Response-Time Analysis of Fork/Join Tasks in Multiprocessor Systems

CISTER - Research Center in
Real-Time & Embedded Computing Systems

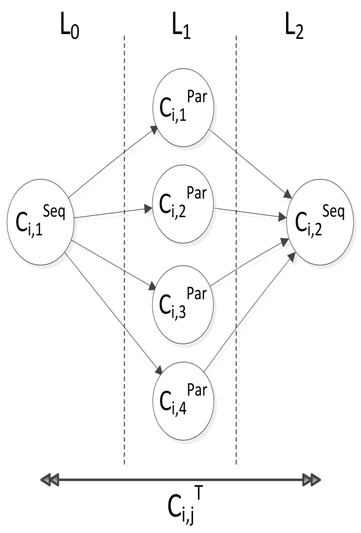
Cláudio Maia, Luís Nogueira, Luis Miguel Pinho, and Marko Bertogna
{clrrm, lmn, lmp}@isep.ipp.pt, {marko.bertogna}@unimore.it

Motivation & Context

- Multicore platforms and intra-task parallelism
- Most multiprocessor real-time scheduling results are focused on sequential tasks
- Parallel models are still restrictive in nature (static, less general)
- Task decomposition enables the application of known schedulability analysis techniques, but parallelisation is not supported by default

Fork/Join

- Emerged as a promising technique for parallel programming
- Programmers may divide applications into smaller blocks that can be assigned to CPUs
- A job is a sequence of several regions - sequential and parallel
- There is no restriction on the number of parallel blocks in a parallel region



Contributions & Objectives

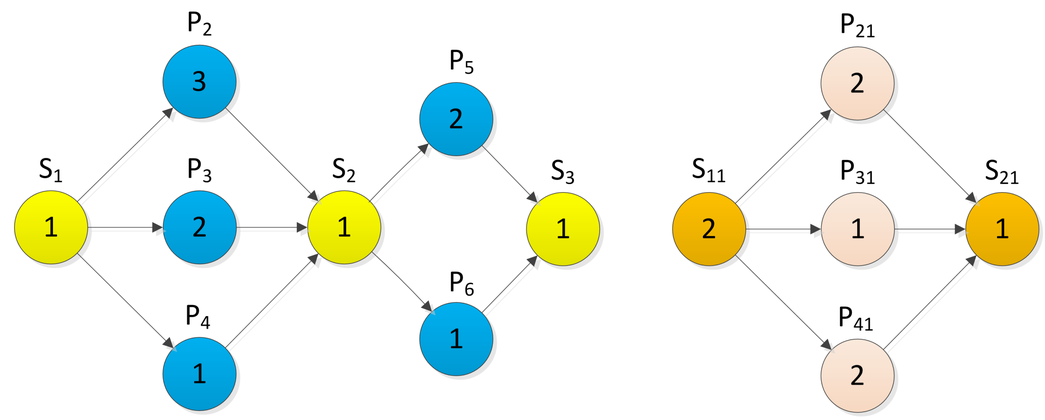
- Schedulability analysis of fork/join tasks from a response-time perspective
- Capture real-time behaviour of parallel applications which can be modelled as real-time fork-join tasks
- Scheduling of tasks that inherently present a parallel behaviour

Future work

- Schedulability analysis of fork/join tasks considering the largest interference possible
- Extend previous analysis to consider the following restrictions
- Extend work to strict fork/join tasks where nested parallelism is allowed as well as other general parallel task models
- Precedence constraints among tasks
- Migration costs and preemption costs

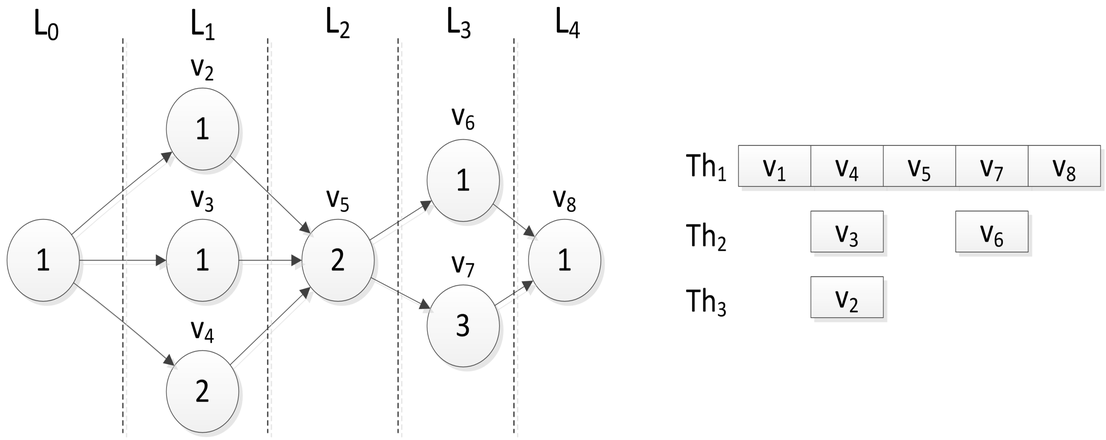
Using slack

- Slack can be used to refine the computation of the worst-case interfering workload for other tasks
- By examining the response-time of each parallel stage of a task it is possible to achieve a tighter estimation of the interference imposed by such a task



Using decomposition

- Decompose the parallel task into a set of threads
- The main thread (Th1) is composed of all the sequential parts and the worst-case parallel part of each level Li
- The remaining threads (Th2 and Th3) are composed of sets of parallel jobs belonging to different levels in the graph
- Once the task decomposition into different threads is accomplished, classical methods to bound the interfering contribution of each sequential thread can be applied



CISTER, project FCOMP-01-0124-FEDER-022701
REGAIN, project FCOMP-01-0124-FEDER-020447
SENODs, project FCOMP-01-0124-FEDER-012988
PhD Grant SFRH/BD/88834/2012

Co-financed by:

CISTER Research Centre/INESC-TEC
ISEP, Polytechnic Institute of Porto
Rua Dr. Antº Bernardino de Almeida, 431
4200-072 PORTO Portugal
tel: +351-228340502
fax: +351-228340509
http://www.cister.isep.ipp.pt
cister-info@isep.ipp.pt