웹 기반 슈퍼컴퓨터 서비스 포털 개발에 관한 연구

박주원, 우준, 홍태영
한국과학기술정보연구원 슈퍼컴퓨팅인프라센터 책임연구원
juwon.park@kisti.re.kr, wjnadia@kisti.re.kr, tyhong@kisti.re.kr

A Study on Web-based Supercomputer Service Portal

Ju-Won Park, Joon Woo, Taeyoung Hong
Korea Institute of Science and Technology Information

요 약

전통적으로 컴퓨팅 자원을 많이 필요로 하는 기초과학 분야 뿐만 아니라 최근 딥러닝, ChatGPT 와 같은 인공 지능 분야에서도 대규모의 컴퓨팅 자원에 대한 요구가 지속적으로 증가하고 있다. 그러나 스케줄러 기반으로 운영되는 기존의 HPC 클러스터의 경우 성능 최적화를 위해 구성된 소프트웨어 스택의 경직성으로 인해 새로운 서비스 도입에 어려움이 있다. 이러한 어려움을 해결하기 위해 본 논문에서는 컨테이너 기반의 가상화 기술로 개발된 MyKSC 서비스를 소개한다. MyKSC 는 기존의 HPC 클러스터와 쿠버네티스 클러스터가 공존하는 약 결합 아키텍처를 기반으로 설계되어 슈퍼컴퓨터 운영 센터에서는 기존의 소프트웨어 스택 변경 없이 손쉽게 적용할 수 있다. 또한 컨테이너 가상화 기술이 적용되어 사용자 맞춤형 서비스가 가능하며 웹 기반의 그래픽 사용자 인터페이스를 통해 슈퍼컴퓨터를 처음 사용하는 사용자도 쉽게 슈퍼컴퓨터를 활용할 수 있다.

1. 서론

전통적으로 기상, 화학, 고에너지 물리와 같은 기초과학 분야에서는 슈퍼컴퓨터와 같은 고성능 컴퓨팅 자원을 이용하여 대규모의 작업을 실행하고 있다. 슈퍼컴퓨팅 운영 센터에서는 고성능의 컴퓨팅 자원을 다수의 사용자가 효율적으로 활용할 수 있도록 SLURM[1], PBS[2]와 같은 배치 작업 스케줄러 기반의 HPC (High Performance Computing) 클러스터를 구축하여 서비스를 제공하고 있다. 즉, 사용자는 로그인 노드에 원격으로 접속한 후 작업을 실행하기 위한 작업 스크립트 파일을 작성하여 배치작업 스케줄러에 제출하면 배치 작업 스케줄러가 사용자가 요청한 자원과 유휴 자원 (available resource)을 매칭하여 작업을 실행한다. 더욱이 슈퍼컴퓨팅 센터에서는 컴퓨팅 성능 향상을 위해 작업 실행에 필요한 모든 환경을 사전에 구축하여 제공하고 있으며 커널 파라미터 튜닝에 많은 노력을 기울이고 있다. 그러나 이러한 운영 방식은 성능 최적화에는 도움이 되지만 소프트웨어 스택의 경직성을 가져오고 이로 인해 다양한 서비스 지원의 제약을 가져온다. 최근 ChatGPT 와 같은 AI, 빅데이터 분석과 같은 컴퓨터 과학 분야에서도 대규모의 컴퓨팅 자원을 요구하고 있으나 이러한 소프트웨어 경직성으로 인하여 많은 슈퍼컴퓨터 운영 센터에서 새로운 서비스의 도입 및 제공을 주저하고 있다. 이러한 어려움을 해결하기 위해 컨테이너, 하이퍼바이져 기반의 가상화 기술은 매우 좋은 대안이 될 수 있다[3][4]. 특히, 컨테이너 기반의 가상화 기술은 매우 경량화된 가상화 계층을 통해 우수한 성능을 보여주기 때문에 HPC 분야에서도 적극 도입을 검토하고 있다[5]-[6].

본 논문에서는 슈퍼컴퓨팅 자원의 유연한 활용을 통해 새로운 서비스를 효과적으로 제공하기 위해 개발된 MyKSC (KISTI 슈퍼컴퓨터 웹 서비스 포털)을 소개한다. KISTI 슈퍼컴퓨터 사용자는 MyKSC 를 통해 MPI 와 같은 기존의 HPC 응용 서비스와 함께 Jupyter, RStudio 와 같은 새로운 형태의 데이터 분석 플랫폼 서비스를 이용할 수 있다. MyKSC 의 주요 특징은 다음과 같다. 첫째, 약 결합(loosely-coupled) 아키텍처를 기반으로 기존의 소프트웨어 스택의 변경 없이 손쉽게 도입할 수 있다. MyKSC 에서는 전체 시스템을 새롭게 등장하는 서비스 제공을 위한 쿠버네티스 (Kubernetes) 클러스터와 기존 HPC 서비스를 위한 스케줄러 기반의 클러스터가 분리되어 공존한다. 이렇게 분리된 자원을 기반으로 MyKSC 는 사용자가 선택한 서비스의 형태에 따라 적합한 클러스터 자원을 선택하여 응용 프로그램을 실행한다. 즉, Jupyter, Rstudio 와 같은 새로운 형태의 서비스는 쿠버네티스 클러스

터에서 실행되고 기존의 HPC 서비스는 쿠버네티스 클러스터에서 실행되는 batchjob 서비스를 통해 병렬 작업에 필요한 작업 스크립트을 생성하고 기존 HPC 클러스터에 제출함으로써 실제 작업은 HPC 클러스터에서 수행된다. 둘째는, 쿠버네티스 기반의 컨테이너 서비스를 제공함에 따라 사용자 맞춤형 서비스가 가능하다. 사용자는 도커 (docker)[7] 이미지를 기반으로 자신이 원하는 플랫폼을 선택하고 자신의 컴퓨터에서 관리자 권한으로 필요한 라이브러리을 설치함으로써 이미지를 사용자화 (customize) 할 수 있으며 생성된 자신의 이미지를 슈퍼컴퓨터를 통해 실행할 수 있다. 셋째, 쿠버네티스 클러스터와 기존의 HPC 클러스터는 스토리지 연동을 통해 동일한 데이터를 접근할 수 있다. 기존의 사용자는 SSH 와 같은 터미널 기반의 원격 접속 프로그램을 통해 파일을 전송하거나 작업 스크립트 파일을 작성하였다. 터미널 기반의 사용 환경에 익숙하지 않은 사용자에게 있어 이는 슈퍼컴퓨터 사용에 큰 걸림돌이었다. 그러나 MyKSC 에서는 웹 기반의 GUI (Graphical User Interface)를 통해 쉽게 HPC 클러스터에 파일 업로드/다운로드가 가능하다. 또한 MyKSC 를 통해 HPC 클러스터에 작업을 제출하고 관리할 수 있기 때문에 슈퍼컴퓨터를 처음 사용하는 사용자도 쉽게 대규모 작업을 실행할 수 있다.

본 논문의 구성은 다음과 같다. 먼저 2 절에서는 MyKSC 의 아키텍처를 살펴보고 제 3 절에서는 MyKSC 구현 관련 상세 내용을 알아본다. 마지막으로 제 4 절에서는 결론을 제시한다.

2. 아키텍쳐

기존의 많은 논문에서는 HPC 클러스터를 통해 빅데이터 분석 및 머신러닝과 같은 데이터 분석 플랫폼을 제공하기 위해 클라우드 기술을 활용하는 연구가 많이 진행되었다. 그러나 클라우드 서비스의 경우에는 기존의 HPC 서비스와는 근본적인 차이가 몇가지 존재한다.

먼저, 자원을 활용하는 방식이다. 기존의 HPC 응용의 경우 규모가 큰 자원을 이용하여 높은 성능을 추구하기 때문에 자원 할당은 주로 독점적인 형태(exclusive)로 진행되었다. 반면 클라우드 서비스는 주로 주문형 (on-demand) 방식으로 자원을 활용하기 때문에 독점적인 형태보다는 자원을 공유 형태(shared)로 자원이 할당된다.

둘째, 작업의 규모와 서비스 실행 시간이다. HPC 응용은 큰 규모의 자원을 할당 받아 오랜 시간 동안 실행된다. 그러므로 대부분의 슈퍼컴퓨터 운영 센터에서는 사용자 간의 공정한 자원 사용을 위해 실행 시간 제한(walltime)이 설정되어 그 이상으로 작업이 실행되는 것을 방지한다. 그러므로 사용자는 자신의 작업이 스케줄러에 의해 강제로 종료되는 상황에 대비하여 일정 간격으로 체크포인트(Checkpoint)를 저장함으로써 추후 작업이 재시작될 경우 저장된 시점에서 실행될 수 있도록 대비하여 작업을 수행한다. 반면 클라우드 서비스의 경우에는 다수의 마이크로 서비스(micro-service)가 REST API 를 통해 연결되어 실행되는 경우가 많다. 즉, 필요에 따라 마이크로 서비스가 실행되고 종료되는 과정이 반복적으로 발생할 뿐만 아니라 다양한 문제로 인하여 연결이 끊어지는 경우가 많기 때문에 연결이 끊어질 경우 다른 마이크로 서비스를 찾고 이와 연결하는 방식으로 서비스가 이루어진다. 그러므로 클라우드 서비스에서는 체크포인트와 같은 기법이 아닌 서로의 서비스의 실행 상태를 주기적으로 모니터링하고 동적으로 서비스를 생성/연결/종료할 수 있는 방법이 매우 중요하다.

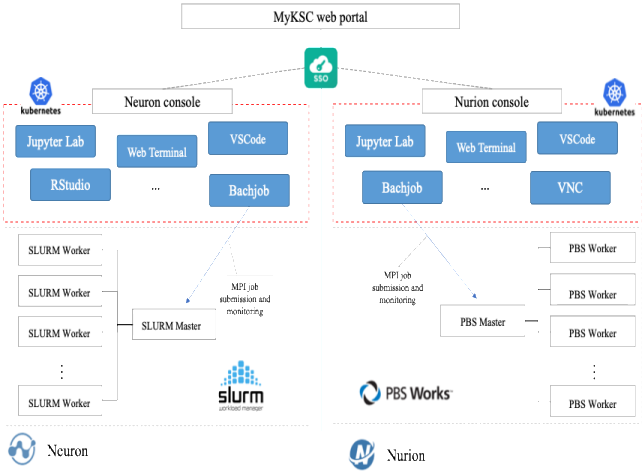
셋째, 서비스가 실행되는 형태이다. HPC 응용의 제일 중요한 요소는 성능이다. HPC 응용에서 요구하는 성능을 만족시키기 위해 모든 계산 노드에 HPC 응용 프로그램 실행에 필요한 모든 패키지 및 라이브러리가 설치되고 커널 파라미터까지 최적화한다. 그러므로 HPC 응용은 이렇게 최적화를 통한 폐쇄된(closed) 환경에서 배치 형태로 작업이 실행된다. 반면 클라우드 서비스에서는 매우 유연한 실행 방식이 요구되어진다. 즉, 사용자의 다양한 요구 사항을 만족시킬 수 있도록 실행 환경을 동적으로 구성 제공해야 한다. 특히, 클라우드에서는 특정 응용 프로그램을 위해 모든 노드를 최적화하지 않고 다양한 응용 환경을 제공하기 위해 성능을 희생하더라도 추상화(abstraction)를 통해 유연성을 향상 시킨다.

이러한 HPC 서비스와 클라우드 서비스의 근본적인 차이로 인하여 실제 시스템을 운영하는 슈퍼컴퓨팅 센터에서는 컨테이너 가상화 기술 도입을 주저하고 있다. 이러한 문제를 해결하기 위해 본 논문에서는 약 결합(loosely-coupled)된 형태의 아키텍쳐 기반으로 구현된 MyKSC 를 소개하고자 한다.

그림 1 은 MyKSC 의 아키텍쳐를 보여준다. MyKSC 에서는 클라우드 서비스를 제공하기 위해 전체 시스템 자원을 쿠버네티스 기반의 클라우드 클러스터와 기존의 스케줄러 기반의 HPC 클러스터로 양분하여 구축하였다. 그러므로 슈퍼컴퓨터 운영 센터 입장에서는 기존의 HPC 클러스터 시스템 구성 및 소프트웨어 스택은 변경하지 않고 쿠버네티스 클러스터와 공존하게 됨으로써 손쉽게 도입할 수 있다. 이러한 쿠버네티스 클러스터의 도입을 통해 MyKSC 는 다음 3 가지의 장점을 가진다. 첫째, MyKSC 에서는 서비스

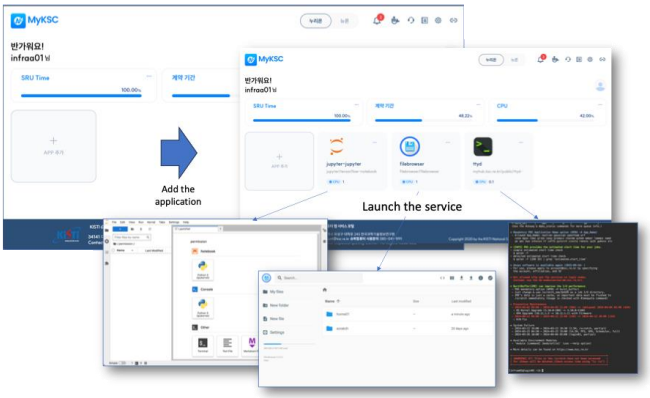
배포가 매우 빠르다. MyKSC 의 경우에는 클라우드 기반의 데이터 분석 서비스를 제공하기 위해 컨테이너 기술을 활용하였다. 이를 통해 매우 빠른 서비스 배포가 가능하다. 둘째, 다양한 사용자 요구에 따른 맞춤형 서비스가 가능하다. 데이터 분석의 경우 사용자가 선호하는 플랫폼 및 패키지가 매우 다양해서 기존의 HPC 환경에서는 맞춤형 서비스를 제공하는데 어려움이 있었다. 그러나 MyKSC 의 경우에는 쿠버네티스 클러스터를 통해 사용자가 직접 생성한 도커 이미지를 기반으로 서비스 배포가 가능하다. 셋째, 모든 서비스가 웹 기반의 그래픽 사용자 인터페이스 (Graphical User Interface: GUI)를 통해 사용자 친화적 서비스를 제공한다.

(그림 1) MyKSC 아키텍쳐



3. 구현

MyKSC 에서는 컨테이너 오케스트레이션의 사실상 표준이 되는 쿠버네티스 플랫폼을 기반으로 구축되었다. MyKSC 에서는 컨테이너 런타임 인터페이스 (CRI)로써 conatinerd 와 NVIDIA container toolkit 을 사용하였다. 또한 파드(Pod) 간의 네트워킹 및 정책 관리를 위해 calico 를 이용하였다. Calico 의 경우 XVLAN 기반의 L2 와 더불어 BGP 라우팅을 통한 L3 연결도 지원한다. 이는 KISTI 에서 현재 운영 중인 네트워크의 복잡성을 해결할 수 있어 선택하였다. 서비스의 부하분산 및 고가용성을 제공하기 위해 MetalLB 를 설치하여 구성하였으며 Ingress 제어를 위해 Istio 를 선택하였다.



(그림 2) MyKSC 대시보드 화면

그림 2 는 MyKSC 대시보드 화면을 보여준다. MyKSC 대시보드는 크게 2 구역으로 구분된다. 상단에는 로그인 사용자의 계약 기간과 함께 사용자에게 할당된 자원량, 현재 사용 중인 자원량을 보여준다. 하단은 배포된 서비스 아이콘을 카드 형태로 제공된다. 서비스를 배포하기 위해서는 '+ APP 추가' 버튼을 클릭하고 1) 서비스 이름, 2) 실행하고자 하는 Docker 이미지, 3) 서비스에 할당하고자 하는 자원량을 입력한 후 생성 버튼을 클릭하면 동적으로 서비스가 배포된다. 배포가 완료되면 카드 배열 형태로 서비스가 생성되고 사용을 위한 준비가 완료되면 활성화된다. 서비스를 이용하기 위해서는 활성화된 서비스 아이콘을 클릭하면 별도의 창에서 서비스가 실행된다.

MyKSC 의 서비스 생성 방식은 다음과 같다. 먼저 사용자가 로그인할 경우 MyKSC 는 인증 서버에서 사용자의 UID, GID, 사용자 이름과 같은 사용자의 기본 정보를 가져와 네임 스페이스(namespace)를 생성한다. MyKSC 에서는 로그인한 사용자가 생성하는 모든 서비스는 네임스페이스를 통해 생성 관리된다. MyKSC 를 통해 제공되는 서비스는 그 용도에 따라 아래 3 가지로 구분할 수 있다.

① **프로그램 개발 및 데이터 분석**: 사용자는 프로그램 코드 작성 및 데이터 분석을 위해 Jupyter, RStudio, VNC, VSCode 를 이용할 수 있다. 사용자가 Jupyter, RStudio, VNC, VSCode 서비스 생성을 요청할 경우 myKSC 는 kube api-server 를 통해 파드와 서비스(service)를 생성한다. 파드 생성 시 보안을 위해 'runAsUser' 와 'runAsGroup' 값을 인증 서버에서 얻어진 사용자의 UID, GID 값으로 설정한다. 이렇게 생성된 파드는 istio 의 API Gateway 를 통해 클라이언트와 연결된다.

② **데이터 관리**: 사용자는 Filebrower 응용을 통해 GUI 기반으로 파일 관리할 수 있다. 이를 위해 Filebrower 서비스 생성 시 MyKSC 에서는 호스트 패스(hostPath) 형태로 Filebrower 파드에

KISTI 슈퍼컴퓨터에서 제공하는 2 개의 개인 디렉토리를 마운트해준다.

③ **HPC 작업 제출 및 관리**: MyKSC 의 경우에는 제 3 절에서 언급한 바와 같이 기존의 HPC 클러스터 시스템과 쿠버네티스 클러스터 시스템을 병행으로 구축하였다. 그러므로 MyKSC 의 BatchJob 서비스를 통해 기존의 HPC 클러스터에 작업을 제출하고 모니터링 할 수 있다. 즉, BachJob 는 GUI 통해 사용자가 입력한 큐 이름, 입력 파라미터, 실행 명령어 등을 기반으로 작업 스크립트 파일을 생성한다. 사용자가 작업 제출 버튼을 클릭하면 생성된 작업 스크립트 파일을 HPC 클러스터의 작업 스케줄러로 제출함으로써 실제 작업은 HPC 클러스터에서 실행된다.

4. 결론

본 논문에서는 KISTI 에서 운영하는 슈퍼컴퓨터를 대상으로 컨테이너 기반의 서비스 제공을 위한 MyKSC 를 소개하였다. MyKSC 경우에는 기존 HPC 클러스터의 소프트웨어 스택의 변경 없이 유연한 서비스 제공을 위해 약 결합 아키텍쳐 기반으로 설계되어 적용되었다. 즉, 전체 시스템 자원을 기존의 스케줄러 기반의 HPC 클러스터와 새로운 서비스 제공을 위한 쿠버네티스 클러스터로 분리하여 구축하였다. 이를 통해 빠른 서비스 배포 및 사용자 맞춤형 서비스가 기능해졌다. 또한 모든 서비스를 웹 기반의 GUI 형태로 제공함에 따라 슈퍼컴퓨터 사용의 진입 장벽을 낮출 수 있었다.

사사

이 논문은 2024 년도 한국과학기술정보연구원(KISTI)의 기본사업으로 수행된 연구입니다. (과제번호: K-24-L2-M1-C1)

참고문헌

[1] Yoo, A. B., Jette, M. A., and Grondona, M., "Slurm: Simple linux utility for resource management," in Proc. of Workshop on Job Scheduling Strategies for Parallel Processing, pp. 44-60, June 2003.

[2] Feng, H., Misra, V., and Rubenstein, D., "PBS: a unified priority-based scheduler, " in Proc. of the 2007 ACM SIGMETRICS Int. Conf. on Measurement and Modeling of Computer Systems, pp. 203-214, June 2007.

[3] Odun-Ayo, I., Ajayi, O., and Okereke, C., "Virtualization in cloud computing: Developments and trends, " in Proc. of 2017 Int. Conf. on Next Generation Computing and Information Systems, pp. 24-28, Dec. 2017.

[4] Y.-M. Bae, S.-J. Jung, and W.-Y. Soh, "Comparative Analysis of the Virtual Machine and Containers Methods through the Web Server Configuration, " Journal of the Korea Institute of Information and Communication Engineering, vol. 18, no. 11. pp. 2670–2677, 30-Nov-2014.

[5] Abraham, S., Paul, A. K., Khan, R. I. S., and Butt, A. R., "On the use of containers in high performance computing environments," in Proc. of 2020 IEEE 13th Int. Conf. on Cloud Computing, pp. 284-293, Oct. 2020.

[6] Keller Tesser, R., and Borin, E., "Containers in HPC: a survey," The Journal of Supercomputing, 79(5), 5759-5827, 2023.

[7] Boettiger, C., "An introduction to Docker for reproducible research," ACM SIGOPS Operating Systems Review, 49(1), 71-79, 2015.