Performing Biology Research on the Odyssey Cluster

Amir Karger

Life Sciences Research Computing

rchelp@fas.harvard.edu

http://software.rc.fas.harvard.edu/training/bio_cluster

FAS

INFORMATION TECHNOLOGY

Outline

- Commercials and Annoying Reminders
- Cluster: modules, queues, LSF, storage
- BLAST – serial
- The Scriptome – simple data munging
- BLAST – "fake" parallel (Job Array)
- MrBayes – serial and "really" parallel
- More software & resources
- Your questions?

Why?

- Why computers?
  - Big data sets, hard math, boring repetition
- Why cluster?
  - High throughput, shared resources
  - Run jobs in parallel (different kinds of parallel)
- Why Research Computing?
  - Knowledge (computer geeks who know science)
  - Experience (we' ve made mistakes already)
  - We worry about computers so you can do biology
    - Backup, security, software installation, network, data analysis

FAS INFORMATION TECHNOLOGY

Talk to us!

- Talk to us **before** you do lots of work
- Save time
  - We can automate, make code run faster
- Save effort
  - Maybe we’ve worked on a similar problem before?
  - Or we know someone else who has?
- Do better science?
  - A more appropriate program, an overlooked parameter

- This is the most important slide of the day

FAS INFORMATION TECHNOLOGY

Annoying Reminders

- Tickets
  - Research questions to rchelp@fas.harvard.edu
  - Other questions to help@fas.harvard.edu
  - Put individual RC staff in the message if you want
- Don't share cluster passwords
  - Really.
  - Not even with us.
- FAQ etc.: http://rc.fas.harvard.edu
- Class site: http://isites.harvard.edu/icb/icb.do?keyword=k60501

Cluster Vocabulary and Usage

- Node: one computer in the cluster
- Head node: iliadaccess01, 02, 03
  - If you ssh/PuTTY/Terminal/sftp to odyssey.fas, you get here
  - Do **not** run long programs here (They'll die)
  - **Do** submit (long or short) jobs from here
- Interactive nodes: `bsub -q interact -Is bash`
  - good for testing 5-minute runs, interactive Matlab
  - Don't submit new jobs from here. "exit" and submit from head nodes
- `http://rcnx.fas.harvard.edu` - graphical cluster login
- Core: one "processing unit" (sort of)
  - Each node on Odyssey has 2-8 cores, so it can run 2-8 jobs

Storage

- Lab folders
  - Located in /n, /n/Lab_Folders - **stable** (maybe backed up)
  - /n/data, /n/data1, /n/nobackup1 or 2, etc. - **less stable**
  - Often accessible from Windows/Mac (on VPN, but not Wi-fi)
  - Users, Group, LSDIV/Everyone (WWW, …)
  - Your PI can buy backed-up or scratch storage (some free?)
- Local /scratch on nodes
  - Faster to write temporary output to, some space per node
  - Not visible from head nodes (so copy final output files)
- Large file transfer
  - http://fta.fas.harvard.edu

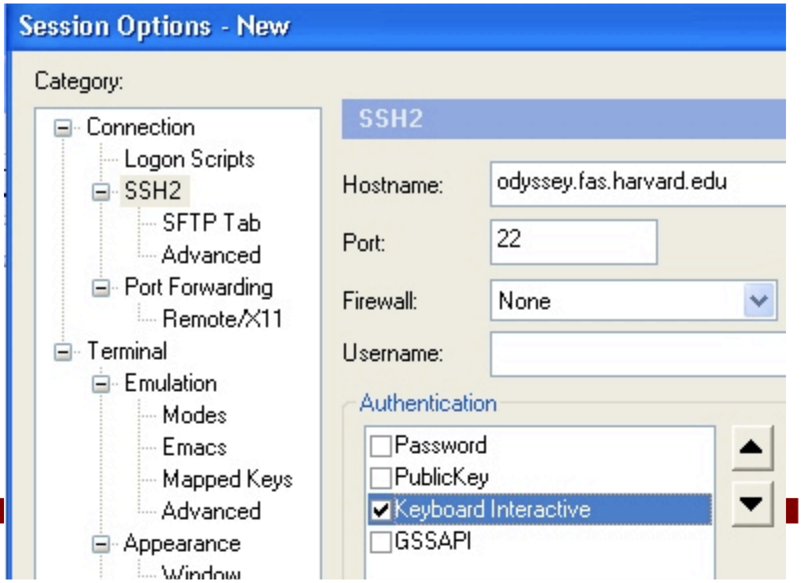
Memory

- Storage: a place to put data files
- Memory: (RAM) needed to run programs with big data sets
- Different nodes have different amounts of memory
  - bsub -R will let you ask for big memory if you need it
- Running out of memory can make jobs crash
  - Contact rchelp@fas and forward the LSF crash email

Cluster login - from Windows

- Login to odyssey.fas.harvard.edu
  - Use PuTTY or SecureCRT
  - Type host name odyssey.fas.harvard.edu (make sure port is 22)
  - Open. Enter password, hit return. Enter fob passcode, hit return
  - SecureCRT only: Set KeyboardInteractive should be the ONLY checked option on the SSH2 options page

- You can't use the same fob passcode twice
  - Even in two different windows!
  - Beware lockouts



Hostname: odyssey.fas.harvard.edu
Port: 22
Firewall: None
Username:

Authentication
- ☐ Password
- ☐ PublicKey
- ☑ Keyboard Interactive
- ☐ GSSAPI

Cluster login - from Mac

- Login to odyssey.fas.harvard.edu
  - Use the Terminal application
  - Shell->New Remote Connection, Secure Shell (ssh) service
  - Select server odyssey.fas.harvard.edu (or add it)
  - Enter user name and click Connect
  - Enter password, hit return.
  - Enter fob passcode, hit return

- You can't use the same fob passcode twice
  - Even in two different windows!
  - Beware lockouts

Getting Sample Data

- Work in your home directory or cd to your lab folder
- Copy workshop sample data
  - `cp -r /n/nobackup2/workshop_bio ./workshop_bio`
  - `cd workshop_bio`

Modules

- Give you access to new commands
  - Load a module to be able to run BLAST
  - One module may give access to many commands
- Set environment variables
  - How does BLAST know where to find nr database?
- Possibly load other modules
  - Parallel MrBayes needs a “run in parallel” module
- Simplify our life and yours
  - Fewer PATH conflicts, simpler process

Modules Commands

- `module avail`
  - What modules are available (Long list!)
  - `module avail hpc/bla` shrinks the list
  - We're gradually moving many bio modules to `bio`/
- `module keyword -i blast`
  - Search *description* (not perfect - ask us)
- `module load hpc/blastall`
  - Get functionality
  - `module unload` may help avoid conflicts

Modules Commands II

- `module list`
  - What modules have I loaded?
- `module display hpc/blastall`
  - Tells you what the module does
  - (I.e., which environment variables are set, etc.)
- Automatic module loads at login
  - You can put module load commands at the end of your `~/.bashrc`

Don't Break the Cluster

- Submitting > 500 jobs
  - Always try 3-5 jobs first
  - Talk to us the first time you plan to do this
- `echo "useful file" > ~/.lsbatch`
  - Makes LSF put temporary output in local /tmp
  - Faster, and keeps you from filling up ~
  - You may first need to (carefully) `rm -rf ~/.lsbatch`
- Writing lots of data
  - Your lab folder
  - /n/nobackup*
  - local /scratch (Make sure to copy stuff you need!)

Exercises: Cluster Intro

- `echo "useful file" > ~/.lsbatch`
- Find your lab folder
- Play with `module avail`, etc.
  - Find your favorite program (mrbayes, beast, BayesPhylogenies, velvet, genscan, maq, …)

Running Software X on Odyssey

- (Email `rchelp@fas` to download/create a module)
- Load the appropriate module
```
module load hpc/something
```
- Test: run the program on a tiny example
- Make a new directory in your lab folder & cd to it
- Write a bsub script called, say, `my_script.bsub`
  - Or copy an old one and change it
  - Reproducible science!
- Submit the job (don't forget the < sign!)
```
bsub < my_script.bsub
```

FAS INFORMATION TECHNOLOGY

BLAST on Odyssey

- `cd blast_serial`
- Load the module
  - `module load hpc/blastall`
  - Also lets you use formatdb, fastacmd
- Test: run the program on a tiny example

```
blastall -p blastn -i Scer_2.fasta -m8 -o
Scer_2.m8 -d ../blastdb/fungi -e 1e-10 -b 25 -v 25
```

- What?!

BLAST Options

- Command-line BLAST is just like the website

```
blastall -p blastn -i Scer_2.fasta -m8 -o
Scer_2.m8 -d ../blastdb/fungi -e 1e-10 -b 25 -v 25
```

- `-p`: BLAST type (blastp, blastn, blastx, …)
- `-i`: input file (Scer_2.fasta)
- `-o`: output file (Scer_2.m8, or Scer_2.blast)
- `-e`: Max. E-value (set based on query/db)
- `-d`: database (default nr, looks in BLASTDB)
- `-m`: output format (see next slide)
- `-b`/`-v`: max hit sequences/alignments per query
- Many others: "`blastall -`" gives a summary

BLAST Output Formats

- `-m0` (or no `-m` option): long text
  - Looks like website, without colors & links
- `-m8`: tabular ("hit table")
  - Throw into Excel, use with the Scriptome
- `-m9`: tabular with comments
  - See column names (but harder to script)
- `-m7`: XML
  - Long. Used in blast2go tool, e.g.
- etc.

bsub from the Command Line

- Just type “bsub” and then the command

  ```
  bsub blastall -p blastn -i Scer_2.fasta -m8 -o
  Scer_2.m8 -d ../blastdb/fungi -e 1e-10 -b 25 -v 25
  ```

  – Runs in your default queue (normal_serial? Your lab’s queue?)

  – Better to type `bsub -q short_serial blastall -p …`

- bsub flags vs. program flags
  – bsub flags: anything **before** the program name
  – program flags: anything **after** the program name

- Now watch job with bjobs, kill with bkill, etc.

bsub Script

```
# Options to bsub go here.
# DON'T put BLAST options here!
# Lines starting with # are comments
# EXCEPT lines with #BSUB are options to bsub
#BSUB -q short_serial

# Command: whatever you would type on command line
blastall -p blastn -i Scer_2.fasta -m8 -o Scer_2.m8
  -d ../blastdb/fungi -e 1e-10 -b 25 -v 25
```

Fancier bsubs

- Output file: `-o` (sort of like blastall -o)
  - Send mail despite `-o`: `-N`
  - (Otherwise, all the output gets mailed to you!)
- Error file: `-e` (NOT like blastall -e)
  - STDERR, "error output" vs. STDOUT, "regular output"
- Resource request: `-R "mem > 15000"`
  - Contact RC or `man bsub` about other -R options
- Name your job: `-J "some name"`
  - Also for job arrays
- Rerunnable (if a machine goes down): `-r`
  - Does NOT restart if a job dies
  - Careful: always starts from the beginning

bsub Script with Options

```
# Don't put BLAST options up here!
#BSUB -q short_serial
#BSUB -e blast_simple.err
# Make sure to email me at below address
#BSUB -N
#BSUB -u akarger@cgr.harvard.edu
#BSUB -J easy_blast

# Whatever you would type on command line
blastall -p blastn -i Scer_2.fasta -m8 -o Scer_2.m8
  -d ../blastdb/fungi -e 1e-10 -b 25 -v 25
```

formatdb

- `cd ../formatdb`
- Format a database for running BLASTS
  - my.fasta ➔ my.nhr, my.nsq, … (or .phr, .psq, …)
  - Now blastall … -d my (if my.n* are in . or BLASTDB)
  - Or full path: -d ~/dir1/dir2/my for ~/dir1/dir2/my.n*
  - Only formatdb once, then BLAST many times
- Note: RC already has nr, nt, swissprot, …
- Indexing your database: must have "nice" IDs

formatdb Options

```
formatdb -p F -o T -t "Fungal ORFs (DNA)" -n
  fungi -i fungi_orfs.fasta
```

- `-p T` to format a protein database, `-p F` for DNA
- `-t` Title for the database (use quotes)
- `-n` Database name (what you use in blastall -d)
- `-i` Input file
- `-o T` Index (lets us search database with fastacmd)

Might need to bsub formatdb for huge databases

fastacmd

- `cd ../fastacmd`
- Get FASTA sequences from a BLAST database
  - `fastacmd -d ../blastdb/fungi -s "lcl|Calb--orf19.10, lcl|Calb--orf19.100"`
  - `fastacmd -d ../blastdb/fungi -i ids.in -o out.fasta`
- Or get information on the database
  - `fastacmd -d ../blastdb/fungi -I`
  - Gives title (formatdb -t), size, date created
- You got fastacmd and formatdb when you loaded the blastall module

FAS INFORMATION TECHNOLOGY

Checkpointing, aka, insurance

- Checkpoint: save your job every N minutes
  - Extremely useful for three-week jobs
  - Also good if your job gets suspended for a long time
  - Don't use N < 30 - too big a strain on resources

```
# Checkpoint, save every 60 minutes. Don't forget ""
#BSUB -k "myblast.ckpt 60 method=blcr"
export LD_PRELOAD=libcr_run.so.0 # Goes BEFORE blastall
blastall …
```

- If job dies (or you bkill it), you can restart it
  - Go into the same directory you ran job from originally
  - `brestart myblast.ckpt`

Exercises: blastall

- Play with blastall
  - **Change the email address in the bsub scripts!**
  - Blast one or two input sequences against nr (slow)
  - Try bjobs, bkill, etc.
  - Blast with different E-values
  - Blast with different output formats
- Play with formatdb
  - Create a one-fungus database from a FASTA file in /n/bluearc/mol/seq/fungi/ORFs/coding_orf/
  - Or a protein database: /n/bluearc/mol/seq/fungi/ORFs/trans
  - Now you can run blastx

Introducing the Scriptome

- Biologists need to merge/transform/filter/sort data
  - A lot of data (maybe too big or badly formatted for Excel)
  - Wide variety of formats, questions, …
  - Most biologists aren't programmers
- Q: Can non-programmers "munge" data?
- A: The Scriptome
  - A cookbook of simple "data munging" tools
  - No programming
  - No install (Windows: one-click ActiveState install)
  - (Almost) no memorization or learning

Using the Scriptome

- sysbio.harvard.edu/csb/resources/computational/scriptome
  - or Google scriptome
- Using a tool
  - Pick a tool type
  - Browse table of contents to get a tool (or use quickbrowse)
  - Change parameters and filenames as needed
  - Expand code to see how it's done (optional)
  - Cut and paste to command line
- Find BLAST results with > 96% identity
  - Use column 2, not 3 (first column is 0)
- Build a protocol (or use an existing one)

Command-Line Scriptome I

- `cd ../scriptome`
- `module load bio/hpc_data_tools`
- List all "change" tools on the Scriptome website

  ```
  Scriptome -t change
  ```
- Run a tool

  ```
  Scriptome -t change_fasta_to_tab
  Scer_redundant.fasta > redundant.tab
  ```

Command-Line Scriptome II

- Program will ask you for parameters, if needed

```
Scriptome -t choose_cols redundant.tab >
some.tab
```

  - Voilà! Easy way to get FASTA IDs
- Or set parameters on command line: scriptable

```
Scriptome -t choose_cols -p '@cols=(1, -1, 3)'
ordered.tab > reordered.tab
```

- ScriptPack (Resources page)
  - Scriptome for your laptop
  - Replace "Scriptome" in commands above with "ScriptPack"
  - Note: won't get updated tools from the website

Scriptome Examples

- Manipulate FASTAs
- Filter large BLAST result sets
- Merge gene lists from different experiments
- Translate IDs between different databases
- Calculate 9000 orthologs between two species of *Drosophila*

- Contact RC about using Scriptome
  - Or about something Scriptome-ish that Scriptome can't do

Exercises: Scriptome

- Remove duplicate sequences from `Scer_redundant.fasta`
- Change FASTA file to tab, then get ID column (or description colum)
- Sort `ordered.tab` by gene start position
- Protocol: remove sequences < 500 bp
- Try exercises using command-line, too

BIG Blasts on the Cluster

- Q. How do I blast 200,000 454 reads against nr?
- A. “Fake” parallel BLAST
  - Divide input sequences into 10 separate files
  - BLAST each smaller input file on a separate core
  - Running on 10 cores will be almost exactly 10x as fast!
- Why “fake” parallel?
  - Cores don’t need to talk to each other
  - You could just submit 10 jobs individually
  - Not to be confused with “really” parallel mpiBLAST et al.
- But we don’t want to submit 100 jobs by hand…

FAS INFORMATION TECHNOLOGY

Job Arrays I

- Job Arrays let you submit N jobs with one bsub
- `bsub -J ”bla[1-10]"` submits 10 jobs
  - Job array gets one numeric Job ID
  - `bjobs 1234` (or `bjobs bla`) lists all sub-jobs in job array 1234
  - `bjobs "1234[3]"` gets info on third sub-job
  - Quotes are needed for anything with [brackets], to avoid confusing the shell
- Similarly, you can bkill a whole array or one job

Job Arrays II

- In **bsub** options, `%I` stands for sub-job index
  - `#BSUB -o blast%I.out blastall …` yields blast1.out, blast2.out, etc. for sub-job 1, 2, etc.
  - Also can use `%I` with bsub's `-e`, etc.

- In **program** options, use `${LSB_JOBINDEX}`
  - In bla.bsub: `blastall … -i in_${LSB_JOBINDEX}.fasta`
  - Uses in_1.fasta, in_2.fast, etc. for jobs bla[1], bla[2], etc.
  - bsub on command line (not `bsub < a.bsub`): use `\$` instead of `$`

```
bsub -N -q short_serial -e bla%I.err
        blastall -i in_\${LSB_JOBINDEX}.fasta
```

  - (LSF sets environment variable LSB_JOBINDEX for each core)

BLAST Job Array Script

```
# Use serial queue since it’s only “fake” parallel
#BSUB -q short_serial
# Run four numbered jobs in job array
#BSUB -J easy_blast[1-4]
#BSUB -u akarger@cgr.harvard.edu
# %I will be replaced by 1, 2, etc. in -e and -o
#BSUB -e blast_array%I.err
#BSUB -o blast_array%I.m8
#BSUB -N
# ${LSB_JOBINDEX} will be replaced by 1, 2, etc.
blastall -p blastn -i Scer_10_${LSB_JOBINDEX}.fasta
  -m8 -d ../blastdb/fungi -e 1e-10 -b 25 -v 25
```

FAS
INFORMATION TECHNOLOGY

Fake Parallel BLAST - Finally!

- `cd ../blast_parallel`
- Split 40 FASTA sequences (Scer_40.fasta)
  ➔ 4 files: Scer_10_1.fasta, Scer_10_2.fasta, …
  ```
  Scriptome -t change_split_fasta Scer_40.fasta
  ```
  – Parameters are 10 and "Scer_10_NUMBER.fasta"
  – (Put the quotes around the filename to be safe)
  – (Or just cut and paste from the web)
- Blast each little FASTA against the database
  ```
  bsub < blast_array.bsub
  ```
- Concatenate resulting output files
  ```
  cat blast_array*.m8 > blast_40_seqs.m8
  ```

MrBayes

- `cd ../mrbayes_serial`
- MrBayes performs phylogenetic analysis
  - Input is a .nex Nexus file
- Loading the module
  - `module load hpc/mrbayes-3.1.2-patched`
- Running mb from command line
  - `mb blah.nex`
- bsub from the command line:
  - `bsub -q short_serial -J my_mb -o blah.out mb blah.nex`

Serial MrBayes Script

```
# Use a serial queue
#BSUB -q short_serial
#BSUB -o mrbayes_serial.out
#BSUB -e mrbayes_serial.err
# Send email even though I'm using -o
#BSUB -N
#BSUB -u example@example.com
#BSUB -J mrbayes_job
mb ND4_BAYESinv.nex
```

What does parallel mean, anyway?

- Parallel programs use more than one core
  - The program splits up the job, sends a piece to each core, and collects the results
  - Cores can be on one or more nodes
- Running parallel programs on Odyssey
  - Load different module (mvapich or openmpi in module name)
  - Use `-n` option to bsub to say how many cores you're using
  - Use `-a` option to say what kind of parallel (mvapich or openmpi)
  - Use `mpirun.lsf` in the bsub script before the command name
  - Use a program specially written to be parallel (may or may not have the same name)

Parallel MrBayes

- `cd ../mrbayes_parallel`
- MrBayes has an MPI parallel version
  - Cores talk to each other using Message-Passing Interface
  - 4 cores may be 2-3x as fast (depending) as a single core
  - Often have diminishing returns as #nodes grows
  - "Real" parallel compared to BLAST's "fake" parallel
  - Use #core = #chains
- Requires a different module
  - hpc/mrbayes-3.1.2-patched_openmpi-1.3.2_intel-11.0.083
  - Runs an mb executable that's in a different directory
  - So don't load both mrbayes modules simultaneously

Parallel MrBayes Script

```
# The -a is the important one! Run a parallel openmpi job.
#BSUB -a openmpi
# Use a parallel queue this time
#BSUB -q short_parallel
# Run on two cores
#BSUB -n 2
#BSUB -o mrbayes_parallel.out
#BSUB -e mrbayes_parallel.err
#BSUB -u example@example.com
mpirun.lsf mb ND4_BAYESinv.nex
```

FAS INFORMATION TECHNOLOGY

Other Bio Programs on Odyssey

- Phylogenetics
  - BayesPhylogenies, BEAST, BEST, Garli, im, Lamarc, PAML, PAUP, PHYLIP, PhyML, poy, RaxML
- Sequence analysis
  - blat, clustalw, EMBOSS, RepeatMasker, t-coffee
- Next-generation sequencing
  - bowtie/tophat/cufflinks, maq, velvet
- Molecular dynamics
  - GROMACS, CHARMM
- Math and programming
  - Matlab, Mathematica, Perl (BioPerl), Python, R (BioConductor)

More Cluster Resources

- Biological databases
  - /n/bluearc/mol/seq/* (may change soon to /n/bioseq/…)
  - ls -l before using. Some data is old, some updated
- More info: http://rc.fas.harvard.edu
- Ask rchelp@fas.harvard.edu:
  - What program(s) to use
  - To install programs not in `` `module avail` ``
  - How to use programs effectively
  - How to interpret results (command-line vs. web blast)
  - Before cutting and pasting 1000 cells in Excel
  - Before using 1000 cores for 6 weeks to write 100 terabytes