Parallel Algorithms in STAPL
Implementation and Evaluation

Jeremy Vu
Faculty Mentor: Dr. Nancy Amato
Supervisor: Dr. Mauro Bianco
Parasol Lab, Texas A&M University

http://parasol.tamu.edu
Presentation Outline

• Parallelism
  • What is parallelism?
  • What are some motivations/problems?
  • What are some solutions?

• STAPL: What is it?

• My work for the summer.
  • Example: p_unique
  • Performance testing
What is parallelism?

- In terms of computing, parallelism is simultaneous occurrence/coincedence.

- We perform multiple computations simultaneously. The general idea is that we can divide a problem into smaller ones that we can solve at the same time.
  - We can solve problems faster this way (hopefully).

- Example:
  - If the Girl Scouts only sent one scout to sell cookies, they almost certainly wouldn’t make money fast enough to finance all of their activities.
  - Reality: They have many scouts out selling cookies at the same time. They earn money much faster this way.
Okay?

- Parallel computing has been practiced for a long time in High-Performance Computing (HPC).
  - Ex: weather forecasting models
  - Ex: simulation of nuclear weapon detonations
  - Ex: hydrodynamics (study of liquids in motion)

- It's becoming more popular.
  - Large scale parallel machines are getting larger.
  - Small scale parallel machines are mainstream.
    - re: many-core processors

- As more people get into parallel computing, the issues associated with it become more prevalent, and new ones arise.
Challenges and Solutions

• **Productivity:**
  - Scalability and efficiency
  - Portable efficiency
  - Programmability

• **Hardware:**
  - Memory setup (shared, distributed, distributed-shared)
  - Communication implementation (interprocessor, processor-memory)

• **Generalized solutions include:**
  - programming languages (Chapel by Cray)
  - libraries (TBB by Intel, various in Microsoft .NET)
  - APIs (OpenMP, MPI)

• STAPL, the project I’m a part of, is one such solution.
**STAPL:** A library of parallel, generic constructs based on the C++ Standard Template Library (STL)

- Provides parallel data structures and generic parallel algorithms
- Helps programmers address the difficulties of parallel programming by:
  - Allowing programmers to work at a high level of abstraction
  - Hiding many of the details of parallel programming
  - Providing a high degree of productivity, performance, and portability.
Example: p_unique

STL unique → p_unique

Input: A sequence of elements and a binary relation.
Output: A sequence of elements consisting of only the first element of each group of consecutive duplicate elements.
    - The binary relation is used to determine the duplicate elements.
    - ex: ‘=’
      - {1, 1, 2, 2, 3, 3, 4, 4} --> {1, 2, 3, 4}
    - Sequentially, there’s only one way to do this; in parallel, there are multiple cases.

- Case 1, symmetric + transitive:
  - ex: ‘=’
  - Compare, in parallel, each element to the next element in the sequence using the relation and keep or remove it based on the result.
  - If we have enough processors, all of the comparisons can happen simultaneously. This is ideal.
Example: p_unique

- Case 2, transitive: Requires parallel prefix algorithm. Parallel prefix allows us to “prefix” each chunk of data with the appropriate initial element. Then we complete in the same manner as for case 1.
  - ex: ‘<‘ operator
  - \{100, 80, 70, 20, 40, 30, 15, 10\} ---> \{100, 80, 70, 20, 15, 10\}

- Case 3, neither: Sequential, unfortunately; only comparison must be done in order, one at a time.
  - ex: ‘is in love with’, ‘is the father/mother of’
Example: p_unique

```c
void stapl_main(int, char **) {
    ...

    p_unique(p_vector);

    ...

    p_unique_copy(p_vector, p_array, '<');

    ...

    p_unique_copy(p_list, p_vector, my_relation);
}
```
What I Do in STAPL (cont.)

Some of the things I test

• Parallel speedup (scalability) – How does the addition of more processors affect the performance?
  • Ideally, linear speedup.

• Performance vs. theoretical model and vs. sequential counterpart(s).

• Performance in different situations.
  • Ex: data is distributed poorly in a distributed memory system (all data on a single location)

• Different algorithms for the same problem, such as how I choose to schedule activities.
More information…

Questions? Comments? Concerns?

http://parasol.tamu.edu/stapl

{cvu, amato, bmm}@cse.tamu.edu