Influence of data clustering on in-order multi-core processing systems

D. Claeys, H. Bruneel, B. Steyaert, W. Mélange, and J. Walraevens

In multi-core in-order processing systems, only one core can be utilized when the instruction at the head of the instruction queue produces data input for the next instruction in the queue. Although in-order processing has been studied in the past, the influence of data clustering, i.e., the extent to which subsequent instructions rely on each other's data, has been largely overlooked. We therefore develop a queueing model and provide closed-form formulae for the stability condition and the average time before instructions are executed. These expressions clearly reflect that data clustering can have a devastating impact.

Introduction: Multi-core processing systems (architectures) cannot always work at full capacity as instructions might rely on data produced by other instructions. In order to alleviate this problem, the majority of the modern architectures adopt the "out-of-order execution paradigm" [2, 3], which means that instructions do not necessarily have to be executed by the original order in a program. However, this requires some additional software and transistors and thus leads to a larger cost and energy consumption. Therefore, Intel ATOM microprocessors, which are specifically developed for smartphones, PDA's and tablets where cost and energy efficiency are of primordial importance, adopt "in-order execution" [4, 5]. In such systems, only one processor core is active if the instruction at the head of the instruction queue produces data input for the next instruction in the queue , even if other instructions are present in the instruction queue that do not require that data. In this paper, we investigate the influence of "data clustering", i.e., the tendency of instructions to rely on data from previous instructions, on the overall performance. We believe that this effect has been largely overlooked in the existing literature. In the next section, we develop an analytic (queueing) model and we establish a formula for the mean delay of instructions. We then discuss this expression and some numerical examples.

Analytic model: Instructions arrive at the processing unit ("the system") and are placed in an instruction queue in awaitance of being executed by a processor core. The time axis is divided into fixed-length contiguous time periods which correspond to clock cycles, i.e., the time to execute an instruction. The number of arriving instructions during consecutive clock cycles is modelled by a sequence of independent and identically distributed random variables, with common probability generating function $E(z)$. The average number of arriving instructions during a clock cycle is denoted by λ and is by definition equal to $E^{'}(1)$ (we use primes to indicate derivatives). As the instruction queue is very large in practice to avoid loss of instructions due to a full buffer, it is assumed that the instruction queue has an infinite capacity. As in contemporary Intel Atom processors (Atom N5xx, D5xx, D2500, D2700, N2600 and N2800 [6, 7]), we consider a dual-core system, i.e., two processor cores are available. When two or more instructions are present, we denote the probability that the instruction at the head of the instruction queue produces data input for the next instruction in the queue by α. In that case, the second instruction cannot be executed although two cores are available and, as instructions are processed in order, i.e., in a first-come-first-served manner, the second available core cannot execute any instruction (the two instructions block all the others). In the other case, both cores execute an instruction.

In [1], a model has been analyzed whereby two classes of customers, called 1 and 2, enter a system with two types of servers, say A and B. Server A can only serve class-1 customers whereas server B is dedicated to class-2 customers. Customers of both classes are accomodated in one common queue and are served in their order of arrival. Subsequent customers belong to the same class with probability α.

Although the model in [1] is fundamentally different as the model in the present paper, because one class of customers (the instructions) and two identical servers (the processor cores) are considered, both systems behave identical: when at least two customers (instructions) are present, only one can be served (executed) with probability α and two customers (instructions) can be served (executed) with probability $1-\alpha$. We can thus rely on the results that have been deduced in [1]. The stability condition then reads

$$\lambda < 2 - \alpha \ . \tag{1}$$

The stability condition describes for which range of mean arrival rates it is guaranteed that all instructions can be executed within a finite time. The right-hand-side of (1) thus represents the supremum of the tolerable mean arrival rate of the processing system. When the stability condition is fulfilled, the average time - denoted by $\mathrm{E}[d]$ - until an instruction is executed equals:

$$\mathrm{E}[d] = \frac{-2u(0)(1-\alpha) + 2(2-\alpha-\lambda)(\lambda+1) + 2(\alpha\lambda-1) + E^{''}(1)}{2\lambda(2-\alpha-\lambda)} \ , \tag{2}$$

whereby

$$u(0) = \frac{(2-\alpha-\lambda)z_1}{(1-\alpha)(z_1-1)} \ ,$$

with z_1 the unique root of $z^2 - (1-\alpha+\alpha z)E(z)$ inside the open complex unit disk $\{z \in \mathbb{C} : |z| < 1\}$. For instance, in the special case of geometric arrivals, i.e.,

$$E(z) = \frac{1}{1+\lambda-\lambda z} \ ,$$

expression (2) for $\mathrm{E}[d]$ transforms into the following closed-form formula:

$$\mathrm{E}[d] = \frac{1-2\lambda-\sqrt{1+4\lambda(1-\alpha)}}{2\lambda(\lambda-2+\alpha)} \ .$$

Discussion of results and numerical examples: In this section, we discuss our results and some numerical examples. Let us first examine the stability condition. Equation (1) exhibits that α has a direct impact: the supremum of the tolerable mean arrival rate decreases linearly with α. In addition, when $\alpha = 0$, both cores are always active whenever at least two instructions are present, because no data dependency appears, whereas in the opposite case ($\alpha = 1$), instructions are always data dependent so that the system becomes equivalent with a single-core processing system.

Next, in Fig. 1, the average time $\mathrm{E}[d]$ until instructions are executed is depicted versus λ, for various values of α and for a geometric distribution of the number of per-cycle instruction arrivals. We observe

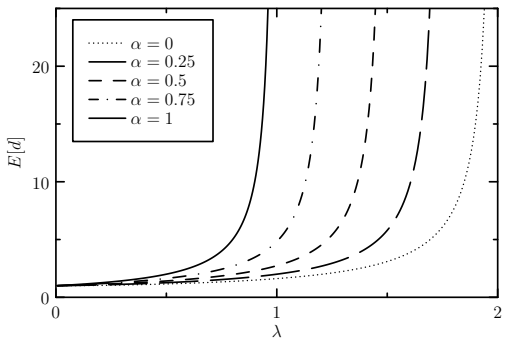

Fig. 1. $\mathrm{E}[d]$ versus λ, for various values of α

that data clustering has a negative impact, in the sense that $\mathrm{E}[d]$ increases and the stability region shrinks. Finally, $\mathrm{E}[d]$ is shown versus α for various values of λ in Fig. 2. Fig. 2 highlights that data clustering has a devastating impact, except for small mean arrival rates. Indeed, in the latter case, the number of arriving instructions is considerably less than the number of instructions that can be executed by a single core and therefore, the question of whether the second core is also active or not - determined by α - is not very relevant.

Conclusion: We have developed an analytic model in order to investigate the influence of data clustering, i.e., the tendency of instructions to rely on data from previous instructions, on the performance of a dual-core in-order processing system. We have provided an expression for the supremum of the average tolerable mean arrival rate of instructions as well as for the average time until instructions are executed. These formulas exhibit the very direct and devastating impact of data clustering. When data is

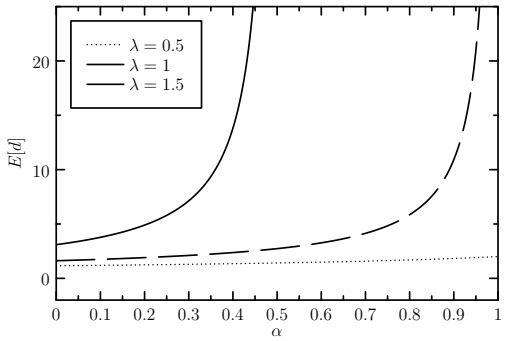

Fig. 2. E $[d]$ versus α, for various values of λ

heavily clustered, the system even becomes nearly equivalent with a single-core processing system. Only in case of small loads, data clustering has a negligible impact.

Acknowledgment: The last author is a Postdoctoral Fellow with the Fund for Scientific Research, Flanders (F.W.O.-Vlaanderen), Belgium.

D. Claeys, H. Bruneel, B. Steyaert, W. Mélange and J. Walraevens (*Ghent University, Department of Telecommunications and Information Processing, SMACS Research Group, Sint-Pietersnieuwstraat 41, B-9000 Gent, Belgium*)

E-mail: dclaeys@telin.UGent.be

References

1 Bruneel, H., Mélange, W., Steyaert, B., Claeys, D., and Walraevens, J.: 'Impact of blocking when customers of different classes are accommodated in one common queue', *Proceedings of the 1st International Conference on Operations Research and Enterprise Systems (ICORES'12)*, 2012, pp. 31-38.
2 Hennessy, J.L., and Patterson, D.A.: *Computer Architecture: A Quantitative Approach, 5th Edition*, Elsevier, 2012.
3 http://en.wikipedia.org/wiki/Out-of-order_execution
4 http://www.intel.com/content/www/us/en/processors/atom/atom-processor.html
5 http://software.intel.com/sites/products/documentation/hpc/atom/application/optimized-for-intel-atom-processor.pdf
6 http://en.wikipedia.org/wiki/Intel_Atom
7 http://ark.intel.com/search/advanced?FamilyText=Intel%C2%AE%20Atom%E2%84%A2%20Processor