# Ubrzavanje izvođenja primjenom koncepata MIMD i SIMD

*Matija Folnović*

Zagreb, 23.01.2015.

# Uvod

- Množenje matrica - vrlo važan problem
- Klasičan način ($O(N^3)$)
- OpenMP
  - API unutar većine prevodioca za C, C++ i Fortran
  - Služi kao jednostavno i fleksibilno sučelje za razvoj aplikacija koje koriste paralelizam
- MKL (engl. Intel Math Kernel Library)
  - Pruža **jako puno** gotovih, optimiziranih, paraleliziranih implementacija matematičkih izračuna (linearna algebra itd.)
- OpenBLAS (engl. BLAS - Basic Linear Algebra Subprograms)
  - C++ Omotač (engl. wrapper) oko najstarije i najrazvijenije biblioteke BLAS - pisane u Fortranu
  - Dodatno paralelizira BLAS-ove (Fortran) procedure

# Gdje?

1. 24x Intel(R) Xeon(R) CPU E52620 @ 2.00GHz
   - Dva procesora ($2 \times 12$ jezgri)
   - 6 fizičkih jezgri (Hyperthreading $=>$ 12 virtualnih)
2. Xeon Phi
   - $\sim$ 1GHz po jezgri
   - Ali, 60 fizičkih jezgri (240 virtualnih)

- Jedna od ideja je usporedba: "malo" jakih jezgri VS puno "slabih" jezgri

# Klasičan način

- Najednostavnija varijanta množenja matrica!
- $A$ i $B$ su matrice koje množimo, $C$ je rezultantna matrica
- Sve matrice su dimenzija $N \times N$
- Evaluacija se odvija nad matricama dimenzija $2000 \times 2000$

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

1. 13.65 s (ICC) / 15.09 s (GCC)
2. $\sim$ 190 s

# Paralelizacija

- Dodana jedna linija (OpenMP ♡)!

```
int i, j, k;
#pragma omp parallel for shared (A, B, C) private(i, j, k)
    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++)
            for (k = 0; k < N; k++)
                C[i][j] += A[i][k] * B[k][j];
```

1. 1.43 s (ICC) / 1.66 s (GCC) $\simeq$ 9.5x ubrzanje
2. 3.35 s $\simeq$ 57x ubrzanje

# Vektorizacija (SIMD)

- Dodano još (par) linija

```
int i, j, k;
#pragma omp parallel for shared (A, B, C) private(i, j, k)
    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++) {
            double sum = 0.0;
#pragma omp simd reduction(+:sum)
            for (k = 0; k < N; k++) {
                sum += A[i][k] * B[k][j];
            C[i][j] = sum;
        }
```

1. 1.07 s (ICC) $\simeq$ 12.75x ubrzanje (u odnosu na klasičan način)
2. 2.56 s $\simeq$ 74x ubrzanje (u odnosu na klasičan način)

# Gotove implementacije

1. MKL

```
cblas_sgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans, N,
    N, N, 1.0, A, N, B, N, 0.0, C, N);
```

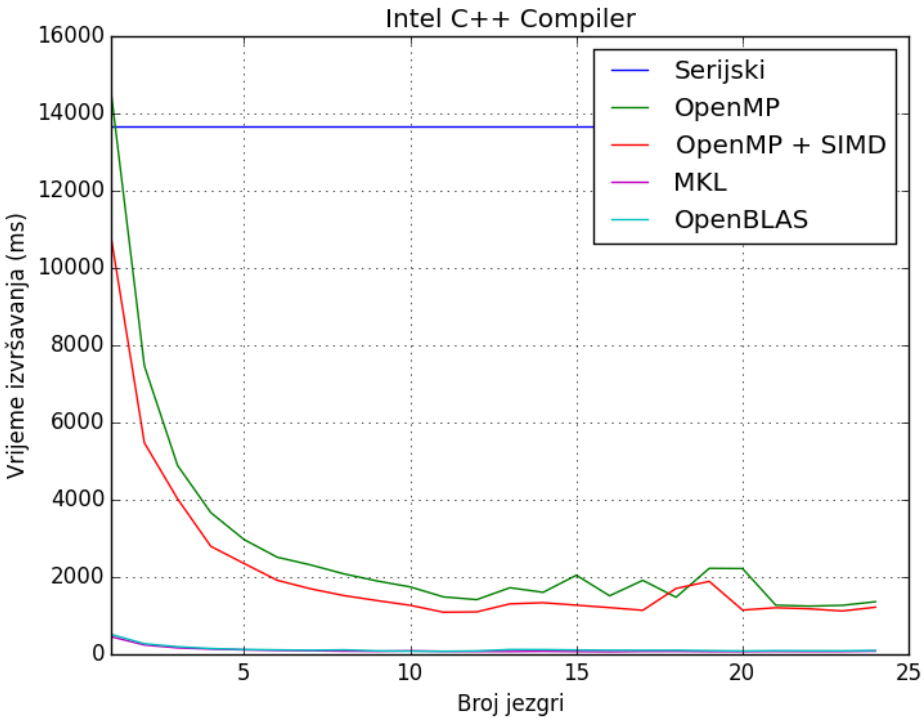
   1. 53.40 ms (ICC) $\simeq$ 255x ubrzanje (u odnosu na klasičan način)
   2. 51.10 ms $\simeq$ 3558x ubrzanje (u odnosu na klasičan način)

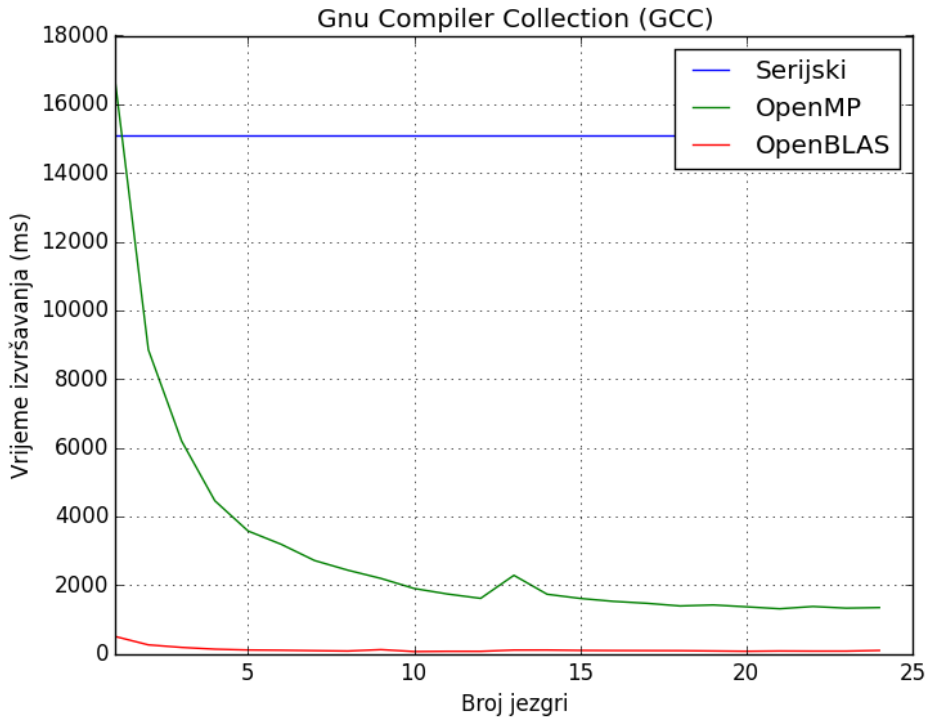
2. OpenBLAS

```
cblas_sgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans, N,
    N, N, 1.0, A, N, B, N, 0.0, C, N);
```

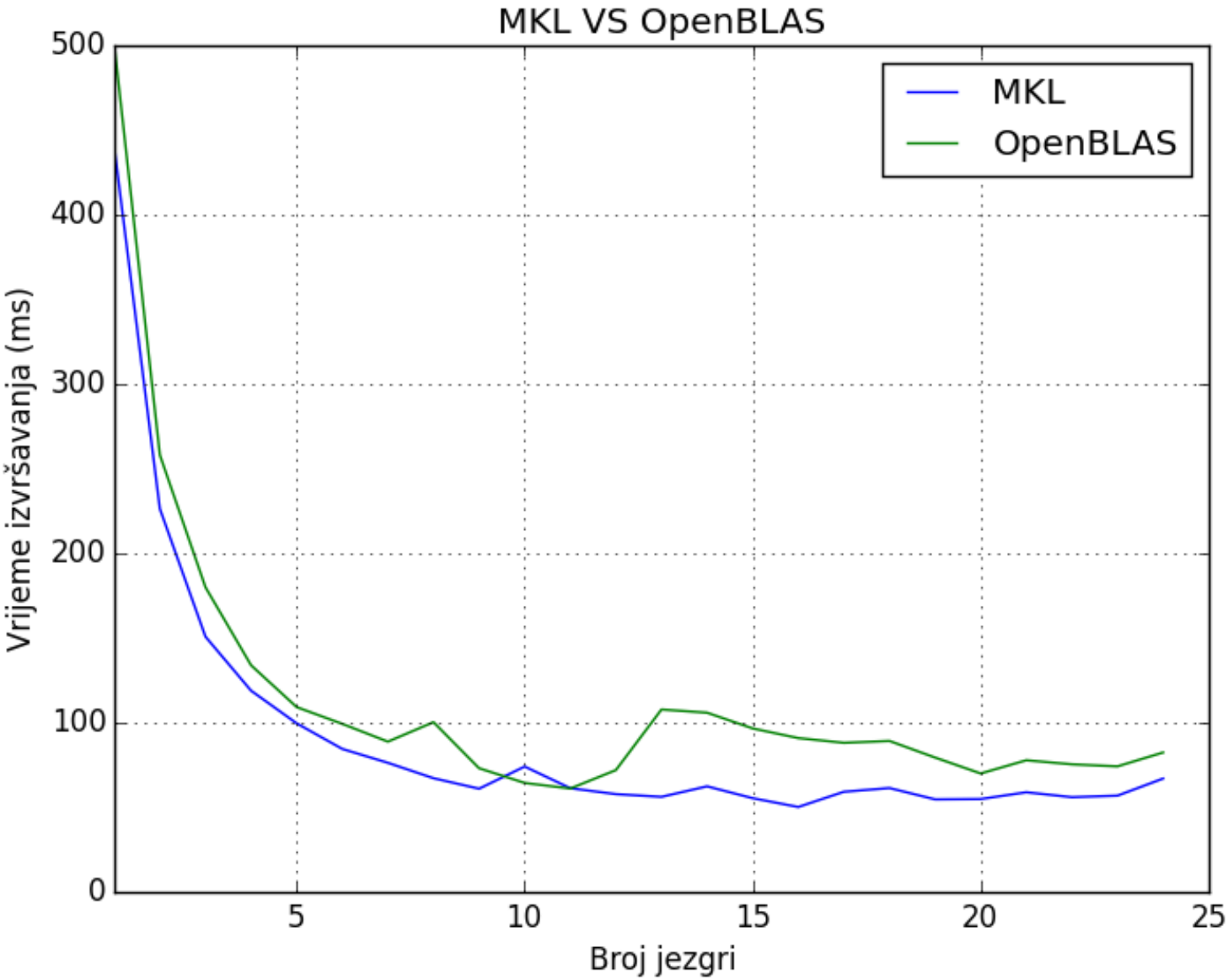
   1. 69.50 ms (ICC) / 68.30 ms (GCC) $\simeq$ 200x ubrzanje (u odnosu na klasičan način)

# Grafovi - 24x Intel(R) Xeon(R) CPU E52620 @ 2.00GHz

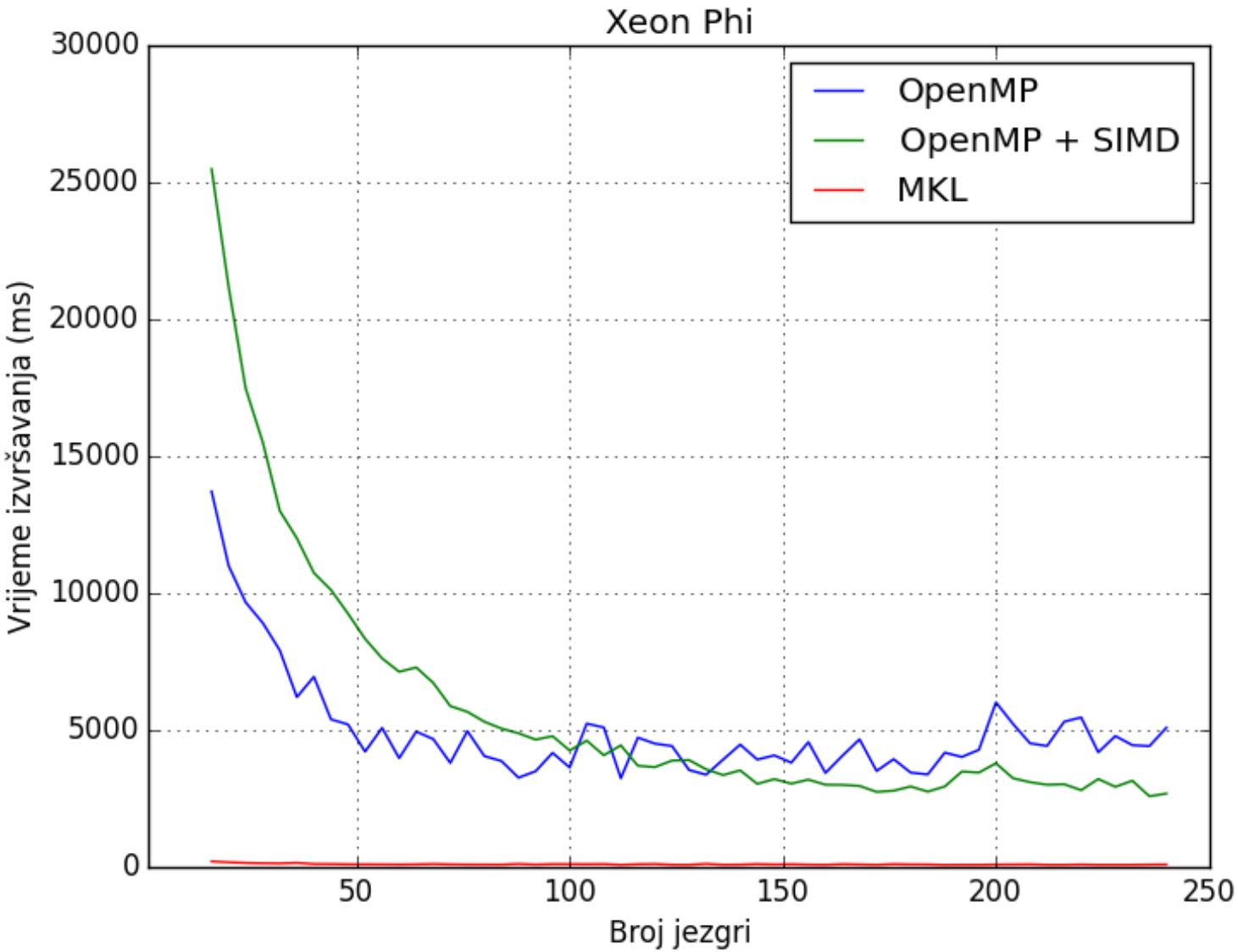


# Grafovi - MKL VS OpenBLAS



# Grafovi - Xeon Phi



# Kraj

Hvala na pažnji!

# Kraj

Hvala na pažnji!

Pitanja, Komentari?