Introduction to FastFlow programming

SPM lecture, December 2016

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

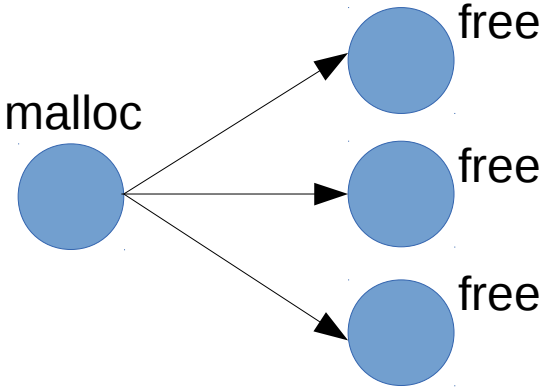
- Many tools available (open source and not)
- **oprofile** (http://oprofile.sourceforge.net/doc/)
  - very powerful open-source system profiler
- **valgrind + cachegrind**
  - valgrind --tool=cachegrind
- **PAPI** (http://icl.cs.utk.edu/papi/)
  - very useful if you have to profile a specific piece of code
- **Intel vtune amplifier**
  - Tutorials available here:

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

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

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



FastFlow memory allocator

- 2 different interfaces: (see <fastflow-home>/ff/allocator.hpp file)
  - **ff_allocator**: can be used only for the patterns described before (only one specific thread can call malloc)
    - The thread 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

- Take a look at the code contained in the public/Allocator folder of the course machine
  - alloc_std.cpp and alloc_ff.cpp

General Purpose Efficient Allocators

- Hoard allocator (http://www.hoard.org/)
- Intel TBB allocator (Intel web site, provided with the TBB framework)
- Jemalloc allocator (http://jemalloc.net/)

- All of them can be used as drop-in replacement of the standard libc allocator by setting LD_PRELOAD env variable
- For example:

  export LD_PRELOAD="${JEMALLOC_HOME}/lib/libjemalloc.so.2"

  or simply

  LD_PRELOAD="${JEMALLOC_HOME}/lib/libjemalloc.so.2" **your-app**