Journal of Quantum Information Science, 2016, 6, 105-120
Published Online June 2016 in SciRes. http://www.scirp.org/journal/jqis
http://dx.doi.org/10.4236/jqis.2016.62010

Interactive Quantum Development Environment (IQDE)

Nikolay Raychev

Varna University of Management, Varna, Bulgaria
Email: nikolay.raychev@vumk.eu

Received 12 May 2016; accepted 25 June 2016; published 29 June 2016

Abstract

This report presents a second version of the Interactive Quantum Development Environment (IQDE), virtualized parallel simulation platform for optimized testing of quantum software. IQDE is an interactive quantum simulator intended for implementation of a classical computer that can simulate numerous controlled and time-dependent operations. The research presents different relations between the operations that can be typically simulated. The virtualized simulation platform carries out numerous single-node and multi-node optimizations, including vectorization, parallelization, cache sharing, as well as overlapping of the computations with the communication. A common strategy for modeling for shared memory is implemented, as well as realistic parallel simulation with cluster management of the parallelization. A detailed analysis of the implementation is performed in order to be demonstrated that the simulator achieves good operation and high efficiency of the hardware, which is only limited by the available memory and the bandwidth of the machine.

Keywords

Quantum Operators, Phase Space, Quantum Circuit

1. Introduction

The use of a classic computer for simulation of the quantum system is important for the better understanding of its behavior. Such simulators may be used for the validation of the complexity of the new quantum algorithms in order to examine the quantum circuits, which can hardly be characterized analytically, or to examine the functioning of the circuits in the presence of noise.

Using a simulator with optimal parallelism, the effects of the parameters on the functioning of the algorithm can be studied. This in turn helps to minimize the quantity of quantum sources, which are required for limiting the errors below a given value. Alternatively, different model of noises can be applied, in order to determine the minimum required times for non-coherence for a given level of accuracy.

There are a large number of techniques for effective simulation of specific quantum circuits [1]-[5], while the simulations of generic quantum circuits of classic computer are very inefficient because of the exponential fraction [6]-[9]. More specifically, the fundamental challenge is that the size of the state, or the number of quantum amplitudes, grows exponentially with the number of the qubits. In case of n qubits, the size of the state is 2^n complex amplitudes, or 2^{n+4} bits. In this way the capacity of the memory of the classical system imposes an upper limit of the size of the simulation. In addition, the size of the quantum circuit (the number of its operators) may give a result of significant requirements for the time of functioning of the classical system.

The effect of these challenges may be withdrawn using highly operational distributive computation. The currently existing quantum simulators can function on one central processor [2]. Similar simulations are limited to approximately 30 qubits, due to the limited capacity of the memory and the bandwidth of a single node. Several distributive quantum simulators are developed, which can simulate more qubits, in comparison with a single node, because of the large memory capacity of the system.

The quantum simulator is a parallel computing platform which offers functional features for design and execution of quantum chains. The aim of this simulator is to give an opportunity for implementation, testing and visualization of quantum algorithms. The application offers a possibility for implementation and execution of flexible quantum algorithms or quantum schemes through visually oriented representation. Based on the principles of the quantum mechanics, the quantum simulator provides application of quantum registers. The major operations for manipulation of the registers like the Hadamard gate or the C-NOT gate are accessible through user-friendly interface. The measurement can be performed over single qubits or over the whole size of the register. Besides its application as quantum computing device, the quantum simulator is also capable of simulating the evolution of random hamiltonians including ones, which are time independent. This is achieved through numerical integration of the Schrodinger equation. The calculation of the evolution time is possible thanks to exact diagonalization of the time independent hamiltonians. The upper part of the interface contains the 36 embedded in the simulator quantum gates. The lower left part of the interface contains the matrix selected operators of the scheme. Following is the representation of the state after the marked operator (every cell of the matrix represents one classic state which represents the value of the amplitude). The lower right part of the interface represents through a matrix the probability of the end state. Within this chain the so-called "Review" gates can be included as well and they represent the percentage values of the temporary probabilistic distribution. They also indicate the probability that the qubit/the line could be in the "1" state if the appropriate measurement is done at the specific moment. Developed interactive environment for implementation and simulation of quantum algorithms has the following features:

- Time evolution uses diagonalization of fourth order, including optimization of partial Hamiltonians;
- Possibility for emulation of random quantum algorithms;
- Integrated de-coherency for realistic quantum calculations;
- Integrated formalism for density operator;
- Possibility for realization of the Shore factoring algorithm and the search Grouver algorithm.

The parallel computing platform adds up a lot to the dynamic nature of IQDE. When operating with thousands of virtual processes, many new challenges arise, such as the issue—how is it supposed to interact with any and all of them or with different subgroups of them? Where will be placed the debugger, and if we decide where to situate it, how will the other machines interact with it? How to keep the dynamic nature of the simulator in such an environment? Because if we want to give up from the interactivity and dynamic nature, IQDE may also function better only with the aid of sequential processing implemented with Python. These are some quite difficult issues and they are calling for new creative approaches for interactivity in parallel environments. Can we achieve them without changing the foundations of the simulator? The future studies will show.

While the aggregate capacity of a computing system is fixed, the time of the quantum simulation may be further improved. Here we describe the implementation of *IQDE* and the optimization, which is required for achievement of excellent performance and efficiency of the hardware. By using the maximum available allocation, we simulate quantum circuits of up to 8 qubits. For a 8-qubit system, when no communication is required, the operations controlled by the operator with single or two qubits are determined by the bandwidth of the memory and will take 0.43 and 0.21 seconds, respectively. The cache sharing optimization results in an additional ≈2.56× reduction of the operation of these two operators. When communication is required, their time for functioning increases 10 times, which is proportional to the memory to bandwidth ratio. Finally, using distributive simulation, we simulate 8-qubit transformation.

Chapter 2 briefly reviews the operations of the operators with single and two qubits. Chapter 3 describes the single- or multiple-nodes implementation of both operations in *IQDE*. Chapter 4 discusses the architectural and algorithmic optimizations, while Chapter 5 presents the experimental results and detailed analysis of the functioning of *IQDE*. Future recommendations about how to improve the performance of the simulator are described in Chapter 6.

2. Environment

The current version of *IQDE* propagates pure and mixed states. And therefore, we operate on the density matrix $2^N \times 2^N$. This approach concludes the simulation of mixed states when studying the noise effects and operator errors. This is due to the fact that there are reconstruction methods for the density matrix from much iteration of simulated pure states [10]-[15].

Work with Complex Numbers

The common state of the quantum calculation consists of complex numbers designated as amplitudes. To represent the amplitudes one can use the following square diagram, which represents in a visual form the different components of the complex number.

The real part is the X shift of the arrow, the Y shift is the imaginary part, the size is the length of the arrow and the phase is related to the direction of the arrow. Because the square size of the amplitude is also an important value (it describes the probability of a state), it is represented through the relationship of the filled in part of the square around the arrow to the whole size of the square (width = height = 2). Representation of the complex numbers as arrows is useful practice. These rotating arrows are equivalent of the complex numbers. After all the summing is as easy as placing the rotation of the arrows from end to end.

The fact that these arrows were equivalent to complex numbers that represent clearly the size and the phase also facilitates the visualization of the multiplication, because the influence of the phases is independent from the effect on the sizes. To multiply two complex numbers one must sum their phases and multiply their sizes.

One should pay attention to the fact how the matrix multiplication mixes the two input amplitudes of the state with the two output amplitudes of the state (the endmost blue values on the right) through the action of the matrix (the yelow values in the middle). Every input may accelerate or couterreact to the influence of other input data because of differences in the phase. This ability for creation of disturbances may be seen as something bad, but in fact it is what gives the quantum computer power coparated to the probabilistic or clasic computers operating without information for the phase.

In the quantum computations the quantitative values of the angular moments are calculated with differential operators $\left\{\hat{L}_x, \hat{L}_y, \hat{L}_z\right\}$, e.g.

The angular distance is $p_x \mapsto -i\hbar\frac{\partial}{\partial x}$, where the differential operator: $i\hbar\frac{\partial}{\partial t}$ represents the energy

$$L_x \to \hat{L}_x = -i\hbar\left(y\frac{\partial}{\partial z} - z\frac{\partial}{\partial y}\right),$$

$$L_y \to \hat{L}_y = -i\hbar\left(x\frac{\partial}{\partial z} - z\frac{\partial}{\partial x}\right),$$

$$\text{and}\quad L_z \to \hat{L}_z = -i\hbar\left(x\frac{\partial}{\partial y} - y\frac{\partial}{\partial x}\right).$$

The switches are $\left\{\hat{L}_x, \hat{L}_y, \hat{L}_z\right\}$, e.g.,

$$\left[\hat{L}_x,\hat{L}_y\right]=\left[-i\hbar\left(y\frac{\partial}{\partial z}-z\frac{\partial}{\partial y}\right)\cdot -i\hbar\left(x\frac{\partial}{\partial z}-z\frac{\partial}{\partial x}\right)-i\hbar\left(x\frac{\partial}{\partial z}-z\frac{\partial}{\partial x}\right)\cdot -i\hbar\left(y\frac{\partial}{\partial z}-z\frac{\partial}{\partial y}\right)\right]$$
$$=-\hbar^2\left[\left(y\frac{\partial}{\partial z}-z\frac{\partial}{\partial y}\right)\cdot\left(x\frac{\partial}{\partial z}-z\frac{\partial}{\partial x}\right)-\left(x\frac{\partial}{\partial z}-z\frac{\partial}{\partial x}\right)\cdot\left(y\frac{\partial}{\partial z}-z\frac{\partial}{\partial y}\right)\right]$$
$$=-\hbar^2\left[y\frac{\partial}{\partial z}\left(x\frac{\partial}{\partial z}\right)-y\frac{\partial}{\partial z}\left(z\frac{\partial}{\partial x}\right)-z\frac{\partial}{\partial y}\left(x\frac{\partial}{\partial z}\right)+z\frac{\partial}{\partial y}\left(z\frac{\partial}{\partial x}\right)-x\frac{\partial}{\partial z}\left(y\frac{\partial}{\partial z}\right)+x\frac{\partial}{\partial z}\left(z\frac{\partial}{\partial y}\right)+z\frac{\partial}{\partial x}\left(y\frac{\partial}{\partial z}\right)-z\frac{\partial}{\partial x}\left(z\frac{\partial}{\partial y}\right)\right]$$
$$=-\hbar^2\left[xy\frac{\partial^2}{\partial z^2}-y\frac{\partial}{\partial x}-yz\frac{\partial^2}{\partial x\partial z}-xz\frac{\partial^2}{\partial y\partial z}+z^2\frac{\partial^2}{\partial x\partial y}-xy\frac{\partial^2}{\partial z^2}+x\frac{\partial}{\partial y}+xz\frac{\partial^2}{\partial y\partial z}+yz\frac{\partial^2}{\partial x\partial z}-z^2\frac{\partial^2}{\partial x\partial y}\right]$$
$$=-\hbar^2\left[-y\frac{\partial}{\partial x}+x\frac{\partial}{\partial y}\right]=-\hbar^2 L_z.$$

The rotations of matrices have the same commutator algebra.

$$R_x(\phi)=\begin{bmatrix}1&0&0\\0&\cos\phi&-\sin\phi\\0&\sin\phi&\cos\phi\end{bmatrix},\ R_y(\psi)=\begin{bmatrix}\cos\psi&0&\sin\psi\\0&1&0\\-\sin\psi&0&\cos\psi\end{bmatrix},\ R_z(\theta)=\begin{bmatrix}\cos\theta&-\sin\theta&0\\\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix}.$$

The Taylor series expand the matrix elements of the first line:

$$\begin{bmatrix}0&0&0\\0&0&-1\\0&1&0\end{bmatrix},\ \begin{bmatrix}0&0&1\\0&0&0\\-1&0&0\end{bmatrix},\ \begin{bmatrix}0&-1&0\\1&0&0\\0&0&0\end{bmatrix}.$$

The implementation of commutators is formed by the rotation of matrices under commutation into the group. The factor *i* corresponds to the indices of the angular moment, *e.g.*, $\left[R_x,R_y\right]=-R_z$.

We focus on the implementation of general operators with one qubit, as well as with controlled operators with two and three qubits (including, *controlled*-NOT operator, which are known as universal (14). The operation of quantum single-qubit operator *k* can be represented with a unitary transformation:

$$U=I\otimes I\otimes\cdots\otimes Q\otimes\cdots\otimes I\otimes I \tag{1}$$

where Q is 2 × 2 unitary matrix,

$$Q=\begin{pmatrix}q_{11}&q_{12}\\q_{21}&q_{22}\end{pmatrix}$$

But one does not have to construct the entire *U*, to carry out the transformation *Q* directly on the vector of the state, as shown in **Figures 1-3** using an example for two qubits. The application of the operations of the single-qubit operator on qubit 0 is equivalent to the application of *Q* to each pair of amplitudes, whose indices have 0 and 1 in the first bit, while all other bits remain the same. In a similar way with the application of an operator with single-qubit to qubit 1 is applied *Q* to each pair of amplitudes, whose indices differ in their second bit. Generally speaking, the implementation of a single-qubit operator on qubit *k* of *n*-qubit quantum register is applied for *Q* to pairs of amplitudes, whose indices differ in *κ*-th bits of their binary index:

$$\begin{aligned}a_{*\ldots*}0_{k*\ldots*}&=q_{11}\cdot a_{*\ldots*}0_{k*\ldots*}+q_{12}\cdot a_{*\ldots*}1_{k*\ldots*}\\a_{*\ldots*}0_{k*\ldots*}&=q_{21}\cdot a_{*\ldots*}0_{k*\ldots*}+q_{22}\cdot a_{*\ldots*}1_{k*\ldots*}\end{aligned} \tag{2}$$

When the vector state is dense and stored sequentially in the memory, the difference between $\alpha_{*\ldots*}0_{k*\ldots*}$ and $\alpha_{*\ldots*}1_{k*\ldots*}$ is 2^{κ}. The applied operations of the operator of high-order qubits result in large steps and present a challenge for the single and distributive implementations, which are described in the following chapters.

A generalized two-qubit *controlled*-*Q* operator, with a control qubit c and target qubit *t*, operates as follows: If *c* is set to 1, *Q* applies to *t*; otherwise *t* remains unchanged:

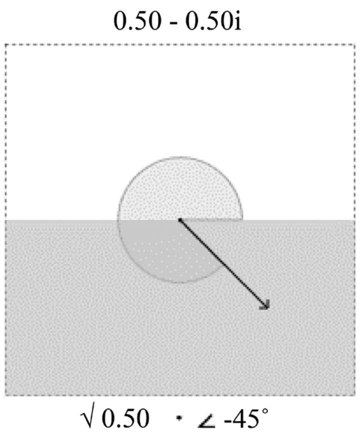

Figure 1. Reresentation of complex numbers.

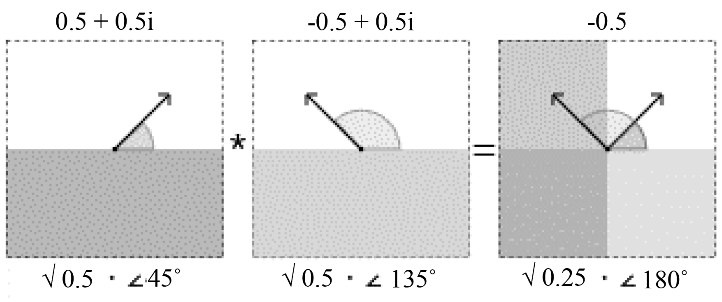

Figure 2. Summing complex numbers.

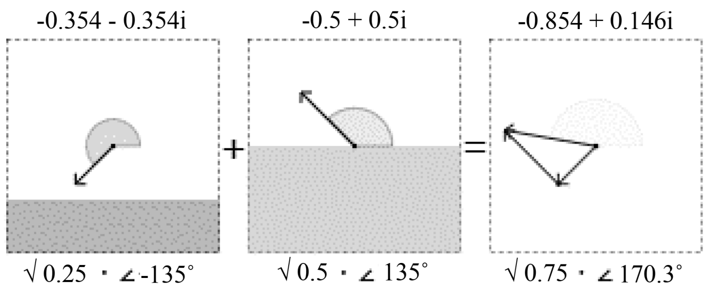

Figure 3. Multiply complex numbers.

$$\begin{aligned} a_{*1c*}0_{t*\ldots*} &= q_{11}\cdot a_{*1c*}0_{t*\ldots*} + q_{12}\cdot a_{*1c*}1_{t*\ldots*} \\ a_{*1c*}0_{t*\ldots*} &= q_{21}\cdot a_{*1c*}0_{t*\ldots*} + q_{22}\cdot a_{*1c*}1_{t*\ldots*} \end{aligned} \tag{3}$$

3. Implementation

This chapter describes the application of *IQDE* of single or controlled-*Q* operators for single and multiple nodes.

3.1. Quantum Chains

The lines in a quantum chain correspond to qubits and they store the state of the system. They are called qubits instead of bits, because the system may be placed in superpositions of various possible clasic states. System with *n* bits has 2^n possible classic states, but system with *n* qubits has a sequence of quantum states corresponding to different amplitude weights (superpositions) of 2^n classic states. The space of the state of *n* bits may be visualized through the angles of *n*-dimensional cube, while the space of the state of *n* qubits may be visualized in an abstract form as points in the volume of $4^n - 1$-dimensional sphere, like the Bloch sphere. For instance if there are two lines then the quantum system is in pure classic state as [*OnOn*:1, *OnOff*:0, *OffOn*:0, *OffOff*:0], where the two line are definitely *On* or in mixed state as [*OnOn*:0, *OnOff*: $\frac{1}{\sqrt{2}}$, *OffOn*: $\frac{i}{\sqrt{2}}$, *OffOff*:0], where only one line is *On*, but it is not clear which one exactly. (These states may be represented in a more compact form in bracket notation, namely as $|OnOn\rangle$ and $\frac{1}{\sqrt{2}}\left(i|OffOn\rangle|OnOff\rangle\right)$. The phases of the states are of importance when some disturbance is observed. The inputs of the quantum chain correspond to the operations executed on the state of the lines. Everyone is equivalent to multiplication of the state of unitary matrix. Let's investigate a simple example of a quantum chain. The simplest gate is the NOT gate which switches the On/Off states of a single line. The application of the NOT gate corresponds to the multiplication of the state of the system through the unitary matrix $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. The input is in an *On* state which effectively selects the first column of the matrix as an output in *Off* state. If the gate was in *Off* state, then the second column would have been chosen and the output would have been in *On* state. The NOT gate may be executed in non-quantum computers and therefore it is not very much interesting. Now let's investigate simple quantum gate without classic equivalent. A Hadamar gate is introduced, corresponding to the $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ matrix and used for creation of single combinations. Figure 4 illustrates the initial *On* state and *Off* state, mixed and separated through Hadamar gates.

The Hadamard gate is interesting because it transforms the two *On* and *Off* states in a 50% mixed *On/Off* state, but when applied second time it revokes its own effect and restores its initial state. This would have been impossible with the classic or probabilistic computers, because the 50%/50% gate should give one and the same result in both cases. If the above picture is analysed with more scrutiny, one can see how exactly the process is accomplished: the original value is coded in the phases of the 50%/50% state (the phases are synchronized if the original state was *On* and are in oposition if it was *Off*).

3.2. More Lines

Things become a little bit more complicated if there are more lines and if the Hadamar gate is applied to only one line in a chain with *n* lines. At the moment 2 × 2 matrix is available, but the operations on the chain must be represented with $2^n \times 2^n$ matrixes. It should be noted that the state of the line which will be affected is not limited to 2 from 2^n possible states of all lines (possible global states). Every possible global state includes the state of the line which shall be affected. If one line must be touched, every global state should be touched. The main idea of what should be done is to repeat the 2 × 2 matrix for every combination of states of the other lines. If there are two lines then one copy is necessary for the case when the other line is *On* and one copy for the case when the other line is *Off*. The next picture shows chain with two lines that applies the Hadamar gate over one line and then over the other. One should note how the repeating records of the 2 × 2 matrix are positioned differently depending on the affected line:

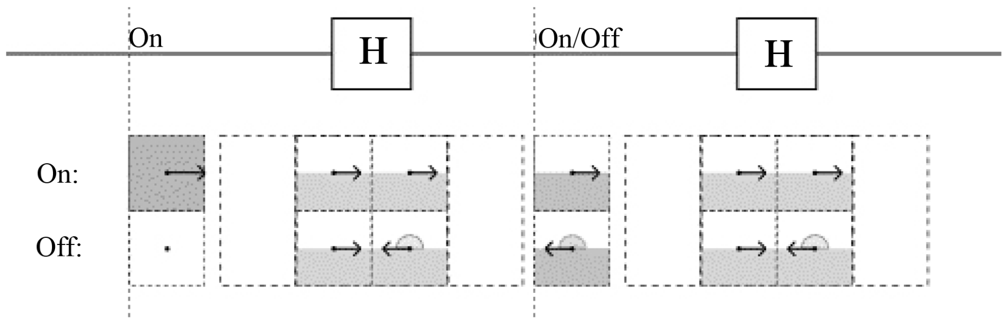

Figure 4. Hadamard gate is used for creating the uniform mixes. The entry states are mixed and unmixed.

An interesting feature of the Hadamar gates which can be seen on **Figure 5** is that their application to a number of lines has the same mixed effect as in the case with one line. Every pure input state will be the same mixed output state with the original state, coded in phases and the application of the gates again will return the mixed state to its initial pure state.

3.3. Notation of Some of the Implemented Gates (Figure 6)

Anti-Control: The linked operators are applied only when the control Qubit is 0. The Anti-Control operator operates as the Control operator except for the condition: 0 instead of 1. The linked operations are performed only in the parts of the superposition where the control Qubit is 0;

•, Control: The linked operators are applied only when the Control Qubit is 1. The Control operator is in fact rather modifier, which places condition to the other operators that they shall be executed only when the control Qubit is 1. When the control Qubit is in superposition of 1 and 0, the other operations are applied only in the parts of the superposition where the control Qubit is 1;

↻, Pauli rotation: (0, 0, 0.25): Clockwise rotating phase gate, multiplies the ON phase by –i (without affecting the OFF state). This rotating phase gate is one of the four square roots of the Pauli Z gate. The gate is inverted to the counter rotating phase gate.;

H(t): Evoluting Hadamard gate. Smoothly interpolates from no-op to Hadamard gate and back through configured time interval. Continuous rotation around XZ basis of the Bloch sphere;

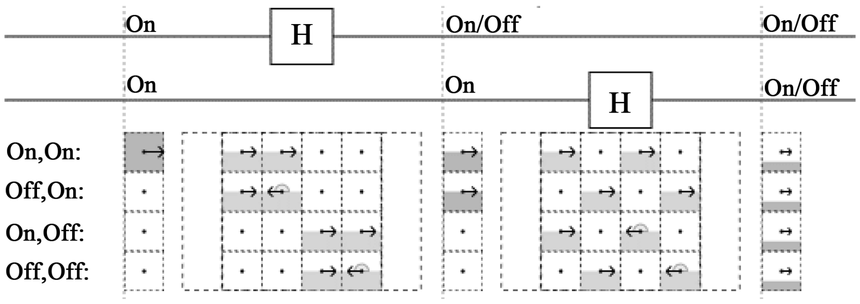

Figure 5. Hadamard gate applied to two lines.

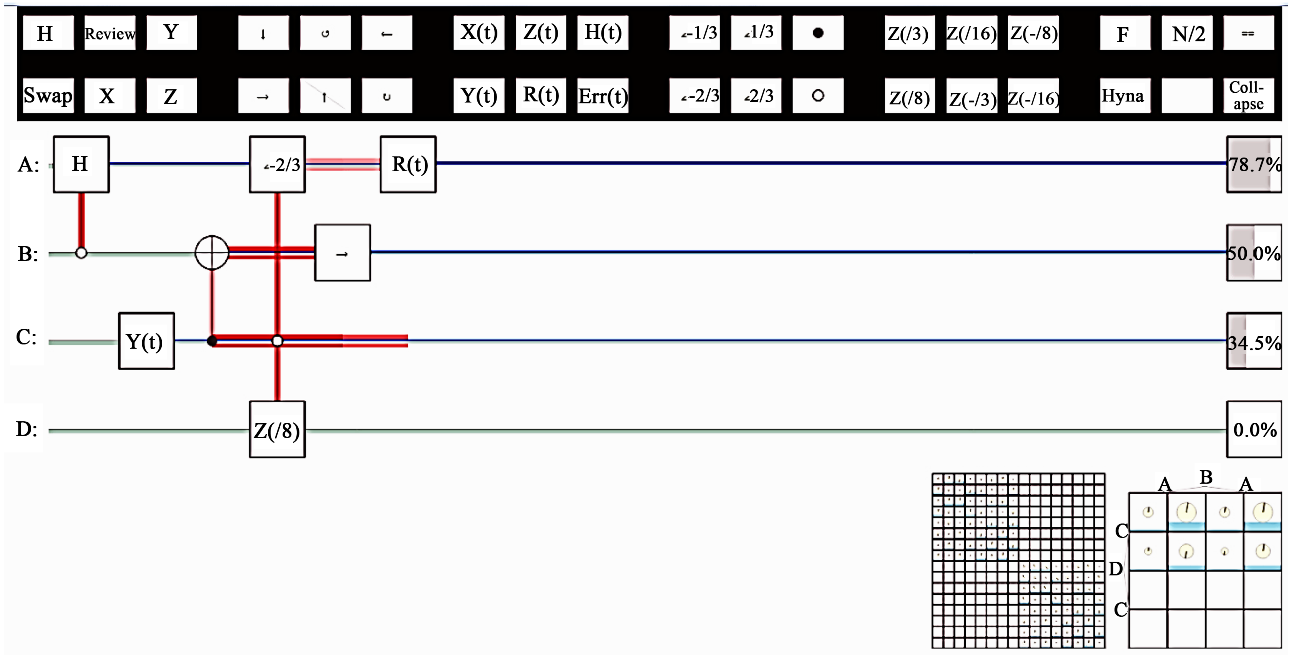

Figure 6. Interactive environment for quantum simulations.

Err(t): Evoluting Noise gate. Creates random noise effect;

R(t): Evoluting rotating gate. Rotating gate with increasing angle of rotation, cycles through the time interval;

X(t): Evoluting X gate. Smoothly interpolates from no-on to Pauli X gate and back through configured time interval. Continuous rotation around X basis of the Bloch sphere;

Y(t): Evoluting Y gate. Smoothly interpolates from no-on to Pauli Y gate and back through configured time interval. Continuous rotation around Y basis of the Bloch sphere;

Z(t): Evoluting Z gate. Smoothly interpolates from no-on to Pauli Z gate and back through time interval. Phase gate with increasing angle of rotation, cycles through time interval. Continuous rotation around Z basis of the Bloch sphere;

||: Specific operation based on rotation. With x = 0 and y = 0 returns Phase rotation, otherwise returns Pauli rotation around <x = 0, y/x, z/n>;

º Phase: Rotates the phase of the qubit ON state on the configured/predetermined levels, leaving only the OFF state. Standard Pauli Z gate, corresponds to Z(180˚);

∠n⁄m: Rotating gate tuned for transition initialization of the OFF qubit to the n/m probability to become ON. Equivalent to R(acos(√(**n⁄m**))).);

The experiments carried out with the developed quantum simulator IQDE show that the Hadamard and 45˚ phase gates can approximate each single qubit operation. This is in fact easily provable, since each single qubit operation corresponds to a rotation in 3D. The Hadamard operation is a rotation of 180˚ around the XZ axis, which is equivalent to 90˚ rotation around X, then around Z, then around the X axes. The phase 45˚ gate represents a 45˚ rotation around the Z axis. The combination of these two rotations allows the carrying out of all other rotations. It is possible to be obtained an approximation on each target rotation, although to achieve this it may be necessary to make a composition of a longer series from the described two rotations.

On second place, the quantum operations must be factored in single qubits operations and CNOT gates. Each quantum operation corresponds to a rotation of an unitary matrix whose action is similar to the extraction of scalar values from a matrix, swapping and adding rows upon inversion with Gaussian[1] elimination. These simulations may prove to be with exponentially many factors because these matrices are with exponential size, but can always be factored.

The factoring creates an exponential increase of the number of operations, but the same exponential increase occurs also at the classical circuits. Classically, these are 2^{2^N} operations, which receive the N bit at the input and return one bit at the output, and there are only $\approx G^G$ ways to link together the G NAND gates. This exponential increase would have been even bigger at quantum computations.

Each quantum operation can be approximately simulated by 45˚ phase, CNOT and Hadamard gate.

Measuring and pairing can be classically simulated through CNOT operations, creation of superposition and interference can be simulated through Hadamard operations, while the complex quantum interference can be simulated through phase gates.

With the help of the 36 quantum gates, which were defined, can be created arrays, which to be applied consistently to the quantum system. Such an array of quantum gates is also called a quantum circuit. For example, if a 2-qubit system is given and a NOT gate and after that a CNOT gate is applied, then the resulting quantum circuit is {NOT, CNOT}. Therefore, the quantum gates are the building elements of the quantum circuits. With this basis it is possible to be initiated the design of effective quantum circuits for performance of quantum walk in one and two dimensions.

3.4. New Functionalities-3-Qubit Dense Matrices

There are 2 possibilities to generate 3-qubit dense matrices by 2 qubit circuit decompositions, or by decomposition $H_{2^3} = C^2 \otimes C^2 \otimes C^2$ into four 2-dimensional subspaces or into two 4-dimensional such.
$8 = 2 \oplus 2 \oplus 2 \oplus 2$

First we will define a 2-dimensional subspace

$$\Delta_{00} = span\left\{e_0 \otimes e_0 \otimes e_0, \otimes e_1 \otimes e_1 \otimes e_1\right\} \tag{4}$$

and for any two binary μ and υ we define

[1]The Gaussian method is an algorithm for solving linear equations systems. It represents a sequence of operations, which are performed on the associated matrix of the coefficients. This method is used to detect the rank of a matrix, in order to calculate the determinant and the inverse value of a reversible square matrix.

$$\Delta_{\mu\upsilon} = s\left(\mathbb{I} \otimes S^{\mu} \otimes S^{\upsilon}\right)\Delta_{00} \tag{5}$$

Then we find:

$$\begin{aligned} \Delta_{01} &= span\{e_0 \otimes e_0 \otimes e_1, \otimes e_1 \otimes e_1 \otimes e_0\}, \\ \Delta_{10} &= span\{e_0 \otimes e_1 \otimes e_0, \otimes e_1 \otimes e_0 \otimes e_1\}, \\ \Delta_{11} &= span\{e_0 \otimes e_1 \otimes e_1, \otimes e_1 \otimes e_0 \otimes e_0\}, \end{aligned} \tag{6}$$

and

$$H_{2^3} = \Delta_{00} \oplus \Delta_{01} \oplus \Delta_{10} \oplus \Delta_{11} \tag{7}$$

Then we create 3-qubit density matrix in the following type

$$\rho = \rho_{00} \oplus \rho_{01} \oplus \rho_{10} \oplus \rho_{11} \tag{8}$$

where each $\rho_{\mu\upsilon}$ is supported by $\Delta_{\mu\upsilon}$, Therefore it is necessary:

$$\rho_{\mu\upsilon} = \sum\nolimits_{i,j=0}^{1} x_{ij}^{\mu\upsilon} e_{ij} \otimes S^{\mu}{}_{e_{ij}} S^{\mu*} \otimes S^{\upsilon}{}_{e_{ij}} S^{\upsilon*} \tag{9}$$

The normalization $T_{r\rho}$ is equivalent to:

$$T_{\mathrm{r}} = \left(x^{00} + x^{01} + x^{10} + x^{11}\right) = 1 \tag{10}$$

Therefore we obtain the following block matrix:

$$\rho = \left(\begin{array}{cc|cc||cc|cc} x^{(00)}_{00} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x^{(00)}_{01} \\ \cdot & x^{(01)}_{00} & \cdot & \cdot & \cdot & \cdot & x^{(01)}_{01} & \cdot \\ \hline \cdot & \cdot & x^{(10)}_{00} & \cdot & \cdot & x^{(10)}_{01} & \cdot & \cdot \\ \cdot & \cdot & \cdot & x^{(11)}_{00} & x^{(11)}_{01} & \cdot & \cdot & \cdot \\ \hline\hline \cdot & \cdot & \cdot & x^{(11)}_{10} & x^{(11)}_{11} & \cdot & \cdot & \cdot \\ \cdot & \cdot & x^{(10)}_{10} & \cdot & \cdot & x^{(10)}_{11} & \cdot & \cdot \\ \hline \cdot & x^{(01)}_{10} & \cdot & \cdot & \cdot & \cdot & x^{(01)}_{11} & \cdot \\ x^{(00)}_{10} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x^{(00)}_{11} \end{array}\right), \tag{11}$$

where the vertical and horizontal lines represent the splitting into blocks corresponding to a tensor product with a structure C^2 ⊗ C^2 ⊗ C^2. The Double lines represent the splitting into four blocks, corresponding to C^2 ⊗ (C^2 ⊗ C^2) and the single lines represent the splitting within each 4 × 4 block, corresponding to the second tensor product (C^2 ⊗ C^2). Then we perform partial transposition. There are three independent transformations.

$$\tau_{ab} = \mathbb{I} \otimes \tau^{a} \otimes \tau^{b} \tag{12}$$

with binary indices *a* and *b*

$$\begin{aligned} \tau_{01} &= \mathbb{I} \otimes \mathbb{I} \otimes \tau, \\ \tau_{10} &= \mathbb{I} \otimes \tau \otimes \mathbb{I}, \\ \tau_{11} &= \mathbb{I} \otimes \tau \otimes \tau \end{aligned} \tag{13}$$

From this equation it is easy to see that $\tau_{ab\rho}$ has the same circular structure as the original ρ defined with new 2 × 2 matrices $y^{(\mu\upsilon)[ab]}$, this is

$$\tau_{ab\rho} = \sum\nolimits_{\mu\upsilon=0}^{1} \sum\nolimits_{i,j=0}^{1} y_{ij}^{(\mu\upsilon)[ab]} e_{ij} \otimes S^{\mu}{}_{e_{ij}} S^{\mu*} \otimes S^{\upsilon}{}_{e_{ij}} S^{\upsilon*} \tag{14}$$

3.5. Distributed Implementation

In our distributed implementation there is a vector with a state of 2^n amplitudes (2^{n+4} bits), as each node stores a local state of 2^{n-p} amplitudes. Let $m = n - p$. Of course, 2^{m+4} must be less than the capacity of the memory of the node.

When the operation Q of a single-qubit operator is on qubit κ, if $\kappa < m$, the operation is contained completely in the node. When $\kappa \geq m$, the first and second elements of the pair are located on two different nodes and communication is required. Note that the distance between both communicating processors is the virtual topology $2^{\kappa-m}$. We implement the communication scheme described in (37). Our improvements to this scheme are described in Chapter 4. When the local state of the vector is of 2^m complex amplitudes, each node reserves 2^{m-1} words extra memory as temporary storage. Any vector with local state can logically be divided into two halves. Both nodes perform exchange of these two halves; P_i sends the first half to P_j, while P_j sends its second half to $P_{i.}$. Each node puts the resulting half in their own temporary storage. Then, P_i applies Q to the first half and the temporary storage (which contains the first half of P_j), while P_j applies Q to the second half and the temporary storage (which contains the second half of P_j). This as a result causes the update of the second half of P_j from P_i, and the update of the first half of P_i from P_j. This is followed by another exchange of halves, where P_i sends the updated half of P_j back to P_j, while P_j sends the updated half of P_i to P_j. This completes the distributed update of the state. The advantage of this approach is that it distributes the work equally between pairs of nodes.

The distributed implementation of a controlled operation of the operator with controlled qubit c and targeted qubit t is more active. If $t < m$, there is no communication, while at $t \geq m$, communication is needed. In addition, for each of these two cases, we use different kernels, depending on whether $c < m$, or $c \geq m$. Therefore, there are a total of four cases for a controlled qubit operator, in comparison with only two for a single-qubit operator. We skip over more of the details for the sake of brevity.

4. Architectural and Algorithmic Optimization

In this chapter we describe several optimizations with single node and with multiple nodes, which we apply, to achieve greater functionality of IQDE.

The software architecture of IQDE is presented on **Figure 7**. The top layer of the virtual platform is a user code after the implementation phase. The core of the system is responsible for the creation and destruction of

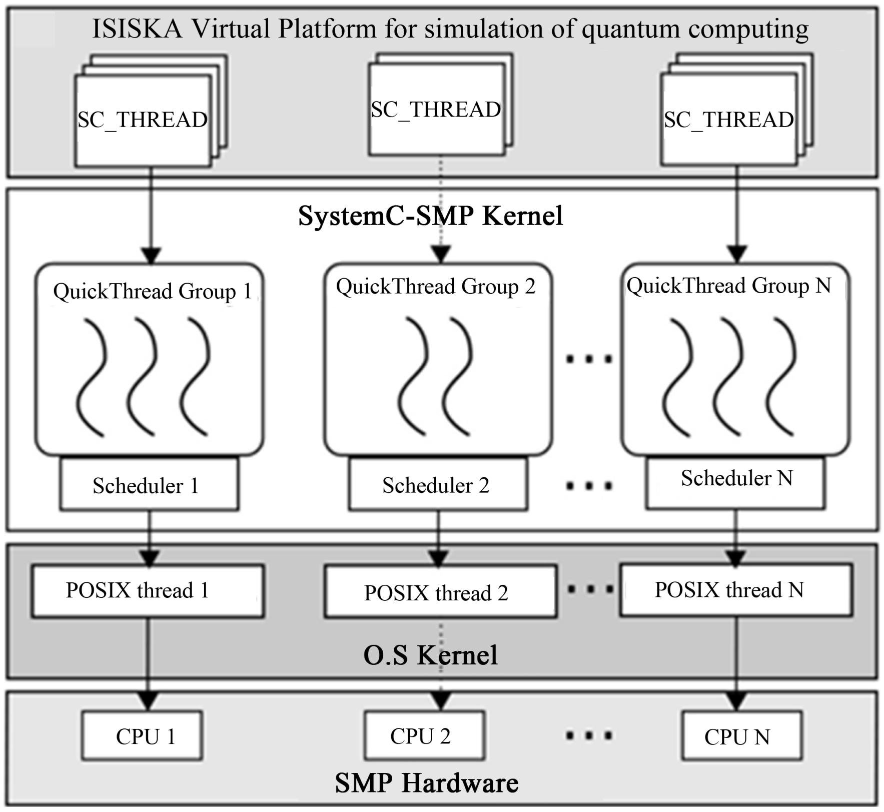

Figure 7. IQDE Software architecture.

simulation objects. It realize the sharing of the objects and makes them visible for all local distributors. And the local distributors are responsible for the implementation of all threads, which are executed by the same physical processor. All threads are realized as Fast Threads. The core of the OS is a system ensuring the realization of API for the POSIX-threads. Each local distributor of the system is executed in a POSIX-thread (pthread) of the OS. With this software architecture it is possible to be used CPU-dependent functions, provided by the OS and so each thread to be associated with a physical processor. The hardware of the multiprocessor systems implements sharing of memory between several physical processors.

The proposed new approach for time modeling ensures the realization of virtual distribution of the shared memory. This requires the use of the principles for parallel simulation of individual events and the use of distributed management of the temporary instead of the global simulation. In order to use the maximum efficiency of the parallel behavior is presented also a new approach for parallel simulation at Kernel level. On a dual-core workstation the system gets 1.9 acceleration in comparison with the previous version. This is very close to the theoretical upper limit. The used by IQDE algorithmic model of the simulation decreases the possibilities for locking with the aid of small critical sections. For this reason, this approach is a truly scalable and can be used on workstations containing more cores.

4.1. Structural Optimization through Batch Methods

The data transfer from the processors to the application proved to be a weak spot in the first version of the platform. Each communication takes around ten milliseconds. When transferring dozens of different data structures to a distance from the processors, this brings significant delays in the implementation of the entire process as it appears that the data processing is carried out more quickly.

It appears that the transfer of the data maintains in general one and the same speed, regardless of the read-out volumes. The slow pace of the communication is not a problem coming from the width of the bandwidth, this is latent problem. As a solution we have implemented the following optimization—we've united all parallel processed data in a large structure, which is transferred at once.

The performance of a single transfer of a huge amount of data from the processors has fixed the problems with the productivity, but introduced problems with the encoding. For example, when one qubit state needs to be divided into code for processing and code for interpretation/when the processing phase is followed by a qubit state for interpretation is created duplicate code/. More importantly it is difficult to be created an abstract model by which the results of all parallel processed structures of the individual quantum lines are indexed in the total communication pool. Algorithmically must be collected all these structures, then carefully to be separated and indexed the individual results.

Indexing of areas in a structured object of data. By unifying the fields of a particular object and potential further study of each field we are able to create a mapping between these areas and the indices of an array. At the next phase the mapping can be applied reversibly to create a modified version of the object using the values from a new array. The described logical model can be used to create an object structure around array processing method.

For the implementation of this logic scheme we need a method for extracting the values from one object. There are possible numerous options for the design of this method. For example: To be examined only the fields of the highest level, or to examine the entire tree by seeking objects leaves from a specific type?

Here is a part of the code of this method. We simply go round the switches and the elements of the fields of the arrays, until there are values added in the result.

At the second step is required an inversely proportional method. This method must have access to the original object, in order to ensure mapping with ratio one to one when indexing the fields and the array with the updated values. Then is created a modified version of the object with keys of the fields determined by the given values of the objects and values of the fields determined by an array.

At the third step is created also a method to manage the decomposition and recomposition around random batch functions:

Improvement in the Communication: Improvement of the communication scheme, which requires an extra 2^{m-1} words for temporary storage per node. The first version of this application proved to be non-productive. Take for example a node with 48 GB memory. In theory, can be used 32 GB for the state of the vector and the remaining 16 GB for temporary storage, in order to be simulated a system with 31 qubits. In practice, no application is able to use the entire memory of the machine without significant storage operation, because a portion of the memory is reserved for the OS, system software and other purposes. Therefore we can only simulate quantum system with 30 qubits in this case.

To reduce the requirements of the memory for temporary storage, we divide the distributed phase into several steps. In each step we exchange and reserve temporary storage only for a small portion of the state of the vector, as opposed to the entire half in the original approach. We apply operations of the operator only for this portion. Each step reuses the same temporary storage, which is dramatically reduced. As we continue with the previous example, we can use 8 GB of temporary storage instead of 16 GB. This requires 2 steps, but facilitates the simulation of 31 qubits-one qubit more than the original approach. As far as the amount of data exchanged at each step is sufficient to saturate the network bandwidth, the overall operation remains as in the original approach.

The total operation of the distributed application, T_{total}, is a function of the computing time, $T_{computation}$ and the time for communication $T_{communication}$. The first implementation of the quantum simulator carries out these phases separately, as $T_{total} = T_{computation} + T_{communication}$. When using the multistep approach, we overlap the communication with the computation in step i with the exchange of the state in steps $i - 1$ and $i + 2$. This results in $T_{total} = max\ (T_{computation}, T_{communication})$, thus partially hiding the upper limit of the communication.

4.2. Vectorization

The operations of both single and controlled operator have inner loop. This cycle is with parallel data: Each iteration performs the same operation on a different set of data. One of the most applied and energy efficient methods for using the parallelism at data level are via Single Instruction of Multiple Data (SIMD) implementation. In the implementation of SIMD, one instruction operates on numerous data elements simultaneously. This is usually applied by extending the scope of the registers and arithmetic logic units, allowing them to hold and operate with numerous data elements, respectively. The rest of the system remains unchanged.

The modern Intel processors maintain SIMD instructions, such as AVX2 (4), which can perform 4 operations with double precision simultaneously on 4 elements of the input registers. We trace every two iterations of the inner loop of up to 4 SIMD instructions; each iteration, which operates on a complex number consisting of real and imaginary parts shall be traced up to a pair of entries in the SIMD register. We have developed specialized code, using the essential characteristics, to implement effectively complex arithmetic using SIMD instructions.

4.3. Threads

The modern processors have multiple cores, and some have several threads per core. In order to achieve excellent operation, it is important to parallelize the workload in these threads. There are two levels of parallelization of the code in one above the inner loop, and the other above the outer loop. Note that the outer loop performs 2^{n-k-1}, iterations , and the inner loop performs 2^{κ} iterations, while a smaller κ results in more (less) external (internal) iterations of the cycle, while larger κ gives the opposite effect. For example, when $k = n - 4$, the outer loop carries out only eight iterations, which leaves some cores idle when the loop is parallelized on the central processor with more than eight threads. To select which loop to parallelize, we dynamically check the number of iterations and parallelize the level of the nest with the most work.

4.4. Cache Sharing through Operator Synthesis

The single and controlled qubit operations carry out small quantity of computations. Therefore, their operation is limited by the bandwidth of the memory, when the size of the state of the vector exceeds the Last Level Cache (LLC). The modern central processors have large LLC, tens of megabytes per socket. LLC has a much higher bandwidth than that of the memory (although, much smaller capacity), but by taking advantage of the high bandwidth of the LLC requires restructuring the algorithm so that the work set to be fitted in one LLC.

Through the use of synthesized operators we may block the computation in LLC. We assume that the LLC has a size of 2^{lc}. For a given quantum circuit, we identify groups of consecutive operators, where each operator operates on some qubit κ, $\kappa < lc$. We iterate above the blocks 2^{lc} amplitudes of the state of the vector (Line 1). Each of the synthesized operators is applied for this block (Lines 2-4), while the block remains in LLC and therefore may take advantage of the high bandwidth of the LLC.

5. Functioning

5.1. Smaller Restrictions on the Achievable Functioning

The intensive application for the current data, the quantum circuit simulator is limited by the bandwidth of the memory, when functioning on a single node, or from the network bandwidth, when functioning in distributed way. The limited memory is equal to the total required traffic of the memory divided into the stable bandwidth of the memory (*Bmem*), measured by the benchmark. The network limit is equal to the total quantity of network traffic divided by the stable network bandwidth (*Bnet*), measured from the OSU benchmark of the bandwidth. On our system, *Bmem* = 40 GB/s, while *Bnet* = 5.5 GB/s (2-way), per each socket. The closer the actual application to these two limits, the greater the efficiency of the hardware is.

Here, 2^{m+4} is the size of the vector of the state in bytes, while the additional factor of the two is due to the fact that the state of the vector is read and written, which doubles the quantity of traffic of the memory. As our system has 16 GB memory per socket, it can simulate at most 2 qubits in a socket, because the state of the vector occupies 8.5 GB memory. With bandwidth of the stream B_{mem} = 40 GB/c, the restriction of the memory per one socket is 0.43 seconds for 29-qubit quantum simulation. As the controlled qubit operator has access only to half of the state of the vector, its expected operation is 0.22 seconds, also when there is no communication. When communication is required, a pair of nodes carries out two exchanges of half of the state of another node. This results in network limit of $(2^{m+5})/B_{mem}$. For m = 29, the relevant network limit is 3.12 seconds, which is 7.3 times higher than the limit of the memory and is an effective ratio between the bandwidths of the memory and the network.

In the rest of this chapter we use these limits, to explain the results of our experiments.

5.2. Single Node Functioning

Our operation of single-qubit operator flows close to the border of the memory, regardless of which qubit the operator is applied to. While the value of the control qubit increases, the operation is improving, and starting from control qubit 10, it is approaching the limit of the memory. The reason for suboptimal functioning is the smaller number of qubits, and this is as follows. Recall that a control operator affects only the peak amplitudes of those with "c" bit set to 1. Thus, the patter of access to the memory have access only to the second half of each sector of the memory 2^c. These "holes" interrupt the stride and hardware preliminary storage, which is processed to store preliminary elements which have a permanent stride between themselves. The inability of the preliminary storage to recognize the stride and to show the deficiencies and latency of the cache, results in consumed bandwidth of the memory, due to the useless data at the preliminary storage. This results in increase of the operation.

The operation when mixing the optimization of a single node for the *Inverse Quantum Fourier Transform* (IQFT), as the number of qubits varies between 4 and 29. IQFT of n-qubit quantum register that is composed of n steps. At step i IQFT applies the operator of Hadamard as well as $n - i - 1$ operator with the controlled rotation to qubit i + 1, ⋯ n, respectively.

IQFT naturally gains from the mixing of the optimization of the operator: All stages from 0 to l_c may be mixed and therefore blocked in LLC. The results of the application of the optimization by mixing on IQFT are shown in **Figure 7**, for the sizes of the quantum register between 18 and 29 qubits.

For a given number qubits n the functioning in respect of the achieved bandwidth (GB/s), calculated as cumulated amount of traffic of the memory for all operators divided by the overall operation of IQFT. The closer the achieved bandwidth is to the bandwidth of 40 GB/s, the closer is the functioning of IQFT to the limit of the memory. Because on our system the size of LLC is 20 MB, l_c = 20, and in this way we can combine up to the 19th stage of IQFT, regardless of n.

The operation without fusion is ≈ 80 GB/s per operator, which is twice higher than the bandwidth of memory of 40 GB/s. This is due to the fact that for these problematic dimensions, the full quantum state naturally fits in LLC, and we are witnessing dramatic reduction in the operation to 40 Gb/s, which is expected from the bandwidth. The operation remains 40 Gb/s until reaching 2 qubits.

At operation with merging of the operator for 2 qubits we obtain 100 GB/s bandwidth of the memory per operator, even though the state can no longer be merged with LLC. In this way we see that the optimization of the merging increases the operation by almost 2.5 times. With the increasing of the number of the qubits, the operation decreases gradually, due to the fact that the number of stages which cannot be mixed increases. But even for 2 qubits, we achieve bandwidth of 70 GB/s, which is almost 2 times more.

5.3. Multi Node Operation

Strong Scaling: For 4 sockets we achieve almost linear acceleration of 4 times more per socket, when applying the operator for qubits 15 to 26, as there is no communication. Qubits 27 and 28, on the other hand, require communication between the sockets. This gives rise to 2 times more delay in comparison with one socket, whose functioning is limited by the memory. This is expected. When communication is required, the operation is limited by the bandwidth of the network, which is ≈ 7.2 times lower than the bandwidth of the memory. As a result, the 7.2 times reduction rejects the expected acceleration by 4 times, and results in a delay of 0.5 times (≈4/7.2).

We are observing almost linear acceleration for the sockets when no communication is required, which is followed by proportional reduction of the acceleration when the sockets must communicate. But unlike the case with 4 sockets, the larger number of nodes compensate for the reduction of the operation due to lower bandwidth of the network, and as a result we are witnessing accelerations of the sockets from 2.4 to 9.3 times.

For sockets 2048 we observe superlinear acceleration for qubits 5, 6 and 7. This is also expected. With the increase of the number of sockets, the size of the state per socket decreases. And in particular, on 2048 sockets the state occupies only 4.2 MB memory per socket (=$2^{29+4-11}$), and thus merges in LLC. As described in Chapter 5.3, LLC limited operation is 2× faster than the limit of the operation of the memory of a single node, which results in proportional operation and increase of the volume.

Weak Scaling: At operation of a single-qubit operator on multiple nodes we fix the local vector of the state in order to use the maximum amount of memory available on a socket. With the increase of the number of the qubits, we use also more sockets, and as a result the size of the local vector of the state on a socket remains one and the same.

The operations of the operator applied to qubits 0 - 8 require no communication for all four quantum systems, and is achieved functioning of ≈ 0.44 sec per operator, which is very close to the border of the memory from 0.43 seconds.

The operators applied for higher qubits require communication. For the configuration of 32 nodes, we see ≈ 3.53 sec per operator, which is within 88% limit of the network of 3.12 sec. With the increase of the number of nodes, the time per operator is also increasing, compared to the border of the network. For example, the operations of the operator applied to qubit 35 on configuration from 256 nodes takes 8.7 seconds, which is an increase of 2.5×, in comparison with the border of the network. There are two reasons for such steep increase of the time per operator. First, this is due to the conflict situation in the network. The system uses topology of two levels of Clos (13) 20 nodes (40 sockets) to the switches at first level. As a result, sockets, which are separated by more than 40 sockets will communicate via the switches at second level. They have a lower bandwidth from those at first level which gives rise to a conflict situation and increase in the functioning. Simulations that use 256, 1 K and 2 K sockets, cover the switches at first level. This requires communication by means of the switches at second level, and resulting in increased conflict situation between the communicating sockets ([4]). In addition, there is interference with other jobs operating on the system at the same time, as it is observed to some time of variability between the operations in our experiments. In the worst case, for the 2K nodes, the time per operator increases to 11.15 seconds, which corresponds to 3.5× (11.15/3.12) increase in time per operator, in comparison with the restriction of the network.

IQDE possesses many features that are not available at the static quantum simulators, such as:

- interactive work at the time of creating the algorithms, dynamic testing and debugging at each step;
- integrated 36 quantum operators;
- automatic garbage collection;
- the system provides much more possibilities for capturing and handling of exceptions from the static simulators;
- fully dynamic environment in which code, different types of operators and data can be replaced during the implementation until they perform different tasks;
- the ability to control the interactive environment itself at all levels, including both implementation and testing, allows the user to effectively create specific problematic-oriented decisions;

All this is applicable to the parallel computation and will enable new programming techniques, paradigms and algorithms to be created on the basis of a program model bottom-up, in an interactive environment with a high level of abstraction. The interactive environment opens up opportunities for completely new approaches for organization of the computation process.

6. Conclusions

The practical limit of the size of the quantum system that can be simulated cannot be drastically increased due to large exponential requirements for the memory for storing the entire state of the vector.

An interactive environment for quantum simulations is a critical tool for the development of reversible quantum circuits. The concept of quantum circuit simulation can still be scrutinized further in future research. Future work can be done in the area of definition of custom gates, save and load circuits, and option to add qubits.

The operation of a quantum simulator is limited by the memory and especially by the network bandwidth. The network bandwidth is also improved, but within a small speed of 26% per year.

Large opportunities for further acceleration of IQDE are created by the methods with avoidance of the communication. These methods combine the ideas for cache sharing, described in this publication, and offer restructuring. The cache sharing that uses the merging of operators decreases the amount of traffic of the memory, but the possibilities for merging are specific for the circuit and limited in scope. The reordering of the state, on the other hand, offers opportunities for merging. In particular, the reordering of the qubits changes in such a way that the qubits of higher order, which require communication between the nodes, become qubits of lower order, thus avoiding the communication of the quantum operators that operate on these qubits. The main challenge is to maximize the possibilities for reordering of a given quantum circuit, while minimizing the limits for computational reordering. And finally we have to pay attention that the optimization by reordering may allow storing in the state of the vector in a secondary device with higher capacity, such as disk, thus reducing the costs for transfer of data from and to the main memory. In principle, this may facilitate the quantum simulations with more than 8 qubits."

References

[1] Ishizaki, A. and Fleming, G.R. (2009) Theoretical Examination of Quantum Coherence in a Photosynthetic System at Physiological Temperature. *Proceedings of the National Academy of Sciences of the United States of America*, **106**, 17255-17260. http://dx.doi.org/10.1073/pnas.0908989106

[2] List of QC Simulators. https://quantiki.org/wiki/list-qc-simulators

[3] Jozsa, R. and Linden, N. (2003) On the Role of Entanglement in Quantum-Computational Speed-Up. *Proceedings of the Royal Society of London Series A*, **459**, 2011-2032. http://dx.doi.org/10.1098/rspa.2002.1097

[4] Kim, J., Dally, W.J., Scott, S. and Abts, D. (2008) Technology-Driven, Highly-Scalable Dragonfly Topology. *SIGARCH Computer Architecture News*, **36**, 77-88. http://dx.doi.org/10.1145/1394608.1382129

[5] Kitaev, A. (1995) Quantum Measurements and the Abelian Stabilizer Problem. http://arxiv.org/abs/quant-ph/9511026

[6] Lam, M.D., Rothberg, E.E. and Wolf, M.E. (1991) The Cache Performance and Optimizations of Blocked Algorithms. *SIGPLAN Notices*, **26**, 63-74. http://dx.doi.org/10.1145/106973.106981

[7] Li, S., Ahn, J.H., Strong, R.D., Brockman, J.B., Tullsen, D.M. and Jouppi, N.P. (2009) McPAT: An Integrated Power, Area, and Timing Modeling Framework Formulticore and Manycore Architectures. In *Proceedings of the* 42nd *Annual IEEE/ACM International Symposium on Microarchitecture, MICRO*, New York, 12-16 December 2009, 469-480.

[8] Lomont, C. (2004) The Hidden Subgroup Problem—Review and Open Problems. http://arxiv.org/abs/quant-ph/0411037

[9] Markov, I.L. and Shi, Y. (2008) "Simulating Quantum Computation by Contracting Tensor Networks. *SIAM Journal on Computing*, **38**, 963-981. http://dx.doi.org/10.1137/050644756

[10] Raychev, N. (2014) Reply to "The Classical-Quantum Boundary for Correlations: Discord and Related Measures". *Abstract and Applied Analysis*, **94**, 1455-1465.

[11] Raychev, N. (2015) Mathematical Approaches for Modified Quantum Calculation. *International Journal of Scientific and Engineering Research*, **6**, 1302-1309. http://dx.doi.org/10.14299/ijser.2015.08.006

[12] Raychev, N. (2015) Quantum Computing Models for Algebraic Applications. *International Journal of Scientific and Engineering Research* , **6**, 1281-1288. http://dx.doi.org/10.14299/ijser.2015.08.003

[13] Tóth, G. and Gühne, O. (2005) Entanglement Detection in the Stabilizer Formalism. *Physical Review A*, **72**, 022340.

[14] Raychev, N. (2015) Indexed Cluster of Controlled Computational Operators. *International Journal of Scientific and Engineering Research*, **6**, 1295-1301. http://dx.doi.org/10.14299/ijser.2015.08.005

[15] Raychev, N. (2015) Quantum Multidimensional Operators with Many Controls. *International Journal of Scientific and Engineering Research*, **6**, 1310-1317. http://dx.doi.org/10.14299/ijser.2015.08.007