CS3211
AY22/23 Sem 2
github.com/SeekSaveServe

Lectures

Introduction

L0 and L1

Program Parallelization

Decomposition: Decompose a sequential algorithm into tasks (programmer)
- Granularity of tasks are important
- Tasks have dependencies (data or control) between each other which defines the execution order

Scheduling: Assign tasks to processes (programmer / compiler)

Mapping - Map processes to cores (OS)

Von Neumann Computation Model instruction and data are stored in memory, and processors computes.

Memory Wall disparity between memory speed and processor speed ($\leq$ 1 ns VS $\geq$ 100 ns)

Processing unit refers to a core that can execute a kernel thread

Interconnect busses betwen different components in the machine

Node Machine in a distributed system

Why Parallel

Primary Reasons

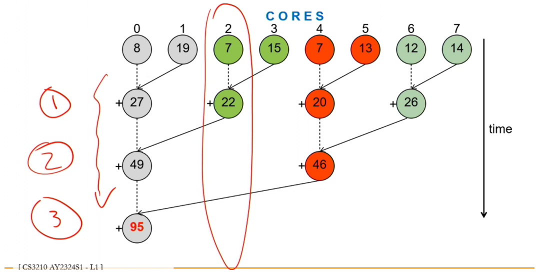
1 OVercome limits of serial computing
2 Solve larger problems
3 Save (wall-clock) time

Other Reasons
- Take advantage of non-local resources
- Cost/energy saving - use multiple cheaper computing resourcees
- Overcome memory constraints

Computational Model Attributes
- **Operation mechanism** Primitive units of computation or basic actions of the computer on a specific Architecture
- **Data Mechanism** How we access and store data in address space
- **Control Mechanism** How primtive units of computation are scheduled
- **Communication Mechanism** Modes and patterns of exchanging information between parallel tasks (e.g message passing, shared memory)
- **Synchronization Mechanism** ensures to ensure needed information arrives at the right time

Dependencies and Coordination
- Dependencies among tasks impose constraints on scheduling
- Memory organizations: Shared-memory (threads), distributed-memory (processes)
- Coordination (synchronisation) imposes additional overheads

Two algorithms



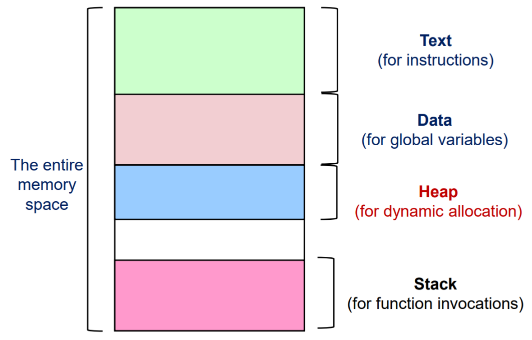
- Core 0 is active throughout the execution
- Some cores are idle
- This is a lot better than having all cores idle while the master core is executing

Parallel Performance
- Execution time Vs Throughput
- Parallel execution time = computation time + parallelization overheads
- Overheads: Distribution of work(tasks) to porocesses, information exchange, synchronisation, idle time, etc

Background on Parallelism

L2: Processes and Threads

Process
- Identified by PID
- Program counter, global data (open files, network connections), stack or heap, current values of the registers (GPRs and Special)
- These information are abstracted in the PCB, and each proecss can be viewed as having exclusive access to tis address space
- Explicit communication is needed
- **Disadvantage**
1. High overhead of system calls
2. Potential re-allocation of data-structures
3. Communication goes through OS (system calls) and context switch is costly



Multi tasking
- Overhead: Context switching (PCB change) is needed and states of suspended process must be saved
- Time slicing: Pseudo-parallelism
- Child processes can use parent's data

Inter-process communication (IPC)
- Shared memory: need to protect access with locks
- Message passing: Blocking, unblocking, Synchronous, unsynchronous

Exceptions
- Executing a **machine level instruction** can cause exception
- For example: Overflow, Underflow, Division by Zero, Illegal memory address, Mis-aligned memory access
- **Synchronous**
  - Occur due to program execution
- Have to execute an **exception handler**

Interrupts
- External events can interrupt the execution of a program
- Usually hardware related: Timer, Mouse Movement, Keyboard Pressed etc
- **Asynchronous**
  - Occur **independently** of program execution
- Have to execute an **interrupt handler**

Threads
- A process may have multiple indepedent control flows called threads
- Each thread has its own stack and registers (PC, SP, registers), but share the same address space
- Shared memory model and Shared memory architecture

Architecture

L3: Processor and memory organization

L7: Cache coherence and memory consistency

L11: Interconnection networks

Parallel Computation Models

L4: Shared-memory programming models

L6: Data parallel models (GPGPU)

L9,10: Distributed-programming models

Performance and Scalability of Parallel Programs

L5: Performance of parallel systems

L8: performance instrumentation

New Trends

L12: Energy efficient computing