Introduction to Parallel Computing

Murat Manguoğlu

Middle East Technical University
Computer Engineering

CENG478 – Spring 2020
Random Access Machine (RAM) Model

Random Access Machine is a favorite model of a sequential computer. Its main features are:

• Computation unit with a user defined program
• Read-only input tape and write-only output tape
• **Unbounded** number of local memory cells
• Each memory cell is capable of holding an integer of unbounded size
• Instruction set includes operations for moving data between memory cells, comparisons and conditional branches, and simple arithmetic operations
• Execution starts with the first instruction and ends when a HALT instruction is executed
• All operations take unit time regardless of the lengths of operands.

• **Time complexity** = the number of instructions executed
• **Space complexity** = the number of memory cells accessed

Source: [http://pages.cs.wisc.edu/~tvrdik/2/html/Section2.html](http://pages.cs.wisc.edu/~tvrdik/2/html/Section2.html)
Parallel Random Access Machine (PRAM) Model

- Natural extension of RAM: each processor is a RAM
- Processors operate synchronously
- Earliest and best-known model of parallel computation
- 3 types based on shared memory access:
  - Exclusive Read Exclusive Write (EREW), Concurrent Read Exclusive Write (CREW) and Concurrent Read Concurrent Write (CRCW)
Flynn’s Taxonomy of Computers
Michael J. Flynn - 1966

• Single Instruction Single Data (SISD)
• Single Instruction Multiple Data (SIMD)
• Multiple Instruction Single Data (MISD)
• Multiple Instruction Multiple Data (MIMD)
Shared Address Space

Uniform Memory Access (UMA) vs Nonuniform Memory Access (NUMA)
Distributed Address Space

Communication Network
Programming models and another classification based on the address space

- **Process** based programming model, communicate via messages (message passing) and can use
  - Shared address space (UMA and NUMA)
  - Distributed address space
  - or Shared/Distributed address space platforms
- **Thread** based programming model communicate via shared variables and can use
  - Usually shared address space platforms only
Limiting (hardware) factors in parallel computing

- Processor
- Memory
- Datapath
Difficulties/Limitations in Parallel Computing

• Automatic parallization is not efficient and developing/implementing parallel algorithms is not easy
  • Usually «the best sequential algorithm» ≠ «best parallel algorithm»
  • How to partition the problem? Load balancing? Communication/Synchronization? How to debug parallel code?

• Highly platform dependent → often low level
• Finding enough parallelism in a problem is not always possible
  → Amdhal’s Law and Gustafson’s Law
Difficulties/Limitations in Parallel Computing

How much parallelism can we have to complete the task of planting just one tree?

What are the limitations of planting trees to create a forest?

Image: http://www.metu.edu.tr/tr/system/files/gallery/img_2067.jpg

Image: https://metusharks.files.wordpress.com/2014/11/20141102_112440.jpg
Applications/Problems/Algorithms

- Embarrassingly parallel
- Fine grained
- Coarse grained
Applications/Problems/Algorithms

- CPU bound
- Memory bound
- I/O Bound
Levels of Parallelism

- Bit Level
- Instruction Level
- Data Level
- Task Level

Easier for the compiler/processor to handle
Applications

1. Scientific/Engineering
2. Others
What is Scientific Computing?

“Scientific computing is the collection of tools, techniques, and theories required to solve on a computer mathematical models of problems in science and engineering... In summary, scientific computing draws on mathematics and computer science to develop the best ways to use computers to solve problems from science and engineering.”

Figure 1.1: Scientific Computing and Related Areas

Applications of Parallel Computing

• It used to be the largest problems in the past
• Today the applications are everywhere, but still the largest problems have more **impact** and more **challenging**
• **example:** George Smoot who won the Nobel Prize says:
  «...Computing has played a big role from the very beginning of what is a long series of experiments, and the computing requirements have grown with each experiment...»
  «...For five years, or probably 10 or 15 years, cosmology is going to stress large-scale computing in a serious way..»

Source: [http://scidacreview.org/0704/html/interview.html](http://scidacreview.org/0704/html/interview.html)
Applications of Parallel Computing

Uncovering Alzheimer's complex genetic networks

Two CyberShake hazard models for the Los Angeles region calculated on Blue Waters using a simple 1D earth model (left) and a more realistic 3D earth model (right). Seismic hazard estimates produced using the 3D earth model show lower near-fault intensities due to 3D scattering (1), much higher intensities in near-fault basins (2), higher intensities in the Los Angeles basins (3), and lower intensities in hard-rock areas (4).

Source: http://www.ncsa.illinois.edu/news/story/do_the_wave

And others: nanotechnology, environmental engineering, quantum computing, physics, biology, chemistry, mechanical engineering, social sciences,.....
## Applications and Kernels/Algorithms

<table>
<thead>
<tr>
<th>Applications</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>lattice gauge (QCD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>quantum mechanics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>weather</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CFD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>geodsy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inverse problems</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>structures</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>device simulation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>circuit simulation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>electromagnetics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. linear systems
2. least squares
3. nonlinear systems
4. eigenvalues/SDVs
5. fast transforms
6. rapid elliptic solvers
7. multigrid
8. stiff ODE
9. Monte Carlo
10. integral transforms

Motif/Dwarf: Common Computational Methods

What do applications have in common?

(Red Hot → Blue Cool)

1 Finite State Mach.
2 Combinational
3 Graph Traversal
4 Structured Grid
5 Dense Matrix
6 Sparse Matrix
7 Spectral (FFT)
8 Dynamic Prog
9 N-Body
10 MapReduce
11 Backtrack/ B&B
12 Graphical Models
13 Unstructured Grid

Source: http://www.cs.berkeley.edu/~demmel/cs267_Spr12/
Figure 5. Intel’s RMS and how it maps down to functions that are more primitive. Of the five categories at the top of the figure, Computer Vision is classified as Recognition, Data Mining is Mining, and Rendering, Physical Simulation, and Financial Analytics are Synthesis. [Chen 2006]
HPC Software Ecosystem

- Programming Languages/Models
  - C, C++, Fortran, ...
  - MPI, OpenMP, pthreads, cuda ...

- Libraries/programming environments
  - Low level
    - BLAS
    - Lapack/Scalapack
    - Various Parallel Sparse Linear Solvers
    - PARPACK - eigensovlers
    - ...
  - High level
    - General Purpose
      - PETSC
      - Trilinos
      - Julia
    - Domain specific
HPC: Programming languages and parallel programming paradigms

source: https://www.nag.com/content/not-only-fortran-and-mpi-pops-view-hpc-software-europe
BLAS: Basic Linear Algebra Subroutines

- Basic dense matrix/vector operations
  - Vector-Vector (Level 1)
  - Matrix-Vector (Level 2)
  - Matrix-Matrix (Level 3)

- Usually optimized for a given platform usually multithreaded
  - Intel MKL BLAS
  - Atlas BLAS
  - Goto BLAS
  - cuBLAS
  - ....
LAPACK/ScaLAPACK: Scalable/ Linear Algebra Package

- Dense matrix operations that are higher level than the ones in BLAS, LAPACK/ScaLAPACK use BLAS
  - solving linear systems
  - solving least-square problems
  - solving eigenvalue problems
  - solving singular value problems
- Usually optimized for a given platform, LAPACK is sequential/multithreaded and ScaLAPACK is parallel using message passing
  - Intel MKL
  - MAGMA (~cuda LAPACK)
  - ...
Sparse Linear Solver Libraries

- Pardiso  [http://www.pardiso-project.org/](http://www.pardiso-project.org/)
- ...
Sparse Eigensolvers

- Parpack  http://www.caam.rice.edu/~kristyn/parpack_home.html
- Slepc    http://slepc.upv.es/
- FEAST    http://www.feast-solver.org/
- ...
PETSC: Portable, Extensible Toolkit for Scientific Computation

Structure of PETSc

https://www.mcs.anl.gov/petsc/
Trilinos: Algorithms and Enabling Technologies for Large-Scale Applications

Two-level design:
• Self-contained packages (50+)
• Leveraged common tools.
  • Version Control
  • Build System
  • Test Harness

Nonlinear, Transient & Optimization Solvers
Linear & Eigen Solvers
Scalable Linear Algebra
Geometry, Meshing & Load Balancing
Framework, Tools & Interfaces

http://trilinos.sandia.gov
Trilinos: A Layered Collection of C++ Libraries

- Standard C++, Not a language extension
  - *Not* a language extension: OpenMP, OpenACC, OpenCL, CUDA
  - In *spirit* of Intel’s TBB, NVIDIA’s Thrust & CUSP, MS C++AMP, ...

- Uses C++ template meta-programming
  - Previously relied upon C++1998 standard
  - Now require C++2011 for lambda functionality
    - Supported by Cuda 7.0; full functionality in Cuda 7.5
  - Participating in ISO/C++ standard committee for core capabilities

Application & Library Domain Layer(s)

- Trilinos Sparse Linear Algebra
- Kokkos Containers & Algorithms
- Kokkos Core

Back-ends: Cuda, OpenMP, pthreads, specialized libraries ...
SU2: An open source CFD Code

http://su2.stanford.edu/
OpenFOAM: An open source CFD code

http://www.openfoam.com
LAMMPS: Molecular Dynamics Simulator

function mandel(z)
    c = z
    maxiter = 80
    for n = 1:maxiter
        if abs(z) > 2
            return n-1
        end
        z = z^2 + c
    end
    return maxiter
end

function randmatstat(t)
    n = 5
    v = zeros(t)
    w = zeros(t)
    for i = 1:t
        a = randn(n,n)
        b = randn(n,n)
        c = randn(n,n)
        d = randn(n,n)
        P = [a b; c d]
        Q = [a b; c d]
        v[i] = trace((P.'*P)^4)
        w[i] = trace((Q.'*Q)^4)
    end
    std(v)/mean(v), std(w)/mean(w)
end