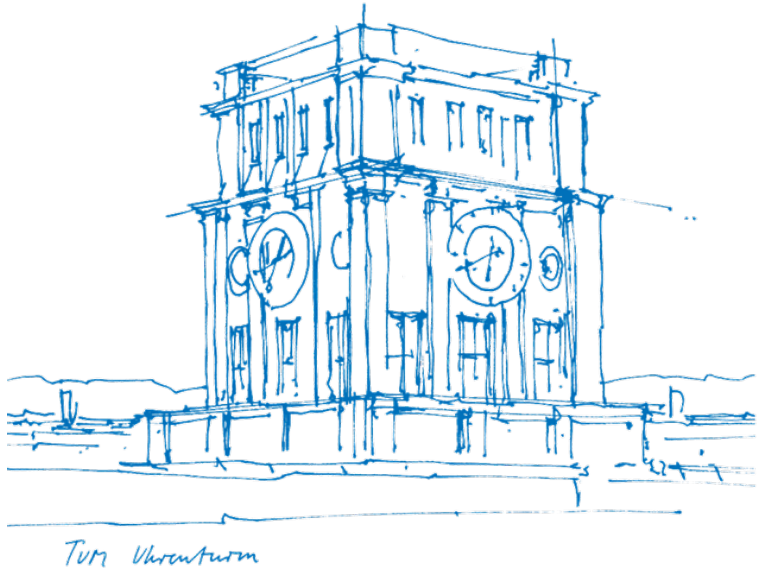
Organization

HPC - Algorithms and Applications

Alexander Pöppl
Technical University of Munich
Chair of Scientific Computing

October 24st 2016



References

- D. Kirk, W. Hwu:
  Programming Massively Parallel Processors, Morgan Kaufmann, 2010

HPC tutorial

- Purpose: Teach parallel computing
- Parallelization hierarchy: Distributed memory (MPI), shared memory (OpenMP, TBB), vectorization (SSE, AVX)
- We will focus on vectorization, but on a larger scale
- Use GPUs for parallel computing: GPGPU computing
- Domain-specific parallel programming language: CUDA

CUDA minimum requirements

- CUDA-capable Nvidia GPU:
  `https://developer.nvidia.com/cuda-gpus`
- (Linux) Check your hardware by typing `lspci | grep VGA` into a terminal
- Up-to-date graphics drivers (current version: 344.16)
- CUDA Toolkit (current version: 7.5)

CUDA minimum requirements (2)

- (Windows) Microsoft Windows XP or Windows Server 2008
- (Windows) Microsoft Visual Studio 2008 or a corresponding version of Microsoft Visual C++ Express
- (Windows) TUM students get free educational licenses for Microsoft products at `http://dreamspark.rbg.tum.de/`
- (Linux) Up-to-date Linux version (i.e. Ubuntu 12.04, OpenSUSE 13.1, etc.) with gcc 4.x
- (OS X) At least OS X 10.9, current clang (e.g. bundled with XCode).

CUDA installation steps

For details, refer to the Getting Started Guides for Windows/Linux/Mac on `https://developer.nvidia.com/cuda-downloads`

- Download CUDA toolkit from here: `https://developer.nvidia.com/cuda-downloads`
- (Linux) Execute the `.run` file or install debian package `.deb`, update package list `sudo apt-get update` and install `sudo apt-get install cuda`

Compiling CUDA-code

- Open a terminal
- Add the the CUDA binary path to your `$PATH`.
  - Linux: `export PATH=$PATH:/usr/local/cuda/bin`
  - OS X: `export PATH=$PATH:/Developer/NVIDIA/CUDA-7.5/bin`
- To include the CUDA library path, type
  `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH: +`
  - Linux x86: `/usr/local/cuda/lib`
  - Linux x86-64: `/usr/local/cuda/lib64`
  - OS X: `/Developer/NVIDIA/CUDA-7.5/lob`
- Now, compile your code using `nvcc <source files.cu>`

Remote compilation on MAC Cluster

What if no CUDA capable device is available?

- Fill the form for an account on our MAC cluster
  `http://www.mac.tum.de/wiki/index.php/MAC_Cluster`
- (Linux) Open an ssh connection via
  `ssh -X mac-login-intel.tum-mac.cos.lrz.de`
- Change your password after the first login using `passwd`

Remote compilation on MAC Cluster

- The cluster uses a module System to manage the provided compilers, runtimes and libraries.
  - `module load gcc/4.8` to load the compatible GCC
  - `module load cuda/6.5` to load CUDA
- The cluster uses the `SLURM` scheduler (`https://www.lrz.de/services/compute/linux-cluster/batch_parallel/`) to allocate job time.
- Start an interactive shell on the NVidia nodes by executing `salloc --ntasks=1 --partition=nvd`
- Compile and run your code. When you are done, type `exit`
- There are only 4 Nvidia nodes. Allocation may take time, sometimes all nodes are blocked by other users.
- So be nice and don't block a node for too long (max. 30 min)! Code and compile offline, test online!