The ALMA Common Software (ACS) framework lacks the real-time capabilities to control the antennas’ instrumentation —as has been proved by previous works—, which has lead to relying on real-time tasks, carried out in a non-academic context. Thus, the frame of this work is to provide a new implementation of ACS based in the Container/Component model of the Linux RT-PREEMPT kernel, that meets real-time requirements and that confirms this statement. This work addresses the problem of design and integrate a real-time service for ACS, providing to the framework an implementation such that the control operations over the different instruments could be done within real-time constraints. This implementation is compared with the current time service, showing the difference between the two systems when assimilating them to common scenarios. Also, the new implementation is done following the POSIX specification, ensuring interoperability and portability through different operating systems.

**Linux RT-PREEMPT**

One of the newest solutions in the world of real-time operating systems (RTOS) is the usage of the Linux kernel with the Linux RT-PREEMPT patch. The modification of the kernel provides additional capabilities to the operating system, making it suitable for hard-real-time operations, even from the user-space. Its usage is generally growing up, as at the point that its usage has been studied within the ALMA project [7]. To test the real-time capabilities of the patch, we stressed the machine while running the applications [9] with the following tasks:

- a test on DTrace usage through the Ethernet interface,
- copied core and again a big file to a new one,
- Complied with C application, cleaned it, and compiled it again
- Run the compiled application (Generic Algorithm)
- Run the biggest workload, with a real-time running typical amount.

The table on the left summarizes the results for this setup, showing the differences between the two systems with the experimental values of the realization of the real-time tasks (in µs) with a base unit of 10000.

<table>
<thead>
<tr>
<th>Task Description</th>
<th>Experimental Value (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTrace</td>
<td>20</td>
</tr>
<tr>
<td>Copy core</td>
<td>10</td>
</tr>
<tr>
<td>Compile and clean</td>
<td>10</td>
</tr>
<tr>
<td>Run compiled</td>
<td>20</td>
</tr>
<tr>
<td>Total workload</td>
<td>70</td>
</tr>
</tbody>
</table>

With the obtained results, it is clear that the Linux RT-PREEMPT-patched kernel is superior to the original one regarding to the deterministic on the time of invocation, at least of one order of magnitude.

**Solution**

The proposed solution is based in the Container/Component model. This model extends to support real-time environments. This is done by adding the real-time capabilities to the operating system, making it suitable for hard-real-time operations, even from the user-space. This setup is generally growing up, as it has been studied within the ALMA project [7].

To schedule the different jobs throughout the entire execution, we use a priority based scheduler. This is done by assigning a priority value to each job, which depends on the period of the given task, and with the same priority that is able to deal with different granularities of the two time services. This allows us to plan the timeslots and to assign priorities range from 1 to 100.

For our particular case, we define the minimum and maximum values as follows: minP = 75, maxP = 200, P = 100, and maxP = 5000. These values are in agreement with the results of the ALMA project, and, in general, are useful for a great variety of requirements.

Finally, the node on the figure represents equation 1 with the parameters identified.

The Linux RT-PREEMPT kernel shows a much more predictable behavior than the common vanilla Linux kernel, even when the machine is put under stress of heavy disk operations, network traffic and CPU-intensive applications. This makes it suitable for hard real-time tasks, due to the fact that the real-time jobs always get a minimum priority, and so, even when more tasks are present, and at the same time deals better with the different granularities of the two time services.

Also, the new implementation of the Time Service provides real-time support to the ALMA Common Software, as it maintains the requirements of the CONTROL subsystem. Thanks to its design and the usage of the POSIX specification, our new implementation presents much lower jitter values when stressing the system with several requests. Even more, in lower stressed systems, the new implementation presents much lower jitter values than the original one.

In contrast, the harmonic clients with the new implementation shows even better results than those of the ACS Time Service, while the bus was not intentionally driven by our implementation. Once again, the harmonic clients, which represent the worst case scenario for the ACS Time Service, show much better results than those of the original ACS Time Service.

Conclusions

The Linux RT-PREEMPT kernel provides a much more predictable behavior than the common vanilla Linux kernel, even when the machine is put under stress of heavy disk operations, network traffic and CPU-intensive applications. This makes it suitable for hard-real-time tasks, due to the fact that the real-time jobs always get a minimum priority, and so, even when more tasks are present, and at the same time deals better with the different granularities of the two time services.

Also, the new implementation of the Time Service provides real-time support to the ALMA Common Software, as it maintains the requirements of the CONTROL subsystem. Thanks to its design and the usage of the POSIX specification, our new implementation presents much lower jitter values when stressing the system with several requests. Even more, in lower stressed systems, the new implementation presents much lower jitter values than the original one.

In contrast, the harmonic clients with the new implementation shows even better results than those of the ACS Time Service, while the bus was not intentionally driven by our implementation. Once again, the harmonic clients, which represent the worst case scenario for the ACS Time Service, show much better results than those of the original ACS Time Service.