postgreSQL의 Sort-Merge Join을 이용한 SSD와 HDD의 Temp tablespace로서의 성능 비교

이재훈°, 박종혁, 이상원
성균관대학교
{ljhh0611, akindo19, swlee} @ skku.edu

Comparison SSD and HDD for Temp tablespace using Sort-Merge Join in postgreSQL

Jae-Hun Lee°. Jong-Hyeok Park, Sang-Won Lee
Sungkyunkwan University

요 약

빠른 성능과 한층 저렴해진 SSD의 출현으로 SSD의 사용률이 증가하는 추세이다. 하지만 대부분의 상용 DBMS는 HDD에 최적화된 알고리즘으로 설계되었다. SSD는 HDD보다 Random IO가 현저히 빠르기 때문에 이러한 특성을 활용하여 DBMS 설계에 적용한다면 성능을 향상시킬 수 있다. 본 논문에서는 오픈소스 DBMS인 postgreSQL환경에서 sort merge join 과정의 Temp tablespace를 각각 SSD와 HDD로 하였을 때, work_mem 크기와 client 수에 따른 성능을 비교하였다. 실험 결과, SSD를 사용하였을 때 HDD보다 성능향상이 있었다. 이는 SSD의 특성을 고려한 DBMS의 설계의 필요성을 제기한다.

1. 서 론

저발열, 저전력, 무소음, 가벼운 무게, 충격에 강한 장점을 가진 저렴한 가격의 SSD(Solid-State Drive)의 출현으로 개인컴퓨터는 물론 기업 서버시장에서도 저장장치로 HDD(Hard Disk Drive)보다 SSD의 사용률이 증가하는 추세이다. 반면, 대부분의 상업용 DBMS는 메모리를 디스크에 대한 버퍼로 사용하고 메모리에 비해 수백 배 느린 HDD를 저장장치로 가정하여 이를 보완할 수 있는 다양한 알고리즘으로 최적화되어 있다.

SSD는 HDD보다 Random IO 속도가 현저히 빠르기 때문에 DBMS에서 이러한 특성을 활용하면 높은 성능을 낼 수 있다. 따라서 적합한 알고리즘과 기법들이 연구개발되고 있다. [1] 오라클 환경에서의 경우 작은 단위의 Random I/O를 하는 것을 활용함으로써, HDD에 비하여 6배 성능이 향상된 연구결과가 존재한다.

객체관계형 DBMS 오픈소스인 postgreSQL 또한 기존의 상업용 DMBMS처럼, 저장장치를 HDD로 가정하고 그에 최적화 하여 설계되어있다. SSD의 특성을 활용한다면 postgreSQL 역시 성능향상을 기대할 수 있다.

본 논문에서는 postgreSQL환경에서 Temptablespace로 각각 SSD와 HDD를 사용하였을 때, SMJ(Sort-Merge Join) 과정에서 성능을 비교하는 실험을 통해 성능 차이를 비교 한다. 실험 결과를 분석하면서 postgreSQL 에서 SSD를 저장장치로서 활용하는 것에 대한 효율성을 재고해본다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 Postgres에서의 SMJ과정을 살펴보고, 3장에서는 2장에서 살펴본 특성을 기반으로 postgreSQL환경에서 Temptablespace를 각각, SSD와 HDD로 사용했을 때의 성능 비교 실험을 한 뒤, 실험 결과에 대한 분석을 한다. 4장에서는 실험 결과의 결론을 맺고 향후 연구로 논문을 마무리 한다.

2. Sort-Merge Join in PostgreSQL

PostgreSQL의 SMJ은 [2] 다중페이즈 병합 알고리즘을 이용하여 External Sort를 한 뒤 Join한다. 이 때 External Sort 알고리즘은 전부 work_mem이라는 main memory 공간에서 진행 된다. External Sort는 크게 Initial Sort phase와 Merge phase로 구분 할 수 있다.

2.1 Initial Sort phase

Initial Sort는 가장 먼저 relation의 tuple들을 run으로 나눈다. 초기 run의 크기는 평균적으로 work_mem의 크기와 동일하다. run을 하나의 단위로 두고, Tape에 분류한다. 이 때, Tape는 work_mem의 크기에 따라 결정되는 값이고, run들을 논리적으로 구분하는 단위로 사용된다. 마지막 Tape는 비워두기 위해 Sort phase 에서는 사용하지 않는다. 이때, level이라는 단위를 만들어 level에 따라 Tape에 넣는 run의 개수를 정하는데, level 1일 때 각 Tape들에 넣을 수 있는 run의 개수는 1개이다. 현재 level에서 모든 Tape에 run을 값만큼 넣으면 level을 올리는데, 새로 정하는 값은 아래 [표 1]과 같다.

[표 1] 새 level t 에서 Tape에 들어갈 run의 값

$T_n(t)$	$T_1(t-1)$
$T_i(t)$ $(1 \le i < n)$	$T_1(t-1) + T_{i+1}(t-1)$

표를 설명하면, $T_k(t)$ $(1 \le k \le n$, T_n= 마지막 Sort Tape) 는

k번째 Tape가 t-level에서 채울 run의 개수를 의미한다. 새 level의 마지막 Tape인 $T_n(t)$는 이전 level의 첫 번째 Tape $T_1(t-1)$의 값이고, 그 외의 Tape $T_i(t)$는 이전 level의 첫 번째 Tape와 자신 다음의 Tape $T_{i+1}(t-1)$값의 합으로 결정된다.

이런 방식으로 level을 올리면서 초기 run들을 Tape에 모두 분류했다면, 각 Tape 마다 다 채우지 못한 값은 dummy값으로 두어 실제로는 없지만 run이 있는 것으로 가정하여 모든 Tape가 값에 맞추어 run들이 채워진 것처럼 만든다. run들을 분류할 때는 최대한 Tape들의 dummy값이 같도록 분류한다. 위의 과정을 모두 마치면 Merge phase를 위한 Initial Sort를 마친다.

2.2 Merge phase

Merge phase에서, 각 Tape들은 비어있는 하나의 Target Tape에 run을 merge하여 저장한다. run이 Tape 마다 불균등한 수로 분배되어있기 때문에, 모두 같은 양의 run을 Target Tape에 merge하게 되면 한 Tape만 run이 모두 소모 되어 빈 Tape가 된다. 그 때, level을 하나 내리고, 빈 Tape를 Target Tape로 바꾼 뒤, 바꾼 Target Tape에 merge를 계속하는 식으로 Merge phase가 진행된다. postgreSQL에서는 level이 1이 되고 Tape에 run이 하나씩 있게 되면 sort를 종료하고, 남은 한 번의 merge는 join을 하는 phase에서 on-the-fly하게 진행한다.

2.3 Join phase

Join algorithm은 일반적인 SMJ와 마찬가지로 sorted inner relation과 sorted outer relation을 첫 번째부터 차례로 비교하면서 Join하고, qualification을 만족하는 tuple들을 select하는 방식을 따른다.

3. 성능 비교 실험

[표 2] 실험환경

운영체제	Ubuntu 14.04.3 LTS
프로세서	Intel(R) Xeon(R) CPU X5650 @ 2.67GHz (24 CPUs)
메모리 (RAM)	4.00 GB
저장장치	Samsung 850 PRO SSD, Hitachi HDS721010DLE630 HDD
데이터베이스	PostgreSQL 9.4.1

성능 비교에 사용된 테이블의 크기는 각각 0.5GB와 5GB 이고, client 마다 독립적으로 테이블에 접근하도록 했다. 자세한 실험 환경은 [표 2]와 같다.

본 논문의 실험은 work_mem의 크기와 동시 접속하여 작업하는 client 수를 변화시키며 진행했다. work_mem 크기는 default인 4MB부터 16MB, 32MB, 64MB로 두었고, client 수는 1부터 16까지 증가시켰다. 작업을 수행한 클라이언트들의 평균 execution time을 측정하는 방법으로 실험을 진행했다.

위의 환경에서 진행한 실험 결과는 HDD의 경우 아래의 [표 3]와 같고, SSD의 경우는 [표 4]와 같다.

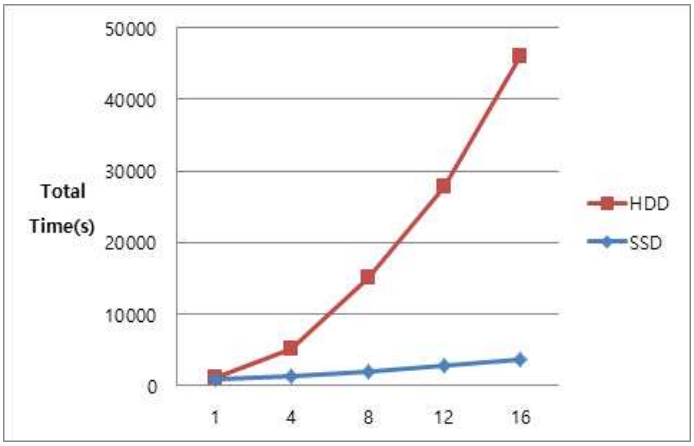
[표 3] HDD 실험 결과 (단위: 초)

	work_mem size			
clients	4MB	16MB	32MB	64MB
1	1185.177	1001.804	882.007	897.848
4	5246.500	2084.843	1264.290	1105.893
8	15033.745	4427.630	2294.230	1646.733
12	27847.074	7103.203	3362.652	2490.273
16	45997.846	11050.824	4594.552	3691.786

[표 4] SSD 실험 결과 (단위: 초)

	work_mem size			
clients	4MB	16MB	32MB	64MB
1	927.963	856.117	818.601	825.985
4	1350.994	971.608	825.0535	827.7993
8	2089.262	1322.306	968.9605	900.1268
12	2775.197	1690.184	1304.005	1021.796
16	3602.078	2171.048	1675.454	1286.495

work_mem의 default 크기인 4MB일 때를 그래프로 비교하여 나타내면 [그림 1]과 같다.



[그림 1] work_mem이 4MB일 때 Total execution Time비교

성능 측정 및 비교 결과, Total execution Time은 평균적으로 SSD일 때 3.5배 정도 더 빨랐으며, 특히 work_mem이 4MB이고 Clients수가 16인 상황에서, SSD를 사용할 때가 HDD를 사용 할 때보다 12.8배 더 빠르다. 전반적으로 client 수가 많을수록 execution time 차이가 크고, work_mem이 클수록 그 차이가 줄어드는 양상을 보였다. SMJ total execution time을 External Sort time과 Join time으로 나누어보면 HDD의 경우 [표 5], SSD의 경우 [표 6]과 같다.

[표 5] HDD Sort & Join time (단위: 초)

		work_mem size			
type	clients	4MB	16MB	32MB	64MB
Sort	1	428.5	248.1	119.6	53.0
	4	4404.7	1243.2	427.0	206.0
	8	11470.1	2956.6	905.0	395.6
	12	21107.3	5082.6	1244.1	643.6
	16	35572.1	7923.7	1606.2	972.3
Join	1	756.7	753.7	762.4	844.9
	4	841.8	841.6	837.3	899.9
	8	3563.7	1470.1	1389.2	1251.1
	12	6739.8	2020.6	2118.5	1846.7
	16	10425.8	3127.1	2988.3	2719.5

[표 6] SSD Sort & Join time (단위: 초)

		work_mem size			
type	clients	4MB	16MB	32MB	64MB
Sort	1	163.4	88.6	54.6	44.8
	4	673.3	266.2	128.6	121.6
	8	1365.6	546.3	252.8	208.4
	12	2023.3	799.0	369.9	288.5
	16	2673.7	1054.2	514.4	380.4
Join	1	764.6	767.6	764.0	781.2
	4	677.7	705.4	696.5	706.2
	8	723.7	776.0	716.2	691.7
	12	751.9	891.1	934.0	733.2
	16	928.4	1116.8	1161.1	906.1

[표 7]과 [표 8]은 Join 시의 device utilization을 측정한 것이다.

[표 7] Join시 HDD의 utilization (단위: %)

	work_mem size			
clients	4MB	16MB	32MB	64MB
1	15.701	10.926	19.763	9.767
4	97.589	97.865	87.841	50.739
8	95.737	77.885	99.412	92.386
12	98.106	99.521	98.907	98.052
16	99.889	95.665	94.401	99.006

[표 8] Join시 SSD의 utilization (단위: %)

	work_mem size			
clients	4MB	16MB	32MB	64MB
1	2.360	1.904	2.676	2.677
4	10.720	9.906	9.769	9.812
8	35.861	17.819	19.876	19.020
12	57.851	22.042	21.208	25.555
16	64.828	24.1778	22.440	28.471

[표 5]와 [표 6]에서, SSD와 HDD의 경우 모두 Sort time의 변화 양상은 total execution time의 비율 양상과 유사하다. 그러나 Join time에선 HDD의 경우는 client 수가 증가할수록 시간도 증가하는 반면, SSD의 경우 Join time은 client수의 영향을 적게 받는다. [표 7]과 [표 8]에 따르면 HDD에서 Join utilization은 client수에 따라 급격히 증가하여 100%에 가까워지는 반면, SSD는 client수가 많아져도 50% 내외의 값을 유지한다. 이는 SMJ과정에서 Random IO가 많이 발생하며 HDD는 Random IO에 취약하기 때문이다.

4. 결론 및 향후 연구

본 논문에서는 postgreSQL환경의 Temp tablespace로 HDD와 SSD를 사용했을 경우 SMJ execution time을 통해 성능을 비교하였다.

성능 비교 결과, SSD를 사용하는 경우가 HDD를 사용하는 경우보다 평균 3.5배 빠른 성능 향상이 있었다. work_mem이 작고 client 수가 많은 상황에서는 Join time의 차이로 인해 최대 약 13배가량 더 빠른 성능을 보였다. HDD는 client 수가 증가함에 따라 join time이 급격히 늘어나고 total execution time도 급격히 증가한다. 그러나 SSD는 client수가 증가해도 join time의 차이는 크지 않고, total execution time의 변화도 크지 않다.

향후 연구에서는 현재 HDD에 최적화 되어있는 External Sort algorithm을 SSD에 맞게 최적화 하여 성능을 다시 비교 해 볼 계획이다.

참고문헌

[1] S.-W. Lee, B. Moon, C. Park, J.-M. Kim, and S.-W. Kim, A case for flash memory ssd in enterprise database applications, In SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pp.1075

[2] Knuth, Donald Ervin. *the art of computer programming volume 3.* Massachusetts: Addison-Wesley, 1998.

사사

이 논문은 2016년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (R0126-16-1108, 비휘발성 메모리 기반 개방형 고성능 DBMS 개발)