An Introduction to Profiling with Intel Trace Collector and Analyzer (ITAC)

Ayesha Afzal, Georg Hager, Gerhard Wellein (NHR@FAU)

ayesha.afzal@fau.de

hpc@fau.de

User and Reference Guide

https://www.intel.com/content/www/us/en/developer/tools/oneapi/trace-analyzer.html

Tools Poll

- Intel
  - Advisor
  - VTune Amplifier
  - Application Performance Snapshot (APS)
  - **Trace Analyzer and Collector (ITAC)**
- Non-Intel
  - **GNU gprof profiler**
  - NHR@FAU LIKWID
    ("Like I Knew What I'm Doing")
  - VI-HPS Score-P
    (A Joint Performance Measurement Infrastructure for Scalasca, Periscope, TAU, and Vampir)
  - BSC Paraver
  - …

User and Reference Guides of Tools

- **Intel Trace Analyzer and Collector**
  https://www.intel.com/content/www/us/en/docs/trace-analyzer-collector/user-guide-reference/2021-10/overview.html
  https://www.fau.tv/clip/id/19956

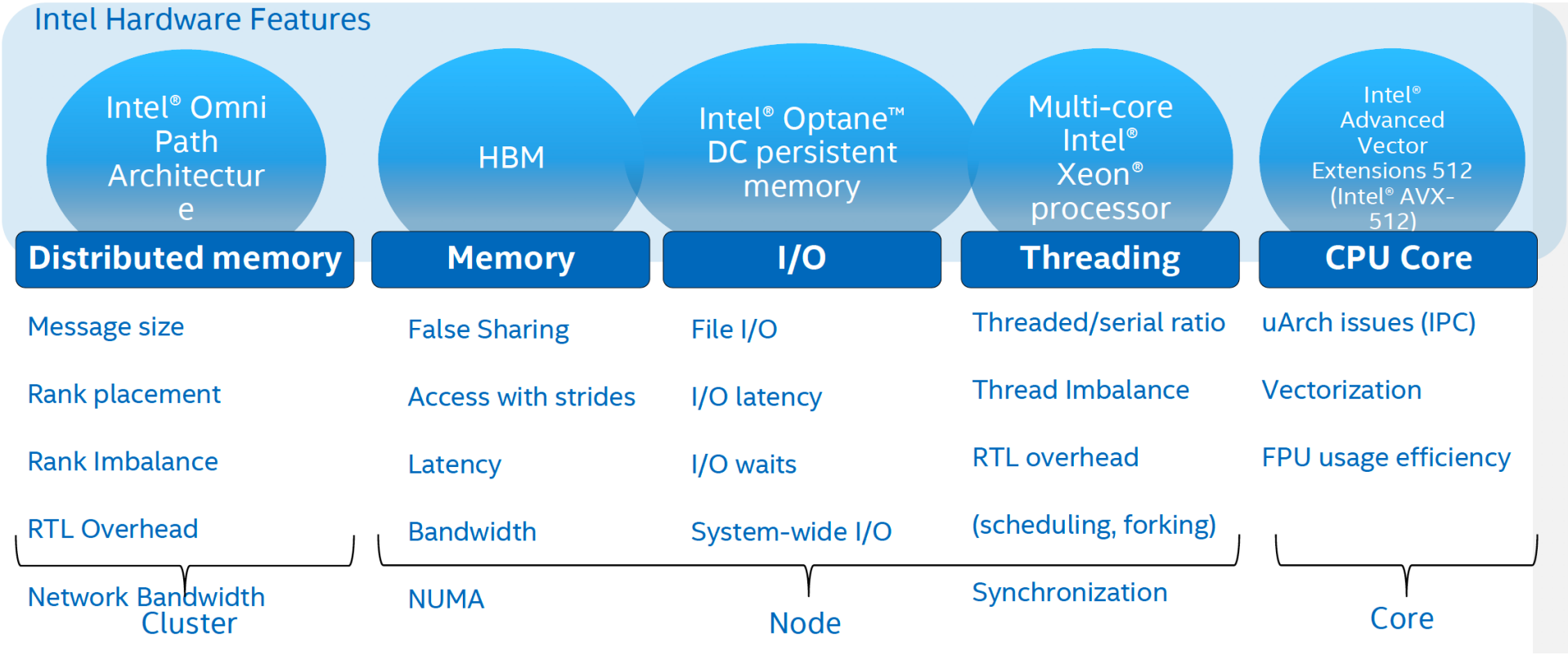
- **Intel VTune Profiler Performance Analysis Cookbook**
  https://www.intel.com/content/www/us/en/docs/vtune-profiler/cookbook/2024-0/overview.html

- **Score-P: A Joint Performance Measurement Infrastructure for Scalasca, Periscope, TAU, and Vampir**
  https://vi-hps.org/training/documentation

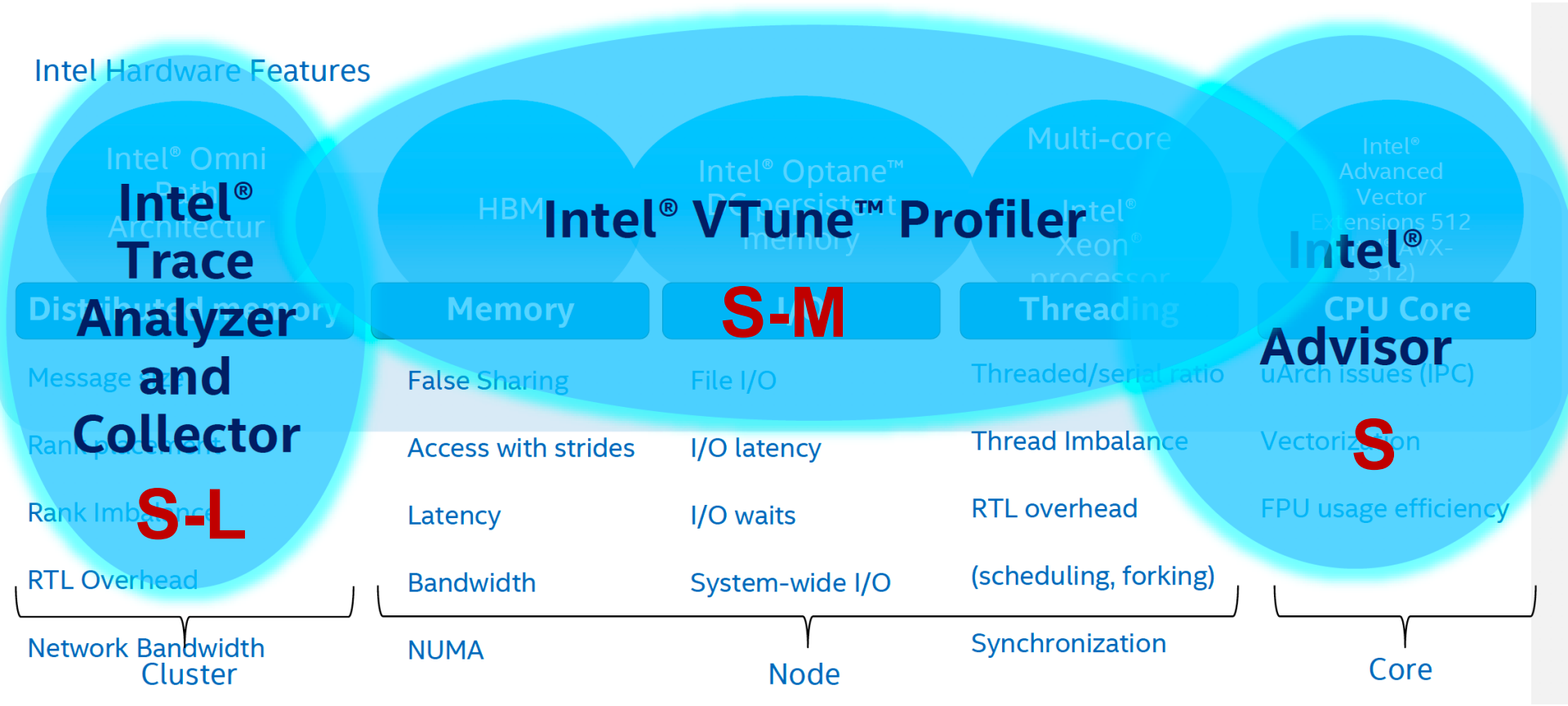
- **Paraver Performance Analysis Tool**
  https://tools.bsc.es/paraver

- **Arm DDT Debugger**
  https://developer.arm.com/documentation/101136/22-1-2/DDT
  https://fau.tv/clip/id/19957

Overview


Intel Hardware Features
Intel® Omni Path Architecture
HBM
Intel® Optane™ DC persistent memory
Multi-core Intel® Xeon® processor
Intel® Advanced Vector Extensions 512 (Intel® AVX-512)
Distributed memory
Memory
I/O
Threading
CPU Core
Message size
Rank placement
Rank Imbalance
RTL Overhead
Network Bandwidth
False Sharing
Access with strides
Latency
Bandwidth
NUMA
File I/O
I/O latency
I/O waits
System-wide I/O
Threaded/serial ratio
Thread Imbalance
RTL overhead
(scheduling, forking)
Synchronization
uArch issues (IPC)
Vectorization
FPU usage efficiency
Cluster
Node
Core
© Intel

Overview



© Intel

Maximum collection times: L=long (hours) M=medium (minutes) S=short (seconds-few minutes)

Application focus

Intel Trace Collector and Analyzer

Environment preparation

$ module load **itac intel intelmpi**

-trace

Collector: compilation, linking and run

$ mpiicc -trace myProg.c -o ${binary}

$ mpirun -n 4 ./${binary}

format

STF: Structured Trace File

Analyzer

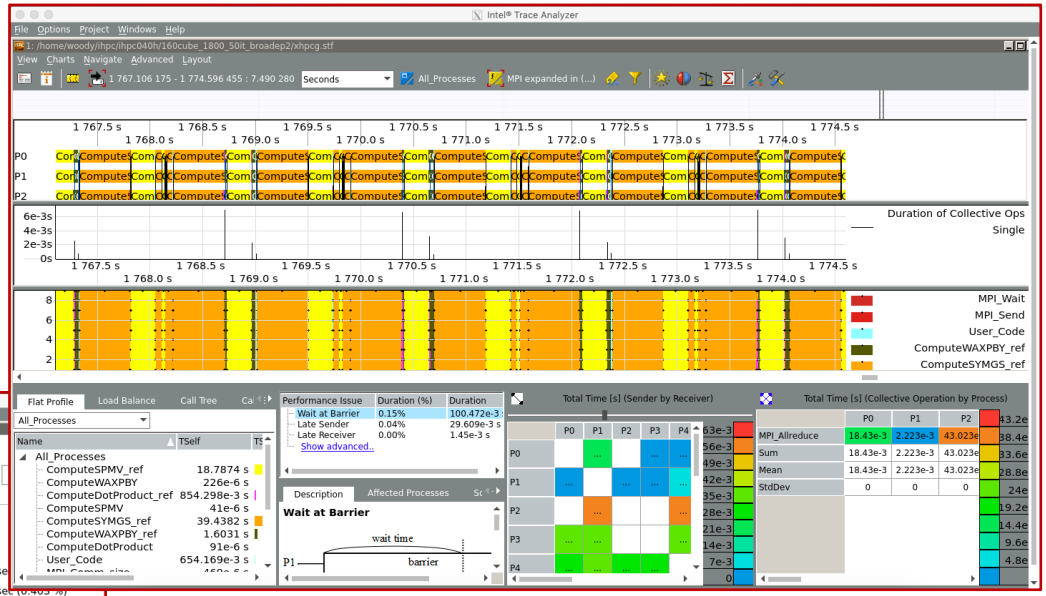
$ traceanalyzer ${binary}.stf &

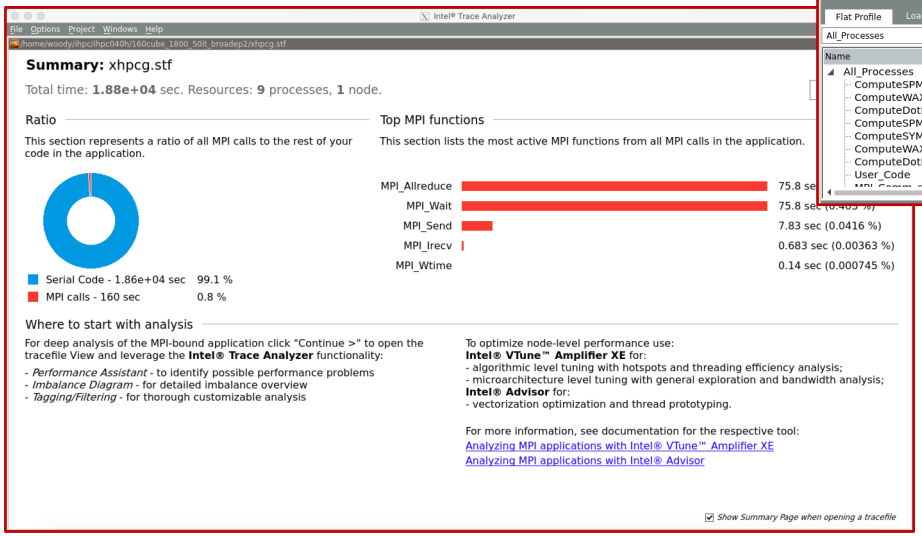
Basic features

Profiling with graphical visualization and statistical data

Intel Trace Collector and Analyzer

Event-based approach that record
- user function calls
- MPI communication calls

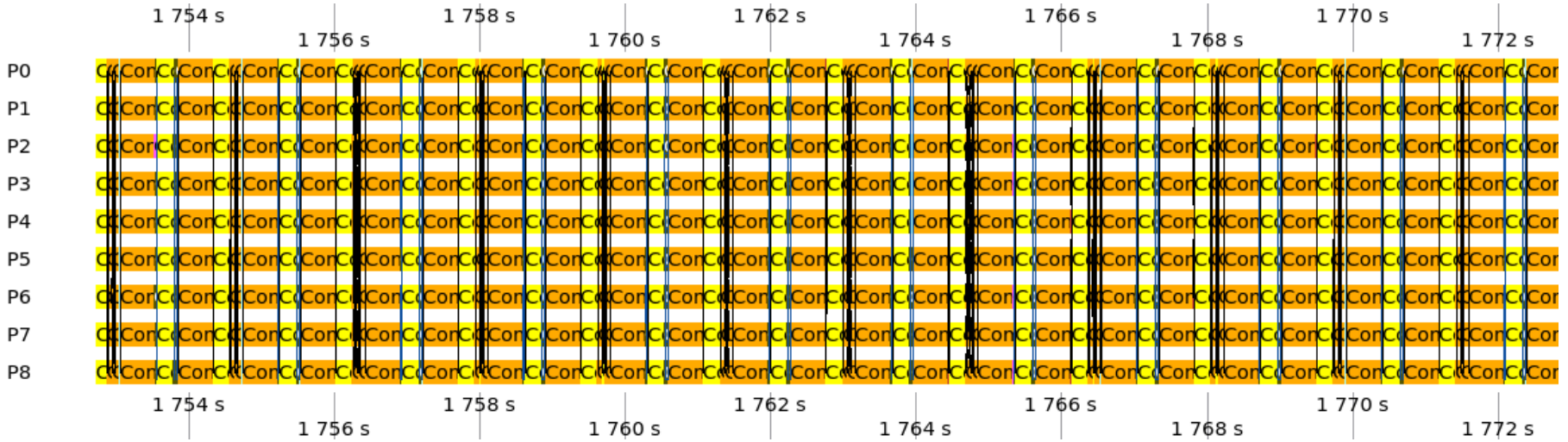




Intel Trace Collector and Analyzer

Event timeline

- program structure
- event details
  - functions
  - messages
  - collectives operations



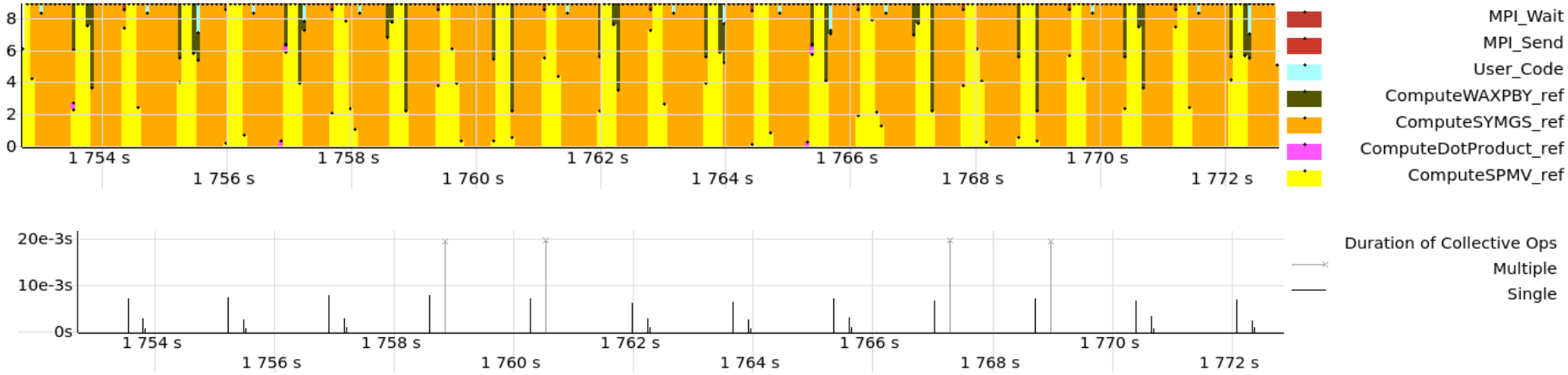
Intel Trace Collector and Analyzer

Event timeline

- program structure
- event details
  - functions
  - messages
  - collectives operations

Quantitative + quantitative timeline

- transfer duration
- transfer volume
- activities measure



Intel Trace Collector and Analyzer

Event timeline

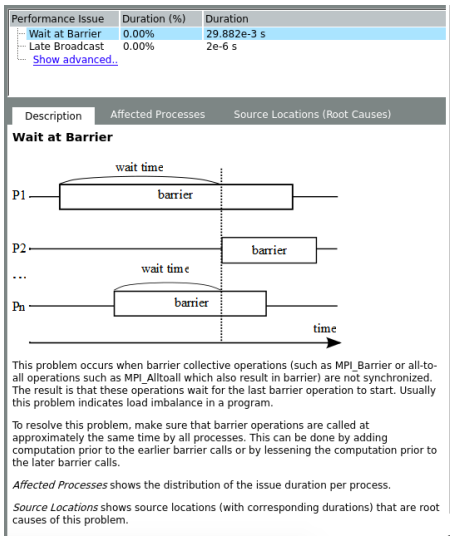
- program structure
- event details
  - functions
  - messages
  - collectives operations

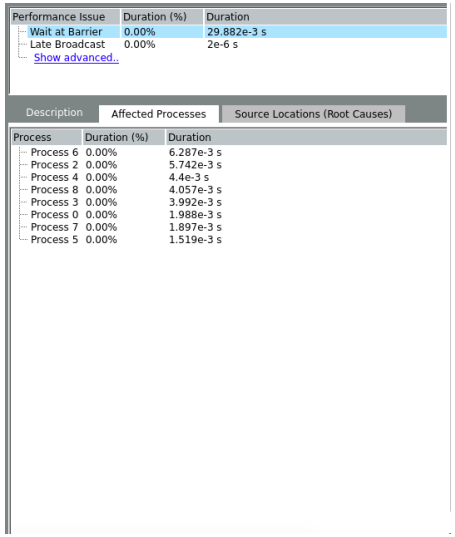
Quantitative + quantitative timeline

- transfer duration
- transfer volume
- activities measure

Performance assistance

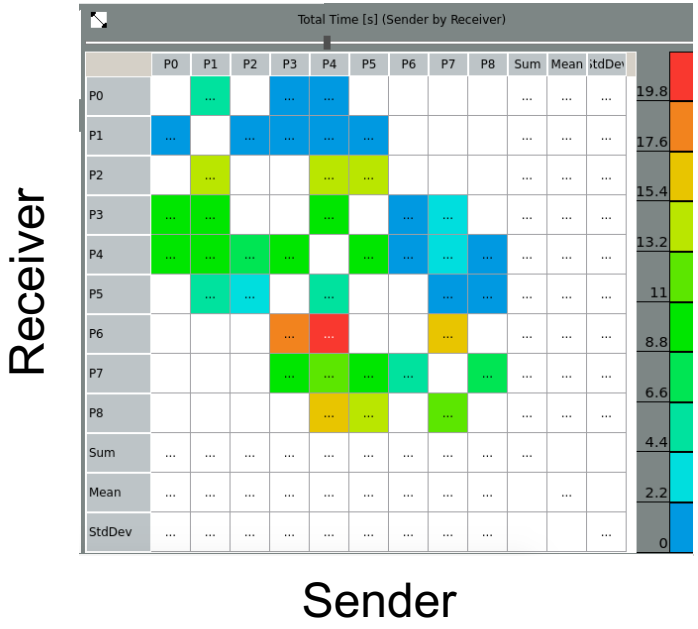
- identify performance issues with explanation
- tips on potential solutions





Intel Trace Collector and Analyzer

Event timeline	Quantitative + quantitative timeline	Performance assistance	Message profile
•program structure •event details •functions •messages •collectives operations	•transfer duration •transfer volume •activities measure	•identify performance issues with explanation •tips on potential solutions	•message volume, time, count •finding communication hotspots •communication patterns

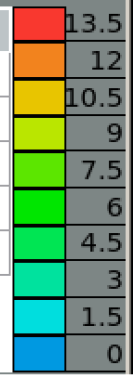


Intel Trace Collector and Analyzer

Event timeline	Quantitative + quantitative timeline	Performance assistance	Message profile	Collective ops profile
•program structure •event details •functions •messages •collectives operations	•transfer duration •transfer volume •activities measure	•identify performance issues with explanation •tips on potential solutions	•message volume, time, count •finding communication hotspots •communication patterns	•all collective operations for each process

Total Time [s] (Collective Operation by Process)

	P0	P1	P2	P3	P4	P5	P6	P7	P8	Sum	Mean	StdDev
MPI_Bcast	5e-6	7e-6	7e-6	7e-6	7e-6	7e-6	6e-6	7e-6	7e-6	60e-6	6.66667e-6	666.667e-9
MPI_Allreduce	6.98827	2.41008	14.1332	9.46671	9.80818	2.28141	12.1689	7.89127	10.6684	75.8164	8.42405	3.81376
Sum	6.98828	2.41009	14.1332	9.46671	9.80818	2.28142	12.1689	7.89127	10.6684	75.8165		
Mean	3.49414	1.20504	7.06659	4.73336	4.90409	1.14071	6.08444	3.94564	5.33422		4.21203	
StdDev	3.49413	1.20504	7.06658	4.73335	4.90409	1.1407	6.08444	3.94563	5.33422			5.00135



Intel Trace Collector and Analyzer

Hotspot

Event timeline
- program structure
- event details

Quantitative + quantitative timeline
- transfer duration
- transfer

Performance assistance
- identify performance issues with explanation

Message profile
- message volume, time, count

Collective ops profile
- all collective operations for each process

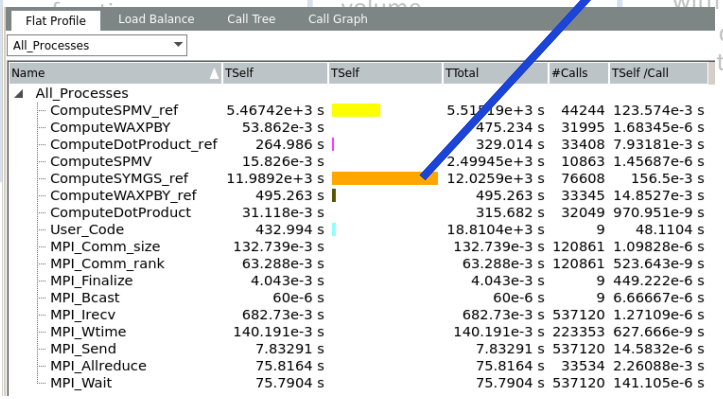
Function profile

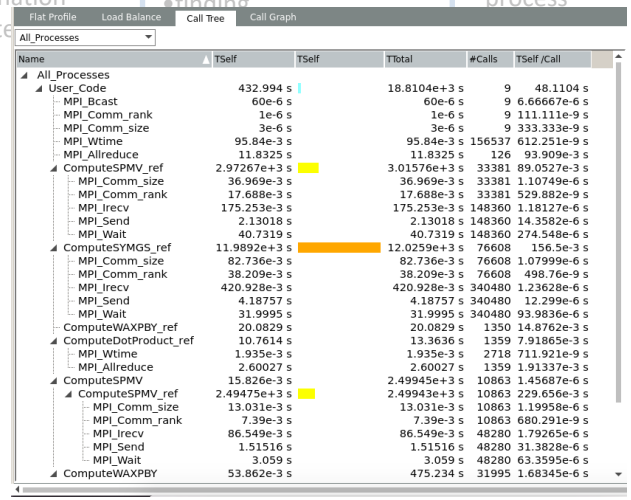
functions statistics

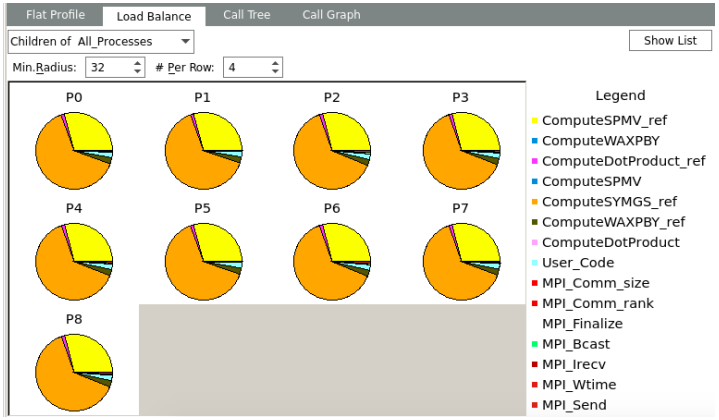
Call tree
- split to children of Group All_Processes
- filtered view with expansion into various MPI and Application groups

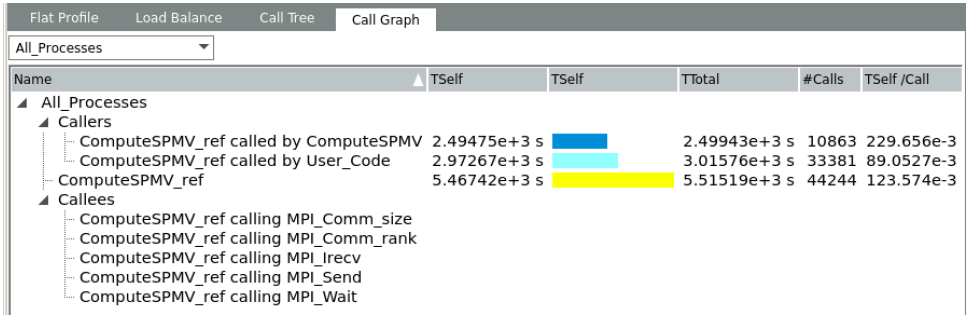
Load balance

Call graph









Event timeline visualization

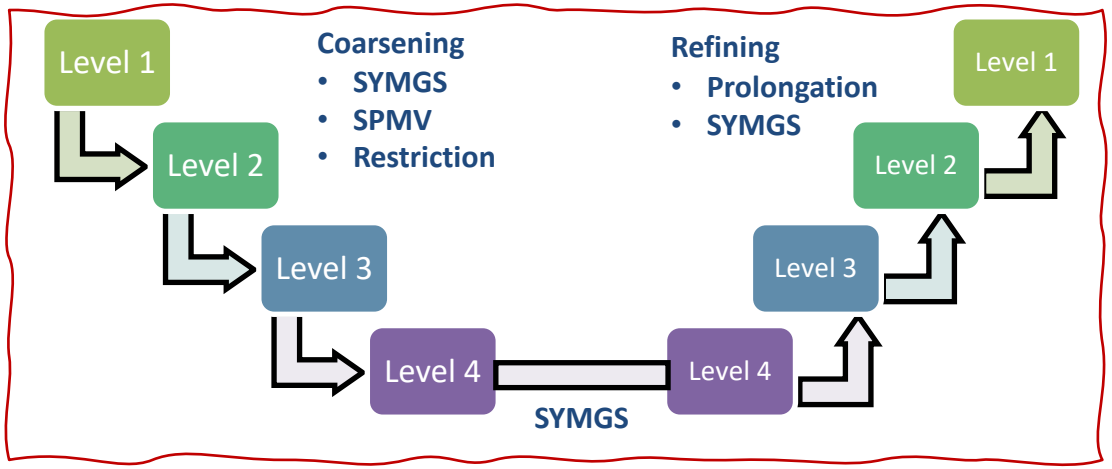
MPI-parallel HPCG

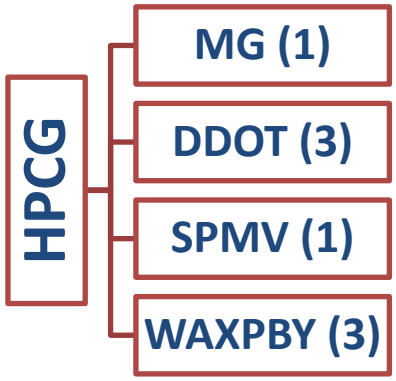
Algorithmic overview: preconditioned Conjugate Gradient

$\vec{p}_0 \leftarrow \vec{x}_0, \qquad \vec{r}_0 \leftarrow \vec{b} - A\vec{p}_0$

for $i = 1,2,$ *to* *max_iterations* **do**

- $\vec{z}_i \leftarrow M^{-1}\vec{r}_{i-1}$
- **if** $i = 1$ **then**
  - $\vec{p}_i \leftarrow \vec{z}_i$
  - $\alpha_i \leftarrow \text{dot\_prod}(\vec{r}_{i-1}, \vec{z}_i)$
- **else**
  - $\alpha_i \leftarrow \text{dot\_prod}(\vec{r}_{i-1}, \vec{z}_i)$
  - $\beta_i \leftarrow \alpha_i / \alpha_{i-1}$
  - $p_i \leftarrow \beta_i \vec{p}_{i-1} + \vec{z}_i$
- $\alpha_i \leftarrow \text{dot\_prod}(\vec{r}_{i-1}, \vec{z}_i) / \text{dot\_prod}(\vec{p}_i, A\vec{p}_i)$
- $\vec{x}_{i+1} \leftarrow \vec{x}_i + \alpha_i \vec{p}_i$
- $\vec{r}_i \leftarrow \vec{r}_{i-1} - \alpha_i A\vec{p}_i$
- **if** $\|\vec{r}_i\|_2 <$ *tolerance* **then**
  - STOP





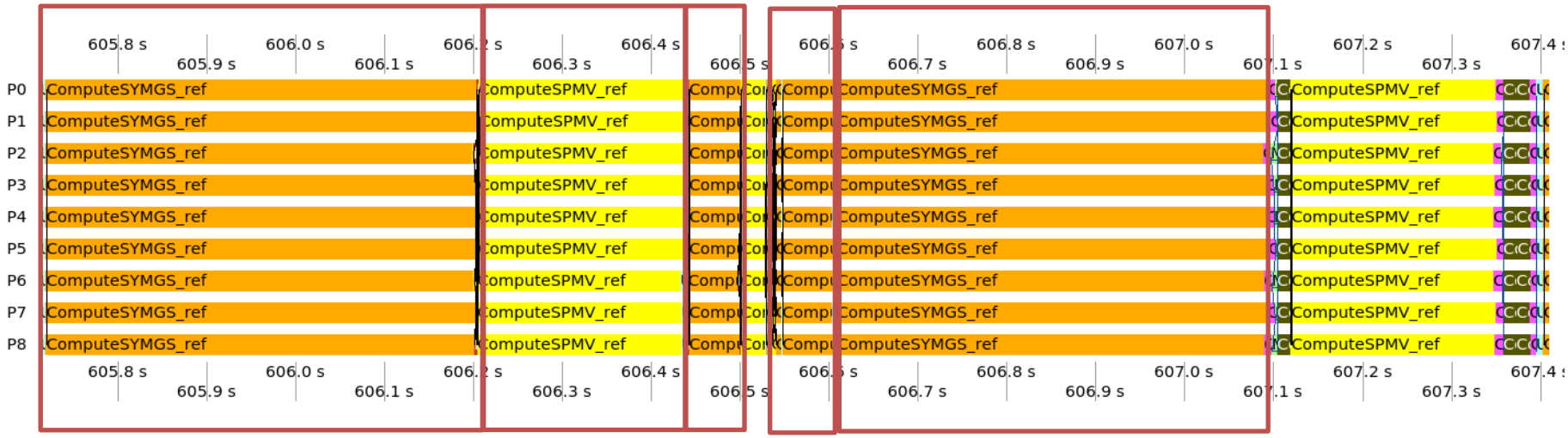
https://www.hpcg-benchmark.org

MPI-parallelizable preconditioned CG

12 user-defined functions + MPI communication

- SYMGS1+SPMV1+restriction+prolongation+SYMGS2

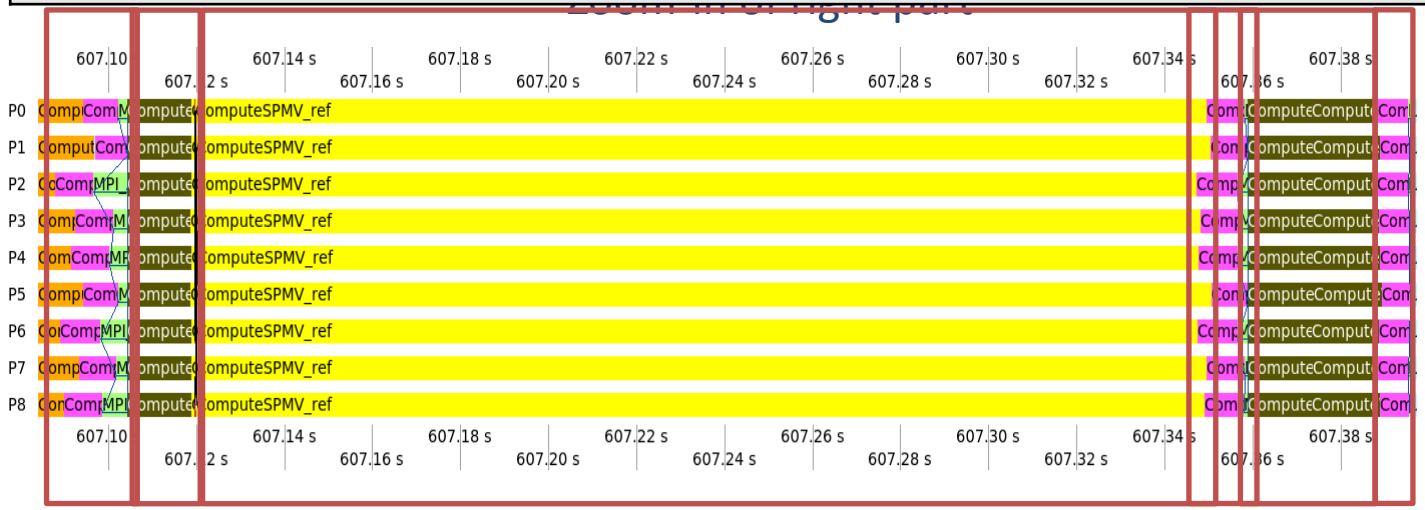
Default
Blue = Computation (user code)
Red = Communication (MPI)
Black lines: Communication dependencies

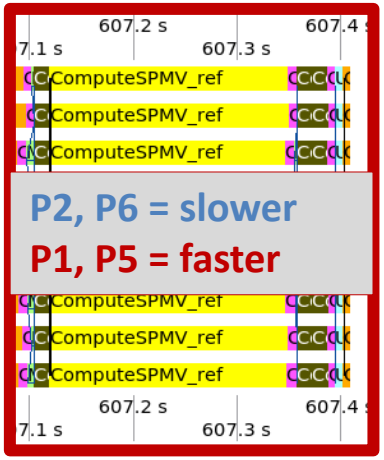


160^3 problem size , 1800 second run, Broadwell EP, one socket, 9 processes, 3 GB memory per process

MPI-parallelizable preconditioned CG

Broadwell EP	Min (ms)	Max (ms)	Mean (ms)	Median (ms)	Variance (ms)	Skewness (ms)	Kurtosis (ms)	SD (ms)	IQR=Q3-Q1 (ms)
DDOT1	7.355	9.045	8.398	8.474	0.272	-0.913	0.697	0.521	0.663
DDOT2	8.300	9.270	8.793	8.713	0.109	-0.080	-0.880	0.330	0.305
DDOT3	6.543	6.890	6.725	6.688	0.013	0.129	-0.607	0.112	0.139

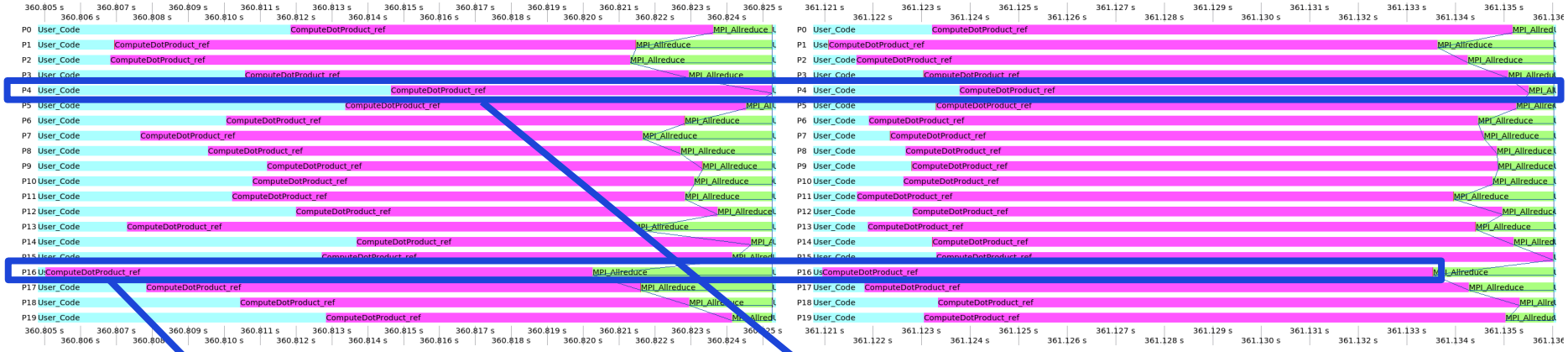




160^3 problem size , 1800 second run, Broadwell EP, one socket, 9 processes, 3 GB memory per process

Preconditioned CG on Cascade Lake SP

Cascade Lake EP	Min (ms)	Max (ms)	Mean (ms)	Median (ms)	Variance (ms)	Skewness (ms)	Kurtosis (ms)	SD (ms)	IQR=Q3-Q1 (ms)
DDOT1	10.592	15.225	12.635	12.426	1.761	0.372	-0.880	1.327	2.162
DDOT2	11.728	12.577	12.233	12.170	0.078	0.156	-1.142	0.279	0.536
DDOT3	7.603	8.008	7.723	7.686	0.014	1.364	0.909	0.116	0.111



Usage with cautions!

Compiler switches, API

Be careful

check pinning correctness ✓

$ impi_info
$ export I_MPI_DEBUG=4

use filtering options for large problems ✓

If not handled carefully,
generates a lot of unnecessary data

Reduce startup time

$ traceanalyzer --cli ${binary}.stf -c0 –w

VT API

Manually instrument the code to profile only interesting parts of an application or a subset of iterations

- run without ”-trace”
- #include <VT.h>
- -I${VT_ROOT}/include
- inserting calls to VT_traceoff() and VT_traceon()
- VT_begin(mark), VT_end(mark)

Compiler switches

1. `-trace`
2. `-tcollect -trace`
   function profile at a high price of maximum overhead
3. `-tcollect-filter=func.txt -tcollect -trace`

   `$cat func.txt:  'funcX' ON|OFF`

   ```
   '.*' OFF
   '.*ComputeDotProduct.*' ON
   '.*ComputeSYMGS.*' ON
   '.*ComputeSPMV.*' ON
   '.*ComputeWAXPBY.*' ON
   ```

1. `-qopt-report`
   generate a full list of file and function names that the compiler recognizes from the compilation unit
2. `-tcollect -qopt-report -qopt-report-phase=tcollect`
   generate an optimization report with tcollect information to get a list of the file or routine strings that can be matched by the regular expression filters

Binary instrumentation at runtime

1. `-trace-imbalance`
   trace only the MPI functions that cause application load imbalance (idle at some point of the application run)
2. `-trace-collectives`
   trace only about collectives operations
3. `-trace-pt2pt`
   trace only about point-to-point operations

Environment variables

Environment variables	Default	Description
VT_FLUSH_PREFIX	/tmp	control directly for temporary flush files
VT_LOGFILE_PREFIX	current working directory	control directly for physical trace information files
VT_LOGFILE_FORMAT	STF	`SINGLESTF`: rolls all trace files into one file (.single.stf)
VT_LOGFILE_NAME	${binary}	control the name for the trace file
VT_MEM_BLOCKSIZE	64 KB	trace data in chunks of main memory
VT_MEM_FLUSHBLOCKS	1024	flushing is started when the number of blocks in memory exceeds this threshold
VT_MEM_MAXBLOCKS	1024	maximum number of blocks in main memory, if exceed the application is stopped until AUTOFLUSH/ MEM-OVERWRITE/ stop recording trace info
VT_CONFIG_RANK	0	control the process that reads and parses the configuration file

https://www.intel.com/content/www/us/en/docs/trace-analyzer-collector/user-guide-reference/2021-10/configuration-options.html

Environment variables: set up in the
1. corresponding environment variables
2. command line when running your application
3. configuration file

$ export VT_CONFIG=<config_file>

```
# enable all Application activities
ACTIVITY Application ON

# disable all MPI activity
ACTIVITY MPI OFF

# enable all bcasts, barrier, allreduce, recvs and sends
SYMBOL MPI_WAITALL ON
SYMBOL MPI_IRECV ON
SYMBOL MPI_ISEND ON
SYMBOL MPI_BARRIER ON
SYMBOL MPI_ALLREDUCE ON
```

Use ITAC wisely with custom filtering preferences