Parallel Programming

Marek Zimanyi

Parallel Programming Models 1

Message Passing vs. Threading

- Individual processes exchange messages
  → explicit com'
- Works on clusters and on parallel computers
  → Distributed memory and shared memory
- Manual parallelization

- One process (environment), multiple threads
  → implicit com'
- (Usually) only on Symmetric MultiProcessor systems
  → (virtually) shared memory systems
- (Semi-) automatic parallelization too

Material based on EG 2001 tutorial
http://www.gris.uni-tuebingen.de/~bartz/tutorials/eg2001tutorial/

Parallel Programming Models 2

Message Passing vs. Threading

- Interconnection network (switches, ethernet, etc.)
- Distributed memory (no common accessible memory)
- MPI 2.x / PVM 3.x
- Limited concurrency control

- One common interconnect (bus,crossbar)
- UMA / NUMA memory
- OpenMP / pthreads
- Flexible concurrency control

Outline

- **Message Passing**
  - Message Passing Interface (MPI)
  - (Parallel Virtual Machine) (PVM)
- **Threading in Shared Memory**
  - OpenMP
  - Pthreads

Message Passing (1)

Overview

- Individual processes
- Explicit communication by exchange of messages
- On shared-memory and on distributed memory systems

Message Passing (2)

Message-Passing Interface (MPI) **vs.** **Parallel Virtual Machine (PVM)**

- Limited session control
- Supports portability only
- Rich com' functionality
- Performance oriented

- Rich session control
- Supports portability and interoperability
- Flexibility and fault-tolerance

Message Passing (4)

When to use MPI, when to use PVM?
(Answers in Geist et al. 96)

- **MPI for parallel computers**
- **PVM for clusters**

However, PVM seems to loose significance

Outline

- **Message Passing**
  - Message Passing Interface (MPI)
  - (Parallel Virtual Machine) (PVM)
- **Shared Memory**
  - OpenMP
  - Threading using Pthreads

MPI

Message Passing Interface

- Current version 2 (MPI 2.x)
- Supports portability, not interoperability
- Works on clusters, but focus is on “large multi-processors”
- Task distribution done by vendor implementation

MPI

Message Passing Interface, cont'd

- Individual processes synchronize at one point of execution, or exchange messages
- Rich variety of communication mechanisms
- (Almost) no resource/session management in standard, but vendor specific tools are usually available

MPI - Management

Management:

- Individual process subscribe to MPI:
  `int MPI_init(int* argc, char **argv);`
- Parallel code/tasks, synchronization and exchange of messages
- Unsubscribing from MPI:
  `int MPI_finalize(void);`

  error return code

MPI – Grouping amd communication

- **Groups provide**
  - Support for parallel libraries (hides internal communication)
  - Scope for communication (com') and synchronization (sync')
- **Elements: communicator, group, context**
- Intracommunication - within a group
  - Point-To-Point com' (pairwise)
  - Collective com' (root to members)

PVM - General

Parallel Virtual Machine

- Current version 3 (PVM 3.x)
- Good for communication (com') in cluster
- Virtual Machine interoperability

PVM - General

- Meets specific needs of cluster computing:
  - Dynamic resource management
  - Fault-tolerant applications
- One master: distributes data and subtasks
- Several slaves: perform subtasks on data
- Participating systems listed in hostfile
- PVM daemons (virtual machines) are running on participating systems

Parallel Volume Rendering
(some notes …)

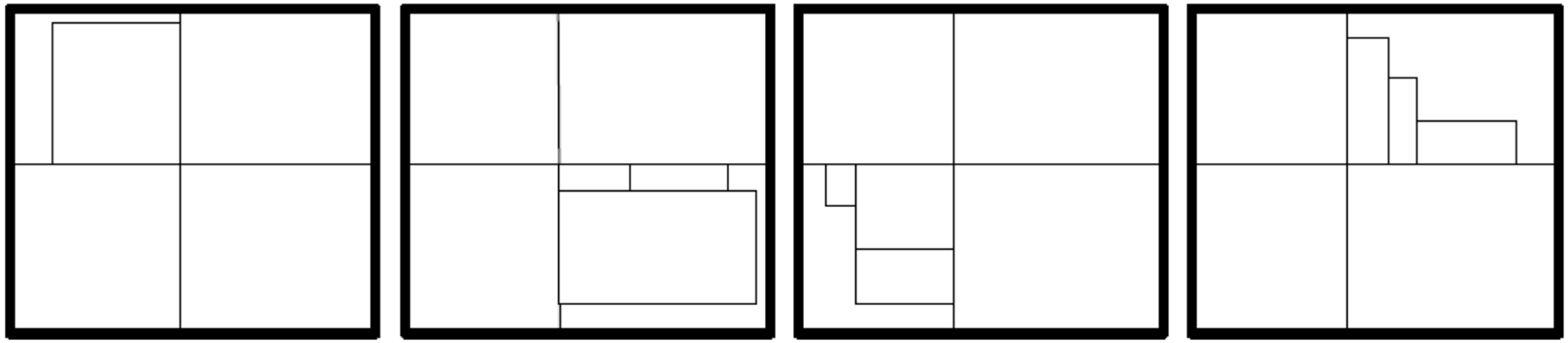
- **Regular-Grid Techniques**
  - **Shared-Memory Ray Casting**
  - **Distributed-Memory Ray Casting**
  - **Shared-Memory Shear-Warp**
- Problem: Load balancing

Load Balancing

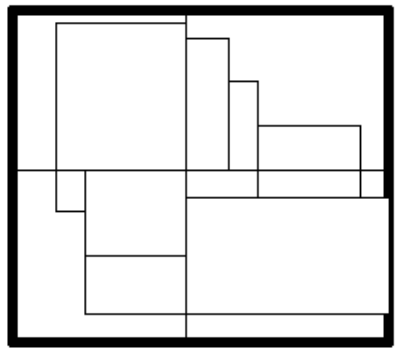
- **Static versus Dynamic**
- **Data locality and cache coherence**
- **Scalability**

Solution

- K-d trees
  (**Ma, Painter, Hansen, Krogh 93**)



Final Image:



Solution

- BSP trees (**Silva 94, 96, 99**)
  - **Always subdivide along the largest "axis"**
- **Optimal Processor Allocation for BSP-tree compositing**