1. In the following sequence of events, we use *R*i(*X*); to mean "transaction *Ti* starts, and its read set is the list of database elements *X*." Also, *Vi* means "*Ti* attempts to validate," and *W*i(*X*); means that "*Ti* finishes, and its write set was *X*." Tell what happens when each sequence is processed by a validation-based scheduler.

a) *R*1(*A, B*); *R*2(*B, C*); *V1*; *R*3(*C, D*); *V3*; *W*1(*A*); *V2*; *W*2(*A*); *W*3(*B*); **(4 marks)**  
b) *R*1(*A, B*); *R*2(*B, C*); *V1*; *R*3(*C, D*); *V3*; *W*1(*A*); *V2*; *W*2(*A*); *W*3(*D*); **(4 marks)**

1. a) R1(A, B); R2(B, C); V1; R3(C, D); V3; W1(A); V2; W2(A); W3(B);

RS1{A, B}, WS1{A}; RS2{B, C}, WS2{A}; RS3{C, D}, WS3{B}

V1 validates, V3 validates, however during V2, RS2{B,C} conflicts with

WS{B} hence T2 is rolled back.

b) R1(A, B); R2(B, C); V1; R3(C, D); V3; W1(A); V2; W2(A); W3(D);

RS1{A, B}, WS1{A}; RS2{B, C}, WS2{A}; RS3{C, D}, WS3{D}

V1 validates, V3 validates, and V2 validates.

How many such DataNodes you would need to read 100TB data in your Hadoop Cluster? The configuration of each available DataNode is as follows: You have a Hadoop Cluster with replication factor = 3 and block size = 64 MB. How long it takes to read it?

• 8 GB RAM

• 10 TB HDD

• 100 MB/s read-write speed



So 30 Nodes will takes 291.27/30 = 9.7 hours

1. Assume that you have just built a dense B+ tree index using Alternative on a heap file containing 20,000 records. The key field for this B+ tree index is a 40-byte string, and it is a candidate key. Pointers (i.e., record ids and page ids) are (at most) 10-byte values. The size of one disk page is 1000 bytes. The index was built in a bottom-up fashion using the bulk-loading algorithm, and the nodes at each level were filled up as much as possible.

a. How many levels does the resulting tree have? **(10 marks)**

b. For each level of the tree, how many nodes are at that level? **(10 marks)**

A.Since the index is a primary dense index, there are as many data entries in the B+ tree as records in the heap file. An index page consists of at most 2*d* keys and 2*d*+1 pointers. So we have to **maximize *d* under the condition that**

**2*d·*40+(2*d*+1)*·*10 *≤* 1000**.

The solution is ***d* = 9,** which means that we can have 18 keys and 19 pointers on an index page. A record on a leaf page consists of the key field and a pointer. Its size is 40+10=50 bytes. Therefore a leaf page has space for (1000/50)=20 data entries. The resulting tree has

**log19 (20000*/*20)+ 1= 4** levels.

B. Since the nodes at each level are filled as much as possible, there are 20000*/*20= 1000 leaf nodes (on level 4). (A full index node has 2*d*+1 = 19 children.)

Therefore there are 1000*/*19= 53 index pages on level 3, 53*/*19= 3 index pages on level 2, and there is one index page on level 1 (the root of the tree).