Janus
Parallel Generic Programming for Scientific Applications

Jens Gerlach

RWCP Parallel and Distributed Systems
Fraunhofer-Laboratory at FIRST

Jens.Gerlach@first.fraunhofer.de
Scope of Janus

• Data Parallelism
  – Simultaneous operations across large sets of data, rather than on multiple threads of control

• Generic Programming
  – Programming for families of abstractions
  – Well-known through C++ standard library
Objects in Scientific Applications

- Spatial structures
  - grids
  - triangulations
  - graphs
  - cells

- Associated data
  - grid functions
  - element matrices
  - sparse matrices
  - particles

Spatial structures are sets of objects with relations
Associated data are functions on these sets
Features of Spatial Structures

- Spatial Structures are very diverse
  - complex geometric and combinatoric informations
  - regular or irregular
  - static or dynamic
  - distributed representation

- It is hard to define a general framework for the spatial structures
General Features

• Spatial Structures are more stable than the associated data

• Janus is a C++ template library that provides building blocks for data parallel scientific applications
The Janus Framework

- Domain
  - Finite sets
- Relation
  - Element dependences
- Domain Function
  - Data on domain/relation
- Algorithms
What is a Domain?

- Finite sequence of elements
  
  ```cpp
  jns::domain<int> x; // node set of a mesh
  x.insert(3);
  x.insert(42); // ...
  x.freeze();
  for(size_t i = 0; i < x.size(); i++)
    cout << x[i] << endl;
  ```

- Every element has a unique index (*position*)

  ```cpp
  size_t pos = x.position(3);
  ```
What is a Domain?

- Mapping (for distributed processing) can be specified at insertion time
  - x.insert_at(3, 1);
  - x.insert_at(42, 0);
  - x.freeze();

- Clear separation of insertion and retrieval
  - Insertion is allowed only before freeze
  - Retrieval is allowed only after freeze
Domain Classes

- vertices
- edges
- triangles

Example: triangulation with three domains

```cpp
jns::grid<N> (regular domain)
jns::domain<Type> (irregular domain)
```
Domain Function Concept

- Association through same positions
- Can be modelled through 1D arrays
What is a Relation?

• Describes dependences of domain elements
  • Domains have no visible structure!
  • Uses positions of domain elements
  • Represented as sparse adjacency matrix

• Provides sparse matrix operations
  • Matrix vector multiply (also transposed)
Relation Classes

\[
\text{jns::stencil\langle N, \text{Generator}\rangle} \\
\text{jns::n\_relation\langle \text{Domain1}, \text{Domain2}, \text{Generator}\rangle} \\
\text{jns::relation\langle \text{Domain1}, \text{Domain2}\rangle}
\]
Communication Algorithms

Relation.pull(const Source& b, Target& a)
Relation.push(const Source& a, Target& b)

...
More about freeze

- Relations can also be frozen
  - Useful for irregular relations
- Freeze defines *Insertion* and *Retrieval* phases
  - Fits usage patterns of scientific applications
- Freeze enables important optimizations
  - Internal restructuring of domains for faster access
  - Evaluate mapping information only in freeze
  - Negotiating buffer sizes only once
Applications

Janus is well suited for

- Cellular automata
- Finite Element and Finite Volume Methods
- Multigrid methods
- Data parallel graph algorithms
Finite Element Method

- Adaptive refinement of a distributed FE mesh
- Automatic repartitioning with Mosaik
- Also used for parallel 3D crack simulations at Cornell University
Refinement and Coarsening

- Adaptive refinement and coarsening of a distributed FE mesh
- Repartitioning with Mosaik
Parallel Graph Algorithms

- Bellman-Ford (single source) shortest path algorithm
- Tree construction based on a parallel relaxation algorithm
Platforms

• Required software
  – Good C++ compilers (GCC, KAI, Intel)
  – MPI
  – OpenMP port exists

• Supported operating systems and hardware
  – Linux clusters
  – Windows cluster (Cornell)
  – CrayT3E, IBM SP2
Summary of Janus

- Conceptual framework to model spatial structures and associated data

- Efficient generic data structures for regular and irregular (distributed) problems
  - Works also with C and Fortran arrays

- *Basic* building blocks *not* complete solutions
Who are we?

- Janus is developed by the PROMISE group and can be downloaded
  [www.first.gmd.de/janus](http://www.first.gmd.de/janus)
- Funded by the Real World Computing Partnership, Japan ([www.rwcp.or.jp](http://www.rwcp.or.jp))
- See you at the RWCP booth at SC'2001