New Relic®

New Relic Node VM 지표 활용

2021 년 1 월 26 일

개요

Node VM 의 지표 확인이 왜 필요한 것인지 검토하고 뉴렐릭 서비스를 활용하여 Node VM 지표를 어떻게 확인하고 활용할 수 있는 지 알아봅니다.

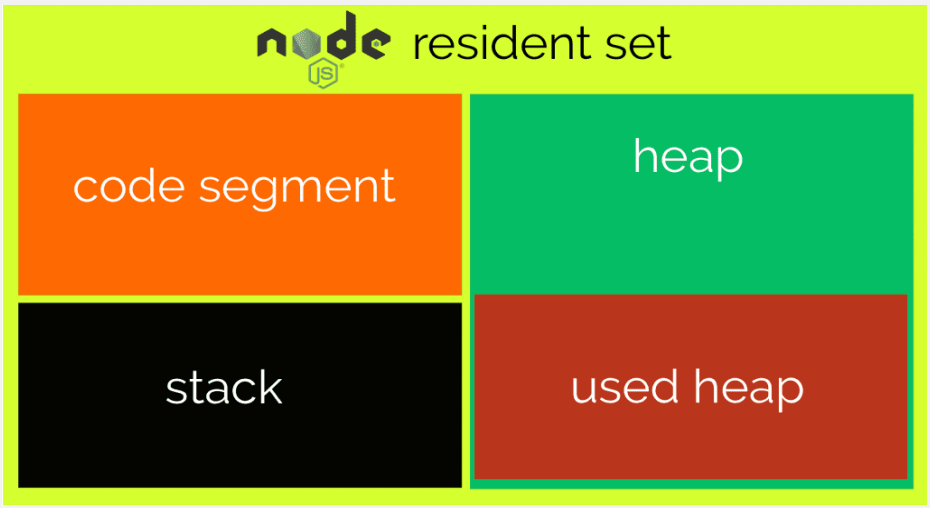
작성

뉴렐릭 코리아 기술 본부

2021 년 1 월

내용

Node VM 이해와 메모리



Node VM 힙은 객체, 문자열 및 구분자를 저장하는 데 사용되는 메모리 세그먼트입니다. 실행 중인 node.js 프로세스는 자신의 모든 메모리를 레지던트 세트 안에 저장합니다. 레지던트 세트 안에는 실제 자바스크립트 코드(코드 세그먼트 내)와 모든 변수의 영역인 스택도 포함되어 있습니다. 위 그림은 이러한 개념을 도식화 합니다.

Node.JS 애플리케이션을 실행하는 도중에 발생하는 문제를 해결하기 위해서 Node VM 의 차트를 이해해야 할 필요성이 대두됩니다. 애플리케이션에 문제를 야기하는 문제 발생 동기 여부는 고객의 접속과 서비스 이용의 증가에 의한 문제와 아울러 서비스를 실행하는 인프라 스트럭쳐의 컨테이너(도커, 가상머신, 베어메탈 서버등 애플리케이션이 수행되는 환경의 통칭) 들의 변화에 의해서도 발생하게 되며, 특히 후자의 경우 애플리케이션에서 나타난 현상과 연관되어진 충격을 Node VM 이 반영하고 있을 수 있기 때문입니다.

예) AWS 스팟 인스턴스를 활용하여 오토 스케일링을 사용하는 도중, 애플리케이션이 필요로 하는 경우 보다 부족한 메모리를 가진 인스턴스가 생성, 할당되어졌다고 하는 경우, NodeVM 은 애플리케이션 수행을 위한 이전 인스턴스에서는 필요로 하지 않는 부가적인

NodeVM 운영 활동이 증가합니다. 힙의 확보라던가, 스택의 확보와 같은 활동이 요구되어지고 이는 Node VM 의 GC 활동 활성화로 전파됩니다. 결과적으로 이러한 GC 활동은 NodeVM 의 부가적인 CPU 활동을 가동하게 합니다.

예) 사용자의 애플리케이션의 구조적 결함으로 인하여 메모리 릭 현상(메모리를 지속적으로 할당만 하고 반환하지 않는 현상)이 발생한다면 서비스 초기에는 나타나지 않는 GC 문제가 상당 시간 지나서 발생함에 따라 문제를 조기에 발견하기 어려울 수 있습니다. 이러한 경우 문제가 발생되기 이전에 발견을 하기 위해서 지속적이고 꾸준한 NodeVM 지표를 관리할 필요가 나타납니다.

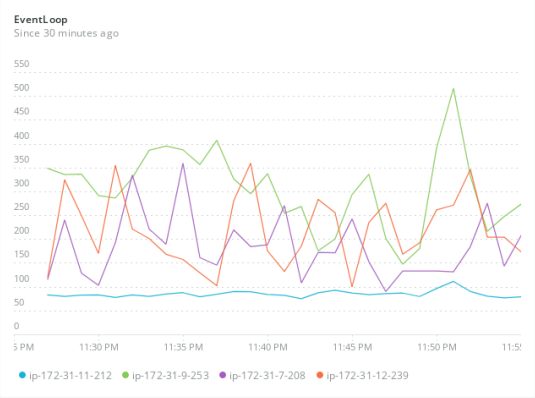
Node VM 활동 통계 Statistics

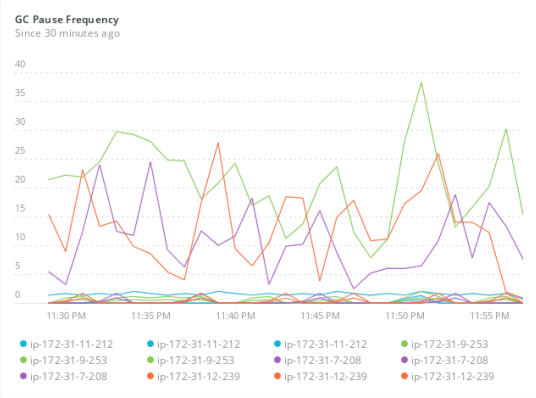
메모리 릭(누수 현상)은 종종 눈에 띄지 않은 채 일어납니다. 더구나, 생산 성과 지표에 각별히 주의를 기울일 때 문제로 드러나게 되는 경우가 있습니다. 서비스 애플리케이션에서 메모리 누출의 첫 번째 증상은 뚜렷한 이유없이 시간이 지남에 따라 호스트 머신의 메모리, CPU 사용량 및 부하 평균이 증가한다는 것입니다. CPU 사용량이 100%에 도달할 때까지 응답 시간이 점점 더 높아지고 응용 프로그램이 응답을 중지합니다. 메모리가 가득 차 있고 스왑이 충분하지 않을 때 서버는 SSH 연결 조차 받아들이지 않을 수도 있습니다. 뉴렐릭으로 시스템 지표를 내보내는 것을 실패하여 Host Not Responding 현상이 나타나기도 합니다. 하지만 응용 프로그램이 다시 시작되면 모든 문제가 마법처럼 사라진다! 그리고 아무도 무슨 일이 일어났는지 이해하지 못하기 때문에 다른 우선순위로 넘어가지만, 문제는 주기적으로 반복됩니다.

이러한 메모리 누수 현상과 애플리케이션의 의한 CPU 의 비정상적 사용에 따르는 문제를 사전에 방지하기 위하여 뉴렐릭은 다음과 같은 Node VM 의 활동 지표를 제공합니다.

```
GC pause time
GC pause frequency
GC pause time by type
Memory usage
CPU utilization
Event loop — ticks per minute
Event loop — max CPU time per tick
```

위 지표로 다음과 같은 현상을 확인해 봅니다. 응용 프로그램이 더 많은 개체를 사용하기 시작하면 메모리 사용량이 증가하고 GC(가비지 수집)가 더 자주 실행됩니다. GC 에 소요된 시간 때문에 CPU 사용률이 증가할 것입니다. 동기화 코드가 비정상적으로 오래 실행되면 CPU 사용률이 증가할 수 있습니다. 눈금 차트당 **이벤트 루프 최대 CPU 시간에 스파이크가 표시**됩니다.





Node VM 클러스터 관찰

복수개의 Node.js 프로세스들이 동일 서버상에서 같은 뉴렐릭 애플리케이션에 리포트를 하고 있을 때, 아래와 같은 차트를 이용하여 통합된 데이터를 관찰할 수 있습니다. 클러스터 상의 작업 프로세스가 각각 자신만의 Node.js runtime(실행시간)을 부여 받고 분리된 데이터를 소유하기 때문입니다.

클러스터	설명
모든 작업 프로세스들의 합을 보여주는 차트	• GC pause time - Total time per minute • GC pause frequency • GC pause time by type • CPU utilization • Event loop ticks per minute
모든 작업 프로세스들 값의 평균을 보여주는 차트	• Memory usage • GC pause time – average
모든 작업 프로세스들 값의 최대값을 보여주는 차트	• GC pause time – max • Event loop — max CPU time per tick

Node VM 관찰을 위한 측정

뉴렐릭 Node.js 에이전트는 Node.js VM(V8)의 주요 지표에 대하여 시간 단위 데이터를 수집합니다. 이러한 지표를 통하여 Node.js VM 의 행위의 이해와 통찰력을 확보하여 애플리케이션의 성능을 안정적으로 유지 혹은 개선하는 것에 도움을 줄 수 있습니다. 또한, 애플리케이션의 주요 워크로드 변화 시, Node.js VM 의 변화를 기록 혹은 마킹 해 둠으로써,

비정상적인 서비스 발생시 빠르고 정확한 판단을 얻을 수 있습니다. 에이전트는 GC 지표를 분석해야 할 때 참고용으로 CPU 지표를 함께 제공합니다.

준비해야 할 것

뉴렐릭은 Node.js VM 지표 데이터를 수집하기 위하여 뉴렐릭 '네이티브 모듈'인 부가적인 모듈을 필요로 합니다. (Native Metrics) 이 모듈은 Node.js 의 네이티브 계층에서 가로채기를 하여 뉴렐릭 Node.js 에이전트에게 필요한 지표들을 제공합니다.

뉴렐릭 Node.js agent v2.0.0 이후에는 선택적 모듈로서 네이티브 모듈을 자동 설치하려고 시도합니다. 자동 설치가 되지 않는 경우 다음과 같은 방법을 통하여 설치를 할 수 있습니다.

```
npm install @newrelic/native-metrics
```

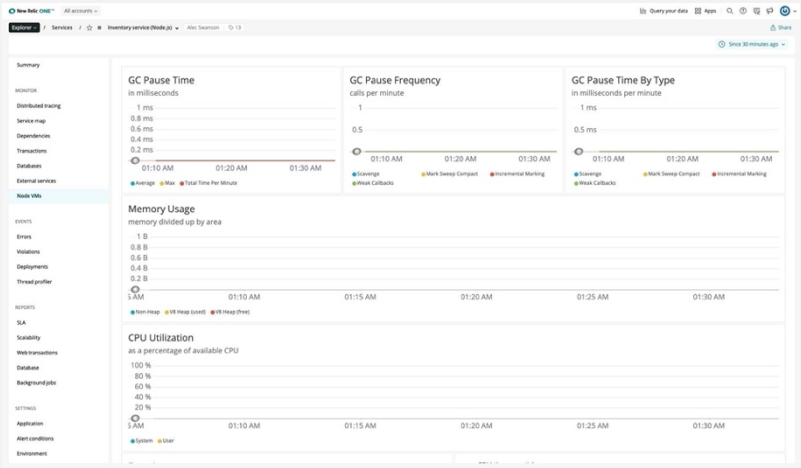
혹은

```
yarn add @newrelic/native-metrics
```

네이티브 모듈의 추가 설치가 필요하지 않다면 설치전 다음과 같은 환경 변수 설정을 통하여 설치를 사전에 방지할 수 있습니다.

```
export NR_NATIVE_METRICS_NO_DOWNLOAD=true
```

리눅스가 아닌 윈도우즈 환경에서는 추가 매뉴얼을 확인하시기 바랍니다.



Node VM 지표 데이터의 이해

(모든 노드 버전에서 데이터를 수집하지 못할 수 있습니다. Node 의 LTS 버전의 사용을 추천합니다.)

다음과 같은 지표 판단을 기준으로 이후 제공되는 지표의 내용을 참고합니다.

지표 판단 기준

GC Cycle Duration

Scavenge 또는 Mark-sweep 주기가 길어지면 GC 주기마다 노드가 싱글스레딩되기 때문에 애플리케이션 요청의 대기 시간이 길어집니다.

메모리 릴리즈

정상 애플리케이션에서 모든 GC 사이클은 연결이 끊어진 개체를 할당 릴리스해야 하며 할당된 총 힙의 크기가 유휴 애플리케이션 수준으로 되돌아와야 합니다. 응용 프로그램 요청 수가 증가함에 따라 할당된 메모리가 증가함을 알 수 있습니다. 그리고 요청 횟수가 줄어든 후에는 릴리스되지 않는 다면 이는 애플리케이션의 어딘가에서 메모리 누수가 발생하여 결국 애플리케이션의 장애로 연결될 수 있다는 것을 나타냅니다.

Time in garbage collection

적용 환경

- Node.js agent v1.35.1 or higher
- Node v4 or higher
- New Relic Node.js `@newrelic/native-metrics` v1.0.0 or higher

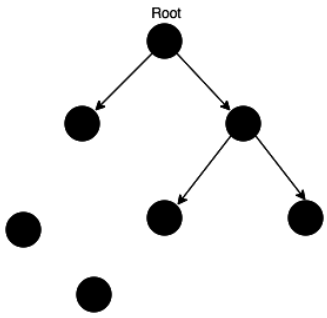
Node 프로세스가 가비지 컬렉션에 소모한 시간의 양을 보여줍니다. 시간은 누적된 값(Pause)과 타입에 따라 분리된 값(<type>)으로 각각 나누어 모두 수집합니다. 타입에 따라 다음과 같은 지표를 수집합니다.

GC type	설명
`Scavenge`	가장 일반적인 GC 방법. Node 는 VM 이 한가해질 때 항상 기동되는 방법이다. 종종 일어나며 빠르다. 힙상에 생성되는 상대적으로 작은 오브젝트들을 클리닝한다.
`MarkSweepCompact`	부담을 주는 GC 타입이다. 이 타입의 이벤트가 발생하면 오브젝트의 개수를 줄이거나 힙의 리밋을 늘려야 한다. 정리할 수 있는 개체를 탐지하는 데 시간이 더 오래 걸리기 때문에 덜 자주 발생하는 GC 이벤트이다. 첫째, 마크 스위프 알고리듬은 스캐빈지 사이클에서 살아남은 모든 물체를 스캔한다. 힙의 루트에서 시작하여 그래프상에 각 개체를 표시하고, 표시가 완료되면 루트 개체에서 연결할 수 없는 개체를 제거한다.
`IncrementalMarking`	애플리케이션의 정지 시간의 양을 줄이기 위하여 애플리케이션 로직과 결합하여 단계별로 발생하는 GC 입니다. (V6+ only)
`ProcessWeakCallbacks`	GC 가 발생한 이후 V8 Node VM 은 이미 할당 해제된 오브젝트에 등록된 레퍼런스 콜백을 회수합니다. (Only in Node v6 or higher.)

Memory

적용 환경:

- Node.js agent v1.36.0 or higher
- Node v4 or higher



뉴렐릭 에이전트는 Node VM 에서 다음과 같은 메모리 지표를 수집합니다.

지표	설명
`Memory/Physical`	Node 프로세스에 의해서 사용되는 총 물리적인 메모리. `process.memoryUsage().rss` Node AP 를 통하여 기록된다
`Memory/Heap/Max`	자바스트립트 오브젝트를 위하여 V8 에 의해서 할당된 총 힙(MB)의 양. `process.memoryUsage().heapTotal` Node API 를 통하여 기록된다.
`Memory/Heap/Used`	앱에 의해서 현재 사용되고 있는 V8 힙의 값. `process.memoryUsage().heapUsed` Node API 를 이용하여 기록된다.
`Memory/Heap/Free`	V8 메모리로 할당이 되었으나 사용되고 있지 않는 메모리 양 다음과 같은 연산으로 나타난다 (`heapTotal - heapUsed`).

`Memory/NonHeap/Used`	V8 힙이외의 용도로 사용되는 메모리 양. 다음의 계산으로 나타난다 (`rss` - `heapTotal`). 이 지표는 V8 힙이외의 영역에서 나타나는 메모리 누수를 찾는 데 유용하다. 가령, Node VM 시스템의 스트림이나 버퍼를 사용하는 경우에 관리 지표이다.

CPU

적용 환경

- Node >= v6.1.0, Agent v1.34.0 or higher
- Node v4 - v6.0.0, Agent v1.35.2 with `@newrelic/native-metrics` v1.0.0 or higher

CPU 지표는 Nove v6.1.0 이후 버전에서 수집된다. `process.cpuUsage()` Node API 를 이용하여 기록되며, 노드의 옛 버전의 경우 '네이티브 모듈' 설치를 반드시 확인해야 한다. 에이전트는 다음과 같은 CPU 관련 지표를 수집한다.

지표	설명
`CPU/User Time`	사용자 코드를 실행하면서 사용된 CPU 시간. (초)
`CPU/User/Utilization`	사용자 코드를 실행하기 위해 사용된 시간을 논리적인 프로세서의 개수로 나누어진 값. Node VM 은 항상 하나의 코어만을 이용하여 실행하기 때문에, 이 지표에 의해서 보고되는 최대 값(싱글 코어

	최대 소진 값)은 총 시스템 100 을 코어수로 나눈 값이다.
`CPU/System Time`	Node 프로세스와 관련되어 시스템 커널 실행을 위하여 CPU 를 사용한 시간.
`CPU/System/Utilization`	시스템 커널을 실행하기 위하여 소비된 시간을 논리적인 프로세서의 개수로 나눈 값. Node VM 은 항상 하나의 코어만을 이용하여 실행하기 때문에, 이 지표에 의해서 보고되는 최대 값(싱글 코어 최대 소진 값)은 총 시스템 100 을 코어수로 나눈 값이 된다.

Event Loop

적용 환경:

- Node.js agent v1.37.0 or higher
- `@newrelic/native-metrics` v2.1.0 or higher
- Node v4 or higher

Node 이벤트 루프와 관련된 성능 지표를 수집합니다. 이벤트 루프 지표는 '네이티브 모듈'을 반드시 필요로 합니다. 에이전트는 다음과 같은 지표를 수집합니다.

지표	설명
`Nodejs/EventLoop/CPU/Usage`	매번 이벤트 루프 틱에서 실행되는 총 CPU 시간. 이 지표는 애플리케이션의 콜백을 실행하는 것을 포함한다. 그러나, Node VM 자신을 위한 콜백도 포함한다. (단위는 초) 비정상적으로 긴 이벤트 루프 틱은 동기화된 실행에 문제가 발생했음을 의미한다. 애플리케이션 최적화가 필요할 수 있다. (예, `process.nextTick` 리커시브 콜).