Diseño y Desarrollo de Sistemas de Tiempo Real

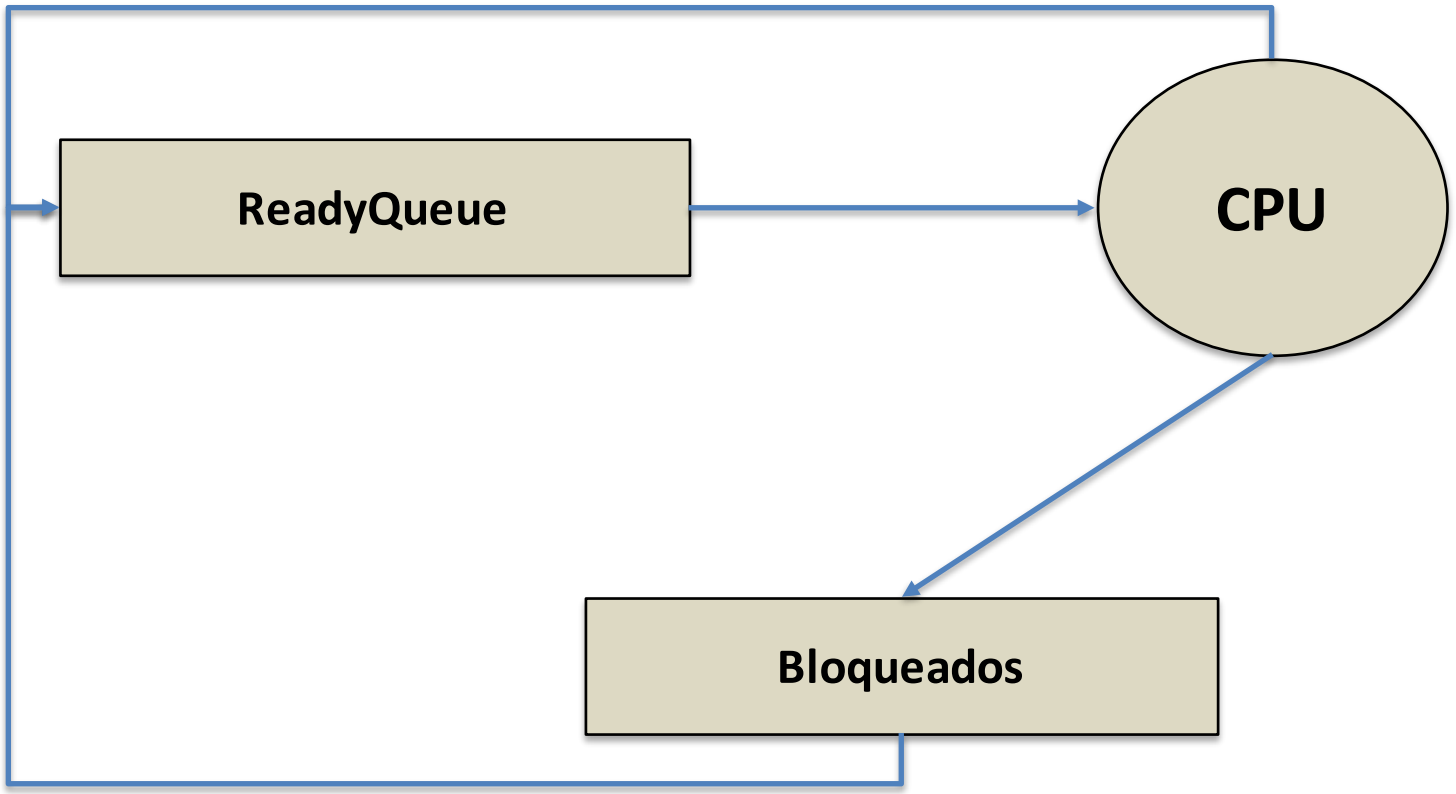
Alfons Crespo

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Diseño Planificador

- Estructura de datos
- Planificador
- Resultados

Diseño Planificador



Estructuras de datos

- ReadyQueue:
  - Cola de tareas preparadas para ser ejecutadas
  - Ordenadas por un criterio:
    - Prioridad
    - Plazo absoluto
  - Estructura Heap
    - La clave puede ser una tupla:
    - El valor puede ser una tupla

Estructuras de datos

- ReadyQueue Ejemplo
  - Añadir a la cola
    - heappush(readyQueue, ((prio), (tid, period, rdead, adead, wcet, 0, nActiv + 1)))
    - La clave es la prioridad
    - El valor: (id de tarea, periodo, plazo relativo, plazo absoluto, wcet, tiempo ejecutado, número de activación)
  - Sacar de la cola:
    - ((prio), (cTaskId, period, rdead, adead, wcet, texec, nActiv)) = heappop(readyQueue)
  - Leer la primera posición sin extraerlo:
    - ((prio), (cTaskId, period, rdead, adead, wcet, texec, nActiv)) = readyQueue[0]
  - Saber el tamaño:
    - len(readyQueue)

Estructuras de datos

- Cola de procesos bloqueados:
  - Cola de tareas esperando para ser activadas cuando cumplan su periodo (periodicas)
  - Ordenadas por un criterio:
    - Tiempo de activación (release time)
  - Estructura Heap
    - La clave puede ser una tupla:
    - El valor puede ser una tupla
  - La denominamos como releaseTime

Estructuras de datos

- ReleaseTime Ejemplo
  - Clave: (tiempo de activación, prioridad/plazo absoluto)
  - Añadir a la cola
    - Al inicio:
      - heappush(releaseTime, ((0, prio), (tid, period, rdead, adead, wcet, 0, nActiv)))
    - Despues:
      - nxtActiv = nActiv * period
      - adead = nxtActiv + rdead
      - heappush(releaseTime, ((nxtActiv, prio), (cTaskId, period, rdead, adead, wcet, 0, nActiv)))
  - Leer la primera posición sin extraerlo:
    - (time, prio) = releaseTime[0][0]
    - (tid, period, rdead, adead, wcet, texec, nActiv) = releaseTime[0][1]
  - Sacar de la cola
    - elem = heappop(releaseTime)
  - Saber el tamaño:
    - len(releaseTime)

Planificador: inicialización

- Cabecera: def schedRun(ticks):
- Construir una lista con las tareas
  - Global: todas las tareas de todas las particiones
  - Local: una por cada core con las tareas de las particiones asociadas al core
- Prioridades:
  - Ordenar la lista por periodos
    - Si el primer parámetro de la lista es el periodo: tList.sort()
    - Recorrer la lista asignandole prioridades crecientes (1 mayor prioridad, n menor prioridad)
    - Una vez asignada la prioridad, añadirlas a la cola releaseTime con tiempo 0

Planificador

- Inicialización
- Clock = 0
- Mientras clock < nticks
  - Actualizar la ReadyQueue desde la ReleaseTime al tiempo de release
  - Seleccionar la tarea a ejecutar (primera de la ready queue)
  - Incrementar tiempo de ejecución
  - Si termina:
    - Se añade a la releaseTime con tiempo = siguiente activacion
  - Si no termina:
    - Se añade a la readyQueue con valores actualizados
  - Incrementar clock

Planificador: inicialización

- Cabecera: def schedRun(ticks):
- Construir una lista con las tareas
  - Global: todas las tareas de todas las particiones
  - Local: una por cada core con las tareas de las particiones asociadas al core
- EDF:
  - Ordenar la lista por plazos relativos
    - Si el primer parámetro de la lista es el plazo: tList.sort()
    - Añadirlas a la cola releaseTime con tiempo 0

Planificador: ejecutar

Codigo del ejemplo

Trazador

```
##Module: Utils
import random

Traces = {}
mcores = 0
Activations = {}
last = ""
lastTask = ""

def traceInit(ncores):
    global last
    mcores = ncores
    for i in range(ncores):
        Traces[i] = []
    Activations = {}
    last = ""

def traceExecBegin(ncore, time, taskId):
    global last, lastTask
    list = Traces[ncore]
    if (last == "B"):
        list.append((lastTask, "TE", time))
    list.append((taskId, "TB", time))
    #print "TB", time, taskId
    Traces[ncore] = list
    lastTask = taskId
    last = "B"
    if (Activations.has_key(taskId)):
        Activations[taskId] = Activations[taskId] + 1
    else:
        Activations[taskId] = 1

def traceExecEnd(ncore, time, taskId):
    global last, lastTask

    list = Traces[ncore]
    list.append((taskId, "TE", time))
    Traces[ncore] = list
    #print "TE", time, taskId
    last = "E"


def traceShow(ncore):
    res = ""
    trz = Traces[ncore]
    for i in range(len(trz)):
        if ("TB" in trz[i]):
            startTime = trz[i][2]
            res += trz[i][0] + " [" + str(startTime)
        elif ("TE" in trz[i]):
            endTime = trz[i][2]
             duration = endTime - startTime
            res += ", " + str(duration) + "]\n"
    print res
    print Activations
```

trabajos

1. Planificador global multicore Prioridades fijas RM
2. Planificador global multicore Prioridades dinámicas EDF
3. Planificador local multicore Prioridades fijas RM
4. Planificador local multicore Prioridades dinámicas EDF
5. Planificador local multicore con:
    1. N cores con Prioridades dinámicas EDF
    2. M-n cores con Prioridades fijas RM

Medidas de eficiencia:

tiempo de simulación (sin mensajes)