Spark Bench

- Parameter Tuning -

1. Default

A. Test subjects

benchmark=(“wordcount” “sort”)

benchtest=(0 1) # 0: wordcount 1: sort

B. Parameters

Parameter	Definition	Explanation
parallelism	parallelism value	데이터를 나눌 파티션 개수
driver_mem	driver memory size	Driver 메모리 크기
worker_mem	worker memory size	연산에 사용되는 메모리 크기
work_inst	number of worker	Spark 실행시 생성되는 worker 수
cores	number of cores in worker	Spark 실행시 사용할 스레드 수
caching	RDD persistence	해당 데이터의 메모리/디스크 캐싱여부

C. Baseline

parallelism=(4 4)

driver_mem=(2g 2g)

worker_mem=(2g 2g)

worker_inst=(2 2)

cores=(1 1)

D. Scores

Your score of wordcount is 52.67723893380841441688 / 100

Your score of sort is 50.58034578232847704752 / 100

2. Parameter Testing

A. Failures

우선적으로 여러가지를 시도해보았다.

첫 번째로 알아낸 것은 wordcount 와 sort 를 둘 다 벤치마킹할 경우, 매우 미묘하게 더 낮게 나오는 것 같다는 것이었다. 함께 실행해보았을 때 46~50 가량의 스코어를 받은 반면, 각자 실행했을 때는 46~52 스코어가 나왔다. 사실 sort 를 벤치마킹하는데 시간이 너무 오래 걸렸기 때문에 자주 돌릴 수 없었다. 아마 전혀 상관없는 문제일 수도 있다. 하지만 어차피 시간 문제도 있고 혹시나가 있으니 일단 각자 실행하기로 했다.

두 번째로 알아낸 것은 같은 parameter 로 실행하더라도 벤치마킹 결과는 조금씩 다르게 나온다는 것이었다. 큰 의미는 없지만 스코어 120 이상이 한 번은 나와야 안정적으로 100 전후가 보장될 것으로 보인다.

세 번째로 알아낸 것은 parameter 의 작은 변경은 큰 의미가 없다는 것이었다. 우선적으로, wordcount 를:

parallelism=8

driver_mem=4g

worker_mem=4g

worker_inst=4

cores=2

로 변경했다. 모든 parameter 를 두 배로 늘렸다. 스코어가 53 이 나왔다. 작은 변경은 큰 의미가 없는 듯 했다.

다음으로 모든 parameter 를 10 배로 늘렸다:

parallelism=40

driver_mem=20g

worker_mem=20g

worker_inst=20

cores=10

스코어는 27 이 되었다. 크게 키웠다는 생각은 없었지만 모든 것을 다 키우는 것은 역시 손해보는 일이었다. 무엇보다도 16GB 메모리인데 driver 와 worker 로 각각 20g 를 제공한 것은 말도 안되는 일이었다고 생각한다.

이후 parameter 를 하나씩 수정하기로 했다. 몇 차례 각 parameter 값을 변경하며 수행을 반복했지만 스코어가 60 초반을 넘지 못했다.

B. WASP: Wordcount

parallelism=512

driver_mem=2g

worker_mem=10g

worker_inst=1

cores=4

궁여지책으로 이전 수업에서 보았던 WASP 세팅을 빌려 써보기로 했다. 놀랍게도 43 이라는 최저 스코어를 받았다. parallelism 은 128 배로 크게 키웠으나 전혀 영향을 끼치지 못하는 것으로 보였고, driver_mem 은 그대로였다. baseline 에 비해 worker_mem 은 5 배였고, cores 도 4 배였다. 유일하게 worker_inst 가 반으로 줄었다. worker_inst 가 의심스러운 상황이었다.

C. Tests: Wordcount

확인을 해야하기 때문에 다른 parameter 는 WASP 세팅 그대로 worker_inst 를 64 로 변경해서 다시 돌려보기로 했다. 놀랍게도 결과는:

Your score of wordcount is 402.07283506246030065636 / 100

예상치 못한 결과가 나왔다. 100% worker 인스턴스 수가 큰 영향을 끼치는 것으로 보인다. 이 뒤로 다른 parameter 를 하나씩 baseline 으로 되돌리면서 다시 돌려보기 시작했다. 되돌린 순서는 스코어에 악영향으로 보이는 것 혹은 영향을 끼치지 않는 것으로 보이는 것부터 되돌렸다:

[parallelism=(4)] 435.96155894886363636363 / 100

[cores=(1)] 536.94905778591739360933 / 100

[worker_mem=(2g)] 475.51089367253750815394 / 100

놀랍게도 parameter 를 하나씩 되돌릴 때마다 스코어가 높아졌다. 마지막의 경우 오히려 스코어가 떨어졌지만 메모리 부족 에러가 나왔기 때문인 것으로 보인다. 메모리는 10g 를 유지하거나 키우는 것이 좋을 것으로 보였다.

실험삼아 worker_mem=(12g), worker_inst=(128)로 설정하고 결과를 보기로 했다. 이대로라면 메모리 부족 에러가 나지 않는 이상 가장 좋은 스코어가 나와야할 것이다. 하지만 메모리 부족은 커녕 메모리를 제대로 배당할 수 없다는 에러메세지가 출력되었다. 인스턴스 수가 너무 많기 때문인 것으로 보인다. 고로 다시 인스턴스 수는 64 로 줄였다:

Your score of wordcount is 497.92757898743814236771 / 100

현재까지 결과를 보아서는 worker 메모리를 10g 로 두는 것이 가장 나아보인다.

마지막으로 실험삼아 cores=2, worker_inst=32 로 최종 코어 수는 같지만 인스턴스 수를 줄여서 실행해 보았다:

Your score of wordcount is 163.42869064952590645974 / 100

무려 300 가량 스코어 차이가 난다.

D. Tests: Sort

위의 결과를 따라 sort 도 마찬가지로 parameter 를 설정하기로 했다:

parallelism=4

driver_mem=2g

worker_mem=10g

worker_inst=64

cores=1

결과는 만족스러웠다. 굳이 더 수정할 필요성을 느끼지 못했다:

Your score of sort is 401.59987822822060987366 / 100

4. Results

A. Parameters

parallelism=(4 4)

driver_mem=(2g 2g)

worker_mem=(10g 10g)

worker_inst=(64 64)

cores=(1 1)

B. Scores

Your score of wordcount is 496.54562277580071174377 / 100

Your score of sort is 323.36890045543266102797 / 100

(역시 함께 실행하면 스코어가 더 낮게 나오는 듯 하다.)

C. After-thoughts

Caching 을 시도해봤어야 하는데 시도하기도 전에 높은 스코어가 나와서 시도할 일이 없었다. 사실 레포트 작성을 거의 모두 마친 후에야 caching 에 대해 떠올릴 수 있었다. 한가지 더 신경 쓰이는 점은 왜 이렇게 worker 인스턴스의 영향력이 강한가이다. 다른 어떤 paramter 도 worker_inst 만큼의 영향력을 보이지 않았다. 노드 안에 인스턴스가 있고 그 안에 스레드인 코어가 있는데, 최종 코어 수가 같더라도 유의미하게 큰 차이의 결과가 나타난다. 인스턴스 안 코어 수를 늘리는 것이 관리에 할당되는 메모리를 높여 오버헤드가 발생하게 된다고 가정하더라도 지나치게 스코어 차이가 난다고 생각한다.