

# Unichos: A Full System Simulator for Thin Client Platform

Ning Qu                      Yulai Zhao                      Xuetao Guan                      Xu Cheng

Microprocessor Research and Development Center, Peking University  
Computer Science and Technology Department, Peking University  
Peking University, Yi He Yuan Street No.5, Beijing, China  
86-10-62756231

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

## ABSTRACT

Thin client is a kind of interactive and graphics device in client/server and browser/server environment, which combines local and remote computing resources. The applications on the thin client (e.g., browsers) often heavily rely on the support of operating system and the functionalities of network and graphics. Hence, traditional performance evaluation methods, such as instrumentation and application-level simulation, cannot help due to their inherent limitation. This paper presents the design and implementation of Unichos, a full system simulator for thin client platform. Unichos models the complete target hardware system in object-oriented structure, and supports the unmodified Linux 2.4 kernel and graphics and network applications. Unichos is the first full system simulator, which focuses on portability for more architectures of thin client platforms by combining the retargetable instruction template and the extensible device model. Finally, we present the Unichos performance under detailed simulation status and introduce two case studies which demonstrate the advantages of Unichos for performance evaluation.

## Categories and Subject Descriptors

C.4 [Performance of Systems]: Measurement techniques, Modeling techniques.

## General Terms

Performance, Design.

## Keywords

computer architecture, thin client, full system simulation, performance evaluation.

## 1. INTRODUCTION

Thin client is a kind of interactive and graphics device in client/server and browser/server computing environment. An important characteristic of the thin client is that it relies heavily on the support of the network and graphics devices. Related

research has shown that omitting the behavior of the OS introduces errors as high as 100% for the SPECint 2000 benchmarks [1], thus the widely used application-level simulators are not suitable for evaluating a thin client platform. On the other hand, profiling and binary instrumentation cannot catch all the behaviors of applications and operating system, to some extent, may have great side effects on the final results. In other words, they are insufficient for quantitatively evaluating the impact of OS, network and graphics on the performance of the thin client (e.g., to find out which component is the performance bottleneck of the system).

To address this situation, we develop a new full system simulator, Unichos, which targets on the PKUnity [2] network computer, one type of general thin client platform based on PKUnity SoC [2]. Since the network and graphics 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 is the first full system simulator, which focuses on the portability by combining the characteristics of retargetable instruction template in SimpleScalar [3], [4] and extensible device model in Bochs [5], which make it suitable for more architectures of thin client platforms. Unichos also proposes the shadow models to separate the performance support from the functional simulation. In this sense, Unichos is treated as a timing-directed simulator [6]. The shadow models mean all of the detailed hardware models in the performance support, which are created dynamically and provide timing information during performance simulation. Unichos also has several other key features: 1) It models target hardware system in object-oriented structure for modularity; 2) It supports runtime simulation status switching between the functional simulation and performance simulation. Moreover, it supports runtime configuration to allow user to specify the detailed parameters according to different simulation requirements. The complete evaluation and analysis 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 presents the Unichos performance and demonstrates the advantages of Unichos by introducing two case studies.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'07, March 11-15, 2007, Seoul, Korea.

Copyright 2007 ACM 1-59593-480-4/07/0003...\$5.00.

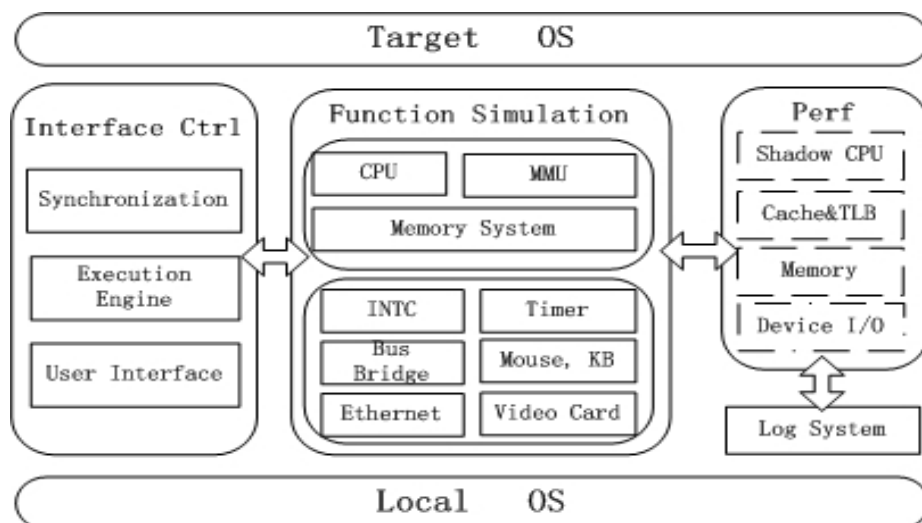


Figure 1. The architecture of Unichos

## 2. RELATED WORKS

SimpleScalar [3] [4] 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 [7], 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 [5] 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 [8] [9] 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 [10] [11], SimOS-PPC [12], Linux/ SimOS [13] and PHARMSim [14], 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 [15] [16], M5 [17], TFSim [6], 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 nontrivial [18]. 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. TFSim is based on Simics and decouples simulator organization into

separate function simulator and time simulator. The organization of TFSim is more loosely coupled than Unichos and it neglects to model network I/O in detail. The primary goal of Unichos is to provide an open, modular and retargetable full system simulator with complete support of graphics and network for thin client platform.

## 3. 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 which 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.

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 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 Configuration Process

To model a complex thin client system, Unichos requires a powerful method to specify the simulation configuration. Unichos has two types of configuration files for different purposes: config-arch and config-micro.

Config-arch describes the hardware system architecture and organization of thin client. Hence, config-arch is only loaded at the boot time of Unichos and is used to construct a complete hardware system for functional simulation, including processor, memory system, I/O devices and the connection among them.

Config-micro is designed for specifying the parameter and structure of micro-architecture when switching to performance simulation status. During the running of Unichos, user can switch between performance simulation status and functional simulation status through user interface or a particular instruction embedded in application binary. Under performance simulation status, user can choose to reuse the performance objects created before. Otherwise, a new config-micro is loaded to provide the parameters for creating new detailed hardware models dynamically together with all the timing information, such as cache and TLB structures and timing information.

Supporting two sets of configuration files clearly classifies the configuration information into difference purposes and provides additional flexibility because Unichos only loads this information when necessary.

### 3.3 Functional simulation of Unichos

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 [2], 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 [2] 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.

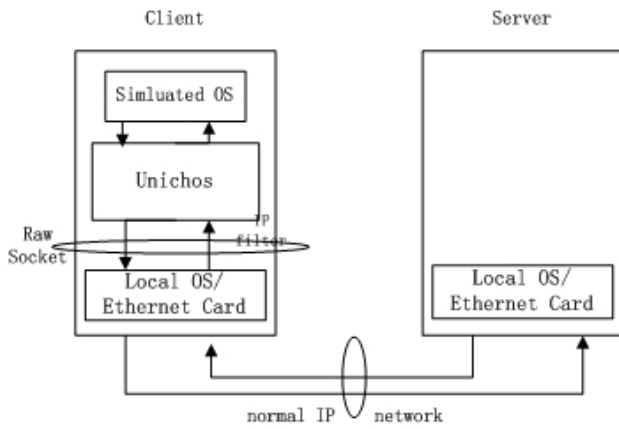
**I/O device models:** Unichos models and implements almost all the I/O devices in the target hardware system, including interrupt controller, timer, PCI Bridge, Ethernet card, video card, keyboard, mouse, etc. In the following, we will introduce some features of I/O device models in Unichos.

**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 2. 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



**Figure 2. Network communication mechanism in Unichos**

functional 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: Realtek 8139D is based on fixed length ring buffer, while SoC Ethernet controller is based on more popular scatter-gather linked buffers.

### 3.4 Performance Support of Unichos

The basic idea of the shadow model mechanism is as follows: during functional simulation with the simple function hardware models, there are still some stubs for those deeply simplified or even no function hardware, such as cache, TLB and bus. 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 (e.g., instruction-level and pipeline processor models). Finally, these detailed models replace the simple functional one or the stubs and Unichos behaves as an event-driven simulator with timing information for performance simulation. When exiting performance simulation, Unichos synchronizes the hardware states again and activates the functional models and stubs as before. These detailed hardware models in the performance support are called shadow models because in most cases they are shadowed by their function models. Next we describe some implementation details in the performance support.

**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 [2]: a single-issue, 5-stage pipeline processor model and a 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 happen seldom and have very little impacts. However, Unichos is designed for thin client in which applications 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 network in consideration. The current model allows us to study on the behavior of network workloads thoroughly.

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

### 3.5 Other Characteristics

More detailed simulation will make running entire application take too much time, so Unichos supports 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 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 first present the Unichos performance under performance simulation status, and then introduce two case studies in which Unichos serves as a performance simulator.

### 4.1 Unichos Performance

**Table I Unichos Performance**

Workload	MIPS	Workload	MIPS
164.zip	1.488	255.vortex	1.254
175.vpr	1.725	256.bzip2	1.698
176.gcc	1.333	300.twolf	1.581
181.mcf	1.354	Kaffe-fft	1.503
186.crafty	1.301	Kaffe-heapsort	1.438
197.parser	1.447	mplayer-ffmpeg	1.609
252.eon	1.709	mplayer-mpeg12	1.679
253.perlbmk	1.132	Rdesktop	1.483
254.gap	1.557	Konqueror	1.454

Table I summarizes the Unichos performance under highly detailed simulation status, which models PKUnity SoC into a detailed single issue, 5-stage pipeline CPU, separate 8K data and instruction cache, separate 64 entries data and instruction TLB, a detailed memory controller and DMA controllers. We execute the simulations on a Dell GX240 workstation equipped with a 1.6 GHz Pentium 4 CPU, 512MB SDRAM. The workloads we use in the comparison are the followings: SPECint 2000 [19]; Kaffe as an embedded JVM; Mplayer with ffmpeg and mpeg12 decoder; Rdesktop as a graphical remote desktop client; Konqueror/Embedded, a real web browser.

For the detailed simulation of PKUnity SoC, the performance is obviously very low from 1.132 MIPS to 1.725 MIPS, but still acceptable and useful to study the SoC platform more thoroughly.

## 4.2 Case Studies of Performance Simulation

**Case study I:** In the research of characterizing the d-TLB behavior of typical applications on thin client [20], 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 to analysis the impacts that the characterization of multi-process brings.

**Case study II:** In another study [21], we design a separated sequential cache dedicated to accommodate “sequential data” which has strong spatial locality but poor temporal locality (like multimedia stream data). In this work, Unichos provides full support on trace-generation and performance simulation of network and streaming applications and OS.

## 5. CONCLUSION

This paper presents the design and implementation of Unichos, a full system simulator for thin client platform. Nowadays, Unichos has been effectively used in three areas: architectural evaluation, system software development, workload behavior characterization. Unichos also has implemented the functional CPU model for ARM architecture, and the support for x86 architecture and WinCE operating system is still under development. Since we plan to make a public release of Unichos at the end of 2006, its portability to other SoC architectures will be a powerful and flexible simulation environment for studying other 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. Enhancing the performance model of Unichos is one of the most important work in the future.

## 6. REFERENCES

- [1] Cain, H. W., Lepak, K. M., Schwartz, B. A., and Lipasti, M. H. Precise and Accurate Processor Simulation. In *Workshop on Computer Architecture Evaluation Using Commercial Workloads, HPCA*, Feb. 2002.
- [2] MPRC. *PKUnity SoC Architecture and System Programming Guide*. Technical Report, Microprocessor Research and Development Center, Peking University, 2006.
- [3] Austin, T. M. SimpleScalar Version 4.0 Release. In *Proceedings of the IEEE International Solid State Conference*, 2001.
- [4] Austin, T., Larson, E., and Ernst, D. SimpleScalar: An Infrastructure for Computer System Modeling. *IEEE Computer*, vol. 35, Feb. 2002.
- [5] Bochs. <http://bochs.sourceforge.net>.
- [6] Mauer, C., Hill, M., and Wood, D. Full System Timing-First Simulation. In *Proceedings of the 2002 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 2002.
- [7] SimpleScalar. <http://www.simplescalar.com>.
- [8] Rosenblum, M., Herrod, S. A., Witchel, E., and Gupta, A. Complete Computer System Simulation: The SimOS Approach. *IEEE Parallel and Distributed Technology: Systems and Applications*, 3, 1 (1995), 34–43.
- [9] Rosenblum, M., Bugnion, E., Devine, S., and Herrod, S. A. Using SimOS Machine Simulator to Study Complex Computer Systems. *Modeling and Computer Simulation*, 7, 1 (1997), 78–103.
- [10] Barroso, L., Gharachorloo, K., and Bugnion, F. Memory System Characterization of Commercial Workloads. In *Proceedings of the Twenty-Fifth International Symposium on Computer Architecture*, 1998.
- [11] SimOS-Alpha. <http://research.compaq.com/wrl/projects/simos/simos.html>.
- [12] SimOS-PPC. <http://www.cs.utexas/users/cart/simos>.
- [13] Won, C., Lee, B., and Yu, C. Linux/SimOS -A Simulation Environment for Evaluating High-speed Communication Systems. In *Proceedings of the International Conference on Parallel Processing (ICPP'02)*, 2002.
- [14] PHARMSim. <http://www.cdl.edu/pharmsim/index.html>.
- [15] Magnusson, P., Christianson, M., Eskilson, J., and et al. Simics: A Full System Simulation Platform. *IEEE Computer*, 35, 1 (Feb. 2002), 50–58.
- [16] Introduction to The Simics Full-system Simulator without Equal. *Virtutech White Paper*, 2002.
- [17] Binkert, N. L., Hallnor, E. G., and Reinhardt, S. K. Network-oriented Full-system Simulation Using M5. In *The Sixth Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW)*, Feb. 2003.
- [18] Ekman, M. Architectural Simulators for Computer Systems. 2003.
- [19] Henning, J. L. SPEC CPU2000: Measuring CPU Performance in the New Millennium. *IEEE Computer*, 33, 7 (2000), 28–35.
- [20] Qu, N., Yuan, P., Guan, X., and Cheng, X. Characterizing the D-TLB Behavior of Typical Applications on Network Computer. *Journal (Natural Science) of Peking University (JNSPU)*, 43, 1 (2007).
- [21] Gou, X., Liu, S., Qu, N., and Li, X. *Split Sequential Data Cache*. Technical Report, Microprocessor Research and Development Center, Peking University, 2006