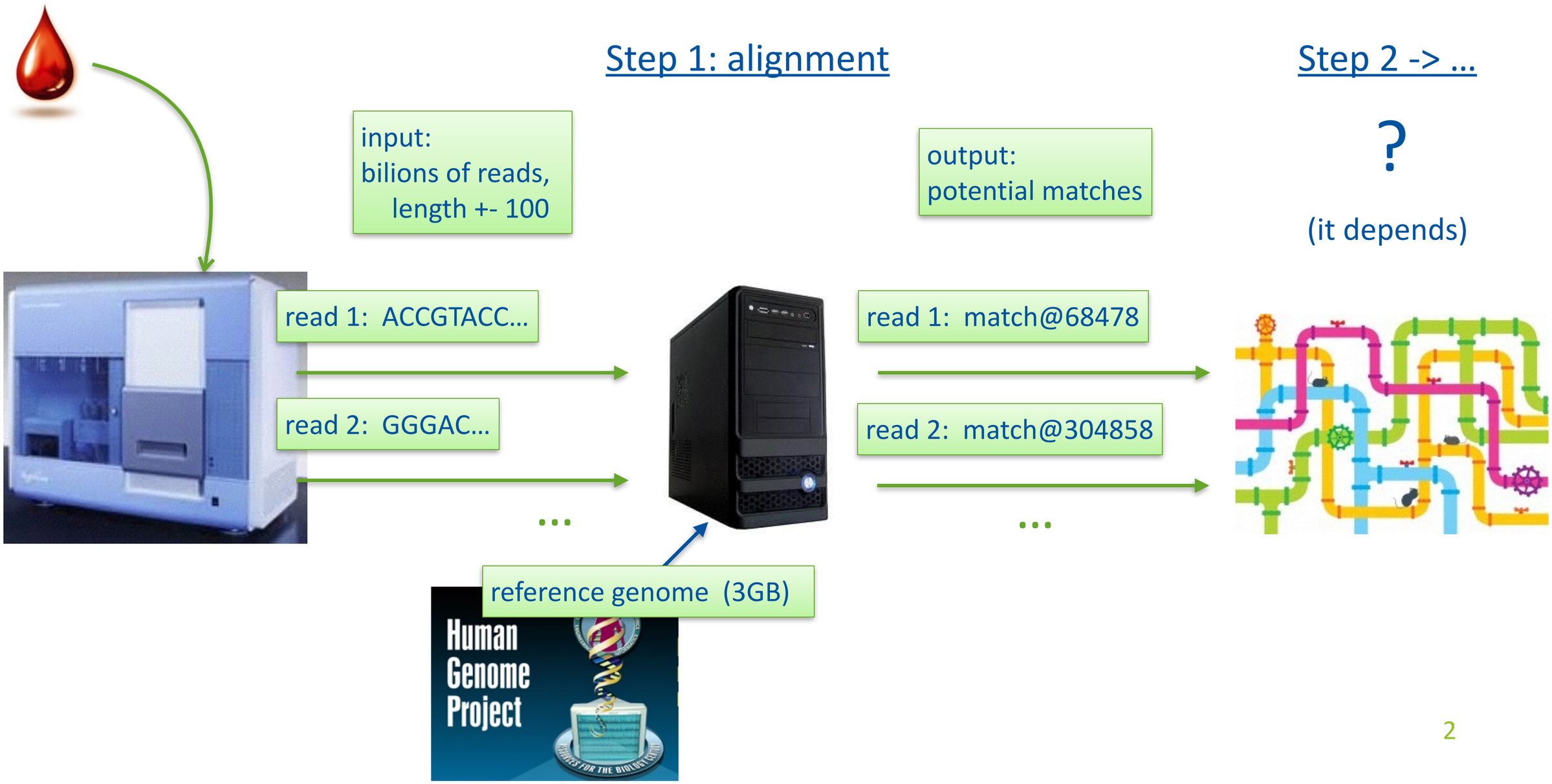
Parallel Gene Sequencing without the Pain

Yves Vandriessche

Software Languages.Lab
Vrije Universiteit Brussel

Sept. 18th, 2014

Case: Gene Sequencing

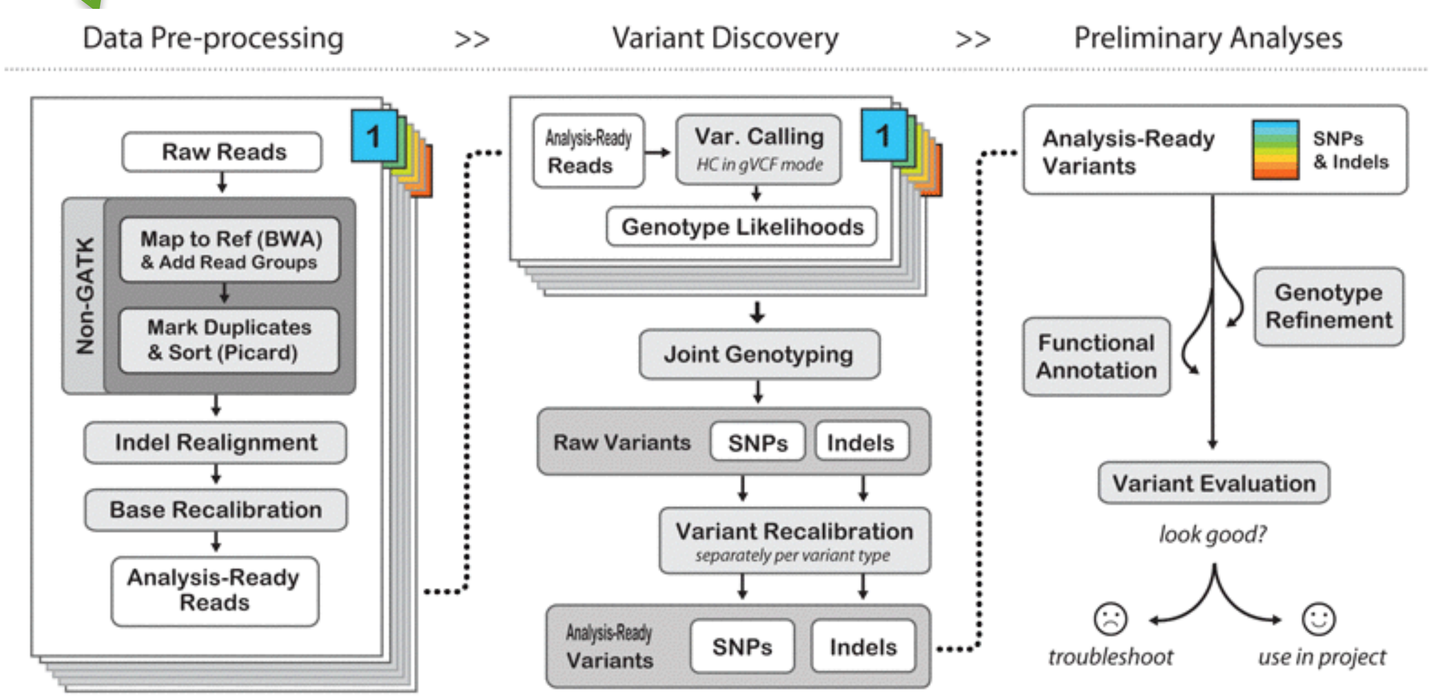


Case #1:
Whole Genome Sequencing Pipeline

application / experiment

- sequencing tools
- how they reason

what they write




```
log=$results/timings.log
gatkref=~/data/ucsc_hg19/ucsc.hg19.fasta
bwa=~/bwa/bin/bwa_pthreads
samtools=~/samtools/samtools-0.1.19
vcftools=NIL
picardtools=~/picard/dist/
gatk=~/gatk/GenomeAnalysisTK-2.7-2-g6bda569
dbsnp=/home/caherzee/bwa/example/bundle/vcf/dbsnp_137.hg19.vcf
mkdir -p $results
# convert sai to sam
echo "Converting SAI to SAM" >> $log \
{ time -f "Time: %E real,%U user,%S sys" \
$bwa/bwa samse $ref $target $reads > $results/$results.sam \
;} 2>> $log
# convert sam to bam
echo "Converting SAM to BAM" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
$samtools/samtools view -@ 24 -bS $results/$results.sam > $results/$results.bam \
;} 2>> $log
# removing unmapped reads, cause picard does not like them
echo "Removing unmapped reads from BAM" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
$samtools/samtools view -@ 24 -h -F 4 -b $results/$results.bam > $results/$results.only_m
;} 2>> $log
# sort for contigs
echo "Sorting BAM for contigs to gatk ref" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -jar $picardtools/ReorderSam.jar I= $results/$results.only_mapped.bam O= $results/
$results.only_mapped.reordered-contigs.bam REFERENCE= $gatkref \
;} 2>> $log
# sort bam file
echo "Sorting BAM for coordinate order" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
$samtools/samtools sort -@ 24 $results/$results.only_mapped.reordered-contigs.bam $result
$results.only_mapped.reordered-contigs.sorted \
;} 2>> $log
# remove duplicates
echo "Removing Duplicates from BAM" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -jar $picardtools/MarkDuplicates.jar \
     I=$results/$results.only_mapped.reordered-contigs.sorted.bam O=$results/$results.onl
contigs.sorted.deduplicated.bam M=$results/$results.picard-duplicates \
;} 2>> $log
# add read group information
echo "Adding Read Group information to BAM" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -jar $picardtools/AddOrReplaceReadGroups.jar VALIDATION_STRINGENCY=SILENT \
     I=$results/$results.only_mapped.reordered-contigs.sorted.deduplicated.bam O=$results
$results.only_mapped.reordered-contigs.sorted.deduplicated.read-group.bam \
     RGID=group1 RGLB= lib1 RGPL=illumina RGPU=unit1 RGSM=sample1 \
;} 2>> $log
# build bam index
echo "Building BAM index" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -jar $picardtools/BuildBamIndex.jar I=$results/$results.only_mapped.reordered-
contigs.sorted.deduplicated.read-group.bam VALIDATION_STRINGENCY=SILENT \
;} 2>> $log
# realign around indels
echo "Updating BAM with Realigning around indels (1/2)" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -Xmx2g -jar $gatk/GenomeAnalysisTK.jar \
   -T RealignerTargetCreator \
   -R $gatkref \
   -I $results/$results.only_mapped.reordered-contigs.sorted.deduplicated.read-group.bam
   -o $results/forIndelRealigner.intervals \
;} 2>> $log
echo "Updating BAM with Realigning around indels (2/2)" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -jar $gatk/GenomeAnalysisTK.jar -T IndelRealigner -R $gatkref -I $results/
$results.only_mapped.reordered-contigs.sorted.deduplicated.read-group.bam \
     -targetIntervals $results/forIndelRealigner.intervals -o $results/$results.only_mapp
contigs.sorted.deduplicated.read-group.realigned.bam \
;} 2>> $log
# build bam index
echo "Rebuilding BAM index for gatk" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -jar $picardtools/BuildBamIndex.jar I=$results/$results.only_mapped.reordered-
contigs.sorted.deduplicated.read-group.realigned.bam VALIDATION_STRINGENCY=SILENT \
;} 2>> $log
#Base recalibration
echo "Base recalibration on BAM for gatk (1/3)" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -jar $gatk/GenomeAnalysisTK.jar -T BaseRecalibrator -R $gatkref -I $results/
$results.only_mapped.reordered-contigs.sorted.deduplicated.read-group.realigned.bam \
     -knownSites $dbsnp -o $results/recal.grp \
;} 2>> $log
echo "Base recalibration on BAM for gatk (3/3)" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -jar $gatk/GenomeAnalysisTK.jar -T PrintReads -R $gatkref -I $results/$results.only_
contigs.sorted.deduplicated.read-group.realigned.bam \
     -BQSR $results/recal.grp -o $results/$results.only_mapped.reordered-contigs.sorted.d
group.realigned.recalibrated.bam \
;} 2>> $log
# Calling variants
echo "Calling variants" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -jar $gatk/GenomeAnalysisTK.jar -R $gatkref -I $results/$results.only_mapped.reorder
contigs.sorted.deduplicated.read-group.realigned.recalibrated.bam \
```

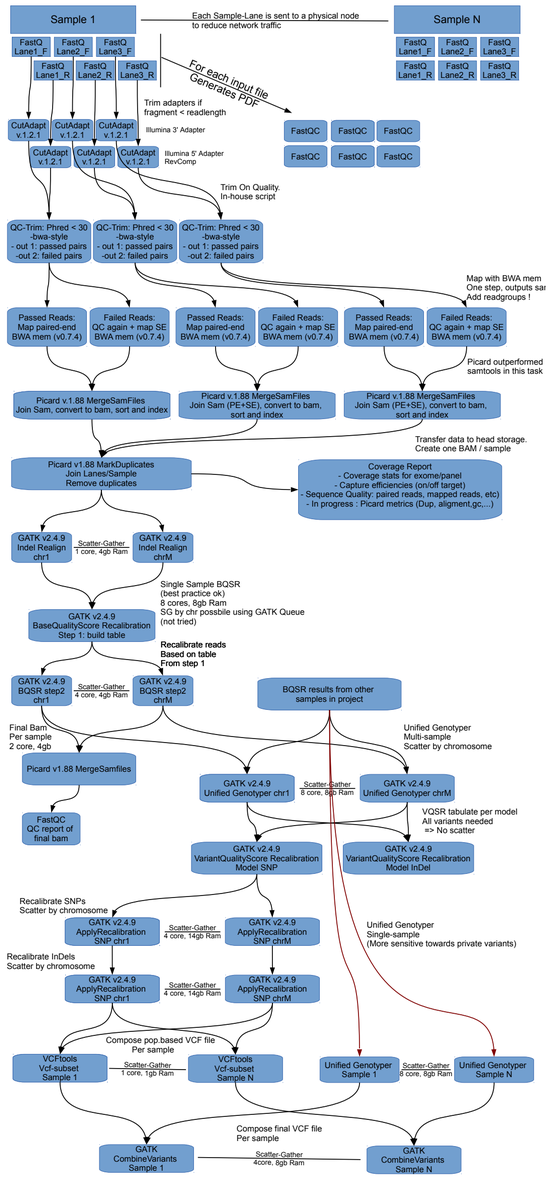
```
target=bioplanet.out
log=$results/timings.log
gatkref=~/data/ucsc_hg19/ucsc.hg19.fasta
bwa=~/bwa/bin/bwa_pthreads
samtools=~/samtools/samtools-0.1.19
vcftools=NIL
picardtools=~/picard/dist/
gatk=~/gatk/GenomeAnalysisTK-2.7-2-g6bda569
dbsnp=/home/caherzee/bwa/example/bundle/vcf/dbsnp_137.hg19.vcf
mkdir -p $results
# convert sai to sam
echo "Converting SAI to SAM" >> $log \
{ time -f "Time: %E real,%U user,%S sys" \
$bwa/bwa samse $ref $target $reads > $results/$results.sam \
;} 2>> $log
# convert sam to bam
echo "Converting SAM to BAM" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
$samtools/samtools view -@ 24 -bS $results/$results.sam > $results/$results.bam \
;} 2>> $log
# removing unmapped reads, cause picard does not like them
echo "Removing unmapped reads from BAM" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
$samtools/samtools view -@ 24 -h -F 4 -b $results/$results.bam > $results/
$results.only_mapped.bam \
;} 2>> $log
# sort for contigs
echo "Sorting BAM for contigs to gatk ref" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -jar $picardtools/ReorderSam.jar I= $results/$results.only_mapped.bam O= $results/
$results.only_mapped.reordered-contigs.bam REFERENCE= $gatkref \
;} 2>> $log
# sort bam file
echo "Sorting BAM for coordinate order" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
$samtools/samtools sort -@ 24 $results/$results.only_mapped.reordered-contigs.bam $results/
$results.only_mapped.reordered-contigs.sorted \
;} 2>> $log
# remove duplicates
echo "Removing Duplicates from BAM" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -jar $picardtools/MarkDuplicates.jar \
     I=$results/$results.only_mapped.reordered-contigs.sorted.bam O=$results/
$results.only_mapped.reordered-contigs.sorted.deduplicated.bam M=$results/$results.picard-
duplicates \
;} 2>> $log
# add read group information
echo "Adding Read Group information to BAM" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -jar $picardtools/AddOrReplaceReadGroups.jar VALIDATION_STRINGENCY=SILENT \
     I=$results/$results.only_mapped.reordered-contigs.sorted.deduplicated.bam O=$results/
$results.only_mapped.reordered-contigs.sorted.deduplicated.read-group.bam \
     RGID=group1 RGLB= lib1 RGPL=illumina RGPU=unit1 RGSM=sample1 \
;} 2>> $log
# build bam index
echo "Building BAM index" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -jar $picardtools/BuildBamIndex.jar I=$results/$results.only_mapped.reordered-
contigs.sorted.deduplicated.read-group.bam VALIDATION_STRINGENCY=SILENT \
;} 2>> $log
# realign around indels
echo "Updating BAM with Realigning around indels (1/2)" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -Xmx2g -jar $gatk/GenomeAnalysisTK.jar \
   -T RealignerTargetCreator \
   -R $gatkref \
   -I $results/$results.only_mapped.reordered-contigs.sorted.deduplicated.read-group.bam \
   -o $results/forIndelRealigner.intervals \
;} 2>> $log
echo "Updating BAM with Realigning around indels (2/2)" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -jar $gatk/GenomeAnalysisTK.jar -T IndelRealigner -R $gatkref -I $results/
$results.only_mapped.reordered-contigs.sorted.deduplicated.read-group.bam \
     -targetIntervals $results/forIndelRealigner.intervals -o $results/
$results.only_mapped.reordered-contigs.sorted.deduplicated.read-group.realigned.bam \
;} 2>> $log
# build bam index
echo "Rebuilding BAM index for gatk" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -jar $picardtools/BuildBamIndex.jar I=$results/$results.only_mapped.reordered-
contigs.sorted.deduplicated.read-group.realigned.bam VALIDATION_STRINGENCY=SILENT \
;} 2>> $log
#Base recalibration
echo "Base recalibration on BAM for gatk (1/3)" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -jar $gatk/GenomeAnalysisTK.jar -T BaseRecalibrator -R $gatkref -I $results/
$results.only_mapped.reordered-contigs.sorted.deduplicated.read-group.realigned.bam \
     -knownSites $dbsnp -o $results/recal.grp \
;} 2>> $log
echo "Base recalibration on BAM for gatk (3/3)" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
java -jar $gatk/GenomeAnalysisTK.jar -T PrintReads -R $gatkref -I $results/
$results.only_mapped.reordered-contigs.sorted.deduplicated.read-group.realigned.bam \
     -BQSR $results/recal.grp -o $results/$results.only_mapped.reordered-
contigs.sorted.deduplicated.read-group.realigned.recalibrated.bam \
;} 2>> $log
# Calling variants
echo "Calling variants" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
```

Observations : complex glue

- Pure file-based I/O
- heavy use of environment variables
    - program parameters (bio domain)
    - program paramaters (technical)
    - tool and file locations
- different tools, different invocation styles, naming conventions
- occasional implicit input and residue files
- not strictly a linear pipeline
- heterogeneous tools
    - big monolithic stages (GATK)
    - small file-manipulation stages (remove unmapped reads)

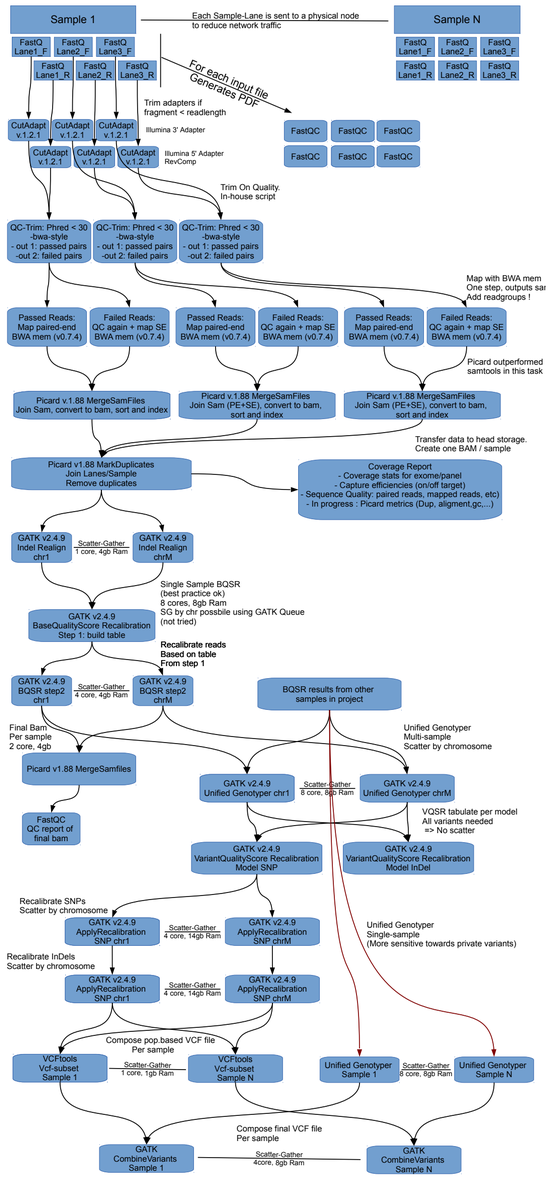
▸ bash / perl does the job

Case #2:
distributed Exome Sequencing



(2500 lines of Perl code removed due to permission issues)

Observations :



complex glue + **complex coordination**

- multiple types of nodes
- use of various job schedulers, grid managers
- data preparation
- manual dependency tracking
- manual splitting/parallelism
- cluster-specific
- mail notification, reporting, experimentation logistics
- monitoring

- order of magnitude more work
- ad-hoc solutions, accidental complexity

Scientific scripting crisis

- So what? We've been gluing with bash/perl scripts for 30 years now.

  game changer: *Computational & Data requirements through the roof*

  Parallel & Distributed computing unavoidable

  complexity through the roof

- Why bash scripting? Stop being lazy and write a parallel application!

  toolbox of sequential programs:

  decades of encoded domain-specific knowledge, heuristics, etc.

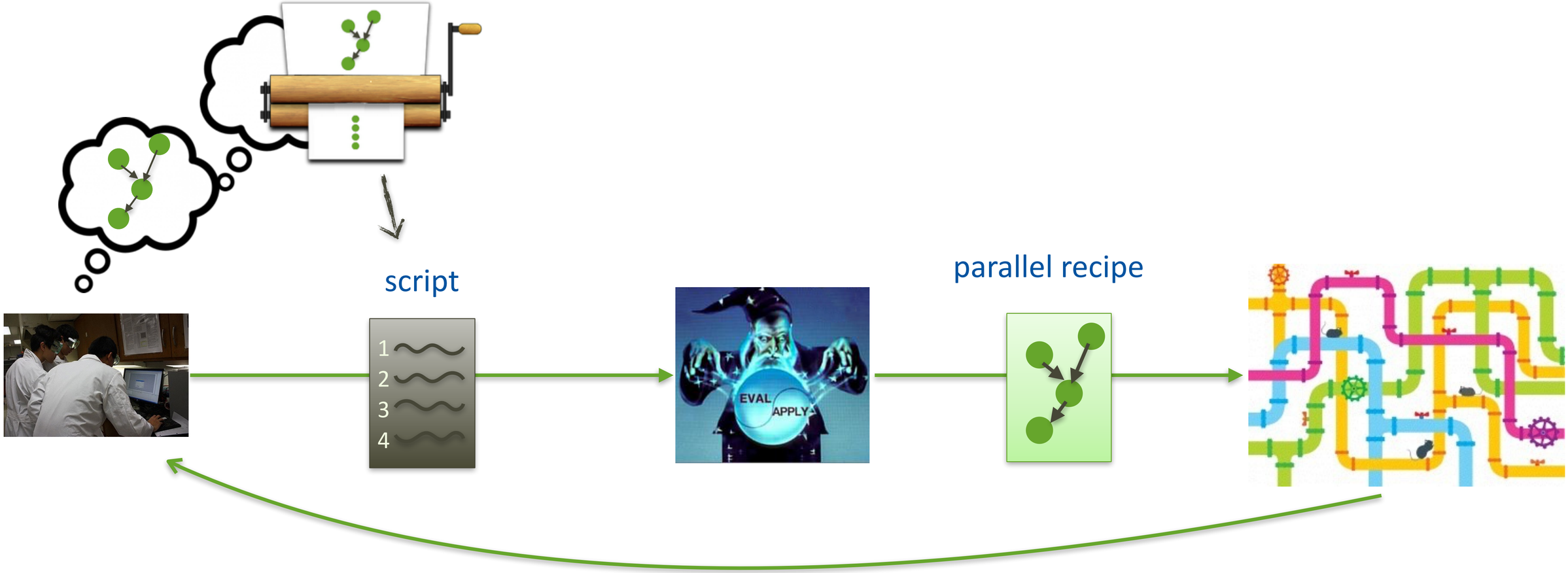
  fast-changing, *heterogeneous* and *dynamic* set of tools

  gold standard switches to tool B while you are still parallelising tool A

  dynamic pipeline structure

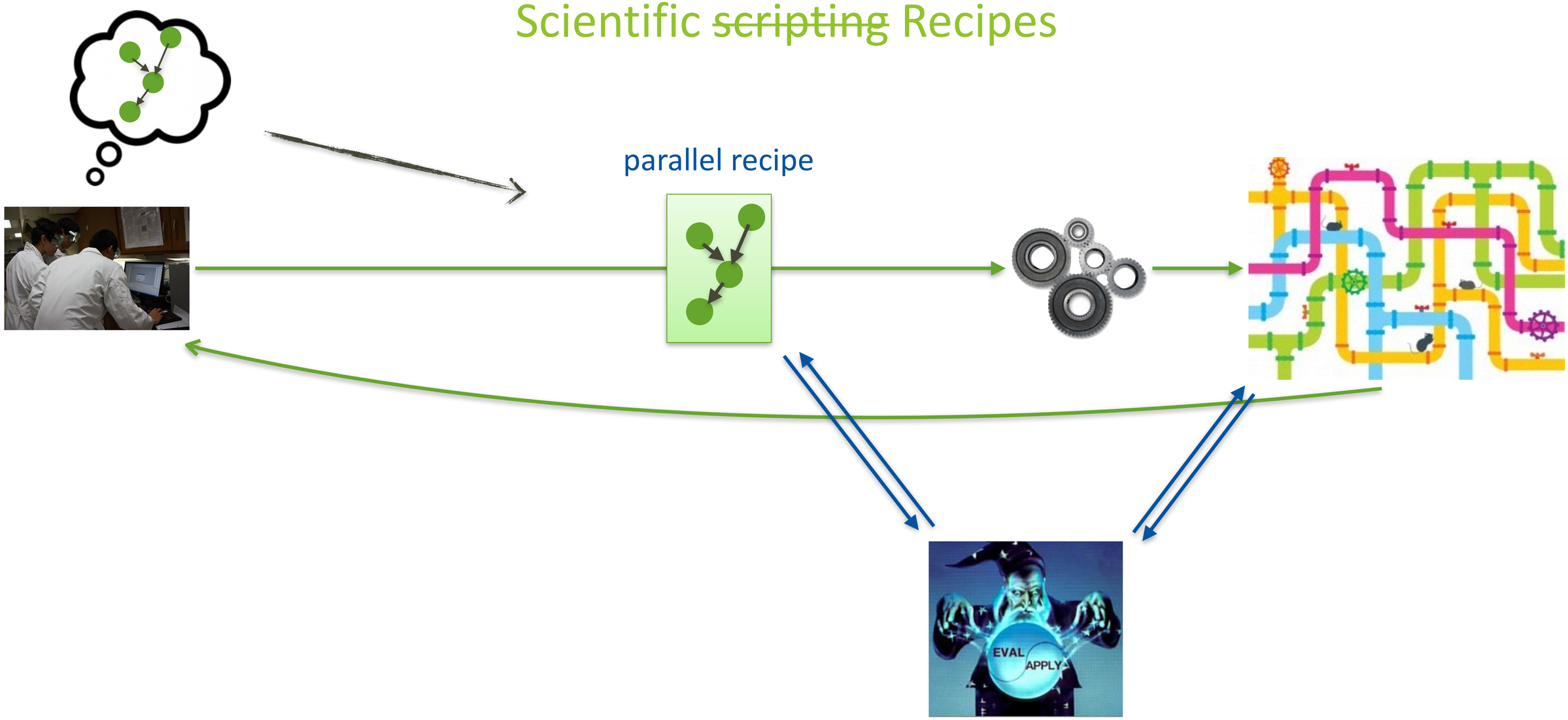
  pipeline designers are **domain experts**, not performance experts

Scientific Scripting:
"Throw it over the Wall" approach



Vision

Scientific ~~scripting~~ Recipes



Experiment: WGS pipeline recipe

Test vision: investigate using WGS recipe pipeline **proto-app**

- Keep it - simple
  - general
- Remove accidental complexity
- Small incremental improvements
- Focus on - multiple samples parallelism
  - distribution

➡ How far can we take this?
➡ How well does this work?
➡ Where does it break down?
➡ Can it be generalised?

Experiment: WGS pipeline recipe

- writing it down: analyse existing script

```
# align reads
echo “Aligning reads using aln" >> $log
$BWA aln -t 24 -q 15 $REF/hg19.fa {}.fastq > {}.sai
# convert sai to sam
echo "Converting SAI to SAM" >> $log \
{ time -f "Time: %E real,%U user,%S sys" \
$bwa/bwa samse $ref $target $reads > $results/$results.sam \
;} 2>> $log
# convert sam to bam
echo "Converting SAM to BAM" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
$samtools/samtools view -@ 24 -bS $results/$results.sam > $results/$results.bam \
;} 2>> $log
# removing unmapped reads, cause picard does not like them
echo "Removing unmapped reads from BAM" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
$samtools/samtools view -@ 24 -h -F 4 -b $results/$results.bam > $results/
$results.only_mapped.bam \
;} 2>> $log
# sort for contigs
# etc. etc.
```

Experiment: WGS pipeline recipe

- writing it down: isolate steps

```
# align reads
echo "Aligning reads using aln" >> $log
$BWA aln -t 24 -q 15 $REF/hg19.fa {}.fastq > {}.sai
echo "Converting SAI to SAM" >> $log \
{ time -f "Time: %E real,%U user,%S sys" \
$bwa/bwa samse $ref $target $reads > $results/$results.sam \
;} 2>> $log
# convert sam to bam
echo "Converting SAM to BAM" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
$samtools/samtools view -@ 24 -bS $results/$results.sam > $results/$results.bam \
;} 2>> $log
# removing unmapped reads, cause picard does not like them
echo "Removing unmapped reads from BAM" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
$samtools/samtools view -@ 24 -h -F 4 -b $results/$results.bam > $results/
$results.only_mapped.bam \
;} 2>> $log
# sort for contigs
# etc. etc.
```

```
$BWA aln -t 24 -q 15 $REF/hg19.fa $reads > $target
```

```
$bwa/bwa samse $ref $target $reads > $results/$results.sam
```

```
$samtools/samtools view -@ 24 -bS $results/$results.sam > $results/$results.bam
```

```
$samtools/samtools view -@ 24 -h -F 4 -b $results/$results.bam > $results/
$results.only_mapped.bam \
```

Experiment: WGS pipeline recipe

- writing it down: identify input/output

```
# align reads
echo "Aligning reads using aln" >> $log
$BWA aln -t 24 -q 15 $REF/hg19.fa {}.fastq > {}.sai

echo "Converting SAI to SAM" >> $log \
{ time -f "Time: %E real,%U user,%S sys" \
$bwa/bwa samse $ref $target $reads > $results/$results.sam \
;} 2>> $log

# convert sam to bam
echo "Converting SAM to BAM" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
$samtools/samtools view -@ 24 -bS $results/$results.sam > $results/$results.bam \
;} 2>> $log

# removing unmapped reads, cause picard does not like them
echo "Removing unmapped reads from BAM" >> $log
{ time -f "Time: %E real,%U user,%S sys" \
$samtools/samtools view -@ 24 -h -F 4 -b $results/$results.bam > $results/
$results.only_mapped.bam \
;} 2>> $log
# sort for contigs
# etc. etc.
```

```
$BWA aln -t 24 -q 15 $REF/hg19.fa $reads > $target
```

```
$bwa/bwa samse $ref $target $reads > $results/$results.sam
```

```
$samtools/samtools view -@ 24 -bS $results/$results.sam > $results/$results.bam
```

```
$samtools/samtools view -@ 24 -h -F 4 -b $results/$results.bam > $results/
$results.only_mapped.bam \
```

Experiment: WGS pipeline recipe

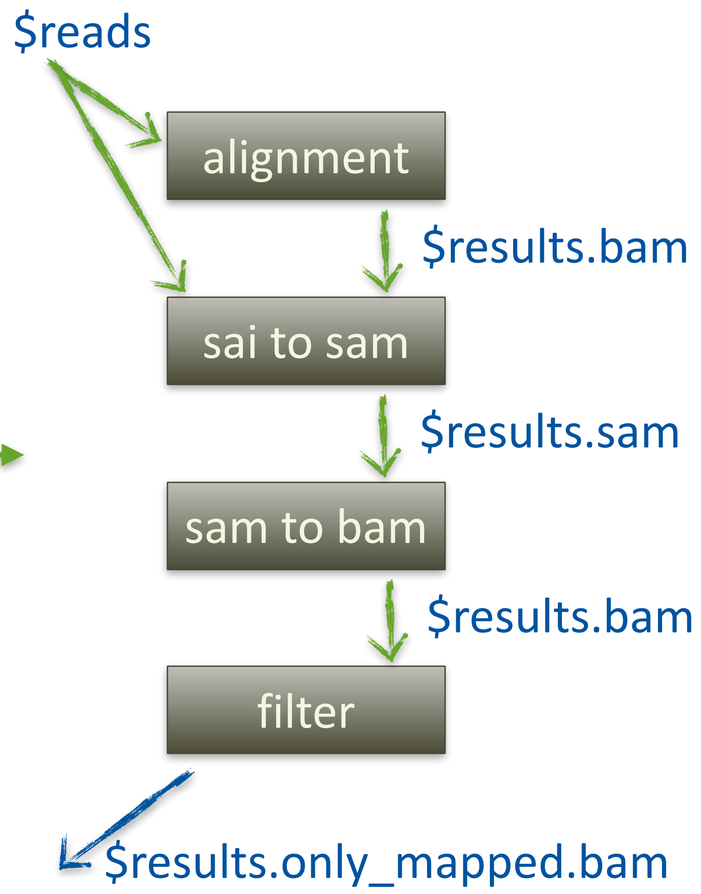
- writing it down: identify dependencies

```
$BWA aln -t 24 -q 15 $REF/hg19.fa $reads > $target
```

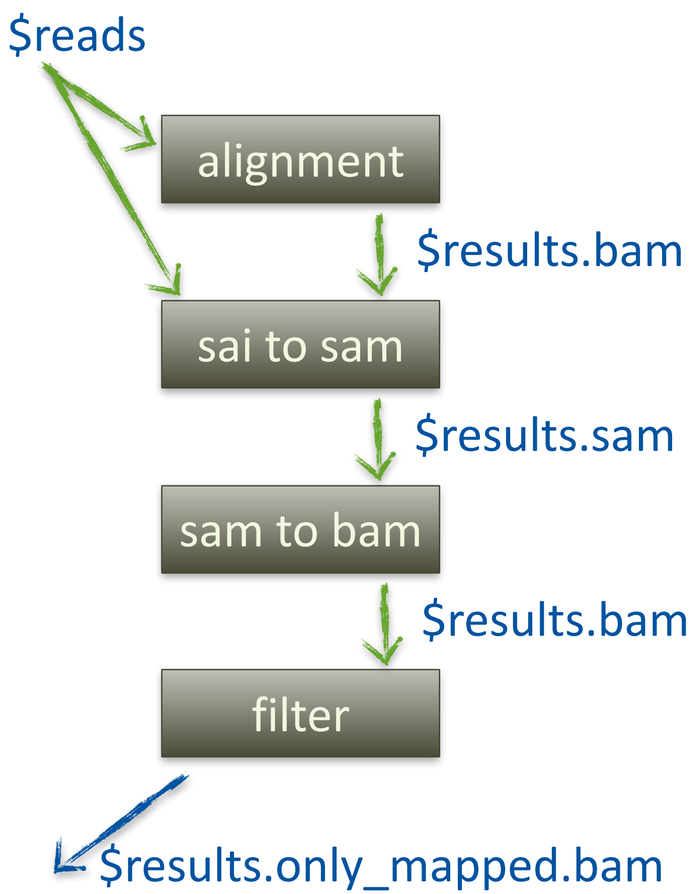
```
$bwa/bwa samse $ref $target $reads > $results/$results.sam
```

```
$samtools/samtools view -@ 24 -bS $results/$results.sam > $results/$results.bam
```

```
$samtools/samtools view -@ 24 -h -F 4 -b $results/$results.bam > $results/
$results.only_mapped.bam \
```



Experiment: WGS pipeline recipe

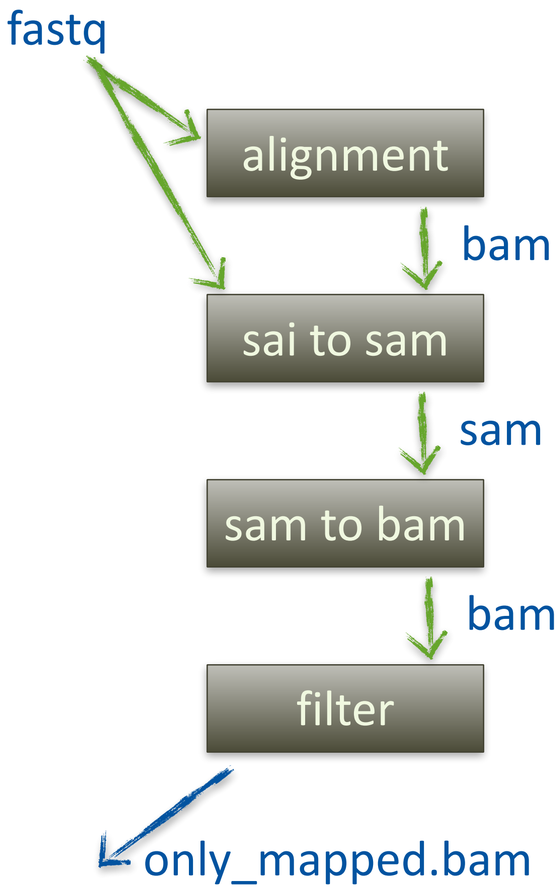


?

How to concretely

- write it down
- execute in parallel

- writing it down: pipeline recipe



write down

JSON pipeline configuration file

```
{
 "stages" : {
    "alignment" : {
       "command" : " $BWA aln -t 24 -q 15 $REF/hg19.fa {}.fastq > {}.sai",
       "in"  : "fastq", // also reads hg19
       "out" : "sai"
    },
    "sai_to_sam" : {
       "command" : " $BWA samse $REF/hg19.fa {}.sai {}.fastq > {}.sam",
       "in"  : ["fastq", "sai"], // also reads hg19
       "out" : "sam"
    },
    "sam_to_bam" : {
       "command" : " $SAMTOOLS/samtools view -@ 24 -bS {}.sam > {}.bam",
       "in"  : "sam",
       "out" : "bam"
    },
 ...
```

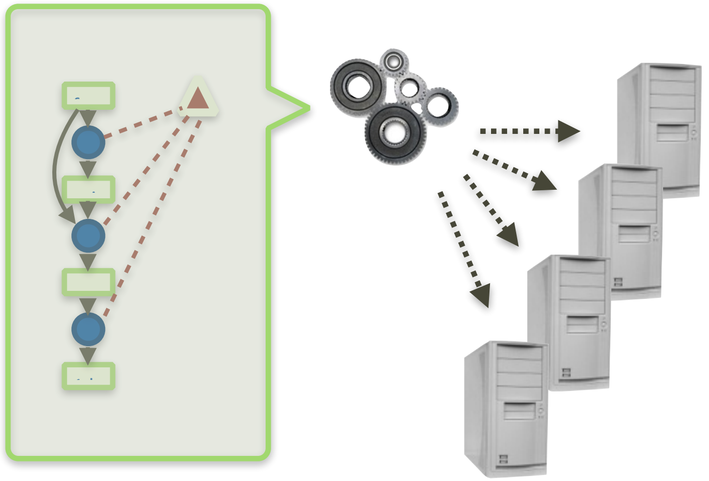
- writing it down: pipeline recipe

JSON pipeline configuration file

```
{
 "stages" : {
        "alignment" : {
           "command" : " $BWA aln -t 24 -q 15 $REF/hg19.fa {}.fastq > {}.sai",
           "in"  : "fastq", // also reads hg19
           "out" : "sai"
        },
        "sai_to_sam" : {
           "command" : " $BWA samse $REF/hg19.fa {}.sai {}.fastq > {}.sam",
           "in"  : ["fastq", "sai"], // also reads hg19
           "out" : "sam"
        },
        "sam_to_bam" : {
           "command" : " $SAMTOOLS/samtools view -@ 24 -bS {}.sam > {}.1.bam",
           "in"  : "sam",
           "out" : "1.bam"
        },
 ...
```

transform to internal representation

parallel execution engine



- execution

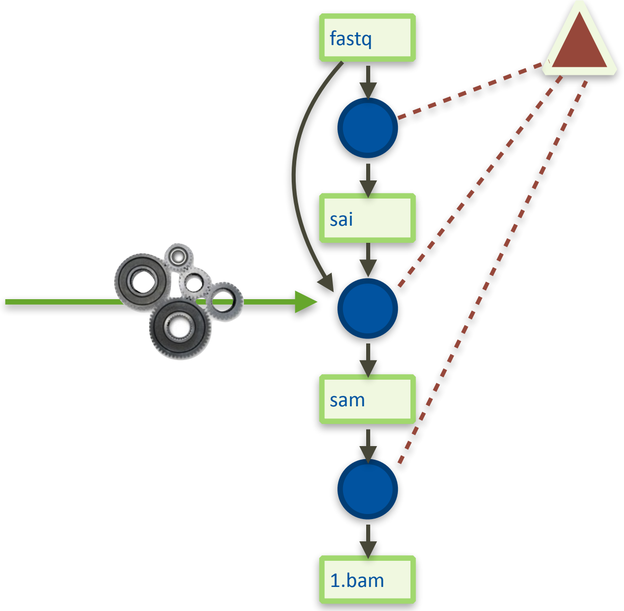
bash$ ./pipeline biopipe.json sample_{00..07}

```
// create CnC graph
> pipeline = parse_json(argv[1])
```

>

biopipe.json

```
{
 "stages" : {
      "alignment" : {
         "command" : " $BWA aln -t 24 -q 15 $REF/hg19.fa {}.fastq > {}.sai",
         "in"  : "fastq", // also reads hg19
         "out" : "sai"
      },
      "sai_to_sam" : {
         "command" : " $BWA samse $REF/hg19.fa {}.sai {}.fastq > {}.sam",
         "in"  : ["fastq", "sai"], // also reads hg19
         "out" : "sam"
      },
      "sam_to_bam" : {
         "command" : " $SAMTOOLS/samtools view -@ 24 -bS {}.sam > {}.1.bam",
         "in"  : "sam",
         "out" : "1.bam"
      },
 ...
```

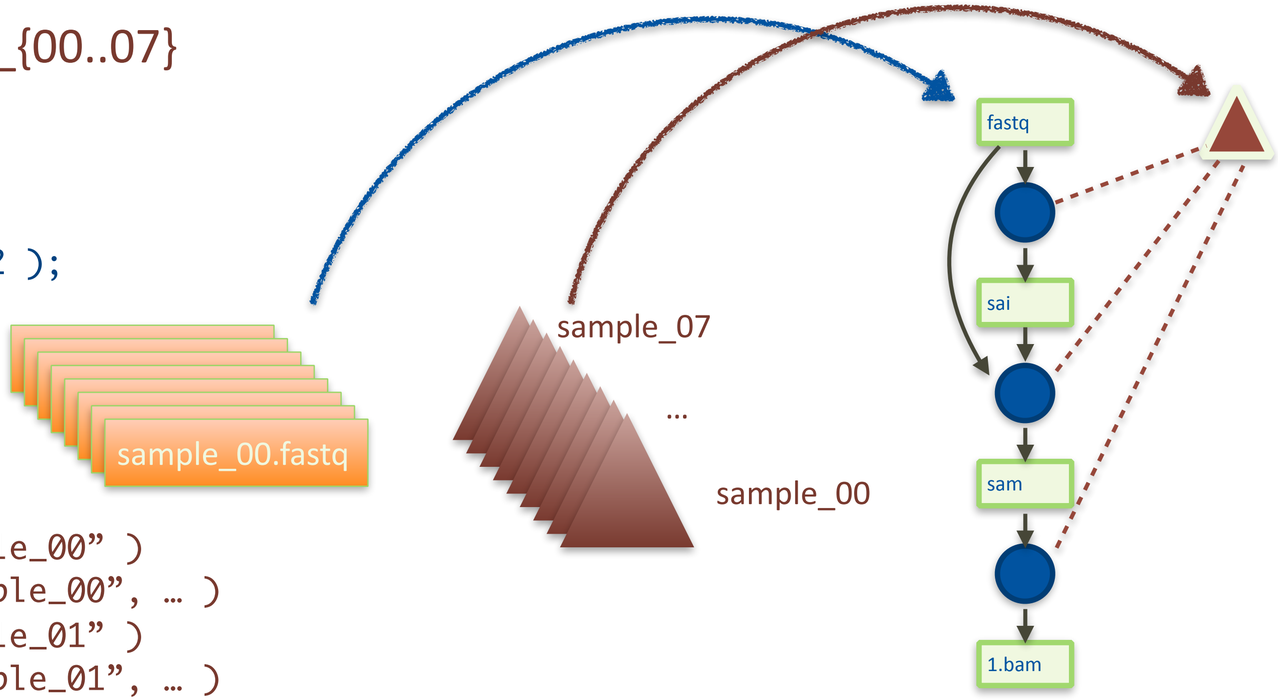



```
function alignment(tag) {
   file = fastq.get(tag)
   b_j = b.get(j)
   new_tag = size_b*i + j
   c.put( new_tag, a_i*b_j )
}
```

- execution

bash$ ./pipeline biopipe.json sample_{00..07}

```
// start running samples in parallel
> for( int i = 2; i < argc; ++i )
    pipeline.run( argv[i], “fastq”, i-2 );
```




```
> pipeline.tags.put( “sample_00” )
> pipeline.fastq.put( “sample_00”, … )
> pipeline.tags.put( “sample_01” )
> pipeline.fastq.put( “sample_01”, … )
…
```

- execution

```
bash$  ./pipeline biopipe.json sample_{00..07}
```

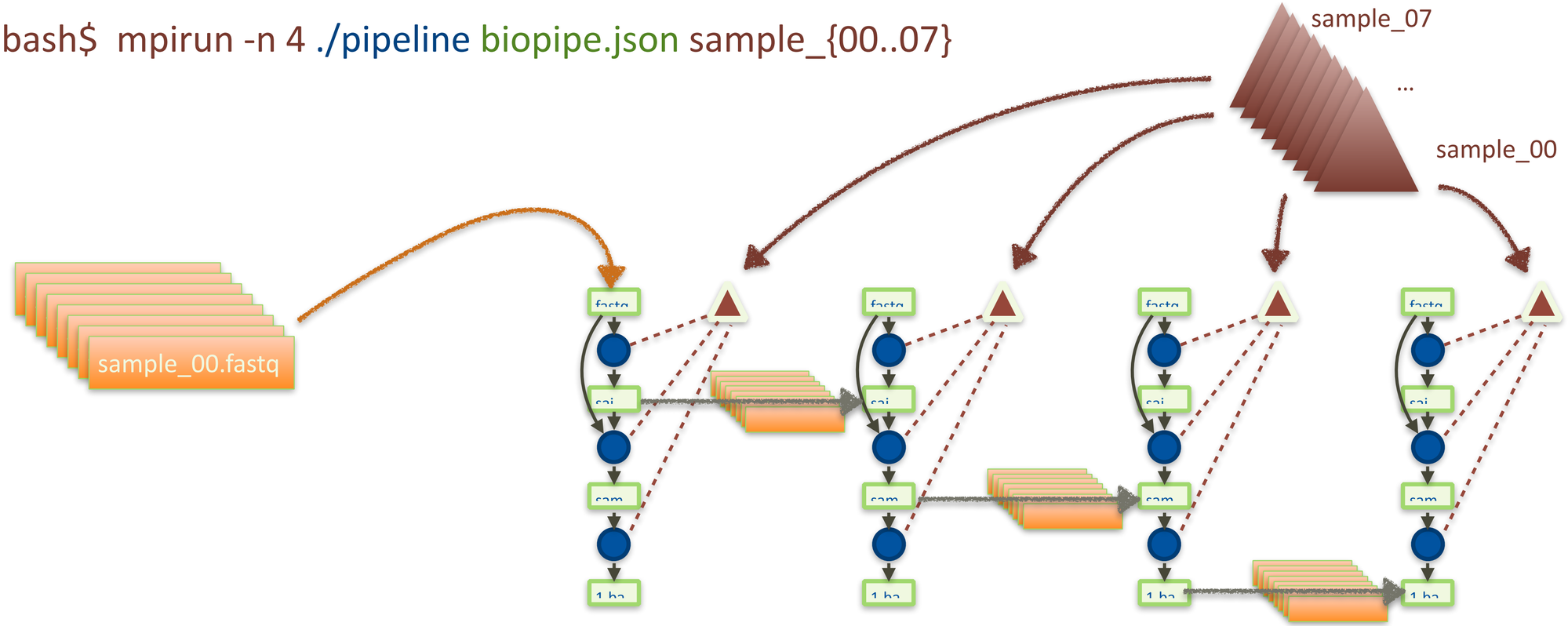
Works great! Parallel speedup on single node!

What about distributed execution?

- distributed execution : step 1

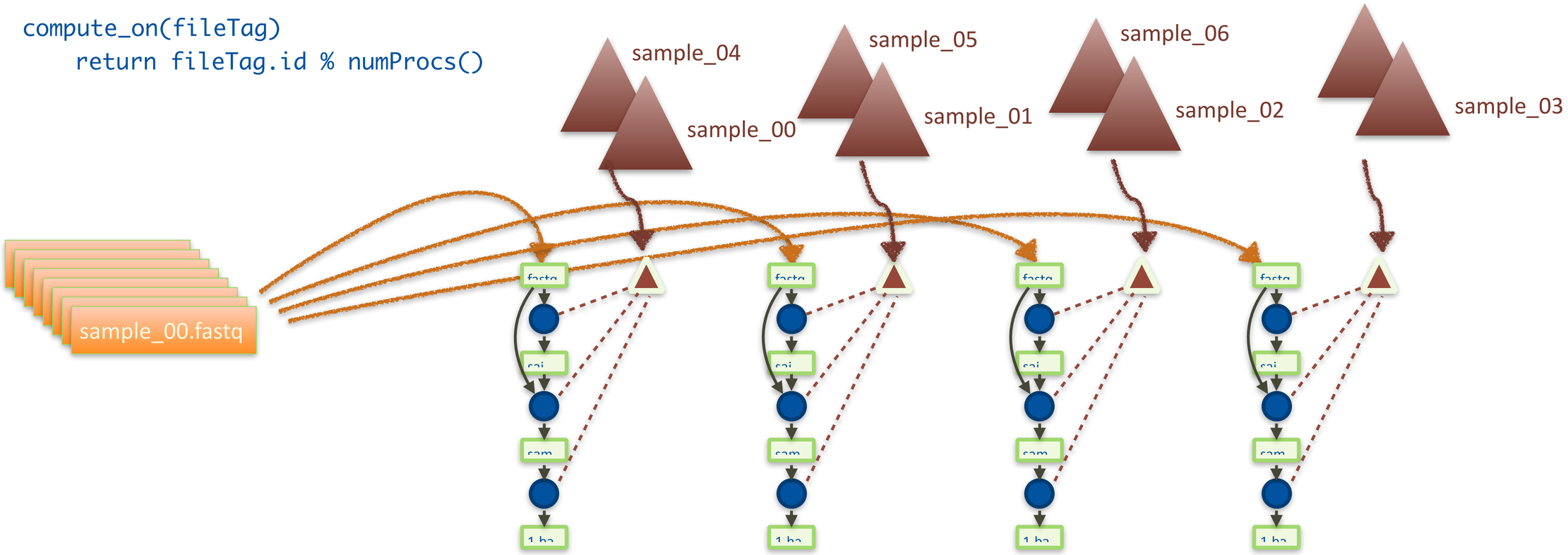
default round robin scheduling of steps not ideal

bash$ mpirun -n 4 ./pipeline biopipe.json sample_{00..07}



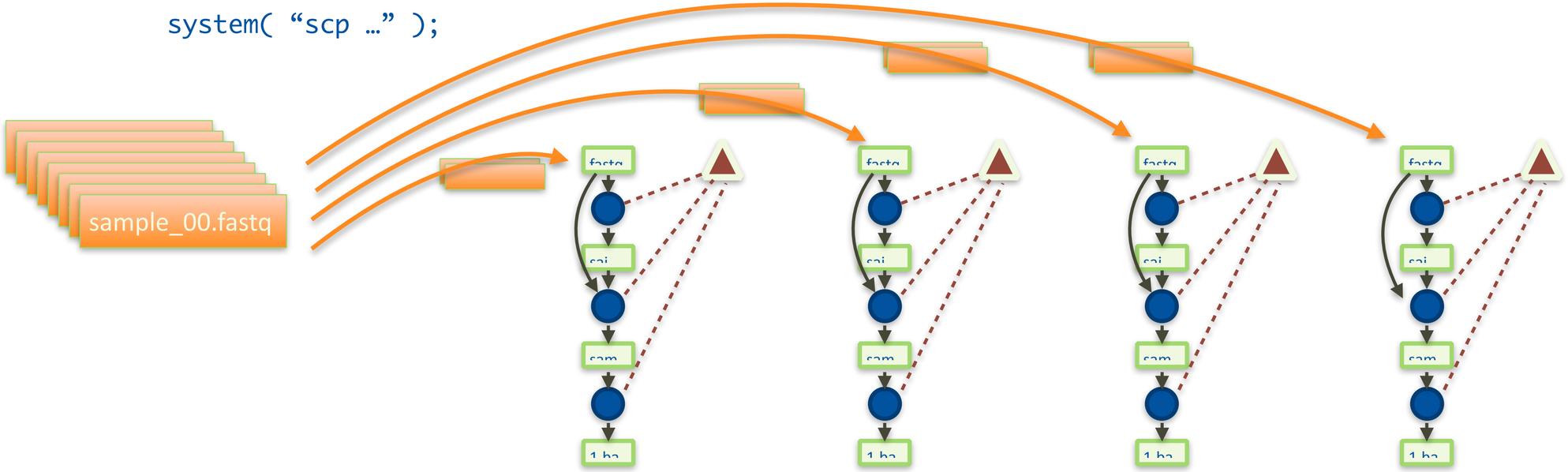
- distributed execution : step 2
  - tuning: define own tag distribution

```
compute_on(fileTag)
    return fileTag.id % numProcs()
```

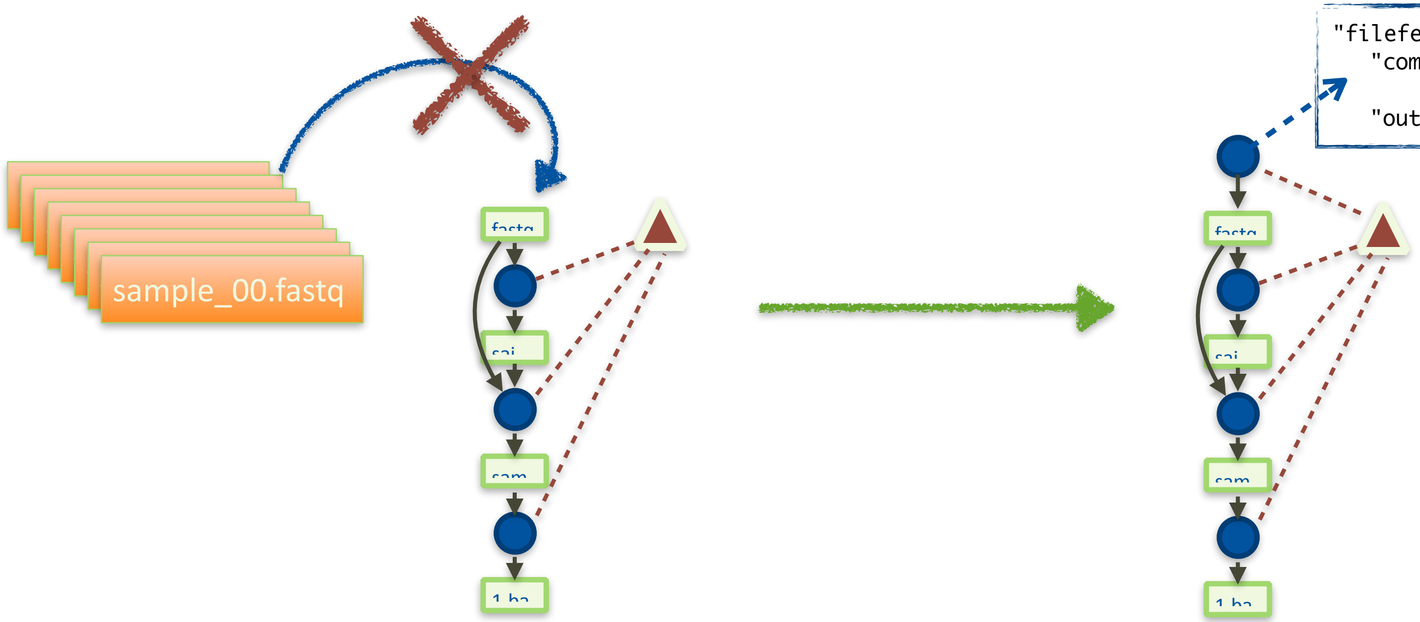


- distributed execution : step 3
  - reliance on distributed File System bottleneck!
  - change needed: use local disk exclusively
    - change {}.ext filenames in script by $LOCAL_DIR/{}.ext
    - define serialisation, copy over network

```
system( “scp …” );
```



- distributed execution : step 4
  - use local disk exclusively
  - simple alternative:
    - define ‘file fetcher’ step explicitly
    - pipeline.run() can remove hardcoded file item adds




```
"filefetcher" : {
   "command" : "cp $READS/{}.fastq
                  $LOCAL_DIR/{}.fastq",
   "out" : “fastq"  }
```

./pipeline biopipe.json sample_{00..07}

```
> pipeline.tags.put( “sample_00” )
> pipeline.tags.put( “sample_01” )
> …
```

Experiment: Benchmarks

Benchmark Setup

- Illumina WGS dataset ERR091575_1 (16GB)
  - cut into ~750MB pieces (21 samples)
  - compromise multiple samples, realistic workload, benchmarkable timeframe
- Lynx cluster nodes
- Reference pipeline, multithreaded where possible (Cilk BWA aln, Picard)

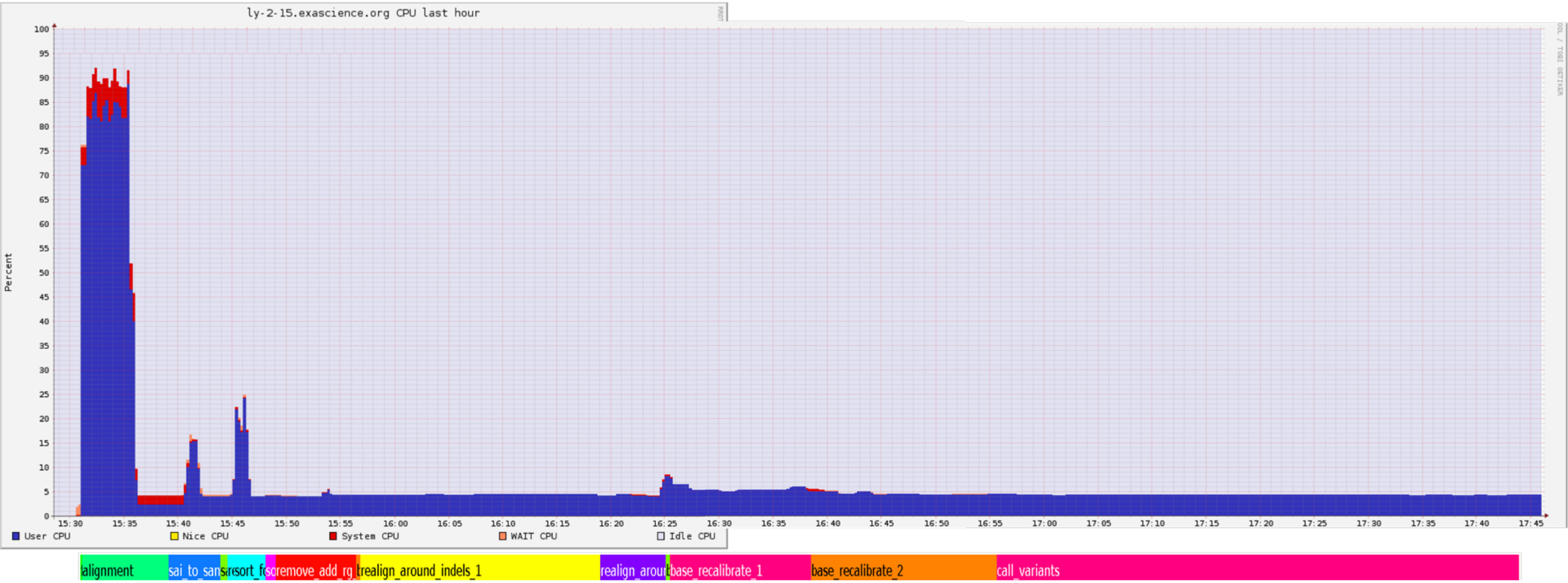
Baseline

16 samples, 1 node, 1 worker

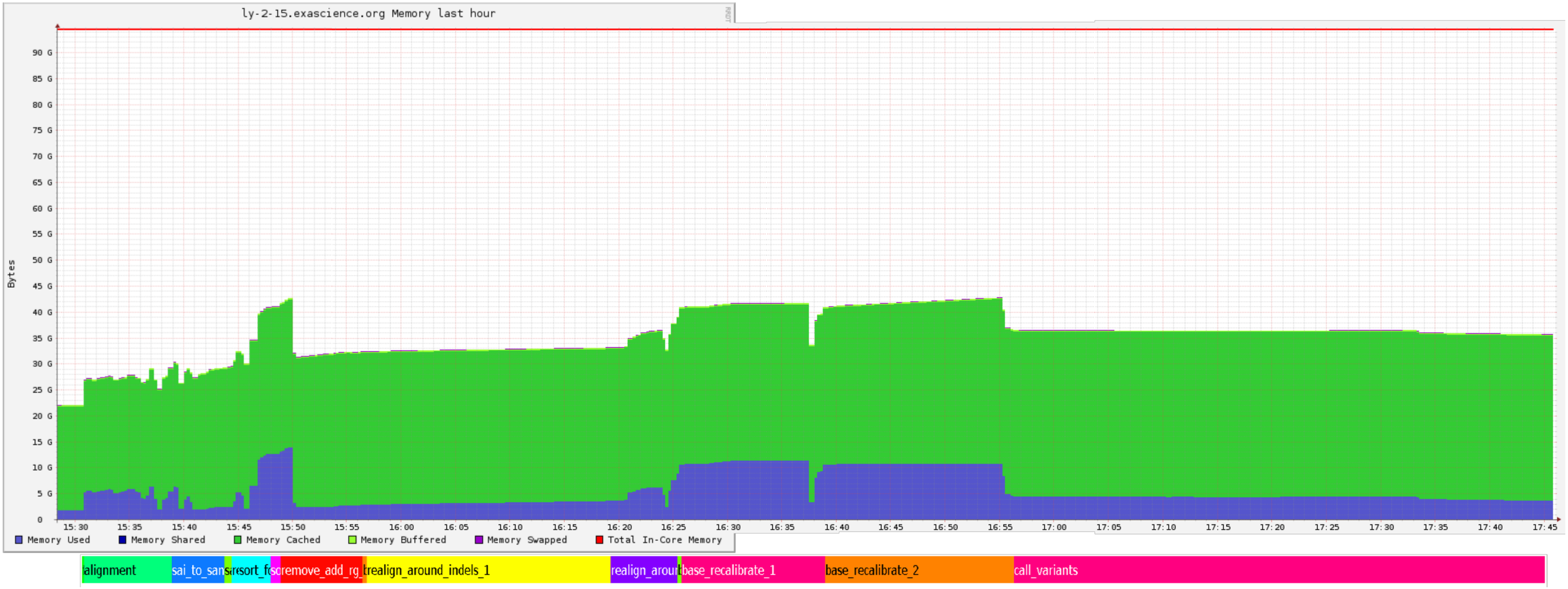
52h 59m 24s

52h 40m 2s

Baseline observations — CPU

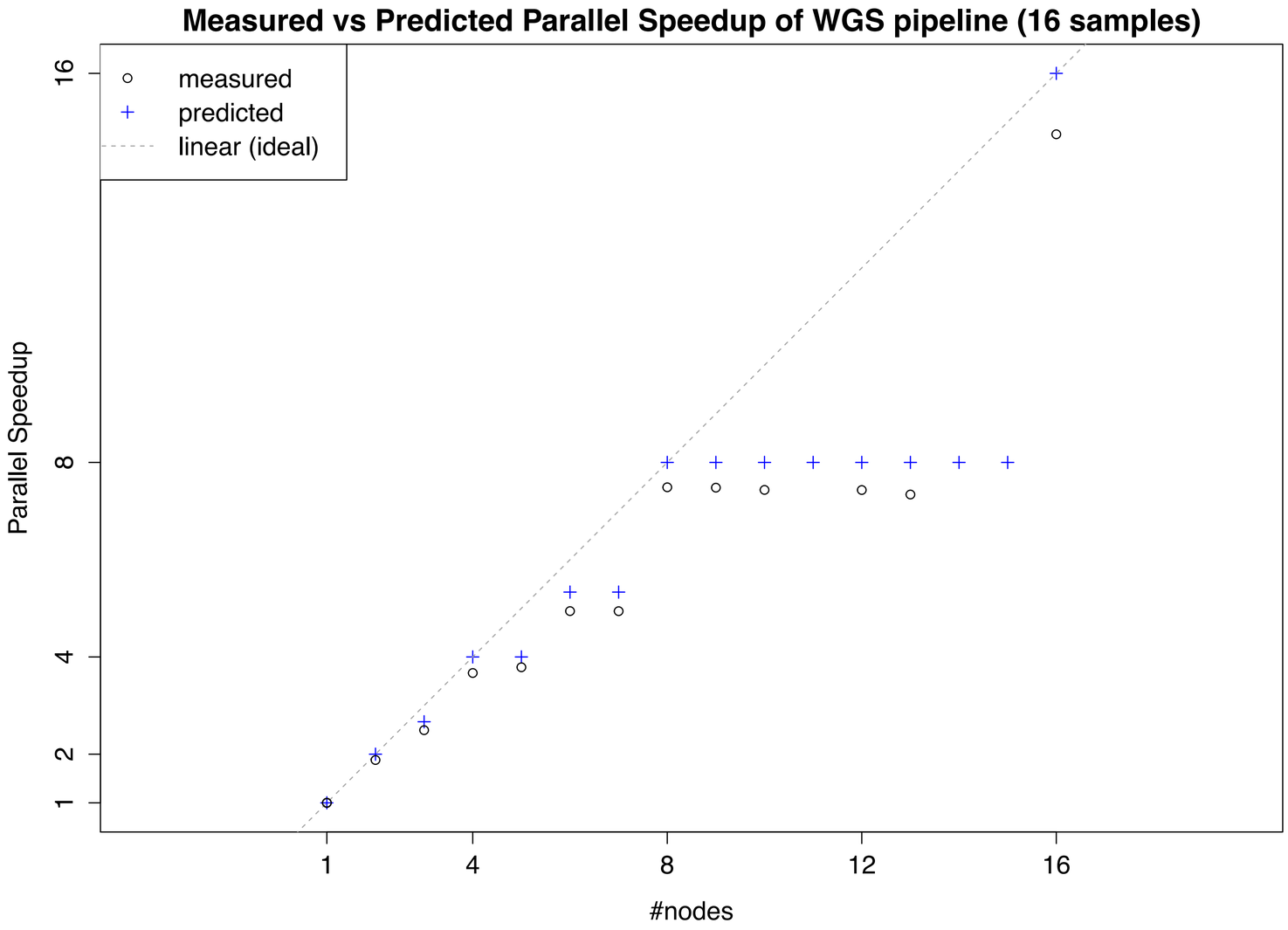


Baseline observations — memory



Scaling experiment

16 samples, N nodes, 1 worker



1 node:	52h	49m43s	
4 nodes:	14h	24m8s	—> **3.67x**
8 nodes:	7h	3m 16s	—> **7.49x**
16 nodes:	3h	34m56s	—>**14.74x**

16 samples, 4 nodes, 1 workers

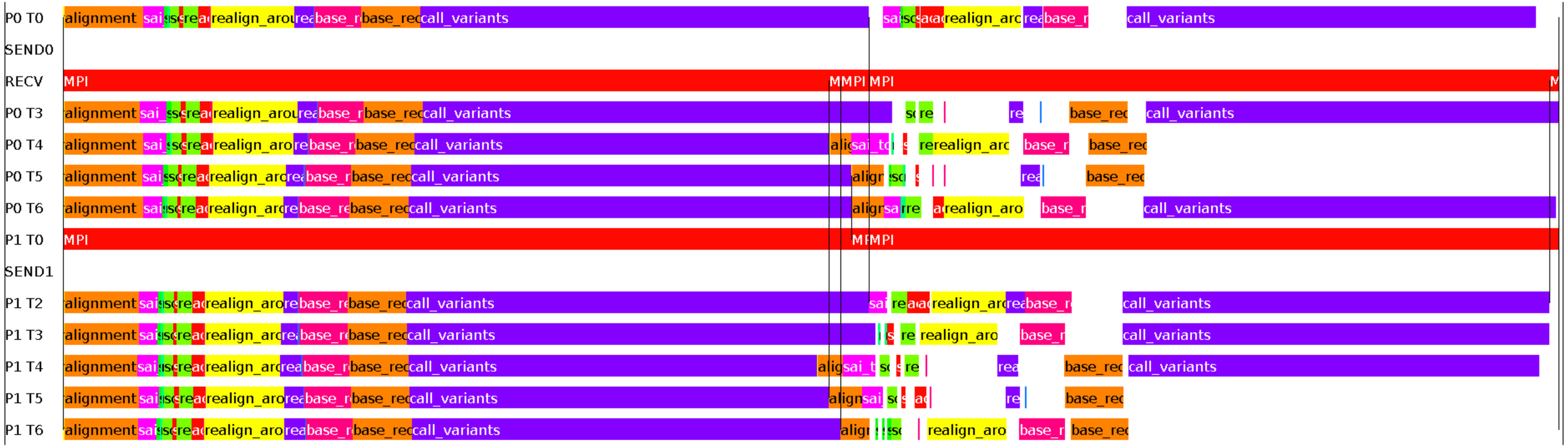
14h 24m 8s 3,7x speedup

16 samples, 4 nodes, 7 workers

3h 56m 29s 13,4x speedup

16 samples, 2 nodes, 5 workers

7h 37m 04s 6,9x speedup



Sequencing Experiment: Conclusions

The good

CnC delivers: proper parallel execution basis

- cluster-level and node-level parallelism
- determinate execution
- flexible parallel execution model
- stable & practical implementation

Tuning for distribution

- small changes
- external to pipeline code

Integrates well with external environment:

- for benchmarks: 50% SLoC is bash scripts
- finicky sequencing tools, setting up environment
- setting up all files, preparing samples/experiment, etc.

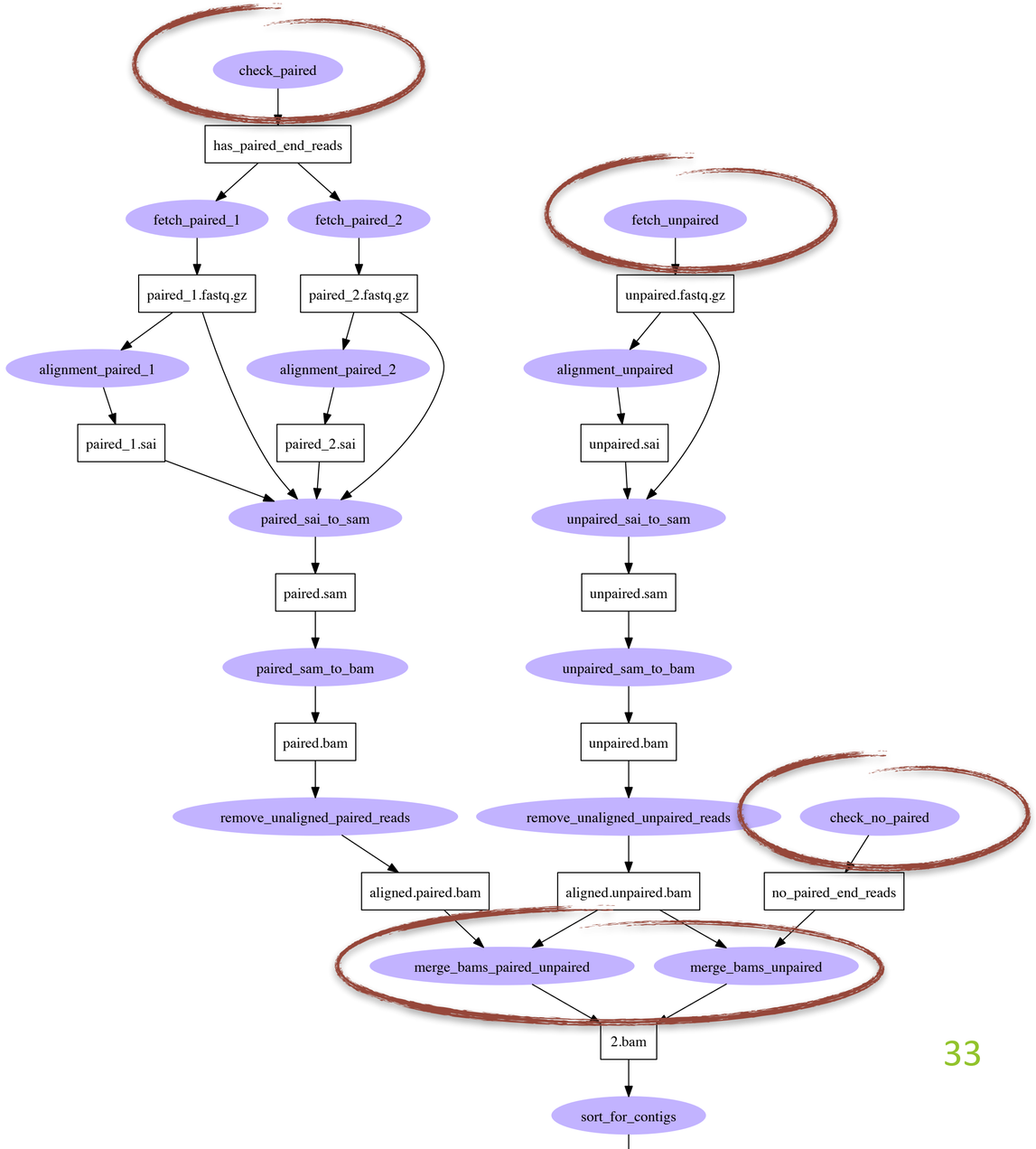
Sequencing Experiment: Conclusions

The good

Intel Trace Analyser Toolkit !

Data-driven logic matches domain logic:

- concurrent (?) file fetching
- event-driven choices



Whole-Genome Experiment: Conclusions

The bad

Manual resource management

- #worker threads / node
- #nodes
- manually delete intermediate results

Static load balancing

- round robin on input ‘samples’

Towards the future

Wish List — Front End

Galaxy or Taverna workflow -> pipeline script?

quality of life

- environment variables, setup
- failure reporting & handling
- separate distribution configuration
- dryrun mode
- expected speedup feedback

front end language ?

- generate the json pipeline script
- avoid errors due to
  duplication, tedium, breaking conventions

```
define hgref file(~/data/ref/hg19.fa)
stage alignment( in: reads, out: sai ) {
   bwa aln -q 15 @hgref @reads > @sai
}
```

```
"alignment" : {
   "command" : " $BWA aln -t 24 -q 15 ~/data/ref/hg19.fa {}.fastq > {}.sai",
   "in"  : "fastq", // also reads hg19
   "out" : "sai"
}
```

getting dangerously close to Argonne's Swift language

Towards the future

Pragmatic — Back end

Deal with real-world issues

CnC provides the infrastructure & hooks for these

load balancing

- scalable, cluster-wide, dynamic and distributed
- resource-aware scheduling, collect runtime heuristics (memory, disk, …)
- data-driven scheduling
- task priorities (file transfers -> background)

collect garbage

error/failure resilience

Database integration?

Thank You!

Questions?