

面向瘦客户端平台可重定向的全系统模拟器

曲宁, 赵雨来, 管雪涛, 程旭

(北京大学微处理器研究开发中心, 北京, 100871)

北京大学计算机科学技术系, 北京, 100871)

{quning, zhaoyulai, guanxuetao, chengxu}@mprc.pku.edu.cn

摘要:

高端嵌入式系统的硬件设计具有很强的多样性, 其上的应用程序也跨越嵌入式和中低端桌面领域。瘦客户端系统是客户/服务器和浏览器/服务器环境中一种典型的高端嵌入式系统, 能够整合本地和远程的计算资源。本文提出了面向瘦客户端系统而设计的 Unichos 全系统模拟器。针对嵌入式系统硬件平台多样化的特点, Unichos 模拟器结合了可重定向的指令模板以及可扩展的设备模型, 使其成为具有良好可重定向性的全系统模拟器。Unichos 模拟器采用面向对象的方法对整个目标硬件系统建模, 其上可以运行无修改的 Linux 2.4 内核以及瘦客户端系统中所有应用程序。同时, Unichos 模拟器能够灵活的支持不同粒度的性能模拟。最后本文通过两个实验案例进一步说明了 Unichos 模拟器在性能评测过程中的优势。

关键词: 计算机系统结构, 瘦客户端, 全系统模拟, 可重定向, 性能评测

中图分类号: TP316 **文献标识码:** A

A Retargetable Full System Simulator for Thin Client Platform

Ning Qu, Yulai Zhao, Xuetao Guan, Xu Cheng

(Microprocessor Research and Development Center, Peking University, Beijing, 100871)

Department of Computer Science and Technology, Peking University, Beijing, 100871)

{quning, zhaoyulai, guanxuetao, chengxu}@mprc.pku.edu.cn

Abstract:

There are varieties of hardware designs in high-end embedded systems, and the applications on it cover embedded areas and low-end desktop areas. Thin client is a typical high-embedded system in client/server and browser/server environment, which combines local and remote computing resources. This paper presents the design and implementation of Unichos, a full system simulator for thin client platform. Unichos focuses on retargetability for more architectures of thin client platforms by combining the retargetable instruction template and the extensible device model. Unichos models the complete target hardware system in object-oriented structure, and supports the unmodified Linux 2.4 kernel and all of the applications for thin client. In addition, Unichos can flexibly provide support for performance simulation in different level of detail. Finally, we introduce two case studies based on Unichos, which demonstrate the advantages of Unichos in performance evaluation.

Keywords: computer architecture, thin client, full system simulation, retargetability, performance evaluation

1. INTRODUCTION

There are varieties of hardware designs in high-end embedded systems, and the applications on it cover embedded areas and low-end desktop areas. Thin client is a typical high-embedded system in client/server and browser/server environment, which combines local and remote computing resources. In addition, applications on thin client platform rely heavily on the support of the network and graphics devices and the support of operating system.

To thoroughly study and analysis the characterization of applications and OS on such kind of platform, we develop a

new full system simulator, Unichos, which targets on the PKU-Unity network computer, a general thin client based on PKUnity SoC (System-On-a-Chip). Unichos focuses on the retargetability by combining the characteristics of retargetable instruction template in SimpleScalar [1][2] and extensible device model in Bochs [3], which make it suitable for more kinds of thin client platforms. Since the typical applications (such as browsers and remote desktop graphics client) spend a significant portion of execution time in the operating system, Unichos models the complete target hardware system faithfully to run an unmodified Linux 2.4 on network computer. Moreover, Unichos sup-

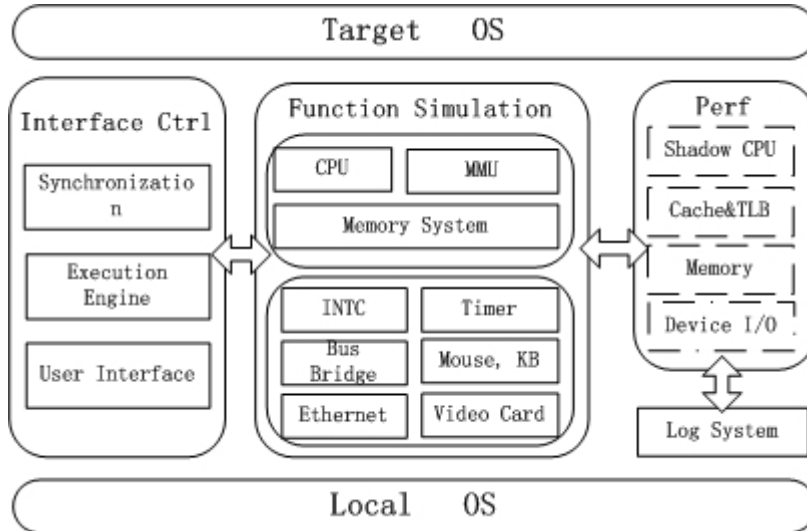


Figure 1. The architecture of Unichos

ports runtime simulation status switching between the functional simulation and performance simulation and runtime configuration according to different simulation requirements. The complete simulation and evaluation environment provided by Unichos has been widely used in the development and research on thin client platform of PKUnity network computer.

The rest of this paper is organized as follows. Section 2 gives an overview of the related works on full system simulation. Section 3 details the design and implementation of Unichos. Section 4 demonstrates the advantages of Unichos by introducing two case studies.

2. RELATED WORKS

SimpleScalar [1][2] toolkit provides several application-level simulators, which make up of a retargetable high performance evaluation platform widely used in system research area. SimpleScalar 4.0 provides *ss-os* [1], a full system simulator for ARM architecture, which integrates Bochs device model into SimpleScalar. To some extent, *ss-os* is similar to the design of Unichos; however, up to now it has not been released yet. As a full system simulator, Unichos integrates instruction template of SimpleScalar into an object-oriented design, which has the characteristics of well-defined abstraction and modularity.

Bochs [3] is a free and complete x86 PC simulator, which can support many devices in PC and can run popular x86 OSs. Bochs, however, has no support for performance simulation and has limitation in some important hardware simulation. For example, it only models a NE2000 Ethernet card without DMA function. In contrast, Unichos simulates a complete modern thin client, PKUnity network computer, and provides necessary

timing information for performance simulation.

SimOS [4][5] is one of the earliest full system simulators and places emphases on simulation performance. However, SimOS does not support graphics. Its network implementation relies on a separate proxy process, which may have side effects on network performance when communicating with the outside world. Later derivatives of SimOS, including SimOS-Alpha, SimOS-PPC, Linux/SimOS [6] and PHARMsim [7], do not include a timing model for the network.

In recent few years, there exists some other well-known full system simulators, such as Simics [8][9], M5 [10] and etc. Simics supports most of the mainstream architectures and simulates the function for graphics and network. It provides uniform extension mechanism to create custom hardware models, whereas adding a new ISA support in Simics is still non-trivial [11]. M5 implements a simulation system targeting network intensive workloads, so it provides detailed performance model of I/O subsystem. However, M5 does not support graphics applications and only runs commercial operating system on alpha architecture.

The primary goal of Unichos is to provide an open, modular and retargetable full system simulator with complete function support for thin client platform.

3. THE UNICHOS FULL SYSTEM SIMULATOR

3.1 The Overview of Unichos

The architecture of Unichos is shown in Figure 1. Unichos contains three components: the interface control, the functional simulation and the performance support. These three parts are loosely coupled and interact with each other by well-defined

interfaces. In terms of modularity and extensibility, all of the hardware models in both the functional simulation and the performance support are designed completely in object-oriented structure.

The Interface control is to control the process of simulator’s booting, configuring and running. In more detail, execution engine schedules the running of functional simulation, user interface interacts with user, and synchronization mechanism harmonizes between the functional simulation and the interface control. The functional simulation simulates the basic logic function of target hardware system, which follows the Von Neumann model, including an instruction level processor, a memory system with MMU, and I/O devices. The performance support is designed to provide necessary timing information when Unichos enters performance simulation status. It supports a series of detailed hardware models including detailed processor model, multiple levels of cache and TLB hierarchy, and the detailed model for important I/O devices such as network DMA transfer. These detailed hardware models are called shadow models because they are dynamically created and are only active during performance simulation. We will explain the details of shadow model in section 3.4. Besides, the performance support provides trace capture and runtime data analysis.

The hierarchy structure of main objects in the functional simulation is shown in Figure 2. In Unichos, both the functional simulation and the performance support are designed in a hierarchy object-oriented structure, which brings several benefits. With the help of C++ abstraction and protecting mechanism, Unichos limits inter-object communication to well-defined interfaces, which aids in maintaining realistic simulation models. A set of clear interfaces for a simulation object make it easier to modify the behavior of a component without affecting unrelated parts. For example, since different Ethernet card simulations share the same basic Ethernet card model, they export the same set of interfaces. With the help of

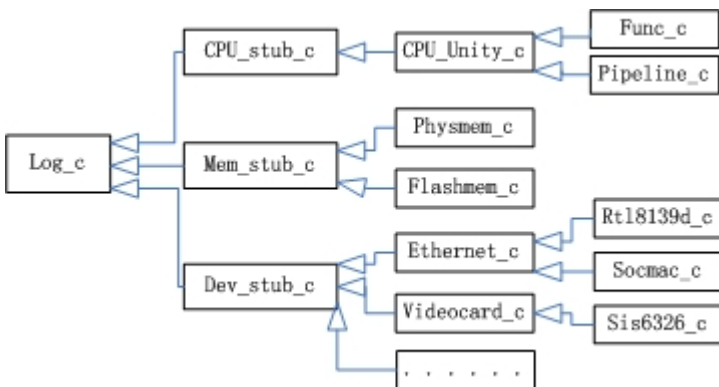


Figure 2. Hierarchy object-oriented structure in Unichos

C++ inheritance mechanism, the realization for same kind of hardware component in different level of details derives from the same base class, which shares many common behaviors and inner data structures. The object-oriented design is the key feature to implement the runtime switch between functional models and shadow models, e.g., switching between an instruction-level functional processor model and a fine-grained pipeline-level timing processor model. In addition, the object-oriented design of Unichos makes it easier to extend to other architectures and devices and checkpoint a simulation state by serialization.

3.2 The Functional simulation

This section introduces the features in the design and implementation of the functional simulation in Unichos.

Processor model: Processor model is the core of computer simulation. Retargetability is regarded as one of the most important aspects in the simulation of ISA, until now the widely used instruction template of SimpleScalar is one of key solutions that achieve this goal. However, SimpleScalar only supports the application-level simulation. Unichos integrates the instruction template mechanism into processor model and enhances it in following two aspects. First, it adds complete support for privileged instructions, such as Cache and TLB management, special instructions to access user space from system status. Second, it adds support for exceptions and interrupts handling. The processor model in the functional simulation simulates a simple in-order instruction-level functional processor, which is similar to sim-safe.

Memory system model: Memory system in the functional simulation is constructed by the MMU (Memory Management Unit) and physical memory. In order to simulate the virtual memory hardware precisely, Unichos implements the full logic function of MMU in PKUnity SoC. To translate the virtual addresses into physical address, MMU directly accesses main memory to get page table information. When encountering the permission violation or page fault, it raises an exception and transfers control back to processor. Besides, MMU is also in charge of maintaining the access and dirty bits of page table entry as well as dispatching access to main memory or I/O devices. There is also an optional and simplified cache and TLB function to help accelerate the simulation speed of memory system model.

I/O device models: Unichos models and implements almost all the I/O devices in the target hardware system, including

interrupt controller, timer, PCI Bridge, Ethernet card, video card, keyboard, mouse, etc. In the following, we will introduce some features of I/O device models in Unichos.

Bus model: In the functional simulation, bus model connects all the components on it together, including devices and memory system. Its primary function is to pass requests directly among different components in a simplified way. Because most of the computer systems have more than one kind of bus, the bus model is hierarchically designed. Different buses are connected by bus bridge model, which is only in charge of passing requests, translating address of the requests and providing information the bridge specification requires. For PKUnity network computer, we implements two-level bus structure connected with an AHB-PCI bridge. The bridge simulates functions of PCI 2.2 specification to support PCI device scanning and auto-configuration during Linux booting.

Video card model: The Basic video card model simulates the function of the standard VGA (text) mode and VESA (graphics) mode for a general video card, which makes Unichos support graphics applications on the operating system. For the SIS6326 video card used in PKUnity network computer, Unichos also implements a SiS6326 Video card model, which inherits the basic video card model and extends the function support to some specific SiS6326 2D acceleration and other extension operations.

Ethernet card model: The Ethernet card model in Unichos supports connectivity to real hosts outside the simulation environment. This is important because thin client frequently interacts with servers and completes most of its jobs with the help of servers on network. Because we focus on the analysis of the

thin client instead of the server, a traditional software simulator will degrade the performance of a server running in it dramatically, which means that we cannot run both client and server in the same simulator (similar to SimOS or M5).

The connectivity of the Ethernet card model is achieved by using raw socket mechanism to bind the Ethernet cards in local host. The filter of the raw socket is initialized to accept raw packets only from a new IP address assigned to the simulated Ethernet card. The Ethernet card model sends and receives the simulated Ethernet packets directly via the raw socket as shown in Figure 3. The basic Ethernet card model implements the control of raw socket and provides the basic function support for programmed I/O and DMA transfer. Based on it, Unichos implements two DMA-based Ethernet card models, which completely simulate the logic design of Realtek 8139D (a commercial PCI Ethernet card) and PKUnity SoC Ethernet (an AHB Ethernet controller). The main difference in implementing these two models: the function of receiving in Realtek 8139D is based on fixed length ring buffer, while the function of sending and receiving in SoC Ethernet controller is based on more popular scatter-gather linked buffer.

3.3 The Performance Support

The basic mechanism of performance support in Unichos is as follows: during functional simulation besides the simple function hardware models, there are still some stubs for those deeply simplified or no function hardware, such as cache, TLB. When Unichos enters performance simulation status, a series of detailed hardware models are created, and then the hardware state is synchronized between the functional model and detailed model. Finally, these detailed models replace the simple functional one or the stubs. Next, Unichos behaves as an event-driven simulator with timing information. When exiting performance simulation status, Unichos synchronizes the hardware states again and activates the functional models and stubs as before. Because both the functional model and the detailed model are derived from the same abstract base class as shown in Figure 2, the process of the state synchronization is easy to implement. We introduce some implementation details of the performance support in Unichos in the following,

Processor model: The shadow processor model is also based on instruction template which makes it easier to extend from any existing detailed processor model (e.g., sim-outorder). For PKUnity SoC, we have implemented two detailed processor models: a single-issue, 5-stage pipeline processor model and a

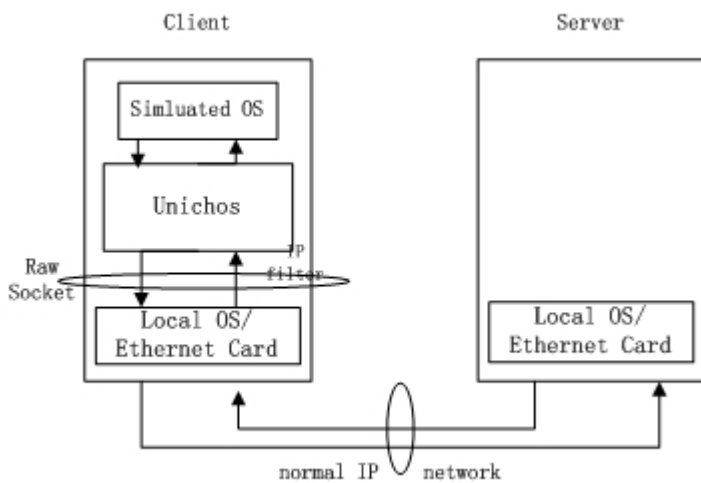


Figure 3. Communication between simulator and real server

single issue, 8-stage pipeline processor model.

Memory system model: The memory system model in Unichos is fully event-driven which allows flexible and accurate modeling of timing. Unichos supports multiple levels of cache and TLB hierarchy. Hence, we can flexibly parameterize to model caches and TLB with different organization and timing during run time.

Device I/O model: The I/O device is hard to model with accurate timing information. In the common case, most of the device I/O accesses do not happen frequently and have very little cost. However, Unichos is designed for thin client and many applications on it heavily rely on network, we pay attention to the accuracy of the network model. We detail the timing of DMA transfer and take the cost on the real IP network in consideration. The current model allows us to perform thorough study on the behavior of network workloads.

Until now, Unichos has implemented a set of detailed hardware models, which are necessary for studying the characteristics of the thin client system.

3.4 Other Characteristics

More detailed simulation will make running entire application take too much time, so we make Unichos support runtime switching among different simulation modes. Unichos also provides flexible control on sampling an application, which is very useful for understanding application behavior because application's execution phases usually have different characteristics. For instance, sampling a workload consists of executing it in functional simulation status for a given number of simulated instructions, and then switching to performance simulation status to warm up the detailed hardware models, such as cache and TLB, and finally starting to collect the performance simulation results statistically and periodically.

Besides, Unichos implements inner debugger and GDB stub. Inner debugger is useful for assembler level debugging. From connecting to a gdb process through remote serial protocol, Unichos provides a more powerful source level debugging for operating system kernel. These debugging features have been invaluable in making full system simulation work.

4. EXPERIMENTS

In this section, we introduce two case studies based on Unichos, which served as a performance simulator.

4.1 Case study I

In the research of characterizing the d-TLB behavior of typical applications on thin client [12], the selected benchmarks

include SPECint 2000 [13], browser (Konqueror/Embedded), JVM (Kaffe), multimedia benchmarks (Mibench [14]) and RDP graphics clients (Rdesktop). These applications either need the function of graphics and network, or have the characteristics of multi-process, so only full system simulators like Unichos can capture their behavior and evaluate performance with reasonable accuracy. In the experiments, Unichos is used to evaluate d-TLB miss rate and its contribution to the performance penalty. We also use Unichos to generate data access traces of operating system and applications for detailed analysis, such as the impacts that the characterization of multi-process brings.

4.2 Case study II

In another study, we design a separated sequential cache dedicated to accommodate "sequential data" which has strong spatial locality but poor temporal locality (like multimedia stream data). This study has two motivations: the first phase is based on trace driven simulation to study the characteristic of data access pattern in network and streaming applications, the second is based on execute-driven to evaluate the impacts on the cache miss rate and performance when using the separated sequential cache. The benchmarks include SPECint 2000 [13], Netperf, IO-Zone, Mplayer] and Rdesktop. All of these benchmarks have a large amount of data accesses in file reading/writing or network data processing, which shows the characteristics of sequential access. In this work, Unichos provides full support on trace-generation and performance simulation, as well as the evaluation for both operating system and these benchmarks.

5. CONCLUSION

This paper presents the design and implementation of Unichos, a full system simulator for thin client platform. Unichos helps properly analyze and evaluate future architecture along with full simulation of OS for thin client platform. Beside, Unichos has implemented the functional CPU model for ARM architecture, moreover the supports for x86 architecture and WinCE operating system are still under development. The re-targetability of Unichos makes it a powerful and flexible simulation environment for studying all aspects of other embedded platforms.

Up to now, the performance model of Unichos is not a full-fledged one yet, and there are many important extensions to be incorporated into it, including the detailed bus model, the memory controller model etc. E.g., without a memory system that models contention [15], the timings reported may be overly

optimistic. So enhancing the performance model of Unichos is one of the most important works in the future.

6. REFERENCE

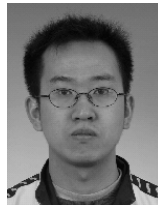
- [1] T. M. Austin. "SimpleScalar Version 4.0 Release", Proc. of the IEEE International Solid State Conference, Washington, D.C., USA, 2001.
- [2] T. Austin, E. Larson, and D. Ernst. "SimpleScalar: An Infrastructure for Computer System Modeling", IEEE Computer, vol. 35, No. 22, pp.59-67, Feb. 2002.
- [3] Bochs. <http://bochs.sourceforge.net>. 2005.
- [4] M. Rosenblum, S. A. Herrod, E. Witchel, and A. Gupta. "Complete Computer System Simulation: The SimOS Approach", IEEE Parallel and Distributed Technology: Systems and Applications, Vol.3, No.1, pp.34-43, 1995.
- [5] M. Rosenblum, E. Bugnion, S. Devine, and S. A. Herrod. "Using SimOS Machine Simulator to Study Complex Computer Systems", Modeling and Computer Simulation, Vol.7, No.1, pp.78-103, 1997.
- [6] C. Won, B. Lee, and C. Yu. "Linux/SimOS -A Simulation Environment for Evaluating High-speed Communication Systems", Proc. of the International Conference on Parallel Processing (ICPP'02), Vancouver, Canada, pp.193-202, 2002.
- [7] PHARMSim. <http://www.cdl.edu/pharmsim/index.html>.
- [8] P. Magnusson, M. Christianson, J. Eskilson, and et al. "Simics: A Full System Simulation Platform", IEEE Computer, Vol.35, No.1, pp.50-58, 2002.
- [9] "Introduction to The Simics Full-system Simulator without Equal", Virtutech White Paper, 2002.
- [10] N. L. Binkert, E. G. Hallnor, and S. K. Reinhardt. "Network-oriented Full-system Simulation Using M5", Proc. of the Sixth Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW), Anaheim, California, USA, pp.36-41, 2003.
- [11] M. Ekman. "Architectural Simulators for Computer Systems", Technical Report, 2003.
- [12] N. Qu, Peng Yuan, X. Guan, and X. Cheng. "Characterizing the D-TLB Behavior of Typical Applications on Network Computer", Journal (Natural Science) of Peking University (JNSPU), Vol.43, No.1, 2007. (in Chinese)
- [13] J. L. Henning. "SPEC CPU2000: Measuring CPU Performance in the New Millennium", IEEE Computer, Vol.33, No.7, pp.28-35, 2000.
- [14] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin,

T. Mudge, R. B. Brown. "MiBench: A Free, Commercially Representative Embedded Benchmark Suite", Proc. of the IEEE 4th Annual Workshop on Workload Characterization, Austin, Texas, USA, pp.1-12, 2001.

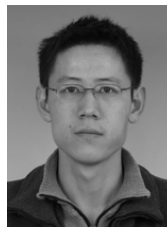
- [15] L. Barroso, K. Gharachorloo, and F. Bugnion. "Memory System Characterization of Commercial Workloads", Proc. of the Twenty-Fifth International Symposium on Computer Architecture, Barcelona, Spain, pp.3-14, 1998.



Qu Ning: Born in 1978. He received B.S. degree in computer science from Peking University in 2001. Currently he is a Ph.D candidate in Microprocessor Research and Development Center of Peking University. His research interests include operating system, computer architecture.



Zhao Yulai: Born in 1979. He received B.S. degree in computer science from Peking University in 2002. From 2002, he is a Ph.D candidate in Microprocessor Research and Development Center of Peking University. His research interests include computer architecture, particularly in the optimizations of high-end microprocessors for energy efficiency.



Guan Xuetao: Born in 1974. He received B.S. degree in computer science from Shandong University in 1997 and Ph.D degree in computer science from Peking University. Currently he is lecturer in Microprocessor Research and Development Center of Peking University. His research interests include operating system, embedded system, system software and system chips.



Cheng Xu: Born in 1964. He is a professor and doctoral supervisor of Peking University. He is also the member of VLSI Design Expert Group of National 863 Program. He received Ph.D. degree from Harbin Institute of Technology in 1994 and worked as post-doctor in Computer Science Department of Peking University in 1996. His main research interests are high performance microprocessor design, SoC design, embedded system, instruction level parallelism, compiler optimization and hardware software co-design.