BIOINFORMATICS REVIEW

Vol. 29 no. 17 2013, pages 2075–2083
doi:10.1093/bioinformatics/btt352

Sequence analysis

Advance Access publication June 20, 2013

Harnessing virtual machines to simplify next-generation DNA sequencing analysis

Julie Nocq[1,2,†], Magalie Celton[1,2,3,†], Patrick Gendron[1], Sebastien Lemieux[1,4] and Brian T. Wilhelm[1,2,*]

[1]Institute for Research in Immunology and Cancer, University of Montreal, [2]Laboratory for High-Throughput Genomics, Department of Medicine, University of Montreal, QC H3T 1J4, Canada, [3]INRA, UMR1083, Sciences pour l'Oenologie, Montpellier, France and [4]Laboratory for Functional and Structural Bioinformatics, Computer Sciences and Operation Research, University of Montreal, QC H3T 1J4, Canada

Associate Editor: Jonathan Wren

ABSTRACT

Motivation: The growth of next-generation sequencing (NGS) has not only dramatically accelerated the pace of research in the field of genomics, but it has also opened the door to personalized medicine and diagnostics. The resulting flood of data has led to the rapid development of large numbers of bioinformatic tools for data analysis, creating a challenging situation for researchers when choosing and configuring a variety of software for their analysis, and for other researchers trying to replicate their analysis. As NGS technology continues to expand from the research environment into clinical laboratories, the challenges associated with data analysis have the potential to slow the adoption of this technology.

Results: Here we discuss the potential of virtual machines (VMs) to be used as a method for sharing entire installations of NGS software (bioinformatic 'pipelines'). VMs are created by programs designed to allow multiple operating systems to co-exist on a single physical machine, and they can be made following the object-oriented paradigm of encapsulating data and methods together. This allows NGS data to be distributed within a VM, along with the pre-configured software for its analysis. Although VMs have historically suffered from poor performance relative to native operating systems, we present benchmarking results demonstrating that this reduced performance can now be minimized. We further discuss the many potential benefits of VMs as a solution for NGS analysis and describe several published examples. Lastly, we consider the benefits of VMs in facilitating the introduction of NGS technology into the clinical environment.

Contact: brian.wilhelm@umontreal.ca

Received on March 13, 2013; revised on June 4, 2013; accepted on June 14, 2013

1 INTRODUCTION

1.1 Next-generation sequencing

In recent years, novel approaches in the development of massively parallel DNA sequencing chemistry technologies have resulted in a dramatic increase in the amount of DNA sequence data that is being produced (Mardis, 2011). As a result, these novel sequencing machines, often called second-generation or next-generation sequencing (NGS) technologies, have simultaneously reduced the cost of sequencing a human genome by a factor of >1 million, while reducing the time required from years to a single day. Moreover, this process of improvement is continual: the amount of sequencing data obtained from an Illumina DNA sequencer, for instance, has quintupled in 2009 alone (Langmead *et al.*, 2009), and to date, this trend shows no signs of reaching a plateau. In addition, the cost of sequencing hardware has also decreased, resulting in an extremely rapid adoption of this (Stein, 2010).

Although the scientific value of this immense and ongoing flood of data has been profound, it has created dramatic new bioinformatic challenges in managing and analyzing the resulting data. The 'mapping' of short DNA sequences ('sequence reads') to a reference genome, a frequent first step of analysis, typifies the problem. Although traditional DNA mapping software, such as BLAST (Altschul *et al.*, 1990) and BLAT (Kent, 2002), are highly accurate, they operate at speeds that are orders of magnitude slower than the rate at which NGS sequence data can be generated (Li *et al.*, 2008a, b). This large (and continually growing) asymmetry between data generation and analysis capacity has forced a rapid evolution in the algorithms for read mapping. A recent example is provided by the Spliced Transcripts Alignment to a Reference (Dobin *et al.*, 2012) short read-mapper, which can process >550 million 76 bp reads/h, on a 12 core server. The requirement to rapidly adapt or develop bioinformatic tools suitable for the volume and nature of NGS data being generated by large-scale genomics projects worldwide is not limited to read mapping, but also implicates every step of the downstream analysis. An enormous collection of tools (both commercial and open-source) for NGS analysis are currently available, a small subset of which are shown in Table 1, grouped by analysis task.

Even as the bioinformatics community has clearly responded to the need to develop new tools for NGS analysis, the diversity of available tools has also paradoxically created another level of complexity, hindering the potential impact of this technology. In a typical NGS analysis, for instance, 5–8 separate bioinformatic tools might be required to map reads, call sequence variants, analyze splicing patterns, assess differential expression and perform gene set enrichment analysis [e.g. DAVID (Da Wei Huang and Lempicki, 2008)]. Ideally, these tools would be installed and

*To whom correspondence should be addressed.
†The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint First Authors.

Table 1. Selection of various NGS programs and analysis packages

	Tools	Reference
Read mapping	BFAST	(Homer *et al.*, 2009)
	MAQ	(Li *et al.*, 2008a, b)
	BWA	(Li and Durbin, 2009)
	Bowtie	(Langmead *et al.*, 2009)
	SHRiMP	(Rumble *et al.*, 2009)
	ELAND2	(Cret *et al.*, 2009)
	SOAP2	(Li *et al.*, 2009)
	SSAHA2	(Ning *et al.*, 2001)
Read assembly	Velvet	(Zerbino and Birney, 2008)
	SSAKE	(Warren *et al.*, 2007)
	ABySS	(Simpson *et al.*, 2009)
	Trans-ABySS	(Robertson *et al.*, 2010)
	AllPaths	(Butler *et al.*, 2008)
Variant calling	SamTools	(Li, 2011)
	VarScan	(Koboldt *et al.*, 2009)
	SNVMix	(Goya *et al.*, 2010)
	VarSifter	(Teer *et al.*, 2012)
	SomaticSniper	(Larson *et al.*, 2012)
Copy number analysis	CNV-seq	(Xie and Tammi, 2009)
	SegSeq	(Chiang *et al.*, 2008)
	CNVnator	(Abyzov *et al.*, 2011)
	SeqCBS	(Shen and Zhang, 2012)
	ExomeCNV	(Sathirapongsasuti *et al.*, 2011)
Differential gene expression	DEseq	(Anders and Huber, 2010)
	DEGseq	(Wang *et al.*, 2010)
	Cufflinks	(Trapnell *et al.*, 2012)
	edger	(Robinson *et al.*, 2010)
	baySeq	(Hardcastle and Kelly, 2010)
	NOISeq	(Tarazona *et al.*, 2011)
Alternative splicing	TopHat	(Trapnell *et al.*, 2009)
	AltAnalyze	(Salomonis *et al.*, 2009)
	MATS	(Shen and Zhang, 2012)
ChIP peak finding	PeakSeq	(Rozowsky *et al.*, 2009)
	MACS	(Zhang *et al.*, 2008)
	FindPeaks	(Fejes *et al.*, 2008)
Visualization tools	SAMSCOPE	(Popendorf and Sakakibara, 2012)
	IGV	(Robinson *et al.*, 2011)
	UCSC Browser	(Karolchik *et al.*, 2009)
	Artemis	(Rutherford *et al.*, 2000)
	Genoview	(Laczik *et al.*, 2012)

configured in such a way that the raw sequence data could be pushed through each of the tools sequentially (in a bioinformatic 'pipeline') in a largely automated way, to generate biologically meaningful results. Creating such a pipeline first requires end users to download a number of individual programs, in addition to dependencies, such as gene annotation files, while ensuring that the format of the annotations is compatible with all of the tools. Using multiple NGS analysis programs also presents other challenges: despite many tools having a standard form of input, in many cases the resulting output from one program cannot be directly used in a subsequent tool without additional (often minor) reformatting to ensure compatibility. Furthermore, most NGS programs have a wide variety of specific parameters that need to be set, all of which may produce subtle or gross differences in the output. Although default values are generally included, end users may not often understand the precise impact of changing these values for their analysis without investing substantial time to perform systematic testing. A final complication in this scenario arises from the constant and asynchronous development and release of individual software tools, which can lead to losses of compatibility and breakdowns in established analysis pipelines when software is updated. The end result of this situation is that the process of establishing and maintaining a bioinformatic pipeline is neither trivial nor does it foster reproducibility of analysis results, given the wide variety of tools and parameters involved (Nekrutenko and Taylor, 2012).

The impact of the complexity of NGS bioinformatic analysis is not limited to difficulties in extracting biological meaning from experiments, but it may also directly impede the adoption of NGS technology (e.g. within clinical settings where it has already been demonstrated to have enormous utility). Indeed, novel DNA sequencing technologies have empowered the concept of 'personalized medicine', whereby medical treatment is tailored to differences in individual patient genomes (Schweiger *et al.*, 2011). Despite the success of NGS-based studies of cancer (Pfeifer and Hainaut, 2011) and rare diseases (Ng *et al.*, 2010), the use of sequencing-based diagnostics is still lagging far behind its use in research. Part of the reason for this is certainly the validation of such techniques, but another significant factor is likely to be the limited bioinformatics resources available within hospitals to perform sequence analysis. Although general IT resources are likely to be available in major hospitals and clinics, the specialized knowledge to install and optimize NGS bioinformatics software, and to configure these for a variety of clinical uses (e.g. cancer genomics, bacterial meta-genomics), is unlikely to exist. As a result of this lack of resources, the potential benefits of NGS technology will undoubtedly take longer to arrive as a standard of practice for medical diagnostics.

In light of the problems highlighted above, there are several basic requirements that all bioinformatics tools developed for NGS analysis should ideally fulfill to offer the greatest benefit to the biomedical community:

(1) **Compatibility:** Within the domain of NGS analysis, the data generated are principally images and DNA sequences derived from these images, although in practice the primary images are no longer retained. The past 5–6 years have seen the emergence of *de facto* standards for sequence files (FASTQ) (Li *et al.*, 2008), sequence alignments [sequence alignment/map (SAM) or binary alignment/map (BAM) files] (Li *et al.*, 2009) and sequence variants (variant call formats). Despite this 'common currency' for information exchange, not all programs accept every one of these formats, and some trivial differences [such as slight variations in quality scores in FASTQ files, lack of standard file extension (.fq versus .fastq), output format, etc] can impact compatibility between programs.

(2) **Facility to reproduce analysis:** At present, a majority of NGS tools was designed to be used on a command line in a Unix environment. The effort involved in exactly recreating a specific configuration for a server with identical software and annotation used for a published analysis

Table 2. Selection of integrated packages for NGS analysis

	Tools	Reference	URL	Open-source?	Clinically certified?	Read mapping?	Cost model
Integrated packages	Avadis NGS	Inc., San Francisco, CA, USA	www.avadis-ngs.com	No	No	Yes	Annual licence
	Genespring	Agilent Technologies	genespring-support.com	No	No	No	Annual licence
	Goldenhelix	Inc. Bozeman, MT, USA	www.goldenhelix.com	No	No	No	Annual licence
	Galaxy	(Goecks *et al.*, 2010)	https://main.g2.bx.psu.edu/root	Yes	No	Yes	Free; storage limit
	CLC Genomics	Aarhus, Danemark	www.clcbio.com	No	No	Yes	Annual licence
	Genomatix	Genomatix Software GmbH	www.genomatix.de/	No	No	Separate	Annual licence
	NextGENe	SoftGenetics, State College, PA	www.softgenetics.com/ NextGENe.html	No	No	Yes	Annual licence

represents a significant barrier for direct comparisons between experimental datasets.

(3) **Open-source tools:** There is a broad understanding that publicly funded academic software should be freely available to the entire research community, and the distribution of the source code obviously has numerous advantages. The distribution of open-source software not only allows other researchers the opportunity to critically evaluate the underlying algorithm, but examination of the methodology used can also serve as a mechanism to stimulate research into novel approaches to improve performance.

1.2 Centralized resources and cloud computing

To address these concerns, a practical method should be developed to distribute NGS tools that ensure that all tools work compatibly and allow a comprehensive analysis. An excellent example of such a package is Genome Analysis Tool Kit (GATK) (McKenna *et al.*, 2010) developed by the Broad Institute that, excluding initial read mapping, allows users to perform an integrated analysis using a single application. Individual tools within GATK (called ‘walkers’) must still be run individually at the command line, however, using all appropriate parameters. Other commercial software packages that allow for an integrated analysis are also available (Tables 1 and 2); however, the license fees required and proprietary/closed source code can be disincentives for their use by some laboratories. Simultaneously, the development of centralized resources, in contrast to tools meant primarily to be downloaded by end users, have also been developed. One popular example of such a resource is Galaxy, an open-source web-based platform developed by Penn State and Emory University for data analysis and manipulation (Goecks *et al.*, 2010). Galaxy was developed in 2005, and the published description of the system 5 years later notes that the public web server processes ~5000 jobs/day, clearly demonstrating the demand for tools for large-scale integrative analysis. Within Galaxy, multiple steps in an analysis method can be saved and shared as a ‘workflow’, enabling others to precisely replicate the same process used by another group. Despite such useful features, Galaxy still requires users to either upload their data to the public Web site (presenting practical limitations when working with large amounts of raw sequence data) or to download the Galaxy software and configure it to run locally. Because the Galaxy software is currently not distributed containing a suite of bioinformatic software, a local installation presents similar issues for configuration of all of the individual software tools as discussed above. Although the Galaxy interface can be customized to include whatever tools are required, it is not clear that the system was designed to meet Clinical Laboratory Improvement Amendments/Health Insurance Portability and Accountability Act certification requirements for use within clinical settings (although it is possible that it could be modified in such a way as to do this). Therefore, despite its utility and popularity, Galaxy does not at the moment represent a simple, hands-off and pre-configured solution for NGS analysis for all potential users.

Because of the significant computational resources required for NGS analysis, there has also been growing interest in the use of cloud computing to address current bioinformatic bottlenecks. The dramatic growth in remote computing services by such companies as Amazon (Amazon EC2), Microsoft (Windows Azure), Rackspace (Rackspace cloud) and Google (Google Cloud platform) now means that researchers have the option of simply paying for their computing requirements on an *ad hoc* basis, rather than building and maintaining their own physical computing infrastructure. For many small laboratories, this represents a highly cost-effective solution, especially when their demands for such hardware are highly periodic. These same cloud resources can be used for large-scale data storage and sharing, which is also an attractive option for small to mid-sized laboratories that do not wish to invest in computer hardware. The costs associated with the use of cloud computing for storage and analysis are low enough that it would not represent a barrier to their use, although the research community will likely still prefer ‘permanent’ repositories for data (e.g. Gene Expression Omnibus (GEO)/SRA at NCBI, Arrayexpress at EBI) for long-term data storage. Although cloud computing does offer many advantages for NGS analysis, there have been sporadic large-scale failures of cloud resources (e.g. Amazon EC2) in the past that have left popular hosted web services unavailable. In the context of highly critical, rapid turnaround diagnostic sequencing, the potential risk of such outages for cloud-based clinical NGS analysis might be considered too high.

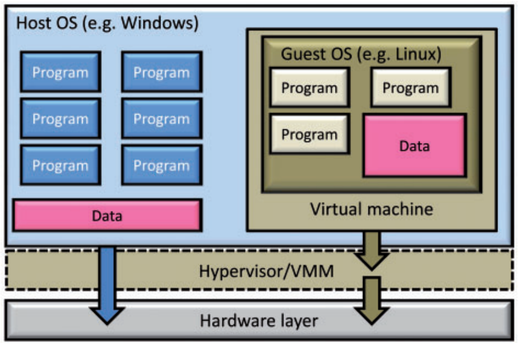

Fig. 1. Schematic view of a VM. A typical system VM and the guest operating system (guest OS) will exist in the form of a number of files on the hard drive of the host operating system (host OS). The guest OS is accessed through software that runs through the hypervisor or virtual machine manager (VMM), which interprets instructions from the guest OS to execute these on the underlying hardware. Data and programs of the host and guest OS generally exist in strong isolation even on the same hardware, allowing large numbers of operating systems to co-exist independently

2 VIRTUAL MACHINES

Another potential solution to the problems with NGS data analysis has arisen through an unrelated advance in computer software, namely, the development of virtual machines (VMs). The basic principle of a VM is the use of software to emulate the architecture, behavior or functionality of another computer system. As discussed below, because VMs can be created and distributed already containing the software that will run within the VM, they can be used to make collections of NGS tools available as a single downloadable unit.

2.1 Development of VMs

The implementation of the concept of a VM has evolved from pioneering work carried out by International Business Machine (IBM) in the 1960s and 70s (Creasy, 1981), with their development of VM operating systems for mainframe computers, to the enormous range of systems currently available. Although the term 'virtual machine' is used to describe a wide variety of systems, they can be broadly classified into two categories, based on the level for which they provide abstraction (Smith and Nair, 2005). For the purpose of encapsulating analysis pipelines, we will focus on system VM where, using a virtual machine manager (VMM), an entire hardware platform is emulated. The VM thus encompasses software tools, libraries and operating systems and, when active, reserves both physical disk space and RAM on the underlying hardware (Fig. 1).

Such VMs enable what are termed 'guest operating systems' (guest OS) to simultaneously operate on the same hardware as a native OS, but in strong isolation. Because an active VM is seen as a simple process to the host OS, it can readily be interrupted and its exact state saved to a file. This 'snapshot' can then be archived and reactivated at any time, including after being transferred to a different hardware platform. A number of different VMMs have been developed over the past decade (Table 3). Because of their capacity to abstract and share underlying hardware, the development of VM has also largely underpinned the growth of cloud computing services offered by companies such as Amazon, Google, Microsoft and others. The growth of VMs has also been aided through efforts of commodity processor manufacturers [e.g. Integrated Electronics (Intel), Advanced Micro Devices (AMD)] to include hardware support for virtualization, in the form of new processor extensions to the x86 architecture. This has led to substantial improvements in both performance and flexibility of VMs. This growth in support, along with the utility of VMs within a variety of commercial domains, has created what is now a highly competitive market for VMMs. Interestingly, there has been little adoption of this software within the setting of academic and biomedical research, despite its potential to address the problems discussed above.

2.2 VMs as a solution for NGS software distribution

The application of VMs for the distribution of tools for NGS analysis has a number of potentially compelling advantages, largely because it offers the possibility of following an object-orientated paradigm of encapsulating data and methods together. Because VMs can recreate all aspects of an entire OS, it is possible for such virtual systems to be created already containing a variety of preconfigured bioinformatic software. The distribution of such a VM would then allow end users to circumvent issues of ensuring compatibility between various versions of software and compiling and configuring such tools. Given that many research centers with large NGS projects will likely have already invested substantial resources in testing and optimizing software choices and parameters for their analysis, a VM built to reflect this testing would allow many other centers to benefit from this effort and avoid 'reinventing the wheel'.

Indeed, this idea of a central repository of validated file versions is similar to the approach taken for the distribution of various flavors of Linux. At the same time, the distribution of VMs containing bioinformatic pipelines, rather than individual tools, would also enable researchers to more easily replicate precise analysis steps performed by other groups, helping to ensure that conclusions drawn from such comparisons are not artifacts of software parameters used for analysis. This is facilitated by the fact that NGS-VMs exist as a collection of files, making it simple to save a 'snapshot' of the NGS–VM used for a publication, which can then be directly shared with other laboratories. An illustration of the differences in current and potential VM-based NGS software distribution is shown in Figure 2. For archiving, repositories such as GEO and ArrayExpress, which already store NGS data (which represents 99% of the space required for an NGS–VM), it should be straightforward to store the data within a VM, allowing end users to download and rerun exactly the same analysis as the original authors. Such transfers could also be made directly to cloud compute resources to facilitate analysis or reuse of the published methods with new data.

Although such 'NGS–VMs' would not, by themselves, remove issues of different NGS software choices between groups, it is entirely likely that there would be coalescence around a few specific VMs developed by large institutes (e.g. genome centers), in the same way as has occurred for individual NGS tools [e.g. BWA (Li and Durbin, 2009) or Bowtie (Langmead *et al.*, 2009) for read mapping].

Table 3. List of virtual machine software

	System requirements			Guest OS supported																									
VM Software Name	GB of RAM (Min, Recommended)	Minimum HD space	Host OS: (W)indows, (L)inux, (O)S X	Windows 8 (32/64-bit)	Windows 7 (32/64-bit)	Windows Vista (32/64-bit)	Windows 2000	Windows XP (32/64-bit)	Windows NT	Windows 98	Ubuntu 5.10 /6.06 Desktop /7.04 - 12.04	Debian 6.0	SUSE 9/10.0	openSUSE 11.0-11.3	Mandriva 2009.0/2009.1	Mandrake 10.1	Fedora Core 1/4/5/6	RHEL6, Oracle Linux 6	Red Hat Linux 9	Xandros 3/4	ArchLinux	Oracle Solaris	Solaris 10 5.08 and later	Solaris 11	Mac OS X Server (Leopard, Snow Leopard)	FreeBSD	OpenBSD	DOS	OS/2
VirtualBox	1	NS	W,L,O	▨	■	■	■	■	■	▨	■	■	■	■	■	■	■	■	▨	■	■	□	■	■	■	▨	▨	▨	■
Vmware	NS*	>1 GB	W,L	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	□	□	□	■	■	■	■	□	■	■
Parallels Desktop	2, 4	15 GB	O	■	■	■	■	■	■	■	■	■	■	■	■	□	■	■	□	□	□	□	■	■	■	■	□	■	■
Windows Virtual PC	2,>2	15 GB	W	■	■	■	□	■	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
Oracle VM Server	1,2	6 GB	Not required	□	■	■	□	■	□	□	□	□	□	□	□	□	□	■	□	□	□	■	□	□	□	□	□	□	□
KVM	NS*	>1 GB	L	□	■	■	■	■	■	□	■	■	■	■	□	□	■	■	■	■	■	□	■	■	□	■	■	■	□

■ supported □ not supported ▨ reported to be supported

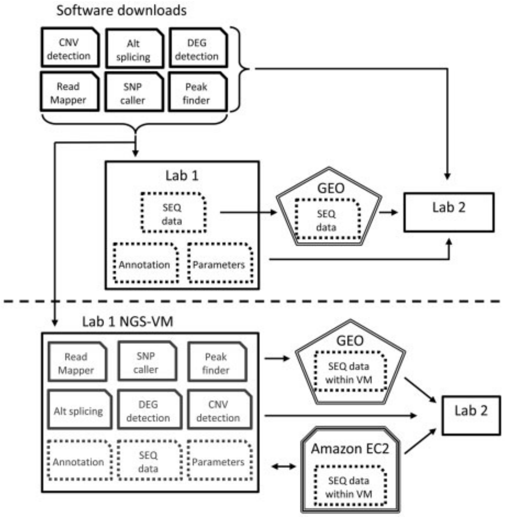

Fig. 2. Comparison of NGS software distribution methods. Traditional methodologies require individual laboratories to independently download (black arrows) identical versions of software and then configure them using equivalent parameters. Raw sequencing data are generally transferred from central repositories (e.g. GEO at NCBI), wheras annotation data and parameters are obtained from the laboratory or publication. Alternatively, an NGS–VM used by the first laboratory (below black dotted line) could be redistributed as a single unit, either directly or through storage intermediates, that could include the annotation, precise parameters and the data used

Even with this possibility, however, it would remain straightforward for end users to download such a VM and adjust any of the parameters for the bioinformatic tools within the VM exactly as they would for the tool in a native system. Given the relatively small size of most NGS software, a slightly more sophisticated approach for NGS–VMs could be to include a wide variety of software (e.g. several mappers and several variant callers) such that multiple pipelines could be available to end users. Indeed, given the fairly universal availability of high-speed Internet connections, even the distribution of NGS–VMs containing preinstalled genomes and sequence databases could be a practical solution. The development of scientific peer-to-peer networking tools, such as BioTorrents (Langille and Eisen, 2010), could also simplify the distribution of large NGS–VMs. With the addition of a simple graphical user interface to direct the underlying bioinformatic tools and access parameters for the programs, NGS–VMs could dramatically simplify the process of performing NGS data analysis.

2.3 Published examples of bioinformatics VMs

The potential application of NGS–VMs in research has already started to be recognized with the development of several examples of systems to facilitate specific types of analyses. For example, a recent publication described Cloud BioLinux, a publicly accessible VM composed of >135 bioinformatics packages, along with documentation, desktop interface and graphical software applications (Krampis *et al.*, 2012). This resource is publicly available as a web service, hosted on the Amazon EC2 cloud, providing high-performance infrastructure for bioinformatics computing that can be configured according to user requirements. This tool presents a useful and scalable solution that allows researchers to specify the required environment for their data, in addition to the time required for analysis. Although designed to run on compute farms, such as EC2, this is not a requirement of the system and local private versions can also be installed. Although this VM contains an extensive array of bioinformatic tools, it is not specifically tailored to NGS analysis, but rather represents a sort of comprehensive bioinformatics tool box. In contrast, other published VMs such as CloVR (Angiuoli *et al.*, 2011) use similar methodologies to Cloud BioLinux, but in this case the VM has been specialized to deal with a specific type of NGS analysis. Using either a cloud-hosted VM or a local instantiation, the CloVR VM uses a variety of software targeted toward metagenomics, microbial genomics and small genome phylogenetics. The CloVR VM features a dashboard function with color-coded pre-configured analysis pipelines that allow users to alter any parameters for the analysis though graphical user interface widgets (e.g. drop-down menus and radio buttons). Aside from downloading the required free VM software (e.g. VirtualBox or VMware), users only need to download the appropriate VM image file (.vdi or .vmdk, respectively) to get started. Another targeted NGS–VM program suite for distributed *de novo* genome construction and motif finding has also recently been developed (Corwin *et al.*), and Corwin and colleagues also demonstrate the applicability of their VM in running either locally or on a cluster. The development of such NGS–VMs is highly encouraging; however, at the moment they remain highly targeted for specific analysis types. Additionally, although there are more examples of software developed and distributed as VMs, such as XperimentR (Tomlinson *et al.*, 2013) and NGS–SNP (Grant *et al.*, 2011), a simple and comprehensive NGS–VM pipeline, complete with scripts to automate underlying tools that could address all of the issues we have raised, has, to our knowledge, yet to be published.

3 VM PERFORMANCE

Although the use of a comprehensive NGS–VM would remove a number of critical bottlenecks in NGS analysis, VMs have historically suffered from the inherent inefficiency caused by the requirement of trapping and emulating privileged instructions within the host OS. This step is essential to allow commands from the guest OS in the VM to be converted and executed on the underlying hardware. Although significant improvements have been made in hardware support for virtualization in the past few years, any VM will still be limited by the underlying memory, central processing unit (CPU), input/output (I/O) and disk space requirements, all of which must also be shared with the host OS. Despite these constraints, the continuing trend for rapidly decreasing costs for memory and disk space, along with the modest requirements for most VM software, means that this does not represent a real limitation. Regardless, the systematic evaluation of the extent to which current VMs perform relative to native hardware has not been well explored, and so we examine this issue.

3.1 Performance of VMs as a barrier to practical use

To directly assess the impact of some of these constraints in a setting relevant for NGS analysis, we performed a benchmarking comparison using five different configurations on identical hardware (Table 4). We first examined VM performance through mapping and CPU intensive analysis of 8 million 100 bp RNA sequence reads using TopHat (Trapnell *et al.*, 2009), Casava and SAMtools (Durbin, 2009) software. All software was run on either on a native Linux OS or a guest Linux OS running on either a Linux or Windows host OS using VMware or VirtualBox. As expected, most VM configurations suffered a performance penalty of ∼25% on average, relative to the same task run on the native Linux configuration. Although such a modest decrease in performance would likely be acceptable in most academic settings, one of the configurations (VirtualBox 4.2.0 running Scientific Linux 6.3 on Windows 7 Enterprise

Table 4. Benchmarking of virtual systems[a]

	Linux native	VMWare/Linux	VirtualBox/Linux	VMWare/Windows	VirtualBox/Windows	Average	Runtime increase (%)	Amazon EC2 m1.xlarge	Runtime increase (%)
TopHat mapping	175	229	229	208	185	212.8	22	336	92
(2 threads)	266	313	355	282	257	301.8	13	428	61
Casava alignment	71	93	106	88	79	91.5	29	129	82
(2 threads)	142	175	205	164	150	173.5	22	258	82
Casava build (sort,bam)	18	23	28	21	20	23.0	28	29	61
samtools mpileup chr1	7.22	9.97	9.97	8.75	6.75	8.9	23	13.33	85

Synthetic I/O tests, identical hardware[b]

	Linux native	VMWare/Linux	VirtualBox/Linux	VMWare/Windows	VirtualBox/Windows	Average	Standard deviation	Speed decrease (%)
Bonnie++	108 669	98 791	87 404	78 785	89 513	88 623.3	8214.9	−18.4
write buff	204 891	129 239	146 835	190 057	200 329	166 615.0	34 031.2	−18.7
Readc	94 627	49 018	65 586	84 936	76 994	69 133.5	15 585.5	−26.9
read buff.	229 532	93 002	115 556	236 038	166 686	152 820.5	63 467.3	−33.4
seek/s	548	645.4	327	354	442	442.1	144.2	−19.3

Note: Hardware: 6 core Intel Xeon X5675@3.07GHz,12 MB cache,24 GB RAM ECC DDR3 1333 MHz,2x500 GB HD (6Gbps NL SATA), RAID 0. Software: Scientific Linux 6.3, Windows 7 Enterprise Edition, VMware player 5.0.0, VirtualBox 4.2.0. Amazon EC2 m1.xlarge instance: 4 cores Intel Xeon E5-2650@2.00 GHz, 15 GB; $0.48/hr.
[a]runtime results in minutes.
[b](write/read, in KB/s).

Edition) was in fact remarkably efficient. Although this particular configuration averaged only an ~7% decrease in speed compared with the native Linux configuration, in two of the benchmark tests (variant calling and multithreaded read mapping) it was actually faster than the native configuration. This VM/OS combination is of particular interest, given that many end users work in a Windows environment, whereas most NGS software is developed to run specifically on Linux machines.

Given the large file sizes associated with NGS data, we also wanted to test the hard drive and file system performance of these configurations using the Bonnie++ utility, a file system benchmarking tool that measures specific parameters such as file creation, write, read and removal times (Coker, 2001). As with the CPU tests, we found that the performance penalty of a VM was relatively modest, on average (~23%); however, there were again specific tests where VM configurations remarkably outperformed the native Linux configuration. Although our VM benchmarking was not exhaustive (especially given the number of VMs and bioinformatic tools available), the results nevertheless suggest that, perhaps contrary to common belief, VMs do not necessarily have significantly poorer performance compared with a native OS. In agreement with our results, other benchmarking tests have also concluded the computational 'cost' of VMs is relatively low compared with native execution on hardware (Corwin *et al.*). In addition, because VMs physically exist as several files on the hard drive of the host machine, the entire system, and all the data it contains, can simply be transferred to newer hardware on a regular basis to further improve performance when required. Despite the caveat that the final performance will ultimately be highly dependent on the hardware setup, software tool configuration and the specific VM used, our results indicate that a suitable NGS–VM could be configured in such a way that reduces its impact on performance.

Given the interest in cloud computing, we also performed benchmarking of our VM on Amazon EC2 in an m1.xlarge instance. Although the results indicate a longer runtime on EC2, it is important to note that differences in the hardware used (e.g. processor differences, RAM available; Table 4), and the potentially way in which instances are managed on EC2, likely influenced performance results. More importantly, the Amazon cloud allows job scaling over orders of magnitude, such that the absolute time required for an analysis could be arbitrarily reduced by simply initiating additional instances and dividing the data among the instances. Although we cannot directly compare our local versus cloud-based results, the ease with which we could transfer and run our VM highlights the potential of these remote resources to enable small laboratories with no computation infrastructure to download a VM, populate it with their data, and then transfer this to a cloud compute center for rapid and on-demand analysis.

3.2 Potential for NGS–VM growth in biomedical field

The concept of bundling software within a VM is not novel; indeed one of the largest VM developers, VMware, currently markets this configuration of tools within a VM as a 'Virtual Appliance'. The possibility for NGS–VM usage to be broadly adopted by the research community will be dependent on a significant change in philosophy, although there is already a growing appreciation that there are problems with the current methods of software use and distribution (Nekrutenko and Taylor, 2012). Beyond the research environment, a more compelling push for the development of NGS–VMs will likely come from the growing application of NGS to clinical diagnostics. According to the American Hospital Association, there were almost 37 million hospital admissions in the USA alone in 2012 (http://www.aha.org/research/rc/stat-studies/fast-facts.shtml). If only a fraction of these admissions would require diagnostic sequencing (e.g. metagenomic analysis of bacterial infections prior to antibiotic treatment), the informatics requirements at thousands of hospitals would be enormous. Therefore, as great as the current difficulties are in research centers (where some bioinformatic resources are often available), the problem for hospitals and medical clinics may soon be orders of magnitude worse. The possibility of developing a standardized, and clinically certified, NGS-VM could dramatically reduce the complexity of integrating NGS technologies into diagnostic settings, allowing its broader adoption and use. With respect to clinical certification for VMs (e.g. Clinical Laboratory Improvement Amendments /Health Insurance Portability and Accountability Act compliance), issues around information security, audit trails, access controls, etc. could conceivably be addressed within the initial design of the VM itself, similar to any other hospital information system. For example, if one hospital (in conjunction with a laboratory or research center) developed an NGS–VM that could be clinically validated for a specific purpose (e.g. identifying mutations in a panel of cancer genes), this same NGS–VM should be able to be downloaded and used with little or no modification by another hospital for the same purpose. Because there are a finite number of diagnostic procedures likely to be adapted to NGS technology, a repository of such clinical NGS–VMs could be created, along with any details of their implementation. This approach would not only remove the incredible redundancy of every hospital and clinic independently downloading and configuring multiple tools for analysis, but it would also facilitate meta-analysis of diagnostic results between centers.

4 CONCLUSIONS

The advent of extremely rapid, inexpensive high-throughput DNA sequencing will likely be considered to be one of the most significant developments in the history of biomedical research. At the same time, the power of this approach has become limited by the very factors that made it revolutionary to begin with, namely, the ability to rapidly and cheaply produce massive amounts of sequence data. Although new bioinformatic tools have successfully dealt with specific challenges associated with analyzing NGS data, the creation of a large and disparate collection of tools has made it difficult to ensure reproducibility of computational analysis and is hindering the adoption of NGS technology. VMs present a simple and realistic option for addressing both of these concerns, and their potential has, until now, been only partially exploited. Despite the requirement for a significant shift in the paradigm for distribution of bioinformatic tools, the future alternative of an even great collection of individual OS-specific NGS tools will carry far fewer benefits for end users of this technology. By exploiting the object-oriented model of distributing data and methods as a single entity, VMs cannot only simplify NGS analysis, but will also facilitate the growth and integration of this technology into additional fields.

ACKNOWLEDGEMENTS

We would like to thank Samuel Marguerat and colleagues within IRIC for critical review of the manuscript and helpful comments.

Funding: Cole Foundation, Le Fonds de recherché du Québec – Santé grant number #25341 (to B.T.W), Genome Quebec (to B.T.W. and S.L.) and Institut National de la Recherche Agronomique (to M.C.).

Conflict of Interest: none declared.

REFERENCES

Abyzov,A. *et al.* (2011) CNVnator: an approach to discover, genotype, and characterize typical and atypical CNVs from family and population genome sequencing. *Genome Res.*, **21**, 974–984.

Altschul,S.F. *et al.* (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Anders,S. and Huber,W. (2010) Differential expression analysis for sequence count data. *Genome Biol.*, **11**, R106.

Angiuoli,S.V. *et al.* (2011) CloVR: a virtual machine for automated and portable sequence analysis from the desktop using cloud computing. *BMC Bioinformatics*, **12**, 356.

Butler,J. *et al.* (2008) ALLPATHS: de novo assembly of whole-genome shotgun microreads. *Genome Res.*, **18**, 810–820.

Chiang,D.Y. *et al.* (2008) High-resolution mapping of copy-number alterations with massively parallel sequencing. *Nat. Methods*, **6**, 99–103.

Coker,R. (2001) *Bonnie++*. http://www. coker. com. au/bonnie+, (03 July 2013, date last accessed) 53.

Corwin,J. *et al.* A virtual machine program-suite for distributed de novo genome construction and motif finding. http://www.blueideas.de/ecs234_s10_cloudcomputing.pdf (03 July 2013, date last accessed).

Creasy,R.J. (1981) The origin of the VM/370 time-sharing system. *IBM J. Res. Devel.*, **25**, 483–490.

Cret,O. *et al.* (2009) A hardware algorithm for the exact subsequence matching problem in DNA strings. *Rom. J. Inf. Sci. Technol.*, **12**, 51–67.

Da Wei Huang,B.T.S. and Lempicki,R.A. (2008) Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. *Nat. Protoc.*, **4**, 44–57.

Dobin,A. *et al.* (2012) STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, **29**, 15–21.

Durbin,R. (2009) The sequence alignment/map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.

Fejes,A.P. *et al.* (2008) FindPeaks 3.1: a tool for identifying areas of enrichment from massively parallel short-read sequencing technology. *Bioinformatics*, **24**, 1729–1730.

Goecks,J. *et al.* (2010) Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol.*, **11**, R86.

Goya,R. *et al.* (2010) SNVMix: predicting single nucleotide variants from next-generation sequencing of tumors. *Bioinformatics*, **26**, 730–736.

Grant,J.R. *et al.* (2011) In-depth annotation of SNPs arising from resequencing projects using NGS-SNP. *Bioinformatics*, **27**, 2300–2301.

Hardcastle,T.J. and Kelly,K.A. (2010) baySeq: empirical Bayesian methods for identifying differential expression in sequence count data. *BMC Bioinformatics*, **11**, 422.

Homer,N. *et al.* (2009) BFAST: an alignment tool for large scale genome resequencing. *PLoS One*, **4**, e7767.

Karolchik,D. *et al.* (2009) The UCSC genome browser. *Curr. Protoc. Bioinformatics*, 1.4. 1-1.4. 26.

Kent,W.J. (2002) BLAT—the BLAST-like alignment tool. *Genome Res.*, **12**, 656–664.

Koboldt,D.C. *et al.* (2009) VarScan: variant detection in massively parallel sequencing of individual and pooled samples. *Bioinformatics*, **25**, 2283–2285.

Krampis,K. *et al.* (2012) Cloud BioLinux: pre-configured and on-demand bioinformatics computing for the genomics community. *BMC Bioinformatics*, **13**, 42.

Laczik,M. *et al.* (2012) Geno viewer, a SAM/BAM viewer tool. *Bioinformation*, **8**, 107.

Langille,M.G. and Eisen,J.A. (2010) BioTorrents: a file sharing service for scientific data. *PLoS One*, **5**, e10071.

Langmead,B. *et al.* (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.*, **10**, R25.

Larson,D.E. *et al.* (2012) SomaticSniper: identification of somatic point mutations in whole genome sequencing data. *Bioinformatics*, **28**, 311–317.

Li,H. (2011) A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics*, **27**, 2987–2993.

Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, **25**, 1754–1760.

Li,H., Ruan,J. and Durbin,R. (2008) Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Research*, **18**, 1851–1858.

Li,H. *et al.* (2008a) Maq: mapping and assembly with qualities, Version 0.6, 3. *Genome research*, **18**, 1851–1858.

Li,H. *et al.* (2009) The sequence alignment/map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.

Li,R. *et al.* (2008b) SOAP: short oligonucleotide alignment program. *Bioinformatics*, **24**, 713–714.

Li,R. *et al.* (2009) SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics*, **25**, 1966–1967.

Mardis,E.R. (2011) A decade/'s perspective on DNA sequencing technology. *Nature*, **470**, 198–203.

McKenna,A. *et al.* (2010) The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res.*, **20**, 1297–1303.

Nekrutenko,A. and Taylor,J. (2012) Next-generation sequencing data interpretation: enhancing reproducibility and accessibility. *Nat. Rev. Genet.*, **13**, 667–672.

Ng,S.B. *et al.* (2010) Massively parallel sequencing and rare disease. *Hum. Mol. Genet.*, **19**, R119–R124.

Ning,Z. *et al.* (2001) SSAHA: a fast search method for large DNA databases. *Genome Res.*, **11**, 1725–1729.

Pfeifer,G.P. and Hainaut,P. (2011) Next-generation sequencing: emerging lessons on the origins of human cancer. *Curr. Opin. Oncol.*, **23**, 62.

Popendorf,K. and Sakakibara,Y. (2012) SAMSCOPE: an OpenGL-based real-time interactive scale-free SAM viewer. *Bioinformatics*, **28**, 1276–1277.

Robertson,G. *et al.* (2010) De novo assembly and analysis of RNA-seq data. *Nat. Methods*, **7**, 909–912.

Robinson,J.T. *et al.* (2011) Integrative genomics viewer. *Nat. Biotechnol.*, **29**, 24–26.

Robinson,M.D. *et al.* (2010) edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, **26**, 139–140.

Rozowsky,J. *et al.* (2009) PeakSeq enables systematic scoring of ChIP-seq experiments relative to controls. *Nat. Biotechnol.*, **27**, 66–75.

Rumble,S.M. *et al.* (2009) SHRiMP: accurate mapping of short color-space reads. *PLoS Comput. Biol.*, **5**, e1000386.

Rutherford,K. *et al.* (2000) Artemis: sequence visualization and annotation. *Bioinformatics*, **16**, 944–945.

Salomonis,N. *et al.* (2009) Alternative splicing in the differentiation of human embryonic stem cells into cardiac precursors. *PLoS Comput. Biol.*, **5**, e1000553.

Sathirapongsasuti,J.F. *et al.* (2011) Exome sequencing-based copy-number variation and loss of heterozygosity detection: ExomeCNV. *Bioinformatics*, **27**, 2648–2654.

Schweiger,M.R. *et al.* (2011) The power of NGS technologies to delineate the genome organization in cancer: from mutations to structural variations and epigenetic alterations. *Cancer Metastasis Rev.*, **30**, 199–210.

Shen,J.J. and Zhang,N.R. (2012) Change-point model on nonhomogeneous Poisson processes with application in copy number profiling by next-generation DNA sequencing. *Ann. Appl. Stat.*, **6**, 476–496.

Simpson,J.T. *et al.* (2009) ABySS: a parallel assembler for short read sequence data. *Genome Res.*, **19**, 1117–1123.

Smith,J.E. and Nair,R. (2005) The architecture of virtual machines. *Computer*, **38**, 32–38.

Stein,L.D. (2010) The case for cloud computing in genome informatics. *Genome Biol.*, **11**, 207.

Tarazona,S. *et al.* (2011) Differential expression in RNA-seq: a matter of depth. *Genome Res.*, **21**, 2213–2223.

Teer,J.K. *et al.* (2012) VarSifter: visualizing and analyzing exome-scale sequence variation data on a desktop computer. *Bioinformatics*, **28**, 599–600.

Tomlinson,C.D. *et al.* (2013) XperimentR: painless annotation of a biological experiment for the laboratory scientist. *BMC Bioinformatics*, **14**, 8.

Trapnell,C. *et al.* (2009) TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics*, **25**, 1105–1111.

Trapnell,C. *et al.* (2012) Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. *Nat. Protoc.*, **7**, 562–578.

Wang,L. *et al.* (2010) DEGseq: an R package for identifying differentially expressed genes from RNA-seq data. *Bioinformatics*, **26**, 136–138.

Warren,R.L. *et al.* (2007) Assembling millions of short DNA sequences using SSAKE. *Bioinformatics*, **23**, 500–501.

Xie,C. and Tammi,M.T. (2009) CNV-seq, a new method to detect copy number variation using high-throughput sequencing. *BMC Bioinformatics*, **10**, 80.

Zerbino,D.R. and Birney,E. (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res.*, **18**, 821–829.

Zhang,Y. *et al.* (2008) Model-based analysis of ChIP-Seq (MACS). *Genome Biol.*, **9**, R137.