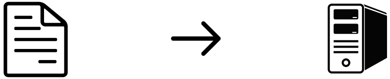
Communication-aware task placement for workflow scheduling on DaaS-based Cloud

Hadrien Croubois, Eddy Caron

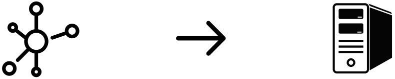
PhD Student at Avalon, Laboratoire de l'informatique du Parallélisme
École Normale Supérieure de Lyon, France

The scheduling problem

Scheduling: Matching jobs and resources

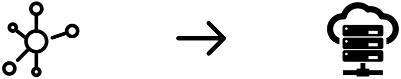


Scheduling: Matching jobs and resources



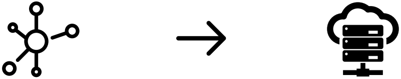
- Jobs definition is changing,

Scheduling: Matching jobs and resources



- Jobs definition is changing,
- Resources are changing.

Scheduling: Matching jobs and resources

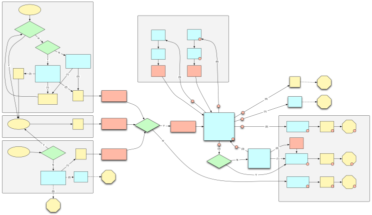


Challenge

The matching logic need to consider chose changes.

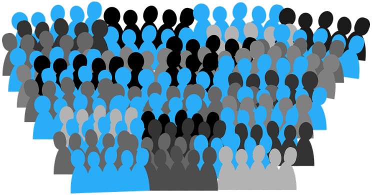
Challenge: Accounting for three factors

- Complex jobs (workflows);



Challenge: Accounting for three factors

- Complex jobs (workflows);
- Multi-Tenant (collaborative);



Challenge: Accounting for three factors

- Complex jobs (workflows);
- Multi-Tenant (collaborative);
- Dynamic platform (IaaS Cloud with DaaS storage).

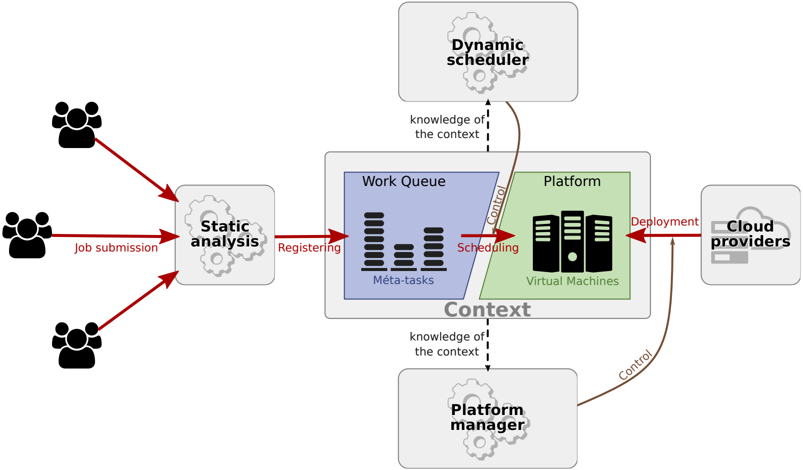
Challenge: Accounting for three factors

- Complex jobs (workflows);
- Multi-Tenant (collaborative);
- Dynamic platform (IaaS Cloud with DaaS storage).

State of the art

Previous work considers at most 2 of those 3 factors.

Problem subdivision



Framework architecture

DCP static scheduling algorithm

```
$\mathcal{C} \leftarrow$ empty clustering                    ▷ (one node per task)
compute $BL$ and $TL$ for each task using $\mathcal{C}$
while $\exists$ unmarked dependency between tasks do
    $(u, v) \leftarrow$ edge with the largest path length (most critical).
Resolve ties by edge size (select largest).
    $\mathcal{C}' \leftarrow \mathcal{C}.mergeClusters(u, v)$
    compute $BL'$ and $TL'$ for each task using $\mathcal{C}'$
    if $DCPL(BL', TL') \leq DCPL(BL, TL)$ then
        $(\mathcal{C}, TL, BL) \leftarrow (\mathcal{C}', TL', BL')$
    end if
    mark $(u, v)$
end while
return $\mathcal{C}$
```

Y.-K. Kwok and I. Ahmad, "A static scheduling algorithm onto multiprocessors," in *Proceedings of the 1994 International Conference on Parallel Processing*.

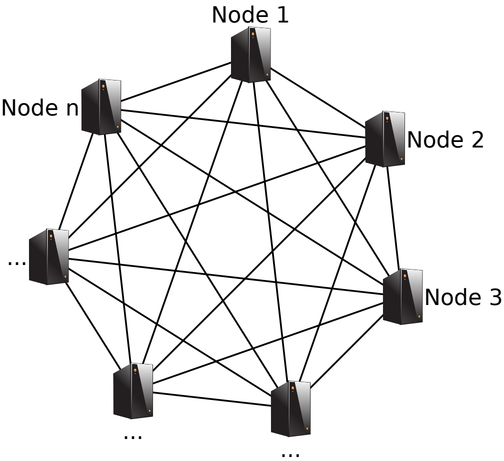
The DCP algorithm

$$c(u,v) = \begin{cases} 0 & \text{if } \mathcal{C}(u) = \mathcal{C}(v) \\ \omega(u \to v) & \text{otherwise} \end{cases}$$

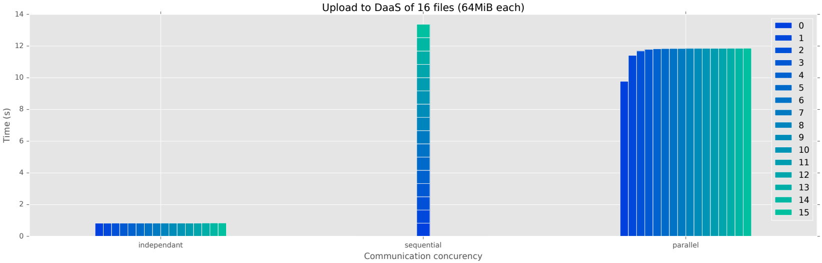
$$TL(v) = \begin{cases} 0 \text{ if } v \text{ has no predecessor} \\ \max\limits_{u \in pred(v)} \big(TL(u) + \omega(u) + c(u,v), \\ \hfill avail_{TL}(\mathcal{C}, v) \big) \end{cases}$$

$$BL(u) = \begin{cases} \omega(u) \text{ if } u \text{ has no successor} \\ \omega(u) + \max\limits_{v \in succ(u)} \big(c(u,v) + BL(v), \\ \hfill avail_{BL}(\mathcal{C}, u) \big) \end{cases}$$

DCP implicit network topology

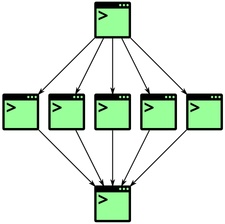


Interference between concurrent communications



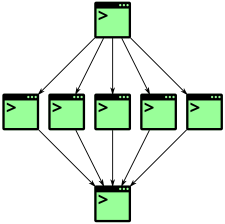
Transferring files between one node in sagittaire cluster (Grid'5000)//and a DaaS (storage5K)

See dependencies from the Dataflow point of view

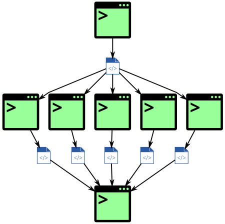

Legacy representation

Representation of a fork-join DAG with $n = 5$ independent jobs.

See dependencies from the Dataflow point of view



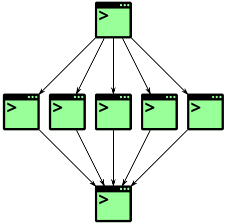
Legacy representation



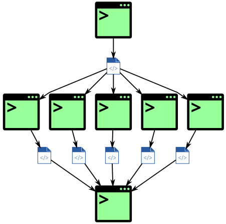
Our representation (with single data upload)

Representation of a fork-join DAG with $n = 5$ independent jobs.

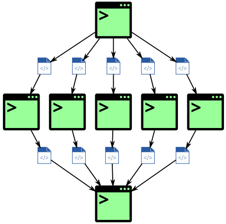
See dependencies from the Dataflow point of view



Legacy representation



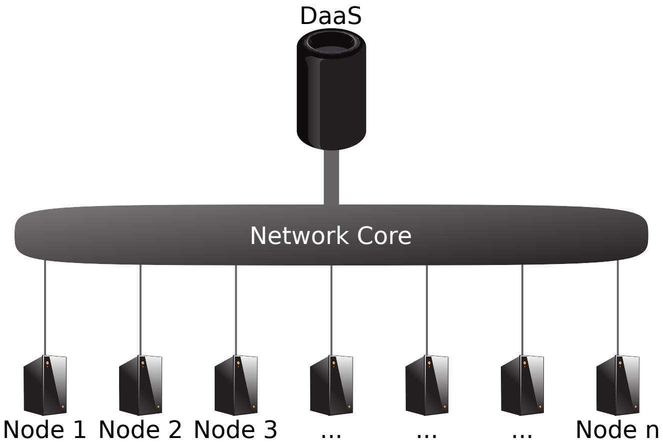
Our representation (with single data upload)



Our representation (with multiple data upload)

Representation of a fork-join DAG with $n = 5$ independent jobs.

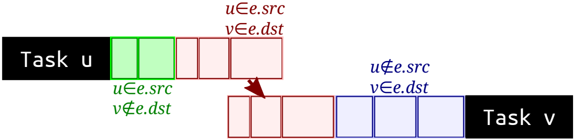
Reconsidering the network topology



A generic model of DaaS-based network topology.

Adapting the Critical Path computation

Communications between two tasks on a DaaS-based platform.



$$c(u,v) = 0 \text{ if } proc(u) = proc(v)$$

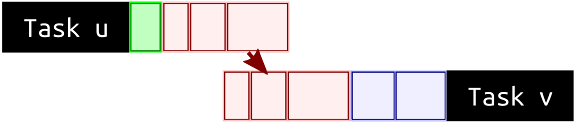
$$\begin{aligned} c(u,v) = & \sum_{\substack{d \in edges \\ u = d.src \\ v \notin d.dst}} \frac{d.size}{network\_up} \\ & + \sum_{\substack{d \in edges \\ u = d.src \\ v \in d.dst}} \frac{d.size}{\min(network\_up, network\_down)} \\ & + \max_{\substack{d \in edges \\ u = d.src \\ v \in d.dst}} \frac{d.size}{\max(network\_up, network\_down)} \\ & + \sum_{\substack{d \in edges \\ u \neq d.src \\ v \in d.dst}} \frac{d.size}{network\_down} \end{aligned}$$

Locality

$$islocal(d, v) = \begin{cases} 1 & \text{if } proc(d.src) = proc(v) \\ 0 & \text{otherwise} \end{cases}$$

$$islocal(d) = \prod_{v \in d.dst} islocal(d, v)$$

Communications between two tasks on a DaaS-based platform (with locality).

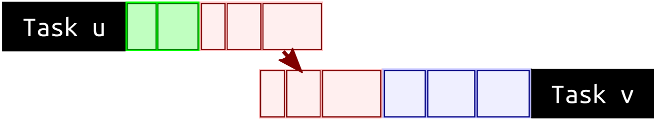


$$c_{loc}(u,v) = 0 \text{ if } proc(u) = proc(v)$$

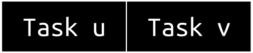
$$\begin{aligned} c_{loc}(u,v) = & \sum_{\substack{d\in edges\\ u=d.src\\ v\notin d.dst\\ islocal(d)=\mathbf{0}}} \frac{d.size}{network\_up} \\ & + \sum_{\substack{d\in edges\\ u=d.src\\ v\in d.dst}} \frac{d.size}{\min(network\_up, network\_down)} \\ & + \max_{\substack{d\in edges\\ u=d.src\\ v\in d.dst}} \frac{d.size}{\max(network\_up, network\_down)} \\ & + \sum_{\substack{d\in edges\\ u\neq d.src\\ v\in d.dst\\ islocal(d,v)=\mathbf{0}}} \frac{d.size}{network\_down} \end{aligned}$$

Adapting the Critical Path computation

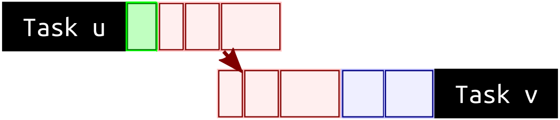
Different nodes w\ no locality, worst case communications i→j



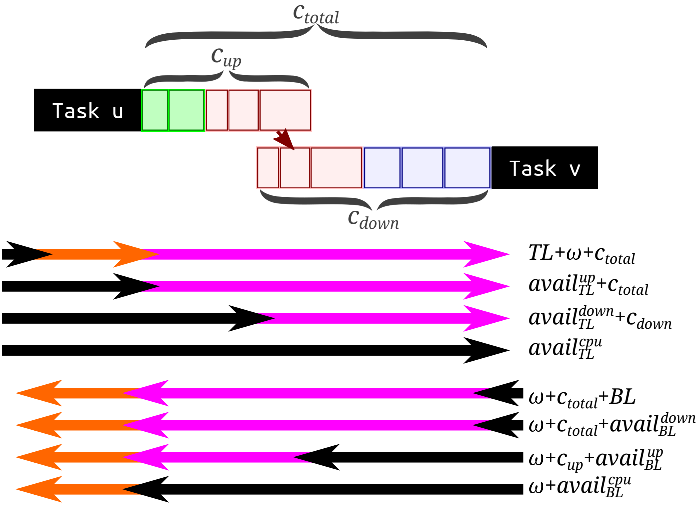
Same node, no communication i→j



Different nodes w\ locality, worst case communication i→j



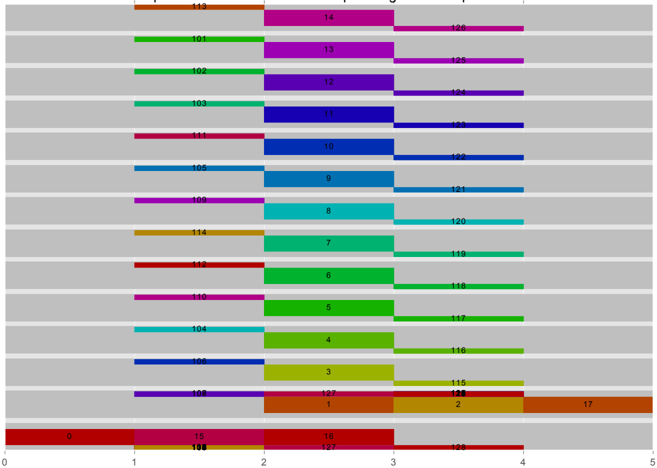
Communication aware task scheduling


c_{total}
c_{up}
Task u
Task v
c_{down}
$TL+\omega+c_{total}$
$avail_{TL}^{up}+c_{total}$
$avail_{TL}^{down}+c_{down}$
$avail_{TL}^{cpu}$
$\omega+c_{total}+BL$
$\omega+c_{total}+avail_{BL}^{down}$
$\omega+c_{up}+avail_{BL}^{up}$
$\omega+avail_{BL}^{cpu}$

Preview of the critical path computation taking the machine network availability into account in DaaS-based platform.

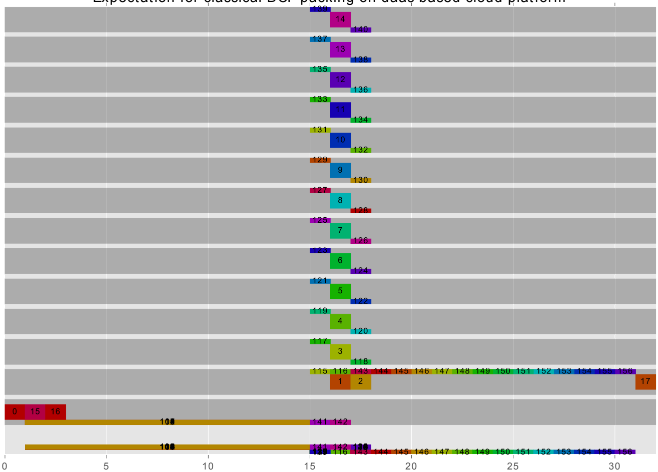
Results

Expectation for classical DCP packing on ideal platform



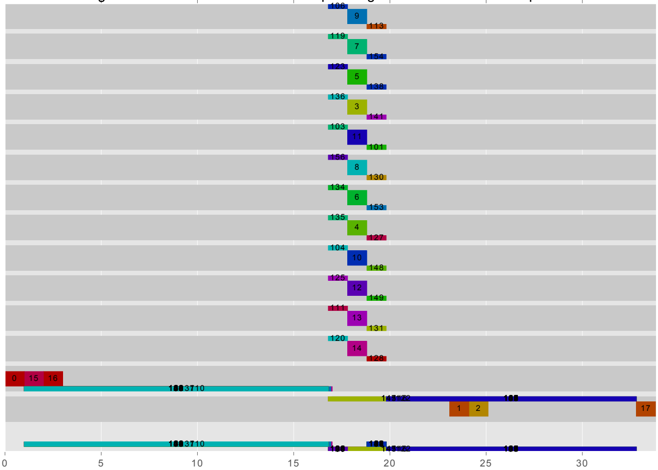
Results

Expectation for classical DCP packing on daas based cloud platform

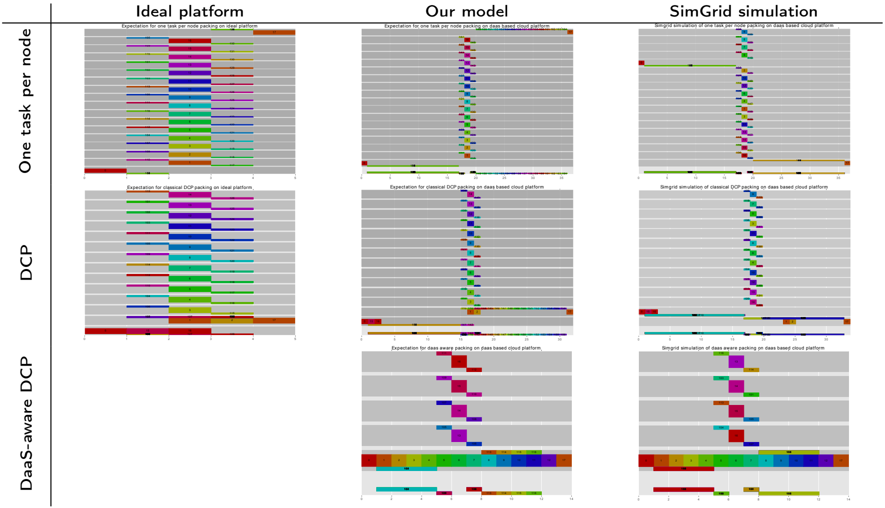


Results

Simgrid simulation of classical DCP packing on daas based cloud platform

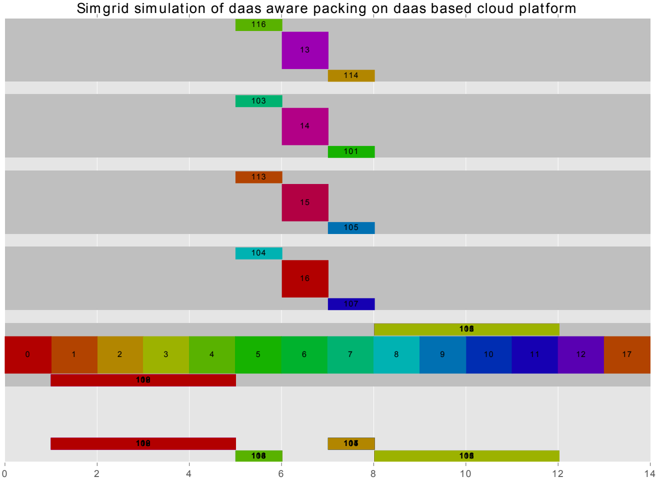


Results



Comparison of the different clustering policies (Gantt charts and their associated makespan) for a multiple data fork-join DAG ($n = 16$).

Results


Simgrid simulation of daas aware packing on daas based cloud platform
116
13
114
103
14
101
113
15
105
104
16
107
108
0
1
2
3
4
5
6
7
8
9
10
11
12
17
109
109
103
108
108
0
2
4
6
8
10
12
14

DAG	Algorithm	#Nodes	**Makespan** $\left(t\right)$	**Cost** $\left(\sum_{nodes}(t)\right)$
Single Data Fork-join	One task per node	18	22.024	67.204
	Single node	1	18.000	**18.000**
	DCP	14	18.024	56.168
	DaaS aware DCP	2	**13.012**	20.012
Multiple Data Fork-join	One task per node	18	37.024	82.204
	Single node	1	18.000	**18.000**
	DCP	14	33.803	70.156
	DaaS aware DCP	5	**14.000**	26.048

Cost and makespan details of the different clustering policies for single data or multiple data fork-join DAG ($n = 16$).

Conclusion

- Network infrastructures and communication pattern are key;
- Legacy algorithm is not necessarily bad and can be updated;
- Simulations of the resulting clustering gives better results;
- We keep the advantages of DCP: can be applied to any DAG.

Future work

Regarding task clustering and Clouds:

- Deciding which instance to use;

Broader:

- Designing the rest of our framework;
- Implementing everything in a user-friendly platform manager;
- Designing extra features: QoS description, instance selection, . . .

Thank you

Thank you for your attention.

Do you have any question ?