The Linux Microcode Loader

Authors: Fenghua Yu <fenghua.yu@intel.com>
Borislav Petkov <bp@suse.de>

The kernel has a x86 microcode loading facility which is supposed to provide microcode loading methods in the OS. Potential use cases are updating the microcode on platforms beyond the OEM End-Of-Life support, and updating the microcode on long-running systems without rebooting.

The loader supports three loading methods:

Early load microcode

The kernel can update microcode very early during boot. Loading microcode early can fix CPU issues before they are observed during kernel boot time.

The microcode is stored in an initrd file. During boot, it is read from it and loaded into the CPU cores.

The format of the combined initrd image is microcode in (uncompressed) cpio format followed by the (possibly compressed) initrd image. The loader parses the combined initrd image during boot.

The microcode files in cpio name space are:

on Intel:
    kernel/x86/microcode/GenuineIntel.bin
on AMD :
    kernel/x86/microcode/AuthenticAMD.bin

During BSP (BootStrapping Processor) boot (pre-SMP), the kernel scans the microcode file in the initrd. If microcode matching the CPU is found, it will be applied in the BSP and later on in all APs (Application Processors).

The loader also saves the matching microcode for the CPU in memory. Thus, the cached microcode patch is applied when CPUs resume from a sleep state.

Here's a crude example how to prepare an initrd with microcode (this is normally done automatically by the distribution, when recreating the initrd, so you don't really have to do it yourself. It is documented here for future reference only).

```
#!/bin/bash

if [ -z "$1" ]; then
    echo "You need to supply an initrd file"
    exit 1
fi

INITRD="$1"

DSTDIR=kernel/x86/microcode
TMPDIR=/tmp/initrd

rm -rf $TMPDIR

mkdir $TMPDIR
cd $TMPDIR
mkdir -p $DSTDIR

if [ -d /lib/firmware/amd-ucode ]; then
        cat /lib/firmware/amd-ucode/microcode_amd*.bin > $DSTDIR/AuthenticAMD.bin
fi

if [ -d /lib/firmware/intel-ucode ]; then
        cat /lib/firmware/intel-ucode/* > $DSTDIR/GenuineIntel.bin
fi

find . | cpio -o -H newc >../ucode.cpio
cd ..
mv $INITRD $INITRD.orig
cat ucode.cpio $INITRD.orig > $INITRD

rm -rf $TMPDIR
```

The system needs to have the microcode packages installed into /lib/firmware or you need to fixup the paths above if yours are somewhere else and/or you've downloaded them directly from the processor vendor's site.

Late loading

There are two legacy user space interfaces to load microcode, either through /dev/cpu/microcode or through /sys/devices/system/cpu/microcode/reload file in sysfs.

The /dev/cpu/microcode method is deprecated because it needs a special userspace tool for that.

The easier method is simply installing the microcode packages your distro supplies and running:

```
# echo 1 > /sys/devices/system/cpu/microcode/reload
```

as root.

The loading mechanism looks for microcode blobs in /lib/firmware/{intel-ucode,amd-ucode}. The default distro installation packages already put them there.

Builtin microcode

The loader supports also loading of a builtin microcode supplied through the regular builtin firmware method CONFIG_EXTRA_FIRMWARE. Only 64-bit is currently supported.

Here's an example:

```
CONFIG_EXTRA_FIRMWARE="intel-ucode/06-3a-09 amd-ucode/microcode_amd_fam15h.bin"
CONFIG_EXTRA_FIRMWARE_DIR="/lib/firmware"
```

This basically means, you have the following tree structure locally:

```
/lib/firmware/
|-- amd-ucode
...
|   |-- microcode_amd_fam15h.bin
...
|-- intel-ucode
...
|   |-- 06-3a-09
...
```

so that the build system can find those files and integrate them into the final kernel image. The early loader finds them and applies them.

Needless to say, this method is not the most flexible one because it requires rebuilding the kernel each time updated microcode from the CPU vendor is available.