Case Study: Atmosphere Model

- Simulation of atmosphere for weather prediction, studying climate, etc.
- Must solve a set of partial differential equations describing the basic fluid dynamical behavior of the atmosphere
- 3D grid of values
- Time integration to determine state at future time, based on current state
- Stepwise refinement using finite difference method, 9-point stencil in horizontal direction, 3-point in vertical
  - Fluid dynamics of the atmosphere
  - Physics computations: radiation, convection, and precipitation simulations within vertical columns

Atmosphere Model Algorithm Design

- Partitioning
  - Domain decomposition is a logical choice
  - For maximum possible concurrency, use one task per grid point
- Communication
  - 3 distinct computations
    - Finite difference stencils
      - 9-point horizontal (each communicates with 8 neighbors)
      - 3-point vertical (with 2 neighbors)
    - Global operations
      - Computes total mass of atmosphere to verify simulation is proceeding correctly
      - Parallel summation
    - Physics computations
      - Prefix product over each vertical column
      - On the order of 30 communications per grid point and per time step
      - This component is a bit problematic with respect to communication

Atmosphere Model Algorithm Design (cont.)

- Agglomeration
  - Approximately 10^5 to 10^7 tasks
  - Agglomeration to 4 (2x2) horizontal grid points per task reduces communication form 8 to 4 messages per time step
  - Agglomeration in vertical direction avoids communication for the finite difference stencil (2 messages) as well as physics computation (30 messages)
    - Also avoids software engineering costs because can reuse sequential code
- Mapping
  - Simple straight-forward mapping, yielding a SPMD program with each processor responsible for several columns
  - May be unbalanced if physics computations vary from region to region; if so, consider a cyclic mapping

Case Study: Floorplan Optimization

- Layout of a full-custom ASIC (application-specific integrated circuit)
  - Production of a set of indivisible rectangular blocks, called cells
  - Relative placement determined using interconnection information
  - Floorplan optimization: implementations are selected for each cell so as to minimize the total area
- The problem
  - Alternative implementations for each cell, with varying areas and aspect ratios (length and width)
  - G and H graphs that specify which cells must be adjacent in the vertical and horizontal directions, respectively
  - Find the implementation that minimizes total area, subject to G and H constraints

Floorplan Optimization Algorithm Design

- Explore a search tree representing all possible configurations
  - Level i corresponds to implementations chosen for i cells
  - Feasible only for small implementations
- Branch and bound search
  - If the area of a node in the tree is greater than the best known solution so far, the subtree rooted at this node can be abandoned (or pruned)
    - In one experiment, pruning reduced number of nodes from $4x10^{15}$ to $6x10^{6}$
  - Difficulties
    - Must keep track of best (lowest area) solution seen so far
    - Must manage the order in which the tree is explored (preferably depth-first search)

Floorplan Optimization Algorithm Design (cont.)

- Partitioning
  - No obvious data structure for domain decomposition
  - Use fine-grained functional decomp., one task for each search tree node
    - Tasks will be created as a wavefront, and tree will tend to be explored breadth-first
    - Only tasks on the wavefront can execute concurrently
    - How to handle best solution so far? Assume a single task maintains it
- Communication
  - Tasks must obtain and update best soln.
  - Centralized task to maintain best solution
    - Simple, may work, but not very scalable
    - Check with manager only periodically
    - Use subtrees, each with a submanager
    - May need performance model to evaluate

Floorplan Optimization Algorithm Design (cont.)

- Agglomeration
  - Create one task for each search until the search reaches a certain depth, then switch to depth-first search and evaluate sequentially
  - Tasks are presumably (?) executed in the order they are created
    - Tends toward breadth-first search
    - Must control this search order in mapping
- Mapping
  - Task scheduling: tasks become problems executed by worker tasks
  - Central manager explore search tree to some depth, then creates worker tasks on demand and assigns them to idle workers
    - Some pruning possible
    - Each processor executes a whole subtree, so can do some more pruning