WHITE PAPER

Genome-wide Association Test on Intel Reference Architecture – A Scaling Study

Abstract

In this study, we demonstrate how the Intel Reference Architecture scales efficiently in the compute of the chi-square statistic on millions of SNPs. When this study was performed to compare Wellderly with 1000 Genomes Phase1 and 1000 Genomes Phase3 datasets, these were our observations:

1. The total processing times are directly proportional to the input data sizes of 1000 Genomes and Wellderly datasets being acted upon.
2. The compute-intensive operations took about twice as long as the data-intensive operation. That time increased proportionally as data sizes increased.
3. The Intel Reference Architecture displays very linear but consistent performance with increasing big data sizes.
4. The average processing time increased by about 19% when increasing the average dataset size for 1000 Genomes by about 115% (Wellderly data size was consistent between the runs).

Introduction

Genome-wide association study (GWAS) is one of the most important analytics methods in genetics research. The goal of the association study is to identify the polymorphisms (naturally occurring variations in the DNA), which are associated with increased probability of the occurrence of a certain trait, most frequently a certain disease. The basic concept of the association study is to compare two groups of individuals differing with respect to a certain feature – e.g., a case group of diseased individuals and a control group of healthy people. For a given genomic position, the frequencies of particular genotypes in healthy and affected groups are compared. Typically, a statistical test such as chi-square test is performed to assess the significance of the observed segregation of the genotypes. If a desired significance level is achieved, the polymorphism (most frequently a single nucleotide polymorphism or SNP) is considered associated with the disease in question. The same statistical test is usually performed for a vast number of SNPs or 'genome-wide' (theoretically, it could be done for every single genomic position if only the complete genome sequences for cases and controls were available). Until recently, GWAS studies were usually based on the genotype data obtained with genotyping arrays and thus spanned 0.5 to 2 million SNPs. Now, the next generation sequencing technology has enabled reading the genomic sequences across many millions of SNPs, which allows performing GWAS analyses at the unprecedented depth. This new technology, however, has also posed new analytics and computational challenges. Next

Abhi K. Basu
Intel Corporation, Big Data Solutions/
Data Center Group
Portland, OR, USA

Weronika Sikora-Wohlfeld
Division of Systems Medicine
Department of Pediatrics, Stanford University
Stanford, CA, USA

Atul J. Butte
Division of Systems Medicine
Department of Pediatrics, Stanford University
Stanford, CA, USA

Institute for Computational Health Sciences
University of California
San Francisco, CA, USA

Monica Martinez-Canales
Intel Corporation, Big Data Solutions/
Data Center Group
Santa Clara, CA, USA

Table of Contents

generation sequencing experiments generate vast amounts of data, the storage and analysis of which requires novel computational solutions.

Intel has recently developed a specialized infrastructure for analyzing large biological datasets, including genome sequencing data, referred to as the Intel Reference Architecture[1]. This complete system of hardware components and software solutions optimized to deal with the 'big data' allows storing the genome sequence data in a tabularized format and manipulating them in 'nearly real-time' using the in-memory SQL-like queries[1]. Recently, we have demonstrated that the Intel Reference Architecture can be applied successfully to perform advanced operations typically conducted in genomics research, such as calculating allele and genotype frequencies, and computing the Hardy-Weinberg statistics and the chi-square statistics[2]. To illustrate the chi-square test, we have assessed the segregation of the genotypes between East Asian and African populations for 12 type 2 diabetes-associated SNPs[2].

In the current study, we demonstrate how the Intel Reference Architecture can be used to efficiently perform the chi-square test on millions of SNPs.

Chi-square Test for Wellderly and 1000 Genomes Data

To illustrate the chi-square test run on the Intel Reference Architecture, we used two publicly available datasets: Wellderly and 1000 Genomes. Wellderly is a repository of genome sequences of over 500 elderly individuals who were unusually healthy despite their advanced age. The data has been collected and maintained by Scripps Translational Science Institute[3]. 1000 Genomes is a large set of human genomic sequences from more than 2,500 individuals (~1,000 sequences released in phase 1[4] and ~2,500 sequences released in phase 3[5]). Since no specific criteria were applied to select the participants, the 1000 Genomes dataset is expected to cover the average, relatively healthy individuals and, as such, is meant to constitute a representative cohort that could be used as control in genetic studies. In this paper, we conducted a GWAS-like analysis using Wellderly and 1000 Genomes data as case and control cohorts, respectively. Using the in-memory Impala queries, we calculated chi-square statistic for every SNP shared between Wellderly and 1000 Genomes data. In addition to measuring the time needed to perform genome-wide association test, the other main objective of the analysis was to investigate how the runtime of the calculation scales with the size of the dataset in question. Therefore, we conducted two separate analyses – Wellderly against 1000 Genomes data phase 1 (~1,000 samples) and Wellderly against 1000 Genomes data phase 3 (~2,500 samples) – and we compared the runtime of these tests.

Technical Specifications

Datasets

Raw Data
The three datasets used in the study are listed below. The numbers inside the parentheses denote the size of the original files once they were uncompressed.

- Wellderly (615 GB) - http://www.stsi-web.org//wellderly/detail/wellderly/
- 1000 Genomes – phase 1 (1.2 TB) - ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/phase1/analysis_results/integrated_call_sets/ (referred to as 1Kgp1)
- 1000 Genomes – phase 3 (2.5 TB) - ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/release/20130502/ (referred to as 1Kgp3)

Tools/Platforms

The following software components in [1] were used:

- Cloudera Hadoop Distribution 5.1[6]
- Impala 1.4.0[7]
- Hive 0.12[8]
- Python 2.6[9]
  - Impyla library 0.8.1[10]
  - Thrift library 0.8.0[11]
- Hadoop Distributed File System (HDFS)[12]

Cluster Details

The following hardware components in [1] were used:

- 6 Node Hadoop Cluster + 1 Edge Node (running Hadoop Client)
  - Intel® Xeon®[13] E5-2680 2 socket CPU
  - 192 GB RAM
  - 24 TB HDD / 300 GB SSD
- 10 Gbps Networking

Data Transformation Steps

Data Scale

In any population study, as larger datasets are added to increase the population sample size, most data scientists examine the dataset sizes and relative growth over their initial datasets to determine if there will be a scalability limitation in their infrastructure capability.

Although the Wellderly dataset remains constant during the study, we observe that the 1000 Genomes dataset has increased significantly (115% on average) from phase 1 to phase 3 (Table 1), over the 23 chromosome files. For frame of reference, we can observe the following:

1. The maximum increase (chrX) is about 133% more in phase 3 over phase 1.
2. The median increase (chr7) is about 112% more in phase 3 over phase 1.
3. The minimum increase (chr6) is about 107% more in phase 3 over phase 1.

TABLE 1: Size of each chromosome-specific input table for 1000 Genomes phase 1 (1kgp1), 1000 Genomes phase 3 (1kgp3) and Wellderly (well). The last column shows the percentage increase in size from 1kgp1 to 1kgp3.

CHROMOSOME	1KGP1 DATAROWS	1KGP3 DATAROWS	WELL DATAROWS	PERCENTAGE INCREASE IN DATAROWS FROM 1KGP1 TO 1KGP3
Chr1	3007196	6468094	5002461	115.09
Chr2	3307592	7081600	5487856	114.10
Chr3	2763454	5832276	4627545	111.05
Chr4	2736765	5732585	4809994	109.47
Chr5	2530217	5265763	4212658	108.12
Chr6	2424425	5024119	4048835	107.23
Chr7	2215231	4716715	3686823	112.92
Chr8	2183839	4597105	3389229	110.51
Chr9	1652389	3560687	2561956	115.49
Chr10	1882663	3992219	2902349	112.05
Chr11	1894908	4045628	2991743	113.50
Chr12	1828006	3868428	3061735	111.62
Chr13	1373000	2857916	2426912	108.15
Chr14	1258254	2655067	2072217	111.01
Chr15	1130554	2424689	1725840	114.47
Chr16	1210619	2697949	1717332	122.86
Chr17	1046733	2329288	1697346	122.53
Chr18	1088820	2267185	1751878	108.22
Chr19	816115	1832506	1351312	124.54
Chr20	855166	1812841	1252697	111.99
Chr21	518965	1105538	838257	113.03
Chr22	494328	1103547	726881	123.24
ChrX	1487477	3468093	2658420	133.15

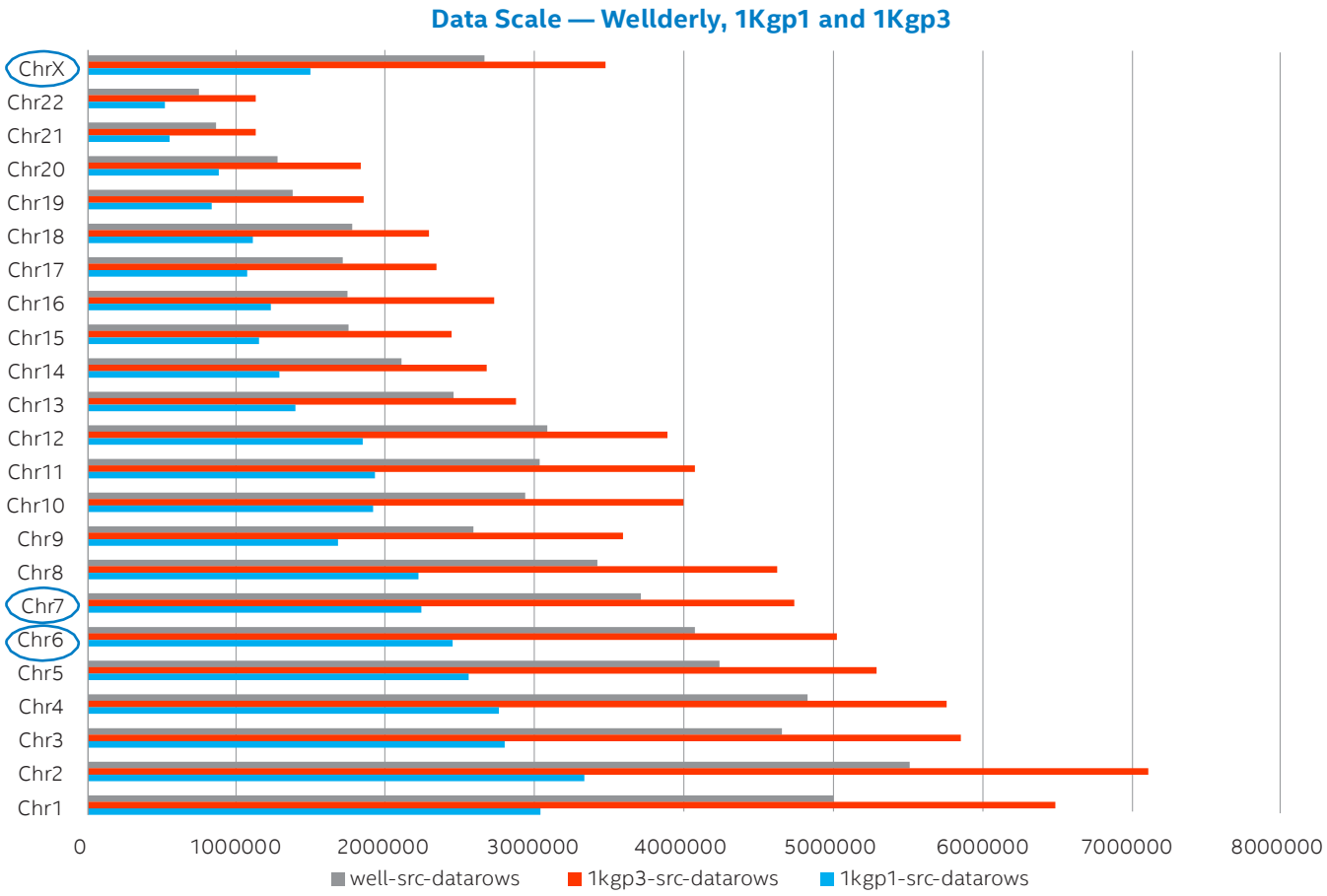


Figure 1: Data size comparison between 1000 Genomes Phase1, 1000 Genomes Phase 3 and Wellderly. The three highlighted chromosomes are our frames of reference with max (chrX), min (chr6) and median (chr7) dataset increases between 1kgp1 and 1kgp3

Data Preparation

1. Genotype code (GT) was extracted from raw tables (23 chromosome-specific tables for each of the three datasets: Wellderly, 1000 Genomes phase 1 and 1000 Genomes phase 3) and represented as the number of alternate alleles according to the convention: 0|0 = 0, 0|1 or 1|0 = 1 and 1|1 = 2. Sample data shown in Table 2.

TABLE 2: Sample genotype data for Wellderly and 1000 Genomes tables. The column headers contain the human genome project sample number.

HG00096	HG00097	HG00099	NA20778
1	0	1	1
1	0	0	0
1	0	0	0
1	0	0	0
1	0	0	1

2. For each SNP, all possible genotypes (reference/reference, reference/alternate and alternate/alternate) across all samples (columns, e.g., hg00096, etc.) were counted using the following convention:
   - ref_ref = number of samples with genotype '0'
   - ref_alt = number of samples with genotype '1'
   - alt_alt = number of samples with genotype '2'

   Note: 'ref_ref ', 'ref_alt' and 'alt_alt' refer to reference homozygote, heterozygote and alternate homozygote, respectively.

   Now, all chromosome-specific Wellderly and 1000 Genomes tables contain only eight columns - chrom, pos, ref, alt, ref_ref, ref_alt, alt_alt and rowid columns. Note that chrom, pos, ref, and alt still follow the standard VCF format.

Queries

We designed and implemented a set of queries that join Wellderly and 1000 Genomes tables to calculate chi-square statistic. All queries were generated using Python code and the Impyla library was used to run the queries against the Impala nodes of the Hadoop cluster in a distributed fashion. These will be referred to as the three steps of data transformation.

Here, chrx corresponds to one of {chr1, chr2, ..., chrX}

1. Join the two sets of tables together on chromosome, position, ref and alt columns.

```
CREATE TABLE well_1kgp3_chrx
AS SELECT
w.w_chrom as chrom, w.w_pos as pos, w.w_alt as alt, w.w_ref as ref,
w.w_ref_ref as x1, w.w_ref_alt as x2, w.w_alt_alt as x3,
s.s_ref_ref as x4, s.s_ref_alt as x5, s.s_alt_alt as x6,
w.w_rowid, s.s_rowid
FROM well_chrx_gen w JOIN sc_1kgp3_chrx_gen s
ON (w.w_chrom = s.s_chrom AND w.w_pos = s.s_pos AND w.w_ref = s.s_ref AND
w.w_alt = s.s_alt);
```

For instance, suppose that chrx refers to chr1. Then, the sc_1kgp3_chr1_gen table refers to the 1000 Genomes Phase 3 data for chromosome 1. The well_chr1_gen table refers to the Wellderly data for chromosome 1. The joined table is then referred to as well_1kgp3_chr1.

2. Generate expected values of genotype counts.

```
CREATE TABLE well_1kgp3_chrx_exp
AS SELECT
chrom, pos, alt, ref,
x1, x2, x3, x4, x5, x6,

(x1+x2+x3) * (x1+x4) / (x1+x2+x3+x4+x5+x6) as e1,
(x1+x2+x3) * (x2+x5) / (x1+x2+x3+x4+x5+x6) as e2,
(x1+x2+x3) * (x3+x6) / (x1+x2+x3+x4+x5+x6) as e3,
(x4+x5+x6) * (x1+x4) / (x1+x2+x3+x4+x5+x6) as e4,
(x4+x5+x6) * (x2+x5) / (x1+x2+x3+x4+x5+x6) as e5,
(x4+x5+x6) * (x3+x6) / (x1+x2+x3+x4+x5+x6) as e6
FROM well_1kgp3_chrx;
```

Once we have the well_1kgp3_chr1 table, the expected values on GT counts result in the well_1kgp3_chr1_exp table.

3. Calculate chi-square statistic values.

```
CREATE TABLE well _ 1kgp3 _ chrx _ chisq
AS SELECT
chrom, pos, alt, ref,
 x1, x2, x3, x4, x5, x6,
 e1, e2, e3, e4, e5, e6,
(pow(cast(x1-e1 as double), 2) / e1) + (pow(cast(x2-e2 as double),2) / e2) +
(pow(cast(x3-e3 as double),2) / e3)
+ (pow(cast(x4-e4 as double),2) / e4) + (pow(cast(x5-e5 as double),2) /
e5) + (pow(cast(x6-e6 as double),2) / e6) as chi _ sq
FROM well _ 1kgp3 _ chrx _ exp;
```

From the expected values table well_1kgp3_chr1_exp table, we finish computing the chi-square statistic to generate the well_1kgp3_chr1_chisq.

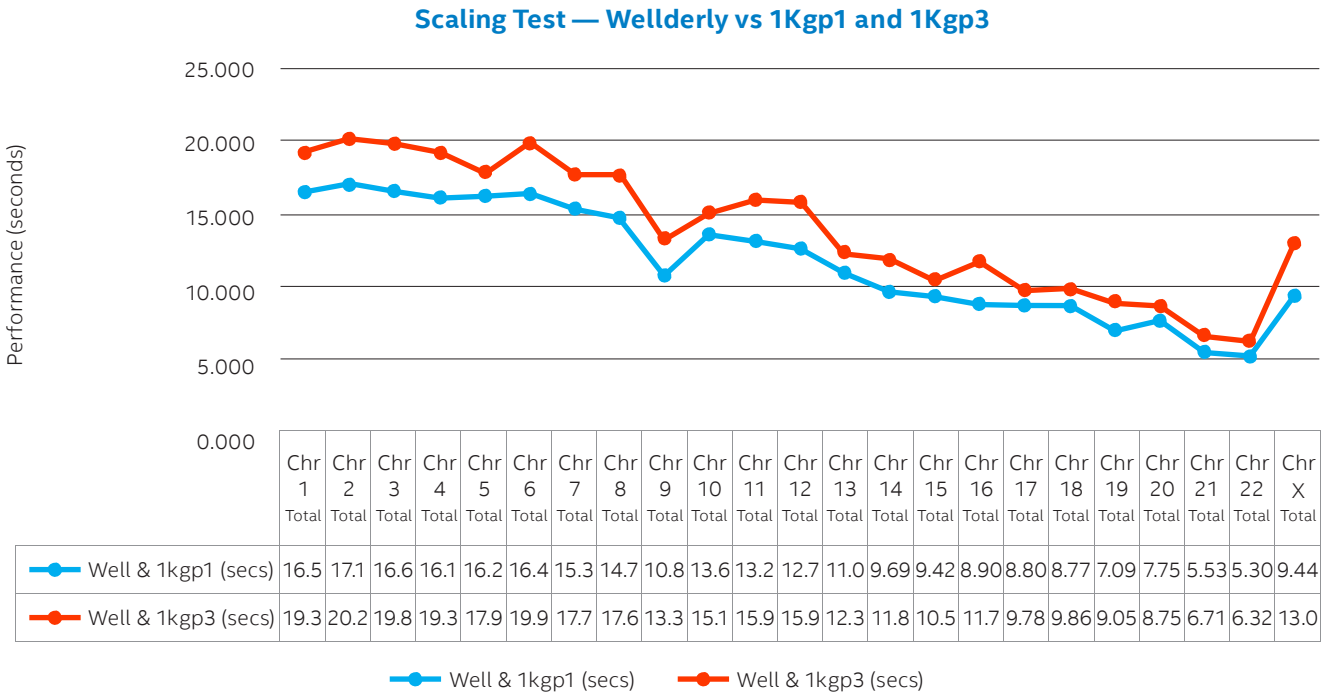
These three queries are applied across all the chromosome tables across the three different datasets. Appendix - Table 5 (1000 Genomes Phase 1 and Wellderly) and Appendix - Table 6 (1000 Genomes Phase 3 and Wellderly) summarize how the data table sizes (rows and columns) change through each transformation step leading to calculating the chi-square statistic.

GWAS Chi-Square Test Performance

Table 3 demonstrates the average run times of all cumulative queries per chromosome (5 runs each averaged), contrasted between the two versions of 1000 Genomes and Wellderly datasets.

TABLE 3: Average cumulative query runtimes per chromosome for Wellderly with 1000 Genomes phase1 vs. 1000 Genomes phase 3.

CHROMOSOME	WELL & 1KGP1 (SECS)	WELL & 1KGP3 (SECS)
Chr1 Total	16.557	19.313
Chr2 Total	17.123	20.218
Chr3 Total	16.679	19.856
Chr4 Total	16.169	19.310
Chr5 Total	16.262	17.951
Chr6 Total	16.427	19.952
Chr7 Total	15.377	17.780
Chr8 Total	14.793	17.656
Chr9 Total	10.833	13.370
Chr10 Total	13.625	15.184
Chr11 Total	13.207	15.999
Chr12 Total	12.721	15.934
Chr13 Total	11.034	12.392
Chr14 Total	9.693	11.855
Chr15 Total	9.421	10.531
Chr16 Total	8.901	11.786
Chr17 Total	8.804	9.784
Chr18 Total	8.770	9.864
Chr19 Total	7.095	9.054
Chr20 Total	7.758	8.755
Chr21 Total	5.538	6.712
Chr22 Total	5.303	6.322
ChrX Total	9.445	13.019



	Chr 1 Total	Chr 2 Total	Chr 3 Total	Chr 4 Total	Chr 5 Total	Chr 6 Total	Chr 7 Total	Chr 8 Total	Chr 9 Total	Chr 10 Total	Chr 11 Total	Chr 12 Total	Chr 13 Total	Chr 14 Total	Chr 15 Total	Chr 16 Total	Chr 17 Total	Chr 18 Total	Chr 19 Total	Chr 20 Total	Chr 21 Total	Chr 22 Total	Chr X Total
Well & 1kgp1 (secs)	16.5	17.1	16.6	16.1	16.2	16.4	15.3	14.7	10.8	13.6	13.2	12.7	11.0	9.69	9.42	8.90	8.80	8.77	7.09	7.75	5.53	5.30	9.44
Well & 1kgp3 (secs)	19.3	20.2	19.8	19.3	17.9	19.9	17.7	17.6	13.3	15.1	15.9	15.9	12.3	11.8	10.5	11.7	9.78	9.86	9.05	8.75	6.71	6.32	13.0

Figure 2. Cumulative average run times per chromosome comparison between Wellderly and 1000 Genomes Phase 1 and 1000 Genomes Phase 3.

Figure 2 shows how the total processing time increased linearly (between 1Kgp1 and 1Kgp3) even though the datasets increased by over 115% on average, over all 23 chromosome files. The performance was recorded to be very consistent for both runs (with 1Kgp1 and 1Kgp3) and scalability was not a problem for this reference architecture.

Next, we looked at total run time separated into data- and compute-intensive operations, to observe any patterns.

The join (step 1) was the data-intensive operation while expected value calculation (step 2) and chi-square calculation (step 3) were added together and considered the compute-intensive operation. Figure 3 (data-intensive) and Figure 4 (compute-intensive) also point to the same pattern of small but linear increase in processing time as datasets for 1000 Genomes grew by almost 115% (average).

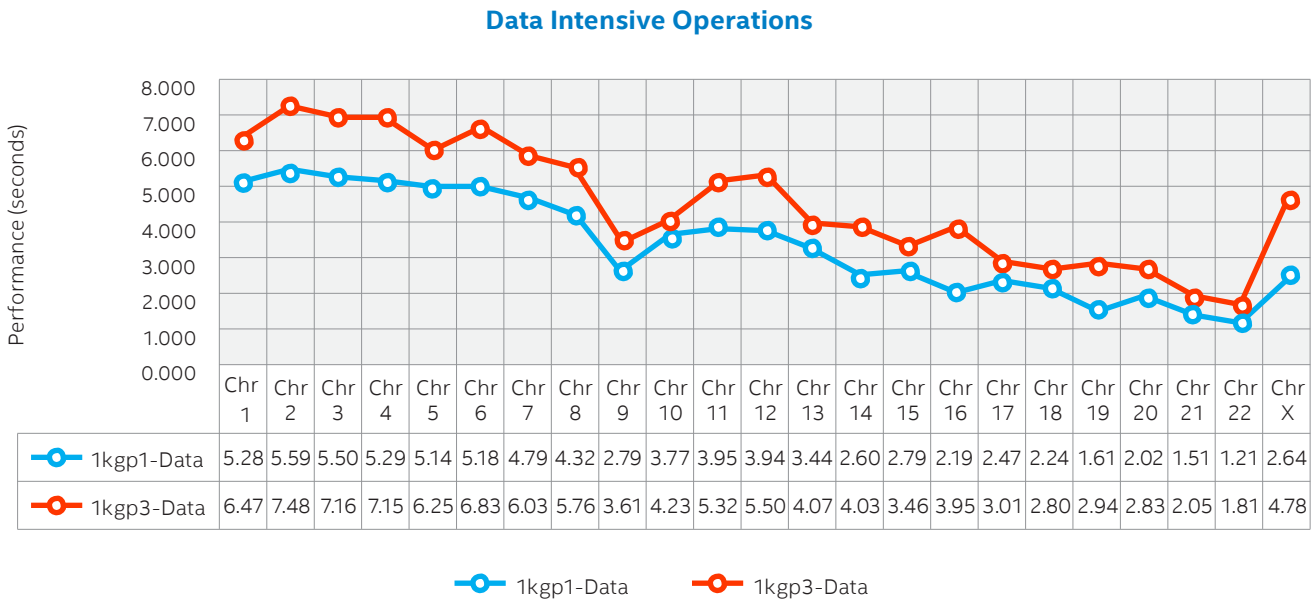
Here is the comparison among our frame of reference data points:

TABLE 4: Close look at reference data points – maximum, median and minimum data set increases between 1000 Genomes Phase 1 and Phase 3. All times in seconds.

CHROMOSOME	WELL & 1KGP1 (TOTAL)	WELL & 1KGP3 (TOTAL)	WELL & 1KGP1 (DATA-INTENSIVE)	WELL & 1KGP3 (DATA-INTENSIVE)	WELL & 1KGP1 (COMPUTE-INTENSIVE)	WELL & 1KGP3 (COMPUTE-INTENSIVE)	1KGP1 DATA SIZE (ROWS X 8 COLS)	1KGP3 DATA SIZE (ROWS X 8 COLS)	WELLDERLY DATA SIZE (ROWS X 8 COLS)
Chr6 (min)	16.43	19.95	5.18	6.83	10.86	12.69	2424425	5024119	4212658
Chr7 (median)	15.37	17.78	4.79	6.04	10.18	11.34	2215231	4716715	3686823
ChrX (max)	9.45	13.019	2.65	4.78	6.45	7.86	1487477	3468093	2658420

Key Takeaways

1. We observe that the total processing times are directly proportional to the input data sizes of 1000 Genomes and Wellderly datasets being acted upon.
2. In our study, the compute-intensive operations took about twice as long as the data-intensive operations. That time increased proportionally as data sizes increased.
3. The Intel Reference Architecture displays very linear but consistent performance with increasing big data sizes.
4. The average processing time increases by about 19% when increasing the average dataset size for 1000 Genomes by about 115% (Wellderly data size was consistent between the runs).



	Chr 1	Chr 2	Chr 3	Chr 4	Chr 5	Chr 6	Chr 7	Chr 8	Chr 9	Chr 10	Chr 11	Chr 12	Chr 13	Chr 14	Chr 15	Chr 16	Chr 17	Chr 18	Chr 19	Chr 20	Chr 21	Chr 22	Chr X
1kgp1-Data	5.28	5.59	5.50	5.29	5.14	5.18	4.79	4.32	2.79	3.77	3.95	3.94	3.44	2.60	2.79	2.19	2.47	2.24	1.61	2.02	1.51	1.21	2.64
1kgp3-Data	6.47	7.48	7.16	7.15	6.25	6.83	6.03	5.76	3.61	4.23	5.32	5.50	4.07	4.03	3.46	3.95	3.01	2.80	2.94	2.83	2.05	1.81	4.78

Figure 3: Run times of data-intensive operations by chromosome.

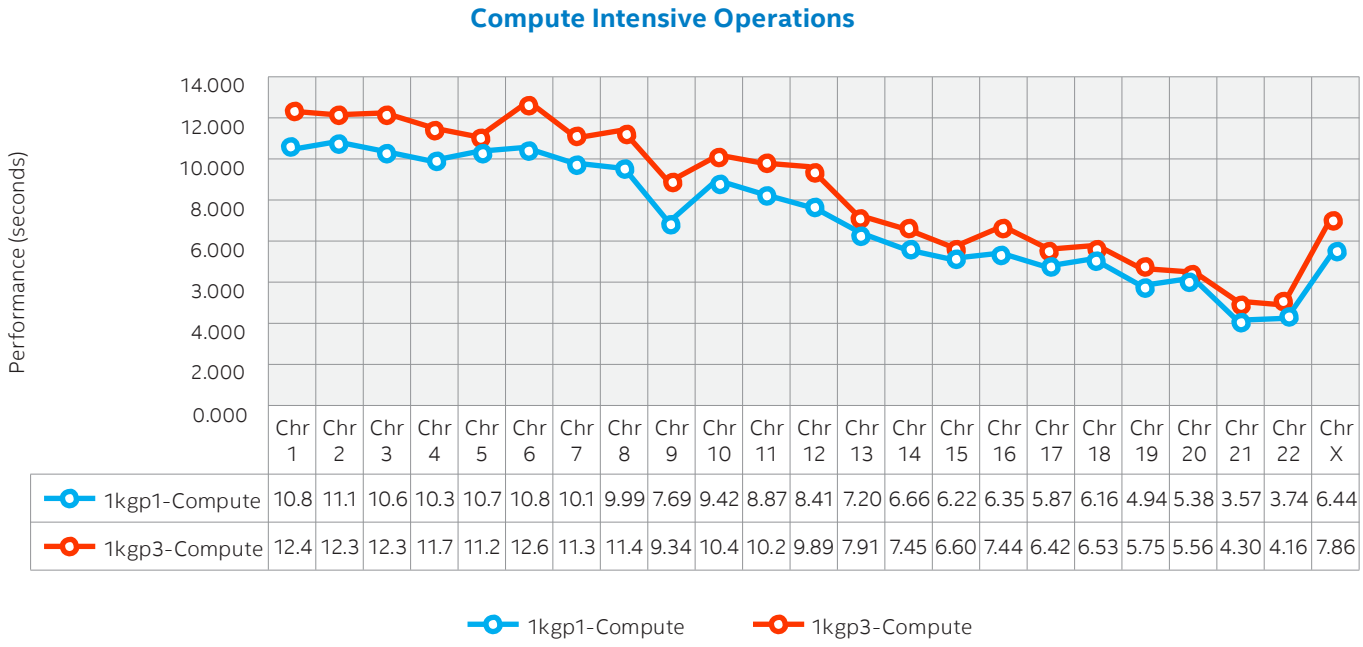


Figure 4: Run times of compute-intensive operations by chromosome.

Conclusions

In this scaling study, we have demonstrated how Intel Reference Architecture can be used to perform analysis on big genome sequencing datasets. The architecture is scalable for increasing data sizes. Overall performance is found to be linear and consistent given the size of datasets, and there is a small penalty with increase in datasets. We have shown how "near real-time" SQL queries operate on a Hadoop cluster on vast datasets, and the architecture is able to adapt and scale to the increasing demands of even bigger data sets.

Appendix

The three queries (Join, Expected Value and Chi-Squared) are applied across all the chromosome tables across the three different datasets. Table 5 (1000 Genomes Phase 1 and Wellderly) and Table 6 (1000 Genomes Phase 3 and Wellderly) summarize how the data table sizes (rows and columns) change through each transformation step leading to calculating the chi-square statistic.

To read Tables 5 and 6, look at the first 3 rows of Table 5 which correspond to Steps 1-3 of the Query data transformation on chromosome 1:

- **Step 1:** sc_1kgp1_chr1_gen is input 1 and well_chr1_gen is input2 to the JOIN operation whose output is well_1kgp1_chr1. Input1 has r1 rows and c1 columns. Input2 has r2 rows and c2 columns. Output has r3 rows and c3 columns. Therefore, we see that sc_1kgp1_chr1_gen (3007196 rows x 8 cols) JOINED with well_chr1_gen (5002461 rows x 8 cols) generates well_1kgp1_chr1 (834289 rows x 12 cols).
- **Step 2:** Calculations on input1 well_1kgp1_chr1 (834289 rows x 12 cols) generate output well_1kgp1_chr1_exp (834289 rows x 18 cols). There is no input2 at this step.
- **Step 3:** Calculations on input1 well_1kgp1_chr1_exp (834289 rows x 18 cols) generate output well_1kgp1_chr1_chisq (834289 rows x 19 cols). There is no input2 at this step.

Reading the remainder of the tables follow similarly.

TABLE 5: Breakdown of data transformation for each chromosome through all the 3 steps of processing – JOIN, Expected Value calculation and Chi-Square calculations – of 1000 Genomes Phase 1 and Wellderly datasets.

PYTHON 1KG-P1/WELLDERLY CHI-SQUARE STATISTICS LOG									
CHR	INPUT1	R1	C1	INPUT2	R2	C2	OUTPUT	R3	C3
chr1	sc_1kgp1_chr1_gen	3007196	8	well_chr1_gen	5002461	8	well_1kgp1_chr1	834289	12
	well_1kgp1_chr1	834289	12				well_1kgp1_chr1_exp	834289	18
	well_1kgp1_chr1_exp	834289	18				well_1kgp1_chr1_chisq	834289	19
chr2	sc_1kgp1_chr2_gen	3307592	8	well_chr2_gen	5487856	8	well_1kgp1_chr2	902711	12
	well_1kgp1_chr2	902711	12				well_1kgp1_chr2_exp	902711	18
	well_1kgp1_chr2_exp	902711	18				well_1kgp1_chr2_chisq	902711	19
chr3	sc_1kgp1_chr3_gen	2763454	8	well_chr3_gen	4627545	8	well_1kgp1_chr3	754656	12
	well_1kgp1_chr3	754656	12				well_1kgp1_chr3_exp	754656	18
	well_1kgp1_chr3_exp	754656	18				well_1kgp1_chr3_chisq	754656	19
chr4	sc_1kgp1_chr4_gen	2736765	8	well_chr4_gen	4809994	8	well_1kgp1_chr4	748889	12
	well_1kgp1_chr4	748889	12				well_1kgp1_chr4_exp	748889	18
	well_1kgp1_chr4_exp	748889	18				well_1kgp1_chr4_chisq	748889	19
chr5	sc_1kgp1_chr5_gen	2530217	8	well_chr5_gen	4212658	8	well_1kgp1_chr5	678871	12
	well_1kgp1_chr5	678871	12				well_1kgp1_chr5_exp	678871	18
	well_1kgp1_chr5_exp	678871	18				well_1kgp1_chr5_chisq	678871	19
chr6	sc_1kgp1_chr6_gen	2424425	8	well_chr6_gen	4048835	8	well_1kgp1_chr6	687225	12
	well_1kgp1_chr6	687225	12				well_1kgp1_chr6_exp	687225	18
	well_1kgp1_chr6_exp	687225	18				well_1kgp1_chr6_chisq	687225	19
chr7	sc_1kgp1_chr7_gen	2215231	8	well_chr7_gen	3686823	8	well_1kgp1_chr7	612601	12
	well_1kgp1_chr7	612601	12				well_1kgp1_chr7_exp	612601	18
	well_1kgp1_chr7_exp	612601	18				well_1kgp1_chr7_chisq	612601	19
chr8	sc_1kgp1_chr8_gen	2183839	8	well_chr8_gen	3389229	8	well_1kgp1_chr8	596768	12
	well_1kgp1_chr8	596768	12				well_1kgp1_chr8_exp	596768	18
	well_1kgp1_chr8_exp	596768	18				well_1kgp1_chr8_chisq	596768	19
chr9	sc_1kgp1_chr9_gen	1652389	8	well_chr9_gen	2561956	8	well_1kgp1_chr9	459234	12
	well_1kgp1_chr9	459234	12				well_1kgp1_chr9_exp	459234	18
	well_1kgp1_chr9_exp	459234	18				well_1kgp1_chr9_chisq	459234	19

continued from previous page

PYTHON 1KG-P1/WELLDERLY CHI-SQUARE STATISTICS LOG

CHR	INPUT1	R1	C1	INPUT2	R2	C2	OUTPUT	R3	C3
chr10	sc_1kgp1_chr10_gen	1882663	8	well_chr10_gen	2902349	8	well_1kgp1_chr10	535362	12
	well_1kgp1_chr10	535362	12				well_1kgp1_chr10_exp	535362	18
	well_1kgp1_chr10_exp	535362	18				well_1kgp1_chr10_chisq	535362	19
chr11	sc_1kgp1_chr11_gen	1894908	8	well_chr11_gen	2991743	8	well_1kgp1_chr11	520756	12
	well_1kgp1_chr11	520756	12				well_1kgp1_chr11_exp	520756	18
	well_1kgp1_chr11_exp	520756	18				well_1kgp1_chr11_chisq	520756	19
chr12	sc_1kgp1_chr12_gen	1828006	8	well_chr12_gen	3061735	8	well_1kgp1_chr12	495166	12
	well_1kgp1_chr12	495166	12				well_1kgp1_chr12_exp	495166	18
	well_1kgp1_chr12_exp	495166	18				well_1kgp1_chr12_chisq	495166	19
chr13	sc_1kgp1_chr13_gen	1373000	8	well_chr13_gen	2426912	8	well_1kgp1_chr13	380847	12
	well_1kgp1_chr13	380847	12				well_1kgp1_chr13_exp	380847	18
	well_1kgp1_chr13_exp	380847	18				well_1kgp1_chr13_chisq	380847	19
chr14	sc_1kgp1_chr14_gen	1258254	8	well_chr14_gen	2072217	8	well_1kgp1_chr14	358823	12
	well_1kgp1_chr14	358823	12				well_1kgp1_chr14_exp	358823	18
	well_1kgp1_chr14_exp	358823	18				well_1kgp1_chr14_chisq	358823	19
chr15	sc_1kgp1_chr15_gen	1130554	8	well_chr15_gen	1725840	8	well_1kgp1_chr15	312156	12
	well_1kgp1_chr15	312156	12				well_1kgp1_chr15_exp	312156	18
	well_1kgp1_chr15_exp	312156	18				well_1kgp1_chr15_chisq	312156	19
chr16	sc_1kgp1_chr16_gen	1210619	8	well_chr16_gen	1717332	8	well_1kgp1_chr16	343719	12
	well_1kgp1_chr16	343719	12				well_1kgp1_chr16_exp	343719	18
	well_1kgp1_chr16_exp	343719	18				well_1kgp1_chr16_chisq	343719	19
chr17	sc_1kgp1_chr17_gen	1046733	8	well_chr17_gen	1697346	8	well_1kgp1_chr17	295240	12
	well_1kgp1_chr17	295240	12				well_1kgp1_chr17_exp	295240	18
	well_1kgp1_chr17_exp	295240	18				well_1kgp1_chr17_chisq	295240	19
chr18	sc_1kgp1_chr18_gen	1088820	8	well_chr18_gen	1751878	8	well_1kgp1_chr18	308591	12
	well_1kgp1_chr18	308591	12				well_1kgp1_chr18_exp	308591	18
	well_1kgp1_chr18_exp	308591	18				well_1kgp1_chr18_chisq	308591	19
chr19	sc_1kgp1_chr19_gen	816115	8	well_chr19_gen	1351312	8	well_1kgp1_chr19	238370	12
	well_1kgp1_chr19	238370	12				well_1kgp1_chr19_exp	238370	18
	well_1kgp1_chr19_exp	238370	18				well_1kgp1_chr19_chisq	238370	19
chr20	sc_1kgp1_chr20_gen	855166	8	well_chr20_gen	1252697	8	well_1kgp1_chr20	247693	12
	well_1kgp1_chr20	247693	12				well_1kgp1_chr20_exp	247693	18
	well_1kgp1_chr20_exp	247693	18				well_1kgp1_chr20_chisq	247693	19
chr21	sc_1kgp1_chr21_gen	518965	8	well_chr21_gen	838257	8	well_1kgp1_chr21	140810	12
	well_1kgp1_chr21	140810	12				well_1kgp1_chr21_exp	140810	18
	well_1kgp1_chr21_exp	140810	18				well_1kgp1_chr21_chisq	140810	19
chr22	sc_1kgp1_chr22_gen	494328	8	well_chr22_gen	726881	8	well_1kgp1_chr22	152568	12
	well_1kgp1_chr22	152568	12				well_1kgp1_chr22_exp	152568	18
	well_1kgp1_chr22_exp	152568	18				well_1kgp1_chr22_chisq	152568	19
chrX	sc_1kgp1_chrX_gen	1487477	8	well_chrX_gen	2658420	8	well_1kgp1_chrX	358869	12
	well_1kgp1_chrX	358869	12				well_1kgp1_chrX_exp	358869	18
	well_1kgp1_chrX_exp	358869	18				well_1kgp1_chrX_chisq	358869	19

TABLE 6: BREAKDOWN OF DATA TRANSFORMATION FOR EACH CHROMOSOME THROUGH ALL THE 3 STEPS OF PROCESSING – JOIN, EXPECTED VALUE CALCULATION AND CHI-SQUARE CALCULATIONS – OF 1000 GENOMES PHASE 3 AND WELLDERLY DATASETS.

PYTHON 1KG-P3/WELLDERLY CHI-SQUARE STATISTICS LOG									
CHR	**INPUT1**	**R1**	**C1**	**INPUT2**	**R2**	**C2**	**OUTPUT**	**R3**	**C3**
chr1	sc_1kgp3_chr1_gen	6468094	8	well_chr1_gen	5002461	8	well_1kgp3_chr1	1008238	12
	well_1kgp3_chr1	1008238	12				well_1kgp3_chr1_exp	1008238	18
	well_1kgp3_chr1_exp	1008238	18				well_1kgp3_chr1_chisq	1008238	19
chr2	sc_1kgp3_chr2_gen	7081600	8	well_chr2_gen	5487856	8	well_1kgp3_chr2	1086150	12
	well_1kgp3_chr2	1086150	12				well_1kgp3_chr2_exp	1086150	18
	well_1kgp3_chr2_exp	1086150	18				well_1kgp3_chr2_chisq	1086150	19
chr3	sc_1kgp3_chr3_gen	5832276	8	well_chr3_gen	4627545	8	well_1kgp3_chr3	906185	12
	well_1kgp3_chr3	906185	12				well_1kgp3_chr3_exp	906185	18
	well_1kgp3_chr3_exp	906185	18				well_1kgp3_chr3_chisq	906185	19
chr4	sc_1kgp3_chr4_gen	5732585	8	well_chr4_gen	4809994	8	well_1kgp3_chr4	899435	12
	well_1kgp3_chr4	899435	12				well_1kgp3_chr4_exp	899435	18
	well_1kgp3_chr4_exp	899435	18				well_1kgp3_chr4_chisq	899435	19
chr5	sc_1kgp3_chr5_gen	5265763	8	well_chr5_gen	4212658	8	well_1kgp3_chr5	803293	12
	well_1kgp3_chr5	803293	12				well_1kgp3_chr5_exp	803293	18
	well_1kgp3_chr5_exp	803293	18				well_1kgp3_chr5_chisq	803293	19
chr6	sc_1kgp3_chr6_gen	5024119	8	well_chr6_gen	4048835	8	well_1kgp3_chr6	819695	12
	well_1kgp3_chr6	819695	12				well_1kgp3_chr6_exp	819695	18
	well_1kgp3_chr6_exp	819695	18				well_1kgp3_chr6_chisq	819695	19
chr7	sc_1kgp3_chr7_gen	4716715	8	well_chr7_gen	3686823	8	well_1kgp3_chr7	738162	12
	well_1kgp3_chr7	738162	12				well_1kgp3_chr7_exp	738162	18
	well_1kgp3_chr7_exp	738162	18				well_1kgp3_chr7_chisq	738162	19
chr8	sc_1kgp3_chr8_gen	4597105	8	well_chr8_gen	3389229	8	well_1kgp3_chr8	714813	12
	well_1kgp3_chr8	714813	12				well_1kgp3_chr8_exp	714813	18
	well_1kgp3_chr8_exp	714813	18				well_1kgp3_chr8_chisq	714813	19
chr9	sc_1kgp3_chr9_gen	3560687	8	well_chr9_gen	2561956	8	well_1kgp3_chr9	549936	12
	well_1kgp3_chr9	549936	12				well_1kgp3_chr9_exp	549936	18
	well_1kgp3_chr9_exp	549936	18				well_1kgp3_chr9_chisq	549936	19
chr10	sc_1kgp3_chr10_gen	3992219	8	well_chr10_gen	2902349	8	well_1kgp3_chr10	640275	12
	well_1kgp3_chr10	640275	12				well_1kgp3_chr10_exp	640275	18
	well_1kgp3_chr10_exp	640275	18				well_1kgp3_chr10_chisq	640275	19
chr11	sc_1kgp3_chr11_gen	4045628	8	well_chr11_gen	2991743	8	well_1kgp3_chr11	626302	12
	well_1kgp3_chr11	626302	12				well_1kgp3_chr11_exp	626302	18
	well_1kgp3_chr11_exp	626302	18				well_1kgp3_chr11_chisq	626302	19
chr12	sc_1kgp3_chr12_gen	3868428	8	well_chr12_gen	3061735	8	well_1kgp3_chr12	596230	12
	well_1kgp3_chr12	596230	12				well_1kgp3_chr12_exp	596230	18
	well_1kgp3_chr12_exp	596230	18				well_1kgp3_chr12_chisq	596230	19
chr13	sc_1kgp3_chr13_gen	2857916	8	well_chr13_gen	2426912	8	well_1kgp3_chr13	454739	12
	well_1kgp3_chr13	454739	12				well_1kgp3_chr13_exp	454739	18
	well_1kgp3_chr13_exp	454739	18				well_1kgp3_chr13_chisq	454739	19

continued from previous page

PYTHON 1KG-P3/WELLDERLY CHI-SQUARE STATISTICS LOG

CHR	INPUT1	R1	C1	INPUT2	R2	C2	OUTPUT	R3	C3
chr14	sc_1kgp3_chr14_gen	2655067	8	well_chr14_gen	2072217	8	well_1kgp3_chr14	428638	12
	well_1kgp3_chr14	428638	12				well_1kgp3_chr14_exp	428638	18
	well_1kgp3_chr14_exp	428638	18				well_1kgp3_chr14_chisq	428638	19
chr15	sc_1kgp3_chr15_gen	2424689	8	well_chr15_gen	1725840	8	well_1kgp3_chr15	375584	12
	well_1kgp3_chr15	375584	12				well_1kgp3_chr15_exp	375584	18
	well_1kgp3_chr15_exp	375584	18				well_1kgp3_chr15_chisq	375584	19
chr16	sc_1kgp3_chr16_gen	2697949	8	well_chr16_gen	1717332	8	well_1kgp3_chr16	419272	12
	well_1kgp3_chr16	419272	12				well_1kgp3_chr16_exp	419272	18
	well_1kgp3_chr16_exp	419272	18				well_1kgp3_chr16_chisq	419272	19
chr17	sc_1kgp3_chr17_gen	2329288	8	well_chr17_gen	1697346	8	well_1kgp3_chr17	362718	12
	well_1kgp3_chr17	362718	12				well_1kgp3_chr17_exp	362718	18
	well_1kgp3_chr17_exp	362718	18				well_1kgp3_chr17_chisq	362718	19
chr18	sc_1kgp3_chr18_gen	2267185	8	well_chr18_gen	1751878	8	well_1kgp3_chr18	366047	12
	well_1kgp3_chr18	366047	12				well_1kgp3_chr18_exp	366047	18
	well_1kgp3_chr18_exp	366047	18				well_1kgp3_chr18_chisq	366047	19
chr19	sc_1kgp3_chr19_gen	1832506	8	well_chr19_gen	1351312	8	well_1kgp3_chr19	295926	12
	well_1kgp3_chr19	295926	12				well_1kgp3_chr19_exp	295926	18
	well_1kgp3_chr19_exp	295926	18				well_1kgp3_chr19_chisq	295926	19
chr20	sc_1kgp3_chr20_gen	1812841	8	well_chr20_gen	1252697	8	well_1kgp3_chr20	295170	12
	well_1kgp3_chr20	295170	12				well_1kgp3_chr20_exp	295170	18
	well_1kgp3_chr20_exp	295170	18				well_1kgp3_chr20_chisq	295170	19
chr21	sc_1kgp3_chr21_gen	1105538	8	well_chr21_gen	838257	8	well_1kgp3_chr21	170012	12
	well_1kgp3_chr21	170012	12				well_1kgp3_chr21_exp	170012	18
	well_1kgp3_chr21_exp	170012	18				well_1kgp3_chr21_chisq	170012	19
chr22	sc_1kgp3_chr22_gen	1103547	8	well_chr22_gen	726881	8	well_1kgp3_chr22	184233	12
	well_1kgp3_chr22	184233	12				well_1kgp3_chr22_exp	184233	18
	well_1kgp3_chr22_exp	184233	18				well_1kgp3_chr22_chisq	184233	19
chrX	sc_1kgp3_chrX_gen	3468093	8	well_chrX_gen	2658420	8	well_1kgp3_chrX	453523	12
	well_1kgp3_chrX	453523	12				well_1kgp3_chrX_exp	453523	18
	well_1kgp3_chrX_exp	453523	18				well_1kgp3_chrX_chisq	453523	19

References

[1] Sikora-Wohlfeld W, Basu AK, Butte AJ, Martinez-Canales M: Accelerating Secondary Genome Analysis Using Intel Reference Architecture. http://www.intel.com/content/www/us/en/high-performance-computing/high-performance-xeon-e5-genome-analysis-study.html 2014.

[2] Sikora-Wohlfeld W, Basu AK, Butte AJ, Martinez-Canales M: Statistical Analysis of Genome Sequencing Data with Intel Reference Architecture. http://www.intel.com/content/www/us/en/healthcare-it/statistical-analysis-genome-sequencing-paper.html; 2015.

[3] Scripps Translational Science Institute. http://www.stsiweb.org/wellderly/.

[4] 1000 Genomes Project Consortium, Abecasis GR, Auton A, Brooks LD, DePristo MA, Durbin RM et al. An integrated map of genetic variation from 1,092 human genomes. Nature. 2012;4917422:56-65.

[5] 1000 Genomes. http://www.1000genomes.org.

[6] Cloudera Hadoop Distribution. Available at http://www.cloudera.com/content/cloudera/en/products-and-services/cdh.html. Accessed on June 14, 2015.

[7] Cloudera Impala. Available at http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/impala.html. Accessed on June 14, 2015.

[8] Apache Hive. Available at https://hive.apache.org/. Accessed on June 14, 2015.

[9] Python. Available at https://www.python.org/. Accessed on June 14, 2015.

[10] Impyla Library. Available at https://github.com/cloudera/impyla. Accessed on June 14, 2015.

[11] Apache Thrift. Available at https://thrift.apache.org/. Accessed on June 14, 2015.

[12] HDFS. Available at https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html. Accessed on June 14, 2015.

[13] Intel® Xeon®. Available at http://www.intel.com/content/www/us/en/processors/xeon/xeon-processor-e7-family.html. Accessed on March 16, 2015.

Printed in USA 0815/AB/MIM/001/PDF Please Recycle 332922-001US