慶應義塾大学理工学部　物理情報工学科 Keio University

Trading Classical for Quantum Computation Using Indirection

Rodney Van Meter rdv@tera.ics.keio.ac.jp MS+S 2004

Any software problem can be solved by adding another layer of indirection.
Steven M. Bellovin

Goal

Quantum computation is slow, expensive, and error-prone. By doing some of the work classically, we can lower the *total cost* of the computation.

Basic Idea

Use part of superposition |*x*> as an index into a table. For Shor's algorithm[1], the table contains classically computed parts of the modular exponentiation of *a*. This allows us to reduce the number of multiplications necessary in the quantum domain by a factor of *w*, in exchange for 2^w more classical multiplications.

Indirection

Pointers (memory addresses) and array indices are the most common forms of indirection. Indirection saves space, allows sharing of objects, and allows data to be filled in later in the computation.
Indirection may be an arbitrary number of levels deep; we are using only one, for the moment.

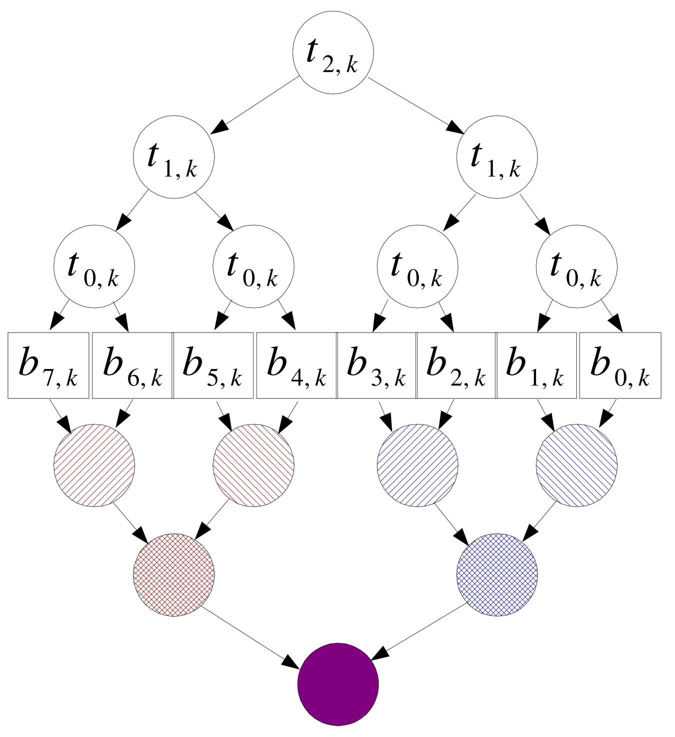
Arguments by value: add 4,3 (zero levels of indirection)
Arguments by reference: add this, that (one level of indirection)

4

3

Quantum-Addressable Classical Memory (QACM)

The best implementation will use an array that holds classically-computed values, and allows a superposition of addresses to be used to retrieve a superposition of those values. We call this a *quantum-addressable classical memory* (QACM)[2].



Three-bit Example

Address superposition used to steer reader to appropriate data cells

Classical data cells

Output superposition

We use $|b[|t>]>$ to indicate the superposition retrieved from the array *b* by using $|t>$ as the index. Our QACM will need to hold 2^w entries of *n* bits each to speed up modular exponentiation of an *n* bit number by a factor of *w*.

References

1) P. Shor, "Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer", *SIAM J. Comp.*, 1997.
2) M. Nielsen & I. Chuang, *Quantum Computation and Quantum Information*, pp. 266—268, Cambridge University Press, 2000.
3) N. Kunihiro, "Practical running time of factoring by quantum circuits", *EQIS* 2003.
4) A. Barenco *et al*, "Approximate Fourier transform and decoherence", *PRA*, 1996.

Modular Exponentiation in Shor's Algorithm

Modular exponentiation is the dominant cost in Shor's algorithm. *n* classical modular multiplications and *n* quantum modular multiplications are used in the standard method.

Let $d_j = a^{2^j} \ mod\ N$
d_j values are computed classically[3].

To factor the *n* bit number N, we must evolve to hold: $\frac{1}{\sqrt{q}}\sum_{j=0}^{q-1} |a^j \ mod\ N\rangle$

Binary expansion of $x : x_{n-1}x_{n-2}\ldots x_0$

$$a^x \ mod\ N = \prod_{j=0}^{n-1} d_j^{x_j} \ mod\ N$$

Iterating By Words Instead of Bits

Divide *x* into *l* words of length *w*, $l = n/w$

$|t_k(x)\rangle$ will be index into array *b*.

For iteration *k*, $b_{m,k} = a^{m2^{wk}} \ mod\ N$

$$a^x \ mod\ N = \prod_{j=0}^{l-1} b_{t_j(x),j} \ mod\ N$$

$$\underbrace{x_{n-1}x_{n-2}\ldots}_{t_{l-1}(x)}\ldots \underbrace{x_{2w}x_{2w-1}\ldots x_w}_{t_1(x)}\underbrace{x_{w-1}\ldots x_0}_{t_0(x)}$$

For the superposition |*x*>, this becomes

$$\frac{1}{\sqrt{q}}\sum_{x=0}^{q-1} |a^x \ mod\ N\rangle = \frac{1}{\sqrt{q}}\sum_{x=0}^{q-1} |\prod_{j=0}^{l-1} b_{t_j(x),j} \ mod\ N\rangle$$

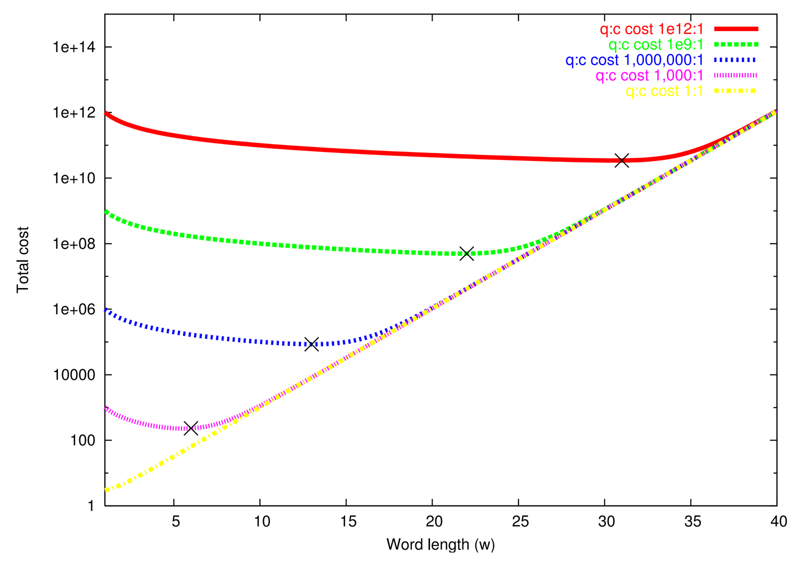
Putting It All Together: the Algorithm

1) Initialize |*product*> to 1
2) Classically calculate initial $b_{j,0}$ table values for all *j*, $0 <= j < 2^w$
3) For *k* from 0 to *l*-1, do
   a) In quantum domain, use $|t_k(x)>$ as index into *b*,
      $|c> = |b[|t_k(x)>]>$
   b) multiply |*product*> by |c>, modulo N
   b) Update *b* array: classically multiply each element by itself (square it, modulo N) *w* times to create $b_{j,k+1}$

One-Time Running Time

$lw2^w = n2^w$ classical modular multiplications and
$l = n/w$ quantum multiplications
(compared to *n* classical and *n* quantum using standard algorithm)

Total Cost and Selecting Word Size w



Q:C cost ratio	Optimum *w*	Entries in *b* array	Cost reduction
1000	6	64	4.3x
1.00E+006	13	8K	11.7x
1.00E+009	22	4M	20.2x
1.00E+012	31	2G	29.1x

Minimum cost depends on ratio of quantum to classical multiplication cost. Quantum cost must factor in repetitions of Shor's algorithm.
Curve can be used for different definitions of cost, e.g., economic, or wall clock time. (Graph assumes zero QACM cost.)

Repetition and Reuse

Barenco *et al* showed that the probability of success with Shor's algorithm may be 10^{-5} for sizes of a few kilobits[4].
When repeating the algorithm, the classical parts can be saved and reused. The quantum portion of the exponentiation must be redone.