Dynamic Tracing (uftrace & dyntrace)

Anas balboul Ahmad shahnejat

December, 6

OUTLINE

- INTRODUCTION
- PREVIOUS WORK
- DYNTRACE IMPROVEMENT
  - DEPENDENCY REMOVAL
  - CMAKE TO AUTOTOOLS
- DEMO
- FUTURE WORK

Introduction

- ## DynTrace:

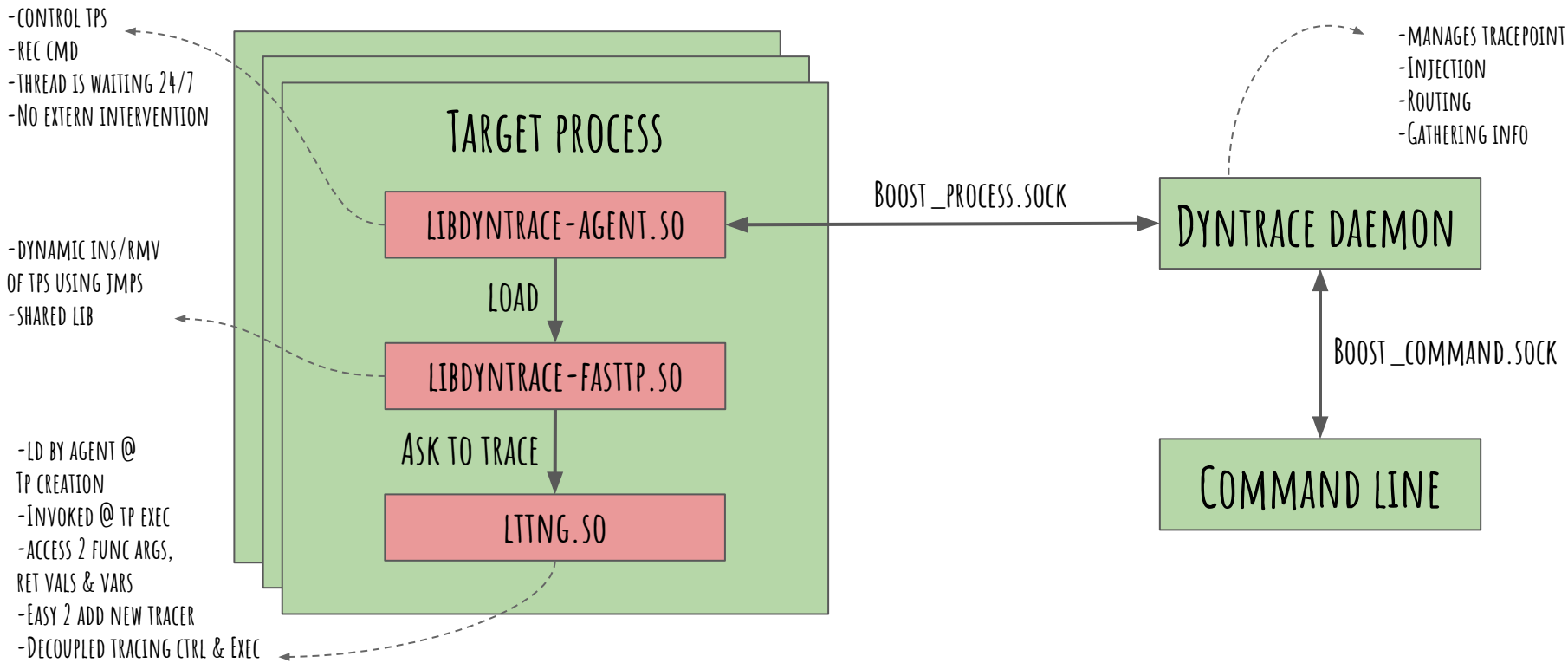
  -Userspace dynamic tracing tool
  -Implements a fast tracepoint insertion for x86(_64) ecosystem on Linux

- ## Boost_library:

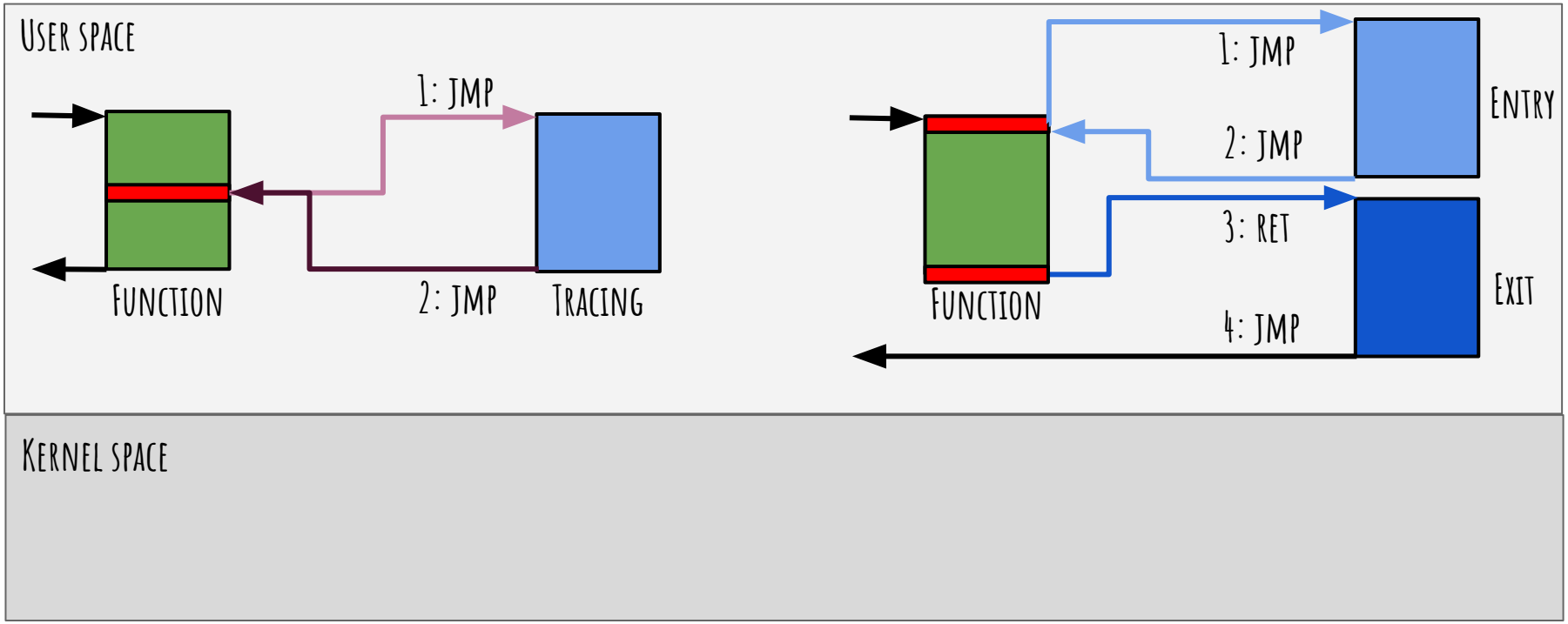
  -A C++ library which contains over
  80 individual libraries.

  -Provides support for multithreading,
  image processing, regular expressions, unit testing...

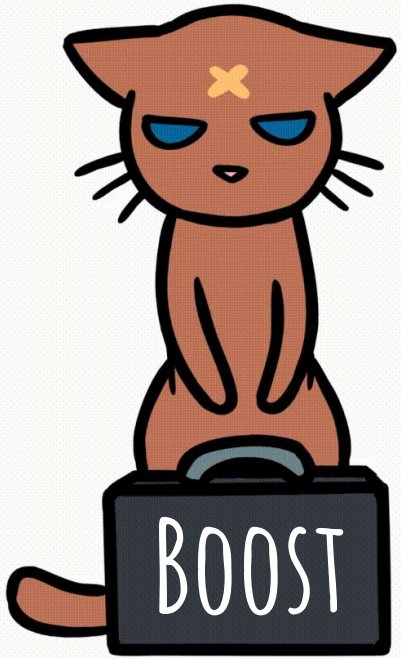
DYNTRACE


-CONTROL TPS
-REC CMD
-THREAD IS WAITING 24/7
-NO EXTERN INTERVENTION
TARGET PROCESS
LIBDYNTRACE-AGENT.SO
LOAD
LIBDYNTRACE-FASTTP.SO
ASK TO TRACE
LTTNG.SO
-DYNAMIC INS/RMV OF TPS USING JMPS
-SHARED LIB
-LD BY AGENT @ TP CREATION
-INVOKED @ TP EXEC
-ACCESS 2 FUNC ARGS, RET VALS & VARS
-EASY 2 ADD NEW TRACER
-DECOUPLED TRACING CTRL & EXEC
BOOST_PROCESS.SOCK
DYNTRACE DAEMON
-MANAGES TRACEPOINT
-INJECTION
-ROUTING
-GATHERING INFO
BOOST_COMMAND.SOCK
COMMAND LINE

DYNTRACE

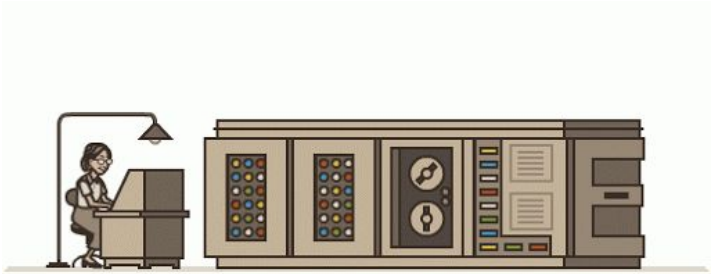

User space
1: JMP
Function
2: JMP
Tracing
1: JMP
2: JMP
Entry
3: RET
Function
Exit
4: JMP
Kernel space

BOOST _LIBRARY

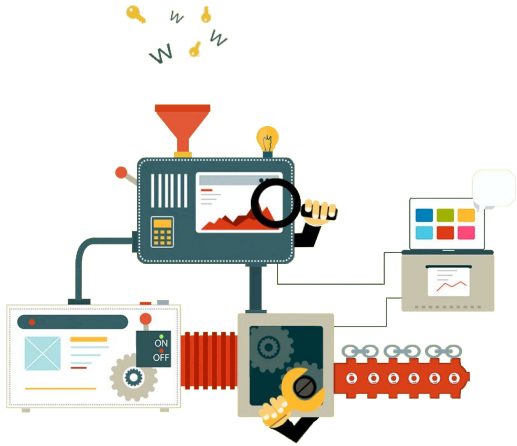


Dyntrace Clean up

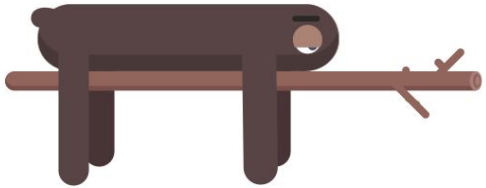
- Boost uses lots of templates!
- Embedded developers && modern subset of c++!
- The advanced C++ techniques && old platforms!
- General purpose library && specific needs!
- Compile time && boost #includes!



Dyntrace Clean up



- It is usually not worth transitioning away from the in-house library of functionality! It would be a major porting effort that would destabilize a lot of code.
- Complexity && readability!
- Slow performance!
- Cmake --> Autotools!



BOOST

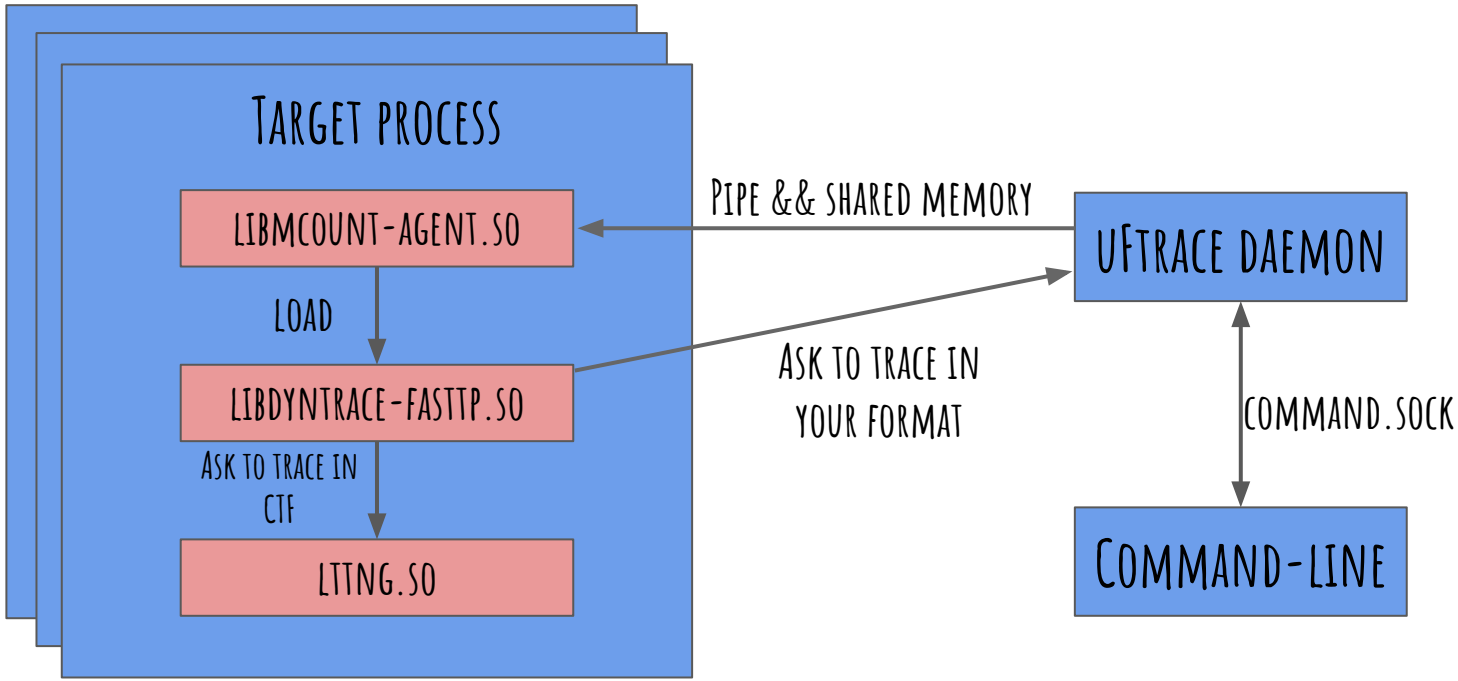
Linux call

Demo

DEMO

Future work

uftrace integration



Insertion performance

Tool	Time(µs)
Dyntrace	28,6
DynInst	9’434,0
Uprobe (in kernel 4.16.14)	25,0

Execution performance(1-thread)

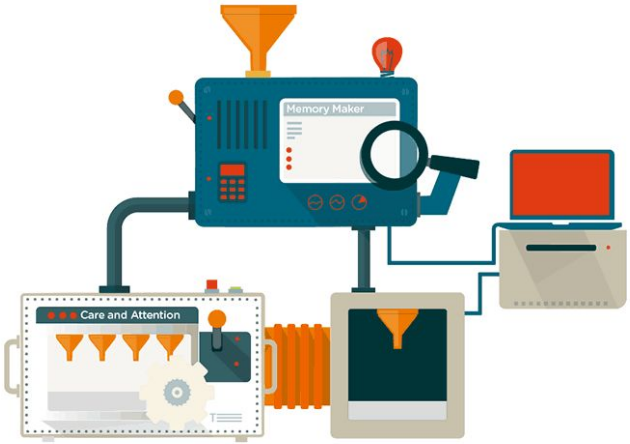
Tool	Execution time(ns)
Dyntrace “Point”	178
Dyntrace “Entry-Exit”	360
Uprobe “Point”	1’933
Uprobe “Entry-Exit”	2’650

Outline

- Introduction
  - fasttp library
  - uftrace
- Contribution
- Future work

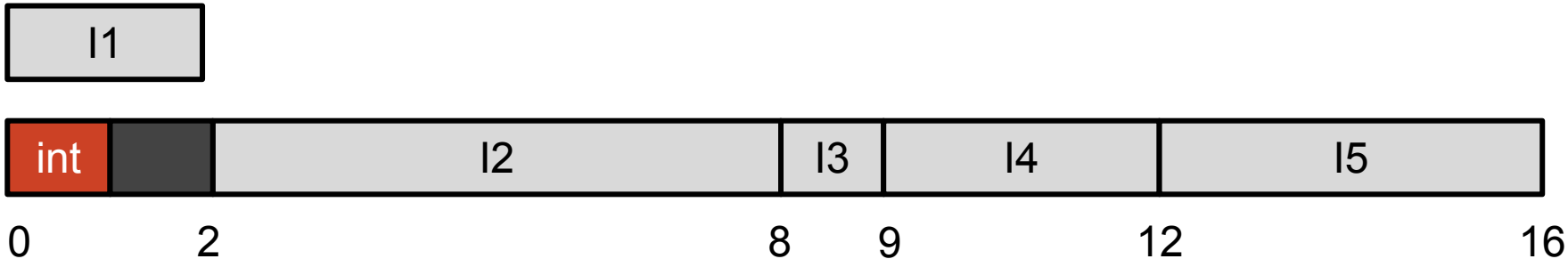
UFTRACE

- Userspace tracing tool.
- Mainly static tracepoints. (-pg, recompiling
- Supports dynamic tracing. (-mnop-mcount)
- Cannot attach and trace a target process. (forks and exec)



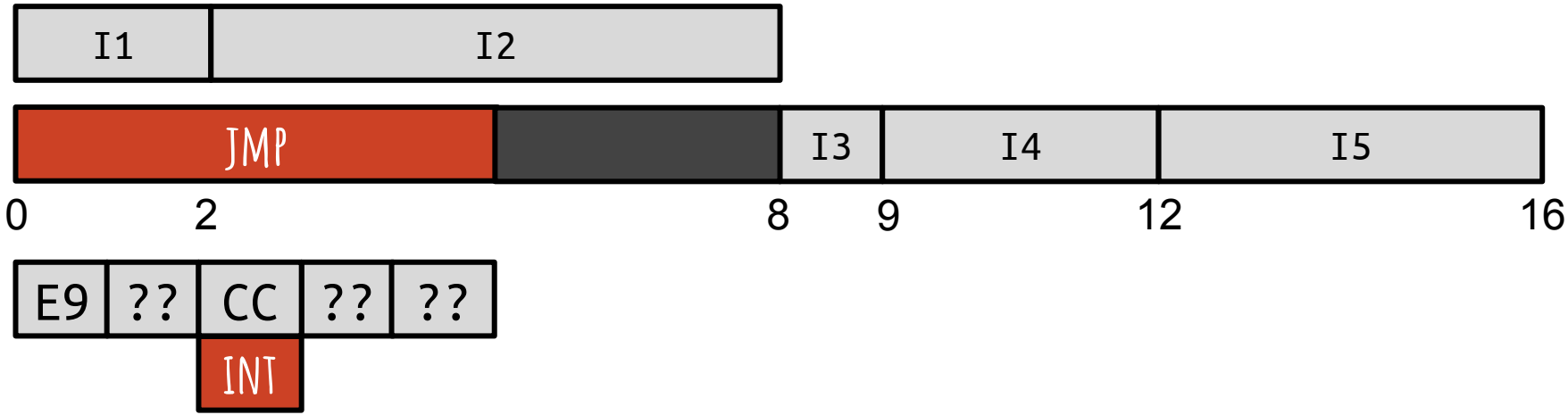
FastTP(Fast_Tracepoint)

- Dynamic tracepoint. (run time insertion, no recompilation needed)
- Limited number of inserted tracepoints.
- Uses a combination of jump and trap instructions.



FASTTP(FAST_TRACEPOINT)

- Faster than the trap based trace point.



Contribution(FASTtp)

- Production-ready and stability.
- Support for special cases:
  - inusual instructions.
  - Tracepoints in recursive and nested cases.
- Made the caller function’s address available in the tracepoint handler.

Contribution(uftrace)

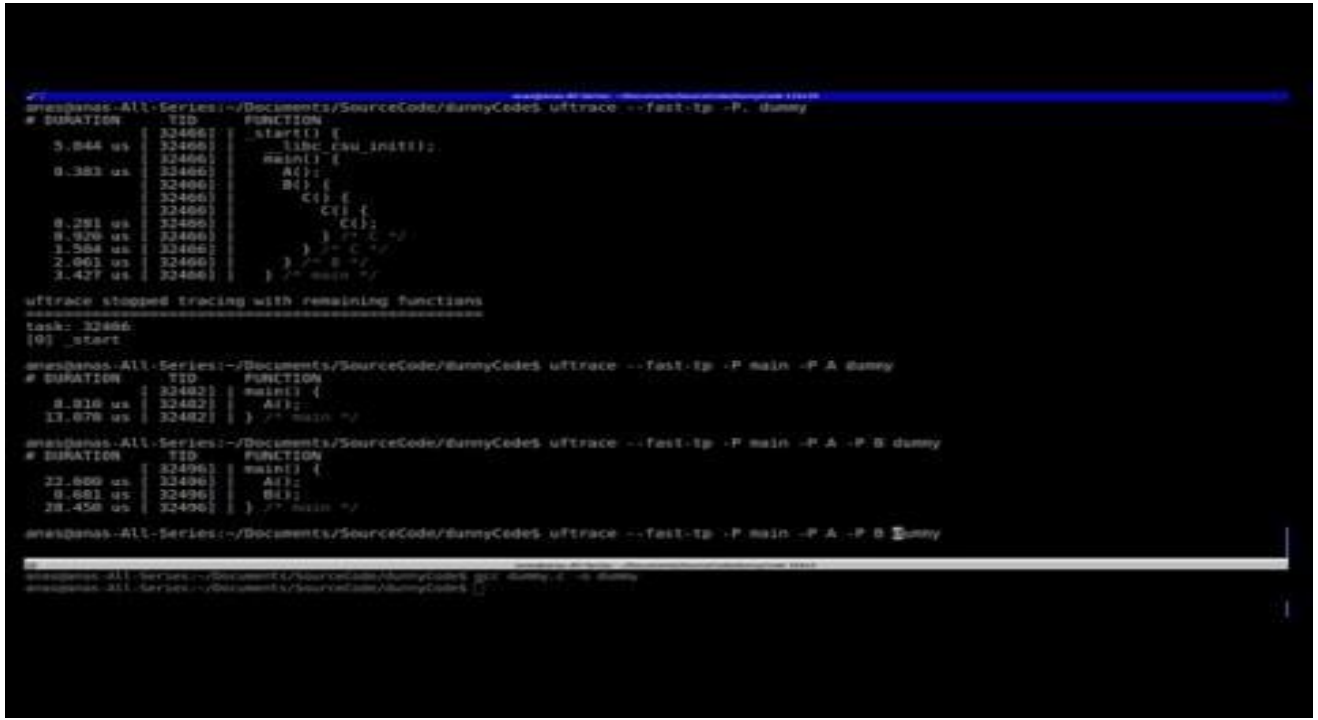
- Integrated and adapted fasttp to uftrace (most the features are integrated too).
- Uftrace can insert fasttp tracepoints (no need for re…) .
- No compiler/linker flags needed anymore.

Future WORK

- fasttp
  - powerPC or ARM64 support.
  - surpass number of tracepoints limitation.
  - more stability.
- Uftrace
  - attach and trace a running process.
  - output in CTF trough LTTng.

Demo

DEMO


anas@anas-All-Series:~/Documents/SourceCode/dummyCode$ uftrace --fast-tp -P. dummy
# DURATION TID FUNCTION
[32466] | _start() {
5.844 us [32466] | __libc_csu_init();
[32466] | main() {
0.383 us [32466] | A();
[32466] | B() {
[32466] | C() {
[32466] | C() {
0.281 us [32466] | C();
0.920 us [32466] | } /* C */
1.584 us [32466] | } /* C */
2.061 us [32466] | } /* B */
3.427 us [32466] | } /* main */
uftrace stopped tracing with remaining functions
task: 32466
[0] _start
anas@anas-All-Series:~/Documents/SourceCode/dummyCode$ uftrace --fast-tp -P main -P A dummy
# DURATION TID FUNCTION
[32482] | main() {
8.810 us [32482] | A();
13.070 us [32482] | } /* main */
anas@anas-All-Series:~/Documents/SourceCode/dummyCode$ uftrace --fast-tp -P main -P A -P B dummy
# DURATION TID FUNCTION
[32496] | main() {
22.600 us [32496] | A();
8.681 us [32496] | B();
28.450 us [32496] | } /* main */
anas@anas-All-Series:~/Documents/SourceCode/dummyCode$ uftrace --fast-tp -P main -P A -P B dummy

DEMO


DURATION
TID
FUNCTION
ngx_worker_process_cycle() {
ngx_worker_process_init() {
ngx_set_environment() {
ngx_array_push();
ngx_pool_cleanup_add() {
ngx_palloc();
ngx_alloc();
ngx_get_cpu_affinity();
ngx_event_process_init() {
ngx_event_timer_init();
ngx_epoll_init() {
ngx_alloc();
ngx_alloc();
ngx_alloc();
ngx_get_connection();
ngx_epoll_add_event();
ngx_http_upstream_init_worker();
ngx_add_channel_event() {
ngx_get_connection();
ngx_epoll_add_event();
ngx_setproctitle() {
ngx_sprintf();
ngx_sprintf();
ngx_process_events_and_timers() {