IX International Course of Massive Data Analysis FOR GENOMICS

ciberer
CENTRO DE INVESTIGACIÓN BIOMÉDICA EN RED DE ENFERMEDADES RARAS

PRINCIPE FELIPE
CENTRO DE INVESTIGACION

INB

Joaquín Tárraga
jtarraga@cipf.es

HPG Aligner and sequence alignment libraries

Overview

- Introduction
- Getting the source code: downloading, compiling and building
- Browsing the source code
  - FastQ and BAM file libraries
    - Hands-on
  - BWT and SW aligment libraries
    - Hands-on
  - Parallelization schema
    - Hands-on

Let's talk about...

- Introduction
- Getting the source code: downloading, compiling and building
- Browsing the source code
  - FastQ and BAM file libraries
    - Hands-on
  - BWT and SW aligment libraries
    - Hands-on
  - Parallelization schema
    - Hands-on

Introduction
What's HPG Aligner ?

- A High Performance mapper for Genomic sequences: ultrafast and highly sensitive

Reference genome : AGCATG**TTAGATAAGATAGCTG**TGCTAGTAGGCAGT**CAGCGCCAT**TA....

1 7 37

Input: reads (FastQ file)

```
@read1
CAGCGCCAT
+
#?BBA##2;b#
@read2
TTAGATAAAGGATACTG
+
B<<!>?@?<@CCCCCCC
…
...
```

./hpg-aligner dna

Output: alignments (BAM file)

```
@HD …
@SQ SN:1 LN:2323424
..
read1 83 1 37 30  9= = 7 -39 CAGCGCCAT *
read2 163 1 7 30 8M2I4M1D3M = 37 39 TTAG...
…
...
```

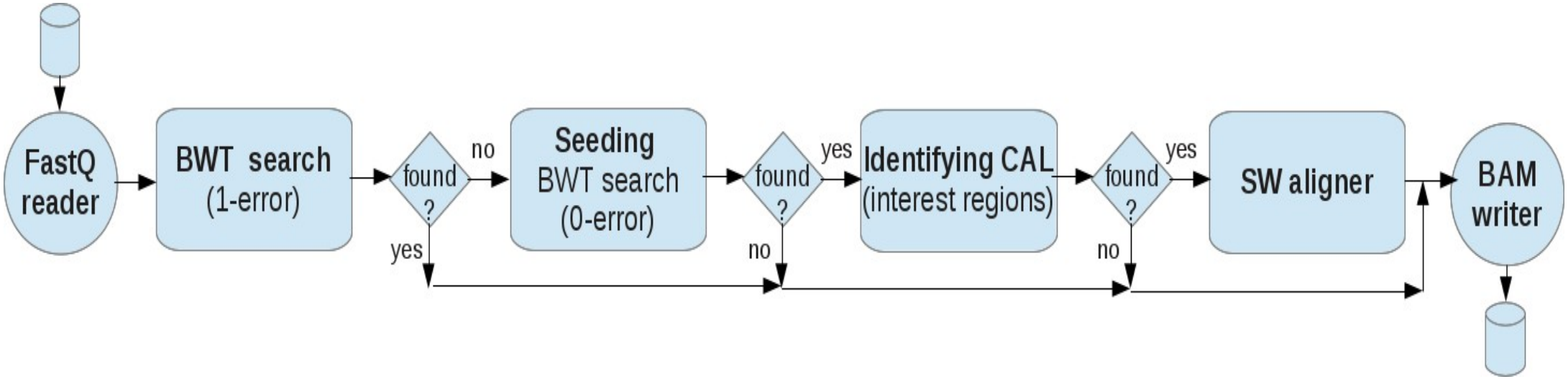
ref: TTAGATAA ** GATAGCTG
read2: TTAGATAAAGGATA* CTG

Joaquín Tárraga
jtarraga@cipf.es

Introduction
Why ultrafast and highly sensitive ?

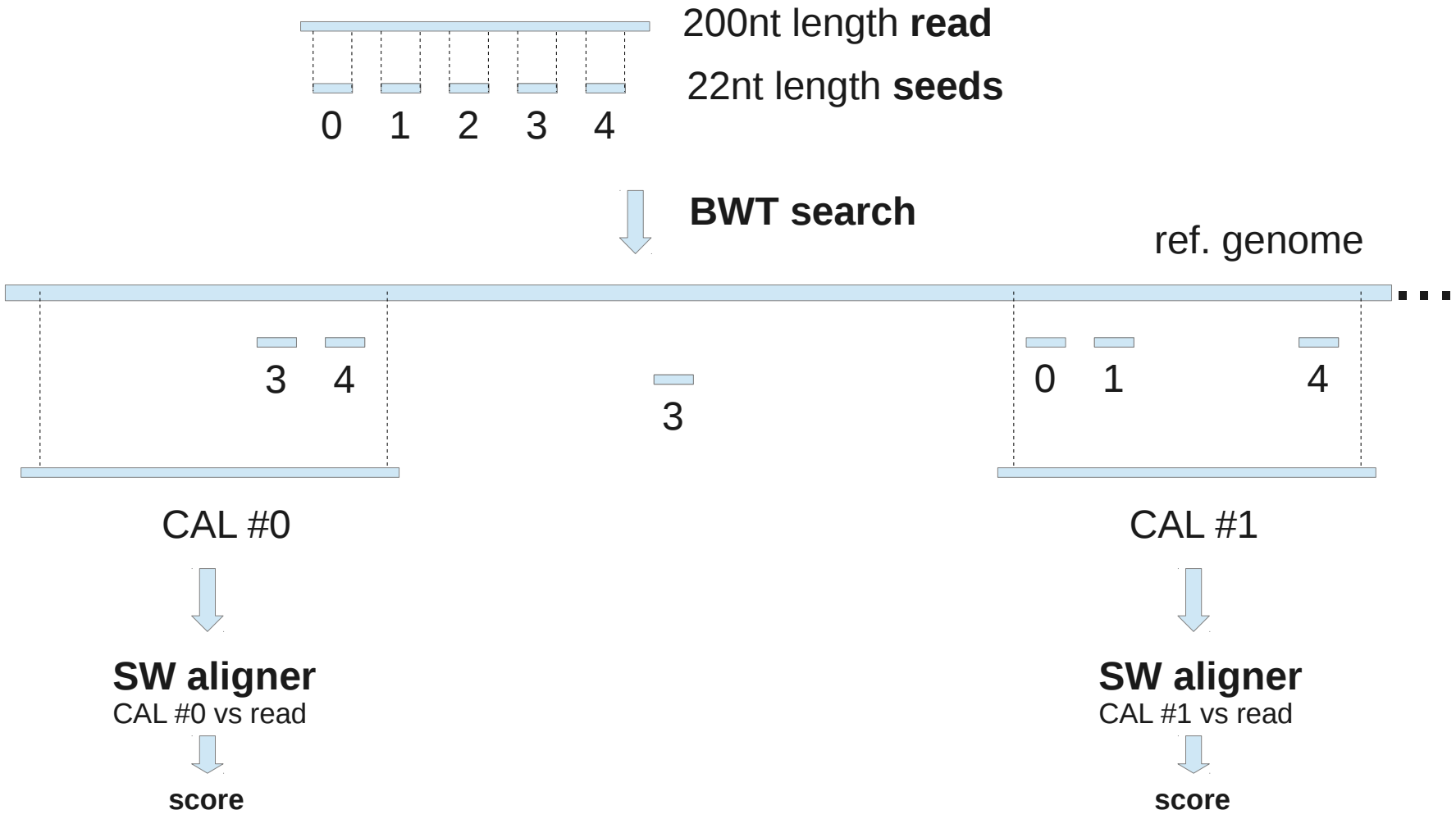
- Based on an efficient pipeline that maximizes the effective hardware utilization
- Combined strategy of two algorithms
    - Burrows-Wheeler Transform (BWT)
        - Fast but low sensitivity (few errors allowed)
    - Smith-Waterman (SW)
        - High sensitivity (many errors allowed, big indels,...) but slow
- Based on HPC technologies to provide the fastest runtime on multicore machines: OpenMP (multitrheading), SSE instructions, GPUs...
- Efficient programming language: C (C99)

Introduction
Pipeline & BWT-SW combined strategy



BWT = Burrows-Wheeler Transform
CAL = Candidate Alignment Localization
SW = Smith-Waterman algorithm

Introduction
Seeding & identifying candidate alignment localizations (CALs)



Introduction
HPG Aligner benchmarks and results: *DNA* sequencing

- First results show an amazing ***performance*** and the best ***sensitivity***

DNA 2M simulated datasets

Program	100nt %mapped Time(min)	150nt %mapped Time(min)	250nt %mapped Time(min)
HPG Aligner *dna* mode	96.22% 1.26min	96.98% 1.9min	97.83% 3.7min
BWA 0.6.2	93.58% 4.3min	92.6% 6.3min	98.0% 9.7min
Bowtie 0.12.8	79.95% 1.8min	60.11% 2.42min	-
Bowtie2 2.0.0	94.71% 2.48	96.75% 3.25	98% 5.75

DNA 2M simulated INDEL datasets

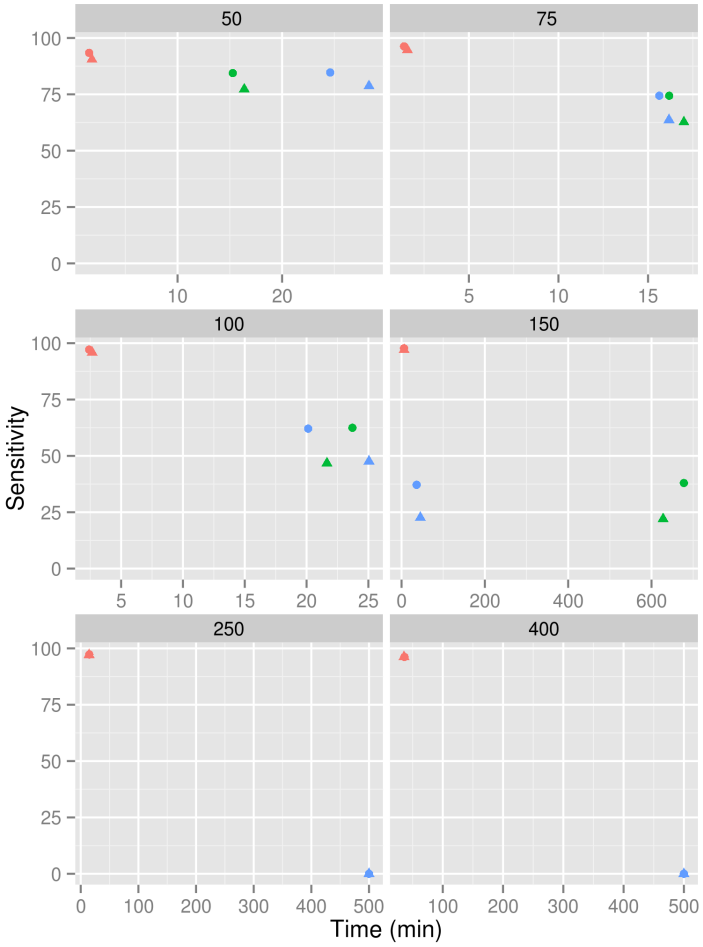
Program	250nt %mapped I10	400nt %mapped I10	800nt %mapped I15
HPG Aligner *dna* mode	66.00%	66.00%	56.00%
BWA-SW 0.6.2	27.44%	26.93%	98.0%
Bowtie2 2.0.0	10.06%	4.80%	0.13%

- Right mapped results
- No GPUs were used
- Other tools were benchamarked: GEM, SOAP, BFAST,... but no positive result were obtained

Joaquín Tárraga
jtarraga@cipf.es

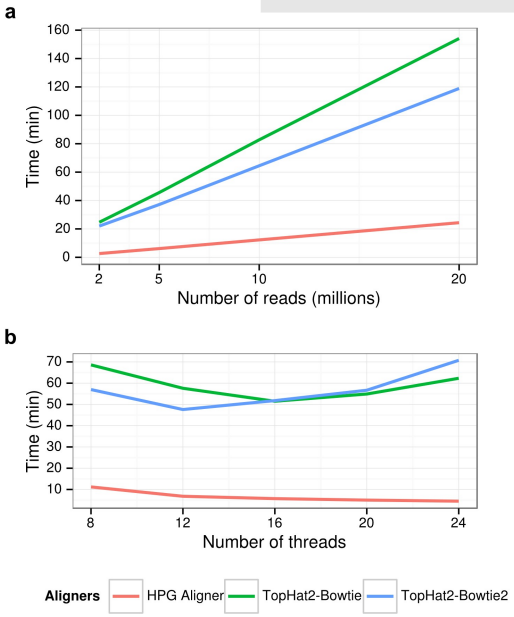
Introduction
HPG Aligner benchmarks and results: *RNA* sequencing

Comparative results with TopHat2 using both Bowtie and Bowtie2



RNA-seq 1M simulated datasets

Program	75nt %mapped Time(min)	100nt %mapped Time(min)	150nt %mapped Time(min)
HPG Aligner *rna* mode	96.5% 1.38min	97.1% 2.41min	97.7% 5.91min
TopHat 2.0.4	74.4% 15.6min	62.1% 20.1min	37.1% 36.2min



Notes:

- Max errors allowed
- Similar results are obtained with real datasets
- TopHat doubles disk space and big memory needs

10x faster!!

Hardware scalability tests

Joaquín Tárraga
jtarraga@cipf.es

Let's talk about...

- Introduction
- Getting the source code: downloading, compiling and building
- Browsing the source code
  - FastQ and BAM file libraries
    - Hands-on
  - BWT and SW aligment libraries
    - Hands-on
  - Parallelization schema
    - Hands-on

Getting the source code
Main resources and documentation

- Resources at www.opencb.org
  - Overview
  - Downloads
  - User manual, getting started, tutorial
  - Developers' corner
  - Release notes and roadmap
  - Results and bechmarks

Getting the source code
Downloading the source code (I)

- Fork the following repositories at github.com
  - opencb-hpg/hpg-aligner
  - opencb-hpg/bioinfo-libs
  - opencb-hpg/common-libs

Getting the source code
Downloading the source code (and II)

- Clone the repositories
- Recommended cloning setup
  - $HOME/appl/bioinfo-c/hpg-aligner
  - $HOME/appl/bioinfo-c/libs/bioinfo-libs
  - $HOME/appl/bioinfo-c/libs/common-libs

Getting the source code
Configuring the working directory

- Initialize the submodules pointing to libraries
  - cd $HOME/appl/bioinfo-c/libs/common-libs
  - touch .git/git-daemon-export-ok
  - Repeat for bioinfo-libs
  - cd $HOME/appl/bioinfo-c/hpg-aligner
  - git submodule update --init

Getting the source code
System requirements and library dependencies

- Requires a C compiler, Linux headers and the Scons build-system
- Library dependencies
  - cprops
  - samtools
  - argtable2
  - config
  - Use the *builvars.py* file at the working directory to indicate where the libraries are installed
    - $HOME/appl/bioinfo-c/hpg-aligner/buildvars.py

Getting the source code
File hierarchy

- hpg-aligner folder:
  - bin
  - src
    - build-index
    - dna
    - rna
  - lib
- SConstruct file
- buildsvar.py file

- hpg-aligner/lib folder
  - common-libs
    - commons
    - containers
  - bioinfo-libs
    - bioformats
      - fastq
      - bam
      - ...
    - aligners
      - bwt
      - sw

Getting the source code
Hands-on: using hpg-aligner

- To compile
  - cd ~/hpg-aligner-hands-on/hpg-aligner
  - scons
- To build the BWT index
  - ./hpg-aligner build-index -g ../data/Homo_sapiens.GRCh37.70.dna.chromosome.20.fa -i ../index/ -r 10
- To map
  - ./hpg-aligner dna -i ../index/ -f ../data/test_1.fq -o ../out/ --prefix test_1
  - samtools view ../data/out/test_1_alignments.bam | head

Let's talk about...

- Introduction
- Getting the source code: downloading, compiling and building
- Browsing the source code
  - FastQ and BAM file libraries
    - Hands-on
  - BWT and SW aligment libraries
    - Hands-on
  - Parallelization schema
    - Hands-on

Browsing the source code
Working with FastQ and BAM files: bioinfo-libs/bioformats (I)

- bioinfo-libs/bioformats/fastq
  - fastq_read.[hc]
  - fastq_file.[hc]
  - fastq_stats.[hc]
  - ...
- bioinfo-libs/bioformats/bam
  - alignment.[hc]
  - bam_file.[hc]

Browsing the source code
Working with FastQ and BAM files (and II)

- Hands-on
  - cd ~/hpg-aligner-hands-on/exercises/
  - Code to complete: ex1.c
    - Read the n first reads of a FastQ file, and store them in an array list (fastq_file.h, fastq_read.h, array_list.h)
    - Display for each read in the array list: its ID and nucleotide sequence
    - Calculate and display the statistical parameters for each read (fastq_stats.h)
  - To compile and execute:
    - scons
    - ./ex1 <fastq filename> <number of reads>
    - ./ex1 ../data/test_1.fq 10
  -

Browsing the source code
Working with BWT and SW aligners: bioinfo-libs/aligners (I)

- bioinfo-libs/aligners/bwt
  - bwt.[hc]
  - ...
- bioinfo-libs/bioformats/sw
  - smith_waterman.[hc]
  - sse.[hc]

Browsing the source code
Working with BWT and SW aligners (and II)

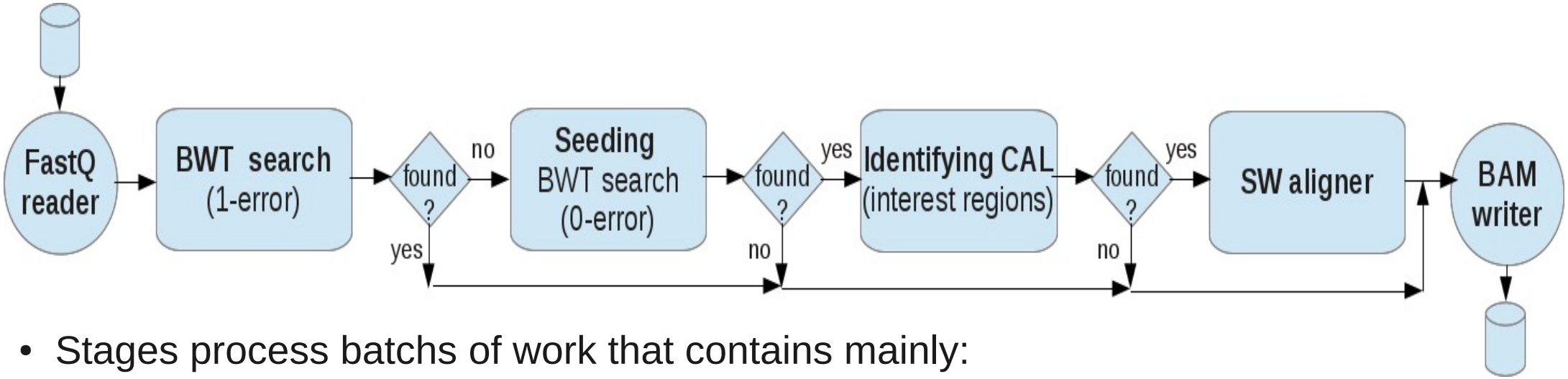
- Hands-on
  - cd ~/hpg-aligner-hands-on/exercises/
  - Code to complete: ex2.c
    - Read the n first reads of a FastQ file and map each read using BWT (fastq_file.h, fastq_read.h, bwt.h)
    - For each read, display its alignments: strand, chromosome, position and cigar code (array_list.h, alignment.h)
  - To compile and execute:
    - scons
    - ./ex2 <fastq filename> <BWT index dirname> <number of reads> <number of errors>
    - ./ex2 ../data/test_1.fq ../index/ 5 1

Browsing the source code
HPG Aligner main function

- The main function (src/hpg_aligner.c) of the application involves:
  - Parsing the command-line options (argtable library, src/options.c)
  - Building the aligner workflow (src/dna/dna_aligner.c, src/rna/rna_aligner.c)
  - Running the workflow aligner in parallel with the specified number of threads (workflow_functions.c, common-libs/commons/workflow_scheduler.c)
    - Reading the FastQ file (bioinfo-libs/bioformats/fastq)
    - Computing the BWT and SW alignments, seeding... (bioinfo-libs/aligners/bwt, bioinfo-libs/aligners/sw)
    - Writing the BAM file (bioinfo-libs/bioformats/bam)

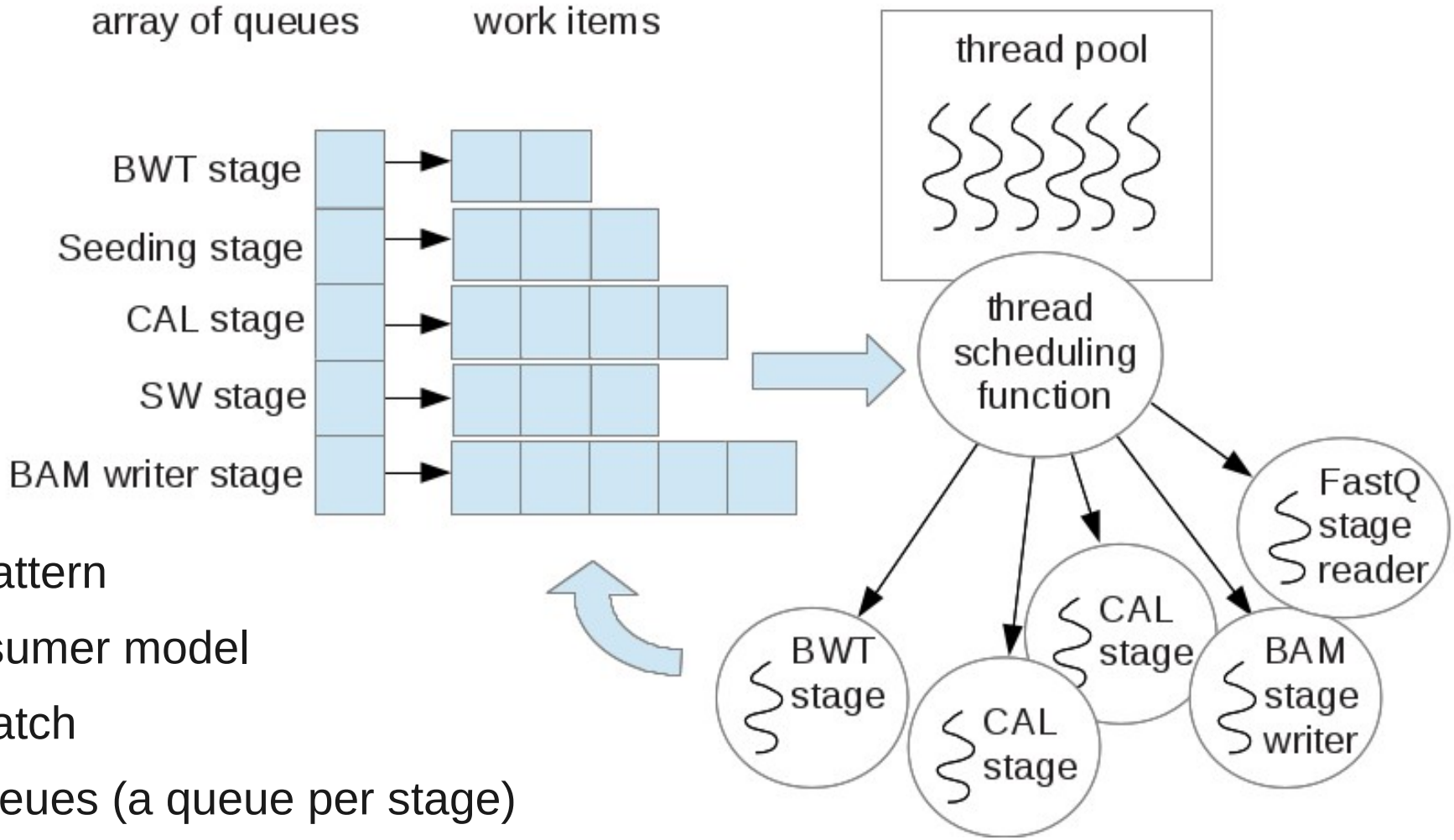
Browsing the source code
Parallelization schema: workflow

- HPG Aligner workflow



- Stages process batchs of work that contains mainly:
  - A list of FASTQ reads and
  - A *target array* with the indices of the reads to be processed (the processing depends on the stage)
- A stage 1) takes a batch, 2) processes it, 3) updates the *target array* and 4) passes the batch to the next stage

Browsing the source code
Parallelization schema: implementation



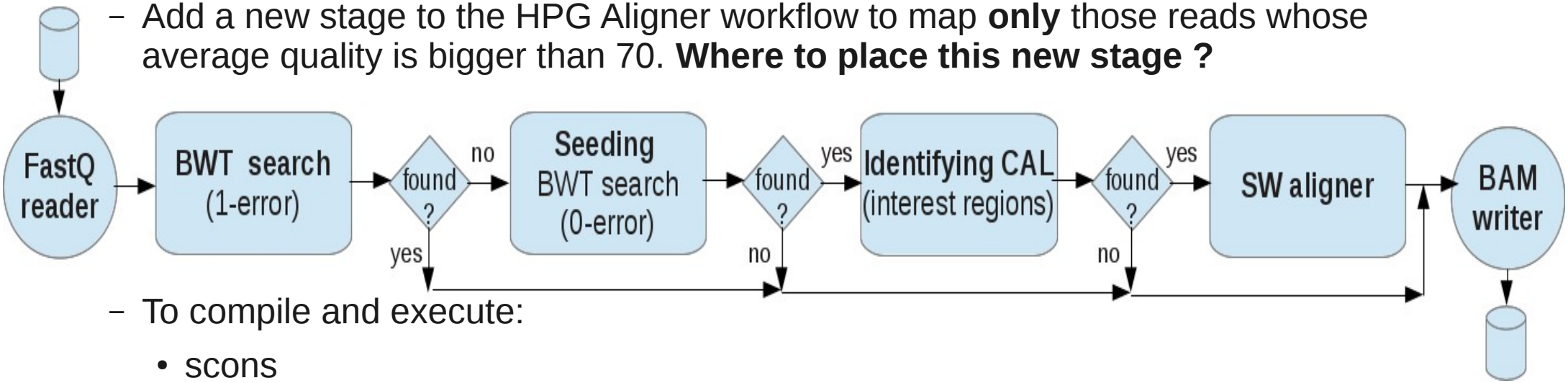
- Thread pool pattern
- Producer-consumer model
- Work item = batch
- Work items queues (a queue per stage)

Browsing the source code
Parallelization schema: source code

- Source code
  - common-libs/commons
    - workflow_scheduler.[hc]
  - hpg-aligner
    - dna/dna_aligner.[hc]
    - rna/rna_aligner.[hc]
    - workflow_functions.[hc]

Browsing the source code
Parallelization schema: hands-on

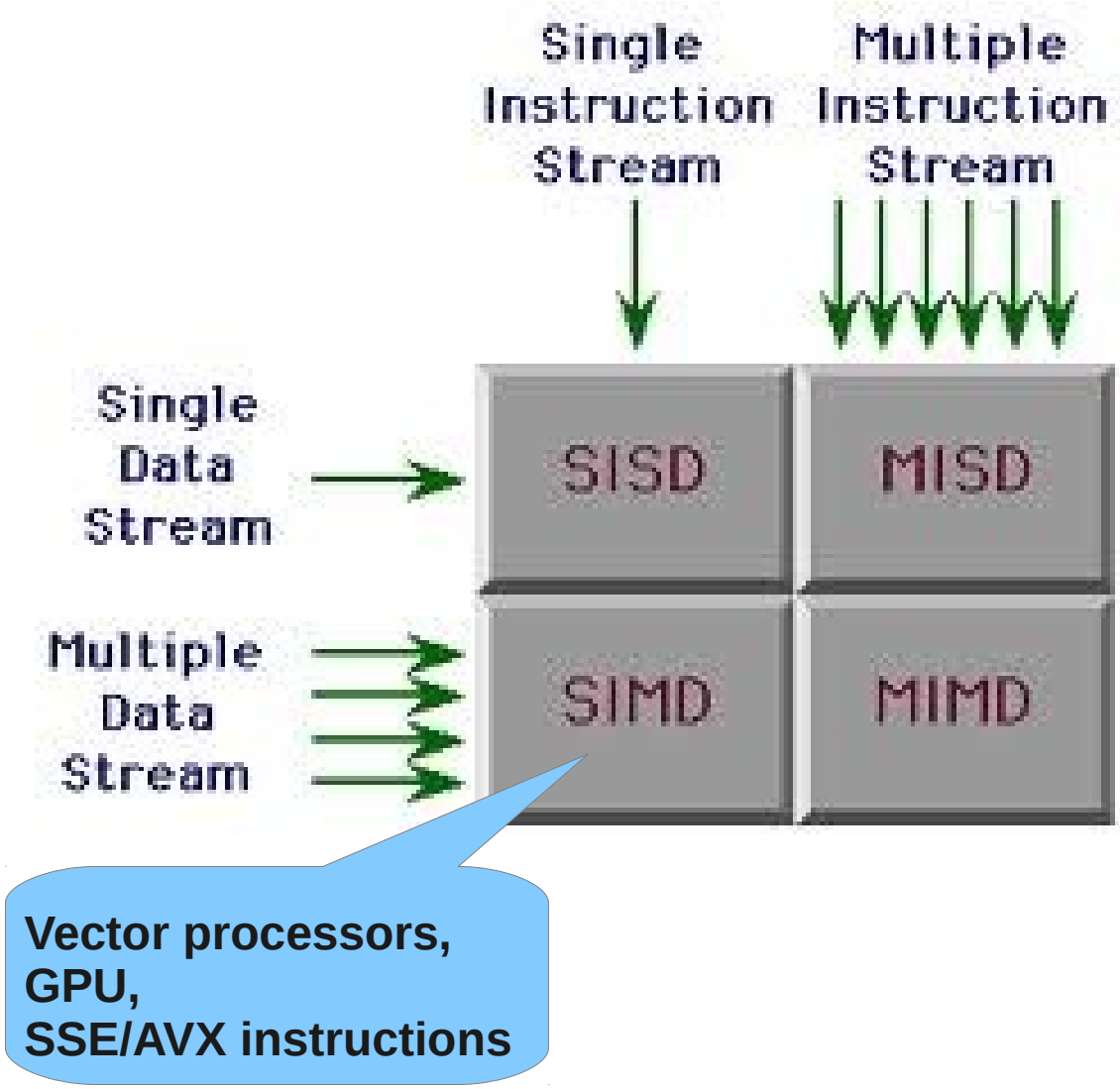
- Hands-on
  - cd ~/hpg-aligner-hands-on/hpg-aligner
  - Add a new stage to the HPG Aligner workflow to map **only** those reads whose average quality is bigger than 70. **Where to place this new stage ?**



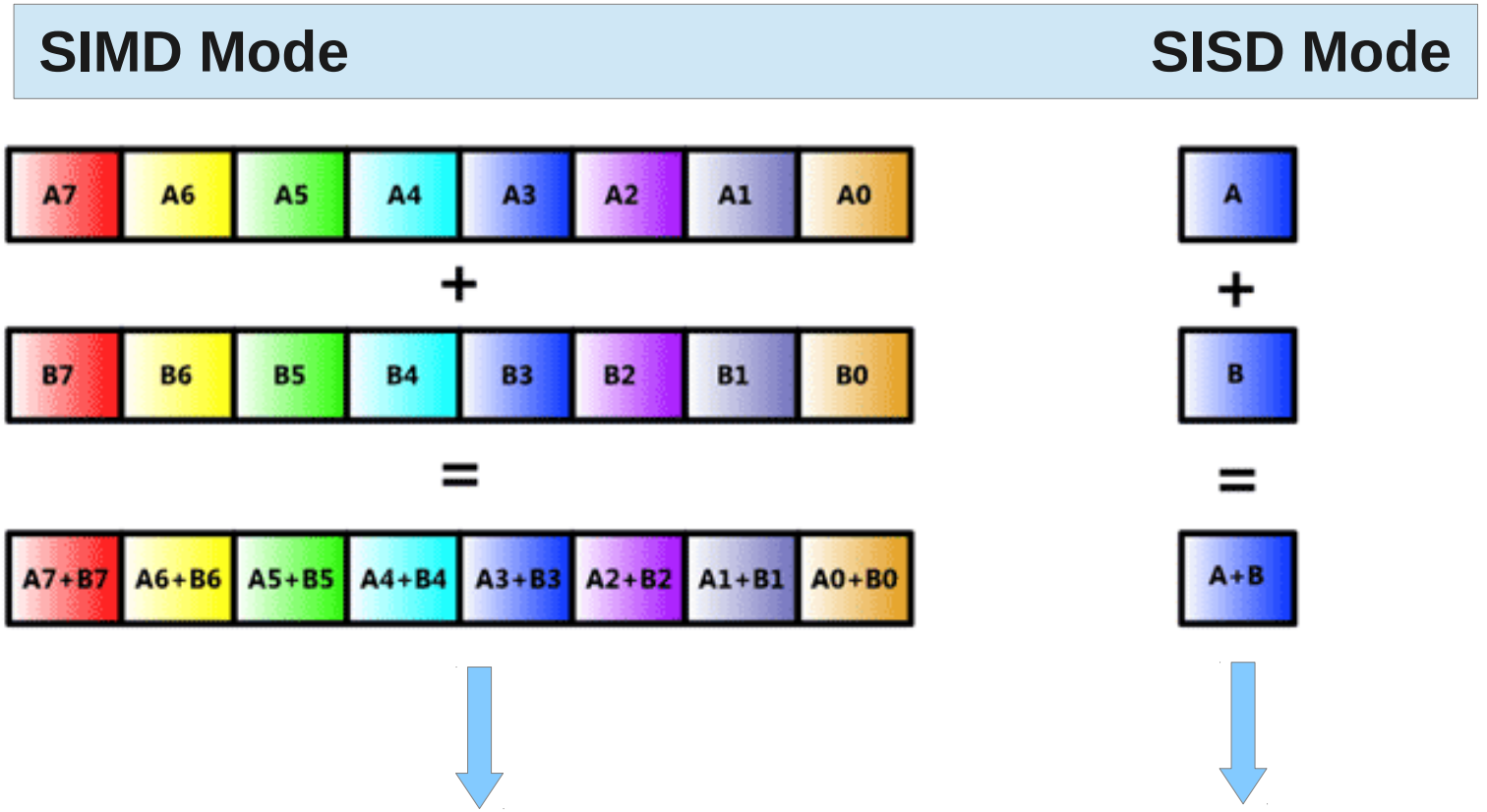
  - To compile and execute:
    - scons
    - ./hpg-aligner -f <fastq filename> -i <BWT index dirname> -o <output dirname> --prefix <prefix name>
    - ./hpg-aligner -f ../data/test_1.fq -i ../index/ -o ../out –prefix test_1

THANKS !

Flynn's taxonomy, 1966

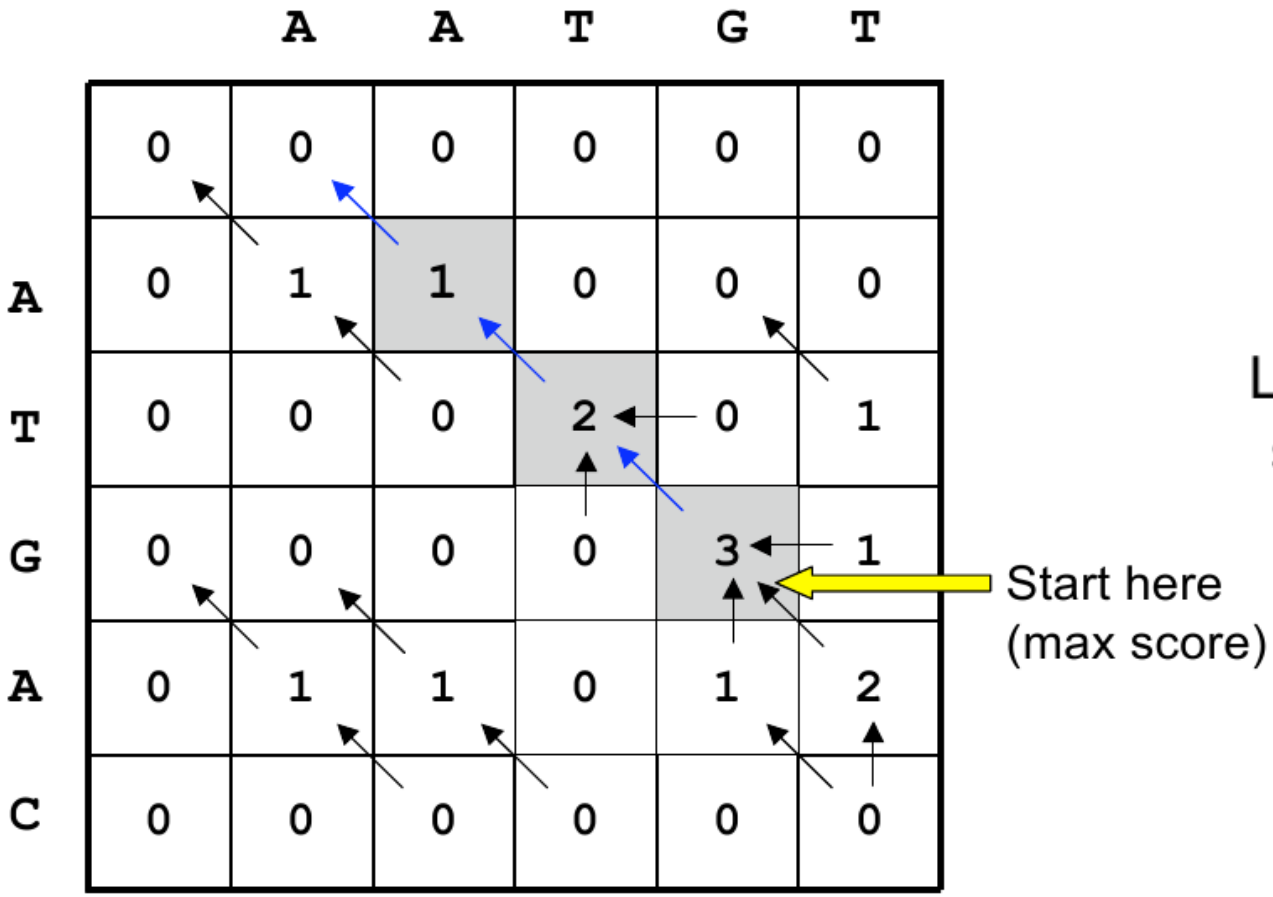


SIMD vs SISD


SIMD Mode
SISD Mode
A7
A6
A5
A4
A3
A2
A1
A0
+
B7
B6
B5
B4
B3
B2
B1
B0
=
A7+B7
A6+B6
A5+B5
A4+B4
A3+B3
A2+B2
A1+B1
A0+B0
A
+
B
=
A+B
1 instruction
8 arithmetic ops.
1 instruction
1 arithmetic op.

Smith-Waterman: alignment score matrix

- Smith-Waterman traceback begins with the maximum score element until reach an element with zero score



Local Alignment shown in blue:

AATGT

ATGAC