monitoring environment using Nagios and PerfParse

Nagios と PerfParse を用いた監視環境

Version 1.0

保証免責
本書は記載事項またはそれに関わる事項について、明示的あるいは黙示的な保証はいたしておりません。
したがいまして、これらを原因として発生した損失や損害についての責任を負いません。

表記について

本書では以下の書体を使用しています。

- イタリック文字

  本文中でのコマンド、ファイル名、変数など可変なパラメータ値を表します。

- 等幅文字

  ファイルの内容やコマンドの入出力例に使います。入力の場合にはボールドで表します。

```
$ cd /usr/src/sys/i386/conf
$ ls
GENERIC         Makefile      OLDCARD       SMP
GENERIC.hints   NOTES         PAE           gethints.awk
$
```

- 省略文字

  ファイルの内容やコマンドの入出力例を省略する場合に'...'を使います。

```
$ vi /etc/rc.conf
  ...
sshd_enable="YES"
named_enable="YES"
  ...
$
```

- プロンプト

  一般または、管理権限を持った実行環境をそれぞれ、'$'(ドル)、'#'(シャープ)のプロンプトで表します。

```
$ su
Password: root's passwd
#
```

目次

1. 概要

1.1. 本書について

監視システムの構築に関するリファレンス情報をまとめました。OS のセットアップ方針、利用アプリケーション情報、設定方針など具体的なサンプルを示して把握しやすく努めました。

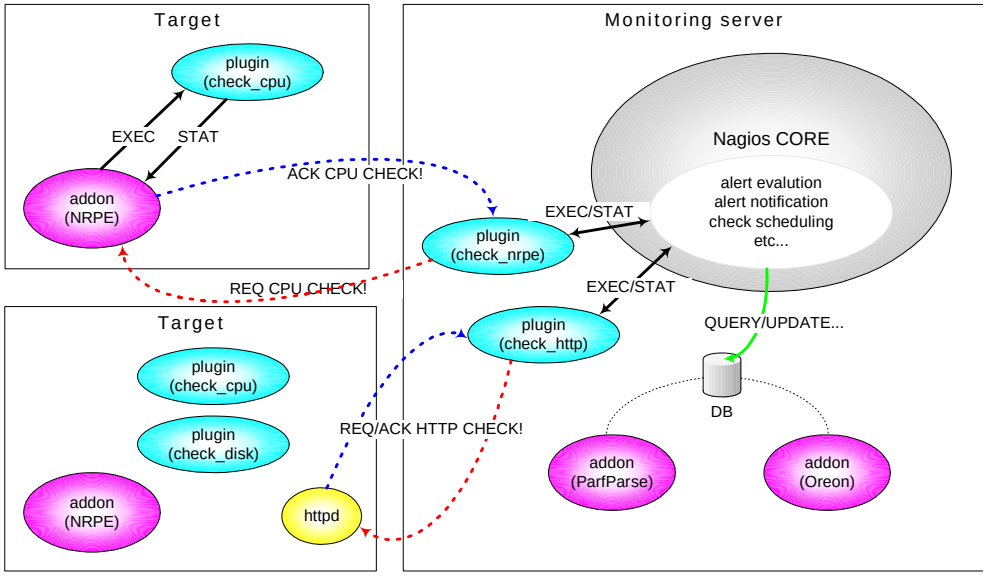
1.2. 対象となる読者

Unix、または Windows の基本的なユーザオペレーションと、管理オペレーションが可能であることを想定しています。また、一般的な Internet プロトコルや、それに基づいて実装されたアプリケーションなどについて知っている必要があります。

本書では、ソフトウェア上の設定に関して、*parameter = value* といった実際の設定情報についてのみ記述します。これらの設定情報についての詳細は関連マニュアルを参照するべきでしょう。

1.3. システム概要

監視システムは、Nagios スイートで構築します。Nagios スイートは、以下のコンポーネントで表すことができます。



- Nagios コアは、インシデントの評価、アラートの通知、監視エントリのスケジュールなどを行います。
- Nagios プラグインは、実際の監視を行い、Nagios コアへステータスを返します。

- Nagios アドオンは、Nagios コアの機能を拡張するものです。この例では、NRPE が図示されていますが、これは監視エージェントに相当します。

Nagios コアが監視ステータスを取得する場合には、いつでもプラグインを実行します。
外部監視で HTTP や SMTP などの可用性をポーリングする場合には、Nagios コアから直接 check_http、check_smtp プラグインなどが起動されます。

内部監視でリソースなどをポーリングする場合には、Nagios コアから check_nrpe プラグインが起動され、このプラグインは Nagios アドオン(NRPE)と通信を行い、監視ステータスを取得します。

1.4. ソフトウェアヴァージョン

利用する監視ソフトウェアのヴァージョンは以下の通りです。

監視ソフトウェア情報

Nagios コア	**Nagios 1.2** 1.x 系ではこれが最新であり、同時に開発も終了していると考えられます。 公式サイト http://nagios.org/download/ よりダウンロード可能です。
Nagios プラグイン	**plugins 1.4** Unix 系 OS では、これが 2005 年 05 月現在最新です。プラグインは常に最新版を利用するようにしてください。 公式サイト http://nagios.org/download/ よりダウンロード可能です。
	NRPE plugins Windows 系 OS では利用するプラグインの集合。このプラグインは上述のようにオフィシャルなものではなく、ヴァージョン管理はされていないようです。 NRPE の接頭詞が付いているのは、おそらく NRPE-NT の配布物に含まれていたプラグインが Windows 用としては初めてまとまったものであったことに起因すると思われます。こちらは常に最新版を利用するべきであると考えられます。 http://www.nagiosexchange.org/NRPE_Plugins.66.0.html からダウンロード可能です。
Nagios アドオン	**NRPE 2.0** このアドオンは、Unix 系 OS で利用します。2.x 系ではこれが 2005 年 05 月現在、最新であり、NRPE は常に最新版を利用するようにしてください。 公式サイト http://www.nagiosexchange.org/NRPE.77.0.html よりダウンロード可能です。
	NRPE-NT 0.8 beta このアドオンは、Windows 系 OS で利用します。0.8 系ではこれが 2005 年 05 月現在、最新であり、NRPE-NT は常に最新版を利用するようにしてください。 http://www.miwi-dv.com/nrpent/ からダウンロード可能です。
	PerfParse 0.105.6 これが 2005 年 05 月現在最新であり、このアドオンは常に最新版を利用するようにしてください。 公式サイト http://perfparse.sourceforge.net/ からダウンロード可能です。
データベース	**MySQL 4.1.11** 4.1.x 系では 2005 年 05 月現在最新であり、MySQL は常に最新版を利用するようにしてください。 PerfParse の動作要件として、4.0.11 以上でなければなりません。また、Nagios は MySQL を利用することができます。

	公式サイト http://dev.mysql.com/downloads/ よりダウンロード可能です。
HTTP サービス	**apache 1.3.33** 1.3 系では 2005 年 05 月現在最新であり、apache は常に最新版を利用するようにしてください。 Nagios と PerfParse は標準で CGI を提供していますので、apache 経由で監視ステータスの閲覧や管理を行うことができます。 公式サイト http://httpd.apache.org/ よりダウンロード可能です。

1.5. ソフトウェア構築のターゲット

利用するソフトウェアのターゲットは以下の通りです。

ターゲット

Nagios コア	監視サーバ上で構築します。
Nagios プラグイン	監視サーバと監視対象機器上で構築します。 監視サーバ上では、Unix 用プラグインを構築します。監視対象機器では、プラットフォームに合わせて構築する必要があります。
Nagios アドオン (NRPE)	監視対象機器上で構築します。 監視対象機器では、プラットフォームに合わせて構築する必要があります。
Nagios アドオン (PerfParse)	監視サーバ上で構築します。
データベース	監視サーバ上で構築します。
HTTP サービス	監視サーバ上で構築します。

1.6. 確認環境

本書を作成する上で、確認用に使用した環境は以下の通りです。

監視サーバ環境

プラットフォーム	Sun Microsystems Ultra10
OS	FreeBSD 5.3-RELEASE-p15/sparc64
ホスト名	yasu

監視対象機器環境

ホスト名	プラットフォーム	OS
tadashi	PC/AT	Microsoft Windows 2000 Server
masaki1	PC/AT	Microsoft Windows 2000 Server
masaki2	PC/AT	Microsoft Windows 2000 Server
cha1	PC/AT	FreeBSD 5.2-RELESE/i386
cha2	PC/AT	Fedora Core 1/i386
wataru	Sun Microsystems Blade100	Solaris8/sparc

1.7. 参考文献

本書を作成する上で、参照した資料は以下のとおりです。

文献

文献	著者	リンク
Nagios FreeBSD 検証資料	kazzs	
Nagios FreeBSD 検証資料 補足	kazzs	
Nagios® Version 1.x Documentation	Ethan Galstad	http://nagios.sourceforge.net/docs/1_0/
Nagios® ヴァージョン 1.0 ドキュメント	佐藤省吾 大槻泰士 岩井成樹	http://nagios.x-trans.jp/Nagios-doc/JAPANESE/
FreeBSD Man Pages	The FreeBSD Project.	http://www.freebsd.org/cgi/man.cgi
MySQL リファレンスマニュアル	MySQL AB	http://dev.mysql.com/doc/mysql/ja/index.html
PerfParse Installation Guide	Garry Cook Ben Clewett Yves Mettier Flo Gleixner	http://perfparse.sourceforge.net/docs.php
FreeBSD ハンドブック	FreeBSD ドキュメンテーションプロジェクト	http://www.freebsd.org/doc/ja_JP.eucJP/books/handbook/index.html
Nagios EXCHANGE		http://www.nagiosexchange.org/

2. 監視サーバの基本構成

2.1 ハードウェア

特にベンダ指定はありませんが、以下の諸元を満たしていることが望ましいでしょう。

諸元

プロセッサ	Intel プロセッサか、それと同等の CPU。(IA) マルチプロセッサである必要はありません。pentium 4 からサポートされたハイパースレッディングでも動作上問題はないと思われます。
メモリ	512MB 程度。
ハードディスク	40GB 程度。 ハードウェア RAID が組めればなおよいでしょう。
NIC	100M - 1 ポート。 ただし、踏み台用途などで用いられるケースもあるので、オンボードで 2 ポート実装した PXE 対応 NIC が望ましいでしょう。
I/O	PCI インタフェースは、最低 1 スロットあった方が望ましいでしょう。 シリアルポートは必須。D-sub 9 ピン - オスのインタフェースが望ましいでしょう。 USB ポートは必須。最低 2 ポートあった方が望ましいでしょう。
外部入出力	VGA インタフェースは必須。 フロッピードライブ、CD-ROM ドライブはなくてもよい。
保守	最低 1 年間のオンサイト対応、および無償修理交換。
形状	1U ラックマウント

2.2. OS

2005 年 05 月現在では FreeBSD を推奨します。なお、本書の構築サンプルでは、FreeBSD をベースに記述しています。OS のインストールタイプは、以下を参考にしてください。

インストールタイプ

OS	コンポーネント	内容
FreeBSD	base	バイナリベース(必須)。
	man	マニュアル。
	src	すべてのソースコード。
Fedora Core	Development Tools	開発ツール群。
	Kernel Development	カーネルのソースコード。
Solaris	Entire Distribution	OEM を除いたすべてのパッケージ。

2.3. ディスクパーティション

40GB のハードディスク 1 本が実装されている場合を例にすると、パーティション分割は以下のように行います。残りは固有の用途でパーティションを作成してください。

ディスクパーティション

マウントポイント	サイズ	用途
/	300MB	ルートパーティション。
swap	物理メモリの 2 倍程度	スワップ領域。
/usr	15GB	アプリケーション用に確保。
/var	15GB	アプリケーションのテンポラリ、および MySQL のインスタンス用に確保。
/home	1GB	メンテナンスアカウント用のホームディレクトリ。

2.4. デーモン

サービス提供に必要と思われる以下のデーモンのみ起動します。例として、起動するデーモンは、以下のとおりになります。

```
$ cat /etc/rc.conf
  ...
sshd_enable="YES"
sendmail_enable="NONE"
mysql_enable="YES"
nagios_enable="YES"
apache2_enable="YES"
snmptrapd_enable="YES"
perfparse_enable="YES"
  ...
```

2.5. 時刻同期

インシデントの検出時刻、およびメール通知に時間的正確性が求められますので、時刻同期を行ってください。最寄りの NTP サーバを指定して ntpupdate で周期的に時刻同期を行います。

```
$ cat /etc/crontab
  ...
0    *    *    *    *    root    ntpdate -s server > /dev/null 2>&1
```

2.6. ロギング

デフォルト設定に準じますが、SNMP トラップのサービスチェックを行う場合はファシリティ設定を追加する必要があります。添付資料 **B. SNMP トラップのサービスチェック**を参照してください。

2.7. ログ･ローテーション

デフォルト設定に準じますが、以下のログ･ローテーション設定を追加しました。一般ユーザでもログの監査が行えるように権限を与えます。

```
$ cat /etc/newsyslog.conf
  ...
/var/log/auth.log                    644 7 100 * J
/var/log/maillog                     644 7 * @T00 J
/var/log/nagios_httpd_access.log     644 7 * @T00 J /var/run/httpd.pid
/var/log/nagios_httpd_error.log      644 7 * @T00 J /var/run/httpd.pid
/var/log/httpd-error.log             644 7 * @T00 J /var/run/httpd.pid
```

```
  /var/log/snmptrapd.log            644 7 * @T00 J
  ...
```

2.8. シリアル接続

シリアルポートへ接続ができるように設定します。

```
$ cat /etc/ttys
  ...
ttyd0   "/usr/libexec/getty std.9600"  vt100   on   secure
  ...
```

また、ブートシーケンスもシリアルコンソール上で確認できるように、以下の設定も追加しておくとよいでしょう。

```
$ cat /boot.config
-P
```

2.9. 仮想コンソール

基本的にコンソールはシリアル接続で利用します。ただし、これが利用できない場合を考慮して、1 つ仮想コンソールを利用できるようにしてください。

```
$ cat /etc/ttys
  ...
ttyv0  "/usr/libexec/getty Pc"        cons25  on  secure
ttyv1  "/usr/libexec/getty Pc"        cons25  off  secure
ttyv2  "/usr/libexec/getty Pc"        cons25  off  secure
ttyv3  "/usr/libexec/getty Pc"        cons25  off  secure
ttyv4  "/usr/libexec/getty Pc"        cons25  off  secure
ttyv5  "/usr/libexec/getty Pc"        cons25  off  secure
ttyv6  "/usr/libexec/getty Pc"        cons25  off  secure
ttyv7  "/usr/libexec/getty Pc"        cons25  off  secure
  ...
```

2.10. メンテナンス用アクセス

ssh による遠隔ログインのみを許可します。以下のように、sshd_config ファイルの設定を確認してください。OpenSSH_3.8.1p1 では、デフォルトの設定で問題ありません。

基本的にはパスワード認証を拒否し、公開鍵認証によるログインのみ許可します。
接続するクライアントのプロトコルヴァージョン 1 の RSA、プロトコルヴァージョン 2 の RSA、または DSA 鍵ペアとなる公開鍵が本サーバのメンテナンスアカウント~/.ssh/authorized_keys にコピーされていなければなりません。

2.11. メール送信

監視サーバは、インシデント、または自身のステータスに関するメールを送信する必要があります。このために既にインストールされている sendmail を MUA として利用します。

sendmail は 8.12 からセキュリティ関連の見直しがなされ、メールリレーを行う機能(MTA)と、ローカルからのメール送信機能(MSP)が分割されて 2 つのプロセスが動作します。
MUA として動作させるためには、submit.cf ファイルを設定します。

```
$ cat /etc/mail/submit.cf
  ...
D{MTAHost}[1st MX]:[2nd MX]
  ...
```

MTAHost マクロの内容は、最初に *1st MX* へ送信を試み、SMTP セッションを開始できなかった場合には、*2nd MX* へ SMTP セッションを試みます。
一時的なエラーによって、すべての MX と SMTP セッションを開始できない場合には、/var/spool/clientmqueue へキューされます。

キューされたメールが再送されるように、以下の設定を行っておくとよいでしょう。

```
# crontab -l
  ...
MAILTO=administrator's mail addr

*/5 * * * * /usr/sbin/sendmail -Ac -q
```

また、/etc/periodic.conf.local ファイルを作成して、通知メール先を再定義します。

```
$ cat /etc/periodic.conf.local
daily output="administrator's mail addr"
daily_status_security_output="administrator's mail addr"
weekly_output="administrator's mail addr"
monthly_output="administrator's mail addr"
  ...
```

2.12. リゾルバ

メール送信時やプラグインの動作時に、名前解決が必要な場合は以下のように設定します。

```
$ cat /etc/resolv.conf
domain your domain
nameserver primary NS
nameserver secondary NS
```

3. 監視サーバの構築

3.1. 構築手順

最初に、portupgrade と perl がインストールされていなければ、これらをインストールしておきましょう。

```
$ cd /usr/ports/sysutils/portupgrade
# make install clean
  ...
# portupgrade --new lang/perl5.8
  ...
```

次に、ポートを利用してアプリケーションをインストールする場合ですが、依存するソフトウェアヴァージョンの整合性が問題になる場合があります。この問題は、アプリケーションのインストール順序に関係がありますので、以下の順序でインストールを行います。

インストールする順序

順序	アプリケーション	ポート
1	データベース	/usr/ports/databases/mysql41-server
2	Nagios コア/プラグイン	/usr/ports/net-mgmt/nagios12
3	HTTP サーバ	/usr/ports/www/apache
4	PerfParse	なし
5	check_nrpe	/usr/ports/net-mgmt/nrpe2

また、アプリケーションのインストールに際して、関連するポートがある場合には自動的にそれらのポートもインストールされます。

3.2. MySQL の構築

① 以下のようにインストールします。

```
# portupgrade --new databases/mysql41-server
```

② my.cnf を設定するために、サンプルファイルをコピーしておきます。

```
# mkdir /var/db/mysql
# chown mysql:mysql /var/db/mysql
# cd /var/db/mysql
# cp /usr/local/share/my-medium.cnf ./my.cnf
# chown mysql ./my.cnf
```

サンプルは以下の目安で適宜選択してください。監視サーバの用途から考えると medium を推奨します。

実装メモリとインスタンスの関係

サンプルファイル名	説明
my-innodb-heavy-4G.cnf	4G のメモリを実装し、InnoDB テーブルのみ使用するシステム用。
my-huge.cnf	1GB～2GB のメモリを実装しているシステム用。

my-large.cnf	512MB 程度のメモリを実装しているシステム用。
my-medium.cnf	128MB 程度のメモリを実装しているシステム用。
my-small.cnf	64MB 程度のメモリを実装しているシステム用。

サンプルをコピーしたら、設定の確認と変更を行ないます。

my.cnf

パラメータ	内容
default-table-type=InnoDB	テーブルを作成したときのデフォルトタイプを InnoDB にします。InnoDB タイプはトランザクション処理に最適化したテーブルタイプです。
#log-bin #server-id = 1	バイナリログを作成しないようにします。これはレプリケーションを行う場合には必須ですが、現在は複製環境がないため、コメントアウトします。 詳細は、 http://dev.mysql.com/doc/mysql/ja/replication-howto.html を参照してください。
innodb_data_home_dir = /var/db/mysql/ innodb_data_file_path = ibdata1:10M:autoextend innodb_log_group_home_dir = /var/db/mysql/ innodb_log_arch_dir = /var/db/mysql/ innodb_buffer_pool_size = 256M innodb_additional_mem_pool_size = 20M innodb_log_file_size = 5M innodb_log_buffer_size = 8M innodb_flush_log_at_trx_commit = 1 innodb_lock_wait_timeout = 50	InnoDB テーブルを使用する場合には、innodb_* の設定を有効にします。 各パラメータの意味は、名前の通りです。 詳細は、 http://dev.mysql.com/doc/mysql/ja/innodb-start.html を参照してください。

③ rc.conf に以下の設定を追加して、起動スクリプトから起動可能か確認します。

```
# cat /etc/rc.conf
  ...
mysql_enable="YES"
  ...
# /usr/local/etc/rc.d/mysql-server.sh start
Starting mysql.
#
```

④ 初期状態の root ユーザのパスワードを設定しておき、それ以外のパスワードのない匿名ユーザ、root ユーザは削除しておきます。

また、データベースへ接続する Nagios、PerfParse アプリケーション用のユーザを作成します。

```
$ mysql -u root mysql
mysql> set password for root@localhost = password('new_passwd');
  ...
mysql> delete from user where Password = '';
  ...
mysql> grant all privileges on nagios.* to nagios@localhost identified by
'nagios_passwd';
mysql> grant all privileges on perfparse.* to perfparse@localhost ident
ified by 'perfparse_passwd';
  ...
mysql> quit
```

3.3. Nagios コア/プラグインの構築

① make extract を実行すると、Nagios コア用のオプションメニューが表示されますので、MYSQL のみを選択します。base/utils.c を修正する必要がありますので、portupgrade ではなく make のターゲットを細かく指定しながら作業することになります。

base/utils.c の修正は、プラグインから返却される OUTPUT フィールドに 2 バイト文字を含めることを可能とするためです。この修正は、2 バイト文字を出力するプラグイン(ログ監視など)に有効です。

また、Makefile も以下のように修正します。

```
$ cd /usr/ports/net-mgmt/nagios12
# make extract
# make patch
# cp -p work/naigos-1.2/base/utils.c work/nagios-1.2/base/utils.c.orig
# vi -c 'set nu' work/nagios-1.2/base/utils.c
  ...
292  ...  strncat(output_buffer,(macro_output==NULL)?"":clean_macro_cha
rs(macro_output,0),buffer_length-strlen(output_buffer)-1);
  ...
# cp -p Makefile Makefile.orig
# vi -c 'set nu' Makefile
  ...
81   .if defined(WITH_FS_SPEC)
82   CONFIGURE_ARGS+=--enable-embedded-perl ¥
83                   --with-perlcache ¥
84                   --disable-statuswrl ¥
85                   --with-template-extinfo ¥
86                   --with-default-perfdata
87 .endif
  ...
# env WITH_FS_SPEC= make build
```

② make install を実行すると、Nagios プラグイン用のオプションメニューが表示されますので、NETSNMP、MYSQL を選択します。途中で Nagios アプリケーションのユーザ/グループアカウント作成確認を促してきますので y を選択します。

ユーザ ID を管理している場合は、自動生成でなくあらかじめアカウントを作成した方がよいでしょう。

```
# make install
  ...
You need a group "nagios".
Would you like me to create it [y]? y

You need a user "nagios".
Would you like me to create it [y]? y
```

③ Nagios 用のデータベースを作成します。

http://dev.mysql.com/doc/mysql/en/upgrading-from-4-0.html により AUTO_INCREMENT 列にデフォルト値を指定するとエラーが発生してしまいますので SQL スクリプトから DEFAULT '0'を削除するように修正します。

```
$ mysql -u nagios -p
Enter Password: nagios's passwd
mysql> create database nagios;
mysql> quit
```

```
$ cd /usr/ports/net-mgmt/nagios12/work/nagios-1.2/contrib/database
# cp -p create_mysql create_mysql.orig
# vi -c 'set nu' create_mysql
  ...
12    hostcomment_id int(11) NOT NULL auto_increment,
  ...
28    hostdowntime_id int(11) NOT NULL auto_increment,
  ...
165   servicecomment_id int(11) NOT NULL auto_increment,
  ...
182   servicedowntime_id int(11) NOT NULL auto_increment,
  ...
$ mysql -u nagios -p nagios < ./create_mysql
Enter Password: nagios's passwd
```

④ rc.conf に以下の設定を追加して、起動スクリプトから起動できるようにします。

```
$ cat /etc/rc.conf
  ...
nagios_enable="YES"
```

⑤ Nagios の基本設定を確認します。この段階ではまだ十分に動作しませんが、本格的な設定方法は別章 **5. 監視設定の方針**で解説します。

```
# /usr/local/bin/nagios -m
  ...
External Data I/O
-----------------
Object Data:       TEMPLATE
Status Data:       DATABASE (MySQL)
Retention Data:    DATABASE (MySQL)
Comment Data:      DATABASE (MySQL)
Downtime Data:     DATABASE (MySQL)
Performance Data:  DEFAULT

Options
-------
* Embedded Perl compiler (With caching)
#
```

⑥ grouplist アドオンをインストールしておきます。

http://apan.sourceforge.net/download.html からソースをダウンロードします。

```
$ tar xvf grouplist.tar.gz
  ...
# cp /usr/local/share/nagios/index.html /usr/local/share/nagios/index.h
tml.orig
# cd grouplist
# cp index.html side1.html side2.html /usr/local/share/nagios
# cp grouplist.cgi /usr/local/share/nagios/cgi-bin
```

grouplist.cgi を修正する必要があるかもしれません。egrep の正規表現から^(サーカムフレックス)を除きます。

```
# cd /usr/local/share/nagios/cgi-bin
# vi -c 'set nu' grouplist.cgi
  ...
23  if [ -f $CFGFILE ]; then
24    HOSTGROUPS=`egrep -e "hostgroup_name" $CFGFILE|awk '{print $2}'| tr
",," " "`
26    GROUPLIST="$GROUPLIST $HOSTGROUPS"
27  fi
  ...
```

3.4. apache の構築

① 以下のようにインストールします。

```
# portupgrade --new www/apache2
```

② httpd.conf の設定を、以下のように修正します。

```
$ cd /usr/local/etc/apache2
# vi httpd.conf
  ...
# vi Includes/virtualhost.conf
  ...
```

httpd.conf

パラメータ	内容
ServerName empty	
#DocumentRoot "/usr/local/www/data"	
#LoadModule userdir_module libexec/apache2/mod_userdir.so	ユーザ･ホームディレクトリ上のコンテンツを公開しない。
#CustomLog /var/log/httpd-access.log combined	
#ServerTokens Full	
#Alias /icons/ "/usr/local/www/icons/"	デフォルトのコンテンツを公開しない。
#AliasMatch ^/manual(?:/(?:de|en|es|fr|ja|ko|ru))?(/.*)?$ "/usr/local/share/doc/apache2$1"	
#ScriptAlias /cgi-bin/ "/usr/local/www/cgi-bin/"	

virtualhost.conf

パラメータ	内容
NameVirtualHost *IP address*:80 ServerTokens ProductOnly	
<VirtualHost *IP address*:80> ServerAdmin *administrator's mail addr* ServerName *FQDN or URL* DocumentRoot /usr/local/share/nagios ErrorLog /var/log/nagios_httpd_error.log CustomLog /var/log/nagios_httpd_access.log common	
ScriptAlias /nagios/cgi-bin/ /usr/local/share/nagios/cgi-bin/ <Directory /usr/local/nagios/sbin> AllowOverride AuthConfig Options ExecCGI FollowSymLinks order allow,deny allow from all </Directory>	Nagios、PerfParse CGI を動作させる。
Alias /nagios/ /usr/local/share/nagios <Directory /usr/local/share/nagios> AllowOverride AuthConfig Options None order allow,deny allow from all </Directory>	Nagios、PerfParse 静的コンテンツを表示させる。
ErrorDocument 403 http://*FQDN or URL*/	

ErrorDocument 404 http://*FQDN or URL*/ </VirtualHost>	

③ rc.conf に以下の設定を追加して、起動スクリプトから起動可能か確認します。

```
# cat /etc/rc.conf
  ...
apache2_enable="YES"
  ...
# /usr/local/etc/rc.d/apache2.sh
Starting apache2.
#
```

④ /etc/group に以下の設定を追加します。cgi.cfg と perfparse.cfg にはデータベースにアクセスするための認証情報が平文で記述されているため、other パーミッションを 0 にしたいところですが、apache から CGI が動作する際には別のユーザ/グループ権限で起動するため、other パーミッションは読取り権限が必要となります。

other パーミッションを 0 にするために、cgi.cfg と perfparse.cfg が読取れるようにグループ権限を与えます。

```
# cat /etc/group
  ...
nagios:*:gid:www
  ...
```

3.5. PerfParse の構築

① 最初に glib ライブラリをインストールしておきます。

```
# portupgrade --new devel/glib20
```

② PerfParse の ports が存在しないので、ソースコードからコンパイルします。tar ボールを取得しておいてください。cgi/Makefile を修正して make を行ないます。

```
$ tar zxvf perfparse-0.105.6.tar.gz
  ...
$ cd perfparse-0.105.6
$ ./configure --with-imagedir=/usr/local/share/nagios/images ¥
--with-cgidir=/usr/local/share/nagios/cgi-bin ¥
--with-http_image_path=/nagios/images ¥
--sysconfdir=/usr/local/etc/nagios ¥
--localstatedir=/var/spool/perfparse
  ...
$ cp -p cgi/Makefile cgi/Makefile.orig
$ vi -c 'set nu' cgi/Makefile
  ...
168   GLIB_LIBS = -L/usr/local/lib -lglib-2.0 -liconv -lgd
  ...
$ make
$ make install
```

③ PerfParse 用のデータベースを作成します。

```
$ mysql -u perfparse -p
Enter Password: perfparse's passwd
mysql> create database perfparse;
mysql> quit
  ...
$ mysql -u perfparse -p perfparse < perfparse-0.105.6/scripts/mysql_cre
ate.sql
```

```
Enter Password: perfparse's passwd
```

④ rc.conf に以下の設定を追加して、起動スクリプトから起動できるようにします。

```
$ cat /etc/rc.conf
  ...
perfparse_enable="YES"
```

⑤ PerfParse の基本設定を確認します。この段階ではまだ十分に動作しませんが、本格的な設定方法は別章 **5. 監視設定の方針**で解説します。

```
# /usr/local/bin/perfparsed --show-config
Error, can't open configuation file: "//usr/local/etc/nagios/perfparse.cfg
#
```

⑥ PerfParse の配布物にはデーモン起動用のスクリプトが含まれていませんので、以下のように作成してください。

```
$ cat /usr/local/etc/rc.d/perfparse.sh
#!/bin/sh
#
# $FreeBSD: perfparse 2005/05/30 18:30:02 yasu Exp $
#

# PROVIDE: perfparse
# REQUIRE: mysql
# KEYWORD: FreeBSD shutdown

. /etc/rc.subr

name="perfparse"
rcvar=`set_rcvar`
command="/usr/local/bin/perfparsed"
command_args=""
required_files="/usr/local/etc/nagios/perfparse.cfg"
pidfile="/var/spool/perfparse/perfparsed.lock"
perfparse_user=nagios

start_precmd=start_precmd

start_precmd() {
   rm -f /var/spool/nagios/rw/perfdata-service.log
   rm -f $pidfile
}

load_rc_config $name
run_rc_command "$1
```

⑦ Nagios コンソールに PerfParse 用のリンクを記述します。

```
# cat /usr/local/share/nagios/side2.html
  ...
<tr>
<td    width=13><img    src="/nagios/images/greendot.gif"    width="13"
height="14" name="perfparse-dot"></td>
<td       nowrap><a       href="/nagios/cgi-bin/perfparse.cgi?all_bin=1"
target="main" onMouseOver="switchdot('perfparse-dot',1)" onMouseO
ut="switchdot('perfparse-dot',0)" class="NavBarItem">PerfData Graphs</a>
</td>
</tr>
  ...
```

3.6. check_nrpe の構築

NRPE アドオンは、監視対象機器に対して構築するものですが、この配布物に含まれる check_nrpe は監視サーバ側で必要としますので構築する必要があります。

① 以下のようにインストールします。

make extract を実行すると、NRPE アドオン用のオプションメニューが表示されますので、ARGS のみを選択します。SSL は可能な限り選択しないでください。

```
$ cd /usr/ports/net-mgmt/nrpe2
# make extract
# make patch
# cp -p Makefile Makefile.orig
# vi -c 'set nu' Makefile
  ...
24   CONFIGURE_ARGS+= --libexecdir=${PREFIX}/libexec/nagios --bindir=${
REFIX}/sbin --sysconfdir=${PREFIX}/etc/nagios
  ...
45   ${INSTALL_DATA} ${WRKSRC}/nrpe.cfg ${PREFIX}/etc/nagios/nrpe.cfg-
sample
# make build
# make install
```

② check_nrpe2 がインストールされたか確認します。

```
$ ls /usr/local/libexec/nagios/check_nrpe2
/usr/local/libexec/nagios/check_nrpe2
```

③ NRPE アドオンが自動起動しないように、以下の作業を行ってください。

```
# cd /usr/local/etc/rc.d
# mv nrpe.sh nrpe.sh.disable
```

4. 監視エージェントの導入

4.1. 導入概要

監視対象機器に対して、Nagios プラグインと NRPE アドオンを導入します。
導入実績のあるプラットフォームと、その固有のパッケージ情報については以下の通りです。

導入実績のあるプラットフォーム

プラットフォーム	パッケージ情報
Solaris (8)	http://www.nagiosexchange.org/NRPE.77.0.html http://www.nagiosexchange.org/Solaris.50.0.html
Linux (2.2, 2.4, 2.6)	http://dag.wieers.com/packages/nagios-nrpe/
FreeBSD (5-RELEASE)	ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-stable/All/nrpe-1.9_2.tbz
Windows (NT, 2000, 2003)	http://www.nagiosexchange.org/Windows.49.0.html http://www.nagiosexchange.org/NRPE_Plugins.66.0.html

パッケージを利用することで、導入の手間が軽減できますが、プラットフォームごとの取扱いや展開されるバイナリ構成の違い、ヴァージョン不整合などがあり、欠点もあります。
ソースコードをコンパイルした場合では、共通の手順で導入可能になりますが、開発環境が必要であるなど、いくつかの要件も必要です。

ただし、重要視する部分は、NRPE のコンフィグレーションです。この部分はどのプラットフォームでも共通なので、これらが適切に行えれば、どの導入方法を用いても構わないと考えることができます。

したがってこの章では、Unix 系プラットフォームはソースからコンパイルを行い、Windows 系は自己解凍型のバイナリファイルを展開する手法の説明を行います。この手法ではどのようなバイナリが動き、どのように相互作用するのか可視的に確認できます。

この点が理解できれば、状況に応じてパッケージを利用することに問題は発生しないでしょう。

4.2. Unix 系プラットフォームへの導入

① Nagios プラグインをインストールします。いくつかのプラグインは依存ソフトウェアの要因でコンパイルされませんが、無視してください。詳細は配布物に含まれる REQUIREMENTS ファイルを参照してください。

```
$ gzip -dc nagios-plugins-1.4.tar.gz | tar xvf -
$ cd nagios-plugins-1.4
$ ./configure
$ make
# make install
```

② NRPE アドオンをインストールします。

```
$ gzip -dc nrpe-2.0.tar.gz | tar xvf -
$ cd nrpe-2.0
$ ./configure --disable-ssl --enable-command-args
$ make
# cp src/nrpe /usr/local/nagios/bin
```

③ nrpe の実行ユーザアカウント nagios とグループ nagios を作成します。アカウント作成の方法はプラットフォームごとに異なりますので man を参照してください。

④ ブート時、自動的に nrpe がデーモン化されるように起動スクリプトをコピーします。コピー先はプラットフォームごとに異なります。

Solaris では、以下のようにコピーします。

```
# cp init-script.freebsd /etc/init.d/nrpe
# ln /etc/init.d/nrpe /etc/rc2.d/S99nrpe
```

FreeBSD では、以下のようにコピーします。

```
# cp init-script.freebsd /usr/local/etc/rc.d/nrpe.sh
# ln /etc/init.d/nrpe /etc/rc2.d/S99nrpe
```

Linux では、以下のようにコピーします。

```
# cp init-script /etc/rc.d/init.d/nrpe
# /sbin/checkconfig --add nrpe
```

4.3. Windows 系プラットフォームへの導入

あらかじめ配布用一式を lha32 などで自己解凍型圧縮ファイルにして監視対象機器へコピーします。

ファイルを展開する場合には、C:¥nagios フォルダにすべてのファイルをコピーします。

NRPE_NT をサービスとして登録するには、以下のように行います。

```
> cd ¥nagios
> nrpe_nt -n -i
NRPE_NT Service sucessfully installed!
>
```

-n オプションは、SSL による通信を無効化します。

NRPE_NT サービスを起動するには、以下のように行います。

```
> net start nrpe_nt
```

4.4. NRPE の設定

各プラットフォームでの共通の設定内容は以下のように行います。

nrpe.cfg

パラメータ	内容
server_port=5666	リスニングポート。
#server_address=	バインドアドレスを指定します。 通常、ここはコメントアウトしてください。
allowed_hosts=*monitor servers*	アクセス可能な監視サーバの IP アドレスを指定します。 複数ある場合は、カンマ(,)で区切ります。
nrpe_user=nagios nrpe_group=nagios	nrpe の実行ユーザとグループ名。
dont_blame_nrpe=1	監視パラメータを引数で受け取ることを許可します。 閾値などのパラメータは監視サーバに集約させる方針ですので 1 にしてください。
debug=0	
command_timeout=60	nrpe から起動されたプラグインのタイムアウト値。 サーバ側の check_nrpe プラグインのタイムアウトと同値になるように設定してください。
#include=<somefile.cfg> #include_dir=<somedirectory> #include_dir=<someotherdirectory>	設定内容を他のファイルやサブディレクトリに分散させることができます。

Unix 系プラットフォームでの標準的な監視設定は以下のように行います。

nrpe.cfg(共通部分からつづく)

command[check load]=/usr/local/nagios/libexec/ check_load -w $ARG1$ -c $ARG2$	ロードアベレージチェック。 $ARG1$には WARNING 値が渡されます。 $ARG2$には CRITICAL 値が渡されます。
command[check_mem]=/usr/local/nagios/libexec/ check_mem2 -u -w $ARG1$ -c $ARG2$	メモリ使用率チェック。 $ARG1$には WARNING 値が渡されます。 $ARG2$には CRITICAL 値が渡されます。
command[check_disk]=/usr/local/nagios/libexec/ check_disk -w $ARG1$ -c $ARG2$ -p $ARG3$	ディスク使用率チェック。 $ARG1$には WARNING 値が渡されます。 $ARG2$には CRITICAL 値が渡されます。 $ARG3$にはパーティション名が渡されます。
command[check_proc]=/usr/local/nagios/libexec/ check_procs -c $ARG1$ -C $ARG2$	プロセスチェック。 $ARG1$にはプロセス数の最小:最大が渡されます。 $ARG2$にはプロセス名が渡されます。
command[check_log]=/usr/local/nagios/libexec/ check_log3 -l $ARG1$ -s $ARG2$ -p "$ARG3$"	テキストログチェック。 $ARG1$にはログファイル名が渡されます。 $ARG2$にはログファイルのシーク情報を持ったファイル名が渡されます。このファイルは事前に touch コマンドなどで作成しておく必要があり、nagios ユーザが読取り可能な権限を与えておきます。 $ARG3$にはマッチングパターンが渡されます。

Windows 系プラットフォームでの標準的な監視設定は以下のように行います。

nrpe.cfg(共通部分からつづく)

パラメータ	内容
command[check_load]=C:¥nagios¥ cpuload_nrpe_nt3.exe $ARG1$ $ARG2$	ロードアベレージチェック。 $ARG1$には WARNING 値が渡されます。 $ARG2$には CRITICAL 値が渡されます。
command[check_mem]= C:¥nagios¥ memload_nrpe_nt2.exe $ARG1$ $ARG2$	メモリ使用率チェック。 $ARG1$には WARNING 値が渡されます。 $ARG2$には CRITICAL 値が渡されます。
command[check_disk]= C:¥nagios¥ diskspace_nrpe_nt2.exe $ARG1$ $ARG2$ $ARG3$	ディスク使用率チェック。 $ARG1$にはドライブ名が渡されます。 $ARG2$には WARNING 値が渡されます。 $ARG3$には CRITICAL 値が渡されます。
command[check_proc]=C:¥nagios¥ cscript.exe //Nolog //T:30 C:¥nagios¥check_process.wsf "$ARG1"	プロセスチェック。 $ARG1$にはプロセス名が渡されます。
command[check_service]= C:¥nagios¥ service_nrpe_nt.exe "$ARG1"	Windows サービスチェック。 $ARG1$には Windows サービス名が渡されます。
command[check_event]=C:¥nagios¥ Eventlog.exe -e $ARG1$ -t $ARG2 -a	イベントログチェック。 $ARG1$にはイベントのログファイル名が渡されます。 $ARG2$にはイベントの種類が渡されます。
command[check_log]=C:¥nagios¥ cscript.exe //Nolog //T:30 C:¥nagios¥check_process.wsf /L:"$ARG1" /S:"$ARG2$" /P:"$ARG3"	テキストログチェック。 $ARG1$にはログファイル名が渡されます。 $ARG2$にはログファイルのシーク情報を持ったファイル名が渡されます。このファイルは事前に touch コマンドなどで作成しておく必要があり、nagios ユーザが読取り可能な権限を与えておきます。 $ARG3$にはマッチングパターンが渡されます。

NRPE 経由でプラグインを起動する際の閾値情報を$ARG1$、$ARG2$のようなパラメータ渡しで行なう場合にはセキュリティに注意する必要があります。
このパラメータ渡しには、インジェクション攻撃を受ける可能性があり、Nagios コアでもシェルが解釈するメタキャラクタをエスケープする支援も行なってはいますが、パラメータを受け取るプラグインはこれらのパラメータを十分に検査している必要があります。

インターネット経由ではパラメータ渡しによる監視は行なわない方がよいでしょう。しかし、パラメータ渡しは、監視情報の管理を Nagios コア側で一元化できるメリットもあります。一定の安全性が保たれた LAN 内ではパラメータ渡しを使っても問題ないでしょう。

5. 監視設定の方針

5.1 概要

以下の方針に基づいて設定していきます。この章での*.cfg 設定ファイルは一例であることを忘れないでください。

- http://nagios.sourceforge.net/docs/1_0/beginners.html のページを読んで心の準備をしてください。わからないことがあれば、リファレンスマニュアルを読んでください。それは、http://nagios.sourceforge.net/docs/1_0/toc.html にあります。
- 実機上のコンフィグレーションを眺めてください。これらはよいサンプルです。
- Nagios コアの配布物に含まれているサンプルコンフィグレーションファイルを修正するようにカスタマイズします。
- サービスチェックを行う場合には、グループ化を上手に使ってください。顧客単位で? サービス単位で? 機器単位で? その他 etc... 考えがまとまったら、それをディレクトリ、またはファイル単位で分割しながら表現します。
- いつでも設定を変更したならば nagios -v でチェックを行うようにします。

5.2 cgi.cfg

Nagios コアの配布物に含まれる CGI の制御に使用されます。このファイルのパーミッションは other フィールドを 0 にしてください。

cgi.cfg

パラメータ例	内容
nagios_check_command=/usr/local/libexec/nagios/check_nagios_db.pl	Nagios 自身のステータス情報を MySQL に問い合わせます。
default_user_name=nagios authorized_for_system_information=nagios authorized_for_configuration_information=nagios authorized_for_system_commands=nagios authorized_for_all_services=nagios authorized_for_all_hosts=nagios authorized_for_all_service_commands=nagios authorized_for_all_host_commands=nagios	CGI が動作する際の実行権限を設定します。
host_unreachable_sound=hostdown.wav host_down_sound=hostdown.wav service_critical_sound=critical.wav service_warning_sound=warning.wav service_unknown_sound=warning.wav	これらのパラメータを有効にすると、ブラウザ上で警告音を鳴らすことができます。
xsddb_host=localhost xsddb_port=3306 xsddb_database=nagios	ステータスデータは MySQL で管理されます。nagios データベースへアクセスするための認証情報を設定します。**3.2 MySQL の構築** ④を参照してください。

xsddb_username=nagios xsddb_password=*passwd*	
xcddb_host=localhost xcddb_port=3306 xcddb_database=nagios xcddb_username=nagios xcddb_password= *passwd*	コメントデータは MySQL で管理されます。nagios データベースへアクセスするための認証情報を設定します。**3.2 MySQL の構築** ④を参照してください。
xcddb_host=localhost xdddb_port=3306 xdddb_database=nagios xdddb_username=nagios xdddb_password= *passwd*	ダウンタイムデータは MySQL で管理されます。nagios データベースへアクセスするための認証情報を設定します。**3.2 MySQL の構築** ④を参照してください。
xedtemplate_config_file=/usr/local/etc/nagios/hostextinfo.cfg xedtemplate_config_file=/usr/local/etc/nagios/serviceextinfo.cfg	拡張ホスト情報、拡張サービス情報データは MySQL で管理可能ですが、この情報はファイルベースで管理した方が楽なので、ファイル名を設定します。

5.3 checkcommand.cfg

Nagios コアが実行するコマンド定義を追加します。

checkcommand.cfg

パラメータ例	内容

define command {
command_name check_snmptrap
command_line $USER1$/check_log3 -l $ARG1$ -s $ARG2$ -p $ARG3$ $ARG4$
}
	トラップチェックの定義です。
	本書では、サービスチェックの 1 つとして定義します。
	実際にはトラップ内容がファイルにロギングされるため、ログチェックの機能として動作させています。

define command {
command_name check_icmp
command_line $USER1$/check_ping -H $HOSTADDRESS$ -w 3000.0,100% -c 5000.0,100% -p 5
}
	ICMP チェックの定義です。
	Nagios では自動的にホストチェックによる ping 疎通確認が行われますが、本書ではサービスチェックの 1 つとして別途定義します。

define command {
command_name check_nrpe
command_line $USER1$/check_nrpe2 -H $HOSTADDRESS$ -t 40 -c $ARG1$
}
	NRPE と通信するプラグインの定義です。

5.4 contacts.cfg

インシデントの通知先定義を追加します。グループ毎に定義するとメンテナンスが楽です。

contacts.cfg

パラメータ例	内容
define contact { contact_name *groupname* alias *alias_for_groupname* service_notification_period 24x7 host_notification_period 24x7 service_notification_options w,u,c,r host_notification_options d,u,r service_notification_commands *groupname_notify_by_email*	通知先定義で重要なのは、contact_name、service_notification_period 、service_notification_options、email です。ここは間違えないように注意してください。

host_notification_commands host-notify-by-email email *notification mail addr* }	

5.5 contactgroup.cfg

インシデントの通知先グループ定義を追加します。グループ毎に定義するとメンテナンスが楽です。

contactgroup.cfg

パラメータ例	内容

define contactgroup {
contactgroup_name *groupname_contact*
alias *alias_for_groupname_contact*
members *groupname*
}
	この通知先グルーピングは、それほど重要ではないと思いますが、contact.cfg で定義した個々の通知先はいずれかの通知先グループに所属しなければなりません。
	
	本書では、通知先、通知先グループは常に 1 対 1 で定義するシンプルな方法をとっています。

5.6 dependencies.cfg

ホスト、またはサービスの依存性を定義します。ここは空のファイルにしてください。
この cfg ファイルの用途は、例えば以下のような状況を考えてみてください。

> ホスト A に ICMP サービスチェックとプロセスサービスチェックを設定している。もし、ICMP サービスチェックで CRITICAL が発生すれば、プロセスサービスチェックでも CRITICAL が発生するだろう...
> この場合、2 つのアラートメールが送信されるだろうが、そもそも ICMP サービスチェックで問題があれば、プロセスサービスのチェックを行わなければよいのではないか? そうすれば、アラートメールは 1 通で済み、ICMP サービスチェックのトラブルシューティング中に、プロセスサービスチェックのアラートメールで埋もれてしまうことはなくなるだろう...

この状況では、「プロセスサービスチェックは、ICMP サービスチェックに依存している」と定義すれば ICMP サービスチェックでアラートが発生した場合、Nagios コアはプロセスサービスのチェックを行わなくなるでしょう。結果として、アラートメールは ICMP サービスチェックについてのみ受け取ることになり、インシデントの嵐に悩まなくても済むかもしれません。

本書ではすべてのインシデントについて計数するので、この機能は使用しません。

5.7 escalations.cfg

ホスト、またはサービスについて、通知のエスカレーション先を定義します。ここは空のファイルにしてください。この cfg ファイルの用途は、例えば以下のような状況を考えてみてください。

ホスト B に ICMP サービスチェックを 1 分間隔で設定しており、CRITICAL が発生すると OK が発生するまで 60 回は継続して(つまり 1 時間)アラートメールを送信するようになっている。通常アラートメールは 1st エンジニア宛に送信されることになっている。

ある時、ホスト B が障害で 3 時間停止した。体制上の方針により、10 分経過(つまり 9 回目の通知)しても OK が発生しない場合には 2nd エンジニア宛にアラートメールを送信することになっている。さらに、20 分経過(つまり 19 回目)しても OK が発生しない場合にはマネジャ宛にアラートメールを送信しなければならない...

この状況は、自明ですがエスカレーションの概念そのものです。Nagios コアはこのエスカレーション体制をシステム的にサポート可能です。しかし、エスカレーション体制は組織毎に異なると思われますので(例えば直接的な電話など)、本書ではこの機能は利用しません。

5.8 hostextinfo.cfg / serviceextinfo.cfg

ホストやサービスの拡張情報を定義します。通常、ホストの OS 情報や画像などを表示させるために使用します。

hostextinfo | serviceextinfo.cfg

パラメータ例	内容

define hostextinfo{
host_name yasu
notes_url
icon_image freebsd.png
icon_image_alt FreeBSD 5.3-RELEASE-p15
statusmap_image freebsd.png
}
	画像と、OS ヴァージョンのポップアップを表示させます。

5.9 misccommand.cfg

Nagios コアが実行する通知コマンド定義を追加します。また、デフォルトで定義されている process-service-perfdata、process-host-perfdata コマンドは、コメントアウトしてください。これらは、後述する PerfParse 用の設定コマンドで追加定義します。

misccommand.cfg

パラメータ例	内容

define command{
command_name *groupname_notify_by_email*
command_line /usr/local/bin/notice.pl
--notificationtype=$NOTIFICATIONTYPE$ --servicedesc="$SERVICEDESC$"
--hostname=$HOSTNAME$--hostalias="$HOSTALIAS$"
--hostaddress=$HOSTADDRESS$ --servicestate=$SERVICESTATE$ --datetime="$DATETIME$" --output="$OUTPUT$"
--contactemail="$CONTACTEMAIL$"
--customername="*customer_name*"
}
	command_line は実際には 1 行で記述します。

5.10 nagios.cfg

Nagios コアの制御に使用されます。

nagios.cfg

パラメータ例	内容
cfg_dir=/usr/local/etc/nagios/*_groupname*	サブディレクトリにある*.cfg ファイルを再帰的に参照します。サービスチェックのグルーピングをディレクトリ分割することで表現しています。 このサブディレクトリ内に、hosts.cfg、services.cfg ファイルを定義するようにしましょう。
cfg_file=/usr/local/nagios/etc/nagios_perfparse.cfg	PerfParse 用の cfg ファイルを指定します。
process_performance_data=1	パフォーマンスデータを取得します。
check_external_commands=1	CGI からの外部コマンド実行をチェックします。 外部コマンドとは、チェックエントリのリスケジュールやダウンタイム指定などが該当します。
use_syslog=0	syslog を使わず、Nagios コア自身でロギング、ローテーションを行います。
log_initial_states=1	サービスの初期状態を記録します。
service_perfdata_command=process-service-perfdata	パフォーマンスデータを取得するようにします。
max_concurrent_checks=	サービスチェックの最大並列数を指定します。 デフォルトは 0 ですが、この場合には無制限に並列チェックを行います。 ただし、チェックレイテンシが大きい時にはこのパラメータを調整してください。レイテンシは、Nagios コンソール上で Performance Info リンクの Check Latency:で確認できます。レイテンシが 15 秒以上発生しているようであれば、nagios -s nagios.cfg コマンドで出力される Recommend value:をこのパラメータに指定してみてください。
check_for_orphaned_services=1	孤立サービスチェックを有効にします。 なんらかの理由でサービスチェックの結果が Nagios コアに返されず、再スケジューリングされない場合があります。 このパラメータを有効にするとコアはこのようなサービスチェックを走査して、必要であれば再スケジューリングします。 Solaris では max_concurrent_checks を調整しても孤立サービスが発生する場合がありますが、そのような場合にはこのパラメータを有効にしてみるとよいでしょう。

5.11 nagios_perfparse.cfg

Nagios コアから起動される PerfParse のコマンド定義を変更します。

nagios_perfparse.cfg

パラメータ例	内容

define command{
command_name process-service-perfdata
command_line
/usr/local/bin/perfparse_nagios_command.pl
/var/spool//perfparse/perfdata-service.log "$TIMET$"
"$HOSTNAME$" "$SERVIC
EDESC$" "$OUTPUT$" "$SERVICESTATE$"
"$PERFDATA$"
}
	command_line は実際には 1 行で記述します。
	perfdata-service.log のディレクトリ名を変更します。

define command{
command_name process-host-perfdata
command_line
/usr/local/bin/perfparse_nagios_command.pl
/var/spool/perfparse/perfdata-host.log "$TIMET$"
"$HOSTNAME$" "$OUTPU
T$" "$PERFDATA$"
}
	command_line は実際には 1 行で記述します。
	perfdata-host.log のディレクトリ名を変更します。

5.12 perfparse.cfg

PerfParse の制御に使用されます。このファイルのパーミッションは other フィールドを 0 にしてください。

perfparse.cfg

パラメータ例	内容
Service_Log="|/var/spool/perfparse/perfdata-service.log" Service_Log_Position_Mark_Path="/var/spool/perfparse"	Nagios コアから出力されるパフォーマンスデータのファイル名を設定します。
Error_Log="/var/spool/perfparse/log/perfparsed" Drop_File ="/var/spool/perfparse/log/" Output_Log_File="no"	PerfParse のロギング関連ファイルを設定します。
DB_User="perfparse" DB_Name="perfparse" DB_Pass="*passwd*" DB_Host="localhost"	PerfParse はパフォーマンスデータを MySQL で管理します。perfparse データベースへアクセスするための認証情報を設定します。
Dummy_Hostname="*hostname*"	
Storage_Modules_Dir="/usr/local/lib" Storage_Modules_Load="mysql" Storage_Modules_Status_File="/var/spool/perfparse/storage_modules.status"	MySQL へアクセスするためのモジュール情報を設定します。
Daemon_Lock="/var/spool/perfparse/perfparsed.lock" Daemonize = "yes"	PerfParse をデーモンプロセスとして動作させるための設定です。
Periodic_Cleanup="yes" Periodic_Cleanup_Lock="/var/spool/perfparse/periodic_cleanup.lock" Periodic_Cleanup_Hour = "0230"	perfparse データベースのデータをパージするための設定です。

5.13 resource.cfg

Nagios コアで処理されるマクロ定義やデータベースへのアクセス情報を定義します。このファイルのパーミッションは other フィールドを 0 にしてください。

resource.cfg

パラメータ例	内容
$USER1$=/usr/local/libexec/nagios	この値はデフォルトのままでいいはずですが、念のため確認してください。
xsddb_host=localhost xsddb_port=3306 xsddb_database=nagios xsddb_username=nagios xsddb_password=*passwd* xsddb_optimize_data=1 xsddb_optimize_interval=3600	ステータスデータ。
xcddb_host=localhost xcddb_port=3306 xcddb_database=nagios xcddb_username=nagios xcddb_password=*passwd* xcddb_optimize_data=1	コメントデータ。
xdddb_host=localhost xdddb_port=3306 xdddb_database=nagios xdddb_username=nagios xdddb_password=*passwd* xdddb_optimize_data=1	ダウンタイムデータ。
xrddb_host=localhost xrddb_port=3306 xrddb_database=nagios xrddb_username=nagios xrddb_password=*passwd* xrddb_optimize_data=1	リテンションデータ。 リテンションとはホスト、サービスの最新ステータスです。 このデータは、Nagios コアが停止してもデータベースに残ります。

5.14 timeperiods.cfg

Nagios コアで処理される通知、またはサービスチェックの有効な時間帯を定義します。

この定義ファイルは変更する必要はないでしょう。

5.15 hosts.cfg

ホスト定義を行います。基本的に/usr/local/etc/nagios/hosts.cfg は空のままにしておいて、サブディレクトリに*.cfg サフィックスの付いたホスト定義ファイルを作成してください。

楽をするコツは、ほとんど変更する必要のないパラメータをテンプレートにまとめてしまうことです。以下の例を参照してください。

HOSTS_TEMPLATE.cfg

パラメータ例	内容

define host{
name generic-host
notifications_enabled 0
event_handler_enabled 1
flap_detection_enabled 0
process_perf_data 1
retain_status_information 1
retain_nonstatus_information 1
 	
checks_enabled 0
check_command check-host-alive
max_check_attempts 10
notification_interval 120
notification period 24x7
notification_options d,u,r
 	
register 0
}
	本書ではホストチェックは行なわず、すべてサービスチェックとして監視を行う方針ですので、左のようにパラメータ設定しておけばよいでしょう。
	
	check_enabled を 0 にしているのはそのためです。また、register もテンプレートとして利用するために 0 にしておかなければなりません。
	
	ホスト定義では、この generic-host テンプレートを use で継承します。

hosts.cfg

パラメータ例	内容

define host{
use generic-host
 	
host_name cha2
alias cha2-groupware
address 192.168.253.8
parents yasu
}
	parents は監視に関してまったく影響はありません。これは、Status Map の見栄えを良くするためだけです。
	
	ホストを追加するときは、既存のエントリをコピーして host_name、alias、address を変更してやればよいでしょう。楽じゃないですか?

define host{
use generic-host
 	
host_name masami
alias masami-pppoe
address 192.168.253.1
parents yasu
}	

5.16 services.cfg

サービス定義を行います。基本的に/usr/local/etc/nagios/services.cfg は空のままにしておいて、サブディレクトリに*.cfg サフィックスの付いたサービス定義ファイルを作成してください。

楽をするコツは、ホスト定義と同様にほとんど変更する必要のないパラメータをテンプレートにまとめてしまうことです。以下の例を参照してください。

SERVICES_TEMPLATE.cfg

パラメータ例	内容

define service{
name generic-service
active_checks_enabled 1
passive_checks_enabled 1
parallelize_check 1
obsess_over_service 1
check_freshness 0
notifications_enabled 1
event_handler_enabled 1
flap_detection_enabled 1
process_perf_data 1
retain_status_information 1
retain_nonstatus_information 1
 	
notification_interval 0
max_check_attempts 3
normal_check_interval 5
retry_check_interval 1
is_volatile 0
check_period 24x7
contact_groups *groupname_contact*
notification_period 24x7
notification_options w,u,c,r
 	
register 0
}
	本書での方針として、左のようにパラメータ設定しておけばよいでしょう。register はテンプレートとして利用するために 0 にしておかなければなりません。
	サービス定義では、この generic-service テンプレートを use で継承します。
	
	active_checks_enabled を 1 にすることでアクティブチェックを行わせます。
	notifications_enabled を 1 にすることで通知を有効にします。
	notification_interval を 0 にすることで、アラートメールの通知は WARNING/CRITICAL の 1 回だけに限定します。
	is_volatile は基本的に 0 です。これは、「インシデントのオープン/クローズが概念的に存在するサービスである」ということを示します。
	check_period、notification_period は 24x7 です。これはどの時間帯に因らずサービスチェックを行うということを示します。
	notification_options は基本的に w,u,c,r です。これはすべてのステータス変更が発生した場合(WARNING/UNKNOWN/CRITICAL/RECOVER)に通知を行うということです。

services.cfg

パラメータ例	内容

define service{
use generic-service
 	
host_name cha2
service_description icmp: alive
check_command check_icmp
}
	ICMP サービスチェックです。

define service{ use generic-service host_name cha2 service_description load: crit -gt 2.00 max_check_attempts 10	ロードアベレージチェックです。 check_nrpe を使用して、閾値を NRPE へ渡します。 ここで、max_check_attempts が 10 に再定義されています。SERVICES_HOSTS.cfg にも 3 と定義されていますが、この場合オブジェクト指向的に 10

check_command check_nrpe!check_load -a "2,2,2" "2,2,2" }	で再定義された値が有効となります。 10 という値は本書の方針です。

define service{
use generic-service
 	
host_name cha2
service_description mem: crit -gt 95p.c.
check_command check_nrpe!check_mem -a "95" "95"
}
	メモリ使用率チェックです。
	check_nrpe を使用して、閾値を NRPE へ渡します。

define service{
use generic-service
 	
host_name cha2
service_description disk: /db crit free -le 40p.c.
check_command check_nrpe!check_disk -a "40%" "40%"
}
	ディスク使用率チェックです。
	check_nrpe を使用して、パーティション名、閾値を NRPE へ渡します。

define service{
use generic-service
 	
host_name cha2
service_description proc: httpd crit -le 5 and crit -gt 20
check_command check_nrpe!check_proc -a "5:20" "/usr/local/apache/bin/httpd" "httpd"
}
	プロセスチェックです。
	check_nrpe を使用して、プロセス名、プロセス数を NRPE へ渡します。
	この例では、httpd プロセスが 5 未満 20 超過した場合に CRITICAL が発生します。

define service{
use generic-service
 	
host_name cha2
service_description log: /var/log/messages - /ntpdate/
is_volatile 1
max_check_attempts 1
normal_check_interval 5
notification_options w,u,c
check_command check_nrpe!check_log -a "/var/log/messages" "/usr/local/nagios/var/messages.seek" "ntpdate"
}
	テキストログチェックです。
	check_nrpe を使用して、ログファイル名、差分ファイル名、パターンを NRPE へ渡します。
	
	ここでは、is_volatile、max_check_attempts、max_check_interval、notification_options が再定義されています。
	is_volatile を 1 にするとステータスを変化させません。この場合は、パターンにマッチしたログを見つけた場合だけ、アラートが発生するということです。
	
	max_check_interval を 1 にすると監視間隔が 1 分となります。これはログの検出を可能な限り早く行うためです。
	
	max_check_attemps を 1 にしているのは、ログ監視の場合は再試行の必要はないからです。
	
	notification_options に r が欠けているのは、RECOVER の概念がないためです。
	これはイベントログ監視でも同様ですし、トラップ監視の場合にも適用可能なインシデントでしょう。

A. カーネル再構築の例

カーネルの再構築を行う前に、cvsup で最新のセキュリティブランチを追跡してアップデートしておきます。cvsup は、/stand/sysinstall で net カテゴリから cvsup-without-gui-16.1h をインストールしておいてください。

以下の設定ファイルを準備します。

```
# cd /usr/local/etc
# mkdir cvsup
# cd cvsup
# cp /usr/share/examples/cvsup/ports-supfile .
# cp /usr/share/examples/cvsup/stable-supfile .
# touch /etc/make.conf
```

設定内容を以下のように修正します。

ports-supfile ファイル

パラメータ	用途･説明
*default host=cvsup6.jp.FreeBSD.org	最新ポートをダウンロードする CVSup サイトを指定。
*default base=/var/db	ステータスファイルの保存場所を指定。
ports-all	すべてのポートをダウンロードする。

stable-supfile ファイル

パラメータ	用途･説明
*default host=cvsup.jp.FreeBSD.org	最新ソースをダウンロードする CVSup サイトを指定。
*default base=/var/db	ステータスファイルの保存場所を指定。
*default release=cvs tag=RELENG_5_3	セキュリティブランチのタグ名を指定。

make.conf ファイル

パラメータ	用途･説明
SUP_UPDATE=yes	CVSup によるアップデートを行う。
SUP=/usr/local/bin/cvsup	
SUPFLAGS=-g -L 2	make 時のオプション。
SUPHOST=cvsup3.jp.freebsd.org SUPHOST=cvsup4.jp.freebsd.org	CVSup サーバを指定。適宜指定します。
SUPFILE=/usr/local/etc/cvsup/stable-supfile	
PORTSSUPFILE=/usr/local/etc/cvsup/ports-supfile	
KERNCONF=*hostname*	コンフィグレーションファイルを指定。
NOINET6=yes	IPv6 を無効化する。

カーネルパラメータを調整します。オリジナルの/usr/src/sys/i386/conf/GENERIC ファイルからコピーしてファイルを修正していきます。コピー先のファイル名はホスト名を大文字にしたものを指定します。

コンフィグレーションは、小さなハードウェア故障に備えて多数の NIC や USB ドライバ等を有効にしておくとよいでしょう。

GENERIC -> *HOSTNAME* ファイル

パラメータ	用途・説明
ident *HOSTNAME*	GENERIC から *HOSTNAME* に変更。
#options INET6	IPv6 を無効にする。

設定ファイルの準備を終えたら、以下のようにソースツリーの更新を行います。

```
# cd /usr/src
# make update
  ...
```

シングルユーザモードへ移行して、ユーザランドとカーネルの構築を行います。

```
# shutdown now
# make clean
# make buildworld
# make buildkernel
  ...
```

カーネルのインストールを行います。インストール後はリブートし、シングルユーザモードで起動します。

```
# make installkernel
# shutdown -r now
  ...
```

起動とともにユーザランドのインストールを行う場合には、以下の手順で行います。

```
# fsck -p
# mount -u /
# mount -a -t ufs
# swapon -a
# cd /usr/src
# make installworld
  ...
```

リブートします。

```
# shutdown -r now
  ...
```

B. SNMP トラップのサービスチェック

最初に Net-SNMP をインストールしておく必要があります。Net-SNMP 配布物に含まれる snmptrapd を利用してトラップの受信を行います。

```
# portupgrade --new net-mgmt/net-snmp
  ...
```

インストール後、起動スクリプトを以下のように変更します。

```
# cat /usr/local/etc/rc.d/snmptrapd.sh
  ...
snmptrapd_flags=${snmptrapd_flags-"-Ls0 -p /var/run/snmptrapd.pid"}
  ...
```

syslog にファシリティ local0 を追加して、ログファイルを作成します。

```
# cat /etc/syslog
  ...
local0.*                /var/log/snmptrapd.log
  ...
# kill -HUP `cat /var/run/syslog.pid`
# touch /var/log/snmptrapd.log
# chmod 644 /var/log/snmptrapd.log
```

以上の設定で SNMP トラップを受信することが可能です。トラップは監視サーバ自身が受信します。トラップの受信を検出するためには、Nagios コアの設定を行います。

```
# cat /usr/local/etc/nagios/checkcommand.cfg
  ...
define command{
  command_name  check_snmptrap
  command_line  $USER1$/check_log3 -l $ARG1$ -s $ARG2$ -p $ARG3$ $ARG4$
  }
  ...

# cat myself_services.cfg
  ...
define service{
  use  generic-service

  host_name  myself
  service_description  trap:
  is_volatile  1
  normal_check_interval  1
  max_check_attempts  1
  notification_options  w,u,c
  check_command  check_snmptrap!/var/log/snmptrapd.log!/var/spool/nagio
s/snmptrapd.log.seek!'.*'
}
  ...
```

トラップの受信内容をログファイルに出力しますので、実質的にはログ監視の機構を利用しています。

monitoring environment using Nagios and PerfParse

改版履歴

Version 1.0	2005/05/19	新規作成。

製作

Isidore.

本書は 2005 年 5 月現在の情報を元に作成されております。本書に記載されております内容は、許可なく変更されることがあります。