Workload Management in Multicore Processing

Will Wu
Electrical and Computer Engineering
University of Rochester
500 Computer Studies Building
Rochester, NY 14627
ywu37@ur.rochester.edu

I. INTRODUCTION

Queues are ubiquitous in our consumerist way of life. Every economic transaction is explained by analyzing supply and demand curves. An athlete needs a new pair of shoes. A battery powered device needs a new set of batteries to function. A mother orders from a takeout restaurant for the entire family. In all three of the previous scenarios, a customer demands a service and enters the queueing system to be serviced or have its demand satisfied by any one of the servers provided by the service and then exits the queueing system. A queueing line most prominently appears when the rate of customer arrivals demanding a service and the rate of customers being serviced is not identical.

In the realm of computer architecture, the proliferation of multicore processing systems in an attempt to execute multiple different programs on multiple different datasets has been a compelling reason to tie in the study of graph theory. With the introduction of more unsynchronized cores, communication among those cores is necessary to ensure unnecessary excess work is performed, bloating execution time. The use of queues are necessary to hold metadata that allows for cross-core communication and synchronization. For the purposes of this paper, the term ”core” is not restricted to a computation core soldered onto a microprocessor. It is used to generally talk about a device that is accomplishing work in the realm of program execution. This ranges from as small as an arithmetic logic unit running a reduced instruction set architecture to a graphics processing unit working a large image dataset to as large as a large server room handling database queries or TCP connections and everything in between.

II. THE NATURE OF MULTICORE PROCESSING

Before diving into the study of how graph and queue theory can help in the design of multicore processors, it is helpful to define some terms from computer architecture. Latency is defined as the time differential between a customer entering a system and that customer leaving the system. This is also referred to as the sojourn time in the realm of graph theory. Throughput defined as the amount of work done over some period of time. This paper will focus on the three components of interconnect design are topology, routing, and flow control. The term topology relates to the geometric structure of the network: the relationship between nodes and wires connecting those nodes. This is the most visual portion of design and relates closely with graph theory. The term routing relates to the algorithms that dictate the protocol of how packets traverse through the network. Flow control has to do with the resources of the network and how they are used to allow packets of datagrams to traverse through the network. Flow control is much less associated with the topology of a network whereas routing is immensely affected by the network's topology. Routing is a programmable algorithm of getting a packet from a source node to a destination node. Flow control consist of the hardware components that is part of the interconnect. Topology dictates exactly which nodes the flow control connects.

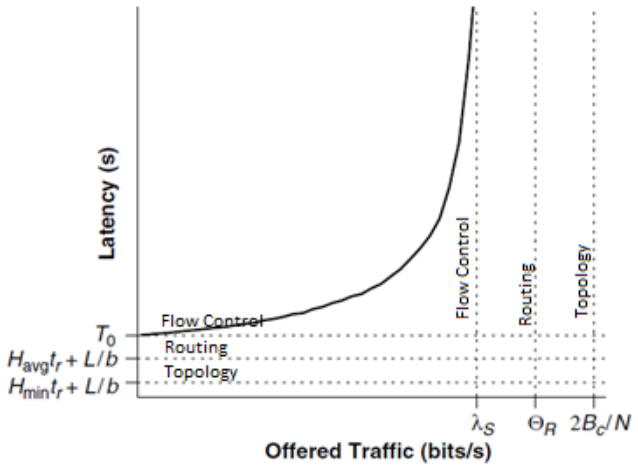

Fig. 1. The cumulative effects of Topology, Routing, and Flow Control on the latency of packet movement.

The interconnection network is the name given for the system of programmable devices that handles communications between many different terminals. For the most part, these terminals correspond with a processing core on the chip while some terminals are dedicated to the memory system or the input/output ports. Such an organization scheme is shown in the figure below. It is important to note that the interconnect does not necessarily have to connect every port to every other port. Low latency is more difficult as the

number of nodes scale up, the amount of space the wiring between devices grows more slowly than the expected traffic through those wires [19].

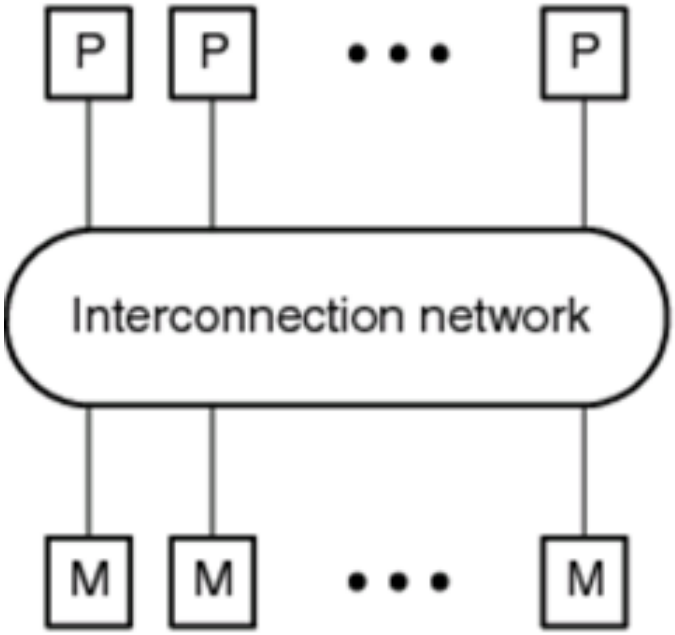

Fig. 2. The interconnection network in relation to processing cores ports and memory ports [1]

The bisection bandwidth is one of the most important characteristics in the design of interconnect. It is defined as the minimum bandwidth of all bisections of a network and approximately 50% of the traffic through a network traverses through that bisection [1]. The bisection bandwidth is outlined in the figure below. With the current design paradigm, the bisection area grows at a scale of $N^{2/3}$ while the traffic grows by a scale of N nodes [19].

There are many different ways to organize processing cores among themselves. The simplest interconnect topology between especially small number of cores is a bus topology where every port to every other port . A ring topology can be thought of a bus topology with its two ends tied together. In a ring interconnect system, every port is no longer required to be connected to every other port in the system. Neighboring nodes communicate with each other. A mesh topology is a grid layout of nodes with immediate neighbor communication capabilities and is not restricted to two dimensions. In a two-dimension mesh network, the entire network can be modeled with a Cartesian plane. In a three-dimensional mesh network, the entire network can be modeled with a cube. A torus topology is a mesh topology with its ends wrapping on itself. A common everyday model for the torus network is a donut. Torus networks are quite prevalent in present-day designs. It has been found that torus networks always gives the lowest latency due to the highest bandwidth channels and most direct physical route between two nodes. While higher-dimensional figures is a good method of understanding the structure of the topology in question, one discrepancy between the model and the development of the network topology is the realization that everything must be laid out on silicon which is typically in a two-dimensional plane. To illustrate this point, the connections between ports of neighboring nodes are shown in the figure below.

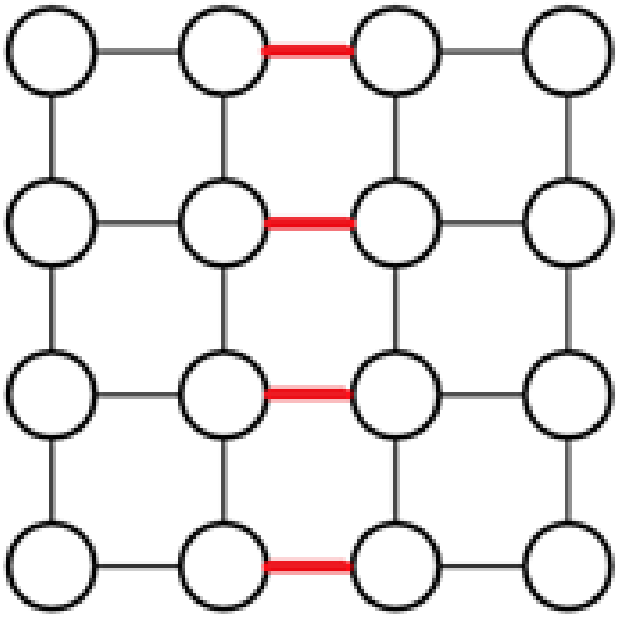

Fig. 3. The bisection bandwidth of this 4-ary 2-cube network is outlined in red.

There is a naming convention for interconnects by indicating a networks dimensionality, topology, and radix. A network of N nodes can be organized into a k^n structure where k is the radix and n is the dimension. This information is then compressed in two terms called k-ary n-naming convention. The term after the 'n' describes the topology of the network. For example: a 4-ary 2-cube network is a mesh network of radix 4 in 2 dimensions which is shown in the figure above. When wire cost is considered, low dimensional networks such as torus networks offer lower latency than high dimensional networks such as binary n-cubes.

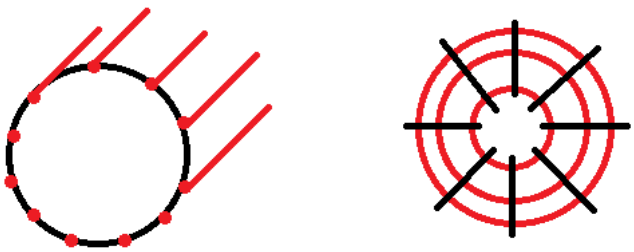

Fig. 4. A ring of rings is a good way of visualizing a donut or torus network.

One fundamental aspect of program execution is that the number of requests present mid-execution at any one time

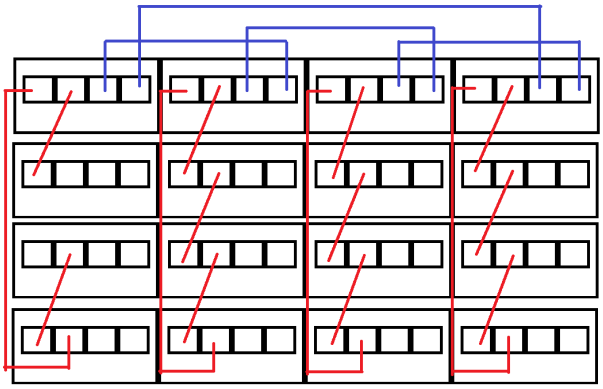

Fig. 5. Wiring a torus network (not all blue wires are shown as to not clutter the figure)

cannot be negative. Such a behavior can also be called a birth-death process. This is where the study of queueing theory is invaluable.

III. QUEUEING THEORY

Now that a connection has been made between customers in a queue and threads and requests from processes, we can break down the anatomy of a queueing system. A queue has three main attributes: the customer arrival rate, the customer departure rate or service rate, and the number of servers. All other attributes can be derived from the first three attributes. In the realm of computer processing, customers are thread and process requests or the working dataset.

The most common arrival distribution follows that of a Poisson distribution whereas the most common service distribution exponential distribution [2]. The rate of service for a queue behaves in an exponentially distributed manner with no regard to the service of other customers. Given the nonidentical behaviors of the job arrival rate and job service rate, it is quite necessary for queues to be implemented for each node.

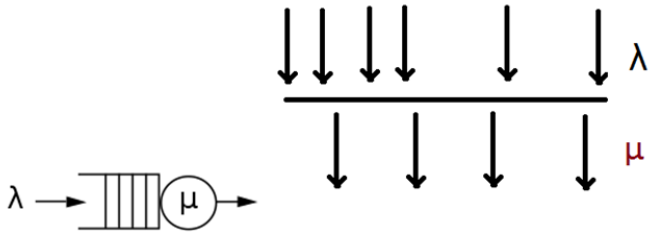

Fig. 6. A visualization of the queue model. lambda represents the arrival rate and mu represents the service rate.

In the realm of queueing theory, Kendall's notation is a standardized method of describing different queues. The M/M/1 queue structure is a good introduction to queueing theory and some time will be dedicated to talking about the other characteristics of the queue structure in the context of the M/M/1 queue. The 'M' stands for the Markovian process

Kendalls Notation: A/S/c/K/N/D
A = time between arrivals
S = size of jobs
c = number of servers
K = capacity of queue
N = size of population of jobs to be served
D = queueing discipline

TABLE I
KENDALL'S NOTATION

that arrivals behave in. As such, the rate of arrivals can be modeled by the following probability density function:

$$f(t) = \lambda e^{-\lambda t} \tag{1}$$

The probability that the number of customers in the queue is less than X at any point after time t can derived from the following:

$$P(t, t = \Delta t | t > t_0) = \frac{\lambda e^{-\lambda t}}{\int_{t_0}^{\infty} \lambda e^{-\lambda t} dt} = \frac{\lambda e^{-\lambda t}}{e^{-\lambda t_0}} = \lambda e^{-\lambda (t - t_0)} \tag{2}$$

When the final three parameters are not specified, K = ∞ N = ∞ and D = first in first out (FIFO) [1] by convention. The Kendall notation is very useful for modeling the queue but for realistic workload scenarios, K is bounded by the size of memory reserved and N is bounded by the number of requests. The queueing discipline may not be a FIFO scheme for stack or recursive based workloads or priority based workloads.

Server utilization is only moderately important for the purposes of scaling up the workload. Low server utilization may suggest imbalanced load assignment. The average service time is one aspect that contributes to aggregate execution time. The service time has been rigorously minimized over the years. The more important time metric is the customer wait time. This is because changes made to the interconnect vastly affects the skew between the customer arrival distribution and the customer service distribution.

$$\rho = \lambda T_s \tag{3}$$

The steady state average number of customers in the queue is the product of the steady state arrival rate and the average time a customer spends in queue and in execution. This simple relationship known as the Little's Law is independent of the service distribution or the method that requests enter the queueing system.

$$L = \lambda T_q = \frac{\rho}{(1 - \rho)} \tag{4}$$

$$P(n >= N) = \rho^n \tag{5}$$

[1] first in first out (FIFO) can also be referred to as first come first serve (FCFS)

IV. ADJUSTMENTS TO THE TORUS

Network pruning involves removing some of the links such that the network is not fully connected in accordance with the donut model. Directed One idea is to permit single directional routing. Due to both the directed nature of the network, the network diameter is no longer akin to that of a ring topology. Due to the routing restriction, the team also had to compute the network diameter which is the maximum distance between any combination of two nodes in the network. Pruning also influences latency in the aspect of hop count, or the number of intermediate nodes that a packet must traverse between source and destination node.

Parhami et al. clearly outlined their torus network Tera multithreaded architecture with the following adjacency matrix Ω, pruning matrix Ψ. Sets a and b consists of all nodes in the network and g is some element in the adjacency matrix and Ψ is derived by b = a + ΨΠΦg. Directed matrix were construed using the adjacency matrix Ω and inversing all negative values of the orientation matrix. [17]

$$\Omega = \left\{ \begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix}, \begin{bmatrix}k-1\\0\\0\\\vdots\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\\vdots\\0\end{bmatrix}, \begin{bmatrix}0\\k-1\\0\\\vdots\\0\end{bmatrix}, \ldots, \begin{bmatrix}0\\0\\0\\\vdots\\1\end{bmatrix}, \begin{bmatrix}0\\0\\0\\\vdots\\k-1\end{bmatrix} \right\}$$

$$\Psi = \begin{bmatrix}1 & 0 & \ldots & 0 & 0\\0 & 0 & \ldots & 0 & 1\\0 & 1 & \ldots & 0 & 0\\\vdots & \vdots & \ddots & \vdots & \vdots\\0 & 0 & \ldots & 1 & 0\end{bmatrix}.$$

$$\prod_{i=0}^{2} \Phi_i^{a_i} = \begin{bmatrix}1 & 0 & 0\\0 & -1 & 0\\0 & 0 & -1\end{bmatrix}^{a_0} \begin{bmatrix}-1 & 0 & 0\\0 & 1 & 0\\0 & 0 & -1\end{bmatrix}^{a_1} \begin{bmatrix}-1 & 0 & 0\\0 & -1 & 0\\0 & 0 & 1\end{bmatrix}^{a_2}$$

$$= \begin{bmatrix}(-1)^{a_1+a_2} & 0 & 0\\0 & (-1)^{a_0+a_2} & 0\\0 & 0 & (-1)^{a_0+a_1}\end{bmatrix}.$$

It was found that the pruning mechanism helped lower message latency with undirected connections. The directionality of the connections provide some improvement in the unpruned networks but pruning those networks causes the routing to be too restrictive and worsens the latency under heavy loads.

nD k-Torus Variant	In/Out Degree d	Diameter D	Average Internode Distance Δ	Fault Diameter D_F	Bisection Width B
Unpruned, Undirected	6	$1.5k$	$0.75k$	$D+1$	$2k^2$
Pruned, undirected	4	$1.5k$	$0.75k + 2/k - 2/k^2$	$\leq D + \lceil k/2 \rceil - n + 2$	k^2
Unpruned, directed	3	$1.5k+1$	$0.75k + 1 - 4/k^3$	n/a	k^2
Pruned, directed	2	$1.5k+3$	$\cong 0.75k + 3.5 - 4/k$	n/a	$0.5k^2$

Fig. 7. The performance models of 3D torus networks [17]

V. VARIOUS CASE STUDIES

A. The Butterfly Topology

A butterfly topology is a simpler version of a mesh topology. Nodes feed into (n-1) routers with a fan-in of k

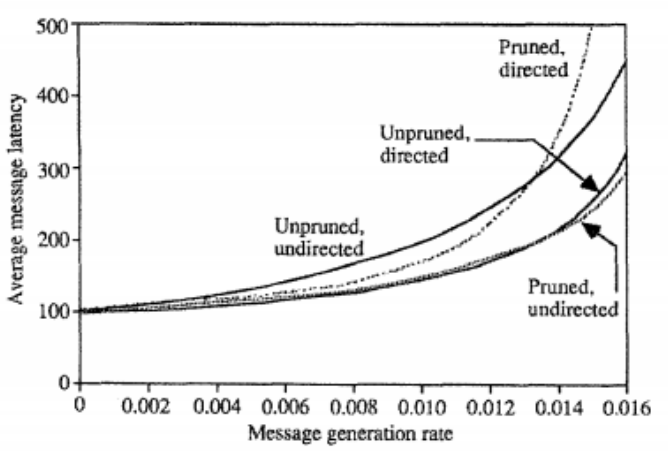

Fig. 8. Message latency with radix 16, message length 384 bytes [17]

ports and a fan-out of k ports. The butterfly network is most commonly used for Internet routing as it is the most scalable with n stages of routing devices for k^n end user nodes. The routing protocol is very simple as it involves computing the hamming distance between source and destination nodes. The distance dictates the number of switching made during the packet's journey through the crossbar switch in each router. The latency is has the least variance with every end node being equidistant from every other end node. As such the hop count is incredibly low.

B. Task Assignment Schemes

As a reminder, the processor sharing discipline is a queue discipline that is part of the Kendall's queue notation (under the letter representation 'D') explained earlier in the paper. By default, a queue's discipline is FIFO. Dynamic Task Allocation such as Processor Sharing is modeled under a M/G/1 queue (where G stands for geometric distribution).

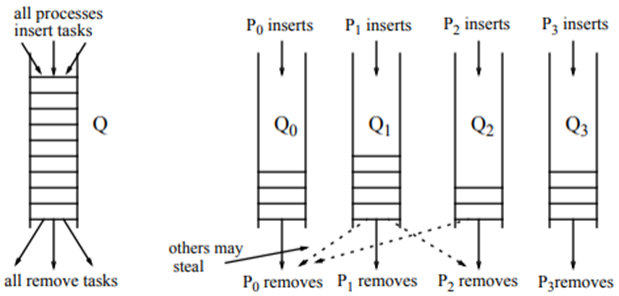

Fig. 9. Dynamic Task Allocation through Task Snooping [3]

A method different from the task snooping explained above is increasing the redundancy of the work done. At first glance this may seem wasteful but this method is used for big data computing using remote servers, where powerful hardware is not the limiting factor and timing is [16]. Tasks are assigned to multiple different cores with the core who completes the task first invalidating all other cores also completing the same task.

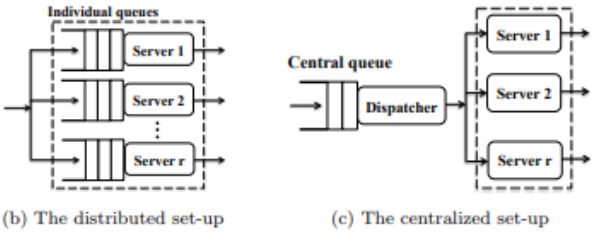

Fig. 10. Distributed vs Centralized Redundant Task Allocation [3]

C. SGI

The SGI Origin server machine utilizes a cube and hypercube topology. When the network is not fully saturated, it has the capability to utilize the hypercube's idle resources to effectively decrease the network diameter. This at best cuts down the time spent in queue by half. This sort of detection is CrayLink interconnect found in each node. The SGI also uses a dynamic router and flow control mechanism called the wavefront allocator [4]. This allocator assigns all tasks in waves across the interconnect mesh. A set of nodes are assigned together if they are equidistant from a local source node. The task arbitration is done independently from previous or future stages. This allocation technique allows for the latency to scale logarithmically with respect the number of nodes or linearly with the network diameter.

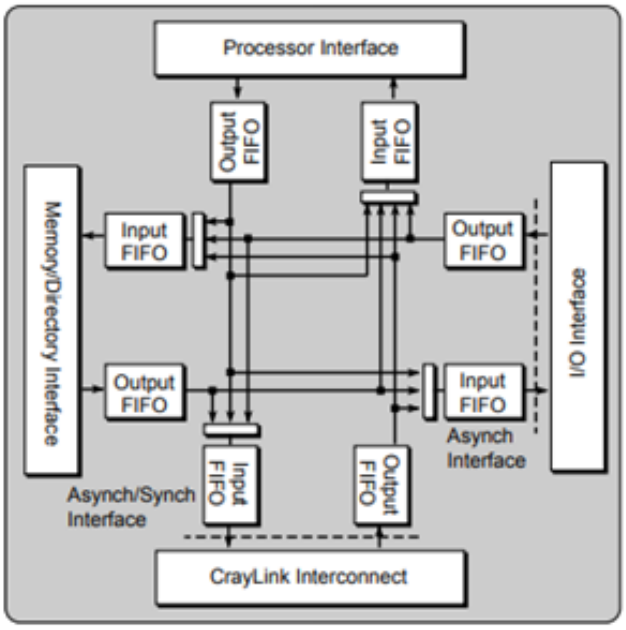

Fig. 11. The CrayLink detection module in each node.

VI. FINAL REMARKS

Throughout the research procedure in this deep pool of knowledge, development in the areas of interconnect can be broadly split into two categories. The pruned torus networks from Parham et al. provide a design where specific connections between nodes are intentionally missing. The directionality of torus networks restricts the movement of packets from bidirectional to unidirectional. One the other end of the spectrum, some designs such as that of the SGI provide redundancy through increased circuitry. Qui et al. also disseminated increased redundancy to make up for server failure rates. So it can be generalized that network development has a conservative approach (where routing is restricted and more formulaic) and a liberal approach (where multiple different path possibilities are viable).

It bears repeating but engineering has always been about cost/benefit analysis. There can never be a design where the network interconnect is both reduced from that of the model and more redundant than the model. The industry can never have its cake and eat it too. So there must be a prescriptive approach to problem solving; different problems require different types of networks solutions. A more conservative approach is to be used on a specialized workload or where the routing algorithm is less reliable. This comes at the expense of latency and overall time spent executing for execution correctness. A more liberal approach can work where timing is more demanding and any inaccuracies in execution are tolerated or can be corrected before program termination.

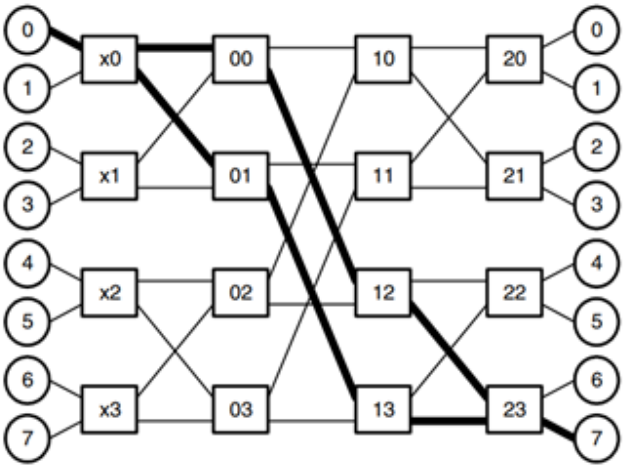

Fig. 12. Adding an extra stage to this butterfly network allows for increased path diversity. [1]

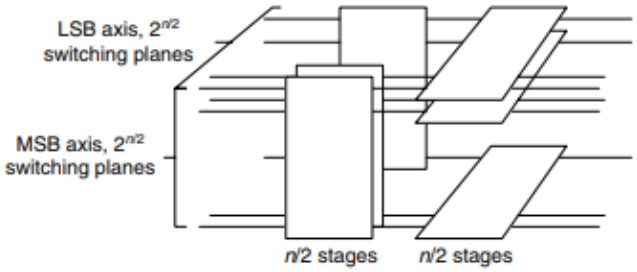

Fig. 13. The Construction of a Butterfly Network using Perpendicular Planes [1].

REFERENCES

[1] Principles and Practices of Interconnection Networks, By Dally and Towles, Morgan Kaufmann 2004.

[2] Parallel Computer Organization and Design, Dubois, Annavaram, and Stenstrom. Cambridge University Press 2012.

[3] "Parallel computer architecture a hardware/software approach" D. Culler, J. Singh, and A. Gupta. , 1999.

[4] James Laudon and Daniel Lenoski. The SGI origin: A ccNUMA highly scalable server. In Proceedings of the 24th Annual International Symposium on Computer Architecture (ISCA-97), volume 25,2 of Computer Architecture News, pages 241251, New York, June2 4 1997. ACM Press.

[5] Daniel F. Spulber; Christopher S. Yoo, On the Regulation of Networks as Complex Systems: A Graph Theory Approach, 99 Nw. U. L. Rev. 1687 (2005)

[6] C. Guo, H. Wu, K. Tan, L. Shiy, Y. Zhang, and S. Lu. Bcube: A high performance, server-centric network architecture for modular data centers. In SIGCOMM, 2009.

[7] J. Arjona Aroca and A. Fernandez Anta. Bisection (Band)Width of Product Networks with Application to Data Centers. IEEE TPDS, 25(3):570580, March 2014.

[8] A. Thomasian. Analysis of fork/join and related queueing systems. ACM Computing Surveys, 47(2):17:117:71, 2014.

[9] Olvera-Cravioto, Mariana, and Octavio Ruiz-Lacedelli. "Parallel queues with synchronization." arXiv preprint arXiv:1501.00186 (2014).

[10] J. Dean and S. Ghemawat, MapReduce: simplified data processing on large clusters, Comm. of the ACM, vol. 51, pp. 107113, Jan. 2008

[11] J. F. Prez, R. Birke and L. Y. Chen, "On the latency-accuracy tradeoff in approximate MapReduce jobs," IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, Atlanta, GA, 2017, pp. 1-9. doi: 10.1109/INFOCOM.2017.8057038

[12] G. Joshi, E. Soljanin, and G. Wornell, Queues with redundancy: Latency-cost analysis, ACM SIGMETRICS Workshop on Mathematical Modeling and Analysis, jun 2015.

[13] GUE K RKIM H H. Predicting departure times in multi-stage queueing systems[J]. Computers & Operations Research201239(7)1734-1744.

[14] Kawanishi K, Takine T (2016) MAP/M/c and M/PH/c queues with constant impatience times. Queueing Syst 82:381420

[15] S. Asmussen and J. R. Mller. Calculation of the steady state waiting time distribution in GI/PH/c and MAP/PH/c queues. Queueing Syst., 37:929, 2001.

[16] Qiu, Zhan, Juan F. Prez, and Peter G. Harrison. "Tackling Latency via Replication in Distributed Systems." Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering. ACM, 2016.

[17] Behrooz Parhami and Ding-Ming Kwai, Comparing Four Classes of Torus-Based Parallel Architectures: Network :Parameters and Communication Performance, Mathematical and Computer Modeling, vol. 40, no. 7-8, pp. 701720, 2004.

[18] Egorova R, Mandjes MRH, Zwart AP (2007) Sojourn time asymptotics in processor sharing queues with varying service rate. Queueing Syst 56: 169181

[19] William J. Dally. Performance analysis of k-ary n-cube interconnection networks. IEEE Transactions on Computers, 39(6):775785, June 1991.

[20] Liebeherr, J, Akyildiz, IF (1995) Deadlock properties of queueing networks with finite capacities and multiple routing chains. Queueing Syst 20(34): 409431