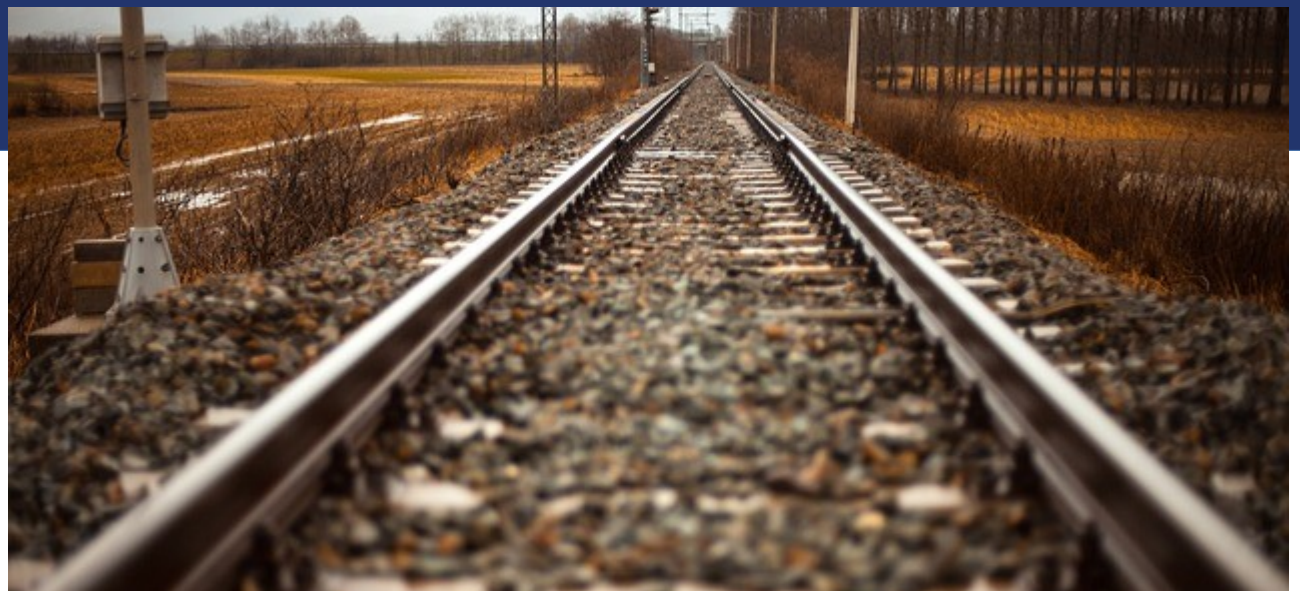
Parallel programming

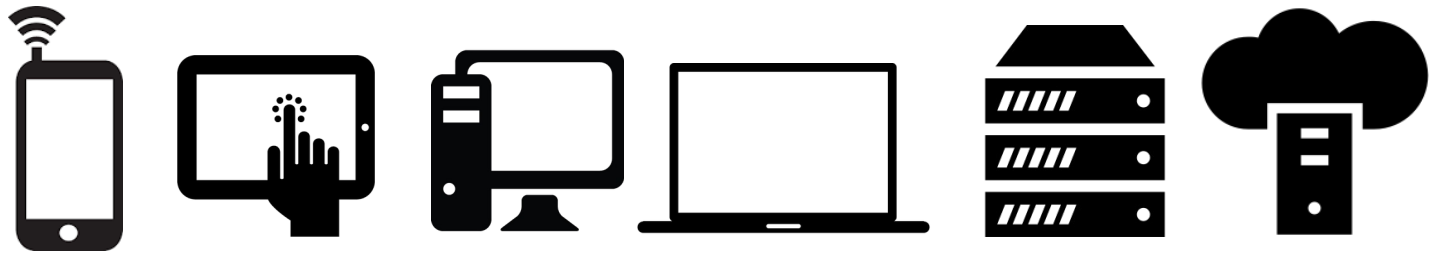
Introduction



FAKULTA
ELEKTROTECHNICKÁ
ČVUT V PRAZE

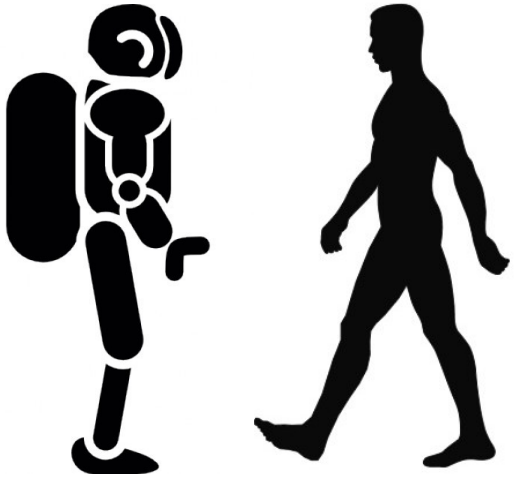
Why should you care about it?

- Parallel computing is a dominant player in scientific and cluster computing. Why?
  - Moore law is reaching its limits
    - Increase in transistor density is limited
    - Memory access time has not been reduced at a rate comparable with processing speed
- How to get out of this trap?
  - Most promising approach is to have multiple cores on a single processor.
  - Parallel computing can be found at many devices today:



Ok; However, It should be task for compiler and not for me!!!

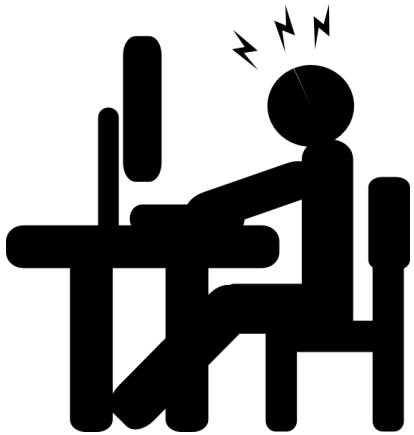
- Yes, compiler can help you, but without your guidance, it is not able pass all the way to the successful result.
  - Parallel programs often look very different than sequential ones.
  - An efficient parallel implementation of a serial program may not be obtained by simply parallelizing each step.
  - Rather, the best parallelization may be obtained by stepping back and devising an entirely new algorithm.



What is the aim of the labs?

- To get the feel for parallel programming
  1) Understand what makes the parallelisation **complicated**
  2) Which **problems** can occur during the paralellisation
  3) What can be a **bottleneck**
  4) How to think about **algorithms** from the paralellisation point of view

- To get basic skills in common parallel programming frameworks
  1) for Multicore processors – C++11 threads, OpenMP
  2) for Computer clusters – MPI

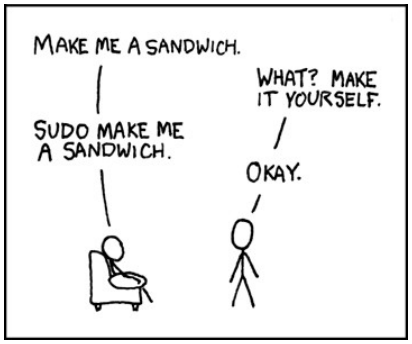


Course web

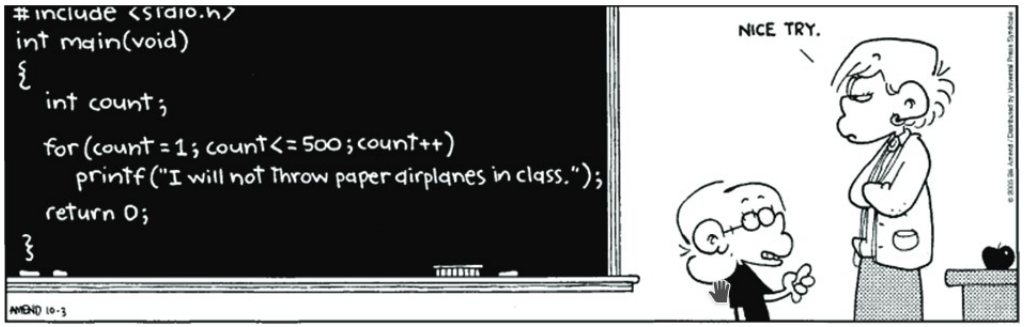
- Course page https://cw.fel.cvut.cz/b191/courses/pag/start
  - Plan of the labs, grading

What this course requires?

- Basic skills with Linux – shell, ssh, etc. (for MetaCentrum)



- Knowledge of C and C++ language



- Analytical thinking and opened mind

Setting up your programming environment

Linux, Mac OS, Windows

- CMake and gcc
- Recommended IDE: **CLion**
  - https://download.cvut.cz, JetBrains
- Homework and semestral project skeletons provided only as Cmake projects
- **Windows:** see next slide
- **Mac OS:** see next+1 slide

Windows+Visual Studio? :(

- Use at your own risk
- **Do not use MSVC** (no support for newer OpenMP)
- Instead, **use** Intel icc compiler (part of Intel Parallel Studio)

Ubuntu toolchain

- You can use inofficial PPA for the Clion, see this link.
- Instal g++ and cmake

```
>> sudo apt install g++ cmake
```

Windows toolchain

- Install msys2, see this link
- In the msys2 console do the following

```
>> pacman -Syu
>> pacman -Su
>> pacman -S base-devel mingw-w64-x86_64-toolchain
```

- Create MinGW toolchain in CLion, see this link. If msys2 is installed in default location, set `C:\msys64\mingw64` as your MinGW Environment path (everything else should be detected automatically)
- Add msys2 directories to your PATH environment variable, e.g.,

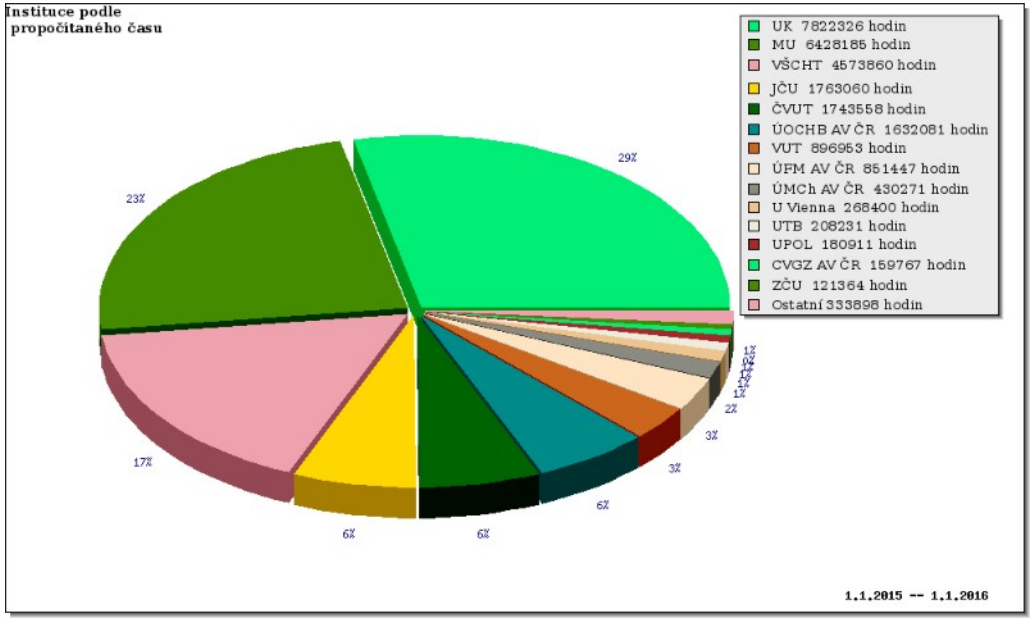
```
C:\msys64
C:\msys64\mingw64\bin
```

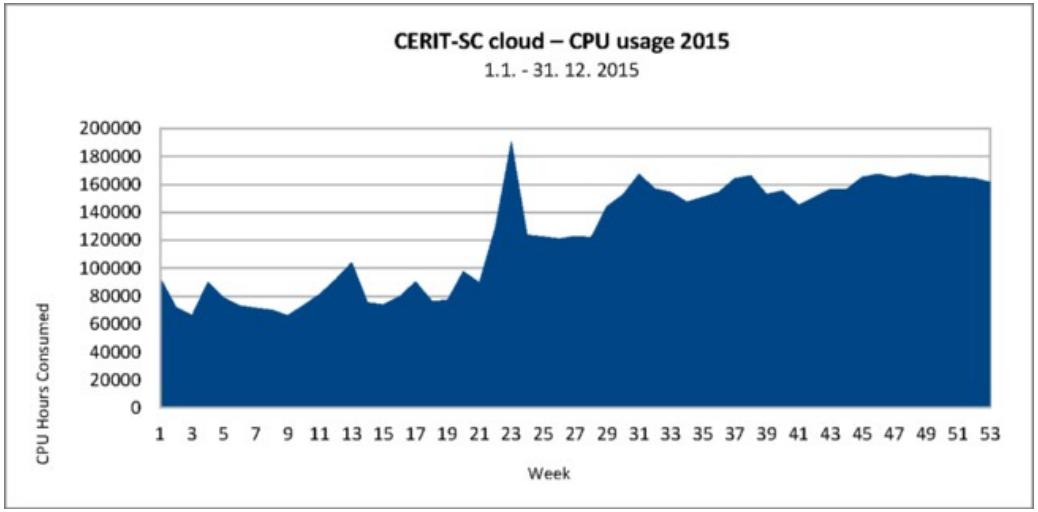
MacOS toolchains

- Using g++ **(recommended)**
  - Install g++ from Homebrew
    ```
    >> brew install gcc
    ```
  - Find the installed g++ executable. Usually a program called ***g++-FŇ*** where ***FŇ*** is the version (can be found using TAB completion), e.g., ***g++-9***
  - Set ***g++-FŇ*** compiler in CLion: Settings → Build, execution, Deployment → Toolchains → C++ compiler
- Using clang
  - Install OpenMP runtime from Homebrew
    ```
    >> brew install libomp
    ```
  - Check where libomp is installed, usually `/usr/local/opt/libomp`
    ```
    >> brew --prefix libomp
    ```
  - Link OpenMP into CMakeLists.txt
    ```
    include_directories("/usr/local/include" "/usr/local/opt/libomp/include")
    link_directories("/usr/local/lib" "/usr/local/opt/libomp/lib")
    ```

MetaCentrum system

- operates and manages **distributed computing** infrastructure consisting of computing and storage resources owned by **CESNET**
- MetaCentrum membership is free for researchers and students of academic institutions in the Czech Republic





MetaCentrum – Sign up