SDSU CS 505 Lectures Slides
Spring 2002

Instructors:

Amit Majumdar

Faramarz Valafar

SAN DIEGO STATE UNIVERSITY

Introduction

(Review of serial computing as needed)

- Parallel Computing – Real Life Scenario
- Cost Effectiveness of Parallel Processors
- What is parallel computing?
- Why do parallel computing?
- Types of parallel computing
- What are some limits of parallel computing?

Parallel Computing – Real Life Scenario

- **Stacking or reshelving of a set of library books. Assume books are organized into shelves and shelves are grouped into bays.**

Single worker can only do it in a certain rate.

We can speed it up by employing multiple workers.

What is the best strategy ?

1. **Simple way is to divide the total books equally among workers. Each worker stacks the books one at a time. Worker must walk all over the library.**
2. **Alternate way is to assign fixed disjoint sets of bay to each worker. Each worker is assigned equal # of books arbitrarily. Workers stack books in their bays or pass to another worker responsible for the bay it belongs to.**

Parallel Computing – Real Life Scenario

- **Parallel processing allows to accomplish a task faster by dividing the work into a set of substacks assigned to multiple workers.**
- **Assigning a set of books to workers is *task partitioning*. Passing of books to each other is an example of *communication* between subtasks.**
- **For some problems assigning work to multiple workers might be more time consuming than doing it locally.**
- **Some problems may be completely serial; e.g. digging a post hole. Poorly suited to parallel processing.**
- **All problems are not equally amenable to parallel processing.**

Wheather Modelling and Forcasting

Consider 3000 X 3000 miles, and height of 11 miles.

- **For modeling partition into segments of 0.1X0.1X0.1 cubic miles = ~10^{11} segments.**

Lets take 2-day period and parameters need to be computed every 30 min. Assume the computations take 100 instrs. A single update takes 10^{13} instrs. For two days we have total instrs. of 10^{15} . For serial computer with 10^{10} instrs./sec, this takes 280 hrs to predict next 48 hrs !!

Lets take 1000 processors capable of 10^{8} instrs/sec. Each processor will do 10^{8} segments. For 2 days we have 10^{12} instrs. Calculation done in 3 hrs !!

Currently all major weather forcast centers (US, Europe, Asia) have supercomputers with 1000s of processors.

Other Examples

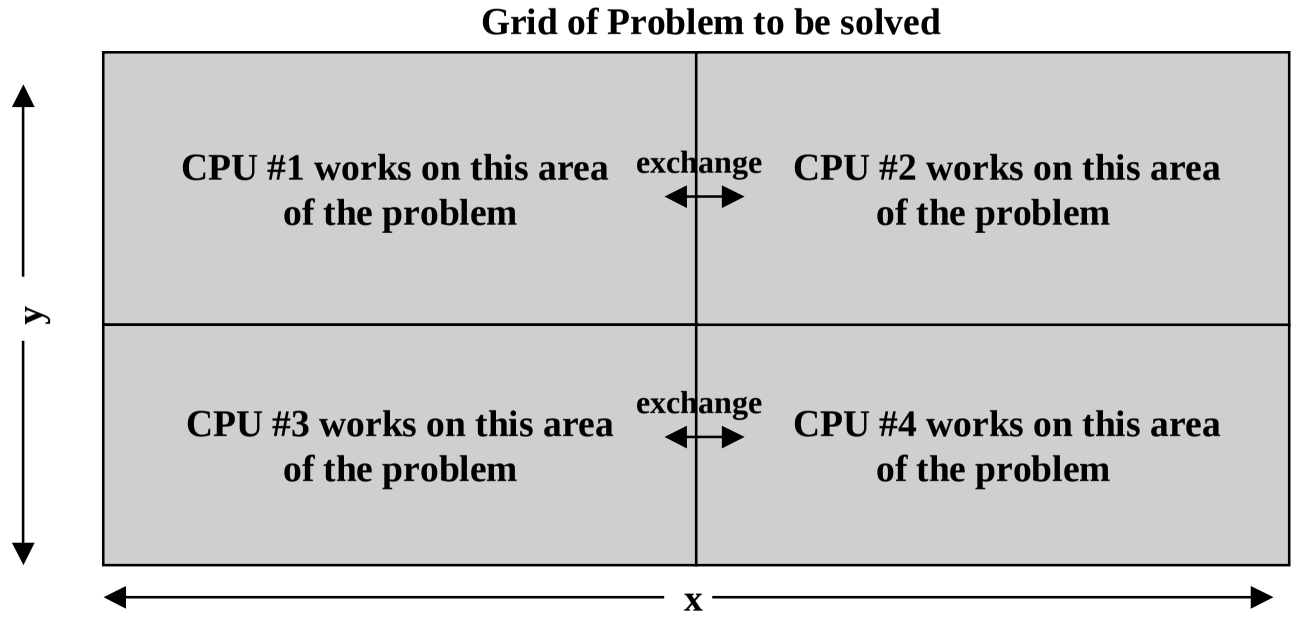
- Vehicle design and dynamics
- Analysis of protein structures
- Human genome work
- Quantum chromodynamics, Astrophysics
- Ocean modeling
- Imaging and Rendering
- Petroleum exploration
- Nuclear Weapon design
- Database query
- Ozone layer monitoring
- Natural language understanding
- Study of chemical phenomena
- And many other grand challenge projects

Cost Effectiveness of Parallel Processors

- Currently the speed of off-the shelf micro processors is within one order of magnitude of the fastest serial computers. Micro processors cost many orders of magnitude less.
- Connect only a few micro processors together to form a parallel computer with speed comparable to fastest serial computers. Cost of such parallel computer is less.
- Connect a large number of processors into parallel computer overcomes the saturation rate of serial computers.
- Parallel computers can provide much higher raw computation power than fastest serial computes.
- Need to have actual applications that can take advantage of this high power.

What is Parallel Computing?

- **Parallel computing: use of multiple computers or processors working together on a common task.**
  - Each processor works on its section of the problem
  - Processors can exchange information



Why Do Parallel Computing?

- **Limits of single CPU computing**
  - Available memory
  - Performance
- **Parallel computing allows:**
  - Solve problems that don't fit on a single CPU's memory space
  - Solve problems that can't be solved in a reasonable time
- **We can run…**
  - Larger problems
  - Faster
  - More cases

Types of Parallelism : Two Extremes

- **Data parallel**
  - Each processor performs the same task on different data
  - Example - grid problems
- **Task parallel**
  - Each processor performs a different task
  - Example - signal processing
- **Most applications fall somewhere on the continuum between these two extremes**

Typical Data Parallel Program

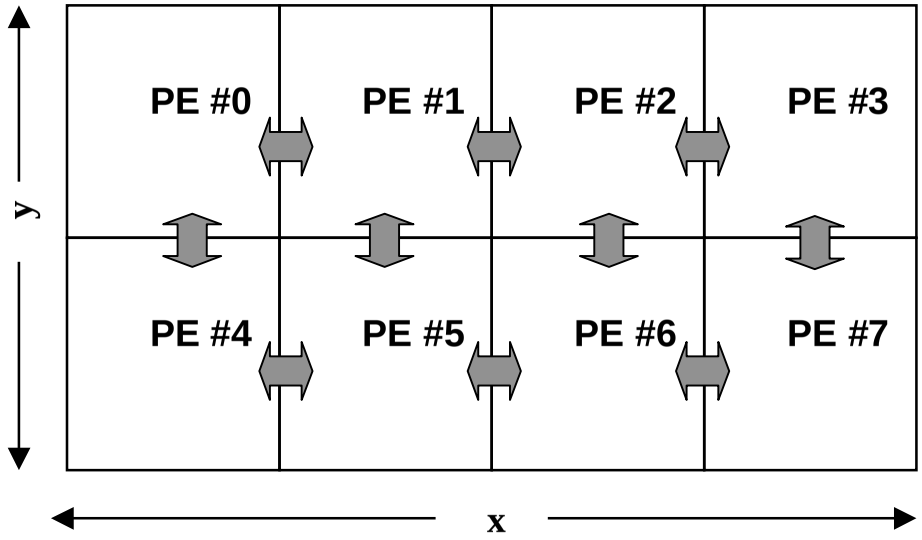
- **Example: integrate 2-D propagation problem:**

Starting partial differential equation:

$$\frac{\partial \Psi}{\partial t} = D \cdot \frac{\partial^2 \Psi}{\partial x^2} + B \cdot \frac{\partial^2 \Psi}{\partial y^2}$$

Finite Difference Approximation:

$$\frac{f_i^{n+1} - f_i^n}{\Delta t} = D \cdot \frac{f_{i+1}^n - 2 f_i^n - f_{i-1}^n}{\Delta x^2} + B \cdot \frac{f_{i+1}^n - 2 f_i^n - f_{i-1}^n}{\Delta y^2}$$



Basics of Data Parallel Programming

- **One code will run on 2 CPUs**
- **Program has array of data to be operated on by 2 CPU so array is split into two parts.**

```
program:
…
if CPU=a then
   low_limit=1
   upper_limit=50
elseif CPU=b then
   low_limit=51
   upper_limit=100
end if
do I = low_limit,
upper_limit
   work on A(I)
end do
...
end program
```

CPU A

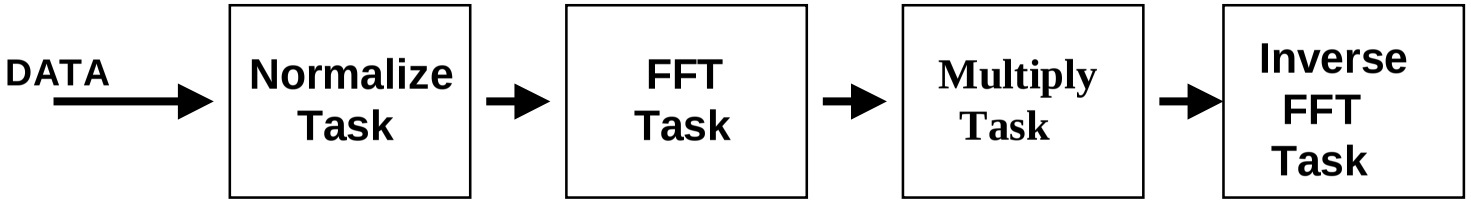
```
program:
…
low_limit=1
upper_limit=50
do I= low_limit,
upper_limit
   work on A(I)
end do
…
end program
```

CPU B

```
program:
…
low_limit=51
upper_limit=100
do I= low_limit,
upper_limit
   work on A(I)
end do
…
end program
```

Typical Task Parallel Application

- **Example: Signal Processing**
- **Use one processor for each task**
- **Can use more processors if one is overloaded**



Basics of Task Parallel Programming

- **One code will run on 2 CPUs**
- **Program has 2 tasks (a and b) to be done by 2 CPUs**

```
program.f:
…
initialize
...
if CPU=a then
   do task a
elseif CPU=b then
   do task b
end if
….
end program
```

CPU A

```
program.f:
…
initialize
…
do task a
…
end program
```

CPU B

```
program.f:
…
initialize
…
do task b
…
end program
```

Limits of Parallel Computing

- **Theoretical Upper Limits**
  - Amdahl's Law
- **Practical Limits**
  - Load balancing
  - Non-computational sections
- **Other Considerations**
  - time to re-write code

Theoretical Upper Limits to Performance

- **All parallel programs contain:**
  - Serial sections
  - Parallel sections
- **Serial sections limit the parallel effectiveness**
- **Speedup is the ratio of the time required to run a code on one processor to the time required to run the same code on multiple (N) processors**
- **Amdahl's Law states this formally**

Amdahl's Law

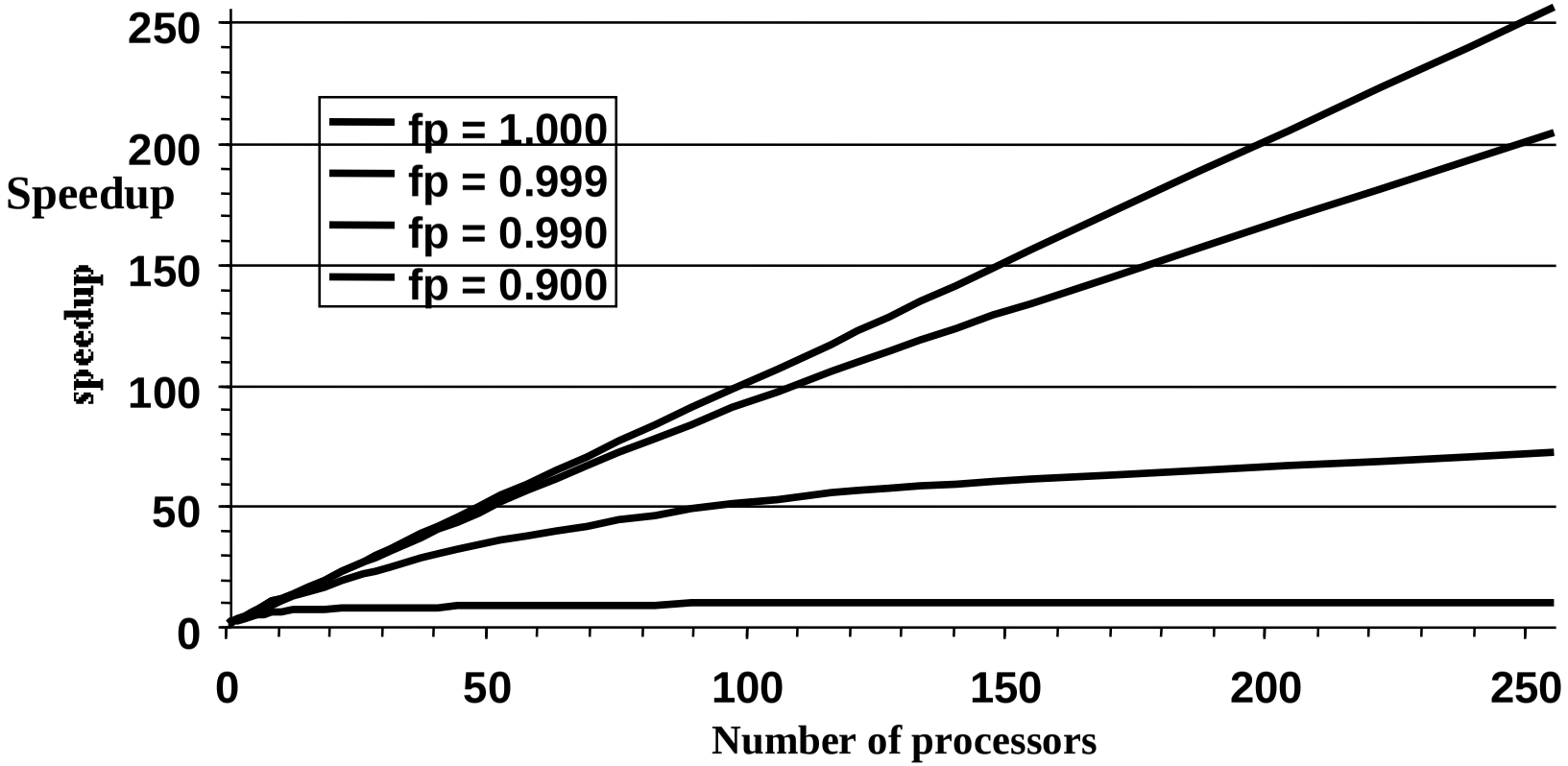
- **Amdahl's Law places a strict limit on the speedup that can be realized by using multiple processors.**
  - Effect of multiple processors on run time

$$t_n = \left(f_p / N + f_s \right) \; t_1$$

  - Effect of multiple processors on speed up
  - Where $S = \dfrac{1}{f_s + f_p / N}$
    - fs = serial fraction of code
    - fp = parallel fraction of code
    - N = number of processors
    - t_n = time to run on N processors

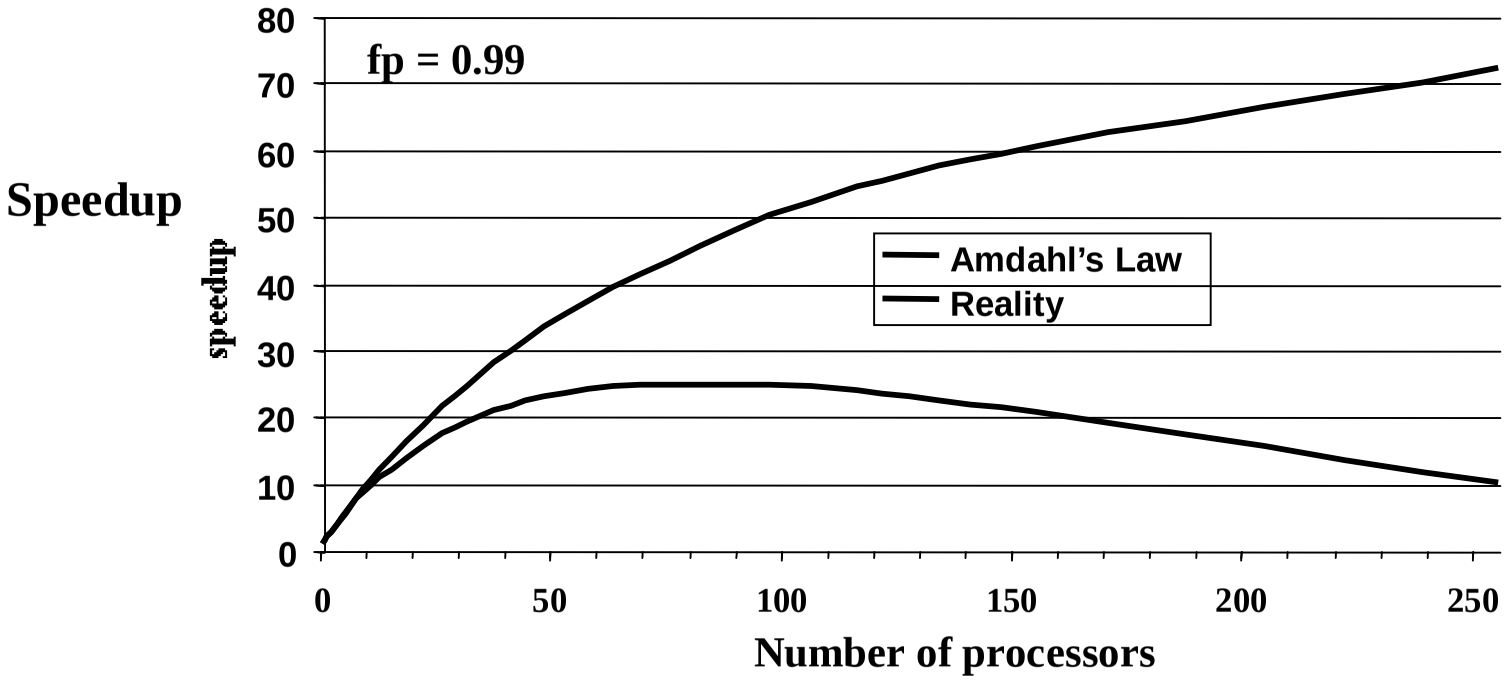
Illustration of Amdahl's Law

It takes only a small fraction of serial content in a code to degrade the parallel performance.



Practical Limits: Amdahl's Law vs. Reality

Amdahl's Law provides a theoretical upper limit on parallel speedup assuming that there are no costs for speedup assuming that there are no costs for *communications*. In reality, communications will result in a further degradation of performance.



Practical Limits: Amdahl's Law vs. Reality

- **In reality, Amdahl's Law is limited by many things:**
  - Communications
  - I/O
  - Load balancing (waiting)
  - Scheduling (shared processors or memory)

Other Considerations

- **Writing effective parallel application is difficult!**
  - Load balance is important
  - Communication can limit parallel efficiency
  - Serial time can dominate

- **Is it worth your time to rewrite your application?**
  - Do the CPU requirements justify parallelization?
  - Will the code be used just once?

Issues in Parallel Computing

- **Design of parallel computers :** Design so that it scales to a large # of processor and are capable of supporting fast comunication and data sharing among processors.
- **Design of Efficient Algorithms :** Designing parallel algorithms are different from designing serial algorithms. Significant amount of work is being done for numerical and non-numerical parallel algorithms
- **Methods of Evaluating Parallel Algorithms :** Given a parallel computer and a parallel algorithm we need to evaluate the performance of the resulting system. How fast problem is solved and how efficiently the processors are used.

Issues in Parallel Computing

- **Parallel Computer Languages :** Parallel algorithms are implemented using a programming language. This language must be flexible enough to allow efficient implementation and must be easy to program in. Must efficiently use the hardware.
- **Parallel Proramming Tools :** Tools (compilers, libraries, debuggers, other monitoring or performance evaluation tools) must shield users from low level machine characteristics.
- **Portable Parallel Programs:** This is one of the main problems with current parallel computers. Program written for one parallel computer require extensive work to port to another parallel computer.

Issues in Parallel Computing

- **Automatic Programming of Parallel Computers :** This is about design of parallelizing compilers which extract implicit parallelism from programs that have not been explicitly parallelized. But this approach has limited potential for exploiting the power of large parallel machines.