Texts in Computer Science

Editors
David Gries
Fred B. Schneider

For further volumes:
www.springer.com/series/3191

Kenneth P. Birman

Guide to Reliable Distributed Systems

Building High-Assurance Applications and Cloud-Hosted Services

Springer

Kenneth P. Birman
Department of Computer Science
Cornell University
Ithaca, NY, USA

ISSN 1868-0941 e-ISSN 1868-095X
Texts in Computer Science
ISBN 978-1-4471-2415-3 e-ISBN 978-1-4471-2416-0
DOI 10.1007/978-1-4471-2416-0
Springer London Dordrecht Heidelberg New York

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2012930225

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Setting the Stage

The *Guide to Reliable Distributed Systems: Building High-Assurance Applications and Cloud-Hosted Services* is a heavily edited new edition of a prior edition that went under the name *Reliable Distributed Computing*; the new name reflects a new focus on *Cloud Computing*. The term refers to the technological infrastructure supporting today's web systems, social networking, e-commerce and a vast array of other applications. The emergence of the cloud has been a transformational development, for a number of reasons: cost, flexibility, new ways of managing and leveraging large data sets. There are other benefits that we will touch on later.

The cloud is such a focus of product development and so associated with overnight business success stories today that one could easily write a text focused on the cloud "as is" and achieve considerable success with the resulting text. After all, the cloud has enabled companies like Netflix, with a few hundred employees, to create a movie-on-demand capability that may someday scale to reach every potential consumer in the world. Facebook, with a few thousand employees, emerged overnight to create a completely new form of social network, having the importance and many of the roles that in the past one associated with vast infrastructures like email, or the telephone network. The core Google search infrastructure was created by just a few dozen employees (by now, of course, Google has tens of thousands, and does far more than just search). And the cloud is an accelerator for such events: companies with a good idea can launch a new product one day, and see it attract a million users a week later without breaking a sweat. This capability is disruptive and profoundly impactful and is reshaping the technology sector at an accelerating pace.

Of course there is a second side to the cloud, and one that worries many corporate executives both at the winners and at other companies: the companies named above were picked by the author in the hope that they would be success stories for as long as the text is in active use. After all this text will quickly seem dated if it seems to point to yesterday's failures as if they were today's successes. Yet we all know that companies that sprang up overnight do have a disconcerting way of vanishing just as quickly: the cloud has been a double-edged sword. A single misstep can spell doom. A single new development can send the fickle consumer community rushing to some new and even more exciting alternative. The cloud, then, is quite a stormy

place! And this book, sadly, may well be doomed to seem dated from the very day it goes to press.

But even if the technical landscape changes at a dizzying pace, the cloud already is jam-packed with technologies that are fascinating to learn about and use, and that will certainly live on in some form far into the future: BitTorrent, for example (a swarm-style download system) plays key roles in the backbone of Twitter's data center, Memcached (a new kind of key-value store) has displaced standard file system storage for a tremendous range of cloud computing goals. MapReduce and its cousin Hadoop enable a new kind of massively parallel data reduction. Chubby supports scalable locking and synchronization, and is a critical component at the core of Google's cloud services platform. ZooKeeper plays similar roles in Yahoo!'s consistency-based services. Dynamo, Amazon's massively replicated key-value store, is the basis for its shopping cart service. BigTable, Google's giant table-structured storage system, manages sparse but enormous tabular data sets. JGroups and Spread, two commercially popular replication technologies, allow cloud services to maintain large numbers of copies of heavily accessed data. The list goes on, including global file systems, replication tools, load balancing subsystems, you name it. Indeed, the list is so long that even today, we will only touch on a few representative examples; it would take many volumes to cover everything the cloud can do, and to understand all the different ways it does those things. We will try and work our way in from the outside, identifying deep problems along the way, and then we will tackle those fundamental questions. Accordingly, Part I of the book gives a technical overview of the whole picture, covering the basics but without delving deeply on the more subtle technology questions that arise, such as data replication. We will look at those harder questions in Parts II and III of the text; Part IV covers some additional technologies that merit inclusion for reasons of completeness, but for which considerations of length limit us to shallow reviews.

Above, we hinted at one of the deeper questions that sit at the core of Parts II and III. If the cloud has a dark side, it is this: there are a great many applications that need forms of high assurance, but the cloud, as currently architected, only offers very limited support for scalable high assurance computing. Indeed, if we look at high assurance computing in a broad way, and then look at how much of high assurance computing maps easily to the cloud, the only possible conclusion is that the cloud really does not support high assurance applications at all. Yes, the cloud supports a set of transactional-security features that can be twisted this way and that to cover a certain class of uses (as mentioned earlier, those concerned with credit card purchases and with streaming copyright-protected content like movies and music from the cloud to your playback device), but beyond those limited use cases, high assurance technologies have been perceived as not scaling adequately for use in the cloud, at least in the scalable first tier that interacts with external clients.

The story is actually pretty grim. First, we will encounter two theorems about things we cannot do in cloud settings: one proves that fault-tolerant distributed computing is impossible in standard networks, and the second that data consistency cannot be achieved under the performance and availability requirements of the cloud. Next, we will find that the existing cloud platforms are designed to violate consistency as a short-cut towards higher performance and better scalability. Thus: "High

assurance in the cloud? It cannot be done, it cannot scale to large systems, and even if it could be done and it could be made to scale, it is not the way we do it."

The assertion that high-assurance is not needed in most elements of most modern cloud computing applications may sound astonishing, yet if one looks closely, it turns out that the majority of web and cloud applications are cleverly designed to either completely avoid the need for high-assurance capabilities, or find ways to minimize the roles of any high assurance components, thereby squeezing the high-assurance side of the cloud into smaller subsystems that do not see remotely as much load as the main systems might encounter (if you like, visualize a huge cache-based front end that receives most of the workload, and then a second smaller core system that only sees update transactions which it applies to some sort of files or databases, and then as updates commit, pushes new data out to the cache, or invalidates cached records as needed).

For example, just to pick an example from the air, think about a massive government program like the Veteran's Administration Benefits program here in the United States. This clearly needs strong assurance (all sorts of sensitive data moves back and forth), money changes hands (the VA system is, in part, a big insurance system), sensitive records are stored within the VA databases. Yet if you study such a system carefully, as was done in a series of White House reviews during 2010 and 2011, the match with today's cloud is really very good. Secure web pages can carry that sensitive data with reasonable protection. The relatively rare transactions against the system have much the same character as credit card transactions. And if we compare the cost of operating a system such as this using the cloud model, as opposed to having the Veteran's Administration run its own systems, we can predict annual savings in the tens of millions hundreds! Yet not a single element of the picture seems to be deeply at odds with today's most successful cloud computing models.

Thus, our statements about high-assurance are not necessarily statements about limitations that every single high assurance computing use would encounter. E-commerce transactions on the web work perfectly well as long as the transactional system is not down, and when we use a secured web page to purchase a book or provide a credit card number, that action is about as secure as one can make it given some of the properties of the PCs we use as endpoints (as we will see, many home computers are infected with malware that does not do anything visibly horrible, yet can still be actively snooping on the actions you as the user take, and could easily capture all sorts of passwords and other security-related data, or even initiate transactions on its own while you are fast asleep!) Notice that we have made a statement that does not demand continuous fault-tolerance (we all realize that these systems will sometimes be temporarily unavailable), and does not expose the transactional system to huge load (we all browse extensively and make actual purchases rarely: browsing is a high-load activity; purchasing, much less so). The industry has honed this particular high-assurance data path to the point that most of us, for most purposes, incur only limited risks in trusting these kinds of solutions. Moreover, one cannot completely eliminate risk. When you hand your credit card to a waiter, you also run some risks, and we accept those all the time.

Some authors, knowing about the limitations of the cloud, would surely proclaim the web to be "unsafe at any speed;" Indeed, I once wrote an article that had this title (but with a question mark at the end, which does change the meaning). The bottom line is that even with its limitations today, such a claim would be pure hyperbole. But it would be quite accurate to point out that the vast majority of the web makes do with very weak assurance properties. Moreover, although the web provides great support for secure transactions, the model it uses works for secure transmission of a credit card to a cloud provider and for secure delivery of the video you just licensed back to your laptop or Internet-connected TV, not for other styles of high-assurance computing. Given the dismal security properties of the laptops, the computing industry views Web security as a pretty good story. But could we extend this model to tackle a broader range of security challenges?

We can then ask another question. Is it possible that *scalable* high-assurance computing, outside what the cloud offers today, just is not needed? We emphasized the term "scalable" for a reason: the cloud is needed for large-scale computing; the methods of the past few decades were often successful in solving high-assurance computing challenges, but also limited to problems that ran on more modest scales. The cloud is the place to turn when an application might involve tens of thousands of simultaneous users. With six users, the cloud could be convenient and cheap, but is certainly not the only option. Thus unless we can identify plenty of important examples of large-scale uses that will need high assurance, it might make sense to conclude that the cloud can deal with high-assurance in much the same way that it deals with credit card purchases: using smaller systems that are shielded from heavy loads and keep up with the demand because they aren't really forced to work very hard.

There is no doubt that the weak-assurances of the cloud suffice for many purposes; a great many applications can be twisted to fit them. The proof is right on our iPads and Android telephones: they work remarkably well and do all sorts of amazing tricks and they do this within the cloud model as currently deployed, and they even manage to twist the basic form of web security into so many forms that one could easily believe that the underlying mechanism is far more general than it really is. Yet the situation would change if we tried to move more of today's computing infrastructure as a whole to a cloud model. Think about what high assurance really means. Perhaps your first reaction is that the term mostly relates to a class of very esoteric and specialized systems that provide services for tasks such as air traffic control, banking, or perhaps management of electronic medical records and medical telemetry in critical care units. The list goes on: one could add many kinds of military applications (those might need strong security, quick response, or other kinds of properties). There is a lot of talk about new ways of managing the electric power grid to achieve greater efficiency and to share power in a more nimble way over large regions, so that we can make more use of renewable electric generation capacity. Many government services need to be highly assured. And perhaps even non-politicians would prefer that it was a bit harder to hack their twitter, email and Facebook accounts.

So here we have quite a few examples of high assurance applications: systems that genuinely need to do the right thing, and to do it at the right time, where we're defining "right" in different ways depending on the case. Yet the list did not include very many of what you might call bread-and-butter computing cases, which might lead you to conclude that high assurance is a niche area. After all, not many of us work on air traffic control systems, and it is easy to make that case against migrating things like air traffic control to cloud models (even privately operated cloud models). Thus, it is not surprising that many developers assume that they do not really work on systems of this kind.

We're going to question that quick conclusion. One reason is that the average enterprise has many high assurance subsystems playing surprisingly mundane roles; they operate the factory floor equipment, run the corporate payroll, and basically keep the lights on. These are high assurance roles simply because if they are not performed correctly, the enterprise is harmed. Of course not many run on the cloud today, but perhaps if cloud computing continues to gain in popularity and continues to drop in cost (and if the reliability of the cloud were just a touch higher), operators may start to make a case for migrating them to cloud settings.

This is just the area where scalability and high assurance seem to collide: if we imagine using the cloud to control vast numbers of physical things that can break or cause harm if controlled incorrectly, then we definitely encounter limitations that today's cloud cannot easily surmount. The cloud is wonderful for scalable delivery of insecure data, and adequate for scalable delivery of certain kinds of sensitive data, and for conducting relatively infrequent purchase-style transactions. All of this works wonderfully well. But the model does not fit nearly so well if we want to use it in high-assurance control applications.

This is a bit worrying, because the need for high assurance cloud-hosted control systems could easily become a large one if cloud computing starts to displace other styles of computing to any substantial degree, a trend the author believes to increasingly probable. The root cause here is the tendency of the computing industry to standardize around majority platforms that then kill off competitors simply for economic reasons: lacking adequate investment, they wither and die. As cloud computing has flourished, it has also become the primary platform for most kinds of application development, displacing many other options for reasons of cost, ease of development, and simply because the majority platform tends to attract the majority of developers.

Some of the most exciting future uses of computing presume that computers will penetrate into the home and car and office to such a degree that we will be able to start to do intelligent, environmentally aware, dynamic control of those kinds of systems. Traffic lights and water heaters will begin to be cloud-controlled systems. Fragile, elderly patients will manage to live at home for many years, rather than in assisted living settings, because computing systems will play vital monitoring and assistance roles. Cars will literally drive themselves on densely packed highways, at far higher speeds and with tighter spacings than today's human drivers can manage. Those kinds of visions of the future appear, at least superficially, to presume a new kind of high assurance cloud computing that appears, at least superficially, to

be at odds with what today's cloud platforms are able to do. Indeed, they appear, again superficially, to be at odds with those theorems we mentioned earlier. If fault-tolerant computing is impossible, how can we possibly trust computing systems in roles like these? If the cloud cannot offer high assurance properties, how can the US government possibly bet so heavily on the cloud in sensitive government and military applications?

Accordingly, we must pose a follow-on question. What are the consequences of putting a critical application on a technology base not conceived to support high assurance computing? The danger is that we could wander into a future in which computing applications, playing critical roles, simply cannot be trusted to do so in a correct, secure, consistent manner.

This leads to the second and perhaps more controversial agenda of the present text: to educate the developer (be that a student or a professional in the field) about the architectures of these important new cloud computing platforms and about their limitations: not just what they can do, but also what they *cannot* do. Some of these limitations are relatively easily worked around; others, much less so.

We will not accept that even the latter kind of limitations are show-stoppers. Instead, the book looks to a future well beyond what current cloud platforms can support. We will ask where cloud computing might go next, how it can get there, and will seek to give the reader hands-on experience with the technologies that would enable that future cloud. Some of these enablers exist in today's commercial market place, but others are lacking. Consequently, rather than teaching the reader about options that would be very hard to put into practice, we have taken the step of creating a new kind of cloud computing software library (all open source), intended to make the techniques we discuss here practical, so that readers can easily experiment with the ideas the book will cover, using them to build applications that target real cloud settings, and could be deployed and used even in large-scale, performance-intensive situations. A consequence is that this text will view some technical options as being practical (and might even include exercises urging the reader to try them out him or herself using our library, or using one of those high-assurance technologies), and if you were to follow that advice, with a few hundred lines of code and a bit of debugging you would be able to run your highly assured solution on a real cloud platform, such as Amazon's EC2 or Microsoft Azure. Doing so could leave you with the impression would be that the technique is perfectly practical. Yet if you were to ask one of those vendors, or some other major cloud vendor, what they think about this style of high-assured cloud computing, you might well be told that such services do not belong in the cloud!

Is it appropriate to include ideas that the industry has yet to adopt into a textbook intended for real developers who want to learn to build reliable cloud computing solutions? Many authors would decide not to do so, and that decision point differentiates this text from others in the same general area. We will not include concepts that we have not implemented in our Isis^2 software library (you will hear more and more about Isis^2 as we get to Parts II and III of the book, and are welcome to download it, free of any charges, and to use it as you like) or that someone we trust has not worked with in some hands-on sense—anything you read in this book is real enough

that someone has built it, experimented with it, and gained enough credibility that the author really believes the technique to be a viable one. Just the same our line in the sand does not limit itself to things that have achieved commercial acceptance on a large-scale. You can do things with a technology like Isis^2 (and can do them right on cloud platforms like Amazon's EC2 or Microsoft's Azure) that, according to the operators of those platforms, are not currently available options.

What is one to make of this seeming disconnect? After all, how could we on the one hand know how to do things, and on the other hand be told by the operators and vendors in the cloud area that they do not know how to do those same things? The answer revolves around economics. Cloud computing is an industry born from literally billions of dollars of investment to create a specific set of platforms and tools and to support some specific (even peculiar) styles of programming. We need to recognize the amazing power of today's cloud platforms, and to learn how the solutions work and how to adapt them to solve new problems. Yet today's platforms are also limited: they offer the technologies that the vendors have gained familiarity with, and that fit well with the majority of their users. Vendors need this kind of comfort level and experience to offer a technology within a product; merely knowing how to solve a problem does not necessarily mean that products will embody the solutions the very next day. For the vendor, such choices reflect economic balancing acts: a technology costs so much to develop, so much more to test and integrate into their product offerings, so much more after that to support through its life style. Doing so will bring in *this* much extra revenue, or represent *such-and-such* a marketing story. Those kinds of analyses do not always favor deploying every single technical option. And yet we should not view cloud computing as a done deal: this entire industry is still at in its early days, and it continues to evolve at a breathtaking pace. The kinds of things we offer in our little library are examples of technologies that the author expects to see in common in use in the cloud as we look a few years out into the future.

This somewhat personal view of the future will not necessarily convince the world's main cloud providers to align their cloud platforms with the technologies covered in this text on day one. But change is coming, and nothing we cover in this text is impractical: everything we will look at closely is either already part of the mainstream cloud infrastructure, or exists in some form of commercial product one could purchase, or is available as free-ware, such as our own Isis^2 solution. A world of high-assurance cloud computing awaits us, and for those who want to be players, the basic elements of that future are already fairly clear.

Acknowledgements

Much of the work reported here was made possible by grants from the U.S. National Science Foundation, the Defense Advanced Research Agency (DARPA), the Air Force (specifically, the offices of the Air Force CIO and CTO, the Air Force Research Laboratory at Rome NY (AFRL), and the Air Force Office of Scientific Research (AFOSR)). Grants from a number of corporations have also supported

this work, including Microsoft, Cisco, Intel Corporation, Google and IBM Corporation. I wish to express my thanks to all of these agencies and corporations for their generosity. The techniques, approaches, and opinions expressed here are my own; they may not represent positions of the organizations and corporations that have supported this research. While Isis^2 was created by the author, his students and research colleagues are now becoming involved and as the system goes forward, it seems likely that it will evolve into more of a team effort, reflecting contributions from many sources.

Many people offered suggestions and comments on the earlier book that contributed towards the current version. I remain extremely grateful to them; the current text benefits in myriad ways from the help I received on earlier versions. Finally, let me also express my thanks to all the faculty members and students who decide to use this text. I am well aware of the expression of confidence that such a decision represents, and have done my best to justify your trust.

Ithaca, USA Ken Birman

Trademarks

Unix is a Trademark of Santa Cruz Operations, Inc. CORBA (Common Object Request Broker Architecture) and OMG IDL are trademarks of the Object Management Group. ONC (Open Network Computing), NFS (Network File System), Solaris, Solaris MC, XDR (External Data Representation), Jaa, J2EE, Jini and JXTA are trademarks of Sun Microsystems, Inc. DCE is a trademark of the Open Software Foundation. XTP (Xpress Transfer Protocol) is a trademark of the XTP Forum. RADIO is a trademark of Stratus Computer Corporation. Isis Reliable Software Developer's Kit, Isis Reliable Network File System, Isis Reliable Message Bus, and Isis for Databases are trademarks of Isis Distributed Computing Systems, Inc. Orbix is a trademark of Iona Technologies Ltd. Orbix+Isis is a joint trademark of Iona and Isis Distributed Computing Systems, Inc. TIB (Teknekron Information Bus), Publish-Subscribe and Subject Based Addressing are trademarks of TIBCO (although we use these terms in a more general sense in this text). Chorus is a trademark of Chorus Systems, Inc. Power Objects is a trademark of Oracle Corporation. Netscape is a trademark of Netscape Communications. OLE, COM, DCOM, Windows, Windows XP, .NET, Visual Studio, C#, and J# are trademarks of Microsoft Corporation. Lotus Notes is a trademark of Lotus Computing Corporation. Purify is a trademark of Highland Software, Inc. Proliant is a trademark of Compaq Computers, Inc. VAXClusters, DEC MessageQ, and DECsafe Available Server Environment are trademarks of Digital Equipment Corporation. MQSeries and SP2 are trademarks of International Business Machines. PowerBuilder is a trademark of PowerSoft Corporation. Ethernet is a trademark of Xerox Corporation. Gryphon and WebSphere are trademarks of IBM. WebLogic is a trademark of BEA, Inc.

Among cloud computing products and tools mentioned here, Azure is a trademark of Microsoft Corporation, MapReduce, BigTable, GFS and Chubby are trademarks of Google, which also operates the GooglePlex, EC2 and AC3 are trademarks of Amazon.com, Zookeeper is a trademark of Yahoo!, WebSphere is a trademark of IBM, BitTorrent is both a name for a technology area and standard and for a product line by the BitTorrent Corporation, Hadoop is an open-source implementation of MapReduce.

Other products and services mentioned in this document are covered by the trademarks, service marks, or product names as designated by the companies that market those products. The author respectfully acknowledges any that may not have been included.

Contents