Performance Analysis of Parallel Applications Using LTTng & Trace Compass

Naser Ezzati

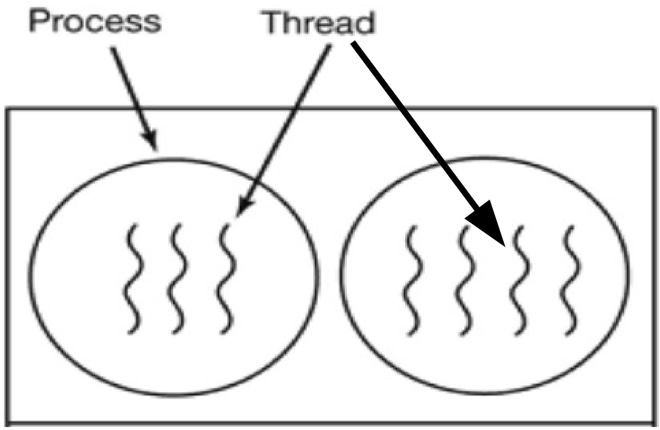
DORSAL LAB
Progress Report Meeting
Polytechnique Montreal
Dec 2017

What is MPI?

- Message Passing Interface (MPI)
  - Industry-wide standard protocol for passing messages between parallel processors.
  - MPI is for communication among processes with separate address spaces.
  - Interprocess communication consists of
    - Synchronization
    - Movement of data from one process’s address space to another’s.

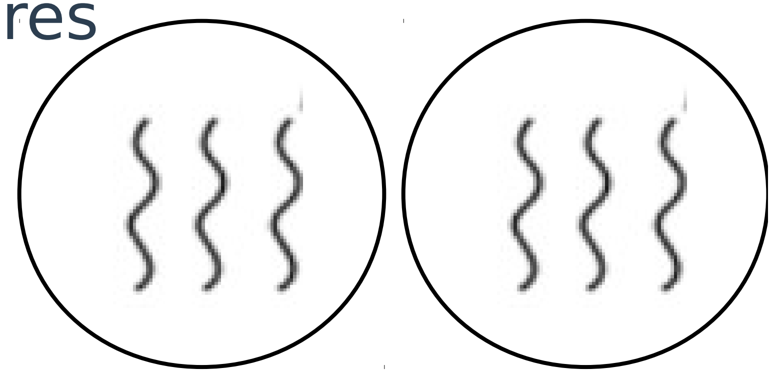
MPI vs Threads

- Thread parallelism provides a shared memory model within a process.
  - Pthread: an execution model allows a program to control multiple different overlapping flows of work.
  - OpenMP: looplevel parallelism, created and managed by the compiler, based on user directives.
- MPI describes parallelism between processes (with separate address spaces).
    - MPICH, OpenMPI



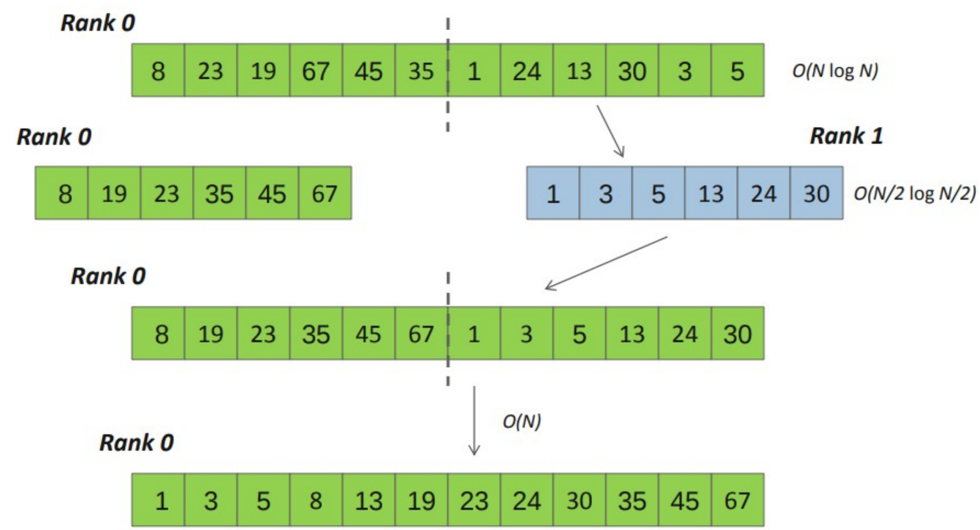
MPI + Threads

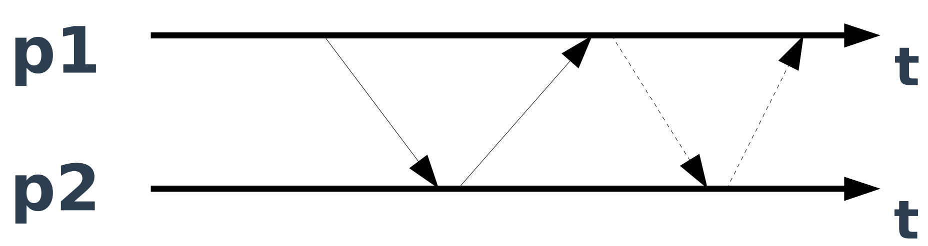
- **MPI-only:**
  - Single thread of execution
    - System resources do not scale at the same rate as processing cores
- **MPI+threads:**
  - Multiple threads executing simultaneously
    - Allows sharing of system resources



Parallel Sort Using MPI

- #include <mpi.h>
- #include <stdio.h>
- int main(int argc, char ** argv)
- {
- int rank, a[1000], b[500];
- MPI_Init(&argc, &argv);
- MPI_Comm_rank(MPI_COMM_WORLD, &rank);
- if (rank == 0) {
  - MPI_Send(&a[500], 500, MPI_INT, 1, 0, MPI_COMM_WORLD);
  - sort(a, 500);
  - MPI_Recv(b, 500, MPI_INT, 1, 0, MPI_COMM_WORLD,
  - MPI_STATUS_IGNORE);
  - /* Serial: Merge array b and sorted part of array a */
- }
- else if (rank == 1) {
  - MPI_Recv(b, 500, MPI_INT, 0, 0, MPI_COMM_WORLD,
  - MPI_STATUS_IGNORE);
  - sort(b, 500);
  - MPI_Send(b, 500, MPI_INT, 0, 0, MPI_COMM_WORLD);
- }
- MPI_Finalize(); return 0;
- }





MPI Performance Analysis

- **Debugging can be difficult!**
  - In a serial program, you can find slow functions (by profiling, tracing, etc.) and optimize them.
- **Investigating of all participating processes**
  - In MPI, if MPI_Recv() takes a lot of time in one process, we should look at other processes, as well as the problematic function/process.
    - What happens on other processes, when this process/function is slow?
  - The synchronization among the processes can be a concern and should be debugged.

MPI Performance Analysis (contd)

- **Network performance is important:**
  - Latency: The time from when a Send is initiated until the first byte is received by a Receive.
  - Bandwidth: The rate at which a sender is able to send data to a receiver.
- **Profilers:**
  - Aggregate statistics at run time
    - Amount of time spent in MPI functions, number of messages sent, etc.
- **Tracers**
  - Collect events from different parts of the execution.

PMPI: MPI Profiling

- **PMPI refers to the MPI standard profiling interface.**
  - Profiling layer of MPI
  - Implemented via additional API in MPI library
  - Different name: PMPI_Init()
    - Same functionality as MPI_Init()
- **Each standard MPI function can be called with an MPI_ or PMPI_ prefix.**
  - MPI_Send() or PMPI_Send().
- **This allows one to write functions with the MPI_ prefix that call the equivalent PMPI_ function.**

```
Int MPI_Init(…)
{
trace_entry();
  PMPI_Init(…);
trace_exit();
}
```

PMPI: MPI Profiling (contd)

- **User may choose subset of MPI routines to be profiled**
  - Useful for building performance analysis tools
  - LD_PRELOAD or dynamic linking
- **Tools**
  - Vampir: Timeline of MPI traffic (Etnus, Inc.)
  - Paradyn: Performance analysis (U. Wisconsin)
  - mpiP: J. Vetter (LLNL)
  - ScalaTrace: F. Mueller et al. (NCSU)
  - Paraver
  - Jumpshot
  - LTTng & Trace Compass ?

MPI Tracing with LTTng

```
MPI_Send(…)
{
  Tracepoint (mpi,mpi_send_entry,…)
  PMPI_Send(…)
  Tracepoint (mpi,mpi_send_exit,…)
}
```

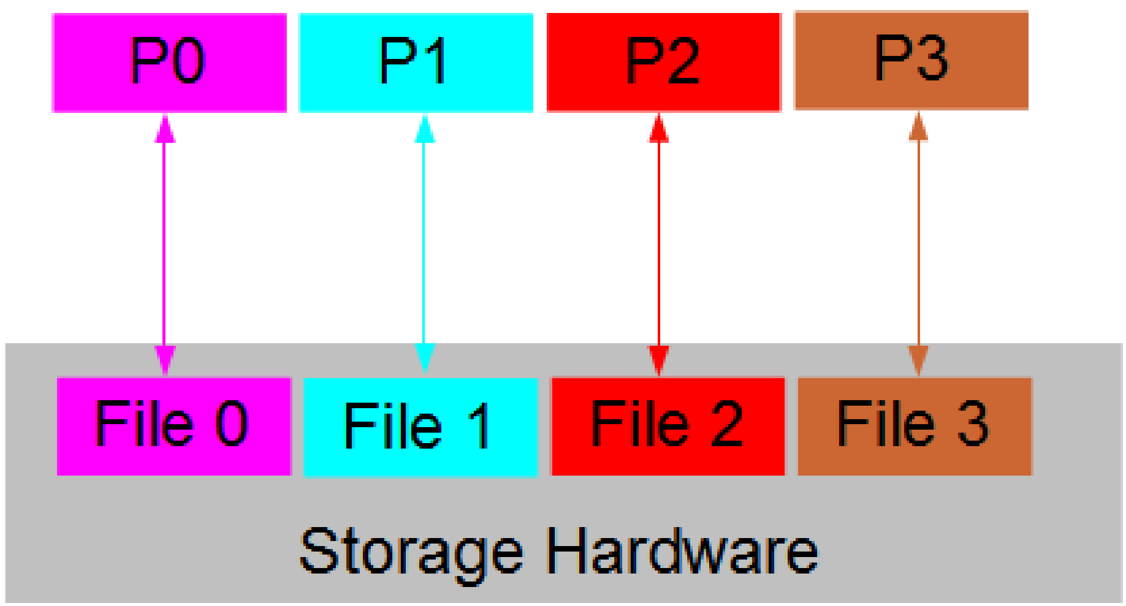
- **Compile and LD_Preload it**
  - LD_Preload is to intercept methods
  - Load this before all of other libraries

LTTng-UST Tracepoints

- **MPI (50 methods)**
  - **Environment Management Functions** (MPI_Init, MPI_Finalize MPI_Abort, etc.)
  - **Point to Point Communication Functions** (MPI_Send, MPI_Recv, MPI_Isend,MPI_Irecv, MPI_Ssend, MPI_Wait*, etc.)
  - **Collective Communication Functions** (MPI_Barrier, MPI_Bcast, MPI_Scatter, MPI_Gather, MPI_Allgather, MPI_Reduce, MPI_Allreduce,
  - **MPI-IO Functions** (MPI_File_open, MPI_File_read, MPI_File_write, MPI_File_close, etc.)
  - **HDF5 (20 methods)**
    - Hierarchical Data Format functions
  - **Pthread (20 methods)**
  - Thread management function
    - Thread create, join, etc.
  - Locks
    - Spin locks,Mutex, R/W locks

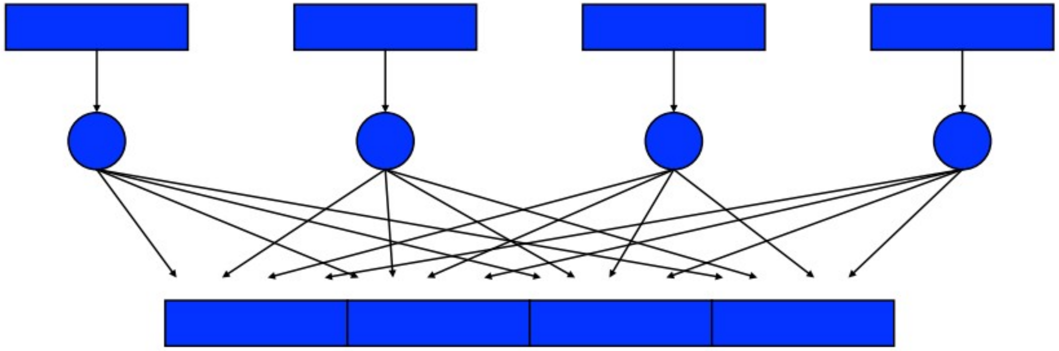
Parallel I/O Styles

- **Independent Parallel I/O**
  - Parallelism but lots of small files to handle
  - With or without MPI



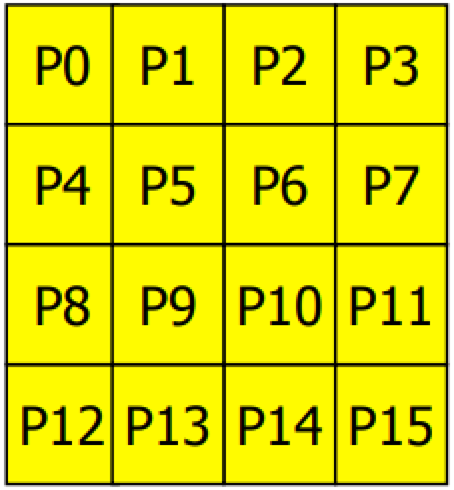
Parallel I/O Styles (contd)

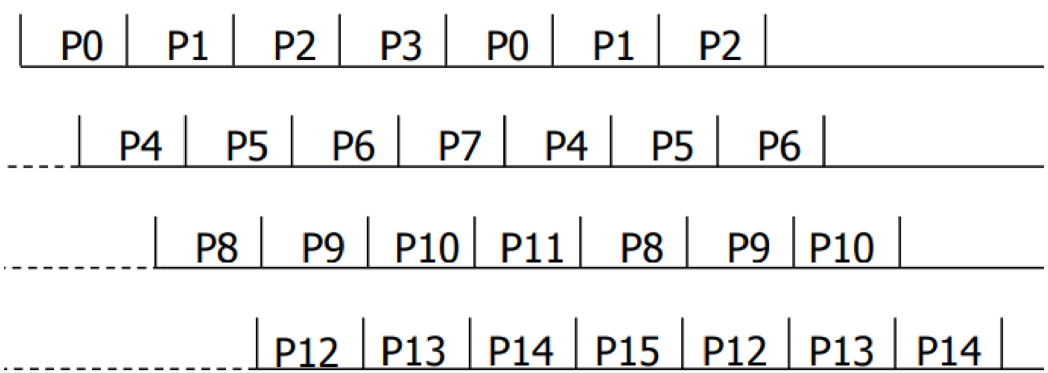
- **Cooperative Parallel I/O**
  - Parallelism
  - Only with MPI!



Parallel I/O Example

- **Distributed Array Access**
  - Large array distributed among n processes
  - Each square represents a sub-array in the memory of a single process



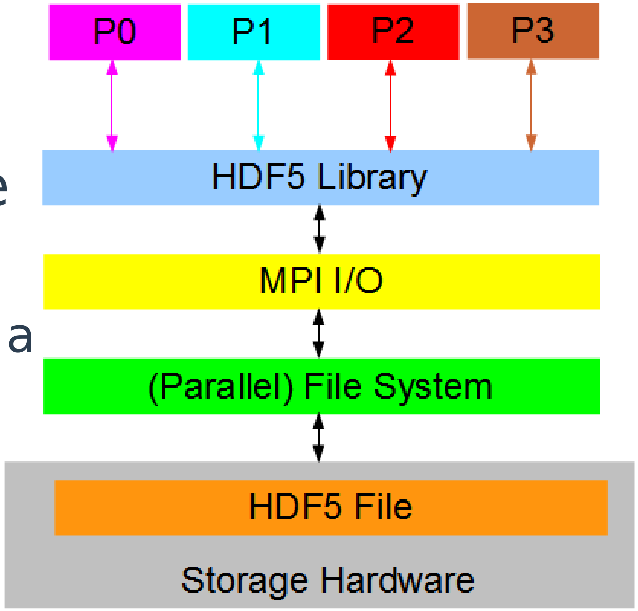


Parallel I/O in MPI

- **MPI has replacement functions for POSIX I/O**
- **Why not use POSIX?**
  - Single file (instead of one file / process)
  - Parallel performance
- **Multiple types of I/O in MPI**
  - Some are not possible without MPI
  -

HDF5

- **A high level open-source parallel I/O library**
  - Interface between the app and the parallel MPI-IO.
  - Encapsulates the MPI-IO library, optimize and add more features to build a high level parallel I/O library.
  - The users only need to apply this knowledge in their parallel program.
    - Save the development and optimization time for a parallel I/O application.
  - Is been used to manage large and complex data collections in several companies like NASA, etc.



LTTng-UST Tracepoints for Parallel I/O

- **Different I/O levels**
  - **HDF5**
    - Hierarchical Data Format functions (h5x_create,h5x_write,h5x_close, etc.)
  - **MPI**
    - **MPI-IO Functions** (MPI_File_open, MPI_File_read, MPI_File_write, MPI_File_close, etc.)
  - **File system**

Trace Compass Views

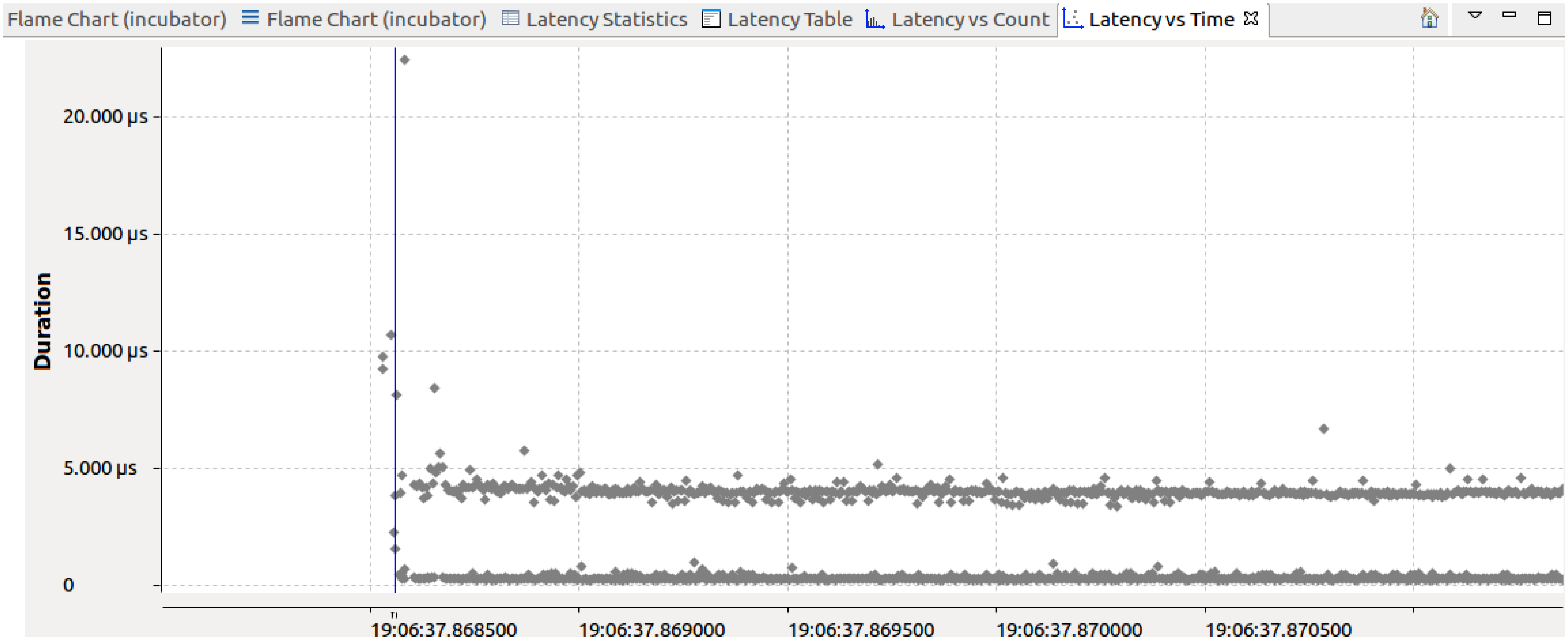
- **Profiling**
  - Amount of time spent in MPI functions,
  - Number of messages sent, the IO of each process and the whole system,
  - Different latency values, etc.
- **Multilevel Call Stack**
  - MPI--> Pthread--> Kernel
  - HDF5 --> MPI-IO → POSIX --> Kernel

Trace Compass Views (contd)

Properties | Tasks | State System Explorer | Flame Chart (incubator) | Flame Chart (incubator) | Latency Statistics | Latency Table

Start Time	End Time	Duration	Name	Content
19:06:37.659 787 533	19:06:37.868 526 185	208,738,652	mpi:mpi_init	threadID=22822
19:06:37.895 679 755	19:06:37.929 753 468	34,073,713	mpi:mpi_finalize	threadID=22823
19:06:37.895 690 377	19:06:37.928 175 500	32,485,123	mpi:mpi_finalize	threadID=22822
19:06:37.881 234 161	19:06:37.895 181 373	13,947,212	mpi:mpi_recv	threadID=22823
19:06:37.859 748 690	19:06:37.868 526 097	8,777,407	mpi:mpi_init	threadID=22823
19:06:37.871 962 773	19:06:37.879 105 641	7,142,868	mpi:mpi_recv	threadID=22823
19:06:37.879 475 363	19:06:37.880 407 771	932,408	mpi:mpi_recv	threadID=22823
19:06:37.868 582 007	19:06:37.868 604 736	22,729	mpi:mpi_recv	threadID=22823
19:06:37.881 081 572	19:06:37.881 101 757	20,185	mpi:mpi_recv	threadID=22823
19:06:37.895 184 923	19:06:37.895 200 562	15,639	mpi:mpi_recv	threadID=22823
19:06:37.879 106 690	19:06:37.879 118 195	11,505	mpi:mpi_recv	threadID=22823
19:06:37.880 482 090	19:06:37.880 493 170	11,080	mpi:mpi_recv	threadID=22823
19:06:37.868 548 444	19:06:37.868 559 284	10,840	mpi:mpi_recv	threadID=22823
19:06:37.871 731 183	19:06:37.871 741 800	10,617	mpi:mpi_recv	threadID=22823
19:06:37.895 177 570	19:06:37.895 188 040	10,470	mpi:mpi_recv	threadID=22822
19:06:37.868 529 249	19:06:37.868 539 169	9,920	mpi:mpi_barrier	threadID=22822
19:06:37.880 409 030	19:06:37.880 418 818	9,788	mpi:mpi_recv	threadID=22823
19:06:37.868 529 275	19:06:37.868 538 648	9,373	mpi:mpi_barrier	threadID=22823
19:06:37.871 698 536	19:06:37.871 707 673	9,137	mpi:mpi_recv	threadID=22823

Trace Compass Views (contd)

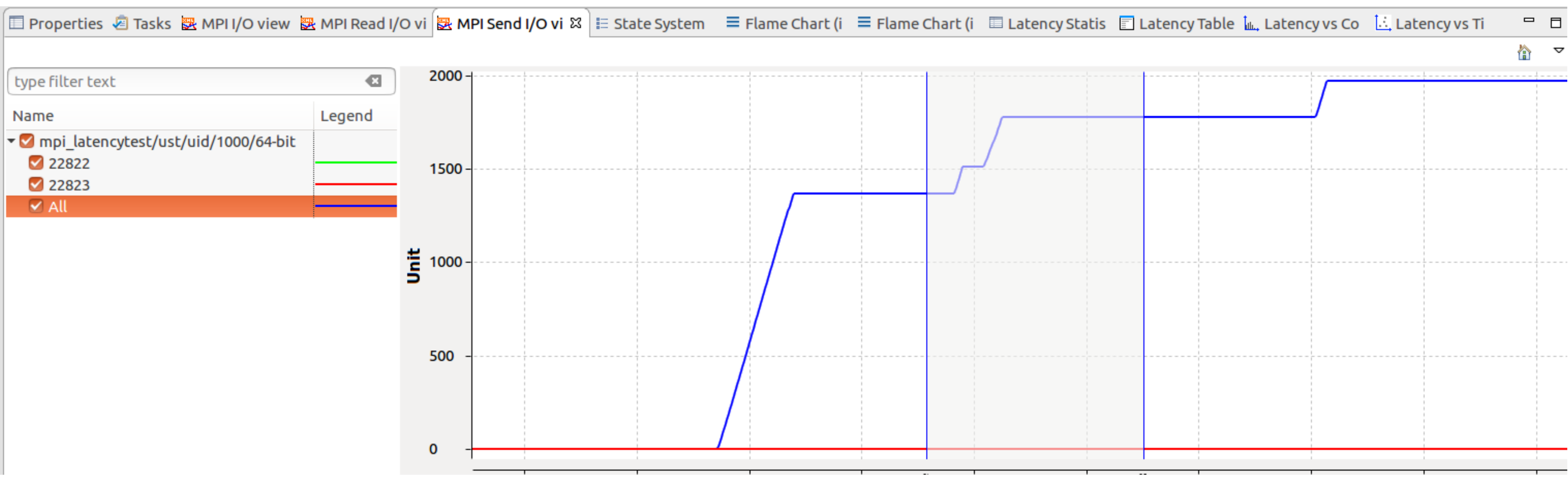


Trace Compass Views (contd)

Properties | Tasks | State System Explorer | Flame Chart (incubator) | Flame Chart (incubator) | Latency Statistics

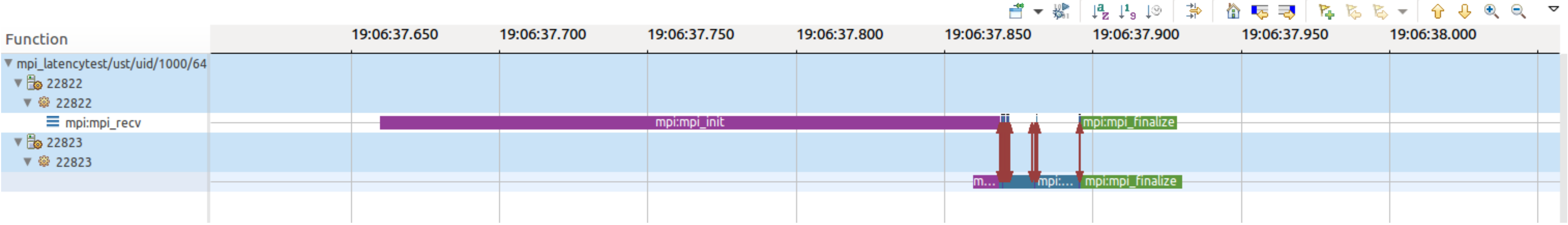
Level	Minimu	Maximum	Average	Standard Deviation	Count	Total
Total	244 ns	208.739 ms	77.791 µs	3.392 ms	4006	311.631 ms
mpi:mpi_send	244 ns	4.883 µs	310 ns	224 ns	2000	620.009 µs
mpi:mpi_recv	378 ns	13.947 ms	13.458 µs	350.853 µs	2000	26.917 ms
mpi:mpi_barrier	9.373 µs	9.920 µs	9.646 µs	---	2	19.293 µs
mpi:mpi_init	8.777 ms	208.739 ms	108.758 ms	---	2	217.516 ms
mpi:mpi_finalize	32.485 ms	34.074 ms	33.279 ms	---	2	66.559 ms

Trace Compass Views (contd)

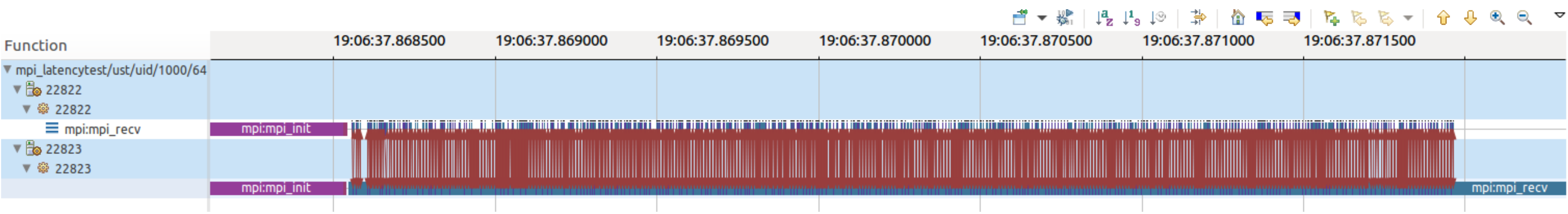


Trace Compass Views (contd)

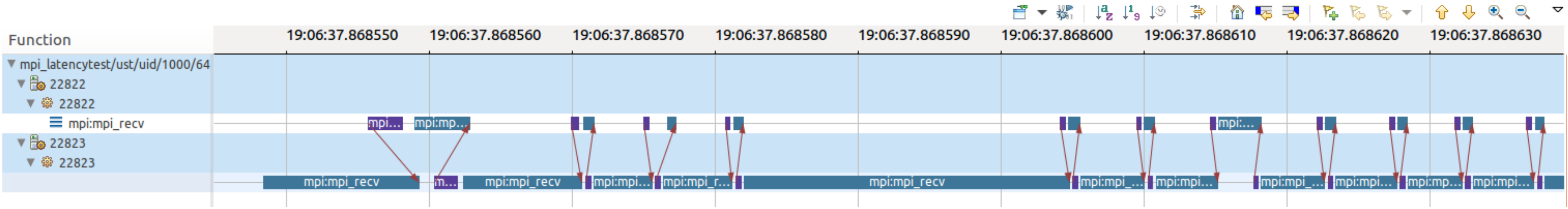
Case 1: ping-pong



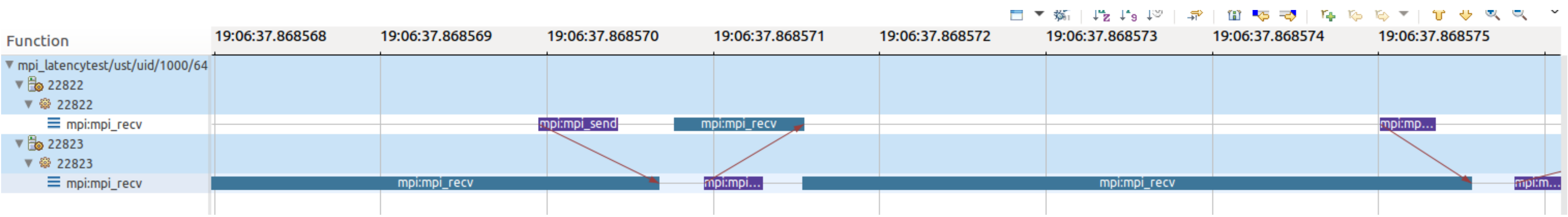
Trace Compass Views (contd)



Trace Compass Views (contd)

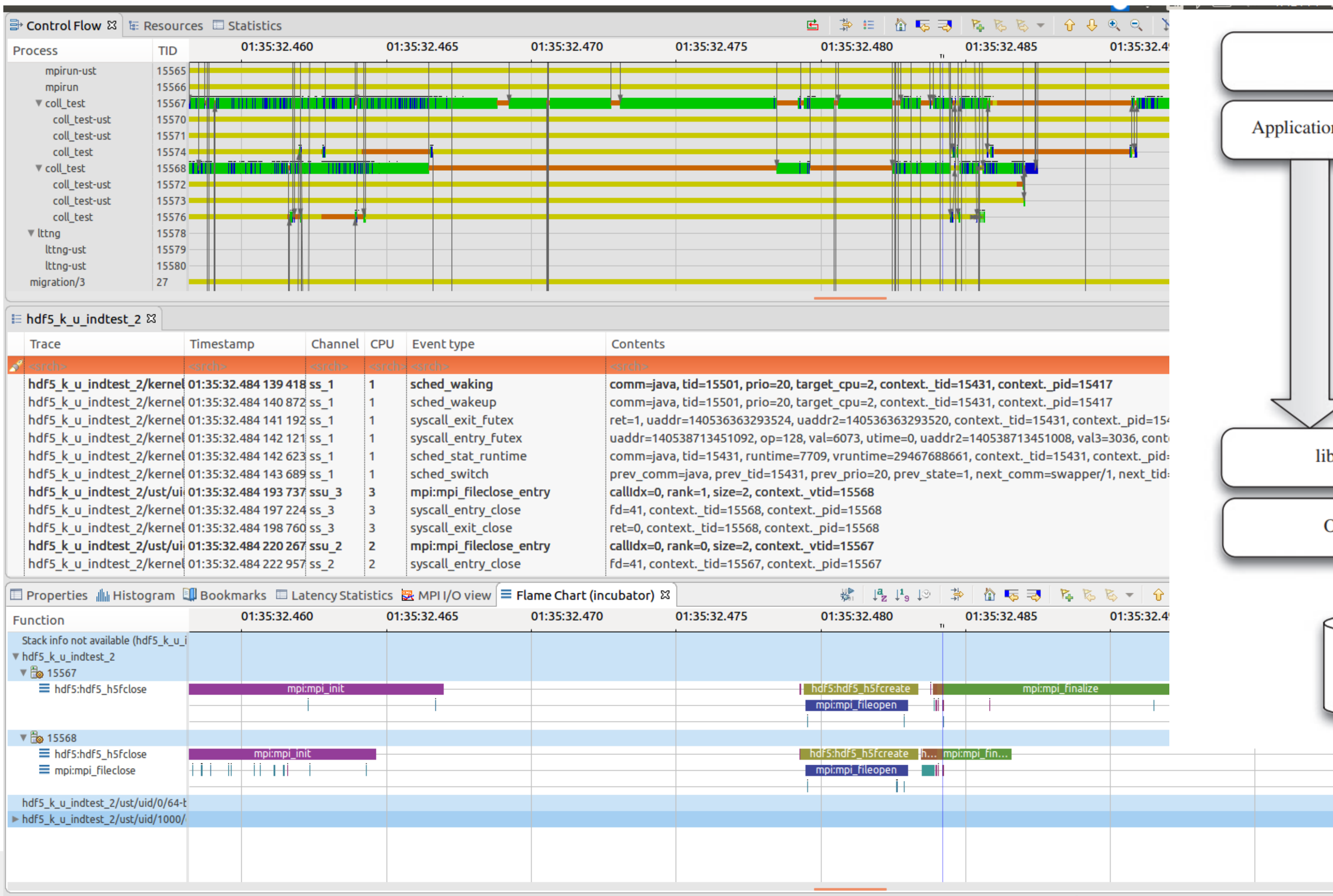


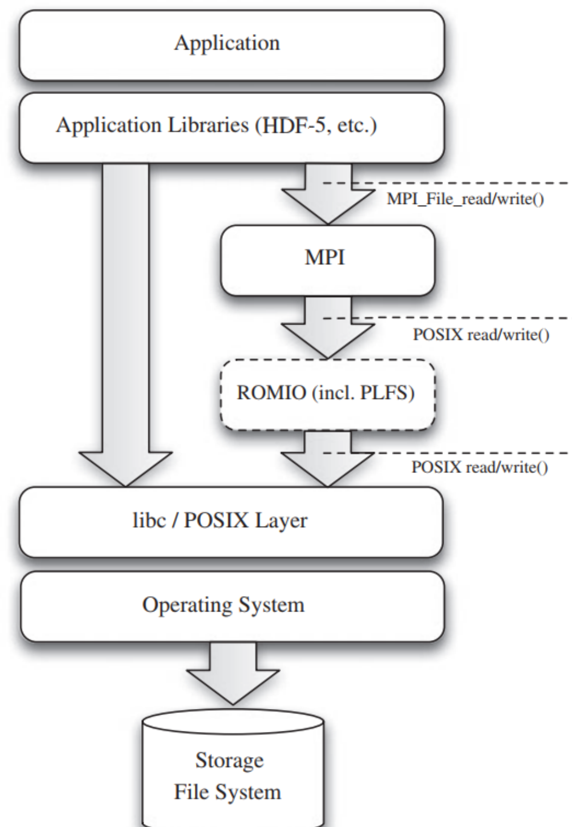
Trace Compass Views (contd)



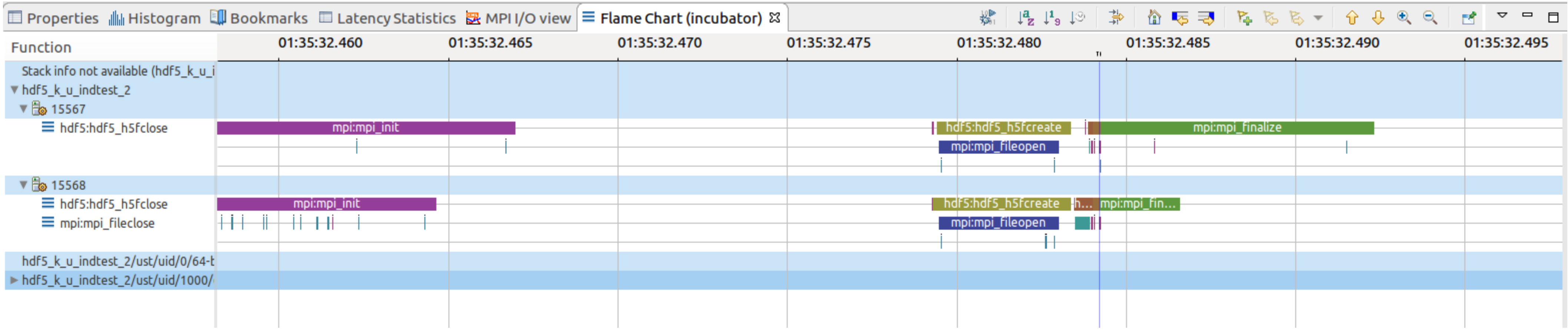
Trace Compass Views (contd)

Case 2: Multi-level Parallel I/O

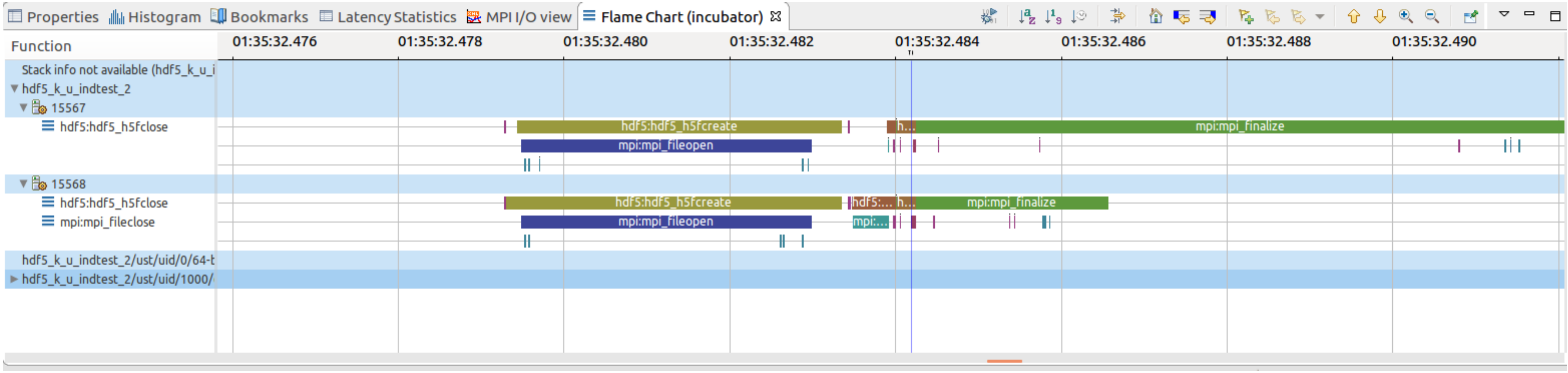




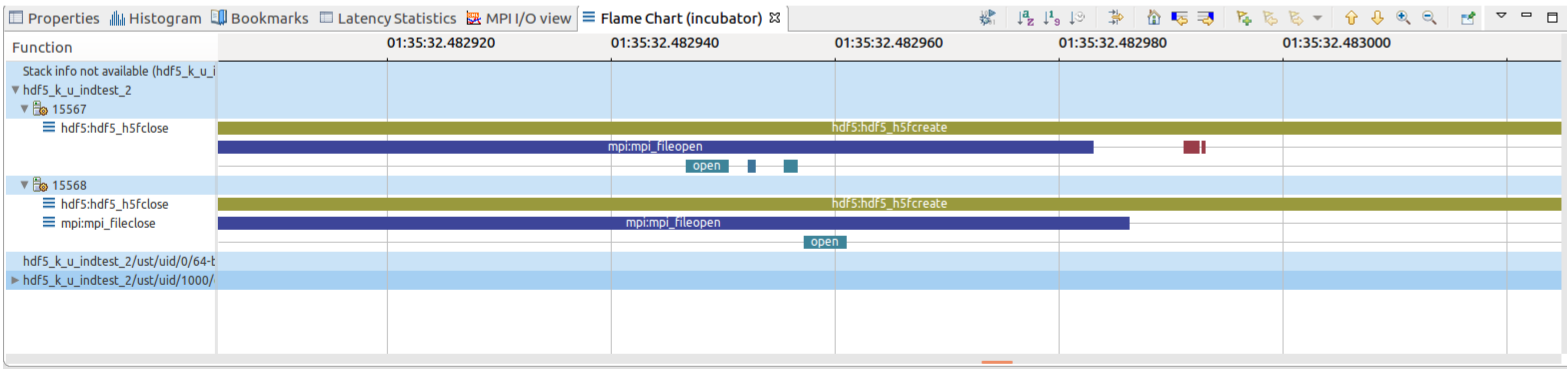
Trace Compass Views (contd)



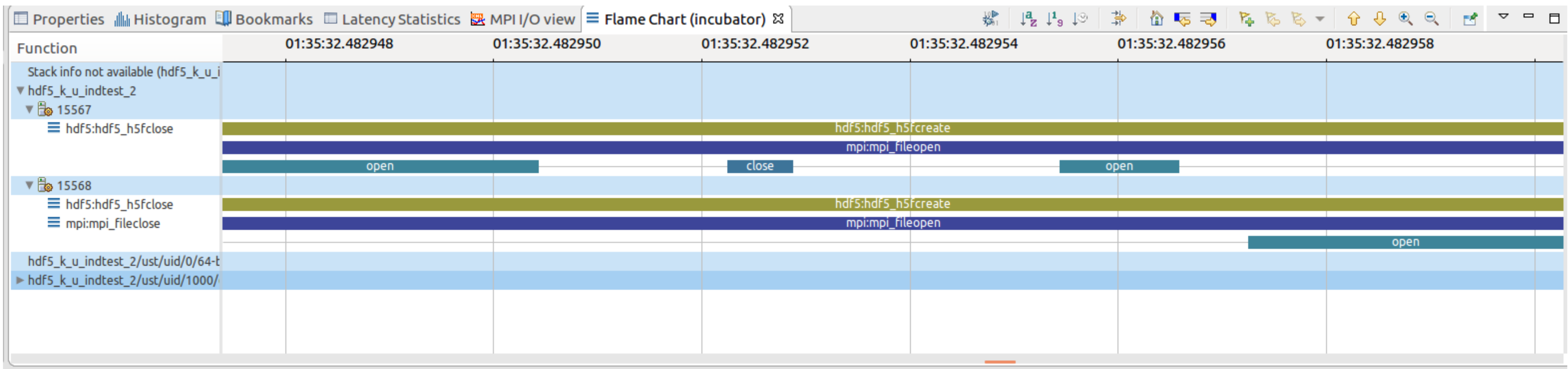
Trace Compass Views (contd)



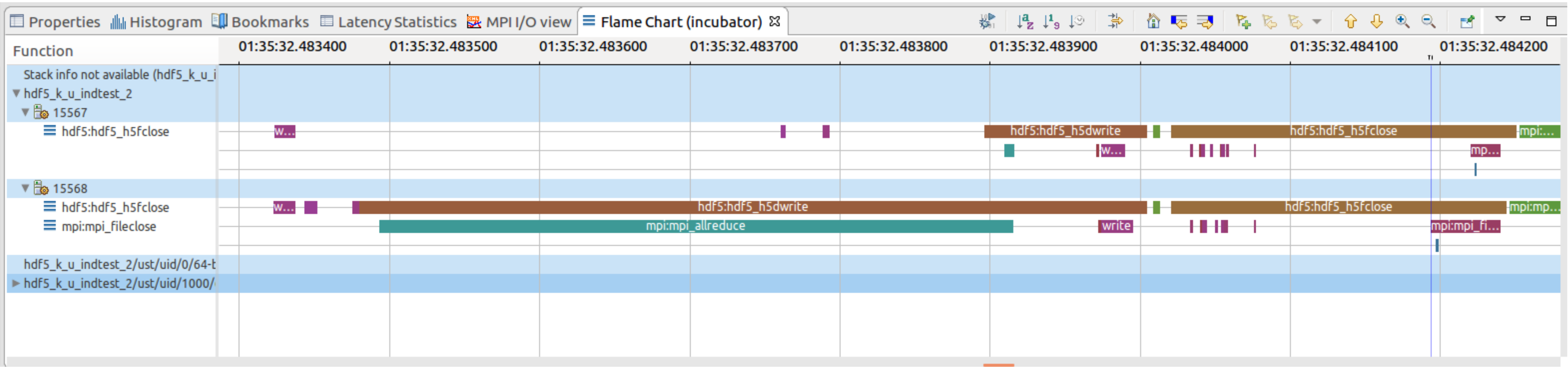
Trace Compass Views (contd)



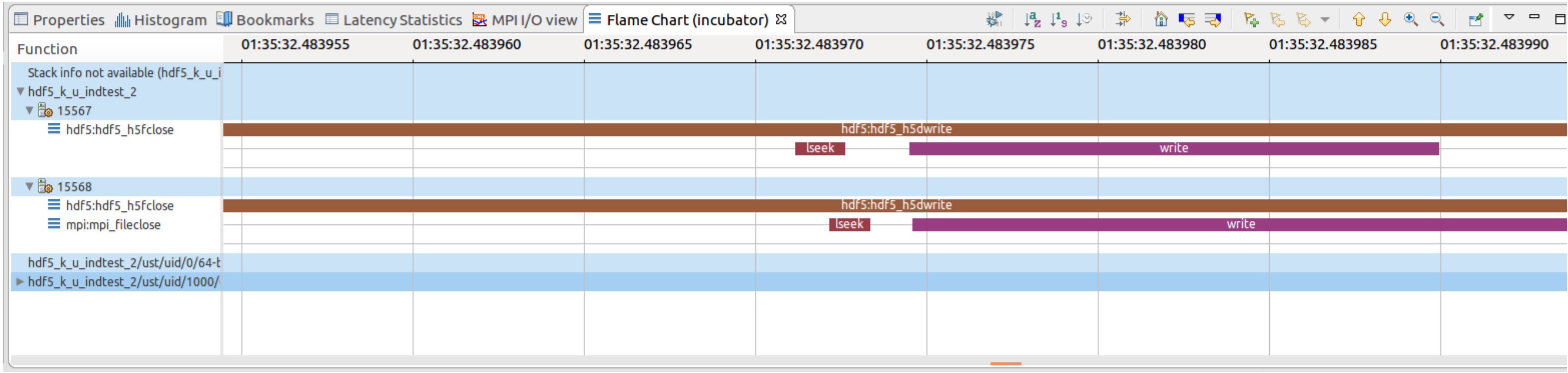
Trace Compass Views (contd)



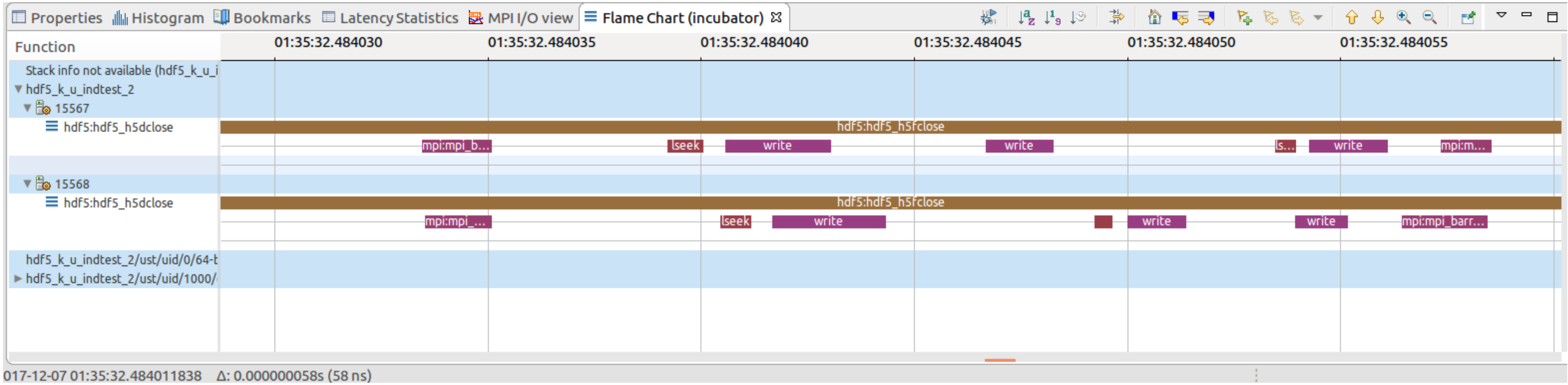
Trace Compass Views (contd)



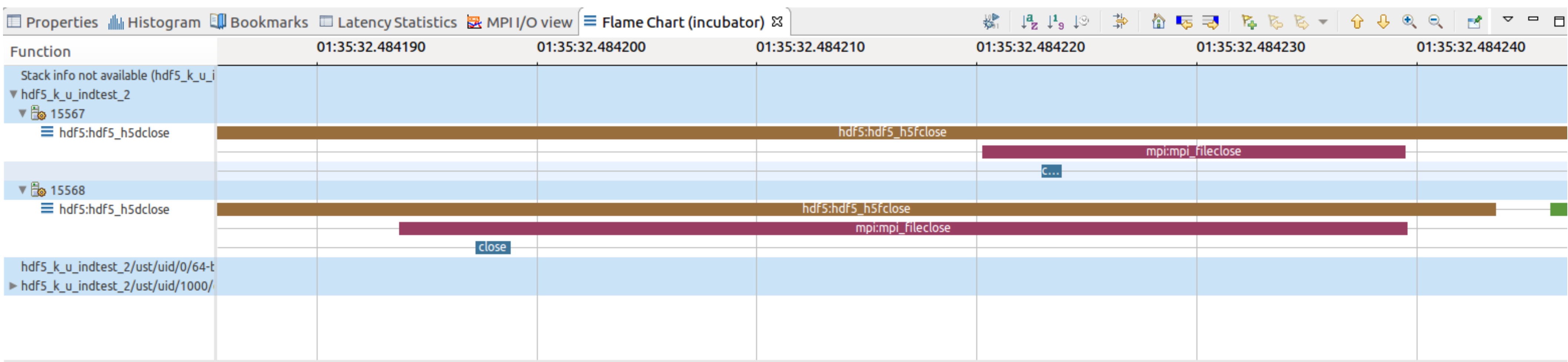
Trace Compass Views (contd)



Trace Compass Views (contd)

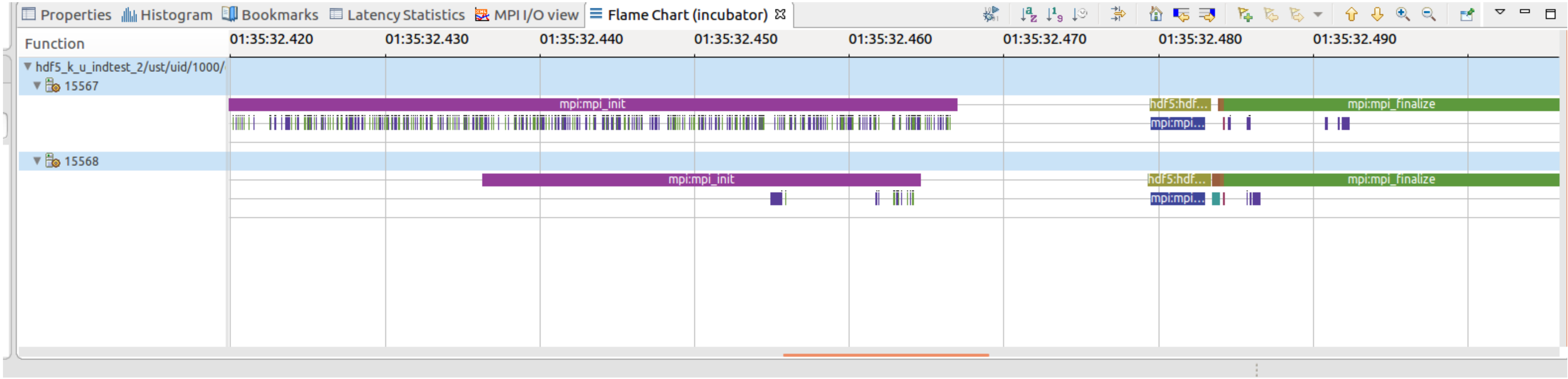


Trace Compass Views (contd)

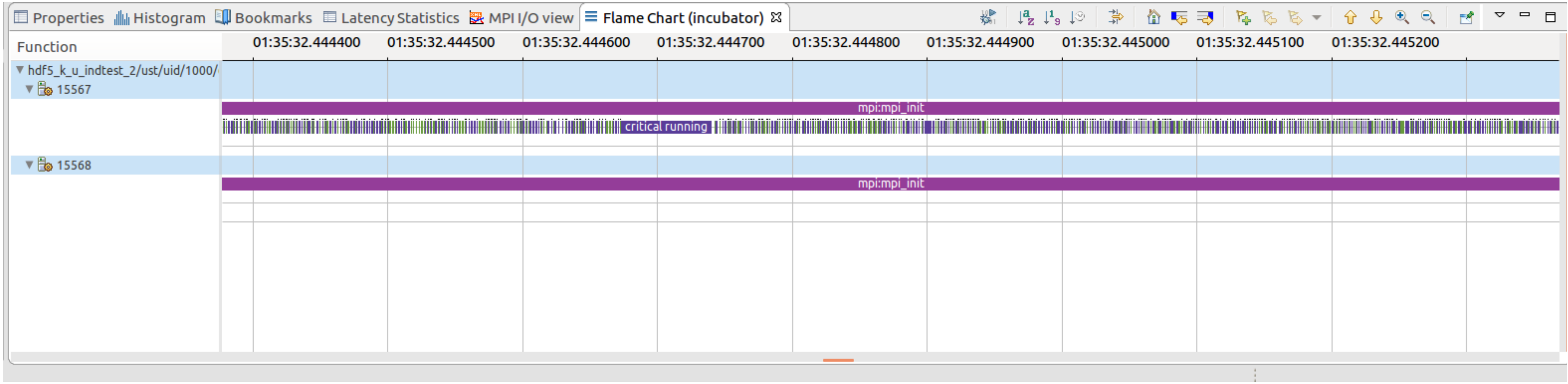


Trace Compass Views (contd)

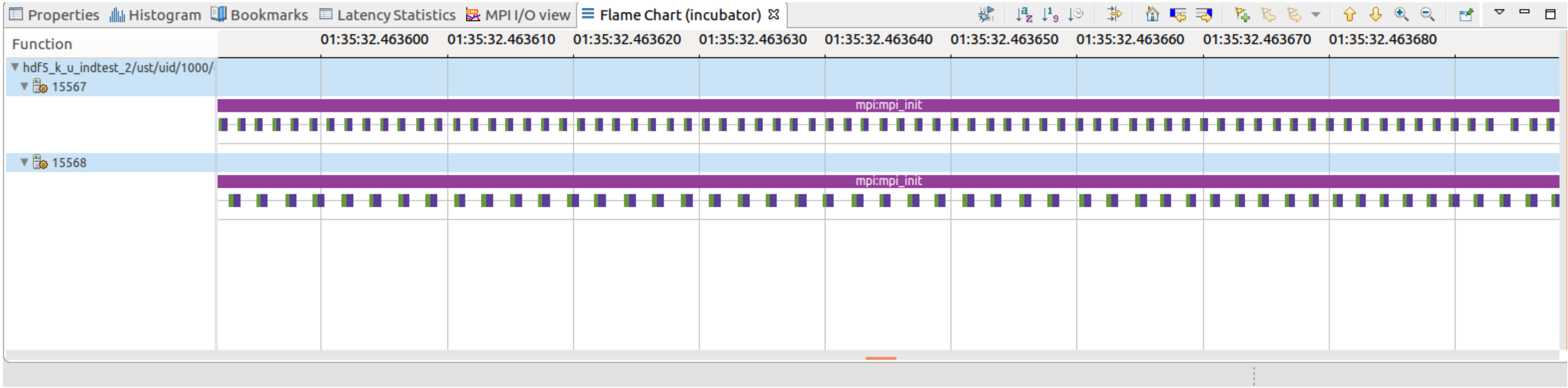
Case 3: MPI & Pthread



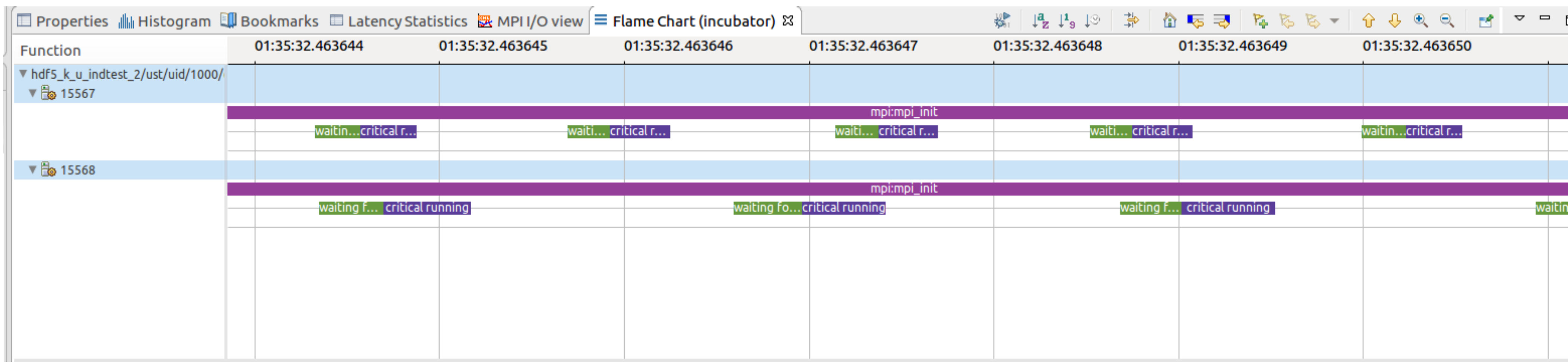
Trace Compass Views (contd)



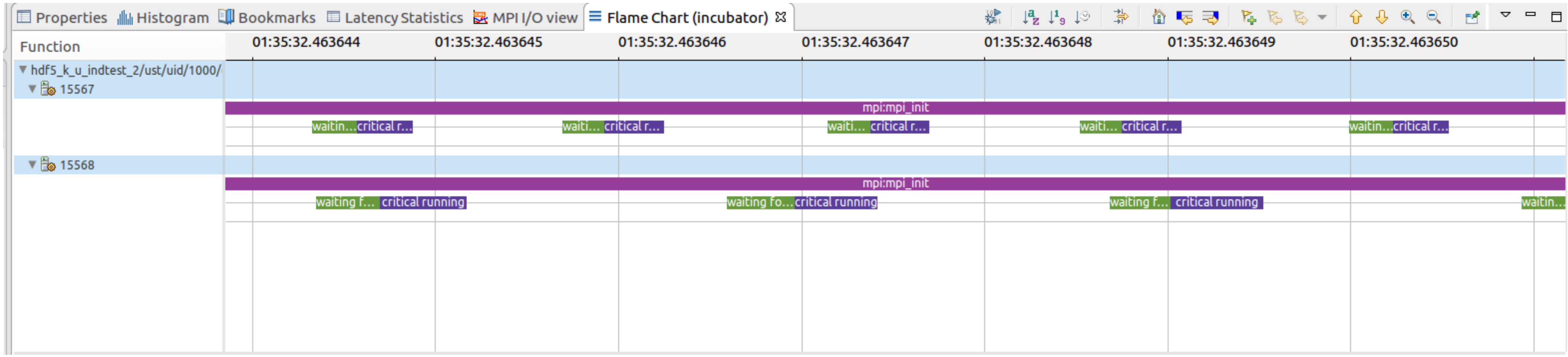
Trace Compass Views (contd)



Trace Compass Views (contd)



Trace Compass Views (contd)



Thank you

Any Question?

github.com/naser

n.ezzati@polymtl.ca