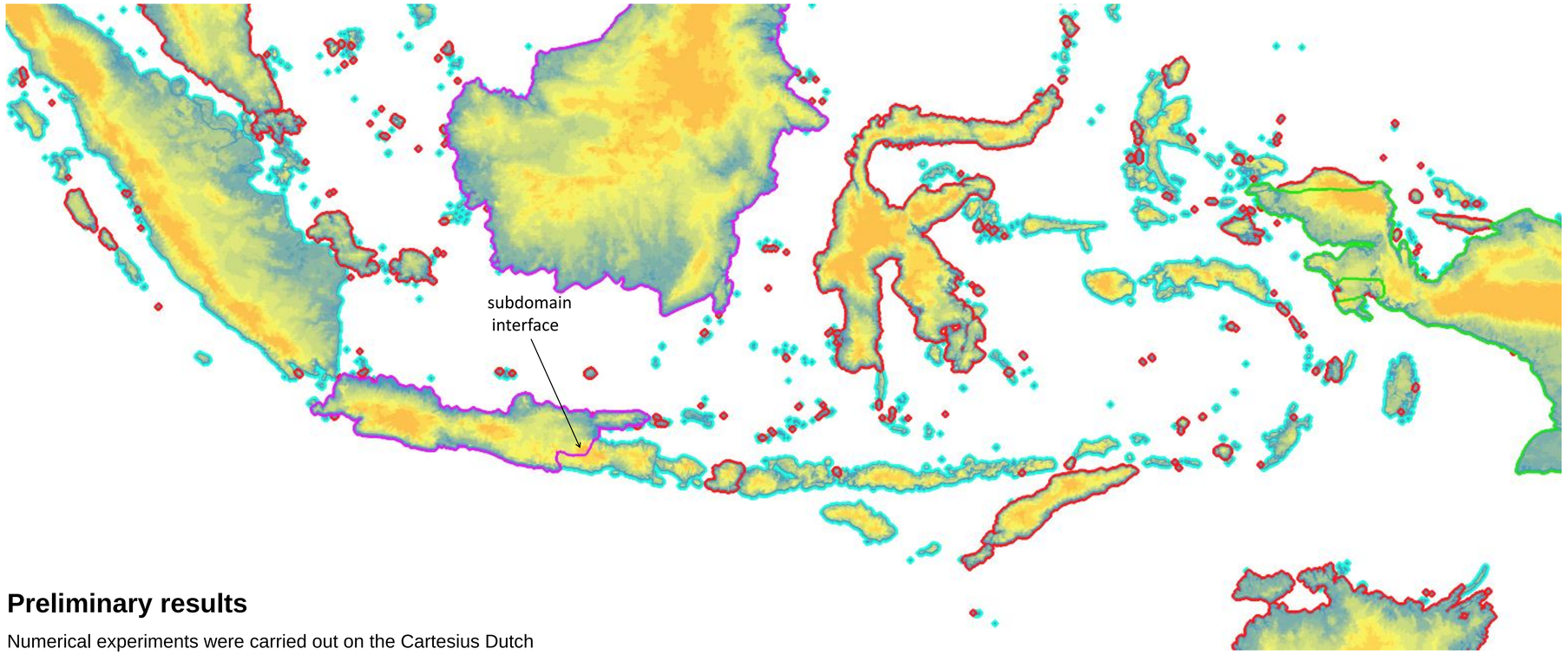
A Hybrid, Parallel Krylov Solver for MODFLOW using Schwarz Domain Decomposition

Jarno Verkaik & Joseph D. Hughes & Edwin H. Sutanudjaja *(jarno.verkaik@deltares.nl)*



Overview

In order to support decision makers in solving hydrological problems, detailed high-resolution models are often needed. These models typically consist of a large number of computational cells and have large memory requirements and long run times.

An efficient technique for obtaining realistic run times and memory requirements is parallel computing, where the problem is divided over multiple processor cores. The new Parallel Krylov Solver (PKS) for MODFLOW-USG is presented here. It combines both distributed memory parallelization by the Message Passing Interface (MPI) and shared memory parallelization by Open Multi-Processing (OpenMP).

Mathematical model

PKS includes the Preconditioned Conjugate Gradient (PCG) and Preconditioned BiConjugate Gradient Stabilized (BiCGSTAB) linear Krylov solvers. For these solvers, an overlapping Schwarz domain decomposition preconditioning is applied:

$$M^{-1} = \sum_i \tilde{R}_i^T A_i^{-1} R_i$$

where A_i correspond to the interior coefficients of the linear system for the (overlapping) subdomain, $\tilde{R}_i$ and R_i are the restriction operators for the non-overlapping and overlapping subdomain, respectively. For the subdomain solve, an incomplete ILU-factorization is used as a preconditioning only. In both the PCG and BiCGSTAB methods, parallel vector updates and interior products are done in a non-overlapping way.

For the preliminary results, we restrict ourselves to non-overlapping PCG exclusively. Furthermore, for the non-linear solution, we restrict ourselves to Picard iteration.

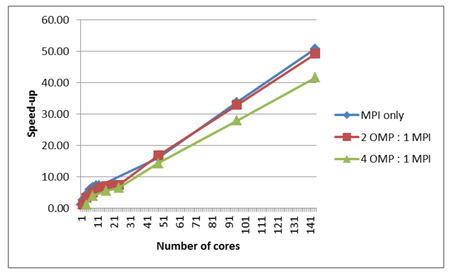
Implementation

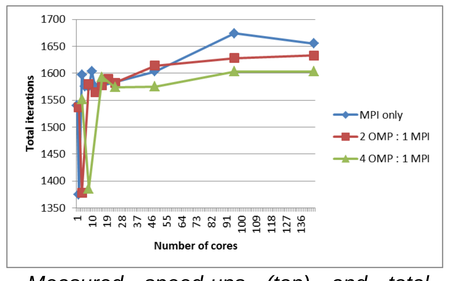
The PKS package is added to the MODFLOW-USG v1.2.00 code, respecting the original code as much as possible. PKS supports both structured and unstructured grids: for structured, several partitioning options can be used, including the recursive coordinate bisection method. For unstructured, the METIS graph partitioning library is used. Input can be read in serial and in parallel; output is written in parallel. PKS is largely based on the unstructured PCGU-solver, and supports OpenMP for parallellizing BLAS-like operations. Depending on the available hardware, PKS can run exclusively with MPI, exclusively with OpenMP, or with a hybrid MPI/OpenMP approach.

Preliminary results

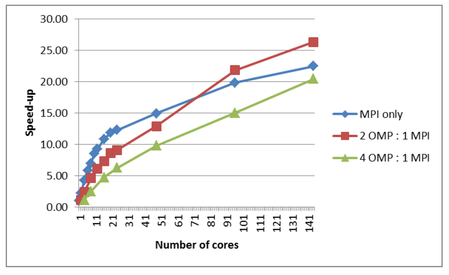
Numerical experiments were carried out on the Cartesius Dutch National supercomputer. This machine is ranked 45 in the TOP500 having ~40,000 computational cores and a fast InfiniBand interconnect. Experiments were done on nodes consisting of 2 Haswell 12-core CPUs (E5-2690 v3), having 64 GB RAM. Two steady-state cases were considered: a synthetic case for testing PKS-structured (~ 112 million cells, 7500 x 7500 x 2) and the Indonesia groundwater model (~ 4 million cells) for testing PKS-unstructured. Both models use square, uniform, cells. The synthetic case simulates groundwater flow for a 10km x 10km square, in two confined aquifers, each having a heterogeneous conductivity distribution. For both tests, two pre-processing steps were done: 1. the actual partitioning (structured: uniform in row/column direction, unstructured: METIS), 2. reading the partitioning data and clipping all the raster data. A HCLOSE stopping criterion of 0.001 m was used for all simulations.

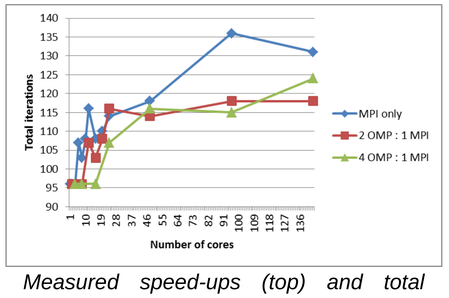
The scaling results show that OpenMP can compensate for the increasing number iterations compared to pure MPI, and hence, hybrid can be most optimal. For the Cartesius, it seems that using 2 OpenMP threads for each MPI task is most optimal. Besides adding more overlap, for which we will not present the results here, we believe that this hybrid MPI-OpenMP approach can increase speed-ups. We expect that this benefit is even larger for clusters with slower interconnects such as Gigabit Ethernet.





Measured speed-ups (top) and total iterations (bottom) for the synthetic model up to 144 cores, overlap 1 cell. Serial computation takes ~1 hour 43 minutes and requires ~53 GB RAM memory.





Measured speed-ups (top) and total iterations (bottom) for the Indonesia model up to 144 cores, overlap 1 cell. Serial computation takes ~48 seconds and requires ~2 GB RAM memory.