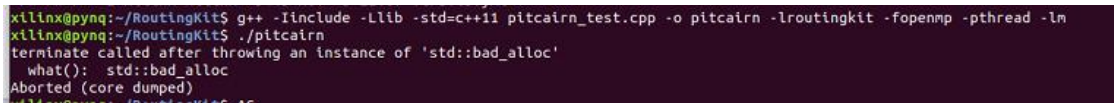
Performance and Analysis Optimization – Project Report

Dinu Diana–Gabriela
Cristea Ioana Catalina
Mihailescu Bogdan

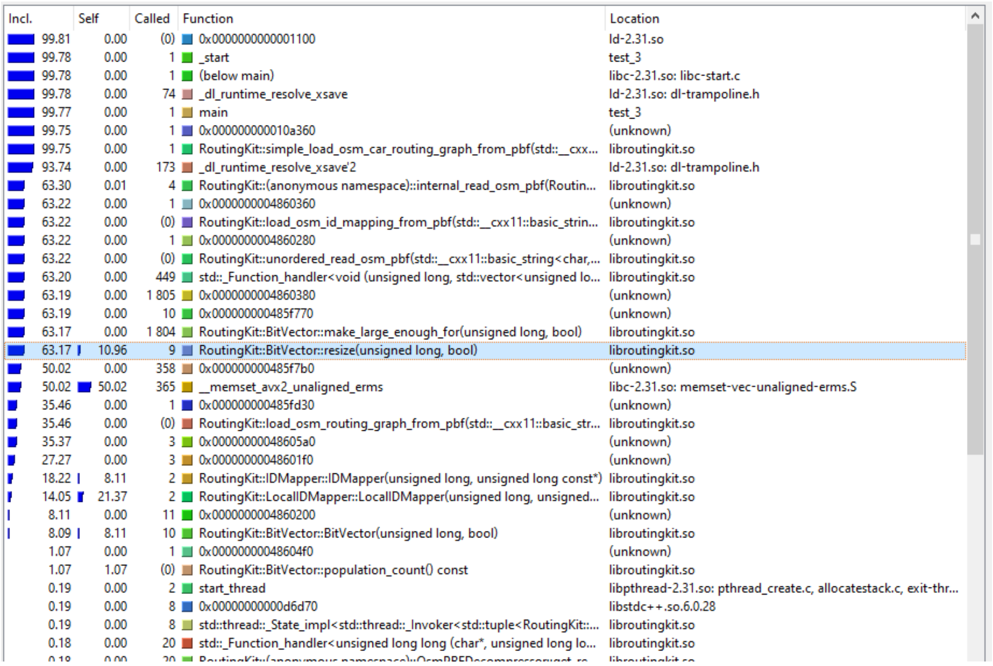
First analysis of the code

The analysis was performed on a personal Computer because it was observed that the program was using way too much RAM than the Pynq Board had available, which was something unexpected. It was later confirmed by an issue that someone had previously filed in the RoutingKit project, that the pbf decoder was using a lot of memory (a base of 1GB and additional storage directly proportional to the size of the map). We tried with the smallest pbf maps available (smaller than 100KB) and even with custom maps (by downloading an osm map with a couple of streets and converting it to pbf maps using a tool named Osmosis) but nothing made it possible to run the RoutingKit on the Pynq board.



```
xilinx@pynq:~/RoutingKit$ g++ -Iinclude -Llib -std=c++11 pitcairn_test.cpp -o pitcairn -lroutingkit -fopenmp -pthread -lm
xilinx@pynq:~/RoutingKit$ ./pitcairn
terminate called after throwing an instance of 'std::bad_alloc'
  what():  std::bad_alloc
Aborted (core dumped)
```

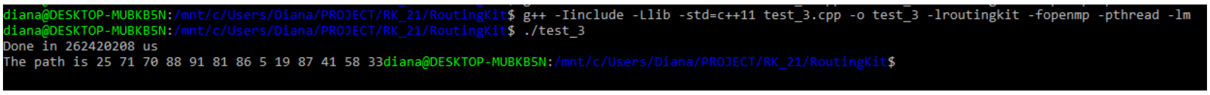
The profiling information can be seen below. Where it can be observed that the main hotspot is, as expected, the load_osm_car_routing_graph_from_pbf function.



Incl.	Self	Called	Function	Location
99.81	0.00	(0)	0x0000000000001100	ld-2.31.so
99.78	0.00	1	_start	test_3
99.78	0.00	1	(below main)	libc-2.31.so: libc-start.c
99.78	0.00	74	_dl_runtime_resolve_xsave	ld-2.31.so: dl-trampoline.h
99.77	0.00	1	main	test_3
99.75	0.00	1	0x000000000010a360	(unknown)
99.75	0.00	1	RoutingKit::simple_load_osm_car_routing_graph_from_pbf(std::__cxx...	libroutingkit.so
93.74	0.00	173	_dl_runtime_resolve_xsave'2	ld-2.31.so: dl-trampoline.h
63.30	0.01	4	RoutingKit::(anonymous namespace)::internal_read_osm_pbf(Routin...	libroutingkit.so
63.22	0.00	1	0x0000000004860360	(unknown)
63.22	0.00	(0)	RoutingKit::load_osm_id_mapping_from_pbf(std::__cxx11::basic_strin...	libroutingkit.so
63.22	0.00	1	0x0000000004860280	(unknown)
63.22	0.00	(0)	RoutingKit::unordered_read_osm_pbf(std::__cxx11::basic_string<char,...	libroutingkit.so
63.20	0.00	449	std::_Function_handler<void (unsigned long, std::vector<unsigned lo...	libroutingkit.so
63.19	0.00	1 805	0x0000000004860380	(unknown)
63.19	0.00	10	0x000000000485f770	(unknown)
63.17	0.00	1 804	RoutingKit::BitVector::make_large_enough_for(unsigned long, bool)	libroutingkit.so
63.17	10.96	9	RoutingKit::BitVector::resize(unsigned long, bool)	libroutingkit.so
50.02	0.00	358	0x000000000485f7b0	(unknown)
50.02	50.02	365	__memset_avx2_unaligned_erms	libc-2.31.so: memset-vec-unaligned-erms.S
35.46	0.00	1	0x000000000485fd30	(unknown)
35.46	0.00	(0)	RoutingKit::load_osm_routing_graph_from_pbf(std::__cxx11::basic_str...	libroutingkit.so
35.37	0.00	3	0x00000000048605a0	(unknown)
27.27	0.00	3	0x00000000048601f0	(unknown)
18.22	8.11	2	RoutingKit::IDMapper::IDMapper(unsigned long, unsigned long const*)	libroutingkit.so
14.05	21.37	2	RoutingKit::LocalIDMapper::LocalIDMapper(unsigned long, unsigned...	libroutingkit.so
8.11	0.00	11	0x0000000004860200	(unknown)
8.09	8.11	10	RoutingKit::BitVector::BitVector(unsigned long, bool)	libroutingkit.so
1.07	0.00	1	0x00000000048604f0	(unknown)
1.07	1.07	(0)	RoutingKit::BitVector::population_count() const	libroutingkit.so
0.19	0.00	2	start_thread	libpthread-2.31.so: pthread_create.c, allocatestack.c, exit-thr...
0.19	0.00	8	0x00000000000d6d70	libstdc++.so.6.0.28
0.19	0.00	8	std::thread::_State_impl<std::thread::_Invoker<std::tuple<RoutingKit::...	libroutingkit.so
0.18	0.00	20	std::_Function_handler<unsigned long long (char*, unsigned long lo...	libroutingkit.so
0.18	0.00	20	RoutingKit::(anonymous namespace)::OsmPBFDecompressor::get_re...	libroutingkit.so

In order to fully analyze the performance of the program, the time that it takes for it to run was also measured. The command used for compiling was the one recommended in the installation and setup section of the project:
g++ -Iinclude -Llib -std=c++11 test_3.cpp -o test_3 -lroutingkit -fopenmp -pthread -lm

The program took 262s complete, as it can be seen below.



```
diana@DESKTOP-MUBKB5N:/mnt/c/Users/Diana/PROJECT/RK_21/RoutingKit$ g++ -Iinclude -Llib -std=c++11 test_3.cpp -o test_3 -lroutingkit -fopenmp -pthread -lm
diana@DESKTOP-MUBKB5N:/mnt/c/Users/Diana/PROJECT/RK_21/RoutingKit$ ./test_3
Done in 262420208 us
The path is 25 71 70 88 91 81 86 5 19 87 41 58 33diana@DESKTOP-MUBKB5N:/mnt/c/Users/Diana/PROJECT/RK_21/RoutingKit$
```

The bottleneck was also identified to be tied to the pbf decoder function, as when the graph is created, a large amount of memory is required, as it was observed that the program was using all the available RAM.

Optimizations

Optimization was a challenge, as the memory bottleneck would make any improvement hard to be seen, and also because parallelization with openmp was already used. We started with improving the compilation command to include optimization flags (-O3 for maximum optimization).

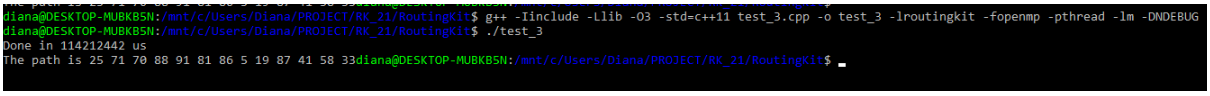
Besides this we also noticed when looking at the code, that many assertions were present, which should not be present in a released application. In order to disable them, we used the –DNDEBUG flag. The final command used is:

g++ -Iinclude -Llib -O3 -ffast-math -std=c++11 test_3.cpp -o test_3 -lroutingkit -fopenmp -pthread -lm -DNDEBUG

Also, we looked for what function calls were often performed. We stopped at the resize and the make_large_enough_for functions, which were called a lot in the program. We tried to reduce function calls as a performance technique, but only succeeded for the resize function (this can be seen in the second profiling analysis screenshot).

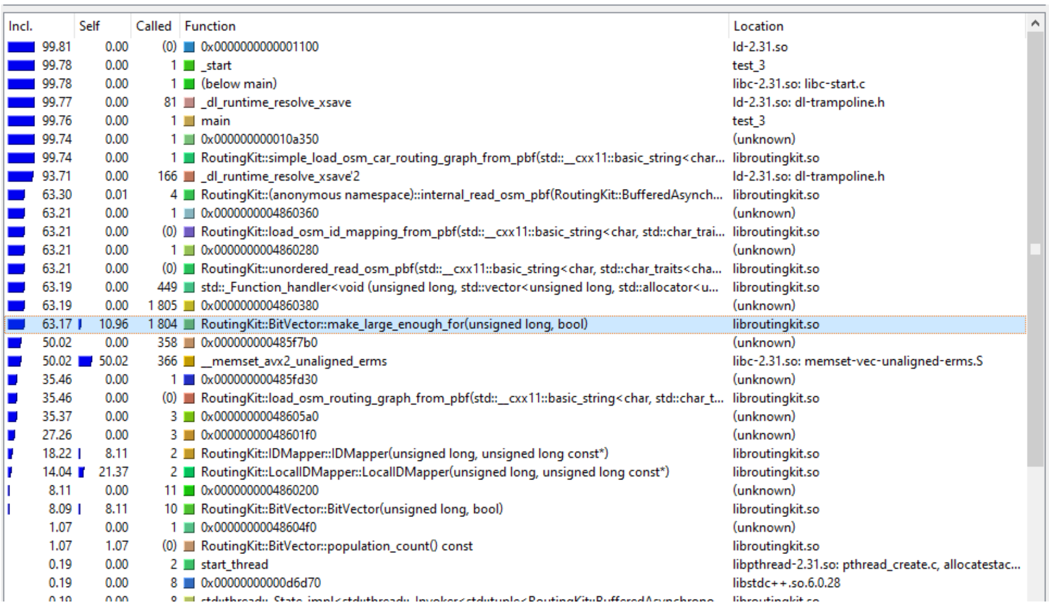
Second performance analysis

By using these techniques a performance increase was seen in the time it takes for the program to run, as it can be seen below, as the runtime was only 114s:



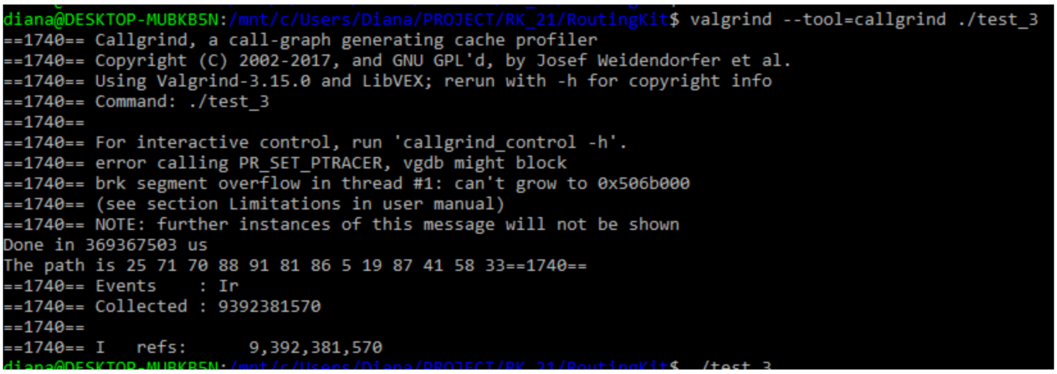
```
diana@DESKTOP-MUBKB5N:/mnt/c/Users/Diana/PROJECT/RK_21/RoutingKit$ g++ -Iinclude -Llib -O3 -std=c++11 test_3.cpp -o test_3 -lroutingkit -fopenmp -pthread -lm -DNDEBUG
diana@DESKTOP-MUBKB5N:/mnt/c/Users/Diana/PROJECT/RK_21/RoutingKit$ ./test_3
Done in 114212442 us
The path is 25 71 70 88 91 81 86 5 19 87 41 58 33diana@DESKTOP-MUBKB5N:/mnt/c/Users/Diana/PROJECT/RK_21/RoutingKit$ _
```

Also, in the second profiling analysis it can be seen that the resize function is not called anymore, because it was integrated in the make_large_enough_for function. Also, the main hotspots remained the same, as it did not allow for any further optimization.



Incl.	Self	Called	Function	Location
99.81	0.00	(0)	0x0000000000001100	ld-2.31.so
99.78	0.00	1	_start	test_3
99.78	0.00	1	(below main)	libc-2.31.so: libc-start.c
99.77	0.00	81	_dl_runtime_resolve_xsave	ld-2.31.so: dl-trampoline.h
99.76	0.00	1	main	test_3
99.74	0.00	1	0x000000000010a350	(unknown)
99.74	0.00	1	RoutingKit::simple_load_osm_car_routing_graph_from_pbf(std::__cxx11::basic_string<char...	libroutingkit.so
93.71	0.00	166	_dl_runtime_resolve_xsave'2	ld-2.31.so: dl-trampoline.h
63.30	0.01	4	RoutingKit::(anonymous namespace)::internal_read_osm_pbf(RoutingKit::BufferedAsynch...	libroutingkit.so
63.21	0.00	1	0x0000000004860360	(unknown)
63.21	0.00	(0)	RoutingKit::load_osm_id_mapping_from_pbf(std::__cxx11::basic_string<char, std::char_trai...	libroutingkit.so
63.21	0.00	1	0x0000000004860280	(unknown)
63.21	0.00	(0)	RoutingKit::unordered_read_osm_pbf(std::__cxx11::basic_string<char, std::char_traits<cha...	libroutingkit.so
63.19	0.00	449	std::_Function_handler<void (unsigned long, std::vector<unsigned long, std::allocator<u...	libroutingkit.so
63.19	0.00	1 805	0x0000000004860380	(unknown)
63.17	10.96	1 804	RoutingKit::BitVector::make_large_enough_for(unsigned long, bool)	libroutingkit.so
50.02	0.00	358	0x000000000485f7b0	(unknown)
50.02	50.02	366	__memset_avx2_unaligned_erms	libc-2.31.so: memset-vec-unaligned-erms.S
35.46	0.00	1	0x000000000485fd30	(unknown)
35.46	0.00	(0)	RoutingKit::load_osm_routing_graph_from_pbf(std::__cxx11::basic_string<char, std::char_t...	libroutingkit.so
35.37	0.00	3	0x00000000048605a0	(unknown)
27.26	0.00	3	0x00000000048601f0	(unknown)
18.22	8.11	2	RoutingKit::IDMapper::IDMapper(unsigned long, unsigned long const*)	libroutingkit.so
14.04	21.37	2	RoutingKit::LocalIDMapper::LocalIDMapper(unsigned long, unsigned long const*)	libroutingkit.so
8.11	0.00	11	0x0000000004860200	(unknown)
8.09	8.11	10	RoutingKit::BitVector::BitVector(unsigned long, bool)	libroutingkit.so
1.07	0.00	1	0x00000000048604f0	(unknown)
1.07	1.07	(0)	RoutingKit::BitVector::population_count() const	libroutingkit.so
0.19	0.00	2	start_thread	libpthread-2.31.so: pthread_create.c, allocatestac...
0.19	0.00	8	0x0000000000d6d70	libstdc++.so.6.0.28
0.19	0.00	8	std::thread::_State_impl<std::thread::_Invoker<std::tuple<RoutingKit::BufferedAsynchrono...	libroutingkit.so

Overall, we saw an improvement in performance, but also, the program did not allow us to use a big range of optimization techniques. Also the memory profiling could not be fully performed, as the Valgrind tool reached its limitations when profiling this application as it can be seen in the screenshot below.



```
diana@DESKTOP-MUBKB5N:/mnt/c/Users/Diana/PROJECT/RK_21/RoutingKit$ valgrind --tool=callgrind ./test_3
==1740== Callgrind, a call-graph generating cache profiler
==1740== Copyright (C) 2002-2017, and GNU GPL'd, by Josef Weidendorfer et al.
==1740== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==1740== Command: ./test_3
==1740==
==1740== For interactive control, run 'callgrind_control -h'.
==1740== error calling PR_SET_PTRACER, vgdb might block
==1740== brk segment overflow in thread #1: can't grow to 0x506b000
==1740== (see section Limitations in user manual)
==1740== NOTE: further instances of this message will not be shown
Done in 369367503 us
The path is 25 71 70 88 91 81 86 5 19 87 41 58 33==1740==
==1740== Events    : Ir
==1740== Collected : 9392381570
==1740==
==1740== I   refs:      9,392,381,570
diana@DESKTOP-MUBKB5N:/mnt/c/Users/Diana/PROJECT/RK_21/RoutingKit$ ./test_3
```