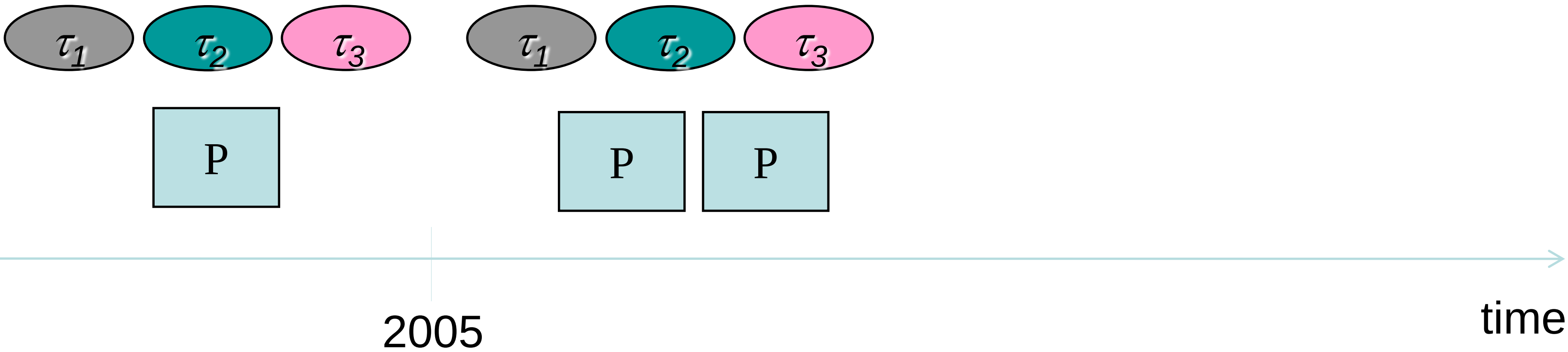
Assigning Real-Time Tasks on Heterogeneous Multiprocessors with Two Unrelated Types of Processors

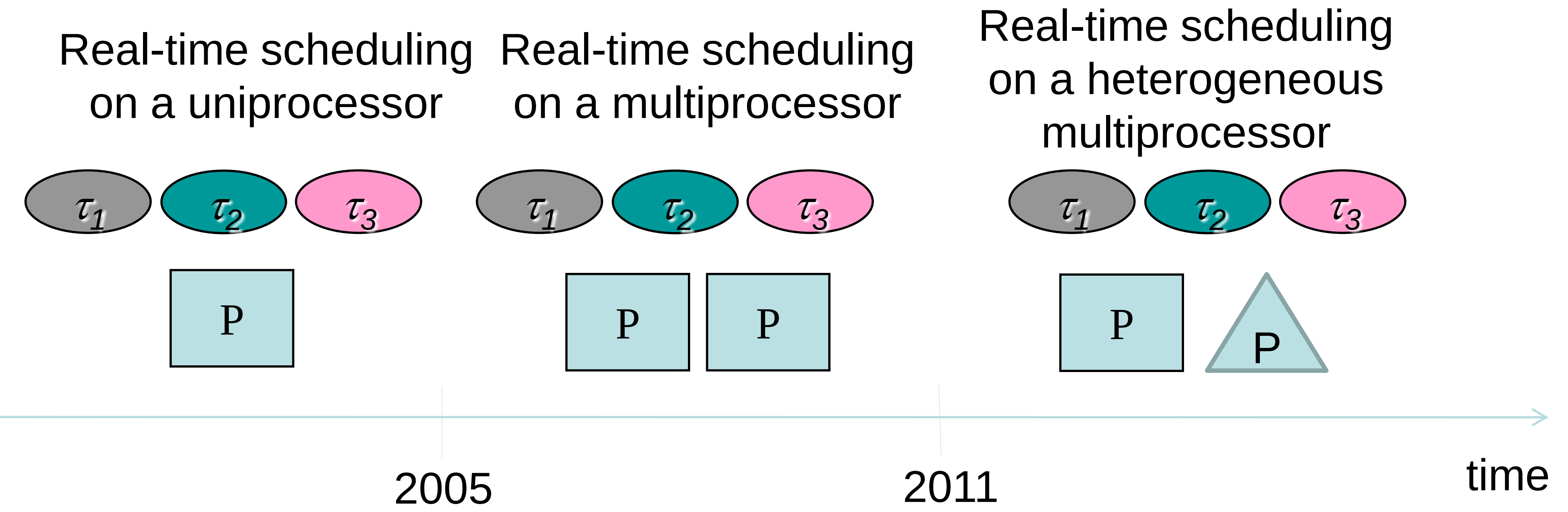
Björn Andersson, Gurulingesh Raravi and Konstantinos Bletsas

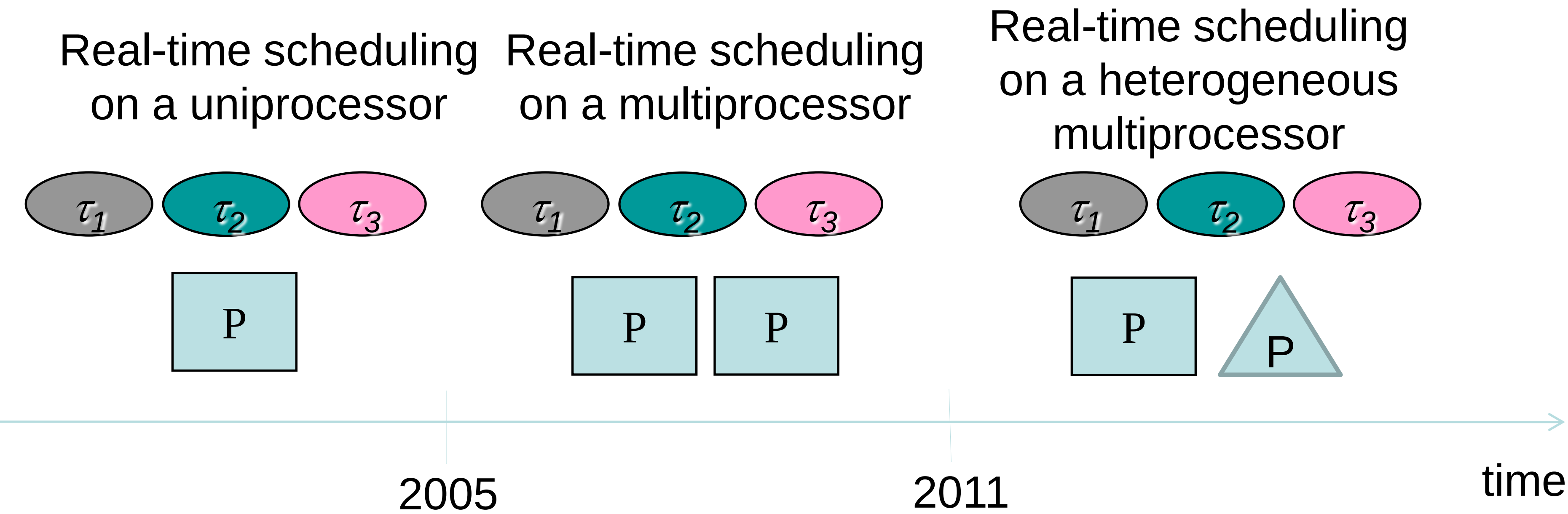
Real-time scheduling on a uniprocessor

Real-time scheduling on a uniprocessor

Real-time scheduling on a multiprocessor




Real-time scheduling on a uniprocessor
Real-time scheduling on a multiprocessor
Real-time scheduling on a heterogeneous multiprocessor
τ_1 τ_2 τ_3
τ_1 τ_2 τ_3
τ_1 τ_2 τ_3
P
P P
P P
2005
2011
time

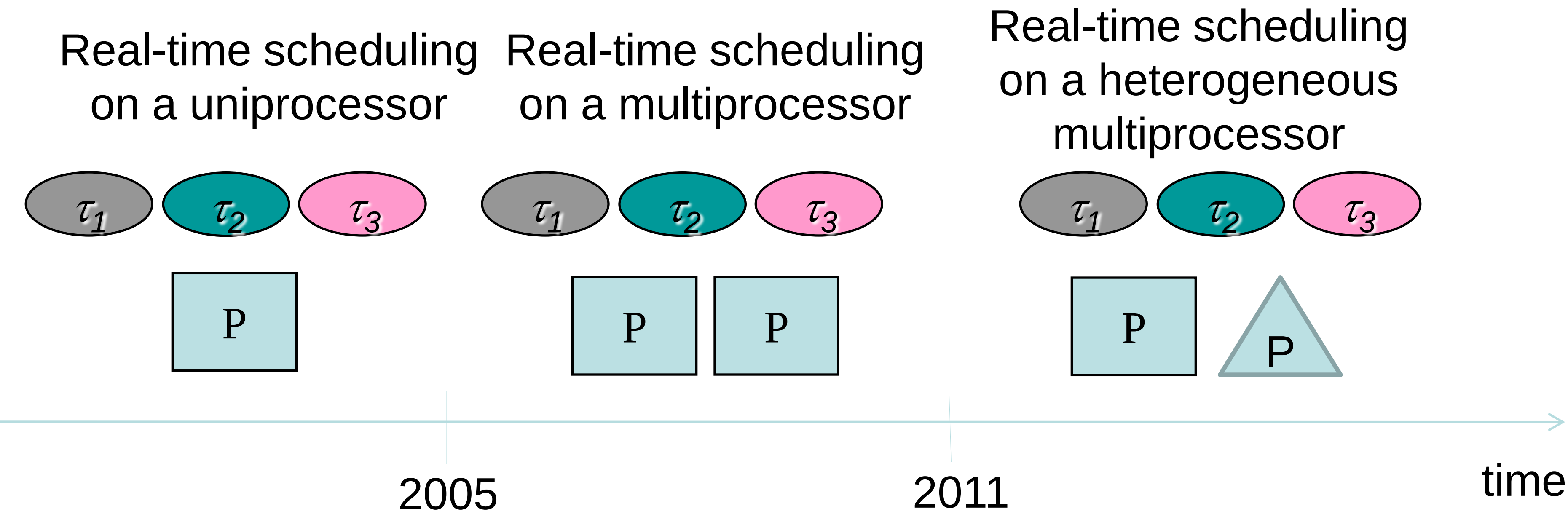


Scheduling related challenges for heterogeneous multiprocessors in real-time systems:

-Precedence constraints

-Sharing of low-level hardware resources (caches, interconnection networks);

-The execution time of a task depends on which processor it executes on.



Scheduling related challenges for heterogeneous multiprocessors in real-time systems:

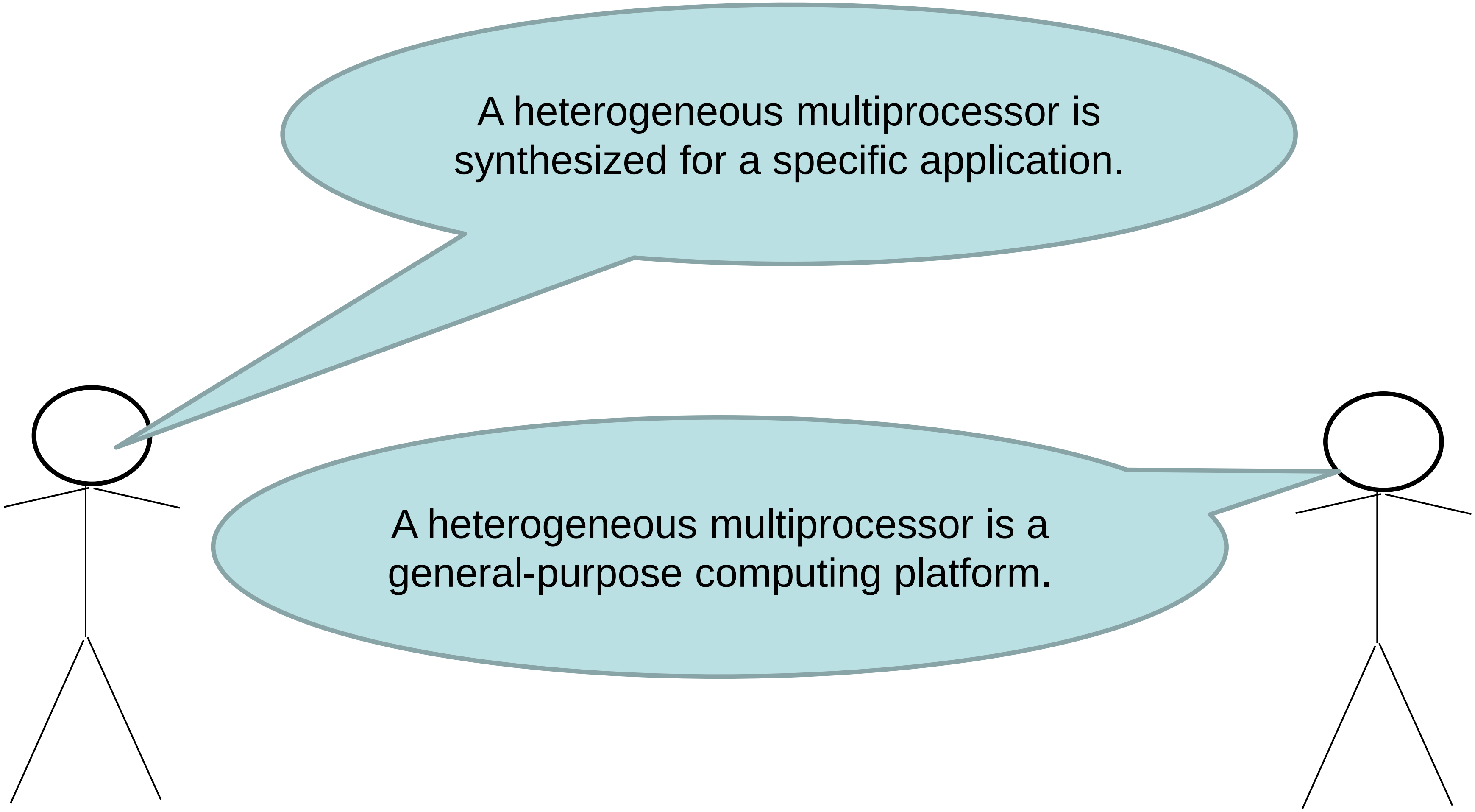
-Precedence constraints

-Sharing of low-level hardware resources (caches, interconnection networks);

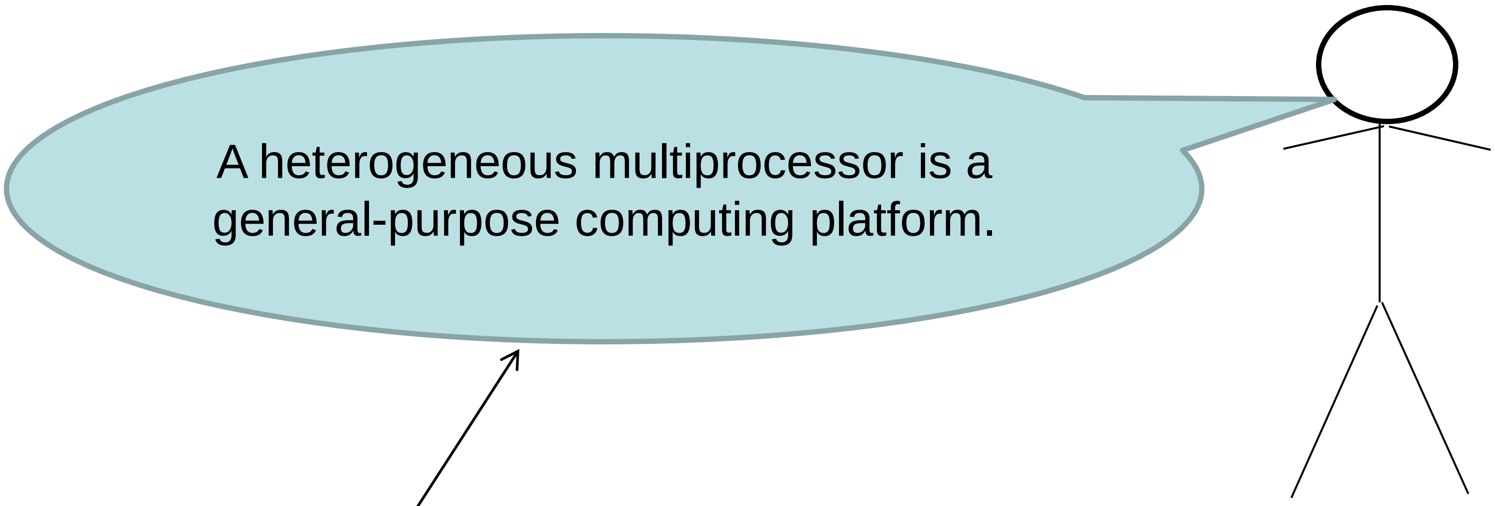
-The execution time of a task depends on which processor it executes on.

Focus of this talk.

Different views on a heterogeneous multiprocessors:



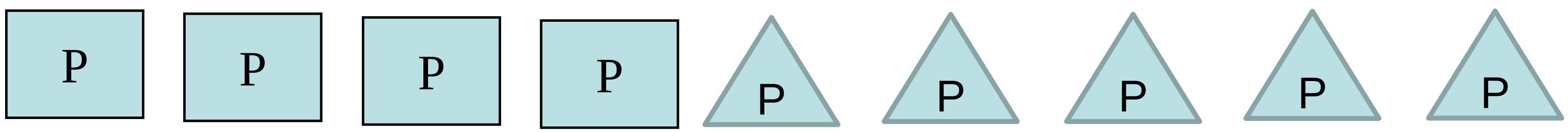
Different views on a heterogeneous multiprocessors:



View taken in this talk.

How many different types of processors does the computer system have?

-Two types of processors

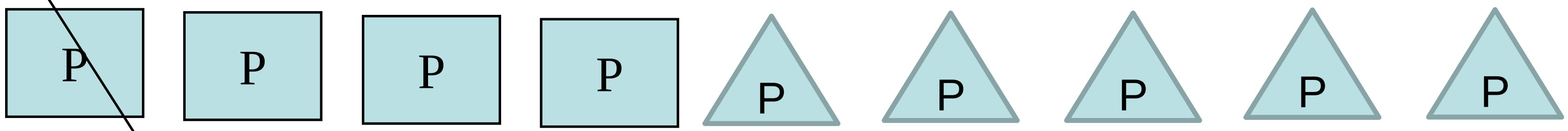


- More that two types of processors

- 

How many different types of processors does the computer system have?

-Two types of processors



- More that two types of processors

Considered in this talk.

Different assumptions about task migration:

-A task can migrate to any processor;

-A task can migrate but only between processors of the same type;

-A task cannot migrate.

Different assumptions about task migration:

-A task can migrate to any processor;

-A task can migrate but only between processors of the same type;

-A task cannot migrate.

Considered in this talk.

Different task models

-Dependent tasks: An arrival of a task is dependent on an event related to another task.

- Independent tasks: An arrival of a task is independent of events related to other tasks.
  + periodic tasks
    * implicit deadline
    * explicit deadline
  + sporadic tasks
    * implicit deadline
    * explicit deadline

Different task models

-Dependent tasks: An arrival of a task is dependent on an event related to another task.

- Independent tasks: An arrival of a task is independent of events related to other tasks.
  + periodic tasks
    * implicit deadline
    * explicit deadline
  + sporadic tasks
    * implicit deadline
    * explicit deadline

Considered in this talk.

Different task models

- Dependent tasks: An arrival of a task is dependent on an event related to another task.
- Independent tasks: An arrival of a task is independent of events related to other tasks.
    + periodic tasks
        * implicit deadline
        * explicit deadline
    + sporadic tasks
        * implicit deadline
        * explicit deadline

Different scheduling algorithms:

- RM
- EDF

Different task models

- Dependent tasks: An arrival of a task is dependent on an event related to another task.
- Independent tasks: An arrival of a task is independent of events related to other tasks.
  + periodic tasks
    * implicit deadline
    * explicit deadline
  + sporadic tasks
    * implicit deadline
    * explicit deadline

Different scheduling algorithms:

- RM
- EDF

Considered in this talk.

Model

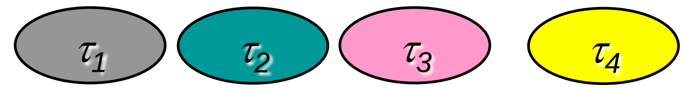
- P^1 denotes the set of all processors of type-1.
- P^2 denotes the set of all processors of type-2.
- τ denotes a set of tasks $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$;
- A task τ_i assigned to a processor of type-1 has utilization U_i^1.
- A task τ_i assigned to a processor of type-2 has utilization U_i^2.

Problem statement

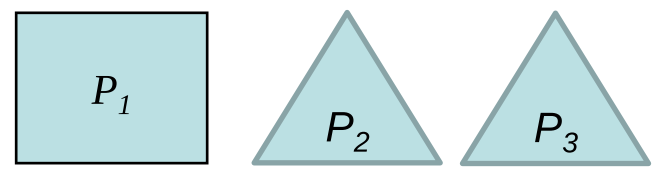
Assign tasks to processors so that each processor is utilized to at most 100%.

Example of a problem instance

$\tau=\{\tau_1, \tau_2, \tau_3, \tau_4\}$ $P^1=\{\mathrm{P}_1\}$, $P^2=\{\mathrm{P}_2, \mathrm{P}_3\}$.



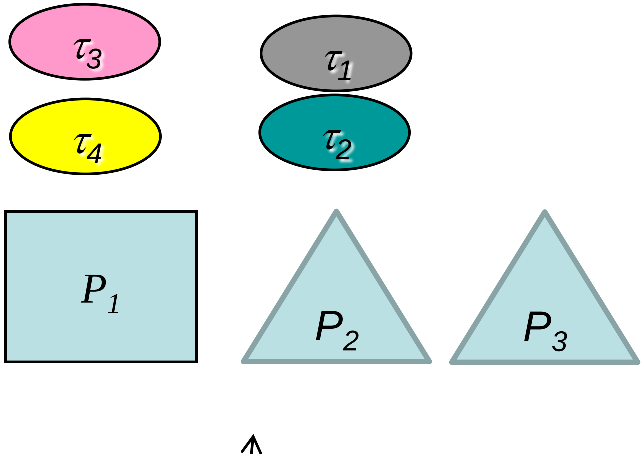
	Processor type-1	Processor type-2
τ_1	$U_1^1=0.90$	$U_1^2=0.40$
τ_2	$U_2^1=0.90$	$U_2^2=0.40$
τ_3	$U_3^1=0.40$	$U_3^2=0.80$
τ_4	$U_4^1=0.40$	$U_4^2=0.80$



Example of a problem instance

$\tau=\{\tau_1,\tau_2,\tau_3,\tau_4\}$ $P^1=\{P_1\}$, $P^2=\{P_2,P_3\}$.

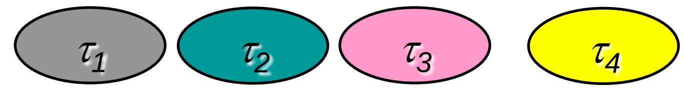
	Processor type-1	Processor type-2
τ_1	$U_1^1=0.90$	$U_1^2=0.40$
τ_2	$U_2^1=0.90$	$U_2^2=0.40$
τ_3	$U_3^1=0.40$	$U_3^2=0.80$
τ_4	$U_4^1=0.40$	$U_4^2=0.80$



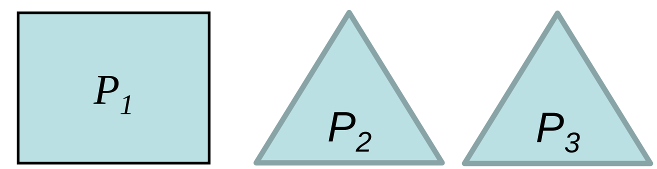
We can do the assignment like this.

Let us try First-Fit

$\tau=\{\tau_1,\tau_2,\tau_3,\tau_4\}$ $P^1=\{P_1\}$, $P^2=\{P_2,P_3\}$.

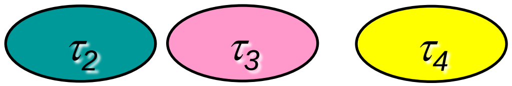


	Processor type-1	Processor type-2
τ_1	$U_1^1=0.90$	$U_1^2=0.40$
τ_2	$U_2^1=0.90$	$U_2^2=0.40$
τ_3	$U_3^1=0.40$	$U_3^2=0.80$
τ_4	$U_4^1=0.40$	$U_4^2=0.80$

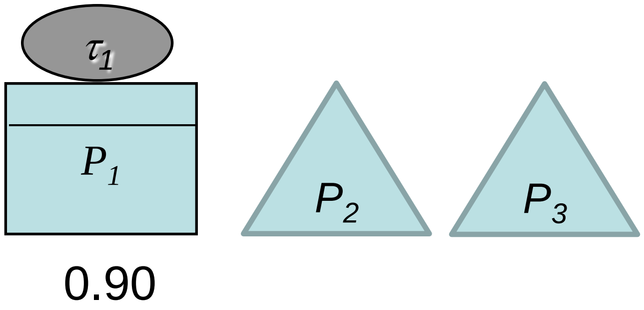


Let us try First-Fit

$\tau=\{\tau_1, \tau_2, \tau_3, \tau_4\}$ $P^1=\{P_1\}$, $P^2=\{P_2, P_3\}$.

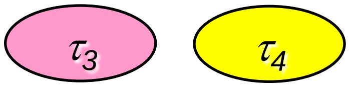


	Processor type-1	Processor type-2
τ_1	$U_1^1=0.90$	$U_1^2=0.40$
τ_2	$U_2^1=0.90$	$U_2^2=0.40$
τ_3	$U_3^1=0.40$	$U_3^2=0.80$
τ_4	$U_4^1=0.40$	$U_4^2=0.80$

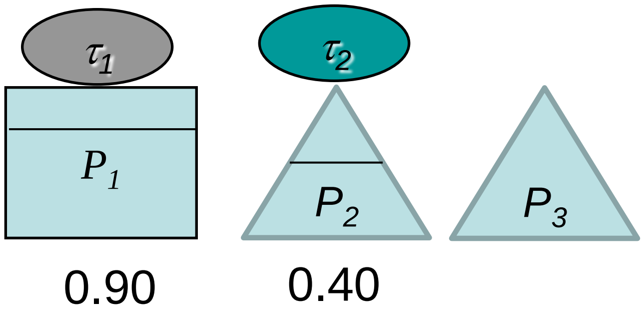


Let us try First-Fit

$\tau=\{\tau_1, \tau_2, \tau_3, \tau_4\}$ $P^1=\{P_1\}$, $P^2=\{P_2, P_3\}$.



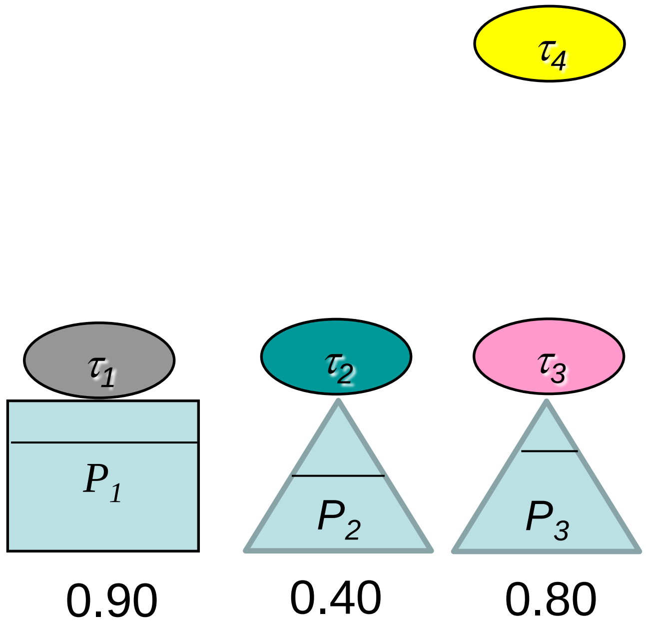
	Processor type-1	Processor type-2
τ_1	$U_1^1=0.90$	$U_1^2=0.40$
τ_2	$U_2^1=0.90$	$U_2^2=0.40$
τ_3	$U_3^1=0.40$	$U_3^2=0.80$
τ_4	$U_4^1=0.40$	$U_4^2=0.80$



Let us try First-Fit

$\tau = \{\tau_1, \tau_2, \tau_3, \tau_4\}$ $P^1 = \{P_1\}$, $P^2 = \{P_2, P_3\}$.

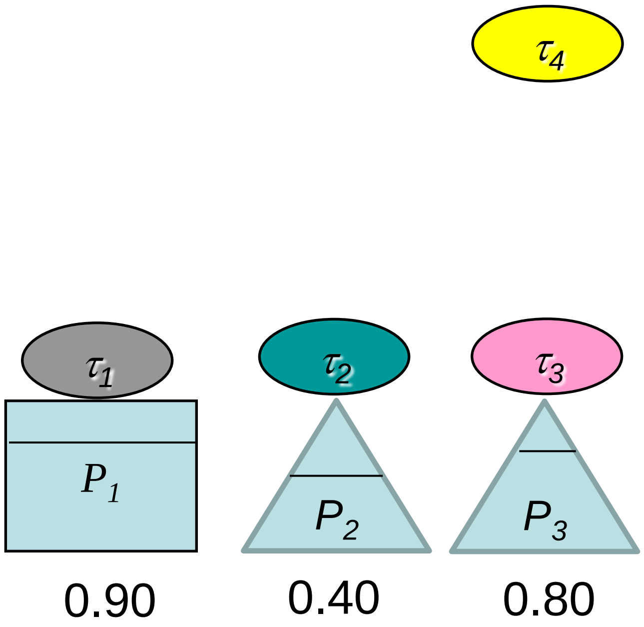
	Processor type-1	Processor type-2
τ_1	$U_1^1=0.90$	$U_1^2=0.40$
τ_2	$U_2^1=0.90$	$U_2^2=0.40$
τ_3	$U_3^1=0.40$	$U_3^2=0.80$
τ_4	$U_4^1=0.40$	$U_4^2=0.80$



Let us try First-Fit

$\tau=\{\tau_1, \tau_2, \tau_3, \tau_4\}$ $P^1=\{P_1\}$, $P^2=\{P_2,P_3\}$.

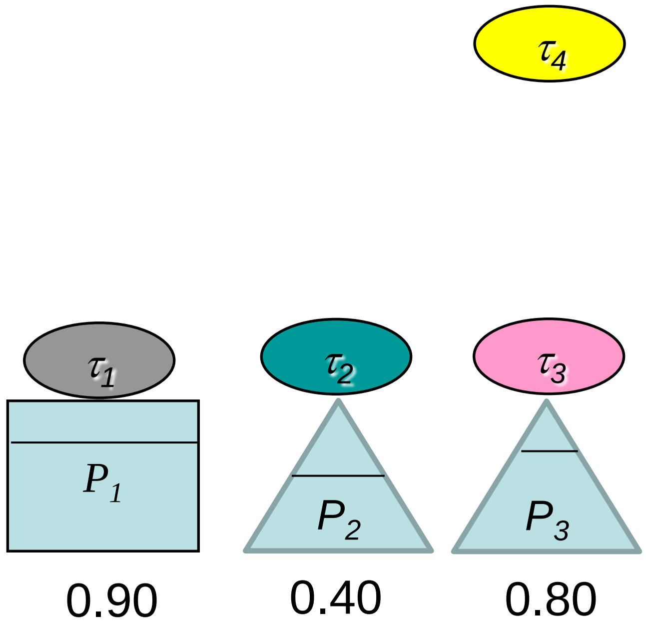
	Processor type-1	Processor type-2
τ_1	$U_1^1=0.90$	$U_1^2=0.40$
τ_2	$U_2^1=0.90$	$U_2^2=0.40$
τ_3	$U_3^1=0.40$	$U_3^2=0.80$
τ_4	$U_4^1=0.40$	$U_4^2=0.80$



There is no processor on which τ_4 can be assigned.

Let us try First-Fit

$\tau=\{\tau_1, \tau_2, \tau_3, \tau_4\}$ $P^1=\{P_1\}$, $P^2=\{P_2, P_3\}$.



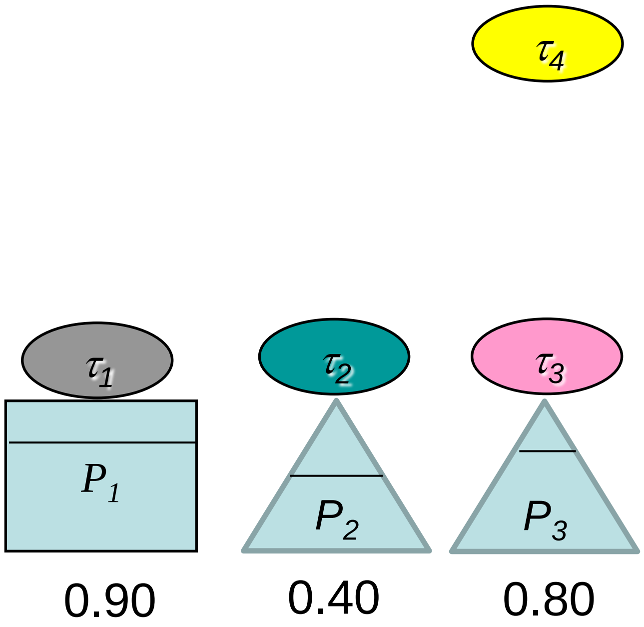
	Processor type-1	Processor type-2
τ_1	$U_1^1=0.90$	$U_1^2=0.40$
τ_2	$U_2^1=0.90$	$U_2^2=0.40$
τ_3	$U_3^1=0.40$	$U_3^2=0.80$
τ_4	$U_4^1=0.40$	$U_4^2=0.80$

First-Fit fails on this task set.

Let us try First-Fit

$\tau=\{\tau_1, \tau_2, \tau_3, \tau_4\}$ $P^1=\{P_1\}$, $P^2=\{P_2,P_3\}$.

	Processor type-1	Processor type-2
τ_1	$U_1^1=0.90$	$U_1^2=0.40$
τ_2	$U_2^1=0.90$	$U_2^2=0.40$
τ_3	$U_3^1=0.40$	$U_3^2=0.80$
τ_4	$U_4^1=0.40$	$U_4^2=0.80$



First-Fit has inifinite competitive ratio on heterogeneous multiprocessor with two types (shown in the paper).

Design Ideas

Idea1: Try to assign a task on the processor where its utilization is smaller.

Idea 2: if $U_i^1 \leq$ THRESHOLD and $U_i^2 >$ THRESHOLD then assign task τ_i to processor of type-1.

Design Ideas

Idea1: Try to assign a task on the processor where its utilization is smaller.

Idea 2: if $U_i^1 \leq$ THRESHOLD and $U_i^2 >$ THRESHOLD then assign task τ_i to processor of type-1.

Partition the task set

$\tau^1 = \{ \tau_i \propto \tau$ such that $U_i^1 \leq U_i^2 \}$

$\tau^2 = \{ \tau_i \propto \tau$ such that $U_i^1 > U_i^2 \}$

Design Ideas

Idea1: Try to assign a task on the processor where its utilization is smaller.

Idea 2: if $U_i^1 \leq$THRESHOLD and $U_i^2 >$THRESHOLD then assign task τ_i to processor of type-1.

Partition the task set

τ^1={ $\tau_i \propto \tau$ such that $U_i^1 \leq U_i^2$}
τ^2={ $\tau_i \propto \tau$ such that $U_i^1 > U_i^2$}

$H1$ = { $\tau_i \propto \tau^1$ such that $U_i^2 > 1/2$}
$F1$ = { $\tau_i \propto \tau^1$ such that $U_i^2 \leq 1/2$}

Design Ideas

Idea1: Try to assign a task on the processor where its utilization is smaller.

Idea 2: if $U_i^1 \leq$ THRESHOLD and $U_i^2 >$ THRESHOLD then assign task τ_i to processor of type-1.

Partition the task set

$\tau^1 = \{ \tau_i \propto \tau$ such that $U_i^1 \leq U_i^2 \}$
$\tau^2 = \{ \tau_i \propto \tau$ such that $U_i^1 > U_i^2 \}$

$H1 = \{ \tau_i \propto \tau^1$ such that $U_i^2 > 1/2 \}$
$F1 = \{ \tau_i \propto \tau^1$ such that $U_i^2 \leq 1/2 \}$
$H2 = \{ \tau_i \propto \tau^2$ such that $U_i^1 > 1/2 \}$
$F2 = \{ \tau_i \propto \tau^2$ such that $U_i^1 \leq 1/2 \}$

Algorithm Outline

1. Form the sets *H1*,*H2*,*F1*,*F2*
2. first-fit(H1, P^1)
3. first-fit(*H2*, P^2)
4. first-fit(*F1*, P^1)
5. first-fit(*F2*, P^2)

Partition the task set

τ^1={ $\tau_i \propto \tau$ such that $U_i^1 \leq U_i^2$}
τ^2={ $\tau_i \propto \tau$ such that $U_i^1 > U_i^2$}

H1 = { $\tau_i \propto \tau^1$ such that $U_i^2 > 1/2$}
F1 = { $\tau_i \propto \tau^1$ such that $U_i^2 \leq 1/2$}
H2 = { $\tau_i \propto \tau^2$ such that $U_i^1 > 1/2$}
F2 = { $\tau_i \propto \tau^2$ such that $U_i^1 \leq 1/2$}

Algorithm Partition the task set

Form sets $H1$, $H2$, $F1$, $F2$
$\forall p$: U[p] := 0
$\forall p$: τ[p] := $\emptyset$
if first-fit($H1$, P^1) $\neq H1$ **then** declare FAILURE
if first-fit($H2$, P^2) $\neq H2$ **then** declare FAILURE
$F11$:= first-fit($F1$, P^1)
$F22$:= first-fit($F2$, P^2)
if $(F11 = F1) \wedge (F22 = F2)$ **then** declare SUCCESS
if $(F11 \neq F1) \wedge (F22 \neq F2)$ **then** declare FAILURE
if $(F11 \neq F1) \wedge (F22 = F2)$ **then**
    $F12 := F1 \setminus F11$
    if first-fit($F12$, P^2) = $F12$ **then**
        declare SUCCESS
    else
        declare FAILURE
    end
end
if $(F11 = F1) \wedge (F22 \neq F2)$ **then**
    $F21 := F2 \setminus F22$
    if first-fit($F21$, P^1) = $F21$ **then**
        declare SUCCESS
    else
        declare FAILURE
    end
end

τ^1={ $\tau_i \propto \tau$ such that $U_i^1 \leq U_i^2$}
τ^2={ $\tau_i \propto \tau$ such that $U_i^1 > U_i^2$}

$H1$ = { $\tau_i \propto \tau^1$ such that U_i^2>1/2}
$F1$ = { $\tau_i \propto \tau^1$ such that U_i^2≤1/2}
$H2$ = { $\tau_i \propto \tau^2$ such that U_i^1>1/2}
$F2$ = { $\tau_i \propto \tau^2$ such that U_i^1≤1/2}

FF-3C Partition the task set

Form sets $H1$, $H2$, $F1$, $F2$
$\forall p$: U[p] := 0
$\forall p$: τ[p] := $\emptyset$
if first-fit($H1$, P^1) $\neq H1$ **then** declare FAILURE
if first-fit($H2$, P^2) $\neq H2$ **then** declare FAILURE
$F11$:= first-fit($F1$, P^1)
$F22$:= first-fit($F2$, P^2)
if $(F11 = F1) \wedge (F22 = F2)$ **then** declare SUCCESS
if $(F11 \neq F1) \wedge (F22 \neq F2)$ **then** declare FAILURE
if $(F11 \neq F1) \wedge (F22 = F2)$ **then**
  $F12 := F1 \setminus F11$
  if first-fit($F12$, P^2) = $F12$ **then**
    declare SUCCESS
  else
    declare FAILURE
  end
end
if $(F11 = F1) \wedge (F22 \neq F2)$ **then**
  $F21 := F2 \setminus F22$
  if first-fit($F21$, P^1) = $F21$ **then**
    declare SUCCESS
  else
    declare FAILURE
  end
end

τ^1={ $\tau_i \propto \tau$ such that $U_i^1 \leq U_i^2$}
τ^2={ $\tau_i \propto \tau$ such that $U_i^1 > U_i^2$}

$H1$ = { $\tau_i \propto \tau^1$ such that U_i^2>1/2}
$F1$ = { $\tau_i \propto \tau^1$ such that $U_i^2 \leq$1/2}
$H2$ = { $\tau_i \propto \tau^2$ such that U_i^1>1/2}
$F2$ = { $\tau_i \propto \tau^2$ such that $U_i^1 \leq$1/2}

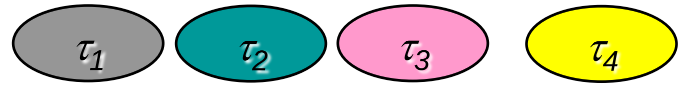
FF-3C

Form sets $H1$, $H2$, $F1$, $F2$
$\forall p$: U[p] := 0
$\forall p$: τ[p] := $\emptyset$
if first-fit($H1$, P^1) $\neq H1$ **then** declare FAILURE
if first-fit($H2$, P^2) $\neq H2$ **then** declare FAILURE
$F11$:= first-fit($F1$, P^1)
$F22$:= first-fit($F2$, P^2)
if $(F11 = F1) \wedge (F22 = F2)$ **then** declare SUCCESS
if $(F11 \neq F1) \wedge (F22 \neq F2)$ **then** declare FAILURE
if $(F11 \neq F1) \wedge (F22 = F2)$ **then**
  $F12 := F1 \setminus F11$
  if first-fit($F12$, P^2) = $F12$ **then**
    declare SUCCESS
  else
    declare FAILURE
  end
end
if $(F11 = F1) \wedge (F22 \neq F2)$ **then**
  $F21 := F2 \setminus F22$
  if first-fit($F21$, P^1) = $F21$ **then**
    declare SUCCESS
  else
    declare FAILURE
  end
end

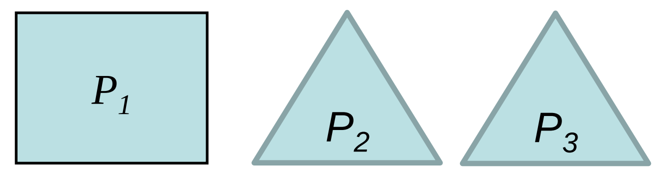
1. **function** first-fit(ts : set of tasks; ps : set of processors) return set of tasks
2. assigned_tasks := $\emptyset$
3. If ps consists of type-1 (type-2) processors, then order ts by decreasing U_i^2/U_i^1 (resp., incr. U_i^1/U_i^2). Use any order for processors ps, but maintain it during the execution of the function first-fit.
4. τ_i := first task in ts
5. p := first processor in ps
6. Let k denote the type of processor p (either 1 or 2)
7. **if** U[p]+$U_i^k \leq 1$ **then**
8. U[p] := U[p]+U_i^k
9. τ[p] := τ[p] $\cup \{\tau_i\}$
10. assigned_tasks := assigned_tasks $\cup \{\tau_i\}$
11. **if** remaining tasks exist in ts **then**
12. τ_i := next task in ts
13. go to line 5.
14. **else**
15. **return** assigned_tasks
16. **end if**
17. **else**
18. **if** remaining processors exist in ps **then**
19. p := next processor in ps
20. go to line 6.
21. **else**
22. **return** assigned_tasks
23. **end if**
24. **end if**

Applying FF-3C on an example

$\tau=\{\tau_1,\tau_2,\tau_3,\tau_4\}$ $P^1=\{P_1\}$, $P^2=\{P_2,P_3\}$.



	Processor type-1	Processor type-2
τ_1	U_1^1=0.90	U_1^2=0.40
τ_2	U_2^1=0.90	U_2^2=0.40
τ_3	U_3^1=0.40	U_3^2=0.80
τ_4	U_4^1=0.40	U_4^2=0.80



FF-3C

Form sets $H1$, $H2$, $F1$, $F2$
$\forall p$: U[p] := 0
$\forall p$: τ[p] := $\emptyset$
if first-fit($H1$, P^1) $\neq H1$ **then** declare FAILURE
if first-fit($H2$, P^2) $\neq H2$ **then** declare FAILURE
$F11$:= first-fit($F1$, P^1)
$F22$:= first-fit($F2$, P^2)
if $(F11 = F1) \wedge (F22 = F2)$ **then** declare SUCCESS
if $(F11 \neq F1) \wedge (F22 \neq F2)$ **then** declare FAILURE
if $(F11 \neq F1) \wedge (F22 = F2)$ **then**
  $F12 := F1 \setminus F11$
  if first-fit($F12$, P^2) = $F12$ **then**
    declare SUCCESS
  else
    declare FAILURE
  end
end
if $(F11 = F1) \wedge (F22 \neq F2)$ **then**
  $F21 := F2 \setminus F22$

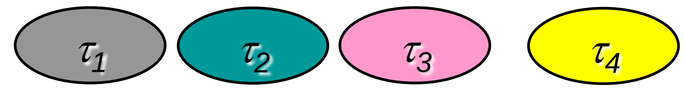
    declare FAILURE
  end
end

1. **function** first-fit(ts : set of tasks; ps : set of processors) return set of tasks
2. assigned_tasks := $\emptyset$
3. If ps consists of type-1 (type-2) processors, then order ts by decreasing U_i^2/U_i^1 (resp., incr. U_i^1/U_i^2). Use any order for processors ps, but maintain it during the execution of the function first-fit.
4. τ_i := first task in ts
5. p := first processor in ps
6. Let k denote the type of processor p (either 1 or 2)
7. **if** U[p]+$U_i^k \leq 1$ **then**
8. U[p] := U[p]+U_i^k
9. τ[p] := τ[p] $\cup \{\tau_i\}$
10. assigned_tasks := assigned_tasks $\cup \{\tau_i\}$
11. **if** remaining tasks exist in ts **then**
12. τ_i := next task in ts
13. go to line 5.
14. **else**
15. **return** assigned_tasks
16. **end if**
17. **else**
18. **if** remaining processors exist in ps **then**
19. p := next processor in ps
20. go to line 6.

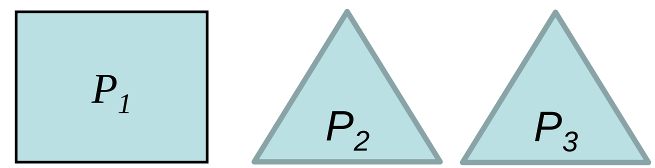
Let us execute this line.

Applying FF-3C on an example

$\tau = \{\tau_1, \tau_2, \tau_3, \tau_4\}$ $P^1 = \{P_1\}$, $P^2 = \{P_2, P_3\}$.



	Processor type-1	Processor type-2
τ_1	$U_1^1=0.90$	$U_1^2=0.40$
τ_2	$U_2^1=0.90$	$U_2^2=0.40$
τ_3	$U_3^1=0.40$	$U_3^2=0.80$
τ_4	$U_4^1=0.40$	$U_4^2=0.80$



$\tau^1 = \{\tau_3, \tau_4\}$ $H1 = \{\tau_3, \tau_4\}$ $F1 = \{\}$

$\tau^2 = \{\tau_1, \tau_2\}$ $H2 = \{\tau_1, \tau_2\}$ $F2 = \{\}$

FF-3C

1. Form sets $H1$, $H2$, $F1$, $F2$
2. $\forall p$: U[p] := 0
3. $\forall p$: τ[p] := $\emptyset$
4. **if** first-fit($H1$, P^1) $\neq H1$ **then** declare FAILURE
5. **if** first-fit($H2$, P^2) $\neq H2$ **then** declare FAILURE
6. $F11$:= first-fit($F1$, P^1)
7. $F22$:= first-fit($F2$, P^2)
8. **if** $(F11 = F1) \wedge (F22 = F2)$ **then** declare SUCCESS
9. **if** $(F11 \neq F1) \wedge (F22 \neq F2)$ **then** declare FAILURE
10. **if** $(F11 \neq F1) \wedge (F22 = F2)$ **then**
11. $F12 := F1 \setminus F11$
12.   **if** first-fit($F12$, P^2) = $F12$ **then**
13.   declare SUCCESS
14.   **else**
15.   declare FAILURE
16.   **end**
17. **end**
18. **if** $(F11 = F1) \wedge (F22 \neq F2)$ **then**
19. $F21 := F2 \setminus F22$
20.
21.
22.
23.   declare FAILURE
24.   **end**
25. **end**

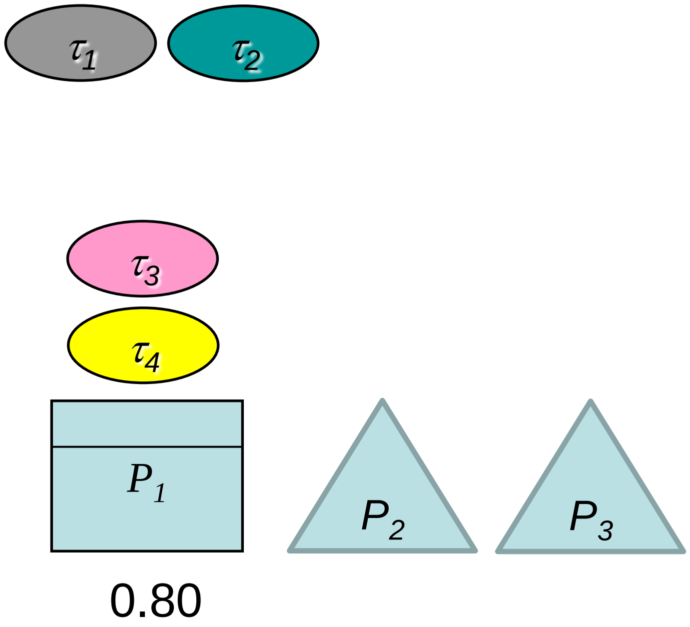
1. **function** first-fit(ts : set of tasks; ps : set of processors) return set of tasks
2.   assigned_tasks := $\emptyset$
3.   If ps consists of type-1 (type-2) processors, then order ts by decreasing U_i^2/U_i^1 (resp., incr. U_i^1/U_i^2). Use any order for processors ps, but maintain it during the execution of the function first-fit.
4. τ_i := first task in ts
5. p := first processor in ps
6.   Let k denote the type of processor p (either 1 or 2)
7.   **if** U[p]+$U_i^k \leq 1$ **then**
8.   U[p] := U[p]+U_i^k
9. τ[p] := τ[p] $\cup \{\tau_i\}$
10.   assigned_tasks := assigned_tasks $\cup \{\tau_i\}$
11.   **if** remaining tasks exist in ts **then**
12. τ_i := next task in ts
13.   go to line 5.
14.   **else**
15.   **return** assigned_tasks
16.   **end if**
17.   **else**
18.   **if** remaining processors exist in ps **then**
19. p := next processor in ps
20.   go to line 6.
21.   else

Let us execute this line.

Applying FF-3C on an example

$\tau=\{\tau_1,\tau_2,\tau_3,\tau_4\}$ $P^1=\{P_1\}$, $P^2=\{P_2,P_3\}$.

	Processor type-1	Processor type-2
τ_1	$U_1^1=0.90$	$U_1^2=0.40$
τ_2	$U_2^1=0.90$	$U_2^2=0.40$
τ_3	$U_3^1=0.40$	$U_3^2=0.80$
τ_4	$U_4^1=0.40$	$U_4^2=0.80$



$\tau^1=\{\tau_3,\tau_4\}$ $H1=\{\tau_3,\tau_4\}$ $F1=\{\}$

$\tau^2=\{\tau_1,\tau_2\}$ $H2=\{\tau_1,\tau_2\}$ $F2=\{\}$

FF-3C

Form sets $H1$, $H2$, $F1$, $F2$
$\forall p$: U[p] := 0
$\forall p$: τ[p] := $\emptyset$
if first-fit($H1$, P^1) $\neq H1$ **then** declare FAILURE
if first-fit($H2$, P^2) $\neq H2$ **then** declare FAILURE
$F11$:= first-fit($F1$, P^1)
$F22$:= first-fit($F2$, P^2)
if $(F11 = F1) \wedge (F22 = F2)$ **then** declare SUCCESS
if $(F11 \neq F1) \wedge (F22 \neq F2)$ **then** declare FAILURE
if $(F11 \neq F1) \wedge (F22 = F2)$ **then**
  $F12 := F1 \setminus F11$
  if first-fit($F12$, P^2) = $F12$ **then**
    declare SUCCESS
  else
    declare FAILURE
  end
end
if $(F11 = F1) \wedge (F22 \neq F2)$ **then**
  $F21 := F2 \setminus F22$
    declare FAILURE
  end
end

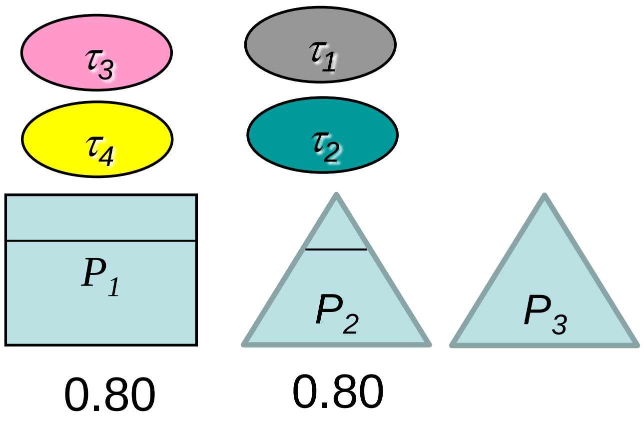
1. **function** first-fit(ts : set of tasks; ps : set of processors) return set of tasks
2. assigned_tasks := $\emptyset$
3. If ps consists of type-1 (type-2) processors, then order ts by decreasing U_i^2/U_i^1 (resp., incr. U_i^1/U_i^2). Use any order for processors ps, but maintain it during the execution of the function first-fit.
4. τ_i := first task in ts
5. p := first processor in ps
6. Let k denote the type of processor p (either 1 or 2)
7. **if** U[p]+$U_i^k \leq 1$ **then**
8. U[p] := U[p]+U_i^k
9. τ[p] := τ[p] $\cup \{\tau_i\}$
10. assigned_tasks := assigned_tasks $\cup \{\tau_i\}$
11. **if** remaining tasks exist in ts **then**
12. τ_i := next task in ts
13. go to line 5.
14. **else**
15. **return** assigned_tasks
16. **end if**
17. **else**
18. **if** remaining processors exist in ps **then**
19. p := next processor in ps
20. go to line 6.

Let us execute this line.

Applying FF-3C on an example

$\tau = \{\tau_1, \tau_2, \tau_3, \tau_4\}$ $P^1 = \{P_1\}$, $P^2 = \{P_2, P_3\}$.

	Processor type-1	Processor type-2
τ_1	$U_1^1=0.90$	$U_1^2=0.40$
τ_2	$U_2^1=0.90$	$U_2^2=0.40$
τ_3	$U_3^1=0.40$	$U_3^2=0.80$
τ_4	$U_4^1=0.40$	$U_4^2=0.80$



$\tau^1 = \{\tau_3, \tau_4\}$ $H1 = \{\tau_3, \tau_4\}$ $F1 = \{\}$

$\tau^2 = \{\tau_1, \tau_2\}$ $H2 = \{\tau_1, \tau_2\}$ $F2 = \{\}$

FF-3C

Form sets $H1$, $H2$, $F1$, $F2$
$\forall p$: U[p] := 0
$\forall p$: τ[p] := $\emptyset$
if first-fit($H1$, P^1) $\neq H1$ **then** declare FAILURE
if first-fit($H2$, P^2) $\neq H2$ **then** declare FAILURE
$F11$:= first-fit($F1$, P^1)
$F22$:= first-fit($F2$, P^2)
if $(F11 = F1) \wedge (F22 = F2)$ **then** declare SUCCESS
if $(F11 \neq F1) \wedge (F22 \neq F2)$ **then** declare FAILURE
if $(F11 \neq F1) \wedge (F22 = F2)$ **then**
  $F12 := F1 \setminus F11$
  if first-fit($F12$, P^2) = $F12$ **then**
    declare SUCCESS
  else
    declare FAILURE
  end
end
if $(F11 = F1) \wedge (F22 \neq F2)$ **then**
  $F21 := F2 \setminus F22$

end

1. **function** first-fit(ts : set of tasks; ps : set of processors) return set of tasks
2. assigned_tasks := $\emptyset$
3. If ps consists of type-1 (type-2) processors, then order ts by decreasing U_i^2/U_i^1 (resp., incr. U_i^1/U_i^2). Use any order for processors ps, but maintain it during the execution of the function first-fit.
4. τ_i := first task in ts
5. p := first processor in ps
6. Let k denote the type of processor p (either 1 or 2)
7. **if** U[p]+$U_i^k \leq 1$ **then**
8. U[p] := U[p]+U_i^k
9. τ[p] := τ[p] $\cup \{\tau_i\}$
10. assigned_tasks := assigned_tasks $\cup \{\tau_i\}$
11. **if** remaining tasks exist in ts **then**
12. τ_i := next task in ts
13. go to line 5.
14. **else**
15. **return** assigned_tasks
16. **end if**
17. **else**
18. **if** remaining processors exist in ps **then**
19. p := next processor in ps
20. go to line 6.

Since F1=∅ and F2= ∅, nothing happens when these lines are executed.

FF-3C

Form sets $H1$, $H2$, $F1$, $F2$
$\forall p$: U[p] := 0
$\forall p$: τ[p] := $\emptyset$
if first-fit($H1$, P^1) $\neq H1$ **then** declare FAILURE
if first-fit($H2$, P^2) $\neq H2$ **then** declare FAILURE
$F11$:= first-fit($F1$, P^1)
$F22$:= first-fit($F2$, P^2)
if $(F11 = F1) \wedge (F22 = F2)$ **then** declare SUCCESS
if $(F11 \neq F1) \wedge (F22 \neq F2)$ **then** declare FAILURE
if $(F11 \neq F1) \wedge (F22 = F2)$ **then**
  $F12 := F1 \setminus F11$
  if first-fit($F12$, P^2) = $F12$ **then**
    declare SUCCESS
  else
    declare FAILURE
  end
end
if $(F11 = F1) \wedge (F22 \neq F2)$ **then**
  $F21 := F2 \setminus F22$
    declare FAILURE
  end
end

1. **function** first-fit(ts : set of tasks; ps : set of processors) return set of tasks
2. assigned_tasks := $\emptyset$
3. If ps consists of type-1 (type-2) processors, then order ts by decreasing U_i^2/U_i^1 (resp., incr. U_i^1/U_i^2). Use any order for processors ps, but maintain it during the execution of the function first-fit.
4. τ_i := first task in ts
5. p := first processor in ps
6. Let k denote the type of processor p (either 1 or 2)
7. **if** U[p]+$U_i^k \leq 1$ **then**
8. U[p] := U[p]+U_i^k
9. τ[p] := τ[p] $\cup \{\tau_i\}$
10. assigned_tasks := assigned_tasks $\cup \{\tau_i\}$
11. **if** remaining tasks exist in ts **then**
12. τ_i := next task in ts
13. go to line 5.
14. **else**
15. **return** assigned_tasks
16. **end if**
17. **else**
18. **if** remaining processors exist in ps **then**
19. p := next processor in ps
20. go to line 6.

The algorithm terminates here.

Theorem 1: The speed competitive ratio of FF-3C is at most two.

A task set τ is feasible on a computing platform π ➔ FF-3C schedules τ on the computing platform 2* π

Algorithm FF-4C and FF-4C-NTC and FF-4C-COMB: like FF-3C but with improved average-case performance

Related Work

- Formulate the problem as Integer Linear Program
  - Minimize U subject to:
    1. $\sum_{j=1}^{m} x_{i,j} = 1$, (i = 1,2,...,n)
    2. $\sum_{i=1}^{n} (x_{i,j} * u_{i,j}) <= U$, (j = 1,2,...,m)
    3. $x_{i,j} = 0$ or $x_{i,j} = 1$ (i = 1,2,...,n); (j = 1,2,...,m)

Related Work

- Formulate the problem as Integer Linear Program
  - Minimize U subject to:
    1. $\sum_{j=1}^{m} x_{i,j} = 1$, (i = 1,2,...,n)
    2. $\sum_{i=1}^{n} (x_{i,j} * u_{i,j}) <= U$, (j = 1,2,...,m)
    3. $x_{i,j} = 0$ or $x_{i,j} = 1$ (i = 1,2,...,n); (j = 1,2,...,m)
  - NP-complete: cannot be solved in polynomial time

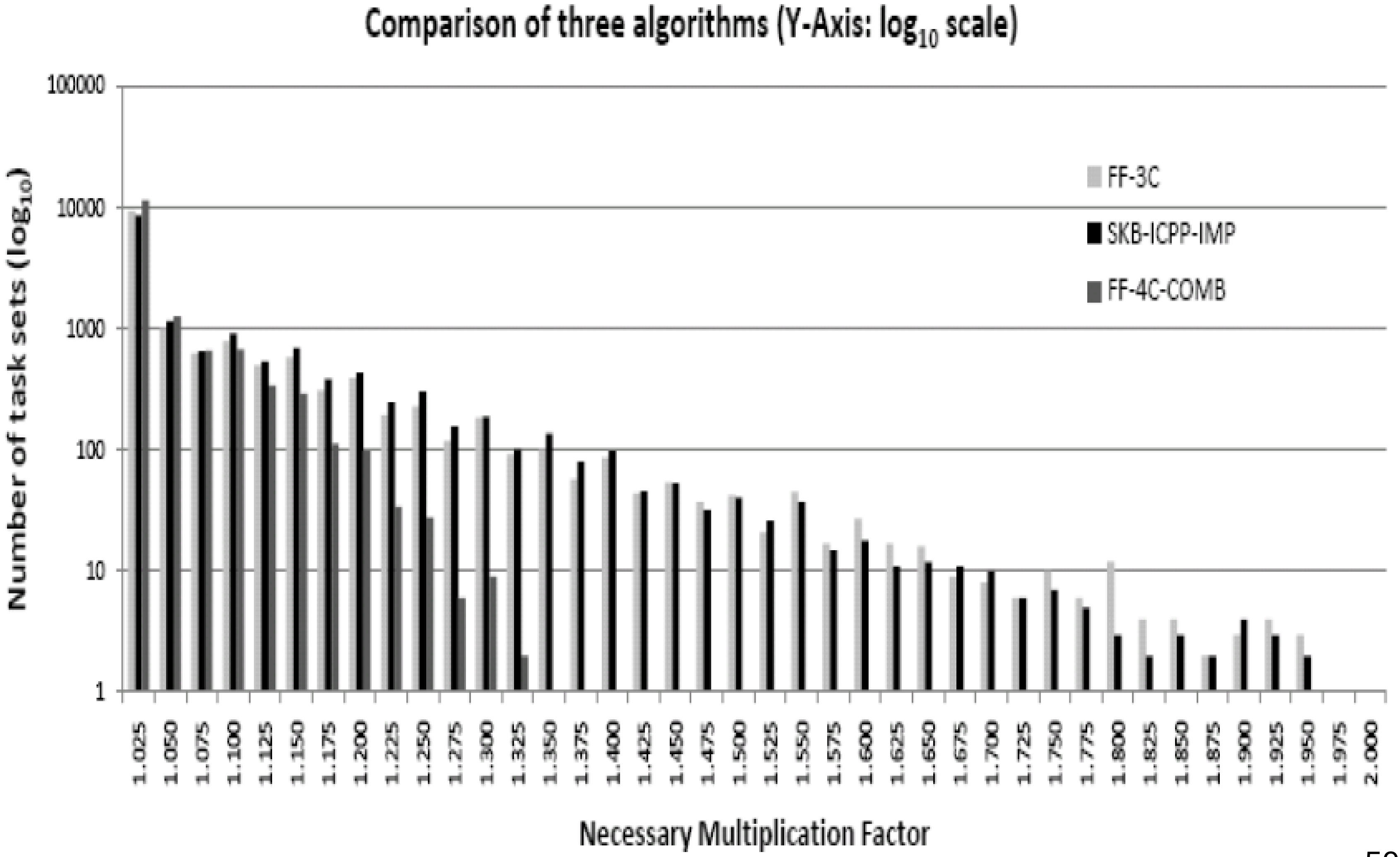
Related Work

- Formulate the problem as Integer Linear Program
  - Minimize U subject to:
    1. $\sum_{j=1}^{m} x_{i,j} = 1$, (i = 1,2,...,n)
    2. $\sum_{i=1}^{n} (x_{i,j} * u_{i,j}) <= U$, (j = 1,2,...,m)
    3. $x_{i,j} = 0$ or $x_{i,j} = 1$ (i = 1,2,...,n); (j = 1,2,...,m)
  - NP-complete: cannot be solved in polynomial time
- Relax it to Linear Programming
    3. $0 <= x_{i,j} <= 1$ (i = 1,2,...,n); (j = 1,2,...,m)
  - Solvable in polynomial time
  - At most 'm' *fractional* tasks

Related Work

- Formulate the problem as Integer Linear Program
  - Minimize U subject to:
    1. $\sum_{j=1}^{m} x_{i,j} = 1,$ (i = 1,2,...,n)
    2. $\sum_{i=1}^{n} (x_{i,j} * u_{i,j}) <= U,$ (j = 1,2,...,m)
    3. $x_{i,j} = 0$ or $x_{i,j} = 1$ (i = 1,2,...,n); (j = 1,2,...,m)
  - NP-complete: cannot be solved in polynomial time
- Relax it to Linear Programming
    3. $0 <= x_{i,j} <= 1$ (i = 1,2,...,n); (j = 1,2,...,m)
  - Solvable in polynomial time
  - At most 'm' *fractional* tasks
- Assign the fractional tasks *integrally*
  - Exhaustive enumeration (RTAS04)
  - Bi-partite matching (ICPP04)

Average-case performance evaluation



Average-case performance evaluation

	New Algorithms				Old Algorithms							
	Measured avg exec time				Measured avg exec time incl CPLEX overhead				Measured avg exec time incl CPLEX overhead — avg CPLEX overhead			
Multiplication factor	FF-3C	FF-4C	FF-4C -NTC	FF-4C -COMB	SKB-RTAS	SKB-RTAS -IMP	SKB-ICPP	SKB-ICPP -IMP	SKB-RTAS	SKB-RTAS -IMP	SKB-ICPP	SKB-ICPP -IMP
1.00	0.85	0.76	0.93	1.08	32481.61	32545.39	394715.80	369120.15	14324.45	14388.23	164603.39	161727.00
1.25	0.52	0.52	0.51	0.53	31657.49	31572.03	393758.65	325045.97	13500.33	13414.87	163646.24	149405.05
1.50	0.49	0.49	0.45	0.48	31751.65	31729.69	381899.86	297359.20	13594.49	13572.52	161185.38	140149.17
1.75	0.47	0.46	0.42	0.46	31744.69	31582.66	337182.98	290084.67	13587.52	13425.49	151049.23	137254.26
2.00	0.49	0.48	0.40	0.48	31736.95	31768.30	291714.93	287719.46	13579.79	13611.13	137972.10	136531.41

Table 1. Comparison of average execution time of algorithms (in microseconds)

Conclusions

+ Bin-packing is possible, with good performance, on heterogeneous multiprocessors with two types of processors.
+ Such bin-packing performs well.

Conclusions

+ Bin-packing is possible, with good performance, on heterogeneous multiprocessors with two types of processors.
+ Such bin-packing performs well:
  * FF-3C has speed competitive ratio at most two;
  * FF-4C-COMB has speed competitive ratio at most two;
  * FF-4C-COMB requires on average processors of lower speed than the previously best algorithm;
  * FF-4C-COMB runs more than 10000 times faster than previously best known algorithm.

Recent extensions to the work

- **Theorem 2**: The speed competitive ratio of FF-3C is at most 1/(1-a)
  - ‘a’ is the maximum utilization of a task

- FF-4C and FF-4C-NTC and FF-4C-COMB
  - like FF-3C but with improved average-case performance

Thank You!