ROMS-UCLA code review: profiling

Devin Dollery

UCLA
CESR - ATMOS

September 3, 2020

Overview

Intel Profiling Suite

Intel Parallel Studio:

Intel Profiling Suite

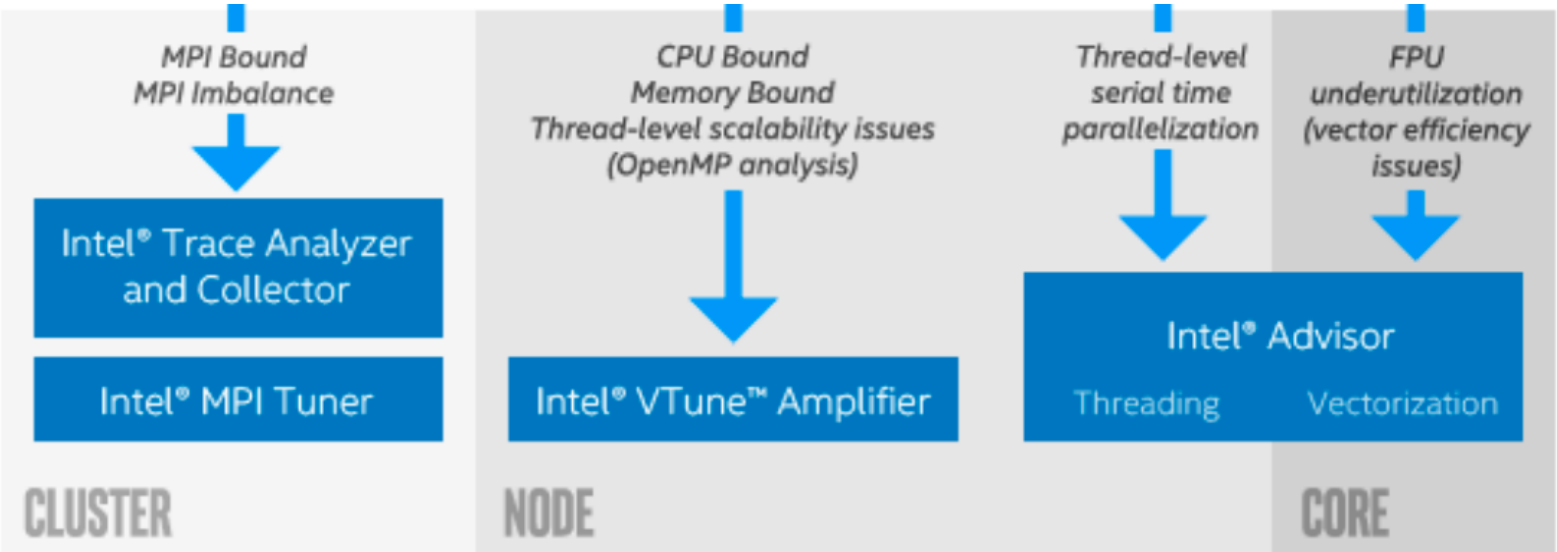
Intel Parallel Studio:

- Fortran and c-compilers (ifort, icc, mpiifort, etc)

Intel Profiling Suite

Intel Parallel Studio:

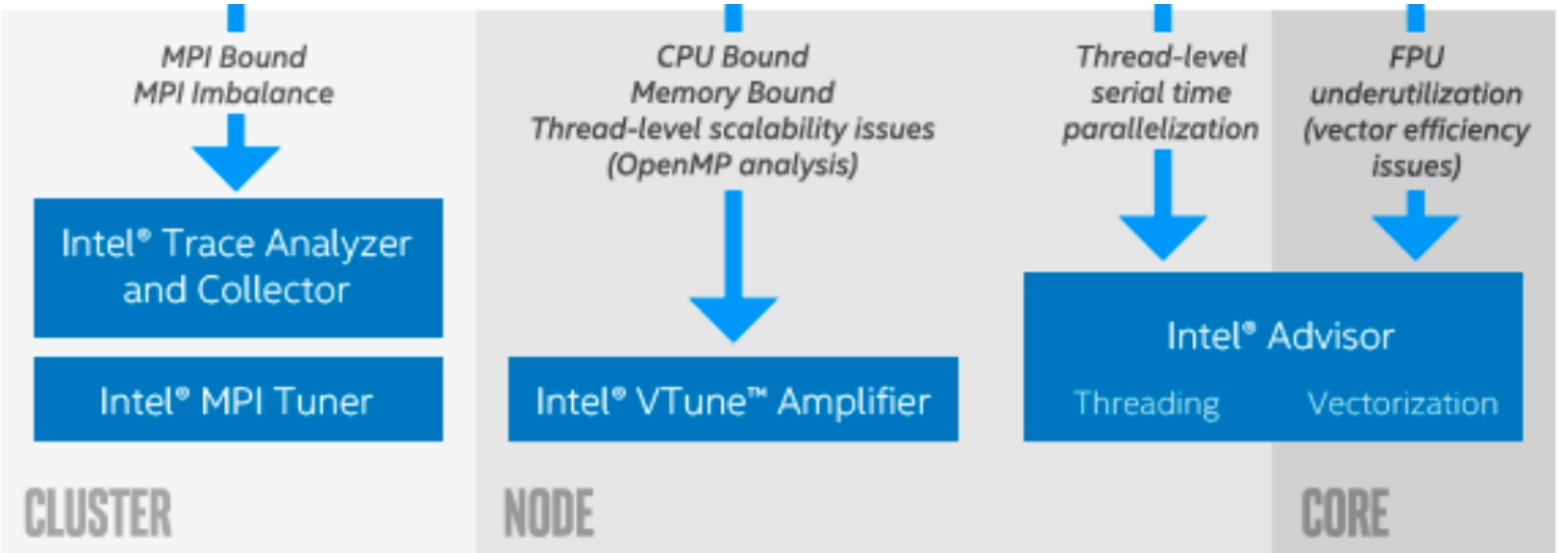
- Fortran and c-compilers (ifort, icc, mpiifort, etc)
- Profiling tools



Intel Profiling Suite

Intel Parallel Studio:

- Fortran and c-compilers (ifort, icc, mpiifort, etc)
- Profiling tools

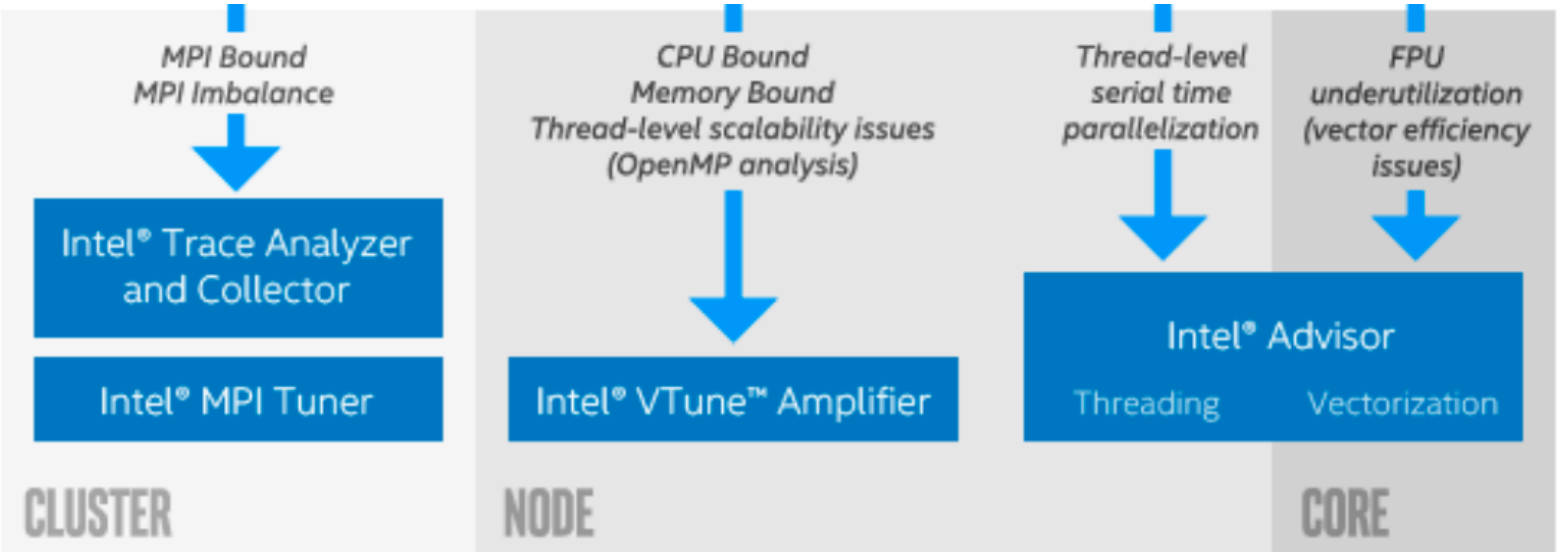


- Intel MPI

Intel Profiling Suite

Intel Parallel Studio:

- Fortran and c-compilers (ifort, icc, mpiifort, etc)
- Profiling tools



- Intel MPI
  - Currently using MPICH on maya

Test Simulation

Test Simulation

- Constraint
  - Small domain - rapid development testing on laptop
  - 1 Time step $<$ 1 second to compute

Test Simulation

- Constraint
  - Small domain - rapid development testing on laptop
  - 1 Time step $<$ 1 second to compute
- US West-Coast simulation
  - Delphine's WEC research
  - Compare old vs new ROMS
  - Speeds similar
  - New code larger maximum time-step

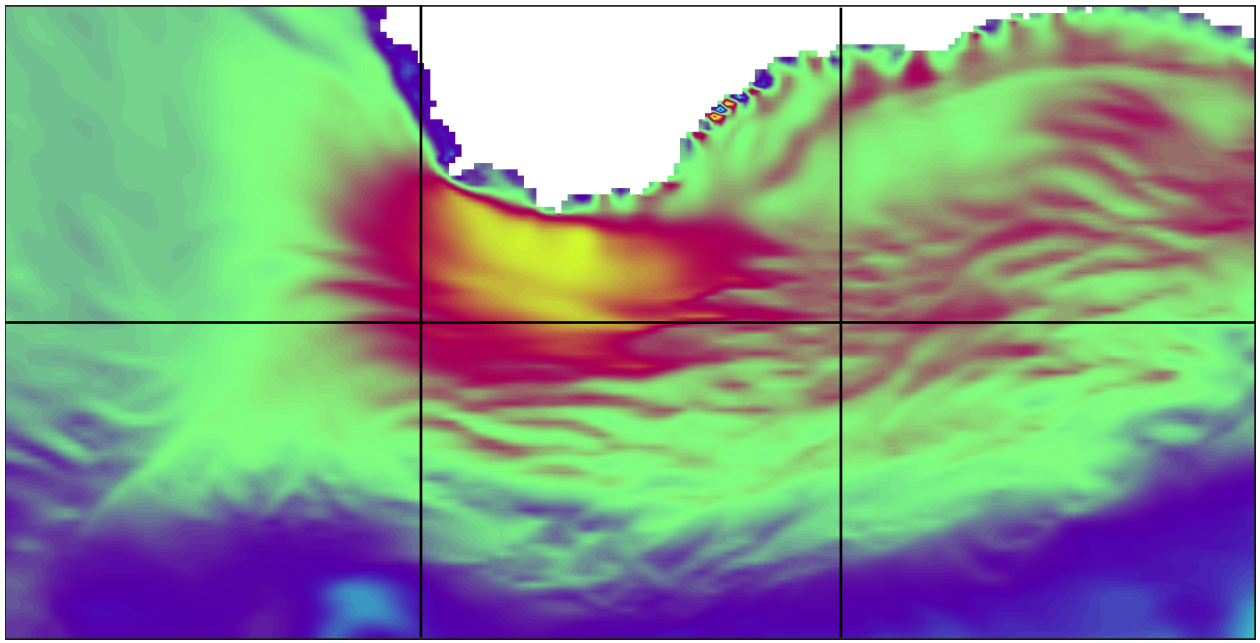
Test Simulation

- Constraint
  - Small domain - rapid development testing on laptop
  - 1 Time step $<$ 1 second to compute
- US West-Coast simulation
  - Delphine's WEC research
  - Compare old vs new ROMS
  - Speeds similar
  - New code larger maximum time-step
- Modules
  - WEC (ported into new code)
  - Bulk forcing (ported into new code)

Test Simulation

- Constraint
  - Small domain - rapid development testing on laptop
  - 1 Time step $<$ 1 second to compute
- US West-Coast simulation
  - Delphine's WEC research
  - Compare old vs new ROMS
  - Speeds similar
  - New code larger maximum time-step
- Modules
  - WEC (ported into new code)
  - Bulk forcing (ported into new code)
- Land to test masking

USWC Simulation



- 6 MPI tiles (3x2)
- Nodes = 200x*100y*50z = 1e6

Maya nodes

Maya nodes

- ifort - 2014 fortran compiler
  - Differences from 2020 ifort (modules & common blocks)

Maya nodes

- ifort - 2014 fortran compiler
  - Differences from 2020 ifort (modules & common blocks)
- Profiling tools installed, not working - not compatible with Kernel 4.18.

Maya nodes

- ifort - 2014 fortran compiler
  - Differences from 2020 ifort (modules & common blocks)
- Profiling tools installed, not working - not compatible with Kernel 4.18.
- Problems installing 2020 Intel suite.
  - Maya O.S. is Mageia 6 (quite old).

Maya nodes

- ifort - 2014 fortran compiler
  - Differences from 2020 ifort (modules & common blocks)
- Profiling tools installed, not working - not compatible with Kernel 4.18.
- Problems installing 2020 Intel suite.
  - Maya O.S. is Mageia 6 (quite old).
- Asked Cody & Henry to upgrade maya O.S., probably latest centOS distro.

Hardware Specs

- Laptop
  - Intel i7 1.8GHz - 4 core - hyper (8 logical CPU's)
  - cache: 128K, 1M, 8M
  - Runtime: 386s [385s, 385s, 387s] (1080 steps)

Hardware Specs

- Laptop
  - Intel i7 1.8GHz - 4 core - hyper (8 logical CPU's)
  - cache: 128K, 1M, 8M
  - Runtime: 386s [385s, 385s, 387s] (1080 steps)
- Zulu
  - Intel i7 3.4GHz - 6 core - hyper (12 logical CPU's)
  - cache: 32K, 256K, 15M
  - Runtime: 243s [208s, 249s, 271s]

Hardware Specs

- Laptop
  - Intel i7 1.8GHz - 4 core - hyper (8 logical CPU's)
  - cache: 128K, 1M, 8M
  - Runtime: 386s [385s, 385s, 387s] (1080 steps)
- Zulu
  - Intel i7 3.4GHz - 6 core - hyper (12 logical CPU's)
  - cache: 32K, 256K, 15M
  - Runtime: 243s [208s, 249s, 271s]
- Maya head node
  - Intel Xeon 2.5GHz - 2 sockets x 4 cores - 8 physical cores
  - cache: 32K, 6M
  - Runtime: 770s [811s, 749s, 750s]
  - Slower: 2 sockets, only 2 cache levels, other users?

Intel Vtune Profiler

Intel Vtune Profiler

- Results currently only indicative, no code improvements have been made. (Not tested on cluster)

Intel Vtune Profiler

- Results currently only indicative, no code improvements have been made. (Not tested on cluster)
- Profiling aggregates information - great for summaries

Intel Vtune Profiler

- Results currently only indicative, no code improvements have been made. (Not tested on cluster)
- Profiling aggregates information - great for summaries
- Can't see what's happening instantaneously (tracing)

Intel Vtune Profiler

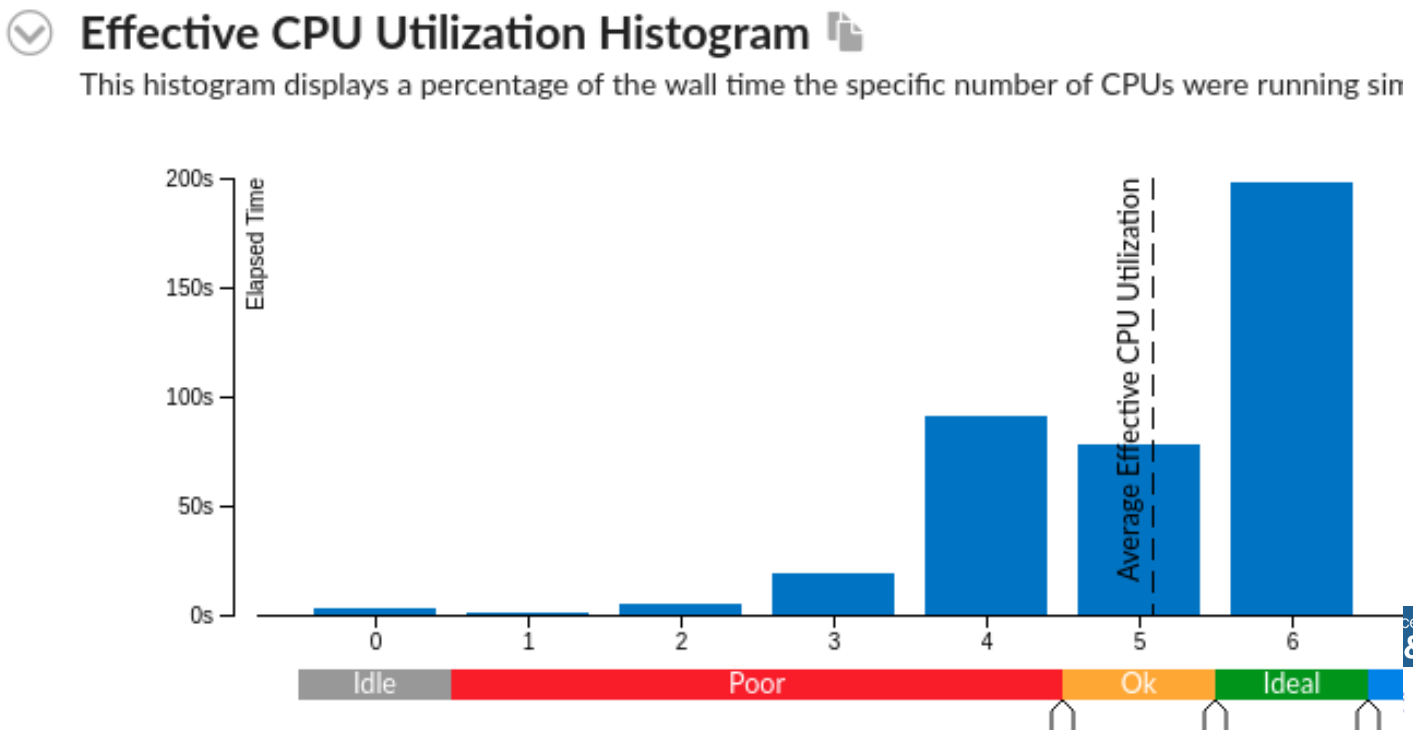
- Results currently only indicative, no code improvements have been made. (Not tested on cluster)
- Profiling aggregates information - great for summaries
- Can't see what's happening instantaneously (tracing)
- Tools also give suggestions to improve code

Intel Vtune Profiler

- Results currently only indicative, no code improvements have been made. (Not tested on cluster)
- Profiling aggregates information - great for summaries
- Can't see what's happening instantaneously (tracing)
- Tools also give suggestions to improve code
- Overhead of profiling tool (395s vs 385s)

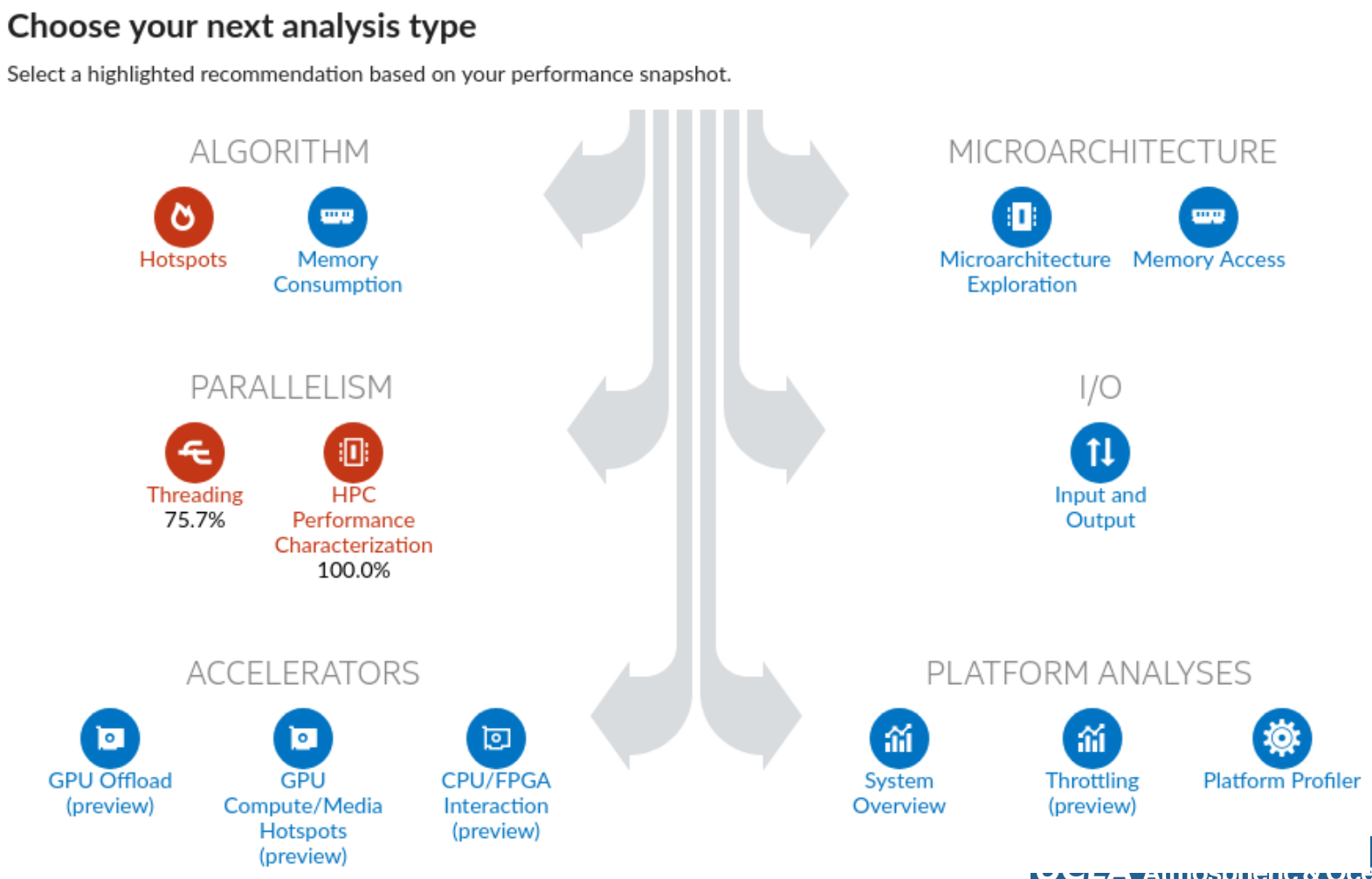
Intel Vtune Profiler

- Results currently only indicative, no code improvements have been made. (Not tested on cluster)
- Profiling aggregates information - great for summaries
- Can't see what's happening instantaneously (tracing)
- Tools also give suggestions to improve code
- Overhead of profiling tool (395s vs 385s)

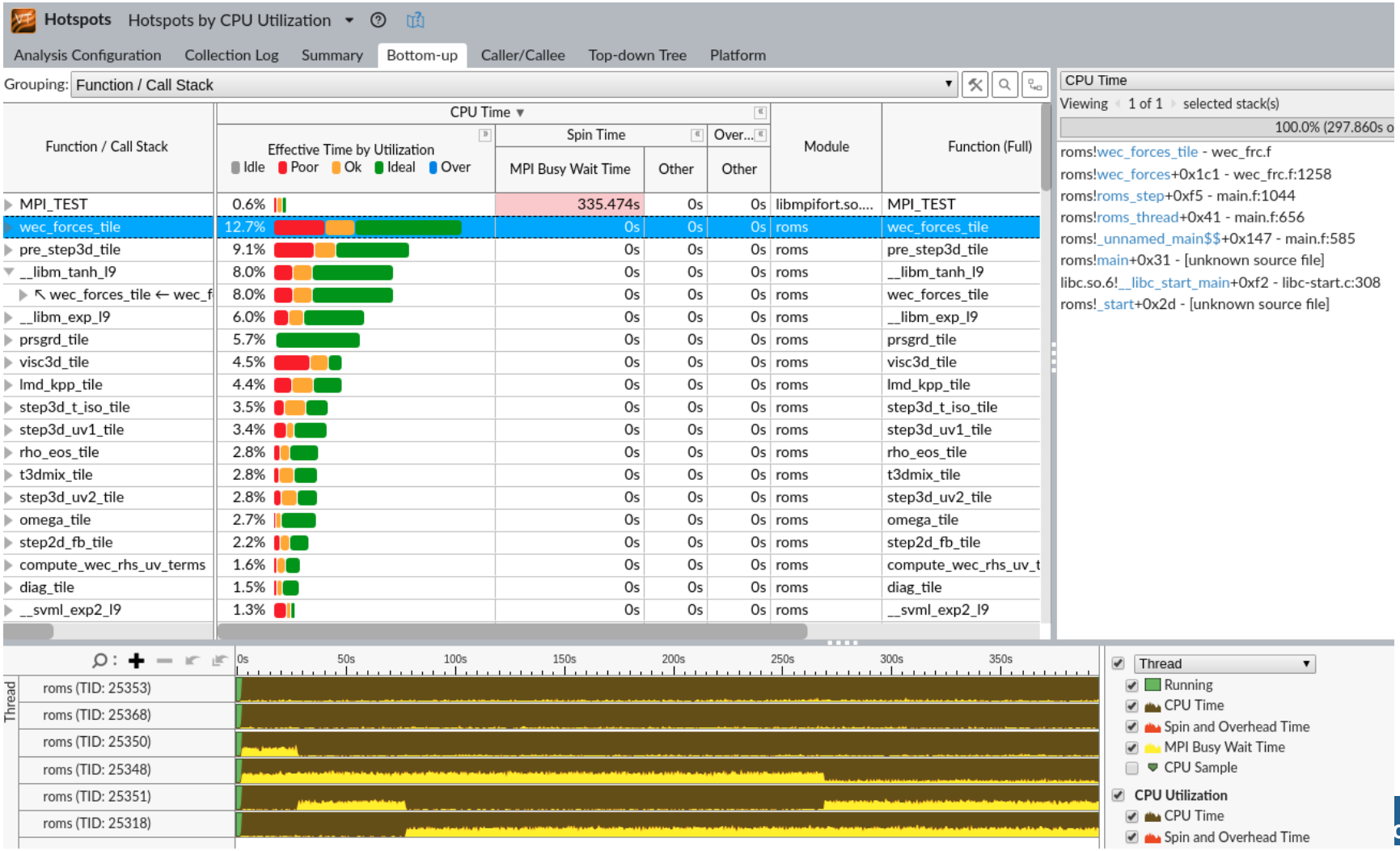


Vtune analysis types

Vtune has many analysis types:



Hotspots


Hotspots
Hotspots by CPU Utilization
Analysis Configuration
Collection Log
Summary
Bottom-up
Caller/Callee
Top-down Tree
Platform
Grouping: Function / Call Stack
Function / Call Stack
CPU Time
Effective Time by Utilization
Idle Poor Ok Ideal Over
Spin Time
MPI Busy Wait Time
Over...
Other
Module
Function (Full)
MPI_TEST 0.6% 335.474s 0s 0s libmpifort.so.... MPI_TEST
wec_forces_tile 12.7% 0s 0s 0s roms wec_forces_tile
pre_step3d_tile 9.1% 0s 0s 0s roms pre_step3d_tile
__libm_tanh_l9 8.0% 0s 0s 0s roms __libm_tanh_l9
↖ wec_forces_tile ← wec_f 8.0% 0s 0s 0s roms wec_forces_tile
__libm_exp_l9 6.0% 0s 0s 0s roms __libm_exp_l9
prsgrd_tile 5.7% 0s 0s 0s roms prsgrd_tile
visc3d_tile 4.5% 0s 0s 0s roms visc3d_tile
lmd_kpp_tile 4.4% 0s 0s 0s roms lmd_kpp_tile
step3d_t_iso_tile 3.5% 0s 0s 0s roms step3d_t_iso_tile
step3d_uv1_tile 3.4% 0s 0s 0s roms step3d_uv1_tile
rho_eos_tile 2.8% 0s 0s 0s roms rho_eos_tile
t3dmix_tile 2.8% 0s 0s 0s roms t3dmix_tile
step3d_uv2_tile 2.8% 0s 0s 0s roms step3d_uv2_tile
omega_tile 2.7% 0s 0s 0s roms omega_tile
step2d_fb_tile 2.2% 0s 0s 0s roms step2d_fb_tile
compute_wec_rhs_uv_terms 1.6% 0s 0s 0s roms compute_wec_rhs_uv_t
diag_tile 1.5% 0s 0s 0s roms diag_tile
__svml_exp2_l9 1.3% 0s 0s 0s roms __svml_exp2_l9
CPU Time
Viewing 1 of 1 selected stack(s)
100.0% (297.860s o
roms!wec_forces_tile - wec_frc.f
roms!wec_forces+0x1c1 - wec_frc.f:1258
roms!roms_step+0xf5 - main.f:1044
roms!roms_thread+0x41 - main.f:656
roms!_unnamed_main$$+0x147 - main.f:585
roms!main+0x31 - [unknown source file]
libc.so.6!__libc_start_main+0xf2 - libc-start.c:308
roms!_start+0x2d - [unknown source file]
0s 50s 100s 150s 200s 250s 300s 350s
Thread
roms (TID: 25353)
roms (TID: 25368)
roms (TID: 25350)
roms (TID: 25348)
roms (TID: 25351)
roms (TID: 25318)
Thread
Running
CPU Time
Spin and Overhead Time
MPI Busy Wait Time
CPU Sample
CPU Utilization
CPU Time
Spin and Overhead Time

Hotspots

- wec forces calculation function (not module) - 34.5% of cpu time!
- tanh - 8% of cpu time

Hotspots — Hotspots by CPU Utilization

Analysis Configuration | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

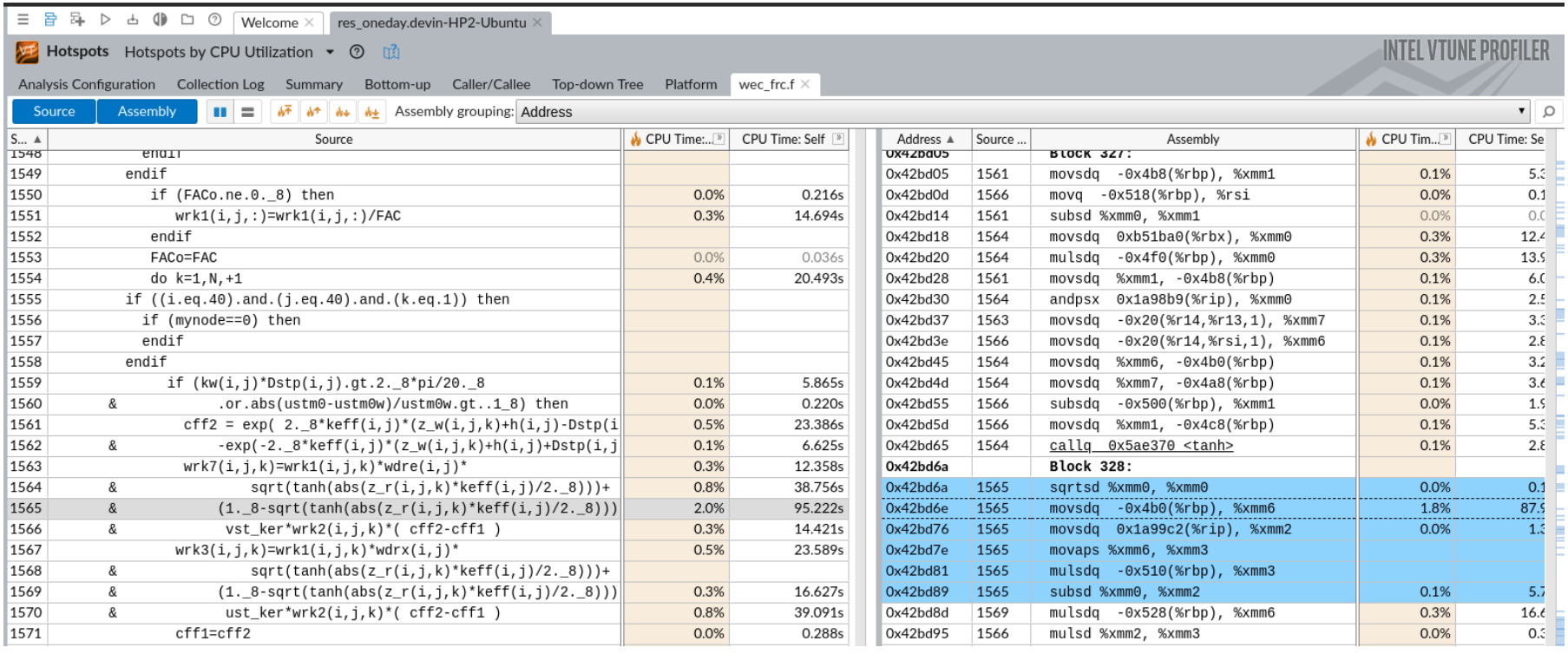
Function	CPU Time: Total	CPU Time: Self	Module	Function (Full)	Source File
_unnamed_main$$	100.0%	0s	roms	_unnamed_main$$	main.f
main	100.0%	0s	roms	main	
_start	100.0%	0s	roms	_start	
__libc_start_main	100.0%	0s	libc.so.6	__libc_start_main	libc-start.c
roms_thread	100.0%	0s	roms	roms_thread	main.f
roms_step	99.9%	0.036s	roms	roms_step	main.f
wec_forces	34.6%	0s	roms	wec_forces	wec_frc.f
wec_forces_tile	34.5%	297.860s	roms	wec_forces_tile	wec_frc.f
MPI_TEST	15.0%	350.161s	libmpifort.so.12	MPI_TEST	
pre_step3d	13.0%	0s	roms	pre_step3d	pre_step3d4S.f
pre_step3d_tile	13.0%	213.420s	roms	pre_step3d_tile	pre_step3d4S.f
__libm_tanh_l9	8.0%	187.354s	roms	__libm_tanh_l9	
lmd_vmix	7.0%	0s	roms	lmd_vmix	lmd_vmix.f
lmd_kpp_tile	6.5%	104.519s	roms	lmd_kpp_tile	lmd_kpp.f
step2d	6.1%	0s	roms	step2d	step2d_FB.f
__libm_exp_l9	6.0%	140.812s	roms	__libm_exp_l9	
mpi_exchange8_3_tile	6.0%	20.147s	roms	mpi_exchange8_3_tile	mpi_exchange8T
prsgrd	5.7%	0.016s	roms	prsgrd	prsgrd.f
prsgrd_tile	5.7%	134.481s	roms	prsgrd_tile	prsgrd.f
exchange2d_4_tile	5.5%	0s	roms	exchange2d_4_tile	exchange.f

Callers	CPU Time: Total	CPU Time: Self
wec_forces_tile	100.0%	297.860s
wec_forces	100.0%	297.860s
[Unknown stack frar	0.0%	0s

Callees	CPU Time: Total	CPU Time: Self
wec_forces_tile	100.0%	297.860s
__libm_tanh_l9	23.1%	187.354s
__libm_exp_l9	17.2%	139.232s
exchange2d_4_tile	16.0%	0s
__svml_exp2_l9	3.6%	29.586s
exp	1.1%	8.799s
mpi_exchange8_3_ti	0.9%	4.799s
tanh	0.4%	3.391s
exchange2d_3_tile	0.4%	0s
__libm_exp_e7	0.3%	2.473s
__svml_pow2_l9	0.1%	1.124s
__libm_cosh_l9	0.1%	0.444s

Hotspots - wec_forces.f

WEC source code and assembly code:



S...	Source	CPU Time:...	CPU Time: Self
1548	endif		
1549	endif		
1550	if (FACo.ne.0._8) then	0.0%	0.216s
1551	wrk1(i,j,:)=wrk1(i,j,:)/FAC	0.3%	14.694s
1552	endif		
1553	FACo=FAC	0.0%	0.036s
1554	do k=1,N,+1	0.4%	20.493s
1555	if ((i.eq.40).and.(j.eq.40).and.(k.eq.1)) then		
1556	if (mynode==0) then		
1557	endif		
1558	endif		
1559	if (kw(i,j)*Dstp(i,j).gt.2._8*pi/20._8	0.1%	5.865s
1560	& .or.abs(ustm0-ustm0w)/ustm0w.gt..1_8) then	0.0%	0.220s
1561	cff2 = exp(2._8*keff(i,j)*(z_w(i,j,k)+h(i,j)-Dstp(i	0.5%	23.386s
1562	& -exp(-2._8*keff(i,j)*(z_w(i,j,k)+h(i,j)+Dstp(i,j	0.1%	6.625s
1563	wrk7(i,j,k)=wrk1(i,j,k)*wdre(i,j)*	0.3%	12.358s
1564	& sqrt(tanh(abs(z_r(i,j,k)*keff(i,j)/2._8)))+	0.8%	38.756s
1565	& (1._8-sqrt(tanh(abs(z_r(i,j,k)*keff(i,j)/2._8)))	2.0%	95.222s
1566	& vst_ker*wrk2(i,j,k)*(cff2-cff1)	0.3%	14.421s
1567	wrk3(i,j,k)=wrk1(i,j,k)*wdrx(i,j)*	0.5%	23.589s
1568	& sqrt(tanh(abs(z_r(i,j,k)*keff(i,j)/2._8)))+		
1569	& (1._8-sqrt(tanh(abs(z_r(i,j,k)*keff(i,j)/2._8)))	0.3%	16.627s
1570	& ust_ker*wrk2(i,j,k)*(cff2-cff1)	0.8%	39.091s
1571	cff1=cff2	0.0%	0.288s

Address	Source ...	Assembly	CPU Tim...	CPU Time: Se
0x42bd05		Block 327:		
0x42bd05	1561	movsdq -0x4b8(%rbp), %xmm1	0.1%	5.3
0x42bd0d	1566	movq -0x518(%rbp), %rsi	0.0%	0.1
0x42bd14	1561	subsd %xmm0, %xmm1	0.0%	0.0
0x42bd18	1564	movsdq 0xb51ba0(%rbx), %xmm0	0.3%	12.4
0x42bd20	1564	mulsdq -0x4f0(%rbp), %xmm0	0.3%	13.9
0x42bd28	1561	movsdq %xmm1, -0x4b8(%rbp)	0.1%	6.0
0x42bd30	1564	andpsx 0x1a98b9(%rip), %xmm0	0.1%	2.5
0x42bd37	1563	movsdq -0x20(%r14,%r13,1), %xmm7	0.1%	3.3
0x42bd3e	1566	movsdq -0x20(%r14,%rsi,1), %xmm6	0.1%	2.8
0x42bd45	1564	movsdq %xmm6, -0x4b0(%rbp)	0.1%	3.2
0x42bd4d	1564	movsdq %xmm7, -0x4a8(%rbp)	0.1%	3.6
0x42bd55	1566	subsdq -0x500(%rbp), %xmm1	0.0%	1.9
0x42bd5d	1566	movsdq %xmm1, -0x4c8(%rbp)	0.1%	5.3
0x42bd65	1564	callq 0x5ae370 <tanh>	0.1%	2.8
0x42bd6a		Block 328:		
0x42bd6a	1565	sqrtsd %xmm0, %xmm0	0.0%	0.1
0x42bd6e	1565	movsdq -0x4b0(%rbp), %xmm6	1.8%	87.9
0x42bd76	1565	movsdq 0x1a99c2(%rip), %xmm2	0.0%	1.3
0x42bd7e	1565	movaps %xmm6, %xmm3		
0x42bd81	1565	mulsdq -0x510(%rbp), %xmm3		
0x42bd89	1565	subsd %xmm0, %xmm2	0.1%	5.7
0x42bd8d	1569	mulsdq -0x528(%rbp), %xmm6	0.3%	16.6
0x42bd95	1566	mulsd %xmm2, %xmm3	0.0%	0.3

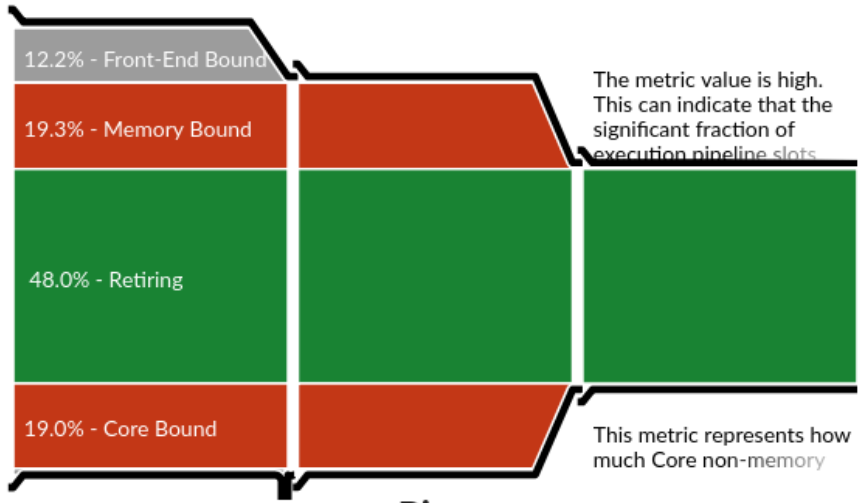
Micro-architecture exploration

Microarchitecture Exploration Microarchitecture Exploration

Analysis Configuration | Collection Log | Summary | Bottom-up | Event Count | Platform

Elapsed Time: 340.532s

Clockticks:	4,641,534,000,000	
Instructions Retired:	5,757,469,200,000	
CPI Rate:	0.806	
MUX Reliability:	1.000	
Retiring:	48.0%	of Pipeline Slots
Front-End Bound:	12.2%	of Pipeline Slots
Bad Speculation:	1.5%	of Pipeline Slots
Back-End Bound:	38.3%	of Pipeline Slots
Memory Bound:	19.3%	of Pipeline Slots
L1 Bound:	9.5%	of Clockticks
L2 Bound:	3.9%	of Clockticks
L3 Bound:	2.8%	of Clockticks
DRAM Bound:	13.2%	of Clockticks
Memory Bandwidth:	29.4%	of Clockticks
Memory Latency:	19.1%	of Clockticks
Store Bound:	1.3%	of Clockticks
Core Bound:	19.0%	of Pipeline Slots
Divider:	8.1%	of Clockticks
Port Utilization:	28.0%	of Clockticks
Cycles of 0 Ports Utilized:	16.4%	of Clockticks
Cycles of 1 Port Utilized:	12.3%	of Clockticks
Cycles of 2 Ports Utilized:	15.5%	of Clockticks
Cycles of 3+ Ports Utilized:	32.1%	of Clockticks
Vector Capacity Usage (FPU):	42.7%	
Average CPU Frequency:	2.3 GHz	
Total Thread Count:	6	
Paused Time:	0s	



µPipe

This diagram represents inefficiencies in CPU usage. Treat it as a pipe with an output flow equal to the "pipe efficiency" ratio: (Actual Instructions Retired)/(Maximum Possible Instruction Retired). If there are pipeline stalls decreasing the pipe efficiency, the pipe shape gets more narrow.

Comparing code performance

- GUI allows direct comparison of runs. (More useful when testing on same hardware and making code changes)

Comparing code performance

- GUI allows direct comparison of runs. (More useful when testing on same hardware and making code changes)
- Zulu faster than laptop but ratios of time spent in routines similar.

Elapsed Time: 394.689s - 228.114s = 166.576s

Total Thread Count:	Not changed, 6
Paused Time:	Not changed, 0s
CPU Time:	2349.190s - 1360.930s = 988.260s

Top Hotspots

This section lists the most active functions in your application. Optimizing th performance.

Function	Module	CPU Time
MPI_TEST	libmpifort.so.12	350.161s - 204.800s = 145.361s
wec_forces_tile	roms	297.860s - 185.187s = 112.673s
pre_step3d_tile	roms	213.420s - 124.948s = 88.472s
__libm_tanh_l9	roms	187.354s - 70.387s = 116.967s
__libm_exp_l9	roms	140.812s - 61.979s = 78.833s
[Others]	N/A*	1159.583s - 713.629s = 445.953s

*N/A is applied to non-summable metrics.

Vtune summary

- WEC force calculation function (not module) needs improving ($\approx$ 35% of CPU time).

Vtune summary

- WEC force calculation function (not module) needs improving ($\approx$ 35% of CPU time).
- New netcdf functionality and module structure functioning ok, no alarming hotspots.

Vtune summary

- WEC force calculation function (not module) needs improving ($\approx$ 35% of CPU time).
- New netcdf functionality and module structure functioning ok, no alarming hotspots.
- MPI wait time we believe is partly a cause of hyper-threading beyond inherent latency.

Vtune summary

- WEC force calculation function (not module) needs improving ($\approx$ 35% of CPU time).
- New netcdf functionality and module structure functioning ok, no alarming hotspots.
- MPI wait time we believe is partly a cause of hyper-threading beyond inherent latency.
- Need to wait for maya (cluster) results for more concrete insights.

Intel Trace Analyzer & Collector (ITAC)

Intel Trace Analyzer & Collector (ITAC)

- Traces code execution on all nodes at any instant in time.

Intel Trace Analyzer & Collector (ITAC)

- Traces code execution on all nodes at any instant in time.
- Can see what process is waiting for another process.

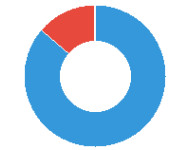
Intel Trace Analyzer & Collector (ITAC)

- Traces code execution on all nodes at any instant in time.
- Can see what process is waiting for another process.
- Particularly useful for optimizing MPI communications.

ITAC

Ratio

This section represents a ratio of all MPI calls to the rest of your code in the application.



■	Serial Code - 100 sec	86.2 %
■	OpenMP - 0 sec	0 %
■	MPI calls - 16 sec	13.7 %

Top MPI functions

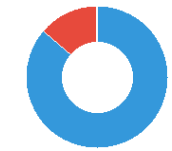
This section lists the most active MPI functions from all MPI calls in the application.

MPI_Test	15.3 sec (13.2 %)
MPI_Isend	0.27 sec (0.233 %)
MPI_Bcast	0.226 sec (0.195 %)
MPI_Irecv	0.0762 sec (0.0657 %)
MPI_Recv	0.0237 sec (0.0205 %)

ITAC

Ratio

This section represents a ratio of all MPI calls to the rest of your code in the application.

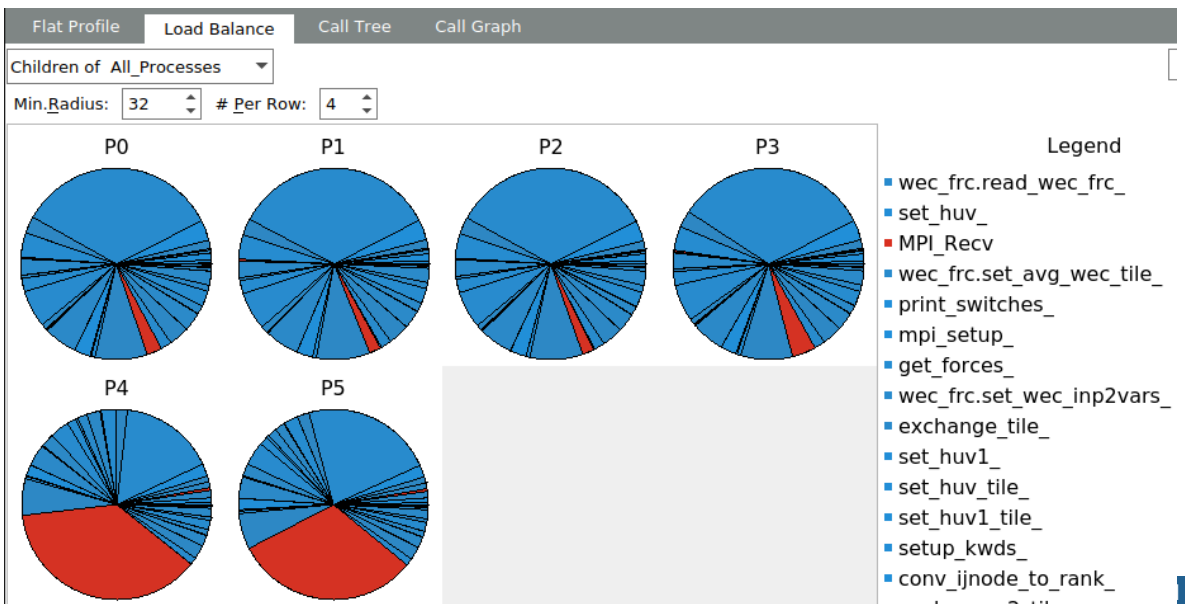


■	Serial Code - 100 sec	86.2 %
■	OpenMP - 0 sec	0 %
■	MPI calls - 16 sec	13.7 %

Top MPI functions

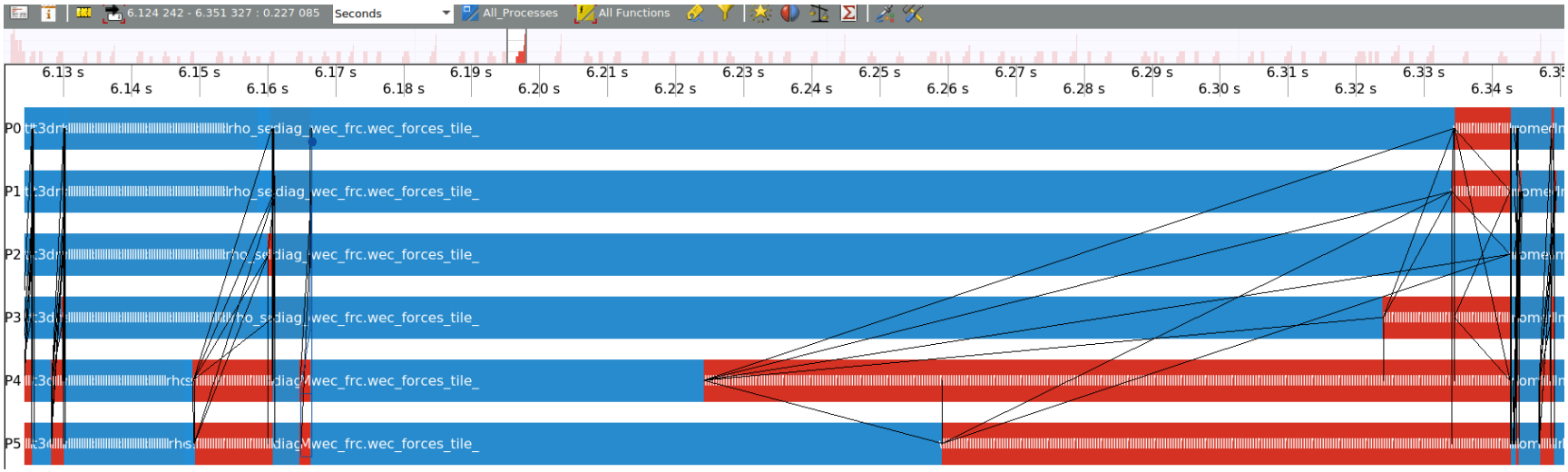
This section lists the most active MPI functions from all MPI calls in the application.

MPI_Test	15.3 sec (13.2 %)
MPI_Isend	0.27 sec (0.233 %)
MPI_Bcast	0.226 sec (0.195 %)
MPI_Irecv	0.0762 sec (0.0657 %)
MPI_Recv	0.0237 sec (0.0205 %)



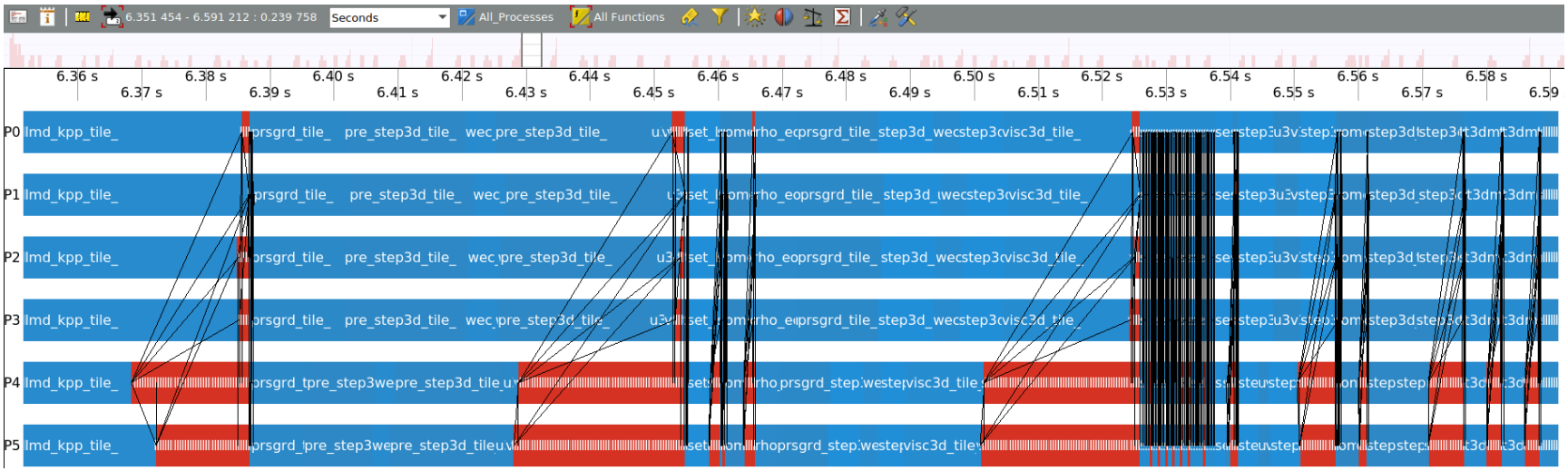
ITAC

First half of time-step:



ITAC

Second half of time-step:



ITAC summary

- Very useful to see what is happening at any moment.

ITAC summary

- Very useful to see what is happening at any moment.
- Waits on 2 processes we believe are due to hyper-threading.

ITAC summary

- Very useful to see what is happening at any moment.
- Waits on 2 processes we believe are due to hyper-threading.
  - Have some tests we can do to test this theory (use fewer cores to enable node per physical core).

ITAC summary

- Very useful to see what is happening at any moment.
- Waits on 2 processes we believe are due to hyper-threading.
  - Have some tests we can do to test this theory (use fewer cores to enable node per physical core).
- Again, most useful on maya (cluster) where memory is not on same CPU.

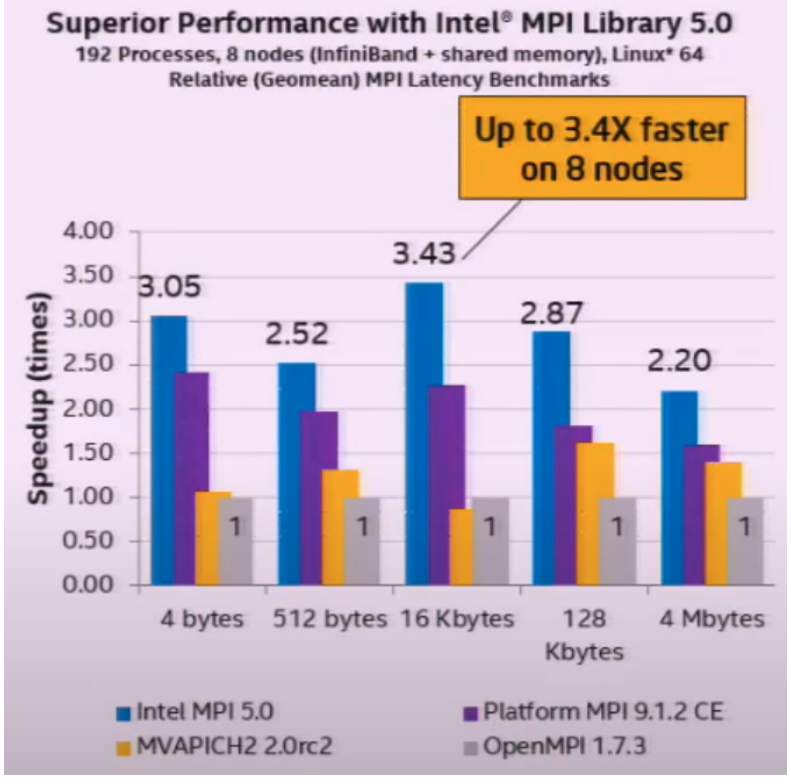
Intel MPI (IMPI)

Intel MPI (IMPI)

- Better compatibility with Intel profiling tools.

Intel MPI (IMPI)

- Better compatibility with Intel profiling tools.
- Potentially better run times with intel processors (maya).



Intel MPI (IMPI)

- To be tested on our hardware and cataloged.

Intel MPI (IMPI)

- To be tested on our hardware and cataloged.
- MPI tuning
  - Tool that runs benchmark examples and suggests compiler flags to optimize MPI for your hardware.

Conclusions & Outlook

Conclusions & Outlook

- Cataloging run times against repo versions and hardware.

Conclusions & Outlook

- Cataloging run times against repo versions and hardware.
- Using tools alongside development in an effective way.

Conclusions & Outlook

- Cataloging run times against repo versions and hardware.
- Using tools alongside development in an effective way.
- Likely increased complexity in collection and analysis on clusters of nodes.

Conclusions & Outlook

- Cataloging run times against repo versions and hardware.
- Using tools alongside development in an effective way.
- Likely increased complexity in collection and analysis on clusters of nodes.
- Get latest intel parallel studio (ifort, profile tools, intel MPI) on CESR clusters (maya, etc)

The End