# Introduction to FastFlow programming

Massimo Torquati    <torquati@di.unipi.it>

Computer Science Department, University of Pisa - Italy

# Debugging Tools

- Many debugging tools available (open source and not)

- For Linux OS the de-facto standard tool is **gdb**

- Debugging programs with multiple threads is not easy

  - https://sourceware.org/gdb/onlinedocs/gdb/Threads.html

  - Take a look at least to the following commands:

    - info threads

    - thread *threadno*

    - bt (backtrace)

- **valgrind** (http://www.valgrind.org/)

  - very useful to find memory leaks

  - take a look at the Helgrind tool and the DRD tool

  - valgrind --tool=helgrind/drd

# Profiling Tools

- Many tools available (open source and not)

- **oprofile** (http://oprofile.sourceforge.net/doc/)

  - very powerful open-source system profiler

- **valgrind + cachegrind**

  - valgrind --tool=cachegrind

- **PAPI** (http://icl.cs.utk.edu/papi/)

  - very useful if you have to profile a specific piece of code

- **Intel vtune amplifier**

  - Tutorials available here:

    https://software.intel.com/en-us/articles/intel-vtune-amplifier-tutorials


- Let's have a look at vtune amplifier using a simple example

# FastFlow memory allocator

- The standard allocator is not very efficient when allocating small memory areas

- FastFlow provides a memory allocator for this cases

  - …. but the interface is not equal to the standard one

  - allocates large chunk of memory slicing them into smaller chunks

- The allocator has been optimized for the patterns

  - 1-to-1  1 thread executing malloc 1 thread executing free

    malloc                    free

  - 1-to-N 1 thread executing malloc N threads executing free

                              free

    malloc                    free

                              free
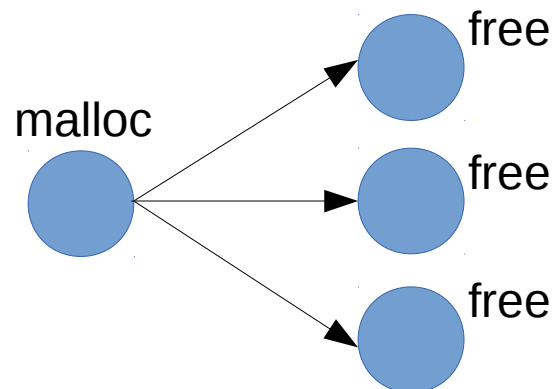
# FastFlow memory allocator

- 2 different interfaces: (<ff/allocator.hpp>)

  - **ff_allocator**: can be used only for the patterns described before (only one specific thread can call malloc)

    - The threads calling malloc has to register himself as an allocating thread and has to initialize the allocator.

  - **FFAllocator**: can be used by any threads regardless they are allocating or deallocating memory areas

- Let's have a look at an example showing how to use the ff_allocator

  - alloc_std.cpp and alloc_ff.cpp inside the public/Allocator folder