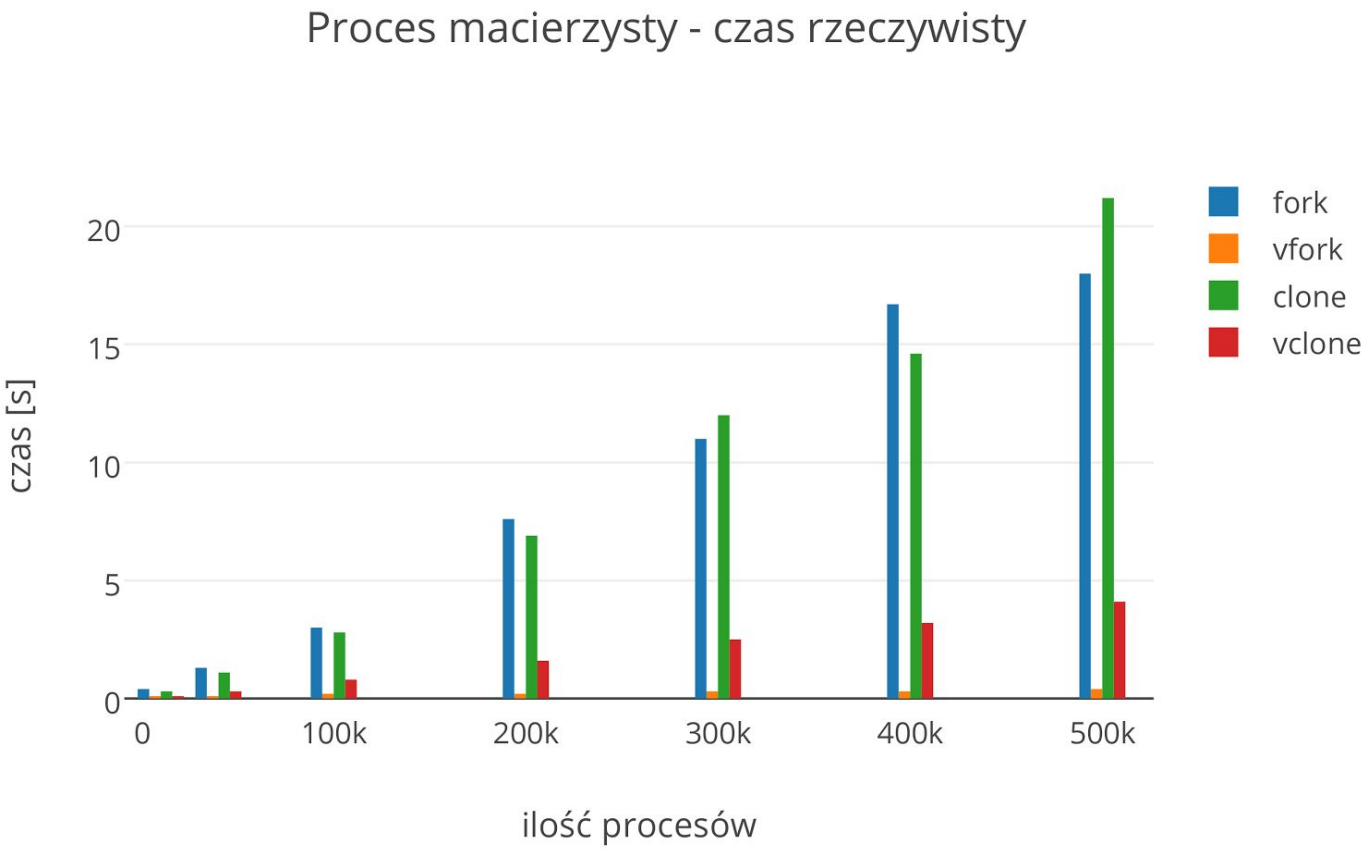
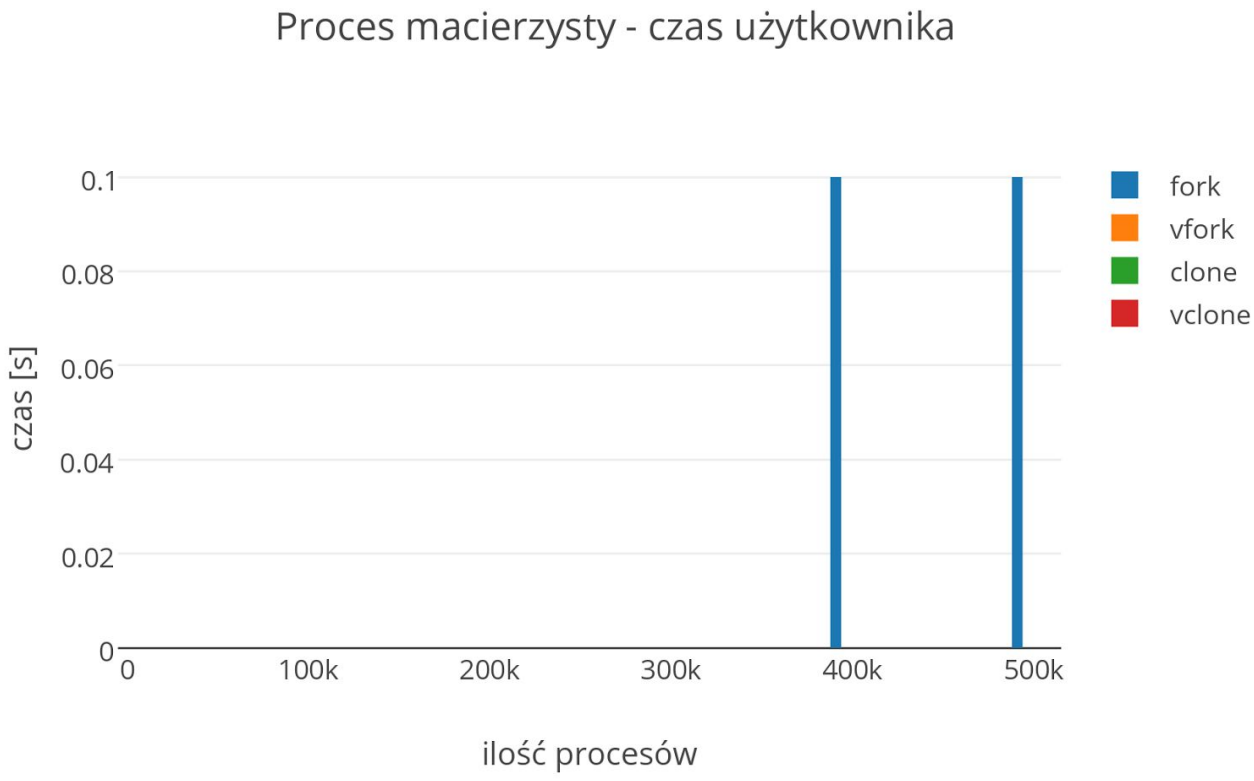
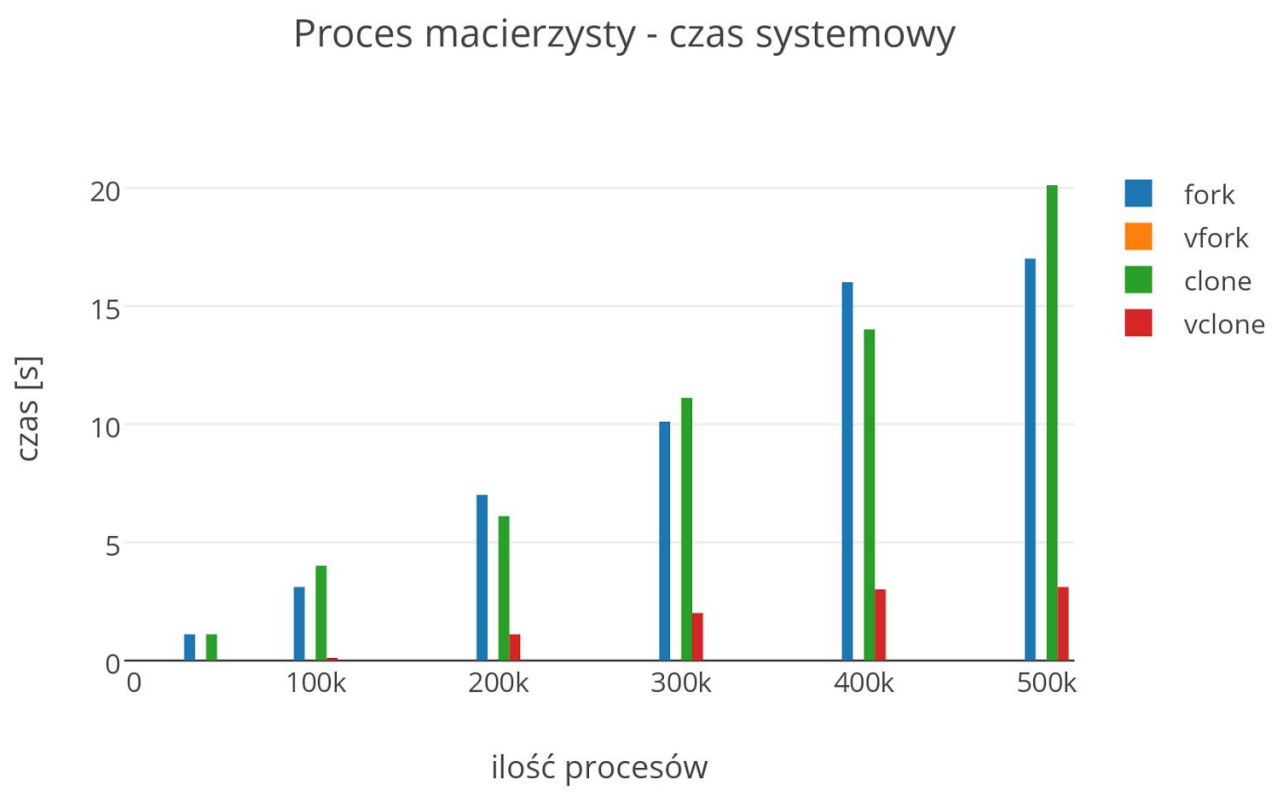
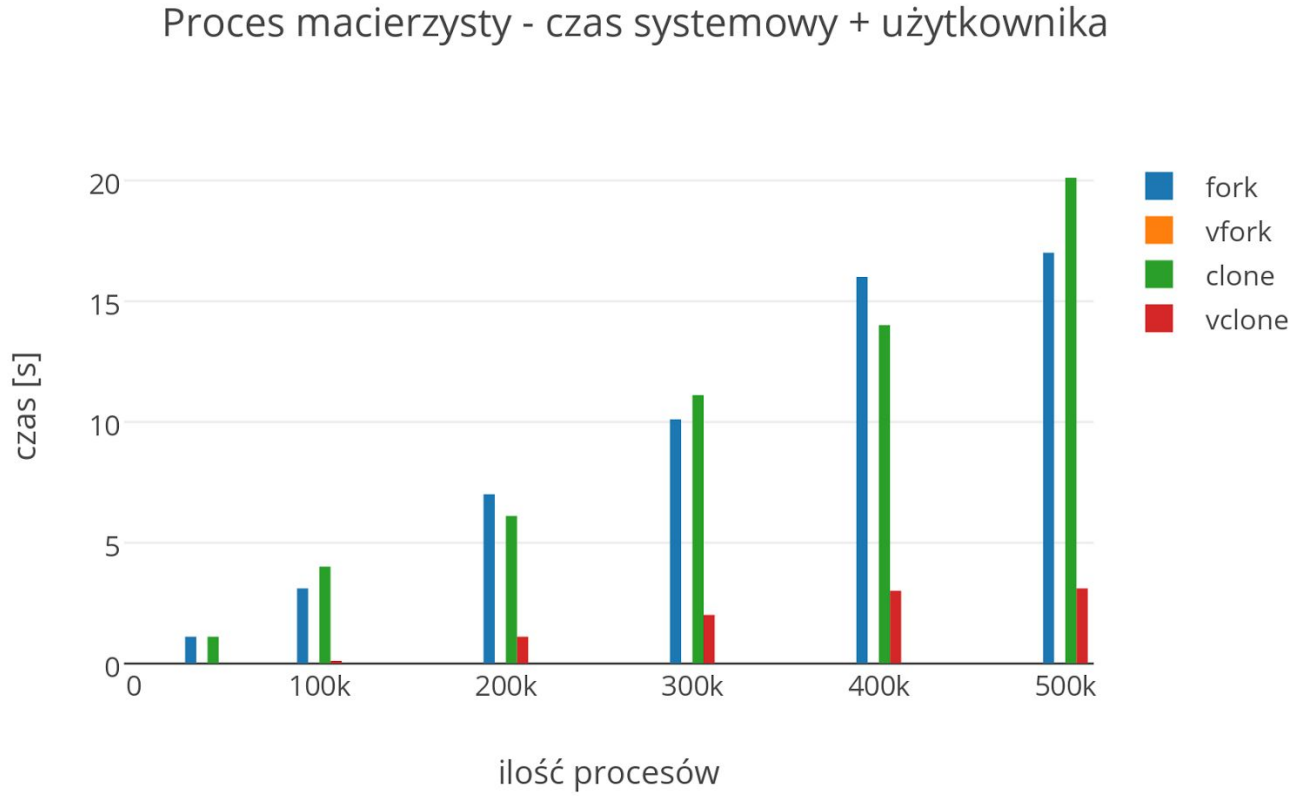
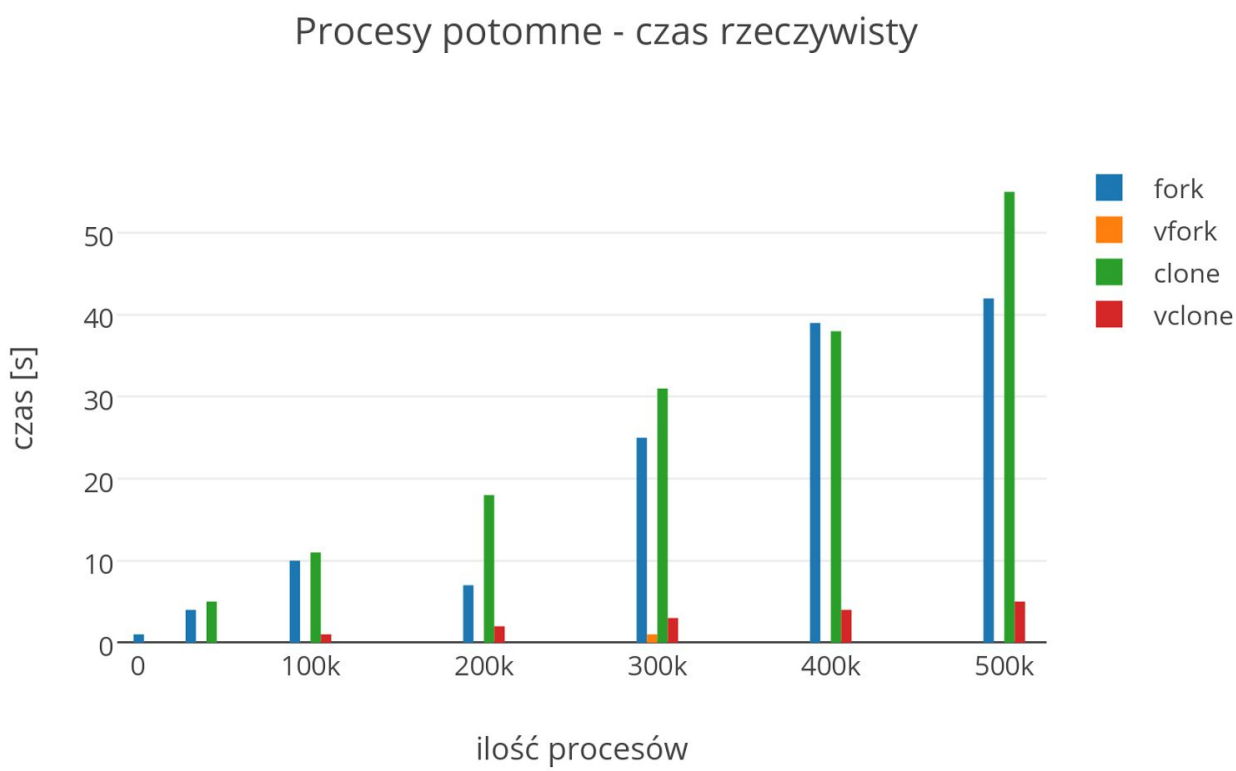
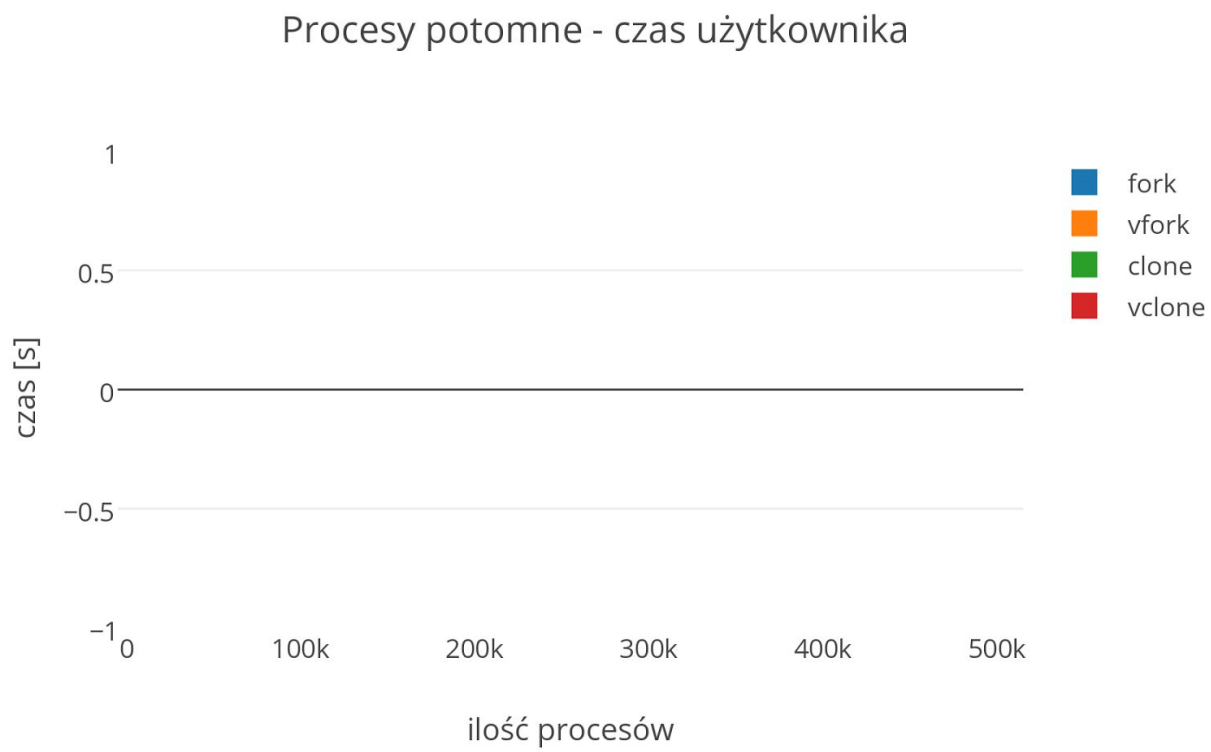

Proces macierzysty - czas rzeczywisty
fork
vfork
clone
vclone
20
15
10
5
0
czas [s]
0
100k
200k
300k
400k
500k
ilość procesów

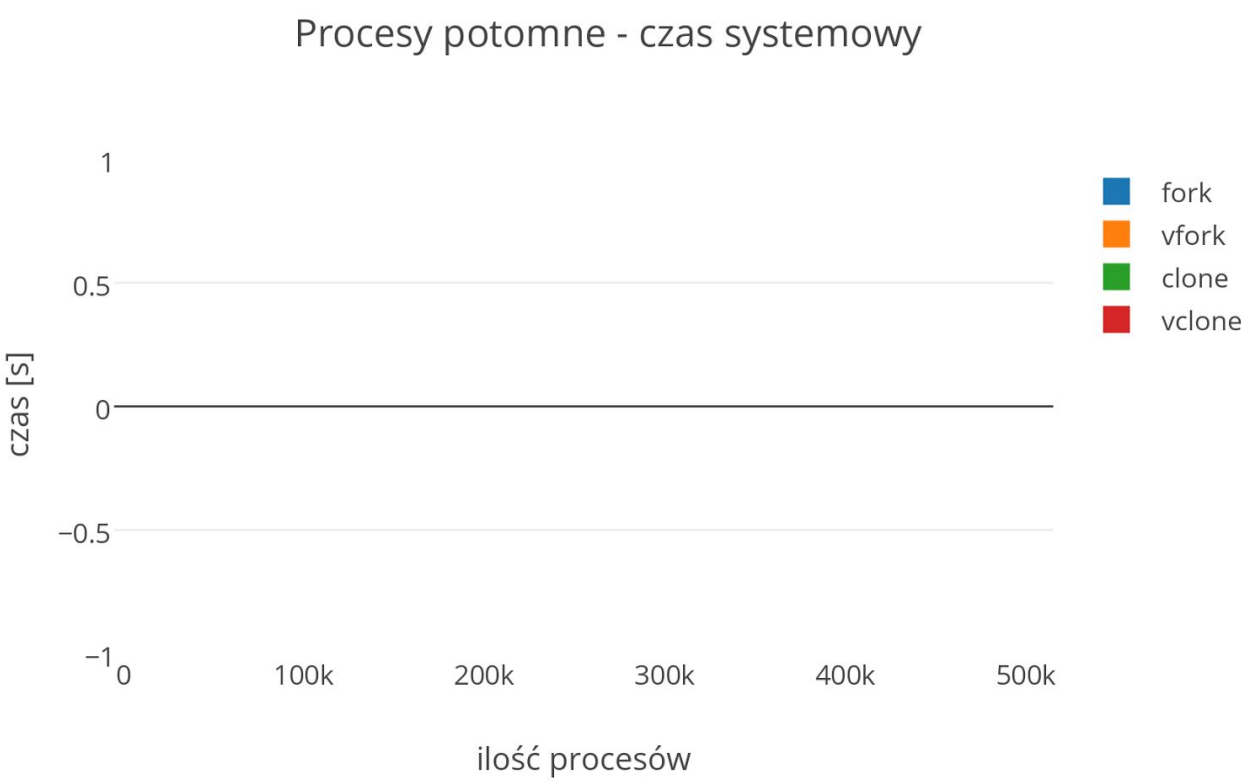

Proces macierzysty - czas użytkownika
fork
vfork
clone
vclone
0.1
0.08
0.06
0.04
0.02
0
czas [s]
0
100k
200k
300k
400k
500k
ilość procesów

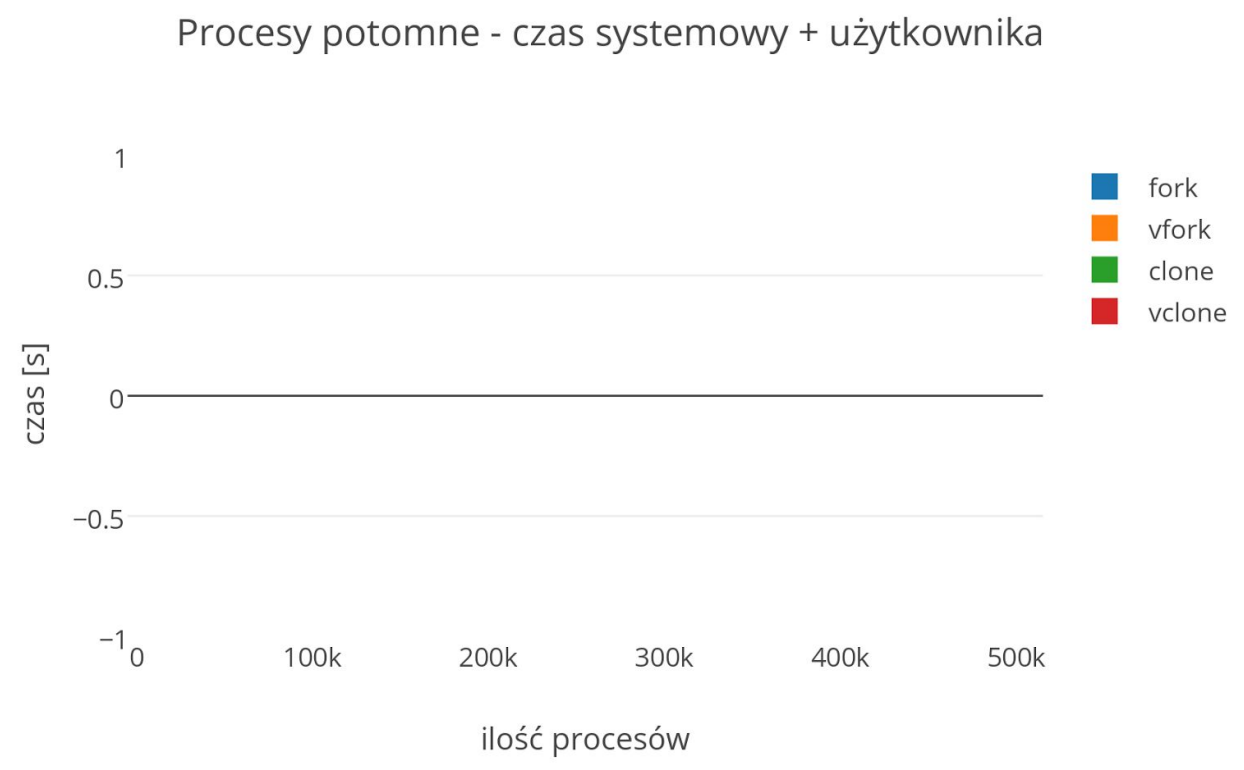

Proces macierzysty - czas systemowy
20
15
10
5
0
czas [s]
0
100k
200k
300k
400k
500k
ilość procesów
fork
vfork
clone
vclone

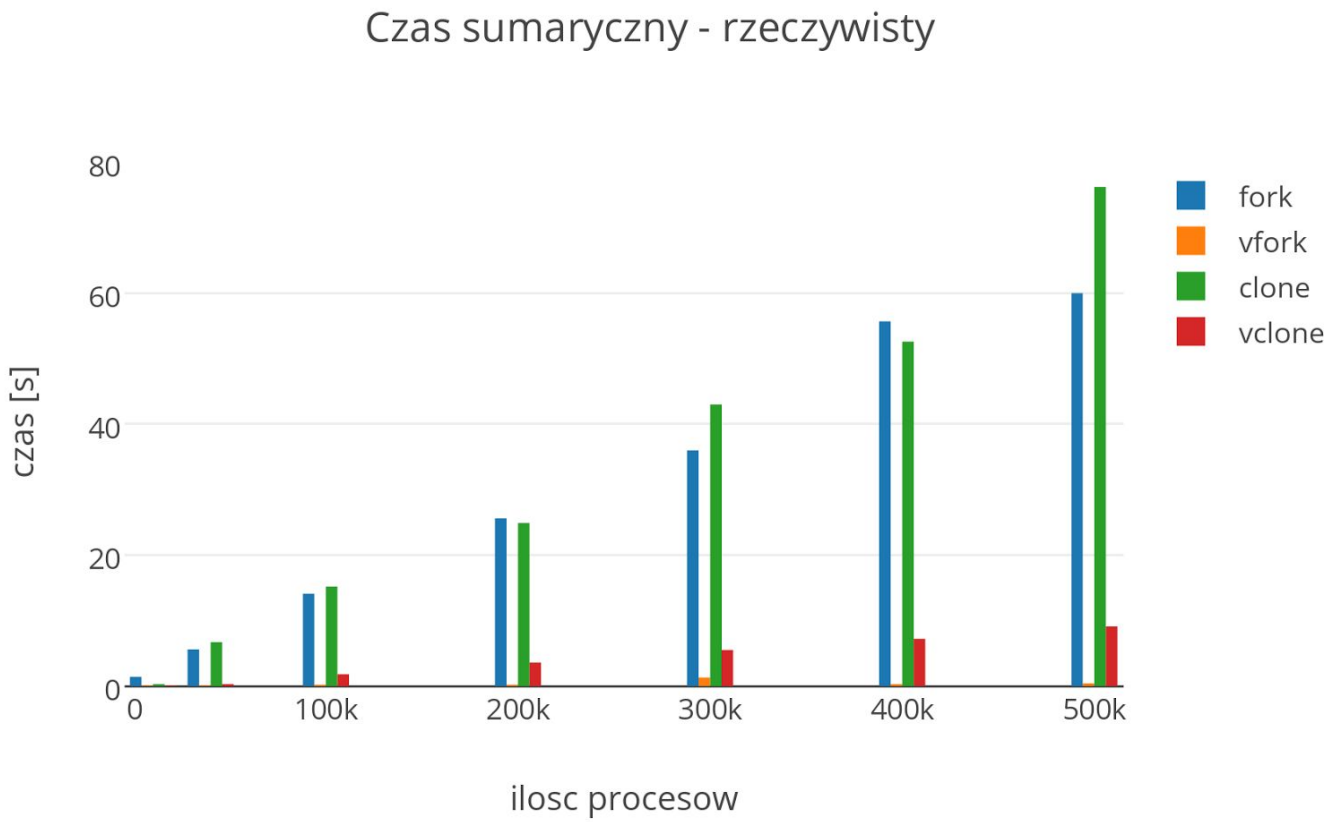

Proces macierzysty - czas systemowy + użytkownika
20
15
10
5
0
czas [s]
0
100k
200k
300k
400k
500k
ilość procesów
fork
vfork
clone
vclone

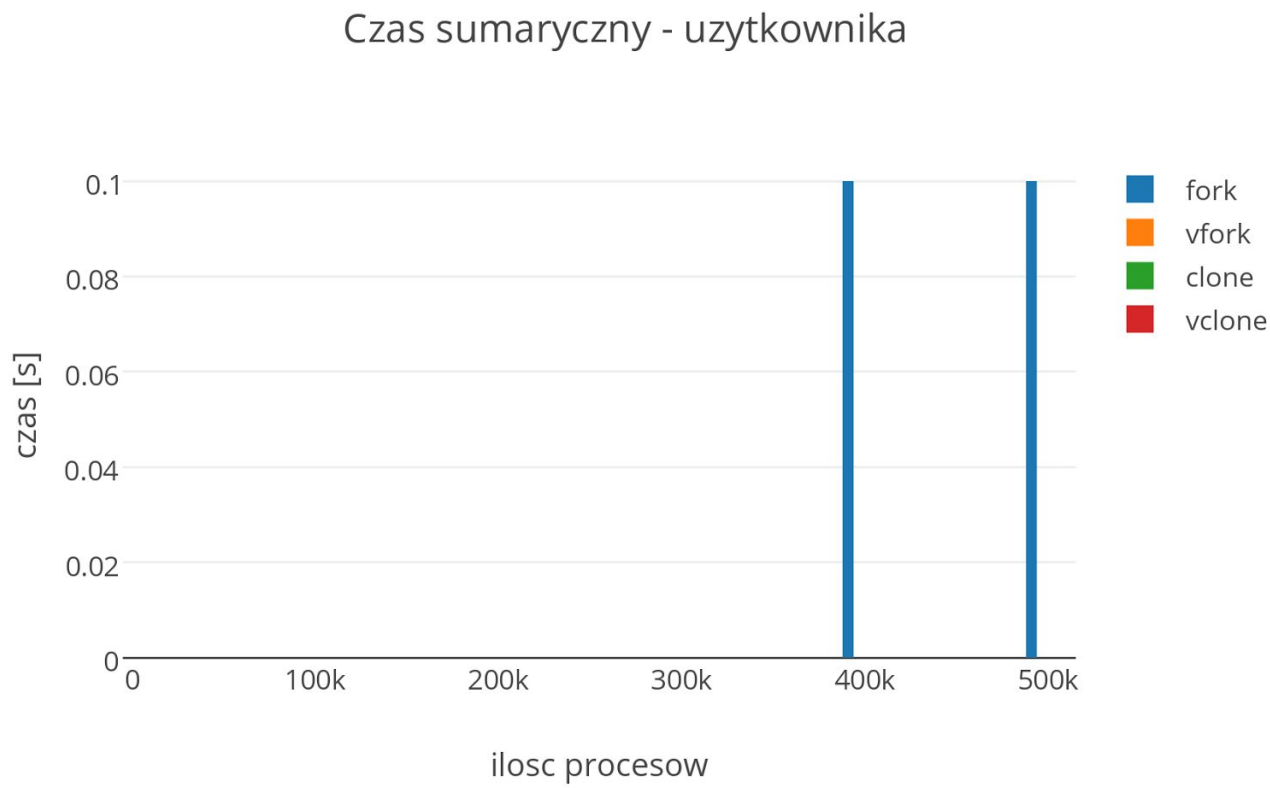

Procesy potomne - czas rzeczywisty
fork
vfork
clone
vclone
50
40
30
20
10
0
czas [s]
0
100k
200k
300k
400k
500k
ilość procesów

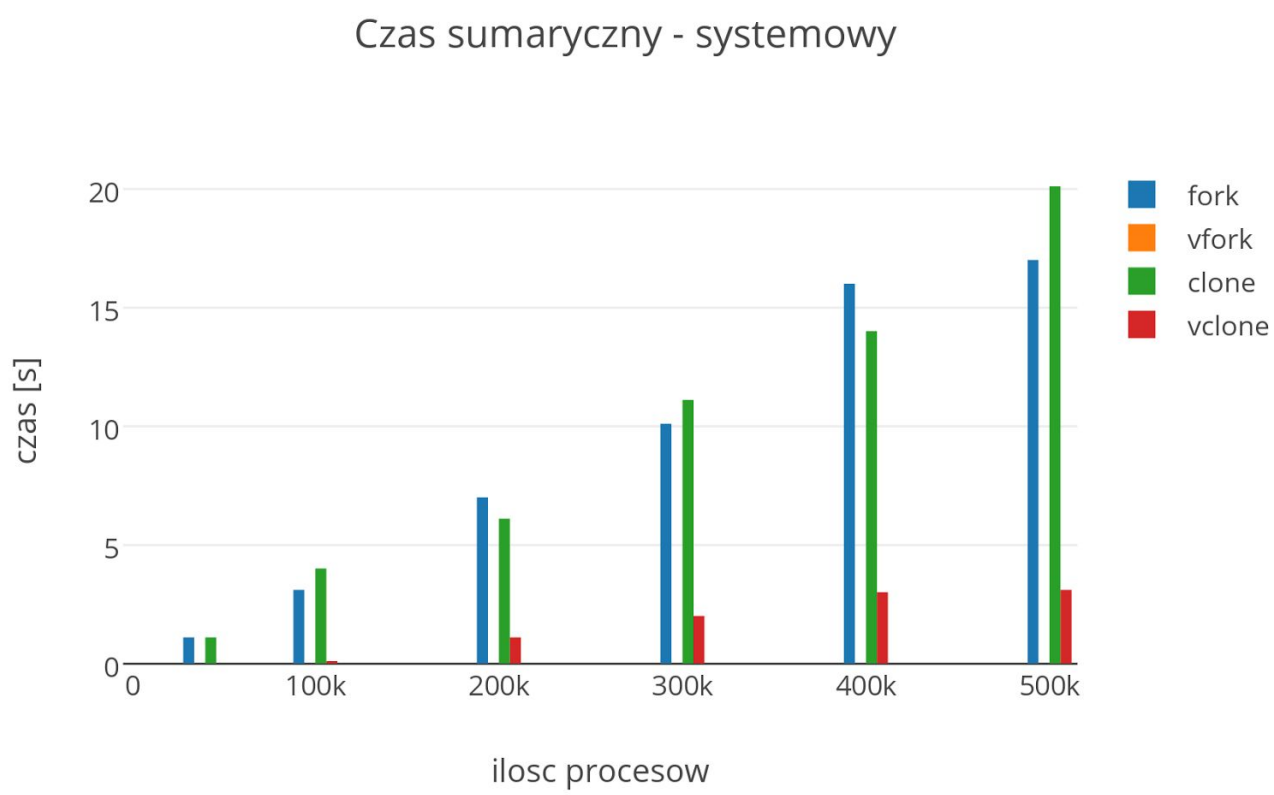

Procesy potomne - czas użytkownika
fork
vfork
clone
vclone
1
0.5
0
−0.5
−1
czas [s]
0
100k
200k
300k
400k
500k
ilość procesów

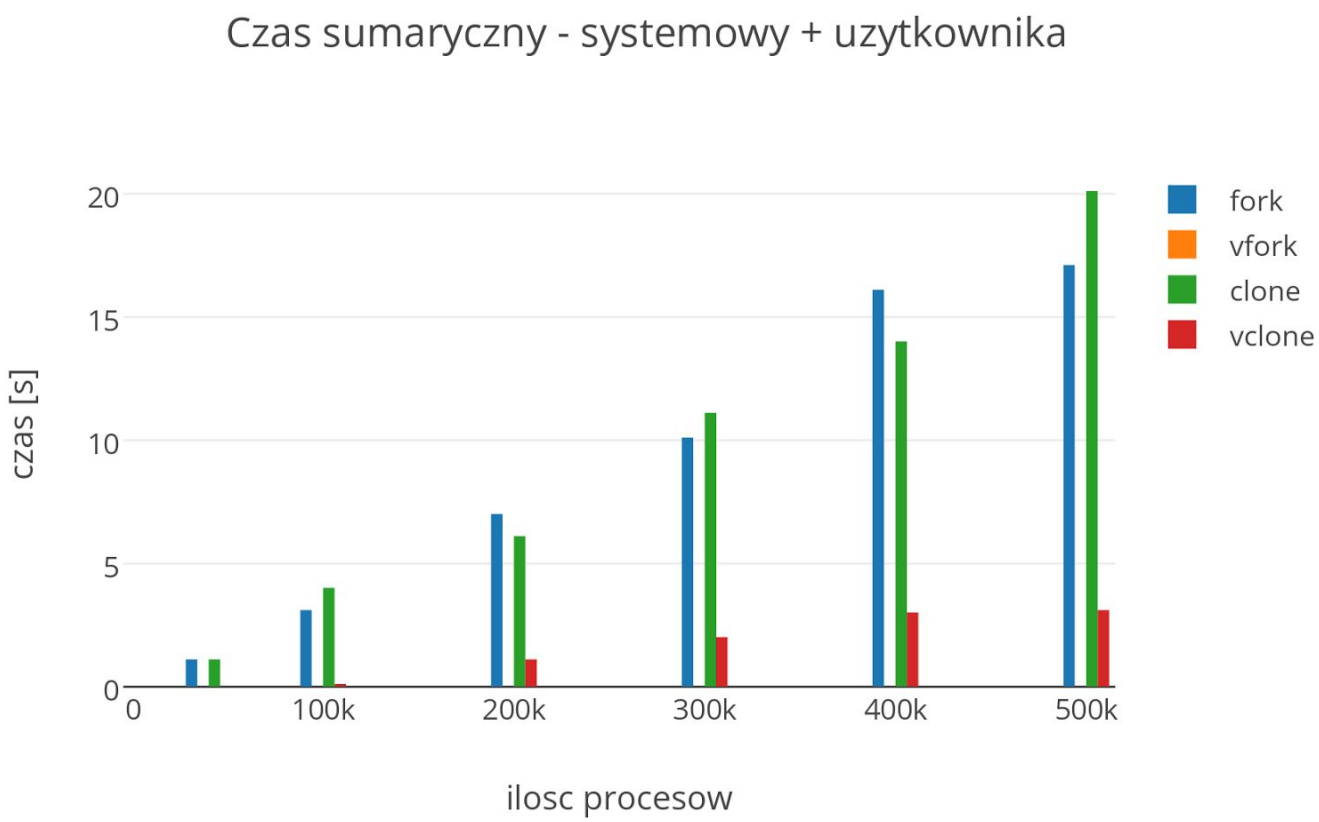

Procesy potomne - czas systemowy
1
0.5
0
−0.5
−1
czas [s]
0
100k
200k
300k
400k
500k
ilość procesów
fork
vfork
clone
vclone


Procesy potomne - czas systemowy + użytkownika
1
0.5
0
−0.5
−1
czas [s]
0
100k
200k
300k
400k
500k
ilość procesów
fork
vfork
clone
vclone


Czas sumaryczny - rzeczywisty
80
60
40
20
0
czas [s]
0
100k
200k
300k
400k
500k
ilosc procesow
fork
vfork
clone
vclone


Czas sumaryczny - uzytkownika
0.1
0.08
0.06
0.04
0.02
0
czas [s]
0
100k
200k
300k
400k
500k
ilosc procesow
fork
vfork
clone
vclone


Czas sumaryczny - systemowy
20
15
10
5
0
czas [s]
0
100k
200k
300k
400k
500k
ilosc procesow
fork
vfork
clone
vclone


Czas sumaryczny - systemowy + uzytkownika
20
15
10
5
0
czas [s]
0
100k
200k
300k
400k
500k
ilosc procesow
fork
vfork
clone
vclone

Komentarz:

1. Kopiowanie przy zapisie niestety nie zadziałało - czas dla forka jest nieporównywalnie większy od czasu vforka.
2. Czas wykonania rośnie liniowo w funkcji ilości procesów niezależnie od wybranej metody.
3. Najszybszą metodą jest vfork - jest to naturalnie spowodowane faktem, że nie kopiuje ona danych procesu macierzystego.
4. Wyniki dla forka i clone'a z flagą SIGCHILD zwracają podobne wyniki - jest to jak najbardziej zgodne ze standardem, gdyż:

Since version 2.3.3, rather than invoking the kernel's **fork**() system call, the glibc **fork**() wrapper that is provided as part of the NPTL threading implementation invokes ***clone****(2)* with flags that provide the same effect as the traditional system call. (A call to **fork**() is equivalent to a call to ***clone****(2)* specifying *flags* as just **SIGCHLD**.)