Announcements

- **Final exam** will be held
  Thursday, June 12th, 3PM to 6PM
- Closed book, closed notes
- Calculators will not be permitted in the exam.
- You are responsible for all material covered in the course, including all assigned readings, classroom discussions, and the homework

Terms and concepts

- Know the definition and significance of …
- Parallel speedup, superlinear speedup, scaled speedup, efficiency, scalability, cost, cost-efficient, inefficiency
- Amdahl’s law, serial bottlenecks
- SPMD, SIMD, MIMD, Client server
- Granularity, surface-to-volume effect
- Multiprocessors and multicomputers: shared memory, message passing
- Interconnect
  - Hypercube, ring, bisection bandwidth, diameter
  - Message startup, half power point, peak bandwidth,
- Shared memory
  - Cache coherence, cache consistency
  - False sharing, write policies (update v. invalidate)
  - Sneaking protocol (biased coherence)
  - UMA (SMP) vs CC-NUMA designs
  - Critical sections, nondeterminacy, atomicity, locks, barriers

Implementation techniques

- Message passing with MPI
  - asynchronous, blocking and non-blocking communication
  - collective communication: reduction, broadcast, barrier synchronization, all to all (total exchange)
  - communicators and datatypes
- Shared memory programming and architecture
  - Performance issues: false sharing, write policies
  - Pthreads and OpenMP interfaces
- Data parallel programming
  - forall semantics, data layouts, implicit communication
- Load balancing
  - blocked data decompositions (uniform and irregular)
  - cyclic decomposition, recursive coordinate bisection (orthogonal recursive bisection), self-scheduling, client/server computation, implicit vs. explicit load balancing
- Performance modeling and performance measurement techniques
Algorithms

- Know the purpose of the following algorithms, and the significant implementation issues affecting performance. Be familiar with performance models for each and be prepared to analyze performance and scalability.

- **ODE solver**
  - The algorithm entails repeatedly updating a set of points on 1-dimensional. Each point is updated using nearest neighbors only.
  - To enable parallelization, we separate the points into even and odd sets. We maintain ghost cells holding off processor values.

- **Particle method**
  - Ring and chaining mesh strategy (you’ll only be asked to analyze performance of the ring algorithm).

- **Matrix algorithms**
  - Matrix multiplication (Canon’s algorithm), LU decomposition

- **Fast fourier transform**
  - Transpose algorithm, multidimensional transform

- **Sorting**
  - Enumeration sort, odd-even sort, odd-even transposition sort, shell sort, bucket sort, sample sort (just higher level view)
  - Be prepared to analyze performance for all but sample sort

- **Collective communication**:
  - Ring algorithm, hypercube algorithm as appropriate for reduction/broadcast.

Workloads for problem 1

- **Pick the preferred design for each of two workloads**
- **A:** Multiple serial computations with small amounts of input (haiku is poetry with only 17 syllables), 30 sec of computation, small output (a single score)

- **B:** LU decomposition on 70,000 x 70,000 matrix

Practice problems

- **Problem 1: Matching applications to architecture**
  - **UMA.net**—a Uniform Memory Access (UMA) multiprocessor with 64 processors and 32 GB of shared memory.
  - **Distributed.net**—loose connection of 145K processors with 9 TB RAM connected via the internet with 56.6kbit/s connections, avg average of 800ms between processors.

Problem 2 – shared memory programming

- List all possible outputs that result when the following OpenMP annotated C code is executed:

```c
#pragma omp parallel for shared(j,k) private(i)
for ( int i=0, j=0, k=0; i< 2; i++ )
  j = j + 10;
  k = j + 10;
}
cout << "k = " << k << endl;
```
Problem 2 – barrier synchronization

• Explain how the code works, demonstrating correct operation on 3 processes
• Why are two lock variables needed, arrival and departure
• Initialization?

```c
void barrier(...)
{
    lock (arrival);
    count++;
    if (count < n$proc) unlock (arrival);
    else unlock (departure);
    lock (departure);
    count--;
    if (count > 0) unlock (departure);
    else unlock (arrival);
    return;
}
```

Problem 3

• Which program is faster?

Program 1:
(1) for (i=0; i<n; i++)
(2) \[ F += \text{abs}(X[i] - X); \]

Program 2:
(1) \[ T = X; \]
(2) for (i=0; i<n; i++)
(3) \[ F += \text{abs}(X - T); \]
(4) \[ T = \text{CSHIFT}(T, 1); \]
(5) \[

Problem 4

• Distribute two arrays over 4 processors
  \[ \text{double x[16], y[16];} \]
• With BLOCK decomposition, how many values stored on P_0, including ghost cells?
• How many data values must be sent by processor 3 to execute the following loop correctly across the 4 processors?
  \[ \text{forall (i=0; i<15; i++)} \]
  \[ y[i] = x[i] + x[i+1]; \]
• Repeat with CYCLIC decomposition