2020 한국차세대컴퓨팅학회
하계학술대회

장　　소 : 제주 JDC(엘리트빌딩)

일　　시 : 2020. 8. 20(목) 14:00 ~ 8. 22(토) 12:00

주최 · 주관 : 한국차세대컴퓨팅학회, 제주드론산업협회

Change Point Detection 기반의 효율적인 자원 사용량 데이터 압축 프로세스

Efficient Data Compression Process Based on Change Point Detection

이병용[1], 박기웅[2,*]

Byeong-Yong Yi, Ki-Woong Park

[1](05009) 서울시 광진구 능동로 209, 세종대학교 시스템보안연구실

[2](05009) 서울시 광진구 능동로 209, 세종대학교 정보보호학과

yibyeongyong@sju.ac.kr, woongbak@sejong.ac.kr

요 약

데이터 압축 기법은 데이터를 인코딩함으로써, 저장공간을 효율적으로 사용하는 방법의 하나다. 최근 클라우드 컴퓨팅, IoT 등으로 인해 거대해진 규모의 컴퓨팅 환경에서 발생하는 막대한 양의 로그 데이터를 효율적으로 압축하기 위한 연구가 계속되고 있다. 압축 기법에 관한 연구들은 로그 데이터가 특정 규격을 가지고 기록된다는 점에 주목하여 데이터 압축 전에, 유사한 데이터를 한데 모아 압축 성능을 높이는 데이터 클러스터링 기반의 압축 연구를 주로 수행해왔다. 본 논문은 압축 성능을 높이는 방법에 대한 관점을 달리하여, 구간별 엔트로피 기반의 새로운 압축 프로세스를 제안한다. 제안하는 방법은 전체 자원 사용량 데이터에서 데이터가 급격하게 변하는 시점인 Change Point를 기반으로 구간을 나눔으로써, 각 구간의 데이터 엔트로피를 낮추고 이를 통해 압축 성능을 높인다. 제안하는 방법은 기존에 제안된 압축 기법들과 함께 사용할 수 있어, 압축 성능을 한층 높일 수 있을 것으로 예상한다.

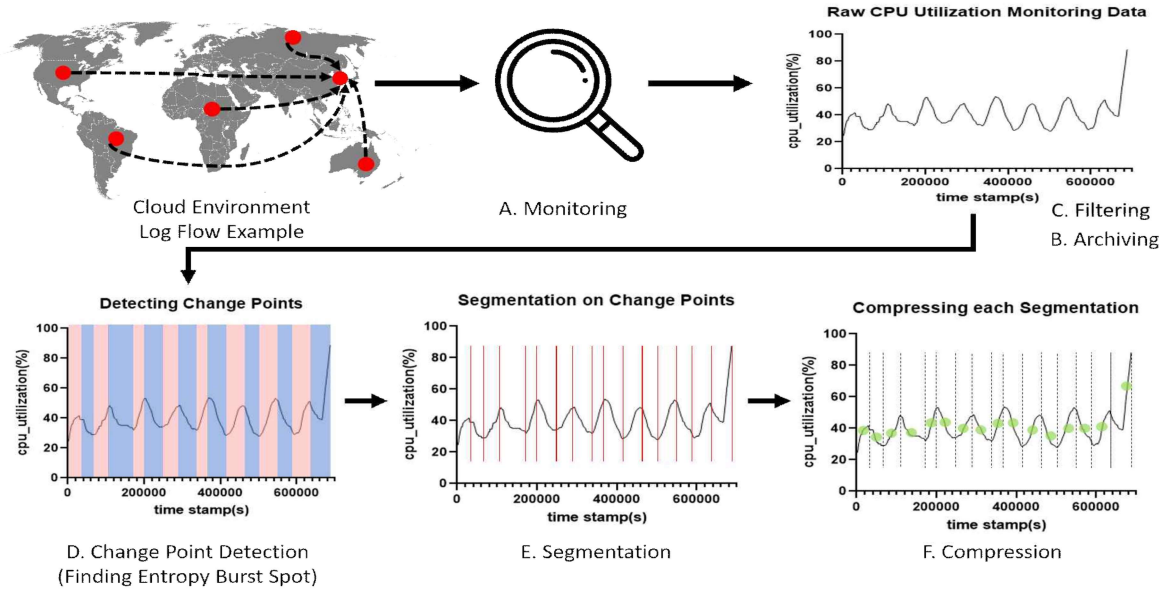
키워드: Resource usage, Data, Log, Compression, Change Point Detection, Entropy

1. 서론

로그 데이터는 시스템 관리에 사용되는 활용도가 높은 데이터이다. 그중에서도 자원 사용량 모니터링 데이터는 성능 분석[1], 시스템 이상 탐지[2] 등의 중요한 요소에 다방면으로 활용 가능한 유용한 데이터이다. 하지만, 자원 사용량 모니터링 데이터는 사전 설정된 주기에 따라서 끊임없이 기록되기 때문에, 숫자로만 이루어진 데이터일지라도 수집 시간에 따라서 상당히 많은 저장공간을 차지하는 경우가 많다. 이렇게 많은 양의 데이터를 차지하는 자원 사용량 모니터링 데이터를 효율적으로 관리하기 위해서 일반적으로 압축 기법을 사용하고 있으며, 압축률과 압축 시간으로 대표되는 압축 성능을 높이기 위해 다양한 압축 기법이 연구되고 있다. 압축 기법은 원본 데이터의 엔트로피에 따라서 압축 성능이 좌우되는 경향이 있다[3]. 일반적으로 같은 압축 기법이라고 해도 엔트로피가 높을수록 압축 성능이 낮으며, 엔트로피가 낮을수록 압축 성능이 높다. 자원 사용량 데이터를 포함하는 로그 데이터의 압축 성능을 높이기 위해서, 최근 연구들은 전체 데이터에서 유사한 데이터를 클러스터링함으로써 엔트로피를 낮추는 방법을 통해서 압축 성능을 높이는 데이터 전처리하고, 압축을 병렬적으로 수행하는 연구가 수행되기도 하였다[4].

본 논문은 자원 사용량 모니터링 데이터에서 엔

* 교신저자



(그림 1) 클라우드 환경에서 발생하는 자원 사용량 모니터링 데이터의 수집, 축적, 급변지점 탐지, 시계열 데이터 구간 분할, 개별 압축으로 구분된 제안하는 압축 프로세스

트로피가 급격하게 변하는 지점을 기준으로, 구간을 나누어 효율적인 압축을 수행하는 압축 프로세스를 제안한다. 엔트로피가 급변하는 지점은 데이터가 급격하게 변하는 지점을 찾아내는 CPD(Change Point Detection) 기법을 통해 확인하였으며, 해당 기술을 본 논문의 테스트 데이터에 적용하기 위해서 다양한 CPD 기법이 구현된 파이선 Rupture[5] 모듈을 활용하였다. CPD 기법을 통해 도출된 을 기점으로 데이터를 나눌 때, 실제로 압축 효율이 상승하는지에 대한 검증을 수행한다.

본 논문은 아래와 같이 구성되어 있다. 2장은 관련 연구를 소개한다. 3장은 제안하는 압축 기법을 구체적으로 설명하고, 가설들을 증명하기 위한 실험을 설계한다. 4장은 실험을 통해, 본 논문이 제안하는 방안의 우수성을 증명한다. 5장은 결론과 추후 연구에 관해서 서술한다.

2. 관련 연구

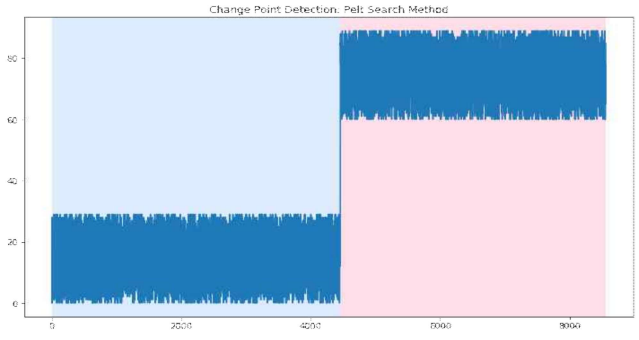
압축 기법의 성능을 높이기 위한 연구는 다양한 방면에서 시도되었다. 그중 Skretting et al[6]. 연구진은 무손실 압축에서 사용하는 허프먼 코딩 방식[7]에서 연속적인 신호 데이터를 작은 데이터로 나눌수록 압축률이 높아질 수 있다는 것을 보였다. 또한, BB Alagoz[8]는 전체 데이터에서 낮은 엔트로피를 가지고 있는 작은 데이터의 집합으로 구간을 나눔으로써, 압축 성능이 높아질 수 있음을 수학적으로 증명하였다.

CPD(Change Point Detection)는 주로 통계학에서 사용되어온 개념이며, 시계열 데이터에서 갑작스러운 값의 변화가 나타나는 지점을 찾는 탐지 기법이다[9]. 을 찾는 시점에 따라서 실시간으로 찾는 온라인 CPD와 오프라인 CPD로 나눌 수 있다. 온라인 CPD는 실시간으로 급변 시점을 찾을 수 있다는 장점이 있으나 탐지 정확도가 오프라인 CPD에 비교해서 상대적으로 낮다. 오프라인 CPD는 전체 데이터를 확보한 후에 탐지할 수 있다는 단점이 있지만, 온라인 CPD보다 탐지 정확도가 상대적으로 높다. 본 논문은 실제 로그 데이터의 저장공간에서 이루어지는 데이터의 축적, 압축 과정과 유사한 상황을 가정하기 위하여 오프라인 CPD를 사용하였다.

3. 제안 방법

3.1 CPD 기반의 압축 프로세스 설계

본 논문이 제안하는 압축 프로세스의 기반이 되는 가설은 CPD를 통해 확인한 시점이 곧 엔트로피가 급변하는 지점이며, 이러한 정보를 기반으로 데이터를 분할 압축하였을 때, 효율적인 압축이 가능하다는 것이다. 이를 증명하기 위한 실험 수행한



(그림 2) 샘플 데이터를 대상으로 CPD의 Pelt Search Method를 통한 데이터 급변지점 탐지 결과

절차는 아래 (그림 1)과 같다. 클라우드 환경에서 자원 사용량 데이터에 대한 모니터링을 수행하고(A. Monitoring), 이에 대한 Raw 자원 사용량 자료를 축적하는 아카이빙 과정(B. Archiving)을 수행한다. machine_usage.csv 데이터의 9가지 데이터 중, CPU 사용률에 대한 정보가 기록된 cpu_util을 대상으로 데이터를 필터링 후(C. Filtering) 실험을 수행하였다.

엔트로피는 클라우드 새넌이 제안한 개념으로써, 정보의 불확실성을 나타내는 값이다[10]. 엔트로피값이 높을수록 더욱 복잡한 데이터라고 할 수 있고 압축 성능이 낮을 가능성이 크며, 반대로 엔트로피값이 낮을수록 더욱 단순한 데이터라고 할 수 있고, 압축 성능이 높을 가능성이 크다. 제안하는 압축 프로세스는 이러한 엔트로피와 압축 기법의 특징을 활용하여, 전체 자원 사용량 모니터링 데이터에서, 엔트로피를 최대한 낮출 수 있는 구간을 하나의 세그먼테이션으로 하여 각자 압축을 수행하는 것이며, 그 구간 정보는 CPD를 통해 확보한다. (그림 2)의 샘플 데이터를 대상으로 Pelt Search Method[12]를 적용한 CPD 기법을 사용할 경우, 그림 (그림 3)과 같이 데이터 급변지점을 찾아 구분하는 것을 확인할 수 있다.

4. 실험

4.1 실험 환경

실험을 진행한 서버 환경은 아래 표 1과 같으며, 압축 도구는 gzip[11](버전 1.6)을 사용하였고, 압축 시간은 perf[12](버전 5.8)을 통해 측정하였다.

<표 1> 실험 환경

구분	상세 환경
CPU	Intel(R) Xeon(R) Silver 4114 CPU
Memory	376GB
OS	Ubuntu 5.4.0-42-generic

4.2 실험

기반 가설을 증명하기 위한 절차는 (그림 2)와 같이 분포한 임의의 데이터를 생성함으로써 증명한다. 본 논문에서 제안하는 압축 프로세스의 효율성을 간단하게 증명하는 예시이다. 현재 (그림 2)에서는 0에서 30까지의 범위에서 임의의 값을 갖는 A 구간과 60에서 90까지의 범위에서 임의의 값을 갖는 B 구간, 그리고 전체 구간을 나타내는 T 구간으로 구성되어 있다. 이 경우, A 구간의 엔트로피를 H_1, B 구간의 엔트로피를 H_2, 전체 구간 T의 엔트로피를 H_3로 정의한다. A 구간과 B 구간 각각에 대해서 엔트로피를 계산했을 때(H_1와 H_2의 합) 전체 A 구간과 B 구간의 합인 전체 구간 T에 대해서 엔트로피를 계산할 때(H_3)를 비교한다. 압축을 수행하기 전에, 엔트로피의 크기를 비교하는 이유는 엔트로피의 크기에 따라서 압축 성능이 좌우될 가능성이 매우 크기 때문이다. 정해진 범위에서 임의로 숫자로 구성된 같은 크기의 A 구간과 B 구간 파일 그리고 그 합인 T 구간 파일을 파이선의 랜덤 모듈을 통해 생성하였으며, 특정 파일의 엔트로피를 계산하는 단순한 파이선 프로그램을 통해서 엔트로피를 계산하였다. 성능 비교는 로그 데이터 압축에 널리 사용 중인 gzip 압축 기법을 통해 각 구간 파일을 압축함으로써 압축 성능을 비교하였다. 엔트로피 계산 결과와 압축 성능을 비교하면 <표 2>와 같은 결과를 확인할 수 있었다. 결괏값을 확인해보면 A와 B 구간을 따로 압축하였을 때, 약 2.9%가량 손해가 있음을 확인할 수 있었으나, 압축 시간은 A 구간과 B 구간을 따로 압축할 때가 전체 구간 T를 압축할 때의 약 31%로서 성능 향상이 있음을 확인할 수 있었다. 이러한 결과가

<표 2> 임의로 생성한 데이터의 엔트로피 비교(gzip)

구분	엔트로피	압축 전 크기(bytes)	압축 후 크기(bytes)	압축률 (%)	압축 시간(초)
A	4.9004	16392	4363	26.61	0.0019
B	4.9026	16386	4007	24.45	0.0019
T	5.9002	32778	8130	24.80	0.0121

나타난 이유는 원본 자원 사용량 모니터링 데이터 파일을 두 파일로 나누면서, 헤더 값의 증가로 압축률 상의 손실이 약간 발생하나, 파일을 나눔으로써 엔트로피를 낮추어 압축이 수월하도록 일종의 클러스터링을 수행하여, 압축 시간은 낮아진 것으로 추정된다.

5. 결론

본 논문은 대용량 로그 데이터를 압축하기 위해서 CPD 정보 기반의 데이터 분할을 통한 효율적인 압축 프로세스를 제안하였으며, 효율성의 기반이 되는 가설을 간단한 실험을 통해 실제로 효율이 있음을 확인하였다. 제안하는 프로세스는 기존에 공개된 gzip 등의 압축 도구를 사용하기 전에 엔트로피를 낮추기 위한 일종의 데이터 전처리를 수행함으로써 효율적인 압축을 수행할 수 있어, 압축 성능을 한층 끌어올릴 수 있었다. 추후 연구에선, 더 다양한 압축 기법을 대상으로, 임의의 데이터가 아닌 실제 자원 사용량 수집 데이터를 대상으로 제안하는 압축 기법이 효율성이 있는지 실험을 진행할 예정이다. 또한, 실시간으로 축적되는 자원 사용량 모니터링 데이터를 오프라인 CPD가 아닌 온라인 CPD로 데이터 을 탐색하여, 실시간성을 갖춘 압축 프로세스를 연구할 예정이다.

Acknowledgement

본 연구는 2019년도 과학기술정보통신부의 재원으로 정보통신기획평가원의 지원(No.2018-0-00420, No.2019-0-00273) 및 한국연구재단 연구과제(NRF-2017R1C1B2003957)의 지원을 받아 수행된 연구임.

참고문헌

[1] Dhingra, Mohit, J. Lakshmi, and S. K. Nandy. "Resource usage monitoring in clouds." 2012 ACM/IEEE 13th International Conference on Grid Computing. IEEE, 2012.

[2] Chuah, Edward, et al. "Linking resource usage anomalies with system failures from cluster log data." 2013 IEEE 32nd International Symposium on Reliable Distributed Systems. IEEE, 2013.

[3] Cover, Thomas M. Elements of information theory. John Wiley & Sons, 1999.

[4] Liu, Jinyang, et al. "Logzip: extracting hidden structures via iterative clustering for log compression." 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2019.

[5] Rupture. https://github.com/deepcharles/ruptures

[6] Skretting, Karl, John Håkon Husøy, and Sven Ole Aase. "Improved Huffman coding using recursive splitting." Proceedings of Norwegian Signal Processing, NORSIG. 1999.

[7] Huffman, David A. "A method for the construction of minimum-redundancy codes." Proceedings of the IRE 40.9 (1952): 1098-1101.

[8] Alagoz, B. Baykant. "Effects of Sequence Partitioning on Compression Rate." arXiv preprint arXiv:1011.0338 (2010).

[9] Basseville, Michèle, and Igor V. Nikiforov. Detection of abrupt changes: theory and application. Vol. 104. Englewood Cliffs: prentice Hall, 1993.

[10] Shannon, Claude E. "A mathematical theory of communication." The Bell system technical journal 27.3 (1948): 379-423.

[11] gzip. https://www.gzip.org/

[12] perf. https://github.com/torvalds/linux/tree/master/tools/perf

[13] Killick, Rebecca, Paul Fearnhead, and Idris A. Eckley. "Optimal detection of changepoints with a linear computational cost." Journal of the American Statistical Association 107.500 (2012): 1590-1598.