UNIVERSITY of CALIFORNIA · IRVINE

BigData on Linux

Linux: Harry Mangalam
harry.mangalam@uci.edu

UCI Data Science Initiative

HPC Emails

- Always cc: <hpc-support@uci.edu>
- Joseph Farran <jfarran@uci.edu>
- Harry Mangalam <hmangala@uci.edu>
- Garr Updegraff <garru@uci.edu>
- Adam Brenner <aebrenne@uci.edu>
- Edward Xia <xias@uci.edu>

Course Survey

http://goo.gl/yPS6WK

Intentions

- Not a HOWTO on specific BigData techniques
- Introduction of how to think about large-scale computing.
- What I wish someone had told me when I was starting out with Unix/Linux.
- I am not a CS guy so a few of my explanations may be formally wrong.
- But mostly I'm right, or right enough.
- Remember...

Good Judgement comes from Experience

Experience comes from Bad Judgement

I assume...

- You are now familiar with Linux and at least a little familiar with cluster computing.
- You're bright: can Google, and read further by yourself.
- If I speak too fast; let me know
- Questions, ASK THEM, but I may not answer them immediately. – "*You don't know what you don't know*"

Some of you...

– Writing your own apps
– Starting with interpreted languages
– Maybe moving to compiled languages
– Trying to parallelize your work (trivial or sophisticated approaches).

This involves BEING a programmer.

All of you..

- – Cleansing your data (bash, utilities)
- – Writing qsub scripts → SGE
- – Running pre-written apps with your data
- – Pushing large amounts of data thru HPC
- – Developing your own workflows to do this

All these tasks require THINKING like a programmer.

Computing Philosophy

Unlike your Science...

- Be lazy.
- Copy others.
- Don't invent anything you don't have to.
- Re-USE, re-CYCLE, DON'T re-invent.
- Do the easy stuff first.
- Don't be afraid to ask others.
- Try it, but try it small at first.
- Resort to new code only when absolutely necessary.
- Optimize only as a last resort.

Linux & the HPC Cluster

Introduction to Linux on the HPC Cluster

- Linux
- Bash shell & variables
- Commands
- Pipes
- The HPC cluster
- Distributed file systems

Getting Help

- Fix IT Yourself with Google <goo.gl/05MnTi>
- Listservs, forums, IRCs are VERY useful for more involved questions
- The HPC Doc list: <http://hpc.oit.uci.edu/>
- HPC HOWTO <http://goo.gl/kzlqI>
- Us – Adam, Harry, Garr, Joseph.
- BUT!! Please ask questions intelligently.

How to Ask Questions

- Reverse the situation: if you were answering the question, what information would you need?
- Not Science, but it is Logic.
- Include enough info to recreate the problem.
- Exclude what's not helpful or ginormous (use <pastie.org> or <tny.cz>)
- Use text, not screenshots if possible.

Bad Question

Why doesn't "X" work?

Good Question

I tried running the new podunk/2.8.3 module this morning and it looks like I can't get it to launch on the Free64 queue. My output files aren't helping me figure out what is wrong.

I am working out of /bio/joeuser/RNA_Seq_Data/

and the qsub script is 'job12.sh'. The output should be in

- /bio/joeuser/RNA_Seq_Data/output

When I submit the job, it appears to go thru the scheduler but then dies immediately when it hits the execution node.

I can't find any output to tell me what's wrong, but the Error messages suggest that there's a problem finding libgorp.so.3

HELP US HELP YOU

We need this information:

- the directory in which you're working (*pwd*),
- the machine you're working on (*hostname*)
- modules loaded (*module list*)
- computer / OS you're connecting from
- the command you used and the error it caused (in /text/, not screenshot)
- much of this info is shown by your prompt

```
Wed Nov 05 21:24:48 [0.91 1.04 1.08]  hjm@stunted:~/nacs/bigdata
517 $
```

On to HPC

What is the H_{igh} $P_{erformance}$ $C_{omputing}$ Cluster?

and…

Why do I need HPC?

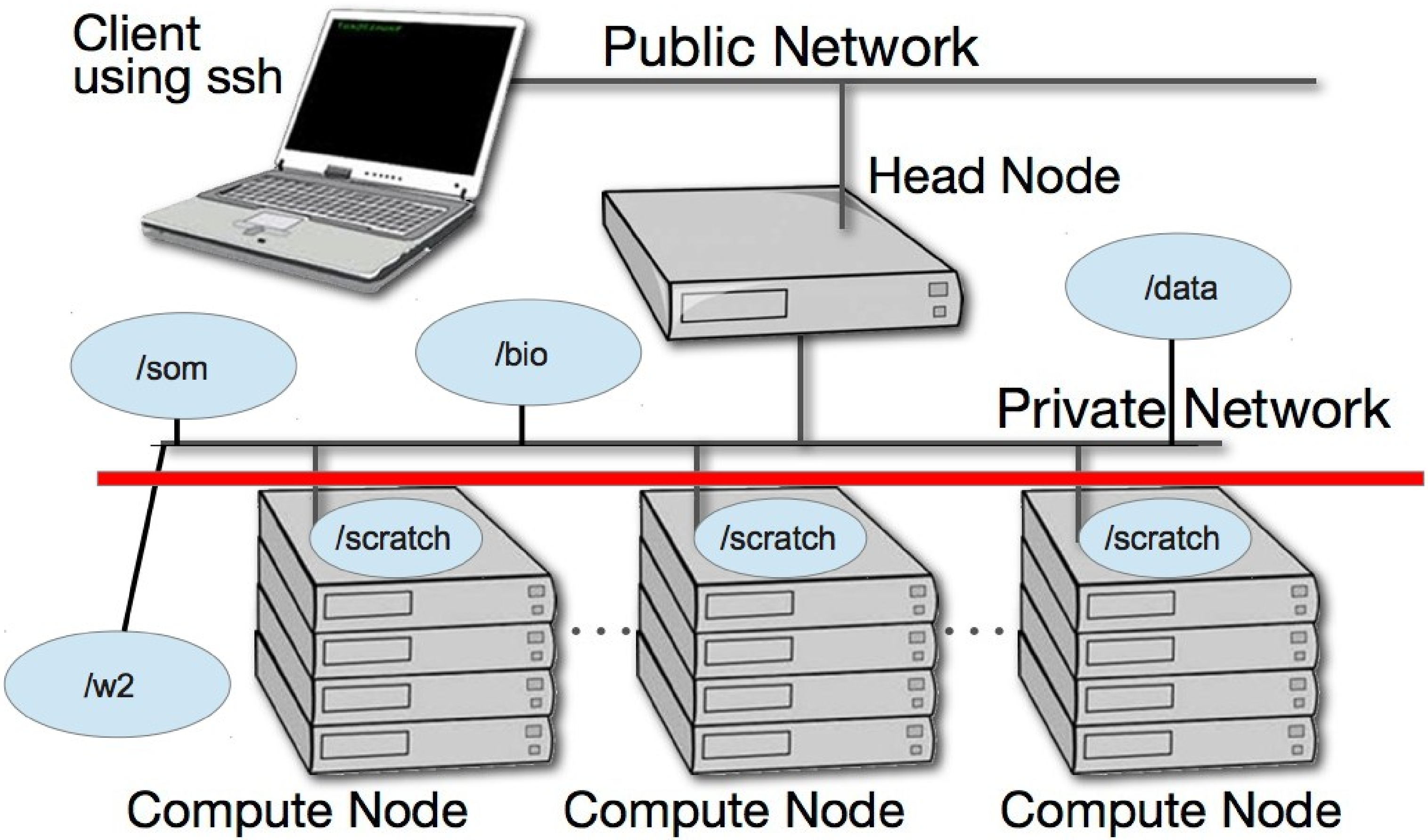
What is a Cluster?

- bunch of big general purpose computers
- running the ***Linux*** Operating System
- linked by some form of networking
- have access to networked storage
- that can work in concert to address large problems
- by scheduling jobs very efficiently

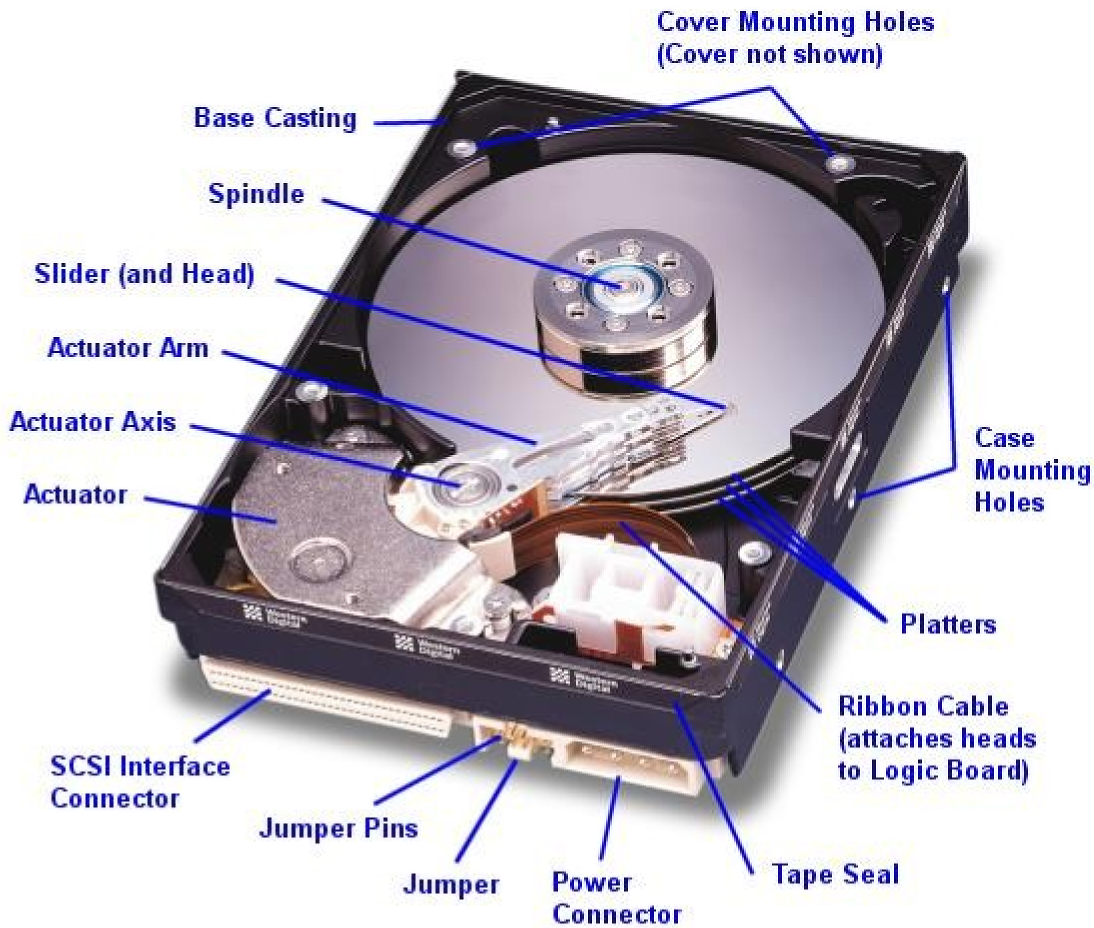
HPC @ UCI in Detail

- ~5500 64b Cores – Mostly AMD, few Intel
- 4+ Nvidia Tesla GPUs (2880 cores each)
- ~14TB aggregate RAM
- ~1PB of storage (1000x slower than RAM)
- Control network = 1Gb ethernet (100MB/s)
- Data network = QDR IB (5GB/s)
- Grid Engine Scheduler to handle Queues
- > 650 users, 100+ are online at anytime

Overview

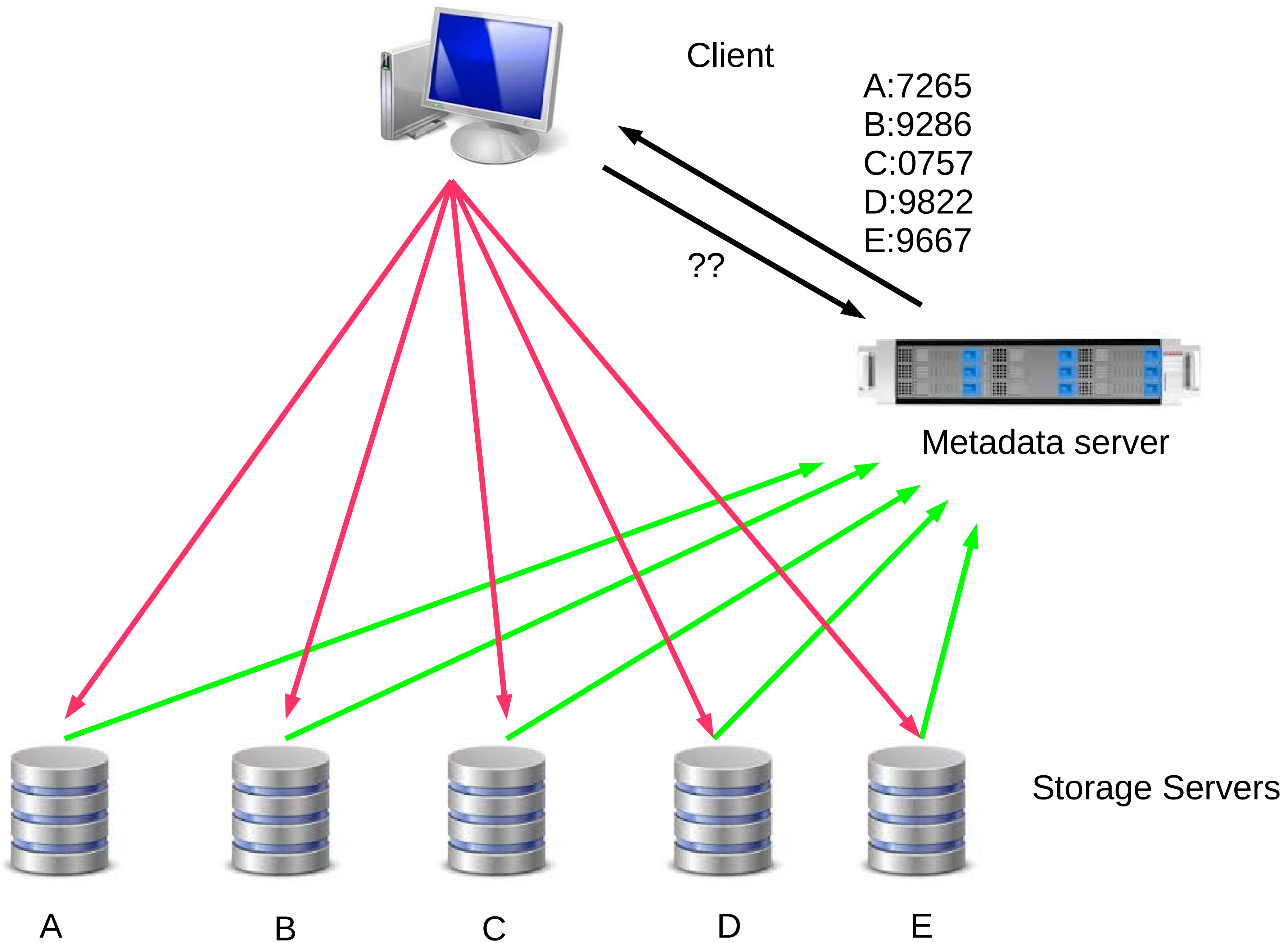

Client
using ssh
Public Network
Head Node
/data
/som
/bio
Private Network
/scratch
/scratch
/scratch
/w2
Compute Node
Compute Node
Compute Node

A hard disk


Cover Mounting Holes
(Cover not shown)
Base Casting
Spindle
Slider (and Head)
Actuator Arm
Actuator Axis
Actuator
Case
Mounting
Holes
Platters
Ribbon Cable
(attaches heads
to Logic Board)
SCSI Interface
Connector
Jumper Pins
Jumper
Power
Connector
Tape Seal

Storage Server

Rear View


Client
A:7265
B:9286
C:0757
D:9822
E:9667
??
Metadata server
Storage Servers
A
B
C
D
E

Applications on HPC

- We use the 'module system' to set up environments for specific applications, libraries, and compilers.
- module purge
- module avail *prefix*
- module list
- module whatis *name*
- module load *name/version*
- *Rec NOT preloading a lot of modules.*

What HPC is NOT

- NOT your personal machine
- What you do on your machine affects YOU
- What you do on HPC affects the 1000s of other jobs running
- Think before you hit Enter.

What HPC is also NOT

NOT BACKED UP

WHAT. SO. EVER.

DATA IS NOT BACKED UP

- Agitate to your PIs to get us more $ if you want this.
- Most data is stored on RAID6
- BUT! Any of that can disappear at any moment
- IF ITS VALUABLE, back it up elsewhere --- or the code that generated it.

HPC FileSystem Layout

Orange – Cluster Wide

Black – Node Specific

```
/
├── data/          NFS Mount
     |–apps        All Programs are installed here
     |–users       Users home directory                                               – 50GB LIMIT PER USER
├── w1/            Public NFS Server      → Going away                                – 14TB Space
├── w2/            Public NFS Server      → Going away                                – 40TB Space
|----- pub/        Replacement for /w1, /w2
├── bio/           Space for BIO group  → /dfs1
├── som/           Space for SOM group → /dfs1
├── cbcl/          Space for CBCL group → /dfs1
├── dfs1/          Fraunhofer FileSystem – new, Distributed File System               ~380TB Space
├── scratch        Node-specific temporary storage per job (faster than all above)    ~1TB – 14TB of Space
├── fast-scratch   High Speed Fraunhofer FileSystem for temporary storage             - 13TB
|----- ssd-scratch Very High IOPS for DB, other jobs.
├── /tmp           Same as scratch
```

Disk Space / Quotes / Policies

- You can only have so much space
- 50GB for /data/ ($HOME directory)
- 1yr or older without use – please remove from cluster
- More for Condo owners or Groups who have bought extra disk space.
- Regardless, NO DATA IS BACKED UP

SGE and qsub scripts

- SGE / GE is the HPC scheduler
- A complex app that matches resource requests with the cluster resources.
- Resources are:
  - # of CPU cores
  - RAM
  - Special hardware (GPUs)

SGE Queues

- There are 3 main types of Qs
  - Free*: open to everyone
  - Group: open to the group
  - Owner: Open to the owner/lab
- To see what Qs you can submit to:
  - 'q'

SGE Utilities

- qstat: list the status of ALL the jobs
  - qstat -u <you> more useful
- qdel: delete your jobs
  - qdel -u <you> deletes ALL your jobs
  - qdel <GEJobID> (not PID)
  - qdel -f <GEJobID> force-kills the job
- qsub job.sh submits 'job.sh' to scheduler

qsub

- qsub script is just a bash script with some special SGE directives.
- Bash comments prefixed with '#'
- SGE directives prefixed with '#$'
  - To reserve CPUs, RAM,
  - particular CPU-loading
  - Checkpointing
  - Set up job arrays
- Job arrays?

SGE Job types

- The only type of job that can't be run via the scheduler is one that requires human intervention.
- Serial jobs
- Parallel jobs – faster! Or not?
- Job Arrays
- Checkpointing

Example qsub jobs

- Sleeper <http://goo.gl/EsGOgD>
- Generic qsub with lots of comments <http://goo.gl/qfqieL>
- Job that uses /scratch <http://goo.gl/6uY1hh>
- Array Job <http://goo.gl/rwurvX>
- Python qsub script generator <http://goo.gl/oIya1E>

And some warnings:

- qsub scripts are bash scripts with some GE directives; if they don't run in your bash shell, they won't run under GE.
- Run them with small sets of data until you know their behavior and how many resources they'll use.
- Once they run fine from the shell, submit them to GE with small sets of data.
- Then submit the full data set.
- And use mail carefully. (No Array jobs!)

Questions?

Preferences?

Specific Techniques?

Some follow-ups...

- MacOSX, hilite file and [cmd+i]→ full path
- rsync – beware of '--delete' and if you're going to use it, use '-n' 1st
- Problem of 2 login nodes – use 'byobu'
- Identifying & Stopping processes.
- *Permissions, chmod*, and #!shebang
- Environment variables
- ~/.bashrc, aliases, DirB
- sshfs and where it makes sense to use it

Some more follow-ups...

- How to set up ssh keys .
- IO Redirection
- The grep family

Before BigData,
How to think about Data in general

Data as a 747

Think of your data as an airplane

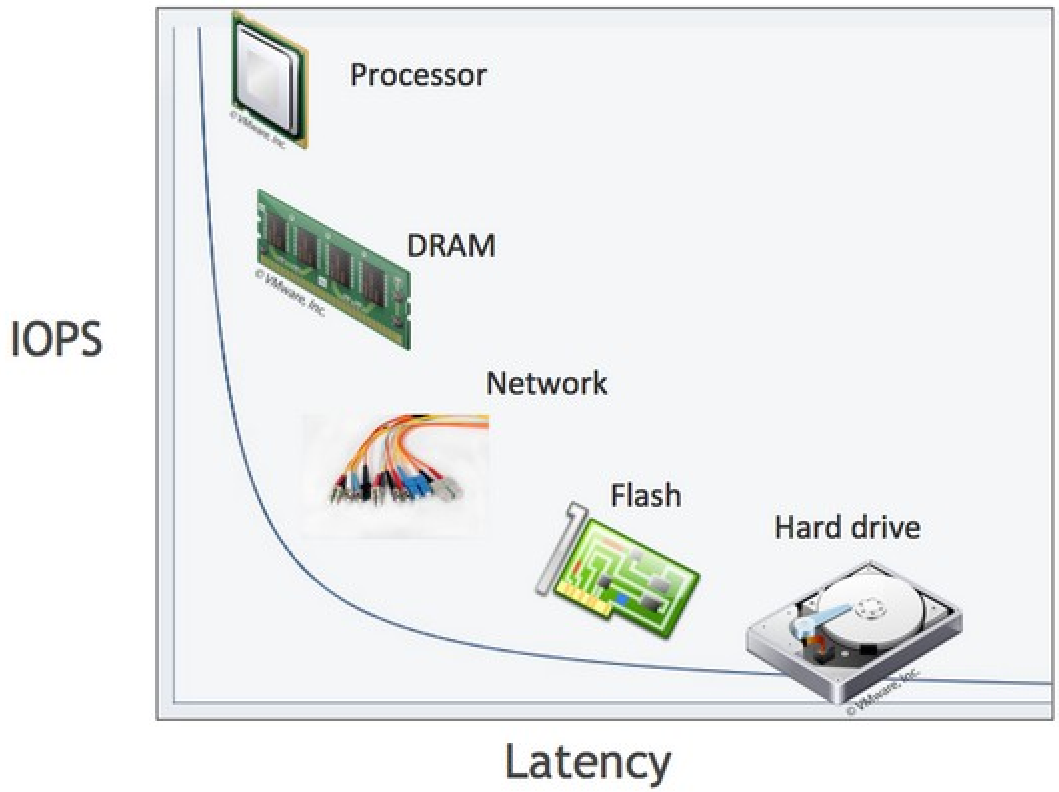
- Takes huge energy & time to get off the ground
- Once in flight, keep it in flight.
- Every time it lands, takes a lot of time to get it flying again.

Time of Byte Flights

Path / Timing of bytes thru the cluster

- CPU Registers: $1x10^{-10}$ sec
- 1^{o} cache: $10\text{-}50x10^{-10}$ sec
- 2^{o} cache: $100\text{-}500x10^{-10}$ sec
- Main RAM: $1\text{-}10x10^{-9}$ sec
- Network: $10x10^{-6}$ sec
- Flash Memory: $200x10^{-6}$ sec
- Disk: $5x10^{-3}$ sec

Data Latency



Data Latency Analogy

If Memory = Minute
Network = Weeks
Flash = Months
Disk = Decades

Inodes and ZOT Files

- Inodes contain the metadata for files and dirs
- Inodes are pointers to the data
- Regardless of size, a file needs at least one inode to locate it.
- A file of 1 byte takes up the same minimum inode count as a file of 1TB
- DO NOT USE ZOTFILES!! – Zillions of Tiny Files

How not to write ZOTfiles

- Append to a single file (100s of processes across many nodes can write to a single file via file-locking. See <http://goo.gl/EOf4qW>.
- Write to a Relational Database.
- Write continuously to custom Binary format.
- Write to a language-specific DATADUMP format.
- Write to a well-documented data format such as HDF5, FITS, netCDF, etc.

Processing Data on Linux

- The bash language is mostly awful.
- The redirection operators (<,>,|,>>,&>,2>, tee) are awesome, incredibly powerful, and often aggravating.
- bg, fg, jobs, scheduler, and cluster computing are incredible powerful. Learn to use them.
- **free** Linux utilities allow stream-oriented data parsing, cleansing, slicing, and dicing.

Unix Philosophy

- Even longer than Linux, there is a long legacy of free, Open Source tools.
- Typically do a few things but do them well & fast. Input ← STDIN, Errors → STDERR, Output → STDOUT.
- Lots of these tools, developed over 50 yrs of various shells, OS variants, languages.
- The interface tends not to change very much, so learn it once and know it forever.

Excel (gacckkkk) files

- A lot of data is still in Excel files, so..
- Learn how to use it on Linux.
- Via LibreOffice (similar to MS Office)
- Or extract the data and process in pipelines
- With the native app.
- Or via cmdline utilities.

Excel data extraction

Some interchange utilities:

- Tika – interconverts many, many formats. Needs: `alias tika="java -jar /data/hpc/bin/tika-app-1.6.jar"`
- antiword, xls2csv, pdftotext
- The output of these utilities usually need further cleaning with other utilities.
- It never ends...

If you must write ASCII..

- Write delimited, tabular data so it can be parsed more easily.
- Don't replicate data pointlessly.
- Write into large buffers 1st, then write to files in large chunks.
- Truncate floating point values to useful accuracy (23.47063848577682764101945 → 23.47)
  26 bytes vs 5 bytes for no extra value
- Don't confuse high precision with high accuracy.

The line eater: Perl

```
while (<>) {
 $N = @values = split(/token/);
 # some kind of eval
}
```

The line eater: Python

```
import sys
for line in sys.stdin:
    values = data.split('token');
```

Slicing & Dicing ASCII data

- ASCII will be your 1st exp with data on Linux
- ..and before any analysis: Data Cleansing
- Select rows: grep based on a regex
- Select columns: cols, cut/scut
- Often have to merge files
  - Needle and haystack problem (relational join): join, scut
  - Bulk merge: cat, paste, diff, comm, pr

Binary Data

- All data is binary, but...
- Binary storage is a special case of data representation.
- Data is stored as the byte-wise representation of the data, not character-wise
- ie: '123' could be be stored in 1 byte, not 3.
- 9814.98 floating point representation.
  - single precision FP (32b → 4bytes)
  - double precision FP (64b → 8bytes)
  - And even higher (128b)

More Binary Data

- In binary, values are stored without separation tokens so numbers are packed more efficiently as well.
- Some data formats allow specification of the precision of the value so they can use the most efficient representation of the number.

```
163631823645618364912152ducksheepshark387ratthingpokemon
   \   \    \  \        \   \         \  \       \
%3d, %4d, %5d,%3d,%9.4f, %4s,  %10s,   %3d,  %8s, %7s # read spec
```

Compression

- Compression saves disk space and network bandwidth and speed.
- It costs CPU time to both compress and decompress, but compression is much more costly.
- Lossy vs Lossless compression. (JPEG vs gzip)
- ASCII text can be compressed ~ 2-3X
- XML can be compressed ~ 20X
- Random data doesn't compress well at all.

```
$ time dd if=/dev/urandom of=urandom.1G count=1000000
bs=1000
1000000+0 records in
1000000+0 records out
1000000000 bytes (1.0 GB) copied, 82.8871 s, 12.1 MB/s
real    1m22.889s
  ================================================
$ time gzip urandom.1G
real    0m34.601s
  ================================================
$ ls -l urandom.1G.gz
-rw-r--r-- 1 hjm hjm 1000162044 Nov 14 12:03 urandom.1G.gz
---------------------------------------------------------
---------------------------------------------------------
```

So compressing ***nearly random data*** actually results in ***INCREASING*** the file size.

Compressing pure repetitive data from the '/dev/zero' device:

```
--------------------------------------------------------------------------------
$ ls -l zeros.1G
-rw-r--r-- 1 hjm hjm 1000000000 Nov 14 11:31 zeros.1G
    ==========================================================
$ time gzip zeros.1G
real    0m7.100s
    ==========================================================
 $ ls -l zeros.1G.gz
-rw-r--r-- 1 hjm hjm 970510 Nov 14 11:32 zeros.1G.gz
--------------------------------------------------------------------------------
```

So in much less time (7s vs 34s), we get a 1000X compression.

But wait, there's more!

What about bzip2?
It does a much better job:

```
--------------------------------------------------
$ ls -l zeros.1G
-rw-r--r-- 1 hjm hjm 1000000000 Nov 14 11:31 zeros.1G
  ====
$ time bzip2 zeros.1G
real    0m10.106s
  ====
$ ls -l zeros.1G.bz2
-rw-r--r-- 1 hjm hjm 722 Nov 14 11:32 zeros.1G.bz2
--------------------------------------------------
```

Or, about 1.3MillionX compression (about the same as you get if you compress Electronic Dance Music)

More Compression

- Many utilities will enable in-line compression.
- This is fine for small transfers, but for large transfers, it's often better to archive and then use parallel compression.
- pigz – parallel form of gzip
- pbzip2 – parallel form of bzip2
- Both are almost perfectly parallel.

[De]Compression

- If your applications can deal with compressed data, KEEP IT COMPRESSED. Many popular apps (esp bioinfo) now allow this.
- If they can't, try to use pipes (|) to decompress in memory and feed the decompressed stream to the app.
- Use native utilities to examine the compressed data (zcat/unzip/gunzip, grep, archivemount, vitables, ncview, etc.

Moving BigData

- 1st: Don't.
- Otherwise, plan where your data will live for the life of the analysis, have it land there, and don't move it across filesystems.
- Don't DUPLICATE DUPLICATE DUPLICATE BigData
- See: <http://goo.gl/2iaHqD>

- rsync for modified data
- bbcp for new transfers of large single files, regardless of network
- tar/netcat for deep/large dir structures over LANs
- tar/gzip/bbcp to copy deep/large dir structures over WANs

rsync

- If you only want to use one tool, it's rsync.
- rsync -av /from/here /to/there
- Can encrypt and compress data (but don't try to compress already compressed data)
- Specialized variants for multi-TB data.

```
$ rsync -av /this/dir/   /that/DIR
#                  ^
# note that trailing '/'s matter.
# above cmd will sync the CONTENTS of '/this/dir' to
'/that/DIR'
# generally what you want.

$ rsync -av /this/dir   /that/DIR
#                  ^
# will sync '/this/dir' INTO '/that/DIR',
# so the contents of '/that/DIR' will
# INCLUDE '/this/dir' after the rsync.
```

bbcp

- If you only want to use 2 tools, the 2nd one is **bbcp.**
- Used almost like rsync.
- But is much worse for doing recursive copies
- Especially with lots of small files.
- Will compress, but does NOT encrypt data.

```
$ bbcp  bigfile   user@host:/high/perf/raid/file
# can get about 50-60MB/s over 1GbE

bbcp -P 10 -w 2M -s 10 bigfile \
user@host:/high/perf/raid/file
# this can get us 80-110MB/s over 1GbE.
```

Checksums

- Represent the identity of a file. If one **bit** changes, the checksum changes.
- md5sum / jacksum
- Use MANIFEST files & copy them along with the data files.
- See checksum example.
- Integrate checksums as part of your qsub scripts

Timing and profiling

- Only applies to writing your own code, but it's good to start thinking about this early.
- top, atop, free, htop, pstree
- 'time', '/usr/bin/time'
- oprofile, perf, HPCToolkit, valgrind

htop

```
  1  [|                0.5%]   5  [                 0.0%]     9  [                 0.0%]   13 [|                0.5%]
  2  [|                0.9%]   6  [                 0.0%]     10 [||               1.4%]   14 [                 0.0%]
  3  [                 0.0%]   7  [                 0.0%]     11 [                 0.0%]   15 [                 0.0%]
  4  [                 0.0%]   8  [                 0.0%]     12 [                 0.0%]   16 [                 0.0%]
  Mem[|||||||||||||||||||||||||||||22510/258312MB]     Tasks: 377, 323 thr; 1 running
  Swp[|                                 93/4999MB]     Load average: 0.10 0.09 0.03
                                                       Uptime: 43 days, 06:52:31

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
    1 root       20   0 19364  1504  1212 S  0.0  0.0  0:39.50 /sbin/init
31821 crackauc   20   0 57716   452   396 S  0.0  0.0  0:00.00 `- ssh-agent -s
31810 yuw2       20   0  127M  4524  2272 S  0.0  0.0  0:02.39 `- /usr/libexec/gconfd-2
31808 yuw2       20   0 19732   848   620 S  0.0  0.0  0:00.00 `- /bin/dbus-daemon --fork --print-pid 5 --pri
31807 yuw2       20   0 18148   784   544 S  0.0  0.0  0:00.00 `- dbus-launch --autolaunch 7b544a5b335191c2be
31658 yuw2       20   0  105M  1504  1252 S  0.0  0.0  0:00.00 `- /bin/bash /usr/bin/x2goruncommand 157 31038
31803 yuw2       20   0  261M 13412  9924 S  0.0  0.0  5:35.11 |  `- /usr/bin/gnome-terminal
31813 yuw2       20   0  261M 13412  9924 S  0.0  0.0  0:00.00 |     `- /usr/bin/gnome-terminal
31812 yuw2       20   0  106M  1972  1504 S  0.0  0.0  0:00.02 |     `- bash
31811 yuw2       20   0  6548   588   484 S  0.0  0.0  0:00.00 |     `- gnome-pty-helper
31139 ataffard   20   0  287M 14488 10356 S  0.0  0.0  0:30.57 `- gnome-terminal
31142 ataffard   20   0  287M 14488 10356 S  0.0  0.0  0:00.00 |  `- gnome-terminal
31141 ataffard   20   0  105M  1900  1500 S  0.0  0.0  0:00.01 |  `- bash
31140 ataffard   20   0  6548   592   488 S  0.0  0.0  0:00.00 |  `- gnome-pty-helper
31038 yuw2       20   0  140M 70108  5348 S  0.0  0.0  2:30.45 `- nxagent.bin -extension XFIXES -nolisten tcp
30896 ataffard   20   0  521M 16528 12568 S  0.0  0.0  0:22.62 `- /usr/libexec/clock-applet --oaf-activate-ii
30894 ataffard   20   0  282M 10020  7924 S  0.0  0.0  0:06.66 `- /usr/libexec/notification-area-applet --oaf
30883 ataffard   20   0 39004  2644  2228 S  0.0  0.0  0:00.00 `- /usr/libexec/gconf-im-settings-daemon
30803 ataffard   20   0 67784  3092  2364 S  0.0  0.0  0:03.39 `- ./escd --key_Inserted="/usr/bin/esc" --on_S
30751 ataffard   20   0  110M  4760  3948 S  0.0  0.0  0:00.01 `- /usr/libexec/im-settings-daemon
30733 ataffard   20   0  140M  3252  2608 S  0.0  0.0  0:02.16 `- /usr/libexec/gvfsd-trash --spawner :1.7 /or
30729 ataffard   20   0  142M  3320  2688 S  0.0  0.0  0:00.28 `- /usr/libexec/gvfs-gdu-volume-monitor
F1Help  F2Setup F3SearchF4FilterF5SortedF6CollapF7Nice -F8Nice +F9Kill  F10Quit
0-$  1*$ new1  2$ bash  3$ claw4  4$ calw5  5@!$ claw7  6$ bduc      hjm@bduc-login 128.200.15.20 Menu:<F9>
 @  Debian 7.6                                 ^70kb v26kb 1# 45d1h 0.01 4x2.0GHz 15.8GB12% 2014-11-14 17:02:04
```

$ free -g -l

```
             total       used       free     shared    buffers     cached
Mem:           252        244          7          0          0        222
Low:           252        244          7
High:            0          0          0
-/+ buffers/cache:         21        230
Swap:            4          0          4
```

time (bash built-in)

```
$ time ./tacg -n6 -S -o5 -s  < hg19/chr1.fa  > out

real    0m10.599s
user    0m10.456s
sys     0m0.145s
```

/usr/bin/time

```
$ /usr/bin/time ./tacg -n6 -S -o5 -s  < hg19/chr1.fa  > out

10.47user 0.14system 0:10.60elapsed 100%CPU
(0avgtext+0avgdata 867984maxresident)k
0inputs+7856outputs (0major+33427minor)pagefaults 0swaps
```

oprofile

```
$ operf ./tacg -n6 -S -o5 -s  < hg19/chr1.fa  > out
operf: Profiler started

$ opreport --exclude-dependent --demangle=smart --symbols ./tacg
Using /home/hjm/tacg/oprofile_data/samples/ for samples directory.
CPU: Intel Ivy Bridge microarchitecture, speed 2.501e+06 MHz

samples  %        symbol name
132803   43.1487  Cutting
86752    28.1864  GetSequence2
49743    16.1619  basic_getseq
9098      2.9560  Degen_Calc
7522      2.4440  fp_get_line
7377      2.3968  HorribleAccounting
6560      2.1314  abscompare
4287      1.3929  Degen_Cmp
2600      0.8448  main
704       0.2287  basic_read
212       0.0689  BitArray
112       0.0364  PrintSitesFrags
3        9.7e-04  ReadEnz
3        9.7e-04  hash.constprop.2
2        6.5e-04  hash
1        3.2e-04  Read_NCBI_Codon_Data
1        3.2e-04  palindrome
```

Big Data

- Volume
  - Scary sizes, and getting bigger
- Velocity
  - Special approaches to speed up analysis
- Variety
  - Domain-specific standards (HDF5/netCDF, bam/sam, FITS), but often aggregations of unstructured data
- No one-technique-fits-all, but will present general techniques that should help with a number of approaches.
- BigData Hints for Newbies
  <http://goo.gl/aPj4az>

Big Data – How Big is Big?

# Bytes	Byte name / Abbriev'n	Approximation
1/8	bit (b)	0 or 1: the smallest amount of information.
1	Byte (B)	8 bits, the smallest chunk normally represented in a programming language.
2^{10}	1,024 B (1 KB)	a short email is a few KBs
2^{20}	1,048,576 B (1 MB)	a PhD Thesis ; Human Chr 1 is ~250 MB
2^{30}	1,073,741,824 B (1 GB)	the Human Genome is 3,095,693,981 B (optimized, ~780 Mb @ 2b/base) ; a BluRay DVD holds 25GB per layer (most movie BluRays are dual-layer = 50GB); a Genomic bam file is ~150GB
2^{32}	4,294,967,296 (4GB)	**fuzzy border between SmallData (32b) and BigData (64b)**
2^{40}	1,099,511,627,776 B (1 TB)	1/10th Library of Congress (LoC); the primary data fr. an Illumina HiSeq2K is ~5 TB
2^{50}	1,125,899,906,842,624 B (1 PB)	100X LoC; ~HPC's aggregate storage; ~100 PB is the yearly storage requirements of YouTube.
2^{60}	1,152,921,504,606,846,976 B (1 EB)	the est. capacity of the NSA's data facility is ~12 EB

Integer Byte Sizes

word size	#bits	range of variable
byte or char	8	256
int	16	65,536
long int	32	4,294,967,296
long long int	64	1.84467440737e+19

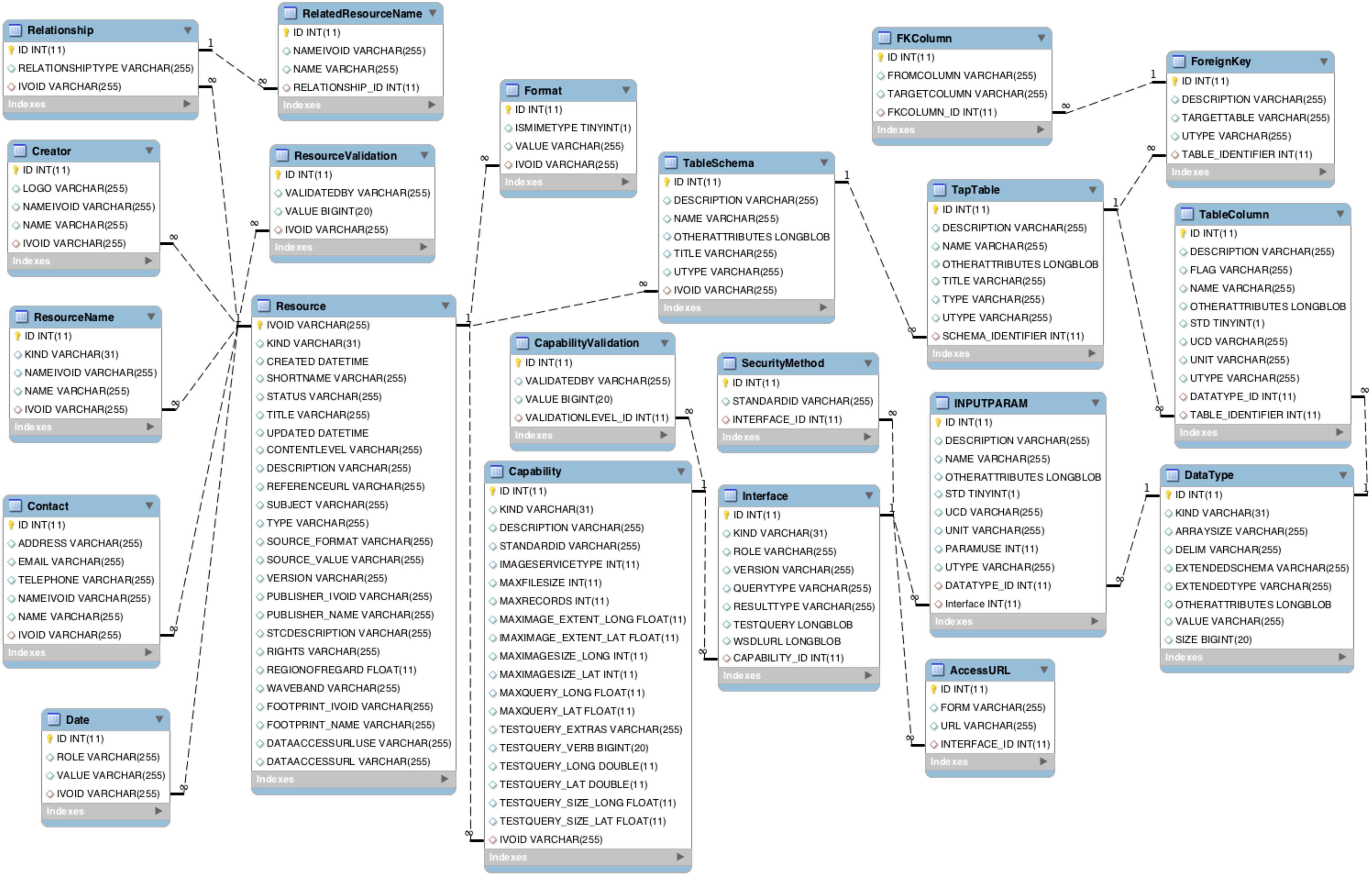
Data Types

- Alphanumeric Strings
  “the rain in spain is green”
- Integers
  12, 4, 126987, -4432, 2014, 0
- Floats
  -234.2987, 3.633E17, 5.51e-5
- Booleans
  1, 0, T, F,
- Vectors of above

Processing BigData

- Files (HDF5, bam/sam) and specialized utilities (nco/ncview, [Py/Vi]tables, R, Matlab)
- Relational Dbs (SQLite, Postgres, MySQL)
- NoSQLs (MongoDB, CouchDB)
- Binary Dumps (Perl's Data::Dumper, Python's pickle)
- Non-Storage (pipes, named pipes/FIFOs, sockets)
- Keep it RAM-resident.

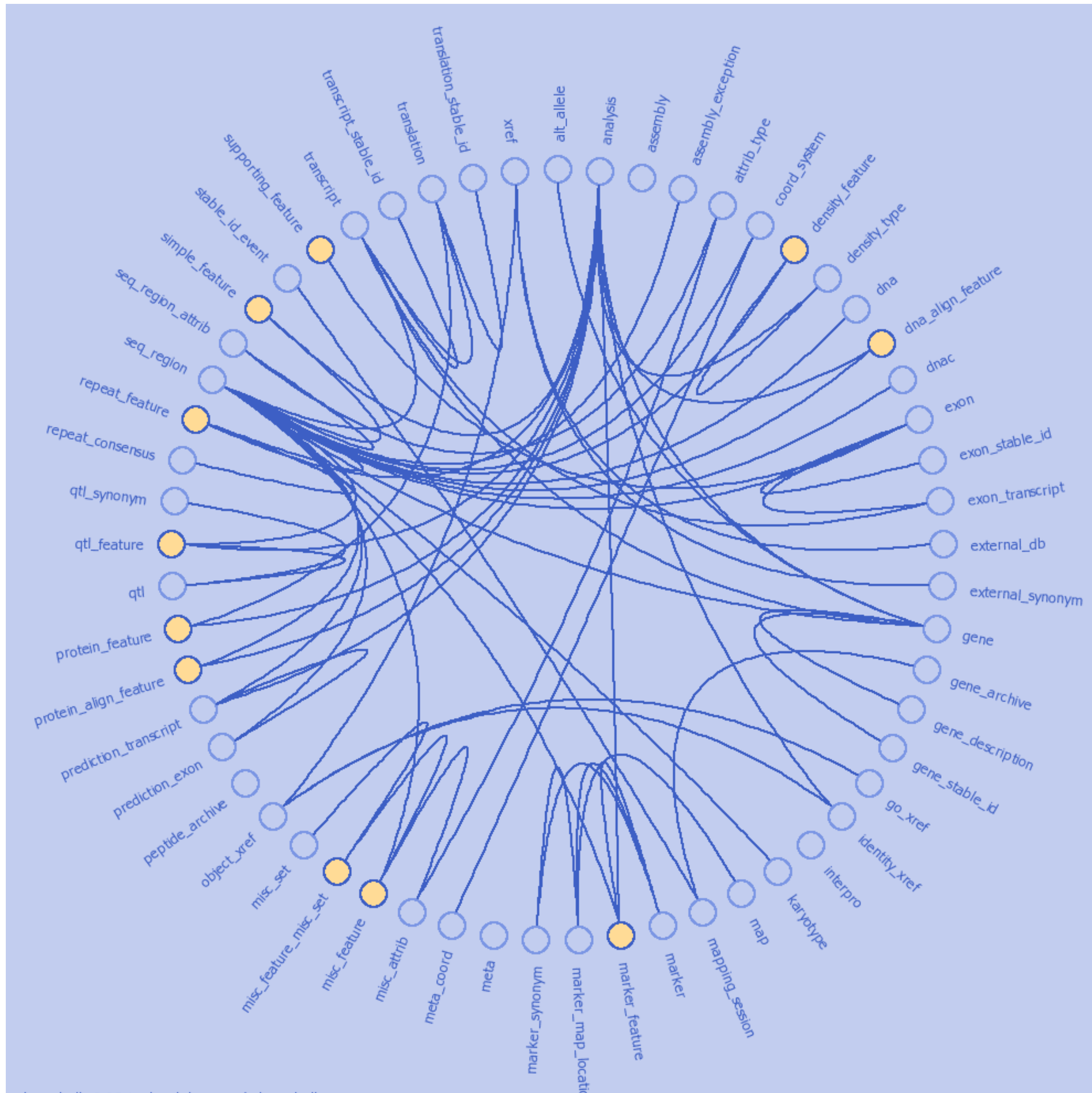
Formal Relational Schema



EMBL String DB Schema

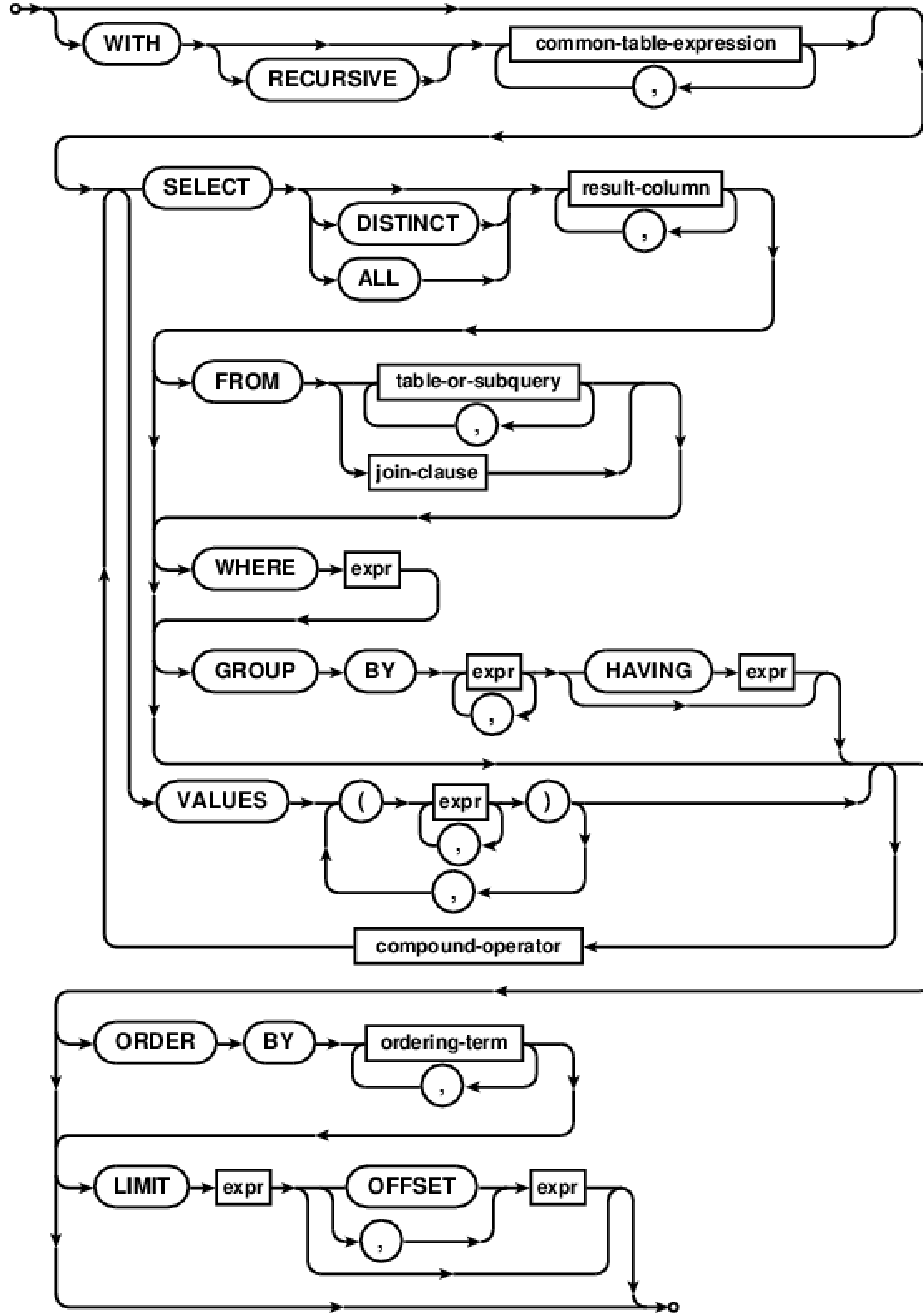
http://string71.embl.de/newstring_download/database.schema.v7.1.pdf

Schematic Schema (Circos)



Querying an RDB with SQL

- Structured Query Language (SQL) is a formal query language for admin'g RDBs & specifying relationships across tables.
- Ugly, unintuitive, but very powerful.
- **Select** statements will be your entry to SQL



Fomal grammar flowchart
of the SELECT clause.

Select Example:

SQL Statement:

Edit the SQL Statement, and click "Run SQL" to see the result.

```
SELECT * FROM Customers;
```

Run SQL »

Result:

Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain

Your Database:

Tablename	Records
Customers	91
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29

Restore Database

http://www.w3schools.com/sql/default.asp

Queries: http://goo.gl/S5L3fE

NoSQL Databases

- A BigData-driven development
- Designed for ***Scale and Speed*** over reliability.
- Most designed to shard or distribute ops
- Not really designed for relational operations.
- Many designed for *Key:Something* mappings
- Many are not ACID (Atomic, Consistent, Isolated, Durable).
- Many variants now available, many OSS.

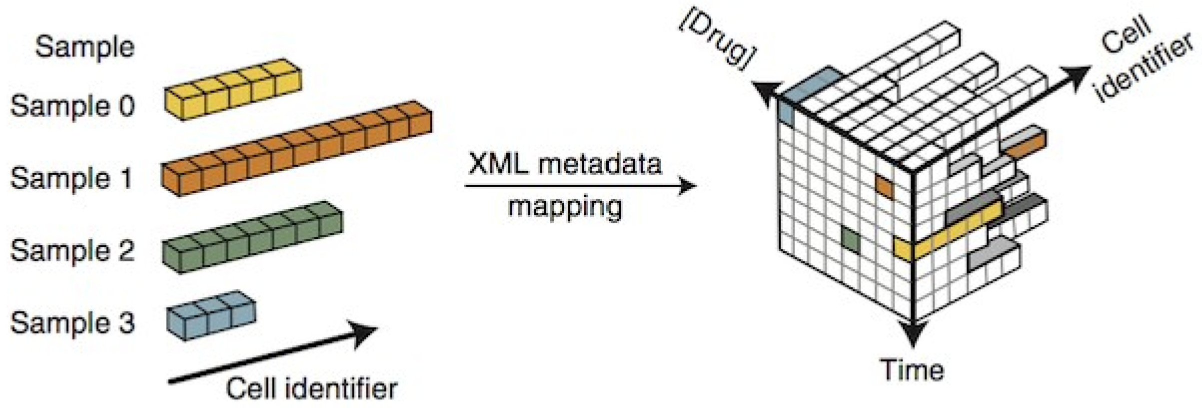
Slicing & Dicing Big Data

- Use format-specific tools. At this scale, *cut, grep*, etc don't work so well.
- ncview, nco, for netCDF
- h5py, pytables, vitables, R, hdfview, for HDF
- well-documented APIs for most languages; even specific books.
- Writing and reading such formats is not as hard as it might appear.
- These formats are just data containers, much like ASCII files.

HDF5 Internal Structure

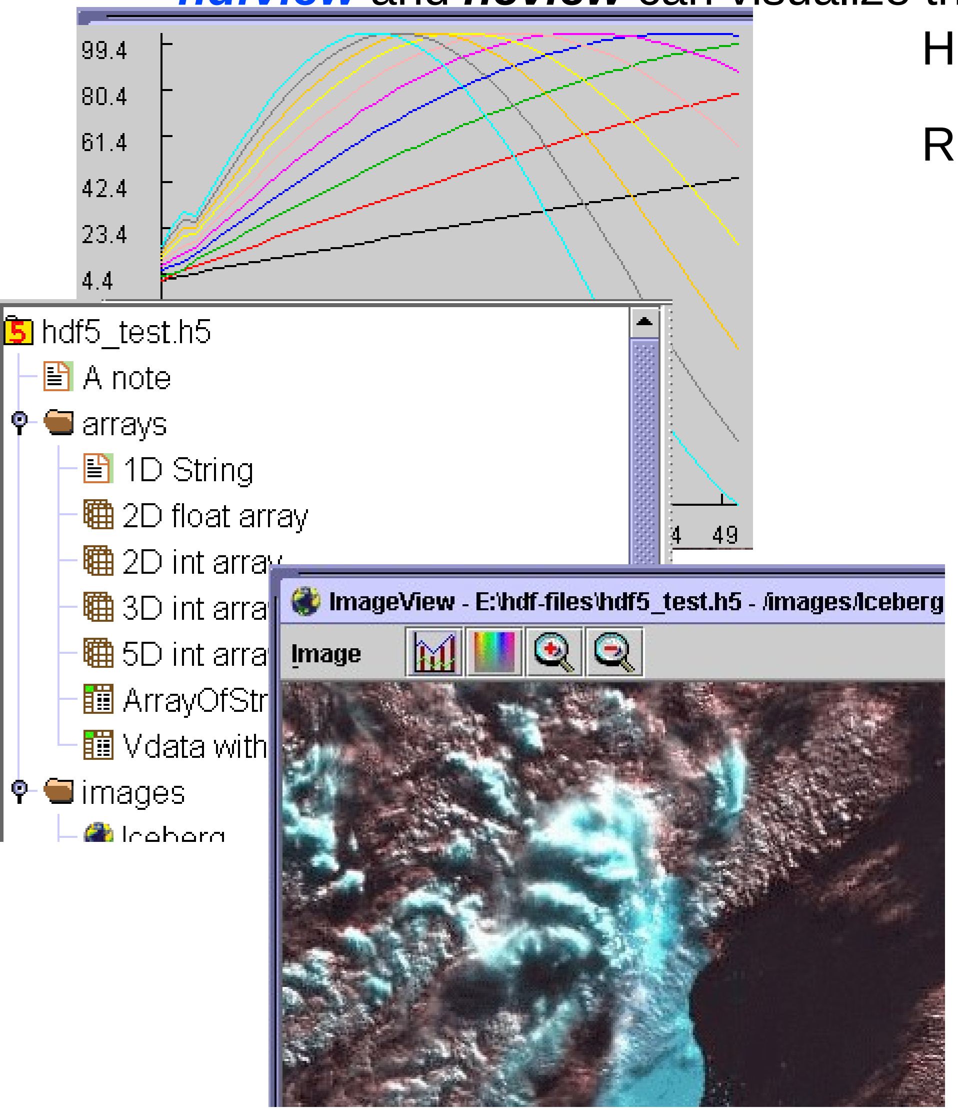
- **Datasets:** arrays of homogeneous types – int's, floating points, strings, bools.
- **Groups:** collections of 'Datasets' or other 'Groups', leading to the ability to store data in a hierarchical, directory-like structure, hence the name.
- **Attributes:** Metadata about the Datasets, which can be attached to the data. Internal or external. (as with XDF or SDCubes).

HDF5 file format



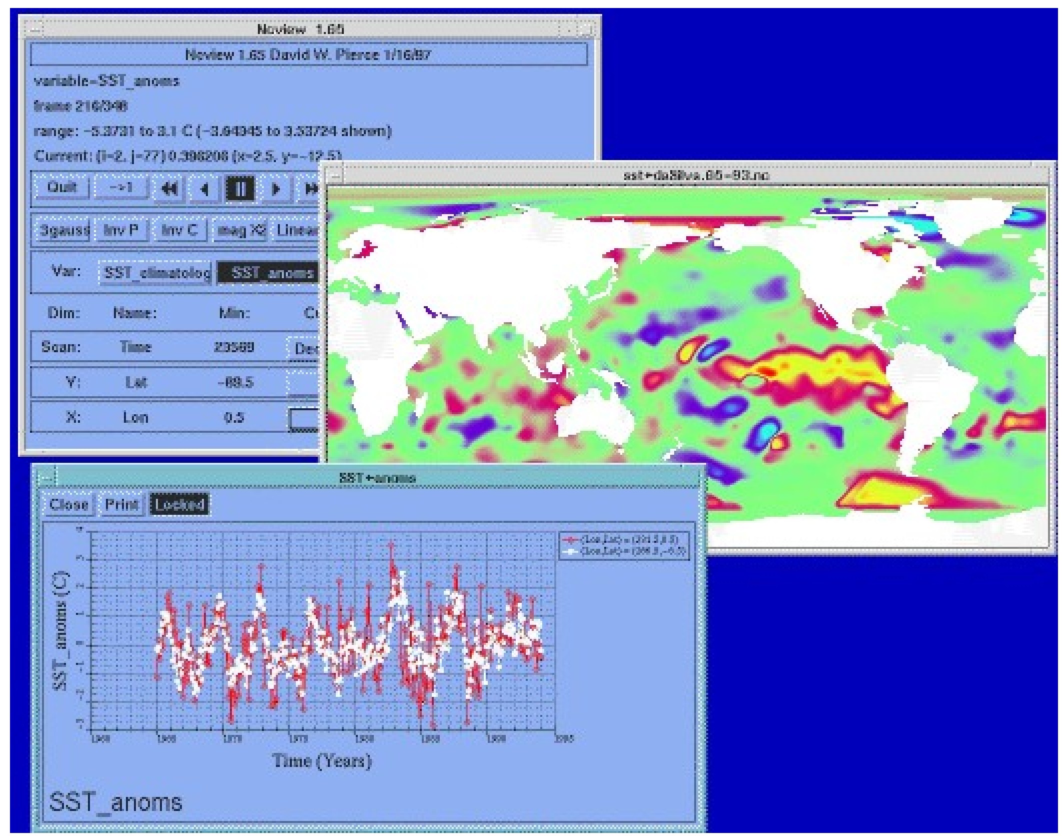
HDF5 visualizers

hdfview and ***ncview*** can visualize the layout and data of HDF5 & netCDF files



HDF5 used as primary storage for PacBio data

R can read HDF5 files with ***h5r*** and ***pbh5***



Relational vs Hierarchical

- HDF5 (& similar formats) are designed to allow large amounts of numerical data to be read and written (and re-written).
- Relational Databases are designed to answer relational queries and allow small, fast data inserts and modifications.
- These 2 approaches are quite different
- Be careful which approach you take.

Optimization

- To process BigData, you need efficient code.
- To find inefficient code, you profile it.

- **'time'** vs '**/usr/bin/time -v'**
  - gross overview of how long it tool
- **Oprofile**
  - Easily gives you per-function time sinks
- **HPCToolkit**
  - Per-line time & hardware counter execs

BigData needs Parallelism (//)

- The bigger the data, the more you need //ism.
- **Easy:** what's given to you on the cluster.
  - // filesystem.
- **Pretty Easy:** Splitting your analysis & data into independent streams & chunks.
  - Using SGE, Job Arrays, // functions, and all the spare cores on the cluster.
- **Damn Hard:** Writing your own programs to do analysis in //, using OpenMP, MPI, CUDA, OpenCL, Julia

Embarrassingly Parallel (EP)

- Where the analysis of any chunk of data is independent of the analysis of any other chunk.
- Break the data into equal sized pieces and spread them out over all the CPUs you can.
- aka Single Process, Multiple Data (SPMD)
- more loosely: Scatter/Gather
- What GPUs are REALLY good at.

Hadoop / MapReduce

- Special cases where you have EP jobs and lots of cores to throw at it.
- Hadoop is actually the underlying parallel FS
  - Not a general-purpose FS; not POSIX (and HPC already has a // FS).
  - MapReduce (~Producer / Consumer model)
    - Map decomposes the data into required form.
    - Reduce does the analysis.

Map(Shuffle)Reduce

- **Map:** Each worker node applies the "map()" function to the local data, writes the output to a temporary storage (HDFS). A master node orchestrates that for redundant copies of input data, only one is processed.

- **Shuffle:** Worker nodes redistribute data based on the output keys (produced by the "map()" function), such that all data belonging to one key is located on the same worker node.

- **Reduce**: Worker nodes now process each group of output data, per key, in parallel.

Hadoop improvements

- Spark – more sophisticated, in-memory analytics engine (replaces MapReduce)
- Hive – Data warehouse built on top of HadoopFS
- Shark – Spark on Hive
- Pig – Language (PigLatin) for automating the production of MapReduce programs – sort of an SQL for MR pipelines.

Many of these technologies require Hadoop-ish semantics, but HPC already has a fast // FS and Hadoop can be emulated on top of the exiting FS.

BigData, not ForeverData

- HPC is not backed-up.
- Cannot tolerate old, unused BigData.
- RobinHood is looking for your old BigData.
- Please help us by doing your own data triage.
- Ask your PIs to bug our boss to provide more resources so we can provide more resources.

Visualizing BigData

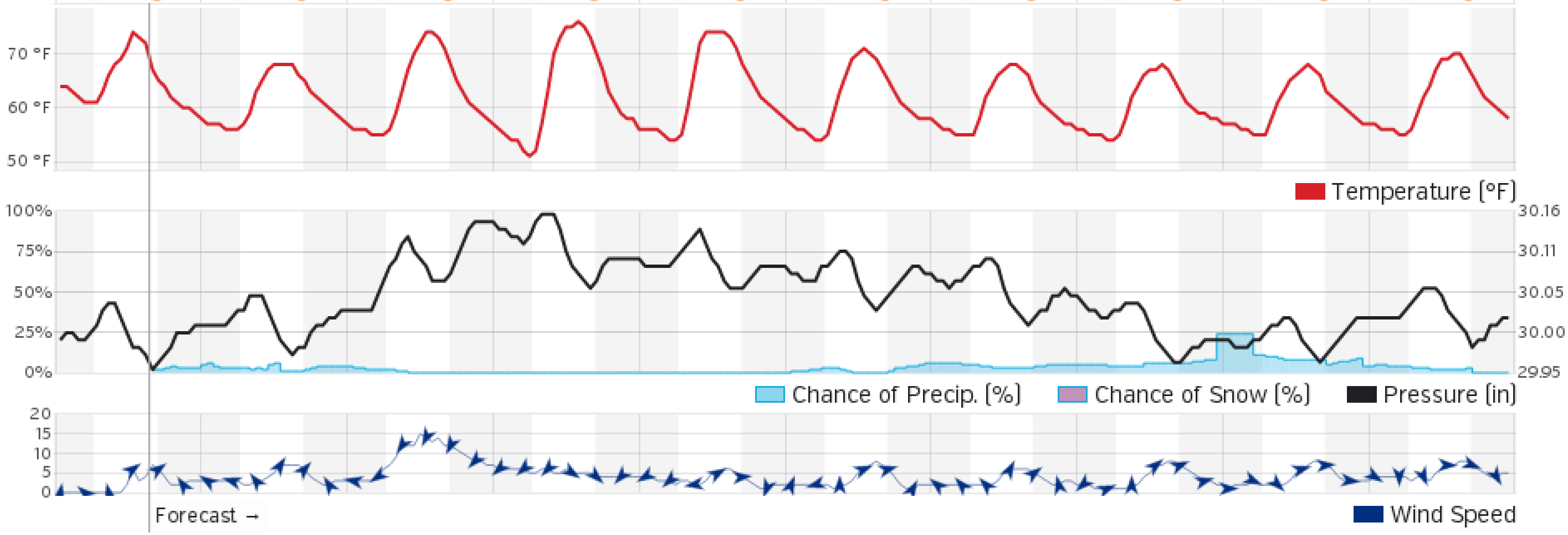
- Lots of points means special apps for visualizing them.
- Visualization techniques for mapping variables onto color, texture, symbol types and sizes, transparency, vectors, time series, maps, interactivity
- Wunderground, gapminder, Circos, gephi

10-Day Weather Forecast

Graph | Table | Descriptive

Daily | Hourly | Customize

Fri 11/14	Sat 11/15	Sun 11/16	Mon 11/17	Tue 11/18	Wed 11/19	Thu 11/20	Fri 11/21	Sat 11/22	Sun 11/23
74° | 55°	69° | 54°	76° | 50°	78° | 53°	74° | 53°	71° | 54°	70° | 53°	70° | 54°	70° | 54°	72° | 53°
Partly Cloudy	Clear	Partly Cloudy	Clear	Partly Cloudy	Clear	Partly Cloudy	Clear	Partly Cloudy	Partly Cloudy
0 in	0 in	0 in	0 in	0 in	0 in	0 in	0 in	0 in	0 in



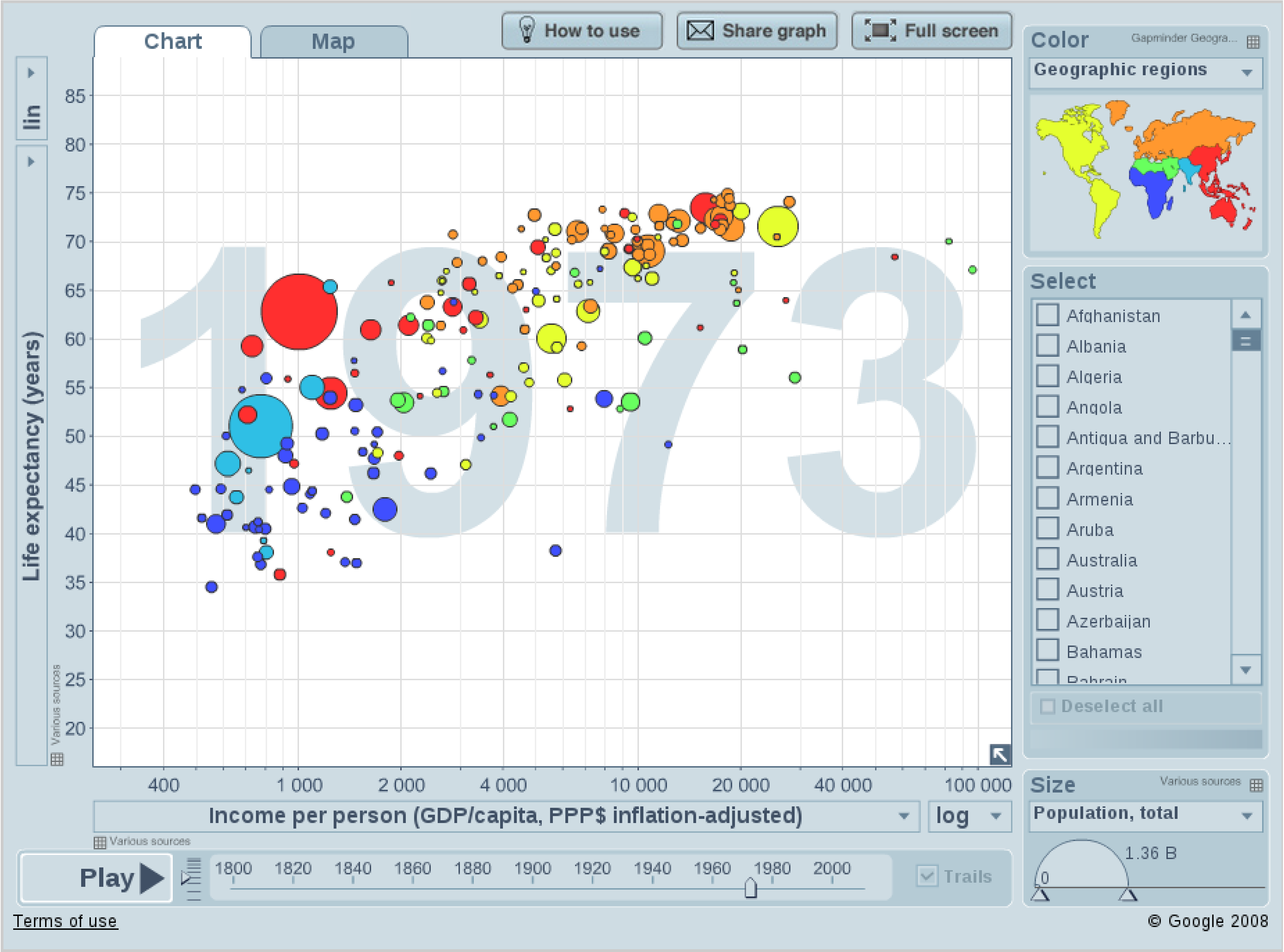
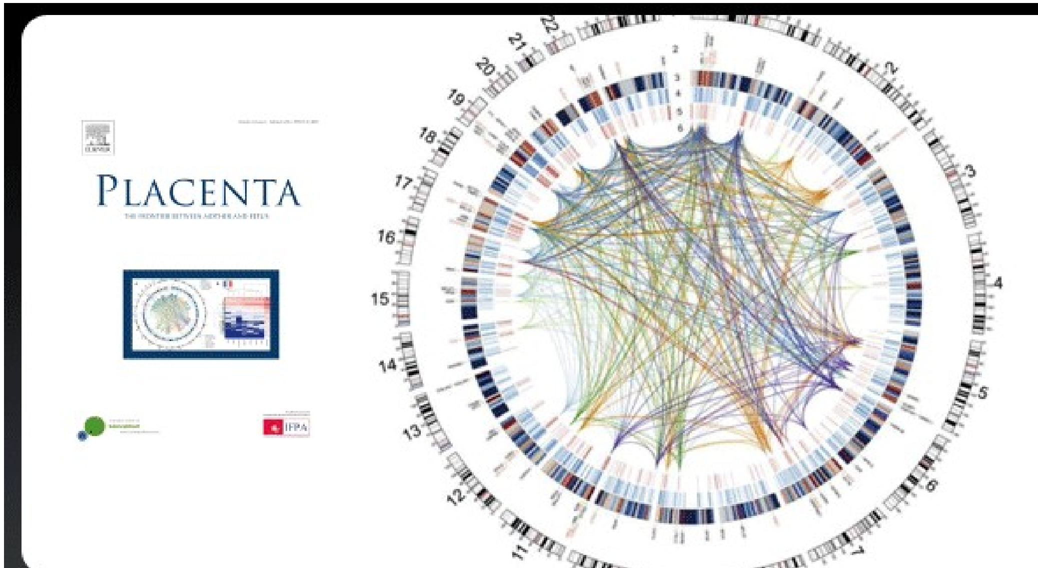


Chart
Map
How to use
Share graph
Full screen
lin
Life expectancy (years)
85
80
75
70
65
60
55
50
45
40
35
30
25
20
Various sources
1973
400
1 000
2 000
4 000
10 000
20 000
40 000
100 000
Income per person (GDP/capita, PPP$ inflation-adjusted)
log
Various sources
Play
1800
1820
1840
1860
1880
1900
1920
1940
1960
1980
2000
Trails
Color
Gapminder Geogra...
Geographic regions
Select
Afghanistan
Albania
Algeria
Angola
Antigua and Barbu...
Argentina
Armenia
Aruba
Australia
Austria
Azerbaijan
Bahamas
Deselect all
Size
Various sources
Population, total
1.36 B
0
Terms of use
© Google 2008

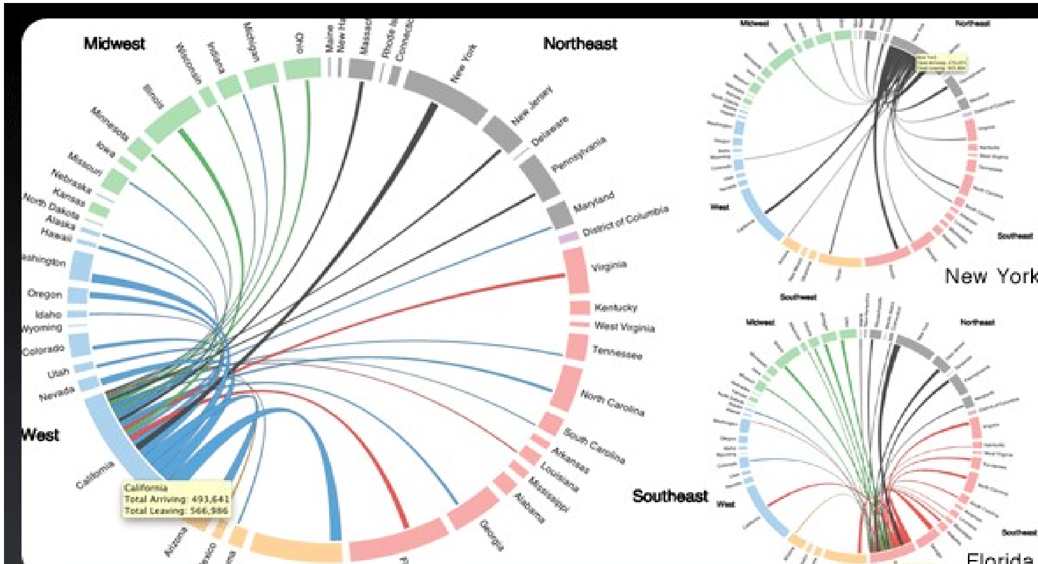
Circos visualizations



CIRCOS CHARTS THE PLACENTA TRANSCRIPTOME

Saben *et al.* use Circos to visualize the transcriptome and gene expression of placenta from 20 healthy women in their article [A comprehensive analysis of the human placenta transcriptome](#).

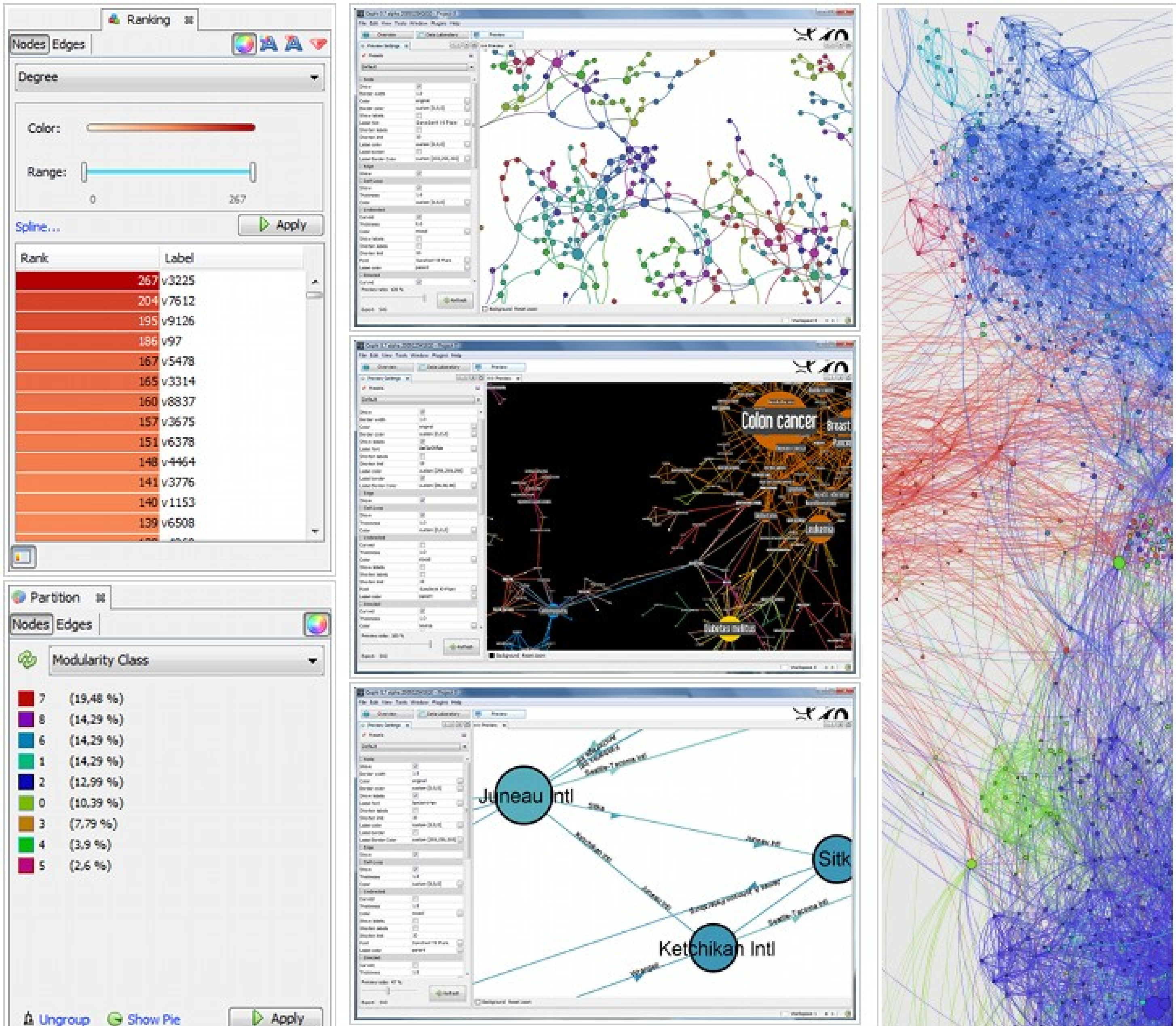
Saben J, Zhong Y, McKelvey S *et al.* (2014) [A comprehensive analysis of the human placenta transcriptome](#) *Placenta* **35**:125-131.



CIRCOS MAPS AMERICA'S RESTLESS INTERSTATE MIGRATION WITHOUT A MAP

Wired has a [writeup about migration patterns within the US](#) that shows the data using d3.js [chord diagrams](#), modeled after how [Circos shows tabular data](#).

Gephi Visualizations




00:00 - 09:00

Visualization Apps

- Simple Data Visualization
  <http://goo.gl/TNJv8h>
- Multivariate Data Visualization
  <http://goo.gl/32AXAO>
- Roll your own with
  <https://processing.org>