HPC: Shared memory systems

UROŠ LOTRIČ

Synchronization

High-level synchronization

- critical
- atomic
- barrier
- ordered
- master
- Single

Low-level synchronization

- flush
- locks

Synchronization: flush

OpenMP supports a shared memory model

- Main memory

Processors can have their own cache

- Cache coherence
- When a thread updates shared data, the new value will first be stored back to the local cache
- The updates are not necessarily immediately visible to other threads

The flush directive makes thread's temporary view of shared data consistent with the value in memory

- Thread-visible variables are written back to memory

Synchronization: lock routines

A lock implies a memory fence of all thread-visible variables.

- With locks we can gurantee that only one thread accesses a variable at a time
- Avoid race conditions.
- Lock structure: omp_lock_t or omp_nest_lock_t.
  - Ordinary and nested locks
- Simple Lock routines:
  - omp_init_lock
  - omp_set_lock
  - omp_unset_lock
  - omp_test_lock
  - omp_destroy_lock

Synchronization: lock routines

How to use locks:

- Define the lock variables
- Initialize the lock
- Set the lock or test for locked
  - Test checks whether the lock is available before attempting to set it
- Unset a lock after the work is done
- Destroy the lock

- More common naming: mutex (Mutual Exclusion)

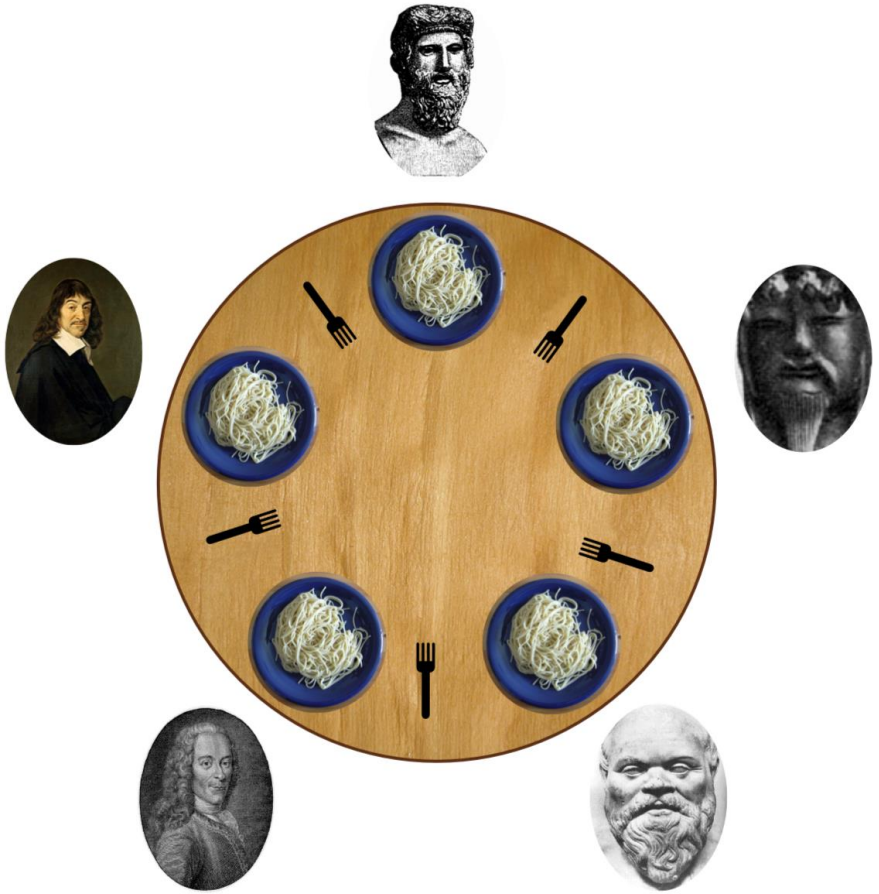
```
omp_init_lock (&my_lock);

#pragma omp parallel for
for (int i = 0; i < N; ++i)
{
    omp_set_lock (&my_lock);
    count++;
    omp_unset_lock (&my_lock);
}

omp_destroy_lock (&my_lock);
```

Synchronization: lock routines

Dijkstra

Dining Philosophers Problem

- Five philosophers, plates of spaghetti and five forks.
- Philosophers have a discussion: they think and talk, become hungry, eat, think and talk, ...
- Each philosopher eats with two forks, he can only take a fork of his neighbor
- How to prevent a dead-lock?



Performance issues

Loop interchange to increase cache locality

- placement of matrix in memory

Avoid parallel overhead when number of iterations is low

- #pragma omp parallel for **if (iters > 100)**
- for(i = 0; i < iters; i++)
- …

Performance issues

Move synchronization point outwards

- Split omp parallel (thread creation) and omp for (iterations)
- Example: Conway‘s game of life

```
while (iters < MAXITERS)
{
    #pragma omp parallel for collapse(2) private (neighs)
    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++)
        {
```

```
#pragma omp parallel private(neighs)
while (iters < MAXITERS)
{
    #pragma omp for collapse(2)
    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++)
        {
```

Vector unit

SSE (Streaming SIMD Extensions)

- 128-bit registers

AVX (Advanced Vector Extensions)

- AVX & AVX2 (256-bit registers), AVX-512 (512-bit registers)
- Parallel operations
  - Single precision FP: 8 x 32 bit
  - Double precision FP: 4 x 64 bit

Compilers of today attempt to do vectorization automatically

- Compiler switch -O2 is a must

OpenMP & SIMD

#pragma omp simd

- Introduced with OpenMP 4.0
- The simd directive can be thought essentially as a directive to the compiler, saying: „Try harder“.

Explicit vectorization of for loops

- Automatic vectorization not always possible – compiler does not know that data structures do not overlap
- Specified in the same way as with parallel
- Can be combined with parallel

OpenMP & SIMD

#pragma omp simd

Clauses

- simdlen (len): recommended size (iterations) per chunk, vectorization is performed in chuks of simdlen
- safelen (len): specifies the size of a chunk with no data dependency
- aligned(vars: bits): informs omp that variable is aligned in memory
- collapse, private, firstprivate, reduction, …

OpenMP & SIMD: memory alignment

Data transfer is faster when memory addresses are aligned

Allows for faster hardware instructions to load vector

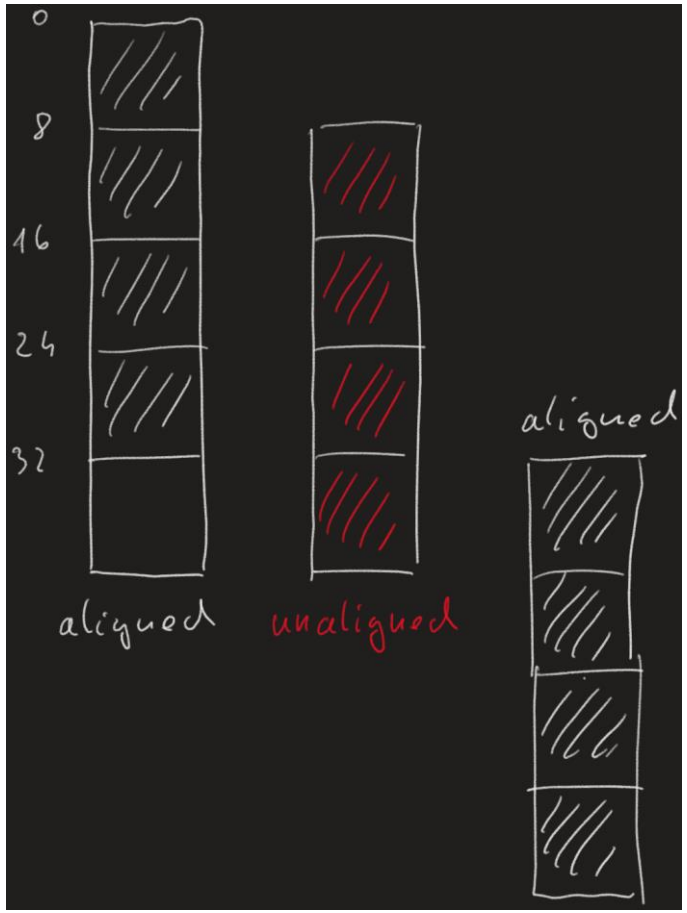
64B byte cache line

- AVX1&2: best is to align at 32 bytes (256 bits)
- AVX-512: best is to align at 64 bytes (512 bits)

Aligned memory allocation

- Important to use with custom data types
- Usage:

```
buffer = aligned_alloc(32, num_bytes);
…
#pragma omp simd aligned(buffer, 32)
…
free(buffer);
```



OpenMP & SIMD: functions

Call of a function from a loop

We can instruct the compiler that a function can be vectorized

#pragma omp declare simd

- Indicate a function that can explicitly use vectorization
- Instruct compiler to prepare different versions of the function

```
#pragma omp simd
for(int i = 0; i < n; i++)
{
    c[i] = mymin(a[i], b[i]);
}
```

```
#pragma omp declare simd
int __attribute__ ((noinline)) mymin (int a, int b)
{
    return a < b ? a : b;
}
```