Linux
Getting started with OProfile
Linux
Getting started with OProfile
Note

Before using this information and the product it supports, read the information in “Notices” on page 13.
<table>
<thead>
<tr>
<th>Contents</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Getting started with OProfile</strong></td>
<td>1</td>
</tr>
<tr>
<td>Introduction to OProfile</td>
<td>1</td>
</tr>
<tr>
<td>Installing OProfile with the IBM Advance Toolchain and the SDK</td>
<td>2</td>
</tr>
<tr>
<td>Installing OProfile with distribution tools</td>
<td>3</td>
</tr>
<tr>
<td>Basic profiling with OProfile</td>
<td>4</td>
</tr>
<tr>
<td><code>operf</code> and <code>ocount</code> examples</td>
<td>4</td>
</tr>
<tr>
<td>Tips for basic profiling</td>
<td>6</td>
</tr>
<tr>
<td>Reviewing OProfile results with <code>opreport</code></td>
<td>6</td>
</tr>
<tr>
<td><code>opreport</code> examples</td>
<td>6</td>
</tr>
<tr>
<td>Basic OProfile scenarios</td>
<td>7</td>
</tr>
<tr>
<td>Profiling Java</td>
<td>7</td>
</tr>
<tr>
<td>Handling kernel throttling and lost samples with <code>operf</code></td>
<td>8</td>
</tr>
<tr>
<td>Related tools</td>
<td>9</td>
</tr>
<tr>
<td>Advanced profiling with OProfile</td>
<td>9</td>
</tr>
<tr>
<td>Installing kernel debug symbols</td>
<td>9</td>
</tr>
<tr>
<td>Creating advanced profiling reports</td>
<td>11</td>
</tr>
<tr>
<td>Tips for advanced profiling</td>
<td>11</td>
</tr>
<tr>
<td><strong>Notices</strong></td>
<td>13</td>
</tr>
<tr>
<td>Trademarks</td>
<td>14</td>
</tr>
<tr>
<td>Code license and disclaimer information</td>
<td>14</td>
</tr>
</tbody>
</table>
Getting started with OProfile

This topic gives you the information that you need to get started quickly with OProfile on Power Systems™ servers running Linux.

Before you begin

Note: By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 12.

Introduction to OProfile

OProfile is a statistical profiler for Linux systems, capable of profiling all running code at low overhead.

You can use OProfile to profile performance in a single process or running across your entire system. OProfile contains support for hardware performance counters. It profiles hardware and software interrupt handlers, kernel modules, the kernel, shared libraries, and applications.

OProfile supports all POWER® architectures that run Linux. OProfile support for Linux on Power is functionally equivalent to support on x86. OProfile is an industry accepted open source project that is shared across multiple system platforms and operating systems.

OProfile provides several commands and utilities to make profiling your application or system easier. The primary commands are `operf`, `ocount`, and `opreport`. Additional commands, including `opannotate` and `oparchive`, are described briefly in the following section.

`operf`

`operf` is a profiling tool that uses the Linux kernel performance events subsystem (`perf`) to profile a single process or the entire system. If you are profiling system-wide, you need to have root authority. `operf` was introduced in OProfile 0.9.8 and is available with most Linux distributions.

`ocount`

`ocount` is a tool that can be used to count native hardware events that occur in either a given application, a set of processes or threads, a subset of active system processors, or the entire system.

`opreport`

The `opreport` command provides a flexible and powerful reporting mechanism for the OProfile tools.

`opannotate`

The `opannotate` command provides annotated source or assembly code intermingled with instruction sampling counts.

`oparchive`

The `oparchive` utility gathers all of the profile data, executable, and libraries necessary to analyze profiling sessions into a specified directory where it can be easily tarred up and moved to other systems for subsequent analysis.
opcontrol

opcontrol was the original method for controlling profiling with OProfile. The opcontrol utility has been depreciated in later versions of OProfile in favor of the perf utility.

Installing OProfile with the IBM Advance Toolchain and the SDK

This topic documents the simplest path for installing OProfile with the IBM Advance Toolchain and the SDK on Power Systems servers.

About this task

OProfile is available with most Linux distros. If a newer version of the tool is required, it can be downloaded from the SourceForge repository, built, and installed on any Linux system. IBM® includes OProfile in the IBM Advance Toolchain for Linux on Power® packages that are hosted on the IBM Tools Repository. Complete these steps to set up the repository and install the latest IBM Advance Toolchain for Linux on Power packages and tools.

The latest OProfile package is available with the IBM Advance Toolchain for Linux on Power installation. Follow these steps to install the Advance Toolchain:

Procedure

1. First, download and install the YUM repository configuration RPM. This package configures your Linux installation to automatically download and install packages using yum or zypper from the YUM repository.
   b. In the Configuration RPM file for YUM repository section, click the Configuration RPM file link to download it.
   c. Transfer the RPM file to the Power system.
   d. On the Power system, enter the following command to install the configuration RPM file:
      ```
rpm -ivh ibm-power-repo-version.ppc.rpm
      ```
      Note:
      - Replace version with the version of the PowerLinux YUM repository configuration RPM file that you downloaded.
      - Root authority is typically required to install packages.

2. Install the recommended service and utility tools for your distribution and IBM Power Systems server, as described on the IBM Tools Repository website. Replace package_name with the appropriate package name for your distribution and Power Systems server. For example, `ibm-power-nonmanaged-rhel6`.
   a. For Red Hat Enterprise Linux, enter the following command:
      ```
yum install package_name
      ```
   b. For SUSE Linux Enterprise Server, enter the following command:
      ```
zypper install package_name
      ```

3. Install the latest IBM Advance Toolchain for Linux on Power packages. Advance Toolchain includes the most current version of OProfile.
   a. For Red Hat Enterprise Linux, enter the following commands:
      ```
yum install advance-toolchain-atversion-runtime
y whole install advance-toolchain-atversion-devel
y whole install advance-toolchain-atversion-perf
      ```
   b. For SUSE Linux Enterprise Server, enter the following commands:
Replace `version` with the downloaded version of the IBM Advance Toolchain for Linux on Power, for example, "6.0". This step installs the three required IBM Advance Toolchain for Linux on Power RPM packages, in the correct order.

For information about IBM Advance Toolchain for Linux on Power, see [Improving performance with IBM Advance Toolchain for Linux on Power](http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/performance/advancetoolchain/advancetoolchain.htm).

4. Optional: Install the latest IBM Software Development Kit for Linux on Power. IBM SDK for Linux on Power allows you to profile your applications with OProfile through an Eclipse-based interface.

   • For Red Hat Enterprise Linux, enter the following command:
     
     ```
     yum install ibm-sdk-lp
     ```

   • For SUSE Linux Enterprise Server, enter the following command:
     
     ```
     zypper install ibm-sdk-lp
     ```

   For information about IBM SDK for Linux on Power, see [Developing software using the IBM Software Development Kit for Linux on Power](http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaal/iplsdkmain.htm).

5. Update the path on your system to add the Advance Toolchain directory.

   • To update the path for a single session, enter the following commands:
     
     ```
     export PATH=/opt/atversion/bin:$PATH
     export MANPATH=/opt/atversion/share/man:$MANPATH
     ```

     **Note:** Replace `version` with the version of the Advance Toolchain that you downloaded.

   • To make these environment changes persistent, add these export commands to the users profile (for example, .bashrc) or to a system profile (for example, /etc/profile.d/ ). See your Linux distribution documentation for more information on how to properly set the values.

6. Finally, to validate that the path changed and packages installed correctly, enter this sequence of commands:

   ```
   $ which operf
   operf --version
   operprof --version
   /opt/at9.0/bin/operf
   /opt/at9.0/bin/ocount
   operf: oprofile 1.1.0git compiled on Nov 18 2015 19:42:05
   operprof: oprofile 1.1.0git compiled on Nov 18 2015 19:41:53
   ```

   After completing these steps, you can use the OProfile tools, including the `operf`, `ocount`, and `operreport` commands.

---

### Installing OProfile with distribution tools

Install OProfile using tools available from your Linux distribution. These instructions do not use IBM Advance Toolchain or the SDK.

**Note:** These instructions are provided as a guide. Review the documentation associated with your installed distribution for updates or additional steps and options regarding how to install additional packages.

**Fedora, Red Hat Enterprise Linux**

```
$ yum install oprofile
```

**SUSE Linux Enterprise Server, OpenSUSE**

```
$ zypper install oprofile or yast2 -i oprofile
```
Ubuntu, debian
$ apt-get install oprofile

Basic profiling with OProfile

After you have OProfile installed, you can get started with basic profiling. These examples show how to gather the profiling information from the system.

Profiling sessions typically consist of two steps: Collection of data using `operf` or `ocount`; and then a review of that data using `opreport` or `opannotate`. For example, for data collection, use one of the following commands:

```
operf [options] <my-app>
operf [options] -p <pid-to-trace>
operf [options] [ --system-wide ] <ctrl-c to stop>
ocount [options] <my-app>
```

followed by some combination of the following commands:

```
opreport [options]
opannotate [options]
```

**operf and ocount examples**

These examples demonstrate using the `operf` and `ocount` commands for a single application.

**Note:** By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 12.

**Event data collection**

The following example profiles your application with the default cycles-based event for the processor type on which you are running.

```
operf application [arguments]
$ operf ./load_v2 > load_v2.log
operf: Profiler started
profiled app exited with the following status: 30
Profiling done.
```

Note that the `operf` command will run until the application has finished running. If necessary, you can stop the data collection by pressing Ctrl+C. This action ends the data collection and terminates the application. The message: profiled app exited with the following status: 30 in the previous example reflects the non-zero return code of the application.

**System-wide data collection**

Use the `operf` command for system-wide data collection over the run of the workload. The system-wide option can only be run as root user.

In terminal #1:

```
$ operf --system-wide
operf: Press Ctrl-c or 'kill -SIGINT 31262' to stop profiling
operf: Profiler started
```

In terminal #2:

```
$ ./load_v2 > load_v2.log
# upon completion, go back to terminal #1;
```
In terminal #1:

`^C`

Profiling done.

When using the system-wide option, the `operf` command will run until you stop it with Ctrl+C or by entering the `kill -SIGINT PID` command, where `PID` is the process identifier of the `operf` process.

**Data storage**

System-wide profiling is performed as root so that the profile data will be created and stored in subdirectories that will be owned by root. These subdirectories will initially be unreadable by the user, and subsequent `operf` operations by the user will fail. Use the `chown` command to authorize these subdirectories to be viewed by non-root users. For example, run `chown -R USERID oprofile_data`

By default, `operf` stores the sample data in `current-directory/oprofile_data/samples/current`, where `current-directory` is the directory from which you run `operf`.

**Events**

Power systems contain hundreds of PMU events that can be measured. Some particularly useful PMU events include the following events:

**Eliminate idle loops from measurements**

```
PM_RUN_CYC:100000,PM_RUN_INST_CMPL:100000
```

This event allows for a quick cycles per instruction (CPI) calculation. Reasonable CPI values will vary depending on the system environment and the workload. A CPI of 4 or greater for a section of code is a strong indicator of a hazard, which can be further studied with the following events.

**Measure Load-Hit-Store impacts**

```
PM_LSU_REJECT:10000,PM_LSU_REJECT_LHS:10000,PM_FLUSH_LSU:10000
```

This event covers the specific LHS case and identifies the REJECTS versus the FLUSHes.

**Measure cache misses**

```
PM_DATA_FROM_LMEM:1000,PM_DATA_FROM_RMEM:1000
```

This event provides stats on L3 misses for both local and remote memory.

```
PM_DATA_FROM_L2:10000,PM_DATA_FROM_L3:10000,PM_DATA_FROM_MEM:10000
```

This event provides stats on L1 misses resolved on the local core.

**Note:** The IBM SDK tools use `PM_RUN_CYC` as a default event. When run directly, the oprofile tools uses the `CYCLES` event by default. `CYCLES` includes idle time while `PM_RUN_CYC` limits data collection to non-idle time. `PM_RUN_CYC` is the recommended initial event for Power.

**Tips for profiling on a specific event**

To profile on specific events, consider the following:

- Use the `--events` option and provide a comma-separated list of events.
- When specifying an event for profiling (`operf`), it is also necessary to specify a count. The default (recommended) count value can be found running `ophelp` and searching for the event name. The count value determines how many times the PMU will see the event before triggering a sample. In the following example, the count for `CYCLES` is 100000.

```
$ ophelp | grep -A 1 CYCLES
CYCLES: (counter: 0)
  Cycles (min count: 100000)
```

- You can specify any combination of events. If necessary, the kernel performs multiplexing of the events.
When specifying an event for counting (`ocount`), do not specify a count.

**Note:** Because a multiplexed event is not counted continuously throughout the profile run, the absolute number of samples that are collected is less than otherwise. If you profile a short-lived application using several events where multiplexing occurs, some events might never be counted at all. Profiling for a longer time smooths out the sampling periods and produces a more statistically valid profile.

- View the man pages for `perf` and the output from `ophelp` for more information about how to request profiling on a specific event.

**ocount examples**

Calculate a CPI ratio for the application:

```
$ ocount --events PM_RUN_CYC,PM_RUN_INST_CMPL ./load_v2
<...>
Event counts (actual) for /home/willschm/load_v2:
Event          Count            % time counted
PM_RUN_CYC    856,069,007          100.00
PM_RUN_INST_CMPL  631,054,215          100.00
```

The numbers in this example show a CPI of approximately 1.3.

**Tips for basic profiling**

Use the following tips when you use `perf` or `ocount` for basic profiling.

- To include profiling of the kernel with kernel symbol information, include the `--vmlinux` option with the `perf` command, and specify the fully qualified file name, including path, to a non-compressed `vmlinux` file. For example, `--vmlinux=/boot/vmlinux-3.0.101-63-ppc64`. In some environments, the non-compressed `vmlinux` can be found in `/usr/lib/debug/lib/modules/*/vmlinux`.
- Use the `ophelp` command to list the available events to profile.
- Use the `ocount` command to collect raw event counts for your application or system.
- Use the `oparchive` utility to create an archive of the profile data that can be subsequently analyzed on another system. For example: `oparchive --output-directory=~/.Nov18.1.oparchive`.

**Reviewing OProfile results with oreport**

Use the `opreport` command to generate a report of the results.

For example, use the following command to view a high level, per-function report.

```
$ oreport --symbols
```

**Note:** `opreport` (and `opannotate` and `oparchive`) first look in `current-directory/opprofile_data/samples/current` for sample data, where `current-directory` is the directory from which you run `perf`. If no data is found there, the commands then look in `/var/lib/opprofile/samples/current`.

**opreport examples**

This example shows the results of using `opreport` after you use `perf` for a single application.

**Note:** By using the code examples, you agree to the terms of the [Code license and disclaimer information](#) on page 12.

In this example, `perf` was used to profile an application that is named “load_v2”.

Then, the `opreport` command was used to show the results:

```
$ oreport --symbols
```
The opreport output for this example then is as follows:

$ opreport --symbols -t 2
Using /home/willschm/oprofile_data/samples/ for samples directory.
warning: /no-vmlinux could not be found.
warning: [hypervisor_bucket] (tgid:31407 range:0x2130c-0x15ee9dc) could not be found.
CPU: ppc64 POWER7, speed 3000 MHz (estimated)
Counted CYCLES events (Processor Cycles) with a unit mask of 0x00 (No unit mask) count 100000

<table>
<thead>
<tr>
<th>samples %</th>
<th>image name</th>
<th>symbol name</th>
</tr>
</thead>
<tbody>
<tr>
<td>6351</td>
<td>load_v2</td>
<td>.main</td>
</tr>
<tr>
<td>3527</td>
<td>load_v2</td>
<td>.sum_sub</td>
</tr>
<tr>
<td>1937</td>
<td>load_v2</td>
<td>.sum_add</td>
</tr>
</tbody>
</table>

Tip: As shown in the sample output, with "-t 2", specify a threshold to eliminate content with a very low overall impact.

The following example describes results of using opreport after using operf for system-wide profiling.

If you profiled system-wide, and have collected information for all running applications on the system, running opreport can produce a large amount of output data. If you are interested in just a particular application, you can filter out the extra data. To show results for only a particular running application, specify the application path as the last argument of the opreport command.

For example, collect data with a command similar to the following line:

$ sudo operf --system-wide --vmlinux=/usr/lib/debug/lib/modules/2.6.32-431.el6.ppc64/vmlinux

Then, use the following command to show the results:

$ opreport --symbols /home/willschm/load_v2 -t 0.01
Using /home/willschm/oprofile_data/samples/ for samples directory.
warning: [hypervisor_bucket] (tgid:31454 range:0x2130c-0x16e8c6c) could not be found.
CPU: ppc64 POWER7, speed 3000 MHz (estimated)
Counted CYCLES events (Processor Cycles) with a unit mask of 0x00 (No unit mask) count 100000

<table>
<thead>
<tr>
<th>samples %</th>
<th>image name</th>
<th>symbol name</th>
</tr>
</thead>
<tbody>
<tr>
<td>6481</td>
<td>load_v2</td>
<td>.main</td>
</tr>
<tr>
<td>3412</td>
<td>load_v2</td>
<td>.sum_sub</td>
</tr>
<tr>
<td>1963</td>
<td>load_v2</td>
<td>.sum_add</td>
</tr>
<tr>
<td>10</td>
<td>vmlinux</td>
<td>.memcpy_power7</td>
</tr>
<tr>
<td>9</td>
<td>vmlinux</td>
<td>.update_mmu_cache</td>
</tr>
<tr>
<td>4</td>
<td>vmlinux</td>
<td>.arch_prepare_kprobe</td>
</tr>
<tr>
<td>2</td>
<td>ext4</td>
<td>/ext4</td>
</tr>
<tr>
<td>2</td>
<td>jbd2</td>
<td>/jbd2</td>
</tr>
<tr>
<td>2</td>
<td>vmlinux</td>
<td>.offline_pages</td>
</tr>
<tr>
<td>2</td>
<td>vmlinux</td>
<td>.raw_local_irq_restore</td>
</tr>
</tbody>
</table>

Filtered in this way, the results are similar to the results in opreport examples on page 6.

---

**Basic OProfile scenarios**

Here are some basic scenarios and examples for using OProfile.

**Profiling Java**

This topic provides information about profiling Java applications with OProfile.

IBM Advance Toolchain for Linux on Power already includes the OProfile Java agent libraries. When Advance Toolchain is installed, the system's library search path is updated to include those libraries. Therefore, if you previously installed Advance Toolchain, you need only initialize JVM with the correct option.

**Handling kernel throttling and lost samples with operf**

This topic provides information about handling kernel throttling that results in lost samples.

**Kernel throttling**

*Note:* By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 12.

Kernel throttling controls the profiling rate, which is the number of samples that are taken for a specific period. Throttling might occur during a profiling session if the kernel determines that the initial profiling rate is causing excessive system usage.

*Note:* The event names might vary between hardware platforms. POWER8 systems have more events than POWER7 Systems, and some events might have different names. The *ophelp* command lists the events that are currently supported on the system you run it on.

The **minimum count value** *(min count:)* can be found in the output of *ophelp* associated with the event. For example:

```
$ ophelp | grep -A 1 PM_RUN_  
PM_RUN_CYC: (counter: 5)  
  Run_cycles. (min count: 100000)  
PM_RUN_INST_CMPL: (counter: 4)  
  Run_Instructions. (min count: 100000)
```

With system-wide profiling, and sometimes with single-application profiling, depending on the load on the system, this sampling rate might be too high. If the sampling rate is too high, the following message is issued:

```
*** WARNING: Profiling rate was throttled back by the kernel ***
The number of samples actually recorded is less than expected, but is probably still statistically valid. Decreasing the sampling rate is the best option if you want to avoid throttling.
```

To avoid profiling rate throttling, you can specify a higher samples per event rate using the **--events** parameter with *operf*.

For example:

```
$ operf --events PM_RUN_CYC:100000 ./load_v2 > foo.log
<...>
*** WARNING: Profiling rate was throttled back by the kernel ***
$ operf --events PM_RUN_CYC:1000000 ./load_v2 > foo.log
Profiling done.
```

**Lost samples**

Occasionally, you may see one of the following messages after you run *operf*, or when running *opreport*:

```
WARNING: Lost samples detected! See /current-directory/opprofile_data/samples/operf.log for details.
```

This message is printed if any samples are lost.

The following example shows the *operf.log* output after such a message is printed:

```
Profiling started at Mon Oct 12 13:48:43 2015
```
-- OProfile/operf Statistics --
Nr. non-backtrace samples: 16244
Nr. kernel samples: 1837
Nr. user space samples: 14407
Nr. samples lost due to sample address not in expected range for domain: 0
Nr. lost kernel samples: 0
Nr. samples lost due to sample file open failure: 0
Nr. samples lost due to no permanent mapping: 0
Nr. user context kernel samples lost due to no app info available: 0
Nr. user samples lost due to no app info available: 0
Nr. backtraces skipped due to no file mapping: 0
Nr. hypervisor samples dropped due to address out-of-range: 8
Nr. samples lost reported by perf_events kernel: 0

Small numbers of lost or dropped samples, as shown in the previous example, can safely be ignored. It is impossible to say definitively how many lost samples can be safely ignored. However, most likely anything over 1% indicates a problem that requires investigation or fixing. You may be able to prevent or minimize the lost samples by lowering the sampling rate, as described earlier.

Related tools
You can add to your profiling options by using these related tools.

The IBM Software Development Kit for Linux on Power is a free, Eclipse-based integrated development environment (IDE). The IBM SDK for Linux on Power integrates C/C++ source development with the IBM Advance Toolchain for Linux on Power, Post-Link Optimization, and classic Linux performance analysis tools, including OProfile and Valgrind. For information about IBM SDK for Linux on Power, see [Developing software using the IBM Software Development Kit for Linux on Power](http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaal/iplsdkmain.htm).

The Linux Performance Customer Profiler Utility (LPCPU) package consists of scripts that you can use to capture and format system information for performance analysis. The script, which runs on both x86 and Power Systems servers, gathers much of the information that is needed to understand, debug, and analyze performance problems. Go to [Linux Performance Customer Profiler Utility (lpcpu)](https://www.ibm.com/developerworks/mydeveloperworks/wikis/home?lang=en#/wiki/W51a77fc4fdd_f4b40_9d82_4466be23c550/page/Linux%20Performance%20Customer%20Profiler%20Utility%20%20%28lpcpu%29) on the IBM developerWorks® PowerLinux Community wiki for details.

Advanced profiling with OProfile
When you are ready to go beyond the basics, these topics describe more advanced profiling techniques and examples.

Installing kernel debug symbols
You can display kernel symbols by installing the kernel-debuginfo and kernel-debuginfo-common debug packages.

Procedure
1. Install the debuginfo packages. The naming of these packages may vary depending on the Linux distribution.

   **Note:** These instructions are provided as a guide. Review the documentation associated with your installed distribution for updates or additional steps and options regarding how to install the debuginfo packages.

   • Red Hat Enterprise Linux:
     Install kernel-debuginfo and kernel-debuginfo-common packages.
Red Hat Enterprise Linux 6: `sudo yum install kernel-debuginfo kernel-debuginfo-common`
Red Hat Enterprise Linux 7: `sudo debuginfo-install kernel`

- SUSE Linux Enterprise Server:
  SUSE Linux Enterprise Server 11: `/sbin/yast2 -i kernel-ppc64-debuginfo`
  SUSE Linux Enterprise Server 12: `zypper --non-interactive install kernel-default-debuginfo`


- Ubuntu:
  Add additional repositories that contain the debuginfo packages:
  ```
  $ echo "deb http://ddebs.ubuntu.com $({lsb_release -cs}) main restricted universe multiverse"
  | sudo tee -a /etc/apt/sources.list.d/ddebs.list
  $ echo "deb http://ddebs.ubuntu.com $({lsb_release -cs}-updates main restricted universe multiverse"
  | sudo tee -a /etc/apt/sources.list.d/ddebs.list
  - refresh repository info
  $ apt-get update
  - install the debug symbols package
  $ sudo apt-get install linux-image-$(uname -r)-dbgsym
  ```

2. Identify the correct vmlinux file.
   - The vmlinux file must match the currently running kernel version. This can be verified with `uname -a` or `uname -r`.
   - The vmlinux file must not be “vmlinuz”. The “z” indicates that the file is the compressed version of the vmlinux file.

For Red Hat Enterprise Linux, after you have installed the debug packages, the correct vmlinux file is in the path `/usr/lib/debug/lib/modules/`uname -r`/vmlinux` or `/usr/lib/debug/lib/usr/lib/modules/`uname -r`/vmlinux`, depending on the release.

For SUSE Linux Enterprise Server, the debug image is `/boot/vmlinux`.

For Ubuntu, the debug image is `/usr/lib/debug/boot/vmlinux-version.debug`

**Note:** Some kernel address information may be disabled by default. In these scenarios, a message is displayed:

Samples for vmlinux kernel will be recorded, but kernel module profiling is not possible with current system config. Set `/proc/sys/kernel/kptr_restrict` to 0 to see samples for kernel modules.

To enable kernel pointers, as root, run the following:

```
# echo 0 > /proc/sys/kernel/kptr_restrict
```

3. To profile with the Linux kernel, you can run `operf` with the `--vmlinux=kernel_image_path` option.

   `operf` profiles without the Linux kernel by default, unless you specify the `--vmlinux` option.

4. To display the symbol information, enter the following command:

```
$ oreport --symbols
```

For example:

```
CPU: ppc64 POWER8, speed 4157 MHz (estimated)
Counted CYCLES events (Cycles) with a unit mask of 0x00 (No unit mask) count 100000
samples % image name symbol name
788 59.1148 load_v2 .main
264 19.8050 load_v2 .sum_add
234 17.5544 load_v2 .sum_sub
10 0.7502 vmlinux-3.0.101-65-ppc64 .clear_user_page
3 0.2251 ld-2.11.3.so .dl_relocate_object
3 0.2251 vmlinux-3.0.101-65-ppc64 .copy_page
<...>
```

Alternatively, to display both the kernel function names where the samples were taken as well as the source file and line number where the function is found, enter the following command:

```
$ oreport --symbols --debug-info
```
For example:

CPU: ppc64 POWER8, speed 4157 MHz (estimated)
Counted CYCLES events (Cycles) with a unit mask of 0x00 (No unit mask) count 100000

<table>
<thead>
<tr>
<th>samples</th>
<th>linenr</th>
<th>image</th>
<th>name</th>
<th>symbol</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>7472</td>
<td>70.1135</td>
<td>load_v2.c:17</td>
<td>load_v2</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>1609</td>
<td>15.0981</td>
<td>load_v2.c:129</td>
<td>load_v2</td>
<td>sum_add</td>
<td></td>
</tr>
<tr>
<td>1525</td>
<td>14.3098</td>
<td>load_v2.c:134</td>
<td>load_v2</td>
<td>sum_sub</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.0657</td>
<td>mem.c:433</td>
<td>vmlinux-4.2.0-16-generic</td>
<td>clear_user_page</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.0282</td>
<td>hash_utils_64.c:996</td>
<td>vmlinux-4.2.0-16-generic</td>
<td>hash_page_mm</td>
<td></td>
</tr>
</tbody>
</table>

Creating advanced profiling reports

You can create advanced profiling reports, such as annotated and callgraph profiling reports, to obtain more details for analysis.

When you use oprofile with the --callgraph option, you can see what functions are calling other functions in the output. The OProfile manual at http://oprofile.sourceforge.net contains detailed information about this option. See [Callgraph output](http://oprofile.sourceforge.net/doc/opreport.html#oprofile-callgraph).

You can use the opannotate command to create annotated source files and assembly listings. The OProfile manual includes the details about this utility. See [Outputting annotated source (oprofile)](http://oprofile.sourceforge.net/doc/opannotate.html).

Tips for advanced profiling

Use the following tips for advanced profiling.

By default, OProfile reports do not directly show elapsed time. Follow these suggestions to help approximate elapsed time.

- Profile on RUN_CYCLES events to count cycles only when the processor is not idle.
  - Use the PM_RUN_CYC event for profiling to collect samples for when the application is running.

  **Note:** Under certain circumstances, such as large multi-core systems that are running multi-threaded applications, OProfile might not be able to process all the samples the kernel collects. If this occurs, some samples might be dropped.
  - If you find that samples are being dropped, increase your sampling interval.
- Generate a summary report for profiling.
  - After the profile is collected, you can generate a summary report using oreport by only specifying the application name. For example, oreport load_v2. The summary report lists the samples for the application, as well as system library and kernel activity done on behalf of the application.
  - You can then generate a symbol-level report using oreport with the --symbols option. This command lists the number of samples at the function level. Use the -t # threshold parameter to eliminate low impact content.
  - If your symbol-level report includes samples that were taken in kernel modules, the sampled addresses might not show the correct module functions. To resolve the sampled addresses, use the --image-path option with oreport, and specify the path for the kernel modules. For example, oreport --image-path /lib/modules/`uname -r`/.
- Use the reports and collected data to approximate the time elapsed. Use the following calculations:
  - \( \frac{CPU\_rated\_speed}{OProfile\_sample\_count} = Number\_of\_samples\_per\_second \)

  **Note:** \( CPU\_rated\_speed \) is the clock rate as reported by /proc/cpuinfo, represented as Hz. (convert from MHz).
  - \( \frac{Number\_of\_application\_samples}{Number\_of\_samples\_per\_second} = Approximate\_time\_elapsed \)
Note: The approximations as seen here do not take into account a number of factors that may throw off these numbers. Variations will be seen due to power saving modes, shared processors, affects of other workloads within the same system, and overhead due to activity at the hypervisor layer. Doing these calculations using other event types (such as cache misses), can give you an idea of the overall time spent handling the specified event.

For example:

$ grep -m 1 clock /proc/cpuinfo
clock : 4157.000000MHz
# convert from MHz, our CPU Rated_speed is 4157000000 Hz.

$ operf --event CYCLES:400000 /load_v3
# our sample count as specified here is 400000.

4,157,000,000 (hz) / 400000 (sample count) = 10392.5 samples per second. (theoretical max)

$ oreport load_v3
<...>
  200253 100.000 load_v3

200253 (samples) / 103292.6 (samples/sec) = 19.29 seconds.

# now for comparison, run the app under 'time' and measure directly.

# note: Specify the full path to the time binary, avoid using the shell built-in to get the format string capability.
$ /usr/bin/time -f "%e" /load_v3
18.58

Code license and disclaimer information

IBM grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.

SUBJECT TO ANY STATUTORY WARRANTIES WHICH CANNOT BE EXCLUDED, IBM, ITS PROGRAM DEVELOPERS AND SUPPLIERS MAKE NO WARRANTIES OR CONDITIONS EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY.

UNDER NO CIRCUMSTANCES IS IBM, ITS PROGRAM DEVELOPERS OR SUPPLIERS LIABLE FOR ANY OF THE FOLLOWING, EVEN IF INFORMED OF THEIR POSSIBILITY:

1. LOSS OF, OR DAMAGE TO, DATA;
2. DIRECT, SPECIAL, INCIDENTAL, OR INDIRECT DAMAGES, OR FOR ANY ECONOMIC CONSEQUENTIAL DAMAGES; OR
3. LOST PROFITS, BUSINESS, REVENUE, GOODWILL, OR ANTICIPATED SAVINGS.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF DIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.
Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Dept. LRAS/Bldg. 903
11501 Burnet Road
Austin, TX 78758-3400
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.
For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

**Trademarks**

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® and ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

**Code license and disclaimer information**

IBM grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.

SUBJECT TO ANY STATUTORY WARRANTIES WHICH CANNOT BE EXCLUDED, IBM, ITS PROGRAM DEVELOPERS AND SUPPLIERS MAKE NO WARRANTIES OR CONDITIONS EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR
CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY.

UNDER NO CIRCUMSTANCES IS IBM, ITS PROGRAM DEVELOPERS OR SUPPLIERS LIABLE FOR ANY OF THE FOLLOWING, EVEN IF INFORMED OF THEIR POSSIBILITY:

1. Loss of, or damage to, data;
2. Direct, special, incidental, or indirect damages, or for any economic consequential damages; or
3. Lost profits, business, revenue, goodwill, or anticipated savings.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF DIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.