DeviceTree Booting

During the development of the Linux/ppc64 kernel, and more specifically, the addition of new platform types outside of the old IBM pSeries/iSeries pair, it was decided to enforce some strict rules regarding the kernel entry and bootloader <-> kernel interfaces, in order to avoid the degeneration that had become the ppc32 kernel entry point and the way a new platform should be added to the kernel. The legacy iSeries platform breaks those rules as it predates this scheme, but no new board support will be accepted in the main tree that doesn't follow them properly. In addition, since the advent of the arch/powerpc merged architecture for ppc32 and ppc64, new 32-bit platforms and 32-bit platforms which move into arch/powerpc will be required to use these rules as well.

The main requirement that will be defined in more detail below is the presence of a device-tree whose format is defined after Open Firmware specification. However, in order to make life easier to embedded board vendors, the kernel doesn't require the device-tree to represent every device in the system and only requires some nodes and properties to be present. For example, the kernel does not require you to create a node for every PCI device in the system. It is a requirement to have a node for PCI host bridges in order to provide interrupt routing information and memory/IO ranges, among others. It is also recommended to define nodes for on chip devices and other buses that don't specifically fit in an existing OF specification. This creates a great flexibility in the way the kernel can then probe those and match drivers to device, without having to hard code all sorts of tables. It also makes it more flexible for board vendors to do minor hardware upgrades without significantly impacting the kernel code or cluttering it with special cases.

Entry point

There is one single entry point to the kernel, at the start of the kernel image. That entry point supports two calling conventions:

a) Boot from Open Firmware. If your firmware is compatible with Open Firmware (IEEE 1275) or provides an OF compatible client interface API (support for "interpret" callback of forth words isn't required), you can enter the kernel with:

r5 : OF callback pointer as defined by IEEE 1275 bindings to powerpc. Only the 32-bit client interface is currently supported

r3, r4 : address & length of an initrd if any or 0

The MMU is either on or off; the kernel will run the trampoline located in arch/powerpc/kernel/prom_init.c to extract the device-tree and other information from open firmware and build a flattened device-tree as described in b). prom_init() will then re-enter the kernel using the second method. This trampoline code runs in the context of the firmware, which is supposed to handle all exceptions during that time.

b) Direct entry with a flattened device-tree block. This entry point is called by a) after the OF trampoline and can also be called directly by a bootloader that does not support the Open Firmware client interface. It is also used by "kexec" to implement "hot" booting of a new kernel from a previous running one. This method is what I will describe in more details in this document, as method a) is simply standard Open Firmware, and thus should be implemented according to the various standard documents defining it and its binding to the PowerPC platform. The entry point definition then becomes:

r3 : physical pointer to the device-tree block (defined in chapter II) in RAM

r4 : physical pointer to the kernel itself. This is used by the assembly code to properly disable the MMU in case you are entering the kernel with MMU enabled and a non-1:1 mapping.

r5 : NULL (as to differentiate with method a)

Note about SMP entry: Either your firmware puts your other CPUs in some sleep loop or spin loop in ROM where you can get them out via a soft reset or some other means, in which case you don't need to care, or you'll have to enter the kernel with all CPUs. The way to do that with method b) will be described in a later revision of this document.

Board supports (platforms) are not exclusive config options. An arbitrary set of board supports can be built in a single kernel image. The kernel will "know" what set of functions to use for a given platform based on the content of the device-tree. Thus, you should:

a) add your platform support as a _boolean_ option in arch/powerpc/Kconfig, following the example of PPC_PSERIES, PPC_PMAC and PPC_MAPLE. The latter is probably a good example of a board support to start from.

b) create your main platform file as "arch/powerpc/platforms/myplatform/myboard_setup.c" and add it to the Makefile under the condition of your `CONFIG_` option. This file will define a structure of type "ppc_md" containing the various callbacks that the generic code will use to get to your platform specific code

A kernel image may support multiple platforms, but only if the platforms feature the same core architecture. A single kernel build cannot support both configurations with Book E and configurations with classic Powerpc architectures.