머신러닝을 활용한 NFV 시스템 로그 분석

오성근, 유헌창
고려대학교 컴퓨터정보통신대학원
e-mail : {ohsk, yuhc}@korea.ac.kr

NFV Log Analysis using Machine Learning

SeongKeun Oh, HeonChang Yu
Graduate School of Computer & Information Technology, Korea University

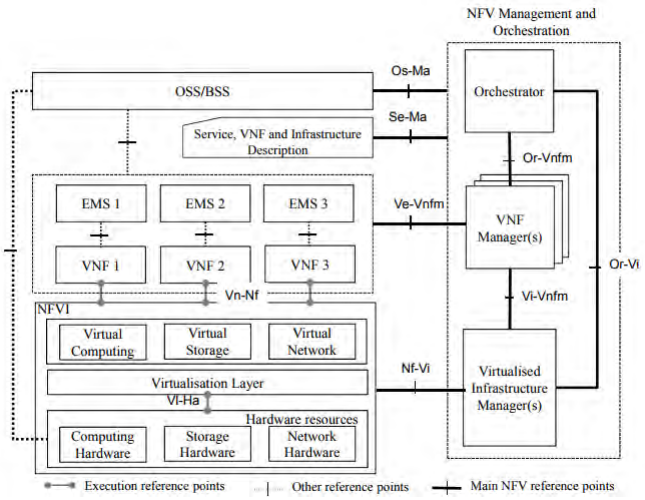
요 약

모바일 이동통신망의 Core 노드들은 2G CDMA, 3G WCDMA, 4G LTE 교환기를 비롯하여 IMS 및 다양한 부가장비들로 이루어져 있다. 최근 5G 로 진화하는 과정에는 NFV(Network Function Virtualization)가 그 중심에 서 있다. NFV 환경에서는 기존 통신 노드와 다르게 범용서버 및 범용 운영체제가 주축이 되어, 일반 IT 툴로도 통신망 내부 노드의 로그분석이 용이해 졌다. 또한 다양하고 복잡한 Core 네트워크에서 빅데이터로 발생하는 로그 또한 머신러닝으로 분석이 가능하며, 운용에 활용할 수 있다. 따라서 본 연구에서는 vDPI, vMMSGW OS 로그를 대상으로 분석하였으며, 잠재되어 있는 문제점들을 확인할 수 있었다. 또한 어플리케이션의 비정형화 된 로그에서도 비정상적인 패턴들을 발견하여 대용량 트래픽이 발생하며 SLA 가 유난히 높은 통신환경에서도 비지도 머신러닝 분석이 유용함을 확인하였다.

1. 서론

최근 교환기를 비롯한 이동통신망 네트워크의 Core 영역의 노드들은 NFV(Network Function Virtualization) 시스템으로 진화해 가고 있다[1]. NFV 는 Capex(Capital expenditures), Opex(Operating Expenses) 등을 줄일 수 있는 좋은 기술로서 소개되곤 하는데, NFV 로 발전하게 되면서 이 구조 특성상 기술적인 변화뿐만 아니라 운용환경에서도 많은 변화가 뒤따르고 있다. 그 변화들 중, 전용장비에서 범용서버로의 변화도 크게 바뀌는 점 중 하나이다. NFV 에서는 범용 서버로도 전통적인 통신장비들과 동등한 빠른 통신속도를 제공하기 위해 DPDK(Data Plane Development Kit), Pass-through 등의 기능들이 도입 되어 범용서버와 범용 운영체제· 리눅스 기반으로도 통신노드가 동작할 수 있게 되었다. 즉, 하이퍼바이저를 포함한 NFVI 영역 (Network Function Virtualization Infrastructure), GuestOS 가 위치하며 VNF 를 이루는 요소인 VNFC 영역 (Virtual Network Function Component) 등에 모두 범용 운영체제가 자리잡게 된 것이다. 덕분에 이 장비들의 운용 로그들도 기존 IT 툴로 용이하게 분석이 가능해 졌다. 특히 본 연구에서는 통신서비스와 맞닿아 밀접한 VNFC 영역의 범용 리눅스 로그를 다양한 측면에서 필터 검색 및 시각화하고, 머신러닝 기술을 활용하여 분석하였다. 그리고 그 결과로 안정적인 NFV 시스템 운용에 범용적인 운영체제 로그분석 및 머신러닝 로그분석이 유용한 것을 확인하였다.

(그림 1)은 NFV 를 주도적으로 이끌어가는 ETSI NFV ISG 에서 제안한 모델이며 상당수의 사업자들이 이 모델을 기반으로 NFV 를 설계하고 있다[2].



(그림 1) NFV 구조

2. 관련연구

[3]에서는 오픈스택을 기반으로 이상징후를 감지할 수 있는 구조에 대해서 설명하였다. 하지만 이 연구에서는 프로토타입 VNF(Virtual Network Function) 선정에 있어 OPNFV(OpenNFV) 중 vIMS(virtual IP

Multimedia System) 인 Clearwater 프로젝트에 1 개에 대해서만 데이터를 분석하였고, 분석 또한 메트릭 데이터 중심으로만 이상현상을 분석하였다.

3. VNF 의 범용 리눅스 로그 및 메트릭 데이터 기반 이상 징후 확인

3.1 구성환경

분석을 위해 VNF(Virtual Network Function) 2 개를 선정하였다. vDPI(virtual Deep Packet Inspection)은 네트워크 트래픽을 조정하기 위해 쓰이는 기술 및 장비이며, PGW(Public data network GateWay) 에서도 규격상 일부 지원할 수 있지만, 이보다 더 많은 패킷을 분석할 수 있으며 추가 기능을 가지고 있다. vMMSGW(virtual Multimedia Messaing Service GateWay)는 멀티미디어 메시지를 보내는데 필요한 장비로 LTE Core 기준으로 PGW 와 MMSRelay 등 메세징 처리장비 사이에서 게이트웨이 역할을 한다. 각 어플리케이션이 동작되고 있는 기본 환경은 <표 1>과 같다.

<표 1> 로그분석에 사용한 vDPI, vMMSGW 의 환경

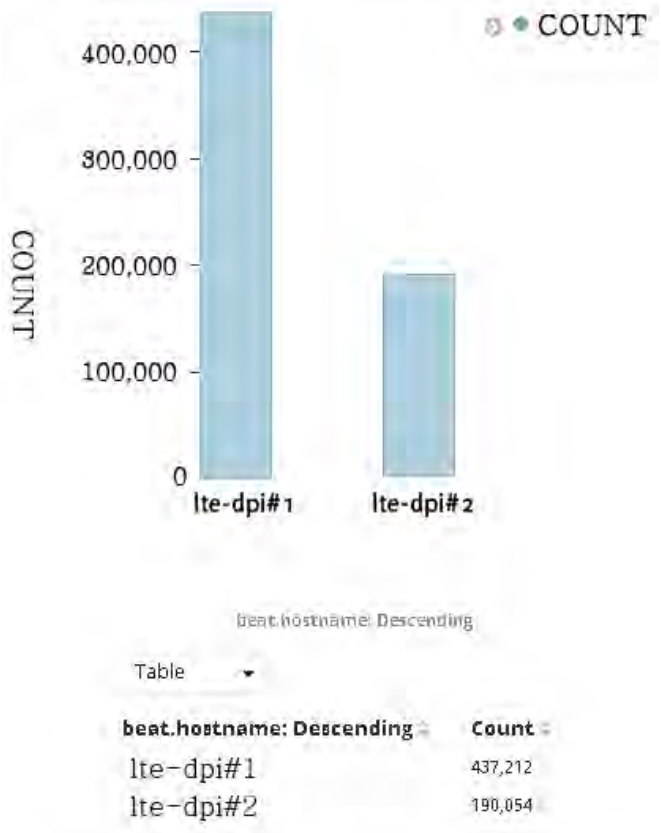
		vDPI	vMMSGW
물리 서버	CPU	44 core * 2.2Ghz	24core * 2.497Ghz
	메모리	256GB	256GB
	스토리지	600GB	10TB
	하이퍼바이저	Linux KVM QEMU 2.6.0 (libvirt 1.2.16)	VMware ESXi 6.0
가상 서버	가상 CPU	20core	8core
	가상메모리	32GB	32GB
	스토리지	150GB	150GB
	GuestOS	CentOS7.2	CentOS6.4

3.2 Elastic 머신러닝(Prelert)을 이용한 로그분석

로그 분석에 유용한 솔루션인 Elastic Stack [4]은 ElasticSearch, LogStash, Kibana, Beats 로 이루어진 오픈소스이다. 데이터 수집(Beats, LogStash)부터 분석(Elastic Search)을 거쳐 시각화(Kibana)에 이르기까지 유기적으로 연동하여 서비스를 제공하며 2017 년 3 월 1 억건 다운로드를 기록하였다. Prelert 는 2009 년 미국 보스톤에서 설립된 비지도 머신러닝 기반의 이상징후 감시 솔루션이자 회사명이며, 2016 년 Elastic 에 인수되었다. 이후 Prelert 는 2017 년 5 월, Elastic Stack 에 완전히 통합되었다.[3] 본 로그 분석에서는 Elastic Stack 의 로그 분석기능과 머신러닝을 활용하여 잠재되어 있는 문제와 이상징후 패턴을 발견하였다.

3.3 분석사례 1: 기본 로그 분석을 통한 잠재 문제 발견

/var/log 는 리눅스에서 기본적으로 사용하는 로그들이 주로 위치하는 디렉토리이다. 본 분석에서는 이 디렉토리를 기준으로 VNFC 에 각각 설치되어 있는 Beats (Filebeat, MetricBeat) 에이전트가 ElasticSearch 에 해당 로그 메시지들을 전송하는 구조로 설정하였다. 일반적으로 통신노드의 로그들은 대용량 트래픽에 따라 상당히 많이 쌓이기 때문에 파티션과 디렉토리를 별도로 설정하여 마운트 하는 경우가 많고, /var/log 의 디렉토리는 리눅스 운영체제와 일부 시스템 데몬의 로그들이 주로 쌓이게 된다. 이 때 통신노드의 역할과 구조에 대해 알고 있다면 로그의 이상유무에 대해 분석을 하기에 용이하다.

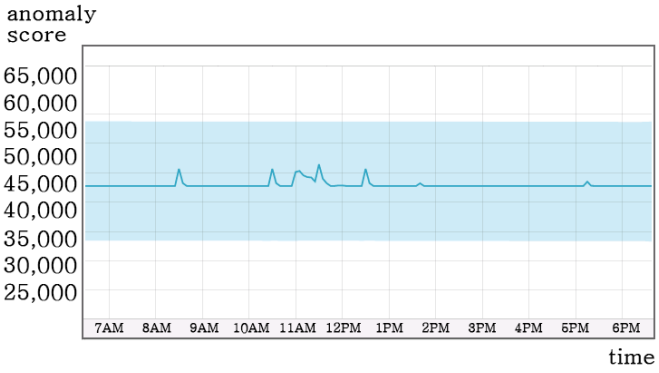


(그림 2) OS 로그 분석을 통한 잠재문제 발견

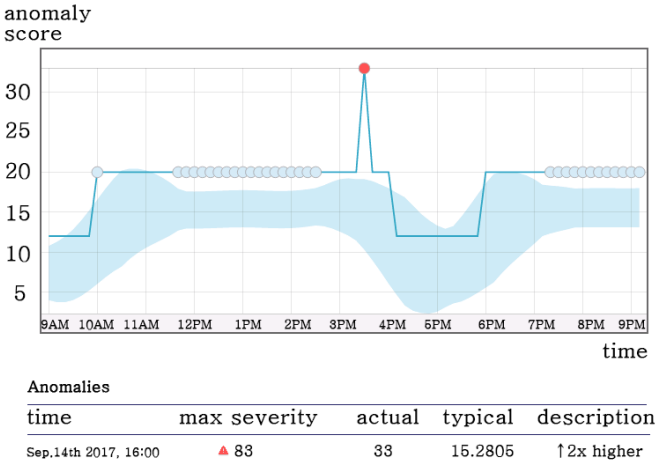
(그림 2)에서는 일주일간 생성된 생성된 VNFC 중 두 vDPI 로그들을 표시해보았다. 두 시스템이 합쳐 일주일간 60 여만여건의 로그 카운트가 쌓이게 되었다. 그런데 특이한 사항은 vDPI#1 노드와 vDPI#2 노드가 Act/Act 구조로 설계되어 있는데 vDPI#1 만 43 만 7 천건으로, vDPI#2 의 19 만건에 비해 유난히 로그가 많이 쌓여져 있는 것이다. 동일한 물리 서버 모델, 동일한 하이퍼바이저, 동일한 GuestOS 버전, 동일한 어플리케이션 구조 등으로 구축한 시스템인데도 불구하고, 특정 노드에서만 로그가 비정상적으로 많이 쌓이는 현상은 의심을 불러 일으킬 수 있다. 위와 같은 문제를 시간대별로 나누어 분석해보니, vDPI #1 에 유입되는 패킷 내 특정 모바일가입자의 IMEI (International Mobile Equipment Identity) 번호가 규격에 맞지 않아 vDPI 가 분석하면서 주의 단계의 로그를 많이 발생시킨 것이 원인이었다. 이 외에도 vDPI 특성상 정책을

관리하는 PCRF(Policy Charging Rule Function) 등과 연동할 수도 있는데, 연동이 비정상적인 경우에도 연동 관련 로그 등을 통해서 비정상 케이스를 확인할 수 있었다.

3.4 분석사례 2: 머신러닝(Prelert) 을 활용하여 이상징후 패턴 발견



(그림 3) vMMSGW 리소스 패턴



(그림 4) 이상징후 패턴 발견

(그림 3)은 vMMSGW 의 CPU/메모리/디스크/네트워크 등 기본 리소스에 대한 메트릭 패턴을 머신러닝으로 분석한 결과이다. 위 데이터에서는 vMMSGW 가 상용이 아닌 테스트용도의 단말에 대해서만 호처리한 상태이며 모델링이 정교하게 되지 않은 초기상태의 분석이라 큰 패턴의 움직임은 보여지지 않았다. 한편 (그림 4)는 vDPI 와 vMMSGW 를 같은 로그 데이터로 묶고 두 로그 데이터의 패턴을 머신러닝 기법으로 분석한 결과이다. 이와 같이 두 어플리케이션의 성격이 전혀 다르더라도, 운용환경에 따라서 한번에 같은 경로의 디렉토리를 대상으로 묶어서 모델링해 볼 수도 있다. 물론 종합해서 묶어 보는 경우에는 원인 분석 필요 시, 추가적인 분석단계가 필요할 수 있겠지만, 많은 VNF 가 존재하는 환경에서는 이와 같은 종합감시, 분석이 용이할 수도 있을 것이다. (그림 4)의 데이터의 의미를 확인하면, 음영부분이 그 동안의 데이터를 모델링 하여 예측하는 정상범위 임계치이다. 그리고 이 영역을 벗어나는 유난히 튄 시점이 있는데, Anomaly score 가 30 점이 넘은 오후 3 시 데이터 이상징후를 확인해 보았다. 실제로 (그림 4)의 이상현상은 해당 vDPI#1,2 장비의 인수시험에서 고의적으로 장애를 낸 케이스로 연동이 비정상적이어서 vDPI 에서 발생한 로그였다. 위와 같이 머신러닝 기법을 이용하면 특별히 알람 및 로그를 명시하지 않더라도 이상현상에 대해서 추가적으로 감시 및 분석을 할 수 있는 근간을 마련할 수 있고, 운용자가 미처 지정하지 못한 영역도 감시하며 분석할 수 있는 토대가 될 수 있다.

4. 결론 및 향후 연구

본 연구에서는 vDPI 와 vMMSGW 를 대상으로 NFV 환경에서 VNFC 의 다양한 로그들에 대해 분석하여 잠재 문제에 대한 원인을 파악하였다. 또한 각 VNF 간의 로그를 머신러닝으로 분석하여 이상현상에 대해서 확인하고, 통신 노드에도 머신러닝이 적용될 수 있음을 확인하였다. 본 논문은 이동통신사업과 관련된 시스템에 대해서 한정지었지만, 이런 사례들은 각 산업군의 특성에 맞게 분석할 수 있는 문제이다. 다만 이동통신 산업 군에서는 그 산업 특성상 벤더에 종속적인 전용장비가 많아, 그 동안 일반 IT 솔루션 및 오픈소스 등으로는 분석할 수가 없었는데, 최근 NFV 로 발전됨에 따라 본 연구와 같이 분석할 수 있는 토대가 만들어졌다. 이 논문에서는 VNFC 로그들을 중심으로 분석이 이루어졌지만 사실상 NFV 를 이루는 영역에는 VNFC 외에도 NFVI 및 MANO (Management & Orchestration) 영역의 로그 또한 존재하고 있다. 향후 연구에서는 NFVI, VNFC, MANO 전체적인 영역의 로그를 종합하고 분석하여 운용상 성능, 효율성 또는 안정운용 차원에서 개선할 수 있는 방안에 대한 연구를 수행할 계획이다.

참고문헌

[1] ETSI ISG NFV, "Network Functions Virtualization-Introductory White Paper #1,#2, #3, 2014.

[2] ETSI NFV ISG, “Network Functions Virtualization (NFV) Architectural Framework,” ETSI GS NFV 002 V1.1.1, October 2013.

[3] Anton Gulenko, Marcel Wallschlager, Florian Schmidt, Odej Kao, Feng Liu, “A system architecture for real-time anomaly detection in large-scale NFV systems”, International Workshop on Applications of Software-Defined Networking in Cloud Computing, 2016.

[4] elastic stack. http://www.elastic.co