JST: Smart Systems and Devices
Volume 34, Issue 2, May 2024, 018-026

Extending Amdahl's and Gustafson Baris's Laws by Adding Communication Overheads

Pham Van Hai[1], Ho Khanh Lam[2], Dung Nguyen Hoang[3*]
[1] Ha Noi Open University, Ha Noi, Vietnam
[2] Hung Yen University of Technology and Education, Hung Yen, Vietnam
[3] Ha Noi University of Science and Technology, Ha Noi, Vietnam
[*]Corresponding author email: dung.nguyenhoang@hust.edu.vn

Abstract

The extension of Amdahl's law and Gustafson-Barsis' law presented in this article provides insight into the relationship between communication costs and network topology in parallel computing. By considering communication latency in parallel computation execution time, these extensions can help researchers and developers optimize the interconnected network architecture of processing nodes and improve the performance of parallel computing systems. Today, parallel computer systems consisting of hundreds and thousands of processing nodes based on multiprocessor chip technology, high-speed optical transmission such as supercomputers are being researched, developed, and applied in many fields many areas. Although chip technology has progressed to the 3 nm process, the network architecture connecting processing nodes continues to be a problem that greatly affects the communication delay in the parallel computation time of the applications. In this paper, extensions of Amdahl's law and Gustafson-Barsis' law are presented with the addition of communication costs depending on the topology of the topology. These extensions provide insight into the relationship between communication costs and network topology in parallel computing. By considering communication latency in parallel computation execution time, these extensions can help researchers and developers optimize the interconnected network architecture of processing nodes and improve the performance of parallel computing systems.

Keywords: extensions of Amdahl’Law, Gustafson-Barsis, interconnection network topology, supercomputer, communication overhead, speedup.

1. Introduction

Today's supercomputers are deployed to big data, metaverse centers, and cloud computing. They applied in many fields, such as weather forecasting, aerodynamic research, climate change research, earthquake simulation, building radiological models, and probability analysis, simulating nuclear explosion in 3D, quantum biology, molecular and cellular biology, as well as protein folding, simulating the human brain, investigate and create models of physical phenomena, research and simulate AI (artificial intelligence), dark matter research, big bang reconstruction, and astronomical Research. Technology corporations such as IBM, HP, Cray, Dell, NEC, Intel, Lenovo, Fujitsu, Acer, and Oracle have been producing the most powerful supercomputers in the world, reaching speeds of several tens to thousands of petaflops (or some exaflops). Especially there are most efficient supercomputers in the Green500 list in terms of energy efficiency, for example, measured at 65.091 GFOPS/watt of Henri (Lenovo, Flatiron Institute, US, 2022), at 62.684 GFLOPS/watt of Frontier TDS (HP, US, 2022) [1].

Fig. 1 is shown Interconnection network of multicore computer nodes in modern supercomputers. Interconnection networks in cloud computing, multiprocessor systems, and supercomputers contain thousands of multiprocessor computer nodes, installed in hundreds of racks and attached to the clusters. The multiprocessor computes node contains several multicore/multithreaded CPUs and GPUs, memory modules, local storage disks, and network adapters. As the number of connected computer nodes increases, interconnection networks become larger and more critical than computing or memory modules, since communication delays (or communication overhead) between computer nodes can be greater if the choice of interconnection network configurations is inefficient. So, the communication overhead negatively affects the performance of parallel execution time of application problems. For many years, different transmission technologies have been put to practical use. InfiniBand is a channel-based fabric that has high-speed, very high throughput, and very low latency networking communications between interconnected nodes used in high-performance computing [2]. InfiniBand was developed from 1999 to 2025 and has many interconnection network topology options in HPC and

ISSN 2734-9373
https://doi.org/10.51316/jst.174.ssad.2024.34.2.3
Received: Jul 4, 2023; revised: Mar 11, 2024;
accepted: Mar 12, 2024

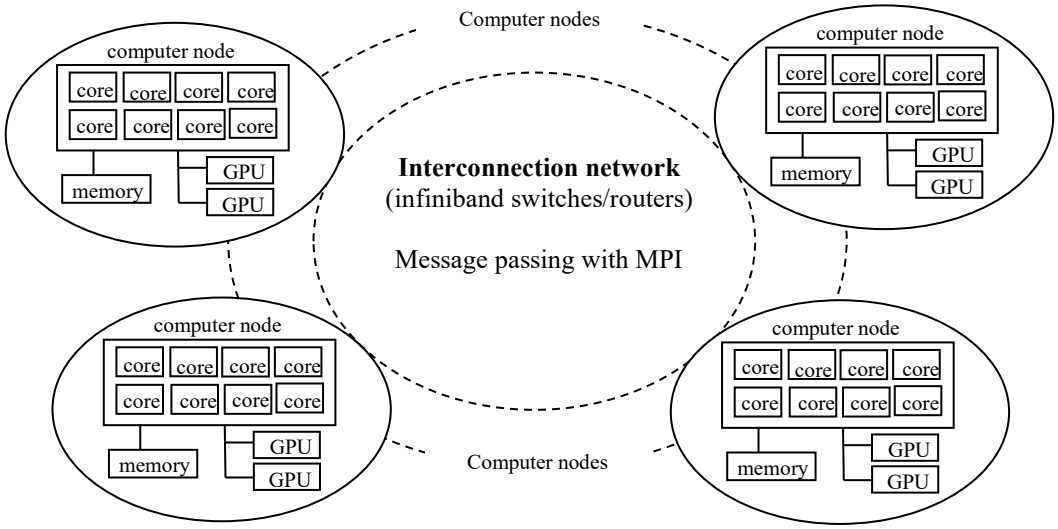

Fig. 1. Interconnection network of multicore computer nodes in modern supercomputers

AI infrastructures with several versions: SDR, DDR, QDR, FDR, EDR, HDR, NDR, and XDR. As of 2019, the Mellanox InfiniBand and Ethernet technologies are used in 59% of TOP500 supercomputer systems. InfiniBand accelerates the top 3 supercomputers in the world – #1 (ORNL - USA), #2 (LLNL - USA), and #3 (Wuxi - China), and 6 of the top 10 – #5 (TACC - USA), #8 (AIST - Japan), and #10 (LLNL - USA) [3].

From the original Amdahl's law year, there have been some research extending this law for multi-core systems, where the network delay linking processor cores is getting smaller and smaller as chip technology evolves. Mark D. Hill and Michael R. Marty Mark introduced formulas that extend Amdahl's law speedup for chips: SMP, AMC, and DMC. The integration in a multi-core chip with many different heterogeneous functional units leads to deciding how to distribute the available limited resources on a chip, such as an area and power, among all the computational units. That's why, T. Zidenberg, I. Keslassy, and U. Weiser [4] introduce MultiAmdahl, which considers the workload, the performance of each computational unit, and the total available resource. The results obtained by MultiAmdahl allow us to provide a closed-form solution for an optimal asymmetric-offload chip and to analyze the impact of different design constraints on an optimal chip architecture. Similarly, the research of L. Yavits, A. Morad, U. Weiser, and R. Ginosar has extended MultiAmdahl for resource allocation in heterogeneous architectures. They concluded that reduction in constant system power should be met by reallocating resources from general-purpose computing to heterogeneous accelerator-dominated computing, to keep the overall energy consumption at a minimum. We extend this conclusion to offer an intuition regarding energy-optimal resource allocation in data center computing. Amdahl's and Gustafson's law have been applied to multicore chips but inter-core communication has not been taken into account and they low restrict supercomputer applications and building ever bigger supercomputers [5]. Tian Huang and others find optimized parameters according to the speedup model using evaluating interconnections on multi-core processors. They also present a case study to show the necessity of incorporating interconnection into Amdahl's and Gustafson's law. Uri Verner, Avi Mendelson, and Assaf Schuster [6] proposed to extend Amdahl's law for multicore processors with built-in dynamic frequency scaling mechanisms such as Intel's Turbo Boost mitigated the speedup limitations obtained under Amdahl's law by providing higher performance for the same energy budget. Songwen Pei, Myoung-Seo Kim, and Jean-Luc Gaudiot [7] have proposed Amdahl's law extension by considering the overhead due to data preparation (ODP) in multicore system types: symmetric multicore, asymmetric multicore, dynamic multicore, heterogeneous CPU-GPU multicore, and dynamic CPU-GPU multicore. It represents that potential innovations in heterogeneous system architectures are necessary to reduce ODP. Chaitanya Poolla, and Rahul Saxena [8] have proposed to extend Amdahl's law to equip multiple configurable resources into the overall speedup equation and to transform the speedup equation into a multivariable regression problem suitable for the machine learning. To prove the correctness of the proposed Amdahl's law extension, they use experimental data from fifty-eight tests spanning two benchmarks (SPECCPU 2017 and PCMark 10) and four hardware platforms (Intel Xeon 8180M, AMD EPYC 7702P, Intel CoffeeLake 8700K, and AMD Ryzen 3900X) and analytical models are developed and cross-validated. Ami Marowka investigated how energy efficiency and scalability are affected by the power constraints imposed on contemporary hybrid CPU-GPU processors. Analytical models were developed to extend Amdahl's Law by accounting for energy limitations before examining the three processing modes available to heterogeneous

processors, i.e., symmetric, asymmetric, and simultaneous asymmetric [9]. Avi Mendelson and Assaf Schuster extend Amdahl's law for multicore processors with built-in dynamic frequency scaling mechanisms such as Intel's Turbo Boost. Using a model that captures performance dependencies between cores, we present tighter upper bounds for the speedup and reduction in energy consumption of a parallel program over a sequential one on a given multicore processor and validate them on Haswell and Sandy Bridge Intel CPUs [9]. It can be said that most of the research focuses on extending Amdahl's law to multi-core processor technologies.

Our paper focuses on an overview of the performance parameters of interconnection network topologies commonly used in supercomputers in recent years and thereby extending Amdahl's law and Gustafson-Barsis law by adding the communication overhead, which depends on the topologies.

2. Communication Overhead and Extending Speedup Laws

2.1. Characteristic Values of the Internetwork Network Topologies

Cost and performance values of the interconnection network topologies are as follows:

- Cost values:

+ Number of Host nodes (computer nodes) supported

+ Total Switches

+ Number of Links (Wires)

- Performance values:

+ Bisection Width: the minimum number of links cut to divide the network into two halves

+ Channel Width: the number of wires connecting two nodes.

+ Channel Rate: the number of bits/sec/wire. E.g. for a switch with 40 ports of rates/ports: 200Gb/s HDR, Mellanox Quantum offers an amazing 16Tb/s of bidirectional throughput and 15.6 billion messages per second in only TSW=130ns of port-to-port switch latency. So, the Channel Rate for 1 link is 200Gb/s.

+ Channel bandwidth: the product of (Channel Width) (Channel Rate).

+ Bisection Bandwidth: the sum of the bandwidths of the minimal number of links that are cut when splitting the system into two parts. It's like bisection width. Bisection bandwidth is frequently used to describe how well a network performs.

+ Diameter: max number of hops between two nodes

Several high-performance interconnection structures have been widely used in supercomputers, shown in Table 1, such as Crossbar (IBM Power5, Sun Niagara I/II), Ring (Intel Haswell, Larrabee, IBM Cell), 2D Mesh (Intel XP/S 140 Paragon), 3D Torus (IBM BlueGene/L, BlueGeneP, Cray XT3, Cray Jaguar, Cray Titan), 5D Torus (IBM Sequoia Blue Gene/Q), Tofu: 6D Mesh/Torus (Fujitsu K Computer and Fugaku) [10], Fat Tree (IBM Roadrunner, NUDT Tianhe-2, NUDT Tianhe-1A, TMC CM-5, Cray X2, Summit, Sierra, CS2, Altix systems), Crossbar (NEC The Earth Simulator), 3D Crossbar (Hitachi SR2201), Omega (IBM ASCI White), Slingshot (Frontier-HPE Cray EX235a, US), Dragonfly+ (Niagara, Canada).

Fat Tree *k*-ary *n*-layer (*k*-port switch and *n*-level tree) is one of the most widely used topologies. Table 2 shows the performance and cost values of this network topology. Fat trees are suitable for different applications with low latency and maximum data throughput for a variety of traffic patterns; however, it is relatively costly on a large scale due to the large number of switches and links it requires.

2.2. Extending Amdahl's Law Speedup

Amdahl's Law states that in parallelization, if f is the fraction (the parallelizable part within the total problem code), that can be improved by making parallel, and the rest 1-f is the fraction, that remains serial (non-parallelizable part), then the maximum speedup $S(f, n)$ that can be theoretically achieved using n processors in a multiprocessor system is:

$$S(f,n) = \frac{1}{(1-f) + \frac{f}{n}} \quad (1)$$

As n grows the speedup tends to 1/(1-f)

$S(f,n)$: denotes Amdahl's speedup model is limited by the total time needed for the sequential (serial) part of the total problem code.

Suppose *p:* the size of the total problem code, including the parallelizable part f and non-parallelizable part (1-f) (which cannot be executed in parallel).

Tseq(p): Execution time of the sequence part (non-parallelizable part) (1-f) within the total problem code.

Tpar(p): Execution time of the parallelizable part f of the total problem code in a multiprocessor system.

$T(p,1) = T_{seq}(p) + T_{par}(p)$; *T(p,1)* is total execution time of the total problem code on the sequential computer (with one processor).

$T(p,n) = T_{seq}(p) + \frac{T_{par}(p)}{n}$; *T(p, n)* is total execution time of the total problem code on the multiprocessor system (with n processors). This is explained in Fig. 2.

Table 1. Performance and cost values of several network topologies

Topology	Number of Nodes	Number of Links	Diameter	Bisection Width
Full connected	N	$N(N-1)/2$	1	$N^2/4$
Binary Tree	$N=2^d-1$	$N-1$	$2(d-1)=2\log\frac{(N+1)}{2}$	1
(k-ary n-tree) Fat Tree: k-port and n-level	$2\left(\frac{k}{2}\right)^n$	nN	2n	$N/2$
Ring	N	N	$N/2$	2
k-ary d-cube: d-dimension array with k elements in each dimension	$N=k^d$	Nd	$d[k/2]$	$2k^{d-1}$
Hypercube (Binary d-cube)	$N=2^d$	$d\left(2^{d-1}\right)=\frac{N}{2}\log N$	$d=\log N$	$2^{d-1}=N/2$;
2D mesh	$N=p^2$	$2\left(N-\sqrt{N}\right)$	$2(p-1)=2\left(\sqrt{N}-1\right)$	$p=\sqrt{N}$
nD mesh	$N=p^n$	$n\left(N^{n-1}-N^{(n-1)/n}\right)$	$2n(p-1)=2n(N^{1/n}-1)$	$p^{n-1}=N^{(n-1)/n}$
2D Torus	$N=p^2$	$2N$	$2\left[\frac{\sqrt{N}}{2}\right]$	$2p=2\sqrt{N}$
nD Torus	$N=p^n$	nN	$n\left[\frac{N^{1/n}}{2}\right]$	$2p^{(n-1)}=2N^{(n-1)/n}$
Butterfly (k-network order)	$N=(k+1)2^k$	$\frac{N}{2}\log N$	$\log N$	N
Crossbar	$N=n^2+2n$	N^2	1	N
Omega	N	$N/2$	$\log N$	$N/2$
Dynamic Tree	N	$N-1$	$2\log N$	1

Table 2. Performance and cost values of Fat Tree network topology

k-ary n-layer Fat Tree	**n-layer**	**2-layer**	**3-layer**	**4-layer**
Cost Values:				
Number of Host supported, N	$2(k/2)^n$	$2(k/2)^2$	$2(k/2)^3$	$2(k/2)^4$
Number of Pods	$2(k/2)^{n-2}$	NA	k	$2(k/2)^2$
Number of Core Switches	$(k/2)^{n-1}$	k/2	$(k/2)^2$	$(k/2)^3$
Number of Edge Switches	$2(k/2)^{n-1}$	k	$2(k/2)^2$	$2(k/2)^3$
Total Switches	$(2n-1)(k/2)^{n-1}=(2n-1)N/k$	$3(k/2)=3N/k$	$5(k/2)^2=5N/k$	$7(k/2)^3=7N/k$
the number of edge/aggregation switches in each Pod	$(k/2)^{n-2}$	1	$(k/2)$	$(k/2)^2$
Each Pod connects to # servers	$(k/2)^{n-1}$	k/2	$(k/2)^2$	$(k/2)^3$
Number of Links	nN	2N	3N	4N
Performance Values:				
Bisection Bandwidth	$N/2=(k/2)^n$	$(k/2)^2$	$(k/2)^3$	$(k/2)^4$
Diameter	2n	4	6	8

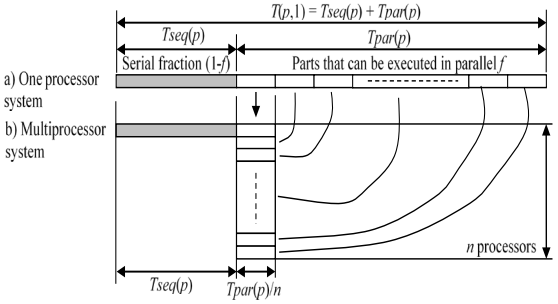

Fig. 2. Interpretation of the parallel program execution time in the multiprocessor system.

The theoretical speedup achieved when running a problem on a multiprocessor system is:

$$S(p,n)=\frac{T(p,1)}{T(p,n)}=\frac{T_{seq}(p)+T_{par}(p)}{T_{seq}(p)+\frac{T_{par}(p)}{n}} \quad (2)$$

Parallel programming algorithms must be such that the sequential part takes much less time than the time execution parallel part, that means:

$$T_{par}(p) >> T_{seq}(p)$$

Let p be the size of the program code to execute and the time to execute the program and output the result (the sequence of the code): $(15000+p)\mu s$. Parts of the program can execute in parallel with time: $\left(\frac{p^2}{64}\right)\mu s$.

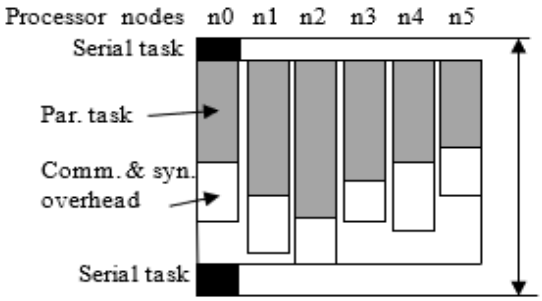

Fig. 3. Load imbalance makes a variation in execution time and transmission between tasks.

Then, the maximum speed up for a program size p=10000 calculated by (2) is:

$$S(p,n)=\frac{T(p,1)}{T(p,n)}=\frac{T_{seq}(p)+T_{par}(p)}{T_{seq}(p)+\frac{T_{par}(p)}{n}}$$

$$=\frac{25000+10^6}{25000+10^6/64}=29.3$$

Parallel programming algorithms can create good or bad load balance for the processing nodes to perform tasks of an application problem. In addition to the sequential and parallel execution times on a system of n processing nodes, the times required for synchronization and communication between nodes that process parallel tasks are due to the load imbalance (Fig. 3). Synchronization can arise from unbalanced loads for all processor nodes, waiting for common execution points of tasks with shared data, or with result data for sequential algorithms. Load balancing refers to the practice of approximately equally distributing workloads between tasks such that all tasks are busy all the time. This can minimize task idle time. Load imbalance creates a large overhead for synchronization and data transmission between processing nodes: $T_{trans}(p,n)$+$T_{syn}(p,n)$. The small or large transmission latency also depends on the interconnection network characteristics such as cost values (number of server nodes, number of links, number of switches/routers), and performance values (bisection bandwidth, diameter), the data packet size. In addition to transmission and synchronization overhead, there are other time delays $T_{etc}(p,n)$: due to starting tasks, finishing tasks, extra computation overheads, parallel compilers, libraries, operating systems, etc….

Thus, the actual execution time of an application problem of size p on a parallel computing system consisting of n processors must have a communication delay component $T_O(p,n)$.

$$T_O(p,n)=T_{trans}(p,n)+T_{syn}(p,n)+T_{etc}(p,n) \quad (3)$$

With the fixed execution time and size of the program: p=10000, time it takes to execute the program and output the result (the sequence of the code) is a $(15000+p)\mu s$ and parts of the program can execute in parallel with time: $\left(p^2/100=1000000\right)\mu s$.

Then the actual speedup should be as follows:

$$S(p,n)=\frac{T(p,1)}{T(p,n)}=\frac{T_{seq}(p)+T_{par}(p)}{T_{seq}(p)+\frac{T_{par}(p)}{n}+T_O(p,n)} \quad (4)$$

Since the communication overhead depends mainly on the packet size the interconnection network parameters, where the diameter và bisection bandwidth integrates cost and performance values (number of nodes, number of links, maximum hop count, and channel bandwidth). It is possible to define a communication overhead as a function of diameter, the bisection bandwidth, and the number of links:

$$T_O(p,n)=T_{trans}(p,n)+T_{syn}(p,n)+T_{etc}(p,n)$$
$$\geq\frac{\frac{T_{par}(p)}{n}(Diameter)}{(Bisection\,bandwith)(Numberof\,links)} \quad (5)$$

From here we can determine the extending formula of Amdhal's law considering the communication overhead in dependence mainly on values of the interconnection network topology: diameter, bisection bandwidth and number of links.

$$S(p,n) \le \frac{T_{seq}(p)+T_{par}(p)}{T_{seq}(p)+\frac{T_{par}(p)}{n}+\frac{\frac{T_{par}(p)}{n}(Diameter)}{(Bisection\,bandwidth)(Number\,of\,links)}} \quad (6)$$

Table 3. Speedup in extending amdahl's law of several interconnection network topologies.

Network Topology	Number of processor nodes							
	64	128	256	512	1024	2048	4096	8192
2Dtorus	25.1929	31.2236	35.4548	38.0276	39.4583	40.2145	40.6035	40.8008
2Dmesh	25.0801	31.1801	35.4408	38.0236	39.4572	40.2142	40.6034	40.8008
Hypercube	25.2286	31.2378	35.4594	38.0290	39.4587	40.2146	40.6035	40.6035
8-ary n-level Fat Tree	25.2213	31.2363	35.4592	38.0289	39.4586	40.2146	40.6035	40.8008
4-ary d-cube	25.0801	31.1223	35.4408	38.0262	39.4572	40.2138	40.6034	40.8007
Full connected	14.3584	21.2148	27.9381	33.2234	36.7018	38.7313	39.8332	40.4081
Omega	25.2137	31.2343	35.4588	38.0289	39.4586	40.2146	40.6035	40.8008
Butterfly	25.2213	31.2363	35.4592	38.0289	39.4586	40.2146	40.6035	40.8008

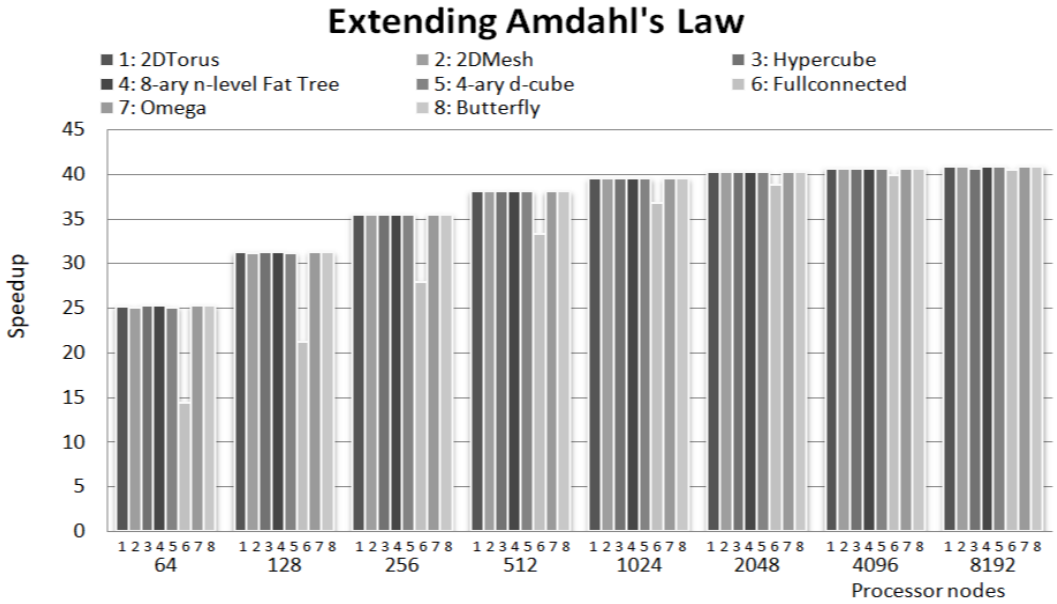

Fig. 4. Chart of Speedup results from Table 3.

Rewrite 2Dtorus network topology:

$$S(p,n) \le \frac{25000+10^6}{25000+\frac{10^6}{64}+\frac{\frac{10^6}{64}\sqrt[2]{64}}{\left(4\sqrt[2]{64}\right)64}} = 25.1929$$

Write for the execution time portion of the sequential part and the parallel part:

$$S(f,n) \le \frac{1}{(1-f)+\frac{f}{n}+\frac{\frac{f}{n}(Diameter)}{(Bisection\,bandwidth)}} \quad (7)$$

We can compare speedup values in Table 3 and Table 4 speedup values according to amdahl's extended law of different network topologies. Fig. 4 and Fig. 5 are shown in the chart of speedup results from Table 3 and Table 4, respectively. Table 5 shows the comparision speedup in Amdahl's law and Amdahl's law extension of the fat tree and 2Dtorus. Notice that, with a doubling of the time taken to execute the parallel part of the program while the time to execute the sequential part, the speedup nearly doubled.

$$S(p,n) = s + n(1-s) = n + (1-n)s \quad (8)$$

Table 4. Acceleration in amdahl's law extension of several topologies

Network Topology	Number of processor nodes							
	64	128	256	512	1024	2048	4096	8192
2Dtorus	35.922	49.8087	61.6999	70.0494	75.1291	77.9545	79.4481	80.2166
2Dmesh	35.6902	49.6968	61.6569	70.0356	75.1251	77.9535	79.4479	80.2165
Hypercube	35.9956	49.8453	61.7142	70.054	75.1304	77.9549	79.4483	80.2166
8-ary n-level Fat Tree	35.9805	49.8415	61.7134	70.0539	75.1304	77.9549	79.4483	80.2166
4-ary d-cube	35.6902	61.983	61.6569	70.0448	75.1257	77.952	79.4479	80.2164
Fully connected	17.194	28.2699	41.8592	55.1719	65.6284	72.506	76.5173	78.6947
Omega	35.9648	49.8363	61.7121	70.0537	75.1304	77.9549	79.4483	80.2166
Butterfly	35.9805	49.8415	61.7134	70.054	75.1304	77.9549	79.4483	80.2166

Table 5. Compare speedup in Amdahl's law and Amdahl's law extension of the fat tree and 2Dtorus.

	Number of processor nodes							
Speedup of 8-ary fat tree	64	128	256	512	1024	2048	4096	8192
Amdahl's law	36.0000	49.8462	61.7143	70.0541	75.1304	77.9549	79.4483	80.2166
Extending Amdahl's law	35.9805	49.8415	61.7134	70.0539	75.1304	77.9549	79.4483	80.2166
Speedup of 2DTorus	64	128	256	512	1024	2048	4096	8192
Extending Amdahl's law	35.9221	49.8087	61.6999	70.0494	75.1291	77.9545	79.4482	80.2166

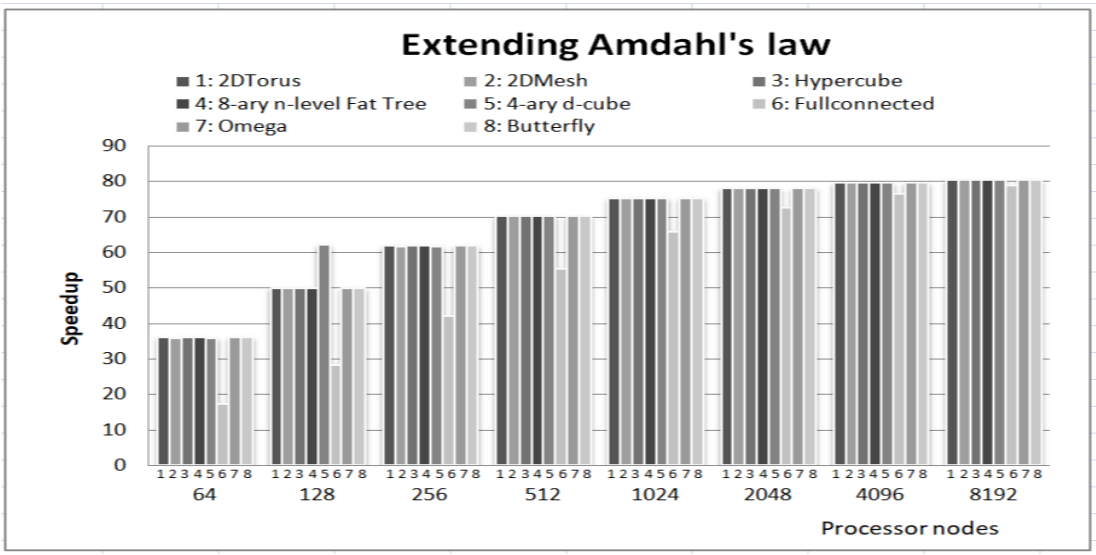

Fig. 5. Chart of Speedup results from Table 4

2.3. Extending Gustafson-Barsis Law

Amdahl law shows that the speedup of program execution increases proportionally to the size of the program (p) for a specified number of n processors. Thus, the program size must be increased, and the parallel execution is efficient with a specified number of processors. This is practically not always achieved. John L. Gustafson and his colleague Edwin H. Barsis, and was presented in the article *Reevaluating Amdahl's Law* in 1988 [11] define the scaled speedup.

This Gustafson-Barsis law overcomes the limitation of Amdahl's Law, and it determines how fast the number of processors increases for any given program size.

Where *s* and *(1-s)* are the fraction of time spent executing the serial parts and the parallel parts of the program on the multiprocessor parallel system.

Gustafson-Barsis law defines the scaled speedup including the execution time of sequence items s and a specific number of n processors, regardless of program size. However, this law also ignores the communication overhead that exists in parallel computing on multiprocessor systems. So, we include in Gustafson-Barsis law the time fraction *c* of the communication overhead *To (p,n)* inherent in parallel execution on a multiprocessor system.

$$s = \frac{T_{seq}(p)}{T_{seq}(p) + \frac{T_{par}(p)}{n} + T_O(p,n)}; \quad (9)$$

$$(1-s) = \frac{T_{par}(p)/n}{T_{seq}(p) + \frac{T_{par}(p)}{n} + T_O(p,n)} \quad (10)$$

$$c = \frac{T_O(p,n)}{T_{seq}(p) + \frac{T_{par}(p)}{n} + T_O(p,n)} \quad (11)$$

$$T_{seq}(p) = s\left(T_{seq}(p) + \frac{T_{par}(p)}{n} + T_O(p,n)\right); \quad (12)$$

$$T_{par}(p) = n(1-s)\left(T_{seq}(p) + \frac{T_{par}(p)}{n} + T_O(p,n)\right) \quad (13)$$

$$T_O(p,n) = \frac{\frac{T_{par}(p)}{n}(Diameter)}{(Bisection\,bandwidth)(Number\ of\ links)} \quad (14)$$

The scaled speedup formula of extending Gustafson-Barsis law is:

$$S(p,n) \le \frac{s + n(1-s)}{s + (1-s) + c} = \frac{s + n(1-s)}{1+c} = \frac{n + (1-n)s}{1+c} \quad (15)$$

When value c = 0, we have Gustafson-Barsis law (8).

The extending Gustafson-Barsis law (15) determines the scaled speedup including execution time of the sequence items and a specific number of n processors without considering the size of the program but considering the *c* - portion of the communication overhead. The c value can be calculated from *To (p,n)* according to the types of network connection topologies. For example, an application running on *n = 64* processors must spend s = 4% of its time on execution of sequential codes, so the speedup determined by (15) is:

$$S(p,n) \le \frac{n + (1-n)s}{1+c} = \frac{64 + (1-64)(0.04)}{1+c} = \frac{61.48}{1+c} \quad (16)$$

Knowing the maximum scaled speedup when *c=0*, and the number of processors, and the fraction of time to execute the sequential codes of a parallel program, we can choose interconnection network topology and transmission technology to ensure a small value c suitable for the required speedup.

It is also possible from the selection value *c* suitable for the interconnection network topology with the number of processing nodes we determine the execution times of the sequential parts and the parallel parts and the corresponding scaled speedup in the extending Gustafson-Baris law.

3. Result and Discussion

The nDTorus, k-ary n-level Fat tree networks are the types of networks that give small communication overhead and thus high speed up with increasing parallel parts size and the number of processing nodes. Proposals to extend Amdahl's law and Gustafson-Barsis law by introducing communication overhead that depends primarily on the characteristics of the interconnection network topologies in supercomputing systems can help computer designers with the choice of interconnection topologies and transmission technology to meet the performance requirements of applications, along with the choice of multi-core and multi-core technologies for CPUs and GPUs for computer nodes and in terms of energy efficiency. With current transmission technologies, the larger the number of computer nodes, the larger the size of the network connecting them, and the more communication overhead affects the performance of the supercomputer's parallel computation. Because it is not possible to build good parallel algorithms that eliminate the causes of communication delays for large and complex application problems running on supercomputers.

References

[1] GREEN500 LIST (accessed in June, 2022). Online Avaiable: https://www.top500.org/lists/green500/list/2022/06/.

[2] Anthony Spadafora, Next-gen Nvidia Mellanox InfiniBand will take supercomputers to the next level. Online Avaiable: https://www.techradar.com/news/next-gen-nvidia-mellanox-infiniband-will-take-supercomputers-to-the-next-level. published November 19, 2020.

[3] InfiniBand Accelerates Six of the Top Ten Supercomputers in the World, Including the Top Three, and Four of the Top Five on June's TOP500 (accessed in June 17, 2019). Online Avaiable: https://nvidianews.nvidia.com/news/infiniband-accelerates-six-of-the-top-ten-supercomputers-in-the-world-including-the-top-three-and-four-of-the-top-five-on-june-s-top500

[4] T. Zidenberg and I. Keslassy and U. Weiser, MultiAmdahl: How Should I Divide My Heterogenous Chip? IEEE Computer Architecture Letters, Volume: 11, Issue: 2, July-Dec. 2012. https://doi.org/10.1109/L-CA.2012.3

[5] János Végh, How Amdahl's low restricts supercomputer applications and building ever bigger supercomputers. University of Miskolc, Hungary. [v1] Fri, 4 Aug 2017 11:56:45 UTC (44 KB). [v2] Fri, 29 Dec 2017 06:26:45 UTC (33 KB)

[6] Uri Verner, Avi Mendelson, and Assaf Schuster, Extending Amdahl's Law for Multicores with Turbo Boost. Dept. of Computer Science, Technion, Israel. Published 2017. Computer Science IEEE Computer Architecture Letters. https://doi.org/10.1109/LCA.2015.2512982

[7] S. Pei, M-Seo Kim, and J. Luc Gaudiot, Extending Amdahl's Law for Heterogeneous Multicore Processor with Consideration of the Overhead of Data Preparation. January 2016. IEEE Embedded Systems Letters 8(1) https://doi.org/10.1109/LES.2016.2519521

[8] Chaitanya Poolla, Rahul Saxena, On extending Amdahl's law to learn computer performance. Microprocessors and Microsystems. Volume 96, February 2023, 104745. © 2022. Elsevier B.V. All rights reserved. https://doi.org/10.1016/j.micpro.2022.104745

[9] Ami Marowka, Extending Amdahl's Law for Heterogeneous Computing. Conference: Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on. https://doi.org/10.1109/ISPA.2012.47

[10] Yuichiro Ajima, The Tofu Interconnect D for Supercomputer,. Fujitsu Limited, June 20 2019

[11] Gusfson John L, Gustafson's Law. All content following this page was uploaded by John Gustafson on 24 January 2022.