Importance of Snakemake Workflows for Admins and HPC-Users

Providing DataAnalytics Service - v1.3.1

Dr. Christian Meesters, HPC Group Mainz
NHR SouthWest
The "Snakemake Teaching Alliance"

14. Januar 2025

Why use Workflow Managers?

1. Why Workflow Managers?
2. About Snakemake
3. Software Environment
4. Getting to Work
5. Workflow Parameterization - for HPC users

from Ewa Bres & Christian Bittner

What is this about?

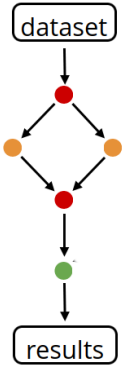
Questions

- Let them learn the batch system! Really?
- What is the benefit of a workflow system for admins?
- What distinguishes a workflow system from a “pipeline”?

Objectives

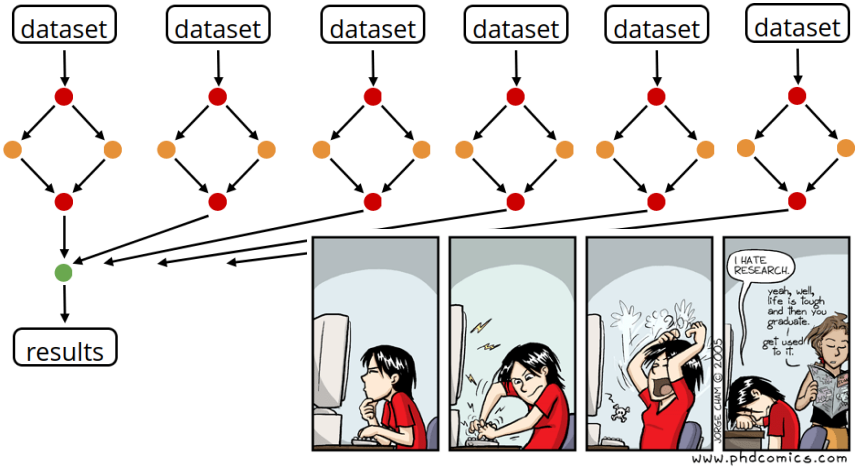
1. Introducing workflow engines (particularly **Snakemake**)!

Data Analysis



Idea from the official **Snakemake** course (with permission), image from PhD comics.

Data Analysis

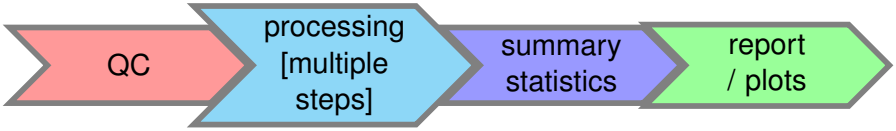


Idea from the official **Snakemake** course (with permission), image from PhD comics.

Let them learn SLURM!

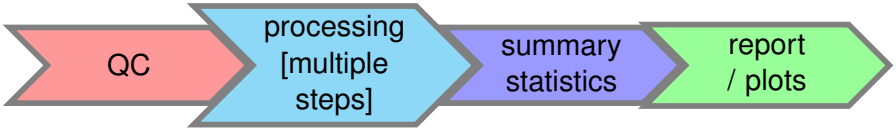
Constituents of a Data Analysis Workflow . . .

. . . (almost) regardless of research topic!



Constituents of a Data Analysis Workflow . . .

. . . (almost) regardless of research topic!



Hint

Not all steps are "HPC-worthy"! E.g. plotting / visualization, moving data, etc.

Meet: The DAG!

Documentation

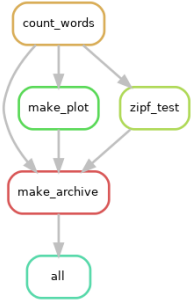
Every data analysis workflow can be expressed as a **Directed Acyclic Graph - a DAG.**

Meet: The DAG!

Documentation

Every data analysis workflow can be expressed as a **Directed Acyclic Graph - a DAG.**

The rulegraph:

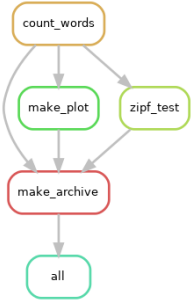


Meet: The DAG!

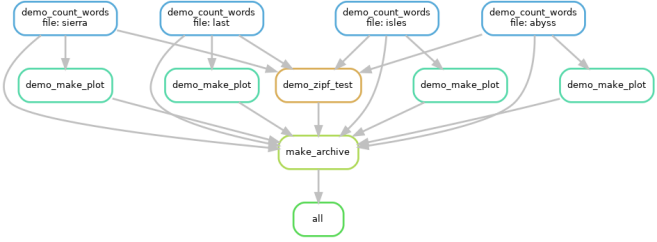
Documentation

Every data analysis workflow can be expressed as a **Directed Acyclic Graph - a DAG.**

The rulegraph:



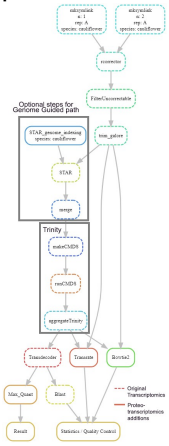
The full DAG - for every sample:



Interlude: An Example

Background

This example is one coded in Bash with SLURM dependencies *and* **Snakemake** . This is its DAG:



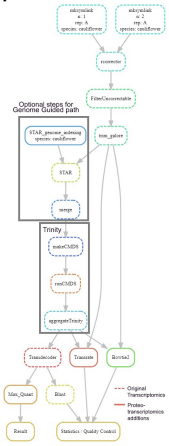
Question

What does it take to code this using Bash and SLURM commands, only?

Interlude: An Example

Background

This example is one coded in Bash with SLURM dependencies *and* **Snakemake** . This is its DAG:



Question

What does it take to code this using Bash and SLURM commands, only?

Warning

Fasten your seat-belts!

Asynchronous Execution using the Batch System

An Proteomics-Workflows will serve as an example (Bash Files + wildly cloned internet stuff):

```
#!/bin/bash

#SBATCH -N 1
##SBATCH -c 1
#SBATCH -A jgu-multiomics
#SBATCH -p nodeshort                    # may consider running on a bigmem node for large dataset
##SBATCH -e fastqc_%A.err               # File to which STDERR will be written
#SBATCH -o fastqc_%A.out                # File to which STDOUT will be written
#SBATCH -J fastqc_%A                    # Job name
#SBATCH --mem-per-cpu=8G
##SBATCH --mem=50G                      # Memory requested (mega default units)
##SBATCH --ramdisk=150G
#SBATCH --time=05:00:00                 # Runtime in D-HH:MM:SS

module purge
set -x

REFDIR=$1
SPECIES=$2
REP=$3
OUTPUT_FOLDER=fastqc_results/${SPECIES}/rep_${REP}
mkdir -p ${OUTPUT_FOLDER}



for i in ${REFDIR}/Sample_imb_butter_*_${SPECIES}_${REP}_*.fastq.gz
do
    fastqc -o ./${OUTPUT_FOLDER}/ $i
done
```

Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -c 32 #64
#SBATCH -A m2_imb-pga
#SBATCH -p bigmem #parallel
#SBATCH -C skylake
#SBATCH -e correctr_%A.err            # File to which STDERR will be written
#SBATCH -o correctr_%A.out            # File to which STDOUT will be written
#SBATCH -J READcorrect_%A              # Job name
#SBATCH --mem=354000 #350G             # 50G originally Memory requested (mega default units)
##SBATCH --ramdisk=350G  # 150G originally
#SBATCH --time=05:00:00               # Runtime in D-HH:MM:SS

# LOAD RELEVANT MODULES FOR STEP 2
module purge
#module load bio/Rcorrector/1.0.4-foss-2019a
module load bio/Jellyfish/2.2.6
module load lang/Perl/5.26.1-foss-2017a
module load lang/Python/2.7.13-foss-2017a
set -x

# RESERVE RAMDISK SPACE
JOBDIR=/localscratch/$SLURM_JOB_ID
RAMDISK=$JOBDIR
mkdir -p $RAMDISK

#SET VARIABLES
#REFDIR=/gpfs/fs2/project/jgu-multiomics/Michal/EvoReg/RNAseq_rawdata/imb_butter_2014_04_EvoReg_RNASeq
#REFDIR=$1
#SPECIES=$2
#REP=$3
OUTPUT_FOLDER=trimmed_reads/${SPECIES}/rep_${REP}
mkdir -p ${OUTPUT_FOLDER}


#COPY FASTQ.GZ FILES TO RAMDISK
file_list=( $(find $REF_DIR -name ${REP}*.gz ) )
for fname in "${file_list[@]}"; do
    #cp -rL ${fname} ${RAMDISK}/.
    STEM=$(basename "${fname}" .gz)
    gunzip -c ${fname} > ${RAMDISK}/${STEM}
done
#R1_file=( $( find ${RAMDISK} -regextype sed -regex '.*/*[\-\_\.]\(R1\|read1\|Read1\|READ1\)[\-\_\.]*\(fastq\|fq\)' ) )
#R1_file=( $( find ${RAMDISK} -regex '.*/*[\-\_\.]\(R1\|read1\|Read1\|READ1\)[\-\_\.]*\(fastq\|fq\)' ) )
R1_file=( $( find ${RAMDISK} -type f | egrep ".*[._-](read1|Read1|r1|R1)*[._-](fq|fastq)" ) )
echo $R1_file
#R2_file=( $( find ${RAMDISK} -regextype sed -regex '.*/*[\-\_\.]\(R2\|read2\|Read2\|READ2\)[\-\_\.]*\(fastq\|fq\)' ) )
R2_file=( $( find ${RAMDISK} -type f | egrep ".*[._-](read2|Read2|r1|R2)*[._-](fq|fastq)" ) )
#R2_file=( $( find ${RAMDISK} -regex '.*/*[\-\_\.]\(R2\|read2\|Read2\|READ2\)[\-\_\.]*\(fastq\|fq\)' ) )
echo $R2_file

# FIRST STEP: REMOVE K-MERS
${RAMDISK}/*[-_.]{R1|read1|Read1|READ1}[-_.]*fastq ${RAMDISK}/*[-_.]{R2|read2|Read2|READ2}[-_.]*fastq

perl programs_ProTrans/rcorrector/run_rcorrector.pl -od ${RAMDISK}/ -t $SLURM_CPUS_PER_TASK -p ${R1_file} ${R2_file}
wait
module purge
module load lang/Python/2.7.13-foss-2017a
```

Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -c 12
#SBATCH -A m2_imb-pga
#SBATCH -p smp                          # may consider running on a bigmem node for large dataset
#SBATCH -e build_genome_%A.err          # File to which STDERR will be written
#SBATCH -o build_genome_%A.out          # File to which STDOUT will be written
#SBATCH -J build_genome_%A              # Job name
#SBATCH --mem=100G                      # Memory requested (mega default units)
#SBATCH --time=00:30:00                 # Runtime in D-HH:MM:SS
#SBATCH --mail-type=ALL                 # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=m.levin@imb.de      # Email to send notifications to
#SBATCH --ramdisk=100G

module load bio/STAR/2.5.3a-foss-2017a

STAR_FOLDER=${GENOME_FILE%.*.*}_STAR_index

mkdir -p $STAR_FOLDER


JOBDIR=/localscratch/$SLURM_JOB_ID
RAMDISK=$JOBDIR/ramdisk

#gunzip $GENOME_FILE
zcat $GENOME_FILE > $RAMDISK/temp.fa
#GENOME_FILE_uz="$(ls $GENOME_FILE/*.fa)"

STAR --runMode genomeGenerate --outTmpDir $RAMDISK/tmp --genomeDir $STAR_FOLDER --genomeFastaFiles $RAMDISK/temp.fa --runThreadN $SLURM_CPUS_PER_TASK --limitGenomeGenerateRAM 60000000000 --outFileNamePrefix $STAR_FOLDER
```

Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -p parallel
#SBATCH -C skylake
#SBATCH -N 1
#SBATCH -c 16
#SBATCH -A m2_imb-pga
##SBATCH -e MAP4Trinity_%A.err          # File to which STDERR will be written
#SBATCH -o MAP4Trinity_%A.out          # File to which STDOUT will be written
#SBATCH -J MAP4Trinity_%A              # Job name
#SBATCH --mem=177000 #350G             # Memory requested (mega default units)
#SBATCH --ramdisk=450G
#SBATCH --time=03:00:00  #05:00:00             # Runtime in D-HH:MM:SS
#SBATCH --mail-type=ALL                # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=m.levin@imb.de # Email to send notifications to

module load bio/STAR/2.5.3a-foss-2017a

STAR_FOLDER=${STAR_FOLDER_LOC}
OUTPUT_FOLDER=${WORK_DIR}/mapped/${SPECIES}/

mkdir -p $OUTPUT_FOLDER
mkdir -p ${OUTPUT_FOLDER}/log_files
set -x

JOBDIR=/localscratch/$SLURM_JOB_ID
RAMDISK=$JOBDIR/ramdisk
mkdir -p $RAMDISK

REFDIR=trimmed_reads/${SPECIES}/rep_${REP}

cp $REFDIR/*R1.cor_val_1.fq $RAMDISK/R1.fq
#gzip $RAMDISK/'$species'_R1.fq'
cp $REFDIR/*R2.cor_val_2.fq $RAMDISK/R2.fq
#gzip $RAMDISK/'$species'_R2.fq'

wait

STAR --runThreadN $SLURM_CPUS_PER_TASK --genomeDir $STAR_FOLDER --outSAMtype BAM SortedByCoordinate --outFileNamePrefix $RAMDISK/${SPECIES}_rep_${REP}_ --alignIntronMax $MAX_INTRON_SIZE --limitBAMsortRAM 6000463617 --outFilterMultimapNmax 1 --outFilterScoreMin 2


wait

cp $RAMDISK/${SPECIES}_rep_${REP}_Aligned.sortedByCoord.out.bam $OUTPUT_FOLDER
cp $RAMDISK/${SPECIES}_rep_${REP}_Log* $OUTPUT_FOLDER/log_files
cp $RAMDISK/${SPECIES}_rep_${REP}_SJ.out.tab $OUTPUT_FOLDER/log_files
```

Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -p bigmem
#SBATCH -C skylake
#SBATCH -A m2_imb-pga
#SBATCH -c 64
#SBATCH --ramdisk=400G
#SBATCH -e trinity_normalization_%A.err              # File to which STDERR will be written
#SBATCH -o trinity_normalization_%A.out              # File to which STDOUT will be written
#SBATCH -J trinity_normalization_%A                  # Job name
#SBATCH --mem=354000 #350G                  # Memory requested (mega default units)
##SBATCH --mem=100G                        # Memory requested (mega default units)
#SBATCH --time=1-20:00:00 #07:00:00 #1-20:00:00              # Runtime in D-HH:MM:SS
#SBATCH --mail-type=ALL                     # Type of email notification - BEGIN,END,FAIL,ALL
#SBATCH --mail-user=m.levin@imb.de # Email to send notifications to

#module load bio/SAMtools/1.9-foss-2018a
module purge
module load bio/Trinity/2.8.4-foss-2018a
#module load lang/Python/3.6.4-foss-2017a
#module load bio/Jellyfish/2.2.6-foss-2017a
#module load bio/SAMtools/1.5-foss-2017a
#module load bio/Bowtie2/2.3.2-foss-2017a
module load bio/Salmon/0.13.1-foss-2018a

OUTPUT_FOLDER=trinity_GF_split_allbamcomb/$SPECIES
mkdir -p $OUTPUT_FOLDER/trinity

### START THE TRINITY ASSEMBLY ONLY IF THE RESPECTIVE trinity_GG.cmds FILE DOES NOT YET EXIST
if [ ! -f $OUTPUT_FOLDER/trinity/recursive_trinity.cmds ]; then
    echo "recursive_trinity.cmds for species "$SPECIES" does not yet exist. Starting job to create"

        JOBDIR=/localscratch/$SLURM_JOB_ID
        RAMDISK=$JOBDIR/ramdisk
        #RAMDISK=$JOBDIR
        mkdir -p $RAMDISK
        mkdir -p $RAMDISK/trinity

        REFDIR=${WORK_DIR}/trimmed_reads/${SPECIES}

        cat $(find ${REFDIR}/ -type f -name '*.R1.*')  > ${RAMDISK}/R1.fq
        cat $(find ${REFDIR}/ -type f -name '*.R2.*')  > ${RAMDISK}/R2.fq
        wait

        #Trinity --genome_guided_bam $RAMDISK/*.bam --genome_guided_max_intron $INTRON_MAX --max_memory $((SLURM_MEM_PER_NODE / 1024))G --CPU $SLURM_CPUS_PER_TASK --SS_lib_type FR --output $OUTPUT_FOLDER/trinity

        #Trinity --genome_guided_bam $RAMDISK/*.bam --genome_guided_max_intron $INTRON_MAX --max_memory $((SLURM_SPANK_JOB_RAMDISK / 1024))G --CPU $SLURM_CPUS_PER_TASK --SS_lib_type FR --full_cleanup --verbose --output $RAMDISK/trinity

        Trinity --seqType fq --SS_lib_type RF --max_memory 50G --min_kmer_cov $min_kmer_cov --CPU $SLURM_CPUS_PER_TASK --left $RAMDISK/R1.fq --right $RAMDISK/R2.fq --output $OUTPUT_FOLDER/trinity --full_cleanup --verbose --no_distributed_trinity_exec



        # --max_memory $((SLURM_MEM_PER_NODE / 1024))G
else
        echo "Trinity.fasta for species "$SPECIES" already exists - nothing to do here!"
fi
```

Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -A m2_imb-pga
#SBATCH -n 1
#SBATCH --job-name=trinity_arrayJob
##SBATCH --output=trinity_arrayJob_%A_%a.out # redirecting stdout
##SBATCH --error=trinity_arrayJob_%A_%a.err  # redirecting stderr
##SBATCH --array=1-712
##SBATCH --array=5
#SBATCH --time=15:00:00  #15:00:00
#SBATCH --partition=smp #parallel
#SBATCH --ntasks=1              # number of tasks per array job
#SBATCH --mem-per-cpu=20000 #4000  #20000
##SBATCH --cpus-per-task=1
#SBATCH --mail-type=ALL                 # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=m.levin@imb.de # Email to send notifications to
##SBATCH --mem-per-cpu=8000
module load bio/Trinity/2.8.4-foss-2018a
module load bio/Salmon/0.13.1-foss-2018a

CMD_FILE=recursive_trinity.cmds
CMD_NO=$(wc -l recursive_trinity.cmds | awk '{print $1}')
echo "Number of commands in recursive.cmds file is:"
echo $CMD_NO
echo "This is array No.:"
echo $SLURM_ARRAY_TASK_ID
((CHUNKSIZE=(CMD_NO/SLURM_ARRAY_TASK_COUNT)+1))
((TOTALREST=500-CHUNKSIZE))
((REDUCED_JOBS=TOTALREST-CMD_NO))
((NUM_JOBS=500-((REDUCED_JOBS/CHUNKSIZE)+1)))
arrayID=$SLURM_ARRAY_TASK_ID

if [ "$arrayID" -lt "$NUM_JOBS" ]; then
        indexes=`seq $(((((arrayID - 1) * CHUNKSIZE) + 1))) $(((((arrayID - 1) * CHUNKSIZE) + CHUNKSIZE)))`
        echo "Processing command lines: "
        echo $(((((arrayID - 1) * CHUNKSIZE) + 1)))
        echo "to"
        echo $(((((arrayID - 1) * CHUNKSIZE) + CHUNKSIZE)))
        for i in $indexes ; do
                echo "################################"
                echo "Processing cmds line:"
                echo $i
                echo "################################"
                bash -c "$(head -n $i ${CMD_FILE} | tail -n 1)"
        done
elif [ "$arrayID" -eq "$NUM_JOBS" ]; then
        indexes=`seq $(((((arrayID - 1) * CHUNKSIZE) + 1))) $CMD_NO`
        echo "This is the last array and processes the remaining commands till end of cmd file"
        echo "Processing command lines: "
        echo $(((((arrayID - 1) * CHUNKSIZE) + 1)))
        echo "to"
        echo $CMD_NO
    for i in $indexes ; do
                echo "################################"
                echo "Processing cmds line:"
                echo $i
                echo "################################"
                bash -c "$(head -n $i ${CMD_FILE} | tail -n 1)"
        done
```

Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -c 1
#SBATCH -A m2_imb-pga
#SBATCH -p smp                    # may consider running on a bigmem node for large dataset
#SBATCH -e ph_%A.err              # File to which STDERR will be written
#SBATCH -o ph_%A.out              # File to which STDOUT will be written
#SBATCH -J ph_%A                  # Job name
#SBATCH --mem=50M     #500M       # Memory requested (mega default units)
#SBATCH --time=03:00:00           # Runtime in D-HH:MM:SS
#SBATCH --mail-type=ALL           # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=m.levin@imb.de # Email to send notifications to

module load bio/Trinity/2.8.4-foss-2018a

REF_DIR=${WORK_DIR}/trinity_GF_split_allbamcomb/${SPECIES}/trinity

#find ${REF_DIR}/read_partitions/   -name '*inity.fasta' | /gpfs/fs1/cluster/easybuild/nehalem/software/bio/Trinity/2.8.4-foss-2017a/trinityrnaseq-Trinity-v2.8.4/util/support_scripts/partitioned_trinity_aggregator.pl TRINITY_DN > ${REF_DIR}/Trinity-GF.fasta

find ${REF_DIR}/read_partitions/   -name '*inity.fasta' | /cluster/easybuild/broadwell/software/bio/Trinity/2.8.4-foss-2018a/trinityrnaseq-Trinity-v2.8.4/util/support_scripts/partitioned_trinity_aggregator.pl --token_prefix TRINITY_DN --output_prefix ${REF_DIR}/
```

Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -c 1
#SBATCH -A m2_imb-pga
#SBATCH -p smp
##SBATCH -e ph_%A.err              # File to which STDERR will be written
##SBATCH -o ph_%A.out             # File to which STDOUT will be written
##SBATCH -J ph_%A                 # Job name
##SBATCH --mem=1K                 # Memory requested (mega default units)
#SBATCH --time=00:30:00           # Runtime in D-HH:MM:SS
##SBATCH --mail-type=ALL          # Type of email notification- BEGIN,END,FAIL,ALL
##SBATCH --mail-user=m.levin@imb.de # Email to send notifications to

echo 'Moving final file!'

REF_DIR=${WORK_DIR}/trinity_GF_split_allbamcomb/${SPECIES}/trinity
FINAL_DIR=${WORK_DIR}/Trinity_GF_assemblies

cp ${REF_DIR}/Trinity-GF.fasta  ${FINAL_DIR}/Trinity-GF_${SPECIES}_mincov${min_kmer_cov}.fasta
wait
mkdir -p ${WORK_DIR}/trinity_GF_split_allbamcomb/${SPECIES}/blanktemp
rsync -a --delete ${WORK_DIR}/trinity_GF_split_allbamcomb/${SPECIES}/blanktemp/ ${WORK_DIR}/trinity_GF_split_allbamcomb/${SPECIES}/
wait
```

Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -p bigmem
#SBATCH -C skylake
#SBATCH -A m2_imb-pga
#SBATCH -c 64
#SBATCH --ramdisk=400G
#SBATCH -e trinity_normalization_%A.err              # File to which STDERR will be written
#SBATCH -o trinity_normalization_%A.out              # File to which STDOUT will be written
#SBATCH -J trinity_normalization_%A                  # Job name
#SBATCH --mem=354000 #500G                  # Memory requested (mega default units)
#SBATCH --time=1-20:00:00 #10:00:00 #1-20:00:00              # Runtime in D-HH:MM:SS
#SBATCH --mail-type=ALL                  # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=m.levin@imb.de # Email to send notifications to

set -x

#module load bio/SAMtools/1.9-foss-2018a
module purge
module load bio/Trinity/2.8.4-foss-2018a
#module load lang/Python/3.6.4-foss-2017a
#module load bio/Jellyfish/2.2.6-foss-2017a
#module load bio/SAMtools/1.5-foss-2017a
#module load bio/Bowtie2/2.3.2-foss-2017a
module load bio/Salmon/0.13.1-foss-2018a

OUTPUT_FOLDER=trinity_GG_split_allbamcomb/$SPECIES
mkdir -p $OUTPUT_FOLDER/trinity

### START THE TRINITY ASSEMBLY ONLY IF THE RESPECTIVE trinity_GG.cmds FILE DOES NOT YET EXIST
if [ ! -f $OUTPUT_FOLDER/trinity/trinity_GG.cmds ]; then
    echo "trinity_GG.cmds for species "$SPECIES" does not yet exist. Starting job to create"

    JOBDIR=/localscratch/$SLURM_JOB_ID
    RAMDISK=$JOBDIR/ramdisk
    mkdir -p $RAMDISK
    mkdir -p $RAMDISK/trinity

    REFDIR=${WORK_DIR}/mapped/${SPECIES}
    #REL_FILE=zebrafinch_rep_C_Aligned.sortedByCoord.out.bam
    ### use the biggest bam file available for the species
    FILES_2_COMB=$(find ${REFDIR}/ -name '*_Aligned.sortedByCoord.out.bam' | xargs echo)
    #REL_FILE=$(du -cks $REFDIR/*.bam | sort -n -r | awk '{print $2}' | head -n2 | tail -n1 | cut -d'/' -f11)
    samtools merge $RAMDISK/merged.bam $FILES_2_COMB

    wait

    Trinity --genome_guided_bam $RAMDISK/merged.bam --genome_guided_max_intron $MAX_INTRON_SIZE --genome_guided_min_coverage $genome_guided_min_coverage --max_memory $((SLURM_MEM_PER_NODE / 1024))G --CPU $SLURM_CPUS_PER_TASK --SS_lib_type FR --full_cleanup --ver


    else
        echo "Trinity-GG.fasta for species "$SPECIES" already exists - nothing to do here!"
fi
```

Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -A m2_imb-pga
#SBATCH -n 1
#SBATCH --job-name=trinity_arrayJob
##SBATCH --output=trinity_arrayJob_%A_%a.out # redirecting stdout
##SBATCH --error=trinity_arrayJob_%A_%a.err  # redirecting stderr
#SBATCH --array=1-712
##SBATCH --array=5
#SBATCH --time=15:00:00  #15:00:00
#SBATCH --partition=smp
#SBATCH --ntasks=1              # number of tasks per array job
#SBATCH --mem-per-cpu=20000 #4000 #20000
##SBATCH --cpus-per-task=1
#SBATCH --mail-type=ALL                 #
Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=m.levin@imb.de # Email to send notifications to

module load bio/Trinity/2.8.4-foss-2018a
module load bio/Salmon/0.13.1-foss-2018a

CMD_FILE=trinity_GG.cmds
CMD_NO=$(wc -l trinity_GG.cmds | awk '{print $1}')

echo "Number of commands in recursive.cmds file is:"
echo $CMD_NO
echo "This is array No.:"
echo $SLURM_ARRAY_TASK_ID
((CHUNKSIZE=(CMD_NO/SLURM_ARRAY_TASK_COUNT)+1))
((TOTALREST=500-CHUNKSIZE))
((REDUCED_JOBS=TOTALREST-CMD_NO))
((NUM_JOBS=500-((REDUCED_JOBS/CHUNKSIZE)+1)))
arrayID=$SLURM_ARRAY_TASK_ID

if [ "$arrayID" -lt "$NUM_JOBS" ]; then
        indexes=`seq $(((((arrayID - 1) * CHUNKSIZE) + 1))) $(((((arrayID - 1) * CHUNKSIZE) + CHUNKSIZE)))`
        echo "Processing command lines: "
        echo $(((((arrayID - 1) * CHUNKSIZE) + 1)))
        echo "to"
        echo $(((((arrayID - 1) * CHUNKSIZE) + CHUNKSIZE)))
        for i in $indexes ; do
                echo "################################"
                echo "Processing cmds line:"
                echo $i
                echo "################################"
                bash -c "$(head -n $i ${CMD_FILE) | tail -n 1)"
        done
elif [ "$arrayID" -eq "$NUM_JOBS" ]; then
        indexes=`seq $(((((arrayID - 1) * CHUNKSIZE) + 1))) $CMD_NO`
        echo "This is the last array and processes the remaining commands till end of cmd file"
        echo "Processing command lines: "
        echo $(((((arrayID - 1) * CHUNKSIZE) + 1)))
        echo "to"
        echo $CMD_NO
        for i in $indexes ; do
                echo "################################"
                echo "Processing cmds line:"
                echo $i
                echo "################################"
```

Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -c 1
#SBATCH -A m2_imb-pga
#SBATCH -p smp
#SBATCH -e ph_%A.err              # File to which STDERR will be written
#SBATCH -o ph_%A.out              # File to which STDOUT will be written
#SBATCH -J ph_%A                  # Job name
#SBATCH --mem=100G                  # Memory requested (mega default units)
#SBATCH --time=08:00:00               # Runtime in D-HH:MM:SS
#SBATCH --mail-type=ALL               # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=m.levin@imb.de # Email to send notifications to

module load bio/Trinity/2.8.4-foss-2018a

REF_DIR=${WORK_DIR}/trinity_GG_split_allbamcomb/${SPECIES}/trinity

find ${REF_DIR}/Dir_* -name '*inity.fasta' | /cluster/easybuild/broadwell/software/bio/Trinity/2.8.4-foss-2018a/trinityrnaseq-Trinity-v2.8.4/util/support_scripts/GG_partitioned_trinity_aggregator.pl TRINITY_GG > ${REF_DIR}/Trinity-GG.fasta
```

Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -c 1
#SBATCH -A m2_imb-pga
#SBATCH -p smp
##SBATCH -e ph_%A.err              # File to which STDERR will be written
##SBATCH -o ph_%A.out             # File to which STDOUT will be written
##SBATCH -J ph_%A                 # Job name
##SBATCH --mem=1K                 # Memory requested (mega default units)
#SBATCH --time=00:30:00             # Runtime in D-HH:MM:SS
##SBATCH --mail-type=ALL           # Type of email notification- BEGIN,END,FAIL,ALL
##SBATCH --mail-user=m.levin@imb.de # Email to send notifications to

echo 'Moving final file!'

REF_DIR=${WORK_DIR}/trinity_GG_split_allbamcomb/${SPECIES}/trinity
FINAL_DIR=${WORK_DIR}/Trinity_GG_assemblies

mv ${REF_DIR}/Trinity-${ASSEMBLY_MODE}.fasta  ${FINAL_DIR}/Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${genome_guided_min_coverage}.fasta
wait
mkdir -p ${WORK_DIR}/trinity_${ASSEMBLY_MODE}_split_allbamcomb/${SPECIES}/blanktemp
rsync -a --delete ${WORK_DIR}/trinity_${ASSEMBLY_MODE}_split_allbamcomb/${SPECIES}/blanktemp/ ${WORK_DIR}/trinity_${ASSEMBLY_MODE}_split_allbamcomb/${SPECIES}/
wait
#rm -r $REF_DIR
```

Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -c 1
#SBATCH -A m2_imb-pga
#SBATCH -p smp
##SBATCH -e ph_%A.err               # File to which STDERR will be written
##SBATCH -o ph_%A.out              # File to which STDOUT will be written
##SBATCH -J ph_%A                  # Job name
##SBATCH --mem=1K                  # Memory requested (mega default units)
##SBATCH --time=00:02:00                  # Runtime in D-HH:MM:SS
##SBATCH --mail-type=ALL                  # Type of email notification- BEGIN,END,FAIL,ALL
##SBATCH --mail-user=m.levin@imb.de # Email to send notifications to

echo 'cleaning up Transdecoder Folder!'

rm *.cmds
rm *.fasta
rm *.sbatch
rm -r Trinity-${ASSEMBLY_MODE}_*.fasta.transdecoder_dir*
```

Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -c 1
#SBATCH -A m2_imb-pga
#SBATCH -p smp
##SBATCH -e trandec_%A.err          # File to which STDERR will be written
#SBATCH -o transdec_%A.out         # File to which STDOUT will be written
#SBATCH -J transdec_%A             # Job name
#SBATCH --mem=64G #50G             # Memory requested (mega default units)
##SBATCH --gres=ramdisk:1G
#SBATCH --time=03:00:00  #01:00:00          # Runtime in D-HH:MM:SS
#SBATCH --mail-type=ALL            # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=m.levin@imb.de # Email to send notifications to

module load bio/TransDecoder/5.5.0-Perl-5.30.0
#module load lang/Perl/5.26.1-foss-2017a
set -x

#JOBDIR=/localscratch/$SLURM_JOB_ID
#RAMDISK=$JOBDIR/ramdisk

#REFDIR=Trinity_GG_assemblies
RESULTS_FOLDER=${WORK_DIR}/Trinity_${ASSEMBLY_MODE}_Transdecoder/$SPECIES
#mkdir -p $RESULTS_FOLDER
#cp ${REFDIR}/Trinity-GG_${SPECIES}.fasta $RESULTS_FOLDER/
#cp $REFDIR/Trinity.fasta $RAMDISK/.
REFDIR=${WORK_DIR}/Trinity_${ASSEMBLY_MODE}_assemblies
cp ${REFDIR}/Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${mincov}.fasta $RESULTS_FOLDER/Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${mincov}_ORFsize${ORF_size}aa.fasta
wait

#~/TransDecoder-TransDecoder-v5.4.0/TransDecoder.LongOrfs -m ${ORF_size} -t Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${mincov}_ORFsize${ORF_size}aa.fasta
TransDecoder.LongOrfs -m ${ORF_size} -t Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${mincov}_ORFsize${ORF_size}aa.fasta
wait

#~/TransDecoder-TransDecoder-v5.4.0/TransDecoder.Predict -t Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${mincov}_ORFsize${ORF_size}aa.fasta
TransDecoder.Predict -t Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${mincov}_ORFsize${ORF_size}aa.fasta
```

Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -c 16
#SBATCH -A m2_imb-pga
#SBATCH -p parallel
#SBATCH -C skylake
##SBATCH -e blastp_%A.err              # File to which STDERR will be written
##SBATCH -o blastp_%A.out              # File to which STDOUT will be written
##SBATCH -J blastp_%A                  # Job name
#SBATCH --mem=50G    #50G               # Memory requested (mega default units)
#SBATCH --ramdisk=50G
#SBATCH --time=03:00:00   #05:00:00                  # Runtime in D-HH:MM:SS
#SBATCH --mail-type=ALL               # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=m.levin@imb.de # Email to send notifications to

module load bio/BLAST+/2.9.0-gompi-2019a
set -x

#SPECIES_LAT=$1
#SPECIES=$2
#ENSEMBL_VS=$3
#ASSEMBLY_MODE=$4
#WORK_DIR=$5
#ORF_size=$6
#ref_cov=$7
#OUTPUT_FOLDER=$8
#PROT_REF_DIR=$9

JOBDIR=/localscratch/$SLURM_JOB_ID
RAMDISK=$JOBDIR/ramdisk
mkdir -p $RAMDISK


TRINITY_FILE=${WORK_DIR}/Trinity_${ASSEMBLY_MODE}_Transdecoder/$SPECIES/Trinity*_mincov${ref_cov}_ORFsize${ORF_size}aa*.pep

DB_FILE=$PROT_REF_DIR
#DB_FILE=${ls ${PROT_REF_DIR}/${SPECIES_LAT}/Ensembl_${ENSEMBL_VS}}
#OUTPUT_FOLDER=/gpfs/fs2/project/jgu-multiomics/Michal/EvoReg/EvoReg_workflow/blastp_results/
#mkdir -p $OUTPUT_FOLDER

STEM=$(basename "${DB_FILE}" .gz)
gunzip -c ${PROT_REF_DIR} > $RAMDISK/"${STEM}"
makeblastdb -in $RAMDISK/"${STEM}" -dbtype prot
cp $TRINITY_FILE $RAMDISK/.
# $REFDIR/Trinity.fasta $RAMDISK/.
#blastp -query Trinity-GG.fasta.transdecoder_reduced.pep  -db $DB_DIR  -max_target_seqs 1 -outfmt 6 -evalue 1e-3 -num_threads 20 > blastp.outfmt6

blastp -query $RAMDISK/*.pep -db $RAMDISK/${STEM} -outfmt 6 -evalue 1e-10 -num_threads 16 > $RAMDISK/Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${ref_cov}_ORFsize${ORF_size}aa.blastp.outfmt6
wait

/cluster/easybuild/broadwell/software/bio/Trinity/2.8.4-foss-2018a/trinityrnaseq-Trinity-v2.8.4/util/analyze_blastPlus_topHit_coverage.pl $RAMDISK/Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${ref_cov}_ORFsize${ORF_size}aa.blastp.outfmt6 $RAMDISK/*.pep $RAMDISK/${
wait

/cluster/easybuild/broadwell/software/bio/Trinity/2.8.4-foss-2018a/trinityrnaseq-Trinity-v2.8.4/util/misc/blast_outfmt6_group_segments.pl $RAMDISK/Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${ref_cov}_ORFsize${ORF_size}aa.blastp.outfmt6 $RAMDISK/*.pep $RAMDISK/${
wait

/cluster/easybuild/broadwell/software/bio/Trinity/2.8.4-foss-2018a/trinityrnaseq-Trinity-v2.8.4/util/misc/blast_outfmt6_group_segments.tophit_coverage.pl $RAMDISK/Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${ref_cov}_ORFsize${ORF_size}aa.blastp.outfmt6_grouped >
wait
```

Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -c 64
#SBATCH -A m2_imb-pga
#SBATCH -p bigmem #parallel
#SBATCH -C skylake
##SBATCH -e blastp_%A.err              # File to which STDERR will be written
##SBATCH -o blastp_%A.out              # File to which STDOUT will be written
##SBATCH -J blastp_%A                  # Job name
#SBATCH --mem=354000                  # Memory requested (mega default units)
#SBATCH --ramdisk=500G    #150G
#SBATCH --time=12:00:00  #1-00:00:00                  # Runtime in D-HH:MM:SS
#SBATCH --mail-type=ALL               # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=m.levin@imb.de # Email to send notifications to

#module load bio/Trinity/2.8.4-foss-2017a
#module load bio/Bowtie2/2.3.4.3-foss-2018a
#module load bio/SAMtools/1.9
module load bio/Bowtie2/2.3.5.1-GCC-8.3.0
module load bio/Trinity/2.11.0-foss-2019b-Python-3.7.4


set -x

#SPECIES=$1
#ASSEMBLY_MODE=$2
#WORK_DIR=$3
#ref_cov=$4
#OUTPUT_FOLDER=$5

JOBDIR=/localscratch/$SLURM_JOB_ID
RAMDISK=$JOBDIR/ramdisk
mkdir -p $RAMDISK

TRINITY_FILE=${WORK_DIR}/Trinity_${ASSEMBLY_MODE}_assemblies/Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${ref_cov}.fasta
#cp $TRINITY_FILE $RAMDISK/ref.fasta
wait
#bowtie2-build ref.fasta ${SPECIES}
cd $RAMDISK
bowtie2-build $TRINITY_FILE ${SPECIES}

cat $(find ${WORK_DIR}/trimmed_reads/${SPECIES} -name '*R1.cor_val_1.fq' | xargs echo) > $RAMDISK/read1_file.fq
wait
cat $(find ${WORK_DIR}/trimmed_reads/${SPECIES} -name '*R2.cor_val_2.fq' | xargs echo) > $RAMDISK/read2_file.fq
wait


#assembly="/home/michal/Documents/BOMO_trinity_denovo_assembly/trinity_assemblies/Trinity.fasta"
#read1_file="/home/michal/Documents/BOMO_trinity_denovo_assembly/trinity_assemblies/raw_fastq_files/fixed_Bombyx_mori_RNA_R1.cor_val_1.fq"
#read2_file="/home/michal/Documents/BOMO_trinity_denovo_assembly/trinity_assemblies/raw_fastq_files/fixed_Bombyx_mori_RNA_R2.cor_val_2.fq"

####bowtie2 -p 64 --local --no-unal --nofw -x $RAMDISK/ref.fasta -q -1 $RAMDISK/read1_file.fq -2 $RAMDISK/read2_file.fq | samtools view -Sb - | samtools sort -no - - > $RAMDISK/bowtie2.Trinity.nameSorted.bam

bowtie2 -p 64 --local --no-unal -x $SPECIES -q -1 $RAMDISK/read1_file.fq -2 $RAMDISK/read2_file.fq | samtools view -Sb - | samtools sort -no - - > $RAMDISK/bowtie2.Trinity.nameSorted.bam

wait
#samtools sort $RAMDISK/bowtie2.Trinity.nameSorted.bam -o $RAMDISK/bowtie2.Trinity.nameSorted_sorted.bam
#samtools index $RAMDISK/bowtie2.Trinity.nameSorted_sorted.bam
```

Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -c 12
#SBATCH -A m2_imb-pga
#SBATCH -p bigmem #parallel #smp #parallel  # smp #parallel                    # may consider running on a bigmem node for large dataset
#SBATCH -C skylake
#SBATCH -e maxquant_test_%A.err             # File to which STDERR will be written
#SBATCH -o maxquant_test_%A.out             # File to which STDOUT will be written
#SBATCH -J maxquant_test_%A                 # Job name
#SBATCH --mem=354000 #150G                  # Memory requested (mega default units)
##SBATCH --ramdisk=350G #150G
#SBATCH --time=1-20:00:00 #00:20:00 #1-20:00:00              # Runtime in D-HH:MM:SS
#SBATCH --mail-type=ALL                     # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=m.levin@imb-mainz.de # Email to send notifications to

#module load lang/Mono/5.4.1.6-GCCcore-8.3.0
module load bio/MaxQuant/1.6.17.0

set -x
transdecoder_reference_folder=${WORK_DIR}/Trinity_${ASSEMBLY_MODE}_Transdecoder/${SPECIES}

RESULTS_DIR=${WORK_DIR}/Maxquant_results/${SPECIES}/combined_${SPECIES}_${ASSEMBLY_MODE}_mincov${mincov}_ORFsize${minaa}aa_${mode}
mkdir -p $RESULTS_DIR

JOBDIR=/localscratch/$SLURM_JOB_ID
#RAMDISK=$JOBDIR/ramdisk
RAMDISK=$JOBDIR
mkdir -p $RAMDISK

if [ $mode == "reference" ]
        then
        zcat ${PROT_REF_DIR} > $RAMDISK/ref.fa
        #     find ${PROT_REF_DIR} -name "*.fa.gz" -exec zcat '{}' > $RAMDISK/ref.fa \;
        #     zcat "${ensembl_reference_folder}/"*".pep.all.fa.gz" > $RAMDISK/ref.fa
else
        cp ${transdecoder_reference_folder}/Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${mincov}_ORFsize${minaa}aa.fasta.transdecoder.pep $RAMDISK/ref.fa
fi
wait

cp $RAW_MS_DATA/*.raw $RAMDISK/

cp mqpar_default_arbitrary.xml ${RESULTS_DIR}/mqpar_${SPECIES}_mincov${mincov}_ORFsize${minaa}aa_${mode}.xml
sed -i "s|search_space.fasta|${RAMDISK}/ref.fa|" ${RESULTS_DIR}/mqpar_${SPECIES}_mincov${mincov}_ORFsize${minaa}aa_${mode}.xml

oo=$( find $RAW_MS_DATA -name *.raw -exec basename {} \; | wc -l )
echo "$oo"

files=$( find $RAW_MS_DATA -name *.raw -exec basename {} \; )
raw_names_entry=$( for i in $files; do echo '        <string>'$RAMDISK/$i'</string>' ; done )
sed -i "s|raw_MS_files_100pp|${raw_names_entry//$'\n'/\\n}|" ${RESULTS_DIR}/mqpar_${SPECIES}_mincov${mincov}_ORFsize${minaa}aa_${mode}.xml
sed -i "s/\r//g" ${RESULTS_DIR}/mqpar_${SPECIES}_mincov${mincov}_ORFsize${minaa}aa_${mode}.xml

exp=$(find $RAW_MS_DATA -name *.raw -exec basename {} \; | cut -d_ -f1 | rev | cut -d_ -f2-  |rev | cut -d_ -f5-)
experiments_entry=$( for j in $exp; do echo '        <string>'$j'</string>' ; done)
sed -i "s|experiments_100pp|${experiments_entry//$'\n'/\\n}|" ${RESULTS_DIR}/mqpar_${SPECIES}_mincov${mincov}_ORFsize${minaa}aa_${mode}.xml
sed -i "s/\r//g" ${RESULTS_DIR}/mqpar_${SPECIES}_mincov${mincov}_ORFsize${minaa}aa_${mode}.xml
```

Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -c 1
#SBATCH -A m2_imb-pga
#SBATCH -p smp
##SBATCH -e ph_%A.err               # File to which STDERR will be written
##SBATCH -o ph_%A.out              # File to which STDOUT will be written
##SBATCH -J ph_%A                  # Job name
##SBATCH --mem=1K                   # Memory requested (mega default units)
##SBATCH --ramdisk=1G
##SBATCH --time=00:02:00                 # Runtime in D-HH:MM:SS
##SBATCH --mail-type=ALL                 # Type of email notification- BEGIN,END,FAIL,ALL
##SBATCH --mail-user=m.levin@imb.de # Email to send notifications to

echo 'Nothing to do here!'
```

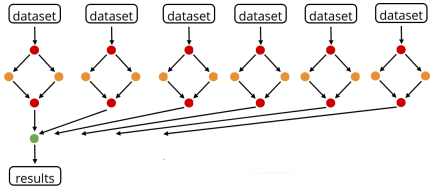
Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -c 32
#SBATCH -A m2_imb-pga
#SBATCH -p parallel # bigmem #parallel #smp #parallel  # smp #parallel                    # may consider running on a bigmem node for large dataset
#SBATCH -C skylake
#SBATCH -e transrate_%A.err              # File to which STDERR will be written
#SBATCH -o transrate_%A.out              # File to which STDOUT will be written
#SBATCH -J transrate_%A                  # Job name
#SBATCH --mem=50G #354000 #150G                  # Memory requested (mega default units)
##SBATCH --ramdisk=350G #150G
#SBATCH --time=05:00:00  #00:20:00  #1-20:00:00                  # Runtime in D-HH:MM:SS
#SBATCH --mail-type=ALL                  # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=m.levin@imb-mainz.de # Email to send notifications to

module load bio/Transrate/1.0.3-gompi-2019b
#SPECIES="50000000"
RESULTS_DIR=${WORK_DIR}/Trinity_transrate/${SPECIES}/Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${mincov}

JOBDIR=/localscratch/$SLURM_JOB_ID
#RAMDISK=$JOBDIR/ramdisk
RAMDISK=$JOBDIR
mkdir -p $RAMDISK
mkdir -p $RAMDISK/transrate_results
mkdir -p $RESULTS_DIR
cp Trinity_${ASSEMBLY_MODE}_assemblies/Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${mincov}.fasta $RAMDISK/Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${mincov}.fasta

REFDIR=${WORK_DIR}/trimmed_reads/${SPECIES}

cat $(find ${REFDIR}/ -type f -name '*.R1.*')  > ${RAMDISK}/left.fq
cat $(find ${REFDIR}/ -type f -name '*.R2.*')  > ${RAMDISK}/right.fq

#cp trimmed_reads/$SPECIES/rep_1/fixed_1.R1.cor_val_1.fq $RAMDISK/left.fq
#cp trimmed_reads/$SPECIES/rep_1/fixed_1.R2.cor_val_2.fq $RAMDISK/right.fq
transrate --loglevel debug --assembly $RAMDISK/Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${mincov}.fasta --left $RAMDISK/left.fq --right $RAMDISK/right.fq --threads ${SLURM_CPUS_PER_TASK} --output=$RAMDISK/transrate_results

cp -vr  ${RAMDISK}/transrate_results/* ${RESULTS_DIR}/
#cp -vr --preserve=timestamps ${RAMDISK}/combined/proc ${RESULTS_DIR}/
```

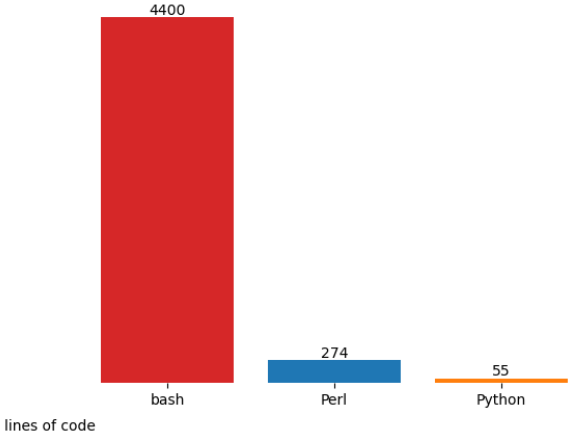
Asynchronous Execution using the Batch System

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -c 1
#SBATCH -A m2_imb-genevop6
#SBATCH -p smp
##SBATCH -e trandec_%A.err          # File to which STDERR will be written
#SBATCH -o transdec_%A.out          # File to which STDOUT will be written
#SBATCH -J transdec_%A              # Job name
#SBATCH --mem=5G #50G               # Memory requested (mega default units)
##SBATCH --gres=ramdisk:1G
#SBATCH --time=01:00:00  #03:00:00              # Runtime in D-HH:MM:SS
#SBATCH --mail-type=ALL             # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=a.ceron@imb-mainz.de # Email to send notifications to

module load bio/TransDecoder/5.5.0-Perl-5.30.0
#module load lang/Perl/5.26.1-foss-2017a
set -x

TRANSCRIPTOME_FILE=<enter transcriptome fasta file path her, make sure to remove the <> brackets afterwards>
ORF_size=100 # adjust according to your needs
RESULTS_FOLDER=./Transdecoder_output
wait

#~/TransDecoder-TransDecoder-v5.4.0/TransDecoder.LongOrfs -m ${ORF_size} -t Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${mincov}_ORFsize${ORF_size}aa.fasta
TransDecoder.LongOrfs -m ${ORF_size} -t ${TRANSCRIPTOME_FILE} --output_dir ${RESULTS_FOLDER}
wait

#~/TransDecoder-TransDecoder-v5.4.0/TransDecoder.Predict -t Trinity-${ASSEMBLY_MODE}_${SPECIES}_mincov${mincov}_ORFsize${ORF_size}aa.fasta
TransDecoder.Predict -t ${TRANSCRIPTOME_FILE} --output_dir ${RESULTS_FOLDER}
```

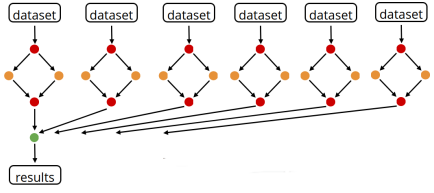
Data Analysis - A Batch-System-Only example!



This workflow with 1 shepherd (master), 11 jobs scripts (for multiple, concurrent execution, incl. dependency handling) was coded by one student:



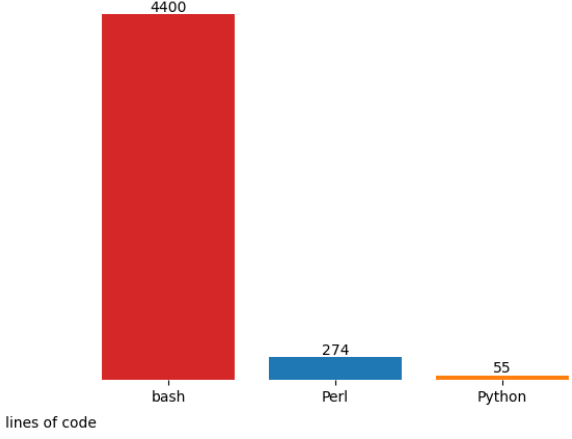
Data Analysis - A Batch-System-Only example!



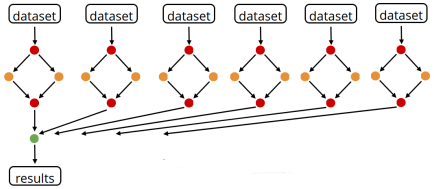
Question

How portable is this? How maintainable? How many HPC users are able to accomplish this?

This workflow with 1 shepherd (master), 11 jobs scripts (for multiple, concurrent execution, incl. dependency handling) was coded by one student:



Data Analysis - A Batch-System-Only example!

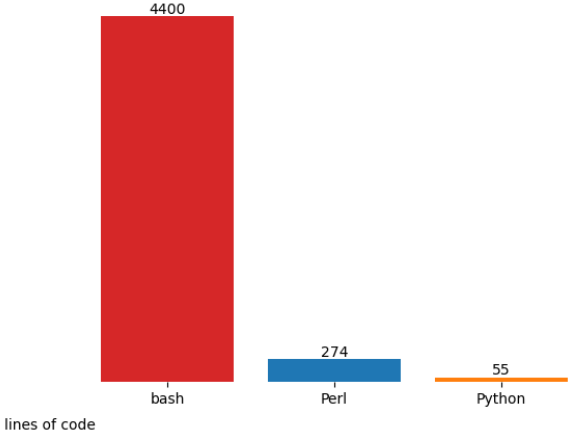


Question

How portable is this? How maintainable? How many HPC users are able to accomplish this?

Which conclusions will your users in need of complex analysis steps draw?

This workflow with 1 shepherd (master), 11 jobs scripts (for multiple, concurrent execution, incl. dependency handling) was coded by one student:



Let them learn! Really?


Contemplate . . .

Let them learn! Really?

Contemplate . . .

When do new users apply for HPC time?

What do users expect to analyze their data?

What will users think, if their expectations aren't met?

Let them learn! Really?


Contemplate . . .


When do new users apply for HPC time?

With BIG DATA - or HUGE SIMULATIONS - not for anything else!

Let them learn! Really?

Contemplate ...

When do new users apply for HPC time?

With BIG DATA - or HUGE SIMULATIONS - not for anything else!

What do users expect to analyze their data?

- all necessary software – no install hiccups

Let them learn! Really?

Contemplate ...

When do new users apply for HPC time?

With BIG DATA - or HUGE SIMULATIONS - not for anything else!

What do users expect to analyze their data?

- all necessary software – no install hiccups
- to *just* calculate – no complaints about IO issues or the like

Let them learn! Really?

Contemplate ...

When do new users apply for HPC time?

With BIG DATA - or HUGE SIMULATIONS - not for anything else!

What do users expect to analyze their data?

- all necessary software – no install hiccups
- to *just* calculate – no complaints about IO issues or the like
- to get their results *fast* – after all, it's an HPC system!

Let them learn! Really?

Contemplate ...

When do new users apply for HPC time?

With BIG DATA - or HUGE SIMULATIONS - not for anything else!

What do users expect to analyze their data?

- all necessary software – no install hiccups
- to *just* calculate – no complaints about IO issues or the like
- to get their results *fast* – after all, it's an HPC system!

Let them learn! Really?


Contemplate . . .

When do new users apply for HPC time?

With BIG DATA - or HUGE SIMULATIONS - not for anything else!

What do users expect to analyze their data?

- all necessary software – no install hiccups
- to *just* calculate – no complaints about IO issues or the like
- to get their results *fast* – after all, it's an HPC system!

What will users think, if their expectations aren't met?

- what a workload compared to "my server"??

Let them learn! Really?

Contemplate ...

When do new users apply for HPC time?

With BIG DATA - or HUGE SIMULATIONS - not for anything else!

What do users expect to analyze their data?

- all necessary software – no install hiccups
- to *just* calculate – no complaints about IO issues or the like
- to get their results *fast* – after all, it's an HPC system!

What will users think, if their expectations aren't met?

- what a workload compared to "my server"??
- what a workload to get here (bureaucracy) and now this??

Let them learn! Really?

Contemplate ...

When do new users apply for HPC time?

With BIG DATA - or HUGE SIMULATIONS - not for anything else!

What do users expect to analyze their data?

- all necessary software – no install hiccups
- to *just* calculate – no complaints about IO issues or the like
- to get their results *fast* – after all, it's an HPC system!

What will users think, if their expectations aren't met?

- what a workload compared to "my server"??
- what a workload to get here (bureaucracy) and now this??
- abandon HPC!

Let them learn! Really? - II

How do users plan?

- is data management accounted for?

Let them learn! Really? - II

How do users plan?

- is data management accounted for?
- are their jobs always (well enough) parameterized?

Let them learn! Really? - II

How do users plan?

- is data management accounted for?
- are their jobs always (well enough) parameterized?
- where do get they their infos from? The "internet"? A labmate? A PI?

Let them learn! Really? - II

How do users plan?

- is data management accounted for?
- are their jobs always (well enough) parameterized?
- where do get they their infos from? The "internet"? A labmate? A PI?

Let them learn! Really? - II

How do users plan?

- is data management accounted for?
- are their jobs always (well enough) parameterized?
- where do get they their infos from? The "internet"? A labmate? A PI?

How do you meet expectations?

- software provisioning schemes

Let them learn! Really? - II

How do users plan?

- is data management accounted for?
- are their jobs always (well enough) parameterized?
- where do get they their infos from? The "internet"? A labmate? A PI?

How do you meet expectations?

- software provisioning schemes
- data management support

Let them learn! Really? - II

How do users plan?

- is data management accounted for?
- are their jobs always (well enough) parameterized?
- where do get they their infos from? The "internet"? A labmate? A PI?

How do you meet expectations?

- software provisioning schemes
- data management support
- curated workflows

Let them learn! Really? - II

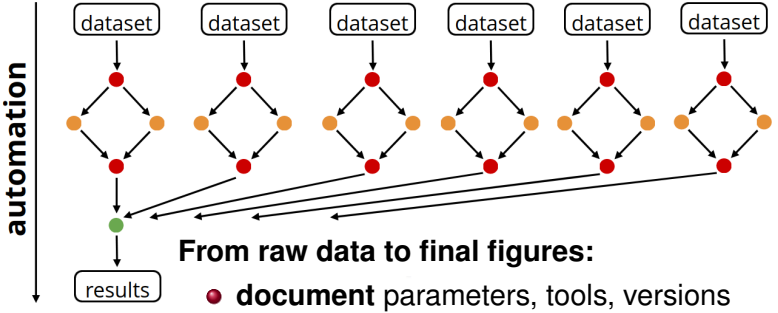
How do users plan?

- is data management accounted for?
- are their jobs always (well enough) parameterized?
- where do get they their infos from? The "internet"? A labmate? A PI?

How do you meet expectations?

- software provisioning schemes
- data management support
- curated workflows
- every item feasible to which extend? With your manpower?

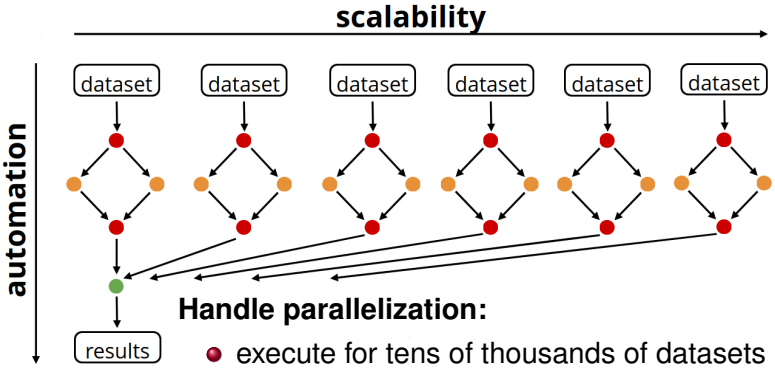
Reproducible Data Analysis



From raw data to final figures:

- **document** parameters, tools, versions
- **execute** without manual intervention

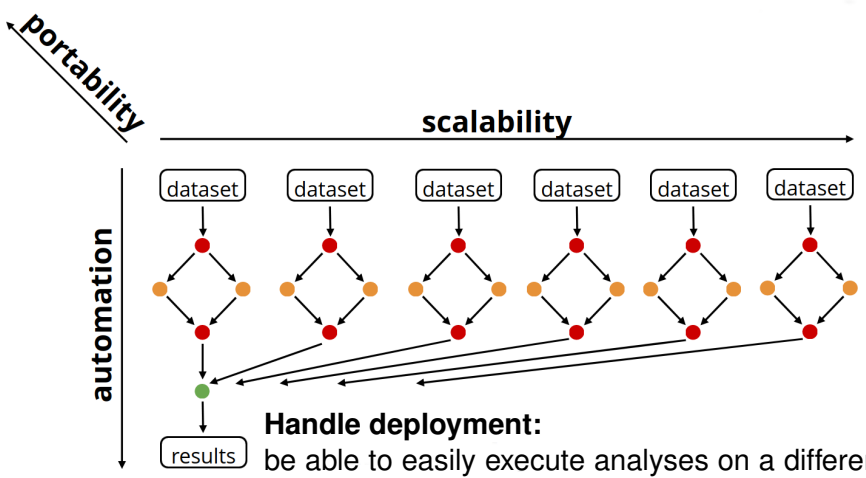
Reproducible Data Analysis



Handle parallelization:

- execute for tens of thousands of datasets
- efficiently use any computing platform

Reproducible Data Analysis



Handle deployment:
be able to easily execute analyses on a different system/platform/infrastructure

Beyond Reproducibility

Reproducibility

From the official **Snakemake** -paper.

Beyond Reproducibility

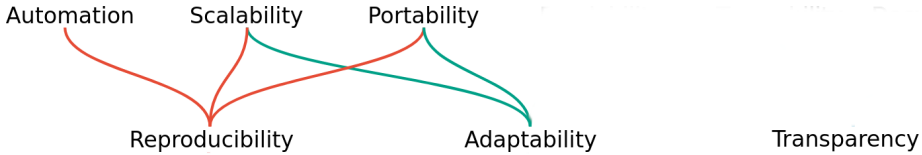
Reproducibility

Adaptability

Transparency

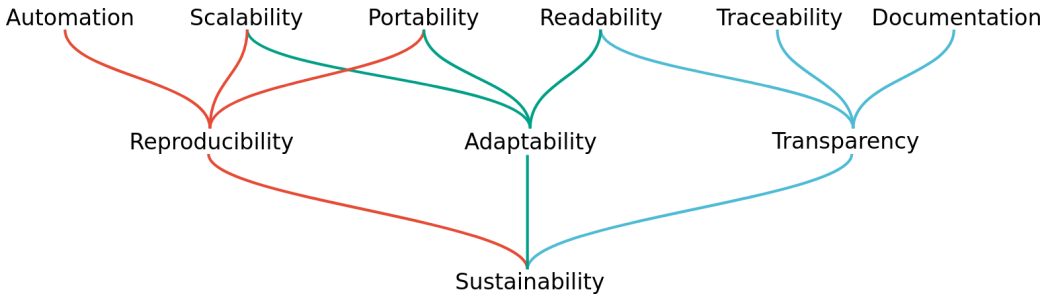
From the official **Snakemake** -paper.

Beyond Reproducibility

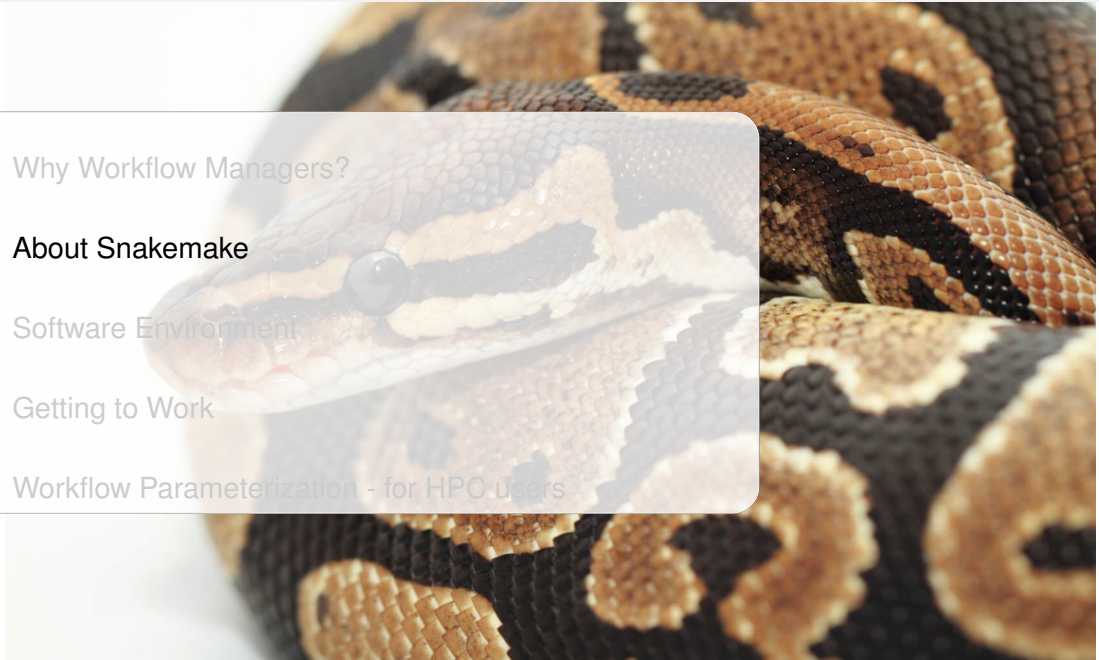


From the official **Snakemake** -paper.

Beyond Reproducibility



From the official **Snakemake** -paper.



1 Why Workflow Managers?

2 About Snakemake

3 Software Environment

4 Getting to Work

5 Workflow Parameterization - for HPC users

What is this about?

Questions

- What is **Snakemake** ?
- How does it work on a Cluster?
- What about other Workflow Management Systems?

Objectives

1. Introduction to **Snakemake** Usage (in-depth introduction for users, only)
2. Get an Mini-Overview about Workflow Systems

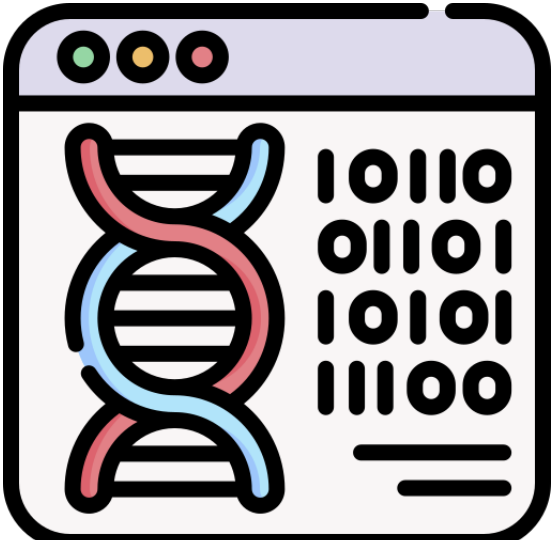
>1e6 downloads since 2015

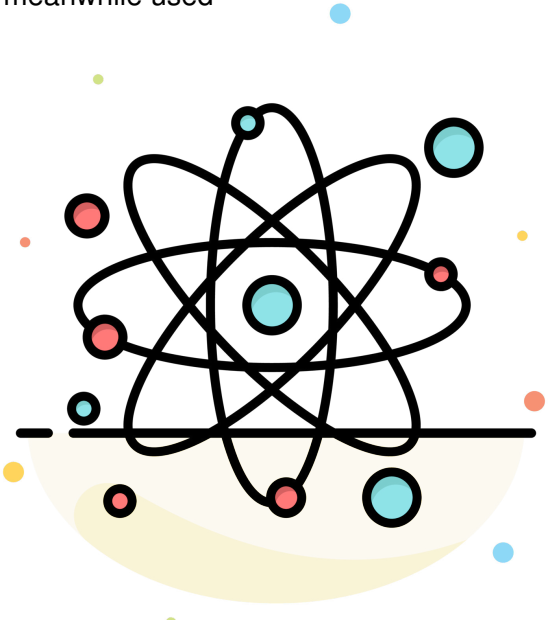
>3000 citations

>7 citations per week since 2021

Snakemake – Bioinformatics to Physics

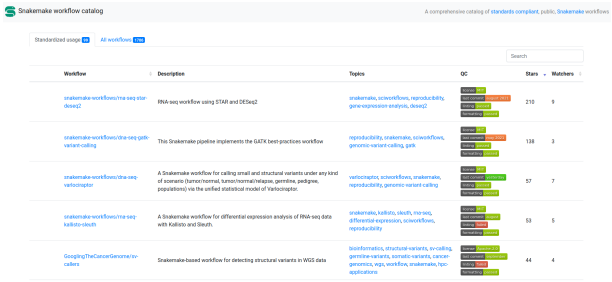
Although developed first for Bioinformatics users, meanwhile used





The Snakemake Catalogue

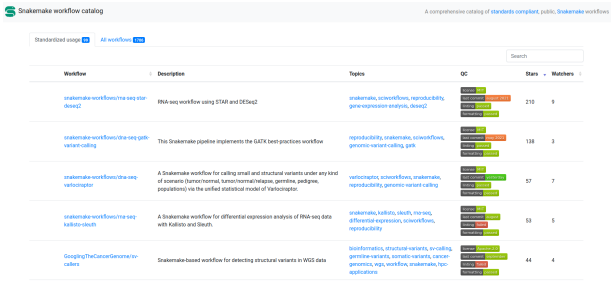
- Extremely feature rich: over 3000 workflows (mostly bioinformatics)



Screenshot of the Workflow Catalogue

The Snakemake Catalogue

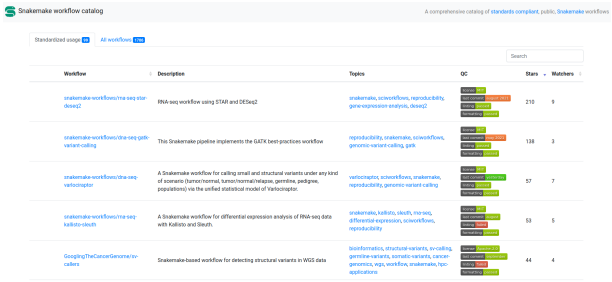
- Extremely feature rich: over 3000 workflows (mostly bioinformatics)
- Almost three hundred standardized workflows ready to use (meaning: well documented and with automatic deployment)



Screenshot of the Workflow Catalogue

The Snakemake Catalogue

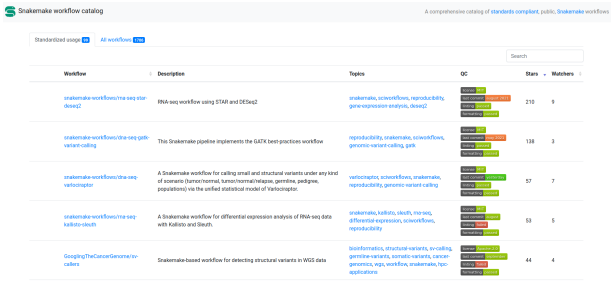
- Extremely feature rich: over 3000 workflows (mostly bioinformatics)
- Almost three hundred standardized workflows ready to use (meaning: well documented and with automatic deployment)
- Cluster batch systems are supported (and support for various cloud systems, too)



Screenshot of the Workflow Catalogue

The Snakemake Catalogue

- Extremely feature rich: over 3000 workflows (mostly bioinformatics)
- Almost three hundred standardized workflows ready to use (meaning: well documented and with automatic deployment)
- Cluster batch systems are supported (and support for various cloud systems, too)
- There is an option to include Nextflow wrappers, too.



Screenshot of the Workflow Catalogue

How does Snakemake on a Cluster work?

- Snakemake is triggered on the command line:

```
$ snakemake [<arguments>]
```

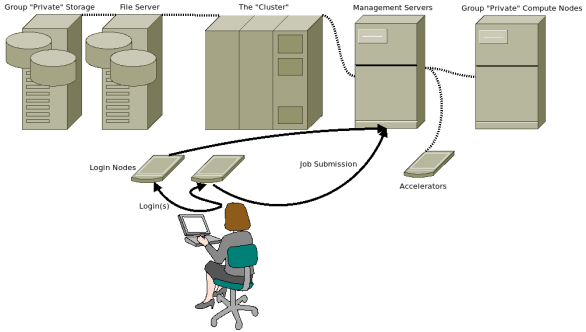
How does Snakemake on a Cluster work?

- Snakemake is triggered on the command line:

```
$ snakemake [<arguments>]
```

- users need to fill in the parameters of their workflow (e. g. input path(s) in a file)

How does Snakemake on a Cluster work?

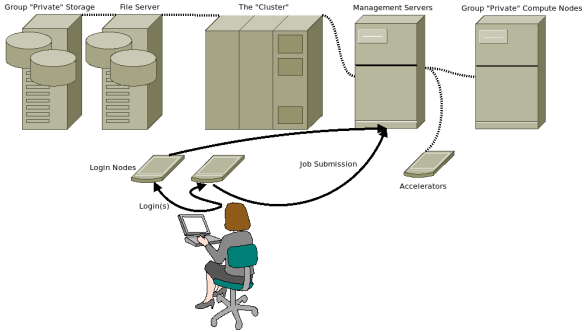


- Snakemake is triggered on the command line:

```
$ snakemake [<arguments>]
```

- users need to fill in the parameters of their workflow (e. g. input path(s) in a file)
- Snakemake will run on the login-node and spawn your jobs on the cluster using a

How does Snakemake on a Cluster work?

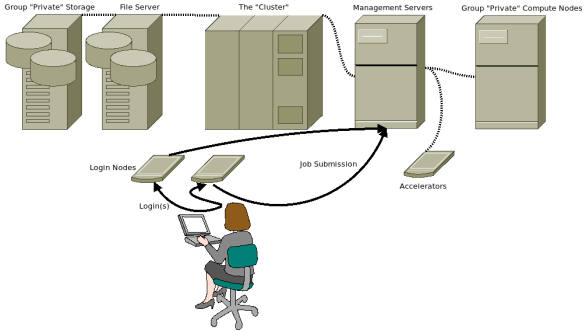


- Snakemake is triggered on the command line:

```
$ snakemake [<arguments>]
```

- users need to fill in the parameters of their workflow (e. g. input path(s) in a file)
- Snakemake will run on the login-node and spawn your jobs on the cluster using a
- cluster specific configuration file

How does Snakemake on a Cluster work?

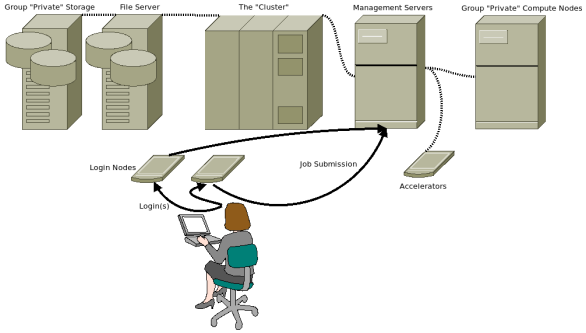


- Snakemake is triggered on the command line:

```
$ snakemake [<arguments>]
```

- users need to fill in the parameters of their workflow (e. g. input path(s) in a file)
- Snakemake will run on the login-node and spawn your jobs on the cluster using a
- cluster specific configuration file

How does Snakemake on a Cluster work?



- Snakemake is triggered on the command line:

```
$ snakemake [<arguments>]
```

- users need to fill in the parameters of their workflow (e. g. input path(s) in a file)
- Snakemake will run on the login-node and spawn your jobs on the cluster using a
- cluster specific configuration file

It is capable to remove temporary files and support various archiving systems.

"Spawn Jobs on a Cluster?!" - What does it mean?

Hint

Here, we explain to users the difference between PC and Servers and HPC Systems. You will get the Admin background, only.

"Spawn Jobs on a Cluster?!" - What does it mean?

Hint

Here, we explain to users the difference between PC and Servers and HPC Systems. You will get the Admin background, only.

- **Snakemake** is triggered on a login-node, only.

"Spawn Jobs on a Cluster?!" - What does it mean?

Hint

Here, we explain to users the difference between PC and Servers and HPC Systems. You will get the Admin background, only.

- **Snakemake** is triggered on a login-node, only.
- It will submit jobs . . .

"Spawn Jobs on a Cluster?!" - What does it mean?

Hint

Here, we explain to users the difference between PC and Servers and HPC Systems. You will get the Admin background, only.

- **Snakemake** is triggered on a login-node, only.
- It will submit jobs . . .
- . . . and keeps running (without CPU load!) to monitor the job execution.

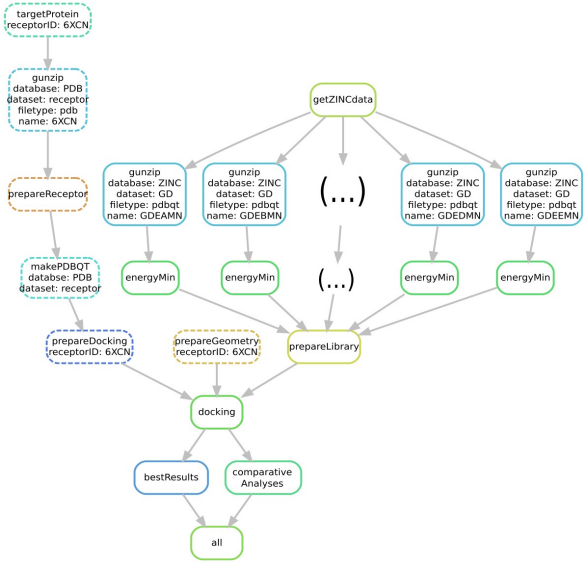
"Spawn Jobs on a Cluster?!" - What does it mean?

Hint

Here, we explain to users the difference between PC and Servers and HPC Systems. You will get the Admin background, only.

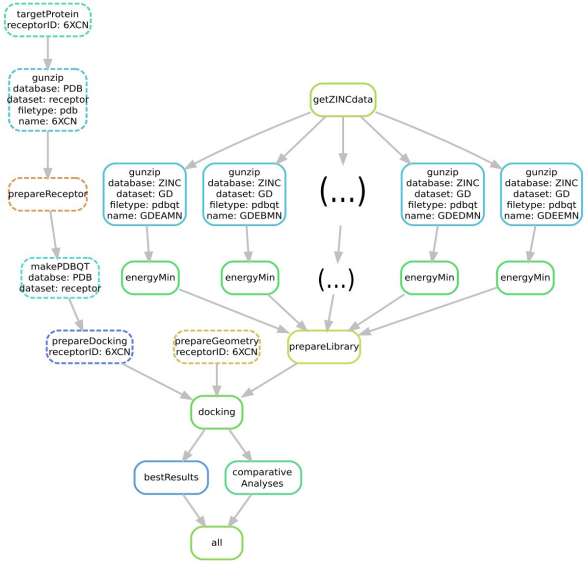
- **Snakemake** is triggered on a login-node, only.
- It will submit jobs . . .
- . . . and keeps running (without CPU load!) to monitor the job execution.
- depending on the workflow, it will download or plot with minor CPU load on the login node. (Hard to notice with a simple `top` command.)

Benefit of Cluster Usage



- **Snakemake** offers
  - to carry out Nextflow wrappers

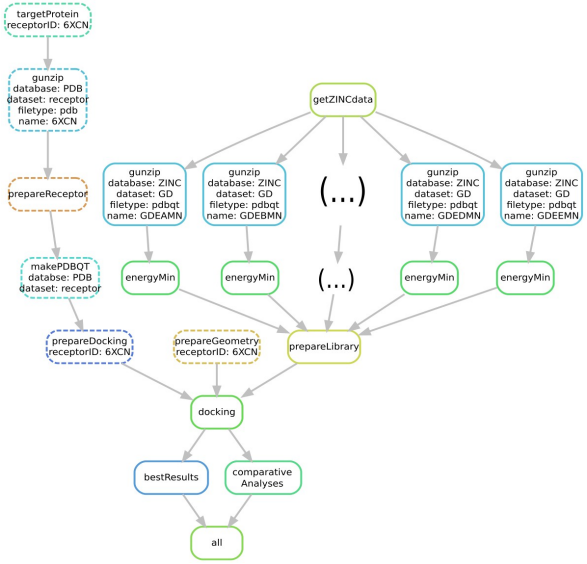
Benefit of Cluster Usage



Snakemake offers

- to carry out Nextflow wrappers
- to react on your input
  $\Rightarrow$ more samples, more jobs

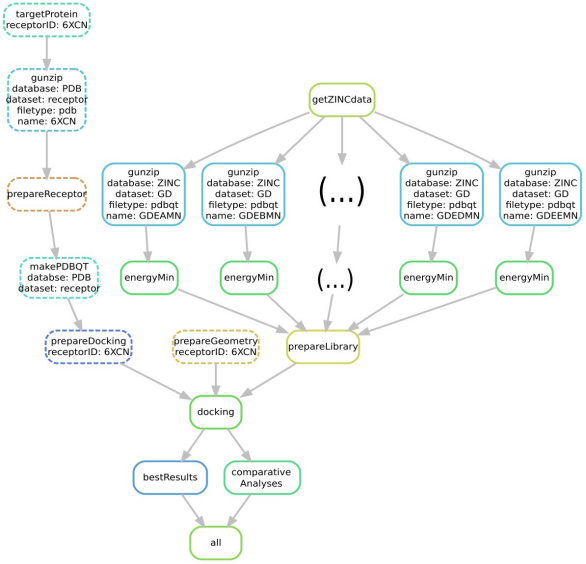
Benefit of Cluster Usage



Snakemake offers

- to carry out Nextflow wrappers
- to react on your input
  $\Rightarrow$ more samples, more jobs
- to be a remedy to I/O contention

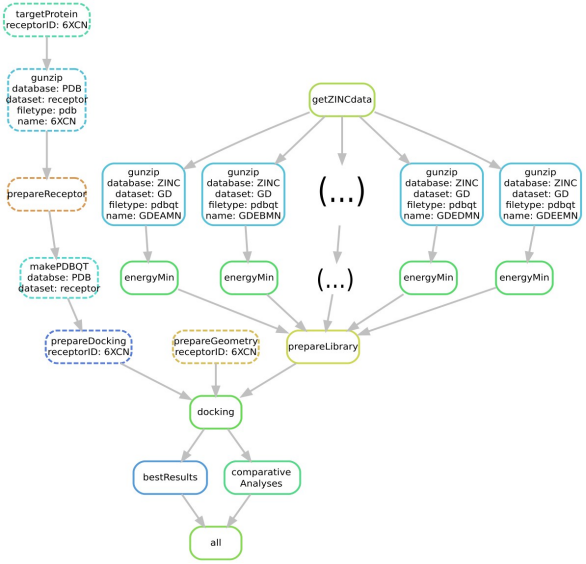
Benefit of Cluster Usage



Snakemake offers

- to carry out Nextflow wrappers
- to react on your input
  $\Rightarrow$ more samples, more jobs
- to be a remedy to I/O contention
- to be ready for real time
  computation with cluster support

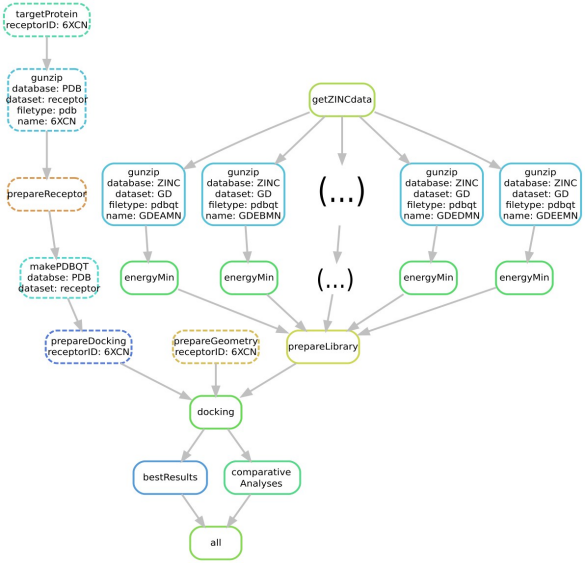
Benefit of Cluster Usage



Snakemake offers

- to carry out Nextflow wrappers
- to react on your input
  $\Rightarrow$ more samples, more jobs
- to be a remedy to I/O contention
- to be ready for real time
  computation with cluster support
- to support you from your input
  to publication

Benefit of Cluster Usage



Snakemake offers

- to carry out Nextflow wrappers
- to react on your input
  $\Rightarrow$ more samples, more jobs
- to be a remedy to I/O contention
- to be ready for real time
  computation with cluster support
- to support you from your input
  to publication
- ...

Software Environment

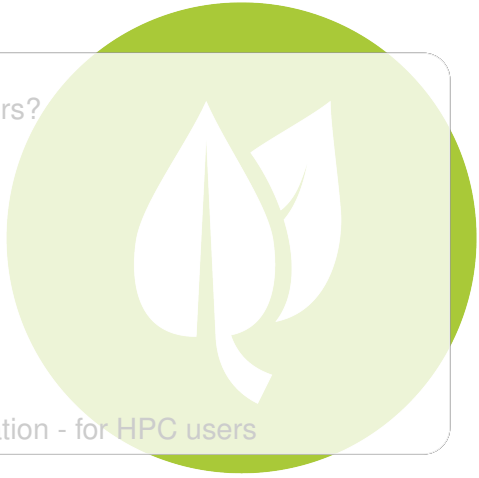
1 Why Workflow Managers?

2 About Snakemake

3 Software Environment

4 Getting to Work

5 Workflow Parameterization - for HPC users



What is this about?

Questions

- How does your software support integrate with Snakemake ?

Objectives

1. knowing software selections supported by Snakemake
2. avoiding software selection conflicts

Snakemake's Software Provisioning - Overview

- **Snakemake** basically offers 3 software deployment methods:
  - Container

Snakemake's Software Provisioning - Overview

- **Snakemake** basically offers 3 software deployment methods:
  - Container
  - Environment Modules

Snakemake's Software Provisioning - Overview

- **Snakemake** basically offers 3 software deployment methods:
  - Container
  - Environment Modules
  - Conda (and its derivatives)

Snakemake's Software Provisioning - Overview

- **Snakemake** basically offers 3 software deployment methods:
  - Container
  - Environment Modules
  - Conda (and its derivatives)
  - *in development*: Conda Translation to EESSI

Snakemake's Software Provisioning - Overview

- **Snakemake** basically offers 3 software deployment methods:
  - Container
  - Environment Modules
  - Conda (and its derivatives)
  - *in development*: Conda Translation to EESSI

Snakemake's Software Provisioning - Overview

Snakemake basically offers 3 software deployment methods:

- Container
- Environment Modules
- Conda (and its derivatives)
- *in development*: Conda Translation to EESSI

While Conda is the preferred and *portable* way with Snakemake, other software methods can be selected with

```
$ snakemake --sdm=[conda,envmodules,apptainer]
```

`--sdm` is short for `--software-deployment-method`.

Snakemake's Software Provisioning - Overview

Snakemake basically offers 3 software deployment methods:

- Container
- Environment Modules
- Conda (and its derivatives)
- *in development*: Conda Translation to EESSI

While Conda is the preferred and *portable* way with Snakemake, other software methods can be selected with

```
$ snakemake --sdm=[conda,envmodules,apptainer]
```

`--sdm` is short for `--software-deployment-method`.

Hint

Using environment modules or container based deployment *is possible*, but tedious if used for more than a *few* 3rd party programs.

What we teach . . .

- first we show them how to install software into en environment

What we teach . . .

- first we show them how to install software into en environment
- later one environment per worfklow

What we teach . . .

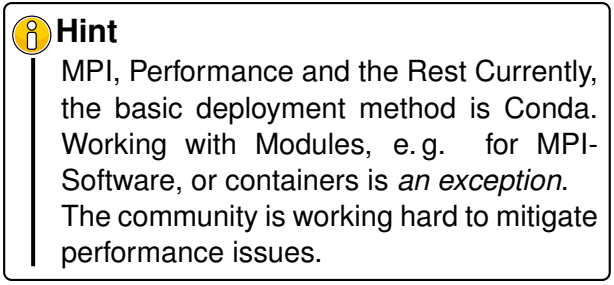
- first we show them how to install software into en environment
- later one environment per worfklow
- then how to delegate the deployment to **Snakemake** – and live with a very basic environment

What we teach . . .

- first we show them how to install software into en environment
- later one environment per worfklow
- then how to delegate the deployment to **Snakemake** – and live with a very basic environment

What we teach . . .

- first we show them how to install software into en environment
- later one environment per worfklow
- then how to delegate the deployment to **Snakemake** – and live with a very basic environment



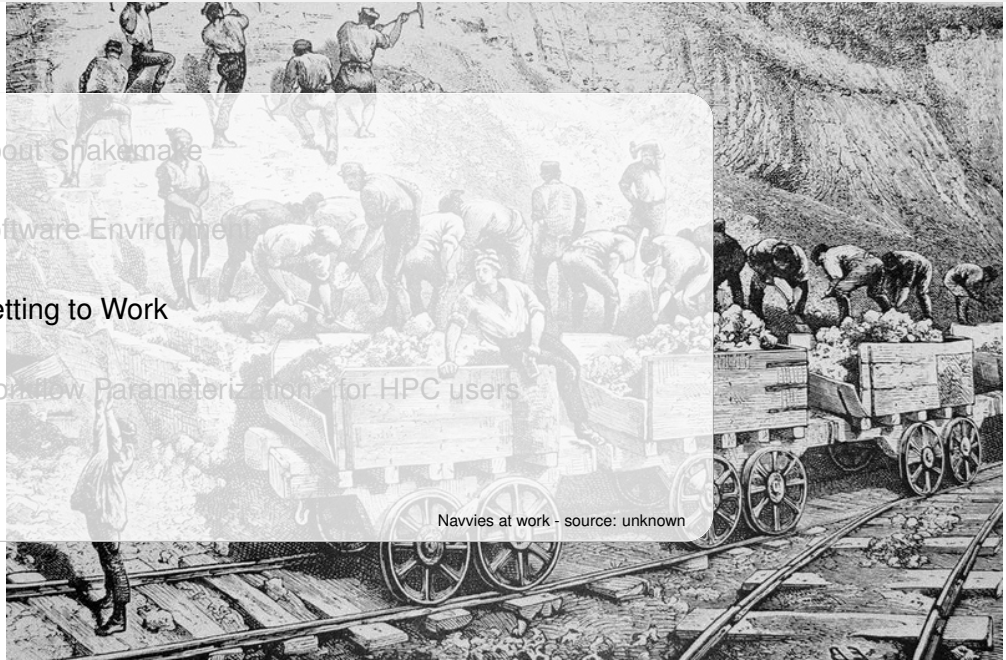
How does Snakemake work?

2 About Snakemake

3 Software Environments

4 Getting to Work

5 Workflow Parameterization for HPC users



Navvies at work - source: unknown

What is this about?

Questions

- How does **Snakemake** actually work?
- Which features are provided?
- How to write and select a workflow?

Objectives

1. Understand the structure of a workflow.
2. Getting to know *some* features.
3. Plus some HPC-oriented features.

An (incomplete) Feature List

```
rule bwa_map:
    input:
        "data/genome.fa",
        "data/samples/A.fastq"
    output:
        "mapped_reads/A.bam"
    shell:
        "bwa mem data/genome.fa"
        " data/samples/A.fastq"
        " | samtools view -Sb - >"
        " mapped_reads/A.bam"
```

- rule oriented, data centric

An (incomplete) Feature List

```
rule bwa_map:
    input:
            "{reference}",
            "{sample}.fastq"
        output:
             "mapped_reads/A.bam"
        shell:
             "bwa mem {input}"
             " | samtools view -Sb - >"
             " {output}"
```

- rule oriented, data centric
- readable and scalable (wildcards!)

An (incomplete) Feature List

```
rule plot_quals:
    input:
        "calls/all.vcf"
    output:
        "plots/quals.svg"
    script:
        "scripts/plot-quals.py"
```

- rule oriented, data centric
- readable and scalable (wildcards!)
- direct integration of Python, R, Perl, Bash scripts

An (incomplete) Feature List

```
rule plot_quals:
    input:
        "calls/all.vcf"
    output:
        "plots/quals.svg"
    run:
        import matplotlib as mpl
        ...
```

- rule oriented, data centric
- readable and scalable (wildcards!)
- direct integration of Python, R, Perl, Bash scripts
- integrated support for Python snippets

An (incomplete) Feature List

```
rule filter_vcf:
    input:
        "{sample}.vcf"
    output:
        "{sample}.filtered.vcf"
    params:
        extra="--chr 1 --recode-INFO-all"
    wrapper:
        "v5.5.0/bio/vcftools/filter"
```

- rule oriented, data centric
- readable and scalable (wildcards!)
- direct integration of Python, R, Perl, Bash scripts
- integrated support for Python snippets
- auto-download of wrappers helping with quirky software

Ensured Portability

```
rule select_by_country:
    input:
        "data/worldcitiespop.csv"
    output:
        "by-country/{country}.csv"
    conda:
        "envs/xsv.yaml"
    shell:
        "xsv search -s Country "
        "'{wildcards.country}' "
        "{input} > {output}"
```

By integration with the Conda package manager and containers, all software dependencies of each workflow step are automatically deployed upon execution.

Turing Completeness

```
def get_data(wildcards):
    # use arbitrary Python logic to
    # aggregate over the required
    # input files
    return ...

rule plot_histogram:
    input:
        get_data
    output:
        "plots/hist.svg"
    script:
        "scripts/plot-hist.py"
```

Being a syntactical extension of Python, you can implement arbitrary logic beyond the plain definition of rules. Rules can be generated conditionally, arbitrary Python logic can be used to perform aggregations, configuration and metadata can be obtained and postprocessed in any required way.

Human Readable

Despite its complexity semantic helper functions ease readability.

```
rule plot_histogram:
  input:
   branch(
      lookup(dpath="histogram/somedata",
             within=config),
      then="data/somedata.txt",
      otherwise="data/someotherdata.txt"
    )
  output:
    "plots/hist.svg"
  script:
    "scripts/plot-hist.py"
```

Dynamic Workflows

```
rule all:
    input:
        from_queue(all_results,
            finish_sentinel=...)

checkpoint somestep:
    input:
        "samples/{sample}.txt"
    output:
        "somestep/{sample}.txt"
    shell:
        "somecommand {input} > {output}"
```

- using so-called "checkpoints" **Snakemake** can adapt workflows as runtime
- the `from_queue` command allows almost "realtime" computing, e. g. drawing data from measurement devices via a mounted file system (smb/gio, sshfs, etc.)

Generic but Portable

Workflows are generic, i. e. not altered when ported. Using `snakemake --workflow-profile ...` users can select a yaml file with deviating cluster settings, e. g.

```
default-resources:
    slurm_account: "nhr-zdvhpc"
    slurm_partition: "smallcpu"
    mem_mb_per_cpu: 1800

set-resources
  map_reads:
    mem_mb_per_cpu: 2700
    runtime: "30min"
    slurm_partition: "parallel" # pooled

  ...

set-threads:
  map_reads: 32
  ...
```

HPC Specifics - General

Question

What does the SLURM-plugin for **Snakemake** provide?

- submission of jobs ✔

HPC Specifics - General

Question

What does the SLURM-plugin for **Snakemake** provide?

- submission of jobs ✔
- status tracking ✔

HPC Specifics - General

Question

What does the SLURM-plugin for **Snakemake** provide?

- submission of jobs ✔
- status tracking ✔
- distinction between cluster jobs and local jobs (e.g. plots) ✔

HPC Specifics - General

Question

What does the SLURM-plugin for **Snakemake** provide?

- submission of jobs ✔
- status tracking ✔
- distinction between cluster jobs and local jobs (e.g. plots) ✔
- auto-cleanup of SLURM logs (logs of successful jobs), no more manual cleaning of zillions of files! ✔

HPC Specifics - General

Question

What does the SLURM-plugin for **Snakemake** provide?

- submission of jobs ✔
- status tracking ✔
- distinction between cluster jobs and local jobs (e.g. plots) ✔
- auto-cleanup of SLURM logs (logs of successful jobs), no more manual cleaning of zillions of files! ✔
- multicluster support ✔

HPC Specifics - General

Question

What does the SLURM-plugin for **Snakemake** provide?

- submission of jobs ✔
- status tracking ✔
- distinction between cluster jobs and local jobs (e.g. plots) ✔
- auto-cleanup of SLURM logs (logs of successful jobs), no more manual cleaning of zillions of files! ✔
- multicluster support ✔

HPC Specifics - General

Question

What does the SLURM-plugin for **Snakemake** provide?

- submission of jobs ✔
- status tracking ✔
- distinction between cluster jobs and local jobs (e.g. plots) ✔
- auto-cleanup of SLURM logs (logs of successful jobs), no more manual cleaning of zillions of files! ✔
- multicluster support ✔
- feature selection with `--constraint` ✔

HPC Specifics - General

Question

What does the SLURM-plugin for **Snakemake** provide?

- submission of jobs ✔
- status tracking ✔
- distinction between cluster jobs and local jobs (e.g. plots) ✔
- auto-cleanup of SLURM logs (logs of successful jobs), no more manual cleaning of zillions of files! ✔
- multicluster support ✔
- feature selection with `--constraint` ✔
- improved CPU binding ✔

HPC Specifics - General

Question

What does the SLURM-plugin for **Snakemake** provide?

- submission of jobs ✔
- status tracking ✔
- distinction between cluster jobs and local jobs (e.g. plots) ✔
- auto-cleanup of SLURM logs (logs of successful jobs), no more manual cleaning of zillions of files! ✔
- multicluster support ✔
- feature selection with `--constraint` ✔
- improved CPU binding ✔
- retries ✔

HPC Specifics - General

Question

What does the SLURM-plugin for **Snakemake** provide?

- submission of jobs ✔
- status tracking ✔
- distinction between cluster jobs and local jobs (e.g. plots) ✔
- auto-cleanup of SLURM logs (logs of successful jobs), no more manual cleaning of zillions of files! ✔
- multicluster support ✔
- feature selection with `--constraint` ✔
- improved CPU binding ✔
- retries ✔
- requeuing (after preemption) ✔

HPC Specifics - General

Question

What does the SLURM-plugin for **Snakemake** provide?

- submission of jobs ✔
- status tracking ✔
- distinction between cluster jobs and local jobs (e.g. plots) ✔
- auto-cleanup of SLURM logs (logs of successful jobs), no more manual cleaning of zillions of files! ✔
- multicluster support ✔
- feature selection with `--constraint` ✔
- improved CPU binding ✔
- retries ✔
- requeuing (after preemption) ✔
- JobArrays ✗ (in development)

HPC Specifics - General

Question

What does the SLURM-plugin for **Snakemake** provide?

- submission of jobs ✔
- status tracking ✔
- distinction between cluster jobs and local jobs (e.g. plots) ✔
- auto-cleanup of SLURM logs (logs of successful jobs), no more manual cleaning of zillions of files! ✔
- multicluster support ✔
- feature selection with `--constraint` ✔
- improved CPU binding ✔
- retries ✔
- requeuing (after preemption) ✔
- JobArrays ✗ (in development)
- Pooling SMP jobs ✗ (in development)

HPC Specifics - General

Question

What does the SLURM-plugin for **Snakemake** provide?

- submission of jobs ✔
- status tracking ✔
- distinction between cluster jobs and local jobs (e.g. plots) ✔
- auto-cleanup of SLURM logs (logs of successful jobs), no more manual cleaning of zillions of files! ✔
- multicluster support ✔
- feature selection with `--constraint` ✔
- improved CPU binding ✔
- retries ✔
- requeuing (after preemption) ✔
- JobArrays ✗ (in development)
- Pooling SMP jobs ✗ (in development)
- I/O pattern annotation ✗ (hopefully March)

HPC Specifics - MPI

```
rule simulate:
    input: ...
    output: ...
    resources:
        mpi: "srun"
    shell: "gmx {params} ..."
```

- remember: `resources` can be specified in a file, too.
- using `srun` can achieve *any* MPI topology under SLURM
- `srun` or `mpiexec` or any other MPI-starter can be specified (not all MPI software accepts `srun`)

HPC Specifics - Accelerators

- for ML or otherwise - GPU selection with `gres` or `gpus`, analogous to SLURM
- allows to set CPUs per GPU
- allows multi-GPU jobs

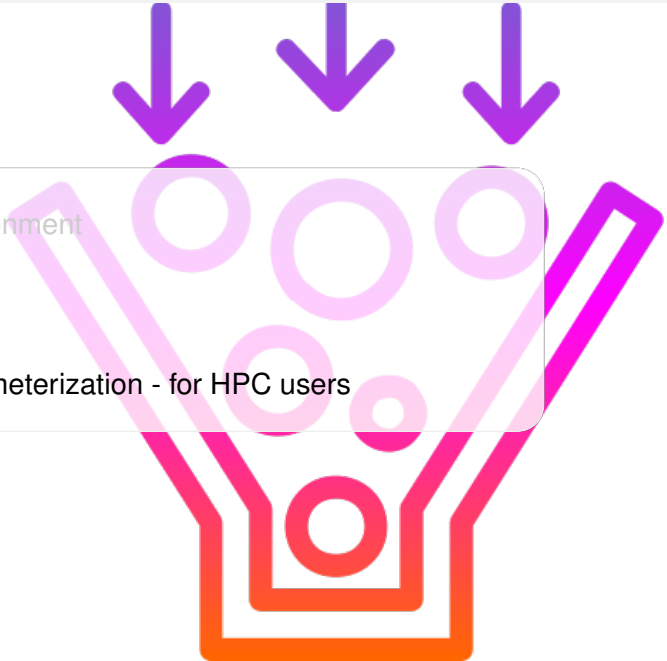
```
rule simulate:
    ...
    resources:
        gpus=2,
        cpus_per_gpu=4
    shell:
        "gmx mdrun {params} ..."
```

Avoiding I/O Contention

3 Software Environment

4 Getting to Work

5 Workflow Parameterization - for HPC users



from Flaticon

What is this about?

Questions

- Avoiding I/O contention! How?
- Accounting for FS Latency! How?

Objectives

1. Learn how to tune **Snakemake** to mitigate I/O contention.
2. Learn how to configure **Snakemake** to allow for file system latency.

Interlude – Random Access and File System Latency

Hint

Here, we explain to users what random access patterns are and how they arise.
We tell – roughly – the story of file system latencies, their causes and characteristics of different file systems and speeds.

Interlude – Random Access and File System Latency

Hint

Here, we explain to users what random access patterns are and how they arise.
We tell – roughly – the story of file system latencies, their causes and characteristics of different file systems and speeds.
We do not feel it's appropriate to lecture you. Instead, we show how easy **Snakemake** will offer a solution to I/O contention.

Hint

Profiles can shorten command lines and can be an easy remedy for I/O issues!

Hint

Profiles can shorten command lines and can be an easy remedy for I/O issues!

Two kinds of profiles are supported:

- A global profile that is defined in a system-wide or user-specific configuration directory (on Linux, this will be `~/.config/snakemake` or `/etc/xdg/snakemake`, you can find the answer for your system via `snakemake --help`).

Hint

Profiles can shorten command lines and can be an easy remedy for I/O issues!

Two kinds of profiles are supported:

- A global profile that is defined in a system-wide or user-specific configuration directory (on Linux, this will be `~/.config/snakemake` or `/etc/xdg/snakemake`, you can find the answer for your system via `snakemake --help`).
- A workflow specific profile that is defined via a flag (`--workflow-profile`) or searched in a default location (profile/default) in the working directory or next to the `Snakefile`.

Our first line defines the so-called executor to be set for SLURM:

```
executor: slurm
```

No more, `snakemake --executor slurm ...` !

The next line tells Snakemake to wait for a few seconds, if output files are not present. This is more than enough time, even for NFS-Filesystems (usually).

```
executor: slurm
latency-wait: 5
```

The entire rest, will tell the storage plugin (`snakemake-storage-plugin-fs`) to stage in to the node-local storage on Mogon, for *every* job and to copy back your results. When dealing with I/O intensive jobs, this can boost your performance tremendously.

```
executor: slurm
latency-wait: 5
default-storage-provider: fs
shared-fs-usage:
  - persistence
  - sources
  - source-cache
remote-job-local-storage-prefix: /localscratch/$SLURM_JOB_ID
local-storage-prefix: /dev/shm/$USER
```

```
executor: slurm
latency-wait: 5
default-storage-provider: fs
shared-fs-usage:
  - persistence
  - sources
  - source-cache
remote-job-local-storage-prefix: /localscratch/$SLURM_JOB_ID
local-storage-prefix: /dev/shm/$USER
```

The complete configuration out to be in `~/.config/snakemake/config.yaml` per user or globally on your cluster at `/etc/xdg/snakemake`.

```
executor: slurm
latency-wait: 5
default-storage-provider: fs
shared-fs-usage:
- persistence
- sources
- source-cache
remote-job-local-storage-prefix: /localscratch/$SLURM_JOB_ID
local-storage-prefix: /dev/shm/$USER
```

An example Call

```
snakemake -j unlimited --workflow-profile profile/<dir> --configfile <file> \
> --directory not_HOME --sdm conda --conda-cleanup-pkgs \
> --conda-prefix [<HOME or not HOME>]
```

- `-j` a semaphore

An example Call

```
snakemake -j unlimited --workflow-profile profile/<dir> --configfile <file> \
> --directory not_HOME --sdm conda --conda-cleanup-pkgs \
> --conda-prefix [<HOME or not HOME>]
```

- `-j` a semaphore
- `--directory` , deploy workflow (scripts) in `HOME`, work on parallel FS workdir

An example Call

```
snakemake -j unlimited --workflow-profile profile/<dir> --configfile <file> \
> --directory not_HOME --sdm conda --conda-cleanup-pkgs \
> --conda-prefix [<HOME or not HOME>]
```

- `-j` a semaphore
- `--directory` , deploy workflow (scripts) in `HOME`, work on parallel FS workdir
- `--conda-prefix` , if dealing with file quotas in `HOME`

- `snakemake` - with numerous dependencies, e. g. `GraphViz`

The Snakemake Software Stack

- `snakemake` - with numerous dependencies, e. g. `GraphViz`
- `snakemake–executor–plugin–slurm` - SLURM stuff (plus dependency)

The Snakemake Software Stack

- `snakemake` - with numerous dependencies, e. g. `GraphViz`
- `snakemake-executor-plugin-slurm` - SLURM stuff (plus dependency)
- `snakemake-interface-storage-plugins` - top interface module

The Snakemake Software Stack

- `snakemake` - with numerous dependencies, e. g. `GraphViz`
- `snakemake-executor-plugin-slurm` - SLURM stuff (plus dependency)
- `snakemake-interface-storage-plugins` - top interface module
- `snakemake-interface-report-plugins` - for generating reports

The Snakemake Software Stack

- `snakemake` - with numerous dependencies, e. g. `GraphViz`
- `snakemake-executor-plugin-slurm` - SLURM stuff (plus dependency)
- `snakemake-interface-storage-plugins` - top interface module
- `snakemake-interface-report-plugins` - for generating reports
- `snakemake-storage-plugin-fs` - for stage-in/-out

The Snakemake Software Stack

- `snakemake` - with numerous dependencies, e. g. `GraphViz`
- `snakemake–executor–plugin–slurm` - SLURM stuff (plus dependency)
- `snakemake–interface–storage–plugins` - top interface module
- `snakemake–interface–report–plugins` - for generating reports
- `snakemake–storage–plugin–fs` - for stage-in/-out
- `snakemake–wrapper–utils` - for wrapper support

Upcoming Courses

Do you want to learn:

- How to create & publish **Snakemake** workflows suitable for HPC clusters?
- Learn how to deploy and adapt 3rd-party workflows?
- and more . . .

Check out these courses in

- Mainz, 29. & 30. Jan.
- Dresden, 26. & 27. Feb.

The End

Thank you for your attention!