Matching Theory and Virtual Machines

Kristen Hines
School of Electrical and
Computer Engineering
Virginia Tech
Email: kphines@vt.edu

Ferdinando Romano
School of Electrical and
Computer Engineering
Virginia Tech
Email: fmromano@vt.edu

***Abstract*—Insert abstract here.**

I. Introduction

This project is focused on applying matching theory to a virtual machine job assignment problem.

II. Related Work

Most similar topics: VM Cosheduling and VM Migration papers.

Related to VM Coscheduling: Distributed Selfish Load Balancing

Related to VM Migration: Seen As Stable Marriage and Online VM Shuffling (continuation of VM Migration work).

III. Method Description

High level description if needed.

A. Problem Formulation

The problem this paper is exploring is how to optimize job assigment to separate computer clusters. Each of these jobs perform differently on different cores types. These core types can be graphical processors, computational processors, or something else. Each of these clusters have different core types. Each job can only be divided into a finite number of threads, and each job is assigned to one computer at a time.

(probably a good place to introduce matching theory and the college admissions game). This problem can be formed as a college admissions game. The differences are that the institutions have multiple quotas, each applicant can fill multiple slots of different types, applicants prefer different slot types over others, and an applicant cannot be divided among multiple institutions.

Key assumptions for this problem are: the virtual machines will be treated like flexible computers, jobs are submitted at the same time, chosen jobs are completed simulatenously, unchosen jobs will be submitted with the next round, no indifference, and no externalities.

B. Proposed Algorithm

Things to do: Write the math and assumptions, state why we are doing this, and show that this will be stable as well.

1) Stability of Algorithm: The proposed algorithm produces a stable matching because in each iteration, the college admissions game is used to find a set of stable matchings. Out of the jobs listed in the resultant set of stable matchings, the job that can use the most processors is preferred most by every computer. Thus, that job will be matched with its first choice and its matching to a computer is a stable matching. Thus, each pair produced by an iteration of the proposed algorithm is stable and therefore the final matching is stable.

2) Optimality of Algorithm: Whether the matching is optimal can be understood in multiple senses. In this section, three different approaches to optimality are discussed as they apply to the proposed algorithm.

<u>Resource Utilization</u>: A simple goal of the proposed algorithm would be to maximize processor utiliztion so that no computing resources go unused/wasted.

The proposed algorithm does not alway maximize processor utilization. However, it does in every iteration where the preferred computer of the job with the greatest possible processor utilization has at least as many processors available as either i) that job can use or ii) any other computer has. This situation is common because, often, a computer with more available processors will outperform one with fewer. The exceptions occurs where there is a computer that has special purpose procesors that significantly outperform those available at other computers and this computer does not meet either of conditions i) or ii) listed above.

<u>Total Job Completion Time</u>: The total computation time, i.e., the sum of total computation times for each job, is another good measure of the optimality of the proposed algorithm.

Assuming individual jobs cannot take advantage of processors previously used by other jobs that have completed, the proposed algorithm minimizes total computation time whenever jobs that use more processors are jobs that would take longer to complete than any other job. By 'take longer to complete', we mean take longer than other jobs if the other jobs were to use a subset or superset of the processors used by the first job. When this condition is met, the job that takes the longest is given the greatest speed possible, the job that takes the 2nd longest is given the next greatest speed possible for it, and so on. Thus, total computation time is minimized.

In the proposed algorithm, this condition that jobs use more processors take longer is not guaranteed. However, it is strongly encouraged by the proportional fairness of the algorithm: Jobs that would take longer to complete are incentivized to be able to use more processors.

Proportoinal Fairness: In the proposed algorithm, jobs' individual computation times/total required processing are not factored into the preferences and so have no bearing on the matchings. Instead, it is the processor utilization ability of a job that effects its ranking. This leads to a proportional fairness in which jobs that are shorter are still given a fair amount of processing power so that they will not take very long. On the other hand, jobs that require more processing power, i.e. would take longer, are incentivized to be able to use more pocessors than jobs that do not take as long.

For a job that would take time to complete $\tau_1 > \tau_2$, where τ_2 is the completion time for a second job, Job 1 would reduce its completion time by an absolute amount $\Delta\tau_1 = \tau_1 - \frac{\tau_1}{f}$ if it could increase its speed by a factor f. Similarly, for Job 2, $\Delta\tau_2 = \tau_2 - \frac{\tau_2}{f}$. Thus, $\tau_1 = \Delta\tau_1(1 - \frac{1}{f})$ and $\tau_2 = \Delta\tau_2(1 - \frac{1}{f})$. Since $\tau_1 > \tau_2$, we have $\frac{\Delta\tau_1}{1-\frac{1}{f}} > \frac{\Delta\tau_2}{1-\frac{1}{f}} \implies \Delta\tau_1 > \Delta\tau_2$. Therefore, Job 1 has more to gain by increasing its speed by a given factor than Job 2 does and so Job 1 has a greater incentive to be able to use more processors.

IV. Results and Discussion

Show results and discuss what they represent.

V. Conclusion

The conclusion goes here.

VI. Future Work

List possible future work here.

VII. Individual Contributions

Enumerated individual contributions.

Acknowledgment

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.