Debugging hibernation and suspend

1. Testing hibernation (aka suspend to disk or STD)

To check if hibernation works, you can try to hibernate in the "reboot" mode:

```
# echo reboot > /sys/power/disk
# echo disk > /sys/power/state
```

and the system should create a hibernation image, reboot, resume and get back to the command prompt where you have started the transition. If that happens, hibernation is most likely to work correctly. Still, you need to repeat the test at least a couple of times in a row for confidence. [This is necessary, because some problems only show up on a second attempt at suspending and resuming the system.] Moreover, hibernating in the "reboot" and "shutdown" modes causes the PM core to skip some platform-related callbacks which on ACPI systems might be necessary to make hibernation work. Thus, if you machine fails to hibernate or resume in the "reboot" mode, you should try the "platform" mode:

```
# echo platform > /sys/power/disk
# echo disk > /sys/power/state
```

which is the default and recommended mode of hibernation.

Unfortunately, the "platform" mode of hibernation does not work on some systems with broken BIOSes. In such cases the "shutdown" mode of hibernation might work:

```
# echo shutdown > /sys/power/disk
# echo disk > /sys/power/state
```

(it is similar to the "reboot" mode, but it requires you to press the power button to make the system resume).

If neither "platform" nor "shutdown" hibernation mode works, you will need to identify what goes wrong.

a) Test modes of hibernation

To find out why hibernation fails on your system, you can use a special testing facility available if the kernel is compiled with CONFIG_PM_DEBUG set. Then, there is the file /sys/power/pm_test that can be used to make the hibernation core run in a test mode. There are 5 test modes available:

freezer
- test the freezing of processes

devices
- test the freezing of processes and suspending of devices

platform
- test the freezing of processes, suspending of devices and platform
  global control methods(*)

processors

- test the freezing of processes, suspending of devices, platform
  global control methods(*) and the disabling of nonboot CPUs

core
- test the freezing of processes, suspending of devices, platform global
  control methods(*), the disabling of nonboot CPUs and suspending of
  platform/system devices

(*) the platform global control methods are only available on ACPI systems
    and are only tested if the hibernation mode is set to "platform"

To use one of them it is necessary to write the corresponding string to
/sys/power/pm_test (eg. "devices" to test the freezing of processes and
suspending devices) and issue the standard hibernation commands. For example,
to use the "devices" test mode along with the "platform" mode of hibernation,
you should do the following:

echo devices > /sys/power/pm_test
echo platform > /sys/power/disk
echo disk > /sys/power/state

Then, the kernel will try to freeze processes, suspend devices, wait 5 seconds,
resume devices and thaw processes. If "platform" is written to
/sys/power/pm_test , then after suspending devices the kernel will additionally
invoke the global control methods (eg. ACPI global control methods) used to
prepare the platform firmware for hibernation. Next, it will wait 5 seconds and
invoke the platform (eg. ACPI) global methods used to cancel hibernation etc.

Writing "none" to /sys/power/pm_test causes the kernel to switch to the normal
hibernation/suspend operations. Also, when open for reading, /sys/power/pm_test
contains a space-separated list of all available tests (including "none" that
represents the normal functionality) in which the current test level is
indicated by square brackets.

Generally, as you can see, each test level is more "invasive" than the previous
one and the "core" level tests the hardware and drivers as deeply as possible
without creating a hibernation image. Obviously, if the "devices" test fails,
the "platform" test will fail as well and so on. Thus, as a rule of thumb, you
should try the test modes starting from "freezer", through "devices", "platform"
and "processors" up to "core" (repeat the test on each level a couple of times
to make sure that any random factors are avoided).

If the "freezer" test fails, there is a task that cannot be frozen (in that case
it usually is possible to identify the offending task by analysing the output of
dmesg obtained after the failing test). Failure at this level usually means
that there is a problem with the tasks freezer subsystem that should be
reported.

If the "devices" test fails, most likely there is a driver that cannot suspend
or resume its device (in the latter case the system may hang or become unstable
after the test, so please take that into consideration). To find this driver,
you can carry out a binary search according to the rules:
- if the test fails, unload a half of the drivers currently loaded and repeat
(that would probably involve rebooting the system, so always note what drivers
have been loaded before the test),
- if the test succeeds, load a half of the drivers you have unloaded most

recently and repeat.

Once you have found the failing driver (there can be more than just one of them), you have to unload it every time before hibernation. In that case please make sure to report the problem with the driver.

It is also possible that the "devices" test will still fail after you have unloaded all modules. In that case, you may want to look in your kernel configuration for the drivers that can be compiled as modules (and test again with these drivers compiled as modules). You may also try to use some special kernel command line options such as "noapic", "noacpi" or even "acpi=off".

If the "platform" test fails, there is a problem with the handling of the platform (eg. ACPI) firmware on your system. In that case the "platform" mode of hibernation is not likely to work. You can try the "shutdown" mode, but that is rather a poor man's workaround.

If the "processors" test fails, the disabling/enabling of nonboot CPUs does not work (of course, this only may be an issue on SMP systems) and the problem should be reported. In that case you can also try to switch the nonboot CPUs off and on using the /sys/devices/system/cpu/cpu*/online sysfs attributes and see if that works.

If the "core" test fails, which means that suspending of the system/platform devices has failed (these devices are suspended on one CPU with interrupts off), the problem is most probably hardware-related and serious, so it should be reported.

A failure of any of the "platform", "processors" or "core" tests may cause your system to hang or become unstable, so please beware. Such a failure usually indicates a serious problem that very well may be related to the hardware, but please report it anyway.

b) Testing minimal configuration

If all of the hibernation test modes work, you can boot the system with the "init=/bin/bash" command line parameter and attempt to hibernate in the "reboot", "shutdown" and "platform" modes. If that does not work, there probably is a problem with a driver statically compiled into the kernel and you can try to compile more drivers as modules, so that they can be tested individually. Otherwise, there is a problem with a modular driver and you can find it by loading a half of the modules you normally use and binary searching in accordance with the algorithm:
- if there are n modules loaded and the attempt to suspend and resume fails, unload n/2 of the modules and try again (that would probably involve rebooting the system),
- if there are n modules loaded and the attempt to suspend and resume succeeds, load n/2 modules more and try again.

Again, if you find the offending module(s), it(they) must be unloaded every time before hibernation, and please report the problem with it(them).

c) Advanced debugging

In case that hibernation does not work on your system even in the minimal configuration and compiling more drivers as modules is not practical or some

modules cannot be unloaded, you can use one of the more advanced debugging techniques to find the problem. First, if there is a serial port in your box, you can boot the kernel with the 'no_console_suspend' parameter and try to log kernel messages using the serial console. This may provide you with some information about the reasons of the suspend (resume) failure. Alternatively, it may be possible to use a FireWire port for debugging with firescope (ftp://ftp.firstfloor.org/pub/ak/firescope/). On x86 it is also possible to use the PM_TRACE mechanism documented in Documentation/s2ram.txt .

2. Testing suspend to RAM (STR)

To verify that the STR works, it is generally more convenient to use the s2ram tool available from http://suspend.sf.net and documented at http://en.opensuse.org/s2ram . However, before doing that it is recommended to carry out STR testing using the facility described in section 1.

Namely, after writing "freezer", "devices", "platform", "processors", or "core" into /sys/power/pm_test (available if the kernel is compiled with CONFIG_PM_DEBUG set) the suspend code will work in the test mode corresponding to given string. The STR test modes are defined in the same way as for hibernation, so please refer to Section 1 for more information about them. In particular, the "core" test allows you to test everything except for the actual invocation of the platform firmware in order to put the system into the sleep state.

Among other things, the testing with the help of /sys/power/pm_test may allow you to identify drivers that fail to suspend or resume their devices. They should be unloaded every time before an STR transition.

Next, you can follow the instructions at http://en.opensuse.org/s2ram to test the system, but if it does not work "out of the box", you may need to boot it with "init=/bin/bash" and test s2ram in the minimal configuration. In that case, you may be able to search for failing drivers by following the procedure analogous to the one described in section 1. If you find some failing drivers, you will have to unload them every time before an STR transition (ie. before you run s2ram), and please report the problems with them.