CTA200
Computational Astrophysics

Harald Pfeiffer

Today:

1. Introduction
2. CITA's Computing Network
3. Command Line 101
4. Remote connectivity
5. Version control systems
6. Emacs

Goals of this course

- Introduce a variety of software tools that are essential for effective use of computing in science
  - Various useful programs and techniques
  - Scripting to automate recurring tasks (shell, python)
  - Visualization (2-D and 3-D)

- Survey of some of the computational techniques in astro-physics
  - Cosmic microwave background analysis
  - Galaxy colissions (N-body codes)
  - Black hole simulations
  - ...

Structure

- Course runs May 7-17
- 9:30-noon: Lectures
  - with breaks
  - questions/discussions encouraged -- you learn more, and it's more fun
- Homework assignments (practical focus)
  - May 8 due May 9
  - May 9 due May 11
  - May 11 due May 14
  - May 14 due May 17
- One larger project
  - begins next week, due May 24 or May 31 (tbd)

Audience

- All interested U of T students
- Summer research students located at
  - Canadian Institute of Theoretical Astrophysics (CITA)
  - Department of Astronomy
  - Dunlap Institute

- This is a formal U of T course (CTA200H1)
  - U of T students can enroll to receive credit
    - Instructor enrolls students manually -- contact me
  - non U of T summer research students very welcome to audit

Level & prerequisites

- No formal requirements
- Useful to have some programming experience
  - e.g. Python
- Focus is on breadth, not exhaustive depth
  - Introduce tools
  - Show their uses
  - Explain where to find more information
  - *... so you remember when you have a certain task and know where to look*
- New course
  - Adjustments likely as we go along
  - Your feedback is important: Tell me (as soon as possible) what works for you, and what doesn't. I *do* want to know

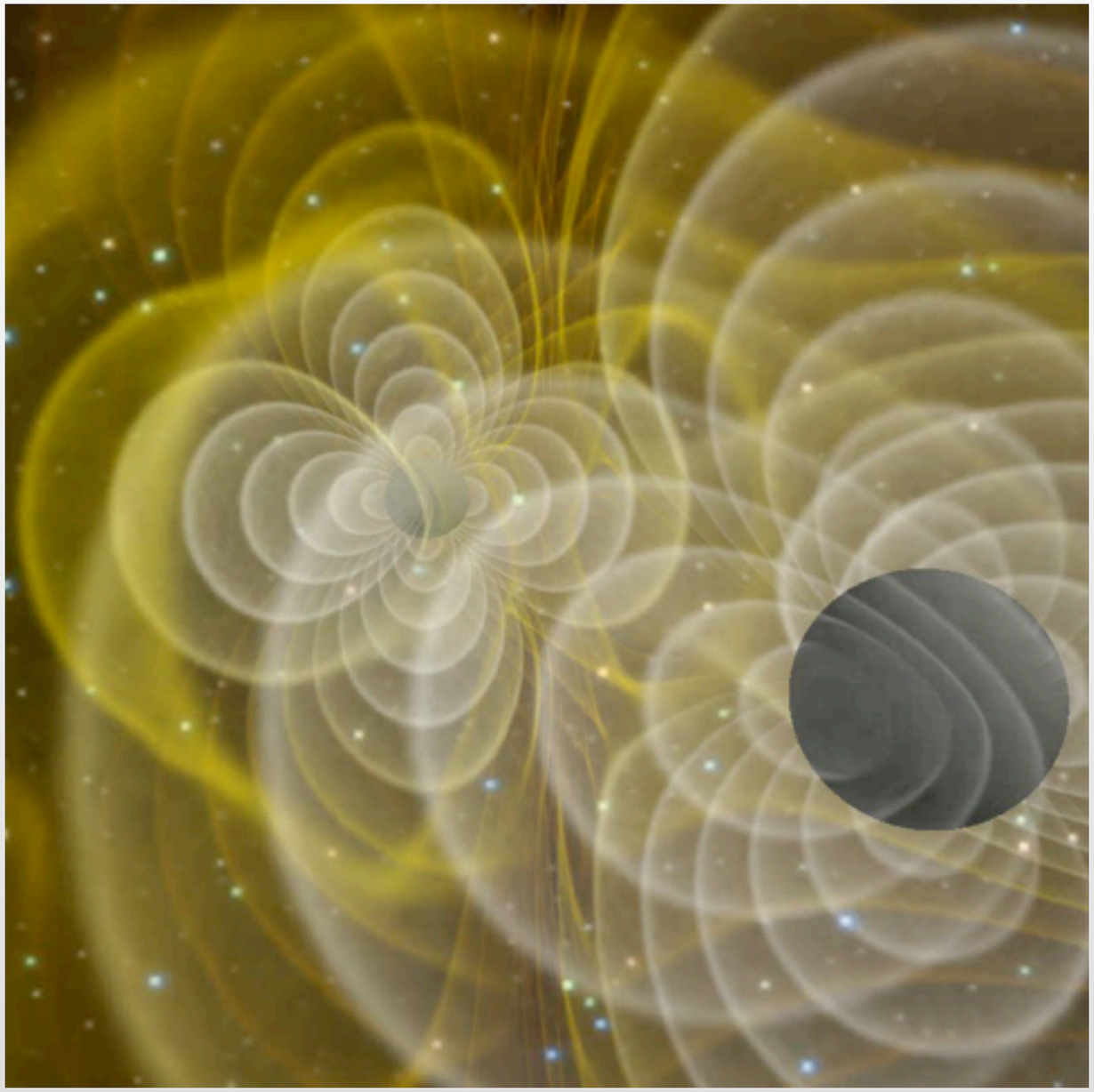
Your instructor

CITA–ICAT

- Harald Pfeiffer
  - pfeiffer@cita.utoronto.ca
  - MP1309 (diagonally across this floor)

- Research area: Simulate colliding black holes

The two-body problem



(Courtesy J. Centrella, Goddard)

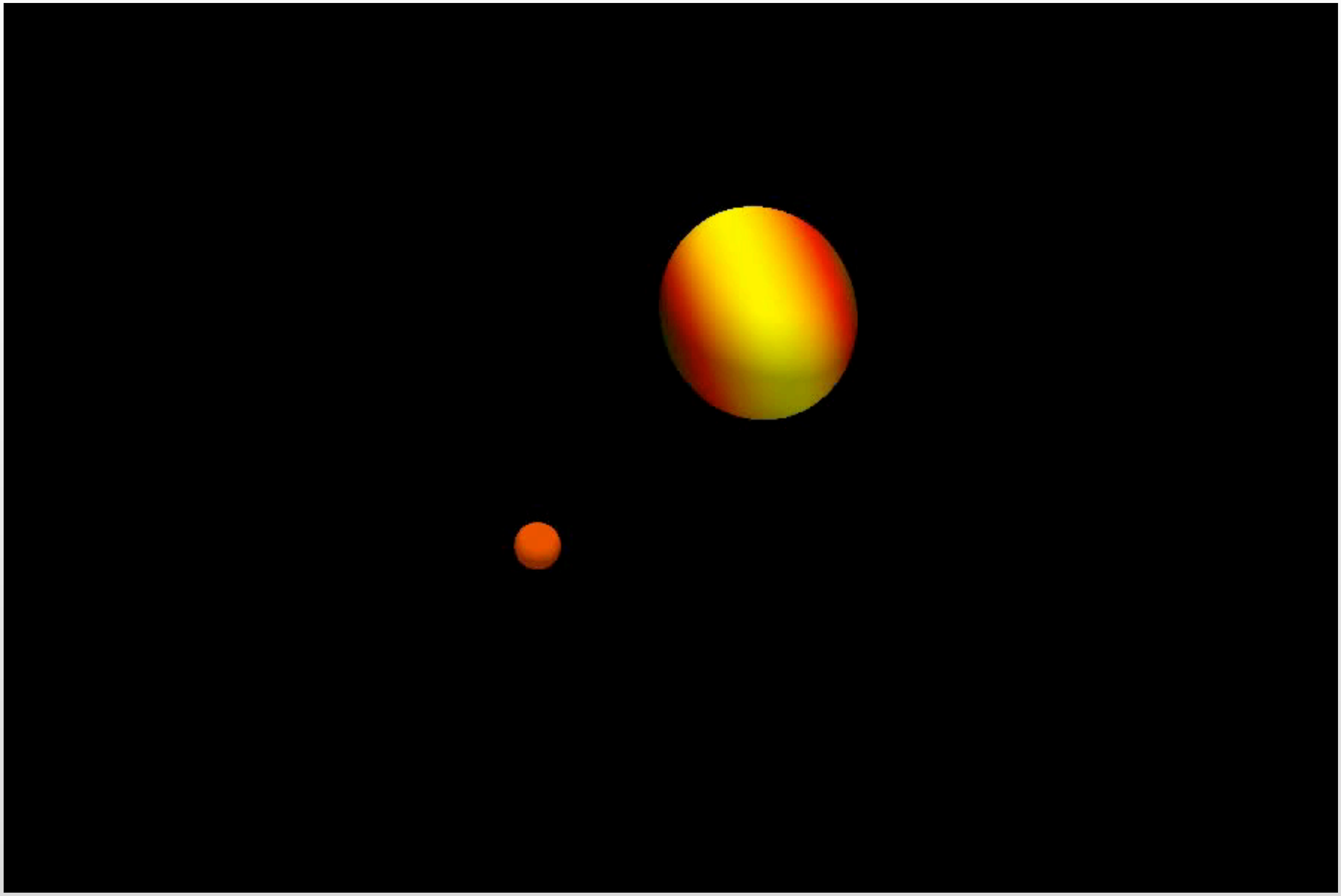
$$L_{\rm max} = 10^{23} L_\odot \sim L_{\rm universe}$$

- at distance 10Gpc

$$\Phi_{\rm max} \sim 10^4 \Phi_{\rm Moon}$$

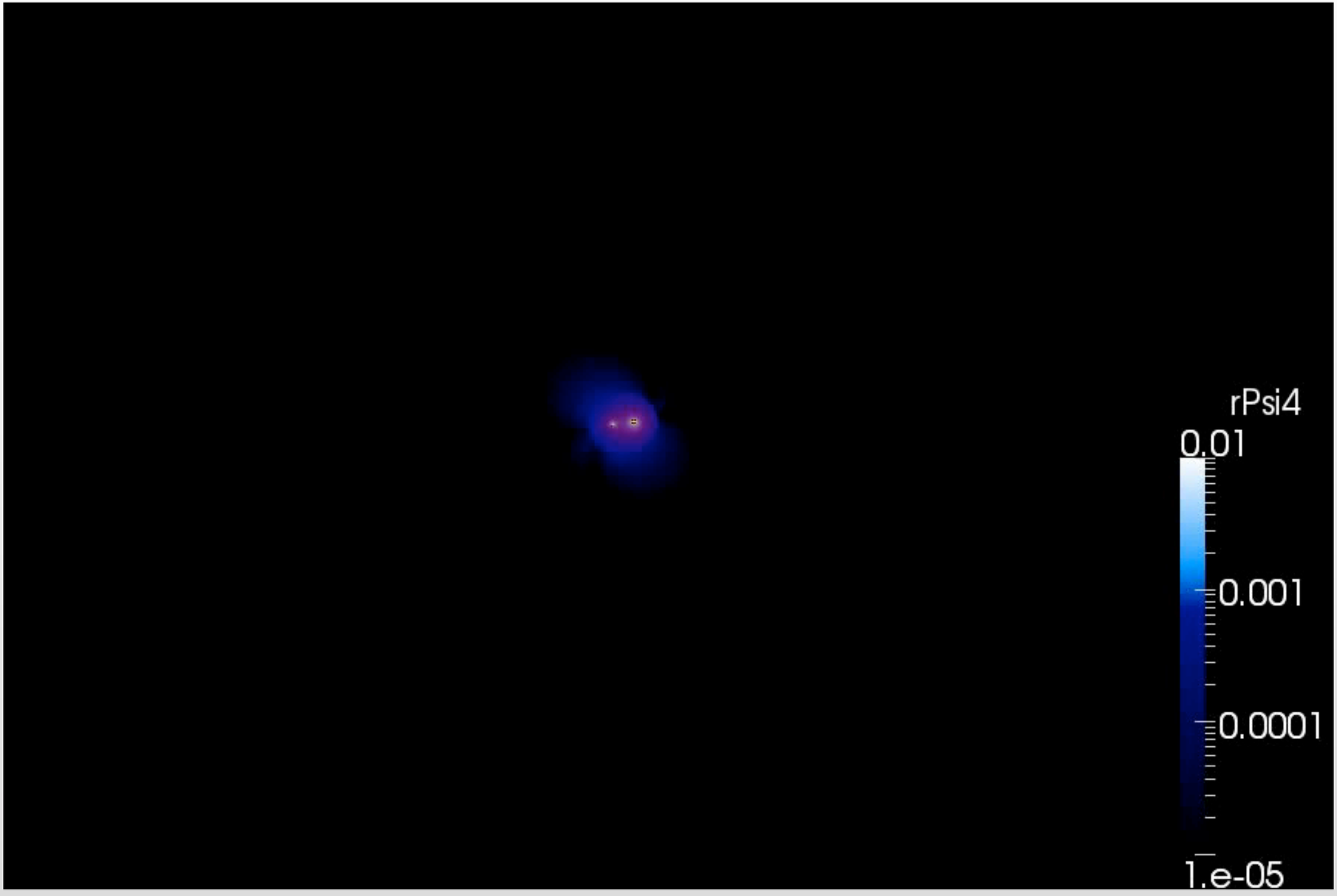
- No electro-magnetic emission
  - gravitational radiation

Large BH spinning (note precession)



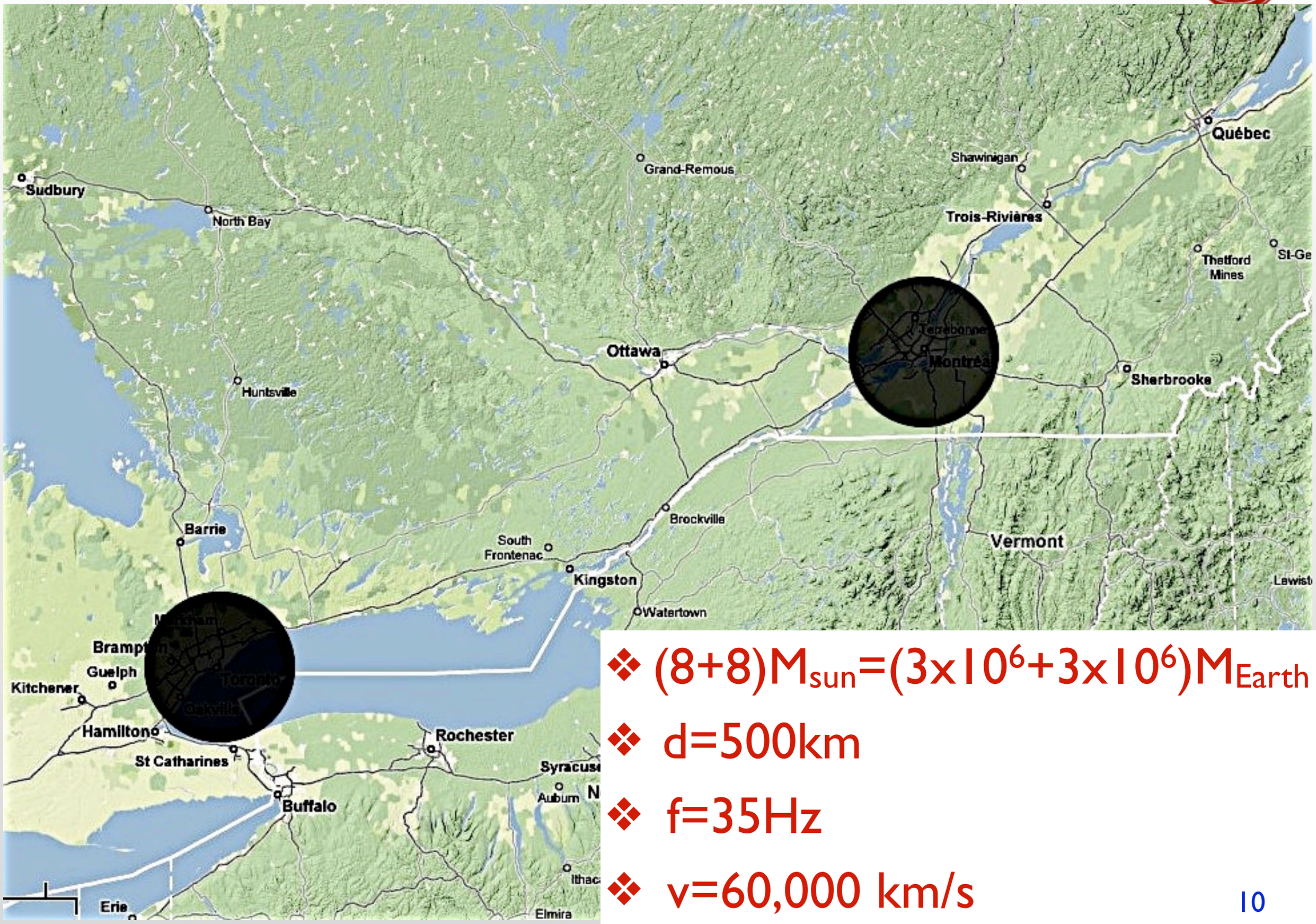
Movie by U of T undergraduate Patrick Fraser

One more movie...



Movie by U of T undergraduate Patrick Fraser

Dimensions



- $(8+8)M_{sun}=(3x10^6+3x10^6)M_{Earth}$
- d=500km
- f=35Hz
- v=60,000 km/s

Gravitational wave detectors

VIRGO (Italy+France)

LIGO (USA)

GEO (Germany+UK)

KAGRA in Japan funded
Probably: LIGO-India

Small Survey of you

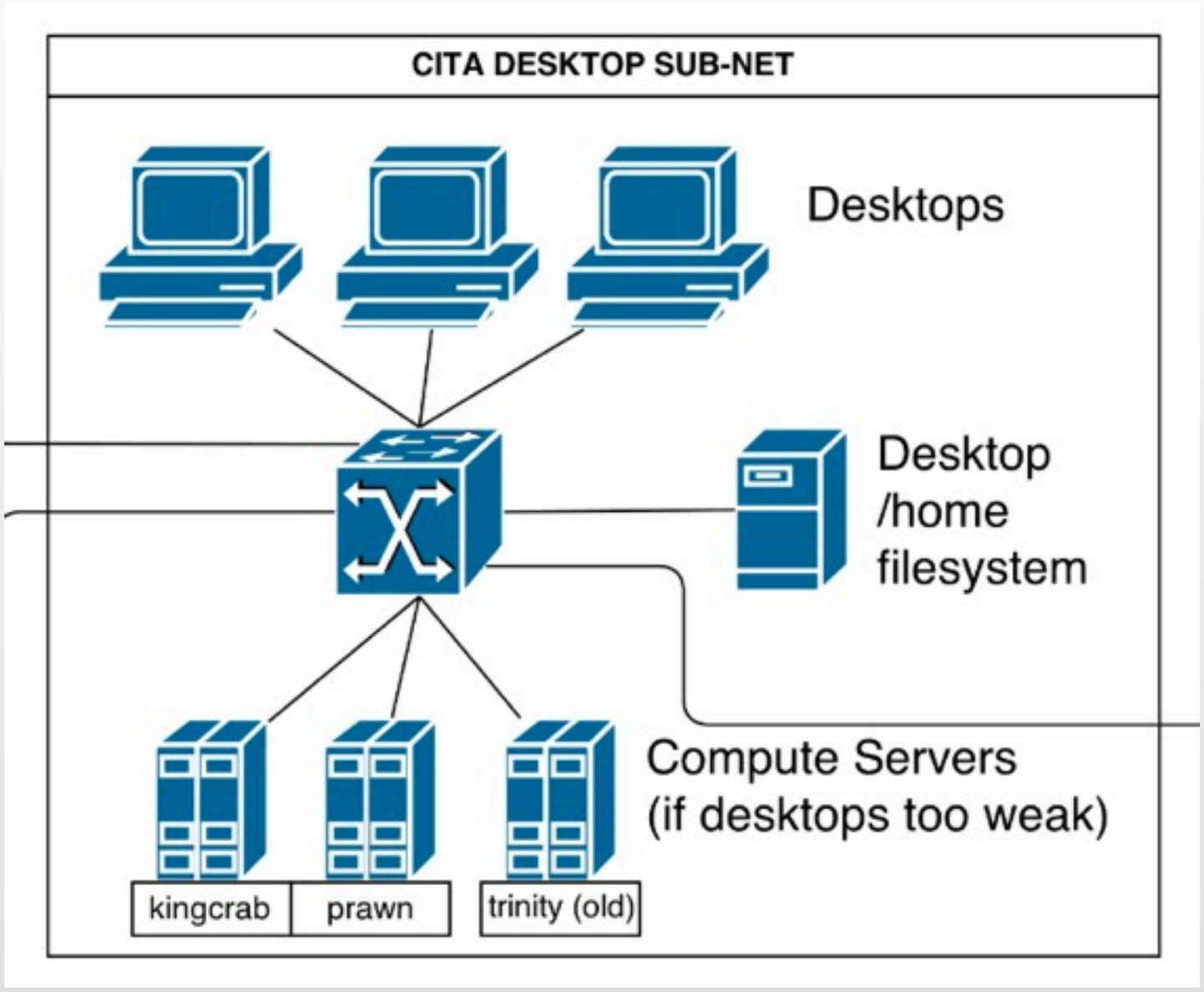
Keyboard vs. Mouse

CITA–ICAT

- GUIs are great for new users, or single use
- Serious deficiencies for scientific use
  - Working in GUIs tends to be slow
  - Difficult to automate
  - Remote work difficult
    - Long response-time to a machine elsewhere
    - The other machine may not have your favorite software
  - Reproduction of results difficult/impossible
    - cut-and-paste numbers into spreadsheet
    - manually process data & create graph (e.g. matlab or python)
    - ***Every*** *computation and data-processing should be reproducible*
- GOALS: Efficient. Robust. Reproducible.

Scientific computer networks: e.g. CITA

- Users sit at desktops or just use laptops
- *Same* /home storage accessible from all machines (i.e. doesn't matter where one sits)
- More powerful servers when desktop/laptops too weak
  - Kingcrab 16 cores, 64GB
  - Prawn 12 cores, 130GB, several GPUs

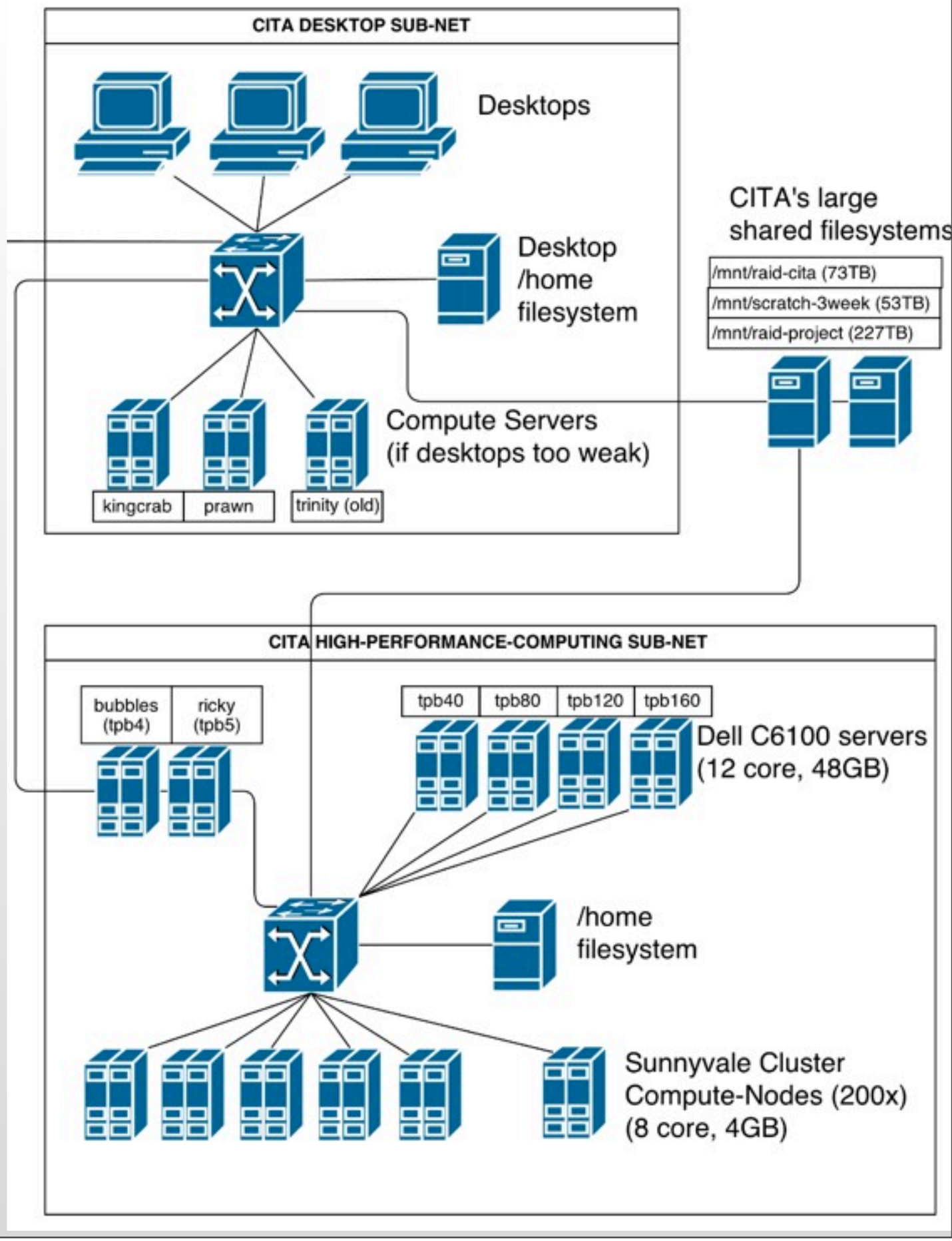


Already remote access

CITA II

- "desktop sub-net"
- "HPC sub-net"
  - the heavy computing power
  - separate /home storage
  - must access through "login nodes" (security)
    - bubbles.cita.utoronto.ca
    - ricky.cita.utoronto.ca
- Large storage-servers visible from both sub-nets

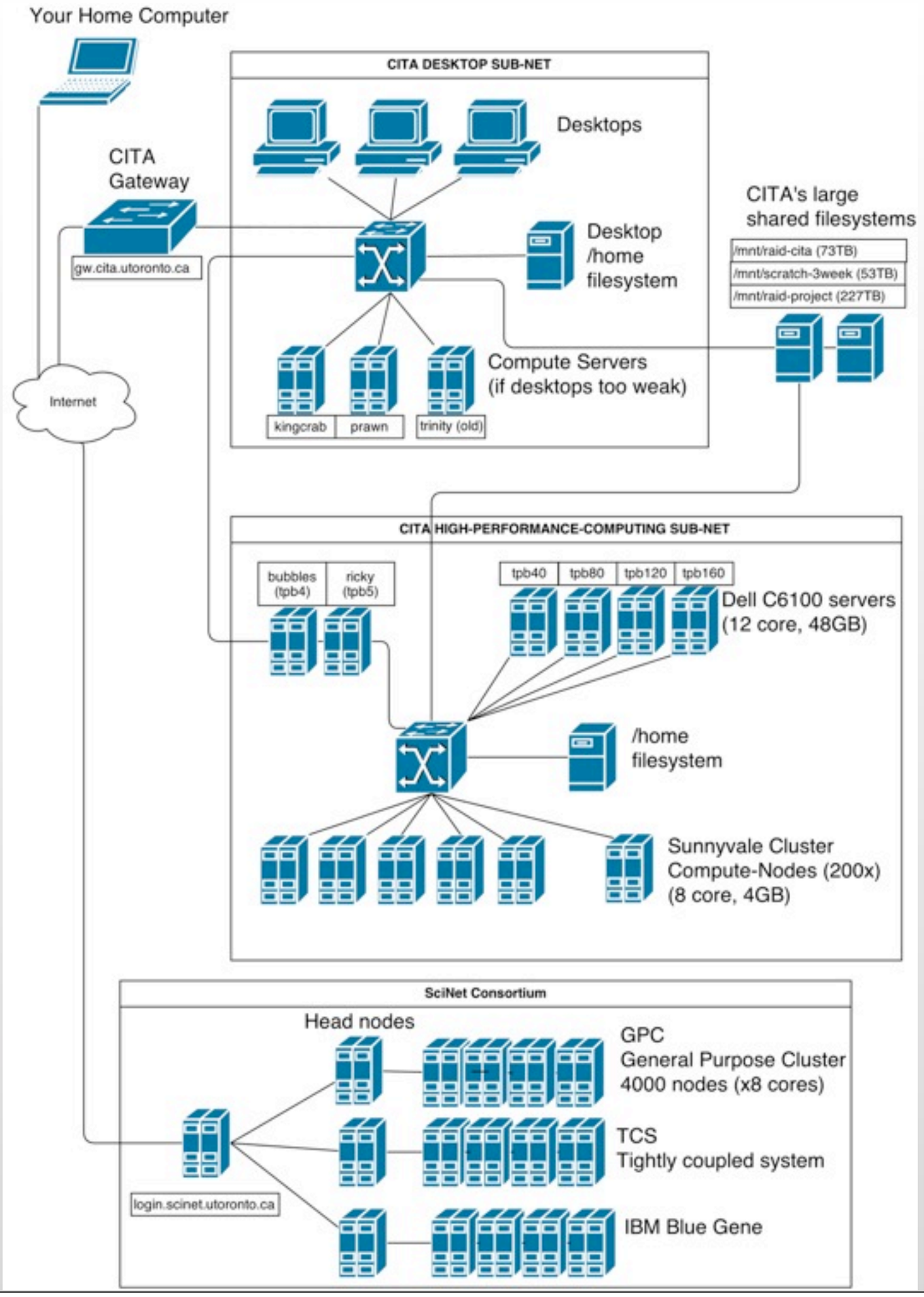
Yet more remote access separate /home directory



CITA III

- All access to CITA machines from outside must go through a "gateway" (security)
  - Example:
    your laptop
    -> gw.cita.utoronto.ca
    -> bubbles
    -> tpb160

- ComputeCanada data-centres
  - the ***very*** big guns
  - same setup:
    - first log into login-node
    - then jump to desired machine

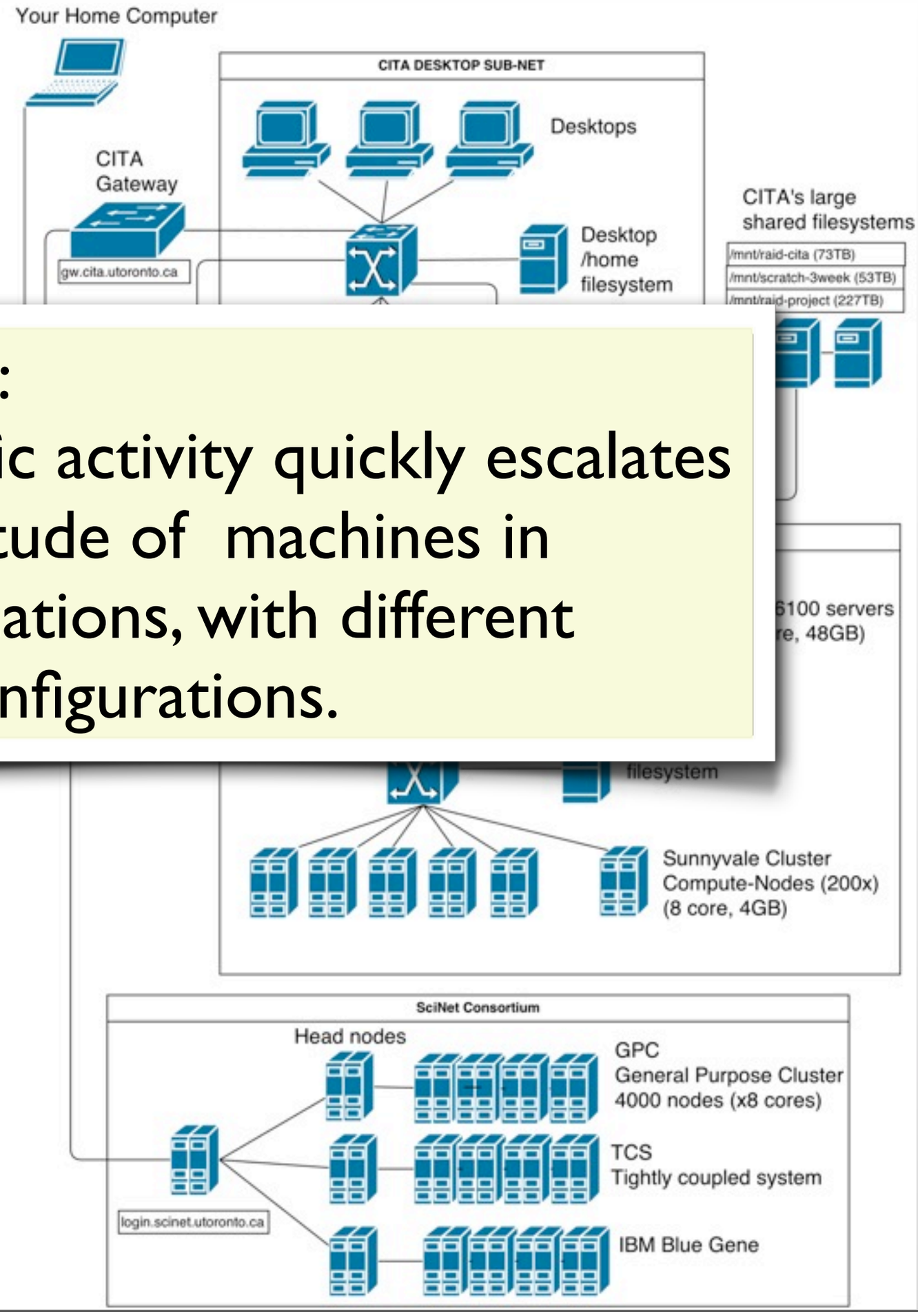


CITA III

- All access to CITA machines from outside must go through a "gateway" (sec
  - Example:
    your laptop
    -> gw.cita.uto
    -> bubbles
    -> tpb160

- ComputeCanada data-centres
  - the ***very*** big guns
  - same setup:
    - first log into login-node
    - then jump to desired machine

Bottom line:
Any scientific activity quickly escalates into a multitude of machines in different locations, with different software configurations.



Unix/Linux command-line

- The interface to all your computer's powers
- Works by typing in commands

```
pfeiffer@lenovo-07c199e5:~/Documents/MyTalks/2013 — ⌘6
pfeiffer@lenovo-...ts/MyTalks/2013
13Mar_Maryland.key/     13Mar_SxsVideoConf.key/
lenovo-07c199e5 MyTalks/2013 $ open 13Mar_SxsVideoConf.key/
lenovo-07c199e5 MyTalks/2013 $ open ~/research/
.DS_Store           Papers/             SpEC/
.repositories.txt   Projects/           Talks/
Etc/                Propsals/           Writing/
lenovo-07c199e5 MyTalks/2013 $ open ~/research/Projects/12
12_NR_template_bank/ 12_Samurai2/
lenovo-07c199e5 MyTalks/2013 $ open ~/research/Projects/13
13_100runs/ 13_Inspire/
lenovo-07c199e5 MyTalks/2013 $ open ~/research/Projects/13_
13_100runs/ 13_Inspire/
lenovo-07c199e5 MyTalks/2013 $ open ~/research/Projects/12
12_NR_template_bank/ 12_Samurai2/
lenovo-07c199e5 MyTalks/2013 $ open ~/research/Projects/12_Samurai2/
lenovo-07c199e5 MyTalks/2013 $ open ~/research/Projects/12_Samurai2/
lenovo-07c199e5 MyTalks/2013 $
```

some basic terminal commands

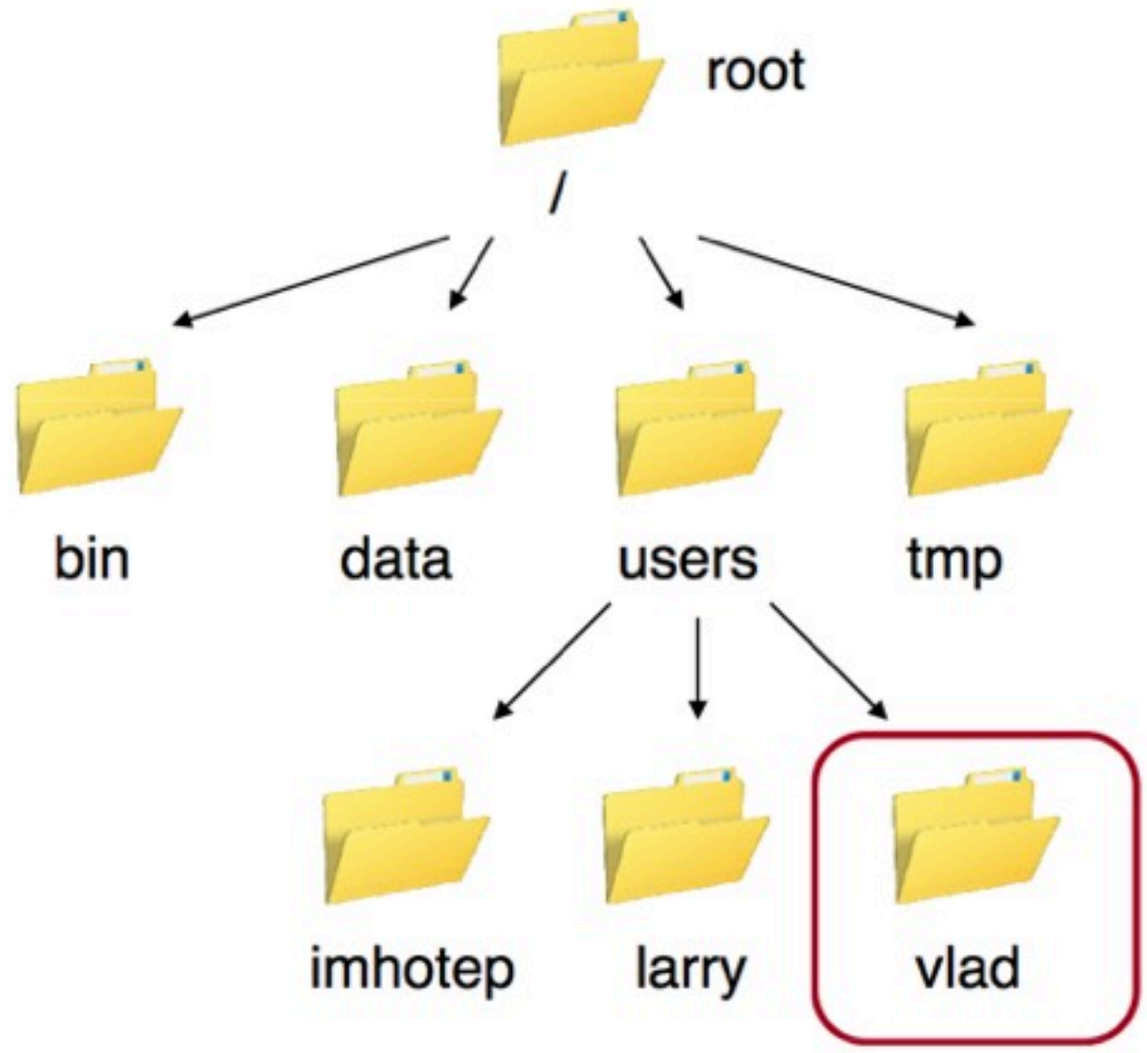
ls	"list"	show directory contents
cd	"change directory"	switch to different directory
pwd	"print working directory"	show current directory

cp	copy	copy file(s) from one directory to another
mv	move	move file(s), or rename file
rm	remove	delete file(s)

cat	concatenate	print file contents to screen
less		show file, allowing for "up/down" movement
man	manual	show information what a Linux command does, and what options it takes

- Almost all commands take the filenames to act on
  - cd /some/other/directory
  - cp file.png /some/other/directory/
  - cat Schedule.txt

```
login: vlad
password: ********
$ whoami
vlad
$ pwd
/users/vlad
$
```



Special directories

- / root directory of file-system
  - leading / indicates path starting from root
- ~ home-directory of user
  - often /User/USER or /home/USER
  - "cd ~" returns you to your home-directory
- .. parent directory
  - one level up
- no leading '/' indicates path relative to current directory

```
retina ~ $ pwd
/Users/pfeiffer
retina ~ $ cd Documents/Teaching/
retina Documents/Teaching $ pwd
/Users/pfeiffer/Documents/Teaching
retina Documents/Teaching $ cd ..
retina ~/Documents $ pwd
/Users/pfeiffer/Documents
retina ~/Documents $ cd ../research/
retina ~/research $ pwd
/Users/pfeiffer/research
retina ~/research $
```

Options

- Commands take a plethora of options to customize what they do:
  - Specified with '-' followed by the letters that determine the option

pfeiffer@lenovo-07c199e5:~/Documents/Teaching/2013_CTA200/CTA200/LectureN...

pfeiffer@lenovo-...o-ssh-git-emacs

```
lenovo-07c199e5 LectureNotes/01-intro-ssh-git-emacs (master u=) $
lenovo-07c199e5 LectureNotes/01-intro-ssh-git-emacs (master u=) $ ls
CITA-Network.png CITA-Network.xml
lenovo-07c199e5 LectureNotes/01-intro-ssh-git-emacs (master u=) $ ls -l
total 816
-rw-r--r--  1 pfeiffer  staff  386615 May  6 22:51 CITA-Network.png
-rw-r--r--  1 pfeiffer  staff   25111 May  6 22:51 CITA-Network.xml
lenovo-07c199e5 LectureNotes/01-intro-ssh-git-emacs (master u=) $ ls -gh
total 816
-rw-r--r--  1 staff   378K May  6 22:51 CITA-Network.png
-rw-r--r--  1 staff    25K May  6 22:51 CITA-Network.xml
lenovo-07c199e5 LectureNotes/01-intro-ssh-git-emacs (master u=) $ ls -g -h
total 816
-rw-r--r--  1 staff   378K May  6 22:51 CITA-Network.png
-rw-r--r--  1 staff    25K May  6 22:51 CITA-Network.xml
lenovo-07c199e5 LectureNotes/01-intro-ssh-git-emacs (master u=) $
```

  - multiple options either with separate 'ls -g -h' or combined "ls -gh"

- Because there are only 52 letters, more complex commands use *long options*. Two dashes, followed by a complete word.

Finding information

- Which commands exist?
  - Somewhat hard to find out.
  - Set of commands fairly small.
  - I'll provide lists of useful commands.
  - Watch others type. Google. Read a book.
- What options do commands take?
  - "man" command displays help text
    - "man ls" lists the manual for the 'ls' command
- Which sequence of commands to use to accomplish 'XXX'?
  - Now THAT's an interesting question!
  - We'll do lots of this later this week. Pay attention
  - Watch others type (e.g. summer students: watch your advisors)

Trivia

- The program that reads and processes the typed input is called "**shell**"

- There are several different shells
  - most widely used are bash, tcsh
  - They differ in arcane issues with customization, scripting, and variables
  - **Use bash**, unless you have a reason to do otherwise

Remote Execution

Secure Shell ssh

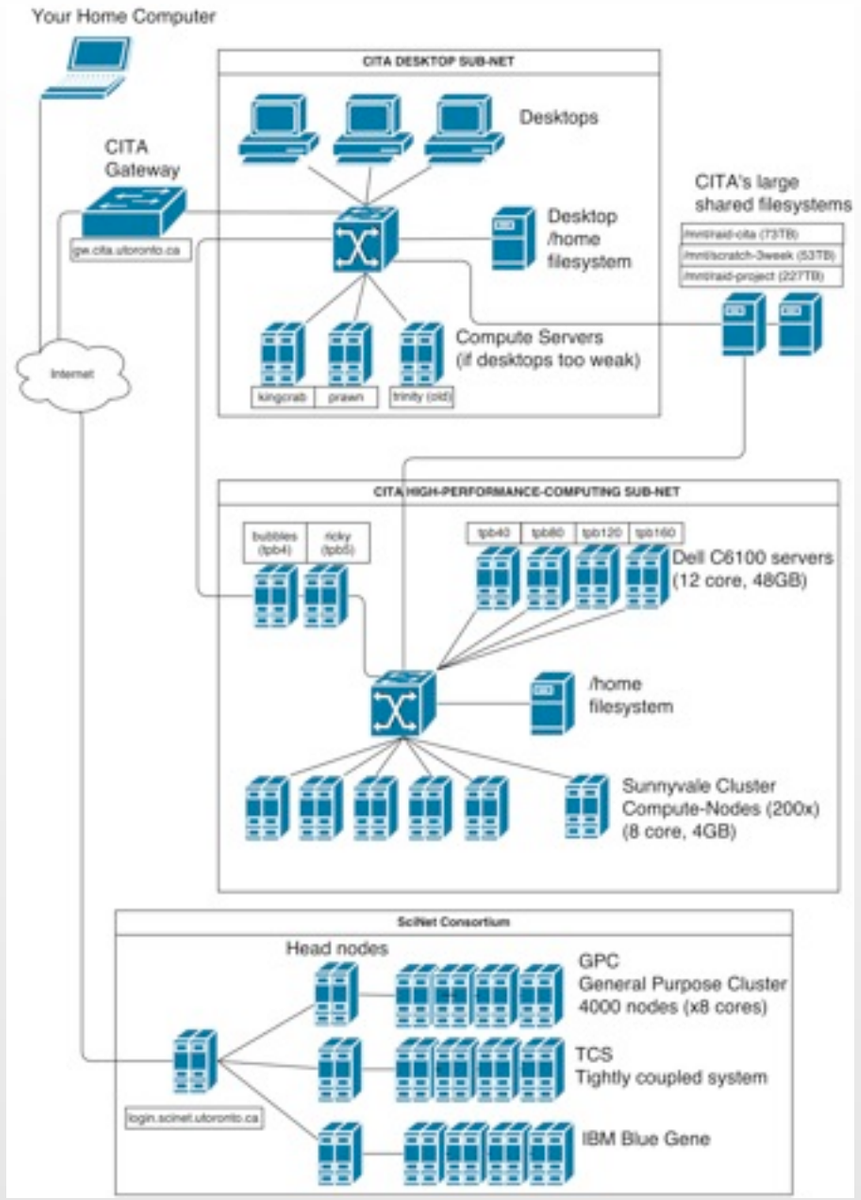
- "ssh" allows to connect to other Linux/Unix machines
  - E.g. from home:

```
pfeiffer@kingcrab:~ — ⌘2
pfeiffer@kingcrab:~
(wd now: ~)
lenovo-07c199e5 ~ $ pwd
/Users/pfeiffer
lenovo-07c199e5 ~ $ ssh gw.cita.utoronto.ca
Welcome to CITA remote gateway!

 * * * * * * * * * * * N O T I C E * * * * * * * * * * *
If you are experiencing problems logging in:
Support Wiki: http://wiki.cita.utoronto.ca/index.php/FAQ
Support Email: requests@cita.utoronto.ca
Alternate Gateway: gw2.cita.utoronto.ca

 * * * * * * * * * * * W A R N I N G * * * * * * * * * * *
THIS SYSTEM IS RESTRICTED TO AUTHORIZED USERS FOR AUTHORIZED USE
ONLY. UNAUTHORIZED ACCESS IS STRICTLY PROHIBITED AND MAY BE
PUNISHABLE UNDER THE COMPUTER FRAUD AND ABUSE ACT OF 1986 OR
OTHER APPLICABLE LAWS. IF NOT AUTHORIZED TO ACCESS THIS SYSTEM,
DISCONNECT NOW. BY CONTINUING, YOU CONSENT TO YOUR KEYSTROKES
AND DATA CONTENT BEING MONITORED. ALL PERSONS ARE HEREBY
NOTIFIED THAT THE USE OF THIS SYSTEM CONSTITUTES CONSENT TO
MONITORING AND AUDITING.

Last login: Mon May  6 22:56:45 2013 from 76.68.36.33
[pfeiffer@gw ~]$ ssh kingcrab
[pfeiffer@kingcrab ~]$
```



Note *absence* of password and username

ssh without nice setup

- You must specify username manually:
  - laptop$ ssh -L USER gw.cita.utoronto.ca
  - gw$ ssh kingcrab

- You will be asked to enter passwords

- Having to type passwords & usernames is poison. Let's set-up ssh to do this for us.

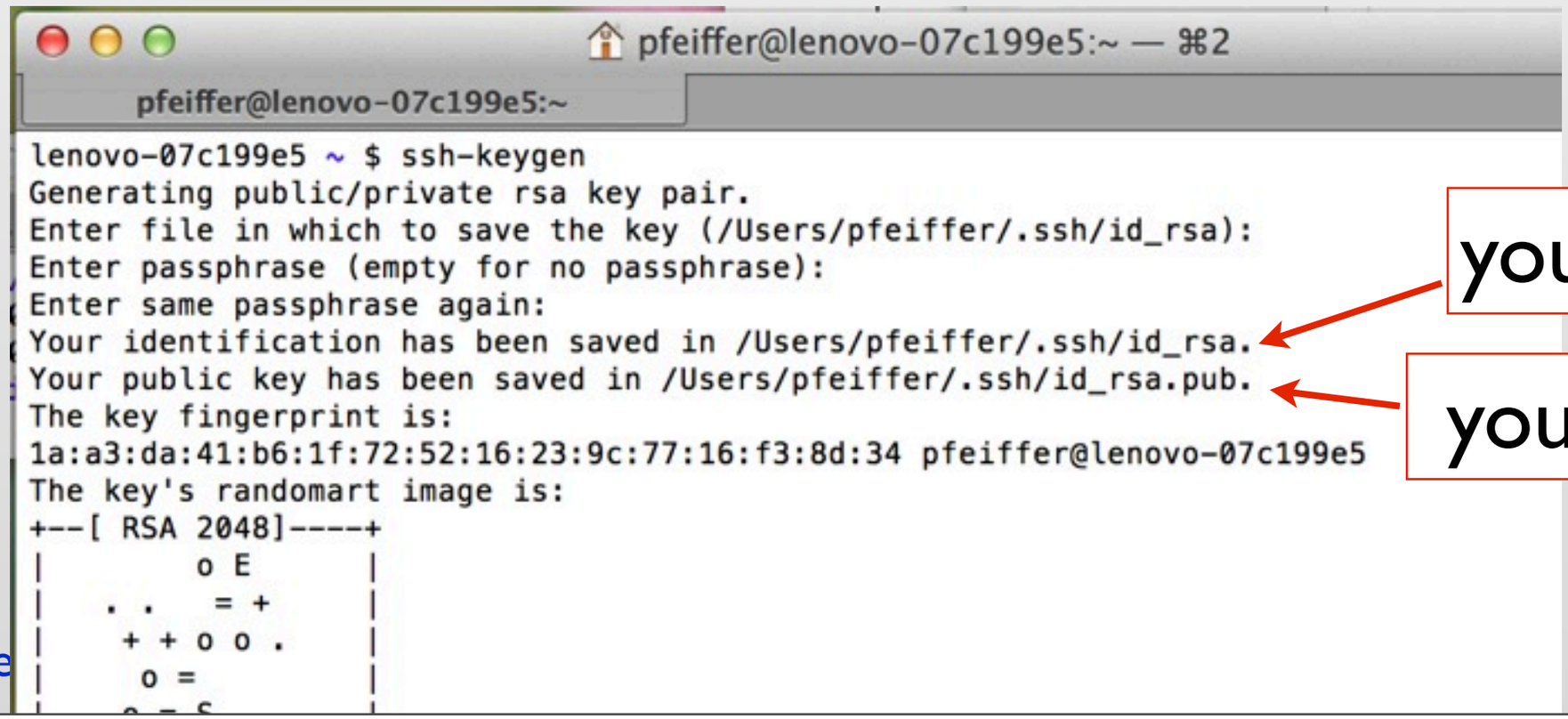
public/private key authentication

- private key ("id_rsa") is secret
  - Available only on the machine where you **type** (your laptop, desktop)
- public key ("id_rsa.pub") is not secret
  - Place public key on machine were you want to log in (let's say gw.cita)

- ssh -l USER gw.cita.utoronto.ca executes this sequence of steps:
  - Our machine connects to gw.cita.
  - gw.cita constructs a challenge, encrypts it with the public key, sends to our machine
  - Our machine uses the private key to decrypt the challenge, and sends it back.
  - Being able to decrypt the message indicates we posses the private key. Thus we are who we say we are. Allow login.

ssh setup I

- keys stored in directory ~/.ssh
  - subdirectory named .ssh in your home directory ~/
  - files starting with dot . are hidden. To show them use option -a:
    - ls -a
- Setup on the machine you **type**
  - Generate ONCE a public/private key-pair "ssh-keygen"
  - Accept defaults, but use a long passphrase.



your private key

your public key

ssh setup II

- public key needs to be copied onto target machine, and there needs to be placed into the file ~/.ssh/authorized_keys:

```
laptop$  scp .ssh/id_rsa.pub USER@gw.cita.utoronto.ca:      (1)
laptop$  ssh USER@gw.cita.utoronto.ca                       (2)
[USER@gw ~]$ cat id_rsa.pub >> .ssh/authorized_keys         (3)
```

(1) copy file to gw.cita.utoronto.ca (scp = secure copy)
--- may have to specify USER@ to indicate username at gw.cita
--- notice ':' at end of command. This indicates that the string before is a computer name.
--- requires password

(2) log into gw.cita yourself (requires password)

(3) append id_rsa.pub to the end of .ssh/authorized_keys
--- ">>" appends to the file given afterwards (more on Thursday)
--- append instead of overwrite to allow multiple public-keys

ssh setup III

- Done!
- Place public key into *every* machine where you want to log into.
- Create public/private keypair only for machines were you type
  - often only 2 private keys: One for laptop, one for desktop

Version Control

What is Version Control

- It's a system where users can commit "snapshots" of their work:
  - homework, papers being written, code being developed, data, ...

- The system keeps logs of all commits,
  and allows to retrieve any previous commit

- The system handles multiple users working on the same repository
  - User A *pushes* his changes into the repository
  - User B *pulls* the changes into his working copy

Main uses

- Personal:
  - Backup
  - Safety against mistakes (can undo changes)
  - Convenient way to synchronize across computers

- Collaborative
  - Joint code development, or writing of manuscripts
  - version control takes care to merge all changes together

git - setup

CITA–ICAT

❖ git is a version control system

- next few pages with examples follow
  https://www.kernel.org/pub/software/scm/git/docs/gittutorial.html

❖ minimal setup (do this once)

- git config --global user.name "Your Name Comes Here"
- git config --global user.email you@yourdomain.example.com

❖ some more useful configurations (do this once)

- git config --global color.ui true # Colors!
- git config --global push.default tracking # 'git push' only pushes the current branch
- git config --global blame.date short # Make 'git blame' readable
- git config --global merge.conflictstyle diff3
  # Conficts show 3 versions: <<<<< yours |||| original ====== theirs >>>>>>

git - Personal repository

CITA–ICAT

❖ Create a repository

- mkdir Project
- cd Project
- git init

```
retina Examples/git $ mkdir Project
retina Examples/git $ cd Project/
retina git/Project $ git init
Initialized empty Git repository in /Users/pfeiffer/Documen
ts/Teaching/2013_CTA200/Examples/git/Project/.git/
retina git/Project (master #) $
```

❖ Adding files

- emacs File1.txt & # create a file
- git status # shows all uncommitted changes
- git add File1.txt # schedule changes to this file to be committed
- git commit # commit changes into repo (be sure to enter
  # a clear description of what the changes are)

❖ Making changes

- emacs File1.txt # modify file
- echo "2 3 4 5" > numbers.txt # create a second file
- git status # shows non-committed changes

```
retina git/Project (master) $ emacs File1.txt
retina git/Project (master *) $ echo "2 3 4 5" > numbers.txt
retina git/Project (master *%) $ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   File1.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       numbers.txt
no changes added to commit (use "git add" and/or "git commit -a")
retina git/Project (master *%) $
```

❖ Committing changes

- git add numbers.txt # can pick which changes to commit
- git commit # commits only numbers.txt
- git status
- git add File1.txt # commit change
- git commit # to File1.txt

❖ Checking the repository

- git log

Basic git workflow

- Fundamental cycle:
  - Edit
  - git status
  - git add
  - git commit

- RULE: Commit often!
  - what's committed is saved.

git GUI's

- Some git command-line knowledge is useful when working remotely
  - check out a source-code repository on a supercomputer
- For local work, GUI's are an excellent tool to navigate a repository
  - gitk -- standard, comes with git
  - gitx -- OS-X variant of gitk, looks nicer
  - SmartGitHg -- Harald's favourite
    - Very powerful and well-thought out GUI
    - Commerical, but free non-commercial license
    - http://www.syntevo.com/smartgithg/index.html

Advanced: Branches

- A repository can have multiple branches of your project
  - git branch crazy_idea # make a branch (no changes yet)
  - git checkout crazy_idea # switch to branch

  - emacs File1.txt # modify
  - git add File1.txt
  - git commit # as always

- You can switch between different branches
  - git checkout master # back to 'master' branch (i.e. the main branch)
  - emacs File1.txt # modify
  - git add File1.txt
  - git commit # as always

- Where are we?
  - git branch # shows all branches, active one with *

Advanced: Branches II

CITA–ICAT

- Merging often just works
  - git checkout master # make sure we're on that branch where we
  - git merge crazy_idea # merge crazy_idea back into master
  - gitk (or SmartGit) # look at commit history
- crazy_idea completed, remove branch
  - git branch -d crazy_idea
  - Note: This deletes only the 'tag' crazy_idea, without it, we cannot switch back onto crazy_idea branch, and cannot edit it further. All intermediate commits still there. (check with SmartGit)

Conflicts

- Ok, let's be nasty
  - git branch conflict # create branch called 'conflict', but do NOT switch
  - emacs File1.txt # change some line inside the file
  - git add File1.txt
  - git commit

  - git checkout conflict
  - emacs File1.txt # change same line as above
  - git add File1.txt
  - git commit

- Let's try to merge
  - git checkout master
  - git merge conflict
  - ***OUCH***

```
retina git/Project (conflict) $ git checkout master
Switched to branch 'master'
retina git/Project (master) $ git merge conflict
Auto-merging File1.txt
CONFLICT (content): Merge conflict in File1.txt
Automatic merge failed; fix conflicts and then commit the result.
retina git/Project (master *+|MERGING) $
```

Conflict resolution

- Use “git status” and “git diff” to find offending file(s)

```
retina git/Project (master *+|MERGING) $ git status
# On branch master
# You have unmerged paths.
#   (fix conflicts and run "git commit")
#
# Unmerged paths:
#   (use "git add <file>..." to mark resolution)
#
#       both modified:      File1.txt
#
no changes added to commit (use "git add" and/or "git commit -a")
retina git/Project (master *+|MERGING) $
```

- Edit by hand, and choose what the correct text should be.
  - emacs File1.txt
  - git add File1.txt
  - git commit

Collaborative use: github

- Your local git repository can *track* a *remote repository*
- The remote repository can be anywhere
  - CITA, your second computer, a friends computer, on a git hosting site...
  - You must be able to access the computer that hosts the remote repository. This can be tricky to setup.

- Quite simple: Track a repository on github
  - git clone git@github.com:CITA/CTA200.git # creates CTA200/
  - cd CTA200
  - ls
  - git log
- CTA200 is a repository we will be using for this course
  - Class notes, assignments

Collaborative work

- e.g. add "solutions" to HW1
  - cd CTA200
  - cd Students
  - mkdir PfeifferHarald
  - cd PfeifferHarald
  - mkdir HW1
  - cd HW1
  - emacs Solution.txt
  - git status
  - git diff
  - git add Solution.txt
  - git commit
- NB: Homework sets are NOT collaborative -- you are expected to do your homework alone, just to submit it into the repository.

Local vs. remote repository

- The local clone of the remote repository is a complete "first class" repository itself. After the initial "clone", we have NOT referred to the remote repository at all.
- Pull changes from the remote repository (i.e. changes from your colleagues)
  - git pull --rebase

```
retina PfeifferHarald/HW1 (master % u=) $ git add Solution.txt
retina PfeifferHarald/HW1 (master + u=) $ git commit
[master 44cccdb] my solution
 1 file changed, 1 insertion(+)
 create mode 100644 Students/PfeifferHarald/HW1/Solution.txt
retina PfeifferHarald/HW1 (master u+1) $ git pull --rebase
Current branch master is up to date.
retina PfeifferHarald/HW1 (master u+1) $ git push
Counting objects: 8, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 499 bytes, done.
Total 6 (delta 1), reused 0 (delta 0)
To git@github.com:CITA/CTA200.git
   0c96397..44cccdb  master -> master
retina PfeifferHarald/HW1 (master u=) $
```

- Push changes to github
  - git pull --rebase # check for changes to remote repo.
  - git push

Some notes

- Many git commands require you to have all your local changes committed:
  - git checkout other_branch # if not committed, couldn't switch back
  - git pull --rebase # if not committed, couldn't revert conflict

  - No problem if you follow the rule to ***Commit often***

- Coloring and display of git branch is NOT standard.
  - You won't see this today.
  - We'll cover this on Thursday with more bash-details

```
retina PfeifferHarald/HW1 (master % u=) $ git add Solution.txt
retina PfeifferHarald/HW1 (master + u=) $ git commit
[master 44cccdb] my solution
 1 file changed, 1 insertion(+)
 create mode 100644 Students/PfeifferHarald/HW1/Solution.txt
retina PfeifferHarald/HW1 (master u+1) $ git pull --rebase
Current branch master is up to date.
retina PfeifferHarald/HW1 (master u+1) $ git push
Counting objects: 8, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 499 bytes, done.
Total 6 (delta 1), reused 0 (delta 0)
To git@github.com:CITA/CTA200.git
   0c96397..44cccdb  master -> master
retina PfeifferHarald/HW1 (master u=) $
```

.gitignore

- Some files should NOT be committed into a repository, because they are created automatically:
  - Backup files ending in $\sim$
  - Helper files of OSX (.DS_Store)

- If one places a file ".gitignore" inside a git repository and lists certain files there, they will be ignored by git

Emacs

Emacs

- A very powerful text-editor
- Just knowing 5% of it will make you much more efficient
- To start in separate window,
  - emacs &
- To start in the present terminal,
  - emacs -nw
- In either case, you can add filenames on the command line
  - emacs machines.txt

Tutorial, Reference card

- The built-in emacs tutorial is excellent. DO IT.

- There are also a variety of reference cards for emacs around, which list the keyboard shortcuts. One of them is in the github repository under
  - CTA200/LectureNotes/Lecture01/refcard.pdf