x86 Topology

This documents and clarifies the main aspects of x86 topology modelling and representation in the kernel. Update/change when doing changes to the respective code.

The architecture-agnostic topology definitions are in Documentation/admin-guide/cputopology.rst. This file holds x86-specific differences/specialities which must not necessarily apply to the generic definitions. Thus, the way to read up on Linux topology on x86 is to start with the generic one and look at this one in parallel for the x86 specifics.

Needless to say, code should use the generic functions - this file is *only* here to *document* the inner workings of x86 topology.

Started by Thomas Gleixner <tglx@linutronix.de> and Borislav Petkov <bp@alien8.de>.

The main aim of the topology facilities is to present adequate interfaces to code which needs to know/query/use the structure of the running system wrt threads, cores, packages, etc.

The kernel does not care about the concept of physical sockets because a socket has no relevance to software. It's an electromechanical component. In the past a socket always contained a single package (see below), but with the advent of Multi Chip Modules (MCM) a socket can hold more than one package. So there might be still references to sockets in the code, but they are of historical nature and should be cleaned up.

The topology of a system is described in the units of:

- packages
- cores
- threads

Package

Packages contain a number of cores plus shared resources, e.g. DRAM controller, shared caches etc.

Modern systems may also use the term 'Die' for package.

AMD nomenclature for package is 'Node'.

Package-related topology information in the kernel:

- cpuinfo_x86.x86_max_cores:

  The number of cores in a package. This information is retrieved via CPUID.

- cpuinfo_x86.x86_max_dies:

  The number of dies in a package. This information is retrieved via CPUID.

- cpuinfo_x86.cpu_die_id:

  The physical ID of the die. This information is retrieved via CPUID.

- cpuinfo_x86.phys_proc_id:

  The physical ID of the package. This information is retrieved via CPUID and deduced from the APIC IDs of the cores in the package.

  Modern systems use this value for the socket. There may be multiple packages within a socket. This value may differ from cpu_die_id.

- cpuinfo_x86.logical_proc_id:

  The logical ID of the package. As we do not trust BIOSes to enumerate the packages in a consistent way, we introduced the concept of logical package ID so we can sanely calculate the number of maximum possible packages in the system and have the packages enumerated linearly.

- topology_max_packages():

  The maximum possible number of packages in the system. Helpful for per package facilities to preallocate per package information.

- cpu_llc_id:

  A per-CPU variable containing:

  - On Intel, the first APIC ID of the list of CPUs sharing the Last Level Cache
  - On AMD, the Node ID or Core Complex ID containing the Last Level Cache. In general, it is a number identifying an LLC uniquely on the system.

Cores

A core consists of 1 or more threads. It does not matter whether the threads are SMT- or CMT-type threads.

AMDs nomenclature for a CMT core is "Compute Unit". The kernel always uses "core".

Core-related topology information in the kernel:

- smp_num_siblings:

  The number of threads in a core. The number of threads in a package can be calculated by:

  ```
  threads_per_package = cpuinfo_x86.x86_max_cores * smp_num_siblings
  ```

Threads

A thread is a single scheduling unit. It's the equivalent to a logical Linux CPU.

AMDs nomenclature for CMT threads is "Compute Unit Core". The kernel always uses "thread".

Thread-related topology information in the kernel:

- topology_core_cpumask():

  The cpumask contains all online threads in the package to which a thread belongs.

  The number of online threads is also printed in /proc/cpuinfo "siblings."
- topology_sibling_cpumask():

  The cpumask contains all online threads in the core to which a thread belongs.
- topology_logical_package_id():

  The logical package ID to which a thread belongs.
- topology_physical_package_id():

  The physical package ID to which a thread belongs.
- topology_core_id();

  The ID of the core to which a thread belongs. It is also printed in /proc/cpuinfo "core_id."

System topology examples

> **Note**
>
> The alternative Linux CPU enumeration depends on how the BIOS enumerates the threads. Many BIOSes enumerate all threads 0 first and then all threads 1. That has the "advantage" that the logical Linux CPU numbers of threads 0 stay the same whether threads are enabled or not. That's merely an implementation detail and has no practical impact.

1. Single Package, Single Core:

   ```
   [package 0] -> [core 0] -> [thread 0] -> Linux CPU 0
   ```

2. Single Package, Dual Core

   a. One thread per core:

   ```
   [package 0] -> [core 0] -> [thread 0] -> Linux CPU 0
               -> [core 1] -> [thread 0] -> Linux CPU 1
   ```

   b. Two threads per core:

   ```
   [package 0] -> [core 0] -> [thread 0] -> Linux CPU 0
                           -> [thread 1] -> Linux CPU 1
               -> [core 1] -> [thread 0] -> Linux CPU 2
                           -> [thread 1] -> Linux CPU 3
   ```

   Alternative enumeration:

   ```
   [package 0] -> [core 0] -> [thread 0] -> Linux CPU 0
                           -> [thread 1] -> Linux CPU 2
               -> [core 1] -> [thread 0] -> Linux CPU 1
                           -> [thread 1] -> Linux CPU 3
   ```

   AMD nomenclature for CMT systems:

   ```
   [node 0] -> [Compute Unit 0] -> [Compute Unit Core 0] -> Linux CPU 0
                                -> [Compute Unit Core 1] -> Linux CPU 1
            -> [Compute Unit 1] -> [Compute Unit Core 0] -> Linux CPU 2
   ```

```
                                        -> [Compute Unit Core 1] -> Linux CPU 3
```

4. Dual Package, Dual Core

   a. One thread per core:

```
[package 0] -> [core 0] -> [thread 0] -> Linux CPU 0
            -> [core 1] -> [thread 0] -> Linux CPU 1

[package 1] -> [core 0] -> [thread 0] -> Linux CPU 2
            -> [core 1] -> [thread 0] -> Linux CPU 3
```

   b. Two threads per core:

```
[package 0] -> [core 0] -> [thread 0] -> Linux CPU 0
                        -> [thread 1] -> Linux CPU 1
            -> [core 1] -> [thread 0] -> Linux CPU 2
                        -> [thread 1] -> Linux CPU 3

[package 1] -> [core 0] -> [thread 0] -> Linux CPU 4
                        -> [thread 1] -> Linux CPU 5
            -> [core 1] -> [thread 0] -> Linux CPU 6
                        -> [thread 1] -> Linux CPU 7
```

   Alternative enumeration:

```
[package 0] -> [core 0] -> [thread 0] -> Linux CPU 0
                        -> [thread 1] -> Linux CPU 4
            -> [core 1] -> [thread 0] -> Linux CPU 1
                        -> [thread 1] -> Linux CPU 5

[package 1] -> [core 0] -> [thread 0] -> Linux CPU 2
                        -> [thread 1] -> Linux CPU 6
            -> [core 1] -> [thread 0] -> Linux CPU 3
                        -> [thread 1] -> Linux CPU 7
```

   AMD nomenclature for CMT systems:

```
[node 0] -> [Compute Unit 0] -> [Compute Unit Core 0] -> Linux CPU 0
                             -> [Compute Unit Core 1] -> Linux CPU 1
         -> [Compute Unit 1] -> [Compute Unit Core 0] -> Linux CPU 2
                             -> [Compute Unit Core 1] -> Linux CPU 3

[node 1] -> [Compute Unit 0] -> [Compute Unit Core 0] -> Linux CPU 4
                             -> [Compute Unit Core 1] -> Linux CPU 5
         -> [Compute Unit 1] -> [Compute Unit Core 0] -> Linux CPU 6
                             -> [Compute Unit Core 1] -> Linux CPU 7
```