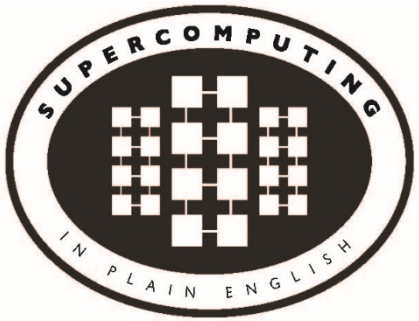
Supercomputing in Plain English

High Throughput Computing

Henry Neeman, University of Oklahoma
Director, OU Supercomputing Center for Education & Research (OSCER)
Assistant Vice President, Information Technology – Research Strategy Advisor
Associate Professor, Gallogly College of Engineering
Adjunct Associate Professor, School of Computer Science
Tuesday April 17 2018



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 phone bridge to fall back on.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF

No matter how you connect, **PLEASE MUTE YOURSELF**, so that we cannot hear you.

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:

supercomputinginplainenglish@gmail.com

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

Download the Slides Beforehand

Before the start of the session, please download the slides from the Supercomputing in Plain English website:

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

That way, if anything goes wrong, you can still follow along with just audio.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

Zoom

Go to:

http://zoom.us/j/979158478

Many thanks Eddie Huebsch, OU CIO, for providing this.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

YouTube

You can watch from a Windows, MacOS or Linux laptop or an Android or iOS handheld using YouTube.

Go to YouTube via your preferred web browser or app, and then search for:

`Supercomputing InPlainEnglish`

(**`InPlainEnglish`** is all one word.)

Many thanks to Skyler Donahue of OneNet for providing this.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

Twitch

You can watch from a Windows, MacOS or Linux laptop or an Android or iOS handheld using Twitch.

Go to:

http://www.twitch.tv/sipe2018

Many thanks to Skyler Donahue of OneNet for providing this.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

Wowza #1

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

http://jwplayer.onenet.net/streams/sipe.html

If that URL fails, then go to:

http://jwplayer.onenet.net/streams/sipebackup.html

Many thanks to Skyler Donahue of OneNet for providing this.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

Wowza #2

Wowza has been tested on multiple browsers on each of:

- Windows 10: IE, Firefox, Chrome, Opera, Safari
- MacOS: Safari, Firefox
- Linux: Firefox, Opera

We've also successfully tested it via apps on devices with:

- Android
- iOS

Many thanks to Skyler Donahue of OneNet for providing this.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

Toll Free Phone Bridge

IF ALL ELSE FAILS, you can use our US TOLL phone bridge:

405-325-6688

684 684 #

NOTE: This is for **US** call-ins **ONLY**.

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 1000 participants.

Many thanks to OU CIO Eddie Huebsch for providing the phone bridge..

Please Mute Yourself

No matter how you connect, **PLEASE MUTE YOURSELF**, so that we cannot hear you.

(For YouTube, Twitch and 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.

PLEASE MUTE YOURSELF.

Questions via E-mail Only

Ask questions by sending e-mail to:

supercomputinginplainenglish@gmail.com

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

DON'T USE CHAT OR VOICE FOR QUESTIONS!

No one will be monitoring any of the chats, and if we can hear your question, you're creating an **echo cancellation** problem.

PLEASE MUTE YOURSELF.
PLEASE MUTE YOURSELF.

Onsite: Talent Release Form

If you're attending onsite, you **MUST** do one of the following:

- complete and sign the Talent Release Form,

OR

- sit behind the cameras (where you can't be seen) and don't talk at all.

If you aren't onsite, then **PLEASE MUTE YOURSELF.**

TENTATIVE Schedule

Tue Jan 23: Storage: What the Heck is Supercomputing?
Tue Jan 30: The Tyranny of the Storage Hierarchy Part I
Tue Feb 6: The Tyranny of the Storage Hierarchy Part II
Tue Feb 13: Instruction Level Parallelism
Tue Feb 20: Stupid Compiler Tricks
Tue Feb 27: Multicore Multithreading
Tue March 6: Distributed Multiprocessing
Tue March 13: **NO SESSION** (Henry business travel)
Tue March 20: **NO SESSION** (OU's Spring Break)
Tue March 27: Applications and Types of Parallelism
Tue Apr 3: Multicore Madness
Tue Apr 10: **NO SESSION** (Henry business travel)
Tue Apr 17: High Throughput Computing
Tue Apr 24: GPGPU: Number Crunching in Your Graphics Card
Tue May 1: Grab Bag: Scientific Libraries, I/O Libraries, Visualization

Thanks for helping!

- OU IT
  - OSCER operations staff (Dave Akin, Patrick Calhoun, Kali McLennan, Jason Speckman, Brett Zimmerman)
  - OSCER Research Computing Facilitators (Jim Ferguson, Horst Severini)
  - Debi Gentis, OSCER Coordinator
  - Kyle Dudgeon, OSCER Manager of Operations
  - Ashish Pai, Managing Director for Research IT Services
  - The OU IT network team
  - OU CIO Eddie Huebsch
- OneNet: Skyler Donahue
- Oklahoma State U: Dana Brunson

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 phone bridge to fall back on.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

Coming in 2018!

- Coalition for Advancing Digital Research & Education (CADRE) Conference: Apr 17-18 2018 @ Oklahoma State U, Stillwater OK USA
  https://hpcc.okstate.edu/cadre-conference
- Linux Clusters Institute workshops
  http://www.linuxclustersinstitute.org/workshops/
  - Introductory HPC Cluster System Administration: May 14-18 2018 @ U Nebraska, Lincoln NE USA
  - Intermediate HPC Cluster System Administration: Aug 13-17 2018 @ Yale U, New Haven CT USA
- Great Plains Network Annual Meeting: details coming soon
- Advanced Cyberinfrastructure Research & Education Facilitators (ACI-REF) Virtual Residency Aug 5-10 2018, U Oklahoma, Norman OK USA
- PEARC 2018, July 22-27, Pittsburgh PA USA
  https://www.pearc18.pearc.org/
- IEEE Cluster 2018, Sep 10-13, Belfast UK
  https://cluster2018.github.io
- **OKLAHOMA SUPERCOMPUTING SYMPOSIUM 2018, Sep 25-26 2018 @ OU**
- SC18 supercomputing conference, Nov 11-16 2018, Dallas TX USA
  http://sc18.supercomputing.org/

Outline

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

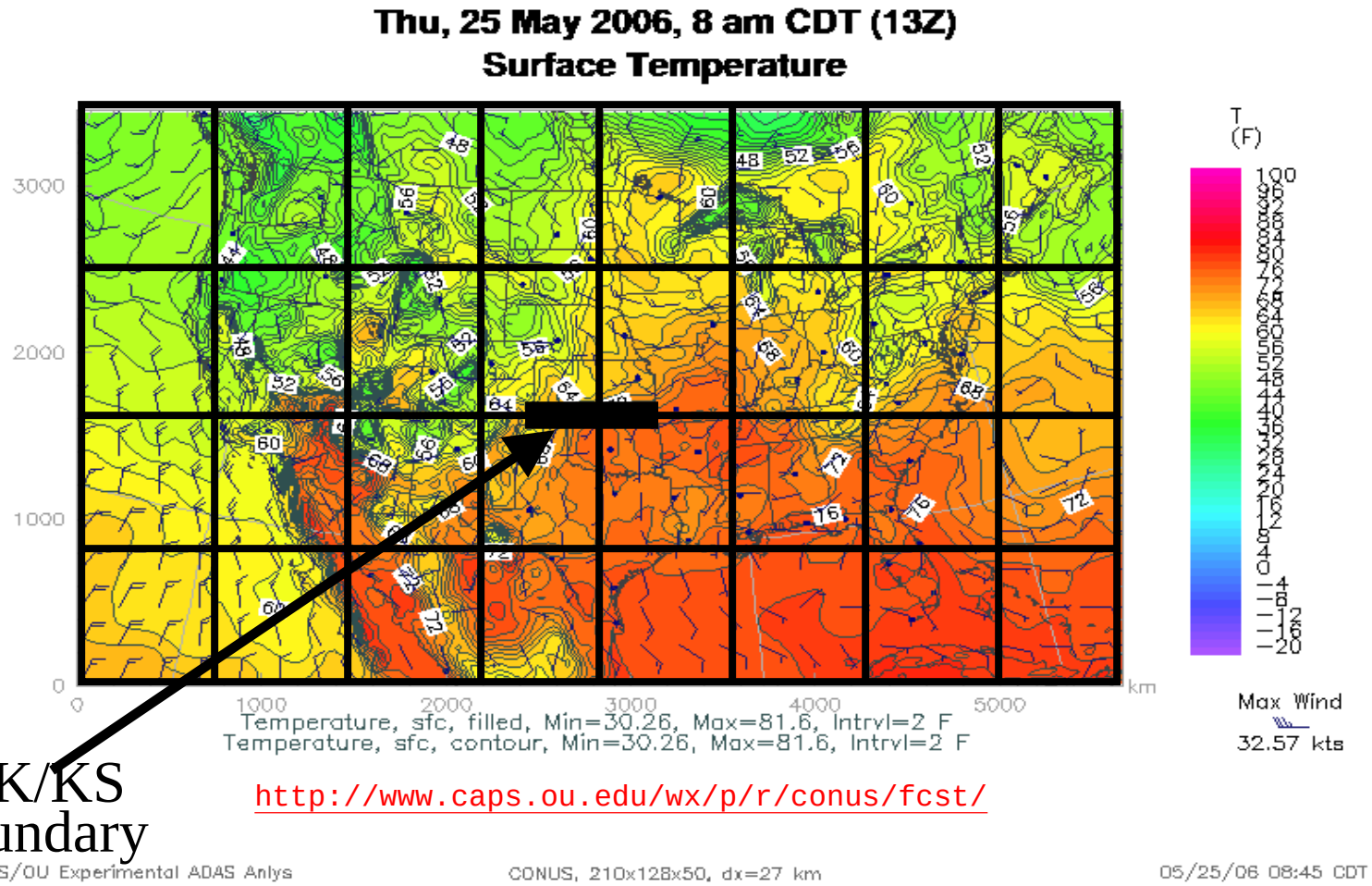
Consider weather forecasting.

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



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.

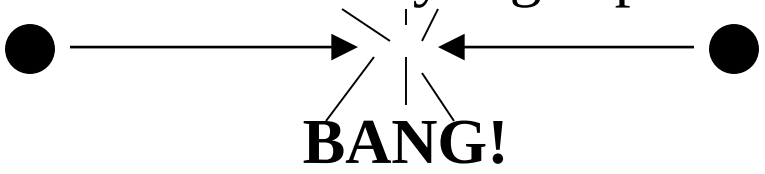
People gamble there; that is, they play games of chance, which involve randomness.

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.



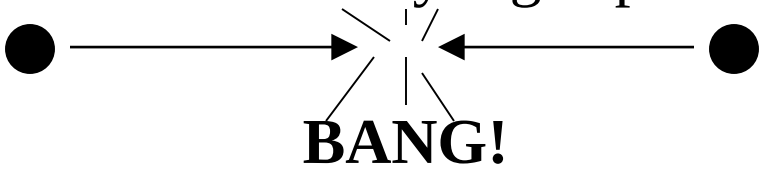
We want to know, say, the average properties of this phenomenon.

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.

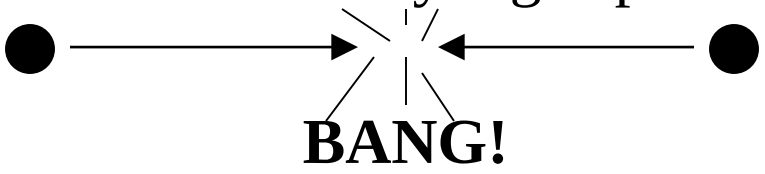
There are infinitely many ways that two particles can be banged together.

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

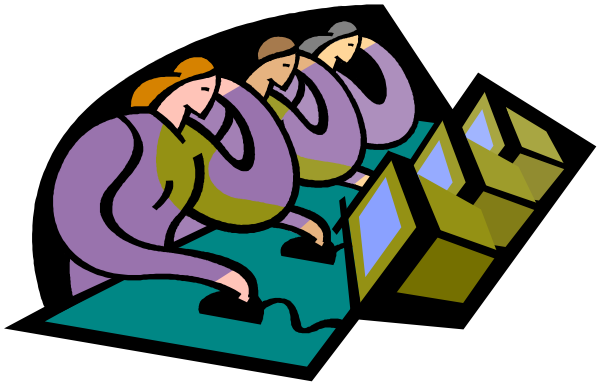
Suppose you have an existing serial Monte Carlo simulation:

```
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

Many institutions have 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 the 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**:

- WorldCommunityGrid, Rosetta@Home (curing human diseases)
- Einstein@Home (Laser Interferometer Gravitational wave Observatory)
- MilkyWay@Home (gravitational potential)
- folding@home (protein folding)
- PrimeGrid
- ClimatePrediction
- …

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 */
```

BOINC: a Big Supercomputer

- As of yesterday (Mon Apr 16 2018), according to the BOINC website, the following were active:
  - 168,847 volunteers
  - 834,604 computers
  - 24-hour average: 10.115 PetaFLOPS
    - If this were a single machine, on the most recent Top 500 list, it'd be the 11th fastest supercomputer in the world.
- Note to self: If your science isn't easy to fund, do amazing computing, so someone will fund that ….

HTCondor

https://research.cs.wisc.edu/htcondor/

HTCondor is Like BOINC

- HTCondor **steals computing time** on existing desktop PCs **when they're idle**.
- HTCondor **runs in background** when no one is sitting at the desk.
- HTCondor 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.

HTCondor is Different from BOINC

- To use HTCondor, **you don't need to rewrite your software** to add calls to special routines; in BOINC, you do.
- HTCondor **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 HTCondor** on your own personal desktop PC; it's straightforward to install a BOINC application such as SETI@home.

Useful Features of HTCondor

- **Opportunistic** computing: HTCondor steals time on existing desktop PCs when they're otherwise not in use.
- HTCondor **doesn't require any changes to the software**.
- HTCondor can **automatically checkpoint** a running job: Every so often, HTCondor 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, HTCondor can **preempt** running jobs if more important jobs come along, or if someone sits down at the desktop PC.
- Likewise, HTCondor can **migrate** running jobs to other PCs, if someone sits at the PC or if the PC crashes.
- And, HTCondor 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.

HTCondor Pool @ Your Institution

You can deploy an HTCondor pool at your institution!

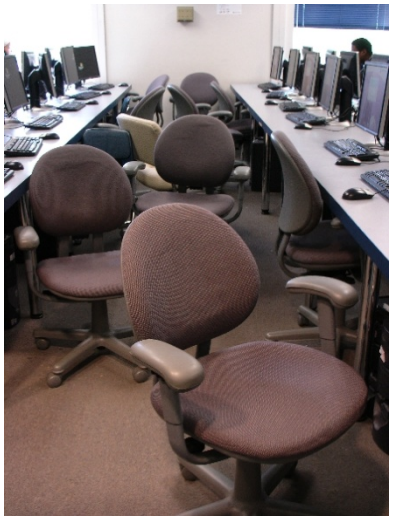
This can provide a significant amount of computing capacity.

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

When we did this at OU, we had been seeing empirically that lab PCs were available for HTCondor jobs about 80% of the time.

So doing this increased bang-for-the-buck by 5x.

We stopped doing this when we realized that the lab team was overwhelmed and didn't have time to maintain HTCondor.



HTCondor Limitations

- The Unix/Linux version has **more features** than Windows, which is referred to as "clipped."
- The MacOS version runs natively on BSD Unix, but isn't mature.
- Under Windows, you can't checkpoint your jobs (though you can pause them).
- 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/CentOS Linux HTCondor pool, you have to be able to **compile your code** with gcc, g++, g77 or gfortran.
- Also, depending on the PCs that have HTCondor on them, you may have limitations on, for example, how big your jobs' RAM footprint can be.

Running a HTCondor Job

Running a job on HTCondor 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 HTCondor system, which decides when and where your job runs, magically and invisibly.

Sample HTCondor Batch Script

```
Universe     = standard
Executable   = /home/hneeman/NBody/nbody_compiled_for_HTCondor
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`

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

HTCondor is Grid Computing

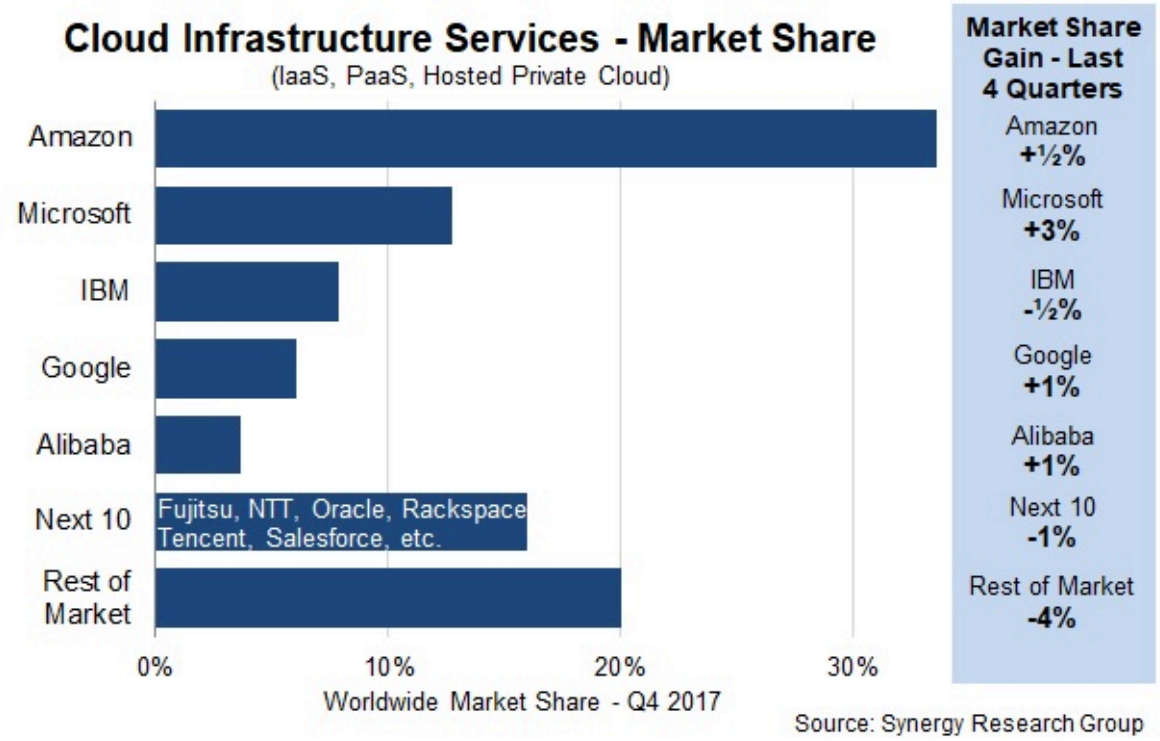
HTCondor creates a grid out of disparate desktop PCs.

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

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.

Cloud Computing

- The term ***cloud computing*** is often used interchangeably with grid computing.
- Mostly, it's come to mean commercial clouds:



https://www.sdxcentral.com/articles/news/microsoft-azure-revenues-grew-98-in-latest-quarter/2018/02/

Cloud Computing is Big Business

- "AWS had $17.459 billion in sales in 2017, up 43% from $12.219 billion in sales in 2016. Operating income for AWS was up 39% from $3.1 billion in 2016 to $4.3 billion in 2017 [+24.6%]."
  http://www.businessinsider.com/amazon-web-services-2017-revenue-2018-2
- "AWS accounted for 73% of [Amazon's] net income [in 2017Q4]."
  http://www.businessinsider.com/amazon-fourth-quarter-2017-earnings-2018-2
- Microsoft: "Azure up 98%, Surface up 1%, and Windows up 4%"
  https://venturebeat.com/2018/01/31/microsoft-reports-28-9-billion-in-q2-2018-revenue-azure-up-98-surface-up-1-and-windows-up-4/
- "IBM posted robust cloud-revenue growth for calendar 2017 of $17 billion, up 24%, with even stronger cloud growth in the fourth quarter with $5.5 billion in cloud revenue, up 30%."
  https://www.forbes.com/sites/bobevans1/2018/01/26/amazon-to-become-1-in-cloud-computing-revenue-by-beating-ibms-17-billion/#31eea59a6b3e

TENTATIVE Schedule

Tue Jan 23: Storage: What the Heck is Supercomputing?
Tue Jan 30: The Tyranny of the Storage Hierarchy Part I
Tue Feb 6: The Tyranny of the Storage Hierarchy Part II
Tue Feb 13: Instruction Level Parallelism
Tue Feb 20: Stupid Compiler Tricks
Tue Feb 27: Multicore Multithreading
Tue March 6: Distributed Multiprocessing
Tue March 13: **NO SESSION** (Henry business travel)
Tue March 20: **NO SESSION** (OU's Spring Break)
Tue March 27: Applications and Types of Parallelism
Tue Apr 3: Multicore Madness
Tue Apr 10: **NO SESSION** (Henry business travel)
Tue Apr 17: High Throughput Computing
Tue Apr 24: GPGPU: Number Crunching in Your Graphics Card
Tue May 1: Grab Bag: Scientific Libraries, I/O Libraries, Visualization

Thanks for helping!

- OU IT
  - OSCER operations staff (Dave Akin, Patrick Calhoun, Kali McLennan, Jason Speckman, Brett Zimmerman)
  - OSCER Research Computing Facilitators (Jim Ferguson, Horst Severini)
  - Debi Gentis, OSCER Coordinator
  - Kyle Dudgeon, OSCER Manager of Operations
  - Ashish Pai, Managing Director for Research IT Services
  - The OU IT network team
  - OU CIO Eddie Huebsch
- OneNet: Skyler Donahue
- Oklahoma State U: Dana Brunson

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 phone bridge to fall back on.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

PLEASE MUTE YOURSELF.

Coming in 2018!

- Coalition for Advancing Digital Research & Education (CADRE) Conference: Apr 17-18 2018 @ Oklahoma State U, Stillwater OK USA
  https://hpcc.okstate.edu/cadre-conference
- Linux Clusters Institute workshops
  http://www.linuxclustersinstitute.org/workshops/
  - Introductory HPC Cluster System Administration: May 14-18 2018 @ U Nebraska, Lincoln NE USA
  - Intermediate HPC Cluster System Administration: Aug 13-17 2018 @ Yale U, New Haven CT USA
- Great Plains Network Annual Meeting: details coming soon
- Advanced Cyberinfrastructure Research & Education Facilitators (ACI-REF) Virtual Residency Aug 5-10 2018, U Oklahoma, Norman OK USA
- PEARC 2018, July 22-27, Pittsburgh PA USA
  https://www.pearc18.pearc.org/
- IEEE Cluster 2018, Sep 10-13, Belfast UK
  https://cluster2018.github.io
- **OKLAHOMA SUPERCOMPUTING SYMPOSIUM 2018, Sep 25-26 2018 @ OU**
- SC18 supercomputing conference, Nov 11-16 2018, Dallas TX USA
  http://sc18.supercomputing.org/

Thanks for your attention!

Questions?

www.oscer.ou.edu