Contribution ID: **58** Type: **not specified**

Consolidating representations of the physical memory

Monday 20 September 2021 09:00 (45 minutes)

Linux kernel uses several coarse representations of the physical memory consisting of [start, end, flags] structures per memory region. There is memblock that some architectures keep after boot, there is iomem_resource tree and “System RAM” nodes in that tree, there are memory blocks exposed in sysfs and then there are per-architecture structures, sometimes even several per architecture.

These abstractions are used by the memory hotplug infrastructure and kexec/kdump tools. On some architectures the memblock representation even complements the memory map and it is used in arch-specific implementation of pfn_valid().

The multiplication of such structures and lack of consistency between some of them does not help the maintainability and can be a reason for subtle bugs here and there. Moreover, the gaps between architecture specific representations of the physical memory and the assumptions made by the generic memory management about the memory layout lead to unnecessary complexity in the initialization of the core memory management structures.

The layout of the physical memory is defined by hardware and firmware and there is not much room for its interpretation. Regardless of the particular interface between the firmware and the kernel a single generic abstraction of the physical memory should suffice and a single [start, end, flags] type should be enough. There is no fundamental reason it is not possible to converge per-architecture representations of the physical memory, like e820, drmem_lmb, memblock or numa_meminfo into a generic abstraction.

Memblock seems the best candidate for being the basis for such generic abstraction. It is already supported on all architectures and it is used as the generic representation of the physical memory at boot time. Closing the gaps between per architecture structures and memblock is anyway required for more robust initialization of the memory management. Addition of simple locking of memblock data for memory hotplug, making the memblock “allocator” part discardable and a mechanism to synchronize “System RAM” resources with memblock would complete the picture.

Extending memblock with necessary funcitonality and gradually bridging the gap between the current per-architecure physical memory representation and the generic one will imporve robustness and maintainablity of the early memory management.

I agree to abide by the anti-harassment policy

I agree

Primary author: RAPOPORT, Mike (IBM)

Presenter: RAPOPORT, Mike (IBM)

Session Classification: Kernel Summit

Track Classification: Kernel Summit