Scientific Computing I

Conclusion and Outlook

Michael Bader

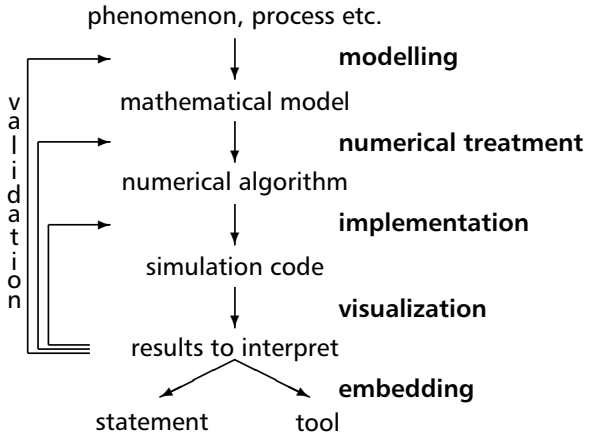
Lehrstuhl Informatik V

Winter 2005/2006

Part I

Implementation

The Simulation Pipeline



Target Architectures

Monoprocessors

- workstation
- mobile computing (laptop, PDA)
- embedded computing

vs. Parallel Computing

- cluster of workstations
- low- to medium-cost parallel computer
- supercomputer (BlueGene, ALTIX)

CPU & Memory

Current Trends:

- RISC, load/store architecture
- superscalar processors, multi-core architecture
- pipelining, vectorization
- memory gap: CPU performance grows faster than speed of memory
- multilevel cache hierarchies

Performance may drop to (considerably) less than 10% of peak performance, if such aspects are not considered.

Parallel Architectures – Classification

- SIMD vs. MIMD
  (*single/multiple instruction, multiple data*)
- UMA (*Uniform Memory Access*, shared memory)
  NORMA (*No Remote Memory Access*, distributed memory)
  NUMA (*Non-Uniform Memory Access*, virtually shared memory)
- Clusters of workstations
  vs. dedicated parallel computers
- communication hardware (Ethernet, Myrinet, Infiniband, vendor solution, . . .)
- topologies (bus, ring, hypercube, crossbar, . . .)

Parallel Programming, Parallel Methods

- vector computing, vectorization
- shared memory computing (OpenMP)
- message passing (MPI):
  explicite communication between processors
- domain decomposition:
  - partitioning of the computational domain
  - compute and combine solution on separate partitions (numerical method)

Parallel Performance

Criteria for parallel performance:

- are there inherently sequential parts in the code?
  Amdahls law: speedup $\leq \frac{1}{\text{seq}}$
- speed/amount of (slow) communication
- load balance and load balancing

Characterization:

- speedup: $S = \dfrac{\text{sequential runtime}}{\text{parallel runtime}}$
- efficiency: $E = \dfrac{S}{\#\text{ processors}}$

Twelve Ways to Fool the Masses

(selection)

- Scale up the problem size with the number of processors, but omit any mention of this fact
- Compare your results against scalar, unoptimized code on Crays
- When direct run time comparisons are required, compare with an old code on an obsolete system
- If MFLOPS rates must be quoted, base the operation count on the parallel implementation, not on the best sequential implementation

Twelve Ways to Fool the Masses (2)

(continued)

- Quote performance in terms of processor utilization, parallel speedups or MFLOPS per dollar
- Measure parallel run times on a dedicated system, but measure conventional run times in a busy environment
- If all else fails, show pretty pictures and animated videos, and don't talk about performance

Part II

Visualization

Visualization

- visual/graphical/optical representation of large sets of data
- data from experiments or measurements:
  satellite images, tomography in medicine, microsopy, ...
- data from simulations:
  - with resolution of space (and time): fluid mechanics, structural mechanics, quantum physics, ...
  - without spatial resolution: vehicle dynamics, optimal control, ...
- methods from image processing, computer graphics, virtual/augmented reality

Image Processing

- geometric: zoom, rotation, . . .
- point-to-point: false colours,
- local: averaging, filtering, edge detection, . . .
- ensemble processing
  (different images of the same scene)
- domain processing
  (Fourier/cosine transform, wavelet transform, tomography)

Computer Graphics

- geometric modelling (representing 3D objects)
- graphical representation/rendering
  (perspective, illumination, shading)
- animation

Visualisation of Simulation Data

Common Techniques:

- (ortho-) slices, projection
- contour lines, isosurfaces
- streamlines, streaklines/-bands
- particle tracing

13 Ways to Say Nothing with Scientific Visualization

(selection)

- Never include a colour legend
- Avoid annotation
- When in doubt, smooth
- Avoid providing performance data
- Never learn anything about the data or scientific discipline

13 Ways to Say Nothing with Scientific Visualization

(continued)

- Never compare your results with other visualization techniques
- Avoid visualization systems
- Claim generality but show results from a single data set
- use viewing angle to hide blemishes
- if viewing angle fails, try specularity or shadows
- "this is easily extended to 3D"

Part III

Embedding

Embedding

step 1:

From numerical methods to software packages

- usability
- interfaces to tools for
  - modelling (e.g. CAD)
  - libraries (BLAS, LAPACK, . . .)
  - visualization
- maintenance of software

$\Rightarrow$ Software Engineering

Computational Steering

step 2:

Computational steering:

- steering of an entire development cycle
- interactive w.r.t.
  - model (assembly components; physical model)
  - simulation parameters
  - visualization technique
  - . . .
- short feedback-loop between development and simulation results
- requires integration of data structures, software components, etc.