CPU Idle Time Management

Copyright: © 2019 Intel Corporation
Author: Rafael J. Wysocki <rafael.j.wysocki@intel.com>

CPU Idle Time Management Subsystem

Every time one of the logical CPUs in the system (the entities that appear to fetch and execute instructions: hardware threads, if present, or processor cores) is idle after an interrupt or equivalent wakeup event, which means that there are no tasks to run on it except for the special "idle" task associated with it, there is an opportunity to save energy for the processor that it belongs to. That can be done by making the idle logical CPU stop fetching instructions from memory and putting some of the processor's functional units depended on by it into an idle state in which they will draw less power.

However, there may be multiple different idle states that can be used in such a situation in principle, so it may be necessary to find the most suitable one (from the kernel perspective) and ask the processor to use (or "enter") that particular idle state. That is the role of the CPU idle time management subsystem in the kernel, called `CPUIdle`.

The design of `CPUIdle` is modular and based on the code duplication avoidance principle, so the generic code that in principle need not depend on the hardware or platform design details in it is separate from the code that interacts with the hardware. It generally is divided into three categories of functional units: *governors* responsible for selecting idle states to ask the processor to enter, *drivers* that pass the governors' decisions on to the hardware and the *core* providing a common framework for them.

CPU Idle Time Governors

A CPU idle time (`CPUIdle`) governor is a bundle of policy code invoked when one of the logical CPUs in the system turns out to be idle. Its role is to select an idle state to ask the processor to enter in order to save some energy.

`CPUIdle` governors are generic and each of them can be used on any hardware platform that the Linux kernel can run on. For this reason, data structures operated on by them cannot depend on any hardware architecture or platform design details as well.

The governor itself is represented by a struct cpuidle_governor object containing four callback pointers, :c:member:\`enable\`, :c:member:\`disable\`, :c:member:\`select\`, :c:member:\`reflect\`, a :c:member:\`rating\` field described below, and a name (string) used for identifying it.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 52); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 52); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 52); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 52); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 52); *backlink*

Unknown interpreted text role "c:member".

For the governor to be available at all, that object needs to be registered with the `CPUIdle` core by calling

:c:func:`cpuidle_register_governor()` with a pointer to it passed as the argument. If successful, that causes the core to add the governor to the global list of available governors and, if it is the only one in the list (that is, the list was empty before) or the value of its :c:member:`rating` field is greater than the value of that field for the governor currently in use, or the name of the new governor was passed to the kernel as the value of the `cpuidle.governor=` command line parameter, the new governor will be used from that point on (there can be only one `CPUIdle` governor in use at a time). Also, user space can choose the `CPUIdle` governor to use at run time via `sysfs`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 57); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 57); *backlink*

Unknown interpreted text role "c:member".

Once registered, `CPUIdle` governors cannot be unregistered, so it is not practical to put them into loadable kernel modules.

The interface between `CPUIdle` governors and the core consists of four callbacks:

:c:member:`enable`

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 92); *backlink*

Unknown interpreted text role "c:member".

```
int (*enable) (struct cpuidle_driver *drv, struct cpuidle_device *dev);
```

The role of this callback is to prepare the governor for handling the (logical) CPU represented by the struct cpuidle_device object pointed to by the `dev` argument. The struct cpuidle_driver object pointed to by the `drv` argument represents the `CPUIdle` driver to be used with that CPU (among other things, it should contain the list of struct cpuidle_state objects representing idle states that the processor holding the given CPU can be asked to enter).

It may fail, in which case it is expected to return a negative error code, and that causes the kernel to run the architecture-specific default code for idle CPUs on the CPU in question instead of `CPUIdle` until the `->enable()` governor callback is invoked for that CPU again.

:c:member:`disable`

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 105); *backlink*

Unknown interpreted text role "c:member".

```
void (*disable) (struct cpuidle_driver *drv, struct cpuidle_device *dev);
```

Called to make the governor stop handling the (logical) CPU represented by the struct cpuidle_device object pointed to by the `dev` argument.

It is expected to reverse any changes made by the `->enable()` callback when it was last invoked for the target CPU, free all memory allocated by that callback and so on.

:c:member:`select`

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 135); *backlink*

Unknown interpreted text role "c:member".

```
int (*select) (struct cpuidle_driver *drv, struct cpuidle_device *dev,
               bool *stop_tick);
```

Called to select an idle state for the processor holding the (logical) CPU represented by the struct cpuidle_device object pointed to by the `dev` argument.

The list of idle states to take into consideration is represented by the :c:member:`states` array of struct cpuidle_state objects held by the struct cpuidle_driver object pointed to by the `drv` argument (which represents the `CPUIdle` driver to be used with the CPU at hand). The value returned by this callback is interpreted as an index into that array (unless it is a negative error code).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 117); *backlink*

Unknown interpreted text role "c:member".

The `stop_tick` argument is used to indicate whether or not to stop the scheduler tick before asking the processor to enter the selected idle state. When the `bool` variable pointed to by it (which is set to `true` before invoking this callback) is cleared to `false`, the processor will be asked to enter the selected idle state without stopping the scheduler tick on the given CPU (if the tick has been stopped on that CPU already, however, it will not be restarted before asking the processor to enter the idle state).

This callback is mandatory (i.e. the :c:member:`select` callback pointer in struct cpuidle_governor must not be `NULL` for the registration of the governor to succeed).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 133); *backlink*

Unknown interpreted text role "c:member".

:c:member:`reflect`

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 145); *backlink*

Unknown interpreted text role "c:member".

```
void (*reflect) (struct cpuidle_device *dev, int index);
```

Called to allow the governor to evaluate the accuracy of the idle state selection made by the `->select()` callback (when it was invoked last time) and possibly use the result of that to improve the accuracy of idle state selections in the future.

In addition, `CPUIdle` governors are required to take power management quality of service (PM QoS) constraints on the processor wakeup latency into account when selecting idle states. In order to obtain the current effective PM QoS wakeup latency constraint for a given CPU, a `CPUIdle` governor is expected to pass the number of the CPU to :c:func:`cpuidle_governor_latency_req()`. Then, the governor's `->select()` callback must not return the index of an indle state whose :c:member:`exit_latency` value is greater than the number returned by that function.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 147); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 147); *backlink*

Unknown interpreted text role "c:member".

CPU Idle Time Management Drivers

CPU idle time management (`CPUIdle`) drivers provide an interface between the other parts of `CPUIdle` and the hardware.

First of all, a `CPUIdle` driver has to populate the :c:member:`states` array of struct cpuidle_state objects included in the struct cpuidle_driver object representing it. Going forward this array will represent the list of available idle states that the processor hardware can be asked to enter shared by all of the logical CPUs handled by the given driver.

**System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api]`

[pm]cpuidle.rst, line 164); *backlink*

Unknown interpreted text role "c:member".

The entries in the :c:member:`states` array are expected to be sorted by the value of the :c:member:`target_residency` field in struct cpuidle_state in the ascending order (that is, index 0 should correspond to the idle state with the minimum value of :c:member:`target_residency`). [Since the :c:member:`target_residency` value is expected to reflect the "depth" of the idle state represented by the struct cpuidle_state object holding it, this sorting order should be the same as the ascending sorting order by the idle state "depth".]

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 170); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 170); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 170); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 170); *backlink*

Unknown interpreted text role "c:member".

Three fields in struct cpuidle_state are used by the existing `CPUIdle` governors for computations related to idle state selection:

:c:member:`target_residency`

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 186); *backlink*
>
> Unknown interpreted text role "c:member".
>
> Minimum time to spend in this idle state including the time needed to enter it (which may be substantial) to save more energy than could be saved by staying in a shallower idle state for the same amount of time, in microseconds.

:c:member:`exit_latency`

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 191); *backlink*
>
> Unknown interpreted text role "c:member".
>
> Maximum time it will take a CPU asking the processor to enter this idle state to start executing the first instruction after a wakeup from it, in microseconds.

:c:member:`flags`

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 199); *backlink*
>
> Unknown interpreted text role "c:member".
>
> Flags representing idle state properties. Currently, governors only use the `CPUIDLE_FLAG_POLLING` flag which is set if the

given object does not represent a real idle state, but an interface to a software "loop" that can be used in order to avoid asking the processor to enter any idle state at all. [There are other flags used by the `CPUIdle` core in special situations.]

The :c:member:`enter` callback pointer in struct cpuidle_state, which must not be `NULL`, points to the routine to execute in order to ask the processor to enter this particular idle state:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 201); *backlink*

Unknown interpreted text role "c:member".

```
void (*enter) (struct cpuidle_device *dev, struct cpuidle_driver *drv,
               int index);
```

The first two arguments of it point to the struct cpuidle_device object representing the logical CPU running this callback and the struct cpuidle_driver object representing the driver itself, respectively, and the last one is an index of the struct cpuidle_state entry in the driver's :c:member:`states` array representing the idle state to ask the processor to enter.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 210); *backlink*

Unknown interpreted text role "c:member".

The analogous `->enter_s2idle()` callback in struct cpuidle_state is used only for implementing the suspend-to-idle system-wide power management feature. The difference between in and `->enter()` is that it must not re-enable interrupts at any point (even temporarily) or attempt to change the states of clock event devices, which the `->enter()` callback may do sometimes.

Once the :c:member:`states` array has been populated, the number of valid entries in it has to be stored in the :c:member:`state_count` field of the struct cpuidle_driver object representing the driver. Moreover, if any entries in the :c:member:`states` array represent "coupled" idle states (that is, idle states that can only be asked for if multiple related logical CPUs are idle), the :c:member:`safe_state_index` field in struct cpuidle_driver needs to be the index of an idle state that is not "coupled" (that is, one that can be asked for if only one logical CPU is idle).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 223); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 223); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 223); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 223); *backlink*

Unknown interpreted text role "c:member".

In addition to that, if the given `CPUIdle` driver is only going to handle a subset of logical CPUs in the system, the :c:member:`cpumask` field in its struct cpuidle_driver object must point to the set (mask) of CPUs that will be handled by it.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 232); *backlink*

Unknown interpreted text role "c:member".

A `CPUIdle` driver can only be used after it has been registered. If there are no "coupled" idle state entries in the driver's :c:member:`states` array, that can be accomplished by passing the driver's struct cpuidle_driver object to :c:func:`cpuidle_register_driver()`. Otherwise, :c:func:`cpuidle_register()` should be used for this purpose.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 237); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 237); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 237); *backlink*

Unknown interpreted text role "c:func".

However, it also is necessary to register struct cpuidle_device objects for all of the logical CPUs to be handled by the given `CPUIdle` driver with the help of :c:func:`cpuidle_register_device()` after the driver has been registered and :c:func:`cpuidle_register_driver()`, unlike :c:func:`cpuidle_register()`, does not do that automatically. For this reason, the drivers that use :c:func:`cpuidle_register_driver()` to register themselves must also take care of registering the struct cpuidle_device objects as needed, so it is generally recommended to use :c:func:`cpuidle_register()` for `CPUIdle` driver registration in all cases.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 243); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 243); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 243); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 243); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 243); *backlink*

Unknown interpreted text role "c:func".

The registration of a struct cpuidle_device object causes the `CPUIdle sysfs` interface to be created and the governor's `->enable()` callback to be invoked for the logical CPU represented by it, so it must take place after registering the driver that will handle the CPU in question.

`CPUIdle` drivers and struct cpuidle_device objects can be unregistered when they are not necessary any more which allows some resources associated with them to be released. Due to dependencies between them, all of the struct cpuidle_device objects representing CPUs handled by the given `CPUIdle` driver must be unregistered, with the help of :c:func:`cpuidle_unregister_device()`, before calling :c:func:`cpuidle_unregister_driver()` to unregister the driver. Alternatively, :c:func:`cpuidle_unregister()` can be called to unregister a `CPUIdle` driver along with all of the struct cpuidle_device objects representing CPUs handled by it.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 258); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 258); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 258); *backlink*

Unknown interpreted text role "c:func".

`CPUIdle` drivers can respond to runtime system configuration changes that lead to modifications of the list of available processor idle states (which can happen, for example, when the system's power source is switched from AC to battery or the other way around). Upon a notification of such a change, a `CPUIdle` driver is expected to call :c:func:`cpuidle_pause_and_lock()` to turn `CPUIdle` off temporarily and then :c:func:`cpuidle_disable_device()` for all of the struct cpuidle_device objects representing CPUs affected by that change. Next, it can update its :c:member:`states` array in accordance with the new configuration of the system, call :c:func:`cpuidle_enable_device()` for all of the relevant struct cpuidle_device objects and invoke :c:func:`cpuidle_resume_and_unlock()` to allow `CPUIdle` to be used again.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 269); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 269); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 269); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 269); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\pm\[linux-master][Documentation][driver-api][pm]cpuidle.rst`, line 269); *backlink*

Unknown interpreted text role "c:func".