CHALMERS
UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG

Dependable Real-Time Systems

Lecture #8

Risat Pathan

Department of Computer Science and Engineering
Chalmers University of Technology

Multiprocessor scheduling

How are tasks assigned to processors?

- Static assignment
  - The processor(s) used for executing a task are determined before system is put in mission ("off-line")
  - Approaches: partitioned scheduling, guided search, non-guided search, ...
- Dynamic assignment
  - The processor(s) used for executing a task are determined during system operation "on-line"
  - Approach: global scheduling

Multiprocessor scheduling

How are tasks allowed to migrate?

- Partitioned scheduling (no migration!)
  - Each instance of a task must execute on the same processor
  - Equivalent to multiple uniprocessor systems!
- Guided search & non-guided techniques
  - Depending on migration constraints, a task may or may not execute on more than one processor
- Global scheduling (full migration!)
  - A task is allowed to execute on an arbitrary processor (sometimes even after being preempted)

Partitioned scheduling

Complexity of schedulability analysis for partitioned scheduling: (Leung & Whitehead, 1982)

The problem of deciding whether a task set (synchronous or asynchronous) is schedulable on m processors with respect to partitioned scheduling is NP-complete in the strong sense.

Consequence:

There cannot be any pseudo-polynomial time algorithm for finding an optimal partition of a set of tasks unless P = NP.

Partitioned scheduling

For any task-to-processor assignment algorithm, the following steps are generally followed:

1. Specify an order for the tasks are to be considered for assignment.

2. Specify an order of the processors to attempt to allocate the task.

3. A task is successfully allocated upon a processor if it fits on the processor.

   Uniprocessor schedulability test is applied: a task fits on a processor if the task's with all the tasks previously allocated to the processor passes the test.

4. The algorithm declares success if all tasks are successfully allocated; otherwise, it declares failure.

We now consider partitioned scheduling of tasks where D=T for each task and there are m identical processors

Partitioned scheduling

Bin-packing algorithms:

Rate-Monotonic-First-Fit (RMFF): (Dhall and Liu, 1978)

- Let the processors be indexed as $\mu_1, \mu_2, \mu_3, \ldots$
- Assign the tasks in the order of increasing periods (that is, RM order).
- For each task τ_i, choose the lowest previously-used j such that τ_i, together with all tasks that have already been assigned to processor μ_j, can be feasibly scheduled according to the utilization-based RM-feasibility test.
- Processors are added if needed for RM-schedulability.

Partitioned scheduling: Fixed Priority (Test 1)

Guarantee bound for RMFF (Oh & Baker, 1998):

The utilization guarantee bound U_{RMFF} for a system with m processors using the RMFF scheduling policy is

$$m\left(2^{1/2}-1\right) \leq U_{RMFF} \leq (m+1)/\left(1+2^{1/(m+1)}\right)$$

Note: $$\left(2^{1/2}-1\right) \approx 0.41$$

Thus: task sets whose utilization do not exceed ≈ 41% of the total processor capacity is always RMFF-schedulable.

Implication of the following

$$m\left(2^{1/2}-1\right) \leq U_{RMFF} \leq (m+1)/\left(1+2^{1/(m+1)}\right)$$

Any system of tasks with total utilization $U \leq m(\sqrt{2}-1)$ is schedulable by RMFF.

For any $m \geq 2$ there is a task system with $U = (m+1)/(1+2^{1/(m+1)})$ that cannot be scheduled upon m processors using RMFF scheduling.

Partitioned scheduling: Fixed Priority (Test 2 and Test 3)

Guarantee bound for RMFF (Lopez et al 2003):

Test 2: The utilization guarantee bound of RMFF for a system with m processors and n tasks with total utilization U is :

$$U \leq (m-1)\left(\sqrt{2}-1\right) + (n-m+1)(2^{\frac{1}{n-m+1}}-1)$$

Test 3: The utilization guarantee bound of RMFF for a system with m processors and n tasks with total utilization U where each task's utilization is at most α is given as follows:

$$U \leq (m-1)(2^{\frac{1}{(\beta+1)}}-1)\beta + (n-\beta(m-1))(2^{\frac{1}{(n-\beta(m-1))}}-1)$$

where $\beta = \lfloor \frac{1}{\log_2(\alpha+1)} \rfloor$

Partitioned scheduling: Fixed Priority

Summary of guarantee bound tests for RMFF :

Test 1: $U \leq m\left(\sqrt{2}-1\right)$

Test 2: $U \leq (m-1)\left(\sqrt{2}-1\right)+(n-m+1)\left(2^{\frac{1}{n-m+1}}-1\right)$

Test 3: $U \leq (m-1)(2^{\frac{1}{(\beta+1)}}-1)\beta+(n-\beta(m-1))(2^{\frac{1}{(n-\beta(m-1))}}-1)$

where $\beta = \lfloor\frac{1}{\log_2(\alpha+1)}\rfloor$

Partitioned Scheduling: EDF Priority

Lopez et al. (2004) considered different combination of tasks order and processors order

Factor 1 (Tasks order). In what order are the tasks considered for assignment?

- **Decreasing Utilization (DU):** the tasks are considered in non-increasing order of their utilizations
- **Increasing Utilization (IU):** the tasks are considered in non-decreasing order of their utilizations
- **Random (R):** the tasks are considered in arbitrary order.

Partitioned Scheduling: EDF Priority

Lopez et al. (2004) considered different combination of tasks order and processors order

Factor 2 (Processors order). When a task is considered for assignment, to which processor does it get assigned?

- **First-fit (FF):** the task is assigned to the first processor on which it fits.
- **Worst-fit (WF):** *the task* is assigned to the processor with the maximum remaining capacity.
- **Best-fit (BF):** The task is assigned to the processor with the minimum remaining capacity

Partitioned Scheduling: EDF Priority

Lopez et al. (2004) considered nine different combination of tasks order and processors order

FFDU, FFIU, FFR
WFDU, WFIU, WFR
BFDU, BFIU, BFR

Given a selection of Factor 1 and Factor 2, the Liu and Layland's utilization bound test for preemptive EDF uniprocessor scheduling is applied to check if a task fits on the target processor.

Partitioned Scheduling: EDF Priority (Schedulability Test)

Approach 1: Successful tasks-to-processors assignment implies schedulability

Observation: Schedulability can be determined by actually doing the task-to-processors assignment.

Approach 2: There is a utilization-bound test that imply that a successful task-to-processors assignment must exist.

Observation: Schedulability can be determine WITHOUT actually doing the task-to-processors assignment.

Partitioned Scheduling: EDF Priority (Utilization Bound Based Test)

FFDU,FFIU, FFR
WFDU, WFIU, WFR
BFDU, BFIU, BFR

A lower bound: Given that the utilization of each task is no more than $\boldsymbol{\alpha}$, the utilization bound of each of the nine algorithms is NOT smaller than $m-(m-1)\alpha$ where m is the number of processors.

Proof (Page 41, BBB): If a task τ_i with utilization u_i cannot be assigned to any processor, it must be the case that each processor already has been allocated tasks with total utilization strictly greater than $(1 - u_i)$. The total utilization of all the tasks (including τ_i) is no smaller than $m(1 - u_i) + u_i \geq m-(m-1)\alpha$

Partitioned Scheduling: EDF Priority (Utilization Bound Based Test)

FFDU,FFIU, FFR
WFDU, WFIU, WFR
BFDU, BFIU, BFR

An upper bound: Given that the utilization of each task is no more than $\boldsymbol{\alpha}$, the utilization bound of each of the nine algorithms is NOT larger than $\frac{\beta m+1}{\beta+1}$ where $\boldsymbol{\beta} = \lfloor \frac{1}{\alpha} \rfloor$ where m is the number of processors.

Proof (Page 41-42 in BBB)

BBB (in Canvas): S. Baruah, M. Bertogna and G. Buttazzo, *Multiprocessor Scheduling for Real-Time Systems*, Springer, 2015, ISBN 978-3-319-08695-8.

Partitioned Scheduling: EDF Priority (Utilization Bound Based Test)

FFDU,FFIU, FFR
WFDU, WFIU, WFR
BFDU, BFIU, BFR

WFIU and WFR: If $\mathbf{U} \leq m - (m-1)\alpha$, then all the tasks are successfully assigned to m processors.

Note that if α is allowed to be 1, the utilization bound is 1 regardless of how many processors are used.

FFDU, FFIU, FFR, WFDU, BFDU, BFIU, BFR: If $\mathbf{U} \leq \frac{\boldsymbol{\beta} \boldsymbol{m}+\mathbf{1}}{\boldsymbol{\beta}+\mathbf{1}}$ where $\boldsymbol{\beta} = \left\lfloor \frac{\mathbf{1}}{\boldsymbol{\alpha}} \right\rfloor$, then all the tasks are successfully assigned to m processors.

Note that if α is allowed to be 1, the utilization bound is $\frac{\boldsymbol{m}+\mathbf{1}}{\mathbf{2}}$.

Task Splitting

Task Splitting

Background

- Global and partitioned method cannot guarantee system utilization more than 50% for all task sets
  - —Partitioned scheduling has task assignment step.
  - —Task assignment to processors is generally done with a bin-packing algorithm.

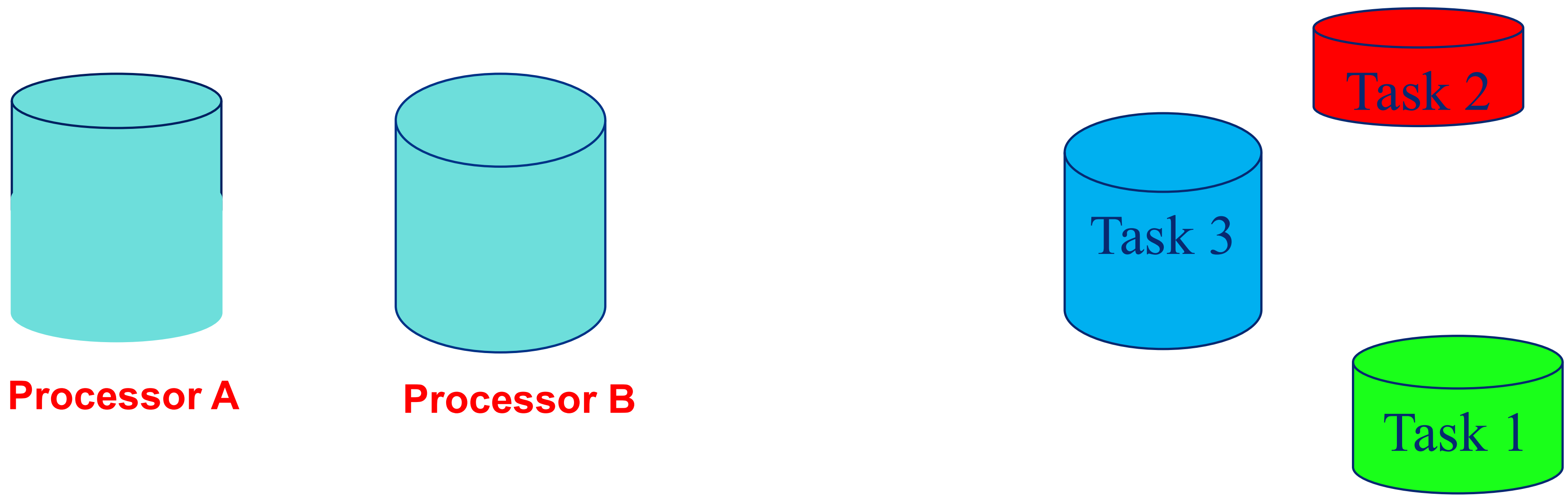
Task Splitting

Background (cont.)

- A *variation of partitioned scheduling* using task-splitting approach can achieve more than 50% system utilization for all task sets.

- History: task-splitting for static-priority were first proposed in July 2009 at CMU

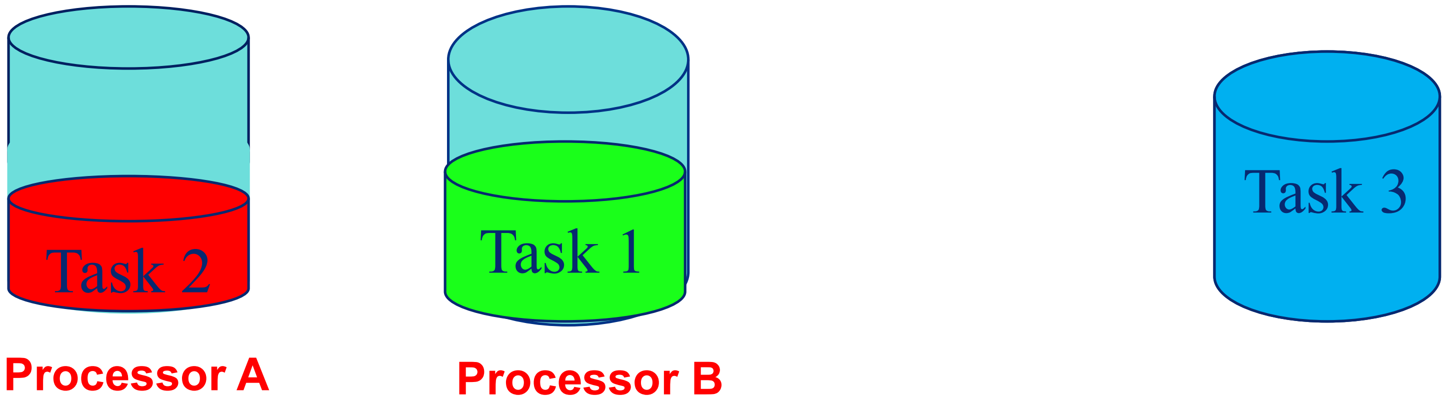
Traditional Partitioned Scheduling



We assume Task 2, Task 1 and Task 3 be the ordering of the tasks to assign to the processors A and B.

Size of each task is proportional to the utilization of the task.

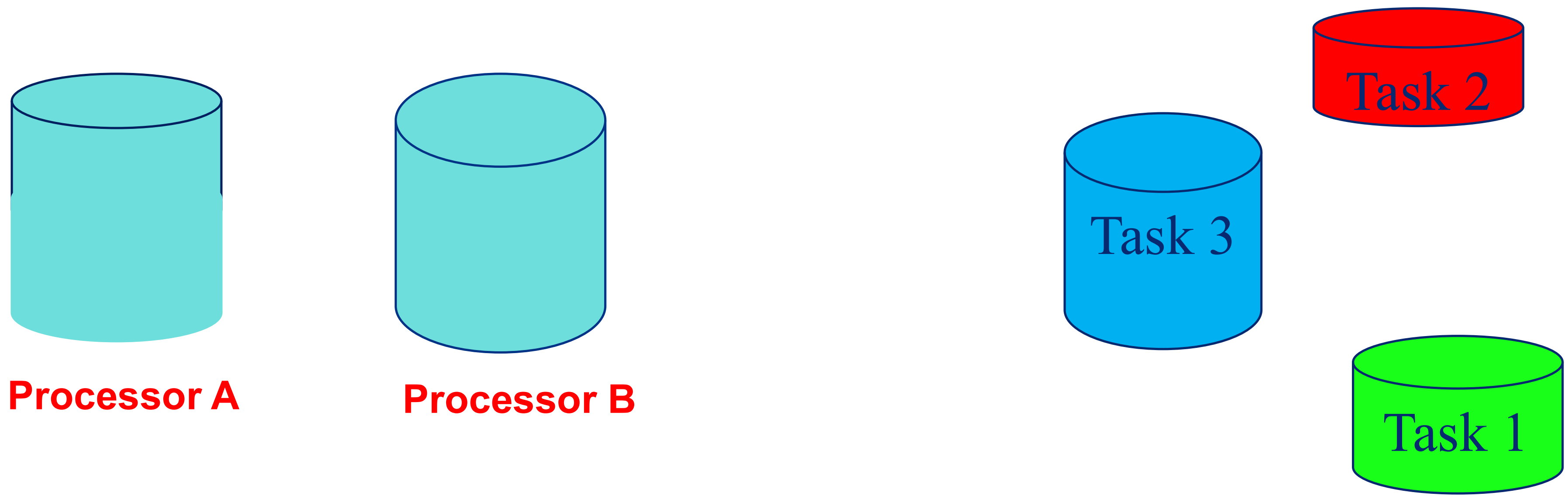
Traditional Partitioned Scheduling



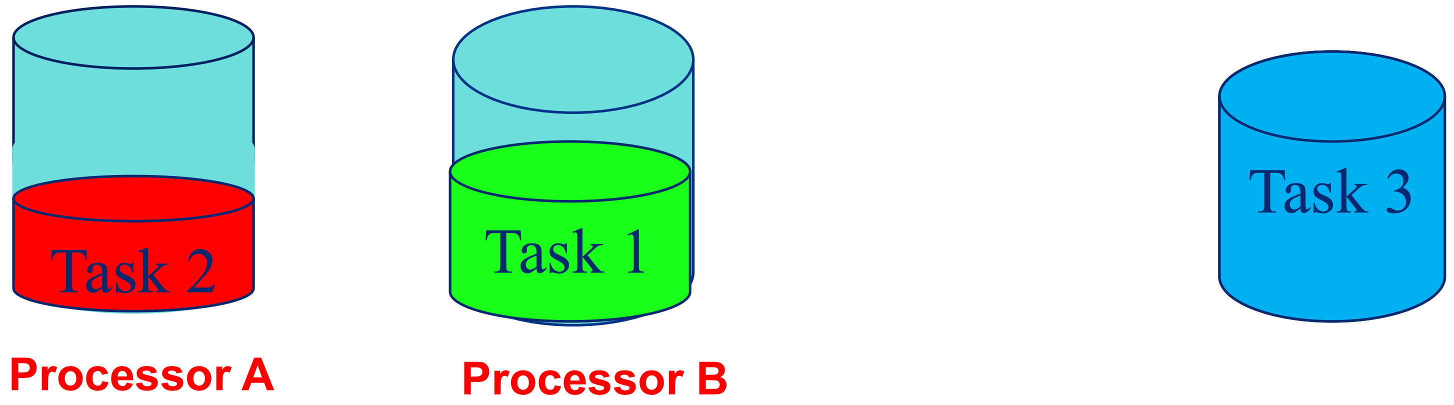
Partition Fails!

Task 3 cannot be assigned to any processor because size of Task 3 is too large

Task-Splitting Partitioned Scheduling



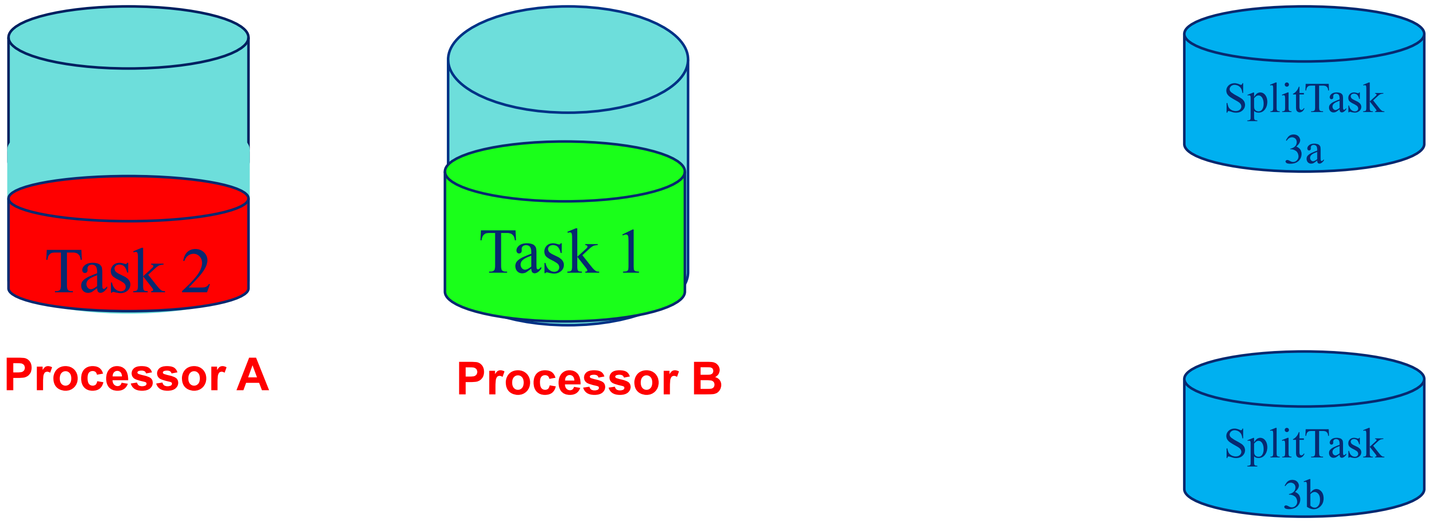
Task-Splitting Partitioned Scheduling



Different subtasks of Task 3 can be assigned to different processors.

To construct the subtasks, we split Task 3.

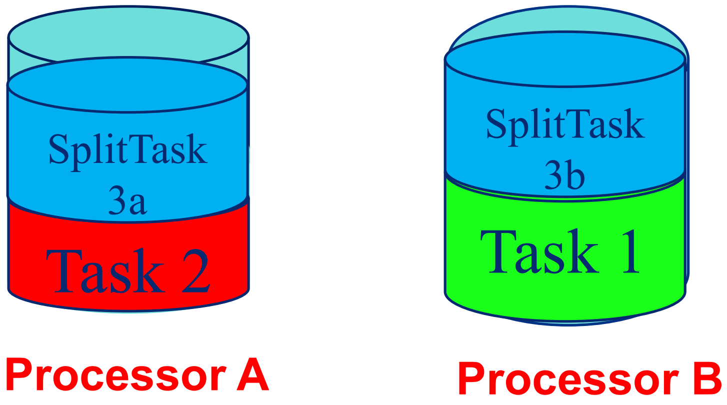
Task-Splitting Partitioned Scheduling



Different subtasks of Task 3 can be assigned to different processors.

To construct the subtasks, we split Task 3.

Task-Splitting Partitioned Scheduling



Partition Success!

Challenges in Task-Splitting

- **How to design the task assignment algorithm?**
  - How many splits of each task?
  - How many tasks to split?
  - How to ensure that subtasks of a split task do not execute in parallel?

- **How to find the guarantee bound for given task assignment algorithm?**

Some Results on Task Splitting

- ECRTS 2009, CMU: Utilization bound 65%
  - Unsorted version: 60%
  - Number of split tasks is (m-1)
  - A task can be splitted in (m-1) parts
- IPDPS 2009, CHALMERS (Our Work):
  - Utilization bound 55.2%
  - Number of split tasks is m/2
  - A task can be splitted in at most 2 parts
- RTAS 2010, UPPSALA
  - (Sorting) Utilization bound 69.3%
  - Number of split tasks is (m-1)
  - A task can be splitted in (m-1)parts

End of lecture #8