BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

NGLYD CALCULATOR – IVS Project

Profiling report

xfignam00
xmalegt00
xnovakf00
xskovaj00

April 17, 2024

Introduction

This report has been created for second project from IVS course. It contains profiling information for program calculating standard deviation from a set of numbers, specifications of hardware and software used for profiling and summary of what could be improved in the program for faster execution.

1 Specifications

1.1 Machine

OS: `Ubuntu 22.04.3 LTS on Windows 10 x86_64`
Kernel: `5.15.146.1-microsoft-standard-WSL2`
Shell: `bash 5.1.16`
CPU: `11th Gen Intel i9-11900H (16) @ 2.496GHz`
GPU: `d0ba:00:00.0 Microsoft Corporation Device 008e`
Memory: `7786MiB`

1.2 Profiling software

`valgrind-3.18.1`
`callgrind-3.18.1`
`kcachegrind-21.12.3`

2 Program

Program used for calculating the standard deviation is implemented in `C++` source file called `stddev.cpp`. It uses math library which is specified in `math_operations.h`. Program takes a sequence of numbers from standard input and prints standard deviation to standard output which is calculated with these formulas:

$$s = \sqrt{\frac{1}{N-1}\left(\sum_{i=1}^{N} x_i^2 - N\bar{x}^2\right)}$$

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

and is executed as so:

```
./stddev <input.txt
```

3 Profiling

Profiling uses `valgrind`, `callgrind` and for visualisation of the output `KCachegrind`. Program is profiled with 3 different input files with sizes of 10, 10^3 and 10^6 values. These are in their respective text files (`data10.txt, data1000.txt, data1mil.txt`).
Execution of profiling is following (shown for only one file):

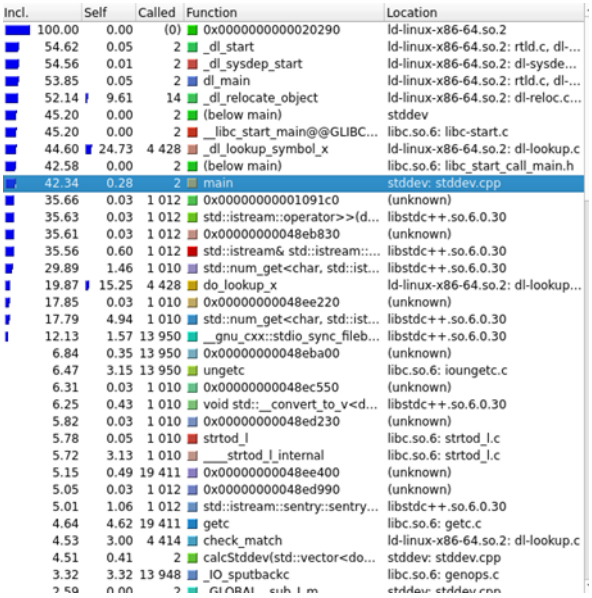
```
valgrind --tool=callgrind ./stddev <data10.txt [1]
kcachegrind callgrind.out.10
```

[1] This will produce file `callgrind.out.<id>`.

4 Profiling outputs

Screenshots from `KCachegrind` program show a table of functions sorted by relative time they take. Other screenshots depict function trees with numbers how much relative time each take and how many times they are executed.

4.1 data10.txt



Incl.	Self	Called	Function	Location
100.00	0.00	(0)	0x0000000000020290	ld-linux-x86-64.so.2
54.62	0.05	2	_dl_start	ld-linux-x86-64.so.2: rtld.c, dl-...
54.56	0.01	2	_dl_sysdep_start	ld-linux-x86-64.so.2: dl-sysde...
53.85	0.05	2	dl_main	ld-linux-x86-64.so.2: rtld.c, dl-...
52.14	9.61	14	_dl_relocate_object	ld-linux-x86-64.so.2: dl-reloc.c...
45.20	0.00	2	(below main)	stddev
45.20	0.00	2	__libc_start_main@@GLIBC...	libc.so.6: libc-start.c
44.60	24.73	4 428	_dl_lookup_symbol_x	ld-linux-x86-64.so.2: dl-lookup.c
42.58	0.00	2	(below main)	libc.so.6: libc_start_call_main.h
42.34	0.28	2	main	stddev: stddev.cpp
35.66	0.03	1 012	0x00000000001091c0	(unknown)
35.63	0.03	1 012	std::istream::operator>>(d...	libstdc++.so.6.0.30
35.61	0.03	1 012	0x00000000048eb830	(unknown)
35.56	0.60	1 012	std::istream& std::istream::...	libstdc++.so.6.0.30
29.89	1.46	1 010	std::num_get<char, std::ist...	libstdc++.so.6.0.30
19.87	15.25	4 428	do_lookup_x	ld-linux-x86-64.so.2: dl-lookup...
17.85	0.03	1 010	0x00000000048ee220	(unknown)
17.79	4.94	1 010	std::num_get<char, std::ist...	libstdc++.so.6.0.30
12.13	1.57	13 950	__gnu_cxx::stdio_sync_fileb...	libstdc++.so.6.0.30
6.84	0.35	13 950	0x00000000048eba00	(unknown)
6.47	3.15	13 950	ungetc	libc.so.6: iougetc.c
6.31	0.03	1 010	0x00000000048ec550	(unknown)
6.25	0.43	1 010	void std::__convert_to_v<d...	libstdc++.so.6.0.30
5.82	0.03	1 010	0x00000000048ed230	(unknown)
5.78	0.05	1 010	strtod_l	libc.so.6: strtod_l.c
5.72	3.13	1 010	___strtod_l_internal	libc.so.6: strtod_l.c
5.15	0.49	19 411	0x00000000048ee400	(unknown)
5.05	0.03	1 012	0x00000000048ed990	(unknown)
5.01	1.06	1 012	std::istream::sentry::sentry...	libstdc++.so.6.0.30
4.64	4.62	19 411	getc	libc.so.6: getc.c
4.53	3.00	4 414	check_match	ld-linux-x86-64.so.2: dl-lookup.c
4.51	0.41	2	calcStddev(std::vector<do...	stddev: stddev.cpp
3.32	3.32	13 948	_IO_sputbackc	libc.so.6: genops.c
2.59	0.00	2	__GLOBAL__sub_I_m	stddev: stddev.cpp

Figure 1: Sorted data for `data10.txt`

As we can see, with 10 values, our `main` function of program takes 42.32% of the whole run, which is less than a half. The other half is automatic setup of the environment.

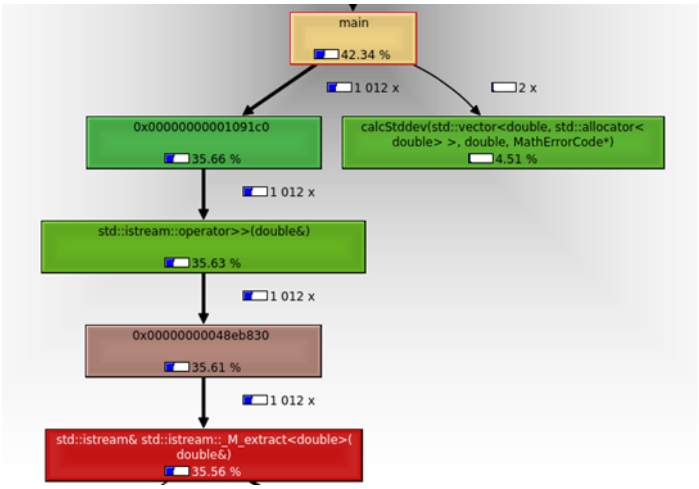


Figure 2: Function tree for `main`, `data10.txt`

In this tree it is clearly visible that parsing the data from input into memory for further manipulation (left) takes significantly more time than actual calculation of the standard deviation (right).

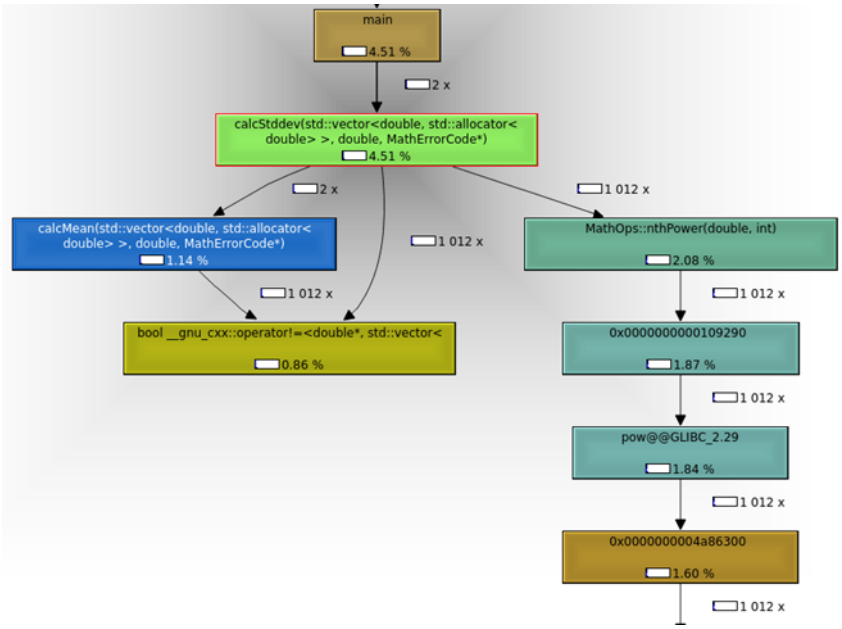


Figure 3: Function tree for `calcStddev`, `data10.txt`

Quite surprisingly, calculation of mean of the numbers (left) takes less time than calculating the power (right) of each of the numbers.

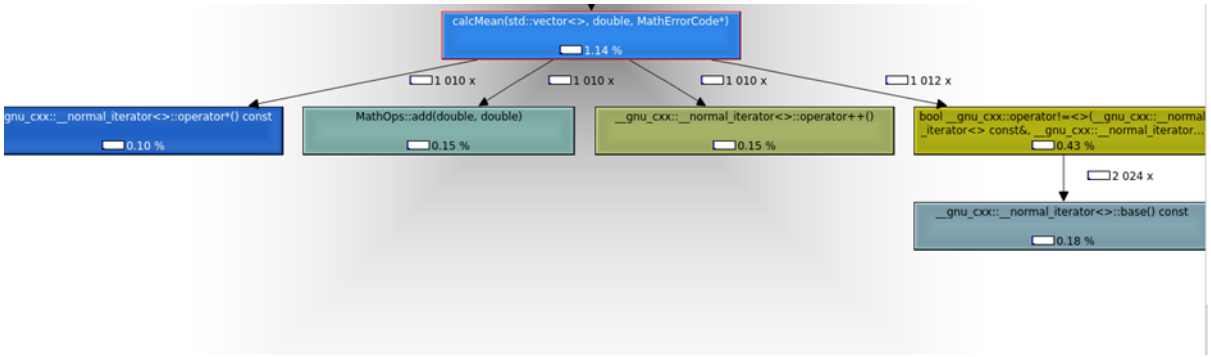


Figure 4: Function tree for `calcMean`, `data10.txt`

Fuction `add` is not as time consuming as function used for iterating through a vector.

In conclusion, with data file `data10.txt`, the biggest time consumers are functions for manipulating with the numbers and iterating through vectors. The only function from out math library that was slower than expected was `nthPower`.

4.2 data1000.txt

Incl.	Self	Called	Function	Location
100.00	0.00	(0)	0x0000000000020290	ld-linux-x86-64.so.2
60.99	0.00	1	(below main)	stddev
60.99	0.00	1	__libc_start_main@@GLIBC...	libc.so.6: libc-start.c
59.13	0.00	1	(below main)	libc.so.6: libc_start_call_main.h
58.96	0.39	1	main	stddev: stddev.cpp
49.96	0.04	1 001	0x00000000001091c0	(unknown)
49.93	0.04	1 001	std::istream::operator>>(d...	libstdc++.so.6.0.30
49.89	0.04	1 001	0x00000000048eb830	(unknown)
49.84	0.84	1 001	std::istream& std::istream::...	libstdc++.so.6.0.30
41.95	2.05	1 000	std::num_get<char, std::ist...	libstdc++.so.6.0.30
38.88	0.04	1	_dl_start	ld-linux-x86-64.so.2: rtld.c, dl-machine.h, ...
38.84	0.01	1	_dl_sysdep_start	ld-linux-x86-64.so.2: dl-sysdep.c, dl-sysde...
38.33	0.03	1	dl_main	ld-linux-x86-64.so.2: rtld.c, dl-prop.h, get-...
37.11	6.84	7	_dl_relocate_object	ld-linux-x86-64.so.2: dl-reloc.c, dl-machin...
31.75	17.60	2 214	_dl_lookup_symbol_x	ld-linux-x86-64.so.2: dl-lookup.c
25.11	0.04	1 000	0x00000000048ee220	(unknown)
25.05	6.95	1 000	std::num_get<char, std::ist...	libstdc++.so.6.0.30
17.05	2.22	13 811	__gnu_cxx::stdio_sync_fileb...	libstdc++.so.6.0.30
14.15	10.85	2 214	do_lookup_x	ld-linux-x86-64.so.2: dl-lookup.c, dl-protec...
9.63	0.49	13 811	0x00000000048eba00	(unknown)
9.12	4.44	13 811	ungetc	libc.so.6: iungetc.c
8.85	0.04	1 000	0x00000000048ec550	(unknown)
8.80	0.61	1 000	void std::__convert_to_v<d...	libstdc++.so.6.0.30
8.19	0.04	1 000	0x00000000048ed230	(unknown)
8.14	0.07	1 000	strtod_l	libc.so.6: strtod_l.c
8.07	4.42	1 000	___strtod_l_internal	libc.so.6: strtod_l.c
7.23	0.69	19 217	0x00000000048ee400	(unknown)
7.03	0.04	1 001	0x00000000048ed990	(unknown)
6.98	1.50	1 001	std::istream::sentry::sentry...	libstdc++.so.6.0.30
6.53	6.52	19 217	getc	libc.so.6: getc.c
6.33	0.57	1	calcStddev(std::vector<do...	stddev: stddev.cpp
4.68	4.68	13 810	_IO_sputbackc	libc.so.6: genops.c
3.22	2.13	2 207	check_match	ld-linux-x86-64.so.2: dl-lookup.c
2.96	1.07	1 532	0x000000000012faf0	libstdc++.so.6.0.30
2.93	0.30	1 001	MathOps::nthPower(double...	stddev: math_operations.cpp
2.80	0.77	5 406	__gnu_cxx::stdio_sync_fileb...	libstdc++.so.6.0.30
2.63	0.04	1 001	0x0000000000109290	(unknown)
2.59	0.34	1 001	pow@@GLIBC_2.29	libm.so.6: w_pow_template.c
2.56	0.04	1 000	0x00000000048eab00	(unknown)

Figure 5: Sorted data for `data1000.txt`

With bigger amount of numbers the percentage of time needed for `main` function has increased and now takes more time than setup of environment. However, this change is not as significant as could be expected with dataset larger $10^2\times$.

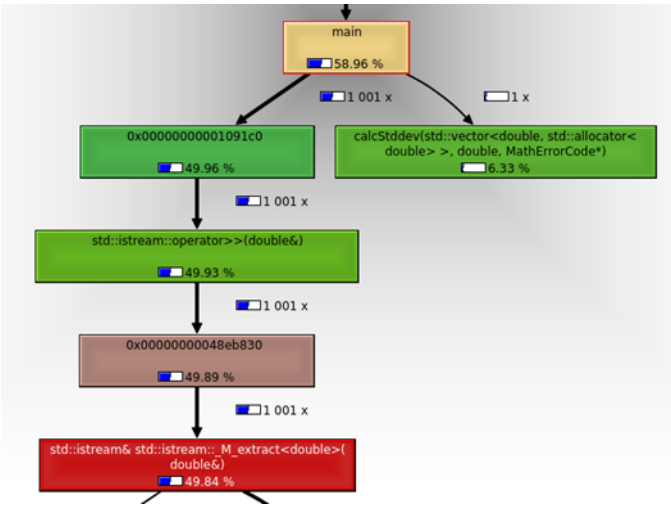


Figure 6: Function tree for `main`, `data1000.txt`

The increase in time needed for parsing the input numbers is noteworthy in comparison with time needed for calculating the standard deviation.

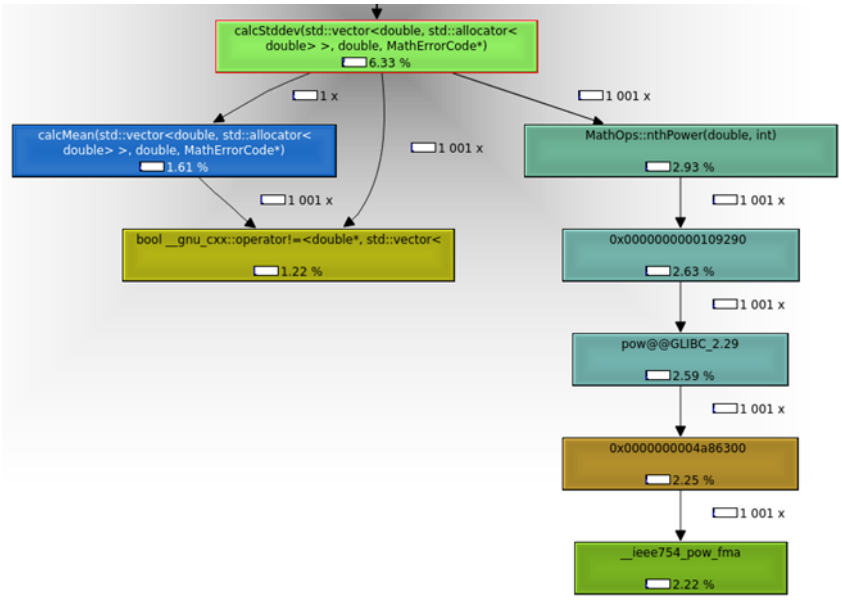


Figure 7: Function tree for `calcStddev`, `data1000.txt`

Difference in time for `calcMean` and `nthPower` is not much larger than in previous scenario with smaller dataset. In this function, there are no significant changes.

In conclusion, with data file `data1000.txt`, the `main` function was slower than the setup of the environment. This was mainly due the need of loading more numbers and storing them in vectors. There were no other surprising changes.

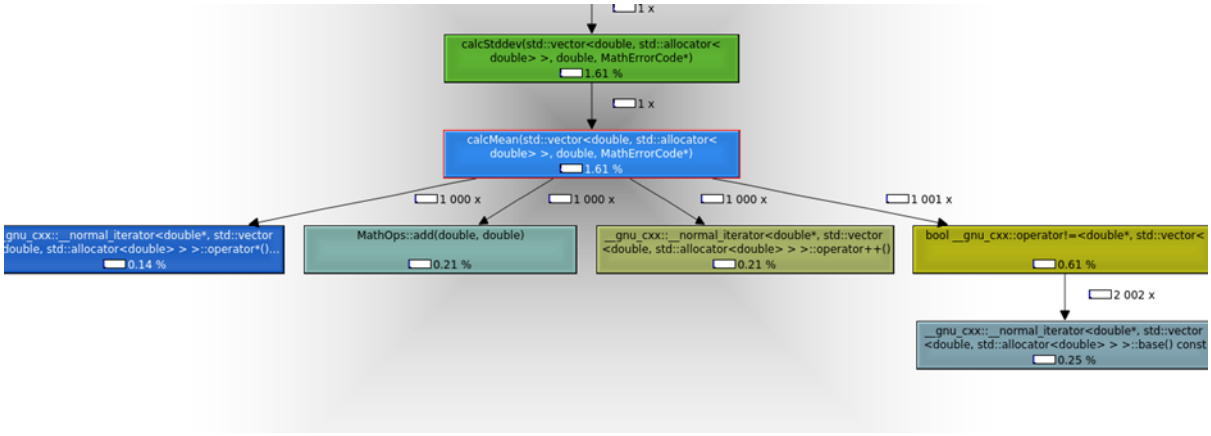


Figure 8: Function tree for `calcMean`, `data1000.txt`

4.3 data1mil.txt

Incl.	Self	Called	Function	Location
100.00	0.00	(0)	0x0000000000020290	ld-linux-x86-64.so.2
99.93	0.00	1	(below main)	stddev
99.93	0.00	1	__libc_start_main@@GLIBC...	libc.so.6: libc-start.c
99.93	0.00	1	(below main)	libc.so.6: libc_start_call_main.h
99.93	0.68	1	main	stddev: stddev.cpp
85.40	0.06	1 000 001	0x00000000001091c0	(unknown)
85.33	0.06	1 000 001	std::istream::operator>>(d...	libstdc++.so.6.0.30
85.27	0.06	1 000 001	0x00000000048eb830	(unknown)
85.21	1.45	1 000 001	std::istream& std::istream::...	libstdc++.so.6.0.30
71.80	3.54	1 000 000	std::num_get<char, std::ist...	libstdc++.so.6.0.30
43.01	0.06	1 000 000	0x00000000048ee220	(unknown)
42.95	11.99	1 000 000	std::num_get<char, std::ist...	libstdc++.so.6.0.30
29.24	3.81	13 779 965	__gnu_cxx::stdio_sync_fileb...	libstdc++.so.6.0.30
16.52	0.85	13 779 965	0x00000000048eba00	(unknown)
15.68	7.63	13 779 965	ungetc	libc.so.6: ioungetc.c
15.20	0.06	1 000 000	0x00000000048ec550	(unknown)
15.14	1.05	1 000 000	void std::__convert_to_v<d...	libstdc++.so.6.0.30
14.09	0.06	1 000 000	0x00000000048ed230	(unknown)
14.03	0.12	1 000 000	strtod_l	libc.so.6: strtod_l.c
13.91	7.61	1 000 000	___strtod_l_internal	libc.so.6: strtod_l.c
12.38	1.18	19 169 948	0x00000000048ee400	(unknown)
11.96	0.06	1 000 001	0x00000000048ed990	(unknown)
11.90	2.58	1 000 001	std::istream::sentry::sentry...	libstdc++.so.6.0.30
11.21	11.20	19 169 948	getc	libc.so.6: getc.c
10.82	0.98	1	calcStddev(std::vector<do...	stddev: stddev.cpp
8.05	8.05	13 779 964	_IO_sputbackc	libc.so.6: genops.c
5.01	0.52	1 000 001	MathOps::nthPower(double...	stddev: math_operations.cpp
4.98	1.80	1 500 885	0x000000000012faf0	libstdc++.so.6.0.30
4.81	1.33	5 389 983	__gnu_cxx::stdio_sync_fileb...	libstdc++.so.6.0.30
4.49	0.06	1 000 001	0x0000000000109290	(unknown)
4.43	0.58	1 000 001	pow@@GLIBC_2.29	libm.so.6: w_pow_template.c
4.30	0.06	1 000 000	0x00000000048eab00	(unknown)
4.24	0.92	1 000 000	std::string::reserve(unsign...	libstdc++.so.6.0.30
4.18	0.06	1 000 004	0x00000000048ec770	(unknown)
4.12	1.17	1 000 004	std::ostream::flush()	libstdc++.so.6.0.30
4.02	0.83	1 500 885	0x0000000000103500	libstdc++.so.6.0.30
3.84	0.06	1 000 001	0x0000000004a86300	(unknown)
3.78	3.78	1 000 001	__ieee754_pow_fma	libm.so.6: e_pow.c, math_config.h
3.32	0.06	1 000 000	0x00000000048ec5b0	(unknown)

Figure 9: Sorted data for `data1mil.txt`

Our `main` function now takes 99.93% of the total time needed for program to run. This is substantial rise from previous scenario, although can be expected, as the dataset is $10^3\times$ larger.

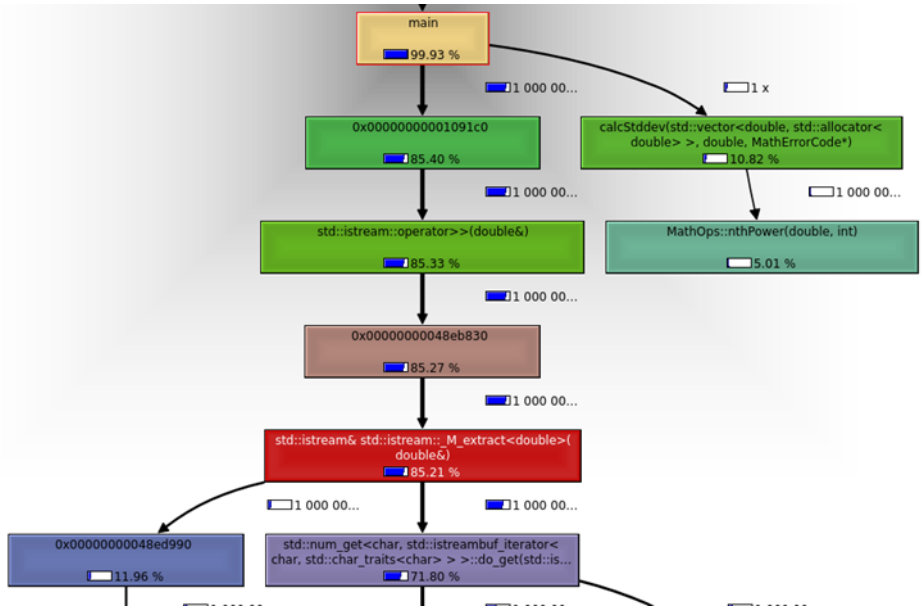


Figure 10: Function tree for `main`, `1mil.txt`

Parsing individual numbers still takes more time than actual calculation of the deviation.

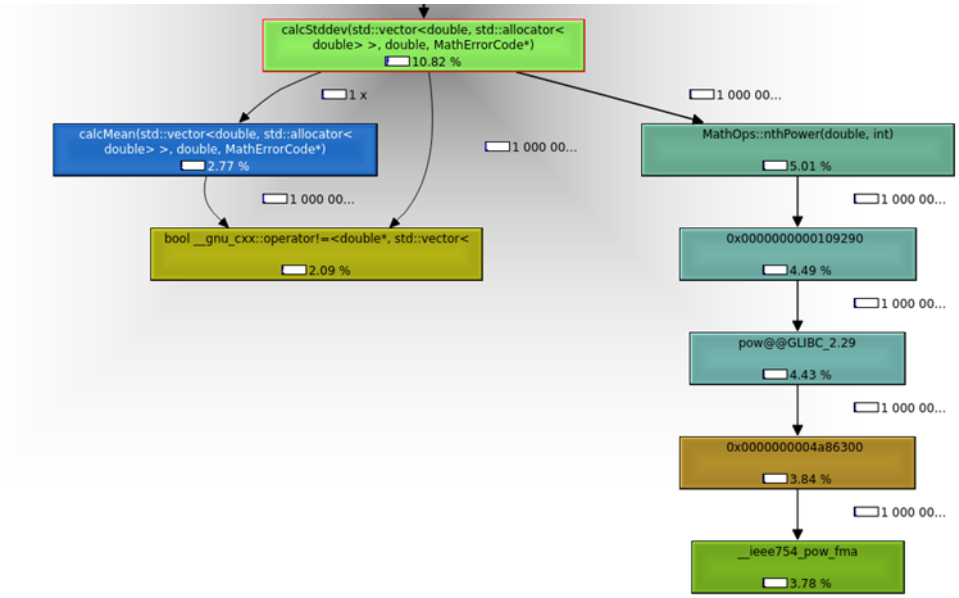


Figure 11: Function tree for `calcStddev`, `data1mil.txt`

Function `nthPower` continues to take more time than `meanCalc`. No significant changes.

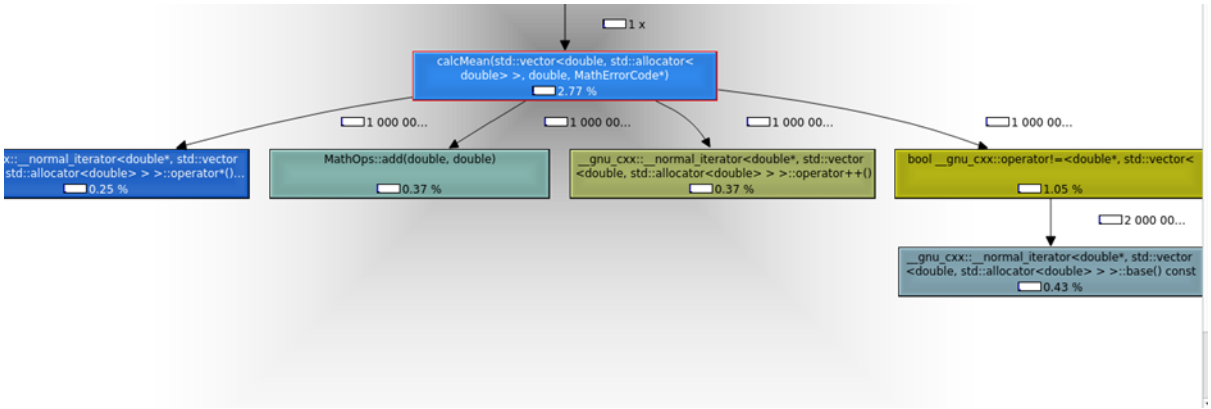


Figure 12: Function tree for `calcMean`, `data1mil.txt`

No significant changes.

In conclusion, with data file `data1mil.txt`, the time for `main` function had risen because of the increase in input data. Layout of other functions was not effected, time for parsing the numbers still takes the most time.

4.4 data10.txt, data1000.txt, data1mil.txt in comparison

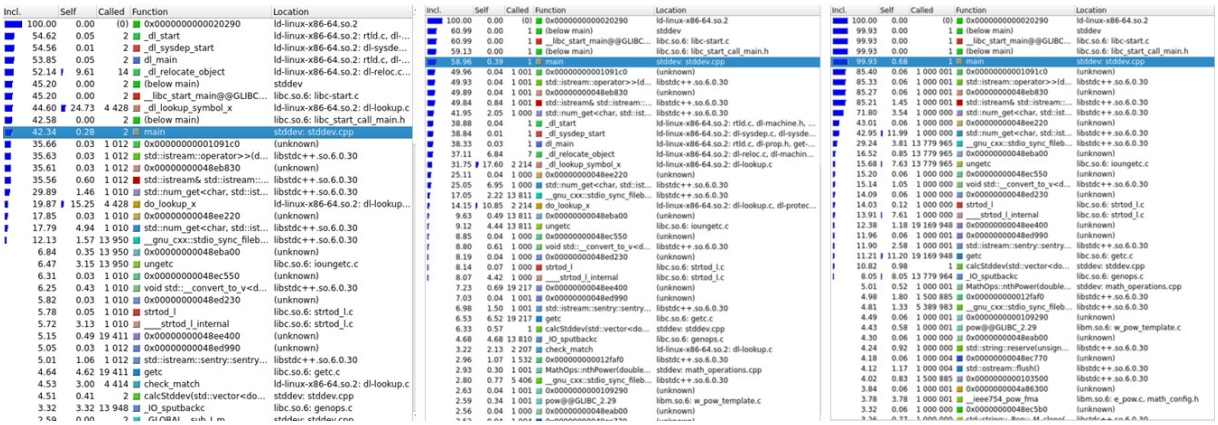


Figure 13: Comparison between all scenarios

Conclusion

To conclude everything that has been stated, it is evident that the most time consuming functions are the ones used for processing inputs. This is why it should be the main focus when the time for optimalisation of `stddev.cpp` would come. Another function that might be optimalised would be `nthPower`, although it does not affect the time that much as previously stated functions.