Languages, Compilers and Run-Time Libraries for Computational Grids

Ken Kennedy
Center for Research on Parallel Computation
Rice University

1. Problem Definition and Goals
2. Available Technologies
3. Three Paradigms for Application Development
   - Task Composition
   - Global Shared Memory
   - Mixed Programming Environment
4. Key Technologies

CRPC

Challenges of Computational Grids

- **Heterogeneity**
  - node power
  - architecture
  - implementation
- **Latencies**
  - long
  - variable
- **Bandwidths**
  - different for different links
  - different based on traffic

Goals of Application Development Support

- **Applications should**
  - be easy to develop
  - be portable
  - achieve high performance
    close to what is possible by hand
- **Application Developer**
  - should be able to concentrate on problem analysis and decomposition
- **System**
  - should handle details of mapping abstract decomposition onto computing configuration
- **Developer and System**
  - should work together to debug and tune the program

Dimensions of Parallelism

- **Among nodes**
  - typically task or object parallelism
- **Within a node**
  - parallel computing today
  - data parallelism typical
- **Within a single processor**
  - overlap of computing data access
  - overlap instructions with one another

Available Technologies I

- **Autoparallelization**
  - dependence analysis
  - program transformation
  - coarse grain parallelism detection
  - interprocedural analysis and optimization
  - disadvantage
    - feasibility
- **Explicit Communication**
  - PVM, MPI, Globus/Nexus, Active Messages
  - disadvantage
    - programming burden

Available Technologies II

- **Distributed Shared Memory**
  - hardware and software
  - disadvantage
    performance with fine-grained parallelism
- **Data Parallel Languages**
  - HPF, HPC++, ...
  - ease of use
  - disadvantage
    performance of compiled code
- **Task Parallelism**
  - extensions to HPF
  - object parallelism
  - disadvantage:
    limited parallelism

Available Technologies III

- **Libraries**
  - ScaLAPACK, DAGH, P++,...
  - disadvantage:
    restricted to objects and functions covered in library
- **Programming Tools**
  - Pablo, Gist, Upshot, ...
  - essential for use with languages
  - disadvantage
    not enough of them
- **Others**
  - latency tolerance and management
  - load balancing
  - run-time compilation

Paradigm 1: Task Composition

- **Composition of applications from components**
  - graphical interface or scripting language
  - implementation:
    - construction of task graph
    - restructuring for better parallelism and load balance
    - assignments of components
- **Disadvantage**
  - performance inhibited at intertask interfaces
- **Key technologies**
  - compilation of scripting languages
  - interprocedural analysis and optimization

Paradigm 2: Grid Shared Memory

- **Extension of Software DSM**
  - permits threading and parallel loops
  - user or automatic computation assignment
- **Disadvantages**
  - performance for finer-grained parallelism
  - requires significant programmer and compiler assistance
  - feasible for distributed computing?
- **Key technologies**
  - software DSM
  - compiler assistance for communication performance
  - performance estimation

Paradigm 3: Compilation for Grid

- **Extension of Languages like HPF**
  - Nexus/Globus communication layer as target
  - differences from HPF
    - nonuniform loads in some dimensions
    - advanced data structure representations
    - mixture of parallelism styles
- **Strategy**
  - data structure decomposition
    - decompositions attached to data structures
    - whole program analysis required
  - functional (task) decomposition
  - adaptivity: migration of work
  - latency management
  - performance estimation
  - integration of libraries
- **Disadvantage**
  - like HPF may not be efficient early
  - feasible?

CRPC

Key Technologies

- **Performance Estimation**
  - whole program
  - including run-time issues
- **Whole-Program Compilation**
  - compile, link, and run time
- **Run-Time Compilation**
  - irregular and adaptive
- **Libraries**
  - data structures and functional
  - integration by compiler
- **Programming Support Tools**
  - debugging
  - performance display and tuning

Summary

- **Distributed Heterogeneous Computing Support Is Hard**
  - differing node powers and architectures
  - long and variable latencies
  - differing bandwidths
- **Existing Technologies Promising But Inadequate**
  - extensions will be needed
- **Three Paradigms**
  - task composition
  - global shared memory
  - compilation for grid