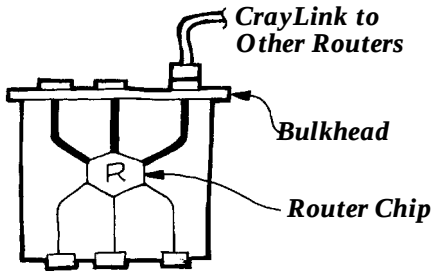
Hypercube Connectivity within ccNUMA Architecture

The Silicon Graphics Origin2000™ and Onyx2™ systems are structured to fit a customer's specific applications and problem sizes. This is accomplished with the company's ccNUMA (cache coherent non-uniform memory access) architecture, which can link multitudes of processors together in such a way that the number of interconnections scales with the growth of the system, avoiding the bandwidth limitations of bus-based architectures. The multidimensional nature of the interconnectivity that allows the architecture to accomplish unprecedented levels of system scalablity is the particular focus of this paper.

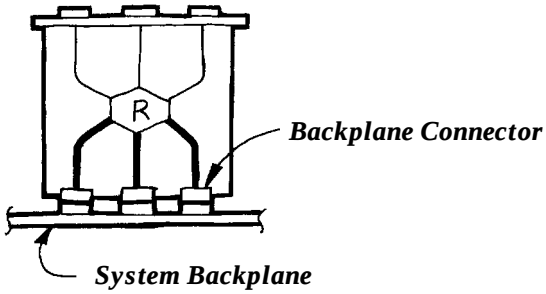
Basic Building Blocks

The ccNUMA architecture, for the purposes of this discussion, consists of two basic components: a node and a router. The node is a circuit board with two CPUs on it. The router is a crossbar chip that resides on the router board. It has a total of six connections for linking together node and router boards:

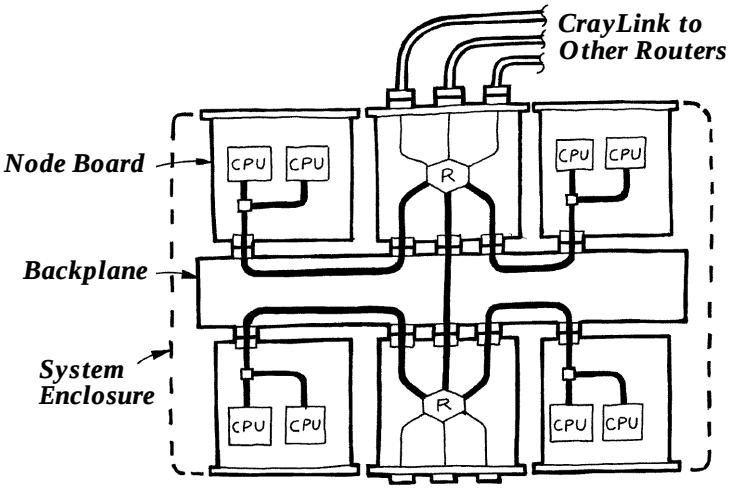
Three of the router connections are on the board's bulkhead and connect to other router boards via CrayLink™ cables.



The three remaining connections mate with the system backplane, allowing the router to connect to two node boards and a second router within the same enclosure.



Since each node board has two CPUs, a router can directly connect to four CPUs within the same enclosure. Additionally, the router to router connection across the backplane ultimately allows eight CPUs to be interconnected within a single enclosure. Further system expansion can be realized by connecting to routers in other enclosures via the CrayLink cables.



Principles of Hypercube Connectivity

The ccNUMA architecture links up multitudes of routers and nodes by emulating a multidimensional geometrical construct known as a hypercube. A hypercube possesses the following properties:

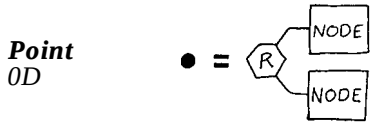
- A hypercube has n spacial dimensions, where n can be any positive integer (including zero)
- A hypercube has 2^n vertices
- There are n connections (lines) that meet at each vertex of a hypercube
- All connections at a hypercube vertex meet at right angles with respect to each other

Conceptually speaking, each router chip is equivalent to the vertex of a hypercube and each cable or backplane connection is equivalent to a line that connects two hypercube vertices. Thus, the hypercube is the underlying router interconnection architecture into which the nodes are connected. This may sound complicated, but hypercubes up to three dimensions are actually familiar structures, which will be demonstrated below.

0D Hypercube

A zero-dimensional hypercube is simply single point or vertex (2^0=1 vertices). This point would be a router, with up to two nodes attached (for clarity, the nodes will not be shown for higher-dimensional hypercubes).

Vertices =**1**
CPUs = 4 CPUs / vertex × 1 vertex = **4**
Total Vertex Connections = **0**

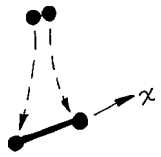


Point
0D

1D Hypercube

A one-dimensional hypercube is 2^1=2 vertices connected together, with n=1 connection at each point. This is simply a line. It can be thought of as a point that has been duplicated and stretched apart in the *x* axis.

Vertices = **2**
CPUs = 4 CPUs / vertex × 2 vertices = **8**
Total Vertex Connections = **1**

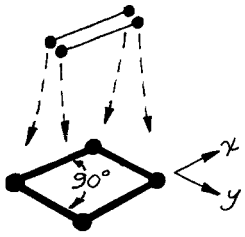


Line
1D

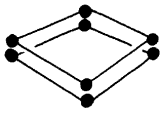
2D Hypercube

A two-dimensional hypercube has 2^2=4 vertices, with n=2 connections at each vertex. This is nothing more than a square. This can be thought of as a line that has been duplicated and stretched apart in the *y* axis.

Vertices = **4**
CPUs = **16**
Total Vertex Connections = **4**



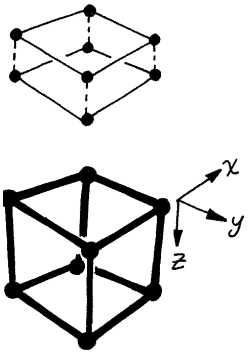
Square
2D



3D Hypercube

A three-dimensional hypercube has 2^3=8 vertices, with n=3 connections at each vertex. This is familiar to us as a cube. It can be constructed by duplicating a square and stretching it apart in the *z* axis. It can be seen that in all of the hypercubes the connections at each vertex meet at right angles to each other.

Vertices = **8**
CPUs = **32**
Total Vertex Connections = **12**



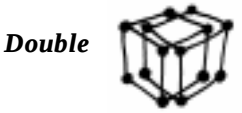
Cube
3D

Note that with the addition of each dimension, the number of vertices in the hypercube doubles. This concept suggests a double-and-stretch method can be used to understand the *connectivity* of hypercubes with greater than three dimensions, even though true visualization of 4+ spacial dimensions is beyond human ability.

Visualizing 4D Hypercube Connectivity

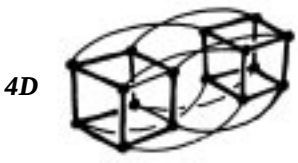
We begin with a 3D cube.

The cube is duplicated, or doubled, upon itself with connections between the twin vertices.



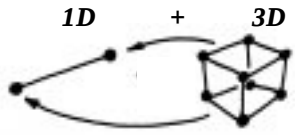
The twin cubes are pulled away from each other in the direction of the new fourth dimension, stretching the links between twin vertices.

The resulting structure is two 3D cubes with eight links connecting the common corners between the cubes. If we had a 4D piece of paper available to us, we could have drawn the 4D cube without any of the lines crossing each other. We cannot construct or imagine a true 4D structure in our 3D universe, but *we can build a system that maintains the same vertex-to-vertex connectivity*.

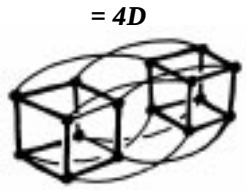


Another very useful technique for visualizing 4D+ hypercubes is available, which we will dub dimension-replacement:

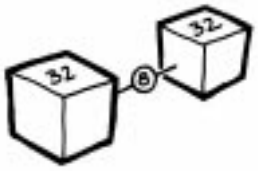
Imagine that a 4D structure is simply a 1D structure combined with a 3D structure. So, we can replace the 0D vertices at the two ends of a 1D line with 3D cubes.



The single line is then replaced with eight lines connecting common corners. It is as if each 0D point “expanded” to 3D and the 1D line split into eight connecting lines. This yields the same structure that resulted from the double-and-stretch method.



The 4D diagram can be cleaned up by drawing the structure as two 3D cubes connected by a line that represents the eight corner-to-corner connections.



Our 4D hypercube (constructed with either method) possesses the following attributes:

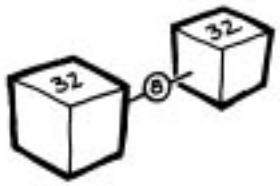
Vertices = **16**
CPUs = 4 × 16 vertices = **64**
Total Vertex Connections = **32**

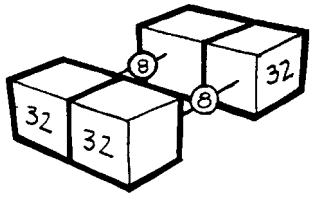
The double-and-stretch and dimension-replacement methods can be extended for all 4D+ hypercubes:

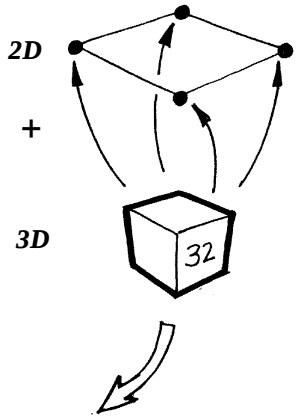
Double and Stretch **Dimension Replacement**

4D Hypercube

Vertices = **16**
CPUs = **64**
Total Vertex Connections =**32**

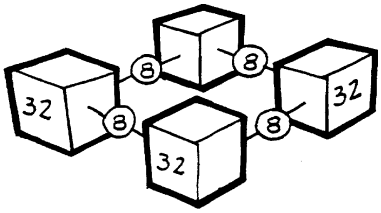


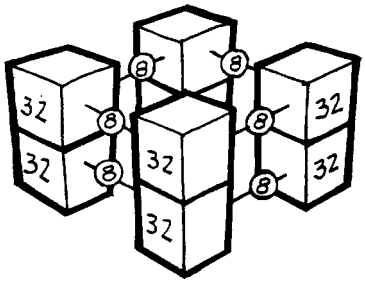


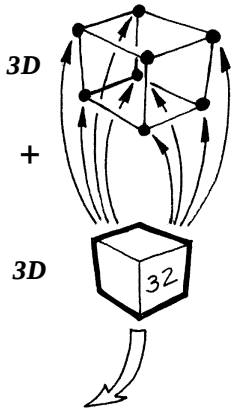


5D Hypercube

Vertices = **32**
CPUs = **128**
Total Vertex Connections = **80**

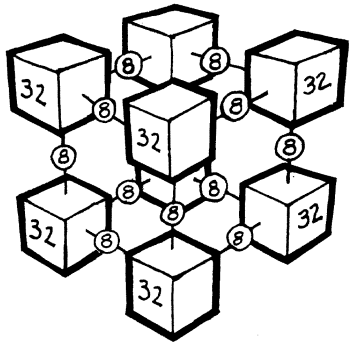






6D Hypercube

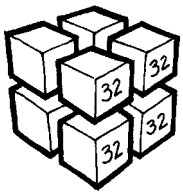
Vertices = **64**
CPUs = **256**
Total Vertex Connections = **192**

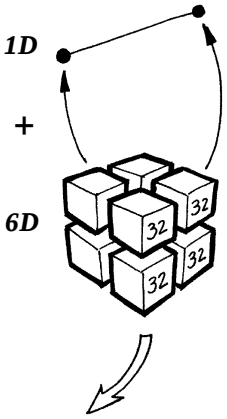


Double and Stretch **Dimension Replacement**

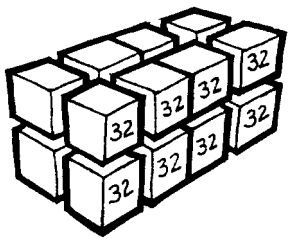
6D Hypercube

For clarity, the 6D cube is redrawn without connections between the 3D cubes.



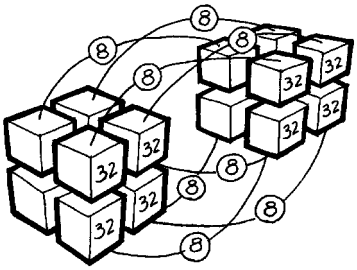


Note that at this point, the patterns again repeat themselves. However, instead of two 3D cubes connected together with eight lines to form a 4D cube, we will have two 6D cubes connected together with 64 lines to form a 7D cube.

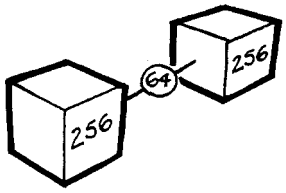


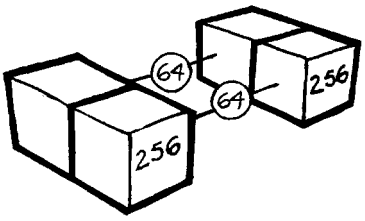
7D Hypercube

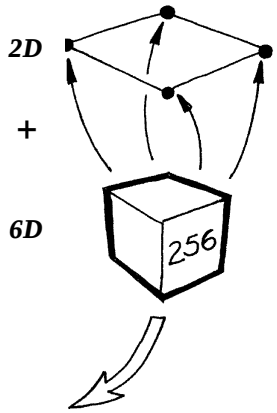
Vertices = **128**
CPUs = **512**
Total Vertex Connections = **448**



Simplified 7D Diagram

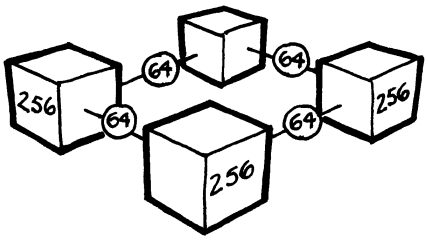






8D Hypercube

Vertices = **256**
CPUs = **1,024**
Vertex Connections = **1,024**



The 8D system, as will be seen, is the practical limit for a six-port router, but the double-and-stretch and dimension-replacement methods can be repeated ad infinitum to visualize multidimensional hypercubes.

Bandwidth Advantages of Hypercube-Based Systems

A common method for evaluating the effectiveness of a system's interconnectivity is to look at the system's bisectional bandwidth. The bisectional bandwidth is determined by dividing a system in half (bisecting) and counting the number of cut connections that normally would have allowed one half of the system to communicate with the other. As the number of CPUs in a system grows, it is desirable to have the bandwidth grow proportionally to prevent bottlenecks. The following examples will illustrate the differences in bisectional bandwidth between hypercube-based and bus-based systems of similar size.

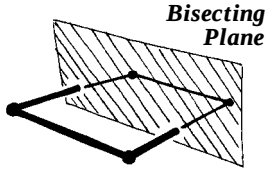
For the purposes of illustration, it will be assumed that within both the hypercube and bus systems, all system interconnection occurs between routers, to which four CPUs are attached.

Hypercube-Based Systems

16 CPUs

Bisectional Cuts = 2

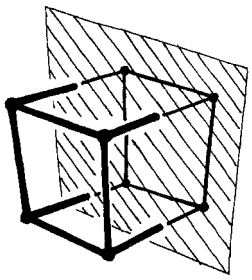
Bisectional Bandwidth
= 2 Cuts / 16 CPUs
= **1/8 Cuts per CPU**



32 CPUs

Bisectional Cuts = 4

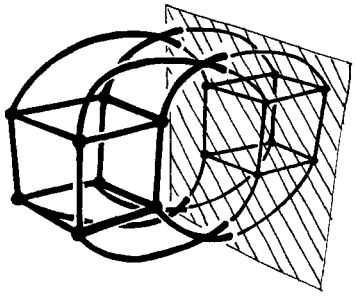
Bisectional Bandwidth
= 4 Cuts / 32 CPUs
= **1/8 Cuts per CPU**



64 CPUs

Bisectional Cuts = 8

Bisectional Bandwidth
= 8 Cuts / 64 CPUs
= **1/8 Cuts per CPU**

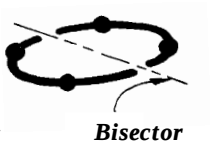


Bus-Based Systems

16 CPUs

Bisectional Cuts = 2

Bisectional Bandwidth
= 2 Cuts / 16 CPUs
= **1/8 Cuts per CPU**



32 CPUs

Bisectional Cuts = 2

Bisectional Bandwidth
= 2 Cuts / 32 CPUs
= **1/16 Cuts per CPU**

64 CPUs

Bisectional Cuts = 2

Bisectional Bandwidth
= 2 Cuts / 64 CPUs
= **1/32 Cuts per CPU**

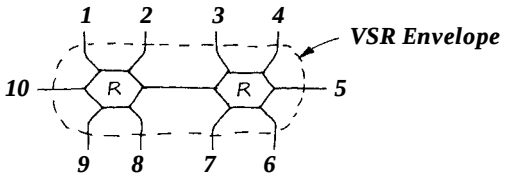
It can be seen that the hypercube system enjoys a constant 1/8 Cuts per CPU, regardless of size, but the bus system suffers from a halving of its connections per CPU every time the system doubles in size. This loss of proportional bandwidth in the bus system can cause severe bottlenecks, especially in large systems.

It should not be surprising that the bandwidth for the hypercube system remains constant, when one considers that an n+1 dimensional hypercube can be constructed by doubling an n-dimensional hypercube and connecting the common vertices. It should also be noted that the bisectional bandwidth of a hypercube is not affected by the direction of the cut. For example, the 32-CPU cube above could have been bisected horizontally, instead of vertically, and still yielded four cut connections. This is due to the fact that hypercubes are symmetrical in all dimensions.

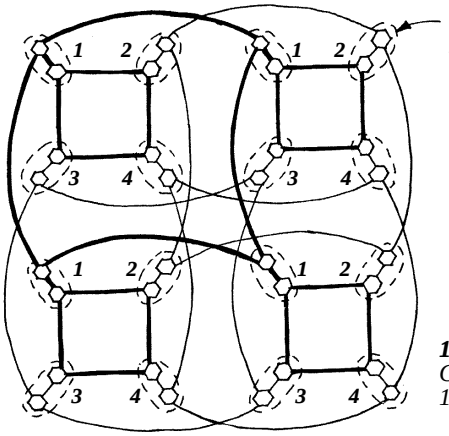
Metarouters and Large System Connectivity

As mentioned, each vertex of a hypercube represents a router with six connections. Two connections link to node cards, while the remaining four connect to other routers. In an n-dimensional system, each router will need n+2 connections: n connections to the other routers and 2 connections to the node cards. This is no problem up to a 4D system, where all six router connections are utilized. However, for an 8D system, a 10-connection super router is required. Such a router is impractical, but it is possible to construct a virtual super router, or VSR, by connecting two normal routers together.

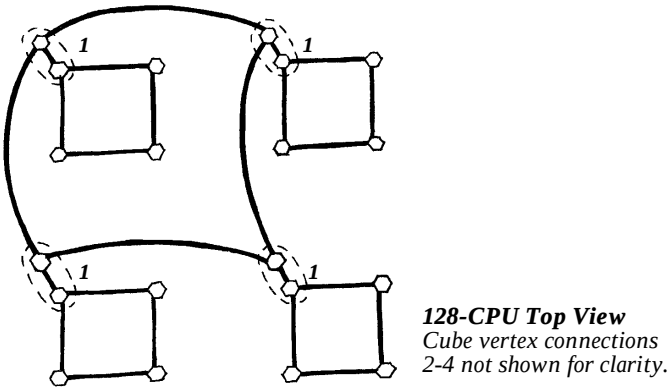
Each router uses one port to connect to the other router. This leaves five ports free on each router for a total of 10 ports leaving the envelope of the VSR.



It is now possible to construct a 5D+ system by replacing all single routers residing at the hypercube vertices with VSRs, as shown in the 128-CPU system at right.

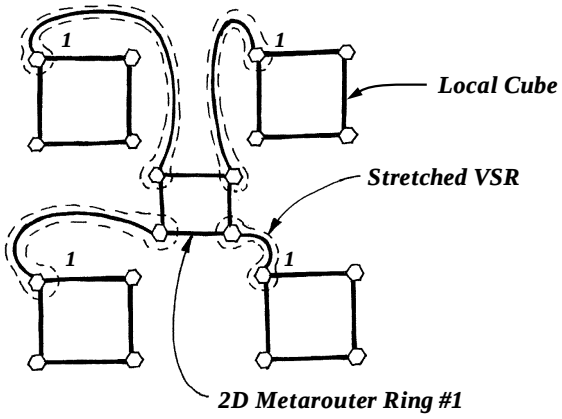


However, it is impractical to fit an entire VSR in a single enclosure. Remembering that a VSR is really two normal routers connected with a CrayLink cable, we can simply stretch apart a VSR to suit our needs. This can be demonstrated with the #1 cube-corner connections of our 128-CPU system.



The resulting structure, is four local 32-CPU cubes connected into a metarouter ring (local will refer to routers located within the 32-CPU cubes; the prefix meta- will refer to routers and associated structures that connect only to other routers).

Performing this operation with the other seven cube corners would yield a total of eight metarings. Routers in one metaring do not connect to routers in another.



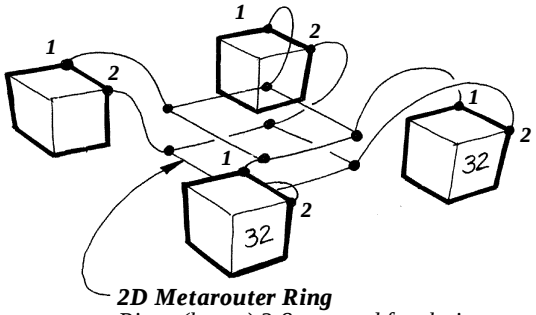
The additional ports provided by the metarouters allow continued system expansion for five dimensions and beyond. This will be demonstrated below, starting with the 5D-metaroutered system that was just constructed.

5D-Based System

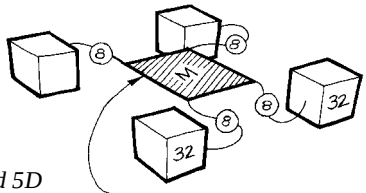
128 CPUs

Each of the local cubes' eight corners connects to a different metarouter layer: #1 corners connect to metalayer #1, #2 corners connect to metalayer #2, and so on. Regardless of system dimensionality, there will *always* be eight metalayers to connect to the eight corners of the local cubes. Also, the metastructure will always have (n - 3) dimensions. Hence, the 5D system at right contains a 5D - 3D = 2D metastructure.

In the simplified diagram, the eight non-connecting metalayers are represented by a single shaded structure.



2D Metarouter Ring
Rings (layers) 3-8 removed for clarity



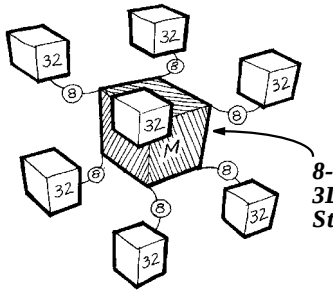
Simplified 5D Diagram

8-Layer 2D Metarouter Structure

6D-Based System

256 CPUs

The 6D system is two 5D systems connected through the metarouter rings in each 5D half of the system. This forms eight 3D (6D - 3D = 3D) metarouter cubes in the center of the system.

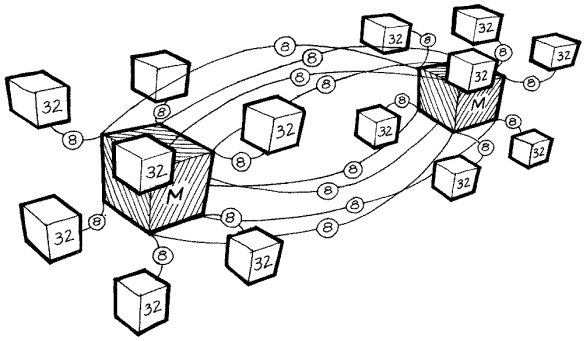


8-Layer 3D Metarouter Structure

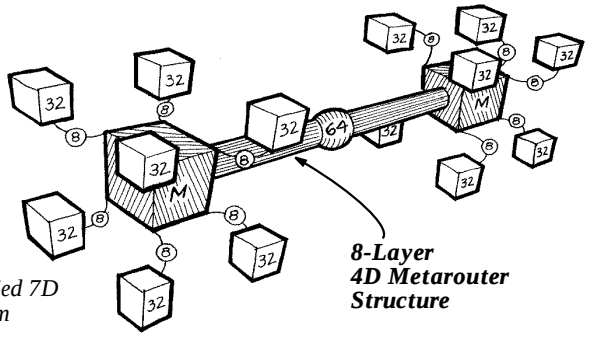
7D-Based System

512 CPUs

The 7D system is two 6D systems connected through the 3D metacubes in the center of each 6D half.



In the simplified diagram, it can clearly be seen that the 7D system can also be thought of as a 4D (7D - 3D = 4D) metastructure, into which 16 local cubes are connected.

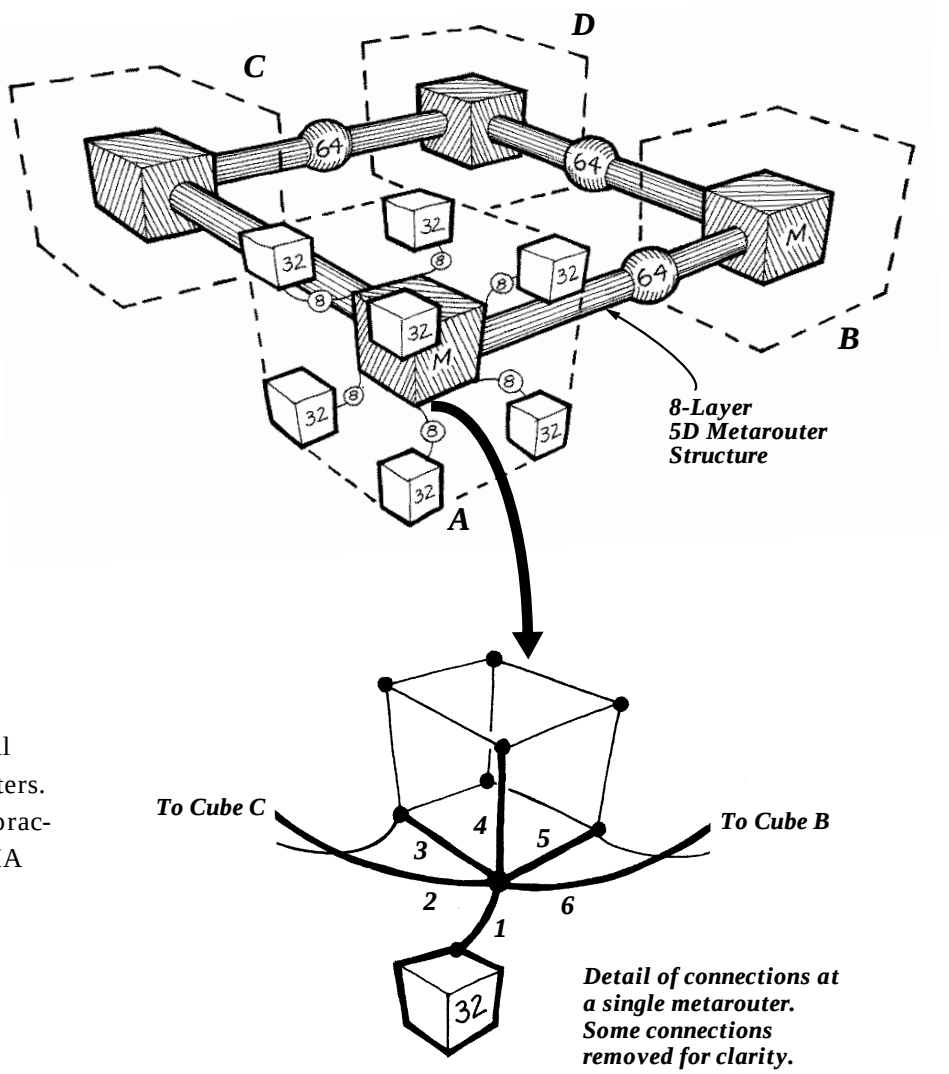


Simplified 7D Diagram

8-Layer 4D Metarouter Structure

8D-Based System
1024 CPUs

Finally, the 8D system is two 7D systems hooked together via the two 4D metacube structures. It can also be thought of as a 5D metastructure, into which 32 local cubes are connected.



The 8D system utilizes all six ports on the metarouters. Thus, 1,024 CPUs is the practical limit for the ccNUMA architecture.

Bisectional Bandwidth of Metaroutered Systems

The use of metarouter structures to achieve 5D+ sized systems may lead one to conclude that the system bisectional bandwidth is somehow altered from the 1/8 connections per CPU that was demonstrated for nonmetaroutered systems. In fact, this bandwidth per CPU is not altered. For example, if one were to vertically bisect the 8D system, above, into two 7D systems, two 64-connection metabundles would be cut, yielding a total of 128 cut connections.

The resulting bandwidth per CPU is 128 connections / 1,024 CPUs = **1/8 connections per CPU.**

Summary of ccNUMA Hypercube Attributes

Dims	CPUs	Routers (vertices)			Router-to-Router Connections		
		Local	Meta	Total	Backplane	CrayLink	Total
0	4	1	-	1	0	0	0
1	8	2	-	2	1	0	0
2	16	4	-	4	2	2	4
3	32	8	-	8	4	8	12
4	64	16	-	16	8	24	32
5	128	32	32	64	16	96	112
6	256	64	64	128	32	204	236
7	512	128	128	256	64	472	536
8	1,024	256	256	512	128	1,072	1,200

Acknowledgments

The description of the dimension-replacement technique was adapted from the following article:

Tamiko Theil, “The Design of the Connection Machine,” *Design Issues,* Volume 10, Number 1, Spring 1994.

Special thanks to Dave Lima, Dan Farmer, Richard Singer and Sally Abolitz for their helpful comments while reviewing this paper.

Comments

Please submit comments to:

Jim Ammon
Silicon Graphics
2011 N. Shoreline Blvd., MS 565
Mountain View, CA 94043

(650) 933-3512 (phone)
(650) 961-9075 (fax)

jammon@engr.sgi.com