Visualization of MPI(-IO) Datatypes

Julian M. Kunkel, Thomas Ludwig

julian.martin.kunkel@informatik.uni-hamburg.de

Scientific Computing
Department of Informatics
University of Hamburg

30.08.2011

informatik
die zukunft

Scope of this presentation

- Visualization of MPI datatypes and non-contiguous I/O in a post-mortem trace analyzer.
- General scheme to illustrate datatypes and I/O access.

informatik die zukunft

informatik die zukunft

Derived datatypes in MPI

Motivation

- Allow to communicate non-contiguous and mixed datatypes
- File-views in MPI-2 permit non-contiguous I/O

Construction of datatypes

- Datatypes are created bottom up
- Use a constructor and defined datatypes $\Rightarrow$ new datatype
- Constructors: contiguous, structure, vector, indexed ...

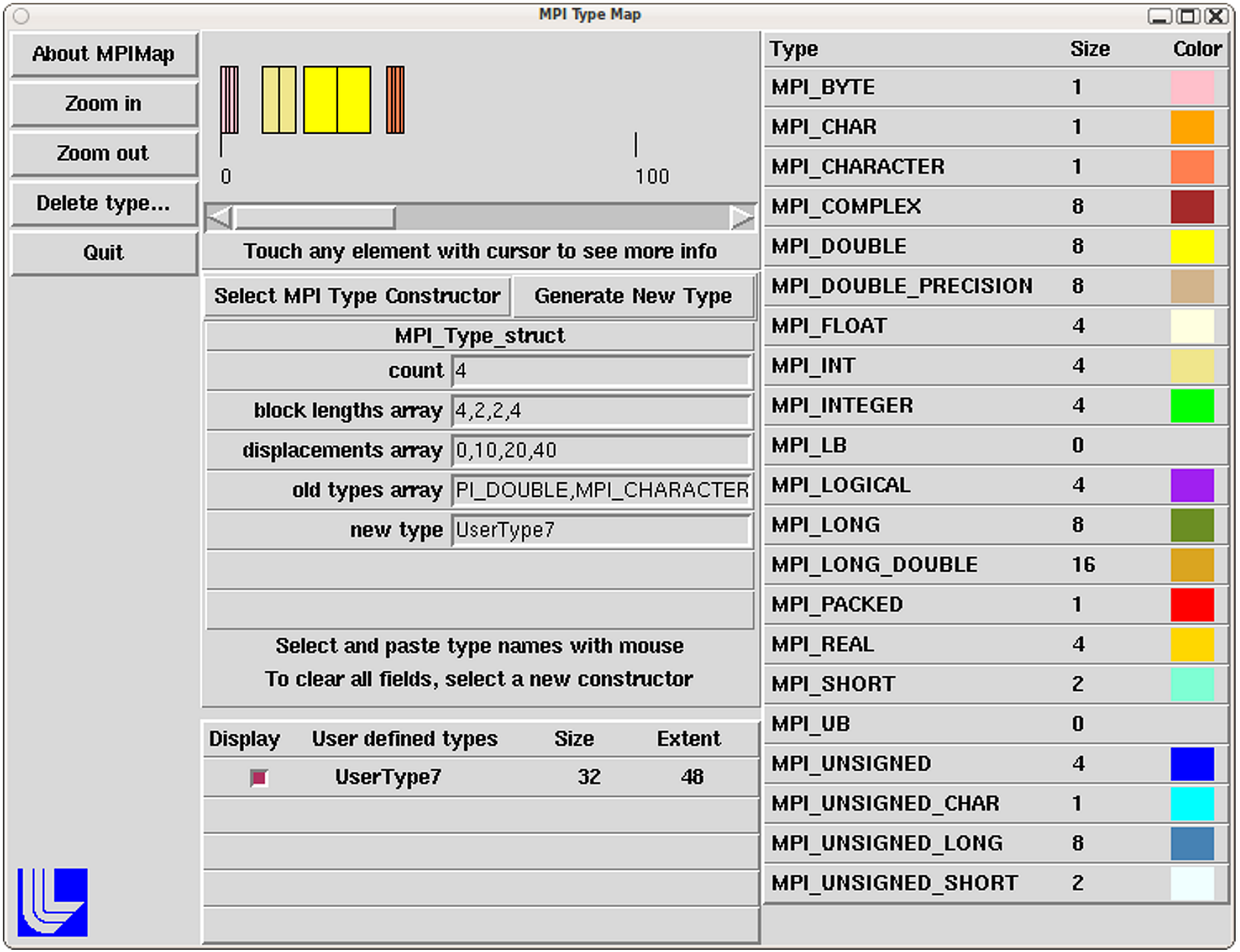
The need to analyze datatypes

- Nested datatypes may be complicated
- Debugging of datatypes
  - Check the right bytes in memory or on disk are accessed
- Reverse engineering

informatik die zukunft

MPIMAP

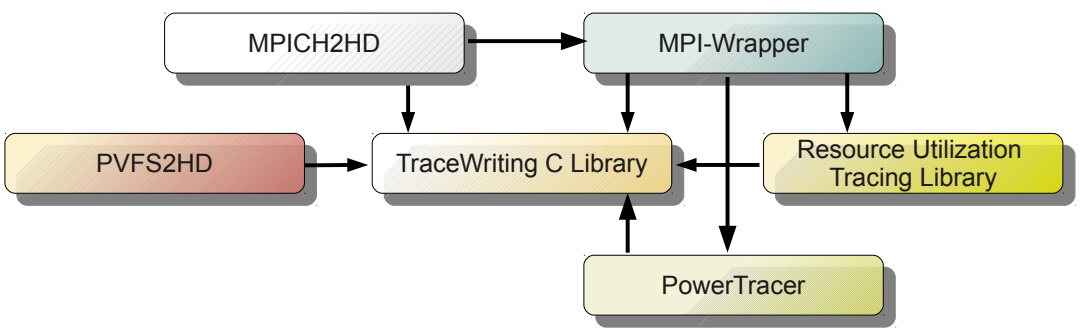
- TCL/TK tool which visualizes the creation of datatypes
- Interactively create up to 4 datatypes
- Unrolls the accessed area on the x-axis
- Runs with a single MPI process


MPI Type Map
About MPIMap
Zoom in
Zoom out
Delete type...
Quit
0
100
Touch any element with cursor to see more info
Select MPI Type Constructor
Generate New Type
MPI_Type_struct
count
4
block lengths array
4,2,2,4
displacements array
0,10,20,40
old types array
PI_DOUBLE,MPI_CHARACTER
new type
UserType7
Select and paste type names with mouse
To clear all fields, select a new constructor
Display
User defined types
Size
Extent
UserType7
32
48
Type
Size
Color
MPI_BYTE 1
MPI_CHAR 1
MPI_CHARACTER 1
MPI_COMPLEX 8
MPI_DOUBLE 8
MPI_DOUBLE_PRECISION 8
MPI_FLOAT 4
MPI_INT 4
MPI_INTEGER 4
MPI_LB 0
MPI_LOGICAL 4
MPI_LONG 8
MPI_LONG_DOUBLE 16
MPI_PACKED 1
MPI_REAL 4
MPI_SHORT 2
MPI_UB 0
MPI_UNSIGNED 4
MPI_UNSIGNED_CHAR 1
MPI_UNSIGNED_LONG 8
MPI_UNSIGNED_SHORT 2

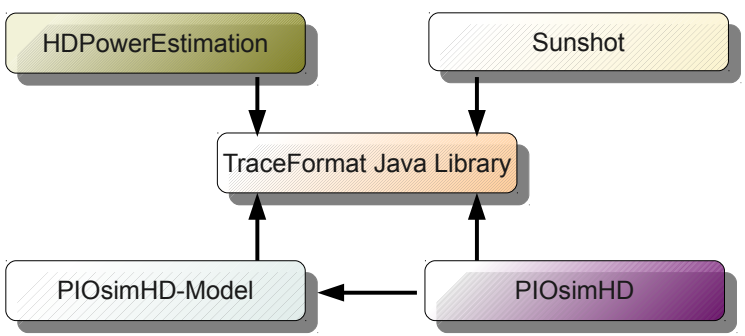
Overview

Research environment for tracing and simulation

- Analyze energy efficiency and I/O activity
- Modeling and simulation of systems executing MPI-IO apps
- Suitable for experiments of medium scale
- XML based trace format with user defined semantics



Components involved in tracing



Components related to simulation and visualization

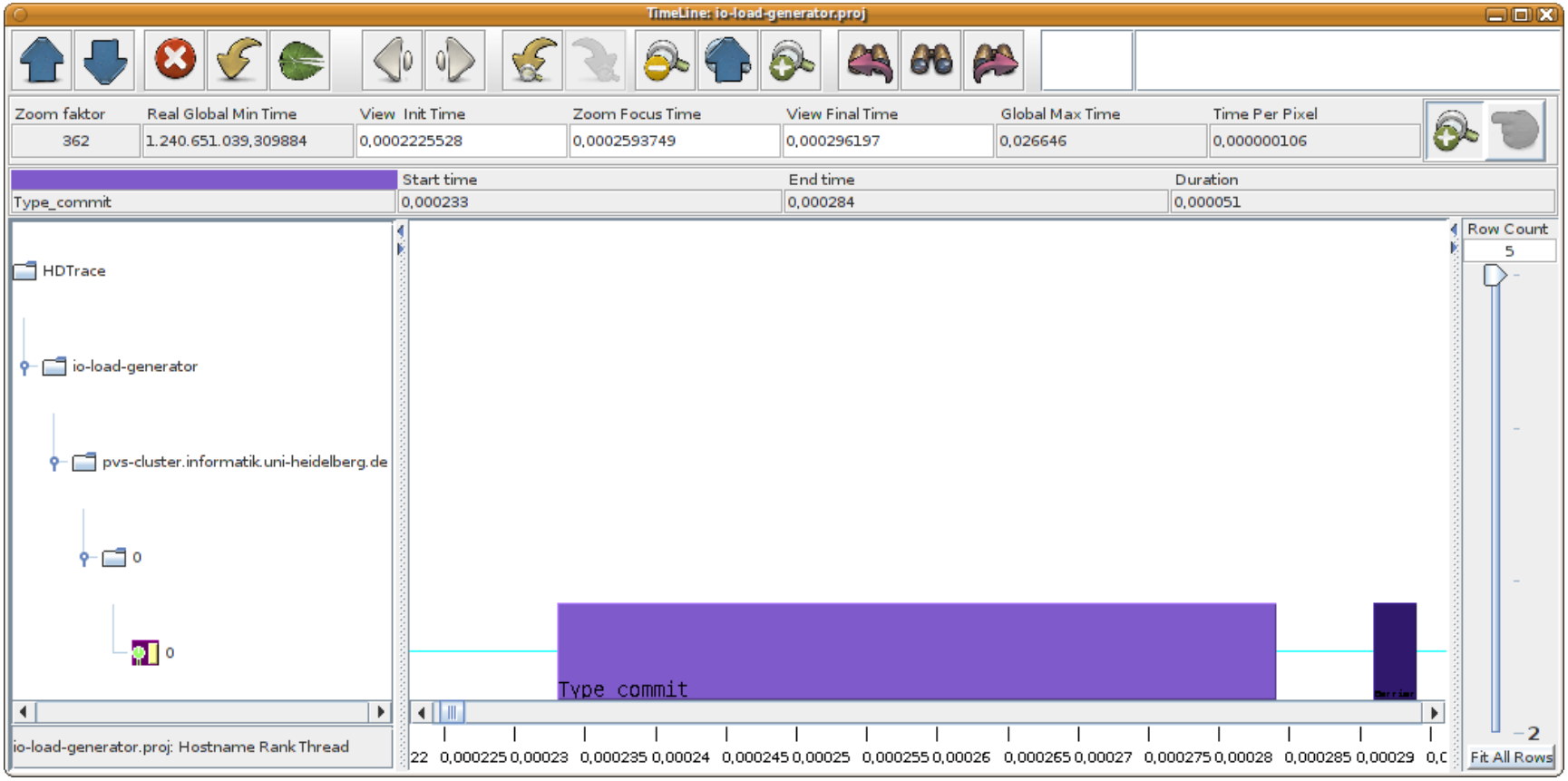
Sunshot

Overview

- Viewer for traces (and profiles)
- Written in Java, based on Jumpshot GUI

Features

- Renders numeric values (statistics) and events in timelines
- Topology maps (named) trace data to timelines
- Interaction with statistics: Histograms, derived timelines, ...
- User supplied filtering of events


TimeLine: io-load-generator.proj
Zoom faktor
362
Real Global Min Time
1.240.651.039,309884
View Init Time
0,0002225528
Zoom Focus Time
0,0002593749
View Final Time
0,000296197
Global Max Time
0,026646
Time Per Pixel
0,000000106
Start time
End time
Duration
Type_commit
0,000233
0,000284
0,000051
HDTrace
io-load-generator
pvs-cluster.informatik.uni-heidelberg.de
0
0
Type commit
Row Count
5
2
Fit All Rows
io-load-generator.proj: Hostname Rank Thread
22 0,000225 0,00023 0,000235 0,00024 0,000245 0,00025 0,000255 0,00026 0,000265 0,00027 0,000275 0,00028 0,000285 0,00029

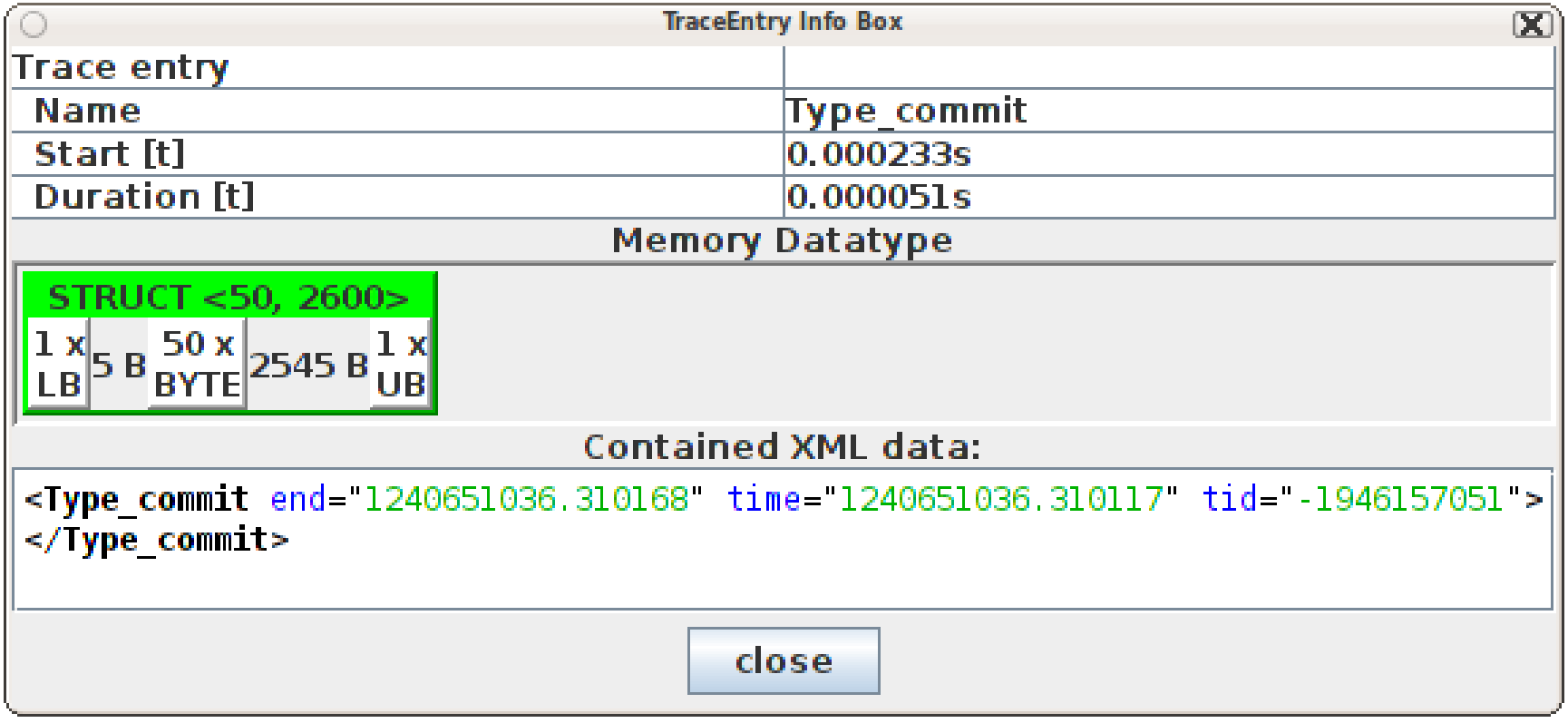
Visualization of datatypes

To start the visualization

- Select an event in the timeline
- Right click to open *TraceEntry information box*

Internal activity invoked upon startup

- Plugins scan the stored attributes of the trace entry
  - When a project is loaded plugins decide if they should be enabled
  - Depends on the topology labels (semantics)
- Memory and I/O datatypes are detected and visualized
  - Logic is implemented according to the MPI datatype constructors


TraceEntry Info Box
Trace entry
Name
Type_commit
Start [t]
0.000233s
Duration [t]
0.000051s
Memory Datatype
STRUCT <50, 2600>
1 x LB
5 B
50 x BYTE
2545 B
1 x UB
Contained XML data:
<Type_commit end="1240651036.310168" time="1240651036.310117" tid="-1946157051">
</Type_commit>
close

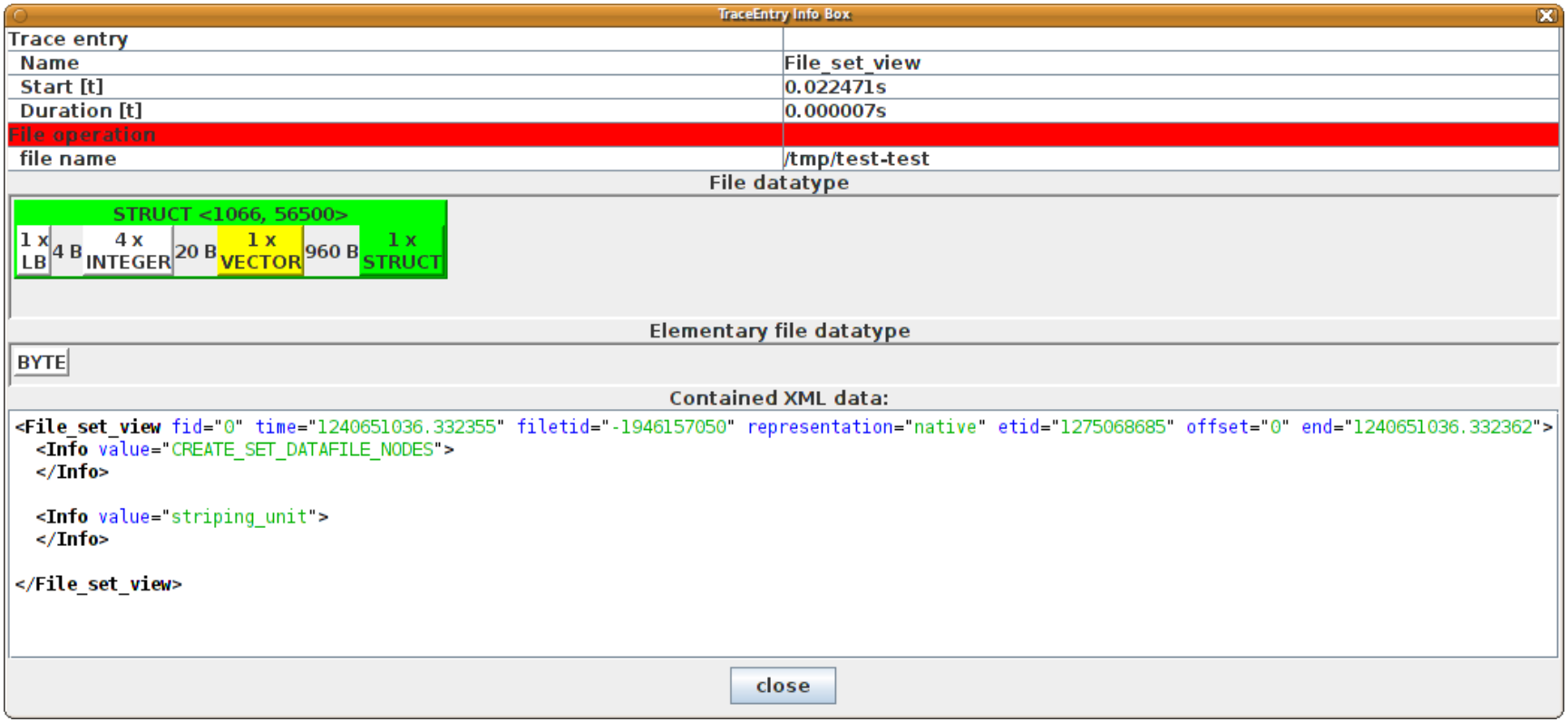
Advanced topics

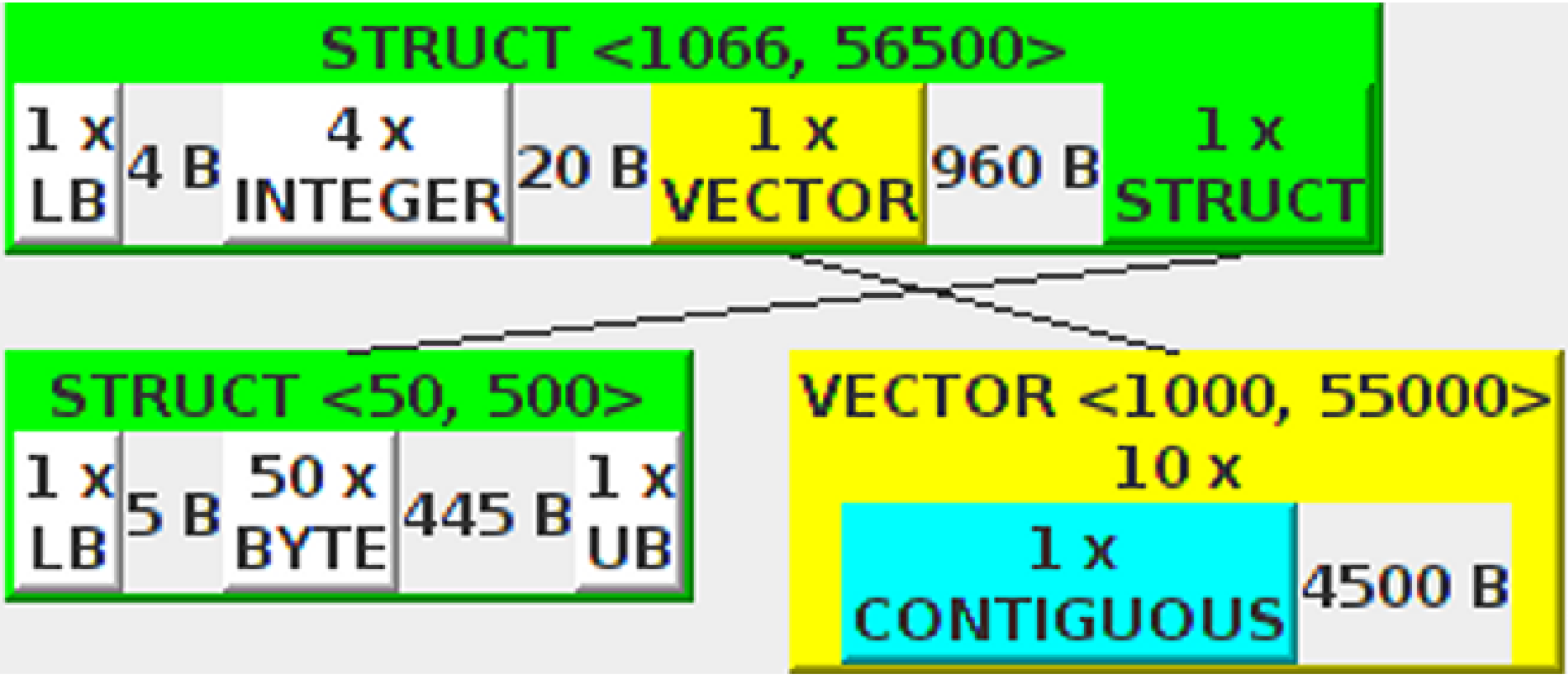
Nested datatypes

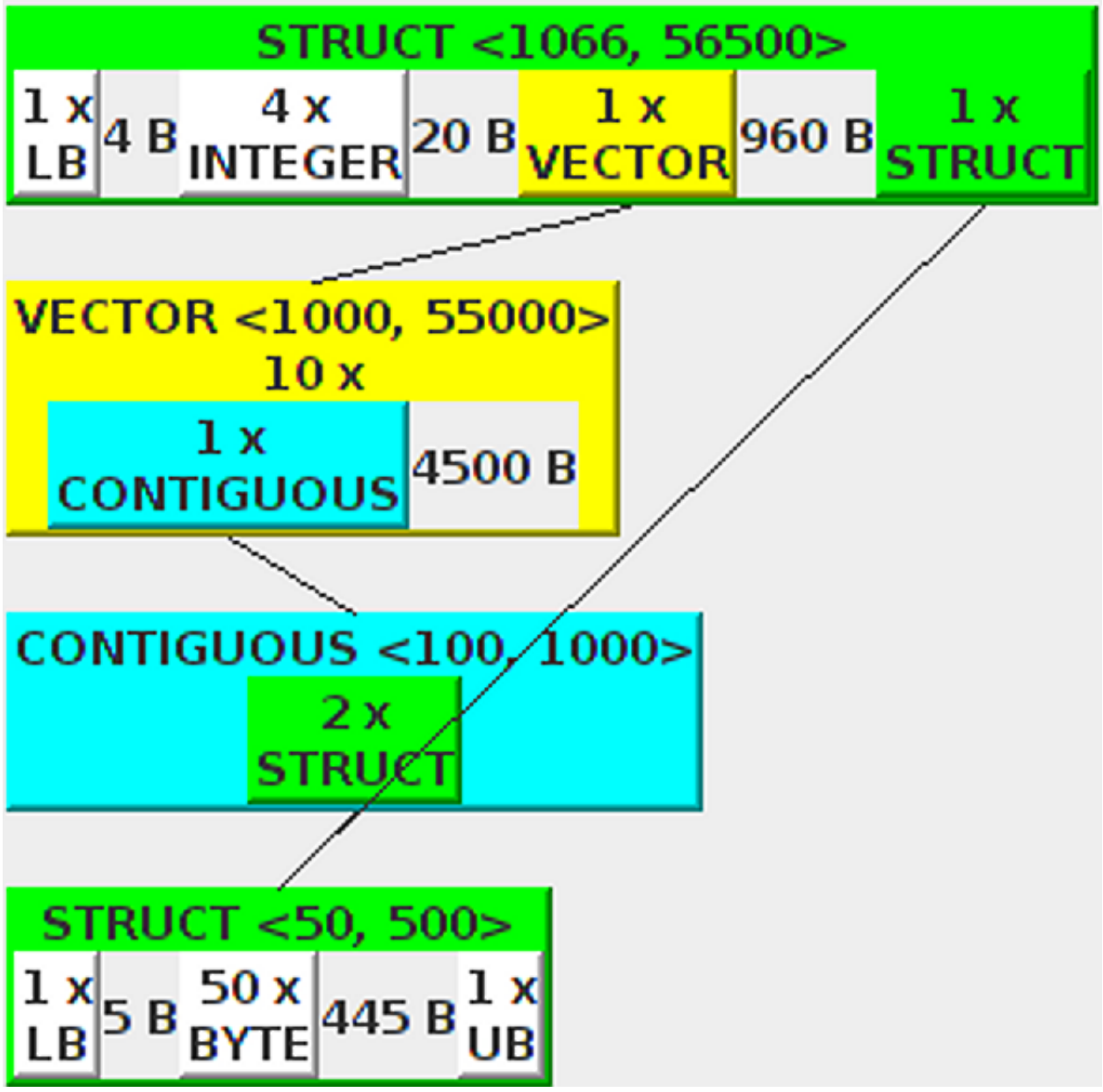
- Draw the (final) memory datatype
- Render child datatypes in a directed graph
- Expansion of the child datatypes is requested by the user
  - Perform a click on the datatype to expand it

Limitations

Ascending offsets in the datatype (limitation to MPI-IO as well).


TraceEntry Info Box
Trace entry
Name
File_set_view
Start [t]
0.022471s
Duration [t]
0.000007s
File operation
file name
/tmp/test-test
File datatype
STRUCT <1066, 56500>
1 x LB
4 B
4 x INTEGER
20 B
1 x VECTOR
960 B
1 x STRUCT
Elementary file datatype
BYTE
Contained XML data:
<File_set_view fid="0" time="1240651036.332355" filetid="-1946157050" representation="native" etid="1275068685" offset="0" end="1240651036.332362">
<Info value="CREATE_SET_DATAFILE_NODES">
</Info>
<Info value="striping_unit">
</Info>
</File_set_view>
close


STRUCT <1066, 56500>
1 x LB
4 B
4 x INTEGER
20 B
1 x VECTOR
960 B
1 x STRUCT
STRUCT <50, 500>
1 x LB
5 B
50 x BYTE
445 B
1 x UB
VECTOR <1000, 55000>
10 x
1 x CONTIGUOUS
4500 B


STRUCT <1066, 56500>
1 x LB
4 B
4 x INTEGER
20 B
1 x VECTOR
960 B
1 x STRUCT
VECTOR <1000, 55000>
10 x
1 x CONTIGUOUS
4500 B
CONTIGUOUS <100, 1000>
2 x STRUCT
STRUCT <50, 500>
1 x LB
5 B
50 x BYTE
445 B
1 x UB

Non-contiguous I/O

Visualization

- Filetype is drawn similar to memory datatypes
- Unrolling of datatype to visualize accessed regions

Datatype unrolling

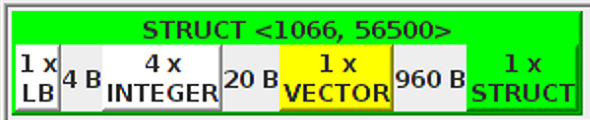
- Non-contiguous I/O might hit file datatype partially
  - Write 20 bytes, but file datatype addresses 100 byte
- Three regions: partially accessed datatype, fully repeated datatype, partially accessed datatype

TraceEntry Info Box

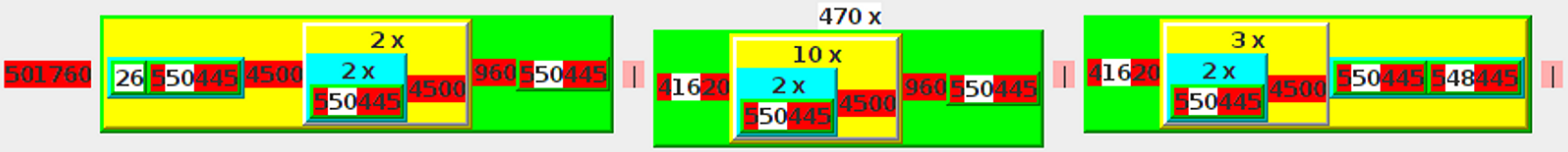
Trace entry	
Name	File_write_at
Start [t]	0.022478s
Duration [t]	0.000624s
File operation	
file name	/tmp/test-test
size	501760
physical offset (after view)	501760
etype size, extent	1, 1

Memory Datatype

File datatype



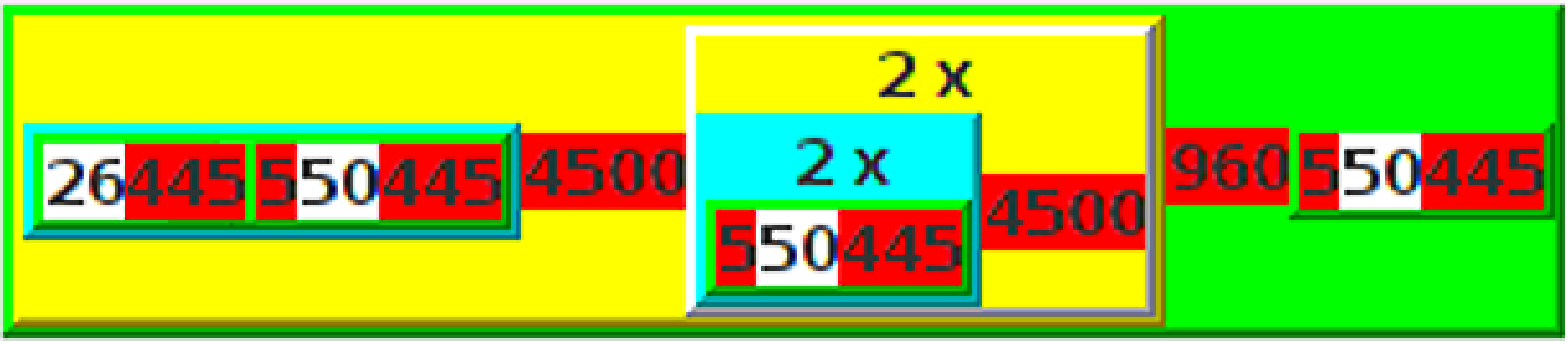
Accessed bytes:

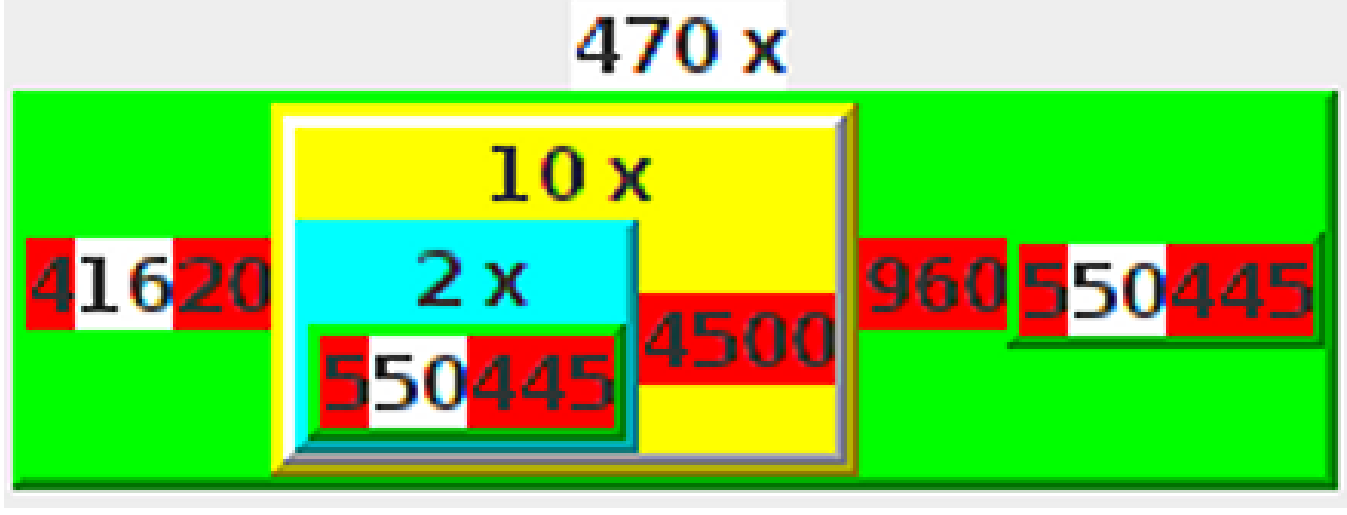


Contained XML data:

```
<File_write_at fid="0" time="1240651036.332362" count="501760" tid="1275068685" offset="501760" end="1240651036.332986" size="501760">
</File_write_at>
```

close


2 x
2 x
26445
550445
4500
550445
4500
960
550445


470 x
10 x
2 x
41620
550445
4500
960
550445

. . .

Conclusions

- HDTrace is a tracing and simulation environment
- Visualization of memory datatypes and I/O regions is important
- Sunshot enables to analyze datatypes in trace files
- Nested datatypes are visualized in a DAG
- Views are unrolled to indicate observed I/O access patterns

Future work

- Show collective (non-)contiguous I/O patterns in one window.
- Extract code into a small tool
  - Ease evaluation of datatypes (similar as MPIMAP)
  - Allow programming of datatypes (and I/O) in Java
  - Generate C code to create visualized datatypes