

# Supercomputing in Plain English

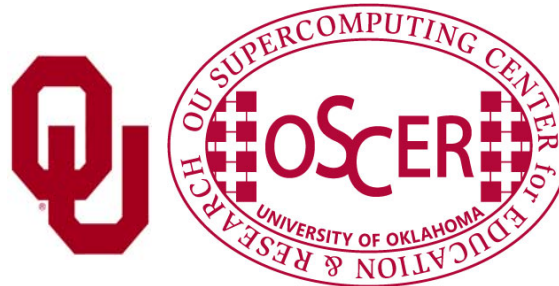
## Part VIII: High Throughput Computing

Henry Neeman, Director

OU Supercomputing Center for Education & Research

University of Oklahoma

Wednesday October 24 2007





# This is an experiment!

---

It's the nature of these kinds of videoconferences that **failures are guaranteed to happen!**

**NO PROMISES!**

So, please bear with us. Hopefully everything will work out well enough.





# Access Grid/VRVS

---

If you're connecting via the Access Grid or VRVS, the venue is:

NCSA Venue Titan

It's available Wed Oct 17 2007 1:00-4:30pm Central Time, but the workshop starts at 3:00pm Central Time.

Many thanks to John Chapman of U Arkansas for setting this up for us.





# iLinc

---

We only have about 40-45 simultaneous iLinc connections available.

Therefore, each institution has at most one iLinc person designated.

If you're the iLinc person for your institution, you've already gotten e-mail about it, so please follow the instructions.

If you aren't your institution's iLinc person, then you can't become it, because we're completely out of iLinc connections.

Many thanks to Katherine Kantardjieff of California State U Fullerton for setting this up for us.





# QuickTime Broadcast

---

If you don't have iLinc, you can connect via QuickTime:

`rtsp://129.15.254.141/neeman_02.sdp`

We strongly recommend using QuickTime player, since we've seen it work.

When you run it, traverse the menus

File -> Open URL

Then paste in the rstp URL the Movie URL space, and click OK.

Many thanks to Kevin Blake of OU for setting this up.



# Phone Bridge

---

If all else fails, you can call into our phone bridge:

1-866-285-7778, access code 6483137#

Please mute yourself and use the phone to listen.

Don't worry, I'll call out slide numbers as we go.

To ask questions, please use Google Talk or Gmail.

Many thanks to Amy Apon of U Arkansas for setting this up for us, and to U Arkansas for absorbing the costs.





# Google Talk

---

To ask questions, please use our Google Talk group chat session (text only).

You need to have (or create) a gmail.com account to use Google Talk.

Once you've logged in to your gmail.com account, go to:

<http://www.google.com/talk/>

and then contact the user named:

oscer.sipe

Alternatively, you can send your questions by e-mail to [oscer.sipe@gmail.com](mailto:oscer.sipe@gmail.com).



# This is an experiment!

---

## REMINDER:

It's the nature of these kinds of videoconferences that **failures are guaranteed to happen!**

## NO PROMISES!

So, please bear with us. Hopefully everything will work out well enough.







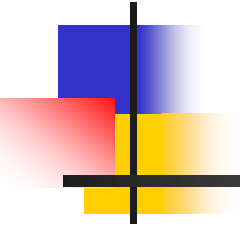
# Outline

---

- What is High Throughput Computing?
- Tightly Coupled vs Loosely Coupled
- What is Opportunistic Computing?
- Condor
- Grid Computing
- OU's NSF CI-TEAM Project (a word from our sponsors)



# What is High Throughput Computing?





# High Throughput Computing

*High Throughput Computing* (HTC) means getting lots of work done per large time unit (e.g., jobs per month).

This is different from *High Performance Computing* (HPC), which means getting **a particular job** done in less time (e.g., calculations per second).





# Throughput vs Performance

- **Throughput** is a side effect of how much time your job takes from when you first submit it until it completes.
- **Performance** is the factor that controls how much time your jobs takes from when it first starts running until it completes.
- Example:
  - You submit a job at 1:00am on January 1.
  - It starts running at 5:00pm on January 2.
  - It finishes running at 6:00pm on January 2.
  - Its performance is fast; its throughput is slow.



# High Throughput on a Cluster?

---

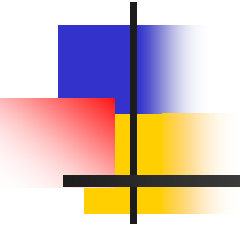
Is it possible to get high throughput on a cluster?

Sure – it just has to be a cluster that no one else is trying to use!

Normally, a cluster that is shared by many users is fully loaded with jobs all the time. So your throughput depends on when you submit your jobs, and even how many jobs you submit at a time.

Depending on a variety of factors, a job you submit may wait in the queue for anywhere from seconds to days.

# Tightly Coupled vs Loosely Coupled





# Tightly Coupled vs Loosely Coupled

- **Tightly coupled** means that all of the parallel tasks have to advance forward in lockstep, so they have to communicate frequently.
- **Loosely coupled** means that the parallel tasks can largely or completely ignore each other (little or no communication), and they can advance at different rates.



# Tightly Coupled Example

---

Consider weather forecasting.

You take your simulation domain – for example, the continental United States – split it up into chunks, and give each chunk to an MPI process.

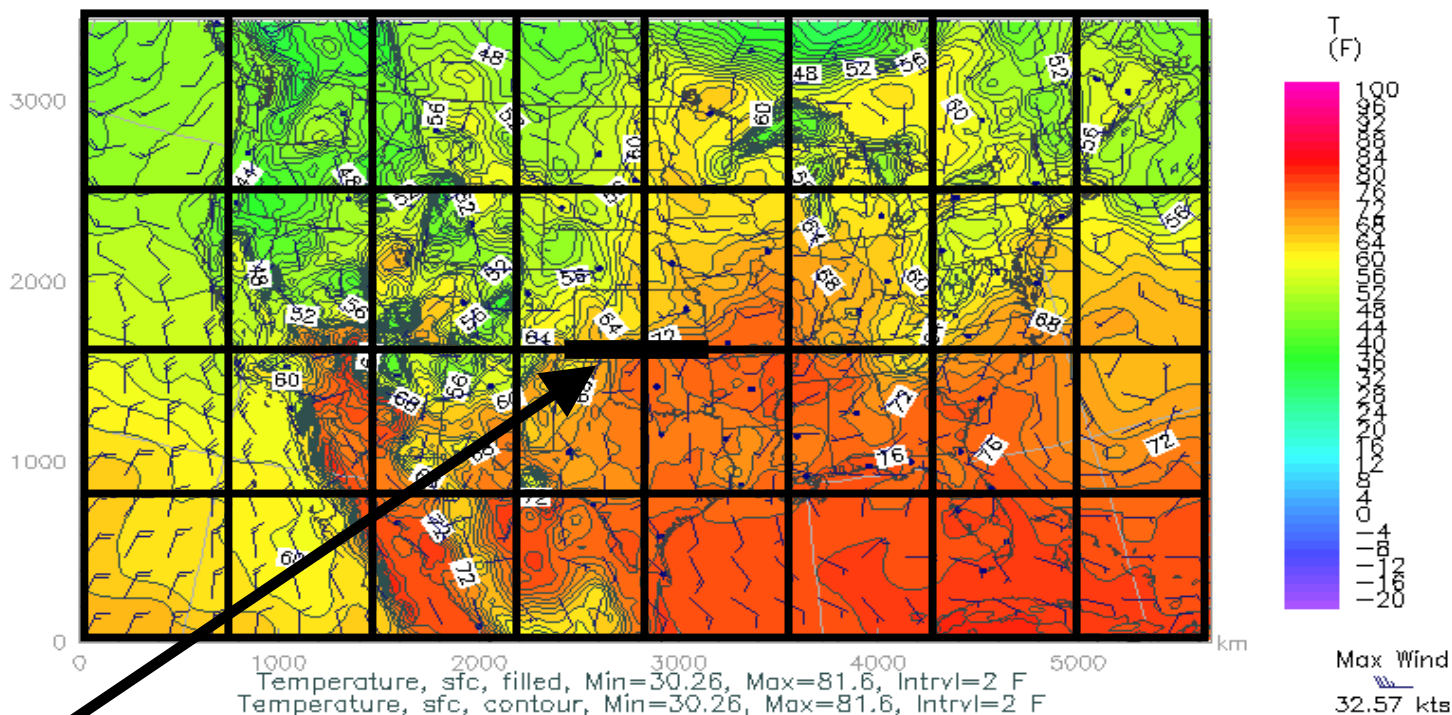
But, the weather in northern Oklahoma affects the weather in southern Kansas.

So, every single timestep, the process that contains northern Oklahoma has to communicate with the process that contains southern Kansas, so that the interface between the processes has the same weather at the same time.



# Tightly Coupled Example

Thu, 25 May 2006, 8 am CDT (13Z)  
Surface Temperature



OK/KS boundary

<http://www.caps.ou.edu/wx/p/r/conus/fcst/>

CAPS/OU Experimental ADAS Anlys

CONUS, 210x128x50, dx=27 km

05/25/06 08:45 CDT



Supercomputing in Plain English: High Throughput Computing  
Wednesday October 24 2007



# Loosely Coupled Example

An application is known as *embarrassingly parallel*, or *loosely coupled*, if its parallel implementation:

1. can straightforwardly be broken up into roughly equal amounts of work per processor, **AND**
2. has minimal parallel overhead (e.g., communication among processors).

We love embarrassingly parallel applications, because they get near-perfect parallel speedup, sometimes with only modest programming effort.



# Monte Carlo Methods

Monte Carlo is a city in the tiny European country Monaco.

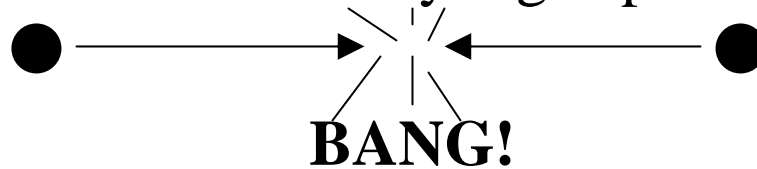
People gamble there; that is, they play games of chance, which involve randomness.

*Monte Carlo methods* are ways of simulating (or otherwise calculating) physical phenomena based on randomness.

Monte Carlo simulations typically are embarrassingly parallel.

# Monte Carlo Methods: Example

Suppose you have some physical phenomenon. For example, consider High Energy Physics, in which we bang tiny particles together at incredibly high speeds.



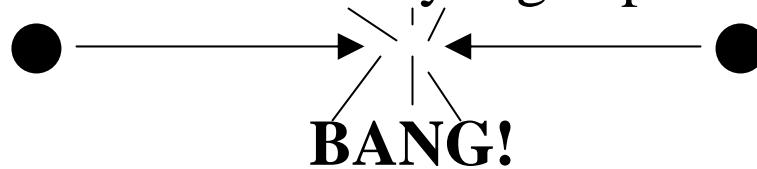
We want to know, say, the average properties of this phenomenon.

There are infinitely many ways that two particles can be banged together.

So, we can't possibly simulate all of them.

# Monte Carlo Methods: Example

Suppose you have some physical phenomenon. For example, consider High Energy Physics, in which we bang tiny particles together at incredibly high speeds.



We want to know, say, the average properties of this phenomenon.

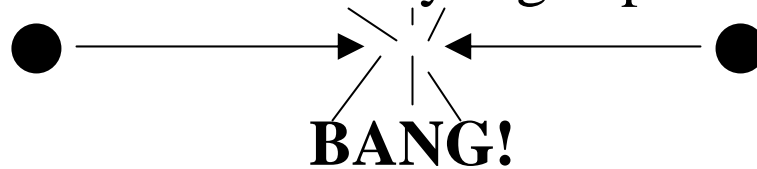
There are infinitely many ways that two particles can be banged together.

So, we can't possibly simulate all of them.

**Instead**, we can **randomly choose a finite subset** of these infinitely many ways and simulate only the subset.

# Monte Carlo Methods: Example

Suppose you have some physical phenomenon. For example, consider High Energy Physics, in which we bang tiny particles together at incredibly high speeds.



We want to know, say, the average properties of this phenomenon.

There are infinitely many ways that two particles can be banged together.

So, we can't possibly simulate all of them.

The average of this subset will be close to the actual average.



# Monte Carlo Methods

In a Monte Carlo method, you randomly generate a large number of example cases (*realizations*) of a phenomenon, and then take the average of the properties of these realizations.

When the realizations' average converges (i.e., doesn't change substantially if new realizations are generated), then the Monte Carlo simulation stops.

This can also be implemented by picking a high enough number of realizations to be sure, mathematically, of convergence.



# MC: Embarrassingly Parallel

Monte Carlo simulations are embarrassingly parallel, because each realization is completely independent of all of the other realizations.

That is, if you're going to run a million realizations, then:

1. you can straightforwardly break up into roughly  $1M / N_p$  chunks of realizations, one chunk for each of the  $N_p$  processes, **AND**
2. the only parallel overhead (e.g., communication) comes from tracking the average properties, which doesn't have to happen very often.





# Serial Monte Carlo

Suppose you have an existing serial Monte Carlo simulation:

```
PROGRAM monte_carlo
  CALL read_input(...)
  DO realization = 1, number_of_realizations
    CALL generate_random_realization(...)
    CALL calculate_properties(...)
  END DO
  CALL calculate_average(...)
END PROGRAM monte_carlo
```

How would you parallelize this?

# Parallel Monte Carlo: MPI

```
PROGRAM monte_carlo_mpi
  [MPI startup]
  IF (my_rank == server_rank) THEN
    CALL read_input(...)
  END IF
  CALL MPI_Bcast(...)
  number_of_realizations_per_process = &
& number_of_realizations / number_of_processes
  DO realization = 1, number_of_realizations_per_process
    CALL generate_random_realization(...)
    CALL calculate_realization_properties(...)
    CALL calculate_local_running_average(...)
  END DO
  IF (my_rank == server_rank) THEN
    [collect properties]
  ELSE
    [send properties]
  END IF
  CALL calculate_global_average_from_local_averages(...)
  CALL output_overall_average(...)
[MPI shutdown]
END PROGRAM monte_carlo_mpi
```

# Parallel Monte Carlo: HTC

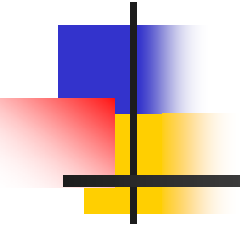
Suppose you have an existing serial Monte Carlo simulation:

```
PROGRAM monte_carlo
  CALL read_input(...)
  number_of_realizations_per_job = &
&   number_of_realizations / number_of_jobs
  DO realization = 1, number_of_realizations_per_job
    CALL generate_random_realization(...)
    CALL calculate_properties(...)
  END DO
  CALL calculate_average_for_this_job(...)
  CALL output_average_for_this_job(...)
END PROGRAM monte_carlo
```

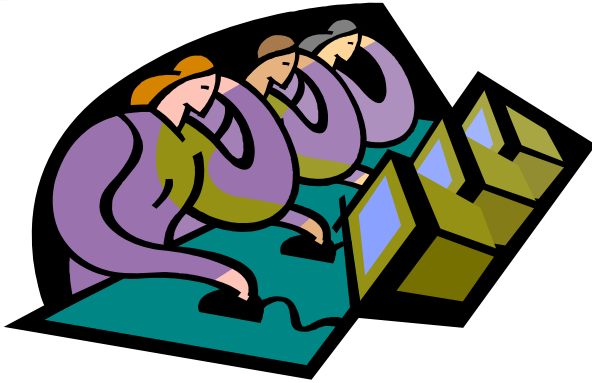
To parallelize this for **HTC**, simply submit **number\_of\_jobs** jobs, and then at the very end run a little program to calculate the overall average.

# What is Opportunistic Computing?

---



# Desktop PCs Are Idle Half the Day



Desktop PCs tend to be active during the workday.



But at night, during most of the year, they're idle. So we're only getting **half their value** (or less).



# Supercomputing at Night

A particular institution – say, OU – has lots of desktop PCs that are **idle during the evening and during intersessions.**

Wouldn't it be great to put them to work on something **useful** to our institution?

That is: What if they could pretend to be a big supercomputer **at night**, when they'd **otherwise be idle anyway?**

This is sometimes known as **opportunistic computing**: When a desktop PC is otherwise idle, you have an opportunity to do number crunching on it.



# Supercomputing at Night Example

SETI – the Search for Extra-Terrestrial Intelligence – is looking for evidence of green bug-eyed monsters on other planets, by mining radio telescope data.

SETI@home runs number crunching software as a screensaver on idle PCs around the world (1.6 million PCs in 231 countries):

<http://setiathome.berkeley.edu/> 

There are many similar projects:

- folding@home (protein folding)
- climateprediction.net
- Einstein@Home (Laser Interferometer Gravitational wave Observatory)
- Cosmology@home
- ...

# BOINC

The projects listed on the previous page use a software package named BOINC (**B**erkeley **O**pen **I**nfrastructure for **N**etwork **C**omputing), developed at the University of California, Berkeley:

<http://boinc.berkeley.edu/>



To use BOINC, you have to insert calls to various BOINC routines into your code. It looks a bit similar to MPI:

```
int main ()
{ /* main */
    ..
    boinc_init();
    ..
    boinc_finish(...);
} /* main */
```





# Condor

---





# Condor is Like BOINC

- Condor steals computing time on existing desktop PCs when they're idle.
- Condor runs in background when no one is sitting at the desk.
- Condor allows an institution to get much more value out of the hardware that's already purchased, because there's little or no idle time on that hardware – all of the idle time is used for number crunching.





# Condor is Different from BOINC

- To use Condor, **you don't need to rewrite your software** to add calls to special routines; in BOINC, you do.
- Condor **works great under Unix/Linux**, but less well under Windows or MacOS (more on this presently); BOINC works well under all of them.
- It's **non-trivial to install Condor** on your own personal desktop PC; it's straightforward to install a BOINC application such as SETI@home.

# Useful Features of Condor

- **Opportunistic** computing: Condor steals time on existing desktop PCs when they're otherwise not in use.
- Condor **doesn't require any changes to the software.**
- Condor can **automatically checkpoint** a running job: every so often, Condor saves to disk the state of the job (the values of all the job's variables, plus where the job is in the program).
- Therefore, Condor can **preempt** running jobs if more important jobs come along, or if someone sits down at the desktop PC.
- Likewise, Condor can **migrate** running jobs to other PCs, if someone sits at the PC or if the PC crashes.
- And, Condor can do all of its **I/O over the network**, so that the job on the desktop PC doesn't consume the desktop PC's local disk.

# Condor Pool @ OU

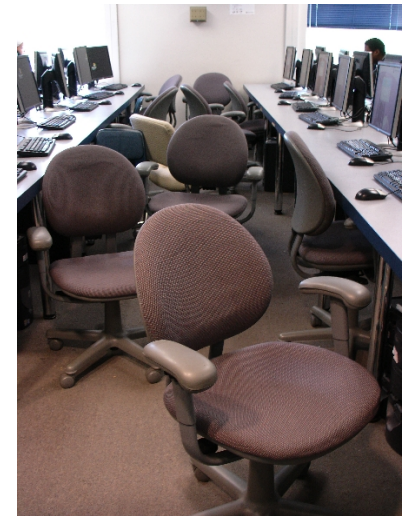
OU IT has deployed a large Condor pool  
(775 desktop PCs in dozens of labs around campus).

OU's Condor pool provides a huge amount of  
computing power – more than OSCER's big  
cluster:

- if OU were a state, we'd be the 10th largest  
state in the US;
- if OU were a country, we'd be the 8th largest  
country in the world.

The hardware and software cost is zero, and the  
labor cost is modest.

Also, we've been seeing empirically that lab PCs  
are available for Condor jobs about 80% of the time.



# Condor Limitations

- The Unix/Linux version has **more features** than Windows or MacOS, which are referred to as “clipped.”
- Your code **shouldn't be parallel** to do opportunistic computing (MPI requires a fixed set of resources throughout the entire run), and it shouldn't try to do any funky communication (e.g., opening sockets).
- For a Red Hat Linux Condor pool, you have to be able to **compile your code** with gcc, g++, g77 or NAG f95.
- Also, depending on the PCs that have Condor on them, you may have limitations on, for example, how big your jobs' RAM footprint can be.



# Running a Condor Job

---

Running a job on Condor pool is a lot like running a job on a cluster:

1. You compile your code using the compilers appropriate for that resource.
2. You submit a batch script to the Condor system, which decides when and where your job runs, magically and invisibly.

# Sample Condor Batch Script

```
Universe      = standard
Executable    = /home/hneeman/NBody/nbody_compiled_for_condor
Notification  = Error
Notify_User   = hneeman@ou.edu
Arguments     = 1000 100
Input         = /home/hneeman/NBody/nbody_input.txt
Output        = nbody_$(Cluster)_$(Process)_out.txt
Error         = nbody_$(Cluster)_$(Process)_err.txt
Log           = nbody_$(Cluster)_$(Process)_log.txt
InitialDir    = /home/hneeman/NBody/Run001
Queue
```

The batch submission command is **condor\_submit**, used like so:

```
condor_submit nbody.condor
```





# Linux Condor on Windows PCs?

If OU's Condor pool uses Linux, how can it be installed in OU IT PC labs? Don't those run Windows?

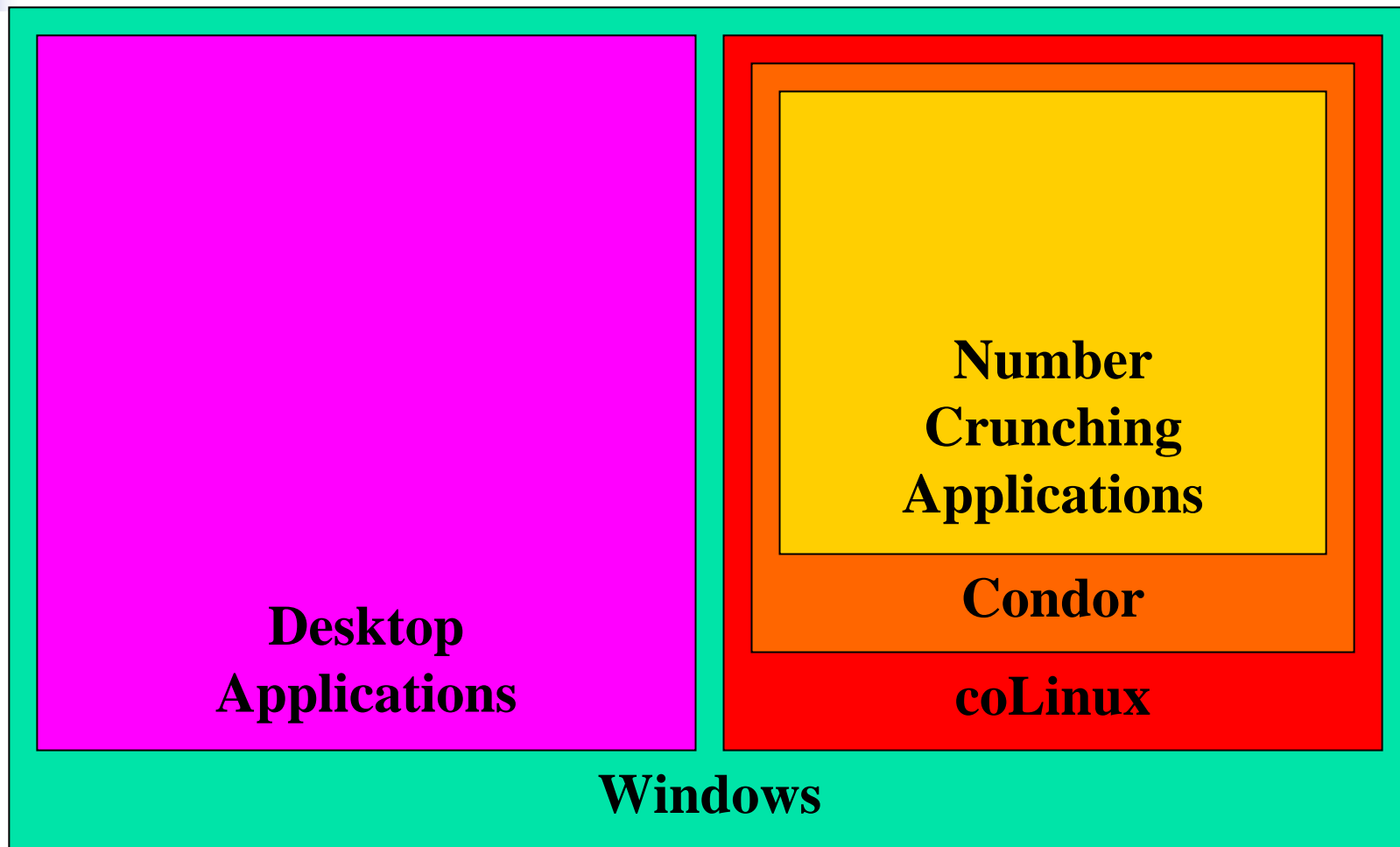
**Yes.**

Our solution is to run Linux inside Windows, using a piece of software named coLinux ("Cooperative Linux"):

<http://www.colinux.org/>



# Condor inside Linux inside Windows



# Advantages of Linux inside Windows

- Condor is full featured rather than clipped.
- Desktop users have a full Windows experience, without even being aware that coLinux exists.
- A little kludge helps Condor watch the keyboard, mouse and CPU level of Windows, so that Condor jobs don't run when the PC is otherwise in use.

Want to try it yourself?

[http://www.oscer.ou.edu/CondorInstall/condor\\_colinux\\_howto.php](http://www.oscer.ou.edu/CondorInstall/condor_colinux_howto.php)





# Grid Computing

---



# What is Grid Computing?

The term *grid computing* is poorly defined, but the best definition I've seen so far is:

“a distributed, heterogeneous operating system.”

A *grid* can consist of:

- compute resources;
- storage resources;
- networks;
- data collections;
- shared instruments;
- sensor networks;
- and so much more ....

# Grid Computing is Like and Unlike ...

IBM's website has a very good description of grid computing:

- *“Like the Web, grid computing keeps complexity hidden: multiple users enjoy a single, unified experience.*
- *“Unlike the Web, which mainly enables communication, grid computing enables full collaboration toward common ... goals.*
- *“Like peer-to-peer, grid computing allows users to share files.*
- *“Unlike peer-to-peer, grid computing allows many-to-many sharing – not only files but other resources as well.*
- *“Like clusters and distributed computing, grids bring computing resources together.*
- *“Unlike clusters and distributed computing, which need physical proximity and operating homogeneity, grids can be geographically distributed and heterogeneous.*
- *“Like virtualization technologies, grid computing enables the virtualization of IT resources.*
- *“Unlike virtualization technologies, which virtualize a single system, grid computing enables the virtualization of vast and disparate IT resources.”*

[http://www-03.ibm.com/grid/about\\_grid/what\\_is.shtml](http://www-03.ibm.com/grid/about_grid/what_is.shtml)





# Condor is Grid Computing

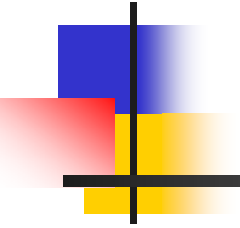
Condor creates a grid out of disparate desktop PCs.

(Actually, they don't have to be desktop PCs; they don't even have to be PCs. You can use Condor to schedule a cluster, or even on a big iron supercomputer.)

From a user's perspective, all of the PCs are essentially invisible; the user just knows how to submit a job, and everything happens magically and invisibly, and at some point the job is done and a result appears.



# OU's NSF CI-TEAM Project







# NSF CI-TEAM Grant

---

“Cyberinfrastructure Education for Bioinformatics and Beyond” (\$250,000, 12/01/2006 – 11/30/2008)

OSCER received a grant from the National Science Foundation’s Cyberinfrastructure Training, Education, Advancement, and Mentoring for Our 21st Century Workforce (CI-TEAM) program.





# OU's NSF CI-TEAM Grant

“Cyberinfrastructure Education for Bioinformatics and Beyond” (\$249,976)

Objectives:

- Provide Condor resources to the national community.
- Teach users to use Condor.
- Teach sysadmins to deploy and administer Condor.
- Teach supercomputing to everyone!
- Teach bioinformatics students to use BLAST on Condor.

**You can join!**



Supercomputing in Plain English: High Throughput Computing  
Wednesday October 24 2007

# OU's NSF CI-TEAM Grant

## Participants at OU

(29 faculty/staff in 16 depts)

- Information Technology
  - OSCER: Neeman (PI)
- College of Arts & Sciences
  - Botany & Microbiology: Conway, Wren
  - Chemistry & Biochemistry: Roe (Co-PI), Wheeler
  - Mathematics: White
  - Physics & Astronomy: Kao, Severini (Co-PI), Skubic, Strauss
  - Zoology: Ray
- College of Earth & Energy
  - Sarkeys Energy Center: Chesnokov
- College of Engineering
  - Aerospace & Mechanical Engr: Striz
  - Chemical, Biological & Materials Engr: Papavassiliou
  - Civil Engr & Environmental Science: Vieux
  - Computer Science: Dhall, Fagg, Hougen, Lakshmivarahan, McGovern, Radhakrishnan
  - Electrical & Computer Engr: Cruz, Todd, Yeary, Yu
  - Industrial Engr: Trafalis
- Health Sciences Center
  - Biochemistry & Molecular Biology: Zlotnick
  - Radiological Sciences: Wu (Co-PI)
  - Surgery: Gusev

## Participants at other institutions

(46 faculty/staff at 30 institutions in 18 states)

1. California State U Pomona (masters-granting, minority serving): Lee
2. Colorado State U: Kalkhan
3. Contra Costa College (CA, 2-year, minority serving): Murphy
4. Delaware State U (masters, EPSCoR): Hubsch, Mulik, Multnovic, Rasamny
5. Earlham College (IN, bachelors): Peck
6. East Central U (OK, masters, EPSCoR): Ferdinand, Myers
7. Emporia State U (KS, masters-granting, EPSCoR): Pheatt, Ballester
8. Harvard U (MA): Altman
9. Kansas State U (EPSCoR): Andresen, Monaco
10. Langston U (OK, masters, minority serving, EPSCoR): Depona, Snow, Tadesse
11. Longwood U (VA, masters): Talaiver
12. Marshall U (WV, masters, EPSCoR): Richards
13. Navajo Technical College (NM, 2-year, tribal, EPSCoR): Ribble
14. Oklahoma Baptist U (bachelors, EPSCoR): Chen, Jett, Jordan
15. Oklahoma Medical Research Foundation (EPSCoR): Wren
16. Oklahoma School of Science & Mathematics (high school, EPSCoR): Samadzadeh
17. Riverside Community College (CA, 2-year): Smith
18. St. Cloud State University (MN, masters): Herath
19. St. Gregory's U (OK, 4-year, EPSCoR): Meyer
20. Southwestern Oklahoma State U (masters, EPSCoR, tribal): Linder, Moseley
21. Syracuse U (NY): Chen
22. Texas A&M U-Corpus Christi (masters): Scherger
23. U Arkansas Fayetteville (EPSCoR): Apon
24. U Arkansas Little Rock (masters, EPSCoR): Jennings, Ramaswamy
25. U Central Oklahoma (masters-granting, EPSCoR): Lemley, Wilson
26. U Illinois Urbana-Champaign: Wang
27. U Kansas (EPSCoR): Bishop, Cheung, Harris, Ryan
28. U Nebraska-Lincoln (EPSCoR): Swanson
29. U North Dakota (EPSCoR): Bergstrom
30. U Northern Iowa (masters-granting): Gray

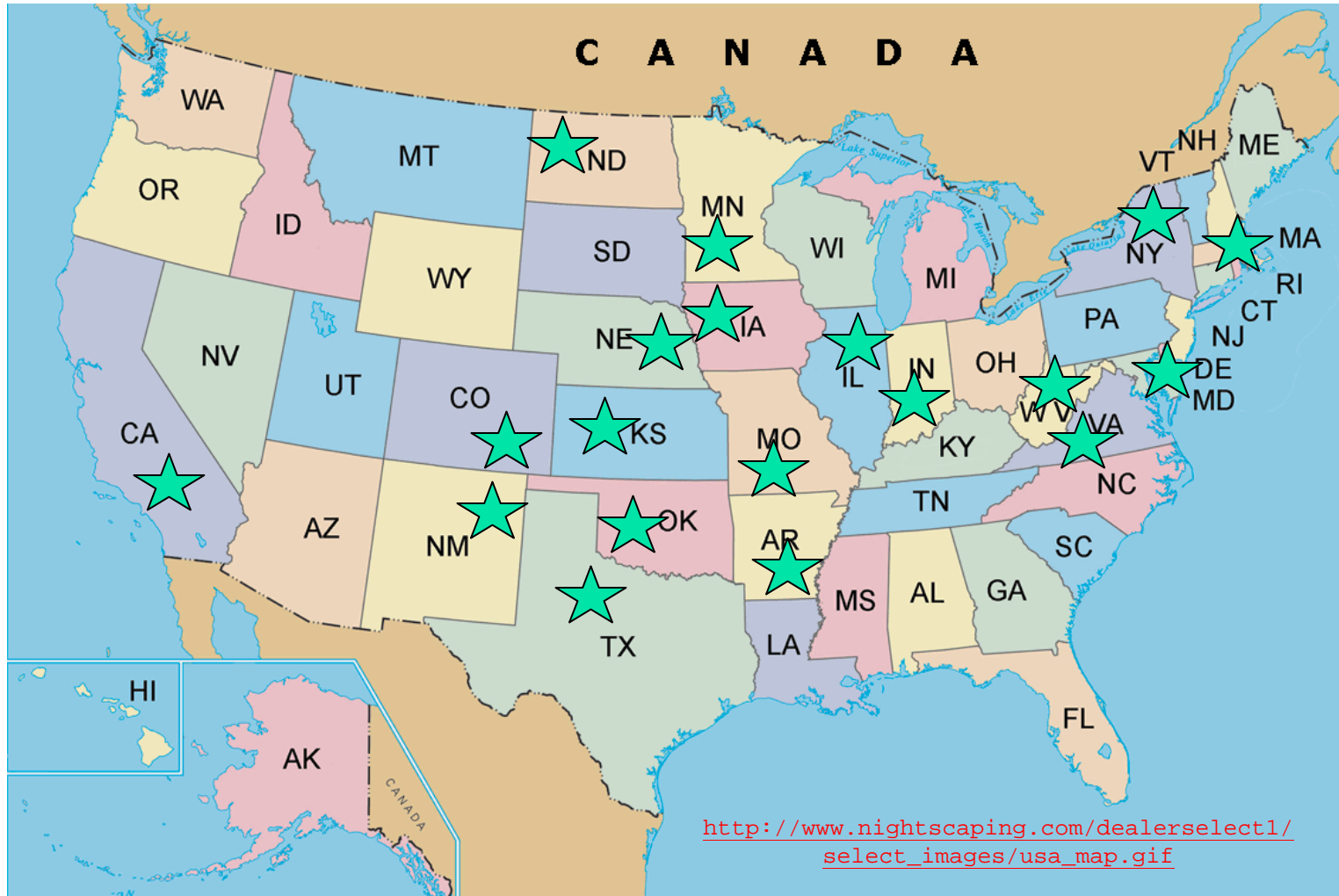
E M E W



Supercomputing in Plain English: High Throughput Computing

Wednesday October 24 2007

# NSF CI-TEAM Participants



Supercomputing in Plain English: High Throughput Computing  
Wednesday October 24 2007



# NSF CI-TEAM Grant

“Cyberinfrastructure Education for Bioinformatics and Beyond” (\$250,000)

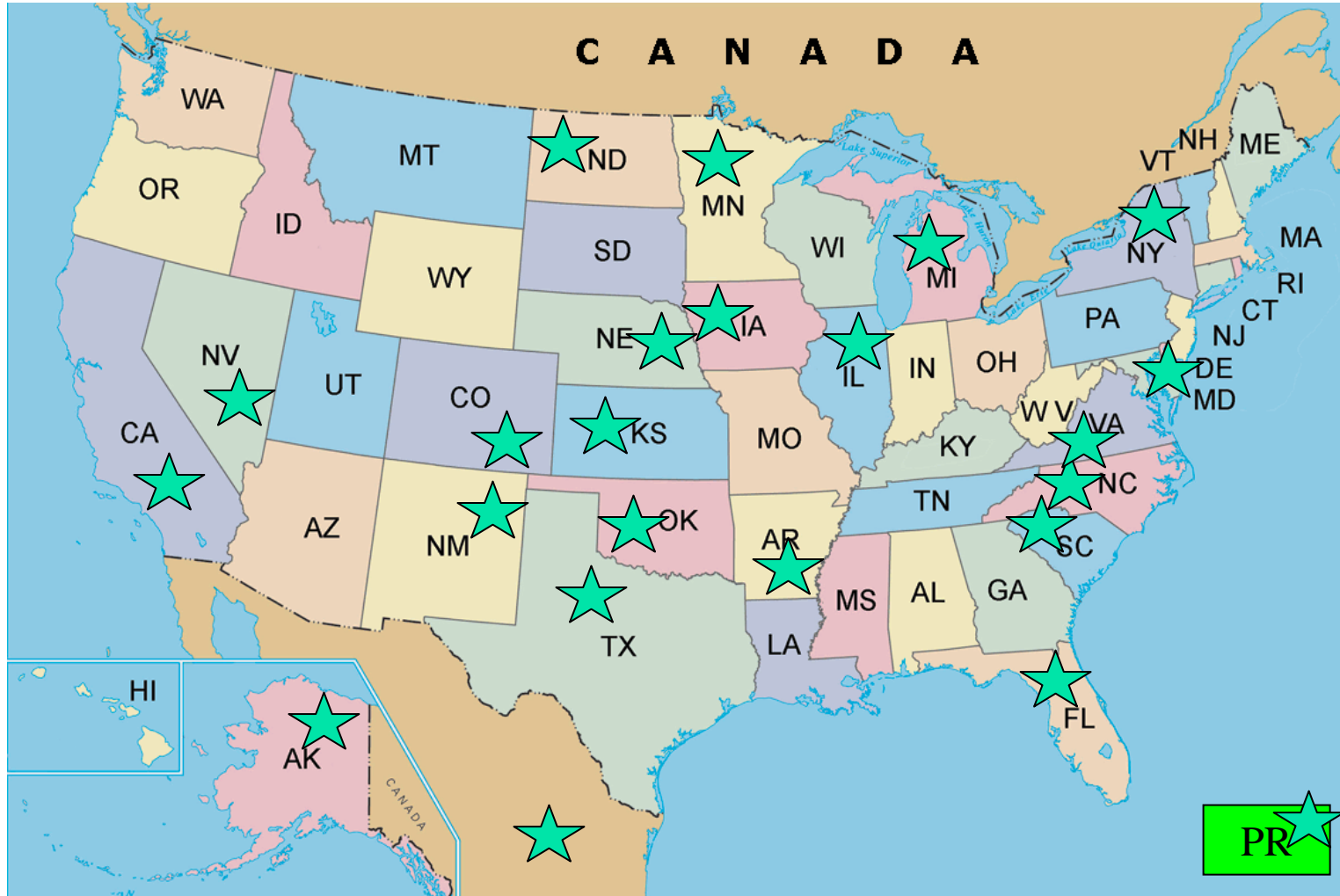
OSCER is providing “Supercomputing in Plain English” workshops via videoconferencing starting in Fall 2007.

~180 people at 29 institutions across the US and Mexico, via:

- Access Grid
- VRVS
- iLinc
- QuickTime
- Phone bridge (land line)



# SiPE Workshop Participants 2007



Supercomputing in Plain English: High Throughput Computing  
Wednesday October 24 2007



# NSF CI-TEAM Grant

---

“Cyberinfrastructure Education for Bioinformatics and Beyond” (\$250,000)

OSCER will be providing supercomputing rounds via videoconferencing starting in Spring 2008.

**INTERESTED?** Contact Henry ([hneeman@ou.edu](mailto:hneeman@ou.edu))





# NSF CI-TEAM Grant

“Cyberinfrastructure Education for Bioinformatics and Beyond” (\$250,000)

OSCER has produced software for installing Linux-enabled Condor inside a Windows PC.

**INTERESTED?** Contact Henry ([hneeman@ou.edu](mailto:hneeman@ou.edu))







# NSF CI-TEAM Grant

---

“Cyberinfrastructure Education for Bioinformatics and Beyond” (\$250,000)

OSCER is providing help on installing Windows as the native host OS, coLinux inside Windows, Linux inside coLinux and Condor inside Linux.

**INTERESTED?** Contact Henry ([hneeman@ou.edu](mailto:hneeman@ou.edu))



# NSF CI-TEAM Proposal

- Follow-on to existing CI-TEAM grant
- Implementation proposal: ~\$1M
- Teach PhD students to be Henry, but remotely via videoconferencing





# Next Time

---

## Part IX: Grab Bag: Scientific Libraries, I/O Libraries, Visualization



Supercomputing in Plain English: High Throughput Computing  
Wednesday October 24 2007



# To Learn More Supercomputing

<http://www.oscer.ou.edu/education.php>



Supercomputing in Plain English: High Throughput Computing  
Wednesday October 24 2007



**Thanks for your  
attention!**

---

**Questions?**