

# Parallel programming

## Introduction



Libor Bukata a Jan Dvořák

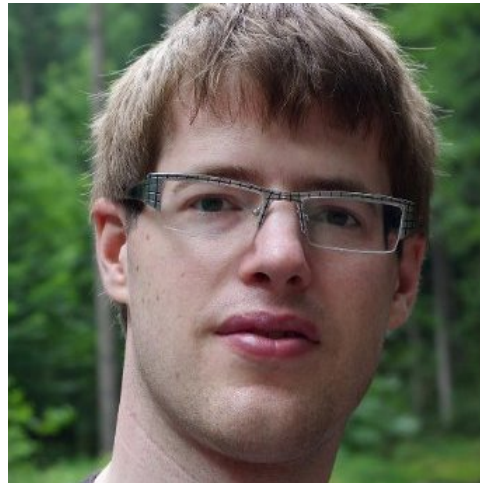


**FAKULTA  
ELEKTROTECHNICKÁ  
ČVUT V PRAZE**



# Instruktors

- Libor Bukata



- Jan Dvořák





# Web sites

- <https://cw.fel.cvut.cz/wiki/courses/b4m35pag/start>

[\[courses:b4m35pag:start\]](#) COURSE WARE

Hide Sidebar Login Search

Trace: • b4m35pag

## Paralelní algoritmy (B4M35PAG)

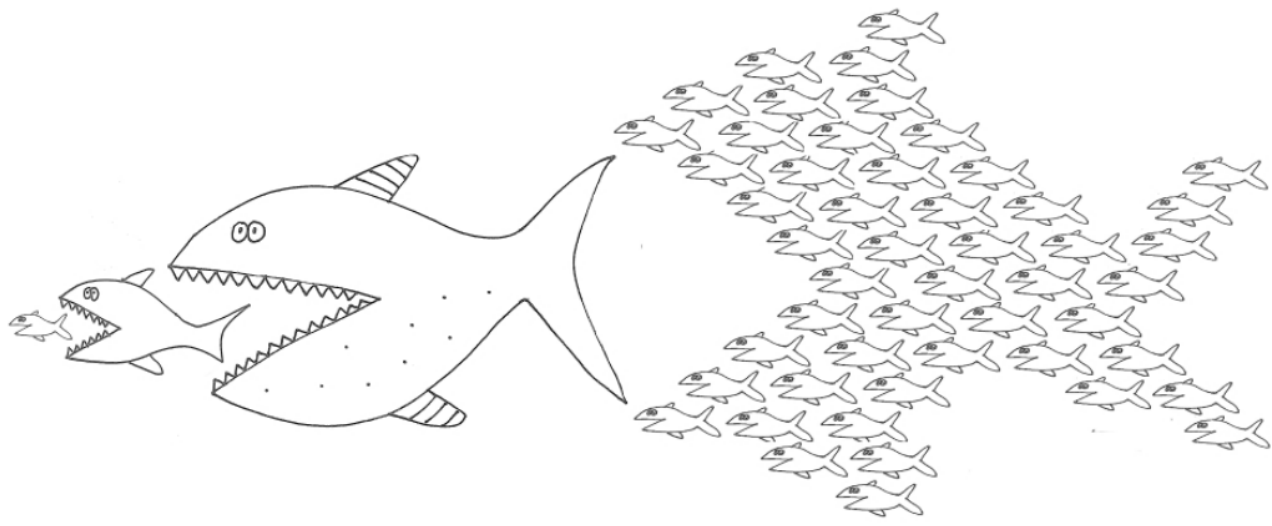
Přednášející: Přemysl Šůcha

Cvičící: Jan Dvořák a Libor Bukata

Rychlé odkazy: [Paralelní algoritmy - cvičení](#), [Upload system](#), [Diskusní Fórum](#), [Kde najdete místnosti](#), [Rozvrh předmětu](#), [Časový plán akademického roku](#)

### Proč paralelní algoritmy?

Table of Contents
• Paralelní algoritmy (B4M35PAG)
• Proč paralelní algoritmy?
• Novinky
• Obsah předmětu
• Osnova přednášek
• Hdnocení a zkouška
• Literatura



### Novinky



Vítáme Vás na stránkách nového předmětu věnovaného paralelním algoritmům. Věříme, že předmět bude pro Vás přínosem.





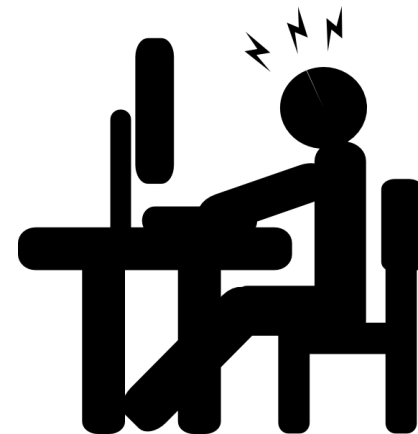
# Academic calendar

04.10.	Introduction to Parallel programming
11.10.	Pthreads
18.10.	C++11 threads
25.10.	1. individual lab assignment (ILA), semestral work assignment
01.11.	Basics of OpenMP
08.11.	Advanced statements in OpenMP
15.11.	2. ILA, <b>deadline for submission of 1. ILA</b>
22.11.	<i>Consultation of the semestral work</i>
29.11.	OpenMPI
06.12.	3. ILA, <b>deadline for submission of 2. ILA</b>
13.12.	<i>Consultation of the semestral work</i>
20.12.	<b>Presentation</b> of results from <b>semestral work</b>
03.01.	<b>Deadline for submission of 3. ILA</b>
10.01.	<i>Reserve</i>



# What is the aim of this lab?

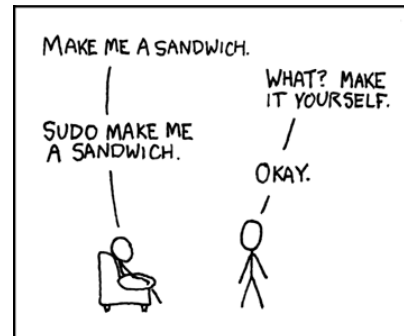
- To get the feel for parallel programming
  - 1) Understand what makes the parallelisation **complicated**
  - 2) Which **problems** can occur during the parallelisation
  - 3) What can be a **bottleneck**
  - 4) How to think about **algorithms** from the parallelisation point of view
- To get basic skills in common parallel programming frameworks
  - 1) for Multicore processors – Pthreads, C++11 threads, OpenMP
  - 2) for Computer clusters – OpenMPI



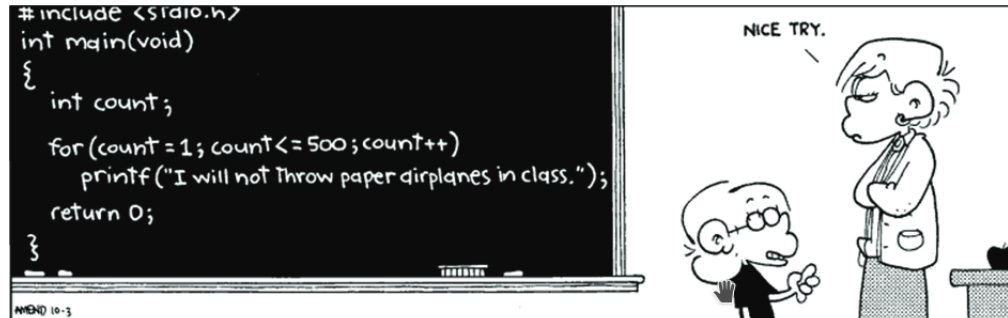


# What this course requires?

- Basic skills with Linux – shell, ssh, etc.



- Knowledge of C and C++ language



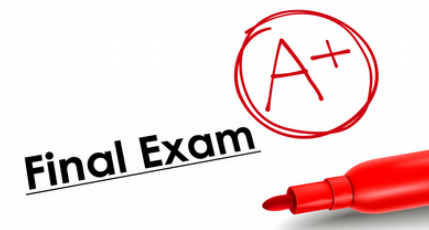
- Analytical thinking and opened mind





# How will we evaluate you?

- Labs
  - Each Individual Lab Assignment – **7 points**
  - Semestral work – **14 points**
- Lectures
  - Teoretical test (optional) – **10 points**
- Final Exam
  - Written exam – **45 points**
  - Oral exam – **10 points**
- **Pass criteria:**
  - **Assignment:** Everything submitted + at least 25 points
  - **Exam:** At least 25 points from written exam





# Parallel programming – the first cut

No questions?

Let's start with our business!







# Why should you care about it?

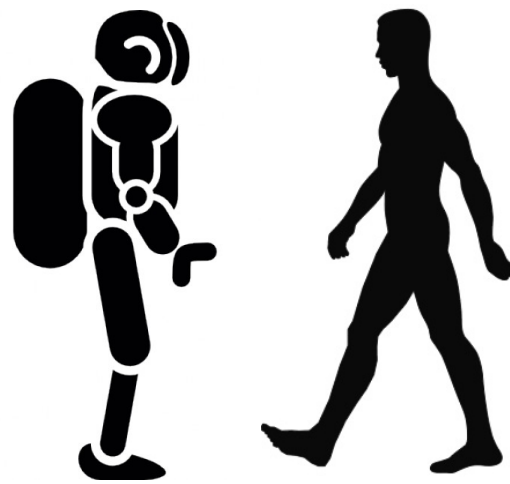
- Parallel computing is a dominant player in scientific and cluster computing. Why?
  - Moore law is reaching its limits
    - Increase in transistor density is limited
    - Memory access time has not been reduced at a rate comparable with processing speed
- How to get out of this trap?
  - Most promising 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.

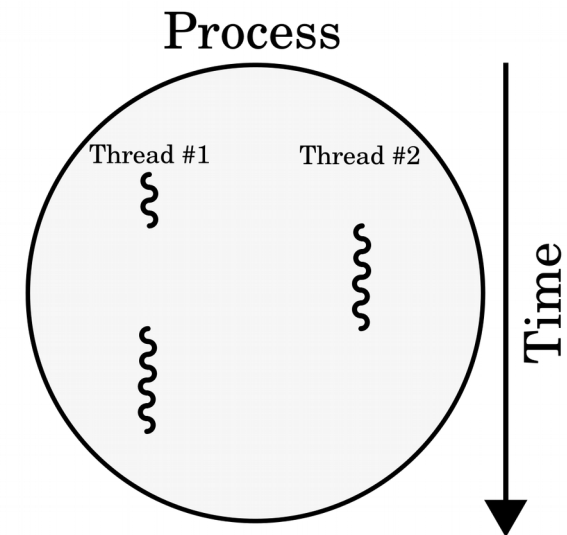


# Basics of Parallel programming theory

## Terms



- Program
  - Collection of instructions designed to perform a group of coordinated functions
- Process
  - Instance of a program that is being executed.
  - Multiple processes are typically independent
  - It has its own memory space.
- Thread
  - Sequence of instructions that is managed independently by system scheduler.
  - Subset of process
  - Multiple threads within process share the memory space.
- Task
  - Unit of execution

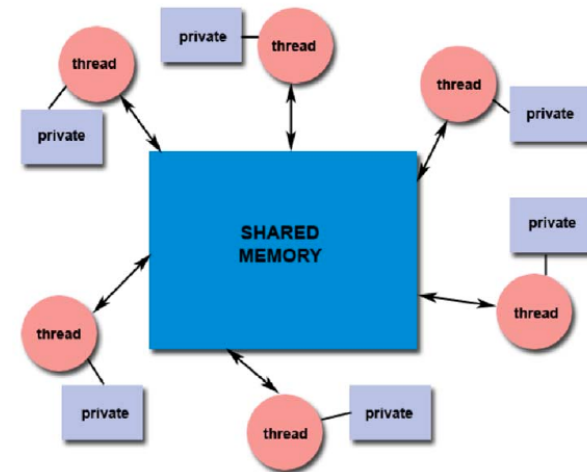


# Basics of Parallel programming theory

## Memory architectures

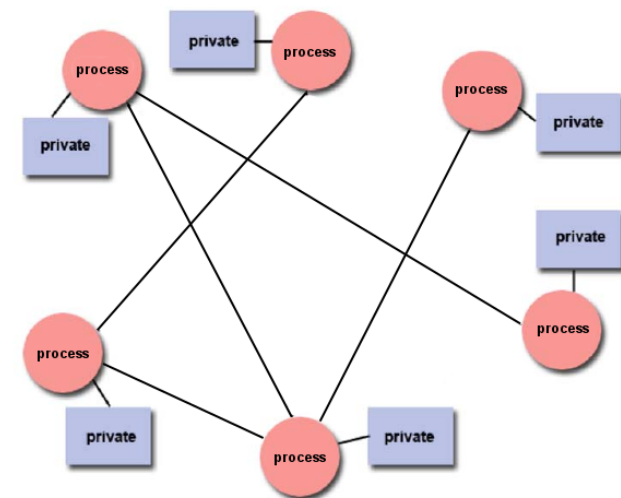
- **Shared memory**

- All functional units share the common memory space.
- When a functional unit share the value in the common memory space another functional unit can access this value.



- **Distributed memory**

- Each functional unit has its own private memory space.
- When two or more units need to share a value, they have to exchange this value by a message transmitted through the network.

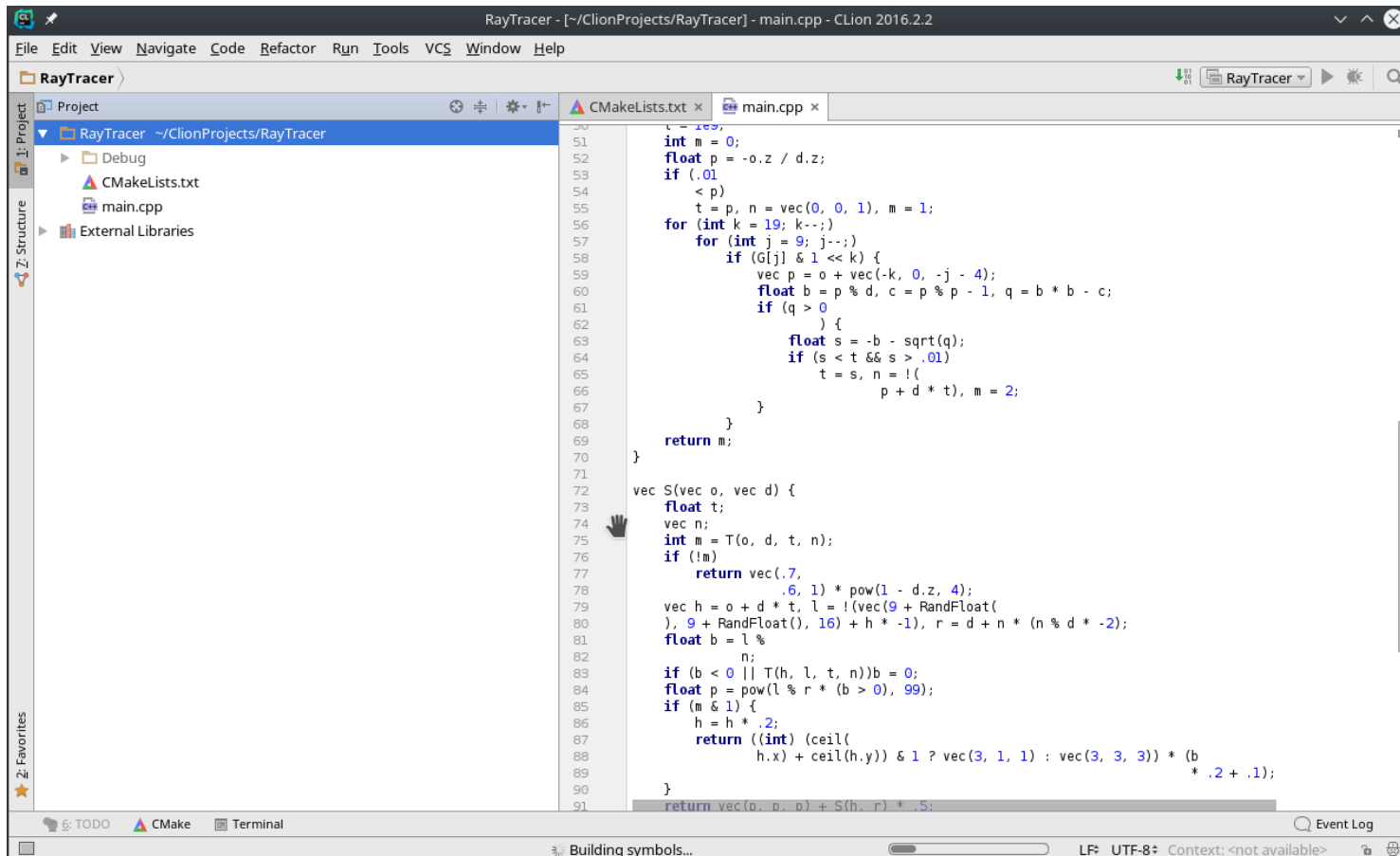




That was theory and now something practical!

## Clion IDE

- Licence server - <https://turnkey.felk.cvut.cz/>



```
RayTracer - [~/ClionProjects/RayTracer] - main.cpp - CLion 2016.2.2
File Edit View Navigate Code Refactor Run Tools VCS Window Help
RayTracer
Project
  RayTracer ~/ClionProjects/RayTracer
    Debug
    CMakeLists.txt
    main.cpp
    External Libraries
Structure
  Z: Favorites
main.cpp
51 int m = 0;
52 float p = -o.z / d.z;
53 if (.01
54 < p)
55 t = p, n = vec(0, 0, 1), m = 1;
56 for (int k = 19; k--;)
57   for (int j = 9; j--;)
58     if (G[j] & 1 << k) {
59       vec p = o + vec(-k, 0, -j - 4);
60       float b = p % d, c = p % p - 1, q = b * b - c;
61       if (q > 0
62           ) {
63         float s = -b - sqrt(q);
64         if (s < t && s > .01)
65           t = s, n = !(
66             p + d * t), m = 2;
67       }
68     }
69   return m;
70 }
71
72 vec S(vec o, vec d) {
73   float t;
74   vec n;
75   int m = T(o, d, t, n);
76   if (!m)
77     return vec(.7,
78               .6, 1) * pow(1 - d.z, 4);
79   vec h = o + d * t, l = !(vec(9 + RandFloat(
80     ), 9 + RandFloat(), 16) + h * -1), r = d + n * (n % d * -2);
81   float b = l %
82     n;
83   if (b < 0 || T(h, l, t, n)) b = 0;
84   float p = pow(l % r * (b > 0), 99);
85   if (m & 1) {
86     h = h * .2;
87     return ((int) (ceil(
88       h.x) + ceil(h.y)) & 1 ? vec(3, 1, 1) : vec(3, 3, 3)) * (b
89       * .2 + .1);
90   }
91   return vec(b, p, p) + S(h, r) * .5;
```



# Hello world for free

- Live example and walk-through
  - Create project, Build code, Run code, Debug code,
  - Code profiling – valgrind, callgrind

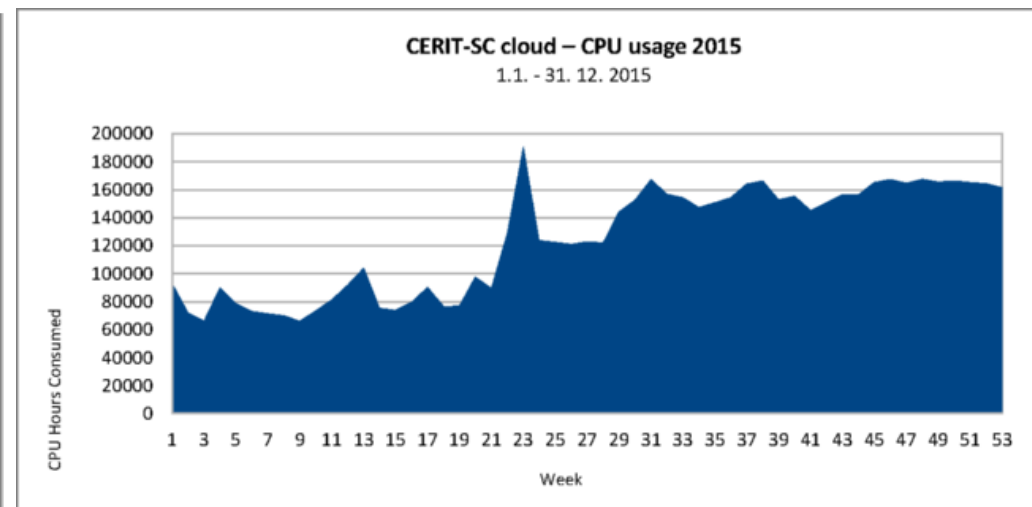
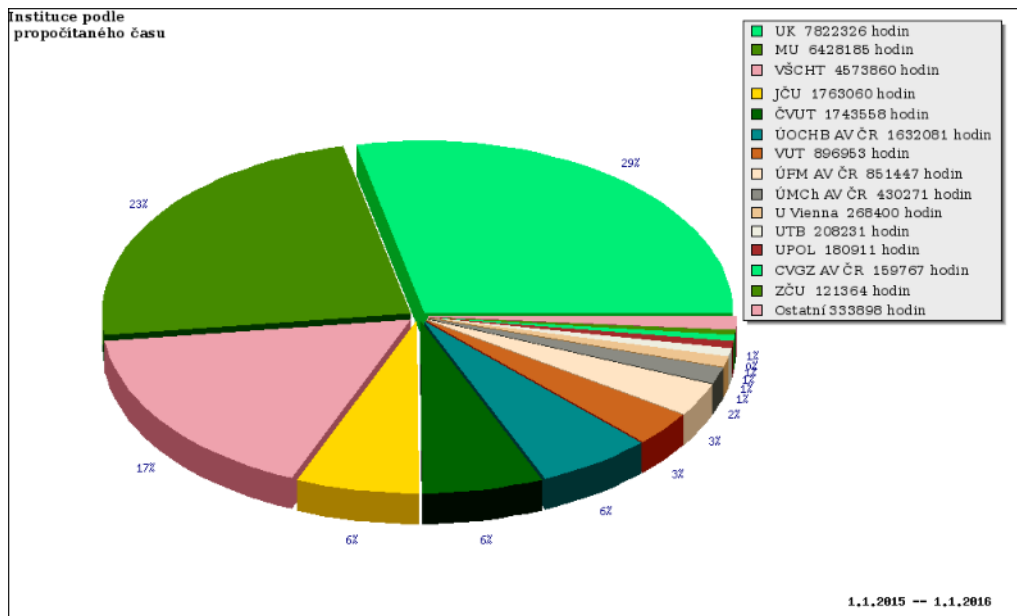
The screenshot shows a CMake IDE window titled "1. seminar - Uvod" with the following code in the editor:

```
1 #include <iostream>
2
3
4 int main() {
5     std::cout << "Hello, World!" << std::endl;
6     return 0;
7 }
```

The IDE interface includes a menu bar (File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help), a toolbar with a "Build All" button, and a sidebar with "Project" and "External Libraries" views. The status bar at the bottom shows "LF UTF-8+ Context: BohatyProjekt [D]" and "Event Log".

# 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

- How to sign up



MetaCentrum





# Metacentrum – How to run code?

- Example of the execution of a program
  - qsub
  - module
  - \$SCRATCHDIR
  - \$PSB\_O\_WORKDIR
- Running jobs in MetaCentrum
  - [https://wiki.metacentrum.cz/wiki/Running\\_jobs\\_in\\_scheduler](https://wiki.metacentrum.cz/wiki/Running_jobs_in_scheduler)
- Detailed description of the scheduling system
  - [https://wiki.metacentrum.cz/wiki/Scheduling\\_system\\_-\\_detailed\\_description](https://wiki.metacentrum.cz/wiki/Scheduling_system_-_detailed_description)
- Application modules
  - [https://wiki.metacentrum.cz/wiki/Application\\_modules](https://wiki.metacentrum.cz/wiki/Application_modules)



# That was nice, wasn't it?

Thank you for your attention...

