heap/stack size optimization

CONTENTS

Xu YangChun Aug/14/2019

Introduction

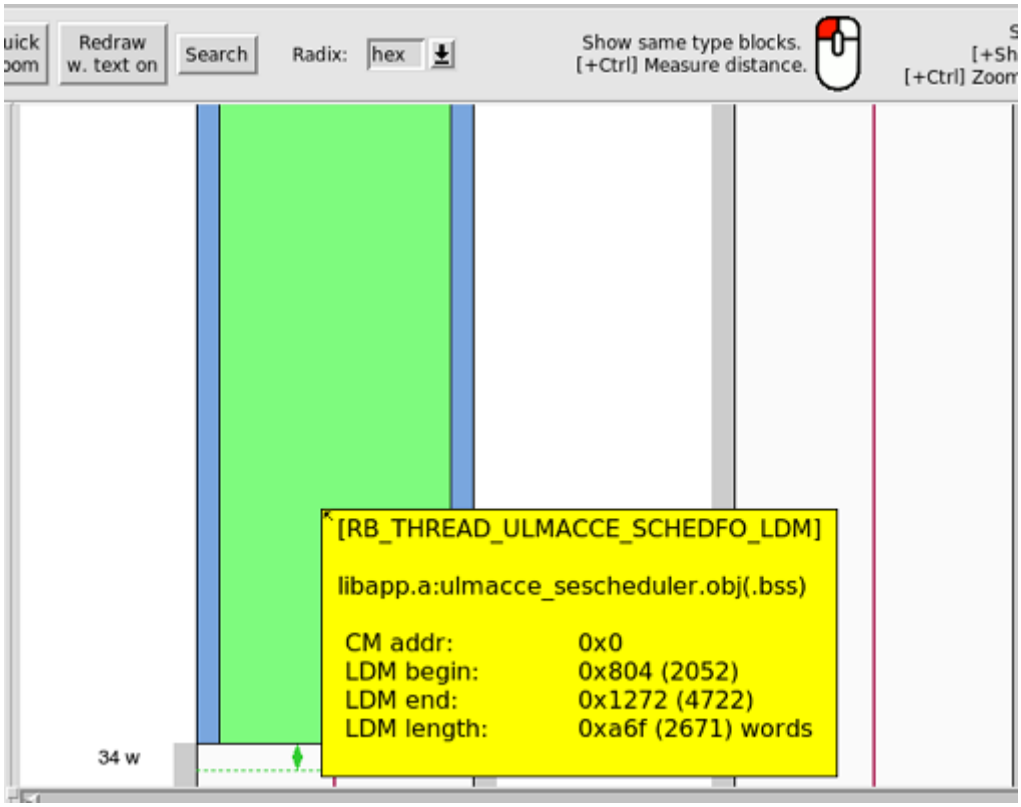
During the maintenance phase, you may met the memory run out issue. If unfortunately, it is legacy code's issue, you have to find out where and how to free some space. This short article take scheduler thread(a FO of MAC layer) as example to describe the methods can be employed.

the issue

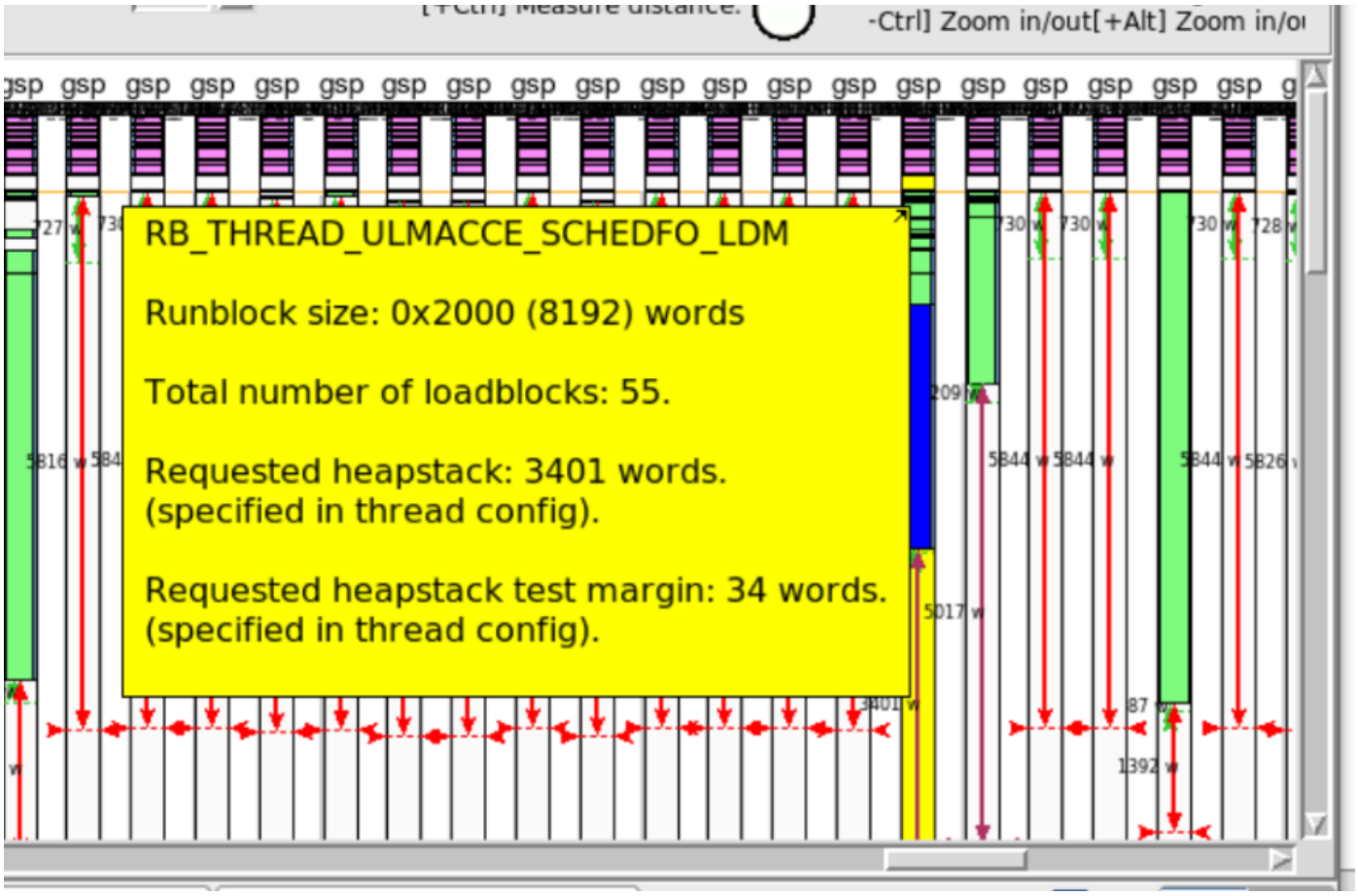
- error reported when building

```
ERROR: List of threads violating the heap & stack limit

Legend:
       +  Requested Heap & stack  violated.
       *  Heap & stack usage test margin violated.

   Used heap Stack|  Requested Heap Stack |Test Margin |Heap Stack limit | Thread Name
 *    3352        |   3383                |     34     |    3349         |test_ULMACCE_SCHEDFO
```

- memory map when unit test



- memory map when deployed (issue solved)


-Ctrl] Zoom in/out[+Alt] Zoom in/o
RB_THREAD_ULMACCE_SCHEDFO_LDM
Runblock size: 0x2000 (8192) words
Total number of loadblocks: 55.
Requested heapstack: 3401 words.
(specified in thread config).
Requested heapstack test margin: 34 words.
(specified in thread config).

global variables refactoring

main approach is reducing its data type's size

```
/* Add global variables that do NOT require initialisation within the following
 * # pragma */
#pragma nostdinit on
ULMACCE_SCHEDFO_foDataS ulSeSched_foData;
ULMACCE_PUSCHTPS_newTxSeDataS
ulSeSched_newSchedData[ELIB_BBBASE_COMMON_MAX_NR_OF_UL_SE_PER_TT_PER_CELL];
UpcDlMacCeFiSePdcchCfmDataS
ulSeSched_pdcchCfmData[ELIB_BBBASE_COMMON_MAX_NR_OF_UL_SE_PER_TT_PER_CELL];
ULMACCE_HARQPROCPOOLPPS_harqProcS
ulSeSched_harqProcData[ELIB_BBBASE_COMMON_MAX_NR_OF_UL_SE_PER_TT_PER_CELL];
ULMACCE_SCHEDFO_internalSeDataS
  ulSeSched_internalSeData[ELIB_BBBASE_COMMON_MAX_NR_OF_UL_SE_PER_TT_PER_CELL];
#pragma nostdinit off
```

packing

```
typedef struct ULMACCE_NEW_newTxSeDataS
{
    U16 maxNrOfSbsForNew;  => 7bits
    S16 averageIslandNrOfSbs;=>7bits
}
```

cutting

```
typedef struct ULMACCE_SCHEDFO_foDataS
{
  ULMACCE_admCtrlDataHeaderS admCtrlData;=>delte
}
when needed:
  ULMACCE_admCtrlDataHeaderS admCtrlData;
  ULMACCE_SHD_PUSCHTPS_getAdmCtrlDataHeader(&foData_p->puschTpsTo, &admCtrlData);
```

Pros & cons

- direct, easy to understand
- performance penalty

- few left to optimize
  e.g
  csiRequestBits actually need 3 bits only
  below U16 var actually , but the change needs lots efforts, especially in test code which use it as U16.

limitation of the compiler Flacc

- don't support c99 fully
- maybe new compiler clang change the situation

example in asm

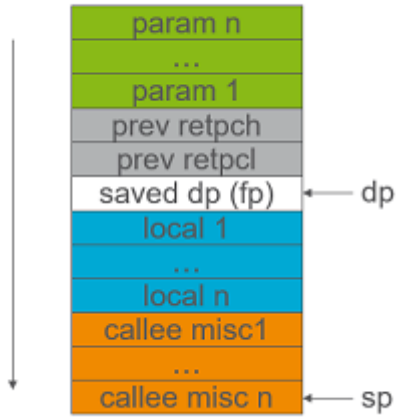
```
struct bits
{
        unsigned short low : 2;
        unsigned short mid: 10;
        signed short : 4;
};
unsigned short i = 0x12;
int main()
{
        struct bits b = {-1};//line 10
        unsigned short xxx;
        xxx = i;
        b.mid = 33;
        xxx = b.mid;
        return 0;
}
/*
        mv          *dp(-2), a0h          // 10 bits.c
        or          49152, a0h            // 10
        mv          a0h, *dp(-2)          // 10
        mv          *dp(-2), a0h          // 10
        and         49167, a0h            // 10
        mv          a0h, *dp(-2)          // 10
        mv          #i, r0                // 12
        mv          *r0, a0h              // 12
        mv          a0h, *dp(-3)          // 12
        mv          *dp(-2), a0h          // 13
        and         49167, a0h            // 13
        addh        528, a0               // 13
        mv          a0h, *dp(-2)          // 13
        mv          *dp(-2), a0h          // 14
        exz         a0, 20, 9, a0h        // 14
        mv          a0h, *dp(-3)          // 14

*/
```

local variables refactoring

stack resue is an edge tool

Flex Stack Frame Layout

- **by caller:**
- in parameters
- saved retpc value (by call instruction)
- **by callee:**
- saved display pointer
- local variables (static)
- callee misc area (dynamic)



what happen when compiled

```c
#include <stdlib.h>
int s;
int main()
{
        {
         int xxx = rand();//line 6
         int yyy = rand();
         int xxx_2 = rand();
         int yyy_2 = rand();
         int xxx_3 = rand();
         int yyy_3 = rand();

         if (xxx>100)
                 s +=(xxx+yyy + xxx_2+yyy_2+xxx_3+yyy_3);
        }
        {
         int xxx = rand();//line 17
         int yyy = rand();
         int xxx_2 = rand();
         int yyy_2 = rand();
         int xxx_3 = rand();
         int yyy_3 = rand();
         if (yyy>300)
                 s -=(xxx+yyy + xxx_2+yyy_2+xxx_3+yyy_3); //line 24
        }
        return s;
}

/*
        callr      #rand                // 6 a.c (*), CurrBundle: 4, CriticalPath: 4
        mv         a0h, *dp(-1)         // 6 (*), CurrBundle: 1, CriticalPath: 1, 1-byte Folded
|       callr      #rand                // 7 (*)
        .rule_off                       // 7
        copy       a0, a5               // 7 (*), CurrBundle: 1, CriticalPath: 1
|       callr      #rand                // 8 (*)
        copy       a0, a6               // 8 (*), CurrBundle: 1, CriticalPath: 1
|       callr      #rand                // 9 (*)
        copy       a0, a7               // 9 (*), CurrBundle: 1, CriticalPath: 1
|       callr      #rand                // 10 (*)
        copy       a0, a4               // 10 (*), CurrBundle: 1, CriticalPath: 1
|       callr      #rand                // 11 (*)
        .rule_on
        mv         *dp(-1), a1h         // (*), 1-byte Folded Reload
        cmp        101, a1h             // 13 (*), CurrBundle: 2, CriticalPath: 2
        if         .a1:ge               // 14
|       addh       a1h, a5              // 14 (*)
|       mv         #s, r0               // 14
        if         .a1:ge               // 14
|       addh       a5h, a6              // 14 (*)
|       mv         *r0, a1h             // 14
        addh       a6h, a7, .a1:ge      // 14 (*)
```

```
        addh       a7h, a4, .a1:ge      // 14 (*)
        addh       a4h, a0, .a1:ge      // 14 (*)
        addh       a1h, a0, .a1:ge      // 14 (*)
        mv         a0h, *r0, .a1:ge     // 14 (*), CurrBundle: 7, CriticalPath: 7
        callr      #rand                // 17 (*), CurrBundle: 1, CriticalPath: 1
        mv         a0h, *dp(-1)         // 17 (*), CurrBundle: 1, CriticalPath: 1, 1-byte Foldec
|       callr      #rand                // 18 (*)
        .rule_off                       // 18
        copy       a0, a5               // 18 (*), CurrBundle: 1, CriticalPath: 1
|       callr      #rand                // 19 (*)
        copy       a0, a6               // 19 (*), CurrBundle: 1, CriticalPath: 1
|       callr      #rand                // 20 (*)
        copy       a0, a7               // 20 (*), CurrBundle: 1, CriticalPath: 1
|       callr      #rand                // 21 (*)
        copy       a0, a4               // 21 (*), CurrBundle: 1, CriticalPath: 1
|       callr      #rand                // 22 (*)
        .rule_on                        // 23
        cmp        301, a5h             // 23 (*)
        brr        #.LBB0_2, .a5:lt     // 23 (*), CurrBundle: 2, CriticalPath: 2

// Procedure estimates:
// Code size: 101 words (28 bits/instr)
// Instructions / cycles: 57 / 39
// Loop weighted cycles: 39
// Stack allocation: 16
*/
```

example

original code

```
void SESCHEDFO_main()//500+ loc
{
  ULMACCE_LAPCSPLIB_traceS trace;
  ULMACCE_SCHEDLIB_traceHeaderS* traceHeader_p;

  traceHeader_p = &trace.header;
  traceHeader_p->cellId = foData_p->commonCellTo.cellInfo.cellId;

  while(expr)
  {
    trace.specificInfo.traceEntryInfo.nrOfSchedulableSes = nrOfSchedulableSes;

    BOOL isNs05seTypeDowngraded = FALSE;
    assignSpectrumSucceeded =
        ULMACCE_SCHEDLIB_assignSpectrumToSe(
            &trace
        );

    if (assignSpectrumSucceeded)
    {
      ULMACCE_PUSCHTPS_ginrWeighterDataS ginrWeightDataS;

    }
  }
}
```

after refactoring

```
void SESCHEDFO_main()//500+ loc
{

  while(expr)
  {
    BOOL isNs05seTypeDowngraded = FALSE;
    {
    ULMACCE_LAPCSPLIB_traceS trace;
    ULMACCE_SCHEDLIB_traceHeaderS* traceHeader_p;
    traceHeader_p = &trace.header;
    traceHeader_p->cellId = foData_p->commonCellTo.cellInfo.cellId;
    trace.specificInfo.traceEntryInfo.nrOfSchedulableSes = nrOfSchedulableSes;

    assignSpectrumSucceeded =
        ULMACCE_SCHEDLIB_assignSpectrumToSe(
            &trace
        );
    }

    if (assignSpectrumSucceeded)
    {
        ULMACCE_PUSCHTPS_ginrWeighterDataS ginrWeightDataS;

    }
  }
}
```

result:
Decreased stack use in test_ULMACCE_SCHEDFO: by 26 words or 1.0%; new stack 2668 vs stack 2694

assume
how much saved = how much resued - how much compiler alredy optimzed

Pros & cons

- indirect & difficult to calculate
- more effort to read code
- no tool to find out the deepest call sequence in stack

data structure refactoring

e.g. any duplicate?

```
typedef struct ULMACCE_SESCHEDFO_seListS
{
  S16 size;
  S16 nrOfActiveElements;
  S16 startIndex;
  S16 iteratorIndex;
  ULMACCE_SESCHED_seDataS* se_p[ULMACCE_PUSCH_MAX_SE_LIST_SIZE];
} ULMACCE_SESCHEDFO_seListS;


typedef struct ULMACCE_SCHEDFO_foDataS
{
  ULMACCE_SCHEDFO_fodS fod;
  ULMACCE_SESCHEDFO_seListS inputNewSeList;
  ULMACCE_SESCHEDFO_seListS scheduledNewSeList;
} ULMACCE_SCHEDFO_foDataS;
```

** in the fod, there is two list: inputNewSeList, scheduledNewSeList.
actually one se can only appear in 1 list, why need 2 space? **

e.g. duplicte index?

```
#pragma nostdinit on
ULMACCE_SCHEDFO_foDataS ulSeSched_foData;
ULMACCE_PUSCHTPS_newTxSeDataS
ulSeSched_newSchedData[ELIB_BBBASE_COMMON_MAX_NR_OF_UL_SE_PER_TT_PER_CELL];
UpcDlMacCeFiSePdcchCfmDataS
ulSeSched_pdcchCfmData[ELIB_BBBASE_COMMON_MAX_NR_OF_UL_SE_PER_TT_PER_CELL];
ULMACCE_HARQPROCPOOLPPS_harqProcS
ulSeSched_harqProcData[ELIB_BBBASE_COMMON_MAX_NR_OF_UL_SE_PER_TT_PER_CELL];
ULMACCE_SCHEDFO_internalSeDataS
  ulSeSched_internalSeData[ELIB_BBBASE_COMMON_MAX_NR_OF_UL_SE_PER_TT_PER_CELL];
#pragma nostdinit off

{//loop
  newSchedData_p = ulSeSched_newSchedData;
  pdcchCfmData_p = ulSeSched_pdcchCfmData;
  harqProcData_p = ulSeSched_harqProcData;
  internalSeData_p = ulSeSched_internalSeData;
  newSchedData_p++;
  //and the other 3 pointer ++
}
```

merging 4 pointer into 1 index

```
static U16 seIndex = 0;

{//loop
  newSchedData_p = &testdata_newSchedData[seIndex];
  harqProcData_p = &testdata_harqProcData[seIndex];
  pdcchCfmData_p = &testdata_pdcchCfmData[seIndex];
  internalSeData_p = &testdata_internalSeData[seIndex];
  seIndex++;
}
```

ULMACCE_PUSCHTPS_newTxSeDataS has much bigger size than reTx, why occupy the same size?
can the array only store the common field which used in scheudling algorith (weight/sort)
the less important field just put a linked list implement via array?

code refactoring

e.g. algorithm