Automated Design of Torus Networks

Konstantin S. Solnushkin
konstantin@solnushkin.org

ABSTRACT
This paper presents an algorithm to automatically design networks with torus topologies, such as ones widely used in large-scale supercomputers. The characteristic feature of our approach is that real life equipment prices and values of technical characteristics are used. As a result, we also have the opportunity to compare costs of torus and fat-tree networks.

The algorithm is useful as a part of a bigger design procedure that selects optimal hardware of cluster supercomputer as a whole.

Categories and Subject Descriptors
C.2.1 [Computer-Communication Networks]: Network Architecture and Design—Network topology; K.6.2 [Management of Computing and Information Systems]: Installation Management—Computer selection

General Terms
Design, Economics

Keywords
Torus network

1. INTRODUCTION
Torus networks are frequently used in large-scale supercomputers as a cost-efficient alternative to other topologies. Recently, it was demonstrated that torus networks for computer clusters can be built from affordable commodity hardware such as InfiniBand.

We describe an algorithm that allows to automatically design torus networks. The algorithm is implemented in a software tool [8].

This algorithm is intended to be used as a part of a CAD system for cluster supercomputers [9]. Such a system would iterate through different combinations of hardware, varying the number of compute nodes and other parameters. Thus, designing an interconnection network for every hardware combination under review is a self-contained and highly repetitive operation that must be performed efficiently.

During the design process, other characteristics of interconnection networks, such as reliability, can be estimated and used as design constraints or as a part of a complex objective function.

The rest of the article is organized as follows. Section 2 describes relevant work in the field of torus networks. Section 3 details the design of the “Gordon” supercomputer. Section 4 introduces the algorithm, and section 5 compares costs of torus networks with fat-trees. Finally, section 6 concludes the article.

2. RELATED WORK
Torus networks have found widespread use in supercomputing. IBM used a 3D torus network in BlueGene/L, and a 5D network [3] in BlueGene/Q. A 6D mesh-torus network was used in “K Computer” [1]. Both are direct networks, where compute nodes are connected directly to their neighbours, as opposed to switched fabrics, where nodes are first connected to switches, and then switches are connected to each other in a torus topology. The example of the latter is a 3D torus network of the Gordon supercomputer [11].

Torus networks are inherently prone to congestion, but this is mitigated by designers by increasing the number of dimensions. Commenting on the Gordon project, Strande [10] quotes the following benefits of torus networks: (a) lower cost compared to fat-trees and (b) easy linear scaling along one of dimensions. However, such scaling may result in unbalanced topologies, leading to bigger latencies and higher congestion on the links in that dimension. Strande also mentions that the torus topology uses short cables, which makes the use of fibre optical cables unnecessary, leading to further cost savings.

Navaridas and Miguel-Alonso [6] analysed performance of 2D switch-based torus topologies and fat-trees for up to 7680 compute nodes, on a range of workloads, using simulation techniques. They conclude that performance degradation from using torus networks, compared to fat-trees, can reach 20-40%, and sometimes more, on communication-intensive workloads, which limits applicability of tori in larger installations.

Camara et al. [2] introduced the technique to turn unbalanced rectangular 2D and 3D tori to twisted tori by rearranging peripheral links, which improves performance characteristics as well as regains network symmetry.

3. 3D DUAL-RAIL TORUS NETWORK OF THE GORDON SUPERCOMPUTER
Gordon supercomputer [11] uses InfiniBand switches with $P = 36$ ports of 4X QDR technology. Switches form a 4x4x4
torus; each switch has 6 neighbours, to which it connects with 3 links, thereby utilizing 18 ports out of 36. 17 more ports are used to connect 16 compute nodes and one I/O node.

The network is dual-rail, therefore there are actually two tori made of switches, and compute and I/O nodes have two network interfaces, one of which is used to connect to the switch in the first torus (“rail”), and the other to the second one. Currently, one rail is used for MPI, and the other one for I/O traffic. According to Strande [10], there are plans to use both rails simultaneously to provide failover capabilities and improve bandwidth.

4. ALGORITHM FOR DESIGNING TORUS NETWORKS

We propose the algorithm to calculate the number of switches in a torus network, using as input the number of compute nodes to be interconnected and, optionally, a blocking factor that determines the distribution of ports on a switch between compute nodes and neighbouring switches. The algorithm is suitable to design networks built with commodity hardware, such as Gordon’s network.

As torus networks are inherently prone to congestion, imposing additional blocking at the switch level is very disadvantageous. However, sometimes it is stipulated by the hardware manufacturer, and cannot be avoided. For example, in [6] the hardware under review was a blade chassis equipped with \( N = 20 \) compute nodes and an InfiniBand switch with \( P = 36 \) ports. Only 16 ports of the switch were used to connect it to the outside world, which resulted in \( Bl = 20/16 = 1.25 \) blocking factor. In order to build torus networks for such hardware with the proposed algorithm, we need to specify the blocking factor as an input.

The algorithm tries to build a network using identical switches with \( P_E \) ports. Let us describe the algorithm by stages. In line 1 we check if the switch has enough ports to connect all \( N \) nodes. In this case, we use the star topology with only one switch and exit.

Otherwise, we will build a ring or a torus. In lines 8..10 we calculate the number of switch ports that go to compute nodes and to the neighbouring switches, and then recalculate the blocking factor for the network. On line 11 we derive the minimal number of switches required to connect \( N \) nodes with a given blocking factor. The actual torus will contain slightly more switches (generally, the increase is within 20% for small networks, and within several percent for the large ones).

On line 12, we use a heuristic to determine the number of torus dimensions, based on the number of switches. It is important to note that there are no hard rules when choosing the number of dimensions. Choosing a low number of dimensions for a high number of compute nodes leads to increased network diameter and therefore latencies. On the other side, choosing a too high number of dimensions for a low number of compute nodes does not provide network performance benefits but results in complex cabling patterns. In the case of direct networks this scenario also requires network adapters with an unnecessarily large number of ports.

The optimal number of dimensions depends on the communication pattern of the application, and can be reliably determined, for any given application, only through benchmarking on real hardware or by using simulation such as in [6]. Therefore we relied on using a heuristic.

Currently, the dimension choice heuristic returns the number of dimensions as per Table 1, up to \( D = 5 \). The layout of switches in the maximal configuration for that number of dimensions is provided in the last column of the table for reference.

If the heuristic returns \( D = 1 \), then we use the ring topology (line 14). Otherwise, we use the torus topology, and need to calculate the number of switches along each of \( D \) dimensions by rounding \( \sqrt{E} \) to the nearest integer (line 17).

Algorithm 1 Design a torus network

| Input: |
| --- |
| \( N \): Number of nodes to interconnect |
| \( Bl \): Blocking factor |
| \( P_E \): Number of switch ports |

| Goal: |
| Optimal network structure: |
| \( D \): Number of torus dimensions |
| \( d = (d_1, \ldots, d_D) \): Number of switches along each dimension |
| \( E \): Total number of switches |
| \( Bl_i \): Resulting blocking factor |
| \( L \): Number of cables |
| \( f \): Objective function for the optimal network structure |

| 1: if \( P_E \geq N \) then |
| 2: \{ If there exists a switch with \( N \) or more ports \} |
| 3: \textbf{print} Topology: star |
| 4: \( E \leftarrow 1 \); \( Bl_r \leftarrow 1 \); \( L \leftarrow N \) |
| 5: Compute \( f \) |
| 6: Exit |
| 7: \textbf{end if} |
| 8: \( P_{En} \leftarrow \lfloor P_E \cdot (Bl/(1 + Bl)) \rfloor \) \{ Ports to nodes \} |
| 9: \( P_{Ec} \leftarrow P_E - P_{En} \) \{ Ports to other switches \} |
| 10: \( Bl_r \leftarrow P_{En}/P_{Ec} \) \{ Resulting blocking \} |
| 11: \( E \leftarrow \lceil N/P_{En} \rceil \) \{ Minimal number of switches \} |
| 12: \( D \leftarrow \text{GetDimCount}(E) \) \{ Heuristic for the number of torus dimensions \} |
| 13: if \( D = 1 \) then |
| 14: \textbf{print} Topology: ring |
| 15: else |
| 16: \textbf{print} Topology: torus |
| 17: \( d_i \leftarrow \text{round}(\sqrt{E}) \) \{ Number of switches along dimensions \} |
| 18: \( d_D \leftarrow \lfloor E/d_D^{D-1} \rfloor \) \{ Switches in the last dimension \} |
| 19: \( E \leftarrow \prod_{i=1}^{D} d_i \) \{ Actual number of switches \} |
| 20: \textbf{end if} |
| 21: \( L \leftarrow N + E \cdot P_{Ec}/2 \) \{ Number of cables \} |
| 22: Compute \( f \) |

This creates a topology close to an ideal square, cube, etc. Packaging constraints, however, may preclude from using this particular ideal layout, and in the resulting unbalanced torus the number of switches along dimensions may differ significantly. The number of switches, \( E \), still remains the same as returned by the algorithm, allowing to correctly calculate equipment cost and other metrics.
The network is expandable from \( N \) cable. This is then multiplied by the number of switches divided by two, because two ports are connected with one node. Inter-switch links run in bundles of approximately \( P \) of switch ports facing to neighbouring switches, \( P \) is determined on line 21. The number of cables is determined on line 21. The number of switch counts along all dimensions \( D \), therefore it is often possible to use cables that integrate several links (such as a 12x InfiniBand cable that integrates three 4x links) to reduce the number of physical cables, simplifying installation.

Sample output of the algorithm for commodity InfiniBand switches with \( P_E = 36 \) ports and a non-blocking network \( (Bi = 1) \) is presented in Table 2.

### Table 1: Heuristic for the number of torus dimensions

| Dimensions, \( D \) | Topology | Max. configuration |
|----------------------|----------|--------------------|
| up to 36             | Ring     | 2 \( \times 6 \times 6 \times 6 \times 5 \) |
| up to 125            | Torus    | 3 \( \times 5 \times 5 \times 5 \) |
| up to 2401           | Torus    | 4 \( \times 7 \times 7 \times 7 \times 7 \) |
| more than 2401       | Torus    | 5 \( \text{(As appropriate)} \) |

### Table 2: Sample output for Algorithm 1

| Compute nodes, \( N \) | Dimensions, \( D \) | Torus topology | Supercomputer of comparable size |
|------------------------|---------------------|----------------|----------------------------------|
| 1,000                  | 3                   | 4x4x4          | Gordon [11]                      |
| 6,000                  | 4                   | 4x4x4x6        | Stampede [12]                    |
| 8,000                  | 4                   | 5x5x5x4        | Tianhe-1A [5]                    |
| 10,000                 | 4                   | 5x5x5x5        | SuperMUC [4]                     |
| 19,000                 | 4                   | 6x6x6x6        | Titan [7]                        |

On the next step, we calculate the number of switches in the last dimension (line 18) and recalculate the total number of switches as the product of switch counts along all dimensions (line 19).

The number of cables is determined on line 21. The number of switch ports facing to neighbouring switches, \( P_{Ec} \), is divided by two, because two ports are connected with one cable. This is then multiplied by the number of switches \( E \). Compute nodes are connected with additional \( N \) cables. The network is expandable from \( N \) up to \( E \cdot P_E \) compute nodes. Inter-switch links run in bundles of approximately \( P_{Ec}/(2 \cdot D) \), therefore it is often possible to use cables that integrate several links (such as a 12x InfiniBand cable that integrates three 4x links) to reduce the number of physical cables, simplifying installation.

Sample output of the algorithm for commodity InfiniBand switches with \( P_E = 36 \) ports and a non-blocking network \( (Bi = 1) \) is presented in Table 2.

## 5. COST COMPARISON OF TORUS AND FAT-TREE NETWORKS

We used real life equipment costs provided by Mellanox Technologies to derive costs of fat-tree and torus networks for up to 3,888 compute nodes. We utilized the tool for automated design of cluster interconnection networks [8]. Equipment costs are given for the older generation of equipment (InfiniBand QDR), and technical characteristics are summarized in Table 3. Cable cost is assumed to be $80.

We consider three models of switches. The first of them, the 36-port switch, is used for building torus networks, and is also utilized on edge level of fat-tree networks. The other two are modular switches that have 108 and 216 ports in their maximal configurations. The actual number of supported ports depends on the number of installed line cards, which leads to 6 and 12 configurations of these switches, respectively. Each configuration has its own set of technical characteristics as well as cost.

The set of equipment described above allows to build non-blocking fat-tree networks with up to \( N_{max} = P_E \cdot P_C/2 \) nodes. On Fig. 1 we plot costs of non-blocking as well as 2:1 blocking fat-tree networks, and torus networks. As expected, the cost of 2:1 blocking fat-trees is lower than of their non-blocking counterparts; but reduction in cost is less than twofold. Torus networks are consistently cheaper than fat-trees; however, their inherent blocking may have detrimental effect on application performance that will not be offset by lower costs.

We also consider an alternative way of building fat-trees: using 36-port switches for both core and edge layers. This allows to build non-blocking fat-tree networks with up to \( N_{max} = 36 \cdot 36/2 = 648 \) nodes. Such networks are characterized by complex wiring patterns between the two layers, but are marginally cheaper to build. Fig. 2 is essentially a close-up of the previous figure, focusing on values of \( N \) up to 648 nodes, with an additional curve representing costs of the alternative fat-tree building method.

As the diagram indicates, using 36-port switches for building fat-trees does indeed lead to certain cost savings: for \( N = 648 \) nodes, per-port cost of such networks is roughly $1,060, while for the usual way of building fat-trees, using modular switches on the core level, the per-port cost is roughly $1,930. However, these savings should be weighted against the cost of compute nodes: if the latter is much higher than the per-port cost of the interconnection network, then cost savings might not justify increased wiring and maintenance complexity of this type of networks.

### Example

Let us assume the cost of a compute node is $5,000. If per-port cost of two types of interconnection networks is $1,000 and $2,000, respectively, then savings from using the network of the first type is 7000/6000, or roughly 17%. Factoring in costs of other equipment, as well as operating expenses, further dilutes savings.

Figure 2 is particularly helpful to emphasize the structure of networks generated by the network design tool [8]. Consider, for example, the case of non-blocking and 2:1 blocking fat-trees, for \( N = 150 \) compute nodes. The costs of these
Figure 1: Cost comparison of fat-tree and torus networks

Figure 2: Cost comparison of alternative fat-tree building methods
| Switch applicability          | Switch model                  | Port count | Size, U | Weight, kg | Power, W | Cost, $  |
|------------------------------|-------------------------------|------------|---------|------------|----------|----------|
| Torus, Fat-tree (edge layer) | Mellanox Grid Director 4036   | 36         | 1       | 7.7        | 202      | 10,820   |
|                              |                               | 18         | 1       | 75.1       | 516      | 78,500   |
|                              |                               | 36         | 1       | 77.8       | 606      | 90,000   |
|                              |                               | 54         | 7       | 80.6       | 696      | 101,500  |
| Fat-tree (core layer)        | Mellanox IS5100               | 72         |         | 83.3       | 786      | 113,000  |
|                              |                               | 90         |         | 86.1       | 876      | 124,500  |
|                              |                               | 108        |         | 88.9       | 966      | 136,000  |
|                              | Mellanox IS5200               | 18         |         | 115.7      | 516      | 125,500  |
|                              |                               | 36         |         | 118.4      | 606      | 137,000  |
|                              |                               | 54         |         | 121.2      | 696      | 148,500  |
|                              |                               | 72         |         | 123.9      | 786      | 160,000  |
|                              |                               | 90         |         | 126.7      | 876      | 171,500  |
|                              |                               | 108        |         | 129.5      | 966      | 183,000  |
|                              |                               | 126        | 10      | 132.2      | 1,056    | 194,500  |
|                              |                               | 144        |         | 135.0      | 1,146    | 206,000  |
|                              |                               | 162        |         | 137.7      | 1,236    | 217,500  |
|                              |                               | 180        |         | 140.5      | 1,326    | 229,000  |
|                              |                               | 198        |         | 143.3      | 1,416    | 240,500  |
|                              |                               | 216        |         | 146.0      | 1,506    | 252,000  |

Table 3: Characteristics of InfiniBand QDR equipment

| Network type          | Non-blocking | 2:1 blocking |
|-----------------------|--------------|--------------|
| Topology              | Star         | Two-level fat-tree |
| Edge level switch     | Mellanox IS5200 (162 ports) | Mellanox Grid Director 4036 (36 ports) |
| Core level switch     | N/A          | Mellanox IS5100 (90 ports) |
| Power, W              | 1,236        | 2,290        |
| Weight, kg            | 137.7        | 140.0        |
| Size, U               | 10           | 14           |
| Cost, $               | 229,500      | 218,960      |

Table 4: Structure comparison for two types of fat-tree networks, for \( N = 150 \) nodes.
two networks are very close, but their structure is entirely different, which is summarized in Table 4.

If the tool is requested to design a non-blocking network, it chooses a star topology with a single modular switch. If, however, a 2:1 blocking network is requested, the result is a two-layer fat-tree, with 36-port switches on the edge level and a 90-port switch on the core level. The latter network is chosen because it is marginally (5%) cheaper. At the same time, it draws 85% more power and requires 40% more space in the rack.

This example illustrates two points: (A) more complex criterion functions, such as total cost of ownership, should preferably be used instead of capital costs; (B) trying to design blocking networks doesn’t necessarily save considerable amounts of money, therefore designers should consider non-blocking networks first.

6. CONCLUSIONS

We presented a simple algorithm for automated design of torus networks. The algorithm relies on a heuristic to choose the number of torus dimensions. We also compared real life costs of torus and fat-tree networks. We found that torus networks are consistently cheaper than non-blocking and 2:1 blocking fat-trees; however, these cost savings may not offset performance penalties, depending on applications used.

7. REFERENCES

[1] Y. Ajima, T. Inoue, S. Hiramoto, and T. Shimizu. Tofu: Interconnect for the K computer. *Fujitsu Sci. Tech. J.*, 48(3):280–285, 2012. http://www.fujitsu.com/downloads/MAG/vol48-3/paper05.pdf.

[2] J. Cámara, M. Moretó, E. Vallejo, R. Beivide, J. Miguel-Alonso, C. Martínez, and J. Navaridas. Mixed-radix twisted torus interconnection networks. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–10. IEEE, 2007.

[3] D. Chen, N. Eisley, P. Heidelberger, R. Senger, Y. Sugawara, S. Kumar, V. Salapura, D. Satterfield, B. Steinmacher-Burow, and J. Parker. The IBM Blue Gene/Q interconnection network and message unit. In *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, pages 1–10. IEEE, 2011. http://mmc.geofisica.unam.mx/edp/SC11/src/pdf/papers/tp19.pdf.

[4] Leibniz Rechenzentrum. SuperMUC. http://www.lrz.de/services/compute/supermuc/.

[5] National Supercomputing Center in Tianjin (NSCC). Tianhe-1A. http://www.nscc-tj.gov.cn/en/.

[6] J. Navaridas and J. Miguel-Alonso. Indirect cube: A power-efficient topology for compute clusters. *Optical Switching and Networking*, 8(3):162–170, 2011.

[7] Oak Ridge National Laboratory (ORNL). Titan. http://www.olcf.ornl.gov/titan/.

[8] K. S. Solnushkin. Fat-tree and torus network design tool at ClusterDesign.org. http://clusterdesign.org/torus/.

[9] K. S. Solnushkin. Computer cluster design automation using web services. In *Proceedings of International Supercomputing Conference, ISC’12*, June 2012.

[10] S. Strande. Gordon – design and performance of a 3D torus interconnect for data intensive computing. In *Proceedings of HPC Advisory Council Held in Conjunction with the International Supercomputing Conference*, 2012. www.hpcadvisorycouncil.com/events/2012/European-Workshop/Presentations/4_SDSC.pdf.

[11] S. Strande, P. Cicotti, R. Sinkovits, W. Young, R. Wagner, M. Tatineni, E. Hocks, A. Snavely, and M. Norman. Gordon: design, performance, and experiences deploying and supporting a data intensive supercomputer. In *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the extreme to the campus and beyond*, page 3. ACM, 2012.

[12] Texas Advanced Computing Center (TACC). Stampede. http://www.tacc.utexas.edu/stampede.