Arquitectura de Sistemas

Práctica 7: Cooperación entre hebras

Gustavo Romero López

Updated: 15 de septiembre de 2022

Arquitectura y Tecnología de Computadores

Objetivos

- Supondremos al alumno familiarizado con varios tipos de hebras:
  - **hebras tipo POSIX o pthreads**
  - **std::threads de C++11**
- Como material de referencia podemos consultar las páginas **cppreference.com** y **cplusplus.com**.
- Descubriremos su funcionamiento mediante una serie de ejemplos.
- Al final el alumno debe crear un programa para comprobar los conocimientos adquiridos.

- Hebras C++11 explicadas para usuarios de pthreads.

`std::thread`

- Clase que representa una hebra de ejecución.
- Miembros:
  - tipos: `native_handle_type`: tipo subyacente (pthread).
  - clases: `id`: representa la identificación de una hebra.
  - funciones:
    - `(constructor)`
    - `(destructor)`
    - `operator=`
    - `joinable`
    - `get_id`
    - `native_handle`
    - `hardware_concurrency`
    - `join`
    - `detach`
    - `swap`

hola.cc

```
#include <iostream>
#include <thread>

void hola() { std::cout << "hola"; }

int main()
{
  std::thread t1(hola);
  t1.join();

  std::thread t2([]{ std::cout << " mundo!\n"; });
  t2.join();
}
```

lambda.cc

- función lambda = función anónima.
- declaración completa: `[ capture-list ] (params)mutable (optional)exception attribute -> ret { body }`
- uso habitual: `[]{ cuerpo }`
- documentación de **funciones lambda**.
- ejemplo:

```
int main()
{
  auto f = []{ cout << "hola, mundo!\n"; };
  f();
}
```

Trabajo a realizar

- Estudie la tres formas de implementar un servidor web mostradas al final del **tema 7, activación**.
  - Basada en **procesos**.
  - Basada en **hebras**.
  - Basada en **grupos de hebras**.
- La misma idea se ha aplicado en estos ejemplos:
  - **server0.cc** (1 proceso/secuencial).
  - **server1.cc** (N procesos/paralelo).
  - **server2.cc** (nº óptimo de procesos /paralelo).
  - **server3.cc** (N hebras nucleo/paralelo).
  - **server4.cc** (nº óptimo de hebras núcleo/paralelo).
  - **server5.cc** (N hebras usuario/paralelo).
  - **server6.cc** (nº óptimo de hebras usuario/paralelo).
- Existen 2 cargas de trabajo, ficheros `work.h`, cada una de las cuales ejercita el uso del procesador o de la E/S.

https://pccito.ugr.es/as/practicas/07/cpu/work.h

```
const int N = 75, M = N * N;

int v1[N][M], v2[M][M], v3[N][M];

void work(int begin, int end)
{
  for (int i = begin; i < end; ++i)
    for (int k = 0; k < M; ++k)
      for (int j = 0; j < M; ++j)
        v3[i][j] += v1[i][k] * v2[k][j];
}
```

https://pccito.ugr.es/as/practicas/07/io/work.h

```
void work(int begin, int end)
{
  int max = 0, i = 0;
  while (begin++ != end)
  {
    try
    {
      std::ifstream ifs(path + std::to_string(rng()) + ".txt");
      while (ifs)
      {
        ifs >> i;
        max = std::max(max, i);
      }
    }
    catch(...) {}
    try
    {
      std::ofstream ofs(path + std::to_string(rng()) + ".txt");
      ofs << max + 1 << std::endl;
    }
    catch(...) {}
  }
```

Trabajo a realizar

- Dado que no puede modificar las implementaciones, intente explicar la diferencia de rendimiento entre ellas.
- En la práctica anterior nos centramos en estudiar el rendimiento con `perf`.
- Ahora vamos a intentar buscar otro tipo de cuellos de botella que puedan ralentizar la ejecución de los programas de prueba:
  - Llamadas al sistema: `strace`
  - Funciones de biblioteca: `ltrace`
- ¿Los programas de prueba funcionan bien?
- ¿Cuál cree que es mejor?
- ¿Se puede optimizar alguna de las cargas de trabajo?