Homework

• 1\textsuperscript{st} Due yesterday
  ➢ 16 of 22 are turned in
  ➢ If you haven’t turned it in and have not talked to me about it. You need to see me ASAP.

• Any Questions about this homework?

• 2\textsuperscript{nd} homework
  ➢ Can be done in Groups
  ➢ I determine the Groups!
    ▪ Please send me an email
      ♦ Subject: group request
    ▪ State if you want to work in a group or alone
Logistics

• Project for the course.
  ➢ Pick a High Performance Computing subject area.
    ▪ This is open to you!!
    ▪ You have to pick the area of interest!!
  ➢ Get two or one good papers in that area.
    ▪ Make an office hour appointment or email papers to me.
      ♦ The same homework teams are project teams.
      ♦ All team members must be at this meeting.
    ▪ Bring your copies and copies of these papers to me for review.
    ▪ We will decide the view or “tact” to take on these or
      ♦ Bring me another rock.
      ♦ Choose “wisely”
  ➢ You will need both a written and oral report for this.
    ▪ Analysis of what is right/wrong
    ▪ Directions where things could be improved
    ▪ What’s missing in the Author’s analysis.
    ▪ How do things work!!
Logistics [2]

- Example for Project (use at your own risk 😊)
  - Area: Computational Chemistry
  - Papers:

- Proposed angle on Project
  - Kinds of algorithms used in Computational Chemistry
  - Kinds of tools used in NWChem
  - Scalability of these algorithms on various architectures
  - Identify the weak link of these applications.

Projects MUST be approved before Spring Break!!!

1/21/2005       ComS 425       Spring 2005       4 of 42: Lecture 5
Parallel Computer Models

• Previously we saw Flynn’s Taxonomy (a Model)
• The von Neumann SISD (a Model)
• An abstract view of the architecture or programming mechanism
• Characterizes capabilities of the machine
  ➢ Usually with some simplifications
    ▪ Ignores actual processor implementation(s)
    ▪ Ignores communication network topology
• Precise enough to think about performance but ignores explicit implementation details.
• Helps users/developers understand the performance of their applications.
Parallel Computer Models

• Defined by coupled, semantic and performance attributes.
  ➢ Think of a parallel computer model as a way of grouping processes to concurrently solve a problem.

• Semantic Attributes
  ➢ Needed to effectively write a parallel program
  ➢ “thinking in parallel”
  ➢ Specific Attributes:
    ▪ Homegeneity
    ▪ Interaction Mechanism
    ▪ Memory Model
    ▪ Synchrony
    ▪ Address Space
Semantic Attributes: Homogeneity

- Is essentially Flynn’s Taxonomy
- How “alike” the processes are on a processor
  - IBM SP can have 5 different kinds of nodes
  - Clusters can come in all shapes and sizes
- Remember that most parallel applications run the same program:
  - Single-Program with Multiple-Data (SPMD)
  - Data shared among interacting processes
  - Work partitioned based on the “identity” of the process
  - See the Example Homework Problem for one way to do this.
Semantic Attributes: Synchrony

• How tightly synchronized the processes are.
• Synchronous Model: Synchronized at the instruction level (SIMD)
• Asynchronous: Synchronized at various periods through some mechanism or not at all.
  ➢ If none then “truly Asynchronous” (MIMD)
  ➢ A group of instructions (SMP) “loosley synchronous”
    ▪ All memory read operations from all processes must complete before a memory write
  ➢ At a given point synchronization is initiated by the application. “loosley synchronous”
    ▪ SPMD on a MIMD computer.
Sem.Att.: Interaction Mechanism

• How processes interact to do work or share data.
  ➢ Shared Variables (e.g., shared memory)
    ▪ multiprocessor
  ➢ Message Passing (scheduled by the application)
    ▪ multicomputer
Semantic Attributes: Address Space

• The memory locations accessible to a process.
• Single address space
  ➢ Seen by all interacting processes
  ➢ A multiprocessor (SMP)
• Multiple address space
  ➢ Seen only by a single process (private address space).
  ➢ A multicompactor (MIMD, NOW, COW)
  ➢ Explicit data management (message passing)
Semantic Attributes: Memory Model

• **Access time to Memory:**
  - **UMA – Uniform Memory Access**
    - Flat. Same time from all processes to memory
  - **NUMA – Non Uniform Memory Access**
    - Path to some memory different than other memory
    - Time difference for access can be up to 3 orders of magnitude.
  - **Distributed Shared Memory (NUMA)**
    - Physically Distributed Memory (local and remote)
    - Hardware and/or software support for addressing remote memory
Semantic Attributes: Memory Model (2)

• Consistency rules:
  ➢ EREW (Exclusive Read Exclusive Write)
    ▪ On any cycle only one process can read or write to a memory location
  ➢ CREW (Concurrent Read Exclusive Write)
    ▪ On any cycle multiple processes can read but only one can write to a memory location
  ➢ CRCW (Concurrent Read Concurrent Write)
    ▪ On any cycle multiple processes can read or write to a memory location.
    ▪ Concurrent Writes require special consideration.
Semantic Attributes: Memory Model (3)

- **Atomic Operations**
  - **Indivisible:**
    - Once started, it cannot be interrupted
    - Intermediate state or data not available to any other entity.
  - **Finite:**
    - Once started it will finish in finite time
Semantic Attributes: Memory Model (4)

- **Transactions (another kind of operation).**
  - **Atomicity** (all or nothing)
  - **Consistency**
    - Transfer from one state to another
    - Dependent upon nature of the program
  - **Isolation**
    - Results not “posted” or “revealed” until they are committed e.g., at the final State.
  - **Durability (PERSISTENCE)**
    - Once transaction is complete, it persists even if the system fails.
Performance Attributes

• Definitions

  ➢ Machine Size
  ➢ Sequential Time
  ➢ Parallel Time
  ➢ Speed
  ➢ Peak Speed
  ➢ Speed Up
  ➢ Efficiency
  ➢ Startup time
  ➢ Bandwidth
  ➢ Asymptotic Bandwidth

  \( n, \quad N_p, \quad N_{\text{proc}} \)
  \( T_1, \quad T_s \)
  \( T_n, \quad T_p \)
  \( P_n, \quad R_n, \quad \text{MFR} = \text{flops/time} \)
  \( P_{\text{peak}}, \quad R_{\text{macho}} \)
  \( S_n = \frac{T_s}{T_p} = \frac{T_1}{T_n} \)
  \( E_n = \frac{S_n}{n} = \frac{S_n}{N_p} \)
  \( t_0, \quad \tau_0 \)
  \( r \)
  \( r_{\infty} \)
Kinds of Operations

- **Computation**
  - Anything that exists in a traditional sequential program
  - Arithmetic/logic
  - Control Flow (branches)
  - Data transfer (local memory copy, disk I/O)

- **Parallelism**
  - Startup/shutdown of processes
  - Grouping or work flow designation (What do I do?)

- **Interaction**
  - Data transfer among processes
  - Synchronization
Operations Context

- **Operations can be either:**
  - **Explicit**
    - Appears as a construct in the program
    - Handled primarily by the application developer
    - E.g., `exit()`, `A = B+C`, `fork()`
  - **Implicit**
    - Does not appear as a construct in your program
    - May appear as a construct in libraries you use:
      - MPI, GA, etc.
    - Handled primarily by the Operating System or your Library Developer or compiler.
    - E.g., Broadcast implies a synchronization.
Overhead (The pain in Parallel)

• **Parallelism**
  - Process initiation/termination
  - Constructs from Libraries you may use

• **Communication**
  - Data exchange operations
    - Collective
    - Non-collective (e.g., point-to-point)

• **Synchronization**
  - Synch operations

• **Load Imbalance**
  - Some processes not doing useful work
Overhead (The pain in Parallel) [2]

- Without overhead algorithms would work perfectly.
  - The time to compute on 2 processors would always be $\frac{1}{2}$ the time on 1 processor.

- Most frequently used parameters in dealing with overhead:
  - Latency
    - Time to communicate a 0-byte message
    - Minimum communication time for any message
  - Asymptotic Bandwidth
    - Rate at which message are sent
    - Message size dependency
Abstract Machine Models

• A more global view of both semantic and performance attributes

• Three models to examine, increasingly more complex.
  ➢ Parallel Random Access Machine Model (PRAM)
  ➢ Bulk Synchronous Parallel Model (BSP)
  ➢ Phase Parallel Model (PPM)
PRAM

- The simplest model
  - MIMD or SIMD
  - Fine Grained
  - Tight synchronization at each cycle with no overhead
  - Shared variables in a Single address space
  - UMA with EREW or CREW
  - Arbitrary size of machine with cycle based time.
  - One cycle is one operation
    - Fetch from or store to memory, arithmetic/logic
  - Load imbalance is only overhead available
DDOT on PRAM

- DDOT or the dot product, inner product, or norm of a vector.
- \( \text{Norm} = \sum A_j A_j \), \( j = 1 \), Length of \( A \) \( \equiv N \)
  - 2N operations (one multiply and add for each)
- In the PRAM model.
  - Local sum of \( 2N/N_{\text{proc}} \) operations on each processor
  - Binary tree reduction of each Local Sum \( \log(N_{\text{proc}}) \)
  - Total time is: \( 2N/N_{\text{proc}} + \log(N_{\text{proc}}) \)
  - Speed up: \( T_s/T_p = (2N)/[2N/N_{\text{proc}} + \log(N_{\text{proc}})] \)
  - \( = N_p/[1+(N_p/2N)\log N_p] \)
Bulk Synchronous Parallel (BSP)

\[ w = \text{Maximum computation time within each superstep} \]
\[ l = \text{Barrier synchronization overhead} \]
\[ g = \text{“h” relation coefficient (abstract communication parameter)} \]
Bulk Synchronous Parallel (BSP)

- **MIMD no SIMD**
- **Superstep (Maximum cycles)**
  - Computation (maximum \(w\))
  - Communication (maximum \(gh\))
  - Barrier (maximum \(l\))
  - Total time is at worst \((w + gh + l)\)
  - Total time is at best \(\max(w, gh, l)\)
- **Variable Grain (Fine or Coarse)**
- **Loosely synchronous**
- **Nonzero overheads**
- **Message Passing or Shared Variable**
An “h” relation

• Abstraction of a communication operation
• Each node sends at most h words to others
• Each node receives at most h words
• Total cycles is g•h
  ➢ g is a proportionality constant based on the type of machine.
  ➢ g is small for systems with good networks
  ➢ g is large for poor networks
BSP norm of a vector

• Assume a 4 processor BSP system

• Superstep 1
  - Local Sum = \( \sum A_j A_j \text{ of } \frac{2N}{N_p} \) [w = 2N/4 cycles]
  - Proc 1 and 3 send local sum to 0 and 2 [g cycles]
  - Synchronize (l cycles)

• Superstep 2
  - Sum partial components 0 += 1  2+=3 [w = 1 cycle]
  - Proc 2 sends partial sum to 0 [g cycles]
  - Synchronize (l cycles)
BSP norm of a vector (2)

- **Superstep 3**
  - Sum partial components \( 0 += 2 \ [w = 1] \)
  - Formally done
  - Most algorithms would have to:
    - Broadcast result to all nodes so SPMD can make the same right decision on all nodes.

- **Total Time** = \( 2N/4 + 2g + 2l + 2 \) cycles
- **Total Time** = \( 2N/N_p + \log N_p (g+l+1) \)
- **Compare PRAM:** \( 2N/N_p + \log N_p \)
Norm with Vector size 1000

![Graph showing time to solution vs. number of processors for PRAM and BSP (good and bad) models.](image-url)
Norm with Vector size 1000
Norm with Vector size 100000
Norm with Vector size 100000

SPEEDUP

- SU[PRAM]
- SU [BSP(good)]
- SU [BSP(bad)]
- Ideal

speedup

number of processors

1/21/2005  ComS 425  Spring 2005  31 of 42: Lecture 5
Theoretical Computational Models

- Parallel Random Access Model
- Bulk Synchronous Processing.
- PRAM and BSP time to compute the norm of a vector.
  - BSP Total Time = $2N/N_p + \log N_p (g+l+1)$
  - PRAM Total Time: $2N/N_p + \log N_p$
Phase Parallel Model

Communications Network
Phase Parallel Model

- First model that is not the Crudest Reasonable Approximation (CRAP)
- Similar to the BSP model
- Program is executed as a sequence of phases that each must complete before the next phase begins (sequence is also called a superstep).
  - Parallelism Phase (parallel overhead)
  - Computation Phase (local computations)
  - Interaction Phase (communication, synchronization, reduction or aggregation)
Phase Parallel Parameters

- **Computation Phase**
  - $w$ the workload in Mflops
  - $t_f$ the average time to execute a computation operation (seconds/flop)
  - Both $w$ and $t_f$ are phase dependent!
    - Each phase may have different values!!
Phase Parallel Parameters (2)

• **Interaction Phase**
  - General form

\[ T_{\text{interact}}(m,n) = t_0(N_p) + m/r_\infty(N_p) \]

\[ = t_0(N_p) + m\cdot t_c(N_p) \]

- \( t_0(N_p) \equiv \text{The startup time} \)
- \( m \equiv \text{The message length (bytes)} \)
- \( r_\infty(N_p) \equiv \text{Asymptotic Bandwidth} \)
- \( t_c(N_p) \equiv \text{per-byte message time} \)
  - reciprocal Asymptotic Bandwidth
The Entire Phase Model Time

- $T_{\text{comp}} = (w + \sigma \sqrt{2\log N_p})t_f$
- $T_{\text{par}} = t_p(N_p)$
- $T_{\text{interact}} = t_0(N_p) + \alpha wt_c(N_p)$
  - $w$ mean per node work load (granularity)
  - $\sigma$ standard deviation of work load (load imbalance)
  - $t_f$ average time to compute operation
  - $t_p$ parallel overhead
  - $t_0$ startup time
  - $\alpha$ communication to compute ratio
  - $t_c$ per byte message time.

- $T_{\text{total}} = T_{\text{comp}} + T_{\text{par}} + T_{\text{interact}}$
The Entire Phase Model Time [2]

\[ T_{\text{total}} = T_{\text{comp}} + T_{\text{par}} + T_{\text{interact}} \]

\[ T_{\text{total}} = (w + \sigma \sqrt{2\log N_p}) t_f + t_p(N_p) + t_0(N_p) + \alpha w t_c(N_p) \]

- Covers overheads
  - Load imbalance
  - Parallelism
  - Communication and synchronization
Vector Norm using Phase Parallel

ParallelFor (i=0; i<Np; i++) {

   \[ (i+1)N_p \]
   \[ \text{localsum} = \sum_{j=iN_p}^{(i+1)N_p} A[j]A[j]; \]
   \[ j=iN_p \]

   \[ \text{sum} = \text{sum\_reduction(localsum)}; \]
}

• Parallelism \( t_p(N_p) \) (different for threads/procs)
• Computation: \( (2N/Np)/t_f = (2Nt_f)/Np \)
• Interaction \( t_0(N_p) \) (network/library dependent)
Computational Complexity

• Assuming a PRAM model (no overheads) and three algorithms that have the following computational complexities:

  - A \( 7N_p \quad O(N_p) \)
  - B \( \frac{N_p \log N_p}{4} \quad O(N_p \log N_p) \)
  - C \( N_p \log \log N_p \quad O(N_p \log \log N_p) \)

• Consider both the real and traditional BigO scaling.

  - Note BigO implies asymptotic behavior \((N_p \to \infty)\)

• \( N_p \log N_p \) is “good” scaling
Scaling [Big O()]

- Np log logNp
- Np
- Np log Np

Time vs. Number of processors graph.
Scaling [without BigO()]

- $7N_p$
- $(N_p \log(N_p))/4$
- $N_p \log \log(N_p)$

The graph shows the relationship between the number of processors and the time in seconds for different scaling functions.