C2115 Practical Introduction to Supercomputing

6th Lesson

Petr Kulhánek, Jakub Štěpán

kulhanek@chemi.muni.cz

National Centre for Biomolecular Research, Faculty of Science
Masaryk University, Kotlářská 2, CZ-61137 Brno

esf european social fund in the czech republic
EUROPEAN UNION
MINISTRY OF EDUCATION, YOUTH AND SPORTS
OP Education for Competitivness
UNIVERSITAS MASARYKIANA BRUNENSIS

INVESTMENTS IN EDUCATION DEVELOPMENT

CZ.1.07/2.2.00/15.0233

Contents

- **Multiuser environment**
  process, thread, multitasking, context switch
- **Exercise**
  parallel application run efficiency

Multiuser environment

- Process
- Thread
- Multitasking
- Context switch

Process and thread

Process is in informatics name of **running computer program**. Process is placed in operating memory as a machine instruction sequence processed by processor. Contains not only code of processed program, but also dymanically changing data, that are processed. **One program** may run as **multiple processes with different data**. Process management is done by operating system, that ensures their separate run, assigns system resources and provides user tools to manage processes.

Thread is lightweighted process, that allows lowering down costs of operating system during context switch, that is necessary to allow massive parallel calculations. While processes are strictly separated, threads share same **memory space** and other structures. One process may handle multiple threads, threads comunicate easily through shared memory, but this brings also possible problems in concurent memory access - **race condition**.

source: www.wikipedia.cz, adjusted

Multitasking

Multitasking is ability of operating system to seemingly process **multiple processes at one time**. Operating system core switches running processes on processor very fast (**context switch**), so these processes seem to be running concurently.

Types:

- non-preemptive multitasking (not used much nowadays)
- preemptive multitasking (modern OS)

Preemptive multitasking means that operating system itself decides about resources assigned to particular processes. **Periodically** (typical rate is approximately 100× to 1000× per second) interupts processing of running process, evaluates situation (number of waiting processes, their priorities etc.) and then decides either to start interupted process again or starts onother process that was waiting. During process switch also **context has to be switched**. Process in preemptive multitasking may ask for context switch and give up its resources (it is put to „sleep“ or waits for slow input-output operatins, as is for example hard disk reading).

source: www.wikipedia.cz, adjusted

Context switch

Context

The term denotes **processor state** (register contents), state of coprocessor and possibly other devices in time of context change. This particular state is saved either to process stack or to dedicated part of process memory space.

Contaxt contains also contents of processor cache memory levels (for example L1 cache or TLB): these are not saved, but their contents **are implicitly or explicitly invalidated during context switch**. Necessity of their new loading is main reason why **context switch is so time demanding** on modern architectures.

Context switch is operation of multitaking operating system that **switches controll among processes**. This implies saving and loading of **current processor state**. This is repeated many times per second. Context changes are usually computationaly intensive.

source: www.wikipedia.cz, adjusted

Exercise

- **Parallel application run efficiency**

Exercise LV.1

1. How many processes contains your machine? Get machine mane, CPU type and number. Use command **lscpu**.
2. Extend and compare data obtained in task 1 with information from file **/proc/cpuinfo**.
3. What is size of cache memory L1, L2 and L3 CPU in your machine?
4. Get your CPU frequency?
5. Compile program **load_cpu.f90**, that is in directory **/home/kulhanek/Data/C2115/programs**

```
$ gfortran -O3 load_cpu.f90 -o load_cpu -lblas
```

6. State runtime of program **load_cpu** using command:

```
$ /usr/bin/time --format=%e ./load_cpu
```

7. Give reason why not to use directly command **time**? What is option **--format?**

Exercise LV.2

1. To run program **load_cpu** use following script in bash.

```
#!/bin/bash

N=4     # parralel runs number

for((I=1;I<=N;I++)); do
        ./load_cpu &  # runs application on backgroud
    done
wait    # waits for all background runs to finish
```

2. Do script function analysis.
3. Measure script runtime. Measured time compare with theoretical expectations based on run of sigle program **load_cpu**. Do measurements for N=1,NCPU,2*NCPU,3*NCPU,4*NCPU,5*NCPU,6*NCPU, where NCPU is CPU number accesible on your machine. Monitor running processes by command **top** in separate terminal.
4. Analyze obtained differences in runtime and explain them.