SHAPE Project ACOBIOM -CINES Partnership: MARS: Matrix of RNA-Seq

D. Piquemal, R Bruno, V Cameo, B Cirou, Laurent Manchon, F Pierrat

Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

SHAPE Project *ACOBIOM - CINES Partnership:*

MARS: Matrix of RNA-Seq

D. Piquemal[b], R. Bruno[b], V. Cameo[a],B. Cirou[a1], L. Manchon[b], F. Pierrat[b]

[a]*Centre Informatique National de l'Enseignement Supérieur (CINES), 950 rue St Priest, 34097 Montpellier, France*
[b]*ACOBIOM, CS77394, 1682 rue de la Valsière, 34184 Montpellier Cedex 4, France*

Abstract

RNA-Seq approach is used in a wide variety of applications. These include identifying disease-related genes, analysing the effects of drugs on tissues, and providing insight into disease pathways. The RNA-Seq is widely used to characterise gene expression patterns associated with tumor formation. Since RNA-Seq provides absolute values and does not require any calibration with arbitrary standards, results can be compared at any time with other data, even raised by independent laboratories. Once collected, this data can be digitalised and then easily and reliably compared in silico with the growing library of RNA-Seq databases generated for normal and pathological situations in other laboratories around the world (Human: ~27000 libraries. Average size of a library: 1.7GB. Total size: 120TB).
The objectives of MaRS database is to provide and centralize a standardized data used in a wide variety of applications like identifying disease-related genes, analysing the effects of drugs on tissues or providing insight into disease pathways.

Introduction

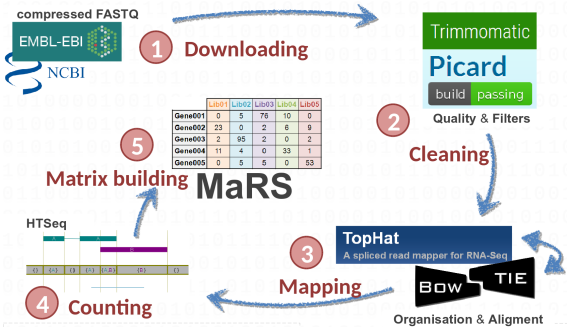

Figure 1: Workflow to produce the Matrix of RNA-Seq

The genomic research community has access to hundreds of terabytes of data. Patients are more acostumized with electronic information and less reticent to share it as they trust that their privacy is respected. Worldwide research teams in hospitals evaluate treatments on cohorts of patients, store individual health, clinical or molecular results in databases and publish their results. A key enabling has been the automatization which avoid the need of hand crafted works on electropohoresis gels thanks to the rise of high throughput sequencers. They delivers data within days for hundreds of dollars. Big resources are then needed for conducting the implied data computations. The cloud or HPC centres are used, like CINES/GENCI one in our case, to extract or produce new results.

1Corresponding author. *E-mail address*: cirou at. cines.fr

The aim of the MaRS (Matrix of RNA-Seq) research program is to collect and to allow the comparison of this data. MaRS is focused on the RNA-Seq method [1], which reflects the expression of the genes in a specific condition. The whole work done is summarised in figure 1, from downloading the data in step one to processing them from step two to five.
The information collected in the final matrix will provide the ability to compare gene expression profiles under different conditions and therapies, to discover new therapeutic targets or stratify patients for personal medicine.

Preparing the data for the production

The list of data (27000 library filenames of Human RNA-Seq profiles) to download was selected as relevant for this computational experiment on the Human Genome. We selected them for their property of having comparable data with same type of data and same technology of sequencing but from multiple sources of laboratories and countries (i.e. different ethnic groups). The corresponding tissues shown by the figure 2 target a broad range of organs and for each of them the pathologies mostly encountered.

Thanks to Jonathan Trow's advices on retrieve tool commands named ascp [7], sftp and wget, we conducted benchmarks. We finally chose wget using HTTP or FTP protocol as four mirrors servers were available whereas only two were accessible through ascp or scp commands. More over thanks to the protocols used those servers had the restart capability and no overhead due to the encryption of the datastream. It is to notice that the raw FASTQ file format access is allowed to private companies. The RNA-Seq libraries were collected from public databases (EBI [6]). Thanks to summer holidays, the academic network RENATER 10Gb/s was free so it took only three weeks to download 120TB of library files from eight nodes hosted at CINES (Montpellier, France). We used a load sharing scripts that maintained a total of sixty four different file transfers occurring at the same time. The downloaded files were directly put, with the same directory hierarchy as on their server, into the fastest but least safe lustre file system /scratch of the Occigen machine. We then submitted SLURM jobs on it to check the integrity by deflating all these files. The list of corrupted files was downloaded anew and checked until all the files were correct. We saved a copy of those files on the Hierarchical Storage file system (i.e. disks and tapes) of the CINES/GENCI Occigen machine.

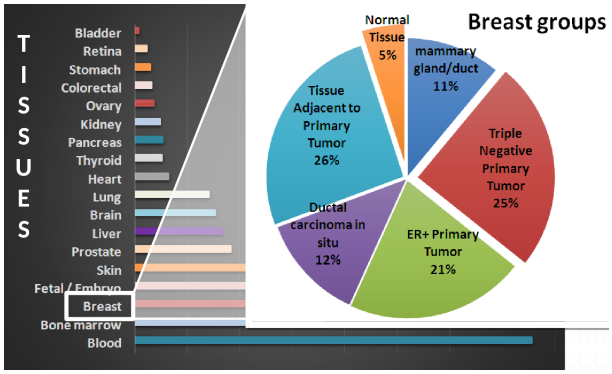

Figure 2: Relative percentage of each tissue considered

Tool chain installation, validation and use through SLURM jobs

We adapted the installation scripts available from each tool (Trimmomatic[9], Samtools [13][2], Tophat2 [11], Picard [14], Bowtie [10] and Htseq [12]) to take advantage of the optimisations specific to the Occigen processors provided by the intel compilers 15.3.

We ran checks and verified the results on one library, then we added automatic checks to resubmit jobs that did not produced the expected files.

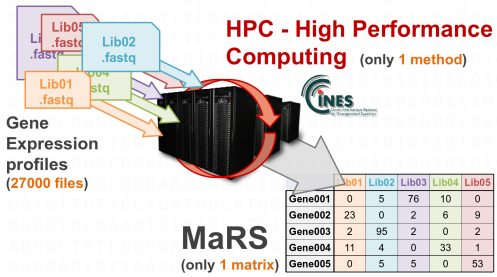

Figure 3: HPC centre usage

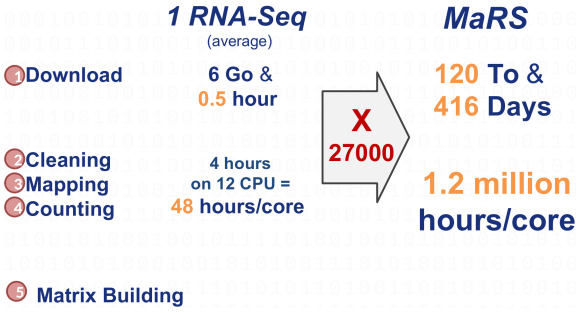

Figure 4: Average statistics for consumed computing hours

As in many supercomputing centres, the CINES policy is to have jobs of maximum 24 hours as a standard. This allows the CINES to apply emergency system updates on average within 12 hours without stopping the production. To meet this constraint, we split the computations in three jobs named part_I, part_II.split.0 and part_II.split.1 with the commands and enhanced parameters from [8]. We reproduced some extracts below:

part_I.sh

```
#### Trimmomatic cleaning step ####
trimmomatic-0.36.jar PE -threads 12 -phred33  [...]    ILLUMINACLIP:$ADAPTERS/TruSeq3-PE-2.fa:2:40:15:8:true  LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36
bowtie2 -x $BOWTIE_INDEX/Homo_sapiens_GRCh38 -q -p 12 -I 0 -X 600 --fr

picard.jar CollectInsertSizeMetrics [...]  HISTOGRAM_WIDTH=500 ASSUME_SORTED=true MAX_RECORDS_IN_RAM=null VALIDATION_STRINGENCY=SILENT
```

part_II.split.0.sh

```
length_read=`zcat $INPUT/$LIB_NAME"_1.fastq.gz" | awk 'NR==2{print length($0);exit}'`
OPTION=""
if [ "$length_read" -le "45" ]; then
        OPTION="--segment-length 15 --segment-mismatches 1 --no-coverage-search";
fi
MIS=`awk 'NR==8{printf("%d",$1);exit}' $OUTPUT/$LIB_NAME"_insert_size_metric.tsv"`
MIS=$[MIS-length_read*2]
STD=`awk 'NR==8{printf("%d",$6);exit}' $OUTPUT/$LIB_NAME"_insert_size_metric.tsv"`

#### Tophat mapping step ####
tophat2 -p 12 -N 6  --read-gap-length 6 --read-edit-dist 6  --library-type fr-unstranded -r $MIS $OPTION --mate-std-dev $STD     --transcriptome-index
$TRANSCRIPT_INDEX --no-mixed     $BOWTIE_INDEX/Homo_sapiens_GRCh38
```

part_II.split.1.sh

```
#### HTSEQ count ####
htseq-count -m intersection-nonempty -s no  - $BOWTIE_INDEX/Homo_sapiens_GRCh38.gtf
htseq-count -m intersection-strict -s no -t exon -i exon_id  - $BOWTIE_INDEX/Homo_sapiens_GRCh38.gtf
```

As shown on figure 3 all the files were processed and the final result is a matrix of integer that contains counts of gene expressions. In figure 4 we give some average timings for a genomic file of 6GB and the total of the resources needed.

We had problems with insufficient elapsed time for some jobs. We tried without success to restart them using bowtie restart files. So we changed the tool chain parameters to speed up the computation on those libraries unable to successfully terminate within 24 hours. We also solved insufficient free space on local SSD /tmp by using instead the lustre file system scratch directory. Finally, we requested the authorisation to have 256 simultaneous running jobs in the job scheduler SLURM for our logins. We were then able to involve ~5000 cores at the same time.

We have identified some bottlenecks in our bioinformatics pipeline, in particular there are still problems in time consuming at different stages of pipeline. Many improvements that we can make using recent bioinformatics tools need to be tested in further approaches to choose the best solution.

Profiling and tuning zlib compressing library

We profiled with Vtune 2016 and found out that one third of the time was spent in compressing and deflating library files (cf. figure 5).

Function (Full)	CPU Time:... Effective T... Idle Poor	CPU Tim... Effective ... Idle Poor	Instru... Retired: Total	Instructions Retired: Self	CPI Rate: Total	CPI Rate: Self	CPU Freq... Ratio: Total	CPU Frequency Ratio: Self	Module
	99.9%	0s	100.0%	0	0.664	0.000	1.212	0.000	
org::usadellab::trimmomatic::trim::IlluminaClippingTrimmer$IlluminaLongClippingSeq::readsSeqCompare(org::usadellab::trimmomatic::fastq::FastqRecord)	24.8%	53.193s	33.5%	340,623,400,000	0.490	0.490	1.211	1.211	[Compiled Java code]
deflate_slow	32.2%	68.992s	26.6%	270,758,800,000	0.805	0.805	1.218	1.218	libz.so.1.2.3
memscan	3.2%	6.892s	6.9%	69,713,800,000	0.315	0.315	1.230	1.230	vmlinux
[obdclass]	6.0%	12.769s	4.1%	41,641,600,000	0.960	0.960	1.207	1.207	obdclass
org::usadellab::trimmomatic::trim::IlluminaClippingTrimmer$IlluminaPrefixPair::palindromeReadsCompare(org::usadellab::trimmomatic::fastq::FastqRecord, org::usadellab::trimmomatic::fastq::FastqRecord)	4.7%	10.093s	3.9%	39,301,600,000	0.816	0.816	1.224	1.224	[Compiled Java code]
inflate_fast	3.4%	7.292s	2.9%	29,811,600,000	0.736	0.736	1.161	1.161	libz.so.1.2.3
[libcfs]	2.6%	5.532s	2.4%	24,520,600,000	0.690	0.690	1.180	1.180	libcfs
org::usadellab::trimmomatic::fastq::FastqParser::next()	1.3%	2.719s	2.2%	22,284,600,000	0.378	0.378	1.194	1.194	[Compiled Java code]
sun::nio::cs::UTF_8$Decoder::decodeArrayLoop(java::nio::ByteBuffer, java::nio::CharBuffer)	0.5%	1.030s	1.3%	13,200,200,000	0.235	0.235	1.161	1.161	[Compiled Java code]
sun::nio::cs::UTF_8$Encoder::encodeArrayLoop(java::nio::CharBuffer, java::nio::ByteBuffer)	0.4%	0.813s	1.3%	12,896,000,000	0.200	0.200	1.221	1.221	[Compiled Java code]
compress_block	1.2%	2.558s	1.2%	12,199,200,000	0.665	0.665	1.222	1.222	libz.so.1.2.3
org::usadellab::trimmomatic::trim::IlluminaClippingTrimmer$IlluminaPrefixPair::calculatePalindromeDifferenceQuality(org::usadellab::trimmomatic::fastq::FastqRecord, org::usadellab::trimmomatic::fastq::...	0.9%	1.986s	1.2%	12,095,200,000	0.515	0.515	1.210	1.210	[Compiled Java code]
org::usadellab::trimmomatic::trim::SlidingWindowTrimmer::processRecord(org::usadellab::trimmomatic::fastq::FastqRecord)	0.8%	1.665s	1.2%	11,811,800,000	0.436	0.436	1.192	1.192	[Compiled Java code]
[lov]	1.4%	3.026s	1.0%	10,452,000,000	0.889	0.889	1.183	1.183	lov
org::usadellab::trimmomatic::trim::TrailingTrimmer::processRecord(org::usadellab::trimmomatic::fastq::FastqRecord)	0.5%	0.973s	0.7%	7,225,400,000	0.403	0.403	1.152	1.152	[Compiled Java code]
org::usadellab::trimmomatic::trim::LeadingTrimmer::processRecord(org::usadellab::trimmomatic::fastq::FastqRecord)	0.7%	1.469s	0.6%	6,203,600,000	0.779	0.779	1.267	1.267	[Compiled Java code]
[lustre]	1.0%	2.164s	0.6%	5,946,200,000	1.118	1.118	1.184	1.184	lustre
[lvfs]	0.7%	1.559s	0.6%	5,774,600,000	0.900	0.900	1.286	1.286	lvfs
jshort_disjoint_arraycopy	0.9%	1.949s	0.5%	5,018,000,000	1.197	1.197	1.188	1.188	[Dynamic code]
org::usadellab::trimmomatic::trim::IlluminaClippingTrimmer::processRecords(org::usadellab::trimmomatic::fastq::FastqRecord[])	0.4%	0.757s	0.5%	4,578,600,000	0.534	0.534	1.246	1.246	[Compiled Java code]
org::usadellab::trimmomatic::fastq::FastqSerializer::writeRecord(org::usadellab::trimmomatic::fastq::FastqRecord)	0.9%	1.838s	0.3%	3,322,800,000	1.782	1.782	1.242	1.242	[Compiled Java code]
fill_window	0.1%	0.267s	0.3%	3,281,200,000	0.279	0.279	1.323	1.323	libz.so.1.2.3
[osc]	0.3%	0.735s	0.2%	2,033,200,000	1.168	1.168	1.246	1.246	osc
org::usadellab::trimmomatic::fastq::FastqRecord::<init>(org::usadellab::trimmomatic::fastq::FastqRecord, int, int)	0.1%	0.296s	0.1%	1,391,000,000	0.585	0.585	1.061	1.061	[Compiled Java code]
updateBytesCRC32	0.3%	0.569s	0.1%	1,341,600,000	1.163	1.163	1.056	1.056	[Dynamic code]
follow_page	0.2%	0.335s	0.2%	1,271,400,000	0.858	0.910	1.270	1.332	vmlinux
__memset	0.4%	0.943s	0.1%	1,185,600,000	2.487	2.487	1.205	1.205	vmlinux
org::usadellab::trimmomatic::threading::BlockOfWork::call()	0.2%	0.479s	0.1%	1,131,000,000	1.306	1.306	1.188	1.188	[Compiled Java code]
inflate	0.2%	0.349s	0.1%	1,094,600,000	0.983	0.983	1.190	1.190	libz.so.1.2.3
oopDesc* PSPromotionManager::copy_to_survivor_space<(bool)0>(oopDesc*)	0.1%	0.215s	0.1%	1,053,000,000	0.590	0.590	1.112	1.112	libjvm.so
org::usadellab::trimmomatic::trim::AbstractSingleRecordTrimmer::processRecords(org::usadellab::trimmomatic::fastq::FastqRecord[])	0.1%	0.301s	0.1%	1,019,200,000	0.916	0.916	1.197	1.197	[Compiled Java code]
memcpy	0.2%	0.490s	0.1%	884,000,000	1.941	1.941	1.350	1.350	vmlinux
sun::nio::cs::StreamDecoder::read(char[], int, int)	0.2%	0.367s	0.1%	876,200,000	1.457	1.457	1.342	1.342	[Compiled Java code]
[ptlrpc]	0.2%	0.411s	0.1%	800,800,000	1.656	1.656	1.244	1.244	ptlrpc
itable stub	0.1%	0.137s	0.1%	795,600,000	0.611	0.611	1.365	1.365	[Dynamic code]
HandleMarkCleaner::~HandleMarkCleaner(void)	0.1%	0.172s	0.1%	790,400,000	0.559	0.559	0.988	0.988	libjvm.so
	6.0%	12.769s	4.1%	41,641,600,000	0.960	0.960	1.207	1.207	

Figure 5: Profiling excerpt from vtune

We tried to adapt a compression library dedicated to genomic data but it was incompatible with the software we used. We saw in [4] that zlib flavours existed. So we conducted benchmarks on GNU zlib, Intel zlib and cloudflare zlib [5] (cf. figure 6) and then opted naturally for the cloudflare one.

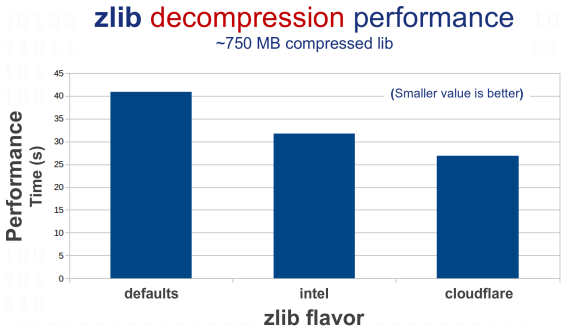

Figure 6: zlib benchmark

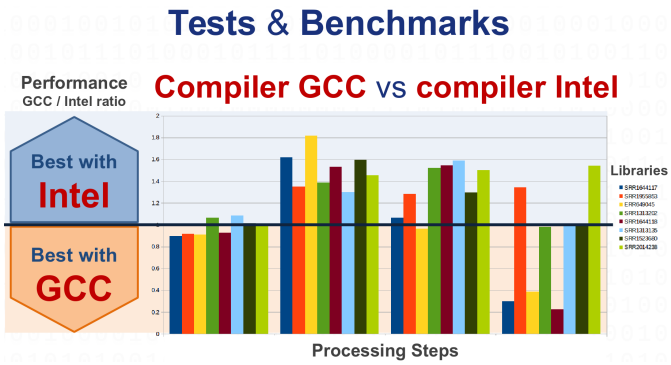

Figure 7: GNU versus Intel performances on genomic data

We then compared the efficiency of the Intel compilers versus the free ones from GNU on eight different computations. It is important matter for a SME to know by how much time it could shorten the computation while paying a license for the compiler because millions hours are in balance (one computing hour is about a few cents cost). What we saw on our benchmarks (cf. figure 7) is that the GNU compilers is almost as good as the Intel compiler for those kind of algorithms on genomic sequences. The throughput, i.e. the number of compressed bytes per second is shown on the figure 8 below for mostly Intel and some GNU runs of tophat.

Conclusion

Many challenges were encountered and solved during the project: downloading the very large amount of data, installation and settings of softwares in the cluster, the optimization by the choice of compiler and the packages and constraints in jobs duration and limitations.
An installation of the code with all its third party libraries were done and used through SLURM jobs. It represented a huge amount of data computed and required the use of High performance computing (HPC) on the cluster Occigen (GENCI/CINES): 120 TB of compressed data downloaded and 1.2 million core hours consumed.
In the next step, we plan to explore of the matrix which was gained thanks to the MaRS project by implementing an advanced search tool to query the matrix in order to highlight biomarkers. Finally, we could extend this work to other species such as mouse widely used for validating future human's treatments.

References

[1] Engström PG et al. (2013) *Systematic evaluation of spliced alignment programs for RNA-seq data*. Nature Methods. 10 (12), 1185-1191.
[2] https://github.com/samtools/samtools/issues/370
[3] http://www.intel.com/content/www/us/en/embedded/training/ia-deflate-decompression-paper.html
[4] http://www.htslib.org/benchmarks/zlib.html
[5] https://github.com/cloudflare/zlib/
[6] http://www.ebi.ac.uk/ena
[7] https://www.ncbi.nlm.nih.gov/books/NBK242625/#_Aspera_Transfer_Guide_BK_Aspera_
[8] Cole Trapnell et al. (2012) *Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks.* Nature Protocols 7: 562-578.
[9] Trimmomatic: https://www.ncbi.nlm.nih.gov/pubmed/24695404
[10] Bowtie http://genomebiology.biomedcentral.com/articles/10.1186/gb-2009-10-3-r25
[11] Tophat2 http://genomebiology.biomedcentral.com/articles/10.1186/gb-2013-14-4-r36
[12] HTSeq https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btu638
[13] Samtools: https://www.ncbi.nlm.nih.gov/pubmed/19505943
[14] Picard Tools: http://broadinstitute.github.io/picard

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EU's Horizon 2020 research and innovation programme (2014-2020) under grant agreement 653838.