

# Применение гибридного OpenMP/GPGPU подхода для ускорения вычислений 3D поля скорости по сейсмическим данным

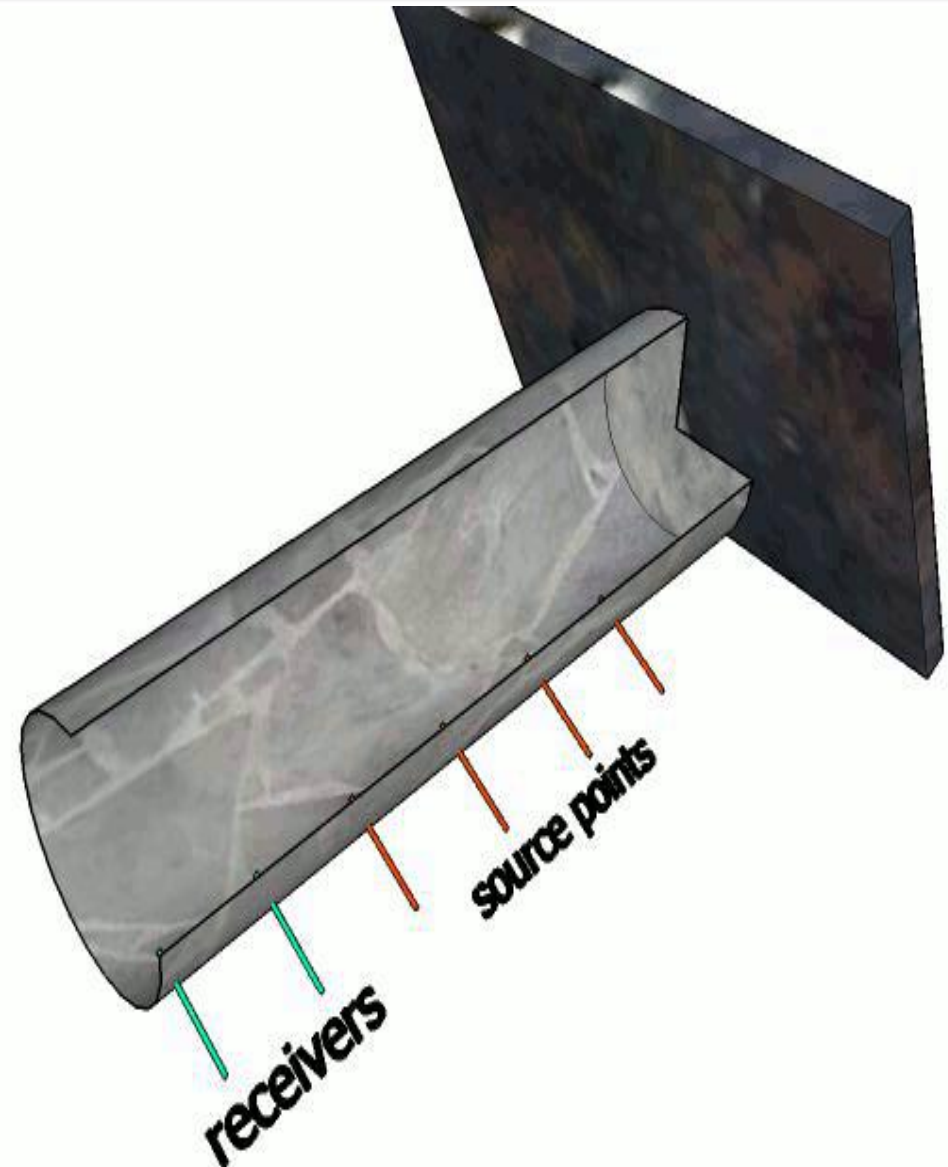


[Singularis Lab](#), Ltd.

Volgograd State Technical University

# О чём пойдёт речь

- Система сейсморазведки при прокладке туннелей
- Оптимизация и ускорение вычислительных процедур



Тестовое оборудование:  
Intel i5-4570  
GeForce GT 640

# Исследование проблем производительности

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Caller/Callee

Elapsed Time: 318.848s

Total Thread Count: 1  
 Overhead Time: 0s  
 Spin Time: 0s  
 CPU Time: 317.616s  
 Paused Time: 0s

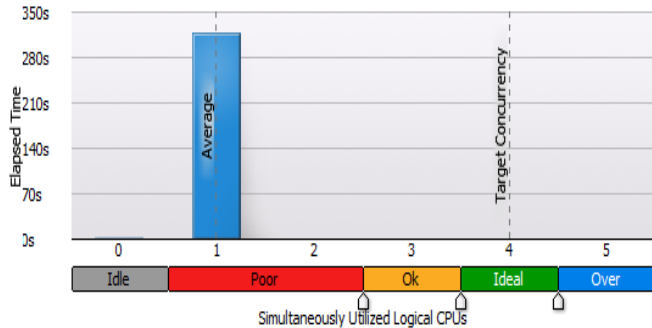
## Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results

Function	CPU Time
<a href="#">tripd2</a>	95.632s
<a href="#">put_velocity_along_ray_path</a>	66.868s
<a href="#">get_velocity_along_ray_path</a>	41.185s
<a href="#">vave2int1drecursivesmooth3</a>	25.408s
<a href="#">smooth_slowness1d_noalloc</a>	24.056s
[Others]	64.467s

## CPU Usage Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. See the Thread Concurrency level if a thread is executing code on a CPU while it is logically waiting. Try to keep your



Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree

Grouping: Function / Call Stack

Function / Call Stack	CPU Time by Utilization					Ov. an.	Module
	Idle	Poor	Ok	Ideal	Over		
tripd2	95.632s					0s	tspvpick3d.exe
put_velocity_along_ray_path	66.868s					0s	tspvpick3d.exe
get_velocity_along_ray_path	41.185s					0s	tspvpick3d.exe
vave2int1drecursivesmooth3	25.408s					0s	tspvpick3d.exe
smooth_slowness1d_noalloc	24.056s					0s	tspvpick3d.exe
sdot	19.040s					0s	tspvpick3d.exe
vsm1d_noalloc	16.677s					0s	tspvpick3d.exe
compute_straight_ray_path	11.433s					0s	tspvpick3d.exe
localxccor3	9.758s					0s	tspvpick3d.exe
_libm_sse2_sqrt_precise	1.862s					0s	MSVCR110.dll
compute_gagc	1.808s					0s	tspvpick3d.exe
find_max	0.879s					0s	tspvpick3d.exe
null1	0.440s					0s	tspvpick3d.exe
null2	0.430s					0s	tspvpick3d.exe
fprintf	0.325s					0s	MSVCR110.dll
vave2int3drecursive	0.300s					0s	tspvpick3d.exe
parabola_extremum_3point	0.263s					0s	tspvpick3d.exe
func@0x1000d0c0	0.224s					0s	TSPNGDataC...
dc_getPoint	0.200s					0s	TSPNGDataC...
quant	0.100s					0s	tspvpick3d.exe
fill_neighborhood	0.080s					0s	tspvpick3d.exe
func@0x10007e90	0.060s					0s	TSPNGDataC...
find_vector_maxima	0.060s					0s	tspvpick3d.exe
std::basic_istream<char,struct std::char_traits<char> >::	0.047s					0s	MSVCP90.dll
smooth_slowness3d	0.040s					0s	tspvpick3d.exe
_startOneArgErrorHandling	0.040s					0s	MSVCR110.dll
neighborhood_equalization_med	0.040s					0s	tspvpick3d.exe
main	0.040s					0s	tspvpick3d.exe

# tripd2

# put\_velocity

# get\_velocity

Assembly	CPU Time: Total by Utilization				
	Idle	Poor	Ok	Ideal	Over
mov edi, dword ptr [ecx+ebx*1]					
mov ecx, dword ptr [ebp+0x8]	1.867s				
mov eax, dword ptr [ebx]	0.020s				
mov ecx, dword ptr [ecx+esi*4]					
sub edi, eax	0.010s				
sub ecx, eax	1.616s				
mov dword ptr [ebp-0x4], edx					
lea ecx, ptr [ecx]					
Block 5:					
movss xmm0, dword ptr [eax]					
movaps xmm1, xmm0	1.640s				
divss xmm1, dword ptr [edi+eax*1]					
add eax, 0x4	25.695s				
dec dword ptr [ebp-0x4]	0.050s				
movss dword ptr [eax-0x4], xmm1	1.588s				
mulss xmm1, xmm0					
movss xmm0, dword ptr [eax+ecx*1-0x4]	6.565s				
subss xmm0, xmm1					
movss dword ptr [eax+ecx*1-0x4], xmm0	5.165s				
jnz 0x41c3c0 <Block 5>	1.859s				
Block 6:					
mov edi, dword ptr [ebp+0x14]					
mov ecx, dword ptr [ebp-0x8]					
Block 7:					
inc esi					
add ebx, 0x4	1.728s				
cmp esi, edi	0.010s				
j1 0x41c3a7 <Block 3>					
Block 8:					
cmp edi, 0x1					
jle 0x41c455 <Block 16>					
Block 9:					
mov ebx, dword ptr [ebp+0x10]					
mov eax, dword ptr [ebp+0xc]					
sub eax, ebx					
mov dword ptr [ebp-0x8], eax					

Assembly	CPU Time: Total by Utilization				
	Idle	Poor	Ok	Ideal	Over
movss dword ptr [eax+ecx*4], xmm0	2.161s				
Block 9:					
mulss xmm1, xmm6	0.540s				
comiss xmm1, xmm4	0.260s				
jbe 0x428ae9 <Block 11>	0.180s				
Block 10:					
mov eax, dword ptr [ebx+edx*4]	0.060s				
mov eax, dword ptr [eax+esi*4+0x4]	0.200s				
movss xmm0, dword ptr [eax+ecx*4+0x4]	0.050s				
addss xmm0, xmm1	0.190s				
movss dword ptr [eax+ecx*4+0x4], xmm0	0.180s				
mov eax, dword ptr [ebp+0x3c]	0.160s				
movss xmm0, dword ptr [edi]	0.010s				
mov eax, dword ptr [eax+edx*4]	0.060s				
mulss xmm0, xmm1	0.060s				
mov eax, dword ptr [eax+esi*4+0x4]	0.170s				
addss xmm0, dword ptr [eax+ecx*4+0x4]					
movss dword ptr [eax+ecx*4+0x4], xmm0	1.020s				
Block 11:					
movss xmm5, dword ptr [ebp-0x8]	0.450s				
mulss xmm3, xmm5	0.239s				
movaps xmm1, xmm3	0.120s				
mulss xmm1, xmm7	0.090s				
comiss xmm1, xmm4	0.575s				
jbe 0x428b2e <Block 13>	0.430s				
Block 12:					
mov eax, dword ptr [ebx+edx*4+0x4]	0.060s				
movaps xmm0, xmm1	0.080s				
mov eax, dword ptr [eax+esi*4]	0.410s				
addss xmm0, dword ptr [eax+ecx*4]	0.260s				
movss dword ptr [eax+ecx*4], xmm0	0.430s				
mov eax, dword ptr [ebp+0x3c]	0.050s				
movss xmm0, dword ptr [edi]	0.400s				
mov eax, dword ptr [eax+edx*4+0x4]	0.180s				
mulss xmm0, xmm1	0.030s				
mov eax, dword ptr [eax+esi*4]	0.060s				
addss xmm0, dword ptr [eax+ecx*4]	0.420s				
movss dword ptr [eax+ecx*4], xmm0	2.500s				

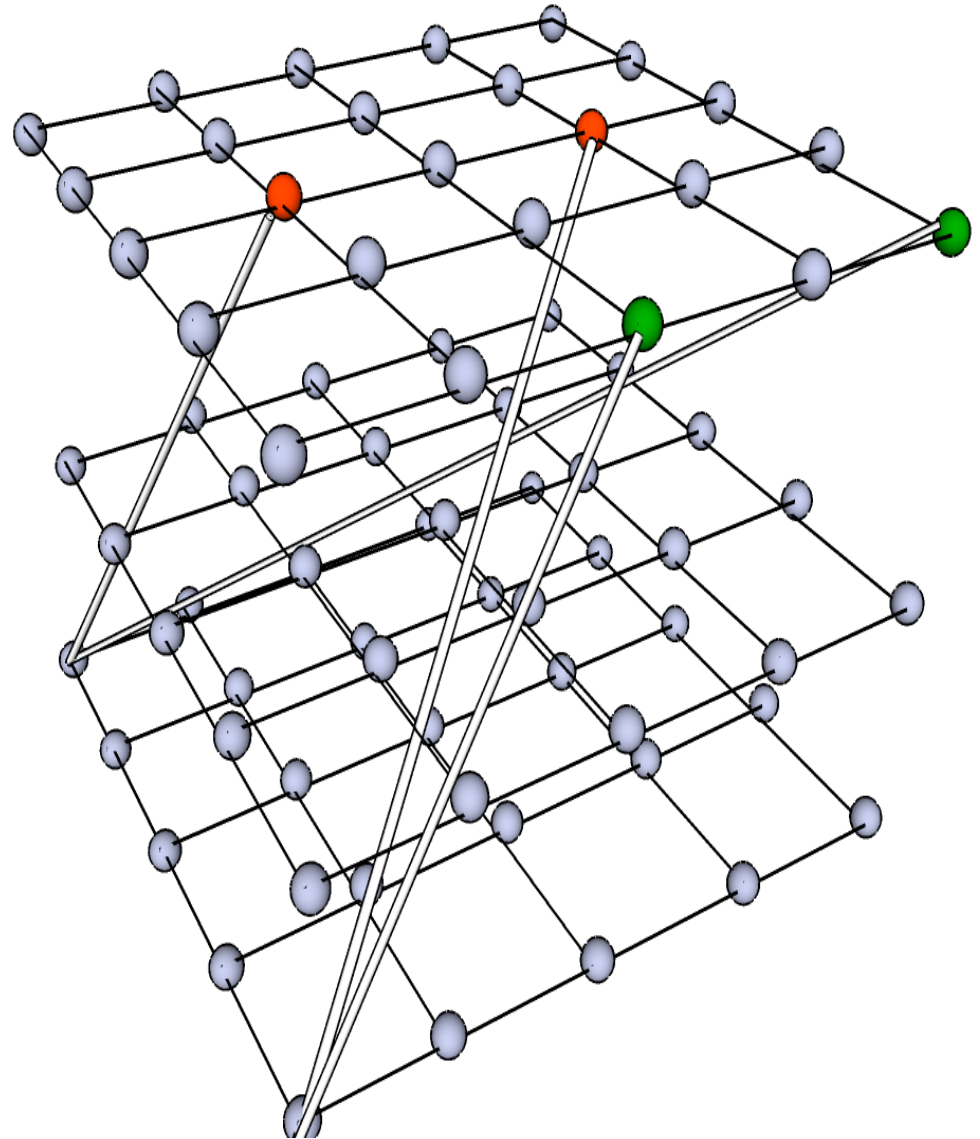
Assembly	CPU Time: Total by Utilization				
	Idle	Poor	Ok	Ideal	Over
cmp ebx, esi	0.090s				
subss xmm4, xmm0	0.480s				
cmovnl ebx, esi	0.040s				
xor eax, eax	0.340s				
cvtupd2ps xmm5, xmm1	0.560s				
test edi, edi	0.350s				
cmovs edi, eax	0.120s				
mov eax, dword ptr [ebp+0x3c]	0.400s				
cmp edi, dword ptr [ebp+0x1c]	0.220s				
mov eax, dword ptr [eax+ecx*4+0x4]	0.120s				
cmovnl edi, dword ptr [ebp+0x1c]	0.406s				
mov edx, dword ptr [eax+ebx*4]	0.279s				
mov esi, dword ptr [eax+ebx*4+0x4]	0.230s				
mov eax, dword ptr [ebp+0x3c]	0.550s				
cvtups2pd xmm0, xmm4	0.020s				
mov eax, dword ptr [eax+ecx*4]	0.400s				
movaps xmm1, xmm2	0.469s				
mov ecx, dword ptr [eax+ebx*4]	0.020s				
mov eax, dword ptr [eax+ebx*4+0x4]	0.341s				
subsd xmm1, xmm0	0.218s				
movss xmm0, dword ptr [eax+edi*4]	0.440s				
movss xmm2, dword ptr [eax+edi*4+0x4]	0.899s				
mulss xmm2, xmm4	0.620s				
cvtupd2ps xmm3, xmm1	2.019s				
movss xmm1, dword ptr [ecx+edi*4+0x4]	0.140s				
mulss xmm0, xmm3	0.400s				
mulss xmm1, xmm4	0.936s				
addss xmm2, xmm0	0.820s				
movss xmm0, dword ptr [ecx+edi*4]	1.628s				
mulss xmm0, xmm3	0.160s				
mulss xmm2, xmm7	0.110s				
addss xmm1, xmm0	3.408s				
movss xmm0, dword ptr [edx+edi*4]	0.150s				
mulss xmm0, xmm3	0.376s				
mulss xmm3, dword ptr [esi+edi*4]	0.160s				
mulss xmm1, xmm5	0.910s				
addss xmm2, xmm1	0.570s				
movss xmm1, dword ptr [edx+edi*4+0x4]	2.447s				
mulss xmm1, xmm4	0.050s				

# Проблемы доступа к данным

Symbol + Offset	DC accesses	DC misses	Misalign access	Ret inst	DTLB L1M L2H	DTLB L1M L2M	DC refills L2/NB
put_velocity_along_ray_path_p	1069808	140624	236	2062302	37716	1752	406600
tripd2_simplify_p	471328	1926	76	873384	718	42	2638
velocities::get_velocity_along_ray_path	440766	63922	154	1121879	17362	612	103980
qcOutput::muladd	313606	296	589284	590939	50	4	246
vsm1d_noalloc	270030	8036	60	618220	3182	66	49296
velocities::compute_straight_ray_path	218116	3440	586	439061	1344	22	18290
smooth_slowness1d_noalloc	104490	1632	58	438653	704	12	7464
velocities::wave2int1drecursivesmooth3	82818	1692	90018	373051	706	12	7530
qcOutput::multiply	34826	200	55774	40199	6	8	534
qcOutput::localxcor3	34676	68	3348	97221	6	4	512
find_max	6820	22	4	23335			4
velocities::wave2int3drecursivefomp\$1	5396	3716	2	9067	1008	90	3288
parabola_extremum_3point	3460	132		5076	24		188
null1	2680	270	4	15735	20	12	2516
sqrt	1850	228	410	2383		4	228
fill_neighborhood	1196	34		3668	36		46
tripd2	1092	14		2287	4		138
main	1040			1547	2		
qcOutput::find_vector_maxima	956	12		1831			78
quant	714			1391			2
neighborhood_equalization	586	14	2	1038			34
vsm3d	496	12		806	2		100
border_vs_main_grid	432			1121			26
neighborhood_equalization_med	336	26		861			54
null3	318	10		537		6	182
moment_simpl	300	4		180	2		4
qcOutput::compute_stackword_2d	108	4		319			2
find_max2d	98	10		256			60
std::vector<float, std::allocator<float> >::resize	82	8		15			30
smooth_slowness3d	70	2		318			26
vcomp_atomic_add_r4	64	2		202			
qcOutput::update_average_velocity	44	2	2	64			8
qcOutput::CalcIntVelDipQualfomp\$1	42	10		55	4	10	26

# Velocity along ray path

- Проблемы:
  - невыровненный доступ к памяти
  - промахи кэша
- Решение
  - уменьшить куб
  - реорганизовать работу с памятью



# Логическая структура программы

get\_velocity\_along\_ray\_path

вычисление траекторий лучей в кубе скоростей

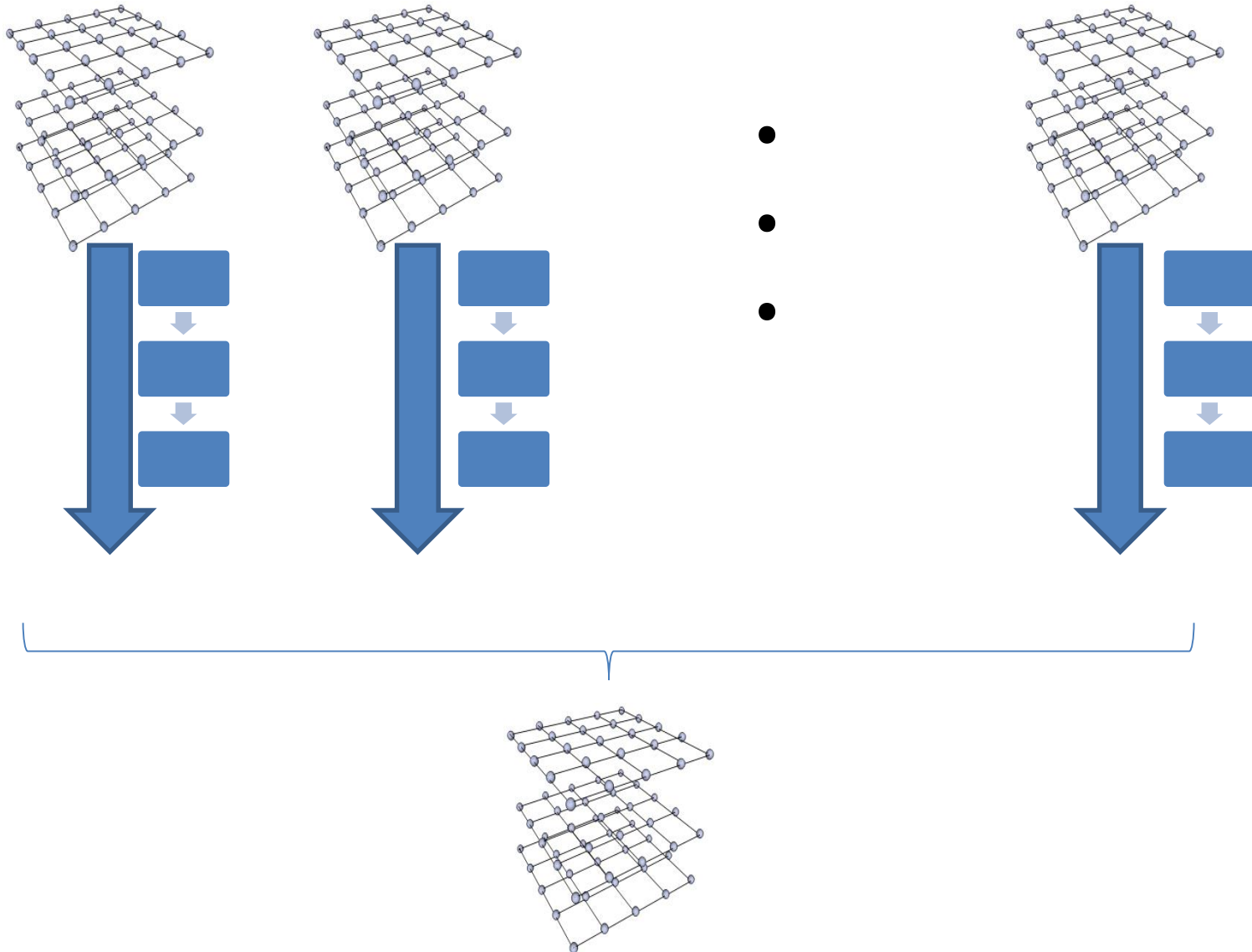
tripd2

сглаживание скорости вдоль траектории луча

put\_velocity\_along\_ray\_path

модификация куба скоростей

# Разделение вычислений между потоками





# Результат с SSE intrinsics, выравниванием памяти и OpenMP

Analysis Target Analysis Type Collection Log Summary Bottom-up Caller/Callee

Elapsed Time: 66.978s

Total Thread Count:	11
Overhead Time:	0.020s
Spin Time:	3.934s
CPU Time:	249.417s
Paused Time:	0s

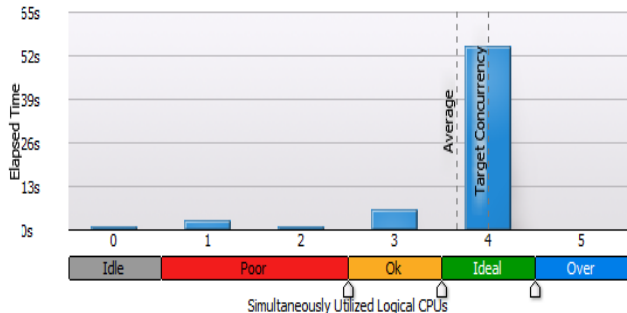
## Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results

Function	CPU Time
<a href="#">fastvelocities::tripd2</a>	77.024s
<a href="#">fastvelocities::put_velocity_along_ray_path</a>	52.637s
<a href="#">fastvelocities::get_velocity_along_ray_path</a>	44.502s
<a href="#">fastvelocities::vave2int1drecursivesmooth3</a>	25.351s
<a href="#">fastcube::muladd</a>	12.340s
[Others]	37.564s

## CPU Usage Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Sp the Thread Concurrency level if a thread is executing code on a CPU while it is logically waiting. Try to keep your



Analysis Target Analysis Type Collection Log Summary Bottom-up Caller/Callee

Elapsed Time: 318.848s

Total Thread Count:	1
Overhead Time:	0s
Spin Time:	0s
CPU Time:	317.616s
Paused Time:	0s

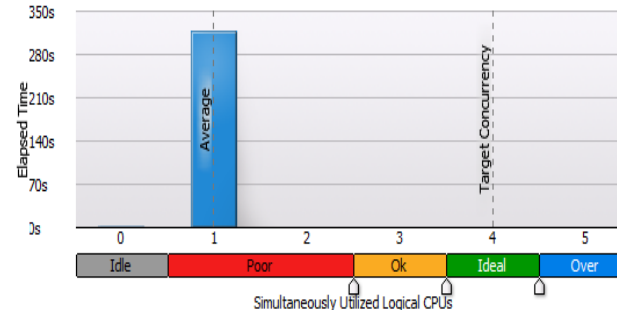
## Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in

Function	CPU Time
<a href="#">tripd2</a>	95.632s
<a href="#">put_velocity_along_ray_path</a>	66.868s
<a href="#">get_velocity_along_ray_path</a>	41.185s
<a href="#">vave2int1drecursivesmooth3</a>	25.408s
<a href="#">smooth_slowness1d_noalloc</a>	24.056s
[Others]	64.467s

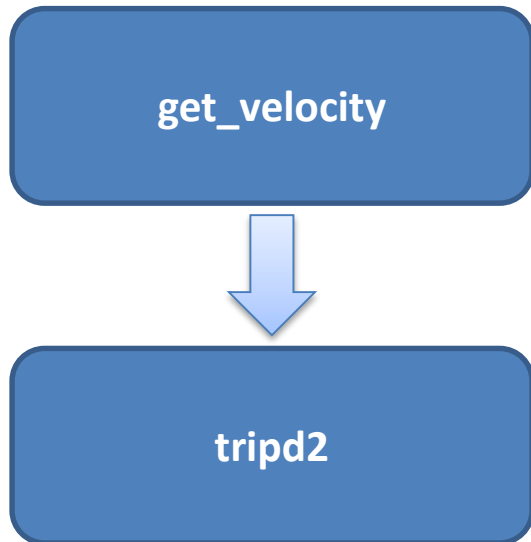
## CPU Usage Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Sp the Thread Concurrency level if a thread is executing code on a CPU while it is logically waiting. Try to keep your



# Использование GPU ускорителя

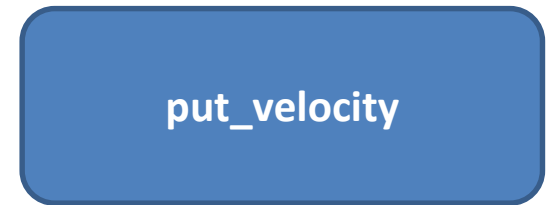
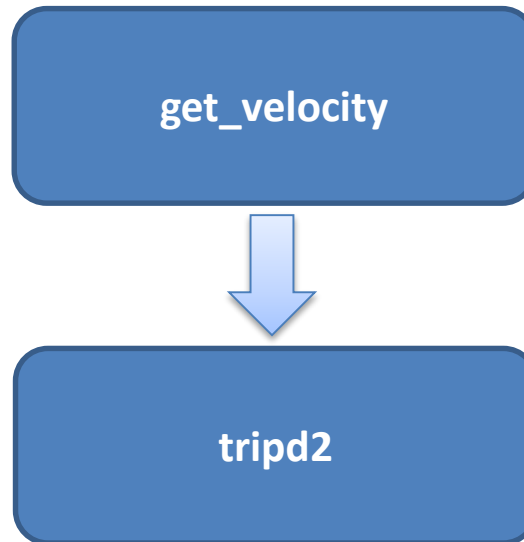
- Step 1: GPU



GPU

- Step 2

CPU





- Step 3





# Сравнение результатов



## Исходный код

 <b>Elapsed Time:</b> <sup>?</sup> <b>318.848s</b> 	
<a href="#">Total Thread Count:</a>	1
<a href="#">Overhead Time:</a> <sup>?</sup>	0s
<a href="#">Spin Time:</a> <sup>?</sup>	0s
<a href="#">CPU Time:</a> <sup>?</sup>	317.616s
<a href="#">Paused Time:</a> <sup>?</sup>	0s

## CPU only

 <b>Elapsed Time:</b> <sup>?</sup> <b>66.978s</b> 	
<a href="#">Total Thread Count:</a>	11
<a href="#">Overhead Time:</a> <sup>?</sup>	0.020s
<a href="#">Spin Time:</a> <sup>?</sup>	3.934s
<a href="#">CPU Time:</a> <sup>?</sup>	249.417s
<a href="#">Paused Time:</a> <sup>?</sup>	0s

## CPU+GPU

 <b>Elapsed Time:</b> <sup>?</sup> <b>35.057s</b> 	
<a href="#">Total Thread Count:</a>	19
<a href="#">Overhead Time:</a> <sup>?</sup>	0.319s
<a href="#">Spin Time:</a> <sup>?</sup>	32.426s
A significant portion of CPU time is spent waiting (due to contention for GPU resources, or adjusting by backing off then rescheduling), or adjusting	
<a href="#">CPU Time:</a> <sup>?</sup>	121.823s
<a href="#">Paused Time:</a> <sup>?</sup>	0s

# Заключение

- Хотспоты были проанализированы с помощью Intel VTune Amplifier
- Приложение было ускорено за счёт одновременных вычислений на GPU и CPU
- Итоговое ускорение на тестовой машине более 9 раз

