Scalable Computing Software Laboratory Technical Report

Department of Computer Science

Illinois Institute of Technology

The Memory Sluice Gate Theory: Have We Found a Solution for Memory Wall?

Xian-He Sun
Department of Computer Science
Illinois Institute of Technology
sun@iit.edu

Yu-Hang Liu
Department of Computer Science
Illinois Institute of Technology
yuhang.liu@ iit.edu

March, 2016

Technical Report No. IIT/CS-SCS2016-1

http://www.cs.iit.edu
10 West 31st Street, Chicago, IL 60616

The Memory Sluice Gate Theory:

Have We Found a Solution for Memory Wall?

Xian-He Sun and Yu-Hang Liu

Computer Science Department, Illinois Institute of Technology

{sun, yuhang.liu}@iit.edu

Abstract: The memory-wall problem is a long standing issue facing the computing community. Many believe the memory-wall problem can only be solved with new memory technologies that improve memory device hardware performance. In this research, we propose an architectural solution, the memory Sluice Gate Theory, for solving the memory-wall problem. The focus of the Sluice Gate Theory is not on hardware peak performance, but the achieved memory stall time. In other words, the focus is not on removing memory wall, but on mitigating the impact of memory wall. In this theory, data access concurrency in addition to data access locality plays a vital role. Based on Sluice Gate Theory, a memory system is built to transfer data and to mask the performance gap between CPU and memory devices during the data transfer process. Sluice gates are designed to control data transfer at each memory layer (sluice stage) dynamically, and a global control algorithm, named layered performance matching, is developed to match the data transfer request/supply at each memory layer (sluice stage) thus matching the overall performance between the CPU and memory system. Formal theoretical analyses are given to show, with sufficient data access concurrency and dynamic hardware support, the memory wall impact can be fully removed. Experimental testing is conducted which confirm the theoretical findings. Sluice Gate Theory calls to fully investigate and utilize FPGA and memory concurrency technologies to eliminate the memory wall impact.

Keywords: Memory Wall, Data Access Concurrency, Data Access Locality, Performance Matching

1. Introduction

Memory wall is a term introduced by Wulf and Makee in 1994 [1][2] to describe the increasingly large performance gap between CPU and memory devices. The memory wall problem, in turn, refers to the relatively slow memory performance forming a wall between CPU and memory. This wall causes CPUs to stall while waiting for data and slows down the speed of computing. Memory wall is a well-recognized issue. Tremendous efforts have been done on improving memory technologies to catch up the advancement of microprocessor technologies. However, the gap between CPU and memory devices continues to enlarge during the last twenty years. Combined with other improvements, doubling the density of transistors has been doubling the computing speed, but not the data access speed. Memory speed only increases nine percent per year on average during the last three decades while CPU speed doubles every eighteen months. With the multi-core/many core technology, this gap could be even larger. Memory wall stems from the large access time ratio among the memory layers [1][2] and the constraint of pin bandwidth [3][4]. According to the prediction of ITRS [5], limited by the physical metal properties, the number of pins will grow slowly, 2/3 of the pins are for power and ground, and only 1/3 are for data transfer. Many-core technologies and data-intensive applications have put even more pressure on memory systems in recent years. In many-core processors, many cores share and thus contend for the same memory system. In data-intensive applications, data access and management is the major concern. Data access speed becomes the primary performance bottleneck of computer system performance. The memory wall problem is a premier performance issue facing the computing community.