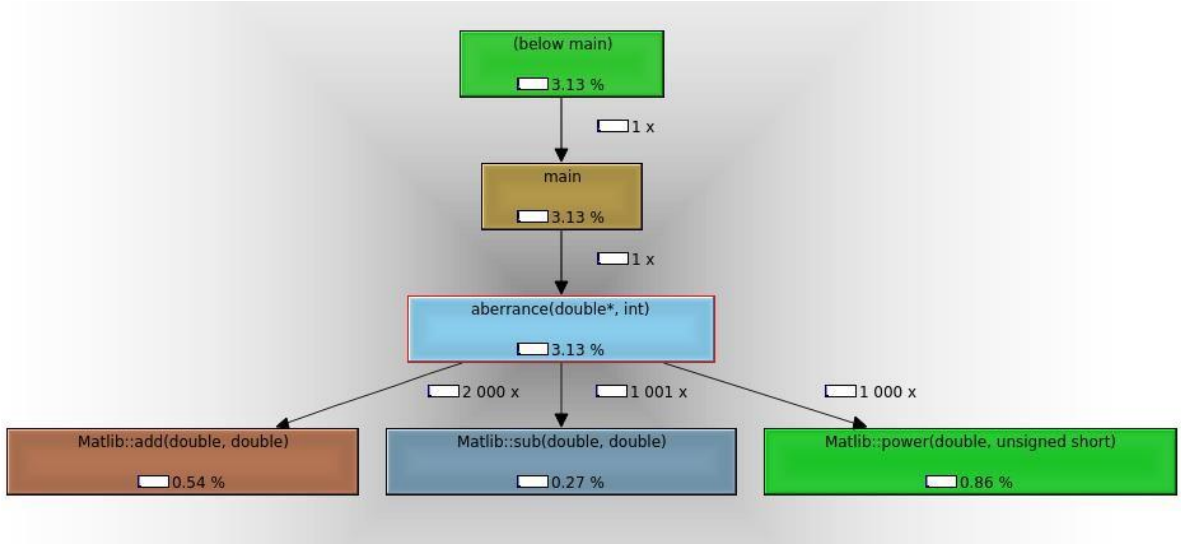
Pro pro 1000 čísel:



1- graf volání funkcí z matlib.h

```
28        double aberrance( double numbers[], int numSize )
29  0.00  {
30            Matlib math; //creating a instation of class
31  0.00      long double sum = 0;
32  0.00      double tmp = 0;
33
34  0.13      for ( int i = 0; i < numSize; i++ )
35            {
36                //count average
37  0.19          tmp = numbers[i];
38  0.30          sum = math.add(sum, tmp);
    0.27  1000 call(s) to 'Matlib::add(double, double)' (stddeviation: matlib.cpp)
39            }
40  0.00      double average = math.div(sum,numSize);
    0.00  1 call(s) to 'Matlib::div(double, double)' (stddeviation: matlib.cpp)
41
42  0.00      long double sumOfNumbers = 0;
43
44  0.13      for ( int j = 0; j < numSize; j++ )
45            {
46                // sumOfNumbers = sumOfNumbers + (( x - average)^2)
47  0.67          sumOfNumbers =  math.add( ( math.power( math.sub( numbers[j], average ), 2) ), sumOfNumbers);
    0.27  1000 call(s) to 'Matlib::add(double, double)' (stddeviation: matlib.cpp)
    0.27  1000 call(s) to 'Matlib::sub(double, double)' (stddeviation: matlib.cpp)
    0.86  1000 call(s) to 'Matlib::power(double, unsigned short)' (stddeviation: matlib.cpp)
48            }
49            // total = 1 / (N-1) * sumOfNumbers
50  0.00      double total = 0;
51  0.00      total = math.mul( sumOfNumbers, math.div( 1, math.sub( numSize, 1 ) ));
    0.00  1 call(s) to 'Matlib::mul(double, double)' (stddeviation: matlib.cpp)
    0.00  1 call(s) to 'Matlib::sub(double, double)' (stddeviation: matlib.cpp)
    0.00  1 call(s) to 'Matlib::div(double, double)' (stddeviation: matlib.cpp)
52  0.00      double result = math.root(total, 2);
    0.03  1 call(s) to 'Matlib::root(double, unsigned short)' (stddeviation: matlib.cpp)
53  0.00      return result;
54  0.00  }
55
```

2- detail volání funkcí z matlib.h zobrazený v kódu

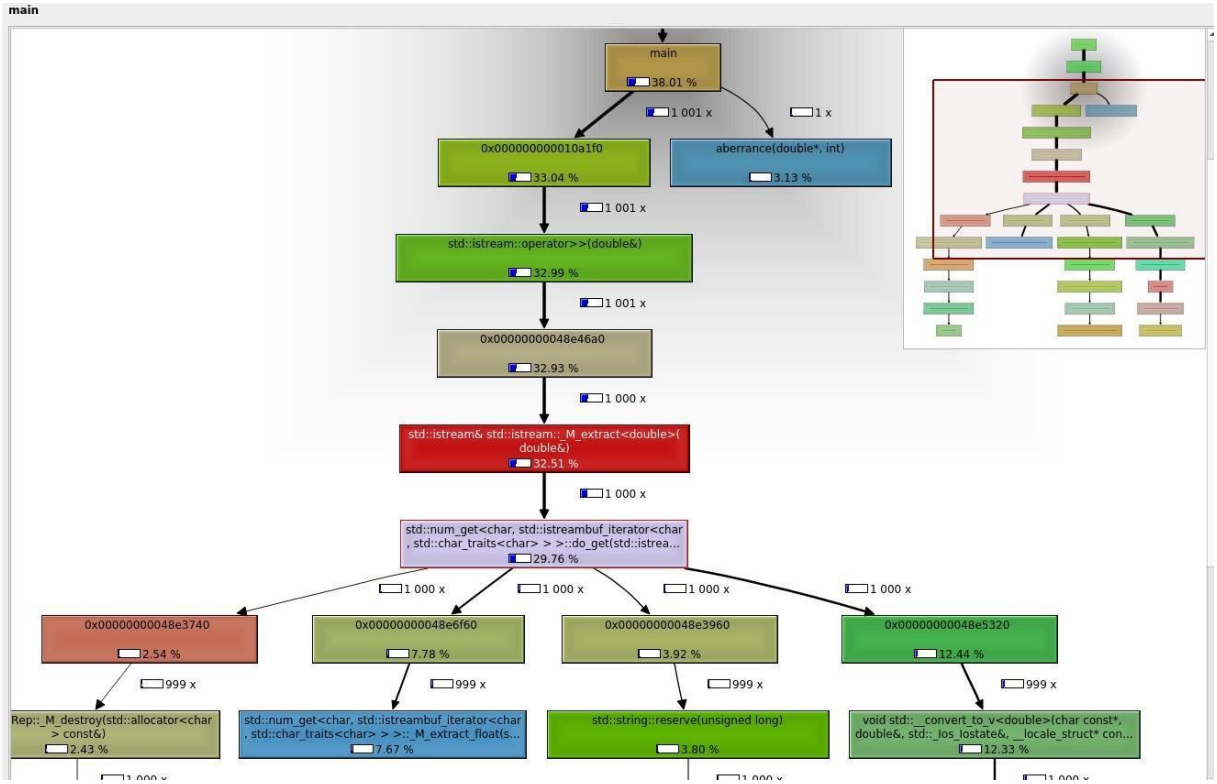
Incl.	Self	Called	Function	Location
100.00	0.00	(0)	0x0000000000001100	ld-2.31.so
56.73	0.03	1	_dl_start	ld-2.31.so: rtld.c, dl-machine.h, get-dynamic-info.h, do-rel.h
56.70	0.02	1	_dl_sysdep_start	ld-2.31.so: dl-sysdep.c, dl-sysdep.c, cpu-features.c, cpu-features.c
56.36	0.04	1	dl_main	ld-2.31.so: rtld.c, dl-prop.h, get-dynamic-info.h, setup-vdso.h, dl-vdso.h, dl-vdso-setup.h, dl-osinfo.h
54.95	10.17	7	_dl_relocate_object	ld-2.31.so: dl-reloc.c, dl-machine.h, do-rel.h, ldsodefs.h
47.26	26.17	2 244	_dl_lookup_symbol_x	ld-2.31.so: dl-lookup.c
41.02	0.00	1	_start	stddeviation
41.02	0.00	1	(below main)	libc-2.31.so: libc-start.c
38.01	0.62	1	main	stddeviation: stddeviation.cpp
33.04	0.05	1 001	0x000000000010a1f0	(unknown)
32.99	0.05	1 001	std::istream::operator>>(d...	libstdc++.so.6.0.28
32.93	0.05	1 001	0x00000000048e46a0	(unknown)
32.51	1.27	1 000	std::istream& std::istream::...	libstdc++.so.6.0.28
29.76	2.75	1 000	std::num_get<char, std::ist...	libstdc++.so.6.0.28
21.08	15.70	2 244	do_lookup_x	ld-2.31.so: dl-lookup.c, ldsodefs.h
12.44	0.05	1 000	0x00000000048e5320	(unknown)
12.33	0.92	999	void std::__convert_to_v<d...	libstdc++.so.6.0.28
11.44	0.05	1 000	0x00000000048e5f30	(unknown)
11.36	0.11	999	strtod_l	libc-2.31.so: strtod_l.c
11.26	6.16	1 000	___strtod_l_internal	libc-2.31.so: strtod_l.c
7.78	0.05	1 000	0x00000000048e6f60	(unknown)
7.67	5.57	999	std::num_get<char, std::ist...	libstdc++.so.6.0.28
7.62	0.11	100	_dl_runtime_resolve_xsave'2	ld-2.31.so: dl-trampoline.h
6.24	0.02	21	_dl_runtime_resolve_xsave	ld-2.31.so: dl-trampoline.h
5.27	3.26	2 229	check_match	ld-2.31.so: dl-lookup.c
3.92	0.05	1 000	0x00000000048e3960	(unknown)
3.80	0.86	999	std::string::reserve(unsign...	libstdc++.so.6.0.28
3.25	0.05	1 002	0x00000000048e3f10	(unknown)
3.13	1.43	1	aberrance(double*, int)	stddeviation: stddeviation.cpp
2.99	0.05	1 000	0x00000000048e5380	(unknown)
2.88	0.70	999	std::string::_Rep::_M_clone(...	libstdc++.so.6.0.28
2.84	0.00	1	__libc_csu_init	stddeviation
2.84	0.00	1	_GLOBAL__sub_I__Z9aberra...	stddeviation: stddeviation.cpp
2.84	0.00	1	__static_initialization_and_d...	stddeviation: stddeviation.cpp, iostream
2.83	0.00	1	0x000000000010a280	(unknown)
2.83	0.01	1	std::ios_base::Init::Init()	libstdc++.so.6.0.28
2.77	0.26	121	_dl_fixup	ld-2.31.so: dl-runtime.c, dl-machine.h, dl-irel.h
2.54	0.05	1 000	0x00000000048e3740	(unknown)
2.43	0.05	999	std::string::_Rep::_M_destr...	libstdc++.so.6.0.28
2.42	0.05	1 001	0x00000000048e4120	(unknown)
2.40	1.97	1 000	round_and_return	libc-2.31.so: strtod_l.c, get-rounding-mode.h, rounding-mode.h
2.32	0.05	1 000	operator delete(void*)	libstdc++.so.6.0.28
2.31	2.31	2 635	strcmp	ld-2.31.so: strcmp.S
2.29	0.05	1 001	0x00000000048e4980	(unknown)
2.25	0.00	1	_dl_init	ld-2.31.so: dl-init.c
2.25	0.01	7	call_init.part.0	ld-2.31.so: dl-init.c
2.22	0.57	1 001	free	libc-2.31.so: malloc.c
2.21	0.05	1 000	0x00000000048e33a0	(unknown)
2.13	0.59	999	std::string::_Rep::_S_create...	libstdc++.so.6.0.28

callgrind.out.376 [1] - Total Instruction Fetch Cost: 3 706 741

3- přehled všech běžících procesů

```
56          //main function - to read numbers from file
57
58          int main( int argc, char *argv[])
59    0.00  {
60    0.00     ifstream file;
      0.17     1 call(s) to '0x000000000010a260'
      0.23     1 call(s) to '0x000000000010a1a0'
61
62
63    0.00     if ( argc != 2 )
64             {
65                cerr << "there must be 1 argument - file with array of numbers" << endl ;
66                return 0;
67             }
68             else
69             {
70    0.00        char *fileNum = argv[1];
71    0.00        file.open( fileNum, ifstream::in );
      0.20     1 call(s) to '0x000000000010a1d0'
72    0.00        if ( !file.is_open() ) //for case program can not open a file
      0.00     1 call(s) to '0x000000000010a290'
73                {
74                   throw new std::runtime_error("Failed to open file");
75                   return 0;
...         ...
77             }
78             double num;
79             double array[10000]; // load numbers to this array
80    0.00     double numSize = 0;
81    0.41     for ( int i = 0; file >> num ; i++ )
      0.16     1001 call(s) to '0x000000000010a270'
     33.04     1001 call(s) to '0x000000000010a1f0'
82             {
83    0.11        array[i] = num;
84    0.11        numSize++;
85             }
86
87    0.00     double deviation = aberrance( array, numSize);
      3.13     1 call(s) to 'aberrance(double*, int)' (stddeviation: stddeviation.cpp)
88    0.00     cout << deviation << endl;
      0.09     1 call(s) to '0x000000000010a240'
      0.37     1 call(s) to '0x000000000010a2d0'
89    0.00     return 0;
90    0.00  }
91
```

4- přehled běžících procesů v main



5- přehled všech běžících procesů znázorněný v grafu