Building Super Scalable Systems: Blade Runner Meets Autonomic Computing in the Ambient Cloud

Wednesday, December 16, 2009 at 9:28AM

Todd Hoff in ambient

"But it is not complicated. [There's] just a lot of it."
-- Richard Feynman on how the immense variety of the world arises from simple rules.

Contents:

We have not yet begun to scale. The world is still fundamentally disconnected and for all our wisdom we are still in the earliest days of learning how to build truly large planet-scaling applications.

Today 350 million users on Facebook is a lot of users and five million followers on Twitter is a lot of followers. This may seem like a lot now, but consider we have no planet wide applications yet. None.

Tomorrow the numbers foreshadow a new Cambrian explosion of connectivity that will look as different as the image of a bare lifeless earth looks to us today. We

will have 10 billion people, we will have trillions of things, and we will have a great multitude of social networks densely interconnecting all these people to people, things to things, and people to things.

How can we possibly build planet scalable systems to handle this massive growth if building much smaller applications currently stresses architectural best practices past breaking? We can't. We aren't anywhere close to building applications at this scale, except for perhaps Google and a few others, and there's no way you and I can reproduce what they are doing. Companies are scrambling to raise hundreds of millions of dollars in order to build even more datacenters. As the world becomes more and more global and more and more connected, handling the load may require building applications 4 or 5 orders of magnitude larger than any current system. The cost for an infrastructure capable of supporting planet-scale applications could be in the 10 trillion dollar range (very roughly estimated at $100 million a data center times 10K).

If you aren't Google, or a very few other companies, how can you possibly compete? For a glimmer of a possible direction that may not require a kingdom's worth of resources, please take a look at this short video:

What are we looking at here? This is F-Secure's time lapse video of computers all over the world, that have unknowingly become infected with botnet code (think

pod people), making IRC (Internet Relay Chat) channel joins. IRC is used as a command and control channel. Each white flash is a bot waiting for its next command to send spam, execute a distributed denial-of-service (DDoS) attack, or carry out some other nefarious deed.

Notice how global, how plentiful, and how fast the flashes flicker. It looks as if the whole world is afire, every ember burning with computation. That's a planet full of compute power. With millions of available nodes botnets wield more collective processing power than the world's top 10 supercomputers.

Now imagine if all this power was available to your application. Take a moment... welcome back. Ironically, you and I don't have millions of nodes at our disposal (criminals always get the good stuff). No cloud currently supports the ability to spin up this many cores and they may never get to that scale, assuming you could afford it.

But even if you did have this power at your disposal, what could you do with it? Imagine trying to build a complex application like Facebook across such a hyper-distributed network. What will systems that scale to the entire world look like? Nobody really knows. That's not how we build things today, but it might be how we need to build them.

We can only imagine what the scalable architectures of tomorrow will look like. Well, let's imagine a bit. What if, in the future, planet-scalable architectures are built in an all pervasive Ambient Cloud that looks a little like a mashup of the wild west of botnets, 4th Generation Warfare politics, the systematic thinking of autonomous computing, the cooperative ethos of Open Source, and the spirit of NoSQL all stretched over a vast compute fabric weaved from smart phones, smart grids, smart everything, PCs, body area networks, public clouds, private clouds, fast cell networks, optical fiber, and WiFi? That's an idea we'll begin exploring in this article.

The word "ambient" means "completely enveloping," like "ambient sound" or "ambient temperature." So an Ambient Cloud is a massively distributed collection of compute resources constructed from wherever they can be found. These compute resources are growing at an exponential rate. The Ambient Cloud isn't a creature produced by a committee or a standards body. Nor is the Ambient Cloud a traditional billion dollar datacenter construction project. It is held together by

informal, tacit, and negotiated relationships. It is irregular in shape, connectivity, policy, cost, technology and capability. What binds this complex world together are applications, not platforms, not APIs, and not services. In the end what we care about is delivering and receiving value and value comes from working applications.

The implication can be drawn, by considering the scope and complexity of the challenge of building palanet-scaling applications, that the only independent platform most of us will have access to capable of hosting planet-scale applications is the Ambient Cloud. It forms a sort of digital potluck where everyone contributes memory, network, and other compute resources from whatever they happen to have available.

Clearly this isn't the clean green future conjured up by well organized diagrams of grids and cloud architectures and compatibility standards. Botnets are way too simple to form the architecture for a generalized application platform. But they do help light the way by clearly demonstrating the immense collective power of the combined might of computers.

There's a bit of chaos here, but maybe that's what the future requires in order to create planet-scale applications? That's why botnets may serve more as a technical and political inspiration for future applications, not a template. In a Worse is Better sort of way this may be the exact mix that is required for long term success.

The basic argument that we'll be exploring in this article goes like this:

1. As more of the world of people and things come on-line and become interconnected, the driving force for applications will be the need to acquire the expertise and resources necessary to planet-scale at an affordable cost.
2. We currently lack the general knowledge of how to build planet-scaling sites. Only a very few companies can afford the massive commitment of money, resources, and talent to build and maintain planet-scaling applications on a planet-scaling infrastructure.
3. Natural monopolies have formed on the web around this expertise. These monopolies will make it possible for applications to planet-scale by adopting their platform, causing developers to enter a sort of neo-feudal relationship with their platform provider.

4. Large pools of compute resources from datacenter and non-datacenter sources are being created at an exponential rate. These compute resources form an Ambient Cloud capable of acting as runtime environment for planet-scale applications.
5. What is needed is an open strategy that harnesses the power of the Ambient Cloud so developers can stay independent and make a living creating the planet-scale applications that will be needed in the future.
6. To make the diverse and the continuously changing set of compute resources in the Ambient Cloud available to applications in real-time, compute resources will be traded in ways similar to how automated trading works in financial markets. Applications will need to become black boxes (independent action, entering contracts, etc) that can consume market information and change their architectures in real-time in response. The drivers are cost and scale. Creating planet-scaling applications is so resource intensive, and web applications provide such a low profit margin, that applications will be forced to be opportunistic and make use of resources from wherever they can be found.
7. By applications becoming black boxes that operate in a global Ambient Cloud composed of wildly different legal and political situations, applications begin to form a separate class of entity, with their own need for economic and political representation. They in effect become virtual states.

Taken together this is a picture how independent developer and user communities could begin to develop and use planet scalable applications.

I wish I was able to see what the implementation of such a system will look like, but after a year of thinking about it I haven't seen far enough, so I finally decided to just write and publish this article and get it over with. Hopefully smarter folks can help figure out the rest.

Have We Reached the End of Scaling?

That's what I asked myself one day after noticing a bunch of "The End of" headlines. We've reached The End of History because the Western liberal democracy is the "end point of humanity's sociocultural evolution and the final form of human government." We've reached The End of Science because of the "fact that there aren't going to be any obvious, cataclysmic revolutions." We've even

reached The End of Theory because all answers can be found in the continuous stream of data we're collecting. And doesn't always seem like we're at The End of the World?

Motivated by the prospect of everything ending, I began to wonder: have we really reached The End of Scaling?

For a while I thought this might be true. The reason I thought the End of Scaling might be near is because of the slow down of potential articles at my HighScalability.com website, which audaciously promises to help people build bigger, faster, and more reliable websites. When I started the site there was a flurry of finding and documenting common solution to scaling problems. After a time there wasn't as much to write about. This isn't to say scaling is easy. Not at all, but the general outline of how to scale small to cloud based systems is more-or-less known. This also isn't to say there isn't anything to report. There is, but it's usually a refinement rather than something completely new.

Then I got to thinking, if I really don't think any of those other "end of" ideas are true, then we probably haven't really reached the end of scaling either. More than likely what we've reached is the end of my vision. Time to get my eyes checked.

With a pair of brand new specs it was clear we are really just getting started at this scalability game. I was way too focused on what was in front of me. Probably my prescription. What I needed to do was look ahead. What is ahead is foreshadowed by some of the difficulties Facebook has in scaling.

Fortunately Facebook is pretty open about their infrastructure, so we learn a lot from their experience. From their information it's clear that building social networks for hundreds of millions of people, who are not even densely connected yet, is very challenging. And they are one the most successful companies in the world at scaling.

Building social networks far harder than scaling typical applications because:

1. All data is active all the time.
2. It's hard to partition this sort of system because everyone is connected.
3. Everything must be kept in RAM cache so that the data can be accessed as fast

as possible.

This is a blueprint for the future. To make all this magic work Facebook spends hundreds of millions of dollars on datacenters, they have 30K+ machines, 300 million active users, 80 billion photos, they serve 600,000 photos a second, they have 28 terabytes of cache, they generate 25TB of logging data per day, and their caching tier services 120 million queries every second. Phew!

Now let us imagine what will happen in the future by thinking about what we can't do yet. No social networking site will support 7 billion friends. No social networking site supports updating presence to 7 billion friends. No site can handle a comment thread of 7 billion people. No photography site will allow 7 billion people to upload pictures. No lifestreaming site can wireless stream HD input from 7 billion eye glass mounted cams. No virtual world allows 7 billion people to roam free and unhindered by meat conventions. No music site allows 7 billion people listen to music at the same time.

In short, we don't have worldwide applications...yet. Even our most popular web apps are used by a relatively few number of users. As you can see from Facebook's experience, ramping up to this scale is something completely different than what we are capable of now. We don't have architectures to meet this scale of problems. A warmed over cloud won't work. A few monolithic datacenters scattered throughout the world won't work either.

Why might it matter? I hope it's not so the entire world can know how another interchangeable starlet is having plastic surgery, while in rehab, during an multiple adoption process. As a world we are facing worldwide problems that can only be solved with worldwide cooperation. Wouldn't it be something if we the people of the world, in order to form a more perfect world, could actually talk to each other without politicians and special interests getting in the way?

Radical? Massive cooperation works as a way to solve problems. What we need is a supporting technology, politics, and culture.

Imagine if Einstein or Mahatma Gandhi were still alive. It would seem plausible that 7 billion people might want to follow them, read about their fresh insights, and what they had for lunch. It seems reasonable that if there's some grand planet impacting

scheme to combat global warming that a worldwide poll of 7 billion people would be run. And if the newest boy robot band sensation released a new song, shouldn't 7 billion people know that immediately?

So planet sized systems are reasonable to think about, they just aren't even vaguely possible at the moment. And that's just considering systems of people only.

Now let's bring in smart sensors. Smart sensors are where everything in the world--books, cars, phones, pets, appliances, etc--will have an IP address and send everything else in the world information about their inner feelings. To software, smart sensors are just like people, only a bit more single minded.

Imagine knowing the instantaneous power usage of an entire city by following millions of power sensors. Imagine creating your own weather forecast by following billions of weather sensors. Imagine competing medical services following information about your blood pressure, which medicines you've taken, and your health care coverage status all in real-time. Imagine a general following every aspect of a battlefield.

Now we are not talking about a lousy 7 billion people anymore, we are talking about designing systems where trillions of entities follow each other, talk to each other, read each other, post to each other, often at prodigious data rates.

Adding another level of scale is the exponentially growing number of connections between entities. This is what often drags down social networking sites. Users with a large number of followers cause a lot work on every post. The social graph must be traversed at least once on every post to see who should get a post. Traversing very large graphs in real-time is still a ways off. And handling the work generated by each post can be overwhelming. Take Digg for example. happen. If the average Digg user has 100 followers that’s 300 million diggs day, 3,000 writes per second, 7GB of storage per day, and 5TB of data spread across 50 to 60 servers. Now imagine what happens with trillions of entities. Oh my!

We are talking about systems 4 or 5 magnitudes (10,000 to 100,000 times) larger than our largest systems today. A magnitude means 10 times larger. For an idea of how big things grow by powers of 10 take a look at this Cosmic Voyage video narrated by Morgan Freeman. It's impressive.

The numbers, the complexity, and the need are there. What we need next is way to build planet-scale applications.

Applications Become Black Boxes Using Markets to Scale and Control Costs

We tend to think compute of resources as residing primarily in datacenters. Given the fast pace of innovation we will likely see compute resources become pervasive. Some will reside in datacenters, but compute resources can be anywhere, not just in the datacenter, we'll actually see the bulk of compute resources live outside of datacenters in the future.

Given the diversity of compute resources it's reasonable to assume they won't be homogeneous or conform to a standard API. They will specialize by service. Programmers will have to use those specialized service interfaces to build applications that are adaptive enough to take advantage of whatever leverage they can find, whenever and wherever they can find it. Once found the application will have to reorganize on the fly to use whatever new resources it has found and let go of whatever resources it doesn't have access to anymore.

If, for example, high security is required for a certain operation then that computation will need to flow to a specialized security cloud. If memory has gone on auction and a good deal was negotiated then the software will have to adapt to take advantage. If the application has a large computation it needs to carry out then it will need to find and make use of the cheapest CPU units it can find at that time. If the latency on certain network routes has reached a threshold the the application must reconfigure itself to use a more reliable, lower latency setup. If a new cheap storage cloud has come on line then the calculation will need to be made if it's worth redirecting new storage to that site. If a new calendar service offers an advantage then move over to that. If a new Smart Meter service promises to be a little smarter then go with the higher IQ. If a new Personal Car Navigation Service offers better, safer routes for less money then redirect. And so on.

In short, it's a market driven approach, mediated by service APIs, controlled by applications. Currently this is not how the world works at all. Currently applications resemble a top down hierarchically driven economy. Applications are built for a

specific environment: platform, infrastructure, network, APIs, management, upgrade, job scheduling, queuing, backup, high availability, monitoring, billing, etc. Moving an application outside that relatively fixed relationship is very difficult and rarely done. For that reason talking about more fluid applications may seem a bit like crazy talk.

There are two driving forces that may make this idea approach sanity: scale and cost. Unifying these forces is the carefully chosen word "economy" used in the previous paragraph.

A line of thought I first began to consider seriously in the article Cloud Programming Directly Feeds Cost Allocation Back into Software Design, is that applications are now in the very beginning stages of becoming rational actors in an economy and a polity. The motivation for this evolution are related to scale and cost.

In the past applications were designed to make use of fixed cost resources (racks, servers, SANs, switches, energy etc) that were incrementally increased in largish allotments acquired through an onerous management processes. You capacity planned, bought resources, installed them, and that was that until you needed more. During the design phase the cost per operation wasn't a consideration because the resource pool was fixed. All that was needed was an architecture that scaled enough to meet requirements.

Opportunities for Loss in Algorithm Choices

This quaint and anachronistic static view of the world changes completely under cloud computing because of one innovation: variable cost usage-based pricing. In the cloud you pay for resources in proportion to how much you use them. Fixed cost assumptions and approaches no longer apply. The implications of this change are subtle and far reaching

Three types of cost overruns are common in the cloud: spikes, free riders, and algorithm choice.

1. **Spikes**. If your site experiences an extinction level event, like getting a link on the front page of Yahoo, then you will have to pay for that instantaneous

traffic surge. You pay even if you didn't want it, even if it doesn't bring in any revenue. A DDoS attack no longer has to bring your site down by making it too busy to handle legitimate traffic, all an attack has to do in the cloud is deplete your budget and you are undone. In the fixed cost scenario your site would crash, but your budget would be spared.

2. **Free riders**. The spike problem is easy enough to understand, but the free rider, death by a thousand cuts problem is less obvious. Web sites often give free accounts in order to attract users, hoping to convert them to paid users later. These free users cost because they use resources in the cloud. Your seed money could sprinkled all over a lot of free user accounts and if not enough account germinate, then you'll go hungry.
3. **Algorithm choice**. Everything costs, but not everything costs equally. Let's say your brilliant implementation idea for faster access trades disk space for CPU by using multiple indexes on a record. As the user data grows you notice an enormous increase in disk space usage because your platform vendor uses a lot more disk per index than expected and for ease of implementation reasons you ended up using more indexes than expected. In this scenario your budget may evaporate under a laser hot heat based on this one innocent design decision. Similar games can played for any other resource in question. Every algorithm and architecture decision opens your project up for failure by an unexpected black swan event.

Opportunities for Profit in Algorithm Choices

Programmers have barely begun to grok the incredible financial impact their architecture and algorithm choices will have both on managing black swans and maximizing profit.

The flip side of the unexpectedly high cost problem is the opportunity to increase profit by creating algorithms that minimize cost, time, etc. If energy costs become a large part of algorithms in the future, for example, then you may be able to make a lot of money by creating a clever algorithm that uses less energy and exploits lower energy resources. It's a totally different game now.

One way to play the game is to pick a platform, stick with it, and figure out how to make it work. On Amazon, for example, one design rule is to favor CPU heavy

algorithms as CPU in EC2 is relatively cheap. This approach won't work at all on Google App Engine because you are limited to 30 seconds of CPU processing at a time. Every platform will have idiosyncrasies like these, once you get used to them you can generally build what you need to build, although often with considerable pain.

The Rise of Ambient Cloud Enabled Markets

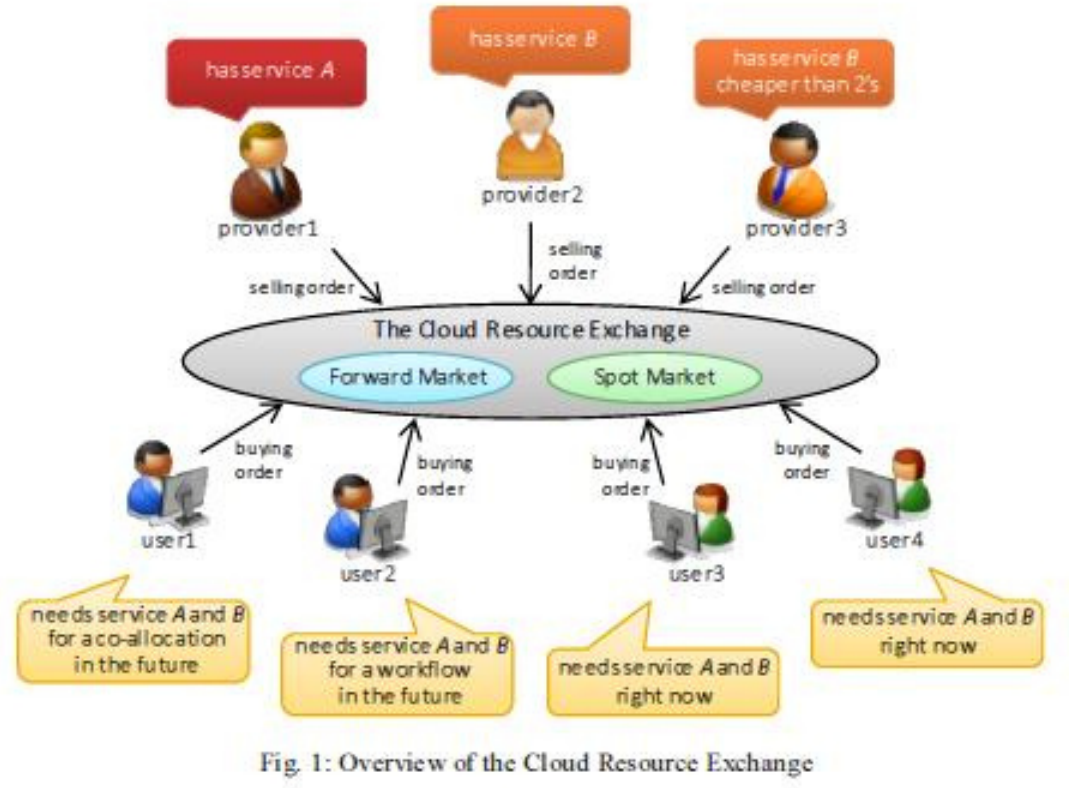


Fig. 1: Overview of the Cloud Resource Exchange

This image shows what a compute resource market might look like. It is taken from Market-based Resource Allocation for Distributed Computing.

The "pick a platform" play is really the only super scaling strategy there is at the moment. We are, however, starting to see multiple cloud options, we are starting to see multiple SaaS offerings for services like cloud storage, and we are still in just the very earliest stages of the Ambient Cloud. A rich and varied supply of compute resources on which to base a market is still in the future, but it is under construction.

The demand is there. The reason is the relationship between scale and cost. To see why consider the plight of the poor programmer. For our hypothetical planet-scaling

application, where are they going to get the resources from? Amazon, Facebook, Microsoft and Google will most likely have the resources, but at what cost? I assume like at present web applications will be low margin affairs, as most of us will be trying to hack together an acceptable living as homesteaders on the edges of the digital frontier. Developers will simply will not be able to afford resources from a tier1 supplier.

This is where the Ambient Cloud steps in as a shadowy bazaar-like resource market. A place where spare memory, CPU, and other goodies can be given away, sold, rented, or exchanged, like on eBay or at a garage sale. We already see this spirit in the Open Source world, charity efforts like Good Will, the explosion of different lending libraries, cooperative projects like SETI, the microlending movement, and the expansion of local currency efforts.

Each of us will have a surprisingly large pool of resources to put into the market: smart phones, smart houses, smart appliances, smart cars, smart prosthetics, PCs, laptops, energy, and so on. Step up a level and companies will have smart buildings, smart fleets, smart networks, smart sensors, datacenters, and so on to contribute into the pool. Step up another level and organizations and governments will be able to contribute smart grids, smart buildings, datacenters, and so on.

Given the size of our individual resource pools, basing a market system on both barter and money makes a lot of sense. Contributing X units of resources (CPU, memory, etc) to the Ambient Cloud would entitle you to Y units of resources on the Ambient Cloud. There's still plenty of room for specialty clouds, like say for low latency data grids. The advantage of such a wide spread resource pooling is that it will build a very resilient community. There will be no centralized point of attack or way to bring the whole system down.

We'll go into more detail on the amount of resources that could be available later, but consider in 5 years some estimate smart phones will have one petabyte of storage, which can store something like 3 billion photos. By 2020 it's estimated there 50 billion devices on the internet. It's also projected that the smart grid could be 1,000 times larger than the internet. And the rate of change in resource availability will be exponential. The amount of memory and CPU available will be staggering and increasing exponentially. Bandwidth on the other hand will not

follow this same growth curve, but it's not clear if that will be sufficient to ruin the party.

We've established both supply and a demand. Pervasive compute resource pools make up the supply. Programmers implementing applications make up the demand. In a typical market economy resources flow to the better deal. Clearly mechanisms will need to put in place to make and manage markets, something far beyond the current here's our API please code to it approach.

We see markets at work today with algorithmic trading in electronic financial markets. It appears no human even understands how these things work, yet they command huge portions of the economy. Go figure. The Financial Information eXchange (FIX) protocol is the messaging standard used to facilitate real-time electronic exchange. Something similar needs to work in the Ambient Cloud.

A major development helping legitimize this approach has been Amazon's introduction of a bidding system for EC2 Spot Instances. The logical implication is to extend this same mechanism everywhere to every type of resource.

This idea actually revolutionizes how applications function in a way similar to how electronic trading revolutionized the trading floor. Humans don't process trades anymore and they are even being squeezed out of even deciding what to trade. That responsibility has been shifted to a black box stuffed with trading models.

Similarly, applications will need to be black-boxed, parameterized, made to process trading information, make decisions, and automatically make trades in the application's architecture. Currently applications hardwire everything in advance. We know exactly how we'll cache records, store largish things, send email, store structured things, etc and the application directly reflects those choices.

Let's say a new specialized memory based cloud just came on-line, it is offering a promotional deal, 25% off for 10TB of replicated memcached storage. Our black box detects the deal, determines it saves us money, that it meets reliability parameters, and offers us even better geographic diversity in a region in which we are under served. The application executes a binding contract and then starts using this service for new data and moves old data over on a scheduled basis.

It's not difficult to think how such a maneuver could save hundreds of thousands of dollars and it all happened without human intervention. It will have to happen without human meddling for the same reason financial markets have done away with humans, we are too slow and too dumb to handle markets that move in fast and mysterious ways. Arbitrage is a game for the quick.

With a little imagination it's possible to see how this same process can work at every level of an application. A intermediary infrastructure service, for example, may aggregate memory from 100 million smart phones and make it available for object storage that will allow a 7 billion person friend list to be stored. A smart grid rents their backend sensor cluster for a map-reduce job that works very well on underpowered CPUs. A queue service offers a good price for reliable low latency queuing so some of your queue load is offloaded to that service. A new key-value service looks promising so you throw some traffic that way as a sort of service A/B test. A new MRI evaluation service opens in India and wins 10% of the business for your Offline Patient Data Mining service. Amazon Turk won a bid to tag a million pictures. The possibilities are endless.

Make no mistake, Google is following a similar strategy with their own infrastructure and with their own applications, they will just own all the parts end-to-end. The system Google is building is called Spanner. Spanner's goal is to support 10 million servers, 10^{13} directories, 10^{18} bytes of storage, spread across 100s to 1000s of locations around the world surviving 10^{9} client machines. Those are staggering, staggering numbers.

It's not just the number that are staggering. What is also impressive is how Spanner plans to go about its business. Spanner will automatically and dynamically place data and computation across the world. It will try to minimize latency and/or cost given bandwidth, packet loss, power, resource usage, failure modes, and user defined goals.

If you are not Google, especially if you are not Google, where will all the money come from? Where will the technology come from? Where will the expertise come from? Technologically speaking this is a moon shot and Google seems to be one of the only ones in the space race.

Automated systems like Spanner are really the only way to handle the scale and complexity of planet-scaling applications. Humans gotta go. The difference is Google will rely on an internal planned economy instead of a public market based mechanism.

This isn't SkyNet or any other kind of AI gibberish. It's just code. Clever code certainly, but not take-over-the-planet-and-kill-all-the-humans type code. Models will need to be built that encode different architectural trade offs. Applications will need to built to respond to the output of the models. The models will need to consider factors like: APIs, cost, power, latency, size, access patterns, packet loss, security, bandwidth, geography, regulations, reliability, algorithms, available alternatives, and so on. Different, but nothing approaching sentience.

Some of this is old, old stuff. The idea of software based broker services has been around for quite a while. At one time the OSI stack was going to lead to automated broker services. Then it was the CORBA Interface Definition Language (IDL). Then it was the Web Interface Definition Language (WIDL).

All were based on the idea that if you completely defined software services then programs will dynamically look each other up and simply plug and play. It didn't quite work out the way. What was missing was a reason. There wasn't a viable market, only a mechanism. What is different now is that we'll have both supply and demand. Creating black box applications will actually make sense.

Let's Welcome our Neo-Feudal Overlords

There's a pattern, already begun, that has accelerated by the need for applications to scale and increase complexity, the end result of which will be that applications give up their independence and enter a kind of feudal relationship with their platform provider.

To understand how this process works, like a glacier slowly and inevitably carving out a deep river valley, here's the type of question I get quite a lot:

> *I've learned PHP and MySQL and I've built a web app that I HOPE will receive traffic comparable to eBay's with a similar database structure. I see all these different opinions and different*

> *techniques and languages being recommended and it's so confusing. All I want is perhaps one book or one website that focuses on PHP and MySQL and building a large database web app like eBay's. Does something like this exist?*

I'm always at a loss for words with these questions. What can I possibly say? The gulf between a LAMP application and eBay is so huge that there's no way to even start explaining it. Yet, what they want is reasonable. They just want to make a site that scales to meet their dreams. This barrier will be reached more and more as people try to make a living out on the digital frontier. They won't be able to do it on their own. So I'm forced to recommend that they find the protection of a King and take a look at Google App Engine, EC2, or one of the other platforms. There's no practical alternative at the moment.

The King Owns the Level 3 Platform You Need

What people will be forced to find is what Marc Andreessen calls a Level 3 Platform. A platform *is a system that can be programmed and therefore customized by outside developers*. A Level 1 Platform is the API. Example: Twitter. Applications using Twitter's API are completely separate from Twitter yet use Twitter's data. A Level 2 Platfrom is a Plug-in API. Example: Facebook. Facebook allows applications to embed themselves into Facebook's UI, but the apps are still separate from Facebook. A Level 3 Platform is a Runtime Environment. Example: Salesforce, Google App Engine, Amazon, Azure, and Heroku. This next step is where application merge into the platform. Developer application code is uploaded and actually runs inside the platform.

As a programmer, leveraging a Level 3 Platform makes perfect sense. As the need to scale becomes greater and greater it becomes harder and harder to do, so we look for a little help from the very few companies that will have the infrastructure to make it happen. Quite understandable. When we can't do something, we naturally seek out a partner. We outsource it to those who promise to do it for us. Applications have started to become mashups where the hard parts are written in terms of a platform that does all those hard parts.

The relationship here is asymmetrical. Once adopted, a platform becomes hard to leave which adds a political dimension to the evolution of future application

architectures. This result may not be readily apparent because we are still so close to the era of small disconnected systems: application writers are becoming 'vassals' to their feudal platform 'lords', a kind of neo-feudalism that bares a striking relationship to the previous agriculture based feudalism.

This somewhat imprecise feudal analogy goes like this: the platform provider (the King) provides services, land, and protection to their platform users (vassals). The vassals pledge to use the King's resources to produce money (crops) using computers (serfs). In exchange, the King gets a slice of the production.

This sounds all melodramatic, but I'm just using feudalism as metaphor to make clear how the power relationships are aligned. As a developer you won't have much power. Not only are you bound by the platform and mashup APIs used to write applications, but you are also bound by a need to pick a partner in order to planet-scale.

I'm not even implying there is anything nefarious or even necessarily conscious about this process. It's simply a natural outgrowth of the forces of scale, complexity, and danger. While I don't think there's an evil intention involved, we should still be aware of where we might unthinkingly end up.

Datacenters are the New Land

In the feudal era having land meant everything. Only land owners could grow crops for food, for trade, and for money. Land was wealth. When the feudal era collapsed industry and trade became the new means for acquiring wealth. The nobility became land rich and money poor. It was more profitable to own shares in the East India Company than it was to be born an Earl. A radical transformation for culture that had lasted hundreds of years.

To stretch the analogy further, in terms of building applications we have been in the hunter gather era, we are entering the feudal era of great land owners, and we are contemplating what might constitute the next industrial revolution. Datacenters have become the equivalent of land, castles, and the new centers of wealth.

This is why we see a consistent stream of new datacenters being built. If you want to farm you need land. If you want to run code you need compute resources. Land in

the datacenter is made from compute resources: SANs, CPUs, switches, memory, networks, and so on. It's not just about hardware though, it's also about services like: elasticity, monitoring, DNS, failover, security, queuing, email, database, storage, geographically disperse operations, backups, documents, searching, and so on. Perfect soil in which to plant, tend, and grow applications.

Datacenters are the New Castles

The fact that the King gets a slice of production is obvious. But one of the key duties of the King in the feudal system is to take up the mantle of protector. It may not seem obvious how datacenter owners become our protector, but as attacks escalate this may become one the most coveted services of all. Datacenters are like castles spread out across the landscape, offering safety and control of their domain.

The digital world is vicious. A DDoS attack can take down even the largest of companies and no small land holder can even begin to defend themselves. But if you enter Google's domain Google will protect you. Google can stop a DDoS attack. They have the technology and the people to make the problem go away. You don't even have to think about it.

Here's my personal story of woe as an illustration. One day my email stopped. The inbox was as empty as a cubicle farm on Thanksgiving. Another day went by, no email. Then another day went by, still no email! Figuring I couldn't have become that unpopular, I contacted my email provider.

They informed me my email was shut down because there was a DDoS attack on my domain. Damn! Who would do such a thing and why? Their standard response is to wait it out. Attackers usually give up soon.

So we waited. We waited six days. Six days without mail and without a letup on the attackers. Persistent little black hats. It seems the resources dedicated to their assault were so inconsequential they that just left the attack going.

For my email provider that attack was anything but inconsequential. They kept my email turned off to protect their system. They simply didn't have the expertise, the equipment, or the resources to deal with these attacks.

It's a little like homesteaders on the frontier being attacked by marauders. On the frontier of the US when homesteaders came under attack and could no longer protect themselves, they had to move to the nearest fort for protection.

That's exactly what I did. I abandoned my log cabin and moved into fort Google for the duration. I made Google the email provider for my domain. As soon as I made the move email started flowing again. Yay, my digital corpse has been resuscitated! Google was not cowed by these attackers. Google took the attack, ate it up, and spit it back out without even blinking. My savior!

Meanwhile back at the frontier my previous email provider said the attack stopped after 10 days and I that could go back. Well, I didn't go back. Why would I? I need protection from a big strong Google with all the hardware and processes and experience. I needed the protection of the King and his castle. It really, really, really kills me to have to say that, but that's what a King can do for you.

The castle walls in the datacenter aren't made of stone, though there is usually a hardened security perimeter, they are more in the form of security appliances, DDoS monitoring systems, SSL appliances, deep packet inspection appliances, and constant operational vigilance. Soldiers are always walking the walls and guarding the gates.

Security isn't the only area that requires protection. How about massive power failures, disk failures and server failures? These are all areas where a large, complex, well tended infrastructure, like in a datacenter, provides protection.

As the world becomes more connected and dangers become more pronounced, these trends will only accelerate. In good times you can homestead in the wilderness, but when the bandits attack and you can no longer bring your goods to market, you seek the protection of the King.

The King has Subtle Forms of Persuasion

Originally I considered the driving forces of scale and complexity as sufficient to compel developers to end up in the arms of platform provider. It's a very organic and understandable process.

Lately there have been some darker ideas explaining why there will be a convergence at the datacenter. First is Google Makes A Bid To Control The Internet by Rob Diana. Second is Tim O'Reilly's The War for the Web. What are they afraid of?

Rob Diana's hints at Google's power to drive technology selections by pounding its big hammer: pagerank. The logic of his post goes:

1. Google is considering penalizing the pagerank of sites that aren't fast enough,
2. Google's search is the market leader, so Google is he who can not be ignored.
3. Google is making a series of initiatives (SPDY, Chrome, GO, DNS, URL Shortener) to increase speed.
4. The only plausible way a majority of sites to comply with the new speed requirements will be to adopt Google's stack.
5. Google's stack becomes the Internet for a large portion of users.

For my thesis it doesn't really matter if this process is part of some master plan or is derived from well meaning steps independently taken. By linking pagerank to performance and providing all the intermediary steps necessary to become performant, the result is likely to drive developers to a platform for refuge. Smaller hosts and less performant stacks (like LAMP) will be difficult to justify when pagerank, and thus revenues, are dependent on speed. Google is certainly not the only platform option, but people will go to a platform people as an easy prepackaged solution to a tough to solve problem.

Tim O'Reilly is his article talks about how natural monopolies have formed on the web and how companies may use monopoly power to reach into other areas:

> *It could be that everyone will figure out how to play nicely with each other, and we'll see a continuation of the interoperable web model we've enjoyed for the past two decades. But I'm betting that things are going to get ugly. We're heading into a war for control of the web. And in the end, it's more than that, it's a war against the web as an interoperable platform. Instead, we're facing the prospect of Facebook as the platform, Apple as the platform, Google as the platform, Amazon as the platform, where big companies slug it out until one is king of the hill. And*

> *it's time for developers to take a stand. If you don't want a repeat of the PC era, place your bets now on open systems. Don't wait till it's too late.*

The point here is as one commenter says, *certain websites have such a strong gravitational field they suck in everything around them and get ever larger*. Exciting capabilities like maps are a big platform adoption driver, but I think the biggest baddest black hole of them all will end up needing to planet-scale.

An article by ZDNet's Dion Hinchcliffe, Cloud computing and the return of the platform wars, seems to back up Tim's fears. Hinchcliffe thinks there will be a protracted platform war with a *fairly predictable and oft-repeated cycle of events for which a small number of large winners are likely to emerge victorious*. He has a

From http://blogs.zdnet.com/Hinchcliffe

cool diagram showing the process:

Hinchcliffe observes the results of this war will end in a reversal of recent trends: *The world of software has recently, at least up until now, been moving slowly and steadily towards an increasingly commoditized, virtualized, and open sourced future. Cloud computing in its present form does appear to herald a return to the classical days of big vendor computing and all the baggage (good and bad) that it implies along with some unique twists of its own*. Exactly the result Tim warns about.

How this cycle becomes important is summarized by the idea of path dependence:

the degree to which initial design choices condition later technological development. This is very real effect, otherwise we wouldn't still be using the deficient by design QWERTY keyboard. Once initial platform choices have been made, the shape of the future is largely determined, regardless of the quality of those early decisions or the quality of later alternatives.

I'm Not a Number, I'm a Free Man

The GooglePlex isn't the only King out there. Others are trying. We have FacebookPlex, AmazonPlex, MicrosoftPlex, and there will be others. As a serf, the key question for me going forward is: how can we create a FreePlex where a free people can carve out a patch of land and make a living?

The Economic Argument for the Ambient Cloud

Many won't find the neo-feudalism line of argument compelling. Their reasoning is that not every application needs to scale. The muscular forces of globalization and the proliferation of things and people into dense social networks will only drive a few specialized applications into the arms of a few dominant platform vendors.

And they are right. Not every application will need to plug into the planetary nervous system. But as applications become more and more mashup based, it will just inexorably and naturally happen. Some service you want use, or a service that uses another service, will require planet-scalability, so your application will need to planet-scale too. It's the same sort of process that will cause every toaster next year to speak Twitter.

So if we accept the general movement to planet-scale application, how will this impact the traditional hosting market? Aren't they at risk?

Many are trying wedge themselves in and grab left over bits of the pie. The problem is cloud hosting is a low margin business with an outrageously high capital cost structure. This is exactly the kind of competitive environment Amazon, with their vast expertise in low margin operations, thinks they are perfectly suited to dominate.

As a "little" to "middle" size vendor it will be very expensive to create your own cloud and compete with incumbent cloud providers who are already financing their

growth with user revenue. This is even with the current situation where building in the cloud is substantially more expensive than a bootstrapped build your own colocation scenario. As competition heats up down the road, companies like Amazon and Google are in a great position to simply use their efficiencies of scale and ride the cost curve down, down, down.

In the hosting market economics today we see a lot of white label hosting providers. A few companies provide a nuts and bolts hosting service and many other people resell those services as their own. There are also many other smaller companies who provide bare bones virtual server hosting. Currently there's enough margin to support these business models because super stud bootstrappers can build a low cost colocation site and profitably resell it.

Will the cloud eventually kill off this business model? It's hard to say of course. But there's certainly a big risk. Efficiencies of scale and the ability to lower margins and ride down the cost curve will make it harder and harder to compete on price, especially as the value-added services (security, load balancing, storage, databases, CDN, monitoring, etc.) offered by the cloud vendors follow a “lure and lock” strategy with developers.

Perhaps hosting providers can band together and create a virtual cloud from their collective colo sites. Maybe a reselling model will still work. Maybe a service and support contract model will work. And it's likely there will always be a market for scrappy competitors who can always just manage to undercut the traditional cloud vendors on price.

Maybe there's another way? Instead of playing the game of trying to compete by always being 20% cheaper, what is needed is a disruptive order of magnitude or more improvement. What I suggest is to shift to a completely different cost structure by building on the Ambient Cloud instead of the traditional cloud.

The idea is if it's hard to beat your competition on one playing field then it's better to shift to a completely different playing field where they are at a disadvantage. So instead of building infrastructure to compete with an already strong opponent using the same tools and infrastructure they use, why not spend all that time and effort leapfrogging ahead of them in a space they won't want to enter? That space is the Ambient Cloud.

What Will Kill the Cloud?

If datacenters are the new castles, then what will be the new gunpowder? As soon as gunpowder came on the scene, castles, which are defensive structures, quickly became the future's cold, drafty hotels. Gunpowder fueled cannon balls make short work of castle walls.

There's a long history of "gunpowder" type inventions in the tech industry. PCs took out the timeshare model. The cloud is taking out the PC model. There must be something that will take out the cloud.

Right now it's hard to believe the cloud will one day be no more. They seem so much the future, but something will transcend the cloud. We even have a law that says so: Bell's Law of Computer Classes which holds that *roughly every decade a new, lower priced computer class forms based on a new programming platform, network, and interface resulting in new usage and the establishment of a new industry.* A computer class in this context is defined as a *set of computers in a particular price range with unique or similar programming environments (e.g. Linux, OS/360, Palm, Symbian, Windows) that support a variety of applications that communicate with people and/or other systems.*

We've been through a few computer evolutions before. Here's a list:

1. Mainframes (1960s)
2. Minicomputers (1970s)
3. PCs and Local Area Networks (1980s)
4. Datacenter is the Computer (1990s)
5. Smartphones (2000s)
6. Wireless Sensor Networks (>2005)
7. Body Area Networks (> 2010). These are *dust sized chips with a relatively small numbers of transistors enable the creation of ubiquitous, radio networked, implantable, sensing platforms to be part of everything and everybody as a wireless sensor network class. Field Programmable Logic Array chips with 10s-100s of million cells exist as truly universal devices for building “anything”.*

The first part of this list may be somewhat familiar. Though hardly anyone living has seen a mainframe or minicomputer, they really did indeed exist, much like the dinosaur. Companies like IBM, Honeywell, HP and DEC dominated these eras.

After mainframes and minis came personal computers. Much like the production of affordable cars brought us the freedom of the open road, PCs freed us from centralized bureaucracies and allowed us to travel the Internets without restriction. Companies like IBM, Microsoft, Apple, and Dell dominated this era. This also happens to be the era we are just leaving. Please wave a hearty and warm good bye.

Fickle beings that we are, it turns out we want both the freedom of an open mobile road and centralized IT. PCs became too complex and way more powerful than even a hardened gamer can exploit. This has brought on the era of the cloud. The cloud treats large clusters of powerful networked computers like a single abstraction, thus the Datacenter is the Computer. Notice how clouds combine aspects of mainframes, minis, and networked PCs into a whole new thing. We've been entering the cloud era for a while, but they are really just starting to take off. Companies like Google, Facebook, and Amazon are dominating the cloud currently. Microsoft, IBM, and others are trying like heck not to get left behind.

At one point I thought the cloud was just a transitory phase until we took the next technological leap. This is still true in the far future, but for the long now I see the cloud not as a standalone concentration of dense resources, but as one part of a diffuse overlay constructed from the cloud and the next three generations of Bell's Classes: Smartphone, Wireless Sensor Network, and Body Area Network.

While we have cloud masters creating exquisite platforms for developers, we haven't had much progress on mastering smartphones, wireless sensor networks, and body area networks. Why this would be isn't hard to understand. Smart phones are just coming into their own as computer platforms. Wireless sensor networks hardly even exist yet. And body area networks don't exist yet at all.

There's also a capability problem. What would kill the cloud is to move the characteristics of the cloud outside the datacenter. Create super low latency, super high bandwidth networks, using super fast CPUs and super dense storage - that would be a cannon shot straight through the castle walls of the datacenter.

The likelihood of this happening is slim. Part of the plan, super fast CPUs and storage are well on their way. What we, in the US at least, won't have are wide spread super low latency and super high bandwidth connections. These exist between within datacenters, between datacenters, and as backhaul networks to datacenters, but on a point-to-point basis the US bandwidth picture is grim.

Nielsen's Law holds that Internet bandwidth grows at an annual rate of 50 percent, while capacity grows at 60 percent. A seemingly small difference, but over a 10-year time period this means computer power grows 100x, but bandwidth only grows at 57x. So network speeds won't grow as quickly as servers can be added. Ironically, this is the same problem chip designers are having with multi-core systems. Chip designs can add dozens of processor cores, but system bus speeds are not keeping pace so all that computational power goes under utilized.

What Bell's 3 Classes do provide in abundance is almost unimaginable parallelism. Fortunately for us the absolute driving force of scale is parallelism. So it seems what is needed to create an open, market driven Ambient Cloud is an approach that exploits massive parallelism, fast CPUs, large pools of fast storage, and a frustratingly slow and unreliable network. In later sections we'll see if this may be possible.

If this sounds impossible take a look at Futurist Ray Kurzweil's explanation of The Law of Accelerating Returns. The basic idea is that change happens much faster than we think: *An analysis of the history of technology shows that technological change is exponential, contrary to the common-sense "intuitive linear" view. So we won't experience 100 years of progress in the 21st century -- it will be more like 20,000 years of progress (at today's rate).*

The lesson is if you think all the stuff is really far off in the future, it's actually just around the corner.

The Amazing Collective Compute Power of the Ambient Cloud

Earlier we talked about how a single botnet could harness more compute power than our largest super computers. Well, that's just the start of it. The amount of computer

power available to the Ambient Cloud will be truly astounding.

2 Billion Personal Computers

The number of personal computers is still growing. By 2014 one estimate is there will be 2 billion PCs. That's a giant reservoir of power to exploit, especially considering these new boxes are stuffed with multiple powerful processors and gigabytes of memory.

7 Billion Smartphones

By now it's common wisdom smartphones are the computing platform of the future. It's plausible to assume the total number of mobile phones in use will roughly equal the number of people on earth. That's 7 billion smartphones.

Smartphones aren't just tiny little wannabe computers anymore either. They are real computers and are getting more capable all the time. The iPhone 3GS, for example, would have qualified as a supercomputer a few decades ago. It runs at 600 MHz, has 256MB of RAM, and 32 GB of storage. Not bad at all. In a few more iterations phones will be the new computer.

The iPhone is not unique. Android, Symbian, BlackBerry, and Palm Pre all going the same direction. Their computing capabilities will only increase as smartphones are fit with more processors and more graphics processors. Innovative browsers and operating systems are working on ways of harnessing all the power.

Tilera founder Anant Agarwal estimates by 2017 embedded processors could have 4,096 cores, server CPUs might have 512 cores and desktop chips could use 128 cores. Some disagree saying this too optimistic, but Agarwal maintains the number of cores will double every 18 months.

That's a lot of cores. That's a lot of compute power. That's a lot of opportunity.

It's not just cores that are on the rise. Memory, following an exponential growth curve, will be truly staggering. One Google exec estimates that in 12 years an iPod will be able to store all the video ever produced.

But all the compute power in the world is of little use if the cores can't talk to each other.

The Cell Network Bandwidth and Latency Question

Aren't cell phone networks slow and have high latencies? Currently cell networks aren't ideal. But the technology trend is towards much higher bandwidth cell connections and reasonable latencies.

Here's a rundown of some of the available bandwidths and latencies. For cell connections your milleage may vary considerably as cell performance changes on a cell-by-cell manner according to the individual site's local demographics, projected traffic demand and the target coverage area of the cell.

1. WiFi networks provide latencies on the order of 1 to 10 milliseconds at 1.9 Mbps (megabits per second).
2. Ping times on the EDGE network are reported to be in the 700 to 1500ms range at 200 Kbps to 400 Kbps (often much lower).
3. In New York HSPDA (High-Speed Downlink Packet Access, which is 3.5G type nework) has latencies between 100 to 250 milliseconds and 300KB/s of bandwidth. Note that's bytes, not bits.
4. Amazon has average latencies of 1ms across availability zones and between .215ms and .342ms within availability zones.
5. In 2012 AT&T plans to move to a 4G network based on LTE (Long Term Evolution). LTE is projected to offer 50 to 100 Mbps of downstream service at 10ms latency.
6. On a 1Gbps network a 1000 bit packet takes about 1 microsecond to transmit.
7. Within datacenters using high bandwidth 1-100Gbps interconnects the latency is less than < 1ms within a rack and less than 5ms across a datacenter. Between datacenters the bandwidth is far less at 10Mbps-1Gbps and latency is in the 100s of ms realm.

Current bandwidth rates and latencies for cell networks don't make them a sturdy a platform on which to build a cloud. Projected out they are looking pretty competitive, especially compared to the between datacenter numbers, which is the key competition for future cloud applications.

True HPC (high performance computing) low-latency-interconnect applications won't find a cell based cloud attractive at all. It's likely they will always host in specialty clouds. But for applications that can be designed to be highly parallel and deal with short latencies, cell based clouds look attractive. Starting 10,000 jobs in parallel and getting the answers in the 100ms range will work for a lot of apps as this is how applications are structured today. Specialty work can be directed to specialty clouds and the results merged in as needed.

Won't faster networking and more powerful processors use more power? Aye, there's the rub. But as we've seen with the new iPhone it's possible to deliver more power and a longer battery life with more efficient hardware and better software. Inductive chargers will also make it easier to continually charge devices. Nokia is working on wireless charging. And devices will start harvesting energy from the surroundings. So it looks like the revolution will be fully powered.

Smart Grid 1,000 Times Larger than the Internet?

Another potential source of distributed compute power are sensors on smart grids. Literally billions of dollars are being invested into developing a giant sensor grids to manage power. Other grids will be set up for water, climate, pollution, terrorist attacks, traffic, and virtually everything else you can think to measure and control.

There could be 50 billion devices on the internet. Others predict the smart grid could be 1,000 times larger than the Internet.

While these are certainly just educated guesstimates about the potential size of the smart grid, it's clear it forms another massive potential compute platform for the cloud.

33 Million Servers (and growing) in the Traditional Cloud

According to the IDC as of 2007 there were 30.3 million servers in the world. Is the number 50 million now? Will it be 100 million in 5 years? New datacenters continually come on-line, so there's no reason to expect the total to stop growing. The Ambient Cloud has full access to these servers as well as non-datacenter computer resources.

Body Area Networks

Body Area Networks are the last of Bell's set of computer classes and are probably the least familiar of the group. BANs are *sensor networks in and around the human body*. They are so strange sounding you may even be skeptical that they exist, but they are real. There's an BAN IEEE Task Group and there's even a cool BAN conferences in Greece. You can't get much realer than that.

Clearly this technology has obvious health and medical uses, and it may also figure into consumer and personal entertainment.

Where do BANs fit into the Ambient Cloud? There are billions of humans and with multiple processors per human and communication network, it will be possible to integrate another huge pool of compute resources into the larger grid.

What if smartphones become the cloud?

Let's compare the collective power of PCs + smartphones + smart grid + smart everything + BANs with the traditional cloud: it's trillions against many 10s of millions. This number absolutely dwarfs the capacity of the traditional cloud.

One author wrote we'll be all set when smartphones can finally sync to the cloud. What if that's backwards? What if instead smartphones become the cloud?

Texai

It's really hard to get feel for what having all this distributed power means. As a small example take a look at Texai, an open source project to create artificial intelligence. It estimates that if one hundred thousand volunteers and users worldwide download their software and donate computer resources, then assuming an average system configuration of 2 cores, 4 GB RAM and 30 GB available disk, they have a potential Texai peer-to-peer aggregate volunteered processing power of: 200,000 cores, 400 TB RAM, 30 PB disk. A stunning set of numbers. I was going to calculate the cost of that in Amazon, but I decided not to bother. It's a lot.

SETI@home as CPU Cycle User

Of course we knew all this already. SETI@home, for example, has been running since 1999. With 5.2 million participants SETI@home now has the staggering ability to compute over 528 TeraFLOPS. Blue Gene, one of the world's fastest supercomputers, peaks at just over 596 TFLOPS. And there are many many more distributed computing projects like SETI@home supplying huge amounts of compute power to their users.

Plura Processing. Their technology lets visitors to participating webpages become nodes in a distributed computing network. Customer buy time on this network to perform massively distributed computations at over a 1/10th the cost of running the same computation on a cluster or in the cloud. Nobody goes hungry at a pot luck.

An example Plura customer is 80legs. 80legs has released an innovative web crawling infrastructure using Plura that can crawl the web for the low low price of $2 per million pages using a network of 50,000 computers. It's cheap because those computers already have excess capacity that can easily be loaned without noticeable degradation.

Exploiting all that Capacity

In the future compute capacity will be everywhere. This is one of the amazing gifts of computer technology and also why virtualization has become such a hot datacenter trend.

It's out of that collective capacity that an Ambient Cloud can be formed, like a galaxy is formed from interstellar dust. We need to find a more systematic way of putting it to good use. Plura is an excellent example of how these resources can be used as a compute grid, the next step is think of all these resources can be used as an application runtime.

Nicholas Carr reminds us in the The coming of the megacomputer, we might not even be able to imagine what can created with all our new toys:

> *Every time there's a transition to a new computer architecture, there's a tendency simply to assume that existing applications will be carried over (ie, word processors in the cloud). But the*

new architecture actually makes possible many new applications that had never been thought of, and these are the ones that go on to define the next stage of computing.

Using the Ambient Cloud as an Application Runtime

The future looks many, big, complex, and adaptive:

1. Many clouds.
2. Many servers.
3. Many operating systems.
4. Many languages.
5. Many storage services.
6. Many database services.
7. Many software services.
8. Many adjunct human networks (like Mechanical Turk).
9. Many fast interconnects.
10. Many CDNs.
11. Many cache memory pools.
12. Many application profiles (simple request-response, live streaming, computationally complex, sensor driven, memory intensive, storage intensive, monolithic, decomposable, etc).
13. Many legal jurisdictions. Don't want to perform a function on Patriot Act "protected" systems then move the function elsewhere.
14. Many SLAs.
15. Many data driven pricing policies that like airplane pricing algorithms will price "seats" to maximize profit using multi-variate time sensitive pricing models.
16. Many competitive products. The need to defend your territory never seems to go away. Though what will map to scent-marking I'm not sure.
17. Many and evolving resource gradients.
18. Big concurrency. Everyone and everything is a potential source of real-time data that needs to processed in parallel to be processed at all within tolerable latencies.
19. Big redundancy. Redundant nodes in an unpredictable world will provide

cover for component failures and workers to take over when another fails.
20. Big crushing transient traffic spikes as new mega worldwide social networks rapidly shift their collective attention from new shiny thing to new shiny thing.
21. Big increases in application complexity to keep streams synchronized acrosss networks. Event handling will go off the charts as networks grow larger and denser and intelligent behaviour attaches to billions of events generated per second.
22. Big data. Sources and amounts of historical and real-time data are increasing at increasing rates.

This challenging, energetic, ever changing world is a very different looking world than today. It's as if Bambi was dropped into the middle of a Velociraptor pack.

How can we design software systems to take advantage of the new world? Adaptation is the powerful unifying concept tying together how we can respond to its challenges. It's at the heart of the black box. Today's standard architectures do not deal well adaptation. Even something as simple as adapting elastically to varying CPU demands is considered a major architectural change.

The future will be more biologically inspired. Crazy numbers of resources. Crazy numbers of options. Crazy activity. Crazy speeds. Crazy change. Redundancy. Independent action without a centralized controller. Failure is an option and is key for creating fitter applications. Transient alliances form to carry out a task and disband. Independent evolution. And an eye constantly focused on more competitive organisms. It's an approach with less predictably and order than will be comfortable.

Potential Ideas and Technologies for the Ambient Cloud

While there's no clear blueprint for how to create this future, there are deep and compelling hints of what technologies and ideas could be part of the solution. We are going to cover some of these next.

Compute Grid

The Ambient Cloud excels as a compute grid, where a compute grid is a

framework that supports parallel execution. As we've already talked about this potential of the Ambient Cloud in our discussion of Plura Processing, I won't go into more detail here.

The Ambient Cloud is a Runtime, Not a Platform

Imagine you are chunk of running code. When you look out, what do you see? The answer, for me, defines the difference between a runtime and a platform.

On a platform you see only what the platform provider allows you to see. In a runtime you see whatever you built the program to see and whatever is out there to talk to in whatever language the things want to talk in. It seems a subtle difference, but it's actually quite major.

The power of a platform like Google App Engine (GAE) comes as much from what you can't do as much as what you can do using Google's baked in services. In GAE you don't have access to the machine you run on, you can't spawn threads, you can't access many built-in APIs, there are quotas on CPU use, and you can only access the external world through a limited set of APIs or through HTTP, as long you can do it in less than 30 seconds. You live in a virtualized world. I'm not in anyway saying this is bad. There are very good and valid reasons for all these restrictions. The problem comes when you need to do something outside what is permitted.

In a runtime environment you would not have restrictions by design. You would have access to whatever is allowed on whatever you are running on. If that's an iPhone, for example, then you are limited only by the iPhone's restrictions. If you want a native look and feel iPhone app then get down and dirty with native APIs and build one. You are able to build an application that can best exploit whatever resources are available..

The Ambient Cloud should not be an abstraction layer that tries to make the multiplicity of the world appear the same no matter how different the underlying things they really are. As seductive as this approach is, it eventually doesn't work. You are limited by the vision of the platform provider and that doesn't work if you are trying to build something unlimited.

There are, have been, and always will be attempts to make a generic APIs, but

innovation cycles usually invalidate these attempts in short order. We often see this cycle: new service, service specific APIs, generic APIs, generic frameworks, and generators from specifications. Then an innovation happens and it all falls apart and the cycle starts over again.

At certain points in the cycle technology becomes inverted. Applications play second fiddle to frameworks until the frameworks no longer look and work as well as the new fine tuned applications. In the Ambient Cloud applications always play first violin because it's only the application that can carry out their objectives while tracking the ever changing generations of technologies and services.

Applications can't wait for standards bodies. Opportunity barely knocks at all, but it certainly hardly every knocks twice. If Facebook games are the new hotness then make that happen. If Facebook is old hat and iPhone social games are the new hotness then jump on that. It's the application's job to make the user not care how it's done. For this reason the dream of utility computing may never be fully achieved. The wide variety of different clouds and different compute resources will make it difficult to come up with a true standardization layer.

There's a parallel here to military strategy. During the Vietnam War Colonel John Boyd wanted to figure out why pilots were dying at a faster rate over the skies of Vietnam than they did in Korea and WWII. After prodigious research Boyd found that the key was the *agility* of the figher plane, not the plane's speed, service ceiling, rate of climb, or even turning ability as was conventionally thought.

Agility: *The ability to change its energy state rapidly. To turn, or climb, or accelerate faster than its opponent. And most importantly, to keep up that high energy state in the grueling, high-G turns that rapidly bled out speed and options. The ability to inflict damage and get out of the way of the counterstroke.*

Conventional wisdom doesn't value agility highly. It likes the straightforward application of massive force. But Boyd found the key to victory was often: agility, speed, precision, and effectiveness. Winners don't always make the best decisions, they instead make the fastest decisions.

The Ambient Cloud values agility. You are able to do whatever is needed to carry out your objectives without being limited by a platform.

Autonomic Computing

Jeffrey Kephart and David Chess of IBM in 2003 published a stunning vision one possible future for computing their article The Vision of Autonomic Computing. Their vision is:

> *Systems manage themselves according to an administrator's goals. New components integrate as effortlessly as a new cell establishes itself in the human body. These ideas are not science fiction, but elements of the grand challenge to create self-managing computing systems.*

The inspiration for their approach is the human autonomic nervous system that unconsciously controls key bodily functions like respiration, heart rate, and blood pressure. Since computing systems don't breath yet, IBM has defined the following four infrastructure areas distributed autonomic nervous system could control:

1. Self-Configuration: Automatic configuration of components;
2. Self-Healing: Automatic discovery, and correction of faults;
3. Self-Optimization: Automatic monitoring and control of resources to ensure the optimal functioning with respect to the defined requirements;
4. Self-Protection: Proactive identification and protection from arbitrary attacks.

This is a compelling vision because at a certain level of complexity humans have to give way to automated solutions. But after a number of years it doesn't seem like we are that much closer to realizing their vision. Why not?

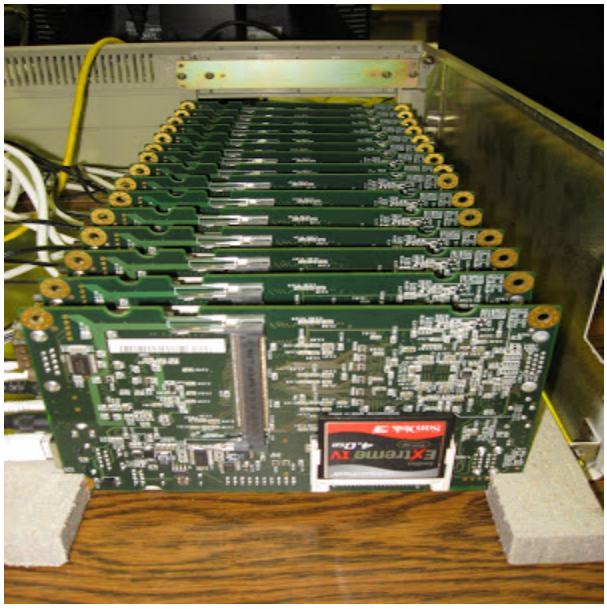
A lot of these features are being subsumed by the cloud. Each cloud vendor creates something like this for their own infrastructure. So we see Autonomic Computing in some shape or form within particular platforms. The question for applications is how far will cloud interoperability go? Will clouds interoperate at such a low level that applications can depends on these features everywhere they go? It seems unlikely as these features are key product differentiators between clouds. That means the responsibility of how to span all the different domains in the Ambient Cloud is up to the application.

One reason Autonomic Computing may not have taken off is because it doesn't contain enough of the Worse is Better spirit. It's a top down formulation of how all components should interact. Top down standards are always a hard sell in growth industries with lots of competition looking to differentiate themselves. Allowing features to bubble up from real world experience, warts and all, seems a more internet compatible model.

Another tactic that may have hindered Autonomic Computing is their emphasis on infrastructure instead of applications. Autonomic Computing hasn't provided a platform on which applications can be built. Without applications it's difficult for enterprises, datacenters, and equipment vendors to rationalize investing in such a complicated infrastructure.

With all the obvious power inherent in Autonomic Computing, it still needs to be more concerned with helping developers build complete applications end-to-end. That's what the Ambient Cloud will need to do too.

FAWN - Distributed Key-Value Database



It's tempting to think all the processors in sensor networks and cell phones can't do any real work. Not so. For a different view on the potential power of all these computer resources take a look at the amazing FAWN (Fast Array of Wimpy Nodes) project out of Carnegie Mellon University. What is FAWN?

> *FAWN is a fast, scalable, and power-efficient cluster architecture for data-intensive computing. Our prototype FAWN cluster links together a large number of tiny nodes built using embedded processors and small amounts (2-16GB) of flash memory into an ensemble capable of handling 1300 queries per second per node, while consuming fewer than 4 watts of power per node.*
>
> *We have designed and implemented a clustered key-value storage system, FAWN-KV, that runs atop these node. Nodes in FAWN-KV use a specialized log-like back-end hash-based datastore (FAWN-DS) to ensure that the system can absorb the large write workload imposed by frequent node arrivals and departures. FAWN-KV uses a two-level cache hierarchy to ensure that imbalanced workloads cannot create hot-spots on one or a few wimpy nodes that impair the system's ability to service queries at its guaranteed rate.*
>
> *Our evaluation of a small-scale FAWN cluster and several candidate FAWN node systems suggest that FAWN can be a practical approach to building large-scale storage for seek-intensive workloads. Our further analysis indicates that a FAWN cluster is cost-competitive with other approaches (e.g., DRAM, multitudes of magnetic disks, solid-state disk) to providing high query rates, while consuming 3-10x less power.*

FAWN is being positioned as a way to create low power clusters. Given the impact of electricity costs on TCO and the restricted availability of power at datacenters, this makes a lot of sense. But I think FAWN has a more intriguing role: opening up huge slices of the Ambient Cloud for greater use. Much of the Ambient Cloud looks suspiciously like what FAWN is already optimized to run on. The impact could be immense if FAWN or something like it could be deployed in the Ambient Cloud.

Although the 500MHz Atom processors being used are affectionately called wimpy, they are at least somewhat comparable to Amazon's small instance type which uses a 1 GHz Opteron. FAWN's 256 MB of main memory is comparable to low end VPS configurations. So the individual nodes are not powerhouses, but they are not without capability either. There is also a 1.6 GHz version of the Atom processor,

which is a quite capable processor slice.

FAWN is not alone in embracing the small is beautiful school of low cost, power-efficient server design. It's part of new approach called physicalization. Physicalization is an *approach to server design that packs multiple, cheap, low-power systems into a single rack space*.

The "physicalization" name is a play on virtualization. Virtualization takes very powerful machines and runs as many virtual machines on them as they can. Machine utilization goes up which reduces total cost of ownership and everyone is happy. Well not everyone, which explains the physicalization strategy.

Physicalization runs in the opposite direction. Virtualization pays a large overhead penalty, especially for I/O intensive applications. Given that data intensive applications are I/O sensitive it makes sense not to share. Instead, make small, efficient Microslice Servers that can do more work for the same energy dollar.

This is the key rational for FAWN's hardware design. It's also the key to understanding why FAWN built a key-value store at its software layer. What FAWN supports is lots of hardware nodes that need to be kept busy. So the key to FAWN is exploiting its massive parallelism using partitioned data sets. This is exactly what key-value stores like Amazon's Dynamo excel at because keys can be hashed and distributed across available nodes.

In essence FAWN is a highly power-efficient cluster for performing key-value lookups, like those used at Amazon and Facebook for fetching pictures, Facebook pages, etc. So one way to think of FAWN is as a memcached replacement that has been architected to handle 100s and 1000s of terabytes of data. While each node isn't exactly a super computer, the aggregate performance and capacity is astounding. Just imagine what that performance will be like when in the not too distant future when devices like smart phones will have 64 cores and 1 PB of flash?

When you start to think about FAWN as a key-value store/memcached replacement it gets exciting because many websites are buying into the NoSQL movement, a large contigent of which are architecting their sites around key-value stores. The important question to ask is where will those key-value stores be hosted? On

expensive EC2 instances? Or might key-value stores run on inexpensive clusters of low power machines in the Ambient Cloud, like Plura?

FAWN is just one part of a solution, it clearly doesn't solve all problems. It uses some old tech and is not currently suitable for CPU intensive tasks like video processing, but that will change once newer more powerful processors are used. Some workloads can't be easily partitioned, for example, relational databases, where the scale-down and multiply approach of FAWN isn't a win. But some hybrid form of a NoSQL store will become important in the future. The combination of horizontally scalability, joinless access, schemaless data models, fast writes and reads, and forgiving transactional semantics makes NoSQL a perfect match for much of the web. Once flash densities increase even the per byte cost advantage that disks have now for large data sets and streaming media may be breached, the Ambient Cloud itself will be a formidable storage device.

The Compound Materials approach of building systems is valuable here. We don't need one backend infrastructure that can do it all. Streaming and static media can still be served from a CDN or S3, but for the highly scalable, key-value component of your system FAWN would be a wonderful option. As a generalized key-value store FAWN can be used to store any kind of data. There's no reason an indexing system couldn't be built on top and there's no reason map-reduce couldn't run on top of FAWN either.

One huge tension we have now when designing systems centers on the great disk vs RAM vs flash memory debate. Currently there's a sort of RAM/disk hybrid as the dominant model. For large datasets RAM is used as a cache and disk as the storage of record. A company like Facebook, for example, keeps all their important warm data in 28 terabytes of memory cache. Google no doubt has astounding amounts of cache also, but they have designed their infrastructure to scale on top of a distributed file system spread across countless hard drives. In their quest to store all the world's information, Google has become the master of disk. Competing with Google based on disk technology is going to be very difficult. A better strategy may be to skip ahead to next generation technologies and architectures.

While new wonder storage technologies are always just over the horizon, our current options are: RAM, disk, and flash. RAM is fast, expensive, low density, and

power hungry, which means the amount of RAM that can be allocated per server is relatively small, especially if you need to store many petabytes of data. Disk is cheap, high density, and slow, which means it is the only game in town for storing and querying big data. Flash/SSD has a larger capacity than RAM, but less than disk, is slower than RAM and faster than disk, is power efficient, and is becoming cheaper than RAM. As flash capacity and cost per unit is hitching a ride on Moore's Law, we can speculate that in a few years that flash and flash/RAM hybrids will start replacing disk based architectures. In the Ambient Cloud that speculation makes a lot of sense as it leverages a huge amount of available unused flash storage.

Flash based systems require a different architecture, which complicates their adoption. Today the entire software stack is architected around optimizing around the deficiencies of a spinning disk. SSD is very different in character. SSD as the "solid state" part in the name implies has no seek time/head settle time, so all the optimization is at best a waste, and at worst completely counter productive. Flash is also comparatively slow to write, which is why FAWN uses a log structured file system. Using flash means the entire software stack needs to change for flash to be the best it can be. FAWN is one of the first large scale, distributed key-value stores optimized for flash, and as such it's a good model for a next generation storage platform.

Another interesting system that is comparable to FAWN is Gordon. Gordon is a *system architecture for data-centric applications that combines low-power processors, flash memory, and datacentric programming systems to improve performance for data-centric applications while reducing power consumption*.

To summarize, FAWN is a good template for a super scalable key-value data storage because:

1. It operates efficiently on the hardware profile common in the Ambient Cloud: lower power and flash storage.
2. It is capable of efficiently storing and accessing large amounts of data.
3. New nodes can dynamically leave and join the system.

Cleversafe - Space and Bandwidth Efficient High Availability

The Ambient Cloud differs from the traditional cloud in one very important way: nodes are unreliable. Take a smart phone as an example. A phone becomes unavailable when it goes in and out of signal range. People lose their phones. Or perhaps they buy a new phone. Or perhaps someone actually uses all the storage on their phone and has to dump their Ambient Cloud partition. Or perhaps sensor nodes may die at a high rate because they live out in the harsh world. There's lots of churn in the Ambient cloud and that's a problem because nobody wants to lose their data.

Reliability is commonly setup by creating a replica or two or three in different geographical locations and that pretty much gives you your high availability. With replicas there are always issues with transactions, syncronization, fail-over, recovery, etc, but since can be fairly confident that your replicas are reliable, there's no need to make dozens of copies.

In the Ambient Cloud nodes are less reliable so you need more copies to be safe. The problem is each copy uses storage and bandwidth in proportion to the number of copies. And transactions and failover become more complicated as replicas are added.

Making copies may seem like the only option for ensuring availability, but there's another little known method called information dispersal, that is used by a company called Cleversafe in their storage product. Here's their explanation:

> *Using information dispersal --slicing the data and spreading it out among servers, only to be reassembled when you need it--that very same file has a 60% storage overhead in a standard configuration. That's it. It doesn't increase. And, if a number of servers where slices are dispersed happen to fail, you can get your entire, uncorrupted and undamaged file back, as long as you have access to a minimum threshold for retrieval. That means, for example, if six of 16 slices are down, you can still get the entire file back. The entire file. At a lower cost. With less complexity. Without replication. Information dispersal does it without the stress, risk, hardware and bandwidth associated with the old way.*

This approach is based on a fascinating field of study called Reed-Solomon coding. For an overview take a look at A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems. Reed-Solomon encoding is not new, it is already used *in CDs, DVDs and Blu-ray Discs, in data transmission technologies such as DSL & WiMAX, in broadcast systems such as DVB and ATSC, and in computer applications such as RAID 6 systems*.

The takeaways for the Ambient Cloud are:

1. Data can be sliced up into multiple slices and spread across the Ambient Cloud. As parallelism is the strength of the Ambient Cloud this is a powerful synergy.
2. Efficient use of bandwidth and storage by not making complete copies.
3. Not all the slices are needed to reassemble data. It's fault tolerant. If even a substantial number of nodes go down the data can still be reassembled.
4. All updates are versioned so old and obsolete data will not be used in reassembling objects. This takes care of the scenario when a node comes back and joins the cloud again. Background rebuild processes take care of rebuilding data that is out of sync.
5. Data is secure because given any one slice you can't reassemble the whole object.

I had a long email thread with a Cleversafe representative and they were very helpful in helping me understand how their system works. While Cleversafe is promising, because it does provide reliability over unreliable elements, there are some issues:

1. Cleversafe, quite understandably, uses an appliance model. To use their system you need to buy their appliances. That won't really work over the Ambient Cloud as the Ambient Cloud is effectively the appliance.
2. Their target market are large files, generally several gigabytes files in length, with one writer at a time. The overhead of dealing with small objects with millions of readers and writers is way too inefficient.

The architecture of their system is based around disk storage managed by sliceservers. It's possible using flash based storage might make a more parallelized design possible, especially if techniques like eventual consistency and client based

read repair were employed.

While I haven't been able to work out the how, I can't help but think there is something essential in this approach, building reliability on top of unreliable components, that meshes well with the Ambient Cloud.

OceanStore - Global Data Store Designed to Scale to Billions of Users

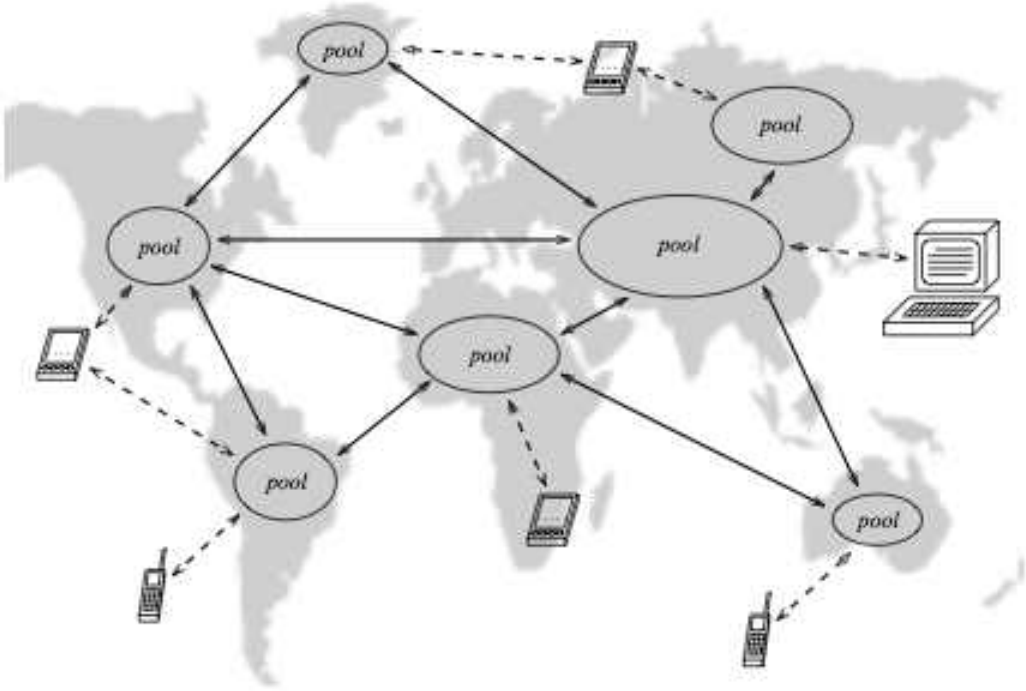


Figure 1: ***The OceanStore system.*** **The core of the system is composed of a multitude of highly connected "pools", among which data is allowed to "flow" freely. Clients connect to one or more pools, perhaps intermittently.**

There have been several attempts at large scale distributed storage systems. One of the most interesting is OceanStore, which was designed for 1 billion users, each storing 10,000 objects, for a total of 10 trillion objects. What is OceanStore?

> *OceanStore is a utility infrastructure designed to span the globe and provide continuous access to persistent information. Since*

> *this infrastructure is comprised of untrusted servers, data is protected through redundancy and cryptographic techniques. To improve performance, data is allowed to be cached anywhere, anytime. Additionally, monitoring of usage patterns allows adaptation to regional outages and denial of service attacks; monitoring also enhances performance through pro-active movement of data.*

Here's a quick overview of OceanStore's main features:

1. Infrastructure is comprised of untrusted servers
2. Data protected through redundancy and cryptographic techniques
3. Uniform and highly-available access to information; separation of information from location
4. Servers geographically distributed and exploit caching close to clients
5. Data can be cached anywhere, anytime, and can flow freely
6. Nomadic data (an extreme consequence of separating information from location)
7. Promiscuous caching: trade off consistency for availability; continuous introspective monitoring to discover data closer to users
8. Named by a globally unique id, GUID
9. Objects are replicated and stored on multiple servers
10. Provide ways to locate a replica for an object
11. Objects modified through updates
12. Messages route directly to destination, avoiding the round-trips that a separate data location and routing process would incur

What is attractive about OceanStore for the Ambient Cloud is that it:

1. Is a generalized object store and not a pure file system.
2. Assumes from the start that the infrastructure is constantly changing and can handle nodes dropping in and out.
3. Provides truly nomadic data that is free to migrate and be replicated anywhere in the world.
4. Built-in continuous testing and repair of information. Automated preventive maintenance.

All these are necessary attributes of a planet-scaling system. Of particular interest is the ability to keep numerous replicas. Imagine millions of users accessing data. If there's only one copy of data then that CPU will be swamped serving that data. So copies need to made in order to let massively parallel streams of users access data in parallel. Not many systems have this feature. Keeping replicas to a minimum is the usual rule because they are hard to maintain. A side benefit is geographic information can also be used to route to the nearest replica.

A prototype called Pond was built around 2003 and at that time Pond, in the wide area, out performed NFS by up to a factor of 4.6 on read intensive phases of the Andrew benchmark, but underperforms NFS by as much as a factor of 7.3 on write intensive phases. The slowness of the writes is attributed largely to erasure coding, which is similar to how Cleversafe works. One possible fix is to use Low-Density Parity-Check (LDPC) codes instead.

In the paper Erasure Coding vs. Replication: A Quantitative Comparison, they show show that a replication based system requires an "order of magnitude more bandwidth, storage, and disk seeks as an erasure encoded system of the same size" to achieve the same mean time to failure (MTTF). So clearly this technology has advantages for the Ambient Cloud. The problem is it also has some drawbacks. Opening connections to a large number of machines in order to reassemble small objects will be slow. So just using erasure encoding to store objects probably won't work.

The innovative solution recommended by the paper is to separate the mechanisms for *durability* from the mechanisms of *latency* reduction. Erasure encoding can be used for durability and a caching layer can be used for latency reduction. This is exactly the approach Pond takes in their implementation.

It's not clear what the current status of OceanStore is, it hasn't been updated for a while so it may be dead. For the project publications see The OceanStore Project.

NoSQL - Thinking Different

NoSQL has been mentioned a few times already. In this section what I mostly mean by NoSQL is the spirit of tackling problems differently, not relying on how things have always been done and being willing to overcome the inertia of the past. That's

the same sort of adventuring spirit that will be needed to planet-scale.

An example of thinking differently at scale is how Google has you code using Google App Engine (GAE). GAE is built on a distributed file system so getting each entity is really retrieving a different disk block from a different cluster. This is already very different than a relational database where the assumption is that lookups are very fast because of how the data is stored. Individual gets in GAE are actually slow, but they are highly scalable because they are distributed across a sea of disks. Doing a table scan becomes very slow because each entity could be anywhere in any cluster. The implication is that as data becomes large aggregate operators like ave() become impossible. For the programmer what this means is that the average has to be calculated on the write operation, not on the read. In other words, precompute on writes so your reads are very fast. And if the write volume is high the sums will have to be sharded. Another difficult concession to scalability concerns.

All these type of techniques, and there are many more, are completely alien to what people are used to with SQL. But what Google did is create a system that could scale while still being usable enough for developers. Amazon's SimpleDB also asks you to make a series of compromises in order to scale. Developers trying to figure out how to structure a system using key-value stores in order to have fast scalable lookups are dealing with similar issues. They are asking, what needs to change so we can scale our applications?

So we have to be willing to step outside our comfort zones. Play with different models of consistency. Play with moving responsibility to the programmer (and hopefully back again). Play with different programming patterns. Play with how data is stored. Play with different ways of making queries using approaches like Pig Latin, MapReduce, HIVE, and Dryad. The idea being that we may have to write applications a lot differently in the Ambient Cloud than when using a centralized SQL database.

Market Based Resource Allocation

The Internet is expanding to include a large number of competing services: CPU, storage, database, search engine, discovery, payment processing, application

specific, security, privacy, memory, pub/sub, email filtering, blacklisting, messaging, job scheduler, and other best of breed services. Both general purpose and specialized clouds will coexist.

Competition between services and freely available resources in the Ambient Cloud creates a continuously variable supply at different price/performance curves. Adaptive applications can reconfigure and redeploy themselves to take advantage of available opportunities when they occur.

These different price/performance curves are to software like chemical gradients are to single-cell organisms. Single-cell organisms move in the right direction by detecting chemicals in their surroundings. If they sense something pleasurable, like food, they follow in the direction of the highest concentration of nutrients (the gradient). If they sense something painful, like a toxic chemical, they make a random move and start over again. Foraging paths follow regions of high food quality. Here we see the beginnings of our nervous system and brain.

Applications in the Ambient Cloud can similarly follow a complex multi-dimensional gradient in order to optimize its goals. Goals can include: minimize cost, maximize throughput, maximize performance, maximize availability, and so on. Cost will become a major part of architecture evolution.

We've already seen this kind of thinking in our discussion of algorithmic trading. But we see it in a lot of other areas of our lives too. eBay is one example of a trading market that most of us have experience with.

If you've flown then you probably know the airline industry uses dynamic pricing in order to increase revenue by selling goods to buyers “at the right time, at the right price.” Airlines now let software make automatic pricing decisions on over 90 percent of tickets sold.

Maybe you've used Priceline.com to find a hotel, airline ticket, or rental car? Priceline use a value-based pricing approach where buyers make an offer on a product and they see if there's a seller willing to make a deal at that price.

For ages we've had commodity markets where raw commodities like food, metals, electricity, and oil are traded. Aren't compute resources are just another commodity

to trade?

What markets are good at is allocating supply to demand. What automated market mechanisms are good at is doing this quickly, efficiently, intelligently, without human intervention. All of which are required if applications are going to trade resources. We've already gone over several examples of how this works in other markets, there's not reason in shouldn't work with compute resources as well.

There are a couple problems with this vision:

1. There is no market currently for compute resources.
2. Application architectures are not flexible enough to integrate market mediated resources even if they were available.

So we are in a bit of a chicken and egg situation here. Both angles need to be worked. With the service model becoming dominant, command and control through APIs is possible, which should enable a market to be made. And traditional architectures are relatively static, rigid, and require a lot of hand holding. More work will have to be done in this area as well.

The reason why I think a market is so critical to the Ambient Cloud is because I think it's the only way to make the dynamic pool of exponentially growing compute resources available to applications. Applications can only consume resources if they are packaged up and made available. Since the resources are dynamic there needs to be some sort of exchange set up to allow applications to know when new resources become available and search for existing resources. Once resources have been found a contract needs to entered, payment made, and then the resource need to be integrated into the application and released when done. Given the complexity, size, and rate of change of this operation, only a market mechanism will work to match supply and demand.

The first few parts of this section have been on possible directions to take on designing software for the Ambient Cloud. There is inspiration in FAWN, Cleversafe, OceanStore, and NoSQL. But that's not enough. What developers need to write applications is a stable, guaranteed environment for that software to run in. And I think a market enabled Ambient Cloud can be that platform.

Applications as Virtual States

As I was writing an article on the architecture of the Storm Botnet, I couldn't help but notice the deep similarity of how Storm works and changes we're seeing in the evolution of political systems. In particular, the rise of the virtual-state. As crazy as this may sound, I think this is also the direction applications will need follow to survive in a complex world of billions of compute devices.

You may have already heard of virtual corporations. Virtual corporations are companies with limited office space, a distributed workforce, and production facilities located wherever it is profitable to locate them. The idea is to stay lean and compete using the rapid development and introduction of new products into high value-added markets. If you spot a market opportunity with a small time window, building your own factories and hiring and engineering team simply isn't an option. Building factories is a bit old fashioned and is left to the select few. These days you get an idea for a product and contract out everything else you possibly can. It doesn't really matter where you are located or where any of your partners are located. If part of your product requires a specialized microprocessor, for example, you'll contract out the R&D and the design. The manufacture will be contracted out to a virtual fab, then the chip will be sent to a contract manufacturing service for integration. Look ma, no hands.

Futurists say land doesn't matter anymore. Nations don't matter anymore. Entire relationships are abstractly represented by flows of money, contracts, information, and products between all these different agents. Interestingly enough, what technology is the absolute master of managing flows? Applications! But we are getting ahead of ourselves here.

Let's first get a definition of virtual state. Foreign Affairs magazine published an article titled The Rise of the Virtual State, which defines virtual state as: *a country whose economy is reliant on mobile factors of production.*

Immediately it looks as if only old style countries can be virtual states. That's really just a historical bias. A quaint throwback to when land was king. R Jagannathan sums it up nicely with this quote: *In the post-modern, post-internet world, many states exist more in the mind than in reality.*

So let's just say a virtual state can be any entity requiring political representation. Surprisingly, software applications fit this definition. Applications deal with high finance, they negotiate, they require global economic access, they execute policy, and they operate globally in many legal and political jurisdictions. If that kind of entity doesn't require political representation then what does? And after all, if you are going virtual, why not go all the way?

The *mobile factors of production* part of the definition is a bit confusing, but turns out to be a key driver for why applications make sense as virtual states. Factors of production *are the resources employed to produce goods and services, but do not become part of the product.*

Historically these factors have been land, labor, and capital goods. And historically agricultural output has been the biggest component of GDP. Agriculture relies on immobile factors of production like land, climate and geography. If you want to grow a crop it will be difficult to compete with someone if they have better land and climate than you do. With their immobile factors of production they have a competitive advantage. For example, I live in the under appreciated Santa Cruz Mountains wine appellation. The wines of this region reflect the dramatic microclimates created by a unique mixture of mountain topology and ocean proximity. Nobody else in the world can recreate this sublime mix of circumstances. However, advantages of this sort have become less and less important in the evolving modern economy.

In the modern era, we've added a few more factors of production: financial capital, technology, human capital, intellectual capital, and business know-how. The big difference in this list when compared with the original list, is that these are mobile factors of production. They are mobile because they are abstract and intangible, they aren't tied to land, they can be anywhere in the world. Money flows. People flow. Knowledge flows. Contracts flow. Skills flow. And trade flows to the entities with the greatest absolute advantage in these factors. If, for example, you land a contract to represent the next great NBA superstar directly out of grade school, you think to yourself why pay Nike all that money to make a sports shoe? You can hire all the experts (design, finance, manufacturing, legal, marketing, etc) and keep all the profit. It doesn't matter where these experts are as long as they can all communicate. They key to the deal is the marketable element, your NBA star, once you have that

you have what matters, the rest can be assembled.

They key takeaways from this analysis are that in the modern economy land isn't key. Making money depends on intangible and abstract skills that can be located anywhere and do not require a physical presence to carry out. The ability to create and manage these flows between networks has become a prime factor of production.

Land not being primary means countries are no longer in the business of acquiring more land, which is quite a change historically as pretty much all of human history has been about killing people to acquire more land. The prime example of this is Hong Kong. A measly 426 square miles in size, Honk Kong has become a major economic player on the world stage.

The emancipation of statehood from land makes for a very confusing world. One consequence I see is that a virtual state is really not different than a virtual corporation, there's simply an added political dimension attached. The political dimension stems largely from historical land entanglements in the form of nation states. To any observer of the political process it's quite clear corporations have their own political interests, so the distinction quickly becomes moot, which means the number of virtual states will multiply radically. John Robb even argues that *Al Qaeda network is a sort of virtual state, with a consistent source of finance, a recognized hierarchy of officials, foreign alliances, an army, published laws, even a rudimentary welfare system. It has declared war on the U.S. for much the same reason that Japan did in 1941: because we appear to frustrate its ambitions to regional hegemony.*

The goals of the state and corporations have also merged. The goal of a virtual state is to make money, not acquire land, rule people, or seek a greater good. It's entirely selfish in nature. A lizard mind with the Internet as its body. There's no centralized bureaucracy and no official armed forces. The role of the virtual state is to develop an overall strategy; invest in people, not expensive production capacity; contract out other functions to states that specialize in or need them. Pretty much exactly the same mandate as a virtual corporation.

Why does this matter? This is a question I try to ask myself before I go to far off the deep end. Well, I don't think applications will be virtual states in the sense of independent AIs and all that. This dystopia is often portrayed in SciFi Novels, but

I'm really trying to picture how this system will work, not just make up a story.

To really make this work humans and applications will form close partnerships and it's that unit that becomes a virtual state. Multi-national corporations have gone way past caring about their association with a nation state, this trend will continue down to lower and lower levels of organizations. The automation process that has always been the strength of software makes this possible. With the hollowing out of the middle class each of us will be left to make a living in this new environment and applications will help make this possible.

Applications can not be autonomous because software doesn't have goals or objectives. Only humans have goals and these come from a long history of physical and cultural evolution. These goals are imprinted into applications and it's the applications job to achieve the state objectives.

We've already seen how applications play this part of the partnership with black box trading in financial markets. These black boxes are programmed with goals and models and tactics by a very select group of people. Few others have any idea how these things work or why the computers decide what they decide.

Another stunning example of how this people-computer partnership would work is detailed in the Wired article The Answer Factory: Demand Media and the Fast, Disposable, and Profitable as Hell Media Model, written by Daniel Roth. The story is about a company called Demand Media. Demand Media data mines the topics people are searching for on the web using a complex monitoring process and algorithmic sifting. For each candidate topic it's calculated what advertisers are willing to pay for hits on the topic. Profitable topics are then put into play. Someone is hired through an automated bidding process to create the content at a very low price, low enough to still be profitable given low adsense payouts, so these people are working for cheap. Once produced the content is published and the adsense rewards harvested.

This is an amazingly sophisticated operation that is almost completely driven, coordinated, and managed by software. It's not hard to project this strategy out for material products as well. Software can scour the web, data mining potential products. Information from human trend spotters can be integrated in. The

determination of what products to make can be made through some combination of automated and manual processes. After that much of the rest of the process can be automated. Financing can be arrange through some sort of automated credit check. Building and designing the products can sub contracted out. Manufactures can be fed the results of the design. The product can be automatically made available retail outlets, put on Amazon, put on eBay, put in vending machines, made available to affiliate programs, and so on. Commercials can be contracted out for shopping channels. Firms can be hired to create advertising and marketing campaigns. Time can be automatically purchased on radio and TV stations. Bloggers can be bought to write articles. Social media experts can work the microblogs. Reviews can be added to Amazon and Yelp. People can buy on-line or at a mall or wherever it's convenient. The accounting, reporting, and billing is automated too. All the while the application is learning using measurable feedback so it can do a better job next time.

And consider: almost this entire process can be automated and controlled by an application. It would operate relentlessly, 24 hours a day of everyday, churning out products and services. This is actually a damn scary thought.

The key is to recognize it's all about automated and structured flows controlling mobile factors of production. Software rules processing these kind of flows. It can do them faster, smarter, and it never gets tired. All that is required is an entire chain willing to work on contract, carry out an objective, and work in a coordinated fashion.

In this way applications become virtual states. Again, not in some silly SkyNet sort of way, but in a very practical this is how applications will need to be structured in order to work across billions of devices sort of way. Software has traditionally been thought of something contained, something within well known and controlled boundaries. The boundary was a single machine at first, then small clusters of machines, then huge clusers of machines, then even entire clouds. Crossing the Ambient Cloud boundary means that software is not contained anymore, software does the containing. It moves where it needs to for reasons dictated by its high level goals. It expands and contracts as needed. It negotiates with 3rd parties. It protects itself from attack. It protects itself from failure. To a great extent software becomes independent, thus the idea of software becoming like a virtual state.

The criminal networks behind botnets like the Storm Botnet have already made the transition to a virtual-state, operating a sort of black hat bazaar that has both economic means and military capability.

For a good overview of the black market economy please read The Cybercrime Blackmarket. It documents an amazingly rich world that doesn't physically exist in any one place, yet is united behind common objectives and methods.

In the cybercriminal black market we have leadership by the criminal heads, a highly skilled and motiviated work force, a means of making money, a currency, and even a loose system of laws and morality.

And oh, they also have a military. Not a military arm not in the conventional sense, but the same DDoS capabilities that are sold on the market can be used as a weapon. Blue Security was literally forced of out business by a relentless DDoS attack carried out by an organized crime ring angry at the success of the firms anti-spam software. Other digital and physical attack strategies are certainly within their grasp.

It's telling that the same group that first learned how to exploit the Ambient Cloud is also one of the first to form a virtual state. The two go hand-in-hand.

Conclusion

A lot of people think we've started on a certain path and the next steps are set in concrete. Everything will move to the cloud, there will be a few big vendors, and everyone will end up dependent on one of these winning platforms. It's as good as done. While I may have felt this way once, I do not anymore. Instead I feel that we are is smack dab in the middle of a revolution and it's right now when we have a say in what direction it goes.

Clay Shirky in Newspapers and Thinking the Unthinkable captures two other points in time that have hung suspended between revolutionary eras. In 1500 Gutenberg’s invention of the printing press was transforming society in ways people weren't consciously aware. Yet everything was changing. The world before the printing press and the world after the printing press were on completely different paths, paths to destinations that could not have been predicted at the start of the

journey.

By analogy, Shirky says the Internet has had a similar effect on newspapers. His radical conclusion is that *we shouldn't be worried about keeping newspapers around at all, that's not what is important. What is important is journalism, not newspapers.* When the power of this statement finally seeped into my little brain, it gave me a jolt as I realized it's not the form that ever mattered, it was the service that mattered. At one point the form and the service became fused in our collective minds and it was hard to untangle them and see them as separate things. We got all worried that newspapers were dying and thought journalism was dying too. Saving journalism is a completely different task than saving newspapers. The former is doable, the latter is not, but that's OK, it was always the journalism that needed saving anyway.

Again by analogy, what we need are applications, not datacenters, or virtualization, or servers, or metered billing, or platforms as a service, or infrastructure as a service. It's the applications we need and use that are important, not the platform. We've confused the two. The Ambient Cloud can revolutionize how we build, deploy, run, and use applications. We just don't fully realize it yet because the Ambient Cloud version of the printing press still needs inventing, but all the components are clicking into place. This quote by Shirky tries to explain what it's like to be in the middle of such a sea change:

> *That is what real revolutions are like. The old stuff gets broken faster than the new stuff is put in its place. The importance of any given experiment isn't apparent at the moment it appears; big changes stall, small changes spread. Even the revolutionaries can't predict what will happen. Agreements on all sides that core institutions must be protected are rendered meaningless by the very people doing the agreeing. (Luther and the Church both insisted, for years, that whatever else happened, no one was talking about a schism.) Ancient social bargains, once disrupted, can neither be mended nor quickly replaced, since any such bargain takes decades to solidify.*

For me, the core institution in this context means datacenter. Datacenters existed long before the cloud and in a sense clouds are really just datacenters with an API. To be revolutionary it's datacenters that must be transcended. The Ambient Cloud,

which includes datacenters, is completely different, going far beyond them. These difference will create their own markets in the same ways clouds have created a market distinct from the datacenter market. I like this explanation of the process (I wish I could give credit, but I simply can not find where I got this from):

> *Actually in many instances nowadays, it is technology that is creating markets. Nobody asked for the internet, web, iPhone, or social networking. Technology is leading the way in creating value in addition to making people and markets aware of the value. Same case with Cloud Computing. The startups, ISVs, and the CIOs have not asked for a cloud just like the consumers never asked for the web. It was created by technologists and markets saw value in it and started using it at a mass scale. So the technology has preceded - driving the markets and users, not the other way around.*

Amazing and unexpected things happen when the right pieces are put together with the right levels of abstraction. The moral of the story for me is that we can't wait for the change to become obvious before acting. We need to make it happen before the path has been laid out. So this is a call to start thinking about the Ambient Cloud now and start building those pieces that can fortuitously fit together and change everything. Build it and they will come. They will come because we will all need someplace open to go.

Our fate is not already written in the clouds. My mind changed on that subject after pondering how crazy it is to help others create--in mind and deed--the new frontier only later to be become digital sharecroppers on that creation. The honor of renting an expensive patch of land in a datacenter doesn't seem a fair exchange.

Why not emigrate instead? Don't we always look for a new land of opportunity? Let's board a ship, let's take a magnificent voyage to this vast new world that is the Undiscovered Country of the Ambient Cloud. The neo-feudal model wins only if we let it.

Related Articles

1. Above the Clouds: A Berkeley View of Cloud Computing by Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica and Matei Zaharia.
2. Vast Peer-to-Peer vs Single Awesome Server - What's Best? by Steve Reed
3. Introducing the FAWN - FAWN is a fast, scalable, and power-efficient cluster architecture for data-intensive computing. A FAWN links together a large number of tiny nodes built using embedded processors and small amounts (2--16GB) of flash memory into an ensemble capable of handling 1700 queries per second per node, while consuming fewer than 4 watts of power per node.
4. InterCloud Portability Reality Check by Rich Miller
5. Cloud computing and the return of the platform wars by Dion Hinchcliffe
6. Newspapers and Thinking the Unthinkable by Clay Shirky.
7. New Multitasking VM Does More for Small Devices by Al Riske
8. Multi-core smartphones! (Not the iPhone, yet) by MG Siegler
9. The Coming Swarm by JOHN ARQUILLA
10. Living In A Post Rails World by Kurt Schrader
11. Eucalyptus Systems - open source private cloud platform
12. Parasitic JavaScript by Nick Jenkin. This project explores a parasitic computing solution to turn web browsers into participants of a distributed computing project.
13. NVIDIA® CUDA - a general purpose parallel computing architecture that leverages the parallel compute engine in NVIDIA graphics processing units (GPUs) to solve many complex computational problems in a fraction of the time required on a CPU.
14. Cloud computing scenarios
15. Here Come the Specialty Clouds by Stacey Higginbotham - include gaming, graphics, transcoding clouds, and hardware specific clouds.
16. Plura Processing - webpages become nodes and perform very small computations for the application running in a distributed computing network.
17. Brain Science Podcast #51: Seth Grant on Synapse Evolution
18. All My Data - secure, decentralized, fault-tolerant filesystem. Filesystem is encrypted and spread over multiple peers in such a way that it continues to function even when some of the peers are unavailable, malfunctioning, or

malicious.

19. The Data Centre as a Computer - an introduction to the design of Warehouse-Scale Computers by Luiz André Barroso and Urs Hölzle of Google
20. High Performance Parallel Computing with Clouds and Cloud Technologies by Jaliya Ekanayake, Geoffrey Fox
21. Who Has the Most Web Servers? by Rich Miller
22. Small Pieces Lossely Joined by David Weinberger
23. Parallelizing the Web Browser by Christopher Grant Jones, Rose Liu, Leo Meyerovich, Krste Asanovi´c, Rastislav Bodík
24. Scaling Bandwidth - SMPs may have cores, but clusters have bandwidth by Douglas Eadline, Ph.D.
25. Incremental Twiddling - As GPU Clusters hit the market, users are finding small code changes can result in big rewards. by Douglas Eadline, Ph.D.
26. Why are Facebook, Digg, and Twitter so hard to scale?
27. Swarm is another related architecture "allowing the creation of web applications which can scale transparently through a novel portable continuation-based approach. Swarm embodies the maxim 'move the computation, not the data'".
28. http://blog.threatexpert.com/2009_09_01_archive.html - Zeus/Zbot is an annoying threat. Its persistence is explained with a fact that it's generated by a large army of attackers who use Zeus builder.
29. What are Smart Objects and the Internet of Things?
30. Connected Things - The World Going Wireless
31. Internet of things What’s up and why?
32. Web 3.0: The API-driven application
33. Pervasive computing - nature does, layered complexity
34. The Trillion-Node Network
35. Online Gizmos Could Top 50 Billion in 2020
36. Bell’s Law for the birth and death of computer classes: A theory of the computer’s evolution
37. Multicore: Fallout of a Hardware Revolution
38. How Google Serves Data from Multiple Datacenters
39. The Web in Danger

40. The Evolving Field of Distributed Storage
41. Farsite - Federated, Available, and Reliable Storage for an Incompletely Trusted Environment
42. Tapestry and OceanStore
43. Erasure Coding vs. Replication: A Quantitative Comparison
44. Distributed Hash Tables Links
45. TPC-C: comparing SSD & disk
46. IDEA: An Infrastructure for Detection-based Adaptive Consistency Control in Replicated Services
47. Market-based Resource Allocation for Distributed Computing
48. Wuala - Secure Online Storage
49. Welcome to Tahoe-LAFS
50. Boffins fawn over dirt cheap server clusters
51. ARM attacks Atom with 2GHz A9; can servers be far behind?
52. Economic Denial of Sustainability
53. Building Scalable, Complex Apps on App Engine
54. A Design for a Distributed Transaction Layer for Google App Engine
55. Somali sea gangs lure investors at pirate lair
56. Extracting Guarantees from Chaos
57. Erasure-Coding Based Routing for Opportunistic Networks
58. V8 JavaScript Engine
59. Scaling JavaScript to Large Web Applications
60. New Multitasking VM Does More for Small Devices
61. Robust and Efficient Data Management for a Distributed Hash Table (2003)
62. Open Hardware and Top-Down Design
63. Google demonstrates quantum computer image search
64. How Robber Barons hijacked the "Victorian Internet"
65. 4GW -- FOURTH GENERATION WARFARE
66. The Metamorphosis to Dynamic Trading Networks and Virtual Corporations
67. Content Farms: Why Media, Blogs & Google Should Be Worried
68. Was Moore's Law Inevitable?
69. Expanding the Cloud - Amazon EC2 Spot Instances
70. Simple RFID tags evolve into complex ultra-low-power SoCs, NXP

expects

71. Makers by Cory Doctorow. “A book about people who hack hardware, business-models, and living arrangements to discover ways of staying alive and happy.”
72. Trading Shares in Milliseconds. *The profits go to the company with the fastest hardware and the best algorithms--advantages that enable it to spot and exploit subtle market patterns ahead of everyone else.*
73. Community Cloud Computing. *Community Cloud Computing (C3) offers an alternative architecture,*
created by combing the Cloud with paradigms from Grid Computing, principles from Digital Ecosystems, and sustainability from Green Computing, while remaining true to the original vision of the Internet.
74. 3D Printing. A form of additive manufacturing technology where a three dimensional object is created by successive layers of material.
75. Bots and Automated Tyranny. *Here's a* brilliant video presentation *by the author Daniel Suarez (*Daemon *and* Freedom*) on the rapid proliferation of computer automated decision making.*
76. Daniel Suarez - Daemon: Bot-Mediated Reality. Narrow AI *bots power the modern world, which gives them so much power as we live in a data rich environment.*
77. Neil Gershenfeld: The beckoning promise of personal fabrication
78. Asankya RAPID Protocol. Multipathing TCP/IP sessions increases bandwidth and cuts latency. One can imagine utilizing mutltiple cell phone towers and wifi to increase mobile netwoking perfomance.
79. Storing and Accessing Live Mashup Content in the Cloud by Krzysztof Ostrowski (Cornell University) and Ken Birman (Cornell University).
80. Using a Market Economy to Provision Compute Resources Across Planet-wide Clusters by Murray Stokely, Jim Winget, Ed Keyes, Carrie Grimes, Benjamin Yolken.
81. Misco: A MapReduce Framework for Mobile Systems
