Executive's Guide to

Cloud Computing

Eric A. Marks

Bob Lozano

Executive's Guide to Cloud Computing

Executive's Guide to Cloud Computing

Eric A. Marks
Bob Lozano

WILEY

John Wiley & Sons, Inc.

Library of Congress Cataloging-in-Publication Data:

Marks, Eric A.
Executive's guide to cloud computing / Eric A. Marks, Bob Lozano.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-470-52172-4 (cloth)
1. Business enterprises—Computer networks—Management. 2. Information technology—Management. 3. Cloud computing. I. Lozano, Bob, 1957- II. Title.
HD30.37.M36427 2010
004.3'6—dc22 2010002002

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Eric Marks
For my wife, Diane, and my great children, Jonathan and Jessica. Thanks, as always, for enduring another business book project. Someday, I'll write one that you'll read and enjoy!

Bob Lozano
In deepest gratitude for my wife Carol and all who make up La Familia Lozano, including in a particular way Raul Jorge Lozano, mi padre who completed his journey more than 30 years ago, my nephew David and my father-in-law James Huckaba who both completed their journeys within the past year . . . words can never suffice.

Contents

Preface

What is cloud computing? Is this real, or simply another overwrought marketing phenomena, which the thoughtful person should best simply let run its course? Suppose it is real—how important is this, what does it mean to our organization, what should we do, and how should we do it?

These questions and more are on the minds, or should be on the minds, of senior executives, leaders of many kinds and at many levels, and clear-thinking leaders-in-the-making at a wide range of organizations around the world.

As with any other area in which there is rapid innovation—and cloud computing is certainly such an area—there are many competing voices with a wide range of views, which can seem to be little more than a discordant cacophony. Fortunately, there are some valuable lessons that have already been learned; fundamental technologies, operational models, and business processes that have already been developed; real possibilities that have already been seen; these realities simply should not—no, *must* not—be ignored.

With all this in mind we set out to provide some basic understanding, clear guidance about the realities of cloud computing: what it is, why it has happened, and what best to do about it.

The term *cloud computing* is of relatively recent vintage. In fact, it was as recent as April 2008 when the nascent cloud community was roiled by a short-lived U.S. trademark on the term itself. The trademark was wisely abandoned quickly by the firm that had originally obtained it, thereby giving name to something which the participants all knew had become very real—not all at once, but gradually, in the convergence of a number of technical, business, even cultural, and sociological developments.

Yet those who had been working on some of the key technical developments had known for some time–five years, in some cases more–that there was something real here, something almost difficult to comprehend in the disruptive potential on the business of computing, something enormously exciting in the nearly breathtaking potential impact on the organizations dependent upon, enabled by, and all too often constrained by the then-present economics and capabilities of traditional computing technologies.

These are indeed exciting times in the computing world—cloud computing is, in fact, a real nexus, a moment when the endeavor of utilizing computing fundamentally changes. We have been in the thick of these developments since 2001, and through a fortuitous confluence of events were brought together to write this book.

That is the need and our intent—what about the book itself?

In many ways this is really ''books within a book,'' and we believe a wide range of people with a wide range of backgrounds and interests will find it helpful.

The beginning of the book (Chapters 1 through 3) and the end (Chapters 8 and 9) are of general interest: While some technical perspective is inevitable, just skip whatever may be a bit too detailed. Take care to understand the main points, particularly of Chapters 1, 2, and 9. Chapters 4 through 6 will be most helpful for the more technology-savvy in a variety of roles, from strategic planner to IT professional. Chapter 7 falls somewhere in between, and should be read as your background suggests.

In any case, here are each of the chapters and a brief description:

Chapter 1, The Sound of Inevitability: This lays the historical context of the broad trends and developments that have led to cloud computing.

Chapter 2, Concepts, Terminology, and Standards: Names the basics, establishes a common language for what is what.

Chapter 3, Cloud Computing and Everything Else: More context, placing cloud computing in relation with everything from virtualization to service-oriented architecture (SOA).

Chapter 4, Strategic Implications of Cloud Computing: Why executives should care.

Chapter 5, Cloud Adoption Lifecycle: An adoption model for the enterprise, with special comments for the startup.

Chapter 6, Cloud Architecture, Modeling, and Design: Focus on creating cloud-enabled applications that work equally well on both private or public clouds; interoperable private and public clouds; and operational models that make use of the potential elasticity, scalability, reliability, and cost reductions.

Chapter 7, Where to Begin with Cloud Computing: Practical steps for planning, executing, and measuring incorporation of cloud computing in a wide variety of organizations.

Chapter 8, All Things Data: Explores how the inexorable drive toward ''big data'' is fundamentally changing nearly everything about how data is stored, found, and manipulated.

Chapter 9, Why Inevitability Is . . . Inevitable: The fundamental reasons why cloud computing is happening, will happen, and consequently is well worth understanding.

In addition there is a brief Appendix that describes the basic categories within the vendor community. Note that it is our intent to maintain a directory of sorts on www.execsguidetocloud.com with vendor descriptions, current cloud-related news, and so forth.

An effort like this does not happen without the help of many. To our customers who have continuously asked ''why?''; our friends, competitors, and erstwhile compatriots throughout the industry; our friends and colleagues at both Appistry and AgilePath who are turning these ideas into practical realities; our editors Sheck Cho and Stacey Rivera and the rest of the team at Wiley; and of course to our families whose contributions are both sublime and essential; to all we acknowledge deep appreciation and offer our thanks for all that you have done to support, challenge, and call us to do better.

It is our sincere hope that this volume will help you gain a deeper understanding of what cloud computing is, why it is, and what you may reasonably do to make good use of what is a truly historic opportunity.

Bob Lozano
www.thoughtsoncomputing.com
www.execsguidetocloud.com
boblozano (twitter)

Eric Marks
emarks@agile-path.com
www.execsguidetocloud.com
ericmarks (twitter)

CHAPTER 1

The Sound of Inevitability

There have been very few fundamental changes in computing.

On the surface, that may sound like the statement of a madman, or perhaps at least someone from an alternate universe. Nonetheless, it is true.

Sure there have been, are, and will likely continue to be a nearly incomprehensible fire hose of particular changes, some rather flashy in and of themselves. Simple things like pocket-sized flash drives that store more than the corporate mainframes of 30 years ago, or perhaps ubiquitous mobile devices for everything from the mundanely practical—e-mail, calendars, and contacts—to the cheerfully sublime. Much more complex developments such as the open source movement; the advent of relational databases; and the rise (and fall) of whole operating systems and their surrounding ecosystems, even those whose perpetual dominance once seemed assured (how many desktop machines are running CP/M these days?). These have come and gone, perhaps lingering in some niche, forgotten by all but a few fanatical devotees.

But truly fundamental change—the tectonic shift that literally changes our landscape—happens only once in a long while, perhaps every ten or more years, even in the computing business. Fundamental change of this magnitude requires a number of smaller innovations to pile up until a true nexus is reached, and we all start marching down a different road.

Of course, as historians are fond of lecturing the rest of us mere mortals, these sort of fundamental changes are nearly impossible to recognize while we are in the middle of them, even as they loom imminently.

When researchers at the University of Pennsylvania were feverishly working on ENIAC—generally recognized as the first programmable, general-purpose electronic computer—as the future of the world hung in the balance in the midst of World War II, do you think they envisioned computers embedded in nearly everything, from greeting cards to automobiles, from microwaves to MRIs? When researchers at the University of California, Los Angeles, and elsewhere in the midst of the Cold War strove to make computer networks more resilient in the face of nuclear attack,[1] do you think any of them envisioned the Internet as we see it today? Likewise, when Tim Berners-Lee and other researchers at CERN were trying to come up with an easy way to create and display content over this new, literally nuclear-grade network, do you think they envisioned the impact on everyday life (both personal and professional) their new creation would have, or even the simple breadth and depth of stuff—from the sublime to the silly—that would be available on this new, supercharged ''Internet'' ? One estimate is that there are more than 500 exabytes—that's 500 *billion* gigabytes—in this ''digital universe,'' and that this will double every 18 months.[2]

The simple truth is that very few, if any, of the people involved in these developments had much of an idea of the consequences of their creations, of the impact on our personal lives, our culture, even the society in which we live—from how we interact with our families to how we conduct business.

Whether you are ''technologically modest,'' or are either by age or temperament not ashamed to let it be known, at least in certain circles, that you are a bit of a geek . . . either way, it is pretty much a given that developments in computing are having a big impact on our society, and more to the point, an even bigger impact on how we conduct our business.

And bigger changes—tectonic shift–scale changes—will have at least commensurate impact on our lives in every dimension, including the fields of commerce. One example, perhaps a seemingly simple one, yet central to many of the changes now underway, will suffice to illustrate this point.

Consider for a moment newspapers. We now face the very real prospect—actually the near-certainty—of at least one (and probably many) major metropolitan area in the United States without a traditional (local, general purpose, print, widely circulated) newspaper. While this eventuality may be stayed—perhaps for quite some time—via government intervention, the fact that this will eventually occur is not in doubt. In a culture still echoing with such reporteresque icons as Clark Kent, or at least the more prosaic Bernstein and Woodward, this was once unthinkable. Now it is simply inevitable.

There was a time when the technology of newspapers—cheap newsprint (paper), high volume printing presses, delivery networks including everything from trucks to kids on bicycles—was the only reasonable means for mass distribution of information. In fact, with help from some of the newer technologies there was even a new national newspaper (*USA Today*) founded in the United States as late as 1982. But with the advent of alternative delivery channels—first radio, then broadcast cable, and satellite television—increasing amounts of pressure were put on the newspapers.

The immediacy of the newer channels led to the widespread death of afternoon newspapers in most markets; anything delivered to the dinner table in a physical paper was hopelessly out of date with the evening news on television or radio. The morning papers had the advantage of broad coverage collected while most people slept, and as a result have held on longer.

However, at the same time intrinsic limitations of the newer technologies made them better for certain types of information, though not as useful for others. For example, a two-minute video from a war zone could convey the brutal reality of combat far more effectively than reams of newsprint, but did little to describe the complex strategic elements—political, economic, cultural—of the conflict itself. As a result, a certain stasis had been reached in which newspapers carved out what appeared to be a sustainable role in the delivery of news.

Then came the Internet.

In particular, the effectively free and ubiquitous—and yes, near-instantaneous—delivery of all sorts of information mortally wounded the newspaper business. As the first round of the web ecosystem grew, the only remaining stronghold of the traditional newspapers—their ad-based revenue model—was made largely irrelevant. eBay, Craigslist, and freecycle (among others) replaced the classifieds, and online ads took out most of what was left.

Some newspapers will undoubtedly manage the transition in some manner or another, perhaps even emerging as something fairly recognizable—particularly national/international properties such as the *Wall Street Journal* and the previously mentioned *USA Today*—and perhaps even financially sound.

But those that do will likely largely do so without their original distribution technologies, and more important, many will not make the transition at all.

All of this upheaval in news delivery—the enormous changes that have already occurred and that which is yet to come—have been enabled by developments in computing technologies, with the widespread adoption of everything from the Internet to the iPhone. It is probably worth remembering that all of this has occurred largely without cloud computing, and as a result we are probably less than 10% of the way through this transition in news delivery, and this is only one industry. One industry, one example, with entire economies yet to transform.

Even so, some things have not changed much, even in the delivery of news. The computing infrastructures range from the stodgy (server, even mainframe-based systems within many newspapers) to circa-2009 state of the art (which we might as well start referring to as ''legacy web,'' web 2.0, old-school web, something like that). By and large these systems still cost too much to acquire, do not adapt to changes in demand nearly easily enough, are not reliable enough, and remain way too complex and costly to operate. Even the few systems that do not suffer from all of these problems are not ideal, to say the least: Some are proprietary, and most are either too complex to create new application software, or simply do not scale well enough, at least for the sort of software that researchers are hard at work developing. In particular, with the first generation of electronic news infrastructures focused on just *delivering* the news, the next generation will be focused on sifting through all of that content, looking for just the right stuff.

All of that sifting and sorting and searching will take orders of magnitude more computing capacity than we have anywhere today. How will we pay for hundreds and thousands, perhaps even tens of thousands *times* more servers and storage than we have today—almost unimaginable quantities of computing? How will we operate them? Write new software for them? It is fair to wonder how we will even power all that gear. Assuming that all of these concerns are resolved, then, we will face a larger question still, one which we presume has many answers: What sort of business models are enabled by all of this, and how do we get there?

Before we leave this example, it is probably worth considering our present circumstances just a bit more. In particular, most of the history of both economics and engineering can be understood by thinking about managing *scarcity*. In other words, how do I get the most done with the least stuff, or within certain limits? For example, that underlying drive to dealing with scarcity, at its core, drives the startup team to work harder and pay less, the Fortune 500 enterprise to optimize manufacturing processes, and entire nations to set energy policies. Allocating scarcity is just Economics 101. Of course, it is also Engineering 101. Dealing with scarcity causes communications engineers to develop better video compression schemes, improve CPU designs to get more done in the same amount of time, and even rethink server packaging to reduce power consumption and labor costs.

While scarcity may be the nemesis of some, it is quite literally a prime mover behind the developments that have together come to be known as cloud computing. What does this mean, and how can it be possible?

A Persistent Vision

Better, faster, cheaper is often heard in technology circles. More than a policy, more than a philosophy, this is literally a way of life within technology communities. In an ideal world imagine that:

Computing—computation, storage, communication—is relatively free, scales up or down as needed, scales as much as needed, operates itself, and always works.

To one degree or another, this is the persistent vision that drives many of those who are developing cloud computing. Is all of this presently possible? Of course not; yet we are inexorably on this path.

Achieving this vision is, of course, a complex endeavor with far more to it than may meet the eye at first glance. That is why there is the rest of this book, for starters!

Before we go further let us elaborate a bit on the dimensions of this vision.

Engineers and mathematicians talk about something being ''within epsilon of zero.'' This is a term that comes from calculus. It simply means the process of approaching a particular limit, from wherever you started to the limit itself. In the case of the cost of computing infrastructure, that limit is zero. For most of computing history the costs of infrastructure have dominated decisions about what to deploy when: How much will those servers cost? How about that storage farm? That network? Now, however, we can start thinking about those costs being ‘‘within epsilon of zero’’; that is, over time the computing infrastructure comes closer and closer to being free. That leaves other costs as the new, more significant considerations—software licensing, data acquisition, for just two examples—and this will be examined more closely later in the book.

A Little History

In one sense the evolution of computing has been one long blur, with change piling on change, products that are ‘‘long in the tooth’’ in less than a year and virtually classic soon after, and with new concepts—Moore’s Law, for example—created simply so that we can describe, understand, and effectively institutionalize this relentless rate of change.

But there are times when these changes pile up in such number, in particular combinations of new capabilities and logical consequences, that the whole industry does head off in a new direction—when the very conversations, the underlying concepts, even the possibilities themselves change.

To help understand the import of our current transition into a computing world dominated by cloud computing, think a bit about where we have been, where we are now (at least just slightly before exactly right now), and both how and why we have travelled these paths. While there are clearly many ways that the history of computing can be written, this one will only focus on the big changes—the nexi[3] themselves—where the very possibilities change.

Three Ages of Computing

While there many ways to get a handle on the evolution of computing, in order to gain an initial understanding just where cloud computing fits, of just how significant and, yes, disruptive it is and will be, it is sufficient to consider the broad sweep of computing history.

First Age

Think about the role of computing within the typical organization prior to the widespread adoption of the Internet. The focus was on automating particular operations, creating supporting business processes, and of course, always improving efficiency.

Notice that the focus was *within* individual organizations, by and large. Yes there were purpose-built networks for interacting between organizations, some of them even fairly large and important (stock trading and manufacturer-specific EDI [electronic data interchange] networks are two notable examples), and even for certain organizations to interact with their customers (e.g., credit card authorization networks), but each of these tended to have a very specific, rather narrow focus. Even more important, these examples were relatively few and far between, and very difficult to achieve.

This was the first age of computing, in which organizations looked internally for the big wins. For the most part the edges of each organization remained the same as they had always been.

At the beginning of the first age the focus was on big infrastructure—mainframes, big point-to-point networks, centralized databases, and big batch jobs. Toward the end, terminals evolved into personal computers, networks went from hierarchical (with the mainframes at the center of each network) to decentralized, with a broader, generally more numerous collection of servers and storage scattered throughout an organization. While batch work still existed, many programs became interactive through this first age, eventually gaining much more visual interfaces along the way.

Infrastructure tended to be associated with particular applications—a practice since pejoratively known as ''application silos''—and important applications generally demanded enterprise-grade (read: expensive) infrastructure—mainframes or big servers, and so forth.

Application architectures tended to follow the same evolutionary path, with earlier applications being generally centralized, large and heavy, while client-server and distributed application architectures became mainstream toward the end.

This period also saw the rise of databases, along with the beginnings of specialized storage infrastructure upon which those databases relied.

Technologies such as parallel computing, artificial intelligence, and even semantic processing remained exotic tools that were employed in only the most demanding problems, where ''cost was no object'' (at least in theory), where the goal was simply to solve ever-bigger, ever-thornier problems—places like the nuclear weapons laboratories, national intelligence agencies, scientific research institutions, and the like.

Despite the rapid, consistent improvements in individual hardware and software technologies throughout this period, the limitations and complaints remained nearly constant. In particular, no matter how much was poured into the IT budget, the foul nemesis of ''application backlog'' was heard in the hallways of nearly every enterprise. Who did not constantly complain about how much IT was costing?

Still, it was at least (generally speaking) possible to automate crucial operations within a company, and as a result overall corporate efficiency steadily increased. More autos were made with less labor, more packages delivered with the same number of employees, higher revenues per store per employee, and so forth.

This period covered about four decades, from the roots of enterprise computing in the 1950s until the rise of the Internet in the mid-1990s. As with all major shifts in a society, its culture and technology, the roots of the end of the first age of computing were sown years before the second age began.

Second Age

The second age of computing is really the story of the rise of the Internet—Sun, Cisco, Mosaic (which became Netscape), web 1.0, eBay, Yahoo, baby.com, and the first Internet Bubble—all of it, good and bad, all of the tumultuous commotion of the first Internet land rush.

While many advances contributed to the beginning of the second age, the two most crucial were the development of the Internet itself, and the development and near-ubiquity of easy-to-use, visually attractive devices that could be used by nearly everyone.

The story of the development of the Internet is well known[4]—starting from a research question (Can we build a more resilient network, one that can survive a nuclear attack?), to a more loosely coupled set of higher level communications protocols (e.g., ftp for file transfers, smtp for e-mail, http for web content) built on top of this newly resilient foundation, then to a whole ecosystem of new software. From browsers to web servers, among many others, the Internet quickly went from ‘‘who cares?’’ to ‘‘must have!’’. By the early 1990s this new, sort of crazy idea began to dominate even mainstream business thought, to the point that normally sane, rational people predicted such improbably outcomes as the elimination of all brick-and-mortar stores, the irrelevance of a nation’s manufacturing base, and in some cases the irrelevance of nations themselves.

This in turn led to truly historic business hysteria: the Internet Bubble. (Truth be told, if not for macro-level economic problems that started in late 2008 the onset of cloud computing may have triggered Internet Bubble 2.0.)

But as the dust settled and all calmed down, it was clear that the world had shifted. Any enterprise intending to prosper now had to consider how best to reach their customers and their ecosystem of suppliers, and where to look for their newest competitors, all in the face of the newest reality—ubiquitous connectivity.

Likewise, the ubiquity of visually rich devices—at first stationary, then evolving to include the ‘‘handheld slabs of glass’’ (iPhone, android phones, Palm pre, and their successors) made it possible for the non-geek to care. While command lines and text terminals were enough for many of the early adopters, the simple reality is that audience is, by definition, limited.

There were people—including one of the authors—who went from cards, to command line, to modern bit-mapped displays (along with a mouse, laser printer, and local area network, all part of the experimental Alto workstations from Xerox PARC[5]), all well within the span of a single year—1979. At the beginning of that year most work was done on a mainframe via cards, printers, and batch jobs; halfway through 1979 work moved to interactive command-line access via dumb terminals; and by the end of the year you could sit in front of a Xerox Altos, mesmerized by mice, bit-mapped displays, and early networked games (Mazewars[6] being a great example).

While both of these trace their earliest roots—at least in forms that we would largely recognize today—to the mid-1970s, they each took 15 to 20 years to gestate sufficiently to have broad impact.

Overall, the biggest technical contribution of the second age was perhaps the network itself. Forced to deal with the possibility of massive network failures caused by a nuclear attack, researchers endowed their invention with the ability to self-organize, to seek out alternate routes for traffic, to adapt to all sorts of unforeseen circumstances.

In doing so (perhaps with only partial intent) these researchers removed the single point of failure that was typical of mainframe-inspired networks: and as a consequence in one fell swoop they removed the biggest technological barrier to scaling–the mainframe-centric network itself. Even more telling, foreshadowing changes that would usher in the third age-when they enabled the networks to take care of themselves-these researchers also removed the biggest obstacle to growth—they made these new networks *much* easier to operate.

It is hard to overestimate the importance of two fundamental realities: (1) with the Internet it was now true that everyone was connected to everyone else, anytime, anywhere; and (2) with the ubiquity of visually attractive devices, the data and services available over that pervasive network could actually be used by mere mortals.

Typical technologies included the J2EE application servers (often in clusters) along with relational databases, themselves often in clusters. Developers and researchers everywhere strove to stretch, push, pull, morph—everything but blowing them up and starting over—to make these application architectures more flexible, scalable, more resilient to failure, and so forth, but were mostly unsuccessful, or at least not successful enough.

There were plenty of innovations in software architectures, ranging from improved data techniques to the first forays into what became service-oriented architectures in the early part of the new millennia.

But what had not changed? Far too much remained as it always had, as things turned out. For starters, infrastructure remained expensive, chunky, siloed, and by modern standards phenomenally overengineered (after all, the infrastructure really should not fail), and consequently even more expensive. Great strides were being made in distributed software architectures, but (outside of the foundational TCP/IP networks themselves) most applications and infrastructure software remained difficult to configure, complex to create, and brittle when faced with failure. As a result, operations remained enormously difficult and therefore both costly and error prone, which in the final analysis was the cruelest constant reality of all.

Before we continue in this narrative, let us take a step back to consider two more constants in computing—the drive for ever-increasing scale and the drive for ever-lower expenditures (i.e., the ''drive for cheap'').

Drive for Scale Remember back to the middle of the first age, in the 1970s and 1980s—most computing was done on relatively mundane, large-scale individual computers, or perhaps in small clusters of relatively big machines. Even then, for the researchers, scientists, or perhaps intelligence agencies who were simply trying to solve the biggest problems possible, this was never enough; for that matter, nothing was ever enough, no matter how big and fast. Those folks were the ones who were exploring the edges of parallel computing and distributed architectures, who were thinking of highly pipelined supercomputers and vector processors.

Yet in the mid-1980s another thread of investigation took root—inspired by biological systems themselves—which started by combining large numbers of relatively slow computers, sometimes loosely coupled via a local area network (these came to be often known as grids) and sometimes linked internally via specialized connections (such as the exotic Connection Machine 1, produced by Thinking Machines, Inc., which was the effort to commercialize the doctoral work of Daniel Hillis). In all cases these alternative architectures were difficult to develop software for, cranky to operate, and enormously expensive. Even though most of those efforts eventually evaporated, they did at least make one very important contribution: They showed that it was indeed possible, particularly for certain applications, to build very large computing facilities out of very modest components.

This drive for scale went mainstream along with the Internet. This was true in many dimensions, but for one easy example just think of the indexing problem itself—whereas an early (circa 1994) Yahoo index might have had less than a hundred, or at most a few hundred entries, and could be manually created, by the beginning of 1995 the number of web sites was doubling every 53 days[7] and was passing anyone's ability to manually index. This growth then created the need for computing infrastructures that could scale at the same rates or faster, as well as application and data storage architectures that could also scale apace.

Yet there was one fly in the ointment that occurred about this same time—the silicon companies (Intel, competitors, and friends) began to reach their practical limit for scaling individual execution units (which came to be known as ''cores''). In fact, this problem had been looming for some time, but the processor designers tended to solve the problem the way they had always done: Throw more hardware at it and hope it would go away. In late 2004 Intel announced that they were largely abandoning their push to increase the ''clock speed'' of individual processing elements, and going forward would instead be, increasing the number of individual processing units (or cores). While, at least in theory, this drive for increased core counts can deliver the same raw computing capacity, in practice it is much more difficult to write application software that can make use of all of these cores.

This is, in essence, the ''parallelization problem,'' which in many ways is the same no matter whether you are writing software for multiple cores within a single piece of silicon, multiple cores on multiple processors within a single computing system, or multiple cores on multiple processors on multiple computing systems within a single grid/cluster/fabric/cloud.

Sound complex? To be honest, it is—successfully writing a parallelizable application can be enormously complex, difficult to do well, even more difficult to do reliably, and more difficult still to make it also easy to operate. In other words, the silicon and systems designers had punted, shifting the burden for scaling to the application software and operational communities.

Drive for Cheap Of course one drive that remains true in every age and in every domain is the drive to reduce costs—cost to acquire, cost to deploy, cost to operate, cost here, cost there, cost anywhere—just reduce them all.

In the midst of the rubble of the first Internet Bubble (bursting), many different groups began to wonder just how to make use of these increasingly capable commodity computers for problems that we really cared about—mission-critical problems, the ones that ''absolutely, positively, have to work.''[8]

For example, the roots of Appistry (a company founded by one of the authors) lie in just such a question. When building a digital recording studio out of purely commodity parts (no label, cheapest fastest stuff that money could buy), after running benchmarks the obvious question came up: Why are we not using cheap stuff like this (meaning the plain label, pure commodity computing parts) for problems that ‘‘we really care about’’?

The answers to that question—how to ensure that commodity infrastructure could be ultimately reliable, easy to operate, easy to bring software into and so on—led to multiple patents, products, and companies, and is a question whose answers are definitely worthwhile.

The economics of utilizing commodity components are compelling, if—and only if—you can safely answer those key questions. The economies of scale with commodity infrastructure, such as general-purpose processors, are simply overwhelming when compared to specialty designs. It is common for a collection of commodity computers to deliver the same capacity for less than 10% of the cost—sometimes far less than 10%—of enterprise-grade servers and mainframes.

It is no longer a question of ‘‘is this possible,’’ but rather ‘‘how, when, and where.’’

That same question—How can we use commodity infrastructure for problems that we care about?—is being asked and answered in various ways by forward-thinking technologists and executives everywhere in the relentless pursuit for ‘‘cheaper, faster, better,’’ and is integral in the transitions to cloud.

Third Age

Now let us resume our narrative. Early in the second age Yahoo had made a name for itself by ‘‘indexing the Internet,’’ which for some time was mostly manually done. While this was sufficient for a while, it soon became apparent that manually built indices could never keep up with the growth of the Internet itself.

Several other indexing efforts began, including AltaVista, Google, and others, but it was Google that brought everything together. While a full understanding of why Google became so dominant–at least as of this writing-is beyond the scope of this book, several key factors can be easily understood.

- First, the collection of data about the current state of the Internet, and the processing of that data had to be as absolutely automated as possible.

- In order to save as much money as possible, the infrastructure would be constructed out of commodity components, out of ‘‘cheap stuff that breaks.’’
- Data storage needed to be done in a simple, yet fairly reliable manner to facilitate scaling (the Google File System, or GFS—notice the lack of a traditional database, but more on that later).
- New types of application development architecture(s) would be required, which came to include the so-called map-reduce family (which inspired open source descendants such as Hadoop) among others.
- Operations needed to be as automatic and dependable as possible.
- Outages in the application were tolerable; after all this was search, and who would miss a few results if an outage occurred?

So almost before anyone really knew what was happening, in order to scale a basic search facility and do so cheaply, Google had created much of what we could probably first recognize as a cloud.

Another interesting case is Amazon. In the first six or seven years Amazon largely built its computing infrastructure the traditional way, out of big, heavy servers, with traditional relational databases scattered liberally throughout. That was fine in the early days, and definitely fine during the first couple of years after the Internet Bubble burst (particularly since much high-end hardware could be had for pennies on the dollar after the first bubble), but as commerce on the Internet began to gain some real momentum it became abundantly clear that the Amazon computing architecture(s) had to change.

At the same time, in order to build customer and vendor stickiness Amazon had begun exposing individual services, even select customer data as callable services—one of the key application lessons that is leading to the third age—and so had accelerated decomposing many of their applications into dozens, or sometimes hundreds, of individually callable services.

About that time (2001–2003) Amazon began to adopt many of the same principles as Google had done early on, but then they took things a step further. Instead of simply offering entire services such as search, e-mail, maps, photo, and so forth with various