Minimizing rental cost for multiple recipe applications in the Cloud

F. Hana, L. Marchal, J.-M. Nicod, L. Philippe, V. Rehn-Sonigo and H. Sabbah

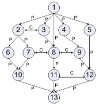
LIP-ENS Lyon – FEMTO-ST institute - UFC/ENSMM Besançon

Aussois - March 29th, 2016

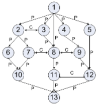
Outline

1. Introduction and motivation
2. Algorithmic solutions
3. Heuristics for the general case
4. Experiments
5. Conclusion

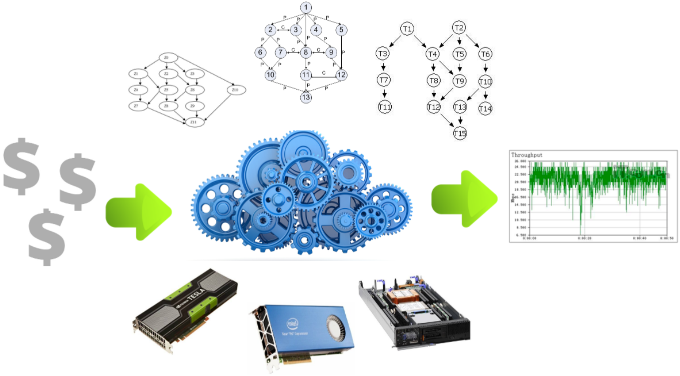
Introduction and motivation



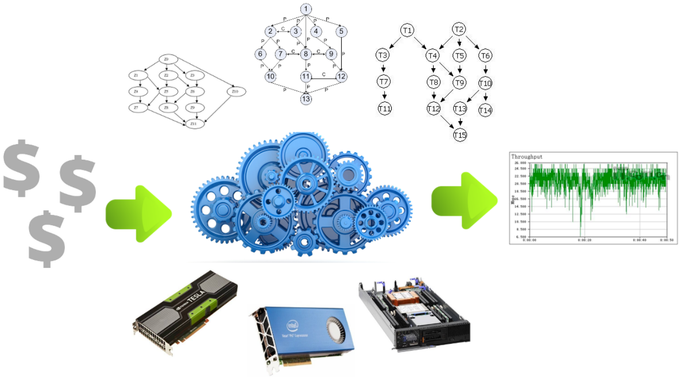
Introduction and motivation



Introduction and motivation



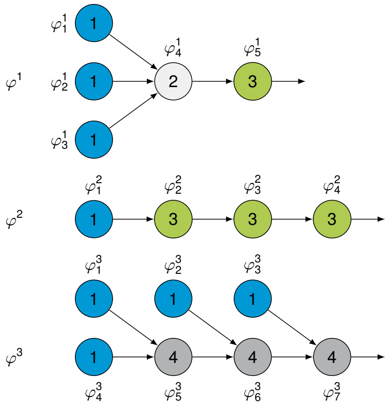
Introduction and motivation



Overall objective: To provision just enough resources to reach the target throughput for a given DAG based streaming application

Application framework

Each workflow application φ^j produces the same result Φ.



- Each task φ^j_i has a task type q
- Target throughput ρ
- Each application can be run at a different throughput ρ_j
- $\rho = \sum_j \rho_j$

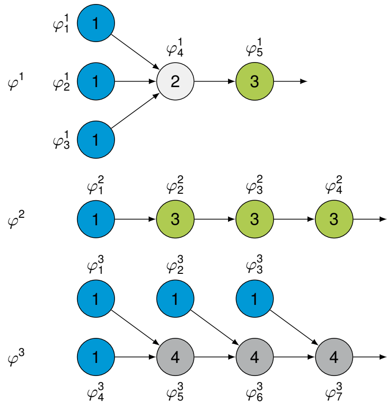
Target platform

One processor type per task type

- c_q: Rental cost for type q
- r_q: Throughput of type q

Application framework

Each workflow application φ^j produces the same result Φ.



- Each task φ^j_i has a task type q
- Target throughput ρ
- Each application can be run at a different throughput ρ_j
- $\rho = \sum_j \rho_j$

Target platform

One processor type per task type

- c_q: Rental cost for type q
- r_q: Throughput of type q

find the cheapest configuration to reach the target throughput

Problem definition

MinCOST: Minimize the global rental cost C

- an application described by J graphs
- a platform described by processor cost c_q and throughput r_q
- a given QoS ρ as a global output throughput

$\Rightarrow$ select which graphs φ^j are used

$\Rightarrow$ chose with which output throughput ρ_j ($\rho_j = 0$ if unused)

$\Rightarrow$ deduce x_q processors of each type

$$MinCOST(\rho) = \min_j\left(\sum_j x_j \cdot c_q\right)$$

$$\text{where} \quad \rho = \sum_j \rho_j \ (1 \leq j \leq J)$$

Problem resolution

Simple case

- the application is described by only one graph

General case

$$\rho = \sum_j \rho_j$$

- Black box application
- Application graphs without shared task types
- Application graphs with shared task types

Simple case: Single application graph

- one application described by one single graph φ^1
- $\forall q$, x_q we can be easily computed:

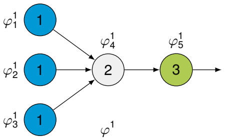
$$x_q = \left\lceil \frac{n_q}{r_q} \cdot \rho \right\rceil$$

- the associated cost C_q

$$C_q(\rho) = \left\lceil \frac{n_q}{r_q} \cdot \rho \right\rceil \times c_q$$

- the final cost C:

$$C(\rho) = \sum_{q=1}^{Q} C_q(\rho) = \sum_{q=1}^{Q} \left\lceil \frac{n_q}{r_q} \cdot \rho \right\rceil \times c_q$$



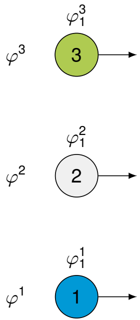
General case: Black box applications

- each graph $\varphi^j = \varphi^j_1$ is one complex task

$$\forall j \text{ and } \forall j' (1 \leq j, j' \leq J): \; t(1,j) = t(1,j') \Rightarrow j = j'$$

- let ρ_q the output of $\varphi^j = \varphi^q$
- x_q can be found by solving the following linear program:

$$\begin{cases} \text{Minimize } C(\rho) = \sum_{q=1}^{Q} x_q c_q \\ \text{Under the constraint } \sum_{q=1}^{Q} x_q \rho_q \geqslant \rho \end{cases}$$

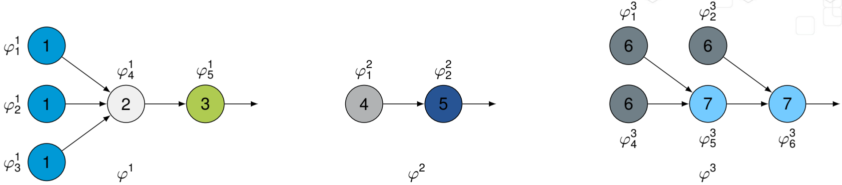


$\Rightarrow$ this resembles a knapsack problem with repetition and negative weights and values

- the knapsack problem is a (unary) NP-Complete problem

$\Rightarrow$ it exists a pseudo-polynomial dynamic program (time complexity $O(J\rho)$)

Application graphs without shared task types



- application Φ can be described by $\varphi^1, \ldots, \varphi^j, \ldots, \varphi^J$ with the same output result
- each task φ_i^j from one graph φ^j has a different type from every other task of an other graph $\varphi^{j'}$

  $$t(i,j) \neq t(i',j') \text{ with } 1 \leq j, j' \leq J \text{ and } j \neq j' \text{ and } 1 \leq i \leq I_j \text{ and } 1 \leq i' \leq I_{j'}$$

- this problem is at least (unary) NP-Complete

$\Rightarrow$ it exists also a pseudo-polynomial dynamic program to solve it

Applications without shared task types

a dynamic program to solve this problem

- let $C(\rho, j$ be the optimal platform cost to reach ρ using the first j application graphs

$$C(\rho, j) = \begin{cases} \sum_{i=1}^{I_1} \left\lceil \frac{n^1_{t(i,1)}}{r_{t(i,1)}} \cdot \rho \right\rceil \times c_{t(1,k)} \text{ if } j = 1 \\ \min_{0 \leq \rho_j \leq \rho} \Bigg(C(\rho - \rho_j, j-1) + \\ \qquad \sum_{i=1}^{I_j} \left\lceil \frac{n^j_{t(i,j)}}{r_{(t(i,j))}} \cdot \rho_j \right\rceil \times c_{t(i,j)} \Bigg) \\ \text{otherwise} \end{cases}$$

$\Rightarrow$ the solution is given by $C(\rho, J)$

Application graphs without shared task types

complexity analysis

- as $\forall q$, $r_q \in \mathbb{N}$, $\forall j$ $\rho_j \in \mathbb{N}$

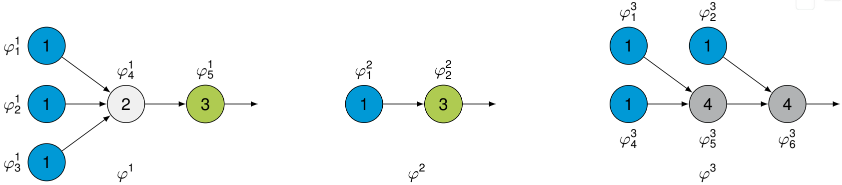
$\rightarrow$ it exists a finite number of ρ_j to test in the previous formulation

- to compute $C(\rho, j)$, all $C(\rho', j')$ with $\rho' \leq \rho$ and $j' \leq j$ has to be computed

$\Rightarrow$ the complexity of the elementary computation is $O(\rho I)$

$\Rightarrow$ the complexity of computing $C(\rho)$ is $O(\rho^2 IJ)$

Application graphs with shared task types



- one application is described by several graphs which share task type

$$\exists\, \varphi^j, \varphi^{j'} (1 \leqslant j, j' \leqslant J, j \neq j'), \exists\, i\, (1 \leqslant i \leqslant I_j),$$
$$\exists\, i' (1 \leqslant i' \leqslant I_{j'}) \text{ s.t. } t(i,j) = t(i',j')$$

$\Rightarrow$ a processor may be shared between several application graphs

Application graphs with shared task types

ILP formulation

Minimizing $C(\rho) = \sum_{q=1}^{Q} x_q \cdot c_q$
under constraints

- ρ has to be at least the sum of ρ_j

$$\sum_{j=1}^{J} \rho_j \geqslant \rho$$

- for each type q we have to provision enough resources (x_q)

$$\forall q \; x_q \cdot r_q \geqslant \sum_{j=1}^{J} \left(\sum_{i=1 \,|\, t(i,j)=q}^{I_j} \rho_j \right),$$

with $q = t(i,j)$ and $x_q \in \mathbb{N}$

Application graphs with shared task types

ILP formulation

Minimizing $C(\rho) = \sum_{q=1}^{Q} x_q \cdot c_q$
under constraints

- ρ has to be at least the sum of ρ_j

$$\sum_{j=1}^{J} \rho_j \geqslant \rho$$

- for each type q we have to provision enough resources (x_q)

$$\forall q \; x_q \cdot r_q \geqslant \sum_{j=1}^{J} \left(\sum_{i=1 \,|\, t(i,j)=q}^{I_j} \rho_j \right),$$

with $q = t(i,j)$ and $x_q \in \mathbb{N}$

the complexity of this case is still open
unary or binary NP-Complete

Application graphs with shared task types

Heuristic approaches

$$\rho = (\rho_1, \rho_2, \ldots, \rho_J)$$

1. H0: random
2. H1: best graph
3. H2: random walk
4. H31: stochastic descent
5. H32/H32Jump: steepest gradient

Application graphs with shared task types

Heuristic approaches

$$\rho = (\rho_1, \rho_2, \ldots, \rho_J)$$

1. H0: random $\quad \rho = (\rho_1, \rho_2, \ldots, \rho_J)$
2. H1: best graph
3. H2: random walk
4. H31: stochastic descent
5. H32/H32Jump: steepest gradient

Application graphs with shared task types

Heuristic approaches

$$\rho = (\rho_1, \rho_2, \ldots, \rho_J)$$

1. H0: random
2. H1: best graph $\qquad \rho = (0, \ldots, \rho, \ldots, 0)$
3. H2: random walk
4. H31: stochastic descent
5. H32/H32Jump: steepest gradient

Application graphs with shared task types

Heuristic approaches

$$\rho = (\rho_1, \rho_2, \ldots, \rho_J)$$

1. H0: random
2. H1: best graph
3. H2: random walk
   - φ_{j1} and φ_{j2} are randomly chosen

$$(\ldots, \rho_{j1}, \ldots, \rho_{j2}, \ldots) \rightarrow (\ldots, \rho_{j1} - \delta, \ldots, \rho_{j2} + \delta, \ldots)$$

$$(\ldots, \rho_{j1}, \ldots, \rho_{j2}, \ldots) \rightarrow (\ldots, 0, \ldots, \rho_{j2} + \rho_{j1} \ldots) \quad \text{if } \rho_{J1} < \delta$$

4. H31: stochastic descent
5. H32/H32Jump: steepest gradient

Application graphs with shared task types

Heuristic approaches

$$\rho = (\rho_1, \rho_2, \ldots, \rho_J)$$

1. H0: random
2. H1: best graph
3. H2: random walk
4. H31: stochastic descent
   - same as H2 except that we keep the same starting configuration as long as we do not obtain any improvement (local minimum)
5. H32/H32Jump: steepest gradient

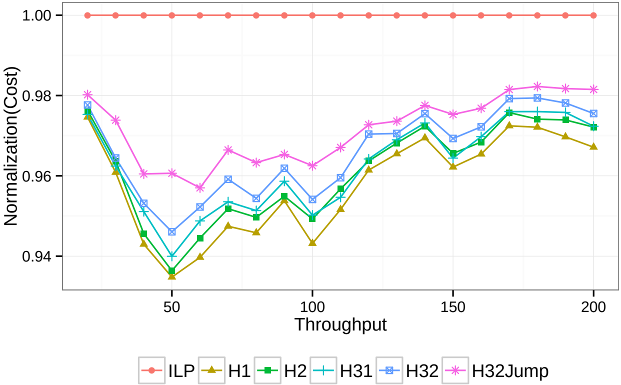
Application graphs with shared task types

Heuristic approaches

$$\rho = (\rho_1, \rho_2, \ldots, \rho_J)$$

1. H0: random
2. H1: best graph
3. H2: random walk
4. H31: stochastic descent
5. H32/H32Jump: steepest gradient
    - H32 same as H2 except we test every exchange $(+/-\delta)$ and keep the best until no improvement is possible (local minimum)
    - H32Jump same as H32 except we allows to explore solution that increases $C(\rho)$ to come out of local minima

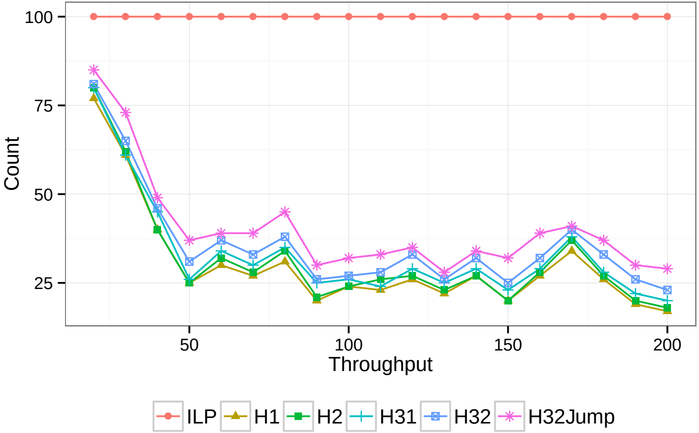
Experiments: small application graphs



ILP: Gurobi
simulator: Python

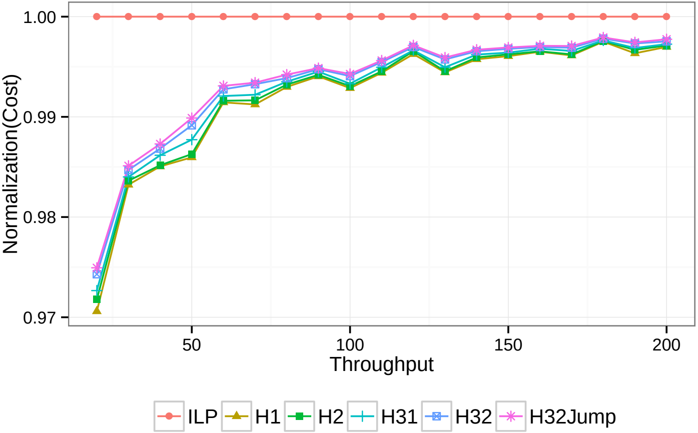
Normalization of cost with the optimal solution
20 alternative graphs, between 5 and 8 tasks for each graph

Experiments: small application graphs



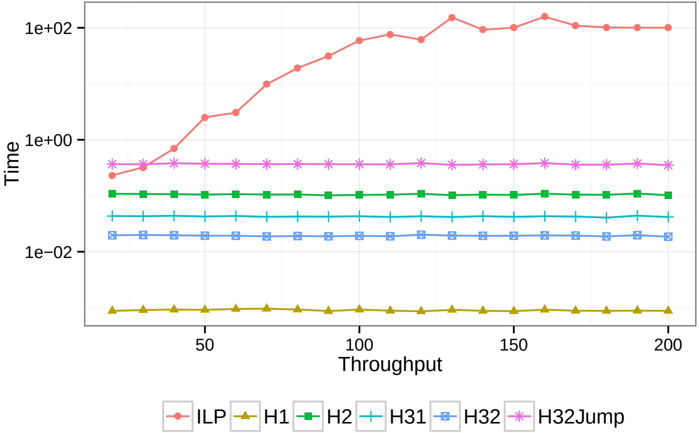
Number of times where each algorithm finds the best
20 alternative graphs, between 5 and 8 tasks for each graph

Experiments: large application graphs



Normalization of cost with the optimal solution
20 alternative graphs, between 50 and 100 tasks for each graph

Experiments: large application graphs



Computation time for the heuristics
20 alternative graphs, between 100 and 200 tasks for each graph

Conclusion

Find the cheapest configuration to reach the target throughput for a given DAG based streaming application

- the issue was to find a suitable distribution between DAGs

$\Rightarrow$ we deduce the platform to rent on the Cloud (minimize the rental cost)

Conclusion

Find the cheapest configuration to reach the target throughput
for a given DAG based streaming application

- the issue was to find a suitable distribution between DAGs

$\Rightarrow$ we deduce the platform to rent on the Cloud (minimize the rental cost)

- in some cases we exhibit algorithms to solve optimally the problem (even if NP-Complete in the weak sens)
- the complexity of the most general case remains open

$\Rightarrow$ an ILP gives a characterization of an optimal solution

Conclusion

Find the cheapest configuration to reach the target throughput
for a given DAG based streaming application

- the issue was to find a suitable distribution between DAGs

$\Rightarrow$ we deduce the platform to rent on the Cloud (minimize the rental cost)

- in some cases we exhibit algorithms to solve optimally the problem (even if NP-Complete in the weak sens)
- the complexity of the most general case remains open

$\Rightarrow$ an ILP gives a characterization of an optimal solution

- Heuristics with good performance (6% from the optimal and asymptotically optimal)

Perspectives

economical cost $\Longleftrightarrow$ energy cost

Green computing

- how to take energy into account when we rent resources on Cloud ?
- how to associate both economical and energergetical criteria

Thanks for your attention