Programming in R and Python

Lecture 10: Linear algebra in Python

Adam Gudyś
Silesian University of Technology
2020

Linear algebra libraries

BLAS (Basic Linear Algebra Subprograms):

- low-level operations (vector-vector, vector-matrix, matrix-matrix),

LAPACK (Linear Algebra Package):

- high-level operations (eigendecomposition; SVD; LU, QR, Cholesky factorizations, etc.).

- Originally implemented in FORTRAN (that time faster than C),
- R – built in the core; Python – NumPy + SciPy packages,
- Possibility to use different implementations:
  Intel Math Kernel Library, cuBLAS, OpenBLAS, etc.

Python – installing packages

pip - recommended Python package manager, automatically installed with Python 2 $\geq$ 2.7.9 and Python 3 $\geq$ 3.4.

Where to find it?

- Linux: /usr/bin/ (added to PATH by default),
- Windows: <PythonDir>/scripts/

Pip package manager

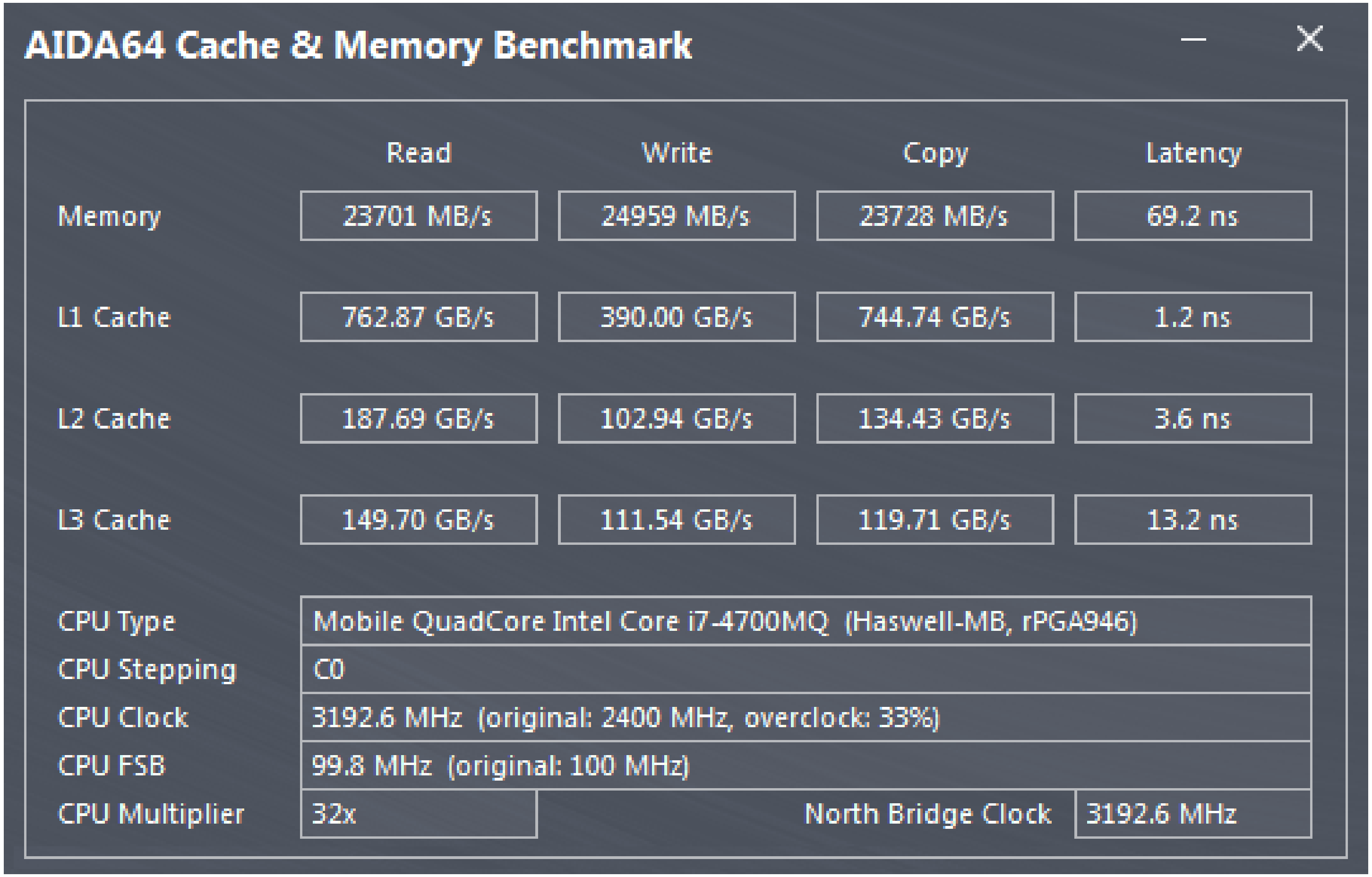
Operation	Command
Install latest version	pip install <name>
Install specified version	pip install <name>=<version>
Install from sources	pip install <src>
Upgrade	pip install –upgrade <name>
Uninstall	pip uninstall <name>

Example:

```
pip install numpy
pip install scipy
```

Code…

Memory organization



AIDA64 Cache & Memory Benchmark

	Read	Write	Copy	Latency
Memory	23701 MB/s	24959 MB/s	23728 MB/s	69.2 ns
L1 Cache	762.87 GB/s	390.00 GB/s	744.74 GB/s	1.2 ns
L2 Cache	187.69 GB/s	102.94 GB/s	134.43 GB/s	3.6 ns
L3 Cache	149.70 GB/s	111.54 GB/s	119.71 GB/s	13.2 ns

CPU Type	Mobile QuadCore Intel Core i7-4700MQ (Haswell-MB, rPGA946)
CPU Stepping	C0
CPU Clock	3192.6 MHz (original: 2400 MHz, overclock: 33%)
CPU FSB	99.8 MHz (original: 100 MHz)
CPU Multiplier	32x
North Bridge Clock	3192.6 MHz

32GB

4x32KB

4x256KB

6MB

Experiment

Matrix multiplication: $C_{M \times N} = A_{M \times K} * B_{K \times N}$

- code written in C, matrices stored as 1D arrays of double.

```
for (int i = 0; i < M; ++i)
    for (int j = 0; j < N; ++j)
        for (int k = 0; k < K; ++k)
            C[i*N+j] += A[i*K+k] + B[k*N+j];
```

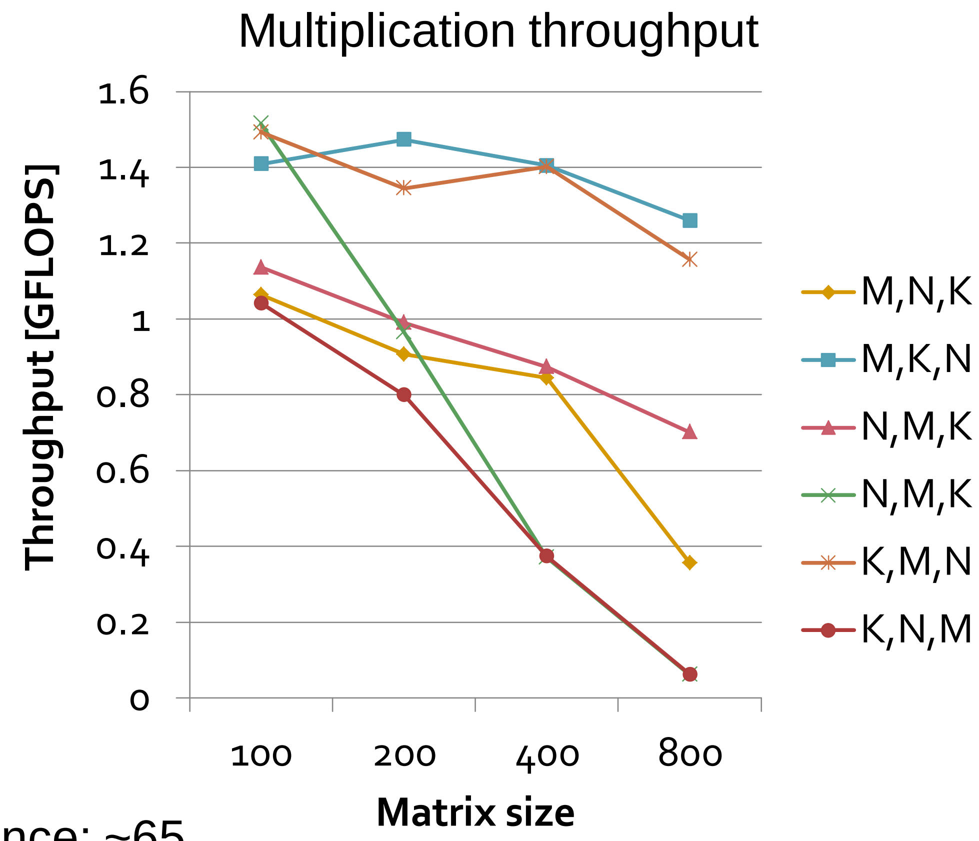
Experimental setting:

- Matrices were assumed to be square: $M = N = K$.
- Different orderings of M, N, K loops were analyzed:
  - each time, same number of operations performed!

Results

Execution times for different loop orderings (milliseconds).

size	100	200	400	800
M,N,K	0.9	8.8	75.8	1437.7
M,K,N	0.7	5.4	45.6	406.5
N,M,K	0.9	8.1	73.3	730.5
N,M,K	0.7	8.3	172.7	8366.2
K,M,N	0.7	6.0	45.7	442.8
K,N,M	1.0	10.0	171.2	8203.0



i7-4700MQ double precision performance: ~65 GFLOPS

Thank you for your attention!