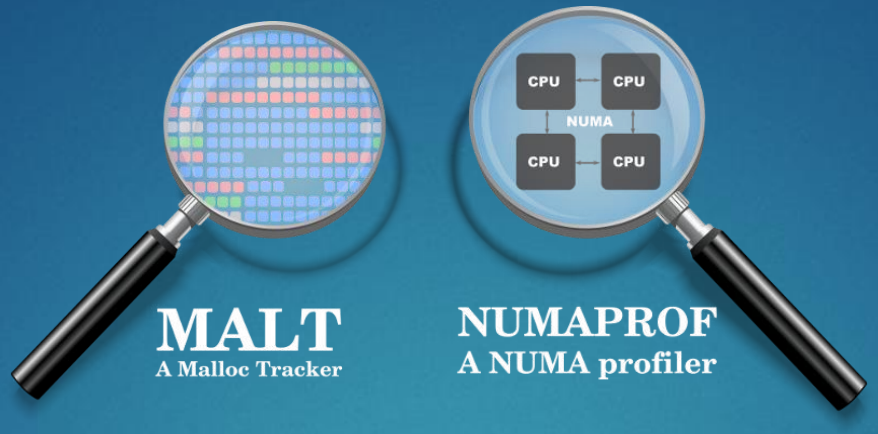
Sébastien Valat - INRIA / LJK
15/02/2024



Allocations mémoires, pourquoi et comment profiler

SÉBASTIEN VALAT – JEUDIS DE GRAY SCOTT – FÉVRIER 2024

Inria

LABORATOIRE JEAN KUNTZMANN
MATHÉMATIQUES APPLIQUÉES · INFORMATIQUE

Origin of the tools

- **PhD**. On **memory management** for **HPC** (at CEA / UVSQ)
- **MALT** : post-doc at Versailles :

- **NUMAPROF** : side project post-doc work at :

- **URL** :

Get both on :
https://memtt.github.io/

Motivation

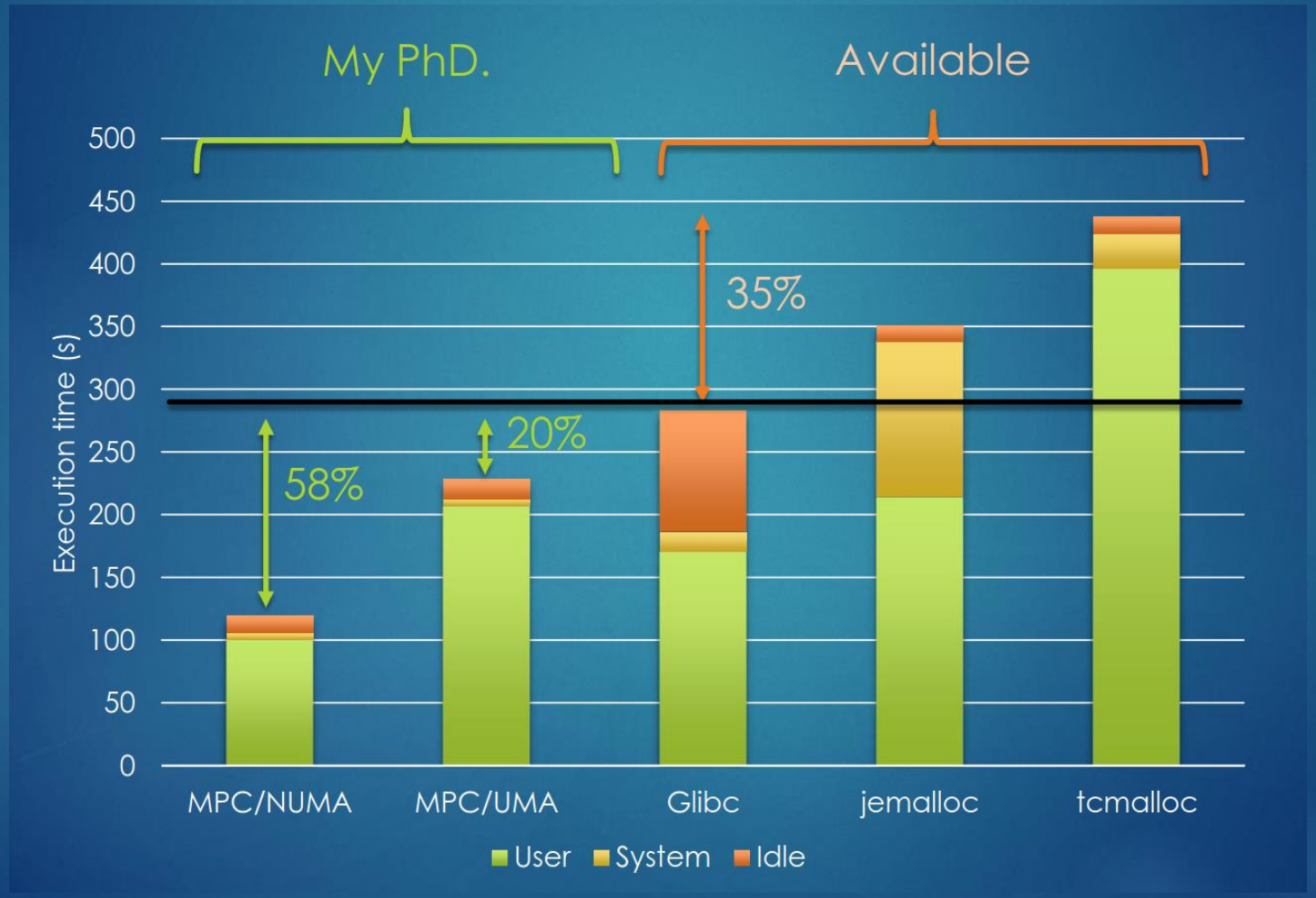
- Lot of **issues** today :
  - **Huge** memory **space** to **manage** (~TB of memory)
  - **Lot more** distinct **allocations** (e.g. 75M in 5 minutes)
  - **Multi-threaded** : 256 threads
  - **Hidden** into large (**huge**) C/C++/Fortran **codes** (**~1M** lines)

  - Access:
    - **NUMA** (Non Uniform Memory Access)
    - **Memory wall !**

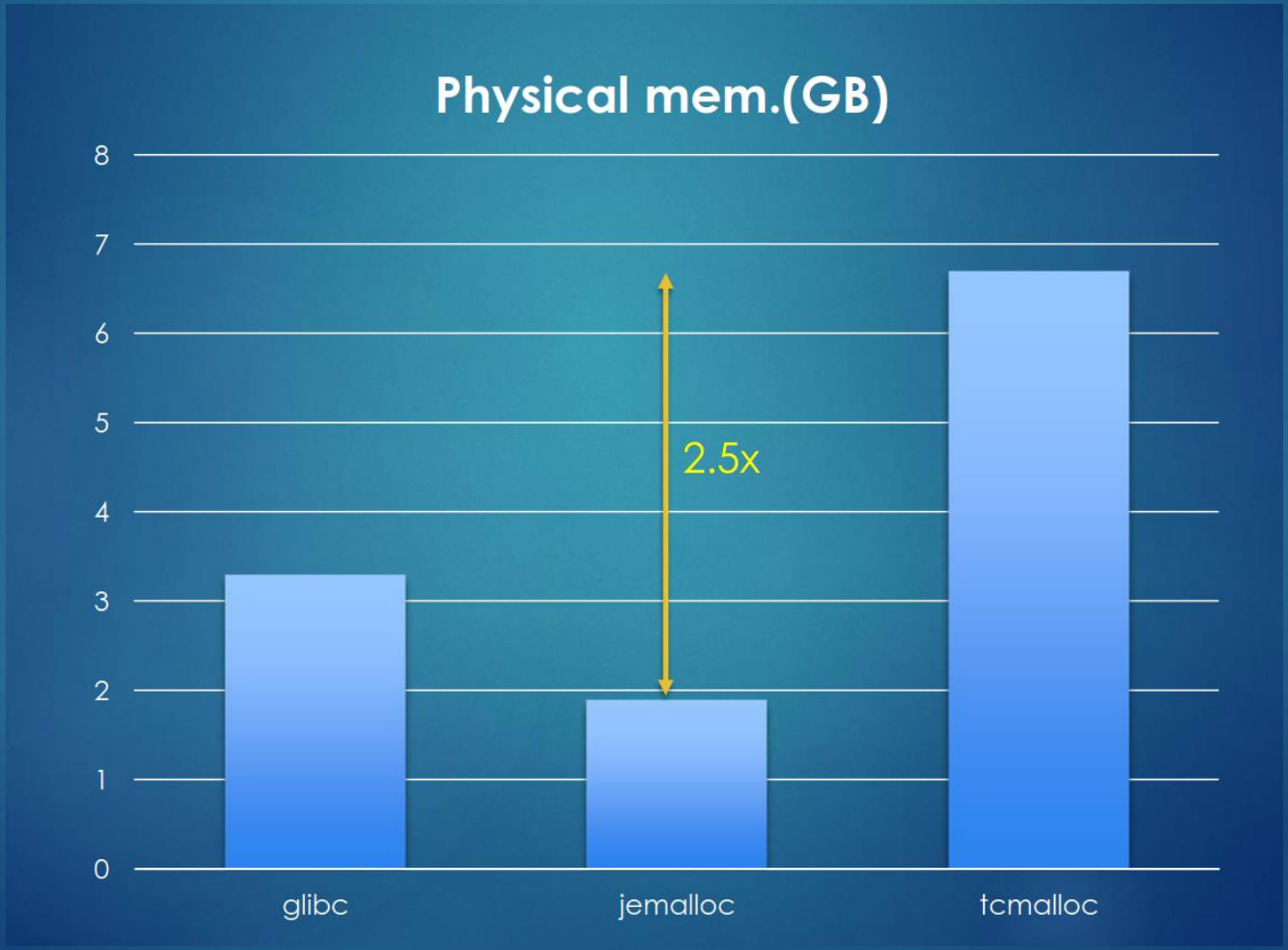
Key today

You need to
well understand memory
Behavior of you (HPC)
Application !

Eg: **>1M lines** C++ simulation. On **128** cores / **16** NUMA CPUs



Same about **memory consumption** on 12 cores



Sébastien Valat - INRIA / LJK
15/02/2024

MALT

MALLOC TRACKER

Goal

- We have **profiling tool** for **timing** (eg. Valgrind or vtune)
- But for **memory usage** ?
- Memory can be an issue :
  - Failed to run (or swap) due to **lack of memory resource**.
  - **Performance impact** of memory management functions
- Three main questions :
  - How to reduce **memory footprint** ?
  - How to improve overhead of **memory management** ?
  - How to improve **memory usage** ?

Some issue examples

- We want to help searching :
  - **Where** memory is allocated.
  - **Properties** of allocated chunks.
  - **Bad** allocation **patterns** for performance.
  - **Leaks**
  - **Global variables** (TLS)

Some issue examples


```
Int gblVar[SIZE];
int * func(int size)
{
        child_func_with_allocs();
        void * ptr = new char[size];
        double* ret = new double[size*size*size];
        for (.....)
        {
                double* buffer = new double[size];
                //short and quick do stuff
                delete [] buffer;
        }
        return ret;
}
```

Global variables or TLS

Indirect allocations

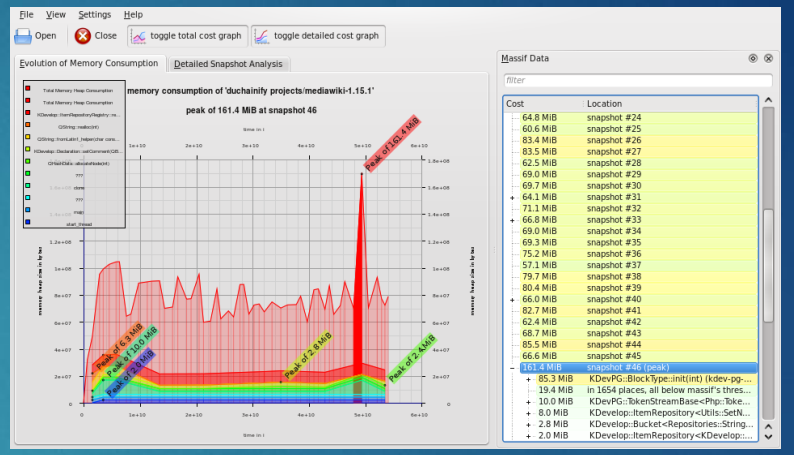
Leak

Might lead to swap for large size

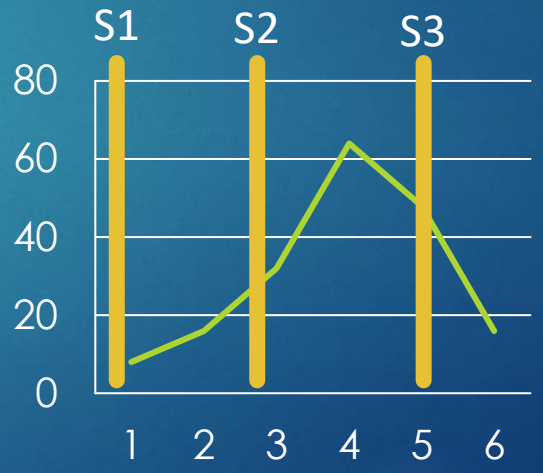
Short life allocations

Existing tools

- Valgrind - massif

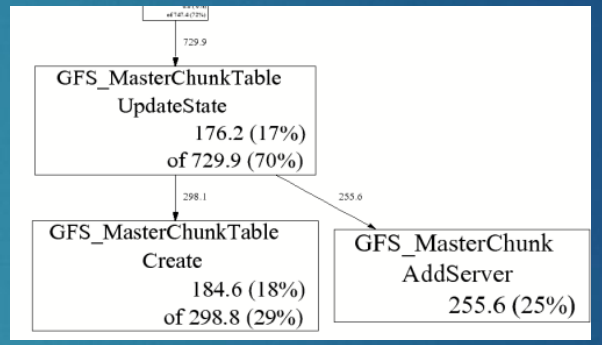


- Valgrind - memcheck

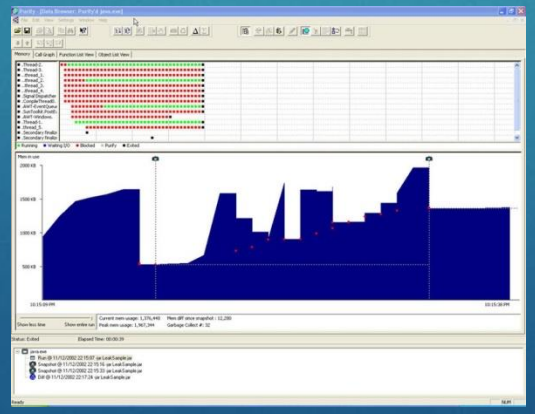


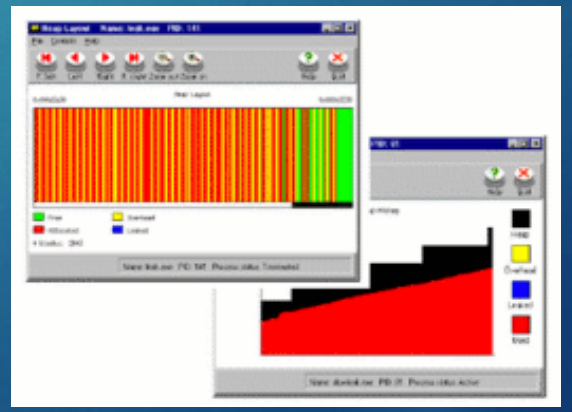
Existing tools

- Google heap profiler (tcmalloc):



- IBM Purify++ / Parasoft Insure++





What I want to provide

- Same **approach** than **valgrind/kcachgind**
- **Mapped allocations** on **sources lines**
- For memory resource usage :
  - **Memory leaks** (malloc without free)
  - **Peak** and total allocated memory
- For performance :
  - Allocation **count**
  - Allocation **sizes** (min/mean/max)
  - Chunk **lifetime** (min/mean/max)

Global summary

Sébastien Valat - INRIA / LJK
15/02/2024

EXECUTION TIME	PHYSICAL MEMORY PEAK	ALLOCATION COUNT	AVAILABLE PHYSICAL MEMORY
00:00:00.25	2.3 MB	379	4.1 Gb

Run description

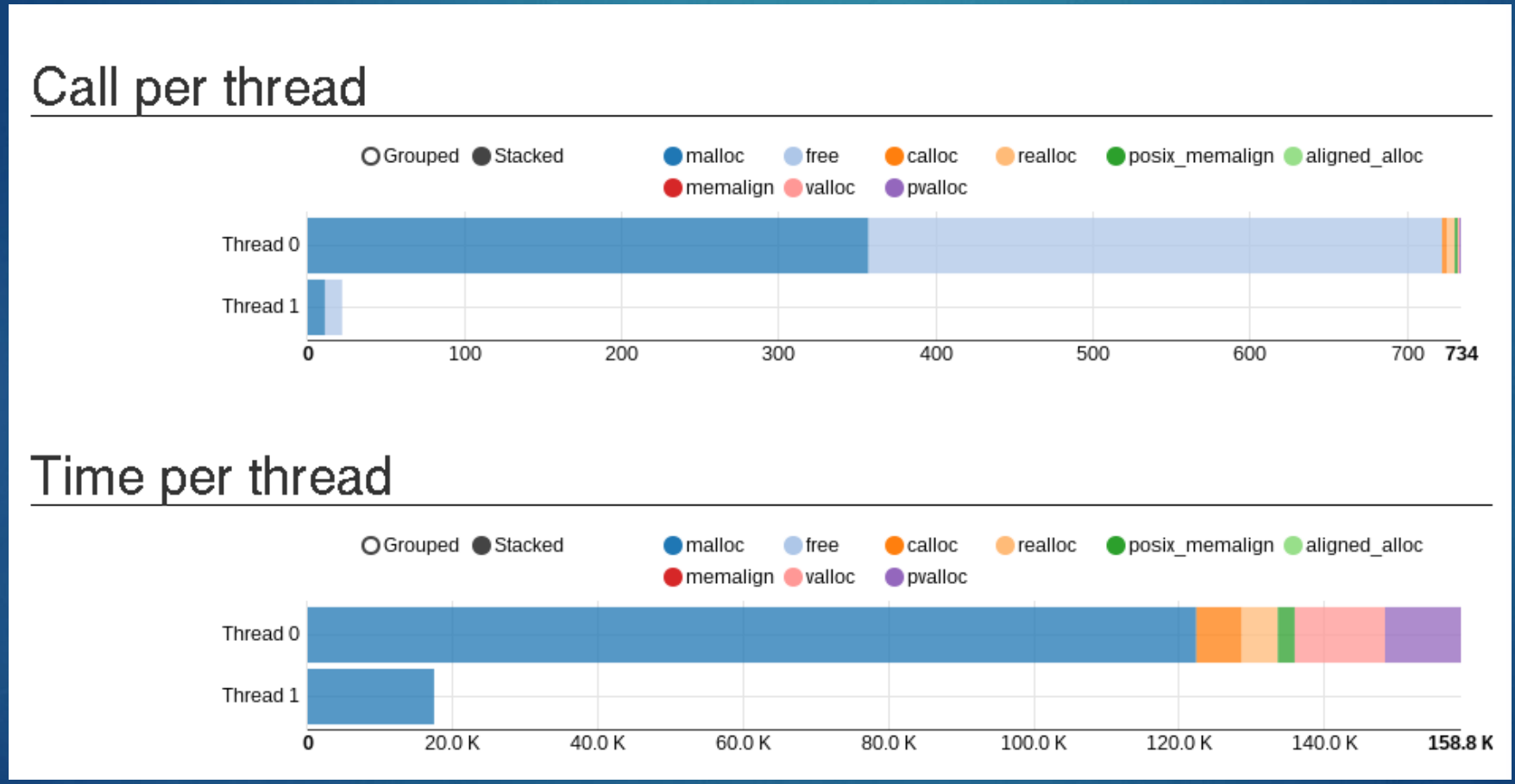
Executable :	simple-case-finstr-linked
Commande :	`./simple-case-finstr-linked`
Tool :	matt-0.0.0
Host :	localhost
Date :	2014-11-26 22:40
Execution time :	00:00:00.25
Ticks frequency :	1.8 GHz

Global statistics

Show all details | Show help

Physical memory peak	2.3 MB
Virtual memory peak	103.7 MB
Requested memory peak	2.0 KB

Per thread statistics

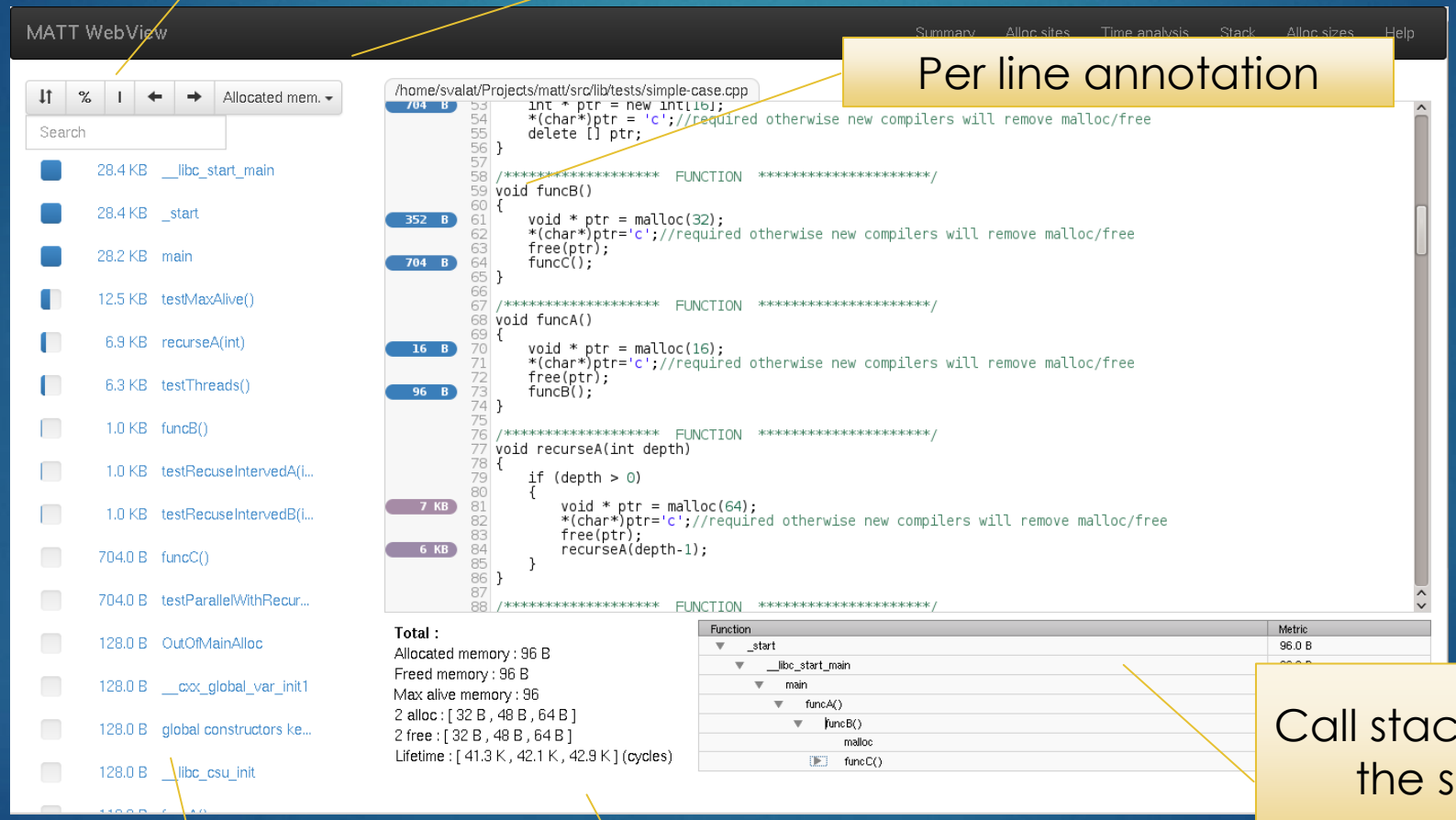


Source annotations

Inclusive/Exclusive

Metric selector

Per line annotation



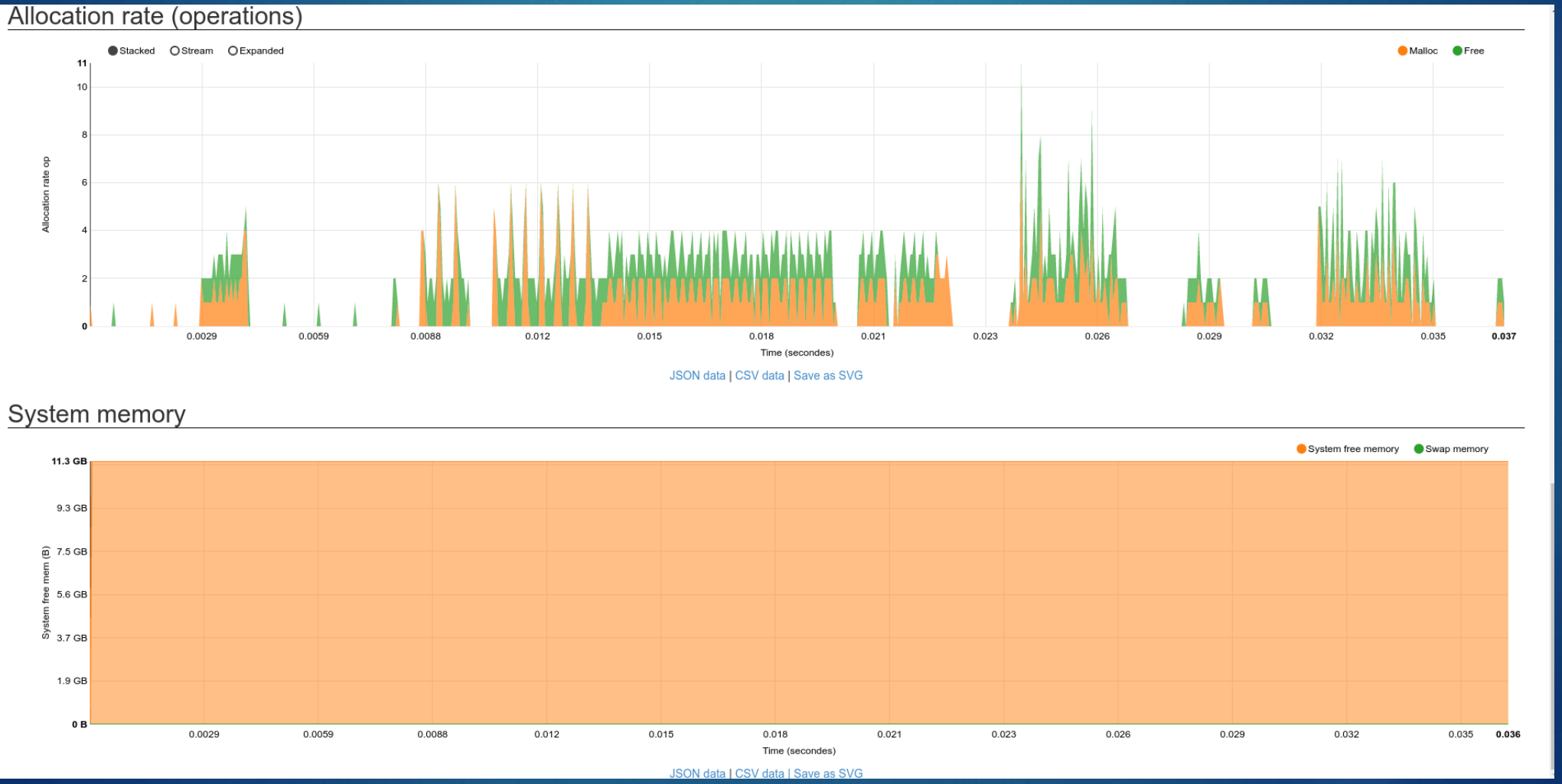
Symbols

Details of symbol or line

Call stacks reaching the selected site.

Time charts

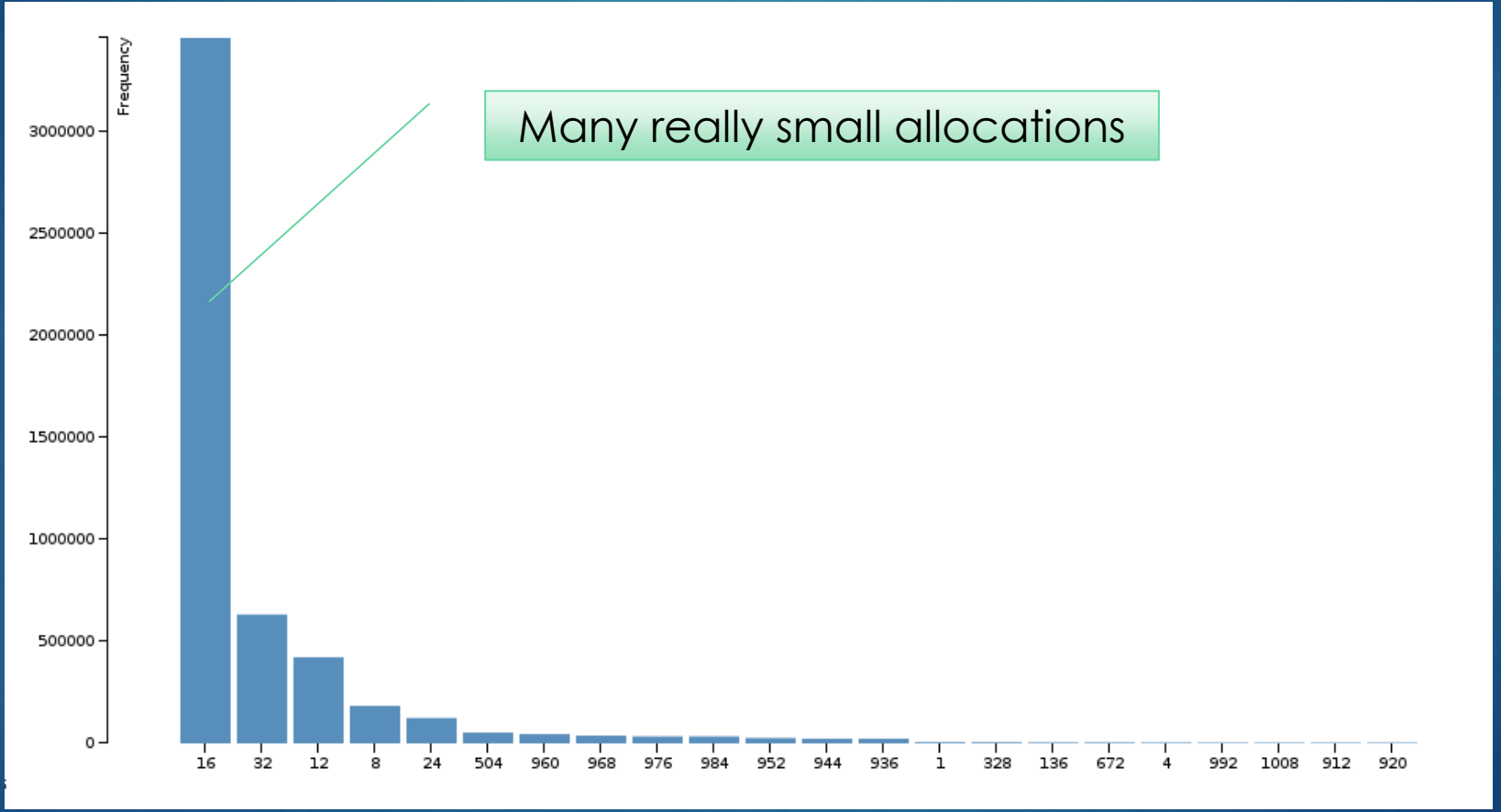
Sébastien Valat - INRIA / LJK
15/02/2024



Chunk size distribution

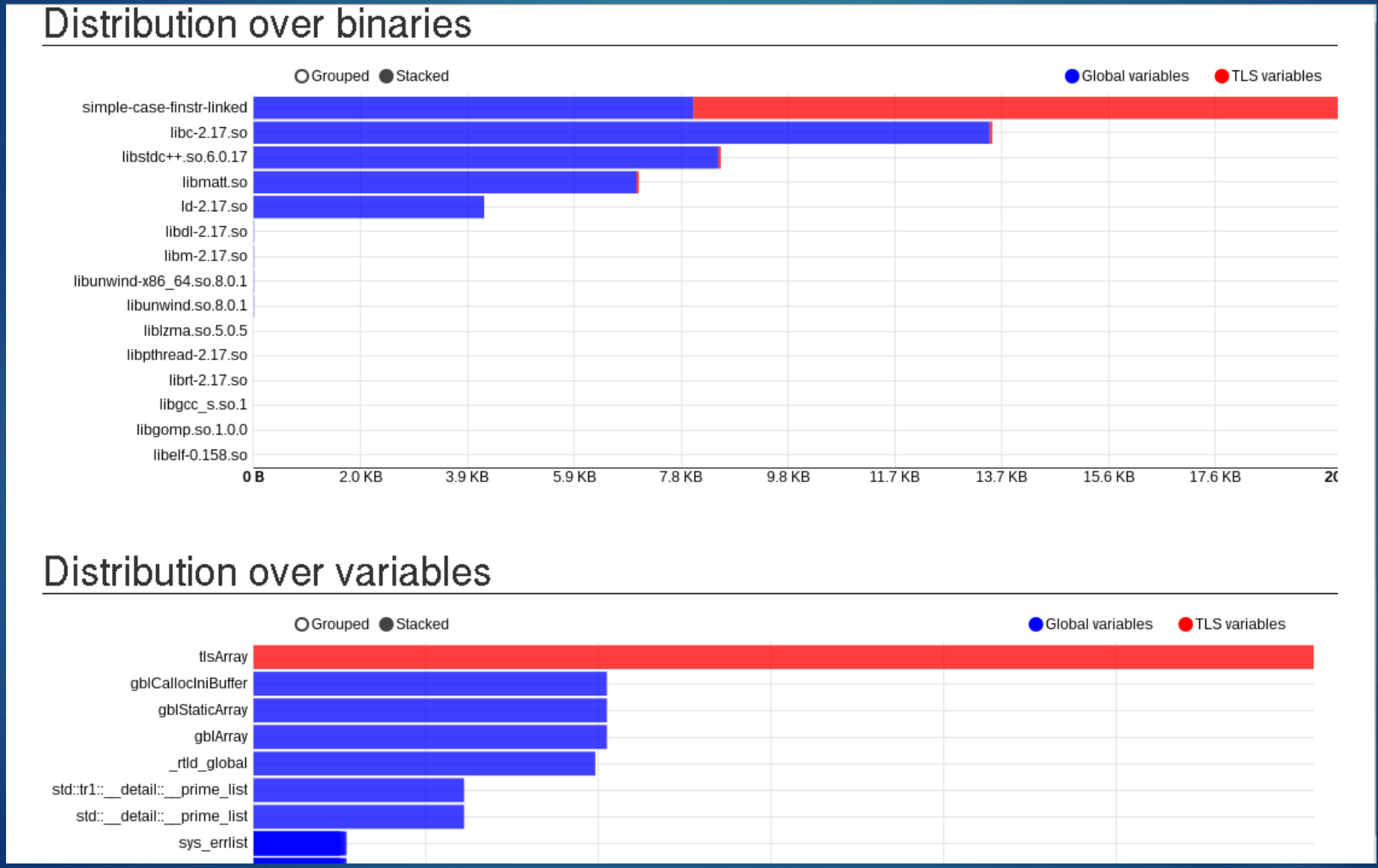
Example from YALES2 with gfortran issue

Sébastien Valat - INRIA / LJK
15/02/2024



Global variables

Sébastien Valat - INRIA / LJK
15/02/2024

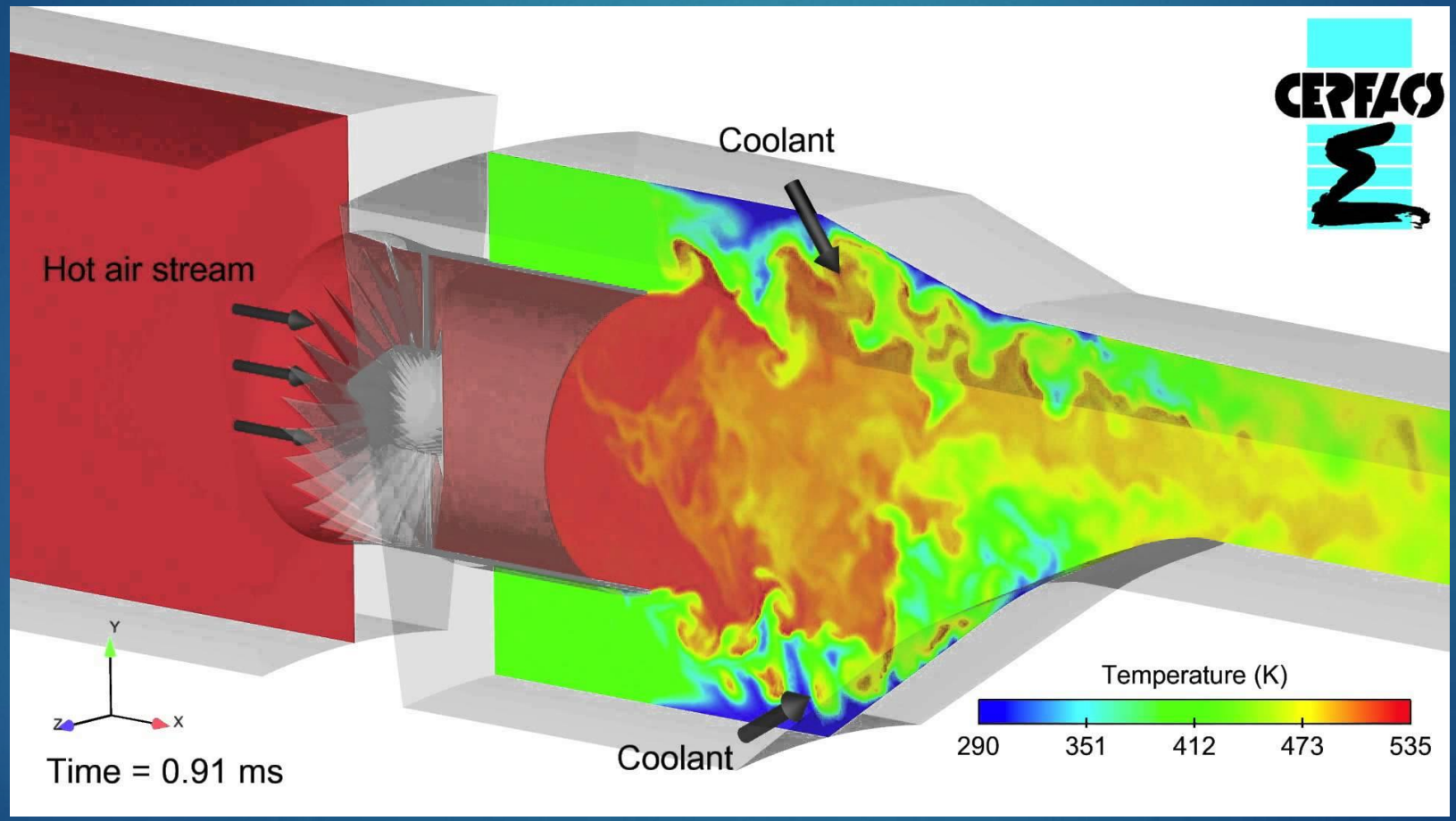


Sébastien Valat - INRIA / LJK
15/02/2024

Real cases

Cerfacs - AVBP- CFD simulator

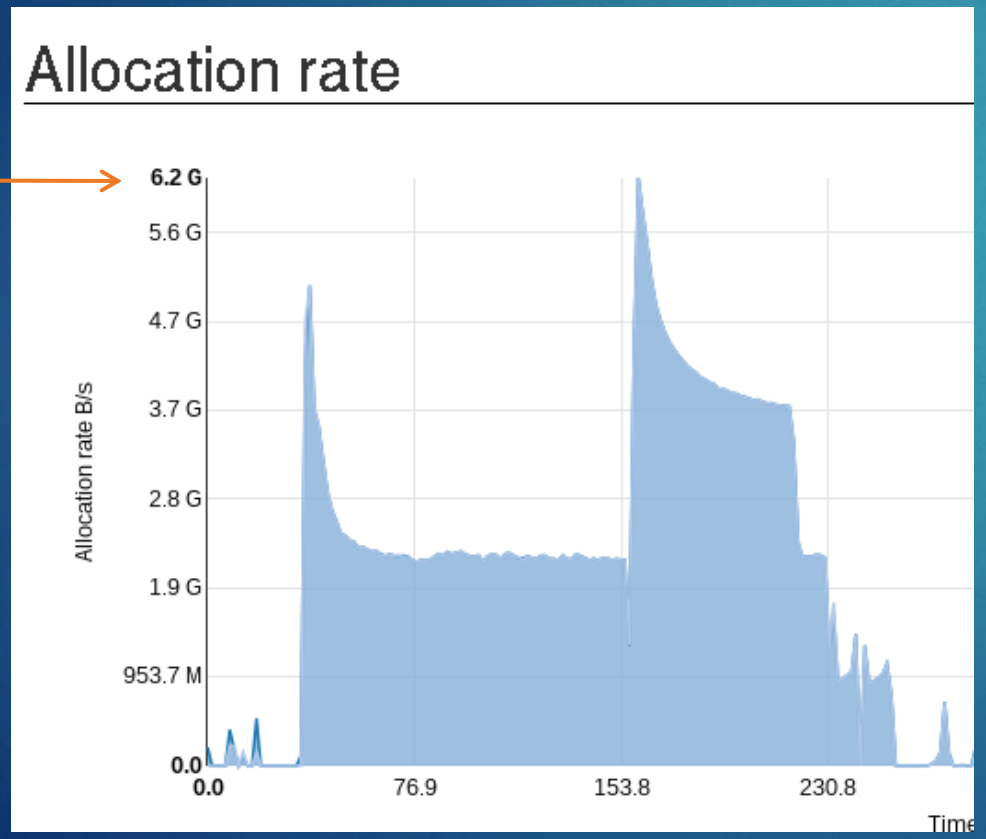
Sébastien Valat - INRIA / LJK
15/02/2024



Example on AVBP init phase

Sébastien Valat - INRIA / LJK
15/02/2024

- Issue with reallocation on init

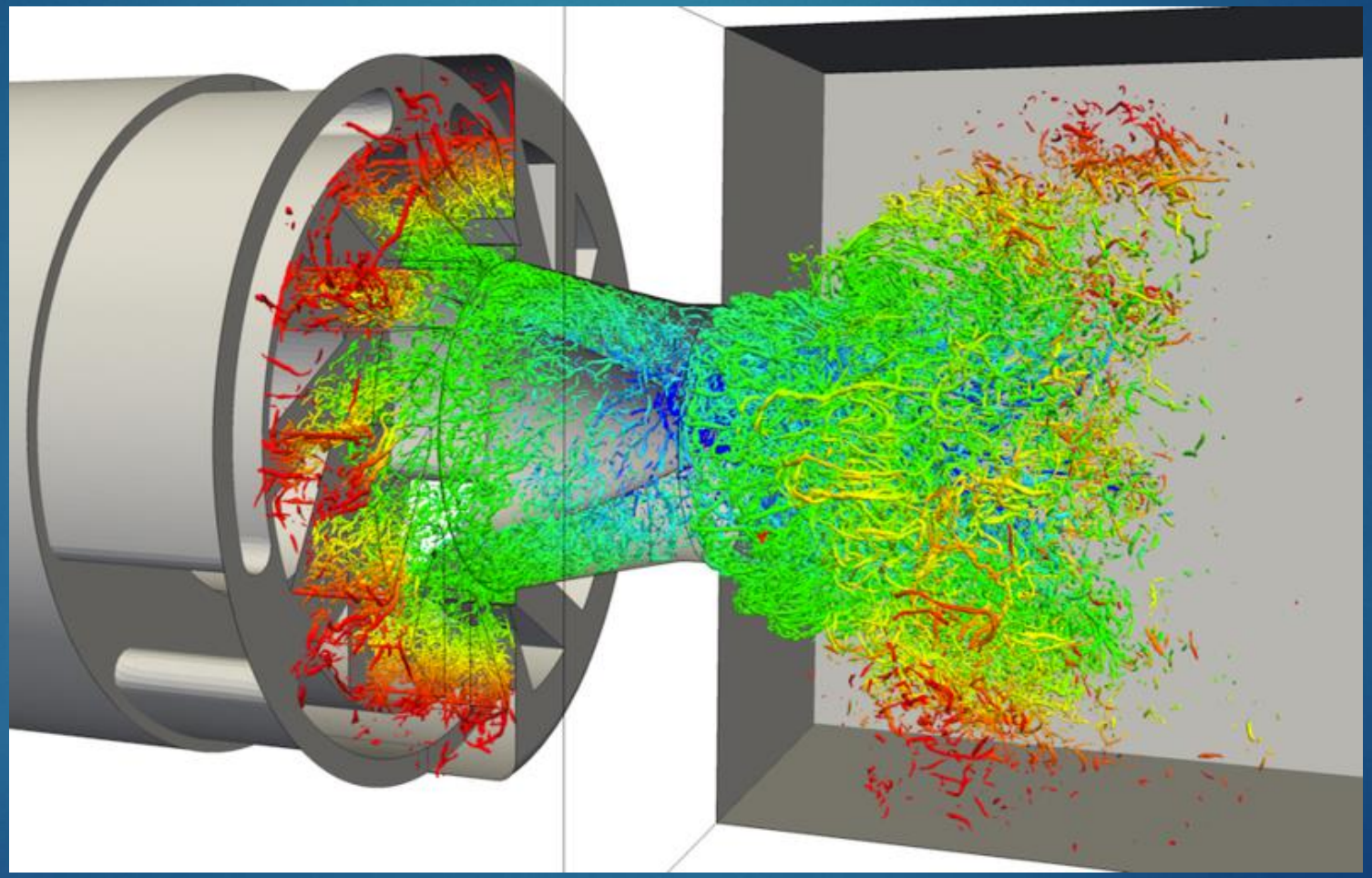



```
100         'array and capacity variable are not
101
102       IF (needed_size>capacity) THEN
103         IF (ALLOCATED ( temp) ) DEALLOCATE(temp)
104 57 GB   ALLOCATE ( temp(capacity))
105
106         DO i=1,capacity
107           temp(i)=array(i)
108         END DO
109
110         DEALLOCATE ( array)
111 57 GB   ALLOCATE ( array(new_cap))
112
113         DO i=1,capacity
114           array(i)=temp(i)
115         END DO
116
117         capacity=new_cap
118       END IF
119
```

Function
▶ _start

Total :
Allocated memory : 56.8 GB
Max alive memory : 135.7 M
3.5 K alloc : [16.0 KB , 16.3 MB , 33.7 MB]
Lifetime : [107.8 K , 26.7 M , 476.7 M] (cycles)
Own :
Allocated memory : 56.8 GB
Max alive memory : 135.7 M
3.5 K alloc : [16.0 KB , 16.3 MB , 33.7 MB]
Lifetime : [107.8 K , 26.7 M , 476.7 M] (cycles)

Coria – YALES2 - Combustion

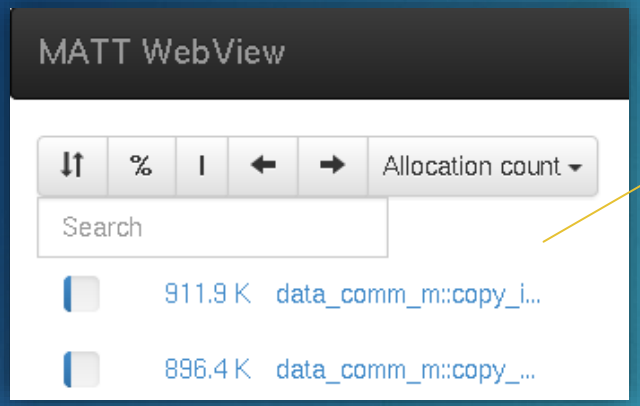


Sébastien Valat - INRIA / LJK

Allocatable arrays on YALES2

Issue only occur with gfortran, ifort uses stack arrays.

Sébastien Valat - INRIA / LJK
15/02/2024



Search intensive alloc functions

Huge number of allocation for a line programmer think it doesn't do any !

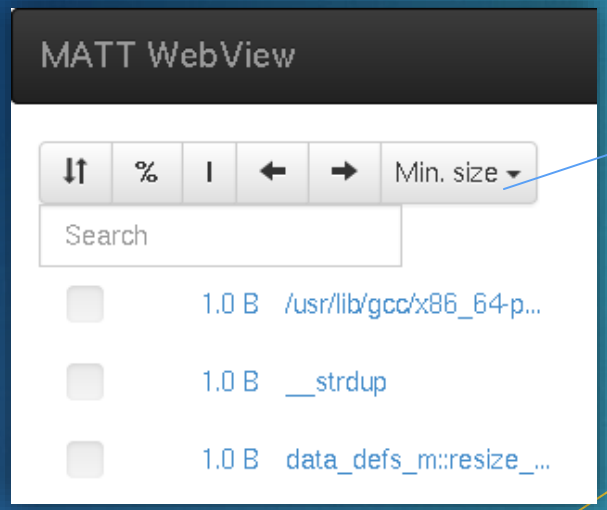
```
608 K
892                 do i=1,nitem_el_grp
893                    el_grp_ind = el_grp_index2int_comm_index%val(1,i)
894                    int_comm_ind = el_grp_index2int_comm_index%val(2,i)
895                    el_grp_r2%val(1:dim1,el_grp_ind) = int_comm_r2%val(1:dim1,int_comm_ind)
896                 end do
```

Total :
Allocated memory : 9.5 MB
Freed memory : 9.5 MB
Max alive memory : 432
608.0 K alloc : [16 B , 16 B , 16 B]
608.0 K free : [16 B , 16 B , 16 B]
Lifetime : [24.5 K , 39.9 K , 37.8 M] (cycles)
Own :
Allocated memory : 9.5 MB

And mostly really small allocations !

We can found allocs of 1B !

Sébastien Valat - INRIA / LJK
15/02/2024



Search for the minimal chunk size.

Many codes produce allocations of 1B.
OK with moderation.

```
530             case (DATATYPE_REAL_NODE_VECTOR,DATATYPE_REAL_ELEM_VECTOR, &
531                   DATATYPE_REAL_FACE_VECTOR,DATATYPE_REAL_PAIR_VECTOR)
532                if (associated(data_ptr%r2_ptrs)) then
533                   deallocate(data_ptr%r2_ptrs)
534                end if
1 B 535                allocate(data_ptr%r2_ptrs(nel_grps))
536                do n=1,nel_grps
537                   NULLIFY(data_ptr%r2_ptrs(n)%ptr)
538                end do
```

Usage & conclusion

Usage

- Backtrace mode :

```
# Optionally recompile with debug flag  to get source lines :
cc –g …
# Run your program
${PREFIX}/bin/malt [--config=file.ini] YOUR_PRGM [OPTIONS]
```

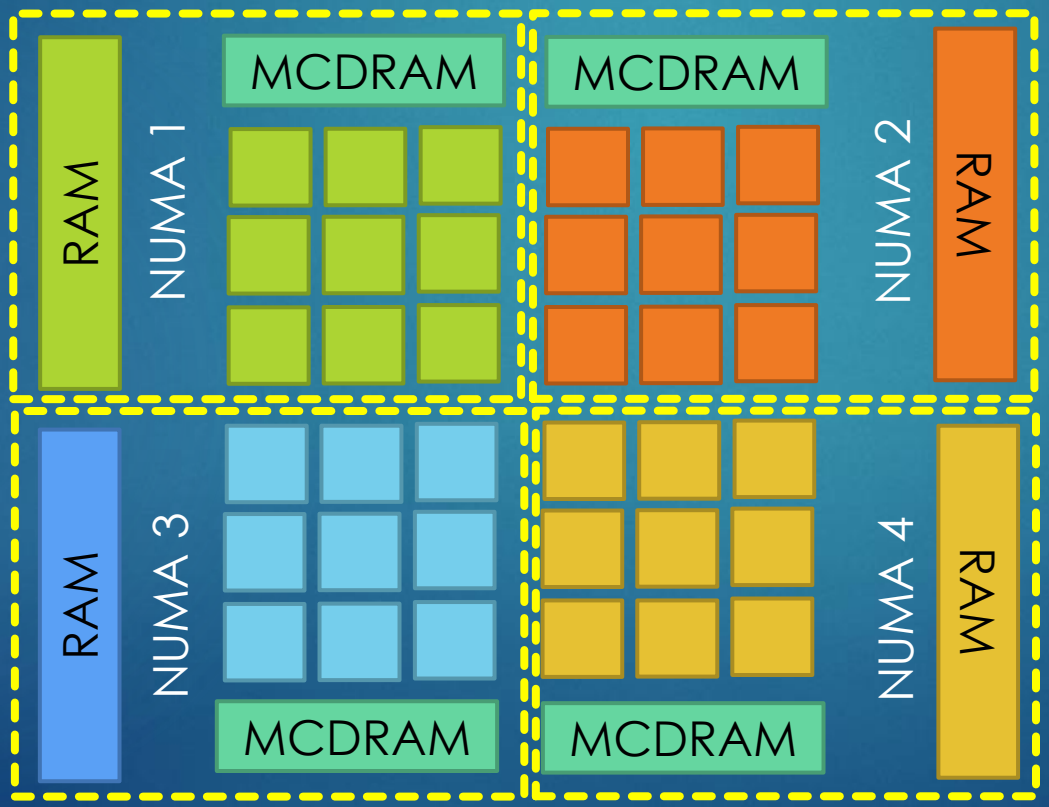
- Use the web view :

```
#Launch the server
malt-webserver -i  malt-{YOUR_PRGM}-{PID}.json
# Connect with your browser on http://localhost:8080
```

Reminder on NUMA

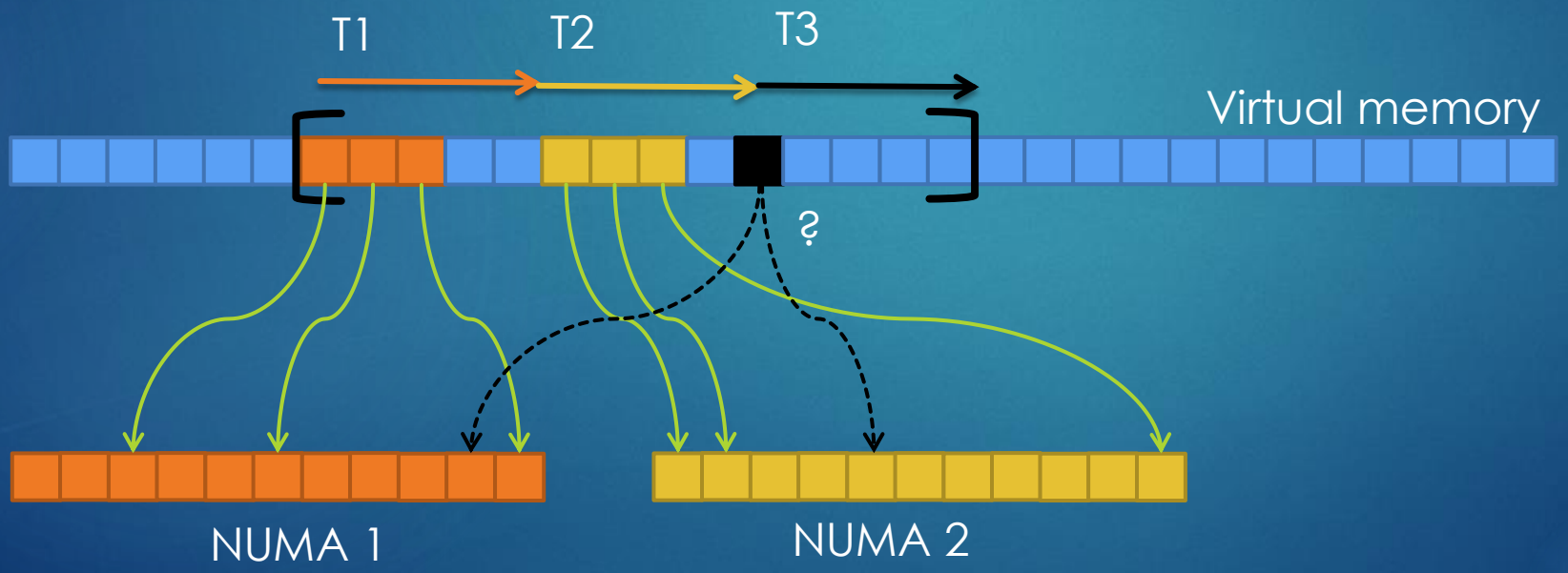
Today topology

- Example current **Intel Knight Landing**, mode **SNC2** or **SNC4**
- Also add fast memory **MCDRAM** presented as **NUMA** or **LLC cache**



Implicit binding : first touch

- New allocated segments are **physically empty**
- They are filled on **first touch**
- Page selection **depend** of the **thread position**



Typical OpenMP mistake

- Make first **init outside of OpenMP** (in thread 1)
- So **each pages** will be first touched **on NUMA 1**

```
#pragma omp parallel for
for (int i = 0 ; i < SIZE ; i++)
        array[i] = 0;
```

- Then access

```
#pragma omp parallel for
for (int i = 0 ; i < SIZE ; i++)
        array[i]++;
```

- **Bad performance** due to remote accesses !

Wish list for a profiling tool…

- We want to know if we make **remote accesses**
- Ideally we need to know **where**…
- We can dream, we want to know **which allocation contain issues**
- We want to know **where** the **first touch** has been done
- On KNL we want to check **MCRAM accesses**

Sébastien Valat - INRIA / LJK
15/02/2024

NUMPROF

HOW TO KNOW IF WE ARE RIGHT IN A REAL APPLICATION ?

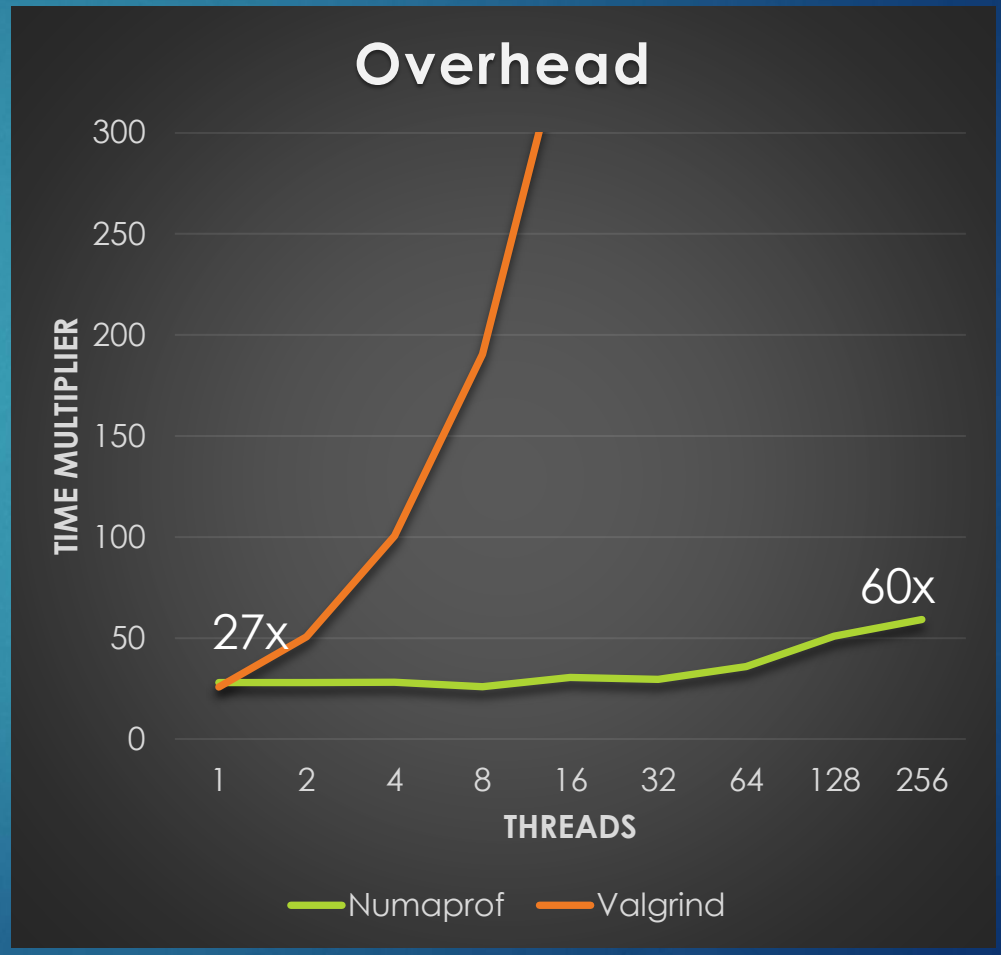
NUMAPROF

- Take back the idea from **MALT**
  - **Web interface**
  - **Source annotation**
  - **Global metrics**

- Use intel **Pin**
  - Permit to **instrument** all **memory accesses**
  - **Parallel** opposite to valgrind
  - **Difficulty**: we cannot easily use libs inside the tool
  - I would have used hwloc and libnuma.....

Overhead and scalability

- Of course overhead is large: **~30x**
- But is **scale**
- Example code hydro on **KNL**:

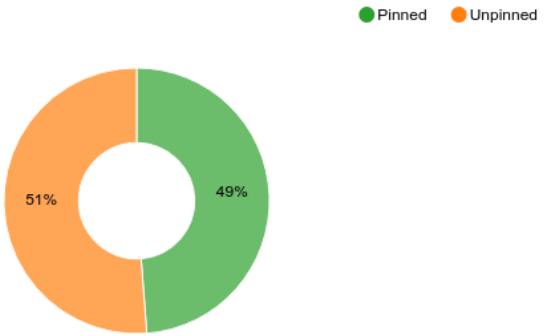


GUI and example

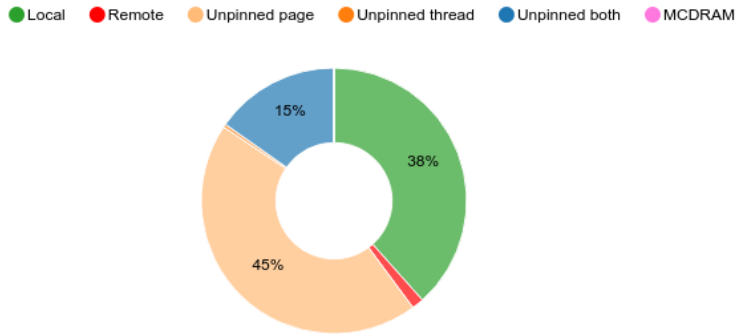
Global summary

Sébastien Valat - INRIA / LJK
15/02/2024

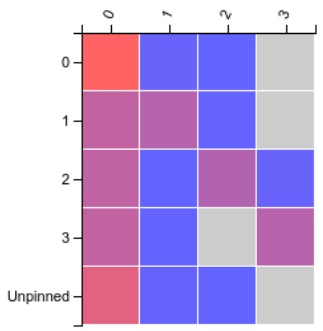
First touch



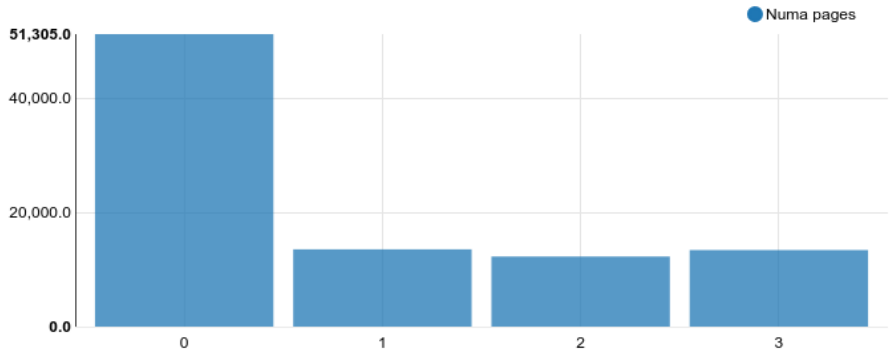
Memory access



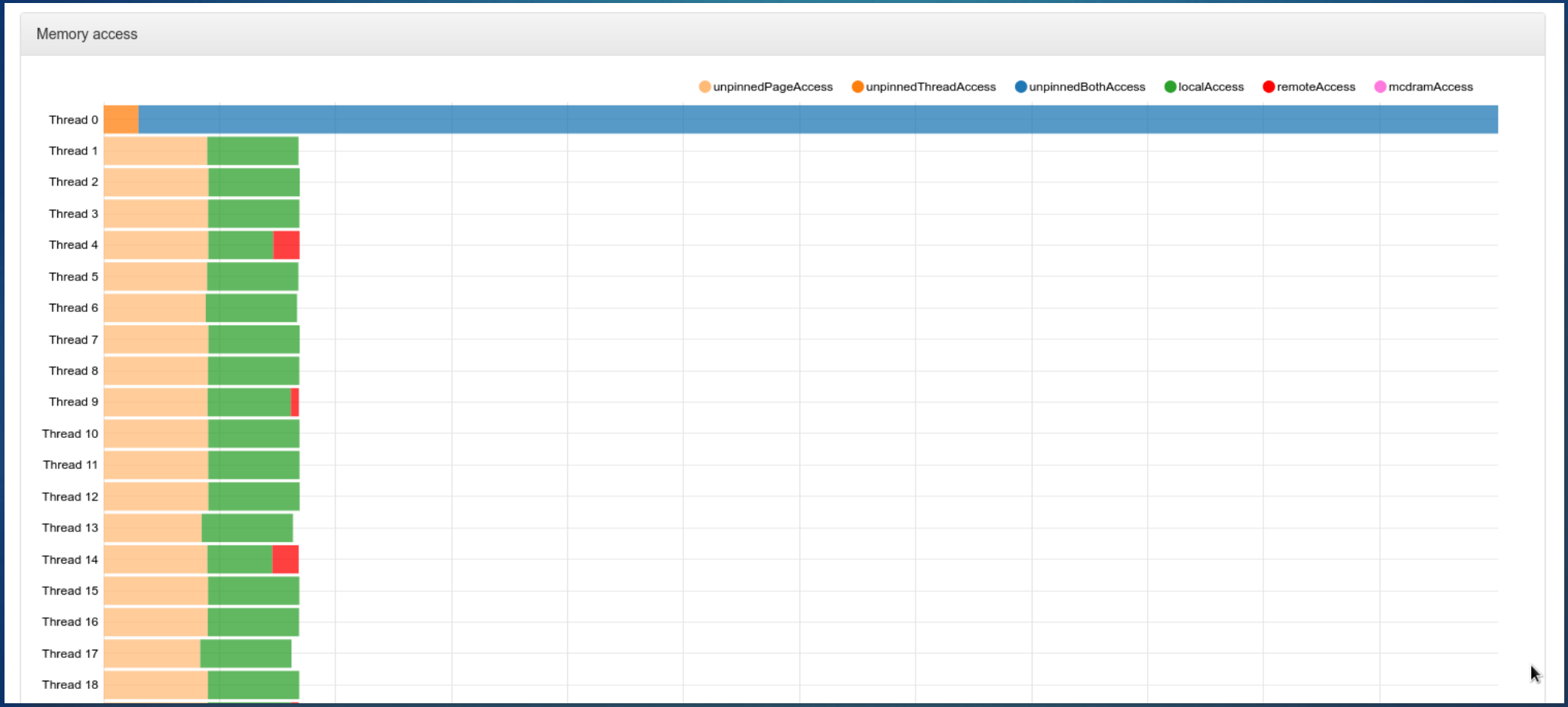
Access matrix



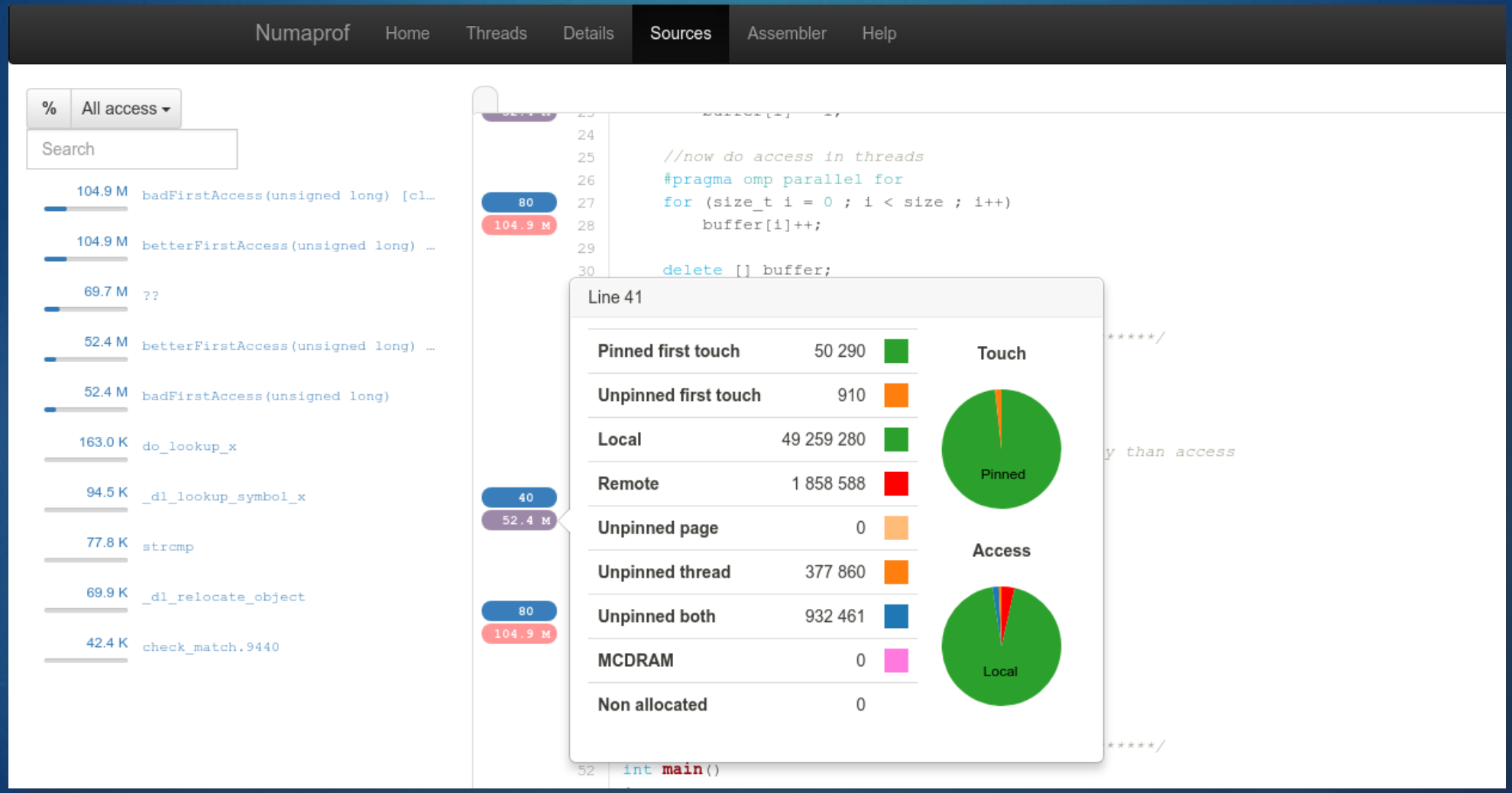
Peak allocated numa pages



Statistics per thread

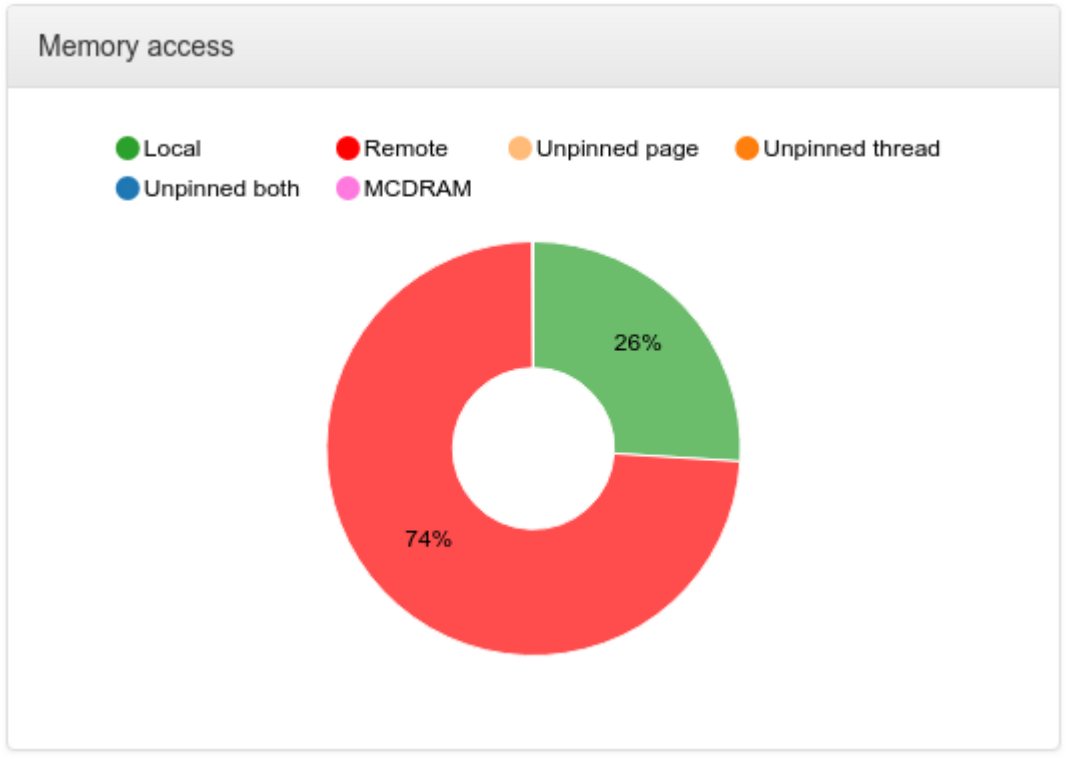


Source & asm annotations

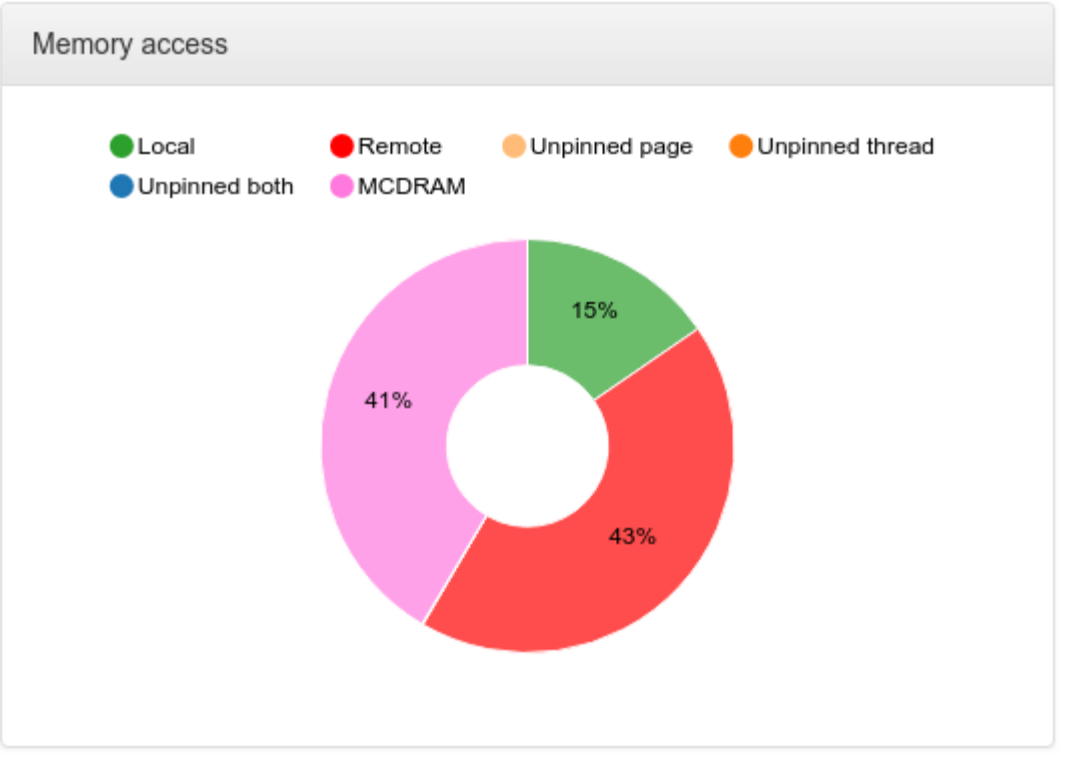


Code Hydro

- KNL Without HBM

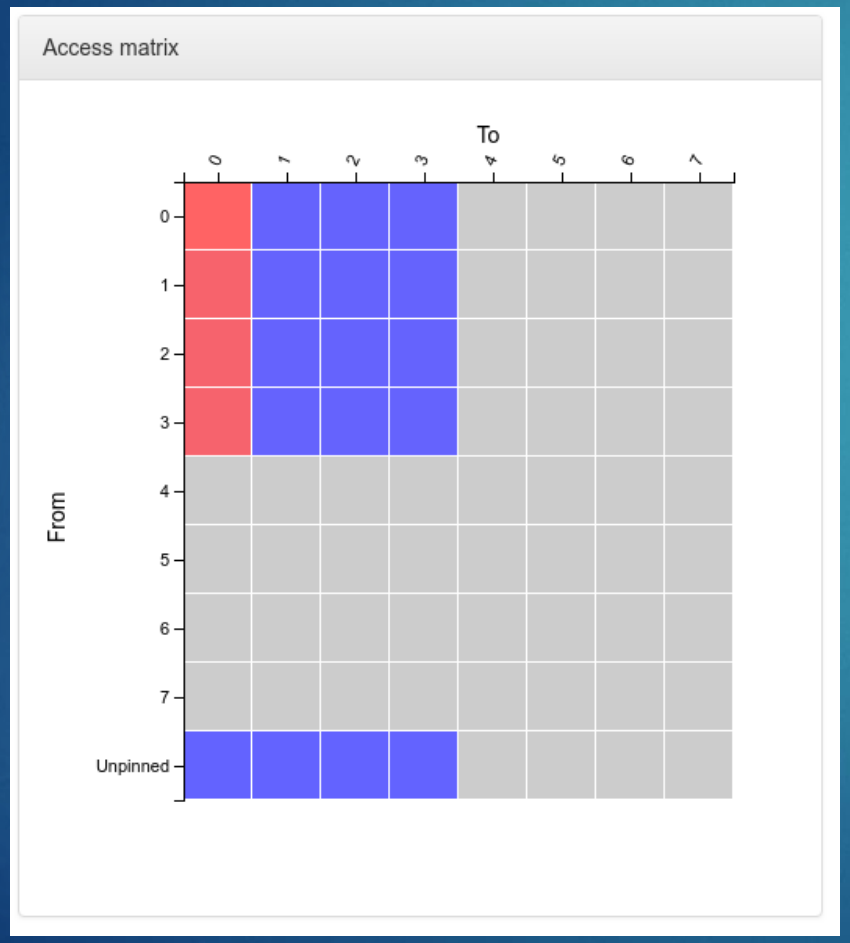


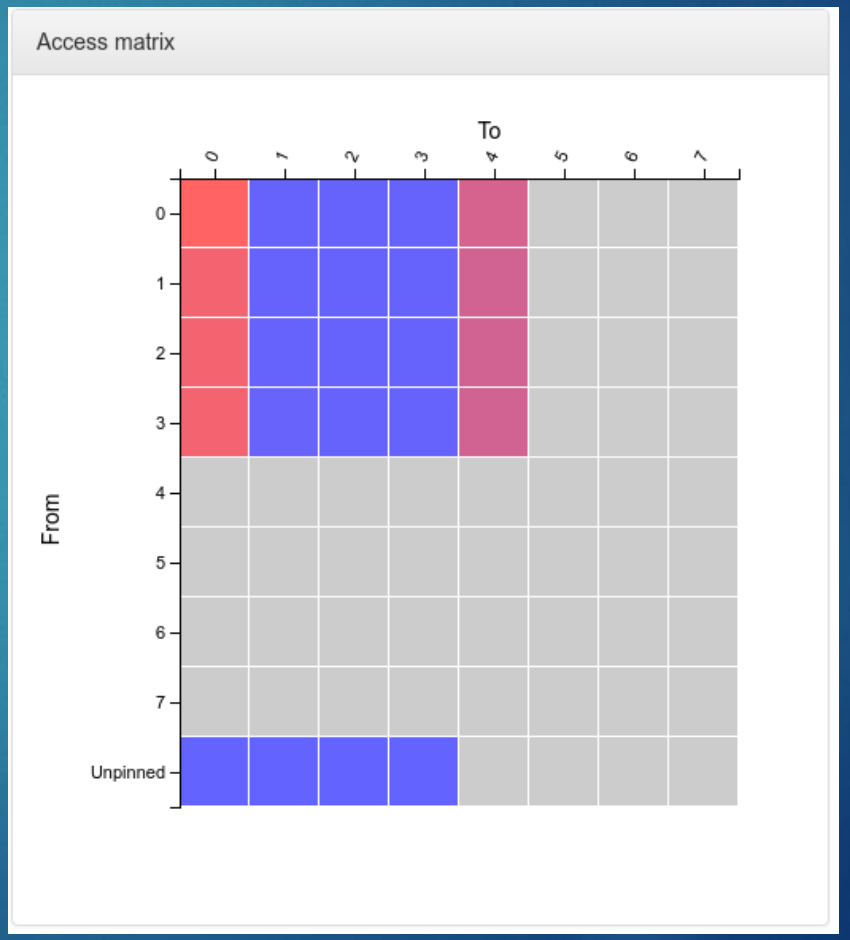
WITH HBM



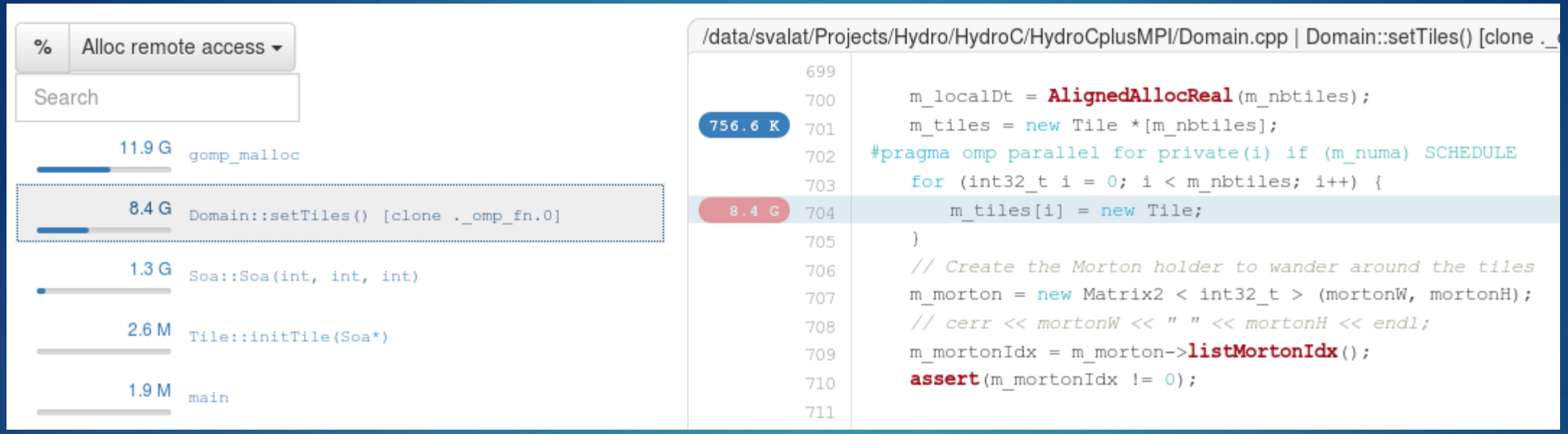
Original Hydro access matrix

Sébastien Valat - INRIA / LJK
15/02/2024



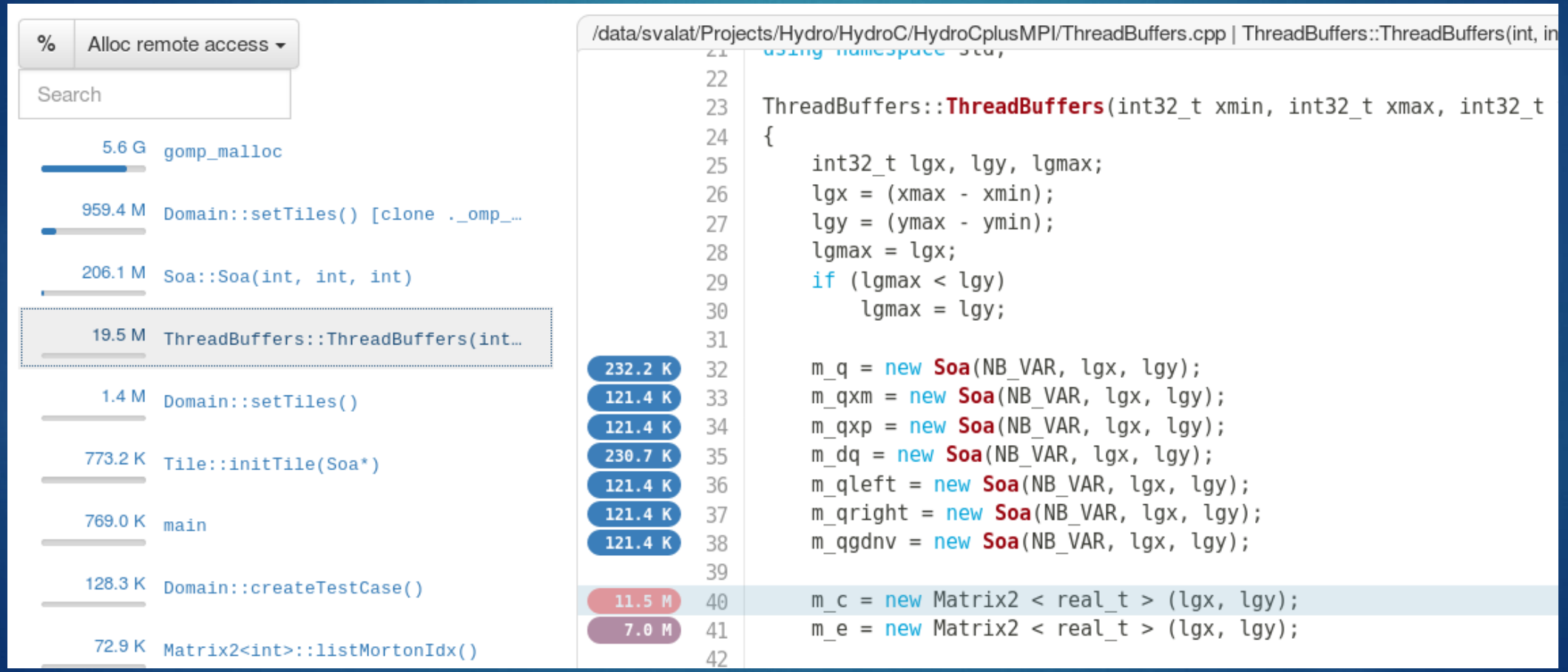


Ordering issue




```
#pragma omp parallel for private(i) if (m_numa) SCHEDULE
        for (int32_t i = 0; i < m_nbtiles; i++) {
                int t = m_mortonIdx[i];
                m_tiles[t] = new Tile;
        }
```

Non parallel allocations



Parallel allocations

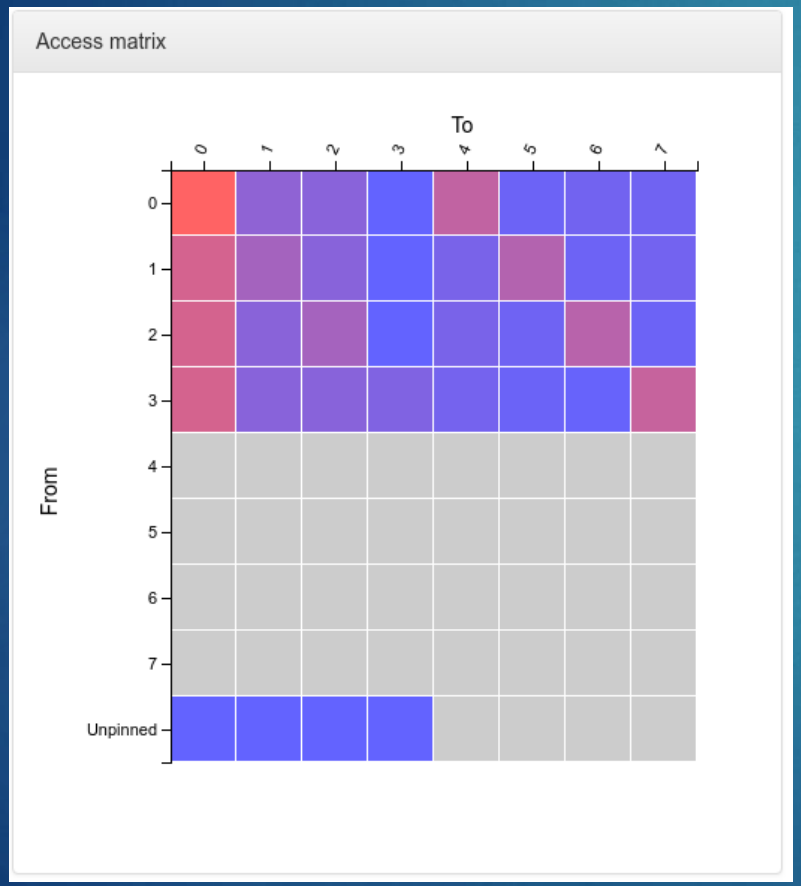
- Original

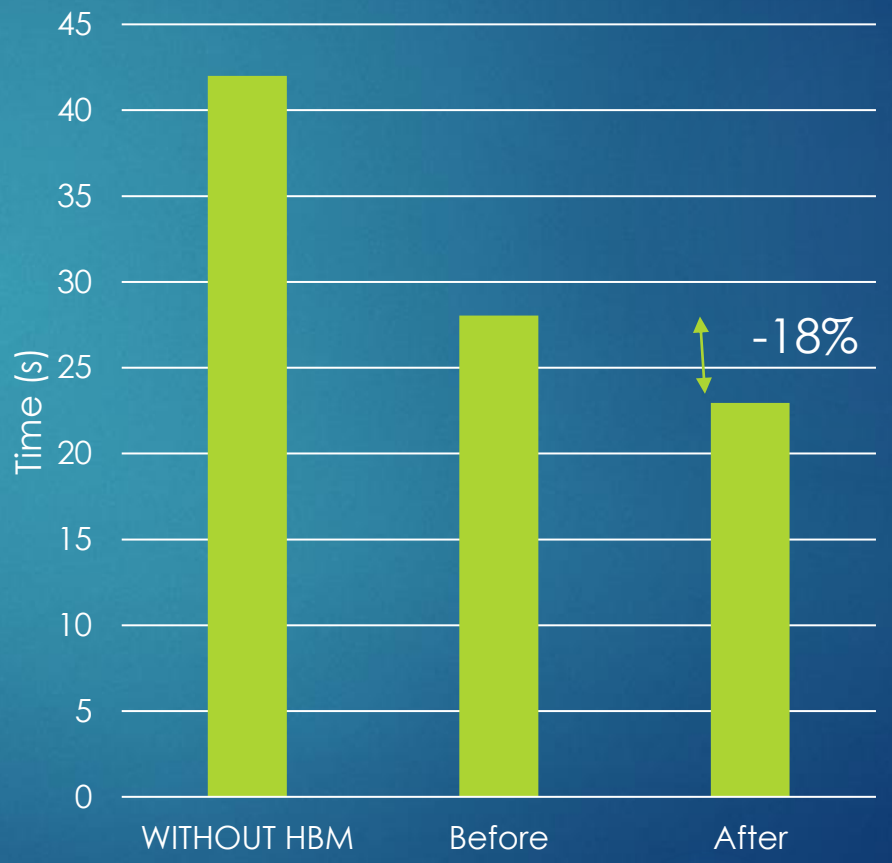
```
for (int32_t i = 0; i < m_numThreads; i++) {
        m_buffers[i] = new ThreadBuffers(…);
        assert(m_buffers[i] != 0);
}
```

- Modified

```
#pragma omp parallel
{
        int i = omp_get_thread_num();
         #pragma omp critical
        m_buffers[i] = new ThreadBuffers(..);
        assert(m_buffers[i] != 0);
}
```

Speed up obtained on Hydro





Conclusion

- **Memory** is **not trivial** to handle **in large programs**
- Need to be taken in account
- Some **tiny mistakes** sometimes **cost a lot**
  - **Possibly everywhere** in the program (**global impact**)
- Be able to get a view is a first help