MALT & NUMAPROF, Memory Profiling for HPC Applications
SÉBASTIEN VALAT – FOSDEM 2019 – TRACK HPC
Origin of the tools

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

![Exascale](image)

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

![CERN](image)
Motivation

Lot of issues today:

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

Access:

- **NUMA** (Non Uniform Memory Access)
- **Memory wall!**
You need to well understand memory behavior of your (HPC) application!
Eg: >1M lines C++ simulation.  
On 128 cores / 16 NUMA CPUs

My PhD.

Available

Execution time (s)

35%

58%

20%

MPC/NUMA  MPC/UMA  Glibc  jemalloc  tcmalloc

User  System  Idle
Same about **memory consumption** on 12 cores

![Physical memory consumption diagram](image)

- glibc
- jemalloc
- tcmalloc

Physical mem. (GB)

2.5x
Tool 1: MALT

- **Memory management** can have **huge impact**
- Tool to **track mallocs**
- Report **properties** onto **annotated sources**

- Same **idea** than **valgrind/kcachegrind**
  - Annotated sources
  - Annotated call graphs
  - + **Non additive metrics** (for inclusive costs, eg. lifetime)
  - + **Time charts**
  - + **Properties distribution** (sizes....)
Web based GUI

Inclusive/Exclusive

Metric selector

Per line annotation

Symbols

Details of symbol or line

Call stacks reaching the selected site.
Example of time based view

Size over time

Lifetime over size
Tool 2: NUMAPROF

- Based on MALT code
- But about NUMA
- How to detect remote memory accesses
- Unsafe & uncontrolled memory binding
Some summary views
Still source annotation to understand code
Short success

- **MALT**
  - 20% CPU saving on my **CERN 32 000** C++ code.
  - Improvement on 2 **commercial simulation** codes
  - Profiled **CERN LHCb 1.5 million** line C++ code

- **NUMAPROF**
  - 20% perf in 20 minutes on 8000 lines simu.
  - NUMA **Linux kernel policy bug** detected.
  - CERN PhD. code **NUMA correctness**
Questions

Both tools under CeCILL-C on http://memtt.github.io
My researches: http://svalat.github.io
Example of success

MALT

- Reduce **CPU usage** of 30% on the CERN app I was developing (mistake with C++11)
  - 32 000 C++ lines running on 500 servers.

- **Too large allocations** in a PhD. Student numerical simulation running on 500 cores while developing the tool.

- **Realloc pattern** in Fortran into an **industrial** R&D simulation code

- Unexpected **allocs generated** by GFortran **compiler** on another **industrial** R&D simulation **code**.

- Successfully ran on **CERN LHCb 1.5M lines** online analysis software
Example of success
NUMAPROF

- **20% performance improvement** in 20 minutes on an unknown 8000 C++ lines simulation on Intel KNL

- **Linux Kernel bug** detected on NUMA management in conjunction with Transparent Huge Pages (while developing the tool). Was detected at same time by other way by Red-Hat.... But.....

- **Confirmation** of NUMA correctness on a CERN/OpenLab PhD. Student code on Intel KNL