HBase Prefix Compression Trie Overview, v1

HBASE-4676: https://issues.apache.org/jira/browse/HBASE-4676
3/13/2012, Matt Corgan, mcorgan@hotpads.com

This following illustrates what a set of KeyValues looks like in encoded form

1) Figure 1: unencoded input KeyValues
    A) key length often uses 1 of 4 bytes
    B) value length often uses 1 of 4 bytes
    C) row key length often uses 1 of 2 bytes
    D) row key bytes repeat for every cell in the row
    E) family is always the same in an HFile and repeats in full in every cell
    F) qualifiers often repeat between row
    G) 8 byte timestamp is often the same for adjacent cells, and often close within a block
    H) KeyValue$Type is one byte per cell and is often 100% type=Put in a given block
    I) values are arbitrary byte[], which are all “VvvV” in this example. i do not aim to compress them

	A	B	C	D	E	F	G	H	I	J
1	key Length	value Length	row Key Length	row Key	family Length	family	qualifier	timestamp	op Type	value
2	0051	0004	29	com.dablog/2011/10/04/boating	4	hits	Chrome	1234567890	P	VvvV
3	0048	0004	29	com.dablog/2011/10/04/boating	4	hits	IE8	1234567890	P	VvvV
4	0052	0004	29	com.dablog/2011/10/04/boating	4	hits	IE9beta	1234567890	P	VvvV
5	0050	0004	28	com.dablog/2011/10/09/lasers	4	hits	Chrome	1234567890	P	VvvV
6	0047	0004	28	com.dablog/2011/10/09/lasers	4	hits	IE8	1234567890	P	VvvV
7	0051	0004	28	com.dablog/2011/10/09/lasers	4	hits	IE9beta	1234567890	P	VvvV
8	0039	0004	17	com.jamiesrecipes	4	hits	Chrome	1234567890	P	VvvV
9	0036	0004	17	com.jamiesrecipes	4	hits	IE8	1234567890	P	VvvV
10	0040	0004	17	com.jamiesrecipes	4	hits	IE9beta	1234567890	P	VvvV
11	0044	0004	22	com.jamiesrecipes/eggs	4	hits	Chrome	1234567890	P	VvvV
12	0041	0004	22	com.jamiesrecipes/eggs	4	hits	IE8	1234567890	P	VvvV
13	0045	0004	22	com.jamiesrecipes/eggs	4	hits	IE9beta	1234567890	P	VvvV

2) Figure 2: PtBlock meta

* the leading bytes of the class
* specify section widths inside block (row key section, qualifier section, value section, etc)
* mostly VInts
* parsed at beginning of each block access

	A	B	C	D
1	**encoded form**	**variable name**	**example value**	**variable description**
2	transient	arrayOffset	0	transient variable used if block data doesn't begin at byte[0]
3	transient	bufferOffset	0	transient variable used if accessing through a ByteBuffer (off heap cache)
4	VInt	numMetaBytes	32	num meta bytes at beginning of block (these variables)
5	VInt	numV1Bytes	688	num bytes if encoded in standard KeyValue format
6	VInt	numRowBytes	126	after meta bytes, num bytes occupied by row key trie
7	VInt	numFamilyBytes	6	then num bytes for family trie
8	VInt	numQualifierBytes	24	then num bytes for qualifier trie
9	VInt	numTimestampBytes	0	num bytes for timestamp deltas
10	VInt	numMemstoreTimestampBytes	0	num bytes for memstore timestamp deltas
11	VInt	numDataBytes	48	num bytes for all values concatenated at end of block
12	VInt	nextNodeOffsetWidth	1	row key trie inter-node pointer width
13	VInt	familyOffsetWidth	1	family trie inter-node pointer width
14	VInt	qualifierOffsetWidth	1	qualifier trie inter-node pointer width
15	VInt	timestampIndexWidth	0	FInt width of index entries at beginning of timestamp section
16	VInt	memstoreTimestampIndexWidth	0	FInt width of index entries at beginning of memstoreTS section
17	VInt	dataOffsetWidth	1	FInt width of FInts specifying value offsets
18	VInt	dataLengthWidth	1	FInt width of FInts specifying value lengths
19	VInt	rowTrieDepth	4	max depth of row key trie, used by reader (possibly unnecessary)
20	VInt	maxRowLength	29	used to allocate reader buffer (possibly unnecessary)
21	VInt	maxQualifierLength	7	used to allocate reader buffer (possibly unnecessary)
22	VLong	minTimestamp	1234567890	other timestamps stored as delta from this one
23	VInt	timestampDeltaWidth	0	FInt width of max delta
24	VLong	minMemstoreTimestamp	0	other memstoreTs's stored as delta from this one
25	VInt	memstoreTimestampDeltaWidth	0	FInt width of max memstoreTS delta
26	byte	allSameType	TRUE	true/false, are all KeyValue$Types the same in the block
27	byte	allTypes	4	value of all KeyValue$Types if all the same
28	VInt	numUniqueRows	0	unique row keys in block (possibly unused)
29	VInt	numUniqueFamilies	1	unique families in block (always 1 in current HFile) (possibly unused)
30	VInt	numUniqueQualifiers	3	unique qualifiers in block (possibly unused)

3) Figure 3: Qualifer trie section of block
    * trie nodes encoded as [tokenLength,token,parentOffset]
    * where parentOffset is offset from beginning of qualifier section
    * readers:
        * start with the end node offset
        * copy the current token to a buffer
        * move backwards to the parent until parentOffset=0 signifying the root node
    * uses this reverse-style trie becase:
        * we need random access to a qualifier pointed to by a row node
        * a qualifier can be referenced by multiple row nodes in the block, so a normal forward-trie doesn't quite work
            * a normal trie would have to have a qualifier trie for each row which wouldn't compress qualifiers between rows

	B	C	D	E	F	G	H
1	**B/N/L**	**PtColumnNode Writer toString**	**output Array Offset**	**token Length**	**token**	**parent Offset**	**rawBytes**
2			**FInt in row trie**	**VInt**	**bytes**	**FInt**	**concatenated**
3	Root	0,[]->0	0	0		0	0 0
4	Branch	2,[IE]->0	2	2	IE	0	2 IE 0
5	Leaf	6,[Chrome]->0	6	6	Chrome	0	6 Chrome 0
6	Leaf	14,[8]->2	14	1	8	2	1 8 2
7	Leaf	17,[9beta]->2	17	5	9beta	2	5 9beta 2

4) Figure 4: Row Trie section of the example block

* row key trie comes after block meta and is generally the biggest of the non-value sections of the block
* example has 7 nodes: 2 branch nodes (including the “com.” root), 1 nub, and 3 leaves
    * a Branch node is internal to the tree, has no cell occurrences, and only points at other “child” nodes
    * a Leaf node has no children, only cells
    * a Nub node has both children but also cells
* branches and nubs have a “fan” which contains the first byte of each child node’s token
    * the child therefore does not hold the first byte of its token
* nubs and leaves have a numOccurrences variable (the number of cells)
    * for each occurrence there is a range of bytes with offsets/lengths of the cell info
    * these are all FInts so we can access them randomly
    * if the block meta indicates so, then they may be empty and not use any bytes, for example if all timestamps are the same, or all qualifiers or all operationTypes
    * individual fields:
        * familyOffset: the offset of the last family token in the family section of the block
        * qualifierOffset: the offset of the last qualifier token in the qualifier section of the block
        * timestampIndex: index of the delta value in the timestamp section
        * memstoreTsIndex: index of the delta value in the memstoreTS section
        * operationType: the byte value of the KeyValue$Type for this cell
        * valueOffset: offset of value in the values section of the block
        * valueLength: length of value
* the 6 nodes are encoded as bytes and concatenated to make the row key section
* columns D through P illustrate the pre-serialized data, and Q illustrates the serialized version

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	**PtRowNodeWriterDebu toString**	**toString pointers**	**B/N/L**	**token Length**	**token**	**fan Out**	**num Cells**	**fan Byte**	**next Node Offset**	**family Offset**	**qualifie Offset**	**timesta Index**	**mem store TS Index**	**op Type**	**value Offset**	**value Lengt**	**raw Bytes**
2				**VInt**	**bytes**	**VInt**	**VInt**	**byte**	**FInt**	**FInt**	**FInt**	**FInt**	**FInt**	**FInt**	**FInt**	**FInt**	
3	B:com.,2,0,[	d:11	Branch	4	com.	2	0	d	11								4 com. 2 0 d 11
4		j:33]						j	33								j 33
5	B:ablog/2011/10/0,2,0,[	4:51	Branch	15	ablog/2011/10/0	2	0	4	51								15 ablog/2011/10/0 2 0 4 51
6		9:74]						9	74								9 74
7	N:amiesrecipes,1,3,{	[0:6::::24:4]	Nub							0	6				24	4	0 6 24 4
8		[0:14::::28:4]								0	14				28	4	0 14 28 4
9		[0:17::::32:4]}								0	17				32	4	0 17 32 4
10		,[/:74]		12	amiesrecipes	1	3	/	74								12 amiesrecipes 1 3 / 74
11	L:/boating,0,3,{	[0:6::::0:4]	Leaf	8	/boating	0	3			0	6				0	4	8 /boating 0 3 0 6 0 4
12		[0:14::::4:4]								0	14				4	4	0 14 4 4
13		[0:17::::8:4]}								0	17				8	4	0 17 8 4
14	L:/lasers,0,3,{	[0:6::::12:4]	Leaf	7	/lasers	0	3			0	6				12	4	7 /lasers 0 3 0 6 12 4
15		[0:14::::16:4]								0	14				16	4	0 14 16 4
16		[0:17::::20:4]}								0	17				20	4	0 17 20 4
17	L:eggs,0,3,{	[0:6::::36:4]	Leaf	4	eggs	0	3			0	6				36	4	4 eggs 0 3 0 6 36 4
18		[0:14::::40:4]								0	14				40	4	0 14 40 4
19		[0:17::::44:4]}								0	17				44	4	0 17 44 4

5) Figure 5: all sections of block combined

* example is simplified to eliminate the following sections
    * family section contains only family name: “hits”
    * timestamp section empty because all input timestamps were the same
    * memstoreTS section empty because the encoder set includeMemstoreTS to false
    * operation types section is empty because all operations were “Put”

	A	B
1	**block section**	**bytes in section**
2	BlockMeta	32 688 126 6 24 0 0 48 1 1 1 0 0 1 1 4 29 7 1234567890 0 0 0 TRUE 4 0 1 3
3	RowNodes	4 com. 2 0 d 11 j 33 15 ablog/2011/10/0 2 0 4 51 9 74 0 6 24 4 0 14 28 4 0 17 32 4 12 amiesrecipes 1 3 / 74 8 /boating 0 3 0 6 0 4 0 14 4 4 0 17 8 4 7 /lasers 0 3 0 6 12 4 0 14 16 4 0 17 20 4 4 eggs 0 3 0 6 36 4 0 14 40 4 0 17 44 4
4	FamilyNodes	hits
5	QualifierNodes	0 0 2 IE 0 6 Chrome 0 1 8 2 5 9beta 2
6	TimestampDeltas	[empty]
7	MemstoreTimestam	[empty]
8	Types	[empty]
9	Values	VvvVVvvVVvvVVvvVVvvVVvvVVvvVVvvVVvvVVvvVVvvVVvvVVvvV

6) Figure 6: raw input bytes

* approximate visualization of input vs output bytes

Input	0051000429com.dablog/2011/10/04/boating4hitsChrome1234567890 PVvvV0048000429com.dablog/2011/10/04/boating4hitsIE812345678 90PVvvV0052000429com.dablog/2011/10/04/boating4hitsIE9beta123 4567890PVvvV0050000428com.dablog/2011/10/09/lasers4hitsChrome 1234567890PVvvV0047000428com.dablog/2011/10/09/lasers4hitsIE81 234567890PVvvV0051000428com.dablog/2011/10/09/lasers4hitsIE9be ta1234567890PVvvV0039000417com.jamiesrecipes4hitsChrome123456 7890PVvvV0036000417com.jamiesrecipes4hitsIE81234567890PVvvV0 040000417com.jamiesrecipes4hitsIE9beta1234567890PVvvV00440004 22com.jamiesrecipes/eggs4hitsChrome1234567890PVvvV0041000422 com.jam iesrecipes/eggs4hitsIE81234567890PVvvV0045000422 com.jamiesrecipes/eggs4hitsIE9beta1234567890PVvvV
Output	3268812662400481110011429712345678900000TRUE40134com.20d11 j3315ablog/2011/10/020451974062440142840173241 2amiesrecipes13/ 748/boating03060401444017847/lasers030612401416401720 44eggs030 6364014404017444hits002IE06Chrome018259beta2VvvVVvvVVvvVVv vVVvvVVvvVVvvVVvvVVvvVVvvVVvvVVvvV