Dynamic parallelization of hydrological model simulations

Tiejian LI [a, b], Guangqian WANG [a], Ji CHEN [b,*], Hao WANG [a]

[a] State Key Laboratory of Hydroscience and Engineering, Tsinghua University, Beijing, 100084, China

[b] Department of Civil Engineering, The University of Hong Kong, Pokfulam, Hong Kong, China

[*] Corresponding author. Tel.: +852-28592646; fax: +852-25595337.

E-mail addresses: litiejian@tsinghua.edu.cn (T. Li), dhhwgq@tsinghua.edu.cn (G. Wang), jichen@hku.hk (J. Chen), whao06@mails.tsinghua.edu.cn (H. Wang)

Resubmission to the Journal of *Environmental Modelling & Software*

ABSTRACT:

This paper introduces the development of a dynamic parallel algorithm for conducting hydrological model simulations. This new algorithm consists of a river network decomposition method and an enhanced master-slave paradigm. The decomposition method is used to divide a basin river network into a large number of subbasins, and the enhanced master-slave paradigm is adopted to realize the function of this new dynamic basin decomposition method through using the Message-Passing Interface (MPI) and C++ language. This new algorithm aims to balance computation load and then to achieve a higher speedup and efficiency of parallel computing in hydrological simulation for the river basins which are delineated by high-resolution drainage networks. This paper uses a modified binary tree codification method developed by Li et al. (2010) to code drainage networks, and the basin width function to estimate the possible maximum parallel speedup and the associated efficiency. As a case study, with a hydrological model, the Digital Yellow River Model, this new dynamic parallel algorithm is applied to the Chabagou basin in northern China. The application results reveal that the new algorithm is efficient in the dynamic dispatching of simulation tasks to computing processes, and that the parallel speedup and efficiency are comparable with the estimations made by using the basin width function.

Keywords: Basin width function; Digital drainage network; Domain decomposition; Dynamic parallelization; Master-slave paradigm; Modified binary tree codification

Software availability

Program title: MPI control

Description: A graphic user interface for the MPI-based parallel hydrological model, including the setup of parameters, the start/stop control of simulation, and the performance and progress displays of the simulation

Developer: Dr. T. Li and Prof. G. Wang

Platform: PC with Microsoft Windows

Source language: Visual C++

Program size: 2 MB

Cost: Free

Availability: Contact the developers

Program title: DWM.main

Description: The platform for parallel hydrological simulation, which dynamically decomposes a basin river network into a number of subbasins and dispatches them to slave computing processes

Developer: Dr. T. Li, H. Wang, and Prof. G. Wang

Platform: PC

Source language: C++

Program size: 1 MB

Cost: Free

Availability: Contact the developers

1. Introduction

Computation capacity has been rapidly advanced since the emergence of parallel computation techniques. Generally, parallel computing can be achieved at four levels, i.e., bit-level, instruction level, data level and task level (Culler et al., 1999). Bit-level is achieved by increasing processor word size, and instruction level is realized by using instruction pipelines and superscalar processors. Data level and task level refer to distributing the data and the execution processes (threads), respectively, across different parallel computing cores. For data and task parallelization, a program is executed simultaneously by multiple processes on a computer system. Usually, each process is conducted by one processor core. In recent years, the number of cores of a computer processor can be a few hundred, which has encouraged more extensive implementation of parallel computation.

The parallel computing can be used to shorten computation time through exploiting the concurrency of a simulation problem. The concurrency refers to that the simulation problem can be decomposed into several subtasks, which can be handled simultaneously. However, during numerical simulation, dependency usually coexists with concurrency. Dependency refers to the execution of some subtasks is subject to the accomplishment of some other subtasks. Therefore, different parallelization problems have different implementation schemes for shortening the simulation duration.

In water and environmental research fields, the algorithms for realizing parallel computing can be categorized into two types; one is for the optimization of model parameters, and the other is for the model simulation through using domain decomposition. For example, Vrugt et al. (2006) developed an optimization algorithm for application of parallel computing to stochastic parameter estimation in environmental models. Sharma et al. (2006a, b) and Muttil et al. (2007) proposed

parallel schemes for calibrating model parameters. For the usage of domain decomposition, parallel algorithms are used to partition the temporal/spatial domain into sub domains; however, because almost all natural processes are temporally successive, domain decomposition is mostly realized spatially. This paper focuses on the development of a technique used in hydrological simulations based on spatial domain decomposition.

The different forms of a basic hydrological simulation unit, such as a river reach, a hydrological response unit, a grid cell and a finite element cell, can be used by different hydrological models. A basin can be partitioned into a number of subbasins. A subbasin usually consists of a number of the basic hydrological simulation units, which are interconnected since stream water flows from a subbasin's upper reach and converge to its downstream subbasin via its outlet reach. River network connectivity is controlled by the basin topography. It is worth noting that groundwater is normally treated as following the direction of streamflow, though the simulation of the groundwater flow is more complex than that of streamflow. It can, therefore, enhance the efficiency of parallel computing remarkably to decompose a basin into a number of subbasins. This has been proved effective by several studies (e.g. Apostolopoulos and Georgakakos, 1997; Vivoni et al., 2005; Cui et al., 2005; Kolditz et al., 2007). The study of Apostolopoulos and Georgakakos (1997) proposed a general parallel algorithm to divide a drainage basin into subbasins for hydrological simulation. Vivoni et al. (2005) developed a parallel version of a distributed hydrological model, tRIBS, by using subbasin decomposition. Cui et al. (2005) parallelized a distributed hydrological model, r.water.fea, by partitioning a river basin into subbasins and distributing them to different computing processes with the control of computation load balance. Kolditz et al. (2007) used an approach to conduct parallel computing of a hydrological soil model

over subbasins.

However, the size of the subbasins and the degree of their related drainage network complexes, which affect the computation time in hydrological simulation, within a basin may vary greatly. Therefore, the control of computation load balance is challenging. In a parallel simulation, a *static* parallel algorithm refers to that one computing process conducts only one subbasin, and a *dynamic* parallel algorithm refers to that a computing process can undertake several subbasin simulations one by one. For a *static* parallel algorithm, a partition method is required to separate a basin equally into a number of subbasins. If the number of subbasins is sufficiently larger than the number of computing processes, a *dynamic* parallel algorithm can be applied, and load balance control can be achieved effectively. However, because of the complicate logic relationships among the river reaches in a drainage network, dynamic parallel algorithms for hydrological simulations have not appeared in the literature yet. This paper investigates the dynamic decomposition of a drainage network into a large number of subbasins with effective control of load balance, and then develops a dynamic parallelization algorithm for hydrological simulation.

The emergence of the parallelization of hydrological models is supported by the easy-to-use parallel programming standards, such as the Message-Passing Interface (MPI) (MPI Forum, 2008) and Open Multi-Processing (OpenMP) application program interface (Chapman et al., 2007). These programming standards make it possible to obtain a new program running in a parallel environment simply by adding functions, compiler directives and inter-process communications to the original serial program code. However, since the parallel programs and simulation models are blended, neither the frameworks nor the codes for parallelization can be reused by other models. Moreover, the simulation proceeding status using these programming standards is not

usually user-friendly. To overcome the above limitations, this paper uses the MPI and C++ language to develop a user-friendly interface for parallel computing, which is reusable by different hydrological models.

For the dynamic parallelization of hydrological simulations, the decomposition of a basin into a large number of subbasins is necessary. Therefore, an effective method for coding a basin drainage network, which can mark the logical relationships among the river reaches, is indispensable. Due to the tree-like shape of drainage networks, complex logical operation is needed to divide a drainage network into subbasins associated with different priorities, which represent the dependency in parallel simulation. Li et al. (2010) developed a modified binary-tree-based method for coding drainage networks, which enhances the efficiency of logical operations on the structure of a drainage network, and this paper uses the codification method to develop the dynamic decomposition method.

2. Dynamic decomposition of simulation tasks

2.1 Dynamic basin decomposition

A basin can be decomposed into as many subbasins as there are computing processes when a *static* parallel algorithm is used. However, for a *static* parallel algorithm, the computing processes for conducting the flow routing of downstream subbasins have to wait for the results from the related upstream subbasins, and the idle time for waiting is difficult to predict and control, which results in lower parallel efficiency. Therefore, a *dynamic* algorithm for domain decomposition is more able to gain higher parallel efficiency for hydrological simulations. This study develops a *dynamic* algorithm (see Fig. 1) for dividing a drainage network into a number of subbasins and dispatching them to each computing process. From Fig. 1, it can be seen

that a subbasin, for example subbasin 7 in the figure, can be decomposed from the whole drainage network and dispatched to an idle process, for example, computing process 1. At the last step of the simulation, the subbasin at the basin outlet, subbasin 16 (see Fig. 1), is dispatched and solely simulated.

From the example given in Fig. 1, it can be observed that there are two types of subbasins. One type is the headwater subbasins, which do not need the input data from the upstream. The other type is those subbasins which need the input data from their related upstream subbasins, and their simulation sequences and the data transferring paths must follow the routes from the upper to downstream subbasins (see the arrow lines in Fig. 1). Intuitively, to minimize the simulation time, the farthermost subbasin (e.g., subbasin 1 in Fig. 1) from the basin outlet should be simulated first.

To achieve the dynamic decomposition of a basin, the binary-tree-based structural method for coding river reaches in a drainage network developed by Li et al. (2010) is adopted in this study. This codification method treats a drainage network as a binary tree, and each river reach as a tree node. The basin outlet reach is the root node, and all the tree nodes are preferentially arranged to the left for indicating mainstem-tributary relationships. In this classification, a left node is primary, which indicates that the associated reach is the local mainstem, and the corresponding right node is the secondary reach, which is the local tributary (see Li et al. (2001) for details). Each river reach associated with its hillslopes is designated by a code with two components, denoted as (L, V). The value of component L represents the level of a node in a binary tree, which is the topological distance of the reach to the basin outlet. For the nodes with the same level of L, their components of V are numbered by a series of sequential integers from left to right, and the series starts with zero and takes vacant nodes into account. According to the way of assigning codes, for any given river reach (L, V), its

primary and secondary upper reaches are (L+1, 2V) and (L+1, 2V+1), respectively, and its downstream reach is (L-1, V\2), where "\" denotes the integer division. By such a method, a river reach, with its associated hillslopes, can be logically coded from the basin outlet to the headwaters.

With the application of the modified binary tree codification, the dynamic basin decomposition method is developed (see Fig. 2 for its flow chart). At the beginning of Fig. 2, a binary tree data structure for describing the basin drainage network is constructed in the computer memory, RAM (Random-access memory), and two simulation status flags, N_{task} and $N_{calculating}$, are initialized for each reach so as to control the scale of split-off subbasins and to indicate the upstream-downstream dependency, respectively. N_{task} of a reach indicates the number of waiting reaches (for conducting the simulation) in its contributing area, and its initial value is the number of all the contributing reaches. $N_{calculating}$ of a reach denotes the number of subbasins undertaking simulation in the reach's associated upper subbasins, and its initial value is 0.

After the initialization, Fig. 2 shows that the split demands from idle computing processes, which are the computing processes just initialized or completing a simulation task, are repeatedly checked. Once a split demand is received, the search procedure is executed to find out an available subbasin. This sequence ensures that a bunch of river reaches (i.e., an available subbasin) far from the basin outlet are split off with high priority. The flag of $N_{calculating}$ determines whether a reach is available for simulation. If $N_{calculating}$ of a reach in a subbasin is greater than 0, which indicates that some reaches in this reach's upper contributing area are still under simulation, the split off of its associated subbasin is not allowed. In addition, the scale of a split-off subbasin is controlled by the comparison of the flag N_{task} with two parameters, namely N_{min} and N_{max} (see Fig. 2). How to determine these two parameter values is discussed in Section

3. When $N_{calculating}$ of a reach is zero and its flag N_{task} is between N_{min} and N_{max}, the reach, with some of its associated upper reaches, can be grouped as a subbasin, which is available to split off from the river network for simulation.

From Fig. 2, it can be observed that once an available subbasin is detected, it is disconnected from the binary tree data structure and converted to an array, where the reaches are arranged in the order of calculation, namely the descending order of the component L of the binary-tree-based river code. Meanwhile, the N_{task} and $N_{calculating}$ of each reach which is downstream of the split-off subbasin are updated; the value of the N_{task} of each reach is subtracted by the number of reaches in the split-off subbasin, and the value of the related $N_{calculating}$ is increased by 1. When the simulation of this subbasin is complete, the $N_{calculating}$ of each of downstream reaches is subtracted by 1, which makes those downstream reaches available for simulation later on. In the above algorithm, the movement of the search cursor is efficient, since the topology of the drainage network is represented by the binary tree data structure (see Li et al. (2010) for details).

2.2 Task scheduling and data transfer

The new algorithm for decomposing a drainage network is dynamic, and a master-slave paradigm (see Fig. 3) introduced by Li et al. (2006) for task scheduling is adopted to fulfill the functions of the new algorithm. The master process takes charge of basin decomposition and task dispatch, and the slave computing processes only conduct hydrological simulation. This study enhances the master-slave paradigm by adding a data transfer process, which temporarily stores intermediate simulation results on the interface of subbasins in RAM and then passes them to slave processes when requested. Therefore, there are three types of processes in the new parallel paradigm, namely, the master process, slave process and data transfer process. Moreover, a relational database

server is included (see Fig. 3), which is used to manage input and output data in hydrological simulations. With the database server, high efficient manipulation of hydrological simulation data can be exploited, and simulation results from different slave computing processes can be saved into the database.

Fig. 3 shows the flow chart of three types of processes. The master process decomposes the drainage network after receiving a split demand from a slave process (see Fig. 2). The master process controls the slave processes through receiving the split demands and dispatching the available subbasin reach codes. Receiving a completion message from a slave process, the master process updates the status flag $N_{calculating}$ and returns to check the next split command.

In Fig. 3, the kernel of all the multiple slave computing processes is a hydrological model. After the initialization or the completion of a simulation task, a slave computing process sends a split demand to the master process, and then receives the reach codes of a subbasin. Accordingly, if the reach codes are valid, the slave process communicates with the data transfer process to obtain simulated inflow results of the subbasin, and reads the reach parameters of the hydrological model from the database server. Then, the slave process conducts the hydrological simulation, and transfers the simulation results to the database. After that, the slave process sends a completion message to the master process, and sends the simulation results (e.g., runoff) of the outlet reach of the subbasin to the data transfer process (i.e., inflow of its downstream subbasin). Afterwards, the slave process issues a new split demand to the master process for conducting the next task. Upon receiving a void reach code, the slave process quits parallel computing.

In parallel simulation, a single slave process does not conduct the simulation for the whole drainage network. Nevertheless, the simulation procedure is driven by the

dialog between the master process and slave processes through the iterative loop of request—split—new request—new split. When the simulation procedure approaches the basin outlet, the number of the slave processes used in the simulation gradually decreases because of the decreasing number of available subbasins, and the last slave process, which is the key slave process mainly for determining the total simulation duration, simulates the subbasin at the outlet (e.g., subbasin 16 in Fig. 1).

The basin decomposition dynamically divides the whole drainage network into a number of subbasins, resulting in the necessity of transferring the intermediate results from an upstream subbasin to its connected downstream subbasin. Once the simulation of a subbasin is completed, the simulation results of the outlet reach of the subbasin are transferred to its next downstream subbasin. However, during parallel computation, this downstream subbasin usually has not been decomposed and dispatched to a slave computing process, and, therefore, those intermediate simulation results are temporarily stored in the RAM of the data transfer process until they are requested by its downstream subbasin (see Fig. 3). Consequently, the logical connections among split-off subbasins in the proposed parallel algorithm can be achieved dynamically and efficiently.

3. Estimation of parallel simulation performance

Compared with *static* parallel algorithms (e.g. Vivoni et al., 2005; Kolditz et al., 2007), the new algorithm for dynamic decomposition of a drainage network ensures the flexibility and scalability of parallel hydrological simulation. Furthermore, to better make use of the concurrency for parallel hydrological simulation, it is necessary to optimize the parameters of N_{min} and N_{max} for obtaining a suitable split-off subbasin size. To this end, this study develops a method for estimating the performance of the new parallel algorithm through using the basin width function (Veitzer and Gupta, 2001).

The performance of a parallel algorithm is generally evaluated by two measurements, namely speedup and efficiency (Scott et al., 2005). The speedup, S_p, of an algorithm, which reflects the speed advantage of using a parallel algorithm, is defined as follows:

$$S_p=T_1/T_p \quad (1)$$

where T_1 and T_p are the simulation durations obtained by using one and p computing processes, respectively. The efficiency, E_p, measures the fraction of time that a computing process is effectively used and is calculated below:

$$E_p=S_p/p \quad (2)$$

It is expected that a parallel simulation can be finished promptly with minimization of wastage of computation resources, and will also maximize the speedup to result in higher efficiency. However, there is a tradeoff between speedup and efficiency; normally, in parallel simulation, the higher the speedup the lower the efficiency. Therefore, the optimization of parallel simulation is necessary.

In this study, to evaluate the speedup and efficiency of the new algorithm, three conditions in hydrological simulation should be clarified. The first is that the resolution of a drainage network used in the hydrological simulation is determined by the resolution of terrain data (e.g., digital elevation model (DEM) data) and the simulation objectives. Therefore, the total number of the river reaches in a drainage network, denoted as n, is fixed, and the hydrological simulation is conducted over each river reach, which is denoted as a basic simulation unit (or a minimum subbasin).

The second condition is that a coarse drainage network should be extracted from the same terrain dataset. Using the dynamic basin decomposition, a number of subbasins are obtained from the simulated drainage network. This coarse drainage network approximately represents the topological relationship of these dynamic split-

off subbasins, and then the estimation of parallel speedup and efficiency can be performed. The resolution of this coarse network is based on the two parameters N_{min} and N_{max} (see subsection 2.1 and Fig. 2), which determine the number of split-off subbasins. Fig. 4(a) shows the drainage network of the Chabagou basin in northern China, which consists of 4912 reaches (Li et al., 2010) and is used to conduct hydrological simulation. Fig. 4(b) displays the relevant coarse drainage network of the same basin with 171 river reaches to represent the topology of the subbasins split off from Fig. 4(a) for conducting hydrological simulation, when N_{min} and N_{max} are set as 17 and 35, respectively.

The third condition is that the number of slave computing processes, p, is determined by the capacity of computer hardware. Generally one processor core executes one computing process, and then the available p cannot be larger than the total number of available processor cores.

The tradeoff between speedup and efficiency is mainly determined by the number of computing processes and the number of split-off subbasins. With the given number of computing processes, if the number of subbasins simulated by each slave computing process is large, the total simulation duration will decrease, which leads to high parallel efficiency. However, if the number of subbasins is too large, the time used for transferring message and data among slave processes, the data transfer process and the database server (see Fig. 3) will increase, resulting in lower parallel efficiency. Therefore, it is critical to choose the appropriate scale of subbasins in parallel hydrological simulation, which is mainly determined by N_{min} and N_{max}. If the average number of river reaches in all the subbasins is n_{avg}, which is between N_{min} and N_{max}, the total number of split-off subbasins, N, of a study basin is computed as below:

$$N=n/n_{avg} \tag{3}$$

In parallel simulation, one of the slave computing processes conducts simulation over the subbasin with the basin outlet reach, and this slave process is named as the key slave process, whose computation time is normally the longest. Therefore, the working time of the key slave process can be approximated as the simulation duration T_p, and the equation of computing T_p can be represented as follows:

$$T_p=(t_{cal}n_{avg}+t_{comm})N_d \quad (4)$$

where t_{cal} is the average time to run the hydrological model over one river reach, including obtaining basic information from the database server, conducting simulation, and saving the required results to the database when necessary (see Fig. 3). N_d is the number of subbasins handled by the key slave computing process. t_{comm} is the time for transferring (i.e., sending and receiving) simulation results for one river reach between a slave process and the data transport process. The last river reach of each subbasin is the boundary unit, and its simulation results need to transfer to the downstream. Therefore, the average communication time for one subbasin can be noted as t_{comm}.

If the simulation for a whole drainage network is conducted by one computing process without any inter computing process communication (i.e., data transfer), the simulation time is determined by the sequential calculation of all the n river reaches as follows:

$$T_1=t_{cal}n \quad (5)$$

Therefore, by substituting Equations (4) and (5) into (1) and (2), the speedup and efficiency of the new parallel algorithm can be calculated as follows:

$$S_p = \frac{t_{cal}n}{(t_{cal}n_{avg}+t_{comm})N_d} = \frac{1}{1+\frac{t_{comm}}{t_{cal}}\frac{1}{n_{avg}}}\cdot\frac{N}{N_d} \quad (6)$$

$$E_p = \frac{1}{1+\frac{t_{comm}}{t_{cal}}\frac{1}{n_{avg}}}\cdot\frac{N}{N_d p} \quad (7)$$

Due to the rather complicated nature of hydrological models, it can be assumed that the time for data transfer is much less than the average time required for the hydrological simulation, namely $t_{comm} \ll t_{cal}$. Furthermore, n_{avg} can be relatively large, and then the value of $t_{comm}/t_{cal}/n_{avg}$ can be assumed to be 0, which implies that the influence of communication time on the speedup and parallel efficiency can be negligible. However, it is worth noting that when the scale of subbasins n_{avg} is too small and the number of slave processes p is relatively large, the relationship of $t_{cal}n_{avg} < t_{comm}(p\text{-}1)$ can happen. Then, when a slave process finishes one subbasin simulation task and wants to communicate with the data transfer process, it is highly likely that the latter is busy in finishing a round of communication with all the other slave processes. Therefore, T_p will be determined by the consumption of the data transfer process instead of the key slave process, which will result in relatively lower speedup and efficiency. Consequently, the relationship of $t_{cal}n_{avg} < t_{comm}(p\text{-}1)$ should be avoided.

In the right-hand side of Equations (6) and (7), p is known, and N can be determined by N_{min} and N_{max} (though this is not explicit). In practice, N is estimated by a trial parallel computation for a short hydrological simulation duration (e.g. 1 hour) over a drainage network, and then the related coarse drainage network with N river reaches, which represent N split-off subbasins from the simulated drainage network, can be obtained. Nevertheless, the value of N_d in Equations (6) and (7) is difficult to estimate, since it is influenced not only by the topologic structure of a drainage network (the dependencies among subbasins) but also by the procedure of dynamic dispatching subbasins to slave computing processes in the parallel simulation.

In this study, the width function (Veitzer and Gupta, 2001) of the coarse drainage network is used to estimate N_d. The width function represents the relationship between the number of river reaches, $W(r)$, and their logical distance, r, (one logical distance is

one river reach) to the basin outlet. For example, Fig. 5 shows the width function of the coarse drainage network of the Chabagou watershed (Fig. 4(b)). From Fig. 5, it can be found that the maximum r, r_{max}, is 32, and, furthermore, the total sum of $W(r)$ from r = 1 to 32 is 171, which indicates that N in Fig. 4(b) is 171.

From the topological width function (see Fig. 5), it can be learned that no matter how many slave computing processes are used, the mainstem subbasins have to be simulated (normally by the key slave process) from the upper reach to downstream one by one. Consequently, the number of subbasins handled by the key slave process should not be smaller than r_{max}, namely $N_d \geq r_{max}$. Then, from Equation (6), the following relationship can be derived:

$$S_p \leq \frac{1}{1+\frac{t_{comm}}{t_{cal}}\frac{1}{n_{avg}}} \cdot \frac{N}{r_{max}} \cong \frac{N}{r_{max}} \tag{8}$$

Accordingly, in Fig. 5, for N=171, r_{max}=32, using Equation (8), the maximum speedup for the Chabagou watershed basin equals 5.3.

To maximize the parallel efficiency, E_p, for a given p, the S_p should remain near its maximum value N/r_{max}. Therefore, the minimum number of slave processes, p, which permits $N_d=r_{max}$, can be determined. In Fig. 5, a straight line representing the number of slave processes, p, is added. This straight line intersects with the width function, and, from Fig. 5, it can be observed that areas between this straight line and the width function are denoted from the right to the left as A_i (one below the straight line) and A'_i (the corresponding one above the straight line) (i = 1, …, m), and A_{m+1}. It is worth noting that if two (or several) consecutive areas are all below (or above) the p straight line, they are counted as one area; therefore, the distribution of the areas of A_i and A'_i alternate below and above the straight line. When the p slave computing processes are used to conduct the simulation from the upper to downstream subbasins following the

descending order of r (i.e., from right to left of x axis in Fig. 5), the areas under the straight line would represent the remaining computing capacity, and the areas above the line represent the task loads beyond the computing capacity. If each task load beyond the computing capacity (the area above the line) can be handled in advance by the remaining computing capacity (the area below the line) to its right in Fig. 5, the minimum value of N_d, which equals r_{max}, can be maintained. Therefore, if the value of p is large enough, the following condition can be met:

$$\sum_{j=1}^{i} A'_j \le \sum_{j=1}^{i} A_j \text{ , for each } i \text{ from 1 to } m \tag{9}$$

where m denotes the number of areas above the p straight line. The minimum value of p, which meets the above condition, can deliver the parallel computation with the maximum speedup and the corresponding highest efficiency for a given N, and is the optimized p. For example, in Fig. 5, for N=171, the optimized p, noted as p_{opt}, equals 7.

For a given N, when the speedup is maximized and the number of slave processes is optimized, the parallel efficiency can be estimated as follows:

$$E_{p,opt} = \frac{S_{p,opt}}{p_{opt}} \cong \frac{N}{r_{max} p_{opt}} \tag{10}$$

where the subscript of *opt* denotes the optimized values. For example, in Fig. 5, $E_{p,opt}$ reaches 5.3/7=0.76. Furthermore, the variables in the right-hand side of Equation (10) can be expressed by the areas in Fig. 5, namely $N = \sum_{r=1}^{r_{max}} W(r)$ and $r_{\max} p = \sum_{r=1}^{r_{max}} W(r) + \sum_{j=1}^{m+1} A_j - \sum_{j=1}^{m} A'_j$, and the parallel efficiency can be expressed as follows:

$$E_{p,opt} \cong \frac{\sum_{r=1}^{r_{max}} W(r)}{\sum_{r=1}^{r_{max}} W(r) + \sum_{j=1}^{m+1} A_j - \sum_{j=1}^{m} A'_j} \quad (11)$$

4. Software realization

According to the new parallel algorithm, the software realization is conducted by using the MPI standard (MPI Forum, 2008) to achieve point to point message-passing among all the processes. There are several cost-free implementations of the MPI standard which are supported by different hardware and software platforms. The MPI parallel environment is highly flexible and can be implemented not only on high performance clusters, but also efficiently on cost-effective clusters composed by personal computers. In this study, the MPICH2 (Gropp et al., 2008) and C++ language are used to develop the program of DWM.main that conducts the activities of decomposing a drainage network and scheduling simulation tasks. DWM.main utilizes the MPI functions for the communication and coordination among computing processes, and is compiled as a console program running in Microsoft Windows. Moreover, several Windows application program interface functions are used to acquire indicators of the performance of simulation.

DWM.main is designed as an object-oriented model. Namely, each river reach and its associated hillslopes are defined as a basic computation unit (coded as a C++ class). Therefore, the parallelization, including basin decomposition and task scheduling, will not affect the properties and simulations of the river reaches. A hydrological model is generally defined as a set of responses of a river reach, and different hydrological models can be integrated with DWM.main. Therefore, DWM.main and MPI control (see the next paragraph for details) can be reusable and constitute a parallel platform for digital watershed modeling. For example, a set of modules developed to simulate the

rainfall-runoff, soil erosion and sediment transport in the Yellow River basin, which is the Digital Yellow River Model (DYRIM) (Wang et al., 2007), has been integrated with the DWM.main.

Furthermore, to visualize the control of the procedure of parallel computing, a graphical user interface, named MPI control, is developed. Fig. 6 shows the schematic of the MPI control. Parallel programs using the MPI functions are all initiated by the mpiexec command (see Fig. 6(a)), which is a console program for displaying the proceeding texts of the parallel program of DWM.main without the need for graphical interfaces (Fig. 6(b) and (c)).

The main function of the MPI control is to convert the proceeding texts to graphical elements for displaying the simulation status (Fig. 6(d)). The proceeding texts arranged in predefined formats are transferred from the console to the MPI control, and then the text messages are interpreted by two categories of the predefined formats, the performance of hosts and the progress of simulation. Performance indicators displayed by the MPI control include the usage of resources (i.e., processor, memory and network) of hosts, time consumption of hydrological simulation, database access and communication and estimated remaining execution time. The progress of simulation is dynamically displayed in a map of the simulated drainage network. In this map, calculated, calculating and uncalculated river reaches are displayed in different colors, and the reaches calculated by different computing processes are also distinguished by different colors (Fig. 6(e)).

5. Application

The new parallel algorithm and related software developed in this paper are applied to the Chabagou River basin (see Li et al. (2010) for details). With an area of 205 km^2, the Chabagou River basin is located in Shaanxi Province in northern China

and it flows into the Wuding River, a tributary of the middle Yellow River (Huanghe in Chinese). The elevation of the study basin ranges from 920 m to 1300 m and the landform is the typical rolling Loess Plateau. Numerous gullies and channels incise the basin area, and, thus, a high-resolution basin delineation is needed for distinguishing hillslopes and gullies. In this application, a drainage network, with 4912 river reaches and a mean hillslope area of 1.67 hectare, is extracted from 50 m × 50 m DEM data (see Fig. 4(a)). Furthermore, floods and soil erosion in this basin are mainly caused by short duration and high intensity torrential rains, and therefore the transport of water and sediment are highly unsteady. Consequently, a high temporal resolution is also needed, and a large amount of computation is required to conduct the simulation of hydrological responses, soil erosion and sediment transport in the Chabagou basin.

Parallel applications of the DYRIM in the Chabagou basin by using DWM.main and MPI control have been conducted for several cases over cost-effective computer clusters (see Wang et al. (2007) and Li et al. (2010) for the hydrological simulation results). In this paper, the simulation of the whole of August in 1967 is chosen to analyze the computation time and parallel speedup, while the whole procedure of calibration, validation and application and the hydrological and sediment results can be found in Li (2008). The time step adopted in the simulation is 6 minutes, and there are 7440 time steps for 31 days in the August.

In this study, the cost-effective cluster is composed by using a 4-way AMD Opteron server and several Intel Core multi-core CPU personal computers, which run the Microsoft Windows operating system and are interconnected by 100 Mbps switch Ethernet. A Hewlett-Packard rx2600 server is used as the database server running the UNIX operating system and the Oracle 9i database system. In this application, at most

14 slave computing processes (one core for each process) are used to examine the performance of the parallel simulation.

The parameters, N_{min} and N_{max}, used in the new parallel algorithm are given in Section 3, and the corresponding coarse river network of the Chabagou basin is shown in Fig. 4(b). Table 1 gives the summary of the parallel speedup, efficiency and time consumption from parallel simulations through using the different numbers of slave computing processes. From Table 1, it can be observed that, along with the increase of the number of slave computing processes, the parallel efficiency generally decreases; if p is less than 13, the simulation duration, T_p, effectively decreases and the speedup increases steadily.

Fig. 7 illustrates the total computation capacity (i.e., T_p*p), and the different portions of computer time (see Table 1 also). The total exploited computer time is the sum of calculation time, database access time, and communication time. The difference between the total computation capacity and the total exploited time is the total idle time of all the slave computing processes. The ratio of the total idle time to the total computation capacity (i.e., $(T_p*p-(t_{cal}n_{avg}N_d+t_{comm}N_d))/(T_p*p)$) indicates the time fraction wasted by those slave processes which are quit simulation before the key slave process completes its last simulation task, which is named as the idle time portion herein. Fig. 8 shows the parallel efficiency of the simulation and the idle time portion of slave processes. In Fig. 8, the difference between 1.0 and the real efficiency can be separated into two parts by the inverted idle time portion line. The upper part is due to the idle time, and the lower part is caused by the increase of time used in calculation, database access and communication.

6. Discussion

It is worth noting that the application results of the speedup and efficiency given in

Table 1, Figs. 7 and 8 in Section 5 do not exactly match the analytical results obtained by using Equations (8) and (10) in Section 3. Also, in Table 1, the estimated highest speedup, i.e. 5.3, is obtained when p is 13 instead of 7 (see Section 3). The main reason causing the difference between the application in Section 5 and the analysis in Section 3 is that during dynamic parallelization the key slave computing process does not usually conduct the simulation for all of those subbasins along the river mainstem which result in the largest logical distance, r_{max}. In addition, for the different numbers of slave processes, the time for accessing the database and for hydrological simulation for each river reach, t_{cal}, is not generally equal. Along with the increasing number of slave processes, t_{cal} will increase (see Table 1), mainly due to two reasons. One is the heavier executing load imposed on the database server (leading to the increasing values in Column 5 of Table 1), and the other is the bottlenecks of simultaneous operation of disk and network on personal computers (leading to the increasing calculation time (see Column 4 of Table 1)).

Nevertheless, the application results in Section 5 (see Table 1, Figs. 7 and 8) prove that the analytical results in Section 3 are valuable. From Section 3, a clear picture of the effects of applying the new parallel algorithm for hydrological simulations can be established. According to the simulation objectives, the high-resolution DEM data and the number of available computing processes, a coarse river network can be obtained by optimizing the parallel parameters, N_{min} and N_{max}. Meanwhile, with the basin width function, the optimized number of slave computing processes, p_{opt}, can be obtained, which can be used to evaluate the possible speedup and efficiency in the parallel simulation.

In Section 3, from Equations (8), (9) and (11), it can be observed that for a given N the number of p_{opt} can be evaluated by the shape of the topological width function of

the corresponding coarse drainage network. If the number of available slave processes is smaller than p_{opt}, the speedup can not reach the maximum value (see Table 1), and the efficiency can not be estimated by Equation (10). Nevertheless, since the remaining computation capacity (i.e., the areas below a smaller p straight line in Fig. 5) becomes smaller, the new dynamic parallel algorithm can balance the computation loads among slave processes more efficiently, and a larger efficiency of parallel computation can be obtained (see Table 1 and Fig. 8). Therefore, for a given N, to efficiently use the concurrency in hydrological simulation, if the number of available computing processes is larger than the current p_{opt}, the values of two parameters, N_{min} and N_{max}, should be reset smaller to obtain another coarse drainage network which would have a larger value of N. Then, the maximum available p can be the p_{opt} for the new coarse drainage network, and as a result the computation capacity can be effectively used. Therefore, in practice, following the above procedure, when the number of maximum available slave processes is known, an appropriate coarse drainage network and related topological width function can be obtained by the trial and error method until the maximum available p is close to or slightly less than the p_{opt}.

For the new parallel algorithm, both the high speedup and efficiency can be achieved by using an appropriate subbasin sizes for the given number of slave processes. For example, in the application (see Section 5), if the available number of processor cores used for simulation is assumed as 8, basically, 7 of them can be used for slave computing processes. By the trial and error method, when N_{min} and N_{max} are set as 17 and 35, respectively, for the Chabagou watershed, p_{opt} is 7 obtained by using Equation (9) (see Section 3). Then, the speedup and efficiency of 4.24 and 0.606 (see Table 1) can be achieved. Comparing with the maximum speedup of 5.34 when p equals 13 (see Table 1), the speedup for the p_{opt} (i.e., 7) obtained by using the width

function is 4.24, which is -20.6% lower than 5.34. However, the efficiency of using p=7 is 0.606, which is 47.4% higher than that (i.e., 0.411) of using p=13 (see Table 1). It can be concluded that the analytically estimated p_{opt} is acceptable in real application.

7. Conclusions

This paper developed a new parallel algorithm by using the method of dynamic basin decomposition, which can divide a drainage network into a large number of subbasins. This new parallel algorithm provides effective and efficient technical support for hydrological simulation on computer clusters. Furthermore, an analytical method is proposed to estimate the parallel speedup and efficiency. Based on the topological width function for drainage basins, this analytical method can be used to determine the maximum speedup, the optimal number of the parallel computing processes and related efficiency.

To fulfill the functions of the new parallel algorithm, an enhanced master-slave paradigm, which includes the data transfer process, is proposed in this paper. The Message-Passing Interface (MPI) standard is used to program the proposed paradigm, and the main program, DWM.main, is produced. Then, a graphical user interface, MPI control, is also developed to make the parallel algorithm more user-friendly. These two programs are designed to be reusable by different hydrological models. In this study, both the programs are integrated with the Digital Yellow River Model (DYRIM), and an application is conducted on a cost-effective computer cluster to investigate the speedup and efficiency of the new parallel algorithm over the Chabagou basin in northern China.

The value of the new developed parallel algorithm may be evidenced in other natural science modeling, especially in the filed of hydro-meteorology. One of the challenges in hydro-meteorology is the demand of increasing computing capacity for

making real-time weather and hydrological forecasts. This demand is mainly due to the fact that the real-time forecast models are run for large river basins with increasing temporal and spatial resolutions and are forced by several weather predictions (ensemble forecasts) (e.g., Cloke and Pappenberger, 2009; Van den Bergh and Roulin, 2010; Brown et al., 2010). Model simulations over a large river basin with finer spatial resolutions can exploit more effective parallel computing by using the domain decomposition. Ensemble weather and hydrological forecasts need repeat operations of multiple models, different weather inputs, initial conditions and parameters for generating multiple predictions. Therefore, the newly developed parallel algorithm will be an efficient solution to this challenge in the field of hydro-meteorology.

Acknowledgement

This research was supported by the National Key Basic Research Program of China (Grant No. 2007CB714100), the Non-profit Fund Program of the Ministry of Water Resources of China (Grant No. 200901016), and the Open Research Fund Program of State Key Laboratory of Hydroscience and Engineering (Grant No. sklhse-2008-A-02). The authors are grateful for the constructive comments and suggestions from two anonymous reviewers and the Editor, Dr. Rizzoli.

References

Apostolopoulos, T.K., Georgakakos, K.P., 1997. Parallel computation for streamflow prediction with distributed hydrologic models. Journal of Hydrology 197, 1-24.

Brown, J.D., Demargne, J., Seo, D.-J., Liu, Y., 2010. The Ensemble Verification System (EVS): A software tool for verifying ensemble forecasts of hydrometeorological and hydrologic variables at discrete locations. Environmental Modelling & Software 25(7), 854-872, doi: 10.1016/j.envsoft.2010.01.009.

Chapman, B., Jost, G., van der Pas, R., 2007. In: Using OpenMP, Portable Shared Memory Parallel Programming. The MIT Press, Massachusetts, USA, 353 pp.

Cloke, H.L., Pappenberger, F., 2009. Ensemble flood forecasting: a review. Journal of Hydrology 375, 613-626, doi:10.1016/j.jhydrol.2009.06.005.

Gropp, W., Lusk, E., Ashton, D., Balaji, P., Buntinas, D., Butler, R., Chan, A., Goodell, D., Krishna, J., Mercier, G., Ross, R., Thakur, R., Toonen, B., 2008. MPICH2 User's Guide Version 1.0.8. Mathematics and Computer Science Division, Argonne National Laboratory. <http://www.mcs.anl.gov/research/projects/mpich2/documentation/files/mpich2-1.0.8-userguide.pdf>.

Cui, Z., Vieux, B.E., Neeman, H., Moreda, F., 2005. Parallelisation of a distributed hydrologic model. International Journal of Computer Applications in Technology 22(1), 42-52.

Culler, D.E., Singh, J.P., Gupta, A., 1999. Parallel Computer Architecture - A Hardware/Software Approach. Morgan Kaufmann Publishers, San Francisco, USA, 1025pp.

Kolditz, O., Du, Y., Burger, C., Delfs, J., Kuntz, D., Beinhorn, M., Hess, M., Wang, W., Grift, B., Stroet, C., 2007. Development of a regional hydrologic soil model and

application to the Beerzee—Reusel drainage basin. Environmental Pollution 148, 855-866.

Li, T., 2008. Mechanism and simulation of river basin sediment dynamics. Ph.D. Dissertation, Tsinghua University, Beijing, China, 150pp.

Li, T., Liu, J., He, Y., Wang, G., 2006. Application of cluster computing in the Digital Watershed Model. Advances in Water Science 17(6), 841-846. (in Chinese)

Li, T., Wang, G., Chen, J., 2010. A modified binary tree codification of drainage networks to support complex hydrological models. Computers & Geosciences 36(11), 1427-1435. doi:10.1016/j.cageo.2010.04.009.

Message Passing Interface Forum, 2008. MPI: A Message-Passing Interface Standard. Version 2.1. 586 pp.
<http://www.mpi-forum.org/docs/mpi21-report.pdf>.

Muttil, N., Liong, S.Y., Nesterov, O., 2007. A Parallel Shuffled Complex Evolution Model Calibrating Algorithm to Reduce Computational Time. In: Oxley, L. and Kulasiri, D. (Eds), MODSIM 2007 International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand, Christchurch, New Zealand, pp. 1940-1946.
<http://mssanz.org.au/MODSIM07/papers/33_s48/AParallelShuffled_s48_Muttil_.pdf>

Scott, L.R., Clark, T., Bagheri, B., 2005. Scientific Parallel Computing. Princeton University Press, New Jersey, USA, 374pp.

Sharma, V., Swayne, D.A., Lam, D., Schertzer, W., 2006a. Parallel shuffled complex evolution algorithm for calibration of hydrological models. In: IEEE Computer Society (Eds.), Proceedings of the 20th International Symposium on High-Performance Computing in an Advanced Collaborative Environment. IEEE Press, Washington DC, USA, pp 30.

<http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01628221>

Sharma, V., Swayne, D., Lam, D., Schertzer, W., 2006b. Auto-calibration of hydrological models using high performance computing. In: Proceedings of the 3rd Biennial meeting of the International Environmental Modelling and Software Society. Vermont, USA, 6 pp. <http://www.iemss.org/iemss2006/papers/s5/202_sharma_02.pdf >

Van den Bergh, J., Roulin, E., 2010. Hydrological ensemble prediction and verification for the Meuse and Scheldt basins. Atmospheric Science Letters 11(2), 64-71, doi:10.1002/asl.250.

Veitzer, S.A., Gupta, V.K., 2001. Statistical self-similarity of width function maxima with implications to floods. Advances in Water Resources 24, 955–965.

Vivoni, E.R., Mniszewski, S., Fasel, P., Springer, E.S., Ivanov, V.Y., Bras, R.L., 2005. Parallelization of a fully-distributed hydrologic model using sub-basin partitioning. American Geophysical Union 2005 Fall Conference, San Francisco, CA, USA. <http://vivoni.asu.edu/pdf/AGU2005parallel.pdf>

Vrugt, J.A., Nuallain, B.O., Robinson, B.A., Bouten, W., Dekker, S.C., Sloot, P.M.A., 2006. Application of parallel computing to stochastic parameter estimation in environmental models. Computers & Geosciences 32, 1139-1155, doi:10.1016/j.cageo.2005.10.015.

Wang, G., Wu, B., Li, T., 2007. Digital Yellow River Model. Journal of Hydro-environment Research 1(1), 1-11, doi:10.1016/j.jher.2007.03.001.

Table 1

Time consumption, speedup, and efficiency in using the different numbers of slave computing processes in parallel simulation for the Chabagou watershed in northern China (see Fig. 4).

Number of slave processes p	Number of subbasins N	Simulation time (s) T_p	Cumulative serial times (s) for			Speedup S_p	Efficiency E_p
			Calculation $t_{cal}n_{avg}N_d$	Database access	Communication $t_{comm}N_d$		
1	166	1530	465	1063	1.6	1.00	0.999
2	171	800	318	1238	1.9	1.91	0.955
3	171	630	317	1509	5.1	2.43	0.809
4	175	516	353	1572	7.4	2.96	0.740
5	174	452	351	1706	7.6	3.38	0.676
6	173	404	375	1739	6.8	3.78	0.630
7	171	360	389	1693	6.4	4.24	0.606
8	172	348	428	1714	7.0	4.39	0.549
9	172	308	422	1726	11.6	4.96	0.551
10	172	303	444	1773	10.9	5.04	0.504
11	172	292	458	1795	34.2	5.23	0.476
12	172	290	471	1799	27.0	5.27	0.439
13	172	286	490	1867	12.6	5.34	0.411
14	171	320	496	1899	21.2	4.78	0.341

Figure Captions

Fig. 1. The diagram of dynamic decomposition of a drainage network, where the subbasins with the boundary line colors of brown, green and pink are dispatched to the computing processes 1, 2 and 3, respectively.

Fig. 2. The flowchart for dynamic decomposition of a basin.

Fig. 3. Flowchart of the execution of master, slave and data transfer processes, in which the bold arrow lines denote the transfer of message and/or data.

Fig. 4. The drainage network of the Chabagou watershed in northern China with the (a) high and (b) coarse spatial resolutions.

Fig. 5. The topological width function, which is derived from a corresponding coarse resolution drainage network (see Fig. 4(b)) and is used to reflect the inter connection of subbasins. The straight line reflecting the number of p slave processes.

Fig. 6. Schematic of the realization of the simulation monitor with graphical user interface, MPI control. The passes of commands and messages are: a) the GUI sends a mpiexec command to initiate the MPI running environment, b) the mpiexec command starts the DWM.main program in multiple processes, c) messages from DWM.main processes are gathered by mpiexec and written in the Windows command console, d) messages in the command console are passed to the GUI via an anonymous pipe, and e) Messages are interpreted so as to draw the chart and map to show the performance and progress of the simulation.

Fig. 7. Different portions of computer time (see Table 1) and the value of the total computation capacity (i.e., T_p*p) for the different number of slave computing processes.

Fig. 8. Parallel simulation efficiency and the idle time portion of slave computing processes (see Table 1).

Figure1

Click here to download high resolution image

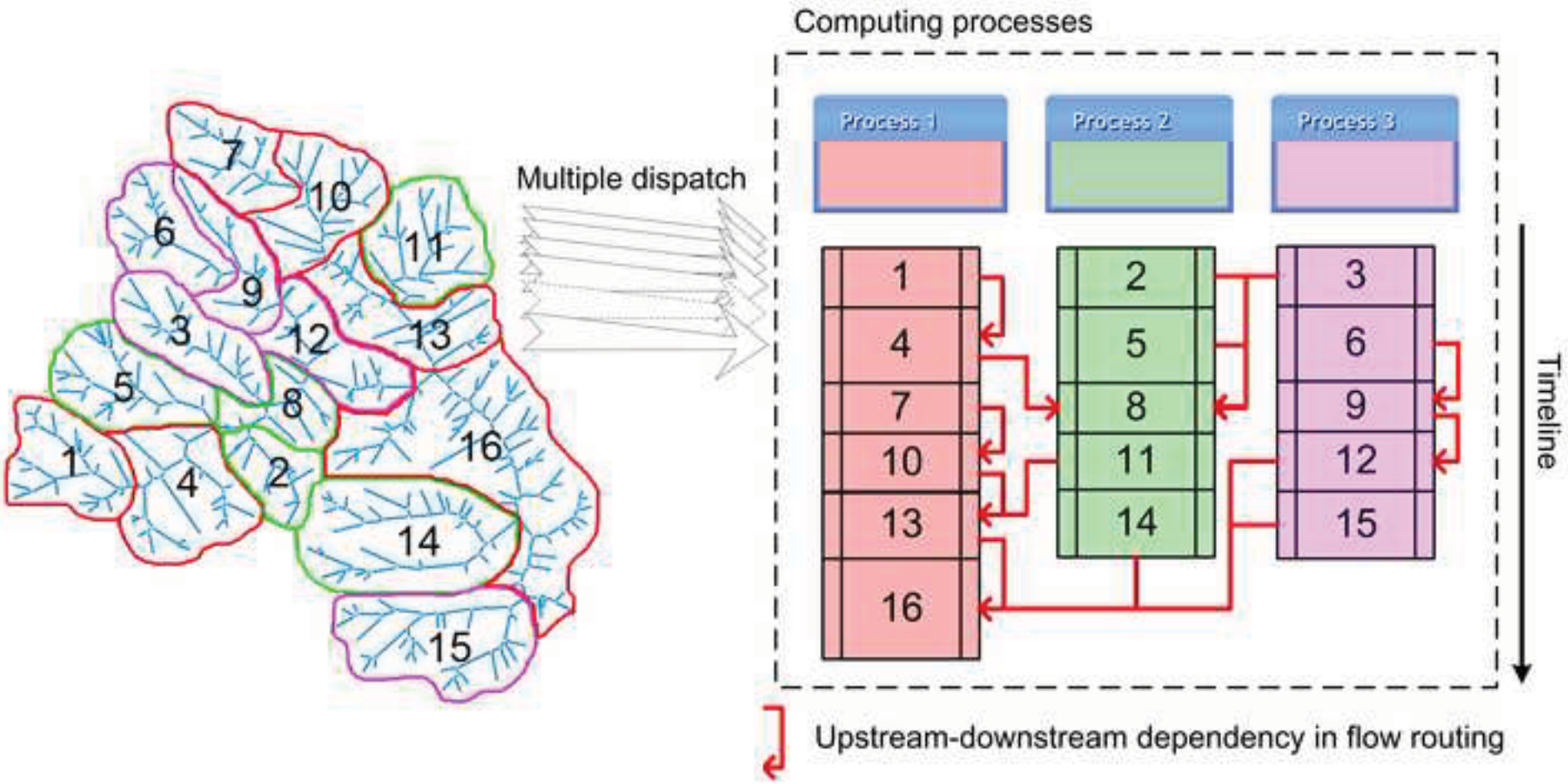

Figure2

Click here to download high resolution image

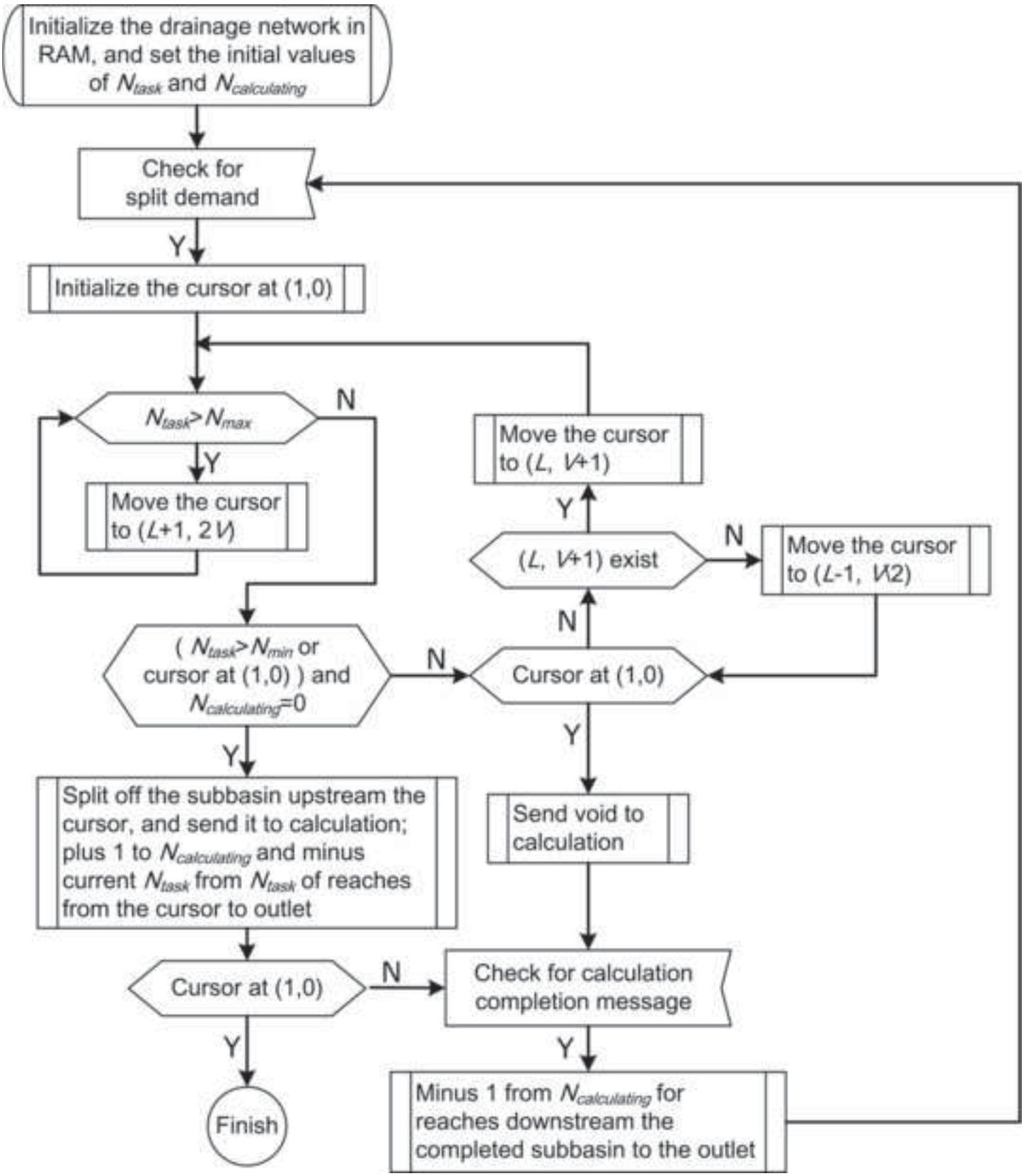

Figure3

Click here to download high resolution image

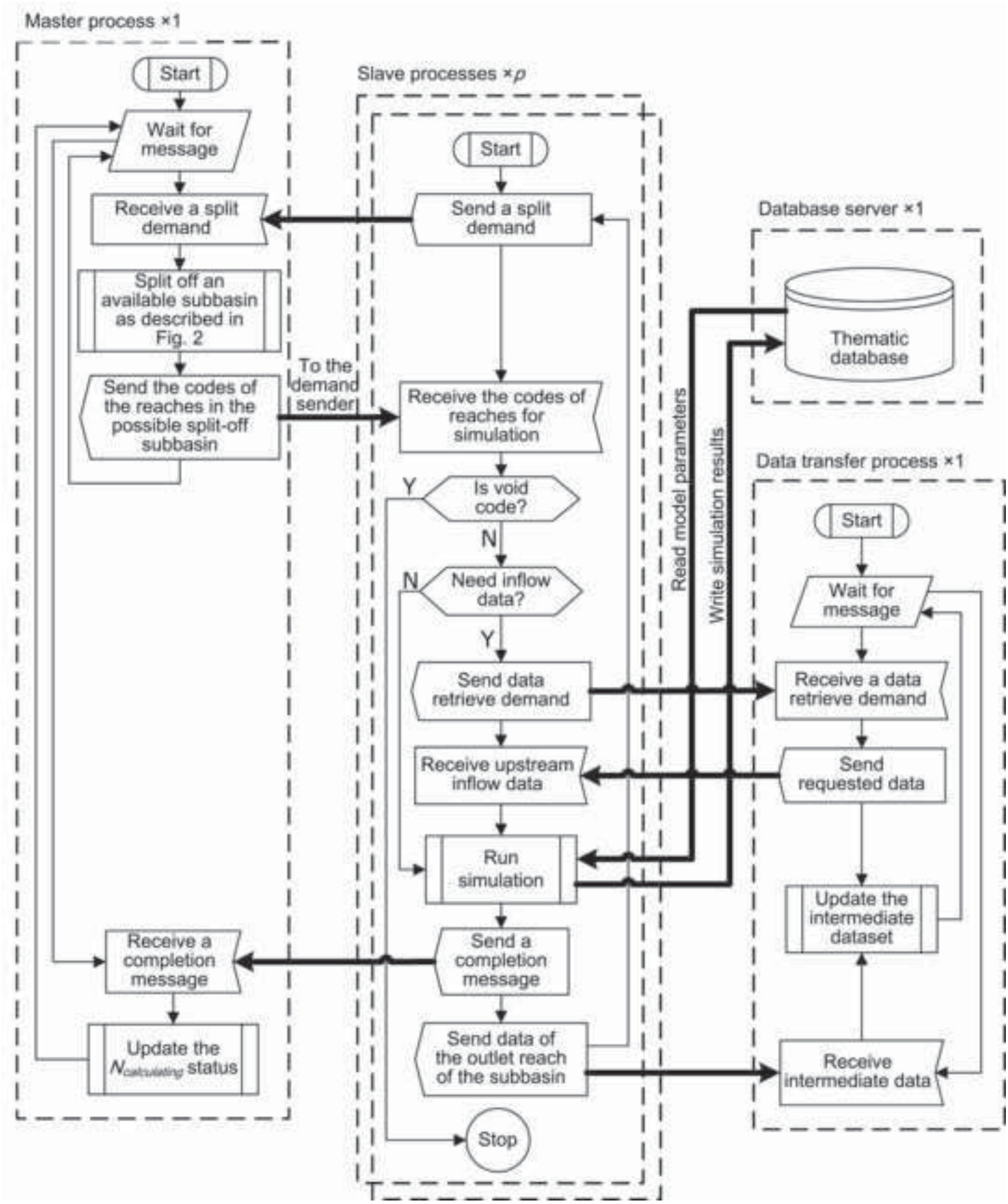

Figure4a
Click here to download high resolution image

Figure4b
Click here to download high resolution image

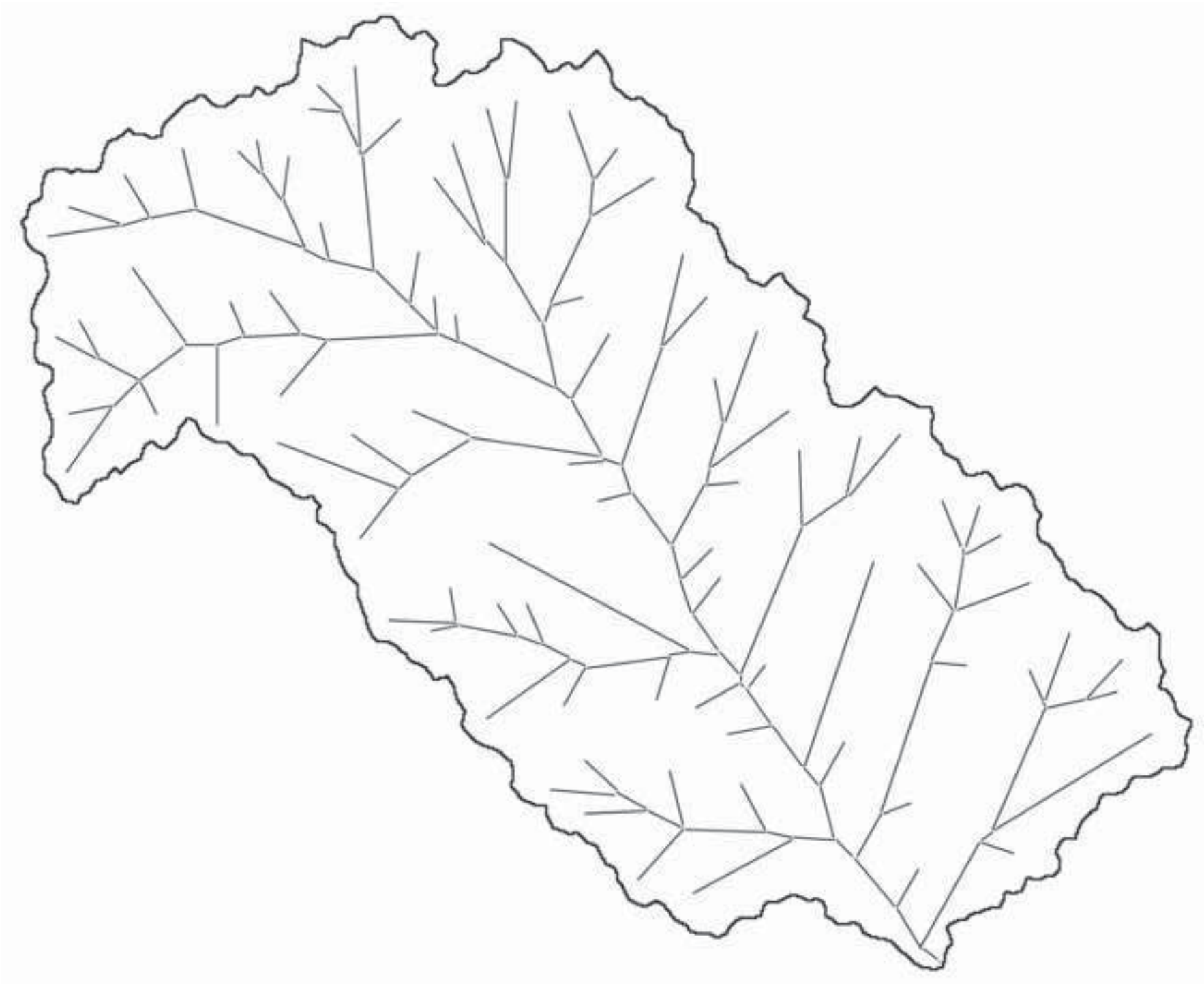

Figure5

Click here to download high resolution image

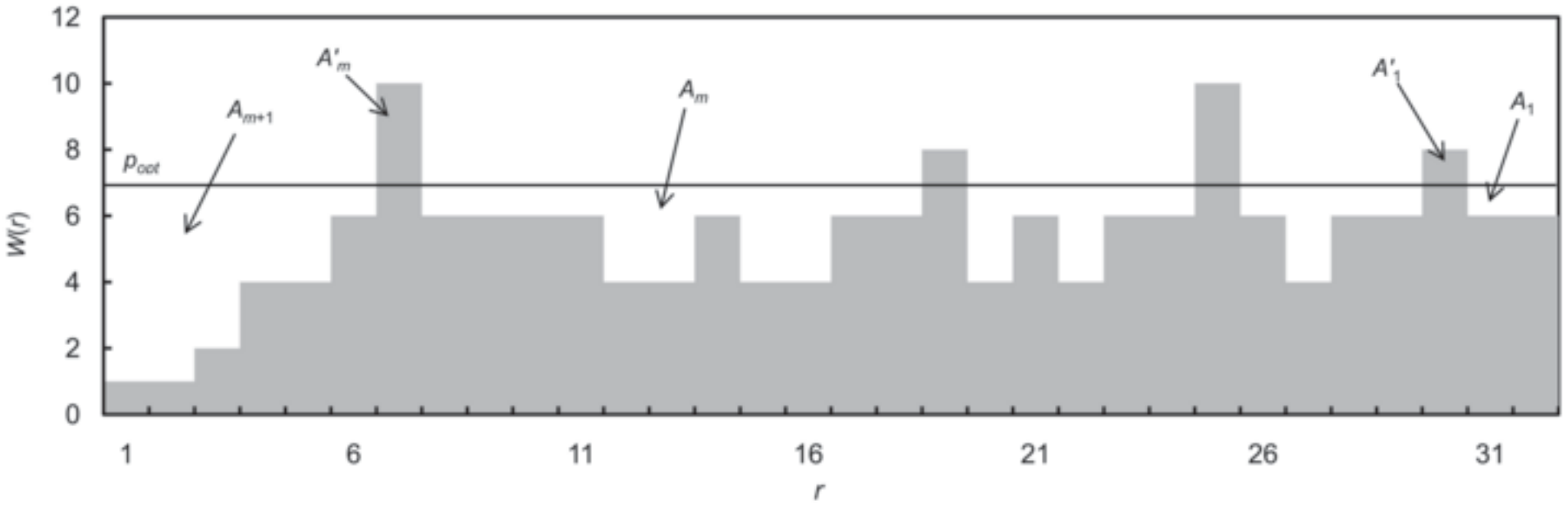

Figure6
Click here to download high resolution image

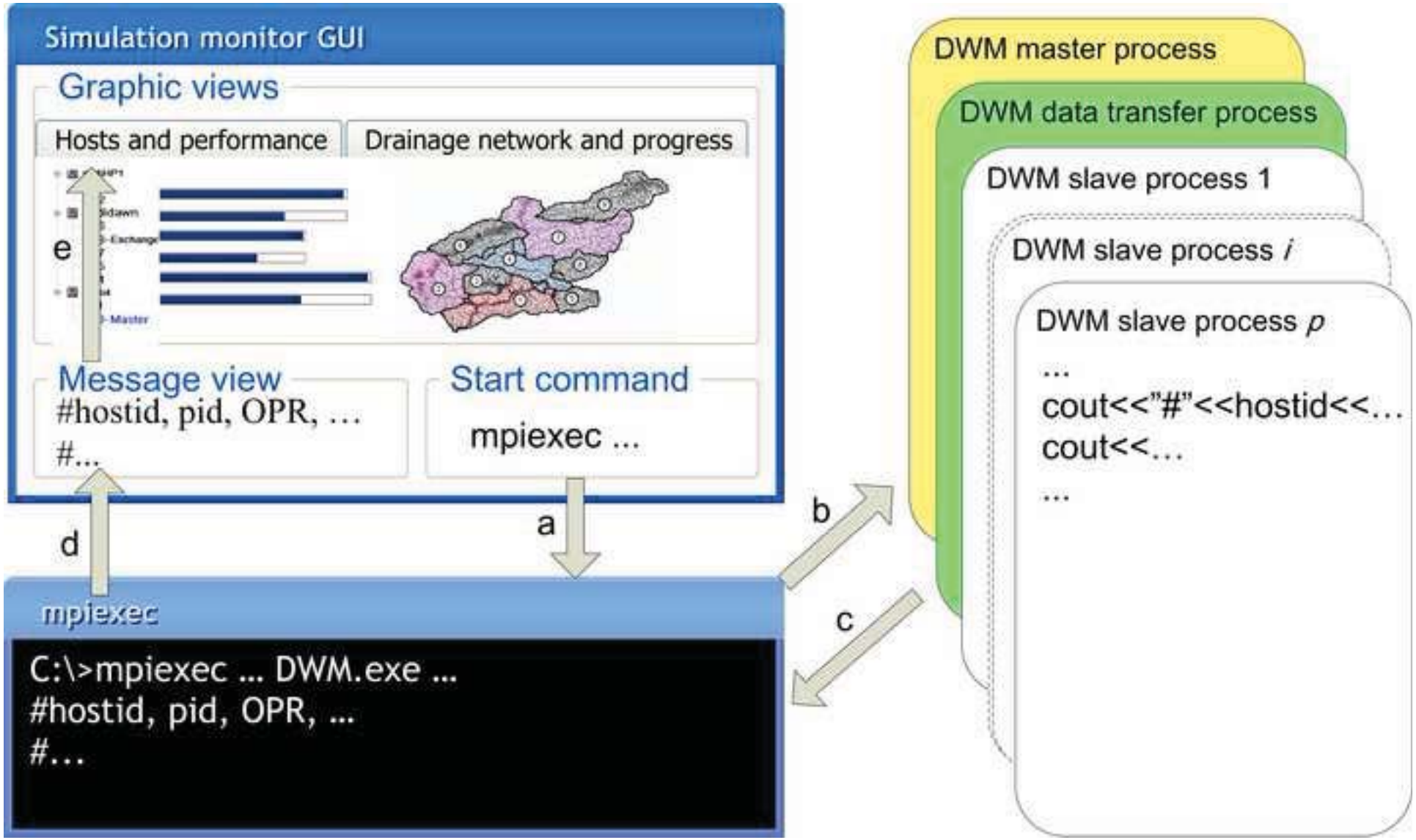

Figure7
Click here to download high resolution image

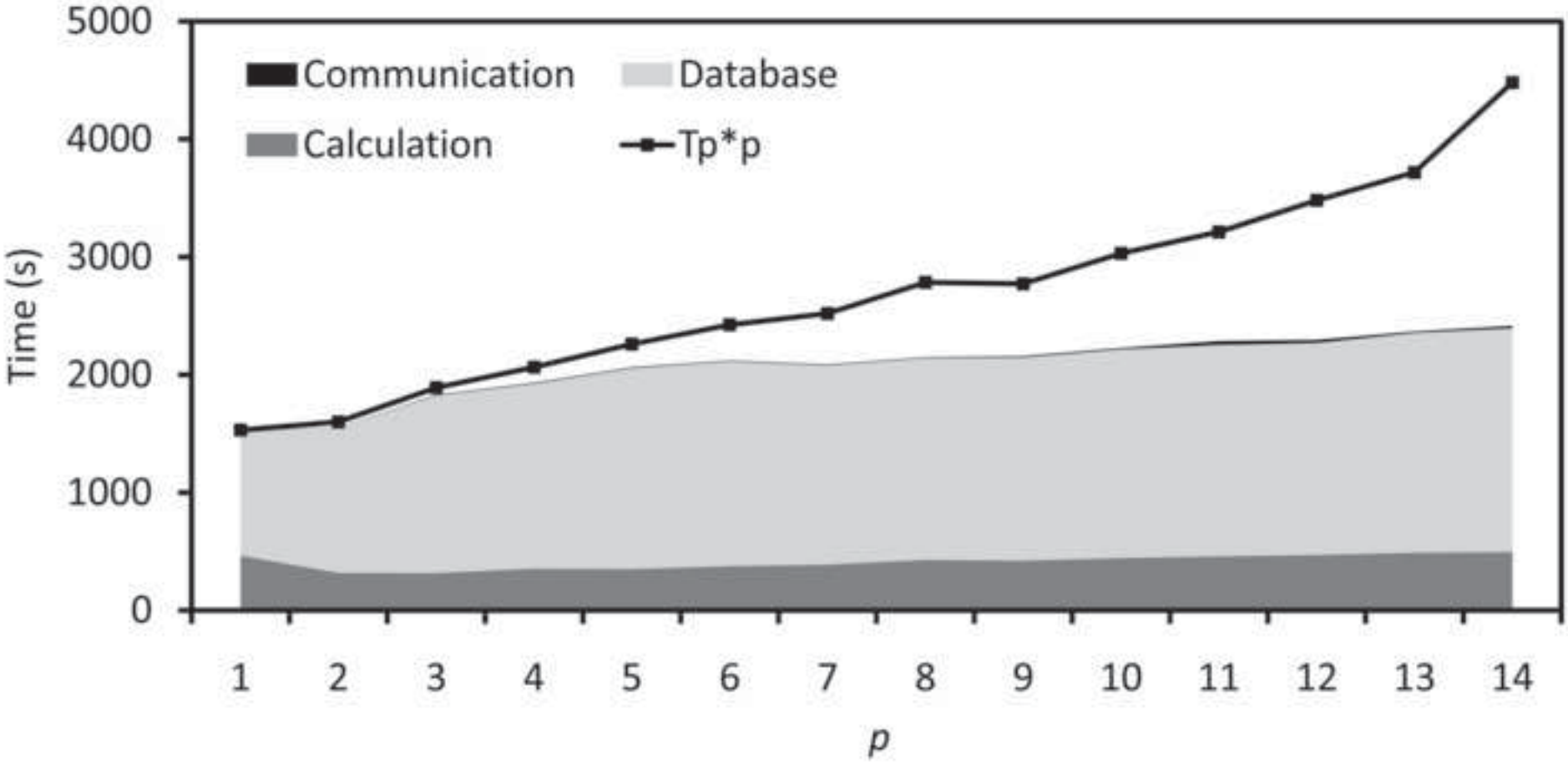

Figure8
Click here to download high resolution image