Using ARC-based grids for NGS read mapping – Grid interface for BWA

Kimmo Mattila*
CSC - IT Center For Science, Espoo, Finland

* corresponding author (Kimmo.Mattila@csc.fi)

Received 3 February 2012; **Accepted** 2 May 2012; **Published** 15 October 2012
Competing Interests: none

Abstract

This technical note describes a command-line Grid interface for the Burrows-Wheeler Aligner (BWA) read-mapping program. With this interface, BWA jobs can easily utilise Advanced Resource Connector (ARC) middleware-based Grid environments. The interface automatically splits the mapping task into sub-tasks that are executed in parallel in a computing Grid. This approach can significantly speed up the read-mapping process.

Availability: **http://www.csc.fi/english/research/sciences/bioscience/programs/BWA**

Introduction

Aligning large sets of short sequences, reads, to a reference sequence set is an essential step in many Next Generation Sequencing (NGS)-based analysis workflows. At the moment, several read-mapping programs are available to perform this alignment task: e.g., tools like Bowtie (Langmead et al, 2009), Burrows-Wheeler Aligner (BWA) (Li and Durbim, 2009) and Maq (Li *et al.* 2008). The resulting alignment files are further analysed with different tools depending on the case.

Different mapping tools apply different algorithms and heuristics to do the alignment. However, the common trend seems to be that tools that produce higher mapping accuracy also require more computing power. Even though the mapping tools are extremely fast compared to the previous generation of sequence alignment tools, mapping hundreds of millions of reads against the reference genome can require weeks of computing time on a normal desktop computer.

One way to speed up the mapping tasks, in addition to parallel computing or special hardware, is to use Grid computing. The read-mapping tasks suit well to Grid computing, as the analysis task can normally be divided into numerous sub-tasks that can later be merged back together. In this note, we present a command-line Grid interface for the commonly used BWA. The goal of the interface is to allow Linux users to utilise Advanced Resource Connector (ARC) (Ellert *et al*, 2007) – middleware-based computing Grids for NGS mapping jobs – without any knowledge about the Grid middleware itself.

The Grid interface of BWA is currently in use at the servers of CSC, IT Center for Science, but in principle it could be installed on any Linux server. Similar interfaces are also available for other tools, like the SHRiMP read-mapping tool (David M et al. 2011).

Methodology

The BWA Grid submission command, grid _ bwa, executes the following five basic steps: 1) checking input; 2) indexing the reference; 3) splitting the mapping task into sub-jobs; 4) executing sub-jobs in the Grid; 5) collecting the results. These steps are discussed more in detail below.

1. Checking the environment, input data and parameters

In the beginning, the *grid_bwa* command checks that the input files have the expected file types (fasta for the reference genome, fastq for the read files). In the case of paired-end analysis, the two query files are checked to have equal amounts of sequences. Even though checking the number of sequences in a file is a simple task, it may take tens of minutes if the input files contain hundreds of millions of sequences.

2. Indexing the reference and storing the indexes to a central repository

Indexing the reference genome or sequence set is the first step of the BWA analysis. In normal usage, this indexing is done with a separate command that launches the actual mapping task. In the *grid_bwa* command, the indexing is integrated as an automatic part of the alignment command. Further, the *grid_bwa* first checks if indices for the reference genome already exist in the user's personal data repository (storage element) in the Grid environment. If indices are not found, they are computed and stored in the Grid repository.

3. Splitting the mapping task into sub-jobs

Splitting the jobs is straightforward, but for large input files it can take several hours. The resulting query subsets are copied to sub-job-specific temporary directories, after which a grid job description file and corresponding job execution file are generated for each sub-job directory. The *grid job description files* use the ARC XRSL format. A *job-description file* contains information about input files that need to be copied to the remote execution site and the output files that should be retrieved when the job is finished. The job-description file also contains the execution time, memory and software requirements of the job. In the case of BWA jobs, fixed 24h and 8GB reservations are used.

The *job execution file* is a shell script that contains commands that are needed to: 1) unpack the input data; 2) run the actual mapping task; 3) post-process the mapping results.

In principle, large mapping tasks could be split into millions of sub-jobs. In practice, splitting the task into more than a few thousands of sub-jobs is not feasible, because managing large amounts or sub-directories is inefficient. Further, the optimal execution time for sub-jobs is in the range of a few hours. In the case of very short sub-jobs, the overhead caused by the job pre- and post-processing can become relatively large. On the other hand, very long sub-jobs often have to wait longer in the batch queues, which again increases the throughput time of the jobs. By default, the command splits the job into about 300 sub-jobs. The number of sub-jobs can be modified with the option *-nsplit.* For small jobs, the job-splitting and result-merging steps can be skipped by setting *-nsplit 1.*

4. Executing the sub-jobs in grid

When the job-splitting phase is ready, the job-specific temporary directory contains from tens to a few thousands of sub-directories, each containing all the data for one ARC middleware-based Grid job. *grid_bwa* automatically checks which computing resources (*i.e.*, clusters connected to the Grid) are available for the user. To submit the jobs to the remote clusters, a job-manager tool, written at CSC, is used. This job manager tries to optimise the usage of the Grid environment. It follows how many jobs are queueing in the clusters, and sends more jobs only when there are free resources available. The job manager keeps track of the executed sub-jobs and starts sending more jobs to those clusters that execute the jobs most efficiently. As some of the clusters may not work properly, part of the jobs may fail for technical reasons. If this happens, the failed sub-jobs are resubmitted to other clusters three times before they are considered as 'failed' sub-jobs. When a job finishes successfully, the job manager retrieves the result files from the Grid to the sub-job-specific directory at the local computer. In the beginning, only a few jobs run, but gradually more and more jobs get to the execution phase and, after a while, there can be hundreds of BWA tasks being executed at the same time, depending on the amount of suitable Grid resources.

5. Collecting the results

When all the sub-jobs are ready, the alignment files are merged into one indexed bam file using the SAMtools package (Li *et al.* 2009). The query sequences from jobs that have not produced a result file are collected into a separate file. For example, if some query subsets require exceptionally long execution time, they may fail because they exceeded the computing-time limit. Such failed query sets can be processed by resubmitting them with the *grid_bwa* command. During

this second iteration, the number of query sequences in each sub-job is normally so small that all the sub-jobs get processed quickly enough.

Performance

Evaluation of the performance of Grid based tools is difficult. In principle, the more Grid resources you have available, the more sub-jobs you can execute in the same time and the faster all the sub-jobs are finished. However, as the general work-load in the Grid environments varies from time to time, so does the computing power that the Grid job can utilise. Pre- and post-processing can also take some time and, because of this, small alignment tasks that do not require more than few hours of computing time do not actually benefit from splitting the job. For larger jobs, utilising distributed Grid computing enables running in one night tasks that would take weeks on a normal desktop computer. Table 1 shows statistics for a *grid_bwa* run, where 264 million read pairs (2*101 bp) were mapped against a pre-indexed human genome, using the default paired-end mapping parameters of BWA. *grid_bwa* split the task into 301 sub-jobs that were processed with a small test cluster (based on 2.67 GHz Intel Xeon CPUs). The average execution time for one sub-job, executed with six computing cores, was about 40 minutes. The environment used in this test was able to process 16 sub-jobs simultaneously (reserving a total of 96 computing cores). With these resources, the mapping task was executed in 14 hours. In comparison, running the same analysis as one job using six cores of an 2.26 GHz Intel Xeon X7560-based server takes about 34 hours (including the conversion of the results into indexed bam files). Thus, in many cases, the actual speed-up gained by using the Grid interface is only moderate. However, in this comparison, we are ignoring the time required to copy the input data to the computing cluster and the time that the job has to wait in the batch queue before the execution starts.

Table 1. Wall-clock times used by the different steps of a grid_bwa run. In the sample task, 264 million reads pairs (read length = 101 bp) were mapped against a pre-indexed human genome using BWA default parameter.

Step	Duration
Checking input and parameters	36 min
Checking pre-calculated indexes	1 min
Splitting the job into 301 subjobs	2h 11 min
Executing the 301 sub-jobs in the grid	5h 43 min
Merging results	4h 26 min
Total	**13h 57 min**

Command line interface

The BWA Grid-submission command, *grid_bwa*, is designed to look much like the normal BWA command. The Grid related tasks are all integrated inside the command-line interface, so it is enough for a user just to create the Grid-proxy certificate on the client machine, before executing the job-submission command. Typically, the certificate is generated with ARC command *arc proxy*.

In normal BWA usage, the basic command to map reads in file *reads.fq* to reference genome *genome.fa*, is:

```
bwa aln genome.fa reads.fq > aln_sa.sai
```

With grid_bwa, the same task could be executed as a distributed Grid job with command:

```
grid_bwa aln -query reads.fq -ref
genome.fa -out aln.bam
```

The two major differences between the normal BWA command and the Grid version is that: 1) in the Grid version, the query, reference and output file must be defined with explicit options (*-query, -ref* and *-out*); 2) the .sai formatted BWA output files are automatically converted into bam format using *bwa samse* or *bwa sampe* commands and the SAMtools package.

All other BWA parameters can be defined with normal command-line options. For example, adjusting seed length to 24, and defining bar-code

```
grid_bwa aln -quey genome.fa -ref reads.
fq -out aln.bam -l 24 -B 6
```

In the case of paired-end data, *grid_bwa* has yet another difference. Normally, paired-end alignment is computed using two *bwa aln* commands, from which the results are combined with the *bwa sampe* command. For example:

```
bwa aln genome.fa reads1.fq > aln1.sai
bwa aln genome.fa reads2.fq > aln2.sai
bwa sampe genome.fa aln1.sai aln2.sai
reads1.fq reads2.fq > aln.sam
```

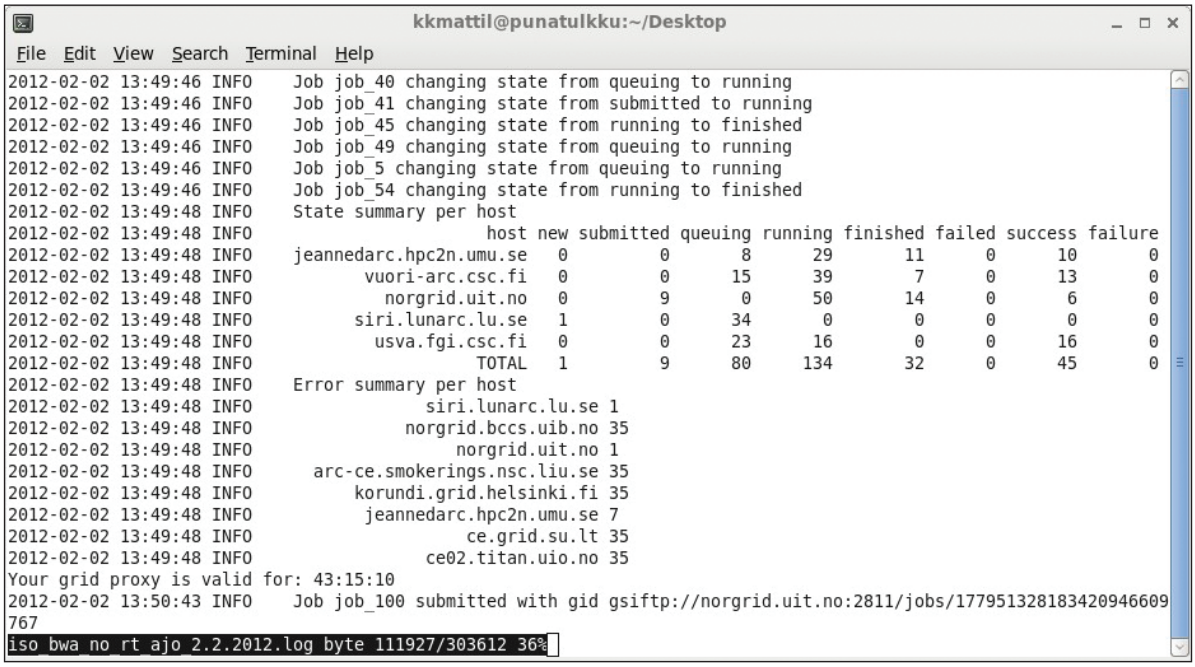

Figure 1. Screen capture of a *grid _ bwa* run. The screen shows the status of a mapping job that is being processed simultaneously in five clusters: two clusters in Finland, two in Sweden and one in Norway. The mapping task was split into 301 sub-tasks. 45 of these sub-tasks are already successfully completed, 32 wait for result retrieval, 134 are currently being executed, 80 are queueing in the clusters, 9 are submitted to the grid and one job waits to be submitted.

In the case of *grid _ bwa*, all these steps are executed using just a single command:

```
grid _ bwa aln -query1 reads1.fq -query2
reads2.fq -ref genome.fa -out aln.bam
```

Adding the option *-query2* defines that a paired-end analysis is executed, and that the post-processing is done with the *bwa sampe* command instead of *bwa samse*.

Just like in the previous example, *bwa aln* para-meters can be defined for this command with normal *bwa aln* options. You can also define parameters for the *bwa sampe* command. This is done with options *-sampe _ a* (corresponding to the *bwa sampe* option *-a*), *-sampe _ o*, *-sampe _ n*, *-sampe _ N*, and *-sampe _ r*.

For example, executing a paired-end alignment task, where the seed length is 24 and, in the post-processing state, the maximum insert size is 400 (bwa sampe -a 400), can be defined with command line:

```
grid _ bwa aln -query1 reads1.fq -query2
reads2.fq -ref genome.fa -out aln.bam -l
24 -sampe _ a 400
```

Once the command is launched, it starts printing out log information about the progress of the job (Figure 1). For longer jobs, it is reasonable to forward the *grid _ bwa* output to a separate file and run the command as a background process. This way you can log out and return later on to check how the job has progressed. For example:

```
grid _ bwa aln -query1 reads1.fq -query2
reads2.fq -ref genome.fa -out aln.bam -l
24 -sampe _ a 400 > log.txt &
```

After launching the job, the *grid _ bwa* command must be kept running until all sub-jobs are processed, and the cleaning and post-processing tasks are done.

Accessing the tool

1. Installing the grid _ bwa command

At the moment, this Grid-job submission tool for BWA is in use only in the servers of CSC. So, the easiest way to use these tools is to apply for a CSC user account and join to the Finnish Grid Initiative[1] or the bioscience virtual organisation of Nordic Data Grid Facility (NDGF)[2]. However, grid _ bwa is just small set of tcsh and python scripts. It can be installed on any Linux machine that has the

1 http://www.csc.fi/english/research/Computing _ services/grid _ environments/fgi

2 www.ndgf.org/

following components: 1) an ARC middleware client; 2) BWA; 3) SAMtools; 4) Python 2.6 or later. Local installation of the EMBOSS (Rice et al., 2000) package is also recommended, as the grid _ bwa uses EMBOSS, if it is available, to check that the input files are in the correct format. The scripts, which form the grid _ bwa tool, can be downloaded from the BWA instruction page of CSC[3]. Further, if you are using other than the FGI Grid environment, you should ask your Grid administrators to install BWA and SAMtools runtime environments onto your ARC clusters. Installation instructions and runtime environment examples can be found form the NDGF runtime environment registry[4].

2. Grid certificates and Virtual Organisations

In addition to the Grid-submission command, the user must have access to some ARC middleware-based Grid environments: e.g., the Grids of NDGF or FGI. These Grid environments, like most middleware-based Grid environments, use personal X.509 certificates for user authentication. Certificates are granted by a Certification Authority (CA), which acts as a trusted third party to ensure that the identity information is valid. For example, Nordic academic Grid users can use the Trans-European Research and Education Networking Association (TERENA)[5] as the certification authority. The certificate is first installed on your Web browser, where it can be used to automatically authenticate you to a Website. You also need to install the certificate on the computing sever that is used to launch the Grid jobs.

Once a researcher has a Grid certificate, he/she can apply for membership of a Virtual Organisation (VO). A VO refers to a group of users or institutions that utilise some Grid resource according to a set of resource-sharing rules and conditions. Typically, VOs focus on some specific branch of science and/or geographic region. A VO is also linked to a distinct set of Grid resources.

At the moment, the tools discussed here can only be used by the members of the Finnish FGI VO however, if needed, the tools can be made available for the NDGF Bio VO, which is open for all Nordic researchers. Researchers working in Finland can join the FGI VO using server[6]. Nordic researchers can join the NDGF Bio VO using server: https://voms.ndgf.org:8443/voms/nordugrid.org/Siblings.do.

Conclusions

The Grid-submission tool described here demonstrates how Grid middleware commands can be embedded in Linux command-line scripts, so that end-users can utilise Grid resources without any knowledge of the Grid middleware. The Grid interface described here performs NGS-read mapping, but similar automatic Grid-submission tools can be set up for other tools too. At CSC, we also provide similar interfaces for SHRiMP, BLAST (Altschul *et al.*,1990), AutoDock (Morris *et al.*, 1998), HHserch (Söding, 2005) and MatLab (MathWorks Inc.).

The Grid interface described here is based on the ARC middleware. In principle, it could also be converted to use other Grid middlewares, but this would require large modifications to the job-managing and data-transport parts of the tool. Further, there already exist BWA implementations that utilise similar approaches for distributing BWA tasks to gLite and BOINC middleware-based grids[7](Luyf *et al.*, 2010).

Acknowledgements

Olli Tourunen is acknowledged for creating the ARC job-manager. This work has been funded by the EGI-Inspire project.

References

1. Altschul S, Gish W, Miller W, Myers E, Lipman D (1990) Basic local alignment search tool *Journal of Molecular Biology* **215** (3), 403–410. doi:10.1016/S0022-2836(05)80360-2
2. David M, Dzamba M, Lister D, Ilie L, Brudno M. (2011) SHRiMP2: sensitive yet practical SHort Read Mapping. *Bioinformatics.* **7**,1011-1012. doi:10.1093/bioinformatics/btr046
3. Ellert M, Grønager M, Konstantinov A, Kónya B, Lindemann J *et al.* (2007) Advanced Resource Connector middleware for lightweight computational Grids. *Future Generation Computer Systems* 23, 219-240. doi:10.1016/j.future.2006.05.008
4. Langmead B, Trapnell C, Pop, Salzberg SL (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology* **10**, R25 doi:10.1186/gb-2009-10-3-r25

3 http://www.csc.fi/english/research/sciences/bioscience/programs/BWA
4 http://gridrer.csc.fi/
5 https://tcs-escience-portal.terena.org/
6 https://voms.fgi.csc.fi:8443/vomses/
7 https://appdb.egi.eu/?#p=L2FwcHMvZGV0YWlscz9pZD02NDE=

5. Li H and Durbin R (2009) Fast and accurate short read alignment with Burrows-Wheeler Transform. *Bioinformatics* **25**, 1754-1760. doi:10.1093/bioinformatics/btp698
6. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J *et al.* (2009) The Sequence alignment/map format and SAMtools. *Bioinformatics* **25**, 2078-2079. doi:10.1093/bioinformatics/btp352
7. Li H, Ruan J, Durbin R (2008) Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Re.*, **18**, 1851-1858. doi:10.1101/gr.078212.108.
8. Luyf AC, van Schaik BD, de Vries M, Baas F, van Kampen AH, Olabarriaga SD (2010)Initial steps towards a production platform for DNA sequence analysis on the grid. *BMC Bioinformatics* **11**, 598-607. doi:10.1186/1471-2105-11-598
9. MathWorks Inc. Natick, Massachusetts, U.S.A
10. Morris, GM, Goodsell DS, Halliday, RS,
11. Huey R, Hart W, Belew RK, Olson AJ (1998), Automated Docking Using a Lamarckian Genetic Algorithm and and Empirical Binding Free Energy Function *J. Computational Chemistry*, **19**, 1639-1662. doi:10.1002/(SICI)1096-987X(19981115)19:14<1639::AID-JCC10>3.0.CO;2-B
12. Rice P, Longden I. and Bleasby A. (2000) EMBOSS : The European Molecular Biology Open Software Suite. Trends in Genetics **16**, 276-277. doi:10.1016/S0168-9525(00)02024-2