Lessons Learned 1, 1&2 (2021)
Submitted: 28.05.2021
Accepted: 09.06.2021
DOI: https://doi.org/10.25369/ll.v1i1/2.17
ISSN: 2749-1293 (Print); 2749-1307 (Online)

Virtual PC pools for computer practicals using the example of materials science

S. Kampmann[1], D. Bodesheim[1], A. Croy[1], T. Schied[1], R. Gutierrez[1], A. Dianat[1], G. Cuniberti[1,2,3,*]

[1] *Chair of Materials Science and Nanotechnology, Institute of Materials Science, Faculty of Mechanical Engineering, Dresden University of Technology*
[2] *Dresden Center for Computational Materials Science (DCMS), Dresden University of Technology*
[3] *Dresden Center for Intelligent Materials (DCIM), TU Dresden*

Abstract

Computerpraktika stellen einen wichtigen Bestandteil vieler Lehrveranstaltungen dar, welche die Grundlagen und Details von computergestützten Methoden vermitteln sollen. In den Materialwissenschaften spielen solche Methoden eine zunehmend wichtige Rolle. Typischerweise setzen die Praktika eine physische Präsenz in den PC Pools voraus, u.a. da eine Vielzahl von verschiedenen Programmen lokal installiert und bereitgestellt werden muss. Um Computerpraktika auch in der Online-Lehre vollumfänglich und weitestgehend unabhängig von den Gegebenheiten der Studierenden einsetzen zu können, wurde im Wintersemester 2020/21 ein virtueller PC Pool auf Basis von virtuellen Maschinen mit Web-basiertem Zugang eingerichtet. Dieser virtuelle PC Pool wurde in verschiedenen Lehrveranstaltungen erfolgreich eingesetzt und kann auch bei hybriden Lehrformaten in verschiedenen Disziplinen verwendet werden.

Computer practicals are an important part of many courses that are designed to teach the basics and details of computer-based methods. In the materials sciences, such methods play an increasingly important role. Typically, the practical courses require a physical presence in the PC pools, among other things because a large number of different programmes have to be installed and made available locally. In order to be able to use computer practicals fully and as far as possible independently of the students' circumstances in online teaching, a virtual PC pool based on virtual machines with web-based access was set up in the winter semester 2020/21. This virtual PC pool has been successfully used in various courses and can also be used in hybrid teaching formats in various disciplines.

*Corresponding author: gianaurelio.cuniberti@tu-dresden.de
This article was originally submitted in German.

1. Introduction

Most courses related to computer simulations include practicals through which students are expected to try out and apply the simulation methods. The computer practicals typically take place in PC pools, where the necessary software is available and questions can be answered by the practical instructor directly on site. Outside the pools, students can in principle install the relevant software on their own PCs, but this is sometimes very difficult due to the heterogeneity of the different hardware and operating systems. A virtual PC pool, in which the required environment is provided on the server side, offers a feasible solution for digital teaching. But here, too, various aspects must be taken into account, especially with regard to secure and low-threshold access [1-3].

In this article, the establishment and use of a virtual PC pool for materials science is described using the example of the courses *Computer Simulation in Materials Science, Computational Methods II*, *Concepts of Molecular Modelling, Computational Materials Science*: *Molecular Dynamics and Continuum Methods,* which are offered to students of *Materials Science* in the 8th and 9th semesters, as well as to students of various Master's programmes (*Computational Modelling and Simulation, Nanobiophysics*, *Organic and Molecular Electronics,* and *Physics*) at the TU Dresden. The solution presented here was developed and implemented with the help of the Centre for Information Services and High Performance Computing (ZIH) at TU Dresden.

2. Requirements

As mentioned at the beginning, the practical courses are intended to supplement the lecture material with own experiments and to deepen it with the help of projects. For this purpose, the students are given tasks which are to be processed with the software provided. The simulation results must then be evaluated and interpreted. Within the framework of the practical courses related to materials science, various (free) software is used for materials simulation under Linux. Different length scales of the materials to be examined must be covered and handled with the software used (see Fig. 1 and Tab. 1).
These programmes should also be available to all students in the virtual PC pool. Additional administration effort through installation support on different hardware platforms and operating systems should be avoided as much as possible. Ideally, students should also have full access to programmes, data and computer capacities outside of course times. In addition, care should be taken to ensure that students are provided with identical virtual desktop environments. Furthermore, data protection must be guaranteed overall.

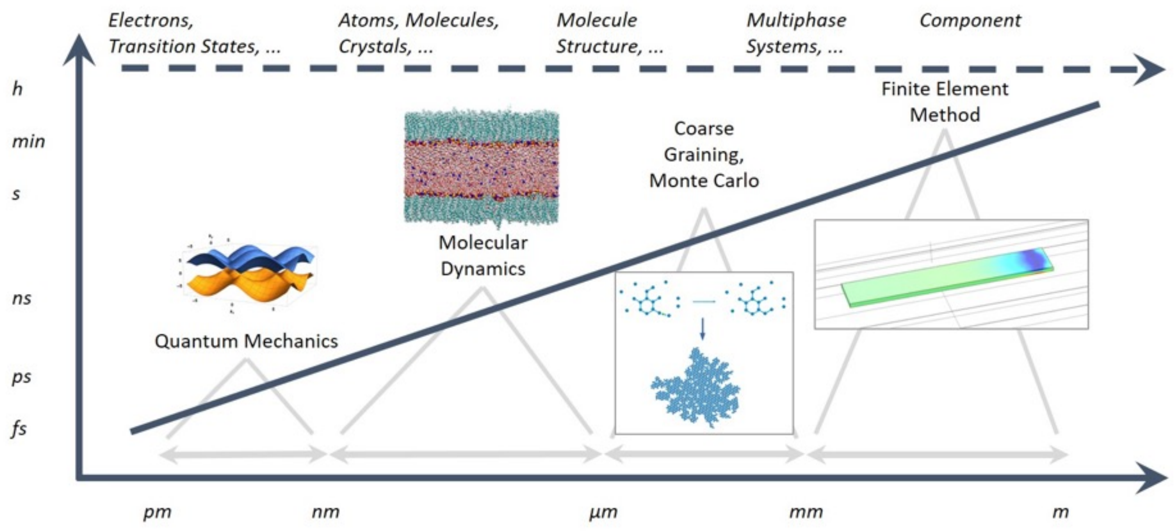


Fig. 1: Overview of methods for describing different time and length scales.

Tab. 1: Software programmes used in the computer practicals and their area of application (scale)

Software	Description	**Atomistic**	**Micro**	**Meso**	**Macro**
Jupyter Notebook [15]	Evaluation and visualisation	☑	☑	☑	☑
OVITO [12]	3D visualisation and analysis	☑	☑	☒	☒
VMD [16]	3D visualisation and analysis	☑	☑	☒	☒
LAMMPS [11]	Molecular dynamics simulations	☑	☑	☒	☒
DFTB+ [13]	Electron structure calculations	☑	☒	☒	☒
Avogadro 2 [17]	Visualisation of molecules and their manipulation	☑	☒	☒	☒
COMSOL [14]	Finite elements software	☒	☒	☑	☑

3. Implementation

For the implementation of the virtual PC pool, which comprised a capacity of 80 virtual machines, a solution was devised based on the following components:

- The virtual machines were operated in a cloud environment at ZIH. The configuration and monitoring of the virtual machines is done via a web interface, which is also provided by ZIH.
- Each virtual machine was based on a predefined installation that included all software. *Ubuntu 20.04* [4] was used as the operating system.
- *Turbo-VNC* was used as the Virtual Network Computing (VNC) server [5]. The VNC server enables remote access to the virtual machine.
- Access was realised via a web interface using the software *noVNC* [6] and secured by Transport Layer Security (TLS). *Xfce* [7] was chosen as the resource-saving desktop interface (see Fig. 2).
- The virtual machines were each assigned a fixed domain.

From the students' point of view, there are a number of advantages with this solution. Each user has their own individual virtual environment. Access is provided via the browser and is therefore largely independent of hardware and operating systems. Even those previously inexperienced with IT can easily gain access on a variety of end devices. Transmission is encrypted and access is password-protected. All software is pre-installed and tested, which means that identical software is available to all participants. Licensed programmes can also be used without local installation, as the range of IP addresses is limited. Data can be provided and accessed via a cloud solution or through network drives.

From an administrator's point of view, the solution found also offers several advantages.

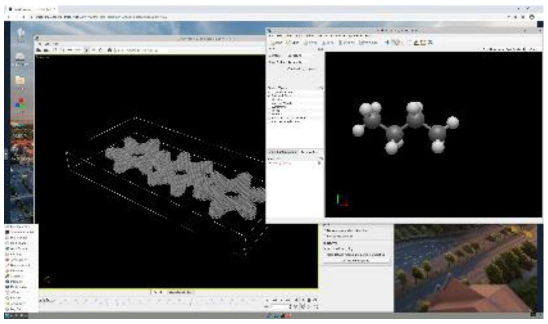


Fig. 2: Screenshot of the virtual desktop. In the foreground are visualisations with OVITO and Avogadro 2.

The software used in the internship can be managed and tested in a centralised manner as far as possible. The resources can be dynamically adapted to the demand and are in principle always available. Access and administration can be done graphically or via terminal. For central administration, the software *Cluster-SSH* [8] can be used to access all virtual machines simultaneously and execute commands. In this way, randomised passwords created for web access were assigned to the individual machines. This also makes it easy to reinstall software or store files. In addition, the virtual machines can easily be returned to their original state at the end of the semester.

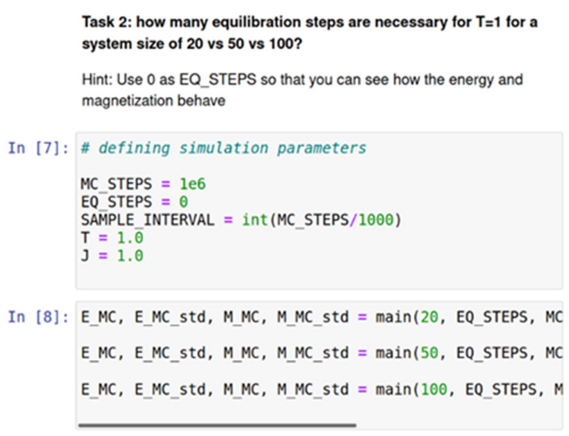
4. Implementation

A total of 80 students took part in the courses *Computer Simulation in Materials Science*, *Computational Methods* and *Concepts of Molecular Modelling* in the winter semester 2020/21. Each participant was assigned their own virtual machine. In addition, the staff supervising the practical courses were each provided with a virtual machine identical to the other virtual machines. During the online practical sessions, the students were able to follow the work steps on the virtual machine's graphic interface with the help of the screen transfer function of common video conferencing systems, such as Big Blue Button or Zoom. Due to the uniform installation, any errors that occurred could be quickly narrowed down.

The OPAL learning platform [9] and the cloud service [10] provided by the TU Dresden were used to make files relevant to the practical course available. Access to these resources on the virtual machine is possible as usual via the browser.

For the *Concepts of Molecular Modelling* lecture in particular, the easy and unrestricted access was very valuable, as there were students from the international Master's programmes who were not in Germany and thus in many cases only had asynchronous access to programmes and data.

In the practicals of this course, students should learn how to use molecular dynamics programs such as *LAMMPS* [11], visualisation software such as *OVITO* [12] and the Python programming language to analyse data and perform Monte Carlo simulations (see Fig. 3). The skills learned were applied at the end of the practical in a small project on molecular dynamics or Monte Carlo carried out by the students. The focus of the *Computational Methods* course is on methods for calculating the electronic structure, which are practically tested using the example of *DFTB+* [13]. The *Density Functional Based Tight Binding* (DFTB) method is an approximation of the density functional theory, which allows the consideration of very large systems in particular. In the practical sessions of the course *Continuum Methods,* basic mass transfer phenomena as well as thermal and mechanical problems are dealt with. At the same time, the handling of the widely used (licence-required) finite element software *COMSOL* [14] is taught by means of calculations of simplified examples of such problems (see Fig. 4). Particularly when using software subject to licensing, the virtual PC pool offers a simple solution for providing access to all students. The understanding of the approach to material science problems and their processing with the help of continuum methods ultimately enables the participants to independently develop solutions to problems of a different nature that are not part of the scope of the course.



Task 2: how many equilibration steps are necessary for T=1 for a system size of 20 vs 50 vs 100?

Hint: Use 0 as EQ_STEPS so that you can see how the energy and magnetization behave

```
In [7]: # defining simulation parameters

        MC_STEPS = 1e6
        EQ_STEPS = 0
        SAMPLE_INTERVAL = int(MC_STEPS/1000)
        T = 1.0
        J = 1.0

In [8]: E_MC, E_MC_std, M_MC, M_MC_std = main(20, EQ_STEPS, MC

        E_MC, E_MC_std, M_MC, M_MC_std = main(50, EQ_STEPS, MC

        E_MC, E_MC_std, M_MC, M_MC_std = main(100, EQ_STEPS, M
```

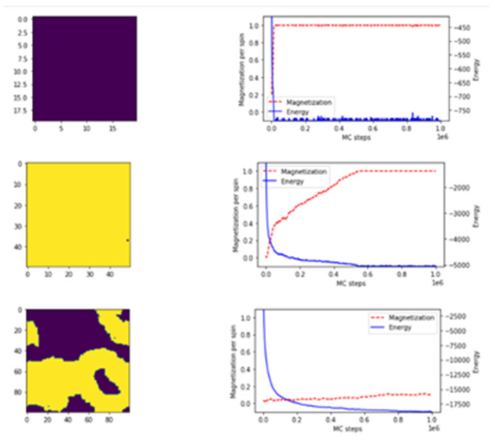


Fig. 3: Extract from a Jupyter notebook on Monte Carlo simulations.

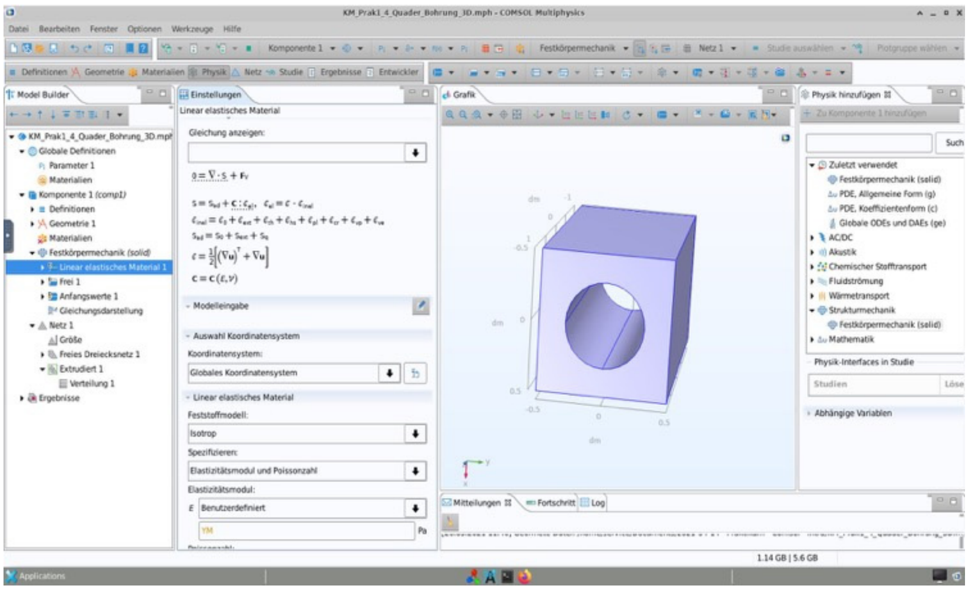


Fig. 4: Screenshot of a calculation example in the finite element software COMSOL [14] from the practical session *of the Continuum Methods course.*

In the following two subchapters, the structure and procedure of a practical unit are presented as examples. First, the use of Python will be discussed and then a question related to molecular dynamics will be described.

4.1 Practical 1: Python

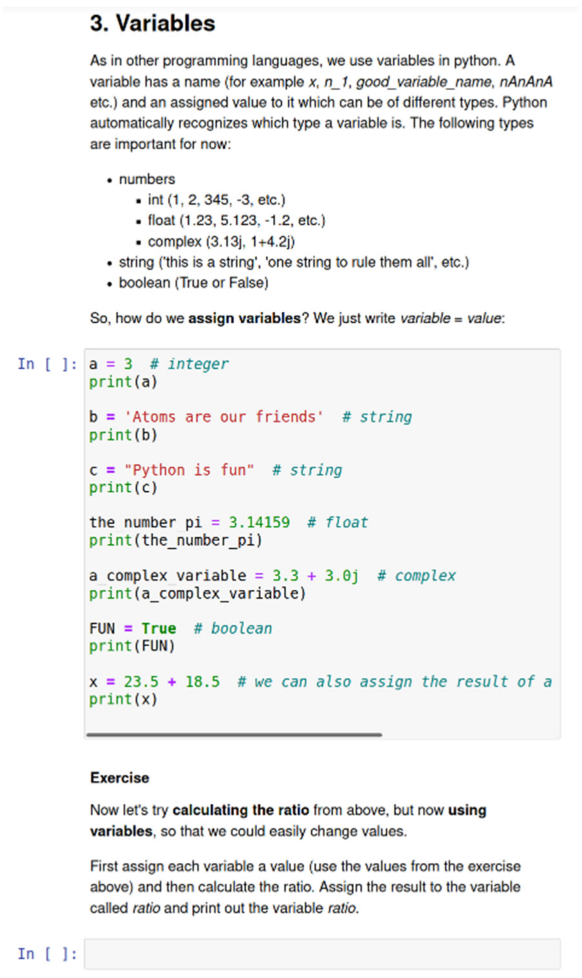
At the beginning of the semester, an introduction to Python was given in order to make it easy for students without any previous knowledge of programming to get started. The aim was for the students to understand simple Python code and also to be able to create scripts independently, for example for data analysis. Therefore, the courses focused on the greatest possible participation and personal work of the students in order to guarantee the greatest possible learning success.

The tutorials were held online via the Zoom platform. The students had to solve programming tasks independently during the tutorial, which were then discussed. Surveys were regularly conducted to check the current progress of the students in completing the tasks in order to determine when the tasks should be discussed. Furthermore, the tutorials were recorded so that asynchronous learning was possible without any problems if, for example, students were in a different time zone or had other commitments.

The programming tasks were created in so-called *Jupyter notebooks* [15]. This is an environment in which parts of code can be executed section by section and text modules can also be inserted (see Fig. 5). These Jupyter notebooks were designed in such a way that precise instructions and explanations were already included and small programming tasks were to be solved. The students were then expected to solve the tasks in the notebooks on their own during the course, while tutors were available to answer questions in the Zoom conference. The notebooks could be downloaded and worked on in the virtual machines via OPAL.

In addition to the introductory courses on Python, there was also a Python-based tutorial on Monte Carlo simulations (see Fig. 3). This tutorial was held with Jupyter notebooks by also including explanations and tasks that were worked on by the students.

At the end of the semester, the students worked on their own small project, on which they prepared a report. Since the virtual machines could be used at any time from home via the browser, all students were able to carry out this project independently from home.



3. Variables

As in other programming languages, we use variables in python. A variable has a name (for example *x*, *n_1*, *good_variable_name*, *nAnAnA* etc.) and an assigned value to it which can be of different types. Python automatically recognizes which type a variable is. The following types are important for now:

- numbers
  - int (1, 2, 345, -3, etc.)
  - float (1.23, 5.123, -1.2, etc.)
  - complex (3.13j, 1+4.2j)
- string ('this is a string', 'one string to rule them all', etc.)
- boolean (True or False)

So, how do we **assign variables**? We just write *variable = value*:

```
In [ ]: a = 3  # integer
        print(a)

        b = 'Atoms are our friends'  # string
        print(b)

        c = "Python is fun"  # string
        print(c)

        the_number_pi = 3.14159  # float
        print(the_number_pi)

        a_complex_variable = 3.3 + 3.0j  # complex
        print(a_complex_variable)

        FUN = True  # boolean
        print(FUN)

        x = 23.5 + 18.5  # we can also assign the result of a
        print(x)
```

Exercise

Now let's try **calculating the ratio** from above, but now **using variables**, so that we could easily change values.

First assign each variable a value (use the values from the exercise above) and then calculate the ratio. Assign the result to the variable called *ratio* and print out the variable *ratio*.

```
In [ ]:
```

Fig. 5: Extract from a Jupyter notebook introducing programming with Python.

4.2 Practical 2: Molecular Dynamics

Molecular dynamics simulations are an important pillar of computational materials science. During the practicals, the students solved given tasks using annotated input scripts. One such task is e.g.

1) Generate an ideal gold nanowire (kfz, L = 8 nm, R = 1.2 nm).
2) Calculate the dependence of the pressure on the strain in the z-direction.
3) Calculate the dependence of the potential energy on the strain during the elongation process and calculate the elongation at break for the nanowire.

The execution of the corresponding script takes only a few minutes. Afterwards, the behaviour of the Nanowire can be visualised with the help of *OVITO* and thus the critical strain can be found (see Fig. 6). The evaluation of the compression-strain and energy-strain curves can then be done, for example, in a Jupyter notebook or by separate software. A total of four practical courses took place, in each of which a task was worked on as described by way of example. Thematically, the following topics were dealt with, among others: Generation and deformation of carbon structures; heating and cooling processes of nanostructures.

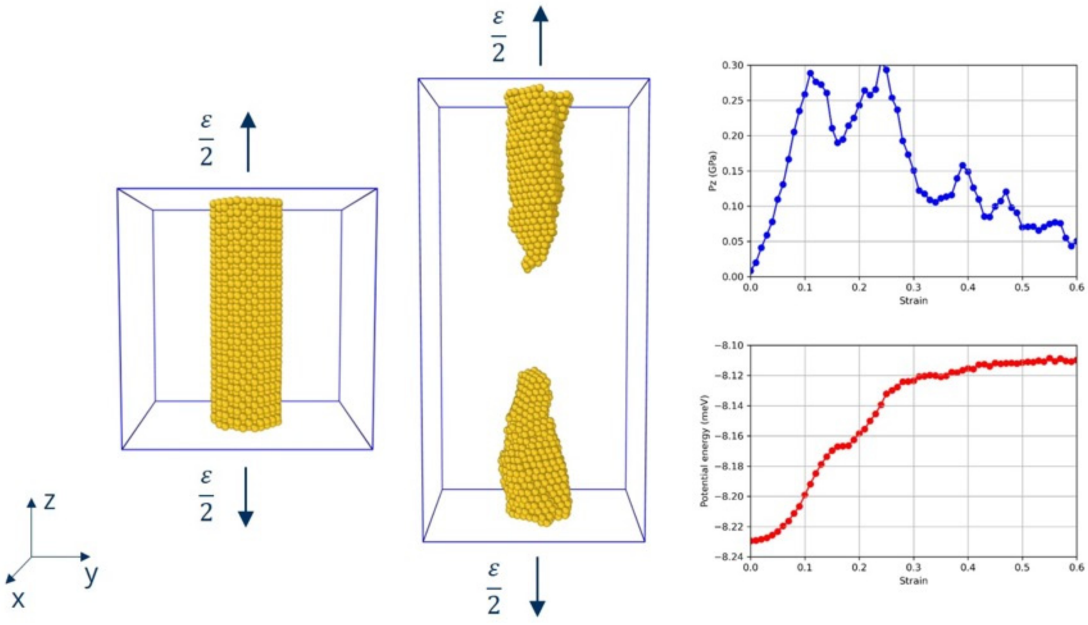


Fig. 6: Visualisation of a gold nanowire in OVITO a) without strain, and b) with 60 percent strain. c) Mechanical stress (top) and potential energy (bottom) as a function of elongation.

5. Conclusion and outlook

All in all, the solution presented here is a convenient and feasible option for conducting computer practicals in the field of hybrid teaching as well. Through access via the virtual machines, the practical courses could be carried out without restrictions. The standardised provision of software and the secure, unproblematic access at any time via the web interface should be emphasised. In future, it will therefore be possible to make the students' tasks and solution progress available in parallel in the computer pools and the virtual machines. In this way, flexible access for students to the practical resources can be guaranteed and individual teaching support strengthened at the same time.

Acknowledgements

We would especially like to thank Mr. Holger Mickler, Mr. Christopher Mosch and Mr. Matthias Jurenz from the ZIH of the TU Dresden for their very helpful support during the concept finding, installation and error analysis.

Literature

[1] Luo, F., Gu, C. and Li, X. 2015. Constructing a virtual computer laboratory based on Open Stack. *10th International Conference on Computer Science & Education (ICCSE). doi: 10.1109/ICCSE.2015.7250353*

[2] Bastidas, C. E. C. 2011. Enabling remote access to computer networking laboratories for distance education. *2011 Frontiers in Education Conference (FIE). doi: 10.1109/FIE.2011.6142731*

[3] Hu, X., Le, H., Bourgeois, A. G. and Pan, Y. 2018. Collaborative Learning in Cloud-based Virtual Computer Labs. *2018 IEEE Frontiers in Education Conference (FIE). doi: 10.1109/FIE.2018.8659018*

[4] https://releases.ubuntu.com/20.04/

[5] https://www.turbovnc.org/

[6] https://novnc.com/

[7] https://www.xfce.org/

[8] https://github.com/duncs/clusterssh

[9] https://bildungsportal.sachsen.de/opal

[10] https://tu-dresden.de/zih/dienste/service-katalog/zusammenarbeiten-und-forschen/datenaustausch/cloudstore

[11] https://lammps.sandia.gov/

[12] https://www.ovito.org/

[13] https://dftbplus.org/

[14] https://www.comsol.de/

[15] https://jupyter.org/

[16] http://www.ks.uiuc.edu/Research/vmd/

[17] https://www.openchemistry.org/projects/avogadro2/