Dispatcher/Worker

Context

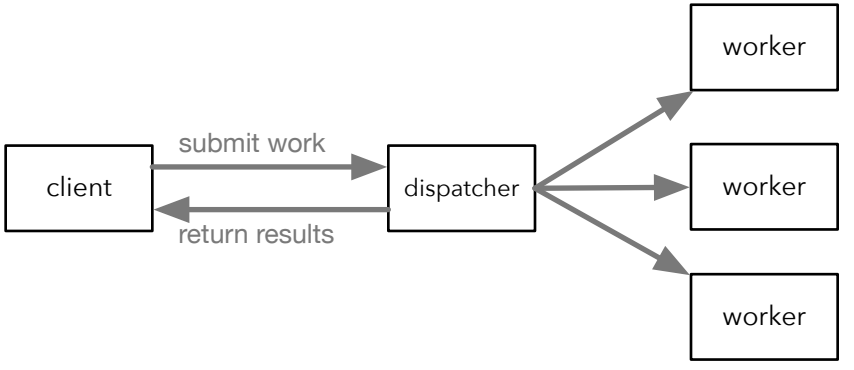
Partitioning work into semantically identical sub-tasks.

Forces

- Clients should not be aware that the whole body of work was subdivided.
- Sub-tasks may need to be coordinated, depending on the algorithm.
- The means of partitioning work and the task granularity should not affect the client or the processors.

Discussion

- The dispatcher may have the same API as the workers, to allow substitution or recursive task breakdown.
- If running in many processes, fault tolerance is a concern.
- Some tasks parallelize easily, others require additional processing to combine the results.
- Workers may be permanent, ephemeral, or drawn from a pool.



Examples: Fork-join, Map/Reduce, GPUs

Aliases: Master/slave, Manager/worker.