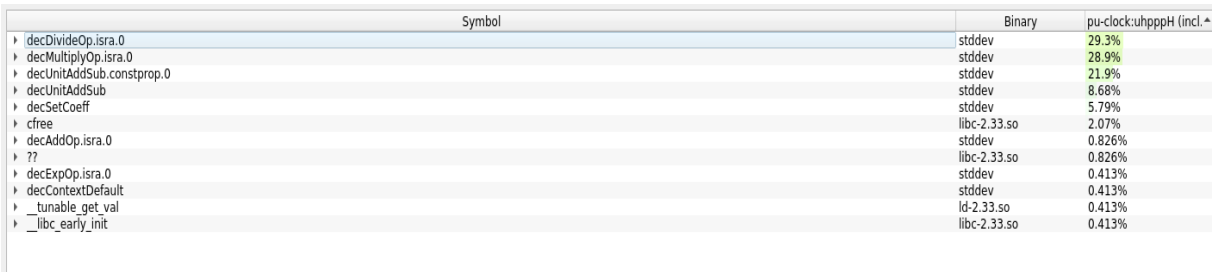
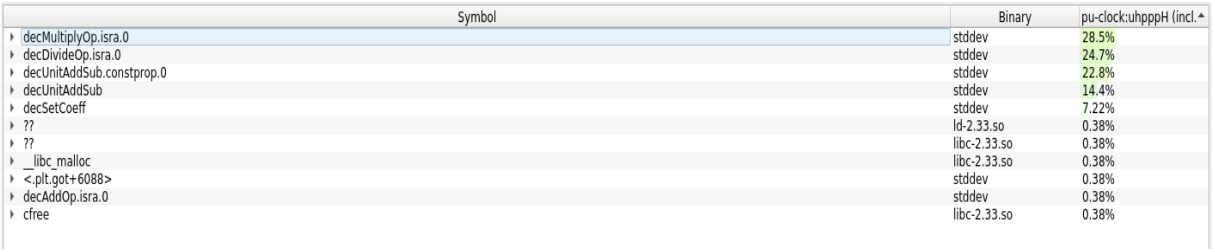
Profiling

10 hodnot



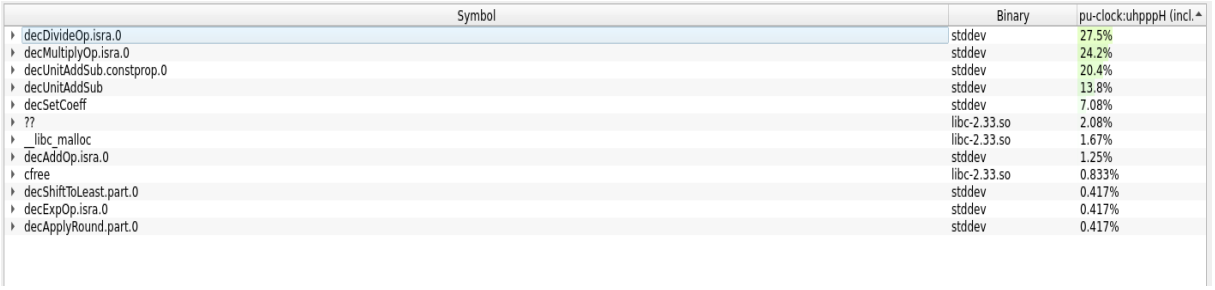
Symbol	Binary	pu-clock:uhpppH (incl.
decDivideOp.isra.0	stddev	29.3%
decMultiplyOp.isra.0	stddev	28.9%
decUnitAddSub.constprop.0	stddev	21.9%
decUnitAddSub	stddev	8.68%
decSetCoeff	stddev	5.79%
cfree	libc-2.33.so	2.07%
decAddOp.isra.0	stddev	0.826%
??	libc-2.33.so	0.826%
decExpOp.isra.0	stddev	0.413%
decContextDefault	stddev	0.413%
__tunable_get_val	ld-2.33.so	0.413%
__libc_early_init	libc-2.33.so	0.413%

100 hodnot



Symbol	Binary	pu-clock:uhpppH (incl.
decMultiplyOp.isra.0	stddev	28.5%
decDivideOp.isra.0	stddev	24.7%
decUnitAddSub.constprop.0	stddev	22.8%
decUnitAddSub	stddev	14.4%
decSetCoeff	stddev	7.22%
??	ld-2.33.so	0.38%
??	libc-2.33.so	0.38%
__libc_malloc	libc-2.33.so	0.38%
<.plt.got+6088>	stddev	0.38%
decAddOp.isra.0	stddev	0.38%
cfree	libc-2.33.so	0.38%

1000 hodnot



Symbol	Binary	pu-clock:uhpppH (incl.
decDivideOp.isra.0	stddev	27.5%
decMultiplyOp.isra.0	stddev	24.2%
decUnitAddSub.constprop.0	stddev	20.4%
decUnitAddSub	stddev	13.8%
decSetCoeff	stddev	7.08%
??	libc-2.33.so	2.08%
__libc_malloc	libc-2.33.so	1.67%
decAddOp.isra.0	stddev	1.25%
cfree	libc-2.33.so	0.833%
decShiftToLeast.part.0	stddev	0.417%
decExpOp.isra.0	stddev	0.417%
decApplyRound.part.0	stddev	0.417%

Na snímcích výše můžeme vidět výstup profilování funkce na výpočet směrodatné odchylky při 10, 100 a 1000 vstupních hodnot.

Funkce tráví nejvíce času používáním „multiply“ a „divide“. Což bude způsobeno složitější prací s daty oproti jednodušším funkcím sčítání a odčítání, ve kterých ale funkce tráví také celkem dost času vzhledem k jednoduchosti práce s daty.

Při optimalizaci kódu bych se zaměřil na vyladění všech 4 základních aritmetických operací, nebo na práci s daty což budou mít funkce společné.