Thursday, September 10. 2009

Thoughts about this DIY-Thumper and storage in general

So ... 16.000 pageviews later, i've read through several reactions to my article "Some perspective to this DIY storage server mentioned at Storagemojo". I wrote that article after reading Robin Harris' article about a DIY high density storage server. Thanks to Google Translate even those in russian language (i've realized that my french is still good enough to read through texts, albeit i have to admit that reading a text in french written by me would be an even more horrible experience than a text in english). Just a warning at the beginning. This text is 32 hard disk marketing KBytes long. 31 of the real ones

IntroductionOkay, the following article isn't really about this Backblaze device, it's more about some misunderstandings regarding storage. For every suggestion i will make in this article, i see a reason why Backblaze didn't go this way. But everyone who just looks at the price and think of Sun Hardware as gold-plated, should rethink. Because for everything Sun (or other storage vendors) did at their devices, there are several good reasons as well.

The only think i've really hated about this article at Backblaze was this figure about the costs. Many people just reposted this figure but not the statement that clearly said, that this isn't a general purpose device. Without their software this storage pod would be just an horrible example to explain how you shouldn't design storage. With their software it's may be an adequate solution for their special needs and workload, and just for their special needs and their workload. No wonder, as the just developed this software for this use case.
"Yes, i've got the point"To get this right at the beginning: I've got the point of the Backblaze storagepod. It's ultra cheap storage for an application that is capable of handling cheap storage. When you have such an application ... perfect ... use such cheap gear ... albeit i see many problems, it should work reasonably well when you really take care of many problems. If not ... well ... use some better hardware.

To make this clear: I didn't write this article to explain Backblaze that they are morons and should design better storage. This article was written for people who think that this pod is comparable to an X4540/X4500. Although Blackblaze even wrote the same, their comparision with the prices was quoted quite often on other location, but sometimes without the constraints mentioned by Backblaze itself. And this put this hardware kludge from Backblaze in the same ballpark than the other solutions from Sun, Dell, EMC et al. for many people.

I think such an pod works well with special applications, acceptable with Hadoop (albeit there isn't enough horsepower) and acceptable in regard of data integrity even for general purpose load when you use it with a filesystem like ZFS. Perhaps it's a good SLIED (single large inexpensive expensive disk). Having several RAID-6 volumes, divide it in several zvols, share them via shared by iSCSI (or another target) and build some volumes on other servers.

There is "unimportant data"But there is another misunderstanding: There is no unimportant data, data is important or it's outdated, but then it was important at a certain point of time. When it isn't or wasn't important you wouldn't have stored it on disk. You can't classify data in important or unimportant. But you seperate in different classes of needed availability. There is data which you need it at high speed at any time and there is data that you need with a low frequency and it's sufficient to have it after minutes, hours, days.

You can make compromises in regard of the access speed (DVD in the bank safe instead of the data on a 8-way mirror with devices in four locations on four continents), but you can't make compromises in regard of the integrity of the data. When you can make compromises in regard of the integrity, why did you stored it at the first place? Obviously the data isn't important to you, because you don't want it back error-free. I think everybody should keep this in mind when thinking over storage: Access time is the variable, quality is the constant. Every architecture should keep this in mind.

You might say, that there are formats that are capable to survive the loss of some bits. But while they perhaps don't need the property "error-free", you have to know that the quality of the data is decreasing so you can take action and repair the documents. We have to measure the quality of an storage solution on it's capability at least to report the loss of information. But most filesystems doesn't use checksums today. When more people would know about the risk, we would see a lynch mob with torches at Redmond, WA politely asking for an NTFS with checksums for the data.

This attitude of "hey, i can afford to lose data" leads to the problem of the digital oblivion. Can we afford to lose more information when our formats to store music are already based on loosing information? Can we afford to lose information, when we compress it and a lost bit may have a massive impact?

Suggestions:
When you want to use really cheap storage, you want software capable to live with it
When you want to use a software without this capability, don't use cheap storage

The art of sizing storage - or: Get a real server, when you want to do storage!It's a large misunderstanding, that the internal storage bandwidth in a RAID system has to designed to be sufficient in regard to your network connection. Of course, this is the bare minimum. The bandwidth in a storage system has to sufficient to enable a recovery of failed components and their data in a acceptable time.

And thats is another interesting question: How long can you allow your RAID to run in a degraded state? It's easy, the maximum time is the assumed time until enough components fail to break an availability protection. The optimum time is "as short as possible" as RAID recovery really hurts performance or it takes forever. Mostly something in between.

When you reconstruct a RAID5 or RAID6 you have to read all data on all surviving disks to reconstruct the failed device. A RAID 6 with 8 disks and 1,5 disks would lead to 10.5 TB worth of data or 10500000 Megabyte. Let's just assume that you are capable to recover 200 MB per second. 14 Hours. At 120 MB per second: 24 Hours. But that the theoretical maximum you can get from a disk. When you have load on your systems, you wont reach this 120 MB thus it could easily take some while. Just think about the point, that the data requested by customers doesn't do you the favor to be the next in line for recovery. So the disk has to do one of it's most time-costly task again and again: repositioning of the heads.

Furthermore you have to consider, that all you data has to go through the PCI busses. While i'm sure that you are able to find a layout that ensures, that there is only one disk at each SATA expander while recovery, you are limited by the four busses. With four controllers and 8 disk wide stripes, it's sure that you have on 3 of 4 controllers two reading disks. PCIe 1x busses are limited to 250 MB/s (matches the theoretically available bandwidth of 250 MByte/s on a 3 GBit/s SATA connection). The recovery of 8 disks distributed on 4 controllers would load the PCI-Bus to and with normal workload over its theoretical maximum on 3 of the PCI busses further slowing down the recovery.

This problem gets even worse with the 15 disks wide stripe configuration at Backblaze: A failure in one of those stripes means that you have to read from 14 disks the full monty: 14*1,5 ... 21 TB. 28 Hours to recovery at 200 MB per seconds. 48 hours at 120 MByte per second. But at 15 Disks. 15 Disks on 4 controllers means 14 disks reading at the same time. 4 Controllers ... this means 3,5 disks per controller. As there aren't half controllers, some will have three will have four and one two. In any case you have one controller with four running devices. And this controller will limit the recovery bandwidth limited to 75 MByte/s. 77 Hours to resync one of this massive RAID sets. Three and a half days to resync.

Perhaps that is still fine for you, but it can get even worse. Without a special configuration that costs you performance, a power failure can force you in a full resync of all disks totally swamping every single byte of bandwidth and every single IOPS for days. Because of this effect, some workarounds were developed in the past. I write about them in the section "I hate RAID5"

I think, i know why they used this extremely wide stripe at Backblaze. It reduces the amount of storage thats "lost" to the parity "disks". When you have a 9 disk RAID6-set, you have 5 RAID sets in a box, thus 10 parity "disks". With 15 disks per raid set, you have 3 raid sets, thus 6 disks worth of parity. The problem: The wide stripe sets are at a higher risks to lose one or two disks. You have to process more data to resync the disks, thus it will take longer.

Furthermore. While 77 hours with 1,5 TB may sound acceptable, what's with 2 TB disks. I assume, we will see 3 TB in 2010. 42 TB at 75 MByte/s? Come back in 7 days to see your RAID resynced. This exactly the reason, why Sun introduced triple-parity RAID in ZFS and i'm pretty sure, that we will see other triple parity RAID implementations in the near future. It allows you to have wider stripe even with larger disks without increasing the risk of loosing data in the time of the resynchronization by other failing disks. Because 7 days are quite a time to let another disaster happen.

Just one paragraph to a topic, that i find rather interesting. Interesting because everybody should know it better. There are people who consider hot spares as waste of space. They think: "Okay, i have RAID6, i can survive another drive failure. But they aren't a waste of space. They aren't even meant as a comfort for the admin. The sense of a hot spare lies in its function to shorten the window of vulnerability.

Just an example: Friday, 20:00 the last admin has left the building. The support over the weekend is remote. Murphy's law mandates: Friday: 20:00:00:01. Disk 3 in RAID6-Set 2 at Server 4 doesn't goe down with flames, but goes down equally fatal. With hot spare disk: Resync starts as soon as your system detects the failure (at least with a decent

implementation). Without hot spare: Monday 08:00 Admin reading log, more often than most people think ... 08:05 ... damned ... no spare part, which moron forgot to order a new spare disk after the last crash, Tuesday: 08:00 Swapping disks ... and the resyncing starts a few days later and take some days. I know the last paragraph is best practice and everybody should know about that ... but it's better to say this again and again. And again.

One suggestion is dependent on how massive your I/O is: Don't use this small desktops mainboards. Use something with a little bit more horsepower. There are nice boards from Tyan or Supermicro for example which provides more than just 3 PCIe boards at 1x. For example the Super Micro X8DTH-6F provides 7 PCIe 2.0 at 8x. I know it costs six times of the mainboard used at Backblaze, but for a storage system it looks like a much better alternative.

To build a storage device in spitting distance to an X4540 i think i would use 6 8-port SATA RAID cards. Something like that, although it looks there aren't many SATA cards with 8 ports without RAID. @bensummers pointed me to an controller at Supermicro ... this card has 8 ports and no RAID, but this device doesn't seem to be supported with Solaris, i would look for an LSI1068 based card). Dedicated cable SATA port for every disk, no port extenders.

Suggestions:

You want HW-RAID controller to keep the recovery from the PCI-busses
An alternative: Use something with vastly wider and vastly more PCIe ports. You don´t want small FSB-Xeon-Stuff. You want to use Opteron or Nehalem, with enough horsepower to drive those busses
You want wider PCIe than 1x or use at least PCIe 2.0
You want at least one hot spare

I hate RAID5/6!Apropos HW-RAID, there is another reason, why you want a decent HW-RAID controller with RAID5 or RAID6. Both RAID levels are susceptible to an effect called write hole. The write hole is the situation, when the data is on disk, but parity hasn't written to disk. HW RAID controller can solve this with a battery. So the RAID controller knows what I/O operations were outstanding at the time at the power outage and additionally you have the data to replay the changes. A Software RAID5 doesn't have this battery and doesn't have an idea after power-failure what it has done to which parts of the disks without further informations.

In the bad, ugly times you could start to resync your complete array after a power-failure with an RAID1/5/6. Thus some workarounds were developed: For example the dirty-region log. I think the Veritas Volume Manager started with this at least almost a decade ago and it found it's way to Linux volume Managers, too The sense of the log: The resync time can be shorten dramatically when you know which regions may have altered shortly before the crash. The problem of the log: A write in a new region, that wasn't dirty before leads to a write in the log. Obviously to generated a hot-spot on your disk at this moment. So you have to decide between a fast recovery and maximum speed.

Another problem with RAID5 and RAID6 is the partial write. When you just change a part of a stripe or don't have enough data for a full stripe you have to do partial stripe write. Partial Strip Write means: Read the parity or parities, read all data in the stripe, recalculate the parity and write the new data and the new parity or parities. The problem is the read. As i told you quite often in the past ... reads are always synchronous. And in the case of RAID5 and RAID6 you have to read, before you even know what you write as the parity needs the old data. Intel calls the effect that you can't write a single byte at an SSD (and have to rewrite a full block instead) write amplification. Perhaps you should call this effect in RAID5/6 read amplification

A way around the partial write performance hit is the usage of RAID controller with memory. You can store the partial writes in the memory. The write is completed for the OS, it doesn't have to wait. The written data waits in the cache, until the reads for the recalculation of the parity has been completed.

RAID-Z/Z2/Z3 of ZFS doesn't know this effects. Every write is a full-stripe write and there is no write hole. So to get back to this spitting-distance DIY-Thumper mentioned above, i think i wouldn't use the RAID functionality of the controllers and configure them as JBOD instead (i know it's a waste).

You want a RAID controller with memory
You want a battery or a super capacitor with the memory on your RAID controller
Alternative: You want a filesystem or software that doesn't know the write holes or partial stripe writes ... for example ZFS. Then you don't need the RAID controller.

The story of UBERYou would say: Okay, i have RAID6 in that example ... i can survive an additional disk failure. Hmmm ... not really ... because with a non-checksumming filesystem the second parity may be your last line of defense in

regard of losing data - or getting corrupt data without knowing it - while reconstruction. Based on an article of Robin Harris, i've wrote that the disk are reaching a size , where the Unrecoverable Bit Error rate starts to play an important role. To get back to the Backblaze storage server. When using 10^14 drives like the Barracuda 7200.11 the probability of a UBER is 1 when reading slightly under 12 Terabyte.

With an RAID5 you don't have a protection against such an UBER, with RAID6 the second parity is your protection. When you can check with the second parity if you've just read rubbish while resyncing. And here comes the problem: Given the same cooling, the same load, the same disk lot, there is a significant possibility that another drive will fail in this time. Furthermore even an unloaded disks goes to maximum throughput when you start to resilver your RAID. Now consider, that a server workload is an out of specs workload for a Barracuda 7200.11 or any other desktop disk, then load this disk with a recovery load. That's like pedal to the metal for a motor that makes funny noise at normal speed.

I dare to speculate, that the probability of an second failure is significant, especially when you consider the bathtub curve of disk failures. A way to get around this? Narrow stripes, better disks and fast recovery to keep the window of vulnerability small. And now think about the 21 TB in the Backblaze device. You have the guarantee to have almost two UBERs while recovering. You can't afford to lose you second line of defense against UBERs.

Suggestions:
 You want at least 10^15 disks
 You want even on this disk another layer of strong data integrity protections like checksums
 You want so called near-line disks with the capability of running 24/7

"I never lost a Byte ... but this fscking Word shredded my file" - Well,really?I find one statement especially interesting: "I didn't have a defunct disk in years". At first there is a question: What is a defunct disk? A disk that doesn't turn? A disk that turns, but just makes a clicking noise? Or is a disk already defunct, when it start to deliver errors. Just to use an analogy: Do you wait until your car throws the sprockets and its oil on the street or do you go to your preferred workshop when the gearbox starts to make funny noises, but you are still capable to switch gears.

Equally interesting is the statement of users, that they never lost files by data corruption in their home storage server. I'm just scratching the head about this statement: Do they really know? Can they really know? There have no way to detect this circumstance without checksums (And with windows and similar tools they have to do this manually via md5 for example or via ZFS checksums). We often attribute data corruption at home to the universal culprit of all damages known by its household name "Windows". But even when it's responsible for a certain amount of damages, it is not responsible for all. "I've never lost a byte in years. But this shitty Word shredded my document". Really?

Many users have the subjective security, that anything written to rotating rust, is a secure method of data storage and that may have hold as true in times where UBER was as high as today (interestingly 10^14 for desktop and 10^16 for enterprise are valid for quite a time now), the disks were small and files were small as well. But thats not longer true today. The engineers of disks should do something about this instead of increasing the density again and again. I would really like to see a cheap desktop drive with 10^16 or more.

The point that many people doesn't know about the integrity of their data isn't an insult against those people. Nobody is able to say, if the data on any given disk are really the ones you've intended to be there. Perhaps we should call this Schroedinger Hard Disk. When you don't check the disk, the data is correct and incorrect at the same time With filesystems like ZFS you are able to check this, but almost all other filesystems doesn't haven't this functionality.

Suggestions:
 You want tools like smartmontools to check your disks to preventively change your disks before they are toast.
 You want checksums for your data at filesystem level to check the complete path from CPU to the rotating rust
 You want to scrub your data with strong checksumming at a regular schedule
 You want to have enough redundancy in your data, that a scrub can repair data at risk

600$ for a disk?I just want to comment on this 600 Dollar drives. Yes .... buying drives at a Tier 1 vendor isn't a cheap endeavor (ever looked at the HP website for a 1 TB disk for a DL-class server?) . I know that. But most people wrongly assume that both disks are really the same. Perhaps they have the same construction, were assembled on the same line. But using standard bears or using premium bearings with a lower tolerance make a difference. Let's say it this way ... When you have a single customer, that buys a few hard disks in its life, damage isn't that expensive.

When you have a customer, that purchases tens of thousands or hundreds of thousands a year and has the right to reject a complete lot just because a sample of the parts wasn't in specification, you think differently about using the

cheapest in-specification part. The manufacturer of the drives knows that, the manufacturer of the system knows that and so it's perfectly possible that a system manufacturer pays more for a disk than when you buy it at el-cheapo shack in the internet.

There are agreements about an assured capability to deliver such drives for a certain time that's longer than the product lifetime of the device. Another large cost factor is qualification testing. In theory there are standards, everything should work together without any problems at top performance. Anyone in working in this chaos called IT business knows: That's theory, the practical side of life is a can full of worms.

When you sell systems instead of the bare drive, you have to check the disks in the system. The moniker "supported configuration" isn't something to annoy the admin. A configuration gets the stamp "supported" when it works in an acceptable reliable way in exactly this configuration. So you have to check this in this configuration. Sometimes the large vendors find errors in the firmware on the devices, that have to be fixed. Sometimes the reason for a delayed introduction of a device lies in the need of fixing error. When you get a firmware update from your drive vendor, you have to recheck all the stuff. And just in case you didn't know it: This costs money

This checks aren't about gold-plating the devices. It's about keeping the customer lucky and don't swamp support services with green bananas in the field. Yes, i know ... everybody can report about problems ... you can't make 100percent checks, you check against a certain set of functions. When a system fails because of a very special and rare condition, that is based in the programming or design of a part, you test it from there on against this condition. It's like building aircrafts and the investigation of aircraft crashes: After every crash there are more recommendations to design or to check aircrafts, to ensure that such an error can't occur again or only with a vastly reduced probability.

Apropos firmware: Aside of the big brown paper bag bugs like the recent one in the 7200.11 drives. When did you get firmware updates for your drives? Since we left ST-506, ST-412, the logic for controlling the disks is on this PCB beneath the platter. This is a quite capable computer (Ever heard of flight-time optimization in conjunction with command queuing? The little computer calculates if the flight time to a sector is sufficient to make a small step to the side to get this data without the need to wait for an additional round of rotational latency.) It would be foolish to assume, that the software of this computer is an exception of the dogma "There is no spoon and no bug-free software".

Why RAID when you own the application?One thing i can't understand is the usage of RAID6 altogether. A different approach of ensuring data availability and integrity sounds more reasonable to me.

Perhaps i should tell you the dirty little secret of the storage business at first: RAID is a kludge. RAID is an extraordinarily resilient kludge. The same for HW RAID controller and volume managers. There was a time, when a simple "Hello World" in a word processing file didn't take 1 MByte. The hard disk were small but they were sufficient. To make the organisation easier the computer science invented filesystems and applications as it was a little bit tedious to scratch the data from the rotating rust by hand. Those filesystems and applications used the interface provided by the harddisk drivers, an interface based on "I want to read block 32, I want to write block 423".

Later more people used the computer and stored bigger files. But there weren't larger disks. And sorting stuff on separate hard disks led to overloaded filesystems on one side and rarely used filesystems on the other side. So someone had the idea "Hey, let's use several disks for a single filesystem". Good idea, but the people who developed the filesystems said "No chance, our filesystem stuff starts to work reasonably well, we don't touch it again". Thus people thought a while and came up with a simple solution "We can't change the block interface of the disk drivers, the people at the filesystem layer wont change their code. Let's develop something, that looks like a hard disk to the filesystem and uses the normal block device layer of the disk drivers. And in between we use a ruleset to distribute the blocks".

Two things were invented with this thought: RAID and a layering a certain filesystem blatantly violated many years later. With the time the industry developed a plethora of distribution rule sets that covered the problems of the initial rule sets like "First block disk A, Second block disk B, Third Block Disk A" called RAID-0 (nice, but a single disk failure kills all data) or "All blocks on both disks" called RAID-1 (nice, hyper secure ... but damned ... you don't have a single additional bit to store things). Later those distribution layers learned to calculate and the parity was born. And the rest is history.

Why do i tell you all this stuff. Well, when you develop your own data management application, it's only a small step to get rid of block based RAID and to implement distribution mechanisms for example based on files or objects.

Yes, no volume manager and no block-based RAID at all! I don't need it! When i have control of the application, i can do anything i want to store the data. Of course you are replicating many of the tasks that a filesystem would do for you, but you would have the advantage of a filesystem dispersed over many servers with http as a communication protocol. And i

could get rid of many problems of RAID5/6. Each server has a small database on an SSD storing meta information of the content of the server.

Separating every file in chunks of let's say 511 kByte. Spread every chunk on a server by modulo the chunk number with the number of servers. Spread the chunks on the disk by modulo the number of the already stored chunks on the device with the amount of disks in the system. Replicate this block to a partner by modulo the number of the already stored chunks. Store the block in a directory with the partner servers name and disks number. Use the last Kilobyte of each chunk to store some metadata and a checksum, where partner copies are, were the last block was stored, where the next block was stored. A RAID-1 based on files instead of blocks would be feasible as well.

Distribution to the servers, is done by a few staging servers, new data gets into the system just by these servers, they encrypt the data, split it in chunks and store it on the pods. Read access is done by a different set of small servers, reading the metadata and delivering the backup chunks back to the application.

A reconstruct process just have to look on every disk if there is a directory with the server name and the disk number. You get selective resilvering en passant, as a recovery would just make a format and copy over current data.

But hey, that's was just an idea while showering. But the point of using RAID6 makes me somewhat nervous. Why do you need an additional layer of data protection, when your control the app.

Apples to Oranges - or: A PB isn't always a PB.At the end of this rather long article i want to go after the most important weaknesses of this Backblaze device, it's not technical ... it's the cost calculation:

It totally omits the costs for developing the hardware and the software that makes this HW usable. You have to factor in this software into the price. You have to factor in all the time to test components, you have to factor in all the time you will need to support your software in the future (Software is like sex. One moment can lead to 18 years support agreement)

And on the other side there is no answer to the question, how many usable capacity you have. I wouldn't use the Backblaze hardware with under 3 copies when i don't have a checksumming filesystem. Out of a simple reason: How do you decide what's the correct copy of a block? It's 1:1. Which one is correct. With 3 disk it's a 2:1 quorum. Or an 1:1:1, but then you know that you have really a major fsckup and should search for your backup And given the hardware, i would even use 3 copies with a checksumming filesystem.

With better hardisks (UBER-wise and in regard of their operational pattern), with strong checksums, checksum based scrubs on a regular schedule, better power distribution and all the stuff i described in this article, i would have no problem by just having two copies of all data. At the end i would yield 500 TB out of a PB instead of 333,3 TB.

And you should factor in, that you have to swap the disk early. The Barracuda 7200.11 is a desktop drive. Perhaps you are able to keep them within their operational specification with a few users. But i have my doubt you can control this, when you have just a few hundred users. People access this at arbitrary intervals and moments.

Perhaps the approach of building you own backup device with such desktop disks is feasible when it's really used as substitute for a tape drive, when there are only a few backup servers that stores backup files there. Then you can control the operational pattern: “Backup is allowed between three to seven AM and 5 to 8 PM”. Yes, sounds strange, but thats the implication of the specification in the harddisks user manual about it's operational envelope. But otherwise ... don't think so. You can't control the access pattern. You can't guarantee that the workload stays in the specified envelope. Using them out of their specification leads to higher stress to the components, thus earlier defects. And together with the wide stripes, with the large disks you have to stay out of the rising flank of the bathtub curve at all costs. A flank that will hit you, especially with "abused" components.

Call it overcautious, but i would introduce a swapping regime with such devices, that ensures that no disk is longer than one to two years in use. Even when they aren't defect. I've worked at an ISP and helped to build it from nothing to full operation. After that i've worked at a unified messaging service. I've learned one thing at those jobs: You can lose the preferences, you can deny the access to their mails because of a misconfiguration. But you aren't allowed to lose a single bit of unrecoverable user data (speak: mails). At any cost. Data loss is loss of reputation. Loss of reputation is loss of business. Loss of business is loss of money. Loss of money is loss of jobs. You can't lose the data, because the data is important for the customer, otherwise he wouldn't have stored it at all.

ConclusionsOkay ... now i want to close this article ... it's gotten long enough with just this few additional thoughts about this Backblaze devices and some general thought about using and developing storage. As this text is already long

enough, i will just stop now. Oh no … just one thing: I thank you in the case, you've read this text up to this last line

Posted by Joerg Moellenkamp in English, Oracle, Solaris, Technology, The IT Business at 20:12

Hi Joerg,
a very good article. I especially like the term "Schroedingers Hard Drive". Another metaphor might be "RAID is like a box of chocolate, you never know what you get".
Anonymous on Sep 11 2009, 11:36

Hi Joerg,

to say it in short: You are 100% right.

But i must admit: The day i read this article about these Backblaze storage Pods i was excited. After a shorted while i began to think about the design and i was wondering why they picked this crappy Linux MD Raid6 and other things. I could write about many examples about people using Linux MD devices and seeing their data going to hell after a while.

But its always the same: Some people never learn.
The say always: "I have no problem with all that stuff. My data is safe." They don't see that quality has its cost. Period. 5$ a month for unlimited storage in a datacenter doesn't make them think about how the provider is doing this. They believe that everything is fine. I'm waiting for the day when these people realize their mistake...

But some people do. Personally i switched back from linux to solaris on my servers only because the presence of ZFS !

btw.: Is there a way to install Opensolaris without GUI in a minimal configuration ? (Like Solaris Installcluster "Reduced networking support" ?)

best Regards

Michael H.G. Schmidt
Anonymous on Sep 11 2009, 12:41

Take a look at http://blogs.sun.com/jkshah/entry/minimal_opensolaris_2009_06_appliance there is a script that you could adapt.
Anonymous on Sep 11 2009, 14:59

Nice article explaining the many tradeoffs.

I switched my home server to Solaris because of ZFS. I've been trying to promote ZFS at work because of my experiences, but it's hard to fight the momentum of existing systems and perceptions.

If I had to stuff that many drives into that much rack space within a budget, I'd probably have to go Linux. Solaris doesn't support SATA port multipliers (yet) so I can't even test the consequences of oversubscribing the bandwidth on Solaris For my home system, that'd probably be ok. USB drives are probably OK for me. At work, no way....
Anonymous on Sep 11 2009, 17:30

Build 122 - Aug-16 SATA Framework Port Multiplier Support - PSARC/2009/394
http://arc.opensolaris.org/caselog/PSARC/2009/394/
Anonymous on Sep 11 2009, 19:53

Sorry to pounce on a single aspect of your article, but:

yes, I agree that it would be jolly nice to have something like smartmontools to monitor the status of the disks. So where did Sun hide those tools in Solaris? I know that the smartmontools are not there, so what is their replacement?
Anonymous on Sep 13 2009, 14:10

1. You can compile smartmontools on Solaris as well.

2.Large parts of an tightly integrated replacement are hidden in the Fault Management Architecture, with the framework to monitor harddisks
Anonymous on Sep 13 2009, 15:02

Excellent article

Regarding UBER... what error rate are the so-called Tier-1 specified at?

A cheap 10^15 disk is the WD Caviar Green (WD10EADS): http://products.wdc.com/library/specsheet/eng/2879-701229.pdf; just ordered 1 TB one for my home NAS. Price: 65 EUR.
Anonymous on Oct 5 2009, 19:17

SATA 10^15, SAS/FC 10^16
Anonymous on Oct 5 2009, 20:05

I found the idea of rather using file based RAID (Reliable Array of Files) fascinating.

I must admit that I blieve one should design against the know probable cause of poblems.
What are the consequence of the most common failures of a disk ?
ie, what happens when a disk fails.
If a total failure happens every now and then, then it seems a file based redundant system does not deliver more benefit than a block based.

What a file based concept deliver is limitation of loss to a file for the probably commoner unrecoverable error limited to a few bytes.
Anonymous on Oct 6 2009, 21:48

Fine article you wrote.

You wrote:
@bensummers pointed me to an controller at Supermicro ... this card has 8 ports and no RAID, but this device doesn't seem to be supported with Solaris, i would look for an LSI1068 based card). Dedicated cable SATA port for every disk, no port extenders.

Here was a discuss about a Supermicro AOC which make 8-Port SATA Port Multiplier:
http://opensolaris.org/jive/thread.jspa?threadID=66128

http://www.supermicro.com/products/accessories/addon/AOC-SASLP-MV8.cfm

It seems to be supported by opensolaris.
http://www.sun.com/bigadmin/hcl/data/components/details/3170.html

I hope solaris too but I can not verify that. Can you verify that this card run under solaris?

Thx
Alex
Anonymous on Nov 29 2009, 23:33

While I agree that comparing the Pod to X4500 isn't correct I think you are missing one of the points that the Pod makes. If you are charging $600 for a disk then I have two solutions: 1) I can buy one of yours for $600 and then put it inside a system that's built to make sure that one disk is highly available or 2) I can buy 12 $50 1TB disks and create a system that solve reliability with massive redundancy.

Every time there is an order of magnitude drop in basic drive prices there is a large shake up in the industry and in basic storage architecture. Clayton Christensen spends the majority of his three books on innovation theory talking about this exact industry.

My question is what changes with storage architecture when for the same price I'm paying for a SAN from Sun, HP, etc I can buy enough commodity 2TB drives to fill an Olympic sized swimming pool. It means you start solving problems with massive redundancy, not highly reliable hardware. "copy on write" becomes "copy everywhere on write" where "everywhere" means hundreds of individual devices. Build a storage architecture where you assume 25% of all of your devices are failing at any given moment.
Anonymous on Dec 8 2009, 17:51

GREAT blogg. Thanks Joergen.

About NAS and OpenSolaris, please have a look at : http://hub.opensolaris.org/bin/view/Main/downloads
and: http://genunix.org/
EON NAS Opensolaris might be what you are looking for.
Anonymous on May 7 2010, 01:24

Nice article, finely something worth reading.

What are your suggestions in order to make the Backblaze pod more reliable (improve redundancy, not highly reliable hardware). can you point on any specific hardware(and keep the low price)?
Anonymous on Jun 9 2010, 01:02

I know, it's been a long time since the post, but any thoughts on the new monstrosity? 135TB/pod and they seem to be even more bold and dragging others down as well. I tried to post about the UBER on backblaze blog, but I got no reply (go figure). These are massive data losing devices and I guess too few people have lost data in this way to be fully cognizant of the problems. They will not reveal how much replication they use or what they are seeing in terms of real world data loss (assuming they even know!).
Anonymous on Oct 26 2011, 16:59