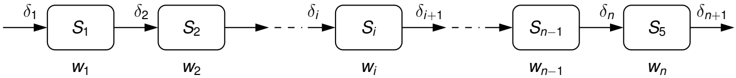
Optimizing Buffer Sizes for Pipeline Workflow Scheduling with Setup Times

Anne Benoit, Jean-Marc Nicod and Veronika Rehn-Sonigo

ENS Lyon – FEMTO-ST institute Besançon, France

Introduction and Motivation

Overall objective: Mapping linear workflow applications, such as image processing or assembly lines, onto parallel platforms

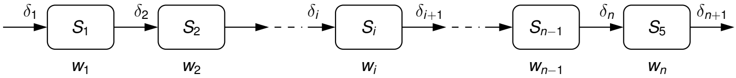


Characteristics:

- The application can be expressed as an ordered sequence of steps (or *stages*)

Introduction and Motivation

Overall objective: Mapping linear workflow applications, such as image processing or assembly lines, onto parallel platforms



Characteristics:

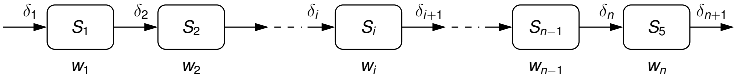
- The application can be expressed as an ordered sequence of steps (or *stages*)

Results for throughput maximization:

- Homogeneous platforms: dynamic program [Subhlok, Vondran 1995,1996]
- Heterogeneous communications: NP-hard [Benoit, Robert 2008]

Introduction and Motivation

Overall objective: Mapping linear workflow applications, such as image processing or assembly lines, onto parallel platforms

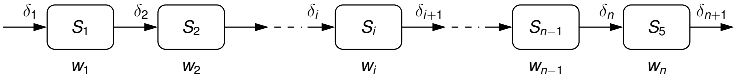


Characteristics:

- The application can be expressed as an ordered sequence of steps (or *stages*)
- If a processor is set to perform multiple stages, a setup cost is required to switch between stages

Introduction and Motivation

Overall objective: Mapping linear workflow applications, such as image processing or assembly lines, onto parallel platforms



Characteristics:

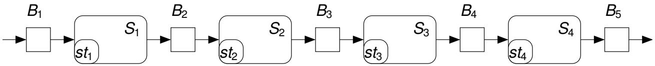
- The application can be expressed as an ordered sequence of steps (or *stages*)
- If a processor is set to perform multiple stages, a setup cost is required to switch between stages

In this talk, we focus on the inner scheduling problem with setup costs

Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

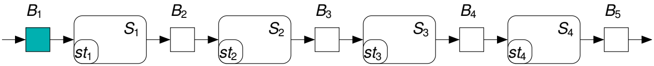
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

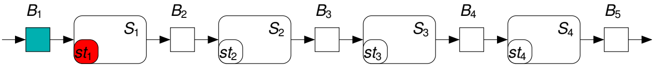
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

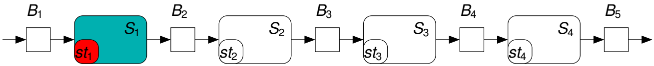
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

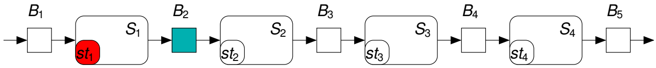
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

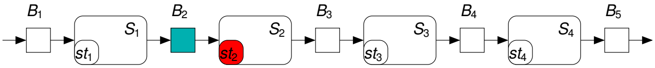
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

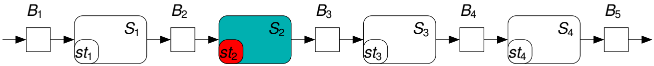
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

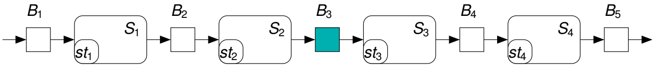
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

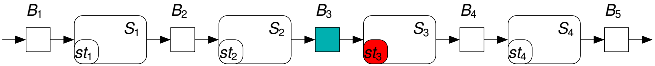
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

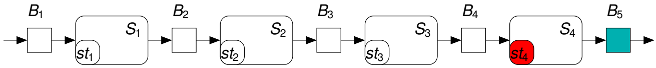
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

Goal: find an optimal inner schedule

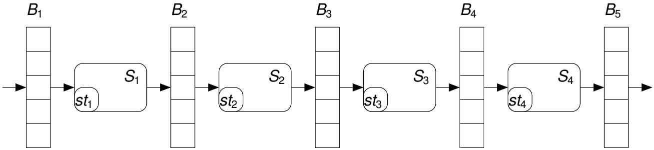


$\Rightarrow$ to output 1 data set, we need 4 setups

Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

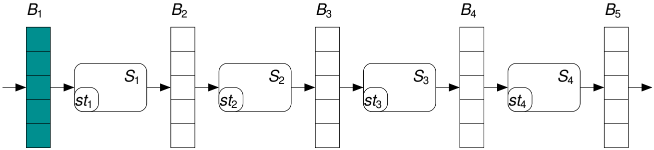
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

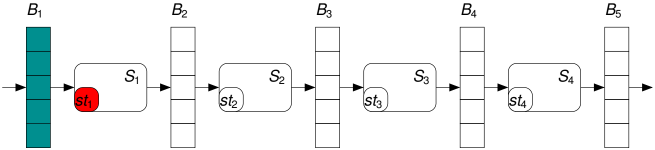
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

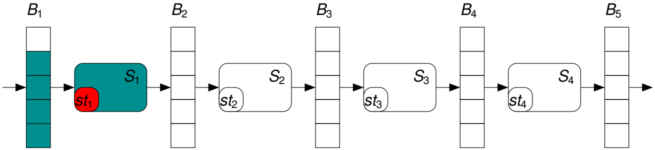
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

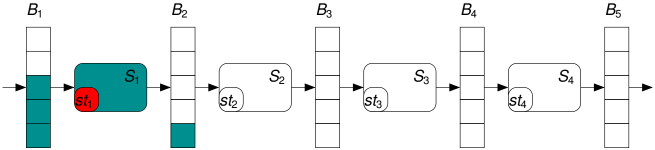
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

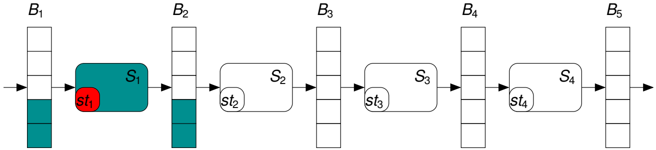
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

Goal: find an optimal inner schedule

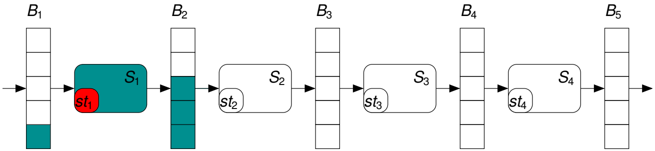


Veronika.Sonigo@femto-st.fr

Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

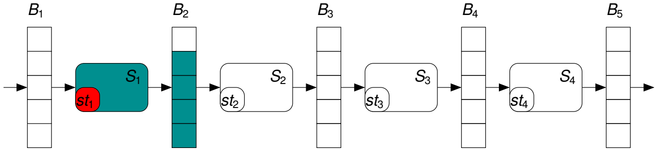
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

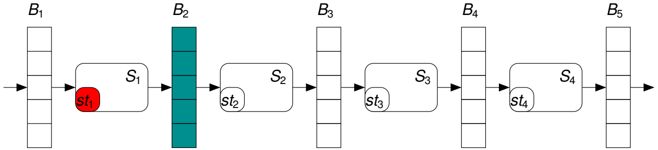
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

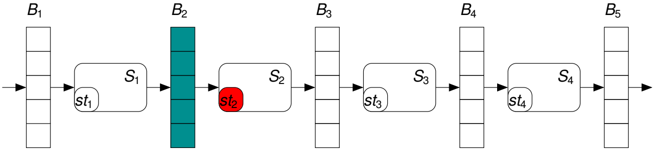
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

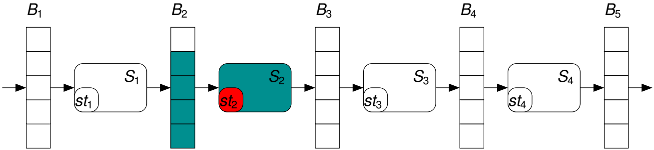
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

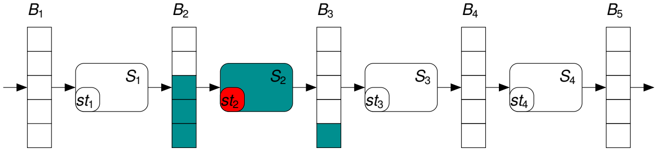
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

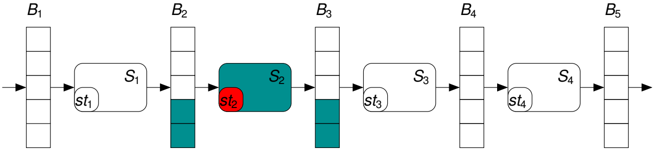
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

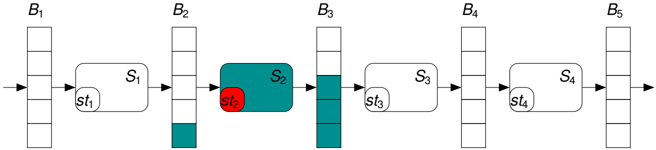
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

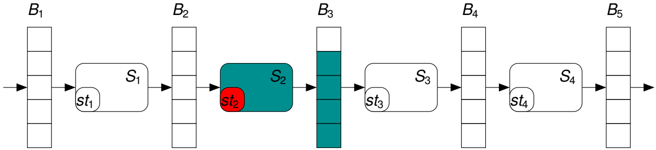
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

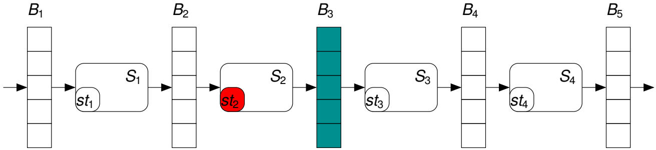
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

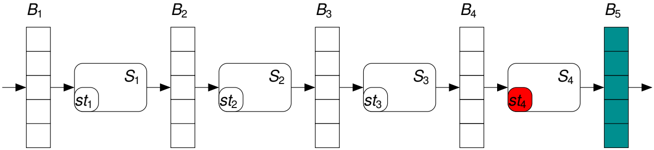
Goal: find an optimal inner schedule



Inner scheduling problem with setup costs

- A single processor is in charge of a linear chain of stages
- A set of buffers can hold in memory some data sets between two consecutive data sets
- Decide in which order each data set and each stage has to be executed, so that the throughput is maximized

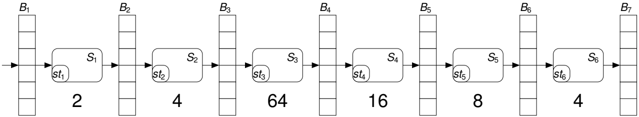
Goal: find an optimal inner schedule



$\Rightarrow$ to output 1 data set, we need $4/5$ setups in average

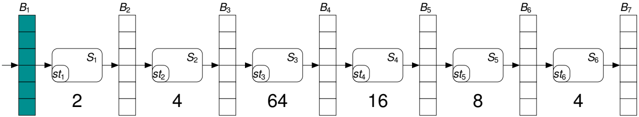
Principle

- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



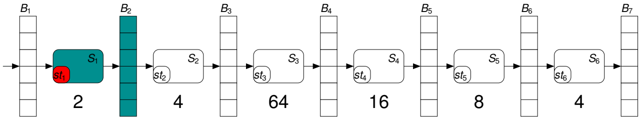
Principle

- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



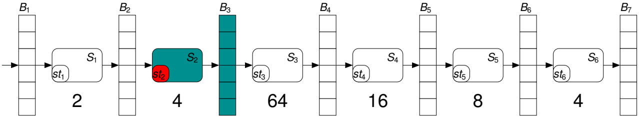
Principle

- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



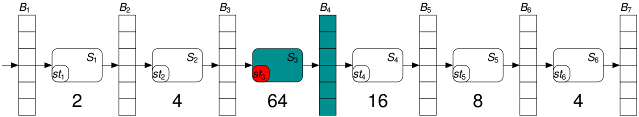
Principle

- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



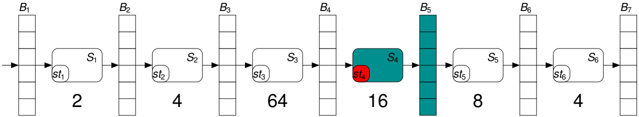
Principle

- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



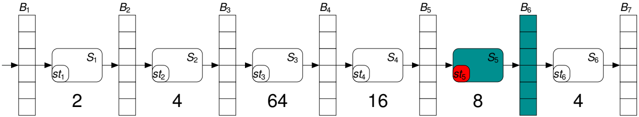
Principle

- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



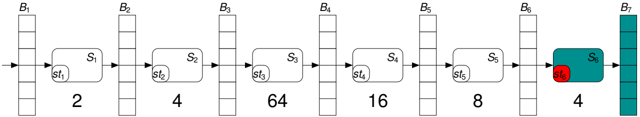
Principle

- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



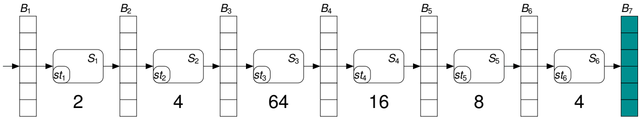
Principle

- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



Principle

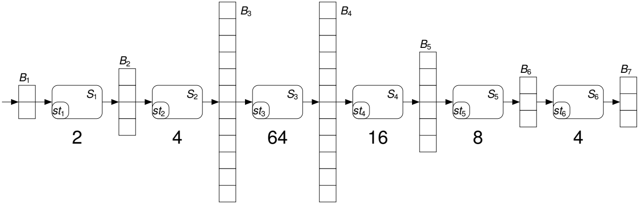
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$$

Principle

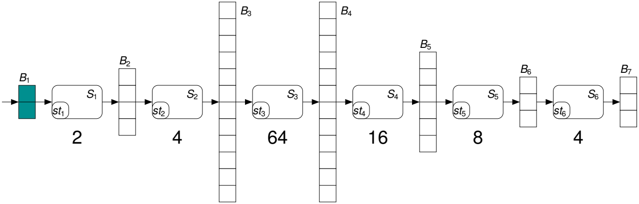
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$$

Principle

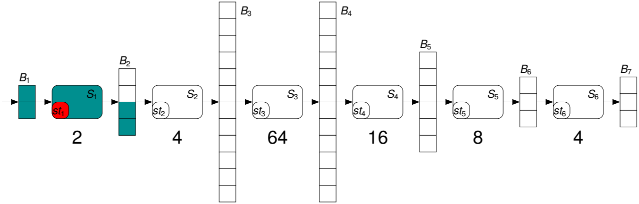
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$\Rightarrow$ $COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$

Principle

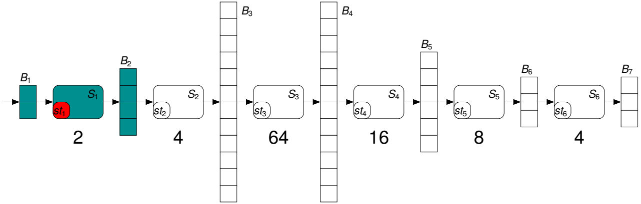
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$

Principle

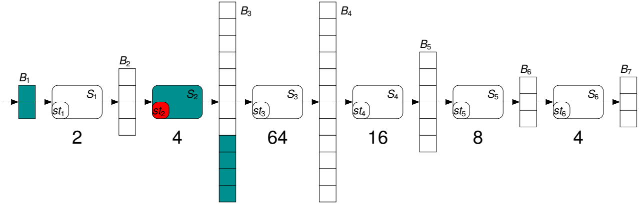
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$$

Principle

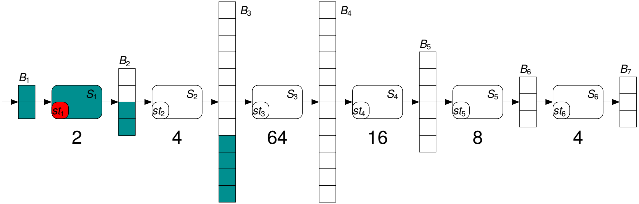
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$$

Principle

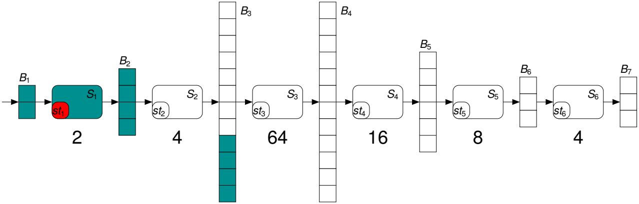
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$

Principle

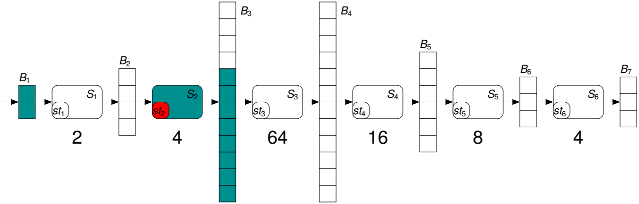
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$

Principle

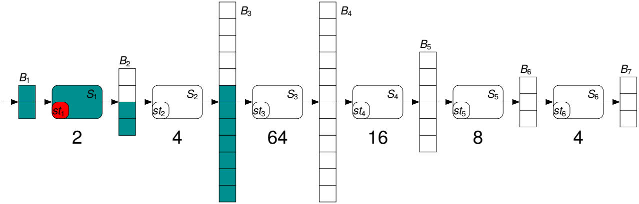
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$$

Principle

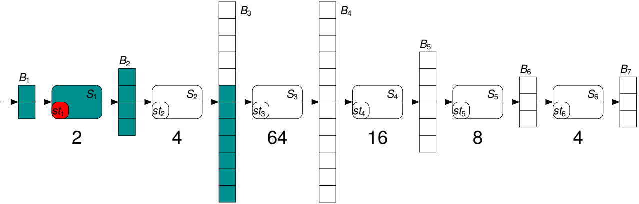
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$\Rightarrow$ $COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$

Principle

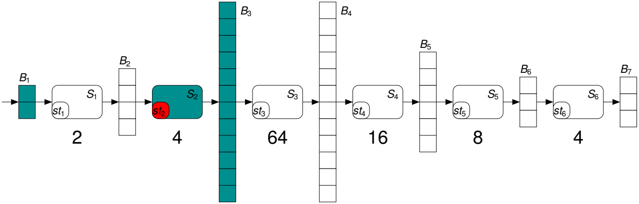
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$

Principle

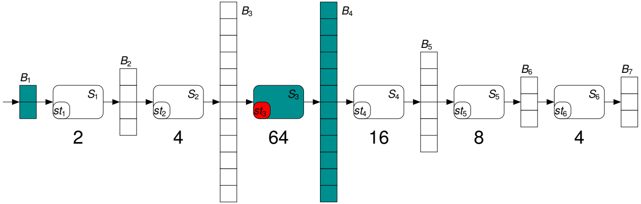
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$

Principle

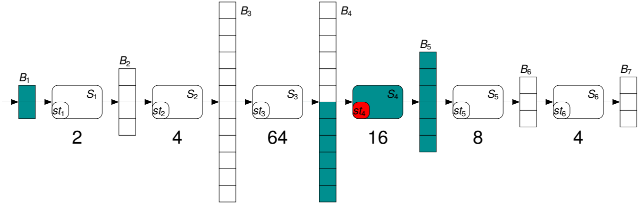
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$\Rightarrow$ $COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$

Principle

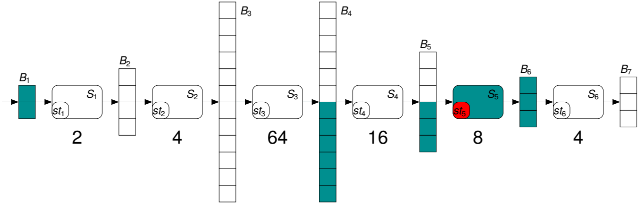
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$$

Principle

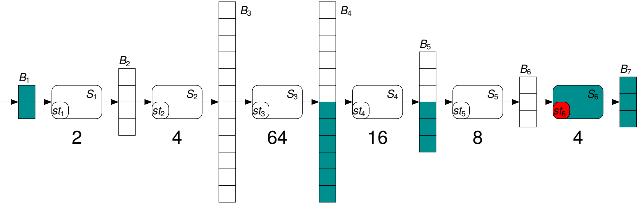
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$\Rightarrow$ $COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$

Principle

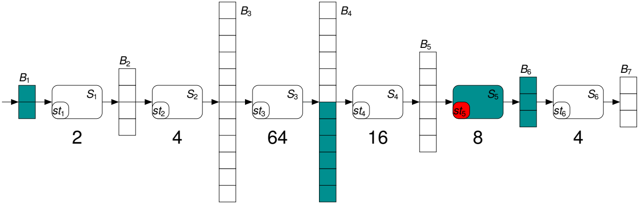
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$

Principle

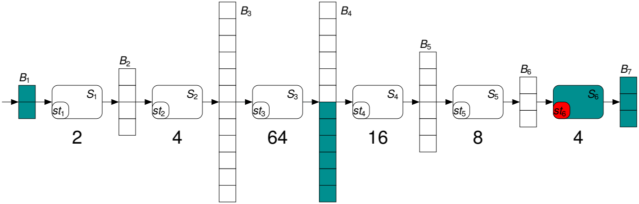
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$

Principle

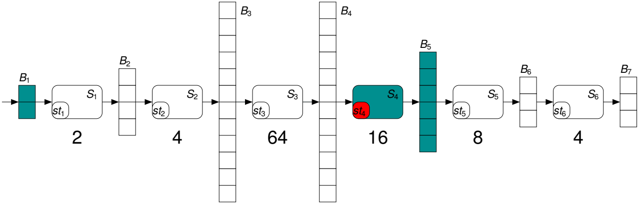
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$$

Principle

- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?

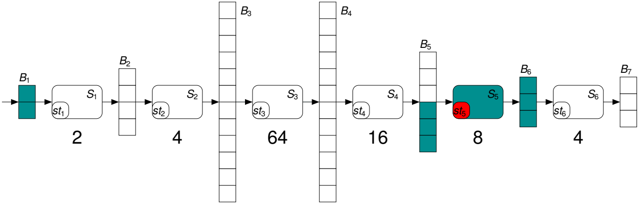


$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$

Veronika.Sonigo@femto-st.fr

Principle

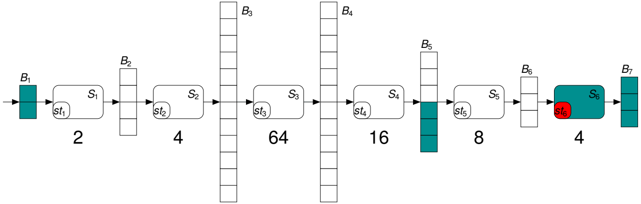
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$

Principle

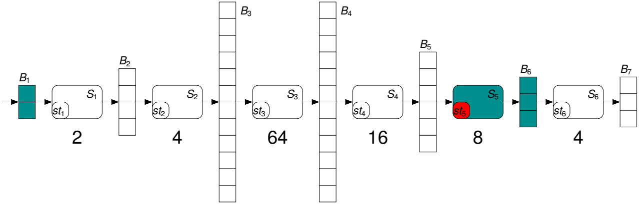
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$$\Rightarrow \textit{COST HOM} = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$$

Principle

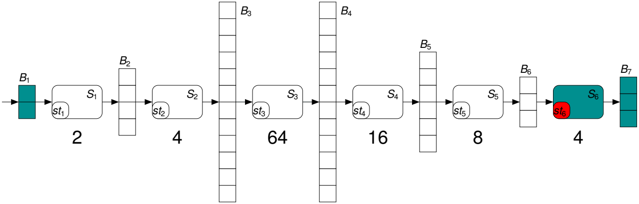
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$

Principle

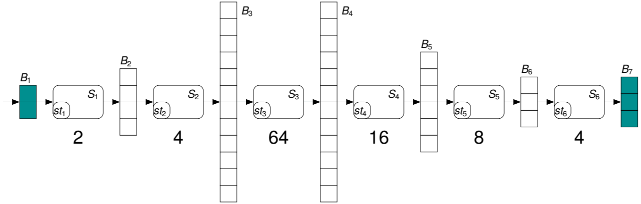
- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$\Rightarrow COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$

Principle

- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?



$\Rightarrow$ $COST\ HOM = 2/6 + 4/6 + 64/6 + 16/6 + 8/6 + 4/6 = 98/6 = 16.33$
$\Rightarrow$ $COST = 2/2 + 4/4 + 64/12 + 16/6 + 8/3 + 4/3 = 168/12 = 14$

Principle

- Homogeneous setup costs: polynomial algorithm [Benoit et al. 2012]
- When setup costs are heterogeneous ?

Memory constraint:

$$\sum_{i=1}^{n+1} \delta_i \times b_i \leq M$$

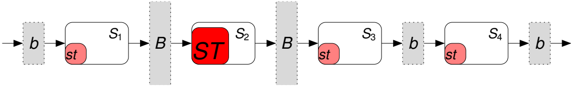
Cost function:

$$C = \sum_{i=1}^{n} \frac{st_i}{\min(b_i, b_{i+1})}$$

Goal: Minimize the cost function, given the memory constraint

$\Rightarrow$ Decide about buffer allocation

Case study



- One task has a larger setup cost than the others, denoted *ST*
- $\delta_i = 1,\ 1 \leq i \leq n+1$

Cost:

$$C = \frac{st}{b} \times (n-1) + \frac{ST}{B}$$

Memory constraint:

$$M \geqslant (n-1)b + 2B$$

- An efficient schedule can be found only if two consecutive buffers are multiples [Benoit at al. 2012]:
  $B = \alpha \times b$, where α is an integer (and $\alpha \geq 1$)

  Bound: $\alpha \leqslant \left\lfloor \frac{M-(n-1)}{2} \right\rfloor$, if $b = 1$
- Replace B by $\alpha \times b$: $b \leqslant \frac{M}{(n-1)+2\alpha}$

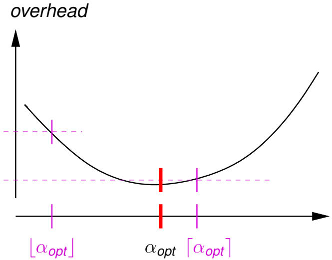
Case study

Assumption: b can be rational: $b = \frac{M}{(n-1)+2\alpha}$

- The cost can then be expressed as a function of α:

$$C(\alpha) = \frac{1}{M}\left(\frac{ST(n-1+2\alpha)}{\alpha} + st(n-1+2\alpha)(n-1)\right)$$

- $C'(\alpha) = \frac{n-1}{M}\left(2st - \frac{ST}{\alpha^2}\right)$
- $C'(\alpha)$ is
  - decreasing for $1 \leq \alpha \leq \sqrt{\frac{ST}{2st}} = \alpha_{opt}$
  - increasing for $\alpha \geq \alpha_{opt}$
- If $\alpha_{opt} > \left\lfloor \frac{M-(n-1)}{2} \right\rfloor$, then we let $\alpha_{opt} = \left\lfloor \frac{M-(n-1)}{2} \right\rfloor$



- Compute the optimal integer values of b and B for $\alpha = \lfloor \alpha_{opt} \rfloor$ and $\alpha = \lceil \alpha_{opt} \rceil$, and we keep the choice of α that minimizes the cost

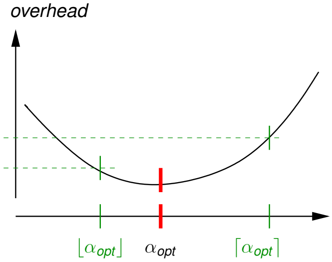
Case study

Assumption: b can be rational: $b = \frac{M}{(n-1)+2\alpha}$

- The cost can then be expressed as a function of α:

$$C(\alpha) = \frac{1}{M}\left(\frac{ST(n-1+2\alpha)}{\alpha} + st(n-1+2\alpha)(n-1)\right)$$

- $C'(\alpha) = \frac{n-1}{M}\left(2st - \frac{ST}{\alpha^2}\right)$
- $C'(\alpha)$ is
  - decreasing for $1 \leq \alpha \leq \sqrt{\frac{ST}{2st}} = \alpha_{opt}$
  - increasing for $\alpha \geq \alpha_{opt}$
- If $\alpha_{opt} > \left\lfloor \frac{M-(n-1)}{2} \right\rfloor$, then we let $\alpha_{opt} = \left\lfloor \frac{M-(n-1)}{2} \right\rfloor$



- Compute the optimal integer values of b and B for $\alpha = \lfloor \alpha_{opt} \rfloor$ and $\alpha = \lceil \alpha_{opt} \rceil$, and we keep the choice of α that minimizes the cost

All setup costs are non-decreasing

- Setup costs are non-decreasing: $st_i \leq st_{i+1} \rightsquigarrow b_i \leq b_{i+1}$
- Cost:

$$C = \sum_{i=1}^{n} \frac{st_i}{\min(b_i, b_{i+1})} = \sum_{i=1}^{n} \frac{st_i}{b_i},$$

- Buffer sizes are multiples two by two: $b_i = \prod_{k=1}^{i} \alpha_k$, for $1 \leq i \leq n+1$
- $b_0 = 1$, $b_i = \alpha_i b_{i-1}$, for $1 \leq i \leq n+1$
- Let $P_a^b = \prod_{\ell=a}^{b} \alpha_\ell$, and $P_a^b = 1$ for $a > b$.
- Due to memory constraint: $\alpha_1 = \frac{M}{\delta_1 + \sum_{k=2}^{n+1} P_2^k \delta_k}$
- $\alpha_i = \sqrt{\frac{\delta_{i-1}}{st_{i-1}} \frac{\sum_{k=i}^{n} st_k P_{k+1}^n}{P_{i+1}^n \sum_{k=i}^{n+1} P_{i+1}^k \delta_k}}$
- $\alpha_n = \sqrt{\frac{\delta_{n-1}}{st_{n-1}} \frac{st_n}{\delta_n + \delta_{n+1}}}$
- $\alpha_{n+1} = 1$ (no gain can be achieved by having a larger last buffer)

The rounding problem remains as the optimal value is rational

General case

Difficulty: it is no longer possible to foresee the value of $\min(b_i, b_{i+1})$

Idea: Reuse of the theoretical results to compute the α_ks:

- Sort setup costs and compute the ratios
- Heuristically decide how to choose integer values of buffer size capacities, while not exceeding the total memory capacity

Design of 7 heuristics

The basic one: SameB

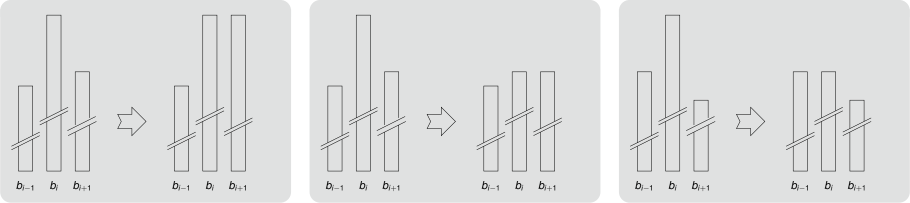
$$b = \left\lfloor \frac{M}{\sum_{i=1}^{n+1} \delta_i} \right\rfloor$$

Heuristics - first series H1

1. Sort the setup values into a non-decreasing order, using a permutation function π such that $st_{\pi(i)} \leq st_{\pi(j)}$ if $\pi(i) < \pi(j)$, for $1 \leq i, j \leq n$
2. Compute the α_k-values backwards
3. Round the α_ks: Flavours: Up, Down, Closest
4. Compute buffer sizes
5. Adapt buffer sizes

Heuristics - first series H1

1. Sort the setup values into a non-decreasing order, using a permutation function π such that $st_{\pi(i)} \leq st_{\pi(j)}$ if $\pi(i) < \pi(j)$, for $1 \leq i, j \leq n$
2. Compute the α_k-values backwards
3. Round the α_ks: Flavours: Up, Down, Closest
4. Compute buffer sizes
5. Adapt buffer sizes



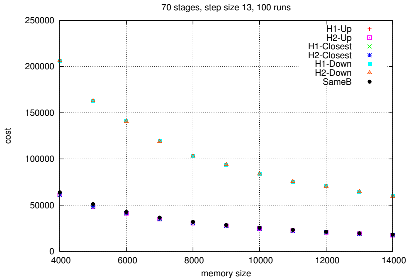
Heuristics - second series H2

Idea: Include the buffer adaption in the ratio computation

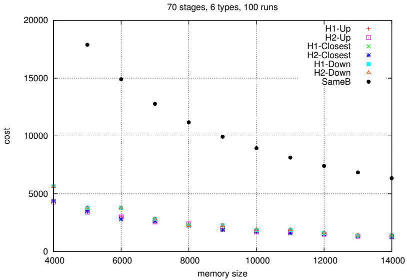
1. Steps 1-4 of H1
2. For each stage S_i with $st_i = \max(st_{i-1}, st_i, st_{i+1})$ $(1 < i < n)$: force b_{i+1} to take the value of b_i
3. Re-evaluate the α_k's by recomputing α_1
4. Flavours: UP, Down, Closest

Results for non-decreasing setup costs

Mean cost over 100 applications



random values

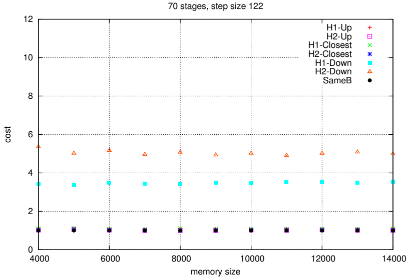


6 setup cost types

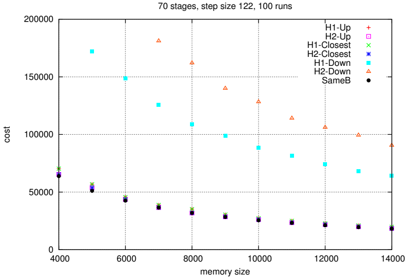
70 stages

Results for the general case (1)

Setup costs are often in the same order of magnitude, tend to zigzag



1 random application

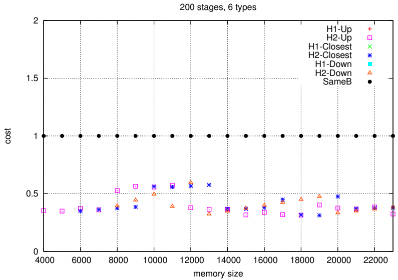


Mean cost over 100 applications

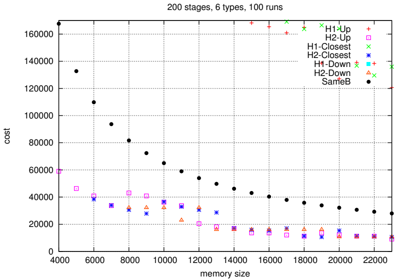
70 stages

Results for the general case (2)

Successive setup costs differ at least one order of magnitude or are the same; peaks appear



1 random application, 6 setup types

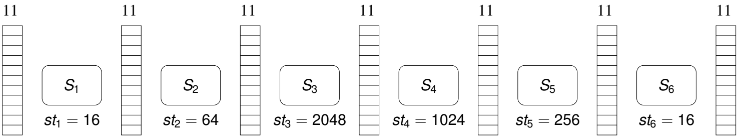


Mean cost over 100 applications

200 stages. Zoom on H2 and SameB

Analyzing one solution

- Memory $M = 77$
- 6 stages



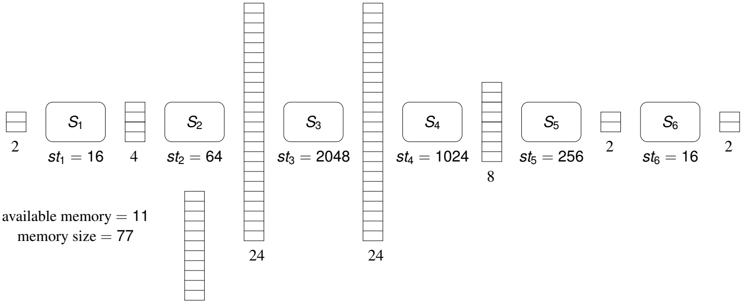
available memory $= \emptyset$

memory $77 = 11 \times 7$

SameB: 11 slots for each buffer $\Rightarrow$ $COST\ HOM = 311.27$

Analyzing one solution

- Memory $M = 77$
- 6 stages

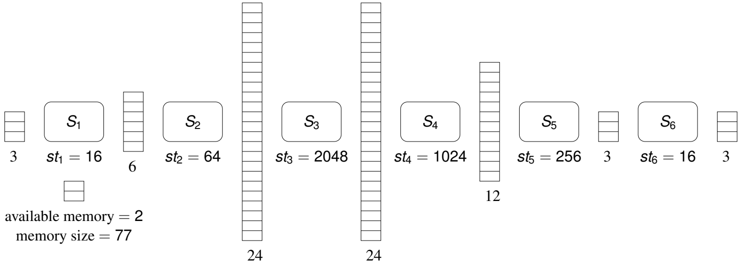


SameB: 11 slots for each buffer $\Rightarrow$ $COST\ HOM = 311.27$

Down: $COST\ DOWN = 373.33$

Analyzing one solution

- Memory $M = 77$
- 6 stages



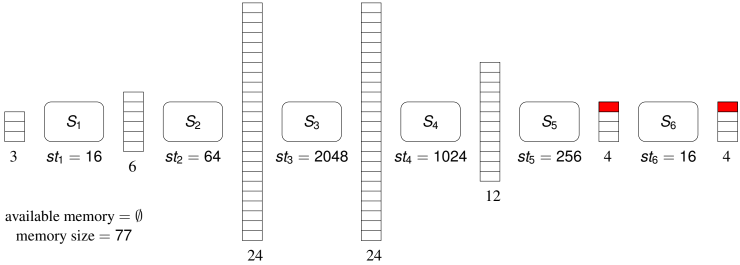
SameB: 11 slots for each buffer $\Rightarrow$ $COST\ HOM = 311.27$

Down: $COST\ DOWN = 373.33$

UP and Closest : $COST\ UP/CLOSEST = 277.33$

Analyzing one solution

- Memory $M = 77$
- 6 stages



SameB: 11 slots for each buffer $\Rightarrow$ $COST\ HOM = 311.27$

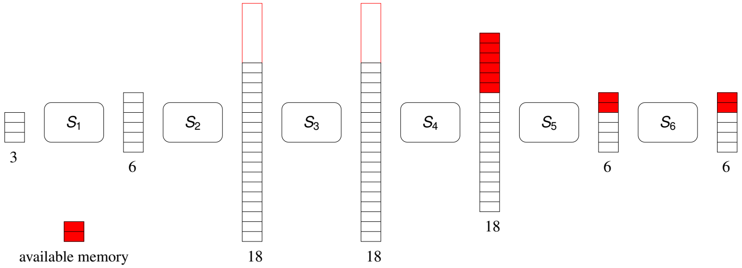
Down: $COST\ DOWN = 373.33$

UP and Closest : $COST\ UP/CLOSEST = 277.33$

Proportional buffers: $COST\ PROP = 254.66$

Analyzing one solution

- Memory $M = 77$
- 6 stages



SameB: 11 slots for each buffer $\Rightarrow$ $COST\ HOM = 311.27$

Down: $COST\ DOWN = 373.33$

UP and Closest : $COST\ UP/CLOSEST = 277.33$

Proportional buffers: $COST\ PROP = 254.66$

Optimal solution: $COST\ OPT = 232$

Summary

Based on the optimal rational solution: proposition of an efficient integer solution

- Importance of the rounding policy
- Applications with little variance:
  SameB heuristic achieves comparable results to the Up and Closest (resp. H1 and H2) heuristics
- Applications with at least one peak:
  - SameB approach fails completely in performance
  - H2 up to 3.3 times better