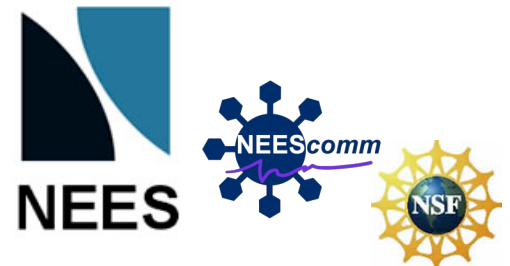
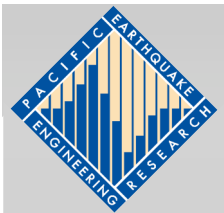


<http://opensees.berkeley.edu/webinars/parallel/material.zip>

# Parallel Processing & Grid Computing With OpenSees

**Frank McKenna**  
UC Berkeley



# The Motivation ...

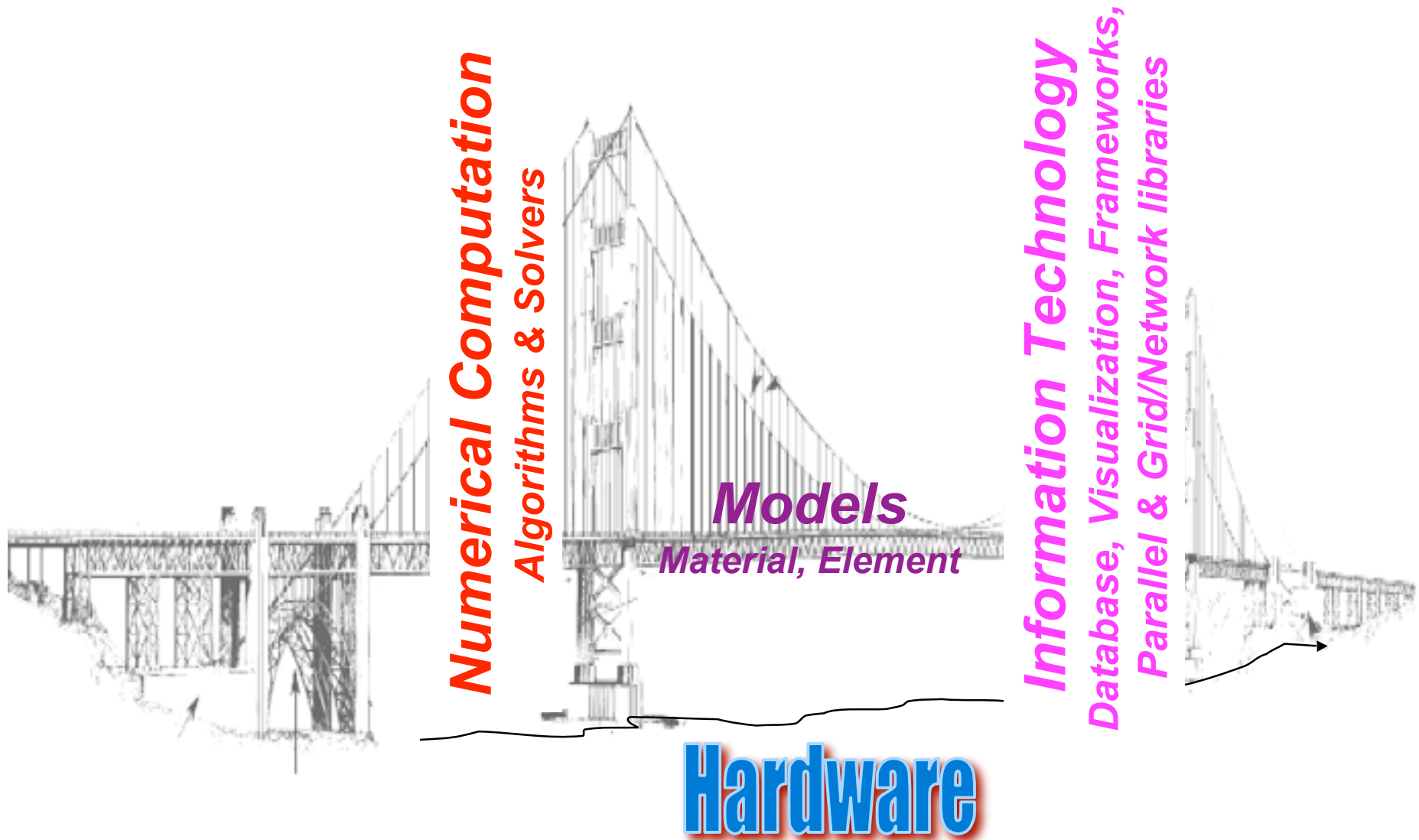
**If your desktop represented the computational resources available to you today to perform large simulations**

**then only using your desktop to perform the simulation is like using your fingers to perform the computations required.**

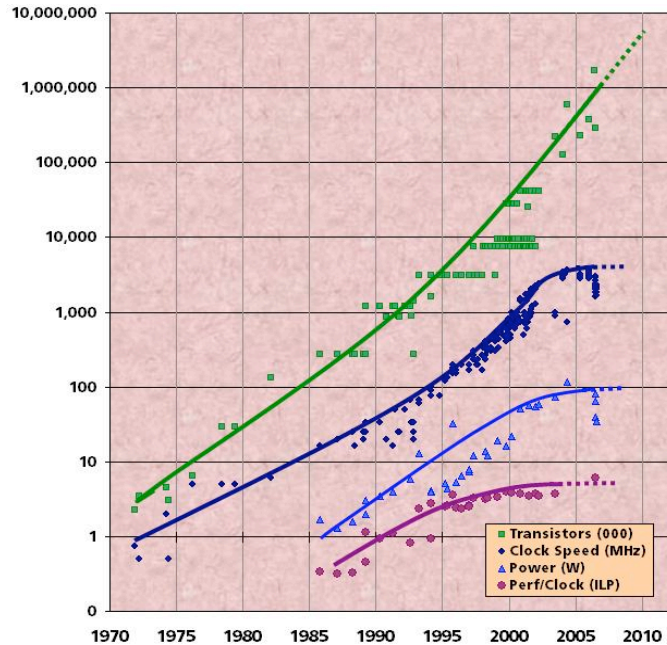
# Outline

- Technology is Changing
- OpenSees & Parallel Processing
- Parallel/Grid Computing on NEEShub

# Building Blocks for Simulation



# Hardware has Changed



CHIP/Socket

## Intel Processor Speed

XeonE7Server	72Gflop
i7Desktop	55GFlop
i7Mobile	30GFlop
i5Desktop	40GFlop
i5Mobile	22GFlop
Core2 Quad	48GFlop
Core2 Duo	25GFlop

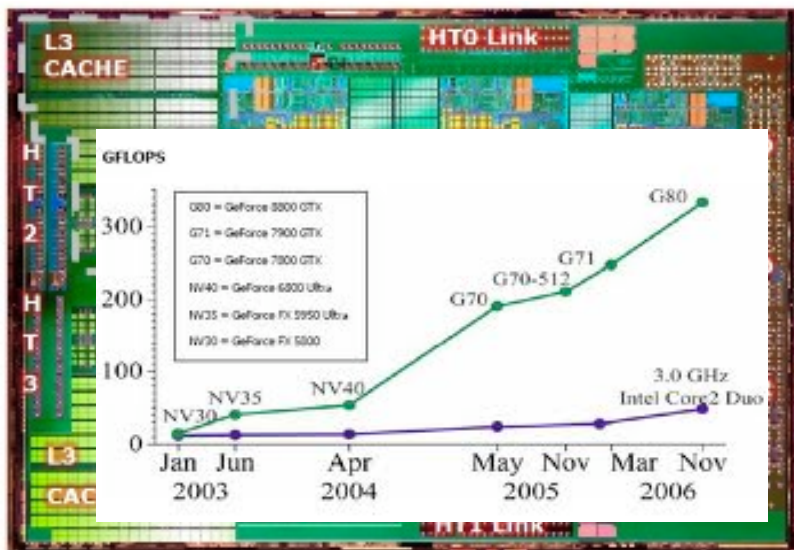
**Game consoles (Wii, Xbox Playstation) have more raw numerical processing power than your desktop!**

## Game Console Speed



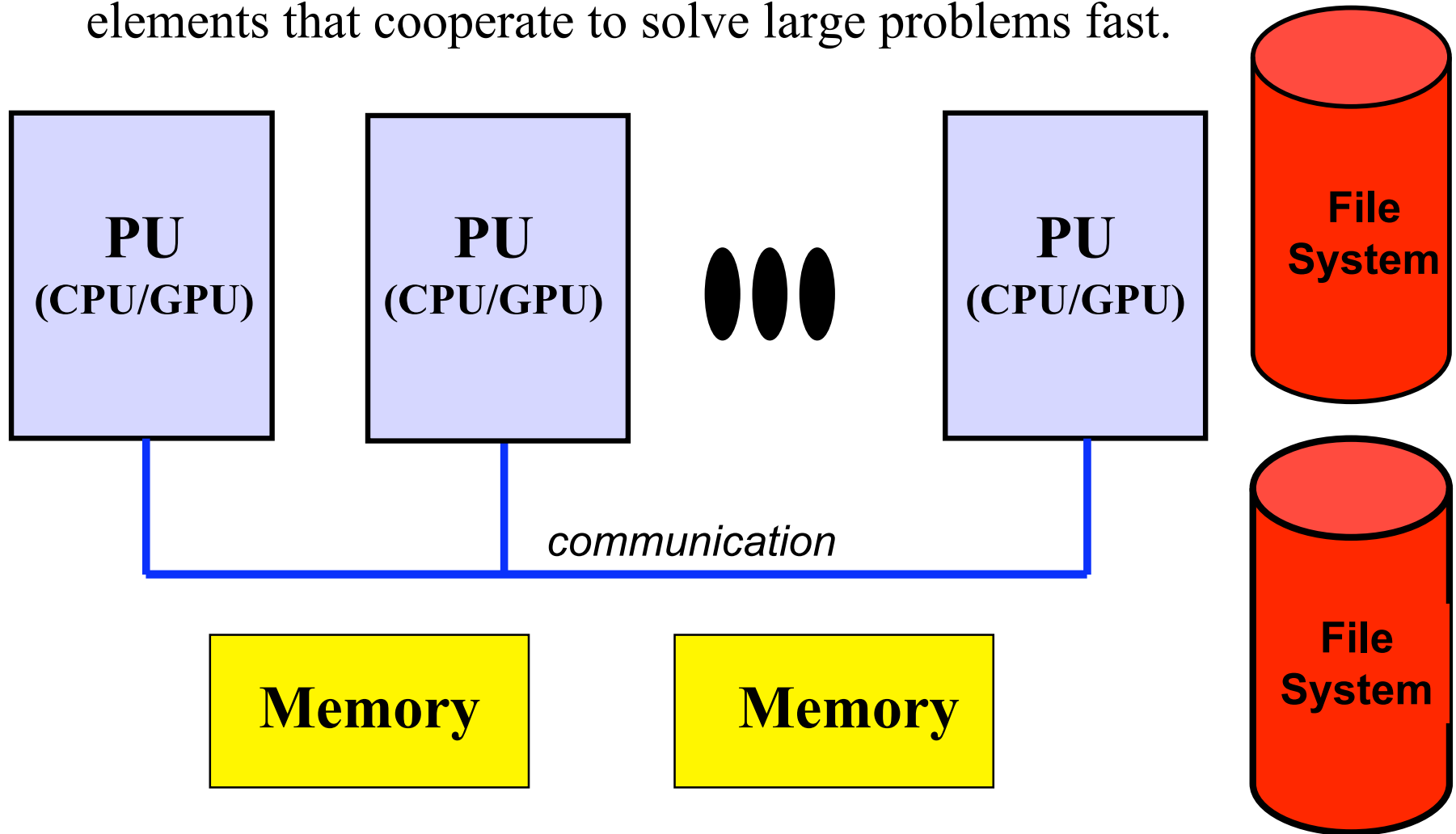
GPU

Nintendo Wii	61Gflop
Xbox360	355Gflop
Sony PS3	2018Gflop



# What is a Parallel Computer?

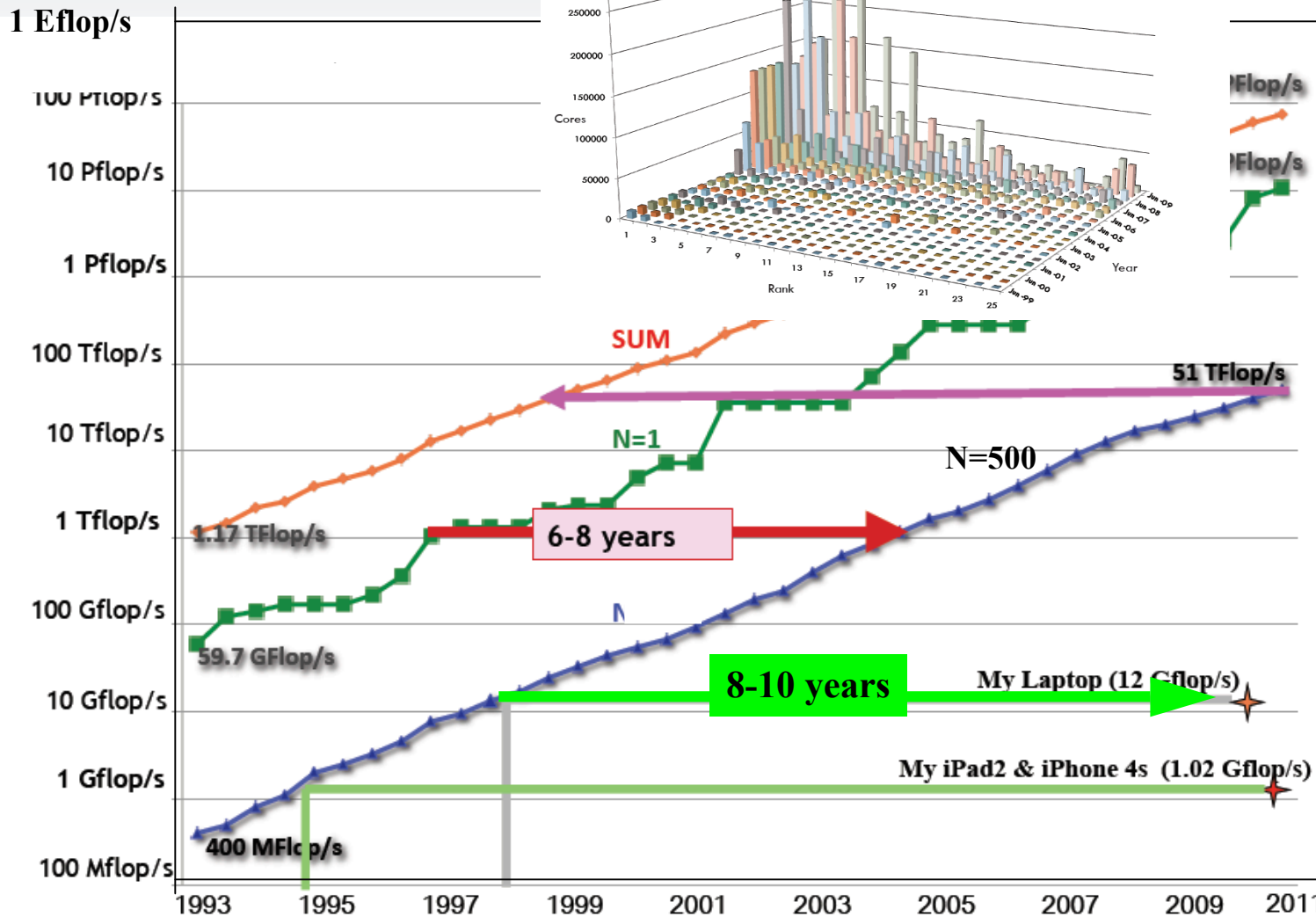
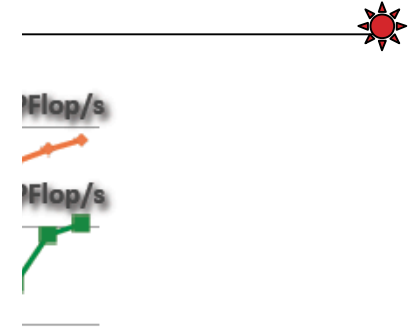
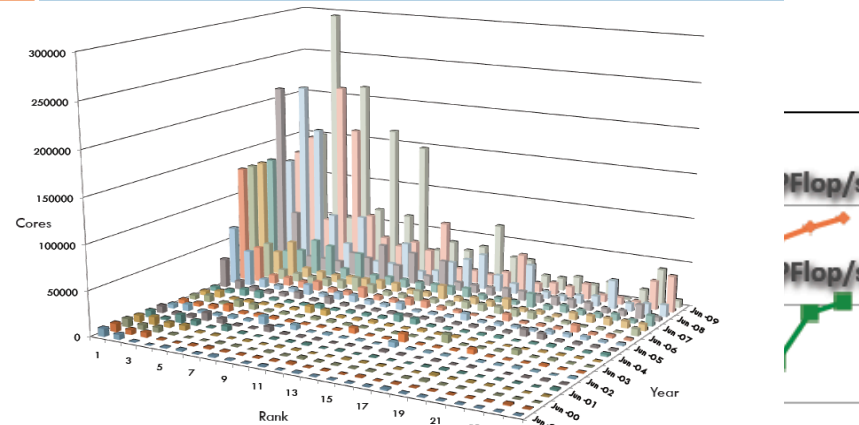
- A *parallel computer* is a collection of processing elements that cooperate to solve large problems fast.





# Performance Projection

Cores in the Top25 Over Last 10 Years



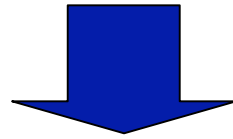
2019

Source: Jack Dongarra, 2011

# BEFORE YOU GET ALL EXCITED

## Speedup & Amdahl's Law

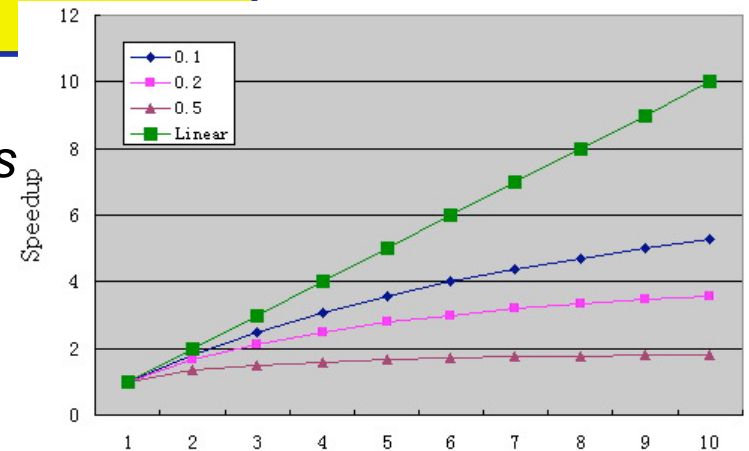
$$speedup_{PC}(p) = \frac{Time(1)}{Time(p)}$$



$$Speedup_{PC} = \frac{T_1}{\alpha T_1 + \frac{(1-\alpha)T_1}{n}} \rightarrow \frac{1}{\alpha} \text{ as } n \rightarrow \infty$$

Portion of sequential

# of processors





# Improving Real Performance

**Peak Performance grows exponentially,  
a la Moore's Law**

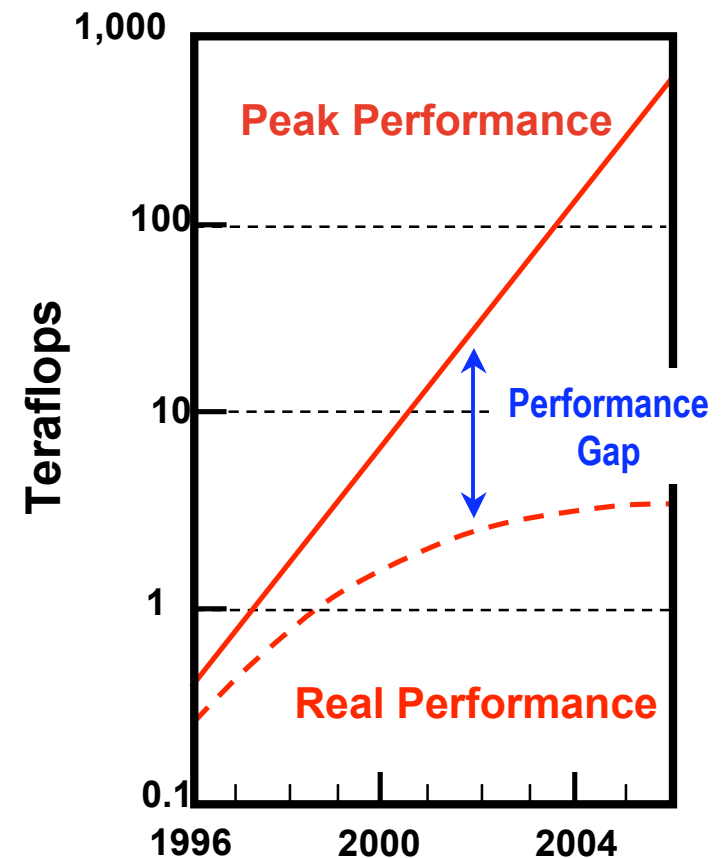
- In 1990's, peak performance increased 100x;  
in 2000's, it will increase 1000x

**But efficiency (the performance relative to  
the hardware peak) has declined**

- was 40-50% on the vector supercomputers  
of 1990s
- now as little as 5-10% on parallel  
supercomputers of today

**Close the gap through ...**

- Mathematical methods and algorithms that  
achieve high performance on a single  
processor and scale to thousands of  
processors
- More efficient programming models and tools  
for massively parallel supercomputers



Source: Jim Demmell, CS267  
Course Notes

# Amdahl's Law Ignores Data Movement

## Data Is Becoming an Even Greater Bottleneck to Speedup

- Data Movement is expensive
- Flops are cheap



- Iterative Solvers
- Mixed Precision Arithmetic
- Duplicate Calculations

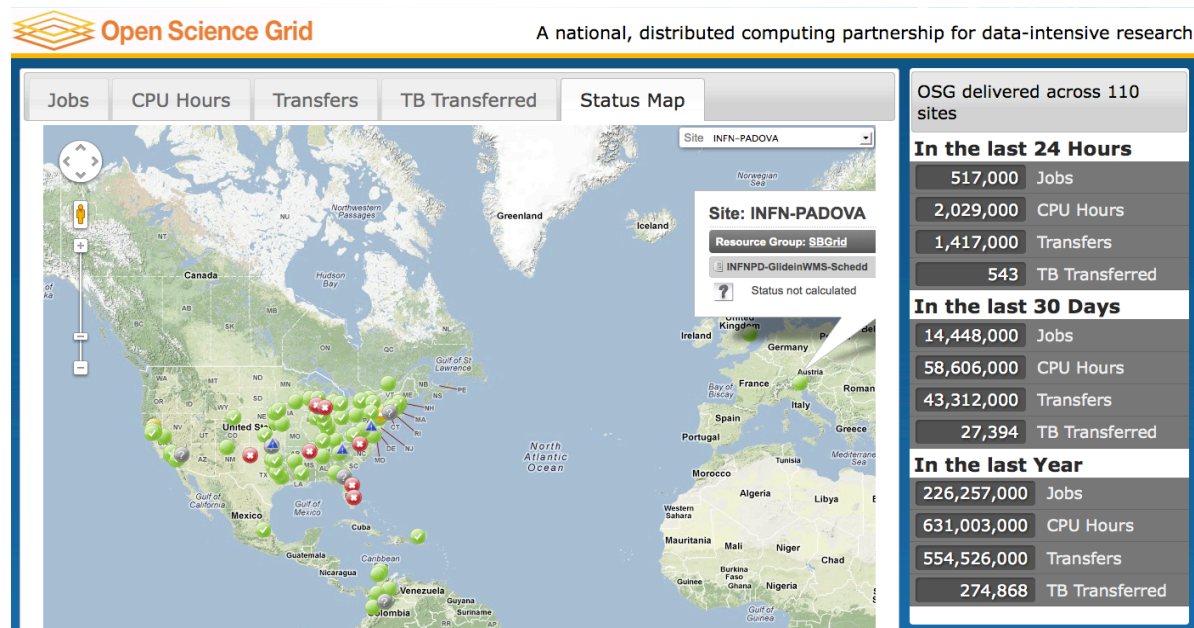
# It's Not Just the High Performance Parallel Computers We Need to Concentrate On

Stand Alone Parallel Machine (i.e. your desktop/device)

Grid Computing

# Grid Computing:

- Most distributed form of parallel computing
- Computers communicating over the internet to solve a given problem
- Low bandwidth and extremely high latency, typically only used for embarrassingly parallel problems, i.e. parameter studies.



# OpenSees in the clouds using Open Science Grid

*André R. Barbosa, Joel P. Conte, and José I. Restrepo, UCSD*

## ❑ Motivation

- Perform parametric studies that involve large-scale nonlinear models of structure or soil-structure systems with large number of parameters and OpenSees runs.

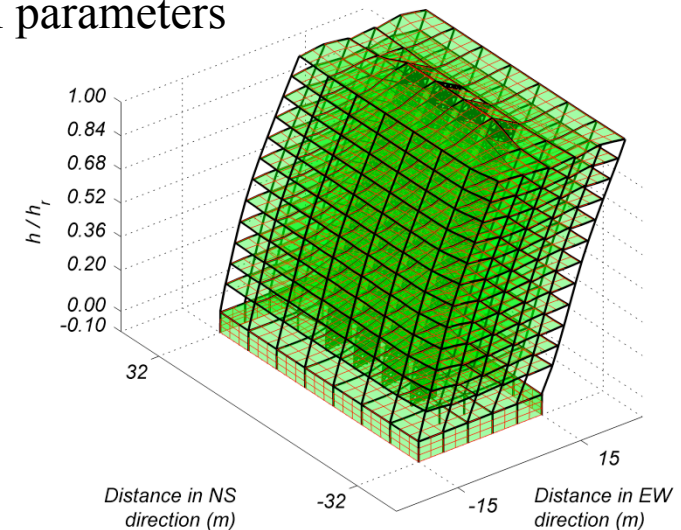


## ❑ Application example

- Nonlinear time-history (NLTH) analyses of advanced nonlinear FE model of a building
- Probabilistic seismic demand hazard analysis making use of the “cloud method”
  - 90 bi-directional historical earthquake records (unscaled and scaled by a factor of two)
- Sensitivity of probabilistic seismic demand to FE model parameters

## ❑ Some numbers

Number of NLTH analyses per parameter set realization	180
Average duration of NLTH analysis	12 hours
Average size of output data	1.5 GB
Parameters considered	6
Perturbations considered	4
Estimated clock time (180x12x[(6x4x2)+1])	<b>106,800 hours (12.2 years)</b>
Estimated output data (180x1.5x[(6x4x2)+1])	<b>12 TB</b>

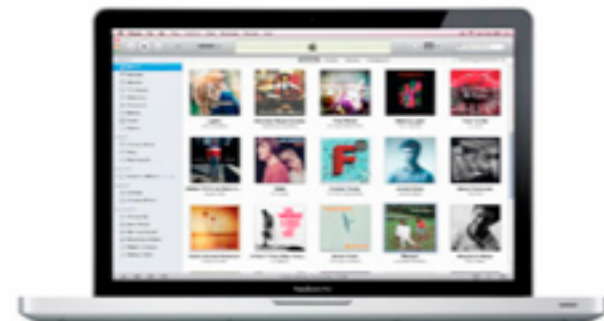


**30 days on  
OSG versus 12  
years on  
Desktop!**

# Cloud Computing (according to Steve Jobs WWDC 2011)

Some people think the cloud is just  
A hard drive in the sky!

Cloud computing is internet-based computing ,  
whereby shared **resources**, software, and  
information are provided to computers and other  
devices **on demand**, like the electricity grid. source:  
wikipedia



“..PC and Mac Demoted to a Device”

# Industry:



# Research: (Earthquake Engineering):

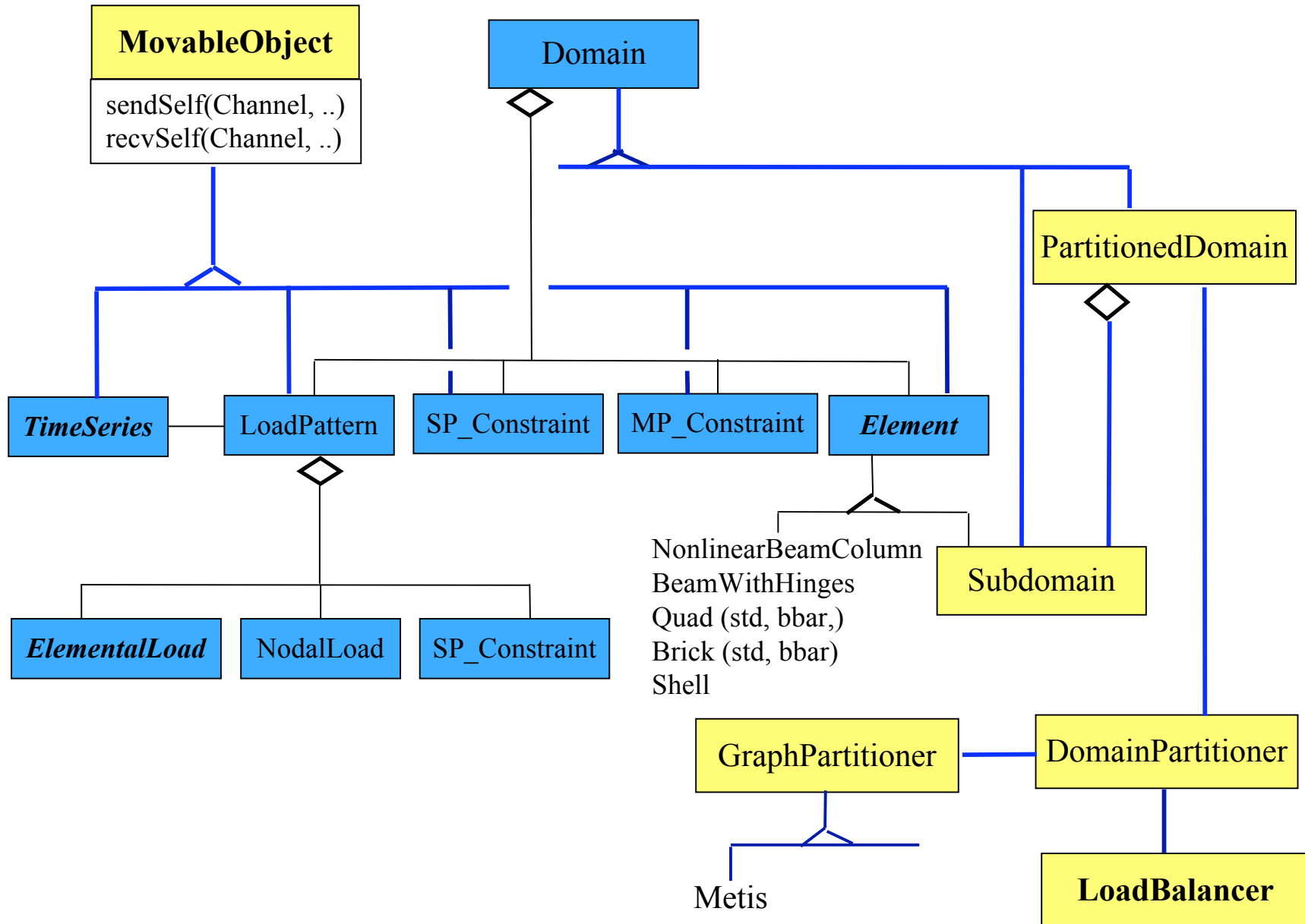


# What is OpenSees?

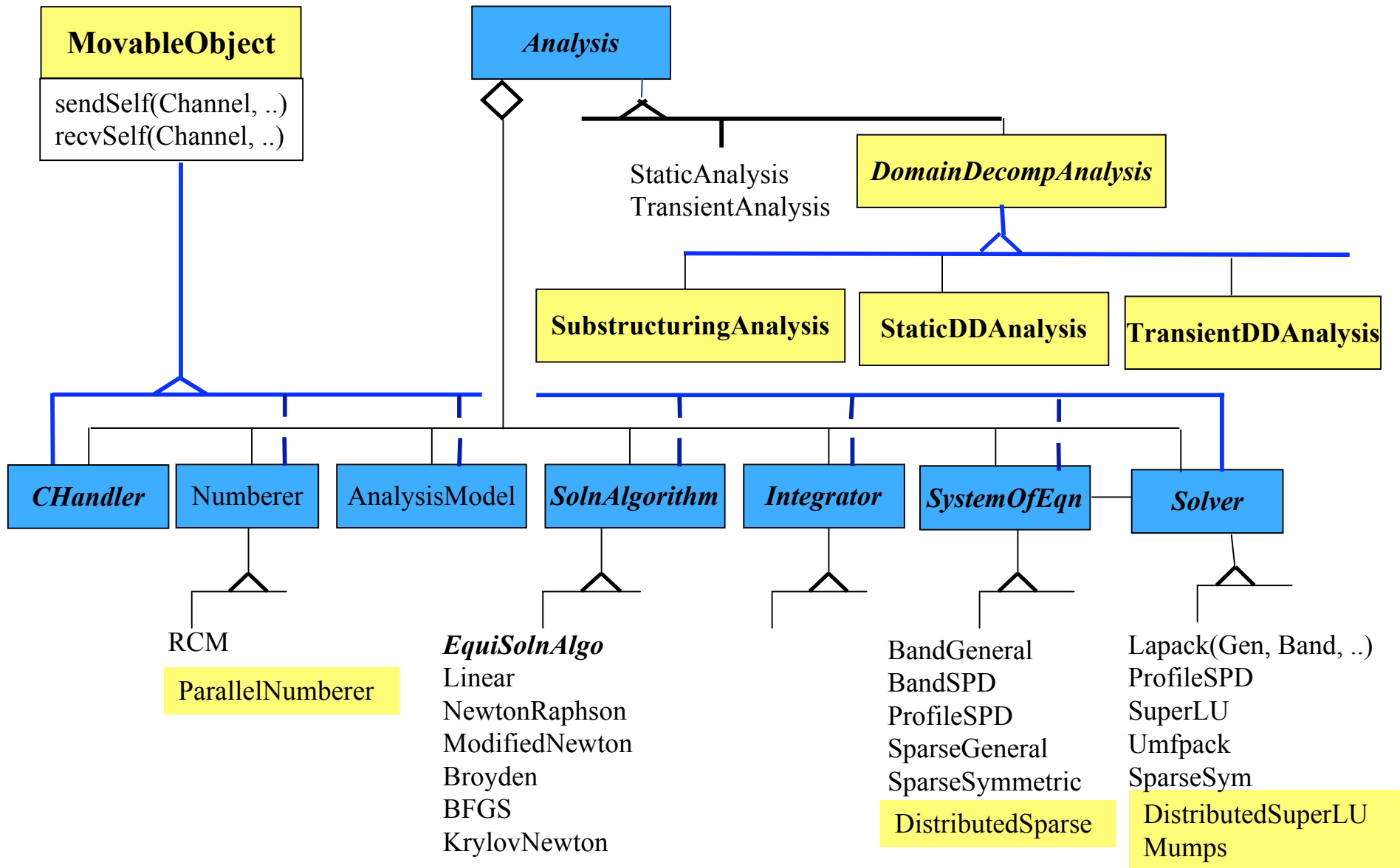
- OpenSees is an Open-Source Software Framework written in C++ for developing nonlinear Finite Element Applications for both sequential and **PARALLEL** environments.



# Domain Classes



# Analysis Classes



# The OpenSees Interpreters

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

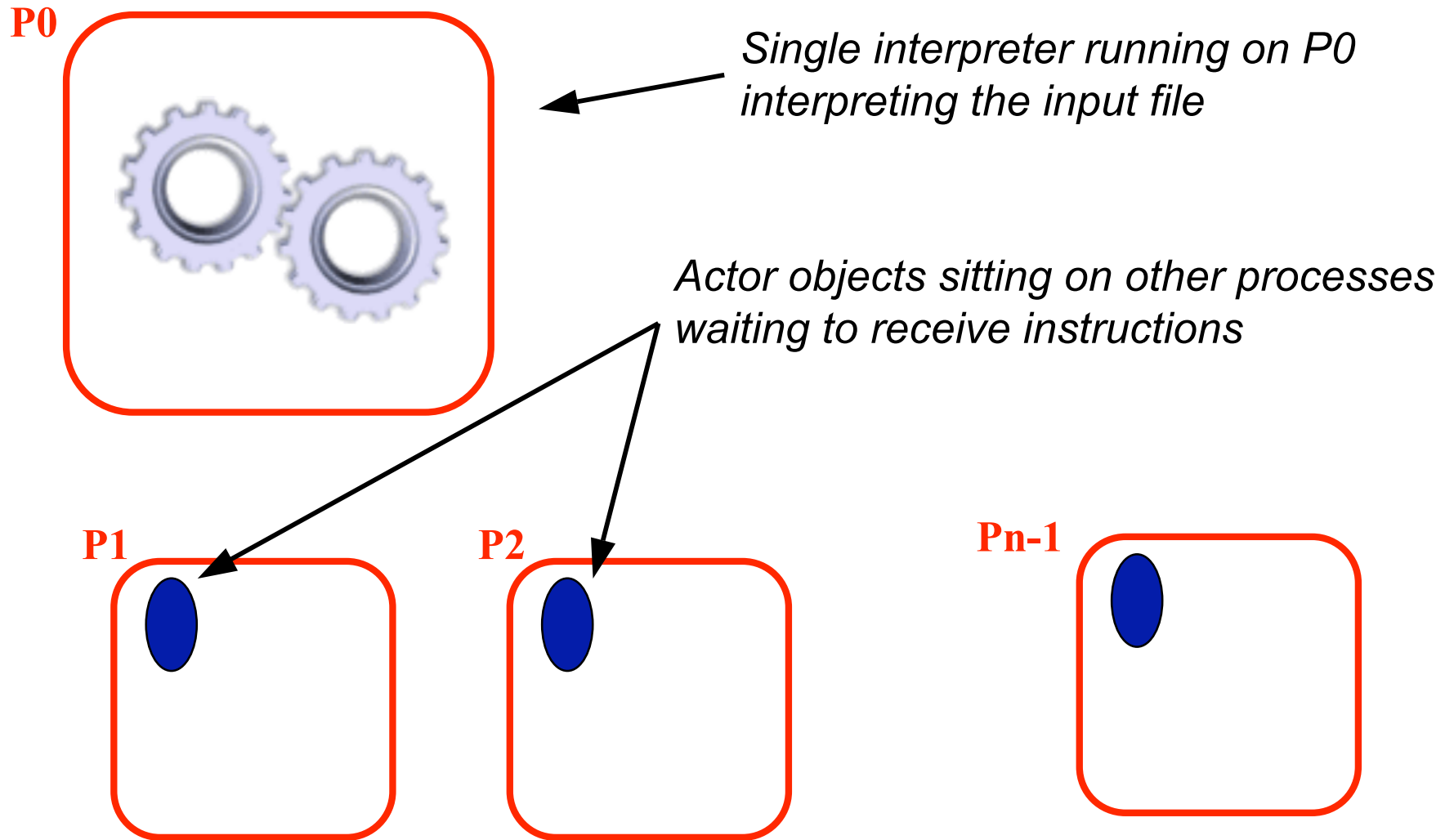
So What are OpenSeesSP.exe  
and OpenSeesMP.exe ?

# 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

Single interpreter running on P0  
Interpreting the input file

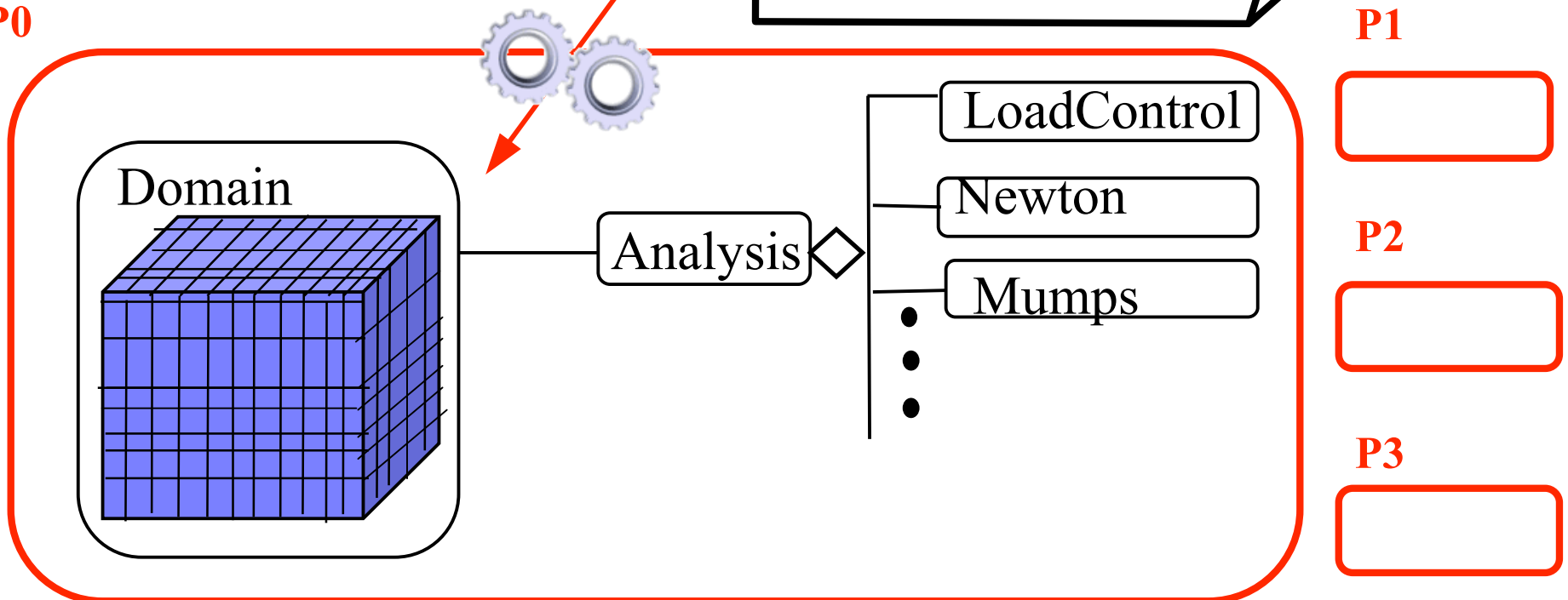
```
#build the model
source model.tcl
#build the analysis
system Mumps
constraints Transformation
numberer Plain
test NormDisplncr 1.0e-12 10 3
algorithm Newton
integrator LoadControl
analysis Static
```

P0

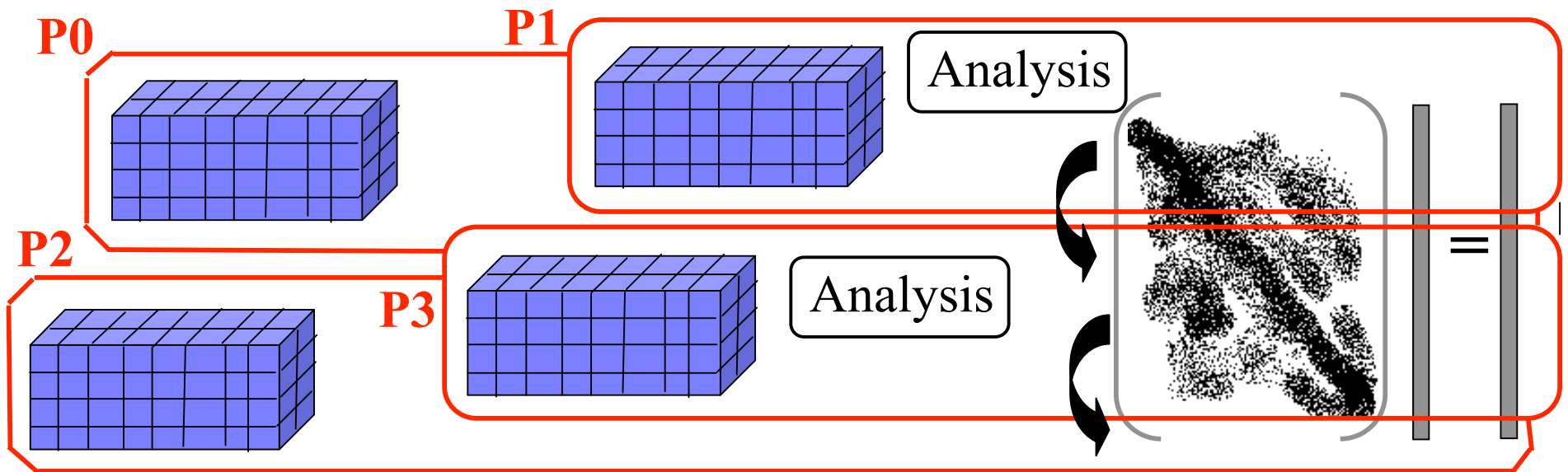
P1

P2

P3

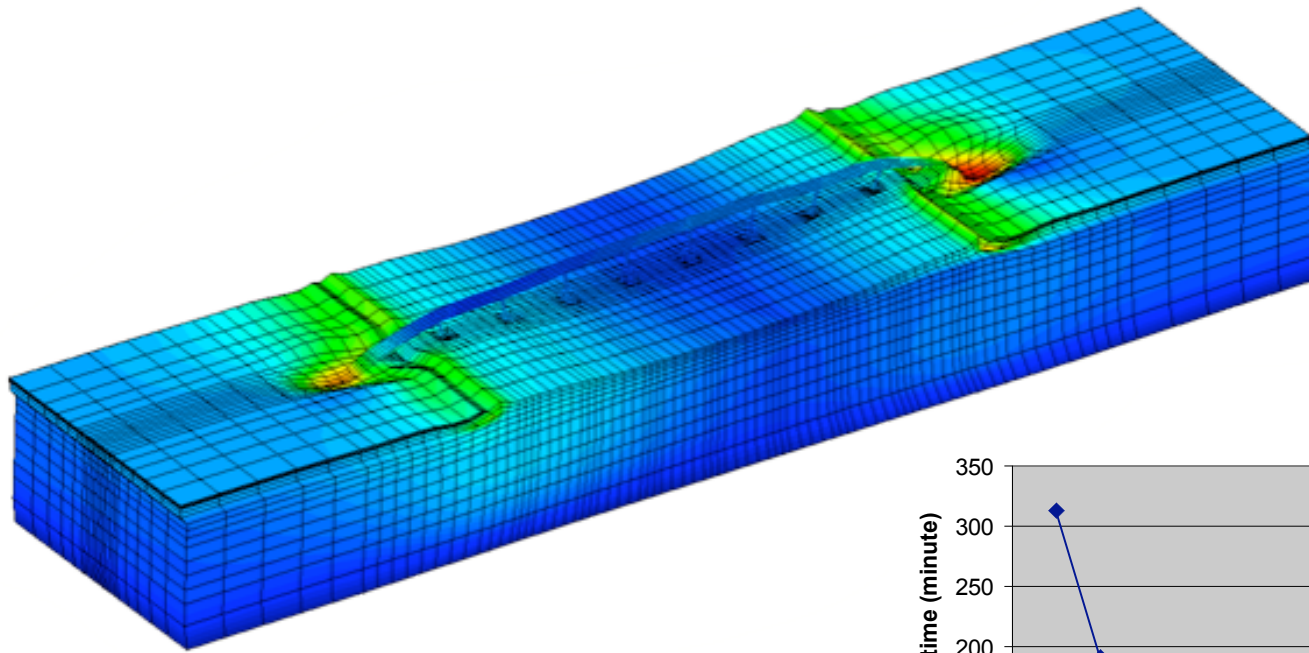


```
#build the model
source modelP.tcl
#build the analysis
system Mumps
constraints Transformation
numberer Plain
test NormDisplncr 1.0e-12 10 3
algorithm Newton
integrator LoadControl
analysis Static
analyze 10
```

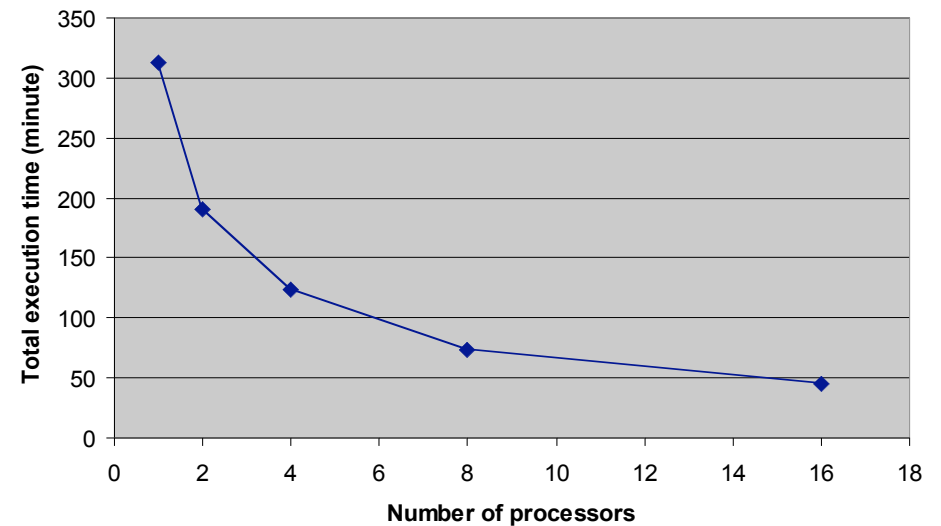




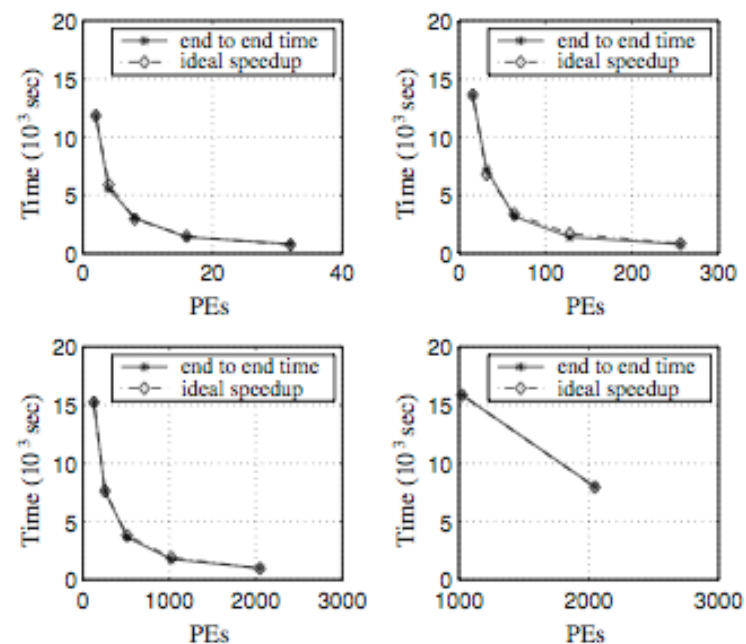
# Example Usage: Humboldt Bay Bridge Model



100,000+ DOF Model  
Implicit Integration  
Mumps Direct Solver

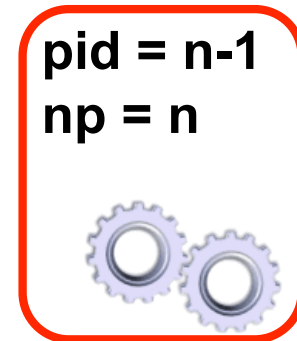
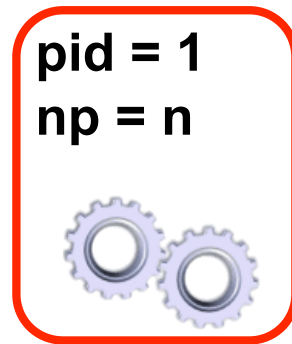
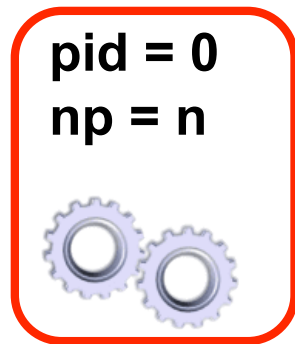


Run	el. size (m)	Elements	Nodes	DOFs
A	20	54,026	59,032	156,768
B	10	404,751	424,512	1,193,283
C	5	3,130,301	3,208,822	9,307,563
D	2.5	24,615,801	24,928,842	73,515,123



**Fig. 18** Fixed-size, scalability plot at SDSC's DataStar. Upper row is runs A (*left*) and B (*right*), lower row is runs C (*left*) and D (*right*) (Table 3)

# OpenSeesMP: An application for Large Models and Parameter Studies



***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 NormDisplncr 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

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  
Pacific Earthquake Engineering Research Center -- 1.7.

(c) Copyright 1999,2000 The Regents of the University of California  
All Rights Reserved

(Copyright and Disclaimer @ <http://www.berkeley.edu/OpenSees>)

```
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

```
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

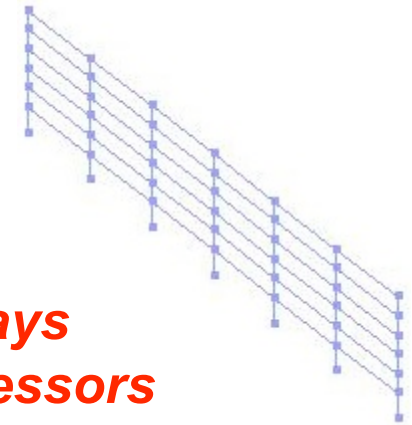
    recorder EnvelopeNode -file $gMotionDir$gMotionName.out -node 3 4 -dof 1 2 3 disp

    doDynamic [expr $dT*$nPts] $dT

    wipe
  }

  incr count 1;
}
```

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



# 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} {

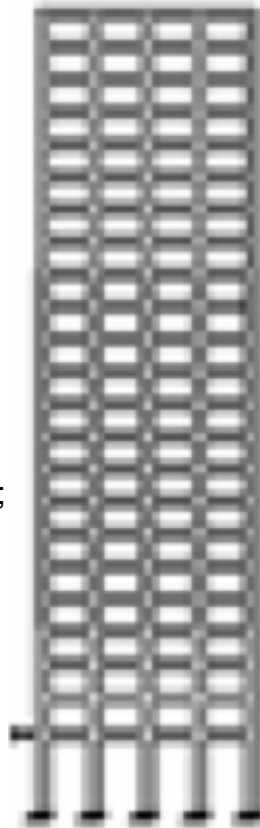
      set inFile $GMdir/$GMfile.AT2
      set outFile $GMdir/$GMfile.g3;
      ReadSMDFileNewFormat $inFile $outFile dt npts;

      wipe
      source GravityAnalysisScript.tcl

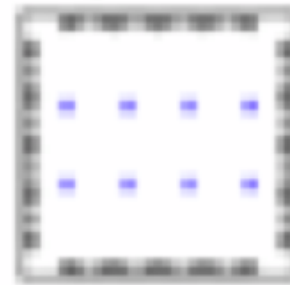
      loadConst -time 0.0;
      wipeAnalysis

      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, would have**  
**taken 1356 hours or 56.5 days**  
**Ran on 452 processors of a**  
**Teragrid in less than 5 hours.**



# 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:

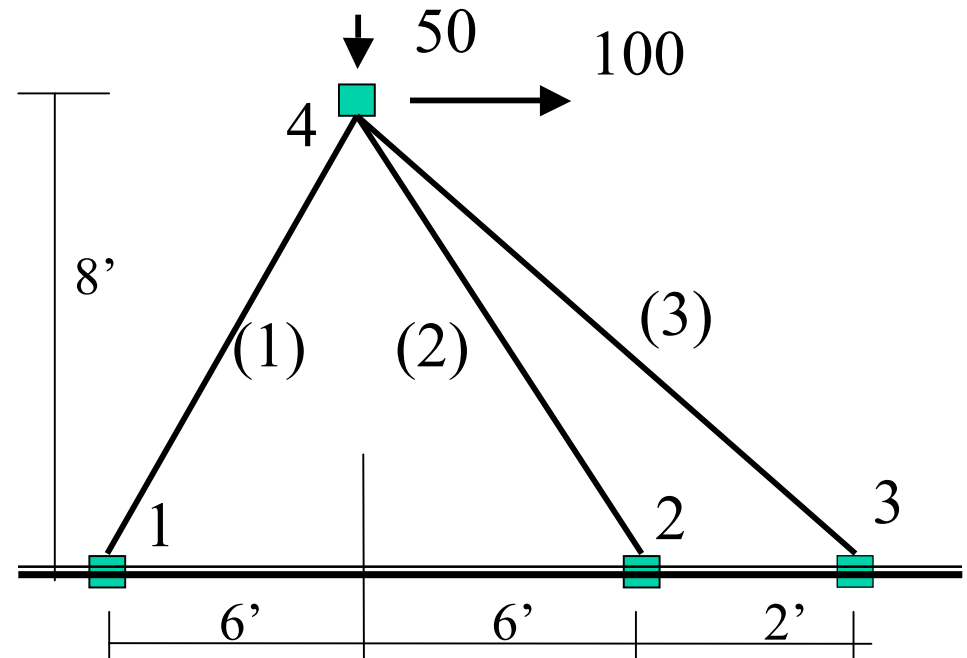
ex4.tcl

```

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

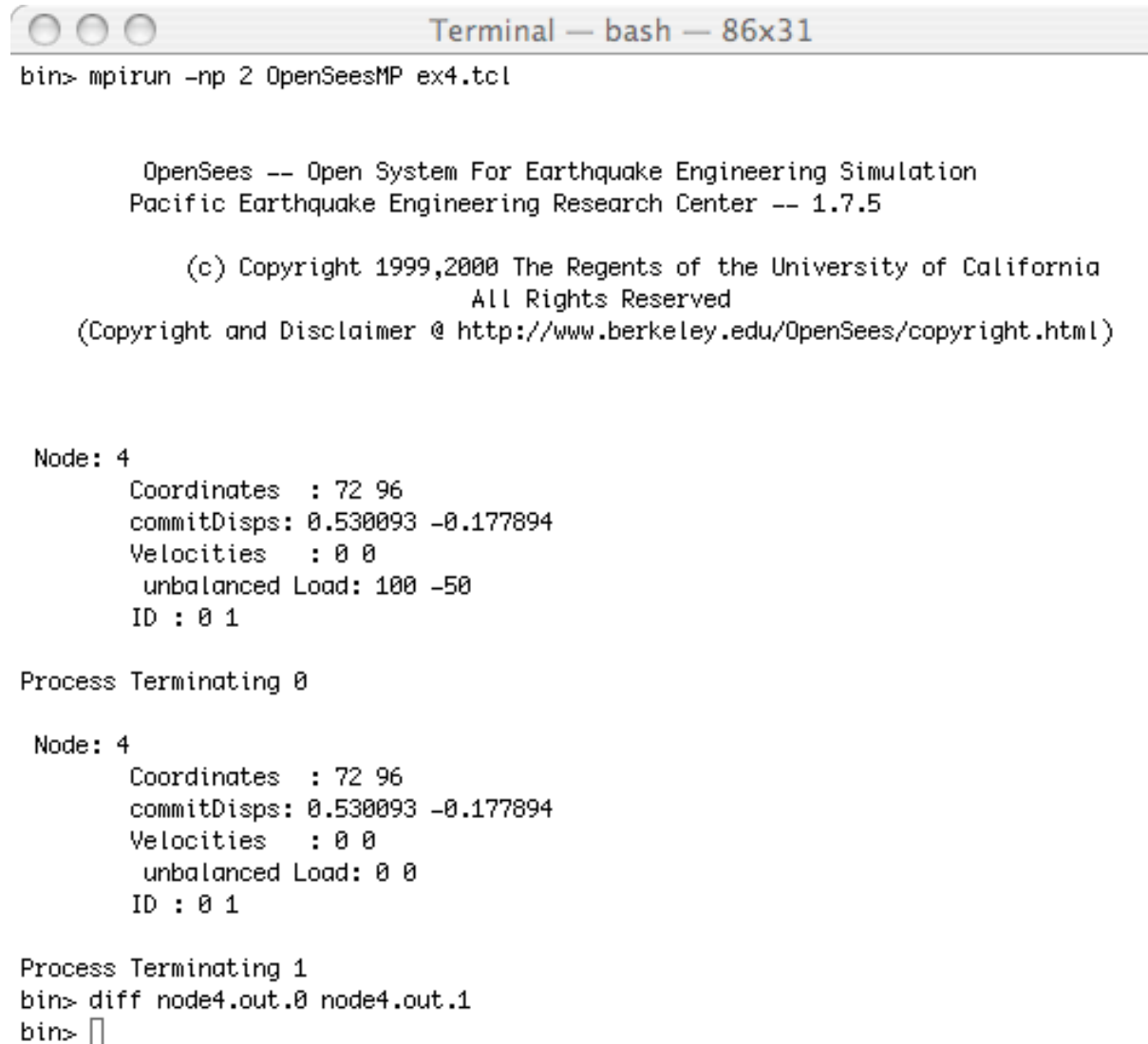
analysis Static

#perform the analysis

analyze 10

# print to screen node 4

print node 4



```
Terminal — bash — 86x31
bin> mpirun -np 2 OpenSeesMP ex4.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 California
All Rights Reserved
(Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)

Node: 4
Coordinates : 72 96
commitDisps: 0.530093 -0.177894
Velocities : 0 0
unbalanced Load: 100 -50
ID : 0 1

Process Terminating 0

Node: 4
Coordinates : 72 96
commitDisps: 0.530093 -0.177894
Velocities : 0 0
unbalanced Load: 0 0
ID : 0 1

Process Terminating 1
bin> diff node4.out.0 node4.out.1
bin> 
```

# Parallel Displacement Control and domainChange!

ex5.tcl

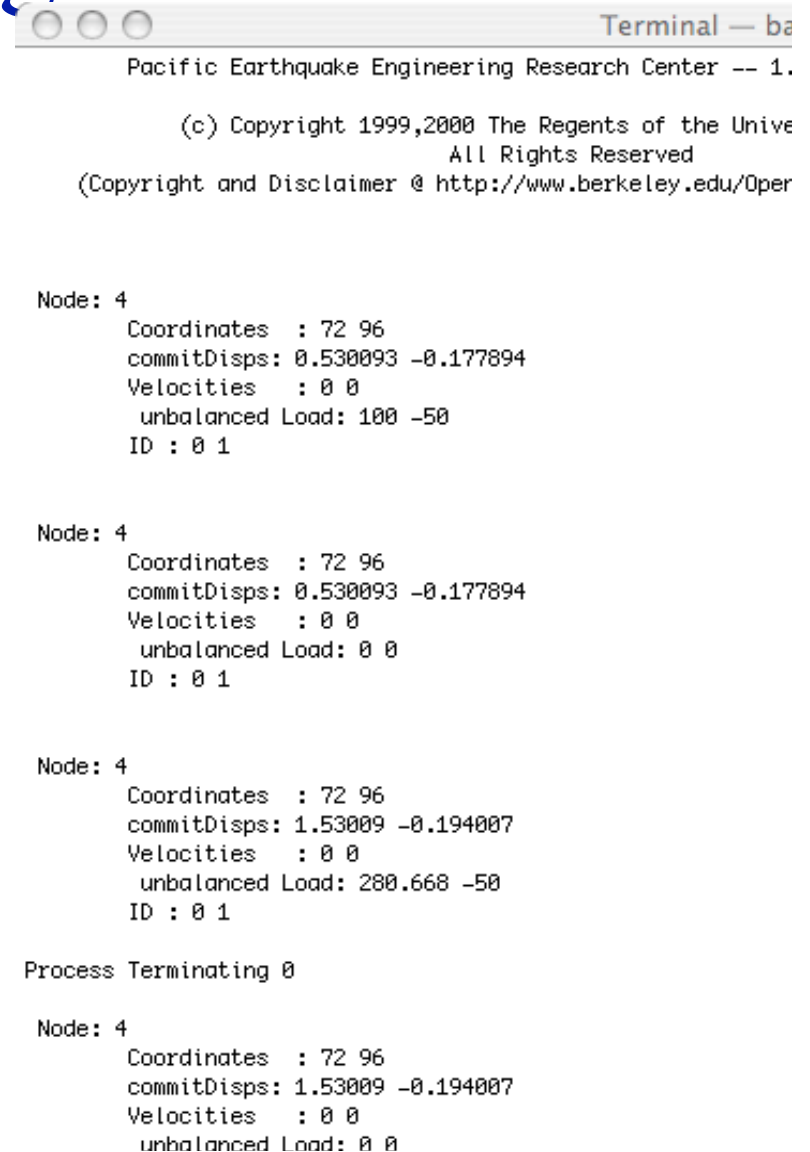
```
source ex4.tcl

loadConst - time 0.0

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

domainChange

integrator ParallelDisplacementControl 4 1 0.1
analyze 10
```



```
Terminal — ba
Pacific Earthquake Engineering Research Center -- 1.

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

Node: 4
Coordinates : 72 96
commitDisps: 0.530093 -0.177894
Velocities : 0 0
unbalanced Load: 100 -50
ID : 0 1

Node: 4
Coordinates : 72 96
commitDisps: 0.530093 -0.177894
Velocities : 0 0
unbalanced Load: 0 0
ID : 0 1

Node: 4
Coordinates : 72 96
commitDisps: 1.53009 -0.194007
Velocities : 0 0
unbalanced Load: 280.668 -50
ID : 0 1

Process Terminating 0

Node: 4
Coordinates : 72 96
commitDisps: 1.53009 -0.194007
Velocities : 0 0
unbalanced Load: 0 0
```

# 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.

# Watch out for Deadlock

- Match every send with a recv
- Watch the order

# 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.

# Race Conditions and the File System

- Remember all processes can be reading/writing to the same files. If same file is opened for reading and writing, e.g. using a global file system to handle shared variable (opening as r+ will do what you want).
- This can also happen if you are modifying the directory structure in your script.

# Watch out for Load Imbalance

- Load imbalance can greatly reduce the performance.
- Dynamic load balancing solutions can always be considered if performance is an issue.



# Parameter Study

*ex6.tcl*

```
set np [getNP]
set pid [getPID]
set count 0
```

```
source model.tcl
source analysis.tcl
```

```
set tStart [clock seconds]
```

```
set recordsFile [open motionList r]
set lines [split [read $recordsFile] \n]
foreach line $line {
```

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

```
        doModel
```

```
        doGravityAnalysis;
```

```
        loadConst -time 0.0
```

```
        set record [lindex $line 0]
```

```
        set npts [lindex $line 1]
```

```
        set dt [lindex $line 2]
```

```
        set accelSeries "Path -filePath $record -dt $dt -factor 386.4"
```

```
        pattern UniformExcitation 2 1 -accel $accelSeries
```

```
        set ok [doDynamicAnalysis $npts $dt]
```

```
        wipe
```

```
    }
```

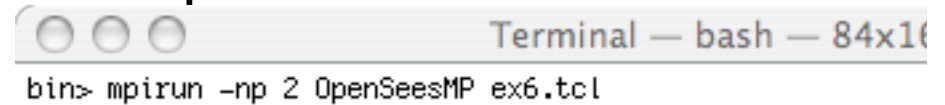
```
    incr count 1
```

```
}
```

```
set tFinish [clock seconds]
```

```
barrier
```

```
puts "Duration Process $pid [expr $tFinish - $tStart]"
```



```
Terminal — bash — 84x16
bin> mpirun -np 2 OpenSeesMP ex6.tcl
```

OpenSees -- Open System For Earthquake Engineer  
Pacific Earthquake Engineering Research Center --

(c) Copyright 1999,2000 The Regents of the U  
All Rights Reserved

(Copyright and Disclaimer @ <http://www.berkeley.edu/>)

```
Duration Process 0 14
Process Terminating 0
Duration Process 1 6
Process Terminating 1
bin> 
```

# Simple Load Balancing

ex7.tcl

```
set np [getNP]
set pid [getPID]
set count 0

source model.tcl
source analysis.tcl

set tStart [clock seconds]
if {$pid == 0} {

    # Coordinator

    set recordsFile [open motionList r]
    set lines [split [read $recordsFile] \n]
    set numLines [llength $lines]
    foreach line $lines {
        recv -pid ANY pidWorker
        send -pid $pidWorker $line
    }

    for {set i 1} {$i < $np} {incr i 1} {
        send -pid $i "DONE"
    }
} else {

    # Worker

    set done NOT_DONE;
    while {$done != "DONE"} {
        send -pid 0 $pid
        recv -pid 0 line
        set record [lindex $line 0]

        if {$record == "DONE"} {
            break;
        }

        set npts [lindex $line 1]
        set dt [lindex $line 2]

        doModel;

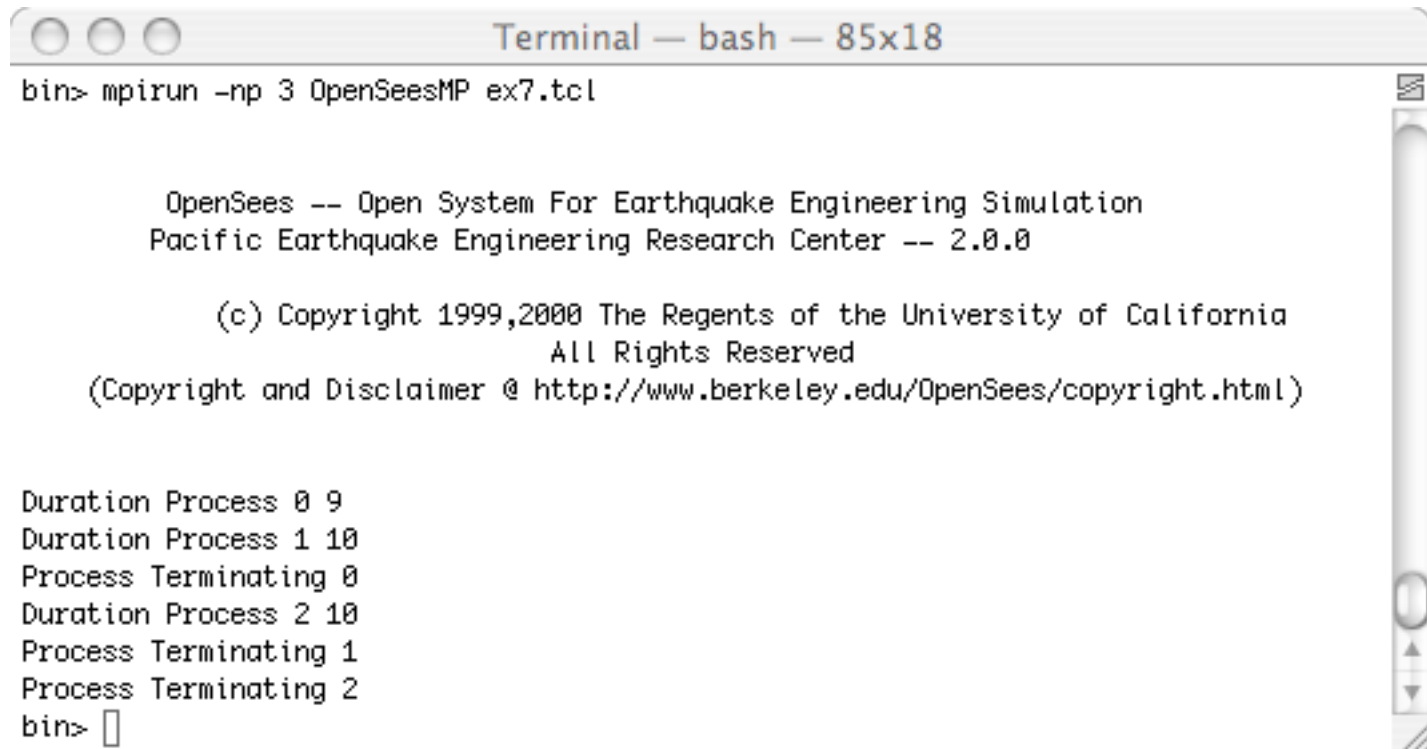
        doGravityAnalysis;

        loadConst -time 0.0

        set accelSeries "Path -filePath $record -dt $dt -factor 386.4"
        pattern UniformExcitation 2 1 -accel $accelSeries

        doRecorders $record $npts $dt
        set ok [doDynamicAnalysis $npts $dt]
        wipe
    }

    set tFinish [clock seconds]
    barrier
    puts "Duration Process $pid [expr $tFinish - $tStart]"
}
```



```
bin> mpirun -np 3 OpenSeesMP ex7.tcl

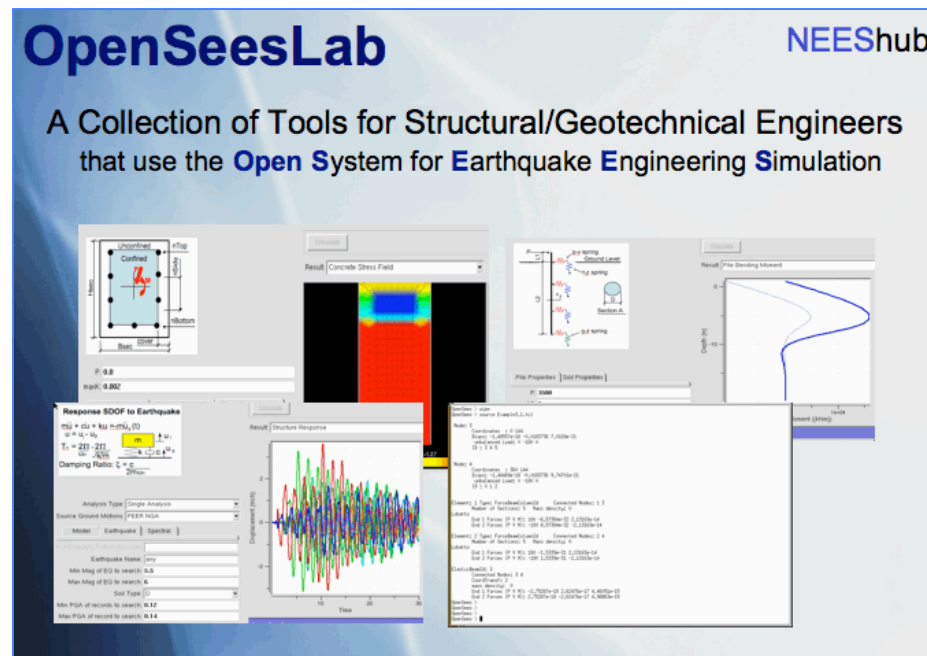
      OpenSees -- Open System For Earthquake Engineering Simulation
      Pacific Earthquake Engineering Research Center -- 2.0.0

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

Duration Process 0 9
Duration Process 1 10
Process Terminating 0
Duration Process 2 10
Process Terminating 1
Process Terminating 2
bin> 
```

# The OpenSeesLab tool:

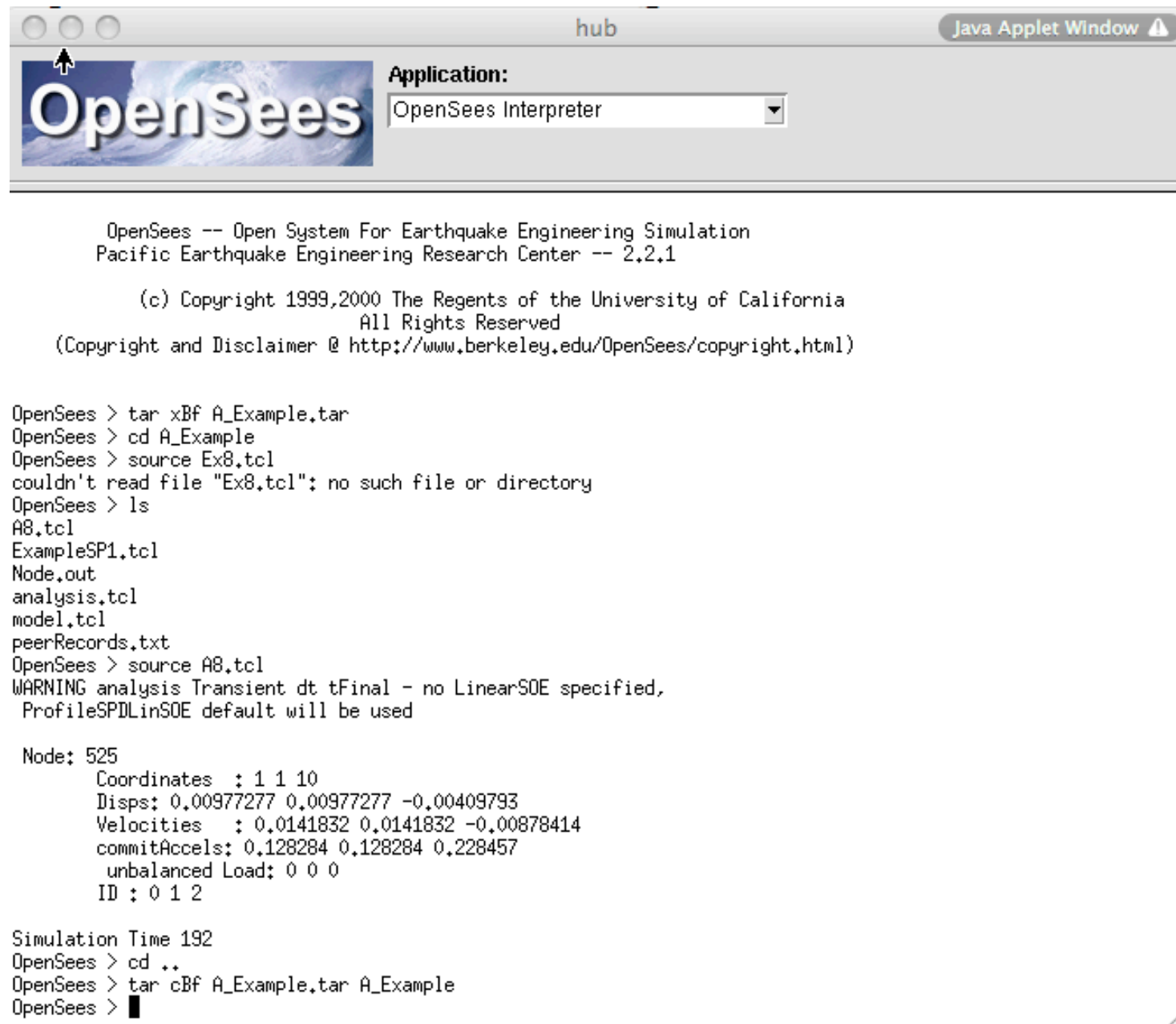
<http://nees.org/resources/tools/openseeslab>



Is a suite of Simulation Tools powered by OpnSees for:

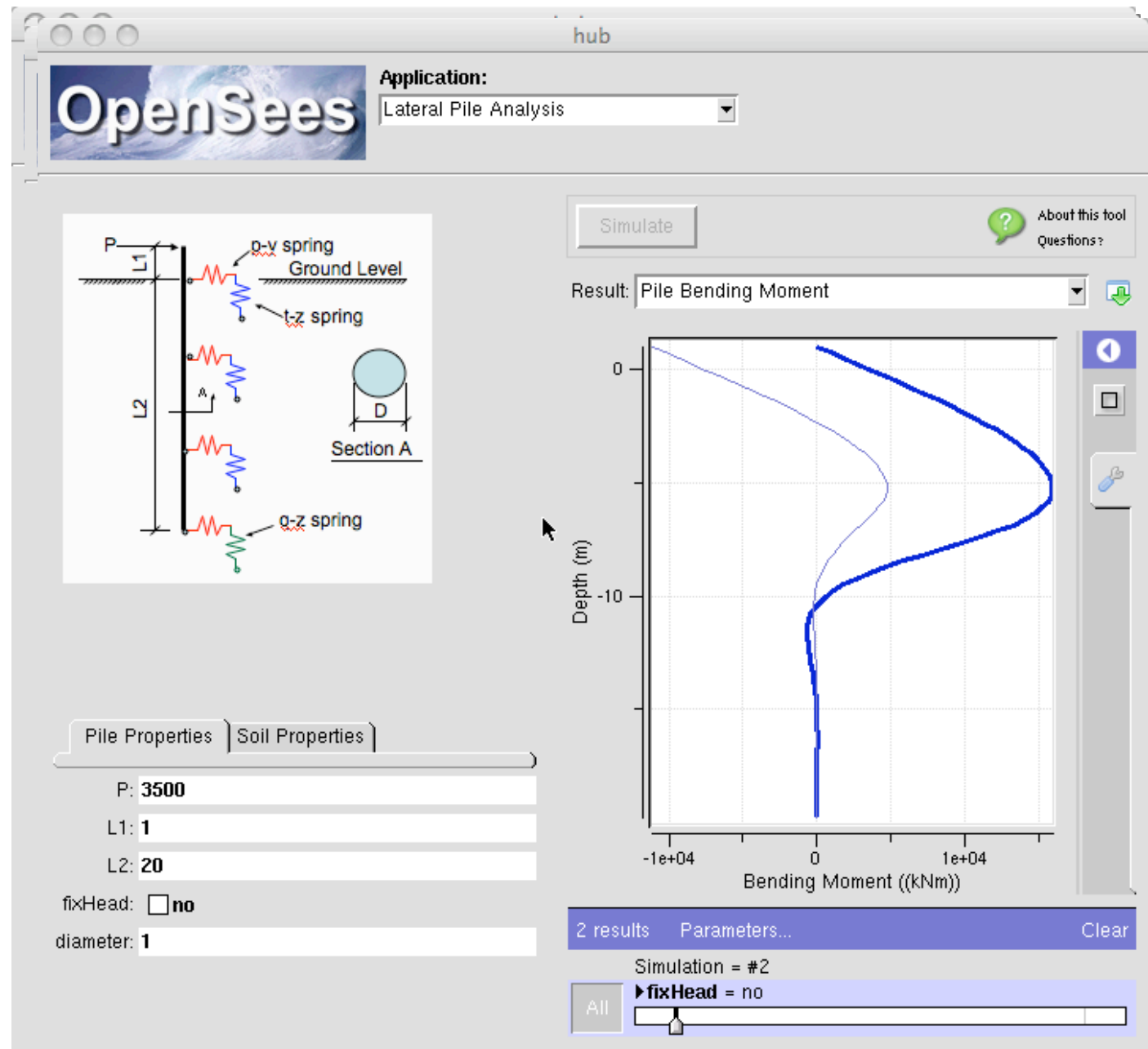
1. Submitting OpenSees scripts to NEEShub resources
2. Educating students and practicing engineers

# OpenSees Interpreter Tool



# Lateral Pile Analysis

[http://opensees.berkeley.edu/wiki/index.php/Laterally-Loaded\\_Pile\\_Foundation](http://opensees.berkeley.edu/wiki/index.php/Laterally-Loaded_Pile_Foundation)  
Chris McGann U. Washington



# Parallel Script Submission Tool

The screenshot shows a web-based interface for submitting OpenSees jobs. The title bar indicates it's running in 'Xnest'. The main header features the 'OpenSees' logo and a dropdown menu for 'Application' set to 'Parallel Job Submission'. On the left, there are input fields for 'Main Script' and 'Resource' (set to 'Kraken'). Below these is a 'Kraken Options' section with 'Application' set to 'OpenSeesMP' and '# Processors' set to '512'. A yellow button labeled 'Simulate' is next to a 'new input parameters' link. The right side contains a 'Parallel Job Submission Tool' section with explanatory text and three notes regarding script location, control flow, and a specific example path.

Xnest

**OpenSees**

Application: Parallel Job Submission

Main Script:

Resource: Kraken

**Kraken Options**

Application: OpenSeesMP

# Processors: 512

Simulate new input parameters

**Parallel Job Submission Tool**

This tool can be used to submit parallel opensees jobs. The user is asked which parallel OpenSees application to use, which parallel machine to run on, how many processes to run and which parallel machine to run these on. The results from the analysis when completed will be placed in the users /scratch directory. The actual directory location will be shown in the result screen.

NOTE: the main script CANNOT be located in your home directory. It and all the files it requires must be in a subdirectory.

NOTE: control will return after the job has been submitted, AND NOT after the job has completed. This means you may have to wait awhile before the actual results are located in your output directory.

NOTE: as an example set the main script as:  
/apps/openseesbuild/current/NEEShubExamples/SmallIMP/Example.tcl

DO NOT SELECT OpenSeesSP for this example.

# OpenSees Parameter Study Tool

hub

**OpenSees**

Application: Parameter Study

Resource: OSG

numParameter: 3

Main Script:

Parameter 1

Name:

File:

Parameter 2

Name:

File:

Parameter 3

Name:

File:

Simulate new input parameters

About this tool Questions?

### Parameter Study Submission Tool

This tool can be used to perform parameter studies with OpenSees. For each parameter, the user inputs the parameter name and the file containing all the values to be tried for that parameter. The user also specifies a main file to run in which the parameter value is used. DO NOT ASSIGN THE VALUE IN THE MAIN SCRIPT, otherwise you will overwrite the value to be used from the parameter file.

example:

If numParameter is set to 2 and the main script contains the following 3 lines:

```
set sum [expr $varA + $varB]
set fileOut [open $varA$varB.out w]
puts $fileOut "$a + $b = $sum"
```

Then, if in the Parameter1 box, we set the name to be varA and the associated file has the number 1 and 2. And if in the Parameter 2 box, we set the name to be varB and the associated file has the number 3, 4 and 5, the output directory when we hit the submit button will contain 6 files 13.out,14.out,15.out,23.out, 24.out and 25.out, each with a different message.

Be careful, make sure the results go to different filename, when using recorders use for example recorder Node -file node\$varA\$varB.out ....

Also only use the OSG (Open Science Grid) option when you have large models to run as it can take from minutes to hours for your job to actually start. The OSG option will place each run in a separate directory. Complain if you don't like this!

NOTE: all sourced files, ground motions, etc. must be in the main script or subscripts directory.

**WARNING:** Don't use a file in home directory as main script

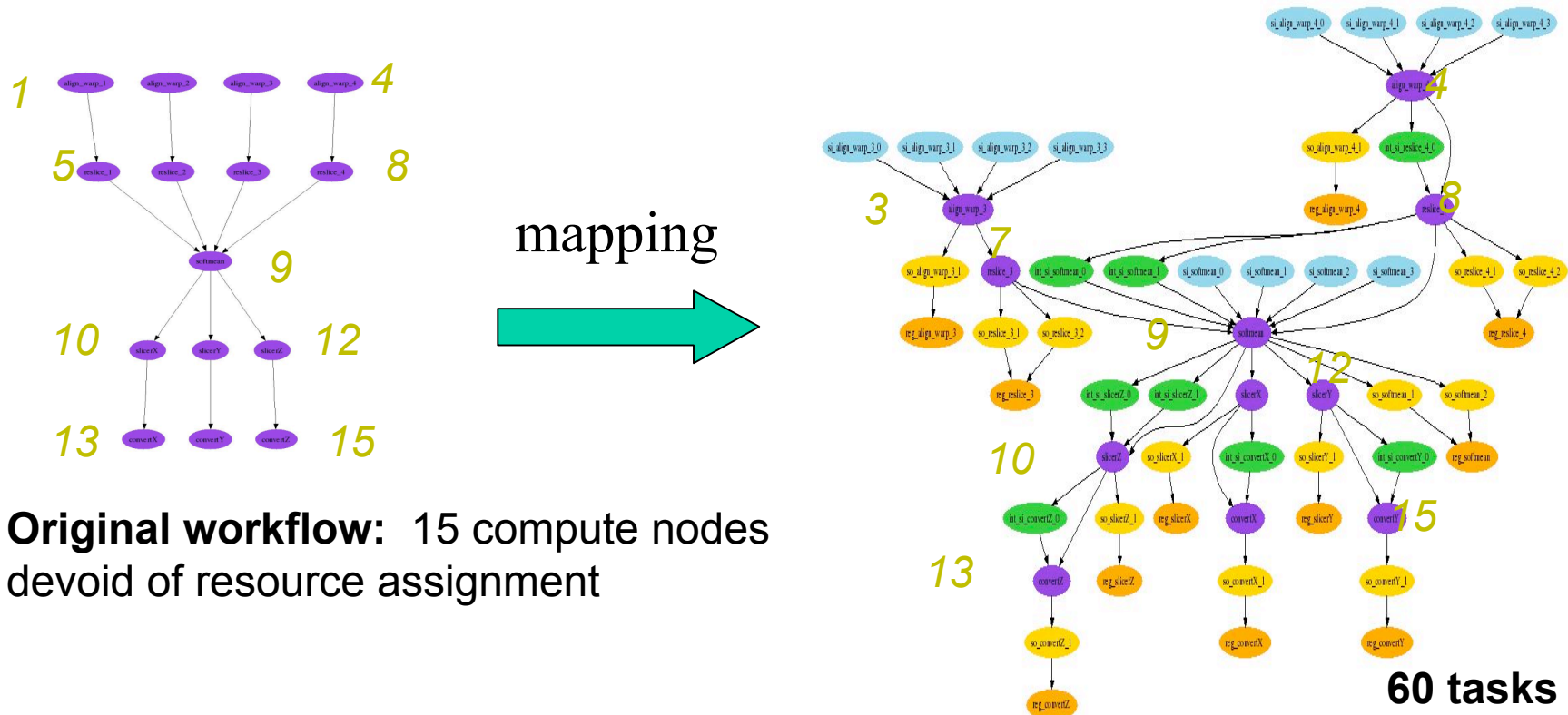


# Workflows in the Cloud

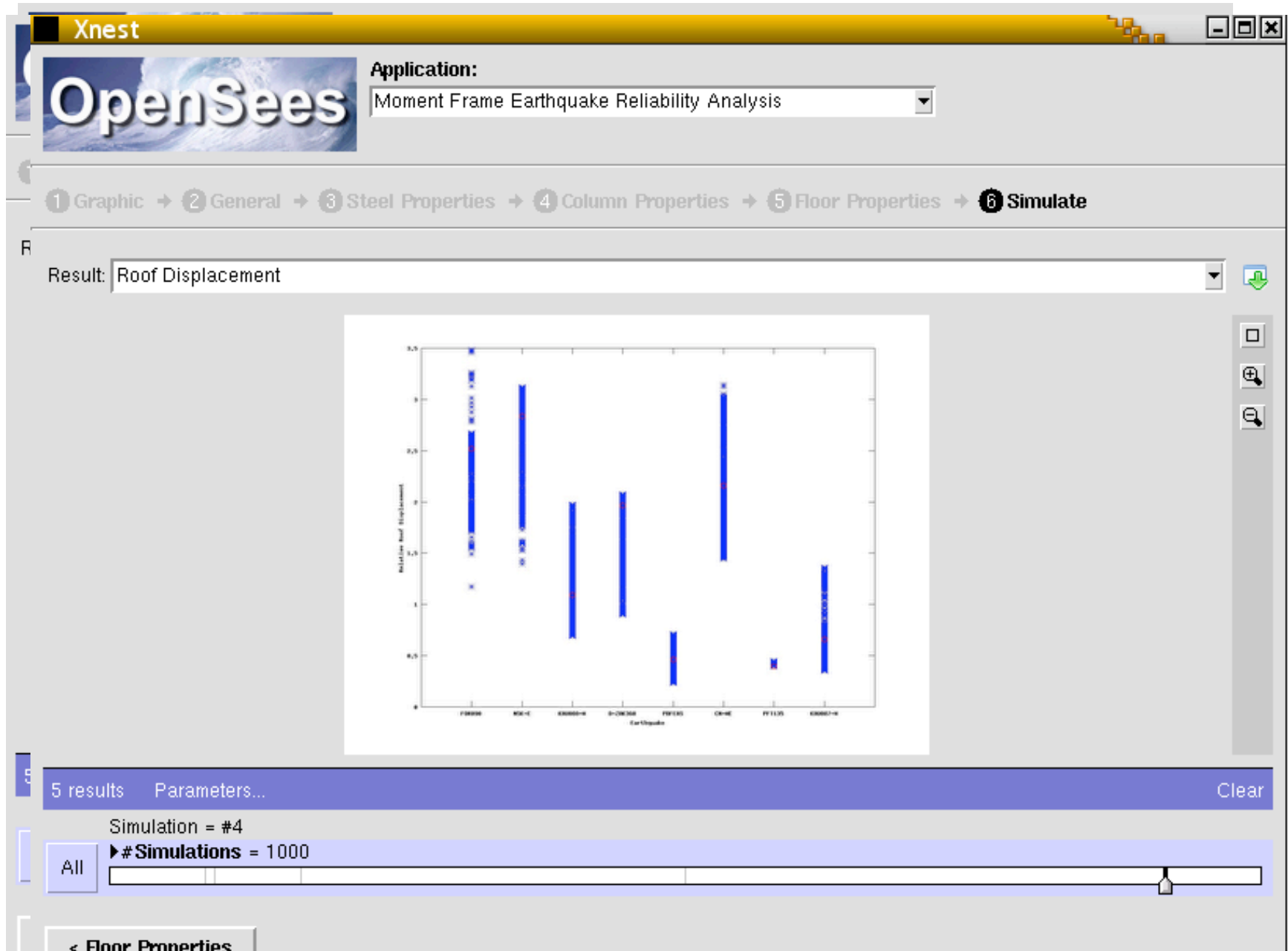


## OpenSees cannot do it all!

Software exists (pegasus, ...) for creating scientific workflows that can take advantage of computational resources in the cloud! A scientific workflow allows engineers to compose and execute a series of computational or data manipulation steps in a scientific application.



# Moment Frame Reliability Analysis



# OpenSees Challenge 2012

At next years OpenSees Days Workshop (August 15-16), **We** will award 2 iPod's:

- 1) One to the person (anyone other than myself) who submits the best **OpenSees powered app** to NEEShub.
- 2) The other to the person who submits best new code module.

All code submitted between last years workshop and this years workshop will be considered. Winner will be judged by Workshop participants.

# Thank You

