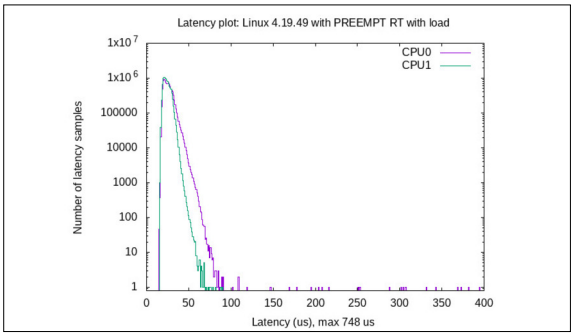
Patrick
Elsener

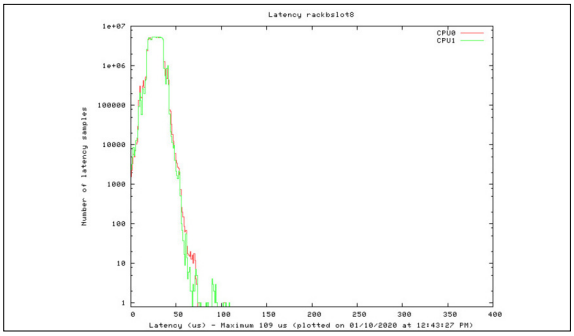
Student	Patrick Elsener
Examinator	Prof. Reto Bonderer
Themengebiet	Software and Systems

Embedded Linux

Konfiguration mit dem Yocto Projekt



Cyclictest Latenz Histogramm: Linux 4.19.49 mit PREEMPT_RT mit Load
Eigene Darstellung



OSADL Cyclictest Latenz Histogramm: ARMv7 Processor rev 0 (v7l) (ARM Xilinx Zync 2x666 MHz), Linux
OSADL, Latency plot of system in rack b, slot 8

Einleitung: Heutzutage gibt es eine grosse Auswahl an Mikroprozessoren und Mikrocontrollern, welche viel Leistung für relativ wenig Geld bieten. Zusätzliche Eigenschaften wie mehrere Rechenkerne, eine MMU führen zu einem Mehraufwand in der Softwareentwicklung. Daher wird heutzutage meistens ein Betriebssystem eingesetzt, welches sich um den Umgang mit diesen Eigenschaften kümmert. Betriebssysteme bieten zudem den Vorteil, dass sie Funktionen bereitstellen, die oft von komplexeren Applikationen benötigt werden (beispielsweise Mutexes). Daher ist es nicht erstaunlich, dass im Embedded Systems Bereich vermehrt Betriebssysteme eingesetzt werden, meistens ein Linux-basiertes Betriebssystem (auch bekannt unter dem Begriff Embedded Linux). Diese Linux-basierten Betriebssysteme müssen wegen den knappen Ressourcen aufs Nötigste reduziert und spezifisch auf die verwendete Plattform angepasst werden.

Vorgehen: Zuerst werden die Eigenschaften und der Aufbau eines Embedded Linux sowie dessen Minimalanforderungen untersucht. Mittels dem Buildsystem das Yocto Project wird demonstriert, wie sich ein Embedded Linux für eine bestimmte Hardware-Plattform erstellen lässt. Danach wird aufgezeigt, wie sich mit dem Yocto Project Softwarekomponenten updaten und Treiber ins System integrieren lassen und sich Hardware ansteuern lässt. Zum Schluss wird untersucht, wie sich ein Embedded Linux realtime-fähig machen lässt und was die Grenzen dieser Realtime-Fähigkeit sind. Dafür wird aufgezeigt, wie sich ein Embedded Linux mittels dem Yocto Project und dem Anwenden der PREEMPT_RT Patches realtime-fähig machen lässt. Anschliessend werden zwei Benchmarks durchgeführt, die die Latenzen eines solchen Systems aufzeigen sollen. Einerseits wird der Benchmark Cyclictest durchgeführt und andererseits wird ein eigener Benchmark entworfen, welcher die Latenzen mit Bezug auf I/O misst.

Fazit: Eine Erkenntnis ist, dass die Verwendung eines Embedded Linux viele Vorteile mit sich bringt. Gerade komplexere Embedded System Projekte können von einem Betriebssystem profitieren, da viele der benötigten Funktionen bereits vorhanden sind. Da die Unterschiede zwischen einem Embedded Linux und einem herkömmlichen Linux nur geringfügig sind, sind die meisten Treiber und Libraries auch in einem Embedded Linux verfügbar. Es zeigt sich, dass das Yocto Projekt ein geeignetes Werkzeug zum Erstellen eines Embedded Linux ist. Die Vielzahl von Layern von Drittquellen (unter anderem auch BSPs von Chipherstellern), ermöglichen es, schnell ein Embedded Linux System zusammenzustellen, welches den eigenen Bedürfnissen entspricht. Bezüglich der Realtime-Fähigkeit eines Embedded Linux können mehrere Erkenntnisse gezogen werden. Den Linux Kernel mittels PREEMPT_RT Patches und der Hilfe des Yocto Projects realtime-fähig zu machen, ist gut umsetzbar. Es scheint jedoch so zu sein, dass dies alleine noch nicht genügt, damit das ganze System realtime-fähig ist. Unsere Benchmarks lassen darauf schliessen, dass möglicherweise die Kernel-Konfguration einen Einfluss auf die Latenzen des Systems hat. Zudem ist nicht jedes Programm, welches einen POSIX-Thread zyklisch aufwachen lässt, automatisch in der Lage, die vorgegebenen Zeiten einzuhalten, wenn es auf einem Linux System mit angewendeten PREEMPT_RT Patches läuft.