# 고속 I/O 시스템에서 HDFS를 위한 동적 코어 친화도 프레임워크 성능 분석

조중연○, 진현욱
건국대학교 컴퓨터공학부
{jycho, jinh}@konkuk.ac.kr

# Performance Analysis of Dynamic Core Affinity Framework for HDFS over High-Performance I/O Devices

Joong-Yeon Cho○, Hyun-Wook Jin
Department of Computer Science & Engineering, Konkuk University

**요 약**

멀티코어 시스템에서 코어 친화도가 네트워크 I/O 성능에 미치는 영향은 다양한 연구들을 통해 관찰 되었다. 점차 고속화되는 네트워크 연결의 발전에 따라 효율적인 코어 친화도 정책은 중요한 성능 요소가 될 수 있다. HDFS를 위한 동적 코어 친화도 프레임워크는 네트워크와 디스크 I/O를 함께 고려한 코어 친화도 정책을 제안하였지만 비교적 낮은 속도의 I/O 장치를 이용하고 있어 고속 I/O 장치에 대한 고려는 이루어지지 않았다. 본 논문에서는 40Gbps 이더넷과 NVMe가 장착된 고속 I/O 시스템에서 HDFS를 위한 동적 코어 친화도 프레임워크의 성능에 대해 실험한다. 실험 결과를 통해 과거의 연구에서 제안하고 있는 최적 코어 결정 정책의 유효성과 한계점에 대해 논의한다. 실험 결과 HDFS를 위한 동적 코어 친화도 프레임워크는 고속 I/O 시스템에서 역시 효용성을 보이고 있지만, 시스템 설정에 따라 성능에 영향을 받을 수 있음을 확인하였다.

## 1. 서 론

멀티코어 시스템에서 코어 친화도가 네트워크 I/O 성능에 미치는 영향은 다양한 연구들을 통해 관찰되었다[1,2,3,4]. 점차 고속화되는 네트워크 연결의 발전에 따라 40Gbps 이더넷이 장착된 시스템을 이용하여 코어 친화도의 영향에 대해 분석한 연구[5]도 진행되었다. 기존의 연구들은 모두 최하위 레벨 캐시(Last Level Cache, LLC) 메모리를 공유하는 두 개의 코어에서 각각 커널 수준의 네트워크 패킷 프로세싱과 응용 수준의 처리를 병렬화해야 한다는 점에서 유사한 결과를 보이고 있다. 하지만 40Gbps 이더넷을 사용한 경우 코어 친화도의 영향에 따른 네트워크 I/O 성능 차이는 더욱 크게 차이가 나는 것을 관찰할 수 있다[5,6]. 따라서 I/O 장치들이 고속화될수록 효율적인 코어 친화도 정책은 중요한 성능 요소가 될 수 있다.

HDFS를 위한 동적 코어 친화도 프레임워크[7]는 코어 친화도가 디스크 I/O 성능에 미치는 영향에 대해 분석하고 네트워크와 디스크 I/O를 함께 고려한 코어 친화도 정책을 제안하였다. 하지만 과거의 연구에서 제안하고 있는 친화도 정책은 비교적 낮은 속도의 I/O 장치를 이용해 분석하고 실험하여 고속 I/O 장치에 대한 고려는 이루어지지 않았다. 따라서 고속 I/O 장치를 사용하는 시스템의 경우 HDFS를 위한 동적 코어 친화도 프레임워크의 영향이 더욱 커질 수 있는 잠재력이 존재하는 동시에 코어 친화도 결정 정책 알고리즘이 고속 I/O 장치에 부적합하여 예상할 수 없는 부작용을 초래할 수도 있다.

본 논문에서는 40Gbps 이더넷과 NVMe(Non-Volatile Memory Express)[8]가 장착된 고속 I/O 시스템에서 HDFS를 위한 동적 코어 친화도 프레임워크의 성능을 실험하고 과거의 연구에서 제안하고 있는 최적 코어 결정 정책의 유효성과 한계점에 대해 논의한다.

본 논문은 다음과 같이 구성되어 있다. 서론에 이어 2장에서는 HDFS를 위한 동적 코어 친화도 프레임워크에 대해 간략히 살펴보고 3장에서는 고속 I/O 장치를 사용하는 시스템에서 HDFS를 위한 동적 코어 친화도 프레임워크가 HDFS 파일 업로드 성능에 미치는 영향에 대해 실험하고 분석하며, 마지막으로 4장에서는 본 논문의 결론을 내린다.

## 2. HDFS를 위한 동적 코어 친화도 프레임워크

HDFS를 위한 동적 코어 친화도 프레임워크는 하드웨어 구조 정보 수집기, 코어별 부하 정보 수집기와 코어 친화도 관리자로 구성되어 있다[7].

하드웨어 구조 정보 수집기는 프로세서의 LLC 구조, I/O 장치의 연결 구조 및 인터럽트 처리 코어와 같은 구조적 정보를 수집한다. 수집된 하드웨어 구조 정보는 코어 친화도 관리자에게 전달되어 HDFS의 파일 저장 쓰레드를 위한 적합 코어를 결정하기 위해 사용된다.

이 논문은 2015년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.B0101-15-0644, 매니코어 기반 초고성능 스케일러블 OS 기초연구).

코어별 부하 정보 수집기는 주기적으로 코어별 부하 정보를 수집한다. 수집된 코어별 부하 정보는 코어 친화도 정책 알고리즘에 의해 일부 코어에 파일 저장 쓰레드가 집중되어 성능이 낮아지는 부작용을 방지하기 위해 사용된다.

코어 친화도 관리자는 HDFS의 파일 저장 쓰레드를 생성하기 위한 데몬 프로세스가 HDFS의 파일 저장 쓰레드를 수행하기 전에 수집된 정보들을 이용해 최적 코어를 결정한다. 코어 친화도 관리자에 의해 선택된 최적 코어는 쓰레드 생성 시 인자로 전달되어 파일 저장 쓰레드가 시스템 콜을 호출하는 방식으로 동작한다.

## 3. 성능 측정 및 실험 결과 분석

### 3.1 실험 시스템 및 실험 방법

실험을 위한 HDFS 클러스터 구성에는 표 1에 명시된 네임노드와 데이터노드를 각각 한 대씩 사용하였다. 네트워크 연결을 위해 네임노드와 데이터노드를 QSFP+ 케이블로 직접 연결하였다. 네임노드로 사용되는 머신은 파일을 업로드하는 클라이언트 노드의 역할을 함께 수행하였다. 각 클라이언트 쓰레드는 5.6GB 크기의 XML 파일을 업로드하며 쓰레드의 개수를 16개까지 증가시키면서 파일 업로드 처리율을 측정하였다. HDFS를 위한 동적 코어 친화도 프레임워크[7]는 수정 없이 사용하였다.

표 1 실험 시스템 구성

| | 네임노드 | 데이터노드 |
|---|---|---|
| CPU | Intel Core i7 3770 | Intel Xeon E5-2680v2 * 2 |
| Memory | 8GB | 128GB |
| Storage | 삼성 SSD (SATA) | 인텔 NVMe (PCIe 3.0) |
| NIC | Chelsio T580-LP-CR 40GbE | |
| OS | CentOS 7.1 (Kernel Version 3.10.0) | |

### 3.2 기본 시스템 설정에서의 성능 영향

기본 시스템 설정에서 코어 친화도가 설정되지 않은 HDFS 파일 업로드 성능과 HDFS를 위한 동적 코어 친화도 프레임워크가 적용된 경우의 HDFS 파일 업로드 성능을 실험하였다. 그림 1은 실험 결과를 보여준다.

그림 1에서 볼 수 있듯이 HDFS를 위한 동적 코어 친화도 프레임워크가 적용된 경우 1개의 클라이언트가 파일을 업로드 하는 경우 23%, 2개의 클라이언트가 파일을 업로드 하는 경우 32% 만큼 성능이 향상된 것을 확인할 수 있다. 하지만 4, 8, 16개의 클라이언트가 파일을 업로드 하는 경우 HDFS를 위한 동적 코어 친화도 프레임워크가 적용되지 않은 환경과 비교하여 성능 향상이 없는 것을 확인할 수 있다.

실험 결과를 통해 적은 수의 클라이언트가 파일을 업로드하는 환경에 제한적이지만 여전히 HDFS를 위한 동적 코어 친화도 프레임워크의 영향이 존재하는 것으로 판단된다. 파일 업로드 클라이언트의 수가 증가할수록 파일 업로드 성능이 낮아지는 원인 중 하나로 네임노드에서 발생할 수 있는 병목현상을 고려할 수 있다. HDFS는 하나의 네임노드가 데이터노드들에 저장된 메타데이터를 모두 저장하고 관리함에 따라 HDFS의 성능에 영향을 미칠 수 있다.

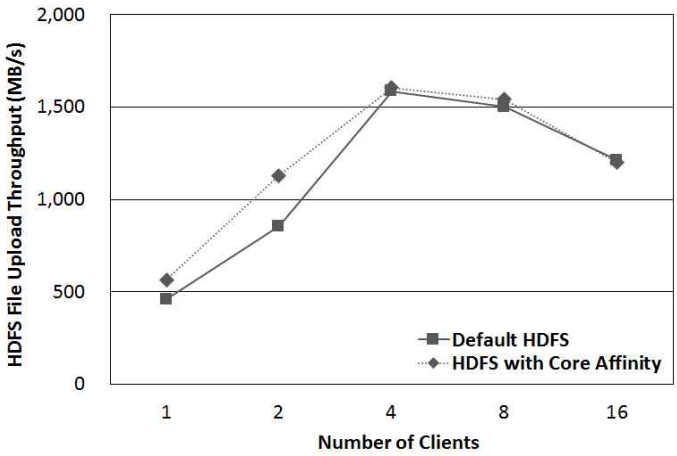


그림 1 기본 설정에서의 파일 업로드 성능

### 3.3 HDFS Federation 설정과 성능 영향

HDFS는 HDFS Federation 설정을 통해 다수의 네임노드를 설정하여 네임노드에 의한 병목현상을 완화시킬 수 있다. HDFS Federation 환경을 구축하고 실험하기 위해 3.1절에 기술한 HDFS 클러스터에 네임노드와 동일한 사양의 머신을 추가하여 두 대의 네임노드와 한 대의 데이터노드로 구성된 HDFS 클러스터를 구성하였다. 세 대의 머신을 연결하기 위해 데이터노드에 장착된 40GbE 네트워크 인터페이스 카드가 제공하는 두 개의 포트 중 첫 번째 포트는 첫 번째 네임노드에, 두 번째 포트는 두 번째 네임노드에 각각 연결하였다. 그림 2는 실험 결과를 보여준다.

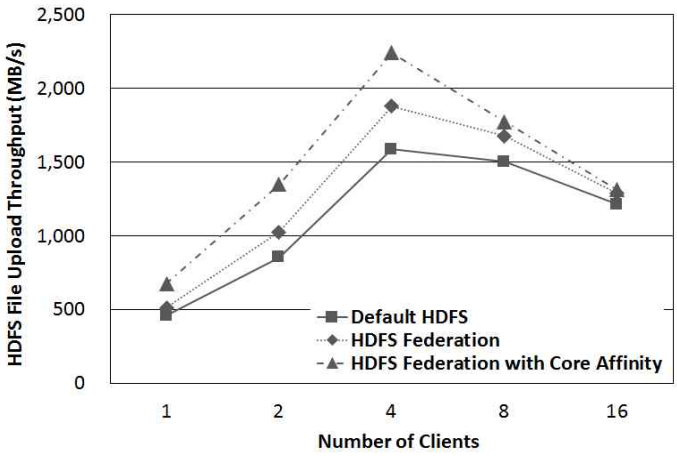


그림 2 HDFS Federation 환경에서의 파일 업로드 성능

그림 2에서 볼 수 있듯이 HDFS Federation 환경의 경우 3.2절에서 사용한 클러스터 구성과 비교했을 때 전

반적으로 5%~20%의 성능 향상이 있음을 확인할 수 있다. HDFS Federation 환경에 동적 코어 친화도 프레임워크를 적용한 경우 3.2절에서 사용한 클러스터 구성과 비교했을 때 8%~60%, HDFS Federation 환경과 비교했을 때 6%~30% 만큼 성능이 향상된 것을 확인할 수 있다.

실험 결과를 통해 HDFS Federation 환경에 HDFS를 위한 동적 코어 친화도 프레임워크를 적용한 경우 3.2절의 실험 결과와 달리 4개의 클라이언트가 파일을 업로드하는 환경에서도 성능 향상이 있는 것으로 확인된다. 하지만 여전히 8개 이상의 클라이언트가 파일을 업로드 하는 경우 성능이 제한되는 것을 확인할 수 있다. NVMe는 비휘발성 메모리와 고속 시스템 버스, 멀티 I/O 큐와 같은 기법을 이용해 높은 성능 향상을 이루었지만 40Gbps 이더넷과의 상대적인 속도를 고려한다면 네트워크와 디스크 I/O에 모두 집중적인 HDFS의 성능에 병목지점이 될 수 있다.

### 3.4 버퍼 캐시의 크기와 성능 영향

네트워크와 디스크 I/O 성능 차이에서 발생할 수 있는 병목현상을 완화시키기 위해 데이터노드에 설치된 리눅스 운영체제의 시스템 설정을 조작해 버퍼 캐시의 크기를 25GB로 설정한 뒤 실험을 진행하였다. HDFS 클러스터 구성은 3.3절에 사용한 구성을 유지하였다. 그림 3은 실험 결과를 보여준다.

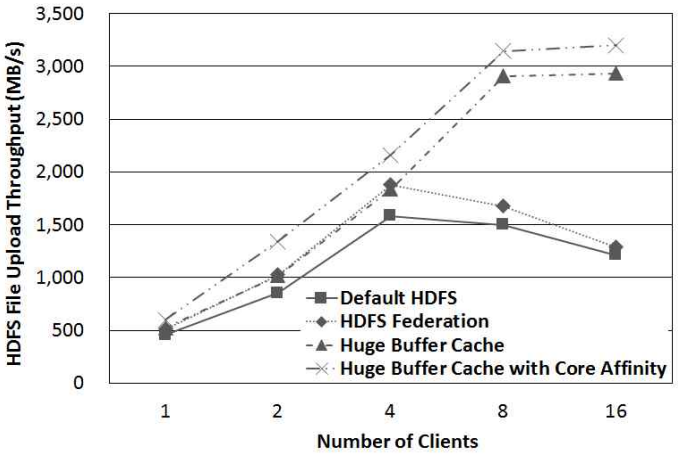


그림 3 버퍼 캐시가 증가된 환경의 파일 업로드 성능

그림 3에서 볼 수 있듯이 HDFS를 위한 동적 코어 친화도 프레임워크가 적용되지 않은 경우와 HDFS를 위한 동적 코어 친화도 프레임워크가 적용된 경우 모두 8, 16개의 클라이언트가 파일을 업로드 하는 환경의 HDFS 파일 업로드 성능이 개선된 것을 확인할 수 있다. 또한 HDFS를 위한 동적 코어 친화도 프레임워크가 적용된 경우 전반적으로 8%~30%의 성능 향상을 보이는 것을 확인할 수 있다.

실험 결과를 통해 HDFS를 위한 동적 코어 친화도 프레임워크는 고속 I/O 시스템에서 역시 I/O 성능을 향상시킬 수 있음을 확인하였다.

## 4. 결론 및 향후 계획

본 논문에서는 실험을 통해 고속 I/O 시스템에서 HDFS를 위한 동적 코어 친화도 프레임워크의 성능을 분석하였다. 실험 결과를 통해 HDFS를 위한 동적 코어 친화도 프레임워크는 고속 I/O 시스템에서 향상된 성능을 보여 새로운 시스템에서도 효율적으로 동작할 수 있음을 확인하였다. 하지만 HDFS 클러스터 구성을 비롯한 시스템 설정에 따라 성능에 영향을 받을 수 있음을 함께 확인하였다.

향후 계획으로는 다양한 환경에서 HDFS를 위한 동적 코어 친화도 프레임워크의 성능을 분석하고 최적화를 수행할 계획이다.

## 참고 문헌

[1] Pesterev, A., Strauss, J., Zeldovich, N., and Morris, R. T., "Improving network connection locality on multicore systems," In Proc. of EuroSys, pp. 337-350, 2012.

[2] Ahuja, V., Farrens, M., and Ghosal, D., "Cache-aware affinitization on commodity multicores for high-speed network flows," In Proc. of ACNS, pp. 39-48, 2012.

[3] Wu, W., DeMar, P., and Crawford, M., "A transport-friendly NIC for multicore/multiprocessor systems," IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 4, pp. 607-615, 2012.

[4] Jang, H.-C., and Jin, H.-W., "MiAMI: Multi-core Aware Processor Affinity for TCP/IP over Multiple Network Interfaces," In Proc. of Hotl, pp. 73-82, 2009.

[5] Hanford, N., Ahuja, V., Balman, M., Farrens, M. K., Ghosal, D., Pouyoul, E., and Tierney, B., "Characterizing the impact of end-system affinities on the end-to-end performance of high-speed flows," In Proc. of NDM, pp. 1:1-1:10, 2013.

[6] Cho, J.-Y., and Jin, H.-W., "An Optimization Tool for Determining Processor Affinity of Networking Processes," KIPS Transactions on Software and Data Engineering, vol. 2, no. 2, pp. 131-136, 2013.

[7] Cho, J.-Y., Jin, H.-W., Lee, M., and Schwan, K., "Dynamic core affinity for high-performance file upload on Hadoop Distributed File System," Parallel Computing, vol. 40, no. 10, pp. 722-737, 2014.

[8] NVMe, http://www.nvmexpress.org/.