Advanced parallel programming

Dottorato di Ricerca
Dip. Informatica di Pisa

M. Danelutto
Giugno-Luglio 2007

Program

- Introduction
- Classical programming models
- Structured programming models
- Skeleton parallel programming environments
- Open problems
- Project

Open problems

- Programmability: skeleton set expandability
  - preserving efficiency, user controlled access to the implementation
- Autonomic management
  - self configuring, optimizing, protecting, healing
- Predictability
  - performance models
- Security
  - tradeoff security/efficiency

Methodology

- Rapid prototyping
  - exploiting existing skeletons
- Fine tuning
  - possibly exploiting “parametric skeleton
- Exploiting performance models
  - predicting rather than backtracking
- Functional debugging
  - vs. support code debugging

❶Minimal disruption
❷Ad hoc parallelism
❸Accommodate diversity
❹Show payback
❺Reuse code
❻Handle heterogeneity
❼Handle dynamicity

Final project

- Basic point
  - “hands on” structured parallel programming
    - using structured parallel programming tools (muskel, assist, eSkel, Muesli, ocamlP3l)
    - using other tools in a structured way
      - defining new template/skeletons/libraries
- goal:
  - demonstrate advantages / feasibility
  - compare implementations
  - verify performance / efficiency

Final project (howto)

- in group (max 2 people)
- public discussion of the results
  - half an hour
  - with “demo”
- to be done by the autumn
  - topic chosen by mid July

Final project proposals (1)

- 2 groups/persons
  - sample application implemented on two different programming environments
    - sequence matching ?
      image processing (smoothing, sharpening, ...) ?
      diff eq solver ?
- comparing
  - developing time (coding, tuning, debugging, performance tuning)
  - efficiency / performance / results

Final project proposals (2)

- 1 group/person
- reverse engineering a structured parallel programming environment implementation
  - Muesli ?
    eSkel ?
- produces:
  - user manual (!)
  - implementation architectural design
  - detailed skeleton implementation design

Final project proposals (3)

- 1 group/person
- program transformation (semi-formal) -> implementation
- sample application (use case)
- application modeling
    - bottleneck analysis + design improvement
- once refined
    - implementation
        - using structured programming models, if possbile
        - or even traditional programming tools

Final project proposals (4)

- 1 group/person
- adapting structured parallel programming methodology to unconventional parallel “architectures”
  - GPUs, multicore, ...
  - *in vivo* or *in vitro* (if any available, using emulators)
- e.g. implement and demonstrate task farm skeleton on multithreaded [G|C]PUs

Final project proposals (5)

- 1 group/person
- characterization of a new (w.r.t. the ones already studied here) parallelism exploitation pattern
  - e.g. divide and conquer
  - parameters
  - grain (seq algorithm + base case)
  - (rough) performance model
  - implementation using MPI / Java+RMI

Final project proposals (6)

- 2 groups/persons
- comparison: concurrent/parallel progr language/library vs. skeletons
  - X10, JCSP, CCA, GlobusToolkit, ...
  - sample application (structured!)
- implementation of the sample application in the two environments
  - comparison of developing times (as in (1))
  - and of the results

Final project proposals (7)

- 1 group/person
- AOP and metaprogramming for structured parallel programming
    - AspectJ, annotations
    - to intercept (user) patterns
        - to drive “structured” implementation of the patterns
- e.g. implement task farm of annotated Java/C# methods

Final project proposals (8)

- 1 group/person
- structured analysis of an application
  - application / algorithm from your own research area
  - parallelism exploitation
  - analysis
  - implementation with one of the environments available
    - or with traditional tools used “the structured way”

Don’t forget

- Thursday afternoon: lesson given by Peter Kilpatrick
- Friday afternoon: ASSIST (by Massimo Coppola)