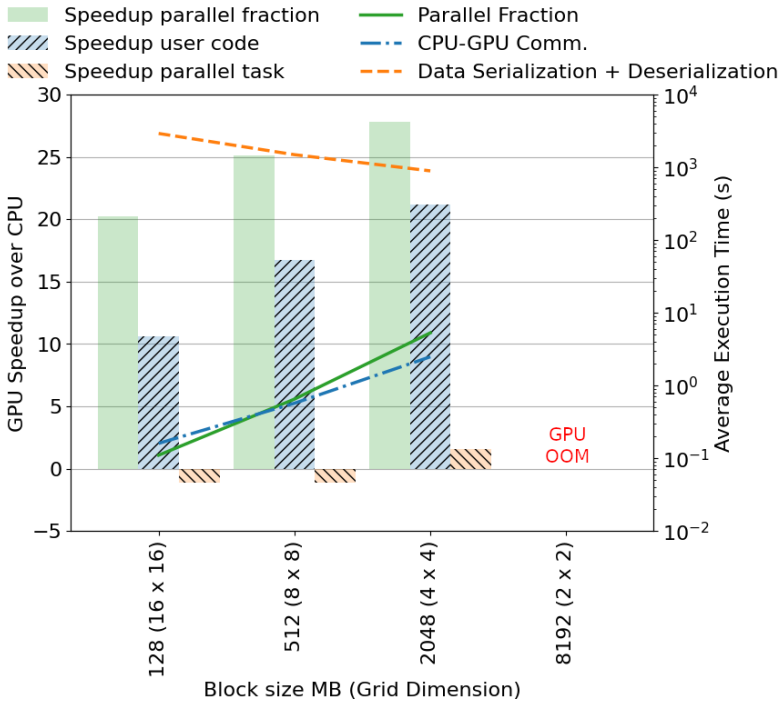
Supplementary Experiments - Paper “Performance Analysis of Distributed GPU-Accelerated Task-Based Workflows” - EDBT 2024

Authors: Marcos Carvalho; Anna Queralt; Oscar Romero; Alkis Simitsis; Cristian Tatu; Rosa M. Badia

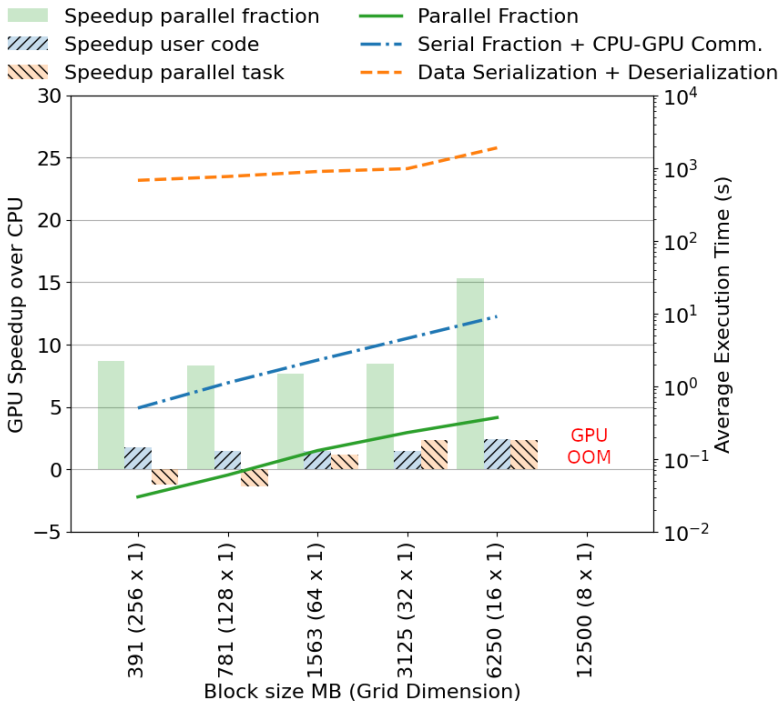
1) Larger datasets

- **Experimental setup:**
  - **Parameters:**
    - **Algorithms:**
      - Matmul (dislib version 0.6.4)
        - Dataset size: 32 GB (64K x 64K elements):
          - Block size 2048 MB (16K x 16K elements), grid dimension 4x4
          - Block size 512 MB (8K x 8K elements), grid dimension 8x8
          - Block size 128 MB (4K x 4K elements), grid dimension 16x16
      - K-means (dislib version 0.6.4)
        - #Clusters: 10
        - Dataset size: 100 GB (125M x 100 elements)
          - Block size 6250 MB (7812500 x 100 elements), grid dimension 16x1
          - Block size 3125 MB (3906250 x 100 elements), grid dimension 32x1
          - Block size 1563 MB (1953125 x 100 elements), grid dimension 64x1
          - Block size 781 MB (976562 x 100 elements), grid dimension 128x1
          - Block size 391 MB (488281 x 100 elements), grid dimension 256x1
    - **Resource:** Minotauro (8 worker nodes each with 16 CPU cores and 4 GPU devices) with shared disk as storage architecture
    - **System:** COMPSs, task order generation as scheduling policy
  - **Monitored metrics:**
    - The same metrics monitored in Figures 7a and 7b in the paper

- **Charts**
  - Matmul dislib 32 GB



  - K-means dislib 100 GB



- **Interpretation of results:**
    - For both algorithms, the point of maximum parallel task speedup remained the same (i.e. grid dimension 4x4 for Matmul and 32x1 for K-means), as observed in Figures 7a and 7b in the paper. This happens because this is the point where GPUs achieve the maximum task parallelism. However, since now tasks process a larger volume of data:
        - The shifting point between the parallel fraction execution time (green line) and the CPU-GPU communication (blue line) in Figure 7a moved to the left of the chart (i.e. larger grid dimensions) in Matmul. As in the previous results for K-means (i.e. with smaller datasets), the serial fraction and CPU-GPU communication times dominate the parallel fraction in all cases.
        - Although not possible to see in the chart (due to GPU OOM), the shifting point of task serialization+deserialization gives the impression of remaining in grid dimension 4x4 in Matmul. The shifting point of task serialization+deserialization remained in grid dimension 32x1 in K-means.

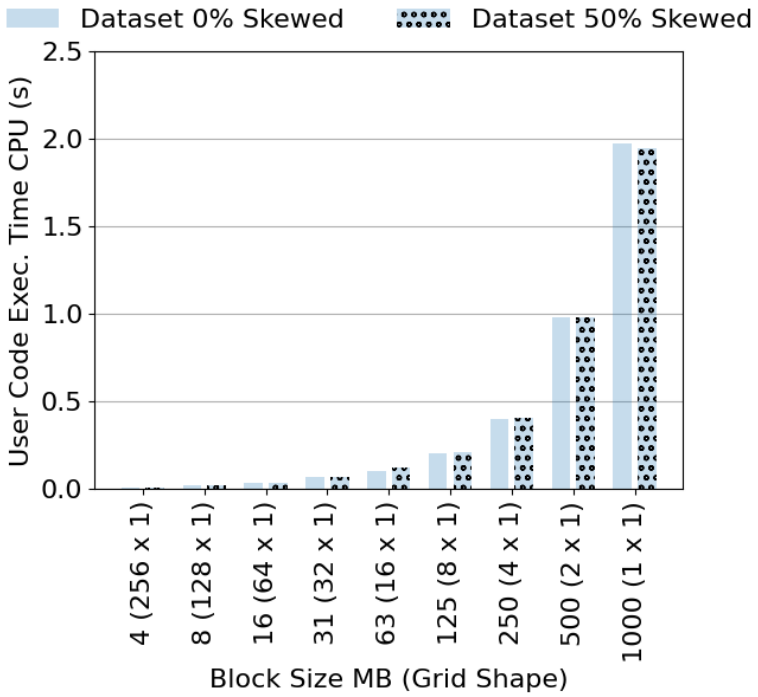
2) Skewed and sparse datasets

For skewed datasets:

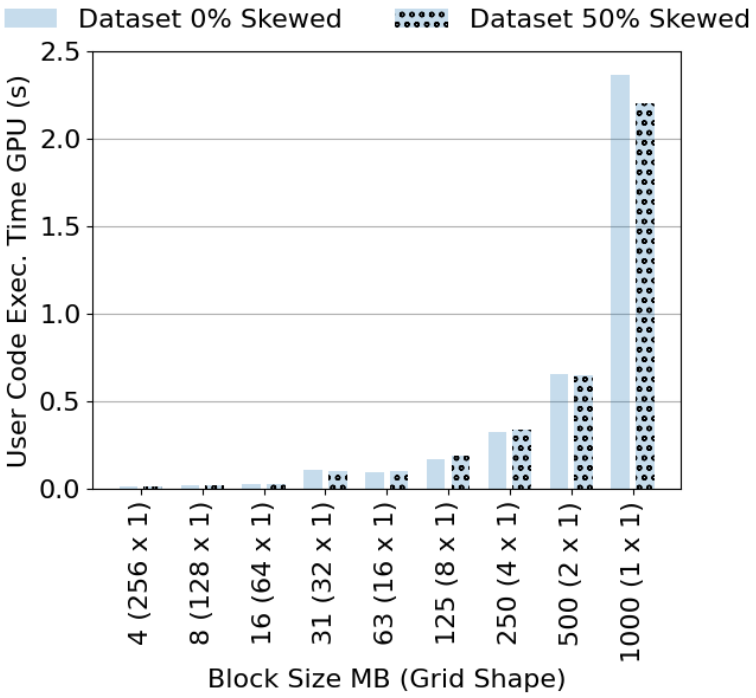
- **Experimental setup**
    - **Parameters:**
        - **Algorithm:**
            - K-means (dislib version 0.6.4)
                - Dataset size: 1 GB (1.25M x 100 elements):
                    - Block sizes (grid dimensions): 3.9 MB (256x1), 7.8 MB (128x1), 15.6 MB (64x1), 31.3 MB (32x1), 62.5 MB (16x1), 125 MB (8x1), 250 MB (4x1), 500 MB (2x1), 1 GB (1x1)
                    - Data distribution skewness:
                        - 0% of skewness (uniform distribution)
                        - 50% of skewness
                - #Clusters: 10
        - **Resource:** Minotauro (8 worker nodes each one with 16 CPU cores and 4 GPU devices) with shared disk as storage architecture
        - **System:** COMPSs, task order generation as scheduling policy
    - **Monitored metrics:**
        - User code execution time

For sparse datasets:

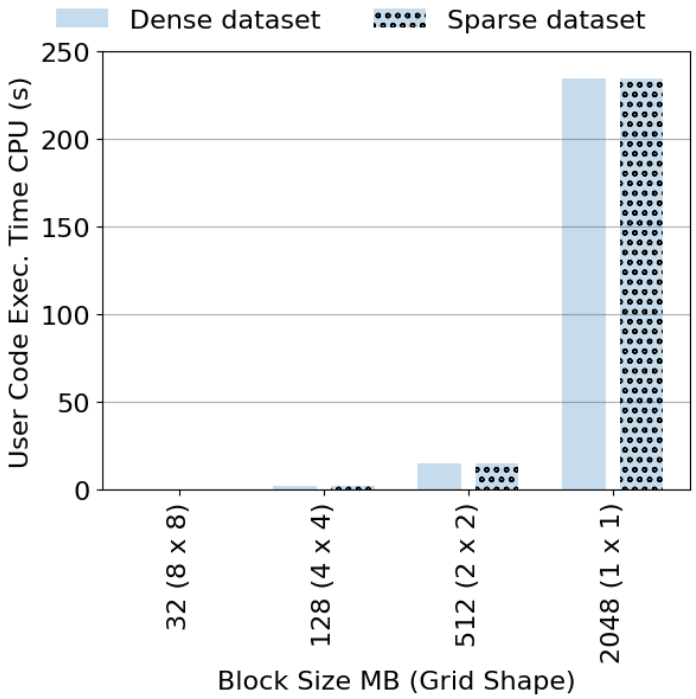
- **Experimental setup:**
  - **Parameters:**
    - **Algorithm:**
      - Matmul (dislib version 0.6.4)
        - Dataset size: 2 GB (16K x 16K elements):
          - Block sizes (grid dimensions): 32 MB (8x8), 128 MB (4x4), 512 MB (2x2), 2 GB (1x1)
          - Data sparsity:
            - 0% of sparsity (all elements non-null (dense))
            - 100% of sparsity (all elements null)
    - **Resource:** Minotauro (8 worker nodes each with 16 CPU cores and 4 GPU devices) with shared disk as storage architecture
    - **System:** task order generation as scheduling policy
  - **Monitored metrics:**
    - User code execution time
- **Charts**
  - K-means dislib 1 GB, CPU executions



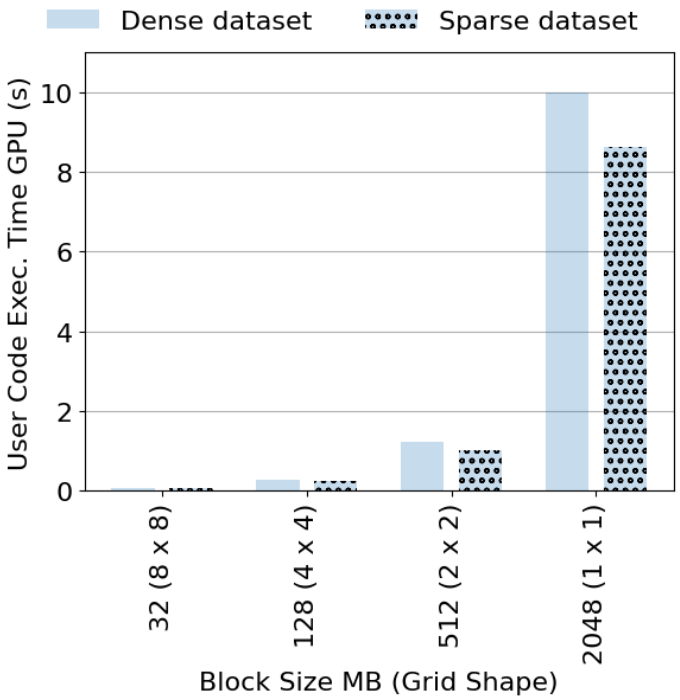
- K-means dislib 1 GB, GPU executions



- Matmul dislib 2 GB, CPU executions



- Matmul dislib 2 GB, GPU executions

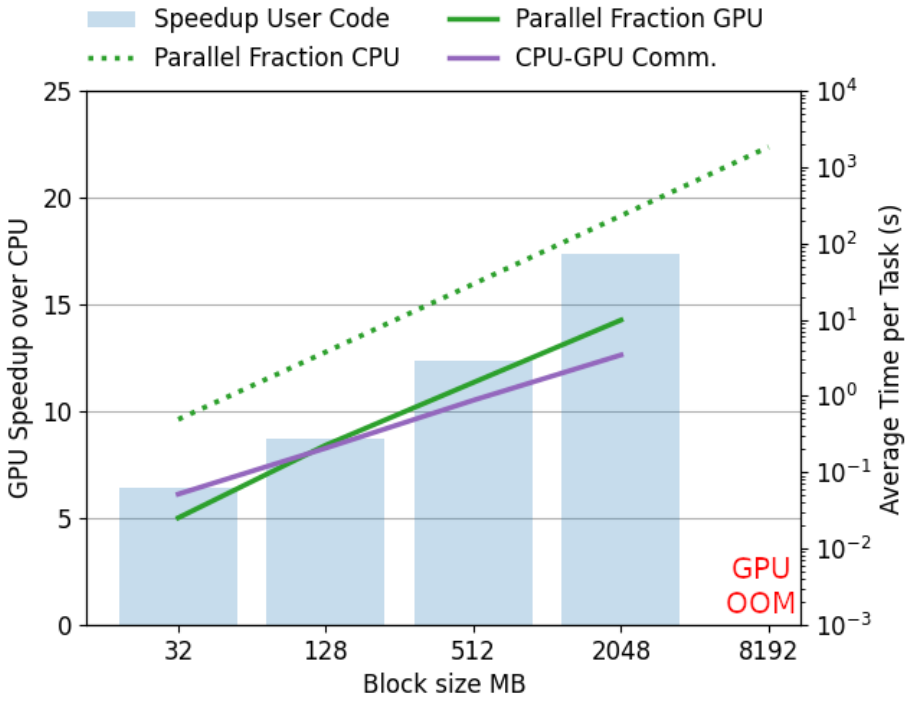


- **Interpretation of results (for skewed and sparse datasets)**:
  - In both cases (i.e., for K-means with skewed data and Matmul with sparse data), we compared the results to the ones obtained using the uniform distribution and no relevant changes between the executions with different datasets were observed. This happens because there is no special treatment of skewed/sparse data in the algorithms and CPUs and GPUs process such data the same way as a uniformly distributed data.

3) Additional algorithm:

- **Experimental setup:**
  - **Parameters:**
    - **Algorithm:**
      - Matmul (Fused Mutiplication Add (FMA) version)
        - Dataset size: 8 GB (32K x 32K elements):
          - Block sizes (grid dimensions): 32 MB (16x16), 128 MB (8x8), 512 MB (4x4), 2 GB (2x2), 8 GB (1x1)
    - **Resource:** Minotauro (8 worker nodes each with 16 CPU cores and 4 GPU devices) with shared disk as storage architecture
    - **System:** COMPSs, task order generation as scheduling policy
  - **Monitored metrics:**
    - Parallel fraction execution time, serial fraction execution time (if applicable to this algorithm), and CPU-GPU communication time

- **Charts**
  - Matmul FMA 8 GB



- **Interpretation of results**:
  - This is another example of a fully parallelizable algorithm that performs both blocked matrix multiplication and matrix sum in a single task (*fused_multiply_add* task), instead of having different tasks for each operation like in the dislib version of Matmul (*matmul_func* and *add_func* tasks). The results revealed the same trends observed in the other algorithms with respect to the user code speedup, parallel fraction, and CPU-GPU communication times.