Interactive Performance Profiling and Debugging

Kevin Pouget
Jean-François Méhaut, Miguel Santana

Université Grenoble Alpes / LIG, STMicroelectronics, France
Nano2017-DEMA project

NANO2017/DEMA Meeting, Grenoble
June 24th, 2016

Profiling today

- `gprof`
  - compile-time instrumentation of function calls
  - whole execution
- `perf stat`
  - OS hooks for hardware counters
  - whole execution or attach/detach
- `papi`
  - OS hooks for hardware counters
  - manual code instrumentation
- trace-based post-mortem analyzers
  - runtime library instrumentation

Profiling today

- `gprof`
  - compile-time instrumentation of function calls
  - whole execution
- `perf stat`
  - OS hooks for hardware counters
  - whole execution or attach/detach
- `papi`
  - OS hooks for hardware counters
  - manual code instrumentation
- trace-based post-mortem analyzers
  - runtime library instrumentation

Profiling today

- `gprof`
  - compile-time instrumentation of function calls
  - whole execution
- `perf stat`
  - OS hooks for hardware counters
  - whole execution or attach/detach
- `papi`
  - OS hooks for hardware counters
  - manual code instrumentation
- trace-based post-mortem analyzers
  - runtime library instrumentation

No efficient way to interactively profile regions of interest

Profiling today

No efficient way to interactively profile regions of interest

Source-level interactive debuggers

Profiling today

No efficient way to interactively profile regions of interest

Source-level interactive debuggers

- Extensive control over the process execution
- Powerful (and intuitive ?) command-line user interface

Profiling today

No efficient way to interactively profile regions of interest

Source-level interactive debuggers

- Extensive control over the process execution
- Powerful (and intuitive ?) command-line user interface
- Ability to dynamically compile and insert code in the process

Profiling today

No efficient way to interactively profile regions of interest

Source-level interactive debuggers

- Extensive control over the process execution
- Powerful (and intuitive ?) command-line user interface
- Ability to dynamically compile and insert code in the process

Let's combine them into an interactive profiling tool!

Profiling today

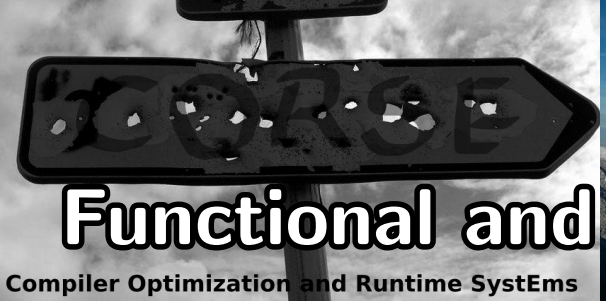
No efficient way to interactively profile regions of interest

Source-level interactive debuggers

- Extensive control over the process execution
- Powerful (and intuitive ?) command-line user interface
- Ability to dynamically compile and insert code in the process

Let's combine them into an interactive profiling tool![a]

[a] only sequential executions / at language-level for now.



Functional and Performance Source-level Debugging

Compiler Optimization and Runtime SystEms

Functional Interactive debugging

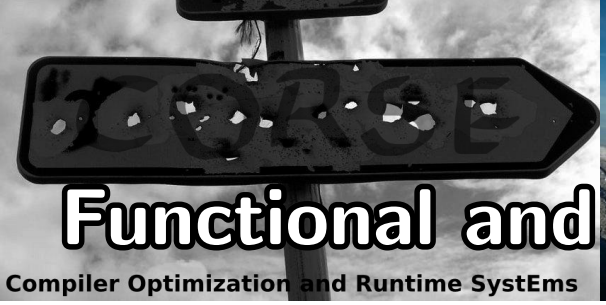
- Navigate in the application execution
- Study its current memory state and execution paths
- Test and verify debugging hypothesis

Functional Interactive debugging

- Navigate in the application execution
- Study its current memory state and execution paths
- Test and verify debugging hypothesis

Performance Interactive Debugging

- Navigate in the application execution
- Profile specific function/region execution
- Test and verify debugging hypothesis



Functional and Performance Source-level Debugging

Compiler Optimization and Runtime SystEms

Functional Interactive debugging

- Navigate in the application execution
- Study its current memory state and execution paths
- Test and verify debugging hypothesis

Performance Interactive Debugging

- Navigate in the application execution
- Profile specific function/region execution
- Test and verify debugging hypothesis

Functional Interactive debugging

- Navigate in the application execution
- Study its current memory state and execution paths
- Test and verify debugging hypothesis

Performance Interactive Debugging

- Navigate in the application execution
- Profile specific function/region execution
- Test and verify debugging hypothesis

Experimentation prototype developed as a GDB Python extension

What to profile?

- During the normal execution:
  - the whole
  - a function execution
  - a region (from line ... to line ... — breakpoints involved)
  - manual start and stops

What to profile?

- During the normal execution:
  - the whole
  - a function execution
  - a region (from line ... to line ... — breakpoints involved)
  - manual start and stops
- Outside of the normal execution:
  - we'll discuss it a bit later

What to profile?

- During the normal execution
- Outside of the normal execution

What is measured?

- `/proc/PID/...` values (*mem usage*, *context switches*, ...)
- `perf stat` counters
  - `LD_PRELOAD` patch to support on-demand counter dump
  - *task-clock*, *context-switches*, *cpu-migrations*, *page-faults*, *cycles*, *instructions*, *branches*, ...
- `gprof` counters
  - preliminary and on-going (inspection of `glibc` profiling counters)
  - call-graph, time per function, ...
- function/address execution count (breakpoints involved)

What to profile?

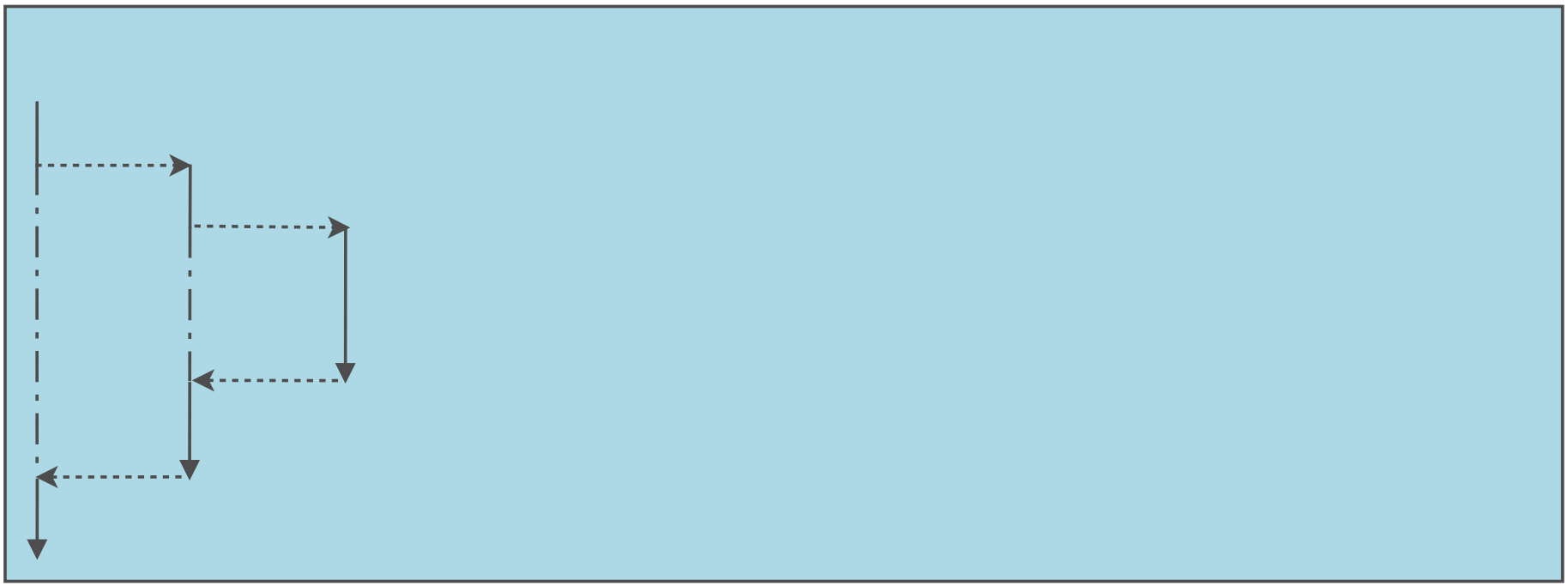
- During the normal execution
- Outside of the normal execution

What is measured?

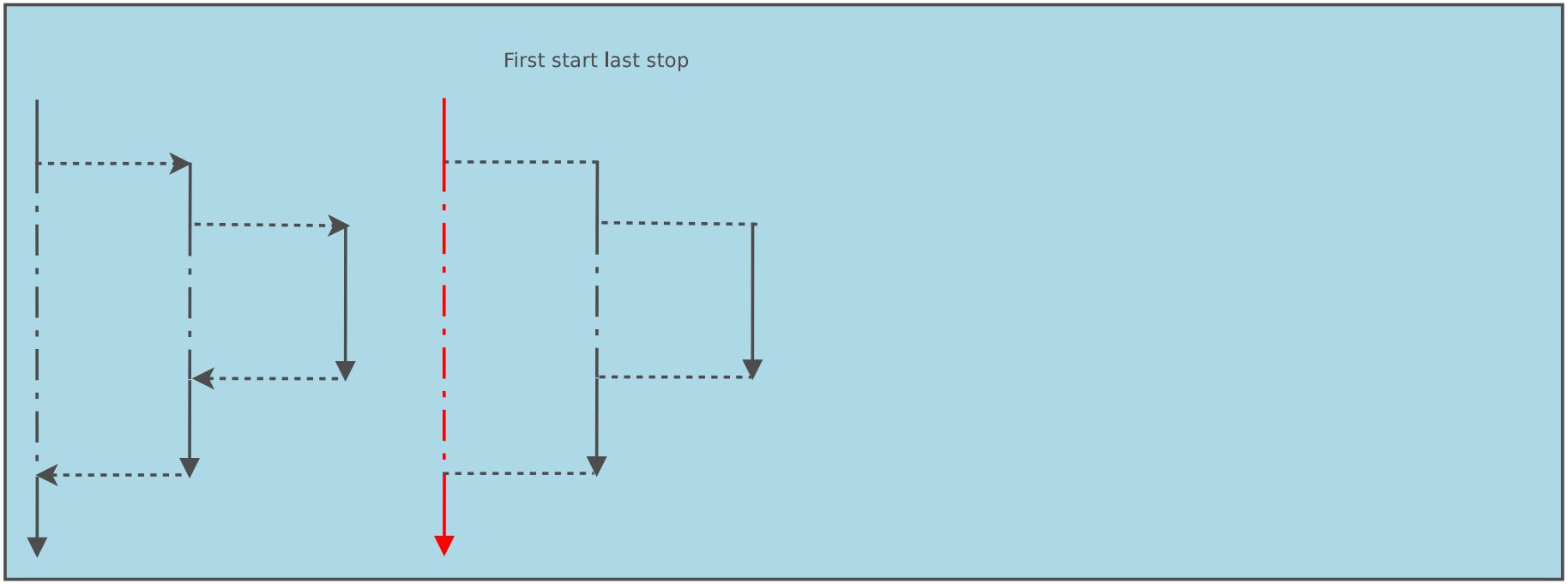
- `/proc/PID/...` values (*mem usage*, *context switches*, ...)
- `perf stat` counters
- `gprof` counters
- function/address execution count (breakpoints involved)

What to do with nesting?

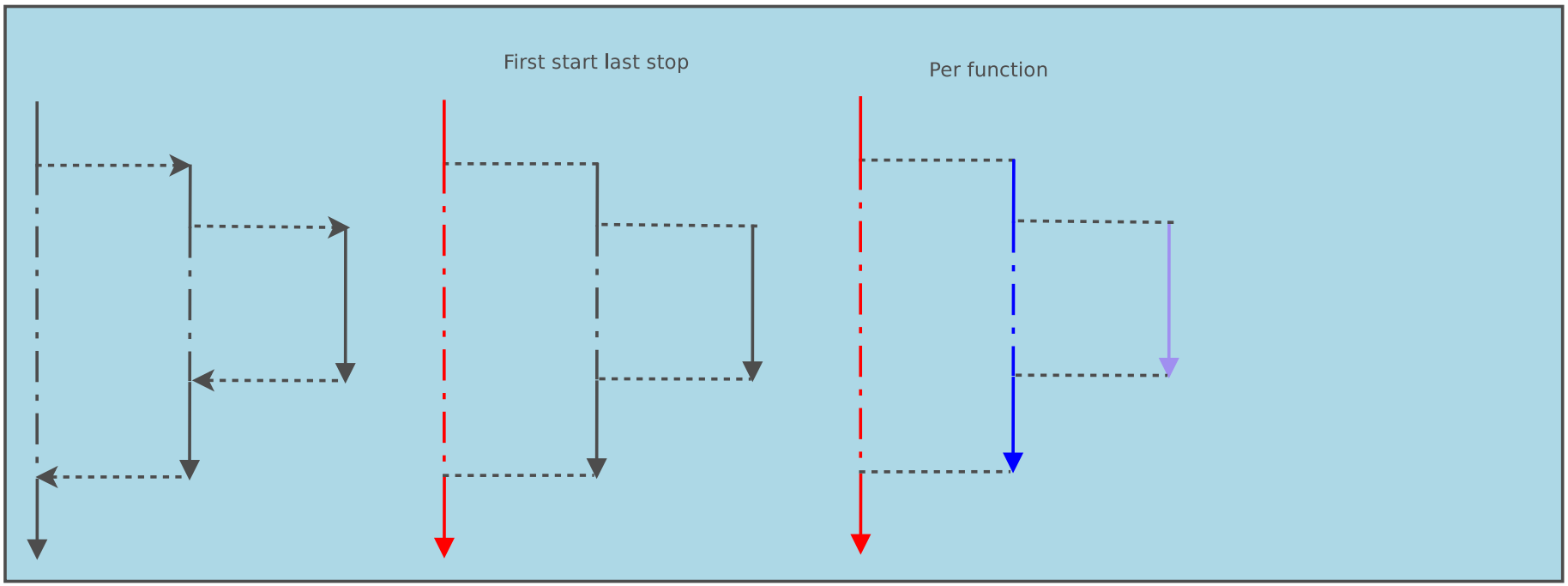
What to do with nesting?



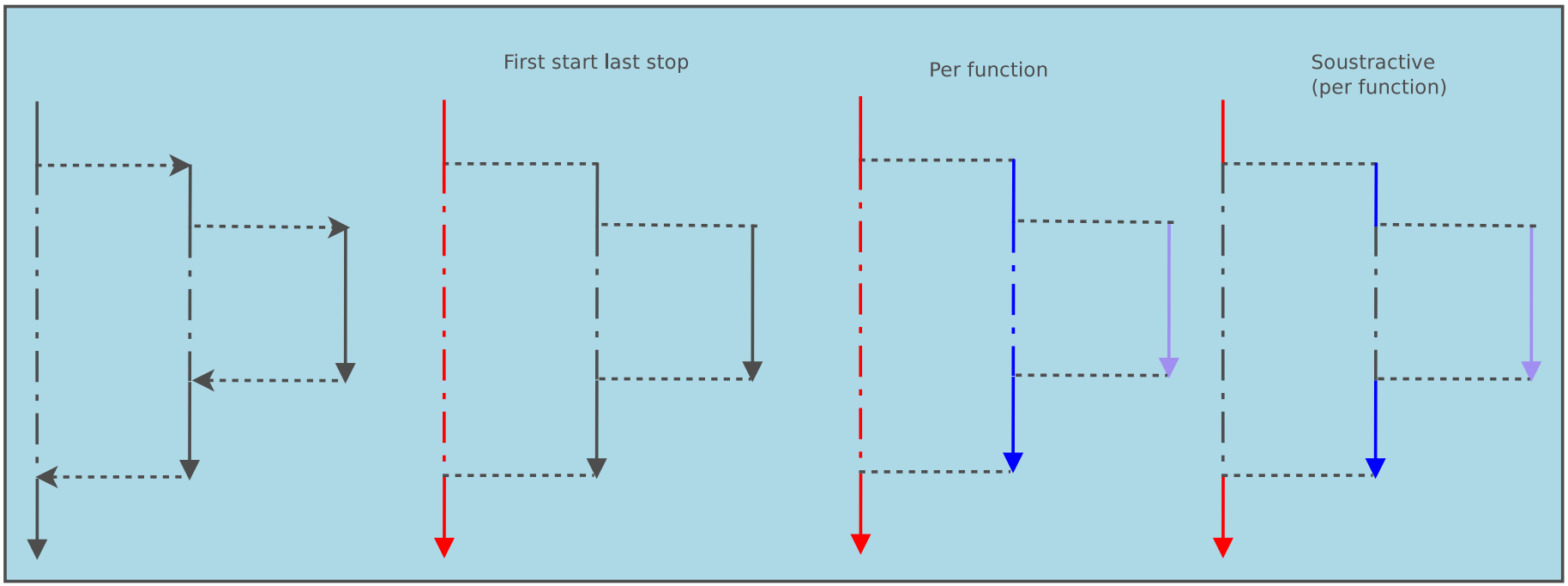
What to do with nesting?



What to do with nesting?

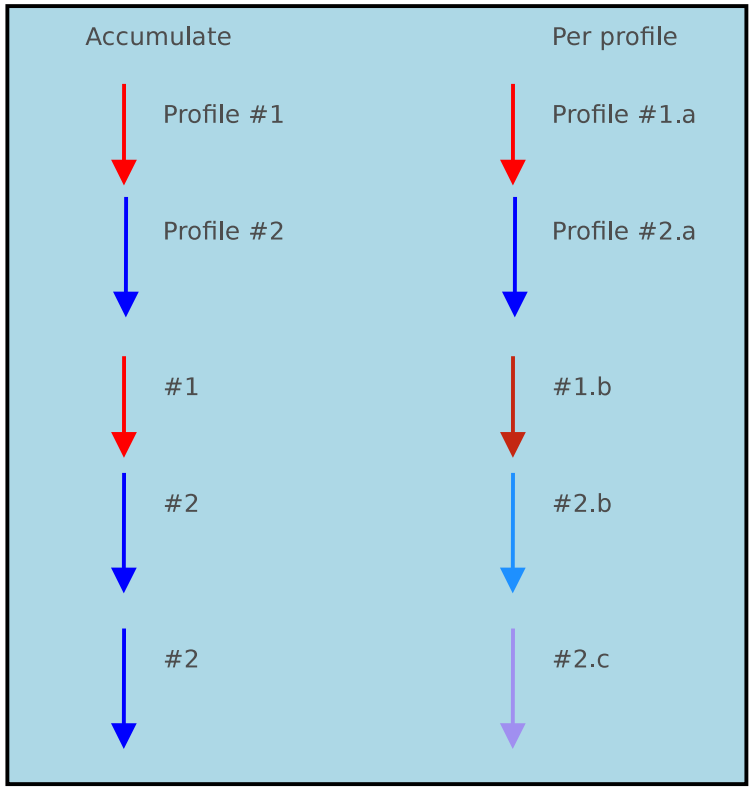


What to do with nesting?



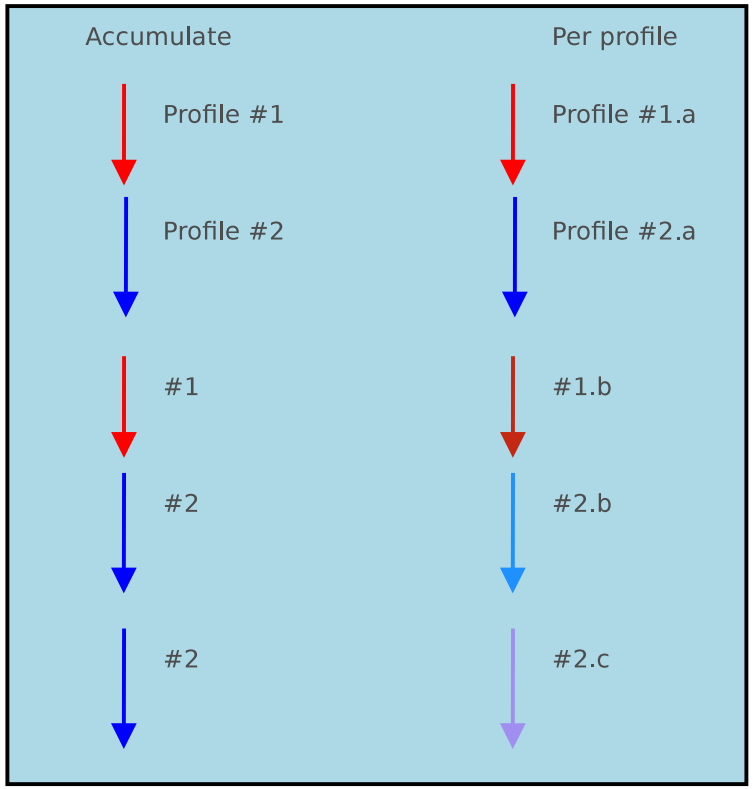
What to do with nesting?

... and multiple hits ?

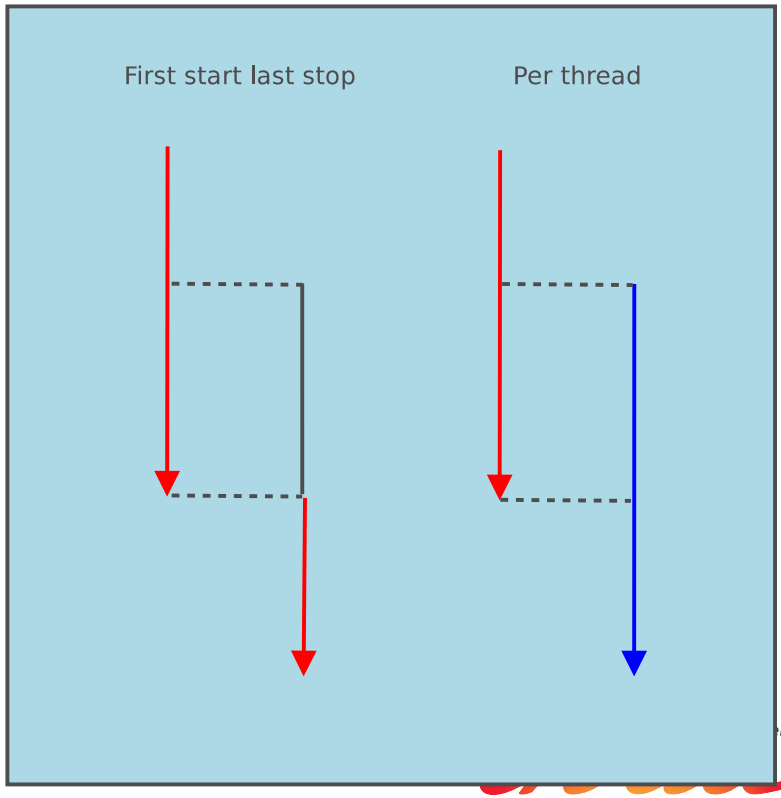


What to do with nesting?

... and multiple hits ?



... and concurrent hits ?



Profiling **outside** of the normal execution

- based on GDB/GCC dynamic compilation feature

Profiling **outside** of the normal execution

- based on GDB/GCC dynamic compilation feature

- `(gdb) profile interactive`
  - `-repeat:10`
  - `-code: fct(param1, param2, ...)`

$\Longrightarrow$ generates:

```
/* start profiling */
for(int i = 0; i < 10; i++) fct(param1, param2, ...);
/* stop profiling */
```

Profiling **outside** of the normal execution

- based on GDB/GCC dynamic compilation feature

- `(gdb) profile interactive`
  - `-repeat:10`
  - `-code: fct(param1, param2, ...)`
  - `-app`: runs the whole app, with same command-line args

Profiling **outside** of the normal execution

- based on GDB/GCC dynamic compilation feature

- `(gdb) profile interactive`
  - `-repeat:10`
  - `-code: fct(param1, param2, ...)`
  - `-app`: runs the whole app, with same command-line args
  - `-flags -O1,-O2,-O3`: compiles and benchmarks with different cflags

Profiling **outside** of the normal execution

- based on GDB/GCC dynamic compilation feature

- `(gdb) profile interactive`
  - `-repeat:10`
  - `-code: fct(param1, param2, ...)`
  - `-app`: runs the whole app, with same command-line args
  - `-flags -O1,-O2,-O3`: compiles and benchmarks with different cflags
  - `-file src.c`: specifies where to find the code to run

Profiling **outside** of the normal execution

- based on GDB/GCC dynamic compilation feature

- `(gdb) profile interactive`
  - `-repeat:10`
  - `-code: fct(param1, param2, ...)`
  - `-app`: runs the whole app, with same command-line args
  - `-flags -O1,-O2,-O3`: compiles and benchmarks with different cflags
  - `-file src.c`: specifies where to find the code to run

- "side-effect" free code only
  - at least no dependency on overwritten values

What to do with the measurements?

- raw display
- aggregation
- graph plot

What to do with the measurements?

- raw display
- aggregation
- graph plot

```
| function profile[compute_forces_crust_mantle_dev]
| ================================================
| min_flt:                                    13
| cycles:                            113,705,103
| branch-misses                          110,791
| task-clock:                             39.381
| branches:                           10,622,409
| instructions:                      188,404,850
| stalled-cycles-frontend:            50,032,346
| stalled-cycles-backend:             15,004,422
| ...
|
```

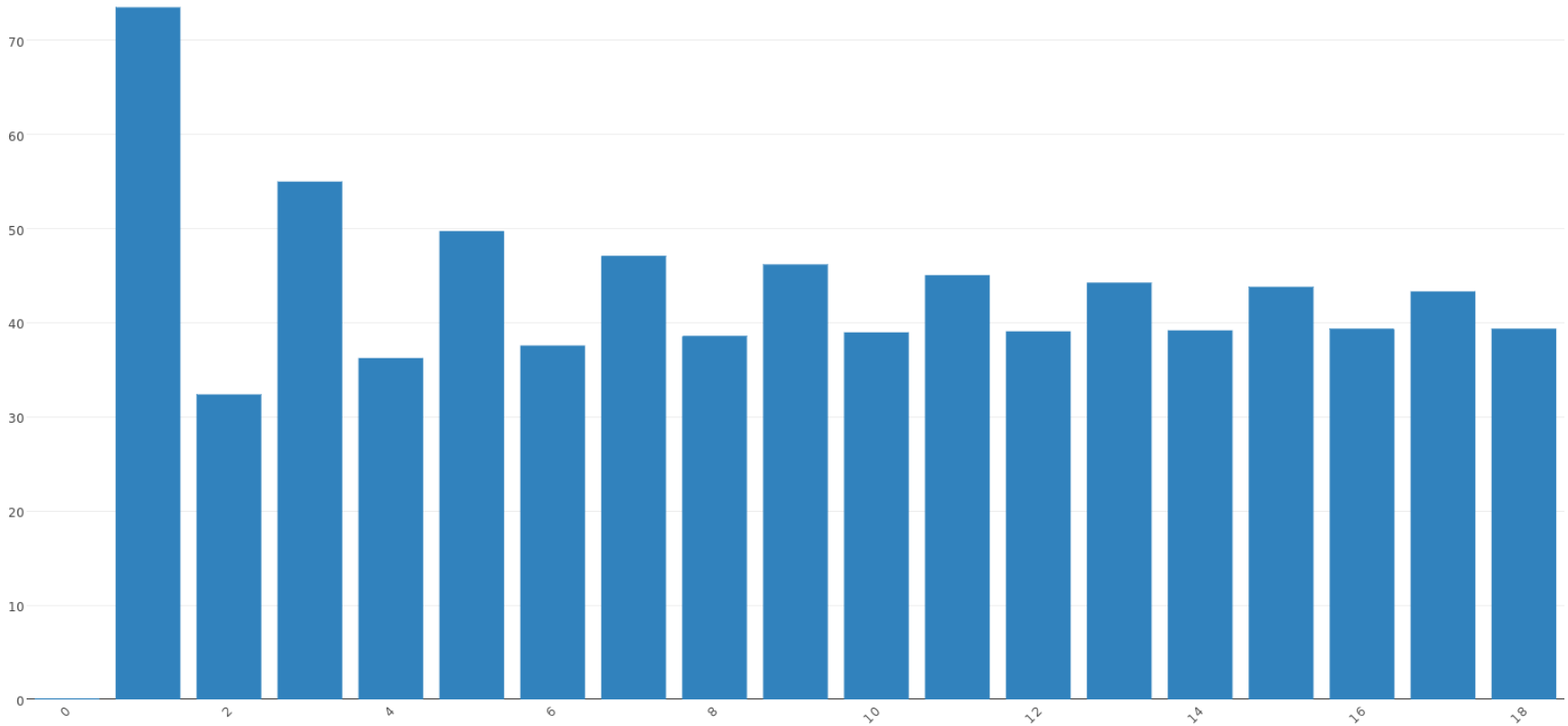
What to do with the measurements?

- raw display
- aggregation
- graph plot

```
| cycles                          avg:  11553677
|                                 min:     13549
|                                 max: 218766806
| -------------------------------
| stalled-cycles-backend          avg:  1517803
|                                 min:  1178455
|                                 max: 26184303
| -------------------------------
| instructions                    avg:  19149774
|                                 min:     18745
|                                 max: 367804777
| ...
```

What to do with the measurements?

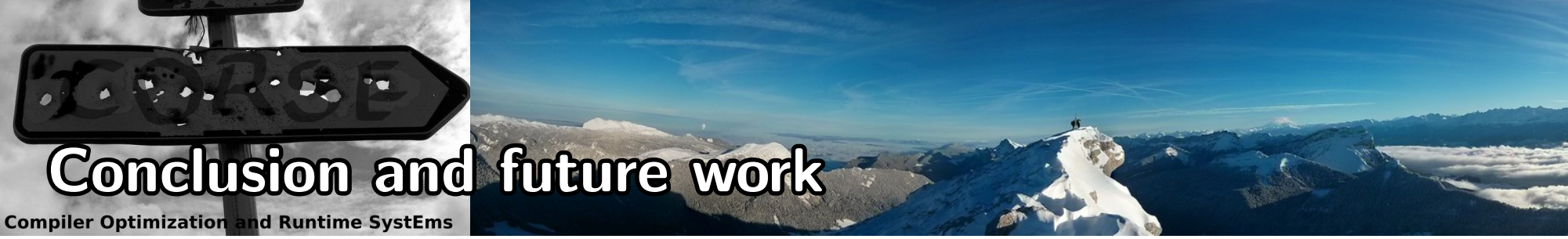
- raw display
- aggregation
- graph plot



Conclusion and future work

Compiler Optimization and Runtime SystEms

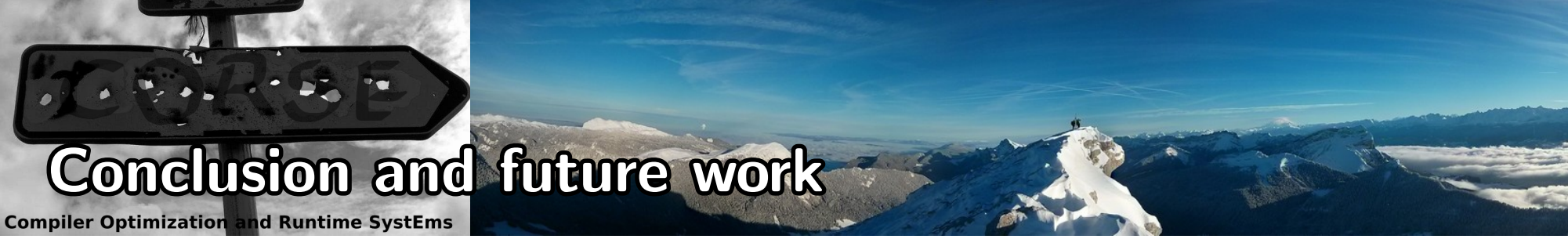
- Fine-grained control of profiling regions
- Ability to incorporate new counters and profiling tools



Conclusion and future work

Compiler Optimization and Runtime SystEms

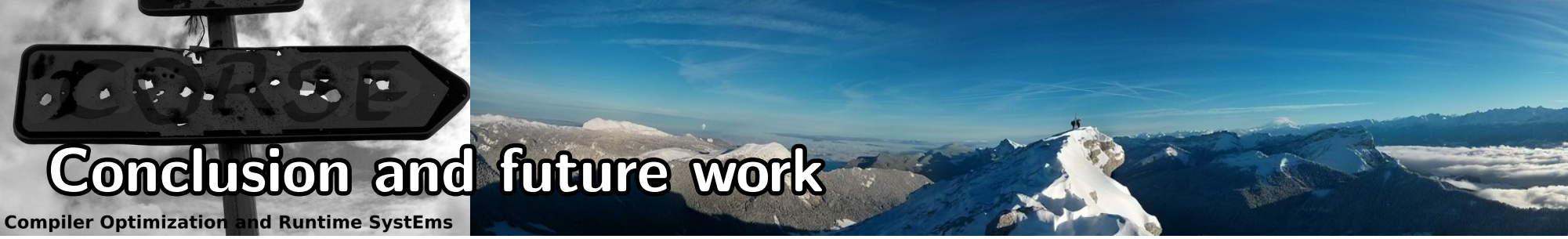
- Fine-grained control of profiling regions
- Ability to incorporate new counters and profiling tools
- Interactivity through GDB/GCC dynamic compilation



Conclusion and future work

- Fine-grained control of profiling regions
- Ability to incorporate new counters and profiling tools
- Interactivity through GDB/GCC dynamic compilation

- Compare our profiling results with other tools
- Profile real-world applications to demonstrate usability
  - Specfem3D kernels



Conclusion and future work

Compiler Optimization and Runtime SystEms

- Fine-grained control of profiling regions
- Ability to incorporate new counters and profiling tools
- Interactivity through GDB/GCC dynamic compilation

- Compare our profiling results with other tools
- Profile real-world applications to demonstrate usability
  - Specfem3D kernels

- Extend to multithreaded environments
- Integrate with mcGDB and its programming models (OpenMP)
  - focus on task/task-graph execution ?

Interactive Performance Profiling and Debugging

Kevin Pouget
Jean-François Méhaut, Miguel Santana

Université Grenoble Alpes / LIG, STMicroelectronics, France
Nano2017-DEMA project

NANO2017/DEMA Meeting, Grenoble
June 24th, 2016