Scientific Computing Libraries

Shusen Wang

Task: Given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times p}$, compute $\mathbf{C} = \mathbf{AB} \in \mathbb{R}^{m \times p}$.

```
Task: Given \mathbf{A} \in \mathbb{R}^{m \times n} and \mathbf{B} \in \mathbb{R}^{n \times p}, compute \mathbf{C} = \mathbf{AB} \in \mathbb{R}^{m \times p}.
```

• Suppose you do not have any vector or matrix multiplication library.

Task: Given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times p}$, compute $\mathbf{C} = \mathbf{AB} \in \mathbb{R}^{m \times p}$.

• Suppose you have only vector-vector multiplication libraries.

```
C = numpy.zeros((m, p))
for i in range(m):
    for l in range(p):
        C[i, l] = numpy.dot(A[i, :], B[:, l])
```

Task: Given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times p}$, compute $\mathbf{C} = \mathbf{AB} \in \mathbb{R}^{m \times p}$.

• Suppose you have matrix-vector multiplication libraries.

```
C = numpy.zeros((m, p))
for l in range(p):
    C[:, l] = numpy.dot(A, B[:, l])
```

Task: Given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times p}$, compute $\mathbf{C} = \mathbf{A}\mathbf{B} \in \mathbb{R}^{m \times p}$.

• Suppose you have matrix-matrix multiplication libraries.

$$C = numpy.dot(A, B)$$

Task: Given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times p}$, compute $\mathbf{C} = \mathbf{AB} \in \mathbb{R}^{m \times p}$.

- Which is the most efficient?
 - 3-level loop of scalar multiplication.
 - 2-level loop of vector-vector multiplication.
 - 1-level loop of matrix-vector multiplication.
 - Directly use matrix-matrix multiplication library.

Task: Given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times p}$, compute $\mathbf{C} = \mathbf{A}\mathbf{B} \in \mathbb{R}^{m \times p}$.

- Which is the most efficient?
 - 3-level loop of scalar multiplication.
 - 2-level loop of **vector-vector multiplication**.
 - 1-level loop of matrix-vector multiplication.
 - Directly use matrix-matrix multiplication library.

• Is your answer the same if the programming language is C or Fortran?

Basic Linear Algebra Subprograms (BLAS)

- **BLAS**: a library of standard building blocks for performing basic vector and matrix operations
- Level 1 BLAS perform scalar, vector, and vector-vector operations.
 - E.g., $\mathbf{y} \leftarrow \alpha \mathbf{x} + \mathbf{y}$, $a \leftarrow \mathbf{x}^T \mathbf{y}$, and $b \leftarrow ||\mathbf{x}||_2$.
- Level 2 BLAS perform matrix-vector operations.
 - E.g, $\mathbf{y} \leftarrow \alpha \mathbf{A} \mathbf{x} + \beta \mathbf{y}$ and $\mathbf{A} \leftarrow \alpha \mathbf{x} \mathbf{y}^T + \mathbf{A}$.
- Level 3 BLAS perform matrix-matrix operations.
 - E.g, $\mathbf{A} \leftarrow \mathbf{A}^T$, $\mathbf{C} \leftarrow \mathbf{A}\mathbf{A}^T$, and $\mathbf{C} \leftarrow \alpha \mathbf{A}\mathbf{B} + \beta \mathbf{C}$.

Implementations of BLAS

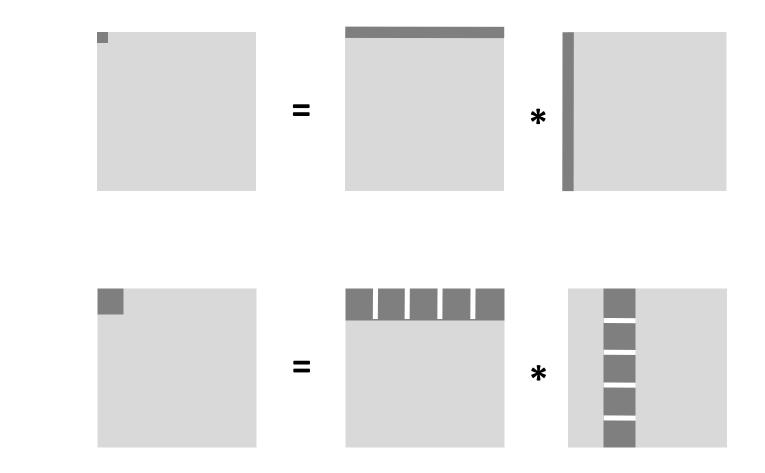
• **Netlib BLAS**: The **official** reference implementation, written in Fortran.

- Intel MKL: optimizations for Intel CPUs.
- NVIDIA cuBLAS: A fast GPU-accelerated implementation.
- Accelerate: Apple's framework for MacOS and iOS.

•

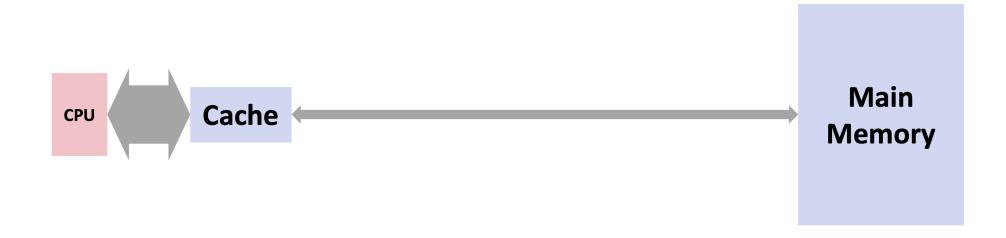
Why is BLAS Fast?

• Efficient algorithms, e.g., blocking, to reduce time complexity.



Why is BLAS Fast?

- Efficient algorithms, e.g., blocking, to reduce time complexity.
- Cache optimization by, e.g., spatial locality.



Why is BLAS Fast?

- Efficient algorithms, e.g., blocking, to reduce time complexity.
- Cache optimization by, e.g., spatial locality.
- Optimized for CPUs/GPUs, e.g.,
 - Intel MKL,
 - NVIDIA cuBLAS

Linear Algebra Package (LAPACK)

- LAPACK provides routines for numerical linear algebra, e.g.,
 - solving least squares,
 - eigenvalue problems,
 - SVD,
 - etc.

Linear Algebra Package (LAPACK)

- LAPACK provides routines for numerical linear algebra, e.g.,
 - solving least squares,
 - eigenvalue problems,
 - SVD,
 - etc.
- LAPACK uses Level 3 BLAS as much as possible.



Linear Algebra Package (LAPACK)

- LAPACK provides routines for numerical linear algebra, e.g.,
 - solving least squares,
 - eigenvalue problems,
 - SVD,
 - etc.
- LAPACK uses Level 3 BLAS as much as possible.
- Numpy uses BLAS and LAPACK for matrix computation.
 - Numpy links against different BLAS on different machines.
 - Check your libraries: numpy. config .show()