OpenMP: A brief overview

David Stewart

August 2020

What is OpenMP?

- An *open* specification for a collection of **compiler directives**, **runtime libraries**, and **environment variables**
- Designed to be used with Fortran, **C** and C++;
  - I will discuss in the context of C
- Extensive compiler support (find full list at https://www.openmp.org/resources/openmp-compilers-tools/)
  - gcc
  - Clang
  - AOMP (AMD);
  - Visual Studio C++
- Latest specification can be found at https://www.openmp.org/specifications/

What is OpenMP?

- First specification published by the OpenMP Architecture Review Board (ARB) in October 1997 for Fortran;
  - Full list of members can be found at https://www.openmp.org/about/members/
- C and C++ support introduced in 1998;
- Major versions were staggered, based on programming language:
  - 2.0 for Fortran released in 2000;
  - 2.0 for C and C++ released in 2002;
  - 5.0 released in November 2018;
  - OpenMP 6.0 slated for November 2023 release;

What is OpenMP?

Figure: Some members of the ARB

What is OpenMP?

- Programming model: shared address space* (intended for usage on shared memory systems, both NUMA and SMP);
  - * some attempts to allow OpenMP be deployed in a distributed memory system
- Mature API, gained a lot of popularity over its 23 year lifespan;
- Broad range of uses: popular in scientific computing, but portability (both hardware and OS) makes it versatile
  - Abstracts from lower level threading like pthreads/winthreads

What is OpenMP?

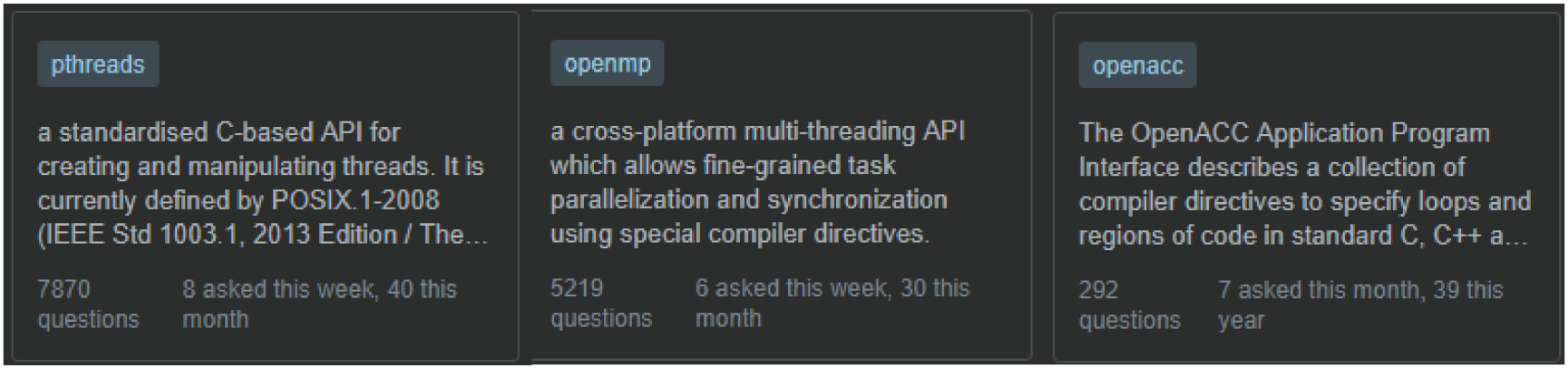


Figure: Popularity of threading libraries on StackOverflow (August 2020)

Compiler directives

- They are *not* part of the grammar of the language itself (although they are *allowed* under the C-standard, and indeed *some* are specified);
- Directives are extra instructions you give to a compiler to tell it how to process your code and generate its outputs
- This all happens *at compile time*
- The majority of the OpenMP-specific code you write will be in the form of compiler directives, starting with the `pragma` directive
- Directives are used for defining which regions can be run in parallel, how to share the work amongst threads and how access will be synchronised.

Annotated code example

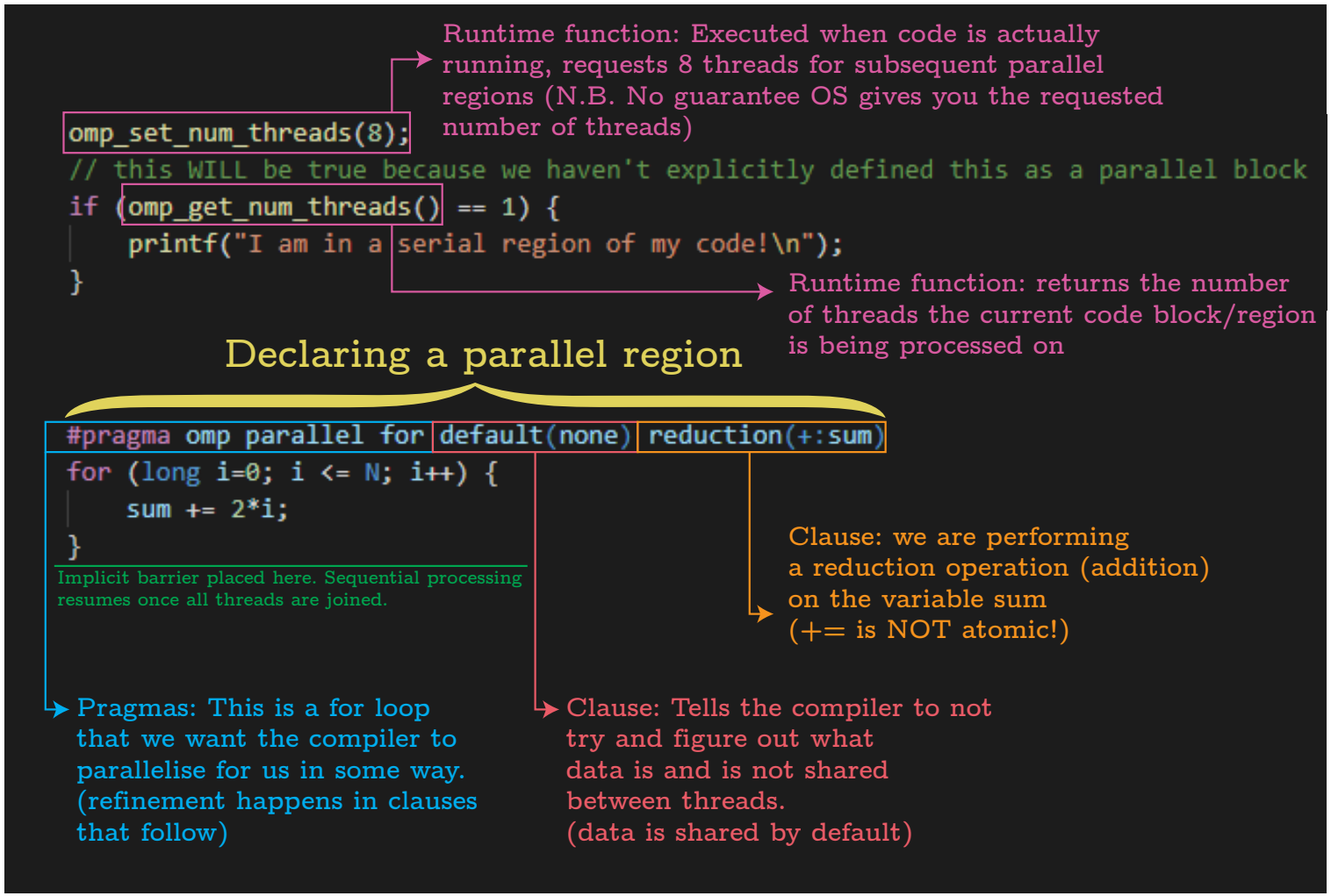


Figure: Annotated code example (OpenMP for C)

Matrix Multiplication

OpenMP implementation

```
omp_set_num_threads(NUM_THREADS);

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

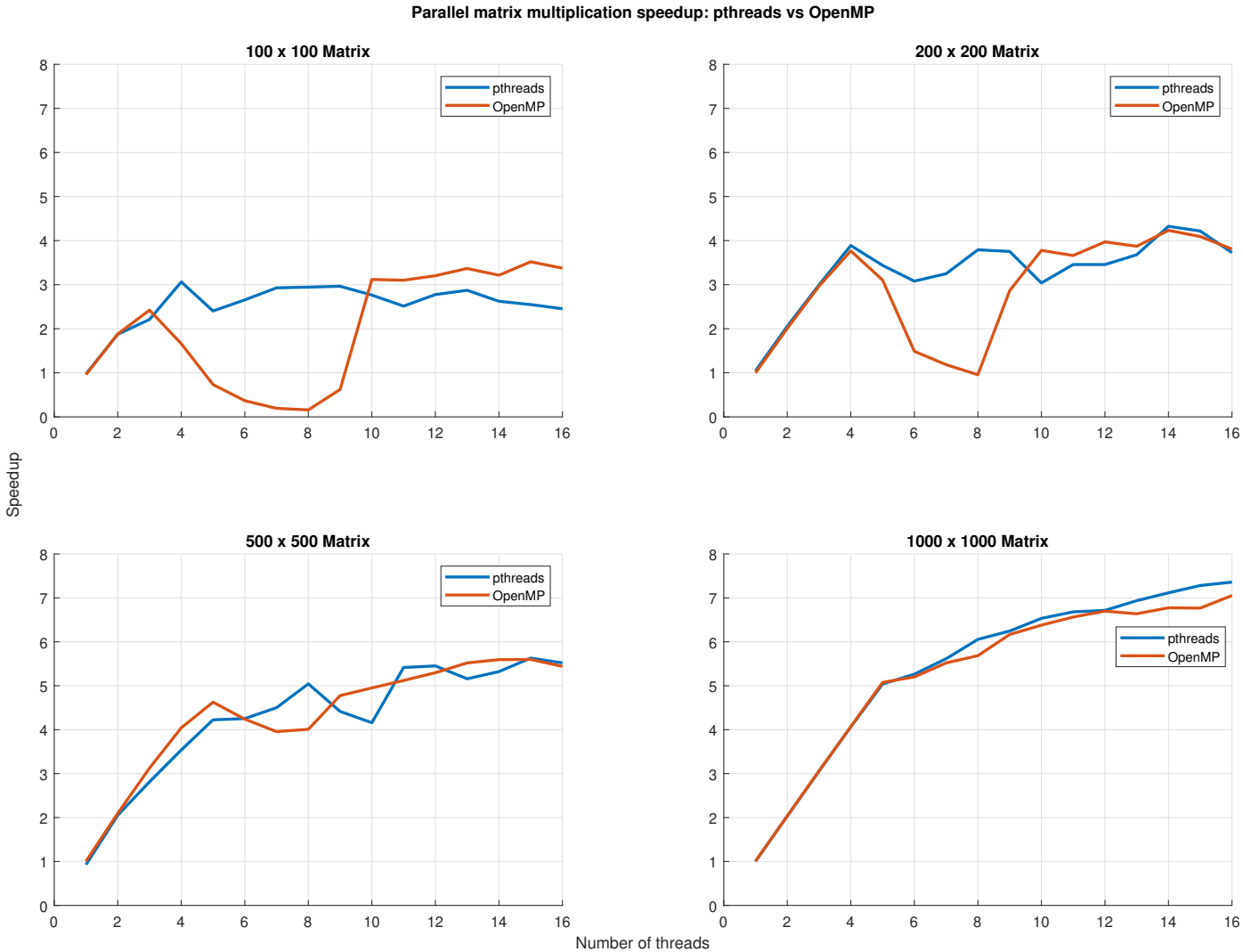
Matrix Multiplication

pthread implementation

- Too much code for embedding in slides, find implementation at https://github.com/davidjmstewart/CAB401-OpenMP

Speedup

pthread implementation



Pros and Cons of OpenMP

OpenMP Pros & Cons	
Pros	**Cons**
Portable	Can be difficult to debug or pinpoint precise causes for performance losses
Syntactically neat API	Not designed for fine grained control (e.g. mapping threads to processors);
Code comprehensibility: write serially, think concurrently	Shared memory only

Further learning

- Introduction to OpenMP (Tim Mattson): Free YouTube course
- Pthreads and OpenMP: A performance and productivity study.