Applications

- Easy (embarrassingly parallel)
  - multiple independent jobs (i.e..., different simulations)
- Scientific
  - linear algebra
  - particule simulations
- Databases
  - biggest success of parallel computing
  - exploits semantics of relational calculus
- AI
  - search problems
  - pattern recognition and image processing (main SIMD use)

Issues in Application Performance

- Speedup
  - ratio of time on n nodes to time on a single node
  - should really compare to best serial time
  - goal is linear speedup
  - super-linear speedup is possible due to:
    - adding more memory
    - search problems
- Amdahl's Law
  - max speedup is 1/(serial fraction of time)
- Computation to Communication Ratio
  - goal is to minimize this ratio

How to Write Parallel Programs

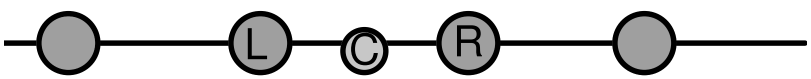
- Use old serial code
  - compiler converts it to parallel
  - called the dusty deck problem
- Serial Language plus Communication Library
  - no compiler changes required!
  - PVM and MPI use this approach
- New language for parallel computing
  - requires all code to be re-written
  - hard to create a language the provides performance on different platforms
- Hybrid Approach
  - HPF - add data distribution commands to code
  - add parallel loops and synchronization operations

Application Example - Weather

- Typical of many scientific codes
  - computes results for three dimensional space
  - compute results at multiple time steps
  - uses equations to describe physics/chemistry of the problem
  - grids are use to discretize continuous space
    - granularity of grids is an important to speed/accuracy
- Simplifications (for example, not in real code)
  - earth is flat (no mountains)
  - earth is round (poles are really pointy)
  - second order properties

Grid Points

- Divide Continuous space into discrete parts
  – for this code, grid size is fixed and uniform
    • possible to change grid size or use multiple grids
  – use three grids
    • two for latitude and longitude
    • one for elevation
    • Total of M * N * L points
- Design Choice: where is the grid point?
  – left, right, or center of the grid



  – in multiple dimensions this multiples:
    • for 3 dimensions have 27 possible points

Variables

- One dimensional
  - m - geo-potential (gravitational effects)
- Two dimensional
  - pi - “shifted” surface pressure
  - sigmadot - vertical component of the wind velocity
- Three dimensional (primary variables)
  - <u,v> - wind velocity/direction vector
  - T - temperature
  - q - specific humidity
  - p - pressure
- Not included
  - clouds
  - precipitation
  - can be derived from others

Serial Computation

- Convert equations to discrete form
- Update from time t to t + delta t

```
foreach longitude, latitude, altitude
        ustar[i,j,k] = n * pi[i,j] * u[i,j,k]
        vstar[i,j,k] = m[j] * pi[i,j] * v[i,j,k]
        sdot[i,j,k] = pi[i,j] * sigmadot[i,j]
end
foreach longitude, latitude, altitude
        D = 4 * ((ustar[i,j,k] + ustar[i-1,j,k]) * (q[i,j,k] + q[i-1,j,k]) +
                        terms in {i,j,k}{+,-}{1,2}
        piq[i,j,k] = piq[i,j,k] + D * delat
        similar terms for piu, piv, piT, and pi
end
foreach longitude, latitude, altitude
        q[i,j,k] = piq[i,j,k]/pi[i,j,k]
        u[i,j,k] = piu[i,j,k]/pi[i,j,k]
        v[i,j,k] = piv[i,j,k]/pi[i,j,k]
        T[i,j,k] = piT[i,j,k]/pi[i,j,k]
end
```