How should a foreman communicate its resources to the master

February 10, 2014

Abstract

We discuss the problem of describing sets of workers for master-foremen relationships.

1 Problem Definition

Let f be a foreman connected to some master. Let W_f be the sets of workers connected to f. How should the foreman present information about W_f to the master?

Definitions

dispatchable task A task that can be executed in at least one worker in the hierarchy. **non-dispatchable** means that the tasks does not fit in any worker.

blocked task A dispatchable task for which the master has no knowledge of the worker in the hierarchy that can execute it. Note that such worker does exist in the hierarchy.

stranded task A stranded task is a dispatchable task that becomes non-dispatchable at the foreman queue. That is, it is a task waiting in the foreman queue that cannot be executed by the workers in W_f. It is considered *stranded* because the master cannot dispatch it to other available workers.

Requirements

- No task should be blocked.
- No task should be stranded.
- Information sent between master and foreman should not impact throughput.
- Throughput should be as close as possible to the case of a master with knowledge of all the workers in the hierarchy.

2 Current Implementation

An average for each resource is computed, and the foreman f sends an "average worker" description to the master. This implementation does not fulfill all the requirements:

- There are blocked tasks. "Large" tasks are never dispatched to the foreman because they are compared against the average worker, even if there is a worker large enough in the foreman hierarchy.
- There are stranded tasks. The task fitting in the average worker does not imply that there is a worker that fits the task in every resource. Once the task is dispatched to the foreman, no worker can execute it.

3 Easy case for discussion – only one resource measured

The foreman sends three worker descriptions: f_{min}, f_{avg}, and f_{max}, which represent the smallest, average, and largest worker, respectively. They are well-defined when there is only one resource measured.

- Tasks at the master are first matched to f_{min}. This is robust, as if all workers with the f_{min} disconnect, all the previously connected workers can still execute the task. Note that this does not mean that the task will execute in f_{min}, as this is left to the foreman to decide.
- If for all foremen the minimum workers fails to match the task, then the task is then compared with f_{avg}. This is done not to overload f_{max} workers.
- Finally, the task is then compared with f_{max}. This is done to avoid blocked tasks, as it provides the master with the knowledge of the largest workers. This is not robust, as when all f_{max} disconnect, the task becomes stranded.
- Stranded tasks are returned after some timeout to the master as a system failure, marking the task as stranded.

This implementation fulfills all of the requirements. The three workers descriptions are equivalent to three more workers connecting, which should be negligible in terms of throughput.

4 General case

f_{min}, f_{avg}, f_{max} describe the minimum, average, and maximum across all the resources.

- f_{min} reasoning and handling remains the same, and it is sill robust. Previously connected workers, per resource, are at least as big as f_{min}.
- Successful matches against f_{avg}, and f_{max} may now produce stranded tasks, even in the absence of disconnections (now f_{max} might not be realizable).

  There are some options to deal with stranded tasks:

  – Stranded tasks are not deleted at the foreman, and also the master is free to re-dispatch the tasks somewhere else. This may lead to some duplication of work, but guarantees there are no stranded nor blocked tasks.

    On task completion the master may inform all the foremen with the stranded task to delete the task. This reduces work duplication, but we need to keep track of all the foremen the task has been dispatched to.

  – Dispatch the task to a different foreman/worker. We would need to keep track of the foremen a task has been already dispatched.

  – (hackish ideas) Stranded tasks are returned to the master of the foreman immediately (which could be other foreman). For each stranded task, the master creates a *dummy task*, with all the resources requirements of the original task. Dummy tasks are then dispatched to other foremen. A foreman can report to its master if there is a worker that could fit the dummy task, and the master dispatches the original task to such worker.

5 Concerns and improvements

- Reliance on the lowest common denominator f_{min} avoids stranded tasks, but we need testing to ensure throughput is not destroyed.
- If the variance in W_f is large and its resource distribution is multimodal, we can create more than one average per foreman. This seems appropriate for multi-level hierarchies.