Budget-aware Static Scheduling of Stochastic Workflows with DIET

Yves Caniou, Eddy Caron, Aurélie Kong Win Chang, Yves Robert

ADVCOMP 2021

Kong Win Chang Aurélie – aurelie.kong-win-chang@inria.fr – Inria

The presenter

Aurélie Kong Win Chang

PhD student formerly in Roma and Avalon teams of LIP/Inria.
When the paper was written, I was working on the scheduling of scientific workflows in Cloud.
I now have started a PhD in SPADES team of Inria, about the use of abstractions for causal analysis and explanations in concurrent programs.

Content

Workflow scheduling in Cloud: simulation

Previously: simulations

- Workflow scheduling algorithms for heterogeneous Cloud
- Constraints: minimizing makespan, respecting a budget
- Experiments: home-made simulator based on simDAG

→ How far (or close) from reality?

→ Wouldn't a real execution outline some interesting points?

Workflow scheduling in Cloud: in real life

Experimenting for real...

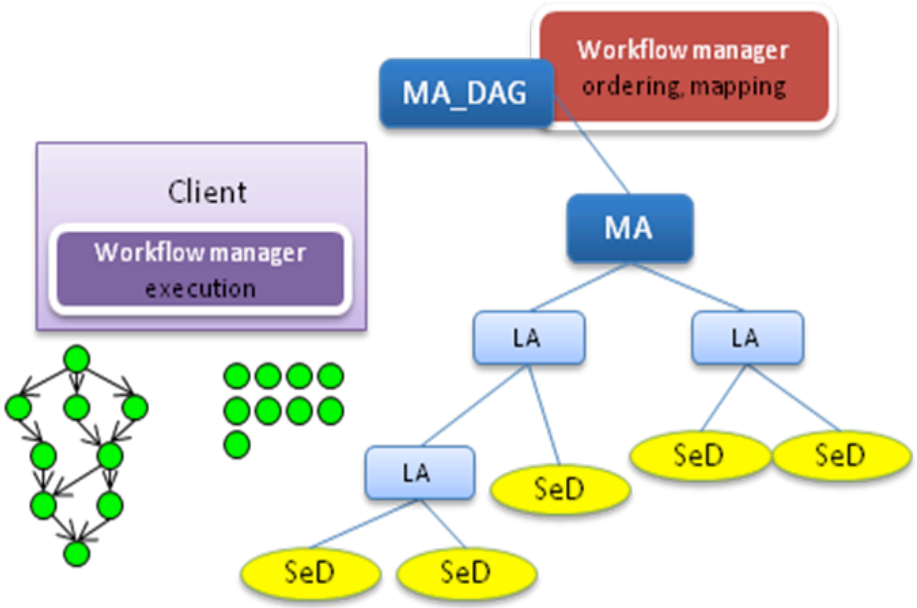
- Platform: the French national validation platform Grid'5000
- Experiments: as close as possible to the simulation study

...based on simulations

- Same workflows
- Similar comparisons

→ Huge amount of experiments: how to automate the process?

DIET with static scheduling 1/2



Hierarchy

- Master Agents and MA_DAG
- Local Agents
- Client

Limits

- Dynamic scheduling only

DIET with static scheduling 2/2

Extended MA_DAG

The client has to give:

- The XML files describing the workflow
- The Desired Schedule File (DSF)
- The Mapping File

The simulator

Uses

- A workflow description
- A platform description
- Experiment parameters

Generates

- The static scheduling
- The simulated results

The workflow generator

Uses

- The same description file of the workflow
- Information about the focused platform
- Information on the simulations
- The static schedule

Generates

- The configuration files
- The workflow itself
- Its data files

Model

Workflow (Dataflow)

Direct Acyclic Graph

- Tasks of a given estimated amount of work
- Dependencies and their given amount of transfered data

Platform

- A Cloud storage
- VMs classified in different categories
- "on-demand" provisioning system

Objective 1/2

Objective

- Minimize the makespan
- Respect the budget

VM	
C_v	total cost for a VM v
$H_{end,v}$, $H_{start,v}$	moment at which a VM v ends/starts
k	number of VM categories
$c_{h,k}$, $c_{ini,k}$	per time-unit cost and initial cost for category k
R_{VM}	set of booked VMs v

Cloud storage	
C_{CS}	total cost for Cloud storage
H_{usage}	time during which the Cloud storage has been used
c_{tsf}	cost of I/O operations
$c_{h,CS}$	per time-unit cost of Cloud storage usage

Objective 2/2

Cost

- Cost for one VM v : $C_v = (H_{end,v} - H_{start,v}) \times c_{h,k_v} + c_{ini,k_v}$
- Cost for the Cloud storage:

$$C_{CS} = (\text{size}(d_{in,CS}) + \text{size}(d_{CS,out})) \times c_{tsf} + H_{usage} \times c_{h,CS}$$

- Total cost of a workflow: $C_{wf} = \sum_{v \in R_{VM}} C_v + C_{CS}$

Algorithms – Principle

Budget distribution

Proportional to:

- The estimated amount of work of the task
- The amount of data involved
- A margin

Tasks affectation

- Amount of money for an affectation = tasks share of the budget + leftovers
- Earliest Finishing Time

Refinement

- Improve the schedule using the leftovers

Algorithms

Algorithms	
BDT	Shares the budget between each level of the workflow.
CG	Distribution of the budget per task, in one pass.
CG/CG+	First distribution *via* CG, then refinement along the critical path until no budget leftover remains unspent.
HEFT	No consideration of the budget. Ranks tasks and follows this ranking to allocate them to VM.
HEFTBudg	Ranks tasks as in HEFT, makes a first attribution of budget for each task, then allocates each task to a VM as in HEFT, but with respect to the budget. Forwards any budget leftover to the next task.
HEFTBudg+	As in HEFTBudg, but once a first allocation has been made, tries to shift the allocated task to a better VM using the budget leftover of the previous allocation. Reiterates until no leftover is left.
HEFTBudg+Inv	Same as in HEFTBudg+, but uses the opposite order for the ranking during reallocation.
HEFTBudgMult	A first HEFTBudg is ran and simulated, then a new allocation is done like in HEFTBudg, but adding leftovers found to the budget of the first task.
MinMin	No consideration of the budget. Allocates the couple <task, VM > with the EFT until all tasks have been allocated.
MinMinBudg	Same as in MinMin, but attributing a part of the budget to each task as in HEFTBudg and using it for the allocation. Same mechanism of forwarding budget leftover as in HEFTBudg.

Grid'5000 and Simulations

Grid'5000

- 31 homogeneous nodes (Intel Xeon E5-2630 v3 CPU 2.4 GHz $\times$ 8 cores)
- Classes of VMs emulated through the amount of work of the tasks
- For the longest workflows, use of proportionnaly shorter ones

Simulations

- Configuration based on experimental measures on Grid'5000

Experimental campain

Scientific Workflows

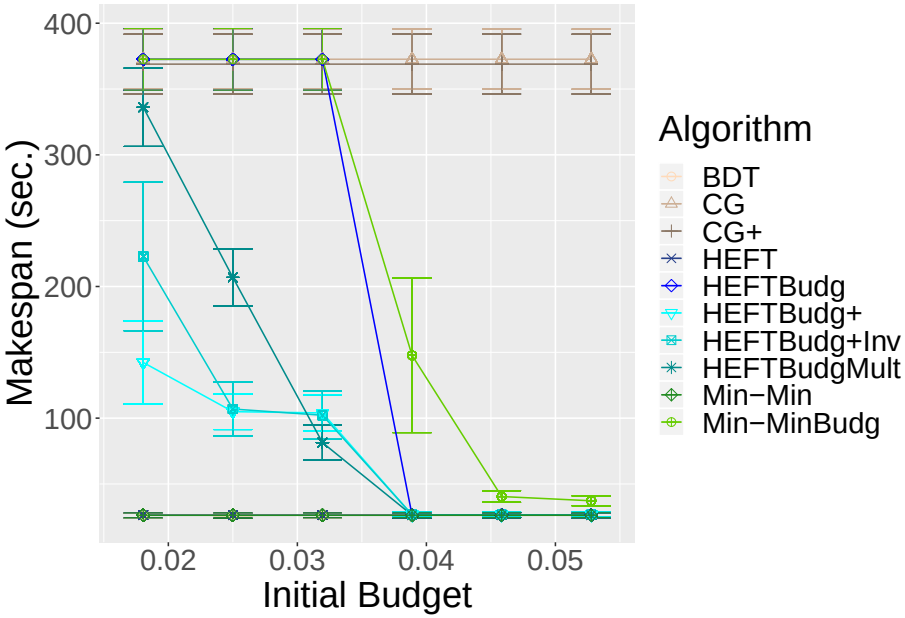
- Montage, Ligo, Cybershake
- 30 tasks

Algorithms

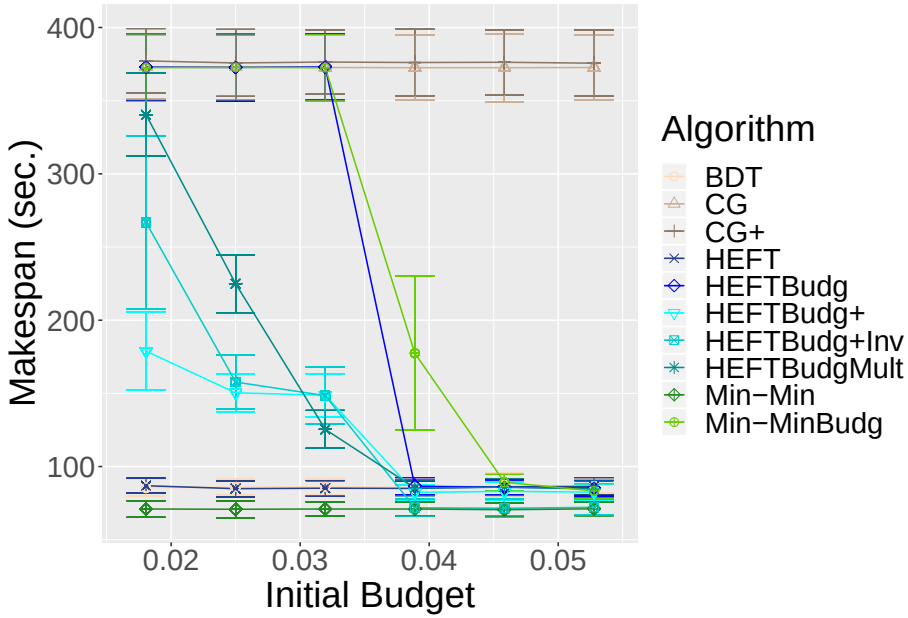
- 10 algorithms described earlier
- Budget: a range of values which have an actual impact on the makespan

$\rightarrow$ Makespan and cost execution for each of the 30 experiments per combination (budget$\times$algorithm$\times$workflow)

Simulation vs. Experiments: Montage



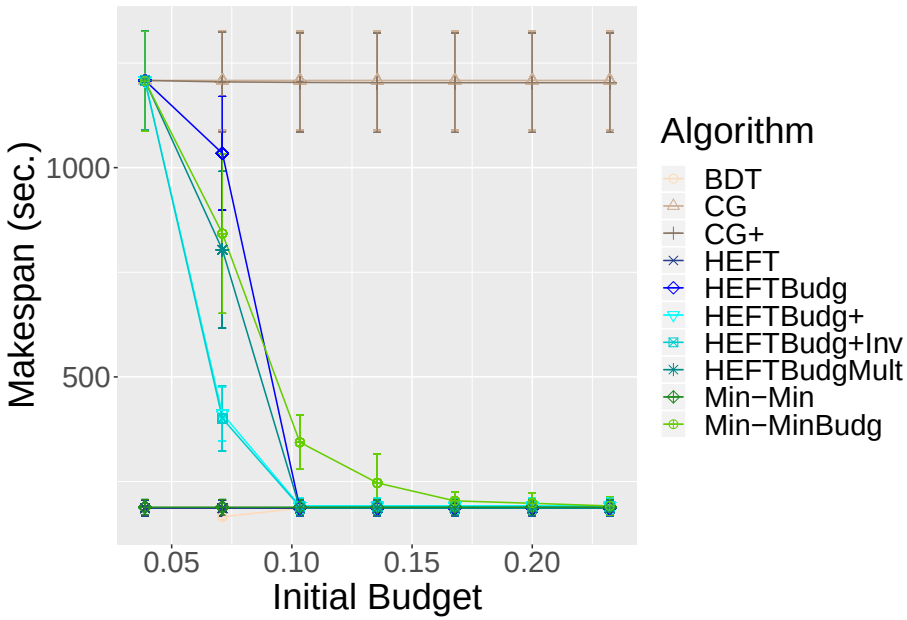
Simulation



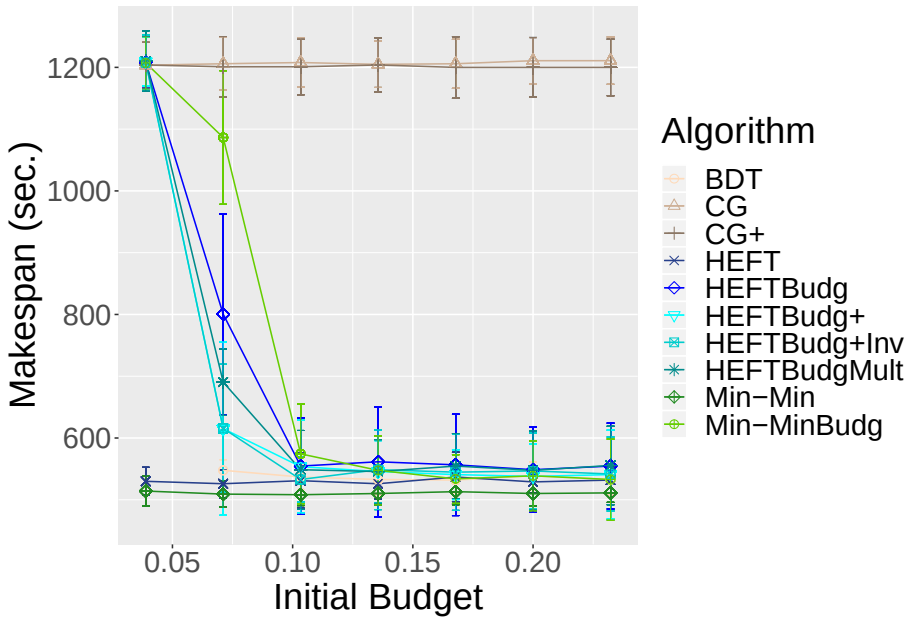
Grid'5000

Montage

Simulation vs. Experiments: Cybershake



Simulation



Grid'5000

Cybershake

Conclusion

Contributions

- Introduction of static scheduling to DIET
- Experimentations on simulator and Grid'5000

Assessment

- Validation of the results from the simulation-based study
- Validation of the DIET improvements
- A potentially interesting new lead