Task scheduling for Jupiter

ANRG Lab - USC

Problem Statement

Given a task graph (DAG) with N subtasks and M machines, find the best allocation/schedule of the N subtasks to the M machines so that cost and makespan can be optimised.

2. Approach

For finding the optimal schedule, the meta-heuristic Binary Particle Swarm Optimization(BPSO) algorithm has been chosen for the intuitive and availability of opensource tools. The solution space and the choice for encoding particles in this space has been made. The complete solution space involves all the permutations of the N subtasks over the M machines. Each solution particle P is represented as a vector of length N taking values on integral points and the set of all solutions for a N-hypercube of side length M. Further, the solution P is encoded into a binary representation where the N components of the particle P are wirtten in base-2 numeral system which gives us a vector Q.

4. Cost function

The costfunction for this optimisation problem contains 2 components -

- $-price of running the DAG task (price) is a linear blending of - 1. Computation cost ($/subtask) -> N x M Compute cost matrix
  2. Communication cost ($/MB data transfer between nodes) -> M x M communication cost matrix
- Time taken for running the DAG Task end to end (makespan) is built from -
  1. Compute time -> N x M compute time matrix
  2. Communication time -> M x M Bandwidth matrix and M x 1 Machine Network Setup Latency Vector.

Total cost is a linear blending of the above two costs using the `price` and the `makespan`

3. Progress so far

After the solution space, the representations of particles and the cost functions were decided,the algorithm is implemented and evaluated using the open source library PySwarms.

For evaluating quality of solutions, bruteforce and random algorithms are also implemented to find the best solution and compare time/complexity advantage of the PSO algorithm for this use case.

Emperical evaluations on problem sizes of (N,M) = (4,8) and (8,16) have been done.

4. In progress

1. Making a testing tool for larger problem sizes and automating evaluation.
2. Implementing Veeramachaneni PSO - a variant of BPSO for k-base numeric system representation of particle solution.

© sampy-2020 | Made with ❤ using StackEdit, Vue.JS, texme(fork), markedjs, KATEX, js-yaml, tocbot & Semantic-ui

Theme -> default