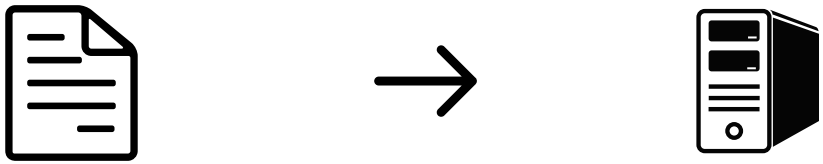
Toward an autonomic approach of workflows distribution on cloud

Hadrien Croubois

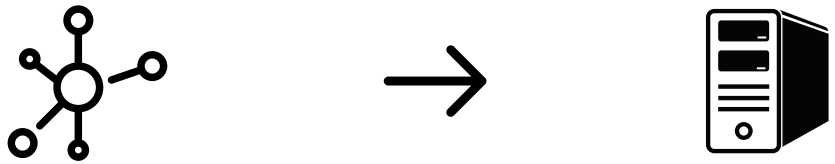
PhD Student at Avalon, Laboratoire de l'informatique du Parallélisme
École Normale Supérieure de Lyon, France
Supervised by Eddy Caron

Introduction

Scheduling: Matching jobs and resources

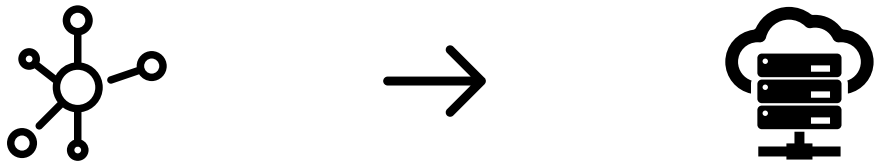


Scheduling: Matching jobs and resources



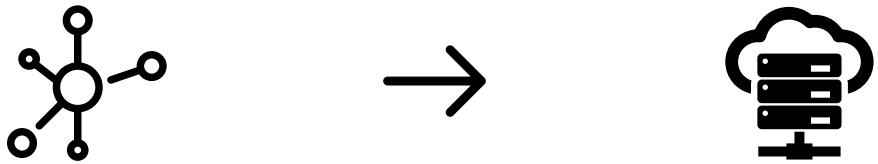
- Jobs definition is changing,

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);
- Multi-Tenant (collaborative);
- Dynamic platform (IaaS cloud) with DaaS storage.

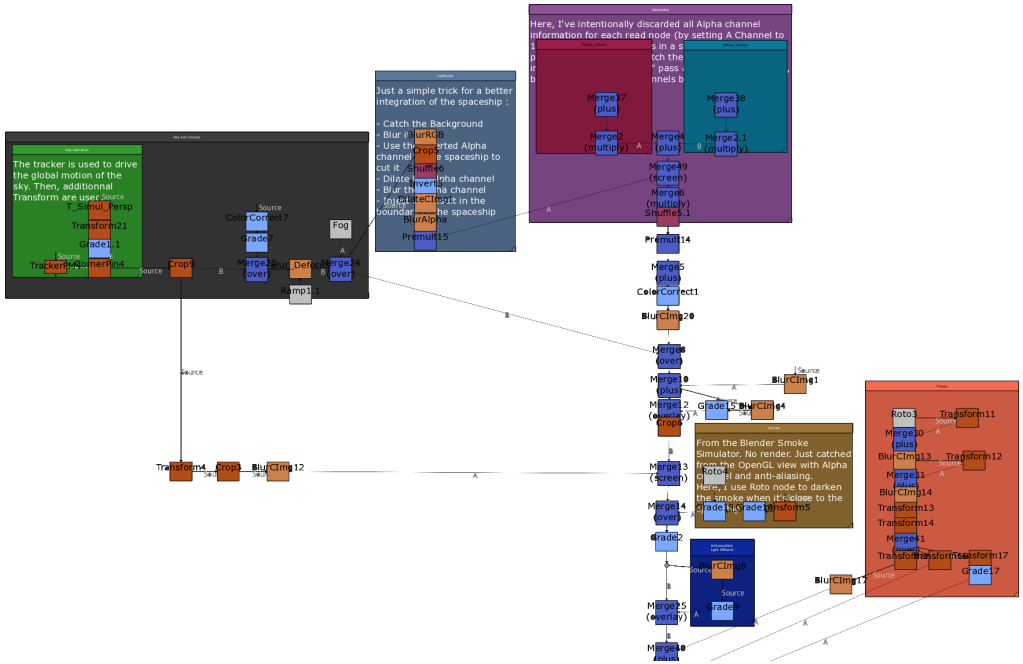


Figure 1: Rendering workflow in Natron

Challenge: Accounting for three factors

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

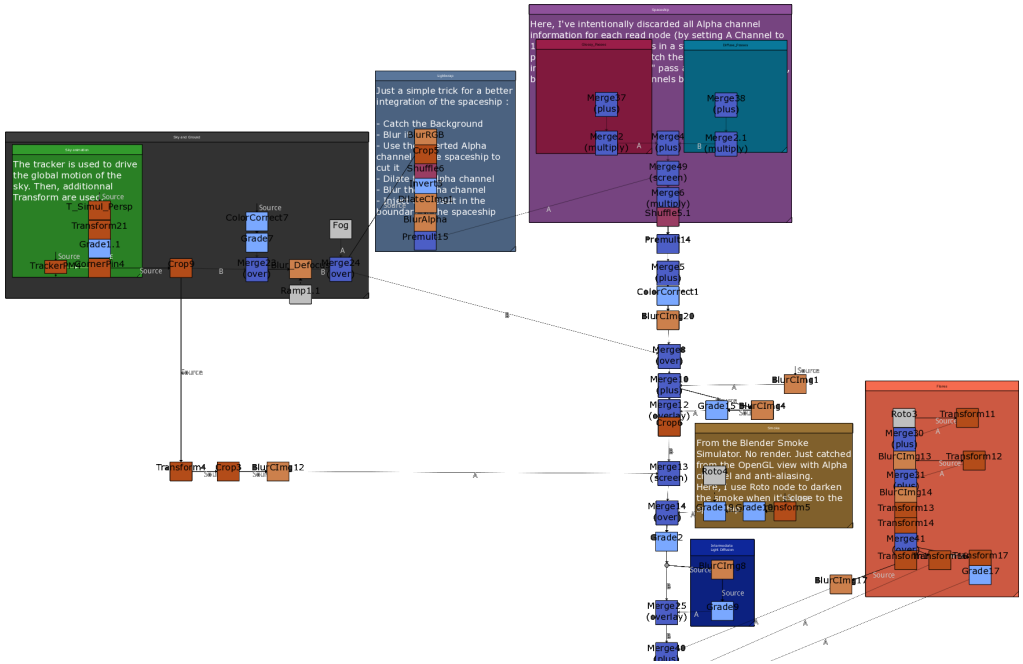


Figure 1: Rendering workflow in Natron

State of the art

Previous work considers at most 2 of those 3 factors.

Our approach

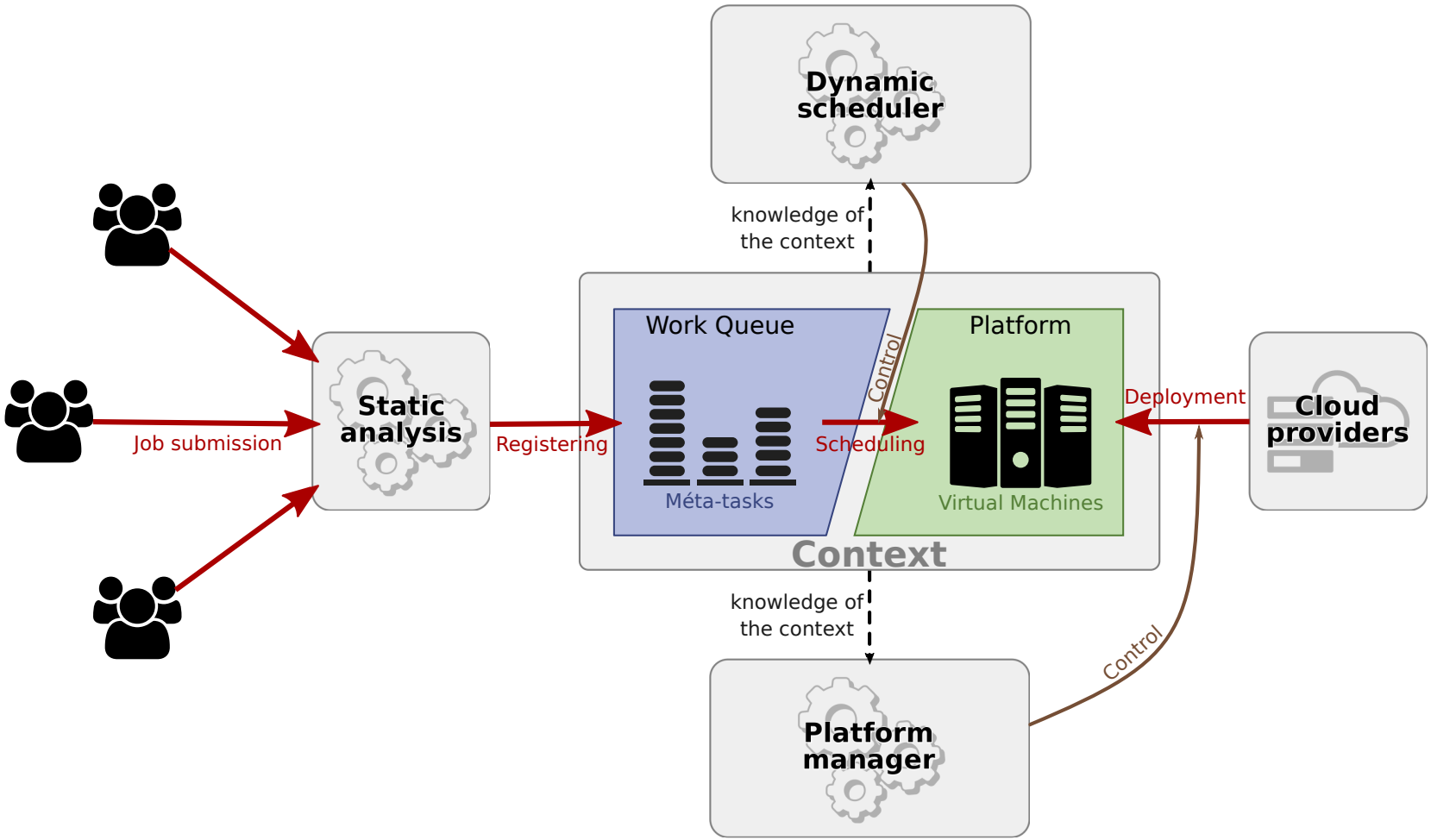
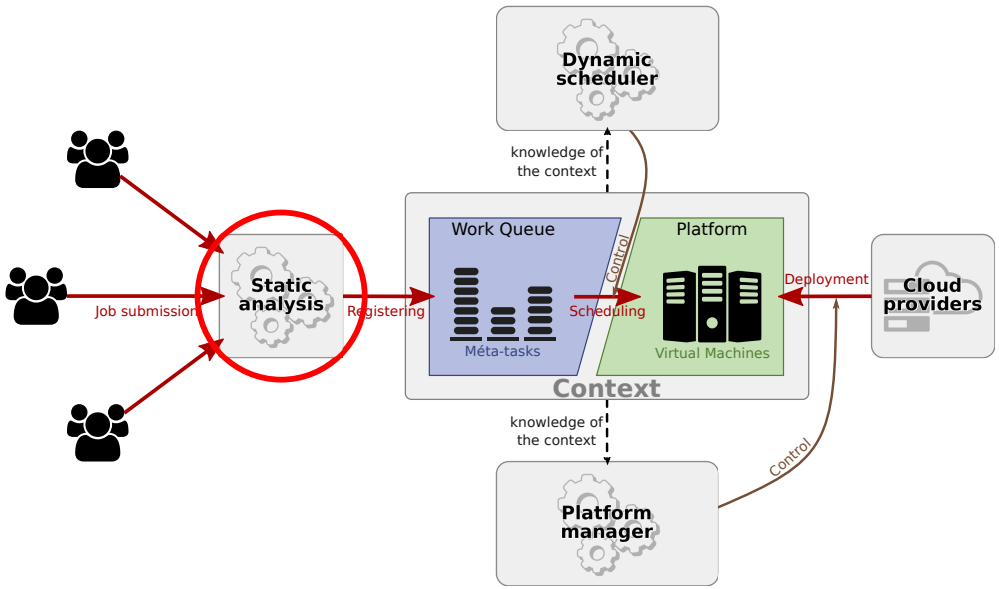


Figure 2: Framework architecture

Tackling problem 1



Algorithm 1 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 $\mathcal{C}$

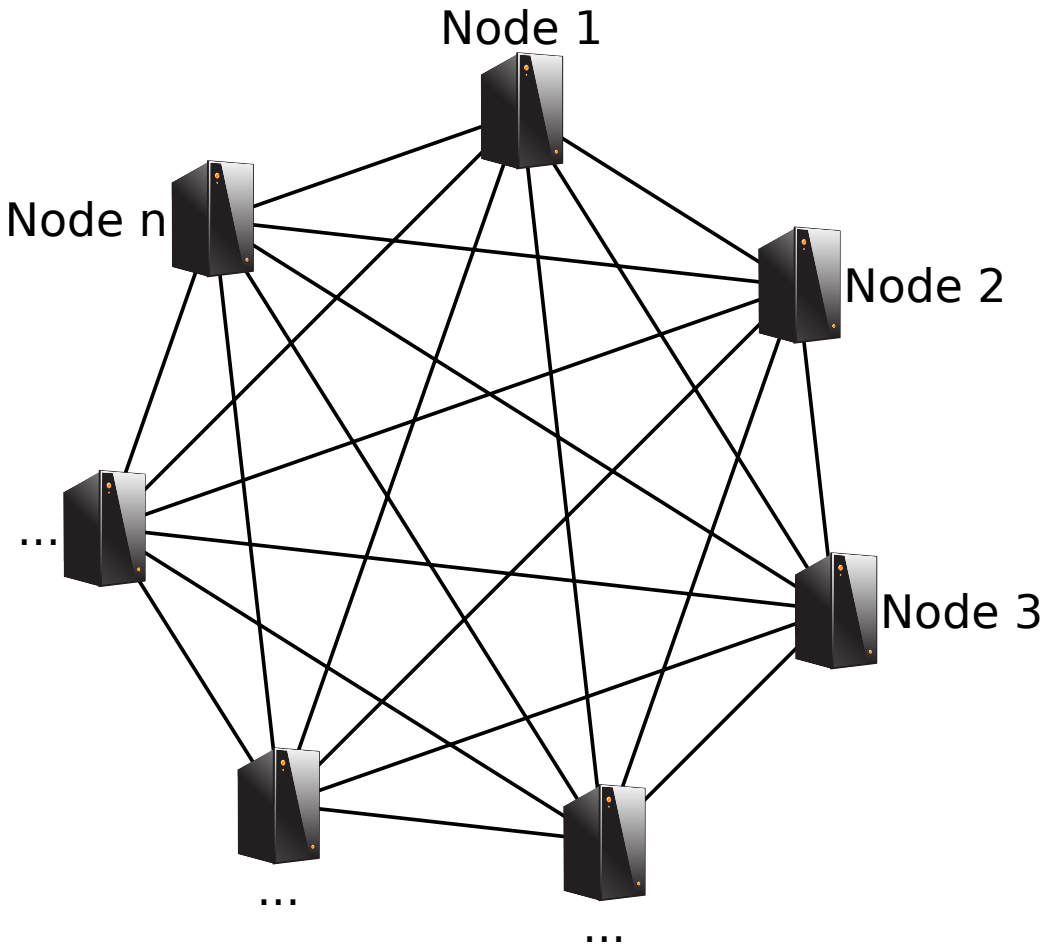
Y.-K. Kwok and I. Ahmad, "A static scheduling algorithm onto multiprocessors," in *Proceedings of the 1994 International Conference on Parallel Processing*.

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

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

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

Underlying network topology



Interference between concurrent communications

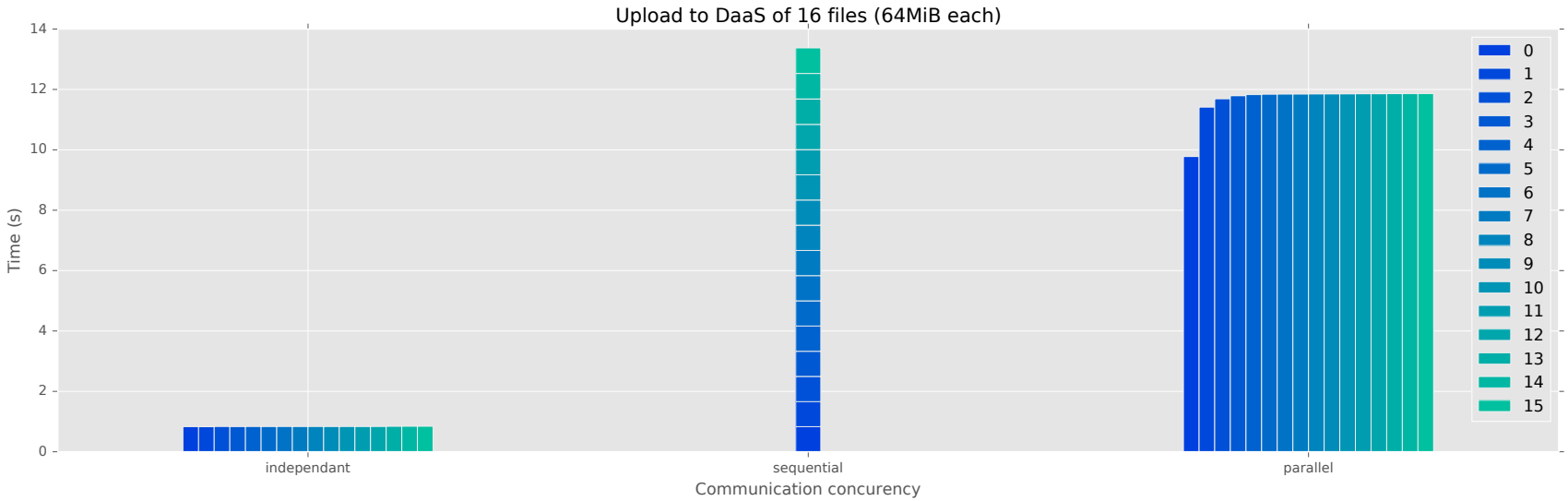
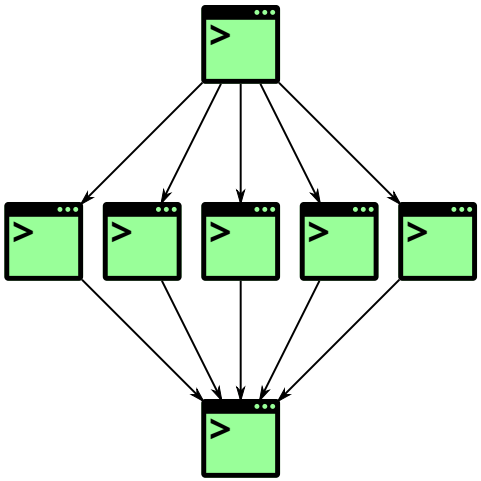
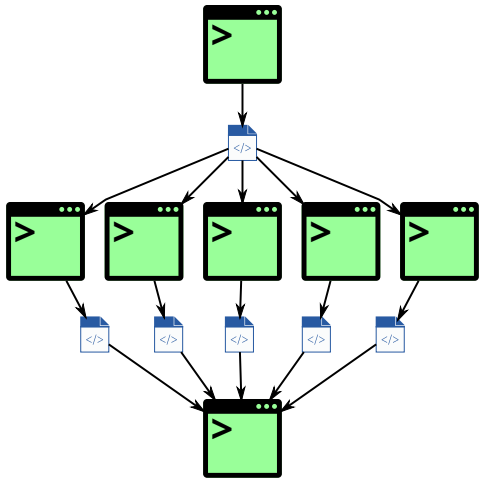


Figure 3: Transferring files between one node in sagittaire cluster (grid5000) and a DaaS (storage5K)

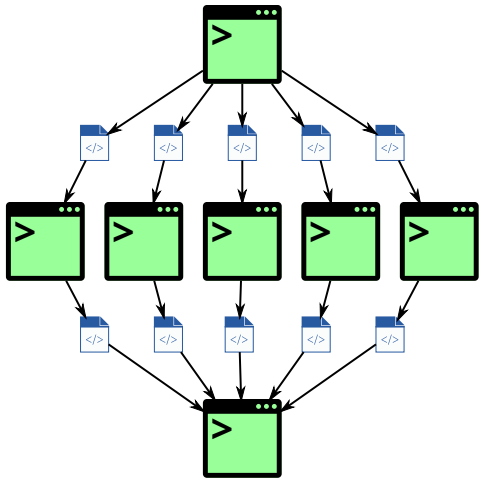
See dependencies from the Dataflow point of view



(a) Legacy representation



(b) Our representation (with single data upload)



(c) Our representation (with multiple data upload)

Figure 4: Representation of a fork-join DAG with n=5 independent jobs.

Reconsidering the network topology

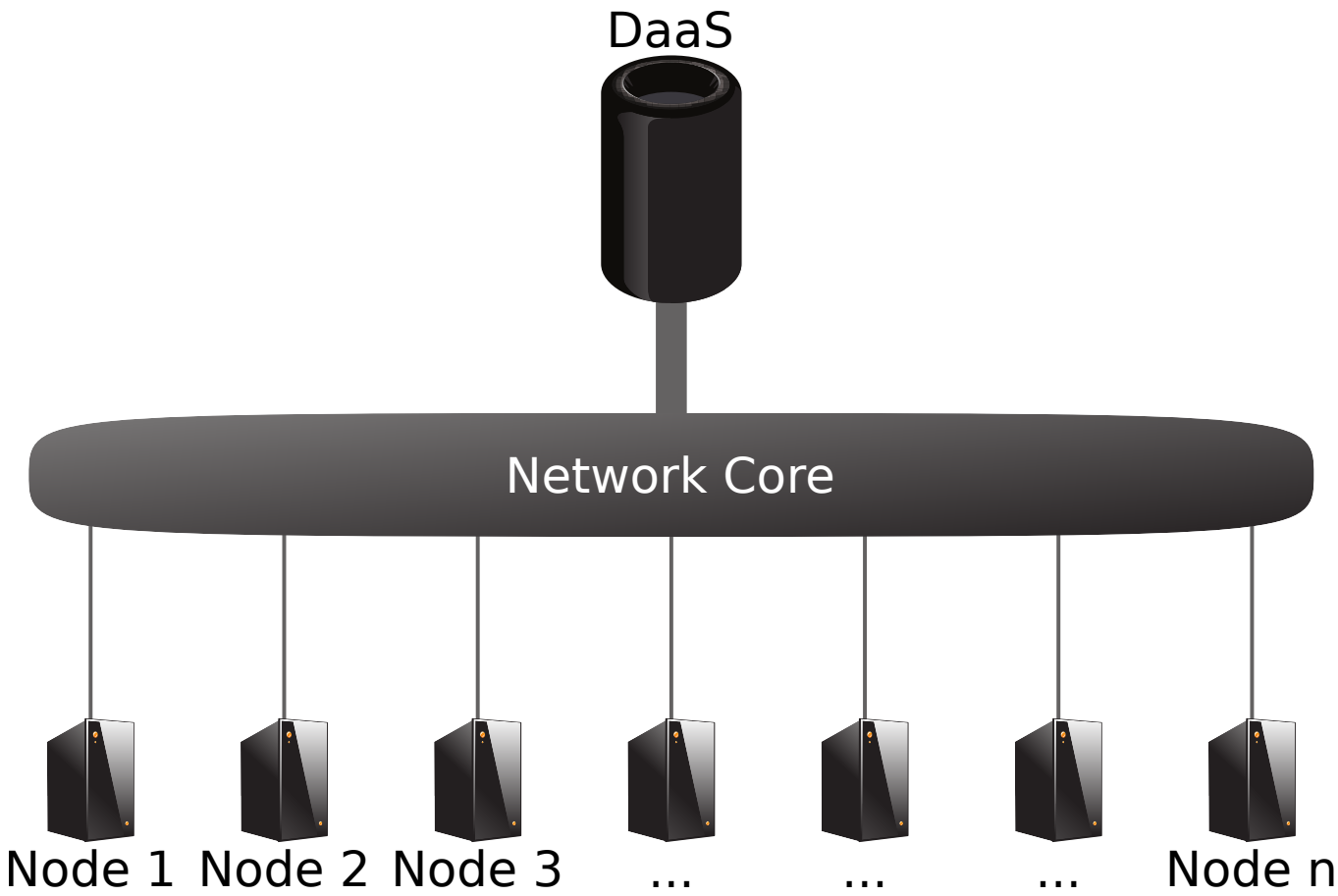


Figure 5: A generic model of DaaS-based network topology.

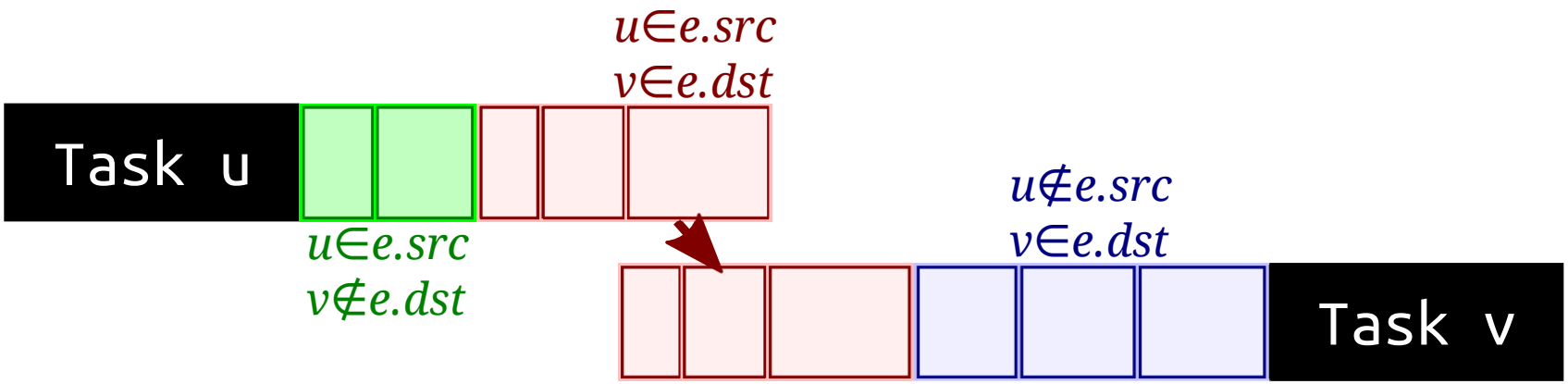


Figure 6: Preview of the communications between two tasks for a data-based workflow on a DaaS-based platform.

Worst case communications

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

Locality

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

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

Worst case communications with locality

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

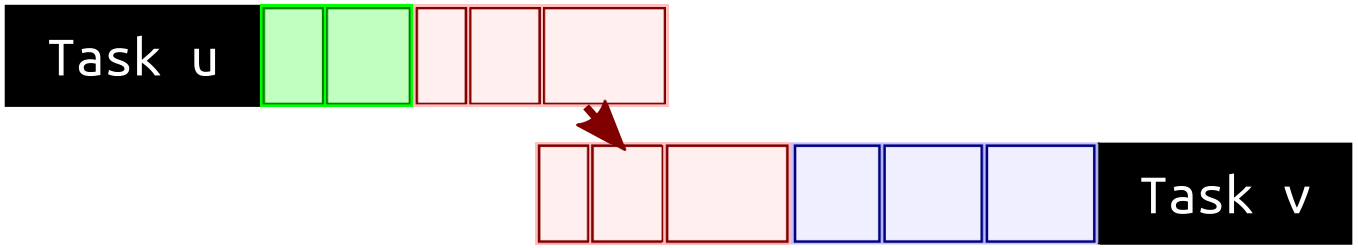
$$c_{loc}(u,v) = \sum_{\substack{d \in edges \\ u=d.src \\ v \notin d.dst \\ islocal(d)=0}} \frac{d.size}{network\_up}$$

$$+ \sum_{\substack{d \in edges \\ u=d.src \\ v \in d.dst}} \frac{d.size}{\min(network\_up, network\_down)} + \max_{\substack{d \in edges \\ u=d.src \\ v \in d.dst}} \frac{d.size}{\max(network\_up, network\_down)}$$

$$+ \sum_{\substack{d \in edges \\ u \neq d.src \\ v \in d.dst \\ islocal(d,v)=0}} \frac{d.size}{network\_down}$$

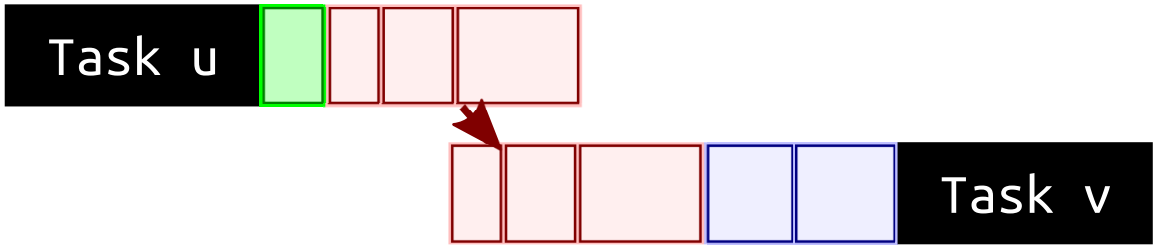
Worst case communications with locality

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



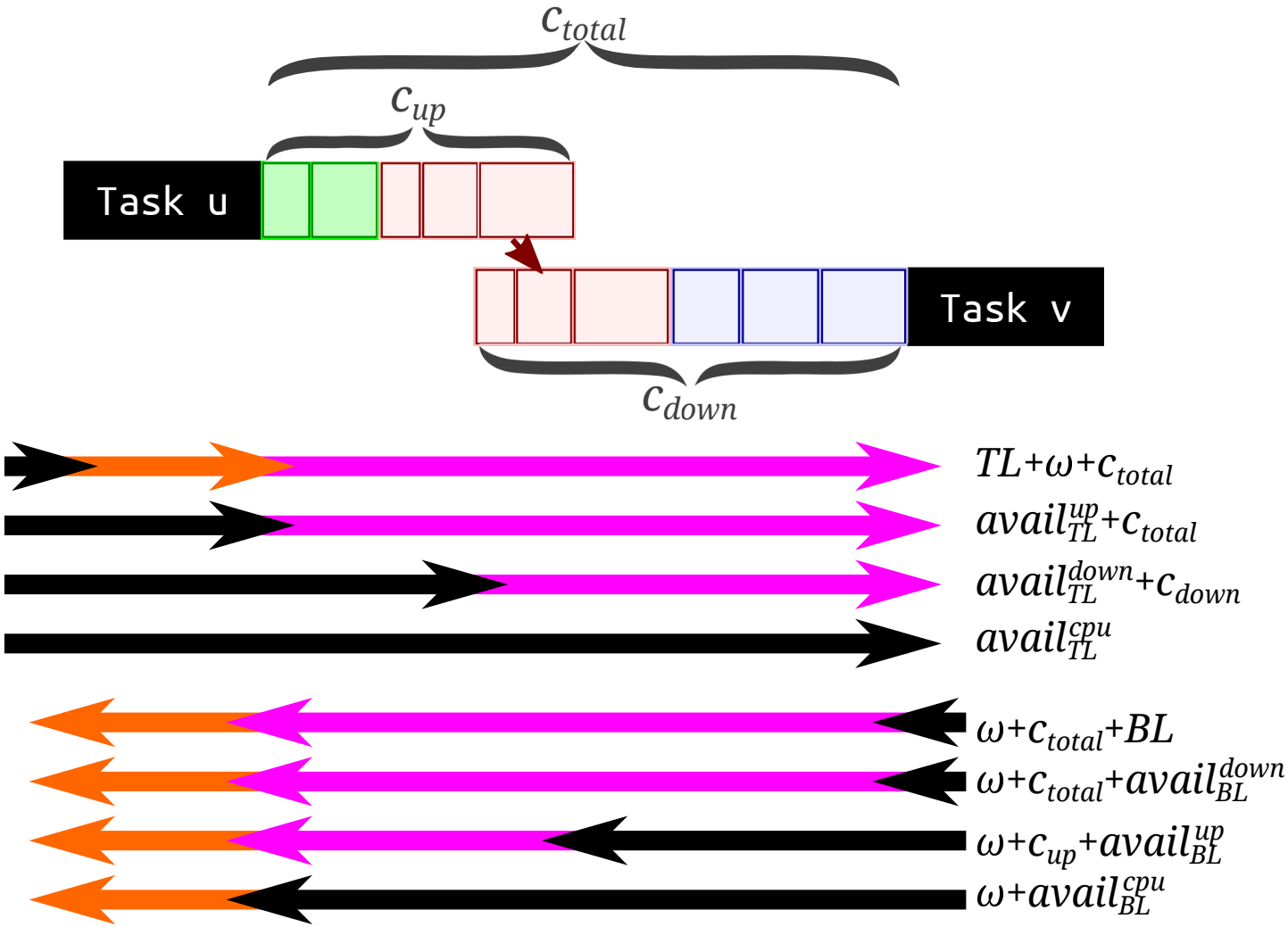


Figure 7: Preview of the critical path computation taking the machine network availability into account in DaaS-based platform.

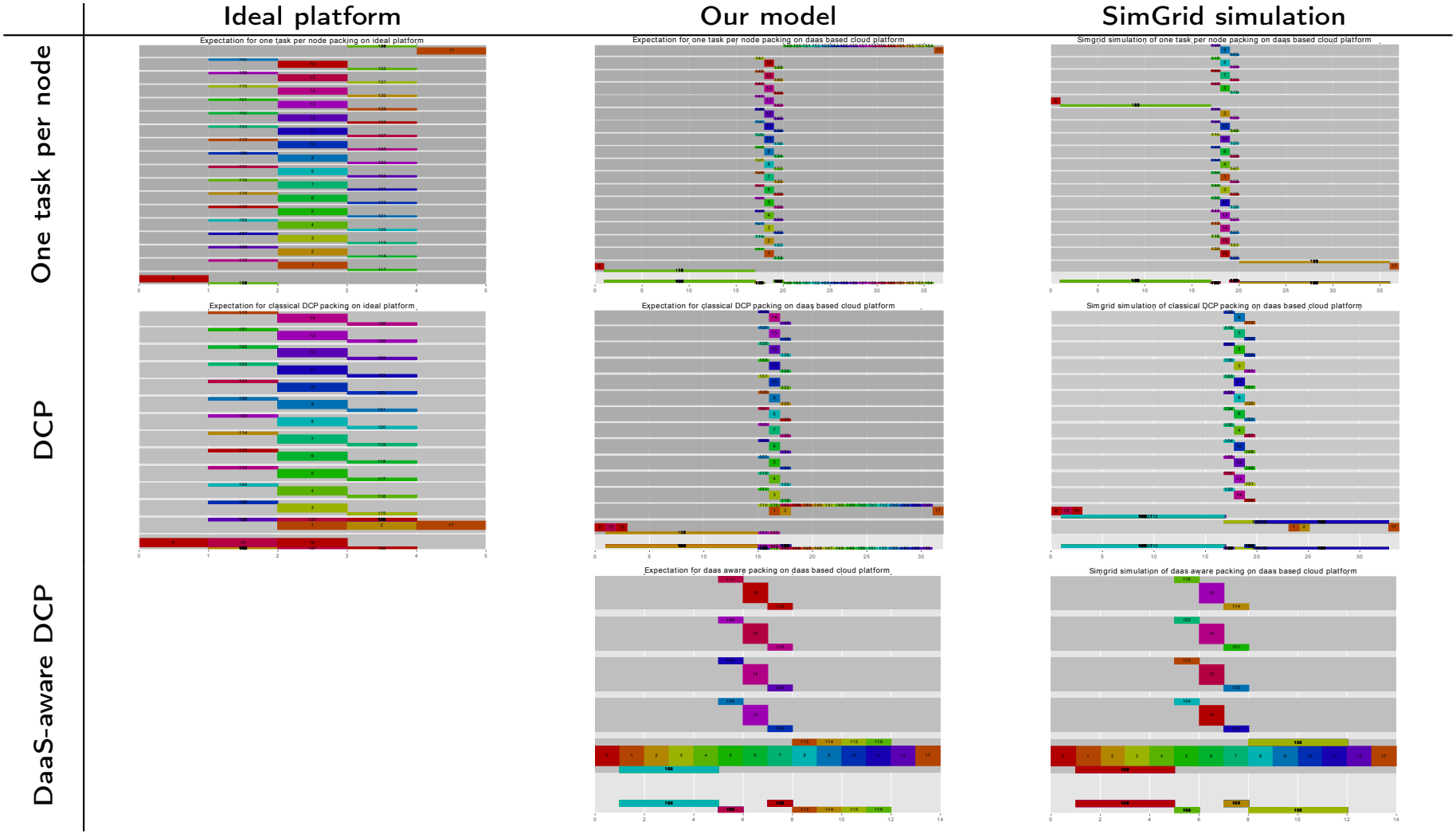


Figure 8: Comparison of the different clustering policies (Gantt charts and their associated makespan) for multi-data fork-join DAG ($n = 16$).

DAG	Algorithm	#Nodes	Makespan (t)	Cost (core×t)
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

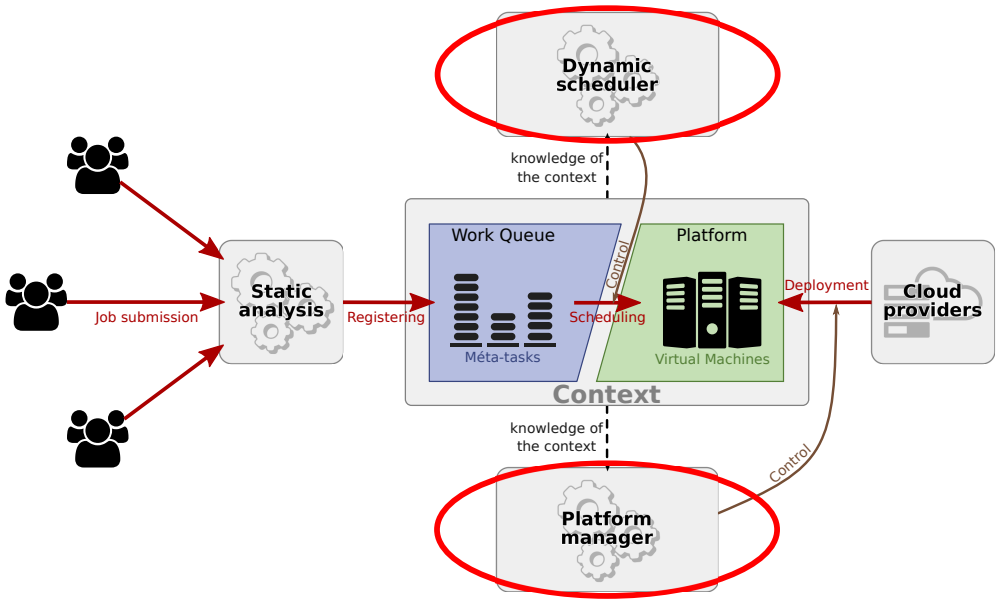
Figure 9: Cost and makespan details of the different clustering policies for single-data or multi-data fork-join DAG ($n = 16$).

Hadrien Croubois and Eddy Caron.
Communication-aware task placement for workflow scheduling on daas-based cloud.
In *Submitted to PDCO2017 (IPDPS workshop)*, 2017.

Instance selection

Cloud platforms are heterogeneous !
We need to select which type of instance to run on.

Tackling problem 2 & 3



Multiple challenges

- Have the computing power needed to meet deadlines;
- Have the lowest deployment cost possible;
- React to sudden increase/decrease of the workload.

Multiple challenges

Challenge

Build a fully automatic solution.

- React to sudden increase/decrease of the workload.

Strategy

- Ranking of ready tasks based on priority;
- Available nodes execute tasks based on their priority (+ other parameter ?);
- Available nodes try to avoid wasting money by committing suicide when no work is available and remaining alive would cost money (end of hour);
- Nodes are deployed when current pool of workers is insufficient.

Strategy

- Ranking of ready tasks based on priority;
- Available nodes execute tasks based on their priority (+ other parameter ?);

Challenge

Determine when to deploy new nodes : an autonomic loop !
cost money (end of hour);

- Nodes are deployed when current pool of workers is insufficient.

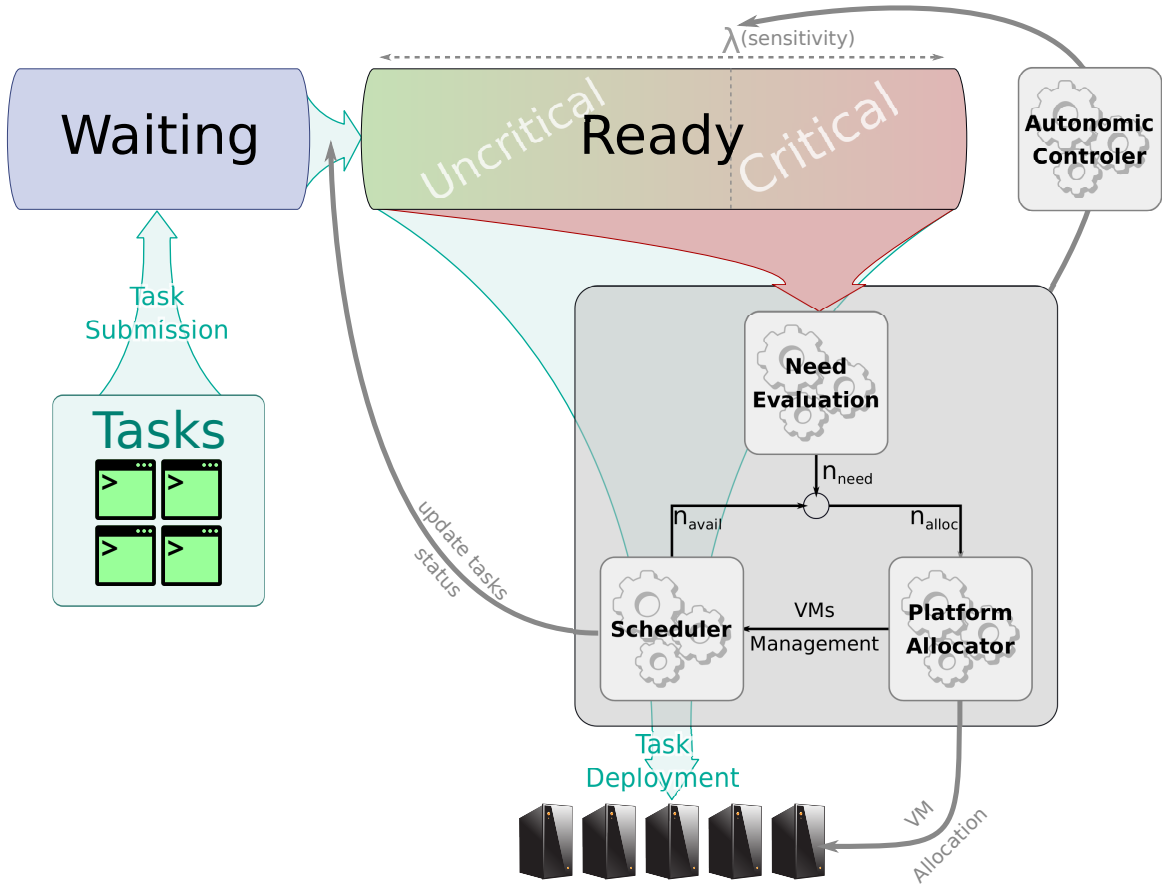


Figure 10: Preview of the scheduler and platform manager structure (threshold with autonomic controler).

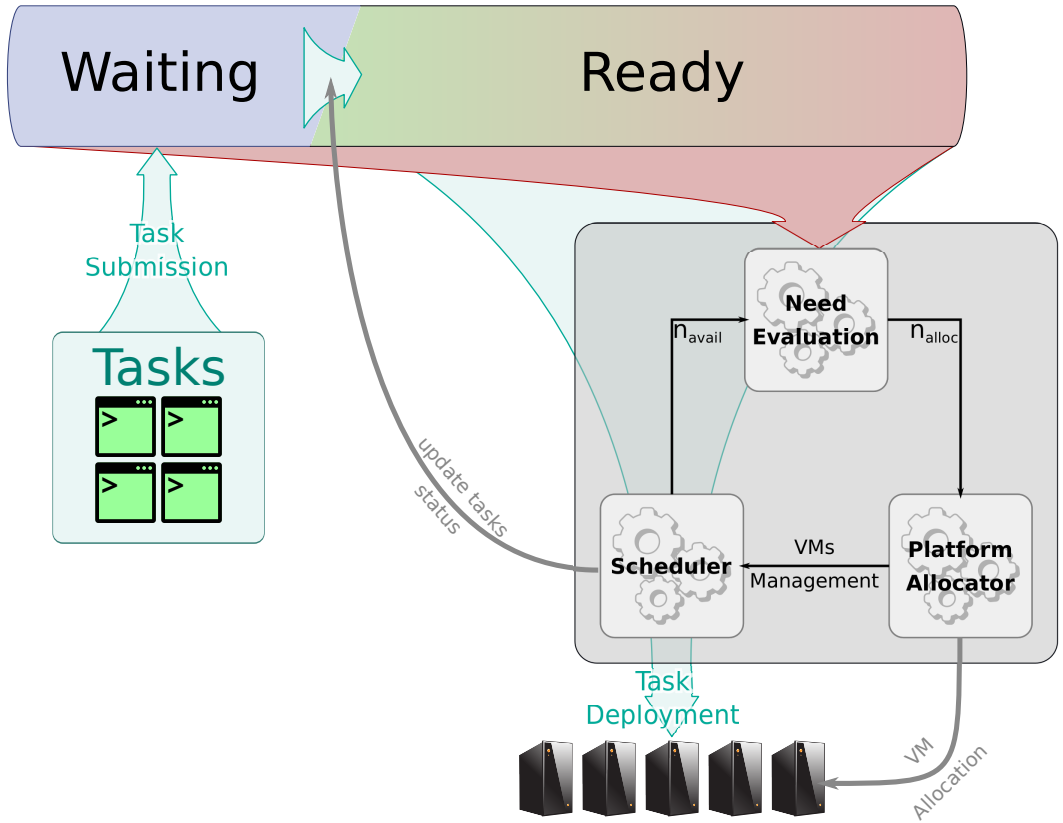
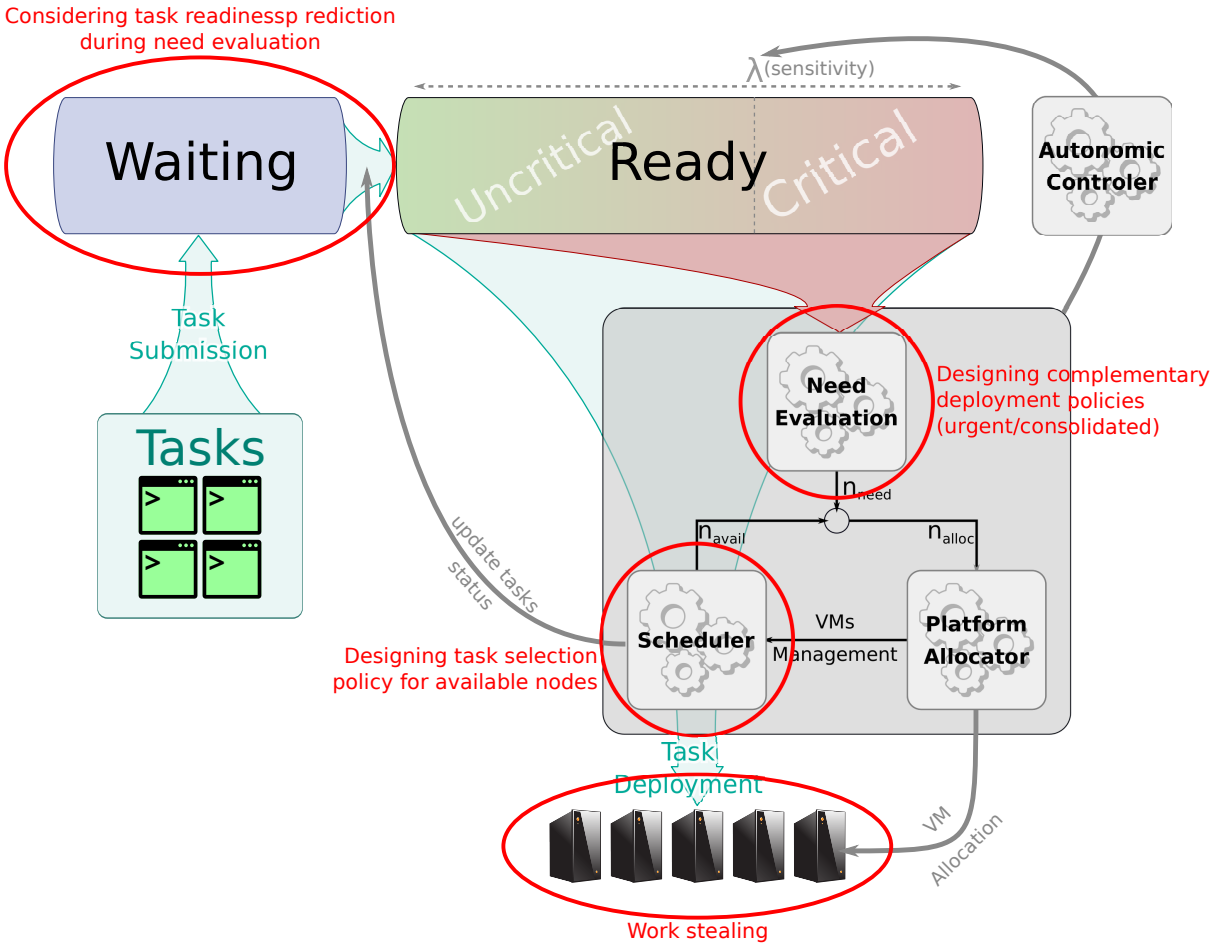


Figure 11: Preview of the scheduler and platform manager structure (simulation).

Current & Future work



Conclusion

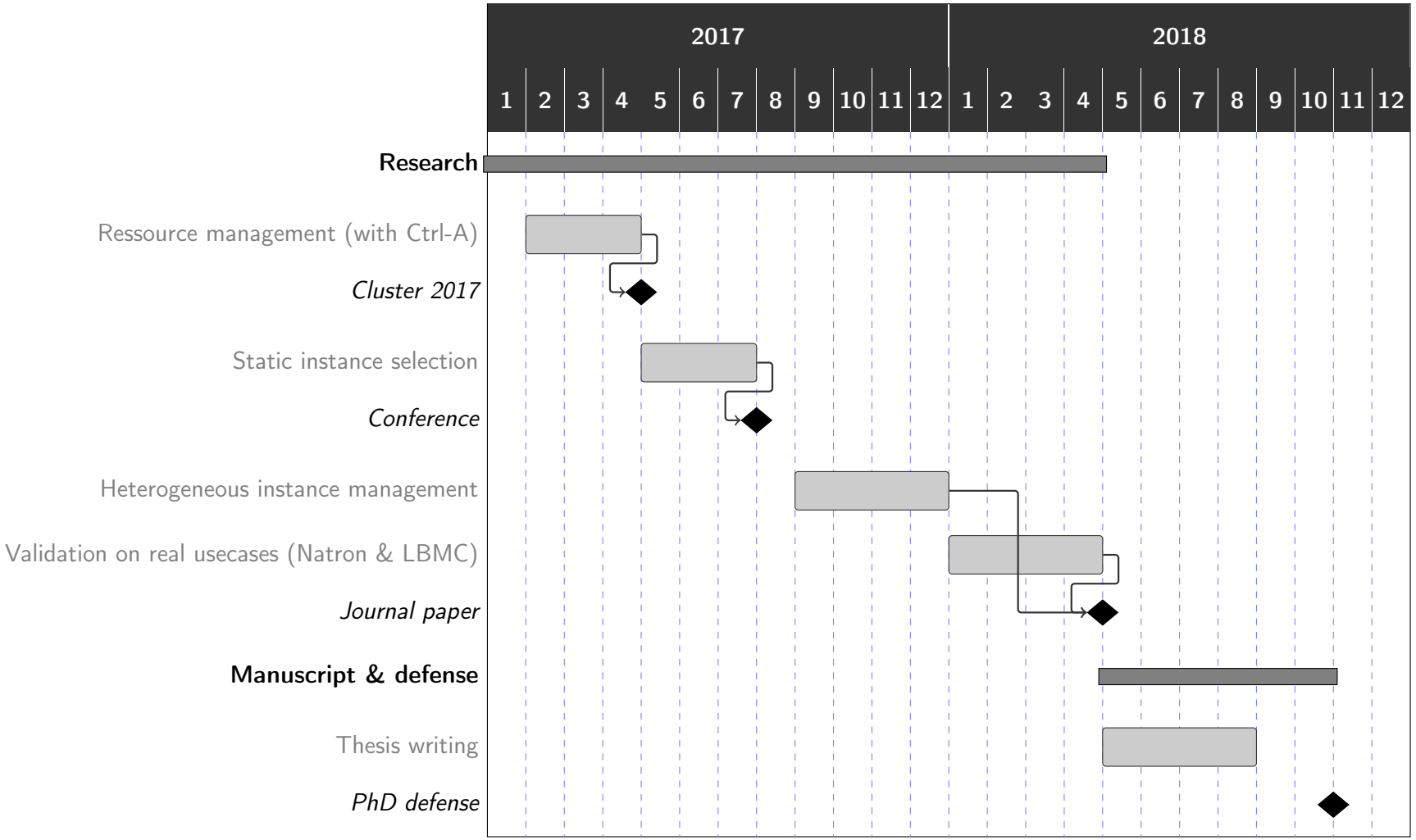
Current status

- We have identified the problem;
- We have a framework for building a solution;
- We have submitted a paper (PDCO) dealing with the clustering;
- We have started a collaboration with Ctrl-A for building an autonomic loop.

What is next ?

- Improving the static analysis step (instance selection),
- Autonomic ressource management,
  - → Co-design of the scheduling component with Aurélie (Roma)
- Implementing and validation the model.

Roadmap


2017
2018
1 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 9 10 11 12
Research
Ressource management (with Ctrl-A)
Cluster 2017
Static instance selection
Conference
Heterogeneous instance management
Validation on real usecases (Natron & LBMC)
Journal paper
Manuscript & defense
Thesis writing
PhD defense

Thank you for your attention.