Variant Calling with GATK on Hoffman2

July 21 2016

Sorel Fitz-Gibbon

sorel@ucla.edu

Day 3

- Parallelization on hoffman2 using Queue
- Queue – hands-on

1. Get updated day3.pdf

/u/nobackup/galaxy/collaboratory/sorel/gatk_workshop/sorel/slides/day3.pdf

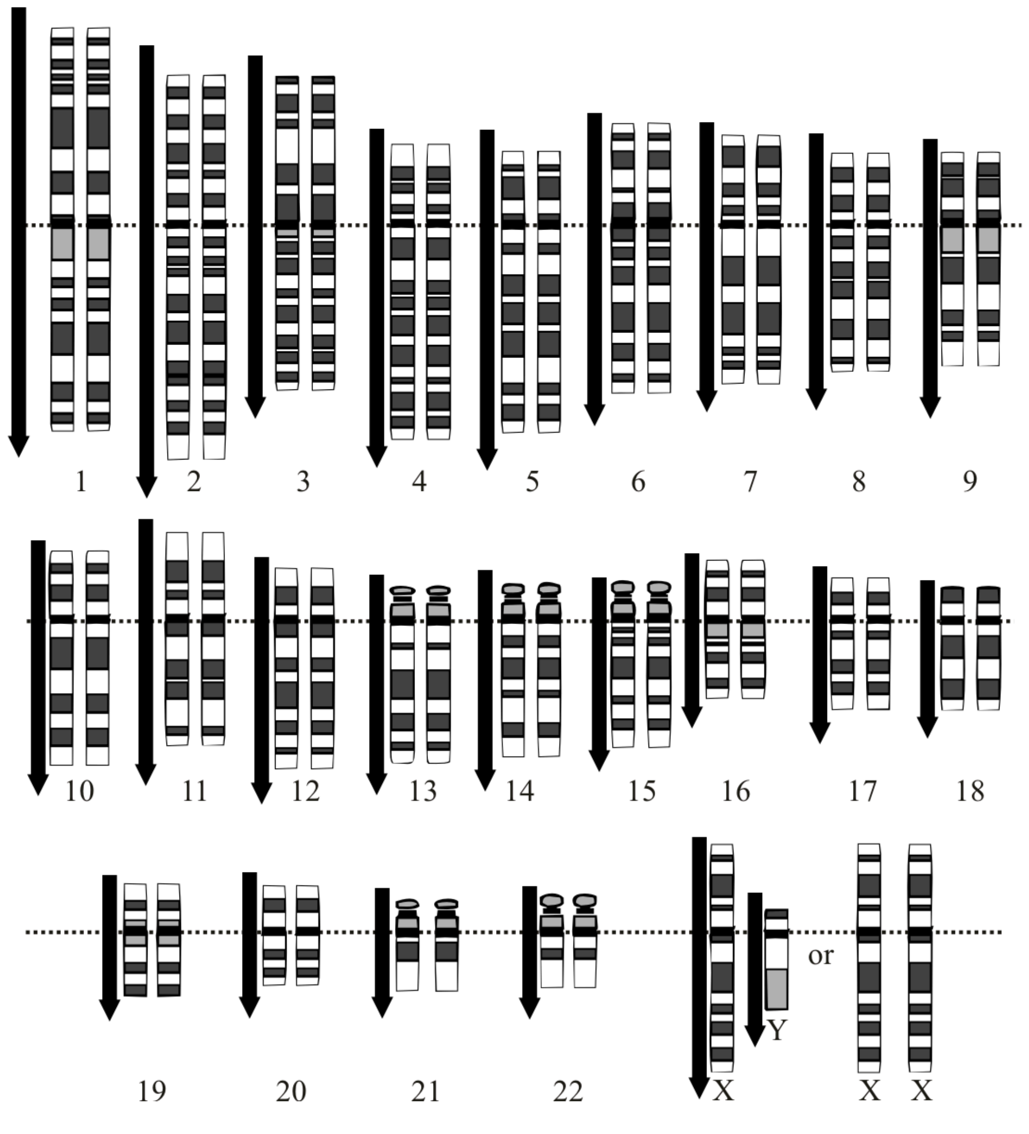
Next 8 Slides are from collaboratory fellow Michael Weinstein (and also included Broad Institute material)

Parallelism with Queue

Analysis in Series

- Computing resources
  - Requires only 1 CPU
  - Minimal overhead
- Time requirement
  - Every step done in series
  - Generally the longest wait time for a job

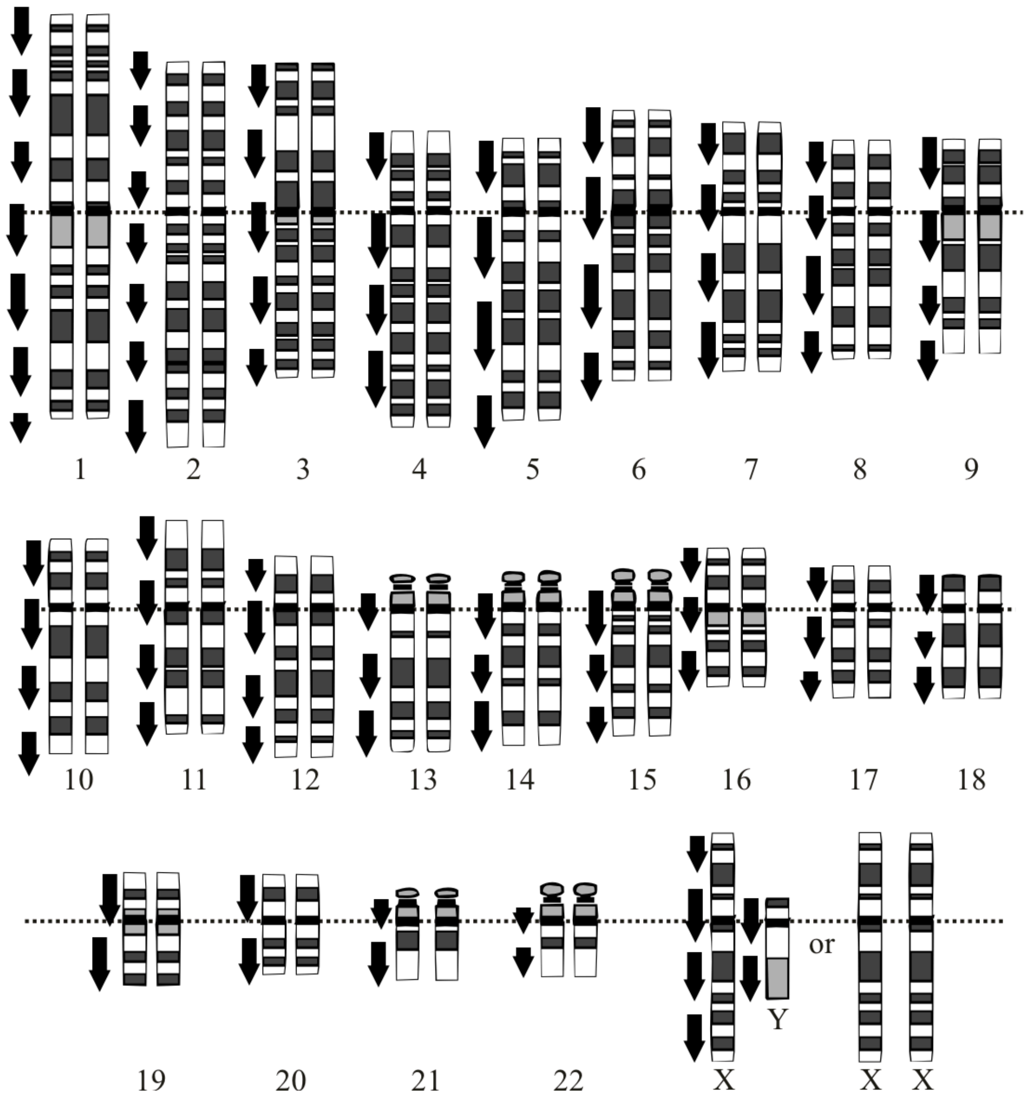
Parallelize by Chromosome



Parallelize by Chromosome

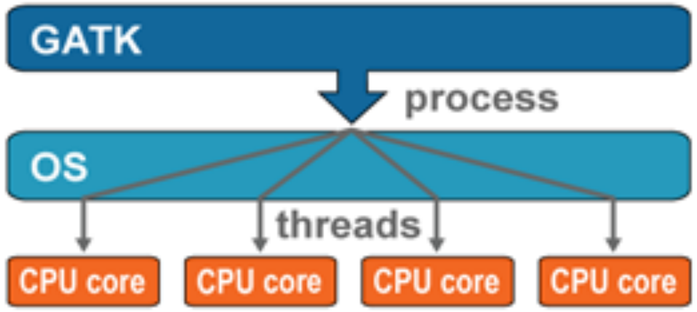
- Computing resources
  - Requires 1 CPU per chromosome (ideally)
  - More overhead
    - Set up multiple jobs
    - Combine results from multiple jobs
- Time requirement
  - Time to run the longest chromosome
  - Setup of jobs
  - Gathering of data

Maximum Parallelization



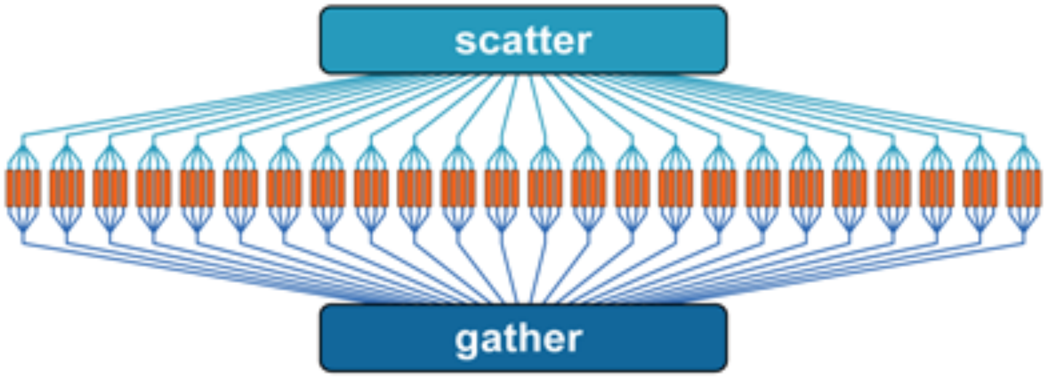
How Computers Parallelize

Multi-thread



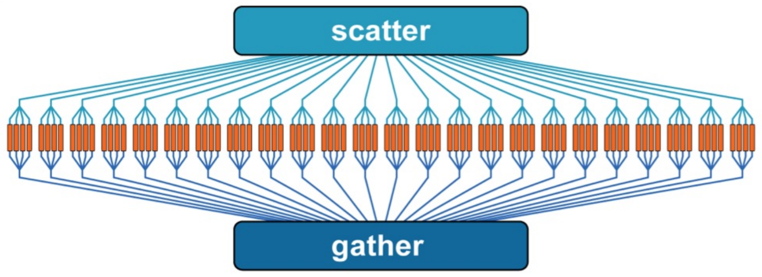
and/or

Multi-system



Using Queue for submitting jobs to the cluster, allowing large data sets to be processed quickly

- Queue is a companion package that makes it easy to
  - Execute GATK pipelines
  - Use scatter-gather parallelism
  - Run on server farm / cluster



Using Queue

- Multiple steps can be integrated in a script
  - Not limited to GATK
  - Picard can be added easily
  - Other apps may take more effort
- Knowledge of Scala will probably become important for advanced usage
- Basic rule: Every step has at least one input and one critical output
  - Queue “draws” a graph of inputs and outputs to decide on step order and parallelism

- Get an interactive shell

```
qrsh –l i,time=2:00:00,mem=4g
```

- Move to your gatk_workshop directory.

```
cd $SCRATCH/gatk_workshop
```

We'll start by using Queue to submit a script for running CallableLoci (although usually you wouldn't bother using Queue for CallableLoci as it runs quickly anyway)

callableLoci.scatter.scala

```
import org.broadinstitute.gatk.queue.QScript
import org.broadinstitute.gatk.queue.extensions.gatk._

class callable_loci extends QScript {
  def script() {
    val cl = new CallableLoci
    cl.reference_sequence = new File ("broad/ref/human_g1k_b37_20.fasta")
    cl.input_file :+= new File ("bams/trio-calling/NA12877_wgs_20.bam")
    cl.intervals :+= new File ("sandbox/two.intervals")
    cl.out = new File ("sandbox/NA12877_wgs_20.bam.QcallableLoci.bed")
    cl.summary = new File ("sandbox/NA12877_wgs_20.bam.QcallableLoci.summary")
    cl.scatterCount = 4
    cl.memoryLimit = 2     //this determines the -Xmx java request for each of the scattered jobs

    add(cl)
  }
}

//run with 'java -Xmx1g -Djava.io.tmpdir=tmp -jar /u/nobackup/galaxy/collaboratory/apps/gatk/Queue.jar -S sorel/
scripts/callableLoci.scatter.scala -startFromScratch -run'
```

Blue: variable names of choice

Red: these must match GATK tool and parameter names. Parameter names must be the long version, find them at the GATK documentation page for the tool

Compare the output from the command line run and the scala script run using the unix tool 'diff'

```
diff sandbox/NA12877_wgs_20.bam.callableLoci.summary sandbox/NA12877_wgs_20.bam.QcallableLoci.summary
```

You should get zero output, which means there are no differences between the files

- But you do get a difference with the current versions (bug?)

Previously we ran HaplotypeCaller like this…

```
gatk –T HaplotypeCaller –R data/ref/human_g1k_b37_20.fasta \
–L sandbox/two.intervals –I data/sandbox/trioBams.list \
–bamout sandbox/trio.activeRegions.bam –o data/sandbox/trio.vcf
```

Now we’re going to run with scatter-gather on the cluster.

haplotypeCaller.scatter.scala

```
import org.broadinstitute.gatk.queue.QScript
import org.broadinstitute.gatk.queue.extensions.gatk._

class callVariants extends QScript {
  def script() {
    val hc = new HaplotypeCaller
    hc.reference_sequence = new File (”broad/ref/human_g1k_b37_20.fasta”)
    hc.intervals :+= new File (”sandbox/two.intervals")
    hc.input_file :+= new File (”sandbox/trioBams.list")
    hc.bamOutput = new File(“sandbox/trio.activeRegions.bam")
    hc.out = new File (“sandbox/trio.scala.vcf")
    hc.scatterCount = 20
    hc.memoryLimit = 2     //this determines the -Xmx java request for each of the scattered jobs
    add(hc)
  }
}
//run with ‘java -Xmx1g -jar /u/nobackup/galaxy/collaboratory/apps/gatk/Queue.jar \
–S sorel/scripts/haplotypeCaller.scatter.scala –startFromScratch -qsub -jobResReq "h_data=4g,h_rt=1:00:00"
–run’
```

See the status of the jobs in the hoffman2 queue

```
qstat –u joebruin
```

```
[sorel@n2136]$ qstat -u sorel
job-ID  prior   name       user         state submit/start at     queue                          slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
 651078 0.00195 QRLOGIN    sorel        r     08/27/2015 08:02:49 cnsi_msa.q@n2142                   1
 651093 0.00195 QRLOGIN    sorel        r     08/27/2015 08:07:21 cnsi_msa.q@n2136                   1
 651159 0.00000 HaplotypeC sorel        r     08/27/2015 09:00:13 msa-smp.q@n2136                    1
 651161 0.00000 HaplotypeC sorel        r     08/27/2015 09:00:13 cnsi_msa.q@n2136                   1
 651162 0.00000 HaplotypeC sorel        r     08/27/2015 09:00:13 msa-smp.q@n2142                    1
 651163 0.00000 HaplotypeC sorel        r     08/27/2015 09:00:13 msa-smp.q@n2138                    1
 651164 0.00000 HaplotypeC sorel        r     08/27/2015 09:01:41 msa-smp.q@n2135                    1
 651166 0.00000 HaplotypeC sorel        r     08/27/2015 09:01:42 msa-smp.q@n2144                    1
 651168 0.00000 HaplotypeC sorel        r     08/27/2015 09:01:42 msa-smp.q@n2136                    1
 651170 0.00000 HaplotypeC sorel        r     08/27/2015 09:01:42 msa-smp.q@n2138                    1
 651171 0.00000 HaplotypeC sorel        r     08/27/2015 09:02:54 msa-smp.q@n2142                    1
 651172 0.00000 HaplotypeC sorel        r     08/27/2015 09:02:54 msa-smp.q@n2136                    1
 651174 0.00000 HaplotypeC sorel        r     08/27/2015 09:02:54 msa-smp.q@n2135                    1
 651177 0.00000 HaplotypeC sorel        qw    08/27/2015 09:03:03                                    1
 651178 0.00000 HaplotypeC sorel        qw    08/27/2015 09:03:36                                    1
```

- Using Queue to scatter/gather jobs will speed up run times and reduce maximum memory needs
- To create a scala script for any GATK task, just follow the template in either callableLoci.scatter.scala or haplotypeCaller.scatter.scala
  - You must use the **long versions** of the parameter names. Find them in the GATK documentation for that tool
  - Note that when a parameter can be used more than once in a gatk command line, e.g. –known or --input_file, the script has a ":+=" rather than just "=".
  - For larger jobs, e.g. variant calling across the whole human genome, set the –scatter parameter higher. 100 or 200 are reasonable.

Running Long Jobs

When your jobs get too long to wait for, you'll want to qsub the original Queue.jar command.

– Put the Queue.jar command into a file with #!/bin/bash as the first line, then qsub that file.

E.g.

Qcmd.sh

```
#!/bin/bash
java –Xmx1g –jar /u/nobackup/galaxy/collaboratory/apps/gatk/Queue.jar \
–S sorel/scripts/haplotypeCaller.scatter.scala –qsub –jobResReq "h_data=4g,h_rt=1:00:00 –run
```

```
qsub –cwd –o ./ –e ./ –M joeBruin@ucla.edu –m a –V –l h_data=4G,h_rt=24:00:00 sorel/
scripts/Qcmd.sh
```

-cwd   execute the job from the current working directory
-o where to put standard output files
-e where to put standard error files
-M email address
-m when to email (a=aborted jobs, b=when jobs start, e=when jobs end)
-V use your usual environmental variable
-l resource request

Running Long Jobs

When your jobs get too long to wait for, you'll want to qsub the original Queue.jar command.

- Put the Queue.jar command into a file with #!/bin/bash as the first line, then qsub that file.

E.g.

You'll want to increase both of these for large jobs

Qcmd.sh

```
#!/bin/bash
java -Xmx1g -jar /u/nobackup/galaxy/collaboratory/apps/gatk/Queue.jar \
-S sorel/scripts/haplotypeCaller.scatter.scala -qsub -jobResReq "h_data=4g,h_rt=1:00:00 -run
```

```
qsub -cwd -o ./ -e ./ -M joeBruin@ucla.edu -m a -V -l h_data=4G,h_rt=24:00:00 sorel/
scripts/Qcmd.sh
```

-cwd execute the job from the current working directory
-o where to put standard output files
-e where to put standard error files
-M email address
-m when to email (a=aborted jobs, b=when jobs start, e=when jobs end)
-V use your usual environmental variable
-l resource request

- Did running Qcmd.sh work?
- Check qstat |grep joebruin
- Check Qcmd.sh.eXXXXXXX and Qcmd.sh.oXXXXXXX
  - This time they may not help
- If it didn’t work, what happened?
- Check your email and try to figure out what happened.

```
qsub -cwd -o ./ -e ./ -M joeBruin@ucla.edu -m a -V -l h_data=8G,h_rt=24:00:00 sorel/
scripts/Qcmd.sh
```

or

```
qsub -cwd -o ./ -e ./ -M joeBruin@ucla.edu -m a -V -l h_data=4G,h_rt=24:00:00 \
 -pe shared 2 sorel/scripts/Qcmd.sh
```

- Sample Qscripts below for reference (from Michael Weinstein)

Sample Queue Script

```
import org.broadinstitute.gatk.queue.QScript
import org.broadinstitute.gatk.queue.extensions.gatk._
class scatterGather extends QScript {
 def script() {
  val ir = new IndelRealigner
  ir.input_file = Seq(new File(“R01-264.143251.sorted.dedup.group.bam"))
  ir.knownAlleles = Seq(new File(“Mills_and_1000G_gold_standard.indels.b37.vcf"))
  ir.reference_sequence = new File(“hs37d5.fa")
  ir.targetIntervals = new File(“R01-264.143251.sorted.dedup.group.intervals.list")
  ir.out = new File(“R01-264.143251.sorted.dedup.group.realigned.bam")
  ir.scatterCount = 20
  ir.memoryLimit = 1
  add(ir)
 }
}
```

Sample Queue Script

```
import org.broadinstitute.gatk.queue.QScript
import org.broadinstitute.gatk.queue.extensions.gatk._
class scatterGather extends QScript {
 def script() {
  val br1 = new BaseRecalibrator
  br1.input_file = Seq(new File(“/R08-612A.143248.sorted.dedup.group.realigned.bam"))
  br1.reference_sequence = new File(“hs37d5.fa")
  br1.knownSites = Seq(new File(“Mills_and_1000G_gold_standard.indels.b37.vcf"), \
                                    new File(“dbsnp_138.b37.vcf"))
  br1.out = new File(“R08-612A.143248.sorted.dedup.group.realigned.recal.table")
  br1.scatterCount = 20
  br1.memoryLimit = 1
  add(br1)
 }
}
```

Sample Queue Script

```
import org.broadinstitute.gatk.queue.QScript
import org.broadinstitute.gatk.queue.extensions.gatk._
class scatterGather extends QScript {
 def script() {
  val br2 = new BaseRecalibrator
  br2.input_file = Seq(new File(“R08-612A.143248.sorted.dedup.group.realigned.bam"))
  br2.reference_sequence = new File(“hs37d5.fa")
  br2.knownSites = Seq(new File(“Mills_and_1000G_gold_standard.indels.b37.vcf"), \
                                      new File(“dbsnp_138.b37.vcf"))
  br2.out = new File(“R08-612A.143248.sorted.dedup.group.realigned.postrecal.table")
  br2.BQSR = new File(“R08-612A.143248.sorted.dedup.group.realigned.recal.table")
  br2.scatterCount = 20
  br2.memoryLimit = 1
  add(br2)
 }
}
```

Sample Queue Script

```
import org.broadinstitute.gatk.queue.QScript
import org.broadinstitute.gatk.queue.extensions.gatk._
class scatterGather extends QScript {
 def script() {
  val br3 = new PrintReads
  br3.input_file = Seq(new File(“R08-612A.143248.sorted.dedup.group.realigned.bam"))
  br3.reference_sequence = new File(“hs37d5.fa")
  br3.BQSR = new File(“R08-612A.143248.sorted.dedup.group.realigned.recal.table")
  br3.out = new File(“R08-612A.143248.sorted.dedup.group.realigned.recal.bam")
  br3.scatterCount = 20
  br3.memoryLimit = 1
  add(br3)
 }
}
```

Sample Queue Script

```
import org.broadinstitute.gatk.queue.QScript
import org.broadinstitute.gatk.queue.extensions.gatk._
import org.broadinstitute.gatk.queue.extensions.picard._
import org.broadinstitute.gatk.tools.walkers.haplotypecaller.ReferenceConfidenceMode
class scatterGather extends QScript {
 def script() {
  val hc = new HaplotypeCaller
  hc.input_file = Seq(new File(“R08-612A.143248.sorted.dedup.group.realigned.recal.bam"))
  hc.reference_sequence = new File(“hs37d5.fa")
  hc.bamOutput = new File(“R08-612A.143248.activeRegions.bam")
  hc.interval_padding = 50
  hc.emitRefConfidence = ReferenceConfidenceMode.GVCF
  hc.out = new File(“R08-612A.143248.g.vcf")
  hc.intervals = Seq(new File(“Broad.human.exome.b37.interval_list"))
  hc.scatterCount = 20
  hc.memoryLimit = 1
  add(hc)
 }
}
```

- The broad workshop slides show good examples of typical errors and false positives
- I’ve shown commands to manipulate the data on hoffman and locally and to view with IGV
- Check out vcftools and vcf-lib