MPG Workshop / February 17, 2011

GATK-Queue:

Command-line job manager and scripting framework for multi-stage genomic analysis

Khalid Shakir (kshakir@broadinstitute.org)
GENOME SEQUENCING AND ANALYSIS, BROAD INSTITUTE

GATK-Queue manages the multiple time consuming steps to produce genomic data

- Given list of BAMs, chip data, intervals, etc
- Call SNPs on BAMs using intervals
- Call Indels
- Filter the SNPs at Indels
- Annotate SNPs
- Annotate Indels
- Calculate QC metrics
- Generate a summary report
- Instead of taking two weeks to run, have the results output by Monday

Goal of GATK-Queue: Easy to define jobs and have them run on time to completion without user interaction

- ✔ As easy as writing a shell script
- ✔ Manages complexity of actually running jobs
- ✔ Responds to transient errors
- ✔ Resume from the point of failure
- Deliver results by a user defined ETA
- Portable scripts that collaborators can run on their own infrastructure

Snapshot of SNP calling script defining the programs, inputs and output files

```
snps = new UnifiedGenotyper with commonArgs
snps.bamFiles = project.bamFiles
snps.dbsnp = project.dbsnp
snps.out = project.name + "SnpCalls.vcf"
snps.memoryLimit = "6g"
add(snps)

indelCallFiles = []
for (bam : project.bamFiles) {
  bamIndels = new IndelGenotyperV2 with commonArgs
  bamIndels.bamFiles = [bam]
  bamIndels.out = bam.name + "IndelCalls.vcf"
  add(bamIndels)
  indelCallFiles += bamIndels.out
}

mergeIndels = new CombineVariants with commonArgs
mergeIndels.mergeFiles = indelCallFiles
mergeIndels.out = project.name + "IndelCalls.vcf"
add(mergeIndels)

filter = new VariantFiltration with commonArgs
filter.variantVCF = snps.out
filter.maskVCFs = [mergedIndels.out]
filter.maskName = "IndelMask"
filter.out = project.name + "FilteredCalls.vcf"
add(filter)
```

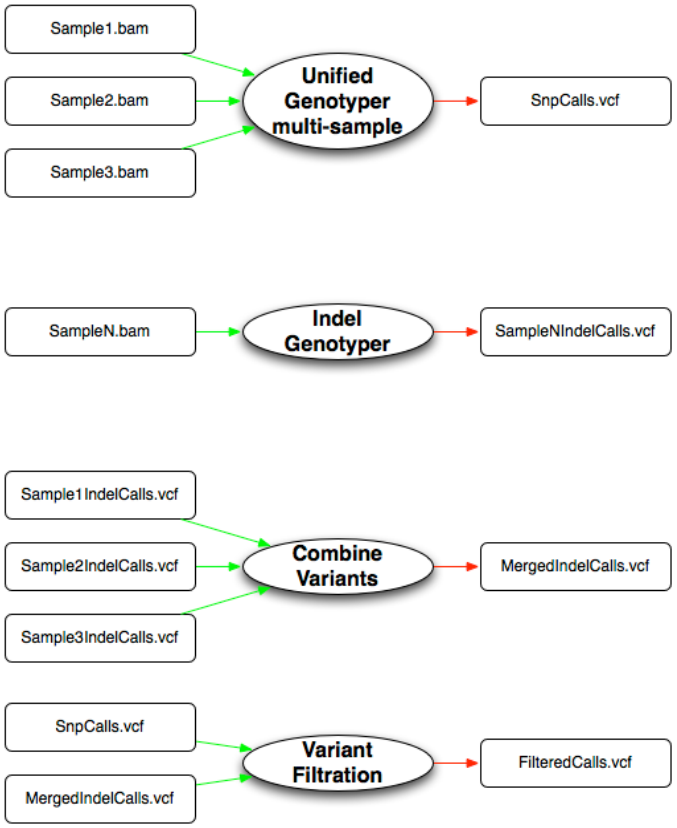
Create instance of GATK command line program

Set paths for inputs & outputs

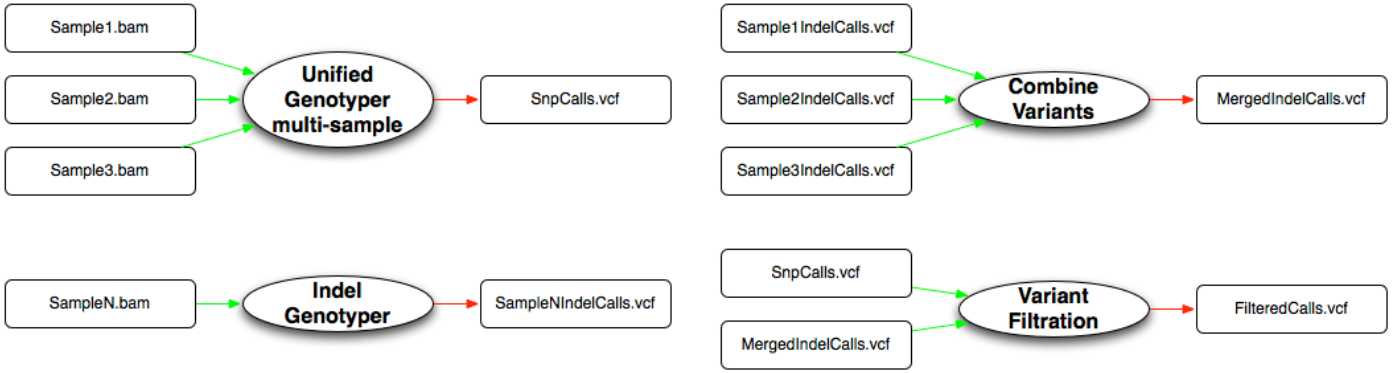
add() the program to Queue

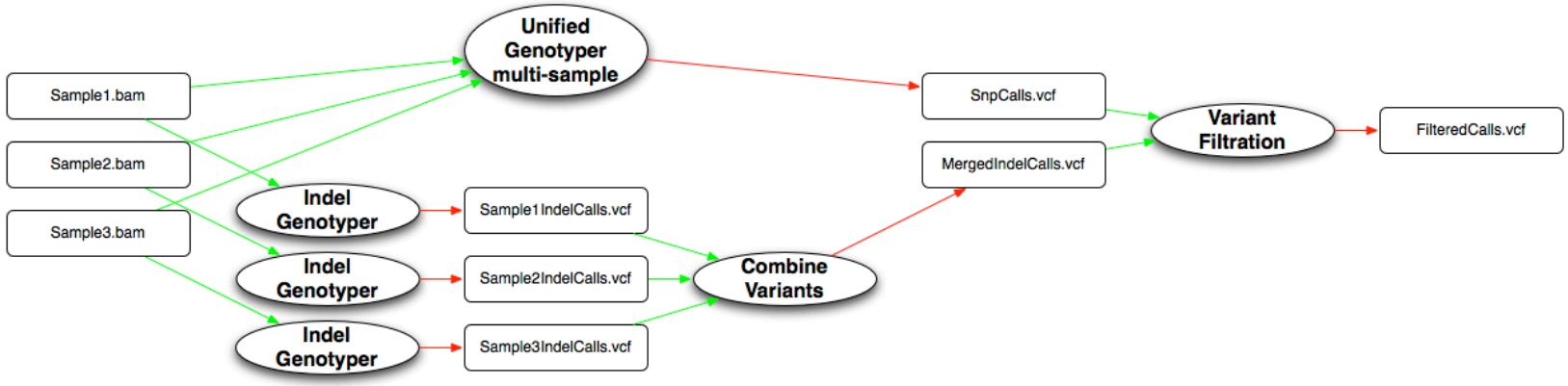
Snapshot of SNP calling script defining the programs, inputs and output files

```
snps = new UnifiedGenotyper with commonArgs
snps.bamFiles = project.bamFiles
snps.dbsnp = project.dbsnp
snps.out = project.name + "SnpCalls.vcf"
snps.memoryLimit = "6g"
add(snps)

indelCallFiles = []
for (bam : project.bamFiles) {
  bamIndels = new IndelGenotyperV2 with commonArgs
  bamIndels.bamFiles = [bam]
  bamIndels.out = bam.name + "IndelCalls.vcf"
  add(bamIndels)
  indelCallFiles += bamIndels.out
}

mergeIndels = new CombineVariants with commonArgs
mergeIndels.mergeFiles = indelCallFiles
mergeIndels.out = project.name + "IndelCalls.vcf"
add(mergeIndels)

filter = new VariantFiltration with commonArgs
filter.variantVCF = snps.out
filter.maskVCFs = [mergedIndels.out]
filter.maskName = "IndelMask"
filter.out = project.name + "FilteredCalls.vcf"
add(filter)
```

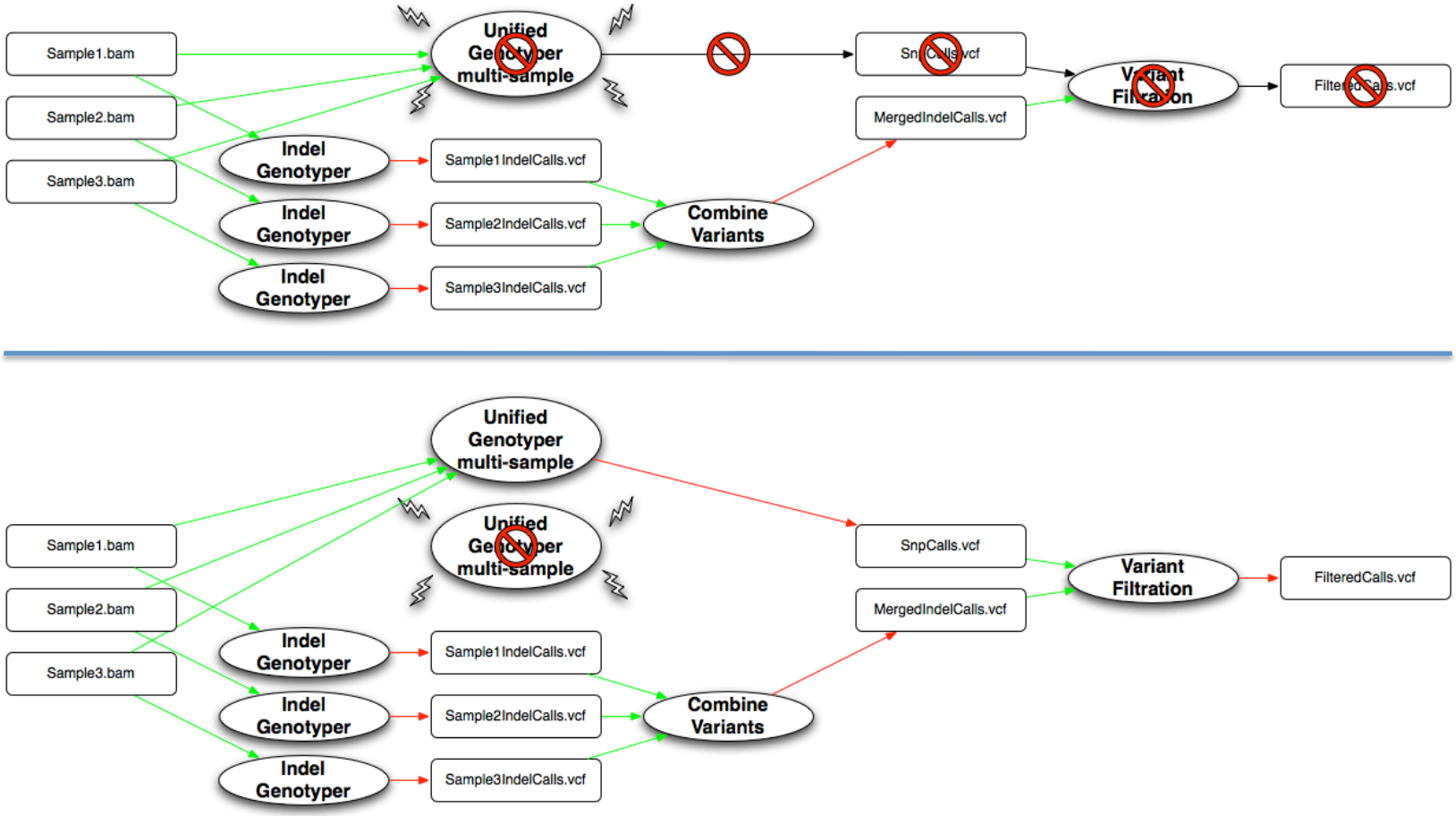


Queue implicitly knows job dependencies by mapping between job inputs and outputs

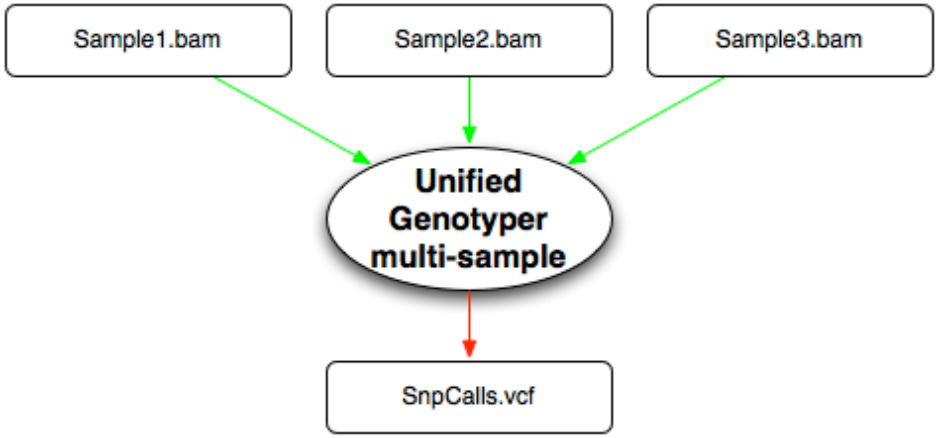




Queue dispatches graph and monitors for success or retries failed jobs before running dependents

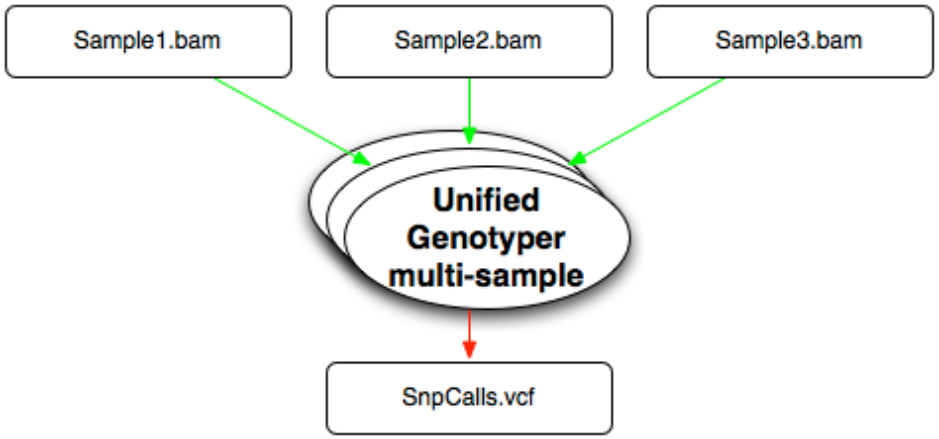


With Queue managing logistics it can produce results faster by using various techniques such as Scatter/Gather



Original multi-sample Unified Genotyper

ETA: 10 Days



Queue 20x parallel multi-sample Unified Genotyper

ETA: 6 Hours

Queue is ready for use today with a variety of features and a few things we'll add in the future

- Separation of logistics from semantics
- Natively understands GATK tools with IDE support for command completion
- Supports arbitrary command line programs (see appendix for samtools example)
- Restart from the point of failure
- Wait and retry for transient issues
  - LSF
  - automount
  - NFS file propagation
- Scripts can easily be modified and updated with the latest programs
- Check out and run portable Queue pipelines today
  - Kiran's pipeline
  - 1000G whole genome low pass pipeline
  - Lots of other examples
- In the future:
  - End-to-end pipelines with best practice for FASTQ-to-tearsheet
  - GridEngine, EC2, etc.

Getting Started Today

- GATK-Queue documentation for users and developers: http://www.broadinstitute.org/gsa/wiki/index.php/GATK-Queue

- Help / support for all GATK tools: http://getsatisfaction.com/gsa

Acknowledgments

Genome Sequencing and Analysis Group (MPG)
Mark DePristo
Matt Hanna
Kiran Garimella
Mauricio Carneiro
Chris Hartl
Corin Boyko
Eric Banks
Ryan Poplin
Guillermo del Angel

Broad postdocs, staff, and faculty
Menachem Fromer
Bob Handsaker
Sarah Calvo
Daniel Lieber
Jason Flannick
Ben Neale

Cancer genome analysis
Kristian Cibulskis
Doug Voet
Gordon Saksena
Aaron McKenna

MPG directorship
Stacey Gabriel
David Altshuler
Mark Daly

Research Computing
Matthew Trunnell
Eric Jones
Aaron Ball
Jean Chang
Ed Lauzier

Genome Sequencing Platform
Tim Fennell
Alec Wysoker

Appendix: samtools example

```
class SamtoolsIndexFunction extends CommandLineFunction {
  @Input(doc="BAM file to index")
  bamFile = ""

  @Output(doc="BAM file index")
  bamFileIndex = ""

  def commandLine = "samtools index %s %s"
    .format(bamFile, bamFileIndex)
}
```

Annotate input and output variables

Format command line using variables

Can be as simple as your program arguments or add more complexity if required. Automatically generated for GATK walkers.