Parallel Computing with Set of Devices

1 차시

담당교수 :
유승수(kelvin@konkuk.ac.kr)

작성자 :
강현우(kangdroid@konkuk.ac.kr)

1 프로젝트 소개

기본 소개

- 여러개의 Embedded Device를 연결해 클러스터를 구축하고, AWS/Azure 처럼 사용자 요청에 따라 자원을 배분해 주는 프로젝트.
- 기본적으로 Minimum Resource Limit은 있지만, Maximum Resource Limit은 없을 예정.

2 프로젝트 소개

기본 소개

- 현재, Jetson Nano 2대로 진행 중이며, 이전에는 테스트 기기로 RPi이용.
- 현 프로젝트의 목표는 CPU의 완벽한 분할 Policy설정 하는 것
- 가능 하다면, GPU의 완벽한 분할도 구현

3 프로젝트 소개

Policy 1: User-Request Resource Distribution

- 사용자가 서버[메인 컴퓨터]에게 메모리 NGb, CPU[Logical] 몇 코어, 저장 공간 TGb 를 요청 할 때,
- 서버는 사용자의 Request의 적절 여부를 결정
  - 적절 여부 = Server 및 Master이 최소한 돌아갈 수 있는 최소 Resource는 남겨 두어 야 함.
- 적절한 경우, debootstrap으로 새로운 OS가 설치된 가상 환경[With Resource Limit] 할당[username + password]

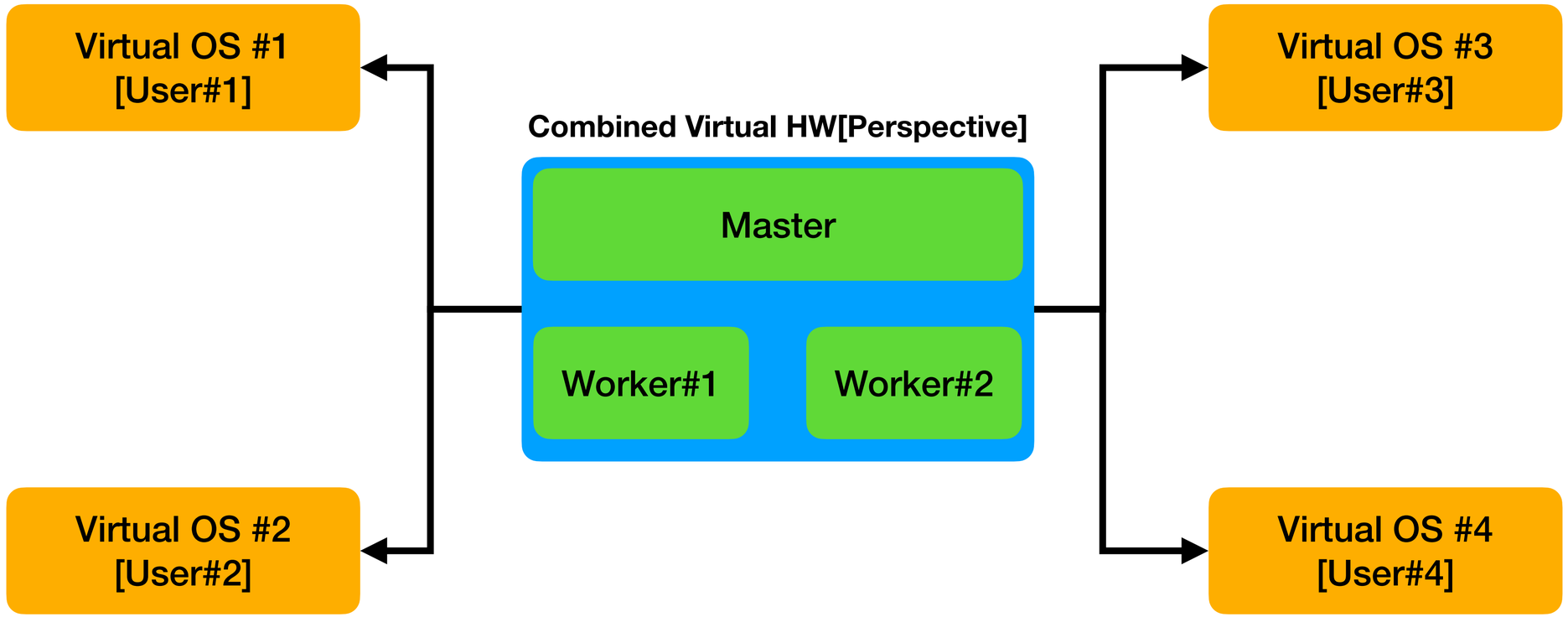
4 프로젝트 소개

Policy 2: Device-Resource-Share Policy

- 서버는 사용자에게 “제한이 없는 리소스”=Maximum Resource를 할당함
  - 가상 OS 및 환경은 그대로 제공
- 각 Master/SubNode 들은 Special Task Scheduler 혹은, Special SysCall 이 필요
  - Master-SubNode들의 자원을 단 “하나”의 자원으로 통일 하고
  - Load Balancing
  - struct_task의 큐, Niceness 등을 확인해야 됨.

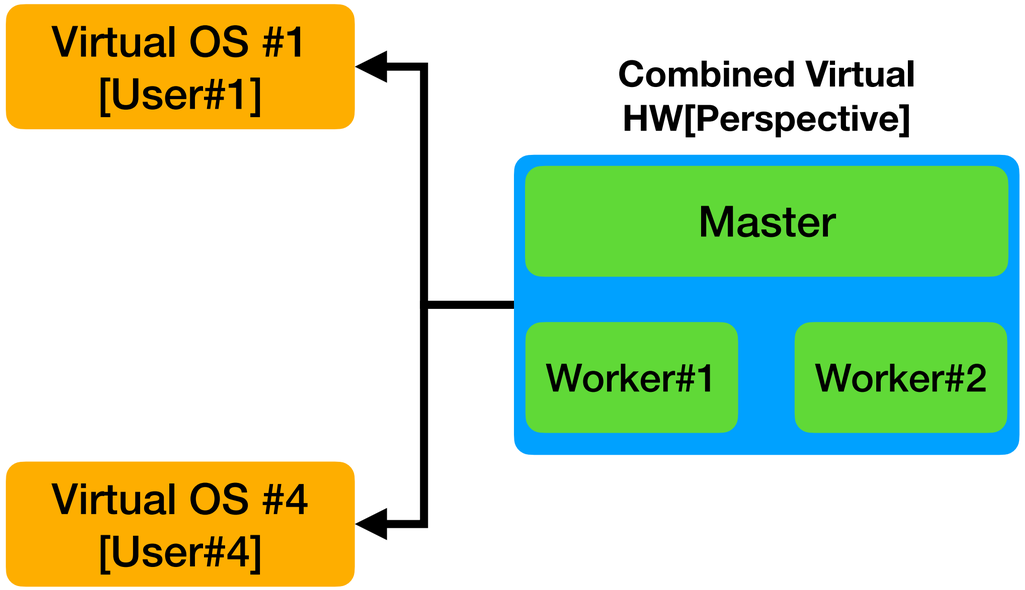
5 프로젝트 소개

Policy 2: Device-Resource-Share Policy



6 프로젝트 소개

Policy 2: Device-Resource-Share Policy

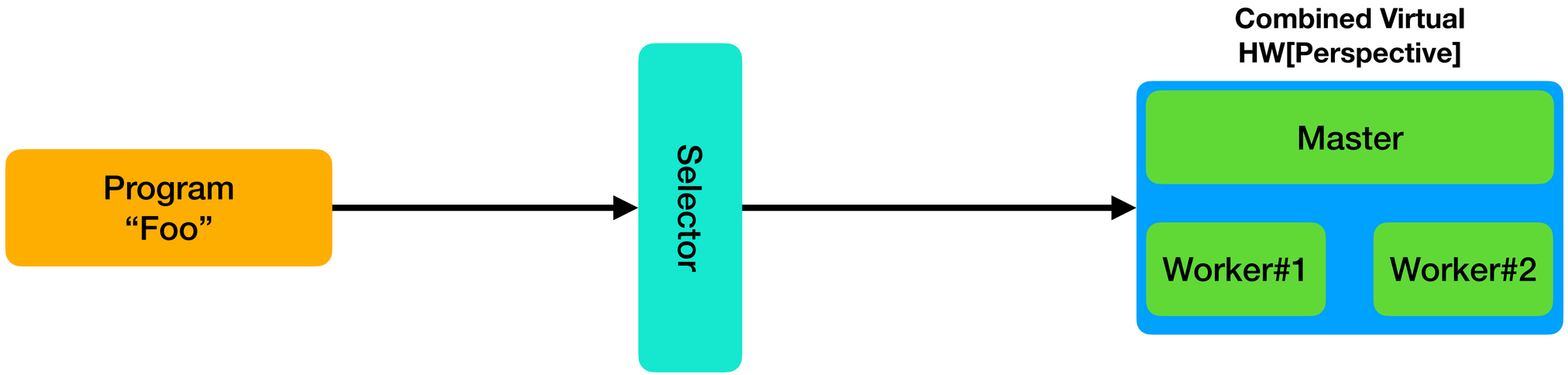


- Total User = 2
- Program Name = “Foo”
- What it does: Sort large number of integers.
  - Let number >= 4Gb.
- Expected: Split integer arrays to half and do execution in parallel[HW]

7 프로젝트 소개

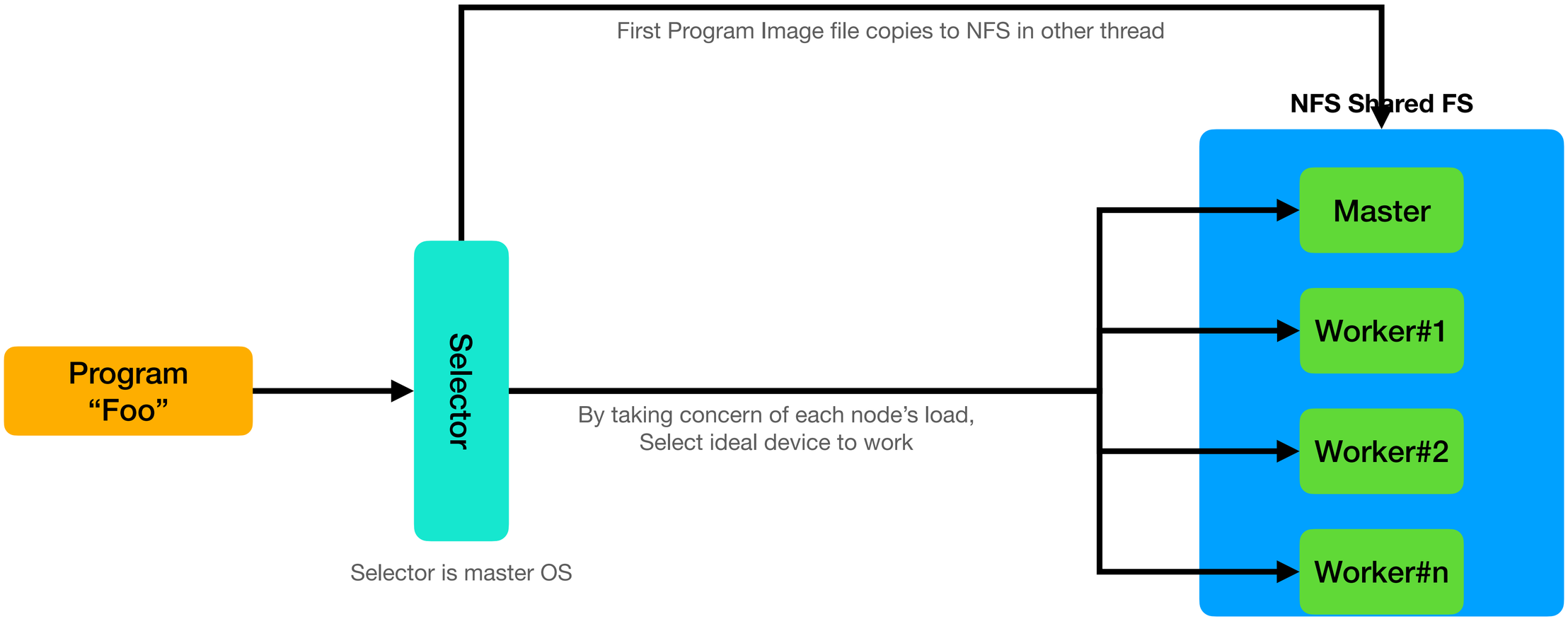
Policy 2: Device-Resource-Share Policy

- What we need
  - Determinant to decide which device will execute program
  - Worker Selector



8 프로젝트 소개

Policy 2: Device-Resource-Share Policy



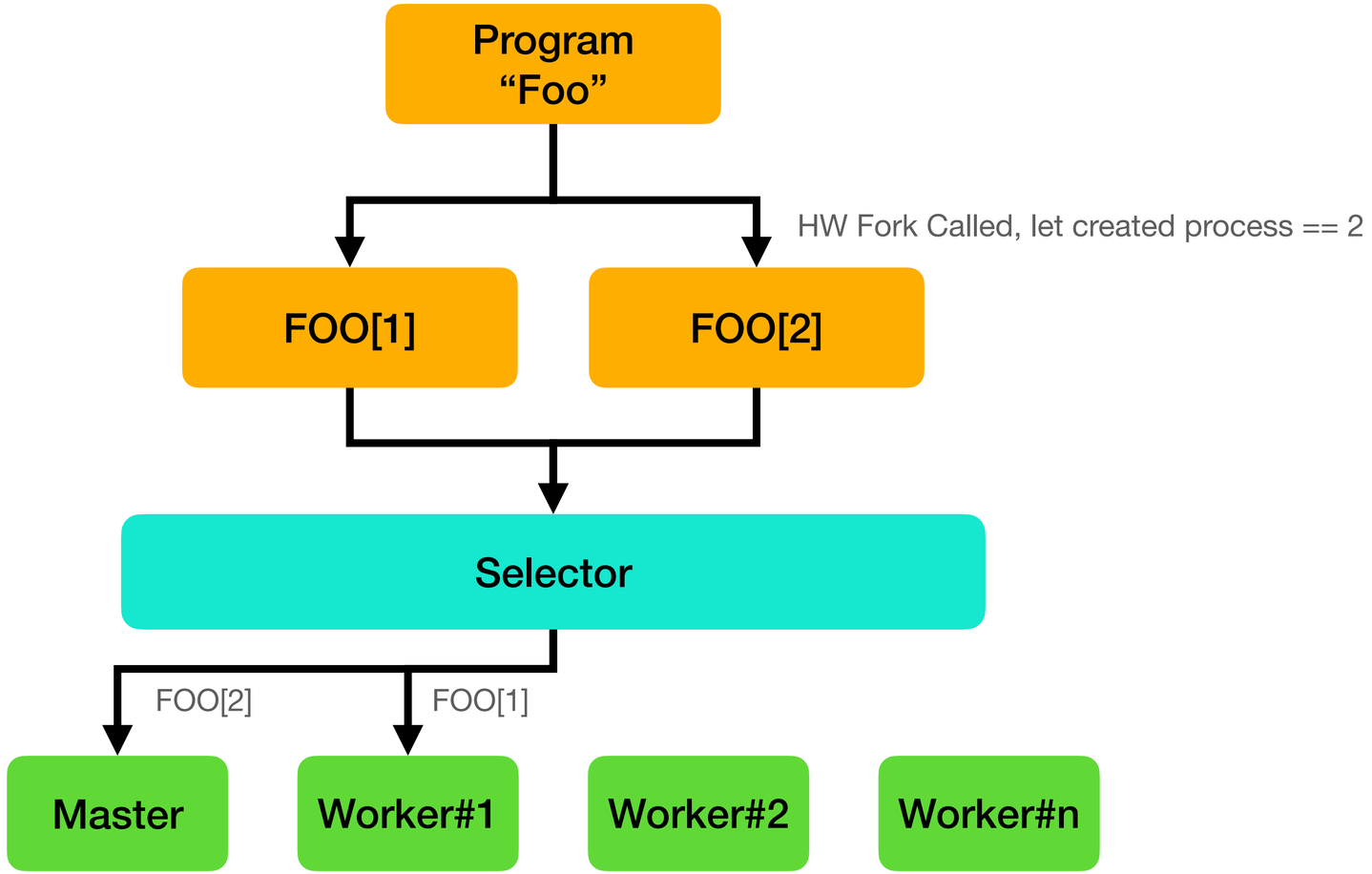
9 프로젝트 소개

Policy 2: Device-Resource-Share Policy

- 프로그램에서 fork와 같이 헤비한 콜을 진행 한 경우, 재귀적인 문제 발생
  - i.e: Swap usage > 90% || task queue > 1020 || load > 70%
- Syscall: HW Fork 고안중
- Fork —> Same Device
- HW Fork —> Not Necessarily

10 프로젝트 소개

Policy 2: Device-Resource-Share Policy



11 프로젝트 소개

Policy 2: Device-Resource-Share Policy

- System Call: HW Wait
- HW Fork로 호출된 자식 프로세스를 부모가 기다리는 것
- TCP 통신 등 방법이 있지만, 일단 NFS의 File로 결과 전송 예정
  - Busy-Wait, Cycle++ —> Temp Solution

12 프로젝트 소개

현재 진행된 것

- User-Space Monitoring Program
  - Temperature[CPU/GPU]
  - CPU Frequency
  - RAM/Disk Usage, Uptime[SysInfo]
  - Load Percentage
- Virtual OS Install with Debootstrap, Partial Resource limit
  - Same distribution with master OS, as well as kernel.

13 프로젝트 소개

진행 중

- Clustering init
  - Setting up NFS, Network, etc.
- User-Space Selector 디자인
- Kernel Modification
  - Load Calculation[Danger, Temp solution]
    —> Load = Total Task Queue length on Kernel Memory area

프로젝트 소개

진행 예정

- Kernel Modification
  - Migrate User-Space Selector to Kernel Space
    - Hooking Selector in SYSCALL(OPEN)
  - HW Fork() && HW Wait()
    - Limit: No recursion on child
  - GPU Fork() && GPU Wait()
    - Could be done with HW Fork
    - Tricky because Nvidia does not make an open source of its cores


347 .widget-area-sidebar
348 font-size: 13px;
349 }
350
351
352 /* =Menu
353
354
355 #access {
356 display:
357
감사합니다