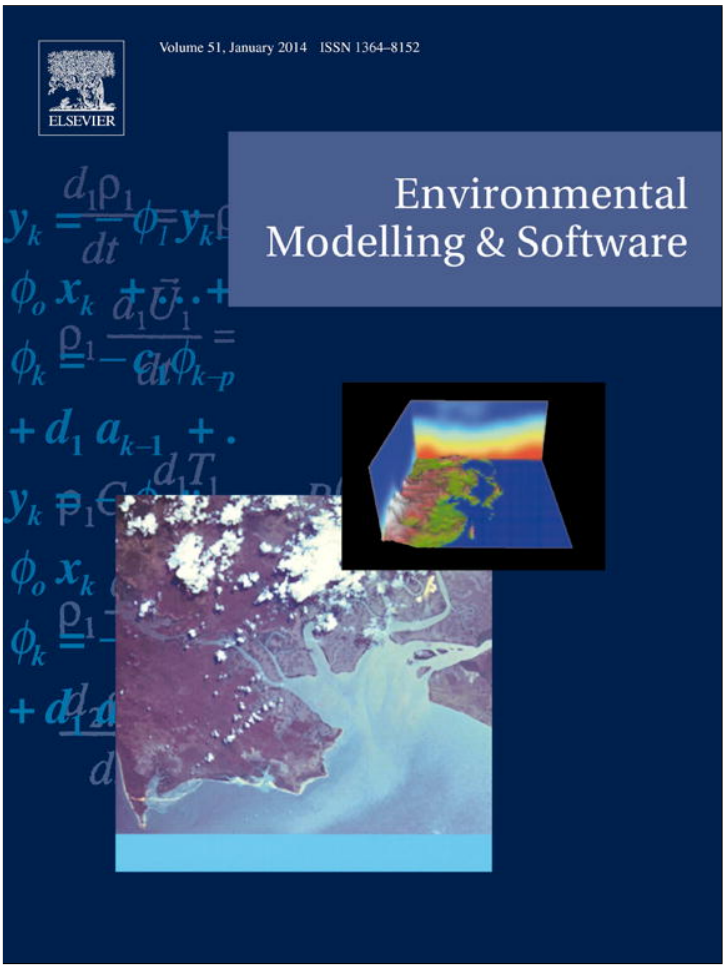

Environmental Modelling & Software 51 (2014) 221–227

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

A layered approach to parallel computing for spatially distributed hydrological modeling

Junzhi Liu [a,b], A-Xing Zhu [a,c,*], Yongbo Liu [d], Tongxin Zhu [e], Cheng-Zhi Qin [a]

[a] State Key Lab of Resources and Environmental Information System, Institute of Geographic Sciences and Natural Resources Research, CAS, Beijing 100101, China
[b] University of Chinese Academy of Sciences, CAS, Beijing 100049, China
[c] Department of Geography, University of Wisconsin–Madison, Madison, WI 53706, USA
[d] Department of Geography, University of Guelph, 50 Stone Road East, Guelph, Ontario N1G 2W1, Canada
[e] Department of Geography, University of Minnesota-Duluth, Duluth, MN 55812, USA

ARTICLE INFO

Article history:
Received 25 April 2013
Received in revised form
1 October 2013
Accepted 3 October 2013
Available online 31 October 2013

Keywords:
Distributed hydrological model
Parallel computing
Domain decomposition
Simulation units Layering
OpenMP

ABSTRACT

Distributed hydrological simulations over large watersheds usually require an extensive amount of computation, which necessitates the use of parallel computing. Each type of hydrological model has its own computational characteristics and therefore needs a distinct parallel-computing strategy. In this paper, we focus on one type of hydrological model in which both overland flow routing and channel flow routing are performed sequentially from upstream simulation units to downstream simulation units (referred to as Fully Sequential Dependent Hydrological Models, or FSDHM). There has been little published work on parallel computing for this type of model. In this paper, a layered approach to parallel computing is proposed. This approach divides simulation units into layers according to flow direction. In each layer, there are no upstream or downstream relationships among simulation units. Thus, the calculations on simulation units in the same layer are independent and can be conducted in parallel. A grid-based FSDHM was parallelized with the Open Multi-Processing (OpenMP) library to illustrate the implementation of the proposed approach. Experiments on the performance of this parallel model were conducted on a computer with multi-core Central Processing Units (CPUs) using datasets of different resolutions (30 m, 90 m and 270 m, respectively). The results showed that the parallel performance was higher for simulations with large datasets than with small datasets and the maximum speedup ratio reached 12.49 under 24 threads for the 30 m dataset.

Published by Elsevier Ltd.

Software availability

Program title: LayeredParallelModel
Description: A parallel Fully Sequential Dependent Hydrological Model (FSDHM) using a layered approach
Developer: Dr. Junzhi Liu and Prof. A-Xing Zhu
Platform: Microsoft Windows, Linux
Source language: C++
Cost: Free
Availability: Contact the developer

1. Introduction

Spatially distributed hydrological modeling has been widely used in hydrological studies and watershed management (Borah and Bera, 2004). In order to conduct detailed simulations of hydrological processes, physically based methods and high spatial and temporal resolutions are required (Hessell, 2005; Rojas et al., 2008). Nevertheless, such simulations on large watersheds usually require an extensive amount of computation. Simulation methods based on serial computation techniques cannot meet the demands of large-scale watershed simulation on computing capacity; thus parallel computing is needed (Vivoni et al., 2011; Liu et al., 2013a). Currently, a growing number of researches have been conducted on parallel computing of distributed hydrological models as well as other environmental and socio-economic models (Apostolopoulos and Georgakakos, 1997; Bryan, 2013; Li et al., 2011; Qin and Zhan, 2012; Zhao et al., 2012, 2013).

* Corresponding author. Institute of Geographic Sciences and Natural Resources Research, CAS, State Key Lab of Resources and Environmental Information System, Beijing 100101, China. Tel.: +86 13810787680; fax: +86 10 64889630.
E-mail address: axing@lreis.ac.cn (A-X. Zhu).

1364-8152/$ – see front matter Published by Elsevier Ltd.
http://dx.doi.org/10.1016/j.envsoft.2013.10.005

Because hydrological simulations are temporally successive, spatial domain decomposition is usually adopted for parallel computing of distributed hydrological models (Wang et al., 2011). Each type of hydrological model has its own computational characteristics and therefore needs a distinct domain decomposition strategy. In this paper, we focus on one type of hydrological model (Fully Sequential Dependent Hydrological Models, FSDHM) in which both overland flow routing and channel flow routing are performed sequentially from upstream simulation units to downstream simulation units. Typical FSDHM include grid-based models such as TOPKAPI (Ciarapica and Todini, 2002), LIS-FLOOD (Van Der Knijff et al., 2010), and flow-tube based models such as TOPOG (Vertessy et al., 1993) and Thales (Grayson et al., 1992).

The research on parallel computing of distributed hydrological models has focused primarily on sub-basin based parallel computing (e.g. Apostolopoulos and Georgakakos, 1997; Li et al., 2011; Wang et al., 2011; Wu et al., 2013; Yalew et al., 2013). The sub-basin based parallel computing methods can apply to FSDHM. However, they cannot utilize the fine-grained parallelizability within each sub-basin at the basic unit (e.g. grid) level, which can also be used to improve the parallel-computing performance. Currently, there is no published research on basic unit based parallel computing for FSDHM (Liu et al., 2013b). In this paper, we aim to develop such a parallel-computing approach for FSDHM utilizing the parallelizability at basic unit level.

In this approach, most hydrological processes are directly parallelized at the basic simulation unit (e.g., grid) level except that the channel flow routing process is parallelized at reach level. The basic principle of this approach is to divide simulation units, whether grids or reaches, into layers. Section 2 presents the principle of the proposed approach. Section 3 describes the implementation of this approach, using a grid-based FSDHM as an example. Section 4 discusses the experiments and results. Section 5 concludes and discusses future research directions.

2. Assumptions of FSDHM

There are several assumptions on the computational characteristics of FSDHM in this paper. These assumptions will affect the design of parallel strategies and are listed as follows:

(1) A watershed is divided into sub-basins and each sub-basin is further divided into basic simulation units such as grids.
(2) Each simulation unit is assigned a topography-based flow direction.
(3) There are no connections between surrounding simulation units except for the adjacent upstream and downstream ones.
(4) For flow-routing calculation within a time step, the calculation of downstream units cannot be started until all its upstream units have been processed.

3. The basic principle of a layered approach to parallel computing

Hydrological processes in FSDHM can be classified into two types: runoff generation processes and flow routing processes. For runoff generation, calculations for different simulation units are independent; therefore parallel computing can be conducted by simply dividing simulation units into equal parts. For flow routing, there are sequential dependences among upstream and downstream simulation units, so parallel computing cannot be conducted by dividing simulation units directly.

The routing calculation of a simulation unit depends on its upstream units, which are defined by flow direction (for example, at grid level) or drainage network (at reach level). This dependence restricts the spatial decomposition strategies for routing processes because downstream simulation units cannot be processed until the calculation of all their upstream units has been finished. In light of this restriction, a layered approach was proposed. This approach first divides simulation units (for example, grids and reaches) into layers and ensures that there are no upstream or downstream relationships between simulation units in the same layer. Thus, the calculations of simulation units in the same layer are independent and can be conducted in parallel. Of course, because the outflow of upstream units is needed by the corresponding downstream units, the calculation of units in upstream layers should be conducted first.

Because the number of basic units (e.g. grid) in a watershed is usually large, data exchange among upstream and downstream units would cause a large amount of communication in parallel computing. To alleviate the communication overhead, the shared-memory programming model, which has good communication efficiency (Rauber and Rünger, 2010), was adopted in this paper.

4. Implementation of the layered approach for a grid-based FSDHM

4.1. Description of a grid-based FSDHM

In this paper, a grid-based FSDHM was parallelized to illustrate the implementation of the proposed layered parallel-computing approach. This model integrated several existing methods to conduct event-based hydrological simulation in semi-arid watersheds (see the following paragraphs for details) and was implemented by the authors using the C++ programming language.

The hydrological processes simulated in this model include interception, infiltration, surface depression, overland flow routing and channel flow routing. Both overland flow routing and channel flow routing were performed sequentially from upstream to downstream according to single flow direction defined by the D8 method (O'Callaghan and Mark, 1984). Because runoff generation is dominated by infiltration excess overland flow in semi-arid watersheds, the interflow and groundwater flow processes were not simulated.

The infiltration process was simulated by a quadratic approximation of the Green-Ampt method (Li et al., 1975). The interception and depression processes were simulated using methods in the Wetspa model (Liu and De Smedt, 2005). The fill-and-spill mechanism was used for the interception process, and the maximum interception storage was calculated using a statistical equation containing leaf area index, vegetal species, and date (Liu and De Smedt, 2004). For the depression process, depression and overland flow were allowed to occur simultaneously even if excess rainfall was less than the depression storage, and the depression storage was estimated by an empirical equation suggested by Linsley et al. (1975).

For the overland flow routing and channel flow routing processes, a one-dimensional kinematic wave model was used in combination with the Manning's equation (Chow et al., 1988), which was also used by the LIS-FLOOD model (Van Der Knijff et al., 2010). The equation for surface-water depth is

$$\frac{\partial h}{\partial t}+\frac{1}{n}\sqrt{S_0}\,\frac{\partial\left(h^{5/3}\right)}{\partial x}=i-f \tag{1}$$

where h is the depth of water on the surface (m); t is the time (s); n is the Manning coefficient; S_0 is the bed slope(m/m); x is the length

of the slope (m); i is the rainfall intensity (m/s); f is the infiltration rate (m/s). This equation was solved by a combination of the 4-point implicit finite difference and Newton's iteration method.

4.2. Method for layering simulation units

The first step of the proposed parallel-computing approach was to divide the simulation units into layers within which the units are independent of each other. In this paper, we used a flow topology to divide simulation units into layers. The single flow direction (D8) method (O'Callaghan and Mark, 1984) was used to define the flow topology. The outlet unit is labeled as layer 1 and the units draining directly to units in layer 1 is defined as layer 2. The units draining directly to units in layer n is defined as layer $n+1$ and so on and so forth until the most upstream units are reached (Wang et al., 2004). Fig. 1 and Fig. 2 illustrate the layering results for grids and reaches, respectively. Because routing calculation should be conducted from upstream to downstream, layers with a larger number will be processed first during routing calculation. The pseudo code of the layering algorithm is shown in Appendix A.1. In this algorithm, the outlet unit was first processed and then its upstream units were processed recursively from downstream to upstream.

It is worth noting that if grid-to-grid channel flow routing is adopted, the layering information at reach level (Fig. 2-b) is implicit in the layering result at grid level (Fig. 1-b), so layering of reaches for the specific model described in Section 4.1 is not indispensable. But the proposed approach in this paper is oriented to FSDHM in general, including those models in which the basic units for channel flow routing are reaches rather than grids. Therefore, the reaches are layered separately in this paper.

4.3. Data structure

Two-dimensional arrays are usually used to store watershed raster data such as slope and flow direction in traditional grid-based distributed hydrological models. However, because the shape of a watershed is usually not a regular rectangle, there are many no-data grids outside the watershed boundary. This causes several drawbacks. First, the no-data values waste memory. Second, the model must check whether a grid contains a no-data value before calculation, and this wastes computing time. Third and the most important, the no-data grids are prone to cause load imbalance due to their irregular spatial distribution.

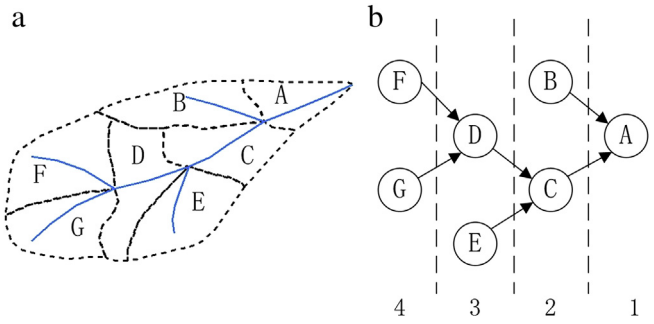

Fig. 2. Division of reaches according to (a) drainage network into (b) layers (Arrows in the figure represent flow directions; A total of 4 reach layer are divided in this example).

To overcome the drawbacks of two-dimensional arrays, we employed a one-dimensional array data structure to manage the watershed raster data. First, the two-dimensional array was directly unfolded to a one-dimensional array by row. Then the no-data grids were erased from the one-dimensional array to get a compact array (Fig. 3). To ensure consistency of raster data in the same watershed, a mask raster was used in the conversion from two-dimensional to one-dimensional array. Because the spatial location and relationship information were lost in the one-dimensional data structure, several lookup tables were used to store the location information (row and column numbers) and flow relationship information.

4.4. Programming implementation

The C++ programming language and the OpenMP application programming interface (API) were used to implement the study's parallel computing approach. OpenMP is the de facto standard for shared-memory parallel programming (Mattson et al., 2004). It is based on the fork/join programming model and the OpenMP compiler directives embedded in the programming language source code are used to achieve parallel computing.

For the runoff generation processes, including interception, infiltration, and surface storage processes, a *pragma* statement was used to parallelize the *for* loop of the one-dimensional array that stored watershed raster data. This divided the watershed data into domains of equal size and dispatched the computing tasks of each domain to different threads. The C++ pseudo code of this parallel algorithm can be found in Appendix A.2.

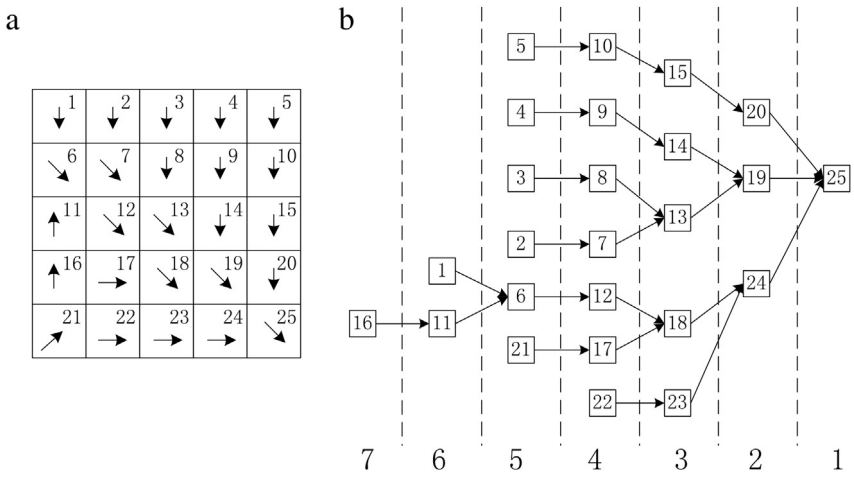

Fig. 1. Division of grids according to (a) flow direction into (b) layers (Arrows in the figure represent flow directions; A total of 7 grid layers are divided in this example).

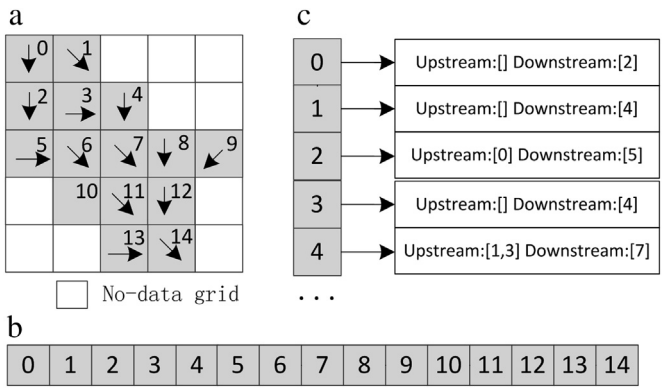

Fig. 3. Watershed raster data represented by (a) two-dimensional array unfolded to (b) compact one-dimensional array with flow relationship information stored in (c) lookup tables. The numbers in this figure represent grid index in (b).

For the routing processes, including overland flow routing and channel flow routing, firstly simulation unit layering was conducted using the algorithm as described in Section 4.2 in the preprocessing phase. Then a two-level nested loop, the outer one for different layers and the inner one for units within each layer, was used for the calculation, and a *pragma* statement was used to parallelize the inner *for* loop. The C++ pseudo code of this parallel algorithm can be found in Appendix A.3. It should be noted that the number of units within a layer may be small in some cases, especially for the reaches (as shown in Fig. 2). If the number of units within a layer is smaller than the number of threads, some threads will be idle during parallel computing.

5. Case studies

5.1. Experimental design

5.1.1. Study area and dataset

The Qingshuihe watershed, located in the Hebei province of China, was selected as the study area (Fig. 4). The watershed area is about 2300 km^2 and the elevation ranges from 781 to 2164 m. It has a sub-arid climate characterized by an average annual precipitation of 480 mm, 80% of which occurs between June and September in the form of intense storms.

Three datasets of different spatial resolutions (30 m, 90 m and 270 m) were used to test the performance of the proposed parallelization method (Table 1). All experiments were performed using a 24-h storm with 50-year recurrence interval at 1-min time step. Because the primary object of this study was to test the parallel efficiency, default input parameters extracted from DEM, landuse map, and soil map, and obtained from literature values were used in these simulations. Only the discharge at the outlet of the watershed was outputted.

5.1.2. Experiment environment

A computer server with 2 CPUs was used as the hardware platform. The CPU type was Intel ® Xeon E5645 2.4G; each CPU had 6 cores (12 logical cores). A Windows 2008 Server Operating System and a VC++ 2010 compiler were used. The OpenMP library was version 2.0.

To test parallel performance, the thread number ranged from 2 to 24, using a varying number of cores. Each experiment was repeated three times to get an average execution time. The computing time of each process was also recorded. The execution

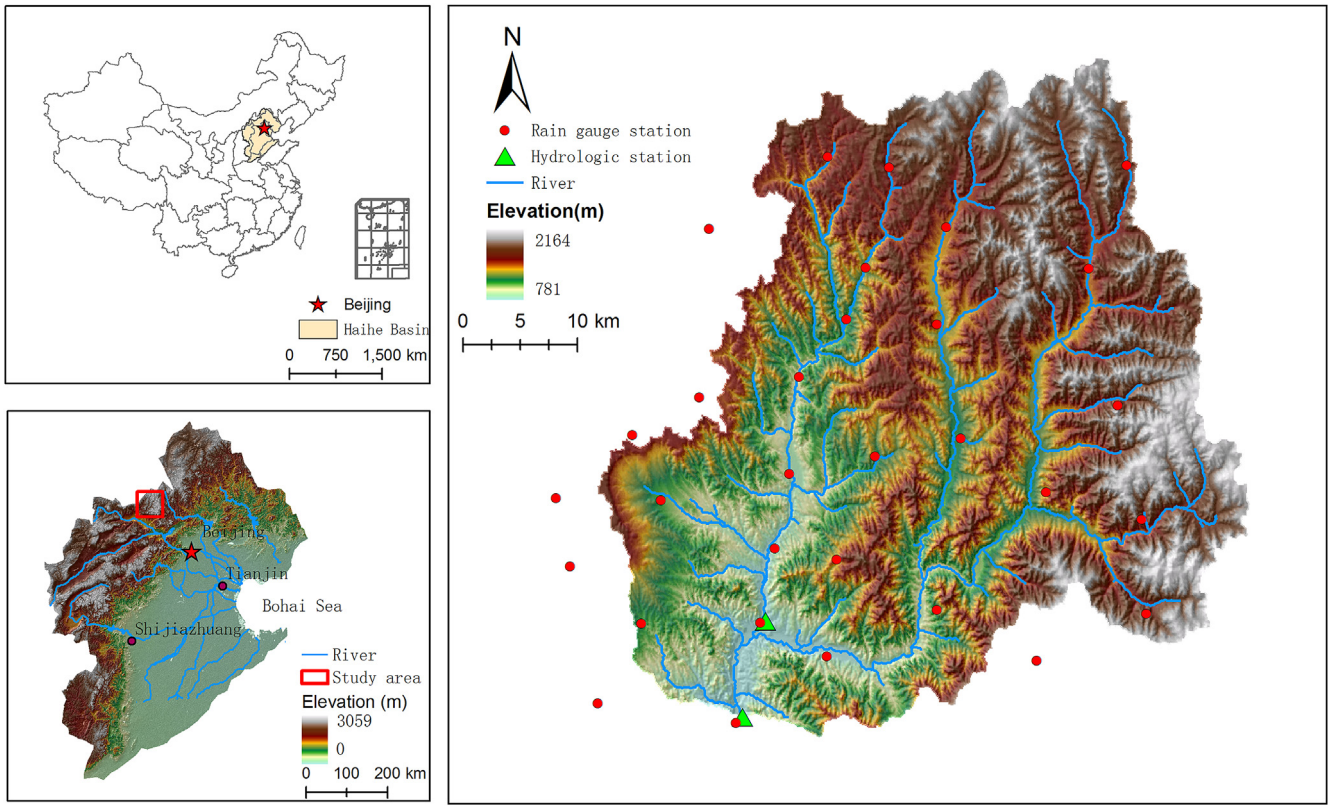

Fig. 4. Location map of the study area.

Table 1
Three datasets with different spatial resolutions.

Experiment	Resolution (m)	Number of grids	Number of layers for overland flow routing	Number of sub-basins	Number of layers for channel flow routing
1	270	29,850	308	115	21
2	90	268,860	971	115	21
3	30	2,381,602	2875	115	21

time here refers to the time from the beginning to the end of the simulation, including data input/output (I/O) time, while the computing time of each process excludes I/O time.

5.1.3. Evaluation index

The speedup ratio and parallel efficiency were used to measure the performance of the proposed method. The speedup ratio is defined as the ratio between the serial computing time and the parallel computing time (Rauber and Rünger, 2010) and the equation is as follows:

$$S = \frac{T_s}{T_p} \tag{2}$$

where S is the speedup ratio; T_s is the serial execution/computing time; T_p is the parallel execution/computing time. The parallel efficiency is defined as the ratio between the speedup ratio and the number of threads (Rauber and Rünger, 2010) and the equation is as follows:

$$E = \frac{S}{N} \tag{3}$$

where E is the parallel efficiency; N is the number of threads.

5.2. Results and discussion

5.2.1. Overall performance

Fig. 5 presents the execution time, speedup ratios and parallel efficiencies for the parallel simulations using different numbers of threads for the three datasets. As expected, the execution time decreased and the speedup ratios increased with the number of threads for all the experiments. The speedup ratio reached 12.49 under 24 threads for the 30 m dataset. When the number of threads was small, this method had very good parallel efficiencies. For example, in our experiments, the parallel efficiencies ranged from 0.89 to 0.97 for 2 threads and from 0.81 to 0.91 for 4 threads, confirming that this method is very effective when used in current mainstream multi-core CPUs. As the number of threads increased, the parallel efficiencies gradually decreased. This was because more computational resources were directed to scheduling rather than to actual computing when the number of threads increased.

We also found that data volume had a direct influence on parallelization performance. The larger the dataset was, the higher the speedup was for a given number of threads. This was because the time spent on actual calculation increased with data volume, which made the rate of scheduling overhead decrease and led to better parallelization performance.

5.2.2. Performance by type of hydrological process

Parallelization strategies varied for different types of hydrological processes, as described in Section 3. The performance of different types of hydrological processes for the 90 m dataset is shown in Fig. 6. The runoff generation processes (i.e. interception, depression, and infiltration), which are displayed as a whole in Fig. 6, performed well, with parallel efficiencies all above 0.6. The performance of the routing processes (i.e. overland flow routing and channel flow routing) was poorer than that of the runoff generation processes.

This result is consistent with the parallelizability of each type of process. The calculations of runoff generation on each simulation unit were independent, so they had good parallelizability. A simple decomposition method was used for their parallelization and, as would be expected, this straightforward parallelization method achieved good performance.

For the overland flow routing process, the parallelization performance was not as good as that of the runoff generation processes. This can be attributed to the increase of scheduling overhead. Within a time step, a two-level nested loop was used for the overland flow routing process and parallel-computing was conducted in the inner loop. So the number of thread scheduling (e.g. thread synchronization) for the overland flow routing process was the number of the outer loop (i.e. the number of layers), which is 971 for the 90 m resolution dataset in this paper. In contrast, the

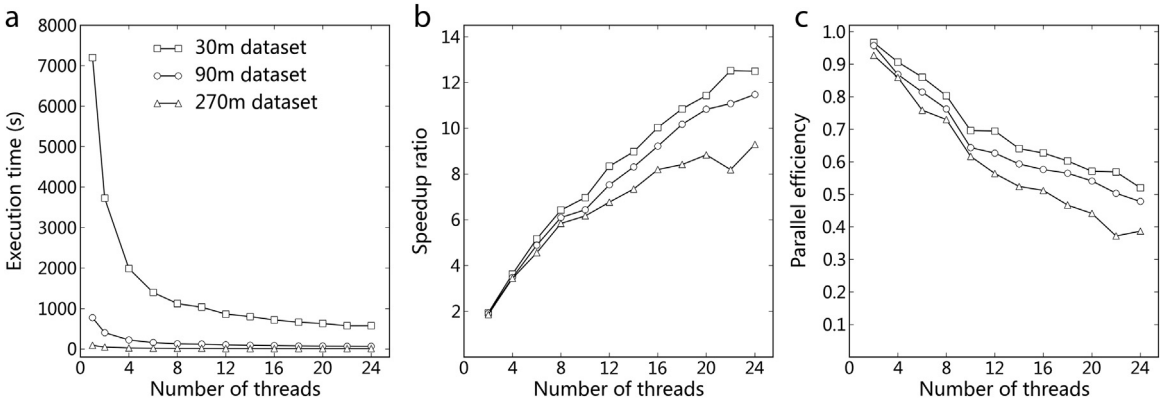

Fig. 5. The (a) execution time, (b) speedup ratios and (c) parallel efficiencies using different numbers of threads for the three datasets (30 m, 90 m and 270 m resolutions). Execution time refers to the time from the beginning to the end of the simulation (including I/O time).

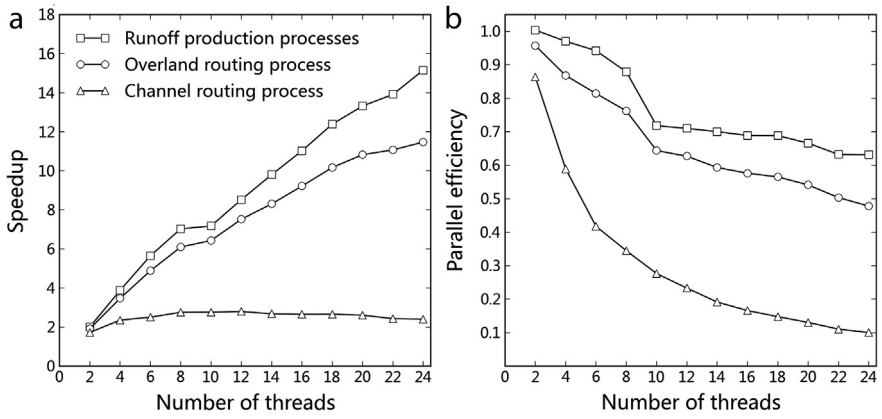

Fig. 6. The (a) speedup ratios and (b) parallel efficiencies of each type of hydrological process for the 90 m resolution dataset.

number of thread scheduling for a runoff generation process was only once. Therefore, the scheduling overhead of the overland flow routing process was much heavier than that of the runoff generation processes.

For the channel flow routing process, the parallel-computing performance was poor. This is partly due to the scheduling overhead as described above. Another reason is that the number of reaches within each routing layer was relatively small, so in many cases the number of threads was larger than the number of reaches within a layer and some threads were idle during parallel computing. In addition, the different amount of calculation among reaches, which was caused by different numbers of reach-grids, also led to load unbalance and low performance.

6. Conclusions

This paper proposed a layered approach to parallel computing for one type of distributed hydrological model (FSDHM). The approach divides the simulation units into layers according to flow direction and ensures that there are no upstream or downstream relationships between simulation units in the same layer. Thus, the calculations on simulation units in the same layer are independent and can be conducted in parallel.

A grid-based FSDHM was parallelized with the OpenMP library to illustrate the implementation of the proposed approach. Its performance was tested in relation to the number of threads and data volume. Overall, this approach provided efficient computational performance. The parallel speedups increased with both the number of threads and data volume. The runoff generation processes had better parallel performance than the routing processes.

In this approach, most hydrological processes are directly parallelized at the basic simulation unit (e.g., grid) level, except that the channel flow routing process is parallelized at the reach level. Because of the significant amount of communication at such a fine level, this approach used the shared-memory programming model, which depends on shared-memory hardware. The hardware environment is usually the limiting factor of this approach. In contrast, sub-basin based parallel computing, with a low communication overhead, can use a more scalable cluster computing environment, but suffers from the limited parallelizability at the sub-basin level (Wang et al., 2012). For these reasons, these two approaches are complementary to each other. A two-level parallel computing strategy could be developed for the multi-core cluster computing environment, in which parallel computing at the sub-basin level is performed among different computing nodes using the message-passing programming model and parallel computing at the basic simulation unit level is performed within each computing node using the shared-memory programming model. In addition, the parallelizability at the time domain can also be utilized in the future. For example, the temporal-spatial discretization parallel-computing method newly developed by Wang et al. (2013), which utilized sub-basin based parallelizability at both the time and spatial domain, can be integrated with the approach proposed in this paper to further improve parallel-computing scalability.

Acknowledgments

This study was funded by the National High-Tech Research and Development Program of China (No. 2011AA120305) and the National Natural Science Foundation of China (No. 41023010). This study was also partly funded by the Program of International S&T Cooperation, MOST of China (No. 2010DFB24140). The support received by A-Xing Zhu through the Vilas Associate Award, the Hammel Faculty Fellow, and the Manasse Chair Professorship from the University of Wisconsin-Madison is greatly appreciated.

Appendix A. Pseudo code of major algorithms

A.1. Pseudo code of the algorithm for layering simulation units

```
function Layering (curID, layerNum)
    // curID is the ID of current unit
    // layerNum is the current layer number
    // layers is a 2D array containing the layering result
    add curID to layers[layerNum]
    for upID in upstream units of curID do
        if (upID is inside watershed)
            LayeringFromOutlet (upID, layerNum+1)
        end if
    end for
end function
// invoke the layering algorithm from outlet
// outletID is the index of the outlet unit
Layering(outletID, 1)
```

A.2. C++ pseudo code of the parallel algorithm for runoff generation processes

```
//nUnits is the total number of units
#pragma omp parallel for
for (int i = 0; i < nUnits; i++)
{
        // conduct simulation in the current unit
        result[i] = function_call(i);
}
```

A.3. C++ pseudo code of the parallel algorithm for routing processes

```
//nLayers is the number of layers
for (int iLayer = 0; iLayer < nLayers; ++iLayer)
{
        //nUnits is the number of units within the current layer
        int nUnits = numUnits[iLayer];
        #pragma omp parallel for
        for (int iUnit = 0; iUnit < nUnits; iUnit++)
        {
                // conduct routing simulation in the current unit
                result[iLayer][iUnit] = function_call(iLayer, iUnit);
        }
}
```

References

Apostolopoulos, T.K., Georgakakos, K.P., 1997. Parallel computation for streamflow prediction with distributed hydrologic models. J. Hydrol. 197 (1–4), 1–24.

Borah, D.K., Bera, M., 2004. Watershed-scale hydrologic and nonpoint-source pollution models: review of applications. Trans. ASAE 47 (3), 789–803.

Bryan, B.A., 2013. High-performance computing tools for the integrated assessment and modelling of social-ecological systems. Environ. Model. Softw. 39, 295–303.

Chow, V., Maidment, D., Mays, L., 1988. Applied Hydrology. McGraw-Hill, New York.

Ciarapica, L., Todini, E., 2002. TOPKAPI: a model for the representation of the rainfall-runoff process at different scales. Hydrol. Process. 16 (2), 207–229.

Grayson, R.B., Moore, I.D., Mcmahon, T.A., 1992. Physically based hydrologic modeling .1. A terrain-based model for investigative purposes. Water Resour. Res. 28 (10), 2639–2658.

Hessell, R., 2005. Effects of grid cell size and time step length on simulation results of the Limburg soil erosion model (LISEM). Hydrol. Process. 19 (15), 3037–3049.

Li, R.M., Simons, D.B., Stevens, M.A., 1975. Nonlinear kinematic wave approximation for water routing. Water Resour. Res. 11 (2), 245–252.

Li, T.J., Wang, G.Q., Chen, J., Wang, H., 2011. Dynamic parallelization of hydrological model simulations. Environ. Model. Softw. 26 (12), 1736–1746.

Linsley, R.K., Kohler, M.A., Paulhus, J.L.H., 1975. Hydrology for Engineers. McGraw-Hlll, New York.

Liu, J.Z., Zhu, A.X., Qin, C.Z., 2013a. Estimation of theoretical maximum speedup ratio for parallel computing of grid-based distributed hydrological models. Comput. Geosci. 60, 58–62.

Liu, J.Z., Zhu, A.X., Qin, C.Z., Chen, L.J., Wu, H., Jiang, J.C., 2013b. Review on parallel computing of distributed hydrological models. Prog. Geogr. 32 (4), 538–547 (in Chinese).

Liu, Y.B., De Smedt, F., 2004. WetSpa Extension, Documentation and User Manual. Department of Hydrology and Hydraulic Engineering, Vrije Universiteit Brussel, Belgium.

Liu, Y.B., De Smedt, F., 2005. Flood modeling for complex terrain using GIS and remote sensed information. Water Resour. Manag. 19 (5), 605–624.

Mattson, T., Sanders, B., Massingill, B., 2004. Patterns for Parallel Programming. Addison-Wesley Professional, Boston.

O'Callaghan, J.F., Mark, D.M., 1984. The extraction of drainage networks from digital elevation data. Comput. Vis. Graphics, Image Process. 28 (3), 323–344.

Qin, C.Z., Zhan, L.J., 2012. Parallelizing flow-accumulation calculations on graphics processing units – from iterative DEM preprocessing algorithm to recursive multiple-flow-direction algorithm. Comput. Geosciences 43, 7–16.

Rauber, T., Rünger, G., 2010. Parallel Programming: For Multi-core and Cluster Systems. Springer-Verlag New York Inc, New York.

Rojas, R., Velleux, M., Julien, P.Y., Johnson, B.E., 2008. Grid scale effects on watershed soil erosion models. J. Hydrol. Eng. 13 (9), 793–802.

Van Der Knijff, J.M., Younis, J., De Roo, A.P.J., 2010. LISFLOOD: a GIS-based distributed model for river basin scale water balance and flood simulation. Int. J. Geogr. Inf. Sci. 24 (2), 189–212.

Vertessy, R.A., Hatton, T.J., Oshaughnessy, P.J., Jayasuriya, M.D.A., 1993. Predicting water yield from a mountain ash forest catchment using a terrain analysis based catchment model. J. Hydrol. 150 (2–4), 665–700.

Vivoni, E.R., Mascaro, G., Mniszewski, S., Fasel, P., Springer, E.P., Ivanov, V.Y., Bras, R.L., 2011. Real-world hydrologic assessment of a fully-distributed hydrological model in a parallel computing environment. J. Hydrol. 409 (1–2), 483–496.

Wang, G.S., Xia, J., Niu, C.W., 2004. Flow routing method and its application in distributed hydrological modeling. Geogr. Res. 23 (2), 175–182 (in Chinese).

Wang, H., Fu, X., Wang, G., Li, T., Gao, J., 2011. A common parallel computing framework for modeling hydrological processes of river basins. Parallel Comput. 37 (6–7), 302–315.

Wang, H., Zhou, Y., Fu, X., Gao, J., Wang, G., 2012. Maximum speedup ratio curve (MSC) in parallel computing of the binary-tree-based drainage network. Comput. Geosci. 38 (1), 127–135.

Wang, H., Fu, X., Wang, Y., Wang, G., 2013. A High-performance temporal-spatial discretization method for the parallel computing of river basins. Comput. Geosci. 58, 62–68.

Wu, Y., Li, T., Sun, L., Chen, J., 2013. Parallelization of a hydrological model using the message passing interface. Environ. Model. Softw. 43, 124–132.

Yalew, S., van Griensven, A., Ray, N., Kokoszkiewicz, L., Betrie, G.D., 2013. Distributed computation of large scale SWAT models on the Grid. Environ. Model. Softw. 41, 223–230.

Zhao, G., Bryan, B.A., King, D., Song, X., Yu, Q., 2012. Parallelization and optimization of spatial analysis for large scale environmental model data assembly. Comput. Elect. Agric. 89, 94–99.

Zhao, G., Bryan, B.A., King, D., Luo, Z., Wang, E., Bende-Michl, E., Song, X., Yu, Q., 2013. Large-scale, high-resolution agricultural systems modeling using a hybrid approach combining grid computing and parallel processing. Environ. Model. Softw. 41, 98–106.