Custom CPU Governor Example

Incorporating Battery Charge State

Bill Gatliff
bgat@billgatliff.com

Freelance Embedded Systems Developer

Functional Overview

Battery charge awareness:

- Decrease cpufreq when battery is low
- Increase available cpufreq when battery is charged
- Maximize cpufreq when connected to charger

Functional Overview

Caveats:

- No uniform battery state API
- What if we have more than one battery?
- What if we want to dial down features too?
- Is CPU really the largest power consumer?

That’s why it’s an “example” :-)

pm8921_bms_get_percent_charge()

```
struct pm8921_bms_chip *the_chip;

int pm8921_bms_get_percent_charge(void)
{
  if (!the_chip)
    return -EINVAL; /* uninitialized */
  return report_state_of_charge(the_chip);
}
EXPORT_SYMBOL_GPL(pm8921_bms_get_percent_charge);
```

batt_gov

```
struct cpufreq_governor batt_gov = {
  .name     = "battery-pct",
  .governor = cpufreq_gov_batt,
  .owner    = THIS_MODULE,
};
...
cpufreq_register_governor(&batt_gov);
...
```

batt_gov_info

```
struct batt_gov_info batt_gov_info = {
  ...
  struct cpufreq_policy *policy;
  struct delayed_work work;
  ...
};
struct batt_gov_info batt_gov_info;
```

cpufreq_gov_batt()

```
int cpufreq_gov_batt(...)
{
  switch (event) {
  case CPUFREQ_GOV_START:
  case CPUFREQ_GOV_LIMITS:
    batt_gov_info.policy = policy;
    INIT_DELAYED_WORK_DEFERRABLE(
        &batt_gov_info.work, do_batt_gov);
    schedule_delayed_work(&batt_gov_info.work, delay);
    break;
  ...
  }
  return 0;
}
```

do_batt_gov()

```
void do_batt_gov(struct work_struct *w)
{
  struct batt_gov_info *i = container_of(w, ...);
  struct cpufreq_policy *p = i->policy;
  int pct, pmax;

  pct = pm8921_bms_get_percent_charge();
  ...
```

do_batt_gov()

```
  ...
  /* set policy max to a percentage of hardware max */
  pmax = (p->cpuinfo.max - p->cpuinfo.min) * pct;
  pmax = (pmax / 100) + p->cpuinfo.min;

  __cpufreq_driver_target(i->policy, pmax,
                          CPUFREQ_RELATION_H);

  schedule_delayed_work(w, delay);
}
```

do_batt_gov()

Still needs work:

- Synchronization!
- Missing or multiple batteries
- Is battery percentage the ONLY consideration?
- How can the user refine the policy settings?
- What about fully-discharged battery?

do_batt_gov()

```
void do_batt_gov(struct work_struct *w)
{
  ...
  if (pm8921_is_dc_chg_plugged_in()) {
    __cpufreq_driver_target(i->policy,
       p->cpuinfo.max, CPUFREQ_RELATION_H);
    ...
  }
  ...
  schedule_delayed_work(w, delay);
}
```

do_batt_gov()

```
void do_batt_gov(struct work_struct *w)
{
   ...
   if (pmax < 30)
      pmax = 30;
   ...
}
```

do_batt_gov()

```
void do_batt_gov(struct work_struct *w)
{
  mutex_lock_interruptible(&i->mutex);
  ...
  __cpufreq_driver_target(i->policy, ...);
  ...
  schedule_delayed_work(w, delay);
  mutex_unlock(&i->mutex);
}
```

do_batt_gov()

```
int cpufreq_gov_batt(...)
{
  switch(event) {
    case CPUFREQ_GOV_STOP:
      cancel_delayed_work_sync(...);
      break;
  }
  ...
}
```

do_batt_gov()

```
void do_batt_gov(struct work_struct *w)
{
  ...
  if (!is_battery_present()) {
    pmax = p->cpuinfo.max; /* ? */
  }
  ...
}
```