Linaro connect Vancouver 2018

Idle loop reordering : what does it solve and how to get benefit ?

The ‘power nightmare’

- Well known issue when the prediction fails
  - Sleep time is much longer than predicted
  - Shallow state selected (huge power consumption on x86 with the polling state)
  - Nothing makes the state to exit

- Proposed solution
  - Don’t disable the tick if a shallow is selected
    - In the worst case, the tick will wake up the CPU
    - Prediction computed again

- Status
  - Merged in v4.18 upstream kernel
  - Merged in Android-4.14

Results

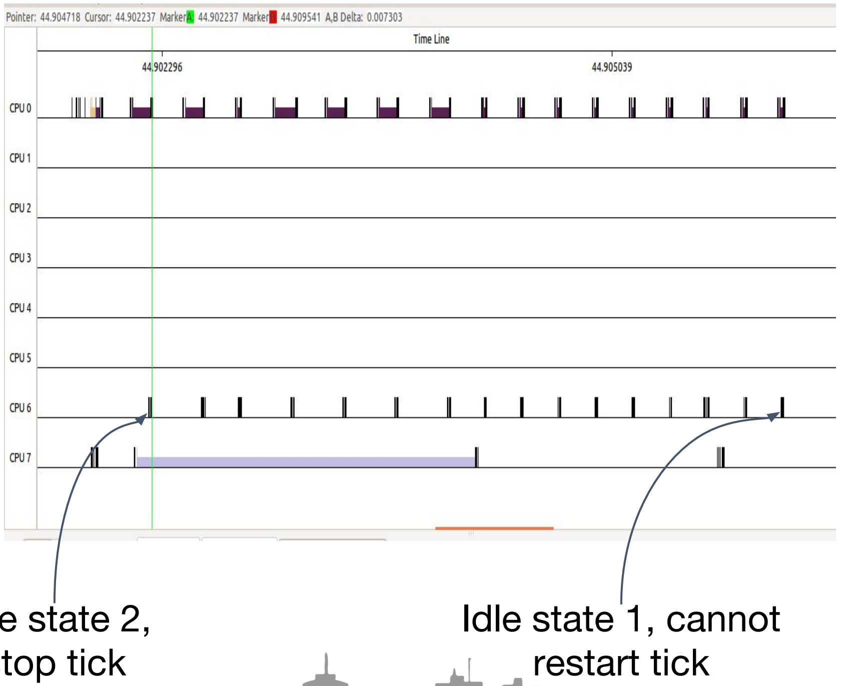
- Prediction failure + shallow state considerably reduced
- More opportunity to go to a deeper idle state
- No performance regression with:
  - Mplayer
  - Jankbench
  - Exoplayer (audio and video)
  - Geekbench
- 30% power improvement with the audio workload
- However ...

Issue 1: Cannot restart tick after stopping it

If the tick has been stopped in idle loop, it’s impossible to restart it in the loop. This issue can be corrected until the next wake up with exiting idle loop.

This issue can be reproduced by:

- Send IPIs with 3ms interval for 10 times so that train typical interval pattern, menu governor will predict shallow state based on typical interval;
- On Hikey the mmc driver can trigger > 10 times timers but without context switching.



Pointer: 44.904718 Cursor: 44.902237 MarkerA 44.902237 MarkerB 44.909541 A,B Delta: 0.007303

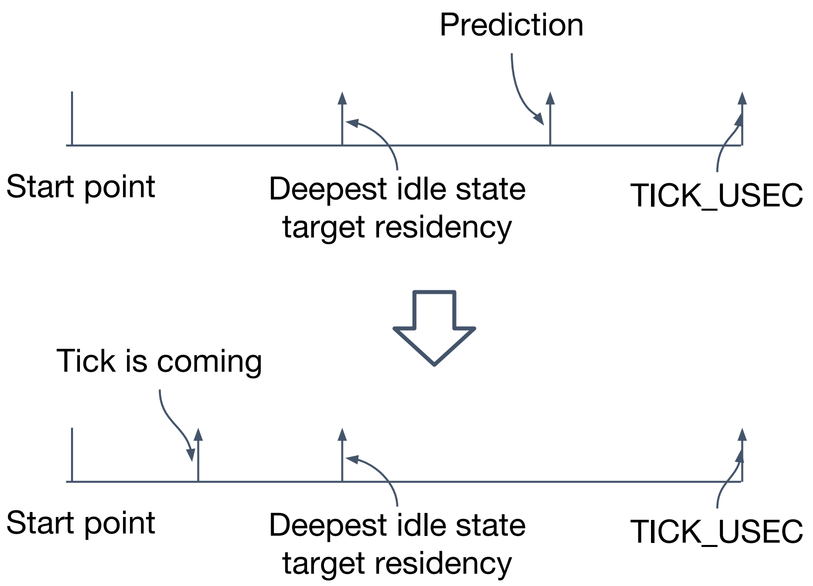
Issue 1: Current mainline code

```
static void cpuidle_idle_call(void)
{
        [...]

                bool stop_tick = true;

                /*
                 * Ask the cpuidle framework to choose a convenient idle state.
                 */
                next_state = cpuidle_select(drv, dev, &stop_tick);

                if (stop_tick || tick_nohz_tick_stopped())
                        tick_nohz_idle_stop_tick();
                else
                        tick_nohz_idle_retain_tick();

                rcu_idle_enter();

                entered_state = call_cpuidle(drv, dev, next_state);
                /*
                 * Give the governor an opportunity to reflect on the outcome
                 */
                cpuidle_reflect(dev, entered_state);
```

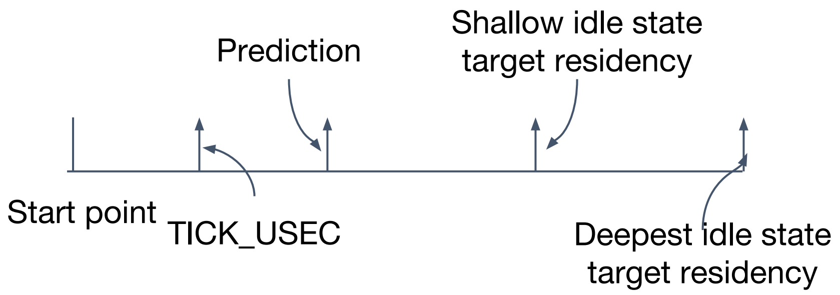
If the tick has been stopped, even the governor tells to enable it, but idle loop will ignore this request.

Issue 2: Wrong decision to stop tick

The first possible wrong decision is menu governor don't stop the tick if the prediction is less than TICK_USEC but the prediction is longer than deepest idle state:


Prediction
Start point
Deepest idle state target residency
TICK_USEC
Tick is coming
Start point
Deepest idle state target residency
TICK_USEC

The second possible wrong decision is menu governor can select one shallow state with stopping tick so this can occur power nightmares:


Prediction
Shallow idle state target residency
Start point
TICK_USEC
Deepest idle state target residency

Issue 3: Absent idle prediction modeling

In the menu idle governor, we should can use metrics `data->predicted_us` and `measured_duration` for idle prediction modeling; especially, we can create the modeling for prediction accuracy:

```
prediction_accuracy =
  stddev(measured_duration - prediction)
```

Currently the metrics are broken for profiling due `data->predicted_us` is broken for tick stopped case and `measured_duration` doesn't alway contain the real measured value and is assigned to big compensation; furthermore, the modeling doesn't take account tick event into prediction.

Issues fixed in mainline

- Commit 757ab15c3f49 : Retain tick when shallow state is selected
- Commit 87c9fe6ee495 : Avoid selecting shallow states with stopped tick

- Still areas of improvements
  - The code itself needs some … clarification
  - Better accuracy of the prediction
  - Irq timings based prediction WIP

Prediction based on irq timings

- Previous approach dropped
  - Self learning mechanism - auto adaptive
  - Efficient but expensive and refused by scheduler maintainer

- Currently implementing a new proof of concept based on irq timings
  - Complex approach
  - Need experimentation

- Need tools to investigate the cpuidle subsystem behavior:
  - Correct predictions vs over or under estimated
  - Source of wakeup
  - Idlestat does that, migrate to eBPF