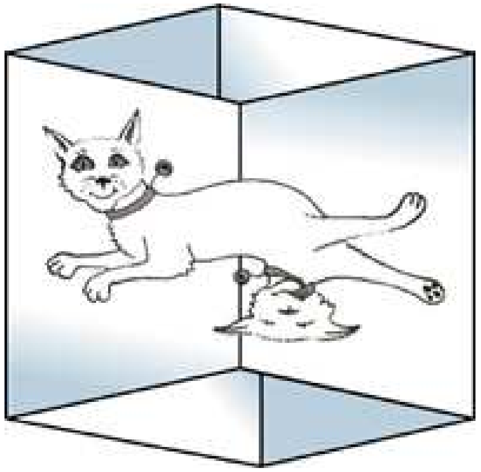
4th year Project demo presentation

Colm Ó hÉigeartaigh

CASE4 - 99387212

`coheig-case4@computing.dcu.ie`

Table of Contents



An Introduction to Quantum Computing

- A qubit has two base states, denoted by the *Dirac* notation, $|0\rangle$ and $|1\rangle$.
- A qubit can be in a linear combination of states, denoted by;

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

- The power of quantum computing largely derives from two physical phenomena; superposition and entanglement.

Superposition and Entanglement

- *Superposition* is the property of being able to exist in multiple states at the same time.
- *Entanglement* is a correlation between qubits that is stronger than any possible correlation in classical physics. An example is one of the Bell states;

$$|\phi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{(2)}} \tag{2}$$

- Bell states are used as the basis for quantum teleportation and super-dense coding.

Quantum Algorithms

- The two most famous quantum algorithms are Shor’s Algorithm and Grover’s Algorithm.
- Shor’s algorithm can factor a large composite number that is the product of two prime numbers in polynomial time.
- Grover’s algorithm can search an unstructured database in quadratic time.

Quantum Algorithms(2)

- Discovering new algorithms is complicated by the difficulty in getting the quantum state to decohere to the wanted values.
- It is also complicated by the fact that every operation in Quantum Computing must be reversible. This means any matrix used must be Unitary. This is a major restriction on what can be done.

The future of Quantum Computing

- It is unclear as yet how powerful the quantum computing paradigm is. The fact that an NP problem such as factoring can be solved in exponential time is encouraging.
- The complexity space of the quantum computer is a subset of *PSPACE*, ie. those problems bounded on memory, but unbounded on time.
- It is probable Quantum Computing will solve a few more NP problems, and speed up the solution to many more.

The Future(2)

- A useful Quantum Computer has never been built, due to the engineering difficulties involved in preventing the quantum state from decohering.
- A Quantum Computer was built in 2001 at IBM with 7 qubits, which demonstrated Shor’s algorithm, factoring 15 into 5 and 3!

The Quantum Computing Language

- The QCL is a programming language designed to approach quantum computing programming using the syntax of a procedural language like "C".
- It provides a base set of operators, yet is extremely powerful.
- The QCL contains a number of classical components such as if statements, for/while/until loops, functions, etc.

The QCL(2)

- Qubits are manipulated by declaring quantum registers, *qureg*, with an arbitrary number of qubits. An operator can be applied to a quantum register.
- QCL defines many operators for quantum registers, among them; Rot(real theta, qureg q) and Mix(qureg q).

The Bloch Sphere

- A qubit is normally represented as a linear combination of the basis states $|0\rangle$ and $|1\rangle$;

(3) $$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$$

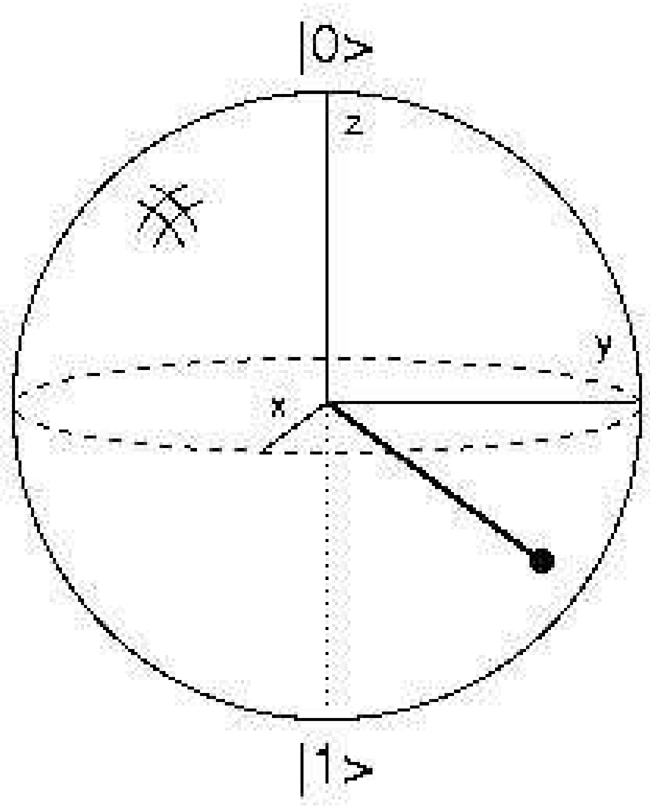
- This can also be represented as;

(4) $$|\phi\rangle = cos\frac{\theta}{2}|0\rangle + e^{i\varphi}sin\frac{\theta}{2}|1\rangle$$

- The numbers θ and φ in equation (2) define a point on the unit three-dimensional sphere. This sphere is called the *Bloch Sphere*.

The Bloch Sphere(2)

Bloch sphere for: 0.894427|0> + 0.447214|1>



The Bloch Sphere(3)

- The Bloch Sphere provides a useful means of visualizing the state of a single qubit. However, there is no simple generalization of the Bloch sphere known for multiple qubits.
- A classical bit would be represented on the bloch sphere as being either at the north pole of the sphere or at the south pole.
- A qubit however, can be a point anywhere on the surface of the sphere.

The Bloch Sphere(4)

- The latitude defines how close the qubit is to the poles, depending on the probability amplitudes.
- The qubit exists on *every* point on the longitude semicircle
- To draw the bloch sphere, the GNU libplot library is used. Libplot is a freely available C/C++ function library for device-independent 2-D vector graphics.

The Bloch Sphere(5)

- The QCL uses GNU Bison and Flex to provide a correct syntax for the language.
- Bison is used in the QCL to ensure that whatever is typed in at the command line is syntactically correct in accordance with the grammar of the QCL.
- Flex is used to scan the input and to execute the corresponding C++ code.
- The semantics of the Bloch Sphere command are handled deeper in the code.

The GUI

- The server program is designed to run on a Linux cluster and monitors the nodes dynamically. It packages this information in a class and broadcasts it to the client programs, which display the information graphically.
- The server program extracts information from the cluster by running and parsing various Linux commands.
- The client program displays the information dynamically, by querying the server every five seconds.

The GUI

- The client uses the proxy pattern to contact the server, and the observer pattern is used on the server, to update the object the server broadcasts whenever the state of the cluster changes.
- The client application is embedded inside an application called *WeirdX*. *WeirdX* is a pure Java X Window System Server.
- It allows you to run a graphical application on a server machine and then to redirect the graphical output to another machine where it is displayed using *WeirdX*.

Parallelizing the QCL-Motivation

- Simulating a quantum computer on a classical computer is a computationally hard problem.
- As the qubits in the quantum register are superposed with each other, the number of basevectos increases exponentially.
- Applying an operation to a quantum state is simply a matrix-vector multiplication.

Parallelizing the QCL(2)

- The CA Linux cluster was used in this project for parallel computation in the QCL.
- The mpich implementation of the Message Passing Interface(MPI) library is used in this project.
- The number of nodes to run the program must be specified on the command line.
- QCL is allowed to execute normally on the head node, all other nodes are trapped inside a loop awaiting instructions from the head node.

Parallelizing the QCL(3)

- Data is sent to nodes from the head node and gathered back in using different MPI operations.
- Various kinds of matrix-vector multiplication algorithms are implemented in the QCL.
- The Block-checkerboard partitioning algorithm sees the matrix being divided up into small squares of size (2x2). Each node gets a block and a portion of the vector.

Parallelizing the QCL(4)

- The Self-Scheduling or Master-Slave algorithm broadcasts the vector X to each node, and the farms out one row at a time to all the nodes. This is inefficient due to the large amount of communication required.
- The Block-striped partioning algorithm stripes a matrix of size (n x n) row-wise among p processes, so that each processor stores n/p rows of the matrix.
- The next approach is combine the *Compressed Sparse Row* storage format with the block-striped partioning approach.

Parallelizing the QCL(5)

- An efficient way of storing spares matrices is the *CSR* approach. Instead of a large rectangular array, three arrays are used to store the matrix. This greatly reduces communication overhead.
- The final approach uses block-checkerboard partitioning again, but it is more advanced than the first example, in that it only sends the vector portions to the head nodes of each column of the matrix, who then redistribute it down the column.

Parallelizing the QCL(6)

- When all the results come back from the nodes, they need to be recombined, and the answer must be stored back in the original quantum register.