Parallelization in SCALE Continuous-Energy Resonance Module GEMINEWTRN and Transport Module NEWT

Zhaopeng Zhong* and Thomas J. Downar

* zzhong@ecn.purdue.edu

Purdue University
School of Nuclear Engineering
West Lafayette, Indiana 47907-1290

Mark D. DeHart, Mark L. Williams

Oak Ridge National Laboratory
Nuclear Science and Technology Division
Oak Ridge, Tennessee 37831-6370

Abstract – *A new resonance module, GEMINEWTRN, has been developed in SCALE, it can calculate the continuous-energy neutron flux within the whole two-dimensional geometry, providing us a rigorous soution. However, the new code needs tremendously amount of computation and memory for practical problem. To relieve the computational burden and memory requirement, parallelization has been implemented into GEMINEWTRN, both angular and spatial decomposition have been adopted so that both the computation and the memory requirement on each processor can be saved considerably, and this effort makes the new resonance method much feasible for practical use. Because the two-dimensional geometry capability and SN/ESC solver of GEMINEWTRN come from lattice physics code NEWT, the similar parallel technique has also been implemented into NEWT, which can also save the computation considerably.*

Introduction

A new code, GEMINEWTRN (Group and Energy Point-wise Methodology Implemented in NEWT for Resonance Neutronics) [1], has been preliminarily developed to improve the accuracy of the resonance self-shielding calculation by developing continuous energy multi-dimensional transport calculations for problem dependent self-shielding calculations. GEMINEWTRN combines the PW solution of CENTRM[2] with the power and flexibility of NEWT's[2] general-geometry 2D solver, it can calculate the continuous energy neutron spectra within the whole 2-D geometry domain, implicitly including the 2-D heterogeneous effect in the resonance self-shielding calculation, removing a lot of approximations introduced by the traditional resonance methods.

Two-dimensional whole assembly S_N calculation using continuous-energy library has been achievable with the progress of computer hardware and computational methods. However, even with these developments in computational environment, the two-dimensional whole assembly calculation S_N calculation using continuous-energy library still requires not only tremendous computation time but also huge memory, i.e., for a typical light water reactor assembly model, there exists more than 30,000 energy points in the resonance region , at each energy point a transport sweep need to be performed within the whole 2-D geometry domain (usually contains more than 10,000 spatial meshes.) Therefore, for these large problems, parallel computation is essential to obtain a solution by distributing both the computational burdens and memory requirements.

The parallelization of GEMINEWTRN code focused not only on parallel computation but also memory distribution. GEMINEWTRN spends most of the computation time on transport sweeps at each energy point; most of the memory is allocated to store the cumulative resonance integrals which are necessary for the point-wise (PW) scattering source calculation, where considerably large amount of computation is also spent. Therefore for transport sweep, GEMINEWTRN is parallelized in the angular domain using the well-known MPI message passing interface or OpenMP parallelization tool; for the scattering source calculation, GEMINEWTRN is parallelized in the spatial domain using MPI. The computational memory distribution is achieved by allocating the cumulative resonance integrals for the assigned spatial domain, because the cumulative resonance integrals use dominant amount of memory during the calculation, the memory burden on each processed can be reduced almost linearly to the number of processors. Because the 2D transport solver in GEMINEWTRN and NEWT are exactly the same, the similar parallel method for transport sweeps has also been implemented into lattice physics code NEWT, which is helpful to reduce the computational burden.

The detailed description for the parallelization of GEMINEWTRN code and the parallel computation flow will be shown below, and the parallel performance of GEMINEWTRN code is presented using an GE12 BWR full assembly model on ORNL CPILE Linux cluster.

Transport Sweeps in GEMINEWTRN and NEWT

The two-dimensional SN/ESC Transport solver as well as the complex geometry capability in GEMINEWTRN comes from lattice physics code NEWT.

The transport equation can be reduced to the form that is most often solved by standard multi-dimensional discrete ordinates codes:

$$\Omega\cdot\nabla\psi(r,u,\Omega)+\Sigma_t(r,u)\psi(r,u,\Omega)=Q(r,u,\Omega) \quad (1)$$

$$Q(r,u,\Omega)=\sum_{lk=0}^{LK}\frac{2l+1}{2}Y_{lk}(\Omega)S_{lk}(r,u)+Q_{ext}(r,u,\Omega) \quad (2)$$

In the two-dimensional discrete ordinates transport sweep, i.e., in Eq. (1), once the source term $Q(r,u,\Omega)$ is determined, the calculation along each direction Ω can be performed independently. In the multi-group transport sweep, as that in NEWT, inner iterations need to be performed to converge the within-group scattering source, so that it's required that the contributions from all the angles should be integrated to calculate the total scalar flux and flux moments after each iteration; however, in the point-wise range, there doesn't exist within-group scattering source which means that neutron can not lose energy without changing flying direction, this advantage can be taken in the parallel computation of GEMINEWTRN, which can improve the parallel performance greatly.

Point-Wise Scattering Source Calculation in GEMINEWTRN

The continuous (point-wise) energy capability of GEMINEWTRN comes from CENTRM, so that the point-wise scattering source calculation follows the same principle as that in

CENTRM based on **submoment expansion** method, and the elastic point-wise scattering source can be expressed as the following expression[6].

$$S_{El}(r,u_n)=\Sigma_{n\to n}\psi(r,u_n,\Omega)+\sum_{lk=1}^{LK}\sum_{j}\frac{2l+1}{2}Y_{lk}(\Omega)\sum_{K=-l}^{l}Z_{lK}^{(j)}h_K(E_n)$$
$$\times\left\{0.5u_{n-1}\bar{\Sigma}^{(j)}(u_{n-1})w_K(E_{n-1})\psi_{lk}(r,u_{n-1})+\left(\Box\left[F_{lk;K}^{(j)};u_{n-1}\right]-\Box\left[F_{lk;K}^{(j)};u_n-\varepsilon^{(j)}\right]\right)\right\} \quad (3)$$

in which:

$$\Box\left[F_{lk,K}^{(j)};u\right]=\int_{u_0}^{u}w_K(E')\Sigma^{(j)}(u')\psi_{lk}(r,u')du'$$
$$\Box\left[F_{lk,K}^{(j)};u_{n-1}\right]-\Box\left[F_{lk,K}^{(j)};u_n-\varepsilon^{(j)}\right]=\int_{u_n-\varepsilon^{(j)}}^{u_{n-1}}w_K(E')\Sigma^{(j)}(u')\psi_{lk}(r,u')du' \quad (4)$$

where $\varepsilon^{(j)}=\ln\left(\frac{1}{\alpha_j}\right)$, $\alpha_j=\left(\frac{A_j-1}{A_j+1}\right)^2$, A_j-atomatic mass of nuclide j

In Eq.(4), the integrand corresponds to the resonance integral associated with space *r*, moment *lk,* submoment *K* and isotope *j*, and $u_n-\varepsilon^{(j)}$ is the smallest lethargy from which neutron can be elastically scattered into current lethargy point u_n for isotope *j*.

The integrand $\Box\left[F_{lk,K}^{(j)};u\right]$ can be calculated cumulatively, as following:

$$\Box\left[F_{lk,K}^{(j)};u_n\right]=\Box\left[F_{lk,K}^{(j)};u_{n-1}\right]+\int_{u_{n-1}}^{u_n}w_K(E')\Sigma^{(j)}(u')\psi_{lk}(r,u')du' \quad (5)$$

And the Excess integral $\Box\left[F_{lk,K}^{(j)};u_n-\varepsilon^{(j)}\right]$ can be got by interpolation from the values that were computed with Eq.(5), however, to do this, the values of $\Box\left[F_{lk,K}^{(j)};u\right]$ must be stored for all the lethargy points in the interval $\left[u_{n-1}-\varepsilon^{(j)},u_{n-1}\right]$. Usually there exists hundreds or even thousands of lethargy points in this lethargy interval, especially for some light nuclide in the moderator, like heavy hydrogen-D, so that the memory requirement to store the cumulative resonance integral is very large.

It can be seen in Eq.(3) that in point-wise scattering source calculation, the contribution of each isotope much be calculated separately and then integrated, so that the computation spent on the point-wise scattering source calculation is also non-trivial, especially when there exist a lot of isotopes in one mixture. It's also obviously to be seen in Eq. (3) that at each spatial point *r*, the scattering source calculation can be performed independently with other spatial points, once we've already get the scalar flux and flux moments from the transport sweeps.

Multi-Group Scattering Source Calculation in NEWT

A two-level Coarse Mesh Finite Difference (CMFD) acceleration technique has been implemented into NEWT [3], which can speedup the convergence of outer iteration greatly;

however, due to the fact that CMFD needs to allocate memory for a set of cross sections in each coarse mesh (usually pin cells), it will also increase the memory requirement a lot, especially for the scattering matrix, i.e., for a 17 × 17 typical PWR assembly model using 238 group neutron library, the scattering CMFD matrix will take more than 130 Mbyte memory for double precision data, and it also add non-trivial computation.

Multi-group scattering source calculation in CMFD can be shown in Eq. (6)

$$\bar{S}_g(r) = \sum_{g'=1}^{G} \bar{\Sigma}_{g'\to g} \bar{\phi}_{g'}(r) \qquad (6)$$

where $\bar{S}_g(r)$ = coarse mesh scattering source

$\bar{\Sigma}_{g'\to g}$ = coarse mesh averaged scattering cross section

$\bar{\phi}_{g'}(r)$ = coarse mesh averaged scalar flux for group *g'*

Multi-group scattering source calculation is also necessary for the transport sweeps, as shown in Eq. (7)

$$S_g(r,\Omega_n) = \sum_{l=0}^{L} \sum_{k=0}^{l} \frac{2l+1}{2} Y_{lk}(\Omega_n) S_{lk,g}(r) \qquad (7)$$

and

$$S_{lk,g}(r) = \sum_{g'=1}^{G} \Sigma_{l,g'\to g} \psi_{lk,g'} = \sum_{g'=1}^{G} \Sigma_{l,g'\to g} \sum_{n=1}^{N} w_n Y_{lk}(\Omega_n) \psi_{n,g'}(r) \qquad (8)$$

where $S_g(r,\Omega_n)$ = scattering source for direction Ω_n

$S_{lk,g}(r)$ = *lk*'th scattering moments for group *g*

$\Sigma_{l,g'\to g}$ = P_l scattering cross section

$Y_{lk}(\Omega_n)$ = real part of the spherical harmonic function evaluated at direction Ω_n.

w_n = weight function of angle *n*.

$\psi_{n,g'}(r)$ = angular flux of angle *n* and group *g'*

. It's also obviously to be seen in Eq. (6) (7) and (8) that at each spatial point *r*, the multi-group scattering source calculation can be performed independently with other spatial points, once we've already get the coarse mesh averaged scalar flux $\bar{\phi}_{g'}(r)$ and angular flux $\psi_{n,g'}(r)$.

Parallelization

The main goals in parallel computation of GEMINEWTRN code are not only parallel computation, but also memory distribution. Table1 shows the required memory for the ACR fuel lattice model, which is shown in Fig. 1. The total required memories are about 2.0 G using ENDF5 library and 3.6 G using ENDF6 library, which means even single assembly-level model is not easy to solve on a 1Gbyte memory computer. Most of the required memory is to store the resonance integrals, to reduce the memory requirement, it is essential for GEMINEWTRN to distribute these variables to local processors. These variables are cell-wise cumulative resonance integrals for each scattering moment, submoment and isotope.

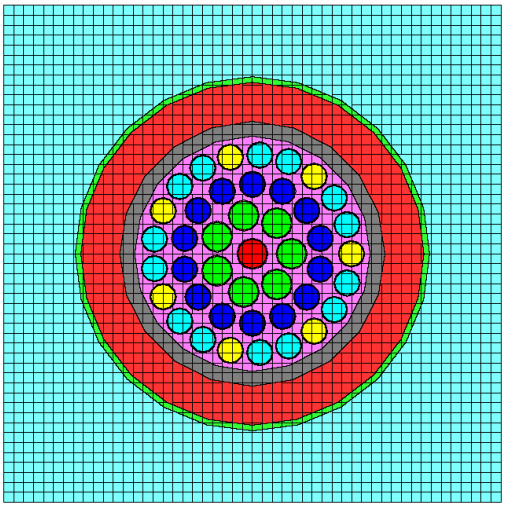

Fig.1 ACR fuel lattice model

Table 1 Memory requirement of GEMINEWTRN for ACR fuel lattice model

Modules	Memory Required (Gbytes) using ENDF5 library	Memory Required (Gbytes) using ENDF6 library
Resonance Integrals	1.711	3.236
Total	2.031	3.552

Table2 shows the computing time breakup of GEMINWTRN for the same ACR fuel lattice model, which is shown in Fig. 1. About 85% of the total computing time is spent on the transport sweep when ENDF5 library is used , the ratio is lower when ENDF6 library is used because in ENDF6 library there exists more isotopes for some mixtures, so that the fraction of computing time spent on PW scattering source calculation increase a lot, which also shows that the computation spent on the PW scattering source calculation is also considerably large . These results give the first priority in the parallel computation to the transport sweep and PW scattering source calculation parts. Therefore, the computational parallelization in the GEMINEWTRN code is realized in the transport sweep and point-wise scattering source calculation routines using the MPI message passing package.

Table 2 Computing time breakup of GEMINEWTRN for ACR fuel lattice model

Modules	Computing Time (min) using ENDF5 library / Fraction	Computing Time (min) using ENDF6 library / Fraction
PW Scattering source calculation	66.5 / 15%	149.8 / 26%
Transport Sweep	364.1 / 85%	421.2 / 74%
Total	430.6 / 100%	571.0 / 100%

Angular domain decomposition is introduced into GEMINEWTRN and NEWT for the transport sweeps, the total angular domain is divided into several angular sub-domain and each

will be assigned to one processor, each processor performs the transport sweeps within the assigned angular sub-domain for one or one group of angles, as shown in Fig.2 for some simple example, assuming the the total angular domain contains 8 angles. This angular domain decomposition is obtained using either the MPI or the OpenMP parallelization tool which forks multiple threads in the parallel section. MPI is for the distributed memory machine such as the LINUX cluster and has a disadvantage in the communication time taken to complete the scalar fluxes and flux moments. OpenMP is for the shared memory machines such as the multi-CPU workstation or supercomputer, but it's limited into one single machine and has the binding difficulties of threads to processors which is system dependent, sometimes OpenMP parallelization tool might be even slower than that of MPI, another disadvantage of OpenMP is that memory can not be saved, so in GEMINEWTRN MPI is used as the primary tools.

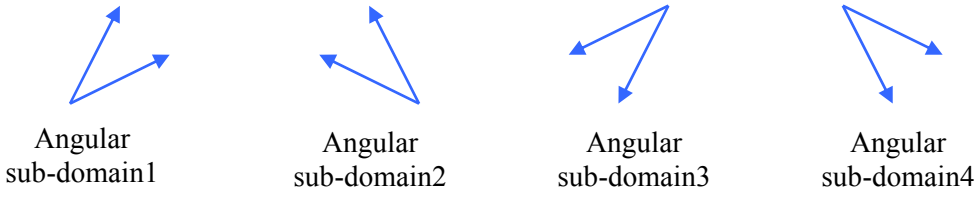

Fig.2 Angular Domain Decomposition

As explained previously, the transport sweep routines solves the angular flux along each direction independently once the source term is determined. In the multi-group range there exists within group scattering source, so that after each inner iteration all the angular fluxes must be integrated for scalar fluxes and flux moments to calculate the within group scattering source, which means that for distributed memory machines all the processors need to exchange data to integrate the total scalar fluxes and flux moments from all the angles after each inner iteration; in the PW range, there doesn't exist within group (energy point) scattering source, this gives us the advantage that the transport sweep along each direction can be performed even more independently, very little data exchange is needed during the iteration, all the processors only need to exchange the data after they get the converged angular flux for the current energy point, integrating the scalar fluxes and flux moments which will be necessary for the following PW scattering source calculation. After each iteration, all the processors need to exchange the out-coming angular fluxes, which are necessary to update the boundary incoming angular fluxes using boundary conditions, making preparation for the next iteration. Although the boundary out-coming angular fluxes need to be exchanged for each iteration, the communication time spent on this part is much smaller than that for the scalar fluxes and flux moments, because the data size to store the boundary out-coming angular fluxes is relatively very small in large model.

In GEMINEWTRN, spatial domain decomposition is also introduced for the point-wise scattering source calculation. Because the length of resonance integral is different for each isotope, memory requirements is also different. The spatial distribution of GEMINEWTRN is based on material, distributing the cells containing the same material onto different processors as evenly as possible, so that the resonance integrals associated with moment space, submoment space and each isotope belonging to this material would be also distributed evenly. A simple example is showing the spatial domain decomposition in Fig.3, the spatial domain is divided into several sub domains, and each will be assigned with one processor, all the variables related to the cumulative resonance integrals are defined only within each spatial sub domain, so that the memory allocated on each processor can be saved greatly.

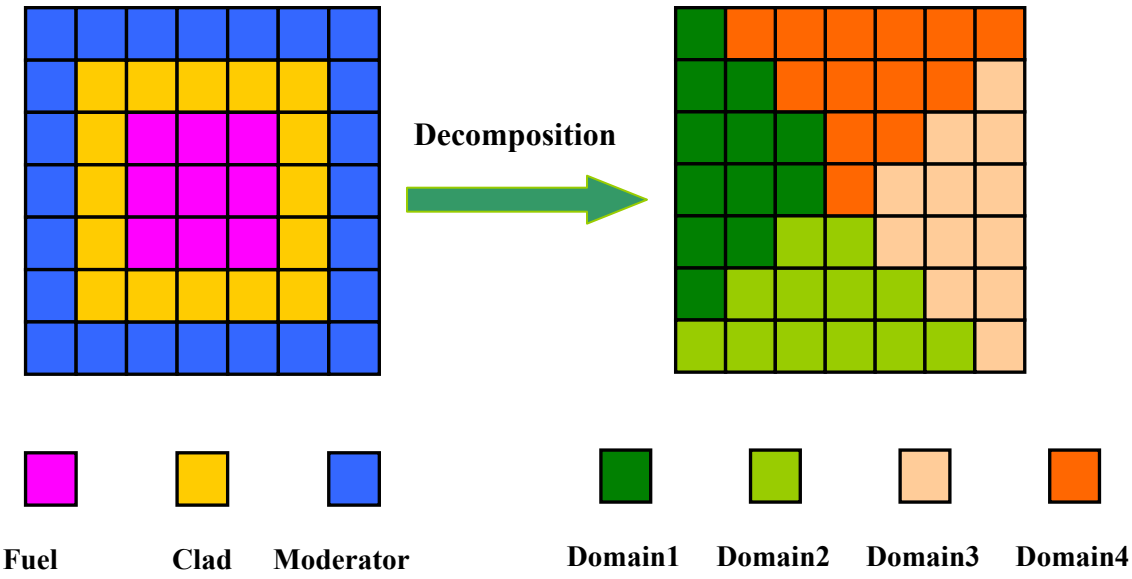

Fig.3 Spatial Domain Decomposition in GEMINEWTRN

Spatial domain decomposition is also implemented into NEWT for multi-group scattering source calculation, and here it's unnecessary to use the material-based distribution scheme as that in GEMINEWTRN, because only macro cross section is needed in multi-group calculation so that the memory size associated with each cell is the same, (we don't need to calculate the contribution of scattering source from each isotope), we just need to decompose the domain evenly.

The spatial domain decomposition for both GEMINEWTRN and NEWT is obtained using the MPI message passing interface. Within the spatial domain decomposition implemented, for GEMINEWTRN each processor performs the point-wise scattering source calculation only within the assigned spatial sub domain; and for NEWT, each processor performs the multi-group scattering source calculation for transport sweeps and CMFD calculation, therefore the computation of each processor is reduced greatly. After all the processors finish the calculation of point-wise or multi-group scattering source, they need to exchange data using MPI to combine them together.

Above all, using MPI for both the angular and spatial domain decomposition implemented, both the computation and the memory assigned on each processor can be reduced greatly.

Parallel Computing Flow

Single Program Multiple Data (SPMD) paradigm is the global parallel structure of GEMINEWTRN and NEWT. That is, we obtain the effect of different programs running on different processors by taking branches within a single program on the basis of process rank, the statements executed by different processors are different at some important part of the code, even though all processors are running the same program. For example, Fig. 4 shows the global parallel computing flow in the GEMINEWTRN code. At first, all the processors will do the initialization to setup the geometry and PW library; then during the memory allocation , some important array will be distributed onto different processors; for the parts that take most of the

computation, different computational branches will split and assigned onto each processor, based on spatial domain decomposition or angular domain decomposition.

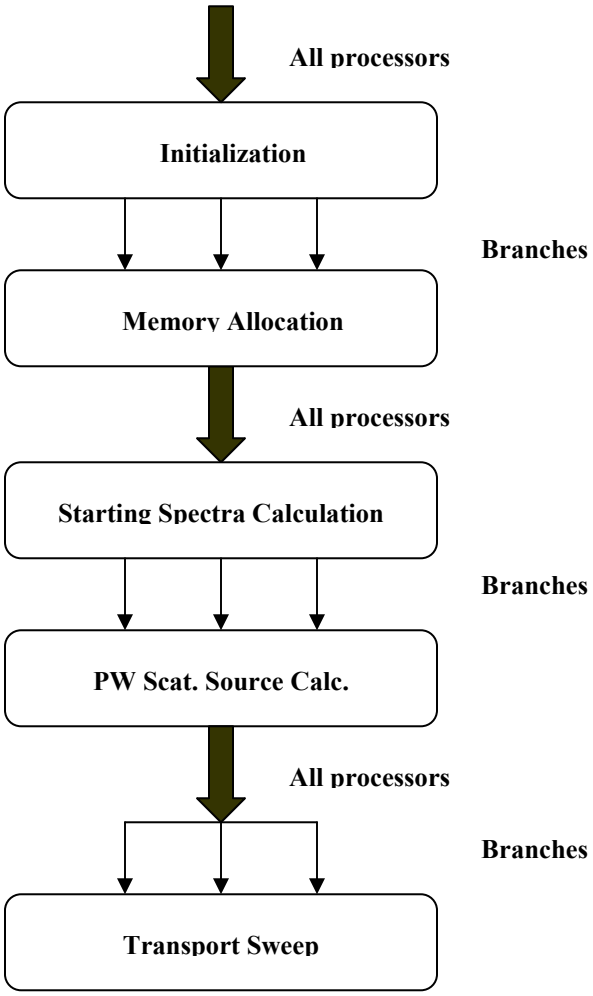

Fig.4 Global parallel computing flow of GEMINEWTRN

From Fig.4, it can be seen that at some important parts, i.e., the transport sweep and the PW scattering source calculation routines, the program is divided into different branches and the computation is performed in parallel. Fig.5 and Fig.8 show the parallel computing flow of transport sweep and PW scattering source calculation routines of GEMINEWTRN in detail separately. The global parallel computing scheme in NEWT follows the similar way. Fig.6 shows the parallel computing flow of transport sweep routine of NEWT.

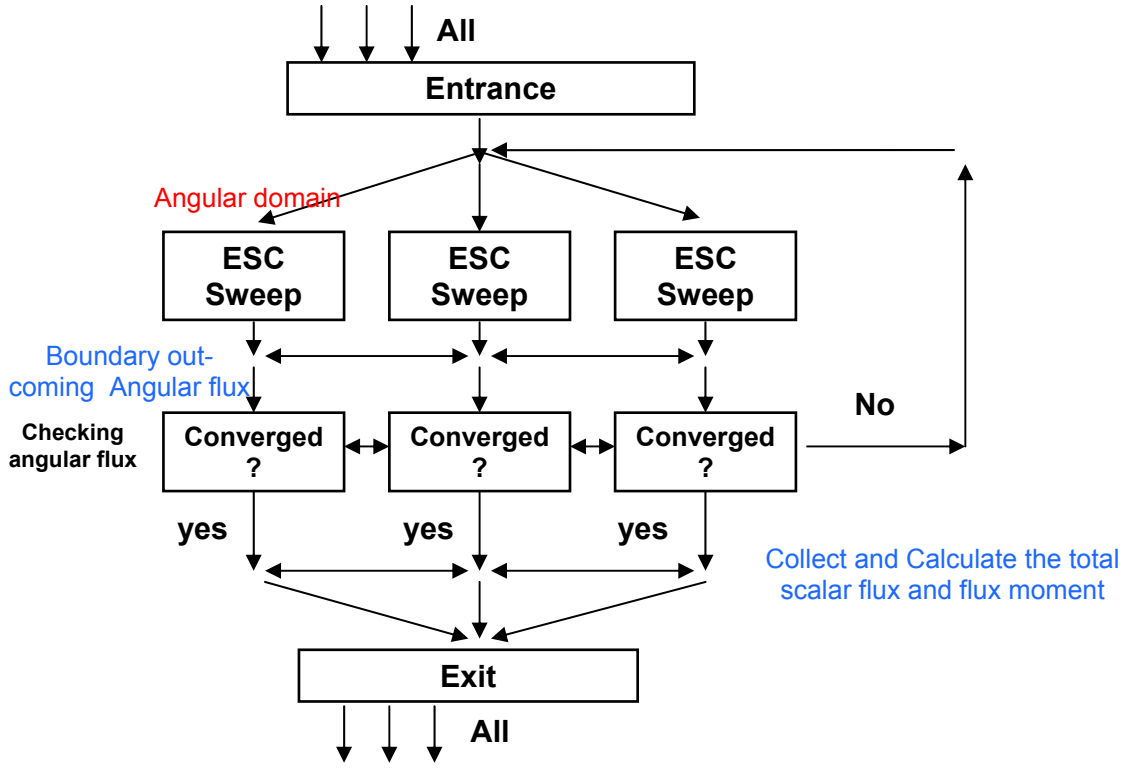

Fig. 5 Parallel computing flow for PW transport sweeps in GEMINEWTRN

In Fig.5, it can be shown that for transport sweeps in point-wise range, each processor is assigned with an angular sub-domain, taking a branch of calculation in parallel. After each iteration, all the processors need to exchange data to update the boundary incoming angular flux using boundary condition. After the angular flux is converged, all the processors need to exchange data to integrate the total scalar fluxes and flux moments, which are required for the following point-wise scattering source calculation. The basic communication algorithm is first, performing a all-to-one reduction, the master processor collects data from al the other processors, after processing the data, performing a one-to-all broadcast, the master processor distribute the processed data to all the other processors.

For the parallelization of transport sweeps in NEWT, as shown in Fig.6, it follows the similar principle as that in GEMINEWTRN, the most important difference is that all the processors need to exchange data to integrate the total scalar fluxes and flux moments to update the within group scattering source, so that it should need more data communication than that in NEWT.

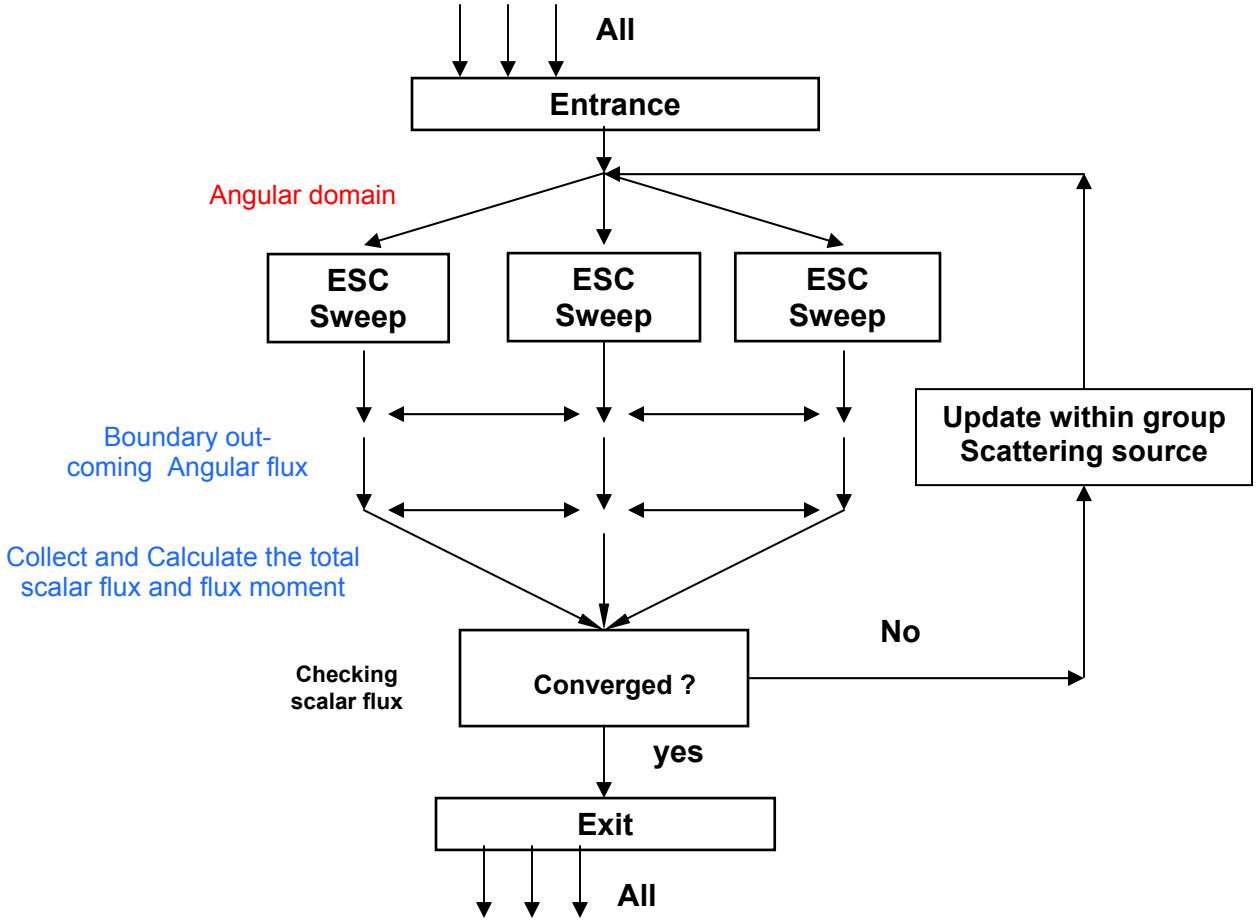

Fig. 6 Parallel computing flow for transport sweeps in NEWT

Figure 7 shows the reduction flow in GEMINEWTRN and NEWT to obtain the final sum from the partial sums which are distributed in each processors of a Processors Group, assuming there exists 8 processors totally. The partial sum data is first divided into the multiple parts to reduce the communicational burden as in Figure 6 where 2 parts are assumed. At the first step, the first part of the data is sent from processors 4, 5, 6 and 7 to 0, 1, 2 and 3, and the second part in the reverse ward. And then, all processors perform "add" operation to reduce the receiving data to the original data. At the second step, the first part of the reduced data at the first step is sent from processors 2 and 3 to 0 and 1, and the second part from 6 and 7 to 4 and 5. And then, the "add" operation is performed by processors 0, 1, 4 and 5. At the third step, the first part of the reduced data at the second step is sent from 1 to 0 and the second part from 5 to 4. The final part sums are obtained by "add" operation of processors 0 and 4. The complete set of reduction operation is obtained just by transmitting the final second part data from processor 4 to 0.

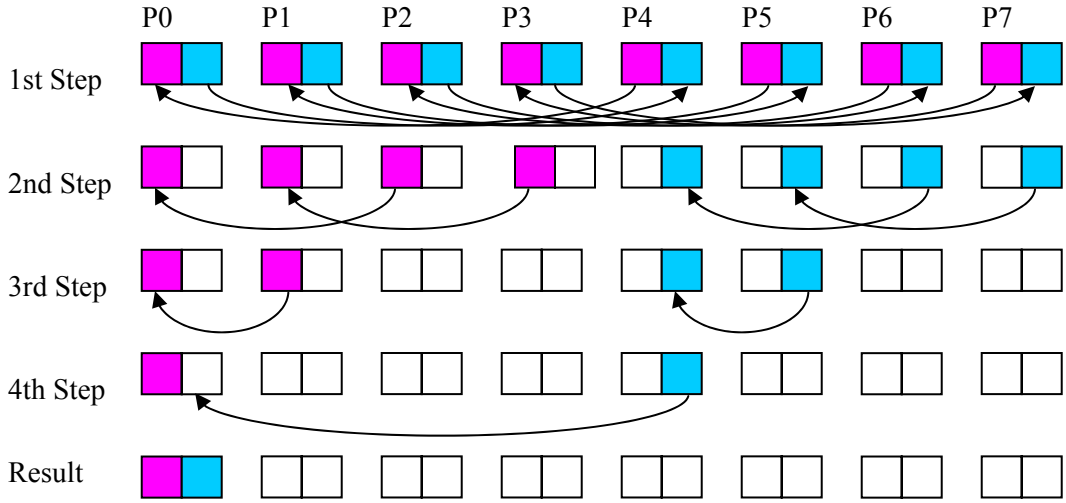

Figure 7 Distributed All-to-one Data Reduction Scheme

To analyze the performance of this data reduction scheme, let's assume the data size of L bytes, the communicational speed S bytes per second (bps), the communication setup time T_0 sec, and the "add" operational time T_1 for L bytes. Then processor 0 requires the following time to obtain the final complete data set.

$$\text{The total communication time} = 4\left(\frac{0.5L}{S}+T_0\right)=\frac{2L}{S}+4T_0$$

$$\text{The total "add" operational time} = 3\times 0.5T_1 = 1.5T_1$$

$$\text{The total reduction time} = \frac{2L}{S}+4T_0+1.5T_1 \qquad (9)$$

Let's compare the above results with other serial scheme, where processor 0 performs the communication and "add" operation with all the processors 1, 2, …, 7, one by one. In this case, the total reduction time can be calculated as follows:

$$\text{The total communication time} = 7\left(\frac{L}{S}+T_0\right)$$

$$\text{The total "add" operational time} = 7T_1$$

$$\text{The total reduction time} = 7\left(\frac{L}{S}+T_0+T_1\right) \qquad (10)$$

By comparing the above two Equations (9) and (10), one can find that Equation (9) is at least 2 times more efficient than Equation (9) in the case that the communication setup time is heavy, and at most 4.5 times efficient in the case that the communication time is trivial.

The total size of data transferred in the transport sweeps at each energy point can be expressed as following:

$$N_{commun} = O\left(n_{bc} n_{dir} n_{it} + N_{numcells} N_{mom} \log\left(n_{pro}\right)\right) \quad (11)$$

And the total amount of computation of each processor spent on the transport sweeps for current energy point can be expressed approximately as :

$$N_{comp} = O\left(N_{numcells} n_{dir} n_{it} / n_{pro}\right) \quad (12)$$

In Eq. (11) and (12), the meaning of each variables is :

N_{commun} = Total size of data communication.
n_{bc} = Total number of boundary surfaces.
n_{dir} = Total number of angles.
n_{it} = Total number of iterations for the current energy point.
$N_{numcells}$ = Total number of cells in the model.
N_{mom} = Total number of flux moments (dependent on the P_N scattering).
n_{pro} = Total number of processors.

The efficiency of parallel algorithm depends on the ratio of communication amount over computation amount, as expressed in Eq.(13), the lower the ration, the better the efficiency.

$$R = \frac{N_{commun}}{N_{comp}} = O\left(\frac{n_{bc}}{N_{numcells}} + \frac{\log\left(n_{pro}\right) n_{mom}}{n_{dir} n_{it}}\right) n_{pro} \quad (13)$$

From Eq. (14) it ‘s obvious that for large model, the first term can be omitted and the ration is dominated by the second term, as following:

$$R = \frac{N_{commun}}{N_{comp}} = O\left(\frac{\log\left(n_{pro}\right) n_{mom}}{n_{dir} n_{it}}\right) n_{pro} \quad (14)$$

It can be seen that the more processors used, the worse the efficiency and the higher the P_N order, the worse the efficiency; the higher the S_N order, the better the efficiency, and the tighter the convergence criteria, the better the efficiency.

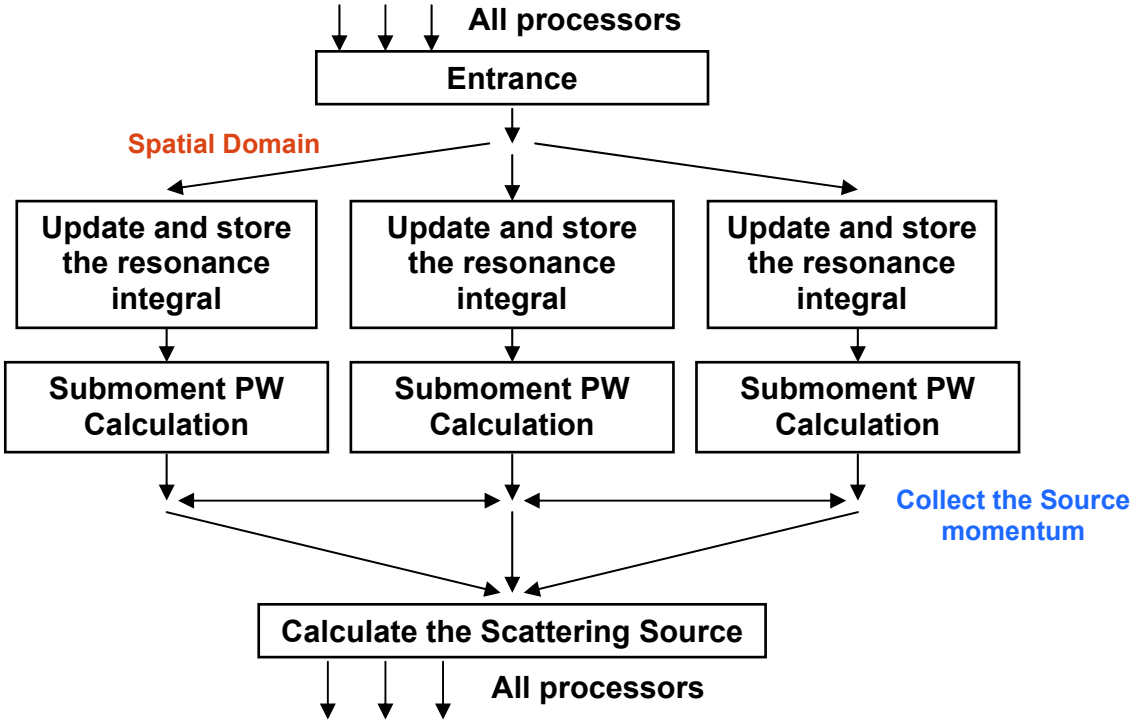

Fig. 8 Parallel computing flow for PW scattering source calculation in GEMINEWTRN

Fig. 8 shows the parallel computing flow for PW scattering source calculation, each processor is assigned with an spatial sub-domain, taking a branch of calculation in parallel. After all the processor finish the PW scattering source calculation using submoment expansion method within the spatial sub-domain, they will exchange data to collect the PW scattering source for the whole spatial domain. Similar with that parallel algorithm in transport sweep, performing a all-to-one reduction, the master processor collects data from al the other processors, after processing the data, performing a one-to-all broadcast, the master processor distribute the processed data to all the other processors.

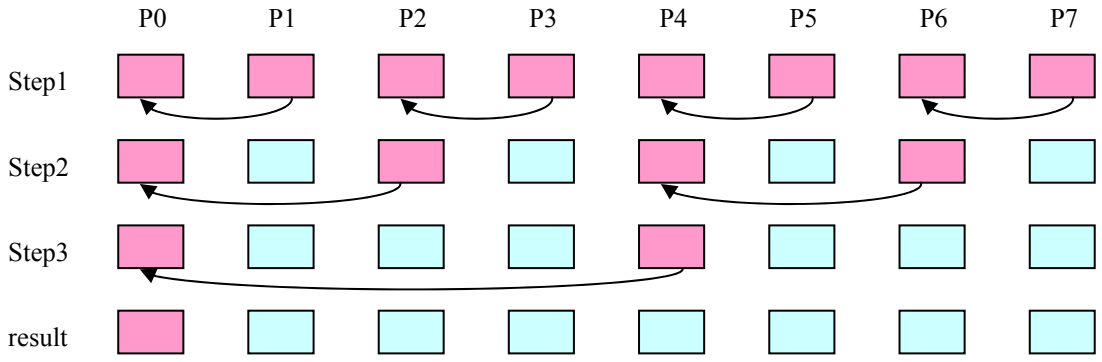

Fig. 9 Double-Tree Structure All-to-one reduction

In the all-to-one reduction in parallelization of PW scattering source, the main processor needs to collect data from all the other processors; however, the data buffer needed to be transferred are already spitted onto each processor associated with spatial decomposition, so that the total amount of data transferred is independent with the number of processors and communication algorithm. Still for the problem consisting 8 processors, performing the double-tree structure data reduction step by step, finally the main processor can get all the data needed.

Let's still assume the total data size of data buffer is L bytes so that the data size assigned onto each processor before communication is L/8 , the communicational speed S bytes per second (bps), the communication setup time T_0 sec.

$$\text{The total reduction time} = \frac{7L}{8S} + 3T_0 \qquad (15)$$

Let's compare the above results with other serial scheme, where processor 0 performs the communication with all the processors 1, 2, …, 7, one by one. In this case, the total reduction time can be calculated as follows:

$$\text{The total reduction time} = \frac{7L}{8S} + 7T_0 \qquad (16)$$

Comparing the above two Equations (11) and (12), it can be found although the active communication time can not be saved, the double-tree structure reduction scheme can reduce the communication overhead, and it's helpful especially when communication setup time is heavy.

The total size of data transferred and total computation amount in the PW scattering source calculation at each energy point can be expressed as following:

$$N_{commun} = O\left(N_{numcells} N_{mom}\right) \qquad (13)$$

$$N_{commun} = O\left(\frac{N_{numcells} N_{mom}}{n_{pro}}\right) \qquad (14)$$

An the ratio of communication amount over computation amount is expressed :

$$R = \frac{N_{commun}}{N_{comp}} = O\left(n_{npro}\right) \qquad (15)$$

It can be seen that the more processors used, the worse the efficiency of the parallel algorithm in the PW scattering source calculation.

Parallel Performance

To examine the parallel performance, the GEMINEWTRN code is applied to an BWR GE12 full assembly model, Figure 10 shows the configuration of the assembly model.

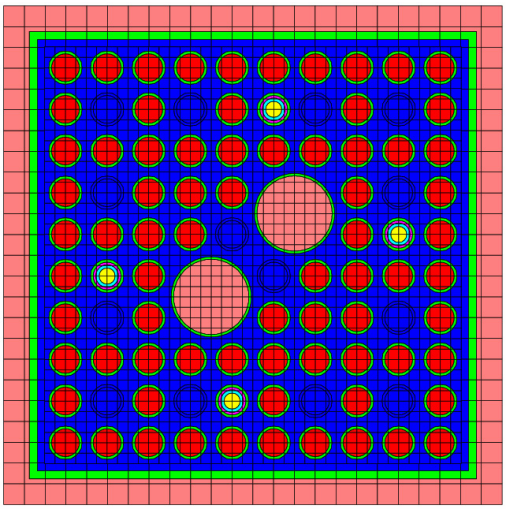

Fig. 10 GE12 BWR Assembly Model

For the fuel assembly model shown in Fig.11, there exists more than 4,000 cells (flat source regions) totally, and the total number of boundary surfaces is 96; there exists more than 32,000 energy points in the resonance region; to save memory, only fresh fuel (UO2 and gadolinium rod) is used.

GEMINEWTRN spectra calculation is performed for this model, to study the speedup performance, both S4/P1 and S8/P1 calculation are performed on CPILE Linux Cluster, the computation and communication time are listed in Table 3 and 4.

Table 3 Computation and communication time of GEMINEWTRN (S4/P1)

Number of Processors	Transport Sweep			[2)]Scattering Source			Total		
	[1)]T_{Com} min.	Total T min.	speedup/ (efficiency)	[1)]T_{Com} min.	Total T min.	speedup/ (efficiency)	[1)]T_{Com} min.	Total T min.	speedup/ (efficiency)
1		636.4			54.0			690.4	
2	6.5	314.1	2.03/(1.02)	2.8	29.7	1.82/(0.91)	9.3	343.8	2.01/(1.01)
3	11.0	213.2	2.98/(0.99)	3.0	22.6	2.38/(0.80)	14.0	235.8	2.93/(0.98)
4	11.0	163.1	3.90/(0.98)	3.1	16.5	3.28/(0.82)	14.1	179.6	3.84/(0.96)
6	12.2	113.4	5.61/(0.94)	3.8	13.8	3.91/(0.66)	16.0	127.2	5.43/(0.91)
12	13.8	65.2	9.86/(0.82)	4.7	9.9	5.04/(0.42)	18.5	75.1	9.19/(0.77)

Table 4 Computation and communication time of GEMINEWTRN (S8/P1)

Number of processors	Transport Sweep			[2)]Scattering Source			Total		
	[1)]T_{Com} min.	Total T min.	speedup/ (efficiency)	[1)]T_{Com} min.	Total T min.	speedup/ (efficiency)	[1)]T_{Com} min.	Total T min.	speedup/ (efficiency)
1		2257.9			70.4			2328.3	
2	15.6	1163.0	1.94/(0.97)	2.1	38.2	1.84/(0.92)	17.8	1201.2	1.94/(0.97)
4	18.5	577.9	3.91/(0.98)	3.3	22.2	3.17/(0.79)	21.8	600.1	3.88/(0.97)
8	19.5	292.3	7.72(0.97)	3.8	14.2	4.95/(0.62)	23.3	306.5	7.59/(0.95)
10	20.5	241.8	9.34/(0.93)	4.7	12.1	5.81/(0.58)	25.3	253.9	9.17/(0.92)
20	22.4	134.0	16.9/(0.84)	6.3	11.7	6.04/(0.30)	28.7	145.4	16.0/(0.80)

1) Communication time
2) PW Scattering source calculation

Fig. 11 and 12 show the parallel performance curve of S4/P1 and S8/P1 calculation. In S4 calculation, there only exists 12 angles, so that the maximal number of processors we can use is 12; in S8 calculation, there exists 40 angles, so up to 40 processors can be used in parallel computation.

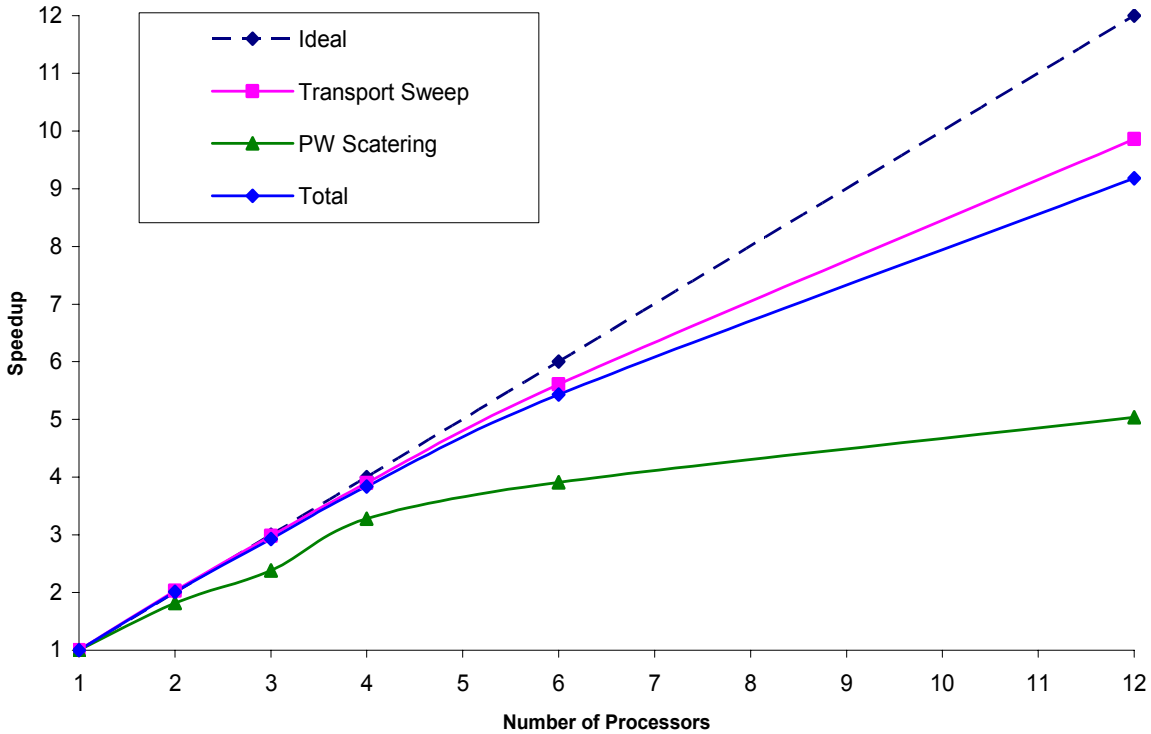

Fig. 11 Speedup performance of GEMINEWTRN (S4/P1)

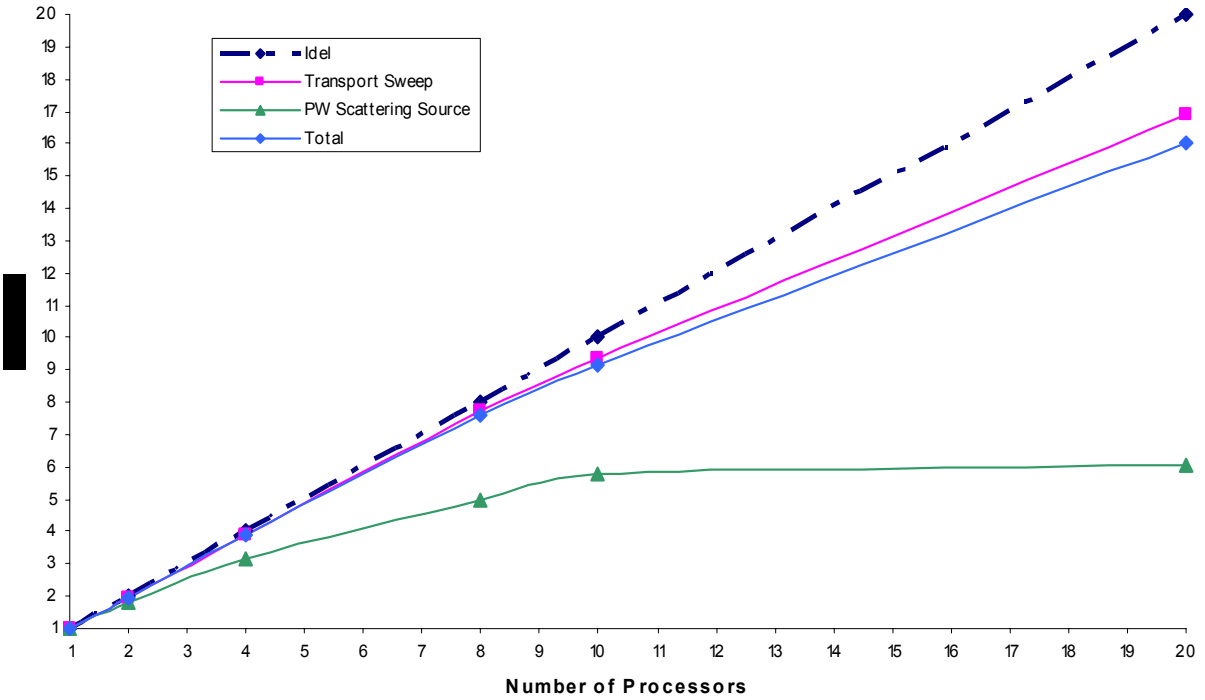

Fig. 12 Speedup performance of GEMINEWTRN (S8/P1)

A comparison of the speedup performance between S4 and S8 calculation is also performed, as shown in Fig.9, and the results are consisted with what we analyzed previously as shown in Eq. (9).

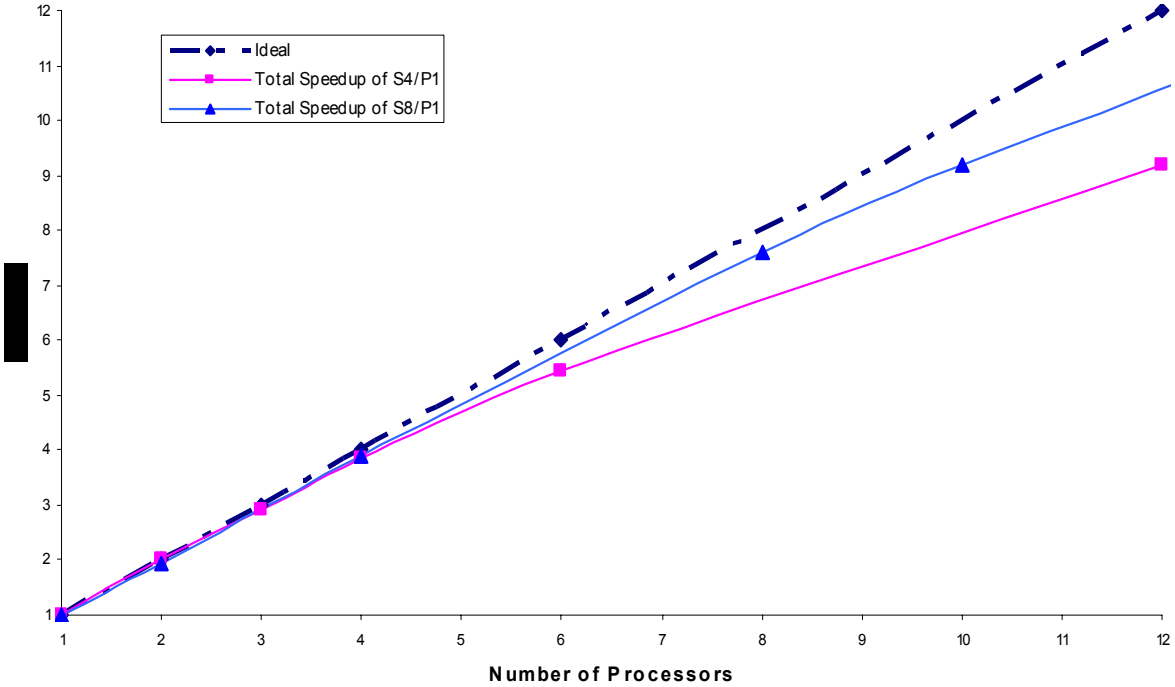

Fig. 13 Comparison of the Speedup performance between S4 and S8 Calculation

The same BWR GE12 full assembly model shown in Fig. 10 is selected to examine the parallel performance of NEWT, S8/P1 calculation is performed using an ENDF/B-V 44g lib. Fig.14 is showing the total speedup curve of NEWT

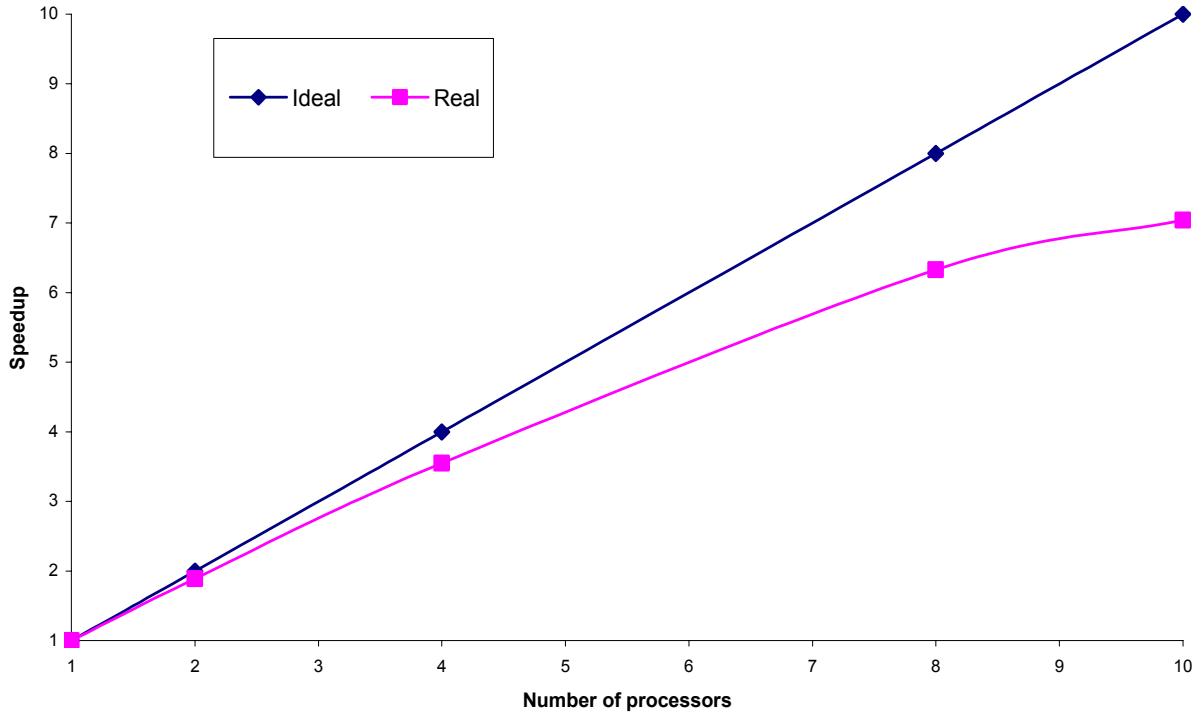

Fig.14 Speedup performance of NEWT (S8/P1)

Conclusion

From the previous results shown in Fig.10, 11 and 12, it can be seen that for S4/P1 calculation, when using 12 processors (the maximal number of processors can be used), the efficient is about 77%; for S8/P1 calculation, when using 20 processors the total parallel speedup efficient is 80%, The parallel efficiency is good. The speedup curve of the scattering source goes down quickly because only fresh fuel is used in this model, and total communication spent on the scattering source calculation is not that large.

Comparing the result shown in Fig.12 and 14, it can be easily seen that the speedup efficiency of NEWT is worse than that of GEMINEWTRN, although they are using the same S8/P1 option, that's because in the transport sweep of NEWT, more data communication is required to update the within group scattering source.

Reference

1. Z. Zhong, T. J. Downar, M. D. DeHart and M. L. Williams, "Continuous-Energy Multidimensional S_N Transport for Problem-Dependent Resonance Self-Shielding Calculation ", ANS 2005 Summer meeting, San Diego, CA June 2005
2 M. L. Williams, M. Asgari, D. F. Hollenbach "CENTRM: A One –Dimensional Neutron Transport Code For Computing Pointwise Energy Spectra," NUREG/CSD-2/V1/R7, Vol.2, Sec.F18, Oak Ridge National Laboratory (2004).
3 M. D. DeHart, " NEWT: A New Transport Algorithm For Two-Dimensional Discrete –Ordinates Analysis in Non-Orthogonal Geometries ," NUREG/CSD-2/R7, Vol.2, Sec.F18, Oak Ridge National Laboratory (2004).
4 Z. Zhong, T. J. Downar, M. D. DeHart " Implementation of a two-level Coarse-Mesh Finite-Difference Accelerator in the NEWT Transport code", ORNL/TM-2004/162, Oak Ridge National Laboratory (2005).