Firmament – overview and discussion

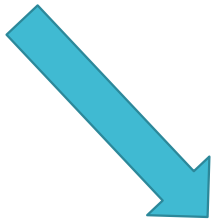
Łukasz Dudziak
Based on: “Firmament: fast, centralized cluster scheduling at scale” by Ionel Gog et al.

Why scheduling is important?

- Intelligent scheduling may significantly speedup execution time
  - Heterogenous systems
  - Multiple levels of cache (what tasks are better to be run on the same machine?)
- Fair scheduling

Task-by-task placement

- Fast, good for decentralized schedulers
- Does not have "global" knowledge
- Tasks might spend significant time in a queue (unfair?)



Batching placement:

- Always considers entire workload
- New task == rescheduling
- Has ability to pre-empt running tasks
- Slow

Why scheduling is important?

- Intelligent scheduling may significantly speedup execution time
  - Heterogenous systems
  - Multiple levels of cache (what tasks are better to be run on the same machine?)
- Fair scheduling

Task-by-task placement

- Fast, good for decentralized schedulers
- Does not have "global" knowledge
- Tasks might spend significant time in a queue (unfair?)

Let's optimize it!

Batching placement:

- Always considers entire workload
- New task == rescheduling
- Has ability to pre-empt running tasks

Slow

Min-cost max-flow

- Send certain amount of flow through a network, such as its overall cost is minimized.
  Formally:

$$Minimize \sum_{(i,j)\in E} cost_{(i,j)} flow_{(i,j)}$$

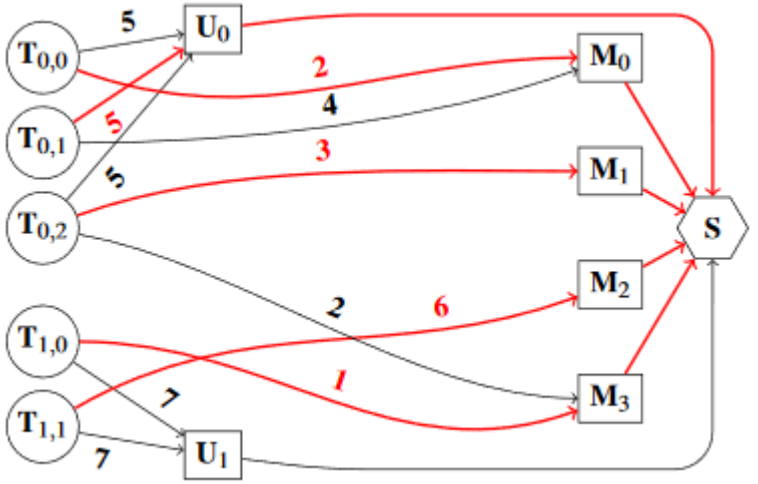
subject to:

$$\forall_{v\in V}\left(\sum_{(v,i)\in E} flow_{(v,i)} - \sum_{(j,v)\in E} flow_{(j,v)} = supply_v\right)$$

$$\forall_{(i,j)\in E}\left(0 \le flow_{(i,j)} \le capacity_{(i,j)}\right)$$

Quincy

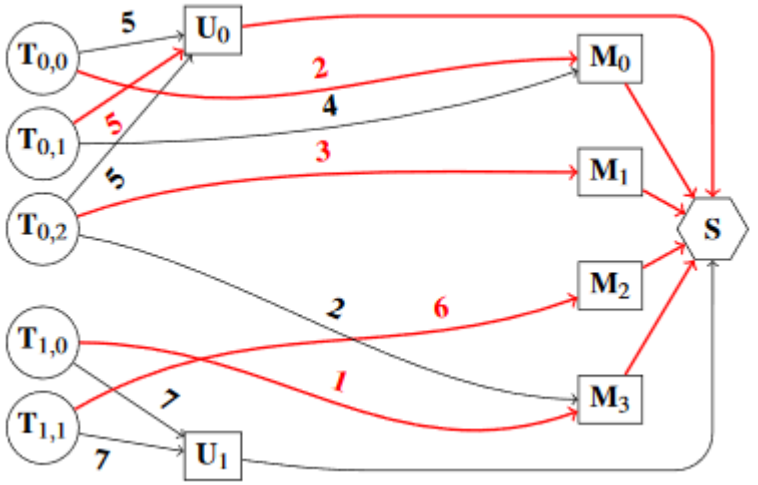
- Solve task scheduling with MCMF problem
- Construct a flow network:
  - Each task is a source node with supply 1
  - Each machine is connected to the sink with a zero-cost edge with capacity 1 (so only one task can be assigned to a machine)
  - Create all relevant connections between tasks and machines, in such a way that an edge between a task T_i and a machine M_i has a cost relative to the cost of running the task on the machine.



Source: "Firmament: Fast, Centralized Cluster Scheduling at Scale" Ionel Gog et al.

Quincy

- Solve task scheduling with MCMF problem
- Construct a flow network:
  - Each task is a source node with supply 1
  - Each machine is connected to the sink with a zero-cost edge with capacity 1 (so only one task can be assigned to a machine)
  - Create all relevant connections between tasks and machines, in such a way that an edge between a task T_i and a machine M_i has a cost relative to the cost of running the task on the machine.

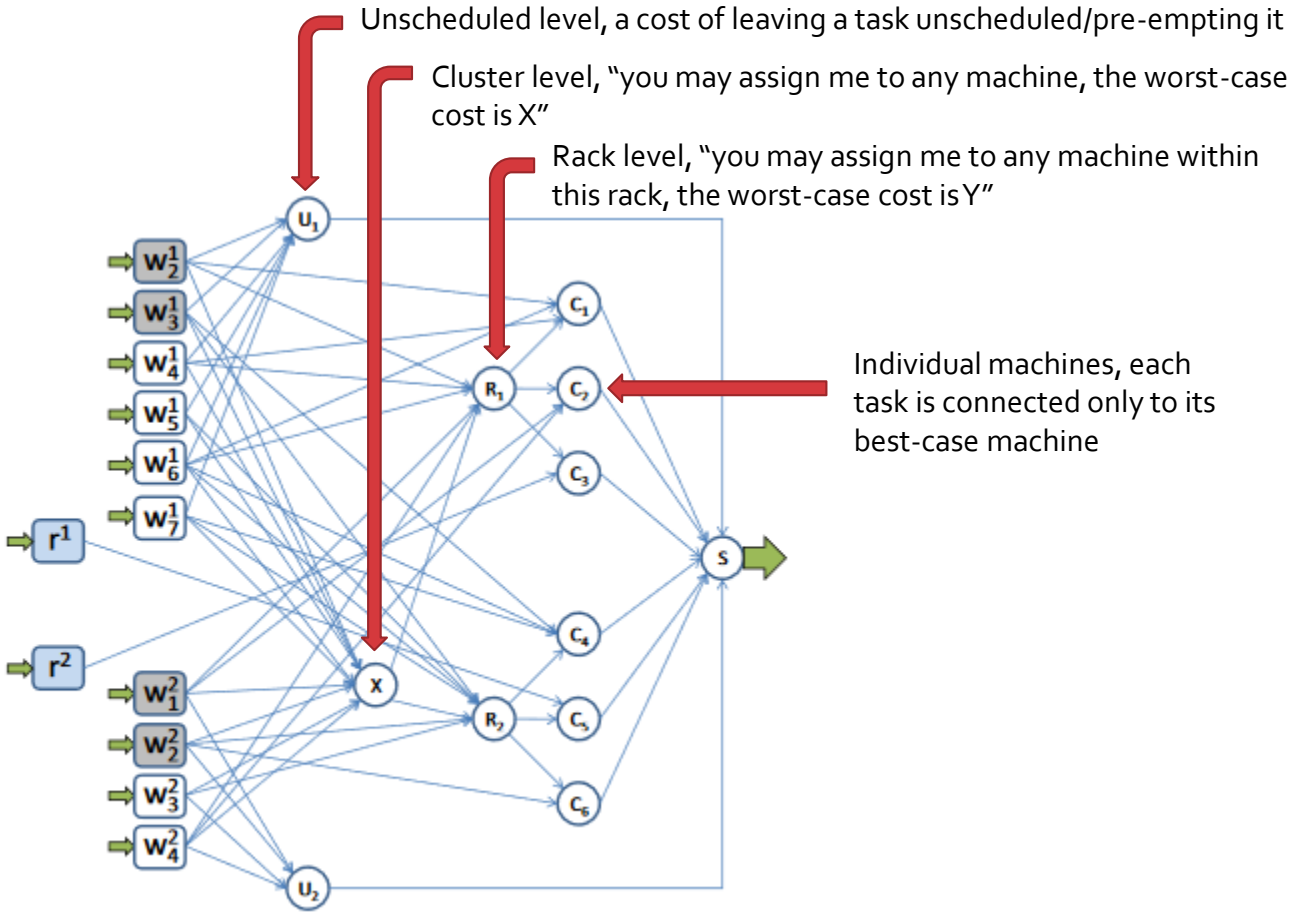


Source: "Firmament: Fast, Centralized Cluster Scheduling at Scale" Ionel Gog et al.

Potentially very large number of edges!

Quincy

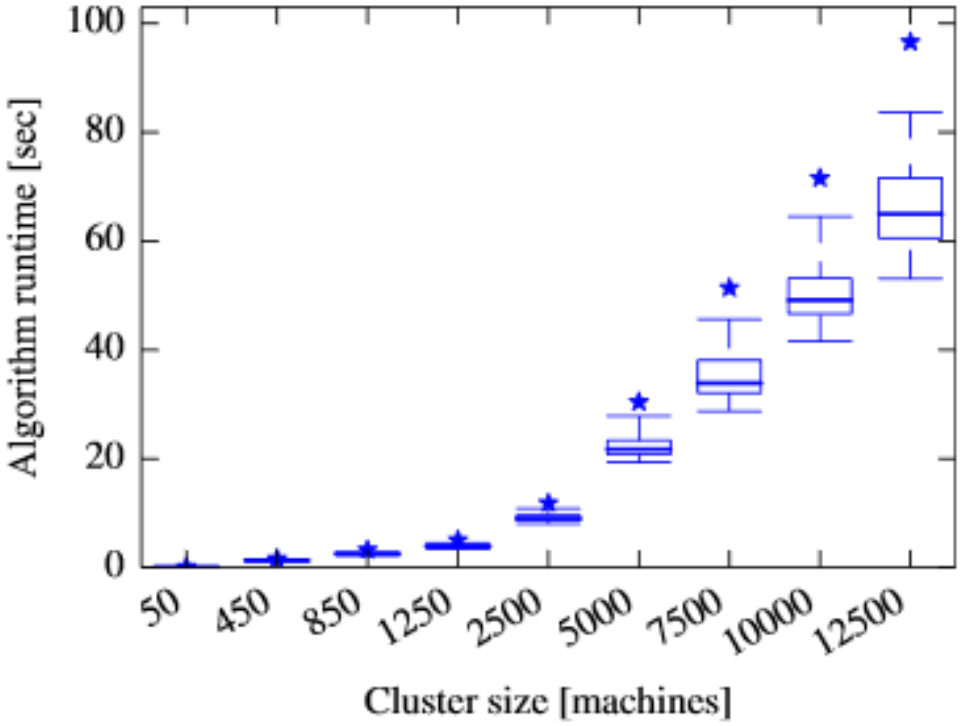
- Solution: aggregators



Source: "Quincy: Fair Scheduling for Distributed Computing Clusters" Michael Isard et al.

Quincy

- Quincy produces good scheduling, but is slow and does not scale…



Source: "Firmament: Fast, Centralized Cluster Scheduling at Scale" Ionel Gog et al.

What can we do about it?

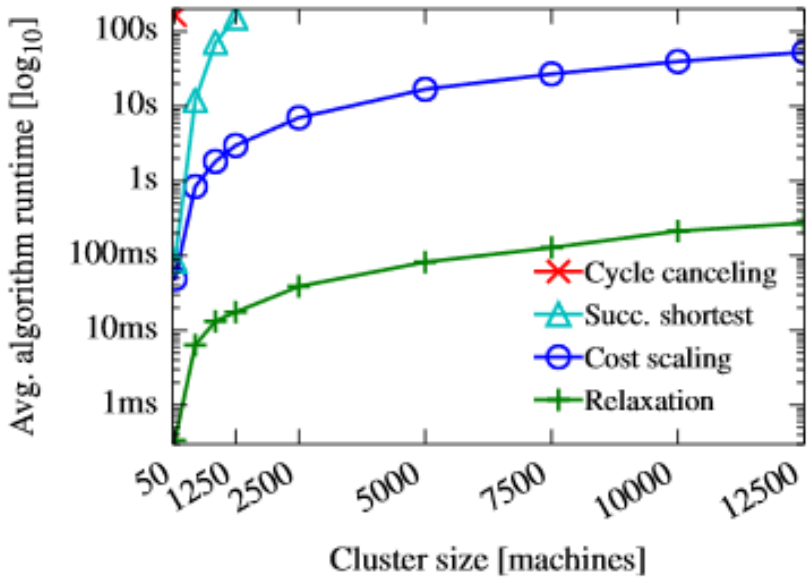
MCMF algorithms overview

Used by Quincy

Algorithm	Worst-case complexity
Relaxation	$O(M^3CU^2)$
Cycle canceling	$O(NM^2CU)$
Cost scaling	$O(N^2M\log(NC))$
Successive shortest path	$O(N^2U\log(N))$

Source: "Firmament: Fast, Centralized Cluster Scheduling at Scale" Ionel Gog et al.

N – number of nodes
M – number of edges
C – the largest edge cost
U – the largest edge capacity
M > N > C > U



Source: "Firmament: Fast, Centralized Cluster Scheduling at Scale" Ionel Gog et al.

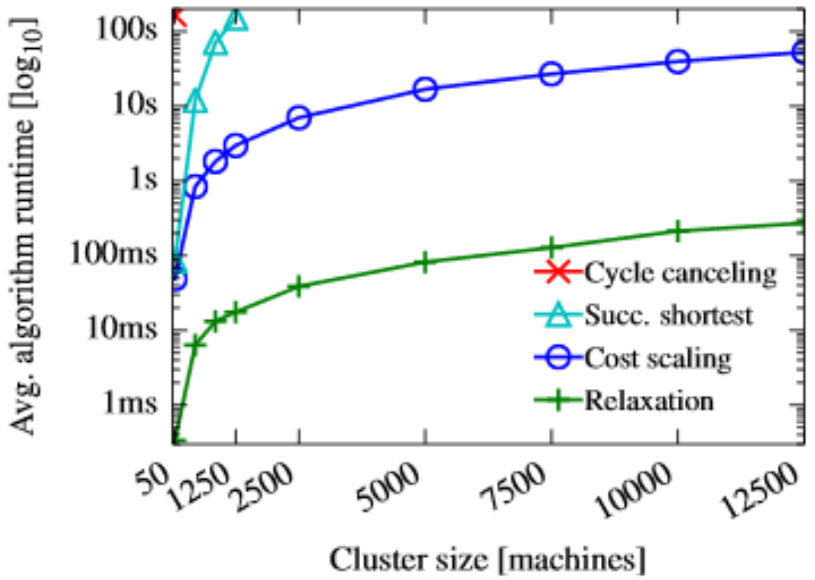
MCMF algorithms overview

Used by Quincy

Algorithm	Worst-case complexity
Relaxation	$O(M^3CU^2)$
Cycle canceling	$O(NM^2CU)$
Cost scaling	$O(N^2M\log(NC))$
Successive shortest path	$O(N^2U\log(N))$

Source: "Firmament: Fast, Centralized Cluster Scheduling at Scale" Ionel Gog et al.

N – number of nodes
M – number of edges
C – the largest edge cost
U – the largest edge capacity
M > N > C > U



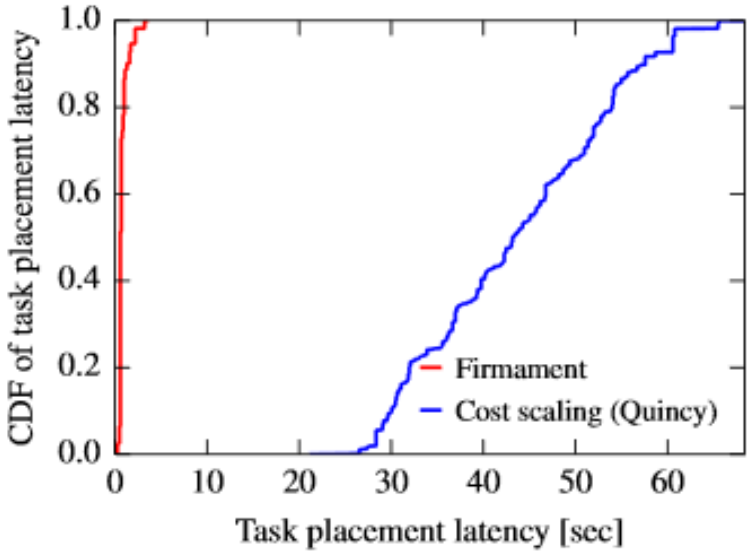
But relaxation is better on average!

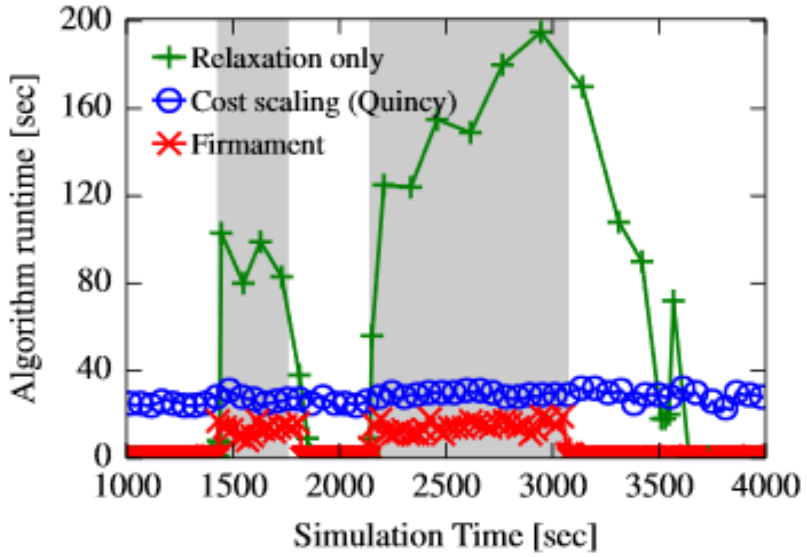
Source: "Firmament: Fast, Centralized Cluster Scheduling at Scale" Ionel Gog et al.

Firmament

- Use relaxation as a primary algorithm
- Use **incremental** version of cost scaling (40% speedup) as fallback
- Return result from whichever algorithm happens to finish first (both are run simultaneously – they are single-threaded)
- Allow for efficient switching between algorithms
- Provide efficient way of updating a flow network and extracting task placement

Results





Source: "Firmament: Fast, Centralized Cluster Scheduling at Scale" Ionel Gog et al.

Discussion

- 20 times faster placement than Quincy when scheduling tasks over 12,500 machines
- **Very detailed analysis of existing MCMF algorithms**
- Generic system which allows user to specify custom policies
  - e.g. network-aware policy
- But… it was shown that a policy can have direct impact on MCMF solver's performance (e.g. load-spreading policy is bad for *relaxation*)
  - The question if the optimizations presented by the authors work well with other policies is left unanswered.

The end

Thank you for attention.