Links:

1. <https://www.linuxjournal.com/article/4444>
2. <http://lars.mec.ua.pt/public/LAR%20Projects/Humanoid/2008_MauroRodrigues/Material%20Adicional/Sistemas%20Operativos/Docs/rtl_vs_rtai.html>
3. <http://mil-embedded.com/guest-blogs/real-hardware-real-software-real-time-performance-real-time-linux/>
4. <https://www.rtai.org/?About_RTAI>
5. <https://www.fsmlabs.com/pdfs/ADITYA_Power_System.pdf>
6. <https://www.windriver.com/products/product-overviews/WR-Linux-Product-Overview/>
7. <https://knowledge.windriver.com/en-us/000_Products/000/010/060/010/050/000_Wind_River_Linux_Open_Virtualization_Features_Guide%2C_LTS_18/000/050>
8. <https://opensourceforu.com/2018/05/a-quantitative-analysis-of-the-real-time-capabilities-of-linux-with-preempt_rt/>
9. <https://www.mepits.com/tutorial/444/embedded-system/vxworks-and-rtlinux-rtos>

RTLINUX

PERFORMANCE PARAMETER:

The purpose of this work is mostly to present a quantitative analysis of RTLinux OS.In order to draw conclusions about the performance, parameters have to be defined which are important in a realtime environment. Basically, this paper deals with external interrupt latencies and scheduling latencies. Other parameters which can give information about the quality of real time capabilities are for example context switch times, process dispatch latencies or the time that is needed to allocate a certain amount of memory.

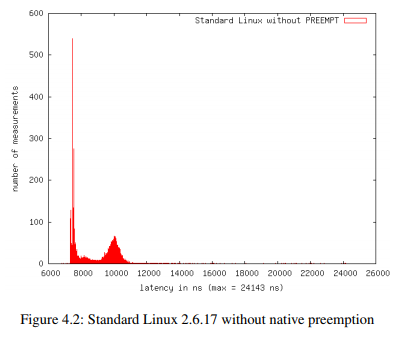
PARAMETERS:

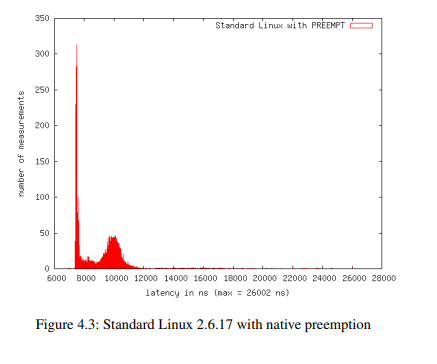
1. Interrupt Latency :

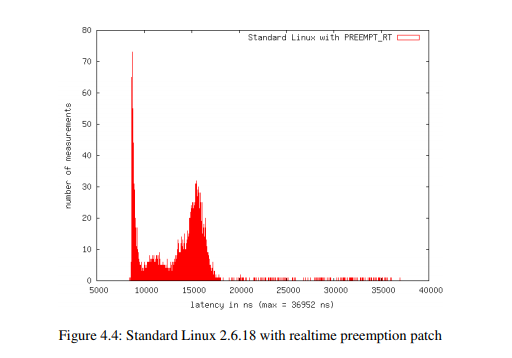
In general, Interrupt latency, also called interrupt response time, is the length of time that it takes for a computer [interrupt](https://whatis.techtarget.com/definition/interrupt) to be acted on after it has been generated. In most computers, a trade-off exists among interrupt latency, [throughput](https://searchnetworking.techtarget.com/definition/throughput), and processor utilization.

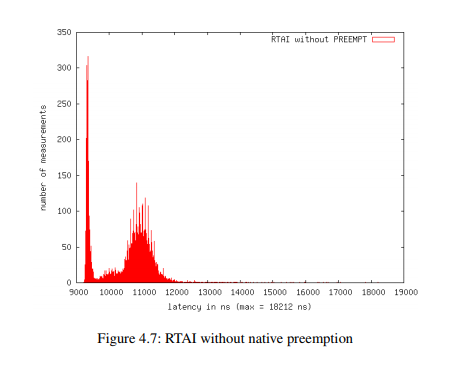
The measurement of the response time to external interrupts gives a good idea about realtime capabilities. The time between the occurrence of an external signal and the start of the corresponding interrupt service routine (ISR) is measured.

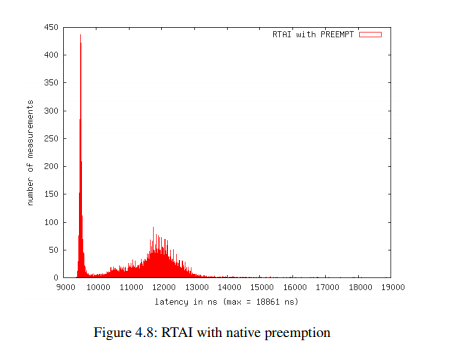
In a first cycle, latencies were measured over a standard Linux 2.6.17 kernel without any preemption switched on. Afterwards, "CONFIG\_PREEMPT" was enabled in order to evaluate the improvements of preemptibility of the Linux kernel. All measurements were made according to an interrupt frequency of 1ms and under heavy stress, by means of I/O- and CPU-load. The results are illustrated throughout the histograms 4.2 to 4.4.





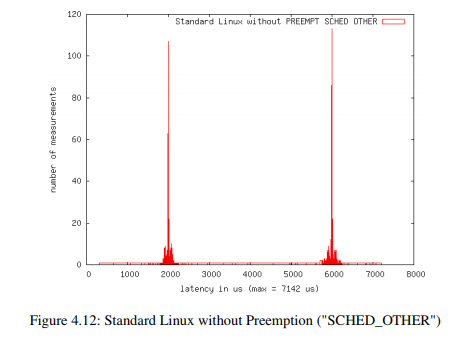


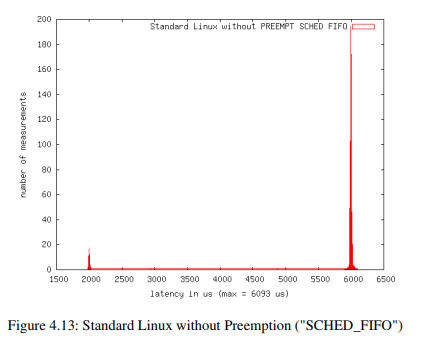


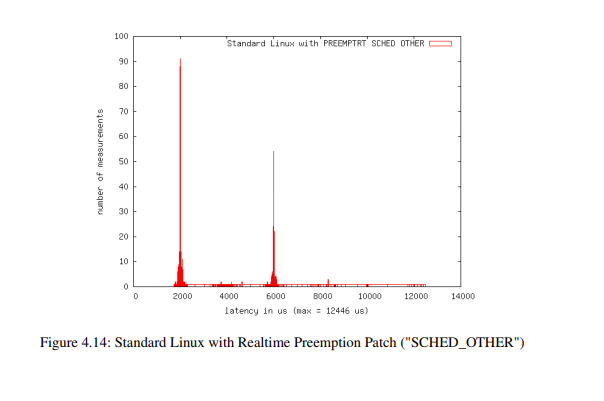


2. SCHEDULING LATENCY: In general, **Scheduling latency** refers to that time from when a task is ready to run to when it is actually gets CPU time.

In a second step, a periodic task was set up and the jitter of activation times has been monitored. The standard Linux kernel was compiled with the standard timer accuracy of 250 Hz (4ms). However, the resolution is quite coarse grained for realtime applications. That’s why the period of the task was adjusted to 10ms. The measurements have been done on a clean vanilla Linux kernel and on a kernel after the application of the Realtime Preemption Patches, by Ingo Molnar. Furthermore, the periodic job was started once with "SCHED\_FIFO" and once with "SCHED\_OTHER" policy.

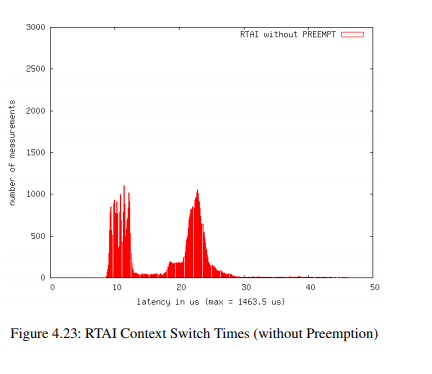


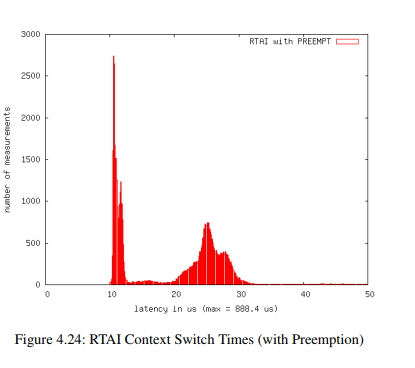




3. CONTEXT SWITCH TIME: A *Context switch* is the time spent between two processes (i.e., bringing a waiting process into execution and sending an executing process into waiting state). This happens in multitasking.The operating system must bring the state information if waiting process into memory and save the state information of the currently running process.

RTAI can benefit from the improvements of native preemptibility in the Linux kernel by means of "CONFIG\_PREEMPT" and "CONFIG\_PREEMPT\_VOLUNTARY". If Linux kernel functions are used, a realtime task is transparently migrated from hard realtime to soft realtime context. Soft realtime means that the task is now under the control of the Linux scheduler. After the service delivery, the task will be migrated back into hard realtime context, as soon as possible. In order to measure the improvements of native preemptibility of the Linux kernel, a periodic task (1 ms) was set up, which switches between RTAI and Linux context, by doing a Linux system call (gettimeofday). The time for this mode switch was measured once with a RTAI kernel with "CONFIG\_PREEMPT" enabled and once without preemption. For clarification reasons, the x-coordinate in these histograms is limited to 50µs, although the worst case latency is much higher.



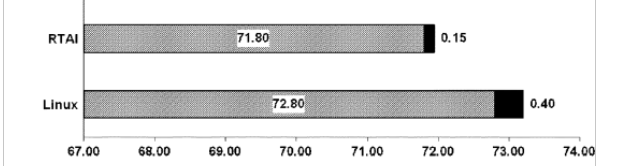


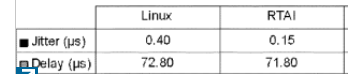
RTAI performance parameters;

In this section we will discuss performance parameters of RTAI which will cover Interrupt Latency, Scheduling Latency, Context Switch time.

1. INTERRUPT LATENCY

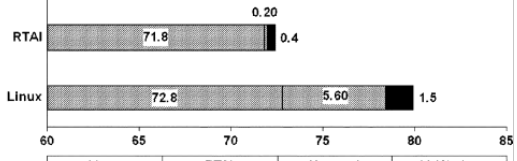
RTAI rely on the Linux IP stack for network communication,RTAI bypasses ADEOS in the management of interrupts, and relies on ADEOS only in the case the interrupt has to be propagated to another domain (Linux). It is worth noting that the performance of RTAI is very close to that of RTlinx and that, remarkably, the jitter appears to be smaller.

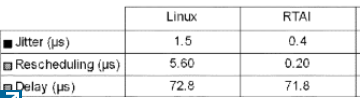




(measured delay and jitter without rescheduling )

In second test, we consider program acqloop, so that the output is not written directly by the Interrupt Service Routine, but by another kernel task which is waiting for a semaphore, set by the ISR. In this case we take into account also the rescheduling task time.

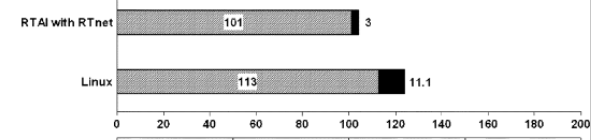


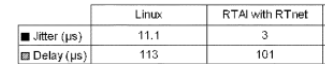


(Measured delay and jitter including rescheduling)

In particular it appears that RTAI has a very efficient scheduler for native real-time tasks.

If we consider network communication.In this case the process waiting for the semaphore does not write the output directly to the DAC, but sends a UDP packet with all acquired data samples to another MVME5500 board, which in turn writes to the DAC converter.





(Measured delays with real-time network communication.)

Here we compare the performance of Linux,using the native IP stack, RTAI with RTnet, since RTnet shows very good performance with little jitter.

2. SCHEDULING LATENCY:

Latencies

To evaluate scheduling performance, we used the la-

tency measurement module of RTAI and adopted it

slightly to ﬁt both environments. This was necessary,

because in the original code, a user-space process

reads the measurement data from a FIFO and logs

them. In our version, measurement data is simply

output by the kernel module via rt printk() instead.

The module sets up a task with a period of 100 µs

which calculates the diﬀerence between the expected

and the true activation time. Each measurement col-

lected 250.000 data values. The RTAI/Linux system

was stressed by CPU and I/O load by using cpuburn

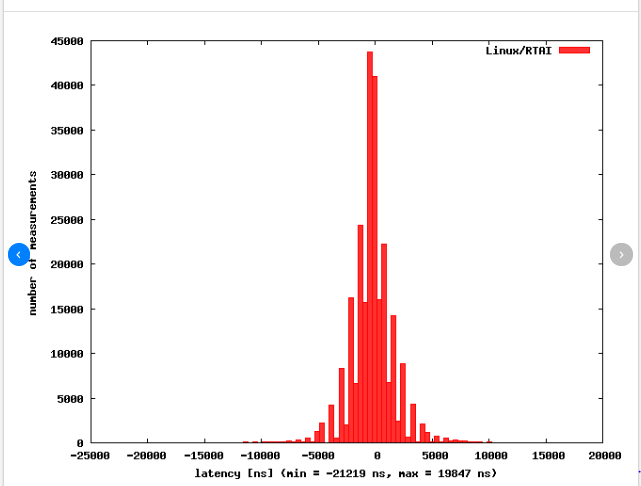
and ﬂood pinging the system. Due to the lack of

driver support, only CPU load could be generated

for LRTAI. This was achieved by ﬁnalizing system

initialization with an endless loop instead of a call to

cpu idle().



`

CONCLUSION/ DISCUSSION/OUTLOOK:

The aim of the reported work has been the performance evaluation of Real Time Linux and RTAI( Real Time Application Interface), for this evaluation we have started with Basics of both RtLinux and RTAI, in this basic discussion we have included diagrams for working environment for both RTAI and RTLinux, after discussion on Basics of RTLinux and RTAI, we have stated the performance analysis of RTLinux and RTAI, for performance analysis we have considered mainly 2 performance parameters that are 1) interrupt latency and 2) scheduling latency for both RTLinx and RTAI, moreover for RTLinux we have studied one more parameter i.e. Context Switching Time, in this performance analysis we have included all the possible areas in which both the system can works in optimal state, also we have included various statistics which are represented by various graphs which states an crystal clear difference between the performance of RTLinux and RTAI in context to interrupt latency and Scheduling Latency.

REFERENCES:

Include 2 papers which i am sending in mail.

THANK YOUUUUUUUUUUUUUU

NOTE: also add working environment diagrams from papers for both……..