Safe “cloudification” of large images through picker APIs

Erich Bremer[1], Tahsin Kurc[1], Yi Gao[1], Joel Saltz[1], Jonas S Almeida[1]*

1) Dept Biomedical Informatics, Stony Brook University (SUNY), NY 11794

* *jonas.almeida@stonybrookmedicine.edu*

Abstract

The “Box model” allows users with no particular training in informatics, or access to specialized infrastructure, operate generic cloud computing resources through a temporary URI dereferencing mechanism known as “drop-file-picker API” (“picker API” for sort). This application programming interface (API) was popularized in the web app development community by DropBox, and is now a consumer-facing feature of all major cloud computing platforms such as Box.com, Google Drive and Amazon S3. This reports describes a prototype web service application that uses picker APIs to expose a new, “cloudified”, API tailored for image analysis, without compromising the private governance of the data exposed. In order to better understand this cross-platform cloud computing landscape, we first measured the time for both transfer and traversing of large image files generated by whole slide imaging (WSI) in Digital Pathology. The verification that there is extensive interconnectivity between cloud resources let to the development of a prototype software application that exposes an image-traversing REST API to image files stored in any of the consumer-facing “boxes”. In summary, an image file can be upload/synchronized into a any cloud resource with a file picker API and the prototype service described here will expose an HTTP REST API that remains within the safety of the user’s own governance. The open source prototype is publicly available at sbu-bmi.github.io/imagebox.

Availability

The accompanying prototype application is made publicly available, fully functional, with open source, at http://sbu-bmi.github.io/imagebox. An illustrative webcasted use of this Web App is included with the project codebase at https://github.com/SBU-BMI/imagebox.

Introduction

The advantages of user-facing cloud computing are particularly hard to realise for specialized image analysis applications because of the dependence on specialized libraries and infrastructure, as is the case for the popular Open Slide platform [Goode 2013]. The challenge of presenting image analysis solutions to those that do not have those specialized resources is significantly compounded if they target large images in private stores. In other words, image informatics applications are currently dependent on specialized informatics infrastructure operated by specialized human resources. This is of course not to the demerit of the specialized libraries themselves, such as the excellent OpenSlide, it instead reflects a missing component in the cloud computing landscape. Two of the most compelling scenarios where these challenges arise are a) when one or more regions in a whole slide image needs to be object of a specialized analysis; and b) when a private image needs to presented to a domain user that does not have access to web-facing image analysis infrastructure (which is the norm rather than the exception).

The immediate motivation for this work is to address requirements associated with the set of intense research efforts that focus on linking morphologic and molecular cancer characterizations and to make use of combined imaging and molecular signatures to substratify patient populations to predict outcome and response to treatment. A number of reports by the National Cancer Institute describe efforts being carried out by the pathology and radiology research communities to link imaging phenotypes with large-scale genomic analyses, ultimately informing personalized medicine clinical trial designs such as NCI-Match (https://clinicaltrials.gov/ct2/show/NCT02465060). Characterization of tissue morphology related phenotypes is a complex effort that requires the development and deployment of a variety of pipelines, many of which require flexible access to image sections. Accordingly, the cloud API mediator described here was developed in the context of Pathology images. However, this “cloudification” approach is equally applicable to other image analysis infrastructure scenarios such as those involving Radiology.

These challenges of commoditized and portable biomedical informatics analysis are of course not a problem exclusive of image analysis applications. Novel REST API architectures are indeed emerging to address them as a generic solution. Specifically, cloud computing infrastructure has responded with new interoperability models [ONC 2015], as illustrated by the central role of the HL7 FHIR API (https://www.hl7.org/fhir), which create novel opportunities for architecturing distributed whole slide image processing. These opportunities for RESTful interoperation are assessed in this report through the use of the accompanying webApp application prototype, which places no demands on the user beyond the client Web Browser. In other words, the functionality of the “cloudified” image traversal API it validated by a prototype Web Application. This application will seek to engage the “Box / file picker” application programming interface (API) exposed by a Box cloud resource, which have emerged to interoperate with infrastructure of a wide variety of cloud providers. Picker APIs, after successful authentication (typically using OAuth 2.0), expose temporarily dereferenceable URLs which can be picked up by cloud-based applications in order to make full use of the high connectivity that characterizes hosting data centers.

The development of high performance “slice servers” (which extract slices of different regions, at different scales, from large image files) has also advanced significantly, as illustrated by the open source IIP project (iipimage.sourceforge.net). Furthermore, the ability to approach image analysis either as a web service or as computation entirely on the client side has also been object of considerable attention. The former is typically a commercial application, such as Aperio’s ePathology product [Leica 2015], and the latter is typically an Academic exploit, such as our own [Almeida 2012]. As a result of these advances in portable image presentation, the obstacles to a consumer-facing approach to interact with large images, wherever they may be, is now reduced to the enablement of an easily distributable programmatic interoperability mechanism.

These developments place the picker API at an interesting intersection between cloud computing and high performance slice servers. As regards Biomedical applications, this intersection has, however, been associated with lack of sufficient security and privacy to allow users such as Pathologists and translational researchers. This final obstacle to exploring cloud (HTTP REST) API implementation architecture was also removed in recent times by the adoption of HIPAA compliant practices by a number of cloud providers, such as Box.com [Box 2015], Google Drive [Google 2015], and Microsoft OneDrive [Microsoft 2015].

Methods

ImageBox WebApp

The imageBox Web App was developed entirely as a within browser application, using only the Web's "assembly language", JavaScript. This approach, which relies exclusively on the Web Platform (webplatform.org) and W3C open standards, leads to applications that are assembled directly in the web browser of the user, with no requirement for software download and installation, concern about native library dependencies or exposure of sensitive data to third parties. As a consequence, the imageBox Web App was developed in the public domain with open source and version control from its onset (https://github.com/SBU-BMI/imagebox). The fully functional application itself is served directly from GitHub by maintaining the application code in the "gh-pages" branch (i.e. live Web App served at sbu-bmi.github.io/imagebox). This signifies that the imageBox Web App is being delivered with versioned hosting, such that sites like https://rawgit.com can expose individual versions programatically.

TCGA images

The operation of ImageBox was tested for whole slide images of The Cancer Genome Atlas. These images were both called directly from TCGA's public web directory, or where downloaded to private Box.com and DropBox folders. The latter served the purpose of validating the desired alignment of imageBox with the governance of private whole slide image files without compromising the content of the hosting folder (see File-picker API below and also Discussion). At the time of writting this report, the public folder with tens of thousands of TCGA images for all 30+ tumor types can be found at ***https://tcga-data.nci.nih.gov/tcgafiles/ftp_auth/distro_ftpusers/anonymous/tumor/<tumor-type>/bcr/nationwidechildrens.org/tissue_images/slide_images/***, using the codes for **<tumor-type>** listed at https://tcga-data.nci.nih.gov/tcga. For example, for Glioblastoma Multiforme, the TCGA slide image folder would be at https://tcga-data.nci.nih.gov/tcgafiles/ftp_auth/distro_ftpusers/anonymous/tumor/gbm/bcr/nationwidechildrens.org/tissue_images/slide_images. However, NCI has announced the intention of moving access to public TCGA data to the new Data Commons resource at https://gdc-portal.nci.nih.gov, so in the future the URL composition may need to be redirected to that new resource.

File-picker API

The implementation of the file-picker API mechanism varies between “Box providers” (cloud resources that expose file-picker APIs, see Introduction) even if they deliver the same final result: a uniquely dereferenceable URL. The two Boxes tested by imageBox are Box.com and DropBox. The details of their file-picker APIs can be found, respectively, at https://developers.box.com/the-box-file-picker and https://www.dropbox.com/developers/dropins/chooser/js.

Cloud-hosted imageBox slicer

This component, represented as a cloud box and labeled “imageBox” in Figure 1, uses the Jetty server (http://www.eclipse.org/jetty) to handle whole-slide microscopy images upwards of 100,000 pixels by 100,000 pixels. To enable the imageBox tiling server to access the SVS image files, the Bioformats toolbox (http://www.openmicroscopy.org/site/products/bio-formats) was used, which was developed to read and write image data between different formats with a focus on digital pathology. As detailed in the project’s documentation page, this toolbox relies on the OME-Model (http://www.openmicroscopy.org/site/support/ome-model) to slice and scale image piramids, and is then exposed by ImageBox API. To retrieve the OME annotation of a imageBox link, add “&format=json” and the end of a slice URL, which will otherwise return a pixel map.

Results

A prototype application (see Availability) was developed to engage a image tiling server (“slice server”, see Methods) as part of a file-picking API call to private cloud storage. This application was used to measure transfer and retrieval rates for both moving the full SVS file to a well connected (cloud) slice server, and then retrieving image sub-regions at different scales. As schematized in Figure 1, this cloud-based “imageBox” resource is equipped with a basic HTTP REST API to support its operation from a variety of applications and external services. The operation of the accompanying application (Figure 2) are used to build the diagram in Figure 1 and to obtain the values in Table 1. Since the whole slide SVS image is de-referenced from a URL, the full file can come from either a public image repository such as TCGA (The Cancer Genome Atlas) or from a private Box. The protoype application uses a TCGA hosted image as a default use-case, but the user is encouraged to point it to private accounts in Box.com and DropBox (Figure 2, see also videcast link in the project github page), as well as other SVS files in the public domain such as TCGA's. As noted in the application user interface (see link under Box buttons), the approach can be extended to many more Box providers using their API directly, and through commercial proxy services such as filepicker.com.

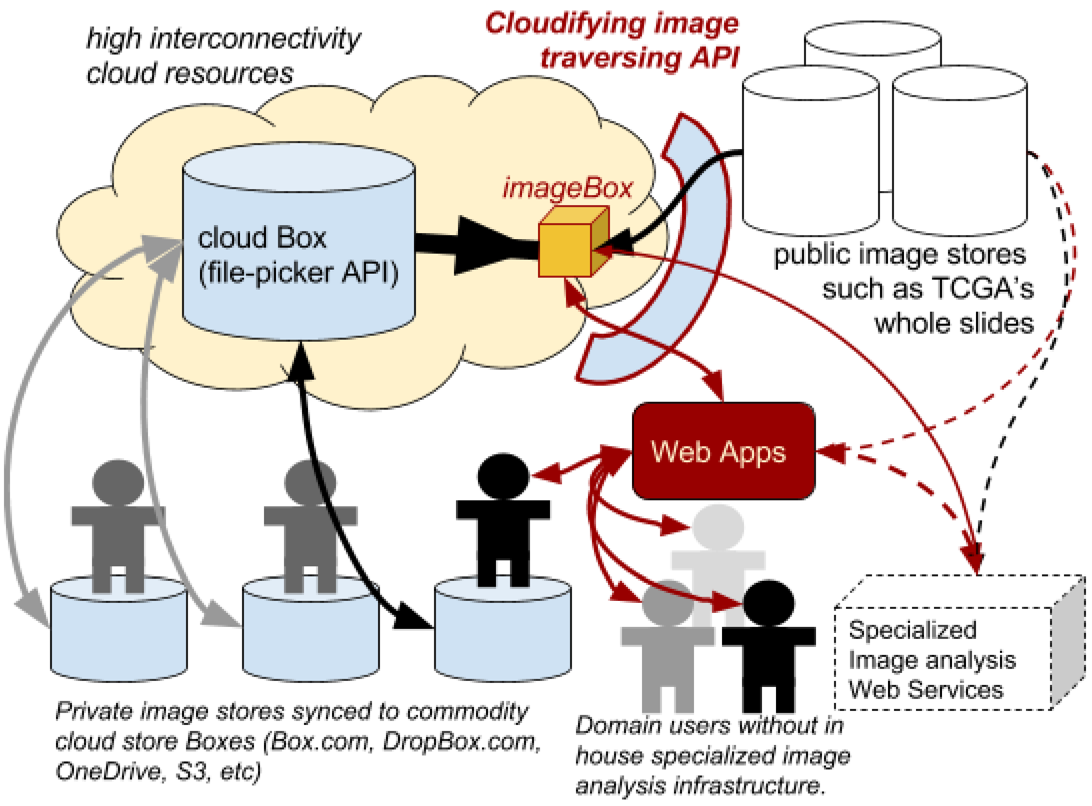
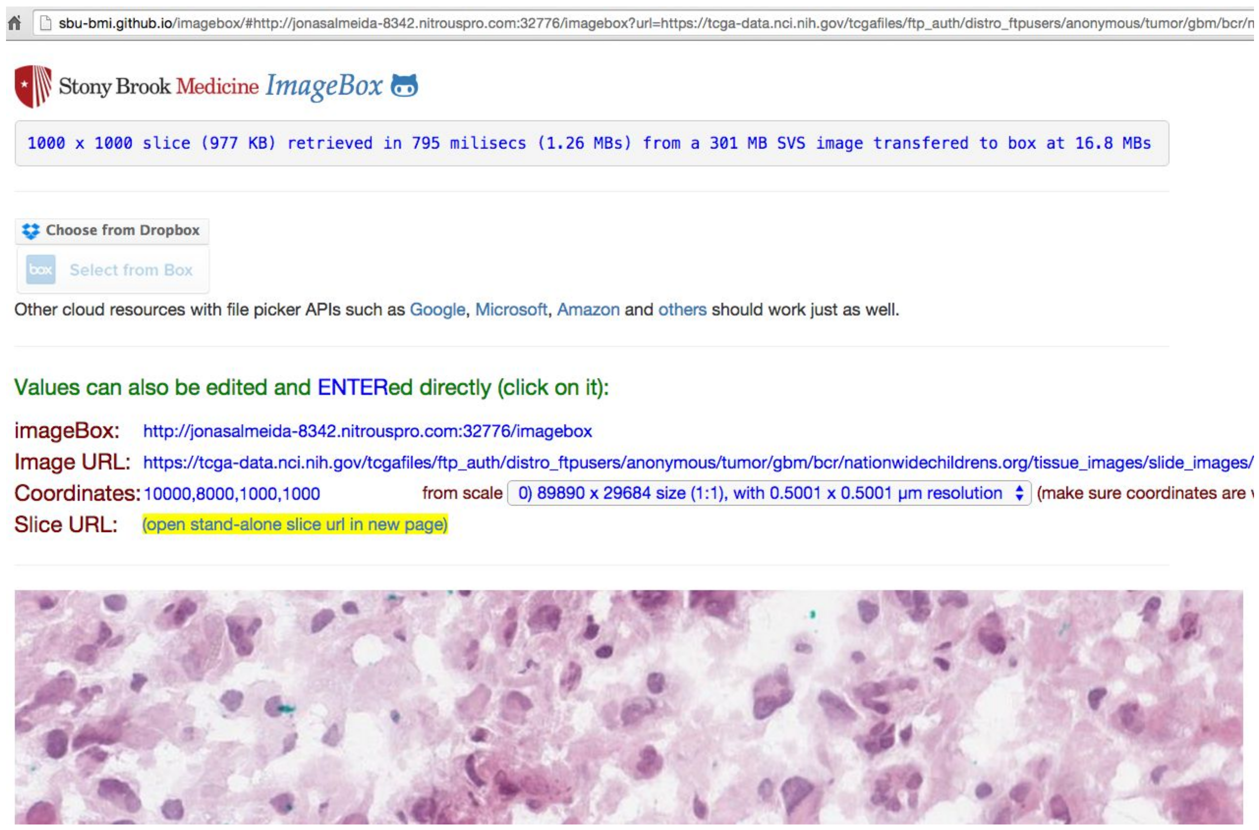

high interconnectivity cloud resources
Cloudifying image traversing API
cloud Box (file-picker API)
imageBox
public image stores such as TCGA's whole slides
Web Apps
Specialized Image analysis Web Services
Private image stores synced to commodity cloud store Boxes (Box.com, DropBox.com, OneDrive, S3, etc)
Domain users without in house specialized image analysis infrastructure.

Figure 1 - ImageBox implementation architecture. At the most basic level, by exposing an image traversing API (in red), the cloud hosted imageBox server acts as a cache for an IIP slice server. That is the case when the external whole slide SVS file is hosted in a public store such as TCGA's (see Methods). The engagement of regular file-picker APIs of cloud stores such as Box.com and DropBox is the most novel aspect of this achitecture, because it allows the serving of image slices without breaching the access constraints of the original image. A user can of course only push to imageBox the files she has access, and in response will receive the unique dereferenceable identifier of the private images being sliced. Finally, the slice URLs (see Figure 2) can be used by Web Apps and other client applications: see project github page for examples of direct use of imageBox from Matlab. This implementation architecture can therefore be straightforwardly extended (dashed lines) to image analysis services.



sbu-bmi.github.io/imagebox/#http://jonasalmeida-8342.nitrouspro.com:32776/imagebox?url=https://tcga-data.nci.nih.gov/tcgafiles/ftp_auth/distro_ftpusers/anonymous/tumor/gbm/bcr/n

```
1000 x 1000 slice (977 KB) retrieved in 795 milisecs (1.26 MBs) from a 301 MB SVS image transfered to box at 16.8 MBs
```

Other cloud resources with file picker APIs such as Google, Microsoft, Amazon and others should work just as well.

Values can also be edited and ENTERed directly (click on it):

imageBox: http://jonasalmeida-8342.nitrouspro.com:32776/imagebox
Image URL: https://tcga-data.nci.nih.gov/tcgafiles/ftp_auth/distro_ftpusers/anonymous/tumor/gbm/bcr/nationwidechildrens.org/tissue_images/slide_images/
Coordinates: 10000,8000,1000,1000 from scale 0) 89890 x 29684 size (1:1), with 0.5001 x 0.5001 µm resolution (make sure coordinates are
Slice URL: (open stand-alone slice url in new page)

Figure 2 - Snapshot of accompanying imageBox Web Application at http://sbu-bmi.github.io/imagebox, used to demonstrate interoperability features on the proposed slice distribution implementation architecture (Figure 1). A videocast of its usage is available in the project home page at https://github.com/SBU-BMI/imagebox.

The imageBox application prototype was pointed to a variety of whole slide image SVS files in Box.com and DropBox cloud folders, as well as in public folders such as The Cancer Genome Atlas (TCGA, see Methods for folder address). The imageBox slice server (See Methods) was in turn hosted in Amazon AWS. Although transfer rates varied, we found that between the within-cloud resource components the values are consistently above 10 MBs, and are routinely twice that much (Table 1). The slice dowload rates, even when the client machine was in a network equipped with generous bandwidth, did not typically exceed 1.5 MBs. The more important number, however, is the availability of individual image slices resolving significant morphological features in under a second, and while placing no storage requirements on the client application. The support for slice URL's (Fig.2, highlighted in yellow: click on "open in new page") has two important attributes. The first one is that they are portable, readily dereferenceable, and can be analyzed and annotated by any client application without requiring the whole image. The second critical advantage is that they do not expose the original hosting source, as can be verified by noting that the same image is rendered with different URLs when retried from a private Box folder by two different users, or even by the same user in two different sessions. These values and behaviours can be verified by using the accompanying imageBox prototype in Fig 2, or by visualizing the webcast on the project github page.

Table 1 - Example transfer rates from DropBox to the imageBox in Fig 2 (http://Nitrous.IO). As is often the case, both resources use Amazon's AWS as their backend, which implies that benchmarking was in fact performed on 10Gbs connected AWS EC2 servers for both EBS-backed system volumes, as well as, SSD-backed system volumes. The reader is encouraged to generate their own measurements by using the accompanying imageBox application (see Availability).

SVS file size (MB)	Time to transfer (seconds)	Data Transfer Rate (MB/sec)
1,064.96	38	28
449.4	16	28.1
171.06	5.3	32.3
175.49	7.1	24.7
116.3	3.4	34.2
75.15	2.3	32.7
2,052.36	72.1	28.5

As discussed in the next section, the scalability of the imageBox architecture benefits from the 10-fold faster within-cloud transfer rates. However, the scalability of the proposed implementation architecture does not: the faster rates only have an effect on the availability of the first slice for a given whole slide image. The slower "first slice" reflects its use as the event that triggers transfer of the full image to the in-cloud imageBox IIP server (Fig1, "imageBox"). The retrieval of subsequent slices benefits from its ready availability in the imageBox slide server (Fig1 "ImageBox") for retrieval by a client application (Fig 1, "WebApps" and "Web Services"). It is really in these subsequent traversals of the pre-existing image that the promise of "cloudified" scalable image analysis is realised. Case in point, pathologically relevant image features, at any scale, should be resolvable by $1,000^2$ and surely by $2,000^2$ image slices. It is worth noting that the image analysis operations on such slices typically exceed the 1 and 4 secs, respectively, that it takes to retrieve them. In summary, with the imageBox implementation architecture, the slice retrieval ceases to be the limiting step, which then moves to the availability of distributed computational resources. Just as important, the nature and governance of those resources is now much broader than in the conventional architecture, for example when using OpenSlide, as will be discussed below.

Discussion

Given its well deserved popularity and wide use of OpenSlide [Goode 2013], this discussion of the imageBox implementation architecture has to start there. OpenSlide's ensemble of an efficient C-based image server and a client side dynamic zoom library established and popularised the core features of availability and scalability of a slice server architecture in general. In that regard, the work described here lays on top of those achievements rather than representing an alternative. The innovative element advanced by imageBox is the programmatic intermediation for distributed image analysis instead of visualization. Accordingly, the emphasis was placed on a) reducing the functionality of the slice server down to the bare minimum of responding to the selection of a scale and a pixel slice, b) wrapping the slice server as a cloud hosted HTTP REST service such that c) it could be programmatically operated not only for public image's (such as TCGA's), but also for images privately hosted in stores equipped with file-picker APIs ("Boxes", see Introduction).

The novel operational features of the imageBox deployment architecture are particularly clear when considering a data-driven, consumer-facing approach. As the accompanying prototype demonstrates, the user of this Web Application does not have to download or maintain any server side resource. Correspondingly, the App developer does not have access, or maintenance responsibilities, to the slice server or the storage where the data is primarily hosted. In a nutshell, the imageBox implementation architecture was designed for full decoupling between application development and application usage. Although, along the lines of QMachine [Wilkinson 2014], one could distribute in-browser image analysis using approaches such as imageJS [Almeida 2012], a more immediate choice would be to expose established image analysis applications, such as 3D Slicer [Fedorov 2012], as web services. That approach, however, passes image sections to a remote location, which may raise privacy concerns or have logistical dependencies the domain user and third party client applications cannot control.

For the reasons discussed above, the prototype deployment accompanying this report stops with the demonstration that a web application (client) is able to retrieve image slices from whole slide images hosted in a private store (making use of the file-picker/Box API) without placing any requests that the user deploys or logs into a server side resource. Furthermore, the inspection of the slice URL composition will reveal a basic HTTP REST API which can be engaged from computational environments well suited for image analysis, not only 3D SLicer [Fedorov 2012] as noted above, but also general purpose image analysis environments such as Matlab (see project github repository for examples). Given the successful delivery of a functional prototype for SVS whole slide images, we would argue that one or more cloud based imageBox slice servers would be a valuable addition as public resource for researchers developing or consuming image analysis applications. We envision that such a resource would be further developed to support a wider range of image formats, and might be configured to accept only a preset list of origins for the full image files. In any case, by removing the need to download the full images and engaging the corresponding specialized analytical libraries [Teodoro 2013] [Ali 2013] programmatically, the imageBox architecture opens the distribution of domain facing applications to any Biomedical Informatics Group able to develop them. The critical data-driven nature of this architecture is that the application code can be executed by those with access to the data, without compromise of the image privacy or any requirement to support server side components. The advantages of the resulting consumer-facing architectures go beyond the promise of commoditizing specialized image analysis operations - they are at the very core of the emergence of new data-intensive disciplines studying disease [Colen 2014] such as Computational Pathology [Louis 2015, Roth 2015]. More to the point, the imageBox mediated analysis of large images such as whole slides, is reduced to engaging the mediation of a HTTP REST application programming interface (API) of a cloud computing resource.

Conclusion

The imageBox implementation architecture was designed to address user-facing large image analysis scenarios. To demonstrate the feasibility of a data-driven deployment, a web application was developed to call whole slide images in SVS format from both public (TCGA) and private (Box.com, DropBox) stores. This open source application is made freely available at sbu-bmi.github.io/imagebox both to demonstrate the feasability of the imageBox architecture, and to offer reference for other box deployments that may emerge to balance the load of their usage. This report argues that the cost-effective distribution achieved in this manner identifies novel opportunities for cloud-based image analysis along a path pioneered by other domains of application such as computational genomics.

Acknowledgements

This work was supported in part by 1U24CA180924-01A1 from the NCI, R01LM011119-01 and R01LM009239 from the NLM.

References

Aji A, Wang F, Vo H, Lee R, Liu Q, Zhang X, Saltz J. Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce. Proceedings VLDB Endowment. 2013 Aug;6(11). pii: p1009. [PMID:24187650]

Almeida JS, E Iriabho, VL Gorrepati, S Wilkinson, DE Robbins, A Grüneberg, JR Hackney (2012) ImageJS: personalized, participated, pervasive and reproducible image bioinformatics in the web browser. J Pathology Informatics 3:25 [PMID:22934238].

Box, Box for Healthcare, retrieved Sep 2015 from www.box.com/healthcare.

Colen R, Foster I, Gatenby R, Giger ME, Gillies R, Gutman D, Heller M, Jain R, Madabhushi A, Madhavan S, Napel S, Rao A, Saltz J, Tatum J, Verhaak R, Whitman G. NCI Workshop Report: Clinical and Computational Requirements for Correlating Imaging Phenotypes with Genomics Signatures. Transl Oncol. 2014 Oct 24;7(5):556-69. doi: 10.1016/j.tranon.2014.07.007. eCollection 2014 Oct. [PMID:25389451] PubMed PMID: 25389451; PubMed Central PMCID: PMC4225695.

Goode, A., Gilbert, B., Harkes, J., Jukic, D., & Satyanarayanan, M. (2013). OpenSlide: A vendor-neutral software foundation for digital pathology. Journal of pathology informatics, 4. [PMID: 24244884].

Google Inc, HIPAA Compliance & Data Protection with Google Apps - HIPAA implementation guide (2015) https://static.googleusercontent.com/media/www.google.com/en/us/work/apps/terms/2015/1/hipaa_implementation_guide.pdf.

Fedorov A., Beichel R., Kalpathy-Cramer J., Finet J., Fillion-Robin J-C., Pujol S., Bauer C., Jennings D., Fennessy F., Sonka M., Buatti J., Aylward S.R., Miller J.V., Pieper S., Kikinis R. 3D Slicer as an Image Computing Platform for the Quantitative Imaging Network. Magn Reson Imaging. 2012 Nov;30(9):1323-41. [PMID:22770690].

Leica 2015 http://www.leicabiosystems.com/pathology-imaging/aperio-digital-pathology.

Louis DN, Feldman , Carter AB9, Dighe AS, Pfeifer JD1, Bry L, Almeida JS, Saltz J1,2, Braun J, Tomaszewski JE, Gilbertson JR, Sinard JH, Gerber GK, Galli SJ, Golden JA, Becich MJ (2015) Computational Pathology. Arch Pathol Lab Med. dx.doi.org/10.5858/arpa.2015-0093-SA [PMID:26098131].

Microsoft 2015 https://onedrive.live.com/about/en-us/business

ONC, The Office of the National Coordinator (2015) Connecting Health and Care for the Nation - A Shared Nationwide Interoperability Roadmap. https://www.healthit.gov/sites/default/files/nationwide-interoperability-roadmap-draft-version-1.0.pdf

Roth KA, JS Almeida (2015) Computational Pathology as the New Big Data Microscope American Journal of Pathology, Vol. 185, No. 3 [PMID:25701882].

Teodoro G, Pan T, Kurc TM, Kong J, Cooper LA, Podhorszki N, Klasky S, Saltz JH. High-throughput Analysis of Large Microscopy Image Datasets on CPU-GPU Cluster Platforms. IPDPS. 2013 May;2013:103-114. [PMID:25419546]

Wilbur, D. C. (2014). Digital pathology: Get on board—the train is leaving the station. Cancer cytopathology, 122(11), 791-795. [PMID: 25236488]

Wilkinson, S. R., & Almeida, J. S. (2014). QMachine: commodity supercomputing in web browsers. BMC bioinformatics, 15(1), 176. [PMID:24913605].