OpenSees

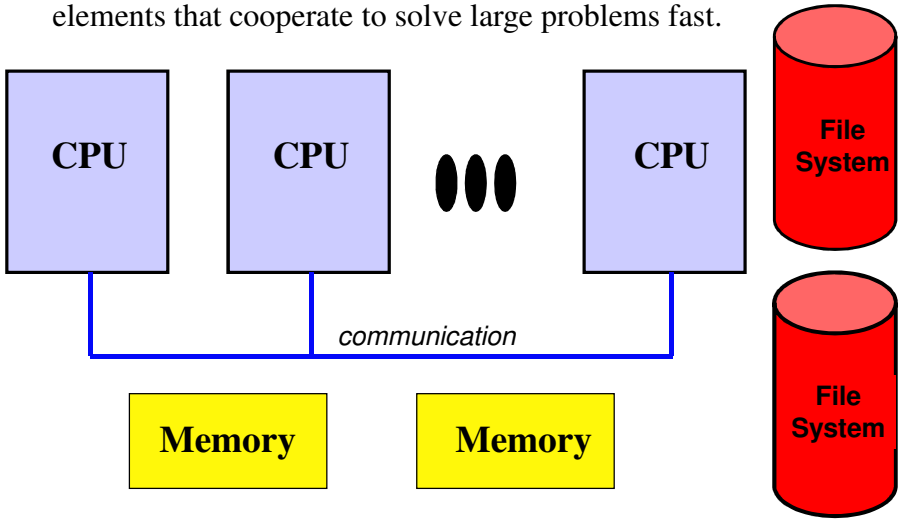
Parallel Processing & OpenSees

Frank McKenna
UC Berkeley

OpenSees User Workshop
September 9, 2008

What is a Parallel Computer?

- A *parallel computer* is a collection of processing elements that cooperate to solve large problems fast.



Why should you care?

- They will save you time
- They will allow you to solver larger problems.
- They are here whether you like it or not!

2008

TOP500 List - June 2008 (1-100)

R_{max} and R_{peak} values are in TFlops. For more details about other fields, check the TOP500 description.

Power data in KW for entire system

ine

Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	DOE/NNSA/LANL United States	Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz , Voltaire Infiniband / 2008 IBM	122400	1026.00	1375.78	2345.50
2	DOE/NNSA/LLNL United States	BlueGene/L - eServer Blue Gene Solution / 2007 IBM	212992	478.20	596.38	2329.60
3	Argonne National Laboratory United States	Blue Gene/P Solution / 2007 IBM	163840	450.30	557.06	1260.00
4	Texas Advanced Computing Center/Univ. of Texas United States	Ranger - SunBlade x6420, Opteron Quad 2Ghz, Infiniband / 2008 Sun Microsystems	62976	326.00	503.81	2000.00

Parallel Machine Classification

- Parallel machines are grouped into a number of types
  1. Scalar Computers (single processor system with pipelining, eg Pentium4)
  2. Parallel Vector Computers (pioneered by Cray)
  3. Shared Memory Multiprocessor
  4. Distributed Memory
     1. Distributed Memory MPPs (Massively Parallel System)
     2. Distributed Memory SMPs - Hybrid Systems
  5. Cluster Systems
  6. Grid

Parallel Programming Models

•A **Programming Model** provides an abstract conceptual view of the structure and operation of a computing system. A computer language and system libraries provide the programmer with this programming model.

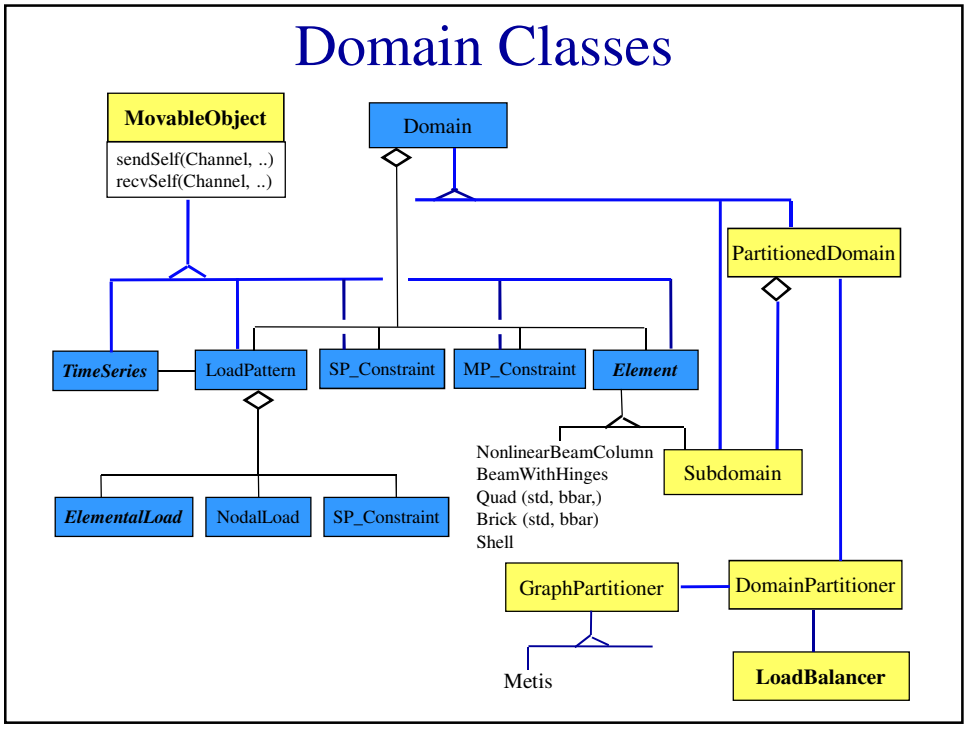
• For parallel programming there are currently **2 dominant** models:

1. **Shared Memory**: The running program is viewed as a collection of processes (threads) each sharing it's virtual address space with the other processes. Access to shared data must be synchronized between processes to avoid race conditions.
2. **Message Passing**:The running program is viewed as a collection of independent communicating processes. Each process executes in it's own address space, has it's own unique identifier and is able to communicate with the other processes.

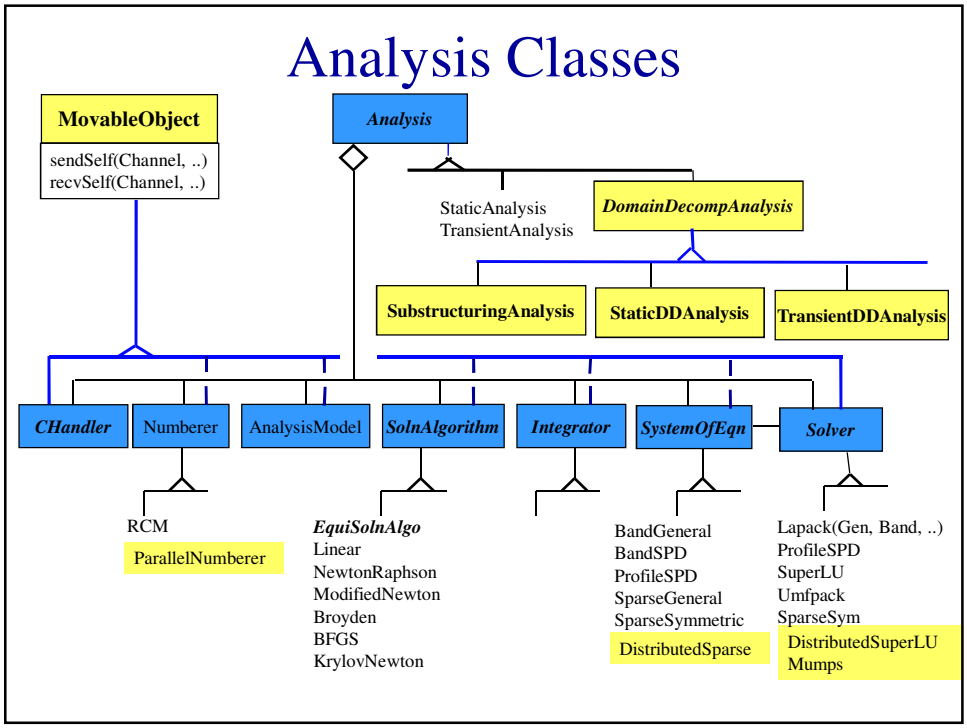
What is OpenSees?

- OpenSees is an Open-Source Software Framework written in C++ for developing nonlinear Finite Element Applications for both sequential and **PARALLEL** environments.
- The OpenSees framework provides classes for the Actor programming model.

Domain Classes



Analysis Classes



The OpenSees Interpreters

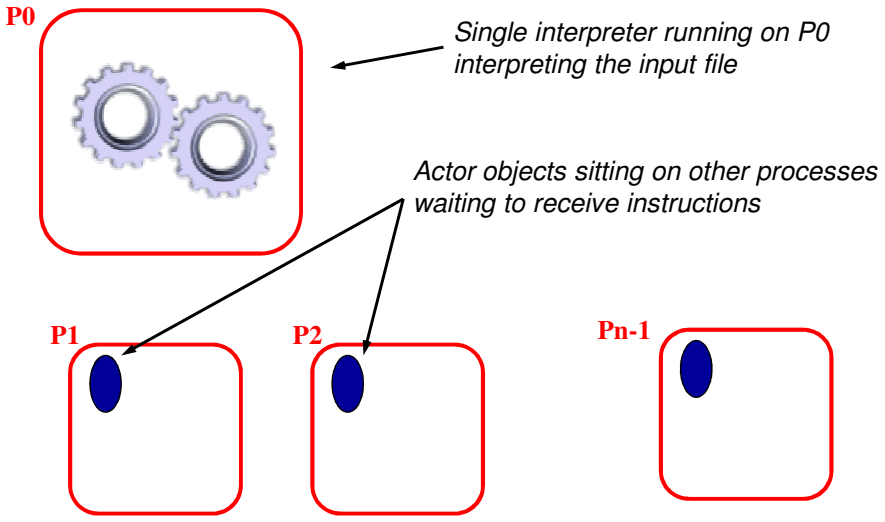
- OpenSees.exe, OpenSeesSP.exe and OpenSeesMP.exe are applications that extend the Tcl interpreter for finite element.

So What are OpenSeesSP.exe and OpenSeesMP.exe ?

Parallel OpenSees Interpreters

- OpenSeesSP: An application for large models which will parse and execute the exact same script as the sequential application. The difference being the element state determination and equation solving are done in parallel.
- OpenSeesMP: An application for **BOTH** large models and parameter studies.

OpenSeesSP: An application for Large Models



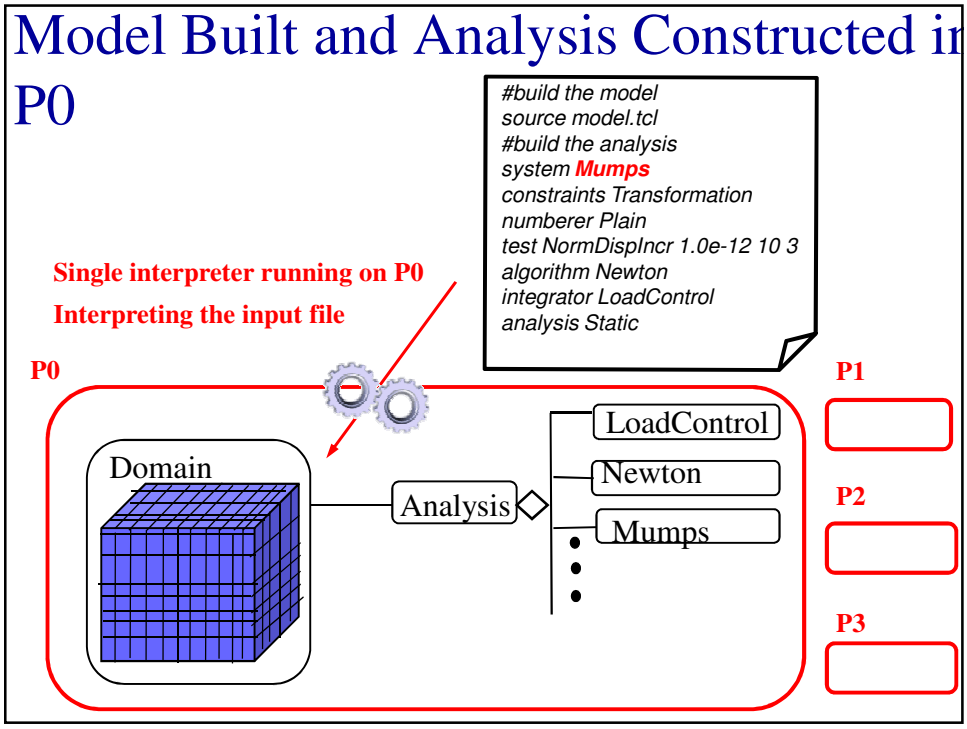
Modified Commands

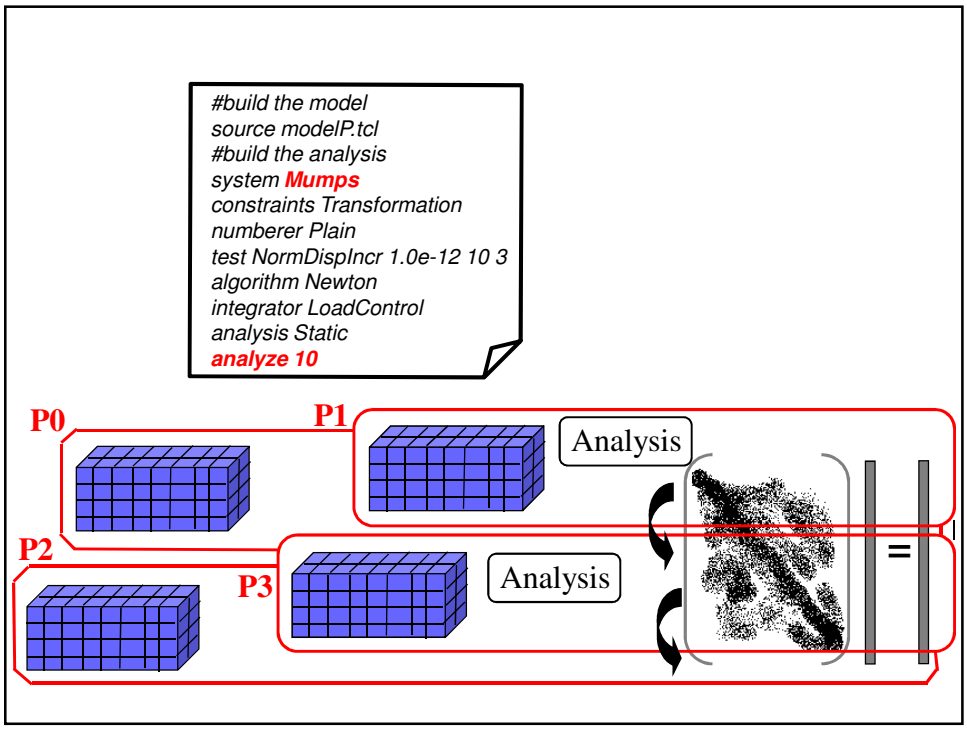
- System command is modified to accept new parallel equation solvers

system Mumps

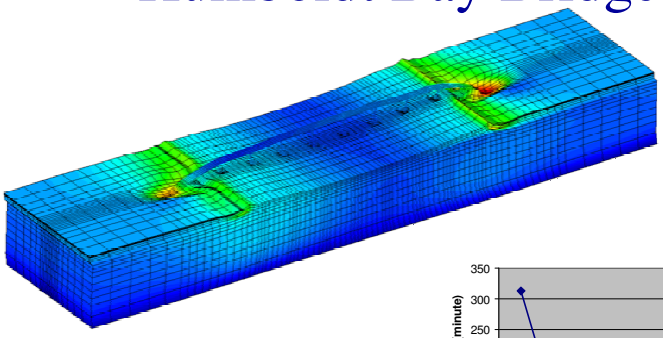
system Diagonal

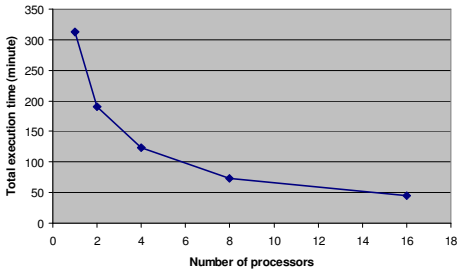
Model Built and Analysis Constructed in P0





Example Usage: Humboldt Bay Bridge Model





OpenSeesMP: An application for Large Models and Parameter Studies

pid = 0
np = n

pid = 1
np = n

pid = n-1
np = n

Each process is running an interpreter and can determine it's unique process number and the total number of processes in computation

Based on this script can do different things

```
# source in the model and analysis procedures
set pid [getPID]
set np [getNP]

# build model based on np and pid
source modelP.tcl
doModel {$pid $np}

# perform gravity analysis
system Mumps
constraints Transformation
numberer Parallel
test NormDispIncr 1.0e-12 10 3
algorithm Newton
integrator LoadControl 0.1

analysis Static

set ok [analyze 10]
return $ok
```

New Commands added to OpenSeesMP:

- A Number of new commands have been added:

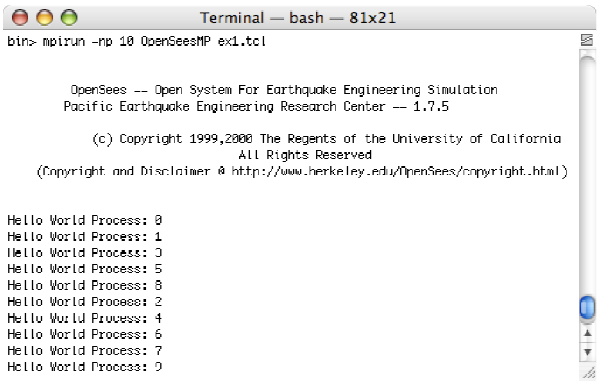
1. getNP returns number of processes in computation.
2. getPID returns unique pocess id {0,1, .. NP-1}
3. send -pid pid? data pid = { 0, 1, .., NP-1}
4. recv -pid pid? variableName pid = {0,1 .., NP-1, ANY}
5. barrier
6. domainChange

- These commands have been added to ALL interpreters (OpenSees, OpenSeesSP, and OpenSeesMP)

Example

ex1.tcl

```
set pid [getPID]
set np [getNP]

puts "Hello World Process: $pid"
```



Another Example

ex2.tcl

```
set pid [getPID]
set np [getNP]
if {$pid == 0 } {
   puts ”Random:"
   for {set i 1 } {$i < $np} {incr i 1} {
      recv -pid ANY msg
      puts "$msg"
   }
} else {
   send -pid 0 "Hello from $pid"
}
barrier
if {$pid == 0 } {
   puts "\nOrdered:"
   for {set i 1 } {$i < $np} {incr i 1} {
      recv -pid $i msg
      puts "$msg"
   }
} else {
   send -pid 0 "Hello from $pid"
}
```

```
Terminal — bash — 80x32
bin> mpirun -np 10 OpenSeesMP ex2.tcl

        OpenSees -- Open System For Earthquake Engineering Si
        Pacific Earthquake Engineering Research Center -- 1.7.

            (c) Copyright 1999,2000 The Regents of the Univers
                                All Rights Reserved
    (Copyright and Disclaimer @ http://www.berkeley.edu/OpenSe


Random:
Hello from 1
Hello from 3
Hello from 5
Hello from 6
Hello from 8
Hello from 2
Hello from 4
Hello from 7
Hello from 9
Ordered:
Hello from 1
Hello from 2
Hello from 3
Hello from 4
Hello from 5
Hello from 6
Hello from 7
Hello from 8
Hello from 9
```

Steel Building Study

7200 records
2 min a record
240 hours or 10 days
Ran on 2000 processors
on teragrid in less than 15 min.

```
set pid [getPID]
set np [getNP]
set recordsFileID [open "peerRecords.txt" r]
set count 0;

foreach gMotion [split [read $recordsFileID] ¥n] {
 if {[expr $count % $np] == $pid} {

 source model.tcl
 source analysis.tcl

 set ok [doGravity]

 loadConst -time 0.0

 set gMotionList [split $gMotion "/"]
 set gMotionDir [lindex $gMotionList end-1]
 set gMotionNameInclAT2 [lindex $gMotionList end]
 set gMotionName [string range $gMotionNameInclAT2 0 end-4 ]

 set Gaccel "PeerDatabase $gMotionDir $gMotionName -accel 384.4 -dT dT -nPts nPts"
 pattern UniformExcitation 2 1 -accel $Gaccel

 recorder EnvelopeNode -file $gMotionDir$gMotionName.out -node 3 4 -dof 1 2 3 disp

 doDynamic [expr $dT*$nPts] $dT

  wipe
 }

 incr count 1;
}
```

Concrete Building Study

```
set pid [getPID]
set np [getNP]
set count 0;
source parameters.tcl
source ReadSMDFileNewFormat.tcl;
foreach GMfile $iGMFile {
   foreach Factor1248 $iFactor1248 {

      if {[expr $count % $np] == $pid} {

         set inFile $GMdir/$GMfile.AT2
         set outFile $GMdir/$GMfile.g3;
         ReadSMDFileNewFormat $inFile $outFile dt npts;

         wipe
         source GravityAnalysisScript.tcl

         loadConst -time 0.0;
         wipeAnalysis

         source  EQ_Recorder.tcl
         source EQAnalysisScript.tcl

        if {$ok == 0} {
            puts "Process $pid $GMfile x $Factor1248 FINISHED OK modelTime [getTime]]"
         } else {
            puts "Process $pid $GMfile x $Factor1248 FINISHED FAIL modeTime [getTime] desiredTime $TmaxAnalysis]"
         }
         incr count 1
      }
   }
}
```

113 records, 4 intensities
3 hour a record
1356 hours or 56.5 days
Ran on 452 processors
on teragrid in less than 5 hours.

Parallel Programming Models

•A **Programming Model** provides an abstract conceptual view of the structure and operation of a computing system. A computer language and system libraries provide the programmer with this programming model.

• For parallel programming there are currently **2 dominant** models:

1. **Shared Memory**: The running program is viewed as a collection of processes (threads) each sharing it's virtual address space with the other processes. Access to shared data must be synchronized between processes to avoid race conditions.
2. **Message Passing**:The running program is viewed as a collection of independent communicating processes. Each process executes in it's own address space, has it's own unique identifier and is able to communicate with the other processes.

Modified Commands

- Some existing commands have been modified to allow analysis of large models in parallel:
  1. numberer

     numberer ParallelPlain

     numberer ParallelRCM
  2. system

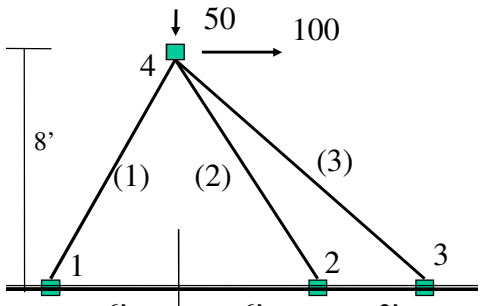
     system Mumps <-ICNTL14 %?>
  3. integrator

     integrator ParallelDisplacementControl node? Dof? dU?
- Use these ONLY IF PARALLEL MODEL

Example Parallel Model:

ex4.tcl

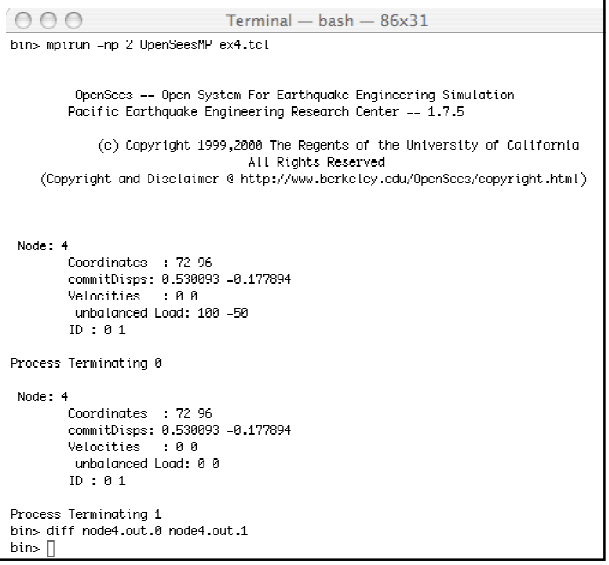
```
set pid [getPID]
set np [getNP]
if {$np != 2}  exit

model BasicBuilder -ndm 2 -ndf 2
uniaxialMaterial Elastic 1 3000
if {$pid == 0} {
   node 1   0.0  0.0
   node 4  72.0 96.0
   fix 1 1 1
   element truss 1 1 4 10.0 1
   pattern Plain 1 "Linear" {
      load 4 100 -50
   }
} else {
   node 2 144.0  0.0
   node 3 168.0  0.0
   node 4  72.0 96.0
   fix 2 1 1
   fix 3 1 1
   element truss 2 2 4 5.0 1
   element truss 3 3 4 5.0 1
}
```



	E	A
1	3000	10
2	3000	5
3	3000	5

Example Parallel Analysis:

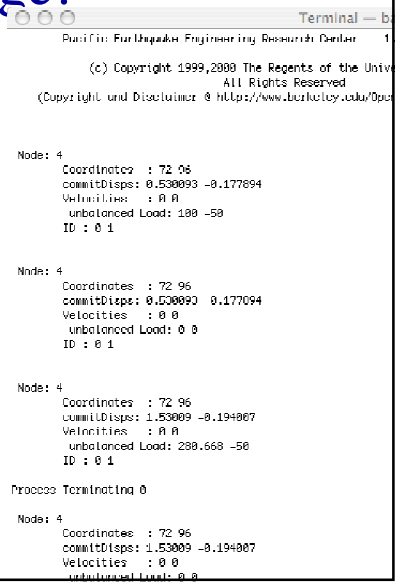
```
#create the recorder
recorder Node -file node4.out.$pid -node 4 -dof 1 2 disp

#create the analysis
constraints Transformation
numberer ParallelPlain
system Mumps
test NormDispIncr 1.0e-6 6 2
algorithm Newton
integrator LoadControl 0.1
analysis Static

#perform the analysis
analyze 10

# print to screen node 4
print node 4
```



Parallel Displacement Control and domainChange!

ex5.tcl

```
source ex4.tcl

loadConst - time 0.0

if {$pid == 0} {
   pattern Plain 2 "Linear" {
      load 4 1 0
   }
}

domainChange

integrator ParallelDisplacementControl 4 1 0.1
analyze 10
```



Things to Watch For

1. Deadlock (program hangs)
   - send/recv messages
   - Opening files for writing & not closing them
2. Race Conditions (different results every time run problem)
   - parallel file system.
3. Load Imbalance
   - poor initial task assignment.

Deadlock Example

ex3.tcl

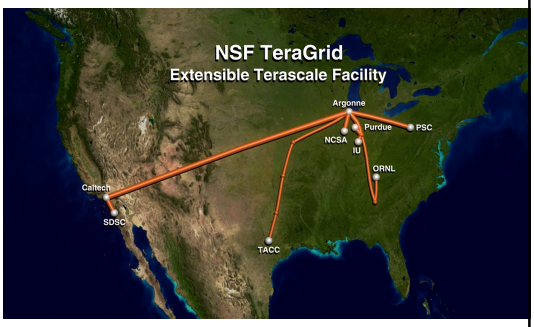
```
set pid [getPID]
set np [getNP]
if {$pid == 0 } {
   puts "Random:"
   for {set i 1 } {$i < $np} {incr i 1} {
      recv -pid ANY msg
      puts "$msg"
   }
} else {
   send -pid 0 "Hello from $pid"
}
#barrier
if {$pid == 0 } {
   puts "\nOrdered:"
   for {set i 1 } {$i < $np} {incr i 1} {
      recv -pid $i msg
      puts "$msg"
   }
} else {
   send -pid 0 "Hello from $pid"
}
```

Terminal — bash — 84x36

```
bin> mpirun -np 10 OpenSeesMP ex2.tcl


        OpenSees -- Open System For Earthquake Engineering Simulation
       Pacific Earthquake Engineering Research Center -- 1.7.5

           (c) Copyright 1999,2000 The Regents of the University of Cali
                                All Rights Reserved
    (Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyrigh



Random:
Hello from 2
Hello from 5
Hello from 2
Hello from 5
Hello from 3
Hello from 4
Hello from 6
Hello from 3
Hello from 4
Ordered:
Hello from 1
^Cmpirun: killing job...

^C--------------------------------------------------------------------------
WARNING: mpirun is in the process of killing a job, but has detected an
interruption (probably control-C).

It is dangerous to interrupt mpirun while it is killing a job (proper
termination may not be guaranteed).  Hit control-C again within 1
second if you really want to kill mpirun immediately.
```

What isTeragrid?

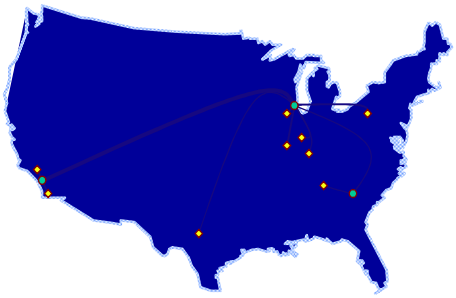
- An NSF sponsored computational science facility supported through a partnership of 13 institutions.
- Provides over 20 high performance computational systems, specialized visualization services and massive storage archives to national community.



- Resources growing exponentially. At end of 2006 the fastest HPC had 2000 processors and 10-15 Teraflops. At end 2007 50,000 processors and 400 teraflops. By 2010 Petascale flops.

The Teragrid Strategy

- Create a ***unified*** national HPC infrastructure that is both ***heterogeneous*** and ***extensible***
  - Single user support resources.
  - Single authentication point
  - Common software functionality
  - Common job management infrastructure
  - Globally-accessible data storage



OpenSees on Teragrid

- We are in the process of building OpenSees, OpenSeesSP and OpenSeesMP on all Teragrid machines.
- For now, the executables will be located in my home/bin directory.
- The location of this and the machines that the applications are installed on can be seen on the parallel web page.
- Feel free to help out!

Access

- Secure Shell -
  - A Protocol that allows data to be transferred securely over the network using public key encryption
  - Common programs
    - ssh for login
    - scp or sftp for file transfer
- Teragrid portal.
- NEES portal under development at NEESit.
  - **SimPortal**

The SimPort Portlet

Developed at the Center for Advanced Vehicular Systems at Mississipi State University by the Cooperative Computing Group under direction of Tomasz Haupt.

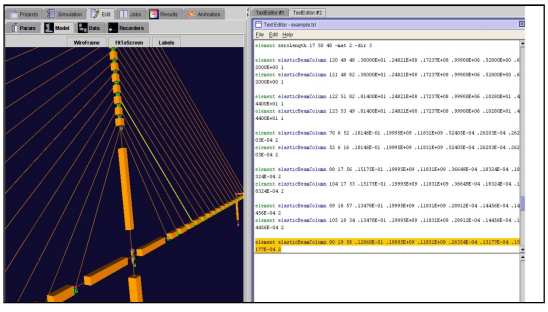
If you have any questions about it contact:

Tomasz Haupt(haupt@cavs.msstate.edu)

Anand Kalyanasundaram(anand@cavs.msstate.edu)

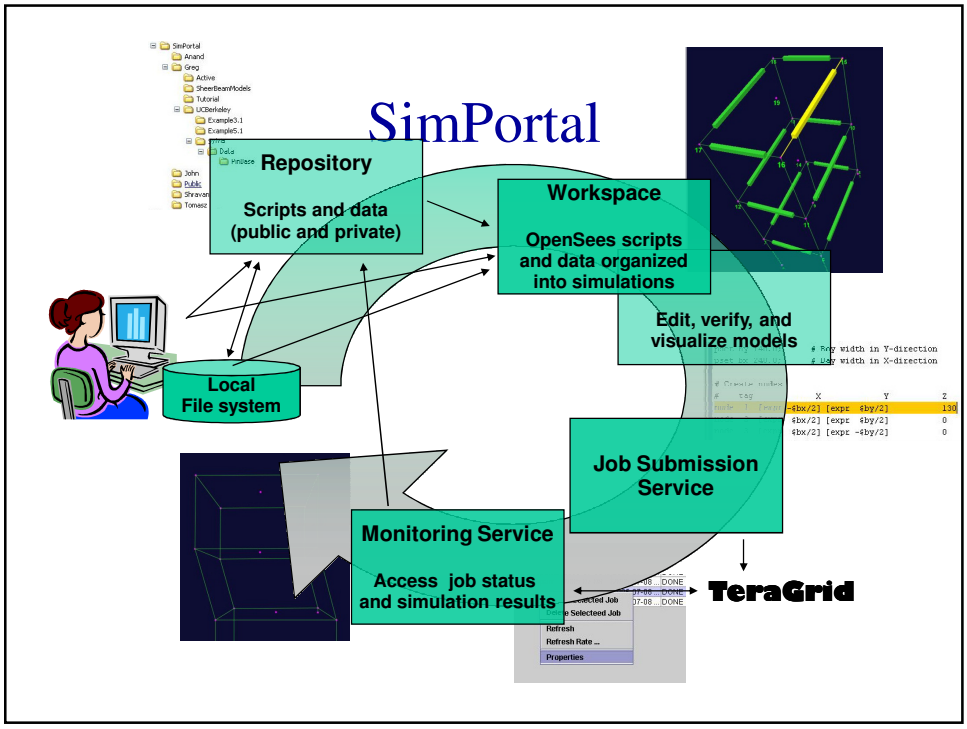
SimPortal

The **SimPortal** provides a graphical computational simulation component for NEESit. This is currently accomplished by providing a Grid Portal for OpenSees.



SimPortal is currently under development at Mississippi State University.

- Compose a simulation by acquiring all necessary OpenSees scripts:
  - the NEESit repository
  - the user desktop
- Edit and verify the correctness of the scripts
- For very complex simulations, the scripts are expected to be developed off-line and uploaded to the portal to be executed on high-performance platforms provided by NEESit.
- Monitor the progress of the simulations, preview the results, and download them for a detailed analysis.
- Share the scripts and simulation results with other users


SimPortal
Repository
Scripts and data
(public and private)
Workspace
OpenSees scripts
and data organized
into simulations
Edit, verify, and
visualize models
Local
File system
Job Submission
Service
Monitoring Service
Access job status
and simulation results
TeraGrid