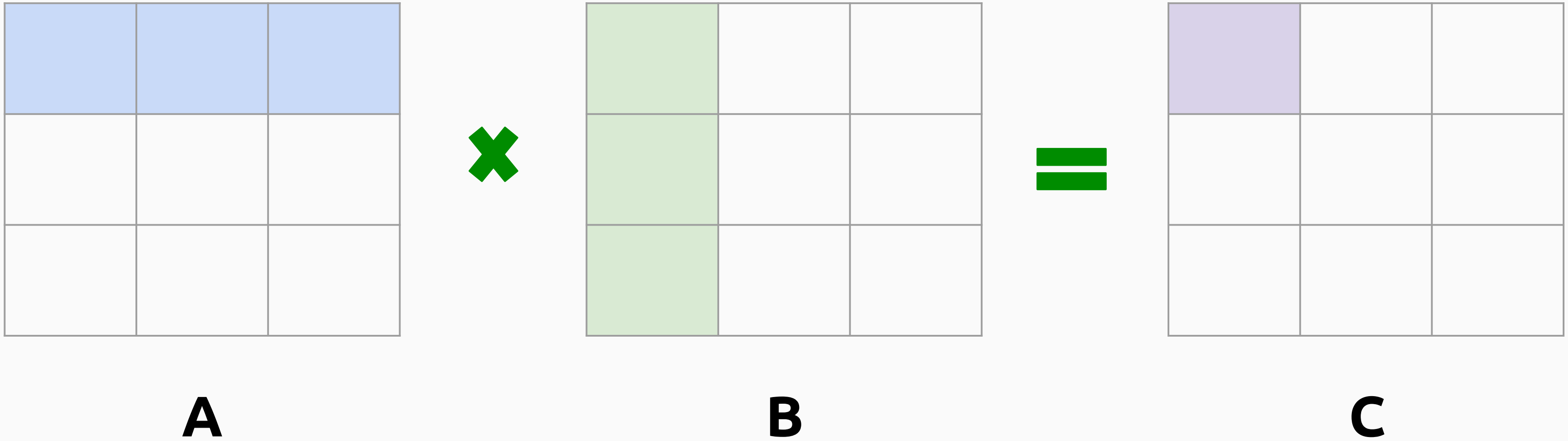
Matrix Multiplication

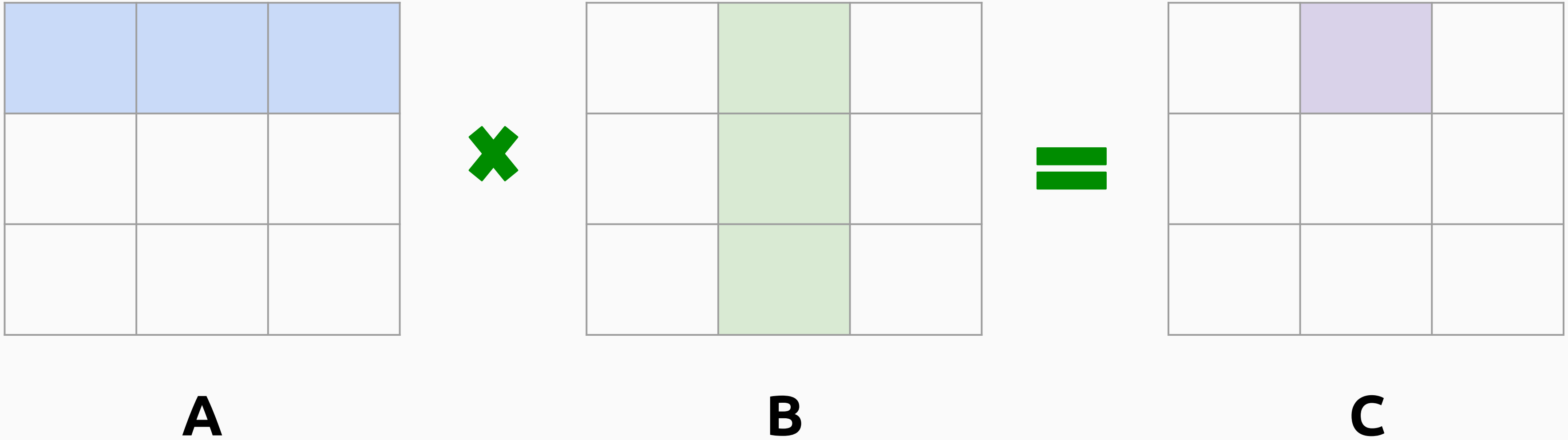
Parallel Computing in Shared Memory using OpenMP

Gabriel Moro - KNOWLEDGE TRANSFER - KT, Porto Alegre - November 2018

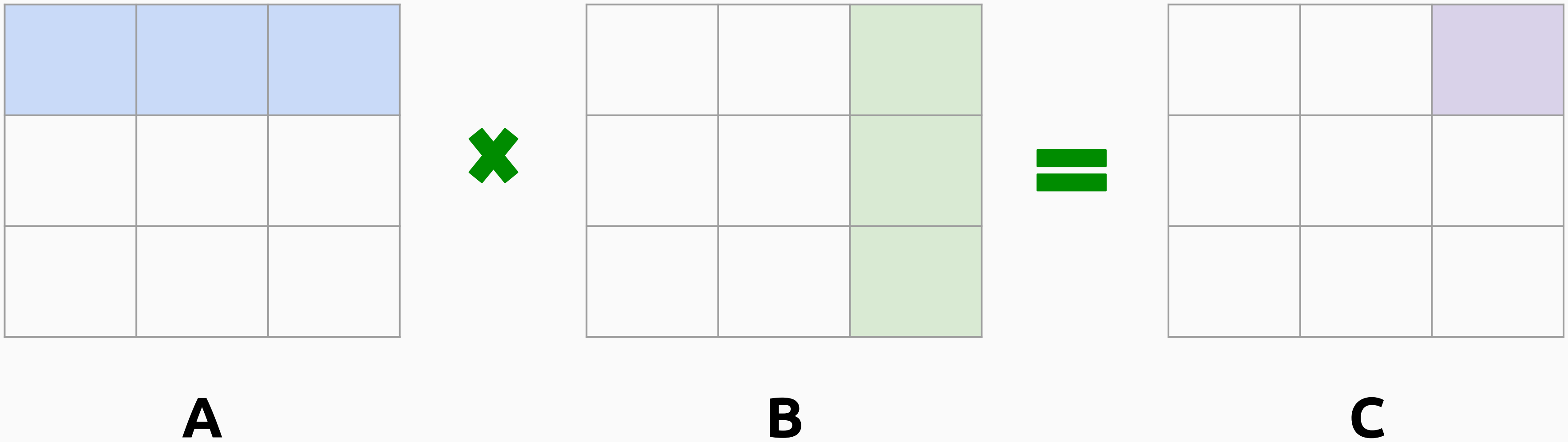
Matrix Multiplication



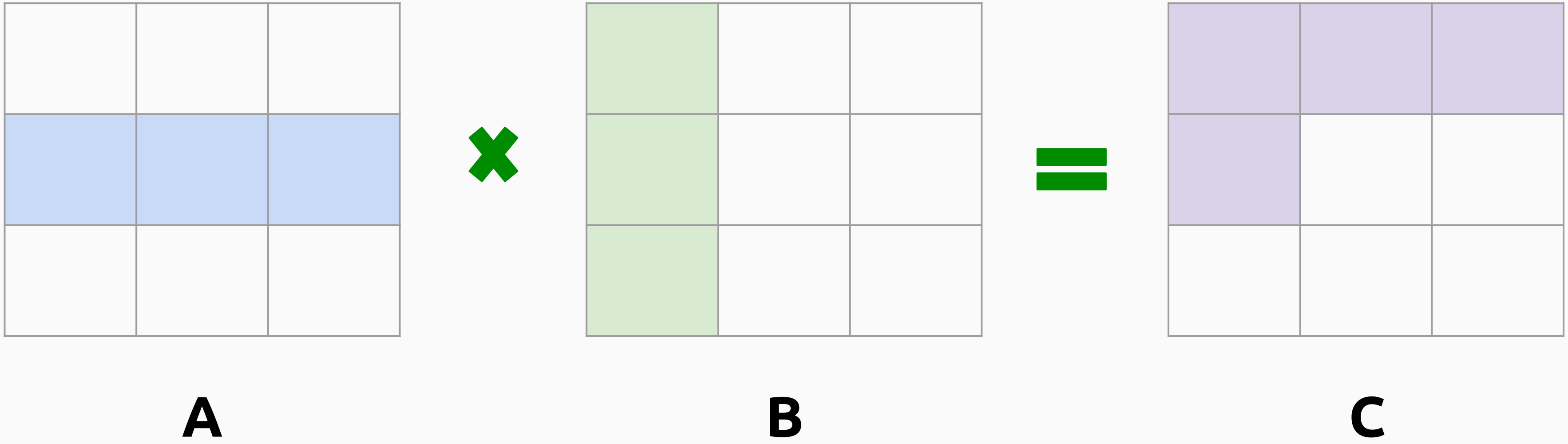
Matrix Multiplication



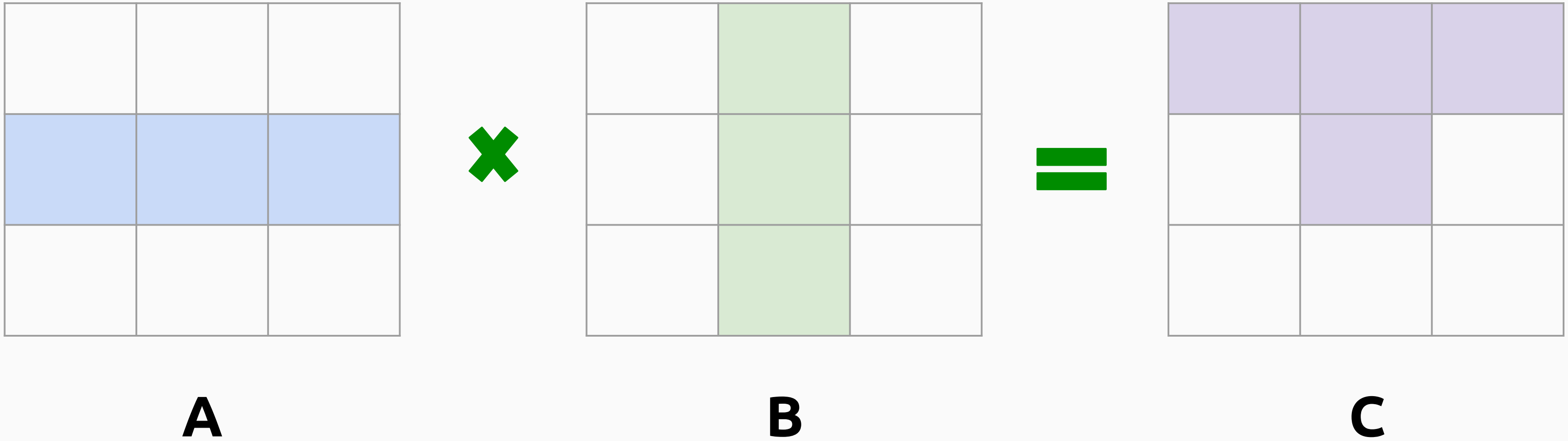
Matrix Multiplication



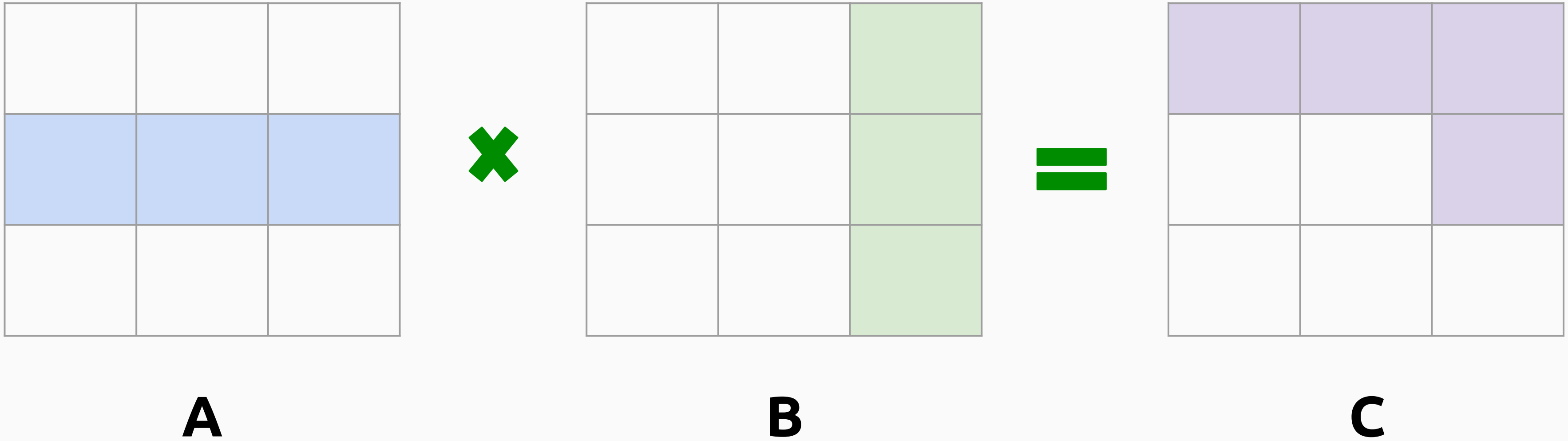
Matrix Multiplication



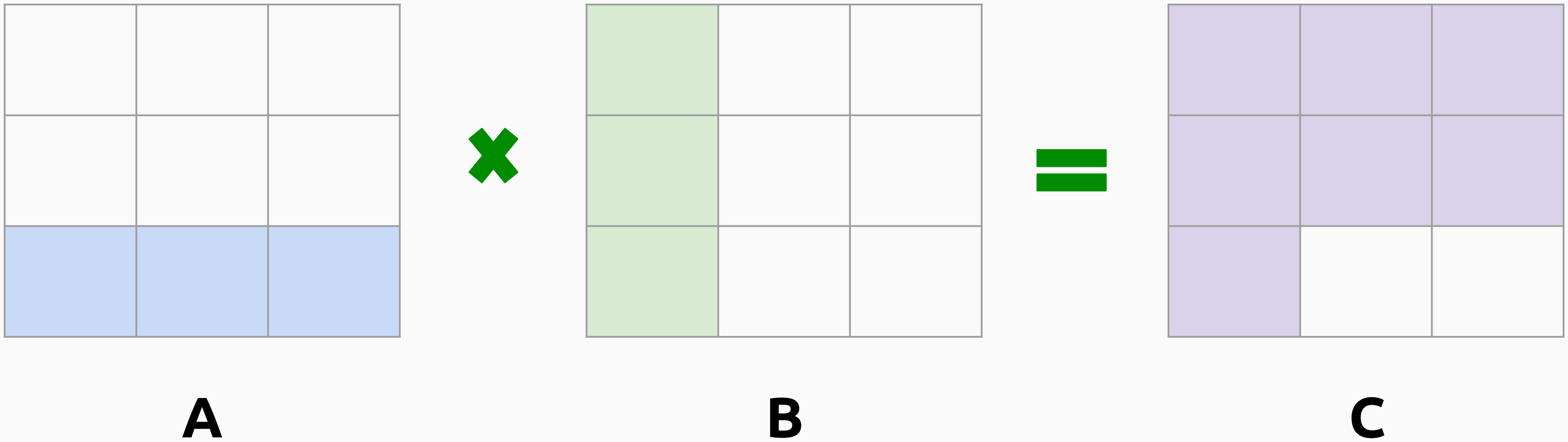
Matrix Multiplication



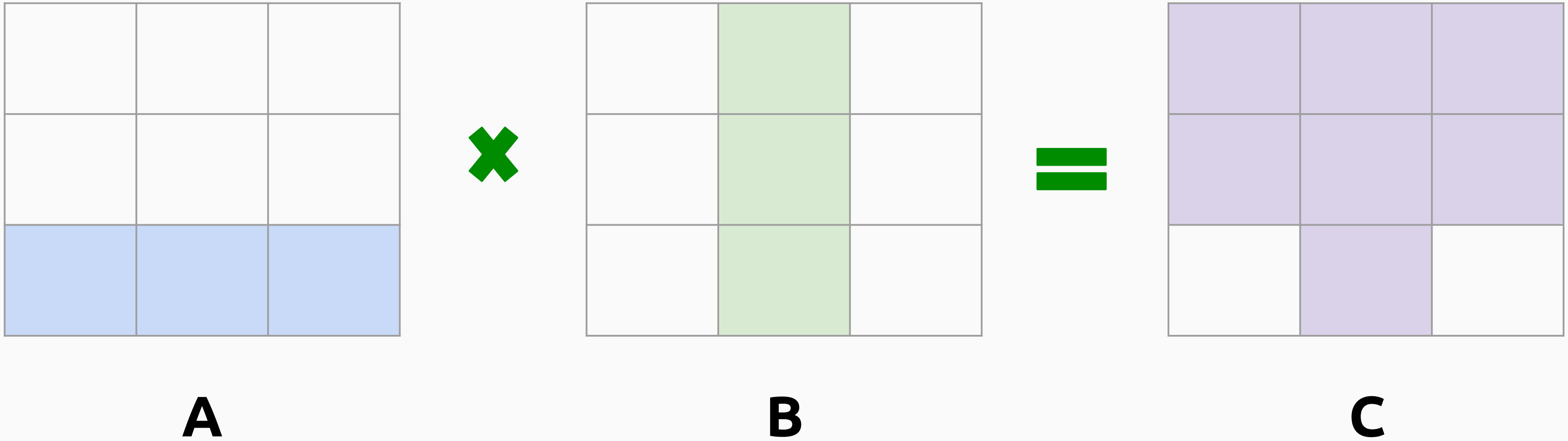
Matrix Multiplication



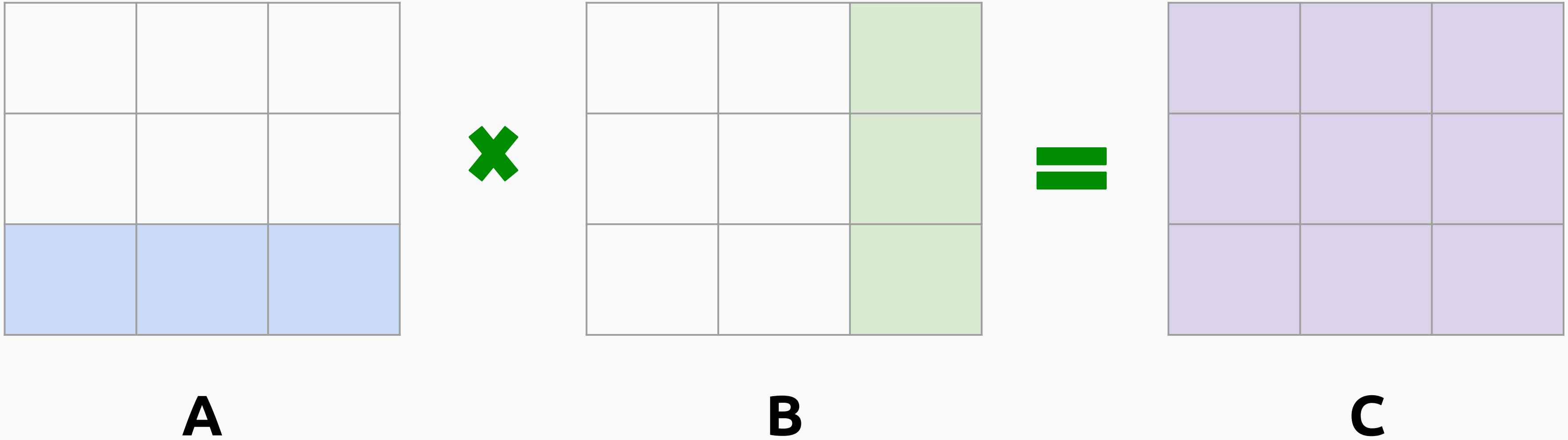
Matrix Multiplication



Matrix Multiplication



Matrix Multiplication



Ways to improve the performance to this algorithm

- Algorithm complexity
- Parallelism

Ways to improve the performance to this algorithm

- Algorithm complexity
- **Parallelism** ←

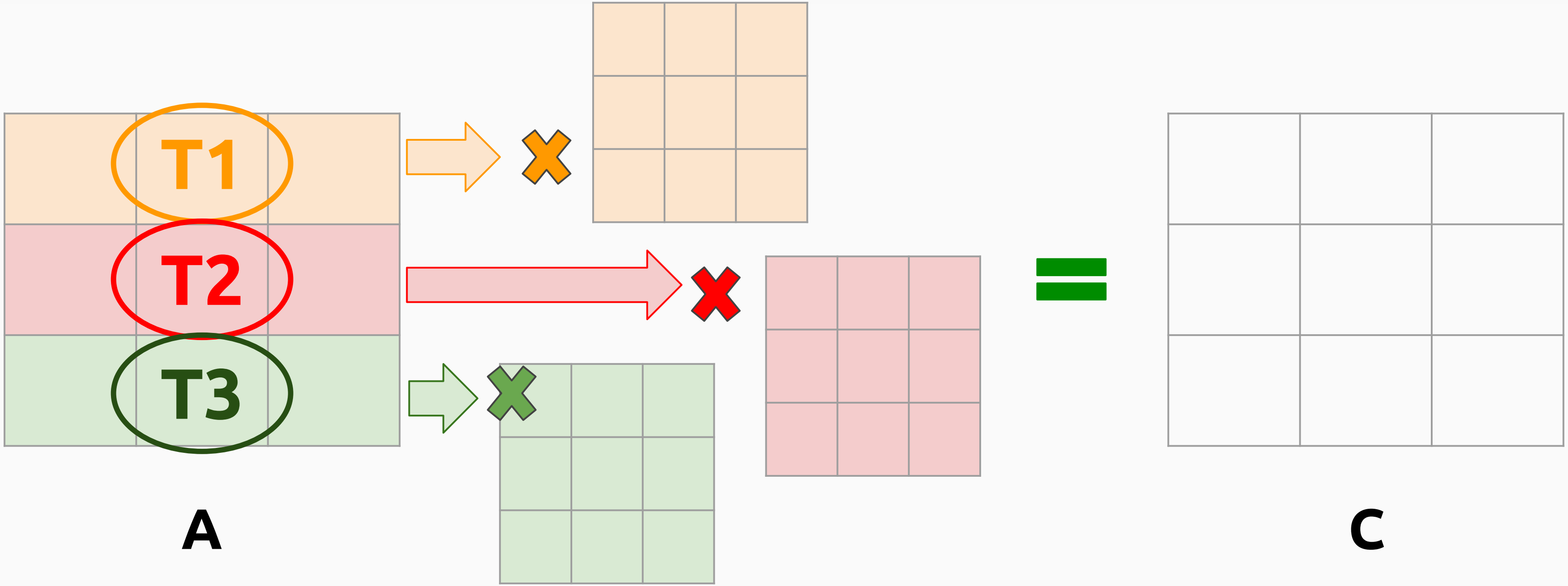
Ways to improve the performance to this algorithm

- Algorithm complexity
- **Parallelism**
  - Shared Memory
  - Distributed Memory

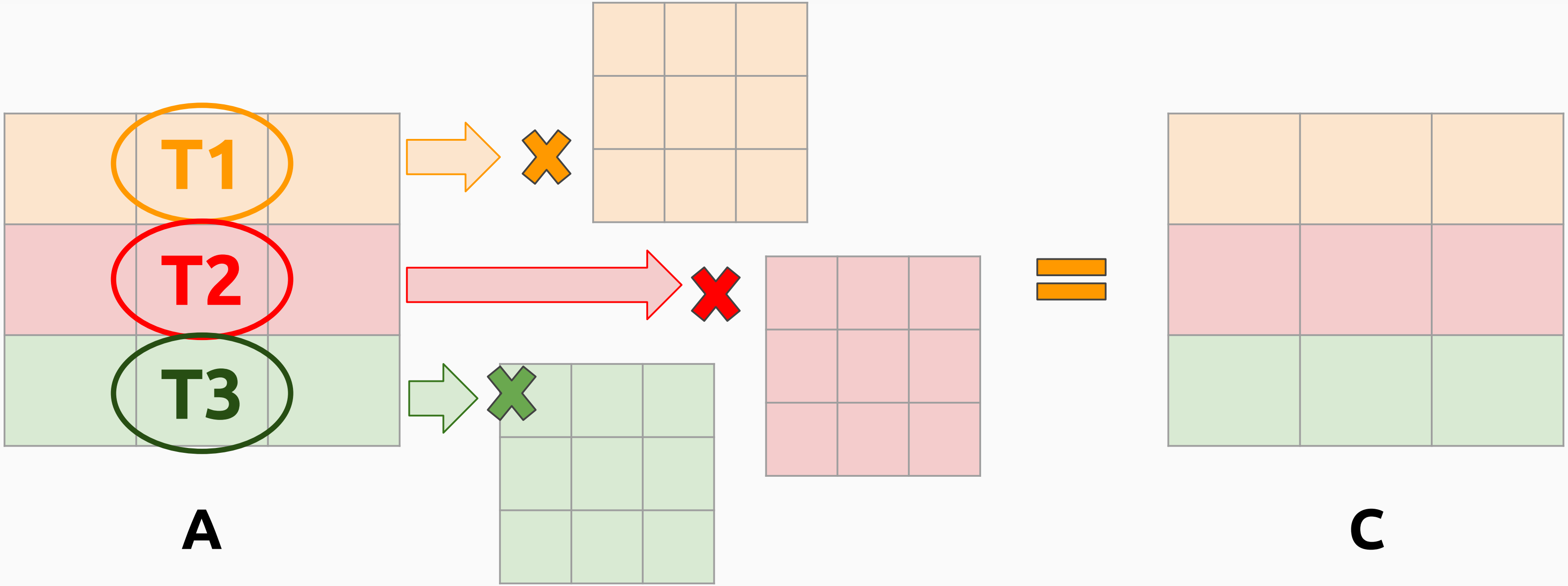
Ways to improve the performance to this algorithm

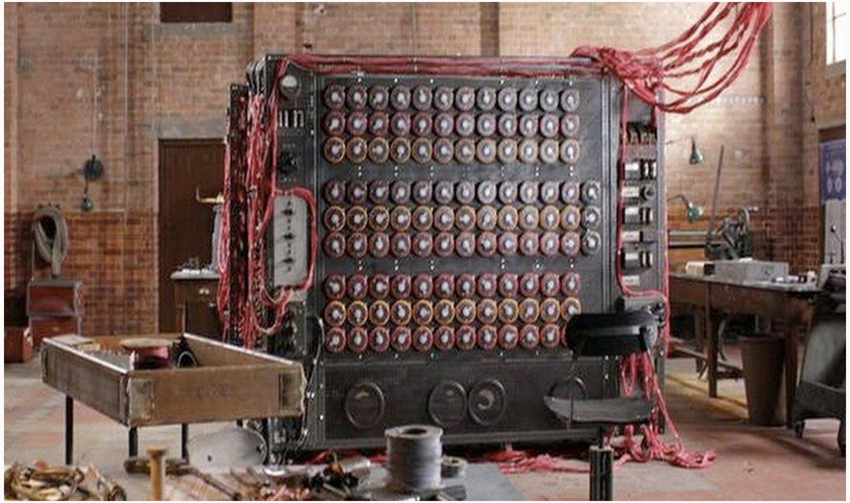
- Algorithm complexity
- **Parallelism**
    - **Shared Memory**
    - Distributed Memory

Parallel OpenMP Model



Parallel OpenMP Model





Turing

- **Processor**
  - 4 x Intel Xeon X7550 Nehalem
  - 32 physical cores
  - HyperThreading
- **Memory**
  - 128GB DDR3
- GPPD-UFRGS

Version: **normal_seq**

```
for(i=0;i < size; i++) {
    for(j=0;j < size; j++) {
        tmp=0;
        for(k=0; k < size; k++)
            tmp = tmp + A[i][k] * B[k][j];
        C[i][j] = tmp;
    }
}
```

Version: **normal_par**

```
#pragma omp parallel for private(i,j,k,tmp)
for(i=0;i < size; i++) {
        for(j=0;j < size; j++) {
            tmp=0;
            for(k=0; k < size; k++)
                tmp = tmp + A[i][k] * B[k][j];
            C[i][j] = tmp;
        }
}
```

Version: **continuos_seq**

```
for(i=0;i < size; i++) {
        for(j=0;j < size; j++) {
            tmp=0;
            for(k=0; k < size; k++)
                tmp = tmp + A[i * size + k] * B[k * size + j];
            C[i * size + j] = tmp;
        }
    }
```

Version: **continuos_par**

```
#pragma omp parallel for private(i,j,k,tmp)
for(i=0;i < size; i++) {
            for(j=0;j < size; j++) {
                  tmp=0;
                  for(k=0; k < size; k++)
                        tmp = tmp + A[i * size + k] * B[k * size + j];
                  C[i * size + j] = tmp;
            }
      }
```

Version: **tiling_seq**

```
register int jj,kk,i,j,k;
double tmp=0;
for(jj=0;jj < size; jj=jj+block) {
    for(kk=0; kk < size; kk=kk+block) {
        for(i=0; i < size; i++) {
            for(j=jj; j < min(jj+block, size); j++) {
                tmp=0;
                for(k=kk; k < min(kk+block,size); k++) {
                    tmp = tmp + A[i][k] * B[k][j];
                }
                R[i][j] = tmp;
            }
        }
    }
}
```

Version: **tiling_par**

```
register int jj,kk,i,j,k;
double tmp=0;
for(jj=0;jj < size; jj=jj+block) {
    for(kk=0; kk < size; kk=kk+block) {
        #pragma omp parallel for private(i,j,k,tmp) schedule(static)
        for(i=0; i < size; i++) {
            for(j=jj; j < min(jj+block, size); j++) {
                tmp=0;
                for(k=kk; k < min(kk+block,size); k++) {
                    tmp = tmp + A[i][k] * B[k][j];
                }
                R[i][j] = tmp;
            }
        }
    }
}
```

Links

- **Top 500:** https://www.top500.org/lists/2018/11/
- **Green 500:** https://www.top500.org/green500/lists/2018/11/
- **NAS Parallel Benchmark:**

  https://www.nas.nasa.gov/publications/npb.html poatek

Thanks!

https://github.com/tido4410/knowledge-transfer-gbmoro.git