Preliminary Thoughts on Services without Servers

Philip Machanick and Kieran Hunt
Department of Computer Science
Rhodes University, Grahamstown
Tel: +27 46 603 8635, Fax: +27 46 603 7608
email: {p.machanick,kieran.hunt}@ru.ac.za

Abstract-Warehouse-scale computing supports cloud-based services such as shared disk space, computation services and social networks. Although warehouse-scale computing is inexpensive per user, the cost to entry is high, and the pressures to generate revenues to cover costs leads service providers to pursue monetizing services aggressively. In this paper, we explore some ideas for removing the need for central servers by exploiting peer-to-peer technologies.

Index Terms—distributed systems, cloud, peer-to-peer

I. INTRODUCTION

Large-scale services such as shared file systems (Google Drive, Apple iCloud, Dropbox and Microsoft's OneDrive) and on-demand computation resources (such as Amazon's Elastic Compute Cloud, or EC2) have started to proliferate with the generic "cloud" label. Such services build on infrastructure originally created to support large-scale services such as Google search, Amazon's bookstore and Facebook. Many of these large-scale services – Google's search, Facebook and Twitter to name a few – are free to use, but have a commercial aspect in that their creators use user traffic to generate revenue streams such as advertising.

Despite impressive gains in implementation of such services [Mishra *et al*. 2010], they fall far short of the promise of distributed computing. They lack a transparent namespace – most such services still look more like networked services with names that appear to relate to a particular server, even if there is some virtualization behind the scenes. Scalability is implemented by large-scale resources in a small number of places, rather than by placing resources near the users. Cost is not shared over the users, except in the indirect sense that a large user base makes for a more attractive target for advertisers. The last point also points to one of the weaknesses of this sort of service from the user point of view: if *you* are seen as the product, as was famously said of Facebook [8], "your" service provider constantly is under pressure to monetize you.

While some services inherently are salable – e.g., Amazon's EC2 generates revenues directly [12] – providing services that users do not expect to pay for should be based on shared cost-sharing, rather than on free services paid for by advertising. Otherwise, the temptation to monetize invasion of privacy is too high.

In this paper, we explore the extent to which an existing shared-cost model, peer-to-peer (P2P) file-sharing, can adapt to a wider range of services. We start with specific services, then generalize to wider possibilities.

The remainder of this paper is structured as follows. Section II summarizes previous work, including approaches to scalability and distributed computing, as well as P2P technologies. Section III outlines how some simple services can be implemented using P2P, including existing work and our own ideas, and Section IV contains conclusions.

II. BACKGROUND

In this section we briefly review the relationship between distributed computing and the cloud, which is a poor approximation to the intent of distributed services, and the broader concept of scalable services.

A. Distributed Computing and the Cloud

Distributed computing in its general form implies a number of properties [14]:

- *location-transparent naming* – a name of an entity should be related to its logical purpose or relationship to other entities, not where it is situated
- *locality-independent resources* – whether a resource is local, on a local network or in a more remote location is a performance detail, and should not be an inherent property of any resource
- *decentralized scalable infrastructure* – a system should be able to work over a wide geographic region, which also implies an appropriate level of security.

Cloud-based services violate basic properties of distributed computing. To the user, it is clear that there is an external server, and hence, a distinction between purely local resources and cloud-based services. Names are therefore not full location-transparent. Further, cloud services require connection to a server (even if limited offline activity may be allowed), making network connection essential rather than a performance detail. Scalability is achieved by concentrating resources in warehouses of computers [3], rather than by distributing resources widely.

B. Scalable Services

In general, how can services be scaled up? Some of the scalability problem is in scaling up large-scale computation; the general case is hard because some problems are not partitionable [1]. Here we constrain ourselves to services where computation is not large-scale; even so we have problems of scaling up naming. Traditionally, name-scaling has been a function of middleware [2]. We can however isolate scalable naming as a single concept as, for example, in distributed hash tables (DHT) [13, 9], which are widely used in P2P systems (though some have argued that sharing

in P2P systems is inherently scalable [5] without DHTs).

In general, scalable services should not depend on the number of users to be viable and – even better – should become more viable as the number of users increases.

III. Simple P2P-Based Shared Services

A number of services layered on P2P already exist. For example, Skype voice over IP (VoIP) is layered on a proprietary P2P protocol [4]. BASS is a scalable video on demand service based on Bittorrent [6].

More recently, the Bittorrent Sync application programming interface (BTS API) has been released[1], allowing applications to be layered on top of the Bittorrent Sync service [7], which provides secure P2P file sharing. Applications that use the BTS API include Vole[2], a twitter-like service that shares the underlying files using BTS rather than a central server, and SyncNet[3], which implements a web server by distributing the files across clients.

These services indicate the general possibility of breaking dependence on central servers.

Our work builds on these foundations. We are investigating the extent to which you can completely break away from networked services and implement true distributed services based on P2P protocols. Our starting point is implementing a twitter-like service on top of the BTS API, using Java for portability – thereby eliminating the web browser. We will follow with a service more like Facebook, then look into how services like email can be made to work in a purely distributed fashion [10].

Implementation of a twitter-like application built on the BTS API should be very simple. Users wanting to share updates exchange their encryption keys (these could be mapped to usernames or handles) to allow other users to request and download their updates. A simple GUI sets such a program apart from traditional, web-based applications. Otherwise, BitTorrent Sync handles most of the operation.

IV. Conclusions

Distributed computing is a powerful idea that has somehow got lost in a network-centric world. Warehouse-scale computing uses distributed computing concepts internally, including highly scalable distributed file systems, yet the interface presented to the user uses network-like names, even if the actual resource named may be disguised.

The proposal presented here is to implement true distributed services without servers, based on P2P technology. The extent to which such services can be implemented is part of our investigation; if we can implement a significant range of such services, we can reduce the need for central resources, and hence the pressure to monetize services even when it is inappropriate to do so.

Further, if these ideas work, not only can they scale very well, but they have a very low barrier to entry.

Acknowledgements

This work was undertaken in the Distributed Multimedia CoE at Rhodes` University, with financial support from Telkom SA, Tellabs, Genband, Easttel, Bright Ideas 39, THRIP and NRF SA (TP13070820716). The authors acknowledge that opinions, findings and conclusions or recommendations expressed here are those of the author(s) and that none of the above mentioned sponsors accept liability whatsoever in this regard.

[1] http://www.bittorrent.com/sync/developers/api

[2] http://vole.cc/

[3] http://jack.minardi.org/software/syncnet-a-decentralized-web-browser/

References

1. G.M. Amdahl. “Validity of the single processor approach to achieving large scale computing capabilities”, *Proc. Spring joint computer conference, AFIPS ’67*, 1967 pp. 483–485.
2. G. Ballintijn, M van Steen and AS Tanenbaum. “Scalable naming in global middleware” *Proc. 13th Int’l Conf. on Parallel and Distributed Computing Systems*, 1999 pp. 624-631.
3. L.A. Barroso and U. Hölzle. “The datacenter as a computer: An introduction to the design of warehouse-scale machines”, *Synthesis lectures on computer architecture*, 2009. doi:10.2200/S00193ED1V01Y200905CAC006
4. S.A. Baset and H. Schulzrinne. "An analysis of the Skype peer-to-peer internet telephony protocol." *arXiv preprint cs/0412017*, 2004.
5. Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. “Making gnutella-like P2P systems scalable.” *Proc. 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003, pp. 407-418.
6. C. Dana, D. Li, D. Harrison and C.-N. Chuah. “BASS: BitTorrent assisted streaming system for video-on-demand.” *IEEE 7th Workshop on Multimedia Signal Processing, 2005* pp. 1-4.
7. J. Farina, M. Scanlon, and M. Kechadi. “BitTorrent Sync: First Impressions and Digital Forensic Implications.” *Digital Investigation* vol. 11, pp S77-S86, 2014
8. N. Friesen. “Education and the social Web: Connective learning and the commercial imperative.” *First Monday* vol. 15, no. 12, 2010.
9. D. Korzun and A. Gurtov. “Survey on hierarchical routing schemes in ‘flat’ distributed hash tables.” *Peer-to-Peer Networking and Applications* vol. 4 no 4, pp 346–375, 2011.
10. P. Machanick. "A distributed systems approach to secure Internet mail." *Computers & Security* vol. 24 no. 6 pp 492–499 2005.
11. A.K. Mishra, J.L. Hellerstein, W. Cirne, and C.R. Das. 2010. “Towards characterizing cloud backend workloads: insights from Google compute clusters”, *SIGMETRICS Perform. Eval. Rev*. vol. 37, no. 4, pp 34-41, March 2010.
12. Y. Sangho, A. Andrzejak, and D. Kondo. “Monetary Cost-Aware Checkpointing and Migration on Amazon Cloud Spot Instances”, *IEEE Transactions on Services Computing*, vol. 5, no. 4, pp 512,524, Fourth Quarter 2012.
13. D. Tam, R. Azimi and H.-A. Jacobsen. “Building content-based publish/subscribe systems with distributed hash tables”, in *Databases, Information Systems, and Peer-to-Peer Computing*. Springer Berlin Heidelberg, 2004, pp 138-152.
14. A.S. Tanenbaum and M. van Steen. *Distributed Systems: Principles and Paradigms*. Upper Saddle River, NJ: Prentice Hall, 2002.

Philip Machanick received his PhD degree in 1996 from the University of Cape Town. His research interests include distributed computing, computer science education and bioinformatics.

Kieran Hunt received his undergraduate degree from Rhodes University in 2013 and is presently studying towards his Honours degree at the same institution. His research interests include distributed computing.