ETH zürich

Working with (big) data

Data Science for Public Policy

Christoph Goessmann
Law, Economics, and Data Science Group (Prof. Elliott Ash)

Course: 860-0033-00L Data Science for Public Policy: From Econometrics to AI, Spring 2024

29 February 2024

Survey

ETH Edu App

Web app

https://eduapp-app1.ethz.ch/

iOS

Android

Paper Presentations

- **From 14 March to 23 May:**
  - 1-2 presentations per lecture
- **2-4 students in one group.**
- **Choose a paper from our list OR propose one in coordination with us.**
- **Sign up by 11 March.**

Course Projects

- **Either a research paper or a web app.**
- **Work on something that you are passionate about!**
- Choose a *type of project* and *topic* with instructors **by 9 April:**
  - Research Paper
    - Sergio Galletta (sergio.galletta@gess.ethz.ch)
  - Web App
    - Christoph Goessmann (christoph.goessmann@gess.ethz.ch)

Introductions

- Name, Department / Uni
- What's your background?
- What do you want to:
  - learn?
  - work on?
- Pain points when working with data so far?

Tentative Outline

Technical Part

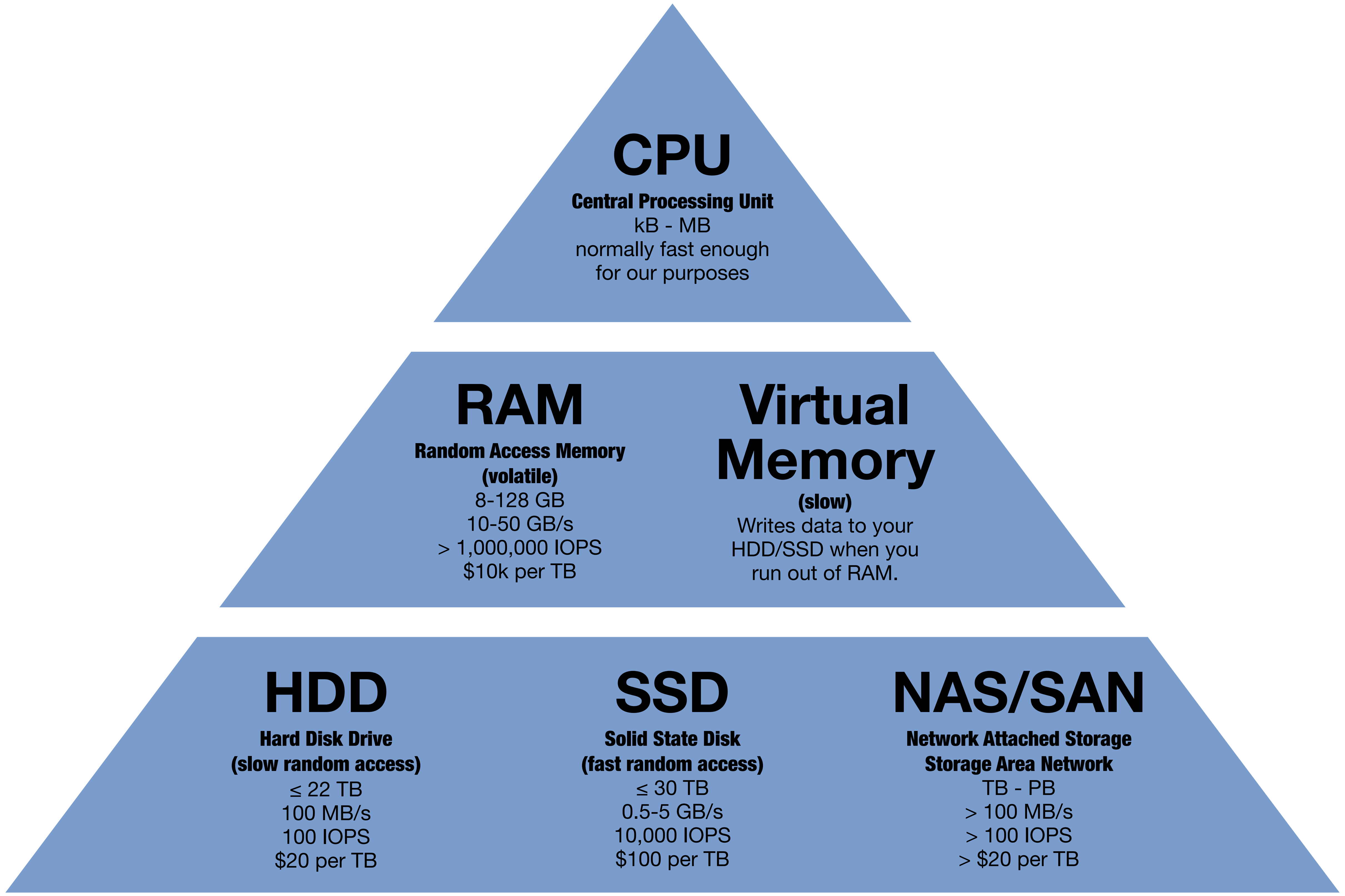
- **How to efficiently work with computers?**
  - Computer architecture (I/O, CPU, GPU)
  - Controlling a computer (command line, ssh)
  - Shapes and forms of data (text vs. binary, compression)
  - Reproducibility and automation (virtual environments, logging, git, etc.)
  - Data acquisition (scraping)
- **APIs and web apps**
- **Machine Learning**

Questions?

- Interrupt with questions / comments anytime!
- Happy to adapt the lectures to your needs.

Computer Architecture

Typical data flow



CPU bound (CPU load ~ 100%)

- increase number of CPUs/cores (parallelization)
- use GPUs instead of CPUs
- change algorithm (e.g., profiling)

I/O bound (CPU load << 100%)

- reduce I/O operations
- switch to faster memory

Computer Architecture

I/O bound vs CPU bound processes

I/O bound

Somewhere in between

CPU bound

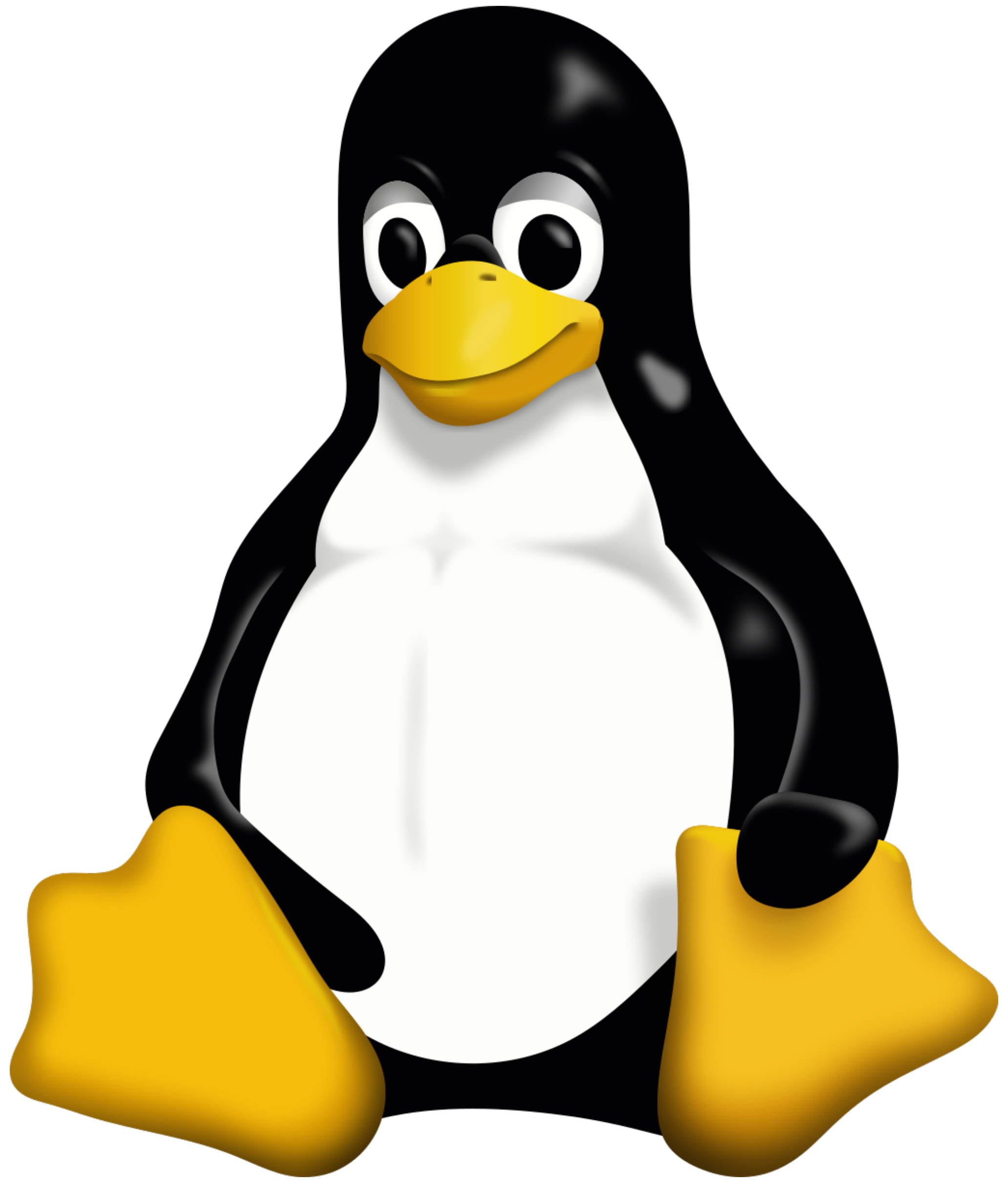
Computer Architecture

Common data types

Type	Size	Range
boolean	1 byte	0,1
smallint	2 bytes	-32,768 to +32,767
int	4 bytes	-2,147,483,648 to +2,147,483,647
bigint	8 bytes	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807
real / single precision float	4 bytes	6 decimal digits precision
double precision float	8 bytes	15 decimal digits precision

Computer Architecture

Operating System



Linux/Unix dominant

- 100% of all TOP500 supercomputers [1]
- > 80% of public servers (web, mail, DNS, etc.) on the internet [2]
- Some packages (e.g., CuPy, PyTorch ROCm) are only fully supported on Linux

[1] https://top500.org/statistics/overtime/, accessed 28 February 2023
[2] https://w3techs.com/, accessed 28 February 2023

Command Line

Basic commands

$ **pwd** (print working directory)

$ **cd *path*** (change directory)

$ **mkdir *newd*** (make directory)

$ **ll / ls -al** (list contents of working directory)

$ **cat *file*** (display content of file)

$ **head -*n* *file*** (display first n lines of file)

$ **tail -*n* *file*** (display last n lines of file)

$ **rm *file*** (remove/delete file)

$ ***com* > *file*** (write output of command com to file)

$ **man *command*** (display command's manual)

$ **CTRL + C** (abort command)

$ **CTRL + D** (end of stream)

Command Line

Text editor

Create new file / edit existing one

`$ vim` **`file.txt`**

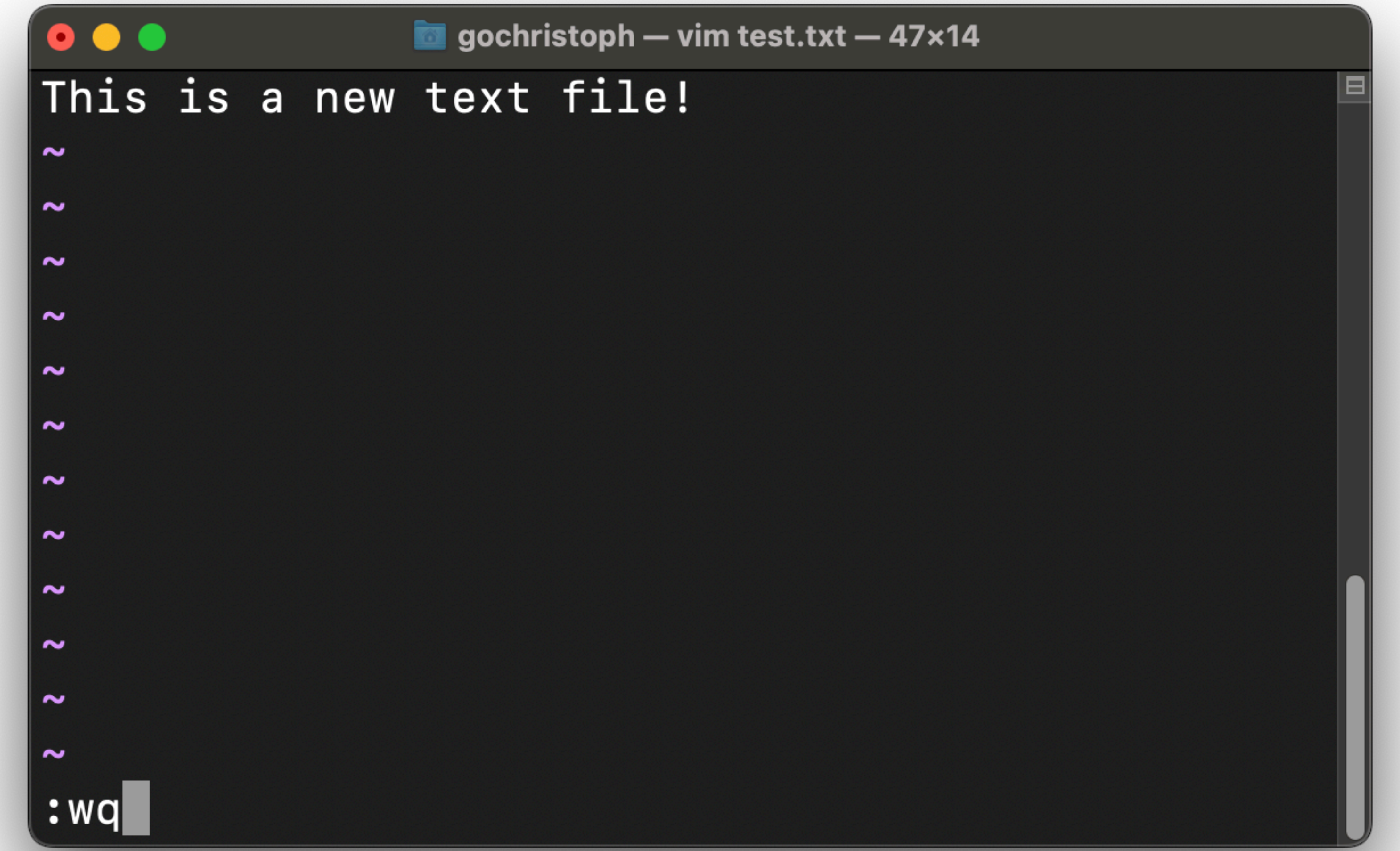
Command mode (default at start, invoke with ESC):

Switch to insert mode: **`i`**

Save: **`:w`** *`+Enter`*

Save and quit: **`:wq`** *`+Enter`*

Quit without saving: **`:q!`** *`+Enter`*



Command Line

Monitoring CPU load & profiling

Task manager

```
$ htop / top
```

Average CPU load of your script

$ time python **my_program.py** *(local)*

$ myjobs –j **jobid** *(ETH Euler cluster)*

Profiling with cProfile and SnakeViz

$ python –m cProfile –o **my_program.prof my_program.py**

$ snakeviz **my_program.prof**

Command Line

Virtual environments wit `pipenv`

Install pipenv

`$ pip install pipenv`

Change to project directory and install a package

`$ cd myproject`

`$ pipenv install` ***`package`***

List of packages and dependencies is written to Pipfile/Pipfile.lock.

Whenever you are in the project directory, you have two options to **use the virtual environment:**

`$ pipenv shell`

`$ pipenv run python` ***`myscript.py`***

Use `pyenv` for managing different python versions on the same machine.

Hands-on

Example

10 million documents

- Each document *i* has a 128-dim. feature vector $a_{i,*}$ (e.g., LDA topics).
- Want to calculate all cosine similarities between documents *i* and *j*:

$$s_{i,j} = \frac{a_{i,*} \cdot a_{j,*}}{\|a_{i,*}\|\|a_{j,*}\|}$$

(naively corresponds to matrix multiplication of 1e7 x 128 matrix with its transpose)

Feature matrix (1e7 x 128):

$$(a_{i,j}) = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,128} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,128} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,128} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{10^7,1} & a_{10^7,2} & a_{10^7,3} & \cdots & a_{10^7,128} \end{bmatrix}$$

Similarity matrix (1e7 x 1e7):

$$(s_{i,j}) = (a_{i,j})(a_{i,j})^T$$

Hands-on

First considerations

Document vector (1 x 128):

$$a_{i,*} = [a_{i,1} \quad a_{i,2} \quad a_{i,3} \quad \cdots \quad a_{i,128}]$$

→ size = 1e7 * 8 byte = **80 MB**

Feature matrix (1e7 x 128):

$$(a_{i,j}) = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,128} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,128} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,128} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{10^7,1} & a_{10^7,2} & a_{10^7,3} & \cdots & a_{10^7,128} \end{bmatrix}$$

→ size = 1e7 * 128 * 8 byte = **10.2 GB**

Similarity matrix (1e7 x 1e7):

$$(s_{i,j}) = (a_{i,j})(a_{i,j})^T$$

→ size = 1e7 * 128 * 8 byte = 800 TB

Is it realistic to calculate all cosine similarities?

Hands-on

1. Create a virtual environment

```
$ pip install pipenv
$ mkdir session1
$ cd session1
$ pipenv install numpy
$ pipenv install snakeviz
$ pipenv shell
```

2. Write a python script (matrix_math_test.py) that:

- creates a random feature matrix
- computes the cosine similarity between the feature matrix and 100 randomly drawn rows

Hands-on

3. Time your script:

$ time python **my_program.py**

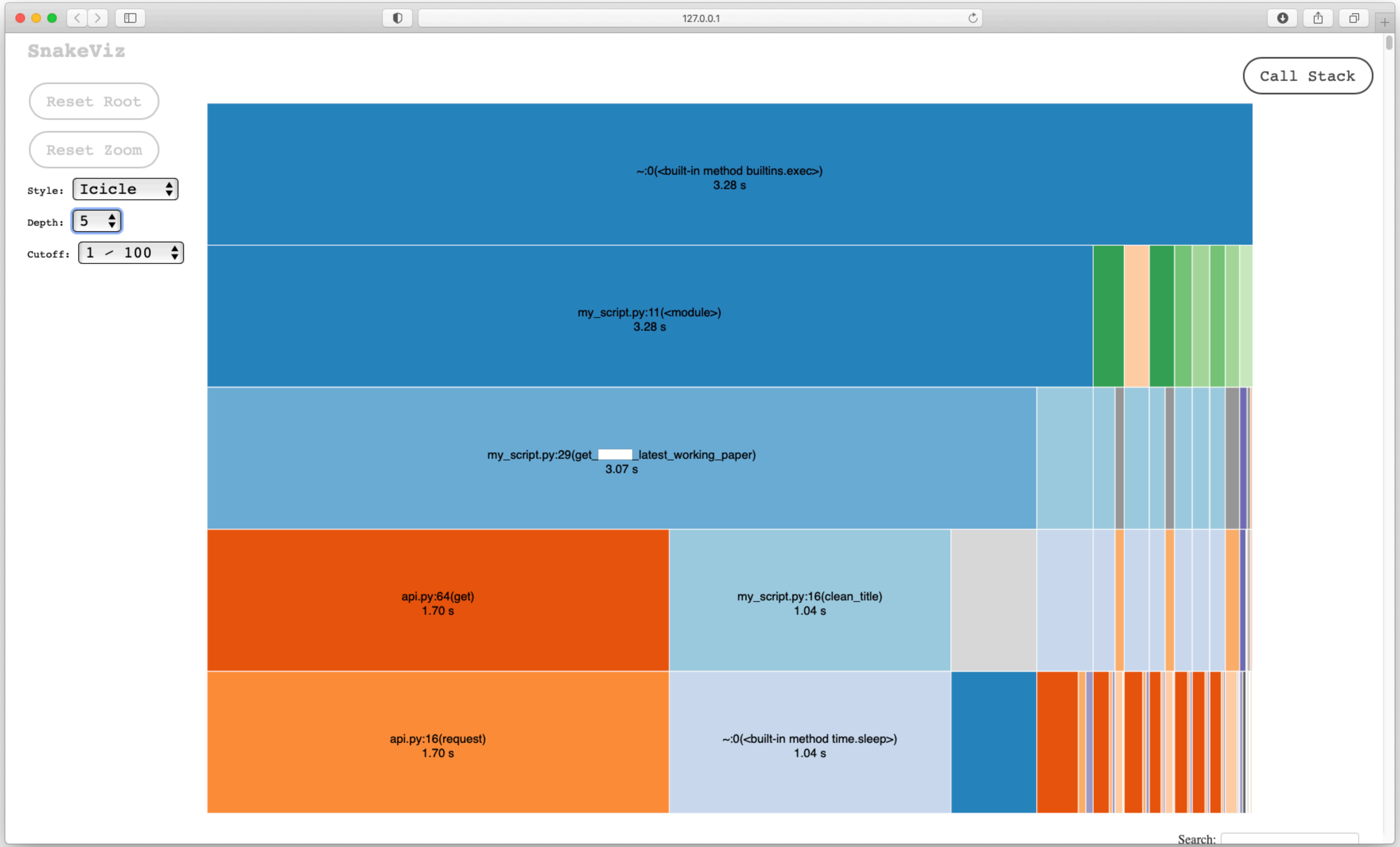
Extra: `export OPENBLAS_NUM_THREADS=`**`1/0`**

4. Profile your script with cProfile and SnakeViz

$ python -m cProfile -o **my_program.prof my_program.py**

$ snakeviz **my_program.prof**

Hands-on


127.0.0.1
SnakeViz
Reset Root
Reset Zoom
Style: Icicle
Depth: 5
Cutoff: 1 / 100
Call Stack
~:0(<built-in method builtins.exec>)
3.28 s
my_script.py:11(<module>)
3.28 s
my_script.py:29(get_ _latest_working_paper)
3.07 s
api.py:64(get)
1.70 s
my_script.py:16(clean_title)
1.04 s
api.py:16(request)
1.70 s
~:0(<built-in method time.sleep>)
1.04 s
Search:

Hands-on

Extras on Euler (Slurm & GPUs)

CPU only

```
#!/bin/bash

#SBATCH -n 1

#SBATCH --cpus-per-task=1

#SBATCH --time=4:00:00

#SBATCH --mem-per-cpu=24576

#SBATCH --mail-type=ALL

python matrix_math_test.py
```

```
$ sbatch …

$ myjobs / seff jobid
```

Runtime: 120s

GPU (numpy -> cupy):

```
#!/bin/bash

#SBATCH --gpus=1

#SBATCH --gres=gpumem:22G

#SBATCH --time=4:00:00

#SBATCH --mail-type=ALL

module load gcc/8.2.0 python_gpu && python
matrix_math_test_gpu.py
```

```
$ sbatch …

$ myjobs / seff jobid
```

Runtime: 5s **(24 x faster)**

Data

	UNCOMPRESSED	COMPRESSED
TEXT	*.xml *.css *.json *.md *.sql *.txt *.csv *.html *.tsv *.py	
BINARY	*.pkl *.tif *.png *.wav	*.mp3 (lossy) *.mp3 *.flac

End-of-Lecture Survey

ETH Edu App

Web app

https://eduapp-app1.ethz.ch/

iOS

Android