Lecture 17

Load Balancing and Irregular Problems
Announcements

• Quiz
• CAPE!
• In class problems
In class problem solving
Prefix sum

- The prefix sum (also called a sum-scan) of a sequence of numbers $x_k$ is in turn a sequence of running sums $S_k$ defined as follows
  \[ S_0 = 0, \quad S_k = S_{k-1} + x_k \]
- Thus, scan (3,1,4,0,2) = (3,4,8,8,10)
- Design an algorithm for prefix sum on the hypercube
- Use the gray coding to define the layout of values across processors
Hypercube algorithm

• Hint: need a second variable

Prefix sum algorithm

1. procedure PREFIX_SUMS_HCUBE(my_id, my_number, d, result)
2. begin
3.   result := my_number;
4.   msg := result;
5.   for i := 0 to d − 1 do
6.     partner := my_id XOR 2^i;
7.     send msg to partner;
8.     receive number from partner;
9.     msg := msg + number;
10.    if (partner < my_id) then result := result + number;
11.   endfor;
12. end PREFIX_SUMS_HCUBE
Time constrained scaling

- Sum $N$ numbers on $P$ processors
- Let $N \gg P$
- Determine the largest problem that can be solved in time $T=10^4$ time units on 512 processors
- Let the startup time be $\alpha$ time units
- Let $\beta = 1/(\text{peak bandwidth})$
- Assume that $\beta = \text{time to perform one addition}$
Performance model

- Local additions: $N/P - 1$
- Reduction: $(1+\alpha)(\log P - 1)$
- $T(N,P) = N/P + \alpha \log P$
- Determine the largest problem that can be solved in time $T=10^4$ time units on $P=512$ processors, $\alpha = 10$ time units, and addition costs one unit of time
- Consider $T(512,N) \leq 10^4$
  $\Rightarrow (N/512) + \alpha \log(512) \leq 10^4$
  $\Rightarrow (N/512) + 90 \leq 10^4$
  $\Rightarrow N \leq 5\times10^6$ (approximately)
Load Balancing
Today’s lecture

• Workload assignment and decomposition
• Different strategies
  ‣ Cyclic Decomposition
  ‣ Self scheduling
• Load balancing
• Motivating applications
Task Decomposition

• How to subdivide the computation and assign to processors?
  ‣ *Decomposition* or *partitioning*
  ‣ *Processor mapping*
  ‣ Dynamic or static?
• Functional decomposition
  ‣ *Task parallelism*
  ‣ *Pipelining*
• Data decomposition
  ‣ The most common technique in parallel computing
  ‣ Are the data decomposed uniformly or non-uniformly?
Irregular Problems

- In the Jacobi application, computational effort is applied uniformly.
- **Irregular** applications apply computational effort non-uniformly.
- A load balancing problem arises when partitioning the data.
- There may not be a mesh.

Courtesy of Randy Bank
Multiblock meshes

Courtesy Mary Wheeler, University of Texas, Austin
Load Balancing of Particle Methods
The N-body problem

• Compute trajectories (in time) of a system of N bodies, moving under mutual influence
  ‣ No general analytic (exact) solution when N > 2
  ‣ Numerical simulations required
• N can ranges from thousands to millions
• A force law governs the way the particles interact
• We may not need to perform all $O(N^2)$ force computations
• Introduces non-uniformity due to uneven distributions
Discretization

• Particles move continuously through space and time
• Because we cannot solve the problem analytically we must solve it numerically
• On a computer we represent continuous values using discrete approximation
The calculation

• Evaluate force field at discrete points in time \( \Delta t, 2\Delta t, 3\Delta t, \ldots \)
  › \( \Delta t \) is called the *time step* (a parameter)

• “Push” the bodies according to Newton’s third law
  \( \mathbf{F} = m\mathbf{a} = m \frac{d\mathbf{u}}{dt} \)

```plaintext
while (current time < end time)
    forall bodies \( i \in 1:N \)
        compute force \( F_i \) induced by all bodies \( j \in 1:N \)
        update position \( x_i \) by \( F_i \Delta t \) forall \( i \)
    current time += \( \Delta t \)
end
```

©2010 Scott B. Baden / CSE 160 / Winter 2010
Timestep selection

- We approximate the velocity of a particle by the tangent to the particle’s trajectory.
- Since we compute velocities at discrete points in space and time, we approximate the true trajectory by a straight line.
- So long as $\Delta t$ is small enough, the resultant error is reasonable.
- If not, then the particle might “jump” to another trajectory: this is an error.

Particle A’s trajectory

Particle B’s trajectory
Computing the force

- The running time of the computation is dominated by the force computation, so we ignore the push phase.
- The simplest approach is to use the direct method, with a running time of $O(N^2)$
  
  \[
  \text{Force on particle } i = \sum_{j=0}^{N-1} F(x_i, x_j)
  \]
- $F(\ )$ is the force law.
- One example is the gravitational force law
  
  \[
  G \frac{m_i \ m_j}{r_{ij}^2} \text{ where } r_{ij} = \text{dist}(x_i, x_j)
  \]
- $G$ is the gravitational constant.
Localized Van Der Waals Force

\[ F(r) = \begin{cases} 
C \left( 2 - \frac{1}{30r} \right) & r < \delta \\
\frac{C}{(30r)^5} & r \geq \delta 
\end{cases} \]

\[ C = 0.1 \]
Implementation

• We don’t compute all $O(N^2)$ interactions
• To speed up the search for nearby particles, sort into a *chaining mesh* (Hockney & Eastwood, 1981)
• Compute forces one mesh box at a time
• Only consider particles in the 8 surrounding cells

*Jim Demmel, U. C. Berkeley*
Parallel Implementation

• As with stencil methods, use ghost cells to store copies of nearby particles
• The ghost region also manages particles that have moved outside the subdomain and must be repatriated to their new owner
Decomposition

- Boxes carry varying amounts of work depending on local particle density
- A uniform partitioning will result in a load imbalance
Static decomposition

• **Block cyclic** decomposition
• As with LU decomposition
• All processors should get about the same amount of work if we choose a reasonable chunk size
• What are the tradeoffs?
Communication overhead

- With a $b=2$, each processor must obtain all the simulation data.
- With $b=4$, this factor decreases to $\frac{1}{2}$.
- If the dependence distance increases, so must the granularity.

Jim Demmel, U. C. Berkeley
Non-uniform partitioning

• Cyclic partitioning is a static strategy
  ‣ Partitions space uniformly and maps work irregularly
  ‣ A statistical sampling method -- does not attempt to measure load imbalance
  ‣ May incur high communication overheads due to fine grain data decomposition
• Another approach is to employ partitioning rather than mapping to mitigate load imbalance
• We partition the work non-uniformly according to the spatial workload
  ‣ Each non-uniform region carries an equal amount of work
  ‣ Particles move: partitioning changes over time
• Avoids high surface to volume ratio of cyclic decomposition
The workload density distribution

- Irregular partitioning algorithms like RCB require that we estimate the workload in space and time
- Embed the problem in a discrete space and construct a workload density mapping \( \rho(x,t) \), that giving the workload at each point \( x=(x,y,z) \) of the discrete problem
- In static problems the workload density mapping depends only on position
- In many applications we can come up with a good estimate of the mapping
Significance of Locality

• Many physical problems exhibit locality in space and time
• The values of the solution at nearby points in space and time are closer than for values at distant points
• Timesteps are “small:” to avoid introducing unacceptable error, the solution changes gradually
• Particle motion is localized in space and in time
• These conditions hold for continuum methods such as classical partial differential equations
• We can adjust the partitioning frequency accordingly
Non-uniform blocked decomposition

• A well known partitioning technique is recursive coordinate bisection (RCB)
An example of RCB in one dimension

• Consider the following loop

\[
\text{for } i = 0 : N-1 \text{ do} \\
\quad \text{if } ( G(i) ) \\
\quad \quad \text{then } y[i] = f1( \ldots ); \\
\quad \quad \text{else } y[i] = f2( \ldots ); \\
\text{end for}
\]

• Assume \( f1( ) \) takes twice as long to compute as \( f2( ) \)
Partitioning procedure

• Let \( W[i] = \textbf{if} \ (G(i)) \ \textbf{then} \ 2 \ \textbf{else} \ 1 \)
  
  \[
  1 \ 1 \ 2 \ 1 \ 2 \ 2 \ 2 \ 1 \ 1 \ 1 \ 1 \ 1 \ 2 \ 2 \ 1
  \]

• Compute the running sum (i.e. the \textbf{scan}) of \( W \)
  
  \[
  1 \ 2 \ 4 \ 5 \ 7 \ 9 \ 11 \ 12 \ 13 \ 14 \ 15 \ 16 \ 17 \ 19 \ 21 \ 22
  \]

• Split into 2 equal parts, at \( 22/2 = 11 \)
  
  \[
  1 \ 1 \ 2 \ 1 \ 2 \ 2 \ 2 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 2 \ 2 \ 1
  \]
Recursive coordinate bisection

• Recurse until the desired number of partitions have been rendered
  
  1 1 2 1 2 2 2 1 1 1 1 1 1 2 2 1

• May be applied in multiple dimensions
Load Balancing with space filling curves

• Multidimensional RCB suffers from granularity problems
• We can reduce the granularity with many partitions per processor
• Introduces higher surface to volume effects
• Another approach: spacefilling curves
Partitioning with filling curves

• Maps higher dimensional physical space onto the line
  ‣ Load balancing in one dimension
  ‣ Many types;
    *Hilbert* curves shown here
• Irregular communication surfaces
• Useful for managing locality: memory, databases

http://www.math.ohio-state.edu/~fiedorow/math655/Peano.html
mathforum.org/advanced/robertd/lsys3d.html
Load balancing efficiency

- If we ignore serial sections and other overheads, then we may express load imbalance in terms of a **load balancing efficiency** metric.
- Let each processor $i$ complete its assigned work in time $T_i$.
- Thus, the running time $T_P = \text{MAX} \ (T_i)$.

Define $\bar{T} = \sum_i T_i$.

We define the **load balancing efficiency** $\eta = \frac{\bar{T}}{PT_P}$.

- Ideally $\eta = 1.0$. 

©2010 Scott B. Baden / CSE 160 / Winter 2010
Sparse Matrices
Sparse Matrices

• A matrix where knowledge about the location of the non-zeroes is useful
• Consider the a 5-point stencil

\[ u'[i,j] = (u[i-1,j] + u[i+1,j] + u[i,j-1] + u[i,j+1]) / 4 \]
Savings with Sparse Matrices

- 7 x 7 grid: 49 x 49 matrix
- Nonzeroes: 349, 30% of the space
- Flops: 2643, 6.7%
- Fill-in: a zero entry becomes nonzero
Web connectivity Matrix: 1M x 1M

1M x 1M submatrix of the web connectivity graph, constructed from an archive at the Stanford WebBase

3 nonzeroes/row

Jim Demmel
Motorola Circuit

$170,998^2$
958,936 nonzeroes
.003% nonzeroes
5.6 nonzeroes/row

www.cise.ufl.edu/research/sparse/matrices/Hamm/scircuit.html
Sparse Matrix Vector Multiplication

• Important kernel used in sparse linear algebra $y[i] += A[i,j] \times x[j]$

• Many formats, common format for CPUs is Compressed Sparse Row (CSR)

Jim Demmel

©2010 Scott B. Baden / CSE 160 / Winter 2010