Trace d'exécutions de systèmes temps-réel

IMA5 2018/2019 P10

Amine El Messaoudi

Encadrant : Julien Forget

Contexte

Mettre au point un outils de trace d'un système temps-réel basé sur l'API ptask.

- Objectif : aide au débbugage d'un système temps-réel.
- Cahier de charges :
  - Documentation sur ptask et LTTng.
  - Implémentation d'un outil d'extraction des traces.
  - Implémentation d'un outil d'analyse et de visualisation des traces.

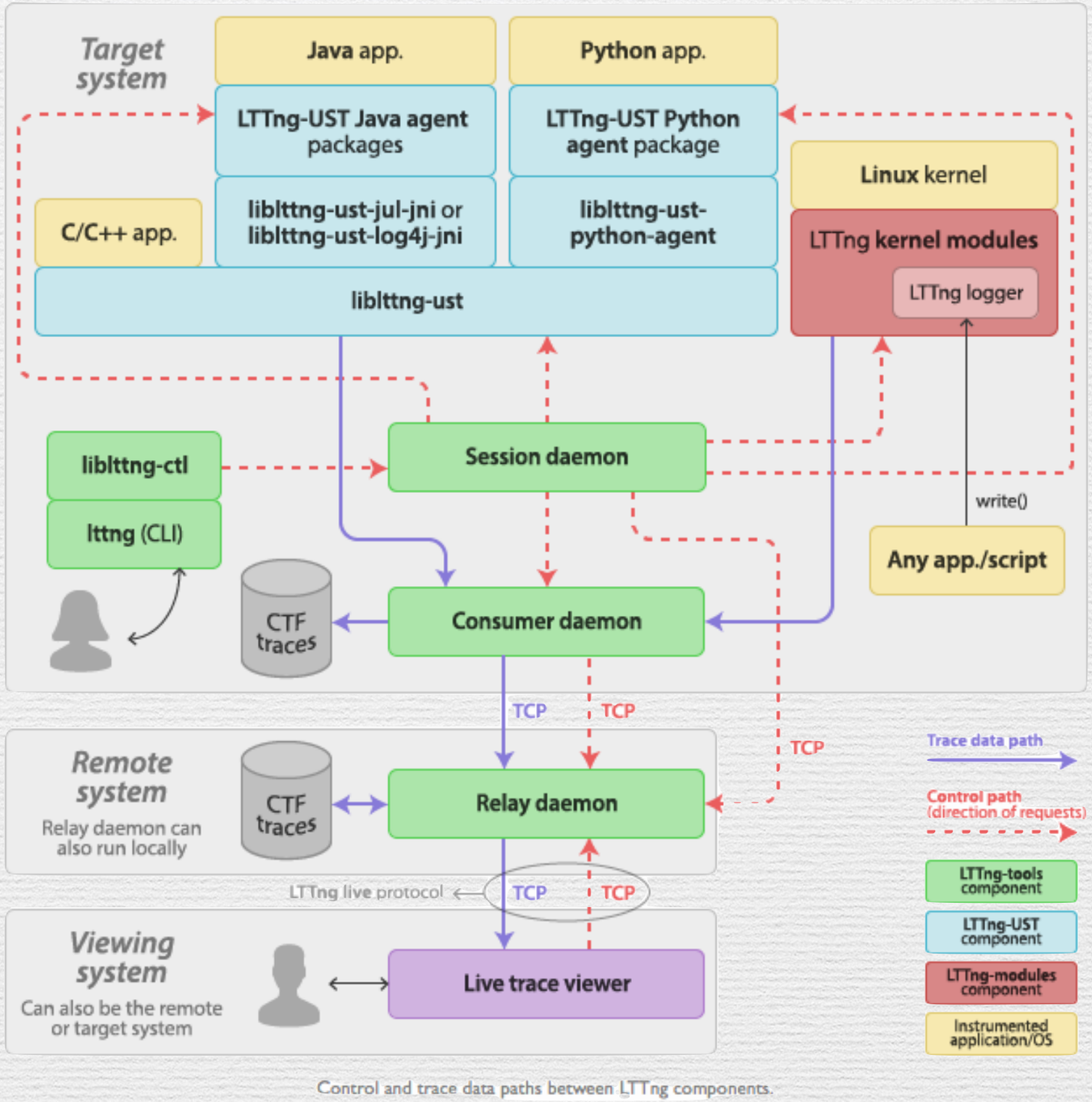
LTTng

- Toolkit opensource pour le traçage du noyau et de l'espace utilisateur linux.
- Se compose de :
  - session,
  - domaine,
  - channel,
  - événement.

Packages :

- LTTng-tool.
- LTTng-modules.
- LTTng-UST.

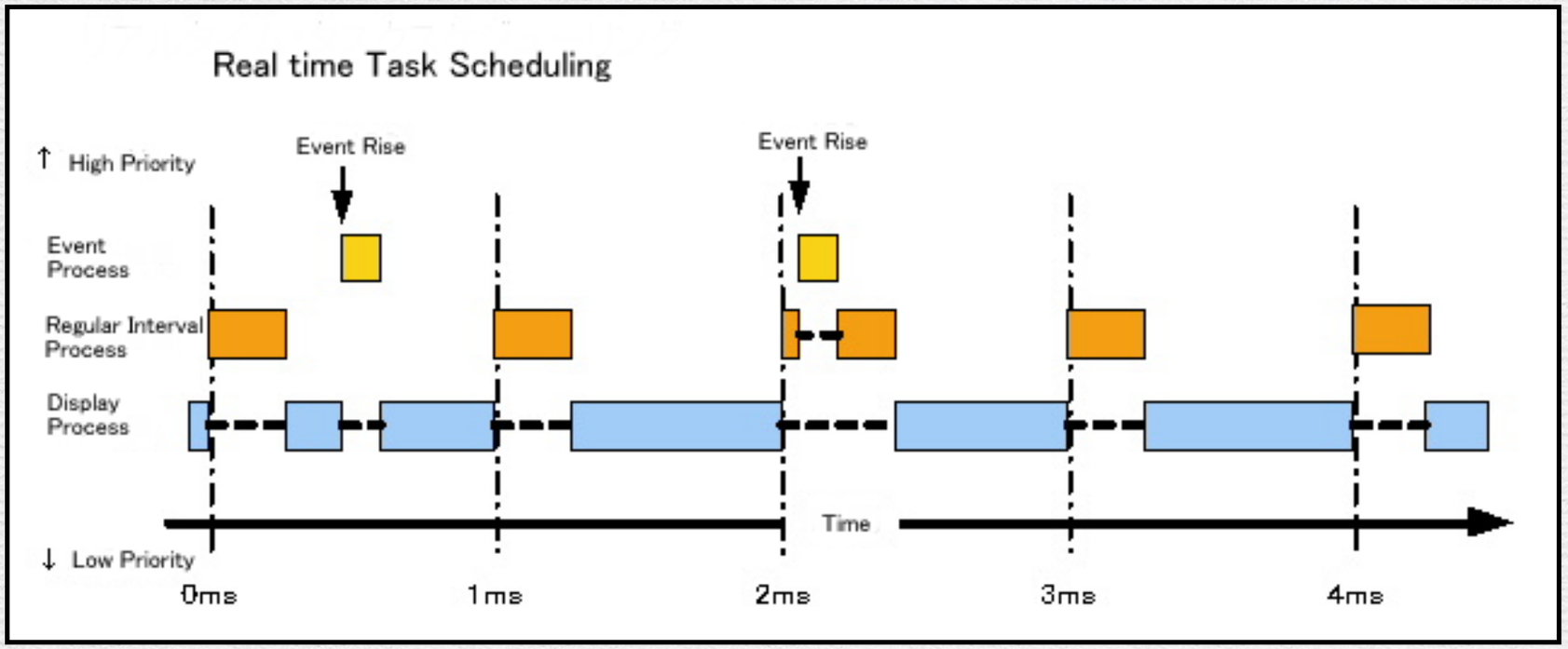
Contrôle



Control and trace data paths between LTTng components.

ptask

- Bibliothèques C pour le développement de tâches temps-réel pour linux.
- Surcouche de la bibliothèque pthread.
- Tâche périodique et tâche apériodique.



Tâche périodique

```c
ptask my_periodic_task() { //example of a periodic task
    int i;
    i = ptask_get_index();
    while (1) {
        // <do useful things as a function of i>
        ptask_wait_for_period(); //waiting till the next period
    }
}

int main() {
    ptask_init(SCHED_FIFO, GLOBAL, PRIO_INHERITANCE); //initialize ptask with the
scheduler, the scheduling type, and the protocol

    // <set up tasks parameters>
    int period, prio;

    ptask_create(my_periodic_task, period, prio, NOW);

    while(1) ;

    return 0;
}
```

LTTng avec ptask

Besoins :

- Date de début et de fin d'exécution ==> Tracepoints ptask :

```
tracepoint(ptask_provider, ptask_tracepoint, pid, tid, ptask_idx, flag, state,
tspec_to_rel(&now, MICRO), _tp[ptask_idx].priority, tspec_to(&_tp[ptask_idx].period,
MICRO), tspec_to(&_tp[ptask_idx].deadline, MICRO));
```

- Date de réveil ==>
  - Appel système `clone` dans le noyau.
- Préemptions ==>
  - Tracepoint du noyau `sched_switch`.

Récupération de traces

- Babeltrace : Implementation de référence CTF

```
[12:20:16.002620588] (+0.000004208) debian-vm syscall_exit_clone: { cpu_id = 0 }, { pid =
16226, tid = 16308 }, { ret = 0 }
[12:20:16.002593777] (+0.000034078) debian-vm syscall_entry_clone: { cpu_id = 0 }, { pid =
16226, tid = 16226 }, { clone_flags = 0x3D0F00, newsp = 0x7FC06FF5CF70, parent_tid =
0x7FC06FF5D9D0, child_tid = 0x7FC06FF5D9D0 }
[12:20:16.842354195] (+0.000009715) debian-vm ptask_provider:ptask_tracepoint: { cpu_id =
0 }, { ptask_flag = "DEFERRED", ptask_state = "b_wait_period", ptask_pid = 16226,
ptask_tid = 16308, ptask_index = 0, ptask_time = 150276421, ptask_priority = 80,
ptask_period = 20000, ptask_deadline = 20000 }
[12:20:16.842358357] (+0.000004162) debian-vm sched_switch: { cpu_id = 0 }, { pid = 16226,
tid = 16308 }, { prev_comm = "ball", prev_tid = 16308, prev_prio = -81, prev_state = 1,
next_comm = "ball", next_tid = 16309, next_prio = 20 }
```

- ANTL*4* : parser.g4, laxer.g4 + Java

Visualisation

Reproduction graphique des traces :

- Déterminer une solution pour le rendu final : jfreechart, svg, …
- Analyse des traces, et organisation des informations : java.

Conclusion

Bilan :

- Instrumentation de ptask.
- Traçage.
- Parsing des traces.

Perspective :

- Analyse et organisation des traces.
- Représentation graphique.