1 Problem Formulation

Harnessing the power of wide-area distributed computing platforms is a major challenge nowadays, and scheduling is crucial for achieving this goal. Traditional scheduling minimizes the makespan of the execution of a given set of jobs. In most practical situations, the exact computation of a minimal makespan is NP-Hard.

2 Proposed Solution

An interesting alternative introduced by D. Bertsimas, D. Gamarnik in 1999, is a schedule that optimizes the steady-state operation of the system. This approach is proven to be particularly well-suited for master-slave tasking, and in general, for divisible load applications.

Because the schedule is periodic and computable in polynomial time, it is possible to observe its actual performance in a period, inject that information into the polynomial methods, and re-compute the optimal steady-state schedule for upcoming periods. This is particularly useful in wide-area distributed systems where hard-to-predict communications jams may occur.

3 Theoretical Framework

Banino et al. (2004) use a nonoriented graph to model a hybrid computer platform. The optimal steady state is defined as the fraction of time spent computing and the fraction of time spent sending or receiving tasks along each communication link, so that the overall number of tasks processed at each time step is maximum.

4 Demo Construction

Our demo (Steady State Scheduler V 1.0) was written in Python® and allows us to change execution and computation times in order to show the effects of adaptivity.

This demo:
- Allows to change communication and execution times.
- Uses GipPR to solve the Master-Slave linear programming problem.
- Uses the previous algorithms to build the schedule.
- Displays the generated schedule in a Gantt chart.

5 Conclusion

Currently the demo accepts variations in communication and execution times. By observing these variations it become apparent that, with one master, no throughput is higher than two, independent of the number of nodes, communication and execution times.

We expect to evaluate a problem with more nodes, simulating a grid based platform and we will used this results in order to construct a prototype system related to a specific problem.

6 References


