Two real-time applications of quantum computing for the evaluation of WCETs

Gabriella Bettonte, Stéphane Louise, Renaud Sirdey

Compas'2021 : Parallélisme / Architecture/ Système/ Temps Réel
MILC - Lyon, France, 5-9 juillet 2021

Two real-time applications of quantum computing for the evaluation of WCETs

Gabriella Bettonte, Stéphane Louise, Renaud Sirdey

Université Paris-Saclay,
CEA List - Palaiseau - France
gabriella.bettonte@cea.fr , stephane.louise@cea.fr , renaud.sirdey@cea.fr

---

Abstract
While the interest in quantum computing is constantly rising, the design of quantum algorithms suitable for real applications is still in its infancy. We hereby investigate the issues of worst-case execution times (WCETs) which are fundamental for the validation of real-time systems in which all time constraints must be met. Since the cache handling in non-deterministic sequence of memory accesses impacts substantially on the execution time, we use it as a case study to design quantum algorithms meant to improve the static analysis of cache misses on programs performing random accesses to the memory and in programs where preemption is allowed. Cache misses are connected to the execution time of a program : for in-order processors, the worst-case execution time is tied to the path with the highest amount of cache misses.

Key words : Calcul quantique, pire temps d'exécution, préemptions, applications temps réel

---

1. Introduction

The evaluation of worst-case execution-times (WCETs) on real-time applications plays a important role in the validation of real-time systems. It is a requirement that all time constraints are met to ensure the validity of a system but, at the same time, there exists the risk of building an over-engineered, and thus too expensive, system. In other words, a trade-off between precision and efficiency must be reached in the evaluation of WCETs. Quantum computing could bring benefits for this purpose because of its intrinsic parallelism that can boost computations but this also comes with a caveat : one must identify precisely the problems where quantum computing can bring real advantages. In this paper we show two applications of quantum computing applied in the evaluation of WCETs. The first is an evaluation of WCETs in a sequence of memory accesses with some random accesses where usually a classical approach performs poorly. In the second application, the WCETs are evaluated in a deterministic sequence of memory accesses that can be preempted at any time, representing a difficult problem because of its low predictability.

2. Background overview

In our discussion we use concepts from quantum computing and cache memories, which we briefly overview in this section.

2.1. On quantum computing

Classic computers rely on bits, that are deterministic : they can be either on (in the state 0) or off (in the state 1). In contrast, qubits (short for quantum bits) have the capacity of being in a superposition of $|0\rangle$ and $|1\rangle$ (in Dirac notation [1]). More precisely,

$$\psi \coloneqq \alpha\,|0\rangle + \beta\,|1\rangle\,, |\,\alpha\,|^2 + |\,\beta\,|^2 = 1, \text{ with } \alpha, \beta \in \mathbb{C} \tag{1}$$

A part of the interest that quantum computing arises, comes from the ability of qubits to generate states of superposition that reflect all the possible outcomes of a given algorithm, up until a measurement is done, which is usually at the end of the algorithm. This property is called "*quantum parallelism*" and could theoretically give a massive performance boost for quantum algorithms. However the advantages of quantum computing is not without caveats : only some classes of problems can be solved by quantum computing with a significant gain in terms of efficiency with respect to classical computing. Indeed, one important research issue related to quantum computing is defining with precision those kinds of problems [10].

To give an idea of the momentum the scientific community gives to quantum computing [3], the Quantum Algorithm Zoo [1] (at the time of writing) cites 430 papers and counting on quantum algorithms, and enterprises such as Google, IBM or Rigetti Computing are investing a lot of resources on research on quantum technologies. Among all quantum algorithms we can identify two main important blueprints. The first one is defined by quantum algorithms able to reach an exponential speedup over classical algorithms on precisely defined and heavily structured mathematical problems, as e.g. Shor's algorithm for integer factorization. The second one is defined by quantum algorithms with a polynomial speedup over classical ones, as e.g. Grover's algorithm for searching an unstructured array. Still, a lot of questions remain open about the real world applications and benefits of quantum computing [7].

2.2. On cache memory and WCETs

Real-time systems are computer systems for which processing outcomes must also meet timing constraints which otherwise would jeopardize the real-time system or its environment, potentially including the life of human beings. With respect to the verification of this kind of systems, the evaluation of a WCET for programs and tasks is an important issue. In this context, the behavior of cache memories has usually the most significant impact on the execution-time of programs and must be taken into consideration while evaluating worst-case performance.

Cache memories are used to address the problem represented by the so-called memory wall : the gap between the speed of processors and the memory latencies increases by an order of magnitude every few years. The solution, largely applied since the '80s, has been to add little amounts of high-speed memory close to the processing parts –the cache memories– to provide faster access to a local copy of often or recently used values in memory, thus speeding up the average access time to memory.

The data accessed – and data close to accessed location, to exploit the principle of locality – are copied into the cache memory. The position of the copy inside the cache memory is decided by several parameters including its so-called associativity A –which determines the number of lines in which a particular copy of the memory can reside– size, and replacement policy.

1. Bra-ket, or Dirac, notations denote state vectors in quantum mechanics. $|0\rangle$ and $|1\rangle$ are two orthogonal base state vectors that denote the observable states of a qubit.

3. Applications

In this section we present two applications where quantum computing has the potential to be beneficial for the evaluation of WCETs. It is worth noting that the two patterns of work are not yet fully explored but they are the seeds of our future work.

3.1. Sequences of random memory accesses

We consider programs that perform non deterministic accesses to the memory, using the count of cache misses as a first proxy evaluation [2] of the WCET (or as the literature calls it, the *Cache memory related delay*). A"dual" approach has already been explored in the literature, with papers about applying static analysis techniques to quantum algorithms to evaluate their performance [6, 2]. Instead, the applications of quantum computing to improve the static analysis of cache misses (and more largely to static analysis problems at all) has still, to the best of our knowledge, to be fully explored.

We designed an algorithm that, using a quantum-inspired formalism, builds a superposition of all the possible sequences of memory accesses. The superposition produced by the algorithm is suitable to be used as input for a yet to be determined quantum algorithm that will boost the speed-up, allowing to count the worst number of cache misses with increased efficiency (i.e. using less time) with regards to the classical equivalent.

We consider evaluating the WCET with respect to a simple model of program that considers only a sequential sequence of memory accesses, some of which are deterministic, and others non deterministicly chosen from a given subset of memory addresses. We study in the following the case of a direct mapped cache memory (i.e. a block of the main memory can be mapped only to a specific block/line of the cache memory).

Despite its simplicity, this non-deterministic model of program we study in this paper is difficult to statically analyze for the evaluation of WCET with classical algorithms [3]. Indeed, the non-deterministic memory accesses produce a large set of possible execution paths for which a non exhaustive analysis can result in a large overestimation of the WCET.

3.1.1. Our algorithm : an overview

The sequence of memory accesses performed by the program, contains non-deterministic accesses which our quantum algorithm superposes in order to take into account, in *one* execution, the number of cache misses for each possible paths. Our final goal, which is still beyond the scope of this paper, is to design an algorithm that returns (with non negligible probability) the largest number of cache misses over every paths, which should give important hints about the WCET for in-order processors. Since the program performs some non-deterministic accesses to the memory there exist multiple possible linear sequences. Our goal is to exploit the quantum parallelism for computing a superposition of the total number of cache misses over each possible execution path. It is not difficult to produce a superposition of all the possible elements that can be chosen by means of a small network of Hadamard gates, in the example section we will give an hint on how it is possible in practise. For every access to the memory in each possible sequence of accesses, the algorithm checks if a cache miss or hit occurs and modifies the cache's content accordingly : after a cache miss, the accessed element is added to the cache to its assigned place while after a cache hit the cache state does not change (since it is direct-mapped,

2. This is true especially for in-order execution for which the execution-time grows monotonically with the number of memory cache misses.

3. It is worth noting that despite its apparent simplicity any single-threaded program is amenable to our model, by replacing at relevant points in the execution, accesses on different branches of the control-flow graph by a corresponding number of non-deterministic accesses.

every element has a unique possible position into the cache).
We designed a circuit $\mathcal{Q}$ that checks if a cache miss occurred, takes note of it and creates a new cache state that contains the considered element. It is worth noting that we have to create a new cache state instead of updating the previous one to maintain the reversibility of the quantum circuit. The circuit $\mathcal{Q}$ is the concatenation of the circuits $\mathcal{M}$ and $\mathcal{N}$ below. The miss operator $\mathcal{M}$ is designed to evaluate if the requested data is available in the cache (cache hit) or the program has to access to the main memory (cache miss). The inputs of the miss operator are the accessed data element, the actual state of the cache and a qubit b to store the result : at the beginning $b = |0\rangle$, while, after the application of the miss operator, the qubit b will be in the state $|1\rangle$ if a cache miss occurred or be left to the state $|0\rangle$ if a cache hit occurred. The operator "*Next*" $\mathcal{N}$ is built to create a new cache state from the previous one. The inputs of the "Next" operator $\mathcal{N}$ are : the accessed data element, the current state of the cache and the new state of the cache. It is required to repeat the $\mathcal{Q}$ circuit as many times as the number of accesses the program performs after the first non-deterministic access). The picture in the appendix represents the main flow of the algorithm. In this figure two iterations of the algorithm are represented. The number of qubits required to represent the considered access or cache state is indicated on the right. As result of the previous steps, we have m cache misses wires (*i.e.* qubits) : the wires associated to a cache miss are in the state $|1\rangle$ while the wires associated to a cache hit are in the state $|0\rangle$. As final step, we can use one of the many quantum sum operators in the literature, e.g. [5], to perform the sum of the qubits : the obtained value, plus the cache misses we counted before the first non-deterministic access, is the total number of memory-cache misses of the sequence we are considering. The largest value of the total number of cache misses for each possible path can be tied to an estimation of the (cache-related) WCET for the program considered in the case of in-order processors.

3.1.2. Example

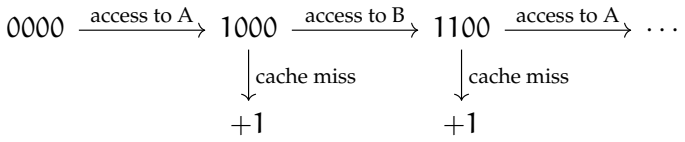
We consider a 1-way associative cache with 2 lines. We suppose to have this sequence of accesses :$ABABE^*AE^*$ where $E^* = \{E, F\}$. This means we have four deterministic accesses (A, B, A, B, A) and two non-deterministic accesses (E^* can be either E or F). We encode the four possible elements we can access to as $\alpha_0\alpha_1$.

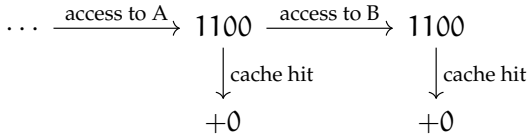
A	00	B	01	E	10	F	11

We need 4 qubits to store the state $\vec{s}$ of the cache : $\vec{s} = s_{00}s_{01}s_{10}s_{11}$.

$$\vec{s_i} = \begin{cases} 1 & \text{if } \alpha \text{ is in the cache} \\ 0 & \text{otherwise} \end{cases}.$$

In our example we have four possible situations that can happen : *ABABEAE*, *ABABEAF*, *ABABFAE* and *ABABFAF*.
At the beginning there are only deterministic accesses : we have 2 cache misses (A,B) and 2 cache hits (A,B).





We apply the Hadamard gate on the second qubits to build a superposition of E and F.
We provide some details of the miss operator $\mathcal{M}$ for our example to give an insight of our algorithm.

$$|\alpha \mid \vec{s} \mid b\rangle \longrightarrow |\alpha \mid \vec{s} \mid b \oplus \text{miss}(\alpha, \vec{s})\rangle \tag{2}$$

where α is the access, $\vec{s}$ the cache state and b the target qubit. In particular :

$$\text{miss}(\alpha, \vec{s}) = m_A \vee m_B \vee m_E \vee m_F \tag{3}$$

More precisely, $m_e = 1$ if we are accessing to the element e and it is not into the cache.
The number of (non-auxiliary) qubits needed by the operator $\mathcal{M}$ for our example is $(2+4+1)$:

3.2. Deterministic memory accesses sequences with preemption

Preemption is the act of interrupting one task to allow the execution of another task on a machine. The advantage is allowing the operating system to allocate the processor to urgent tasks. In particular, in fully preemptive systems, the running task can be interrupted at any time by another task with higher priority and be resumed to continue when all higher priority tasks have completed [4]. When the task is preempted, a large number of memory blocks belonging to the task are flushed away from the cache memory. When the preempted task resumes its execution, a substantial amount of time is spent to reload the cache with the previously displaced memory blocks, leading to a great increase of the task execution time [8].Preemption damages program locality and therefore it causes a degradation of system predictability, making WCETs more difficult to characterize and predict [11] [8] [4]. The total increase of the WCETs of a task is also a function of the total number of preemptions experienced. The usual approach to evaluate WCETs in case of preemption relies on the analysis of NP-problems i.e. finding the actual points of preemption. With our algorithm we want to avoid the NP- problem because even if we would obtain a decreased execution time, since it would be an exponential order of magnitude we would obtain again an exponential time, being very useful in real-world application.

3.2.1. Our algorithm : an overview

We take into consideration the example of one task producing a deterministic sequence of accesses preempted a fixed number of times k. The approach in [9] evaluates the WCETs (that is what interests engineers having to design real-time applications) avoiding to compute the exact point of preemptions and obtaining a speed-up. Following this idea, we are working on building a superposition of all the possible sequences of memory accesses to evaluate their execution time at once, in a similar way as the one described in the first application. In parallel, We are developing another approach with the aim of improving the precision, by finding the point of preemption. Being this a NP-problem, as mentioned before, for now we are considering very simple sequences of memory accesses. We designed a dynamic programming algorithm that given the sequence of memory accesses and the number of preemption $\mathcal{K}$ finds the "optimal" solution, i.e. the minimum number of cache hit and thus the worst-case scenario. We are working on a quantum version of this algorithm, building a superposition on the possible numbers of preemption k', where $k' < \mathcal{K}$. This dynamic programming algorithm, has a polynomial

complexity, but developing a quantum version of it we should obtain a decreased complexity, being a valuable result.

4. Conclusion

This paper aims to show the state of our research project until now, and both the applications we described are still work in progress. In the first application we built the superposition of memory accesses sequences but we still need to find a way to exploit its potential to obtain a valuable speed-up in the execution time. In particular, we need to build an oracle (black box) able to amplify the coefficient of the searched sequence (i.e. the one with the highest number of cache misses). In the second application we proposed two different approaches. In the first one we want to obtain a speed up of the computation of WCETs for the preempted sequence of memory accesses, by avoiding to actually find the points of preemption. In this case we still need to build the superposition, but the operators will probably be similar to the one presented for the first application. The second path privileges precision over efficiency, meaning that, on very simple sequences of memory accesses we want to find the points of preemption. To achieve that, we are building the superposition on the number of preemptions, but then we will have to rely on a quantum searching algorithm, such as Grover's algorithm, to actually isolate the solution with the desired number of preemptions. Both the applications we presented in this paper are paths of works that have the ambition of becoming two fully developed contributions in the next months.

Bibliography

1. The quantum algorithm zoo. `http://math.nist.gov/quantum/zoo/`.
2. Abhari (A. J.), Patil (S.), Kudrow (D.), Heckey (J.), Lvov (A.), Chong (F. T.) et Martonosi. (M.). – Scaffcc : a framework for compilation and analysis of quantum computing programs. *In Proc. 11th ACM Conf. on Computing Frontiers.*, 2014.
3. Bravyi (S.), Gosset (D.), König (R.) et et al. – Quantum advantage with noisy shallow circuits. *Nat. Phys.*, 2020.
4. Buttazzo (G.), Bertogna (M.) et Yao (G.). – Limited preemptive scheduling for real-time systems. *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 9, NO. 1*, 2013.
5. Cherkas (A.) et Chivilikhin (S.). – Quantum adder of classical numbers. *Journal of Physics : Conference Series*, 2016.
6. Facon (A.), Guilley (S.), Lec'Hvien (M.), Schaub (A.) et Souissi (Y.). – Detecting cache-timing vulnerabilities in post-quantum cryptography algorithms. *IEEE 3rd International Verification and Security Workshop*, 2018.
7. Hall (R. J.). – A quantum algorithm for software engineering search. *Proceedings of the 2009 IEEE/ACM Int. Conf. on Automated Software Engineering*, 2009.
8. Lee et al. – Analysis of cache-related preemption delay infixed-priority preemptive scheduling. *IEEE Trans. Comput.*, 1998.
9. Louise (S.). – A first step toward using quantum computing for low-level wcets estimations. *ACM Trans. Archit. Code Optim.*, 2019.
10. Nielsen (M. A.) et Chuang (I. L.). – *Quantum Computation and Quantum Information*. – Cambridge University Press, 2000.
11. Ramaprasad (H.) et Mueller (F.). – Tightening the bounds on feasible pre-emption points. *27th IEEE Real-Time Syst. Symp. (RTSS'06)*, 2006.