Red Hat Satellite 6.5

Red Hat Satellite の監視

Red Hat Satellite 6 からメトリクスを収集する方法

Last Updated: 2019-06-11

Red Hat Satellite 6.5 Red Hat Satellite の監視

Red Hat Satellite 6 からメトリクスを収集する方法

Red Hat Satellite Documentation Team
satellite-doc-list@redhat.com

法律上の通知

概要

本ガイドでは、Red Hat Satellite 6 から分析に使用するメトリクスを収集する方法について説明します。この内容は、Satellite の管理者向けとなります。

目次

第1章 概要

Satellite から収集したメトリクスは、現行の問題のトラブルシュートやキャパシティーのプラニングに役立ちます。本ガイドでは、ライブのメトリクス収集方法と、それらを一定期間アーカイブに保存する方法について説明します。パフォーマンス問題に関するサポートケースを Red Hat に作成する場合は、このアーカイブ化されたデータが貴重な洞察をもたらしてくれます。Red Hat サポートチームがこのアーカイブ化されたデータにアクセスできるようになるのは、お客様がデータをサポートケースにアップロードした場合のみになります。

Satellite からは以下のメトリクスを収集できます:

- Red Hat Enterprise Linux からの基本的な統計数値。システムの負荷やメモリー使用量、出入力操作などが含まれます。
- プロセスの数値。メモリーおよび CPU 使用量などが含まれます。
- Apache HTTP サーバーのアクティビティー統計数値。
- PostgreSQL のアクティビティー統計数値。
- Satellite アプリケーションの統計数値。

Performance Co-Pilot (PCP) を使用して Satellite のメトリクスを収集、アーカイブ化します。

第2章 PERFORMANCE CO-PILOT

Performance Co-Pilot (PCP) は、システムレベルのパフォーマンス測定値を取得、アーカイブ化および分析するためのツールおよびライブラリーのスイートです。PCP は、ライブおよび履歴のメトリクス分析に使用できます。メトリクスは CLI または web UI で取得および提示できます。

2.1. PERFORMANCE METRIC DOMAIN AGENT

Performance Metric Domain Agent (PMDA) は PCP のアドオンで、アプリケーションやサービスのメトリクスにアクセスできるようにするものです。Satellite に関連する全メトリクスを収集するには、Apache HTTP Server および PostgreSQL 向けの PMDA をインストールする必要があります。

第3章 PCP パッケージのインストール

PCP パッケージのインストール方法を説明します。

前提条件

- **/var/log/pcp** ディレクトリーに最小 20 GB の空き領域があること。
  デフォルトの PCP データ保持ポリシーでは、過去 14 日間に収集されたデータのみが保持されます。1 日あたりのデータストレージには通常 100 MB から 500 MB のディスク領域が使用されることが想定されますが、数ギガバイトが使用されることもあります。

- Satellite サーバーが稼働しているベースシステムが、Red Hat Enterprise Linux 7.6. 以降であること。サポートされる PCP パッケージの最小バージョンは PCP バージョン 4.1 です。

手順

1. PCP パッケージを以下のコマンドでインストールします。

```
# yum install pcp \
  pcp-pmda-apache \
  pcp-pmda-postgresql \
  pcp-system-tools
```

2. Performance Metrics Collector デーモンおよび Performance Metrics Logger デーモンを有効にして、開始します。

```
# systemctl enable pmcd pmlogger
# systemctl start pmcd pmlogger
```

3.1. PCP データ収集の設定

以下の手順では、プロセス、Satellite、Apache HTTP サーバーおよび PostgreSQL のメトリクスを PCP が収集するように設定します。

手順

1. PCP が重要な Satellite プロセスについてのデータを収集するように設定します。
   デフォルトでは、PCP は基本的なシステムメトリクスを収集します。このステップを実行すると、以下の Satellite プロセスの詳細なメトリクスが収集されるようになります。

   - Java
   - PostgreSQL
   - MongoDB
   - Dynflow
   - Passenger
   - Pulp
   - Qpid

```
# cat >/var/lib/pcp/pmdas/proc/hotproc.conf <<EOF
#pmdahotproc
Version 1.0

fname == "java" ||
fname ~ /(qdrouterd|qpidd)/ ||
(fname == "postgres" && psargs ~ /-D/) ||
fname == "mongod" ||
fname ~ /^dynflow/ ||
psargs ~ /Passenger RackApp/ ||
fname ~ /^wsgi:pulp/ ||
psargs ~ /celery (beat|worker)/ ||
psargs ~ /pulp_streamer/ ||
psargs ~ /smart-proxy/ ||
psargs ~ /squid.conf/
EOF
```

2. PCP が収集しているプロセスのメトリクスをログ記録するように設定します。

```
# mkdir -p /var/lib/pcp/config/pmlogconf/foreman-hotproc
# cat >/var/lib/pcp/config/pmlogconf/foreman-hotproc/summary << EOF
#pmlogconf-setup 2.0
ident   foreman hotproc metrics
probe   hotproc.control.config != "" ? include : exclude
        hotproc.psinfo.psargs
        hotproc.psinfo.cnswap
        hotproc.psinfo.nswap
        hotproc.psinfo.rss
        hotproc.psinfo.vsize
        hotproc.psinfo.cstime
        hotproc.psinfo.cutime
        hotproc.psinfo.stime
        hotproc.psinfo.utime
        hotproc.io.write_bytes
        hotproc.io.read_bytes
        hotproc.schedstat.cpu_time
        hotproc.fd.count
EOF
```

3. PMDA をモニタリングするプロセスをインストールします。

```
# cd /var/lib/pcp/pmdas/proc
# ./Install
```

4. PCP が Apache HTTP サーバーからメトリクスを収集するように設定します。

   a. Apache HTTP サーバーの拡張ステータスモジュールを有効にします。

```
#cat >/etc/httpd/conf.d/01-status.conf <<EOF
ExtendedStatus On
LoadModule status_module modules/mod_status.so

<Location "/server-status">
PassengerEnabled off
SetHandler server-status
```

```
Order deny,allow
Deny from all
Allow from localhost
</Location>
EOF
```

b. Apache HTTP サーバー PMDA を有効にします。

```
# cd /var/lib/pcp/pmdas/apache
# ./Install
```

c. Satellite インストーラーが拡張ステータスモジュールの設定ファイルを上書きしないようにします。
次の行を **/etc/foreman-installer/custom-hiera.yaml** 設定ファイルに追加します。

```
apache::purge_configs: false
```

5. PCP が PostgreSQL からメトリクスを収集するように設定します。

a. **/var/lib/pcp/pmdas/postgresql** ディレクトリーに切り替えます。

```
# cd /var/lib/pcp/pmdas/postgresql
```

b. インストーラーを実行します。

```
# ./Install
```

c. PCP データベースインターフェイスが PostgreSQL データベースにアクセスできるようにします。
/etc/pcpdbi.conf 設定ファイルに以下の行を挿入します。

```
$database = "dbi:Pg:dbname=foreman;host=localhost";
$username = "foreman";
$password = "6qXfN9m5nii5iEcbz8nuiJBNsyjjdRHA"; (1)
$os_user = "foreman";
```

(1) ここでの **$password** の値は、**/etc/foreman/database.yml** 設定ファイルに保存されています。

d. SELinux を **pcp_pmcd_t** ドメインパーミッションに変更して、PCP の PostgreSQL データベースへのアクセスを許可します。

```
# semanage permissive -a pcp_pmcd_t
```

e. PostgreSQL PMDA が PostgreSQL に接続できることを確認します。
/var/log/pcp/pmcd/postgresql.log ファイルで、接続が確立されていることを確認します。データベース接続が確立されないと、PostgreSQL PMDA はアクティブな状態ですが、メトリクスを提供することはできません。

```
[Tue Aug 14 09:21:06] pmdapostgresql(25056) Info: PostgreSQL connection established
```

/var/log/pcp/pmcd/postgresql.log にエラーがある場合は、pmcd サービスを再起動します。

```
# systemctl restart pmcd
```

6. Satellite で telemetry 機能を有効にします。
Satellite からメトリクスを収集できるようにするには、メトリクスを **statsd** プロトコルで **pcp-mmvstatsd** デーモンに送信する必要があります。メトリクスは集計され、PCP MMV API 経由で入手可能になります。

   a. Foreman Telemetry および **pcp-mmvstatsd** パッケージをインストールします。

   ```
   # yum install foreman-telemetry pcp-mmvstatsd
   ```

   b. **pcp-mmvstatsd** サービスを有効にして起動します。

   ```
   # systemctl enable pcp-mmvstatsd
   # systemctl start pcp-mmvstatsd
   ```

   c. Satellite telemetry 機能を有効にします。
   以下の行を **/etc/foreman/settings.yaml** 設定ファイルに追加します。

   ```
   :telemetry:
     :prefix: 'fm_rails'
     :statsd:
       :enabled: true
       :host: '127.0.0.1:8125'
       :protocol: 'statsd'
     :prometheus:
       :enabled: false
     :logger:
       :enabled: false
       :level: 'INFO'
   ```

7. メトリクスをアーカイブファイルに毎日保存するようにスケジュールします。

```
# cat >/etc/cron.daily/refresh_mmv <<EOF
#!/bin/bash
echo "log mandatory on 1 minute mmv" | /usr/bin/pmlc -P
EOF
# chmod +x /etc/cron.daily/refresh_mmv
```

8. Apache HTTP サーバーおよび PCP を再起動してデータ収集を開始します。

```
# systemctl restart httpd pmcd pmlogger
```

3.2. WEB UI によるメトリクスへのアクセス有効化

以下の手順では、PCP で収集したメトリクスに Web UI でアクセスする方法を説明します。

手順

1. Red Hat Enterprise Linux の **optional** リポジトリーを有効にします。

```
# subscription-manager repos --enable rhel-7-server-optional-rpms
```

2. PCP web API およびアプリケーションをインストールします。

```
# yum install pcp-webapi pcp-webapp-grafana pcp-webapp-vector
```

3. PCP web サービスを起動して、有効にします。

```
# systemctl start pmwebd
# systemctl enable pmwebd
```

4. PCP web サービスへのアクセスを可能にするようにファイアウォールポートを開きます。

```
# firewall-cmd --add-port=44323/tcp
# firewall-cmd --permanent --add-port=44323/tcp
```

3.3. PCP 設定の確認

PCP が正常に設定されたこととサービスがアクティブであることを確認するには、以下のコマンドを実行します。

```
# pcp
```

このコマンドで、アクティブな PCP の設定概要が出力されます。

pcp コマンドの出力例

```
Performance Co-Pilot configuration on satellite.example.com:

 platform: Linux satellite.example.com 3.10.0-862.3.3.el7.x86_64 #1 SMP Wed Jun 13 05:44:23 EDT
2018 x86_64
 hardware: 8 cpus, 4 disks, 1 node, 23380MB RAM
 timezone: AEST-10
 services: pmcd pmwebd
     pmcd: Version 3.12.2-1, 9 agents, 1 client
     pmda: root pmcd proc xfs linux apache mmv postgresql jbd2
 pmlogger: primary logger: /var/log/pcp/pmlogger/satellite.example.com/20180802.00.10
```

この例では、Performance Metrics Collector Daemon (pmcd) と Performance Metrics Web Daemon (pmwebd) の両方のサービスが稼働しています。また、PMDA がメトリクスを収集しています。最後の **pmlogger** がメトリクスを保存しているアーカイブファイルも記載されています。

第4章 PCP メトリクス

メトリクスは、ツリー構造で保存されます。たとえば、ネットワークメトリクスはすべて、**network** というノードに保存されます。各メトリックは単一または複数の値となり、これらはインスタンスと呼ばれます。たとえば、kernel の負荷には、1 分平均、5 分平均、および 15 平均という 3 つのインスタンスがあります。

PCP は各メトリックエントリーについてデータとメタデータの両方を保存します。これにはメトリックの説明、データタイプ、単位、ディメンションなどが含まれます。たとえば、メタデータを使うと、PCP は異なるディメンションで複数のメトリクスを出力できます。

カウンターメトリクスの値は、増加しかしません。たとえば、特定デバイス上でのディスクの書き込み操作のカウントは、増加のみです。カウンターメトリックの値をクエリすると、PCP はこれをデフォルトでレート値に変換します。

CPU やメモリー、kernel、XFS、ディスク、ネットワークなどのシステムメトリクスの他に、以下のメトリクスが設定されます。

メトリクス	説明
hotproc.*	主要 Satellite プロセスの基本的なメトリクス
apache.*	Apache HTTP サーバーのメトリクス
postgresql.*	PostgreSQL の基本的なメトリクス
mmv.fm_rails_*	Satellite のメトリクス

4.1. 利用可能なメトリクスの特定

- PCP で利用可能なメトリクスすべてを一覧表示するには、以下のコマンドを実行します。

```
# pminfo
```

- すべての Satellite メトリクスと説明を一覧表示するには、以下のコマンドを実行します。

```
# foreman-rake telemetry:metrics
```

- アーカイブ化されたメトリクスを一覧表示するには、以下のコマンドを実行します。

```
# less /var/log/pcp/pmlogger/$(hostname)/pmlogger.log
```

- pmlogger デーモンは、データを受け取ると設定に従ってアーカイブ化します。アクティブなアーカイブファイルを確認するには、以下のコマンドを実行します。

```
# pcp | grep logger
```

出力には、以下のようなアクティブなアーカイブファイルの名前が含まれます。

```
/var/log/pcp/pmlogger/satellite.example.com/20180814.00.10
```

第5章 メトリクスの取得

PCP からは CLI または web UI インターフェイスを使ってメトリクスを取得できます。PCP には多くの CLI ツールが用意されており、ライブデータまたはアーカイブ化されたソースからのデータを出力できます。web UI インターフェイスは、Grafana および Vector web アプリケーションが提供します。Vector は PCP デーモンに直接接続し、ライブデータのみを表示できます。Grafana は PCP アーカイブファイルからデータを読み取り、最大 1 年前までのものを表示することができます。

5.1. CLI によるメトリクスの取得

PCP の CLI ツールを使用すると、ライブまたはアーカイブファイルからメトリクスを取得できます。

5.1.1. CLI によるライブメトリクスの取得

ディスクパーティションの書き込みインスタンスに関するメトリクスを出力するには、以下のコマンドを実行します。

```
# pmval -f 1 disk.partitions.write
```

この例では、PCP はディスクパーティションへの書き込み数のカウンター値をレート値に変換します。**-f 1** は、値を小数第 2 位で四捨五入するように指定します。

出力例

```
metric:    disk.partitions.write
host:      satellite.example.com
semantics: cumulative counter (converting to rate)
units:     count (converting to count / sec)
samples:   all

              vda1             vda2              sr0
               0.0             12.0              0.0
               0.0              1.0              0.0
               0.0              1.0              0.0
               0.0              2.0              0.0
```

システムメトリクスを 2 秒間隔でモニターするには、以下のコマンドを実行します。

```
# pmstat -t 2sec
```

5.1.2. CLI によるアーカイブ化されたメトリクスの取得

PCP CLI ツールを使ってアーカイブファイルからメトリクスを取得できます。これを実行するには、**--archive** パラメーターを追加してアーカイブファイルを指定します。

- アーカイブファイル作成時に有効になっていた全メトリクスを一覧表示するには、以下のコマンドを実行します。

  ```
  pminfo --archive archive_file
  ```

- アーカイブファイルでカバーされているホストと時間を確認するには、以下のコマンドを実行します。

```
# pmdumplog -l archive_file
```

例

- アーカイブファイルがカバーしている期間の各パーティションのディスク書き込みを一覧表示するには、以下のコマンドを実行します。

```
# pmval --archive /var/log/pcp/pmlogger/satellite.example.com/20180816.00.10 \
  -f 1 disk.partitions.write
```

- 14:00 から 14:15 までの時間で 2 秒間隔でのパーティション毎のディスク書き込み操作を一覧表示するには、以下のコマンドを実行します。

```
# pmval --archive /var/log/pcp/pmlogger/satellite.example.com/20180816.00.10 \
  -d -t 2sec \
  -f 3 disk.partitions.write \
  -S @14:00 -T @14:15
```

- 14:00 から 14:30 までの時間で、最小値/最大値および実際の最小値/最大値を含むすべてのパフォーマンスメトリクスの平均値を一覧表示するには、以下のコマンドを実行します。出力は表形式になります。

```
# pmlogsummary /var/log/pcp/pmlogger/satellite.example.com/20180816.00.10 \
  -HlfiImM \
  -S @14:00 \
  -T @14:30 \
  disk.partitions.write \
  mem.freemem
```

- 14:00 時以降にアーカイブに保存されたシステムメトリクスを一覧表示するには、以下のコマンドを実行します。メトリクスは **top** ツールと同様のフォーマットで表示されます。

```
# pcp --archive /var/log/pcp/pmlogger/satellite.example.com/20180816.00.10 \
  -S @14:00 \
  atop
```

5.2. WEB UI によるメトリクスの取得

PCP メトリクスへの web UI インターフェイスにアクセスするには、以下のいずれかの URL で web アプリケーションを開きます。

Vector

http://**satellite.example.com**:44323/vector

Grafana

http://**satellite.example.com**:44323/grafana

これらのアプリケーションでは両方ともダッシュボード形式のビューが提供され、デフォルトのウィジェットがメトリクス値を表示します。メトリクスは要件に応じて追加したり、削除したりすることができます。また、各ウィジェットで表示する期間を選択できます。アーカイブ化されたメトリクスからカスタムの時間枠を選択できるのは、Grafana のみになります。

Grafana の使用方法に関する詳細は、Grafana Labs web サイトを参照してください。Vector の使用方法に関する詳細は、Vector web サイトを参照してください。

図5.1 Grafana ダッシュボードの例

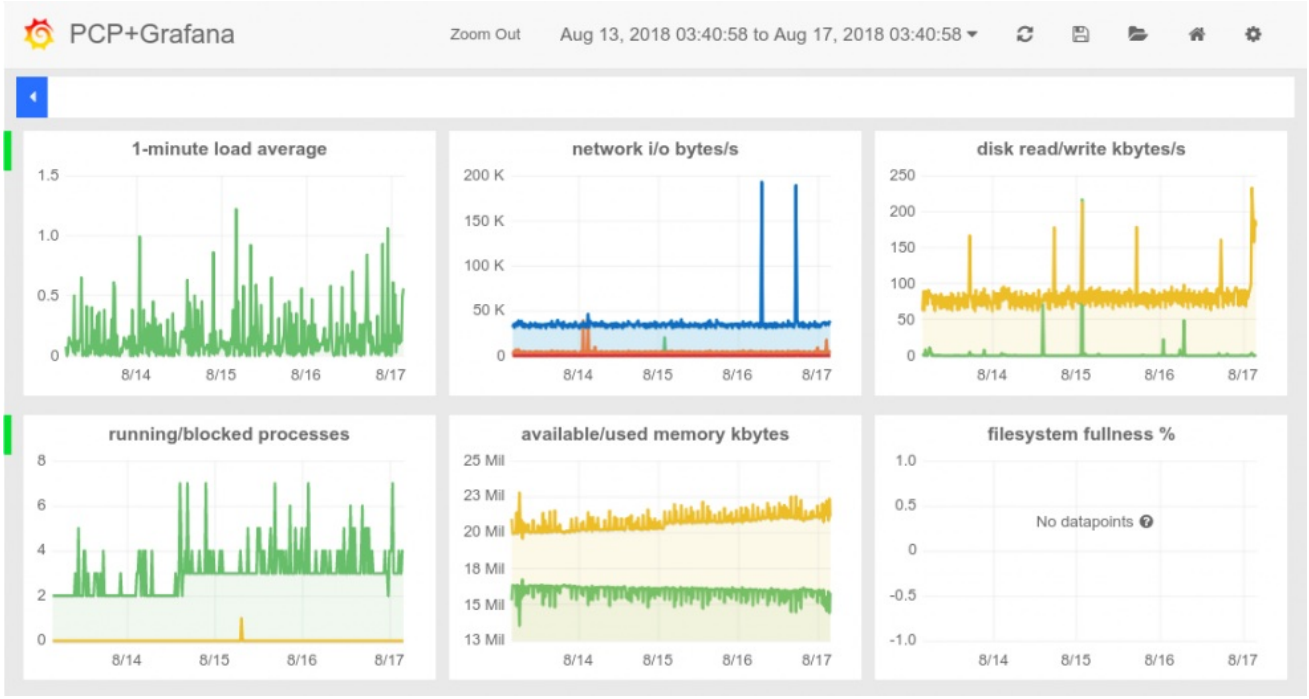
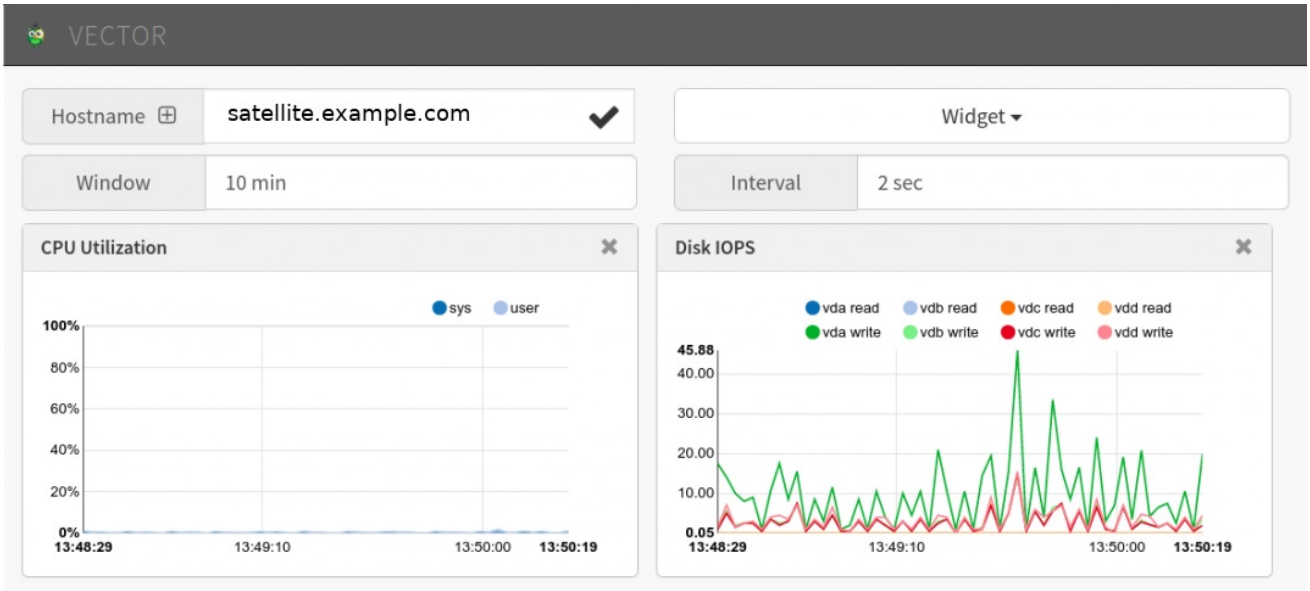


図5.2 Vector ダッシュボードの例



第6章 メトリクスデータの保持

PCP データログに必要なストレージ容量は、以下の要素で決定されます。

- ログ記録されるメトリクス
- ログ間隔
- 保持ポリシー

デフォルトのログ (サンプリング) 間隔は 60 秒です。

デフォルトの保持ポリシーでは、過去 14 日間のアーカイブが保持され、1 日以上前のものは圧縮されます。PCP アーカイブログは、**/var/log/pcp/pmlogger/hostname** ディレクトリーに保存されます。

6.1. デフォルトのログ間隔の変更

デフォルトのログ間隔は、以下の手順で変更します。

手順

1. **/etc/pcp/pmlogger/control.d/local** 設定ファイルを編集します。
2. LOCALHOSTNAME の行を編集し、**-t XXs** を追記します。ここでの **XX** は、新たな間隔 (秒単位) になります。
3. **pmlogger** サービスを再起動します。

6.2. データ保持ポリシーの変更

デフォルトのデータ保持ポリシーは、以下の手順で変更します。

手順

1. **/etc/cron.d/pcp-pmlogger** ファイルを編集します。
2. **pmlogger_daily** のある行を見つけます。
3. 何日後にデータをアーカイブ化するかという日数をパラメーター **-x** の値で設定します。
4. 何日後にデータを削除するかという日数を、パラメーター **-k** の値で設定します。
   たとえば、**-x 4 -k 7** というパラメーターの場合、データは 4 日後に圧縮され、7 日後には削除されます。

6.3. データストレージ使用量の確認

データストレージ使用量を確認するには、以下のコマンドを実行します。

```
# less /var/log/pcp/pmlogger/$(hostname)/pmlogger.log
```

これで利用可能な全メトリクスがログ記録された間隔ごとにグループ化されて一覧表示されます。各グループでは、一覧表示されたメトリクスの保存に必要なストレージも 1 日単位で一覧表示されます。

ストレージ統計数値の例

```
logged every 60 sec: 61752 bytes or 84.80 Mbytes/day
```