The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 22, No. 3, pp.93-98, Jun. 30, 2022. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2022.22.3.93
JIIBC 2022-3-14

가상화 환경에서 고속 스토리지를 위한 워크로드 맞춤형 페이지 크기 모델링

Workload-Aware Page Size Modeling for Fast Storage in Virtualized Environments

반효경*, 박윤주**

Hyokyung Bahn*, Yunjoo Park**

요 약 최근 옵테인 등 고속 스토리지의 출현으로 하드디스크에 적합하게 설계된 메모리 시스템 설정에 대한 재고가 필요한 시점에 이르렀다. 본 논문에서는 고속 스토리지의 탑재에 따라 페이지 크기가 메모리 시스템의 성능에 어떠한 영향을 미치는지를 분석하고, 가상화 환경에서 워크로드 상황에 맞게 페이지 크기를 설정할 수 있는 모델을 설계하였다. 전통적인 시스템의 경우 워크로드 별로 페이지 크기를 설정하는 것이 쉬운 일이 아니지만 최근 클라우드 환경의 활성화로 개별 워크로드 수행을 위해 별도의 가상머신이 생성되므로 가상머신이 시작될 때 해당 가상머신의 페이지 크기를 결정할 수 있어 제안한 모델의 효용이 높을 것으로 기대된다. 다양한 가상머신 시나리오에 대한 시뮬레이션 실험을 통해 제안한 모델이 워크로드 상황에 맞게 페이지 크기를 설정하여 메모리 접근 시간을 크게 개선함을 보인다.

Abstract Recently, fast storage media such as Optane have emerged, and memory system configurations designed for disk storage should be reconsidered. In this paper, we analyze the effect of the page size on the memory system performances when fast storage is adopted. Based on this, we design a page size model that can guide an appropriate page size for given workloads in virtualized environments. Configuring different page sizes for various workloads is not an easy matter in traditional systems, but due to the widespread adoption of cloud systems, page sizing performed in our model is feasible for virtual machines, which are generated for executing specific workloads. Simulation experiments under various virtual machine scenarios show that the proposed model improves the memory access time significantly by configuring page sizes for given workloads.

Key Words : Page size, NVM, cloud, virtual machine, memory access time, fast storage

*정회원, 이화여자대학교 컴퓨터공학과
**정회원, 이화여자대학교 컴퓨터공학과
접수일자 2022년 5월 15일, 수정완료 2022년 6월 3일
게재확정일자 2022년 6월 10일

Received: 15 May, 2022 / Revised: 3 June, 2022 /
Accepted: 10 June, 2022
*Corresponding Author: bahn@ewha.ac.kr
Dept. of Computer Engineering, Ewha University, Korea

Ⅰ. 서 론

수십 년째 메모리 시스템의 페이지 크기로 4KB가 가장 널리 사용되고 있다[1]. 페이지 크기를 변경하려는 시도들이 있었으나 대부분의 운영체제에서는 여전히 4KB를 기본 페이지 크기로 사용하고 있다[2]. 이는 스토리지로 사용되는 디스크의 매체 특성이 거의 변하지 않았다는 점과도 상당한 관련성이 있다[3, 4]. 한번의 헤드 이동으로 많은 데이터를 읽는 것이 효과적인 디스크의 특성을 잘 활용하기 위해서는 큰 페이지가 효과적이다. 이는 데이터의 크기와 무관하게 매 I/O마다 소요되는 디스크 헤드의 이동시간이 스토리지 접근의 가장 큰 시간 비중을 차지하기 때문이다[3]. 다만, 메인 메모리의 크기가 한정돼 있으므로 무작정 큰 용량보다는 실제로 사용될 데이터만 읽어오는 것이 효과적이다. 따라서, 페이지 크기를 4KB로 하고 순차적인 메모리 참조가 확인된 경우 미리읽기를 통해 한번의 I/O시 다수의 페이지들을 한꺼번에 읽어오는 방식이 사용된다[5, 6].

최근 메모리 시스템의 용량 증가로 4MB 등 대용량 페이지가 사용되는 시도가 늘고 있는 추세이며, 리눅스 역시 다양한 크기의 페이지를 지원하고 있다[7]. 하지만, 기존의 운영체제들은 페이지 크기를 변경할 수 있는 옵션만 제공할 뿐 주어진 상황에 맞게 페이지 크기를 결정하는 기능은 제공하지 않는다.

한편, 최근 SSD(Solid State Drives)를 넘어서는 고속 스토리지인 NVM(Non-Volatile Memory)의 등장으로 페이지 크기가 성능상 중요한 문제로 부각되고 있다[8, 9]. NVM은 매체의 접근 시간이 짧고 헤드의 이동이 없어 많은 양의 데이터를 한꺼번에 읽어서 얻게 되는 디스크의 장점을 더 이상 기대할 수 없게 되었다[10]. 이러한 환경에서는 소규모 페이지를 사용하는 것이 효과적이나, 그럴 경우 메모리 주소 변환시간이 늘어 성능 저하가 발생할 수 있다. 고속 스토리지 환경에서는 요청 페이지가 메모리에 존재하지 않아 발생하는 I/O 오버헤드가 줄어든 반면 주소변환, 즉 가상 메모리 주소를 물리적 메모리 주소로 변환하는 시간의 영향력은 상대적으로 커졌기 때문이다. 따라서, 주소변환용 캐쉬인 TLB(translation lookaside buffer)의 적중률을 높이는 것이 중요하며, 이를 위해서는 페이지 크기를 증가시켜 한정된 TLB 용량으로 더 넓은 메모리 영역의 주소변환을 가능하게 해야 함을 뜻한다[11].

이와 같은 상충되는 이유로 NVM 스토리지 환경에서는 워크로드 및 메모리 상황에 따라 효율적인 페이지 크기가 다르므로 주어진 환경에 적합한 페이지 크기를 결정하는 것이 중요한 문제로 부각되었다. 특히, 최근 클라우드 환경의 활성화로 가상머신 별로 수행되는 워크로드의 특성과 가상머신에 할당된 메모리 크기가 미리 결정되면서 이에 적합한 페이지 크기를 결정하는 것이 필요하다[12].

전통적인 시스템에서는 워크로드 별로 페이지 크기를 설정하는 문제가 크게 주목받지 못했는데, 이는 단일 운영체제 하에서 워크로드마다 페이지 크기를 상이하게 적용하는 것이 쉽지 않기 때문이다. 그러나, 최근 PC, 클라우드 등 다양한 환경에서 가상화 기술이 활성화되면서 각 가상머신별로 페이지 크기를 어렵지 않게 설정할 수 있게 되었다. 이는 가상머신이 보통 특정 워크로드의 수행을 위해 생성되고 가상머신 생성 시 메모리 등 자원 상황이 결정된 후 게스트 운영체제가 새롭게 시작되기 때문이다.

본 논문에서는 NVM 스토리지가 탑재된 시스템에서 페이지 크기가 메모리 성능에 어떠한 영향을 미치는지를 분석하고, 가상화 환경에서 워크로드의 상황에 맞게 페이지 크기를 설정할 수 있는 모델을 설계한다. 다양한 가상머신 시나리오에 대한 시뮬레이션 실험을 통해 제안한 모델이 워크로드 및 시스템 상황에 맞게 페이지 크기를 적절히 설정하여 메모리 접근 시간을 크게 개선함을 보인다.

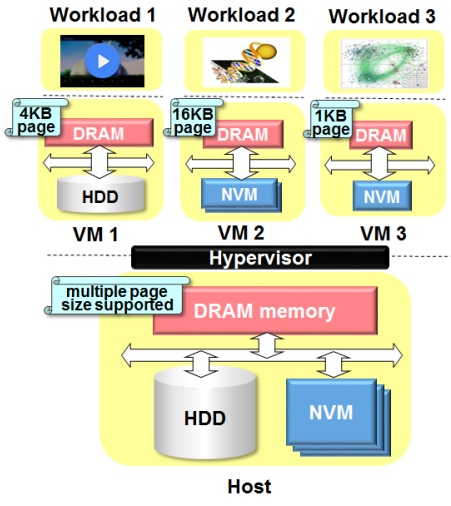


그림 1. 가상머신별 자원 현황 및 워크로드 특성에 따른 페이지 크기 설정의 예

Fig. 1. An example of page size setting for each virtual machine with different resource configurations and workload characteristics.

II. 메모리 접근시간 모델링

메모리 접근 과정은 주소변환과 실제 데이터 접근으로 구성되며, HDD 환경에서는 데이터 접근시 해당 페이지가 메모리에 없어 I/O를 동반하는 경우 오랜 지연을 초래하므로 주소변환시간의 상대적인 비중이 매우 적다. 그림 2(a)는 HDD 환경에서 페이지 크기 변화에 따른 데스크탑 워크로드의 메모리 접근시간을 보여주고 있다(워크로드에 대한 구체적인 소개는 III장에서 이루어질 예정임). 그림에서 보는 것처럼 HDD 환경에서는 데이터 접근시간이 전체 메모리접근시간의 대부분을 차지하며 페이지 크기가 4KB 이상인 구간에서는 큰 변화가 없는 성능을 나타내고 있다.

한편, 그림 2(b)에서 보는 것처럼 스토리지가 NVM인 경우 메모리 접근시간에서 주소변환이 차지하는 비중이 커지게 되었는데 이는 스토리지 성능이 빨라지면서 데이터 접근시간이 크게 줄어들었기 때문이다. 또한, HDD의 경우와 달리 페이지 크기 증가에 따라 메모리 접근시간이 어느 정도 개선되다가 일정 구간 이후부터는 급격히 나빠지는 것을 확인할 수 있다. 이는 주소변환과 데이터 접근이라는 두 시간요소 간의 트레이드 오프에 의해 발생하며, 워크로드 특성 및 메모리 용량에 따라 이러한 추세는 다르게 나타나는 것을 확인할 수 있었다. 이에 본 장에서는 NVM 스토리지 환경에서 주소변환 및 데이터 접근시간을 포괄하는 메모리 접근시간 모델을 설계하고 이를 시뮬레이션을 통해 검증하고자 한다.

메모리 접근시간 T_{TOTAL}은 식 (1)에서 보는 것처럼 주소변환시간 T_A와 데이터 접근시간 T_D로 구성된다.

$$T_{TOTAL} = T_A + T_D \quad (1)$$

주소변환은 D램 메모리에 존재하는 페이지테이블을 통해 이루어지거나, 그 일부를 담은 주소변환용 캐쉬인 TLB를 통해 이루어지며, 그 시간은 식 (2)와 같이 표현될 수 있다.

$$T_A = (1-\varepsilon)*T_{TLB} + \varepsilon*(T_{TLB} + T_{DRAM}) \quad (2)$$

이때, T_{TLB}는 TLB 접근시간, T_{DRAM}은 D램 메모리 접근시간, ε는 TLB 미스율을 뜻한다.

데이터 접근은 해당 페이지가 메모리에 존재하는 경우 D램 메모리 접근으로 이루어지고, 그렇지 않은 경우(이를 "페이지 폴트"라고 부름) I/O를 통한 스토리지 접근이 필요하며, 그 시간은 식 (3)과 같이 표현될 수 있다.

$$T_D = (1-\delta)*T_{DRAM} + \delta*(T_{DRAM} + T_{I/O}) \quad (3)$$

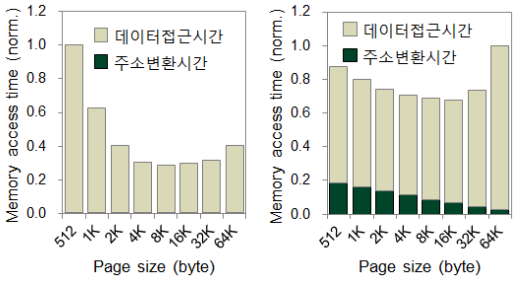


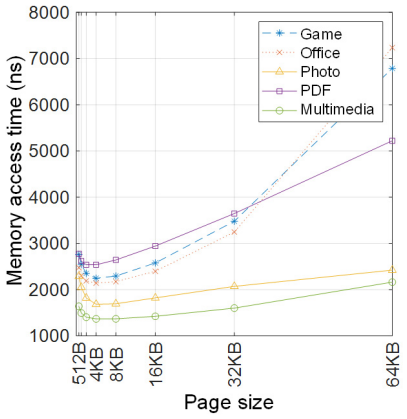
그림 2. 페이지 크기에 따른 메모리 접근 시간
Fig. 2. Memory access time as a function of the page size.

이때, δ는 페이지 폴트율, $T_{I/O}$는 페이지 폴트 처리시간을 뜻한다.

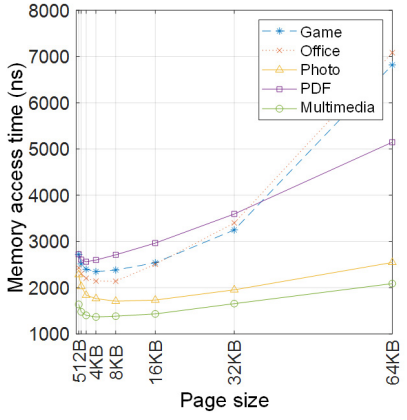
상기의 식에서 나머지 시간요소들은 하드웨어적으로 결정이 되며, TLB 미스율 ε과 페이지폴트율 δ는 워크로드 및 메모리 상황에 의존적이므로 전체 메모리접근시간을 잘 예측하기 위해서는 이들에 대한 예측 모델이 필요하다. TLB 미스율의 경우 페이지 크기가 커질수록 개선되는 특성이 있는데 이는 고정된 TLB 용량으로 더 많은 메모리 영역에 대한 주소변환을 고속의 TLB를 통해 할 수 있기 때문이다. 페이지 크기에 따른 TLB 미스율은 단항을 가진 파워 피팅으로 모델링할 수 있어 본 논문에서도 이를 사용하였다[8].

한편, 페이지 폴트율은 페이지 크기뿐 아니라 메모리가 여유 있는 상황인지 부족한 상황인지에도 의존적이어서 모델링이 다소 복잡하다. 일반적으로 페이지 크기가 커질수록 공간 지역성의 효과가 높아져 페이지 폴트율이 개선되나 한정된 메모리에 담을 수 있는 데이터의 다양성이 떨어져 어느 순간부터는 다시 성능이 나빠지는 지점이 존재한다. 따라서, 페이지 폴트율은 단일항을 이용한 피팅이 불가능하여 2개의 항을 가진 지수 피팅을 통해 모델링을 하였다.

그림 3은 이러한 2가지 피팅을 통해 본 연구의 모델에서 예측한 메모리 접근시간과 실제 시뮬레이션을 통해 얻은 결과치를 비교해서 보여주고 있다. 그림에서 보는 것처럼 제안한 모델은 실제 메모리 참조시간을 잘 예측하는 것을 확인할 수 있으며, 이러한 상황은 메모리 크기나 워크로드 변화에도 잘 동작하는 것을 확인할 수 있었다.



(a) 모델에 의한 메모리 접근시간의 예측치



(b) 시뮬레이션에 의해 얻어진 실제 메모리 접근시간

그림 3. 메모리 접근시간 모델의 검증
Fig. 3. Validation of the memory access time model.

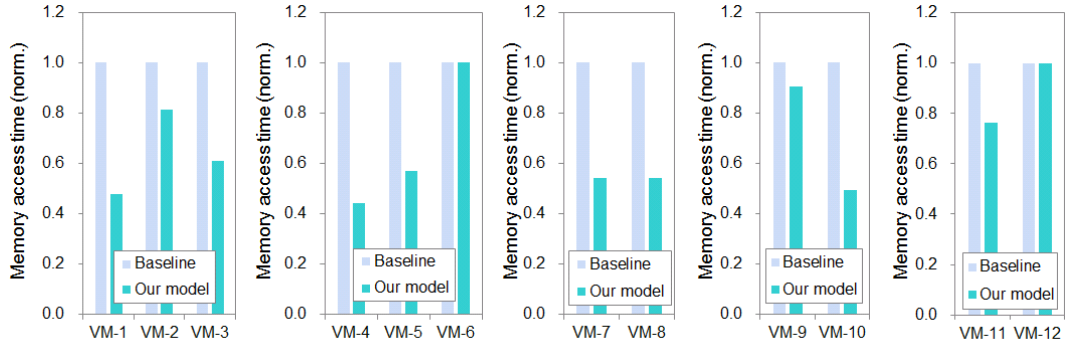
표 1. 실험에 사용된 가상머신 시나리오
Table 1. Virtual machine scenarios used in experiments.

	VM	Workload	Memory size	Storage
Scenario 1	VM-1	Game	80%	NVM
	VM-2	Office	10%	NVM
	VM-3	Photo	60%	NVM
Scenario 2	VM-4	PDF	90%	NVM
	VM-5	Multimedia	50%	NVM
	VM-6	Game	30%	HDD
Scenario 3	VM-7	Office	80%	NVM
	VM-8	Photo	80%	NVM
Scenario 4	VM-9	PDF	20%	NVM
	VM-10	Multimedia	80%	NVM
Scenario 5	VM-11	Game	60%	NVM
	VM-12	Office	40%	HDD

Ⅲ. 성능 평가

본 장에서는 5종의 워크로드를 각기 다른 12개의 가상머신으로 실행시키는 클라우드 시나리오를 가지고 페이지 크기 모델의 효과를 시뮬레이션으로 검증한다. 표 1은 각 실험에서 사용한 클라우드 설정을 보여주고 있다. 워크로드는 데스크탑 게임(Game), 사무용 오피스 프로그램(Office), 포토 편집 및 관리(Photo), 문서 뷰어(PDF), 멀티미디어 플레이어(Multimedia)의 5종을 사용하였으며, 메모리 크기는 해당 워크로드의 전체 풋프린트 대비 가상머신에 할당된 실제 메모리 크기의 비율을 나타낸다.

그림 4는 제안한 모델과 4KB를 사용하는 Baseline 시스템 간의 메모리 접근 시간을 비교해서 보여주고 있다. 그림에서 보는 것처럼 제안한 모델은 Baseline 시스



(a) 시나리오 1 (b) 시나리오 2 (c) 시나리오 3 (d) 시나리오 4 (e) 시나리오 5

그림 4. 가상머신 시나리오별 메모리 접근시간 비교
Fig. 4. Comparison of memory access time for each virtual machine scenarios.

템 대비 메모리 접근 시간을 크게 줄이는 것을 확인할 수 있다. 성능 개선의 효과는 평균 38%였으며, 최대 56%였다. VM-4에서 가장 큰 성능 개선이 있었으며 이는 해당 가상머신의 메모리 크기가 상대적으로 커서 제안한 모델이 4KB와 매우 상이한 페이지 크기를 사용했기 때문이다. 제안한 모델은 메모리 크기가 커짐에 따라 성능 개선 효과가 증가했는데 이는 주소변환시간을 개선했기 때문으로 볼 수 있다. 한편, VM-2는 메모리 크기가 상대적으로 작은 세팅이었지만 본 논문의 모델이 19%의 성능 개선을 나타내었다. 이는 해당 가상머신에서 수행된 워크로드인 Office가 메모리 집약적인 성격을 가지고 있어 페이지 크기가 메모리 성능에 민감한 결과를 보였기 때문이다. HDD를 스토리지로 사용한 VM-6과 VM-12에서는 성능 개선이 없었으며 이를 제외하고는 제안한 모델이 모든 경우에서 성능 개선 효과를 나타내었으며, 그 범위는 10%에서 56%로 워크로드 및 시스템 상황에 따라 조금씩 달랐다.

한편, 제안한 모델이 채택한 페이지 크기는 워크로드의 특성 및 가상머신의 메모리 크기에 따라 1KB에서부터 32KB까지 다양하게 나타났다. VM-2는 1KB의 페이지를 채택했으며, VM-9는 2KB의 페이지 크기를 채택했다. 이에 비해 VM-6과 VM-12는 4KB 페이지를, VM-11은 8KB, VM-1, VM-3, VM-5, VM-8은 16KB 페이지를 채택했으며, VM-4, VM-7, VM-10은 32KB 페이지를 채택했다. 가상머신의 메모리 용량이 상대적으로 작거나 워크로드의 참조 지역성이 약한 경우 페이지 폴트가 메모리 접근시간의 상당 부분을 차지하게 되므로 소규모 페이지를 사용하여 데이터 접근시간을 줄이는 방식으로 동작하는 것을 확인하였다. VM-2과 VM-9가 이에 해당된다. 반면, 가상머신의 메모리 용량이 충분하거나 워크로드의 참조 지역성이 강한 경우 주소변환시간이 메모리 성능에 중요한 영향을 미치며, 이 경우 제안한 모델은 상대적으로 큰 페이지 크기를 설정하여 성능 개선을 도모하는 것을 확인할 수 있었다. VM-1, VM-3, VM-4, VM-5, VM-7, VM-8, VM-10이 이러한 경우에 해당한다.

Ⅳ. 결 론

본 논문에서는 고속 스토리지를 사용할 경우 페이지 크기가 메모리 성능에 어떠한 영향을 미치는지 분석하고, 가상화 환경에서 워크로드 상황에 맞게 페이지 크기를 설정할 수 있는 모델을 제안하였다. 최근 클라우드 환경의 활성화로 각각의 워크로드 수행을 위한 별도의 가상머신이 생성되고 가상머신 별로 메모리 크기가 미리 할당되는 특성이 있다. 전통적인 시스템처럼 여러 워크로드가 단일 OS 위에서 수행될 경우 각 워크로드에 맞는 페이지 크기를 별도로 설정하는 것은 어려운 일이지만 가상화 환경에서는 가상머신이 시작될 때 해당 가상머신의 페이지 크기를 결정하는 것이 비교적 용이하게 실현될 수 있다. 최근, 참조 지역성이 약하고 메모리 풋프린트가 매우 큰 빅데이터 응용의 증가로 대용량의 메모리를 필요로 하는 상황이 늘고 있으나, D램의 집적도 한계 및 에너지 소모 문제로 메모리 용량을 늘리는 대신 고속 스토리지를 사용하는 방안이 대안으로 논의되고 있다. 이러한 상황에서 본 연구는 차세대 빅데이터 응용을 위한 가상머신 최적화에 적극 활용될 수 있을 것으로 기대된다.

References

[1] F. Guvenilir and Y. Patt, “Tailored page sizes,” Proc. ACM/IEEE Int’l Symp. on Computer Architecture (ISCA), pp. 900-912, 2020.
DOI: https://doi.org/10.1109/ISCA45697.2020.00078

[2] P. Weisberg and Y. Wiseman, “Using 4KB page size for virtual memory is obsolete,” Proc. IEEE Conf. on Information Reuse & Integration, pp. 262-265, 2009.
DOI: https://doi.org/10.1109/IRI.2009.5211562.

[3] S. Ng, “Advances in disk technology: performance issues,” IEEE Computer, vol. 31, no. 5, pp. 75-81, 1998.
DOI: https://doi.org/10.1109/2.675641

[4] T. Kim and H. Bahn, “Implementation of the storage manager for an IPTV set-top box,” IEEE Trans. on Consumer Electronics, vol. 54, no. 4, pp. 1770-1775, 2008.
DOI: https://doi.org/10.1109/TCE.2008.4711233

[5] O. Kwon, H. Bahn, and K. Koh, “Popularity and prefix aware interval caching for multimedia streaming servers,” Proc. IEEE Conf. on Computer and Information Technology, pp. 555-560, 2008.
DOI: https://doi.org/10.1109/CIT.2008.4594735

[6] R. Karedla, J.S. Love, and B.G. Wherry, “Caching strategies to improve disk system performance,” IEEE Computer, vol. 27, no. 3, pp. 38-46, 1994.
DOI: https://doi.org/10.1109/2.268884

[7] N. Ganapathy and C.Schimmel, “General purpose operating system support for multiple page sizes,” Proc. USENIX Annual Technical Conf., pp. 91-104,

1998.

[8] Y. Park and H. Bahn, "Modeling of the TLB miss rate and the page fault rate for NVM-based storage systems," Proc. IEEE Int'l Conf. on Information Science and Control Engineering, pp. 856-860, 2020. DOI: https://doi.org/10.1109/ICISCE50968.2020.00178

[9] I. Shin, "Performance evaluation of applying shallow write in SSDs with internal cache," The Journal of KIIT, vol. 17, no. 1, pp. 31-38, 2019. DOI: https://doi.org/10.14801/jkiit.2019.17.1.31

[10] Intel Optane™ Technology, https://www.intel.com/content/www/us/en/architecture-and-technology/intel-optane-technology.html

[11] Y. Park and H. Bahn, "Impact analysis for page size of desktop and smartphone environments under fast storage media," The Journal of The Institute of Internet, Broadcasting and Communication, vol. 22, no. 2, pp. 77-82, 2022. DOI: https://doi.org/10.7236/JIIBC.2022.22.2.77

[12] J. Park and E. Park, "Performance evaluation of IoT cloud platforms for smart buildings," Journal of the Korea Academia-Industrial cooperation Society(JKAIS), vol. 21, no. 5 pp. 664-671, 2020. DOI: https://doi.org/10.5762/KAIS.2020.21.5.664

저 자 소 개

반 효 경(정회원)

- 1997년 2월 : 서울대학교 계산통계학과 학사
- 1999년 2월 : 서울대학교 전산과학과 석사
- 2002년 2월 : 서울대학교 컴퓨터공학부 박사.
- 2002년 9월 ~ : 이화여자대학교 컴퓨터공학과 교수.
- 주관심분야 : 운영체제, 스토리지시스템, 임베디드시스템

박 윤 주(정회원)

- 2015년 2 월: 이화여자대학교 컴퓨터공학과 학사
- 2015년 3월 ~ : 이화여자대학교 컴퓨터공학과 통합과정
- 주관심분야 : 운영체제, 스토리지 시스템, 임베디드 시스템

※ This work was supported by the IITP grant funded by the Korea government (MSIT) (No.2021-0-02068, Artificial Intelligence Innovation Hub) and the ICT R&D program of MSIT/IITP (2018-0-00549, Extremely Scalable Order Preserving OS for Manycore and Non-volatile Memory).