High Performance Computing for Fast Hybrid Simulations

Boris Jeremić and Guanzhou Jie

Department of Civil and Environmental Engineering
University of California, Davis

CU–NEES 2006 FHT Workshop, Nov. 2-3 2006
Outline
Real Time Simulations

- Structural and Geotechnical component testing is rate dependent

- Algorithmic measures are only partially fixing the problem

- Goal: use real time computational simulations for hybrid testing

- In other words, use real time computational simulations to control the test
Fast Computer Simulations: Various Models and Algorithms

- Implicit vs Explicit (on local and global level, both offer advantages and disadvantages)

- Hierarchy of material models (use only required level of sophistication for the purpose)

- Reduced Order Modeling (ROM) now available for nonlinear models
Standard Application Programming Interface: OpenSees

- Core framework (class libraries UCB, modifications by UCD),
- Material models (UCD, UCB, UCSD, UW, Stanford ...)
- Finite Elements (UCD, UCB, Stanford ...)
- Procedures (loading, dynamics, solution control) (UCD, UCB, Stanford ...)
- Parallel Simulations (Plastic Domain Decomposition, UCD)
- Official Manuals available (UCB, personally do not like it)
- Our own effort UCD, covering only our efforts, but will expand to cover other areas of interest.
  - Theory
  - Command (interactive)
  - Examples (verification and validation)
Fast Computer Simulations: Programming (CompSci)

- Mix and match efficient programming languages (C++, C, Fortran, Tcl, Python...)
- Efficient compilers (GNU, Portland Group, Intel (KAI), Metroworks)
- HPC programming techniques (loop unrolling, template metaprograming...)
Fast Computer Simulations: Hardware

- Fast sequential machines (balance multiple levels of cache and RAM, disk, CPUs clock speed, data bus speed...)

- Parallel machines (SMPs and DMPs, networking...)

- It it time for another Finite Element Machine (NASA endeavor from some 20 years ago)
Symmetric Multiprocessor Parallel (SMP) Machines

- aka Shared Memory Parallel machines (dual core machines count too)
- Shared memory communications, bandwidth 555.831 MB/s, latency: 7.897 micro seconds
- Cache coherence problem
- Not scalable above 16 or so CPUs
- High price
- Fine grained parallelism (matrix–vector multiply...)
- Automatic tools for parallelization (somewhat effective)
Distributed Memory Parallel (DMP) Machines

- aka Beowulf clusters (or clusters of SUNs as available at CU in ’95)
- Networked communications: bandwidth 74.935 MB/s, latency: 62.487 micro seconds
- Message passing paradigm for parallel programming
- Scalable to a large number of machines (CPUs, thousands)
- Very good price/performance ratio (low)
- Communication controls performance
- problems arise with power consumption, cooling and placement area
Clusters of Clusters

- aka DMPs of SMPs
- SMPs can be used as DMPs, with a very fast network
- Cluster a number of SMPs (dual core machines count too)
- Even better price/performance ratio (low)
- Multiple levels of communication performance
- Very efficient and flexible setup!
- Example: GeoWulf \((8 \times 2 + 1 \times 4 + 1 \times 8 + 4 \times 1)\) and adding as funding permits
- Example: Hemisphere \((64 \times 2)\)
Interdependence of Software and Hardware

- Specifics of the problem dictate the hardware design
- Example #1: large scale (20,000+ DoF) problems (SFSI for example, solids for soils, beams and plates/shells for structure) can be efficiently simulated on DMPs, while SMPs might be too restrictive (size)
- Example #2: Smaller scale problems (1,000+ DoF) can be efficiently simulated on SMPs and/or DMPs (mostly in real time)
- Example #3: Small scale problems (below 1,000 DoF) are best suited for SMPs can be efficiently simulated on SMPs and even DMPs (in real time)
- PDD is designed to take advantage of all the above architectures automatically
Outline
PDD Method: Design Goals

- Graph partitioning → balance multiple phases simultaneously, while also minimizing the inter-processor communications costs

- It is a multi-objective optimization problem (minimize both the inter-processor communications, the data redistribution costs and create balanced partitions)

- Take into the account (deterministic or probabilistic):
  - heterogeneous element loads that change in each iteration
  - heterogeneous processor performance (multiple generations nodes)
  - inter-processor communications (LAN or WAN)
  - data redistribution costs
PDD Method: Implementation

▶ Perform global optimization for both (a) internal state determination and system of equations solution phases

▶ Adaptive partitioning done using ParMETIS

▶ Parallel direct and iterative system of equations solvers within PETSC interface. Iterative: CG, GMRES; preconditioners BLOCKSOLVE95, HYPRE and SPAI; Direct: SPOOLES, MUMPS, SuperLU_DIST

▶ OpenSees: standard interface and framework

▶ Works on SMPs, local DMPs, grids of computers
Features

- Initial domain partitioning
- Adaptive domain repartitioning depending on CPU imbalance, LAN and/or WAN performance
- Repartitioning works with loads, constraints..., all necessary movable objects
- Available for elements (solid, structural) with standard OpenSees API (sendSelf, RecvSelf, timer or CL weight estimate)
- Scalable to a large number of CPUs
- Performance tuning (GeoWulf (UCD), TeachingMachine (UCD), Hemisphere (CU Boulder), IA64 (SDSC))
Absolute Speedup Overview

![Graph showing parallel speedup and theoretical speedup for different numbers of CPUs and degrees of freedom (DOFs). The graph includes lines for 398,721 DOFs, 118,275 DOFs, 68,451 DOFs, 32,091 DOFs, and 17,604 DOFs.]
PDD vs DD Speedup Overview

The graph shows the relative speedup of the Plastic Domain Decomposition (PDD) method compared to the Domain Decomposition (DD) method for different numbers of CPUs and varying FE Model Size (DOFs). The x-axis represents the number of CPUs, while the y-axis shows the relative speedup. Different bars correspond to different DOF sizes: 17,604, 68,451, 118,275, and 398,721 DOFs. The graph indicates that PDD generally outperforms DD, with the speedup increasing as the number of CPUs increases.
Outline

HPC for FHT: Plastic Domain Decomposition Method

High Performance Simulation Examples
Detailed 3D FEM model

- Construction process
- Two types of soil: stiff soil (UT, UCD), soft soil (Bay Mud)
- Deconvolution of given surface ground motions
- Use of the DRM (Bielak et al.) for seismic input
- Piles → beam-column elements in soil holes
- Structural model developed at UCB (Fenves et al.)
- Element size issues (filtering of frequencies)

<table>
<thead>
<tr>
<th>model size</th>
<th>el. size</th>
<th>$f_{cutoff}$</th>
<th>min. $G/G_{max}$</th>
<th>$\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>12K</td>
<td>1.0 m</td>
<td>10 Hz</td>
<td>1.0</td>
<td>&lt;0.5 %</td>
</tr>
<tr>
<td>15K</td>
<td>0.9 m</td>
<td>&gt;3 Hz</td>
<td>0.08</td>
<td>1.0 %</td>
</tr>
<tr>
<td>150K</td>
<td>0.3 m</td>
<td>10 Hz</td>
<td>0.08</td>
<td>1.0 %</td>
</tr>
<tr>
<td>500K</td>
<td>0.15 m</td>
<td>10 Hz</td>
<td>0.04</td>
<td>3.0 %</td>
</tr>
</tbody>
</table>
Eight cases considered for sand (S) and/or clay (C) beneath each bent.
Soil and Structure Displacement Response
Soil and Structure Acceleration Spectra
Top of Columns Moment Response (Redistribution)
Fast Hybrid Testing Model

- Real time (fast) simulation is imperative for Fast Hybrid Testing (FHT)!

- Hierarchy of models with different levels of sophistication
  - 2D frame
  - 3D frames
  - 2D excitations
  - 3D excitations
  - material modeling
  - SFSI!
Fast Hybrid Testing Computational Setup

- Parallel computer Hemisphere
- Computer nodes used as DMP of SMPs (cluster of cluster)
- Extremely fast, dedicated interconnect (fiber optics) from Hemisphere to FHT lab (latency below 0.1 ms)
- Allows for possible expansion of computational resources (at the FHT lab), while maintaining LAN with Hemisphere.
- This will create a cluster of clusters, a very efficient (price/performance) design.
- Current Hemisphere users can (transparently) use additional computational resources as well
Fast Hybrid Testing (outside) Networking

- Connection from UCD (GeoWulf) is not fast enough
- Physics (speed of light) issue,
- GeoWulf (UCD) - 1428.76km - FHT;
- Surface fiber optics, one way trip $\sim 7.0ms$ or ideally in vacuum $\sim 4.8ms$,
- In reality it is $10\times$ that much (and it is probabilistic)
- Similarly SDSC (NEESit) is 1329.79km away from FHT
SAC Frame Model, PDD and FHT

- Basic model is a 2D frame on fixed foundations
- Elements and material models: Elastic beams and plastic hinge elements
- Sophisticated model (realistic) is a 3D frame with SFSI effects, 3D seismic wave propagation
- Choice of # of computational nodes on Hemisphere parallel machine depends on the size and sophistication of the model
Computational Trials

- Very small model (2D, or 3D but no SFSI) uses efficiently up to 8 CPUs
- More sophisticated model (3D, SFSI) uses efficiently 32 or more CPUs
- Very sophisticated model (3D struct., SFSI, 3D seismic waves) uses efficiently all 128 CPUs (and would use more if available)
- For a very sophisticated model the challenge is performance (real time, fast)
Parallel Computer Hemisphere Performance

Theoretical

32,212 DOFs
Summary

- High performance, parallel simulations are needed
  - Fast Hybrid Testing (control)
  - High fidelity model based simulations
  - Soil–Foundation–Structure control during seismic events
  - . . .

- PDD parallel finite element code is GPL (available)

- http://sokocalo.engr.ucdavis.edu/~jeremic/PDD/