매니코어 환경에서 멀티 유저 확장성을 위한 docker 기반 프로그램 실행 기법

Docker-based program execution technique for multi-user scalability in Manycore enviroment

저자 (Authors)
서동주, 고가을, 임성수
Dongjoo Seo, Gaeul Go, Sung-soo Lim

출처 (Source)
한국정보과학회 학술발표논문집 , 2017.12, 1533-1535(3 pages)

발행처 (Publisher)
한국정보과학회
The Korean Institute of Information Scientists and Engineers

URL
http://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE07322555

APA Style
서동주, 고가을, 임성수 (2017). 매니코어 환경에서 멀티 유저 확장성을 위한 docker 기반 프로그램 실행 기법. 한국정보과학회 학술발표논문집, 1533-1535

매니코어 환경에서 멀티 유저 확장성 을 위한 docker 기반 프로그램 실행 기법

서동주* 고가을 임성수
국민대학교 소프트웨어융합대학
commisori28@gmail.com, eul8828@kookmin.ac.kr, sslim@kookmin.ac.kr

Docker-based program execution technique for multi-user scalability in Manycore enviroment

Dongjoo Seo Gaeul Go Sung-soo Lim
School of Computer Science Kookmin University

요 약

급격히 증가하고 있는 CPU 코어 숫자에 따라 한개의 PC 에서 여러명의 유저의 프로세스를 처리하고자 하는 일들이 늘어나고있다. 하지만 기존 리눅스 커널의 페이지 교체 알고리즘인 LRU(Least Recently Used) 는 하나의 전역 LRU로만 운영되고있다. 이는 여러명의 유저가 접속하는 환경에 적합하지 않다. 이를 위해 Docker 를 구성하고 있는 Cgroups 의 기능인 Private LRU 를 이용하여 각 유저마다 또는 그룹마다 LRU 를 추가해 확장성을 가질 수 있는 기법을 제안하고 성능 분석을 통해 확장성을 입증한다.

1 서 론

CPU 코어 수가 증가함에 따라서 이를 멀티코어를 넘어 매니코어라 부르기 시작하고, 이를 이용하기 위한 많은 연구들이 진행되어 왔다. 그 중 매니코어의 여러가지 확장성을 분석하기 위한 연구도 다양하게 진행되었다. [1].

우리의 이전 연구들은 멀티 프로세스 또는 멀티 스레드 기반 프로그램의 확장성에 대해서 분석하고 이를 해결하였다. [2] [3] [4].

기존 연구 [2] [3] [4]들은 락 경합에 집중했었다. 하지만 연구를 진행하는 과정에서 멀티유저가 매니코어를 이용한다고 가정하였을 때 확장성에 문제를 일으키는 다른 요인들을 발견하였다.

실제 프로그램이 실행될때 메모리 컨트롤러는 하나의 큐와 하나의 교체 알고리즘을 가지고 모든 프로그램들을 관리하게 된다. 이는 각 코어마다 있는 cache coherency 문제를 가지게 된다. 이를위해 Non-Uniform Memory Access (NUMA) 구조를 이용하여 해결하곤 하지만, 하드웨어적 지원은 한계가 있는 것이 사실이다.

이를 해결하기 위해 본 연구는 커널의 Cgroups 를 이용한 Docker 를 사용하여 소프트웨어적으로 각 실행그룹의 메모리 큐를 만들어주고 이를 전역 큐에 포인팅 하는 방식으로 각 그룹별로 메모리 그룹을 만들어주었다. 또한 이를 AIM7 벤치마크를 이용하여 분석하고 입증하였다.

* 이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No. B0101-17-0644, 매니코어 기반 초고성능 스케일러블 OS 기초연구)

2 관련 연구

2.1 AIM7 benchmark

AIM7 벤치마크는 여러 프로세스(task)를 fork 하는 C 언어 프로그램이다. 각 task는 동시의 임의의 순서의 하위 테스트 세트를 실행한다. 53가지 종류의 테스트 세트가 존재하게 되는데, 각각은 디스크 IO, 프로세스 생성, 사용자 가상 메모리 작업, 파이프 I/O 및 연산 바인딩 산술 루프 같은 시스템 기능의 특정 측면들은 수행한다. 또한 순차 읽기, 순차 쓰기, 임의 읽기, 임의 쓰기 및 임의 혼합 읽기/ 쓰기 를 위한 디스크 테스트 또한 포함되어 있다.

2.2 기타 연구

위에서 제시한 문제점을 해결하기 위해 많은 연구들이 진행 되고 있다. 대표적인 예로 CC-NUMA 관련된 연구들 [5] [6]이 있다. 실제 시장에 판매되고 있는 모든 NUMA 컴퓨터들은 cache coherency 를 위해서 하드웨어적으로 이를 완화시키려고 노력하고 있다. 이 부류를 CC-NUMA라고 한다.

표 1: 소프트웨어 및 측정 프로그램

cpu	코어 수	유형	벤치마크 프로그램
Xeon Phi 7210	64	Host	AIM7

3 실험 환경

실험 환경은 표 1과 같은 64코어 인텔 xeon기반 머신과 AIM7 벤치마크를 사용하였다.

4 실험 설계

이번 장에서는 1장에서 소개한 Docker 기반의 프로그램 실행 기법에 대해서 제안한다. 기존 구조는 메모리 페이지를 가지고 있는 큐가 전역에 하나 밖에 존재하지 않는다는 단점이 있다. 이에 반해 Cgroup 을 사용하고 있는 Docker 위에서 프로그램이 실행되고 있을 때의 메모리 큐는 그림 1과 같은 구조를 하고있다. 이 점에 착안하여 커널의 기존 구조와 Docker 로 그룹핑한 상태의 확장성을 비교 분석 하기로 하였다. 이를 확인하기 위한 방법으로 AIM7 벤치마크를 활용한다.

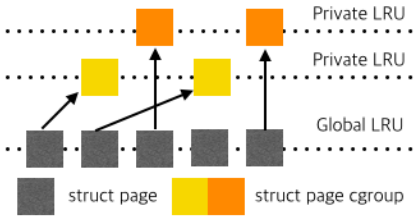

그림 1: 그룹 메모리 큐 구조

5 확장성 실험 결과

이번 장에선 두가지 실험 결과를 바탕으로 결과를 도출해 내었다. 모든 실험은 15회 기준 평균 값으로 진행되었다. 첫번째로 벤치마크의 분당 실행가능한 task 수 를 기준으로 기존 리눅스의 확장성을 확인해보고, 제안한 구조의 확장성과 비교하여 보았다.

5.1 기본 리눅스 확장성

기본 리눅스 커널의 확장성을 알아보기 위해 벤치마크를 이용하여 실험을 진행하였다. 그림 2 을 보면 알 수 있듯이 주어지는 tasks 수에 따라 jobs/min 즉, 실제 수행되고 있는 실행의 숫자가 차차 증가하다 특정 시점 부터 꺾여 성능이 하락하기 시작한다. 본 실험 환경에서 보여줄 수 있는 가장 높은 성능 지점인 고점의 task 수를 기준으로 비교를 진행하였다.

5.2 실험 결과

해당 실험결과는 기본 커널의 확장성이 가장 높을때의 task 숫자를 group의 수만큼 나눠 각 group의 jobs/min을 합한 결과이다.

전체적으로 기존 커널에서 바로 실행하는 것보다 높은 성능을 보였으며 특정구간에선 성능이 60%나 상승하는 결과를 보여주었다. 하지만 64코어 머신인데도 불구하고 64개의 그룹에 각 16개의 task를 기준으로 측정하였을때는 이전 32개의 그룹보다 성능이 크게 하락하였다.

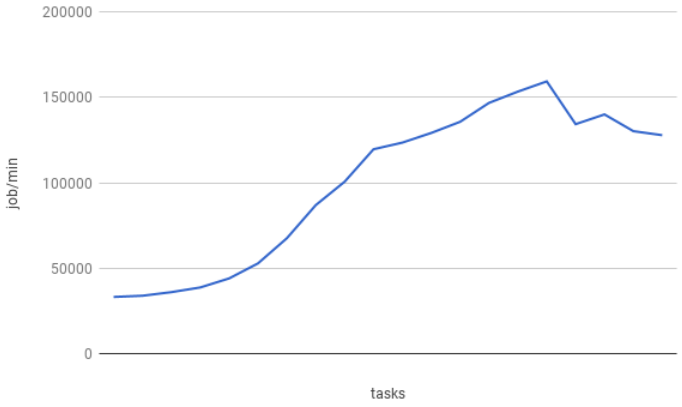

그림 2: AIM7 리눅스 커널 실험.

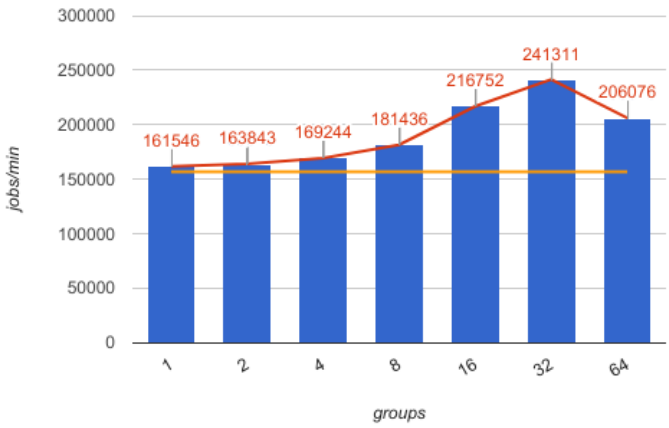

그림 3: 기존 커널과 그룹핑 실행결과 분석.

6 결론 및 향후 연구

본 연구에서는 매니코어 컴퓨팅의 이슈 중 하나인 cache coherence 문제에 대해서 Docker 기반 프로그램 실행기법을 제안했다. 성능측정도구인 AIM7 벤치마크를 이용하여서 우리가 제안한 실행기법은 기존 일반적인 실행 방법보다 성능이 확장됨을 확인했다. 하지만 그룹핑이 어떻게 되느냐에 따라서 성능이 좌우되는 것을 확인 할 수 있었다. 이를 개선하기 위한 연구도 진행되어야 한다. 추가로 해당 연구를 기점으로 Docker에서 사용하는 리눅스 커널 내부의 Cgroups, namespace 등을 직접 이용하여 사용자가 프로그램을 실행해주면 자동으로 확인해서 고립 시켜주는 모듈을 개발할 예정이다.

참고 문헌

[1] 정진환, 김강호, 김진미, 정성인. Manycore 운영체제 동향. 전자통신 동향 분석 29권 5호, 2014.

[2] 경주현, 임성수. 매니코어 시스템을 위한 리눅스 가상 메모리 관리의 락 경합 분석. 한국정보과학회, 한국정보과학회 학술발표논문집 , 2015.6, 1571-1573, 2015

[3] 경주현, 윤성민, 임성수. 매니코어 환경에서 로그 기반 동시적 업데이트 기법을 활용한 리눅스 커널 확장성 개선. 한국정보과학회 학술발표논문집 , 2016.12, 512-513, 2016

[4] 서동주, 경주현, 임성수. 매니코어 환경에서 PARSEC 벤치마크 ROI 확장성 분석. 한국정보과학회, 한국정보과학회 학술발표논문집 , 2017.6, 1525-1527, 2017

[5] Tom Lovett, Russell Clapp STiNG: a CC-NUMA computer system for the commercial marketplace 23th Anual International Symposium on Computer Architecture (ISCA),Pennsylvania, USA, May 1996

[6] Maged M. Michaely, Ashwini K. Nandaz, Beng-Hong Limz, and Michael L. Scotty Coherence Controller Architectures for SMP-Based CC-NUMA Multiprocessors 24th Anual International Symposium on Computer Architecture (ISCA), Denver, CO, June 1997