Charm++ Simplifies Programming for the Cell Processor

David Kunznan, Mengbin Zheng, Eric Bohm, Jim Phillips, Lexmikant V. Kale

Introduction

Charm++ is a high-level, object-oriented parallel programming language. It provides abstractions and system services that hide many details of parallel computing, allowing users to focus on the algorithmic aspects of their applications. Charm++ is designed to be scalable, efficient, and portable, making it suitable for a wide range of applications.

The Cell processor, developed by IBM, is a unique architecture that combines traditional processor elements (PPE) with special purpose engine (SPE) cores to provide high performance computing capabilities. Charm++ has been extended to support programming on the Cell Broadband Engine (Cell/BCE), which is a subset of the Cell processor.

The Cell processor's architecture is designed to execute multiple tasks in parallel, and Charm++ is well-suited to exploit this parallelism. In this paper, we explore the current progress and future directions of Charm++ on the Cell processor, highlighting the benefits and challenges of this approach.

The Cell Processor

The Cell processor consists of a single PowerPC processor (PPE) and eight Streaming Processors (SPEs) on a single chip. The PPE is a 64-bit PowerPC 440 processor with a clock speed of 1.5GHz. The SPEs are specialized cores designed for high performance computing tasks.

The SPEs can execute up to 16 instructions per cycle, which is significantly faster than the PPE. However, the SPEs are limited to executing only specific types of code, known as `chares`, which are compiled specifically for the SPEs. Chares are essentially functions that run on the SPEs and communicate with the PPE.

Charm++ provides a programming model that allows developers to write applications in a high-level language and then automatically generate the necessary code for each processing element. This makes it easier to develop and maintain parallel applications.

Charm++ Code Example ("Hello")

```
// Initialize the Cell
Cell::init();

// Create a chare to say hello
Chare chare;
char *hello = "Hello from element 9 on processor 1 (num:3)"
chare.sayHello(hello);
```

Offload API

The Offload API (OA) provides a mechanism for the Charm++ runtime system to offload computations to the Cell processor. This allows developers to write applications in a high-level language and then automatically generate the necessary code for each processing element. The API includes methods for creating chares, specifying which elements they will run on, and performing work requests.

```
CProxy_Hello chare;
chare = CProxy_Hello::ckNew(nElements);
```

Core Idea

Charm++ has been extended to support the Cell processor, allowing developers to write high performance computing applications in a high-level language and then automatically generate the necessary code for each processing element. This makes it easier to develop and maintain parallel applications.

```
Hello from element 9 on processor 1 (num:3)
```

Suitability to the Cell Processor

Charm++ is well-suited to the Cell processor due to its high-level programming model and the ability to automatically generate code for each processing element. This makes it easier to develop and maintain parallel applications.

```
# endif //__HELLO_SHARED_H__
```

References / Related Papers


Future Work / Directions

- Performance tuning of Charm++ applications on the Cell processor
- Development of new Charm++ features to support the Cell processor
- Exploration of new applications for the Cell processor

```
# define FUNC_SAYHI 1
```

Charm++ and the Offload API

The Charm++ runtime system uses the Offload API to distribute computations across the Cell processor. This allows developers to write high performance computing applications in a high-level language and then automatically generate the necessary code for each processing element.

```
CkPrintf(CkArgMsg *m);
```

Charm++ Code and Class Diagram (Calculation Code)

```
Hello
```

Images Sources:

- Charm++ Code and Class Diagram (Calculation Code)
- Charm++ Code Example ("Hello")
- Future Work / Directions
- References / Related Papers

For more information, please visit: http://charm.cs.uiuc.edu