Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Discrete Mathematics 36 (2010) 119–126
Electronic Notes in DISCRETE MATHEMATICS
www.elsevier.com/locate/endm

Speculative data prefetching for branching structures in dataflow programms

Sergiu Carpov [a,b,1] Renaud Sirdey [a,1] Jacques Carlier [b,1] Dritan Nace [b,1]

[a] *CEA LIST, Embedded Real Time Systems Laboratory, Point Courrier 94, Gif-sur-Yvette, 91191 France.*

[b] *UMR CNRS 6599 Heudiasyc, Université de Technologie de Compiègne, BP 20529, 60205 Compiègne Cedex, France.*

Abstract

This paper deals, to some extent, with the problem of speculative data prefetching for dataflow programming models. We focus on finding optimum prefetch strategies for a simple n-way dataflow branching structure with respect to several objective functions and exhibit polynomial algorithms for doing so.

Keywords: Knapsack, Shortest Path, Parallel Computing, OR in Compilation.

1 Introduction

With the frequency version of Moore's law coming to an end, a new generation of massively multi-core microprocessors is emerging. This has triggered a regain of interest for the so-called dataflow programming models in which one

[1] Emails: `[sergiu.carpov,renaud.sirdey]@cea.fr`, `[carlier,dnace]@hds.utc.fr`

1571-0653/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.endm.2010.05.016

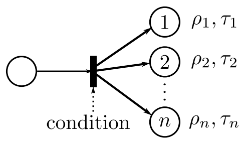

Fig. 1. An n-output branching structure.

expresses computation-intensive applications as networks of processes (also called agents) communicating through (and only through) FIFO channels [4,2].

In particular, one central issue is to efficiently use the bandwidth between the (huge) off-chip external memory and the (scarce) on-chip one in order to keep the many processing cores fed with data and, hence, busy. Doing so relies heavily on prefetching data from this off-chip memory, that is loading data on-chip before it is effectively needed. To achieve high performances in presence of data dependent control, one should further speculate on data prefetching, that is, loading data before it is even known whether it is needed or not.

In this paper, we consider an n-output branching structure as depicted on Fig. 1. Let ρ_i denote the time required for loading the data on which the tasks in the i-th branch depend and let τ_i denote the execution time of those tasks (assuming all the off-chip data have been loaded). We further assume that there are no common data between branches (a mildly restrictive assumption that will be relaxed in a subsequent paper). Our goal, is then to find optimal data prefetching strategies so as to minimize objective functions as the mathematical expectation and the worst-case of the execution time.

In both cases, two different prefetching strategies are examined: a *fractional* strategy, in which one is allowed to prefetch only fractions of branch data, and an *all-or-nothing* strategy in which this possibility is not allowed.

This paper, is organized as follows. Section 2 focuses on mathematical expectation, Section 3 deals with the more complicated case of worst-case execution time and Section 4 concludes.

2 Mathematical expectation of the execution time

We start by investigating the fractional prefetching problem with the mathematical expectation of the execution time as objective. For an n-output branching structure, let p_i denote the probability of the i-th branch to be executed[2] ($\sum_i p_i = 1$). Also suppose that the available prefetching time is constant and denoted by t (after this time elapses, one and only one of the

[2] It is further assumed that subsequent decisions are independent

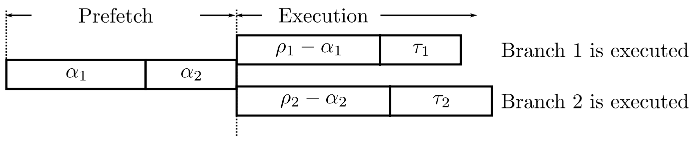

Fig. 2. Example of a 2-output branching structure execution.

branches is executed). We look for optimal prefetching durations $0 \leq \alpha_i \leq \rho_i$, such that the mathematical expectation of the execution time is minimal. An example of a 2-output branching structure execution is presented in Fig. 2.

The available prefetching time t can take any value in the range $\mathbb{D} = [0, \sum_i \rho_i[$, so the degenerate case, when all the data can be prefetched, is omitted.

This problem can be formulated as a linear program. Indeed, the following linear program minimizes the mathematical expectation of the execution time for a branching structure under a prefetching time constraint:

$$\begin{array}{rlcrl} \text{Minimize} & \displaystyle\sum_i p_i \left(\rho_i - \alpha_i + \tau_i\right) & & \text{Maximize} & \displaystyle\sum_i p_i \rho_i x_i \\ \text{s.t.} & \displaystyle\sum_i \alpha_i = t & \Rightarrow & \text{s.t.} & \displaystyle\sum_i \rho_i x_i = t \\ & \alpha_i \in [0, \rho_i], \ \forall i & & & x_i \in [0,1], \ \forall i \end{array}$$

The last linear program is obtained by substituting $\alpha_i = \rho_i x_i$ and taking the complement of objective function, knowing that $\sum_i p_i (\rho_i + \tau_i)$ is constant.

This program is nothing else but the linear programming form of the fractional knapsack problem, which can be solved exactly in polynomial time using the well-known Dantzig algorithm [3]. This algorithm consists in prefetching the branches in decreasing order of their probabilities, as long as the prefetching time allows it.

Although elementary, this is a very interesting result: we obtain a solution which structure does not depend on the available prefething time t. Futhermore, the branches are prefetched in decreasing order of their probabilities, that is, a branch is entirely loaded before the next branch will start to be prefetched. Thus, the resolution of the fractional, in fact, gives an optimum, robust all-or-nothing strategy.

Of course, when the branch probabilities are equal, the order in which the branches are prefetched does not matter. This model is interesting in an iterative compilation process [1], when in function of empirical results (gathered by running the application), accurate estimates of the probabilities can be obtained.

3 Worst-case execution time

As in the previous section, we begin by investigating the fractional prefetching problem, and then, consider the all-or-nothing case.

3.1 Fractional prefetch

As previously, let us consider an n-output branching structure, and suppose that the available prefetching time is constant and equal to t. We look for optimal prefetching durations $0 \le \alpha_i \le \rho_i$, such that the worst-case execution time is minimal.

During the prefetching period (see Fig. 2), branch i is prefetched for α_i time. If the branch i is executed, then the execution time will be equal to $\rho_i - \alpha_i + \tau_i$. Our goal is to minimize the worst-case execution time, thus the largest one of these terms. The problem can be stated as a mathematical program, which can be easily rewritten as a linear program:

$$\begin{array}{rl} \text{Minimize} & \max\limits_i (\rho_i - \alpha_i + \tau_i) \\ \text{s.t.} & \sum\limits_i \alpha_i = t \\ & \alpha_i \in [0, \rho_i], \forall i \end{array} \quad \Rightarrow \quad \begin{array}{rl} \text{Minimize} & \Gamma \\ \text{s.t.} & \rho_i - \alpha_i + \tau_i \le \Gamma, \forall i \\ & \sum\limits_i \alpha_i = t \\ & \alpha_i \in [0, \rho_i], \forall i \end{array}$$

Proposition 3.1. *Let K be the set of branches that verify relation $\tau_k + \rho_k \le \max_i \tau_i$, $k \in K$. The branches from K do not influence the value of the worst-case execution time.*

Proof. Let α_i be the optimal prefetching durations. If after the prefetching period a branch belonging to K is executed then the worst case execution time cannot be lower than $\max_i \tau_i$. □

Hence, without loss of generality we suppose that $\min_k (\tau_k + \rho_k) > \max_i \tau_i$ is verified.

3.2 All-or-nothing prefetch

Contrary to the expectation case, the solution of the fractional problem does not hint at an optimum prefetching time independent all-or-nothing strategy. The purpose of this section is to find such a solution although it does not in the general case always realize the smallest worst-case execution time.

Before describing the problem, we introduce some preliminary notions.

Let $\sigma \in \Pi(n)$ be a branch prefetching order. The time at which the branch at position k in the order σ is prefetched, is denoted by $l_k = \sum_{i \le k} \rho_{\sigma(i)}$.

Definition 3.2. Let $f_\sigma : \mathbb{D} \to [\max_i \tau_i, \max_i (\tau_i + \rho_i)]$ be a bijection, such that $f_\sigma(t)$ equals to the *worst-case execution time* when the available prefetching time is $t \in \mathbb{D}$. Then, for any $k = 1, \dots, n$ and $t \in [l_{k-1}, l_k[$ we have:

$$f_\sigma(t) = \max\left(\Lambda_k, \rho_{\sigma(k)} + \tau_{\sigma(k)} - t + l_{k-1}\right),$$

$$\text{where } \Lambda_k = \begin{cases} \max_{i>k}\left(\tau_{\sigma(i)} + \rho_{\sigma(i)}\right) & \text{if } k < n, \\ \max_i \tau_i & \text{otherwise.} \end{cases}$$

The all-or-nothing prefetch problem with worst-case execution time minimization is formulated as follows. Let us consider an n-output branching structure. We look for a branch prefetching order $\sigma \in \Pi(n)$, such that for any $\sigma' \in \Pi(n)$, $t \in \mathbb{D}$ relation $f_\sigma(t) \le f_{\sigma'}(t)$ is verified.

The problem defined above can have instances for which the solution space is empty. This result is proved in the next proposition.

Proposition 3.3. *An order $\sigma \in \Pi(n)$, that minimizes the worst-case execution time f_σ for any $t \in \mathbb{D}$, cannot be always found.*

Proof. To prove it, we provide an example for which an order that minimizes f_σ does not exist.

Suppose a 2-output branching structure, such that the relations $\rho_1 + \tau_1 < \rho_2 + \tau_2$ and $\tau_1 > \tau_2$ are verified. The two possible branch orders are $\sigma_1 = \langle 1, 2\rangle$ and $\sigma_2 = \langle 2, 1\rangle$. It is easy to see that if $t \in [0, \tau_2 + \rho_2 - \tau_1[$ then $f_{\sigma_2}(t) \le f_{\sigma_1}(t)$, and, if $t \in [\tau_2 + \rho_2 - \tau_1, \rho_1 + \rho_2]$ then $f_{\sigma_1}(t) \le f_{\sigma_2}(t)$. Thus, for this particular case functions f_{σ_1} and f_{σ_2} can not be compared. We conclude that, in the general case, also, an order that minimizes the worst-case execution time is not always defined. □

Rather than attempting to compute a Pareto front, we modify the objective function as follows: we look for a branch prefetching order $\sigma \in \Pi(n)$, such that for any $\sigma' \in \Pi(n)$ we have $E[f_\sigma(t)] \le E[f_{\sigma'}(t)]$ assuming that the available prefetching time is uniformly distributed over $\mathbb{D}$.

Therefore, we have:

$$E[f_\sigma(t)] = \int_{\mathbb{D}} f_\sigma(t) \frac{1}{\sum_i \rho_i} dt = \left(\int_{\mathbb{D}} f_\sigma(t)\, dt\right) \frac{1}{\sum_i \rho_i}$$

Thus, the minimization of the worst-case execution time expectation is equivalent to the minimization of the area of the region bounded by the worst-

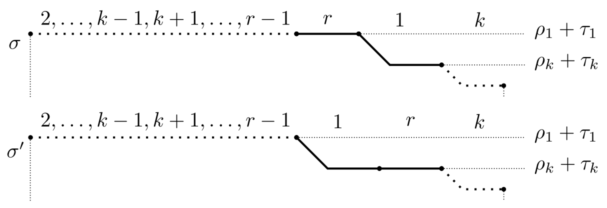

Fig. 3. Illustration of the contradiction from Proposition 3.5.

case execution time function. The integral of the worst-case execution time function over the range $[l_{k-1}, l_k[$ is equal to:

$$\int_{l_{k-1}}^{l_k} f_\sigma(t)\, dt = \Lambda_k \rho_k + \frac{1}{2} \max\left(0, \rho_{\sigma(k)} + \tau_{\sigma(k)} - \Lambda_k\right)^2$$

In what follows, we suppose that the branches are numbered in the decreasing order of $\tau_i + \rho_i$, that is $\rho_1 + \tau_1 \geq \rho_2 + \tau_2 \geq \ldots \geq \rho_n + \tau_n$.

Proposition 3.4. *Let σ be the optimal branch prefetching order. If in this order branches $p+1, \ldots, r$ are ordered before the branch p, then their order does not matter.*

Proof. Since $\rho_p + \tau_p$ is greater than or equal to $\rho_{p+1} + \tau_{p+1}, \ldots, \rho_r + \tau_r$ the worst-case execution time $f_\sigma(t)$ during the prefetch of branches $p+1, \ldots, r$ is equal to $\rho_p + \tau_p$. Therefore, the integral of $f_\sigma(t)$, over the interval when the branches $p+1, \ldots, r$ are prefetched, is constant and does not depend on their order. $\square$

Proposition 3.5. *If σ is an optimal branch prefetching order, then it has the following form: $\sigma = \langle r, \ldots, 1, \sigma' \rangle$, $r \geq 1$, where σ' is an optimal order over the branches $r+1 \ldots n$.*

Proof. Let r be the branch with the largest index ordered before the branch 1 in σ, that is, r is the branch with the lowest $\tau_i + \rho_i$ordered before the branch 1. Suppose that a branch k, $k \in [2, r-1]$, is ordered after the branch 1. By interchanging branch r with 1 (see Fig. 3) we obtain a new subset suborder that is strictly better than the initial order σ, which is in contradiction with the initial hypothesis which states that σ is an optimal order.

In the same manner, the proof is generalized to any sub-set of the branches in place of only one branch k. Also, we can state that the optimal sub-order σ' satisfies this proposition recursively. $\square$

We now are going to give an algorithm for the latter problem. It is based on finding a shortest path in a specific graph and uses the result of Proposition 3.5.

Definition 3.6. Let $G = (V, E, c)$ be a directed graph, where V is a set of nodes, E, a set of edges and $c : E \to \mathbb{R}$, a cost function that assigns a real, non-negative number to each edge of the graph. The graph G contains $n + 1$ nodes numbered from 0 to n. The meaning of the node i is that the branches $1, \ldots, i$ have been prefetched. For any i and j, the graph contains the edge (i, j) if and only if $i < j$. The value associated by the cost function c to the edge (i, j) is equal to the integral of function f_σ over the period of time when the branch order $j, j-1, \ldots, i+1$ is prefetched, taking into account that branches $1, \ldots, i$ have been already prefetched.

An example of such a graph is presented in Fig. 4. It corresponds to the graph built for a 4-output branching structure.

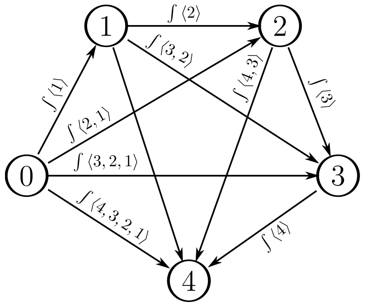
0
1
2
3
4
$\int \langle 1 \rangle$
$\int \langle 2 \rangle$
$\int \langle 3 \rangle$
$\int \langle 4 \rangle$
$\int \langle 2,1 \rangle$
$\int \langle 3,2 \rangle$
$\int \langle 4,3 \rangle$
$\int \langle 3,2,1 \rangle$
$\int \langle 4,3,2,1 \rangle$

Fig. 4. An example of graph G for a 4-output branching structure.

Let $P = \langle i_1 = 0, i_2, \ldots, i_p = n \rangle$ be a path from node 0 to node n in the graph G, defined above. The *branch prefetching order that corresponds to the path* P is built in the following manner: we begin by an empty order $\sigma = \emptyset$, for every $k = 2, \ldots, p$, the partial order $\langle i_k, i_k - 1, \ldots, i_{k-1} + 1 \rangle$ is appended to the end of σ, finally, σ will be the branch prefetching order that corresponds to path P.

Proposition 3.7. *Let $P = \langle i_1 = 0, i_2, \ldots, i_p = n \rangle$ be a path from node 0 to node n in the graph G and σ be the branch prefetching order that corresponds to P. Then, the cost of the path P is equal to the value of the integral of $f_\sigma(t)$ over $\mathbb{D}$, that is $\sum_{k=2}^{p} c(i_{k-1}, i_k) = \int_{\mathbb{D}} f_\sigma(t)\, dt$.*

Proof. The proof of this proposition relies on the following transformations:

$$\sum_{k=2}^{p} c(i_{k-1}, i_k) = \sum_{k=2}^{p} \int_{l_{i_{k-1}}}^{l_{i_k}} f_\sigma(t)\, dt = \int_{l_{i_1}}^{l_{i_p}} f_\sigma(t)\, dt = \int_{l_0}^{l_n} f_\sigma(t)\, dt$$

As $\mathbb{D} = [l_0, l_n]$, the last equality proves the proposition. □

The next proposition describes how from the graph G, defined above, the optimal branch prefetching order is found.

Proposition 3.8. *Let $G = (V, E, c)$ be a graph built as described in Definition 3.6, and, let $P = \langle i_1 = 0, i_2, \ldots, i_p = n\rangle$ be the shortest path from node 0 to node n. Then, the branch prefetching order σ that corresponds to path P is an optimal one.*

Proof. From the definition of the graph G and the propositions 3.4, 3.5, the set of all possible paths, from node 0 to node n, covers the set of all possible branch orders $\Pi(n)$. Since the values of a path and its corresponding branch prefetching order are the same, a shortest path P corresponds to a minimal valued branch prefetching order σ. □

4 Conclusion

This paper is a first examination of the problem of speculative data prefetching in dataflow applications restricted to a single n-way branching structure.

In a subsequent paper, we will address the issue of finding optimum data prefetch strategies in the more realistic settings where several branching structures are embedded in more complex dataflow graphs.

References

[1] G.G. Fursin, M.F.P. O'Boyle, and P.M.W. Knijnenburg. Evaluating iterative compilation. In *Proceedings of the 15th Workshop on Languages and Compilers for Parallel Computing (LCPC'02)*, pages 305–315, 2002.

[2] T. Goubier, F. Blanc, S. Louise, R. Sirdey, and V. David. Définition du Langage de Programmation ΣC. Technical Report DTSI/SARC/08-466/TG, CEA LIST, Saclay, 2008.

[3] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, Berlin, Germany, 2004.

[4] E. A. Lee and T. M. Parks. Dataflow process networks. In *Proceedings of the IEEE*, pages 773–799, 1995.