Rail-dbGaP: a protocol and tool for analyzing protected genomic data in a commercial cloud

Abhinav Nellore [1,2,3,*], Christopher Wilks [1,3], Kasper D Hansen [2,3], Jeffrey T Leek [2,3], and Ben Langmead [1,2,3,*]

[1]Department of Computer Science, Johns Hopkins University, 3400 N Charles St, Baltimore MD, 21218, [2]Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health, 615 N Wolfe St, Baltimore MD, 21205, [3]Center for Computational Biology, Johns Hopkins University, 1900 E Monument St, Baltimore, MD 21205

ABSTRACT

Motivation: Public archives contain thousands of trillions of bases of valuable sequencing data. More than 40% of the Sequence Read Archive is human data protected by provisions such as dbGaP. To analyze dbGaP-protected data, researchers must typically work with IT administrators and signing officials to ensure all levels of security are implemented at their institution. This is a major obstacle, impeding reproducibility and reducing the utility of archived data.

Results: We present a protocol and software tool for analyzing protected data in a commercial cloud. The protocol is applicable to any MapReduce tool running on Amazon Web Services. The tool, Rail-RNA v0.2, is a spliced aligner for RNA-seq data, which we demonstrate by running on 9,662 samples from the dbGaP-protected GTEx consortium dataset. These are important first steps toward making it easy for typical biomedical investigators to study protected data, regardless of their local IT resources or expertise.

Availability: Rail-RNA is available from http://rail.bio, and detailed instructions on running Rail-RNA on dbGaP-protected data using Amazon Web Services are available at http://docs.rail.bio/dbgap/.

Contact: anellore@gmail.com, langmea@cs.jhu.edu

1 INTRODUCTION

The Database of Genotypes and Phenotypes (dbGaP) (Mailman *et al.*, 2007) hosts controlled-access raw and processed human genomic data and associated phenotypic data. While de-identified, they are sensitive, and the NIH requires adherence to security guidelines for their proper handling, from data acquisition through data destruction (NIH, 2015a). These include physical security of computing infrastructure, restricting inbound internet access, multi-factor authentication (MFA) and password policies, encryption of data, enforcing the principle of least privilege, and logging data access. For many investigators, the guidelines pose a practical challenge: institutional computer clusters may not be compliant, requiring IT policy revisions or intervention by IT administrators.

The recent announcement (NIH, 2015b) allowing investigators to request permission to transfer to and analyze dbGaP data in compliant clouds provides a convenient alternative: investigators can use protocols and software tailored for secure analysis of dbGaP data in a commercial cloud, bypassing local infrastructure issues.

Commercial cloud providers allow users to rent computational and storage resources residing in large data centers. Reputable providers like Amazon, Microsoft, and Google ensure physical security of data centers. Resources come in standardized units of hardware and software; the hardware is rented in standard (usually virtualized) units called instances and software comes pre-installed on standard disk images selected at launch time.

Amazon Web Services (AWS) is a popular choice for genomics data analysis. Key datasets such as 1000 Genomes, TCGA, and ICGC are hosted in AWS storage, allowing researchers to use cloud-enabled tools without copying data across the internet (AWS, 2015a,b). Input data, intermediate results, and final results can all be kept in cloud storage before final results are downloaded or browsed. AWS also provides guidance on secure analysis of protected genomic data (Pizarro and Whalley, 2014).

We describe two important first steps toward enabling typical investigators to analyze dbGaP-protected data on AWS: (1) Rail-dbGaP, a protocol for analyzing dbGaP data in the cloud using Elastic MapReduce (EMR), an AWS service; and (2) the v0.2 line of our Rail-RNA tool, which implements the protocol to securely download and align many dbGaP-protected RNA sequencing (RNA-seq) samples at once on EMR. We demonstrate the tool by running it on 9,662 RNA-seq samples from the GTEx project.

2 SECURITY ARCHITECTURE

An EMR Hadoop (Apache, 2015) cluster consists of a master instance and several worker instances. The Rail-dbGaP security architecture (Fig. 1) secures the cluster to satisfy dbGaP guidelines:

- **Cluster is within a subnet of a Virtual Private Cloud (VPC).** A VPC is a logically isolated unit of the cloud providing a private network and firewall. The connection with the cloud storage service (Amazon Simple Storage Service, or S3) is via a "VPC endpoint," which ensures that data transferred never leaves the data center. Each instance has a public IP, though EMR now permits leaving public IPs unassigned and instead routing internet traffic to and from the cluster through a Network Address Translation (NAT) gateway. A question for future investigation is whether improving "defense in depth" this way would significantly impact convenience or speed.
- **Inbound traffic is restricted via security groups.** A master security group for the master instance and a worker security group for worker instances prevent initiation of any connection to the cluster except by essential web services. A security group is essentially a stateful firewall. On EMR, these web services correspond to particular IPs and ports, and the most restrictive sets for master and worker instances are configured automatically. SSH access to the cluster is also restricted: on EMR, the only interaction

*to whom correspondence should be addressed

between user and cluster is via the EMR interface, which presents progress information through the essential web services.

• **Data are encrypted at rest.** During cluster setup, before any sensitive data has reached the cluster, each instance runs a preliminary script ("bootstrap action") that uses Linux Unified Key Setup (LUKS) (Fruhwirth, 2011) to create an encrypted partition with a keyfile. The key is randomly generated on each instance and never exposed to the user. Temporary files, the Hadoop distributed file system, and buffered output to the cloud storage service are all configured to reside on the encrypted partition via symbolic links. Files written to cloud storage are also encrypted. On S3, this is enforced by creating bucket policies barring uploads that do not turn on server-side encryption.

• **Data are encrypted in transit.** Worker instances download dbGaP data using SRA Tools (NCBI, 2015) ensuring encryption of data transferred from dbGaP to the cluster. Secure Sockets Layer (SSL) is enabled for transfers between cloud storage and the cluster as well as between cloud storage service and compliant local storage to which an investigator saves results.

• **Identities are managed to enforce the principle of least privilege.** The principle of least privilege prescribes users have only the privileges required to perform necessary tasks. In the

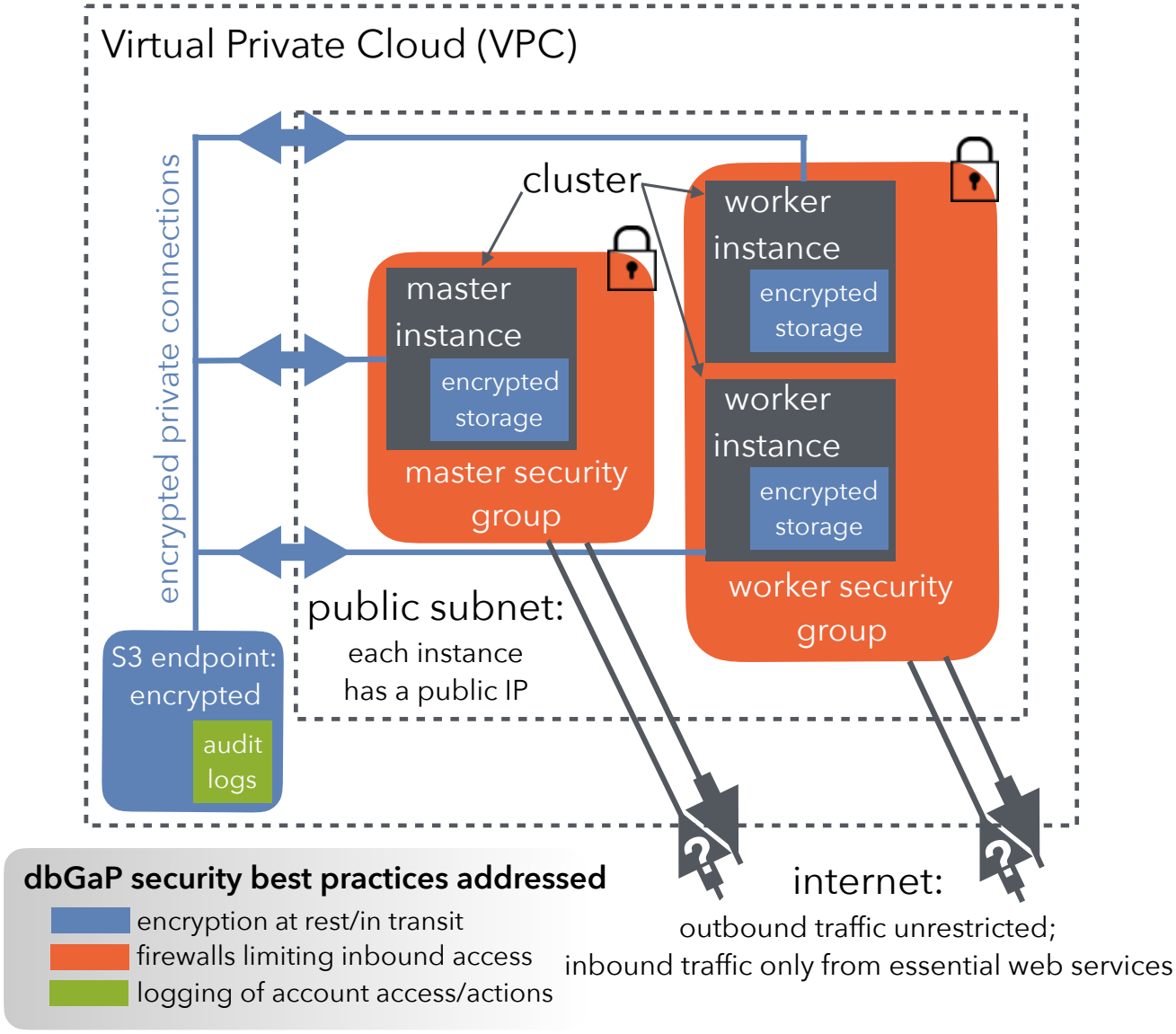

Fig. 1. The Rail-dbGaP security architecture. Security features include a Virtual Private Cloud with a private connection between the computer cluster and cloud storage, audit logs recorded on cloud storage, encryption of sensitive data at rest and in transit, and restricted inbound access to the cluster via security groups.

Rail-dbGaP protocol, an administrator grants the user only those privileges required to run Hadoop programs on EMR clusters. The administrator uses multi-factor authentication and constrains the user to set up a password satisfying NIH's requirements listed in the document (NIH, 2015a) (minimum of 12 characters, no complete dictionary words, etc.) On AWS, an account administrator configures an Identity and Access Management (IAM) user expressly for running Hadoop jobs and retrieving results from S3, and the password rules described above are enforced.

• **Audit logs are enabled.** These record logins and actions taken by the user and on the user's behalf, including API calls made by processes running on the cluster. On AWS, audit logs take the form of CloudTrail logs stored in encrypted S3 buckets.

3 PROTOCOL

The Rail-dbGaP protocol proceeds in a few steps, described in greater depth in the Rail-RNA dbGaP tutorial at http://docs.rail.bio/dbgap/.

First, the administrator configures the account users, roles and password policies to conform to dbGaP security requirements. Second, the administrator creates a CloudFormation stack based on a dbGaP CloudFormation template supplied with Rail-RNA. The stack describes all the infrastructure pictured in Fig. 1 except the instances themselves. Third, the administrator delegates EMR-related authorities to the user. All of the steps described so far need only be executed once per AWS account.

Now the user launches an EMR cluster, creating the instances and starting the analysis. When the user launches the cluster, they must identify the name of the CloudFormation stack created in step 2 above, which links the EMR cluster to the security architecture depicted in Fig. 1.

4 APPLICATION

The Rail-dbGaP protocol is implemented in the v0.2 line of Rail-RNA. Detailed instructions are available at http://docs.rail.bio/dbgap/, and Rail-RNA v0.2.1 may be downloaded at http://rail.bio. We used the Rail-dbGaP protocol to align 9,662 paired-end RNA-seq samples obtained by the GTEx consortium (Lonsdale *et al.*, 2013) in 30 batches across various human tissues over a period of 5 days, 5 hours, and 28 minutes. A total of 896,466,227,499 reads were analyzed, costing US$0.32 per 10 million reads, from downloading raw data from dbGaP to transferring results to local storage.[2] Scripts for reproducing our results are available at https://github.com/nellore/runs/tree/master/gtex.

ACKNOWLEDGEMENTS

We thank the GTEx project for making data available. We thank Christopher Goodson, Lenworth Henry, and Angel Pizarro of Amazon Web Services and Ronald Dowden and Brian Willey of IT@JH for assistance in designing and validating the protocol.

Funding: AN, JTL, and BL were supported by NIH/NIGMS grant 1R01GM105705 to JTL. AN was supported by a seed grant from the Institute for Data Intensive Engineering and Science (IDIES) at Johns Hopkins University to BL and JTL. BL was supported by a Sloan Research Fellowship to BL. Amazon Web Services experiments were supported by AWS in Education Research grants.

[2] Here, "read" refers to a mate for paired-end samples.

REFERENCES

Apache (2015). Apache Hadoop. http://hadoop.apache.org.

AWS (2015a). 1000 Genomes Project and AWS. https://aws.amazon.com/1000genomes/.

AWS (2015b). TCGA on AWS. https://aws.amazon.com/public-data-sets/tcga/.

Fruhwirth, C. (2011). LUKS on-disk format specification. https://gitlab.com/cryptsetup/cryptsetup/wikis/LUKS-standard/on-disk-format.pdf.

Lonsdale, J., Thomas, J., Salvatore, M., Phillips, R., Lo, E., Shad, S., Hasz, R., Walters, G., Garcia, F., Young, N., *et al.* (2013). The genotype-tissue expression (gtex) project. *Nature genetics*, **45**(6), 580–585.

Mailman, M. D., Feolo, M., Jin, Y., Kimura, M., Tryka, K., Bagoutdinov, R., Hao, L., Kiang, A., Paschall, J., Phan, L., *et al.* (2007). The ncbi dbgap database of genotypes and phenotypes. *Nature genetics*, **39**(10), 1181–1186.

NCBI (2015). SRA Tools protected data usage guide. http://ncbi.github.io/sra-tools/pd_usage_guide.html.

NIH (2015a). NIH Security Best Practices for Controlled-Access Data Subject to the NIH Genomic Data Sharing (GDS) Policy. http://www.ncbi.nlm.nih.gov/projects/gap/pdf/dbgap_2b_security_procedures.pdf.

NIH (2015b). Notice for Use of Cloud Computing Services for Storage and Analysis of Controlled-Access Data Subject to the NIH Genomic Data Sharing (GDS) Policy. https://grants.nih.gov/grants/guide/notice-files/NOT-OD-15-086.html.

Pizarro, A. and Whalley, C. (2014). Architecting for genomic data security and compliance in AWS. https://d0.awsstatic.com/whitepapers/compliance/AWS_dBGaP_Genomics....

Abhinav Nellore[1,2,3,*], *Christopher Wilks*[1,3], *Kasper D Hansen*[2,3], *Jeffrey T Leek*[2,3], *and Ben Langmead*[1,2,3,3]

5 SUPPLEMENTARY MATERIAL

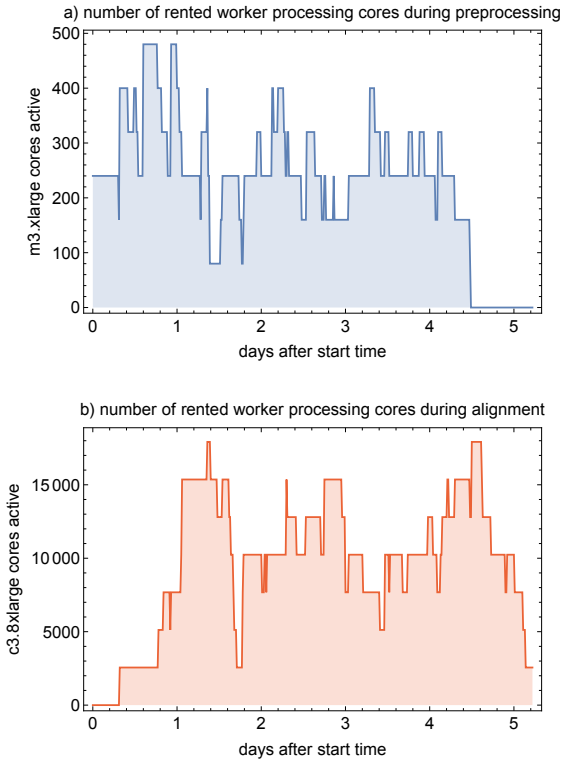

Fig. 2. The number of rented worker cores across clusters running a) preprocess and b) alignment job flows for the duration of all job flows.

5.1 Computation details

We analyzed 9,662 RNA-seq samples from the V6 release by the GTEx consortium. These were divided randomly into 30 batches of approximately the same size: two batches had 323 samples, and the others had 322 samples. Rail-RNA splits analysis of each batch up into a preprocess job flow, which securely downloads raw reads from dbGaP and uploads preprocessed versions to S3; and an alignment job flow, which aligns preprocessed reads and writes results back to S3. Preprocess job flows were run on 21 m3.xlarge Amazon EC2 instances, each with four 2.4 GHz Intel Xeon E5-2670 v2 (Ivy Bridge) processing cores and 15 GB of RAM. Alignment job flows were run on 81 c3.8xlarge Amazon EC2 instances, each with 32 Intel Xeon E5-2680 v2 (Ivy Bridge) processing cores and 60 GB of RAM. One instance of every EMR cluster was a master and the rest were workers, so up to 80 worker processing cores were active for each preprocess job flow and up to 2560 worker processing cores were active for each alignment job flow. All EC2 instances were obtained from the EC2 spot marketplace, which allowed us to reduce our cost to the fluctuating market price, typically a fraction of the standard (on-demand) price. Job flows were submitted manually, with preprocess job flows staggered to minimize issues with simultaneous downloads from the dbGaP server. The number of rented worker cores across the 5 days, 5 hours, and 28 minutes during which job flows were run is summarized in Fig. 2.

5.2 Cost calculations

We used the AWS Cost Explorer to sum costs across the five days over which the computation ran: November 30, 2015 through December 4, 2015. The total cost of our analysis was US$28,368.15. We minimized storage costs by deleting analysis intermediates on S3 as soon as job flows were finished and deleting results on S3 as soon as they were downloaded to local storage. Raw cost data downloaded from the AWS Cost Explorer are available at https://github.com/nellore/runs/blob/master/gtex/costs.csv. Costs broken down by AWS service are depicted in Fig. 3. To obtain the cost US$0.32 per 10 million reads aligned, we divided the the total cost US$28,368.15 by the total number of reads 896,466,227,499, multiplied by 10 million, and rounded to the nearest cent.

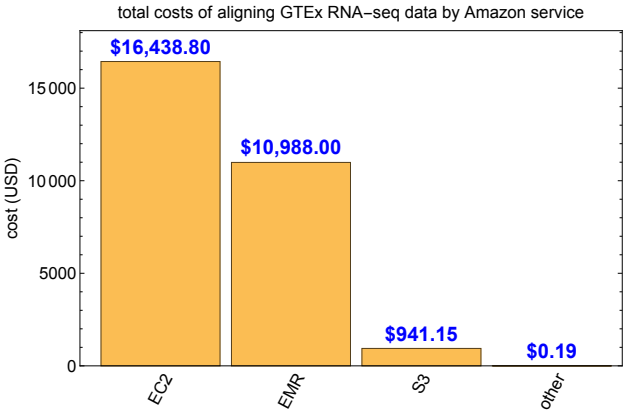

Fig. 3. Total costs of GTEx analysis jobs divided up by Amazon service. The trivial contributions of the “other” costs are from Simple Queue Service (SQS) and SimpleDB.