Content block expected for the “container” directive; none found.

```
.. container:: navbar
```

Part of the challenge for a simple IT manager is to keep abreast of an ever changing landscape without falling into the twin tarpits of hype and cool. No-one can be in this business for long without seeing the next big thing whizz back past us like a dead bunny with a jetpack. But inured as we maybe to the sort of hype that got me buying textbooks on marimba and push technology in 98, it takes many years to notice the plauisbly disguised tarpits and not spotting them can waste time money and effort vanishing down dead ends.

So the simple it manager needs to have a solid evidence based idea on the major features of the it world in next decade. And what we can do about it.

Oh by the way, as i am talking about future trends, i am officially sanctioned to use the phrase ’paradigm shift’, a crime for which the gulags beckon otherwise.

For IT managers there are 4 paradigm shifts coming

- Disks are becoming tapes
- Parallel programming
- Information security is as busted as DRM
- Everything else, probably involving privacy.

Disks are becoming tapes

How big can a hard disk get? I have got several 500GB disks, and can now buy 1TB disks, which considering i remember saving up for a 16K expansion pack for my ZX80[1] is quite terrifying. However, the bods from seagate believe that, just like light waves put a limit on how small a transistor can be photo-etched onto silicon, similar electro-magnetic theories limit the size of a 5 1/4“ hard disk - but we shall reach 20TB before that becomes a manufacturing restraint.

Now for a randomly fragmented hard disk of 20TB the access time for any piece of data is quite slow compared to the total size of the disk. This ratio has been increasing for a while, and it has some serious implications. The amount of data a disk can output depends on 2 things, the *transfer rate* or how quickly it can pull data from under the head and out to the rest of the PC, and how quickly it can find the data on the disk. This last one is important. Disks can now hold 1TB and will grow to 20TB. They can sustain transfer rates of about 50MegaBytes/s, but even high end disks take 5 milliseconds to find a particular location on the disk[2] Lets say we want to pull off mp3 files, snippets of conversation from my life. At a seek time of 5ms, I can get to 200 *locations* a second. This is not the same as 200 files a second - it will be snippets of files, chunks that have to be pulled off and reassembled. Today this does not matter much, but with a 20TB disk, and we choose a file fragment size of say 5KB that is retrieved each access, I have a throughput not of 50MB/s but 1MB/s. That is 20TB / 1MB or 20 million seconds to retrieve the whole disk. That 231 days by the way

This is obviously a worst case style example, but if we continue to use hard disks as we have been in my career, then it will take that long to read a disk. Obviously madness, so when we write to the disk we will have to think about the use of the the data later on. Thats unheard of today. Its back to magnetic tapes.

We will fill these disks. Not just with cute otters on youtube, but with our lives; emails, phone conversations, videos from our camera equiped eyeglasses (and if you think that is unlikely look at urban cyclists, there is a growth in cameras recording the daily commute ready to send in for that all important insurance claim against the driver who just hit you.). We shall go back to a time when the application developer had to think where on a tape the data was stored.

So disks will stop becoming random access and more than likely be time sequential storage. Which is fine when I am looking for my phone conversation from the 22nd of August. But it has a problem when I want all the phone calls I made to my wife, in which I mentioned taking the cats to the vets. The implications for this are quite huge, programmers today live in a world of random access (this is one half of the reason for dominace of relational model). To end that will require massive mindset changes, and we programmers are by and large a conservative lot.

Parallel programming

With one massive change to the way programmers think coming, the next is a doozie.

Moores law is, for its kind, a famous dictum. It basically states that the processing power of a chip doubles every 18mths. Understanding this law is widley held to be one reason microsoft defeated lotus in the speadsheet market[#]_ - bill gates decided to go for the best new features even if it ran slowly, gambling that in time the chips would be twice as fast. Lotus meantime tried to speed up their spreadsheet in the code, and the market chose the new features of excel now running at acceptable speeds.

Moores law has traditionally been implemented by chip manufacturers like intel making transistors on silicon smaller each generation so squeezing more into a chip. As mentioned earlier, the wavelength of light is making it difficult to do this any more, so chip manufacturers are basically cheating, keeping the size of the transistors the same but puting twice as many on there. To get twice as many transistors to work really means two cpus. The problem with this is that a single cpu is in essence linear, one instruction at a time. Programs were written with this in mind, and now, what worked for microsoft in the late eighties stops. If i write my clever new, slow program, in 18 months it will still be just as slow - it works in a linear fashion, is designed in a linear fashion and so will only run on one cpu. That power doubling stops, and one cpu is idle, and next year 3 are idle, then 7, then more.

Intel have already announced a 32 core chip. Now if i take the lotus approach and optimise my slow code for multile cpus, i will run my applications upto 32 times faster than the guy down the road. I do not know what competitive advantage i can get from that, but it must exist. There are new languages designed to handle these parallel programming worlds, they are a odd and hither too little trodden world of *functional programming*, with Lisp as the poster child. However two new languages best represent the choices ahead - Erlang, which passes messages between nodes and essentially shares nothing between these cores, and Haskell which does something clever with shared memory which frankly I do not understand. And there in lies the rub, as these are new languages, new *paradigms* (there I said it), and again programmers are slow to pick up new things. Really new things that is. Eitehr way, to tackle the parallel programming world programmers will need to learn a completely nw class of languages. And look how long it took to go from C to C++ to Java, which basically was objects, then garbage collection. (about 30 years. Can we wait another 30?)

Tarpits

Here we can see tarpits forming, and it is worth commenting on their likely shape too.

there are already rumblings about Intel and others developing compilers that will parallelise your program for you, meaning the programmer does not have to think about the problems of running his app on 300 cpus in different time zones, it just happenns. Yeah, and the Royal Air Force Pig display unit is putting on a show next week too. So much effort is being expended to stop us from having to learn a new thing. Thats not simple.

Both the above lead to the much heralded Grid computing - we will treat computing like we treat electricity - ubiquitous and commoditised. I just do not beleive it. I can see how already commoditised services will be gridified - but lets face it the reason organisations use IT is two-fold: to not lose cost advantages from not doing what everyone else does (the move from letters and memos to email) and to gain competitive advantage by doing something better than others. The first one will be gridified - if your email system works like everyone elses, you do not lose out. But most of the driving force for new IT is to gain a competitive advantage - to be able to do what others cannot. And that can, by defintion, not be commodity.

Information security is as busted as DRM

I have an important database in the office, with lots of important data. If the competition finds this data we could lose sales, if the tax authorities find it we could lose the CFO, and if the press find it we could lose the CEO. So it is important not to let anyone copy it, take it out of the building. That is suddenly not so easy anymore. Mobile phones, PDAs, remote working, on the road, laptops and home offices all mean that the data needs to go to where the people are. I can encrypt the data as it travels over the network, but the only way someone can work on it is if it is decrypted at their end. This is the same as a DRM protected song can be as encrypted as you or EMI like, but at some point it needs to be played through my earphones. At that point someone can copy it, both the songs and the tax data. And lets face it if it can be copied and has a value, it *will* be copied. (The head of Deutsche Post has been forced to resign as details of tax avoiders /evaders where handed over to European authorities. Basically Germany offered a reward for a banker able to prove that germans were dodging taxes. An electronic file was duly copied and sent in)

There is an attempt to control this - Microsoft is trying the ""Trusted Computing"' approach - where the hardware is locked and prevented from running anything Microsoft (and by extension the IP owner) does not want run. However it has been a damp squid so far, and even Apple the most successful hardware-controlled-by-someone-not-the-owner approach is trying to persuade the music business to stop mucking around. And what is failing in the consumer market will fail harder in the business market - the question to be framed is ""You want me to buy a laptop that can stop someone from viewing excel data if Microsoft thinks they should not see it?"'

bibliography

http://www.acmqueue.org/modules.php?name=Content&pa=showpage&pid=43

[1] if you are a british geek and of a certain age the ZX80 will hold a special place in your heart, alongside 'manic miner' and sherbert dipping lollies

[2] http://en.wikipedia.org/wiki/Hard_disk#Capacity_and_access_speed

[3] http://www.joelonsoftware.com/items/2007/09/18.html. The eagle-eyed amoung you may note that while Joel supports my argument on moores law helping microsoft, this article flat out contradicts my idea that this time round betting on Moores law is going to lose.