Microprocessors and Microsystems 38 (2014) 316–324

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Packet triggered prediction based task migration for network-on-chip

Tianzhou Chen, Weiwei Fu, Bin Xie *, Chao Wang

College of Computer Science and Technology, Zhejiang University, China

ARTICLE INFO

Article history:
Available online 12 November 2013

Keywords:
Network-on-chip
Task migration
Mapping

ABSTRACT

The development of IC technology makes Network-on-Chip (NoC) an attractive architecture for future massive parallel systems. Task migration optimize the overall communication performance of NoCs since the changing phases of execution make static task mapping insufficient. It is well-known that the communication behavior of many applications are predictable, which makes it feasible to use prediction to guide task migration. The triggering of activating a task migration is also important. In this paper, we first defined and analyzed predictabilities of applications, and then compared different ways of triggering for migration. We then modified the Genetic Algorithm (GA) based task remapping and proposed two other task migration algorithms: Simple Exchange (SE) and Benefit Assess (BA). A mechanism called node lock is also used to reduce unnecessary and costly migrations. Simulation results on real applications from PARSEC benchmark suites show that the SE, BA and GA algorithms can reduce 21.4%, 34.0% and 34.9% of number of hops, and 17.3%, 27.2% and 26.3% in terms of average latency respectively, compared with the system without task migration; BA and SE reduce 72.0% and 78.7% of migrations without significant performance degradation compared with GA, and the node lock mechanism can further remove 37.3% and 46.0% of migrations while achieving almost the same performance.

1. Introduction

Continuous improvements on semiconductor technology enable higher density of integration of chips and more processing units can be integrated inside a single chip. Traditional on-chip interconnection schemes face technology and design problems such as scalability, wire latency and power [1]. Packet switched Network-on-Chip (NoC) has been proposed for solving these problems [2], which is a more scalable on-chip communication substrate in terms of area, energy and design effort [3].

For many parallel applications, workloads normally experience dynamic variations [4]. So run-time management for efficient NoC design is a challenging task. Task scheduling and mapping are primary run-time resource management concerns. Due to significant vibration of application behaviors, initial task mapping seems to have limited effort on improving the performance, which makes some mapping approaches take behavior variation into consideration. Task migration is such a mechanism, which can migrate thread closer to the nodes which have larger amount of communication volume with the task. In NoC based systems specifically, the total hops to transmit the packets can be reduced. Since the packet latency and energy are closely related with the communication distance, the performance and power dissipation can be optimized. A task migration may trace time variation of traffic patterns caused by time variation of workloads, and accordingly transfer a task from one core to another. The triggering of a migration is an important factor for improving the performance of migration while keeping the algorithm simple.

It is well-known that the behavior of many applications are predictable during a short period. Chronicle and spatial locality of memory accesses is one of the most important predicable parts of application behaviors, which makes the traffic behavior of NoC predictable. In this paper, we first analyze the predictability of some real applications in a quantitative approach and discuss different methods to trigger a task migration. A remapping algorithm is then modified from Generic Algorithm (GA) to enable migration and other two simple migration algorithms are proposed based on these results. Finally, we propose a node lock mechanism to reduce unnecessary migrations. Experimental results show the effectiveness of these algorithms.

The rest of this paper is organized as follows. We review some related works in Section 2. The predictability of real applications and triggering for migration are discussed in Section 3. In Section 4, we proposed the task migration algorithms and the node lock mechanism. Experimental results are described in Section 5 and finally Section 6 concludes the paper.

2. Related works

Mapping and task migration mechanisms have been proposed in former works since 1980s. In [5], authors applied the heuristic

* Corresponding author.
E-mail address: xiebin@zju.edu.cn (B. Xie).

0141-9331/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.micpro.2013.11.004

Genetic Algorithm (GA) to mapping problems, and proposed a two-step GA based mapping algorithm. Dynamic remapping at run-time causes tasks to be migrated. Before NoC was proposed, some migration mechanisms for multi-computer distributed systems have been proposed [6,7]. However, migration mechanisms in multi-computer systems cannot be directly applied to NoC systems because of their limited chip area and computational capability.

There have been many researches on mapping schemes in NoC based systems. At the beginning, the mapping algorithms focused on the optimization of a single factor, such as the one mentioned in [8]. Then some multiple objective mapping algorithms were used. The mapping algorithms presented in [9,10] tried to reduce both latency and energy consumption at the same time. As the interference between different applications plays an important role during task mapping, [11,12] proposed some mapping algorithms which were optimized for concurrent multiple applications. Furthermore, some incremental mapping algorithms were presented in [13,14] to arrange the applications in a non-empty system. All the above mapping algorithms lack the ability to adjust the communication at runtime.

Recent researches on task migration in NoC systems mainly focused on details of the mechanism, including architectural support [17] and policies [18] of task migration and mechanisms to reduce the negative impact on performance caused by the migrations [15,16]. In [19] the authors approached to increase the timing predictability of multi-core architectures aiming at task migration in embedded environments by proposing a new microarchitecture supporting cache line migration. However, few of former researches focused on the triggering of initiating a migration (i.e. when and whether nodes are able to be migrated), which is also an important factor of migration mechanisms. We believe that proper migration triggering can significantly improve performance while keeping the complexity of the algorithm low.

The methods of using application behavior prediction as the guidelines for migration is not well studied by former researchers. Some works on the predictability of migration were related to the overhead of migration and cache behaviors [20,21], but not the network communication behavior itself. Therefore researches on the predictability of network behavior of real application for guiding the migration process and a appropriate trigger for initiating migrations in NoCs are necessitated.

3. Motivation

3.1. Predictability of network behavior

Behaviors of an application, such as memory access patterns, branches and multi-threading messages, depend on the application or the workload itself. Traffic patterns in NoCs are strongly dependant with the behaviors of applications. Although application behaviors change over time, it is possible to model behaviors such as L1 miss/hit and instruction cache access paths. Since on-chip traffic is usually caused by memory accesses and/or inter-core messages, the network traffic on chip can also be modeled and predicted.

To analyze and predict traffic patterns, a sample period has to be decided for information gathering. The method of counting a period (e.g. by cycle, by packet or others) and the duration of a period significantly affect on the accuracy of prediction. Actual accuracy of history-based prediction depends on the similarity of the application behaviors between the sample period and the period after task migrations. More formal definitions and descriptions of the problem are shown as follows.

3.2. Formalization of predictability

To formalize the predictability of an actual traffic, we first introduce following definitions.

Definition 1. A task communication graph *TCG* is described as a directed graph $TCG = (T, C)$, where T is the set of tasks currently running in the NoC-based system as nodes of the graph, C is the set of communication traffic between tasks as edges in the graph.

Definition 2. A period P is defined as a sample period, a specified period of time which is the basic unit of sampling and migration.

Definition 3. A period task communication graph *PTCG* is described as a graph $PTCG = (T, C, P)$, its components T and C have the same meaning with T and C in *TCG* except the value of elements in C is traffic during period P. We use $c^P_{a,b}$ to shortly describe the traffic from task a to task b in period P which is measured in packet. C_P is used to denote aggregate amount of traffic in *PTCG*.

Definition 4. The predictable traffic in certain period PT_P is the traffic in a sample period P which is similar to the traffic in last sample period P'. It is formally described as below:

$$PT_p = \sum_{a,b \in T} Min(c_{a,b}, c'_{a,b}) \tag{1}$$

where $c_{a,b}$ is the traffic from task a to task b in sample period P, $c'_{a,b}$ is the normalized traffic from task a to task b in period P'. Here $c'_{a,b}$ is formally described as:

$$c'_{a,b} = c^{p'}_{a,b} \times \frac{C_p}{C_{p'}} \tag{2}$$

Here C_P is the aggregate amount of traffic in period P, and $C_{p'}$ is the aggregate amount of traffic in previous period P'. $c^{p'}_{a,b}$ is the amount of traffic from a to b in period P'.

Definition 5. The predictability of an application *PoA* is the proportion of aggregate predictable traffic in all traffic. More formally:

$$PoA = \frac{\sum_p PT_p}{\sum_p C_p} \tag{3}$$

Obviously, range of the value of predictability is $[0, 1)$, as the amount of the predictable traffic in the first period is always 0.

3.3. Predictability analysis

Next we try to analyze the predictabilities of network traffic in real applications. We assume a target system with 64 cores running multi-threaded applications from PARSEC [23] benchmark suite with shared memory. The network traffic here is generated by tracing the memory activity of cores. A cycle-accurate full system simulator based on the M5 framework [22] is used to generate the trace data.

Fig. 1 shows the average predictabilities of all the applications using three methods to partition the periods. The horizontal coordinates presents thresholds for dividing periods (in number of packets or cycles). More details about these methods will be discussed later. In Fig. 1, the best predictability of the applications is about 77.4% and the worst predictability is about 65.9%. Though the percentage of predictable traffic is various with different methods and thresholds, we can see that the traffic is predictable, which provides great potential for run-time optimization of overall performance. The thresholds should be carefully set according to the