Novel Algorithms for Load Balancing using Hybrid Approach in Distributed Systems

by:
Chun Yi Chu
A.vijay Shankar

For CMPE-655
Multiple Processor Systems
Instructor:
Muhammad Shaaban
Dynamic Load Balancing

The dynamic load balancing (DLB) received considerable attention.

In DLB, the workload is **distributed** among the available resources in system.

DLB adapts to the changing system state, therefore can achieve better performance in distributed system with *fluctuation workload*.

Can achieve superior performance in terms of:

- Resource Utilization
- Throughput
- Respond time
DLB - Centralized Approach

In traditional centralized approach of DLB, a central node acts as the load balancing unit and dispatch the jobs to different nodes in the system.

Advantages:

- Simple implementation
- Easy to determine parallel computation termination

Drawbacks:

- Scalability – Can became a bottleneck (Contention) and susceptible to failures.
In traditional decentralized approach, all (or many) nodes participating in load balancing.

**Advantages:**
- Better scalability
- Better fault tolerance
- Less effective dynamic tasking overheads

**Drawbacks:**
- Increased inter-node communication and synchronization
- Requiring a termination detection algorithm
Hybrid Dynamic Load Balancing

Hybrid DLB overcomes the limitations of centralized and decentralized approaches and ensure the tradeoff between scalability and efficiency.

The hybrid DLB approach divides the nodes of distributed system into virtual groups (cluster).

Using the notion of divide-and-conquer, hybrid DLB reduce the communication overhead and also eliminates the scalability problem.
Hybrid Dynamic Load Balancing

By doing Hybrid DLB, advantages are:

- Overcomes the constraints of centralized and decentralized approach
- Utilizes efficient load index
- Consider heterogeneity in Computing resources
- Aids highly loaded node to obtain lightly loaded node in lesser time
- Is applicable to wide range of applications viz. CPU-bound, memory-bound, I/O-bound, and any combination of these
Hybrid DLB - Model

Figure of simplified hybrid DLB distributed system:

Example consists of 16 nodes, and clusters to 4 groups.
Hybrid DLB – Handle Overload

*Supernode* periodically collects the workload information from nodes in the cluster, finds out the *average workload* on the system, and set the *dynamic threshold value*.

When a node workload *larger* than *dynamic threshold*, that node becomes *overloaded*. It will searches for the *lightly loaded node* to offset some of its workload.

It will search within the same cluster (group) first, instead of the entire system. This significantly *reduces* message complexity.
Hybrid DLB – Clustering

Clustering is the primary step of the hybrid algorithm, it will divide the nodes into clusters in order to achieve lesser communication overhead.

The algorithm for clustering is based on the conventional theory of integer partition.

\[
\begin{align*}
5 & \quad [1,1,1,1,1], \\
   & \quad [1,1,1,2], \\
   & \quad [1,2,2], \\
   & \quad [1,1,3], \\
   & \quad [2,3], \\
   & \quad [1,4], \\
   & \quad [5] \\
6 & \quad [1,1,1,1,1,1], \\
   & \quad [1,1,1,1,2], \\
   & \quad [1,1,2,2], \\
   & \quad [1,1,3], \\
   & \quad [2,2,2], \\
   & \quad [1,2,3], \\
   & \quad [1,1,4], \\
   & \quad [3,3], \\
   & \quad [2,4], \\
   & \quad [1,5], \\
   & \quad [6] \\
7 & \quad [1,1,1,1,1,1,1], \\
   & \quad [1,1,1,1,1,2], \\
   & \quad [1,1,1,2,2], \\
   & \quad [1,1,1,1,3], \\
   & \quad [1,2,2,2], \\
   & \quad [1,1,1,4], \\
   & \quad [1,1,2,3], \\
   & \quad [2,2,3], \\
   & \quad [1,3,3], \\
   & \quad [1,1,5], \\
   & \quad [1,2,4], \\
   & \quad [3,4], \\
   & \quad [2,5], \\
   & \quad [1,6], \\
   & \quad [7]
\end{align*}
\]
Hybrid DLB – Clustering

1. Generate all possible partitions of $n$
2. Remove all partitions that have 1 as a part and partition that has $n$ as a part
3. Choose one partition randomly from the remaining partitions
4. Set number of clusters $m =$ number of parts in chosen partition
5. Set size of each cluster that is equal to corresponding part in the partition
6. Arrange nodes of system into nondecreasing order of load
7. Create each cluster by inserting nodes from top and bottom of the list alternately while next cluster will continue the same series
Hybrid DLB - Super node Selection Strategies

- Importance of Supernode:

  It has the responsibility of defining the dynamic threshold value.

- How?

  It periodically collects load information from all the nodes in the cluster and calculates the average value and set it as threshold value.

  It then broadcasts new value to the nodes in the cluster.
Drawback of static threshold policy

- It is a fixed threshold value for the entire execution.
- So, it will find out lesser number of nodes for load balancing with respect to current system situation.
- It is not much useful for improving system performance.
Selection Algorithm 1

(a)

(b)

(c)
Input: list of node ids
Output: supernode

//procedure utilized by the node P_i that does not find recent threshold T

Supernode_Algorithm_A(list of node ids)
1. IF (supernode is not selected OR supernode crash is detected)
2.   WHILE (supernode message is not received)
3.     send “Permitted New Supernode” message to P_j
4.   IF (P_j is alive)
5.     receive “Supernode” message
6.   ELSE
7.     j = j-1
8.     IF (j==i)
9.     broadcast “Supernode” message
10.   EXIT
11. EXIT
Message Complexity

**Best Case Scenario** (next highest node id is alive):

- \( \text{Msg} = 1 + (K-1) = k = O(k) \)

  where \( k = \text{size of the cluster}. \)

**Worst Case Scenario** (all higher node ids are down)

- \( \text{Msg} = (K - h) + (K - 1) \)

  where \( h = \# \text{ of nodes that have higher id than the detected node}. \)
Selection Algorithm 2

Input: load of all nodes in the group
Output: supernode
//procedure utilized by the node P_i that does not find recent threshold

Supernode_Algorithm_B() 
1. flag = 1; i=1
2. IF (supernode is not selected OR supernode failure is found)
3.   collect load of other resource nodes in the cluster
4.   arrange load values including its own load value in increasing order in list L
5.   p = length(L)/2
6.   find average-valued-node corresponding to load at mid position p
7. WHILE (supernode message is not received)
8.   send “Permitted New Supernode” message to average-valued-node
9. IF (average-valued-node is alive)
10.   receive “Supernode” message
11. ELSE
12.   IF (flag==1)
13.     p = p-i
14.     flag = 0; i=i+1
15.   ELSE
16.     p = p+i
17.     flag = 1; i=i+1
18.   IF (p<0 or p>length of list)
19.   broadcast “Supernode” message
20. EXIT
Features of Algorithm 2:

- It tries to select the node which is neither highly loaded nor lightly loaded but average loaded node.

- Because,
  - Highly loaded node further increases its load and its performance may degrade,
  - Lightly loaded node will get more load from highly loaded nodes when threshold value is calculated as part of load sharing.

So, average loaded node is selected as supernode

However, this algorithm uses more number of messages than algorithm 1 for its selection process.
Performance Analysis

• Performance is measured by executing in simulation environment called Gridsim Toolkit.
• It allows modelling and simulation of PDC systems, resources, schedulers for design and evaluation of scheduling algorithms.
• Experiment is conducted in heterogeneous distributed system of 50 nodes and 10 users.
Performance of hybrid algorithms

ART = Average Response Time
ARTT = Average Round Trip Time
ACT = Average Completion Time

<table>
<thead>
<tr>
<th>TABLE II.</th>
<th>EMPIRICAL READINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ART (Seconds)</td>
</tr>
<tr>
<td>Hybrid_Random</td>
<td>341.2461</td>
</tr>
<tr>
<td>Hybrid_SAA_A</td>
<td>247.861314</td>
</tr>
<tr>
<td>Hybrid_SSA_B</td>
<td>240.630461</td>
</tr>
<tr>
<td>Decentralized</td>
<td>373.096074</td>
</tr>
</tbody>
</table>

Random is created by considering arbitrary cluster size, arbitrary clusters, static threshold policy.
Performance improvement in hybrid algorithms

**TABLE III.** \textbf{Percentage Performance Improvement of Hybrid Algorithm over Decentralized Algorithm}

<table>
<thead>
<tr>
<th>Hybrid</th>
<th>ART</th>
<th>ARTT</th>
<th>ACT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid_SAA_A</td>
<td>34%</td>
<td>63%</td>
<td>28%</td>
</tr>
<tr>
<td>Hybrid_SSA_B</td>
<td>36%</td>
<td>65%</td>
<td>28%</td>
</tr>
</tbody>
</table>
Conclusion

• The main objective is to reduce response time (communication overhead) which is achieved in hybrid algorithm.

• Also, the major drawback of centralized approach which is scalability is also overcome in this approach.

• Future work includes analyzing the performance under different load scenarios, different cluster configurations and network topologies.
Thank You