





Parallel Software Design

Deborah Stacey, *Chair*
 Dept. of Comp. & Info Sci., University of Guelph
 dastacey@uoguelph.ca

The Parallel Software Design Process



Why Parallel?
Why NOT Parallel?
Why Talk about Design?

Why Parallel?

- It is often not obvious that making an algorithm parallel will have benefits.
- The designer must first answer the question: why parallel?
- There are really only two reasons for going parallel:
 - Faster - doing the same amount of work in a smaller amount of time.
 - Bigger - doing more in the same amount of time.
- Both of these reasons can be argued to produce **better** results.

HPC Resources

Why NOT Parallel?

- Bad parallel programs can be worse than their serial cousins.
 - Slower - because of communications overhead
 - Scale Problems - some parallel algorithms are only successful if the problem is very, very large; smaller problems will not show any significant benefits.
- Not all problems are amenable to parallelism.

Understand the Problem.

HPC Resources

Why Talk About Design?

- There are many choices when it comes to creating a parallel and/or distributed program.
 1. Computation
 2. Data
 3. Communications
 4. Algorithm
 5. Expected Result

HPC Resources



Software Design Principles

*What is Smart Design?
Why is Simple Good?
The Parallel Design Process*

What is Smart Design?

- *Smart* design is
 - Simple
 - Modular
 - Iterative
 - Modifiable
 - Maintainable
 - Successful

HPC Resources

Why is Simple Good?

- Complexity only begets more complexity.
- Simple code can be read and understood.
- Simple code can be changed easily and quickly.
- Simple successful approaches to problems have to be *elegant*.

Elegance = Understanding.

HPC Resources

The Parallel Design Process

1. Determine your goal in making the code parallel.
2. Select the appropriate approach to parallelism.
3. Select the metrics to be used to determine success.
4. Develop the code.
5. Test your system using the pre-determined metrics.
6. Re-evaluate the system using the metrics and determine the next step:
 - Re-design with another approach.
 - Fine-tune the current design.
 - Consider a hybrid approach.

HPC Resources



Different Approaches

Perfectly Parallel
Data/Task Parallelism
Specialty Nodes
Massively Parallel

Perfectly Parallel

- Run copies of the serial algorithm on a number of processors.
- Result is the aggregate of the individual serial answers.
- Appropriate for parameter-driven algorithms.
- No change to the serial algorithm.
- Little communications overhead or concerns about synchronization.
- May have concerns about the location of needed data - this may translate into a communications problem, i.e, bandwidth.

HPC Resources

Data Parallelism

- The data is broken up into appropriate slices and distributed to different processors.
- Serial algorithm must be changed to accommodate this distribution of the dataset.
- May result in problems with synchronization of communications and different sections of code.

HPC Resources

Data Parallelism

- Need to have a theory about how to split and recombine the data.
- Communications overhead is a concern - may only be appropriate when the dataset is HUGE.

Size matters: Determine what big means.

Task Parallelism

- The algorithm itself is divided into independent tasks that can be distributed onto individual processors.
- The parallel algorithm may not greatly resemble the serial one.
- A communications strategy is needed.
- Communications overhead can be significant.

A poor understanding of communications can defeat parallelism.

Specialty Nodes

- Isolate some tasks that all computations use to only a few (or one) processor.
- Some ubiquitous calculations (such as random number generation) are more efficient if done in isolation.
- This approach also allows more flexibility when it comes to develop, refactoring, and testing.

Massively Parallel

- Mimics the structure of the problem very closely.
- Tasks on individual processors are small and simple.
- Communication between nodes is very high.
- Usually done on specialty architectures (e.g. Connection Machine).
- Easy to conceptualize but difficult to implement efficiently.



A Case Study

The Parallel Genetic Algorithm



Genetic Algorithms

An Introduction

Artificial Evolution

- John Holland proposed that any problem in adaptation (natural or artificial) can generally be formulated in genetic terms [*Adaptation in Natural and Artificial Systems*].
- A *genetic algorithm* (GA) simulates Darwinian evolutionary processes and genetic operations on chromosomes.

HPC Resources

Genetic Algorithms

A **genetic algorithm** is a *computational* technique that transforms a set (population) of individual mathematical objects (usually fixed length character or binary strings), each with an associated fitness value, into a new population (next generation) using operations patterned after the Darwinian principle of reproduction and survival of the fittest and after naturally occurring genetic operations (sexual recombination, mutation).

Genetic Programming by John R. Koza (pg 18)

HPC Resources

The Representation Scheme

- The most common representation for individuals in a population is a string of binary digits.
- Our example problem will be represented as a string of length $L=3$ over an alphabet of size $K=2$ (0 and 1).
- The search space for this problem consists of $2^3=8$ possible combinations.

Binary Representation

011
001
110
010

HPC Resources

Generation 0

- The initial generation consists of randomly created individuals.
- We must also set an initial population size, M
- Let us set M to 4 and use the strings **011**, **001**, **110**, **010**.

HPC Resources

Fitness

- For each generation, each individual in the population is tested against the unknown *environment* in order to ascertain its *fitness* in this environment.
- In this example, we will equate fitness and the value of the binary string as a binary number.

HPC Resources

The Initial Population: Generation 0

- | | | | | |
|----------------|---|---|---|---------------------------------|
| • Individual 1 | 0 | 1 | 1 | $f = 3$ |
| • Individual 2 | 0 | 0 | 1 | $f = 1 \leftarrow$ <u>worst</u> |
| • Individual 3 | 1 | 1 | 0 | $f = 6 \leftarrow$ <u>best</u> |
| • Individual 4 | 0 | 1 | 0 | $f = 2$ |

HPC Resources

Fitness Proportionate Reproduction

- Fitness proportionate reproduction copies the individuals in the current population into the next generation with a probability proportional to their fitness.

String	Fitness	Probability	Mating Pool	New Fitness
X_i	$f(X_i)$			
011	3	0.25	011	3
001	1	0.08	110	6
110	6	0.50	110	6
010	2	0.17	010	2

HPC Resources

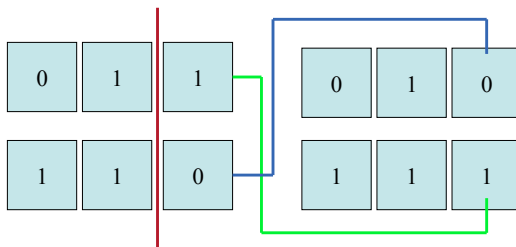
Crossover

- Recombination, or crossover, allows new individuals to be created, thus testing new points in the search space.
- The two individuals participating in crossover are selected proportionate to fitness.
- Crossover produces two offspring which are usually different from their parents and from each other.
- Crossover is applied to a specified percentage of the mating pool, the *crossover probability*, using fitness as a selection criterion.

HPC Resources

Crossover

- Parents
- Children



HPC Resources

Generation 0 vs. Generation 1

Generation 0			Mating Pool after Reprod			After Xover (Gen 1)	
String	Fit	Prob	Mating Pool	X		X_i	$f(X_i)$
X_i	$f(X_i)$		Pool	$f(X_i)$	Pt		
011	3	0.25	001	3	2	111	7
001	1	0.08	110	6	2	010	2
110	6	0.50	110	6	-	110	6
010	2	0.17	010	2	-	110	2

HPC Resources

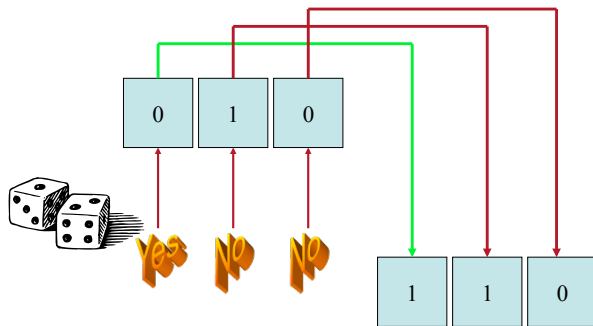
Mutation

- Another genetic operation is *mutation* which is an asexual operation that only operates on one individual.
- An individual is randomly selected from the mating pool and a *mutation point* is randomly selected as a number between 1 and L .

Mutation

- The single character at the mutation point is changed:
 - in a binary alphabet, the character is complemented
- The frequency of application of the mutation operation is controlled by a parameter called the *mutation probability*.

Mutation



Termination

- The GA continues to produce new generations of individuals until some termination criterion is satisfied.
- A particular GA run may go on for thousands of generations.
- This termination criterion can be:
 - when a maximum number of generations has been produced,
 - a perfect solution is recognized, or
 - when the population has certain characteristics.

Using GAs for Problem Solving

- **Step 1.** Determine the representation scheme.
 - the mapping of each possible point in the search space to a fixed-length character string.
- **Step 2.** Determine the fitness measure.
 - must be able to evaluate all strings.
- **Step 3.** Determine the parameters for controlling the algorithm.
 - population size, maximum number of generations,
 - probabilities of reproduction, crossover and mutation.
- **Step 4.** Determine a method for designating the answer and the criterion for termination.

HPC Resources



The Perfectly Parallel GA

Characteristics
PPGA.1
PPGA.2
Advantages and Disadvantages

Characteristics


- The GA is a **parameter-driven** technique.
 - Selection Probability
 - Crossover probability
 - Mutation probability
 - Population size and individual representation scheme
 - Operators - selection, crossover, mutation
 - Extra operators - e.g. elitism
- The GA is a **stochastic** technique (randomness involved).
 - Probabilities
 - Generation of the original population.

HPC Resources

Perfectly Parallel GA Versions

- **The Perfectly Parallel GA (PPGA.1)**
 - Run multiple parameter settings and compare results.
 - Goal: Faster
 - How: Multiple variations done in the same time
- **The Perfectly Parallel GA (PPGA.2)**
 - Run multiple instances using the same parameter settings and generate aggregate results with statistical analysis.
 - Goal: Faster
 - How: To guarantee statistically significant results, the GA would have to be run multiple times and statistics gathered.


HPC Resources


 SHARCNET™

Advantages and Disadvantages

- **Advantages**
 - ✓ **Faster**
 - No communications overhead or synchronization issues.
 - Same data used by all processes - no data splitting/joining.
 - Easy to understand and measure success.
 - The multiple running of the same GA has to be done anyways so why not in parallel?
- **Disadvantages**
 - No speed up of an individual instance of the GA.
 - ⊗ **Bigger**: does not affect the size of problem that can be attacked.
 - Boring


HPC Resources

 SHARCNET™



The Task Parallel GA


*Characteristics
Design and Design Issues
Advantages and Disadvantages*

 SHARCNET™

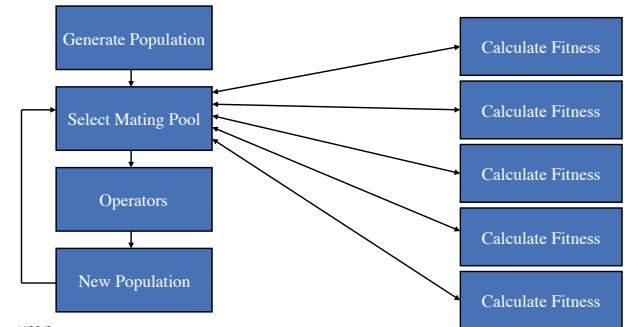
Characteristics

- Algorithm performance analysis - what part of the GA takes the most time?
- For some GA's the most expensive part of the algorithm is the calculation of the fitness function.
- If each individual fitness calculation is expensive then can we speed up the algorithm by doing multiple fitness calculations at the same time?

HPC Resources

 SHARCNET™

Design



```

graph TD
    A[Generate Population] --> B[Select Mating Pool]
    B --> C[Operators]
    C --> D[New Population]
    D --> B
    B --> E[Calculate Fitness]
    B --> F[Calculate Fitness]
    B --> G[Calculate Fitness]
    B --> H[Calculate Fitness]
    B --> I[Calculate Fitness]
  
```

HPC Resources

Design Issues

- How many processors are devoted to fitness calculation?
- How do you distribute all individuals in a population to those processors?
- Do you increase the size of the population?
- Are there other parts of the algorithm that can be parallelized?

Resource allocation is difficult.
How many processors can you successfully use?

HPC Resources

Advantages and Disadvantages

- **Advantages**
 - ✓ Faster
 - ✓ Bigger
 - Still a simple adaptation of the original algorithm.
 - Very little communications overhead and simple synchronization.
- **Disadvantages**
 - Cost-benefit analysis: how many processors are too many?
 - Only applicable when fitness calculation is computational intensive.

HPC Resources



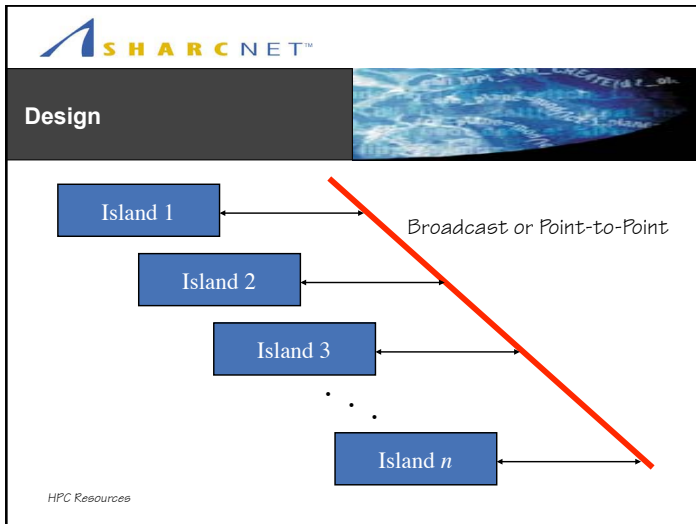
The Data Parallel GA

Characteristics
Design
Advantages and Disadvantages

Characteristics

- There are many variations of the standard GA
- One of the variations is called the Island GA
- In this algorithm, the standard GA is started on a number of separate populations or *islands*
- Using various criteria, individuals are exchanged between these *islands*

HPC Resources



SHARCNET™

Advantages and Disadvantages

- **Advantages**
 - ✓ **Bigger** - total population is large
 - ⊕ **Faster** - faster than serial Island GA but...
- **Disadvantages**
 - How can this technique be evaluated?
 - Possibly complex communications strategy
 - How many islands?

Evaluation is essential and hard.

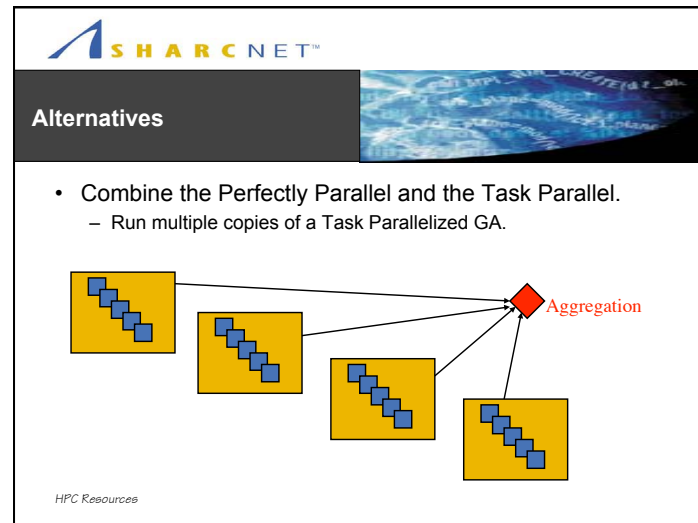
HPC Resources

SHARCNET™

The Hybrid Parallel GA

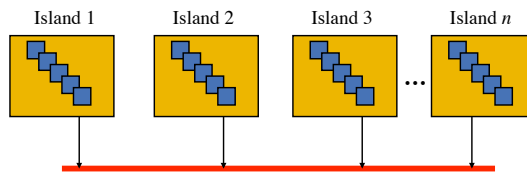
Alternatives
Advantages and Disadvantages

HPC Resources



Alternatives

- Combine the Data Parallel and the Task Parallel.
 - Each Island uses Task Parallelism.



HPC Resources

Alternatives

- Combine any of the other versions or even just a serial GA with a specialty node that just serves out random numbers.
 - Need lots of random numbers for selection, operators, etc.
 - Lots of different random number generators.
 - Specialty node can be generating random numbers continually and serving them out as needed.
 - Cost for each random number is the cost of communication to the specialty node.
- Why?
 - Need lots of random numbers for selection, operators, etc.
 - Lots of different random number generators.
 - Specialty node can be generating random numbers continually and serving them out as needed.
 - Cost for each random number is the cost of communication to the specialty node.

HPC Resources

Advantages and Disadvantages

- Advantages
 - ✓ Faster
 - ✓ Bigger
 - ✓ Modularity
- Disadvantages
 - ⊗ Complexity
 - ⊗ Decisions, decisions, decisions
 - ⊗ Cost-Benefit analysis
 - ⊗ Evaluation
 - Where is the speed-up coming from?
 - Where are the bottle-necks?

HPC Resources



Which Approach is Best?

*Why Parallel? - Revisited
Evaluating the Design
Restructuring or Refactoring
Testing is Central
Testing Strategy Components*

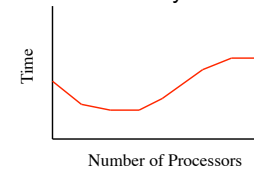
Why Parallel? - Revisited

- Approach should match the goal of making the algorithm parallel
- Faster
 - How much faster does the algorithm have to be to be considered a success?
 - How many resources can be devoted to the parallel version?
- Bigger
 - Why does the problem have to be bigger?
 - Is bigger better?

HPC Resources

Evaluating the Design

- Measure the speed up with a modest set of resources.
- Identify any bottlenecks particularly with communications.
- Identify what aspect of the parallelism is causing the speedup.
- Perform a cost-benefit analysis.



HPC Resources

Restructuring or Refactoring

- **Refactoring** means improving the design of existing code.
 - It is the process of changing a software system in such a way that it does not alter the external behaviour of the code yet improves its internal structure.
- **Restructuring** is the transformation from one design structure to another, while preserving the subject system's external behaviour (functionality and semantics).

HPC Resources

Restructuring or Refactoring

- Often restructuring involves transformations designed to remove complexity or to make complexity easier to handle and maintain.
- Restructuring creates does not normally involve modifications because of new requirements, but it may lead to observations that suggest changes that would improve aspects of the system.

Defeat complexity - embrace simplicity.

HPC Resources

Restructuring or Refactoring

- When should refactoring be used:
 - Adding a new feature - refactoring is good preparation for modifying code.
 - Fixing a bug - understanding the algorithm well enough to fix a known bug means you understand it well enough to simplify it.
 - Performing code review - when you examine the code base implemented by the development team, one of the goals should be to simplify code structure and algorithms in context of the entire application.

[Refactoring: Shifting development back into first gear Issued by: Citigate Ballard King <http://www.itweb.co.za/office/compuwaresa/0301230732.htm>]

HPC Resources

Testing is Central

- Objectives of the testing of parallel programs:
 - Detection of communication problems
 - Detection of performance bottlenecks
 - Debugging
 - Refactoring/restructuring opportunities
- Testing Strategy Components
 - Instrumentation
 - Experiments
 - Analysis

HPC Resources

Testing Strategy Components

- Instrumentation
 - Tools and libraries used to acquire information about a program's execution.
 - This can be dependent on platform specific monitoring resources or can be as simple as embedded generic profiling statements.
- Experiments
 - Random execution: run the program many times to find problems
 - Deterministic execution: the order of logged communication events is verified against the specification
- Analysis
 - Tools that analyze trace files, produce execution profiles (i.e., characterization, classification, and location), and generate graphics for visualization.

HPC Resources



Faster, Bigger, but Better?

Revisit the requirements often.

Fast, Bigger, but Better?

- First, define what is meant by better for the particular algorithm.
- Go back to the requirements for the program:
 - How fast does it have to be to meet the requirements?
 - How much work has to be done to produce a result?
- Do the requirements have to be changed?
 - Increased expectations.
 - Possible efficiencies and simplifications.
- But is the program maintainable?

HPC Resources



Another Case Study

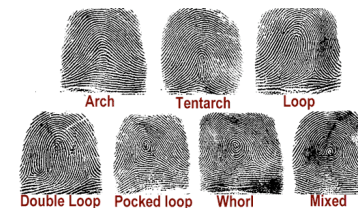
The Problem
The Requirements
Possible Parallel Approaches

The Problem

- Fingerprints are found on all people and some animals.
- They are unique to the individual and usually remain unchanged over a lifetime.
- Worldwide databases of billions of unique fingerprints exist.
- Examiners identify prints by making comparisons of one friction ridge print with another.
- Modern computerized fingerprint analysis allows the process to be automated and speeds up searching for a match.

HPC Resources

Fingerprints



HPC Resources

The Requirements

- Must be able to handle millions of fingerprint records.
- The matching requirements must be a parameter.
 - When a line on a fingerprint stops or splits it is called a *typica*.
 - The amount of typica necessary for a match differs between countries.
 - In the Netherlands, 10 to 12 characteristic points with no differences are required and in South Africa it is 7 points for an identification.
 - In England and the U.S. there is no rule; an expert decides what is sufficient.
- The amount of time for searching is also a parameter.

HPC Resources

Possible Parallel Approaches

- Data Parallelism?
 - How can the data be distributed?
 - What is the possible speed up?
- Task Parallelism?
 - Are there opportunities to parallelize searching/matching?
 - How can this approach be analysed and evaluated?
- Resource Allocation
 - How big is realistic?
 - Can the benefits of parallelism be measured?

HPC Resources