



Surviving Two Disk Failures

Introducing Various RAID 6 Implementations

x y r a t e x •

Notices

The information in this document is subject to change without notice.

While every effort has been made to ensure that all information in this document is accurate, Xyratex accepts no liability for any errors that may arise.

© 2007 Xyratex (the trading name of Xyratex Technology Limited). Registered Office: Langstone Road, Havant, Hampshire, PO9 1SA, England. Registered number 03134912.

No part of this document may be transmitted or copied in any form, or by any means, for any purpose, without the written permission of Xyratex.

Xyratex is a trademark of Xyratex Technology Limited. All other brand and product names are registered marks of their respective proprietors.

Robert Maddock, Nigel Hart & Tom Kean

Issue 3.0 | October, 2007

Contents

Introduction	2
Rationale	2
1. A second disk drive failure	2
2. A read error during rebuild	3
Repair error	3
What can be done?	3
Various design for multiple redundancy.....	4
1. Triple mirroring.....	4
2. RAID-51	4
3. RAID 5&0	4
4. RAID-6	4
RAID-6.....	5
Reed-Solomon code	5
XOR Techniques	6
Rows and Columns.....	6
Horizontal and Diagonal parity	7
X-Code.....	9
ZZS (the 'Russian' code).....	10
Park (IBM).....	10
Hybrid Approaches.....	11
References.....	12
About Xyratex	12

Introduction

The following white paper discusses the reasons why RAID 6 has become more interesting to implement in Storage controllers in recent years. This is in part due to the adoption of high capacity (SATA) disk drives which has meant that the likelihood of failure during a rebuild operation has increased.

The paper discusses the implications generically for RAID 6 algorithms against other techniques to protect against double drive failures. There are multiple RAID 6 algorithms and these are discussed at a high level. Each of these has some differences in their abilities, but generically some applications are more easily deployed onto RAID 6 arrays than others.

Rationale

The capacity of disk drives is continuously increasing, and the cost of storage is falling. So users are storing more and more data. But the reliability of disk drives does not seem to be improving (and moving to ATA drives may make it worse). So the number of data loss incidents in an installation is probably increasing. Customers of course want the number to fall, since dealing with such incidents is a time-consuming and expensive operation. The ideal would be a storage product that 'never loses data'. Is this feasible?

Many storage controllers today offer RAID arrays that can survive the failure of one disk drive without losing data. RAID-5 and RAID-1 (mirroring) are the most common examples. When a disk fails, it will be replaced and the contents rebuilt from the information on the remaining disks. There are two ways in which a further failure can lead to data loss.

1. A second disk drive failure

If a second disk fails before the replace and rebuild is complete, the whole RAID array is lost. All the LUNs on this array will be lost, and if virtualisation is being used, sections of many LUNs may be lost. To reduce the chance of this happening, it is important to do the replace and rebuild as quickly as possible. Having a hot spare available allows the replacement to happen immediately, and rebuilding quickly reduces the exposure time, at the cost of a more severe performance reduction for normal operations. But as disk capacity increases, the disk data rate usually increases by a smaller amount, so the rebuild time increases, and with it the chance of data loss.

Server class disk drives have a specified MTBF of 1 million hours or more. So an 8+P RAID-5 array will suffer a disk drive failure every 111,111 hours. With a replace and rebuild time of 10 hours, the chance of a second failure in an 8+P RAID-5 array, for example, is 80 in 1,000,000. So the chance of array loss is once every 1.4E9 hours.

This sounds as though the chance of having an array failure is incredibly small, but it is doubtful whether the 1,000,000 hour MTBF figure is real. It isn't clear that any disk drives really reach this. Certainly some models of disk drive are worse than others, and if the factory builds a bad batch, they may well end up in a drawer

together built into a RAID array. If the figure is ten times worse, the incidence of array loss is 100 times worse. If the customer has 100 RAID arrays, the figure is 100 times worse again. Now we are at 1.4E5 hours, about 15 years. So over a ten-year life it is likely to happen.

All this assumes the drive failures are independent events. Even worse, although difficult to quantify, are situations where some influence common to the drives of an array leads to failures. Stiction after a power failure was a notorious example in the past, but various environmental problems could cause linked failures.

2. A read error during rebuild

To rebuild the contents of a failed disk drive in an 8+P RAID-5 array, we must read all the data on each of the 8 remaining disk drives. Server class drives specify an unrecoverable read error rate of about 1 in 1E14 bits. It isn't clear exactly what this means, and disk drive manufacturers provide little detail, but it suggests that reading eight drives of 300GB each, i.e. $8 \times 300 \times 10^9 = 2.4 \times 10^{13}$ bits, will have about one chance in 4 of a read error. If such a read error occurs, there is at least one block of the data on the array, which is lost. With or without virtualisation, this will be one block in one LUN. This will happen every 4 disk drive failures, i.e. every 4.4E5 hours. If the customer has 100 arrays it will happen every 4.4E3 hours – every 6 months.

This is with disk drives that meet their specification. If they are ten times worse, it happens 100 times more often – every other day.

Repair error

Apart from data loss, there is one more argument that has been advanced for multiple redundancy. Apparently, it is quite common that when the repairman arrives to repair a RAID-5 array in which a disk has failed, he pulls out the wrong disk. Although this should not lead to long-term data loss, it certainly causes loss of access to data until the mistake is rectified. In an array with multiple redundancy, access would not be interrupted.

What can be done?

It seems unlikely that disk drive manufacturers will come up with orders of magnitude improvements in reliability, as long as we are using spinning platters and magnetics. So the only way to improve things substantially is to use arrays which can survive two (or more) failures without losing data.

There are many different designs for arrays with multiple redundancy, with different advantages and disadvantages. The main trade-off is, as always, between cost and performance. The more disks that are used for a given amount of data, the higher the cost. With disks capable of a given number of operations per second, the number of disk operations required to implement each read or write to the controller determines performance.

(There are cost and performance trade-offs in the controller itself as well. If the algorithms used are complex, the controller is likely to be more expensive, both to develop and to manufacture, for a given level of performance.)

Various design for multiple redundancy

1. Triple mirroring

Three copies of the data are stored on three separate disks. It is expensive, since it uses three times as many disks as JBOD. But it is simple, and it performs quite well. Each read requires one disk read, from any of the three copies, and each write requires three disk writes: one to each copy.

2. RAID-51

Copies of the data are stored on two separate RAID-5 arrays, on separate disks of course. This is moderately expensive, since it uses more than twice as many disks as JBOD. The performance is quite good for reads, but modest for writes, since two RAID-5 writes are required. Each read requires one disk read, from either copy. Each write requires a read-modify-write to two disks on both RAID-5 arrays, although the reads only need to be done from one, making a total of six operations.

As with RAID-5, long sequential writes can be optimised into full-stride writes.

RAID-51 is particularly suitable when the two RAID-5 copies can be geographically separated, since complete failure of either site leaves one complete RAID-5, providing all the data with continuing redundancy.

3. RAID 5&0

A variant of RAID-51 is to store one RAID-5 copy of the data and one RAID-0 copy. This reduces the cost slightly (only one copy of parity) and improves the performance as well, since writing to the RAID-0 copy is more efficient. Writes now require read-modify writes to two disks on the RAID-5, and one disk write on the RAID-0.

4. RAID-6

The term RAID-6 here is used to describe any technique that stores strips of user data plus two derived redundancy strips, similar to the one parity strip in RAID-5. These redundancy strips are such that the data can be recovered after the loss of any two data or redundancy strips. The name RAID-6 should be used with caution, since it has been used for various different things (any salesman would see it as one better than RAID-5!). Another term is RAID-DP (double parity), but this is confusing too, since Network Appliance use this for their double parity technique, and HP use the even more confusing term RAID-5DP for the method used in their VA-7000 product.

RAID-6 offers low cost, since only one copy of the data is stored, plus the redundancy strips. The performance is poor, although the performance per physical disk is no worse than RAID-51. Each read requires one disk read. Each write requires a read-modify-write on three disks. As with RAID-5, long sequential writes can be optimised.

The following table summarises the characteristics of these various methods and compares them to common methods with single or no redundancy.

			Measure of cost		Measures of performance		
	Protection against failure	Total number of disks for N data disks	Ratio of total:data (for N=8)	Disk ops per write	Relative throughput 70R:30W (for N=8)	Relative throughput with one disk failed	Relative throughput per disk
JBOD	none	N	1	1W	1	-	1
RAID-5	any one disk	N+1	1.125	2(R+W)	0.59	0.39	0.52
RAID-1/10	any one disk	2N	2	2W	1.54	1.47	0.77
RAID-6	any two disks	N+2	1.25	3(R+W)	0.5	0.36	0.4
RAID-51	any three disks	2N+2	2.25	2(R+W)+2W	0.9	0.65	0.4
RAID-5&0	any two disks	2N+1	2.125	2(R+W)+1W	0.97	0.78	0.46
Triple mirror	any two disks	3N	3	3W	1.89	1.85	0.63

It is worth noting that the probability of three independent failures is very small, probably less likely than a geographical failure - fire, flood, earthquake, etc. - and so geographical separation is probably more useful than RAID-51 on one site. Such separation can be implemented with RAID 1/10, RAID-51, RAID 5&0, or triple mirroring.

RAID-6

There are several ways of providing the double redundancy of RAID-6, and developing new ones has been an active field of research in recent years. These fall into two categories, those that use the standard XOR techniques as deployed in RAID 5 implementations and those that use redundant polynomial equations such as Reed-Solomon codes to recompute the missing data.

Reed-Solomon code

There are well understood algorithms for taking N data bits and generating M more bits, such that the data can be calculated when several of the N+M bits are incorrect or have been lost. The development of these methods has taken place mainly in communications technology, and goes back more than fifty years. Such codes are also used in the error-correcting code (ECC) used with semiconductor memory.

RAID-5 is of course a simple example of such a code, with N data bits and 1 parity bit, where the data can be retrieved after the loss of any one bit.

For multiple redundancy, the most popular encoding seems to be the Reed-Solomon code, which can be used with any number of check disks. The calculation of the check fields must be done using data words of more than one bit from each data disk. The larger these words, the more data disks can be protected - the important equation here is

$$d = 2^b - 1 - c$$

Where d is the number of data disks, c the number of check disks, and b the number of bits in a word. From this it can be calculated that using bytes of data, up to 253 data disks can be protected against double disk loss by two check disks. So it will be seen that the method is more than sufficient for today's storage systems.

The main difficulty with this technique is that calculating the check codes, and reconstructing data after failures, is quite complex. It involves polynomials and thus multiplication, and requires special hardware, or at least a signal processor, to do it at sufficient speed. With general purpose processors getting faster and faster, maybe this will change, but up to now it has been the case. The use of Reed-Solomon in communications means that hardware components are available for it.

But since most existing storage controllers do not have this hardware, but do have exclusive-OR hardware, there has been more emphasis in recent years on techniques that use exclusive-OR. Whereas Reed-Solomon, like RAID-5, is 'one-dimensional', i.e. a vector of data words across the data disks generates the corresponding check words, all the exclusive-OR techniques seem to need some kind of two-dimensional design.

XOR Techniques

The following are a selection of IP references to solutions to double drive parity/R6 using XOR techniques. Most RAID controller contain dedicated XOR hardware and therefore their designs are extendable to adopt one or other of these techniques.

Rows and Columns

Perhaps the simplest of these is 'rows and columns'. In this, an array of M x N data disks is used, with M parity disks in one dimension and N parity disks in the other.

Disk 11	Disk 12	Disk 13	Disk 14	Disk 1N	Parity 1x
Disk 21	Disk 22	Disk 23	Disk 24	Disk 2N	Parity 2x
Disk 31	Disk 32	Disk 33	Disk 34	Disk 3N	Parity 3x
..
Disk M1	Disk M2	Disk M3	Disk M4	Disk MN	Parity Mx
Parity x1	Parity x2	Parity x3	Parity x4	Parity xN	

In the diagram, each cell represents one disk. There are M x N data disks. The M disks in the rightmost column contain the parity of the data disks in the same row; the N disks in the lowest row contain the parity of the data disks in the same column.

When data is written, we must do read-modify-write on the data, the horizontal parity disk and the vertical parity disk. If a disk fails, if there is only one failed disk in the same column the data can be reconstructed using

the disks in that column, just like RAID-5. Similarly if there is only one failed disk in the same row. So the data can always be reconstructed after two failures, and very often after more.

The disadvantage of the technique is, of course, that for the check disk overhead to be small, there have to be a lot of disks, so the method is really only suitable for very large storage systems where there are always many disks. Note, however, that the reconstruction does not involve all the disks, so the performance after failure is no worse than RAID-5 with N or M disks. It is possible to perform 'full-stride writes', but they do have to be very long because of the number of disks.

The method as shown here puts data and parity on separate disks, with the possible uneven load that this entails (as with RAID-4). It might be possible to design a rotating RAID-5-like version of this, but it is a lot more complicated, and would need the extra disk in the corner.

Horizontal and Diagonal parity

As an alternative to such a large number of disks, it is possible to use multiple stripes on one array of disks as the way to get the 'second dimension'. This is done in the 'RAID-DP' method used by Network Appliance.

Data Disk 1	Data Disk 2	Data Disk 3	Data Disk 4	Horizontal Parity Disk	Diagonal Parity Disk
Data1	Data2	Data3	Data4	Parity	DParity1
Data	Data1	Data2	Data3	Parity4	DParity2
Data4	Data	Data1	Data2	Parity3	DParity3
Data3	Data4	Data	Data1	Parity2	DParity4

The diagram shows a 4+2P array, but any number of data disks can be used. The data disks and the horizontal parity disk are a conventional RAID-4: the parity is the exclusive-OR of the data strips in the same row. The diagonal parity strips are the exclusive-OR of the other strips marked with the same number. Note that the horizontal parity does take part, and that there are some strips that do not contribute to a diagonal parity. Surprisingly, it is possible to reconstruct everything after any two-disk failures.

Read operations are read directly from the disks where the data is stored. Short write operations require read-modify-write operations to the data, the horizontal parity, and usually the diagonal parity. But most writes also require read-modify-write to the diagonal parity to which the horizontal parity contributes. On average, a short writes requires 3.5 read-modify-write operations, more than the minimum three used by some other designs, so short writes are not efficient. Full stride writes work well, but the data length has to be $N*N$ strips.

In the Network Appliance implementation, the parity is not distributed, so the two parity disks do not contribute to read performance. However, the design is easily extensible to allow rotation of successive blocks of $N*(N+2)$ strips.

The algorithm to recover from the loss of a single drive in the array is the normal RAID 5 degraded array reconstruction, using the horizontal parity.

The algorithm to recover from a double drive failure involves a number of stages of using alternating Q and P stripe reconstructions. These in various sequences allow full reconstruction of the original data. The following worked example is a simple case to show the sequence.

Note that the sequence of operations becomes larger as the width of the array becomes wider.

Taking the previous layout of relationship between :-

Data Disk 1	Data Disk 2	Data Disk 3	Data Disk 4	Horizontal Parity Disk	Diagonal Parity Disk
Data1	Data2	Data3	Data4	Parity	DParity1
Data	Data1	Data2	Data3	Parity4	DParity2
Data4	Data	Data1	Data2	Parity3	DParity3
Data3	Data4	Data	Data1	Parity2	DParity4

Should drives 1 and 2 fail, the situation would be :-

Data Disk 1	Data Disk 2	Data Disk 3	Data Disk 4	Horizontal Parity Disk	Diagonal Parity Disk
Data1	Data2	Data3	Data4	Parity	DParity1
Data	Data1	Data2	Data3	Parity4	DParity2
Data4	Data	Data1	Data2	Parity3	DParity3
Data3	Data4	Data	Data1	Parity2	DParity4

Note that there will be a single strip of data that can be re computed :-

Data Disk 1	Data Disk 2	Data Disk 3	Data Disk 4	Horizontal Parity Disk	Diagonal Parity Disk
Data1	Data2	Data3	Data4	Parity	DParity1
Data	Data1	Data2	Data3	Parity4	DParity2
Data4	Data	Data1	Data2	Parity3	DParity3
Data3	Data4	Data	Data1	Parity2	DParity4

From this a standard RAID 5 can recomputed the row :-

Data Disk 1	Data Disk 2	Data Disk 3	Data Disk 4	Horizontal Parity Disk	Diagonal Parity Disk
Data1	Data2	Data3	Data4	Parity	DParity1
Data	Data1	Data2	Data3	Parity4	DParity2
Data4	Data	Data1	Data2	Parity3	DParity3
Data3	Data4	Data	Data1	Parity2	DParity4

Another Q drive relationship can be used to compute an additional strip :-

Data Disk 1	Data Disk 2	Data Disk 3	Data Disk 4	Horizontal Parity Disk	Diagonal Parity Disk
Data1	Data2	Data3	Data4	Parity	DParity1
Data	Data1	Data2	Data3	Parity4	DParity2
Data4	Data	Data1	Data2	Parity3	DParity3
Data3	Data4	Data	Data1	Parity2	DParity4

And so the process goes on until fully reconstructed.

X-Code

A more balanced design is called 'X-Code' (ref 4), for reasons which will become clear.

Disk 1	Disk 2	Disk 3	Disk 4	Disk 5
Data11	Data22	Data33	Data44	Data55
Data52	Data13	Data24	Data35	Data41
Data43	Data54	Data15	Data21	Data32
Parity3X	Parity4X	Parity5X	Parity1X	Parity2X
ParityX4	ParityX5	ParityX1	ParityX2	ParityX3

The diagram shows the layout of an X-Code array of 5 disks. Each box in the diagram represents a strip of data or parity on a disk. ParityXn is the exclusive-OR of the three data strips DataXn, and ParitynX of the three data strips DatanX. The groups of data strips and the parity strips generated from them run in both diagonal directions, hence 'X-Code'. The layout of 5x5 strips shown in the diagram is then repeated across the whole of the 5 disks.

With this encoding, data is read directly from the disks where it is stored, as in RAID-5. On short writes, the data, and the two corresponding parities, must be read, modified and rewritten. It is possible to write in 'full stride mode', but the length of data required is 5x3 strips. If a disk fails, it is possible to reconstruct each lost data strip by calculating exclusive-OR of the other strips in either of the parity diagonals though the strip. If two disks are lost, it is still always possible to find a diagonal with only one missing strip, and reconstruct that one. Then there will be another diagonal with only one missing strip, and so on. In practice it is probably easiest to reconstruct a whole 5x5 section at a time. Both the calculation of the parities and the reconstruction are done entirely with exclusive-OR operations.

The data and parity are distributed, so that the load on the disks is even.

ZZS (the 'Russian' code)

In 1983, three Russians, Zaitsev, Zinov'ev and Semakov (ZZS) published a method which can be applied to disk storage. The paper is in Russian (ref 5), but is described in (ref 6). It is similar to X-Code, but uses fewer stripes on each disk in the repeat pattern.

Disk 1	Disk 2	Disk 3	Disk 4	Disk 5	Disk 6	Disk 7
Data0	Parity0	Parity1	Parity2	Parity3	Parity4	Parity5
Data1	Data2	Data3	Data4	Data5	Data6	Data7
Data8	Data9	Data10	Data11	Data12	Data13	Data14

The number of disks is a prime number (p), and the number of stripes in the repeat pattern is $(p-1)/2$.

Each data strip contributes to two parity strips on other disks, and the parity is computed as the exclusive-OR of the data strips. Which strips are related is not a straightforward geometric pattern, as in X-Code. It is the result of an algorithm, but in an implementation it might be easier to use a table-driven technique.

Read operations proceed as usual by reading the data directly from the disks where it is stored. Short write operations require three read-modify-write operations, and 'full-stride' write operations are possible - the length required is about half that with X-Code.

If a disk fails, each data strip on it can be reconstructed by exclusive-OR of the remaining data and parity strips in either of the parity sets to which it belongs. If two disks fail, it is always possible to find a parity set with only one missing strip, and reconstruct that one first, etc. The parity is distributed, although the load on the disks is not completely even because of the one disk without parity.

This method still has the disadvantage of requiring a prime number of disks, although it is possible to use it with one less than a prime number, by omitting Disk1, and instead using an imaginary Disk1 with all zeroes recorded on it.

Park (IBM)

Park, working at IBM Yorktown, discovered that methods very similar to ZZS could be used for arrays of various numbers of disks. He did this simply by exhaustive computer search, as described in his patent (ref 7), and discovered suitable data-parity patterns for all array sizes from 3 to 38 disks, with the exception of 8 and 9. (Lihao Xu subsequently found a method for 8 and 9 disks). Since the methods were discovered by search, rather than by algorithm, an implementation would have to be table-driven. As with ZZS, the repeat pattern is $(N-1)/2$ for an odd number of disks, and $N/2$ for an even number. Park always has one parity strip on each disk - he does not seem to have discovered that one disk in an odd number can contain all data.

Disk 1	Disk 2	Disk 3	Disk 4	Disk 5	Disk 6
Parity0	Parity1	Parity2	Parity3	Parity4	Parity5
Data0	Data1	Data2	Data3	Data4	Data5
Data6	Data7	Data8	Data9	Data10	Data11

The operation of Park's arrays is similar to ZFS arrays, and they have similar characteristics, with the further advantage of allowing any number of disks.

Hybrid Approaches

RAID 6 P+Q is a hybrid approach of the combination of XOR and polynomial codes. The following table demonstrates the terms that compute the 2 parity fields P&Q.

Stripe	0	1	2	3	4	5	6
0	D(0,0)	D(0,1)	D (0,2)	D (0,3)	D (0,4)	P (0)	Q (0)
1	D(1,0)	D(1,1)	D (1,2)	D (1,3)	P (1)	Q (1)	D (1,4)
2	D(2,0)	D(2,1)	D (2,2)	P (2)	Q (2)	D (2,3)	D (2,4)
3	D(3,0)	D(3,1)	P (3)	Q (3)	D (3,2)	D (3,3)	D (3,4)
4	D(4,0)	P (4)	Q (4)	D(4,1)	D (4,2)	D (4,3)	D (4,4)
5	P (5)	Q (5)	D(5,0)	D(5,1)	D (5,2)	D (5,3)	D (5,4)
6	Q (6)	D (6,4)	D(6,0)	D(6,1)	D (6,2)	D (6,3)	P (6)

P is computed as a standard XOR of D0-4 in this case

Q is a polynomial based on $Q(j) = (k0 \otimes D(j,0)) \oplus (k1 \otimes D(j,1)) \oplus \dots \oplus (k4 \otimes D(j,4))$. This has the characteristic that it can compute 1 term if 2 terms are lost.

Reconstruction in the case of a double drive failure is a process of recreation of 1 of the D values from the Q term and then a standard XOR to compute the final term.

As in this example, the stripes are rotated to balance read IO's over all of the disks in the array.

References

1. http://www.hp.com/products1/storage/products/disk_arrays/infolibrary/mathematics_behind_RAID_5DP.pdf
2. http://www.hp.com/products1/storage/products/disk_arrays/infolibrary/analysis_Of_RAID_5DP.pdf
3. http://www.netapp.com/tech_library/3298.html
4. L. Xu and J. Bruck, X-Code: MDS Array Codes with Optimal Encoding, IEEE Trans. on Information Theory, 45(1), 272-276, Jan., 1999
5. G. V Zaitsev, V.A. Zinov'ev and N.V. Semakov, "Minimum-Check-Density Codes for Correcting Bytes of Errors, Erasures, or Defects", Problems of Information Transmission 19(3), 1983.
6. M. Blaum, R. M. Roth, On Lowest-Density MDS Codes, IEEE Trans. on Information Theory, 45(1), 46-59, Jan. 1999 pdf
7. US patent 5,862,158, Baylor, Corbett and Park (IBM), 1999
8. US patent 5,485,474, Rabin (Harvard) 1996
9. RAID 6 P+Q ,<http://www.intel.com/technology/magazine/computing/RAID-6-0505.htm>

About Xyratex

Xyratex is a leading provider of enterprise class data storage subsystems and storage process technology. The company designs and manufactures enabling technology that provides OEM and disk drive manufacturer customers with data storage products to support high-performance storage and data communication networks. Xyratex has over 20 years of experience in research and development relating to disk drives, storage systems and high-speed communication protocols.

Founded in 1994 in a management buy-out from IBM, and with its headquarters in the UK, Xyratex has an established global base with R&D and operational facilities in Europe, the United States and South East Asia.



UK HQ

Langstone Road
Havant
Hampshire PO9 1SA
United Kingdom

T +44(0)23 9249 6000

F +44(0)23 9249 2284

www.xyratex.com



ISO 14001: 2004 Cert No. EMS91560

©2007 Xyratex (The trading name of Xyratex Technology Limited). Registered in England & Wales. Company no: 03134912. Registered Office: Langstone Road, Havant, Hampshire PO9 1SA, England. The information given in this brochure is for marketing purposes and is not intended to be a specification nor to provide the basis for a warranty. The products and their details are subject to change. For a detailed specification or if you need to meet a specific requirement please contact Xyratex.

