Vol.2020-HPC-173 No.9
2020/3/17

アンサンブル降雨流出氾濫モデルの高速処理方式

根本利弘[1]　中村要介[2]　安川雅紀[1]　小池俊雄[2]　池内幸司[1]　喜連川優[3]

概要：中小河川を対象とした水位予測システムの構築を目指し，研究・開発を行っている．このシステムでは，降雨流出と洪水氾濫を一体的に取り扱い，降雨情報から流域全体でどの地域がどのように氾濫するかを比較的軽量・高速に予測する降雨流出氾濫モデル（Rainfall-Runoff-Inundation: RRI モデル）を用いているが，予測精度を高めるために，初期状態に擾乱を加えてアンサンブル予測するとともに，予測の開始を過去の時点から開始し，過去から現時点までの予測結果と実際の水位の観測データにより各アンサンブルメンバの尤度を求め，予測結果の選択，補正をする．RRI モデルによるアンサンブル予測では，初期状態や入力データにより処理時間に偏りが生じ，システム全体の予測時間の短縮の妨げとなっている．本発表では，RRI モデルのアンサンブル予測の実行時の処理時間の偏りについて述べるとともに，計算の終了したアンサンブルメンバに割り当てられていたプロセッサを計算が終了していないアンサンブルメンバに割り当て，全体の予測時間を短縮する方式について説明し，処理実験によりその効果を示す．

Fast Execution of ensembles of rainfall runoff inundation model

TOSHIHIRO NEMOTO[†1]　YOSUKE NAKAMURA[†2]　MASAKI YASUKAWA[†1]
TOSHIO KOIKE[†2]　KOJI IKEUCHI[†1]　MASARU KITSUREGAWA[†3]

1. はじめに

近年，台風や豪雨による河川氾濫が起こり，多くの人的被害が発生している．大雨や短時間強雨の発生頻度は増加傾向にあり[1]，河川氾濫の発生リスクは今後ますます高まると考えられている．河川氾濫への対策はもはや治水ダムや堤防の強化等のハードウェアのみでは不十分であり，適切な警告や情報を提供するソフトウェア面での対策も合わせて行うことは不可欠である．このような背景のもと，中小河川を対象とした水位予測システムの構築を目指し，研究・開発を行っている[2][3][4]．洪水時の水位予測には高度な技術，高性能なシステムが要求されるため，多くの中小河川では洪水予測が実施されていないのが現状である．一方，中小河川では，大河川に比べて流域の豪雨により急速に水位が上昇するため避難が遅れる危険性は高い．正確な予測を高速に行い，前もって適切に警告，情報を自治体，住民に提供することにより，避難のための十分な時間の確保が可能となる．

中小河川を対象とした水位予測システムでは，中小河川の数が多いことにより，一河川当たりのコストが小さい，すなわち少ない計算機資源で予測を行うことが望まれている．そこで，本中小河川を対象とした水位予測システムでは，降雨流出と洪水氾濫を一体的に取り扱い，降雨情報から流域全体でどの地域がどのように氾濫するかを比較的軽量・高速に予測する降雨流出氾濫モデル（Rainfall-Runoff-Inundation: RRI モデル）[5]を用い，さらに予測精度を高めるために，初期状態に擾乱を加えてアンサンブル予測する．予測の開始を過去の時点から開始し，過去から現時点までの予測結果と実際の水位の観測データにより各アンサンブルメンバの尤度を求め，予測結果の選択，補正を行う．さらに，尤度に基づき次回予測のための初期値を作成し，直接観測ができない，あるいは困難であるモデル内の状態変数の精度を高める．RRI モデルではルンゲ-クッタ法によって収束計算を行なっており，初期状態や入力データによって実行時間に大きな偏りが生じるが，大多数のアンサンブルメンバの計算が完了しても全てのアンサンブルメンバの計算が完了しなければ予測結果を得ることができず，また，次回の予測を開始することもできない．本稿では，この RRI モデルのアンサンブル予測のメンバ間の実行時間の偏りについて述べるとともに，計算の終了したアンサンブルメンバに割り当てられていたプロセッサを計算が終了していないアンサンブルメンバに割り当て，全アンサンブルメンバの計算が完了するまでの時間を短縮する方式について説明し，処理実験によりその効果を示す．

2. 関連研究

近年の河川氾濫による災害の発生頻度の増加，被害の拡大を受け，降雨流出，洪水氾濫のモデルの高速実行に対する要望は高く，高速化に関する研究は少なくない．[6]は，シミュレーション実行中に計算領域を水の流れに合わせて動的に拡大・縮小し，処理量を低減することで高速化を図る手法を述べている．[7][8]は，河川流解析，氾濫解析に GPGPU を用いて高速化を行う手法を述べ，その有用性を示した．また，[9]は，計算対象とする河川流域を分割して複数のノードで並列化し，また各ノードにおいては GPGPU によって計算性能を向上させる手法について検討を行って

1 東京大学
The University of Tokyo
2 土木研究所
Public Works Research Institute
3 東京大学/国立情報学研究所
The University of Tokyo / National Institute of Informatics

いる．これらの研究はいずれも 1 つのシミュレーションを高速に実行することを目的とした手法である．一方，本稿で述べる手法は，アンサンブル計算を実行する際に，そのアンサンブルメンバ間の実行時間の偏りに着目して計算時間の短縮を図るものであり，これらの関連研究とは異なる．

3. 降雨流出氾濫モデルのアンサンブルによる中小河川水位予測システム

3.1 概要

構築を行っている中小河川を対象とした河川水位予測システムでは，定常的に現在時刻（基準時刻）から 6 時間後までの水位を予測し，現在までの水位および予測された水位のグラフを WWW により提供するとともに，予測された河川水位が氾濫危険水位，避難判断水位等のあらかじめ設定された閾値を超えた場合には，メール等により警告を発する．水位は RRI モデルによるシミュレーションにより予測し，RRI モデルの入力として必要な降水データは気象庁により配信される降水短時間予報[11]を利用する．RRI モデルによるシミュレーションでは，斜面水深，河道水深等の初期状態量を与えることで，助走計算なしで予測精度を上げることが可能となるが，対象領域全域の斜面水深，河道水深をリアルタイムで観測，取得することは極めて困難である．そこで，本河川水位予測システムでは，予測精度を高めるため，RRI モデルの初期状態量として与える斜面水深に対して擾乱を加え，64 メンバのアンサンブルによりシミュレーションを行う．この際，現在時刻をシミュレーションの開始時刻とするのではなく，現在時刻から 3 時間前をシミュレーションの開始時刻とし，その時刻から 9 時間先までのシミュレーションを行い，現在時刻から 6 時間先まで予測する．RRI モデルの入力データとする降水データは，3 時間前から現在時刻までは降水短時間予報と同様に気象庁から配信される解析雨量[12]を用い，現在時刻から 6 時間後までは降水短時間予報を用いる．3 時間前から現在時刻までの水位観測所で実測された水位と，これら 64 個のアンサンブルによる 3 時間前から現在時刻までの当該地点の水位により各アンサンブルの尤度を求め，その尤度に基づき各アンサンブルメンバの予測結果の選択，重みづけを行い，システムとしての予測結果とする．解析雨量，降水短時間予報は 30 分毎に配信されており，本河川水位予測システムにおいても，30 分毎に最新の降水データを用いて河川水位予測を行っている．河川水位予測結果と同様に，各アンサンブルの尤度に基づき，シミュレーション開始時刻から 30 分後の時刻における斜面水深，河道水深等の状態変数を選択，重みづけし，次回の RRI モデルの入力値とする．このようにアンサンブルを用いることで予測精度を高めるとともに，RRI モデルの状態量についても現状に即した値に近づける．全てのアンサンブルメンバの結果がそろわなければシステムとしての予測結果，次回の RRI モデルの実行のための初期状態量を取得することができず，遅延なく水位予測を継続するためには全アンサンブルの実行を 30 分以内に完了する必要がある．

3.2 降雨流出氾濫モデル（Rainfall-Runoff-Inundation: RRI Model）

RRI モデルは，土木研究所において公開されている ver.1.4.2.3 を利用している[12]. 公開されている RRI モデルは OpenMP により部分的に並列化されているが，これに加え，予測結果に影響を与えない範囲において，並列稼働箇所の追加，一部処理の効率化といった改良を行った．RRI モデルでは，ルンゲ-クッタ法による収束計算を実行して方程式を解き，河川水位予測を行うが，入力値である初期状態量や降水データにより収束に要する繰り返し回数が異なり，処理時間に差が生じる．特に豪雨など激しい降水が予測される場合には，内部の状態量の現在値から予測値への変化が大きいために収束に要する繰り返し回数が増加し，実行時間が長くなる傾向がある．以降，アンサンブルメンバ間（初期状態量）の実行時間の偏り，予測開始時刻（降水データ）による処理時間の偏りについて実際のシミュレーション結果により示すとともに，RRI モデルの並列処理の効果について示す. シミュレーションの対象は平成 29 年 7 月九州北部豪雨時の大分県日田市の花月川である．具体的なパラメータを表 1 に示す．また，最大 8 ノードの物理サーバを利用し，1 物理サーバ当たり 1 台の仮想サーバを稼働させ，仮想計算機上で 64 メンバのアンサンブルのシミュレーションを並列実行した．1 ノード当たりのホスト計算機，ゲスト計算機の諸元を表 2 に示す.

表 1　RRI モデルシミュレーション対象
Table 1　Target of RRI model simulation

期間	2017 年 7 月 5 日 0:00～2017 年 7 月 6 日 12:00
南西端位置	東経 130.82027779764 北緯 33.080277755507
格子サイズ	880×696
解像度	0.00055555556900° /1 格子

表 2　RRI モデルシミュレーション実行環境
Table 2　Execution environment of RRI model simulation

ホスト	CPU	Intel Xeon Gold 6140 (2.3GHz)×2 36 物理コア/72 論理コア（ハイパースレッド）
	メモリ	DDR4-2666 256GB
	OS	CentOS 7
ゲスト	CPU	72 コア
	メモリ	112GB
	OS	CentOS 7

3.3 アンサンブルメンバ間の実行時間の偏り

図 1 は，基準時刻（現在時刻）が 2017 年 7 月 6 日 0:30 のときの，各アンサンブルメンバの RRI モデルシミュレーションの実行時間を示したものである．64 個のアンサンブルを，1 アンサンブルメンバを 1 プロセスとして均等に 8

つのノードに分散して割り当て，各アンサンブルメンバを 4 スレッドおよび 8 スレッドで並列実行する場合，すなわち各ノードでは 8 つのアンサンブルメンバが並列に動作し，計 32 スレッドまたは 64 スレッドが同時に使用される場合の各アンサンブルメンバの RRI モデルによるシミュレーション実行時間である．この図によると，各アンサンブルメンバの実行時間には大きな偏りがあることが見て取れる．大部分のアンサンブルメンバは実行時間に大きな差はないものの，少数のアンサンブルメンバの実行時間が突出して長いことが分かる．全てのアンサンブルメンバの実行が完了し，シミュレーション結果がそろわなければ次の基準時刻の処理に進むことはできず，全体の実行時間は最も実行時間が長いアンサンブルメンバの実行時間によって決定されるが，少数のアンサンブルメンバの実行時間が突出しているために，全体の実行時間が長くなってしまうことが分かる．30 分毎の逐次処理を遅延なく継続するためには，長くとも 30 分以内には 1 回の水位予測計算を完了しなければならないが，この図に示した場合においては，1 アンサンブルメンバ当たり 4 スレッドを割り当てたときには全アンサンブルメンバ中 2 メンバが，8 スレッドを割り当てた場合にはわずか 1 メンバが 30 分以内に実行を完了できず，全体として遅延が生じることになってしまう．

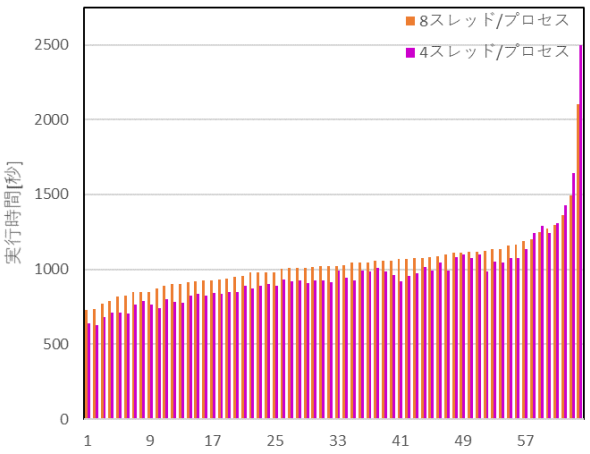

図 1　アンサンブルメンバ毎の実行時間の分布
Figure 1　Distribution of execution time of ensemble members.

1 アンサンブルメンバ当たりのスレッド数に着目すると，実行時間の短いアンサンブルメンバにおいては，1 アンサンブルメンバ当たり 4 スレッドを割り当てた場合のほうが 8 スレッドを割り当てた場合よりも実行時間は短くなるが，実行時間の長いアンサンブルメンバにおいては，8 スレッドを割り当てた場合のほうが実行時間は短くなる．これはシミュレーションに用いた計算機の物理コアが 1 ノード当たり 36 コアであるため，8 スレッド割り当てた場合はアンサンブル計算の実行開始当初はノード当たり 64 スレッドが稼働し，リソース競合によるオーバーヘッドが大きく，各アンサンブルメンバの処理速度が低下するが，実行時間の短いアンサンブルが終了すると，リソースの競合がなくなり，より並列度の高い 8 スレッドを割り当てた場合のほうが処理速度は高まり，実行時間の長いアンサンブルメンバの実行時間が短縮されるためである．

3.4　入力降水データ（基準時刻）による実行時間の偏り

図 2 は，64 個のアンサンブルを 8 ノードに均等に分散し，1 アンサンブルメンバ当たり 8 スレッドを割り当てた場合の，基準時刻（現在時刻）による RRI モデルのシミュレーションの実行時間の推移のグラフである．2017 年 7 月 5 日の午後より実行時間が大きく増加しているが，これはこのころより降雨が激しくなったことによる．7 月 5 日の午前は降水がほとんどなく，モデルの内部状態量の変化が小さいために少ない反復計算で収束するのに対し，7 月 5 日の午後からは降水が激しくなり，そのためにモデルの内部状態量が大きく変化し，内部状態を収束させるための反復計算が増加したためと考えられる．全てのアンサンブルメンバの計算が終了した段階で 6 時間後までの水位の予測値が求められ，また，次の基準時刻の初期値も決定されて次の基準時刻の計算を行うことが可能となるが，7 月 5 日の午後においては最大実行時間が 30 分を超える場合が連続して生じており，洪水の危険性が高まり，水位予測を最も必要とする降水の激しい時間帯に遅延が累積してしまうことになる．

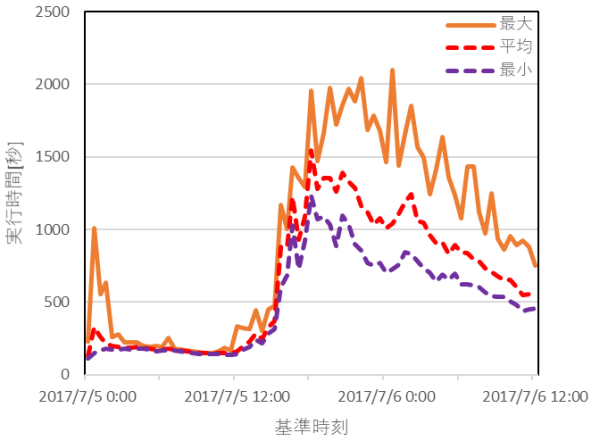

図 2　基準時刻による実行時間の変化
Figure 2　Transition of execution time associated with epoch time.

3.5　並列処理効果

3.5.1 単一アンサンブルメンバ実行時

図 3 は，アンサンブルメンバに計算に割り当てるスレッド数と，実行時間関係のグラフである．基準時刻は 3.4 節において実行時間が最大となった 7 月 6 日の 0:30 であり，その基準時刻において実行時間が最大となったアンサンブ

ルメンバの結果を示している．1 ノードを利用し，当該アンサンブルメンバのみを実行した場合の結果である．RRI モデルは内部計算においてグリッド間の依存を有する部分があり，最大でも 1/7 程度の実行時間の短縮にとどまってはいるが，20 スレッド程度まではスレッド数を増やすことにより実行時間の短縮が見込まれることが分かる．すなわち，CPU リソースが十分に備わっている環境においては，1 アンサンブルメンバに対して割り当てるスレッド数を増やすことにより，実行時間の短縮を図ることが期待できる．

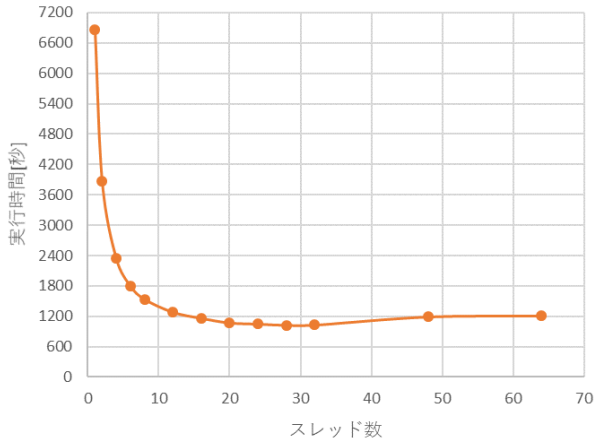

図 3　割り当てスレッド数による実行時間の変化
（単一アンサンブルメンバ実行時）
Figure 3　Execution time vs. number of assigned threads.
(Single ensemble member execution)

3.5.2 64 アンサンブルメンバ同時実行時

図 4 は 1 アンサンブルメンバ当たりに割り当てるスレッド数と，64 アンサンブルメンバの実行時間の最大値，平均値，最小値を示したグラフである．3.3 節，3.4 節の場合と同様，8 ノードを利用し，各ノードで 8 アンサンブルメンバを同時に実行している．すなわち，1 アンサンブルメンバ当たり 20 スレッドを割り当てている場合，1 ノード当たり 72 論理コアに対して計 160 スレッドが動作していることとなる．この図によると，実行時間の最小値が最も小さくなるのは 1 アンサンブルメンバ当たり 4 スレッドの場合である．実行時間が最小となるアンサンブルメンバが動作している間は，すべてのアンサンブルメンバが動作している状態，すなわち 1 ノード当たり 36 物理コアに対して計 32 スレッドが動作している状態である．すなわち，全アンサンブルメンバが動作しているときに，リソースの競合なしに最も多くのスレッドが割り当てられるのが 1 アンサンブルメンバ当たり 4 スレッドを割り当てる場合であり，したがってアンサンブルメンバ内で実行時間が最小となるアンサンブルメンバの実行時間が，1 アンサンブルメンバ当たり 4 スレッドを割り当てたときに最小となる．ているために実行時間が最小となっている．一方，アンサンブルメンバ内で実行時間の最大値が最小となるのは，アンサンブルメンバ当たりの割り当てスレッド数が 8 スレッドの場合である．これは，RRI モデルによるシミュレーションの実行開始直後はリソースの競合のために各アンサンブルメンバの処理速度が低下し，アンサンブルメンバ内の実行時間の最小値は増加するが，実行時間が短いアンサンブルメンバが終了し，動作しているアンサンブルメンバの数が減少するに従いリソースの競合が抑えられ，1 アンサンブルメンバ当たりに割り当てられたスレッドが有効に動作するようになり，実行時間が短縮されるためである．3.5.1 節によると CPU リソースが十分に利用できる場合には 20 スレッド程度までは実行時間が短縮されるが，8 ノード計 288 物理コアに制限されている本環境においては，シミュレーションの実行開始直後からの多数のアンサンブルメンバが動作している状態でのリソース競合による速度低下が，終盤の少数アンサンブルメンバのみが動作し，並列処理の効果が高まることによる実行時間の短縮を上回るため，1 アンサンブルメンバ当たりのスレッド数が 10 スレッド以上になると，かえって実行時間も増加することになる．

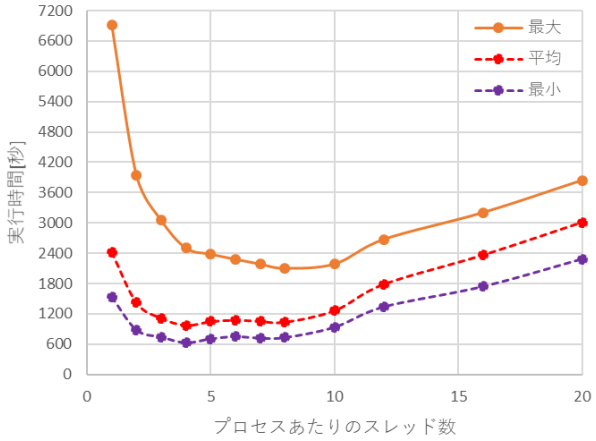

図 4　割り当てスレッド数による実行時間の変化
（64 アンサンブルメンバ同時実行時）
Figure 4　Execution speed vs. number of assigned threads.
(Concurrent execution of 64 ensemble members)

4. RRI モデルのアンサンブル実行における課題と高速化手法

4.1 RRI モデルのアンサンブルメンバ間の偏りと動的なスレッド割り当て

RRI モデルのアンサンブル実行を定常的に遅延なく実行する際の課題として，アンサンブルメンバ間の実行時間の偏りがある．全てのアンサンブルメンバの計算が終了し，その結果をもとに次のステップの計算の初期値が決められるため，実行時間が最大となるアンサンブルのメンバの実行時間が全体の実行時間となる．3.3 節の図 1 の例では，64 個のアンサンブルのうち，30 分以上の実行時間を要し

ているのはわずか 1 個か 2 個のアンサンブルメンバのみであるが，このために全体としては遅延が生じてしまうことになる．大部分のアンサンブルメンバの実行が完了し，少数のアンサンブルメンバが実行されている状態では，ノードが有する物理コアよりも少ないスレッドがアンサンブルメンバの実行に割り当てられており，プロセッサに余裕がある状態である．図 3 に示されるように 1 アンサンブルメンバ当たり 20 スレッド程度までは割り当てられるスレッド数が増えると実行時間の短縮がされるが，図 4 で見たように終盤で実行時間の長いアンサンブルメンバに割り当てられるスレッド数を多くしようとして実行開始時から物理コアを大きく越えるスレッド数を割り当ててしまうと，シミュレーション開始直後の多数のアンサンブルメンバが実行されている状態でリソースの競合が増え，実行時間の長いアンサンブルメンバの実行時間もより増加してしまうこととなる．

この問題に対し，RRI モデルのシミュレーションプログラムに改良を加え，実行時に動的に割り当てスレッド数を変更可能とし，CPU リソースの競合なく，かつ CPU リソースを余らせることがないよう，実行が完了したアンサンブルメンバに割り当てられていたスレッドを，まだ実行が完了していないアンサンブルメンバに割り当てることにより高速化し，全アンサンブルメンバの実行が完了するまでの時間の短縮を図る．RRI モデルのシミュレーションプログラムは，シミュレーション期間を設定されたタイムステップに分割し，タイムステップごとにグリッドの計算を行う．1 タイムステップのグリッドの計算が完了すると次のタイムステップに進むというように逐次的に実行され，グリッドの計算が OpenMP により並列化されている．そこで，各タイムステップの初めにスレッド数が書き込まれたファイルを読み込み，そのスレッド数を OpenMP の OMP_SET_NUM_THREADS 関数により設定することで RRI モデルの割り当てスレッド数を外部から変更可能とする．スレッド数設定ファイルが存在しない場合にはスレッド数の設定を行わず，この場合は変更前の RRI モデルと同様の動作となる．また，設定ファイルの読み込みエラーが生じた場合には，同様にスレッド数の変更は行わずに処理を進め，設定ファイルの書き換えに起因する読み込みエラー等を回避する．

アンサンブル実行は以下のように行う．まず，設定値としてノード内のアンサンブルメンバに割り当てる合計スレッド数および 1 アンサンブル当たりの最大スレッド数を定める．1 アンサンブル当たりの最大スレッド数を設定するのは，図 3 に示したようにスレッド数を増やしても実行速度の向上が見込めない，あるいはかえって実行速度が低下することを回避するためである．アンサンブルメンバへのスレッドの割り当ては，各ノードのスレッド管理プロセスが行う．スレッド管理プロセスはノード内の割り当て可能な合計スレッド数を均等にアンサンブルメンバに割り当て，アンサンブルを実行する．アンサンブルメンバの実行が終了すると，管理プロセスは，終了したアンサンブルメンバに割り当てられていたスレッドを，稼働中のスレッドに均等に配分し，稼働中のアンサンブルメンバのスレッド割り当てファイルを書き換える．この際，実行されるアンサンブルメンバが少数になり，スレッドが余っていても，1 アンサンブルメンバ当たりの最大スレッド数以上に割り当ては行わない．

具体的な例として，合計スレッド数を 32，最大スレッド数を 20 とし，1 つのノードにおいて 8 アンサンブルメンバを実行する場合を示す．管理プロセスはまず，各アンサンブルメンバに 4 スレッドを割り当て，各アンサンブルメンバのスレッド数設定ファイルに 4 を記載して実行する．その後，最初のアンサンブルメンバが終了すると，割り当てられていた 4 スレッドを他の 7 アンサンブルメンバに割り当てる．すなわち，4 つのアンサンブルメンバは 5 スレッドで動作するよう設定され，3 つのアンサンブルメンバは 4 スレッドで動作するよう設定される．次のアンサンブルメンバが終了すると，そのアンサンブルメンバに割り当てられていたスレッドが稼働中のアンサンブルメンバに割り当てられ，2 つのアンサンブルメンバが 6 スレッド，4 つのアンサンブルメンバが 5 スレッドで動作するよう設定される．残りのアンサンブルメンバが 2 個となったときには，それぞれ 16 スレッドが割り当てられ，残り 1 スレッドになると 20 スレッドが割り当てられる．すなわち，1 スレッド当たりの最大割り当てスレッド数に達しない限り，全 32 スレッドが稼働中のアンサンブルメンバに均等に割り当てられて動作する．このようにして，リソースの競合を起こさず，有効にプロセッサを利用することを可能にし，アンサンブル実行を高速化する．

4.2 RRI モデルの入力降水データ（基準時刻）による偏りと動的なノード割り当て

RRI モデルでは，降水がない場合にはモデル内の状態量の変化が小さく，収束計算の繰り返し回数が少なくなり，実行時間は短くなる．一方で，激しい降雨がある場合などは内部状態量も大きく変化することとなり，収束するまでの計算量が増え，実行時間は長くなる傾向がある．すなわち，河川の氾濫の危険性が高く，可能な限り早期に予測が必要なときに最も実行時間が長くなってしまうため，水位予測システムを有効なものとするためには，実行時間が最長となる場合を想定し，その場合でも設定時間内に予測計算が終わるように十分な計算機リソースを割り当てておく必要がある．一方で，一つの河川に着目した場合，河川の氾濫を起こすような豪雨は年に一度あるかないかのことであり，それ以外のときは激しい降水がなく，大部分の計算機リソースは余剰となってしまう．そこで，負荷状況に応じて RRI モデルのアンサンブルを実行するノード数を動的

に変更し，計算機リソースの有効利用を図る．割り当てられなかった余剰な計算機は停止することで省電力を図ることも，あるいは別の河川予測のために利用することも可能である．特に全国にある多数の中小河川の水位予測を行う場合には，全国で同時に激しい豪雨となることはなく，ある河川が計算機リソースを必要となった場合には，他の降水がほとんどない地域の河川のための余剰計算機リソースを利用でき，より少ない計算機で多数の河川の水位予測を行うことが可能となる．RRI モデルのアンサンブルを実行するノード数の変更は，実行時間に上限値，下限値の 2 つの閾値を設定し，実行時間が上限値を超えた場合には次の基準時刻における割り当てノード数を増やし，下限値を下回った場合には割り当てノード数を減らすことで行う．RRI モデルのアンサンブルの実行時間は，図 2 に示されるように徐々に増減することから，実行時間の上限値，下限値，増減させるノード数を適切に設定することにより，負荷の変動に十分に対応できるものと考えられる．

5. RRI モデルの高速化手法の評価

5.1 RRI モデルのアンサンブルメンバへの動的なスレッド割り当て

4.1 節において提案した RRI モデルのアンサンブルメンバへの動的なスレッド割り当て手法を実装し，3 節において示したものと同一のパラメータ，実行環境において，RRI モデルのアンサンブルを実行した．図 5，図 6 は，64 個のアンサンブルを均等に 8 つのノードに分散し，各アンサンブルメンバに割り当てられるスレッド数を動的に変更した場合の，各アンサンブルメンバの実行時間である．ノード当たりの最大合計割り当てスレッド数は 32 スレッド，1 アンサンブルメンバ当たりの最大割り当てスレッドは 20 スレッドとしている．図 1 で示した静的なスレッド割り当ての結果も重ねて表示しており，図 5，図 6 はそれぞれ 1 アンサンブルメンバ当たり 8 スレッドおよび 4 スレッドを割り当てた場合との結果の重ね合わせである．1 アンサンブルメンバに対して 8 スレッドを固定して割り当てた場合との比較である図 5 を見ると，動的なスレッドの割り当てを行った場合には，実行時間が短いアンサンブルメンバから実行時間が長いアンサンブルメンバの全てのアンサンブルメンバにおいて実行時間が短縮されていることが分かる．動的にスレッドを割り当てた場合には，全てのアンサンブルメンバが動いている状態では 1 アンサンブルメンバ当たり 4 スレッドが割り当てられており，リソースの競合によるオーバーヘッドが解消されるため，全アンサンブルメンバの実行速度が向上する．さらに実行時間の長いアンサンブルメンバについては，動作しているアンサンブルメンバが少なくなった際に，割り当てられるスレッド数が 8 スレッドよりも多くなるため，より一層実行時間の短縮が得られる．1 アンサンブルメンバ当たり 4 スレッドを静的に割り当てた場合との比較である図 6 では，実行時間の短いアンサンブルメンバにおいては実行時間の短縮が見られないが，これは実行時間の短いアンサンブルメンバが動作している間は，ほぼすべてのアンサンブルメンバが動作している状態であり，このときには動的なスレッド割り当てを行っている場合でも，各アンサンブルメンバの割り当てスレッド数は 4 スレッドと静的に割り当てた場合と違いがないためである．動作するアンサンブルメンバが少なくなってくると，1 つ当たりのアンサンブルメンバに割り当てられるスレッド数は増加するため，実行時間が長いアンサンブルメンバの実行時間は短縮される．

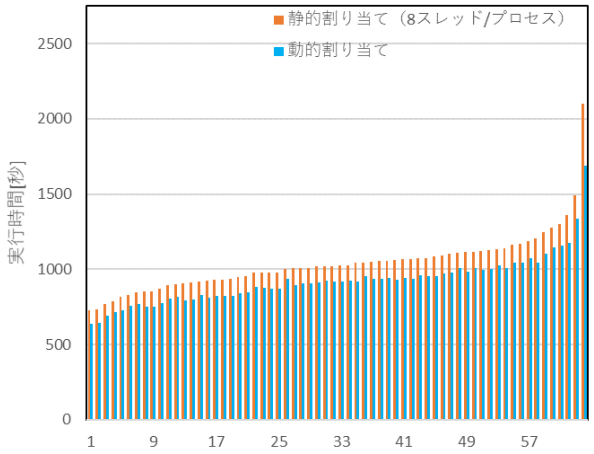

図 5　動的にスレッドを割り当てたときのアンサンブルメンバ毎の実行時間の分布
（8 スレッド静的割り当てとの比較）

Figure 5　Distribution of execution time of ensemble members with dynamic thread assignment.
(Comparison with the case of 8 threads static assignment)

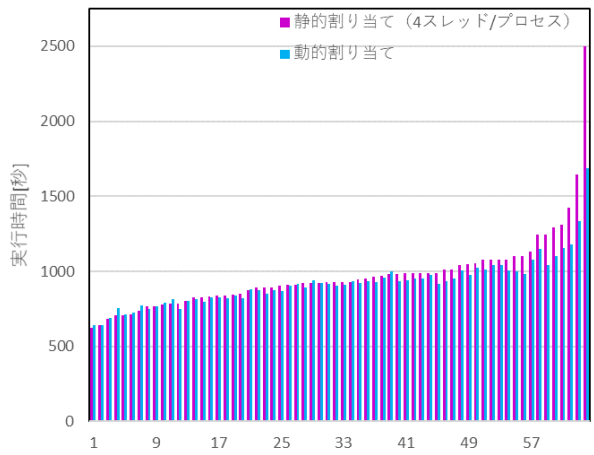

図 6　動的にスレッドを割り当てたときのアンサンブルメンバ毎の実行時間の分布
（4 スレッド静的割り当てとの比較）

Figure 6　Distribution of execution time of ensemble members with dynamic thread assignment.
(Comparison with the case of 4 threads static assignment)

図 7 は，アンサンブルメンバに割り当てるスレッドを動的に変更した場合，およびアンサンブルメンバ当たり 8 スレッドを固定して割り当てた場合の，基準時刻による実行時間の変化である．ほぼすべての基準時刻において実行時間が短縮されていることが分かるが，特に実行時間が長くなっている 7 月 5 日の 15 時以降において，短縮される時間が大きくなっている．河川氾濫の危険性が高まり，迅速な予測が必要とされる場合において，より効果的に実行時間が短縮されることが分かる．

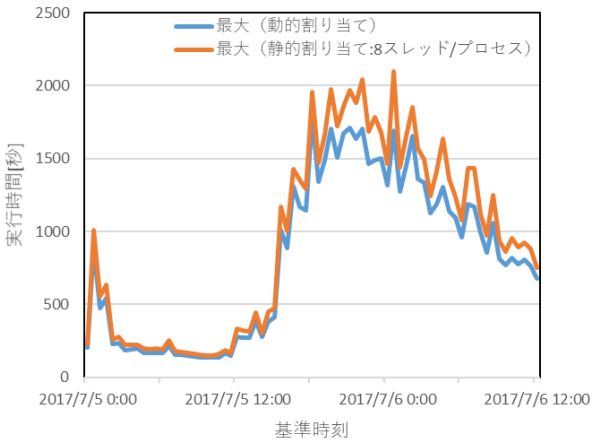

図 7　動的にスレッドを割り当てた場合の基準時刻による実行時間の変化

Figure 7　Transition of execution time associated with epoch time with dynamic thread assignment.

5.2 RRI モデルのアンサンブルへの動的なノード割り当て

図 8 は，1 アンサンブルメンバ当たりのスレッド数を動的に割り当て，さらに RRI モデルのアンサンブル計算を実行するノード数を動的に変化させた場合の，基準時刻による実行時間および割り当てられたノード数の変化を示したものである．ノード数を動的に変化させず，8 ノード固定とした場合の結果もあわせて示している．動的なノード数の割り当ては，初期状態では 8 ノードを割り当て，実行時間が 480 秒以下であれば次の基準時刻における割り当てノード数を 1 減らし，実行時間が 720 秒以上となった場合には次の基準時刻における割り当てノード数を 2 増やし，最低 2 ノード，最大 8 ノードまで増減させるよう設定した．降水がほぼなく，実行時間が短い 7 月 5 日の 15 時頃までは，割り当てノード数が 2〜3 ノードとなるまで，初期状態から割り当てられたノード数が大きく減らされていることが分かる．この間，割り当てられたノード数が減少したことにより実行時間は 500〜900 秒程度に増加しているが，次の基準時刻までの時間内である 1800 秒以内には計算は終了している．このころは激しい降水の予報はなく，河川氾濫の危険性，水位予測の緊急性は小さい．一方，7 月 5 日の 15 時以降は，激しい降水の予報により RRI モデルのアンサンブルの実行時間が増し，それにともない割り当てられるノード数も増加し，7 月 5 日 13:30 以降は上限の 8 ノードが割り当てられており，水位予測の緊急性が高い際には最大限のリソースにより RRI モデルのアンサンブル実行が行われていることが分かる．

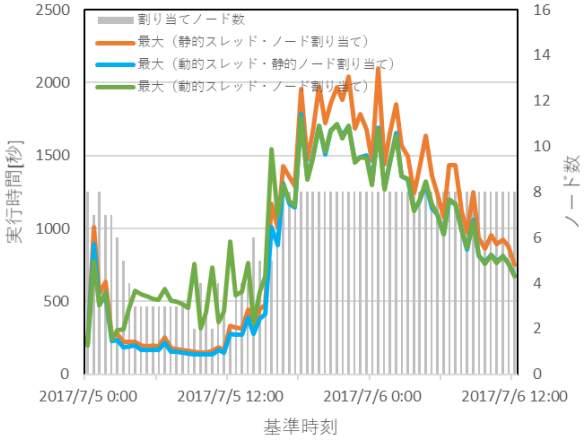

図 8　動的にノードを割り当てた場合の基準時刻による実行時間とノード数の変化

Figure 8　Transition of execution time of epoch time and number of assigned nodes associated with epoch time with dynamic node assignment.

6. おわりに

本稿では，RRI モデルのアンサンブルにおけるアンサンブルメンバ間および入力降水データの違いによる実行時間の偏りについて述べるとともに，計算が終了したアンサンブルメンバに割り当てられていたスレッドを計算が終了していないアンサンブルメンバに割り当て，全体の予測時間を短縮する方式について説明した．また，RRI モデルのアンサンブルを実行するノード数を動的に変更し，効率的に計算機リソースを利用する方式について述べた．処理実験によりその効果を示した．さらに，これらの手法を実装し，実際に RRI モデルのアンサンブルを実行することにより，その効果を示した．

謝辞　本研究は，内閣府「官民研究開発投資拡大プログラム（PRISM）」および文部科学省の委託事業により開発・運用されているデータ統合解析システム（DIAS）の支援により実施された．

参考文献

[1] 気象庁．気候変動監視レポート 2018. https://www.data.jma.go.jp/cpdinfo/monitor/index.html, (参照 2019-10-21).

[2] 中村要介, 小池俊雄, 阿部紫織, 中村和幸, 佐山敬洋, 池内幸司. 粒子フィルタを適用した RRI モデルによる河川水位予測技術の開発, 土木学会論文集 B1（水工学）, 2018, vol.74, no.5, I_1381-I_1386.
[3] 中村要介, 池内幸司, 小池俊雄, 伊藤弘之, 江頭進治, 阿部紫織. 粒子フィルタによる水位と河床変動の逐次推定, 土木学会論文集 B1（水工学）, 2019, vol.75, no.2, I_205-I_210.
[4] Nakamura, Y. et. al. Real-time flood prediction utilizing a particle filter combined with RRI model, EGU General Assembly, 2019, vol.21, EGU2019-12007-1.
[5] 佐山敬洋, 岩見洋一. 降雨流出氾濫（RRI）モデルの開発と応用, 土木技術資料, 2014, vol.56, no.6, pp.18-21.
[6] 山口悟史, 岩村一昭. Dynamic DDM による氾濫シミュレーションの高速化, 情報処理学会論文誌：数理モデル化と応用, 2007, vol.48, no.SIG 6(TOM 17), pp.92-103.4
[7] 吉田圭介, 田中龍二, 前野詩朗. GPU による河川の浅水流計算の高速化, 土木学会論文集 A2（応用力学）, 2014, vol.70, no.2, I_761-I_768.
[8] Nakamura, T. et. al. Efficacy of a GPGPU-Acceleration to Inundation Flow Simulation in Tonle Sap Lake in Cambodia, Engineering Journal, 2018, vol.23, Issue 1, pp.152-169.
[9] 田中涼, 鳥羽奏一朗, 日吉莉菜, 福間愼治, 森眞一郎. 河川流域のオープンデータと連携した実時間志向の簡易流体・拡散連成シミュレーション, 情報処理学会研究報告, 2019, vol.2019-HPC-168, no.9, pp.1-6.
[10] 山口悟史, 楠田尚史. クラウドコンピューティングによる浸水解析の高速化, 土木学会年次学術講演会講演概要集, 2017, vol.72, no.2-001.
[11] “降水短時間予報”. https://www.jma.go.jp/jma/kishou/know/kurashi/kotan_nowcast.html, (参照 2019-10-28).
[12] “解析雨量”. https://www.jma.go.jp/jma/kishou/know/kurashi/kaiseki.html, (参照 2019-10-28).
[13] “RRI model”. http://www.icharm.pwri.go.jp/research/rri/rri_top.html, (参照 2019-10-28).