GiantVM 설치 가이드

- 다음의 내용은 2대의 머신(*node0*, *node1*)을 GiantVM으로 묶어서 가상 1머신으로 구동하기 위한 절차입니다.

- 권장 환경은 2대의 real 머신이 infiniband 네트워크로 연결된 구성입니다.

> **(주의 사항)**
> VirtualBox로 만든 *Linux 가상머신*에서 QEMU/KVM기반으로
> 또 다른 *가상머신*을 기동하는 것은 현재 VirtualBox가 지원하지 않습니다.
> 따라서, 아래의 절차는 real 머신에만 적용됩니다.

1. 준비물

1) H/W

- 인피니밴드(infiniband) 네트워크로 연결된 2대의 real 머신 (node0, node1)

> **(참고)**
> GiantVM은 H/W dependency가 있습니다
> - Infiniband 네트워크로 연결된 구성에서 정상 동작합니다.
> - 이더넷 환경은 아직까지는 충분히 지원되지 않습니다(컴파일 및 기본구동)

2) S/W

- GiantVM 패치 소스 2종 (참조: https://github.com/ememos/GiantVM)
  : Linux-DSM-4.18 (GiantVM KVM patch for kernel 4.18)
  : QEMU-gvm-vcpupin (GiantVM QEMU patch for vcpu pinning)

> **(참고)**
> - 다수의 머신이 연합하여 1개의 GuestOS가 동작하는 것과 같은 효과를 보이게끔, QEMU와 KVM 소스를 수정해 놓은 것이 GiantVM S/W입니다.
>
> - 원래, QEMU는 GuestOS의 코드를 실행할 때 KVM의 도움을 받아 하드웨어 가속 기능을 활용할 수 있습니다. GiantVM S/W에는 QEMU 쪽에 vCPU, I/O 기능이 구현되어 있고, KVM에 분산 공유 메모리(Distributed Shared Memory, DSM) 기능이 구현되어 있습니다.

- 환경 설정 파일
  : config-4.18.20-gvm (Linux-DSM-4.18 컴파일용)

- Guest OS
  : GiantVM 상에서 기동할 guest OS 이미지

- Guest OS 기동용 스크립트(run_gvm.sh)

(GiantVM의 기동 모습 미리 보기)

- 2절 이후의 절차대로 2개의 real 머신(node0, node1)에 GiantVM S/W를 설치하고 나서, 동일한 GuestOS 이미지를 2개 머신에 각각 copy합니다.

- 2개 머신(node0, node1)에서 기동 스크립트(run_gvm.sh)를 실행하는데, vCPU 시작 번호 옵션을 달리하여 각각 실행합니다.

- 기동 스크립트의 내용은 GuestOS 이미지를 QEMU가 실행하는 것입니다. 실행 결과, 2개 머신에서 각 QEMU들이 각 GuestOS 이미지를 이용하여, GuestOS가 글로벌하게 1벌 동작하는 것과 같은 효과를 보이게끔 QEMU/KVM 코드들이 머신 간 협력하여 동작합니다.

- 사용자 관점에서는 node0 머신에서 실행한 기동 스크립트(run_gvm.sh)의 동작 과정에서 글로벌한 가상 머신 1대의 로그인 프람프트가 출력됩니다. 한편, node1 머신에는 메시지 로그형태의 상태값만 계속 출력되며 로그인 프람프트는 없습니다.

- node0 머신에 보이는 로그인 프람프트를 통해 로그인하면, 일반적인 리눅스 사용하듯이 커맨드 라인에서 명령을 입력할 수 있습니다. 여기서부터, GiantVM을 통해 실행되는 글로벌한 1대의 가상 머신을 사용하는 것입니다. memcpy 등의 벤치마크 S/W를 실행해 보시기 바랍니다.

2. GiantVM 패치 소스 2종 준비

$ git clone https://github.com/ememos/GiantVM.git

(결과로서, GiantVM 디렉토리 아래에
Linux-DSM-4.18, **QEMU-gvm-vcpupin** 디렉토리와
config-4.18.20-gvm 파일 등 생성됨)

3. 호스트용 커널 설치 (Linux-DSM-4.18)

2개의 머신(node0, node1)에 3-1~3-4의 절차를 각각 적용합니다.

3-1. 커널 컴파일 준비

```
$ sudo apt-get install -y build-essential libncurses5-dev gcc libssl-dev grub2 bc
$ sudo apt install flex
$ sudo apt install bison
```

```
  $ sudo apt install net-tools
  $ sudo apt install libelf-dev
```

3-2. 호스트용 커널 컴파일 (Linux-DSM-4.18)

```
 $ cd GiantVM
 $ cd Linux-DSM-4.18
 $ vi Makefile   (확인 또는 명칭 수정)

    --> EXTRAVERSION = -gvm

 $ cp ../config-4.18.20-gvm .config

 $ sudo make oldconfig
 $ sudo make deb-pkg -j 10 LOCALVERSION=1

 $ cd ..
 $ sudo dpkg -i linux-image-4.18.20-gvm1_4.18.20-gvm1-1_amd64.deb
 $ sudo dpkg -i linux-libc-dev_4.18.20-gvm1-1_amd64.deb
 $ sudo dpkg -i linux-headers-4.18.20-gvm1_4.18.20-gvm1-1_amd64.deb
 $ sudo dpkg -i linux-image-4.18.20-gvm1-dbg_4.18.20-gvm1-1_amd64.deb
```

3-3. grub 파일(/etc/default/grub) 수정

3-4. 호스트 재부팅

4. 호스트용 QEMU 설치 (QEMU-gvm-vcpupin)

2개의 머신(node0, node1)에 4-1~4-3의 절차를 각각 적용합니다.

4-1. kvm.h 파일 수정

```
$ cd QEMU-gvm-vcpupin
$ vi linux-headers/linux/kvm.h

   (Line 873) #define KVM_CAP_X86_DSM        133 -> 156 (수정)
```

4-2. configure 실행

```
$ sudo ./configure —target-list=x86_64-softmmu —enable-kvm —disable-werror
```

4-3. make 실행

```
$ sudo make -j 10
```

여기까지 진행되면 GiantVM을 실행할 수 있는 준비가 된 상태가 됨

5. GuestOS 기동

- guest OS를 GiantVM에서 기동
  : 아래의 예제는 vcpu를 total 8개 구성하고, node0 머신에서는 0-3번 vcpu를 기동, node1 머신에서는 4-7번 vcpu를 기동하는 절차입니다.

5-1. guest OS 준비 (ubuntu1804.img 파일)

- Tutorial/guest_image 의 BUILD_guestos-rev.pdf 문서 참고
  (https://github.com/ememos/Tutorial/tree/master/guest_image)

2개의 머신(node0, node1)에 동일한 guest OS 이미지를 최초 copy해 놓습니다.

5-2. node0 머신에서 기동 스크립트 실행
```
# ./run_gvm.sh 0
```

5-3. node1 머신에서 기동 스크립트 실행
```
# ./run_gvm.sh 4
```

5-2~5-3의 절차를 실행하면, node0 머신에서만 guest OS의 로그인 프람프트가 출력되며, 로그인하고 나서 일반적인 리눅스 사용하듯이 명령어 및 벤치마크 S/W 등을 실행할 수 있습니다.

(run_gvm.sh 스크립트 내용)

```
#!/bin/bash
START_VCPU=$1
if [ "$START_VCPU" == "" ]; then
        echo "usage: $0 START_VCPU"
        exit
fi

VM_IMAGE=/home/GiantVM/ubuntu1804.img        # GuestOS 이미지 경로 조정

sudo setfacl -m "u:etri:rw" /dev/kvm                # 권한 제어

 # 설치 환경에 맞게끔 QEMU 실행 파일 위치 지정,
 # -smp 옵션은 글로벌하게 보이는 vCPU 개수, -m은 메모리 크기
COMMAND="./QEMU-gvm-vcpupin/x86_64-softmmu/qemu-system-x86_64 "
```

```
COMMAND+="--nographic --enable-kvm "
COMMAND+="-hda ${VM_IMAGE} "
COMMAND+="-cpu host,-kvm-asyncpf -machine kernel-irqchip=off "
COMMAND+="-smp 8 -m $((8*1024)) -serial mon:stdio "
COMMAND+="-monitor telnet:127.0.0.1:1234,server,nowait "

# for NUMA configuration  # 일단, 무시
#COMMAND+="-object memory-backend-ram,size=4G,id=ram0 "
#COMMAND+="-numa node,nodeid=0,cpus=0-3,memdev=ram0 "
#COMMAND+="-object memory-backend-ram,size=4G,id=ram1 "
#COMMAND+="-numa node,nodeid=1,cpus=4-7,memdev=ram1 "

if [ "$START_VCPU" == "0" ]; then
        COMMAND+="-redir tcp:5556::22 "
fi

# GiantVM 체제로 구동할 2대 머신의 주소는 iplist 옵션으로 지정
#   이 예제에서는 node0는 10.10.20.14, node1은 10.10.20.16의 IP 주소 가정함.
#
#  -local-cpu 옵션으로 각 머신에서 locally 사용하는 vcpu 개수 지정.
#
#  start 옵션으로 각 머신에서 locally 사용할 vcpu 집합 중 첫번째 vcpu 인덱스 지정.
#  만일, node0 머신(IP주소: 10.10.20.14)에서
#      -smp 8, -local-cpu 4,start=0,iplist="10.10.20.14 10.10.20.16" 옵션의 스크립트가
#      실행되면 node0에서는 vcpu 중 0~3번까지를 locally 활용하고, vcpu 4~7까지는
#       remote 머신인 node1(IP: 10.10.20.16)의 자원을 활용한다는 의미.
#  한편, node1에서
#      -smp 8, -local-cpu 4,start=4,iplist="10.10.20.14 10.10.20.16" 옵션의 스크립트가
#      실행되면 node1에서 총 8개의 vcpu 중 4~7까지를 locally 활용하고, vcpu 0~3은
#      node1입장에서 remote 머신인 node0의 자원을 활용한다는 의미.

sudo $COMMAND \
        -local-cpu 4,start=$START_VCPU,iplist="10.10.20.14 10.10.20.16"
```