Sprint Documentation

<<Model>> Sprint

Contact: [info@nat-esm.de](mailto:info@nat-esm.de)

Published on <<xx.xx.xx>> on <https://www.nat-esm.de/services/accepted-sprints>

This document is intended to summarize the technical and scientific results, work and solutions performed during the natESM sprints. As indicated on the webpage of the project, the audience for this document is the ESM community/public (PU).

# Summary

Short overview of the reasons for applying for the sprint, the model, programming language, main challenges during the sprint, and main outcomes (technical and scientific) that probably are most interesting for the readers.

# General Information

|  |  |
| --- | --- |
| **Start and end date:** | <<dd.mm.yyyy – dd.mm.yyyy>> |
| **Intended period:** | 6 months |
| **Responsible RSE:** | Sergey Sukov (JSC) |
| **Responsible scientist:** | <<name, affiliation>> |

# Sprint Objectives

Focus of the sprint, objectives explained in detail

# Procedure and Insights

## Technical Approach / Procedure

Technical parts of the sprint. First/second/third “work packages” explained

## General Insights

|  |  |
| --- | --- |
|  |  |
| a) Figure YY a. | b) Figure YY b. |
| Figure YY. | |

|  |
| --- |
|  |
| Figure YY. |

|  |
| --- |
| ! numOmpThreads – constant number of OpenMP threads  ! idAccStream – OpenACC asynchronous stream ID  !$OMP PARALLEL NUM\_THREADS(numOmpThreads) PRIVATE(idAccStream)  idAccStream = omp\_get\_thread\_num() + 1  !$OMP DO  DO iGridNode=1, numGridNodes ! Main loop over grid nodes  ! Adding first GPU kernel to the idAccStream queue  !$ACC PARALLEL … ASYNC(idAccStream)  …  !$ACC END PARALLEL  …  ! Adding last GPU kernel to the idAccStream queue  !$ACC PARALLEL … ASYNC(idAccStream)  …  !$ACC END PARALLEL  …  ! Waiting for all async kernels in idAccStream  ! async queue to complete  !$ACC WAIT(idAccStream)  END DO  !$OMP END DO  !$OMP END PARALLEL |
| Figure YY. |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Problem size | | Execution time (seconds) | | | | | | |
| Matrix size | Number of tasks | | Cuda BLAS subroutines | | OpenACC self-implementation | | CPU Blas subroutines | |
| Loop | Batched | Loop | Batched | |  |
| 40 | 447392 | | 0.81 | < 0.05 | 1.02 | 0.04 | | 0.62 |
| 64 | 176128 | | 0.60 | 0.41 | 0.06 | | 0.61 |
| 96 | 77824 | | 0.15 | 0.19 | 0.10 | | 0.60 |
| 128 | 43008 | | 0.10 | 0.15 | 0.14 | | 0.60 |
| 256 | 12288 | | 0.04 | 0.32 | 0.41 | | 0.73 |
| Table XX. Name | | | | | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Problem size | | Execution time (seconds) | | |
| Matrix size | Number of tasks | | CPU | GPU |
| 40 | 447392 | | 0.34 | 65.34 |
| 64 | 176128 | | 0.34 | 31.51 |
| 96 | 77824 | | 0.34 | 17.82 |
| 128 | 43008 | | 0.34 | 11.85 |
| 256 | 12288 | | 0.39 | 5.85 |
| 512 | 3072 | | 0.50 | 3.66 |
| 1024 | 768 | | 1.20 | 1.55 |
| 2048 | 192 | | 1.91 | 0.79 |
| Table XX. Name | | | | |

|  |  |
| --- | --- |
|  |  |
| a) Figure YY a. | b) Figure YY b. |
| Figure YY. Name | |

These graphs demonstrate, in particular, that the optimal number of asynchronous streams for different code blocks (kernels) differs.

General information gained from the sprint, problems solved, new information gained

# Results

Short overview of technical and scientific results, maybe some graphics/data.

# Conclusions and Outlook

Short conclusions, maybe outlook – what need to be done in future.

|  |
| --- |
| ! sizeBatch – constant number of grid nodes in a batch  numBatches = numGridNodes / sizeBatch ! number of batches  DO iBatch=1, numBatches ! Main loop over grid node batches  sIndex = (iBatch - 1) \* sizeBatch + 1 ! Index of the first grid node in the batch  fIndex = sIndex + sizeBatch ! Index of the last grid node in the batch  ! GPU kernel for processing a batch of grid nodes  !$ACC PARALLEL LOOP ... ASYNC(1)  DO iGridNode=sIndex, fIndex  ...  ENDDO  !$ACC END PARALLEL  !$ACC WAIT(1)  ...  ! Concurrent execution of GPU kernels to process individual grid nodes  !$OMP PARALLEL NUM\_THREADS(numOmpThreadsX) PRIVATE(idAccStream)  idAccStream = omp\_get\_thread\_num() + 1  !$OMP DO  DO iGridNode=sIndex, fIndex  ! Adding GPU kernel to the idAccStream queue  !$ACC PARALLEL … ASYNC(idAccStream)  ...  !$ACC END PARALLEL  ENDDO  !$OMP END DO  !$ACC WAIT(idAccStream)  !$OMP END PARALLEL  ...  ENDDO |
| Figure YY. |

# References

A full documentation can be found on << PLATFORM>>:   
<<LINK: should be open access for everyone>>