Nagios multi-instance material

Nagios マルチインスタンス検証資料

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. はじめに 1
1.1. 本書について 1
1.2. 前提知識 1
1.3. 検証環境 2
1.4. その他の代替案 2
2. 構築と設定の方法 3
2.1. 構築概要 3
2.2. インストール計画 4
2.3. Nagios コアの構築と設定 5
2.4. PerfParse の構築と設定 8
2.5. データベースのインスタンス作成 10
2.6. apache の設定 11
3. コンフィグレーションの例 12
3.1 cgi.cfg 12
3.2 nagios.cfg 13
3.3 nagios_perfparse.cfg 13
3.4 perfparse.cfg 14
3.5 resource.cfg 15
4. まとめ 16
4.1 グループ別認証 16
4.2 負荷予想 16

1. はじめに

1.1. 本書について

Nagios スイートと関連アドオンでは、監視情報をグループ化する機能があります。
しかし、これらの機能はグループ別にアクセス制御を行なえるわけではなく、各グループの監視情報にアクセスする適切な認証方法は提供されていません。

監視情報のグループ化を行うため、Nagios コアと CGI、PerfParse アドオン、データベースインスタンスをグループ別に複数動作させてみてはどうか(本書の命名の由来でもあります)、というアイデアから始まっています。
ここで apache の基本認証メカニズムにより、ディレクトリ階層のアクセス制限を設定することでグループ別に適切なアクセス認証が行なえるか検証しています。

本書は、これらのマルチインスタンス化手法が適切な方法であるのかを議論するためのたたき台として利用してください。

1.2. 前提知識

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。以下に挙げるドキュメントを参照しておくことを推奨します。

文献

文献	著者	リンク
monitoring environment using Nagios and PerfParse	Isidore	
Nagios® Version 1.x Documentation	Ethan Galstad	http://nagios.sourceforge.net/docs/1_0/
PerfParse Installation Guide	Garry Cook Ben Clewett Yves Mettier Flo Gleixner	http://perfparse.sourceforge.net/docs.php
apache	The Apache Software Foundation	http://httpd.apache.org/docs/2.0/

1.3. 検証環境

検証する上で、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 Ultra5 CPU: UltraSparc-IIi 333.00 MHz メモリ: 256MB ディスク: 8GB NIC: hme0 のみ使用(100baseTX)
OS	FreeBSD 5.3-RELEASE-p15/sparc64
ホスト名	nabira
アカウント	公開鍵認証方式で認証します。

ホストグループ

グループ A	www1
グループ B	cha1, cha2, masaki1, masaki2, masami, mx, tadashi, wataru
グループ C	bluenose, pamir, passat, yasu

これらのグループにアクセス制限をかけ、適切なユーザのみが限定された監視情報を閲覧できる仕組みを検証していきます。

1.4. その他の代替案

今回の検証目的として、ユーザ認証による監視情報の制限付き閲覧が主だったものになりますが、マルチインスタンス化以外の方法では Nagios CGI をラップする CGI を実装することが候補に挙げられます。CGI ラッパー案を選択しなかったのは、**Nagios CGI で提供される全機能を利用するのが困難である**点です。

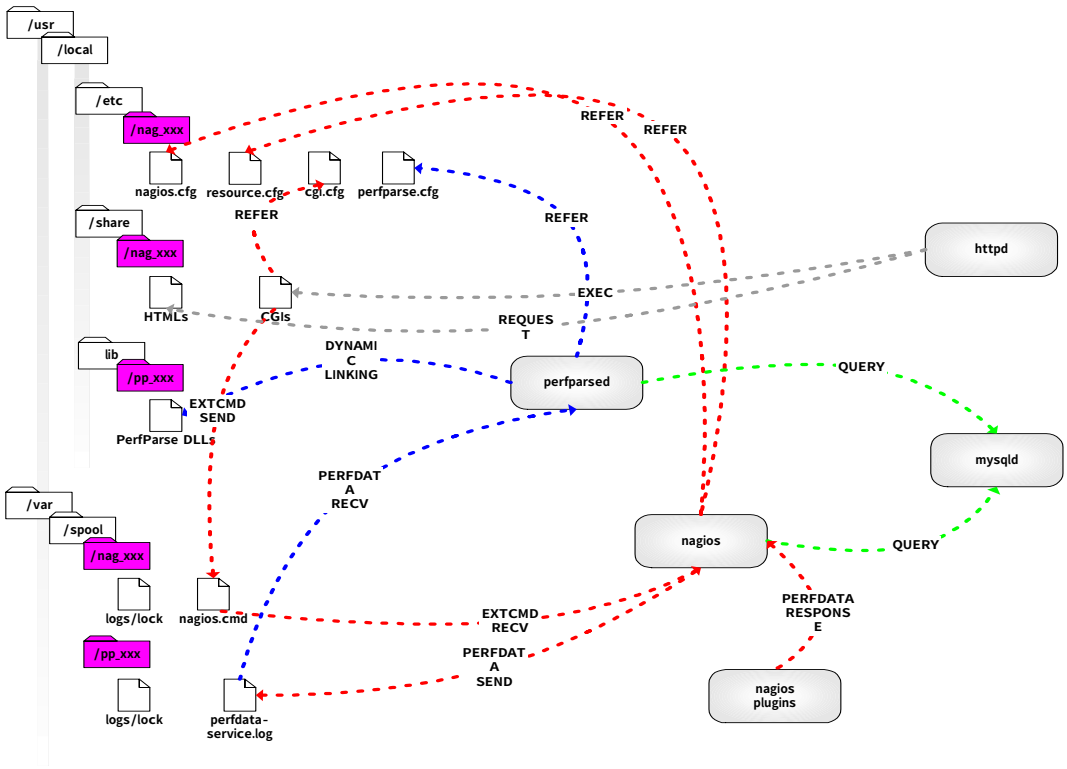
GET メソッドを渡すタイプの CGI であれば、ラッパーは優位性があると考えられますが、そうではないタクティカルモニタ、ログ閲覧、フォームでオブジェクトを選択する形式の標準 CGI では単純にラップできません。

これは、その機能を利用できないということであり、著者としてはこの点について妥協できませんでした。

2. 構築と設定の方法

2.1. 構築概要

グループ別にインスタンス化を行った場合の、ファイルシステムとプロセス相互作用は以下の図のようになります。



上図の nag_xxx、pp_xxx ディレクトリがグループ別に作成されるディレクトリであることを示しており、本書では便宜的に、Nagios 用ディレクトリに nag_プレフィクス、PerfParse 用ディレクトリに pp_プレフィクスを付けた名前を与えています。

このプレフィクス部分は、シングルインスタンス環境では単純に nagios、perfparse と命名されていた部分です。マルチインスタンス環境では、グループ別に、nag_xxx、pp_xxx ディレクトリが作られ、Nagios や PerfParse のデーモンプロセスが起動し、MySQL のテーブルも生成されることになります。

また、apache 経由でのアクセス制限を設けるため、/usr/local/share 以下のディレクトリ階層に nag_xxx ディレクトリを作成します。このディレクトリに対してアクセスするためにはユーザ認証が必要になりますが、認証後でも別のディレクトリ(別のグループ)にはアクセスできないようにします。

2.2. インストール計画

検証用監視サーバは、FreeBSD 上で構築しています。よって、システムの設定変更やアプリケーション等の構築方法は FreeBSD の慣習に沿って記述されています。

もし、FreeBSD をインストールした直後の状態であれば、以下の順序で構築していけばよいでしょう。

構築/設定順序

0	ユーティリティなどのインストール。cvsup、portupgrade、perl など。
1	MySQL、apache。インストールの順序は不同です。
2	Nagios プラグイン、NRPE アドオン(check_nrpe2 用です)。インストールの順序は不同です。
3	Nagios コア。厳密には、ポートによるインストールは行いません。ビルドまで行い、手動で必要ファイルをコピーします。このアプリケーションでは、顧客別グループ毎にこのコピーを行います。
4	PerfParse。ソースによるインストール。このアプリケーションでは、顧客別グループ毎に、configure から make を行います。
5	MySQL へテーブル追加。
6	apache のコンフィグレーション設定。

また、わかりやすいように以下の命名規則を設けます。以降の例では、nag_xxx、pp_xxx 等のメタファを使用しますが、実機上にログインして確認する場合などには、以下のように命名されているということを認識しておいてください。

命名規則

グループ A	Nagios ディレクトリ名は、nag_A と命名します。 PerfParse ディレクトリ名は、pp_A と命名します。 Nagios 用の DB テーブル名は、nag_A と命名します。 PerfParse 用の DB テーブル名は、pp_A と命名します。 apache の BASIC 認証情報は、ユーザ名: A、パスワード: A とします。
グループ B	Nagios ディレクトリ名は、nag_B と命名します。 PerfParse ディレクトリ名は、pp_B と命名します。 Nagios 用の DB テーブル名は、nag_B と命名します。 PerfParse 用の DB テーブル名は、pp_B と命名します。 apache の BASIC 認証情報は、ユーザ名: B、パスワード: B とします。
グループ C	Nagios ディレクトリ名は、nag_C と命名します。 PerfParse ディレクトリ名は、pp_C と命名します。 Nagios 用の DB テーブル名は、nag_C と命名します。 PerfParse 用の DB テーブル名は、pp_C と命名します。 apache の BASIC 認証情報は、ユーザ名: C、パスワード: C とします。

次節から実際の構築や設定方法を記述していきますが、以降の手順はグループ別に行なうということを憶えておいてください。また、やり方が複数ある場合には、最も単純で一般化しやすい方法を選択しています。

2.3. Nagios コアの構築と設定

① 以下のようにインストールします。

初めて 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。

いつものように base/utils.c を修正します。

```
$ 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
  ...
  ...  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
  ...
.if defined(WITH_FS_SPEC)
CONFIGURE_ARGS+=--enable-embedded-perl ¥
                --with-perlcache ¥
                --disable-statuswrl ¥
                --with-template-extinfo ¥
                --with-default-perfdata
.endif
  ...
# env WITH_FS_SPEC= make build
```

② ビルド後に以下のヘッダファイルのマクロを修正して CGI プログラムを再コンパイルします。

```
# cd work/nagios-1.2/common
# vi locations.h
  ...
#define DEFAULT_TEMP_FILE "/var/spool/nag_xxx/tempfile"
#define DEFAULT_STATUS_FILE "/var/spool/nag_xxx /status.log"
#define DEFAULT_LOG_FILE "/var/spool/nag_xxx/nagios.log"
#define DEFAULT_LOG_ARCHIVE_PATH "/var/spool/nag_xxx/archives/"
#define DEFAULT_COMMENT_FILE "/var/spool/nag_xxx/comment.log"
#define DEFAULT_DOWNTIME_FILE "/var/spool/nag_xxx/downtime.log"
#define DEFAULT_RETENTION_FILE "/var/spool/nag_xxx /status.sav"
#define DEFAULT_COMMAND_FILE "/var/spool/nag_xxx/rw/nagios.cmd"
#define DEFAULT_PHYSICAL_HTML_PATH "/usr/local/share/nag_xxx"
#define DEFAULT_URL_HTML_PATH "/nagios"
#define DEFAULT_PHYSICAL_CGIBIN_PATH "/usr/local/share/nag_xxx/cgi-bin"
#define DEFAULT_URL_CGIBIN_PATH "/nagios/cgi-bin"
#define DEFAULT_CGI_CONFIG_FILE "/usr/local/etc/nag_xxx/cgi.cfg"
#define DEFAULT_LOCK_FILE "/var/spool/nag_xxx/nagios.lock"
  ...
# make
```

CGI は実行時に、データベースへのアクセス情報を求めて cgi.cfg を参照しますが、このコンフィグレーションファイルのパス名はコンパイル時に決定されてしまいます。このため、locations.h ヘッダを変更する必要があります。

重要なマクロは DEFAULT_CGI_CONFIG_FILE のみと思われますが、インスタンス固有の情報についても念のため、nag_xxx で変更しておきます。

実行ファイルである nagios は、任意のコンフィグレーションファイル名を指定して動作可能ですので再コンパイルは必要ありません。

③ 必要なディレクトリを作成しておきます。/var/spool に以下のディレクトリを作成します。

```
# cd /var/spool
# mkdir nag_xxx
# chown nagios:nagios nag_xxx
# cd nag_xxx
# mkdir archives rw
# chown nagios:nagios archives
# chown nagios:www rw
```

/usr/local/etc ディレクトリに以下のディレクトリを作成します。

```
# cd /usr/local/etc
# mkdir nag_xxx
  ...
# chown -R nagios:nagios /usr/local/etc/nag_xxx
```

/usr/local/share ディレクトリに以下のディレクトリを作成します。

```
# cd /usr/local/share
# mkdir nag_xxx
# cd nag_xxx
# mkdir cgi-bin
```

④ 再コンパイルしたバイナリ等を手動でコピーします。

```
# cd /usr/ports/net-mgmt/nagios12/work/nagios-1.2
# cp base/nagios /usr/local/bin
# cp p1.pl /usr/local/bin
# chown nagios:nagios /usr/local/bin/nagios /usr/local/bin/p1.pl
# strip /usr/local/bin/nagios
#
# cd html
# cp -r contexthelp docs images index.html main.html media robots.txt ¥
side.html stylesheets
#
# cd ..
# cp -r cgi/*cgi /usr/local/share/nag_xxx/cgi-bin
# strip /usr/local/share/nag_xxx/cgi-bin/*
#
```

⑤ 起動スクリプトを作成します。内容も以下のように修正してください。

```
# cd /usr/ports/net-mgmt/nagios12/work
# cp nagios.sh /usr/local/etc/rc.d/nag_xxx.sh
# vi /usr/local/etc/rc.d/nag_xxx.sh
  ...
required_files=${prefix}/etc/nag_xxx/nagios.cfg
NagiosCfg=${prefix}/etc/nag_xxx/nagios.cfg
NagiosVar=/var/spool/nag_xxx
  ...
```

起動スクリプトを動作させる際には、/etc/rc.conf に nagios_enable="YES"を追加します。

⑥ Nagios コンソール用の side.html の URL パスを変更します。

```
# cd /usr/local/share/nag_xxx
# cat side.html
  ...
<a href="/nag_xxx/cgi-bin/  ...
  ...
```

オリジナルでは、/nagios/cgi-bin/ になっていますが、これをグループ別に /nag_xxx/cgi-bin/と変更します。変更箇所がたくさんありますので、置換コマンドを活用してください。

⑦ *.cfg ファイルの編集を行ないます。「3 章 コンフィグレーションの例」を参照してください。

2.4. PerfParse の構築と設定

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 は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。

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

このケースでは、configure の段階でインストールディレクトリを指定しています。サブディレクトリ個別で再コンパイルする方法もありますが、この方法が一番単純と思われます。make install を実施すると、/usr/local/bin、/usr/local/lib、/usr/local/etc/nag_xxx、/usr/local/share/nag_xxx に必要なファイル群がインストールされます。

③ シェアードライブラリを移動します。

```
# cd /usr/local/lib
# mkdir pp_xxx
# mv libpp_* libnagios_* pp_xxx
```

perfparsed や perfparse.cgi では、実行時にこれらのシェアードライブラリをダイナミックリンクしています。これらのバイナリはデータベースへのアクセス情報についてライブラリ内の関数呼出しにより パラメータを取得します。
ライブラリは perfparse.cfg を参照して呼び出し元へパラメータを返却しますが、perfparse.cfg という名前のコンフィグレーションファイルのパス名はコンパイル時に指定されたオプション等によってハードコーディングされるようです。

このため、シェアードライブラリはグループ毎に固別の実行空間で閉じられている必要があり、ディレクトリを別に分けて perfparsed や perfparse.cgi が動作する際には、LD_LIBRARY_PATH 環境変数を渡して実行するようにします。

perfparse.cgi については、apache のコンフィグレーションで環境変数を渡します。詳細は **2.6. apache の設定**を参照してください。

④ 起動スクリプトを作成します。内容も以下のように修正してください。

```
# vi /usr/local/etc/rc.d/pp_xxx.sh
  ...
# PROVIDE: perfparse
# REQUIRE: mysql
# KEYWORD: FreeBSD shutdown

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/pp_xxx

. /etc/rc.subr

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

start_precmd=start_precmd

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

load_rc_config $name
run_rc_command "$1"
  ...
```

起動スクリプトを動作させる際には、/etc/rc.conf に perfparse_enable="YES"を追加します。

⑤ 必要なディレクトリを作成しておきます。/var/spool に以下のディレクトリを作成します。

```
# cd /var/spool
# mkdir pp_xxx
# mkdir pp_xxx/log
# chown -R nagios:nagios pp_xxx
```

⑥ Nagios コンソールに PerfParse 用のリンクを記述します。

```
# cd /usr/local/share/nag_xxx
# vi side.html
  ...
<tr>
<td  width=13><img  src="images/greendot.gif"  width="13"  height="14"
name="perfparse-dot"></td>
<td      nowrap><a      href="/nag_xxx/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>
  ...
```

⑦ perfparse.cfg ファイルの編集を行ないます。**3 章 コンフィグレーションの例**を参照してください。

2.5. データベースのインスタンス作成

① Nagios 用のアカウントとテーブルを作成します。

```
$ mysql -u root -p mysql
Enter password: root_passwd
mysql> grant all privileges on nag_xxx to nag_xxx@localhost identified
by 'nag_xxx_passwd';
mysql> quit
  ...
$ mysql -u nag_xxx -p
Enter password: nag_xxx_passwd
mysql> create database nag_xxx;
  ...
mysql> use nag_xxx;
  ...
mysql> source /usr/ports/net-mgmt/nagios12/work/nagios-1.2/contrib/data
base/create_mysql;
  ...
```

MySQL 4.1 では、auto_increment 句を含んだ SQL 文に DEFAULT 値を設定できないことに注意してください。create_mysql ファイルから DEFAULT '0' の部分を削除してからテーブルを作成します。

② PerfParse 用のアカウントとテーブルを作成します。

```
$ mysql -u root -p mysql
Enter password: root_passwd
mysql> grant all privileges on pp_xxx to pp_xxx@localhost identified
by 'pp_xxx_passwd';
mysql> quit
  ...
$ mysql -u pp_xxx -p
Enter password: pp_xxx_passwd
mysql> create database pp_xxx;
  ...
mysql> use pp_xxx;
  ...
mysql> source /usr/local/src/perfparse-0.105.6/scripts/mysql_create.sql;
  ...
```

2.6. apache の設定

Nagios や PerfParse は、CGI 経由で web ブラウジング可能ですが、インスタンス毎にアクセスするように仮想ディレクトリを定義します。今回は、ユーザ認証に BASIC 認証を利用します。

httpd.conf の例

パラメータ	内容
ScriptAlias /nag_xxx/cgi-bin/ /usr/local/share/nag_xxx/cgi-bin/ <Directory /usr/local/share/nag_xxx/cgi-bin> SetEnv LD_LIBRARY_PATH /usr/local/lib/pp_xxx AllowOverride AuthConfig Options ExecCGI AuthType Basic AuthName "Restricted nag_xxx zone." AuthUserFile /usr/local/www/.htpasswd Require user *auth_user* order allow,deny allow from all </Directory>	CGI にアクセスした場合の仮想ディレクトリを定義する。 LD_LIBRARY_PATH 環境変数は顧客別グループに沿ったシェアードライブラリを保持しているディレクトリを指定する。 この環境変数は、perfparse.cgi に渡された場合にのみ効力がある。
Alias /nag_infra/ /usr/local/share/nag_xxx/ <Directory /usr/local/share/nag_xxx> AllowOverride AuthConfig AuthType Basic AuthName "Restricted nag_xxx zone." AuthUserFile /usr/local/www/.htpasswd Require user *auth_user* order allow,deny allow from all </Directory>	静的コンテンツにアクセスした場合の仮想ディレクトリを定義する。

htpasswd コマンドで、/usr/local/www/.htpasswd ファイルにユーザを追加しておきます。

3. コンフィグレーションの例

すべてを網羅するわけではありませんが、マルチインスタンス化を構築するにあたって意識するべきコンフィグレーションファイルとそのパラメータ群について列挙しました。

また、すでに構築したインスタンス用のコンフィグをコピーして、機械的にディレクトリ名を置換すると素早く作業できるでしょう。

3.1 cgi.cfg

cgi.cfg

パラメータ	内容
main_config_file=/usr/local/etc/nag_xxx/nagios.cfg	
physical_html_path=/usr/local/share/nag_xxx	
url_html_path=/nag_xxx	
nagios_check_command=/usr/local/etc/nag_xxx/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=nag_xxx authorized_for_system_information= nag_xxx authorized_for_configuration_information=nag_xxx authorized_for_system_commands=nag_xxx authorized_for_all_services=nag_xxx authorized_for_all_hosts=nag_xxx authorized_for_all_service_commands=nag_xxx authorized_for_all_host_commands=nag_xxx	CGI が動作する際の実行権限を設定します。
xsddb_host=localhost xsddb_port=3306 xsddb_database=nag_xxx xsddb_username=nag_xxx xsddb_password=*passwd*	ステータスデータは MySQL で管理されます。
xcddb_host=localhost xcddb_port=3306 xcddb_database=nag_xxx xcddb_username=nag_xxx xcddb_password= *passwd*	コメントデータは MySQL で管理されます。
xcddb_host=localhost xdddb_port=3306 xdddb_database=nag_xxx xdddb_username=nag_xxx xdddb_password= *passwd*	ダウンタイムデータは MySQL で管理されます。
xedtemplate_config_file=/usr/local/etc/nag_xxx/hostextinfo.cfg xedtemplate_config_file=/usr/local/etc/nag_xxx/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。

3.2 nagios.cfg

nagios.cfg

パラメータ	内容
log_file=/var/spool/nag_xxx/nagios.log	
cfg_file=/usr/local/etc/nag_xxx/checkcommands.cfg cfg_file=/usr/local/etc/nag_xxx/misccommands.cfg cfg_file=/usr/local/etc/nag_xxx/contactgroups.cfg cfg_file=/usr/local/etc/nag_xxx/contacts.cfg cfg_file=/usr/local/etc/nag_xxx/dependencies.cfg cfg_file=/usr/local/etc/nag_xxx/escalations.cfg cfg_file=/usr/local/etc/nag_xxx/hostgroups.cfg cfg_file=/usr/local/etc/nag_xxx/hosts.cfg cfg_file=/usr/local/etc/nag_xxx/services.cfg cfg_file=/usr/local/etc/nag_xxx/timeperiods.cfg	
cfg_dir=/usr/local/etc/nag_xxx/_objects	これまではグループ名にちなんだ命名を行ないましたが、マルチインスタンス化では、単にオブジェクトでいいでしょう。
cfg_file=/usr/local/etc/nag_xxx/nagios_perfparse.cfg	PerfParse 用の cfg ファイルを指定します。
resource_file=/usr/local/etc/nag_xxx/resource.cfg	
status_file=/var/spool/nag_xxx/status.log comment_file=/var/spool/nag_xxx/comment.log downtime_file=/var/spool/nag_xxx/downtime.log state_retention_file=/var/spool/nag_xxx/status.sav	データベースを使用している場合は、これらのファイルは不要と思われますが、念のためにインスタンス固有のディレクトリ名に変更しておきます。
command_file=/var/spool/nag_xxx/rw/nagios.cmd	
lock_file=/var/spool/nag_xxx/nagios.lock	
temp_file=/var/spool/nag_xxx/nagios.tmp	
log_archive_path=/var/spool/nag_xxx/archives	
process_performance_data=1	パフォーマンスデータを取得します。
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。

3.3 nagios_perfparse.cfg

nagios_perfparse.cfg

パラメータ	内容
define command{ command_name process-service-perfdata command_line /usr/local/bin/perfparse_nagios_command.pl /var/spool//pp_xxx/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/pp_xxx/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。

3.4 perfparse.cfg

perfparse.cfg

パラメータ	内容
Service_Log="|/var/spool/pp_xxx/perfdata-service.log" Service_Log_Position_Mark_Path="/var/spool/pp_xxx"	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/pp_xxx/log/perfparsed" Drop_File ="/var/spool/pp_xxx/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="pp_xxx" DB_Name="pp_xxx" 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/pp_xxx" Storage_Modules_Load="mysql" Storage_Modules_Status_File="/var/spool/pp_xxx/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/pp_xxx/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/pp_xxx/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。
ServerPort=1976～2000	perfparsed は起動時に ServerPort で指定されたポートをリスニングします。 nagios と perfparsed プロセスが同じホスト上にある場合は、使用されません。 ただし、複数の perfparsed を起動する場合には、このリスニングポートが衝突してしまいますので、衝突しない値を適宜指定しておいてください。 目安のようなものとして、1976～2000 程度の範囲で 1 プロセス毎にインクリメントするような運用を行なってください。

3.5 resource.cfg

resource.cfg

パラメータ	内容
xsddb_host=localhost xsddb_port=3306 xsddb_database=nag_xxx xsddb_username=nag_xxx 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=nag_xxx xcddb_username=nag_xxx xcddb_password=*passwd* xcddb_optimize_data=1	コメントデータ。
xdddb_host=localhost xdddb_port=3306 xdddb_database=nag_xxx xdddb_username=nag_xxx xdddb_password=*passwd* xdddb_optimize_data=1	ダウンタイムデータ。
xrddb_host=localhost xrddb_port=3306 xrddb_database=nag_xxx xrddb_username=nag_xxx 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。

4. まとめ

4.1 グループ別認証

今回は apache の基本認証を利用したが、期待通りの動作をした。
この認証部分は別の認証方法に置換可能と思われる。

4.2 負荷予想

単純にグループ毎に nagios と perfparsed デーモンが起動するので、シングルインスタンス構成より、マルチインスタンス構成の方が、システムリソースを消費すると考えられる。

しかし、マルチインスタンス構成でのグループ別それぞれの監視項目数、およびパフォーマンスデータ数に比例して、各プロセスの CPU 使用率、およびメモリ使用量にばらつきが見られた。
また、シングルインスタンス構成で全体を監視した場合のシステム負荷と、マルチインスタンス構成でのグループ別監視をした場合のシステム負荷はほぼ等しいことを確認した。

シングルインスタンス構成よりもマルチインスタンス構成の方がシステムリソース消費を上回るはずであるが、その差は無視できるものと予想する。

Nagios multi-instance material

改版履歴

Version 1.0　　2005/06/14　　新規作成。

製作

Isidore.

本書は 2005 年 6 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。