プルリクエスト

#8 オープン metadata_adder_new_patches2 → master マージ 編集 却下 承認 0

New (fixed) patch set of metadata_adder discussion

概要 コミット 活動履歴

作成者 Kota Tsuyuzaki

レビューアー

ウォッチをやめる

プル リクエスト機能について

説明 To save comments on the first pull request[1], I made a new pull request, here.

Almost of the comments have been addressed but a few things to consider are remaining. (I'll mark up the remaining thing at the first pull request) However, I guess this pull request got ready to merge because the remaining things can be considered in 1.0.rc timeline.

This patch set has already been tested by both liberasurecode test suite and pyeclib test suite for all supported backends (especially, includes isa-l and shss) in my lab environment.

https://bitbucket.org/tsg-/liberasurecode/pull-request/7

コメント (1)

2 **Kevin Greenan**

3 Kota Tsuyuzaki I have a few minor comments. Other than that, it looks good to go. Leaving the final check and merge to Tushar Gohad

返信 • Create task • 2 hours ago

コメントを入力してください。

変更ファイル (11)

+6	−0	AUTHORS
+13	−0	include/erasurecode/erasurecode.h
+1	−1	include/erasurecode/erasurecode_backend.h
+7	−2	include/erasurecode/erasurecode_helpers.h
+3	−1	src/backends/isa-l/isa_l_rs_vand.c
+36	−26	src/backends/shss/shss.c 1
+19	−2	src/erasurecode.c
+58	−6	src/erasurecode_helpers.c 2
+2	−1	src/erasurecode_postprocessing.c
+11	−9	src/erasurecode_preprocessing.c 1
+85	−3	test/liberasurecode_test.c

AUTHORS 差分を横に並べて表示 ファイルを表示 編集

```
...
 6   6   * Eric Lambert
 7   7   * Mark W Storer
 8   8
     9  +Contributors
    10  +----------------
    11  +
    12  + * Kota Tsuyuzaki
    13  + * Ryuta Kon
    14  +
```

include/erasurecode/erasurecode.h 差分を横に並べて表示 ファイルを表示 編集

```
...
247 247  {
248 248      uint32_t    idx;                /* 4 */
249 249      uint32_t    size;               /* 4 */
    250 +    uint32_t    frag_adder_size;    /* 4 */
250 251      uint64_t    orig_data_size;     /* 8 */
251 252      uint8_t     chksum_type;        /* 1 */
252 253      uint32_t    chksum[LIBERASURECODE_MAX_CHECKSUM_LEN]; /* 32 */
```

```
327 328  */
328 329 int liberasurecode_get_minimum_encode_size(int desc);
329 330
    331 +/**
    332 + * This will return the fragment size, which is each fragment data
    333 + * length the backend will allocate when encoding.
    334 + *
    335 + * @param desc - liberasurecode descriptor/handle
    336 + *        from liberasurecode_instance_create()
    337 + * @param data_len - original data length in bytes
    338 + *
    339 + * @return fragment size
    340 + */
    341 +int liberasurecode_get_fragment_size(int desc, int data_len);
    342 +
330 343 /* ==~=*=~====~=*=~==~=*=~== liberasurecode Error codes =~=*=~==~=~=*=~==~== */
331 344
332 345 /* Error codes */
```

include/erasurecode/erasurecode_backend.h

差分を横に並べて表示 ファイルを表示 編集

```
109 109     char                        soversion[MAX_LEN]; /* EC backend shared library version */
110 110
111 111     struct ec_backend_op_stubs  *ops;               /* EC backend stubs */
112         -    int                        metadata_adder;     /* EC backend custom metadata adder -
    112     +    size_t                     metadata_adder;     /* EC backend custom metadata adder -
113 113                                                         * metadata_adder bytes are added to
114 114                                                         * the fragment size when allocating
115 115                                                         * data/parity fragment buffers */
```

include/erasurecode/erasurecode_helpers.h

差分を横に並べて表示 ファイルを表示 編集

```
45  45
46  46  typedef struct __attribute__((__packed__)) fragment_header_s
47  47  {
48      -    fragment_metadata_t meta;   /* 55 bytes */
    48  +    fragment_metadata_t meta;   /* 59 bytes */
49  49      uint32_t            magic;  /*  4 bytes */
50  50      uint32_t            libec_version; /* 4 bytes */
51  51      // We must be aligned to 16-byte boundaries
52  52      // So, size this array accordingly
53      -    uint8_t             aligned_padding[1];
    53  +    uint8_t             aligned_padding[13];
54  54  } fragment_header_t;
55  55
56  56  /* ==~=*=~==~=*=~==~=*=~==~=*=~==~=*=~==~=*=~==~=*=~==~=*=~==~=*=~==~=*=~==~=*=~== */
```

```
126 126  char *get_data_ptr_from_fragment(char *buf);
127 127  int get_data_ptr_array_from_fragments(char **data_array, char **fragments,
128 128          int num_fragments);
    129 +int get_fragment_ptr_array_from_data(char **frag_array, char **data,
    130 +        int num_data);
129 131  char *get_fragment_ptr_from_data_novalidate(char *buf);
130 132  char *get_fragment_ptr_from_data(char *buf);
131 133  uint64_t get_fragment_size(char *buf);
```

```
133 135  int get_fragment_idx(char *buf);
134 136  int set_fragment_payload_size(char *buf, int size);
135 137  int get_fragment_payload_size(char *buf);
    138 +int set_fragment_adder_size(char *buf, int size);
    139 +int get_fragment_adder_size(char *buf);
    140 +int get_fragment_buffer_size(char *buf);
136 141  int set_orig_data_size(char *buf, int orig_data_size);
137 142  int get_orig_data_size(char *buf);
138 143  int set_checksum(ec_checksum_type_t ct, char *buf, int blocksize);
```

src/backends/isa-l/isa_l_rs_vand.c

差分を横に並べて表示 ファイルを表示 編集

```
480 480
481 481     desc->k = args->uargs.k;
482 482     desc->m = args->uargs.m;
483     -   desc->w = ISA_L_W;
    483 +   if (args->uargs.w <= 0)
    484 +       args->uargs.w = ISA_L_W;
    485 +   desc->w = args->uargs.w;
484 486
485 487     /* validate EC arguments */
486 488     {
```

...

src/backends/shss/shss.c

差分を横に並べて表示 ファイルを表示 編集

...

```
31  31
32  32  #include <stdio.h>
33  33  #include <stdlib.h>
34      -#include <alloca.h>
35  34
36  35  #include "erasurecode.h"
37  36  #include "erasurecode_helpers.h"
```

...

```
40  39  /* Forward declarations */
41  40  struct ec_backend shss;
42  41  struct ec_backend_op_stubs shss_ops;
    42 +struct ec_backend_common backend_shss;
43  43
44  44  typedef int (*shss_encode_func)(char **, size_t, int, int, int, int, long long *);
45  45  typedef int (*shss_decode_func)(char **, size_t, int *, int, int, int, int, int, long long *);
```

...

```
67  67  #define SHSS_SO_NAME "libshss.so"
68  68  #endif
69  69  #define DEFAULT_W 128
70      -
71      -/* TODO:
72      -   metadata_adder is still in discussion. shss needs to a fixed value to allocate extra bytes
73      -   for *each* fragment. However, current liberasurecode calculates the extra bytes as
74      -   "(alined_data_size + metadata_adder) / k" so that shss has to define the METADATA as a big value
75      -   to alloc enough memory for the maximum number of k even if k is smaller than the maximum value.
76      -
77      -   i.e. (shss specification is)
78      -   Enough Extra Bytes (for *each* fragment): 32
79      -   The Maximum Number: 20 (k=20)
80      -*/
81      -#define METADATA 32 * 20
    70 +#define METADATA 32
82  71
83  72  static int shss_encode(void *desc, char **data, char **parity,
84  73          int blocksize)
```

...

```
224 213
225 214  static void * shss_init(struct ec_backend_args *args, void *backend_sohandle)
226 215  {
227     -    static struct shss_descriptor xdesc;
    216 +    struct shss_descriptor *desc = NULL;
228 217
229     -    xdesc.k = args->uargs.k;
230     -    xdesc.m = args->uargs.m;
231     -    xdesc.n = args->uargs.k + args->uargs.m;
232     -    xdesc.w = DEFAULT_W;
    218 +    desc = (struct shss_descriptor *)
    219 +            malloc(sizeof(struct shss_descriptor));
    220 +    if (NULL == desc) {
    221 +        return NULL;
    222 +    }
    223 +
    224 +    desc->k = args->uargs.k;
    225 +    desc->m = args->uargs.m;
    226 +    desc->n = args->uargs.k + args->uargs.m;
    227 +    desc->w = DEFAULT_W;
233 228      args->uargs.w = DEFAULT_W;
234 229
235 230      /* Sample on how to pass extra args to the backend */
236 231      // TODO: Need discussion how to pass extra args.
    232 +    // tentatively we could pass with priv_args2 as the bit_lenght
```

Kevin Greenan

s/bit_lenght/bit_length/

返信 • Create task • 2 hours ago

```
237 233     int *priv = (int *)args->uargs.priv_args2;
238     -   xdesc.aes_bit_length = priv[0]; // AES bit number
    234 +   if(priv != NULL){
    235 +       desc->aes_bit_length = priv[0]; // AES bit number
    236 +   }else{
    237 +       desc->aes_bit_length = 128;
    238 +   }
239 239
240 240     union {
241 241         shss_encode_func encodep;
```

...

```
246 246
247 247     func_handle.vptr = NULL;
248 248     func_handle.vptr = dlsym(backend_sohandle, "ssencode");
249     -   xdesc.ssencode = func_handle.encodep;
250     -   if (NULL == xdesc.ssencode) {
    249 +   desc->ssencode = func_handle.encodep;
    250 +   if (NULL == desc->ssencode) {
251 251         goto error;
252 252     }
253 253
254 254     func_handle.vptr = NULL;
255 255     func_handle.vptr = dlsym(backend_sohandle, "ssdecode");
256     -   xdesc.ssdecode = func_handle.decodep;
257     -   if (NULL == xdesc.ssdecode) {
    256 +   desc->ssdecode = func_handle.decodep;
    257 +   if (NULL == desc->ssdecode) {
258 258         goto error;
259 259     }
260 260
261 261     func_handle.vptr = NULL;
262 262     func_handle.vptr = dlsym(backend_sohandle, "ssreconst");
263     -   xdesc.ssreconst = func_handle.reconp;
264     -   if (NULL == xdesc.ssreconst) {
    263 +   desc->ssreconst = func_handle.reconp;
    264 +   if (NULL == desc->ssreconst) {
265 265         goto error;
266 266     }
267 267
268     -   return (void *)&xdesc;
    268 +   return desc;
269 269
270 270 error:
    271 +   free(desc);
    272 +
271 273     return NULL;
272 274 }
273 275
274 276 static int shss_exit(void *desc)
275 277 {
    278 +   if (desc != NULL) {
    279 +       free(desc);
    280 +   }
276 281     return 0;
277 282 }
278 283
    284 +static bool shss_is_compatible_with(uint32_t version) {
    285 +   return version == backend_shss.ec_backend_version;
    286 +}
    287 +
279 288 struct ec_backend_op_stubs shss_op_stubs = {
280 289     .INIT                       = shss_init,
281 290     .EXIT                       = shss_exit,
```

...

```
284 293     .FRAGSNEEDED                = shss_fragments_needed,
285 294     .RECONSTRUCT                = shss_reconstruct,
286 295     .ELEMENTSIZE                = shss_element_size,
    296 +   .ISCOMPATIBLEWITH           = shss_is_compatible_with,
287 297 };
288 298
289 299 struct ec_backend_common backend_shss = {
```

...

src/erasurecode.c

差分を横に並べて表示 ファイルを表示 編集

...

```
333 333                                 char **encoded_parity)
334 334 {
```

```
335 335     int i, k, m;
    336 +
336 337     ec_backend_t instance = liberasurecode_backend_instance_get_by_desc(desc);
337 338     if (NULL == instance) {
338 339         return -EBACKENDNOTAVAIL;
```

...

```
353 354         for (i = 0; i < m; i++) {
354 355             free(encoded_parity[i]);
355 356         }
356     -
357 357         free(encoded_parity);
358 358     }
359 359
```

...

```
441 441     ret = prepare_fragments_for_encode(instance, k, m, orig_data, orig_data_size,
442 442                                        *encoded_data, *encoded_parity, &blocksize);
443 443     if (ret < 0) {
    444 +       // ensure encoded_data/parity point the head of fragment_ptr
    445 +       get_fragment_ptr_array_from_data(*encoded_data, *encoded_data, k);
    446 +       get_fragment_ptr_array_from_data(*encoded_parity, *encoded_parity, m);
444 447         goto out;
445 448     }
446 449
```

...

```
448 451     ret = instance->common.ops->encode(instance->desc.backend_desc,
449 452                                        *encoded_data, *encoded_parity, blocksize);
450 453     if (ret < 0) {
    454 +       // ensure encoded_data/parity point the head of fragment_ptr
    455 +       get_fragment_ptr_array_from_data(*encoded_data, *encoded_data, k);
    456 +       get_fragment_ptr_array_from_data(*encoded_parity, *encoded_parity, m);
451 457         goto out;
452 458     }
453 459
```

...

```
455 461                                            *encoded_data, *encoded_parity);
456 462
457 463     *fragment_len = get_fragment_size((*encoded_data)[0]);
    464 +
458 465 out:
459 466     if (ret) {
460 467         /* Cleanup the allocations we have done */
```

...

```
625 632      *
626 633      */
627 634     ret = prepare_fragments_for_decode(k, m,
628     -                                   data, parity, missing_idxs,
    635 +                                   data, parity, missing_idxs,
629 636                                     &orig_data_size, &blocksize,
630 637                                     fragment_len, &realloc_bm);
631 638     if (ret < 0) {
```

...

```
1111 1118     return liberasurecode_get_aligned_data_size(desc, 1);
1112 1119 }
1113 1120
     1121 +int liberasurecode_get_fragment_size(int desc, int data_len)
     1122 +{
     1123 +    ec_backend_t instance = liberasurecode_backend_instance_get_by_desc(desc);
     1124 +    // TODO: Create a common function to calculate fragment size also for preprocessing
     1125 +    int aligned_data_len = get_aligned_data_size(instance, data_len);
     1126 +    int size = (aligned_data_len / instance->args.uargs.k) + instance->common.metadata_adder;
     1127 +
     1128 +    return size;
     1129 +}
     1130 +
1114 1131 /* ==~=*=~==~=*=~==~=*=~==~=*=~==~=* misc *=~==~=*=~==~=*=~==~=*=~==~=*=~== */
1115 1132
1116 1133 #if 0
```

...

src/erasurecode_helpers.c

差分を横に並べて表示 ファイルを表示 編集

...

```
163 163  */
164 164 uint64_t get_fragment_size(char *buf)
165 165 {
166     -    fragment_header_t *header = NULL;
167 166
168 167     if (NULL == buf)
```

```
169  168         return -1;
170  169
171       -    header = (fragment_header_t *) buf;
172       -    return (header->meta.size + sizeof(fragment_header_t));
     170  +    return get_fragment_buffer_size(buf) + sizeof(fragment_header_t);
173  171   }
174  172
175  173  /**
```

...

```
188  186      int alignment_multiple;
189  187      int aligned_size = 0;
190  188
191       -    /* Account for any custom metadata the backend wants to add in data_len */
192       -    data_len += instance->common.metadata_adder;
193       -
194  189      /*
195  190       * For Cauchy reed-solomon align to k*word_size*packet_size
196  191       * For Vandermonde reed-solomon and flat-XOR, align to k*word_size
```

...

```
232  227      return num;
233  228  }
234  229
     230  +int get_fragment_ptr_array_from_data(char **frag_array, char **data,
     231  +                                     int num_data)
     232  +{
     233  +    int i = 0, num = 0;
     234  +    for (i = 0; i < num_data; i++) {
     235  +        char *data_ptr = frag_array[i];
     236  +        if (data_ptr == NULL) {
     237  +            data[i] = NULL;
     238  +            continue;
     239  +        }
     240  +        data[i] = get_fragment_ptr_from_data(data_ptr);
     241  +        num++;
     242  +    }
     243  +    return num;
     244  +}
     245  +
235  246  char *get_fragment_ptr_from_data_novalidate(char *buf)
236  247  {
237  248      buf -= sizeof(fragment_header_t);
```

...

```
313  324      return header->meta.size;
314  325  }
315  326
     327  +int set_fragment_adder_size(char *buf, int size)
     328  +{
     329  +    fragment_header_t *header = (fragment_header_t *) buf;
     330  +
     331  +    assert(NULL != header);
     332  +    if (header->magic != LIBERASURECODE_FRAG_HEADER_MAGIC) {
     333  +        log_error("Invalid fragment header (size check)!");
```

Kevin Greenan

(set addr size)

返信 • Create task • 2 hours ago

```
     334  +        return -1;
     335  +    }
     336  +
     337  +    header->meta.frag_adder_size = size;
     338  +
     339  +    return 0;
     340  +}
     341  +
     342  +int get_fragment_adder_size(char *buf)
     343  +{
     344  +    fragment_header_t *header = (fragment_header_t *) buf;
     345  +
     346  +    assert(NULL != header);
     347  +    if (header->magic != LIBERASURECODE_FRAG_HEADER_MAGIC) {
     348  +        log_error("Invalid fragment header (get size)!");
```

Kevin Greenan

(get addr size)

返信 • Create task • 2 hours ago

```
     349  +        return -1;
     350  +    }
     351  +
```

```
+    return header->meta.frag_adder_size;
+}
+
+int get_fragment_buffer_size(char *buf)
+{
+    fragment_header_t *header = (fragment_header_t *) buf;
+
+    assert(NULL != header);
+    if (header->magic != LIBERASURECODE_FRAG_HEADER_MAGIC) {
+        log_error("Invalid fragment header (get size)!");
+        return -1;
+    }
+
+    return header->meta.size + header->meta.frag_adder_size;
+}
+
 int set_orig_data_size(char *buf, int orig_data_size)
 {
     fragment_header_t *header = (fragment_header_t *) buf;
```

...

src/erasurecode_postprocessing.c 差分を横に並べて表示 ファイルを表示 編集

...

```
     set_fragment_payload_size(fragment, blocksize);
     set_backend_id(fragment, be->common.id);
     set_backend_version(fragment, be->common.ec_backend_version);
-
+    set_fragment_adder_size(fragment, be->common.metadata_adder);
+
     if (add_chksum) {
         set_checksum(ct, fragment, blocksize);
     }
```

...

src/erasurecode_preprocessing.c 差分を横に並べて表示 ファイルを表示 編集

...

```
     int i, ret = 0;
     int data_len;           /* data len to write to fragment headers */
     int aligned_data_len;   /* EC algorithm compatible data length */
-    int bsize = 0;
+    int buffer_size, bsize = 0;

     /* Calculate data sizes, aligned_data_len guaranteed to be divisible by k*/
     data_len = orig_data_size;
     aligned_data_len = get_aligned_data_size(instance, orig_data_size);
     *blocksize = bsize = (aligned_data_len / k);
```

Kevin Greenan

bsize should be something more descriptive, like payload_size

返信 • Create task • 2 hours ago

```
+    buffer_size = bsize + instance->common.metadata_adder;

     for (i = 0; i < k; i++) {
-        int payload_size = data_len > bsize ? bsize : data_len;
-        char *fragment = (char *) alloc_fragment_buffer(bsize);
+        int copy_size = data_len > bsize ? bsize : data_len;
+        char *fragment = (char *) alloc_fragment_buffer(buffer_size);
         if (NULL == fragment) {
             ret = -ENOMEM;
             goto out_error;
```

...

```
         encoded_data[i] = get_data_ptr_from_fragment(fragment);

         if (data_len > 0) {
-            memcpy(encoded_data[i], orig_data, payload_size);
+            memcpy(encoded_data[i], orig_data, copy_size);
         }

-        orig_data += payload_size;
-        data_len -= payload_size;
+        orig_data += copy_size;
+        data_len -= copy_size;
     }
```

```
 69   70      for (i = 0; i < m; i++) {
 70       -        char *fragment = (char *) alloc_fragment_buffer(bsize);
      71   +        char *fragment = (char *) alloc_fragment_buffer(buffer_size);
 71   72           if (NULL == fragment) {
 72   73               ret = -ENOMEM;
 73   74               goto out_error;
```

...

```
245  246               num_missing++;
246  247           }
247  248       }
248      -
249      -    return (num_missing > m) ? 1 : 0;
     249  +    // TODO: In general, it is possible to reconstruct one or more fragments
     250  +    // when more than m fragments are missing (e.g. flat XOR codes)
     251  +    return (num_missing > m) ? -1 : 0;
250  252  }
251  253
252  254  int fragments_to_string(int k, int m,
```

...

test/liberasurecode_test.c

差分を横に並べて表示 ファイルを表示 編集

...

```
30   30  #include <stdbool.h>
31   31  #include "erasurecode.h"
32   32  #include "erasurecode_helpers.h"
     33 +#include "erasurecode_preprocessing.h"
33   34  #include "erasurecode_backend.h"
34   35  #include "alg_sig.h"
35   36  #define NULL_BACKEND "null"
```

...

```
212  213      return num_frags;
213  214  }
214  215
     216 +static int encode_failure_stub(void *desc, char **data,
     217 +                                char **parity, int blocksize)
     218 +{
     219 +    return -1;
     220 +}
     221 +
215  222  static void validate_fragment_checksum(struct ec_args *args,
216  223      fragment_metadata_t *metadata, char *fragment_data)
217  224  {
```

...

```
279  286      char *orig_data = create_buffer(orig_data_size, 'x');
280  287      char **encoded_data = NULL, **encoded_parity = NULL;
281  288      uint64_t encoded_fragment_len = 0;
     289 +    ec_backend_t instance = NULL;
     290 +    int (*orig_encode_func)(void *, char **, char **, int);
282  291
283  292      assert(orig_data != NULL);
284  293      rc = liberasurecode_encode(desc, orig_data, orig_data_size,
```

...

```
307  316      rc = liberasurecode_encode(desc, orig_data, orig_data_size,
308  317              &encoded_data, &encoded_parity, NULL);
309  318      assert(rc < 0);
     319 +
     320 +    instance = liberasurecode_backend_instance_get_by_desc(desc);
     321 +    orig_encode_func = instance->common.ops->encode;
     322 +    instance->common.ops->encode = encode_failure_stub;
     323 +    rc = liberasurecode_encode(desc, orig_data, orig_data_size,
     324 +            &encoded_data, &encoded_parity, &encoded_fragment_len);
     325 +    assert(rc < 0);
     326 +    instance->common.ops->encode = orig_encode_func;
     327 +
310  328      free(orig_data);
311  329  }
312  330
```

...

```
501  519
502  520  }
503  521
     522 +static void test_get_fragment_partition()
     523 +{
     524 +    int i;
     525 +    int rc = 0;
     526 +    int desc = -1;
     527 +    int orig_data_size = 1024 * 1024;
```

```
+    char *orig_data = create_buffer(orig_data_size, 'x');
+    char **encoded_data = NULL, **encoded_parity = NULL;
+    uint64_t encoded_fragment_len = 0;
+    int num_avail_frags = -1;
+    char **avail_frags = NULL;
+    int *skips = create_skips_array(&null_args, -1);
+    int *missing;
+
+    desc = liberasurecode_instance_create(EC_BACKEND_NULL, &null_args);
+    assert(desc > 0);
+    rc = liberasurecode_encode(desc, orig_data, orig_data_size,
+            &encoded_data, &encoded_parity, &encoded_fragment_len);
+    assert(0 == rc);
+
+    missing = alloc_and_set_buffer(sizeof(char*) * null_args.k, -1);
+
+    for(i = 0; i < null_args.m; i++) skips[i] = 1;
+    num_avail_frags = create_frags_array(&avail_frags, encoded_data,
+                                         encoded_parity, &null_args, skips);
+
+    rc = get_fragment_partition(null_args.k, null_args.m, avail_frags, num_avail_frags,
+                                encoded_data, encoded_parity, missing);
+    assert(0 == rc);
+
+    for(i = 0; i < null_args.m; i++) assert(missing[i] == i);
+    assert(missing[++i] == -1);
+
+    free(missing);
+
+    for(i = 0; i < null_args.m + 1; i++) skips[i] = 1;
+    num_avail_frags = create_frags_array(&avail_frags, encoded_data,
+                                         encoded_parity, &null_args, skips);
+
+    missing = alloc_and_set_buffer(sizeof(char*) * null_args.k, -1);
+    rc = get_fragment_partition(null_args.k, null_args.m, avail_frags, num_avail_frags,
+                                encoded_data, encoded_parity, missing);
+
+    for(i = 0; i < null_args.m + 1; i++) assert(missing[i] == i);
+    assert(missing[++i] == -1);
+
+    assert(rc < 0);
+
+    free(missing);
+    free(skips);
+    liberasurecode_encode_cleanup(desc, encoded_data, encoded_parity);
+    free(avail_frags);
+    free(orig_data);
+}
+
 static void encode_decode_test_impl(const ec_backend_id_t be_id,
                                     struct ec_args *args,
                                     int *skip)
```

...

```
     int num_avail_frags = 0;
     char *orig_data_ptr = NULL;
     int remaining = 0;
+    ec_backend_t be = NULL;

     desc = liberasurecode_instance_create(be_id, args);
+    be = liberasurecode_backend_instance_get_by_desc(desc);
+
     if (-EBACKENDNOTAVAIL == desc) {
         fprintf (stderr, "Backend library not available!\n");
         return;
```

...

```
         assert(header != NULL);
         fragment_metadata_t metadata = header->meta;
         assert(metadata.idx == i);
-        assert(metadata.size == encoded_fragment_len - frag_header_size);
+        assert(metadata.size == encoded_fragment_len - frag_header_size - be->common.metadata_adder);
         assert(metadata.orig_data_size == orig_data_size);
         char *data_ptr = frag + frag_header_size;
         int cmp_size = remaining >= metadata.size ? metadata.size : remaining;
-        assert(memcmp(data_ptr, orig_data_ptr, cmp_size) == 0);
+        // shss doesn't keep original data on data fragments
+        if (be_id != 5) {
+            assert(memcmp(data_ptr, orig_data_ptr, cmp_size) == 0);
+        }
         remaining -= cmp_size;
         orig_data_ptr += metadata.size;
     }
```

...

```
 553  632     num_avail_frags = create_frags_array(&avail_frags, encoded_data,
 554  633                                          encoded_parity, args, skip);
 555  634     assert(num_avail_frags != -1);
 556       -
 557  635     rc = liberasurecode_decode(desc, avail_frags, num_avail_frags,
 558  636                                encoded_fragment_len, 1,
 559  637                                &decoded_data, &decoded_data_len);
```

...

```
1109 1187         test_fragments_needed_invalid_args,
1110 1188         EC_BACKENDS_MAX, CHKSUM_TYPES_MAX,
1111 1189         .skip = false},
     1190 +    {"test_get_fragment_partition",
     1191 +        test_get_fragment_partition,
     1192 +        EC_BACKENDS_MAX, CHKSUM_TYPES_MAX,
     1193 +        .skip = false},
1112 1194     // NULL backend test
1113 1195     {"create_and_destroy_backend",
1114 1196         test_create_and_destroy_backend,
```

...