Porting and Benchmarking of BWAKIT pipeline on Open POWER architecture

Nagarajan Kathiresan[1], Rashid Al-Ali[1], Puthen Jithesh[1], Zaid Al-Ars[2]

[1] Sidra Medical and Research Center, Biomedical Informatics, Research Division, Doha, Qatar
[2] Delft University of Technology, Computer Engineering Lab, Netherlands

Abstract:
Next Generation Sequencing (NGS) technology produces large volume of genome data, which gets processed using various open source Bioinformatics tools. The configuration and compilation of some of the bioinformatics tools (e.g. BWAKIT, root) are challenging, requiring application porting for some architectures (e.g. IBM Power). Moreover, the application porting should not change the semantics of the program and output generated from different architectures should be similar across different architectures. Burrows-Wheeler Aligner (BWA) is the most popular genome mapping application used in BWAKIT toolset. This BWAKIT provides pre-compiled binaries for x86_64 architecture and end-to-end solution for genome mapping. In this paper, we show how to port various pre-built application binaries used in BWAKIT into OpenPOWER architecture and execute the BWAKIT pipeline successfully. Additionally, we show the validation of the output results on OpenPOWER to confirm the successful porting of BWAKIT.

Keywords: BWAKIT, Genome mapping, Burrows-Wheeler Aligner, parallelization, scalability, efficiency, POWER architecture

NOTE: Many bioinformatics tools are ported to OpenPOWER and maintained at "biobuilds.org". Please refer to "biobuilds.org" first to check if your code is already ported.

Parallelization of Burrows-Wheeler Aligner (BWA)
The Burrows-Wheeler Aligner (BWA) [1] and Genome Analysis ToolKit (GATK) [2] are the two major applications for next generation sequencing workflow. The BWA is a popular genome alignment tool and it consists of three algorithms: BWA-backtrack, BWA-SW and BWA-MEM. These algorithms support (i) thread parallelization (ii) data-parallelization and (iii) data-parallel with concurrent execution [3]. The thread parallelization is controlled by the number of pthreads allocated by using `-t #threads` option. For the data-parallelization, the provided genome data is divided (called as chunks) into number of physical cores in the system using third-party tools (e.g. split) and execute multiple instance of BWA on every chunk of the genome data and merge the partial results in the same order of split data to get a final result [4]. Using the data-parallel approach with concurrent execution, the chunks of genome data are distributed to the available CPUs and execute the BWA on every chunk concurrently with number of "Cores per CPU" set to be the same as the available CPU threads. Finally, we merge the partial results in the same order of split data into the final result.

In this document, we will use the thread parallelization approach to run our experiments and measure the performance of the tools on a single server node.

BWAKIT pipeline implementation

DNA sequence mapping has a couple of steps. First, the genome sequence data (stored in the form of FASTQ files) are mapped to a reference genome resulting into a file with the Sequence Alignment/Map (SAM) format or BAM (Binary version of SAM) format. Then, merging, sorting and marking the duplicates is carried out. BWAKIT provides an end-to-end pipeline solution for genome mapping using a set of predefined scripts. The pipeline scripts use 4 open-source applications: (i) SAMtools, (ii) Trimadap, (iii) BWA-MEM and (iv) Samblaster, that are executed one after the other in the fashion described in Figure 1.

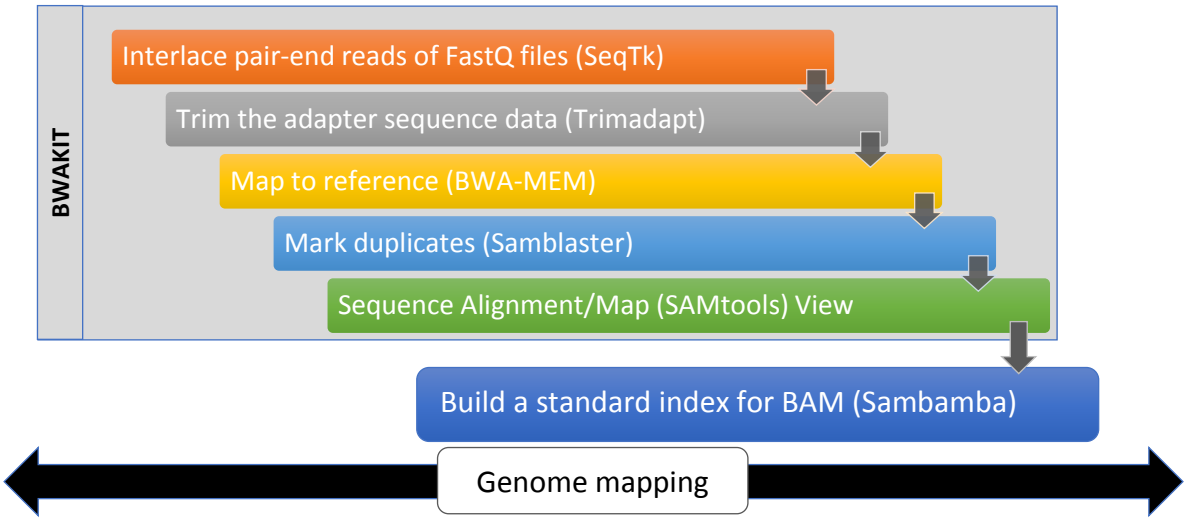

Figure 1. *BWAKIT Pipeline*

SAMtools is used for sequence alignment and mapping. Trimadap is used for trimming the adapter sequence from the FASTQ data. The latest version of the BWA algorithm is BWA-MEM, which provides fast and accurate alignment of genome sequence, supporting long-query and split-alignment. Samblaster helps in marking duplicates as removing duplicates is important to mitigate the effects of polymerase chain reaction (PCR) amplification, and reducing the number of reads to be processed during variant discovery. The K8 and V8 are the Javascript shell (K8) and Google V8 Javascript engine libraries respectively. These are all pre-request to run bwakit run-script but it's not part of the workflow steps. These open-source applications are required to port on the OpenPOWER architecture to execute the BWAKIT pipeline.

List of genome application/tools used

The list of genome application/tools for this BWAKIT porting on POWER architecture is summarized in the Table 1.

# No.	Genome application/ tools	Version	Download link
1	BWAKIT	0.7.15	https://github.com/lh3/bwa/tree/master/bwakit
2.	SeqTk	1.2	https://github.com/lh3/seqtk
3.	Samtools	1.3	http://www.htslib.org
4.	Trimadap	0.1	https://github.com/lh3/trimadap
5.	Samblaster	0.1.23	https://github.com/GregoryFaust/samblaster
6.	K8	0.2.3-r67	https://github.com/attractivechaos/k8
7.	V8	3.14.5.9	https://github.com/v8/v8
8.	BWA	0.7.15	https://github.com/lh3/bwa
9.	Modified source codes and patches for BWAKIT porting	V1.0 (yet to update – required approval)	https://github.com/sidratools/BWA_in_Power8

BWAKIT porting on Open POWER

Step 1: Download the BWAKIT from source forge website.

```
wget https://sourceforge.net/projects/bio-bwa/files/bwakit/bwakit-0.7.15_x64-linux.tar.bz2
```

Step 2: Uncompressing the source code and remove (or backup) pre-build x86_64 binaries which includes seqtk, samtools, trimadap, samblaster, k8 and bwa.

```
tar -xjvf bwakit-0.7.15_x64-linux.tar.bz2
cd bwa.kit

mv seqtk seqtk.linux
mv samtools samtools.linux
mv trimadap trimadap.linux
mv samblaster samblaster.linux
mv k8 k8.linux
mv bwa bwa.linux
```

Step 3: Prepare for porting and configure, build and install of above pre-build x86_64 binaries into OpenPOWER architecture.

```
mkdir Build.PowerBinaries
```

```
cd Build.PowerBinaries
```

Step 3a: Download a seqtk source code and compile for OpenPOWER. The seqtk is a toolkit for processing sequences in FASTA/Q formats. It is a fast and lightweight tool for processing sequences in the FASTA or FASTQ format. It seamlessly parses both FASTA and FASTQ files which can also be optionally compressed by gzip.

Step 3a(i): Clone a latest seqtk source code.

```
git clone https://github.com/lh3/seqtk.git
cd seqtk/
```

Step 3a(ii): Compile the source code

```
make
```

Step 3a(iii): Create a symbolic link of OpenPOWER seqtk binary into BWAKIT pipeline

```
ln -sf ~/bwa.kit/Build.PowerBinaries/seqtk/seqtk ~/bwa.kit/.
```

Step 3a(iv): Verify the working status of seqtk binary on OpenPOWER.

```
cd ../Build.PowerBinaries
./../bwa.kit/seqtk
     Usage:   seqtk <command> <arguments>
     Version: 1.2-r101-dirty
```

Step 3b. Download a samtools source code and compile for OpenPOWER. The samtools depend on high-throughput sequencing data format (HTSlib) library. The pre-requisite to install HTSlib is data compression library (zlib). To build all these dependencies, here are the steps which we followed:

Step 3b(i): Create a samtools directory to build samtools and its associated dependencies.

```
mkdir samtools
```

Step 3b(ii): Download the latest samtools source code and uncompress.

```
wget https://github.com/samtools/samtools/archive/1.3.tar.gz
tar –xzvf 1.3.tar.gz
cd samtools-1.3
```

Step 3b(iii): Download the latest HTSlib and uncompress, assuming zlib library is available.

```
wget https://github.com/samtools/htslib/archive/1.3.tar.gz
tar –xzvf 1.3.tar.gz
cd htslib-1.3/
make
```

Step 3b(iv): Ensure the HTSlib library is compiled properly and it will generate `libhts.so` shared library and `bgzip` & `tabix` binaries.

```
ls libhts* bgzip tabix
lrwxrwxrwx 1 nkathiresan nkathiresan       9 Mar 20 07:30 libhts.so.1 -
> libhts.so
-rwxrwxr-x 1 nkathiresan nkathiresan 2649544 Mar 20 07:30 libhts.so
-rw-rw-r-- 1 nkathiresan nkathiresan   66664 Mar 20 07:30 bgzip.o
-rw-rw-r-- 1 nkathiresan nkathiresan   41736 Mar 20 07:30 htsfile.o
-rwxrwxr-x 1 nkathiresan nkathiresan 1944664 Mar 20 07:30 bgzip
-rwxrwxr-x 1 nkathiresan nkathiresan 2366968 Mar 20 07:30 htsfile
-rw-rw-r-- 1 nkathiresan nkathiresan  122928 Mar 20 07:30 tabix.o
drwxrwxr-x 5 nkathiresan nkathiresan      86 Mar 20 07:30 .
-rwxrwxr-x 1 nkathiresan nkathiresan 2422696 Mar 20 07:31 tabix
```

Step 3b(v): Configure, build and install samtools by modifying the Makefile and config.mk

```
cd ..
cp Makefile Makefile.orig
cp config.mk config.mk.orig
```

In Makefile, modify the prefix into ${PWD}, resulting in the following diff output.

```
diff Makefile Makefile.orig
42c42
< prefix      = ${PWD}
---
> prefix      = /usr/local
```

Modify the config.mk file to point the HTSDIR directory to ./htslib-1.3, resulting in the following diff output.

```
$ diff config.mk config.mk.orig
32c32
< HTSDIR = ./htslib-1.3
---
> HTSDIR = ../htslib
```

Step 3b(vi): Compile the samtools, create a symbolic link to BWAKIT and ensure it is working fine on OpenPOWER.

```
make
cd ../../
ln   -sf   ~/bwa.kit/Build.PowerBinaries/samtools/samtools-1.3/samtools
~/bwa.kit/.
./bwa.kit/samtools --version
      samtools 1.3
      Using htslib 1.3
      Copyright (C) 2015 Genome Research Ltd.
```

Step 3c: Porting trimadap on OpenPOWER: trimadap is a small tool to trim the adapter sequences which is fast for Illumina reads. This trimadap is built with multi-threading enabled support.

Step 3c(i): Download the latest trimadap source code.

```
git clone  https://github.com/lh3/trimadap.git
cd trimadap
```

Step 3c(ii): The query immutable source code (`ksw.c`) needs to be modified to run on POWER. Hence, the modified source code "`ksw-trimadap.c`" and its associated header file "`vec128int.h`" are replaced and copied from IBM directory into the original code.

```
cp IBM/ksw-trimadap.c ksw.c
cp vec128int.h .
```

Step 3c(iii): Compile the source code and create a symbolic into the BWAKIT directory.

```
make
ls –lrta trimadap*
trimadap-mt
cd ../
ln        –sf        ~/bwa.kit/Build.PowerBinaries/trimadap/trimadap-mt
~/bwa.kit/trimadap
```

Step 3d: Porting samblaster on OpenPOWER: samblaster is a tool to mark duplicates and extract discordant and split reads from sam files.

Step 3d(i): Download the latest samblaster source code.

```
git clone git://github.com/GregoryFaust/samblaster.git
```

Step 3d(ii): Compile the source code and create a symbolic link into the BWAKIT directory

```
cd samblaster
make
ln –sf ~/bwa.kit/Build.PowerBinaries/samblaster/samblaster ~/bwa.kit/.
```

Step 3e: Porting K8 on OpenPOWER: K8 is a Javascript shell based on Google's V8 Javascript engine. K8 is implemented in C++ programming language and it depends on zlib and V8. V8 is Google's open source high-performance JavaScript engine written in C++. K8 requires v8-3.16 (as per the readme). The IBM runtime technologies ported and supported v8-3.14 version for POWER architecture. We could build K8 with v8-3.14-ppc version with some minor changes in the V8 API calls in k8.cc source code.

Step 3e(i): Download the v8-3.14ppc version from IBM runtime technologies.

```
git clone https://github.com/ibmruntimes/v8ppc.git
```

Step 3e(ii): Checkout the trunk v8-3.14 and compile the V8 source code

```
cd v8ppc/
git checkout -b 3.14 remotes/origin/3.14-ppc
make dependencies
make ppc64.release
cd ..
```

Step 3e(iii) Download k8 source code

```
git clone https://github.com/attractivechaos/k8.git
mv k8/* . ;
```

Step 3e(iii): Apply the k8.cc patch to support v8-3.14 version as an alternate API that is flexible and supportable for V8.

```
patch –p < k8_patch.txt
```

Step 3e(iv): Compile K8 with v8-3.14 runtime library and create a symbolic link into BWAKIT directory.

```
cd v8ppc/
g++ -O2 -Wall -o k8 -Iinclude ../k8.cc -lz `find out -name "libv8_*.a"
| grep -v nosnap` -lpthread
ln –sf ~/bwa.kit/Build.PowerBinaries/v8ppc/k8 ~/bwa.kit/k8
```

Step 3f: Porting BWA on OpenPOWER: BWA is a software package for mapping the sequences against a large reference genome, such as the human genome. The query immutable source code (`ksw.c`) needs to be modified to run on POWER. The `ksw.c` and associated POWER hardware specific header files (`vec128int.h`) need to be modified/copied to get successful compilation.

Step 3f(i): Download and uncompressed the latest source code from source forge repository.

```
https://sourceforge.net/projects/bio-bwa/files/bwa-0.7.15.tar.bz2
tar -xjvf bwa-0.7.15.tar.bz2
cd bwa-0.7.15/
```

Step 3f(ii): Replace the original source code of `ksw.c` into the modified version. Also, copy the associated header file (`vec128int.h`) which is specific to POWER architecture.

```
mv ksw.c ksw.c.orig
cp IBM/ksw.c .
cp IBM/vec128int.h .
```

Step 3f(iii): Compile and create a symbolic link of BWA binary into BWAKIT directory.

```
make
ln -sf ~/bwa.kit/Build.PowerBinaries/bwa-0.7.15/bwa ~/bwa.kit/.
```

Dataset used for benchmarking:

- Reference genome: The Human genome reference build `hs37d5.fa` is used and can be downloaded from 1000 genome project:
  `ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/phase2_reference_assembly_sequence/hs37d5.fa.gz`
- Genome dataset for benchmarking: The GCAT (Genome Comparison & Analytics Testing) provides benchmarking genomic datasets [6]. We used datasets (`gcat set 041`) that are 100 - 150bp, paired-end and large INDELs. These datasets are used to validate and optimize the OpenPOWER system.

System architecture used for application porting and benchmarking:

The IBM 8247 is a Power System S824L server with NVIDIA graphics processing unit (GPU) and the first Power server designed to accommodate high-wattage adapters such as NVIDIA GPUs. This IBM Power server (8247-42L) supports two processor sockets, offering 20-core 3.42GHz configurations in a 19-inch rack-mount, 4U (EIA units) drawer configuration. All the cores are active. The Power S824L server supports a maximum of 16 DDR3 CDIMM slots. Memory supported are 32 GB, and 64 GB and run at speeds of 1600 Mbps, allowing for a maximum system memory of 1024 GB. In addition, the Power S824L is equipped with at least one NVIDIA GPU, allowing maximum of two. The complete summary of the benchmarked POWER system configuration is shown in Table 2.

Table 2. OpenPOWER system architecture details

Processor	0 - 159
CPU	POWER8E (raw), altivec supported
clock	2061.000000MHz
Revision	2.1 (pvr 004b 0201)
Timebase	512000000
Platform	PowerNV
Model	8247-42L
Machine	PowerNV 8247-42L
Firmware	OPAL v3

Benchmarking BWAKIT on OpenPOWER

- Generate BWA index for the reference human genome data.

```
time -p ~/bwa.kit/bwa index -p ~/ref/hs37d5.fa ~/ref/hs37d5.fa
```

- Genome mapping using multiple CPUs:

We provided the below sample script to run the BWA mapping on multiple CPUs. To execute the scalability of benchmarking experiment, the `NO_CORES=<#>` will be changed and the BWA mapping is executed multiple times on the multi-CPUs.

```
export LC_ALL=en_US.UTF-8
export BWAKIT=/home/nkathiresan/bwa.kit
export NO_CORES=160
export prefix=gcat_set_041
export BAMDIR=/home/nkathiresan/test
mkdir -p ${BAMDIR}/${prefix}
export REF=/home/nkathiresan/ref/hs37d5.fa
export INDIR=/home/nkathiresan/input/gcat_set_041

#CMD="$BWAKIT/run-bwamem -t${NO_CORES} -R
\"@RG\tID:${prefix}\tLB:${prefix}\tSM:${prefix}\tPL:ILLUMINA\" -aHds -o
out $REF ${INDIR}/${prefix}_1.fastq ${INDIR}/${prefix}_2.fastq ; "

CMD="$BWAKIT/run-bwamem -t${NO_CORES} -R
\"@RG\tID:${prefix}\tLB:${prefix}\tSM:${prefix}\tPL:ILLUMINA\" -o
${BAMDIR}/${prefix} $REF ${INDIR}/${prefix}_1.fastq.gz
${INDIR}/${prefix}_2.fastq.gz ; "
echo ${CMD} > runme.sh
chmod +x runme.sh
time -p ./runme.sh | sh ;
```

The POWER system provides the simultaneous multi-threading (SMT) option for better performance. We benchmarked the scalability numbers with CPU=1, 10 (Number of sockets=1) and 20 (Number of sockets=2). Additionally, the scalability benchmarking is extended to simultaneous multi-threading options (SMT=2, 4 and 8), where CPUs=40 (SMT=2 is enabled), CPUs=80 (SMT=4 is enabled) and CPUs=160 (SMT=8 is enabled) for BWA multi-threads experiments on logical cores of POWER system.

Performance metrics used for benchmarking [5]**:**

- **Application execution time (Run time)**: The run time is defined as the time that elapses from the moment that an application execution on parallel computation starts to the moment that the last processor finishes execution. The execution times are classified into two:
  1. Serial execution time (T_s) using 1 core (1 threaded)
  2. Parallel execution time (T_p) using p number of cores (p-threads).

- **Speedup (S)**: The speedup is defined as the ratio of the serial execution time of the best sequential algorithm for solving a problem to the time taken by the parallel algorithm to solve the same problem on p processors.

$$S = \frac{T_s}{T_p}$$

- **Efficiency (E)**: The efficiency is defined as the ratio of speedup to the number of processors. Efficiency measures the fraction of time for which a processor is usefully utilized.

$$E = \frac{T_s}{p \; x \; T_p}$$

The BWAKIT execution time, speedup and efficiency are summarized in the Table 3 and Figure 2.

Table 3. Execution time (E), Speedup(S) and Efficiency (E) for BWAKIT on OpenPOWER architecture

# Cores	Exec. Time (in Min)	Speedup	Efficiency (in %)
1	134.99	1	100%
10	16.75	8	81%
20	9.35	14	72%
40	6.59	20	51%
80	5.37	25	31%
160	5.85	23	14%

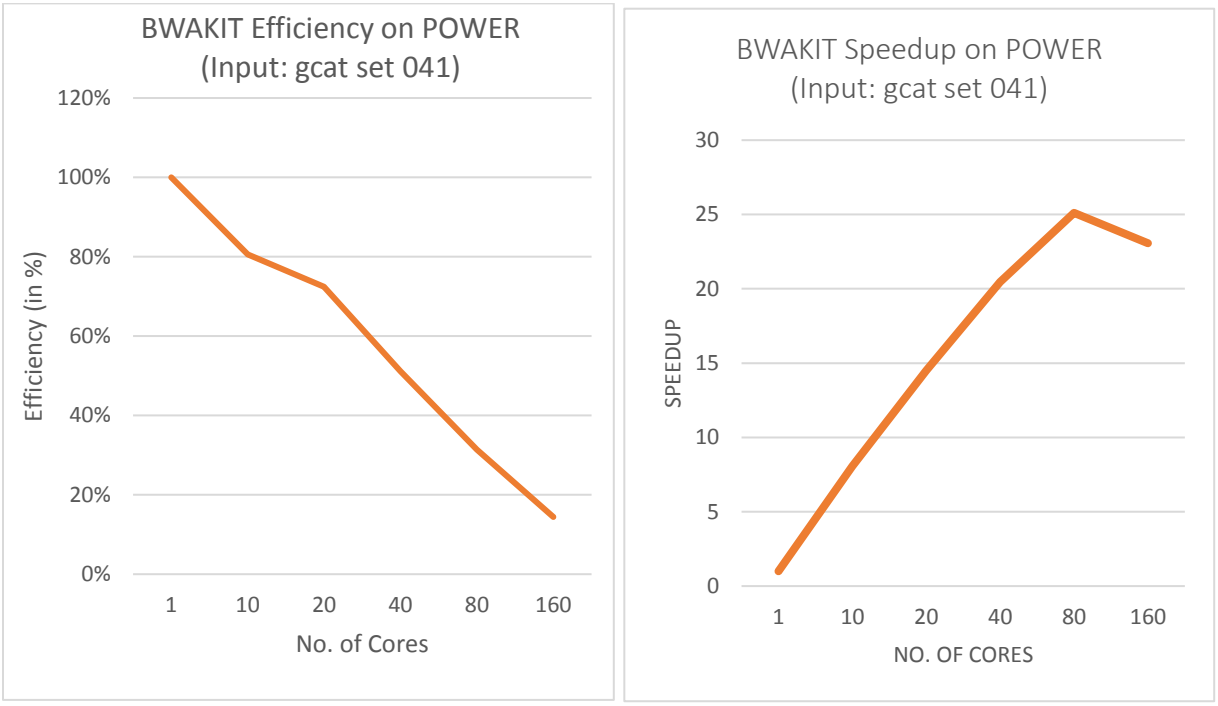

Figure 2. BWAKIT efficiency and speedup on POWER architecture

Validation of BWAKIT results

BWAKIT produces the aligned BAM files after successful execution. During the BWAKIT pipeline porting, couple of source codes are modified and used different APIs. These modifications should not affect the semantics of the application behavior. Hence, we verified the aligned BAM files using BamUtil tools (https://github.com/statgen/bamUtil) across x86_64 and POWER architecture. The results show that the aligned BAM generated on the OpenPOWER architecture is the same as x86_64 architecture. The summary of BamUtil results is shown in Figure 3.

On x86_64

```
Number of records read = 7963505
Number of valid records = 7963505

TotalReads(e6)  7.96
MappedReads(e6) 7.88
PairedReads(e6) 7.96
ProperPair(e6)  7.88
DuplicateReads(e6)      0.00
QCFailureReads(e6)      0.00

MappingRate(%)  98.94
PairedReads(%)  100.00
ProperPair(%)   98.94
DupRate(%)      0.02
QCFailRate(%)   0.00

TotalBases(e6)  1194.53
BasesInMappedReads(e6)  1181.87
Returning: 0 (SUCCESS)
```

On Power8

```
Number of records read = 7963505
Number of valid records = 7963505

TotalReads(e6) 7.96
MappedReads(e6)    7.88
PairedReads(e6)    7.96
ProperPair(e6) 7.88
DuplicateReads(e6) 0.00
QCFailureReads(e6) 0.00

MappingRate(%) 98.94
PairedReads(%) 100.00
ProperPair(%) 98.94
DupRate(%)    0.00
QCFailRate(%) 0.00

TotalBases(e6) 1194.53
BasesInMappedReads(e6)  1181.87
Returning: 0 (SUCCESS)
```

Figure 3. Validation of aligned BAM generated on x86_64 and OpenPOWER architectures.

Acknowledgement:

The authors gratefully acknowledge the access that was provided to OpenPOWER hardware at Forschungszentrum Jülich Supercomputing Center. Special thanks goes to Dr. Dirk Pleiter and Dr. Marcus Richter, Jülich Supercomputing Center, Germany. Additionally, the authors gratefully thank the IBM team, including Mr. Ganesan Narayanasamy, OpenPOWER leader in Education and Research for facilitating the access to OpenPOWER system. Also, the authors would like to thank Mr. Jaideep Bajwa, Mr. Michael Dawson, and Dr. Yinhe Cheng for helping on V8, K8 and trimadap source code modifications for POWER architecture.

Reference:

1. Li, H. & Durbin, R. Fast and accurate short read alignment with burrows–wheeler transform. Bioinformatics, 25(14):1754-1760 (2009).
2. Broad Institute. GATK best practices for the NGS Pipeline. https://goo.gl/mjdmU2 (2016). [Online; accessed 19-Jan-2016].
3. Kathiresan, N., Temanni, R. & Al-Ali, R. Performance improvement of bwa mem algorithm using data-parallel with concurrent parallelization. In Parallel, Distributed and Grid Computing (PDGC), International Conference on 406-411. IEEE (2014).

4. Al-Ali, R., Kathiresan, N., El Anbari, M., Schendel, E. & Zaid, A. Workflow optimization of performance and quality of service for bioinformatics application in high performance computing. Journal of Computational Science, 15, 3-10, (2016).
5. Parallel Computing, Chapter 7 Performance and Scalability [Online access]: https://www.cs.uky.edu/~jzhang/CS621/chapter7.pd
6. Genome Comparison and analysis testing. standard genome data. http://www.bioplanet.com/gcat , (2016). [Online; accessed 19-Jan-2016].