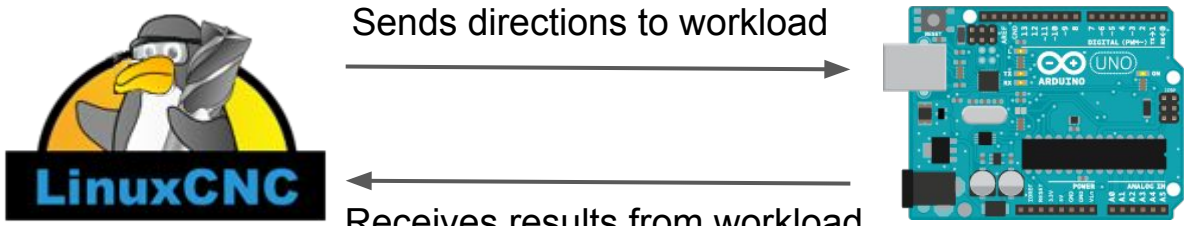
LinuxCNC Real-Time Evaluation

Stephen Decker, Jeremy Tang, Ethan Glassman

Project Recap

- We intend to evaluate real-time performance of the LinuxCNC operating system environment on a laptop.
- We intend to use an Arduino emulating a CNC mill as a workload.
- In this way we can test the real-time performance of the kernel while performing I/O operations without needing a large, complex, and dangerous real load.



Debian Linux Distribution With Real-Time Kernel

Microcontroller

Linux CNC vs Vanilla Linux

- Real-time kernel exists as a layer between hardware and normal kernel
- Hardware interrupts handled by Real-Time kernel
- Normal kernel sees hardware interrupts as software interrupts
- User services given highest priority, normal kernel services given lowest
- Other kernel config modifications (high resolution timers, no CPU frequency scaling)
- LinuxCNC also contains Hardware Abstraction Layer (HAL)
- Allows configuration with many host setups without recompiling kernel
- Achieved by virtualizing expected hardware
- Generally, LinuxCNC does not necessarily replace but rather supplements

What have we tried?

- Writing a real-time kernel module for Linux.
  - The goal was to communicate with an Arduino over serial in real-time, and then measure real-time performance.
  - Had difficulties testing it because USB is terrible
  - Also we crashed the kernel a lot.
  - Seriously a lot.
  - Although the end-goal didn’t pan out, we did still write a real-time kernel module.
- Writing a workload for the Arduino
  - Wrote a simple serial loopback program.
  - Did not progress further due to complications with the laptop side of things.

...what else have we tried?

- Installed RT-Preempt on a Raspberry Pi, getting the latency down to <60 microseconds.
- Thought about designing a real-time Ethernet workload between Raspberry Pi & laptop.
  - RTNet - real-time Ethernet driver.
  - RTNet requires RTAI, which is not available for Raspberry Pi.
  - RTNet is old and not very well supported.
- Thought about using 2x Raspberry Pi allowing us to work with GPIO pins
  - Send analog (high/low voltage signals) instead of being limited by the digital protocols
  - No real time GPIO libraries we could find
    - Someone did start setting up their raspberry pi with Node.js for realtime though?

Stretch Objectives

- Real-time performance comparison of different interfaces (Ethernet, USB)
  - Ethernet almost worked, RTNet hasn’t been updated in years
  - USB is slow and there is no (we found) to access USB serial port from the real time kernel
- Measure scheduling latency of the Linux environment that LinuxCNC uses (Hourglass, Cyclictest, hooking into LinuxCNC/arduino, other benchmarks)
- Determining specific circumstances when LinuxCNC does not maintain real time scheduling
  - Using USB wrecks real time performance
- Create recommended steps for how LinuxCNC can be modified to maintain real time performance
  - Don’t use USB to connect to your machine
  - Don’t use a laptop

Why we had difficulties with USB

- Arduino uses “virtual” serial port disguised as USB
  - Thus, linux kernel sees a usb instead of serial port
- RTAPI is able to write and read byte from serial port
  - However, we need the actual address in memory of serial port
  - Very unclear how to find this actual address for USB with normal linux USB driver
- Possible solution is to write custom USB wrapper to interface with RTAPI
  - Could be a complicated project and a lot of time
  - LinuxCNC people tried and never finished
  - Even with RTAPI wrapper, USB performance could (probably) be poor so not worthwhile

Linux CNC vs Vanilla Linux

- Real-time kernel exists as a layer between hardware and normal kernel
- Hardware interrupts handled by Real-Time kernel
- Normal kernel sees hardware interrupts as software interrupts
- User services given highest priority, normal kernel services given lowest ←
- Other kernel config modifications (high resolution timers, no CPU frequency scaling) ←
- LinuxCNC also contains Hardware Abstraction Layer (HAL)
- Allows configuration with many host setups without recompiling kernel ←
- Achieved by virtualizing expected hardware
- Generally, LinuxCNC does not necessarily replace but rather supplements

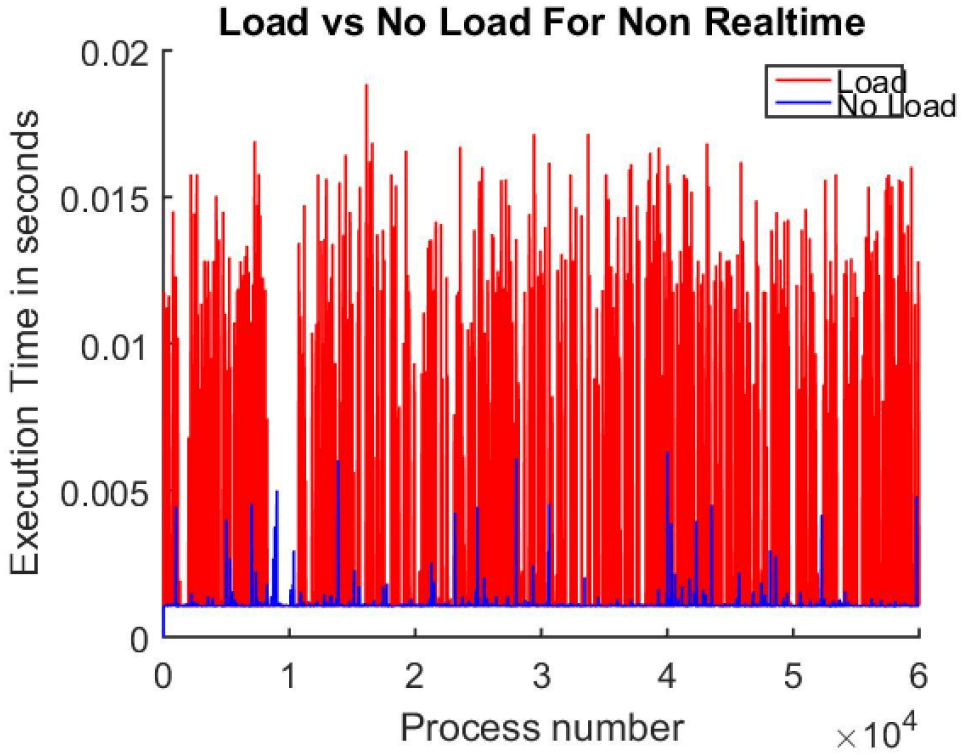
Our Real-Time Kernel Module

- Used RTAPI library provided by Linux CNC
  - Can compile for real-time or normal execution based on flags
- Based on examples provided by Linux CNC
  - Lots of difficulty with compiling examples (last time Linux CNC automatically compiled examples was ~2006)
  - Complicated Makefile difficulties due to complex project
  - Trouble loading the actual real-time kernel module
- Final result should tries to execute every millisecond (just outputting a simple kernel module message and current time in nanoseconds)
- Compare real-time vs non-real-time timing performance.
- Learned a lot about compiling both normal and real-time kernel modules

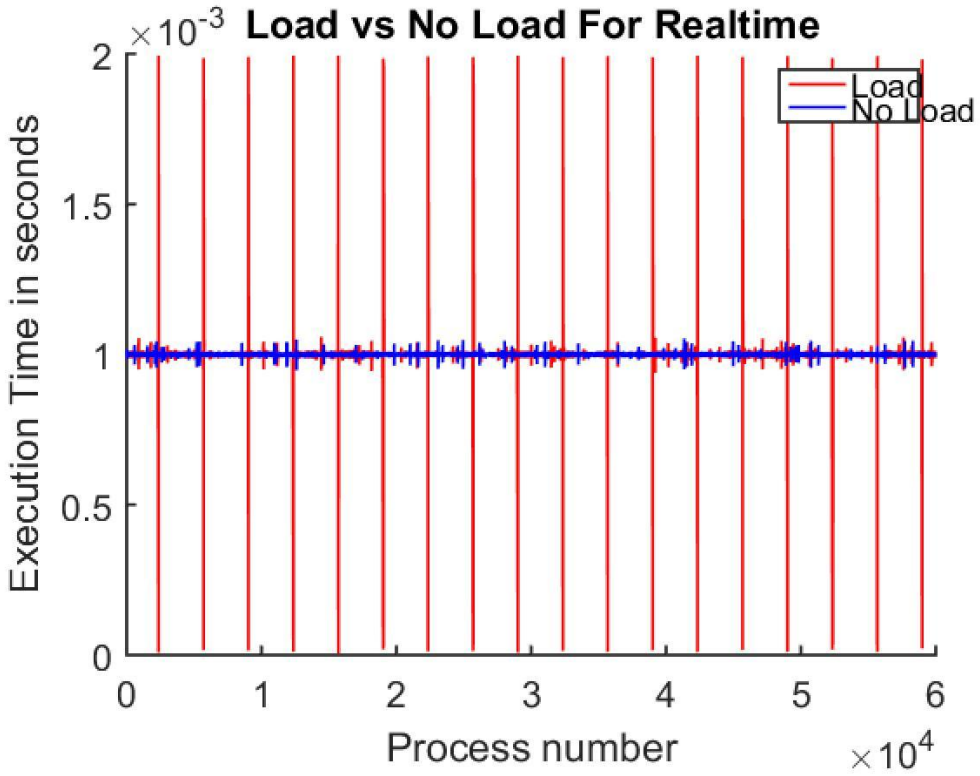
Plots

- Sample line of data:
  - `Date  Time  Program Name  Timestamp  Message`
  - `Nov 23  23:16:28  cse520linux  kernel:  [307.854725]  Hello`
- Plot:
  - realtime__time_1 = xlsread('Results_Load_NoLoad.xlsx',’D5:D60000’)
  - realtime_time_diffs_1 = abs(diff(realtime_time_1)
  - plot(realtime_time_diffs_1)

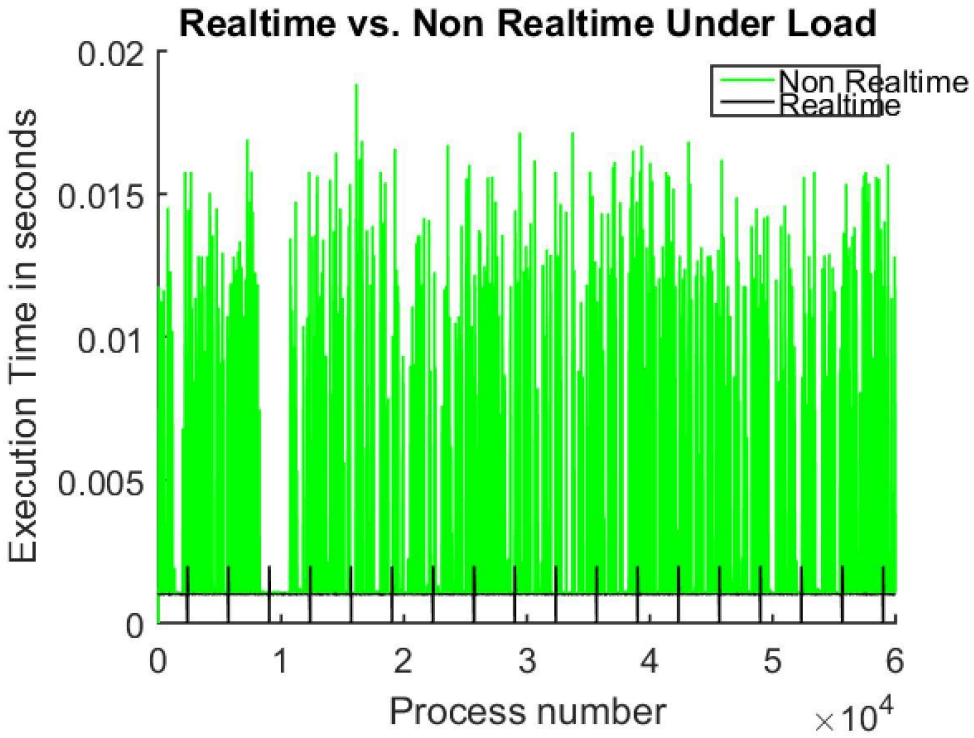
Non Realtime - Load vs. No Load



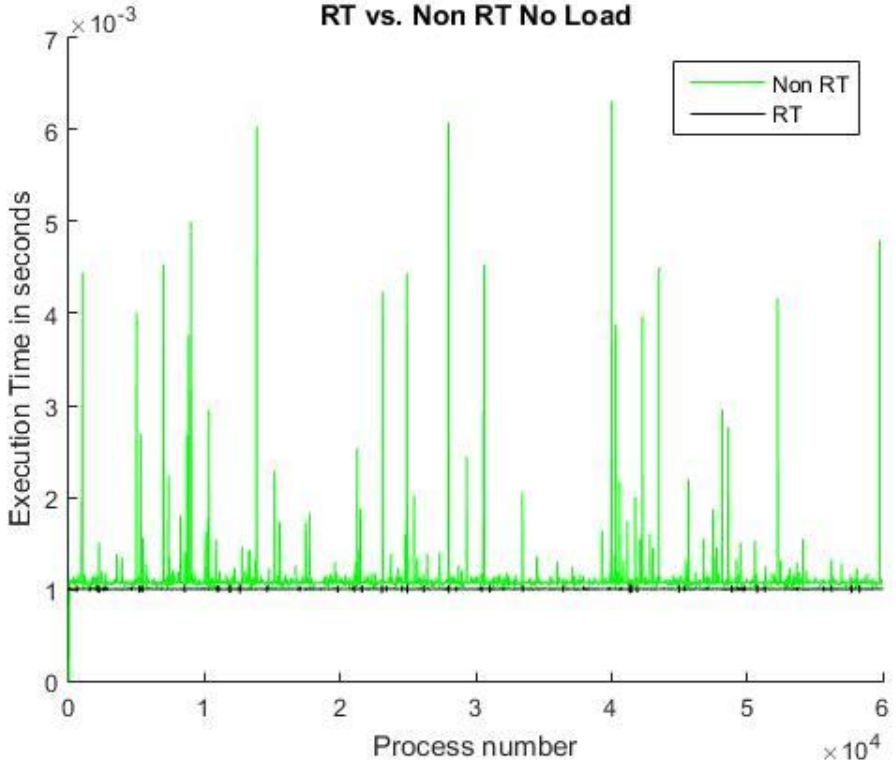
Realtime - Load vs. No Load



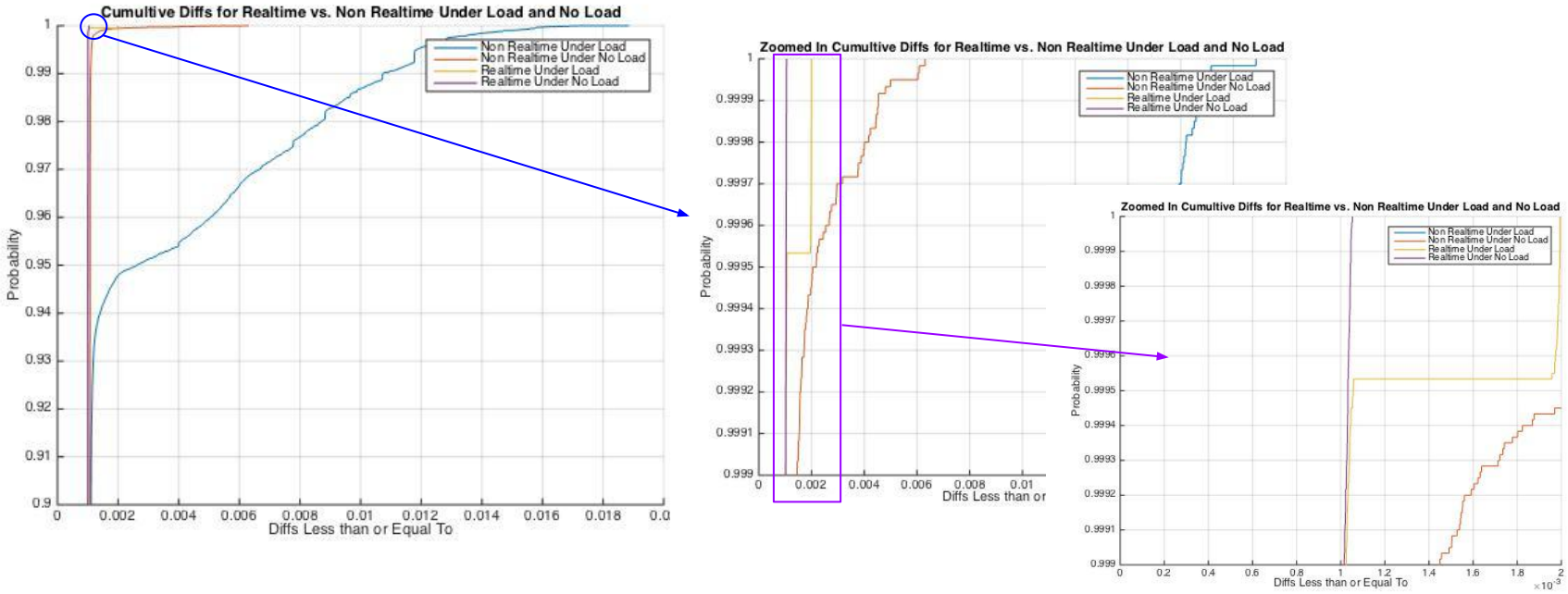
Non Realtime vs. Realtime Under Load



Non Realtime vs. Realtime Under No Load

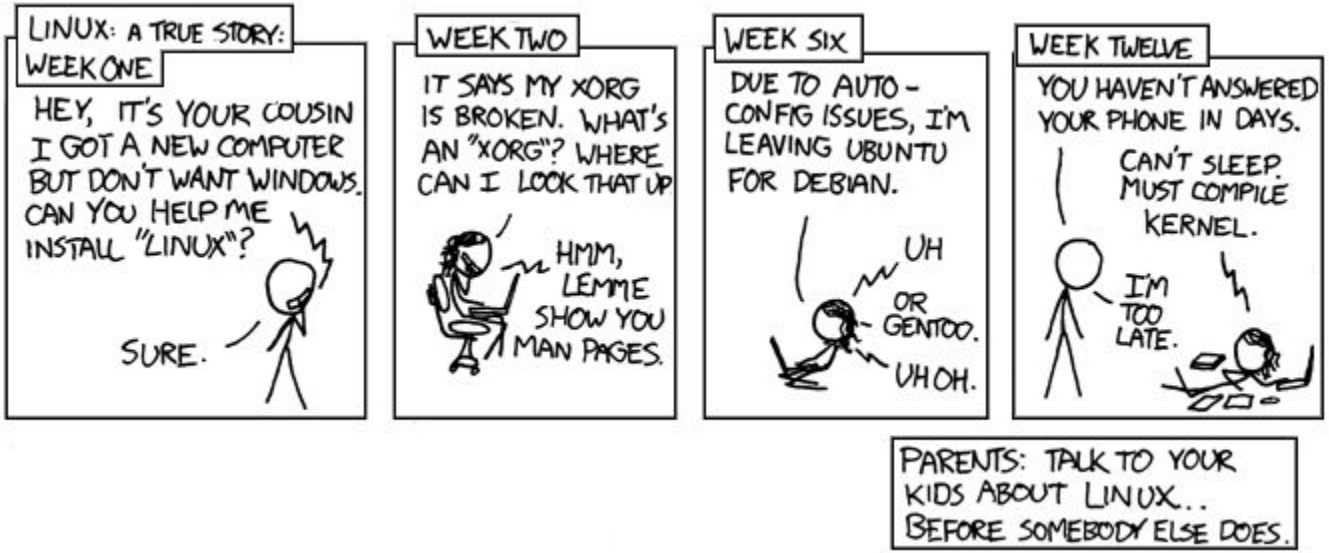


Cumulative Diffs



Conclusions

- Do not use a laptop for LinuxCNC, unless it has a serial port.
- Do not use USB for anything real-time, it’s not worth it.



Questions?