NAGIOS・MRTG によるサーバ監視システムの構築

○田上奈緒 A)、瀬川午直 A)、川田良文 A) 、山田一成 A)

A) 共通基盤技術支援室　情報通信技術系

1　はじめに

サーバ・ネットワーク障害によるメール送受信・WEB 閲覧などのサービス停止は、学内に従事する者の研究・通常業務に多大な被害をもたらす。そのため、サーバ・ネットワーク機器の稼動監視を行って迅速に障害検出し対応することが要求される。その対応策の一端として、フリーソフトである NAGIOS・MRTG によるサーバ監視システムを構築し、テスト運用を開始したので報告する。

WEB サーバ・メールサーバなどのサービス稼働状況は、NAGIOS 本体と既存プラグインを導入し、監視内容を設定するだけで監視が可能になる。

また NIC の送受信バイト数など通信負荷は、MRTG で SNMP 通信による情報収集を行うことで時間経過とともに変化する状況の監視ができる。

これらソフトウェアのインストール・設定方法詳細については省略するが、まず上記を導入・設定することにより、基盤センターで運用する一部のサーバの Mail・WEB 稼働状況と NIC 送受信量監視を実現した。次に以下のような要望事項があったため、なるべくそれらを満たすような監視方法の実現を目指した。

要望１．１台のマシンに監視情報を集約したい。

要望２．各サーバの CPU 使用率、ロードアベレージ・フリーメモリ容量、ディスク残容量、セッション数などのリソース状況もチェックしたい。

2　基本システム構成によるサービス稼動監視

監視側サーバのシステムに関する情報は、以下のとおりである。

OS・・・CentOS release 5 (Final)

HTTP・・・Apache/2.2.3

NAGIOS・・・Nagios 3.0.3（プラグイン：nagios-plugins-1.4.12）

MRTG・・・mrtg-2.14.5-2

Nagios についてはソースをダウンロードし、make・install した。Apache，Mrtg については、システムにインストール済みのパッケージをそのまま使用した。

システムの構成概要を図１、図２に示すが、NAGIOS は本体に含まれるデーモンを起動する。別途プラグインもインストールし、各設定ファイルに監視対象とするサーバとチェック内容を書き込み、デーモンを再起動することで監視が始まる。MRTG は、各サーバの監視内容を設定ファイルに記述しておき、cron で mtrg コマンドを定期実行することで、グラフを表示する WEB ページを作成する。

どちらも設定された閾値を越えた際に、通知メールを送信することができる。

この結果、サービス停止・負荷増大による応答の遅延を検出し、また時間の経過とともに増減する NIC 毎の通信量を確認できるようになった。

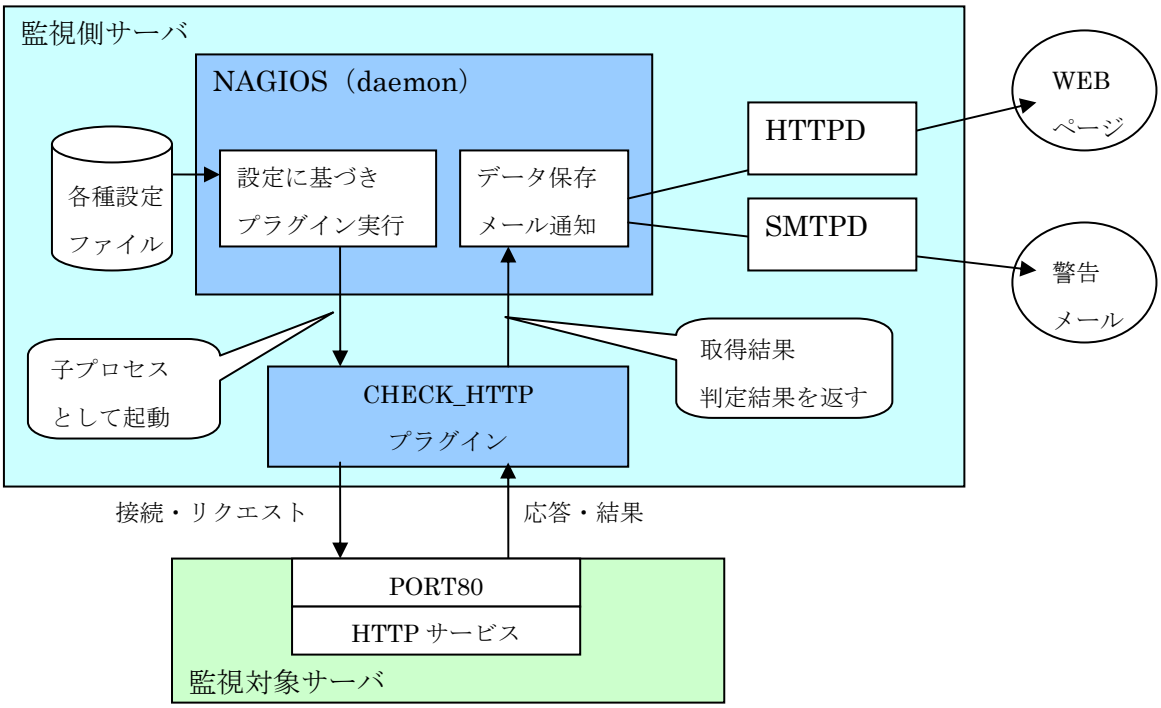

図 1. NAGIOS による監視基本システム構成

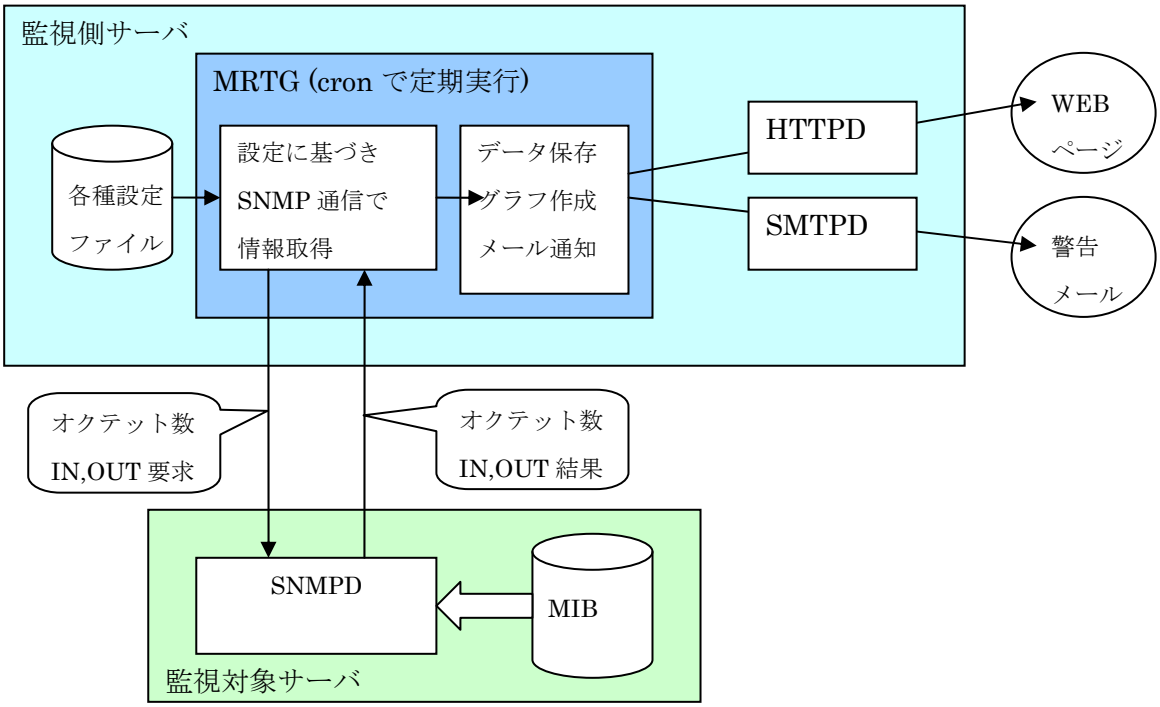

図 2. MRTG による監視基本システム構成

3　リソース情報の監視

さらに、サーバのＣＰＵ負荷が高くなった場合、メモリ・ディスクなどが不足した場合などの障害に備えるため、リソースの監視機能も追加した。

リソースチェック方法として、現時点では以下の３通りが考えられる。

（１）NAGIOS に付属している NRPE デーモン・プラグインを、監視対象側の各サーバにインストール・起動し、監視側の NAGIOS からリソース情報を要求して取得する方法。

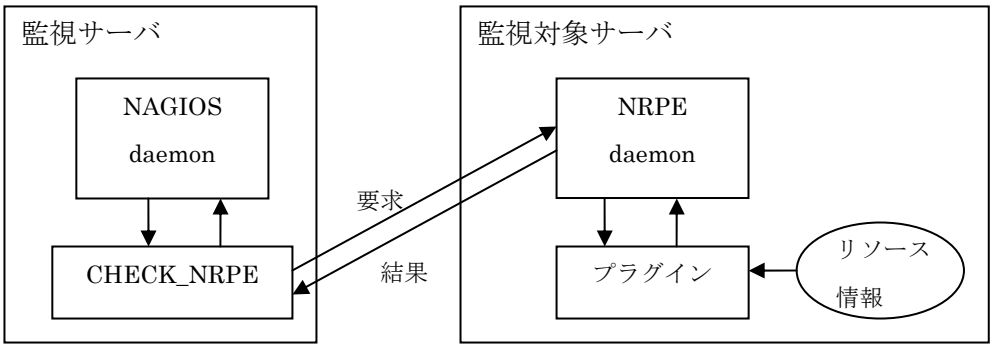

図 3. NRPE デーモンよるリソース情報取得

（２）監視対象側のサーバで SNMP エージェント（snmpd）を起動し、NAGIOS サーバから snmp マネージャコマンド（snmpget,snmpwalk など）を実行することによって、リソース情報を要求し取得する方法。（図解は後述）

（３）NAGIOS サーバから監視対象サーバに SSH 自動ログインが可能な状態にしておき、NAGIOS からリモートコマンドを実行し、結果を受信・解析することでリソース情報を取得する方法。（図解は後述）

この中で（１）の NRPE を使用するのが NAGIOS としては一般的なようだが、今回のテスト運用では、監視対象とするサーバ数が多く、自分が root 権限で作業できないマシンも含まれているため、できるだけ監視対象サーバ側でのインストール、設定変更がない方法を選択したいという要望があった。

そのため、（２）と（３）の方法を試すことにした。

3.1　SNMP コマンドによるリソース監視

SNMP は、サーバ・ネットワーク機器が自身のリソース情報、稼動・負荷状況などを収集して MIB に保管する SNMP エージェントと､エージェントに対して指示をする SNMP マネージャが UDP 通信を行うことで、サーバ・ネットワーク機器の状態管理を行うためのプロトコルである。

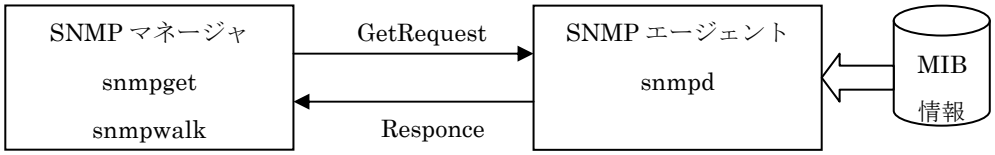

図 4. SNMP マネージャとエージェントの通信

NAGIOS には既存で check_snmp というプラグインが含まれており、内部で snmpget コマンドを実行し、監視対象側の SNMP エージェントから情報を取得する事が可能なようだが、使い方が分かりにくく、自由度が欲しかったため自分でプラグインを作成することにした。プラグインの中では、snmpwalk コマンドを実行して、監視対象サーバの SNMP エージェントから返ってきた結果を読みとり、解析して NAGIOS に渡すという処理を行う。

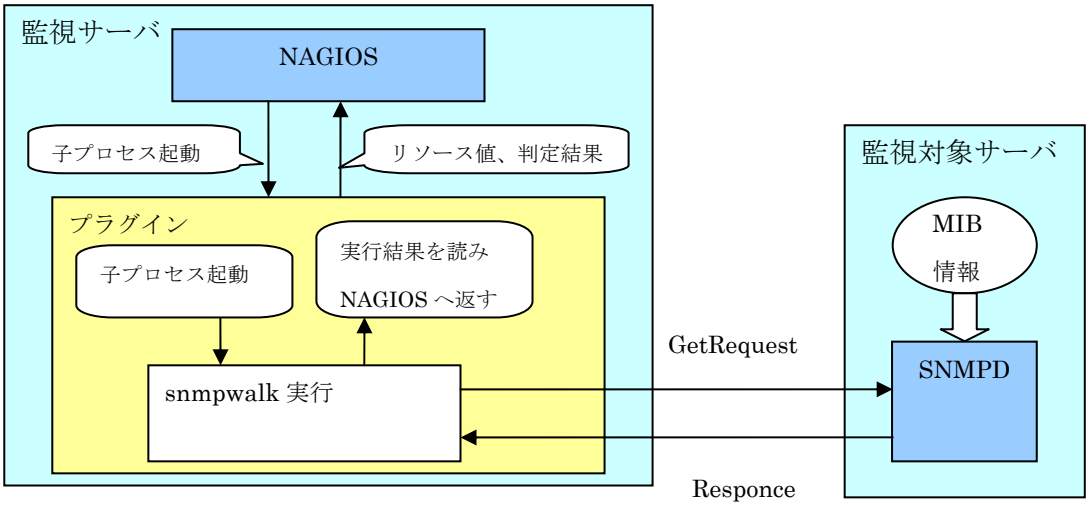

図 5. SNMP 通信によるリソース情報取得

例として監視対象である Linux サーバ（IP：192.168.10.1 とする）のロードアベレージを取得する方法を示す。（SNMP の情報が不必要に参照されるのを防ぐため、コミュニティの設定や iptables などによるフィルタリングが必要になるが、ここでは省略する。）

（１）監視対象サーバの snmpd.conf を設定

「# load average checks」欄内に“#load 12 14 14”と設定されているが、これは「実行プロセス数の最新１分間平均値・最新５分間平均値・最新１５分間平均値が、それぞれの設定値（12,14,14）を超えたら、MIB のエラーフラグを立てる」という意味になる。コメントアウトをはずし、snmpd をリスタートすれば、MIB にロードアベレージの値とエラーフラグの値などが反映されるようになる。

（２）NAGIOS サーバ側から、以下の snmp マネージャコマンドを実行して、監視対象サーバのロードアベレージが取得できることを確認しておく。

“snmpwalk –v 2c –c public 192.168.10.1 laTable”

（３）NAGIOS サーバ側で、作成した“check_loadave”というコマンドを実行する。

“check_loadave -H 192.168.10.1 -C public -P 2c”

結果は標準出力されるため、コマンドラインから実行すると、以下のように表示される。

```
OK -  Load-1(1)=0.020000 Load-5(2)=0.020000 Load-15(3)=0.000000
 Load-1(1)=0.020000 limit=12.000000
 Load-5(2)=0.020000 limit=14.000000
 Load-15(3)=0.000000 limit=14.000000 | Load-1(1)=0.020000 limit=12.000000
 | Load-5(2)=0.020000 limit=14.000000 | Load-15(3)=0.000000 limit=14.000000 |
```

図 6. check_loadave 実行結果

（４）nagios の設定ファイル“commands.cfg”“host.cfg”にて、監視対象サーバ(192.168.10.1)を“check_loadave”でチェックする設定を行い、“/etc/init.d/nagios restart”を実行すると、このプラグインが NAGIOS デーモンによって定期実行されるようになる。リスト１にあるような結果は NAGIOS に読み込まれ、WEB 表示への反映が行われる。１分、５分、１５分平均のいずれかが、監視対象側の snmpd.conf に設定された制限値を超えた場合には EXIT コードを STATUS_CRITICAL にセットしているため、警告メールが送信される。

SNMP を使用してリソース情報を取得する目的で作成したプラグインを表１に示す。

プラグイン名	参照 MIB	監視リソース
check_ifmib	interface	NIC の生死と入出力オクテット数
check_memory	memory	空きメモリ容量（total free サイズ）
check_filemib	fileTable	ファイルサイズ容量（snmpd.conf に、容量を MIB で管理するファイルを設定）
check_sysstat	systemStats	システム状態（CPU アイドル時間）
check_loadave	laTable	最新１分・５分・１５分のロードアベレージ
check_exectbl	extTable	free コマンド実行結果（使用メモリ容量）（snmpd.conf に、実行するコマンドを設定）

表１. SNMP 使用リソース取得プラグイン一覧

3.2　SNMP トラップによるリソース監視

また、NAGIOS サーバ側で SNMP トラップデーモン（snmptrapd）を起動して、監視対象サーバからトラップを通知させ、その情報を NAGIOS まで伝える方法も試した。

この場合 SNMPTRAPD で受信したトラップを NAGIOS まで伝える仕組みが必要となるため、その仲介プログラムを作成して実現した。システムの構成を以下に示す。

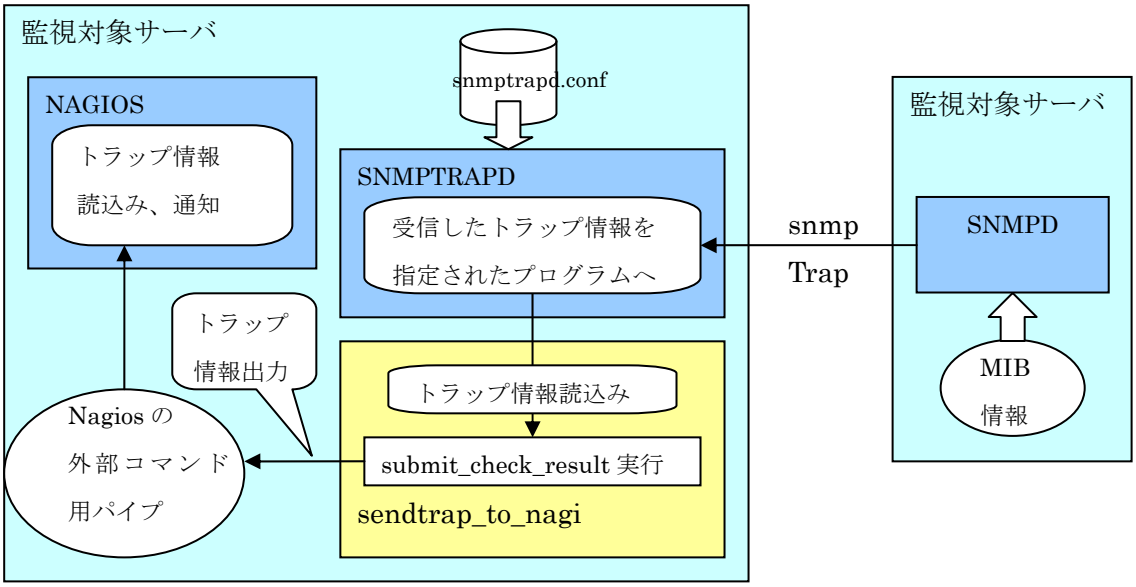

図 7. SNMPTRAP によるリソース情報取得

SNMP マネージャと SNMP エージェントの関係は、マネージャ側から要求し、エージェント側が応答を返すというものだが、SNMP トラップはエージェント側から状態の変化を能動的に通知してくる。そのため監視側で SNMPTRAPD を常駐起動し、トラップを受信する必要がある。

（１）まず監視対象側の snmpd.conf にて、どの状態遷移でトラップを発生させ、どこに送るかを定義する。設定例を図 8 に示す。

```
# トラップの通知先
trap2sink 192.168.1.1 trapprivate     ・・・トラップの送り先となるサーバを指定
（中省略）
# 回線断、復旧のトラップ定義
notificationEvent linkUpTrap     linkUp     ifIndex ifAdminStatus ifOperStatus・・・送りたいトラップと情報
notificationEvent linkDownTrap linkDown ifIndex ifAdminStatus ifOperStatus・・・送りたいトラップと情報
# 60 秒間隔で以下の監視をし、条件を満たしたら上記トラップを発生させる
monitor -r 60 -e linkUpTrap -u _internaluser "Generate linkUp" ifOperStatus != 2
monitor -r 60 -e linkDownTrap -u _internaluser "Generate linkDown" ifOperStatus == 2
```

図 8. snmpd.conf のトラップ通知設定

（２）次に監視側のサーバで SNMPTRAPD の設定をする。

設定ファイル snmptrapd.conf 内に、トラップ毎の送り先となる外部プログラムを指定しておくと SNMPTRAPD が監視対象サーバから受信したトラップを、そのプログラムに渡すことができる。snmptrapd.conf の設定例を図 9 に示す。

この中で特定のトラップに関しては、作成した仲介用プログラムに渡す指定をする。

```
authCommunity log,execute,net trapprivate
traphandle .1.3.6.1.4.1.8072.99999.4 /usr/bin/traptoemail
  –f snmp@xxxx.nagoya-u.ac.jp
  –s mail.xxxx.nagoya-u.ac.jp tanoue@xxxx.nagoya-u.ac.jp          ・・・・①
traphandle .1.3.6.1.4.1.8072.99999.1    /etc/snmp/sendtrap_to_nagi          ・・・・②
traphandle .1.3.6.1.4.1.8072.99999.2    /etc/snmp/sendtrap_to_nagi
```

図 9. snmptrapd.conf のトラップの送り先設定

①は“.1.3.6.1.4.1.8072.99999.4”の受信トラップを/usr/bin/traptoemail というプログラムに渡してメール通知させる例である。

②は“.1.3.6.1.4.1.8072.99999.1”の受信トラップを/etc/snmp/sendtrap_to_nagi というプログラムに渡して処理させる例である。

（３）仲介プログラム sendtrap_to_nagi は、図 10 にあるようなトラップデータを標準入力から読み取ると必要情報を抜き出し、図 11 のように NAGIOS のスクリプトである“submit_check_result”を実行してその情報を渡す。

```
aaaa.xxxx.nagoya-u.ac.jp                          ・・・・このトラップを送信したサーバ
UDP: [192.168.10.1]:32770                          ・・・・送信元 IP、ポート番号
.1.3.6.1.2.1.1.3.0 128:18:44:49.08
.1.3.6.1.6.3.1.1.4.1.0 .1.3.6.1.4.1.8072.99999.2      ・・・・トラップを示す OID
.1.3.6.1.4.1.2021.15.1.2.1 /var/log/maillog          ・・・・監視対象のファイル
.1.3.6.1.4.1.2021.15.1.3.1 101 kB                    ・・・トラップ発生時のサイズ
```

図 10. snmptrapd から受け取ったトラップデータ例

```
/bin/sh -c '/usr/local/nagios/libexec/eventhandlers/submit_check_result
  "aaaa.xxxx.nagoya-u.ac.jp" "TRAP-File-maillog" 2 "FileSize NG
  ( fileName=/var/log/maillog fileSize=101 kB )"'
```

図 11. submit_check_result 実行例

トラップデータの内容は、監視対象側の設定によって付随する情報も異なるため、その内容を sendtrap_to_nagi にあらかじめ定義としておくと該当するトラップに対応することができるようにした。

（4）スクリプト“submit_check_result”は受け取った情報を編集し、NAGIOS のパッシブチェック用パイプに書き込む。

（5）NAGIOS デーモンはパッシブチェック用パイプをポーリングし、情報を読み込んで処理するが、設定ファイルにパッシブチェックの記述をしておくと、WEB 表示、警告メール送信が可能になる。

上記に記述した方法で、Linux サーバのロードアベレージ、残メモリ容量、パケット入出力数、ディスク残容量、ファイルサイズなどの監視を実現した。

ただし、SNMP によるリソース取得は監視対象側の設定が必要になり、サーバの数分同様の作業を行わなければならない。また Linux サーバでは MIB が充実しているが、学内で多く利用されている Solaris サーバ・MacX サーバではリソース情報を管理する MIB は標準では導入されていないようだった。自分自身の勉強不足もあるが SNMP で取得する方法を見つけることができなかったため、Solaris、MacX に関しては次の方法を採用することにした。

3.3　リモートコマンド実行によるリソース監視

本学に導入されているサーバはほとんど OS が UNIX であり、uptime、vmstat などを実行することでリソース情報を確認できる。そのため「NAGIOS サーバから ssh 接続してこれらのコマンドを実行させ、結果を読み取り解析後 NAGIOS に返す」という方法で、リソースの監視を実現した。

NAGIOS サーバでパスフレーズ無しの公開・秘密キーを作成後、監視対象側サーバに公開キーを登録し、クライアント認証させることで SSH ログインし自動実行を可能にする。パスフレーズ無しのキーでログインさせるため、秘密キーが他ユーザから使用できないように厳重保管する必要があるが、監視対象側に施す設定としては最も少なくて済むという利点がある。

この方法でリソース取得するプラグインを以下のとおり作成した。

プラグイン名	実行コマンド	監視リソース
check_vmstat	vmstat	指定したメモリ容量（swap,free など）
check_uptime_la	uptime	ロードアベレージ
check_ssh_rdisk	df -k	ディスク残容量
check_ssh_rfile	ls -l	ファイルサイズ
Check_ssh_netstat	netstat -an	確立セッション数

表 2. リモートコマンド使用リソース取得プラグイン一覧

4　MRTG によるリソース監視

同様に MRTG でも、リモートコマンド実行によりリソースの状態遷移をグラフ化した。mrtg の設定ファイルにリスト 7 のように記述し、この設定ファイルに基づき mrtg コマンドを cron で定期実行させる。

```
Target[work-la]: `/etc/mrtg/get_uptime_la -P /usr/bin/ssh -C 'ssh -l xxxx -i /home/xxxx/.ssh/id_rsa
  192.168.10.1 uptime' -m 5`
MaxBytes[work-la]: 100
Title[work-la]: Load Average
PageTop[work-la]: <H1> Load Average on SolarisTestServer</H1>
YLegend[work-la]: Load Average
ShortLegend[work-la]: num
LegendI[work-la]:1 min ave
```

図 12. MRTG のリモートコマンド実行設定例

この例では、get_uptime_la というコマンド実行を指示しているが、ここで指定したプログラム内にて、１つまたは２つの値を改行で区切り標準出力するだけで、MRTG がその値を拾いグラフを作成する。

監視対象サーバに ssh 接続後、“uptime”“vmstat”“netstat-an | grep ESTABLISHED”などを実行させて結果を読み取り、そこから“ロードアベレージ”“メモリ容量”“特定ポートへのセッション数”を取得して標準出力すれば以下のようなグラフを WEB 表示できる。

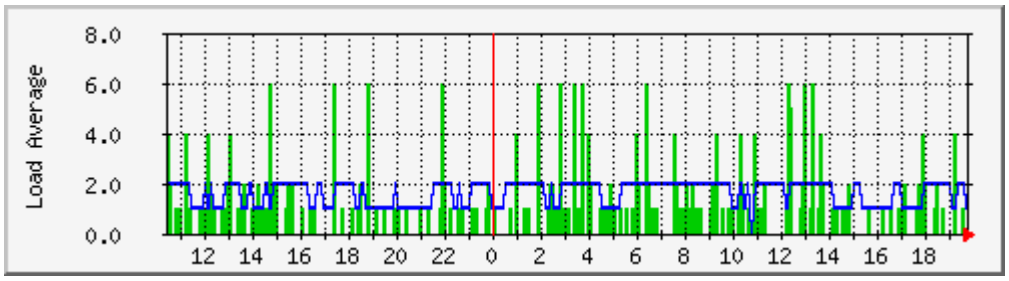

図 13. ロードアベレージ遷移図（緑：１分平均、青：１５分平均）

5　終わりに

今回は NAGIOS と MRTG によるサーバ監視システムの構築を行った。WEB 表示機能については、NAGIOS の方がチェック内容を一覧で確認可能、MRTG では時系列の状態遷移が見られる等それぞれの長所があるため、両者による監視が必要だと思われる。効果的な障害警告を行うための監視項目・閾値調整、より安全で便利な監視対象サーバ側の設定方法などを今後の課題としたい。

参考文献

[1]　“Nagios で作るサーバ／ネットワーク監視システム”　,Software Design,2006 年 12 月号

[2]　“ネットワーク＆システム「見える化」計画”,Software Design,2007 年 10 月号

[3]　Douglas R.Mauro, Kevin J.Schmidt,“入門 SNMP”,オライリージャパン,2007 年 6 月 21 日

[4]　伊藤直也,勝見祐己,田中慎司,ひろせまさあき,安井真伸,横川和哉,“サーバ／インフラを支える技術”,技術評論社,2008 年 9 月 1 日