Linux RT-Preempt Echtzeitkenngrößen ermitteln und testen

Embedded Testing 2017

Thomas Maierhofer Consulting

www.maierhofer.de

Agenda

- Der Begriff Echtzeit
- Echtzeit und Linux
- Echtzeitkenngrößen festlegen
- Echtzeitkenngrößen testen
- Zwei Nützliche Tools

Was ist Echtzeit?

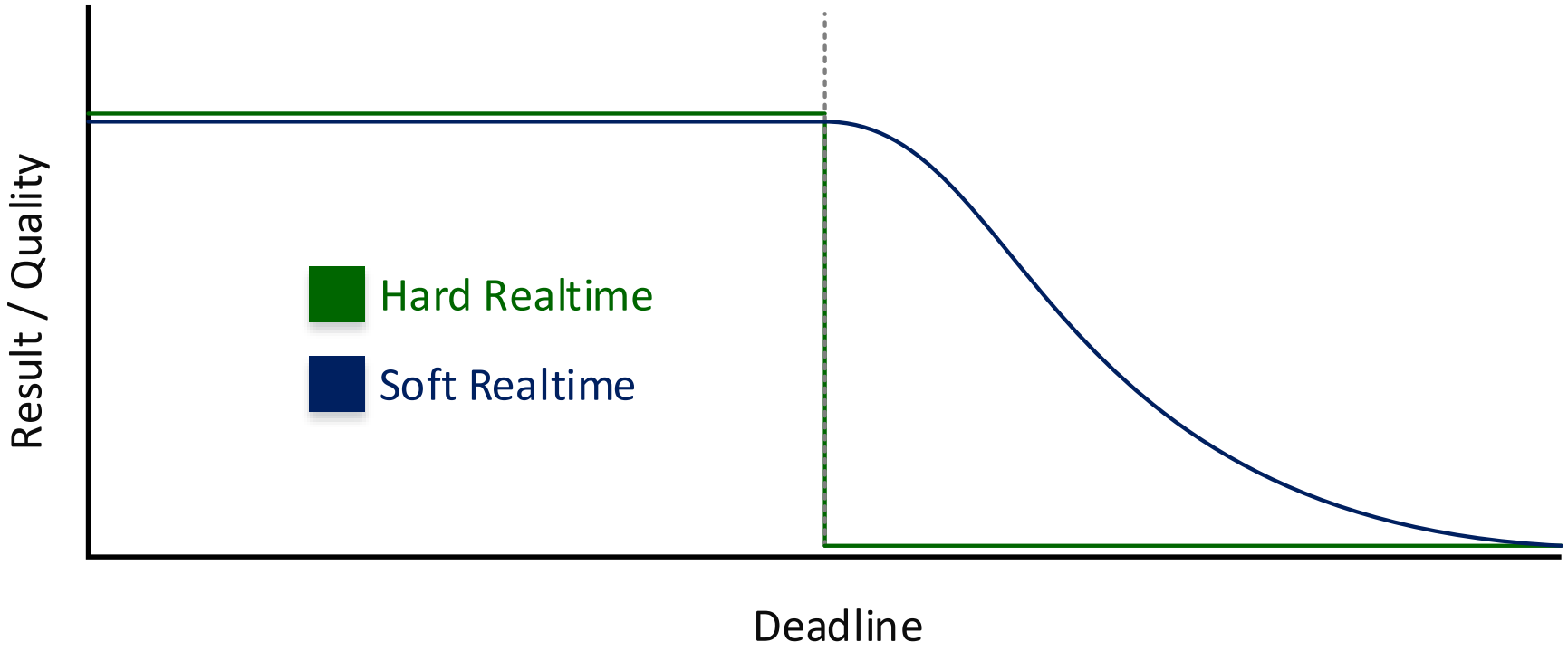
- ~~Schnell / Hoher Durchsatz~~
- Rechtzeitig

  Ergebnisse müssen nicht nur korrekt sondern auch rechtzeitig vorliegen
  - Elektronenstrahlschweißen Magnetfeldregelung < 1 - 10 µs
  - Automatisierung (SPS) < 1 - 10 Millisekunden
  - Computerspiele: < 60 Millisekunden (14-16 Bilder pro Sekunde)
  - Raumtemperaturregelung: < mehrere Sekunden
  - Wettersatelliten Wetterdaten: < mehrere Stunden

Harte / Weiche Echtzeit

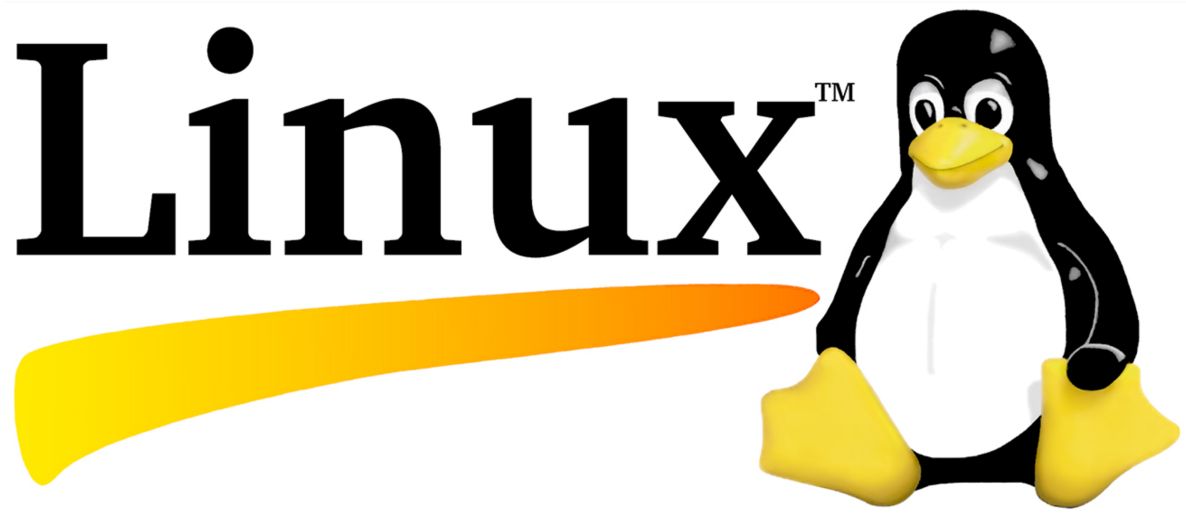
Harte Echtzeit: Resultat nach der Deadline nutzlos

Weiche Echtzeit: Qualität sinkt nach der Deadline



Echtzeit Linux Varianten

- **Echtzeitfähiger Kernel**
  - **RT-Preempt**
- Hypervisor mit Linux und Echtzeit Kernel
  - RT-Linux (Victor Yodaiken/Windriver)
  - RTAI (Paolo Mantegazza)
  - Xenomai (RTAI Fork)
  - …



RT-Preempt Architekturen

- Ein Kernel - eine API
- Viele Komplexe Dienste ohne harte Echtzeitanforderungen
  - HMI – heute oft HTML5/CSS3/JS
  - Netzwerk APIs (z.B. OPC/UA, SCADA Systeme)
  - Backup/Restore, Service Techniker Schnittstelle
- Relativ moderate Anforderungen an die Deadlines
  - Latenzen von < 50 µs mit gängigen ARM A7/A8 Prozessorarchitekturen
  - Zyklen >= 250 µs realisierbar
  - Hoher Scheduling Overhead bei sehr kurzen Zyklen
  - Ungeeignet für Zyklen < 100 µs

SCHED_NORMAL

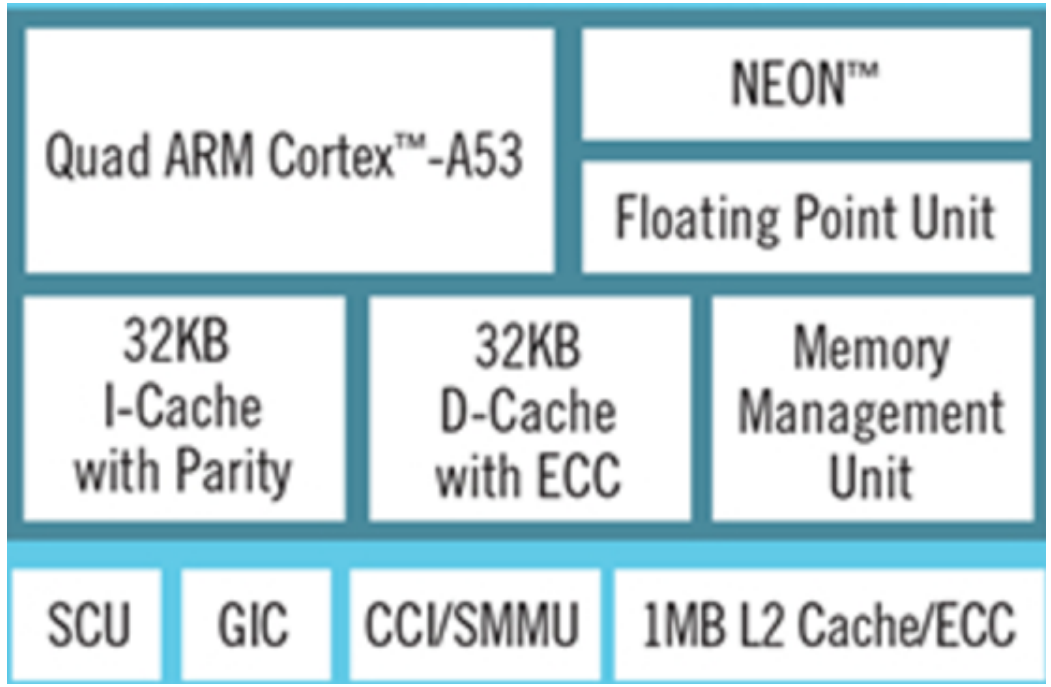
Linux RT-Preempt Schedulers

```
( ) Preemptible Kernel (Low-Latency Desktop)
( ) Preemptible Kernel (Basic RT)
(X) Fully Preemptible Kernel (RT)
```

- Completely Fair Scheduler (SCHED_NORMAL)
  - Keine Echtzeit / Priorisierung
- Realtime First In - First Out (SCHED_FIFO)
  - Echtzeit / Priorisierung
- Realtime Round Robin (SCHED_RR)
  - Echtzeit / Priorisierung
- Realtime Deadline (SCHED_DEADLINE)
  - Echtzeit / Earliest Deadline First

Echtzeit mit „Desktop Prozessoren“?

- Superskalare Prozessoren
  - Mehrere Ausführungseinheiten
  - Pipelines und Pipeline Reordering Out of Order Execution
  - Branch Prediction
  - Mehrschichtige Caches
- Mehrkernprozessoren
- Virtuelle Speicherverwaltung



- Dedizierte Realtime Prozessoren meist ungeeignet für Linux
  - Hauptsächlich wegen fehlender MMU

Echtzeit Prozesse

- Betriebssystem
  - RT-Preempt Gepatchter Kernel
  - Latenzen gemessen und optimert (CYCLICTEST)
  - …
- Echtzeitprozess
  - Echtzeit Scheduler gewählt (SCHED_FIFO, …)
  - Kein virtueller Speicher (Pinned Memory)
  - Stacks für jeden Thread alloziert (Stack Prefaulting)
  - …

Problem: Echtzeitkenngrößen

- Mehrere Echtzeitprozesse mit mehreren Echtzeit Threads möglich
- Echtzeitverletzung wird vom Betriebssystem nicht erkannt
  - Kein vorgegebenes Taskmodell
  - Mehrere Scheduler / Scheduler Algorithmen
  - Interrupthandler stellen eigene RT - Threads dar

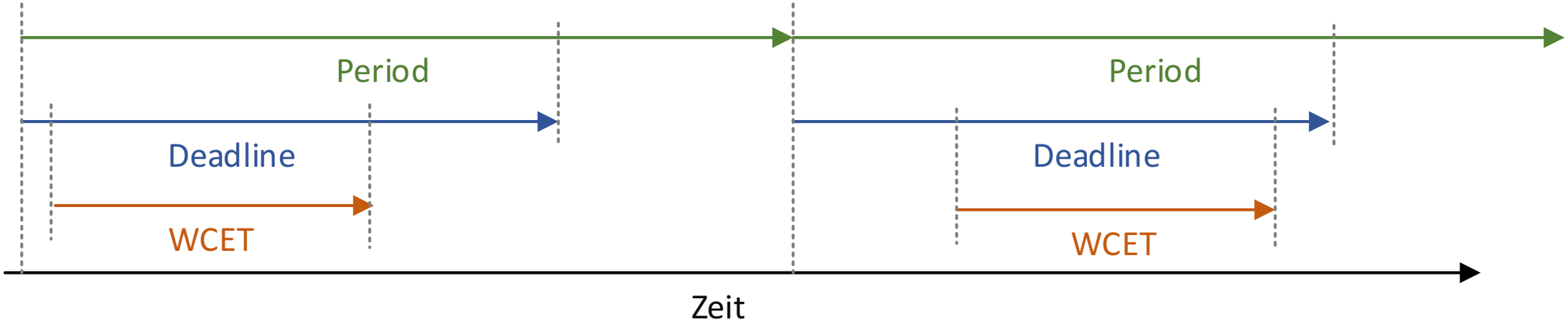
RT-Preempt Linux hat von sich aus keine definierten Echtzeitkenngrößen und somit auch keine Schnittstelle um diese zu messen.

Echtzeit Scheduling Algorithmen

- Rate Monotonic (RMS)
  - Feste Prioritäten / Kürzere Periodendauern sind prior
  - Auslastumg bis zu 69% der CPU Zeit (Liu/Layland)
- Earliest Deadline First (EDF)
  - Variable Prioritäten / Kürzere Deadlines sind prior
  - Auslastung bis zu 100% der CPU Zeit (Liu/Layland)
- … viele weitere

Sporadic Task Model

- Period: Kleinster Zeitlicher Abstand zwischen der erneuten Ausfürung
- Deadline: Zeitpunkt bei dem die Aufgabe beendet sein muss.
- Worst Case Execution Time (WCET)



Kenngrößen der Echtzeit Tasks

Periode

- Vorgegeben – z.B. Feldbuszyklus oder Regelschleife
- Aber auch minimaler zeitlicher Abstand zweier Events

Deadline

- Vorgegeben - Meist gleich der Periode, manchmal auch kleiner um z.B. den nächsten Feldbuszyklus sicher zu treffen.

Worst Case Execution Time

- Ändert sich während der Entwicklung permanent, da sowohl Code, Hardware und auch die konkreten Abläufe darauf Einfluss haben

Echtzeitkenngrößen festlegen

- Durch Architektur und Design
  - Scheduling Algorithmen
  - Task Modell
  - Eventsteuerung und Messaging
  - Detektion von Echtzeitverletzungen
  - Instrumentationsschnittstelle

SCHED_NORMAL

Gestaltung der Echtzeit Tasks

Eventbasiertes System

- Scheduler SCHED_FIFO oder SCHED_RR
- Events und Semaphore zur Synchronisation

Zyklisches System

- Scheduler SCHED_FIFO, SCHED_RR oder SCHED_DEADLINE

Beachte: Durch die Gestaltung der Tasks wird ein Scheduling Algorithmus implementiert, der bestimmte Grenzen hat. Z.B. RMS hat eine Auslastungsgrenze von 69%

Echtzeittests unter Linux

RT-Preempt Linux ist ein vollwertiges Linux Betriebssystem

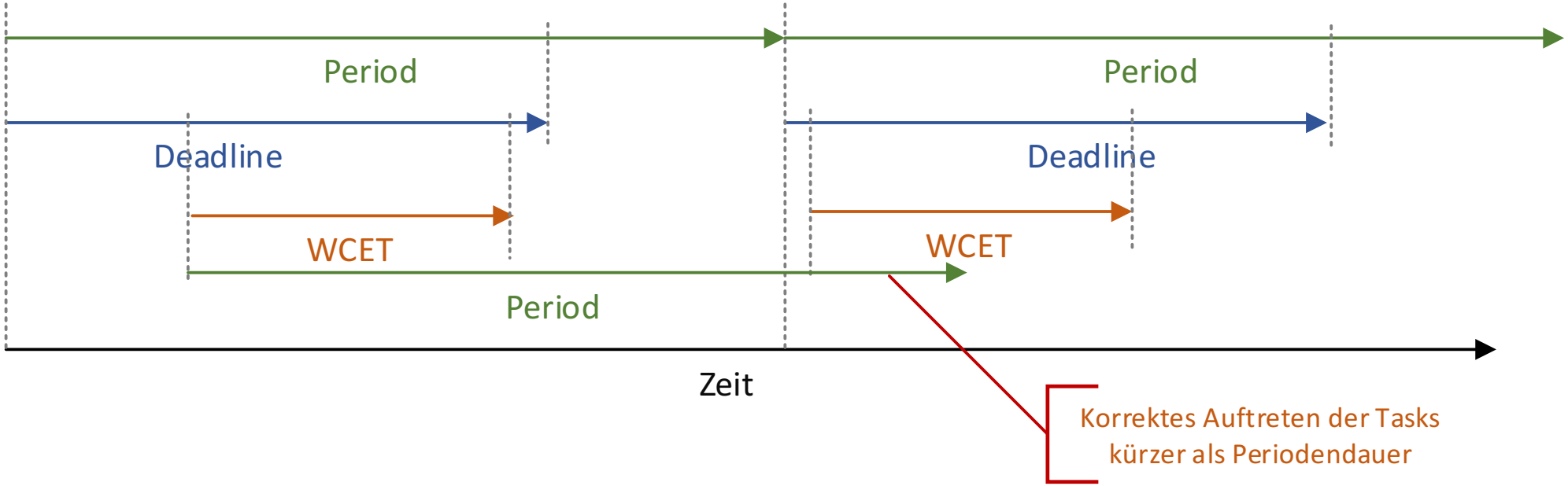
- Netzwerkschnittstellen / Netzwerkdienste
- Benutzerverwaltung und Remotezugriff (SCP/SSH)
- Hintergrundprozesse (systemd)
- …

All diese Features können potentiell Tests verfälschen oder im Betrieb Probleme mit der Echtzeit bereiten.

Testen auf Periodendauerunterschreitung

Bei Ereignisgesteuerten Systemen: z.B. durch den Timestamp des **Auslösers** des vorherigen Tasks

Wichtig: Nicht im Task selber messen! Siehe:



Testen auf Deadline Überschreitung

Bei Ereignisgesteuerten Systemen:
Durch den Timestamp des **Auslösers** des vorherigen Tasks und dem Ende des aktuellen Tasks

Bei Zeitgesteuerten Systemen:
Durch das korrekte Ende des aktuellen Tasks

Merke: Das Detektieren der Deadline Überschreitung muss IMMER in einem eigenen Thread durchgeführt werden.

Wie soll die WCET bestimmt werden?

- Latenzen durch Interrupts / Preemption / Cache Misses
  Pipeline Misses
- Speziell Tasks mit wenigen Instruktionen können in ungünstigen Konstellationen um Faktoren langsamer ausgeführt werden
- Die konkrete Ausführungsreihenfolge ALLER Threads wirkt sich aus.
- Zählen und aufaddieren der Befehlszyklen nicht möglich
  Gegensatz: Unterbrechungsfreie Mikrokontroller Architektur

Methoden der WCET Analyse

- Statische Zeitanalyse / Abstrakte Interpretation
  - Starke Garantien
  - Steuerungen und Steuergeräte mit Sicherheitsfunktionen
  - Für RT-Preempt Linux praktisch nicht durchführbar
- Worst Case Observed Execution Time
  - Für beliebige Systeme durchführbar
  - Keine Garantien
  - Ungeeignet für Steuerungen mit Sicherheitsfunktionen

Worst Case Observed Execution Time?

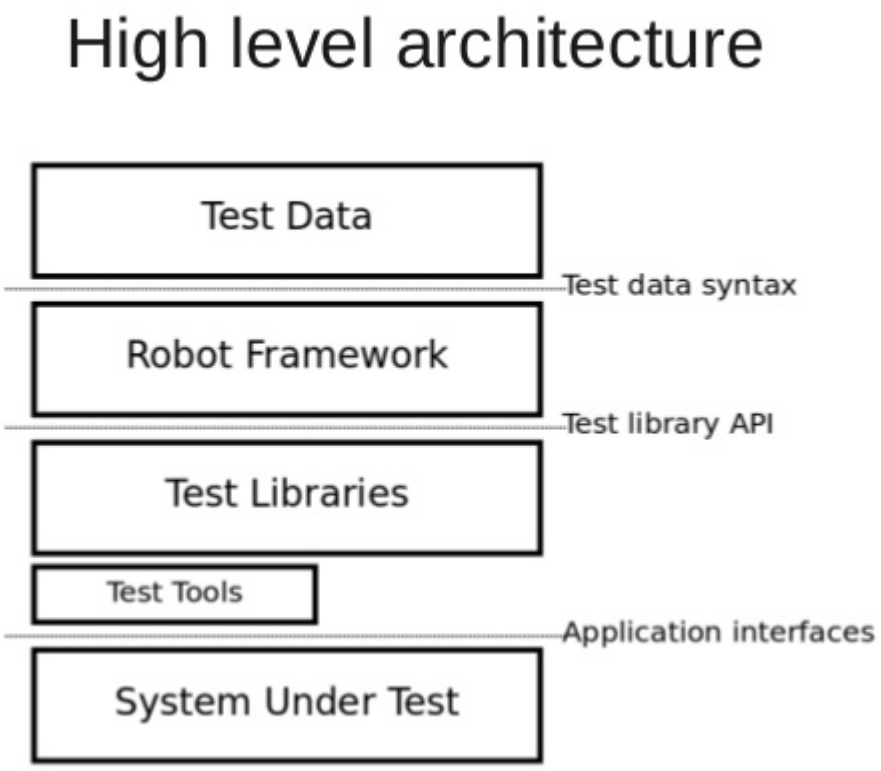
- Vollständiger Echtzeit Kontext wie im Betrieb
- Berücksichtigung der Last weiterer RT Prozesse
- Synthetisch erzeugte maximale Last an Interrupts/DMA z.B. mit „cyclictest“
- Stresstest durch Überlastung der nicht RT Dienste

Testen

- Mit dem Zielsystem testen – wie später ausgerollt
- Testsystem isolieren
  - Eigenes Netzwerk
- Keine zusätzliche Software auf dem Prüfling
  - Insbesondere kein Jenkins Agent / JRE
- Testsoftware und Testdaten mit SSH/SCP/Telnet/FTP übertragen
  - Sofern mit dem Zielsystem ausgerollt.
  - Wenn nichts anderes geht: SD Karten Switch, JTAG …

Testautomatisierung

- Jenkins Agent
- Robot Framework
  - Ähnlich Cucumber / Fitnesse
  - Gherkin Syntax
  - Verfügbar für viele Sprachen

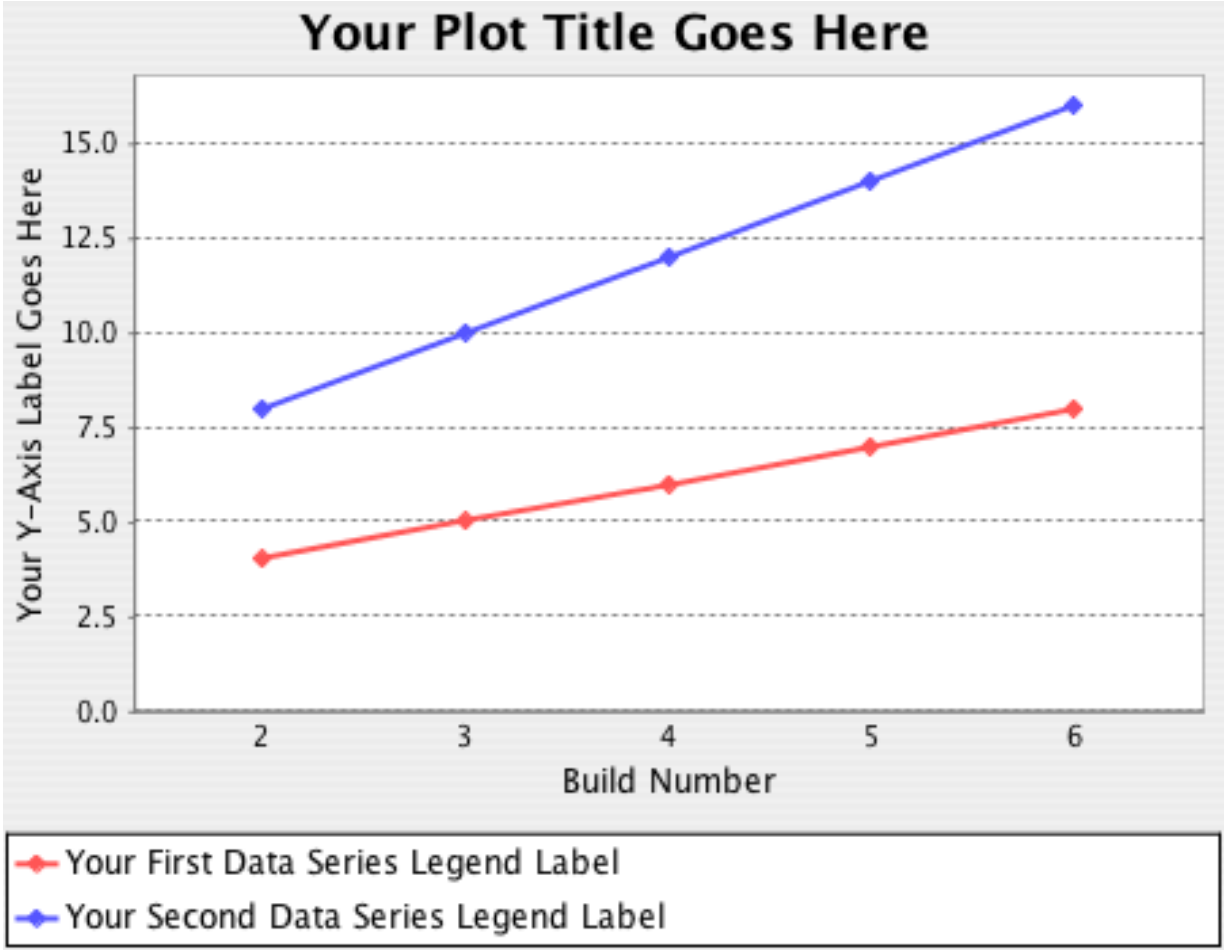


Visualisierung In Jenkins

Performance Dashboard

Mit Plot Plugin

- Zeitserien über mehrere Builds
- Unterschiedliche Datenquellen



Tools Reference

- Jenkins Plot Plugin
  https://wiki.jenkins.io/display/JENKINS/Plot+Plugin
- OSADL Cyclic Test
  https://rt.wiki.kernel.org/index.php/Cyclictest
- Robot Framework
  http://robotframework.org/