Profile Functions with Per-core Calclock

Practical Class 9

Systems and Storage Laboratory

Department of Computer Science and Engineering

Chung-Ang University

Contents

1. Why Per-core Calclock?

- Original calclock adding to kernel (calclock from practical class 7-a) can **degrade** system performance
- Per-core calclock adding to kernel **does not** much affect system performance

2. Per-core Calclock vs Original Calclock

Per-core Calclock

```c
#include "calclock.h"

KTDEF(function_A);
int foo(void) {
	int ret;
	ktime_t stopwatch[2];

	ktget(&stopwatch[0]);
	ret = function_A();
	ktget(&stopwatch[1]);
	ktput(stopwatch, function_A);

	return ret;
}
```

Original Calclock

```c
#include "calclock.h"

unsigned long long count, time;
int foo(void) {
	int ret;
	struct timespec stopwatch[2];

	getrawmonotonic(&stopwatch[0]);
	ret = function_A();
	getrawmonotonic(&stopwatch[1]);
	calclock(stopwatch, &time, &count);

	return ret;
}
```

Per-core Calclock	Original Calclock
KTDEF(func_name)	unsigned long long count, time
ktime_t	struct timespec
ktget	getrawmonotonic
ktput	calclock

2. Per-core Calclock vs Original Calclock

Per-core Calclock

```
#include "calclock.h"

KTDEC(function_A);
void exit_module(void) {
     ktprint(1, function_A);
}
```

Original Calclock

```
#include "calclock.h"

extern unsigned long long count, time;
void exit_module(void) {
     printk("Function A is called %llu times, and the time interval is %llu ns\n",
count, time);
}
```

Per-core Calclock	Original Calclock
KTDEC(func_name)	extern unsigned long long count, time

3. Implementation

- Firstly, in calclock.h, **define CONFIG_CALCLOCK**

calclock.h

```c
#ifndef __CALCLOCK_H
#define __CALCLOCK_H
#include <linux/ktime.h>
#include <linux/percpu.h>

#define CONFIG_CALCLOCK

struct calclock {
    ktime_t time;
    unsigned long long count;
};

// structs and functions are defined here!

#else /* !CONFIG_CALCLOCK */
#define ktget(clock)
#define ktput(localclock, funcname)
#define ktprint(depth, funcname)
#endif /* CONFIG_CALCLOCK */

#endif /* __CALCLOCK_H */
```

3. Implementation

- **Calclock structure**

calclock.h

```
struct calclock {
    ktime_t time;
    unsigned long long count;
};
```

  - **time**: Time interval of the function being measured
  - **count**: The number of this calclock called by its thread

3. Implementation

- **KTDEF**: Defines a new calclock

calclock.h

```
#define KTDEF(funcname) \
    DEFINE_PER_CPU(struct calclock, funcname##_clock) = {0, 0}
```

- **KTDEC**: Declares an existing calclock defined in another file

calclock.h

```
#define KTDEC(funcname) \
    DECLARE_PER_CPU(struct calclock, funcname##_clock)
```

3. Implementation

- **ktget:** Gets current time and saves it into a **local clock**

❑ **Local clock** is a pointer of **ktime_t**

calclock.h

```
static inline void ktget(ktime_t *clock)
{
    *clock = ktime_get_raw();
}
```

Example:

foo.c

```
#include "calclock.h"

KTDEF(function_A);
int foo(void) {
      int ret;
      ktime_t stopwatch[2];

      ktget(&stopwatch[0]);
      ret = function_A();
      ktget(&stopwatch[1]);
      ktput(stopwatch, function_A);

      return ret;
}
```

2 local clocks of ktime_t are declared

ktget gets the start time and end time of 'function_A', saves them into 'stopwatch' local clocks

3. Implementation

- **ktput:** gets current per-core calclock and calculates time interval
  - **localclocks**: Local clocks of ktime_t
  - **funcname**: Name of per-CPU calclock (named freely)

calclock.h

```
#define ktput(localclocks, funcname)                           \
do {                                                           \
    struct calclock *clock;                                    \
    bool prmpt_enabled = preemptible();                        \
                                                               \
    if (prmpt_enabled)                                         \
        preempt_disable();                                     \
    clock = this_cpu_ptr(&(funcname##_clock));                 \
    __ktput(localclocks, &clock->time);                        \
    clock->count++;                                            \
    if (prmpt_enabled)                                         \
        put_cpu_ptr(&(funcname##_clock));                      \
} while (0)
```

3. Implementation

- **ktprint:** prints total time and counts of the per-core calclock
  - ❑ **depth**: The number of "tab" to the left side
  - ❑ **funcname**: Name of per-core calclock (named freely)

calclock.h

```
#define ktprint(depth, funcname)                                          \
do {                                                                       \
    int cpu;                                                               \
    ktime_t timesum = 0;                                                   \
    unsigned long long countsum = 0;                                       \
                                                                           \
    for_each_online_cpu(cpu) {                                             \
        struct calclock *clock = per_cpu_ptr(&funcname##_clock, cpu);      \
        timesum += clock->time;                                            \
        countsum += clock->count;                                          \
    }                                                                      \
    __ktprint(depth, #funcname, timesum, countsum);                        \
} while (0)
```

3. Implementation

- Refer uploaded **calclock.h, calclock.c** files at

(https://github.com/loglamo/CAU-CSE-LinuxApplications-20232)