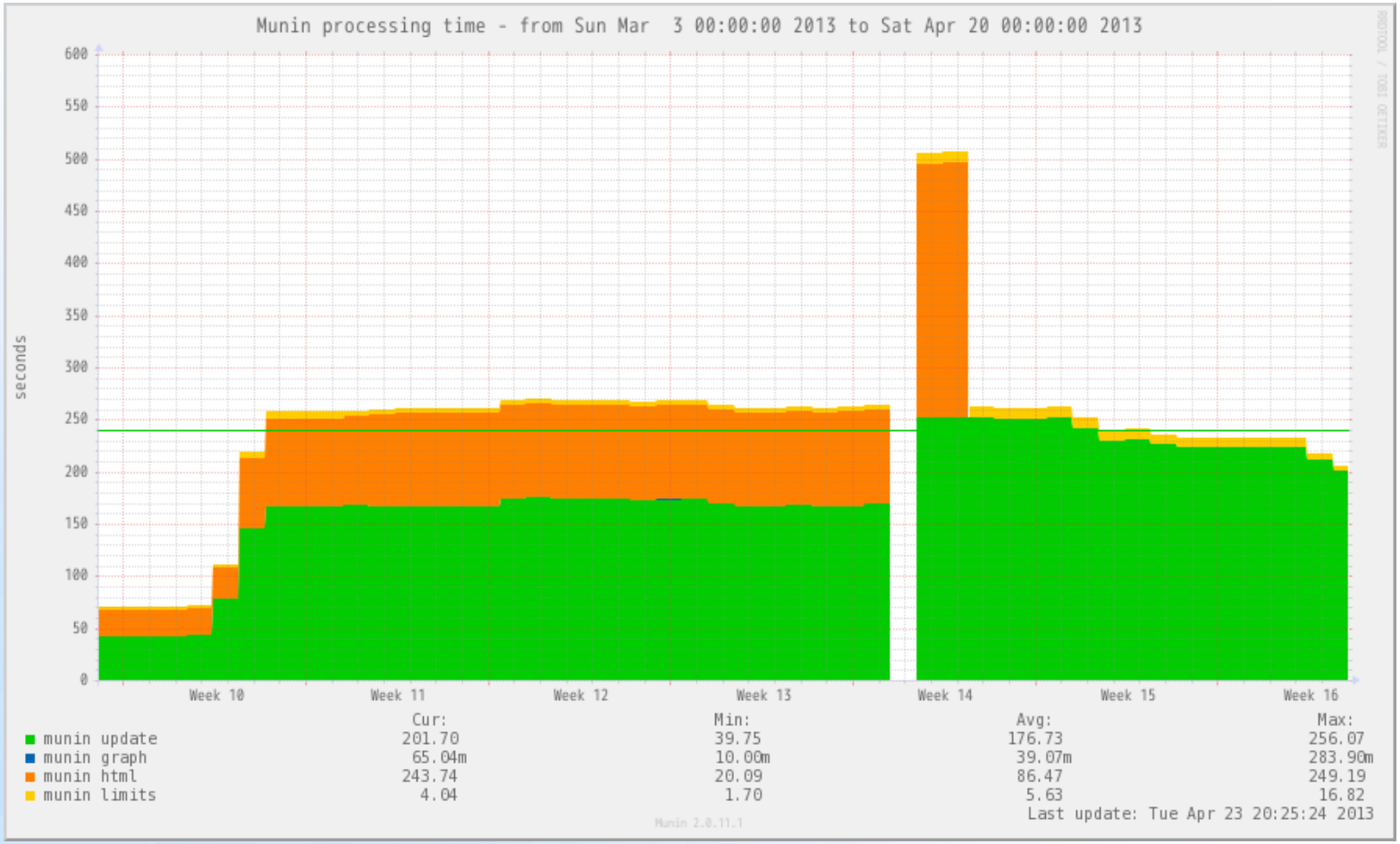
設定する箇所が増えるので結構大変。。。

負荷対策

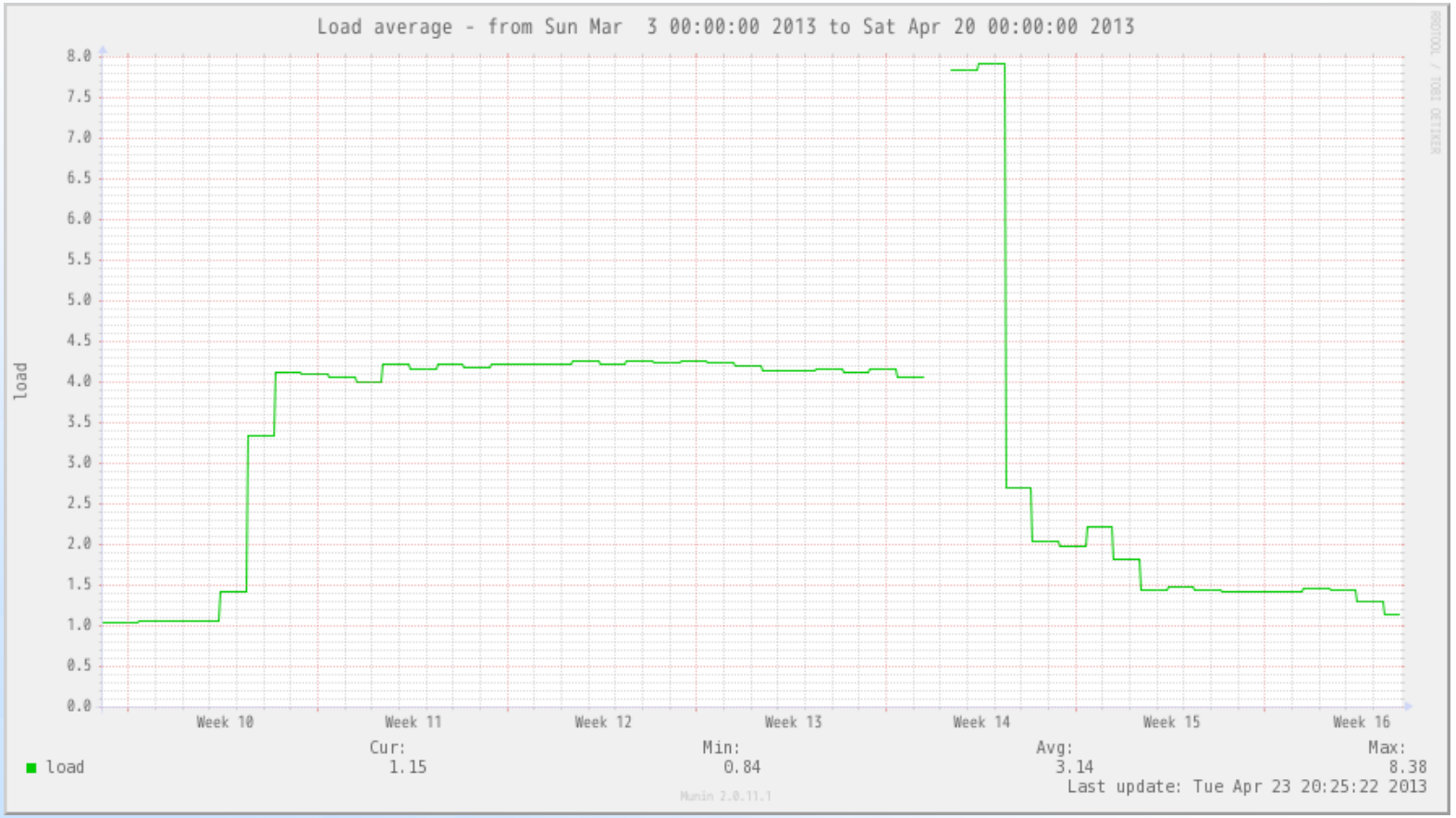
負荷対策(1)

- 監視対象が増えてくると、その分muninを実行しているサーバの負荷も増えます。

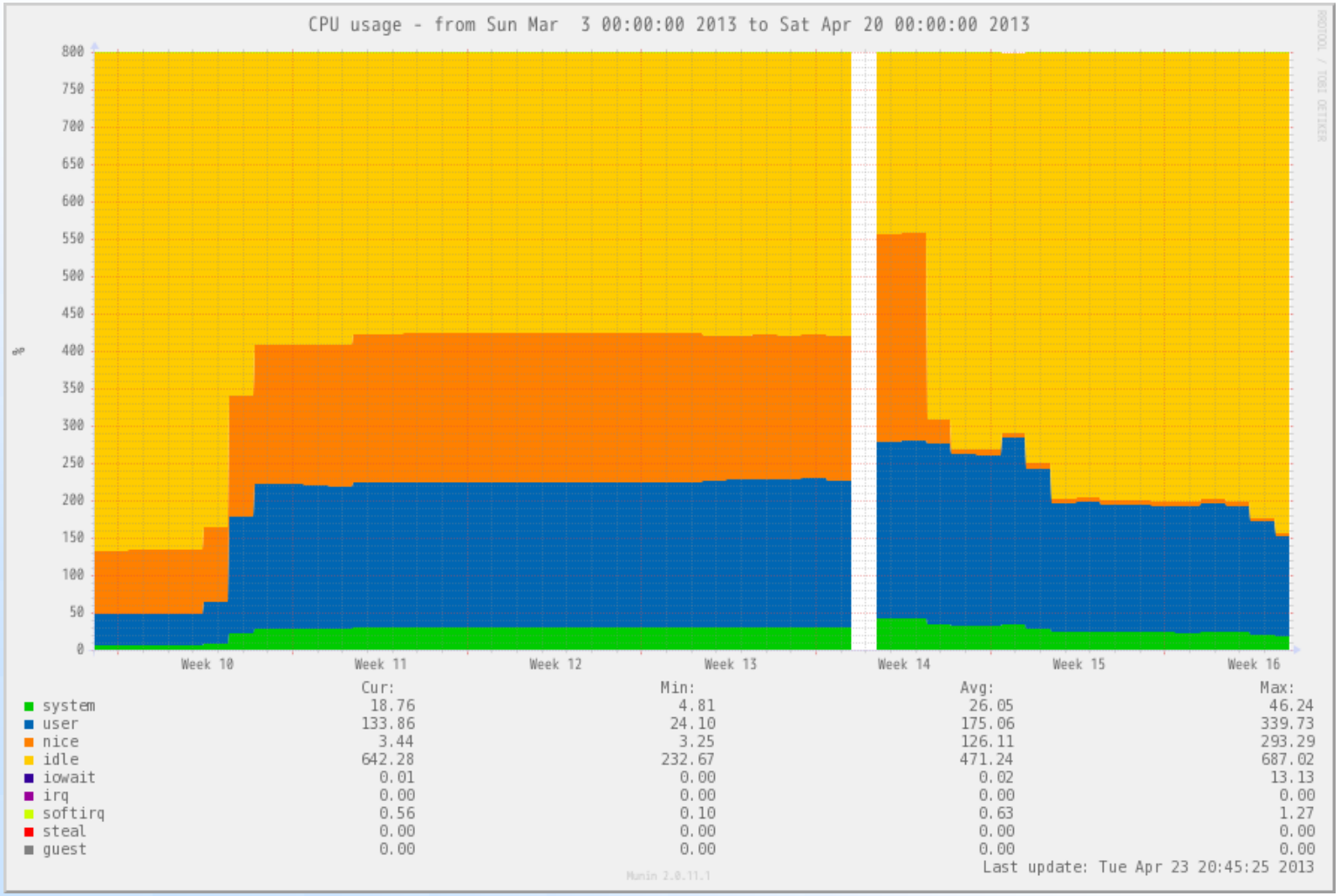
負荷対策(2)



負荷対策(3)



負荷対策(4)



負荷対策(5)

- 当社の監視対象数
  - munin-node　約200
  - snmp　　約130
- Muninのデータ取得から、html作成まで、500秒以上掛かっていました。
- 全対象のデータを取得し始めると負荷が結構な値に。

で、負荷対策!!

- /etc/munin/munin.conf

```
graph_strategy  cron
html_strategy   cron

↓

graph_strategy  cgi
html_strategy   cgi
```

MuninをCGIで実行するように変更

- 通常はcronで実行される度にhtml、画像ファイルが作成されます。
- CGI版にする事により、ウェブでアクセスした時にファイルを作成するようにします。
- メリット
  - 日々のサーバリソースを軽減できます。
  - グラフ参照
- デメリット
  - CRON版に比べ、ページ表示が遅くなります。
  - 感覚的には表示まで1～5秒程度

まとめ

- MRTGに比べ設定が楽
- Munin-nodeがインストールできないネットワーク機器などは今まで通り、MIB情報を取得。
- 標準で使えるプラグインが豊富（約300）。
  - それでも取得できない項目はプラグインを自作。
  - プラグイン自作は作法さえわかれば簡単！
  - 言語はお好みで。
- 閾値の通知はnagiosなど、他プログラムにおまかせ。
- 拡大表示機能で過去のグラフも見返す事が出来る。
- ある程度、監視対象が増えたらCGI版が良い。

・・・ということで皆さんも是非！

さいごに

- Muninのここがイケてない（個人的感想）
  - MRTGと比較してグラフが見づらい
    - 慣れの問題？
  - MIBがわかっていても、それを取得するにはプラグイン作成が必要
  - グラフタイトルの表示名はプラグインで指定
    - Ifの利用者は”description“で確認
  - 通知機能が豊富になればうれしいな～

おしまい

ご清聴ありがとうございました。