Notes on RAID

By: František Ryšánek <rysanek@fccps.cz>

FCC Průmyslové systémy s.r.o.

Abstract

RAID stands for a "Redundant Array of Inexpensive Disks" - a virtual disk drive consisting of several physical disks.

The technology is not necessarily inexpensive.

Small RAID arrays are usually built for reliability. Larger RAID configurations are essentially the only way to get a higher capacity per drive than what is available on the market, given the current "state of the art".

There are caveats related to both reliability and performance that the beginner user may not be aware of. This paper aims to address those that are most important.

There are a number of stand-alone and PCI-based RAID controllers on the market. As the background technologies evolve, several corporate acquisitions have happened in the industry in the recent years. As a result, by now there are a number of venerable brands and countless different products, available and obsolete.

This introductory paper has originated as an assorted collection of notes based on practical experience and further study. It attempts to offer an overview of the rapidly evolving RAID business, focusing on the lower and midrange segment, where the users are most likely to lack deeper expertise.

The original Czech version of this document is accompanied by quick-start cookbooks to a few popular PCI RAID controllers, focusing on disaster recovery procedures, as implemented by the particular manufacturer under different operating systems.

Contents

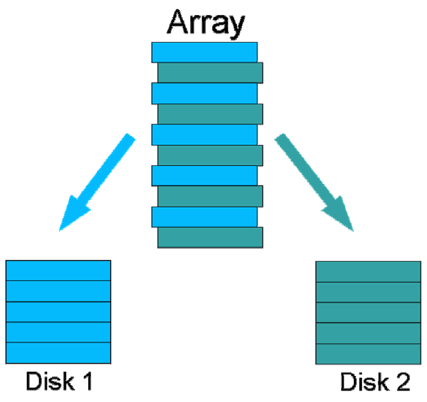
RAID levels

RAID 0 (striping) = maximum performance

Creates a virtual drive with capacity almost equal to the sum of both (all) physical drives. The Virtual drive is "cut into stripes" of e.g. 64 kB each, that get evenly distributed among the physical drives.

Upon reading and writing, this array features almost double (n-fold) performance as compared to an individual physical disk drive.

Caution, this array is dangerous! When a single physical drive fails, the array fails too, irrecoverably! No meaningful block of data can be recovered - each physical drive hold only every second 64k stripe. When a RAID-0 fails, it's as if you ran your data through an office shredder.

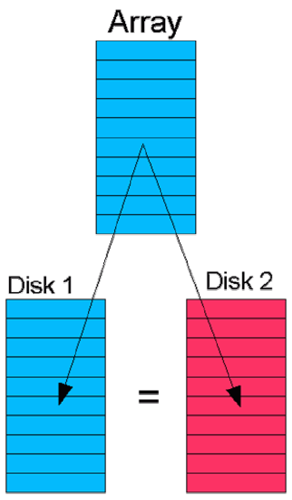


RAID 1 (mirroring) = maximum safety

Creates a virtual drive with capacity equal to a single physical drive of the pair (or one half of the total capacity of a larger set - this type of array can only have an even number of member drives).

The data residing on the virtual drive is stored twice - each of the two drives obtains one copy. There's no point in speaking of "payload" data and parity - the two copies are equivalent. When a single physical drive fails, the array "degrades" - but the virtual drive remains operational, the data is safe. The degradation can be repaired (redundancy restored) - obviously only after the failed drive is replaced.

The performance of this array is equal to that of a single physical drive.



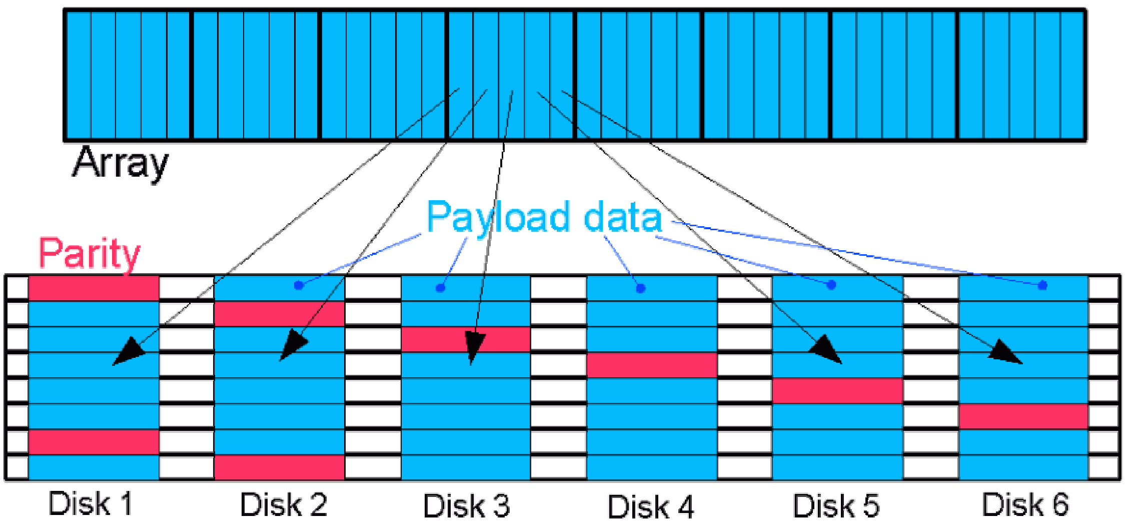
RAID 0+1 (striping+mirroring)

This array features the same reliability and capacity as a mirror, but a somewhat higher performance. That's why some modern RAID controllers cannot even be configured for a plain RAID 1 (mirroring).

RAID 5 (redundancy)

This array can be set up across three or more physical disks (N). It creates a virtual drive with capacity equal to approximately N-1 times the capacity of an individual disk drive.

The data is dispersed among the multiple drives. Simply put, the virtual drive is striped, the stripes are distributed among all the disks, and some parity information is stored into the redundant space. A single parity stripe gets calculated for each stripe set. The key idea is that within each stripe set, the parity stripe is stored on a different disk, so that there's no single parity-only drive - such a dedicated parity drive would represent a bottleneck.



This type of array features performance and capacity somewhat lower than the sum of all the member physical drives. It achieves optimum performance while storing and reading large continuous blocks of data. Which hints at an important caveat: especially when writing short blocks of data, smaller than the stripe set (also known as the "stride"), the array must read all stripes in the set anyway, in order to re-calculate and store a new parity stripe - which may significantly impair performance. This syndrome can be partially alleviated by a large cache on the RAID controller.

When a single drive fails, the array degrades, but will remain operational on the parity stripes. When more than one drive fails, the array fails irrecoverably.

RAID 6 = performance, capacity, double redundancy

This array can be set up across four or more physical disks (N). It creates a virtual drive with capacity equal to approximately N-2 times the capacity of an individual disk drive.

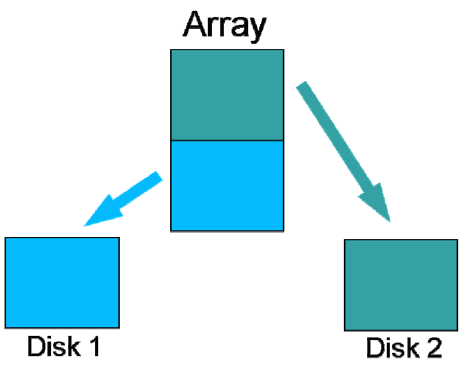
Just like with RAID 5, the data is striped and spread across all the drives.

Just like with RAID 5, parity stripes are calculated and spread across all the drives in a cyclic fashion, but, in contrast to RAID 5, *two* parity stripes are calculated per a payload stripe set – these are called P and Q. In effect, two whole drives in the drive set are sacrificed in favour of parity.
Each of the two parity stripes is calculated using a different algorithm, so that the payload can be reconstructed even in case of a *double* outage, i.e. when *any two* disk drives give up.

Please note that this level of fault-tolerance is not available from any other RAID level, not even RAID 10 or RAID 50 – these survive a lucky double outage, but not *any* double outage. I.e., with these RAID levels, you can find combinations of two failed drives that do crash the array – essentially two drives in the same partial mirror or RAID5 set.

Just like RAID 5, RAID 6 features the caveat of performance sag when writing small blocks of data. RAID 6 has yet somewhat worse capacity and performance efficiency than RAID 5, but the capacity efficiency is still a lot better than RAID 10.

Linear (spanning) = maximum capacity



This array creates a virtual drive with capacity equal to the sum of the member drives. The data is stored only once. The storage space of the physical disks is simply "concatenated", thus comprising the virtual drive.

The performance of this type of array is equal to that of a single member drive. **If a single drive fails, the array fails too, unrecoverably!**

Theoretically one could speculate that it might be possible to recover data manually from an individual disk drive - e.g., one could read an individual file, if one could identify its likely beginning and end by the content of its respective sectors. In practice, this could only work out if the file system on the virtual drive had zero fragmentation and the file being rescued did not exceed a boundary between physical drives.

JBOD = "Just a Bunch Of Drives"

This acronym stands for direct access to individual physical drives on a RAID controller - direct access to drives that are not a part of any array. This mode of physical disk access can be used e.g. on an occasion when there's only a RAID adapter available in a particular machine and we need to copy data from a stand-alone drive, that has been partitioned and mounted on a plain (non-RAID) disk adapter.

This acronym is sometimes also used to characterize storage expansion cabinets that don't contain their own SCSI-to-SCSI RAID adapter.

Other array types (RAID levels)

There are also other, less frequently used "numbered" array types: RAID-3, 4 and 7.

Spare drive

Most RAID controllers are capable of using spare drives. When a recoverable degradation occurs on a redundant array (RAID 1 or 5) present in the system, the RAID controller looks for physical drives designated as spare - if it finds a suitable spare drive, the controller makes the spare drive a member of the degraded array and starts rebuilding. Thus, the system administrator is only required to replace the failed drive and perhaps assign it as a spare.

Redundancy renewal after an array degradation

A degraded RAID 1 (mirror) or a RAID 5/6 can be rebuilt - array redundancy can be restored.

Before the rebuild can start, the failed drive has to be replaced (or the respective other failure repaired). If the hardware allows for hot swapping and the RAID controller's firmware and software can cope, the disk can be replaced without even power-cycling the host machine and operating system.

Some RAID controllers demand that the replacement drive be explicitly initialized and assigned as a spare - only then the array accepts the replacement drive and starts rebuilding.

Other RAID controllers are happy even with a plain drive replacement - the automatic array rebuild starts as soon as the original drive is re-detected (e.g. after it suffered a temporary power outage) or after a factory-clean replacement drive is inserted. The automatic rebuild does not start if the newly inserted disk shows signs of being a member of some other array once upon a time, or if it contains a valid MS-DOS partition table (and the controller is worried that it might destroy useful data on the replacement drive).

Advanced array configurations

More flexible volumes

Any serious RAID adapter allows for multiple "virtual volumes" that are smaller than the capacity available on the physical drives. That way it's possible to create e.g. two RAID-5 volumes across four disk drives, with one of the volumes taking up 30 % of space of each disk and the other one taking up the remaining 70 %.

Particular physical drives can be left outside of any array and accessed directly etc.

Multi-layer arrays

Some RAID controllers allow for the setup of hierarchical arrays. The simplest case is a two-layer array: e.g., at the lower layer, several physical disks comprise a single "virtual drive" and, at the higher layer, several virtual drives comprise a "virtual volume".

Each layer may employ a different algorithm (array level - 0,1,5).

Consequently, the RAID controller may differentiate between local and global spare drives etc.

SAN = Storage Area Network

A storage-area network consists of several intelligent array units, low-level network elements known as SAN switches and optical cabling. (Don't confuse this with NAS = Network-Attached Storage.)

The key networking standard in this area is FibreChannel.

SAN's allow for flexible storage space allocation to a number of user machines or legal entities, access to a central data backup facility, fast distributed filesystems, geographical redundancy of storage volumes etc.

IDE/ATA/SATA vs. SCSI vs. FibreChannel

The abbreviations are the names of different storage busses - some of these busses can be used for connecting other, non-storage devices.

IDE	MBps
IDE (ATA) PIO 0	3,33
IDE (ATA) PIO 1	5,22
IDE (ATA) PIO 2	8,33
IDE Multiword DMA 0	4,16
IDE Multiword DMA 1	13,33
IDE Multiword DMA 2	16,66
EIDE (Fast ATA-2) PIO 3	11,11
EIDE (Fast ATA-2) PIO 4	16,66
Ultra-DMA/0 (ATA/16)	16,66
Ultra-DMA/1 (ATA/25)	25,0
Ultra-DMA/2 (ATA/33)	33,33
Ultra-DMA/3 (ATA/44)	44,4
Ultra-DMA/4 (Ultra-ATA/66)	66,66
Ultra-DMA/5 (Ultra-ATA/100)	100
Ultra-DMA/6 (Ultra-ATA/133)	133
Serial-ATA/150	150
Serial-ATA II / 300	300
(Serial-ATA/600 ?)	(600)

SCSI	Bits	MBps
Asynchronous	8	5
Fast	8	10
Fast Wide	16	20
Ultra	8	20
Ultra Wide	16	40
Ultra2	8	40
Ultra2 Wide	16	80
Ultra160 (Ultra3)	16	160
Ultra320	16	320
FiberChannel 1Gb	2 opt	100
FiberChannel 2Gb	2 opt	200
FibreChannel 4Gb	2 opt	400
(FibreChannel 10Gb)	(2 opt)	(1000)
SAS/300	2	300
(SAS/600 ?)	(2)	(600)
(SAS/1200 ??)	(2)	(1200)

SCSI is perhaps the most venerable and reputable - ever since the ancient times it's been a somewhat expensive technology, used mainly in server machines. During the years of evolution there have been several generations - the king preparing for retirement is Ultra320 SCSI (320 MBps half duplex). In the RAID industry, there's a related standard called SCA that specifies connectors, backplane electronics and SCSI bus protocol enhancements for hot swap functionality (including the SES and SAF-TE interfaces). The SCA enclosures are capable of Ultra2-Wide (80 MBps), U160 or U320 SCSI.

Serial Attached SCSI (SAS), the long-awaited successor to SCSI, should finally start to get some market share. First implementations in disk drives, RAID controllers and enclosures have started to appear around the new year of 2006. The initial standard runs at 300 MBps full duplex. SAS is not a parallel SCSI bus anymore – it's really a twin of the SATA-II/300 specification. SAS and SATA-II/300 have some overlap or cross-compatibility. In contrast to SATA-II, SAS allows for more advanced multi-lane and switched topologies, perhaps resembling PCI Express in some aspects. A SAS device address looks familiar, similar to a SCSI or FC device identifier, except that its structure is different (disk:enclosure:slot vs. channel:ID:LUN or NAI:AL_PA).

FibreChannel is an optical technology, perhaps originally meant as a SCSI successor, nowadays the key standard in SAN communications – really a networking technology. Several variants have historically existed, sporting 1, 2 and 4 Gbps (giga bits, not bytes) - i.e. 100, 200 and 400 MBps respectively, all of them full duplex. A 10Gb FC standard is in the works, perhaps even already implemented in the stacking ports of Qlogic FC switches.

IDE/EIDE/ATA/ATAPI is a cheap disk interface technology, intended for the desktop computers. There have been several generations of IDE, the newest ones being UltraATA/133 and SerialATA-II/300 (133 and 300 MBps). The SerialATA standard and especially SATA-II contains some hot-swap support - the manufacturers usually call it SAF-TE, but unlike with SCSI, where the SAF-TE processor is a SCSI bus device, with SerialATA the controller talks to the hot-swap backplane processor out of band via I2C.

The first-generation SATA standard (SATA150) was really just a replacement for parallel IDE. The serial bus has some important advantages: narrow cabling, simple connectors, low number of bus drivers per port -> low power consumption. Already the first-generation SATA150 standard was effectively based on SCSI commands (just like the parallel ATA enhancement called ATAPI), but without the “Tagged Command Queueing” enhancement known from true SCSI, which effectively prevented full utilization of the full-duplex nature of the SATA link layer.

The “command queueing” concept, now called the NCQ (Native Command Queueing) has been re-introduced by the SATA-II standard, which is commonly associated with the 300 MBps transfer rate.

During the early days of SATA-II, the relevant standardization body has clouded the naming conventions a bit, by giving up the 300MBps transfer rate requirement. Therefore, SATA-II without the /300 suffix may actually mean as much as SATA150 with NCQ support. Take care when studying vendor pricelists.

Also, please note that the current generation of Hitachi SATA-II/300 disk drives (up to and including 500 GB models) are shipped soft-configured for a maximum transfer rate of 150 MBps. If you insist on the full 300MBps tranfer rate, you need to use a software utility by Hitachi (called the Hitachi Feature Tool) to modify this factory setting. The catch is, that this software utility only works on chipset-integrated SATA/SATA-II controllers, emulating legacy IDE ports. At the moment it is verified to work on integrated SATA-II controllers by Nvidia and Intel. This requirement is however not met by external PCI-based SATA-II controllers, made by e.g. Promise or Marvell, including onboard-integrated chips from these vendors – these controllers need a hardware-specific driver and appear as SCSI HBA’s in the host computer’s operating system. The same is true about all sorts of SATA-II RAID controllers, external SCSI/FC-based units or PCI-based. In other words, if you want to use a Hitachi SATA-II/300 disk drive at 300 MBps in a RAID, you need to use a motherboard with a supported on-chip SATA-II controller to configure the drives for 300 MBps. Or you can choose to stay with SATA150+NCQ, which isn’t all that bad after all, given that even the 500GB disk drives don’t exceed 80 MBps of sustained linear transfer rate.

The classic (large and expensive) external RAID arrays contain a SCSI-to-SCSI controller. The host computer connects to such an array via an external SCSI bus and a regular PCI SCSI controller (officially called the "host bus adapter"). The SAN-enabled external storage units use FiberChannel as the external bus.

RAID controllers for the PC are a relatively newer affair. The RAID controller is inserted into the PCI bus and has a SCSI or IDE/SATA interface for the disk drives.

An interesting hybrid between the expensive world of external storage and the cheap world of IDE/SATA are stand-alone IDE-to-SCSI controllers for external storage units - such storage units use IDE/SATA disks on the inside but present a SCSI/FC interface to the outside world.

The SCSI bus addressing rules allow up to 16 devices per bus. One of them is the SCSI controller itself (the host adapter), another ID might be occupied by the hot-swap enclosure chip (the SCA backplane processor with a SES or SAF-TE interface). Hence, theoretically there can be up to at least 14 disks on a single SCSI bus. On the other hand, four todays SCSI drives are able to consume all bandwidth of an U320 bus - thus, in practice, total bandwidth will be the likely limiting factor for drives per bus.

One IDE channel traditionally supports up to two drives and there have been experiments with four drives per channel. With RAID though, one has to be aware that a single drive with failed electronics or poor connector contact can easily jam the whole channel, thus rendering the other drive inaccessible too - which implies an immediate failure, temporary or maybe permanent, of any array, redundant or not, spanning the two drives. Most importantly though, the master/slave attachment significantly cuts overall throughput of the IDE channel. Therefore, the general rule in IDE RAID is that there should be a single drive per IDE channel. Most IDE RAID controllers enforce that rule.

Software vs. Hardware RAID (controllers for the PC)

Hardware RAID

The array is emulated and managed by the controller's hardware. This is typically a specialized RISC/DSP processor, perhaps backed by a hardware accelerator for parity calculations (a XOR chip). Hardware RAID controllers are usually equipped with large amounts of disk cache - up to several hundred megabytes of RAM.

Such a controller can function independently of the state of the host operating system and its applications - e.g., it can continue rebuilding the array while the host system is rebooting, except perhaps for a short pause during the hard PCI bus reset. The controller can even process several tasks simultaneously - e.g., it can be rebuilding several arrays in parallel.

To the host system, a hardware RAID controller usually mimicks a plain SCSI controller and the configured arrays are presented as SCSI disk drives. The true physical drives are not visible from the host operating system. Such is the common behavior of both SCSI RAID controllers and IDE RAID controllers - both typically emulate SCSI devices over the respective type of physical drives. (As an exception to the rules, some SCSI RAID controllers have an optional pass-through feature that allows straight access to selected devices on the RAID-private bus.)

In a historical perspective, by far the favourite processor used by many vendors in their RAID controllers is the Intel i960. More precisely, it has been, in the U160 SCSI era. More recently, Intel has succeeded to promote its younger siblings of the IOP300 family based on its Xscale core. This change of CPU core architectures brought about a certain slow-down on the RAID market and has mixed the cards in the deck quite a bit.

Sometime during 2005, other chip-makers started to roll out competing solutions, following the well-known layout pioneered by Intel.

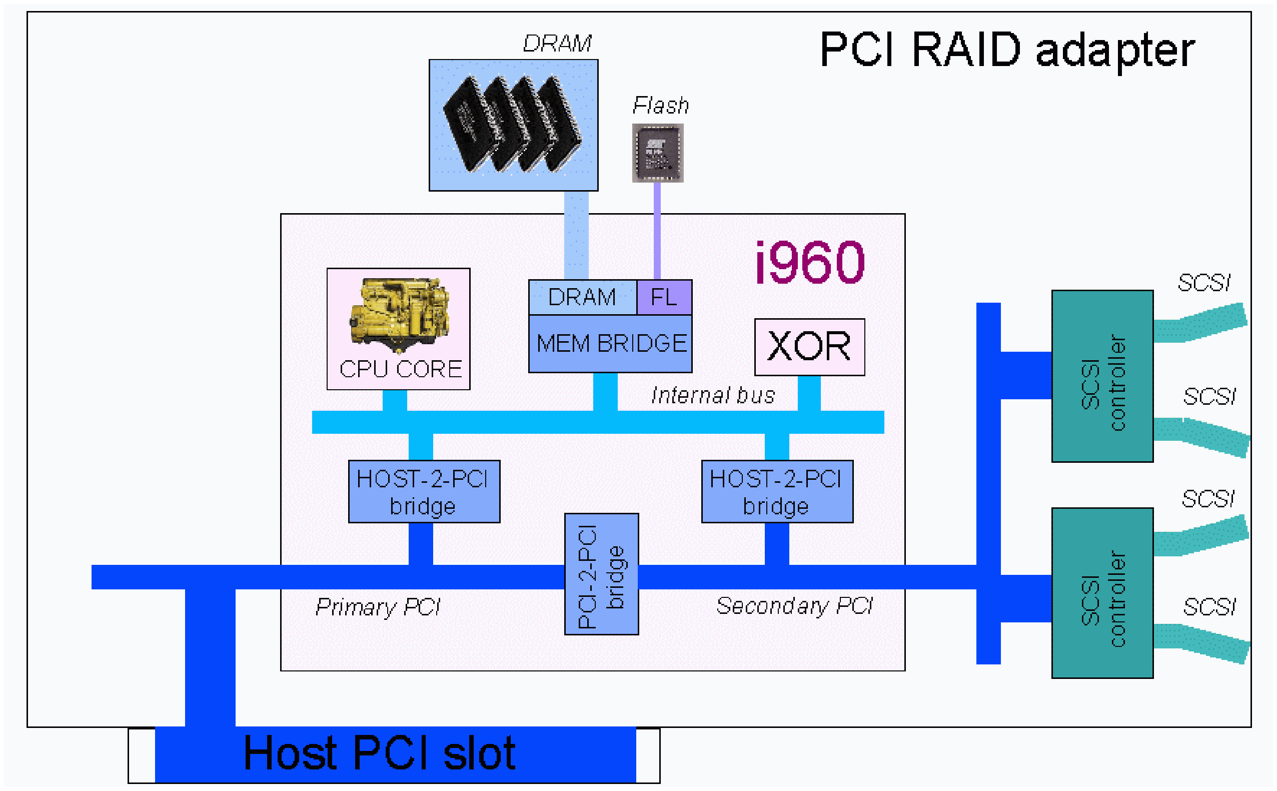
AMCC corp, which happend to acquire 3ware in that same timeframe, introduced the PowerPC 440SP/SPe.

Adaptec Corp. is an exclusive user of chips called the “RoC” (Raid-on-Chip), which have replaced the retired i960 in Adaptec’s FSA=AAC RAID family. This way Adaptec avoided porting its software to the Xscale-based IOP3xx family by Intel. It would appear that there are already two generations of the RoC chips, in this paper we shall nickname them in the following way:

- **RoC1: Adaptec 2130S, 2230S** – PCI-X, a MIPS CPU core (or *maybe* an „overclocked“ i960), contains an on-chip SCSI HBA, shipped with AACCLI
- **RoC2: Adaptec 2420SA, 2820SA, 4800SAS, 4805SAS** – PCI-X or PCI-e x8, a MIPS core, RAID6, on-chip SATA2 or SAS HBA (4-8 channels), shipped with the ARCCONF CLI

It’s difficult to say who manufactures those RoC chips for Adaptec – maybe Broadcom or Vitesse, maybe each generation by a different manufacturer. The important point is, that in terms of performance these RoC chips are quite similar to the Intel processors around IOP321/331.

The Intel i960/IOP processors as well as the competitors' I/O CPU's have been in wide use in SCSI, IDE, SATA-I/II and SAS RAID controllers. A typical RAID controller layout featuring these processors would look like this:



This typical solution features a few ingenious ideas:

- an on-chip XOR accelerator on the RAID processor (newer I/O CPU's can do not only XOR for RAID 5, but also XOR + Reed Solomon for RAID 6)

- at the position of disk bus drivers, the solution uses some generic SCSI (or IDE/SATA) controllers for the PCI bus that can otherwise be seen stand-alone on plain disk controller boards for the PCI. In this solution, they're used as subordinate devices to the i960/IOP that talks to these plain controllers through its onboard secondary PCI. The advantages are clear: RAID controllers can be developed like a LEGO theme, with the components talking to one another using the well defined PCI interface. Many different compliant pieces are available on the market that can be used as subordinate devices.

- the data transfers are DMA'ed to/from the controller's RAM that functions as a disk cache and where the XOR accelerator operates.

- the processor contains a PCI-to-PCI bridge between the primary and the secondary bus, so that theoretically the host computer's operating system could talk directly to the devices on the secondary bus. In practice, this is usually prevented. The built-in PCI bridge in the i960/IOP allows for definition of "private" devices on the secondary bus, even the bridge itself can be prevented from having itself detected as a PCI bridge. That way it can be arranged that the host computer's OS will not find the plain disk controllers and, to the host system, the whole RAID controller card appears to be a single opaque PCI device. On RAID controller boards from different vendors, the i960 presents different PCI Vendor ID's and Device ID's.

If we can get to know the particular i960/IOP processor part number used on a particular RAID controller model (usually this information can be found on the vendor's web), with the help of the following table we can calculate the controller's theoretical throughput - or at least estimate a relative ranking of different controller models available on the market.

Processor (or an IO chip + CPU core chip)	PCI clk. [MHz]	PCI width [bits]	int.bus clk [MHz]*	core clock [MHz]	RAM type*	RAM capacity
Adaptec RoC2 [MIPS] (+6)	133 or x8	64 PCI-X or PCI-e x8	?	?	DDR	?
Adaptec RoC1 [MIPS? i960?]	133	64	?	?	DDR	?
AMCC PowerPC 440SPe+ (+6)	266 x8 x4	64 PCI-X 2.0 PCI-e x8 2x PCI-e x4	2x 166 128bit	až 667	DDR333 DDR2/667	16 GB 8 GB
AMCC PowerPC 440SP+ (+6)	266	3x64 PCI-X 2.0	2x 166 128bit	až 667	DDR333 DDR2/667	4 GB
IOP333 (i80333) (+6)	133 x8	2x 64 PCI-X 1x PCIe x8	333 64b	až 800	DDR333 DDR2/400	2 GB 1 GB
IOP332 (i80332)	133 x8	2x 64 PCI-X 1x PCIe x8	266 64b	až 800	DDR333 DDR2/400	2 GB 1 GB
IOP331 (i80331) (+6)	133	2x 64	266 64b	až 800	DDR333 DDR2/400	2 GB 1 GB
IOP219 (i80219**)	133	64	200 64b	až 600	DDR200	1 GB
IOP321 (i80321)	133	64	200 64b	až 600	DDR200	1 GB
IOP315 (i80314** + 2x i80200)	133	2x 64	133 64b	733	2xDDR200	12 GB!
IOP310 (i80312 + i80200)	66	2x 64	100 64b	733	SDRAM100	512 MB
IOP303 (i80303 =jádro i960)	66	2x 64	100 64b	100	SDRAM100	512 MB
IOP302 (i80302 =jádro i960)	66	2x 64	66 64b	100	SDRAM66	128 MB
i960RN (i80960RN)	33	2x 64	66 64b	100	SDRAM66	128 MB
i960RM,RS (i80960RM,RS**)	33	2x 32	66 64b	100	SDRAM66	128 MB
i960RD (i80960RD**)	33	2x 32	66 32b	66	SDRAM66	128 MB

**the internal bus is 64bit wide, the memory is 64bit wide or with the older 66MHz variants optionally 32bit wide*
***the i960 RS, RD and older processors lack the "Application Accelerator", i.e. the hardware XOR support.*
The same holds true about the IOP219, which is derived from the IOP321.
Also the dual-chip IOP315 solution lacks a XOR accelerator, as it is not intended for "storage" applications
+6: the IOP333, later revisions of the IOP331, the PowerPC 440SP+/SPe+ and the RoC2 have HW RAID6 support

The Intel Xscale architecture used in the new IOP310 and higher chips is based on an instruction set compatible with the original ARM core - as such, it is not backwards compatible with the retired i960. For marketing reasons, or to make the affair a bit more incomprehensible, the 80302 and the 80303 wear an IOP302 / IOP303 sticker, even though they have an i960JT core on the inside.

The IOP310 and IOP315 are two-chip solutions - essentially a north bridge plus a separate Xscale core.

The IOP219 is a stripped-down version of the IOP321, intended for other-than-storage applications. Interestingly, e.g. Areca uses this chip in combination with an external onboard XOR chip, and the result is surprisingly powerful.

For basic orientation, in various RAID controller products the different processors have been observed to show aproximately the following performance values (sequential RAID5 throughput):

IOP302/303 = 50-60 MBps (LSI, Adaptec), IOP321 = 100-120 MBps (LSI 320-2X, Accusys), Adaptec RoC = 120 MBps (2130S), IOP331 = 150 MBps, Areca IOP219 + ext. XOR = 200 MBps

It would seem that the development freeze on the i960 architecture at about 100 Mips presented many a RAID vendor with a grim prospect of having to port most of the firmware to a different core architecture in order to stay competitive. It seemd that the IOP303 was good enough for two U320 channels and noone was exactly eager to top that. This situation reflects in the table below – for maybe two years, the only new product based on the IOP321 was the LSI MegaRAID 320-2X, also sold with an Intel badge, under the SRCU42X partnumber. Intel used to present the LSI solution as a major success story on its IOP website.

There's one other reason why the IOP CPU's with Xscale core were initially only reluctantly adopted. The IOP310 meant two chips instead of one (compared to the IOP303). On the other hand, the IOP321, the first practically usable Xscale IOP, has only one PCI-X port. That's why IOP321-based controllers are usually a part of external solutions, attached to the host computer via SCSI (Accusys, Areca). In this topology, it doesn't matter much that the IOP CPU talks to the disk-side and host-side HBA's over a single common PCI-X segment. The LSI Megaraid 320-2X, being probably the only existing implementation of a host-based PCI RAID HBA with the IOP321, therefore contains an additional PCI-to-PCI bridge on the host side (the big silvery chip package labeled "Tundra"). This additional PCI bridge has both the IOP321 and its private U320 HBA hidden behind it. More chip packages = more pins = higher price. And, more heat to dissipate.

Just for comparison: the LSI MegaRAID SATA300-8x is based on the IOP331, and therefore you wouldn't find an additional PCI bridge on its PCB.

Examples RAID controller boards:

Manufacturer and model	**CPU**	**Disk channels**
LSI MegaRAID SAS 8480E	IOP333	8x SAS/300 ext.
LSI MegaRAID SAS 8408E	IOP333	8x SAS/300 int.
Adaptec 4800SAS (Marauder-X) Adaptec 4805SAS (Marauder-E)	RoC2 (PCI-X) RoC2 (PCI-e)	8x SAS/300 int. + 8x SAS/300 ext.
Adaptec ASR2230S (Lancer)	RoC1	2x U320 SCSI
Adaptec ASR2130S (Lancer)	RoC1	1x U320 SCSI
LSI Logic MegaRAID SCSI 320-4X	IOP321	4x U320 SCSI
Intel SCRU42X = MegaRAID 320-2X	IOP321	2x U320 SCSI
LSI Logic MegaRAID SCSI 320-2X	IOP321	2x U320 SCSI
LSI Logic MegaRAID SCSI 320-2	IOP303	2x U320 SCSI
LSI Logic MegaRAID SCSI 320-1	IOP302	1x U320 SCSI
ICP Vortex GDT8x24RZ	IOP303	2x U320 SCSI
Adaptec ASR2200S (Vulcan)	IOP303	2x U320 SCSI
Adaptec ASR2120S (Crusader)	IOP302	1x U320 SCSI
Adaptec 5400S (Mustang)	StrongARM@233	4x U160 SCSI
Adaptec ASR3410S	IOP303	4x U160 SCSI
Mylex AcceleRAID 352	i960RM	2x U160 SCSI
Mylex AcceleRAID 170	i960RM	1x U160 SCSI
Mylex AcceleRAID 160 == AcceleRAID 170LP	i960RS	1x U160 SCSI
Adaptec 2820SA (Intruder)	RoC2	8x SATA2/300
Adaptec 2420SA (Intruder)	RoC2	4x SATA2/300
Areca ARC1210 Areca ARC1220, 1230, 1260, 1270	IOP332 (PCI-e) IOP333 (PCI-e)	4x 8x, 12x, 16x, 24x SATA2/300
Areca ARC1110, 1120, 1130, 1160, 1170	IOP331	4x, 8x, 12x, 16x, 24x SATA2/300
LSI MegaRAID SATA300-8X	IOP331	8x SATA2/300 (8 disků)
Adaptec 21610SA	IOP303	16x SATA150 (16 disků)
Adaptec 2810SA	IOP303	8x SATA150 (8 disků)
Adaptec 2410SA	IOP302	4x SATA150 (4 disky)
Intel SRCS14L = ICP Vortex GDT8546RZ	IOP303	4x SATA150 (4 disky), 64 MB
ICP Vortex GDT8546RZ	IOP303	4x SATA150 (4 disky), 128 MB
LSI Logic MegaRAID SATA-8*	IOP302	8x SATA150 (8 disků)
LSI Logic MegaRAID SATA-6	IOP302	6x SATA150 (6 disků)
LSI Logic MegaRAID SATA-4	IOP302	4x SATA150 (4 disky)
Promise FastTrak SX6000	i960RM	6x UATA100 (6 disků)
LSI Logic MegaRAID I4	i960RS	4x UATA100 (4 disky)

**this product only occurs in the promotion leaflet, without a photo - it can't be found anywhere else on the web*

There's an interesting young brand of IOP/Xscale-based RAID controllers: the Taiwanese Areca corp, who has already earned significant reputation by its performance and comfortable firmware and management software. Areca got started on external RAID controllers, its first products were based on the IOP321. Based on experience with external RAID solutions, it has also launched a family of internal SATA2 RAID HBA's for the PCI-X and PCI Express, based on the IOP331 and IOP332/333.

The one distinctive feature of Areca controllers is an external onboard XOR accelerator chip with RAID6 support. Owing to this external accelerator ASIC, Areca delivered RAID6 much earlier than the rest of the market – whom it took quite some time and effort to add RAID6 support, which only happened after Intel and its CPU-making competition hastily added RAID6 support to the newer revisions of their respective chip families. Interestingly, as soon as Intel introduced the stripped-down general-purpose IOP219 (featuring no on-chip RAID acceleration at all), Areca combined it with its trade-mark XOR chip in its RAID controllers – and the combination seems surprisingly more powerful than the competitors' products based on the bare IOP321 or IOP331.

There are a few RAID vendors who do not use the i960 - one of the most reputable is InforTrend, marketing external RAID controllers and complete storage units based on the PowerPC processor family. This particular fact used to be properly addressed in one of the whitepapers on InforTrend's corporate website, especially with respect to the dividing line in the Intel IOP family.

Infortrend uses "vanilla" PowerPC 750 processors by IBM, combined with an in-house external "ASIC" – a dual-port PCI bridge + RAM controller + XOR accelerator. Infortrend used to sell ASIC133 with SDRAM memory for a long time, the transition to ASIC266 with DDR memory took place about the time when Xscale-based Intel IOP's finally started to appear on the market. The ASIC266 has been upgraded to RAID6 in its later revisions.

Another vendor not using the i960 is 3ware, famous for its IDE and SATA RAID controllers. The 3ware portfolio contains some rather interesting 8-, 12- and 16-channel UATA/133, SATA and SATA-II controllers. The key component of the multi-channel 3ware controllers, presented in all the manuals and PR materials, is their StorSwitch - a hardware switching matrix, multiplexing the data flows from the individual disk drives. It would seem that the main advantage of the StorSwitch is direct transfers from the "AccelerATA" IDE/SATA ports straight into the controller's RAM via a fast internal bus (i.e., no private PCI).

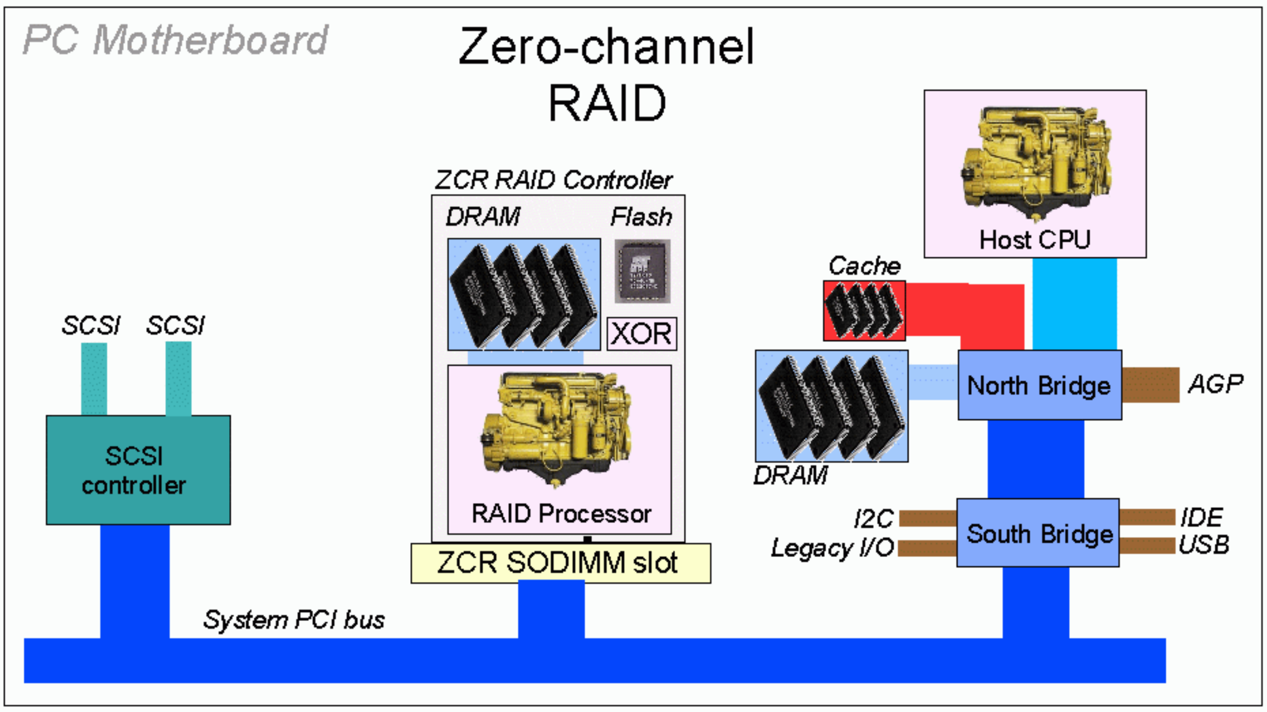
There was a time when 3ware wouldn't disclose the size of the RAID controllers' RAM – it's not officially known about the 3ware 7000 and 8500 families. Upon a closer inspection of physical specimen of these controllers, one can observe that these families feature 512 kB to 2 MB of DRAM that's used as a data cache, plus a smaller SRAM for the controller's CPU. The controller CPU used is a 16bit MCU by SGS Thomson – so much for comparison to the 64bit RISC cores used in Intel IOP's et al.

The 3ware 9500 family of SATA RAID controllers still seems to employ a relatively weak CPU, but it already has 128 MB of SDRAM for data cache, in a SODIMM module, upgradeable to 256 MB.

The first 3ware product line to contain a "full-featured" CPU is the 9550 family, developed already under AMCC ownership. The CPU is a PowerPC, accompanied by 128 or 256 MB of RAM (depending on model) out of the box. Nevertheless, the StorSwitch still remains to be the essential component, as the PowerPC 405 used on the boards is some general-purpose 32bit embedded CPU core by IBM, rather than a dedicated “storage” CPU by AMCC, such as the AMCC PowerPC 440 SPe+ juggernaut.

There’s one important achievement that has earned 3ware lots of well-deserved credit, being a unique solution in the respective market segment: the 3ware 7000-2 and 8006-2 dual-channel IDE/SATA controllers, probably the only dual-channel *hardware* RAID controllers on the market, thus the only serious products in their category.

There is a special class of cost-effective hardware RAID solutions called the Zero Channel Raid (ZCR). In practice this is a SCSI-only affair. The ZCR controller is a snap-in card for motherboards that have a ZCR slot (SODIMM connector). The ZCR card contains only an IO processor and its RAM (used mostly for the RAID cache), but no low-level SCSI bus drivers/controllers. This is because the ZCR uses one or two plain SCSI channels integrated on the host motherboard. Motherboards and PC machines with a SODIMM/ZCR slot are typically sold with an onboard dual-channel U160 or U320 SCSI controller, with a note about the RAID upgrade option somewhere in the manual.



The ZCR SODIMM slot really contains the PCI bus. When the ZCR card is inserted, the card shields away the original onboard SCSI BIOS from the boot sequence and inserts its own BIOS. Thus, upon boot, the ZCR controller's welcome message appears instead of the onboard dumb controller's. During operation, the ZCR module's RAID processor transfers data from the disks to its RAM over the host PCI bus - and then dispatches it to the host PC via that same PCI bus. The major disadvantage of this solution is therefore its lower performance - the ZCR's host PCI bus segment is a bottleneck. Other factors contributing to the low performance stem from the overall "cheapness" of the ZCR solution - small cache, slow RAID processor etc.

A short overview of the eventful history of hardware RAID‘s

The market of RAID solutions is ever in motion, stirred by business alliances and acquisitions, perhaps more importantly than by technical progress.

Mylex, once upon a time a very famous brand, maybe even synonymous to RAID, using Intel i960 processors in its later product lines, started to fade away by the end of 90's.
In 1999, Mylex corp. got purchased by IBM, and in 2002 the "Mylex division" was sold again, this time to LSI Logic. Under ownership of LSI Logic, further development of Mylex AcceleRAID and ExtremeRAID families was stopped.

In the meantime, in 2001, LSI Logic corp. has managed to take over another famous RAID brand: the RAID division of American Megatrends Inc. (AMI), i.e. especially the MegaRAID product line, along with all relevant know-how and 200 employees. This product family is being further developed – under LSI ownership, in close cooperation with Intel, the MegaRAID family survived the transition from the original i960-based IOP's to the Xscale-based IOP321/331/333 (between 2003 and 2006). In this context, please note that LSI RAID controllers are still re-badged and sold under the Intel brand in 2006, besides the original LSI MegaRAID brand.

Another independent brand of quality RAID controllers, formerly a private German company called ICP Vortex, got acquired by Intel in 2001. Development of the ICP Vortex RAID family was stopped after the models based on the IOP302/303 CPU's – don't be mistaken by the irrelevant note about the IOP310 in the official press release addressing the merger. The only material result of the merger was, that Intel was rebadging some older RAID controllers developed by ICP Vortex.

This "Intel episode" of ICP Vortex ended in 2003, when the "ICP Vortex division" was sold to Adaptec. Adaptec first let the ICP Vortex brand optically alive for some more time, consequently Intel still used to sell some rebadged older ICP Vortex controllers in early 2005. For all practical purposes, independent existence of ICP Vortex ended during 2005, when ICP Vortex controllers became ultimately unavailable, both under the original brand and under the Intel label. Adaptec still seems to sell some of its own rebadged controllers under the ICP Vortex brand, just a few models based on the IOP302/303 and RoC – but this appears to be the swan song of ICP Vortex.

3ware, another originally independent company, got acquired by AMCC in the spring of 2004. In the last years of the 20th century, AMCC has managed to become an active supplier of chip solutions (special-purpose application processors), originally mostly for the telecommunications and computer networks, newly also for storage technologies. Apart from 3ware controllers, its portfolio includes powerful IO processors with PowerPC core, a direct competition to Intel – these could appear on the retail RAID market sometime during 2006. The PowerPC core hints close ties with IBM.

The first effect of the acquisition of 3ware by AMCC was a delay in retail supplies of 3ware controllers (at least in some parts of Europe), during a short period of maybe a month or two, while AMCC was absorbing and reorganizing the 3ware distribution network worldwide. In 2005 and 2006, the 3ware brand is alive and well, introducing new product families.

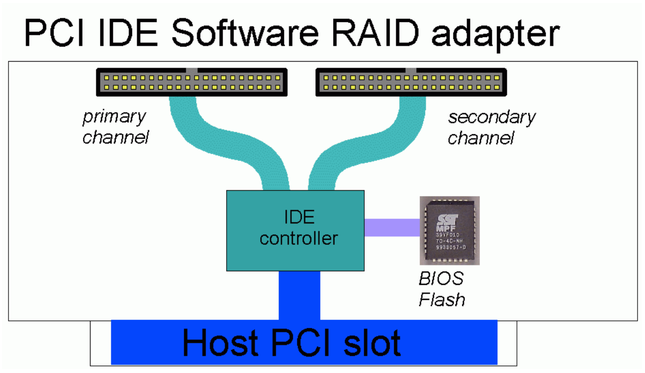
A somewhat earlier acquisition happened by the end of 1999, when DPT got purchased by Adaptec. This acquisition has enriched the Adaptec product portfolio by DPT models (also known as FreeBSD “device asr” or Linux “I2O RAID”), undoubtedly accompanied by significant know-how. The DPT product family went on for some time in the form of zero-channel RAIDs, such as the Adaptec ZCR2005 and 2015. Compared to the conceptually younger RAIDs of the FSA family (also known as AAC RAID), the DPT family had significantly weaker own intelligence and weaker support in terms of comfortable management software. Let alone the historically given lower performance.

The Adaptec FSA/AAC PCI RAID family, characterized especially by a common RAID meta-model and unified management software, appears to have become Adaptec’s chosen basis for future development – it is being actively developed and new products get added. During its development, the FSA/AAC family has undergone a successful transition from the i960 core architecture to the new RoC chips that seem to have a MIPS core. Adaptec apparently avoided Intel Xscale-based chips, somewhat surprisingly, because an early AAC RAID, the ASR5400S, had a StrongARM CPU. Please note that older versions of the AAC firmware are incompatible with newer versions of the management utilities (CLI, and especially the Storage Manager), and vice versa. This is probably a natural consequence of the years of continuous development.

Software RAID

The array is emulated and managed by software (drivers) running on the host computer's CPU, with some minimum assistance on part of hardware, usually even without it. During mirroring operations, the RAID software typically writes both disks in parallel and reads from only one of them. The parallel writing is usually not arranged transparently by hardware - i.e., the data really has to flow twice through the relevant busses (memory bus, northbridge - southbridge, PCI).

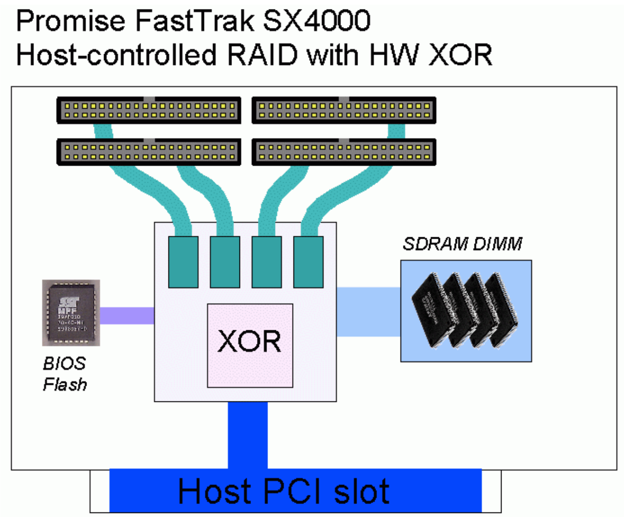
The hardware of such a "RAID" controller and its BIOS take care of distinguishing between stand-alone disks and array members upon boot, in the simplest possible way - so that the operating system boots just the same regardless of whether the array is well or if one or the other drive has failed. With especially simple (and cheap) RAID controllers, upon array degradation one is forced to enter the BIOS util, rebuild the array, wait until the array is rebuilt and only then boot the operating system - the rebuild doesn't run in the background.



The BIOS and drivers obviously contain a lock against being used with a different IDE controller than the one they ship with - although there have been rumors on the USENET that e.g. the Promise software can be hacked to run on any generic IDE controller.

The software RAID controllers are almost exclusively IDE-based. Hence if you are considering a SCSI RAID controller, you don't have to be worried that it could cause excessive load on the host CPU. More precisely, Adaptec does sell *plain* U320 SCSI controllers with an optional "HostRaid" feature in the BIOS - it's decent enough not market them as RAID controllers.

The software RAID typically only supports RAID0, RAID1 and JBOD, i.e. the array types where, even with software emulation, most of the hard work is done by DMA. During RAID5 operation, it is necessary to crunch with XOR the whole volume of data, byte after byte. Given the volume of data, this operation is quite computationally intensive - CPU load would skyrocket if this was done by the host CPU. No RAID vendor risks the kind of bad publicity that would have to be expected from marketing a software-based RAID5. This is why multi-channel IDE RAID controllers are almost exclusively hardware-based - because RAID5 starts to make sense with more than two drives in the system.



Somewhere halfway between hardware and software RAID, there are some multichannel RAID controllers by Promise, e.g. the FastTrak S150 SX4 or the SX4000. These controllers don't have an autonomous CPU, but they do contain an ASIC for hardware XOR acceleration. They don't provide a full-featured hardware RAID. The good thing about them is that they have an onboard RAM (up to 256 MB) that serves as a disk cache and as a workspace buffer for the XOR operations. Thus, the whole volume of data being read or written doesn't travel across the host system's busses several times back'n'forth, but is transferred only once. The hardware accelerator also does the mirroring during RAID1 operation, so that even during write operations, the data travels across the host busses only once. A lot of work related to the mapping of the virtual drive to the physical sectors and instructing the XOR ASIC still remains up to the host CPU and its IRQ system.

There are also pure-software solutions - the most renown is probably the native software RAID support in the Linux kernel. This solution works on top of any physical disk, attached via any SCSI or IDE controller - it works even on top of any other block device. Similarly, the Windows NT Server, as well as the W2K and XP, already have some disk mirroring support in the base system.

Which RAID should I get? A software RAID or a hardware RAID?

Some authors argue that on today's gigahertz processors, a software RAID is often faster than a hardware RAID, and much cheaper at a comparable performance. Yet there are some (aforementioned) caveats:

1) it is only feasible to emulate RAID0, RAID1 and JBOD in software - not RAID5.
2) during disk operations, the software emulation consumes CPU power and bus bandwidth and increases the number of interrupts being served, so that
3) the aforementioned high performance can only be achieved if the system is fairly idle except for the disk operations, which only happens during focused benchmarking of the disk subsystem.
4) the software RAID emulation is not appropriate for systems loaded by other activity (servers), where the performance advantage over a hardware RAID diminishes, as the hardware RAID is not directly affected by the host CPU load.
5) with software RAID, an operating system collapse may damage array integrity. Hardware RAID, on the other hand, is encapsulated - it enforces array integrity regardless of the current state of the host computer's operating system.
6) particular real-world software RAID implementations, being cheap solutions, often show poor behavior in critical scenarios (upon array degradation) - their administration comfort is poor, the administration software tends to be counter-intuitive and buggy in just those critical areas, the claimed background rebuild may actually fail to start etc. The operator must test and evaluate ahead of deployment.

To sum up, a software RAID is a cheap solution, appropriate for desktop systems and small servers, with a low average load, where a one-hour emergency outage necessary for a volume rebuild doesn't matter. Given the today's prices of large IDE drives, a software mirror that costs next to nothing can significantly improve data safety and network availability on a small server. Caution, as most authors point out, this doesn't mean that the admin can give up regular backups!

On the other hand, a quality and expensive hardware RAID solution is the only correct choice for systems that have a higher average load, their system busses have other things to do besides the disk transfers, high throughput is expected of them at any level of load and every minute of outage costs money, so that the possibility of a background rebuild during full operation significantly reduces the stress imposed on the system administrator, his boss and his IT users.

Within the software RAID arena there's one other paradox: owing largely to their tight integration with the operating system (and to the development background of the manufacturer), the native software RAID solutions provided as an integral part of todays OS'es are often more stable, more comfortable and performing better than third-party software RAID's (that are usually supplied with additional pseudo-RAID hardware). This is especially true under Linux.

In other words, when buying a "software RAID controller" these days, one has to think twice before making a decision, as fraud is lurking just around the next corner. Today's onboard integrated UltraATA controllers don't really lag behind the hardware that is sold as part of "retail packaged" software IDE RAID solutions. Even if you have a "RAID-capable" IDE controller integrated on the motherboard, consider using it as a generic IDE controller and running a hardware-independent OS-integrated software RAID on top of it. The ultimate choice is quite individual, dependent on vendor and model of the RAID hardware considered - there's no better advice than the results of a practical test.

The one area where proprietary software RAID still rules is support for hot-swap bays. In most operating systems, native support for SES or SAF-TE (or dumb) hot-swap bays is not available. What's worse, most operating systems are fundamentally incapable of adding and removing hard disks and their partitions on the fly and can't recover from low-level disk I/O errors on "non-removable" media.

Is there an easy way to know a "software RAID controller" from a hardware RAID at a first sight?

Yes there is - by the size and number of integrated circuit packages on the controller card. Moreover, the manufacturers usually display photographs on the web - thus, it's not necessary to buy the product only to find out.

A "software RAID controller" (the IDE variety) usually contains only two larger integrated circuits:

1) a conventional dual-channel IDE/UltraATA controller and
2) a Flash/EEPROM memory containing the BIOS.

The AnandTech site (www.anandtech.com) says literally that a RAID adapter of this type "consists of an ordinary IDE controller and an EPROM".

A true hardware RAID on the other hand contains at least these large IC's:

1) the RAID processor, emulating a SCSI device to the host system and managing the arrays - typically with an i960 core
2) at least one ordinary disk controller, single or dual-channel, IDE or SCSI. Or SATA or SAS, can be quad-channel or octal. A multi-channel RAID adapter can use several disk controller chips. The board contains a private PCI bus connecting the RAID processor to the onboard ordinary disk controllers. Some RAID processors, namely the Adaptec RoC, have disk controllers integrated on-chip.
3) several DRAM chips (SDRAM or DDR) for the RAID processor's firmware and data. During operation, a major part of the total capacity is used as a disk cache.
4) a Flash/EEPROM for the controller's firmware and BIOS (unless integrated on-chip on the RAID processor)
5) a XOR accelerator chip, dedicated to the parity calculations performed during RAID4/RAID5 operation. Nowadays usually integrated on-chip on the RAID processor.

Replacing and erasing disks

Most RAID's store their configuration on the disks (and perhaps also into an onboard EPROM/NVRAM).

Thus, once a particular disk has been assigned to be a member of an array, from that moment on it contains a special block of data saying that it is a member of an array, what type of array, which other disks take part in the array (SCSI channel/Device_ID or IDE channel/position) and what position the current disk occupies.

This way of storing configuration data on the disk is supposed to facilitate array reconstruction after a controller failure.

When a dead controller is being replaced, the disk must be attached at their original positions - if the disks are swapped, the controller may have a hard time finding out what the original layout was.

Some controllers are smart enough to detect swapped disks and absorb the change - at boot time, the controller's BIOS only reports the change and asks the administrator to acknowledge or reject the change.

In a less fortunate scenario, after the disks are swapped, the controller may detect two degraded arrays instead of one that seems all right.

A common moment for misbehaviors of this type to occur is just when the administrator is trying to rebuild a degraded array after a disk drive failure - a spare disk is inserted and lo and behold, another degraded/failed array appears and the controller refuses to use the newly inserted disk for the intended rebuild.

The reason is, that the replacement disk is not clean - it was probably a member of some long forgotten array, from a disassembled computer or left over after some lab experiments with array configurations.

If the controller comes with reasonable software, this "ghostly" array can be simply unconfigured and the disk can be re-initialized. If for some reason this procedure fails, such a RAID-cursed disk must be erased - overwritten with all zeroes, which should make it behave like factory-clean. This erasure operation has to be performed on any ordinary (non-RAID) controller, IDE or SCSI respectively.

The easiest way to perform the erasure is under any Unix using the following command:

```
    cp /dev/zero /dev/<block_device_of_the_disk>
```

There's also an alternative command with the dd program:

```
    dd if=/dev/zero of=/dev/<block_device_of_the_disk>
```

The latter however isn't quite practical - the dd program does a sync() after every block written, as a result of which the erasure lasts significantly longer than using cp. Increasing the block size sure helps, but it's still more appropriate to use cp or cat - these won't leave an "indivisible residue" at the end of the disk and the write operation will be faster anyway, because cp and cat make use of the kernel's write-back buffering.

In Linux, the disk's block device will be called /dev/hda to hdf or /dev/sda to sdf - hd = an IDE drive, sd = a SCSI drive, followed by an ordinal letter. With a greater number of channels and disks, the last letter can be higher than "F".

Perhaps there is a DOS util that could achieve the same effect – some people recommend the Norton Wipe-info in the "whole volume" mode. Straight editing of disk sectors can be done e.g. by the Norton Disk Editor.

When a plain mirror is configured, most controllers store the data on the disks in such a way that, if a member disk is unmounted from the array and attached to an ordinary (non-RAID) controller, it usually contains a correct partition table and the operating system even tries to boot. This works especially with the simpler controllers that strictly mirror a whole disk - i.e., those that can't set up multiple "mirrored logical volumes" (virtual disks) across two physical disks.

Therefore it would seem that, when a mirror is set up, the simpler RAID controllers leave the MBR with a seemingly correct partition table on the disk. The RAID-private data block (also known as the "superblock") is stored somewhere in an "invisible space" that was left intentionally unallocated for array payload.

If such a "RAID-possessed" disk is to be re-used on an ordinary disk controller, it is appropriate to purge it first. A RAID-possessed MBR may contain incorrect information about the start and end of useable space etc. - when used directly by an operating system, the disk might misbehave. This applies even if the disk is re-partitioned using FDISK without zapping the whole partition table in advance.

To remove the "RAID curse" before the disk is used directly by an operating system, it is enough to pad the first few sectors with all zeroes - that can be arranged using the dd command with the following arguments:

```
 dd if=/dev/zero of=<disk block device> bs=1k count=1000
```

After this operation, fdisk won't find a partition table and will act as if the disk was factory-clean (and will create a new partition table from scratch).

Unfortunately, deleting just the MBR and the first few sectors is not always enough. Especially if the disk is to be re-used in a RAID, it is also appropriate to erase the RAID superblock, to prevent the controller from detecting long-forgotten arrays that were not properly unconfigured before the disk was unmounted and put aside. In his case, it is necessary to rewrite the whole disk with zeroes - it takes time, but the result is flawless.

One last note for inquisitive minds: on a physical disk, the aforementioned "invisible" RAID superblock is stored typically right after the MBR or at the very end of the disk (e.g., in the last cylinder/track etc.). Its precise position can be found e.g. by setting up an empty array on a zero-padded disk and by searching the whole disk for non-zero data on an ordinary controller. The knowledge of the superblock's precise position can be used for rapid cleaning of "RAID-possessed" disks.

To sum up, the morale is that it is very appropriate to erase any RAID-possessed disk that's being removed from an array and stored for later use - that way it can't cause any harm later on.

Other notes

Performance aspects of the stripe size

With RAID-0 and RAID-5, the stripe size affects performance. With most RAID controllers, the stripe size can be selected.

Common sense says that with small average file size (or more precisely average transfer size) on the host file system, small stripe size improves efficiency, most notably with RAID-5 in write mode – whereas with large files/transfers, large stripe size reduces processing overhead. The particular influence level differs from vendor to vendor and from model to model and can be reduced by a large cache.

The available sources relate the stripe size to two other factors:

1) the "block group size" or "cluster size" of the filesystem applied on top of the RAID volume.
   In order to evenly distribute the beginnings of clusters among different disks, the cluster size should not be equal to the "stripe set size", also known as the "stride size", nor an integer multiple. See the Linux Software RAID HOWTO, chapter 4.8 and 4.9. This requirement is ultimately stringent with a small average file size – apparently, it is not much of an issue with database data that typically uses large files in the host file systems or even proprietary LBA-level storage. Special rules may apply in this respect depending on the particular DBMS discussed.
2) the track size of the physical disks used
   This factor is mentioned in the 3ware TwinStor "white paper" (see references). The physical track size of modern disk drives may be hard to find out, hard to align to stripe size and can even be changing with track radius on the plate.

The avalanche effect

Some sources claim that with RAID5, upon array degradation and rebuild, the array can show an interesting (and fatal) avalanche effect – the first failed disk is followed by others in a short succession.

The general cause is that the array does not periodically check for defects throughout the whole space of all the physical disks involved. Less used files can be lying unaccessed for quite some time – such space is never checked on any of the disks. Besides, e.g. a mirror array typically optimizes disk accesses for lowest latency by reading from only one of the mirrored drives – from the one whose head is closest to the desired sector. And let's not forget about the free space on the virtual array volume that's obviously never read, either.

All of this means that several bad sectors at different locations of the various disks can escape the RAID controller's attention for quite some time. When the array discovers a bad sector on one of the drives, it signals an error, the administrator replaces the failed drive and starts a rebuild to renew the redundancy. The redundancy rebuild consists in walking the whole space of the original functional drives and calculating the contents of the replaced drive. While walking the drives in this way, the controller may discover further bad sectors in an unused area of yet another drive that so far went unnoticed. Which may seem as if "two drives failed at once". This may be true after all, to a degree – significantly faulty series of disks have occured several times in the past with various manufacturers. Nevertheless, it's not quite likely that two disks would fail within several hours or minutes.

The only cure against the avalanche effect is regular background checking of the surface of all drives in the array – so that newly appearing bad sectors are reported and the affected disk can be replaced before another disk fails. Some RAID controllers have a configuration option with exactly this effect – enabling an on-the-fly surface check running in the background. Obviously the cost is a slight decrease in array throughput.

Individuals on the internet forums at http://www.root.cz have reported that there is another substantially different scenario resulting in several disk drives failing at once. Some disk drives that have been running for a long time won't restart after a power cycle – the spindle bearing gets so tight that the disk can keep spinning at a constant RPM but won't spin up after a power cycle. There are no bad sectors, the surface is fine – only the drive's motor won't move the clutched bearing anymore. The risky period of keeping on going may last for months. Given that disks from the same manufacturing batch tend to share similar properties, it may easily happen that several identical drives in a RAID are well into the grey zone upon an accidental powercycle.

Different people derive different morales – the most important ones are:
1) don't use identical drives (certainly not from the same batch) in one RAID
2) when one drive fails after a long uptime, try not to power-cycle the array before you make a backup of the data, you hot-swap the drive and the array rebuilds.

Emergency operation of a failed array

Common wisdom has it that when more than one drive fails in a RAID1 or RAID5, the array collapses irreversibly. The controller tags both the disks as failed and categorically refuses any access to the array, even though the failure may really span only a few sectors on each of the disk drives. Maybe none of the two failed drives has crashed fatally, both respond to AT commands and the unaffected sectors can actually be read.

That's why some decent controllers allow emergency operation of such a failed array. If multiple disks in an array have bad sectors that however don't clash in a particular stripe set, the controller can reconstruct valid data from the arbitrary functional sectors. Even if some stripe sets are indeed doomed, the controller simply reports a problem with this particular sector offset and the surrounding sectors of the virtual drive remain accessible.

No doubt that the emergency operation is a last-resort solution – the last hope of rescuing at least some data from the sectors that are still functional. The essential trouble is that a hard drive usually wastes a lot of time (many seconds) attempting to read a hopeless sector before finally returning an error – so that even if the controller caches the bad block information, the emergency mode can significantly decrease performance of the array.

How to simulate a disk failure

The documentation that comes with some RAID controllers claims that the only correct way to simulate a disk failure is to use menu XY and select item "fail drive". And that ripping a disk drive out of a hot-swap bay is not quite a correct way to simulate a disk failure.

In contrast, the excellent Linux "software RAID HOWTO" explains without prejudice that a "true" disk failure can not really be simulated – at all. The reason is not that the HOWTO discusses a pure software-based solution. The reason is, that there are several basic categories of typical failures, some of which cannot really be simulated without actually damaging the drive.

Some of the real-world failures are perhaps these:

1) bad sectors occuring. The disk as a whole is alive but can't read data from some sectors. The failure is usually caused by dust particles introduced into the clean compartment with the revolving plates, or by a mechanical defect to the bearings (either the spindle bearing or the heads' arm's hinge). This type of failure usually spreads quickly across the surface.

2) failure in the drive's electronics. The drive's control electronics dies – stops responding to the host computer's commands. If the failure reaches as far as the bus drivers, it can jam the whole IDE channel.

3) poor contact in the connectors or a cable failure. Such a failure can cause bitwise errors resulting in incorrectly stored data (which the array usually fails to detect!) or even detectable parity errors (and array degradation). The disk is not damaged – when the poor contact is tracked down and repaired, the disk can be used again.

4) a completely disconnected data cable. The disk is not damaged – when the issue is solved, the disk can be used again.

5) power outage (a disconnected power line connector). The disk is not damaged – when the issue is solved, the disk can be used again.

Only the cable failures can be simulated, by disconnecting the power or data cable. Performing such experiments without hot-swap bays may seem slightly risky or even drastic. If the disk is disconnected while the whole system is powered down, the simulation is safe, but doesn't test the behavior of the controller upon runtime failures – and, with real-world controllers, many deficiencies can be identified in this area.

RAID is not resistant to bitwise errors

Bitwise errors can occur due to poor contact in cabling or due to an especially insidious disk drive failure.

No RAID performs background parity checking of the stored data - not even the redundant varieties. The redundant information is only kept for emergency cases (array degradation and rebuild). Unless the controller detects a drive failure, it never checks consistency of data vs. parity in the redundant stripe sets, not even in those being actually read. During normal read operations over the virtual volume, the redundant data is not even read from the physical disks.

RAID is not a replacement for proper backups!

By deploying a redundant RAID (a RAID1 or a RAID5), the administrator is by no means relieved of his basic responsibility for doing regular backups.

Different controllers feature a different level of capabilities

The hardware, the firmware, the BIOS, the drivers and the administration software coming as part of different RAID solutions have a very different level of capabilities and there is really no such thing as a perfect RAID controller, that would be flawless in every aspect under under any operating system. For a limited period of time, a particular manufacturer may shine bright like a star on a warm summer night, but there's not guarantee that his products won't soon be turned into a dangerous unsupported relic by the frenetic hardware innovation rate and improvements (and “improvements”) on part of operating systems.

There are decent hardware controllers that, due to poor drivers and administration software, can't manage a background rebuild or even cope with a disk failure (array degradation) in a manner transparent to the operating system. On the other hand, there are cheap host-based products that almost secretly emulate all the important functionality in drivers, and are so good at it that in some operating systems they almost rival the quality hardware-based products.

New versions of commercial and open-source operating systems are released almost every day. Every new release brings a risk (a certainty?) that the vendors (and the volunteer enthusiasts) didn't have enough time to properly test and update the drivers internally for the new release of the operating system.

New models of hardware are launched every day, too. Especially under the OpenSource OS'es, every new hardware revision (let alone a model) means a risk (a certainty?) that the old drivers won't function with the new hardware, even though informal sources who have relationships with the vendor claim that “there was no change to the API”.

The morale is: any new hardware has to be tested before deployment or before mass purchase under the planned operating system. If you can't do that yourselfs, find someone who does that for you.

The harder the training camp, the easier the battlefield

In mission-critical applications (which is frequently the case with RAID systems) it is very appropriate that the administrator tries to simulate an outage and get the array repaired. Especially on part of UNIX operating systems the documentation tends to be relatively weak, which becomes especially significant upon RAID degradation. In such situations, it is invaluable to have at least a vague memory of the restoration process. Any notes regarding the proper rebuild procedure acquired in practical crash tests are worth their weight in gold.

References

Anandtech's storage section
http://www.anandtech.com/storage/index.html

A somewhat dated ATA RAID review by Anandtech
http://www.anandtech.com/storage/showdoc.html?i=1491

Intel IOP processors
http://www.intel.com/design/iio/

Intel IOP315 – the page contains a comparison table with several IOP chips
http://www.intel.com/design/iio/iop315.htm

i960 and IOP chip manuals
IOP321 (Xscale) ftp://download.intel.com/design/iio/manuals/27351702.zip
IOP303 (i960 incl. XOR) ftp://download.intel.com/design/iio/manuals/27335301.zip
i960RD (no HW XOR) ftp://download.intel.com/design/iio/manuals/27273602.pdf

Busses and bridges in the PC
http://www.rambus.com/rdf/rdf2002/pdf/2FeibusIntro.pdf

Adaptec RAID controllers
http://www.adaptec.com/worldwide/product/prodtechindex.html?sess=no&language=English+US&cat=%2fTechnology%2fRAID

Intel RAID controllers
http://www.intel.com/design/servers/buildingblocks/raid.htm?iid=ipp_home+server_raid_cont&

LSI Logic (Mylex a MegaRAID) RAID controllers
http://www.lsilogic.com/products/stor_prod/raid/index.html

ICP Vortex RAID controllers
http://www.icp-vortex.com/english/product/prod_e.htm

3ware homepage and PR-flavored descriptions of their StorSwitch and TwinStor architectures
http://www.3ware.com
http://www.fcenter.ru/articles.shtml?hdd/4243
http://www.3ware.com/products/pdf/twinstor.pdf

Promise S150 SX4 a SX4000 RAID controllers
http://www.promise.com/product/product_detail_eng.asp?productId=112&familyId=2
http://www.promise.com/product/product_detail_eng.asp?productId=94&familyId=2#

External RAID controllers by InforTrend
http://www.infortrend.com

Linux Software RAID HOWTO
http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Software-RAID-HOWTO.html

The Linux Documentation Project (an index of HOWTO's and other documentation)
http://www.tldp.org

Appendix – a program searching for sectors with non-zero data

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
/* #include <fcntl.h> */
#include <asm/fcntl.h>
#include <errno.h>

void usage()
{
   printf("Usage examples:\n");
   printf(" dscan /dev/sdc\n");
   printf(" dscan /dev/hda\n");
   printf(" dscan ./my_ordinary_file\n");
}

int main(int argc, const char** argv, const char** env)
{
   int fd, i;
   unsigned int sector=0, a_hundred_megs=0;
   char data[512];

   if (argc < 2)
   {
      usage();
      exit(1);
   }

   fd = open(argv[1],O_RDONLY|O_LARGEFILE);
   if (fd < 0)
   {
      perror("Error opening input");
      usage();
      exit(1);
   }

   while ( read(fd,data,512) == 512 )
   {
      for (i=0; i<128; i++)
      {
         if ( ((unsigned int*)data)[i] != 0 )
         {
            printf("Non-zero data found at sector %u\n", sector);
            break;
         }
      }

      sector++;
      a_hundred_megs++;
      if (a_hundred_megs == 200000)
      {
         fprintf(stderr, ".\n");
         a_hundred_megs = 0;
      }
   }

   perror("Seems like we're finished");
   close(fd);

   exit(0);
}
```