ETRI Electronics and Telecommunications Research Institute

연구산출물 블라인드 처리 가이드라인

한국전자통신연구원 공개채용은「평등한 기회, 공정한 과정을 위한 공공기관 블라인드 채용」을 따르고 있습니다. 이에 지원서 작성시 첨부하는 연구산출물 증빙자료 블라인드 처리방법에 대해 다음과 같이 안내드리며, 반드시 유의사항을 숙지하시어 전형과정에서 불이익을 받지 않도록 유의 바랍니다.

1. 논문(학위논문 초록 포함) 실적의 블라인드 처리 가이드(샘플 2쪽~5쪽 참조)

가. (블라인드 처리) 저자 소속 등 인적사항

- o 지원자 본인뿐만 아니라 모든 저자의 소속, e-mail(출신학교 노출 가능)에 대해 블라인드 처리

  ※ 교신저자 등 별도로 소속, 연락처가 기재된 것은 모두 블라인드 처리

- o 저널에 따라 페이지 상/하단에 저자정보(인적사항, 성명)이 나타나 있는 부분은 모두 블라인드 처리

나. (블라인드 처리) 사사문구(Acknowledgments)

다. (블라인드 처리) 학위논문 초록 내 학교 워터마크(watermark)

라. (블라인드 처리) 첨부파일 명칭은 게재논문(1), 게재논문(2)과 같이 변경

마. (블라인드 미처리) 저널명, 논문명 및 주요 Article info(게재권호, ISSN 등)

2. 특허 실적의 블라인드 처리 가이드(샘플 6쪽~7쪽 참조)

가. (블라인드 처리) 특허권자, 발명자 인적사항

- o 지원자 본인뿐만 아니라 모든 공동발명자 주소, 소속(출신학교 노출 가능)에 대해 블라인드 처리

나. (블라인드 처리) 사사문구(Acknowledgments)

다. (블라인드 처리) 첨부파일 명칭은 특허(1), 특허(2)과 같이 변경

라. (블라인드 미처리) 특허번호, 등록일자 및 발명의 명칭 등 특허 기본정보

3. 기타 참고사항

가. 학술대회, 프로그램(SW), 저작권 등 연구산출물 및 자격증: 논문, 특허에 관한 블라인드 처리 가이드를 동일하게 적용

나. 응시지원서 제출 시 함께 첨부하는 서류 중 다음에 해당하는 경우 블라인드 미처리 대상

- o 취업지원대상자 증명서(보훈대상자) 및 복지카드(장애)(해당자만 제출)

  ※ 3. 나항 서류는 인사부서에서만 내용을 열람하며, 심사위원에게 비공개

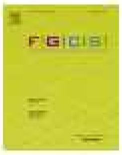
블라인드 처리 샘플(논문)

Future Generation Computer Systems 70 (2017) 26–41

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs



Effects of dynamic isolation for full virtualized RTOS and GPOS guests

[redacted] 저자 이름: 블라인드처리

[redacted]

저자 소속: 블라인드처리

HIGHLIGHTS

- We examine and analyze how a RTOS VM and a GPOS VM interact and influence each other.
- We analyze the explicit and implicit effects of dynamic isolation for vCPUs.
- The dynamic isolation shows low scheduling delay of RTOS and high throughput of GPOS.
- All of the proposed concepts are implemented on a full-fledged hypervisor.

ARTICLE INFO

Article history:
Received 14 May 2016
Received in revised form
25 October 2016
Accepted 17 December 2016
Available online 23 December 2016

Keywords:
Embedded virtualization
Dynamic isolation
Mixed criticality system
vCPU scheduling
Full virtualization

ABSTRACT

Industrial systems currently include not only control processing with real-time operating system (RTOS) but also information processing with general-purpose operating system (GPOS). Multicore-based virtualization is an attractive option to provide consolidated environment when GPOS and RTOS are put in service on a single hardware platform. Researches on this technology have predominantly focused on the schedulability of RTOS virtual machines (VMs) by completely dedicated physical-CPUs (pCPUs) but have rarely considered parallelism or the throughput of the GPOS. However, it is also important that the multicore-based hypervisor adaptively selects pCPU assignment policy to efficiently manage resources in modern industrial systems. In this paper, we report our study on the effects of dynamic isolation when two mixed criticality systems are working on one platform. Based on our investigation of mutual interferences between RTOS VMs and GPOS VMs, we found explicit effects of dynamic isolation by special events. While maintaining low RTOS VMs scheduling latency, a hypervisor should manage pCPUs assignment by event-driven and threshold-based strategies to improve the throughput of GPOS VMs. Furthermore, we deal with implicit negative effects of dynamic isolation caused by the synchronization inside a GPOS VM, then propose a process of urgent boosting with dynamic isolation. All our methods are implemented in a real hypervisor, KVM. In experimental evaluation with benchmarks and an automotive digital cluster application, we analyzed that proposed dynamic isolation guarantees soft real-time operations for RTOS tasks while improving the throughput of GPOS tasks on a virtualized multicore system.

1. Introduction

Traditional industrial computers consist of control processing software for simple missions. In recent years, industrial systems (e.g., consumer electronics, automobile, aeronautic sectors, smart phone, factory automation and grid computing) have been launched with more powerful devices, to interface to more networks and sensing devices. Moreover, a larger variety of application software is now required, with different levels of quality and service. This trend has also increased the number and volume of electronic units, as well as their power requirements. Thus, the job of software now includes not only hardware control but also information processing for sensing devices. In general, real-time operating systems (RTOS) are used as control processing software, since its tasks are mostly time-critical, whereas information processing software can be written on top of general purpose operating systems (GPOS) to maximize throughput. Those systems can be consolidated into a single system by multicore hardware and virtualization techniques. Some research can be found in industrial domains that require RTOS and GPOS applications to be simultaneously executed on a single multicore-based virtualized platform [1,2]. As illustrated in Fig. 1,

[redacted]

http://dx.doi.org/10.1016/j.future.2016.12.020

저자 이름, 저자 관련 정보(E-mail, 연락처): 블라인드처리

저자 이름: 블라인드처리

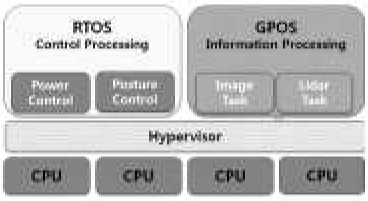


Fig. 1. A RTOS and a GPOS VM for a virtualized Dual OS Platform for a multicore processor.

the sample system contains a hardware control component and data processing system realized by a virtualization layer on a multicore processor. This is an attractive option which enables more sophisticated industrial tasks to be accomplished, reducing overall system complexity and power consumption. However, applications for RTOS and GPOS have different purposes. The real-time property is the most important when RTOS virtual machines are managed on a hypervisor. On the other hand, GPOS VMs require high throughput, in order to efficiently utilize limited resources of the embedded target systems. Thus, the manner in which a hypervisor controls its VMs for multi-OS has a crucial influence on the outcome of given missions.

In a general multicore-based virtual execution environment, load balancing is applied for efficient utilization of the multicore-resource, while all virtual-CPUs (vCPUs) share whole physical-CPUs (pCPUs). However, when GPOS and RTOS are put in service with a single hypervisor, this approach negatively impacts the scheduling latency of RTOS due to the sharing of pCPUs with GPOS vCPUs because preemption delay necessarily occurs and additional scheduler overhead arises such as system management interrupt (SMI), processor emerging from sleep states, cache migration of data used by woken process, and block on sleeping lock [3,4].

Previous studies described real-time operations on KVM. Accordingly, the research in [5–7] predominantly focused on the schedulability of RTOS VMs by CPU shielding and prioritization while completely reserving pCPUs. As shown in Fig. 2(a), RTOS VMs could be avoided against negative impacts by the previous studies. They assumed that the lower scheduling latency of RTOS VMs was the most important consideration during consolidation of RTOS and GPOS. As a result, GPOS VMs were not allowed to access multicore resources. Unfortunately, the overhead does not disappear entirely even with their method. Even though the prioritization technique is already applied, preemption delay necessarily occurs and additional scheduler overhead arises. Moreover, CPU resources dedicated for RTOS VMs are greatly wasted because the tasks running on RTOS have enough scheduling margins. Therefore, resource-conscious assignment policy of physical multicores in a hypervisor becomes important when two different types of OSes are running together on a single hardware.

In fact, in [8,9], GPOS vCPUs were shared between RTOS-reserved cores for high throughput without CPU shielding, as illustrated in Fig. 2(b). They allowed GPOS vCPUs to be run on RTOS-reserved cores only when RTOS was idle. Although they tried to remove the negative influences concerning the scheduling latency of RTOS, efficient resource sharing between the two different VMs could not be achieved on the multicore processor is too much.

We think resource efficiency for computationally intensive jobs needs to be regarded as important as the real time property of the control processing. Furthermore, RTOS tasks do not always require a constant level of responsiveness [10,11]. Even the Linux operating system could be applied in a safety-critical spacecraft domain without hard real-time [12,13]. Therefore, a hypervisor must adaptively select pCPU assignment policy based on the demanded responsiveness of tasks.

In this paper, we present a dynamic isolation method, which is a pCPU assignment policy for multicore-resource efficiency, considering consolidation of RTOS and GPOS VMs, and then analyzed what effects should be controlled by the method. The goal of the study is improving the overall resource efficiency of a virtualized system by addressing some of the negative impacts that the GPOS VM has on the latency of the RTOS VM. The proposed dynamic isolation does not completely remove all of the negative effects, hence we claim that using dynamic CPU isolation method guarantees soft real-time on the RTOS VM while significantly improving the throughput performance on the GPOS VM at the same time, resulting in improvement of multi-core CPU utilization rate. While our base consolidated environment shares pCPUs with CPU shielding and prioritization techniques, a hypervisor with our dynamic isolation should make decisions about when and how to isolate RTOS and GPOS VMs on RTOS reserved pCPUs to reduce negative influences to RTOS VMs. However, the state of each task in guest operating systems is not transparent to the host in full virtualization layer [14]. This semantic gap forces the hypervisor to make decisions when to or how to isolate guest operating systems on itself. In order to resolve the challenge when full virtualization is used for reusability of industrial system, we analyze the interactions and mutual influences of RTOS VMs and GPOS VMs. This analysis does not pertain to real-time scheduling algorithms. Based on the analysis instead, appropriate factors and points were identified for the dynamic isolation.

We perform experimental analysis of explicit effects by both event-driven and threshold-based approaches. In our best practices, those approaches allow each RTOS and GPOS VM to meet the requirement of improving resource efficiency and minimizing negative influences to scheduling latency. However, since the priority of GPOS vCPUs is always lower than that of RTOS vCPUs, the GPOS vCPUs can implicitly waste pCPU cycles by the well-known synchronization problems, such as lock-holder preemption (LHP) [15] and hidden cost problem [16]. Hence, we also propose urgent boosting with dynamic isolation to reduce unnecessary CPU cycles. Lastly, we performed a set of experiments on an automotive digital cluster application and verified whether the rendering frame rate on the digital cluster is maintained at soft real-time level with worst case frame rate.

All schemes were implemented and evaluated on the well-known open source full virtualized hypervisor, KVM [17]. We evaluated whether our methods enabled RTOS and GPOS VMs to run in a multicore-based hypervisor while improving the resource efficiency and the performance of GPOS while avoiding high scheduling latency on RTOS VMs.

The primary contributions of this research are as following.

- Our analysis suggests that static isolation only maintains low latency of RTOS with the trade-off of degrading the throughput of GPOS while fully time-sharing enhances GPOS throughput and increases the scheduling latency of RTOS. We examine and analyze how an RTOS VM and a GPOS VM interact and influence each other when they are integrated on a virtualization layer.
- We introduce dynamic isolation method, which guarantees soft real-time operations for RTOS tasks while improving the throughput of GPOS tasks on a virtualized multicore system. Events for dynamic isolation in full virtualized systems are investigated to analyze dynamic isolation effects.
- We analyzed the explicit and implicit effects of dynamic isolation on the synchronization method for vCPU. Evaluation of our experiments revealed coupling dynamic isolation with vCPU coordinated scheduling has a positive impact on performance. In the experiment of an industrial application moreover, the RTOS VM guaranteed stable performance and GPOS increasing throughput at the same time even though the RTOS VM and the GPOS VM were sharing the same CPU cores.

Fig. 18. Automotive digital cluster hardware and display.

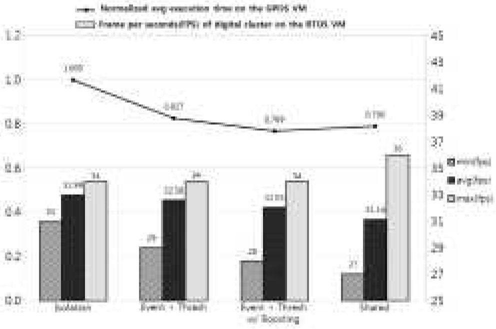


Fig. 19. Effect of dynamic isolation (frame rate on RTOS VM and execution time on GPOS VM).

quality of service (QoS) or SLA. From the viewpoint of RTOS, if the worst case execution time (WCET) can be safely guaranteed in a virtualized environment, the RTOS-reserved resources can be further exploited by GPOS vCPUs within the limit of the WCET of each RTOS task, although it might depend on circumstances. In our experiment about the automotive digital cluster, we identified a threshold value where rendering frame rate change became significant in soft real-time level. Therefore, fine-tuning for SLA or QoS is necessary depending on how much latency is required by the application if our approach is applied to real industrial systems.

In practice, Fig. 15 shows that the scheduling latency value of RTOS was higher with the application *streamcluster* on the GPOS in contrast with results of other applications. This signifies that the behavior of applications is related to scheduling delay. In future works therefore, we will seek to dynamically analyze the behavioral characteristics and workload of applications and attempt to apply to hypervisors the proposed techniques according to the type of various given missions, scenarios and system performance.

7. Related work

The domain of virtualization in various systems increasingly incorporates the simultaneous running of GPOS and RTOS on one hardware platform [1,2,5,6,36–38]. Ma et al. [5], Kiszka [6], and Katharina et al. [37] investigated configurations for running these two types of operating systems in KVM virtualized environments. KVM is a TYPE-II fully virtualized open-source hypervisor, and it is based on the Linux kernel module and the QEMU framework. The CPU on KVM is hardware-assisted, and I/O devices are virtualized using the emulation features on QEMU. Occasionally, para-virtualization via the VirtIO driver is used for performance reasons. In KVM, the vCPU is implemented as a thread. The data structure in Linux kernel can be accessed to change its scheduling policy and priority. So, previous studies sought ways to protect multicore resources reserved for RTOS by means of prioritization, CPU shielding, and interrupt affinity when RTOS VMs and GPOS VMs were sharing a hardware platform. However, they took account of guaranteeing only the real-time execution of RTOS tasks and excluded performance of GPOS tasks.

A few works exist that looked at environments in which a GPOS and an RTOS were sharing a hardware platform [8,9,36]. In their scenarios, the hypervisor allowed execution of GPOS tasks only if the state of the RTOS was idle, to minimize any negative effects on the RTOS VM by the GPOS. In particular, Nakajima et al. [36] stipulated that the scheduling latency of RTOS needs to be minimized and the throughput of GPOS maximized for the purposes of investigations and introduced a virtualization layer for embedded systems called SPUMONE. In SPUMONE, the GPOS allows sharing RTOS-reserved cores when the state of RTOS is idle. Our proposed dynamic isolation method is similar to them. However, SPUMONE should share the host kernel with its RTOS VM in privileged mode, analogous to the multi-kernel approach. Moreover, implementation of SPUMONE necessarily involves modification to the source codes in the guest and host operating systems, thus this is specific to their own design. The CPU migration strategy of SPUMONE is made possible as the innards of the guest are transparent only to their hypervisor. Also, SPUMONE [29] mitigated the LHP with CPU migration, but this solution is available if the internal states of the guest operating systems are visible to the hypervisor. This is not an available option for general cases and the approaches cannot be easily applied to well-known open source hypervisors.

Our work focuses on looking at the general types of mutual influences between RTOS and GPOS in a well-known open source hypervisor, and proposes ways to improve resource efficiency while the scheduling latency of RTOS is kept with minimal negative influences against shared GPOS VMs. Moreover, we analyzed the hidden cost caused by virtual IPI and LHP due to synchronization between vCPUs and further proposed urgent boosting with dynamic isolation. This technique was considered not only with dynamic isolation, but also with vCPU co-scheduling technique, and solves the problems all together.

8. Conclusion

In this paper, we treated issues that could arise in the operation of multicore-based hypervisors for industrial systems and suggested feasible solutions for them. Based on our analysis, dynamic isolation can improve the throughput of the GPOS while avoiding the high scheduling latency of the RTOS when an RTOS and a GPOS are put in service on one hardware platform with a virtualization layer. Appropriate factors to isolate RTOS VMs and GPOS VMs were determined with just the restricted set of data available by a virtualization layer.

Also, we verified that dynamic isolation method indirectly addresses the hidden cost of virtual IPIs and LHPs while running multi-threaded applications on a virtualization environment. Thus, with co-scheduling and dynamic isolation methods applied

저자 이름: 블라인드처리

simultaneously, performance of the mixed system significantly improved. All works were implemented in a well-known open source hypervisor, KVM.

In addition, the discussion section treats limitations of this research and areas of application for the future works. Nevertheless, our research introduced novel work to explore and achieved improvements in the operation of VMs in a multicore-based virtualization system with a RTOS and a GPOS running simultaneously.

Acknowledgments

사사문구: 블라인드처리

[redacted]

References

[1] W. River, Applying multi-core and virtualization to industrial and safety-related application. http://leadwise.mediadroit.com/files/8535WP_Multicore_for_Industrial_and_Safety_Feb2009.pdf (accessed on 24.10.16).

[2] G. Heiser, Virtualizing embedded systems: why bother? in: Proceedings of the 48th Design Automation Conference, 2011, pp. 901–905.

[3] Frank Rowand, Using and Understanding the Real-Time Cyclictest Benchmark, Emdedded linux conference 2013. http://elinux.org/images/0/01/Elc2013_rowand.pdf (accessed on 24.10.16).

[4] Felipe Cerqueria, Boern Brandenburg, A Comparison of Scheduling Latency in Linux, PREEMPT RT, and LITMUSRT, in: 9th annual workshop on Operating Systems Platforms for Em-bedded Real-Time applications July 9, 2013. Paris, France.

[5] Ruhui Ma, Fanfu Zhou, Erzhou Zhu, Haibing Guan, Performance tuning towards a KVM-based embedded real-time virtualization system, J. Inf. Sci. Eng. 29 (2013) 1021–1035.

[6] J. Kiszka, Towards Linux as a real-time hypervisor, in: Proceedings of the 11th Real-Time Linux Workshop, 2009, pp. 205–215.

[7] Tomoki Skiyama, Improvement of Real-time Performance of KVM CloudOpen 2012. http://events.linuxfoundation.org/images/stories/pdf/lcna_co2012_sekiyama.pdf (accessed on 24.10.16).

[8] Seehwan Yoo, et al., Mobivmm: a virtual machine monitor for mobile phones, in: Proceedings of the First Workshop on Virtualization in Mobile Computing, ACM, 2008.

[9] L. Cereia, M. Bertolotti, Asymmetric virtualisation for real-time systems, in: IEEE International Symposium on Industrial Electronics, 2008, ISIE 2008, IEEE, Cambridge, 2008, pp. 1680–1685.

[10] H. Takada, S. Iiyama, T. Kindaichi, S. Hachiya, Linux on ITRON: A hybrid operating system architecture for embedded systems, in: 2002 Symposium on Applications and the Internet (SAINT) Workshops, 2002. Proceedings, IEEE, IEEE Computer Society, Washington, DC, USA, 2002, pp. 4–7.

[11] D. Sangorrin, S. Honda, H. Takada, Dual operating system architecture for real-time embedded systems, in: Proceedings of the 6th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications, OSPERT, Brussels, Belgium. 2010.

[12] Kevin Scharpf, Dave Bryan, Mike Anderson, The Use of Carrier Grade Linux in Space. (2007), in: Proceedings of the AI-AA/USU Conference on Small Satellites, Technical Session XII: Software, SSC07-XII-9.

[13] Christopher Huffine, Linux on a small satellite, Linux J. 2005 (132) (2005) 9.

[14] Kara Nance, Brian Hay, Matt Bishop, Virtual machine introspection, IEEE Comput. Soc. 6 (05) (2008) 32–37.

[15] T. Friebel, S. Biemueller, How to deal with lock holder preemption Presentation at Xen Summit North America, 2008.

[16] Xiaoning Ding, Phillip B. Gibbons, Michael A. Kozuch, A Hidden Cost of Virtualization when Scaling Multicore Applica-tions, in: 5th USENIX Workshop on hot topics in clound compu-ting(HotCloud'13), Jun 2013, San Jose, CA.

[17] A. Kivity, U. Lublin, A. Liguori, KVM: the Linux virtual machine monitor, in: Proceedings of the Linux Symposium. Vol. 1. 2007.

[18] Hyunwoo Joe, et al., Full virtualizing micro hypervisor for spacecraft flight computer, in: 2012 IEEE/AIAA 31st Digital Avionics Systems Conference, (DASC), IEEE, 2012, pp. 6C5-1–6C5-9.

[19] J. Calandrino, H. Leontyev, A. Block, U. Devi, J. Anderson, LITMUSRT: A testbed for empirically comparing real-time multiprocessor schedulers, in: Real-Time Systems Symposium, 2006. (RTSS'06). 27th IEEE International, IEEE, 2006, pp. 111–123.

[20] Real-time linux cyclictest https://rt.wiki.kernel.org/index.php/Cyclictest (accessed on 24.10.16).

[21] Christian Bienia, et al., The PARSEC benchmark suite: Characterization and architectural implications, in: Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques, ACM, 2008.

[22] Yoav Etsion, Dan Tsafrir, Dror G. Feitelson, Effects of clock resolution on the scheduling of interactive and soft real-time processes, in: ACM Sigmetrics Performance Evaluation Review, Vol. 31, ACM, 2003, pp. 172–183. no. 1.

[23] Mathieu Desnoyers, Michel Dagenais, LTTng: Tracing across execution layers, from the hypervisor to user-space. in: Linux Symposium, Vol. 101, 2008.

[24] Perf Tool. https://perf.wiki.kernel.org/ (accessed on 24.10.16).

[25] Intel. Intel 64 and ia-32 architectures software developer's manual, volume 3b: System programming guide, part 2, 2010.

[26] Orathai Sukwong, Hyong S. Kim, Is co-scheduling too expensive for SMP VMs? in: Proceedings of the sixth conference on Computer systems, (Eurosys), ACM, 2011.

[27] Wei Jiang, et al., CFS optimizations to kvm threads on multi-core environment, in: 2009 15th International Conference on Parallel and Distributed Systems, (ICPADS), IEEE, 2009.

[28] Hwanju Kim, et al., Demand-based coordinated scheduling for SMP VMs, in: Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems, ACM, 2013.

[29] Hitoshi Mitake, et al., Using virtual CPU migration to solve the lock holder preemption problem in a multicore processor-based virtualization layer for embedded systems, in: 2012 IEEE 18th International Conference on Embedded and Real-Time Computing Systems and Applications, (RTCSA), IEEE, 2012.

[30] V. Milanovic, A. Kasturi, V. Hachtel, High brightness mems mirror based head-up display (hud) modules with wireless data streaming capability, in: SPIE OPTO, International Society for Optics and Photonics, 2015, pp. 93 750A–93 750A.

[31] Chiyoung Lee, Se-Won Kim, Chuck Yoo, VADI: GPU virtualization for an automotive platform, IEEE Trans. Ind. Inf. 12 (1) (2016) 277–290.

[32] Lui Sha, et al., Single Core Equivalent Virtual Machines for Hard Real—Time Computing on Multicore Processors, 2014, https://www.ideals.illinois.edu/handle/2142/55672 (accessed on 24.10.16).

[33] U.M. Tai-Won, et al., Dynamic resource allocation and scheduling for cloud-based virtual content delivery networks, ETRI J. 36 (2) (2014) 197–205.

[34] Attila Kertesz, Gabor Kecskemeti, Ivona Brandic, Auto-nomic sla-aware service virtualization for distributed systems, in: 2011 19th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, (PDP), IEEE, 2011.

[35] Liang Zhao, Sherif Sakr, Anna Liu, Consumer-centric SLA manager for cloud-hosted databases, in: Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management, ACM, 2013.

[36] Tatsuo Nakajima, Yuki Kinebuchi, Hiromasa Shimada, Alexandre Courbot, Tsung-Han Lin, Temporal and spatial isolation in a virtualization layer for multi-core processor based information appliances, in: Proceedings of the 16th Asia and South Pacific Design Automation Conference, 2011.

[37] Gilles Katharina, et al., Proteus hypervisor: Full virtualization and paravirtualization for multi-core embedded systems, in: Embedded Systems: Design, Analysis and Verification, Springer, Berlin, Heidelberg, 2013, pp. 293–305.

[38] P. Garcia, et al. Towards hardware embedded virtualization technology: architectural enhancements to an ARM SoC, in: VtRES 2013 Workshop on Virtualization for Real-Time Embedded Systems August 21st, 2013, Taipei, Taiwan.

[redacted]

저자 이름, 저자 관련 정보(E-mail, 연락처): 블라인드처리

블라인드 처리 샘플(특허)

(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2013년12월05일
(11) 등록번호 10-1337444
(24) 등록일자 2013년11월29일

(51) 국제특허분류(Int. Cl.)
H04L 12/02 (2006.01)
(21) 출원번호 10-2010-0113959
(22) 출원일자 2010년11월16일
심사청구일자 2012년04월12일
(65) 공개번호 10-2012-0052686
(43) 공개일자 2012년05월24일
(56) 선행기술조사문헌
KR1020010073555 A
KR1020090065878 A
KR1020090021695 A

(73) 특허권자
[redacted]
(72) 발명자
[redacted]
(뒷면에 계속)
(74) 대리인
특허법인이상

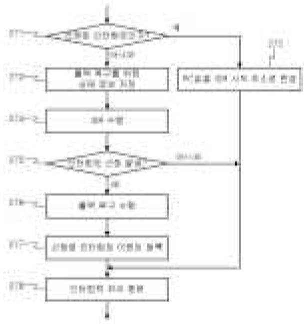
전체 청구항 수 : 총 10 항 심사관 : 이동하

(54) 발명의 명칭 **센서 네트워크 에뮬레이션 장치 및 그 동작 방법**

(57) 요 약

센서 네트워크의 에뮬레이션 성능을 향상시킬 수 있는 센서 네트워크 에뮬레이션 장치 및 그 동작 방법이 개시된다. 먼저, 발생된 인터럽트가 선점된 인터럽트 이벤트인가를 판단하고, 발생된 인터럽트가 선점된 인터럽트가 아닌 경우 롤백 복구를 위한 상태 정보를 저장하고, ISR(Interrupt Service Routine)을 비동기적으로 수행한다. 따라서, 높은 정확성을 유지하면서 에뮬레이션의 수행 시간을 향상시킬 수 있고, 이를 통해 센서 네트워크 응용 프로그램의 개발 시간을 단축시킬 수 있다.

대 표 도 - 도4



- 1 -

(72) 발명자

[redacted]

[redacted]

이 발명을 지원한 국가연구개발사업

과제고유번호 [redacted]
부처명
연구사업명
연구과제명
기 여 율
주관기관
연구기간