THPP21

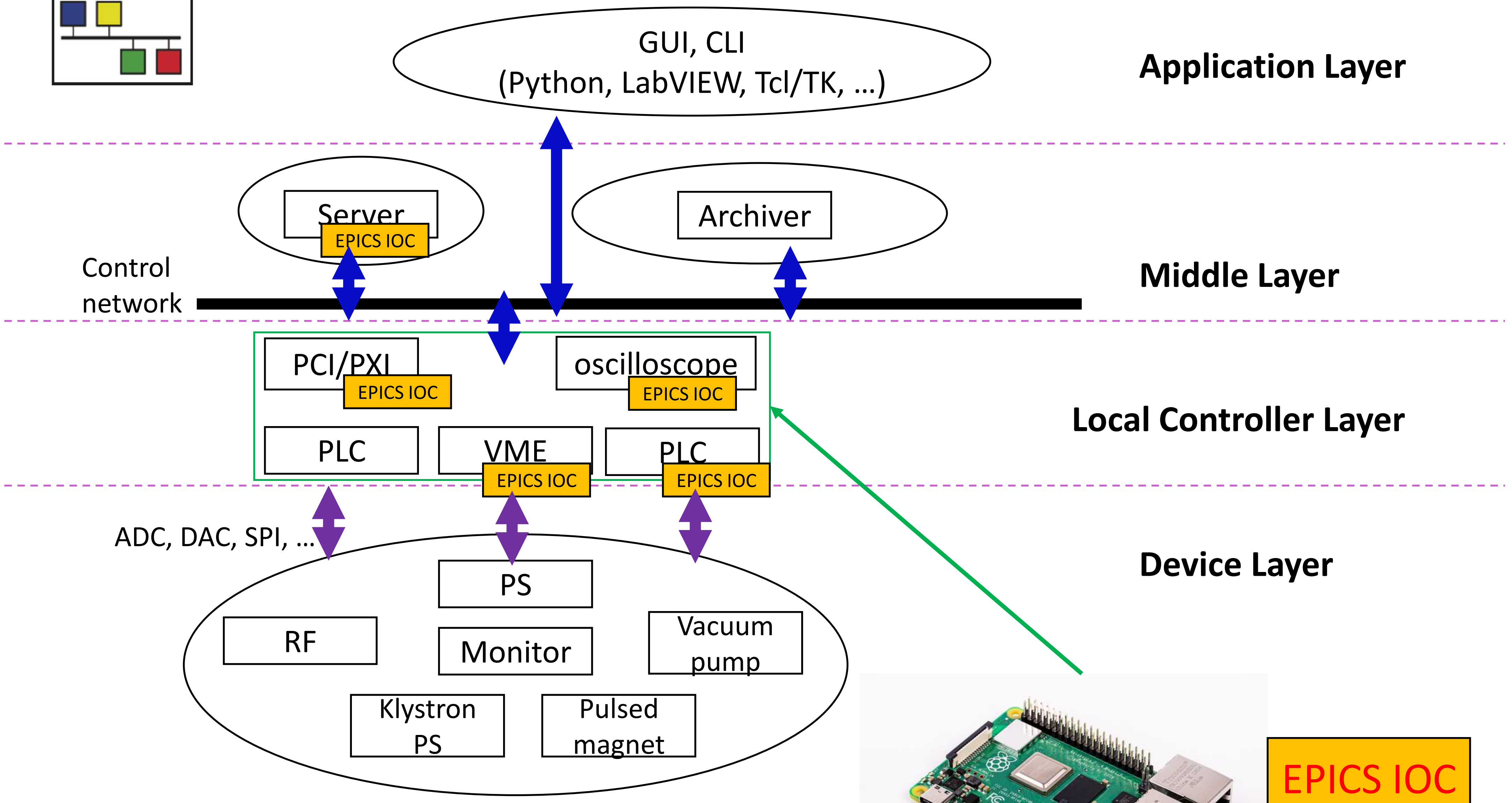
Raspberry Pi を用いたEPICS制御システムのリアルタイム性能試験
Real Time Capability Test on The Raspberry Pi based EPICS Controller

ワン　デイ
王 迪（総研大 加速器科学）， 古川 和朗, 佐々木 信哉, 佐藤 政則, 帯名 崇, 榎本 嘉範
（高エネルギー加速器研究機構）

要旨

Raspberry Pi is a series of small single-board computers designed and published by the Raspberry Pi Foundation. It is also a low cost and popular solution in building the EPICS Input/output Controller (IOC) owing to its high flexibility, expansion capability and affordability. EPICS is also adopted in the electron-positron collider, SuperKEKB accelerator. However, the requirement of the real time capability of the control system is critical since the specific demand of the injection system. The injector linac needs to switch lots of hardware parameters every 20 millisecond to perform the injection into several different storage rings. To assess the possibility and stability of Raspberry Pi based EPICS IOC within the 50 Hz beam repetition rate, we perform a latency test on the Raspberry Pi and compare the standard kernel and real time kernel.

Control System Overview



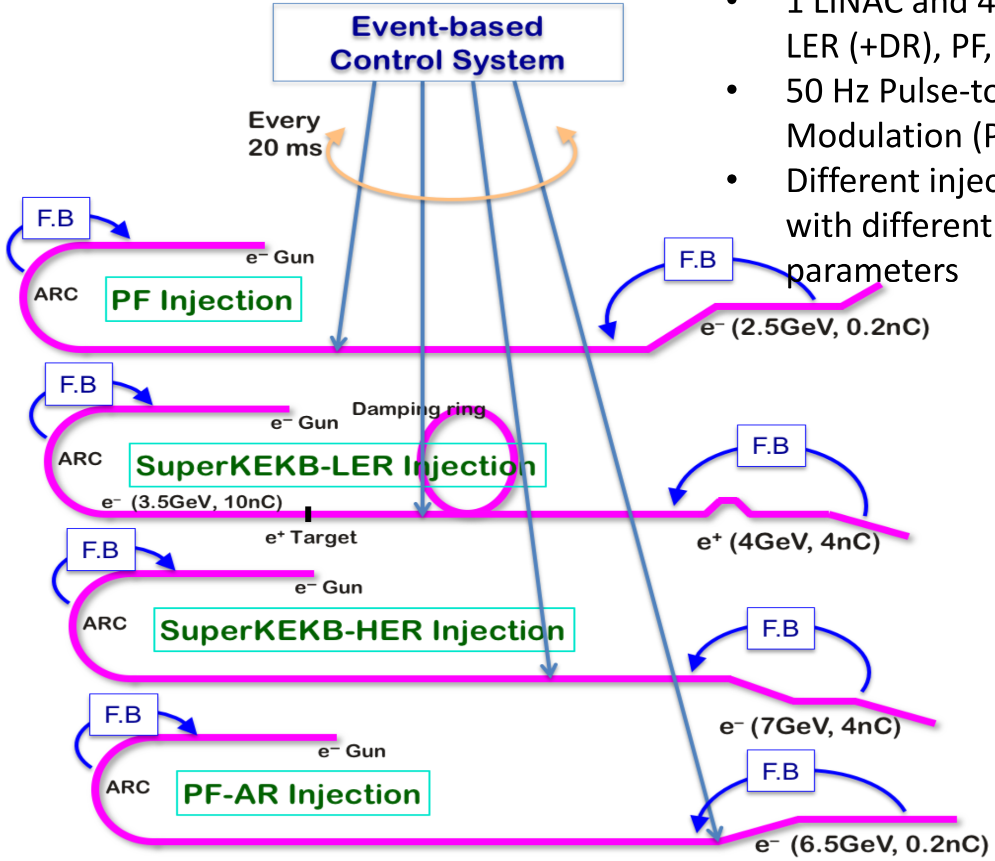
EPICS IOC
???

What is Raspberry Pi

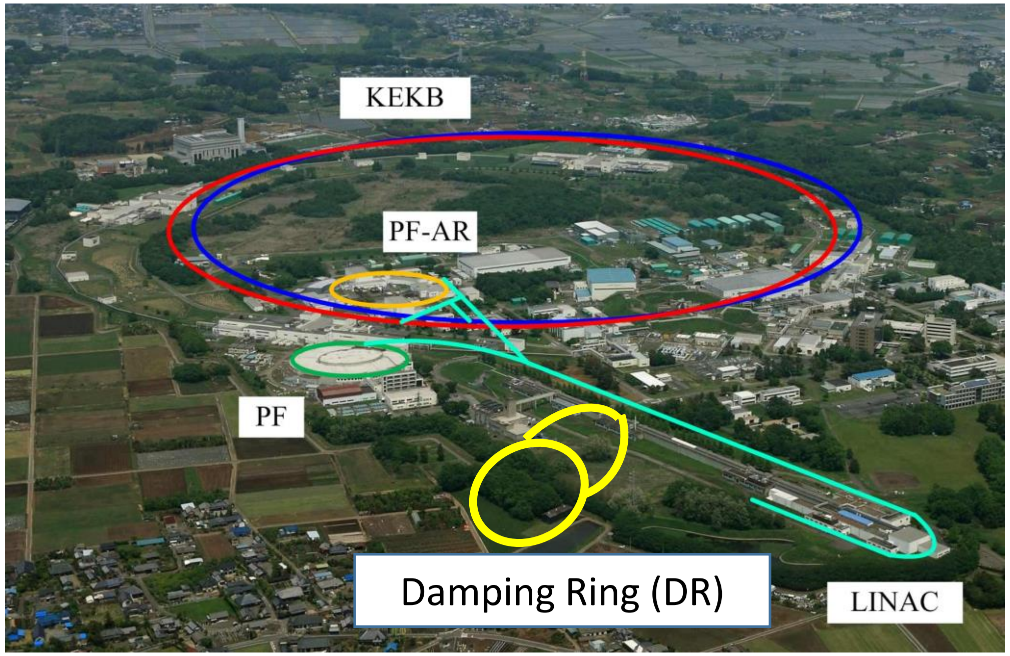
- Low cost
- Small
- Highly customizable
- Tested at several accelerators:
    - Banana Pi used at TPS, Taiwan[1]
    - Raspberry Pi used at SPES, Italy[2]

Raspberry Pi 4 Model B (June, 2019)

CPU	Broadcom BCM2711 quad-core Cortex-A72 64-bit SoC @ 1.5GHz
Memory	2GB, 4GB or 8GB LPDDR4
Network	IEEE 802.11b/g/n/ac wireless Bluetooth 5.0 Gigabit Ethernet
I/O	USB, 40-pin GPIO header
OS	Debian, Raspberry Pi OS
Power	5V DC via USB-C connector



- 1 LINAC and 4(+1) Ring (HER, LER (+DR), PF, PF-AR)
- 50 Hz Pulse-to-Pulse Modulation (PPM)
- Different injection mode with different control parameters



Detail is described at *The fault analysis of timing system in SuperKEKB (**WEPP24**)*

Reference

1, DESIGN AND IMPLEMENTATION OF EMBEDDED APPLICATIONS WITH EPICS SUPPORT FOR ACCELERATOR CONTROLS, IPAC2016, Busan, Korea

2, NEW CONTROL SYSTEM FOR THE SPES OFF-LINE LABORATORY AT LNL-INFN USING EPICS IOCS BASED ON THE RASPBERRY PI, f ICALEPCS2013, San Francisco, CA, USA

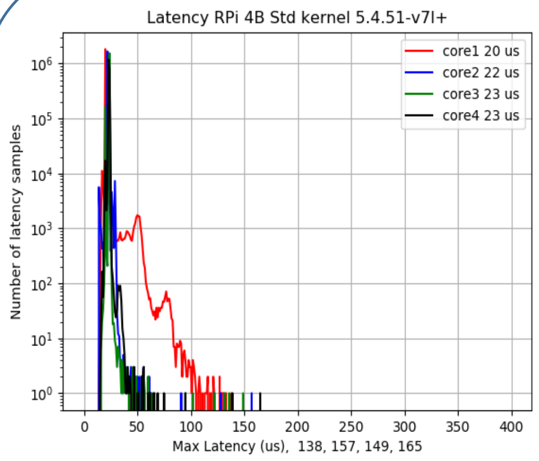
Real Time Test of RPi with two kernels

- Use `*cyclictest*` tool in RT-tests toolbox
- Open source
- Easy to use
- Several conditions will be tested:
  - CPU frequency
  - CPU stress method
  - Thread sleep time
  - Std kernel and RT kernel
- Mostly use `*hackbench*` to stress the CPU, `cat /dev/zero > /dev/null ` and `pi_stress (baed on Priority Inversion)` also used for comparison
- Compile the real time kernel from: https://github.com/raspberrypi/linux/tree/rpi-4.19.y-rt

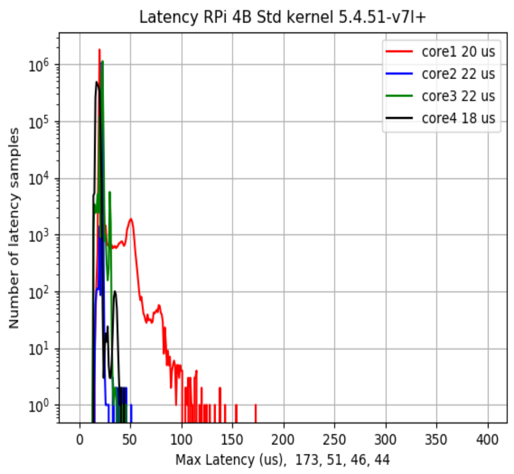
```
sudo apt-get install rt-tests

# test rt-tests
sudo cyclictest -t1 -p 80 -n -i 10000 -l 10000
# output should like (on WSL ubuntu 18.04):
# policy: fifo: loadavg: 0.52 0.58 0.59 1/9 432
#
# T: 0 (  431) P:80 I:10000 C:  10000 Min:     9 Act:  248 Avg:  510 Max:    1429
# min latency: 9 us, average: 510 us, max: 1429 us.
```

- Standard kernel supports CPU frequency: 600MHz, 750MHz, 1000MHz, 1500MHz
- RT kernel supports 600MHz and 1500MHz

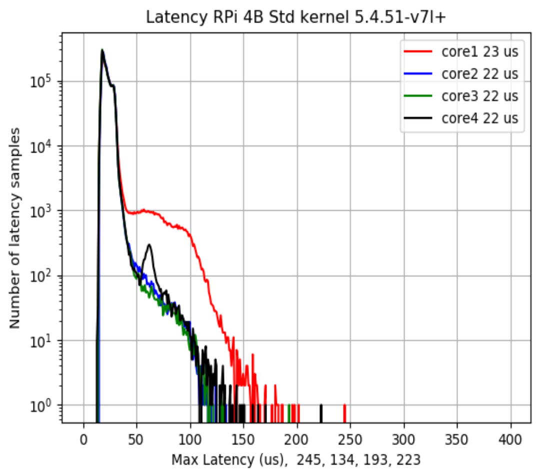
The CPU stress method comparison based on **Std kernel**



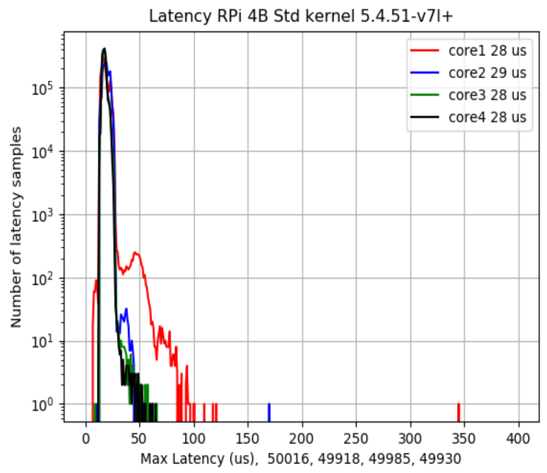
No stress



Use `cat /dev/zero > /dev/null` to stress single CPU



hackbench

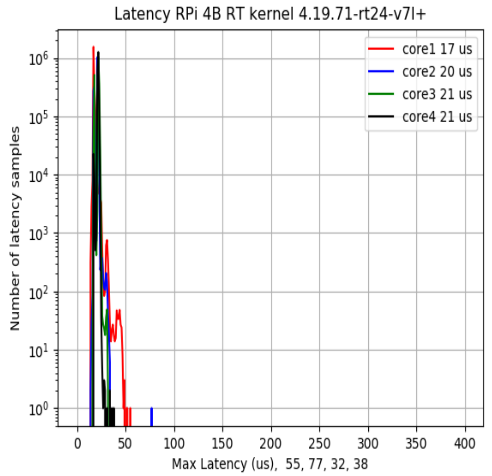


Pi_stress

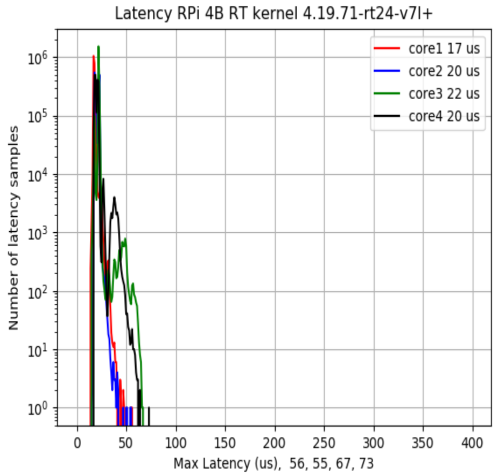
Note that the vertical axis scale changes

Real Time Test of RPi with two kernels

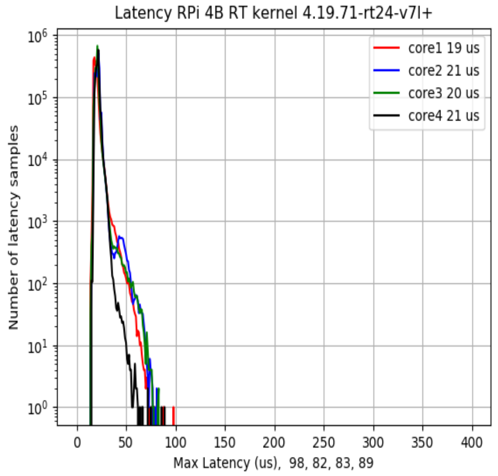
The CPU stress method comparison based on **RT kernel**



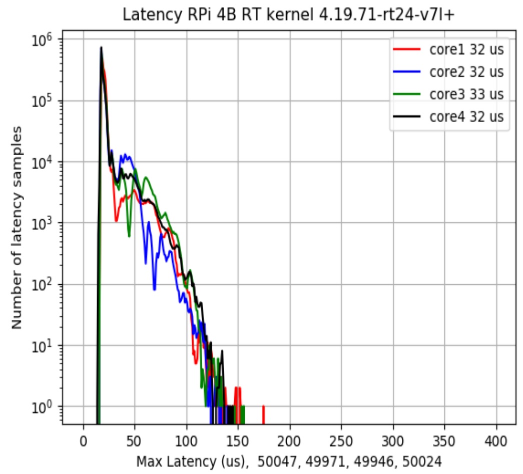
No stress



Use `cat /dev/zero > /dev/null` to stress single CPU

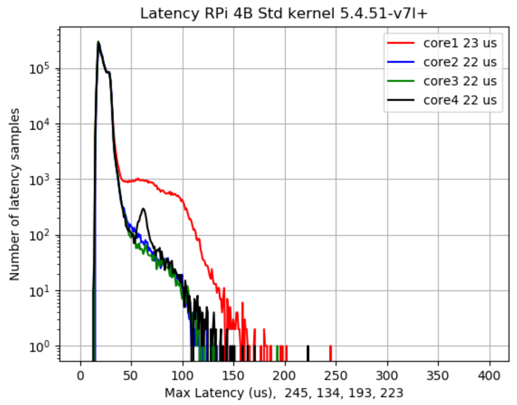


hackbench

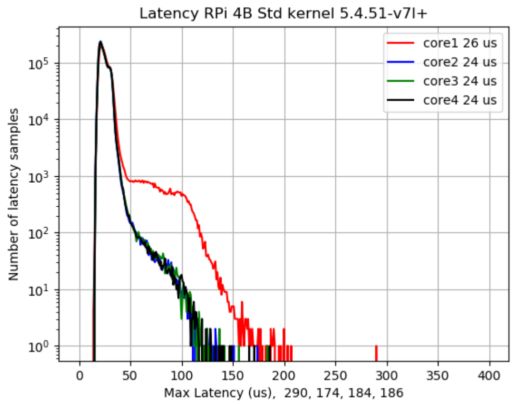


Pi_stress

Program running time comparison



Hackbench stress on Std kernel, runs for 10 minutes

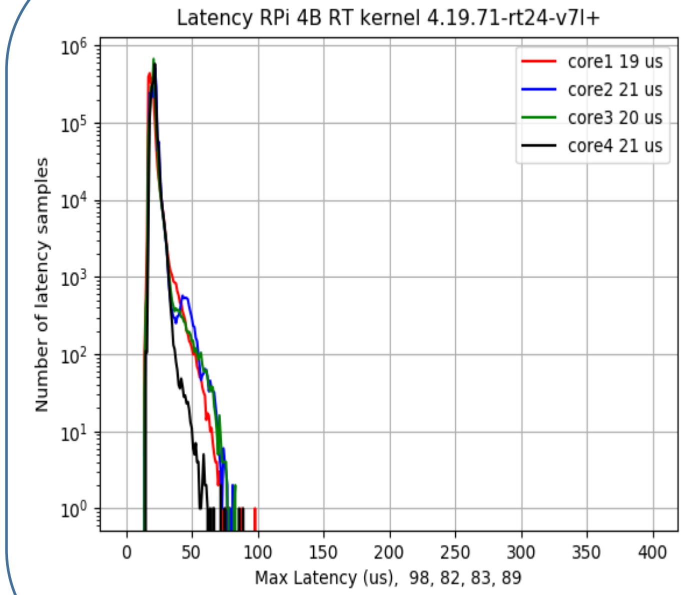


Hackbench stress on Std kernel, runs for 50 minutes

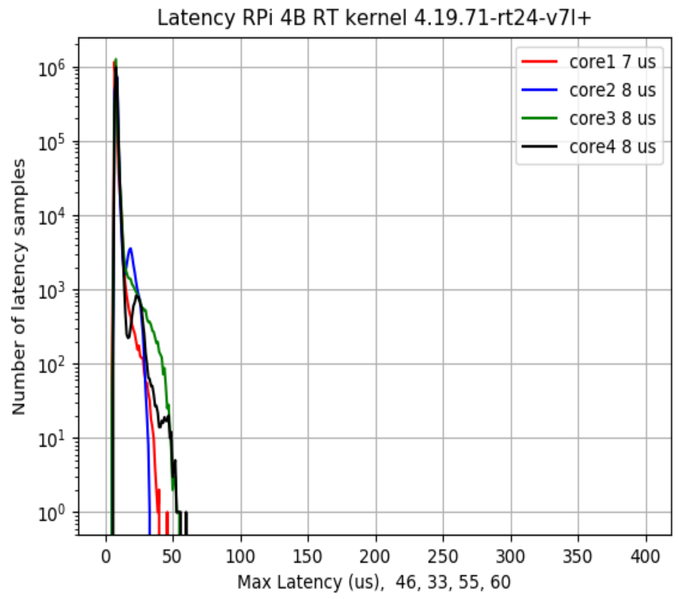
- The average latency remains the same between Std and RT kernel
- The maximal latency of RT kernel is half of the Std kernel
- Stress all CPUs will obviously increase the latency while stressing one CPU will only increase one core's latency
- Priority Inversion stress is a little bit difficult for RPi to handle
- The program running time does not affect the latency both on Std and RT kernel

Real Time Test of RPi with two kernels

CPU frequency comparison of **RT kernel**



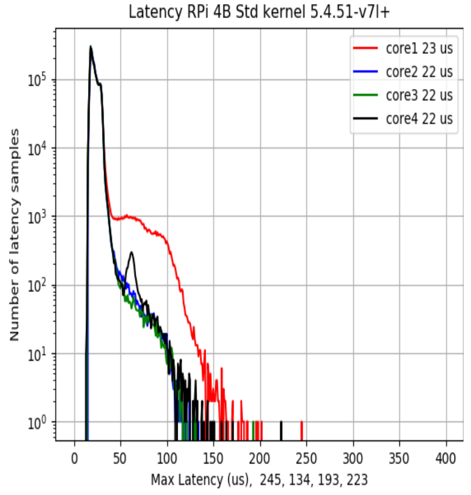
CPU: 600 MHz



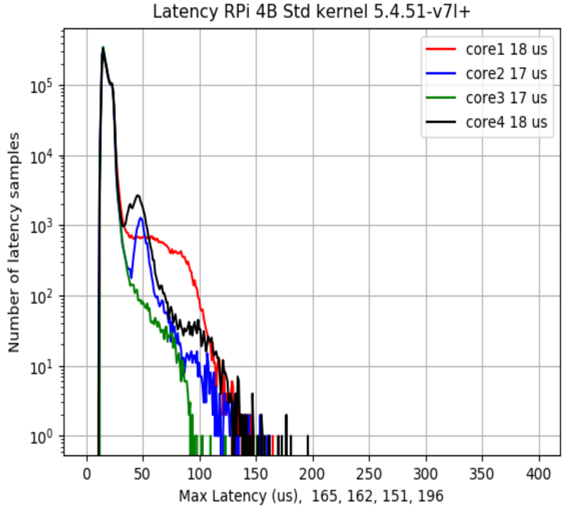
CPU: 1500 MHz

- With the increase of CPU frequency, the latency decreases
- Even on the lowest CPU frequency of RT kernel, the maximal latency (90 us) is still smaller than the high performance CPU of the standard kernel (110 us)
- The best latency is 8 us for average latency and 50 us for maximal latency on RT kernel
- CPU temperature is acceptable with 1.5 GHz on RT kernel without cooling (~65 ℃)

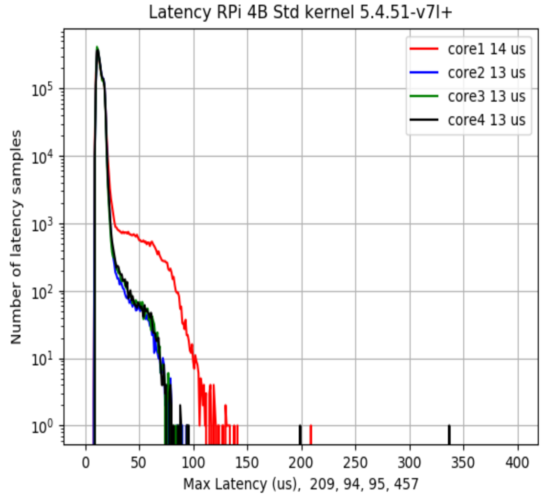
CPU frequency comparison of **Std kernel**



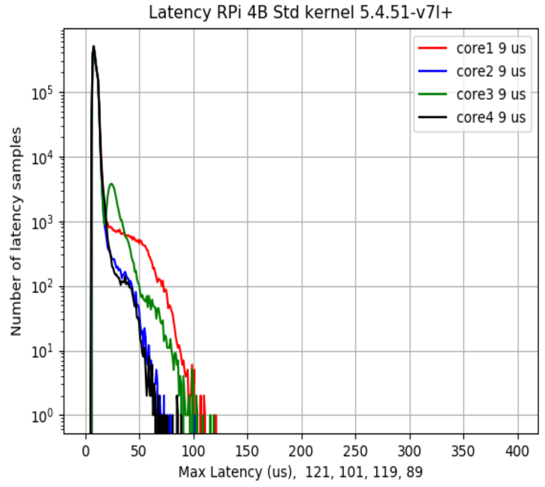
CPU: 600 MHz



CPU: 750 MHz

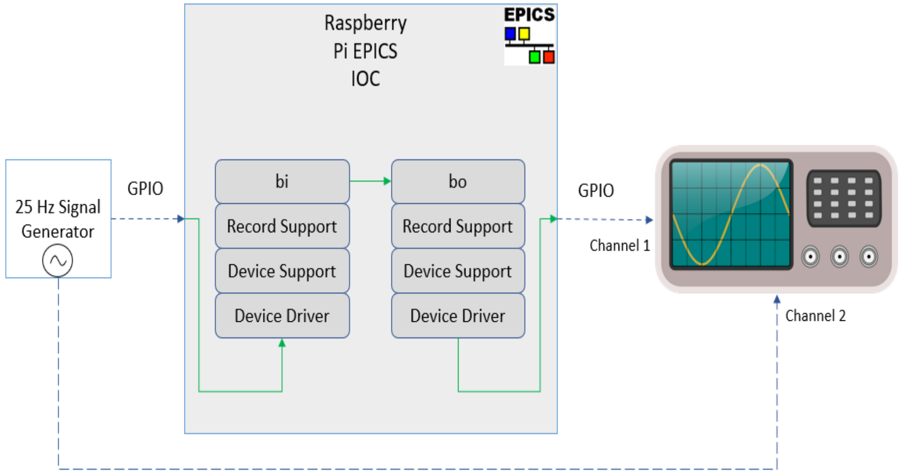


CPU: 1000 MHz

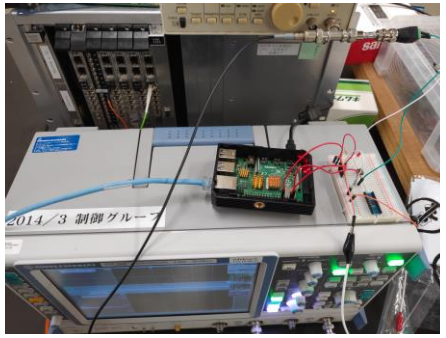


CPU: 1500 MHz

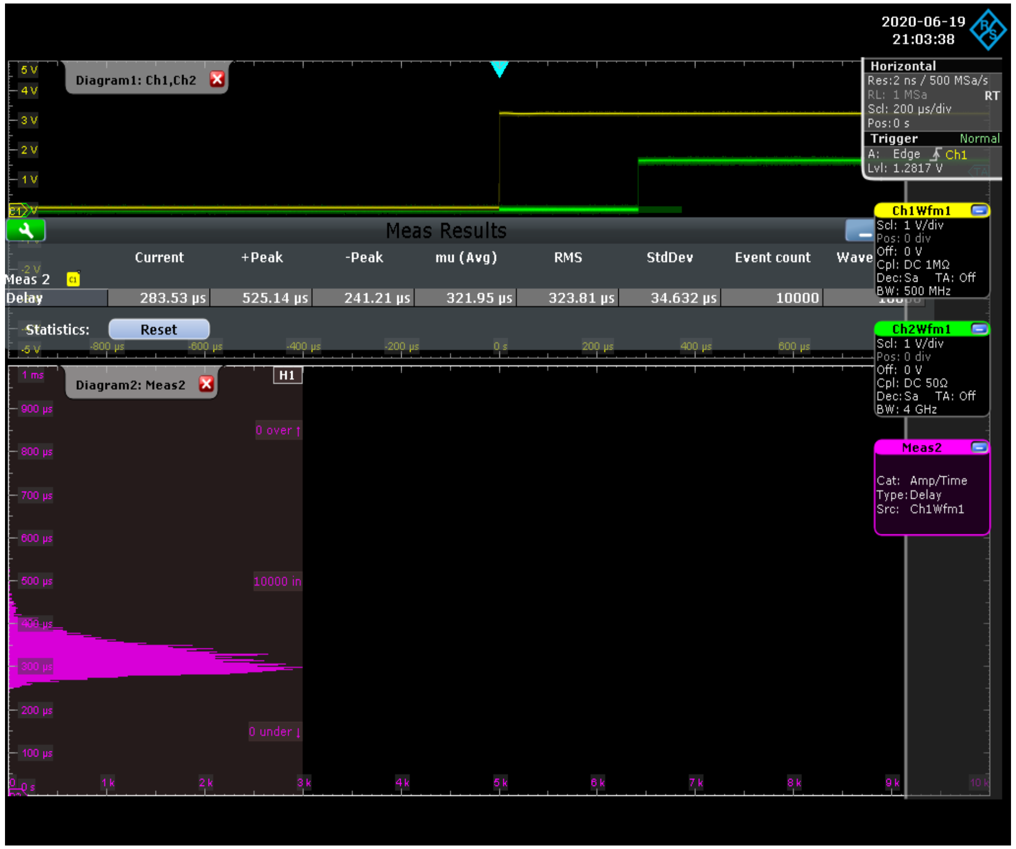
Real Time Test of RPi IOC with two kernels



- 25 Hz square wave (half of the SuperKEKB beam repetition rate)
- Use hackbench to stress the CPU
- EPICS version: 3.14.12.8
- Set IOC priority as 90 (by `chrt` command)
- CPU frequency is 1.5 GHz
- Two PVs only in the IOC
- Device driver uses Linux `epoll` mechanism



Conclusion: Unlike the `cyclictest` result, there is no much difference between IOC on Std kernel and RT kernel. Owing to the interrupt handling and context switch strategy, EPICS IOC on RT kernel takes lightly longer time while the stability is better.


2020-06-19 21:03:38

Horizontal
Res:2 ns / 500 MSa/s
RL: 1 MSa RT
Scl: 200 µs/div
Pos:0 s
Trigger Normal
A: Edge Ch1
Lvl: 1.2817 V

Meas Results

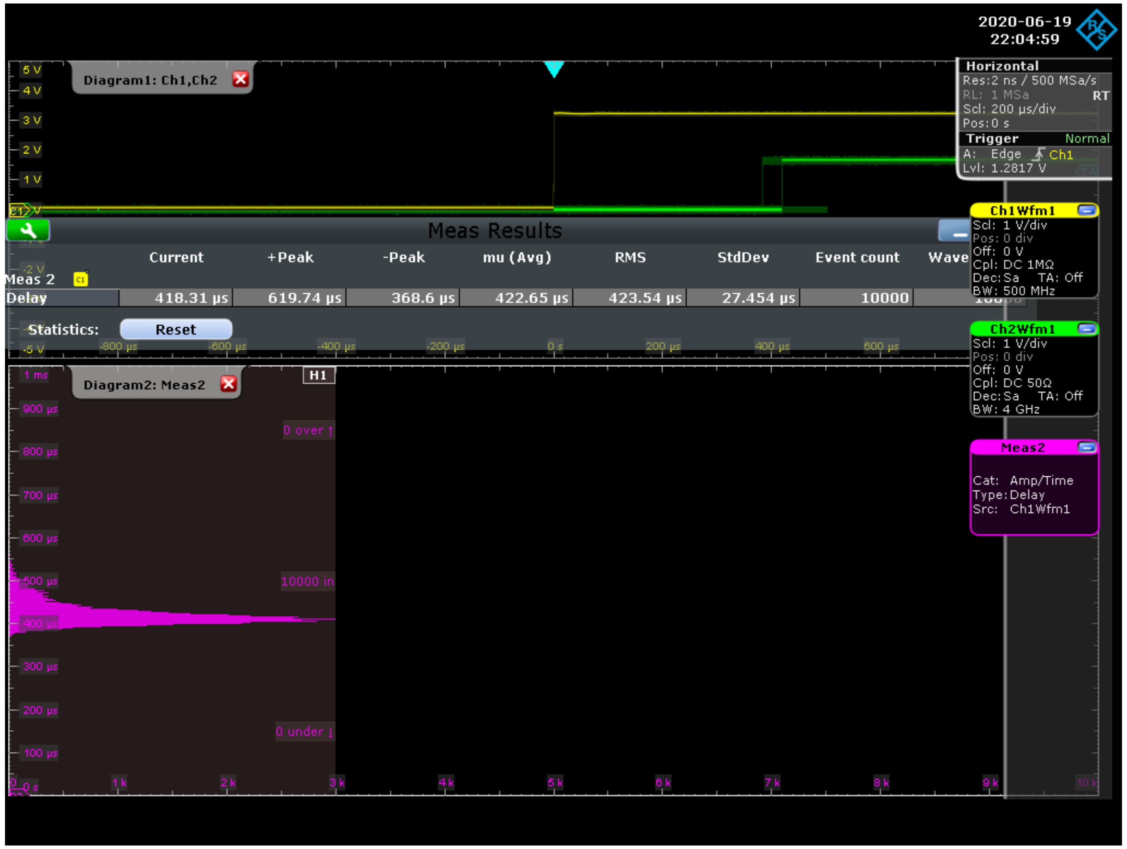
	Current	+Peak	-Peak	mu (Avg)	RMS	StdDev	Event count
Meas 2 Delay	283.53 µs	525.14 µs	241.21 µs	321.95 µs	323.81 µs	34.632 µs	10000

Ch1Wfm1
Scl: 1 V/div
Pos: 0 div
Off: 0 V
Cpl: DC 1MΩ
Dec:Sa TA: Off
BW: 500 MHz

Ch2Wfm1
Scl: 1 V/div
Pos: 0 div
Off: 0 V
Cpl: DC 50Ω
Dec:Sa TA: Off
BW: 4 GHz

Meas2
Cat: Amp/Time
Type:Delay
Src: Ch1Wfm1

Std Kernel on 1.5 GHz CPU with hackbench


2020-06-19 22:04:59

Horizontal
Res:2 ns / 500 MSa/s
RL: 1 MSa RT
Scl: 200 µs/div
Pos:0 s
Trigger Normal
A: Edge Ch1
Lvl: 1.2817 V

Meas Results

	Current	+Peak	-Peak	mu (Avg)	RMS	StdDev	Event count
Meas 2 Delay	418.31 µs	619.74 µs	368.6 µs	422.65 µs	423.54 µs	27.454 µs	10000

Ch1Wfm1
Scl: 1 V/div
Pos: 0 div
Off: 0 V
Cpl: DC 1MΩ
Dec:Sa TA: Off
BW: 500 MHz

Ch2Wfm1
Scl: 1 V/div
Pos: 0 div
Off: 0 V
Cpl: DC 50Ω
Dec:Sa TA: Off
BW: 4 GHz

Meas2
Cat: Amp/Time
Type:Delay
Src: Ch1Wfm1

RT Kernel on 1.5 GHz CPU with hackbench