NoSQL In The Wild: How We Do It at Tekpub

I've received a flood of emails since we launched the revamped Tekpub site (using Rails and MongoDb) a few months back. Many people have been surpised to find out that we use both MongoDb *and* MySQL. Here's the story of how we did it and why.

Wednesday, May 19, 2010
Let's Get This Out Of The Way First

Some people might read this as "Rob's defecting" or "Rob's a 'Rails Guy' now". If you're looking for drama and sassy posts on The One True Framework - there's a site for that. Our choice to move to Rails/NoSQL was largely financial - you can read more about that here. What follows can easily be translated to whatever framework you work with - assuming a Mongo driver exists for it.
TL;DR Summary

We split out the duties of our persistence story cleanly into two camps: reports of things we needed to know to make decisions for the business and data users needed to use our site. Ironically these two different ways of storing data have guided us to do what's natural: put the application data into a high-read, high-availability environment (MongoDb) - put the historical, reporting data into a system that is built to answer questions: a relational data store.

It's remarkable at how well this all fits together into a natural scheme. The high-read stuff (account info, productions and episode info) is perfect for a "right now" kind of thing like MongoDb. The "what happened yesterday" stuff is perfect for a relational system. There are other benefits as well and one of them is a very big thing for us is

We don't want to run reports on our live server. You don't know how long they will take - nor what indexing they will work over (limiting the site's perf). Bad. Bad-bad.

The Verdict It works perfectly. I could *not* be happier with our setup. It's incredibly low maintenance, we can back it up like any other solution, and we have the data we need when we need it.
Dramatis Personae

The actors in this play are making their way to the stage now... as the curtain rises...

* Ruby on Rails 2.3.5
* Ruby 1.8.7
* MongoMapper gem from John Nunemaker
* MySQL 5.1
* DataMapper gem
* ... and of course MongoDb

The server that we're running in Ubuntu 9.1 up on Amazon EC2 - large instance with prepay.
The Application Theater

If you look over Tekpub you'll notice it's not terribly deep, in terms of "programmable concepts". We have all of 6 classes that we store as documents in MongoDb:

    * Account - the thing with the subscription info and who you are
    * Coupon - we hand these out from time to time
    * Production - one of our core classes: "Mastering ASP.NET MVC" for instance
    * Episode - a Production can have one or more of these: "Episode 6: The View", for instance
    * Order - this is a "shopping cart" if you will, not a historical record
    * Payment - a temporary holder class for pings from PayPal (yes, I know we need to get away from them)

Classes using ActiveRecord are typically spartan - letting the database define their structure. For Java/C# devs this might be weird - there are no properties:

```
class Production < ActiveRecord::Base
  #validators, etc
end
```

With MongoMapper, this is done differently - you specify the keys explicitly:

```
class Production

  include MongoMapper::Document

  key :title, String, :required => true
  key :slug, String, :unique => true, :required => true, :index => true
  key :description, String
  key :notes, String
  key :price, BigDecimal, :numeric => true
  key :released_at, Date, :default => Date.today
  key :default_height, String, :default => '600'
  key :default_width, String, :default => '1000'
  key :quotes, String

  #royalty info
  key :producers, String

  timestamps!
end
```

For a lot of .NET/Java devs this will look "messy" - you shouldn't elevate "data concerns" into your model. This argument makes good sense for a large, complex site - that you're building in C# or Java. Typically Ruby focuses on the straight, narrow path and with that comes a dramatic turn towards "doing what you need to do... and no more". This resonates with me - and moreover if I had to change later on it's just a function of changing a few bits of text (you'll see that in a second)

If you wanted to do the same thing with C#/.NET you can - but you don't need to use specific wording to get it to work. If you use NoRM (the MongoDb driver that I lended a hand to) - it translates simple objects into documents, so you can rest easy :).
The Reporting Theater

We use DataMapper for our access to our relational system: MySQL 5.1. We thought about going with

PostGres but decided we (James and myself) knew MySQL pretty well and since it wasn't doing any serious heavy lifting - we could just leave it at that.

DataMapper is appealing to me as I hate migrations. Rails has such a fluid flow to it - from testing with Cucumber, tweaking code, altering your views a bit - all of it is so fluid and simple that when it comes time to alter your database... well it comes to a screetching halt.

DataMapper solves this by a handy feature called "auto_migrate". You can fire this into your console, or if you're using Vim:

```
:! DataMapper.auto_migrate!
```

The classes look a lot like MongoMapper (indeed I think they were the inspiration - given the "Mapper" suffix):

```
class OrderLog
  include DataMapper::Resource

  property :id, Serial
  property :order_number, String, :required => true
  property :total, BigDecimal, :precision => 10, :scale => 2
  property :paypal_user_name, String
  property :transaction_id, String, :required => true
  property :slug, String
  property :month, Integer
  property :year, Integer
  property :created_at, Date, :required => true, :default => Date.today
  property :user_name, String, :length => 500
  property :gift_code, String
  property :description, String, :length => 1200, :required => true
end
```

"Serial" is a way to tell DataMapper to make this an Identity column (with auto-increment) - also notice the specific instructions given for how the data should be stored. I tried to do this with SubSonic (SimpleRepository) and I got pretty close to this functionality.

Something to note here is the denormalization. People familiar with reporting and data warehouses are used to seeing columns spread out and denormalized - the feeling here is that the relational system already dealt with the ACID stuff, and the ETL process (extraction, transformation, and loading) process squeezed out bad or invalid data. There's no reason to keep things in a relational structure when they're living inside an analytical one - which we are now in.

Specifically: note the "month" and "year" breakouts. This is very common in an OLAP scenario - some engines are smart enough to parse this for you, but on larger warehouses if you take the time to split these during ETL, you'll save your self a couple of hours during the processing of the cube.

In fact right there is the whole reason for denormalizing analytical systems: speed. People hate waiting for business reports and that waiting can cost you money. This doesn't apply to us right now - but you never know. It's a solid plan, so I went with it:

Each reporting table should answer one or more questions on its own. If you need a join, you're doing it wrong - default to denormalization.

I know that will raise some eyebrows. To fully grok this you'll need to understand that you're writing a historical record here - a "history book" of your application that focuses on a single "aspect" of the story. In the same way that the books in your school overlapped and discussed the same subjects - your data will overlap (in a denormalized way). It's OK - this stuff doesn't (and shouldn't) change.

More specifically: Orders are orders. If you had a relational system and changed the category of a product - it would change the category of the order. That's not accurate, historically speaking. The user bought that product when it was categoryX - not categoryY and even if you *think* that it should have been categoryY - it wasn't. The good news is that if you have to change this, you can - it's just some update queries.

Deciding What Goes Where

This part might seem difficult - but it can be solved with some simple questions. Questions you should be asking yourself and your client right up front:

* How will the client know the site's working (from a business standpoint)?
* How will the client know if I addressed the business goal?
* How will the client know if users like the site?
* How can I help the client justify their changes in the future?

These questions should be asked right up front - they will pull you to the goal line. For instance - let's take the first question and apply it to Tekpub:

How do we know if Tekpub is providing users with the value we pitched to them?

Putting our business hats on we could answer that in a number of ways:

* Subscribers are signing up
* People are buying our productions
* Subscribers aren't cancelling

So how do we answer these questions in a Relational DB way? Good question! Pulling these notions apart, they resolve down to some specific technical needs:

* A table that tracks subscription "movement" - signups, cancellations, what subscription, coupons/discount, and how much was paid with the dates of each.
* A table that tracks what was sold, when, for how much, discount, coupon used, and dates

This is enough to get rolling. I have 2 tables I need to create: subscription_log and production_sale - so I open up DataMapper:

```
class SubscriptionLog
  include DataMapper::Resource

  property :id, Serial
  property :user_name, String, :length => 600
  property :created_at, Date
  property :counter, Integer
  property :description, String
  property :month, Integer
  property :year, Integer
  property :sub_type, String

end

class ProductionSale

  include DataMapper::Resource

  property :id, Serial
  property :slug, String
  property :user_name, String, :length => 500
  property :title, String
  property :amount, BigDecimal, :precision => 10, :scale => 2
  property :discount, BigDecimal, :precision => 10, :scale => 2
  property :created_at, Date
  property :month, Integer
  property :year, Integer

end
```

And there you have it. Running auto_migrate will push these tables to the database - and I'm ready to start pushing data in.

Hooking Up The Reporting Bits

There are a couple of approaches to this - and it depends how much traffic and activity your site gets. Given the example above, I have 2 choices:

* Add an entry to the subscription_log on each signup/cancellation
* Add an entry to the production_sale table on each sale (or refund)
* Run a "scraper" to pull the data on a periodic basis

If you sell a lot of stuff then it might not make sense to add a couple extra calls to the process, slowing things down. On the other hand - the reports are constantly up to date.

If you're patient (unlike me) - you could run these jobs periodically - nightly if you like - and load up your tables in a detached process. This is completely up to you.

One thing you can do with EF/LinqToSQL if you're using .NET is to override the partial methods OnSave and write out to various logs that way. This is a handy way of doing it as it keeps the hooks on your model - not in your Controller or service code.

Generating Reports

There are a ton of ways to pull this information out and show it to clients. Here are my top choices:

* Use Excel. A lot of people don't know that they can hook Excel right up to SQL Server, or to a web page! Web Reports are a hidden gem, and you can output your data to a web page (in a table) and Excel will read that right in. It will also update it when you ask it to. Your clients can then do all kinds of fun analytics with it - without asking you for specific reports. They can even run pivots!
* Dedicated Reporting App If you use SQL Server you can hook up, for free, SQL Reporting Services (which comes with SQL Express). You'll have to write the reports yourself - but it's pretty simple. There are others out there - Telerik also has a set of tools.
* Homespun. There are ton of free charting apps out there - including Google's chart control and Open Chart - a flash-based chart builder. This might be the simplest thing to do if you need to present your data to your client in specific ways. Be forewarned, however, that you need to be able to add some interactivity here - working with data will spawn questions, and keep you on the hook for constant report writing.
* Bootstrap. If you're in a startup (like Tekpub) and you need some reports run - it's often easier to just kick up a SQL tool and write it by hand - saving the query and offloading it to CSV or just printing out the results. James and I have done screenshares over Skype to scan over the data - both asking questions, both firing up the SQL (seeing who can run the query fastest - James always uses the designer surface).

Summary

If you're wanting to see examples of this in ASP.NET MVC/C# - the Tekpub ASP.NET MVC starter site is divided out exactly this way:

* an ISession specifically for high-read queries
* an ISession specifically for reporting
* an ISession specifically for Unit of Work

I'll be updating it in a week or so to add some EF love to it and also an InMemorySession you can use for testing.

Understanding that your application data story is not a consistent, homogenous thing will save you some serious pain as your app grows. You will inevitably need to add perf tweaks here and there - do that with a dedicated agent. You might also (well... actually you *will*) need to send data off to a separate reporting system at some point. You could do that on a systems level - using replication and painful vbscripting transformations - or you could ask your app to "drop the data off on the way home from work".

It's a bit of a shift from what we're used to - but it's working for me, in the wild, on a site that my livelihood depends on.

* Opinion
* SubSonic
* Tekpub
* ASP.NET MVC

* NoSql
* Rails
* General posts
* MVC Storefront
* Hawaii
* Vim
* Vim Katas

*

Jeff [Moderator] 3 months ago
On the cost factor. It seems to me like that cost for windows on EC2 is minimal compared to it

Linux counterpart(looks to me like it is the exact same price), so to me rails over .net mvc on a cost factor isn't the real question. It is really a MS SQL vs Somthing else. Which is also why I decided on mongo.

Anyway I went through the same thinking as you did, saying if i was going to mongo rails might be easier. But I just couldn't get going with rails. Which I know is weird but I can understand c#, but ruby makes no sense to me. So in the end .net MVC/mongo was easier than rails + mongo and worth the extra $.03 an hour it costs me.

Also, since I haven't had luck getting something of substance running with rails I spent some time talking to peers and reading about performance and other things, and from what I put together performance is really the big drawback to rails. So for me the .net MVC/mongo story presented the best option there.

Although performance might be relative as I see you are running everything from 1 box. But I also know that rackspace dropped rails because it was simply to taxing on there infrastructure (you can still do rails on your own server instances). And even when they did offer it they charged extra for rails. Anyway, I guess what I am saying is that I don't know that your cost comparisons were apple to apples.

I can see why you made that dicison based on other things. Like the fact that your are comfortable with rails as well, and you like the testing and deployment better. Those to me seem like the real meat of your argument since it looks like to me you could have gone .net MVC still for just about the same cost if you really wanted to.
Flag
1 person liked this. Like
*

robconery [Moderator] 3 months ago in reply to Jeff
It's easy to write off costs as "marginal" when you have a job :). It's also easy to say "I didn't find Rails that easy - so that doesn't matter".
Finally - you bring up performance which is a good point.

RE cost - I've been getting lit up on Twitter all day about this. People telling me I can't add :) which is great. Here's the scenarios:

1) Go with an ISP like ThePlanet or GoDaddy. I can pop our little app there running on a shared SQL machine. If we don't grow - all is well and we end up paying $30-50 a month. We can't host our videos there (bandwidth is 2Tb/month) so we'll need to go Amazon. We could just use S3 - but we can't stream effectively without RTMP - so that means CloudFront. We'll do that no matter what - so it's a wash.

Now all we have to do is not grow so we can stay on GoDaddy.

2) Go with Azure. Their comparable server is twice as much as EC2, and if you add SQL to it, it goes to 4x as much. No thanks.

3) Go with Windows on EC2. Yes, the server is the same for a Win license - but you're forgetting SQL server :). Yes, I can use SQL Express to run our

site and that would work - but only until our DB hits 4 gigs. It also will only use a single core so if we do get slammed at some point, we have an issue.

Many people have suggested "that's a good problem to have - if that's the case then SQL licensing should not be an issue". I counter with "let me chop off your arm - if you have enough money you can hire people to feed you anyway". The point here is "WHY?". Mongo, MySQL, PostGreSQL - these are free. And they're fast - Mongo is *really* fast. Faster than SQL server by a good, strong measure.

Anyway - back to the point - ScottGu mentioned to me that I could buy SQL Web for $30/month if I enroll in the VL program. I could up to SQL Standard at $250/mo if I needed to.

Or I could pay nothing and use PostGreSQL - forever. Hmmmm.

Right now we pay $1400 per year for our infrastructure. It's lightning fast and it's built to scale. I *could* pay the same and be locked in to lesser technology, or I could pay more for exactly the same set up. It's an easy decision.

As to scaling - ActiveRecord is usually the culprit with those issues - and developers who write bad queries (which you can do on any stack). Since the advent of Phusion Passenger and other data alternatives (like MongoMapper) - it's not an issue at all any more.

It was a shift in thinking - that's for sure. But it was enjoyable and deployment is very easy. Our tests are much better so maintenance is about 5% of what it was - that's a very big number to us :).
Flag
Like
*

Jeff [Moderator] 3 months ago in reply to robconery
I understood everything you were talking about, and wasn't trying to say you can't add. I do see the major difference is SQL, which is why I choose mongo db myself. I went asp .net MVC/mongo because I was able to use it simpler than trying to get rails up. Believe me I tried to go rails, as I have been hanging around the drop.io folks a lot lately and liked some of there stuff. In the end C# was still easier for me. I haven't written off rails completely I am just not comfortable with it yet.

I understood why you went with rails with the whole cucumber story, and other things you talked about.
My point on cost was just that if you went with an EC2 windows instance you could have still used mongo and had very close to the same price. I was just trying to point out that onthe microsoft stack it seems that in my experience that MsSQL is the culprit for price. But since I had already made the decision no to go SQL anyway, cost was less of a factor(it only added $21.00/month for my scenario).

I am glad to hear that you have found that the majority of performance problems with rails are with active record, makes sense and I will be cautious about that as I move forward learning rails in the future. If I feel that at some point in time I can write ruby code that is as good or better than what I can do in c# (this is me I am talking about, not trying to say c# in general = better) and I wanted to play around a recode my app, then I would do it to save that $21.00. I do not believe in paying more than I need to be.

I just don't know that saying the choices were $1400 or $20,000+ was completely fair. I guess if the only options were to go all Microsoft or no Microsoft.

(Edited by author 3 months ago)
Flag
Like
*

robconery [Moderator] 3 months ago in reply to Jeff
Sure - I know you weren't suggesting I can't add :). I do appreciate the discussion...

RE Windows on EC2 - our thing was that we needed 2 data systems: reporting and app. We could go with MySQL/PostGreSQL on Windows, but the driver story there is horrific. We could have also gone MVC on Windows but the deployment story there is less than exciting.

MySQL/PostGreSQL flies on Linux - much much faster than on Windows. Ubuntu is also free - so there's no upgrade concerns that way when the new not OS comes along.

Going with Rails allows us a lot less friction in the thing we do most: maintain our site. Small changes are incumbent and often, and you wouldn't believe how many times I overwrote Avery's stuff :). With Rails and Capistrano it's amazingly easy to deploy - it's one line "cap deploy" and SSH/Git takes over and ... well it's drop dead simple.

So - RE the math :) - here's the thing:

1) Every framework/server we have is free. So the cost there is 0. I do have to pay Amazon to host it for us which comes out to exatly $1424/yr.

2) If I go with the same setup on the MS stack - the problem comes in with SQL Licensing. Now I could go SQL Web (after joining the Volume Licensing program - which means I have to buy 4 more products) and I get to spend $3200 for SQL Web (and this isn't counting the other 4 licenses I'd have to purchase to qualify). This works great until it doesn't - SQL Web is one step above SQL Express in terms of limitations. If our site grew to the point where I'd ran up against a limitation - than the $7171 per processor kicks in. Quad-core machine - there you have it.

So - the choice boils down to, literally, do I spend $1400/yr, or do I roll the dice with MS licensing and see if I can avoid the Big Fees?

Now I know a lot of people might see this as anti-MS and that's OK I spose. It's why I want to focus on the math - I literally am scrounging every penny and even a discussion of "well - it's ONLY an extra 200/month" is really a deal-breaker for me. I'd love it if MS could understand the startup mentality a bit better. Really.

I do appreciate the thoughtful discussion - we did look into this a lot and believe me I understand the ramifications of my choice (given my role in the MS community). The thing is that I'd like the discussion to turn from "what Rob can do to work with the MS stack" to what Microsoft can do to help out people like me.

And no - BizSpark is not an answer. We're still on the hook for licensing :) even if it is 3 years out.
Flag
Like
*

Jeff [Moderator] 3 months ago in reply to robconery
If your experience with MySQL/PostGreSQL on windows is really that bad than I can see you how you feel you always have to throw that MsSQL license into the mix. In my experience I haven't had any real problems with MySQL (well on a small scale, and it seems to me that just for reporting it would be more that adequate). But to admit I don't do alot of performance measurements with it on Linux so it might be way faster there, All that I know is that it has works for me when I need it.

I do think that based on all the things you have expressed, you made the right decision for you and tekpub. And I would love for Microsoft to make this debate a easier and give us something free to use for persistence that doesn't have limitations. Myself, I am excited to see ravenDB turn into somthing big for that reason, but it doesn't solve the SQL end of your equation. MsSQL doesn't even land on the table when we talk about developing a new app anymore(unless the client has already payed for it, or we know that SQL express will always work). And your right BizSpark is not an answer, but it is a start.

There is alot that Microsoft could do. VS professional(at least) should be free, MSDN should be free (it shouldn't cost us to make our Dev environment, and test environments). And it shouldn't cost so much to certify a product. I myself am wondering were I will be when BizSpark ends on this respect. I would also love them to pick up mono or at least put some of there resources behind it so it is up to the current framework release and feature set. Because I do like .net.

On another note, this doesn't really apply to web stuff. But I would love to see MS do away with CAL's. If you buy Windows Server and have a desktop with a windows OS you SHOULD NOT need a CAL to access the server. I can somewhat see the understanding with Exchange(maybe), but the OS CAL's need to go.

It seems to me that MS is starting to listen to us little people :) and I thought you were a big part of that when you were on the asp.net team. I do hope it keeps going but can understand why they need to take baby steps as they can't just give away all there revenue generators.

(Edited by author 3 months ago)
Flag
Like
*

synapse [Moderator] 2 months ago in reply to Jeff
The question is "why bother"?
What value do MS products add to your project? For most web projects the answer is "none".
Open source frameworks like Django or Rails are more mature than ASP.NET MVC (just one word - "caching"), SQL Server costs a ton (a significant factor if paying for it off your HELOC) etc.

As for Ruby being incomprehensible, well, it's not Haskell, it's yet another dynamic language. Did anyone have any problems with PHP or JavaScript?
Flag
1 person liked this. Like
*

Jeff [Moderator] 2 months ago in reply to synapse
Well for me the value I get from .net is familiarity. I can get something done rather quickly and it will be very performant. I also like visual studio as an IDE. As I have said in my previous posts MsSQL is to expensive and I really stay away from it. Plus I think the .net framework is actually really nice. and yes rails is a more mature MVC framework, however, .net MVC still sits on top the entire .net stack and is moving along nicely.

Also my comment on not being able to understand rails had me at its subject. I am just not as comfortable with it yet, and find myself having a hard time with its syntax (probably because I have been looking at C# for as long as I have been programing) , I am looking at getting more familiar with Ruby and once I do perhaps I will start building some projects out with it. Until then for me, and other who are in the same boat as me .net works and can be deployed inexpensively (As long as you are willing to hook up non-MS alternative data stores)
Flag
Like
*

cowgar [Moderator] 3 months ago in reply to robconery
We could have also gone MVC on Windows but the deployment story there is less than exciting.
well, last time I checked it wasn't that bad, but I do get that rails is better (but really not a deal-breaker here)

We could go with MySQL/PostGreSQL on Windows, but the driver story there is horrific.
wow, care to elaborate? I know "freelance" postgre driver motivation (there is another one which is commercial), but there are lots of folks which use .NET MySQL driver without any hassle...moreover you've just needed the DB for reporting (MySQL isn't any enterprisy stuff anyway) so no high-end interstellar ADO.NET things...

than the $7171 per processor kicks in. Quad-core machine - there you have it.
I don't know if I got this right, but last time I checked it doesn't matter how many cores are inside your CPU. You're paying licences just for the CPU not per its cores (doesn't matter if it is 2 core or 6 core Xenon). I know what you mean thought, it is sick price even per 1CPU...

what a great article and discussion, the other day I was looking on rails I found out Heroku... have you considered it in your options against Amazon? What were the limitations?

Thanks
Flag
Like
*

Tim Heckel [Moderator] 3 months ago in reply to cowgar
Yes, good question -- Rob, I'd like to know if you ran into Heroku at all in your research? It looks like it abstracts lots of the infrastructure issues away from development.

(Edited by author 3 months ago)
Flag
Like
*

robconery [Moderator] 3 months ago in reply to Tim Heckel
Yes - we looked at Heroku and it's a great service - especially for getting off the ground (free!). The elastic scaling is great too. The deal is that they run on top of Amazon and are a sort of "middle man" which is perfect if you don't know/want to deal with Amazon. Also - at the time - they didn't have Mongo integration, so it didn't work out for us.

James knows Amazon really, really well and has been able to manage our stuff quite easily. In terms of Ubuntu and Linux management - we both had some learning to go through but it's incredibly simple stuff and there are resources all over the web if you need to do something.

For instance - installing FTP. If you want FTP on a win box you need to go through the server setup dialogs when you're terminaled in, and then you need to configure it through IIS.

With Ubuntu:

```
sudo apt-get install vsftpd
```

That installs an FTP server for you, and then you vi into the config (which is a text file):

```
sudo vi /etc/vstftpd.conf
```

Then restart

```
sudo /etc/init.d vsftpd restart
```

We were already rolling with a media server on Amazon anyway - so for us it was just a matter of kicking up another instance :).

Flag
Like
*

cowgar [Moderator] 3 months ago in reply to robconery
it's pretty new thing (beta was released at the end of april when tekpub was already heavily running) , but Heroku is "mongoDB capable"...
http://blog.heroku.com/archives/2010/4/30/mongo...

actuall docs
http://docs.heroku.com/mongohq

times to learn rail and forget c#? wow scaaaary :)
but at least you opened the eyes for me...
I'm not amazon experienced guy (did little azure though) so I guess heroku comes with some help here, at least for me.

hope MS and Azure is reading your blog as well, come'n, don't let me leave .NET world MS, do something ;)
Flag
Like
*

robconery [Moderator] 3 months ago in reply to cowgar
Actually the deployment story is a deal breaker. FTP is not acceptable anymore, and i routinely. Overwrite changes that James made because either he forgot to tell me, i forgot to synch source, or i was sleepy. In startup land you don't have time to focus on the mundane... I had to generate content and more than once i brought the site down with a bad push.

The MySql driver issues (mysql connector.net) are legendary. The driver tries to manage connections for you and will often just time out and fail. If you run a benchmark test between ado and SQL and mysql,you'll see a 10x slowdown. Mysql itself runs a hell of a lot slower on windows too. I haven't tried postgresql in terms of benchmarks and i know i could buy other drivers... But why?
Flag
Like
*

Karl Seguin [Moderator] 2 months ago in reply to robconery
Few years ago we ran extensive performance tests of MySQL on Windows, and there was, without a doubt, something very broken with InnoDB on windows. Performance was as much as 10x slower than on Linux. MyISAM didn't seem to have this problem. Running any of these popular OSS web tools (MySQL, PostgreSQL, Memcached, Nginx, Apache ...) on Windows is insane. If you are so afraid of Linux that you can't use it, then don't use those tools - paying for MS alterntives if they exist, or not using any when they don't, is the best solution for YOU. For the rest of us, apt-get and vim are great.

As for driver issues...I did submit a handful of patches in the day to the .NET connector to resolve a number of very bad bugs - typically around performance and threading. Looking at the source now, it looks like a lot has been rewrite. I've also used the Npgsql drivers without much problem. I wouldn't' have a hard time recommending either for production - though you may need to patch a bug here and

there.
Flag
Like
*

cowgar [Moderator] 3 months ago
sorry for not reading the article (but I've read "infoq.com" interview yesterday, but thanks for all those sweet posts you're producing lately) and asking straight away, but got this idea when I woke up this morning ;p

is it possible to do some tekpub series on just released RavenDB with Oren, which would be a bit more advanced and audience would be .NET crowd?

or 2nd thing, as a .NET developer I still see no need for Ruby, as somehow I'm comfortable with MVC2 and all those zillion libraries available, plus working with MongoDB with NoRM is not a big hassle, another take on Docs DBs in .NET world (tekpub?)

is it just me or is Ruby syntax just too strange for C# guy? =)
Flag
1 person liked this. Like
*

chanceusc [Moderator] 3 months ago
Rob,

I read over the interview at InfoQ and I'm curious as to why the price figure came out to be so high? Azure offers the same service as EC2 at about the same price point.

I'm building a new SaaS app and on the .Net stack and I'd love to take advantage of your number crunching (if possible).

Thanks
Flag
Tim Heckel liked this Like
*

robconery [Moderator] 3 months ago in reply to chanceusc
Our setup initially was on a quad-core Win2008 box up at Maximum ASP,
running on SQL 2008 standard. The costs were covered by BizSpark - so all
was fine that way.

The key to this is projecting 3 years into the future and what you're going
to need. We didn't think we'd need SQL Standard in 3 years and that we could
keep with SQL Express, but that seems more than a little ridiculous - our
entire app running on a hobbled DB. There's no reason to do so when there
are a number of free alternatives - MongoDB for one, MySQL/PostGresql for
others.

Also, it's entirely likely that if we wanted to stay on SQL Server we'd need

to pop for the full standard license, which is $7171 per processor (we don't
get the web option as we're not an ISV). Our machine has 4 processors.

The choice boiled down to: SQL Express, $30,000, or Mongo/PostGres - which
are both powerful and free. There really is no choice in this regard.

As to Azure - we could have used it but we would then need to also get cloud
licensing for the OS, and pay bandwidth for video delivery. We could go
Amazon for Video delivery - but now we're split over two Cloud services,
which doesn't make any sense really.

The cheapest, most effective way to do this is with a free OS running a free
DocDB and a free RDBMS with a free framework - paying for bandwidth and
liquid scaling (which amounts to very little per month).

What's foremost in my mind is "lockin" - and I think that in many scenarios
that lockin makes sense - IF you flex what the framework is good at. In our
case we weren't flexxing the stronger points of the Windows system -
collaboration and intranet based stuff that corps might dig. We're a
money-grubbing, penny-pinching startup and the word "FREE" keeps us in
business :).
Flag
1 person liked this. Like
*

Siddown [Moderator] 3 months ago in reply to robconery
About a year or so ago Karl Seguin compared Biz Spark to a Sub Prime Mortgage, and I don't think he's far off. MS, SQL Server in particular, starts to get very expensive when you actually have to foot the bill.

SQL Enterprise addition (if you choose to go that route) is 100k for a quad core, and that just isn't reasonable when you can get the same benefits (or close enough to it) for free from the MySQLs and PostGresqls of the world.

1200/yr to run a business is just insanely efficient. Yes, that doesn't include streaming fees, but streams are generating income anyways (for the most part since I know you have free videos too), so paying for them really isn't a problem.

The interview was definitely a good read, at this point I really can't see why a startup would go with Microsoft, or Oracle for that matter, when there are so many other options out there.
Flag
Tim Heckel and 1 more liked this Like
*

chanceusc [Moderator] 3 months ago in reply to robconery
Rob,

I'm not sure what you mean by having to buy a cloud licensing for the OS. I've been under the impression that Azure bakes the licensing fee of Azure into its pricing model. I just did a quick spike on

Azure's pricing structure and it came out to be $21,915.00 a year but that's using their little tool that got very ambitious for you. It priced out 3 azure instances all running 24/7/365 with inbound @ 2.5 gb/hr and outbound @ 12gb/hr

Additionally, as far as I am aware, you can use MongoDB on Azure. Then all you would need is a small SQL Server instance for your reporting.

(Edited by author 3 months ago)
Flag
1 person liked this. Like
*

robconery [Moderator] 3 months ago in reply to chanceusc
We looked at moving wholesale into the cloud (which is what we did) - which is supposed to be all about mitigating traditional ISP scaling issues - WindowsAzure for running our site, SQLAzure for running our DB - not using AppFrabric at all. Your calcs are correct - it's 1800/month which is more than a bit ridiculous.

EC2 costs us a lot less than that (for the server). Mongo/MySQL is free. I think our total yearly cost is around $1200 (not including stream fees).
Flag
Like
*

chanceusc [Moderator] 3 months ago in reply to robconery
Thanks man, you've definitely given us something to chew on. I really didn't expect to see such a huge difference in pricing between Azure and EC2... *blows dust on the Agile in RoR that's been sitting on my desk with a "Please Read" sticky flapping sadly in the breeze*

Actually, I missed the "not including stream fees" - I see why the price difference is so significant. I guess the major advantage is the streaming on demand and not using a dedicated VM/partial usage of a VM?
Flag
Like
*

Tim Heckel [Moderator] 3 months ago
Thanks Rob -- I figured as much....I just started trying out specflow and so this one jumped out at me...thanks for clarifying :)
Flag
1 person liked this. Like
*

valmont33 [Moderator] 3 months ago
Thanks for the reply. Any change at a MongoDB/NoRM TekPub series? I also noticed in the sample app that there is just a single MongoSession that implements the ISession interface and no MongoReadOnlySession implementing the IReadonlySesstion, is there a reason for this?
Flag
Like

*

valmont33 [Moderator] 3 months ago
In an ASP.NET MVC app would you go with MongoDB or DB40? If I have a site that has 10's of thousands of rows of data (including images) do you still recommend not using a relational DB like MS SQL or MySQL?
Flag
Like
*

robconery [Moderator] 3 months ago in reply to valmont33
Mongo would work that really easily I think.
Flag
Like
*

openarmssoberliving [Moderator] 3 months ago
Can I accomplish all this (dev) with a PC or do I need a MAC???

(Edited by author 3 months ago)
Flag
Like
*

chanceusc [Moderator] 3 months ago in reply to openarmssoberliving
Yes, you can develop with a Windows box. Ruby runs fine on it (infact, you can use the .net framework but the ruby/rails stack is a version behind IIRC)
Flag
Like
*

josh c [Moderator] 3 months ago
So what did you do to wire up the logging? Maybe an after filter that fires off an async task to write the history record might be a good middle ground between a scheduled task to scrape and calling for each action.
Flag
Like
*

robconery [Moderator] 3 months ago in reply to josh c
For our site I just write to the DBs when the records get saved - it's fast
enough to do a simple insert of a single row, and New Relic (which is
watching the perf on our site) is telling us that all is well).
Flag
Like
*

josh c [Moderator] 3 months ago in reply to robconery
just discovered New Relic is an add-on in heroku. awesome. installed. and so is MongoHQ, but its

too limited right now.
Flag
Like
*

KevinStong [Moderator] 3 months ago
great read Rob - as usual...

Question: I'm a .NET/C# guy myself but am interested in learning Rails in an effort to sharper the saw and flesh out my understanding on asp.net mvc as much as possible (reading as much mvc as possible). In reading the article from InfoQ there was a TON of software from the open source/ruby side I'm not at all familar with but definitely interested (Pickle, Cucumber, etc..). Do you guys have any plans on doing pubs on these? in either a individual segment or a "testing and BDD in Rails" type of composite series?

I ask because some of the documentation for the open source stuff is *rough* in respect to finding/digesting. A series on how it all comes together would have high interest from myself and others I would assume. So many choices on the non-ms side of things - some giudeance here would be great instead of trying to hack through it myself...

thoughts?

Keep up the great work - Kev
Flag
Like
*

robconery [Moderator] 3 months ago in reply to KevinStong
Yes - we have a video about all of it (Cuke, Pickle, Mongo, etc) up on Tekpub that I should probably rename :) - it's "Building Your Own Blog-Rails" where I dive into building an app "today" with Rails. It's half (or less) of the cost of any book.
Flag
Like
*

Tim Heckel [Moderator] 3 months ago
Rob -- one question -- and this is based on the link to the infoQ article, you mentioned that while doing BDD, you're not mocking -- do you feel like the two are exclusive? Does BDD eclipse the need for mocking? Is mocking something that is primarily needed when implementing pure TDD? Just looking for some more clarification. Thanks sir.

Also, regarding this post: wonderful stuff on the business value of Rails. I am a late bloomer.
Flag
Like
*

robconery [Moderator] 3 months ago in reply to Tim Heckel

Hey Tim - no no - that came out rather sideways. One thing that you need to mock is the PayPal IPN - I sort of did that, but I let all the systems run wild since it all goes against a test system that you clear out anyway. I don't mind letting it hit the DB - that's accepted more in Rails than in .NET.

So no - Mocking is a good thing :)