Assignment Project Exam Help

COMP3221 Parallel Computation

https://powcoder.com

Peter Jimack

University of Leeds

Add WeChat powcoder

Lecture 2: Introduction to shared memory parallelism (SMP)

Previous lectures

Assignment Project Exam Help

In the last introductory lecture we saw:

- Why **technological limitations** have led to multi-core CPUs.
- Parallel architectures also present in high-performance **clusters**, and **graphics processing units** (GPUs).
- Some general concepts:
  - **Concurrency** (more general than parallelism).
  - **Shared** *versus* **distributed memory**.
  - Potential performance issues related to **communication**.
  - **Flynn's taxonomy**.

https://powcoder.com
Add WeChat powcoder

This lecture

Assignment Project Exam Help

This lecture is the first of six on **shared memory parallelism**, relevant to **multi-core CPUs**.

https://powcoder.com

- The hardware **architecture**, including the memory cache.
- Processes *versus* threads and the thread **scheduler**.
- Languages and frameworks suitable for these systems.
- How to set up and run OpenMP.

Add WeChat powcoder

Multi-core CPUs

Assignment Project Exam Help

A **core** is a single **processing unit** that executes instructions.

- Has components that *fetch*, *decode etc.* instructions.
- **Functional units** for integer and floating point operations.
- Other features such as **instruction level parallelism**.

https://powcoder.com

As its name suggests, **multi-core** processors contain more than one such unit.

Add WeChat powcoder

- **MIMD** in Flynn's taxonomy (*single cores are SISD*).
- Most common now are **dual core**, **quad core** and **octa core**.
- High-performance chips can have many more, *e.g.* SW26010 (used in China's Sunway TaihuLight supercomputer) has 260.

Simultaneous multithreading

Some chips employ **simultaneous multithreading**[1]:

Assignment Project Exam Help

- Two (or more) threads run on the same core.
- If one thread stops execution (*e.g.* to wait for memory access), the other takes over.

https://powcoder.com

Appears as two **logical processors** to the programmer, and only requires 5% increase in chip area.

- Performance improvements only 15%-30%.[2]

Add WeChat powcoder

> When interrogating a framework for the maximum number of available threads, you may get **more** than the number of cores.

[1] Known as **hyperthreading** on Intel chips.

[2] Rauber and Rünger, *Parallel Programming* 2nd ed. (Springer, 2013).

The processor-memory gap

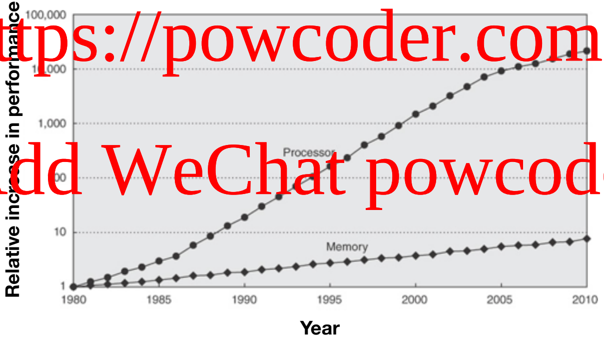
Memory access rates are increasing far slower than processor performance (taking into account number of cores):

Assignment Project Exam Help

- This is the **processor-memory gap**.

https://powcoder.com

Add WeChat powcoder

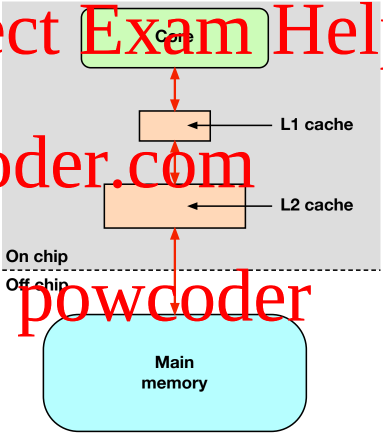


Hennessy and Patterson, *Computer Architecture: A Quantitative Approach* (Morgan Kauffman, 2006).

Single-core memory caches: A reminder

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- **Small, fast on-chip** memory.
- Accessing main memory returns a **line** to the cache (*e.g.* 64 bytes).
- Subsequent accesses return the cache data (a **cache hit** - *fast*), or from main memory (a **cache miss** - *slow*).
- Multiple caches **levels** (*e.g.* L1, L2, L3) arranged **hierarchically**.



Schematic two-level cache

Multi-core memory caches

Different manufacturers choose different ways to incorporate caches into multi-core designs.

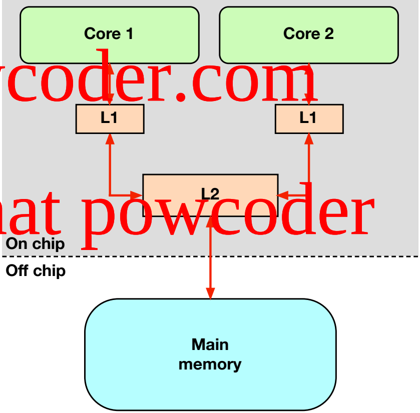
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Often use **hierarchy**:

- Each core has its own L1 cache.
- Share higher level caches.

For *e.g.* quad cores:

- L1 for each core.
- L2 for pairs.
- L3 for all cores.



Cache coherency

Assignment Project Exam Help

1. Core 1 reads an address x, resulting in a line in its L1.
2. Core 2 does the same, resulting in a line in its L1.
3. Core 1 changes the value of x in its L1.
4. Core 2 reads x from its L1, which **still has the old value**.

https://powcoder.com

Maintaining consistent memory views for all cores is known as **cache coherency**.

Add WeChat powcoder

A common way to maintain **cache coherency** is **snooping**:

- The *cache controller* **detects writes** to caches, and updates higher-level caches.

False sharing

Assignment Project Exam Help

Maintaining cache coherency incurs a performance loss.

- If two cores repeatedly write to the same memory location, the higher level caches will be constantly updated.

https://powcoder.com

However, if the cores write to nearby **but different** memory locations **on the same cache line**, updates will still occur.

- *i.e.* hardware **performance loss** with no need.

Add WeChat powcoder

This unnecessary cache coherency is known as **false sharing**

Potential benefit of cache sharing

Assignment Project Exam Help

It is also possible for multiple cores to benefit from shared caches:

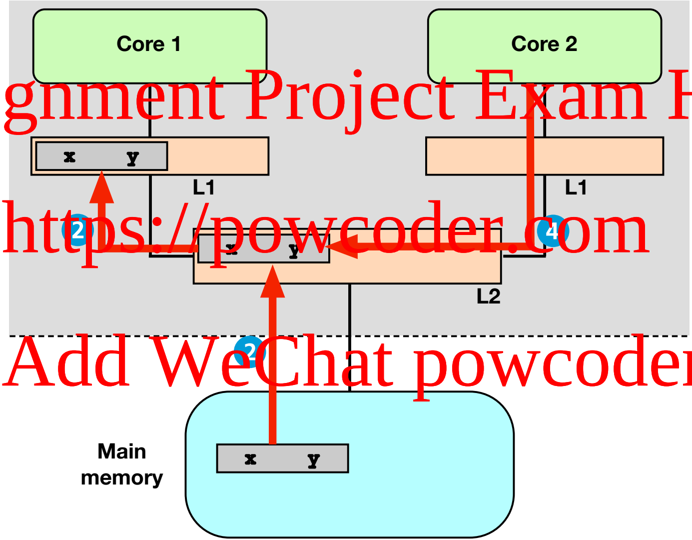
1. Core 1 reads an address `x` from main memory.
2. A line including `x` is read into L2, and the L1 for Core 1.
3. Core 2 now tries to access an address `y` that is 'near to' `x`.
4. If `y` is on the line just copied into L2, Core 2 will **not need to access main memory**.

https://powcoder.com

It is therefore possible for fewer accesses to main memory overall, compared to the equivalent serial code.

Add WeChat powcoder

This can result in parallel speed up **more than the number of cores** (known as superlinear speedup; cf. Lecture 4).



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Processes *versus* threads

Assignment Project Exam Help

Control flows can be either **processes** or **threads**:

Processes:

- **Executable program** plus all required information.
- Register, stack and heap memory, with own address space.
- Explicit communication between processes (*via* sockets).
- **Expensive** to generate (large heap memory).

https://powcoder.com

Add WeChat powcoder

Threads:

- Threads of one process **share** its address space.
- Implicit communication *via* this **shared memory**.
- **Cheap** to generate (no heap memory).

Kernel *versus* user-level threads

Assignment Project Exam Help

The threads that execute on the core(s) are **kernel threads**.

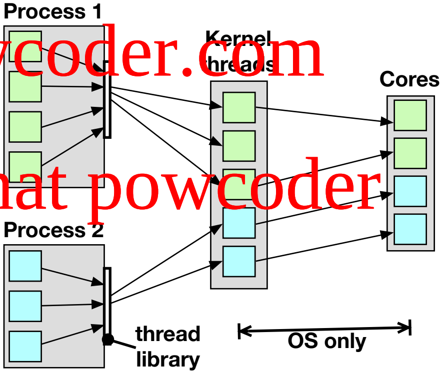
- Only the OS has direct control over kernel threads.

https://powcoder.com

Programmers instead generate **user-level threads**.

- Managed by a **thread library**.
- Mapped to kernel threads by the OS **scheduler**.

Add WeChat powcoder



Thread programming

Assignment Project Exam Help

The choice of programming framework/API/library/etc. **must** be suitable for the architecture on which it will run.

- Multi-core CPUs use **SMP** = Shared Memory Parallelism.

https://powcoder.com

It is possible to program user threads directly:

- Java supported threads early on, through the `Thread` class and `Runnable` interface.
- The C library `pthread` implements POSIX threads.
- C++11 has language-level concurrency support.
- Python has a threading library, although need to work around its **global interpreter lock** to exploit multi-cores.

Add WeChat powcoder

Higher-level threading support

Assignment Project Exam Help

Higher-level options that do not require explicit thread control also exist to reduce development times.

- Java's `Concurrency` library (in `java.util`).
- The OpenMP standard (this module and next slides).

https://powcoder.com

For C/C++, as well as OpenMP there are [see McCool *et al.* in *Structured Parallel Programming* (Morgan-Kaufman, 2012)].

- **Cilk Plus**.
- **TBB** (Threading Building Blocks).
- **ArBB** (Array Building Blocks).
- **OpenCL**, although primarily used for GPUs.

Add WeChat powcoder

The first three are not (yet?) widely implemented in compilers.

OpenMP

Assignment Project Exam Help

For the SMP component of this module we will use **OpenMP**.

- Portable standard devised in 1997 and widely implemented in C, C++ and FORTRAN compilers.

https://powcoder.com

- Maintained by the OpenMP Architecture Review Board.
- Currently up to v5.2, although compilers may only support earlier versions

Add WeChat powcoder

More information available from `http://www.openmp.org`

Compiling C with OpenMP

For this module we will use gcc (GNU Compiler Collection)[1]:

Assignment Project Exam Help

- Compile with -fopenmp
- Must include omp.h

All parallel execution will be undertaken via the Cloud

https://powcoder.com

- We have created a mini-HPC environment using Microsoft Azure.
- You will each get your own INDIVIDUAL account.
- Full instructions for logging into your account will be provided.
- A Linux OS is provided along with the all libraries required for this module:
  - You can run jobs interactively on 2 cores while debugging (can still have more threads!)
  - You can run batch jobs on up to 16 cores via Slurm

Add WeChat powcoder

[1] Easy to install on Macs with homebrew.

helloWorld.c

Code on Minerva: helloWorld.c

Assignment Project Exam Help

```c
#include <stdio.h>
#include <omp.h>    // Run-time OpenMP library routines.

int main()
{
  // The scope after this pragma is in parallel
  #pragma omp parallel
  {
    // Get this thread number, and the maximum.
    int threadNum  = omp_get_thread_num ();
    int maxThreads = omp_get_max_threads();

    // Simple message to stdout.
    printf( "Hello from thread %i of %i!\n", threadNum, maxThreads );
  }
  return 0;
}
```

https://powcoder.com
Add WeChat powcoder

Compiling C: Reminder

Assignment Project Exam Help

If the source code is called `helloWorld.c`:

```
gcc -fopenmp -Wall -o helloWorld helloWorld.c
```

https://powcoder.com

Options:

- `-fopenmp` tells compiler to expect OpenMP `pragma`s.
- `-Wall` turns on all warnings; recommended but not required.
- `-o helloWorld` is the executable name (`a.out` by default).
- `helloWorld.c` is the source code.
- Sometimes need *e.g.* `-lm` for the maths library.

Add WeChat powcoder

#pragma omp parallel

Assignment Project Exam Help

#pragma directives provide information beyond the language

- All OpenMP pragmas start: `#pragma omp ...`

https://powcoder.com

Here, `#pragma omp parallel` is telling the compiler to perform the next scope (*i.e.* the section of code between the curly brackets, from { to }) **in parallel**.

- The code inside this scope is run by **multiple threads**.
- Outside of this scope there is only **one thread**.

Add WeChat powcoder

This is why the `printf` statement is repeated multiple times, **even though it only appears once in code**.

We will look at this in more detail next time.

#include <omp.h>

Assignment Project Exam Help

Include omp.h to use OpenMP runtime library routines:

`int omp_max_thread_num():`

- Returns the maximum number of threads.
- Defaults to **hardware concurrency**, *e.g.* the number of cores.
- May exceed apparent core number with **simultaneous multithreading** (see earlier).

https://powcoder.com
Add WeChat powcoder

`int omp_get_thread_num():`

- Returns the thread number **within the current scope**.
- `0 <= omp_get_thread_num() < omp_max_thread_num()`

Setting the number of threads

Assignment Project Exam Help

If you don't want to use the default number of threads:

void omp_set_num_threads(int):

- Will change the number of threads **dynamically**.
- **Can** exceed hardware concurrency.

https://powcoder.com

Alternatively, use shell **environment variables**:

Add WeChat powcoder

- For bash: export OMP_NUM_THREADS=<num>
- Avoids the need to recompile.
- List all environment variables using env.
- To see all OMP variables: env | grep OMP

Summary and next lecture

Assignment Project Exam Help

Today we have started looking at **s**hared **m**emory **p**arallelism (SMP):

https://powcoder.com

- Relevant to **multi-core CPUs**.
- OS **scheduler** maps **threads** to cores.
- Various languages, frameworks *etc.* support SMP.
- OpenMP is commonly supported by C/C++ compilers.

Add WeChat powcoder

Next time we will look in more detail at what is actually going on at the thread level, for a more interesting example.