```
*** First, setup a 16K tablespace

SQL> alter system set db_16k_cache_size = 80m;

System altered.


SQL> create tablespace ts_16k datafile 'c:\oracle\product\10.2.0\bowie\ts_16k01.dbf' size
1024M uniform size 1M segment space management manual blocksize 16K;

Tablespace created.


*** Next create a simple little table that initially has 2M rows

SQL> CREATE TABLE bowie AS SELECT rownum id, sysdate-10000+mod(rownum,10000) date_field FROM
dual CONNECT BY LEVEL <=2000000;

Table created.


*** Create an index in the default 8K tablespace

SQL> CREATE INDEX bowie_i ON bowie(date_field);

Index created.


*** Now evenly delete 1/2 the rows to defragment the index


SQL> DELETE bowie WHERE mod(id,2)=1;

1000000 rows deleted.


SQL> COMMIT;

Commit complete.


*** Collect current stats

SQL> exec dbms_stats.gather_table_stats(ownname=>'BOWIE', tabname=>'BOWIE',
estimate_percent=>null, method_opt=>'FOR ALL COLUMNS SIZE 1');

PL/SQL procedure successfully completed.


*** Note the db_file_multiblock_read_count is set to 16


SQL> show parameter db_file_multiblock

NAME                                 TYPE        VALUE

------------------------------------ ----------- ------------------------------

db_file_multiblock_read_count        integer     16


SQL> set autotrace traceonly
```

```
*** Flush the buffer cache to make the database have to work hard for the required blocks


SQL> alter system flush buffer_cache;

System altered.


*** Trace the session, very important


SQL> alter session set events '10046 trace name context forever, level 12';

Session altered.


*** Now run some SQL that performs a multiblock read of the bowie_i via an Index Fast Full
Scan (Note: run the query first before the buffer cache flush to reduce parsing related
overheads)


SQL> SELECT /*+ index_ffs (bowie, bowie_i) */ date_field FROM bowie where date_field > '01-
JAN-1978';

1000000 rows selected.

Execution Plan

----------------------------------------------------------

Plan hash value: 251320893

-------------------------------------------------------------------------------------

| Id  | Operation            | Name    | Rows  | Bytes | Cost (%CPU)| Time     |

-------------------------------------------------------------------------------------

|   0 | SELECT STATEMENT     |         |  1000K|  7812K|  1173   (2)| 00:00:15 |

|*  1 |  INDEX FAST FULL SCAN| BOWIE_I |  1000K|  7812K|  1173   (2)| 00:00:15 |

-------------------------------------------------------------------------------------

Predicate Information (identified by operation id):

---------------------------------------------------

   1 - filter("DATE_FIELD">TO_DATE('1978-01-01 00:00:00', 'yyyy-mm-dd

              hh24:mi:ss'))

Statistics

----------------------------------------------------------

          0  recursive calls

          0  db block gets

       6320  consistent gets

       5307  physical reads

          0  redo size
```

```
    5167309  bytes sent via SQL*Net to client
      11385  bytes received via SQL*Net from client
       1001  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
    1000000  rows processed


*** Note that consistent reads is 6320


*** Turn tracing off
SQL> alter session set events '10046 trace name context off';
Session altered.


*** If we look at a tkprof report of the query ...


SELECT /*+ index_ffs (bowie, bowie_i) */ date_field FROM bowie where date_field > '01-JAN-
1978'


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch     1001      0.73       1.23       5307       6320          0     1000000
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total     1003      0.73       1.23       5307       6320          0     1000000


*** we note that CPU is 0.73 seconds, elapsed is 1.23 seconds and query reads is 6320


WAIT #8: nam='db file scattered read' ela= 738 file#=5 block#=242233 blocks=16 obj#=59933
tim=341634522940


*** We note in the raw trace file that most scattered reads read 16 blocks at a time as
expected with a db_file_multiblock_read_count set to 16


*** we want to try and improve this performance.


*** I read somewhere that moving indexes into a bigger block tablespace should help, let's
give it a go ...
```

```
*** So, let's rebuild the index in the 16K tablespace


SQL> alter index bowie_i rebuild tablespace ts_16k;

Index altered.


SQL> alter system flush buffer_cache;

System altered.


SQL> alter session set events '10046 trace name context forever, level 12';

Session altered.


*** Let's see if the performance of the query has improved ? (Note: run the query first
before the buffer cache flush to reduce parsing related overheads)


SQL> SELECT /*+ index_ffs (bowie, bowie_i) */ date_field FROM bowie where date_field > '01-
JAN-1978';

1000000 rows selected.


Execution Plan

----------------------------------------------------------

Plan hash value: 251320893

--------------------------------------------------------------------------------

| Id  | Operation            | Name    | Rows  | Bytes | Cost (%CPU)| Time     |

--------------------------------------------------------------------------------

|   0 | SELECT STATEMENT     |         |  1000K|  7812K|   367   (4)| 00:00:05 |

|*  1 |  INDEX FAST FULL SCAN| BOWIE_I |  1000K|  7812K|   367   (4)| 00:00:05 |

--------------------------------------------------------------------------------

Predicate Information (identified by operation id):

---------------------------------------------------

   1 - filter("DATE_FIELD">TO_DATE('1978-01-01 00:00:00', 'yyyy-mm-dd

              hh24:mi:ss'))

Statistics

----------------------------------------------------------

          1  recursive calls

          0  db block gets

       2317  consistent gets

       1310  physical reads
```

```
          0  redo size
    5167309  bytes sent via SQL*Net to client
      11385  bytes received via SQL*Net from client
       1001  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
    1000000  rows processed


*** Well, for a start, the consistent reads has dropped significantly from 6230 to just 2317
!!


SQL> alter session set events '10046 trace name context off';

Session altered.


SELECT /*+ index_ffs (bowie, bowie_i) */ date_field FROM bowie where date_field > '01-JAN-
1978'


call     count       cpu    elapsed       disk      query    current        rows

------- ------  -------- ---------- ---------- ---------- ----------  ----------

Parse        1      0.00       0.00          0          0          0           0

Execute      1      0.00       0.00          0          0          0           0

Fetch     1001      0.79       1.08       1310       2317          0     1000000

------- ------  -------- ---------- ---------- ---------- ----------  ----------

total     1003      0.79       1.08       1310       2317          0     1000000




*** Fantastic !! Elapsed times has dropped from 1.23 down to just 1.08 (somewhat over 12%
improvement, not bad at all).

*** We note that CPU has gone up a touch from 0.73 to 0.79, but what the hell, response times
have improved.

*** This is PROOF, beyond question that the 16K block size has improved performance, by a
significant 12% !!

*** Isn't that right ??


WAIT #1: nam='db file scattered read' ela= 767 file#=7 block#=46 blocks=8 obj#=59933
tim=342035983379


*** Note that scattered reads only read as a maximum 8 blocks (instead of 16 with the 8K
default) so the size of a multiblock read is the same as before.

*** Although the rebuild in the 16K block tablespace has improved performance, I wonder what
the result would be if we simply rebuilt the index in the old 8K block tablespace
```

```
SQL> alter index bowie_i rebuild tablespace bowie;

Index altered.


SQL> set autotrace traceonly

SQL> alter system flush buffer_cache;

System altered.


SQL> alter session set events '10046 trace name context forever, level 12';

Session altered.


*** (Note: run the query first before the buffer cache flush to reduce parsing related
overheads)


SQL> SELECT /*+ index_ffs (bowie, bowie_i) */ date_field FROM bowie where date_field > '01-
JAN-1978';


1000000 rows selected.


Execution Plan

----------------------------------------------------------

Plan hash value: 251320893

--------------------------------------------------------------------------------

| Id  | Operation            | Name    | Rows  | Bytes | Cost (%CPU)| Time     |

--------------------------------------------------------------------------------

|   0 | SELECT STATEMENT     |         |  1000K|  7812K|   594   (3)| 00:00:08 |

|*  1 |  INDEX FAST FULL SCAN| BOWIE_I |  1000K|  7812K|   594   (3)| 00:00:08 |

--------------------------------------------------------------------------------

Predicate Information (identified by operation id):

---------------------------------------------------

   1 - filter("DATE_FIELD">TO_DATE('1978-01-01 00:00:00', 'yyyy-mm-dd

              hh24:mi:ss'))

Statistics

----------------------------------------------------------

          0  recursive calls

          0  db block gets

       3664  consistent gets

       2655  physical reads
```

```
          0  redo size
    5167309  bytes sent via SQL*Net to client
      11385  bytes received via SQL*Net from client
       1001  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
    1000000  rows processed
```

*** Note that the consistent reads although not as low as with the 16K block tablespace, is much lower that it was originally

```
SQL> alter session set events '10046 trace name context off';

Session altered.
```

```
SELECT /*+ index_ffs (bowie, bowie_i) */ date_field FROM bowie where date_field > '01-JAN-1978'
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1001	0.65	0.99	2655	3664	0	1000000
total	1003	0.65	0.99	2655	3664	0	1000000

*** Oh my goodness, isn't that interesting.

*** Note only is elapsed times better than it was originally, it's actually better than the index in the 16K tablespace, just 0.99 compared to 1.08

*** Note also that CPU is significantly less at just 0.65 compared to 0.79 in the 16K tablespace

```
WAIT #1: nam='db file scattered read' ela= 668 file#=5 block#=222953 blocks=16 obj#=59933 tim=342792899797
```

*** Scattered reads are now back to reading 16 blocks at a time.

*** So it wasn't so much that the 16K tablespace made the index more efficient, but the fact the index *rebuild* made the index more efficient ...