In the case of a machine with 256 nodes configuration, there are 16384 component for the shape type and consequentially 16384 Dsubgrids. Supposing a Dgrid with 50x50x50 dimension, the size of the Dsubgrid will be a maximum of a 4x4x4 considering the extra borders. Then if a machine configuration has a great number of nodes with vector processors there is a great number of small Dsubgrids with extra borders and this fact increases overhead. Considering also the time spent on node synchronization, it is not possible to decrease the time processing of the program in proportion to the number of nodes.

6. The Message Passing Implementation

The MP implementation used the same division of Dgrids into Dsubgrids that the DP implementation used. An advantage of the MP method is the possibility to use standard languages like C so that, in the case of the marching cube algorithm, few changes had to be done. The only big change was adding a new routine to divide the Dgrid in Dsubgrids plus a few small changes in the routines to calculate the normals and the polytriangle strip.

The MP implementation ran fast in the nodes, but global performance decreased due to the data exchange time between the controller and the nodes. To minimize this problem, we defined a scheme to divide numerous Dgrids into Dsubgrids in the nodes as a pre-process before the user begins interaction. Then, only the selection of a specific Dgrid and a isovalue need to be passed to the nodes during user interaction. An outline of program execution is described below. The text in normal and italic fonts indicate what happens in the controller-program (cp) and in the node-program (np), respectively.

```plaintext
cp Dgrids are divided in Dsubgrids
np each node receives a set of Dsubgrids, each Dsubgrid related to a specific Dgrid
    cp nodes receive Dsubgrids
    np

** begin loop **
    cp controller sends the isovalue and Dsubgrid index
    np receives the isovalue and Dsubgrid index
    np executes the algorithm for the specified isovalue and Dsubgrid
    np sends the results to the controller
    cp controller receives the results

** end loop **

cp if necessary, it is re-executed the loop for other isovalue and Dsubgrid index
```

The communications between controller and nodes to send the isovalue and the Dsubgrid index, and to receive the results have a relevant impact in the algorithm performance (Tables 2, 3 and 4).

The program performance drops significantly when we consider communications during program execution, so the maximum number of surfaces generated per second are around 6 (Tables 2 and 3). This program performance is not satisfactory.

The communication problem required a precise analyze. Table 4 presents time measures of the total program and of some of its portions, all measured in different executions (that is, Tm_Total was measured in one execution, Tm_March in another and so on). Then:

\[ Tm_{Total} = Tm_{Rece} + Tm_{March} + Tm_{Trans} \]

The Table 4 column with Tm_Rec parameters (time that a node takes to receive two numbers from the controller) shows high values of time and that they are decreasing in a constant proportion.
### Table 2

<table>
<thead>
<tr>
<th>Nodes Number</th>
<th>Dgrid Dimen. (Surf: Sphere)</th>
<th>Dsubgrid Dimension</th>
<th>Tm_March (seconds)</th>
<th>Neubes/Sec</th>
<th>NP*NC/Sec (millions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>50x50x50</td>
<td>15x15x27</td>
<td>0.20</td>
<td>26000</td>
<td>0.83</td>
</tr>
<tr>
<td>32</td>
<td>100x100x100</td>
<td>28x27x52</td>
<td>0.63</td>
<td>57000</td>
<td>1.82</td>
</tr>
<tr>
<td>64</td>
<td>100x100x100</td>
<td>28x27x28</td>
<td>0.40</td>
<td>47385</td>
<td>3.03</td>
</tr>
<tr>
<td>256</td>
<td>100x100x100</td>
<td>15x15x27</td>
<td>0.16</td>
<td>31850</td>
<td>8.15</td>
</tr>
<tr>
<td>64</td>
<td>200x200x200</td>
<td>53x52x53</td>
<td>1.92</td>
<td>71825</td>
<td>4.60</td>
</tr>
<tr>
<td>256</td>
<td>200x200x200</td>
<td>28x28x52</td>
<td>0.68</td>
<td>54675</td>
<td>14.40</td>
</tr>
</tbody>
</table>

- **Tm_March**: Time to execute the marching cube algorithm in a specific node
- **Neubes/Sec**: Number of cubes processed per second
- **NP*NC/Sec**: Total number of cubes processed per second considering all nodes

### Table 3

<table>
<thead>
<tr>
<th>Nodes. Num.</th>
<th>Dgrid Dimens. (Surf: Sphere)</th>
<th>Tm_RMT (seconds)</th>
<th>NC/Sec</th>
<th>NP*NC/Tm_RMT (millions/sec)</th>
<th>SPS_A</th>
<th>SPS_B</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>50x50x50</td>
<td>0.46</td>
<td>9200</td>
<td>0.30</td>
<td>6.6</td>
<td>2.40</td>
</tr>
<tr>
<td>32</td>
<td>100x100x100</td>
<td>2.21</td>
<td>15294</td>
<td>0.49</td>
<td>1.8</td>
<td>0.50</td>
</tr>
<tr>
<td>64</td>
<td>100x100x100</td>
<td>2.04</td>
<td>8600</td>
<td>0.47</td>
<td>3.0</td>
<td>0.50</td>
</tr>
<tr>
<td>256</td>
<td>100x100x100</td>
<td>2.56</td>
<td>1990</td>
<td>0.51</td>
<td>8.1</td>
<td>0.50</td>
</tr>
<tr>
<td>64</td>
<td>200x200x200</td>
<td>22.23</td>
<td>5960</td>
<td>0.38</td>
<td>0.6</td>
<td>0.02</td>
</tr>
<tr>
<td>256</td>
<td>200x200x200</td>
<td>24.86</td>
<td>1440</td>
<td>0.37</td>
<td>1.8</td>
<td>0.02</td>
</tr>
</tbody>
</table>

- **Tm_RMT**: Time that a specific node takes to receive the isovalues and the Dsubgrid index plus the time to execute the marching cube algorithm plus the time to send the results
- **SPS_A**: Surfaces per second, ((NP*NC/Tm_March)/Grid), considering only the processing time of the algorithm
- **SPS_B**: Surfaces per second, ((NP*NC/Tm_RMT)/Grid), considering the processing time of the algorithm plus communication time between controller and nodes

As described below, the controller sends the grid_id (Dsubgrid index) and the _iso_value_ to the nodes using the function `CMMD_bc_from_host()` that broadcasts a specific parameter to all nodes.

```c
CMMD_bc_from_host (&iso_value, sizeof(float)); /* Transmission 1 */
CMMD_bc_from_host (&current_grid_id, sizeof(int)); /* Transmission 2 */
```

To identify the effect of each of these transmissions on computing time, we measured the time spent for each one (Table 5). Then, the analyses in Table 5 of Tm_Rec, Tm_Rec_1 and Tm_Rec_2 shows that the first nodes require around 1 second to receive the two values and that basically all the time is spent in the first parameter reception.

The same time measures were obtained when we inverted the parameters transmission (that is, the Dsubgrid index is transmitted first). Then it is reasonable to suppose that there is some problem related to sequential synchronization in the processors (Table 6).

To confirm that the time problem was really related to synchronization operations, we measured the node reception time with a user-forced synchronization before the communication process. Thus we used the functions `CMMD_sync_host_with_nodes()` and `CMMD_sync_with_host()`, as shown in the code.