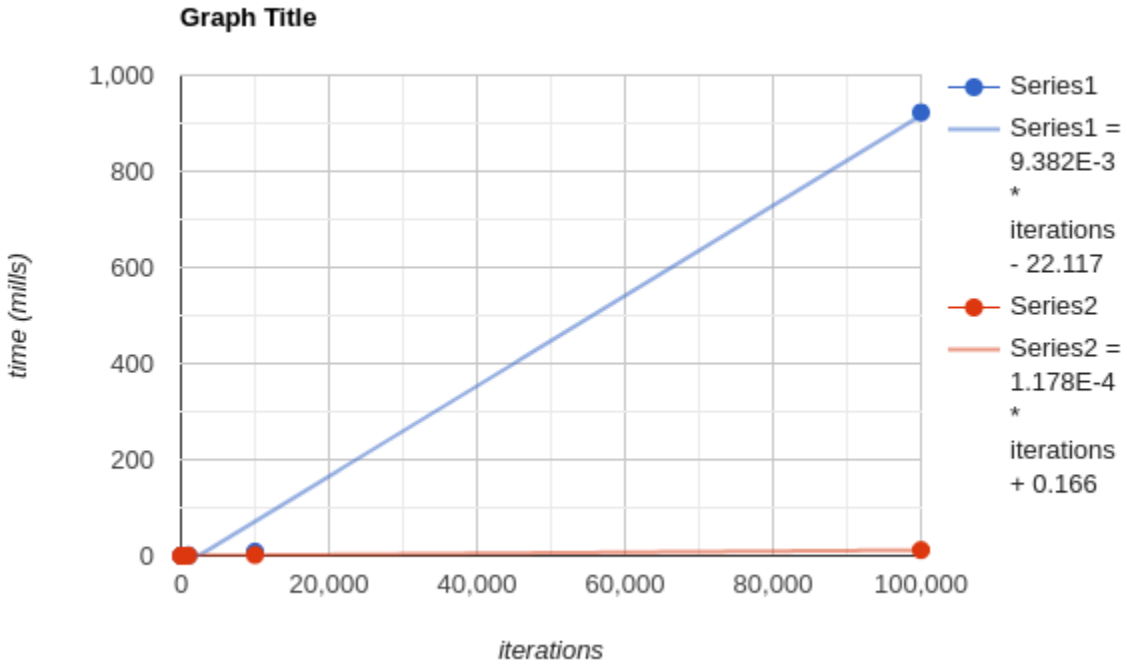
Loops\Programs	Normal C TIme (nano sec)	Bpf C Time (nano sec)
10 (ten)	**33623**	**7882**
100	**109546**	**43703**
1000	**927175**	**194450**
10000	**8542908**	**1768125**
100000	**922254358**	**11907917**
150000		(does not work)
200000		(does not work)
1000000 (million)		(does not work)
10000000		(does not work)
100000000		(does not work)
1000000000 (billion)		(does not work)
10000000000		(does not work)

Loops\Programs	Normal C TIme (milli sec)	Bpf C Time (milli sec)
10 (ten)	**0.033623**	**0.007882**
100	**0.109546**	**0.043703**
1000	**0.927175**	**0.19445**
10000	**8.542908**	**1.768125**
100000	**922.254358**	**11.907917**

https://www.rapidtables.com/tools/scatter-plot.html

Normal c: 10 **0.033623** 100 **0.109546** 1000 **0.927175** 10000 **8.542908** 100000 **922.254358**

BPF: 10 **0.007882** 100 **0.043703** 1000 **0.19445** 10000 **1.768125** 100000 **11.907917**



BPF version is much faster!

If, I don't do new line printing, it put those character in buffer and when buffer gets filled up, that is when it flush and during that flush, it calls syscall. That is why I put \n

Also, getpid() is VDSO (Virtual Dynamic Shared Object) optimized by modern CPU.

How VDSO work?

(i) the kernel maps a portion of memory into the address space of the process, which is calling getcpu()
(i) next time, When any process calls `getcpu()`, the CPU determines that the system call is implemented in the VDSO page. Instead of transitioning to kernel mode, the process executes the `getcpu()` function directly from the VDSO page in user-space.

the kernel maps the VDSO page into the address space of each process during each individual process initialization