반도체SW활용 중간발표

2025.11.12

Presentation by Wee Dayeon

[givemeaplz1312, 32202841, pjuhee23, wida10]@dankook.ac.kr

32214298 조서연 / 32202841 유석 / 32221902 박주희 / 32222797 위다연

단국대학교 DANKOOK UNIVERSITY

INDEX

1. 서론

- **FTL의 주 기능 3가지**
  1. Address Translation
  2. Wear-Leveling
  3. Garbage Collection

- **Garbage Collection Algorithm**
  - 기존에 배웠던 GC 알고리즘들은 **GC Victim 선택 알고리즘**
    1. Greedy Algorithm
    2. Cost-Benefit Algorithm
    3. CAT Algorithm

  - 그렇다면, **GC Execution Policy는 ?**
    - 아직까지 대표적인 Policy: I/O가 발생하면 Write Stall이 발생하는 방식 → Original GC라 칭함
    - 게임 엔진에서는 Incremental GC의 사용을 권장
    - Incremental GC를 FTL에 적용해보자

1. 서론

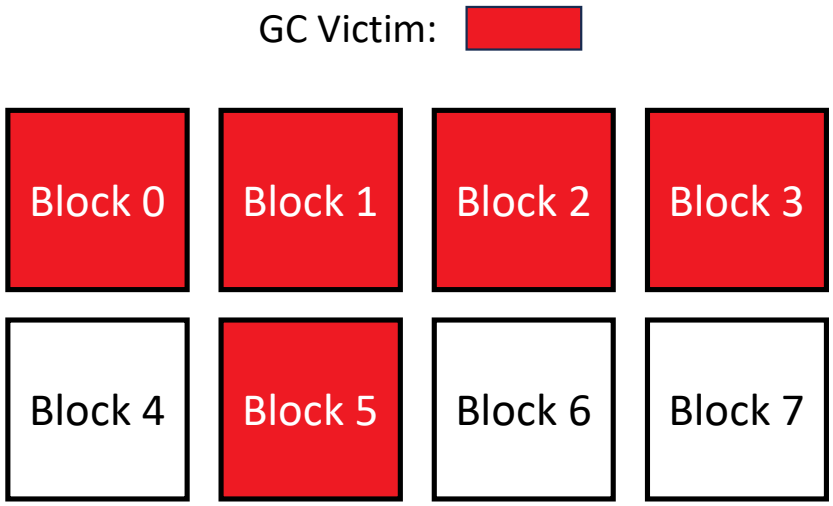
- **FTL의 주 기능 3가지**
  1. Address Translation
  2. Wear-Leveling
  3. Garbage Collection

- **Garbage Collection Algorithm**
  - 기존에 배웠던 GC 알고리즘들은 **GC Victim 선택 알고리즘**
    1. Greedy Algorithm
    2. Cost-Benefit Algorithm
    3. CAT Algorithm

  - 그렇다면, **GC Execution Policy는 ?**
    - 아직까지 대표적인 Policy: I/O가 발생하면 Write Stall이 발생하는 방식 → Original GC라 칭함
    - 게임 엔진에서는 Incremental GC의 사용을 권장
    - Incremental GC를 FTL에 적용해보자

2. GC 동작 과정

- FTL에서 Invalid Page가 많은 블록을 골라,
  Valid Page만 새로운 블록으로 복사한 뒤, 기존 블록을 Erase하여 Free Block으로 회수하는 기법

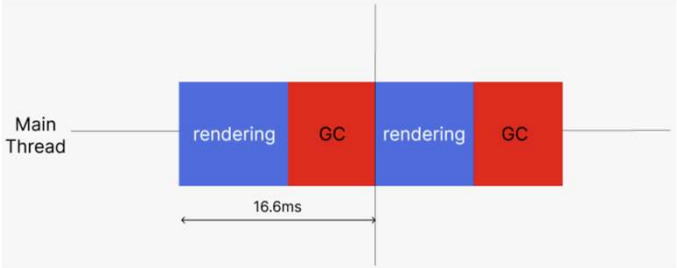


어떤 Block을 GC 대상으로 우선 선택할지 결정하는 알고리즘 3가지
(즉, Victim 블록 선정 기준을 정의하는 알고리즘)

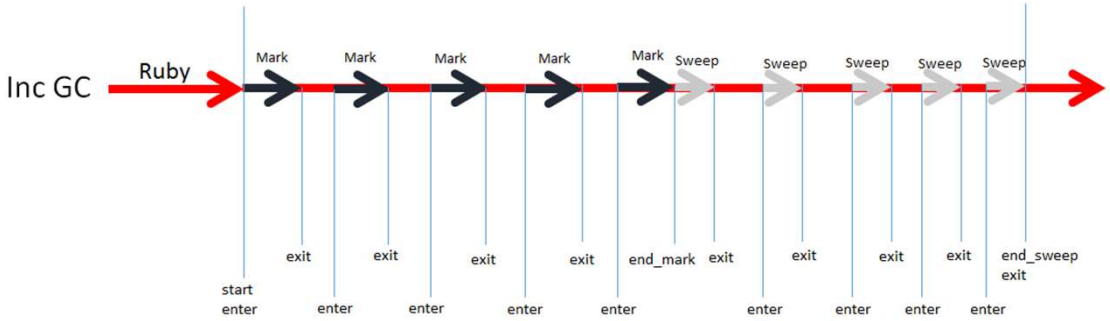
1. Greedy Algorithm
2. Cost-Benefit Algorithm
3. CAT Algorithm

3. GC 알고리즘 종류

- **GC Victim Select Algorithm**
  - Greedy Algorithm: Valid Page가 가장 적은 블록 선택
  - Cost-Benefit Algorithm: 이득 대비 유효 페이지 복사 비용 최소화

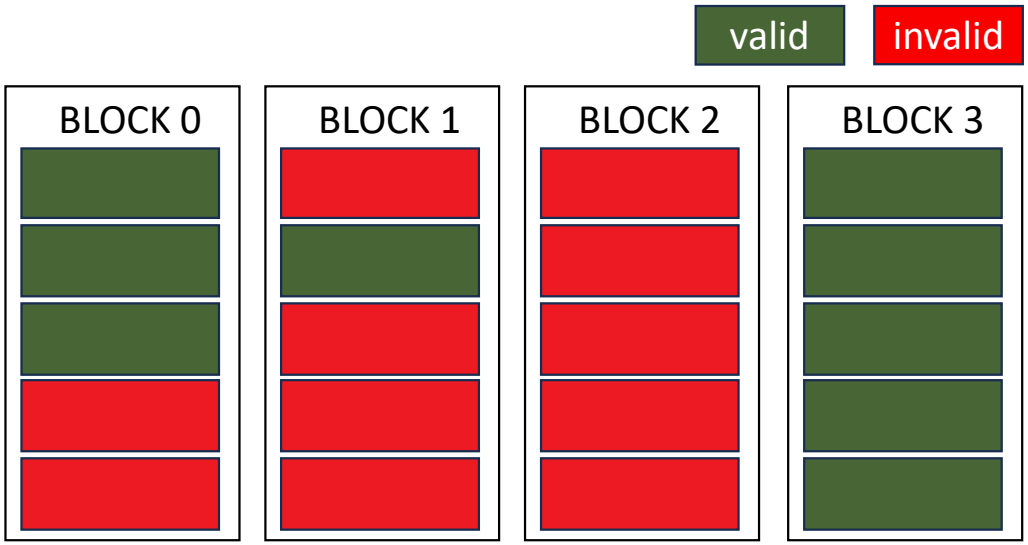


- **GC Execution Algorithm**
  - Incremental GC: 한 번에 전체 작업을 수행하지 않고, 매 프레임 처리마다 GC의 과정을 조금씩 나누어 실행
    - 본래 게임 엔진에서 사용하는 메모리 GC 알고리즘
    - 하나의 GC를 작게 분산하여 주기적으로 수행
    - FTL 환경에 적용하여 I/O 성능 저하 완화



3. GC 알고리즘 종류

- **Greedy Algorithm**
  - Invalid Page 비율(=유효비율 u가 가장 작은 block)이 가장 큰 Block을 Victim으로 선택
    - 즉시 최대 여유 공간을 회수하는데 효과적
    - Wear Leveling에서 불리할 수 있음

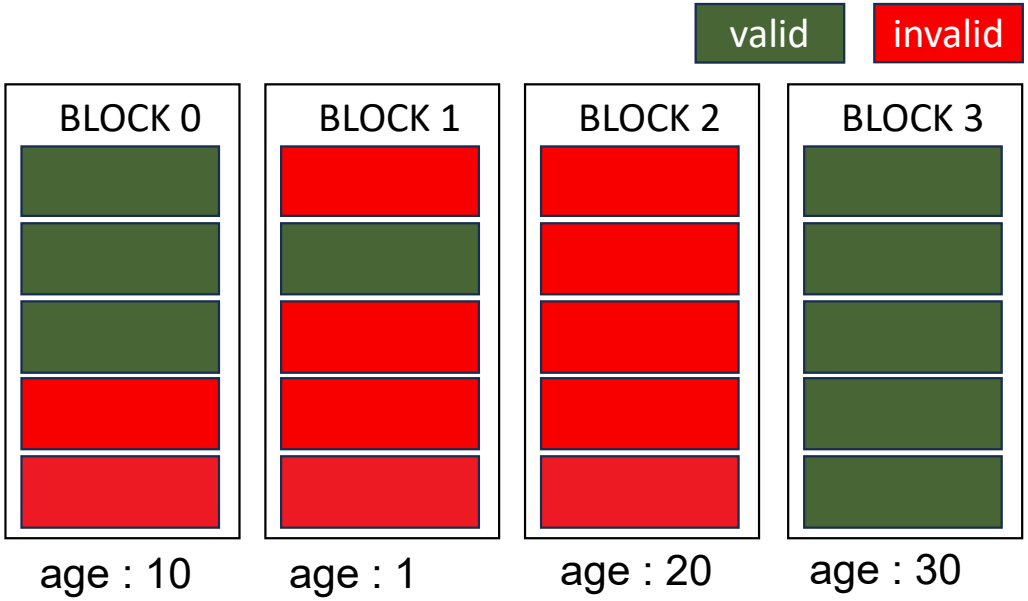


$$u = \frac{\#valid\ pages\ in\ block}{\#total\ pages\ in\ block}$$

- Block 0 : 3/5 = 0.6
- Block 1 : 1/5 = 0.2
- Block 2 : 0/5 = 0.0
- Block 3 : 5/5 = 1.0

3. GC 알고리즘 종류

- **Cost-benefit Algorithm**
  - 아래 식을 계산한 점수가 최대인 Block 선택
    - 회수 이득과 비용, 오래됨을 함께 고려하는 균형 정책
    - Age의 경우, 상한을 두는 경우가 많음



$$cleaning = \frac{age * (1-u)}{2u}$$

- u: 유효 데이터 비율
- $(1-u)$: 회수 가능 공간 비율
- age: 최근 수정 이후 경과 시간
- $2u$: read(u) + write(u)

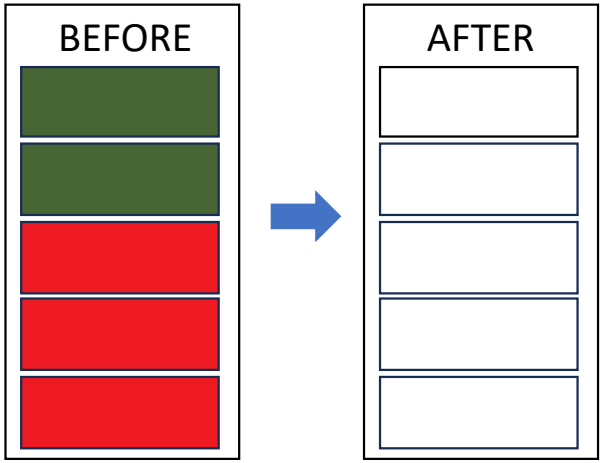
3. GC 알고리즘 종류

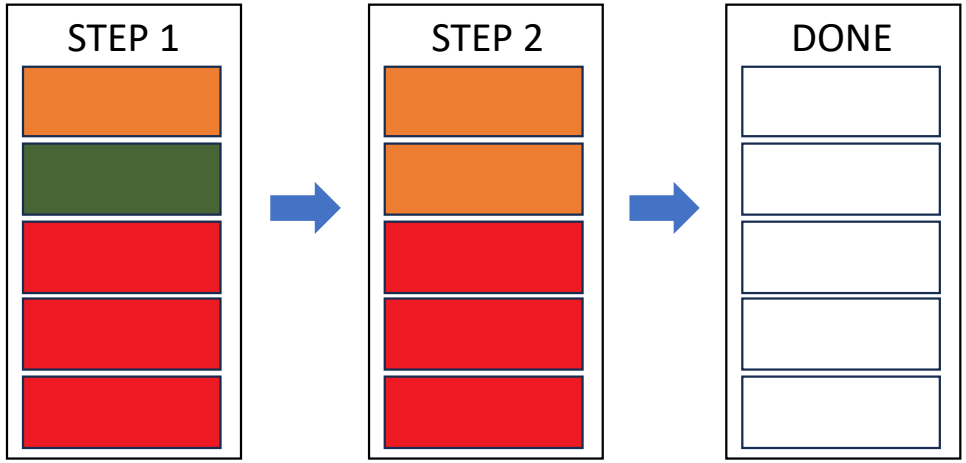
invalid valid processing

- **Incremental Algorithm**
  - GC 작업을 여러 단계로 분할하여 응용 프로그램 중단 시간 최소화

- Original
  - 한 번에 모든 유효 페이지 이동 후 블록 Erase
  - Original: I/O 차단(전체 블록 처리 완료까지)

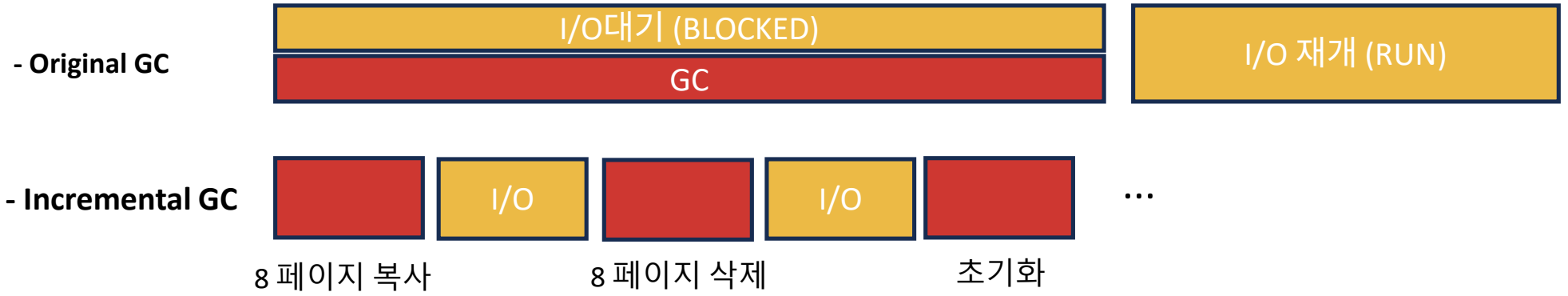
- Incremental
  - 페이지를 나누어 이동 및
  - ON: I/O와 병행 처리(응답 시간 개선)





3. GC 알고리즘 종류

- **Incremental Algorithm**



4. 실험 설계

- **비교 대상**

Greedy Algorithm Original	**Cost-benefit Algorithm** Original
Greedy Algorithm Incremental	**Cost-benefit Algorithm** Incremental

각 알고리즘에 대해 동일하게
Uniform 워크로드를 적용하여
성능 지표 측정 및 비교

- **실험 환경**

CPU	Intel(R) Core(TM) i5-14400
Memory	32GB
SSD	OpenSSD (SSSTC CL4-8D512)
OS	Ubuntu-22.04.5 LTS

5. 성능 지표 및 기대 성능

- **코드 검증**
  - GC가 실제로 발생했는가?: 적절한 위치에 디버깅 메세지 삽입
  - GC Trigger를 위해 Write Size ↑

- **측정할 성능 지표**
  - Greedy vs. Cost-Benefit: WAF, GC Latency
  - Incremental vs. Original: Throughput
  - Global Metric: Memory Spike

Greedy	Cost-Benefit
WAF↑ Latency↓	WAF↓ Latency↑

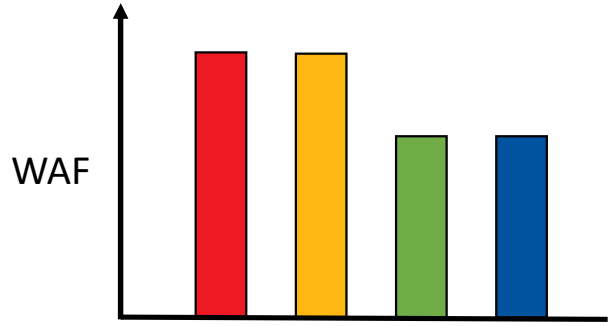
Incremental	Original
Throughput↑ Spike↓↓	Throughput↓ Spike↑↑

5. 성능 지표 및 기대 성능

- **기대 성능**

  1. WAF

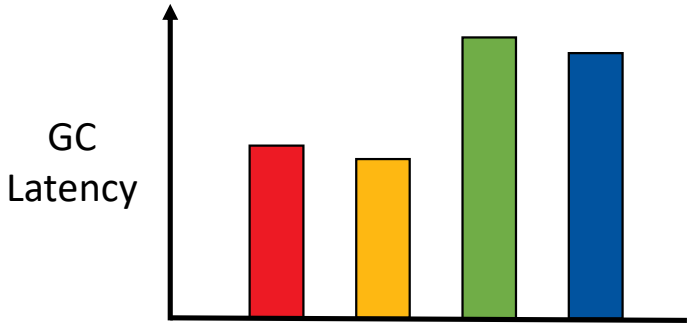
  - Cost-Benefit은 Read+Write 비용까지 고려하기 때문에 Greedy에 비해 WAF ↓
  - Incremental과 Original은 GC 자체는 동일하게 진행을 마치기 때문에 차이 X



  2. GC Latency

  - Greedy에 비해 Cost-Benefit은 계산 오버헤드가 있음
  - Incremental은 Original에 비해 GC를 쪼개서 진행

    → Incremental + Cost-Benefit 의 GC Latency ↑



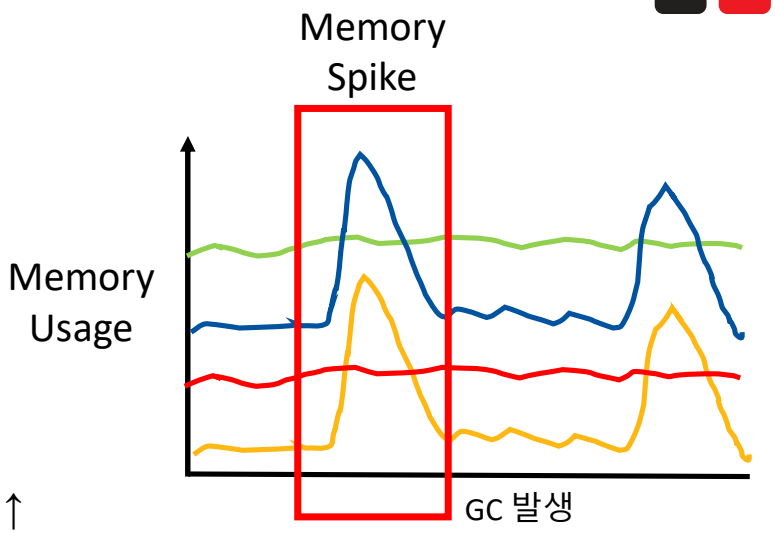
: Incremental + Greedy : Original + Greedy : Incremental + Cost-Benefit : Original + Cost-Benefit

5. 성능 지표 및 기대 성능

- **기대 성능**

3. Memory Usage

- Original의 경우 GC를 수행하는 동안 Memory 사용량 급상승
- 반대로 Incremental의 경우엔 I/O와 GC를 병행하기 때문에 높은 Memory 사용량을 유지하지만, 큰 Spike가 발생하진 않음
- Cost-Benefit은 계산 오버헤드 때문에 Original에 비해 Memory 사용량 ↑

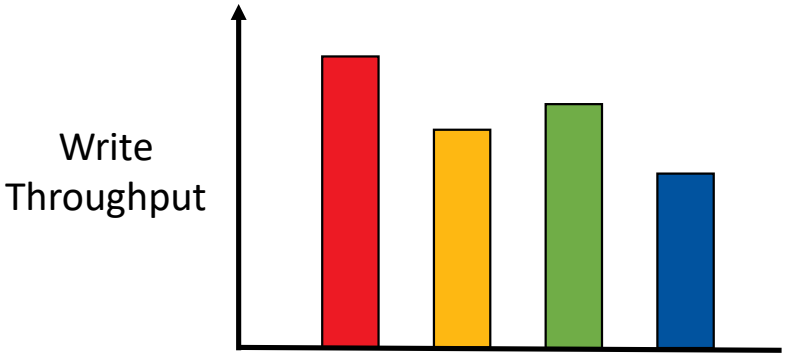


4. Write Throughput

- Incremental은 I/O 작업을 GC와 병행하기 때문에 I/O stall 감소

  → Original에 비해 Write Throughput ↑
- Cost-Benefit은 Read+Write 비용까지 고려하기 때문에 WAF 낮음

  → Greedy에 비해 Write Throughput ↑

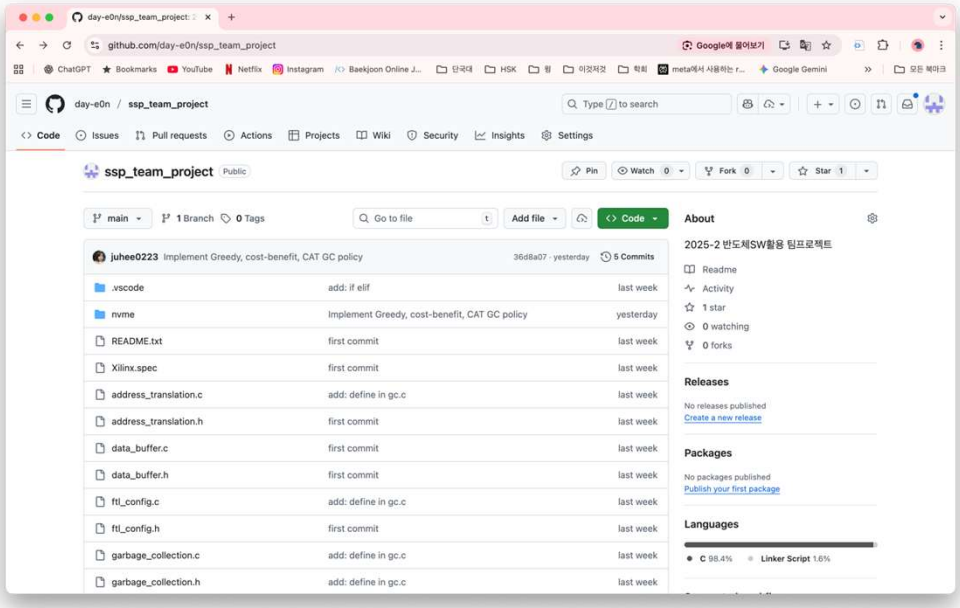


: Incremental + Greedy : Original + Greedy : Incremental + Cost-Benefit : Original + Cost-Benefit

6. 향후 계획

- **향후 계획**
  - 코드 구현 완료 후 검증
  - Memory Spike 측정 방법
  - 실제 결과와 가설이 일치하는지 검증, 일치하지 않다면 원인 분석

  - 깃허브에서 공동 작업 중



반도체SW활용 중간발표

Thank you, Any Question?

2025.11.12

Presentation by Wee Dayeon

[givemeaplz1312, 32202841, pjuhee23, wida10]@dankook.ac.kr

32214298 조서연 / 32202841 유석 / 32221902 박주희 / 32222797 위다연

단국대학교 DANKOOK UNIVERSITY