Documentation Ipanema

Baptiste Pires

May 31, 2024

1 Introduction

2 Structures

Dans cette section nous allons voir les différentes structures utilisées dans Ipanema et leurs utilités

2.1 `struct sched_ipanema_entity`

Nous allons commencer par la structure qui représente une entité d'ordonnancement. La structure est déclarée de cette façon :

```
struct sched_ipanema_entity {
  union {
    struct rb_node node_runqueue;
    struct list_head node_list;
  };

  /* used for load balancing in policies */
  struct list_head ipa_tasks;

  int just_yielded;
  int nopreempt;

  enum ipanema_state state;
  struct ipanema_rq *rq;

  /* Policy-specific metadata */
  void *policy_metadata;

  struct ipanema_policy *policy;
};
```

Les deux premiers champs, `struct rb_node node_runqueue` et `struct list_head node_list` sont utilisés pour maintenir une tâche dans une runqueue. Dû à l'union, on peut être dans une seule runqueue à un moment donner (soit dans une runqueue qui se sert d'un arbre rouge-noir comme CFS, ou une runqueue qui utilise une liste comme un algorithme FIFO par exemple).

Le champ `struct list_head ipa_tasks` est utilisé par les différentes politiques d'ordonnancement pendant l'équilibrage de charge. Par exemple, si on migre des tâches d'un coeur C_1 à un coeur C_2, on a besoin de retirer les tâches de la runqueue de C_1 en ayant le verrou sur celle-ci. Quand on retire les tâches on peut les ajouter à la liste `ipa_tasks`. Une fois qu'on a retirée toutes les tâches qu'on souhaite migrer, on prend le verrour sur la runqueue de C_2 et on peut itérer sur la liste que l'on vient de construire.

`int just_yield` permet de savoir si on vient d'appeler `sched_yield()` ou non.

`int nopreempt`
`enum ipanema_state state` représente l'état d'une entitté Ipanema . Les états possibles sont les suivants :

```
enum ipanema_state {
  IPANEMA_NOT_QUEUED, /* La tache n'est pas dans une runqueue */
  IPANEMA_MIGRATING, /* La tache est en train de migrer d'un coeur a un autre */
  IPANEMA_RUNNING, /* La tache est dans une runqueue et s'execute */
  IPANEMA_READY, /* La tache est dans une runqueue mais ne s'execute pas */
  IPANEMA_BLOCKED, /* La tache est bloque (I/O, lock, etc...) */
  IPANEMA_TERMINATED, /* La tache meurt (?) */
  /*
   * A special state that is equivalent to IPANEMA_READY, but makes it
   * possible to figure out that the state change came from tick().
   */
  IPANEMA_READY_TICK
};
```

`struct ipanema_rq *rq` c'est un pointeur vers la `struct ipanema_rq` sur laquelle la tâche s'exécute (ou va s'exécuter).

2.1.1 `struct ipanema rq`

Now we will take a look at the `struct ipanema_rq`. It's the structure that represents a runqueue in Ipanema , fairly simple struct made of the following fields :

```
struct ipanema_rq {
  enum ipanema_rq_type type;
  union {
    struct rb_root root;
    struct list_head head;
  };

  /*
   * cpu to which this runqueue belongs (for per_cpu runqueues).
   * For runqueues that are not per_cpu, rq->cpu > NR_CPUS.
  */
  unsigned int cpu;
  enum ipanema_state state;

  /* Number of tasks in this runqueue*/
  unsigned int nr_tasks;

  int (*order_fn)(struct task_struct *a, struct task_struct *b);
};
```

`enum ipanema_rq_type` defines what kind of structure the runqueue will use. The enum is defined as follow : `enum ipanema_rq_type { RBTREE, LIST, FIFO };`. As we can see, three types of runqueues are supported now. One with an red-black tree structure, one with a list and with a fifo list.

The union of `struct rb_root` and `struct list_head` is the root of either the tree or list of tasks in this runqueue.

`int cpu` is the id of the cpu this runqueue belongs to.

`enum ipanema_state` demander à Redha pour être sûr. Defines the state in which the tasks must be when added to this runqueue.

`unsigned int nr_tasks` The number of tasks currently in the runqueue.

`int (*order_fn)(struct task_struct *a, struct task_struct *b)` is function pointer that will be called when a task is added to the runqueue or whenever an ordering is needed. It is not mandatory for the FIFO runqueues.

That concludes the part for the runqueue.

3 Ipanema architecture

Dans cette section nous allons voir les détails de l'architecture d'Ipanema .

3.1 Les variables globales d'Ipanema

Dans Ipanema , il y a deux variables globales importantes : `struct list_head ipanema_policies` et `per_cpu(struct task_struct *, ipanema_current)`. Nous allons voir comment ces deux variables sont utilisées pour comprendre leur utilitée.

ipanema_policies La variables `struct list_head ipanema_policies` est une list chaînée qui contient toutes les politiques d'ordonnancement disponibles. Cela correspond à tous les modules qui ont été insérés en tant que politiques. Cette list est protégée par un `rwlock_t ipanema_rwlock` qui permet de la modifier/parcourir en toute sécurité.
Losrque l'on souhaite ajouter une politique, on fait appel à la fonction `ipanema_add_policy(struct ipanema_policy *)` qui va insérer la nouvelle politique dans la liste.
Nous avons maintenant inséré une politique, mais aucun thread ne s'en sert sert pour l'instant. Pour ce faire, il va falloir se servir de l'appel système `SYS_sched_setattr` (voir `tools/ipanema/ipastart.c` pour un exemple d'utilisation). En somme, il appelle la fonction `__sched_setscheduler` qui va, si la politique demandée existe, modifier le champ de la `task_struct.ipanema.policy` pour le faire pointer vers la politique souhaitée. Ensuite, cette liste sera par exemple utilisée dès que l'on a besoin de trouver une tâche à exécuter.

ipanema_current Cette variable est l'une des plus importantes. Elle est définie `per_cpu`, ce qui signifie que pour tous les CPUs, il y a une variable d'allouée. Ces variables vont stocker le processus en cours d'exécution sur le coeur auquel elles appartiennent. Pour voir cela, il faut regarder la fonction `change_state` qui gère toutes les transitions d'états d'Ipanema . On peut voir que pour la transition `x -> RUNNING` (une tâche va commencer son exécution) : `per_cpu(ipanema_current, next_cpu) = p` où `next_cpu` est le CPU sur lequel la tâche va s'exécuter et `p` ladite tâche.
Il faut noter que juste avant, la fonction `change_rq` est appelée et la tâche `p` aura été retirée de sa runqueue où elle attendait à l'état `IPANEMA_READY`.

4 Ecrire un scheduler dans Ipanema

Dans cette section nous allons voir comment écrire un scheduler sous la forme d'un module en utilisant Ipanema .