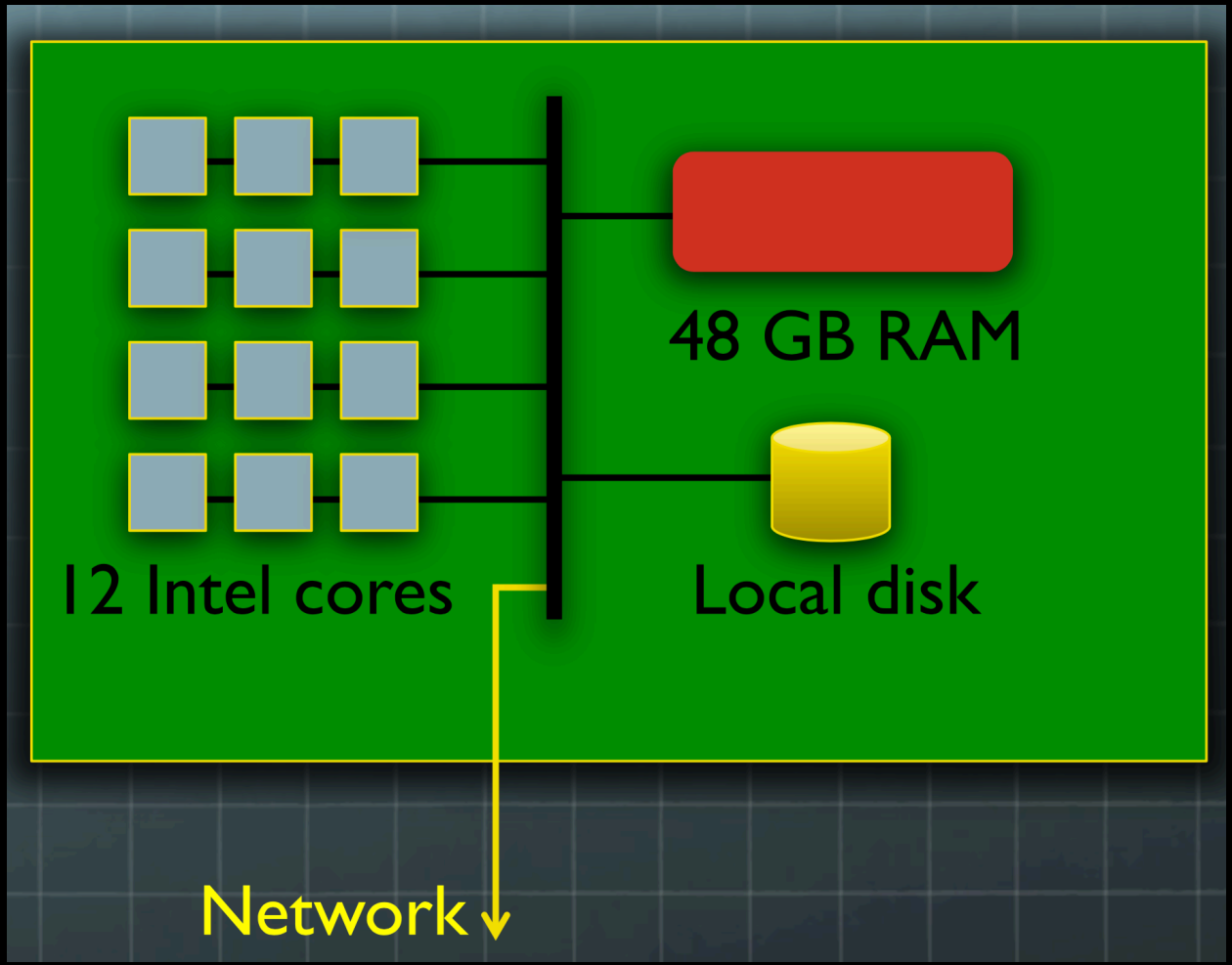
MODULE 03 – HIGH PERFORMANCE COMPUTING

# HIGH-PERFORMANCE COMPUTING ON FLUX

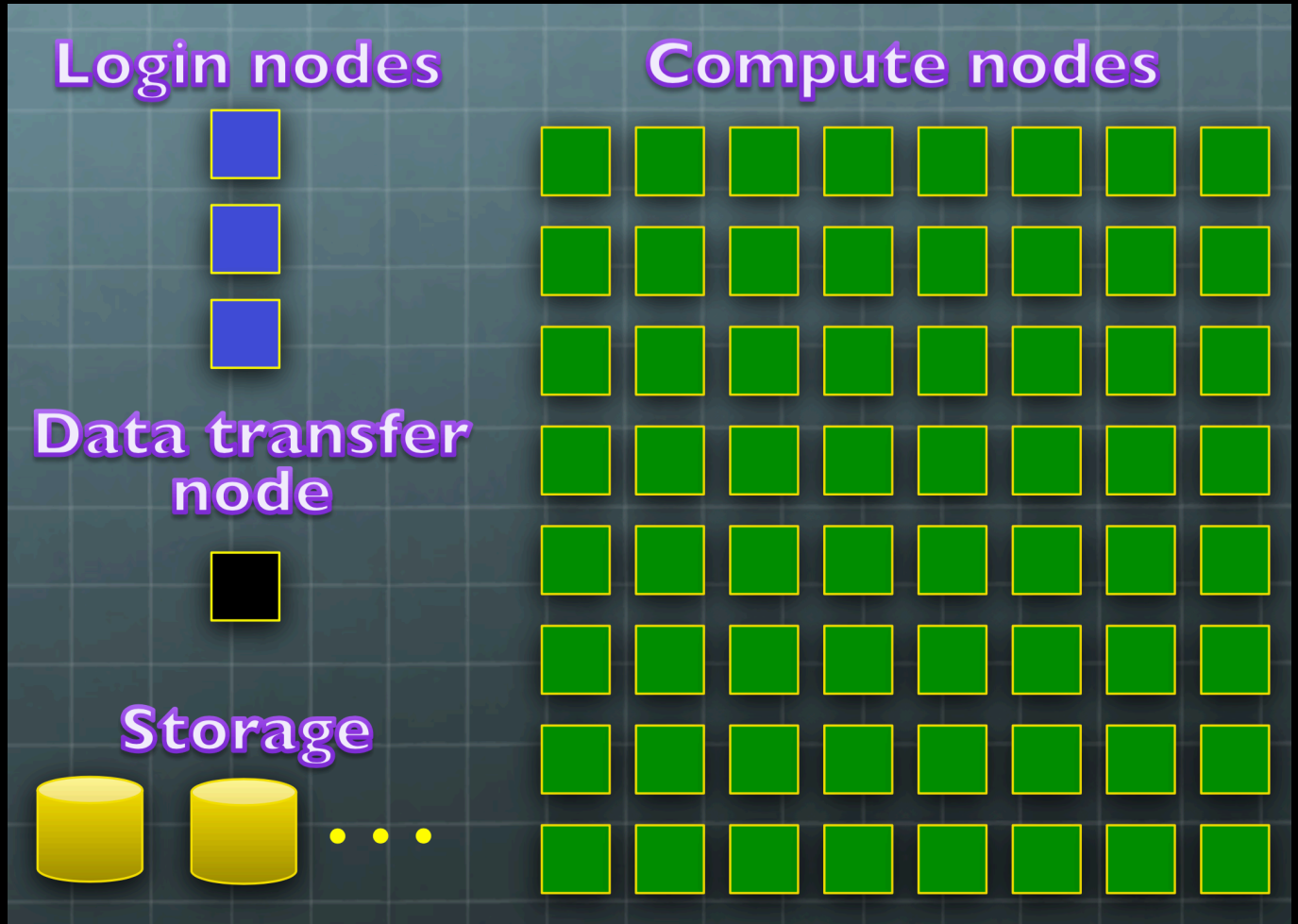
HYUN MIN KANG @ UNIVERSITY OF MICHIGAN

# A flux node




Ken Weiss

# The flux cluster




Ken Weiss

# To login the FLUX cluster…

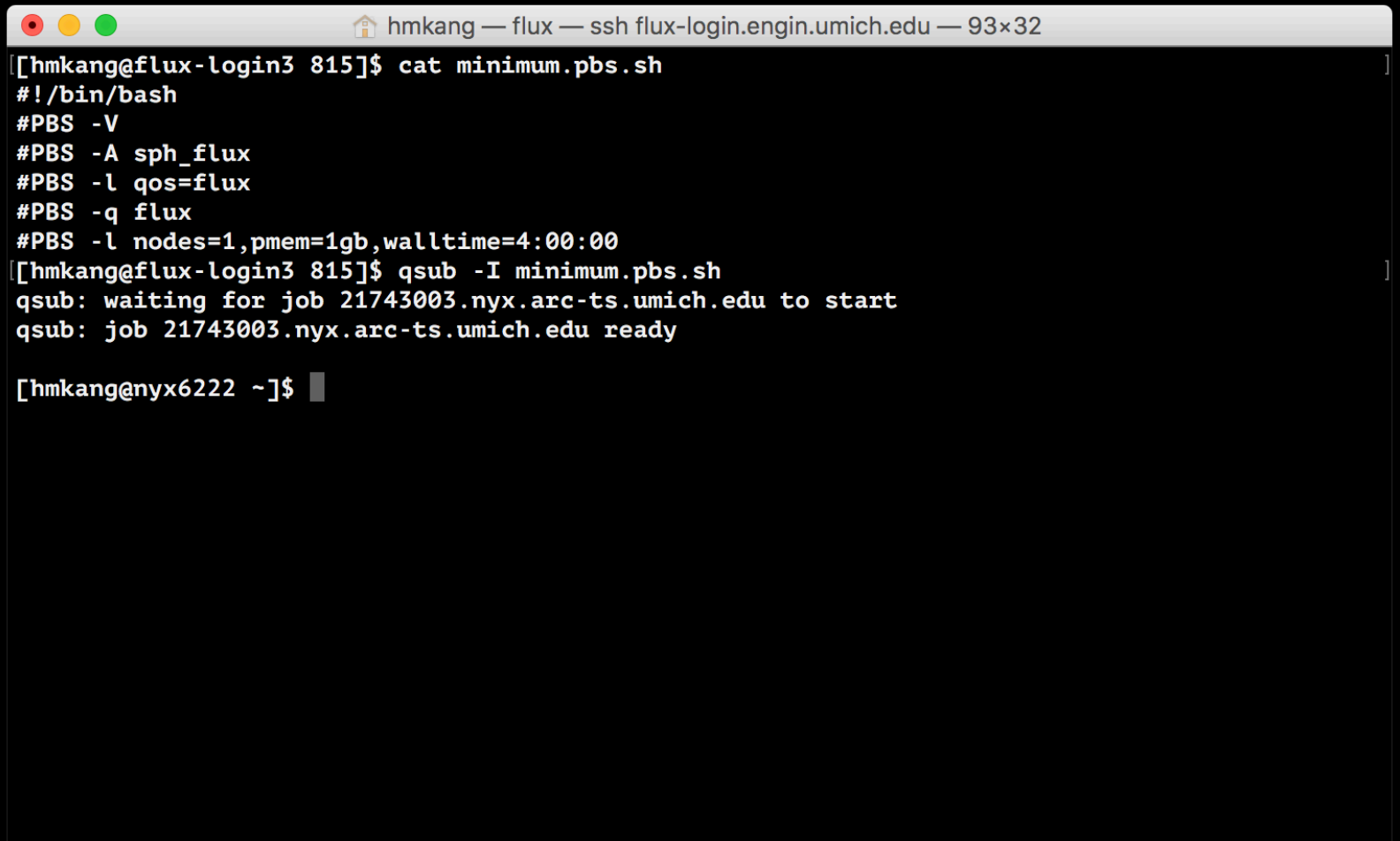
- **You need to be on-campus network**
  - If you're off-campus, use VPN
    http://its.umich.edu/enterprise/wifi-networks/vpn
- **You need to enroll to Duo two-factor authorization**
  http://its.umich.edu/accounts-access/uniqnames-passwords/two-factor-authentication
- **Follow these steps:**
  1. Login to `[uniqname]@flux-login.engin.umich.edu` via SSH
  2. Type your Kerberos password
  3. Complete login through duo two-factor authorization

# Writing PBS batch script

- **Start from the guide**
  http://arc-ts.umich.edu/flux-user-guide/
  http://www.youtube.com/watch?v=SW8Lu1-JaSM
  http://arc-ts.umich.edu/software/torque/pbs-template/

- **At the minimum you will need**
  **# PBS -V** (to inherit the environment variables)
  **# PBS -A bios815_flux** (account info)
  **# PBS -l qos=flux** (quality of service, use flux for standard jobs)
  **# PBS -q flux** (queue of the jobs, use flux for standard jobs)
  **# PBS -l nodes=1,pmem=4gb,walltime=4:00:00**
  (specify the resources you need)

# Running interactive jobs



# Scratch file system

- **`/home` directory in FLUX has limited amount of space available.**
  - Store important scripts, configurations that need to be backed up

- **Any large data files must be stored under `/scratch/[pi_uniqname]_flux/[your_uniqname]/`**
  - These files will not be backed up, and transfer the final outcome to a safe storage

```
[hmkang@flux-login3 815]$ df -h /home
Filesystem                                Size  Used Avail Use% Mounted on
nyxhome.value.storage.umich.edu:/nyxhome   23T   22T  1.3T  95% /home
[hmkang@flux-login3 815]$ df -h /scratch
Filesystem                                Size  Used Avail Use% Mounted on
10.255.7.0@o2ib:10.255.7.1@o2ib:/scratch  1.5P  1.3P  201T  87% /scratch
```

# Example 1 : Gaussian Mixture LRT

- **Consider the following hypothesis testing procedure**

$$H_0 : x_i \sim N(\mu, \sigma^2)$$

$$H_1 : x_i \sim \phi N(\mu_1, \sigma_1^2) + (1 - \phi) N(\mu_2, \sigma_2^2)$$

where $x_1, x_2, \ldots, x_n$ are i.i.d.
In other words, we are testing whether $\pi = 0$ or not.

- **Consider a likelihood-ratio test between these two hypotheses.**
- **We want to calculate the power of the LRT at size $\alpha$ at a particular configuration of** $\theta = (\phi, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2)$

# Constructing a LRT

$$L_0 = L(\hat{\theta}_0|\mathbf{x}) = \prod_{i=1}^{n}\left[\frac{1}{\sqrt{2\pi\hat{\sigma}^2}}\exp\left(-\frac{(x_i-\hat{\mu})^2}{2\hat{\sigma}^2}\right)\right]$$

$$\hat{\mu} = \overline{x}, \qquad \hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^2$$

$$L_1 = L(\hat{\theta}|\mathbf{x}) = \max_{(\phi,\mu_1,\mu_2,\sigma_1^2,\sigma_2^2)}\prod_{i=1}^{n}\left[\frac{\phi}{\sqrt{2\pi\sigma_1^2}}\exp\left(-\frac{(x_i-\mu_1)^2}{2\sigma_1^2}\right)\right.$$
$$\left. + \frac{1-\phi}{\sqrt{2\pi\sigma_2^2}}\exp\left(-\frac{(x_i-\mu_2)^2}{2\sigma_2^2}\right)\right]$$

$$T = 2[\log(L_1) - \log(L_0)] \sim ?$$

# Three key questions

- **How should we compute T and $L_1$?**
  - Calculating $L_1$ is not straightforward analytically

- **How can we construct a size α test?**
  - The asymptotic distribution of T is not straightforward to obtain

- **How should we compute the power?**
  - The closed form of the power function is difficult to obtain

# Answers are “numerical methods”

- **How should we compute T and $L_1$?**
  - Use E-M algorithm!

- **How can we construct a size α test?**
  - Obtain the empirical null distribution via simulation

- **How should we compute the power?**
  - Use simulation to compute the power

# Implementing power simulation in R

```
em2LRT <- function(x) {
    w <- rbinom(n.sample, 1, 0.5)  ## initial assignment
    means <- sample(x,2)
    sds <- rep(sd(x),2)
    for(i in 1:200) {
        llks <- cbind( dnorm(x, means[1], sds[1]) * phi,
                       dnorm(x, means[2], sds[2]) * (1-phi) ) + 1e-100
        w <- llks[,1] / rowSums(llks)
        phi <- mean(w)
        means <- c( mean(w * x)/phi, mean( (1-w)*x ) / (1-phi) )
        sds <- sqrt( c ( mean( w * ( x - means[1] )^2 ) / phi,
                         mean( (1-w)* ( x - means[2] )^2 ) / (1-phi)) )
    }
    llk.alt <- sum(log(rowSums(cbind( dnorm(x, means[1], sds[1])*phi,
                                      dnorm(x, means[2], sds[2])*(1-phi) ) ) + 1e-100))
    llk.null <- sum(log(dnorm(x, mean(x), sqrt(mean(x^2) - mean(x)^2))+1e-100))
    llk.diff <- llk.alt - llk.null
    if ( llk.diff < 0 ) { llk.diff <- 0 }
    return(list(llk1=llk.alt, llk0=llk.null, stat=2*llk.diff,
                log10p = pchisq(2*llk.diff, df=1, lower.tail=FALSE, log.p=TRUE)/log(10)))
}
```

# Handling arguments and running simulations

```
args <- commandArgs(TRUE)
out <- args[1]
phi <- as.double(args[2])
mu1 <- as.double(args[3])
mu2 <- as.double(args[4])
sd1 <- as.double(args[5])
sd2 <- as.double(args[6])
n.sample <- as.integer(args[7])
n.simul <- as.integer(args[8])
rst <- matrix(NA, n.simul, 4)
for(i in 1:n.simul) {  ## perform simulation and record statistics
    w <- rbinom(n.sample,1,phi)
    x <- rnorm(n.sample, mu1, sd1) * w + rnorm(n.sample, mu2, sd2) * (1-w)
    r <- em2LRT(x)
    rst[i,] <- c(r$llk1,r$llk0,r$stat,r$log10p)
}
colnames(rst) <- c("LLK1","LLK0","LRT","LOG10P");
write.table(rst,out,row.names=FALSE,col.names=TRUE,quote=FALSE,sep="\t");
```

# Running the simulation

```
$ /usr/bin/time -v Rscript gmmpower.r tmp.out 0.5 0 0 1 1 1000 1000
Command being timed: "Rscript gmmpower.r tmp.out 0.5 0 0 1 1 1000 1000"
User time (seconds): 54.31
System time (seconds): 0.04
Percent of CPU this job got: 99%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:54.40
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 25840
...
```

# Power calculation via simulation

- To calculate the power, we need to estimate

$$\Pr(T > t_\alpha | \theta)$$

where

$$\Pr(T > t_\alpha | \theta_0) = \alpha$$

- Suppose that we have n samples from the null distribution (X) and m samples from the alternative distribution (Y). We estimate the power using

$$\hat{\beta} = \frac{\sum_{i=1}^{n} I(Y_i > X_{(n-n\alpha+1)})}{m}$$

# Example power calculation

- **For obtaining the null distribution..**
  - n = 1,000,000

- **For each alternative hypothesis...**
  - m = 10,000

- **How many E-M iterations are required?**

- **How long would the job take if it runs with a single core?**

# Running one batch of 10,000 simulations

```
#!/bin/bash
#PBS -V
#PBS -A hmkang_flux
#PBS -l qos=flux
#PBS -q flux
#PBS -l nodes=1,pmem=1gb,walltime=4:00:00
export WD=/scratch/hmkang_flux/hmkang/class/815
Rscript ${WD}/gmmpower.r ${WD}/out1.txt 0.5 0 0 1 1
1000 10000
```

Use **`qsub foo.pbs.sh`** to submit the job to the cluster

# Ways to run 1,000,000 simulations

*(in 10 minutes)*

1. **Create 100 different PBS scripts,**
   **with different output file names,**
   **and submit each PBS script using qsub**

2. **Create one PBS script that can take an argument, and run the script 100 times with different arguments**

3. **Create a job array script and run the script once, specifying the number of simulations to run**

# Creating a job array script

```
#!/bin/bash
#PBS -V
#PBS -A hmkang_flux
#PBS -l qos=flux
#PBS -q flux
#PBS -t 1-100
#PBS -l nodes=1,pmem=1gb,walltime=4:00:00
export WD=/scratch/hmkang_flux/hmkang/class/815
Rscript ${WD}/gmmpower.r
${WD}/array.${PBS_ARRAYID}.out 0.5 0 0 1 1 1000
10000
```

See http://arc-ts.umich.edu/software/torque/job-arrays/ for more details

# Submitting the array job

- Simply run

```
$ qsub gmmpower.array.sh
```

- Check the status of the job using

```
$ qstat | grep [your_uniqname]
```
, or
```
$ qstat -f [job_id]
```

# Passing arguments by environment variables

```
#!/bin/bash
#PBS -V
#PBS -A hmkang_flux
#PBS -l qos=flux
#PBS -q flux
#PBS -l nodes=1,pmem=1gb,walltime=4:00:00
export
WD=/scratch/hmkang_flux/[your_uniqname]/class/815
Rscript ${WD}/gmmpower.r ${WD}/${out} 0.5 0 ${mu} 1 1
1000 10000
```

# Submitting 100 jobs with different arguments

```
$ seq -w 1 100 | xargs -I {} qsub -v
out='null.{}.out',mu='0' gmmpower.pbs.sh
```

- What do you think it will happen?

- For running jobs under the alternative hypothesis, what do we need to do?

# Running simulations under the alternative hypotheses

```
$ seq -w 0 0.1 2 | xargs -I {} qsub -v
out='alt.{}.out',mu='{}' gmmpower.pbs.sh
```

- What do you think it will happen?

- Can you do similar tasks using job array?

# Merging the null distribution

- Merge multiple files can be done with simple UNIX command in many cases

```
$ cat null.*.out | grep –v ^#LLK1 >
merged.null.out
```

- Automating the following analysis is also possible, but it's rather trivial tasks.

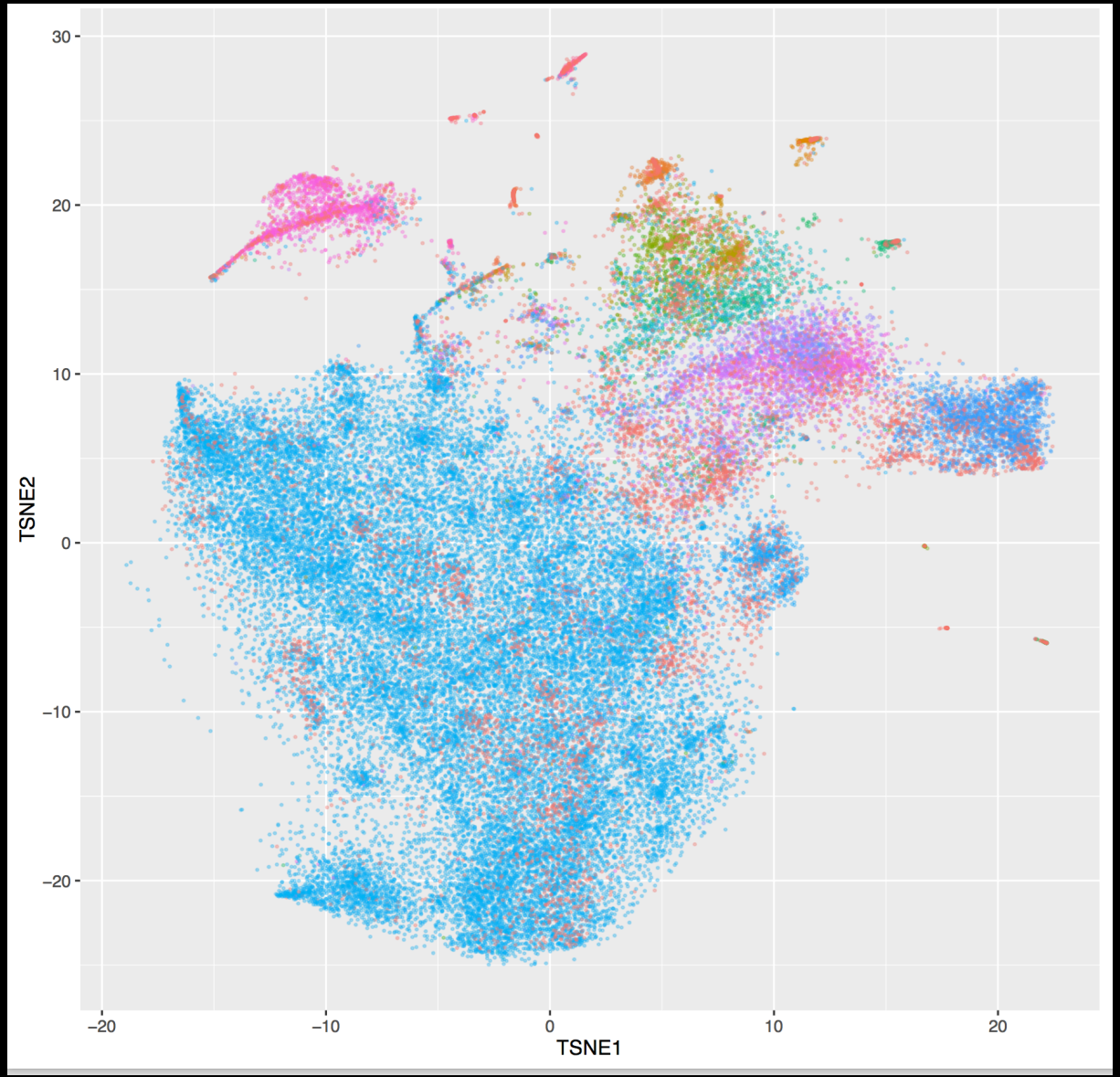
# Handling Big Data

`/scratch/hmkang_flux/hmkang/class/815/GSM1626793_1626799_P14Retina_merged.digital.expression.txt.gz`

`/scratch/hmkang_flux/hmkang/class/815/GSM1626793_1626799_P14Retina_merged.digital.expression.tsne.dat`

These files contains number of RNA transcript in mouse retina measured for each of 49,300 cell and 24,760 transcripts (Macosko et al. 2015)

File size is only 52M (sparse), but contains matrix with 1.2 billion elements


30
20
10
0
−10
−20
TSNE2
−20
−10
0
10
20
TSNE1

# What we want is

- Calculate the p-value for each gene in terms of association with TSNE1 and TSNE2 coordinates, accounting for the difference of total read counts for each cell.

- ...Using either negative binomial, or Poisson regression.

# Reviewing what we have learned..

- Numerical Methods
- Algorithms and data structures
- High-performance computing