Parallel Programming & Cluster Computing

Monte Carlo

Henry Neeman, University of Oklahoma
Paul Gray, University of Northern Iowa

SC08 Education Program’s Workshop on Parallel & Cluster computing
Oklahoma Supercomputing Symposium, Monday October 6 2008
Embarrassingly Parallel

An application is known as *embarrassingly parallel* if its parallel implementation:

1. can straightforwardly be broken up into roughly equal amounts of work per processor, **AND**
2. has minimal parallel overhead (e.g., communication among processors).

We *love* embarrassingly parallel applications, because they get **near-perfect parallel speedup**, sometimes with modest programming effort.

Embarrassingly parallel applications are also known as *loosely coupled*. 
Monte Carlo Methods

Monte Carlo is a European city where people gamble; that is, they play games of chance, which involve randomness. 

*Monte Carlo methods* are ways of simulating (or otherwise calculating) physical phenomena based on randomness. Monte Carlo simulations typically are embarrassingly parallel.
Monte Carlo Methods: Example

Suppose you have some physical phenomenon. For example, consider High Energy Physics, in which we bang tiny particles together at incredibly high speeds.

We want to know, say, the average properties of this phenomenon.
There are infinitely many ways that two particles can be banged together.
So, we can’t possibly simulate all of them.
Monte Carlo Methods: Example

Suppose you have some physical phenomenon. For example, consider High Energy Physics, in which we bang tiny particles together at incredibly high speeds.

There are infinitely many ways that two particles can be banged together. So, we can’t possibly simulate all of them. **Instead, we can randomly choose a finite subset** of these infinitely many ways and simulate only the subset.
Monte Carlo Methods: Example

Suppose you have some physical phenomenon. For example, consider High Energy Physics, in which we bang tiny particles together at incredibly high speeds.

There are infinitely many ways that two particles can be banged together.

We randomly choose a finite subset of these infinitely many ways and simulate only the subset.

The average of this subset will be close to the actual average.
Monte Carlo Methods

In a Monte Carlo method, you randomly generate a large number of example cases (realizations) of a phenomenon, and then take the average of the properties of these realizations.

When the realizations’ average converges (i.e., doesn’t change substantially if new realizations are generated), then the Monte Carlo simulation stops.
Monte Carlo simulations are embarrassingly parallel, because each realization is completely independent of all of the other realizations. That is, if you’re going to run a million realizations, then:

1. you can straightforwardly break up into roughly $1M / N_p$ chunks of realizations, one chunk for each of the $N_p$ processors, **AND**

2. the only parallel overhead (e.g., communication) comes from tracking the average properties, which doesn’t have to happen very often.
Serial Monte Carlo

Suppose you have an existing serial Monte Carlo simulation:

```fortran
PROGRAM monte_carlo
    CALL read_input(…)
    DO realization = 1, number_of_realizations
        CALL generate_random_realization(…)
        CALL calculate_properties(…)
    END DO
    CALL calculate_average(…)
END PROGRAM monte_carlo
```

How would you parallelize this?
PROGRAM monte_carlo

[MPI startup]
IF (my_rank == server_rank) THEN
    CALL read_input(...)
END IF
CALL MPI_Bcast(...)
DO realization = 1, number_of_realizations
    CALL generate_random_realization(...)  
    CALL calculate_realization_properties(...)  
    CALL calculate_local_running_average(...)  
END DO
IF (my_rank == server_rank) THEN
    [collect properties]
ELSE
    [send properties]
END IF
CALL calculate_global_average_from_local_averages(...)  
CALL output_overall_average(...)  
[MPI shutdown]
END PROGRAM monte_carlo
To Learn More

http://www.oscer.ou.edu/
Thanks for your attention!

Questions?