

Grids and Web 2.0 Overlap or Complementary?

Wouter Alexander de Landgraaf
Faculty of Computer Science
Vrije Universiteit Amsterdam

I. INTRODUCTION

Both Computer Grids and Web 2.0 are concepts based on usage of the Internet. Grids have been around for some time in many forms, with standardization currently being the main focus, while Web 2.0 is a recent topic that has enabled new applications and business models via usage of the web. The main questions we will look at in this paper is if these concepts overlap in functionality and design and where these concepts could be combined to provide new and exciting applications of technology.

Research questions

- What is Web 2.0?
- What are (Computer) Grids?
- Which Grid-requirements can be satisfied by Web 2.0?
- Which Grid-requirements can't be satisfied by Web 2.0?
- Where is the mutual benefit between Grids and Web 2.0?

The rest of this paper makes an attempt to answer these 5 questions. If we want to understand what the differences are between Web 2.0 and Grids, we first have to define the underlying concepts of both these terms. In chapter 2, we will look at Web 2.0 and the concepts surrounding it. In chapter 3, we will look at what grids are and what they provide. In chapter 4, we will show the overlap between these two concepts and show where both concepts can be used in order to provide a mutual benefit. In chapter 5 we provide a conclusion of our findings.

II. WEB 2.0: WHAT IS IT?

Since the dot-com crash many products have been launched. Those that have been deemed successful are increasingly being called "Web 2.0", the latest generation of applications on the Web. But is there a single definition with which to summarize what "Web 2.0" is all about?

The term Web 2.0 is nowadays used in many contexts, not in the least by numerous start-ups trying to ride on the hype. There are even serious fears of another Internet bubble[2]. Web 2.0 is a vague term, as can be seen from a number of concepts derived from a small round of projects claiming to be "Web 2.0":

- Collaboration, ratings & tags (ie. collective intelligence, examples: Digg, YouTube)
- User-generated content (examples: blogs, wiki's, YouTube, Flickr)
- Rich Internet Applications, AJAX-enabled websites (JavaScript, CSS)

- Representational State Transfer (ReST) HTTP interfaces
- XML-or-JSON-over-HTTP
- Feeds (RSS/Atom)
- Streaming Media
- Community-oriented, social networking, personal profiles (examples: Myspace, Facebook, Hyves)
- Widgets, personalized homepages (Netvibes, iGoogle)
- Mash-ups (people using above to build something new)
- "Cloud Computing", usage of remote resources via a web interface

The main problem of Web 2.0 thus is that there isn't a single definition. Tim O'Reilly is thought by many to be the founder of the term, but even he notes that there is a large degree of controversy [12]:

... there's still a huge amount of disagreement about just what Web 2.0 means, with some people decrying it as a meaningless marketing buzzword, and others accepting it as the new conventional wisdom.

In the same article, he tries to define Web 2.0 as follows:

- The Web As Platform
- Harnessing Collective Intelligence
- Data is the Next Intel Inside
- End of the Software Release Cycle
- Lightweight Programming Models
- Software Above the Level of a Single Device
- Rich User Experiences

There is a lot to comment about O'Reilly's definition. He even speaks about peer-to-peer networking, scripting languages and business methods that strictly spoken have nothing to do about the web. Never the less, we will look at each of the above 7 "Web 2.0 rules" in order to get to the factual details.

A. The Web As Platform

The key example here, as seen by O'Reilly, is Google. Google is a company that completely relies on providing services via the web: Google Search is the search engine of the World Wide Web, Google AdWords is the largest online advertising network.

The main comparison O'Reilly makes is between Netscape and Google; Google's prime asset is the data in its web search engine, while Netscape's prime asset was its web browser. While Netscape tried to promote the "web as a platform" idea, it failed because it didn't make use of the emerging platform. Nowadays the Web is a true platform; you can

write limited applications using Java applets and Flash/Flex, and Asynchronous Javascript And XML (AJAX) allows web developers to create websites with much more functionality than before.

O'Reilly also tries to compare Akamai and Bittorrent. Although both networks are used to distribute files by using a distributed network of peers, the comparison is one of apples and oranges; Akamai is used as a high-performance distributed proxy, used primarily by large corporations, that serves files depending on the location of the client. Bittorrent on the other hand leverages copies of the files on the user's PC that have been previously downloaded, leading to an expanding number of copies available for downloading, which in turn speed up transfers for new users. The whole comparison is off-target as both Akamai and Bittorrent serve a different purpose, with Bittorrent even being a separate protocol compared to the protocols used for the Web.

B. Harnessing Collective Intelligence

The poster-child of harnessing collective intelligence is undoubtedly the online encyclopedia Wikipedia. According to O'Reilly, sites that enable user-input to drive their sites are a core part of what "Web 2.0" is all about. Users should be able to upload media, stories, links or at the very least be able to rate and tag the content of others.

Another aspect of a user-centric Web is the rise of blogs. Naturally, personal websites and online diaries were more-or-less the first types of sites that were available, however the fact that users can now publish information about their kittens directly via a web browser instead of hard-coding a HTML page is an improvement that has led to millions of weblogs. With so many locations to read information, syndication of this information was the next step, RSS being the prime example.

A user-centric web is the best definition of "Web 2.0". Online collaboration is something nearly all Web 2.0 sites allow in one form or the other, and although it technically doesn't have a lot to do with HTTP or HTML, it does expand the degree of functionality available online.

C. Data is the Next Intel Inside

What O'Reilly meant with this, is that data has become the key to success on the web. Related to the harnessing of collective intelligence, the largest corporations and projects active in Web 2.0 are those with large amounts of data; Google has its search database, Amazon has its book database, Wikipedia has its encyclopedia database. The degree to which this data is made publically accessible differs. Amazon provides an API to search its database and all data in Wikipedia is free to use and distribute, while Google recently has closed off its SOAP programming interface to its search database.

D. End of the Software Release Cycle

What O'Reilly means with this phrase is that software is supplied as a service. With this business model, software is no longer sold as a product over-the-counter, but instead users

access the application directly via their web browser. This model has been used before (even before the Internet, for example via terminals connected to mainframes and dial-up banking services), however recently such services are starting to become automated entirely using Web Services (SOAP, ReST and XML-RPC). Web 2.0 isn't equated directly with Web Services however; there is a strong preference in using ReST over HTTP-enabled Remote Procedure Call protocols like SOAP and XML-RPC.

E. Light-weight Programming Models

The idea behind this phrase is that the web should be composed of easily-reusable components. O'Reilly also mentions a number of "Mash-ups", sites that use commodity web services like Google Maps in order to create a new service.

Reuse of software is nothing new; concepts like object oriented programming, agile development and even functional programming where all to a degree developed in order to facilitate reuse of generic components. The Free Software/Open Source Software community is all about being able to reuse existing software without restrictions, building upon the work of others.

The main difference with reuse via the web is that the functionality is "somewhere out there" on the Internet, instead of locally as a software library. This again is related to the usage of data; developers get an easy-to-use web service interface to large amounts of data and functionality. The bright-side is that developers get access to this API; the drawback is that developers only get access to this API, and not to the data or the source code that is behind it. Such components are easy to deploy and easy to use by a large amount of users, so allowing developers to easily "mash up" multiple web services together in the creation of a new web application.

F. Software Above the Level of a Single Device

The example O'Reilly uses here is the iTunes/iPod combination, where users buy music via the online iTunes store, download the music to their computer and/or iPod.

This example however seems to be counter to the sub-definition of having "software above the level of a single device". Naturally, you can also play your music via iTunes, however the embedded digital rights management (DRM) ensures that the music you download from the iTunes store can't be played on any other portable music player. Although the iPod is very successful, a large majority of iPod users don't download their music from iTunes; instead, they have converted bought CDs into mp3 files or they download these music files from other sources.

What can be distilled is that the web platform allows developers to target multiple devices. Nowadays, you can browse the web from your PC or laptop, but also from your hand-held, your mobile phone, your tablet-PC, your game console (Wii) or media center. In "Web 2.0"-theory the web platform should be one unified target that works equally well for all these devices; in practice the web developer has to fine-tune many aspects of his website in order for it to work well on all these devices.

G. Rich User Experiences

The most common (and visual) "Web 2.0 experience" is the use of Rich Internet Applications (RIA). Instead of each mouse-button click leading to retrieving a whole new web-page, parts of the web-page are updated or modified, possibly by retrieving or sending data to the remote web server asynchronously. The most common technique used to achieve this is Asynchronous Javascript And XML (AJAX), but techniques like Flash, Java applets and Silverlight also fall into the RIA-definition.

The poster-child for AJAX-techniques has been Google with Gmail and Google Maps. The interaction a user has is comparable to the performance of a local application, however without the user having to install any piece of software (other than a web browser).

O'Reilly comments that these techniques aren't particularly new. Both Microsoft and Netscape experimented with these techniques in the '90s, and Adobe's Flash has been around for quite some time as well. Although many people are working on making local PC-applications on the web, it are often new applications (like Flickr, MySpace and Google Maps) that have made the most headway. It is interesting to note that shortly after Google Maps, Google Earth was released which (again) is a local PC-application, but provides much more functionality and a better user interface compared to its web-based counterpart. It is clear that there is a place for local PC-applications, even in the Web 2.0-era.

A Web 2.0 Summary

It is clear from the above sections that there isn't a single Web 2.0 definition. O'Reilly tried it again in a following article[13], with a definition that doesn't prove any more useful:

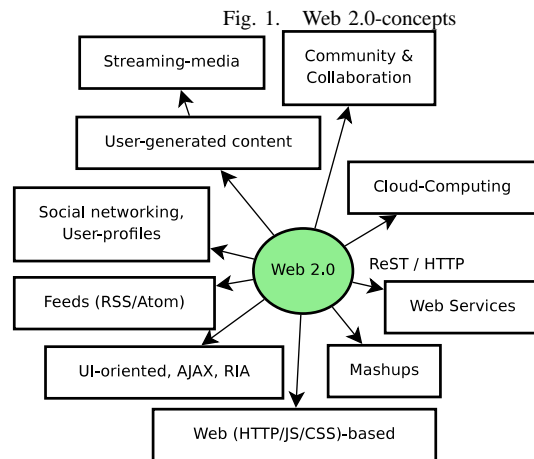
Web 2.0 is the business revolution in the computer industry caused by the move to the Internet as platform, and an attempt to understand the rules for success on that new platform. Chief among those rules is this: Build applications that harness network effects to get better the more people use them.

In summary, Web 2.0 is a combination of all the Web-based approaches that worked since the dot-com crash. We can't provide a single definition, because there isn't one. What we can do however is note which concepts are seen as being "Web 2.0". These concepts are shown in Figure 1.

Using the explanation above and this network-diagram we should have enough information to compare Web 2.0 with Grids.

III. GRIDS: WHAT ARE THEY?

Using multiple computers for a single task isn't new. Already in the 50's and 60's, multiple hundred-thousand-dollar computers were connected together in order to speed up computations. Considering the speed at which those computations took place, it was almost a requirement in order to get more than the most basic of calculations done



within a reasonable amount of time, but this is with the hindsight of nowadays having more computational power in a phone than in a 60s-era supercomputer. Over the last 60 years computing resources have grown at an exponential rate, however there has always been a need for more. With ever-larger amounts of scientific data, there is a strong need for global distributed computing. Termed "grid computing", over the last 10 years researchers, developers and industry have come together in order to allow computing resources to be managed and allocated across organizations [3].

In the late 90's Ian Foster and Carl Kesselman [6], [8] laid out their ideas on what the future of computing should be. Instead of everyone having their own processing power and data, people would be able to 'plug in' to a grid infrastructure that would give users access to these resources, analogous to the electricity grid infrastructure. Foster and Kesselman would go on to be key members in the Open Grid Forum, a group of academic, community and industry partners that work together in order to standardize the grid. Foster and Kesselman have driven the development of the Globus Toolkit [5], the current de-facto grid middleware.

One definition by Foster [4]:

A Grid is a system that coordinates resources that are not subject to centralized control using standard, open, general-purpose protocols and interfaces to deliver nontrivial qualities of service.

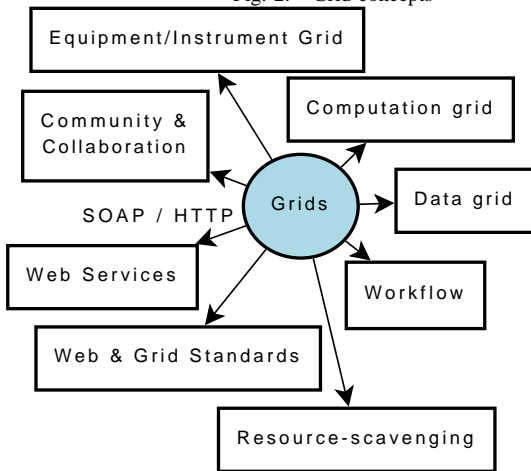
In comparing Web 2.0 with Grids, we will use this definition.

From a completely different point of view, community or scavenger grids became popular in the early 00's. These community-driven projects let users donate unused CPU cycles for various tasks, like protein folding [11], fighting cancer [16] and the search for extra-terrestrial life [10], [1]. These voluntary projects gathered large groups of users, each group trying to out-do the rest in the amount of work units they could complete.

Like Web 2.0, grids too have been subjected to a certain amount of hyperbole. A number of concrete features often appointed to grids:

- Access to shared Computational, Data and Instrument resources via a network

Fig. 2. Grid-concepts



- Multiple organizations collaborating in providing these resources
- Collaboration by users, members of virtual organizations share data and cooperate
- Web/Grid standards, SOA/SOAP-based protocols.
- Workflow management, help in organizing and automating research and data flows
- Scavenging-grids/community-grids, donation of unused resources

These Grid concepts are shown in Figure 2. Together with Figure 1, we can determine which concepts exist in both diagrams in the next chapter.

IV. GRIDS AND WEB 2.0

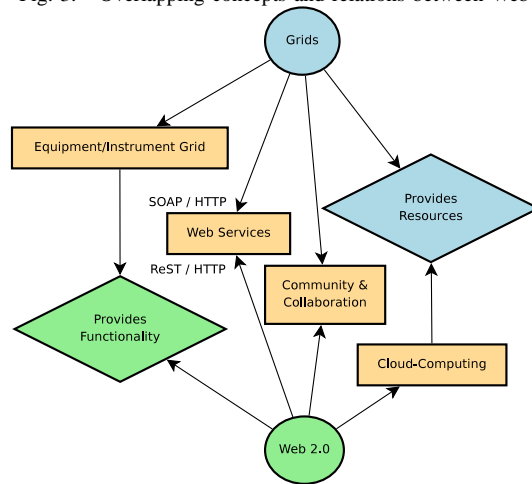
Now we have a description and a list of concepts for both Grids and Web 2.0, we can look at the concepts that are used by both. The prime difference between Grids and Web 2.0 is that Grids provides access to resources (CPU cycles, data storage) while Web 2.0 applications provides access to specific functionalities (for example: sharing of online media, communication between peers in a social network, publishing a blog with the ability to comment and tag stories). Naturally there is overlap between the two, as shown in Figure 3. These concepts we will now explore in detail.

A. Concept-overlap: Web Services and XML

Both Grids and Web 2.0 depend on the availability of Web Services. Grids, in the form of WSRF, have been moving towards open standards based on XML for quite some time[7]. The use of open standards like XML facilitates interoperability between resource providers and Grid-users. Increasingly, Grid middleware uses WSDL-descriptions of web services and SOAP-based XML messages for RPC-communication.

Web 2.0 also has a strong dependency on XML, however it isn't holy. XML is seen as inefficient for communication to and from client-side applications, and JSON-based (JavaScript Object Notation) messages are frequently used instead of XML. Also, the Web 2.0 has adopted ReST

Fig. 3. Overlapping concepts and relations between Web 2.0 and Grids



(Representational State Transfer) instead of SOAP for communication.

The main difference between SOAP and ReST is that SOAP treats HTTP solely as a transport protocol; all the information required for the remote request is stored in SOAP's XML-based message body. SOAP could in principle also use a different protocol other than HTTP. ReST on the other hand uses URIs and HTTP methods (GET, POST) to determine the end-point and type of the remote request. ReST isn't actually a protocol, but more a style of using HTTP. ReST web services, if you can call them that, are document-based and not strictly remote procedure calls. There are many WSDL and SOAP-libraries that help in using web services, ReST on the other hand is simple enough to use with a HTTP library or (web) browser.

In the end it has more to do with convention and politics than it does with technology. Grid standards have become very broad and complicated, from this point of view it might be interesting to develop a minimized ReST service for accessing Grid resources. Certain successful Web 2.0 companies use SOAP, others use ReST and many offer both types of interfaces. The main advantage of ReST is the simplicity in using the interface, while SOAP allows true remote procedure calls.

B. Concept-overlap: Cloud Computing

Cloud computing is a phrase that has become popular only recently. In short, cloud computing is the usage of remote resources via a web application or web service, of which the remote resources can be seen as a grid.

It is likely that cloud computing will become more popular in the future, with large names as Google and Amazon pushing this meme. One should not however confuse this Web 2.0-ish concept with Grid Computing. With Cloud Computing a web-interface is used to access remote resources, however nothing is said about how these resources are organized.

Comparing Cloud Computing to the Grid-definition of Foster ("A Grid is a system that coordinates resources that

are not subject to centralized control”), we can see that Cloud Computing doesn’t mean that grids are used. The resources used in Cloud Computing can be controlled centrally in a single cluster, or even on a single server. Cloud Computing merely is a definition on how these resources can be used.

Amazon especially is focussing on Cloud Computing. With its Simple Storage Service, also known as Amazon S3[14], [9], users have access to a central data storage interface, while with Amazon’s Elastic Compute Cloud (Amazon EC2)[15] you can upload a machine image which is run as a virtualized instance on Amazon’s compute servers. What in Grid Computing is seen as a goal, Amazon is trying out in practice: As much storage as you want, and as many computation nodes as you want. For a fee, of course.

Again, the main difference is that with Cloud Computing there tends to be a central facility instead of the sharing of resources across multiple organizations. The centralization makes deployment of Cloud Computing a lot easier, and this allows suppliers to focus on a dependable infrastructure and easy-to-use interfaces. Amazon offers both ReST and SOAP interfaces and a number of start-ups are using Amazon’s services for computationally-expensive programs, web hosting and the hosting of media in order to offload their own web servers, for example.

The question then arises if true Grid Computing is able to compete against Cloud Computing. With both concepts costs are involved, however with Grid Computing the additional bottleneck of distributing the funds according to resource usage might prove to be quite complicated. On the other hand, usage of Cloud Computing resources is subject to no particular standard or API, while Grid Computing offers a large body of standards one could use.

One interesting approach with Cloud Computing is the use of a virtualized machine image as the means of offering computational resources. This allows users to develop and test their image locally, while being ensured that it will work on the compute cluster. This approach is many times more user-friendly compared to the defining of commands and arguments common with Grid Computing. There is of course a cost; the virtual machine image would run slower compared to a non-virtualized application, however the use of virtualized Linux images has been improved dramatically over the last few years. This, together with efforts by AMD and Intel to provide virtualization at a processor-level, means that the performance of virtual images has become a non-issue. Many providers currently sell single virtual machines for web hosting and companies are turning to virtualization in order to consolidate their server parks.

In short, Grid Computing is about multiple organizations and defined standards in using distributed resources, while Cloud Computing is about a single (commercial) entity offering usage of its resources via a custom web-interface and possibly virtualization.

C. Concept-overlap: Community and Collaboration

One concept frequently used in Grids is the usage of Virtual Organizations. Each VO consists of users from multiple

real organizations that work together in a certain domain and provide each other certain rights on the Grid.

The prime reason for these VOs is trust; if you want to exchange research data with colleagues at your organization and those at other organizations you don’t want to specify the access rights for each particular user, you want to form a VO with equal rights within that VO. Another reason is trust from an organizational point of view. Perhaps your organization only wants to provide resources to a particular VO that is performing ”acceptable” research from a financial point of view. The collaboration-aspect however isn’t focussed on explicitly, although resources are shared. The sharing of results and findings is often done, but informally.

Web 2.0-based communities are focussed instead on user-to-user and user-to-group relationships. People form communities based on their hobbies, interests and location, which is comparable to that of a VO. Collaboration can be very broad (Wikipedia) or very specific (group-based blogs), however this is done in a formal way via a single web-based platform. In virtual organizations it is often vague as to who is working on what; with a Planet-like combination of blogs these virtual organizations could be given more ”body”, so fostering a more community-like approach to research using a Grid-environment.

D. Concept-overlap: Equipment and Instrument Grids

Equipment/Instrument Grids are different compared to Computation and Data Grids in that they provide both a particular functionality (say, to use a number of telescopes) and the resources to use this functionality. The difference is that functionalities can’t be grouped together (you have the functionality to view the sky, but multiple telescopes pointing at the same location doesn’t offer anything more than a single telescope), where combining resources does (you can combine multiple data storage sites to allow storage that wouldn’t fit on a single site).

Instrument Grids are probably in the best position to profit from Web 2.0 concepts. Instrument Grids offer a particular functionality, around which a user-friendly web interface can be developed for remote usage. A community of users can be created around the web interface, who share their collected data, processed results and findings. As with Wikipedia and Amazon, the collected data is the key, instead of a particular resource.

V. CONCLUSION

In this paper we have seen what Web 2.0 and what Grids are. The combination of the two has been shown, and the concepts that overlap have been described and compared.

A. Which Grid-requirements can and can’t be satisfied by Web 2.0?

First we need to go back to Foster’s Grid definition: ”A Grid is a system that coordinates resources that are not subject to centralized control using standard, open, general-purpose protocols and interfaces to deliver nontrivial qualities of service”.

The Cloud Computing concept comes closest to Grid Computing in that resources are coordinated and offered via an interface with a predefined quality of service. As mentioned, the main difference is that Grids offer distributed resources instead of merely centralized resources and Grids offer resources via standardized interfaces.

The coordination-aspect of Grids is present in Web 2.0, but on the social level instead of on the technical level. Web services are used in combining functionalities, however with Grids resources are combined using accepted Grid standards via the Open Grid Forum. Web services do use standards, however these standards are on a lower level compared to those used in a Grid infrastructure

B. Where is the mutual benefit of Grids and Web 2.0?

The mutual benefit between Grids and Web 2.0 lies in particular in the usage of Instrument Grids. Grid developers should also take a look at how commercial entities are pushing Cloud Computing forward and learn from their experiences, even though Cloud Computing is relatively simple compared to Grid Computing.

On the collaboration-level, Grid communities can and do use Web 2.0-concepts in facilitating the exchange of ideas. A Grid-based Virtual Organization should go beyond the question of mere trust and access control towards the fostering of cooperation and relationships between organizations active on similar projects.

REFERENCES

- [1] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. Seti@home: an experiment in public-resource computing. *Commun. ACM*, 45(11):56–61, 2002.
- [2] T. Economist. Business and the internet: Fears of another internet bubble, 2005.
- [3] I. Foster. Internet computing and the emerging grid. *Nature*, 408(6815), 2000.
- [4] I. Foster. What is the grid? a three point checklist. Technical report, GRIDToday, July 2002.
- [5] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal on Supercomputer Applications*, 1997.
- [6] I. Foster and C. Kesselman, editors. *The grid: blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.
- [7] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. Grid services for distributed system integration. *IEEE Computer*, 35(6):37–46, 2002.
- [8] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal on Supercomputer Applications*, 15(3), 2001.
- [9] M. I. Jrn Altmann and A. A. B. Mohammed. Taxonomy of grid business models. *Grid Economics and Business Models*, 4685:29–43, 2007.
- [10] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Lebofsky. Seti@home-massively distributed computing for seti. *IEEE MultiMedia*, 3(1):78–83, 1996.
- [11] S. M. Larson, C. D. Snow, M. Shirts, and V. S. Pande. Folding@home and genome@home: Using distributed computing to tackle previously intractable problems in computational biology, 2002.
- [12] T. O'Reilly. What is web 2.0. design patterns and business models for the next generation of software. Technical report, O'Reilly Media, 2005.
- [13] T. O'Reilly. Web 2.0 compact definition: Trying again. Technical report, O'Reilly Media, 2006.
- [14] M. Palankar, A. Onibokun, A. Iamnitchi, and M. Ripeanu. Amazon s3 for science grids: a viable solution? Technical report, University of South Florida, 2007.
- [15] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. In *EuroSys '07: Proceedings of the ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, pages 275–287, New York, NY, USA, 2007. ACM.
- [16] Worldcommunitygrid.org.