Multiprocessor Resource Sharing Protocol

June 6, 2014

Aim

Develope a schedulability compatible protocol

Response-Time Analysis with PCP/SRP for single-processor systems

$$R_i = C_i + B_i + I_i$$

- $I_i = \sum_{\tau_j \in hp(i)} \lceil \frac{R_i}{T_j} \rceil C_j$
- $B_i = max\{\hat{c}, \hat{b}\}$
- $C_i = WCET_i + \sum_{r^j \in F(\tau_i)} n_i c^j$

System and Task Model

- Fully partitioned system
- Fixed Priorities scheduler
- Sporadic task model
- m identical processors

Resources

- Local resources can be accessed from the same processor
- Global resources can be accessed from all processors
- Accessed under mutual exclusion
- Serialization of the parallel accesses to the global resources

Model - 1

The main aim of the protocol is to bound the global resource's side effect

Response-Time Analisy can be used "as is"

- A job can be blocked at most one time and before its execution
- The job spins until when it gets the access to the resource

Model - 2

In the local processor the combination of PCP/SRP and a spin-based protocol guarantees one single request per cpu at the same time

All resources are assigned a set of ceiling, one per processor

At the global level, the requests are queued in global FIFO queue and the access to the resource is granted when the request reaches the head of the queue

We need a mechanism to bound the time a job waits to get the access, the lock holder can migrate when preempted and executes in the processors where jobs are busy waiting

Implementation

LITMUS-RT provides a set of callback to handle the events in the real time system. We need only a subset of these functions.

We need a mechanism that notify the protocol when a processor is available to execute the lock holder.

MrsP needs to handle a set of events to make partitioned fixed priority scheduler compatible with schedulability analisies:

- schedule
- context switch
- resource's access request
- resource release

Implementation - LITMUS-RT callbacks

pfp scheduler: pfp_schedule

- detect when lock holder is preempted
- keep the processor idle when necessary

pfp scheduler: pfp_finish_switch

- force the lock holder to migrate
- notify the protocol that a processor became available for a migration

mrsp resource: pfp_mrsp_open

- computes the set of ceiling

Implementation - LITMUS-RT callbacks

mrsp resource: pfp_mrsp_lock

- reise task priority to the local ceiling
- add the request to the global FIFO queue
- wake up lock holder if necessary
- busy wait

mrsp resource: pfp_mrsp_unlock

- restore task priority
- remove the request from the head of the queue
- force the job to migrate back

Taskset - 1

Task	Processor	Release time	WCET	C.S. length	Prio
τ_1	P_1	0	3	2	low
τ_2	P_1	4	2	0	high
τ_3	P_2	0	3	2	low
τ_4	P_2	4	1	0	high
τ_5	P_3	0	3	2	low
τ_6	P_3	4	2	0	high

Run-time - 1

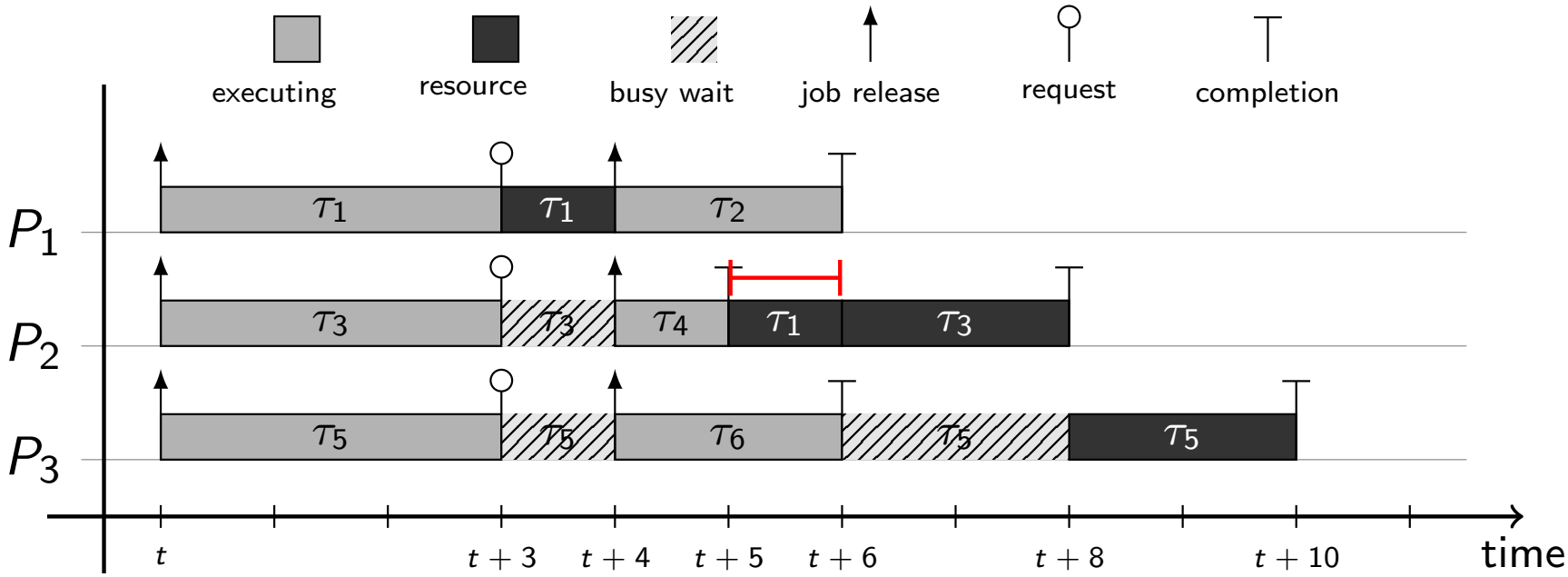

Figure: 3 processors, 6 tasks, 1 global resource

Taskset - 2

Task	Processor	Release time	WCET	C.S. length	Prio
τ_1	P_1	0	2	3	middle
τ_2	P_1	3	2	0	high
τ_3	P_1	0	0	3	low
τ_4	P_2	0	2	3	low
τ_5	P_2	3	1	0	high

Run-time - 2

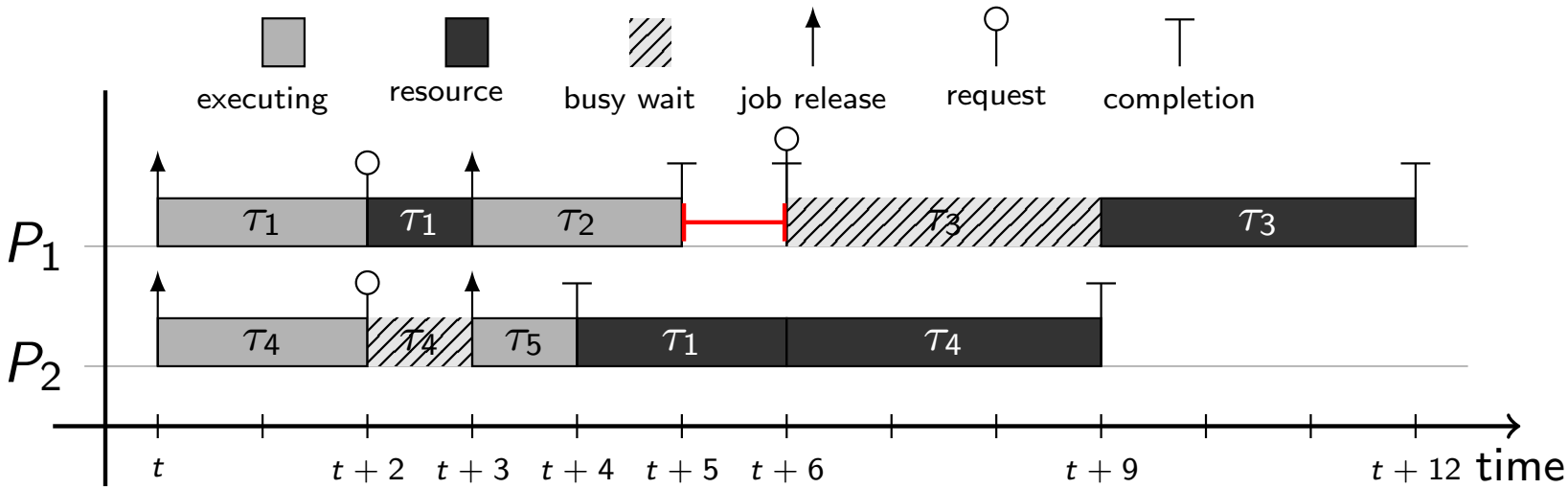

Figure: 2 processors, 5 tasks, 1 global resource

Experiments

Performance comparison between MrsP and a protocol based on SRP and Simple Lock

Finding a relation between the resource's critical section length and the MrsP's overhead, determining whether the protocol is useful or not.

Comparison among different implementations, e.g. deciding when to force the lock holder to migrate back

- as soon as possible
- as late as possible