Issues in Parallel Programming

29 January 2009
CMPT370
Dr. Sean Ho
Trinity Western University

CAREER FAIR

<u>Event Details</u>
Feb. 5, 2009
11:30 am – 3:00 pm

Larson Atrium
Reimer Student Centre,
TWU Campus
7600 Glover Road
Langley, BC V2Y 1Y1

Attire:
Business Professional

Open to all TWU Students and Alumni.

You are invited to attend the 2009 TWU Career Fair. The Fair is a chance to meet prospective employers for full-time careers, internships, co-ops, and seasonal positions. Check website for participating employers. **Register at the Student Portal.**

www.twu.ca/life/career/fair
604-513-2017
career@twu.ca

2009 Career Fair Exhibitors

- Acts Seminaries
- Adventure Teaching
- African Children's Choir
- Angus One Professional Recruitment
- BC Corrections
- BC Ferries
- BC Human Resources Management Association
- BC Transit Police
- Bethesda Christian Association
- BOP Korea Connections
- Canada Revenue Agency
- Canadian Forces
- Certified General Accountants
- Certified Management Accountants
- Communitas Supportive Care Society
- Community Employment Resource Centre
- Corporate Express
- Correctional Service Canada
- Creative Memories
- Developmental Disabilities Association
- Dulay Burke Financial Recruitment
- Edward Jones
- EV Logistics
- Fraser Health Authority
- Freedom 55 Financial
- Institute of Chartered Accountants of BC
- Logos Bible Software
- Kintec Footlabs
- Meyers Norris Penny LLP
- New Westminster Police
- Northern Health
- Pampered Chef
- Power to Change
- Retirement Concepts
- Royal Bank of Canada
- Royal Canadian Mounted Police
- Southwestern Company
- Studibudi Professional Networking In
- Sun Life Financial
- TD Canada Trust
- Township of Langley
- UHY International
- Vancouver Fire and Rescue Services

TRINITY WESTERN UNIVERSITY

Career Fair 2009 – February 5, 11:30am-3:00pm
Larson Atrium, Reimer Student Centre
www.twu.ca/life/career/fair

Pre-Career Fair Workshops: to be announced on website.

Lab2: Your OpenMP program

- Ideas:
  - Numerical integration (like pi-leibniz.c)
  - Generating fractals:
    - See mandelbrot/ example
  - Dictionary/brute force encryption cracking?
  - Prime number generation?

Review last time

- OpenMP
  - #pragma omp parallel: begin parallel section
    - #pragma omp for
    - #pragma omp sections
  - Shared vs. private variables
    - private(), reduction()
  - #pragma omp critical/single/barrier
  - OMP_NUM_THREADS,omp_get_num_threads()
  - Timing: omp_get_wtime()
  - schedule(static/dynamic/guided/runtime)

Addendum: timing

- <time.h> clock(): CPU time w/ tick resolution
  - Usually 100 or 1000 ticks/sec
- <time.h> time(): wall-clock time w/ second res
- <sys/time.h> gettimeofday():
  - Wall-clock time in seconds w/ tick resolution
- double omp_get_wtime():
  - Wall-clock time in seconds w/ tick resolution
  - Platform independent
  - Thread dependent

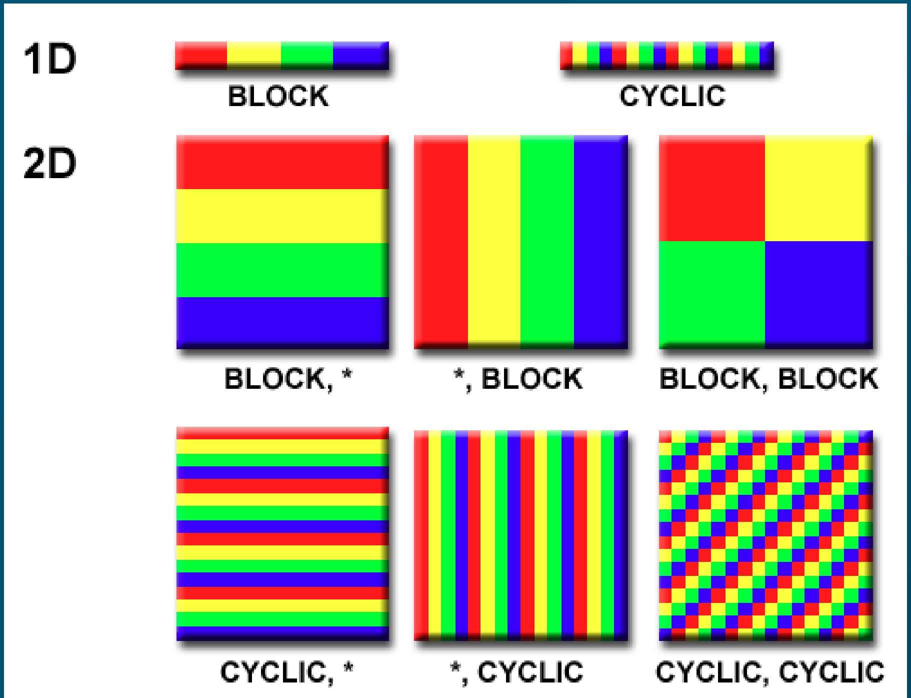
Issues in parallel prog. design

- When designing parallel programs:
  - Understand the problem to be solved
    - Data dependencies
  - Partition the work
    - Domain vs. functional partitioning
    - Granularity
  - Communications!
    - Synchronization
  - Load balancing
  - I/O

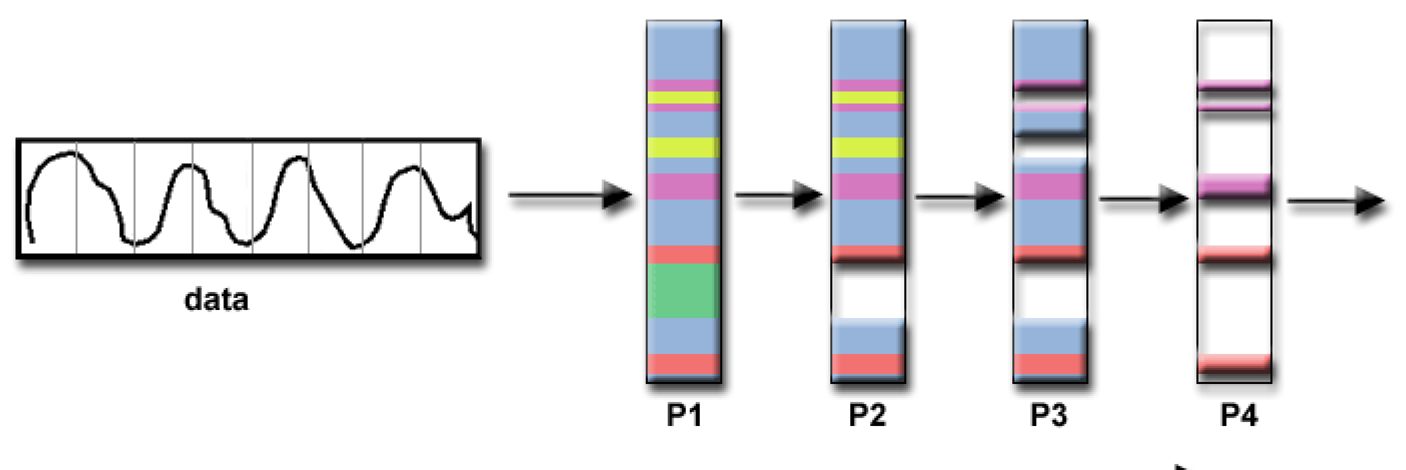
Understand the problem

- Parallelizable:
    - e.g., find next position of 500k atoms
- Not (easily) parallelizable:
    - e.g., find Fibonacci sequence:
        - **fib(n) = fib(n-2) + fib(n-1)**
    - Data dependence
- Find bottlenecks to performance
    - Optimize inner loops
    - Use profiling when necessary

Partition the work



- Data partitioning
  - "SIMD" philosophy
  - Each task operates on part of dataset
- Functional decomposition ("MIMD" philosophy)
  - Each task does different work
  - e.g., series of audio filters



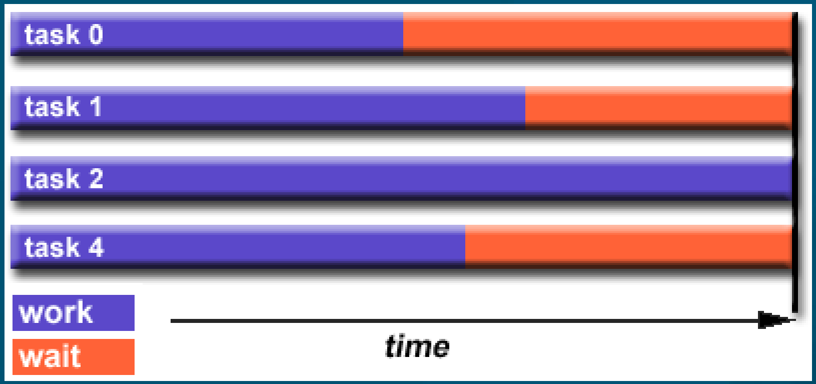
Communication issues

- I/O often bottleneck: minimize communication
  - Coarse-grain parallelism: e.g., FoldingAtHome
- Latency vs. bandwidth
- Unicast (point-to-point) vs. multicast
- Synchronous (blocking) (“phone call”) vs. asynchronous (non-blocking) (“letter”)
- Ease of programming
  - OpenMP abstracts away from programmer
  - MPI makes communication more explicit

Types of synchronization

- Barrier
  - Wait for all tasks to catch up: slowest task
  - Implicit barrier at end of each parallel section
- Lock (semaphore/mutex)
  - Only one thread can hold the lock at a time
  - Wait (block or non-block) for lock to free
  - e.g., #pragma omp critical
- Synchronous communications
  - Both parties must synchronize
  - Blocking → idle CPU → bad!

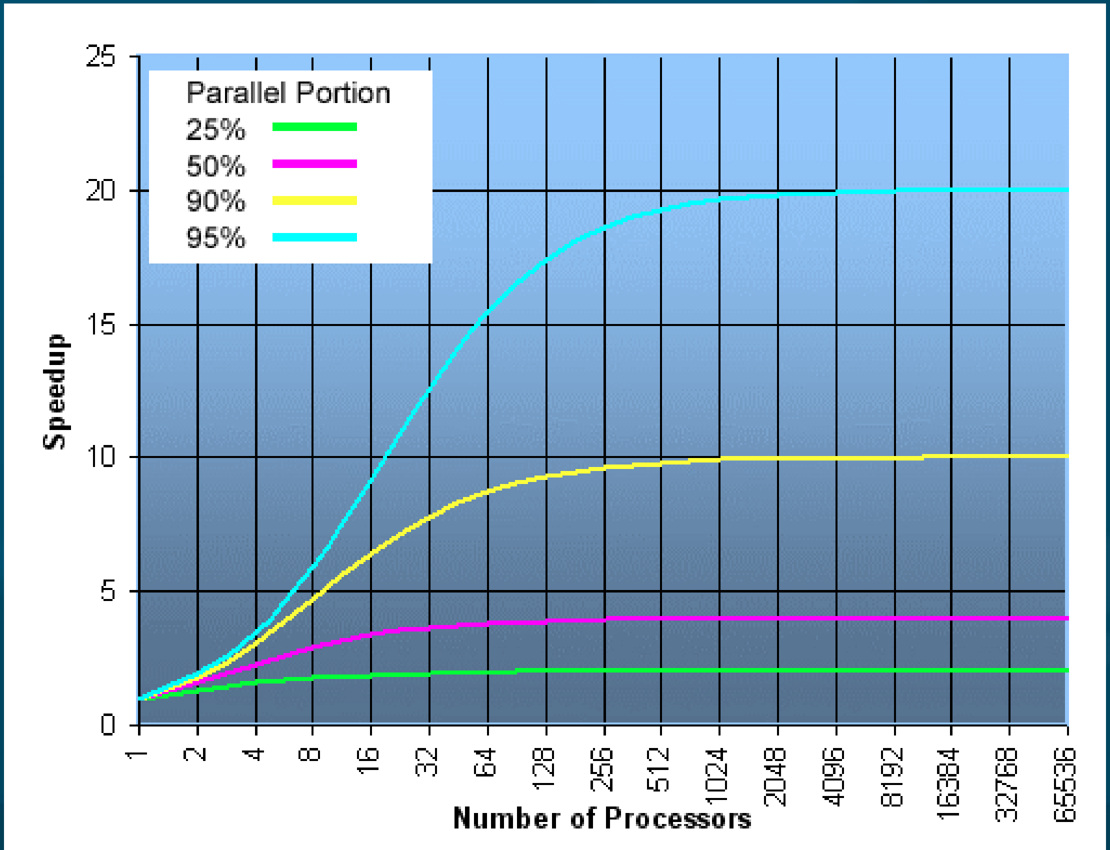
Load balancing



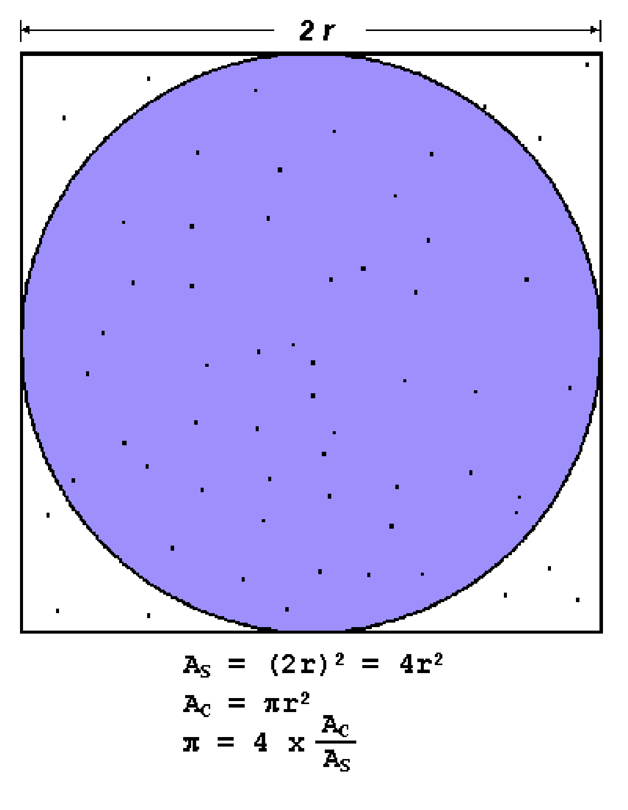
- Want to keep all processors busy as much as possible
- Easy if tasks are fixed in computation time:
  - Array/matrix operations
  - Loops (#pragma omp for)
- Often hard to predict how long a task will take
  - Sparse arrays: most entries zero
  - Adaptive refinement
  - N-body (e.g., gravity) simulations

Amdahl's law: speedup

- If P is the fraction of code that is parallelizable and N is the number of processors:
- Speedup = 1 / (P/N + (1-P))
- The fraction of code that is parallelizable is very important!
- Limits maximum speedup



Sample application: Pi

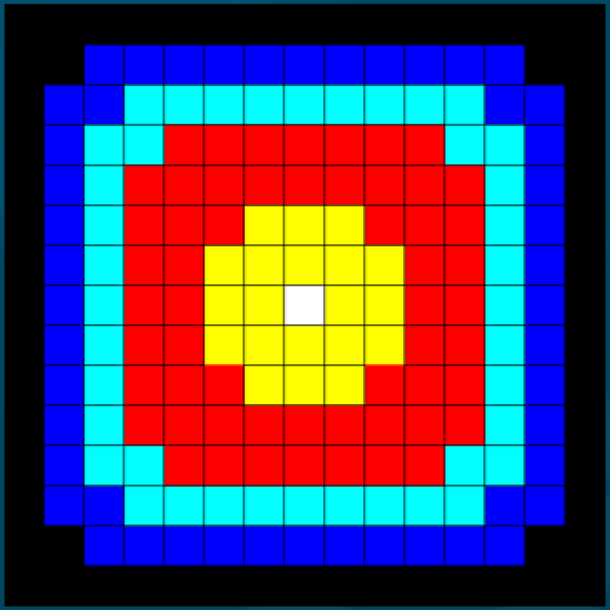


- Generate random dot
- See if it's within unit circle
- Fraction of dots inside circle approximates pi/4
- The more dots, the better precision

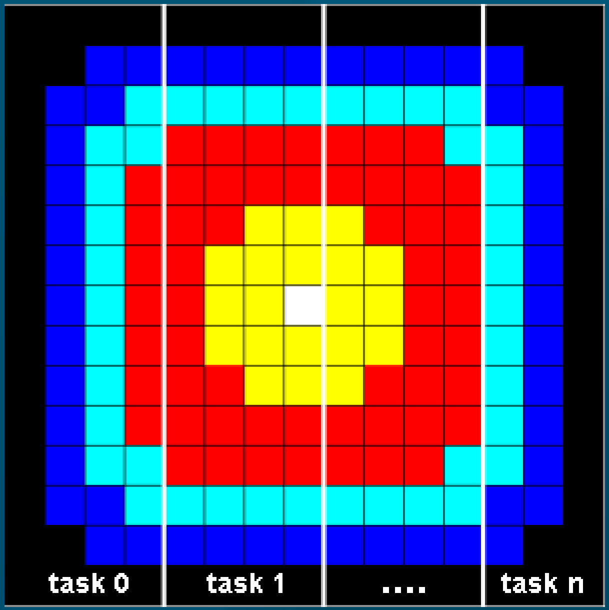
- Parallelizable? What does each thread do?
- Granularity?

Data Parallelism

- Some programs are embarassingly parallel:
  - e.g., add two vectors
- Data dependency: order of execution matters
- Some are parallelizable, but need some communication:
  - Heat equation on a 2D grid
  - Each pixel $U_{x,y}$ +=
    - $C_x (U_{x-1,y} + U_{x+1,y} - 2*U_{x,y}) +$
    - $C_y (U_{x,y-1} + U_{x,y+1} - 2*U_{x,y})$
  - Used for blurring images



Heat equation: boundaries



- Divide work by region of image:
  - Data parallelism
- Interior of region can be done independently
- Boundaries need information from neighbouring threads
- Use non-blocking communication to send/receive boundary pixels from neighbours while processing interior