CG-Cell: An NPB Benchmark Implementation on Cell Broadband Engine

Dong Li, Song Huang, and Kirk Cameron

Department of Computer Science
Virginia Tech
{lid,huangs,Cameron}@cs.vt.edu

Abstract. The NAS Conjugate Gradient (CG) benchmark is an important scientific kernel used to evaluate machine performance and compare characteristics of different programming models. CG represents a computation and communication paradigm for sparse linear algebra, which is common in scientific fields. In this paper, we present the porting, performance optimization and evaluation of CG on Cell Broadband Engine (CBE). CBE, a heterogeneous multi-core processor with SIMD accelerators, is gaining attention and being deployed on supercomputers and high-end server architectures. We take advantages of CBE's particular architecture to optimize the performance of CG. We also quantify these optimizations and assess their impact. In addition, by exploring distributed nature of CBE, we present trade-off between parallelization and serialization, and Cell-specific data scheduling in its memory hierarchy. Our final result shows that the CG-Cell can achieve more than 4 times speedup over the performance of single comparable PowerPC Processor.

1 Introduction

The NAS Conjugate Gradient (CG) benchmark is often used to evaluate computer machine performance and compare characteristics of different programming models. It uses a conjugate gradient method to compute an approximation to the smallest eigenvalue of a large, sparse, symmetric positive definite matrix. CG is one of the memory intensive benchmark in NAS kernels and is typical of unstructured grid computations, which tests irregular long distance communication by employing unstructured matrix vector multiplication [1].

The recent developments in semiconductor technology lead to the debuts of multi-core processors in the computing industry, such as IBM's Cell, Sun Microsystems' Niagara and AMD's Opteron. The multi-core helps multi-programmed workloads which could contain a mix of independent sequential tasks. It presents a chance to study new or existing parallel programming models. However many questions for programming on multi-core are still open, such as how to efficiently handle specific communication and computation patterns. Cell Broadband Engine (CBE), developed jointly by Sony, Toshiba and IBM, is a new heterogeneous multi-core platform. The Cell is a general-purpose microprocessor which offers a rich palette of thread-level and data-level parallelization options to the programmer.

S. Rao et al. (Eds.): ICDCN 2008, LNCS 4904, pp. 263–273, 2008.

The NPB CG presents a communication and computing pattern seen in sparse linear algebra [11]. It would be helpful to see how CG can be implemented on Cell, the new distributed and parallel scenario. Implementing CG on this special multi-core is a challenging topic. Firstly the essences of heterogeneous cores require careful consideration of task schedules while improving execution efficiency. Secondly Cell architecture shows an explicit and special memory hierarchy to users. On one hand, it presents a shared/global view of data to its nine cores through main memory. On the other hand, it presents a distributed view of data since eight of its nine cores have local memory. To achieve good performance on Cell, people need to carefully handle the data distribution and locality of references. Thirdly, due to its unusual architecture, unconventional Cell-specific code optimization approaches should be considered.

In this paper, we present how we solve the above problems in the implement CG on a real Cell multi-core. The main contributions of this paper include:

- We port NAS CG onto CBE. We present an example of how the communication and computation pattern of CG could be implemented on Cell and how we take advantage of the Cell's distributed multi-core nature. The result shows that CBE as a new architecture has good potential for this programming pattern with limited working data sets.
- We quantify Cell-specific code optimizations and assess their impacts using CG. These quantified results are beneficial for application developments on the Cell.
- We explore the parallelization methods on the Cell. We find that sometimes merely exposing task level parallelism is insufficient for high performance computing. We also find that parallelization on the Cell does not always mean performance speedup. Other factors, like overhead for creating threads and Direct Memory Access (DMA) communication, should be considerable when making decisions on parallelization.

The rest of this paper is organized as follows. Section 2 summarizes related works on programming support for Cell and studies of implementing CG using different programming models. Section 3 introduces kernel CG algorithm and section 4 outlines the Cell architecture. Section 5 presents step by step our CG porting and optimization process. In Section 6, we present the performance of parallel CG on Cell. In the end, we conclude the paper in Section 7.

2 Related Work

As a brand-new multi-core architecture, Cell has attracted many attentions in various research communities. Some researches focus on how to develop applications and speedup their performances. These include exploring new programming models and developing compiler supports. Other researches focus on analyzing the performance of the processor in terms of chip architecture.

Pieter et. al. [4] presents a simple and flexible programming model for Cell. It requires the input application source code to follow a certain paradigm. Then based on the paradigm annotation, a source to source compiler builds a task dependency graph of