A framework for hierarchical scheduling on multiprocessors: from application requirements to run-time allocation

Giuseppe Lipari, Enrico Bini

RETIS lab, Scuola Superiore Sant'Anna — Pisa

December 2, 2010

Hierarchical scheduling

In **hierarchical scheduling**, each application is assigned a fraction of the system resources, and has its own local scheduler

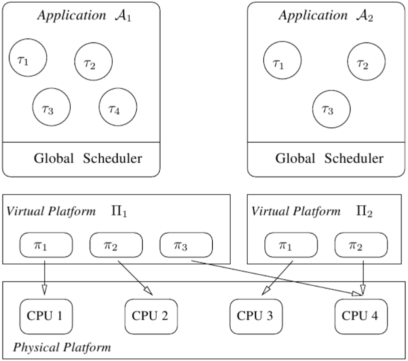
Hierarchical scheduling methodologies are useful to:

- provide temporal isolation and timing guarantees in open systems,
- enable a component-based approach to schedulability analysis;
- reduce the complexity of medium to large-sized applications
- allow application-specific schedulers (also called local schedulers)

In the hierarchical scheduling model, the computational requirement of an application is described by a temporal interface

System model

- An application consists of a set of n independent sporadic tasks
- Each application is executed upon an Virtual Platform



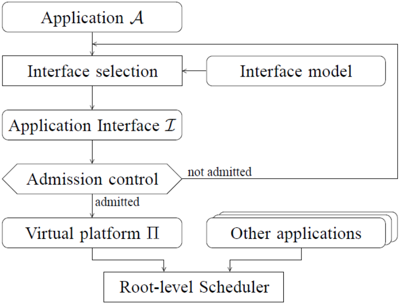
- A virtual platform is modelled by a set of virtual processors $\{\pi_1, \ldots, \pi_m\}$
- Virtual processors can be implemented by reservations, time partitions, etc.
- Each virtual processor is statically assigned to a physical processor
- More than one virtual processor may be allocated on the same physical processor

Design methodology

Problems:

A) How to specify virtual platform parameters (Interface)

- Interface specification model (periodic, bounded-delay, EDP, etc.)
- Parameter selection



B) Run-time allocation

- admission control is executed to see if the virtual platform can be accommodated
- virtual processors are prepared starting from the interface parameters, and allocated to physical processors

Contributions

We propose a framework for designing hierarchical scheduling systems that covers all phases of the design

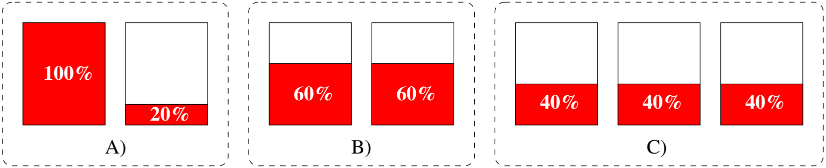
- a novel interface model, called bounded-delay multipartition (**BDM**) interface
- a schedulability test for an application that can be used to derive interface parameters
- an allocation policy, called **fluid bestfit**

We demonstrate by experiments that, our allocation policy performs better that existing policies

Basic idea

Suppose we need to allocate 120% bandwidth for our application

Some possibilities:

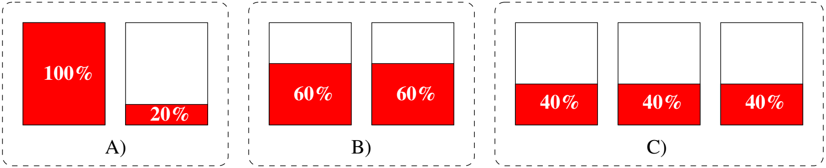


Which one is the best?

Basic idea

Suppose we need to allocate 120% bandwidth for our application

Some possibilities:



Which one is the best?

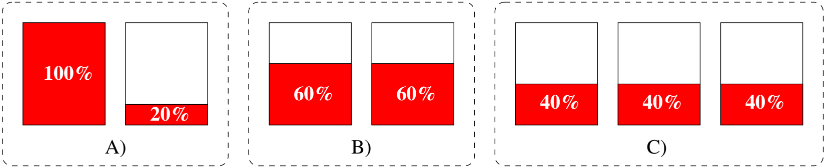
From the application point of view, platform **A)** is better

- *in most cases* it is easier to schedule tasks on such a platform

Basic idea

Suppose we need to allocate 120% bandwidth for our application

Some possibilities:



Which one is the best?

From the application point of view, platform **A)** is better

- *in most cases* it is easier to schedule tasks on such a platform

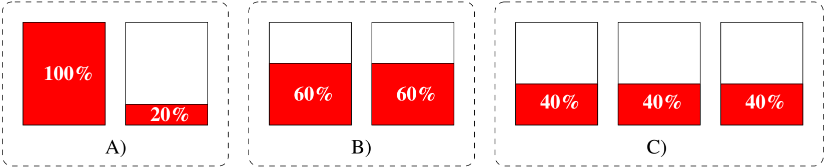
From the VP allocation point of view,

- Platform **C)** is a better candidate in most cases (smaller pieces are easier to allocate)
- The goal should be to use the least number of physical processors

Basic idea

We want to take advantage of this trade-off to add flexibility

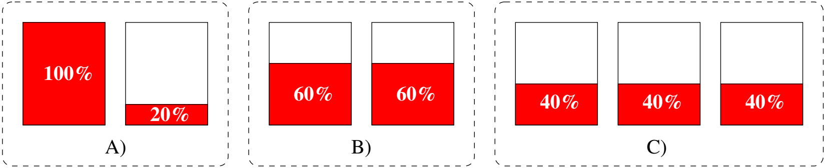
1) Application schedulability. We want to demonstrate that:



Basic idea

We want to take advantage of this trade-off to add flexibility

1) Application schedulability. We want to demonstrate that:

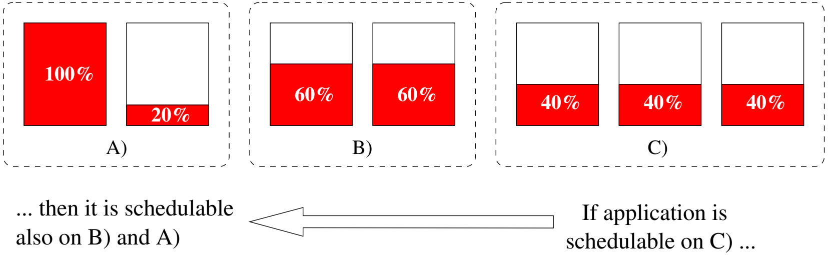


If application is
schedulable on C) ...

Basic idea

We want to take advantage of this trade-off to add flexibility

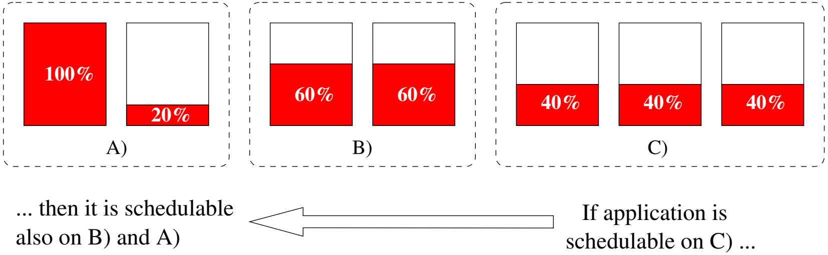
1) Application schedulability. We want to demonstrate that:



Basic idea

We want to take advantage of this trade-off to add flexibility

1) Application schedulability. We want to demonstrate that:

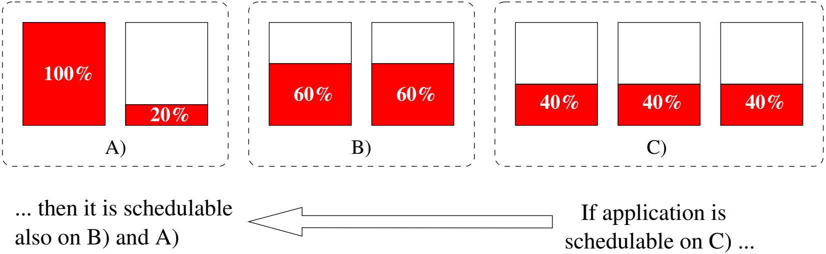


We will see that unfortunately such property does not hold for all interface models

Basic idea

We want to take advantage of this trade-off to add flexibility

1) Application schedulability. We want to demonstrate that:



We will see that unfortunately such property does not hold for all interface models

2) Platform instantiation and allocation. We want to derive an on-line allocation algorithm that, starting from the easier platform **C)**, derives the "best" platform among the compliant ones

Interfaces

A platform interface $\mathcal{I}$ is a predicate on the values that the virtual platform parameters may have

Examples of interface specifications are:

- *all platforms with an overall bandwidth of 2.5*
- *all platforms in which one virtual processor has a bandwidth of at least 0.8*

Hence,

- an interface $\mathcal{I}$ yields naturally the set of all virtual platforms that are compliant with it, called $\mathbf{\Pi}(\mathcal{I})$

Interfaces

A platform interface $\mathcal{I}$ is a predicate on the values that the virtual platform parameters may have

Examples of interface specifications are:

- *all platforms with an overall bandwidth of 2.5*
- *all platforms in which one virtual processor has a bandwidth of at least 0.8*

Hence,

- an interface $\mathcal{I}$ yields naturally the set of all virtual platforms that are compliant with it, called $\mathbf{\Pi}(\mathcal{I})$

An interface specification model is just a template for specifying interfaces

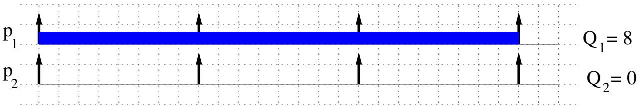
- The model also tells us how to derive a platform that is compliant with the interface.

Multiprocessor Periodic Resource Model

In the MPR model (Shin, Easwaran and Lee [1]), an interface is specified by (P, Q, m):

- All virtual processors are implemented by periodic reservations, and all have the same period P
- All of them share a cumulative budget Q
- It does not matter the exact budget of a virtual processor, as long as the sum is Q
- m is the maximum level parallelism (max number of reservations)

Example: $(P = 8, Q = 8, m = 2)$

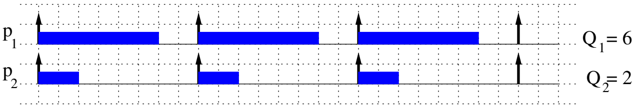


Multiprocessor Periodic Resource Model

In the MPR model (Shin, Easwaran and Lee [1]), an interface is specified by (P, Q, m):

- All virtual processors are implemented by periodic reservations, and all have the same period P
- All of them share a cumulative budget Q
- It does not matter the exact budget of a virtual processor, as long as the sum is Q
- m is the maximum level parallelism (max number of reservations)

Example: $(P = 8, Q = 8, m = 2)$

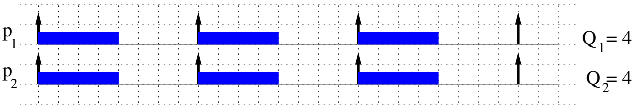


Multiprocessor Periodic Resource Model

In the MPR model (Shin, Easwaran and Lee [1]), an interface is specified by (P, Q, m):

- All virtual processors are implemented by periodic reservations, and all have the same period P
- All of them share a cumulative budget Q
- It does not matter the exact budget of a virtual processor, as long as the sum is Q
- m is the maximum level parallelism (max number of reservations)

Example: $(P = 8, Q = 8, m = 2)$



Scheduling analysis

We need to perform schedulability analysis of an application on a set of resource reservations

Parallel Supply Function (PSF) (Bini et al. [2])

- An extension of the single processor supply bound function (**sbf**)
- A set of functions $\{Y_k\}_{k=1}^{m}$,
- $Y_k(t)$ is the minimum amount of resource provided in any interval of length t by at most k virtual processors.

Given a virtual platform (set of reservations), we can compute $\{Y_k\}_{k=1}^{m}$

An application is schedulable on a given platform if, for every interval of length t, $k \leq m$ exists such that the application requirement in any interval of length t does not exceed $Y_k(t)$.

Worst-case platform

An application is guaranteed on an interface $\mathcal{I}$ if it is guaranteed on all platforms that are compliant with the interface $\mathbf{\Pi}(\mathcal{I})$

- Remember that an interface $\mathcal{I}$ yields a set of compliant platforms $\mathbf{\Pi}(\mathcal{I})$

We say that Π^{wc} is a worst-case platform of interface $\mathcal{I}$ when

- If an application is guaranteed on Π^{wc}, then it is guaranteed on any $\mathbf{\Pi}(\mathcal{I})$

Worst-case platform

An application is guaranteed on an interface $\mathcal{I}$ if it is guaranteed on all platforms that are compliant with the interface $\mathbf{\Pi}(\mathcal{I})$

- Remember that an interface $\mathcal{I}$ yields a set of compliant platforms $\mathbf{\Pi}(\mathcal{I})$

We say that Π^{wc} is a worst-case platform of interface $\mathcal{I}$ when

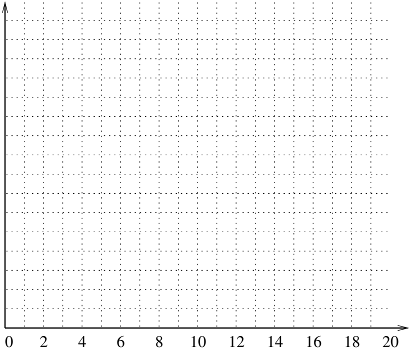
- If an application is guaranteed on Π^{wc}, then it is guaranteed on any $\mathbf{\Pi}(\mathcal{I})$

WARNING: the worst-case platform may not exist for a MPR interface!!

- If periods are not aligned, **there is no worst-case platform** Π^{wc}

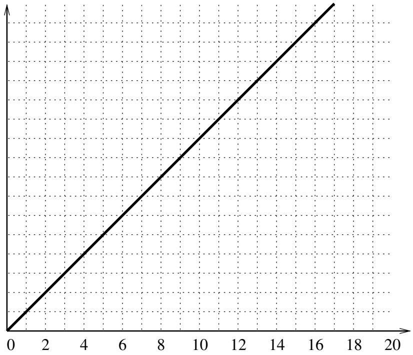
Example

- Consider platform $(Q = 8, P = 8, M = 2)$
- The graph shows $Y_2(t)$



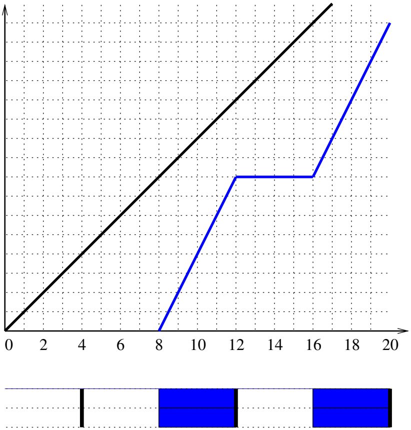
Example

- Consider platform $(Q = 8, P = 8, M = 2)$
- The graph shows $Y_2(t)$
- The black line is $Y_2(t)$ for the **best** platform (one virtual processor with $Q = P = 8$)



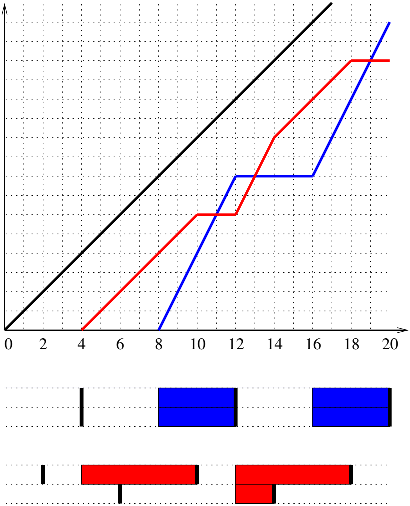
Example

- Consider platform $(Q = 8, P = 8, M = 2)$
- The graph shows $Y_2(t)$
- The black line is $Y_2(t)$ for the **best** platform (one virtual processor with $Q = P = 8$)
- The blue line is $Y_2(t)$ for two virtual processors with $Q_1 = Q_2 = 4$



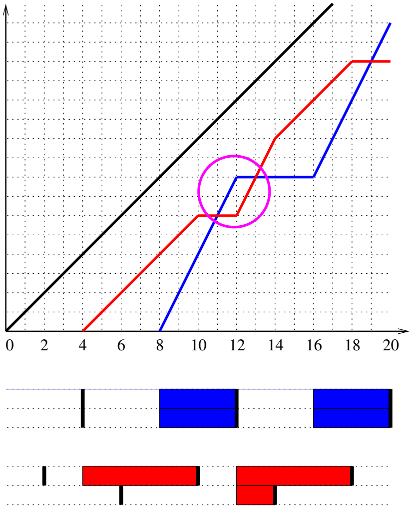
Example

- Consider platform $(Q = 8, P = 8, M = 2)$
- The graph shows $Y_2(t)$
- The black line is $Y_2(t)$ for the **best** platform (one virtual processor with $Q = P = 8$)
- The blue line is $Y_2(t)$ for two virtual processors with $Q_1 = Q_2 = 4$
- The red line is $Y_2(t)$ for the case of $Q_1 = 6$ and $Q_2 = 2$



Example

- Consider platform $(Q = 8, P = 8, M = 2)$
- The graph shows $Y_2(t)$
- The black line is $Y_2(t)$ for the **best** platform (one virtual processor with $Q = P = 8$)
- The blue line is $Y_2(t)$ for two virtual processors with $Q_1 = Q_2 = 4$
- The red line is $Y_2(t)$ for the case of $Q_1 = 6$ and $Q_2 = 2$
- There is no worst-case platform!!



Bounded Delay Multipartition Interface

To solve the previous issue, we use a different interface model, the Bounded Delay Multipartition (BDM)

- An extension of the delay-bound partition model by Mok and Feng [3].

A BDM interface is characterised by a $\mathcal{I} = (m, \Delta, [\beta_1, \ldots, \beta_m])$

- m is the maximum number of virtual processors
- Δ is the worst-case delay (i.e. the longest interval without service)
- β_k is the minimum cumulative service utilisation with k processors
- We impose $0 \leq \beta_k - \beta_{k-1} \leq 1$, and $\beta_k - \beta_{k-1} \geq \beta_{k+1} - \beta_k$

BDM

In practice:

- We approximate all $Y_k(t)$ with linear functions, with an initial *hole* of length Δ, and then a constant slope
- β_k represents the slope of $Y_k(t)$
- Notice: Δ is constant for all k

BDM

In practice:

- We approximate all $Y_k(t)$ with linear functions, with an initial *hole* of length Δ, and then a constant slope
- β_k represents the slope of $Y_k(t)$
- Notice: Δ is constant for all k

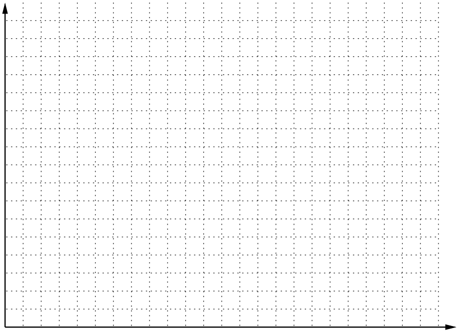
How to derive the virtual processor parameters (Q_k, P_k)?

- We compute the utilisation as follows:
  - $\alpha_1 \geq \beta_1$
  - $\alpha_k \geq \beta_k - \sum_{i=1}^{k-1} \alpha_i$
- Then $Q_k = \frac{\alpha_k \Delta}{2(1-\alpha_k)}$ and $P = \frac{\Delta}{2(1-\alpha_k)}$
- It holds:

$$\sum_{i=0}^{k} \alpha_i \geq \beta_k$$

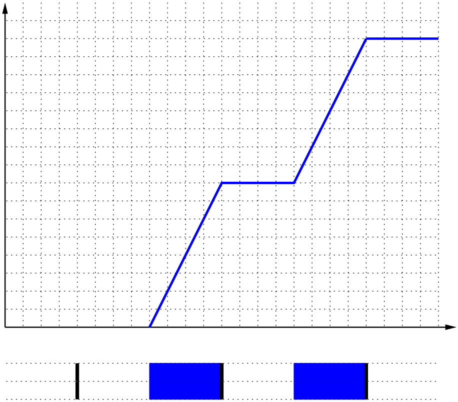
Example of BDM

- $(m = 2, \Delta = 8, [\beta_1 = .5, \beta_2 = 1])$



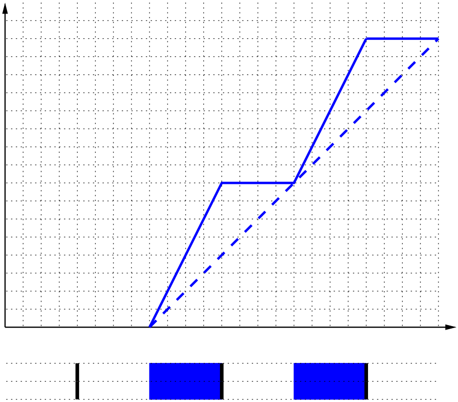
Example of BDM

- $(m = 2, \Delta = 8, [\beta_1 = .5, \beta_2 = 1])$
- Platform 1:
  - $\alpha_1 = \alpha_2 = 0.5,\ \Delta = 8$
  - $\pi_1 = (Q = 4, P = 8)$,
  - $\pi_2 = (Q = 4, P = 8)$



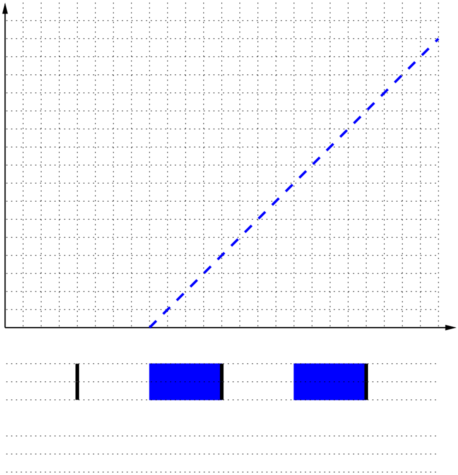
Example of BDM

- $(m = 2, \Delta = 8, [\beta_1 = .5, \beta_2 = 1])$
- Platform 1:
  - $\alpha_1 = \alpha_2 = 0.5,\ \Delta = 8$
  - $\pi_1 = (Q = 4, P = 8)$,
  - $\pi_2 = (Q = 4, P = 8)$
- We are only interested in the linear bound



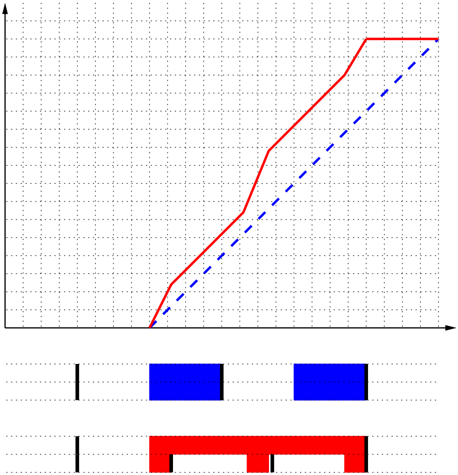
Example of BDM

- $(m = 2, \Delta = 8, [\beta_1 = .5, \beta_2 = 1])$
- Platform 1:
  - $\alpha_1 = \alpha_2 = 0.5, \ \Delta = 8$
  - $\pi_1 = (Q = 4, P = 8)$,
  - $\pi_2 = (Q = 4, P = 8)$
- We are only interested in the linear bound



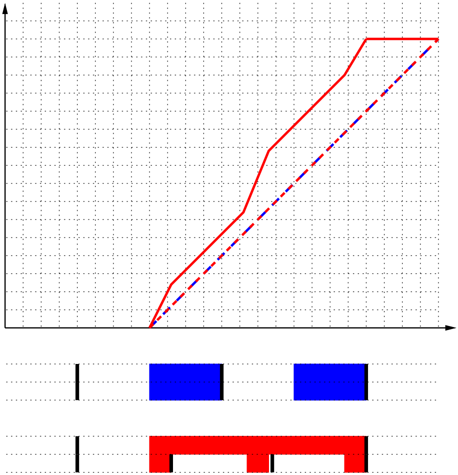
Example of BDM

- $(m = 2, \Delta = 8, [\beta_1 = .5, \beta_2 = 1])$
- Platform 1:
  - $\alpha_1 = \alpha_2 = 0.5$, $\Delta = 8$
  - $\pi_1 = (Q = 4, P = 8)$,
  - $\pi_2 = (Q = 4, P = 8)$
- We are only interested in the linear bound
- Platform 2:
  - $\alpha_1 = .75$, $\alpha_2 = .25$, $\Delta = 8$
  - $\pi_1 = (Q = 12, Q = 16)$
  - $\pi_2 = (Q = 1.33, Q = 5.33)$



Example of BDM

- $(m = 2, \Delta = 8, [\beta_1 = .5, \beta_2 = 1])$
- Platform 1:
  - $\alpha_1 = \alpha_2 = 0.5,\ \Delta = 8$
  - $\pi_1 = (Q = 4, P = 8)$,
  - $\pi_2 = (Q = 4, P = 8)$
- We are only interested in the linear bound
- Platform 2:
  - $\alpha_1 = .75,\ \alpha_2 = .25,\ \Delta = 8$
  - $\pi_1 = (Q = 12, Q = 16)$
  - $\pi_2 = (Q = 1.33, Q = 5.33)$
- The linear bound is the same!



Schedulability Analysis

- To analyse schedulability, we use the work by Bini, Baruah, Bertogna

$$\bigwedge_{i=1,\ldots,n} \bigvee_{k=1\ldots,m} \sum_{j=1}^{k} \alpha_j (D_i - \Delta)_0 \geq kC_i + W_i$$

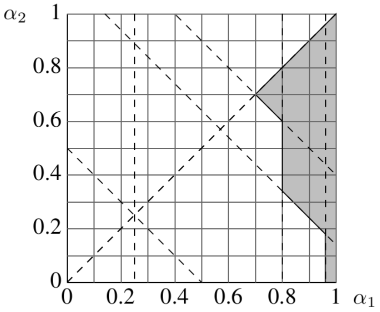
Where W_i is the interference of the task

Example of parameter computation

- Example (3 tasks on two virtual processors)

i	C_i	T_i
1	1	6
2	15	27
3	9	52

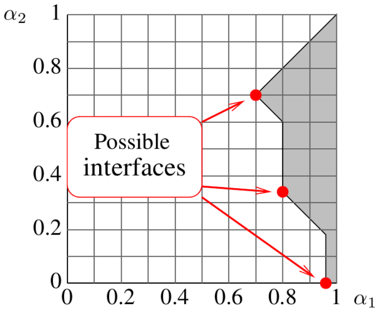
$$\begin{array}{llr} i=1 & k=1 & \alpha_1 \geq \frac{1}{4} \\ & k=2 & \alpha_1+\alpha_2 \geq \frac{1}{2} \\ i=2 & k=1 & \alpha_1 \geq \frac{4}{5} \\ & k=2 & \alpha_1+\alpha_2 \geq \frac{7}{5} \\ i=3 & k=1 & \alpha_1 \geq \frac{48}{50} \\ & k=2 & \alpha_1+\alpha_2 \geq \frac{57}{50} \end{array}$$



Example

- Example (3 tasks on two virtual processors)

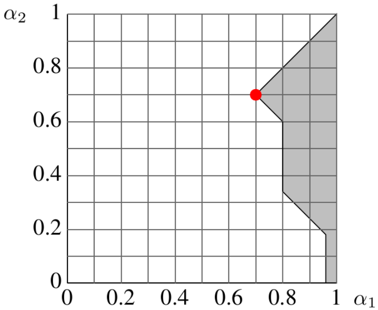
β_1	β_2	α_1	α_2
0.7	1.4	0.7	0.7
0.8	1.14	0.8	0.34
0.96	0.96	0.96	0



Example

- Example (3 tasks on two virtual processors)

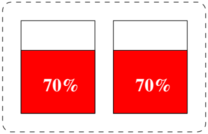
β_1	β_2	α_1	α_2
0.7	1.4	0.7	0.7
0.8	1.14	0.8	0.34
0.96	0.96	0.96	0

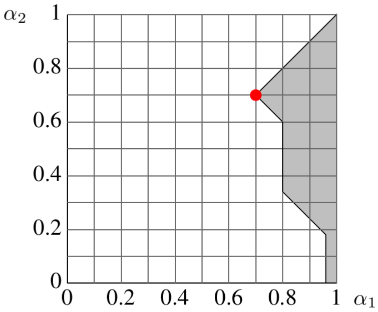


Example

- Example (3 tasks on two virtual processors)

β_1	β_2	α_1	α_2
0.7	1.4	0.7	0.7
0.8	1.14	0.8	0.34
0.96	0.96	0.96	0

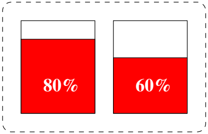


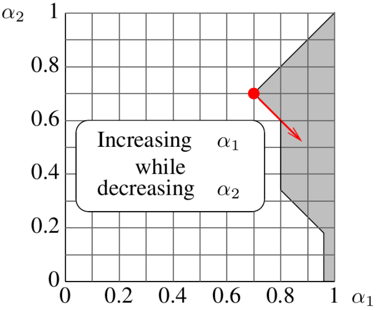


Example

- Example (3 tasks on two virtual processors)

β_1	β_2	α_1	α_2
0.7	1.4	0.7	0.7
0.8	1.14	0.8	0.34
0.96	0.96	0.96	0

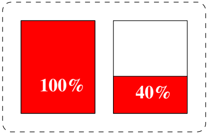


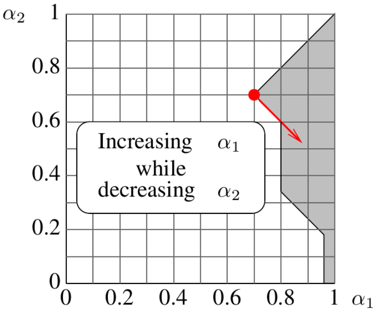


Example

- Example (3 tasks on two virtual processors)

β_1	β_2	α_1	α_2
0.7	1.4	0.7	0.7
0.8	1.14	0.8	0.34
0.96	0.96	0.96	0

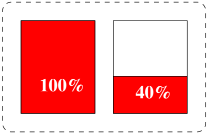


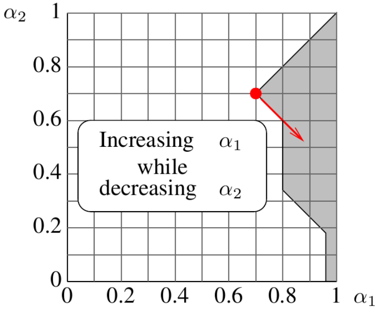


Example

- Example (3 tasks on two virtual processors)

β_1	β_2	α_1	α_2
0.7	1.4	0.7	0.7
0.8	1.14	0.8	0.34
0.96	0.96	0.96	0



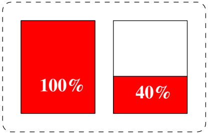


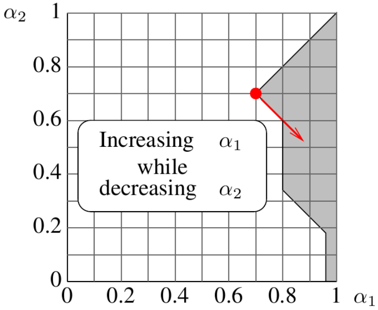
- Moving right-down the application remains schedulable

Example

- Example (3 tasks on two virtual processors)

β_1	β_2	α_1	α_2
0.7	1.4	0.7	0.7
0.8	1.14	0.8	0.34
0.96	0.96	0.96	0



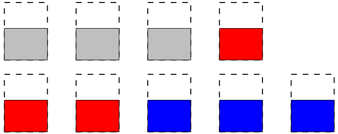


- Moving right-down the application remains schedulable
- We can use this flexibility in the allocation phase

Example of allocation

- Allocate three applications, with interface: $\mathcal{I} = \{3, 2, [\beta_1 = .51, \beta_2 = 1.02, \beta_3 = 1.53]\}$
- Worst-case platform: $\forall i = 1, 2, 3 \quad \alpha_i = 0.51$

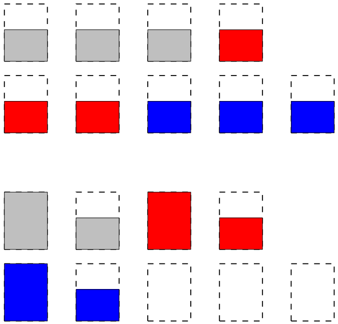
- Best fit decreasing:
  - There is little choice: every virtual processor in a separate physical processor
  - To improve allocation, we have to start from a different partitioning of the bandwidth (which one?)



Example of allocation

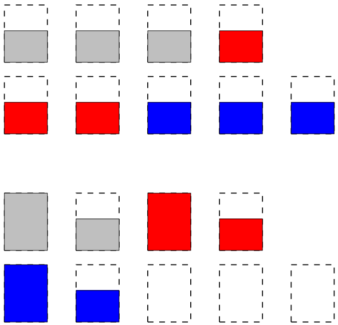
- Allocate three applications, with interface: $\mathcal{I} = \{3, 2, [\beta_1 = .51, \beta_2 = 1.02, \beta_3 = 1.53]\}$
- Worst-case platform: $\forall i = 1, 2, 3 \quad \alpha_i = 0.51$

- Best fit decreasing:
  - There is little choice: every virtual processor in a separate physical processor
  - To improve allocation, we have to start from a different partitioning of the bandwidth (which one?)
- Start from $\alpha_1 = 1, \alpha_2 = .53$
  - We obtain an allocation with **6 processors**, half of them are half-empty



Example of allocation

- Allocate three applications, with interface: $\mathcal{I} = \{3, 2, [\beta_1 = .51, \beta_2 = 1.02, \beta_3 = 1.53]\}$
- Worst-case platform: $\forall i = 1, 2, 3 \quad \alpha_i = 0.51$



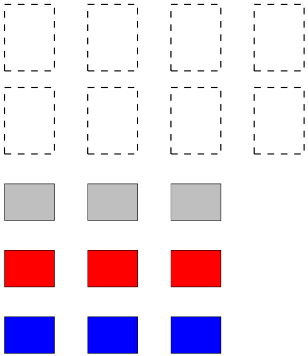
- Best fit decreasing:
  - There is little choice: every virtual processor in a separate physical processor
  - To improve allocation, we have to start from a different partitioning of the bandwidth (which one?)
- Start from $\alpha_1 = 1, \alpha_2 = .53$
  - We obtain an allocation with **6 processors**, half of them are half-empty
- Can we do better?

Fluid Best Fit

- We now propose an algorithm called **Fluid Best Fit**
  - starts from the "worst" platform and tries to allocate it with best-fit;
  - If it does not succeed, there is no way to allocate the interface, exit
  - If it finds an allocation: compresses the virtual processors in a fluid way so to achieve a better allocation
  - In doing so, it maintains the new platform compliant with the interface

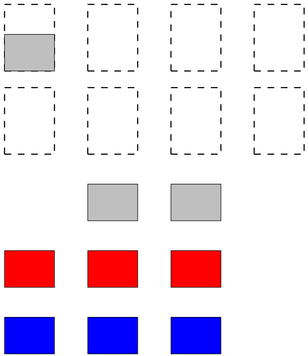
Example with FBF

- Start by allocating the first application (grey):



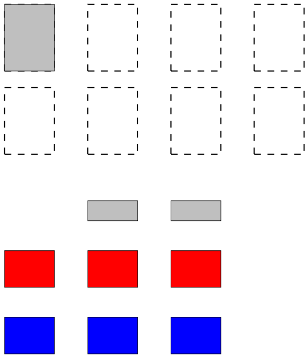
Example with FBF

- Start by allocating the first application (grey):
    - Allocate the first VP



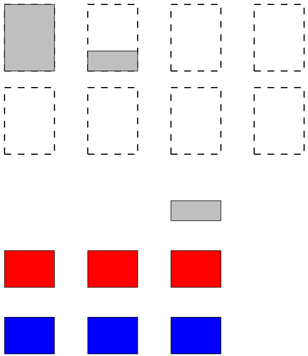
Example with FBF

- Start by allocating the first application (grey):
  - Allocate the first VP
  - extend it until it fills the physical processor



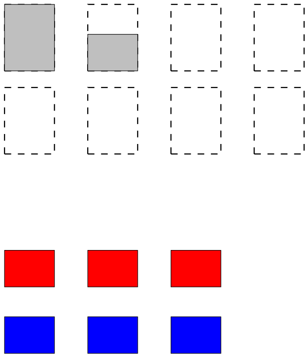
Example with FBF

- Start by allocating the first application (grey):
  - Allocate the first VP
  - extend it until it fills the physical processor
  - Allocate the second VP, extend it until it fills the second processor



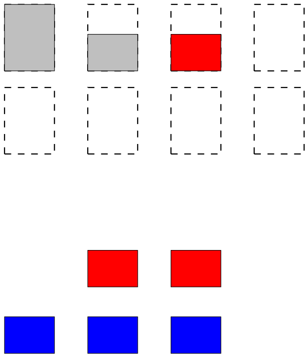
Example with FBF

- Start by allocating the first application (grey):
  - Allocate the first VP
  - extend it until it fills the physical processor
  - Allocate the second VP, extend it until it fills the second processor



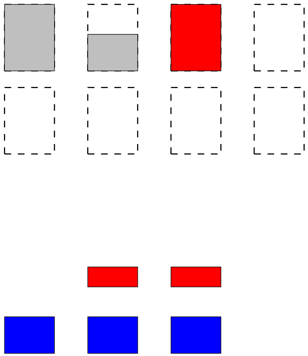
Example with FBF

- Start by allocating the first application (grey):
  - Allocate the first VP
  - extend it until it fills the physical processor
  - Allocate the second VP, extend it until it fills the second processor
- Second application, same procedure



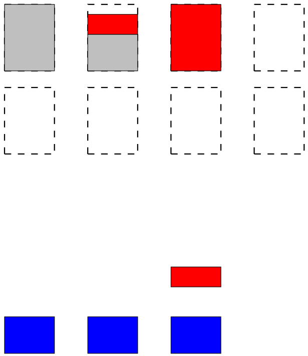
Example with FBF

- Start by allocating the first application (grey):
    - Allocate the first VP
    - extend it until it fills the physical processor
    - Allocate the second VP, extend it until it fills the second processor
- Second application, same procedure



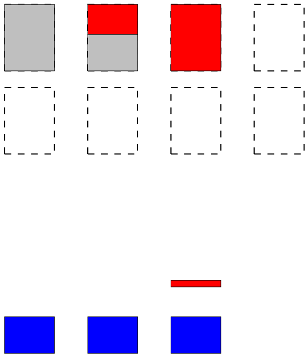
Example with FBF

- Start by allocating the first application (grey):
  - Allocate the first VP
  - extend it until it fills the physical processor
  - Allocate the second VP, extend it until it fills the second processor
- Second application, same procedure



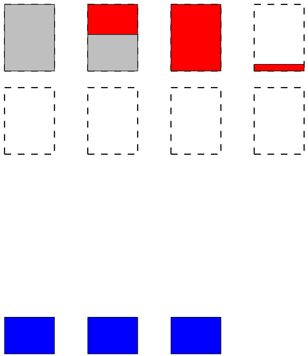
Example with FBF

- Start by allocating the first application (grey):
    - Allocate the first VP
    - extend it until it fills the physical processor
    - Allocate the second VP, extend it until it fills the second processor
- Second application, same procedure



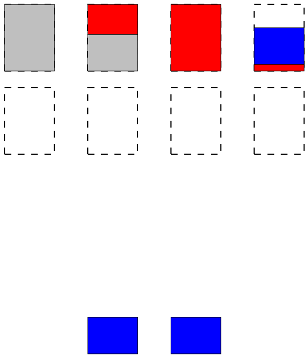
Example with FBF

- Start by allocating the first application (grey):
  - Allocate the first VP
  - extend it until it fills the physical processor
  - Allocate the second VP, extend it until it fills the second processor
- Second application, same procedure
- Third application, same procedure



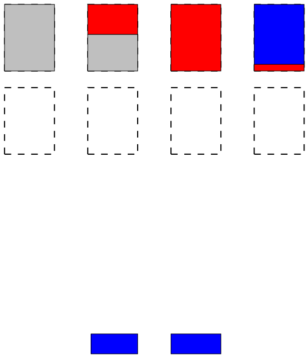
Example with FBF

- Start by allocating the first application (grey):
  - Allocate the first VP
  - extend it until it fills the physical processor
  - Allocate the second VP, extend it until it fills the second processor
- Second application, same procedure
- Third application, same procedure



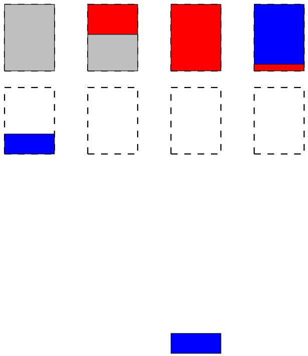
Example with FBF

- Start by allocating the first application (grey):
  - Allocate the first VP
  - extend it until it fills the physical processor
  - Allocate the second VP, extend it until it fills the second processor
- Second application, same procedure
- Third application, same procedure



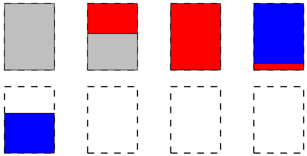
Example with FBF

- Start by allocating the first application (grey):
  - Allocate the first VP
  - extend it until it fills the physical processor
  - Allocate the second VP, extend it until it fills the second processor
- Second application, same procedure
- Third application, same procedure



Example with FBF

- Start by allocating the first application (grey):
  - Allocate the first VP
  - extend it until it fills the physical processor
  - Allocate the second VP, extend it until it fills the second processor
- Second application, same procedure
- Third application, same procedure
- At the end, we filled only 5 processors



Experimental results

We compared FBF against Best-Fit decreasing and First-Decreasing

- Light interfaces: randomly generated utilisation between $[0.2, 0.5] \cdot m$
- Heavy interfaces:randomly generated utilisation between $[0.3, 0.7] \cdot m$
- The **concavity ratio** is a measure of *how unbalanced* is the interface:
  - Concavity ratio equal to 1 implies a rigid interface with $\beta_1 = 1, \beta_2 = 2, \ldots, \beta_k = k$,
  - concavity ratio equal to 0 implies all VPs with the same utilisation α_k (flexible interface)

We measured the *Compaction index*: $\frac{\#\text{ used processors}}{\text{min }\#\text{ needed processors}}$

Results

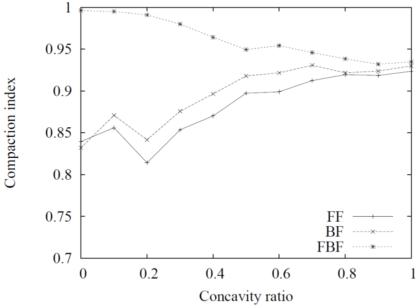


Figure: Light interfaces

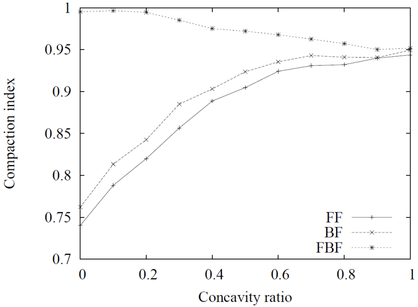


Figure: Heavy interfaces

- For flexible interfaces, FBF is almost optimal
- As the concavity ratio increases (rigid interfaces), the performance of the three algorithms becomes similar
- There is a larger difference for heavy interfaces

Conclusions and future work

In this paper we presented

- BDM – A new interface specification model
- How to compute interface parameters from application parameters
- FBF – A flexible allocation algorithm

Future work

- Improve FBF by providing more alternative interfaces with different concavity ratio
- Consider more complex task models (interacting applications)

Conclusions and future work

In this paper we presented

- BDM – A new interface specification model
- How to compute interface parameters from application parameters
- FBF – A flexible allocation algorithm

Future work

- Improve FBF by providing more alternative interfaces with different concavity ratio
- Consider more complex task models (interacting applications)

Questions ?

Bibliography

- I. Shin, A. Easwaran, and I. Lee, "Hierarchical scheduling framework for virtual clustering multiprocessors," in *Proceedings of the* 20*th* *Euromicro Conference on Real-Time Systems*, Prague, Czech Republic, Jul. 2008, pp. 181–190.
- E. Bini, M. Bertogna, and S. Baruah, "Virtual multiprocessor platforms: Specification and use," in *Proceedings of the* 30*th* *IEEE Real-Time Systems Symposium*, Washinghton, DC, USA, Dec. 2009, pp. 437–446.
- A. K. Mok, X. Feng, and D. Chen, "Resource partition for real-time systems," in *Proceedings of the* 7*th* *IEEE Real-Time Technology and Applications Symposium*, Taipei, Taiwan, May 2001, pp. 75–84.