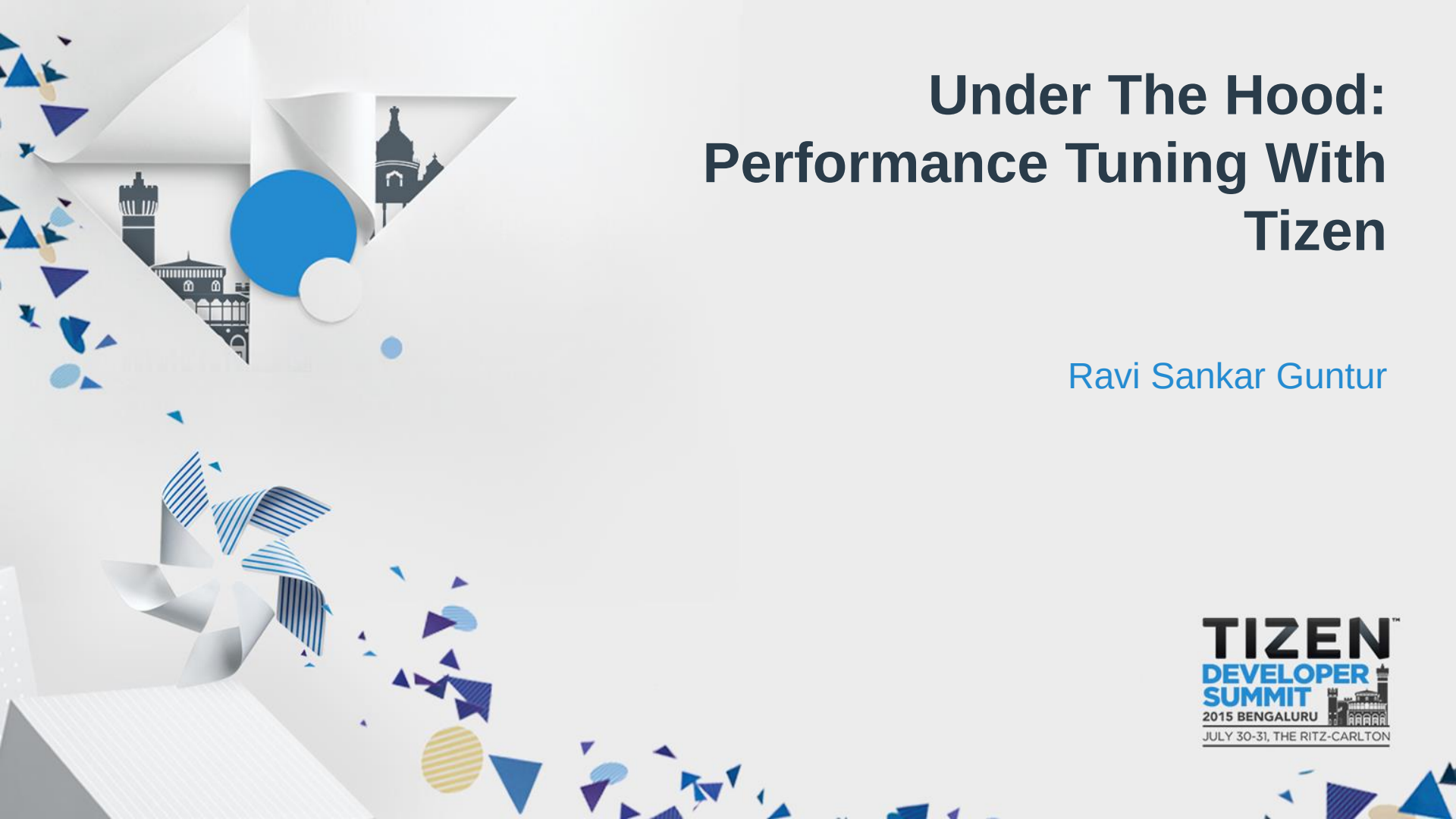
Under The Hood: Performance Tuning With Tizen

Ravi Sankar Guntur



- **How to write a Tizen App**
- **Tools already available in IDE v2.3**
  - Dynamic Analyzer
  - Valgrind

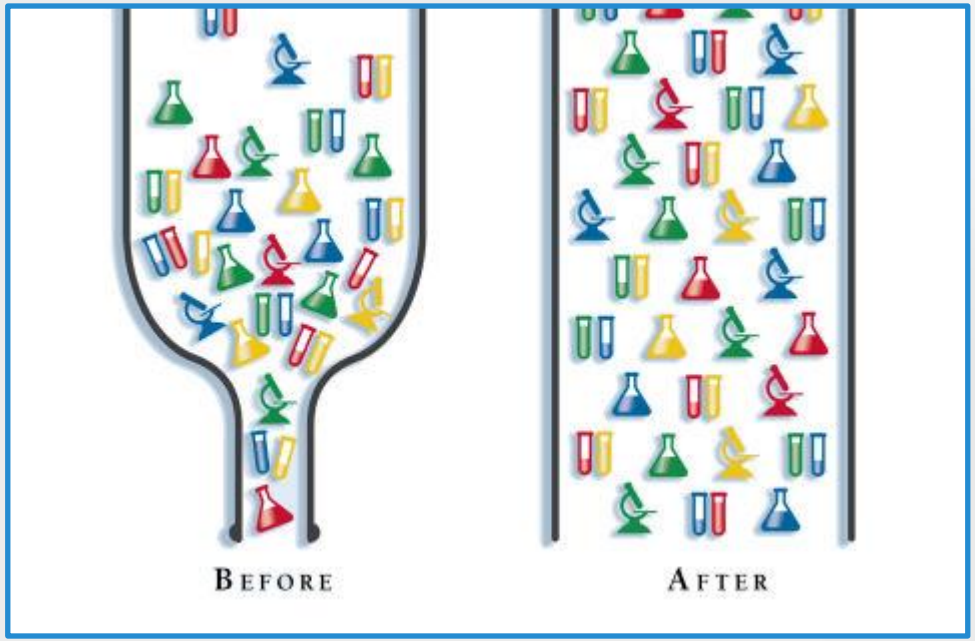
What’s NEXT?

- **Want to optimize my application**
  - App stands out among crowd
  - Better rating!
- **What's in Tizen 2.4**
  - Platform
  - Tools

What would you get from this talk?

- How to profile & analyze data
- Under the hood optimizations

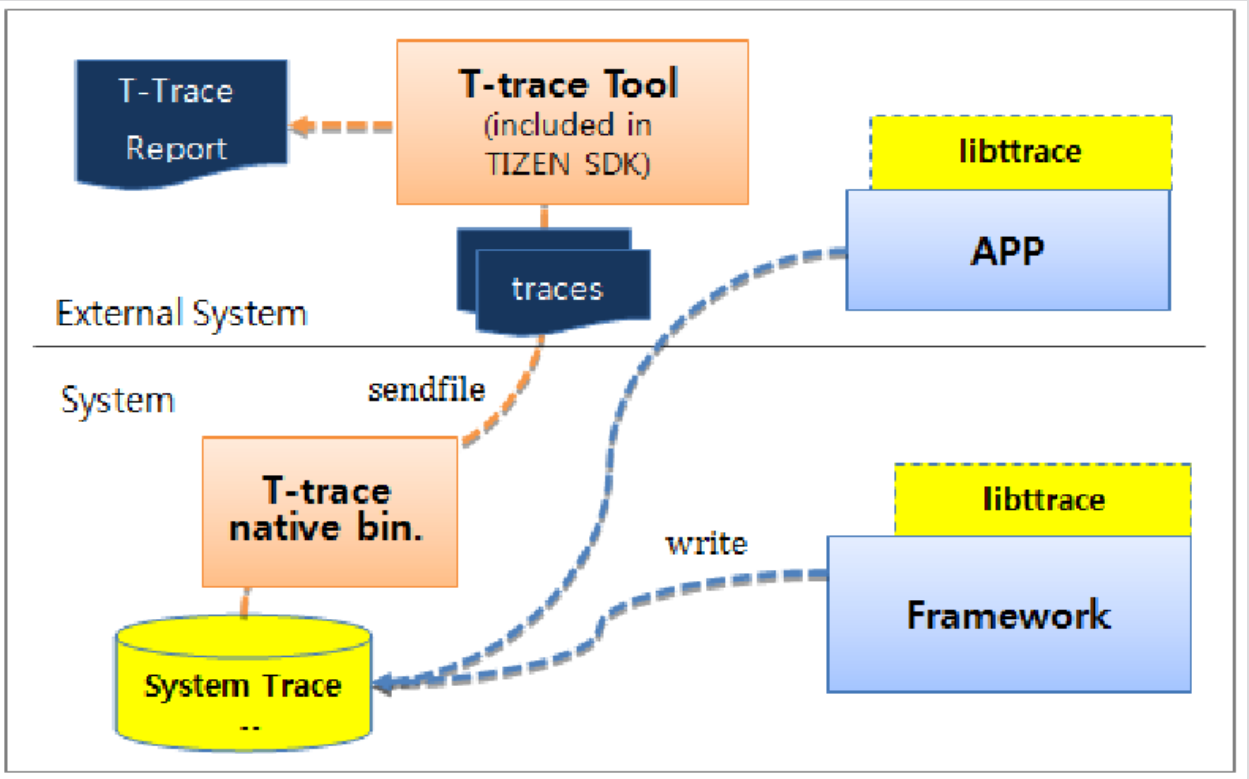
That's Talk Outline



Profile Your App

Premature optimization is the root of all evil - D.Knuth

Tizen Trace



Host Requirements:

- *Python 2.7.x*
- *Google Chrome*

Inserting App Trace Points

```c
int main(void)
{
    int integer = 12;
    trace_begin("event name: %d", integer);

    trace_end();

    return 0;
}
```

```c
void function1()
{
    int cookies_f1 = 123;

    trace_async_begin(cookies, "event name");
}

void function2()
{
    int cookies_f2 = 123;

    trace_async_end(cookies_f2, "event name");

}
```

```c
void function2(int count)
{
    trace_update_counter(count, "event_name");
}
```

Demo

Trace Live Demo

1. Application Launch Time
2. App Freeze Scenario
3. Frame Per Second

Quick Recap – Tizen Trace

- Gives system wide summary in timeline
- Can select trace tags
- Can insert app specific trace points
- Very little overhead
- Trace is analyzed using Chrome

Under The Hood:

Virtue of being ‘OS Of Everything’

Benchmark Devices

	Z1	Other 1	Other 2	Other 3
OS	Tizen 2.3	Android 4.2.2	Android 4.2.2	Android One
CPU	1.2GHz X 2 ARM Cortex-A7	1.2GHz X 4 ARM Cortex-A7	1.3GHz X 2 ARM Cortex-A7	1.3GHz X 4 ARM Cortex-A7
GPU	Mali 400MP	Mali 400MP	Mali 400MP	Mali 400MP
RAM	768MB	768MB	512MB	1024MB
Storage	4GB	4GB	4GB	4GB
LCD	4.0" TFT WVGA (480X800)	4.5" TFT LCD WVGA (480X800)	4.0" TFT WVGA (480X800)	4.5" IPS FWVGA (480X854)
Battery	1,500mAh	2,000mAh	1,500mAh	1,700mAh
Camera	3MP, VGA	5MP, VGA	2MP, VGA	5MP, 2MP

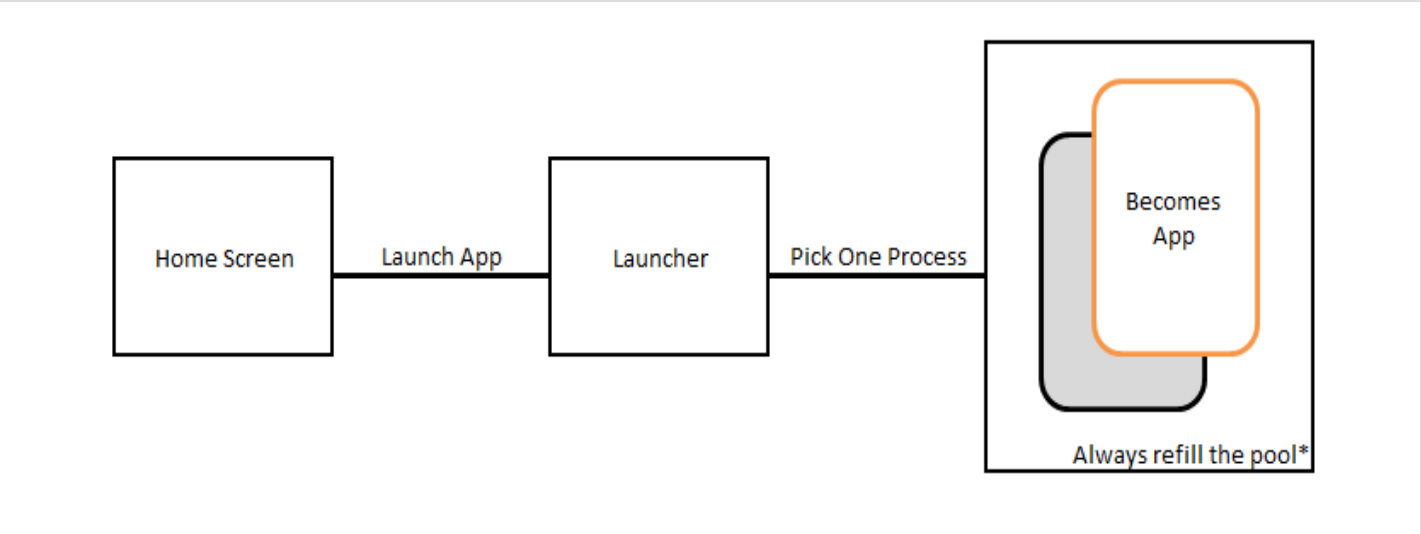
Faster Boot Time

- Effective Multi Core Utilization
- Better Service Dependency Handling
- On Demand Launch of Services

Boot Time (sec)	Z1	Other1	Other 2
Normal	17.87	24.54	26.69

Faster App Launch Times

- Library preload
- Process pool



Benchmark Results – App Launch Times

App Open Times (sec)	Z1	Other 1	Other 2	Other 3
Calculator	0.80	0.72	0.68	0.65
Camera	1.55	2.44	2.11	2.08
Clock	0.99	0.96	1.17	0.90
Contacts	1.39	1.73	0.78	2.71
Phone	1.02	1.72	1.56	3.61
Gallery	1.08	1.08	1.82	2.00
Message	1.22	1.26	0.81	1.25
Music	1.02	1.99	1.75	1.38
Calendar	1.19	0.86	0.8	0.70
Setting	0.98	0.93	0.89	0.83
Average	**1.11**	**1.32**	**1.22**	**1.55**

Smaller And Faster Updates

Download what is changed.

- Saves download time
- Saves data costs

Tizen Package	Version X Size	Version X+1 Size	Delta Size
quiztime	215.5 KB	**218.3 KB**	**19.8 KB**
speedmeter	52.6 KB	**61.6 KB**	**21.7 KB**
applocker	976.2 KB	**1.1 MB**	**131.4 KB**

Recap

Quick Summary

- **Profile Your App**
  - Dynamic Analyzer
  - Valgrind
  - Tizen Trace

Quick Summary

- **Under The Hood**
  - Faster boot
  - Faster open times
  - Delta Upgrades

Conclusion

- Optimization is tough, but with insightful tools and best programming practices we can improve our applications.

! - ?

Reference

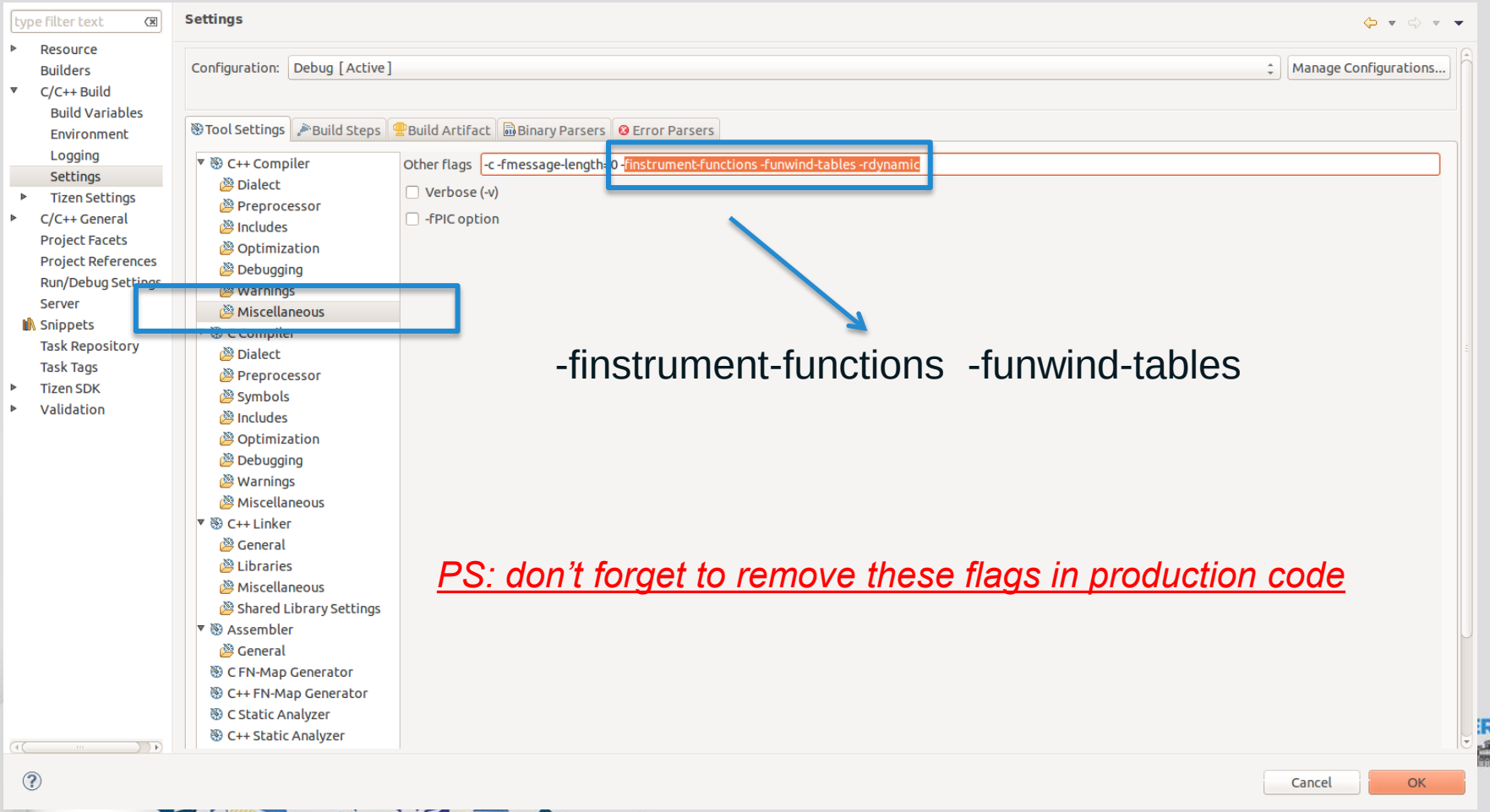
- https://developer.tizen.org/development/dev-guide/2.3.0
- https://docs.enlightenment.org/auto/efl/
- http://valgrind.org/

Thank you


TIZEN
DEVELOPER
SUMMIT
2015 BENGALURU
JULY 30-31, THE RITZ-CARLTON

DIY: Tool Kit

Instrument Code



Instrument Code

```
void
__attribute__ ((constructor))
__attribute__((no_instrument_function))
trace_begin (void)
{
    char trace_path[1024];
    pid_t pid = getpid();

    snprintf(trace_path, 1023, "/opt/usr/media/Others/%s.%d", "trace.out", pid);

 fp_trace = fopen(trace_path, "w");
 dlog_print(DLOG_DEBUG, LOG_TAG, "Trace FILE %s", trace_path);
}

void
__attribute__((no_instrument_function))
__attribute__ ((destructor))
trace_end (void)
{
 if(fp_trace != NULL) {
 fclose(fp_trace);
 }
}
```

Instrument Code

```
void
__attribute__((no_instrument_function))
__cyg_profile_func_enter (void *func,  void *caller)
{
 struct timespec tp;
    void *buffer[2];
    char **strings;
    int nptrs;

 if(fp_trace != NULL)
 {
     nptrs = backtrace(buffer, 2);
     if(nptrs)
     {
         strings = backtrace_symbols(buffer, nptrs);
         clock_gettime(CLOCK_BOOTTIME, &tp);

         fprintf(fp_trace, "%lu E %s %lu.%lu\n", pthread_self(), strings[1], tp.tv_sec, tp.tv_nsec/1000000);
        free(strings);
     }
     else
     {
         clock_gettime(CLOCK_BOOTTIME, &tp);
         fprintf(fp_trace, "%lu E %p %p %lu.%lu\n", pthread_self(), func, caller,  tp.tv_sec, tp.tv_nsec/1000000);
     }
 }
}
```

Instrument Code

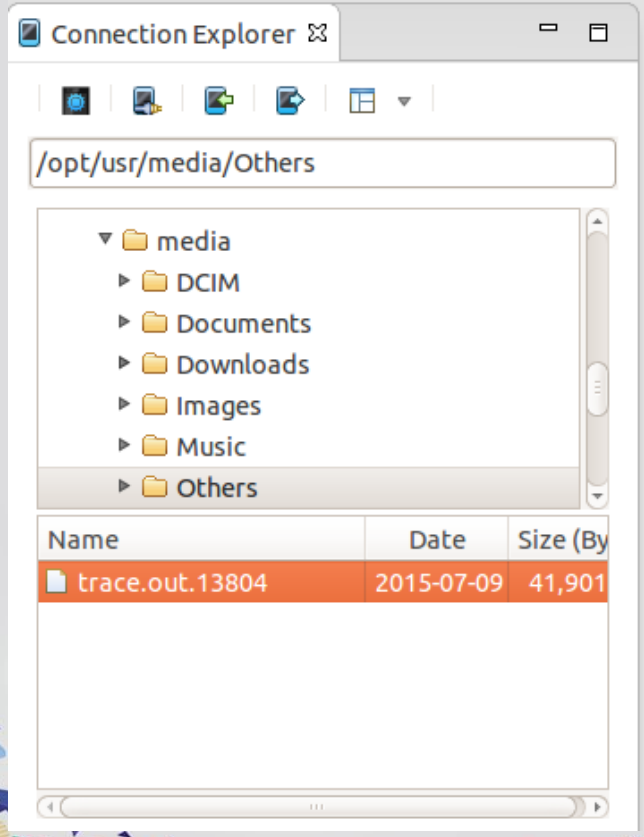
```
void
__attribute__((no_instrument_function))
__cyg_profile_func_exit (void *func, void *caller)
{
    struct timespec tp;
    void *buffer[2];
    char **strings;
    int nptrs;

 if(fp_trace != NULL)
 {
     clock_gettime(CLOCK_BOOTTIME, &tp);

     nptrs = backtrace(buffer, 2);
     if(nptrs)
     {
         strings = backtrace_symbols(buffer, nptrs);
         fprintf(fp_trace, "%lu X %s %lu.%lu\n", pthread_self(), strings[1], tp.tv_sec, tp.tv_nsec/1000000);
         free(strings);

     }
     else
         fprintf(fp_trace, "%lu X %p %p %lu.%lu\n", pthread_self(), func, caller,  tp.tv_sec, tp.tv_nsec/1000000);
 }
}
```

Instrument Code



trace.out.<pid> contains

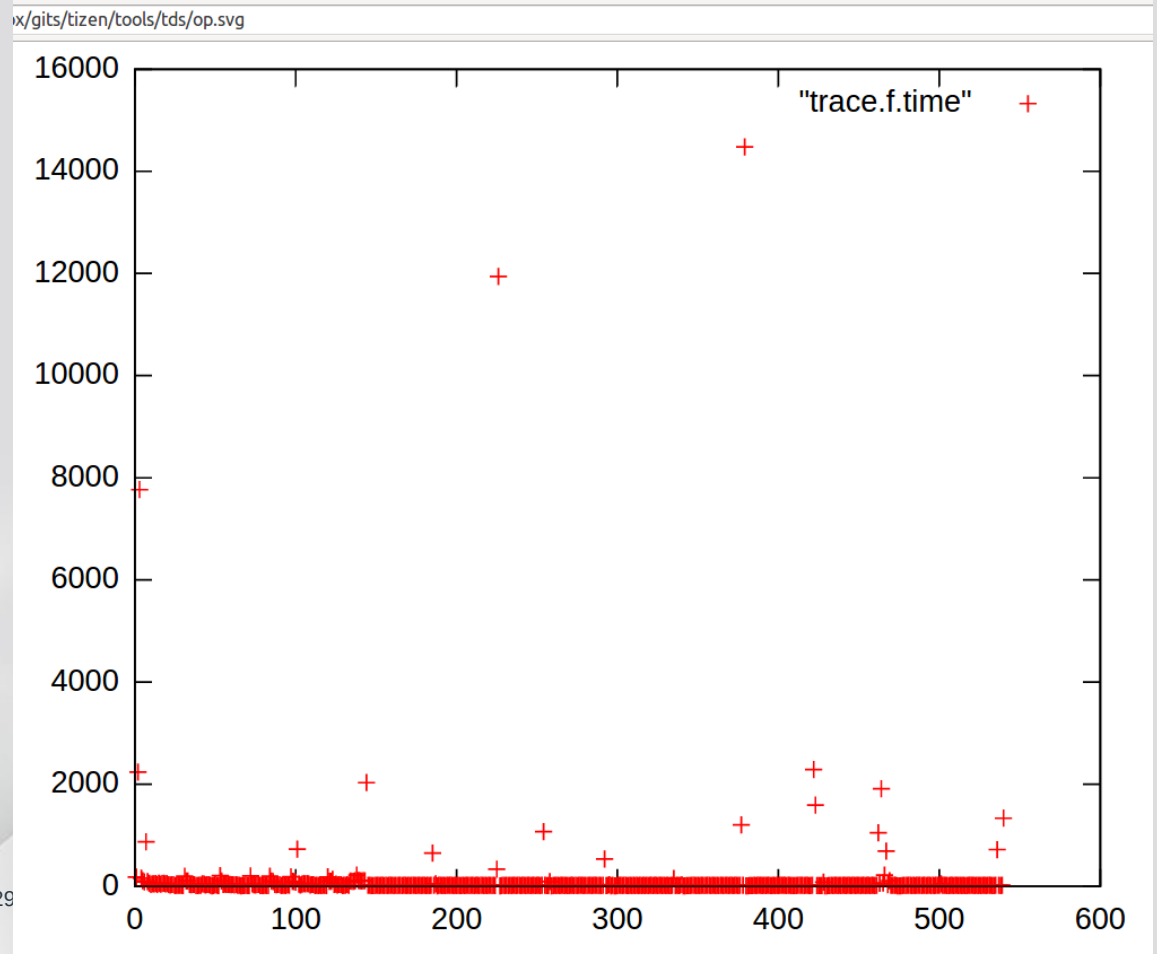
- function call graph with timing info
- fps data

- Use provided scripts to analyze the trace file
- *<TBD> share demo app and scripts code*

Function Call Graph

```
ravi@neo:~/sbox/gits/tizen/tools/tds$ ./convert.sh ~/workspace/Demo/Debug/demo  ~/Desktop/trace.out.13804
Filtering function profile and frame rate data
done
Parse function profile data
********************** trace.t.2939175392 ******************
 pthread_start   7505.873
 --->  compute_prime   7505.875
 --->  compute_prime   7518.3  took (12.425) s.ms
 pthread_start   7518.5  took (12.627) s.ms
********************** trace.t.2939175392 ******************
********************** trace.t.2947564000 ******************
 ethread_start   7505.9
 -->  compute_prime   7505.20
 -->  compute_prime   7525.126  took (19.926) s.ms
 thread_start   7525.127  took (19.227) s.ms
********************** trace.t.2947564000 ******************
********************** trace.t.3039907840 ******************
 main   7466.438
 --->  sensor_check   7466.439
 --->  sensor_check   7466.441  took (.002) s.ms
 --->  camera_init   7466.442
 --->  camera_init   7466.504  took (.062) s.ms
 --->  app_create   7466.531
 ---> --->  create_base_gui   7466.531
 ---> ---> --->  create_bg   7466.627
 ---> ---> --->  create_bg   7466.628  took (.001) s.ms
 ---> ---> --->  create_conform   7466.628
 ---> ---> --->  create_conform   7466.638  took (.010) s.ms
 ---> ---> --->  create_view   7466.650
 ---> ---> ---> --->  create_entry   7466.651
 ---> ---> ---> --->  create_entry   7466.749  took (.098) s.ms
 ---> ---> ---> --->  create_panel   7466.750
 ---> ---> ---> ---> --->  create_panel_basic_content   7466.760
 ---> ---> ---> ---> --->  create_panel_basic_content   7466.828  took (.068) s.ms
 ---> ---> ---> ---> --->  create_panel_advanced_content   7466.829
 ---> ---> ---> ---> --->  create_panel_advanced_content   7466.889  took (.060) s.ms
 ---> ---> ---> --->  create_panel   7466.890  took (.140) s.ms
 ---> ---> --->  create_view   7466.930  took (.370) s.ms
 ---> ---> --->  create_base_gui   7467.43  took (.899) s.ms
 --->  app_create   7467.43  took (.899) s.ms
 --->  app_control   7467.46
 --->  app_control   7467.46  took (0) s.ms
 --->  app_resume   7467.219
 --->  app_resume   7467.219  took (0) s.ms
 --->  clicked_list_cb   7472.113
 ---> --->  create_scroller   7472.115
 ---> --->  create_scroller   7472.131  took (.016) s.ms
 ---> --->  compute_prime   7507.701
 ---> --->  compute_prime   7522.143  took (14.442) s.ms
 --->  clicked_prime_cb   7522.145  took (14.445) s.ms
 --->  clicked_leak_cb   7522.153
 --->  clicked_leak_cb   7522.155  took (.002) s.ms
 --->  ethread_end   7525.127
 --->  ethread_end   7525.129  took (.002) s.ms
 --->  clicked_led_cb   7526.808
 ---> --->  device_led   7526.810
 ---> --->  device_led   7526.859  took (.049) s.ms
 --->  clicked_led_cb   7526.859  took (.051) s.ms
 --->  ui_app_orient_changed   7528.282
 --->  ui_app_orient_changed   7528.282  took (0) s.ms
 --->  rotation_cb   7528.373
 --->  rotation_cb   7528.373  took (0) s.ms
 --->  ui_app_orient_changed   7530.282
 --->  ui_app_orient_changed   7530.282  took (0) s.ms
 --->  rotation_cb   7530.340
 --->  rotation_cb   7530.340  took (0) s.ms
 --->  clicked_led_cb   7531.454
 ---> --->  device_led   7531.457
 ---> --->  device_led   7531.523  took (.066) s.ms
 --->  clicked_led_cb   7531.524  took (.070) s.ms
 --->  clicked_led_cb   7532.46
 ---> --->  device_led   7532.50
 ---> --->  device_led   7532.77  took (.27) s.ms
 --->  clicked_led_cb   7532.78  took (.32) s.ms
 --->  clicked_exit_cb   7533.444
 --->  clicked_exit_cb   7534.446  took (1.002) s.ms
 --->  app_pause   7534.600
 --->  app_pause   7534.603  took (.003) s.ms
 --->  app_terminate   7534.605
 --->  app_terminate   7534.742  took (.137) s.ms
 main   7535.75  took (69.312) s.ms
********************** trace.t.3039907840 ******************
done
Parse frame rate data
done
Frame rate diff
done
```

Frames Per Second Calculator

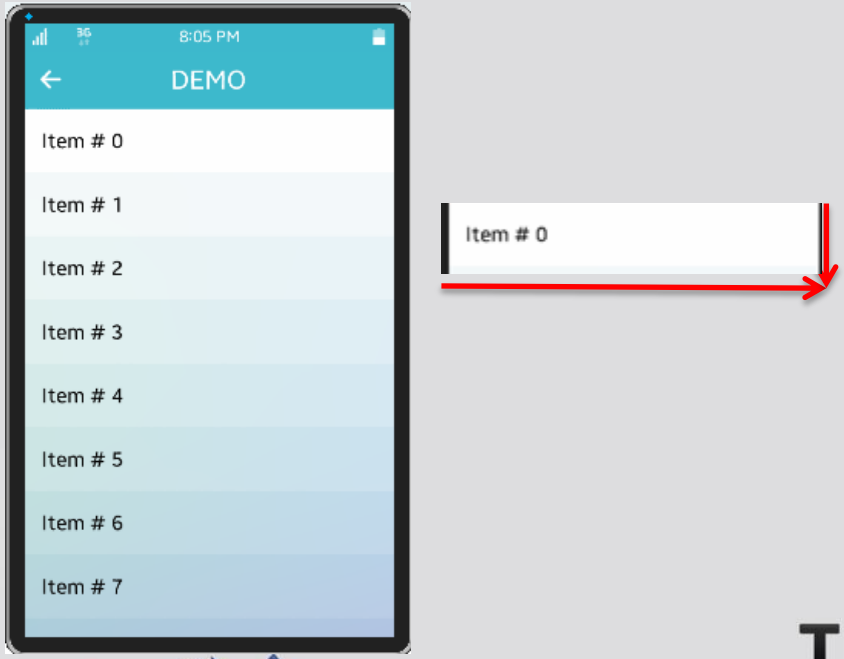


[evas_event_callback_add()]

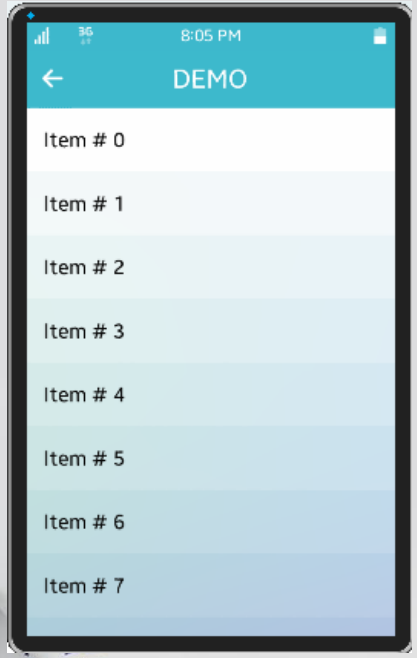
Performance Tips

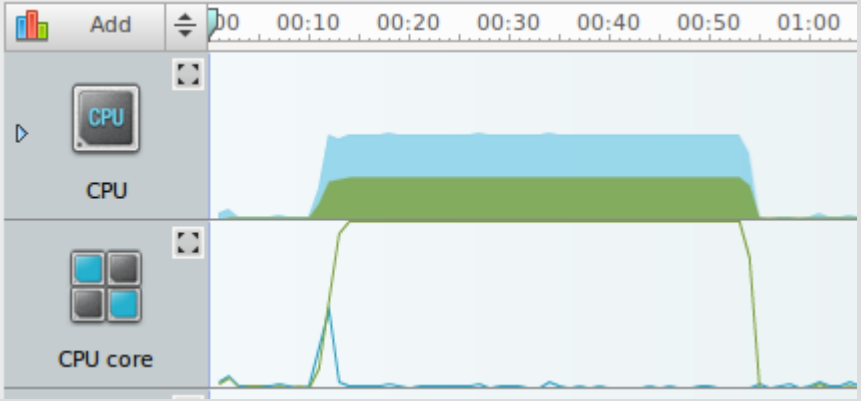
Performance Tips - EFL

- Use *elm_genlist_homogeneous_set* to lazy-loading which increases the performance for scrolling the list

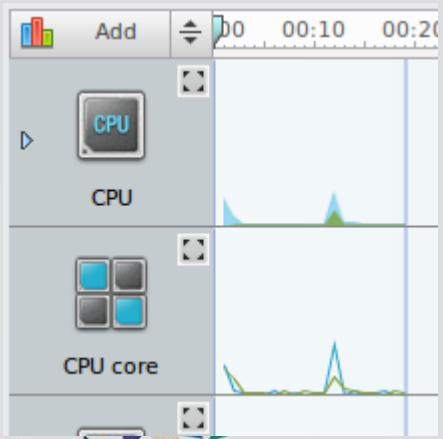


elm_genlist_homogeneous_set





Good



Better

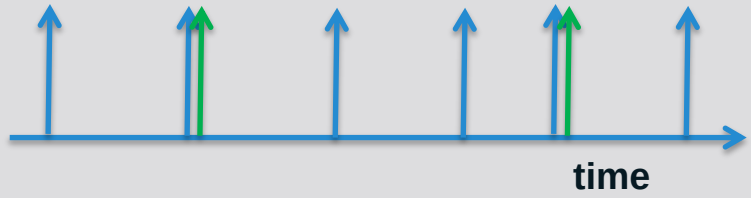
Performance Tips - EFL

- Use *evas_object_image_preload* to preload images in the background. Useful for albums slide show or gallery etc
- Eina supports base data types like hash, array, list. Choose the data structure that fits the best
  - Hash are good for search. Array is good for index based reference
- Recommended to call *app_resource_manager_init()* & *app_resource_manager_release()* during app init and terminate only. (API since 2.4)

Power Aware Programming

- Coalesce work to allow maximum idle time
  - Coalesce disk writes
  - Coalesce NW access
- Use data compression wherever possible in NW transmits
- Wherever possible, use *'ecore_poller_add'* instead of ecore timer
  - Ecore poller tries to call callbacks as many as possible, in one loop

Frequent wakeup



Less wakeups

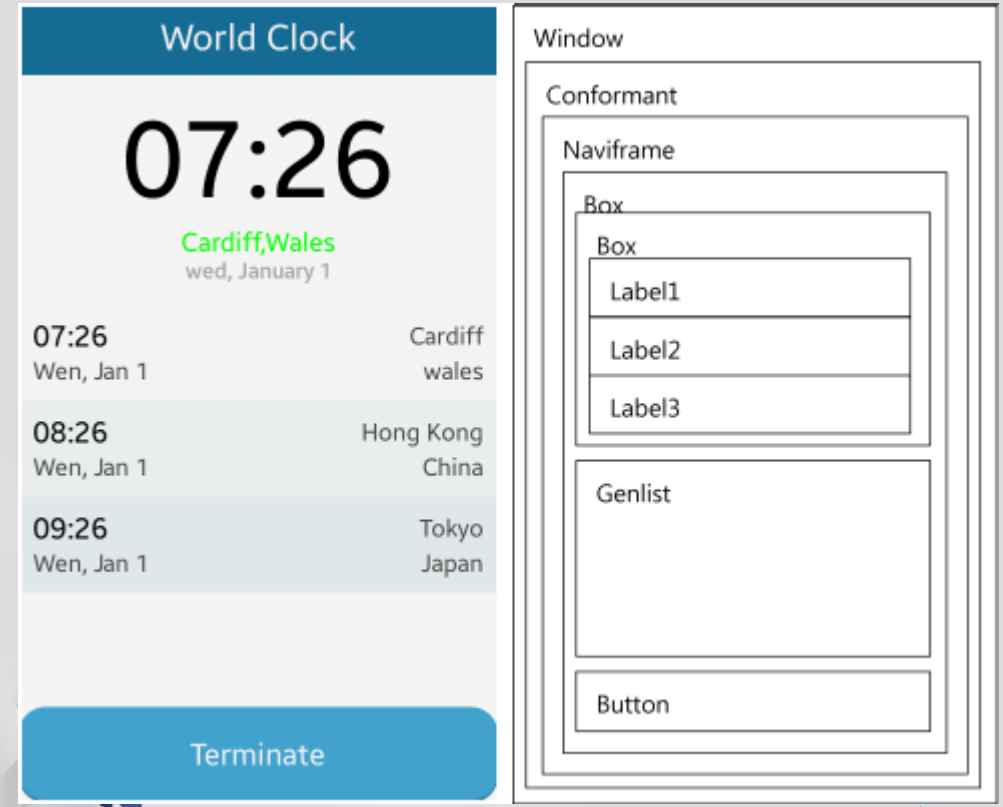
Be Responsive to System Events

- *device_add_callback()* allows you to monitor battery level, display state, charging state etc
  - Use this information to reduce CPU load or power hungry operations in your applications
- Be responsive to Low memory event
  - Free up memory by freeing memory pool, if any
  - Flush object caches. *elm_cache_all_flush()*
  - Any large DS which is not referred frequently, save it to disk and retrieve back later, when memory is OK
  - Fix memory leaks using IDE tool like ‘valgrind’

Running The Ttrace

- From IDE
- From Command Line
  - $ cd TIZEN_SDK_HOME/tools/ttrace
  - $ ./ttrace.py --time=10 --buf-size=102400 --o op_filename.html
  - $ ./ttrace.py --help

App Launch Time = Time To Draw First Screen



1. *Visual Readiness*
2. *Touch Readiness*
3. *On Demand*

App Launch Time – defer code

- **Visual Readiness**

```
$ evas_object_smart_callback_add(elm_win, "focus,in",
  window_focus_in_callback, NULL);
```

- **Touch Readiness**

1 From Window “focus” callback defer these codes using animator

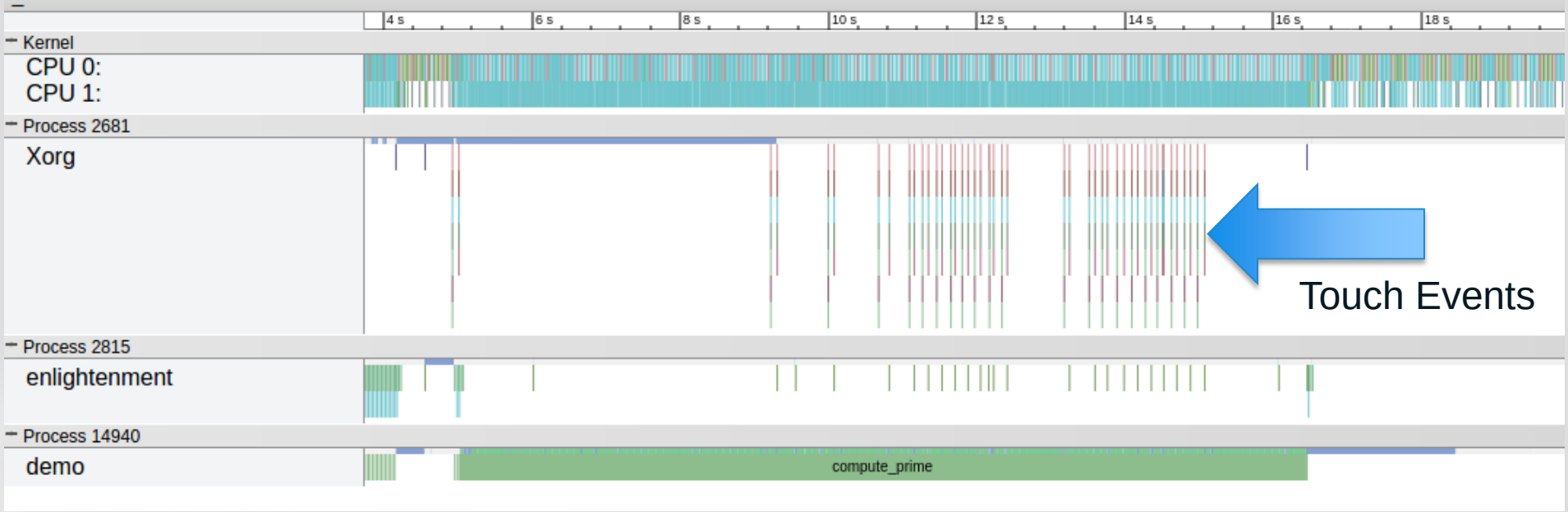
```
$ ecore_animator_add(Ecore_Task_Cb func,const void* data);
```

- **On Demand**

1 Usually done from respective UI call backs

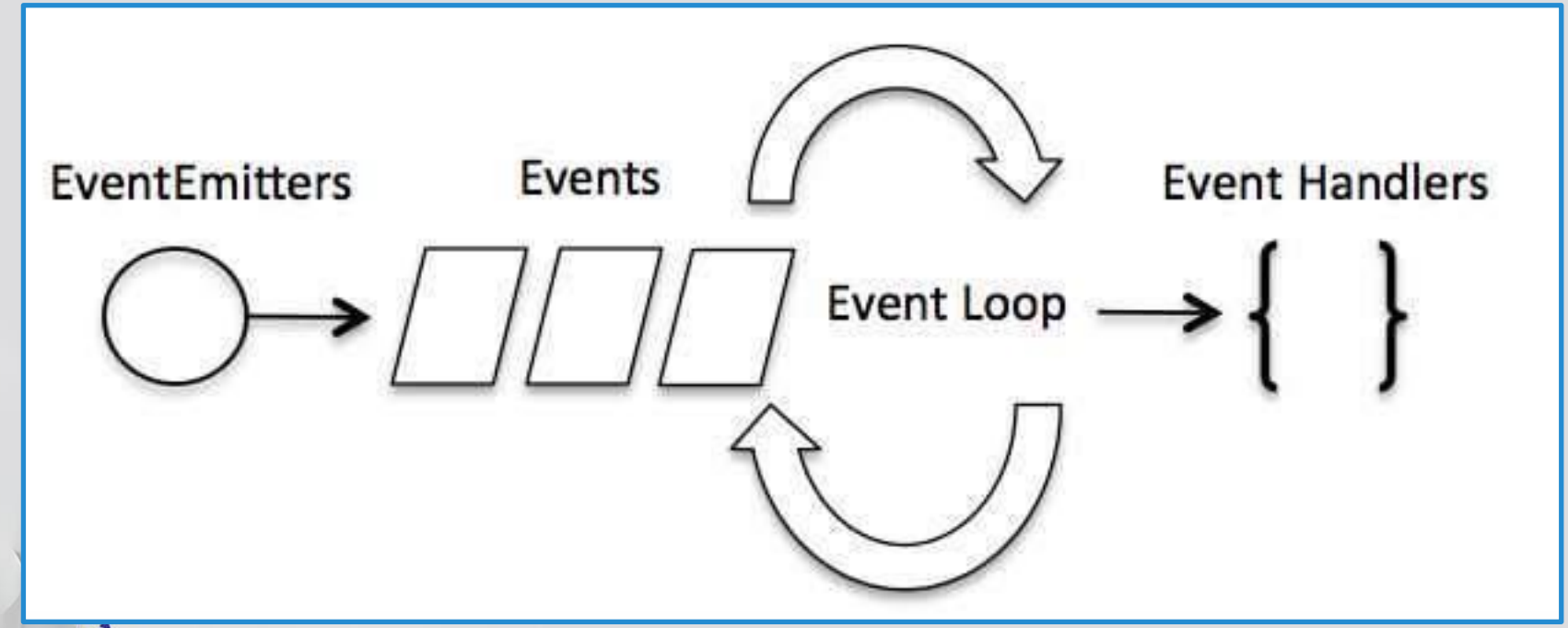
App Launch Time – Trace Example

App Freeze

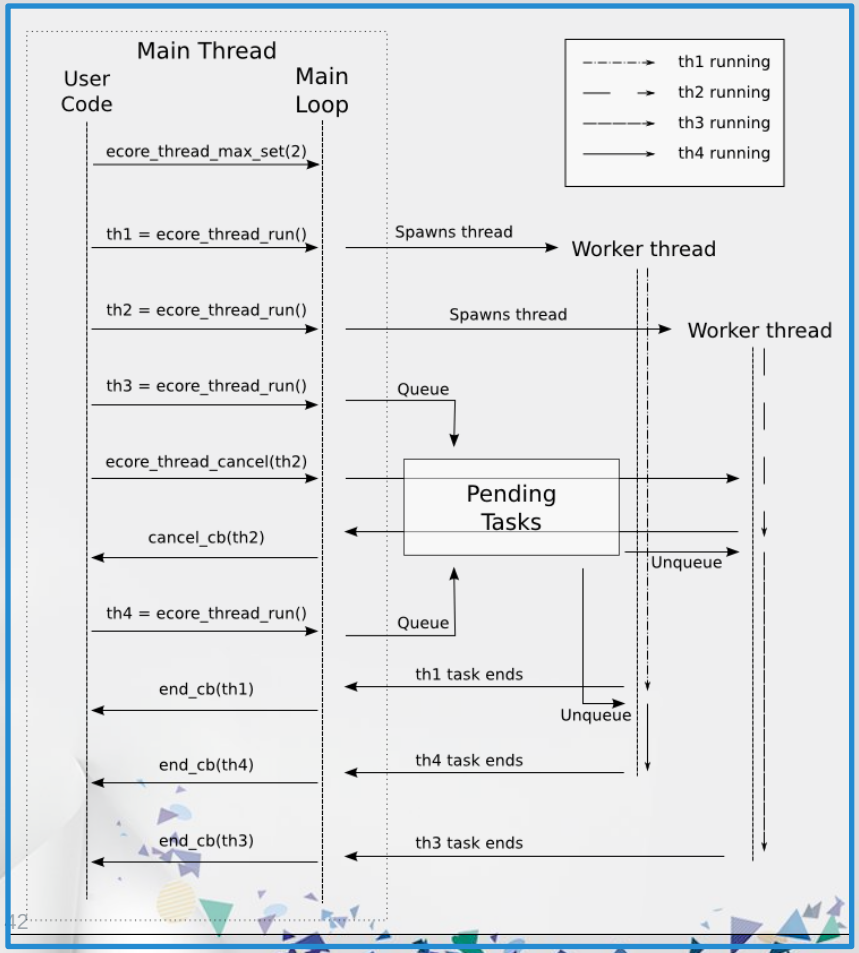


Why did it happen?

UI thread

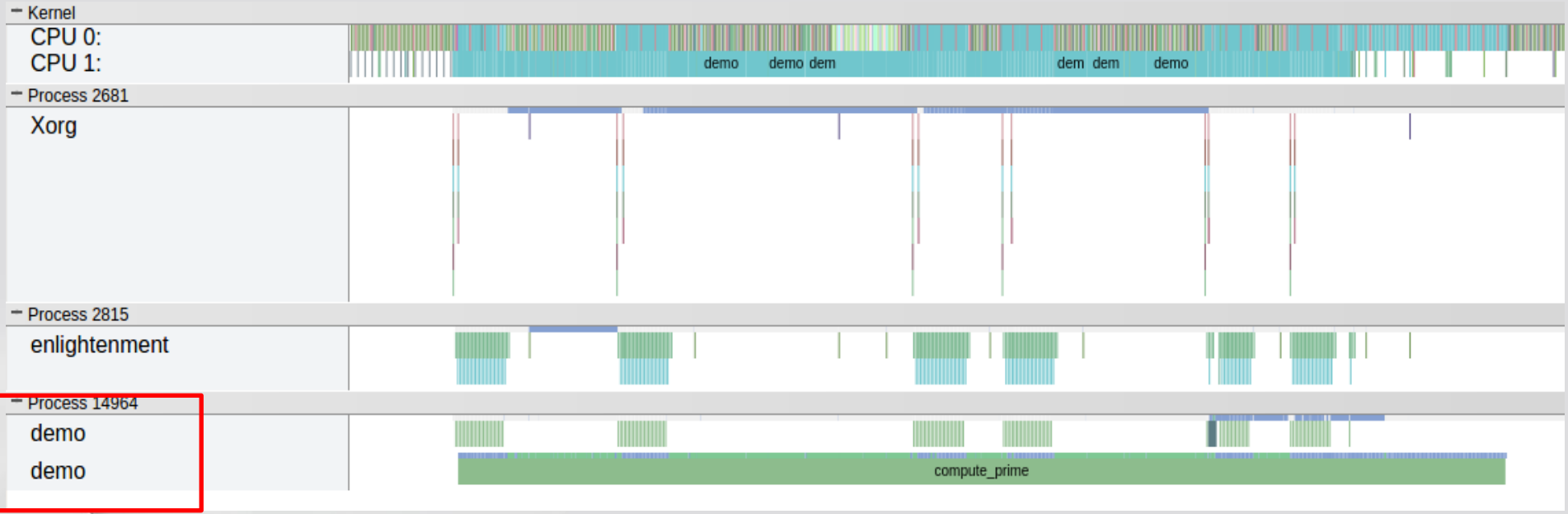


Have fewer blocking calls on UI thread!



** EFL is not thread safe*

Confirm the fix!



Frames Per Second - FPS

- The optimal frame rate is considered to be around **60** frames per second
- This means that application should spend at most **1/60 s = 16.7 ms** serving each frame
- Above that, there isn't much of a perceivable difference
- If its less, then user would notice UI stutter

Checking FPS with Tizen Trace