TECHNISCHE UNIVERSITÄT DRESDEN

Dresden Database Systems Group

Performance Optimization

Programming for Data Science

Org Stuff

Dresden Database Systems Group

Deadline to hand in your coding solutions

- 05.02.2021, for everyone!

Colloquium (KPDA)

- Presentations will be held on 02.02.2021
- Topics will be assigned on 26.01.2021

Please also tell your fellow students.

General Approaches

Dresden Database Systems Group

Performance, Performance, Performance

Parallelize

Push to C

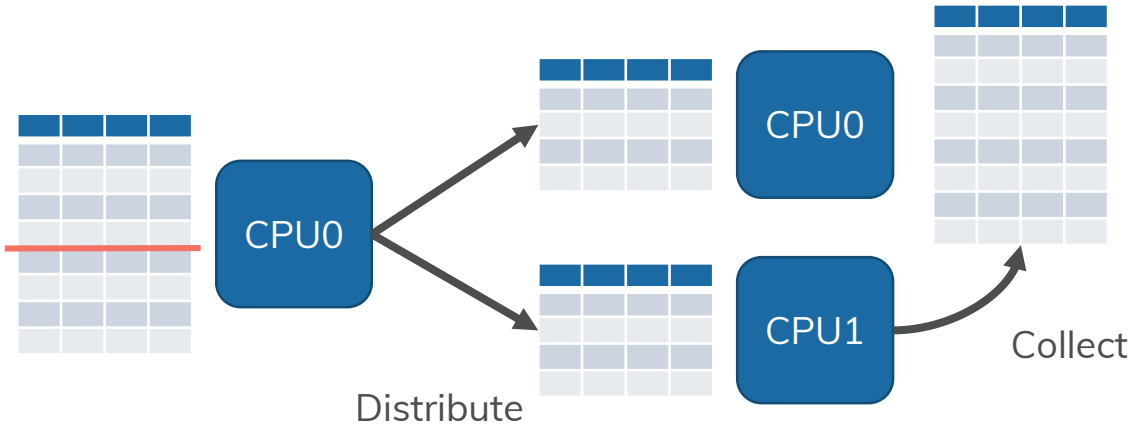
Provide Vectorization

Parallelize

Dresden Database Systems Group

Most Data Science tasks are highly parallel.

- Horizontal data parallelism



Distribute + Collect can be slow!

- R: parallel, python: multiprocessing

Push to C

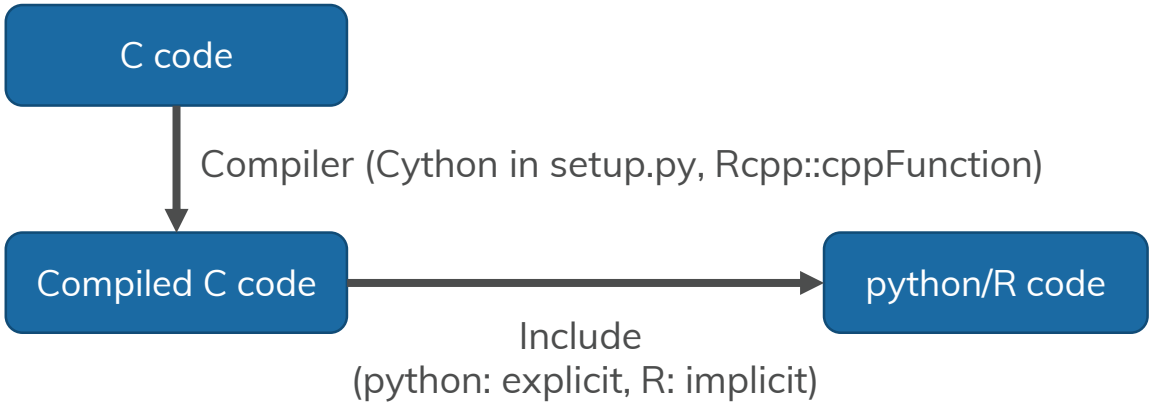
Dresden Database Systems Group

R and python are generally slow

- Interpreted languages
- Weakly typed

C is fast

- Compiled language
- Strongly typed
- Convenient data types from R/python need to be converted. (Done in provided libs)



Provide Vectorization

Dresden Database Systems Group

A for loop can be replaced by a vectorized approach

- Most of the time: matrix solution
- F.e. Euclidean distance

```
for x in X:
  for y in Y:
    ∀i in len(x): sqrt(∑(x[i]-y[i])²)
```

```
M = combinatorial stack columns (X,Y)
sqrt(∑(M[columns from x] - M[columns from y])²)
```

- R: , python: numpy (Not numpy.vectorize!)

Task

Dresden Database Systems Group

Step 0

- Generate two datasets with 2000 two-dimensional points each. (python: `numpy.random.rand`, R: `rnorm`)
- Implement a function* for calculating the Euclidean distances between the two data sets. Only use out-of-the-box functionalities. Measure its run time.

Step 1

- Parallelize* the function from step 0 (python: `multiprocessing.starmap`, R: `parLapply`). Measure its run time.

Step 2

- Implement a C function* (python: edit euclid.pyx → run `python3 setup.py build_ext --inplace` → `from euclid import euclid_c`, R: inline code) for calculating the Euclidean distances. Measure its run time.

Step 3

- Implement a vectorized version* of the Euclidean distance. Measure its run time.

*use your own implementation https://cython.readthedocs.io/en/latest/src/tutorial/cython_tutorial.html#primes

Package suggestions

Dresden Database Systems Group

R

- Rcpp
- parallel
- microbenchmark

python3

- numpy
- Cython
- setup.py (provided)
- multiprocessing
- timeit

Exercise Appointment

Dresden Database Systems Group

We compare and discuss the results

- Tuesday, 19.01.2020,
- Consultation: Please use the forum in Opal.
- Please prepare your solutions! Send us your code!

If you have questions, please mail us:

claudio.hartmann@tu-dresden.de Orga + Code + R
lucas.woltmann@tu-dresden.de Tasks + Python