C3B Consulting's

INSIGHT-SPECTRA

Applying common sense to using technology intelligently

www.insight-spectra.com

Volume 22 Report 1; September 2009

Introduction to INSIGHT-SPECTRA

In 1987 we started the **SAA SPECTRUM**, to provide commissioned commentaries on IBM's System Application Architecture -- which was indubitably the first significant initiative to seek serious heterogeneous system and application integration. In the early 1990s our focus broadened -- still encompassing integration but now embracing all vendors' middleware; accordingly we changed the title to **MIDDLEWARESPECTRA**. For the 20 years until 2007 (when Forrester Research obliged a temporary cessation of publication), we delivered contributions and commentaries from a variety of sources.

In 2009 it seems appropriate that, with the freedom to do so, we should re-awaken the **'SPECTRA**. In this, a new format that borrows from past ones, the name will be **INSIGHT-SPECTRA**, with the tag-line '*Applying common sense to using technology intelligently*'. Indeed, it is this tag-line which provides the key to our future coverage.

With this in mind, **INSIGHT-SPECTRA** will focus on:

- what can be done
- how it can be done
- 'how it was done' (using case studies).

While we expect the overall scope to expand with time, there are initially 6 key areas where **INSIGHT-SPECTRA** will concentrate. These include:

- Non-Linear Licensed Software (or how CIOs can save fortunes)
- cloud computing and virtualization (trying to deliver tangibility out of nebulosity)
- event processing, including CEP (especially when integrating external event sources like GPS, or EP-GPS)
- application management and integration (including ADAD, the Automation of the Discovery of Applications and their Dependencies)
- enterprise architecture (explaining what it is, how it is useful and what enterprise architects need to do)
- combining energy and IT management (or TAMSECE, for Total Automated Metering Solutions for an Energy Conscious Environment)
- etc.

In this Report we start by focusing on on:

- Why CIO's should expect any application to run anywhere at any time
- Non-Linear Licensed Software
- Amy Wohl's Cloudosophy
- ADAD
- Why IBM's InfoSphere Streams is a game changer
- 40GB of DB processing on $10/day.

We hope that you will find this new format — designed to be read easily on screen or as easily printed — both useful and relevant. Please note that **INSIGHT-SPECTRA** may be reproduced for individual use. However, bulk reproduction (more than 5 reproductions) -- whether by electronic means or in print -- requires the written permission of C3B Consulting Ltd.

All and any feedback is welcome.

Charles Brett

Charles Brett

charles.brett@c3bconsulting.com
www.insight-spectra.com
INSIGHT-SPECTRA
(a production of C3B Consulting Ltd © 2009)

Any application running anywhere at any time – a wholly achievable CIO objective today

Amidst the noise about cloud computing, simple truths are easily overlooked. Enterprises typically have hundreds or thousands of applications – the majority running on dedicated servers often specified (along with their back-ups) to support just one particular application. This is insanely expensive to buy, operate and support. Furthermore, system utilization is often even lower than represented (in one financial institution, daily utilization was measured as being over 55%, but investigation revealed this was measured at the peak usage point in each day -- opening and closing – while actual working-hours utilization was less than 20%).

But there is a further cost often not calculated by enterprises – the price of inflexibility. What if there is a sales surge, a market melt-down or a short-term crisis (like a plane, train or bus crashing)? For hours or days, peak usage may need to be far in excess of what can be reconfigured to be available. Yet the standard approach for IT is to plan only for 'reasonably anticipated workloads.' Combine this with application-specific servers and the result is an unintended rigidity acting to prevent (not merely frustrate) flexibility. This is what CFOs and CIOs endure – and for which enterprises pay through the nose.

This is unnecessary. With thought, planning and discipline it is quite practical for enterprises to embrace and deploy their own IT environments where any application can run anywhere at any time – and possess the flexibility to satisfy both anticipated and unexpected work load changes. To make matters worse – this approach can be far cheaper than traditional IT solutions.

How might this be achieved? By the application, across a matrix of three disciplines — thought, planning and consistency — and 6 straightforward principles. Those principles are:

- Adopt commodity purchasing principles for IT infrastructure and operations (buy commodity hardware plus commodity -- or Non-Linear Licensed Software*;
- Introduce infinite serving – to replace dedicated serving (and, in so doing, simplify operations while reducing costs);
- Implement intelligent virtualization — make everything in system software available to every application, without duplication;
- Automate work load management, including activity initiation capabilities;
- Exploit unconventional data stores (like Hadoop — see page 11) to process, analyze and feed (on and off) an enterprise's expensive transactional RDBMSs rather than try to do everything with the latter,
- Focus and remove the enterprise from the 'system software installation business' -- to concentrate on running the applications the business needs where and when required.

A flights of fancy? Not at all.

Analysis

While only a relatively small number of enterprise organizations are doing this today, it is happening. Some may call this enterprise cloud computing. Properly practiced, it is more radical and more radically valuable approach because it is not generic (as in a cloud) but becomes specific to each enterprise's application needs while reducing a wide range of IT-related costs while increasing flexibility.

What more could a CFO or a CIO want?

** Non-Linear Licensed Software is software where, when you add (say) 10 or 100 processors or cores, you do not have to buy 10 or 100 additional licenses. Open Source Software is the most obvious example, but is not the only one. For too long, IT users have been bled dry by vendors exploiting linear licensing combined with application specific servers. This is no longer necessary at system software levels, though it will continue at the applications level — which should be where an enterprise's differentiation occurs. (Also see page 4).*

Charles Brett
President, C3B Consulting Ltd.
(www.c3bconsulting.com)

The profound financial attraction of Non-Linear Licensed Software (N-LLS)

Linearly licensed software is the norm today. When an enterprise adds (say) 1 (or 10 or 100) processors or, nowadays more often, 'cores', most traditional software vendors require that enterprise to buy a corresponding number of additional licenses. Despite 'package deals,' there is usually a direct linear relationship between the number of additional cores and the number of additional licenses that must be purchased. As many enterprises have discovered to their chagrin, this can be unexpectedly expensive.

When an additional license cost is associated with a business application, the expense (though substantial) may reluctantly be acceptable — because it 'contributes to the business.' However, when additional licenses involve 'enabling middleware or infrastructure software' — and C3B Consulting (C3BC) includes databases in this category — then the reaction of most organizations is rarely forgiving.

If this comes as an unexpected shock it is because too many application development teams do not think through the operational implications (as C3BC research confirms, this happens all too frequently). Justifiable fury on the part of those funding IT initiatives are a common result. That said, until relatively recently, there was little alternative but to submit to whatever software vendors demanded.

Some background: C3BC started watching the infrastructure software space some 20+ years ago. In that time the basic picture had not greatly altered.

What has recently changed is that it is now quite possible to implement a complete software infrastructure, from the 'metal through to below the business application' -- using Non-Llinear Licensed Software (N-LLS).

Our recent investigation (April 2009) of Open Source and Grid-enabled software showed that for all the major, traditional infrastructure software elements there are now available one or more N-LLS options (with one possible exception). For example, for an N-LLS operating system, there is Linux; for virtualization, there is KVM; for massive and unconventional data storage/processing, there is Hadoop (which can also be used to 'feed' traditional transactional RDBMSs). For messaging there are OpenJMS or AQM; for process management and job scheduling, Condor is becoming ever more attractive.

Enterprises can go even further. MySQL is an N-LLS relational database. With Fuse or Apache, many enterprises have implemented N-LLS application servers. There is an Open Source version of Tuxedo, an elderly but still elegant transaction and messaging manager.

N-LLS versions of JVMs abound, and Java can pretty much be considered N-LLS, as are Python and many other development tools. If there is plenty more that could be listed, the one area that most noticeably lacks a single N-LLS solution is systems management. But even this is well served, albeit with work. Those involved with the evolution of computing 'Grids' have invested massive efforts into many flavors and dimensions to systems management. With a bit of work, these could be harnessed for managing business workloads.

Analysis

C3BC is not suggesting that introducing, let alone implementing, all of the above is simple; it is not. Serious, sophisticated integration efforts are required. But, larger enterprises wishing to break the stranglehold of LLS (Linear Licensed Software) may be prepared to invest in putting everything together. Why? Because, once done, it is reusable, repeatable plus scalable and elastic. In essence, there are multiple long-term attractions – not least of which are hugely reduced infrastructure acquisition and operation costs that will continue to pay back initial investments for years to come.

This raises many possibilities. While the likes of IBM, Sun and Oracle could all deliver a complete N-LLS solution, their existing software revenue streams (a.k.a. self-interest) inhibits pro-active involvement. Instead, C3BC expects several system integrators—or even innovative specialists—to start 'productizing'

and promoting implementations of combinations of N-LLS. The result will, in effect, apply commodity principles to infrastructure software. If or when that happens, enterprises will be able to obtain savings that in the past seemed impossible. Traditional software developers will need to concentrate on business applications (where the real value lies for enterprises) rather than on the infrastructure (where the money has all too often seemed easier).

Need a parallel? Try shipping containers. Before the container was invented, ports needed armies of stevedores manually to load and unload ships. With the adoption of containerization, the focus switched to speed of transport between ports and what went into the containers – not on the loading and unloading. In due course, the same occurred in the airline industry, which has its own specialized container variations. Now even commodities like scrap metal travel in commodity containers. Expect a parallel to apply to much of the linear licensed infrastructure software that businesses depend on (and pay for) today.

Charles Brett
President, C3B Consulting Ltd.
(www.c3bconsulting.com)

ADAD: eliminate application ignorance, reap benefits plus avoid costs

One of the continuing oddities of enterprise IT is its ignorance about what it possesses at the application level and how this works in practice. For years enterprises have exploited systems management to build sophisticated and well-documented hardware and network topologies that describe what is happening at the lowest levels inside IT. But does the same exist at the application level?

Rarely, in the experience of C3BC – even though the consequences can be profound, as well as profoundly costly. One way to verify this is to talk with enterprise architects, those highly skilled individuals inside enterprises with the deepest, most sophisticated and often most expensive understanding of how IT fits together.

The alarming truth is that these Enterprise Architects will, if only in confidence, admit that they do not have a comprehensive picture of how all their organizations' applications and resources (like databases) fit together, never mind how they depend on each other. In one sense this is not surprising, since IT has become ever more sophisticated and complex. Arguably IT is now too complex for 'manual understanding' – there is just too much to find out, never mind to absorb or document.

But this lack of a coherent or consistent picture of how applications interact and work together has serious implications. It means that, at the business level, there is a yawning hole where data does not exist or is incomplete or is running behind the applications (that it is meant to support). One organization to which C3BC talked even admitted that its — then manual — procedures to update its application topology typically ran at three months or more behind reality. This hardly encourages confidence if a problem occurs.

What can go wrong? Consider some real-world examples:

1. In one enterprise, various client databases and the applications running against these were supposed to be separate; they were – except for the stored procedures libraries, which were shared. This meant that a change to one stored procedure by one developer might affect another application in a quite unintended (and very difficult to trace) way. Or,

2. There was the government department which could not switch off many dozens of elderly systems – because it was not sure what other applications or even other systems depended on these. Or,

3. The e-business which crashed because its website depended on a long established database connection that had never been documented. When that remote database was upgraded, the connection disappeared, bringing down the customer-facing website. Or,

4. The international consultancy that found it had no less than 30% more applications (and servers) in each of its three major data centers than it thought it had.

Analysis

While there are many such horror stories, all are illustrative rather than focusing on any single issue. Three other aspects are of primary importance. The first is that when application dependency problems do occur, they are extraordinarily difficult to explain, never mind cure. The second is the expense and business risk that enterprises incur because of their ignorance. The third is that due to that ignorance, many enterprises do not even realize they operate on, at best, a 'wing and a prayer.'

What compounds the problem is that we (the technology industry and business customers) had, in Y2K, the opportunity to remove uncertainty, because IT had to comb through all its applications for possible date problems. Rather than build on the base of understanding established, this knowledge was allowed to go stale. (When everyone believes information is stale, it is ignored and rarely ever updated again).

Today a different approach is needed – one that takes advantage of automation to provide full application and dependency mapping for enterprises – on an ongoing, constantly updated basis. C3BC describes this as ADAD – Automating the Discovery of Applications and their Dependencies (others call it application discovery or something similar). For enterprise architects, as well as operations, this process has become fundamental and essential. Enterprises must know what they have in order to make decisions (or they risk making bad decisions and breaking what currently works):

- The good news is that there are decent automated tools available from the likes of CA, BMC, EMC, HP, IBM and Tideway — to name a few; heir tools have a systems management pedigree. But, in the view of C3BC, they should be exploited not only by operations but also by enterprise architects and by management.
- The bad news is that most enterprises are now far too complex for manual tools; automation of the application and application dependency environment is a business requirement as important as systems management.

Beyond the innate problems this situation can cause organizations, failure to possess detailed and accurate application/dependency information may also run afoul of Sarbanes-Oxley (and foreign equivalents).

In the view of C3BC, management needs to back enterprise architects to take action, not only to provide good 'housekeeping' but also to avoid failures which could provide good cause for accusations of mismanagement and negligence. If sustained, such accusations could all too swiftly become alarmingly expensive.

Charles Brett
President, C3B Consulting Ltd.
(www.c3bconsulting.com)

IBM's InfoSphere Streams is a game-changer

IBM has made it clear that InfoSphere Streams, the new commercialized offering based on the 'System S' research project that it has had underway for some years, is a priority. It (IBM) is committing substantial investments. In fact, the InfoSphere Streams announcement was hurried to coincide with a major financial analyst meeting on May 13, 2009. IBM CEO Sam Palmisano called it out as an IBM opportunity, and Software Group SVP Steve Mills listed it as one of four themes within his top line Information Agenda message. In addition, John Kelly, IBM SVP and director of IBM Research, was present to add that "System S software is another example of IBM helping clients through our long-term investments in business analytics and advanced mathematics."

That kind of push, involving multiple IBM units collaborating, is being seen more and more, and it makes things happen. That said, I've seen skepticism about InfoSphere Streams from analysts I respect like Curt Monash, Neil Raden and Peter Thomas. That is not surprising — there is much evangelism going on here and sometimes there is a fine line between faith and functionality. So what are the use cases for InfoSphere Streams?

Interestingly, IBM is describing them based on real, moneymaking customer engagements, and it will continue to do so. Let us look at a few things that will likely drive this technology into broader use.

The world is being instrumented – I hardly need to give you the numbers — they are everywhere. It is not just radio telescopes (although these are cool). It also includes traffic cams, RFID tags, trades on financial exchanges and transactions on electronic consumer systems. And that is only the beginning. There is a class of information for which the right answer (as with traditional relational databases) is not 'let's put it into a box and then look at it'. Michael Stonebreaker, the seminal database thinker and innovator, said as much when he started StreamBase a few years ago. That firm's lack of breakthrough success (at least, thus far) only indicates to me that it was (and maybe still is) a bit ahead of the market. We are deploying more instruments streaming data all the time and that will only continue to accelerate.

Early use cases have proved out. Admittedly, some are still literally rocket science, but in the financial industry (where StreamBase is doing most of its business) the notion that one might usefully filter and respond to things as they occur, without building a place to keep them first — or at all — makes sense. In everyday monitoring-based use, who cares about last month's traffic patterns or the shipping record for that package last month? Somebody does — but not for every application we will associate with that business activity. And certainly not for the most interesting ones.

Tools are emerging. There are SQL-based approaches, including the SQLStream's work. IBM has introduced an Eclipse-based IDE, with its language called Stream Processing Application Declarative Engine (SPADE); this includes visual metaphors that are light years ahead of the thorny approaches of the earliest uses. And, with rules engines and decision automation climbing up executives' wish lists, the combinations could be compelling. [Object-oriented databases have been, and always will be, a zero billion dollar market — because only scientists used them; they are not moving toward mainstream use.] That is simply not the case here with 'streaming'.

Complementary technologies are available. IBM is coupling InfoSphere Streams with its recently acquired SolidDB in-memory database for some classes of applications. IBM, along with other vendors, has fashioned adapters to simplify connections to existing industry-standard streams of information in finance, logistics and healthcare. Again, this should shorten the time to deliver something meaningful. When that happens, more streaming standards will emerge.

Expertise will drive proliferation. Complex software is everywhere. It creates employment opportunities for systems integrators (SIs). Who has one of the biggest and most successful SI organizations? IBM. Who has demonstrated world-class capability to replicate hardened IP based on engagements they've done a few times? Ditto. As the InfoSphere Streams flywheel picks up speed — watch out.

Analysis

If you think I'm succumbing to hype, so be it. All things considered, I believe we are at the beginning of an essentially new class of applications that will be fundamentally different from conventional database-based applications whose history dates back to the early days of punch-card systems. As each new level of abstraction emerges, we come to think differently about the world and what we consider to be the art of the possible.

If a baseball player had to compute angles and speeds and gravity and wind and the weight of his bat before every swing, how many hits would he get? And yet, without storing data and doing queries against it, he responds to events with flashes of partial data, context and some rules — and knocks the ball out of the park sometimes. Oh, he misses sometimes, too.

But that just means we need to factor in those possible errors into our thinking about what we hope to accomplish, and what is at stake. Stream-based computing will change the world, and we believe the power of IBM's InfoSphere Streams engine will be a big accelerator of that change. Plus, technology, marketing, sales and services add up to a compelling business equation. Steve Mills' organization provided 43% of IBM's profit in 2008 and his belief that, as he has told me, "Our investments position IBM to do things nobody else can do" is spot on.

Mills told the financial analysts at the May 13 meeting that he is focused on "better optimization of the business decision process and becoming more predictive." Elaborating on that, he spoke about improving federation, pattern recognition and prediction - with InfoSphere Streams as a key enabler. This is visionary, and in sync with IBM's broad "Smarter Planet" theme. It will be relentlessly sold at the highest levels of the world business community. IBM changes markets but what's happened so far has been but a prologue.

Change is gonna come.

Merv Adrian
IT Market Strategy
(www.itmarketstrategy.com)

Amy Wohl's Cloudosophy

To comprehend Cloud Computing, it helps to try to define it. The difficulty is there is an ongoing and vivid debate occurring about just this.

The notion of a cloud (as a technical concept) first appeared in communications. Every diagram of networks, particularly multiple networks, was topped with a 'cloud'. The idea was that the cloud was beyond specific definition, but that it included everything that the network might need, and that it could change over time as technology and applications evolved. In this context it was natural for the biggest and most pervasive network, the Internet, to come to be thought of as a 'cloud'.

Cloud Computing takes the idea of processing-at-a-distance (not a new idea at all) and translates it to the world of the Internet. Cloud Computing has its antecedents in time shared computing when, in order to connect any remote user to the computing environment, expensive, secure physical connections had to be created. In today's Cloud Computing round, we substitute the Internet (with or without secure connectivity) for those specific physical connections. This changes the equation enormously and makes access to the function of the Cloud ubiquitous.

In the beginning it was easy. We had a few clouds which offered access to compute power and storage at a fee, usually per user per hour. All the clouds were public, in the sense that anyone with money (or more realistically, a credit card) and an Internet connection could use Cloud Computing. But the term Cloud Computing was quickly stretched to include not just processing offerings but other services and applications (previously referred to as Software as a Service or SaaS) that clouds were 'thought' to enable.

Initially, I diagrammed this Cloud environment as a layer cake (Figure 1), adding first management and later an application integration layer (as yet largely unrealized). But after a number of discussions with Michael Salsburg, a technical executive and architect at Unisys, I redesigned the 'layer cake' (Figure 2), to represent more accurately the relationships between the various parts of the Cloud. In this diagram, Management moves from the layer cake and becomes a vertical slice affecting all other parts.

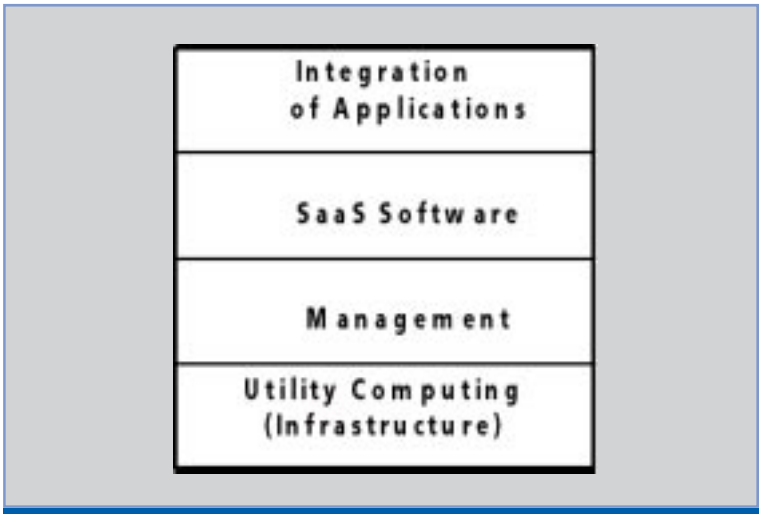

Figure 1

Personally, I used to feel it was crisper to refer to the infrastructure as the cloud and the applications as SaaS, or cloud-based software, or whatever. But it is clear that this argument is lost. everything in Figure 2 singly and in combination, is being referred to as 'cloud' or 'Cloud Computing'.

Nothing, however appealing, remains static and unchanged. Cloud Computing quickly grew in multiple dimensions, becoming ever more nebulous:

- Early Cloud Computing vendors — like Amazon and Google — added functions (often for additional fees) such as storage and development software.
- next, new vendors entered the marketplace — such as Rackspace and OpSource, each with its own focus based on its previous experiences, strengths and customer demands

- then, large customers — who were interested in the idea of Cloud Computing, but unwilling to place their information in a shared environment — wondered if 'private clouds' were possible.

It is this interest in attracting enterprise business and providing additional security and privacy that has caused much of the controversy over the boundaries and definition of Cloud Computing. Many Cloud Computing providers now offer a 'private' cloud for a customer, usually a large enterprise. Until recently, this meant building a new physical data center around a cloud architecture (virtualized servers, flexibility in scaling up and down, and the ability to support as well as connect) users in many geographies). But it was entirely private — 'owned' by the customer. [Ownership, of course, is a flexible concept itself. IBM, for example, will build infrastructure, including private clouds, for customers and charge for them on a lease or lease-to-own basis as well as selling the new facility to the customer.]

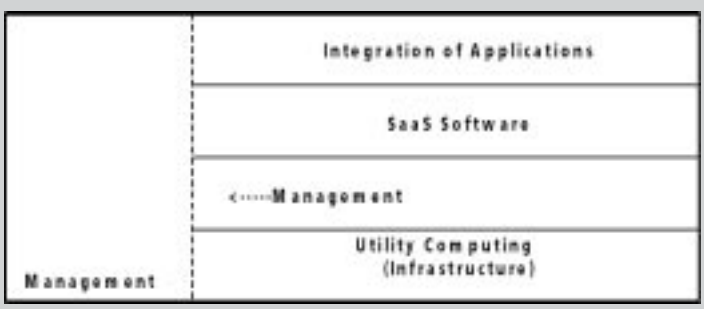

Figure 2

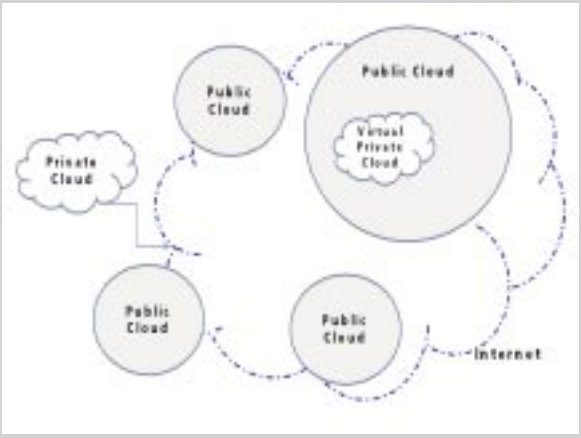

Figure 3

Although access to this private cloud would normally be via a VPN over the Internet, the cloud was invisible and unavailable to anyone other than its owner and his or her invitees. Scalability was limited to the resources of the private cloud and the customer was making an investment in capital infrastructure (or at least a commitment to it).

Recently, several vendors (Amazon and OpSource, for example) have started offering 'virtual private clouds' (or 'private virtual clouds'). In these, the customer contracts with the cloud vendor to create a private cloud within the vendor's larger public cloud. Subject to contract terms, the customer can choose to scale his private cloud up and down, enjoying the capital infrastructure investments and larger resources of the vendor, rather than being limited to his individual organizational initiative. From within the public cloud that enables his private virtual cloud, the customer could choose to interoperate with other clouds or other web-based applications (Figure 3).

And here is the root of the controversy: Is a private cloud a cloud at all?

Some claim that — since a cloud, by definition, offers the ability to buy compute power (and other capabilities and services) on demand, without the need to buy the infrastructure — a private cloud must by definition not be a cloud at all. These same debaters usually argue that a private virtual cloud is a cloud because it is simply a temporary (although temporary could be as long as the customer likes) piece of a public cloud, subject to scaling up and down, as any proper cloud computing offering is intended to do.

Analysis

It is this kind of debate that convinces me that Cloud

Computing is still pretty immature. Not that it does not work; it does. In fact, lots of customers are happily computing away in the cloud, for purposes as diverse as additional compute power, development and test environments of any required configuration, access to software more immediately (with fewer internal resource requirements than traditional), in-house implemented software and with the ability to create shared environments outside the organizational firewall, where all kinds of workers (employees, contractors, suppliers, customers, and others) can collaborate.

Rather, we are all still testing the boundaries of exactly what we would like Cloud Computing to be:

- vendors want to take advantage of a hot new marketing term
- customers bring their normal expectations to a new technology: they expect it to solve every problem.

Cloud Computing is no different than other technologies. It can offer extraordinary and growing capabilities that will make it attractive to most business customers for some things. But no technology will ever solve every problem and it is when the users recognize the boundaries of a new technology and routinely use it for its strengths that a technology begins to mature and be genuinely useful.

Amy D. Wohl
Editor of:
Amy D. Wohl's Opinions and
Amy D. Wohl's Opinions about SaaS
www.wohl.com

40GB of daily database processing — on $10/day

Management introduction

Extrabux, based in San Diego, California (http://www.extrabux.com), is an online comparison shopping engine (CSE). Purchases made through Extrabux attract a discount (negotiated by Extrabux) from stores and retail chains. In addition to finding the lowest price for a product online, by integrating coupons with its comparison shopping engine, an additional differentiator for Extrabux is that it passes the majority of the discount obtained back to the shopper. The shopper can choose to keep it or to donate it to nominated charities.

For the Extrabux service to work it must process huge volumes of data each day -- already at 40GB+/day and climbing. Rather than use a conventional Linear Licensed Software (a.k.a. proprietary) database product, Extrabux uses Non-Linear Licensed Software (NLLS) via the version of Hadoop provided by Amazon's Elastic MapReduce.

As Patrick Salami, Senior Software Architect at Extrabux, discusses in this case study, the results --- while not necessarily simple to obtain -- are spectacular in terms of volumes processed, the reduced time taken and the cost efficiencies obtained by Extrabux data processing system design; this is built entirely in the Amazon cloud.

Some background

Extrabux is a comparison shopping web site. It finds the lowest price online for millions of products by integrating coupons as well as cash back rewards into its powerful search engine. To achieve this, It has direct relationships with a number of top online retailers plus it works with affiliate networks which assist merchants in publishing their data as feeds. Today Extrabux provides access to 2,000 of the top online retailers and continues to expand its reach. It add more retailers every day.

Through its relationships with those retailers, Extrabux currently has access to a total of 70+ million discrete products for sale. The details of all these items and prices are sent to Extrabux daily as a series of

individual data feeds which need to be processed each day in order that the most up-to-date product prices and information to appear on the Extrabux web site. In practical terms this amounts to some 40GB (and growing) of raw data which arrives via batch feeds.

Unfortunately, this raw data is differently formatted by each source. It needs, therefore, to be parsed into a common format so that it can be processed and indexed in order that Extrabux customers will be able to search for and compare products and prices on the Extrabux web site. To do this, the raw data arrives at a certain time early each morning. In order to stay competitive and provide the most up-to-date data to our customers, Extrabux must complete processing and indexing all the disparate data feeds as soon as the raw data is available.

Our first imperative is, therefore, that we be able to complete all the processing in as short a time as is economically possible. To compound this challenge, the size of the data feeds and number of items may vary each day -- as we add new retailers and as retailers send us information about new products. This is highly variable:: one day we might receive (say) 60M items; another it might be 80M. Nevertheless, the window for processing never stretches; by having the most up-to-date data we obtain our competitive advantage. The Extrabux system is, therefore, architected in such a way to enable us to process a virtually unlimited amount of data in exactly the same time window.

In addition, I should emphasize that the data we receive really is raw. Product records are not pre-processed or filtered in any way; there is no sorting, no order and not even a common format across the various feeds. This means that we have to apply a series of relatively complex algorithms to the data before it is even ready to be indexed. For example, we need to categorize each product using a custom machine-learning algorithm plus we must group identical products together by a unique identifier, such as via a UPC. Prior to that, we have to make the data homogeneous which includes filling out missing fields where possible and filtering out incomplete or invalid records.

Why use MapReduce?

Although we have been using MapReduce since we started building this system, we did not consider using it as a replacement for a relational database until we discovered that the RDBMS technology solution was simply not up to the task (I will talk about why, later). The attraction of MapReduce is that it can exploit a very large number of computer nodes in which the Map stage takes input -- record by record -- and, after any applicable processing, re-organizes the records in the desired fashion and distributes them across the distributed computing nodes that comprise the MapReduce cluster.

For us the advantage of the MapReduce approach is that we can distribute our 40GB of raw data across multiple nodes for processing before it is re-assembled into a clean, structured data set that is ready to be indexed. Even though the amount of input data may change dramatically, our processing time remains constant - if more data needs to be processed, we simply add more nodes to the cluster.

In practice we use multiple different JAR files to process the data, which primarily consists of:

- parsing and filtering the data
- applying our custom categorization scheme
- calculating the applicable cash back and coupon elements
- grouping identical products together.

Amazon's Elastic MapReduce service makes it easy for us to process the data in multiple steps, using different JAR files if needed. Amazon also enables us to flex the processing power required depending on each day's input data volume.

In other words, we can add to, or reduce, the number of nodes and storage deployed each day for processing. Indeed, we now use an automated script which looks at the size of the data to be processed and, from this, determines the number of MapReduce and Solr index nodes that are required for that day. Furthermore, we only provision the MapReduce nodes until the data has finished processing and we only pay for the compute hours that we actually use.

From deploying the needed resources to ending the processing the 40GB takes about 90 minutes; of this time, a certain portion is spent writing data to S3 (storage) and other Elastic MapReduce overhead. 'Our' processing only takes about 60 minutes.

In processing terms, our normal usage of Elastic MapReduce exploits some 32 nodes, each with 2 CPU

cores (in Amazon's terms -- 'c1.medium' nodes); uses 1.7GB of RAM memory plus S3 storage for loading the 40GB of input data and for storing all the correctly formatted and sorted output data.

Why was the RDBMS approach discarded?

When we started out we expected that the traditional RDBMS approach would be sufficient for storage and ad-hoc retrieval of our bulk data (in essence, for generating metrics). We thought this because much of the processing we do is complex: there is much more than filtering and sorting. For example, we create detailed metrics about which retailers have the most products as well as the most bad product records (so we can work with them to reduce this) plus we generate statistics that look at how product data changes over time -- which helps us organize the data for our online customers. In addition we have found it necessary to define, and calculate, a myriad of other data points about products, about retailers and about customers, as well the many relationships between these -- so that we understand how to improve our customer experience.

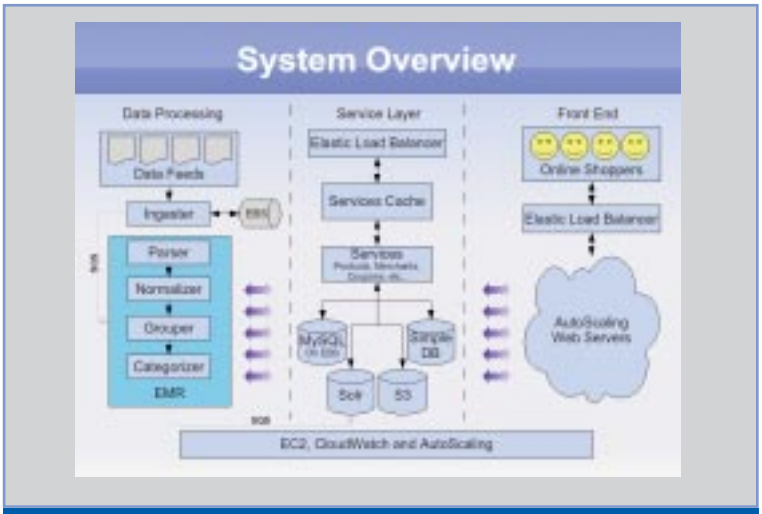

Figure 1

Initially we tried MySQL. Indeed we tried to make this work for too long a time -- because it did seem to be the logical choice.

We did manage to store our daily product data in a MySQL database with over 70m records. But regardless of the configuration, many of our queries would run for hours or even days. That loading the database with the fresh data took upwards of 6 hours, even when using the faster "load data infile" method, defined the scale of the problem.

We saw some performance gains when using MySQL Cluster within the Amazon Cloud (with 4 data nodes, a management node and the MySQL node). But, while this worked better, the latency of the disks and network in the cloud introduced variables that we could not predict. Not only was loading slower but queries still took much longer than we needed, and it was all even more expensive (because of the additional 6 large servers required), not to mention the overhead associated with setting up and maintaining a live MySQL Cluster at all times. (Trying to store 40GB+ in memory and keeping the cluster up at all times was simply not cost effective; the alternative was storing the data on disk, which partially defeated the purpose of using MySQL Cluster.)

The net result was still too slow to work with -- however imaginative we were. We knew that we needed something different. To cap it all, when we discovered that if the relational database approach was to work we would need to hire a full time (and expensive) DBA (which had never been in our business plan), we knew we had to change direction.

Making the change

Looking back, we worked really hard to try to solve the problems with a relational database (we do not think it was MySQL at fault -- using any other RDBMS would have had a minimal material difference). We already knew, from our combined past experience, that a proprietary database of the size/speed we needed from the likes of an Oracle or IBM or Microsoft would be too expensive (and still require that DBA). The decision was taken -- after learning about and taking courses on MapReduce -- that the relational approach was not going to work for us.

We realized that by using Amazon's Elastic MapReduce we could bring up a Hadoop cluster as needed not only to process our data, but also to enable us to perform metrics and data warehousing without the need for an RDBMS. The new approach works. Queries go through in minutes instead of the hours they previously took.

What makes this even more attractive is that Amazon's Elastic MapReduce's APIs let you automatically spin up a cluster on n-number of nodes, run the specified JAR files on the data and then automatically take the whole cluster back down when finished; we only pay for what we use. It got even better. We do not even have to use a conventional database to power the Extrabux web site.

After the data processing in MapReduce, we index the data using a read-only indexing system called Lucene. Lucene actually does the indexing and provides the basic query syntax. We use Solr to sit on top of Lucene to provide an XML interface (into Lucene) and so provide additional features used by our search engine, such as 'faceted search'.

When a shopper makes a search on our web site, the shopper interacts with our PHP front-end which sends the shopper's search term to Solr. Solr then sends an XML formatted response --based on results obtained from the Lucene index -- back to the PHP front-end, with the search results being the products that match the search term. If the shopper then clicks on a particular product, a new query is sent from PHP to Solr -- this time for the details of the particular product being requested, This returns the detailed product information, the list of retailers who carry that product and their respective prices, etc. Due to the large number of products that our search engine comprises, the Solr index is 'sharded' across many servers, all hosted within the Amazon EC2 cloud. Our automation framework determines how many Solr shards are needed each day, based on the amount of data to be indexed, and we deploy each shard with a portion of the data.

To support this process, we leverage other Amazon Web Services such as SimpleDB, SQS (Simple Queue System), AutoScaling and S3, which allow us easily to:

- launch the shards once the data is ready,
- let each shard find a portion of the data to index
- record each shard's status and IP address (so that our front-end knows which shards to talk to when it needs to make a query.)

The beauty of this is that we have not had to create a conventional database for our bulk data (we continue to use MySQL for small-scale data such as merchant information). Everything is handled through non-traditional means, which also happens to be both faster and much, much cheaper.

Pitfalls and lessons learned

The biggest pitfall, if it can be described as this when we have such an ever changing and growing environment, is defining and automating the data flow so that new data is delivered, processed and indexed reliably every day,. We possess a constantly evolving system. All I have described is neither simple nor smooth in practical terms. Attention is always needed; however, we have spent many hours perfecting and automating our current process and we are confident that our system will perform reliably when our new comparison shopping site goes live, even under extreme load.

Our number one lesson learned was that we did not have a need for an RDBMS to process and store our large-scale data; everything we needed to do could be done with scalable open-source solutions. For example:

- much of our data warehousing is now done using Hive, an SQL-like query language that is built on top of MapReduce
- for data analysis we found that we are able to utilize Pig, a scripting language also built on top of MapReduce.

After briefly experimenting with MySQL, we found that using interfaces to MapReduce (like Pig and Hive) meant, we were able to obtain much better results much more easily and cost effectively.

We are now in our fourth major revision of the MapReduce code -- and moving towards a service-oriented architecture (SOA). Now that we have a good understanding of how the different components in our system interact with each other (the coupons, the cash backs, the merchant information, etc.), we are starting to work on making MapReduce as well as our front-end talk to our web services. The various components of our system are nearing feature completion and we have a clear picture of how the components need to interact with each other – which is what makes the SOA approach possible, albeit at the expense of having to re-engineer some of our code.

Although we found that MapReduce is a more sensible solution for Extrabux, is still a relatively new tech-

nology and its documentation and best practices are not as fully developed as for RDBMS systems. Indeed, we had to build our system the hard way -- by trial and error. We learned, for example, that it is often impractical to link our MapReduce jobs directly to other data sources, such as Solr; we had to develop workarounds. Similarly, Solr can be 'touchy' sometimes: a small irregularity in the data can cause it to break (and its administration is less than user-friendly).

Another area we needed carefully to address when developing for the cloud is planning for node failures and building a self-healing and robust system architecture. This is not, of itself, a major problem – but it required us to think about problems differently,. We now think of each server instance as doing only one thing and as being fully self-contained so it can be re-launched automatically in case it fails, without disrupting the system. In addition, we needed to design our system to utilize the underlying Amazon Web Services that are available in addition to EC2 --such as S3, Elastic Block Store, SQS, SimpleDB, Elastic Load Balancing, Auto Scaling, etc. That said, bandwidth between nodes and between different services such as S3 is a continuing source of challenge and requires great care and planning, otherwise jobs can take far too long to process.

Yet, for $10/day for all the processing described, the pitfalls and lessons learned seem a good trade-off.

So I guess my final lesson learned is that, even if you do have 40GB/day of data input to process, you do not have to spend a fortune on traditional software. There are alternatives available, especially when working in the cloud, that work much better and are more flexible and cost-effective if you are able to invest some time in building a custom data processing solution.

Management conclusion

40GB of data processed daily (from load to indexing of 70M constantly changing input records) for US$10/day is astonishing value. That is less than US$5K/year – a trivial % of what a traditional RDBMS would have cost for maintenance alone.

Extrabux demonstrates how powerful and effective Hadoop is. It also confirms that the N-LLS approach is valid and why it will be increasingly attractive.

Patrick Salami
Senior Software Architect
Extrabux
(www.extrabux.com)

INSIGHT-SPECTRA

is published and distributed worldwide by:

C3B Consulting Ltd.
19 St. Michael's Road
Winchester SO23 9JE
UK

Telephone:
+44 787 233 4000
+34 686 116 993

Skype
cccbrett

Email:
insight-spectra@
insight-spectra.com
or
charles.brett
@c3bconsulting.com

World Wide Web:
www.insight-spectra.com