UDC: 004.02, 004.2

METHODS OF EFFECTIVIZATION OF SCALABLE SYSTEMS: REWIEW

O.O. Honcharenko, H.M. Loutskii

The article discusses the problem of inefficiency of modern systems and horizontal scaling as a method of increasing productivity. The main issues that make up the mentioned problem are highlighted. A classification for possible solutions was proposed, according to which they were divided into architectural and network, and an overview was carried out. As part of the architectural class, such approaches as quantum computing and the dataflow paradigm were reviewed, the most promising solutions were analyzed. The comparative analysis shows that by their nature dataflow and quantum computing do not contradict each other, moreover, they complement each other in the context of the problem. At the same time, both types of processors require a certain network for communication, which makes the issue of topology relevant. At the network level, 2 topologies - Fat Tree and Dragonfly - were considered, and their main properties were highlighted. The analysis showed that in the context of the problem Dragonfly is slightly better due to decentralization and smaller diameter. In the conclusions, the main aspects of problem formulation and review are indicated, further prospects and possible methods are considered.

Keywords: *effectivization, scalable systems, high performant computing, architecture, topology*

Introduction

There are 2 main methods of scaling - horizontal and vertical. Vertical scaling is associated with increased performance of individual nodes, but this method has several disadvantages. First, it requires updating all nodes of the system, which is impractical from the point of view of finances. Secondly, the performance of individual processors is based on the clock frequency, the effective maximum of which is limited due to the complexity of the internal structure of the processor itself.

Another approach, horizontal scaling allows you to increase the peak speed linearly. However, there are problems here too. First, Amdahl's law says that dependencies between parallel parts of the task limit its parallelism. Thus, the dependence between the scope of the task, the scope of the system and the acceleration is established. Secondly, the properties of the system itself and the delays that occur in the execution process limit the real user acceleration. So, when the processor cannot continue execution, it must block the task, resulting in idle time. The solution to this problem is to switch to another task, but context switching requires significant time costs. As a result, a significant part of the calculations is occupied by idle time and interruptions. At the same time, energy consumption is proportional to the number of nodes, which raises the question of the feasibility of scaling in each specific case. The only way out of this situation is the modernization of the architecture and computing model: the search for such solutions that would allow either to gain in speed or to reduce costs. This makes the issue of efficiency *urgent*.

Reasons for limitation

The problem of efficiency is complex. This is not a single issue - it is a complex of issues that need consideration, analysis and resolution. Of course, their complete analysis is a separate topic for research, but even superficially analyzing the situation, one can highlight the key limitations of modern parallel supercomputer solutions.

1. *Problem of parallelism*. As mentioned earlier, this problem is that the classical model of computing has certain disadvantages that arise from the architecture of the processor.
2. *Problem of parallel programming*. The fact is that each parallel system is unique and has its own architectural features. At the same time, the task of parallelizing the algorithm is the prerogative of the programmer, which makes the development of each program difficult, and the program itself is architecturally tied to the system (or series of systems) under which it was written.
3. *The problem of inconsistency between the algorithm and the system*. This problem is related to specialized algorithms (for example, AI problems or genetic algorithms). These tasks have a high

degree of internal parallelism, but often during their execution there are restrictions dictated by the system architecture, which limits this internal parallelism. This raises the question of adapting not only the task to the system, but also the system to the task being performed.

4. *The problem of the balance of speed and costs*. A large number of computing facilities allows for acceleration, but the hardware comes at a price and consumes energy. A small amount of equipment limits consumption, but reduces performance. The question arises - how to achieve a balance of resources for each specific task within the framework of the system.
5. *The problem of node interaction*. A large number of nodes in the system leads to the fact that the only way to connect them is the internal network. But at the same time, a number of issues arise, such as the issue of fault tolerance, the issue of bandwidth, the issue of load balancing and the issue of the topological structure of the network as such. Both real performance and the possibility of practical application of the system as such depend on the effectiveness of their solution.

Analysis of the subject area

To find potential solutions, it makes sense to conduct a brief analysis of the subject area. This will allow you to highlight areas that need to be reviewed. Since the subject area is not monolithic, but consists of parts, the considered solutions cannot be called directly focused on it - they are directed only at specific aspects of it, but collectively they can give the key to a general solution. What aspects of the area can be highlighted and what exactly should be included in the review? In order to answer this question, it is necessary to clearly define which parts of the system the action of this or that method is aimed at. In general, the following classes of possible solutions can be distinguished:

1. *A change in architecture or computing paradigm*. Within the framework of this class, it is assumed that the Von Neumann architecture will be abandoned and a transition to other hardware and a different model of calculations will be made. At the same time, both hardware and software of the system are completely changed.
2. *Changing the principles of communication*. Within the framework of this class, the structure of individual computing elements (processors, nodes) is not considered - instead, the object of influence is the means of communication (communication lines, switches), communication protocols, the general structure of the system.
3. *Changing the approach to planning or task allocation*. Within the framework of this class, the methods of dividing a task into subtasks, scheduling calculations and assigning resources are the object of research. In general, this class makes sense to explore when it comes to automatic parallelization.

Within the framework of this review, it is proposed to pay key attention to the first class, since it is the most difficult to implement, but also the most promising. What is mentioned in modern discourse as an alternative to classical systems? As a rule, the most mentioned areas are the following:

1. *Quantum computing*
2. *Dataflow computing*

Analyzing the second class, it can be divided into two subclasses that partially overlap, but at the same time are relatively independent. These subclasses address the following issues:

1. *Network hardware and software*. This includes types of network equipment such as switches or routers, existing network protocols (Gigabit Ethernet, InfiniBand) and architectures (SDN).
2. *Structural composition*. This includes how the elements are connected in the network - that is, its topology. This includes 3D-Tor, Dragonfly, fat trees and other solutions.

As for the third class, it deals with scheduling and how a task can be automatically divided into subtasks. Often, this issue depends on the programmer or the OS, but in planning, 2 key approaches can be distinguished: static and dynamic. Since this question depends, not least, on the architecture, it makes sense to postpone its review until the review of the key decisions of the previous classes.

Quantum computing

Quantum computing is a very promising field of computer science, to which a large number of works are devoted. Although the practical use of modern quantum computers is limited by their high cost and small number of qubits, a number of areas have been highlighted in which they can be effectively applied, such as cryptography [1], financial analytics [2], scientific computing [3]

There are a number of approaches to implementing quantum computers, but the most popular one is based on the idea of a quantum circuit and the concept of a qubit. A qubit is a quantum unit of information that can enter a state of superposition between two binary states 0 and 1. As a result, at the time of calculations, the quantum register is simultaneously in all possible states, but with different probabilities. The quantum algorithm changes this probability to bring the quantum register closer to the response state, after which a readout is performed and decoherence occurs into one of the possible states [4].

This approach allows you to perform calculations not on one, but on all possible values of a variable, which makes it possible to get acceleration in a number of problems, thereby solving computational problems that cannot be solved on classical systems in a reasonable time. This property is called quantum advantage. It has been proven that certain minimum hardware characteristics of the system must be ensured in order to demonstrate quantum superiority, namely: a dimension of 49 qubits, a circuit depth of 40 and a two-qubit error of no more than 0.5% [5]. At the moment, there are a number of systems that meet these requirements. For example, Google's 54-qubit Sycamore processor, which can complete a task in 200 seconds that would take a classic supercomputer 10,000 years [6]. Or the Chinese quantum photonic computer Jiuzhang, which is the second in the world to achieve quantum supremacy [7].

The described systems are full-fledged quantum computers of general purpose, but in the context of the mentioned problems, they have a lot of shortcomings. Their key limitation is the price: they are too expensive. In addition, this is not a mass-produced product that can be purchased just like that: each of these systems is a separate artificial product under the control of the developing organization/country, and therefore it is still too early to talk about the prospects for their commercial use. However, it is worth considering that the development of quantum computers does not stand still. For example, research is being conducted to speed up the development of new quantum systems. For example, Fig. 1 shows a diagram of such a full-stack development of a quantum computer based on transmons.

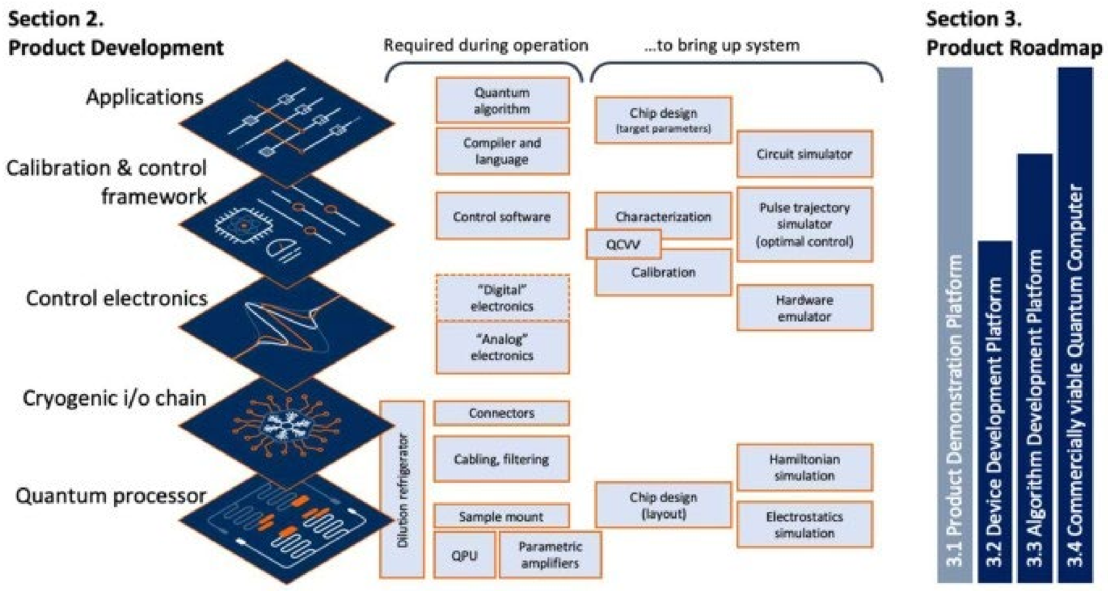


Fig. 1. Scheme of full-stack development of a quantum computer based on transmons [8]

This allows us to say that although modern quantum systems are not a potential solution to the problem of efficiency, the emergence of new, more advanced systems can fundamentally change this situation. Thus, it is necessary to take this fact into account and, considering other architectural solutions, leave room for heterogeneity and quantum integration.

However, such quantum systems are not the only solution. An alternative implementation is presented by the D-Wave company, whose processors have reached a size of 1000 qubits and continue to grow. This became possible thanks to the use of quantum annealing [9]. This method, focused on finding the global minimum of the function, is based on the fact that at the beginning of the calculation, all states of the register have equal probability, but in the process of quantum evolution, there is an ascent to a state in which the energy is minimal. This allows the processor to effectively solve optimization problems and significantly reduces the complexity of scaling, but makes it unsuitable for the execution of ordinary quantum algorithms. On the other hand, within the proposed class of tasks, this type of processor demonstrates extremely high efficiency, besides, D-Wave processors, although they have a considerable price, are available for commercial use. Therefore, it makes sense to consider their architecture in more detail.

In fig. 2 presents the architecture of one of the latest D-Wave Advantage quantum computers, which consists of several layers: an application layer, an Ocean software layer that deals with compilation and assignment of tasks, and a computer resource layer that contains 2 types of processors: classical CPUs and quantum QPUs.

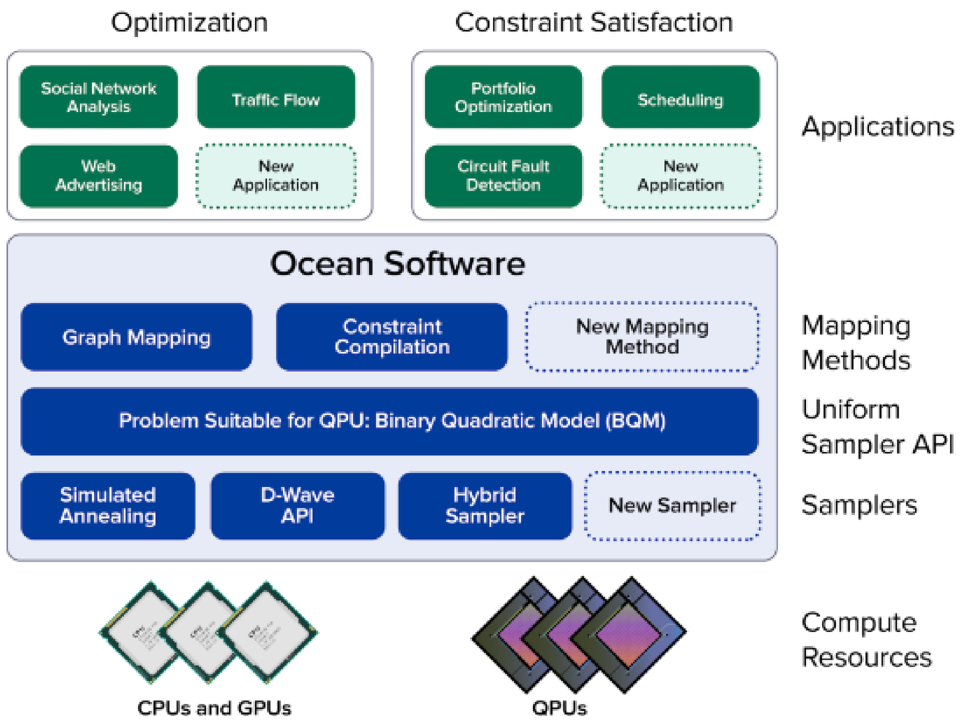


Fig. 2. D-Wave Advantage architecture [10]

Thus, D-Wave's approach allows you to make the system heterogeneous from the very beginning, combining fast, but task-limited QPUs with general-purpose CPUs. This makes it possible to almost completely eliminate any problems related to specialization, as well as to ensure efficient parallelization of calculations.

However, the question arises: how are the QPUs themselves arranged? Just as a classical system consists of nodes connected by a network, a quantum system consists of qubits connected by a graph of possible quantum entangled pairs. In the process of executing a quantum algorithm, the program connects qubits into a general quantum system, using certain connections from those available on the graph, and then performs certain transformations on the system. Thus, the characteristics of a graph are extremely important for a quantum system, because it depends on them which quantum algorithms it can perform and how quickly.

As an example, it makes sense to consider the Chimera graph presented in Fig. 3. This graph includes qubits represented in the form of horizontal and vertical loops that form a lattice. The intersection points of these loops inside each lattice implement internal connections, external connectors connect qubits in a single row or column.

However, newer systems such as the D-Wave Advantage use a slightly different graph called Pegasus (Fig. 4). It differs in that the lattice in it has an offset, which allows to expand the system of connections. It is assumed that the new D-Wave processors will use it (as well as the Zephyr graph) and not the classic Chimera graph. This graph has 3 types of connections: internal (green vertical line) connecting pairs of orthogonal horizontally oriented qubits; external (blue line) that connect adjacent vertical qubits; odd (red line) connecting equally aligned pairs of qubits.

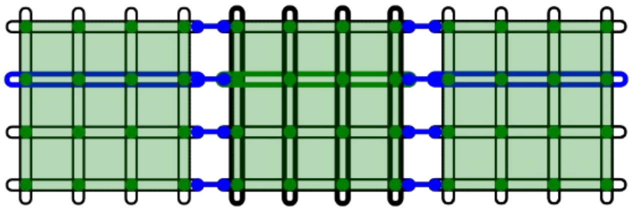


Fig. 3. Chimera graph [11]

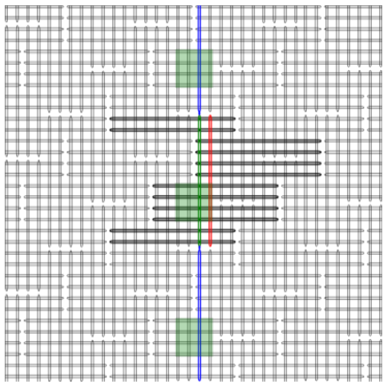


Fig. 4. Pegasus graph [11]

If you bring it to a more familiar form, with nodes and communication lines, you can get the system shown in Fig. 5. In it, green dots represent elements, green lines - connections, gray lines - splitters.

However, it is assumed that the company's future processors may switch to other topological organizations - for example, the Zephyr graph, which is presented in Fig. 6. The feature of this graph is the degree of 20 and the nominal length of 16, which potentially makes it possible to implement more qubits on a chip, as well as connect them more efficiently.

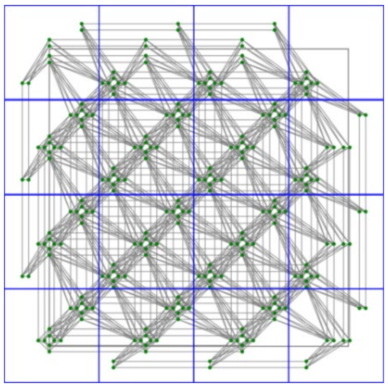


Fig. 5. Pegasus graph – topo representation [11]

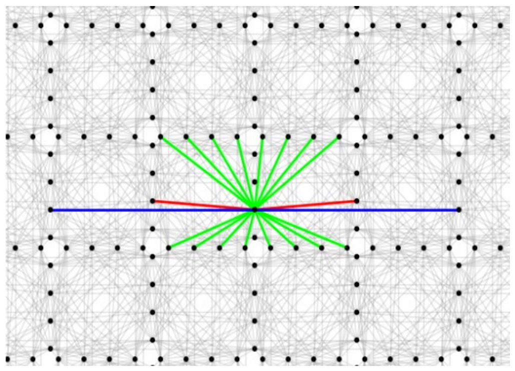


Fig. 6. Zephyr graph [11]

In the context of the problems of the subject area, quantum computing is interesting for the reason that quantum parallelism is an almost ideal form of parallelism as such. The calculation is always performed simultaneously over the entire field of solutions, independently processing each of them, and the subject of processing is not only the values themselves, but the probabilities of obtaining them when reading. This makes it possible to almost completely eliminate data dependencies from the problem, which is one of the key factors of slowdown. Also, quantum problems do not know the inconsistency of the algorithm and the task or balance problems: yes, for each task, the amount of "quantum flows" that it needs is allocated, and it is usually simply impossible to perform the task with a smaller number of qubits than it needs. The key limitations of quantum computing are its cost and the need to develop special quantum software to gain real benefits.

Dataflow as alternative paradigm.

In terms of computing management, 2 key paradigms are distinguished: contolflow and dataflow. The first approach is classic: the system has a command counter, and the program is presented in the form of a strict sequence of instructions. Accordingly, the parallel contolflow system contains a number of devices that have their own counters, and the control relies on processes and flows - a sequence of commands that is the result of breaking down the original task. However, from the point of view of parallelism, this model is not convenient: it loses internal parallelism, additional dependencies and idles appear, and when multitasking - context switching.

An alternative is the dataflow approach, in which the task is presented in the form of computational tasks connected by data. Execution occurs when ready: ready-made tasks can be executed when a free resource is available, regardless of the sequence in which they are described in the program.

Historically, the first dataflow machines implemented parallelism at the command level. However, this approach turned out to be ineffective due to high overhead costs. Therefore, a number of concepts were formed, which can be conventionally divided into 3 ways. The first path, which can be conventionally called hybridization, gave rise to modern RISC processors and superscalar architecture [12]. The idea of this approach is to hide the dataflow component inside another architecture, thereby using common tools at the programming level, and performing dependency detection at the command level, thereby speeding up the execution of independent parts.

The second way, emulation, is based on describing the tasks and data relationships at the programming level and implementing a parallel algorithm so that its parallel parts run when ready, regardless of how they are actually written in the program. Hardware-wise, the system remains pure contolflow. This includes the dataflow approach to programming and modern parallelizing compilers that simulate execution in a dataflow system, and then form a parallel algorithm from a sequential one based on the model [13, 14].

The third way to solve the problem is the development of the architecture itself: streaming, coarse-grained, vector dataflow machines, dataflow networks, and FPGA-based devices oriented to specific tasks. Within the framework of the described problems, it makes sense to focus on them and their hardware components.

Threaded dataflow and coarse-grain solutions extend the classic fine-grained parallelism of dataflow with the possibility of parallelism at the level of threads and subtasks - that is, medium and coarse-grained. This approach allows to reduce dynamics overhead due to the fact that the execution time of each specific task becomes much greater than the time of hardware planning of calculations, which eliminates delays in the execution of individual commands and reduces the need for associative memory. Sometimes dataflow modules in such systems are offered as an add-on to conventional von Neumann processors used as processing elements, but this approach is criticized for the fact that such a system by definition cannot work faster than classic controlflow. An alternative here is the use of simpler elements.

The approach of vector dataflows is similar, but slightly different: instead of increasing a single grain (and increasing the number of operations at the same time as the processing time increases), they propose to increase the volume of the same type of calculations performed simultaneously, due to vectorization. So, in this type of system, the commands are not scalar, but vector, which allows you to ignore the planning time due to the general acceleration of calculations from vectorization.

The third hardware approach, which is extremely popular today, is the use of FPGAs as the hardware base for the system [15, 16]. This approach is based on several things: firstly, the dataflow system is developed here for a certain task, which allows you to optimize it and, if necessary, include additional architectural elements - pipeline, vectorization, matrix or coarse-grained approach. Secondly, due to the manipulation of the number of devices, a certain balance between energy consumption and speed is achieved, which is especially relevant for embedded devices [15].

In the context of modern high-performance dataflow computing, the best known is the dataflow concept of the Maxeler company, which uses FPGA-based accelerators and the parallelizing compiler MaxCompiler, which turns a sequential controlflow program into a parallel dataflow application. This approach solves a number of applied issues at once: on the one hand, it harmonizes new solutions with previously created hardware and software, and on the other hand, it allows you to get a real acceleration and make the system more efficient. Therefore, it makes sense to pay attention to him.

How does this system work? Fig. 7 shows the main stages of its work. First, the Java program is compiled into a .max file of parallel tasks. It then passes through Maxeler's operating system and system software, where some of the computing scheduling tasks are performed. In the next step, the scheduled computing tasks are submitted to the hardware dataflow accelerator (DFE).

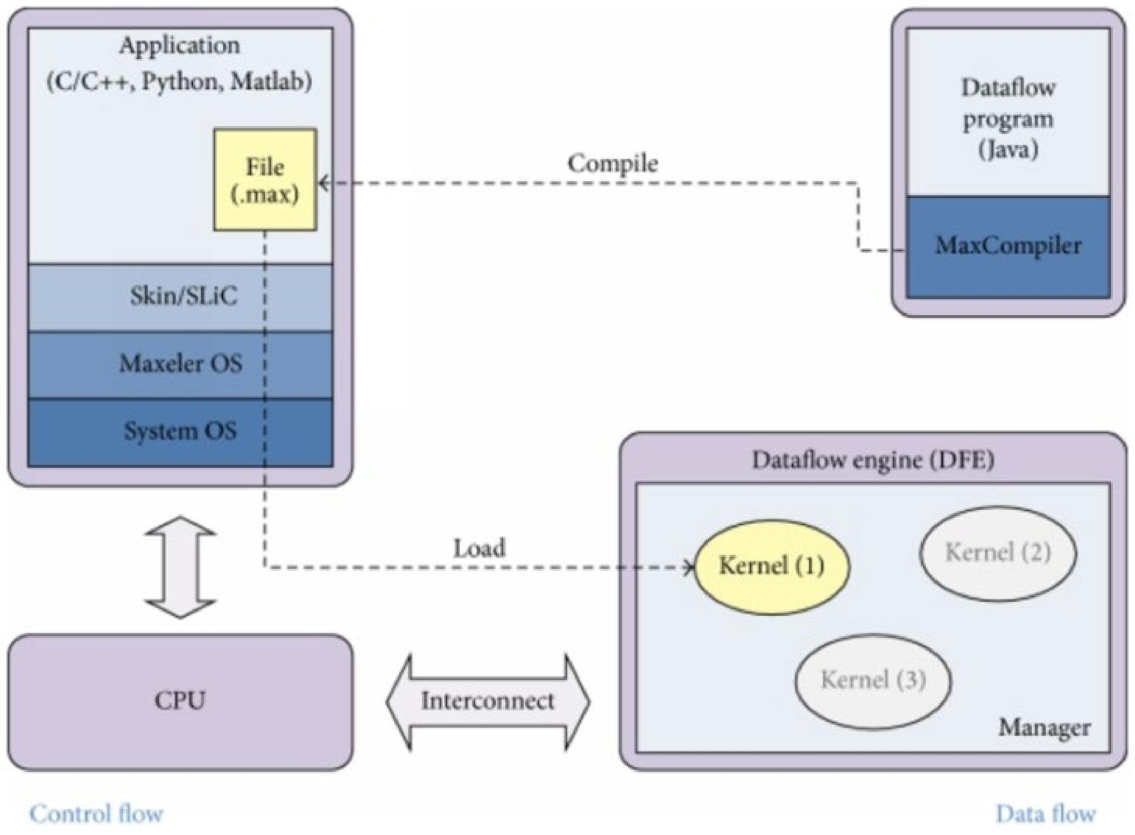


Fig. 7. Maxeler MPCarchitecture [17]

However, this is a general MPC architecture. To better understand how this should work in high-performance computing, it is worth diving deeper into the solutions offered by the company. At the moment, 4 architectures are offered for cluster and network computing (MPC-X, MPC-C, MPC-N, JDFE), the MaxCloud infrastructure, which provides dataflow computing services in the cloud, as well as the Desktop version of the accelerator.

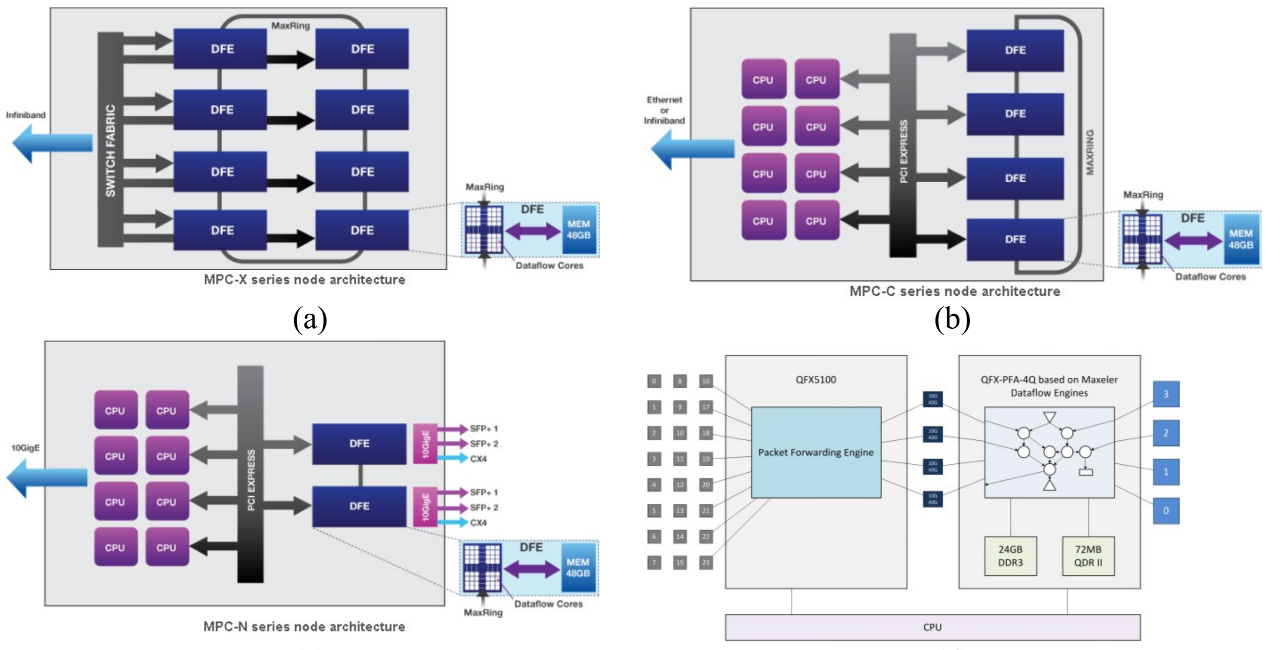
You should start with the MPC-X series. It is focused on the maximum use of dataflow calculations and in the latest MPC-X2000 implementation contains 8 MAX4 (Maia) processors in each node, connected inside the node using MaxRing and the InfiniBand network at the general level. Thus, individual MPC-X2000 nodes contain up to 768 GB of memory and provide more than ten times the acceleration compared to classic x86 servers [18].

A different approach is offered in the MPC-C series. Unlike MPC-X, it is a hybrid, combining CPU and DFE. Each MPC-C500 node contains 4 Vectis DFEs and 12 Intel Xeon CPUs, and provides a total of up to 192 GB of internal memory per CPU and per DFE (up to 384 GB of memory per node). InfiniBand or Ethernet can act as a common network here [19].

The MPC-N series is focused on minimizing delays and fast processing of data streams with a transmission speed of up to 10 Gb/s. In addition to 2 DFE Vectis and 12 Intel Xeon, its nodes contain 4 SPF/SPF+ ports, 2 CX4 ports, and also provide additional synchronization capabilities and network protocol support [20].

The latest series, JDFE aims to combine software-defined networking (SDN) technology and dataflow computing, separating the control plane and the data plane and enabling data plane programming via Maxeler Dataflow. Such a system contains thousands of small dataflow cores, using massive parallelism of calculations and providing almost a 100-fold advantage in speed while maintaining the size and power consumption at the level of a classical system [21].

In fig. 8 shows the architecture of the main 4 series of the Maxeler company, a brief overview of which is given above.



(a) (b)

(c) (d)

Fig. 8. Maxeler's main architectural solutions are: (a) MPC-X series architecture, (b) MPC-C series architecture, (c) MPC-N series architecture, (d) JDFE series architecture

In the context of the described issues, dataflow as a paradigm has 2 main advantages. First, it provides a much higher speed of operation with the same dimensions and the amount of energy consumption of the system. Secondly, the implementation of the dataflow system does not require a non-classical element base (such as qubits), and the presence of programmable FPGA logic circuits allows you to avoid problems associated with the deployment of mass production. A partial drawback is the specificity of dataflow programming, but as Maxeler's experience shows, this problem is not insurmountable.

Network level and its specifics

There are a significant number of aspects in this subject area. Conventionally, they can be divided into hardware and topological, but such a division will not be completely accurate, since the hardware is strongly connected with the protocols, and the protocols - with the topology. Thus, some solutions are complex, while others involve a certain degree of freedom. Therefore, it makes sense to expand the classification:

1. *Hardware level solution*. This includes hardware software or hardware software complexes that offer only hardware tools and common communication protocols, but do not determine the topological level of the network. For example, InfiniBand or SDN.
2. *Structural level solutions*. This includes structural compositions that determine the relationship of elements and the specifics of routing, but do not determine the equipment that the network should consist of. This includes topologies such as 3D-Tor, hypercube or fat tree.
3. *Communication protocols*. This includes protocol solutions and algorithms that do not determine either the hardware or the topological organization of the network. For example, tabular routing protocols.
4. *Combined solutions*. This includes solutions in which hardware, protocol and topology are an integral whole. This includes, for example, the TokenRing protocol, which imposes certain requirements on the equipment and is oriented towards the ring topology.

Analyzing the given classification, one can see some specifics. Yes, hardware-level analysis is key, but it depends on the architecture-level decision, so it must be analyzed separately. Similarly, this applies to the combined approach. Protocol decisions are by definition quite abstract, but they should be chosen based on the needs of the system, which makes general analysis meaningless and even harmful. Thus, the question arises about the expediency of structural level analysis. This level is also abstract, but any network needs a topology, and the issue of routing and data transmission is relevant regardless of the specific architectural approach implemented at the node level. Thus, the current analysis of network solutions should be considered precisely in the context of topology.

As a result, the question arises: what topologies are used in modern systems? At the moment, Fat trees and Dragonfly are quite popular for high-performance systems and network data centers [22].

The idea of a fat tree is as follows: there are elements located in the leaves of the tree, and there are switches that make up the main part of the tree. The closer the switch is to the root, the better the bandwidth parameters it has. In fig. 9 presents the structure of this topological organization, which consists of 4 levels.

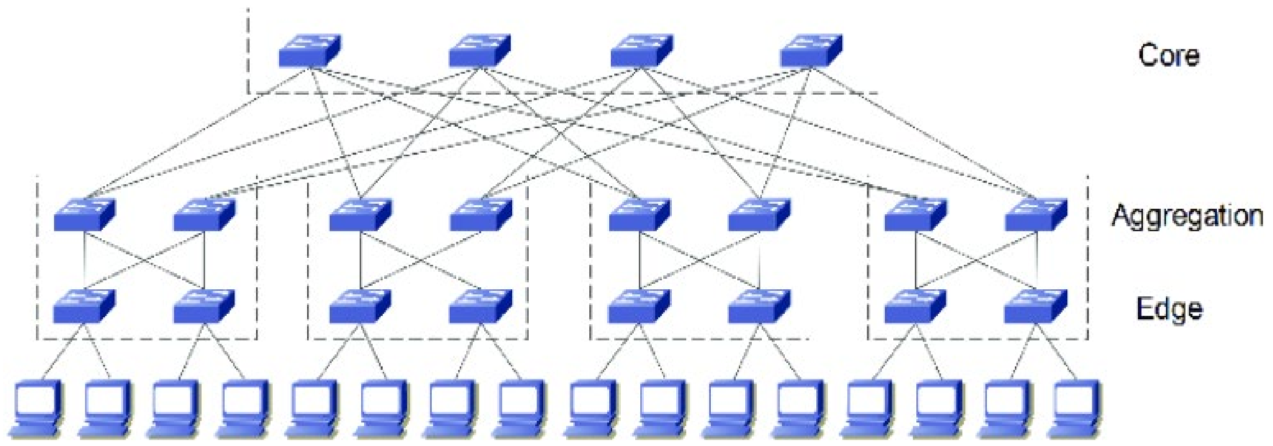


Fig. 9. Fat Tree topology [23]

What does it do in terms of performance? First, since the topology has redundant (compared to a regular tree) connections, it allows you to speed up routing and make it more reliable. Moreover, thanks to the use of multipath routing methods, it becomes possible to simultaneously transmit information through parallel channels, which are quite numerous in this topology. An equally important property of a fatty tree is its variability. Yes, there are a large number of ways to configure the topology depending on the needs and available hardware. Thus, this structural organization allows solving the main problems of supercomputers and data centers, which makes it popular. Yes, it is used in Summit, Sierra supercomputers, as well as other high-performance systems [24].

Another solution is offered by the Dragonfly topology. Its idea is fractality: yes, the system is divided into groups connected by a common network, while the groups consist of routers to which the nodes are connected. As a result, at each of the levels, the diameter of the topology is equal to (or close to) 1, and the number of nodes with scaling grows quite quickly. In fig. 10 presents the structure of the topology.

Like a fat tree, Dragonfly is a variable topology whose structure is defined by four parameters (p, a, g, h). At the same time, p is the number of terminal connections of nodes to the router, a is the number of routers in each group, g is the number of groups, and h is the number of external connections between groups. Varying these parameters allows you to change the characteristics of the network, and therefore - to select such configurations that would satisfy the purpose. In fig. 11 shows some variants of the Dragonfly topology depending on different parameters a, g and h.

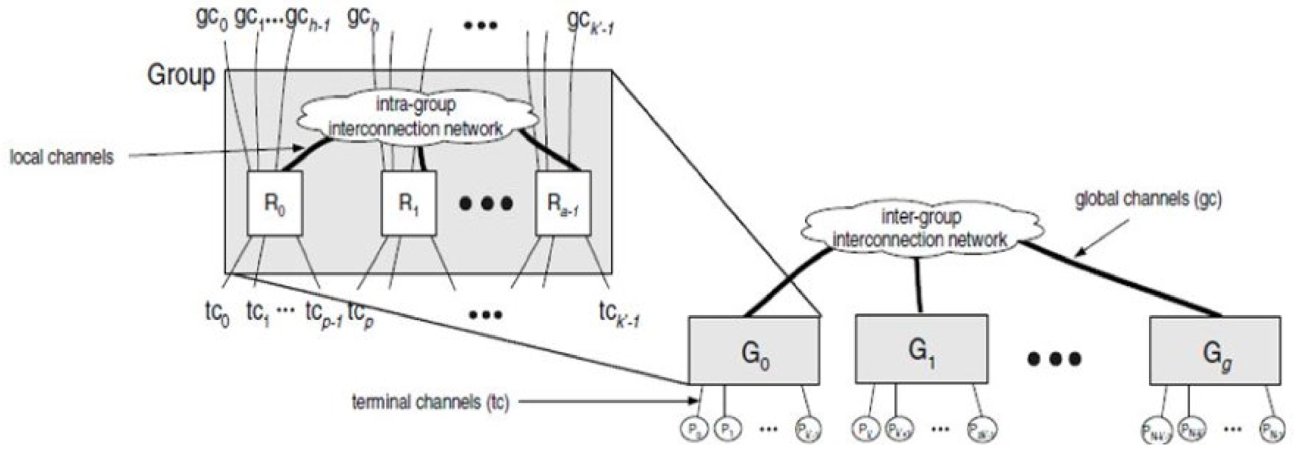
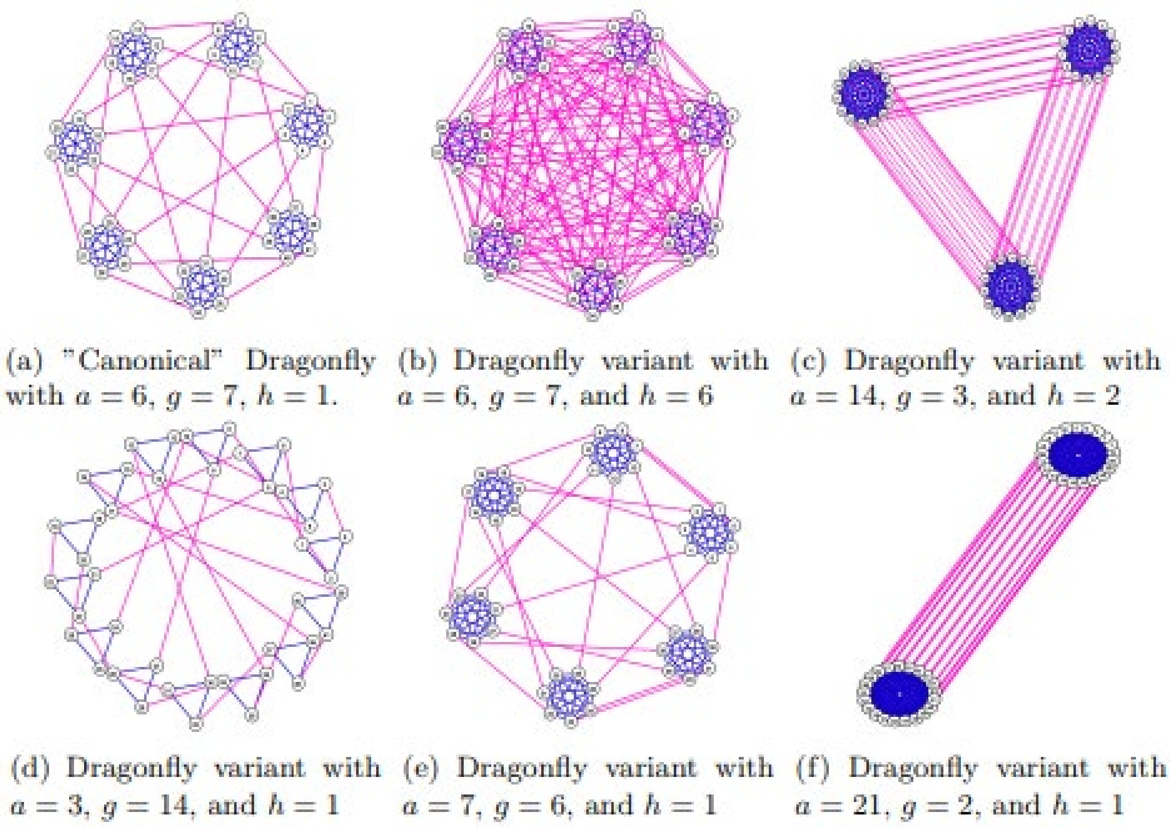

gc0 gc1...gch-1 gch ... gcK-1
Group
intra-group interconnection network
local channels
R0
R1
Ra-1
tc0 tc1 ... tcp-1 tcp ... tcK-1
inter-group interconnection network
global channels (gc)
G0
G1
Gg
terminal channels (tc)

Fig. 10. Dragonfly topology – structure [25]



(a) "Canonical" Dragonfly with $a = 6$, $g = 7$, $h = 1$.
(b) Dragonfly variant with $a = 6$, $g = 7$, and $h = 6$
(c) Dragonfly variant with $a = 14$, $g = 3$, and $h = 2$
(d) Dragonfly variant with $a = 3$, $g = 14$, and $h = 1$
(e) Dragonfly variant with $a = 7$, $g = 6$, and $h = 1$
(f) Dragonfly variant with $a = 21$, $g = 2$, and $h = 1$

Fig. 11. Dragonfly topology – variants [26].

Comparative analysis

Summarizing the review, it makes sense to analyze the solutions one by one. First, specific decisions within each direction. Then - the best decisions of directions within the class among themselves. Table 1 shows a comparison of classical quantum computers with specialized systems of the D-Wave company. The optimal parameters are highlighted in green.

As can be seen from the comparison, D-Wave computers (e.g., D-Wave Advantage), although limited in the class of tasks, provide approximately 16-50 times more resources (qubits) at a price that is 200 times lower. Also, it's worth noting that D-Wave processors are available for purchase, while general-purpose quantum systems are typically only available through a QC-as-a-service model. In the context of high-performance computing, where every clock counts, the lack of physical access to the chip is critical. This makes the choice unequivocal in favor of D-Wave systems.

Table 1.
Comparative analysis of quantum systems

Characteristic	Universal QS (quantum circuit	D-Wave QS (quantum annealing)
Types of tasks solved	All quantum algorithms with acceleration, classical algorithms with CPU speed.	Optimization problem with acceleration
Number of qubits	53 (Google Sycamore), 127 (IBM Eagle)	2048 (q2000), 5640 (Advantage)
Cost	$ 3.000.000.000 (IBM Eagle)	$ 15.000.000 (q2000)

As can be seen from the comparison, D-Wave computers (e.g., D-Wave Advantage), although limited in the class of tasks, provide approximately 16-50 times more resources (qubits) at a price that is 200 times lower. Also, it's worth noting that D-Wave processors are available for purchase, while general-purpose quantum systems are typically only available through a QC-as-a-service model. In the context of high-performance computing, where every clock counts, the lack of physical access to the chip is critical. This makes the choice unequivocal in favor of D-Wave systems.

Table 2 presents a comparison of solutions offered by Maxeler for dataflow computing. Since each of them has its own specific orientation, it is almost impossible to single out the best among them.

Table 2.
Comparative analysis Maxeler MPC Dataflow [18-20]

Characteristic	MPC-X2000	MPC-C500	MPC-N40	MPC-N42
CPU	-	12 Intel Xeon	12 Intel Xeon	
DFE	8 Maia	4 Vectis	2 Vectis	
CPU RAM	-	Up to 192 GB		
DFE RAM	96 GB per DFE	48 GB per DFE	24 GB per DFE	
Inner network	MaxRing	MaxRing (DFE), PCI Express (DFE-CPU)	PCI Express 8x2	
Global network	InfiniBand	Ethernet, InfiniBand	100GigE	
Диски	-	3 x 3.5”, 5 x 2.5”	3 x 3.5”	16 x 2.5”
Additional I/O ports	-	-	4 SFP/SFP+, 2 CX4	
Purpose	Computing	Computing, CPU-DFE hybridization	Data stream processing (10 Gbit/s) with minimal latency	

In the context of high-performance computing, the MPC-X series is the most interesting, but other types of nodes could also be useful depending on the specifics of the task and the overall network architecture.

After completing the analysis within each of the directions, it is useful to perform the same analysis between the directions within each of the classes. Table 3 compares the considered architectural directions, analyzing their suitability for solving the research problem.

As can be seen from the analysis, D-Wave quantum computers solve some problems almost perfectly, but their specialization makes them unsuitable for use instead of CPUs. Another solution is dataflow processors (DFE), which, as Maxeler's experience shows, can both replace classic CPUs and be used in cooperation. But their acceleration is limited.

Table 3.
Comparison of architectural level solutions

Problem	D-Wave quantum computing	Dataflow
The problem of parallelism	It is solved automatically in an ideal way	It is resolved automatically due to dynamics, there are overheads
Programming problem	Not solved: the system needs a special approach	There is potential for automation, there are automated means of detecting dependencies
The problem of matching the task and the system	It is partially solved, but not for all problems	Solved at the destination automation level (parallelism is exposed as fully as possible)
The problem of balancing productivity and costs	Does not occur: the problem can either be solved or not	Can be resolved through OS level or FPGA properties
Interaction problem	There is an interaction of qubits due to quantum entanglement	Available node interaction through classic data transfer

The following analysis, presented in Table 4, deals with networks, and more precisely, with network topologies. However, if architectural solutions need to be analyzed according to the problem, network solutions cannot be analyzed in a similar way, but topological characteristics of networks and their more general properties can be evaluated: for example, the availability of alternative routes, which is important in the context of multipath routing and fault tolerance.

Table 4.
Topological solutions

Characteristic	Fat tree	Dragonfly
Topological characteristics		
Degree	Depends on connectivity, but no more than 4-6.	p + a-1 + h [26]
Diameter	No more than a binary tree	Minimum, with full connectivity within each level – 5 (between terminal nodes)
Possibility to solution of the problem		
Alternative routes	Yes, due to additional inter-level connections	Yes, due to local full connectivity
Structure-oriented routing	Topology-based depth-first search	Based on the properties of connections
Bandwidth problem	Topology is focused on solving this problem	There are methods of topological routing that solve the problem
Optimal types of routing	Centralized	From the source or decentralized

Analyzing the topologies, one can notice certain similarities: both of them are polymorphic, have good characteristics and implement multipath routing. On the other hand, a fat tree relies heavily on centralized routing, where route discovery is performed by a shared controller, allowing for consideration of all nuances with increased bandwidth near the roots. On the other hand, the

Dragonfly offers greater decentralization and a potentially better diameter, making it more promising in terms of efficiency.

Discussions

Analyzing the given solutions, it is possible to draw a conclusion about the possibility of their combination. From the point of view of the dataflow network, the system is not much different from the classic controlflow, which allows you to use a network of any hardware type (InfiniBand, Ethernet, SDN) and any topology. At the same time, quantum systems are specific in this aspect. Combining qubits requires special graphs, as the overview of D-Wave graphs demonstrates, but hypothetically the network between QPUs operates on classical data and therefore can have any structure.

However, a much more interesting combination is the combination of two architectures: dataflow and quantum processors. The first architecture significantly increases the efficiency of calculations, but its acceleration is limited. The second - makes it possible to significantly simplify a number of problems due to quantum advantage, but is specialized. Although at the moment quantum chips are too expensive, there is a significant possibility that with the development of technologies, their price will decrease further, and therefore, at a certain point, it will be possible to use them in part of the nodes of the system as accelerators. As a result, the only question is in which specific parts of the system these nodes should be located. The topology comes in handy here, the right choice of which will allow for effective access to the quantum resource.

Conclusions

Summarizing the review, the following aspects should be noted. The first is that the problem of limited efficiency is complex and consists of a number of smaller problems. There is a large number of possible solutions for each of them, which makes their classification relevant. This article proposes to divide them into those related to the immediate structure of the node (architectural level) and those related to the interaction of nodes and abstracted from their architecture (network level), however, this classification is not final and can be expanded.

The second aspect is that there are a large number of possible solutions for each problem, both mentioned and not mentioned in this review. All of them have their own specifics and can be both compatible with each other and contradict each other. However, since each of the proposed approaches affects only part of the problem and does not solve the problem in general, there is a need to find methods that would allow combining partial solutions into a general one.

Returning to the material of the article, the reviewed solutions allow solving certain tasks: solving the problem of parallelism, partially - simplifying programming, adapting the task to the system and the system to the task, as well as solving interaction problems. Benchmarking shows that key solutions such as the dataflow paradigm and quantum computing complement each other and can therefore be combined for better results.

However, there are a number of unsolved tasks, such as the simultaneous use of QPU and DFE, the combination of quantum and classical algorithms within the program, the planning of calculations at the network and system-wide levels. Also, the question of the relationship between architecture and structure, as well as the search for optimal hardware and topological solutions at the network level, remains unresolved. Similarly, it makes sense to expand and deepen the review, including new modern architectures and paradigms and considering the issue of hybridization as a method of solving the given problem.

References.

1. Mavroeidis V. et al. The impact of quantum computing on present cryptography //arXiv preprint arXiv:1804.00200. – 2018.
2. Orús R., Mugel S., Lizaso E. Quantum computing for finance: Overview and prospects //Reviews in Physics. – 2019. – T. 4. – C. 100028.
3. Cao Y. et al. Quantum chemistry in the age of quantum computing //Chemical reviews. – 2019. – T. 119. – №. 19. – C. 10856-10915.

4. Nielsen M. A., Chuang I. L. Quantum computation and quantum information //Phys. Today. – 2001. – Т. 54. – №. 2. – С. 60.
5. Kelly J. A preview of Bristlecone, Google's new quantum processor //Google Research Blog. – 2018. – Т. 5.
6. Arute F. et al. Quantum supremacy using a programmable superconducting processor //Nature. – 2019. – Т. 574. – №. 7779. – С. 505-510.
7. Ball P. Physicists in China challenge Google's' quantum advantage' //Nature. – 2020. – Т. 588. – №. 7838. – С. 380-381.
8. Alberts G. J. N. et al. Accelerating quantum computer developments //EPJ Quantum Technology. – 2021. – Т. 8. – №. 1. – С. 18.
9. McGeoch C. C. Adiabatic quantum computation and quantum annealing: Theory and practice //Synthesis Lectures on Quantum Computing. – 2014. – Т. 5. – №. 2. – С. 1-93.
10. Quantum Computing [Електронний ресурс] // D-Wave Government – Режим доступу до ресурсу: https://dwavefederal.com/system/.
11. D-Wave QPU Architecture: Topologies [Електронний ресурс] // D-Wave System Documentation – Режим доступу до ресурсу: https://docs.dwavesys.com/docs/latest/c_gs_4.html.
12. Arvind, Brobst S. The evolution of dataflow architectures: from static dataflow to P-RISC //International Journal of High Speed Computing. – 1993. – Т. 5. – №. 02. – С. 125-153.
13. Carkci M. Dataflow and reactive programming systems //Create Space Independent Publishing Platform. – 2014.
14. Pochayevets O. BMDFM: a hybrid dataflow runtime parallelization environment for shared memory multiprocessors : дис. – Technische Universität München, 2006.
15. Gobieski G. et al. Manic: A vector-dataflow architecture for ultra-low-power embedded systems //Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture. – 2019. – С. 670-684.
16. Milutinovic V. et al. DataFlow Supercomputing Essentials. – Cham : Springer, 2017.
17. Kos A. et al. New benchmarking methodology and programming model for big data processing //International Journal of Distributed Sensor Networks. – 2015. – Т. 11. – №. 8. – С. 271752.
18. MPC-X Series [Electronic resource] // Maxeler Technologies. Government – Resource access mode: https://www.maxeler.com/products/mpc-xseries/
19. MPC-C Series [Electronic resource] // Maxeler Technologies Government – Resource access mode: https://www.maxeler.com/products/mpc-cseries/
20. MPC-N Series [Electronic resource] // Maxeler Technologies. Government – Resource access mode: https://www.maxeler.com/products/mpc-nseries/
21. JDFE [Electronic resource] // Maxeler Technologies. Government – Resource access mode: https://www.maxeler.com/products/jdfe/
22. Диброва М. А., Коган А. В., Воробьева А. Л. Способ формирования множества путей в сетевых центрах данных //Вісник НТУУ" КПІ". Інформатика, управління та обчислювальна техніка. – 2015. – №. 63.
23. Wang T. et al. Rethinking the data center networking: Architecture, network protocols, and resource sharing //IEEE access. – 2014. – Т. 2. – С. 1481-1496.
24. Jain N. et al. Predicting the performance impact of different fat-tree configurations //Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. – 2017. – С. 1-13.
25. Kim J. et al. Cost-efficient dragonfly topology for large-scale systems //IEEE micro. – 2009. – Т. 29. – №. 1. – С. 33-40.
26. Teh M. Y. et al. Design space exploration of the dragonfly topology //International Conference on High Performance Computing. – Springer, Cham, 2017. – С. 57-74.