SKPS-LAB4
Sprawozdanie

Matvii Ivashchenko
Katsyaryna Anikevich

1 Przetestowanie działania programów na "gospodarzu"

Zbudowaliśmy program poleceniem **make**.
Ustawienie dla terminala dodaliśmy poleceniem

```
nano .bashrc
```

Na końcu pliku .bashrc dodaliśmy:

```
export PATH="$PATH:/home/student-pl/Pobrane/skps_lab4_student/cw4_owrt_pkg/cwic"
```

Zaktualizowaliśmy konfigurację terminalu przez polecenie:

```
source ~/.bashrc
```

Uruchomiliśmy skrypt:

```
student-pl@lab-44:~$ cw4a 3 100 10000 600000
Client: 1, nsmp=100, del=600000
Client: 2, nsmp=100, del=600000
Client: 0, nsmp=100, del=600000
Sample 0, client 1, delivery time: 40
```

2 Zbudowanie pakietu dla OpenWRT

Otworzyliśmy plik ze ścieżkami:

```
Linux-x86_64$ nano feeds.conf.default
```

Dodaliśmy plik:

```
src-git skps ~home/student-pl/Pobrane/skps_lab4_student/cw4_owrt_pkg
```

Zaktualizowaliśmy plik ze ścieżkami do pakietu

```
scripts/feeds update -a
```

Zainstalowaliśmy pakiet

```
scripts/feeds install -p lab4 -a
```

Wybraliśmy potrzebny pakiet w **menuconfig** w kategorii Examples.
Zbudowaliśmy pakiet

```
make package/cwicz4mak/compile
```

Wysłaliśmy pakiet z hosta do RPi4

```
scp /home/student-pl/Pobrane/openwrt-sdk-24.10.0-bcm27xx-bcm2711_gcc-13.3.0_musl.Linux-x86_64/
    bin/packages/aarch64_cortex-a72/skps/cwicz4mak_1_aarch64_cortex-a72.ipk
    root@10.42.0.248:/root
```

3 Ustalenie granicznej wartości czasu przetwarzania

3.1 3 klientów, 1 rdzeń, pełne obciążenie

Otworzyliśmy konfigurację startową w terminalu RPi4:

```
root@OpenWrt:~# vi /boot/cmdline.txt
```

W **boot/user/cmdline.txt** dodaliśmy **maxcpus=1** i zrobiliśmy reboot RPi4.
Zainstalowaliśmy pakiety stress-ng i htop przez polecenie opkg.
Uruchomiliśmy pełne obciążenie procesora:

```
stress-ng --matrix 0 -t 1m &
```

Dopuszczalny czas opóźnienia (czas przetwarzania = 600000):

```
root@OpenWrt:/# cw4a 3 100 10000 600000
Client: 2, nsmp=100, del=600000
Client: 1, nsmp=100, del=600000
Client: 0, nsmp=100, del=600000
Sample 0, client 2, delivery time: 84
Sample 0, client 1, delivery time: 133
Sample 0, client 0, delivery time: 169
Sample 1, client 1, delivery time: 39
Sample 1, client 0, delivery time: 75
Sample 1, client 2, delivery time: 101
Sample 2, client 2, delivery time: 64
```

Ale przy następnym poleceniu czas opóźnienia jest już za długi (czas przetwarzania = 700000)

```
root@OpenWrt:/# cw4a 3 100 10000 700000
Client: 2, nsmp=100, del=700000
Client: 1, nsmp=100, del=700000
Client: 0, nsmp=100, del=700000
Sample 0, client 2, delivery time: 83
Sample 0, client 1, delivery time: 132
Sample 0, client 0, delivery time: 170
Sample 1, client 1, delivery time: 75
Sample 1, client 0, delivery time: 237
Sample 1, client 2, delivery time: 264
Sample 2, client 0, delivery time: 2275
Sample 2, client 1, delivery time: 2345
Sample 2, client 2, delivery time: 2374
Sample 3, client 2, delivery time: 4827
```

Przekazaliśmy pliki z RPi4 na host z rezultatami dla zbudowania histogramu poleceniem scp.

3.2 3 klientów, 2 rdzenie, pełne obciążenie

W **boot/user/cmdline.txt** dodaliśmy **maxcpus=2** i zrobiliśmy reboot RPi4.
Zrobiliśmy pełne obciążenie

```
stress-ng --matrix 0 -t 1m &
```

Przy następnym poleceniu czas opóźnienia jest już za długi (czas przetwarzania = 800000)

```
root@OpenWrt:~# cw4a 3 100 10000 800000
...
Sample 10, client 0, delivery time: 55
Sample 10, client 2, delivery time: 1088
Sample 10, client 1, delivery time: 8804
Sample 11, client 0, delivery time: 950
Sample 11, client 1, delivery time: 3888
Sample 11, client 2, delivery time: 3927
Sample 12, client 0, delivery time: 949
Sample 12, client 2, delivery time: 3976
Sample 12, client 1, delivery time: 4010
Sample 13, client 0, delivery time: 921
Sample 13, client 1, delivery time: 4383
Sample 13, client 2, delivery time: 5975
Sample 14, client 0, delivery time: 439
Sample 14, client 2, delivery time: 15404
Sample 14, client 1, delivery time: 15852
```

Przekazaliśmy pliki z RPi4 na host z rezultatami dla zbudowania histogramu poleceniem scp.

3.3 3 klientów, 2 rdzenie, bez obciążenia

W **boot/user/cmdline.txt** dodaliśmy **maxcpus=2** i zrobiliśmy reboot RPi4.
Przy następnym poleceniu czas opóźnienia jest już za długi (czas przetwarzania = 900000)

```
root@OpenWrt:~# cw4a 3 100 10000 900000
Client: 0, nsmp=100, del=900000
Client: 2, nsmp=100, del=900000
Client: 1, nsmp=100, del=900000
Sample 0, client 0, delivery time: 36
Sample 0, client 2, delivery time: 139
Sample 0, client 1, delivery time: 245
Sample 1, client 0, delivery time: 13
Sample 1, client 1, delivery time: 165
Sample 1, client 2, delivery time: 6332
Sample 2, client 0, delivery time: 9
Sample 2, client 1, delivery time: 1079
Sample 2, client 2, delivery time: 3918
Sample 3, client 0, delivery time: 2042
Sample 3, client 2, delivery time: 3657
Sample 3, client 1, delivery time: 3993
```

Przekazaliśmy pliki z RPi4 na host z rezultatami dla zbudowania histogramu poleceniem scp.

3.4 1 klient, 4 rdzenie, bez obciążenia

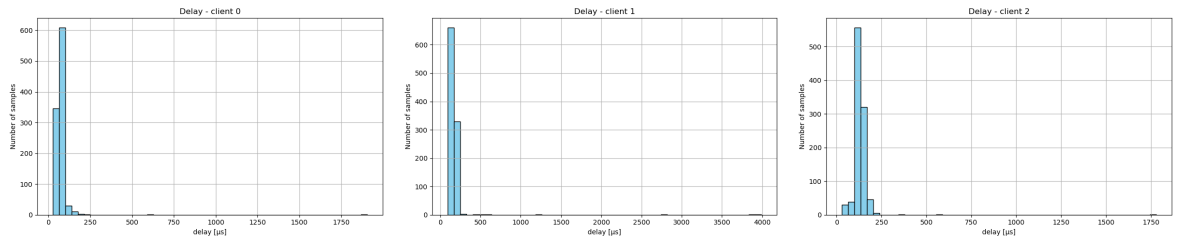
W **boot/user/cmdline.txt** dodaliśmy **maxcpus=4** i uruchomiliśmy ponownie RPi4.
Przy następnym poleceniu czas opóźnienia jest już za długi (czas przetwarzania = 1600000)

```
root@OpenWrt:/# cw4a 1 100 10000 1600000
Client: 0, nsmp=100, del=1600000
Sample 0, client 0, delivery time: 23
Sample 1, client 0, delivery time: 5230
Sample 2, client 0, delivery time: 8920
Sample 3, client 0, delivery time: 8692
Sample 4, client 0, delivery time: 5473
Sample 5, client 0, delivery time: 6035
Sample 6, client 0, delivery time: 6230
Sample 7, client 0, delivery time: 6638
Sample 8, client 0, delivery time: 6004
Sample 9, client 0, delivery time: 5941
Sample 10, client 0, delivery time: 5838
Sample 11, client 0, delivery time: 5043
```

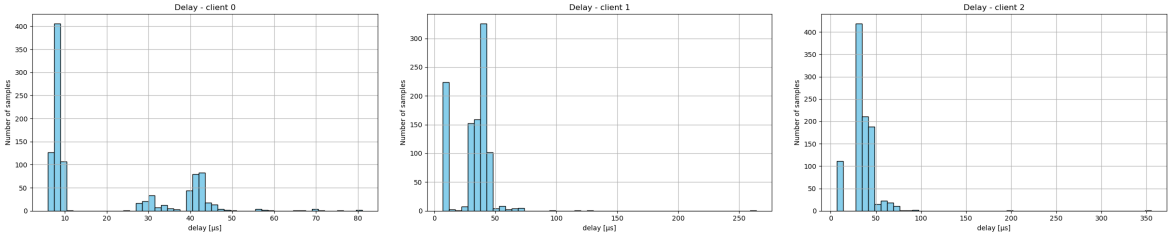
Przekazaliśmy pliki z RPi4 na host z rezultatami dla zbudowania histogramu poleceniem scp.

4 Rozkład czasu dostarczenia danych

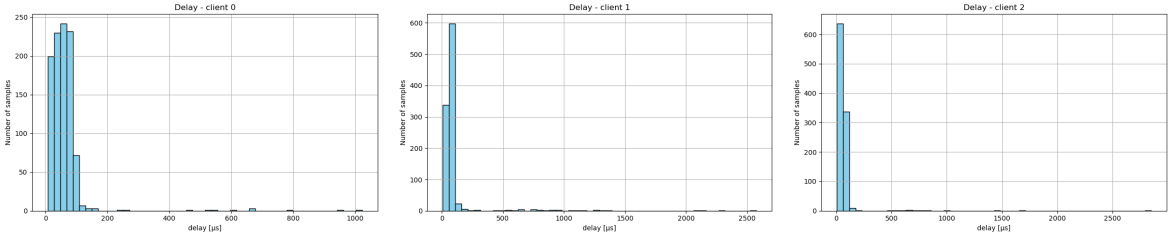
4.1 3 klientów, 1 rdzeń, pełne obciążenie



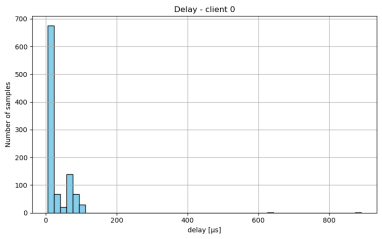
4.2 3 klientów, 2 rdzenie, pełne obciążenie



4.3 3 klientów, 2 rdzenie, bez obciążenia

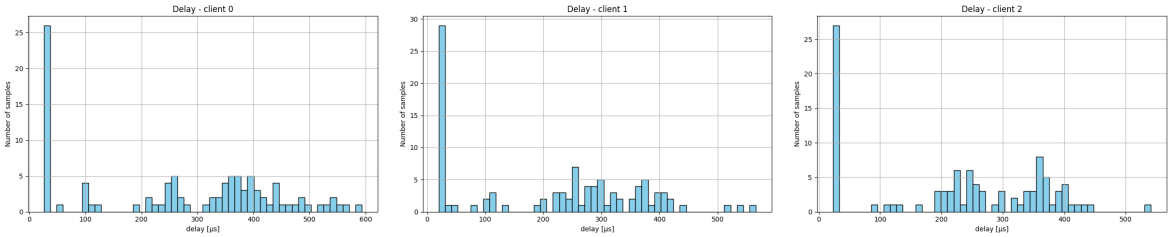


4.4 1 klient, 4 rdzenie, bez obciążenia

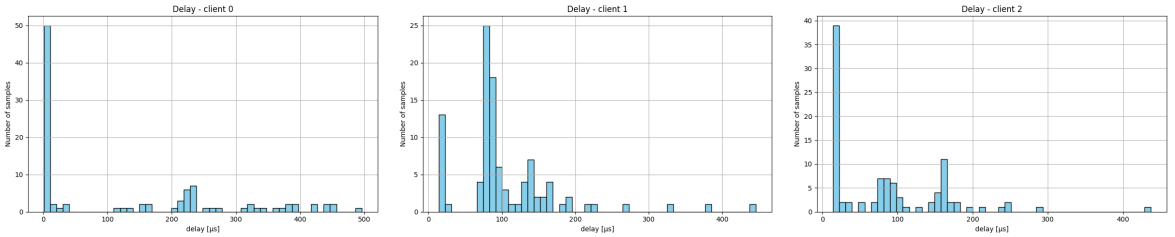


5 Aktywne oczekiwanie

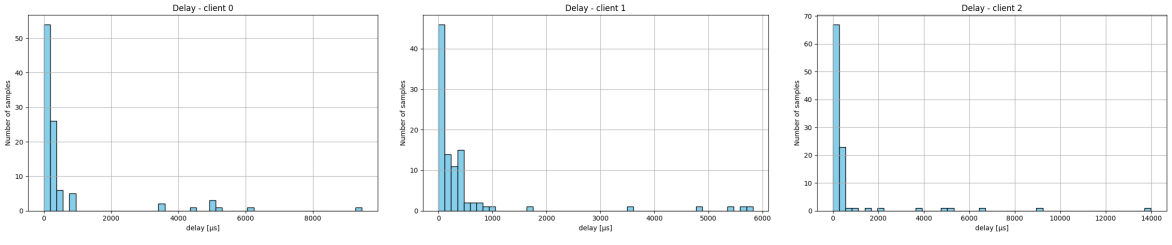
Plik cw4b został zmodyfikowany w ten sposób, że wczytuje tryb oczekiwania z pliku cw4b_mode.txt. Testowe uruchomienie: 3 klienta czekają w zwykłym trybie.



Przy aktywnym oczekiwaniu możemy zauważyć zmniejszenie czasu oczekiwania dla klienta 0, dla innych czas stał się gorszy i nierównomierny.

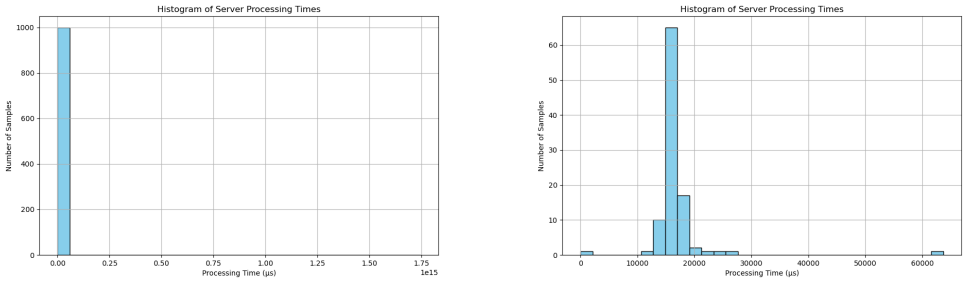


Dla wszystkich czas stał się gorszy i nierównomierny, bo procesy konkurują o czas procesora.



6 Właściwy pomiar czasu

Serwer powinien wykorzystywać timer zamiast blokować się na określony czas. Pierwszy zapis zmieniliśmy na 0.



(a) przed poprawieniem błędu

(b) po poprawieniu błędu