# Enduro/X Core - Feature #322

Feature # 357 (Closed): Enduro/X 7.0 master task

## Index for STRING and CARRAY fields in UBF

05/28/2018 01:57 PM - Madars

| | | | |
|---|---|---|---|
| **Status:** | New | **Start date:** | 05/28/2018 |
| **Priority:** | Normal (Code 4) | **Due date:** | |
| **Assignee:** | | **% Done:** | 0% |
| **Category:** | | **Estimated time:** | 0.00 hour |
| **Target version:** | | | |

**Description**

We could have a index of carray and string fields, either based on index frequency or red-black trees with cut off.

### History

#### #1 - 09/11/2018 06:36 AM - Madars

*- Parent task set to #328*

#### #2 - 12/03/2018 07:11 AM - Madars

*- Parent task changed from #328 to #357*

#### #3 - 01/24/2019 01:56 PM - Madars

Have new env:

NDRX_UBF_OPTS=alloc:+1000;index=16

this would allow to override tpalloc to have some extra space and enable default indexing. This would allow boost the application even with out the rebuild.

#### #4 - 01/24/2019 02:04 PM - Madars

Have new env:

NDRX_UBF_OPTS=alloc:+1000;index=16;indexmax=1000

this would allow to override tpalloc to have some extra space and enable default indexing. This would allow boost the application even with out the rebuild.

If having index=1, then every field in buffer is indexed. Lets have example with index=3.

Lets say we have abuffer
A B C D E F G H I J K

And we index with 3

then index will get to these (marked with scopes, the items non indexed before):

A(0) B C D(2) E F G(2) H I J(2) K

## add example

Lets assume that we insert something in buffer, lets say X Y Z in first block

step 1
A(1) X B C D(4) E F G(3) H I J(3) K
3 / 3 < 2, not adding new index entry

step 2:

A(1) X Y B C D(5) E F G(3) H I J(3) K

5 / (3) < 2, nok

step 3:

AXYZBCD(6)EF G(3) H I J(3) K
1234567

6 / 3 < 2, ok
elm to index: pos of "D" - 3, 7 - 3 => 4

A(1) X Y Z(3) B C D(3*) E F G(3) H I J(3) K

## Now try to delete something:

Lets say "B", we have buffer:

A(1) X Y Z(3) B C D(3*) E F G(3) H I J(3) K

after delete:

A(1) X Y Z(3) C D(2*) E F G(3) H I J(3) K

D(2) count < 3 index step, then we remove this element too from index, so that we keep the index balanced.

Thus we get a index:
A(1) X Y Z(3) C E F G(4) H I J(3) K

#### #5 - 01/24/2019 02:19 PM - Madars

1) On buffer resize/realloc: needs to move index to further end.
2) We add field to UBF and only then try to update index. Firstly we update counts after for which index region in passes. After that we try to add new index entry, if space allows us to do it. if not then silently we ignore the error.

#### #6 - 01/24/2019 02:40 PM - Madars

we could keep two index tables new offsets for carrays and strings. and process as normal fields, by keeping the data right after the carrays. Thus use already existing binary search algorithms used for other fixed fields.

#### #7 - 01/24/2019 02:48 PM - Madars

we could use the BFLD_IDX new type 8 for same carray and strings. just a common index.

#### #8 - 02/24/2020 02:54 PM - Madars

```
exprivate int get_fb_string_size(dtype_str_t *t, char *fb, int *payload_size)
{
    UBF_string_t *str = (UBF_string_t *)fb;
    int data_size = strlen(str->str) + 1;
    /* int aligned;
    int tmp;*/
    if (NULL!=payload_size)
        *payload_size = data_size;

    /*
    aligned = sizeof(BFLDID) + data_size;
    tmp=aligned%DEFAULT_ALIGN;
    aligned = aligned+ (tmp>0?DEFAULT_ALIGN-tmp:0);
    */

    /* Including EOS (+1) */
    /* return aligned; */
    return ALIGNED_SIZE(data_size);
}
```

For strings we also shall store the length of the string, otherwise when looping over it requires each string field to check the length with strlen(), which is not very optimal.

#### #9 - 03/26/2020 12:23 AM - Madars

So index could be new virtual field type which would store all IDs (sorted, filed type is index, field id is real field id). So if UBF is indexed (flag in header), we index all carrays and strings. We the additions to buffer are done twice added data + added index, if data is deleted, then also removed from index.

The data searched is in index area, the offset is read and data is returned. In case of update after field is removed, all indexes after the field must be updated. If we store the relative offset for the start of the type block, then we only need to update fields with correspoding data type i.e. if string is

removed or added, then only next string fields. This means that each time full scan of the index area is needed. Well here is nothing to do.

The index level will not be used. Either all indexed or nothing indexed.