Report Week 11

Achievements:
-Solved issues with multicast packages messing up communication
-Implemented a linked list of unique items on every TCM in order to speed up query processing and unique ID assignment
-Introduced several debugging modes that highlight specific information at specific points in time, in this particular case from timer tick 1000 to 1120 (down below)

```
/* Debugging mode */
#define DEBUG_1 1
#define DEBUG_2 0
#define DEBUG_3 0
#define DEBUG_4 0
#define DEBUG_START 1000
#define DEBUG_END   1120
/* 0 Default information about cores
 * DEBUG_1 Enables information about messages received and sent
 * DEBUG_2 Debug info on the id distribution algorithm
 * DEBUG_3 Information regarding the construction of the linked list dictionary
 * DEBUG_4 Shows timer ticks
 */

#define RECORD_IDS 0
#define RECORD_LINKED_LIST_LENGTHS 1
/* 0 record unique ids
 * 1 record the length of the linked list
 */

//amount of milliseconds the application runs
uint runtime = 10000;
```

The new data structure:

In order to speed up the assignment process, I created a linked list in every core's TCM. This list represents an ID-dictionary:

```
|Entries |ID|      |Dictionary                                  |
|--------|--|      |Entries| ID |Frequency|Index Start|Index End|
|UK      | 1|      |-------|----|---------|-----------|---------|
|UK      | 1| ---> |UK     |   1|        2|          0|        2|
|Germany | 2|      |Germany|   2|        1|          2|        3|
|France  | 3|      |France |   3|        1|          3|        4|
```

Example:

-The rows here store the precise time of an accident down to the single minute.
That means that the number of possible unique entries is 1440

```
Getting profile data
|0%                          50%                         100%|
 ============================================================
2017-12-06 18:41:11 INFO: Time 0:00:00.009541 taken by ProfileDataGatherer
2017-12-06 18:41:11 INFO: |----------------|
2017-12-06 18:41:11 INFO: | Core 0, 0, 2
2017-12-06 18:41:11 INFO: | Rows 3484
2017-12-06 18:41:11 INFO: | List 939
2017-12-06 18:41:11 INFO: | TCM Memory for rows: 6968 bytes
2017-12-06 18:41:11 INFO: | TCM Memory for list: 37560 bytes
2017-12-06 18:41:11 INFO: | TCM Memory total   : 44528 bytes
2017-12-06 18:41:11 INFO: |----------------|
2017-12-06 18:41:11 INFO: | Core 0, 0, 3
2017-12-06 18:41:11 INFO: | Rows 3483
2017-12-06 18:41:11 INFO: | List 938
2017-12-06 18:41:11 INFO: | TCM Memory for rows: 6966 bytes
2017-12-06 18:41:11 INFO: | TCM Memory for list: 37520 bytes
2017-12-06 18:41:11 INFO: | TCM Memory total   : 44486 bytes
2017-12-06 18:41:11 INFO: |----------------|
2017-12-06 18:41:11 INFO: | Core 0, 0, 4
2017-12-06 18:41:11 INFO: | Rows 3483
2017-12-06 18:41:11 INFO: | List 961
2017-12-06 18:41:11 INFO: | TCM Memory for rows: 6966 bytes
2017-12-06 18:41:11 INFO: | TCM Memory for list: 38440 bytes
2017-12-06 18:41:11 INFO: | TCM Memory total   : 45406 bytes
2017-12-06 18:41:11 INFO: |----------------|
```

-The rows here store the pdate of an accident. That means that
there are 365 possible entries

```
 ============================================================
2017-12-06 21:35:11 INFO: Time 0:00:00.010154 taken by ProfileDataGatherer
2017-12-06 21:35:11 INFO: |----------------|
2017-12-06 21:35:11 INFO: | Core 0, 0, 2
2017-12-06 21:35:11 INFO: | Rows 8539
2017-12-06 21:35:11 INFO: | List 122
2017-12-06 21:35:11 INFO: | TCM Memory for rows: 17078 bytes
2017-12-06 21:35:11 INFO: | TCM Memory for list: 4880 bytes
2017-12-06 21:35:11 INFO: | TCM Memory total   : 21958 bytes
2017-12-06 21:35:11 INFO: |----------------|
2017-12-06 21:35:11 INFO: | Core 0, 0, 3
2017-12-06 21:35:11 INFO: | Rows 8539
2017-12-06 21:35:11 INFO: | List 292
2017-12-06 21:35:11 INFO: | TCM Memory for rows: 17078 bytes
2017-12-06 21:35:11 INFO: | TCM Memory for list: 11680 bytes
2017-12-06 21:35:11 INFO: | TCM Memory total   : 28758 bytes
```

The purpose of this output is monitoring memory usage in every core's TCM.
A high number of unique elements translates into a large dictionary (linked list), which can lead to memory overload and crashes if not dealt with properly.
One possible way of mitigating this problem could be implementing a data distribution algorithm on the host machine that limits the amount of different entries on every core.