Study of an Iterative Resource Allocation Technique to Minimize Machine Completion Times in a Distributed Computing System

Luis D. Briceño[1], Mohana Oltikar[1], H. J. Siegel[1,2], and Anthony A. Maciejewski[1]

[1]Electrical & Computer Engineering, [2]Department of Computer Science

{ldbricen, mohana.oltikar, hj, aam}@colostate.edu

This research was supported by the NSF under grant CNS-0615170 and by the Colorado State University George T. Abell Endowment

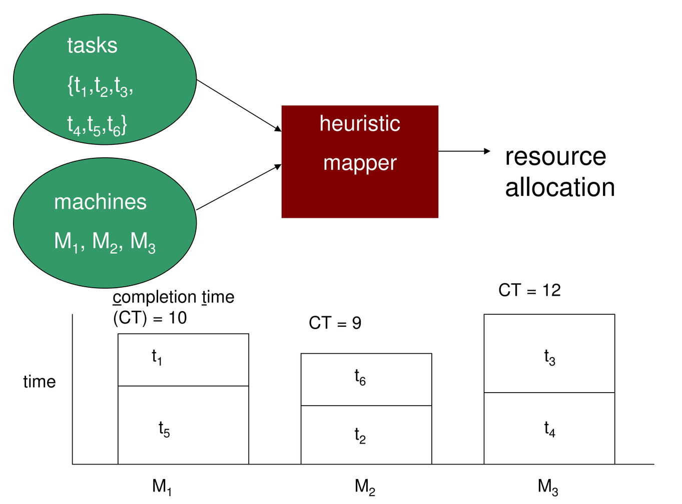
Introduction

- What is heterogeneous computing (HC)?
  - HC is the coordinated use of different types of machines, networks, and interfaces to maximize performance
- How do we maximize the performance?
  - a common optimization criterion is to minimize the completion time of all the machines (makespan)
- makespan machine:
  - machine that is the last to complete
- non-makespan machines:
  - the machines that are not last to complete

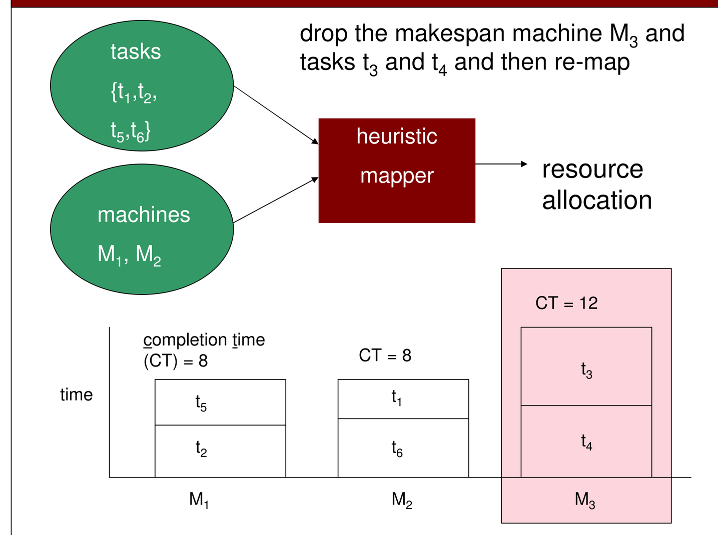
Introduction

- What is the iterative approach?
  - the iterative approach attempts to decrease the finishing time of each machine in a given resource allocation by repeatedly running a mapping heuristic to minimize makespan of the considered machines and tasks
  - the makespan machine and tasks assigned to it are removed from the considered machines and tasks after every mapping
- What is the goal of this study?
  - identify which resource allocation heuristics can effectively employ the iterative approach
- Genitor and Min-Min are examined here

Iterative Approach: Original Mapping



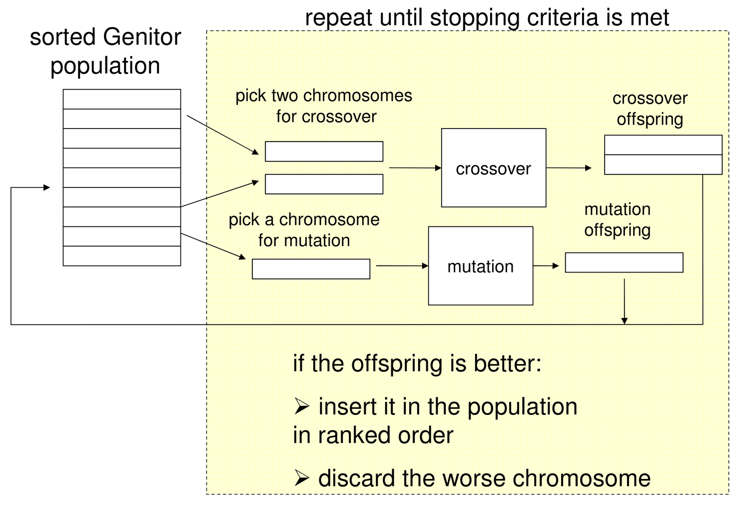
Iterative Approach: First Iterative Mapping



Genitor

- one resource allocation heuristic considered is Genitor
- What is Genitor?
  - Genitor is a type of genetic algorithm (GA)
  - each chromosome represents a possible mapping
  - uses a sorted population with the best chromosomes
  - offspring generated by crossover and mutation
  - offspring are kept based on their rank in the population
    - good mappings stay in the population
    - bad mappings are discarded

Genitor: Procedure



Genitor: Performance with the Iterative Mapping

- the mapping resulting from the previous iteration (of the iterative approach) is seeded into the population of the current iteration
- the ranking in Genitor guarantees the mapping for the current iteration is either:
  - the seeded mapping from the previous iteration
  - or a mapping with a smaller makespan for the considered machines and tasks (compared to that from previous iterations)
- thus, the iterative technique can result in:
  - an improvement
  - or no change

Reference:
L. D. Briceño, M. Oltikar, H. J. Siegel, and A. A. Maciejewski, "Study of an Iterative Technique to Minimize Completion Times of Non-Makespan Machines," submitted.

Note: An e-copy will be available at:
http://www.engr.colostate.edu/~hj/complete_vita.pdf

Min-Min

- Another resource allocation heuristic considered
- Min-Min procedure

1. for each task find the machine that completes the task earliest
2. from all (task t_i, machine m_k) pairs found above select the pair (t_i, m_k) with the minimum completion time
3. add it to the mapping
4. update the ready time value of m_k
5. repeat above steps until all tasks are mapped

Min-Min

- performance with the iterative approach:
  - if ties (in steps 1 and 2) are broken deterministically then the resource allocation will not change over iterations
    - the proof is included in the reference
  - if ties (in steps 1 and 2) are not broken deterministically then the makespan may become larger
    - example included in the reference

Min-Min: Iterative Approach

- Theorem:
Using the Min-Min heuristic the individual completion times for each machine do not improve over iterative mappings of the iterative approach, if ties (in steps 1 and 2) are broken deterministically
- this is proven by using induction
  - the proof is included in the reference

Conclusions

- Genitor can improve the original resource allocation using the iterative approach
- Min-Min does not improve upon the original resource allocation with the iterative approach
  - if ties are broken deterministically, then the resource allocation does not change
  - if ties are broken randomly, then the resource allocation may get worse
- other heuristics were also studied
  - these heuristic are included in the reference