High Performance Parallel Programming (CS61064)

Week -1

Pralay Mitra

Course Information

- Instructor: Pralay Mitra (pralay@cse.iitkgp.ac.in)
- TA: Purnima Gautam
- Course coverage:
  - openMP (shared memory) in C programming language
  - MPI (distributed memory) in C programming language

- Video Lectures will be uploaded in YouTube.
- Lecture slides will be uploaded in moodle.
- Query form will be google form.
- Test will be online in moodle.

Course Information

- **References:**

1. “Using OpenMP” by Barbara Chapman, Gabriele Jost and Ruud van der Pas

2. “MPI: The Complete Reference” by Marc Snir, Jack Dongarra, Janusz S. Kowalik, Steven Huss-Lederman, Steve W. Otto, David W. Walker

3. “Parallel Programming with MPI” by Peter Pacheco

Course and Test Plan

Week	Module	Video + Lecture Slides + Query form	Closing of query form at 10am	Live Interaction session
1	Introduction	After the class	NA	04-03-2021 at 5pm
2	openMP I	07-03-2021	11-03-2021	11-03-2021 at 5pm
3	openMP II	14-03-2021	18-03-2021	18-03-2021 at 5pm
4	MPI I	21-03-2021	25-03-2021	25-03-2021 at 5pm
5	MPI II	28-03-2021	01-04-2021	01-04-2021 at 5pm
6	Long Test 2	08-04-2021 from 5pm to 6:30pm at Moodle		

Holidays:
29-03-2021 Holi 02-04-2021 Good Friday 01-04-2021 WB Assembly Election

Why HPC?

- Weather forecast
- Share market forecast
- Many body interaction
- Massive Database search
- High throughput screening
- Intelligent game design
- Real time analysis
- Flexibility over the search space
- Simulation: Atoms to Planets

The first era (1940s-1960s)

Control Data Corporation (CDC) 6600

The Cray Era (1975-1990)

Cray 1, 1976

1980s

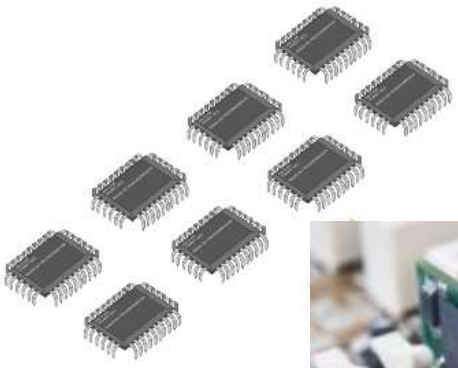
Vectors processors
Shared memory

The cluster Era (1990-2010)

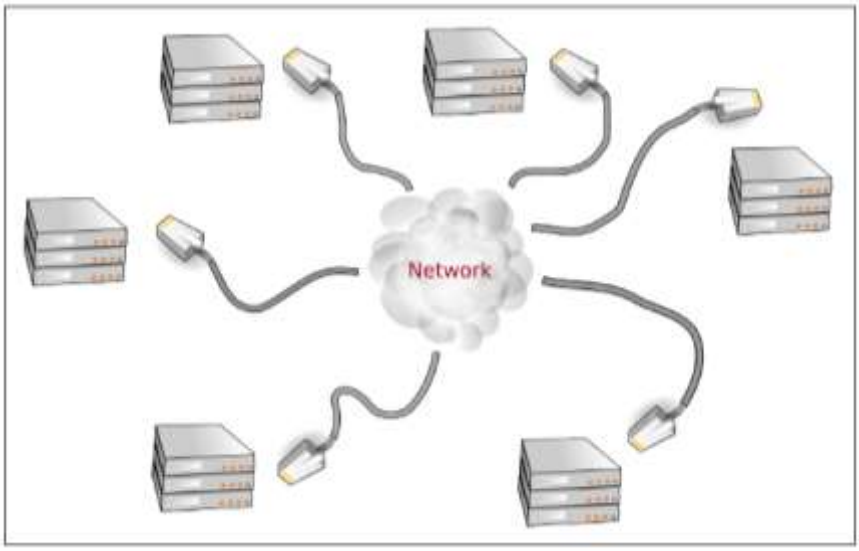
Current Scenario

- The GPGPU and Hybrid Era (2000-
- https://www.top500.org/

Shared or Private



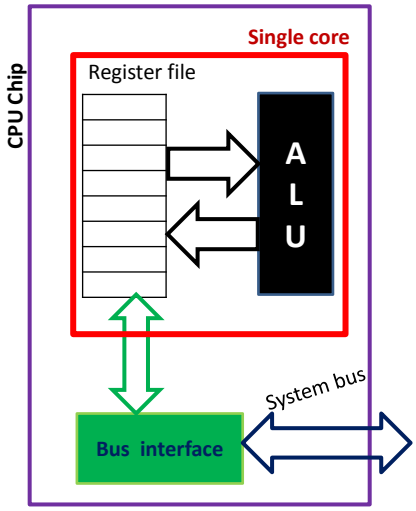
Shared or Private



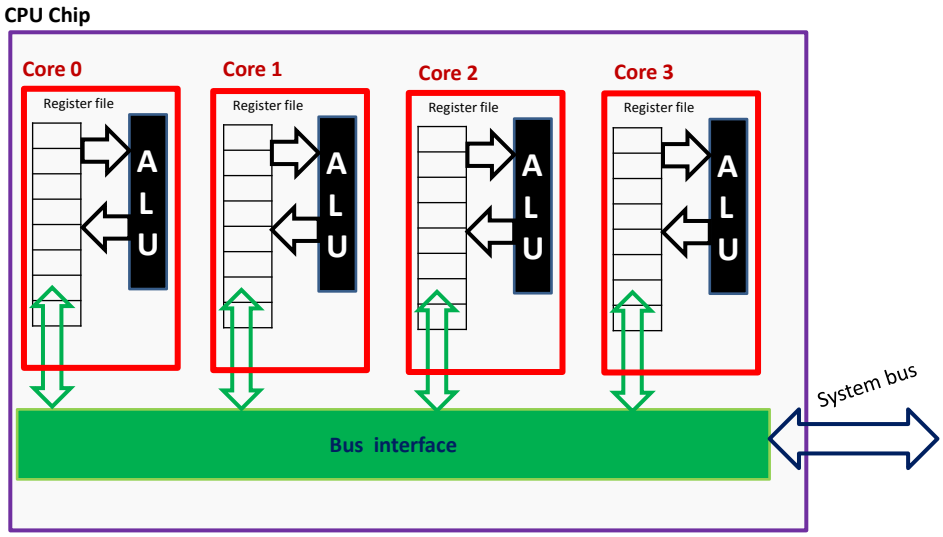
Inside the computer ….

- Core
- Processor
- Single-core
- Multi-core
- …..

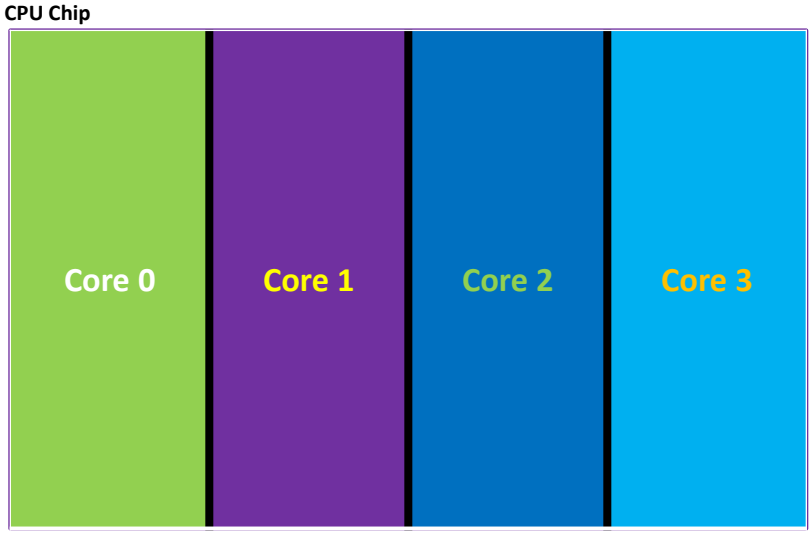
Single-core CPU



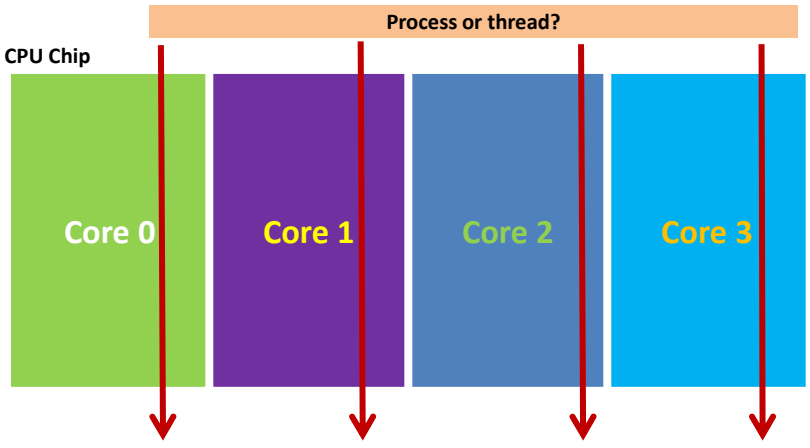
Multi-core CPU



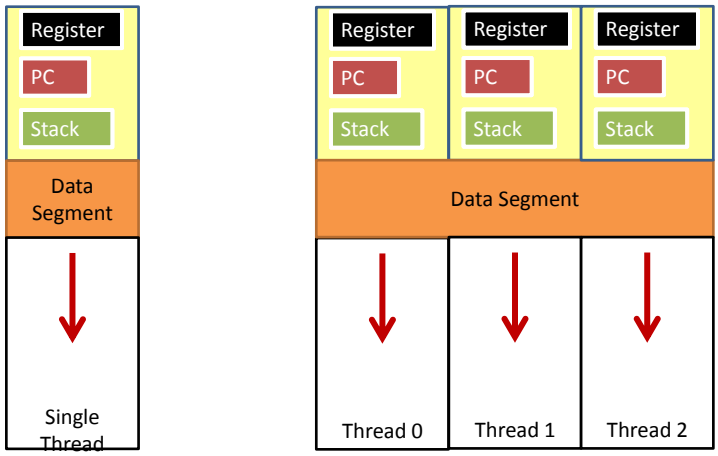
Multi-core CPU



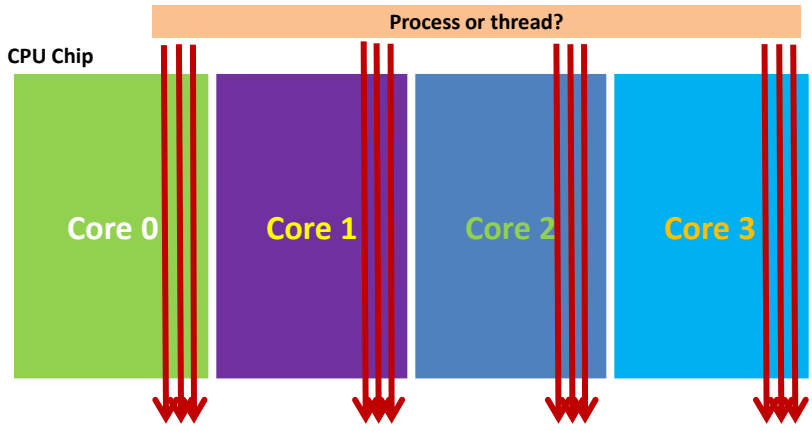
Multi-core CPU



Process vs Thread



Multi-core CPU



An operating system's perspective

Supports multi-core

Perceives each core as a separate processor

Scheduler maps threads/processes to different cores

Multi-core CPU

- ✓ Multi-core processor is a special kind of a multiprocessor where all the processors are on the same chip.
- ✓ Multi-core processors are MIMD where different cores execute different threads (Multiple Instructions), operating on different parts of memory (Multiple Data)
- ✓ Multi-core is a shared memory multiprocessor (SMP) where all cores share the same memory

Multi-core Architecture: Definition

- A multi-core architecture (or a chip multiprocessor) is a general-purpose processor that consists of multiple cores on the same die and can execute programs simultaneously

Multi-core CPU - Applications

- ✓ Database servers
- ✓ Web servers
- ✓ Compilers
- ✓ Multimedia Applications
- ✓ Scientific applications CAD/CAM
- ✓ In general, applications with Thread Level Parallelism (as opposed to instruction level parallelism)

Multi-core CPU – More Applications

- ✓Editing photo while recording a TV show through a digital video converter
- ✓Downloading software while running an anti-virus program
- ✓Anything that can be threaded today will map efficiently to multi-core
- ✓BUT: Not all --- those which are difficult to parallelize

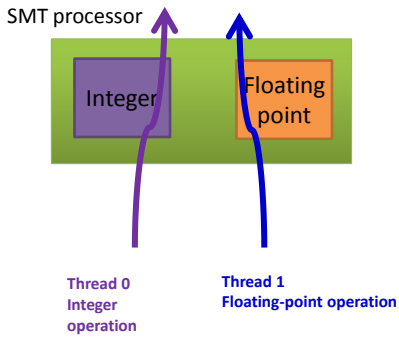
Simultaneous multithreading (SMT)

Permits multiple independent threads to execute simultaneously on the same core

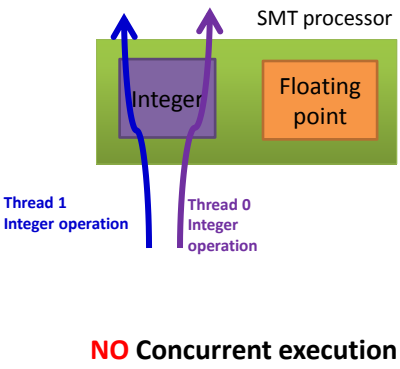
Wearing together multiple “threads” on the same core

Example: if one thread is waiting for a floating point operation to complete, another thread can use the integer units

Simultaneous multithreading (SMT)



Concurrent execution

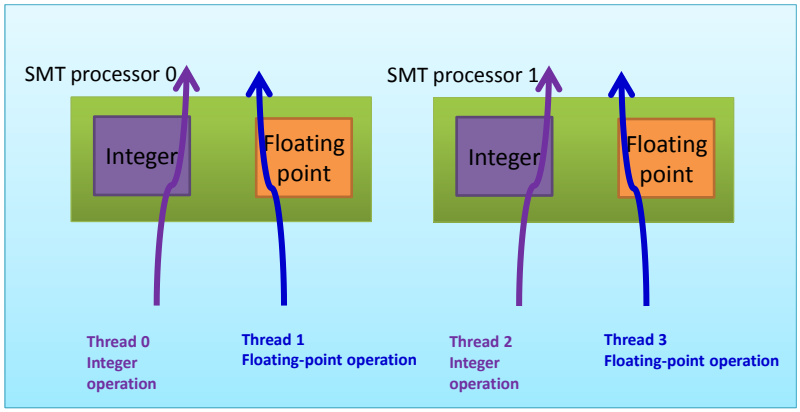


NO Concurrent execution

Combining multi-core and SMT

- Cores can be SMT-enabled (or not)
- The different combinations:
  - Single-core, non-SMT: standard uniprocessor
  - Single-core, with SMT
  - Multi-core, non-SMT
  - Multi-core, with SMT
- The number of SMT threads:
  2, 4, or sometimes 8 simultaneous threads
- Intel calls them "hyper-threads"

SMT Dual-core



Summary on SMT and multi-core

SMT processor: not a true parallel processor

✓Enables better threading (e.g. upto 30%)
✓OS and application perceives each simultaneous thread as a separate "virtual processor"
✓This has only a single copy of each resources
✓Compare to multi-core, in this case each core has its own copy of resources

SMT

✓Great performance on a single thread
✓Mostly exploits instruction-level parallelism

Combining multi-core and SMT
✓Cores can be SMT-enabled (or not)
✓Intel calls them "hyper-threads"

Programming for multi-core

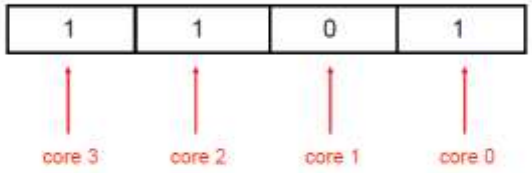
- Programmers must use threads or processes
- Spreads the workload across multiple cores
- Write parallel algorithms
- OS will map thread/processes to cores
- However safety is important:
  - Ready for any time pre-emptive context switching
  - Concurrency bugs exposed much faster with multi-core

How to deal with ….

- **Assigning threads to cores**
  - Each thread has an affinity mask
  - Affinity mask specifies what cores the thread is allowed to run on
  - Different threads can have different masks
  - Affinities are inherited across fork()
- **Affinity masks**
  - Use bit vector for implementing affinity mask
- **Default affinity masks**
  - All threads can run on all processors
  - OS scheduler decides what threads run on what core
  - OS scheduler detects skewed workload and do the allocation

Affinity masks are bit vectors

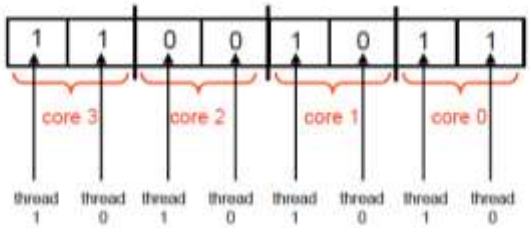
- Example: 4-way multi-core, without SMT



- Process/thread is allowed to run on cores 0,2,3, but not on core 1

Affinity masks when multi-core and SMT combined

- Separate bits for each simultaneous thread
- Example: 4-way multi-core, 2 threads per core



- Core 2 can't run the process
- Core 1 can only use one simultaneous thread

Grandness

- WRF (Weather Research Forecast) ConUS (CONtinental Usa) 2.5km 6hr benchmark
  - Single P6: ~40 hr (though theoretically ~4hr)
  - 4 nodes (128 cores): 0.6 hr
  - 64 nodes (1024 cores): 9 min

Comparison: multi-core vs SMT

- Multi-core:
  - Since there are several cores, each is smaller and not as powerful (but also easier to design and manufacture)
  - However, great with thread-level parallelism
- SMT
  - Can have one large and fast superscalar core
  - Great performance on a single thread
  - Mostly still only exploits instruction-level parallelism

Important: Safety of Thread

- Pre-emptive context switching: context switch can happen AT ANY TIME
- True concurrency, not just uniprocessor time-slicing
- Concurrency bugs exposed much faster with multi-core