A Case Study in Domain Decomposition

Shirley Moore

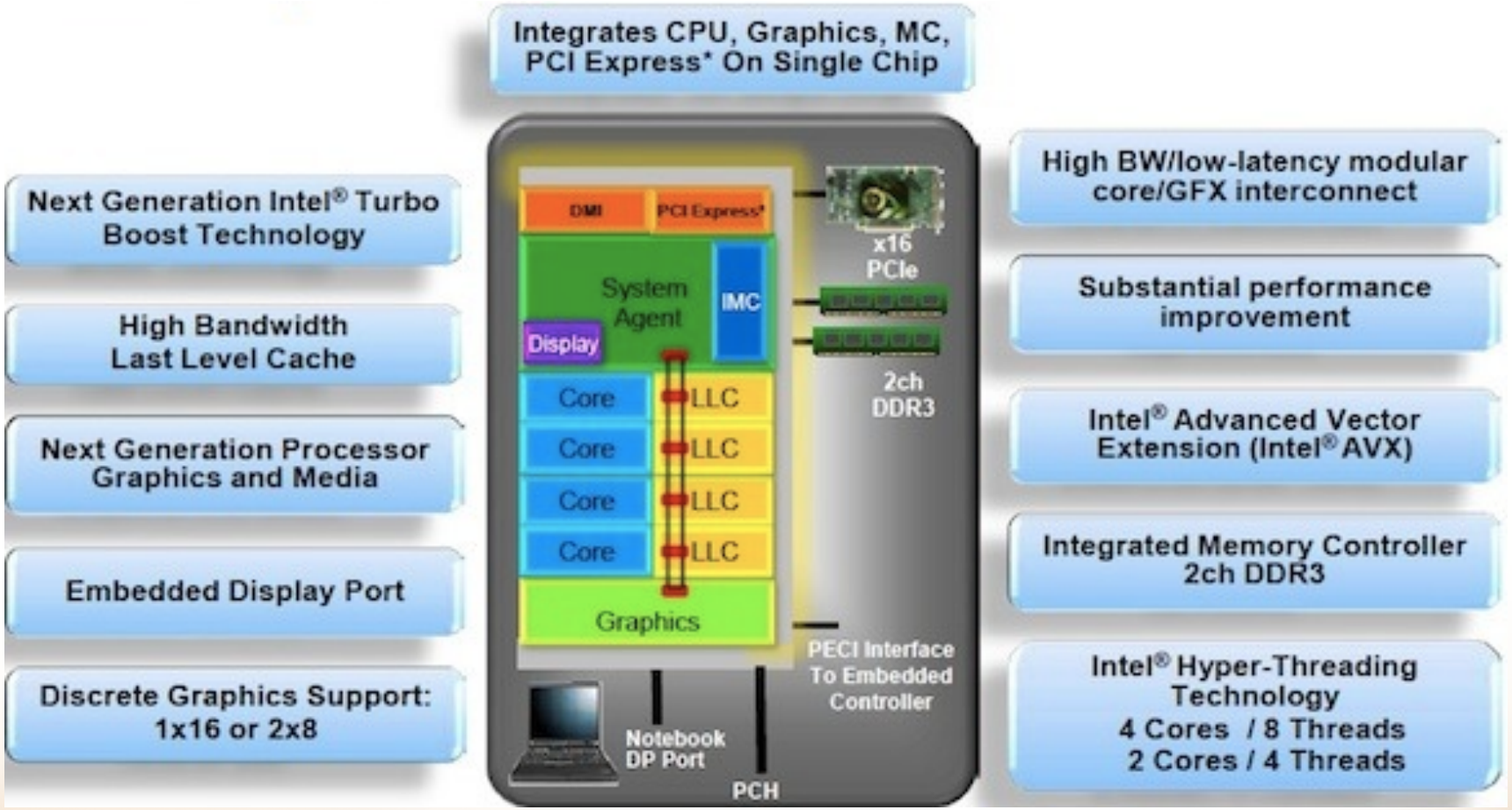
shirley@eecs.utk.edu

CS594 29 Feb 2012

Domain Decomposition

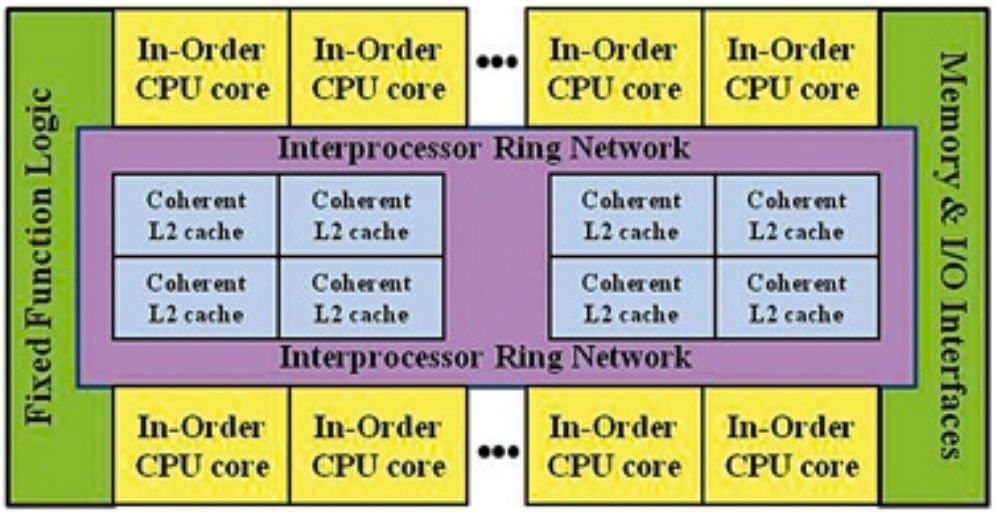
- First step in writing a parallel program
- Identifies work to be performed by each unit of parallel execution
- Should exploit inherent parallelism of the problem
- Alternative decompositions may all be correct but may exhibit very different performance behavior.
- We will consider the domain decomposition for a thread parallel program on a shared memory system.
- Multicore and many-core nodes are becoming pervasive.
  - We must think parallel!

Intel Sandy Bridge



Intel Many Integrated Core (MIC) Architecture

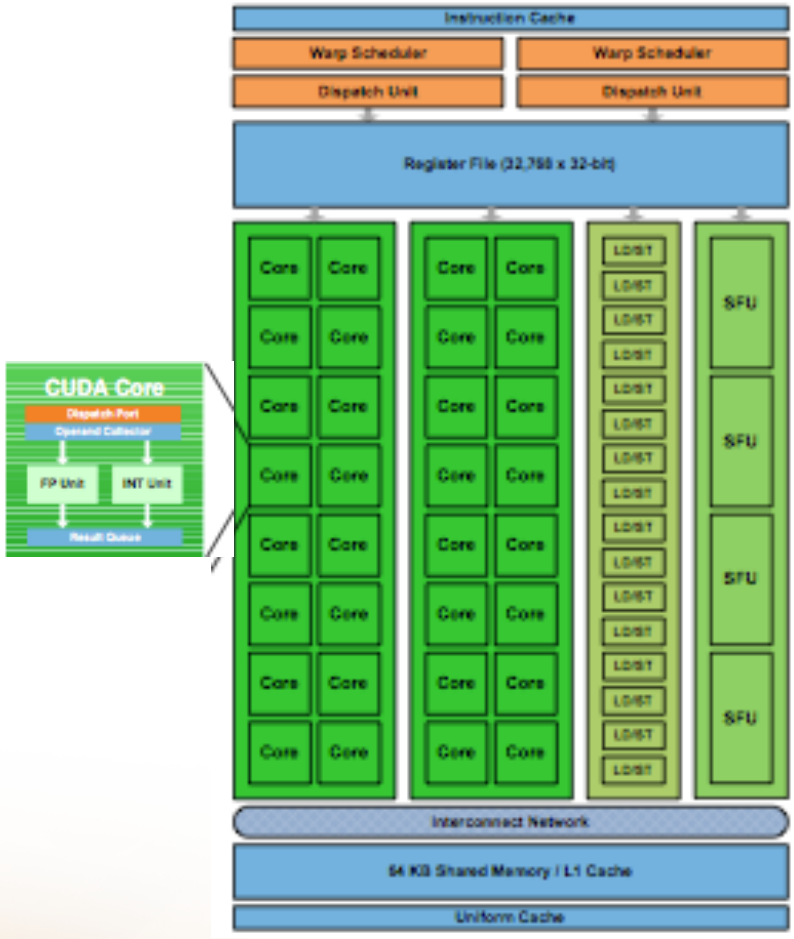
- 32 in-order x86-based processor cores augmented by a 512 bit wide vector processing unit (VPU)



ICL UT

NVIDIA Fermi Streaming Multiprocessor

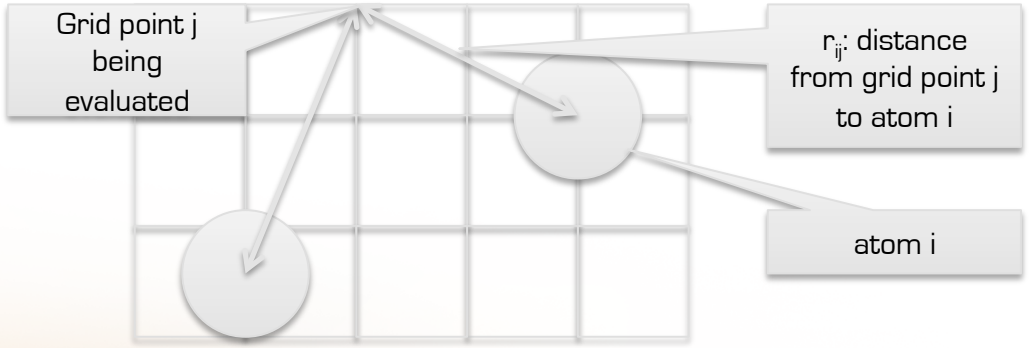
Tesla M2090: 512 cores, 665 Gflops peak performance



ICL UT

Case Study: Electrostatic Potential Calculation

- Used for placement of ions into a grid structure for molecular dynamics simulation
- We will use the direct Coulomb summation (DCS) method for calculating the electrostatic potential map.
  - Calculates the electrostatic potential value of each grid point as the sum of contributions from all atoms in the system.



Direct Coulomb Summation

- Contribution of atom i to electrostatic potential energy at gird point j is the charge of atom i divided by the distance from point j to atom i
- Number of operations is proportional to the product of the number of atoms times the number of grid points
  - Can be very large for realistic molecular system

Pseudocode for DCS

```
for each atom[i]
    for each gridpoint[j]
        dx = gridpoint[j].x – atom[i].x
        dy = gridpoint[i].y – atom[i].y
        energy[j] += atom[i].charge/sqrt(dx*dx + dy*dy)
```

OR

```
for each gridpoint[j]
    for each atom[i]
        dx = gridpoint[j].x – atom[i].x
        dy = gridpoint[j].y – atom[i].y
        energy[j] += atom[i].charge/sqrt(dx*dx + dy*dy)
```

Are the above equivalent in terms of correctness?

Two alternative domain decompositions for thread parallelism

1. Use each thread to calculate the contribution of one atom to all grid points

```
for each gridpoint[j]
    dx = gridpoint[j].x – atom[threadindex].x
    dy = gridpoint[j].y – atom[threadindex].y
    energy[j] += atom[threadindex].charge/sqrt(dx*dx+dy*dy);
```

2. Use each thread to accumulate the contributions of all atoms to one grid point

```
for each atom[i]
    dx = threadindex.x*gridspacing – atom[i].x
    dy = threadindex.y*gridspacing – atom[i].y
    energy[threadindex.x,threadindex.y] +=
        atom[i].charge/sqrt(dx*dx + dy*dy)
```

Group Work

- Divide into small groups
- Assume threads access the atom, gridpoint, and energy arrays stored in shared memory
- Assume a thread writing to a shared memory location must have exclusive access to that location (e.g., by using atomic memory operations)
- Assume no cache effects.
- Assume number of atoms is of the same order as the number of grid points.
- Which domain decomposition alternative would result in better performance on a shared memory architecture? Explain you answer.

Class Discussion

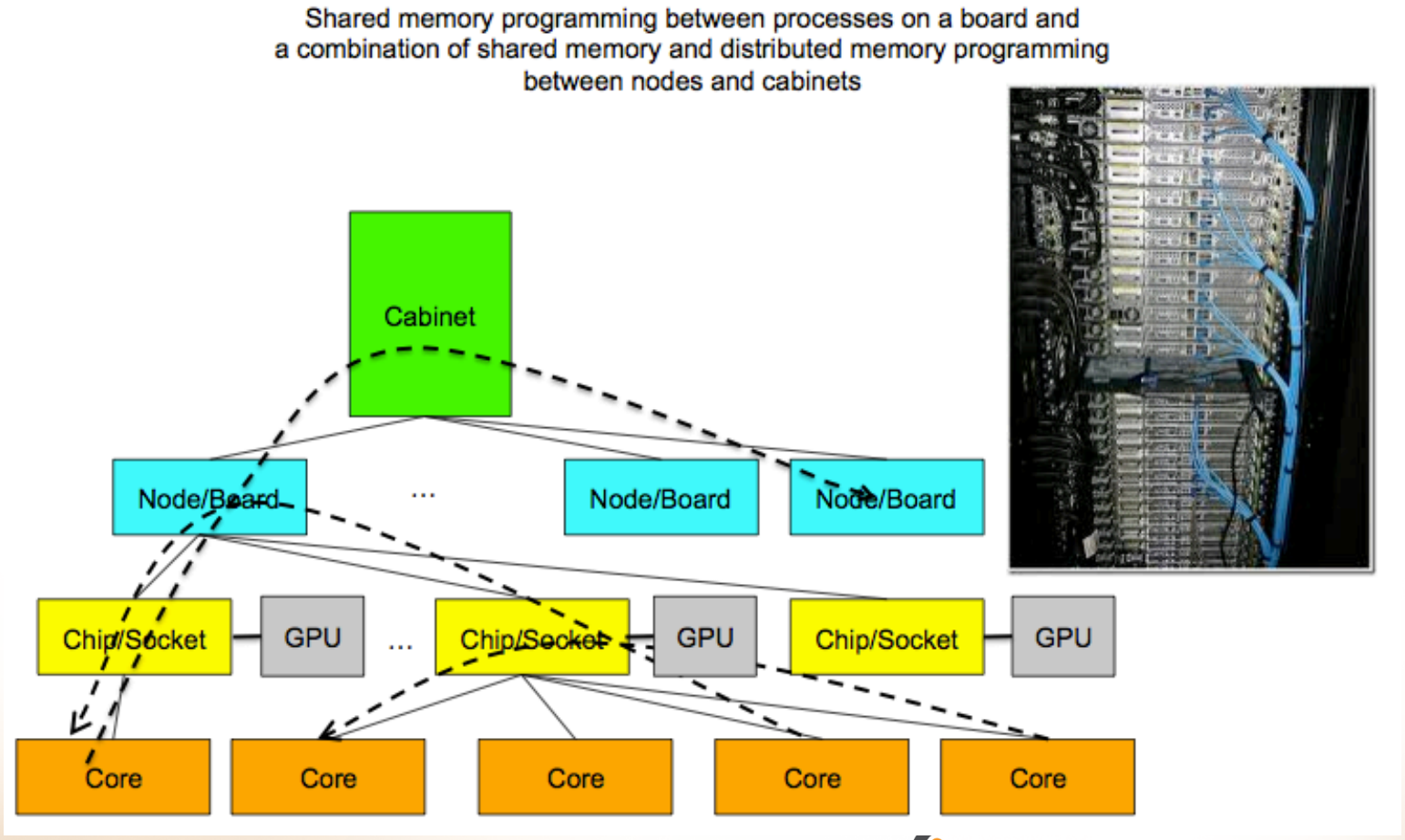
Gather vs. Scatter

- Accumulating information from multiple locations into a single location is called a *gather*.
- Distributing information from a single location to multiple locations is called a *scatter*.
- Which type of operation is carried out by each thread in each of the two alternative domain decompositions?
- Which type of operation is in general more efficient when carried out in parallel by each thread?

Homework Questions

1. Let's assume that when we implement a thread parallel DCS calculation using our chosen domain decomposition on a GPU, the performance is 186 gigaflops. Express the performance in terms of the number of atom evaluations per second, a metric that is likely to be of more interest to an application scientist.
2. Look carefully at the pseudocode for our chosen domain decomposition. What operations are being done redundantly by the threads? How could you modify the domain decomposition to reduce the number of redundant operations and achieve a higher value for the atom evaluations per second metric, assuming that the flops metric stays approximately the same or improves?

Hierarchical heterogeneous architectures



Homework Questions (cont.)

3. We have simplified the problem by assuming a 2 dimensional grid. In a real system, we would have a 3 dimensional grid. Assume that we want to implement the 3 dimensional electrostatic potential map calculation on a hierarchical distributed computer system with shared memory nodes. How can we use the 2D decomposition we have already worked out to construct a 3D decomposition?