

Searching Oracle Text Index with Dynamic Parallel Query

By Igor Geller

Overview

- ◆ The project is done in collaboration with Oracle database groups
- ◆ The project aims to improve the processing speed of Oracle Text Queries on Oracle Database
- ◆ The improvement from performance will come from parallelization

Background

- ◆ Oracle Text Index is sorted by tokens
- ◆ For every word, a query returns a row with compressed list of docids containing the given word and their offsets
- ◆ For every match, it calculates the relevance function and returns the results sorted by relevance
- ◆ As of now only one slave is processing one table

Description

- ◆ A typical Oracle Text query looks like:

select avg(story_seq_num), count() from t_stories where CONTAINS(search_entitlements, 'Harvard')>0;*

where search_entitlements is a set of columns we are searching through.

- ◆ Adding parallel functionality enables the following command:

select /+ parallel */ avg(story_seq_num), count(*) from t_stories where CONTAINS(search_entitlements, 'Harvard')>0;*

this command will spawn multiple slaves that will each call the same function passing a unique number as their slave_id number.

Implementation

For this prototype I was hijacking the partitioned table's parallel slaves in order to achieve parallelism

- ◆ After spawning parallel slaves in different partitions, I re-directed them to partition number one that contained all the data
- ◆ The workload split was decided based on rowid ranges
- ◆ Many additional checks to make sure nothing is changed for the serial case

Testing

- ◆ Basic testing for correctness was done on a small test-case and did not show any benchmark improvement (in fact showed some delay)
- ◆ Benchmark done on the extensive table with 300,000 articles showed an improvement from about 5.5 seconds to 3.5 seconds when parallelism of 4 was used

Problems

- ◆ Since the spawning of parallel slaves was not implemented by the Oracle Parallel Query Database, the prototype was created by hacking into the code rather than properly implementing the functionality
- ◆ The observed improvement was not as good as was hoped for indicating that there might be a redundant overhead introduced

Conclusion

- ◆ The project will be continued in the summer in order to achieve the goal of a proper efficient implementation
- ◆ Working on the project helped me familiarize with the Oracle Text Group code which will be very useful when I start working
- ◆ Taking Parallel Computing class was a great help and inspiration for me to apply the learned theory directly to industrial practice.