Review

May 2023

Slides 1:

- Levels of parallelism
- Amdahl's law

Slides 2:

- Pipelining
  - Dependencies
  - Loop fusion
  - Branch prediction
  - Optimizations

Slides 3:

- Caches
  - Cache hit
  - Cache misses: Capacity, Conflict, Compulsory
  - Temporal locality
  - Spatial locality
  - Cache organization: Direct-mapped, Fully associative, Set-associative
  - Cache organization: Multi-level caches - why?

Slides 4:

- Improving cache performance
  - Cache blocking
  - Examples: Matrix traversal, Jacobi

Slides 5:

- Improving cache performance
  - Examples: Matrix transpose, Matric-Vector, Matrix-Matrix multiplication

---

Slides 6:

- Parallelism: Vector instructions
  - Trust the compiler
  - Assembly language
  - OpenMP SIMD
  - Vector instrincs
- Programmer intervention: __attribute__ __restrict__

Slides 7 and 8:
To know:

- Processes
- Threads
- Hardware and Software threads
- SMT (Simultaneous Multi-Threading) / Hyperthreading - what does it mean?
- Address space of a process
- Role of the Operating System

Multicore programming:

- Cores - do they share caches?
- False sharing - does it affect performance?
- Coherence - what is it?
- Synchronization

Slides 9:

- Coherency and Consistency - what is the difference?

Slides 10 and 11:

- OpenMP
- Scheduling - static, dynamic, guided, auto
- Nested parallelism

Examples:

- Matrix-matrix multiplication
- Jacobi
- SparseMV
- Guass-Seidel

Slides 12: OpenMP: Tasks

- Example: Graph traversal

Slides 13, 14, 15, 16 and 17: To know:

- What does it mean to say - this MPI program is launched on 4 processes?
- Can the 4 processes be located on the same machine?
  - If yes, do they share memory i.e. can a variable declared in one process be accessed by another without using any MPI calls?
  - If no, how do they communicate data?
  - Do they share phyiscal resources like memory and network?

Important functions: MPI_Init MPI_Comm_size MPI_Comm_rank MPI_Send MPI_Recv MPI_Finalize

MPI_Send and MPI_Recv are blocking calls. Read about non-blocking calls. Example: MPI_Isend() MPI_Irecv() Why are non-blocking calls useful?

Other important functions: MPI_Barrier MPI_Bcast MPI_Reduce MPI_Allreduce MPI_Scatter MPI_Gather MPI_Allgather

Questions to ask:

- Can one implement MPI_Barrier or MPI_Bcast using MPI_Send and MPI_Recv?
- How about other functions?

Examples:

- Jacobi
  - Data distribution
  - How did we avoid deadlock?
  - How did we overlap computation and communication?
- Graph traversal
- Jacobi 2D distribution
  - MPI_Type_Vector
  - MPI_Cart_Create
- Matrix product: 1D distribution

Slides 18, 19 and 20:

- We discuss in detail the implementation of Molecular Dynamics using MPI.
- We explore several design choices.

Slides 21:

- MPI + OpenMP
- One-sided communication. Examples:
  - Jacobi 1D tiling
  - Distributed table
- We have slides on other programming paradigms (is not part of exam syllabus, but is useful to know):
  - PGAS
  - Charm++

GPU:
Core ideas:

- What happened to some of the components?
- Fuse fetch-decode logic - what is the consequence of that?
- Store many contexts
- Threads, Blocks, Grid : CUDA cores, warps, SMs
- How is memory organized?
- What is memory coalescing?

Thrust, Streams and Dynamic parallelism - not part of exam syllabus, but useful to know.

GPU OpenMP:
using OpenMP

- omp target
- map
- Jacobi example

Prinicipal Component Analysis example - not part of exam syllabus, but useful to understand how to use OpenMP in GPU better.

MPI-IO: Each process makes independent request; Collective I/O: All processes make a single request.

---

openmp_2 in General Resources: OpenMP example: Merge Sort - not part of exam syllabus, but useful to know.
