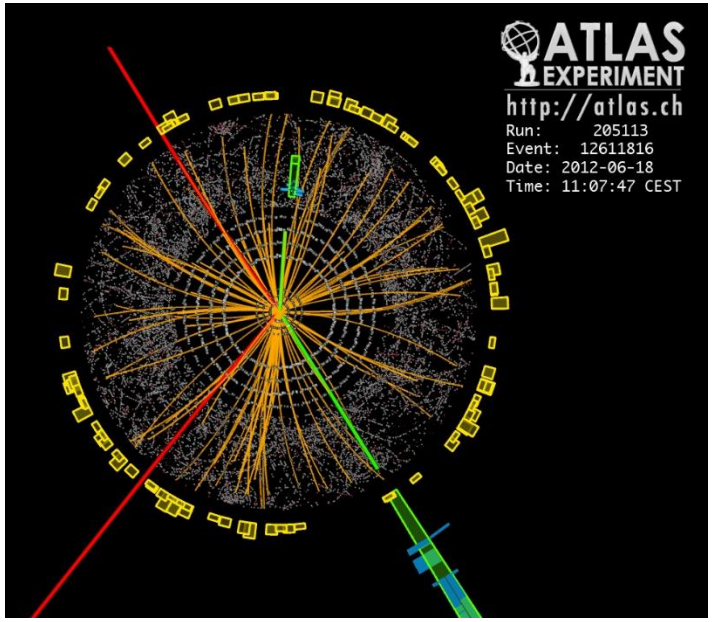
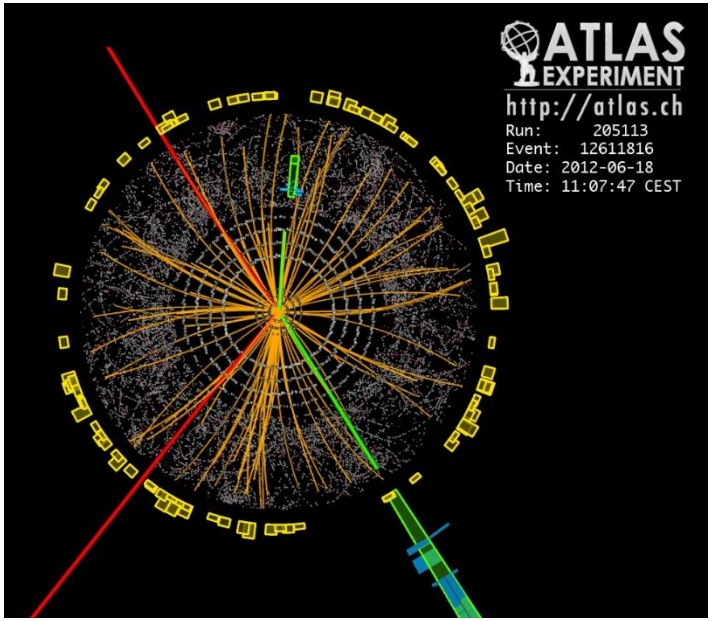
ROOT
A (Big) Data Analysis Framework



David Webber, Ph.D.

Data Scientist, Scanalytics, Inc.

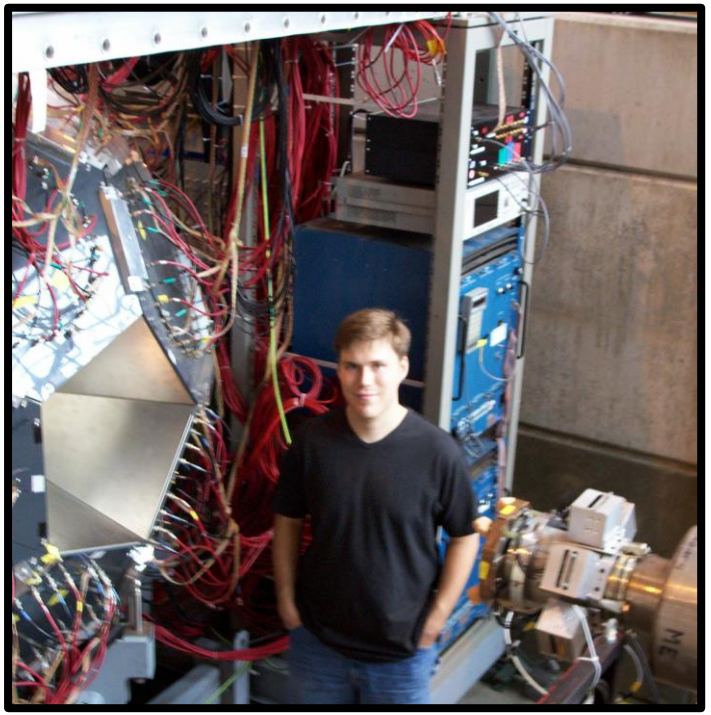
Outline



- About me
- About ROOT
- Big Science and Big Data
- ROOT
  - Installing
  - Exploring Madison Real-Estate data
  - Parameter estimation
  - Further examples
    - Data structures
    - Other things you can do with ROOT
- Summary

About David

- Physicist: measured particle properties 2003-2013. 2 main projects:
  - Muon (a “heavy electron”) Lifetime
  - Neutrino mixing angle θ_{13}
- Data Scientist: 2013-Now
  - Scanalytics, Inc
- Using ROOT > 10 years
- Disclaimer: not associated with CERN, LHC experiments, or ROOT development

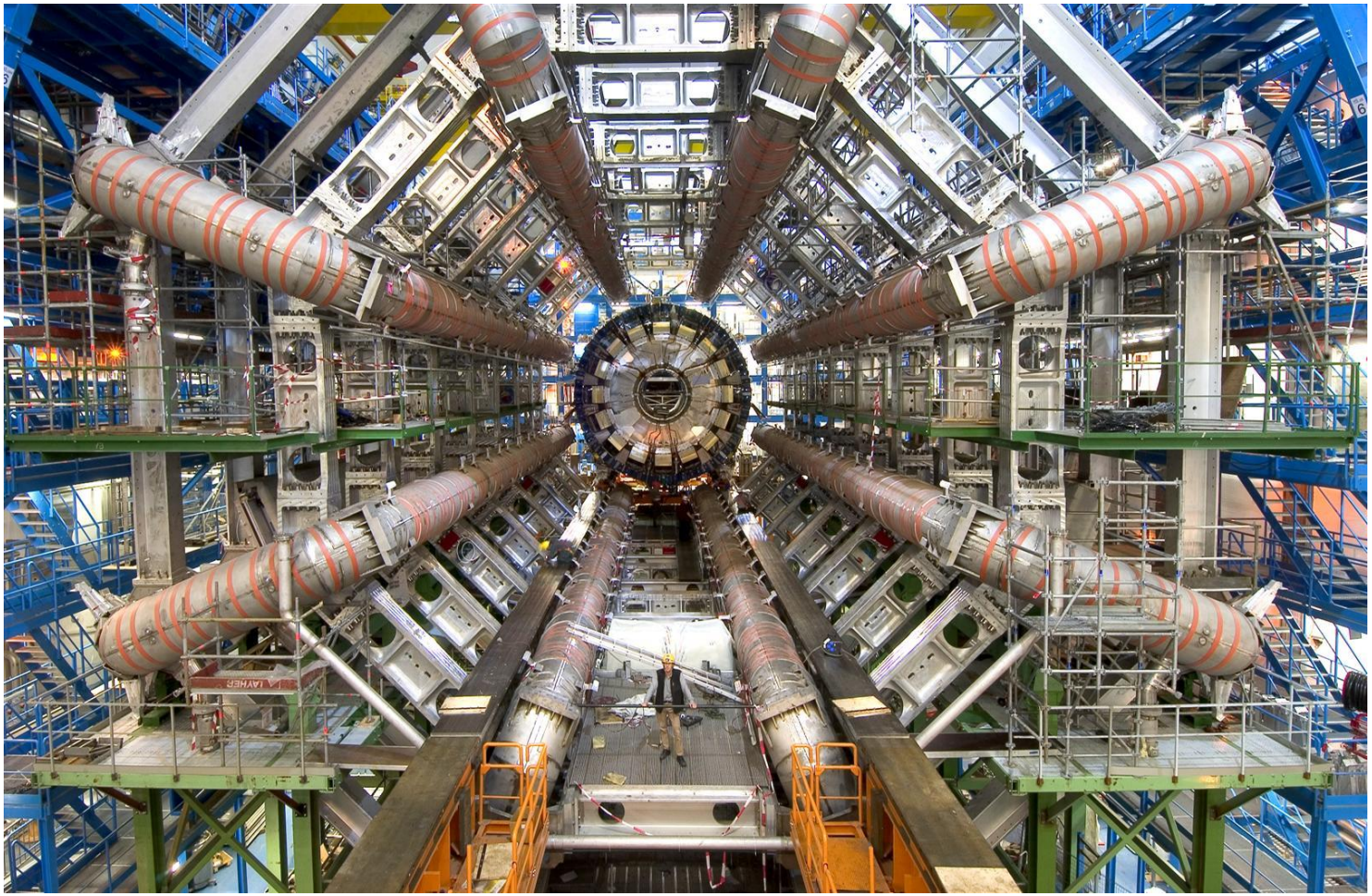


Standing next to a physics experiment (MuLan), circa 2005

About ROOT

- Data analysis framework
- Object-Oriented C++ (and Python and Ruby)
- Extremely Efficient
- Actively developed (1994-Present)
  - In 2012: 6100 forum members, 62000 posts, 1300 mailing list members
- De-Facto standard for High-Energy Physics
  - Data storage, Analysis, plots and charts
- Many physicists' first experience with object-oriented programming
  - Replacement for FORTRAN (CERNLIB, PAW), as of 2003
- I'll talk about
  - Interface: CINT (to be replaced with CLING)
  - Storage: ROOT Trees

Big Science: Big Detectors

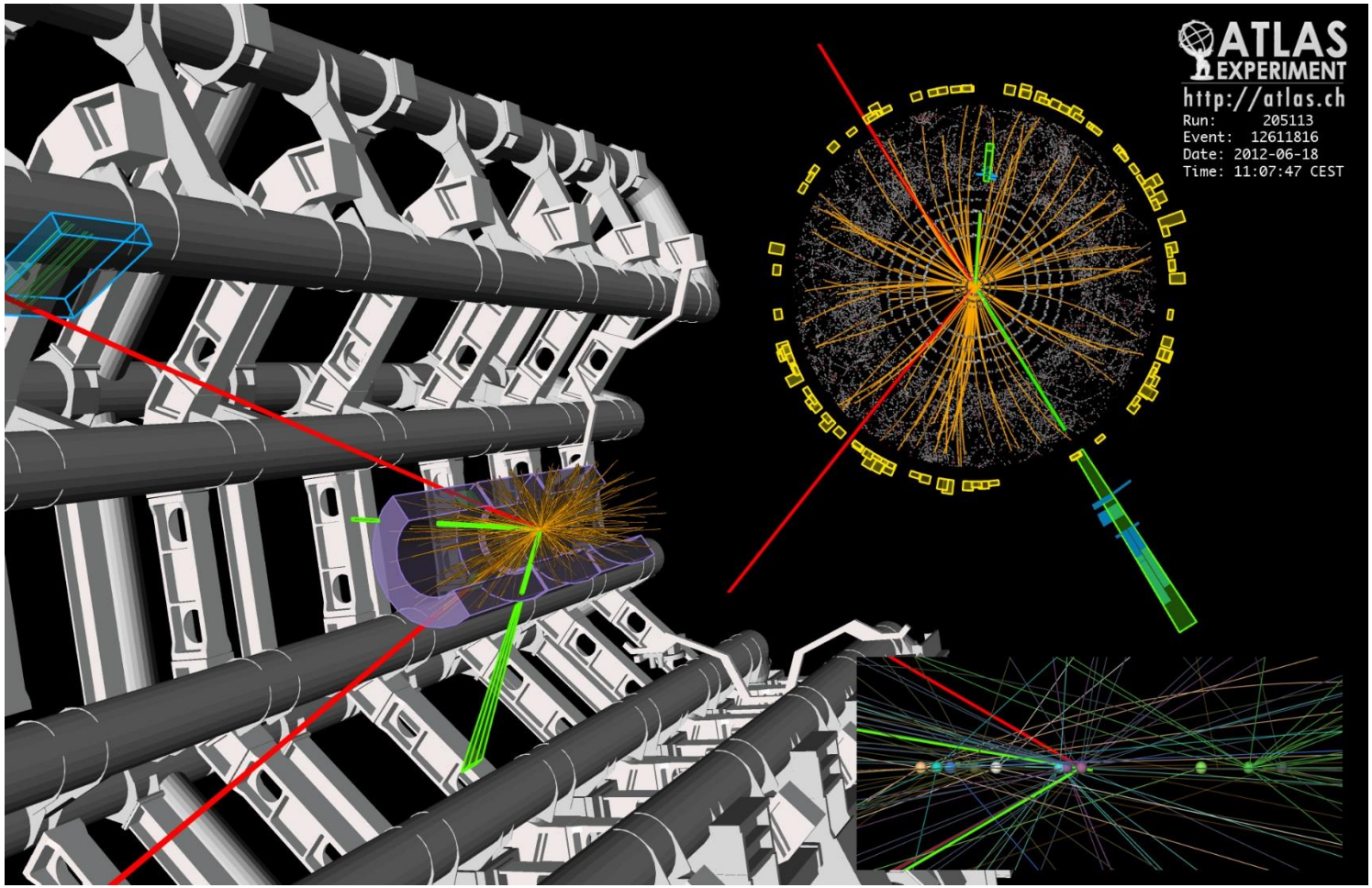


ATLAS detector infrastructure

http://www.nevis.columbia.edu/~atlas/photos/ATLASdetector.jpg

Big Science: Big Data

A “sighting” of the Higgs Boson

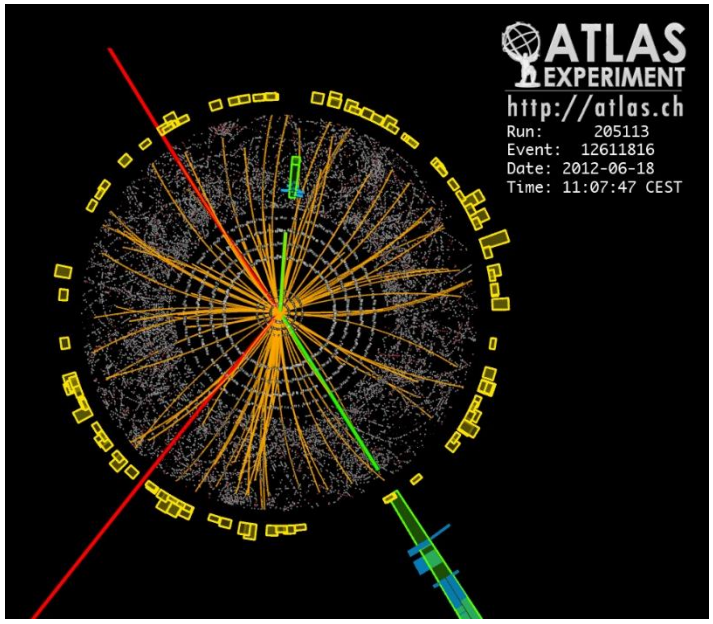


http://www.bnl.gov/atlas/news/news.php?a=11512

Big Science: Big Data

- 1 MByte per collision “event”
- 600 Million events per second
- 100 Million events per second selected for digital reconstruction
- 100-200 events selected for recording
- 1 GB/s raw data rate after event selection
- 15 petabytes (15,000 Terabytes) produced annually

A “sighting” of the Higgs Boson



H -> 2e2mu candidate event

http://home.web.cern.ch/about/computing/processing-what-record

http://www.bnl.gov/atlas/news/news.php?a=11512

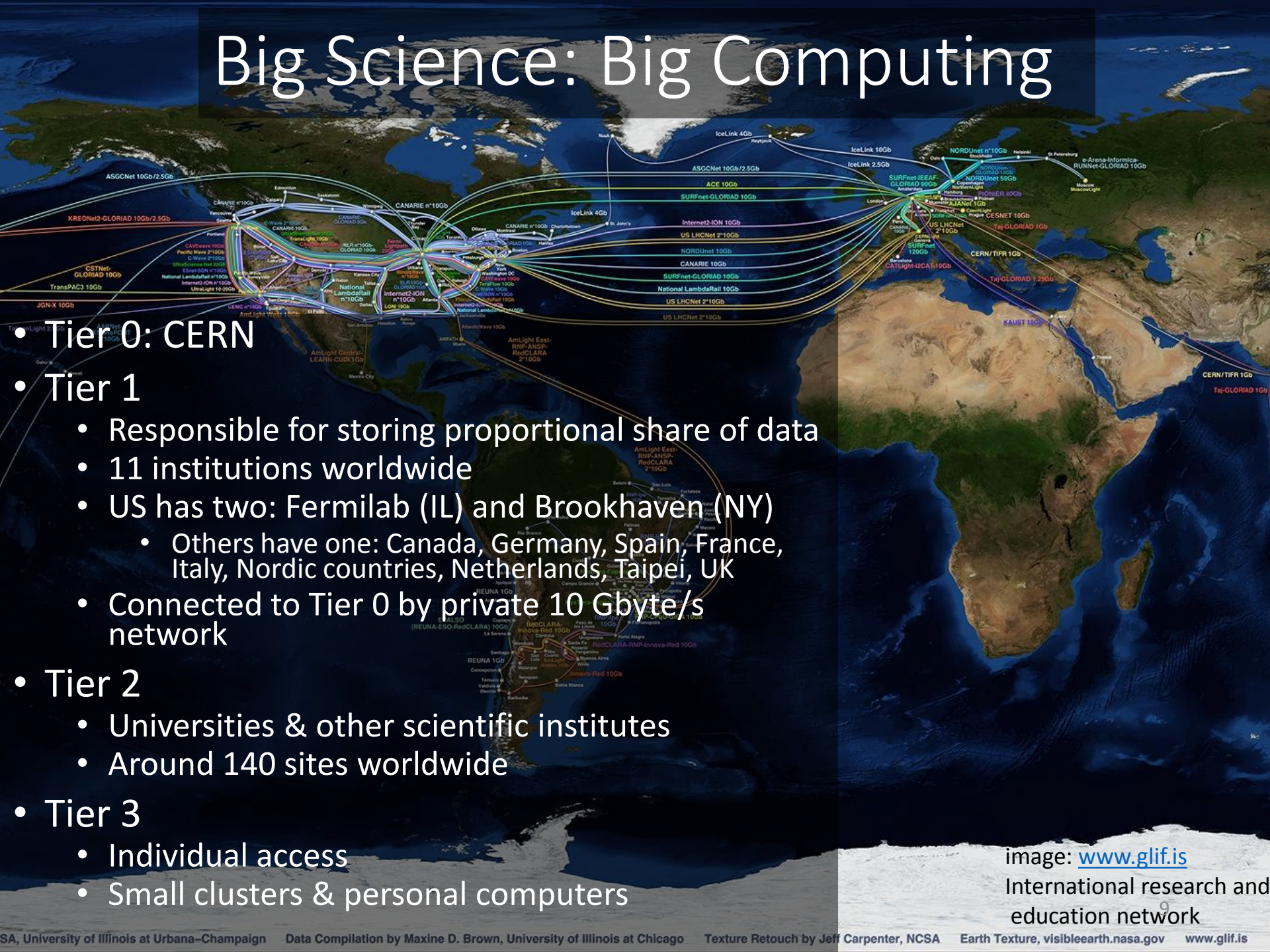
Big Science: Big Computing

Source: home.web.cern.ch/about/computing

Cern Data Center

- 1450 square meters
- 3.5 MW electrical power capacity
- 30 Pbytes (30,000 TB) data storage on disk
- 90,000 processing cores
- Processes about 1 petabyte each day
- Tier 0 of a multi-tier system

Big Science: Big Computing



- Tier 0: CERN
- Tier 1
  - Responsible for storing proportional share of data
  - 11 institutions worldwide
  - US has two: Fermilab (IL) and Brookhaven (NY)
    - Others have one: Canada, Germany, Spain, France, Italy, Nordic countries, Netherlands, Taipei, UK
  - Connected to Tier 0 by private 10 Gbyte/s network
- Tier 2
  - Universities & other scientific institutes
  - Around 140 sites worldwide
- Tier 3
  - Individual access
  - Small clusters & personal computers

image: www.glif.is
International research and education network

Phases of Data Analysis

- Getting the tools
  - Installing ROOT
- Access
  - CSV and ROOT files
- Exploration
  - Visualization
- Discovery
  - Parameter estimation
  - Fitting functions to data
- Documentation/Dissemination
  - Saving your output

Installing ROOT

- From Source (my preferred; do not mix with distribution binaries)
  - Reference: http://root.cern.ch/drupal/content/installing-root-source
  - git clone http://root.cern.ch/git/root.git
  - cd root
  - Grab a specific release
    - git tag -l
    - git checkout -b v5-34-08 v5-34-08
  - Follow a branch
    - git branch –a
    - git checkout v5-34-00-patches
  - ./configure;make OR CMake
- Fedora
  - sudo yum search root
  - sudo yum install –y root*
- Ubuntu
  - sudo apt-get install root-system
- Windows (YMMV)
  - Download installer
  - http://root.cern.ch/drupal/content/production-version-534

Starting ROOT

- If you compiled from source, load the environment
  - Source <ROOT_install_dir>/bin/thisroot.sh
  - (not necessary if you installed pre-built binaries)
- Useful command-line arguments
  - -l skip splash screen
  - -x execute script/macro
  - -q quit after executing script

Sample Data Exploration using TTrees

- Example: flat CSV data file
  - https://data.cityofmadison.com/Property/Assessor-Property-Information/u7ns-6d4x
  - 77 columns, 77915 rows
  - ./col_select.sh > cleaned.csv
- Start ROOT
  - root –l
- Create a TTree object
  - TTree *mytree = new TTree(“mytree”,”mytree”)
- Load the data
  - mytree->ReadFile(“cleaned.csv”)
- Start the viewer
  - mytree->StartViewer()
- Double-click column names to explore!
  - Note, may need to right-click in margin and toggle “SetLogy”

Sample Data Visualization: Command-line examples

- 2D plots
  - scatter
    - mytree->Draw("Current_Year_Improvement_Value:Current_Year_Land_Value")
  - color
    - mytree->Draw("Current_Year_Improvement_Value:Current_Year_Land_Value“,“”,”colz”)
    - SetLogz
- 3D plots
    - mytree->Draw…
    - ("Total_Taxes:Current_Year_Improvement_Value:Current_Year_Land_Value“)
- Cuts on the data
    - mytree->Draw
    - ("Current_Year_Improvement_Value:Current_Year_Land_Value",
    - "Current_Year_Improvement_Value<1e6 && Stories==1",
    - "colz")

Data fitting example: What is the tax rate?

- Prepare the histogram
  - mytree->Draw("Total_Taxes:Current_Year_Total_Value","","colz")
- Prepare the function
  - TF1 *myline = new TF1("myline","[0]+[1]*x",0,100e6)
  - htemp->Fit("myline")
  - slope: 2.47 +/- 0.04 % (or $24.7 +/- 0.4 per $1000)

2013 Dane County Tax Rates

MUNICIPALITY	SCHOOL DISTRICT	RATE PER $1000	Value Ratio Assessed to Market
Cities			
Edgerton	Edgerton	$23.10	101.20%
Fitchburg	Madison	$22.89	100.40%
Fitchburg	Oregon	$23.49	100.40%
Fitchburg	Verona	$23.50	100.40%
Madison	Madison	$24.88	97.70%
Middleton	Mid-Cross Plains	$22.05	95.50%
Monona	Monona Grove	$23.63	97.40%
Stoughton	Stoughton	$23.06	101.70%
Sun Prairie	Sun Prairie	$24.72	100.10%
Verona	Verona	$22.81	99.30%

http://www.toddanddeahmulhern.com/default.asp.pg-PropertyTaxInformation

ROOT Trees are memory-efficient

- Sliding window on data
- No overhead of object creation
- Platform-independent data format
- Assumptions about data
  - Structured
  - Won't change
  - Independent rows
- Supported data structures:
  - C Struct
  - Internal states of objects
  - STL vectors
  - Anything for which you can write a "TStreamer"
- Docs:
  - http://root.cern.ch/drupal/content/users-guide
  - http://root.cern.ch/download/doc/Trees.html

From exploration to programming

- Saving a set of commands as a script
  - Root's history file is ~/.root_hist
  - Just add function names and semicolons
- Compiling the script to increase execution speed
  - Add header.h files to your script
  - define variables (can't rely on interpreter)
  - start root (root –l)
  - Load and compile: ".L macro.cxx+"
- Compiling a stand-along executable
  - g++ -o hello hello.cc `root-config --cflags --glibs`
  - Embed into analysis framework like Condor or Hadoop
  - http://en.wikibooks.org/wiki/ROOT/Getting_Started/Many_Ways_to_Use_ROOT#Building_a_Stand-Alone_Application

More Examples of TTrees

- CERN Staff 1988
  - Shows how to handle text fields
  - cd $ROOTSYS/tutorials/tree
  - root –x cernstaff.C
- Storing STL vectors in trees
  - root -x hvector.C
- Storing objects in trees
  - root -x tree0.C+

A few additional examples

- Explore the Mandelbrot Set
  - $ROOTSYS/tutorials/graphics/mandelbrot.C
- Basic Animation
  - $ROOTSYS/tutorials/graphics/anim.C
- Manipulating an Image
  - $ROOTSYS/tutorials/image/galaxy_image.C
- Talking to SQL
  - $ROOTSYS/tutorials/sql (directory)

Interfaces in Python and Ruby

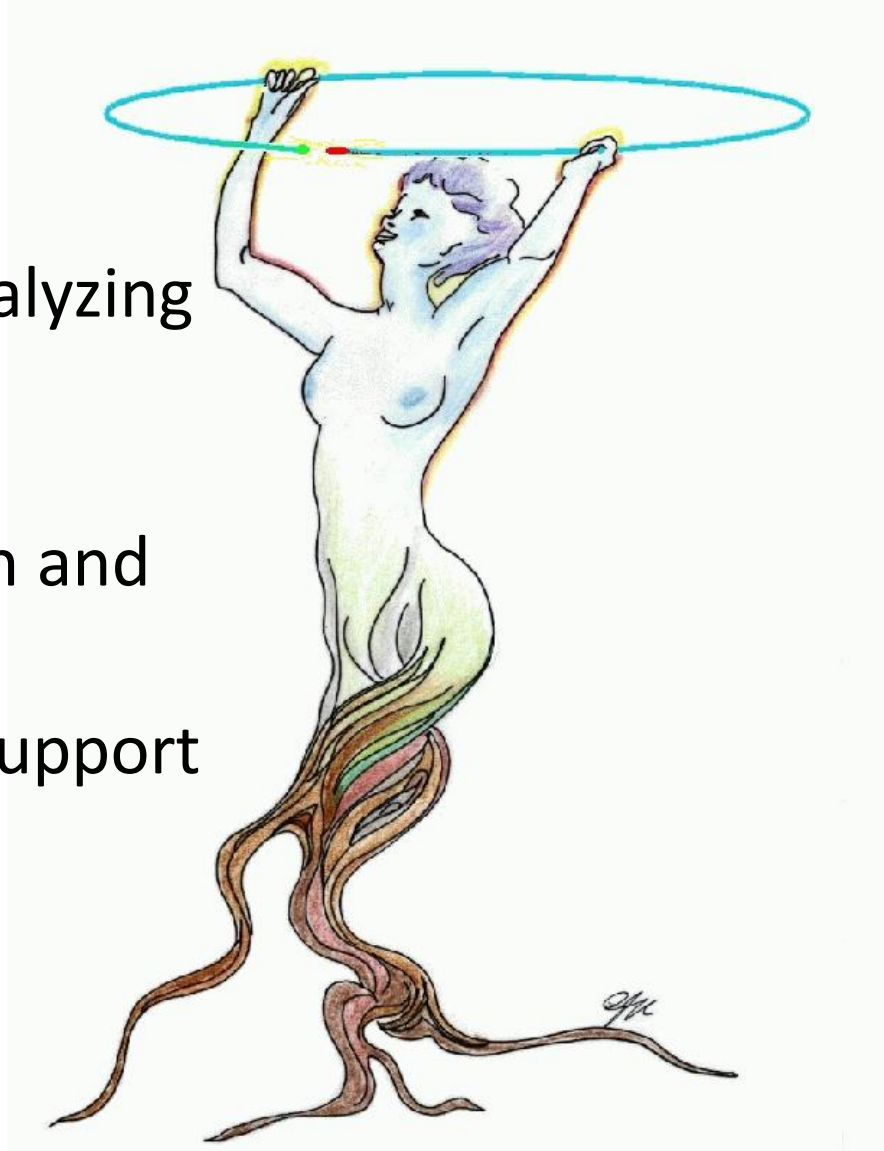
- Python
  - http://root.cern.ch/drupal/content/how-use-use-python-pyroot-interpreter
  - http://root.cern.ch/drupal/content/pyroot
  - http://root.cern.ch/root/html534/tutorials/pyroot/index.html
- Ruby
  - http://root.cern.ch/root/HowtoRuby.html
  - http://root.cern.ch/root/html534/tutorials/ruby/index.html

Pros/Cons of ROOT and TTree

- Good
  - Massive storage
  - Embarrassingly parallel
  - Free as in speech (lesser GPL)
  - Actively developed and supported, well documented
    - http://root.cern.ch
    - ROOTTalk user forum
    - DOXYGEN – source code
    - Manual
- Bad
  - No Sorting (But fast traversal and selection)
- Ugly
  - Steep learning curve
  - Has to support legacy code
  - Named objects (TNamed) and root file directory-based scope

Summary

- ROOT is a powerful tool for analyzing large, complex datasets
- The CINT (soon to be CLING) interpreter enables exploration and code prototyping
- A robust community ensures support for years (decades?) to come



"This pastel drawing has been specially made for the ROOT project by Giuliana Carminati in October 1999."
- http://root.cern.ch/root/RootArt.html

Questions and Answers

- Does ROOT keep all the data in memory?
  - It depends.
  - In the examples here, all data is kept in memory.
  - For large trees already on disk, data is staged into memory and used to build histograms. You can choose to read only certain leaves ("columns") into memory.
  - "…buffers may be automatically written to disk or kept in memory until the Tree attribute fMaxVirtualSize is reached." - http://root.cern.ch/root/html/TTree.html
- Does root have a size limit on objects?
  - No. (It's C++) You may define objects of arbitrary size, but you will run into limits from system memory and swap space from the OS. I have defined histograms with 2GB memory footprint. When writing very large trees, you must attach them to a file and specify the size at which they get flushed to disk.