

HAGEES: A High Availability Guaranteed Energy Efficient

Scheduling Algorithm for Homogeneous Clusters

Ziliang Zong, Mais Nijim, Mohammed Alghamdi, Xiao Qin*

Department of Computer Science

New Mexico Institute of Mining and Technology

Socorro, New Mexico 87801

{ zsong, mais, alghamdi, xqin } @ nmt.edu

Abstract

High performance clusters play a vital role nowadays both in the industry and research area. In the past decade, some scheduling schemes have been investigated to achieve high availability or energy efficiency in large-scale clusters. However, leveraging scheduling algorithms to ensure high availability and performance while conserving energy remains an open problem. To bridge this technology gap, in this paper, we propose a high availability guaranteed energy efficient scheduling strategy (HAGEES for short) that aims at comprehensively consider the most important three factors, namely, availability, energy conservation, and high performance.

1. Introduction

With the development of VLSI technology and high-speed networks, cluster systems have been widely used for a diverse set of parallel applications like molecular design, weather modeling and complex image rendering. However, drawbacks like tremendous energy consumption, lower availability always attack people's confidence in further applying cluster systems to modern industry and scientific research applications.

Energy consumption problems of clusters have been extensively researched by previously studies [1] [2]. The availability issue of cluster system was first presented in [3] and then further studied in [4] [6]. Duplication based strategy [5] was provided to balance the execution time and energy cost for homogeneous clusters in our previous research paper [7].

Although most existing scheduling strategies are conducive to achieving high availability or energy efficiency in large-scale clusters, leveraging scheduling algorithms to ensure high availability and

performance while conserving energy remains an open problem. To bridge this technology gap, in this paper we propose a high availability guaranteed energy efficient scheduling strategy (HAGEES for short) that aims at comprehensively addressing the most important three factors, namely, availability, energy conservation, and high performance.

2. System Model

A cluster system in this study is based on the Beowulf cluster model which has a set $P = \{p_1, p_2, \dots, p_m\}$ of computational nodes connected by high-speed interconnects (Fig.1).

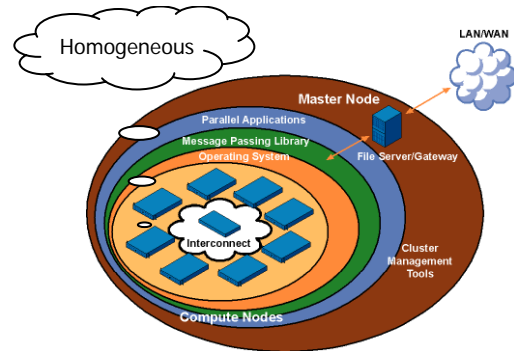


Fig.1 Homogeneous Beowulf cluster model

It is assumed that the computational nodes are homogeneous, meaning that all the computing nodes are identical in their capabilities. Similarly, the underlying interconnection is assumed to be homogeneous and, thus, communication overhead of a message with fixed data size is considered to be the same. Computation and communication can take place simultaneously. This assumption is reasonable because

each computational node in a modern cluster has a communication coprocessor that can be used to free the processor in the node from communication tasks.

The energy consumption experienced by a cluster can be expressed as:

$$E = EN + EL \quad (1)$$

where EN is the energy consumption of computing nodes, and EL is the energy consumption of sending information through links.

The energy consumption EN of computing node in Eq. (1) can be written as

$$\begin{aligned} EN &= \sum_{i=1}^{|V|} en_i = \sum_{i=1}^n (PN \cdot t_i) \\ &= PN \sum_{i=1}^n t_i. \end{aligned} \quad (2)$$

where en_i is the energy consumption caused by node i , PN is the power of the node, and t_i is the total execution time of tasks running on node i . The energy consumed by the interconnections is expressed as

$$\begin{aligned} EL &= \sum_{a=1}^m \sum_{b=1, b \neq a}^m EL^{ab} \\ &= \sum_{i=1}^n \sum_{j=1, j \neq i}^n \sum_{a=1}^m \sum_{b=1, b \neq a}^m (x_{ia} \cdot x_{jb} \cdot PL \cdot c_{ij}). \end{aligned} \quad (3)$$

where EL^{ab} is the energy consumption of the link between nodes p_a and p_b . PL in Eq. (2) is the power of the link. Element x_{ia} is "1" if the i th task is assigned to node p_a and is "0", otherwise. c_{ij} is the communication cost.

The availability of a computational node is quantified as the probability that all tasks allocated to the node can be successfully completed even in the presence of the node's hardware and software faults. Availability of different computing nodes in our research is based on the analysis data from actual cluster systems. The higher availability a node has, the higher precedence a node will have to be allocated tasks for executing.

3. The HAGEES Algorithm

To allocate tasks to different computing nodes, HAGEES will consider three factors in the order of availability, execution time and energy cost. HAGEES will always try to allocate tasks to nodes with the highest availability precedence and try to duplicate as more tasks as possible as long as these duplicated tasks will reduce execution time and will not increase energy consumption tremendously.

Basically, the HAGEES algorithm has three phases to generate the final scheduling list.

Phase1: Generate a task sequence

HAGEES will construct an ordered task sequence using the concept of level, which of each task is defined as the length in computation time of the longest path from the task to the exit task. All the tasks are placed in a queue in an increasing order of the levels. Eq. (4) shows how to calculate the level of task i .

$$L(v_i) = \begin{cases} t_i, & \text{if successor}(i) = \Phi \\ \max_{k \in \text{succ}(i)} (\text{level}(k)) + t_i & \text{otherwise} \end{cases} \quad (4)$$

Phase2: Calculate important parameters

In this phase, HAGEES will calculate some important parameters, which the algorithm rely on. The important notation and parameters are listed in Table 1.

Table 1. Important Notation and Parameters

$EST(v_i)$	Earliest start time of task v_i
$ECT(v_i)$	Earliest completion time of task v_i
$FP(v_i)$	Favorite predecessor of task v_i
$LACT(v_i)$	Latest allowable completion time of task v_i
$LAST(v_i)$	Latest allowable start time of task v_i

The earliest start times of all the other tasks can be calculated in a top-down manner by recursively applying the second term on the right side as follows.

$$EST(v_i) = \begin{cases} 0, & \text{if predecessor}(i) = \Phi \\ \min_{e_{ji} \in E} \left(\max_{e_{ki} \in E, v_k \neq v_j} (ECT(v_j), ECT(v_k) + c_{ki}) \right), & \text{otherwise} \end{cases}$$

The earliest completion time of task v_i is expressed as the summation of its earliest start time and execution time. Thus, we have

$$ECT(v_i) = EST(v_i) + t_i.$$

The favorite predecessor $FP(v_i)$, $LACT(v_i)$, $LAST(v_i)$ are defined as below respectively

$$FP(v_i) = v_j, \text{ where } \forall e_{ji} \in E, e_{ki} \in E, j \neq k \mid ECT(v_j) + c_{ji} \geq ECT(v_k) + c_{ki}.$$

$$LACT(v_i) = \begin{cases} ECT(v_i), & \text{if successor}(i) = \Phi \\ \min_{e_{ij} \in E, v_i \neq FP(v_j)} \left(\min_{e_{ij} \in E, v_i \neq FP(v_j)} (LAST(v_j) - c_{ij}), \min_{e_{ij} \in E, v_i \neq FP(v_j)} (LAST(v_j)) \right), & \text{otherwise} \end{cases}$$

$$LAST(v_i) = LACT(v_i) - t_i.$$

Phase3: Availability-Aware Energy efficient task allocation and duplication phase

```

1. Sort all the available processors in the descending order of availability; /* P0 will have the
highest availability after sorting */
2.  $v =$  first waiting task of scheduling queue;
3.  $i = 0$ ;
4. assign  $v$  to  $P_i$ ;
5. while (not all tasks are allocated to computational nodes) do
6.    $u = FP(v)$ ;
7.   if ( $u$  has already been assigned to another processor) then
8.     if ( $LAST(v) - LACT(u) < c_{uv}$ ) then /* if duplicate  $u$ , we can shorten the schedule length */
9.       moreenergy =  $en_u - el_{uv}$ ; /*energy increase*/
10.      if (moreenergy  $\leq$  threshold  $h$ ) then /* increased energy less than our threshold*/
11.        assign  $u$  to  $P_i$ ; /*duplicate  $u$ */
12.        if  $v$  has another predecessor  $z \neq u$  has not yet been allocated to any node then
13.           $u = z$ ;
14.        else
15.          if  $u$  is entry task then
16.             $u =$  the next task that has not yet been assigned to a node;
17.             $i++$ ;
18.          else
19.            for another predecessor  $z$  of  $v$ ,  $z \neq u$ ,
20.            if ( $ECT(u) + cc_{uv} = ECT(z) + cc_{zv}$ ) and  $z$  hasn't been allocated) then
21.               $u = z$ ; /* do not duplicate*/
22.            else
23.              for another predecessor  $z$  of  $x$ ,  $z \neq u$ ,
24.              if ( $ECT(u) + cc_{uv} = ECT(z) + cc_{zv}$ ) and  $z$  hasn't been allocated) then
25.                 $u = z$ ; /* do not duplicate*/
26.            else allocate  $u$  to  $P_i$ ;
27.             $v = u$ ;
28.            if  $v$  is entry task then
29.               $v =$  the next task that has not yet been allocated to a computational node;
30.               $i++$ ;
31.              assign  $v$  to  $P_i$ ;
32. return schedule list, schedule length and energy consumption;

```

Fig. 2. Pseudocode of phase 3 in the HAGEES algorithm.

In this phase, HAGEES is already given a parallel application presented in form of a DAG, it will allocate each parallel task to a most reliable computational node in a way to aggressively shorten the schedule length of the DAG while conserving energy consumption. The pseudocode in Fig. 2. shows the details of this phase in the HAGEES algorithm, which aims to provide the most reliability, the greatest energy savings when it reaches the point to duplicate a task. In other words, when HAGEES is trying to allocate tasks to a new computational node, it will first check the availability of every idle node and try to find the most reliable one to run the tasks. In the case of tasks which in the same critical path have to be allocated to the same node, HAGEES will check the time benefit. If duplicate

this task can shorten the schedule length, it will further check the energy consumption. If duplicating the task leads to much more energy consumption, which is greater than our threshold, HAGEES will discard the duplication operation. Otherwise, HARGEES will duplicate this task to shorten the schedule length. As such, HARGEES is able to make the best tradeoff among energy savings, high availability and performance in large-scale clusters.

10. References

- [1] L. Benini and G. De Micheli, Dynamic Power Management: Design Techniques and CAD Tools, Kluwer, 1998.

- [2] J. Chase and Ron Doyle, "Energy Management for Server Clusters," *Proc. the 8th Workshop Hot Topics in Operating Systems (HotOS-VIII)*, pp. 165, May 2001.
- [3] O.C. Ibe, A. Sathaye, R.C. Howe, K.S. Trivedi, "Approximate Availability Analysis of VAXcluster Systems", *IEEE Trans. Reliability*, Vol.38, No.1, Apr. 1989, pp146-152
- [4] V.B. Mendiratta, "Reliability Analysis of Clustered Computing Systems", the International Symposium on Software Reliability Engineering, 1998, pp268-272
- [5] S. Ranaweera, and D.P. Agrawal, "A Task Duplication Based Scheduling Algorithm for Heterogeneous Systems," *Proc. Parallel and Distributed Processing Symp.*, pp.445-450, May 2000.
- [6] H. Han, Sun, J.J., Levendel, H., "A generic availability model for clustered computing systems", *Dependable Computing*, 2001. Proceedings. 2001 Pacific Rim International Symposium, 2001, pp241-248
- [7] Z.-L. Zong, A. Manzanares, B. Stinar, and X. Qin, "Energy-Efficient Duplication Strategies for Scheduling Precedence Constrained Parallel Tasks on Clusters", *International Conference on Cluster Computing*, 2006.