IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 10, October 2016

A Survey: Compiler and Simulators used for Optimization of Code for Multiprocessors

H T Prakash[1], Dr. M Srinivas[2], Chetan R[3]

Research Scholar, Dravidian University, Kuppam, Andhra Pradesh, India [1]

Research Guide, Dravidian University, Kuppam, Andhra Pradesh, India [2]

Assistant Professor, SJBIT, Bengaluru, Karnataka, India [3]

Abstract: Optimization is a program transformation technique, which tries to improve the code by making it consume fewer resources (i.e. CPU, Memory) and deliver high speed. Another important topic is high performance computing, which is used for parallel programming to make the execution of the programs faster and increase the performance of the system. This paper describes the various compilers and simulation tools that can be used for optimization of code in multiprocessor systems.

Keywords: Optimization, high performance computing, parallel programming.

I. INTRODUCTION

Embedded systems are becoming more and more important. The products containing embedded systems span from day-to-day household and consumer products, such as digital TVs, mobile phones, and automobiles, to industrial devices and equipment, including, for example, robots, aviation equipment, and high end military and scientific devices such as aircraft.

Previously, because embedded systems were highly limited in computational capability, memory size, and power consumption, much research was dedicated to making the best use of limited system resources.

Multi-core devices, which incorporate two or more processors on the same integrated circuits, are becoming increasingly relevant to the design and implementation of embedded systems. In multi-core platforms, carefully managing communication and synchronization among different cores is important to achieve efficient implementations. Two or more processing cores sharing the same system bus and memory bandwidth limit the achievable performance improvements. The ability of multi-core processors to increase application performance depends on the use of multiple concurrent tasks within applications. Therefore, if code is written in a form that facilitates decomposition into concurrent tasks, the multi-core technologies can be exploited more effectively. Dataflow-based languages are suitable for such decomposition into concurrent tasks, particularly in the broad domain of digital signal processing (DSP) applications.

Dataflow representations of DSP software have been explored actively since the 1980s. Such representations have proved to be useful in identifying bottlenecks in DSP algorithms, improving the efficiency of the computations, and designing appropriate hardware for implementing the algorithms.

Dataflow descriptions have been used in a wide range of DSP application areas, such as multimedia processing, and wireless communications. Among various forms of dataflow modeling, synchronous dataflow (SDF) is geared towards static scheduling of computational modules, which improves system performance and predictability. However, many DSP applications do not fully conform to the restrictions of SDF modeling. More general dataflow models, such as CAL, have been developed to describe dynamically-structured DSP applications. Such generalized models can express dynamically changing functionality, but lose the powerful static scheduling capabilities provided by SDF.

The next section is going to give details about the simulators and the compilers that can be used for optimization of code.

II. RELATED WORK

A. Open Research Compiler (ORC):

This is the outcome of ORC[1] project initiated by Intel Microprocessor Research Labs (MRL). To provide a leading open source IPF (IA-64) compiler infrastructure to the compiler and architecture research community.

- To encourage compiler and architecture research.
- To minimize the resource investments for university groups.
- Performance better than existing IPF open source compilers.
- Fair comparison on a common infrastructure.

C/C++ and Fortran compilers targeting IPF are present in the ORC. This can be used for Interprocedural analysis, Loopnest optimization, Code generation. This can be used in the Linux platform. The figure 1 shows the flow of ORC.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 10, October 2016

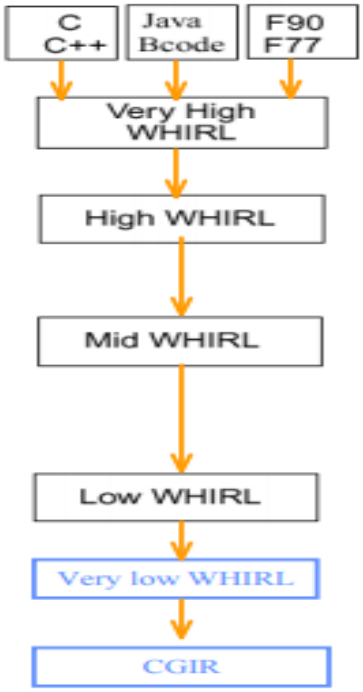

Fig 1: Flow of ORC Compiler

The ORC uses the region based compilation technique.

B. The Collective:
In the Collective system architecture, virtual appliances, and their updated versions, are deposited in repositories. Individual computers run a universal appliance receiver that retrieves the latest copies of virtual machines from repositories upon request. In other words, the computers operate as a cache of appliances. The system uses a number of optimizations to minimize the cost of the storage and transfer of appliances. This approach allows a small number of professional staff to create fully tested, integrated environments that are made available quickly to all users anywhere on the network. Efficient migration of appliances. X86 appliances, complete with operating systems, application programs, and possibly user data, can be very large. We found that the storage and transfer of appliances can be effectively optimized using the techniques of caching, demand paging, memory ballooning to reduce the memory state, and copy-on-write disks to capture changes. The time to transfer an appliance on a DSL link (384 kbps) is typically less than 20 minutes.

- **Virtual appliance networks.** Generalizing the concept of virtual appliances to include a virtual network enables the encapsulation of network management knowledge and sets of related services.
- **The CVL (Collective Virtual appliance Language).** A language for describing composition of virtual appliances to create virtual networks of appliances; the language uses the concept of inheritance to allow appliances be individually configured and customized appliances while retaining the ability to be upgraded automatically.
- **Livewire: An intrusion detection system for virtual machines**. Used to build an intrusion detection system (IDS) that is both difficult to evade and difficult to attack. Like a host-based IDS, it has excellect visibility since it can access all the states of the computer being watched. Like a network-based IDS, it is not vulnerable to being disabled by the attacker.
- **Terra: A virtual machine-based platform for trusted computing**. Terra allows applications to run in an "open box" VM with the semantics of a modern open platform, or in a "closed box" VM with those of dedicated, tamper-resistant hardware. It has attestation primitives to cryptographically identify the contents of closed-box VMs to remote parties and showed how to implement them efficiently.
- **Remote timing attacks**. It demonstrated the first remote timing attack where a private key can be extracted from a web server. Patches to eliminate such vulnerability were developed and applied to the Open SSL library.
- **CRED: a dynamic buffer overrun detector**. CRED (C Range Error Detector) that finds all buffer overrun attacks as it directly checks for the bounds of memory accesses. Unlike the original referent-object based bounds-checking technique, CRED does not break existing code because it uses a novel solution to support program manipulation of out-of-bounds addresses. CRED is implemented as an extension of the GNU C compiler version 3.3.1, and has been tested on over 20 open-source programs, comprising over 1.2 million lines of C code.

C. Multifacet Gems:

The heart of GEMS is the Ruby memory system simulator. As illustrated in Figure 2, Leverages the power of Virtutech Simics to simulate a Sparc multiprocessor system. This enables the simulation of commercial software such as database systems running on the Solaris operating system. By off-loading the correctness requirement to Simics, our "timing-first" simulation can focus on accurate performance modeling rather than correctness details.

- The GEMS Opal module provides a detailed, out-of-order processor model. Opal is flexible and highly configurable to model different branch predictors, issue-widths, execution resources, and etc.
- The GEMS Ruby module provides a detailed memory system simulator. It can simulate a wide variety of memory hierarchies and systems ranging from broadcast-based SMP to a hierarchical directory-based Multiple-CMP system.
- The GEMS SLICC (Specification Language including Cache Coherence) language simplifies and speeds the development of differing memory hierarchies and cache coherence protocols. Systems are specified in a table-driven language which abstracts away much of the manual labor involved with C/C++ coding.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 10, October 2016

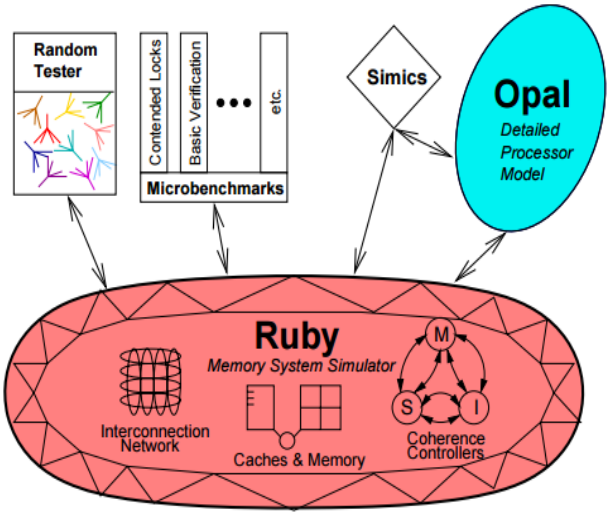

Figure2: Gems Architecture

D. ML-RSIMML-RSIM:
It is an execution-driven computer system simulator that combines detailed models of modern computer hardware, including the I/O subsystem, with a fully-functional operating system kernel. These features make the simulation environment particular attractive for studies involving applications with significant I/O or operating system activity. ML-RSIM executes static SPARC V8 binaries. The Lamix system call interface is compatible with Solaris 2.8. In general, applications compiled for ML-RSIM can execute on native Sparc/Solaris systems without modification. No special libraries or include files are required to compile applications for the simulator.

Features
The simulator includes models of the following hardware components:

- dynamically scheduled SPARC V8 compatible processor
- Level-1 instruction and data cache and a unified level-2 cache
- Split-transaction coherent system bus
- Multi-bank memory controller with SDRAM or RAMBUS memory
- Real-time clock
- SCSI controller (based on Adaptec AIC7770) with SCSI bus
- SCSI hard disk with cache, prefetching and optional write buffering

E. Dinero IV Trace-Driven Uniprocessor Cache Simulator:
Dinero IV is a cache simulator for memory reference traces. It includes the following major changes over Dinero III.

- subroutine-callable interface in addition to trace-reading program
- simulation of multi-level caches
- simulation of dissimilar I and D caches
- better performance, especially for highly associative caches
- classification of compulsory, capacity, and conflict misses
- support for multiple input formats
- cleaned up and modernized code, improved portability

Some deep-seated limitations:

- Dinero IV is not a timing simulator. There is no notion of simulated time or cycles, only references.
- Dinero IV is not a functional simulator. Data & instructions do not move in and out of the caches; in fact they don't exist! The primary result of simulation with Dinero IV is hit and miss information.
- Dinero IV isn't multi-threaded. If you have a multiprocessor with enough memory, you can run multiple independent simulations concurrently.

The basic idea is to simulate a memory hierarchy consisting of various caches connected as one or more trees, with reference sources (the processors) at the leaves and a memory at each root. The various parameters of each cache can be set separately (architecture, policy, statistics). During initialization, the configuration to be simulated is built up, one cache at a time, starting with each memory as a special case. After initialization, each reference is fed to the appropriate top-level cache by a single simple function call. Lower levels of the hierarchy are handled automatically.

F. HASE & SimJava:
HASE and SimJava are systems developed at the University of Edinburgh to support, through simulation, the visualisation of activities taking place inside computers as they execute programs.

HASE is a Hierarchical computer Architecture design and Simulation Environment which allows for the rapid development and exploration of computer architectures at multiple levels of abstraction, encompassing both hardware and software. HASE produces a simulation trace file which can be used to animate the on-screen display of the model so as to show data movements, parameter value updates, state changes, etc.

HASE is available free of charge to academic institutions, and to other organisations that wish to use it for internal, non-commercial, non-profit educational and/or research purposes. Commercial licences can be negotiated. HASE-III is a Java version of HASE that runs on Linux, Windows and MAC OSX. The most recent version (3.7), released in July 2016, can be downloaded from Download HASE-III.

Several Computer Architecture Simulation Models for use with HASE-III, appropriate for use as lecture demonstrations or for practical exercises, are also available for download as tar or zip files.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 10, October 2016

SimJava is a process based discrete event simulation package for Java based on Hase++ (the behavioural description language used in HASE), with animation facilities. SimJava has become successful in its own right but from a HASE perspective it offers an alternative way of creating models, rather than a way of presenting existing HASE models over the WWW.

III. CONCLUSION

This paper is going to give the details of the various simulators their features and limitations. There are more tools that can be used. Among these some tools work on UNIX environment and some work on Windows environment. Depending upon the user requirement they can used these simulators for analyzing their codes and evaluating the performance of the code.

REFERENCES

[1] http://ipf-orc.sourceforge.net/ORC-documentation.htm
[2] http://suif.stanford.edu/collective/
[3] http://research.cs.wisc.edu/gems/
[4] http://www.cs.utah.edu/~lambert/mlrsim/description.php
[5] http://www.cs.wisc.edu/~markhill/DineroIV
[6] http://www.icsa.inf.ed.ac.uk/research/groups/hase/