Huawei Sweden Hackathon 2023

Qualification Phase Task Description Webinar

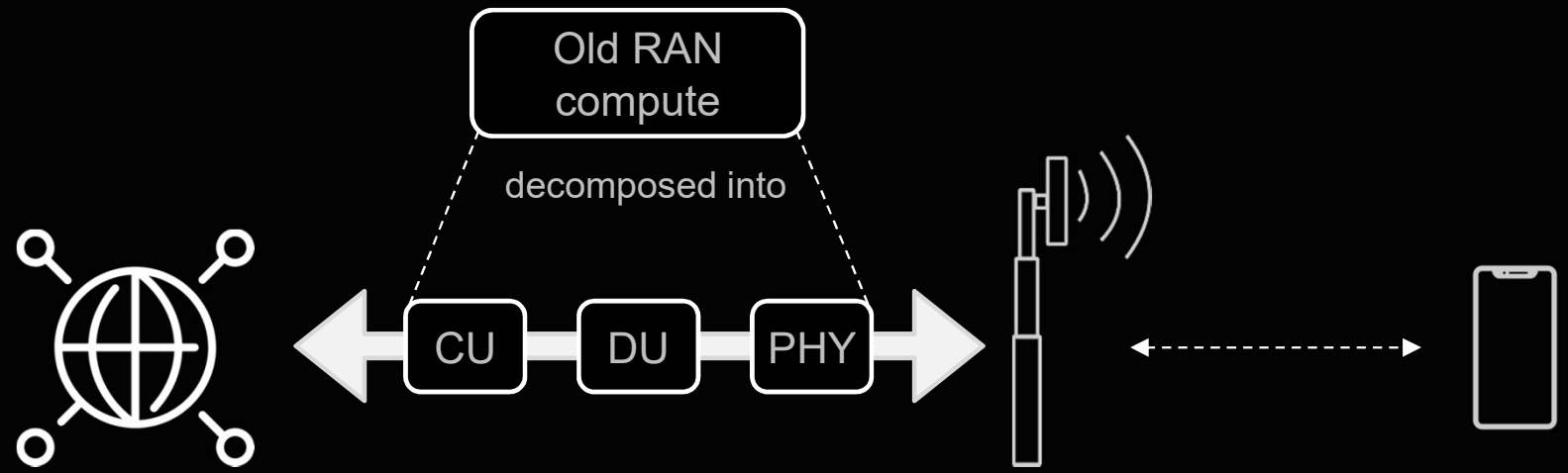
Huawei Sweden Hackathon 2023

EFFICIENTLY DEPLOYING SOFTWARE ON FUTURE NETWORK GENERATIONS

Can you design the most energy and cost efficient
real-time deployment algorithm for 5G and 6G networks?

Motivation – Radio Access Network (RAN)

- New standards open new deployment opportunities by disaggregating computing components
- Cloud computing and container technologies are becoming mature enough to be considered for RAN

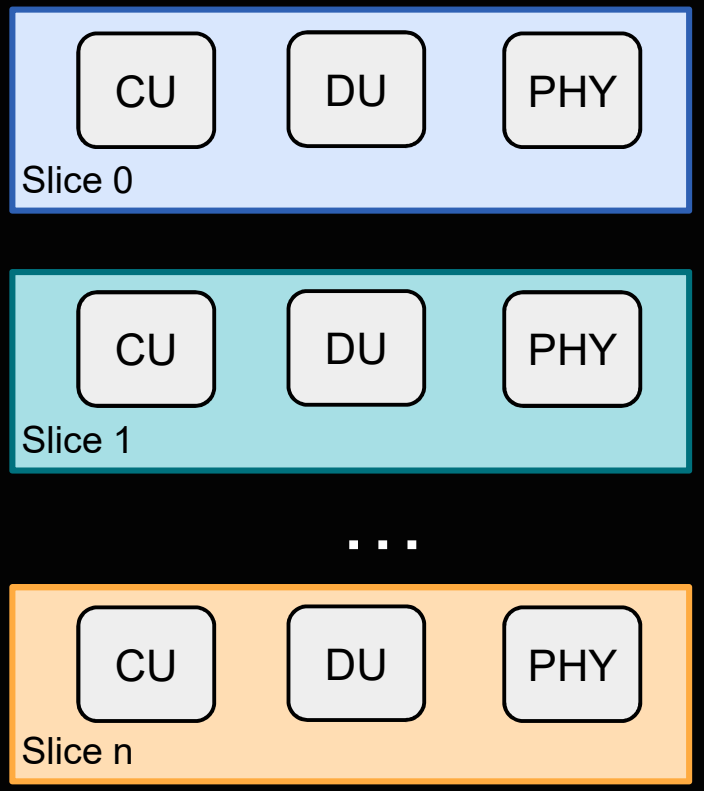


HUAWEI

RAN Application decomposition

- A RAN Application instance is then composed by CU, DU and PHY services. Each service requires different amount of computing resources, and its utilization (amount of each resource) depends on the network traffic experienced in the area to be covered at any given time.

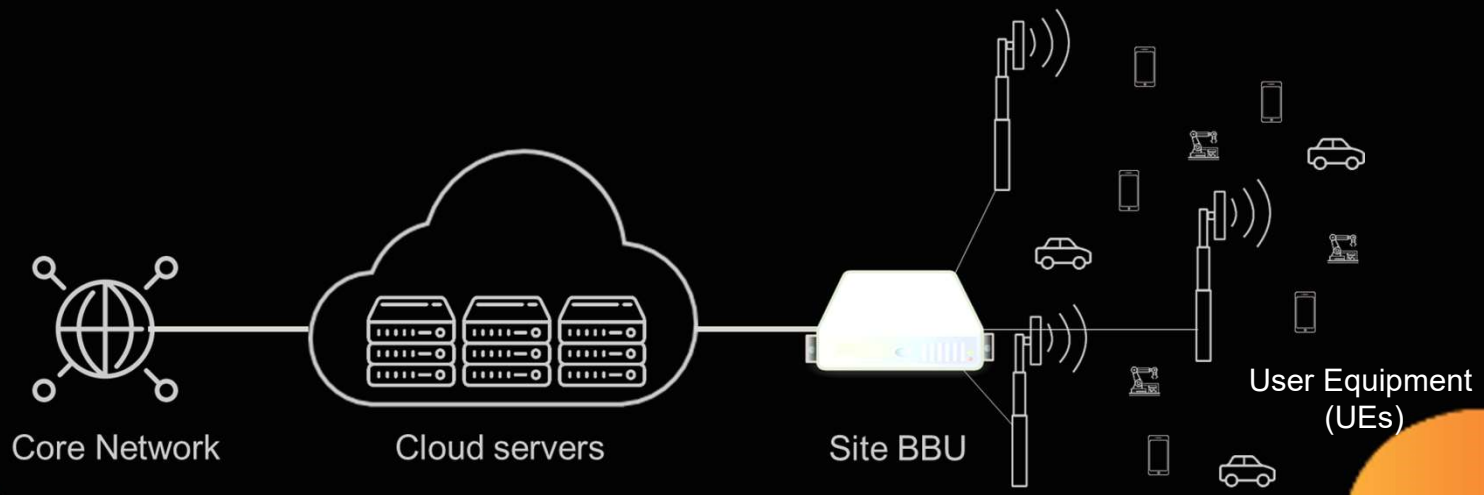
- To provide differentiated services, several RAN application instances (i.e. triplets of CU, DU, PHY) known as slices may be run over the network by the operator (for example to provide service to MVNOs, emergency systems, etc).



HUAWEI

RAN Application decomposition

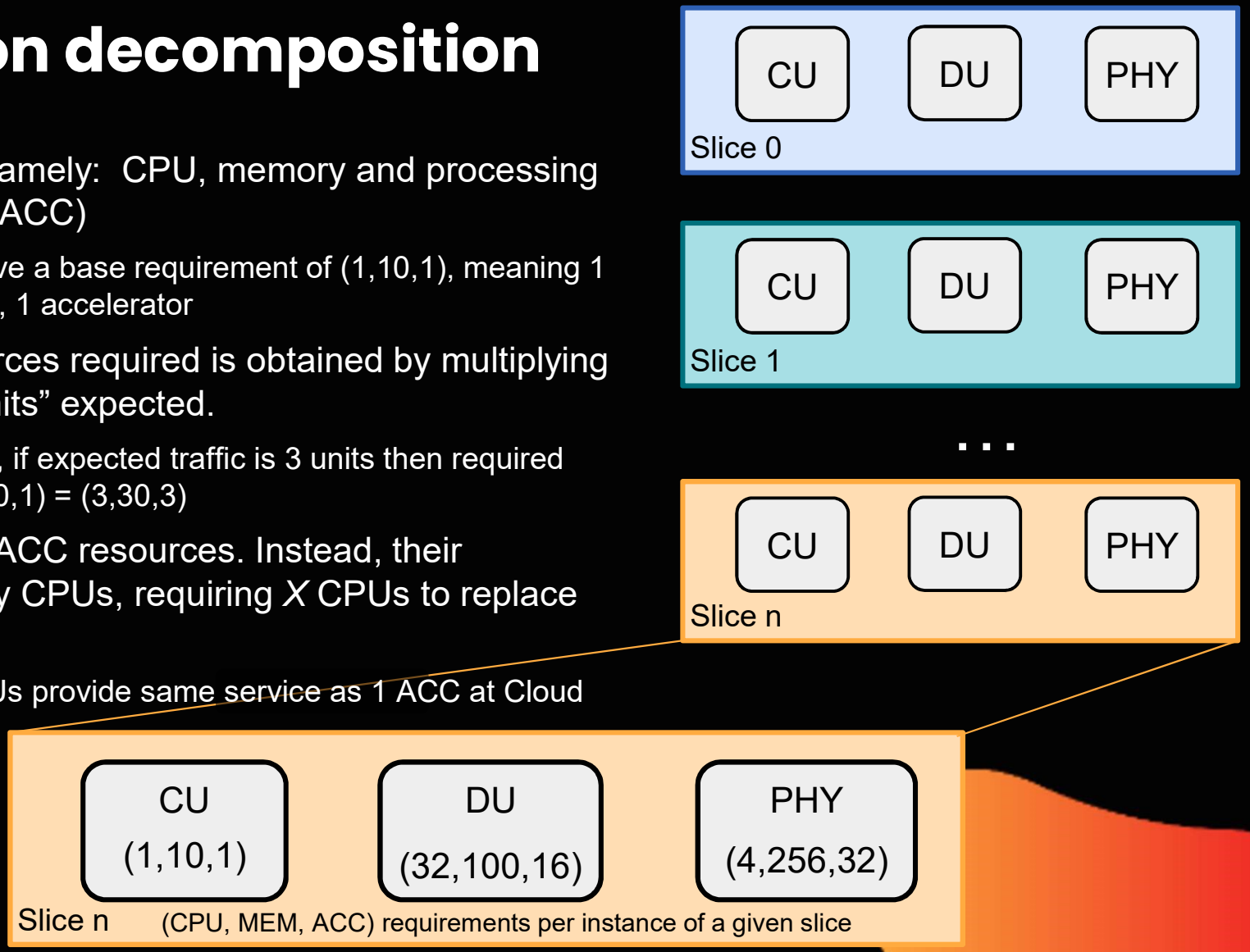
- Suppose an hybrid deployment scenario where an operator can deploy each RAN service instance (each CU, DU, PHY) either at the Base Band Unit (BBU) specialized hardware or at a Cloud server.
- Both Cloud and BBU have an operating cost (OPEX) associated based on their utilization.
  - Cloud servers are charged by the cloud provider exactly according to the resource usage
  - BBU, on the other hand, only runs energy costs, but has limited scaling options



HUAWEI

RAN Application decomposition

- Required resources are, namely: CPU, memory and processing accelerators (CPU, MEM, ACC)
  - Eg: a service might have a base requirement of (1,10,1), meaning 1 CPU, 10 memory units, 1 accelerator
- The total number of resources required is obtained by multiplying this triplet by the “traffic units” expected.
  - Eg: for above example, if expected traffic is 3 units then required resources are: 3 * (1,10,1) = (3,30,3)
- Cloud servers don´t have ACC resources. Instead, their functionality is executed by CPUs, requiring *X* CPUs to replace each ACC.
  - Eg: if X = 4 then 4 CPUs provide same service as 1 ACC at Cloud

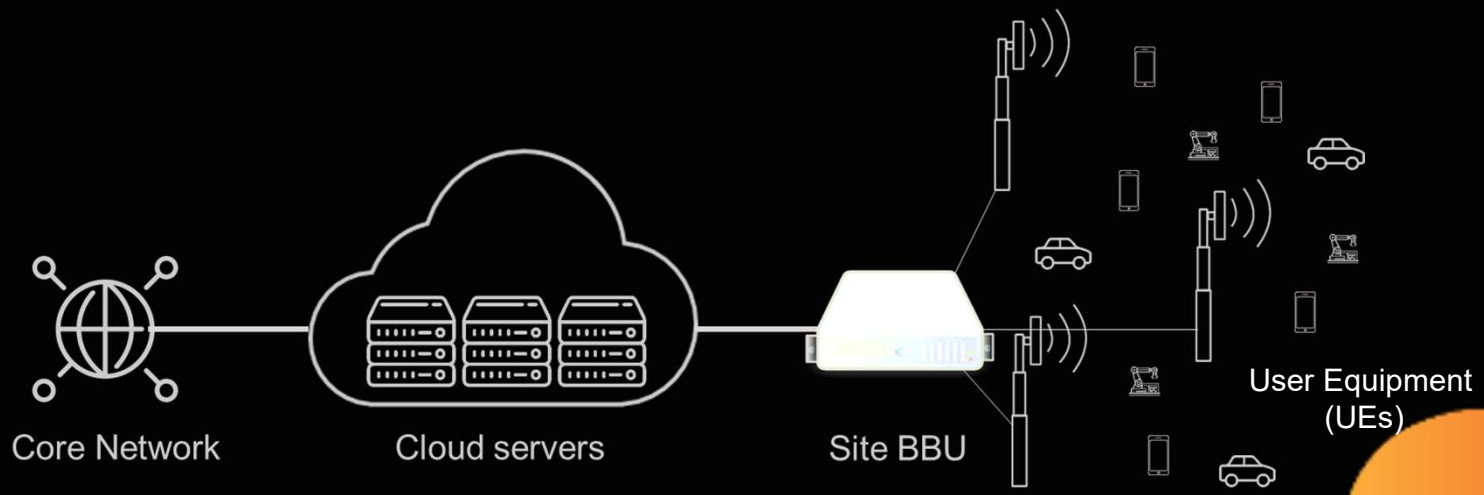


HUAWEI

Competition task

- The problem consists on deploying the CUs, DUs and PHY instances for all slices required by the operator minimizing the operating cost (OPEX), given the expected traffic for each of the slices.

- The problem will span *n* time frames, and you will be given the expected traffic ("*traffic_units*") per time frame *t* and slice *s*



HUAWEI

Deployment rules

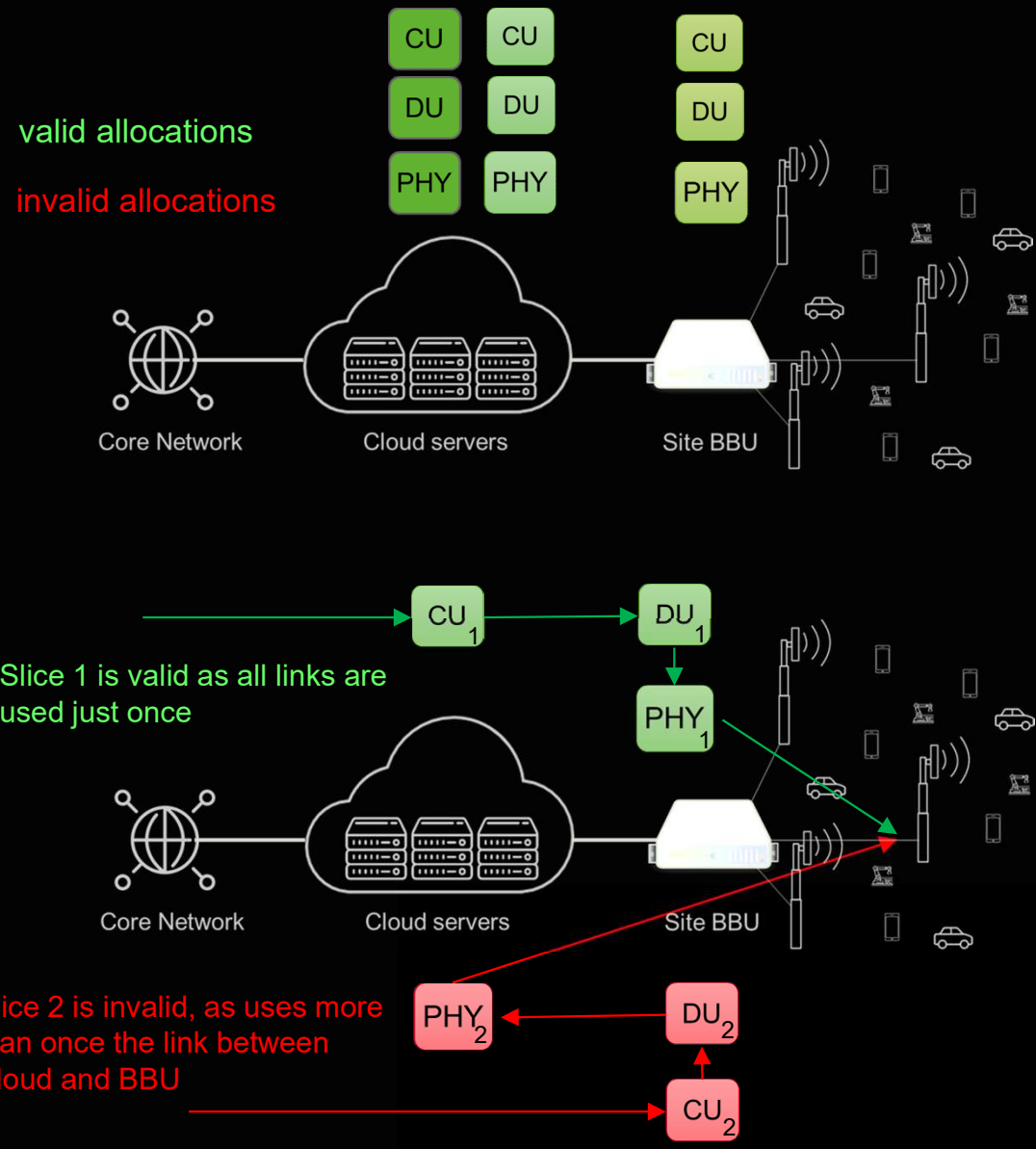
Restriction 1: Data flow inside a slice is

Core Network ↔ CU ↔ DU ↔ PHY ↔ UEs

To go from Core Network to UEs (and opposite) data cannot use more than once any link between Core Network and UEs

Slices are completely independent from each other

- i.e. Restriction 1 only applies to services of the same slice
- e.g. PHY_{slice1} deployed on Cloud only affects the deployment of CU_{slice1} and DU_{slice1} to obtain a valid solution



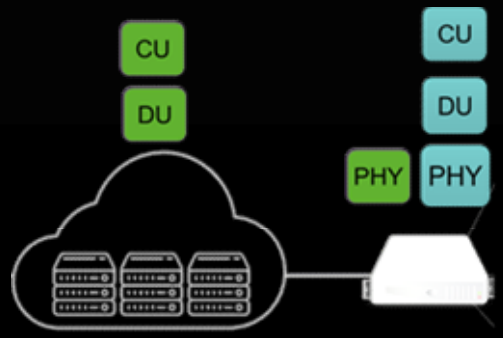
HUAWEI

Deployment rules

- Restriction 2: BBU resources are limited. Therefore, at any time, the sum of resources allocated to BBU should be equal or below the limit.
  - Cloud resources are considered unlimited.

- The operator can deploy, at any time, the APP instances (CU, DU, PHY per slice) over Cloud and BBU as it wishes (respecting Restrictions 1 and 2)
  - In practice, given Restriction 1, only a subset of combinations are possible.
  - Considering a left-to-right order of CU, DU and PHY, and Cloud being coded as ‘C’ and BBU as ‘B’, these are: **CCC**, **CCB**, **CBB** and **BBB**

- Restriction 3: Every service instance (CU, DU, PHY) needs to always have the required computing resources allocated according to the expected traffic



CCB and BBB deployment examples

IO Costs

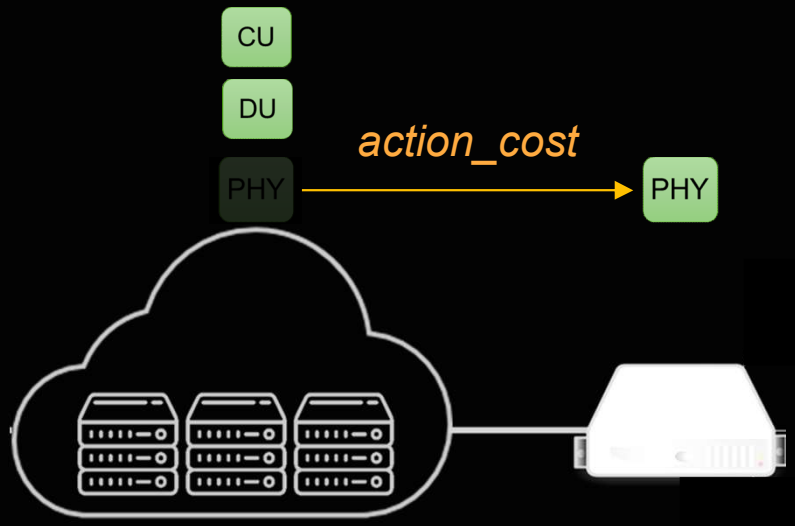
- Each of the mentioned Core Network $\overset{L_A}{\leftrightarrow}$ CU $\overset{L_B}{\leftrightarrow}$ DU $\overset{L_C}{\leftrightarrow}$ PHY $\overset{L_D}{\leftrightarrow}$ UEs links have data being communicated over them. This amount of data is also proportional to the traffic units of demand per slice and time frame.
  - Let the links be L_A, L_B, L_C and L_D

- With the described setup, the operator incurs also in an *IO_cost* in the link between the Cloud and the BBU
  - Eg. If for a given slice and time frame chosen split is BBB, it means everything is executed on the BBU, where we should transfer the L_A link traffic between the Cloud and BBU
  - Example 2: CCB split requires L_C link traffic be transferred between Cloud and BBU

HUAWEI

Action Cost

- Finally, relocating one APP instance from BBU to Cloud or opposite has an *action_cost*
- Initial allocation incurs no cost. After that, for any service of any slice, every time its allocation changes from B to C or C to B, the action cost needs to be added to the total OPEX cost.
  - I.e. for a slice to change from CCC to BBB incurs 3 x *action_cost*



HUAWEI

BBU Resources

- As mentioned, the BBU resources are limited, and can only be provided as fixed sets. Operating cost depends on the number of BBU sets used.
  - You can imagine this as BBU be composed of B boards with (C CPUs; M Memories; A Accelerators; Cost)
  - All sets in the BBU are equally equipped and have the same cost

- The number of sets used at time *t* is determined by the resource type requiring a higher number of sets

$$BBU\ sets = \max\left(\left\lceil\frac{CPU\ allocated}{CPU\ per\ set}\right\rceil, \left\lceil\frac{MEM\ allocated}{MEM\ per\ set}\right\rceil, \left\lceil\frac{ACC\ allocated}{ACC\ per\ set}\right\rceil\right)$$

  - E.g. each set is (10; 100; 1; 10) and at time T, the sum of your BBU allocated resources is (31; 245; 2)
    - Then the number of sets required is 4, driven by your CPU allocation as

      $\max(\lceil 31/10 \rceil, \lceil 245/100 \rceil, \lceil 2/1 \rceil) = \lceil 31/10 \rceil = 4$ BBU_sets
    - And the cost is 40 (10 *cost_per_set* * 4 *BBU_sets*)
    - If you had less then 4 sets available in this test case, the solution would be invalid

HUAWEI

Goal

- Your goal is to provide an allocation per slice and time frame that minimizes the operating cost (OPEX) for each test. This is, to minimize:

$$OPEX = \sum_t CloudCost_t + BBUCost_t + IOCost_t + ActionCost_t$$

- Your will receive a file describing each test case, and you should generate a file for each test case with your solution using the format described in the following slides.

Cost calculations formalized

$$CloudCost_t = \sum_s CPURequired_s * Traffic_{t,s} * CPUCost + MEMRequired_s * Traffic_{t,s} * MEMCost$$

$$\text{BBU}Cost_t = BBUSetsRequired_t * BBUSetCost$$

$$IOCost_t = \sum_s IOCost_{LinkUsed_{t,s}} * Traffic_{t,s}$$

$$ActionCost_t = \sum_s ServiceRellocations_t * ActionCost$$

Remember: Cloud does not have ACC and its functionality is replaced by *X* CPUs

HUAWEI

Ranking

- You will get a score given as the ratio between a given *baseline_cost* and your OPEX for each test case:

$$score = \max(0, \frac{baseline\ cost}{OPEX} - 1)$$

  - If your test case OPEX is worst (higher) or equal than the baseline, or solution is invalid, you get a score of 0
  - If your test case OPEX is better (lower) than the baseline, you get a positive score

- Ranking will be determined by the sum of scores for all test cases

Rules regarding solutions

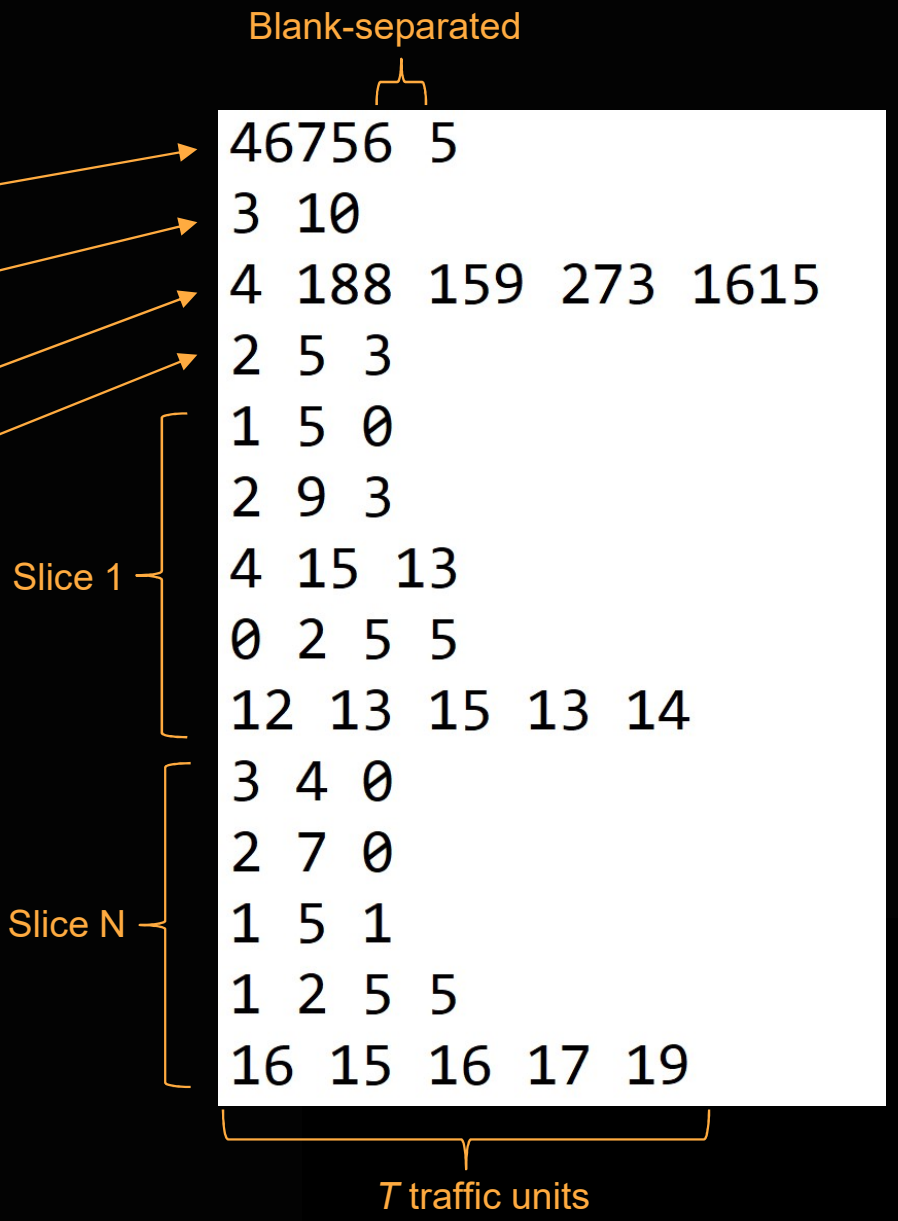
- Restriction 4: Your solution should be general and applicable to any potential test case
  - I.E. it is not acceptable to have a “switch” for test_case_1, test_case_2, … test_case_n or equivalent
- You can of course have different strategies based on the characteristics of the proposed scenario, but not to base the solution on the previous knowledge of the test case itself
  - I.E. it is acceptable to evaluate the test case main characteristics and apply different strategies accordingly
- Restriction 5: your solving time per test case should be below 1 second, otherwise it will be invalid
  - This will be tested on a 4vCPU (x86 instruction set) virtual machine running Ubuntu 22.04 64bit
  - Input file parsing and output time does is not considered as part of solving time

Input for the test cases

- *baseline_cost, action_cost*
- Cloud Costs : *CPU_cost, MEM_cost*
- BBU Profile : *B, CPU, MEM, ACC, cost_per_set*
- # of slices, time horizon, CPU:ACC ratio: *N, T, X*
- The following will be repeated for each slice *s* (*N* repeats)
    - CU resource unit: cpu, mem, acc
    - DU resource unit: cpu, mem, acc
    - PHY resource unit: cpu, mem, acc
    - IO cost: L_A, L_B, L_C, L_D
    - T traffic units: u1, u2, … uT

Input for the test cases

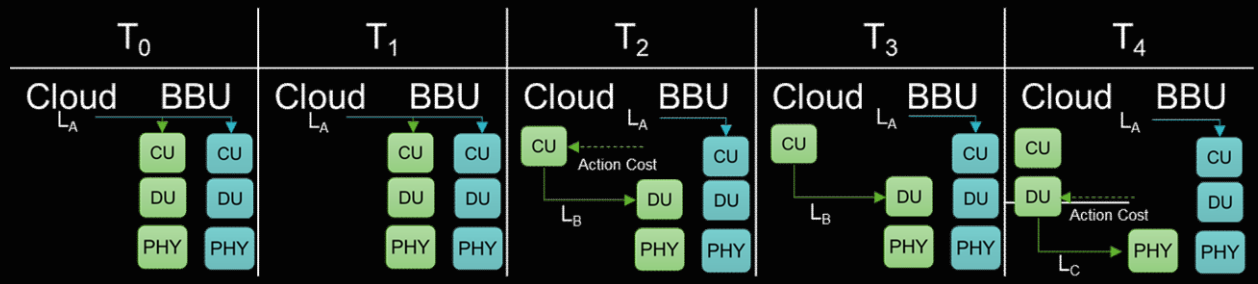
- *baseline_cost, action_cost*
- Cloud Costs : *CPU_cost, MEM_cost*
- BBU Profile : *B, CPU, MEM, ACC, cost_per_set*
- # of slices, time horizon, CPU:ACC ratio: *N, T, X*
- The following will be repeated for each slice *s* (*N* repeats)
  - CU resource unit: cpu, mem, acc
  - DU resource unit: cpu, mem, acc
  - PHY resource unit: cpu, mem, acc
  - IO cost: L_A, L_B, L_C, L_D
  - T traffic units: u1, u2, … uT



Output for the test cases

- The following is repeated for each slice
  - T strings(CCC, CCB, CBB, BBB), one for each timestep: *split(1) split(2) … split(T)*
- T integers denoting Cloud cost at each time step: *cost(1) cost(2) … cost(T)*
- T integers denoting BBU cost at each time step: *cost(1) cost(2) … cost(T)*
- T integers denoting IO cost at each time step: *cost(1) cost(2) … cost(T)*
- *OPEX*
- Score: *max(0, Baseline_Cost / OPEX – 1)*
- Execution time in ms
  - Remember, your solution time per test case must to be <1 second, otherwise it will be invalid

Output



- *Solution matrix*
- *Cloud costs*
- *BBU costs*
- *I/O costs*
- *OPEX*
- *Score*
- *Execution time(ms)*

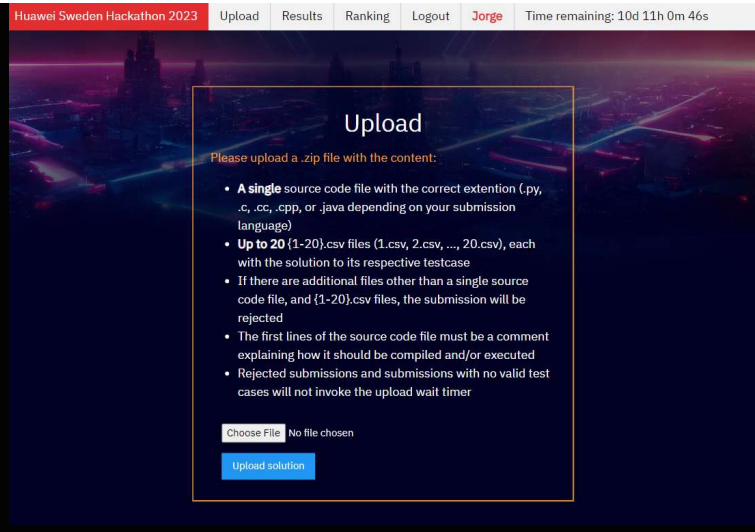
T

N

```
BBB BBB CBB CBB CCB
BBB BBB BBB BBB BBB
0 0 795 689 2464
6460 6460 6460 6460 6460
16 15 46 43 89
36467
0.28214550141223 58
139
```

Blank-separated

Submission process



- Submissions will be accepted at

  compete.huaweihackathon2023.se/qualify

- A submission consists of a .zip file with:
  - A single source code file, implementing your solution.

    Important: add as a comment at the beginning of the file the instructions to run (and compile) your code
  - One or more .csv files, containing your output for each test case

- After a valid submission you will have to wait for 30 seconds to upload a new submission
- Under Results tab you will find a summary of your submissions and a detail for each test case
- Ranking is based on your best submission

HUAWEI

Accepted Languages and Libraries

- You can program your solution using C, C++, Java or Python.
- Accepted libraries are those that come with the standard installation of the toolchains for each language.
- Solutions will be checked using the following toolchains / versions on a Ubuntu 22.04 64bit x86 server:
  - C (up to C17 standard) : gcc 12 (including –lpthreads –lm)
  - C++ (up to C++20 standard) : gcc 12 (including –lpthreads –lm)
  - Java : openjdk-19-jdk
  - Python 3.11

Remember to cite (as a comment) any 3rd party code you use in your solution.

HUAWEI

GOOD LUCK!

Organized BY: