J Arid Land (2015) 7(1): 122–131
doi: 10.1007/s40333-014-0041-5
jal.xjegi.com; www.springer.com/40333

A new parallel framework of distributed SWAT calibration

Qiang LI[1,2,3*], Xi CHEN[2], Yi LUO[2], ZhongHua LU[1], YanGang WANG[1]

[1] Supercomputing Center, Chinese Academy of Sciences, Beijing 100190, China;
[2] Xinjiang Institute of Ecology and Geography, Chinese Academy of Sciences, Urumqi 830011, China;
[3] University of Chinese Academy of Sciences, Beijing 100039, China

Abstract: With the development of large-scale hydrologic modeling, computational efficiency is becoming more and more important. Rapid modeling and analysis are needed to deal with emergency environmental disasters. The Soil and Water Assessment Tool (SWAT) is a popular hydrologic model, which is less applied in large-scale watershed simulation because of its sequential characteristics. For improving the computational efficiency of the SWAT model, we present a new parallel processing solution for hydrologic cycle and calibration based on MPI (Message Passing Interface). We partitioned sub-basins during the processes based on a load balancing method. Then the calibration was parallelized using a master-slave scheme, in which different input parameters were allocated to different processes to run the hydrologic cycle and compute the function value. Because of the slow convergence and local optimization of the SCE-UA (Shuffled Complex Evolution-developed by University of Arizona) algorithm in SWAT calibration, a genetic algorithm (GA) is developed to optimize the calibration step. Then by dividing the default communicator into several sub-communicators, all the hydrologic cycles were parallelized in their own sub-communicators to achieve further acceleration. In this paper the results show speedups for the hydrologic cycle calculations, as well as in the optimized calibration step. In the case study, we tested the parallel hydrologic cycle by four processes, and got a speedup of 3.06. In the calibration section, after applying the GA optimization, with 10 cores, we got a speed increase of 8.0 in our GA parallel framework compared with the GA sequential calibration, which is much better than the original SWAT calibration. After the sub-communicators added, this process was speeded up even further. The study demonstrated that the GA parallel framework with multi-sub-communicators is an effective and efficient solution for the hydrologists in large scale hydrology simulations.

Keywords: parallel computing; hydrologic model; SWAT; parameter optimization; calibration

Citation: Qiang LI, Xi CHEN, Yi LUO, ZhongHua LU, YanGang WANG. 2015. A new parallel framework of distributed SWAT calibration. Journal of Arid Land, 7(1): 122–131. doi: 10.1007/s40333-014-0041-5

The Soil and Water Assessment Tool (Arnold et al., 1998) is a public domain model jointly developed by USDA Agricultural Research Service (USDA-ARS) and Texas A&M AgriLife Research. SWAT is widely used in assessing soil erosion prevention, non-point source pollution control and regional management in watersheds. Through a variety of physical and chemical models, SWAT can simulate runoff, evaporation, groundwater, precipitation and other hydrological processes (Neitsch et al., 2005).

The hydrologic cycle, as simulated by SWAT, is calculated by the following water balance formula:

$$SW_t = SW_0 + \sum_{i=1}^{t}(R_{\text{day}} - Q_{\text{surf}} - E_a - w_{\text{seep}} - Q_{gw}) .$$

Where SW_t is the final soil water amount (mm), SW_0 is the initial soil water amount on day i (mm), t is the time (days), R_{day} is the amount of precipitation on day i (mm), Q_{surf} is the amount of surface runoff on day i (mm), E_a is the amount of evapotranspiration on day i

*Corresponding author: Qiang LI (E-mail: liqiang@sccas.cn)
Received 2013-10-25; revised 2014-08-15; accepted 2014-09-22

(mm), wseep is the amount of water entering the vadose zone from the soil profile on day i (mm) and Qgw is the amount of return flow on day *i* (mm).

The computational costs increased for large-scale modeling with the increase in data sets. The sequential characteristic of the SWAT code excludes the use of multi-core and other parallel systems. The inadequate computational capability is a current central issue in the field of hydrology.

With the development of computer technology, parallel computing has become more and more popular. Instead of traditional sequential computing, parallel computing can utilize multi-core resources to accelerate the execution of a program. On a super computer, parallel programs can utilize thousands of CPU cores through parallel program routines, like MPI (Message Passing Interface, 1994), OpenMP or other parallel interfaces, resulting in better performance than programs running sequentially. So parallel computing is an important method to complex computational problems. In these parallel models, MPI uses multi-processes. The programmers can manage the variables and arrays among different processes. OpenMP (2008) applied multi-threads and shared memory, which can achieve excellent parallel performance in some computation-intensive loops. So MPI can be used to parallelize most programs according to the framework of the model, while OpenMP requires one or more hotspots in the code.

Distributed hydrologic modeling started in the 1990s, and many hydrologists devoted themselves to this field. Duan et al. (1992) developed an efficient global optimization method for conceptual rainfall-runoff models. And he developed a shuffled complex evolution approach for global minimization (Duan et al., 1993). The SCE-UA global optimization method for calibrating watershed models was applied by Duan et al. (1994). This method was selected as the calibration algorithem in SWAT2005. George (1997) demonstrated the SCE-UA modeling was robust and efficient in calibrating rainfall-runoff models. Savic et al. (1999) revised Duan's method and proposed a genetic programming approach to rainfall-runoff modeling and accelerated the optimization.

Liong et al. (2002) proposed a new paradigm in rainfall-runoff modeling, which is the first parallel hydrologic model. Vimal et al. (2006) established a parallel calibration model to reduce calibration time. In this model, a cluster of computers was used to run the calibration in parallel, greatly reducing the computation time. Yalew et al. (2010) first developed a parallel model for SWAT in 2010. This model splits SWAT into several modules running separately based on grid computing. However, decomposing SWAT in this manner is not advisable because of its integrated architecture, and this way is also not appropriate for large scale computation because of the limitations of the platform. Li et al. (2011) built a dynamic parallel model for hydrological simulation. Wang et al. (2011) set up a common parallel computing framework for hydrological modeling. Bacu et al. (2011) presented a method for calibrating the SWAT model on the grid infrastructure using the gSWAT application. In these models, modern parallel techniques were used to accelerate the simulation process on multi-core computers or cluster of computers. Rouholahnejad et al. (2011) presented a methodology for parallelizing the calibration of SWAT hydrological models on the Windows platforms using a Parallel Processing Scheme (PPS). Results indicated that the PPS performed much faster than the nonparallelised version. But the performance is poor, with the efficiency of PPS being no better than MPI. Denisa et al. (2012) compared the parallel execution of the SWAT hydrological model on multi-core and grid architectures and described the importance and possibility of parallel SWAT. Ilya et al. (2012) simulated soil and water conditions by 160,000 grids in an arid field of 16 km^2, but he didn't establish a parallel model, which cost him months of time. Wu et al. (2013) parallelized the SWAT hydrologic cycle and calibration, but the efficiency was not very high and the optimization algorithm was not ideal.

We can conclude that after more than 20 years of development, distributed hydrologic models, like SWAT have attracted more and more attention. Parallel computing has been applied to the study of hydrology, but the experimental results are not ideal because of the relatively dispersed structure of SWAT. When SWAT was developed in the early 1990s, the whole structure was established according to the hydrologic cycle, wthout considering parallel computing.

In this study, a synchronous genetic algorithm is proposed for the master-slave framework to accelerate optimization and convergence. Then the communicator is split into several sub-communicators, and in each sub-communicator the hydrologic cycle is parallelized by two or four processes to compute the function value. In this way, the SWAT calibration is parallelized and more CPU cores can be utilized to reduce computation time.

1 Method description

1.1 Parallelization of hydrologic cycle

The hydrologic cycle is the main process in SWAT. The hydrologic data obtained from the observing station are imported into the SWAT model. By simulating the physical and chemical processes, the hydrologic input data will be processed just like natural circulation. The SWAT model can simulate rainfall, evaporation, permeation, assimilation, underground water, and other processes using corresponding physical and chemical models. The parallelization of the hydrologic cycle can accelerate the calculation. However, the selection of the parallel strategy is important.

In parallel computing, the most popular interfaces are OpenMP (Open Multi-Processing) and Message Passing Interface (MPI). OpenMP applies multi-threading to shared memory, which can then be implemented by the corresponding directives to parallelize sections of the code. This method can be implemented simply and briefly, but besides parallelizing loops, there is not much control over the program. From Fig. 1 we can see the main execution time of the hydrologic cycle is about 2.11s (sub-basin module), which is composed of sumv function (0.33 s), surface function (0.264 s), crpmd function (0.213 s) and other modules. There are some assignment statements and read and write statements. We can see the hotspot of SWAT is not a loop by VTune, but some repeated statements which are dispersed in the sub-basin. OpenMP is not a good selection for good performance in SWAT.

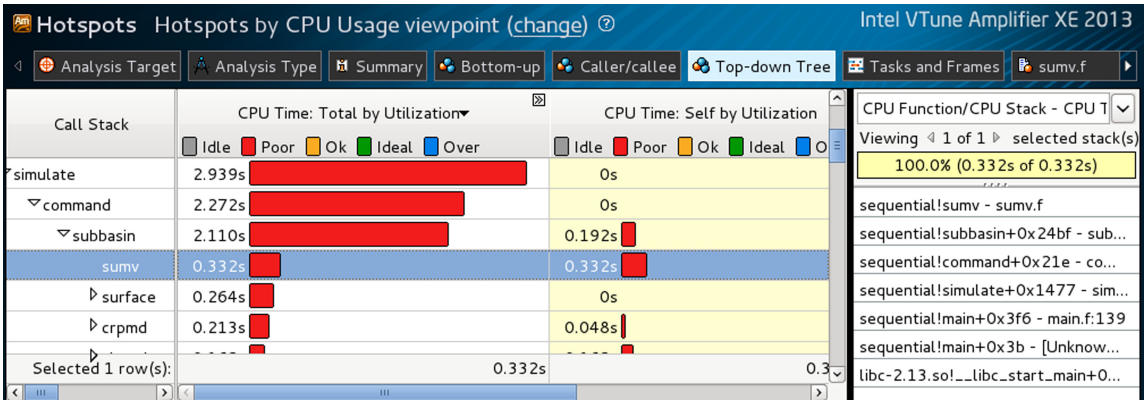

Fig. 1 The run time of each subroutine of SWAT by Vtune

MPI uses a distributed-memory model, then, different jobs can be assigned to different processes, whether it is a loop or not. We can also control process communication by grouping them into MPI communicators. MPI can solve the logical parallel relationship in the program, although it is more complicated than OpenMP. OpenMP can be applied to the parallelism on the code level. In the SWAT program, MPI is more appropriate because of its logical loop in the sub-basin module.

In SWAT, the model of the hydrologic cycle is divided into sub-basin and HRU (hydrologic response units). Each sub-basin is composed of several HRUs, and each HRU is an independent unit, consisting of some homogeneous land-use management and soil characteristics.

The parallelization of the hydrologic cycle can be implemented by parallelizing the sub-basins or HRUs. If we focus on HRU, all the HRUs will be distributed amongst the processes, and in this way, we could get high efficiency. The program must operate at a large scale to implement parallelization because the whole

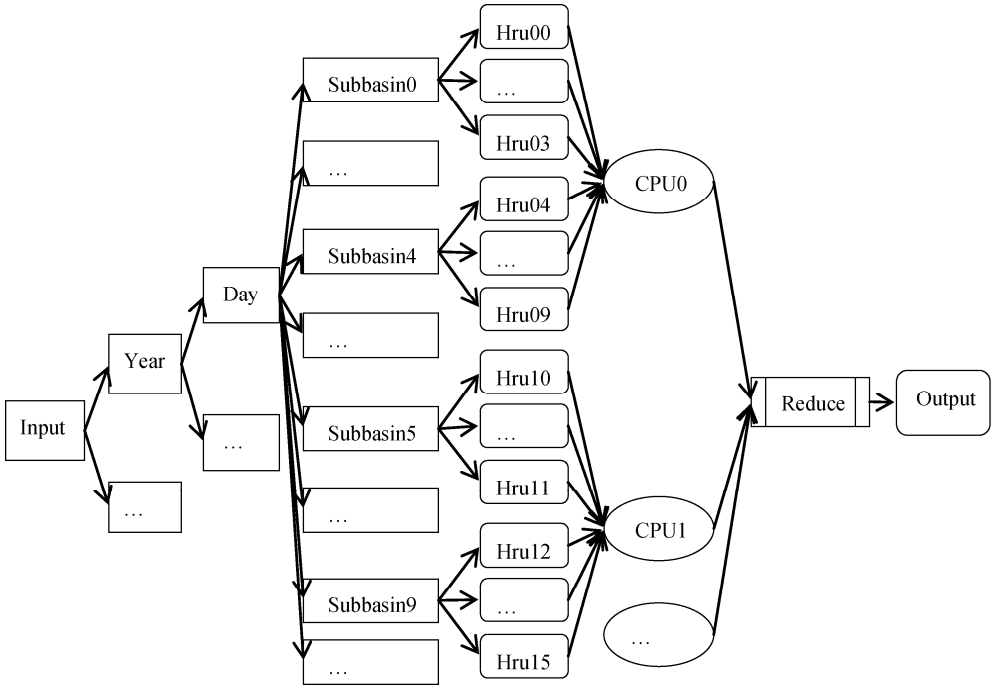

Fig. 2 The parallel strategy of SWAT hydrologic cycle

program was not developed for parallelism. We can try to parallelize the sub-basins, and allocate the sub-basins to different processes. All the HRUs in the sub-basin can be processed sequentially.

1.2 Calibration of SWAT

The objective of calibration is to find out the best input parameter set. After the SWAT model is established for some locations, the appropriate input parameter set is necessary to make simulations or forecasts. Traditionally we search for the optimum point in the feasible space, which is composed of all feasible points. We use some optimization method, like SCE-UA by Duan et al. (1994).

In SCE-UA, the structure is as follows: first the program confirms the number of complexes (*ngs*) and the number of points (*npg*) in each complex and randomly generates the points P_i, $i=1, ..., npt$ ($npt = ngs \times npg$). The components of each point are the *nopt* input parameters x_j, $j=1, ..., nopt$. The objective function can be denoted by a residual value:

$$F(P_i) = \left(swat\left(x_1, x_2, ..., x_{nopt}\right) - observe\left(x_1, x_2, ..., x_{nopt}\right)\right)^2. \quad (1)$$

Where $swat(x_1, x_2, ..., x_{nopt})$ is the SWAT simulation value by the input parameter $x_1, x_2, ..., x_{nopt}$, and $observe$ $(x_1, x_2, ..., x_{nopt})$ is the observed value by the same parameter $x_1, x_2, ..., x_{nopt}$. Beginning with the first iteration, the *npt* points will be partitioned into *ngs* complexes: $C_1, C_2, ..., C_{ngs}$ with the points $O_1, O_2, ..., O_{npt/ngs}$ in each complex. Beginning with the first complex C_1, some points Q_k, $k=1, ..., nps$ ($nps<npt/ngs$) will be selected randomly to comprise a sub-complex. In this complex, the points will be sorted in an order of increasing function values ($F(O'_1) \le F(O'_2) \le ... \le F(O'_{npt/nps})$) by a sort function. The points in the sub-complex are sorted at the same time ($F(Q'_1) \le F(Q'_2) \le ... \le F(Q'_{nps})$). A new point will be generated by the rule:

$$O_{new} = \frac{Q'_1 + Q'_2 + ... + Q'_{nps-1}}{nps} + \alpha \times \left(\frac{Q'_1 + Q'_2 + ... + Q'_{nps-1}}{nps} - Q'_{nps} \right), \quad (2)$$

where α is a constant which may be contracted when needed according to different models. The smaller the function value is, the better the input parameters are, because the function value is computed by the residual of the function value and the observation value. Then if the new point is better than the worst one, it

will be substituted into the sub-complex and complex, (Q_1, Q_2, Q_{nps-1}, O_{new}) and (Q_1, Q_2, ..., Q_{nps-1}, O_{new}), and then sorted. The program will then enter the next complex C_2. If the new point is not better than the worst one, the parameter α will be contracted several times. After that if the new point is still unacceptable, this point will be re-generated randomly until it is acceptable, and then the program will enter the next complex C_2. After all the complexes are finished, the program will enter the next iteration. All the points are sorted in an order of increasing function value, then some of the worst points will be dropped and new points will be generated randomly. The points will be allocated to *ngs* complexes and the optimization in complexes will begin for the second time. Once the normalized standard deviation of the parameter is less than 10^{-3} the iteration will terminated.

The objective of calibration is to narrow the difference between the experimental value and the observed value. The convergence of SCE-UA is not ideal, so we propose a new method to reduce the calibration time.

1.3 A parallel synchronous genetic algorithm for SWAT

The main problem of SWAT calibration is the slow convergence. The reason is that in every iteration, the new point is always the geometric mean point of the current sub-complex. In this way, some good components of the point in the sub-complex can't be transferred to the new point, but some of the worse components may be generated. So the evolution speed is very slow, resulting in slow convergence. For accelerating this optimization, we try to keep the good parameters in every iteration, and ensure that they are inherited by the next generation.

The genetic algorithm was considered. In this algorithm, by the crossover operation, the good genes are shared within complexes and the good parameters can be kept and inherited. Because of the mutation operation, in case a local optimum appears, the points are able to jump out of this field to search for better points. Through the genetic algorithm, the weakness of SWAT calibration can be overcome.

There is good parallelism in the genetic algorithm itself. In each generation, the operations of crossover and mutation only take place on the current complex. There is no relationship among complexes in the evolution process. For avoiding the local optimum, the information among complexes should be shared, and some synchronization is required. On the whole, there is still good parallelism in this algorithm.

In a master-slave mode, when the new points generated, the optimizing algorithm can be stated as follows:

In the master process, the current points are $x_i, i = 1, ..., npt$, which denote the chromosomes in the GA and the componets of the points stand by the genes in the chromosomes. The function values are xf_i, $i = 1, ..., npt$ ($xf_1 \le xf_2 \le ... \le xf_{npt}$), which denote the fitness value in GA. Good chromosomes mean high fitness value. There are *nprocs* slave processes, and the algorithm is as follows:

In master process:

Step 1. Generate points x_i, $i = 1, ..., npt$;

Step 2. Do i=1, *npt*+*nprocs*–1,
receive function value xf_n from process n;
send x_i to one slave process;
end do.

Step 3. Sort the current *npt* points, and the goto step 1.

In slave processes:

Step 1. Send function value to master process;

Step 2. Receive new x_i, $i = 1, ..., npt$ from master process;

Step 3. Check convergence;

Step 4. Compute the function value x_i, $i = 1, ..., npt$ from master process;

Master process in Step 1, the new points generation can be stated as follows:

Step 1. Generate a number R between 0 and 1 randomly;

Step 2. If R>0.5, then corossover. Select the j^{th} component s_j randomly from a random point x_k, then copy s_j of x_k to a temporary varible α. Then, copy s_j of x_{k-1} to x_k, copy s_j of x_{k-2} to x_{k-1}, ..., copy s_j of x_{npt1} to x_1, ..., copy s_j of x_{k+1} to x_{k+2}, copy α to s_j of x_{k+1}.

Step 3. If R<5, then mutation. Select the j^{th} component s_j randomly from the 1st point. If the upper and lower bound of s_j is *bu* and *bl*, and if

$s_j > \frac{bu-bl}{2}$, then $s_j = \frac{bu-bl}{2} - (s_j - \frac{bu-bl}{2})$, if $s_j < \frac{bu-bl}{2}$, then $s_j = \frac{bu-bl}{2} + (\frac{bu-bl}{2} - s_j)$, if $s_j = \frac{bu-bl}{2}$, then reselect s_j untill $s_j \neq \frac{bu-bl}{2}$.

Step 4. Generate new points to replace the last several points randomly.

1.4 Group and communicator management

In a synchronous GA framework, the master process will generate new points by GA, send the points to slave processes and receive the function values. Then it will sort these points according to the function values and return to generate points for the next time. The slave processes will receive the new points, calculate the function value and send them back to master process until the convergence is met. So in this framework, we can add and cut down the number of slave processes easily and the parallelism is acceptable.

The hotspot of the SWAT program is the calculation of the function value. In this framework, the program has been parallelized, and the number of slave processes can be added and cut down easily. But, the number of seeds in GA is a constant. If the number of slave processes is the same constant, the master process can send out all the points one time. So if we add the group and communicator management of MPI to SWAT, we can not only reduce the load of slave processes, but we can make use of more CPU cores.

In MPI, there is a default communicator named *mpi_comm_world*. In this communicator, the collective communication operations can be implemented among all processes. If we want some of the processes to do some collective communication, we must split the default communicator into sub-communicators. In each sub-communicator, we can make the current processes do some collective communication operations, while in other sub-communicators, we can assign other work.

In the synchronous GA framework, we can set all the master and slave processes in the default communicator *mpi_comm_world*. Then, we can split this communicator into some sub-communicators. All the slave processes are allocated evenly into these sub-communicators. There is only one process as the main slave process in each sub-communicator. After the master process sends the new points to the main slave processes, all the processes including the main slave process in each sub-communicator will run the hydrologic cycle in parallel. The framework of synchronous GA with group communication is shown in Fig. 3.

2 Results and discussion

Lu et al. (2012) established the runoff model in Ili River basin by the SWAT model and calibration algorithm, and got good results. The model used a small domain size because of the sequential characteristic of the SWAT model. Li et al. (2011) modeled the Chabagou basin using a parallel framework, and he got the highest speedup of 5.34 by 13 processes. Dai et al. (2013) calibrated the parameters of Manas River Basin using the algorithms by Li et al. (2002, 2003). But the study area is still not large. While in our case, to test the parallelism performance, we applied this parallel SWAT technique to the watershed of the Shule River in Gansu province, China, from Qilian Moutain to Tuolai South Mountain, which is 11, 050.589 km^2 in size, containing 246 HRU, for the year 1961 to 1970. The Shule River is the second largest river in Gansu province, and is a typical river in the arid lands of northwest China. We run this program on a micro super computer with 40 Xeon E5750 CPUs.

From Table 1, we can see that when we run the hydrologic cycle, we can make use of more processes than the original SWAT. If we just parallelize the sub-basin section, the total speedup is still not very high because there are a lot of read and write operations, which are related to the structure of SWAT itself. So if the watershed is not too large, the read and write operation may occupy a large portion of time. But we can expect better performance when the number of sub-basins is increased.

While in the study of Wu et al. (2013), a master-slave framework in parallelization of hydrologic cycle was developed, which was difficult to add to the calibration to get better performance. Even if this framework can be implemented, there will be a master node in each sub-communicator, which is wasteful. Also, the optimization method in calibration was not improved, and overall, the main focus of that

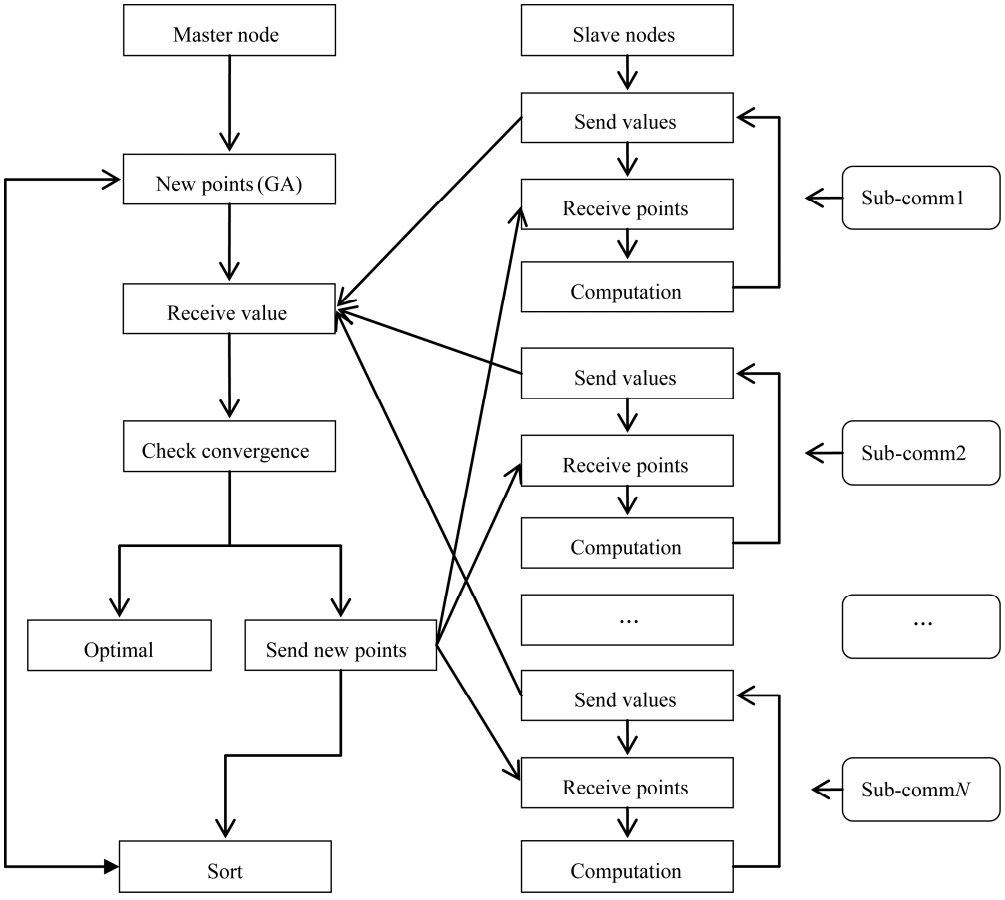

Fig. 3 Parallel SWAT2005 calibration structure based on multi-subcommunicators

Table 1 The execution time and speedup with different number of processes

Number of processes	1	2	4
Sub-basin function (s)	3.61	1.96	1.18
Speedup of sub-basin		1.84	3.06
Total time (s)	5.65	3.70	2.96
Total speedup		1.53	1.91

paper was the parallelization of the hydrologic cycle; the calibration was just mentioned.

Now we will look at the calibration performance in the synchronous genetic algorithm framework. The objective of calibration is to find out the best parameters of the current model by the comparison of the computation value and the observed value. The difference between these two values is the function value of the calibration. So a smaller function value means better parameters.

In this experiment we set 20 seeds and one complex in SCE-UA in sequential GA and parallel synchronous GA. There were total 20 points. We set 16 parameters in this model. The data are from Shule River. We used 10 CPU cores in the parallel version. In Fig. 4, we can see that the function value of SWAT calibration falls much too slowly. More than 10,000 seconds are needed to get optimal parameters. In Fig. 5, the execution time falls to about 1,200 s. While in our parallel synchronous GA framework in Fig. 6, with 10 cores, the function value falls steadily, and was able to find better optimal parameters in less than 150 s. The speedup of this parallel model is about 8.0 compared with the sequential version of GA, and the parallel efficiency is 88.9%.

If the group and communicator management are added to our framework, we can expect that more CPU cores can be used and better performance can be gotten. We still set 20 seeds, and each process of evolution will be achieved by 1 CPU core. We can only theoretically use 20 CPU cores before. But in our

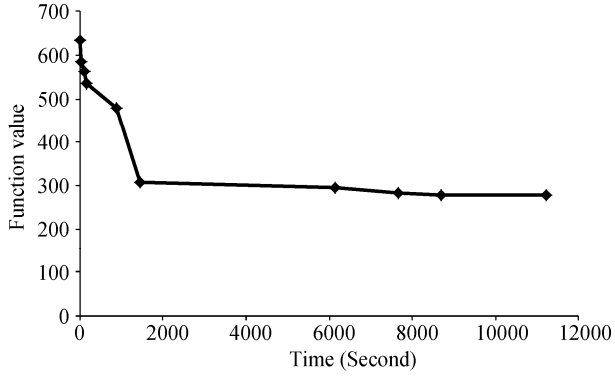

Fig. 4 Sequential SCE Objective Function Value

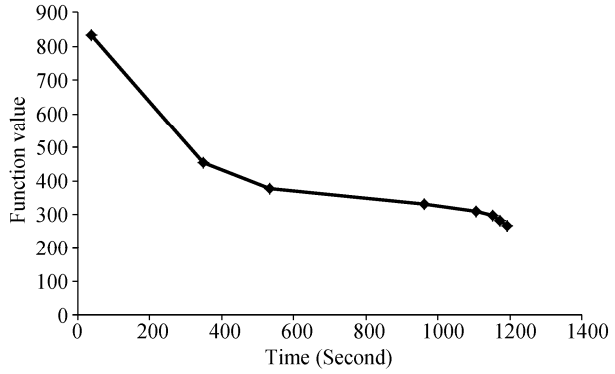

Fig. 5 Sequential GA objective function value

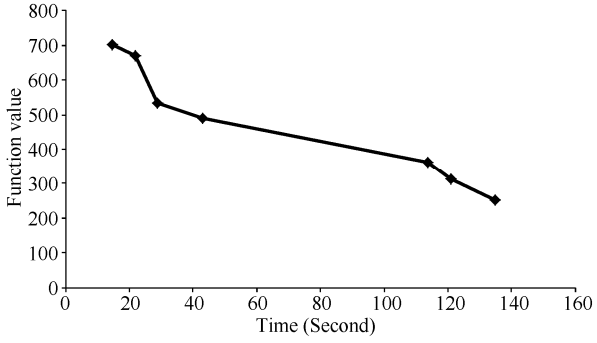

Fig. 6 Parallel synchronous GA function value in 10 cores

framework, 2 or 4 CPU cores will comprise a sub-communicator. In total, we can make use of 40 to 80 CPU cores becuase the process of evolution can be accelerated by 2 or 4 CPU cores according to Table 1.

From Fig. 7, we can see that after the the communicator management is added, we can use more CPU cores to accelerate the simulation. We can not only decrease the function value, but also speed up the program itself. The difference of the function values in these cases results from the stochastic characteristic of the GA. But the whole tendency is similiar, and the optimal parameters are almost the same. We can also see that the time consumed declined after more CPU cores were added, which is similiar to the results shown in Table 1. With 20 cores, the function value falls from 702 to 252 after about 140 s. While with 40 cores, the function value falls from 730 to 230 after 85 s, which is much better than 20 cores. And with 80 cores, the time falls to less than 70 s. According to the analysis above, if the watershed is large enough, we can expect better performance in this module.

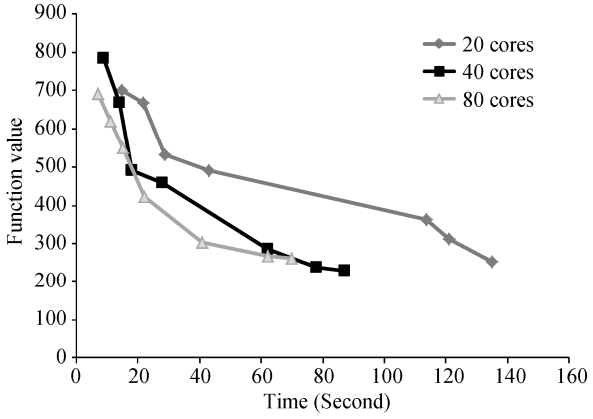

Fig. 7 Function value of different cores in different number of sub-communicators

After the cores are added to the sub-communicators, the framework becomes more complex. To test the stability of this framework in different generations of GA, the execution time of 1 core, 2 cores and 4 cores in each sub-communicator was tested. Figure 8 shows different number of processes in each sub-communicator results in the decrease of execution time with the generations of GA. Here we intercepted the first 20 generations of evolution. We can see although the number of processes in sub-communicator is different, the trendline is regular and steady, so the load balancing and scability of this framework is acceptable.

In this parallel framework, there's not only high computaion efficiency, but the results are also much better than the sequential SCE-UA algorithm. Because of the parallel optimizing stategy on multi-cores, good parameters can be found in acceptable time, instead of long term simulations for months in the past. In Fig. 9, the calibration shows good accuracy of the parameters from the output of the parallel framework.

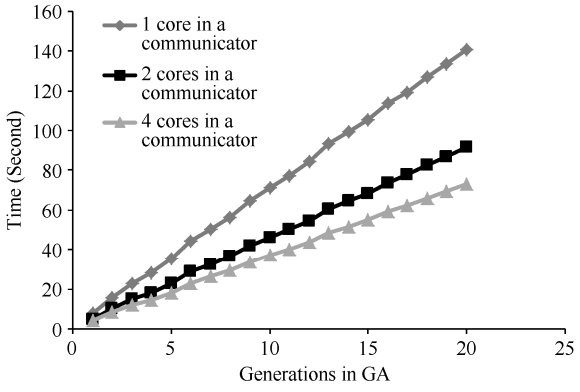

Fig. 8 Time consumed in different number of sub-communicators and in different generations of GA

3 Conclusions

In this paper, we developed a new parallel framewrok for SWAT calibration to enable large scale hydrologic modeling. We established the model based on a master-slave parallel framework. The master node receives the function value, generates new points by the operations of crossover and mutation and sends the new points to slave nodes. The slave nodes receive the new points, compute the function value and send them back to the master node. We used the optimization method based on a synchronous genetic algorithm. In the master node, when generating a new point, those good genes will be passed to the next generation by crossover operation, while the bad ones will be abandoned. For avoiding local optimum, the mutation operation is also added to jump out of the current complex. Then, to make use of more CPU cores, we use MPI communicators. Before the framework starts, we split the MPI default communicator into several sub-communicators, and two or four CPU cores were allocated to each sub-communicator. Each seed evolves in its own sub-communicator, and the cores in this sub-communicator will proceed with the hydrologic cycle in a parallel fashion. Then we can achieve the acceleration in each hydrologic cycle and more CPU cores can be used.

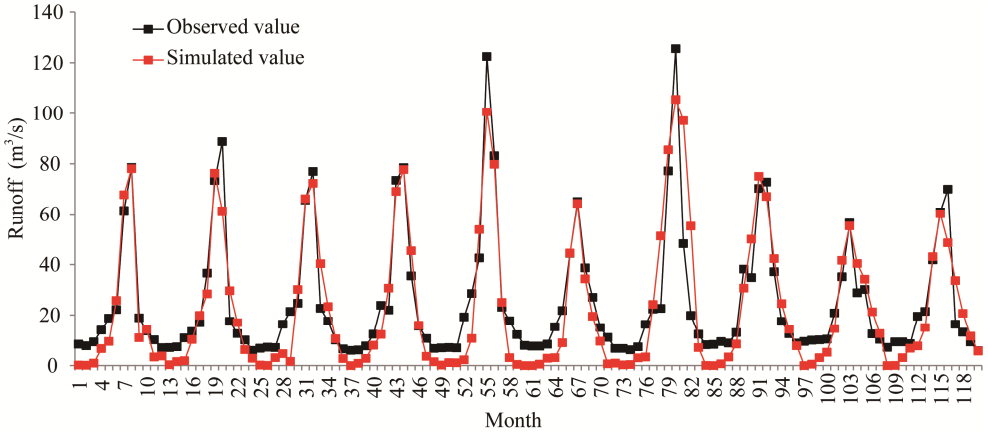

Fig. 9 Parameter calibration in Shule River from 1960 to 1969

The numerical experiments showed that four processes could be used to accelerate in each hydrologic cycle. The speedup is not high because of the numbers of read and write operation after sub-basin computation. We can expect higher speedup when the number of HRUs increases. When the synchronous GA is added to the master-slave parallel framework, the function value falls much more quickly than the SWAT calibration, which means better parameters can be found in less time than before. After the sub-communicator is added, we can use four processes in each sub-communicator. According to the speedup of the hydrologic cycle, we can also achieve similiar speedup in this framwork. Then we can make use of more CPU cores, and get the optimum parameter more quickly.

However, because of the structure of SWAT, the speedup of the hydrologic cycle cannot be very high and no more CPU cores can be used. If we want to get better performance, the whole structure of SWAT must be altered to adapt parallelization. This will be the future goal.

Acknowledgments

This work is supported by the National Basic Research Program of China (2010CB951002). We thank Brian SKJERVEN and Caroline QIAN from University of Minnesota, USA and YanFen YANG from Xinjiang Institute of Ecology and Geography, Chinese Academy of Sciences for their valuable comments.

References

Arnold J G, Srinivasan R, Muttiah R S, et al. 1998. Large area hydrologic modeling and assessment part 1: model development. Journal of the American Water Resources Association, 34(1): 73–89.

Barbara C, Gabriele J. 2008. Using OpenMP: Portable Shared Memory Parallel Programming. Cambridge: MIT Press, 79–152.

Bacu V, Mihon D, Rodila D, et al. 2011. Grid based architectural components for SWAT model calibration, In: 2011 International Conference on High Performance Computing and Simulation (HPCS), Istanbul, 193–199.

Dai S S, Li L H, Xu H G, et al. 2013. A system dynamics approach for water resources policy analysis in arid land: a model for Manas River Basin. Journal of Arid Land, 5(1): 118−131.

Denisa R, Victor B, Dorian G. 2012. Comparative parallel execution of SWAT hydrological model on multicore and grid architectures. International Journal of Web and Grid Services, 8(3): 304–320.

Duan Q Y, Sorooshian S, Gupta V. 1992. Effective and efficient global optimization for conceptual rainfall–runoff models. Water Resources Research, 28(4): 1015–1031.

Duan Q Y, Gupta V K, Sorooshian S. 1993. A shuffled complex evolution approach for effective and efficient global minimization. Journal of Optimization Theory and Applications, 76(3): 501–521.

Duan Q Y, Sorooshian S, Gupta V. 1994. Optimal use of the SCE-UA global optimization method for calibrating watershed models. Journal of Hydrology, 158(3): 265–284.

George K. 1997. Efficient subspace probabilistic parameter optimization for catchment models, Water Resources Research, 33(1): 177–185.

Gropp W, Lusk E, Skjellum A. 1994. Using MPI: Potable Parallel Programming with the Message-passing Interface. Cambridge: MIT Press, 51–108.

Ilya M D, Jonathan J B, Amanda J S. 2012. A high-resolution model of soil and surface water conditions. Ecological Modeling, 237: 109–119.

Li L, Simonovic S P. 2002. System dynamics model for predicting floods from snowmelt in North American prairie watersheds. Hydrological Processes. 16(13): 2645–2666.

Li L, Yakupitiyage A. 2003. A model for food nutrient dynamics of semi-intensive pond fish culture. Aquacultural Engineering, 27(1): 9–38.

Li T J, Wang G Q, Chen J, et al. 2011. Dynamic parallelization of hydrological model simulations. Environmental Modeling & Software, 26(12): 1736−1746.

Liong S Y, Gautam T R, Khu S T, et al. 2002. Genetic programming: a new paradigm in rainfall runoff modeling. Journal of American Water Resources Association, 38(3): 705–718.

Lu Z X, Cai X H, Zou S B, et al. 2012. Application of SWAT model in the upstream of Ili River Basin with Scarce Data. Arid Land Geography, 35(3): 399−407. (in Chinese)

Rouholahnejad E, Abbaspour K C, Vejdani M, et al. 2011. Parallelizing SWAT calibration in windows using SUFI2 program. Environmental Modelling and Software, 31: 28–36.

Savic D A, Walters G A, Davidson J M. 1999. A genetic programming approach to rainfall–runoff modeling. Water Resources Management, 13(3): 219–231.

Sharma V, Swayne D C, Lam W, et al. 2006. Auto-calibration of hydrological models using high performance computing. In: High-Performance Computing in an Advanced Collaborative Environment, 2006. HPCS 2006. 20th International Symposium on Source: IEEE Xplore.

Neitsch S L, Arnold J G, Kiniry J R, et al. 2005. Soil and Water Assessment Tool Theoretical Documentation Version 2005. Taxas: Texas Water Resources Institute.

Wang H, Fu X D, Wang G Q, et al. 2011. A common parallel computing framework for modeling hydrological processes of river basins. Parallel Computing, 37(6–7): 302–315.

Wu Y P, Li T J, Sun L Q, et al. 2013. Parallelization of a hydrological model using the message passing interface. Environmental Modelling & Software, 43: 124−132.

Yalew S G, Griensven A V, Kokoszkiewicz L. 2010. Parallel Computing of a Large Scale Spatially Distributed Model Using the Soil and Water Assessment Tool. Ottawa: School of Computer Science University of Guelph.