Разработка и встраивание планировщика в ОС Linux

Студент: Степанов Даниил

Санкт-Петербургский политехнический университет Петра Великого

2 июня 2017 г.

Планирование

Политики планирования ядра 2.6:

- SCHED_NORMAL
- SCHED_FIFO
- SCHED_RR

SCHED_EDF

SCHED_EDF - это планировщик ЦП в реальном времени, который:

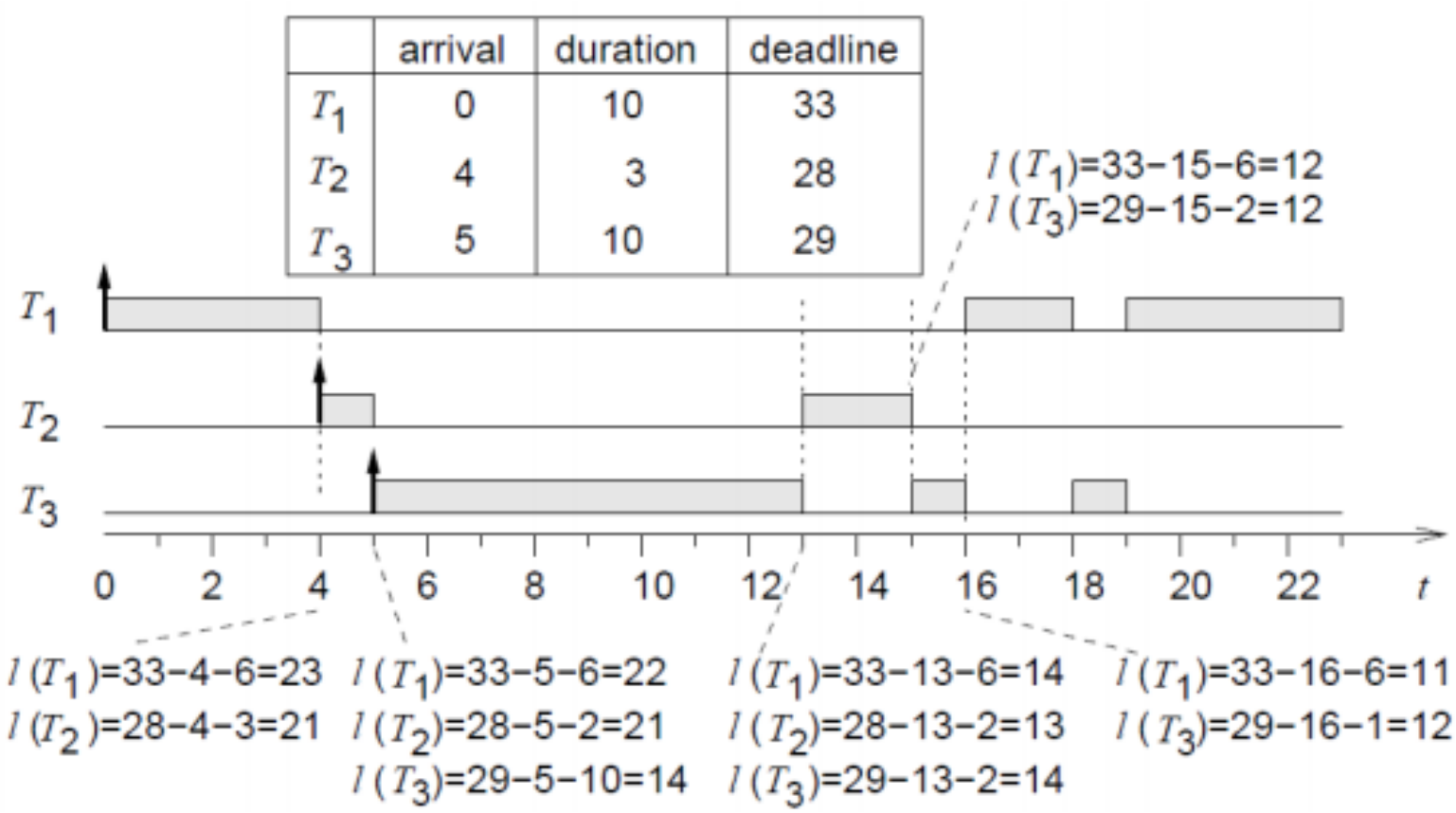
- позволяет указать временные ограничения для каждой задачи
- построен вокруг концепции резервирования ресурсов
- основан на хорошо известных алгоритмах планирования

SCHED_EDF

Задача характеризуется следующими параметрами:

- время готовности
- наихудшее время выполнения
- дедлайн

Earliest deadline first



Для чего полезна SCHED_EDF

Политика устанавливается для достижения двух результатов:

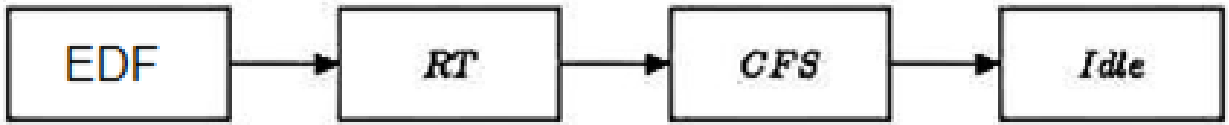
- гаратии времени
- ограничения сроков

Реализация EDF

```
struct edf_task{
  struct rb_node edf_rb_node;
  unsigned long long
       absolute_deadline;
  struct list_head edf_list_node;
  struct task_struct *task;
};
```

```
struct edf_rq {
  struct rb_root edf_rb_root;
  struct list_head edf_list_head;
  atomic_t nr_running;
};
```

Чтобы добавить новую политику планирования в ядро Linux, необходимо создать новый модуль



Согласно модульным правилам планирования, каждый модуль должен реализовывать набор функций, указанных в структуре sched_class

```
const struct sched_class edf_sched_class = {
  .next        = &rt_sched_class,
  .enqueue_task    = enqueue_task_edf,
  .dequeue_task    = dequeue_task_edf,

  .check_preempt_curr = check_preempt_curr_edf,

  .pick_next_task    = pick_next_task_edf,
  .put_prev_task     = put_prev_task_edf,

  .set_curr_task           = set_curr_task_edf,
  .task_tick     = task_tick_edf,
};.
```

Изменения в sched.c для включения файла, реализующего модуль EDF:

```
#include "sched_rt.c"
#ifdef CONFIG_SCHED_DEBUG
#include "sched_debug.c"
#endif
#ifdef  CONFIG_SCHED_EDF_POLICY
#include "sched_edf.c"
#endif

#ifdef  CONFIG_SCHED_EDF_POLICY
  #define sched_class_highest (&edf_sched_class)
#else
  #define sched_class_highest (&rt_sched_class)
#endif
```

Чтобы задать требуемые параметры планирования задачи, необходимо изменить структуру данных struct sched_param:

```
struct sched_param {
  int sched_priority;

#ifdef  CONFIG_SCHED_EDF_POLICY
  unsigned int  edf_id;
  unsigned long long deadline;
#endif
 };
```

Реализация SCHED_EDF

Дополнительные изменения:

- Включение политики в исходную функцию ядра rt_policy
- sched_setscheduler
- _setscheduler

Сборка и встраивание ядра

- С помощью утилиты wget загружаем исходный код ядра
- Распаковываем архив tar –xjvf linux-2.6.24.tar.bz2
- Создаем конфигурационный файл из текущего системного конфигурационного файла: sudo cp /boot/config-2.6.24 .config
- Меняем параметр extraversion в Makefile, чтобы отличать собираемое ядро от других версий EXTRAVERSION= -edf
- Применяем патч: patch -p1 < ../sched.patch
- Компилируем ядро:
  sudo make oldconfig
  sudo make-kpkg –initrd kernel_image 2>../errors
- Начинается компиляция ядра, и если все идет хорошо, создается сжатый образ ядра
- Установим скомпилированную версию ядра, сгенерированную предыдущим шагом sudo dpkg -i kernel_image-2.6.24_xxxx.deb

Тестирование

- Был применен патч и встроено новое ядро
- Написан код, устанавливающий процессу новую политику планировки
- Написан код, устанавливающий процессу политику планировки SCHED_DEADLINE в ядре 3.16

```
[1455]
missed deadlines   = 2856
missed periods     = 821
Total adjustments = 48064 us
deadline    : 1000 us
runtime     : 400 us
nr_periods : 6368
```

```
[1460]
missed deadlines   = 3802
missed periods     = 1026
Total adjustments = 20386 us
deadline    : 1000 us
runtime     : 400 us
nr_periods : 5188
```