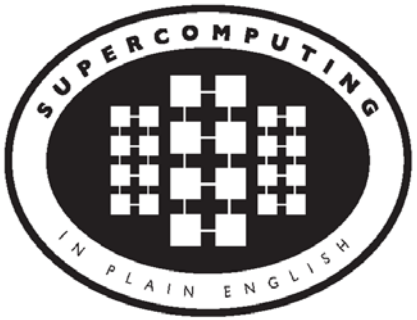
Supercomputing in Plain English

High Throughput Computing

Henry Neeman, Director
OU Supercomputing Center for Education & Research (OSCER)
University of Oklahoma
Tuesday March 26 2013



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.

If you lose your connection, you can retry the same kind of connection, or try connecting another way.

Remember, if all else fails, you always have the toll free phone bridge to fall back on.

H.323 (Polycom etc) #1

If you want to use H.323 videoconferencing – for example, Polycom – then:

- If you AREN'T registered with the OneNet gatekeeper (which is probably the case), then:
  - Dial **`164.58.250.47`**
  - Bring up the virtual keypad.
    On some H.323 devices, you can bring up the virtual keypad by typing: #

    (You may want to try without first, then with; some devices won't work with the #, but give cryptic error messages about it.)
  - When asked for the conference ID, or if there's no response, enter: **`0409`**
  - On most but not all H.323 devices, you indicate the end of the ID with: **#**

H.323 (Polycom etc) #2

If you want to use H.323 videoconferencing – for example, Polycom – then:

- If you ARE already registered with the OneNet gatekeeper (most institutions aren't), dial:

  2500409

Many thanks to Skyler Donahue and Steven Haldeman of OneNet for providing this.

Wowza #1

You can watch from a Windows, MacOS or Linux laptop using Wowza from either of the following URLs:

http://www.onenet.net/technical-resources/video/sipe-stream/

OR

https://vcenter.njvid.net/videos/livestreams/page1/

Wowza behaves a lot like YouTube, except live.

Many thanks to Skyler Donahue and Steven Haldeman of OneNet and Bob Gerdes of Rutgers U for providing this.

Wowza #2

Wowza has been tested on multiple browsers on each of:

- Windows (7 and 8): IE, Firefox, Chrome, Opera, Safari
- MacOS X: Safari, Firefox
- Linux: Firefox, Opera

We've also successfully tested it on devices with:

- Android
- iOS

However, we make no representations on the likelihood of it working on your device, because we don't know which versions of Android or iOS it might or might not work with.

Wowza #3

If one of the Wowza URLs fails, try switching over to the other one.

If we lose our network connection between OU and OneNet, then there may be a slight delay while we set up a direct connection to Rutgers.

Toll Free Phone Bridge

IF ALL ELSE FAILS, you can use our toll free phone bridge:

800-832-0736

* 623 2847 #

Please mute yourself and use the phone to listen.

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

Please use the phone bridge **ONLY** if you cannot connect any other way: the phone bridge can handle only 100 simultaneous connections, and we have over 350 participants.

Many thanks to OU CIO Loretta Early for providing the toll free phone bridge.

Please Mute Yourself

No matter how you connect, please mute yourself, so that we cannot hear you.

(For Wowza, you don't need to do that, because the information only goes from us to you, not from you to us.)

At OU, we will turn off the sound on all conferencing technologies.

That way, we won't have problems with echo cancellation.

Of course, that means we cannot hear questions.

So for questions, you'll need to send e-mail.

Questions via E-mail Only

Ask questions by sending e-mail to:

sipe2013@gmail.com

All questions will be read out loud and then answered out loud.

TENTATIVE Schedule

Tue Jan 22: Overview: What the Heck is Supercomputing?
Tue Jan 29: The Tyranny of the Storage Hierarchy
Tue Feb 5: Instruction Level Parallelism
Tue Feb 12: Stupid Compiler Tricks
Tue Feb 19: Shared Memory Multithreading
Tue Feb 26: Distributed Multiprocessing
Tue March 5: Applications and Types of Parallelism
Tue March 26: Hi Thruput Madness
Tue March 19: NO SESSION (OU's Spring Break)
Tue March 26: High Throughput Computing
Tue Apr 2: GPGPU: Number Crunching in Your Graphics Card
Tue Apr 9: Grab Bag: Scientific Libraries, I/O Libraries, Visualization

Supercomputing Exercises #1

Want to do the "Supercomputing in Plain English" exercises?

- The 3rd exercise will be posted soon at:

  http://www.oscer.ou.edu/education/

- If you don't yet have a supercomputer account, you can get a temporary account, just for the "Supercomputing in Plain English" exercises, by sending e-mail to:

  hneeman@ou.edu

Please note that this account is for doing the **exercises only**, and will be shut down at the end of the series. It's also available only to those at institutions in the USA.

- This week's Introductory exercise will teach you how to compile and run jobs on OU's big Linux cluster supercomputer, which is named Boomer.

Supercomputing Exercises #2

You'll be doing the exercises on your own (or you can work with others at your local institution if you like).

These aren't graded, but we're available for questions:

hneeman@ou.edu

Thanks for helping!

- OU IT
  - OSCER operations staff (Brandon George, Dave Akin, Brett Zimmerman, Josh Alexander, Patrick Calhoun)
  - Horst Severini, OSCER Associate Director for Remote & Heterogeneous Computing
  - Debi Gentis, OU Research IT coordinator
  - Kevin Blake, OU IT (videographer)
  - Chris Kobza, OU IT (learning technologies)
  - Mark McAvoy
- Kyle Keys, OU National Weather Center
- James Deaton, Skyler Donahue and Steven Haldeman, OneNet
- Bob Gerdes, Rutgers U
- Lisa Ison, U Kentucky
- Paul Dave, U Chicago

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.

If you lose your connection, you can retry the same kind of connection, or try connecting another way.

Remember, if all else fails, you always have the toll free phone bridge to fall back on.

Coming in 2013!

From Computational Biophysics to Systems Biology, May 19-21, Norman OK

Great Plains Network Annual Meeting, May 29-31, Kansas City

XSEDE2013, July 22-25, San Diego CA

IEEE Cluster 2013, Sep 23-27, Indianapolis IN

OKLAHOMA SUPERCOMPUTING SYMPOSIUM 2013, Oct 1-2, Norman OK

SC13, Nov 17-22, Denver CO

OK Supercomputing Symposium 2013

2003 Keynote:
Peter Freeman
NSF
Computer & Information Science & Engineering
Assistant Director

2004 Keynote:
Sangtae Kim
NSF Shared Cyberinfrastructure Division Director

2005 Keynote:
Walt Brooks
NASA Advanced Supercomputing Division Director

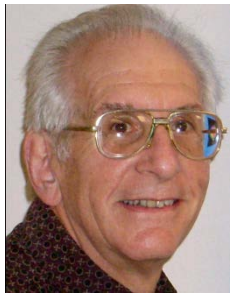
2006 Keynote:
Dan Atkins
Head of NSF's Office of Cyberinfrastructure

2007 Keynote:
Jay Boisseau
Director
Texas Advanced Computing Center
U. Texas Austin

2008 Keynote:
José Munoz
Deputy Office Director/ Senior Scientific Advisor
NSF Office of Cyberinfrastructure

2009 Keynote:
Douglass Post
Chief Scientist
US Dept of Defense HPC Modernization Program

2010 Keynote:
Horst Simon
Deputy Director
Lawrence Berkeley National Laboratory



2011 Keynote:
Barry Schneider
Program Manager
National Science Foundation

2012 Keynote:
Thom Dunning
Director
National Center for Supercomputing Applications

2013 Keynote
to be announced!

FREE! Wed Oct 2 2013 @ OU

http://symposium2013.oscer.ou.edu/

Reception/Poster Session
Tue Oct 1 2013 @ OU
Symposium Wed Oct 2 2013 @ OU

Outline

- What is High Throughput Computing?
- Tightly Coupled vs Loosely Coupled
- What is Opportunistic Computing?
- Condor
- Grid Computing

What is High Throughput Computing?

High Throughput Computing

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

This is different from **High *Performance* Computing** (HPC), which means getting **a particular job** done in less time (for example, 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 very big job at 1:00am on January 1.
  - It sits in the queue for a while.
  - 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, how big your jobs are, 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

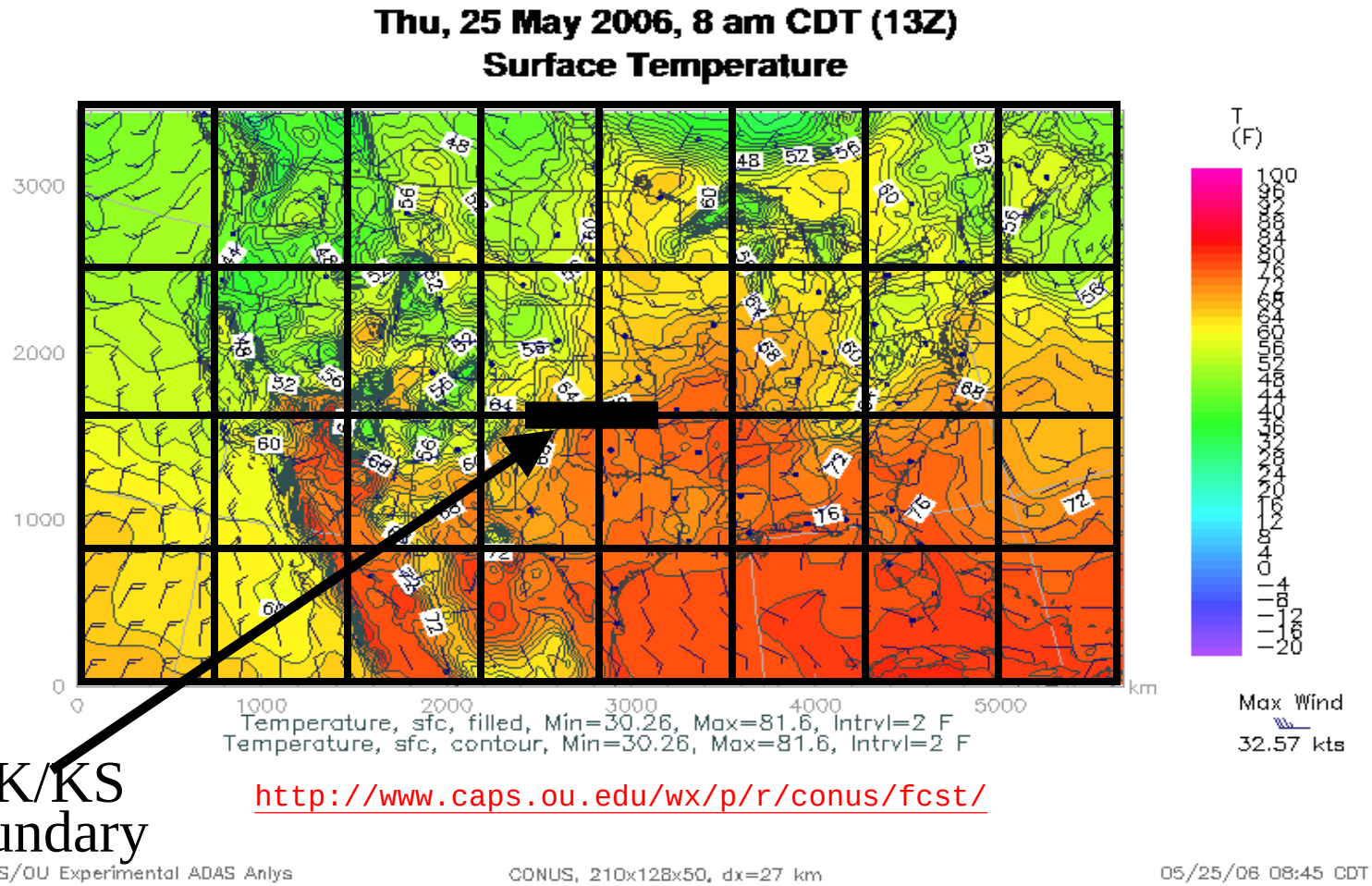
Consider weather forecasting.

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



OK/KS boundary

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 (for example, 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.

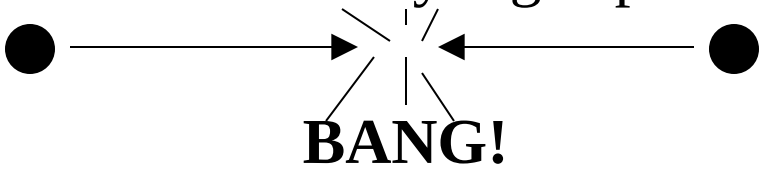
People gamble there; that is, they play games of chance, which involve randomness.

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.



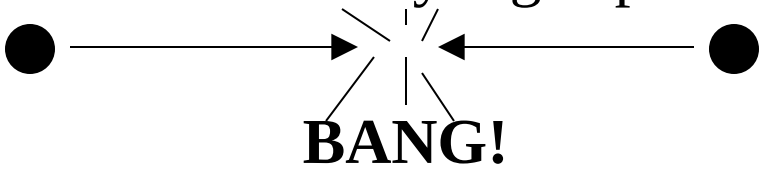
We want to know, say, the average properties of this phenomenon.

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.

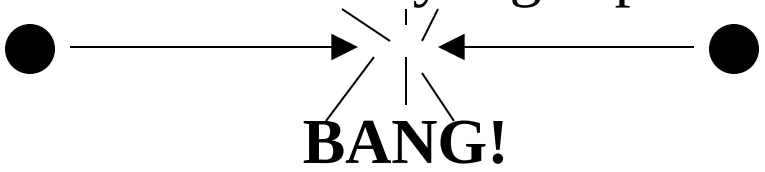
There are infinitely many ways that two particles can be banged together.

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** (that is, 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 (for example, 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

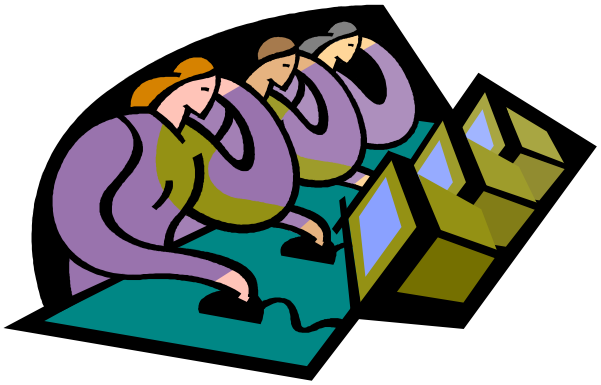
Suppose you have an existing serial Monte Carlo simulation:

```
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 (2+ million PCs in 252 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 PCs local disk.

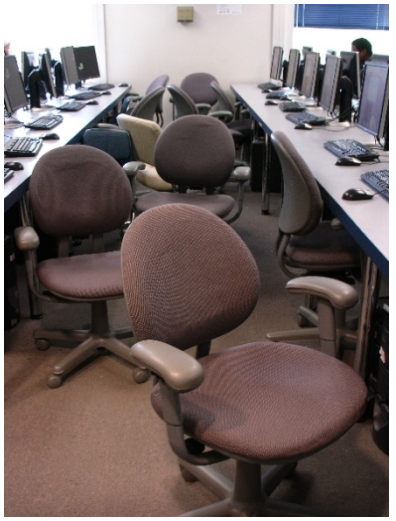
Condor Pool @ OU

OU IT has deployed a large Condor pool (795 desktop PCs in dozens of labs around campus).

OU's Condor pool provides a huge amount of computing.

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 (for example, 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 (which is a Fortran90-to-C translator that then calls gcc).
- 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`

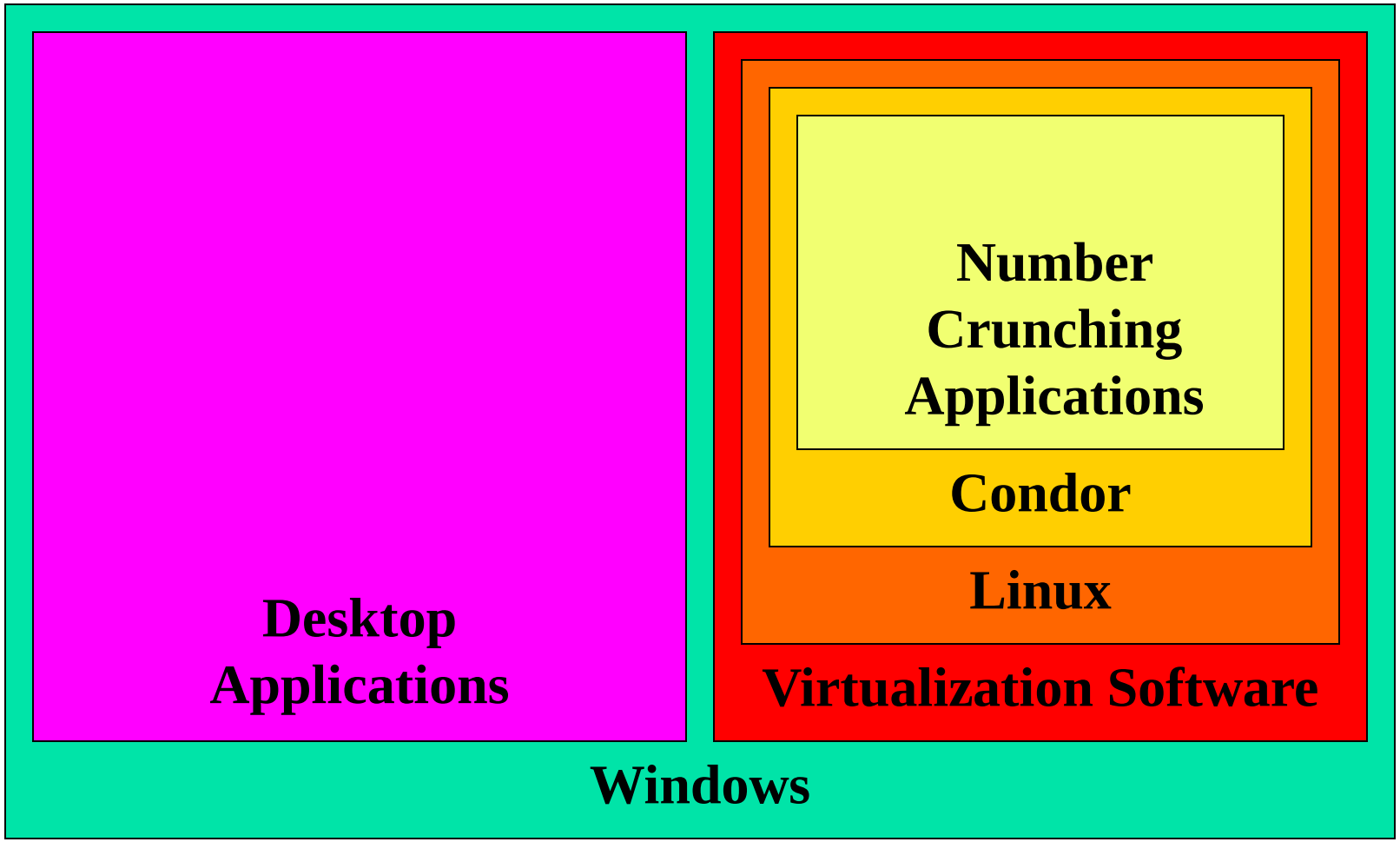
Linux Condor on Windows PCs?

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 Vmware ("virtual machine"), but there are other software packages that can be used (for example, VirtualBox).

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 Condor exists.

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.thocp.net/hardware/grid_computers.htm

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.

Thanks for your attention!

Questions?

www.oscer.ou.edu

OK Supercomputing Symposium 2013

2003 Keynote:
Peter Freeman
NSF
Computer & Information Science & Engineering
Assistant Director

2004 Keynote:
Sangtae Kim
NSF Shared Cyberinfrastructure Division Director

2005 Keynote:
Walt Brooks
NASA Advanced Supercomputing Division Director

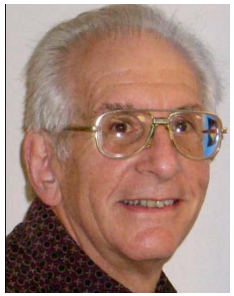
2006 Keynote:
Dan Atkins
Head of NSF's Office of Cyberinfrastructure

2007 Keynote:
Jay Boisseau
Director
Texas Advanced Computing Center
U. Texas Austin

2008 Keynote:
José Munoz
Deputy Office Director/ Senior Scientific Advisor
NSF Office of Cyberinfrastructure

2009 Keynote:
Douglass Post
Chief Scientist
US Dept of Defense HPC Modernization Program

2010 Keynote:
Horst Simon
Deputy Director
Lawrence Berkeley National Laboratory



2011 Keynote:
Barry Schneider
Program Manager
National Science Foundation

2012 Keynote:
Thom Dunning
Director
National Center for Supercomputing Applications

2013 Keynote
to be announced!

FREE! Wed Oct 2 2013 @ OU

http://symposium2013.oscer.ou.edu/

Reception/Poster Session
Tue Oct 1 2013 @ OU
Symposium Wed Oct 2 2013 @ OU

Thanks for your attention!

Questions?

www.oscer.ou.edu