Microprocessors and Microsystems 39 (2015) 599–608

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

A MapReduce scratchpad memory for multi-core cloud computing applications

Christoforos Kachris [a,*], Georgios Ch. Sirakoulis [a], Dimitrios Soudris [b]

[a] Electrical & Computer Engineering Department, Democritus University of Thrace, Greece
[b] Electrical & Computer Engineering Department, National Technical University of Athens, Greece

ARTICLE INFO

Article history:
Available online 2 September 2015

Keywords:
Phoenix MapReduce
Scratchpad memory
Multi-core
Execution time
Energy consumption
Data-centers

ABSTRACT

Phoenix MapReduce is a multi-core programming framework that is used to automatically parallelize and schedule programs. This paper presents a novel scratchpad memory architecture that is used accelerate MapReduce applications by indexing and processing the key/value pairs. The proposed scratchpad memory scheme can be mapped onto programmable logic or multi-core processors chips as a coprocessor to accelerate MapReduce applications. The proposed architecture has been implemented in a Zynq FPGA with two embedded ARM cores. The performance evaluation shows that the proposed scheme can reduce up to 2.3× the execution time and up to 1.7× the energy consumption.

1. Introduction

As the number of processing cores per chip increases, so does the need for efficient programming frameworks that can automate the parallelization of the programs and can provide efficient synchronization and communication mechanisms. Traditional parallel programming frameworks are based either on shared memory frameworks (such as the OpenMP framework [1]) or on message passing protocols (such as MPI [2]). Multi-core programming based on shared memory frameworks, like OpenMP, is easier to program and debug than MPI and the code is easier to be maintained. However, OpenMP requires careful synchronization through locks and this framework is mostly used for loop parallelization. On the other hand, MPI that runs mostly on distributed memory architectures can be used in wider range of applications than OpenMP but it is harder to debug, it is error-prone due to manual management of data movement and the performance is limited by the communication network between the nodes.

In other distributed systems like data centers, the MapReduce framework is widely used for the efficient and automatic programming, scheduling and distribution of tasks to several processor nodes. The same notion which is used in the data center MapReduce application, can be used also for the programming of multi-core systems as in the case of the Phoenix MapReduce framework [3]. However, the overhead of task scheduling using the MapReduce framework may reduce the overall performance of the multi-core systems.

The memory hierarchies of these multi-core systems are based either on multi-level caches, or directly-addressable local scratchpad memories. Caches transparently decide on the placement of data and use coherence to support communication. However, caches lack deterministic response time (non-uniform memory access latency) and make it harder to the software to explicitly control and optimize data locality.

On the other hand, scratchpad memories offer predictable performance which is required in real-time applications. They also offer scalable general purpose performance by allowing explicit control and optimization of data placement and transfers. In the case of scratchpad, the interprocess communication is explicit meaning that the software (the application, compiler, or runtime system) is able to indicate physical placement or transfers. The main drawback of the scratchpad is that it reduces the programming efficiency, since extra effort must given to the synchronization and the consistency of the system.

This paper presents a novel special scratchpad memory architecture for accelerating applications that are based on the MapReduce framework. The proposed scheme is based on a special scratchpad memory architecture that is used to store, access and process the key/value pairs used in the MapReduce framework. The proposed scratchpad memory can be mapped within any multi-core platform in order to accelerate the processing of the applications that are based on the MapReduce framework. It can be mapped both onto high performance multi-core processors as Intel Xeon and onto low energy multi-core processors, targeting

* Corresponding author. Tel.: +30 25410 79547.
E-mail address: ckachris@ee.duth.gr (C. Kachris).

http://dx.doi.org/10.1016/j.micpro.2015.08.007
0141-9331/

microservers, such as ARM processors. The proposed scheme can reduce the execution time of the applications based on the MapReduce framework whether these applications are targeting cloud computing or any other application based on the MapReduce framework. The proposed scheme is prototyped on a heterogeneous FPGA SoC which combines two (2) ARM A9 processors and a programmable logic unit, in which the special scratchpad MapReduce memory is implemented. The main advantage of our scheme is that the proposed MapReduce scratchpad memory can be used extensively for most of the applications that are based on the MapReduce framework with a limited number of modifications on the applications.

Overall the main contributions of this paper are the followings:

- A novel MapReduce scratchpad memory architecture that is used for faster communication and processing of the key/value pairs used in the applications based on MapReduce framework.
- Efficient implementation and prototyping of the proposed architecture in a multi-core FPGA with two embedded ARM cores.
- Performance evaluation on the proposed scheme in the FPGA platform using typical cloud applications based on the Phoenix MapReduce framework showing that the proposed architecture can provide up to 2.3× speedup to the execution time and 1.7× lower energy consumption.

The paper is organized in the following way: Section 2 presents the related work on the use of scratchpad memories in multi-core systems. Furthermore it presents the related work on mapping of cloud applications in multi-core embedded processors and the research on execution of MapReduce applications on FPGAs. Section 3 presents the Phoenix MapReduce framework for the efficient programming of multi-core systems. Section 4 presents the proposed architecture with the special MapReduce scratchpad memory and the efficient implementation in the FPGA. Section 5 presents the performance evaluation in terms of execution time, footprint and energy consumption of the proposed scheme. Finally the conclusions of this work are drawn in Section 6.

2. Related work

In the past, several architectures have been proposed that include the use of scratchpad memories to accelerate the processing of the systems. Scratchpad memories offer the advantage of temporary storage of data with uniform memory access latencies (as opposed to cache memories that provide non-uniform memory access latencies).

Network processors have adopted the use of scratchpad memories to store information that needs to be accessed at low latencies [4]. In the case of packet processing at tens of Gbps, network processors cannot afford the non-uniform memory access latencies of caches. For example, in [5] a low latency scratchpad memory is proposed to provide fast access to IP lookups for the packet routing. In [6], a scratchpad memory is proposed that can be used not only to provide uniform and low latency communication with the processors, but it can be also used to easy the programming of multi-core platform.

Scratchpad memories have been also widely used in the graphics processors. For example, the Cell processor consists of 8 Synergistic Processing Unit (SPU), and each SPU has 256 KB of local storage that can be accessed directly by other processors though DMA [7]. The use of local storage instead of cache can provide faster explicit communication between the processors and the uniform low latency memory access that is critical for real-time graphics processing.

However until now, the use of scratchpad memories was only limited to the domain of application specific processors. In the domain of cloud computing, high performance general purpose processors are used that depend exclusively on cache memory hierarchies. In the last few years, there are efforts in the research community to provide specialized processors for different types of cloud computing applications. For example, Intel has recently unveiled the design of more energy-efficient servers based on embedded processors such as Atom, or using customized Pentium processors for low-end cloud computing applications. The term that has been widely accepted to describe these servers based on embedded processors is *microservers* [8].

In [9], a performance evaluation study has been presented between high performance server cores (e.g. Intel Xeon processors) with low power general purpose cores (e.g. Intel Atom processors). The comparison has shown that low power general purpose cores can achieve better energy efficiency in the domain of web search applications. In [10], it was shown that a Fast Array of Wimpy Nodes (FAWN) that consists of a large number of low-power embedded processor can achieve high energy efficiency in cluster computing applications. In [11], it is shown through detailed measurements that the energy-efficiency ratio of the ARM processor against the Intel workstation is approximately 1.3 in Web server application. However, none of the multi-core embedded processors have been specialized for cloud computing applications such as applications based on MapReduce.

On the other hand, there are also some relevant works that have been focused on the hardware acceleration of the MapReduce applications with FPGAs. In [12], it is presented a MapReduce framework for FPGA but the proposed scheme is especially implemented as a custom design that is used to implement only the RankBoost application entirely on an FPGA. Both of the Map and Reduce tasks for the specific application have been mapped to configurable logic and thus a new design has to be implemented for a different application. In [13] a MapReduce Framework on FPGA accelerated hardware is presented where a cluster of worker nodes is designed for MapReduce framework, and each worker node consists of commodity hardware and special hardware. However, once again each module is specialized for a specific MapReduce application.

LINQits [14] accelerates a domain-specific query language called LINQ. LINQ is designed to operate upon data sets and exposes first-class language constructs for manipulating collections using query operators. However, existing applications have to be rewritten with LINQ in order to benefit extensively from the hardware acceleration which may be tedious and error-prone. Dryad [15] has been also proposed, which is a general-purpose distributed execution engine for coarse-grain data-parallel applications. Dryad runs the application by executing the vertices of a graph on a set of available computers, communicating through files, TCP pipes, and shared-memory FIFOs. However, again the existing applications based on MapReduce need to be re-written using the Dryad API.

The MapReduce scratchpad memory that is presented in this paper has the advantage of being a specialized co-processor for MapReduce applications but it can be utilized by a wide range of MapReduce applications without major modifications of the original MapReduce code.

3. The Phoenix MapReduce framework

One of the most widely used frameworks that are hosted in the data centers is the MapReduce framework. MapReduce is a programming framework for processing and generating large data sets [16,17]. Users specify a **Map** function that processes a *key/value*