『全国洪水氾濫被害額推定のための 2 次元氾濫計算』コードの SX-ACE 向け最適化および MPI 並列化

山下 毅 [1)], 田中 裕夏子 [2)], 江川 隆輔 [3,4)], 滝沢 寛之 [3,4)], 風間 聡 [2)], 多田 毅 [5)]

1) 東北大学 情報部情報基盤課
2) 東北大学 大学院工学研究科
3) 東北大学 サイバーサイエンスセンター
4) 東北大学 大学院情報科学研究科
5) 防衛大学校 建設環境工学科

yamacta@tohoku.ac.jp

Performance Optimization and MPI Parallelization of "Nationwide 2D flood simulation for economic damage estimation" code for SX-ACE .

Takeshi Yamashita[1)], Yukako Tanaka[2)], Ryusuke Egawa[3,4)], Hiroyuki Takizawa[3)]
So Kazama[2)], Tsuyoshi Tada[5)]

1) Information Infrastructure Division, Information Department, Tohoku University
2) Graduate School of Engineering, Tohoku University
3) Cyberscience Center, Tohoku University
4) Graduate School of Information Sciences, Tohoku University
5) Department of Civil and Environmental Engineering, National Defense Academy

概要

東北大学サイバーサイエンスセンターでは大規模科学計算システムの活用と効率的な運用を目的とし、利用者プログラムの高速化支援に取り組んでいる。本稿では、研究室のワークステーション上で開発された『全国洪水氾濫被害額推定のための 2 次元氾濫計算』コードの、ベクトル型スーパーコンピュータ SX-ACE 向け最適化手法、および MPI ライブラリを用いた分散メモリ並列化について報告する。スーパーコンピュータを利用することで従来よりも詳細なモデルを高速に解析することが可能となり、被害額の推定結果の妥当性が向上すると同時に、局地的な現象についても検討することが可能となった。

1 はじめに

東北大学サイバーサイエンスセンター（以下、本センター）の大規模科学計算システムは、日本電気株式会社（以下、NEC）製ベクトル型スーパーコンピュータ SX-ACE を主力計算機とし、汎用アプリケーションの実行環境として NEC 製スカラ型の並列コンピュータ LX406-Re2 の運用を行っている。異なる特性を有する二種類の計算機の運用により、利用者の幅広いニーズに応えるサービスの提供を可能としている。表 1 にこれらシステムの諸元を示す。

本センターでは 1999 年より、ユーザアプリケーションの高精度化、大規模化の支援を目的とした共同研究制度を実施している。利用者、計算機科学を専門とするセンター教員と技術職員、および計算機ベンダーが連携してアプリケーションの高速化に取り組んでいる。図 1 に 1999 年から本センターで取り組んでいるセンター独自の共同研究、学際大規模情報基盤共同利用・共同研究拠点（JHPCN）課題および革新的ハイパフォーマンス・コンピューティング・インフラ（HPCI）課題採択数の推移を示す。センターでは共同研究を通じてユーザアプリケーションの高精度化・大規模化をサポートし、JHPCN および HPCI 採択課題へのステップアップを支援している。我々の継続的な高速化支援活動は一定の成果を上げ、またその成果を広く社会に還元している。

表 1　サイバーサイエンスセンター大規模科学計算システム諸元

	性能	SX-ACE	LX406Re-2
CPU 性能	名称	NEC ベクトルプロセッサ	Intel(R) Xeon(R) E5-2695v2
	コア数	4 個	12 個
	理論最大演算性能	276GFLOPS	230GFLOPS
	最大ベクトル演算性能	256GFLOPS	-
	メモリバンド幅	256GB/sec	4.9GB/sec
	ADB / Cache	1MB/コア	30MB (L3)
ノード性能	CPU 数	1 個	2 個
	理論最大演算性能	276GFLOPS	460GFLOPS
	最大ベクトル演算性能	256GFLOPS	-
	メモリ容量	64GB	128GB
	メモリバンド幅	256GB/sec	9.9GB/sec
	ノード間通信速度	8GB/sec	7GB/sec
システム性能	CPU 数	2,560 個	136 個
	理論最大演算性能	706.6TFLOPS	31.3TFLOPS
	最大ベクトル演算性能	655.4TFLOPS	-
	メモリ容量	160TB	8.5TB

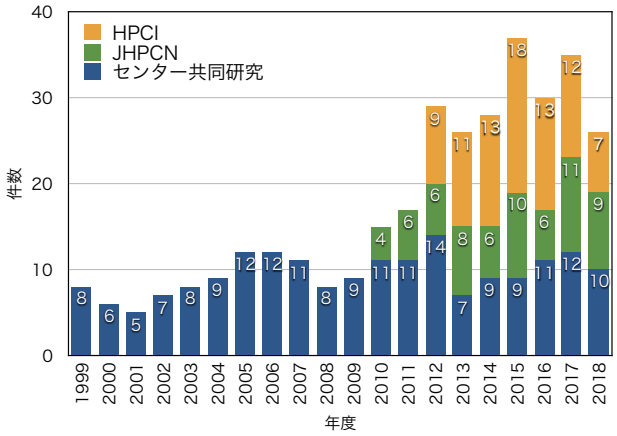

図 1　共同研究課題採択数

以下では高速化支援活動の一例として、東北大学工学研究科土木工学専攻の研究グループが開発中の『全国洪水氾濫被害額推定のための 2 次元氾濫計算』コードの、SX-ACE 向け最適化および MPI 並列化の事例について報告する。

2　アプリケーションの概要

今回 SX-ACE で最適化と MPI ライブラリによる並列化を行ったコード『全国洪水氾濫被害額推定のための 2 次元氾濫計算』は、FORTRAN77 で記述されたコメント行を含む約 490 行からなり、並列化および特有のシステム向けの最適化は施されていない。

2.1　2 次元氾濫計算

本コードは図 2 に示すように、日本全国の洪水氾濫による被害額の分布を定量的に地図の形に示すために、2 次元氾濫計算を行うアプリケーションを表現している。全国を詳細なメッシュに分割し、2 次元氾濫

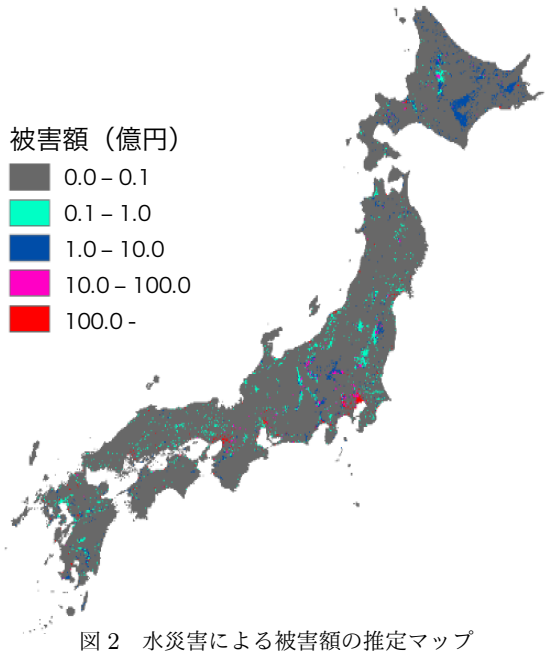

図 2　水災害による被害額の推定マップ

計算を行うことで、各メッシュにおける浸水深が求まる。得られた浸水深と土地利用などの情報を組み合わせることによって、各メッシュでの被害額を推定できる [1]。このアプリケーションにより、将来気候や複合災害下における経済被害額を推定し [2]、優先的に投資するべき地域の抽出や地域に即した治水方法の検討が可能となる。

2.2　コード概要

氾濫流の計算を実施する際、対象領域を IG×JG 個のメッシュに分割し、有限差分法を用いて 2 次元不定流問題を解く。パラメータ IG, JG は解析の空間解像度を規定するパラメータであり、堤防等の微地形の影響を計算結果に反映させるためには、これらのパ

図 3　計算対象領域

ラメータをより大きくとる必要がある。時間進行には 4 段ルンゲクッタ法を採用する。時間刻み DT は IG, JG に依存し、これらの値を大きくとるほど、DT に小さな値を用いる必要がある。以上より、総メッシュ数と時間刻みの両方の影響で、より詳細な結果を得るためにメッシュサイズを小さくすると、その 3 乗のオーダーで計算時間は増加し、より大容量のメモリと計算時間の短縮が必要となる。

本コードのモデルサイズは日本全土を対象とした計算領域 IG=5,032、JG=7,972 であり、250m×250m のメッシュサイズに相当する。ステップ数は観測期間 4 日分に相当する 1,382,400 ステップの演算を行う。計算領域の概略を図 3 に示す。諸島は省略している。

研究室のサーバ（Intel® Core™ i7-6700）では日本全土の計算に約 32 日間を要する。また計算領域を 4 分割（北海道、本州、四国、九州）して演算することが可能で、北海道を対象とした場合は約 12 日間を要する。

本研究においては、初期条件および浸水に関する複数のパラメータを変化させて解析する必要があるため、演算時間の短縮が必要であり、システムへの最適化とマルチノード実行を行うための分散メモリ並列化が要求される。

2.3　初期コードの演算時間調査

SX-ACE および LX406Re-2 で初期コードを実行したときの演算時間を表 2 に示す。SX-ACE 向けには FORTRAN90/SX (Rev.534)、LX 406Re-2 向けには Intel Fortran コンパイラ (17.0.4) をそれぞれ使用した。最適化オプションは使用していない。測定時間短縮のために、対象とする計算領域は北海道とし、観測期間が 1 時間分に相当する 14,400 ステップとした。コンパイラの自動並列化機能を使用し、ノード内での並列実行についても測定を行った。

表 2　初期コードの演算時間比較（対象：北海道）

計算機	並列化	演算コア数	演算時間 [sec]
SX-ACE	なし	1	3,395
SX-ACE	自動並列	4	2,515
LX406Re-2	なし	1	7,984
LX406Re-2	自動並列	24	7,969

1 コアあたりの演算性能は SX-ACE は LX406Re-2 の約 3.6 倍の性能比であるが、演算時間比は約 1/2.4 倍であり、SX-ACE のベクトル最適化が不十分であると推察される。また、コンパイラの自動並列化機能を用いた並列実行では並列コア数に応じた性能向上は得られておらず、MPI ライブラリによる並列化が必要であると判断される。

3　SX-ACE 向け最適化と性能比較

3.1　簡易性能解析機能によるコスト調査

SX-ACE では、コード実行時に性能解析情報の取得を行う簡易性能解析（FTRACE）機能を利用した演算時間のコスト分析が可能である。FTRACE 機能を利用するにはソースコードのコンパイル時とリンク時に-ftrace を指定する。初期コードの FTRACE 情報を表 3 に示す。

主要な演算を行うサブルーチン flood100 の演算割合はほぼ 100% であるが、このサブルーチンのベクトル化率（V.OP RATIO）が 97.83 と低く、改善が必要である事が分かる。

3.2　ベクトル化の促進による高速化

FORTRAN90/SX ではコンパイラが行ったベクトル化などの最適化に関する情報をソースコードの左側に付記し、編集リスト（ソースコード名.L）として出力する機能がある。この機能を利用して詳細なベクトル化診断メッセージを出力するために、コンパイルオプションとして-R5 -Wf,-pvctl fullmsg を指定した。図 4 に初期コードの編集リストの一部を示す。なお、282 行目の IF 文は対象とする計算領域を選択するもので、T_ISL には北海道の領域が指定されている。

278 行目のインデックス I を持つ DO ループにベクトル化された記号 “V” が付加されているが、280 行目から 294 行目にかけてベクトル処理が不可であること

（編集リストの抜粋）

```
  277: |V----->            DO J=J1,J2
  278: ||V---->              DO I=I1(J),I2(J)
  279: |||
  280: |||     AS               IF(X(I,J).LT. 0.5 .OR. X(I+1,J) .LT. 0.5
  281: |||      S       &                     .OR. X(I,J+1) .LT. 0.5) THEN
  282: |||      S                 IF(Isl(I,J).EQ.T_ISL .OR. Isl(I+1,J) .EQ. T_ISL (計算領域を限定：北海道)
  283: |||      S       &                     .OR. Isl(I,J+1) .EQ. T_ISL ) THEN
  284: |||      S                   OUTMN=0.0
  285: |||      S                   IF(M(I-1,J) .LT. 0.0) OUTMN=OUTMN-M(I-1,J)
  286: |||      S                   IF(M(I,J) .GT. 0.0) OUTMN=OUTMN+M(I,J)
  287: |||      S                   IF(N(I,J-1) .LT. 0.0) OUTMN=OUTMN-N(I,J-1)
  288: |||      S                   IF(N(I,J) .GT. 0.0) OUTMN=OUTMN+N(I,J)
  289: |||      S                   OUTD=OUTMN*DT/DX/GV(I,J)
  290: |||      S                   IF(OUTD .GT. D(I,J)) THEN
  291: |||      S                     IF(M(I-1,J) .LT. 0.0) M(I-1,J)=M(I-1,J)*D(I,J)/OUTD
  292: |||      S                     IF(M(I,J) .GT. 0.0) M(I,J)=M(I,J)*D(I,J)/OUTD
  293: |||      S                     IF(N(I,J-1) .LT. 0.0) N(I,J-1)=N(I,J-1)*D(I,J)/OUTD
  294: |||      S                     IF(N(I,J) .GT. 0.0) N(I,J)=N(I,J)*D(I,J)/OUTD
  295: |||                          ENDIF
  296: |||                        ENDIF
  297: |||                      ENDIF
  298: ||V----                ENDDO
  299: |V-----              ENDDO

（コンパイル時最適化メッセージの抜粋）
  277  vec  (   2): ループの一部をベクトル化する。
  278  vec  (   2): ループの一部をベクトル化する。
  285  opt  (1037): 異なる繰り返しで同一の配列要素を定義／参照している。
  287  opt  (1036): 異なる繰り返しで定義された値を参照している可能性がある。(nosync/nodepを指定すれば最適化を行う)
  291  vec  (  20): mにベクトル化不可の依存関係がある。
  292  vec  (  20): mにベクトル化不可の依存関係がある。
```

図 4　初期コード

（編集リストの抜粋）

```
   60:                   REAL*8 OUTD_(IG,JG)
（省略）
  322: |V----->            DO J=J1,J2
  323: ||V---->              DO I=I1(J),I2(J)
  324: |||     A                IF(X(I,J).LT. 0.5 .OR. X(I+1,J) .LT. 0.5
  325: |||              &                     .OR. X(I,J+1) .LT. 0.5) THEN
  326: |||     A                  IF(Isl(I,J).EQ.T_ISL .OR. Isl(I+1,J) .EQ. T_ISL (計算領域を限定：北海道)
  327: |||              &                .OR. Isl(I,J+1) .EQ. T_ISL ) THEN
  328: |||
  329: |||                           OUTMN=0.0
  330: |||     A                     IF(M(I-1,J  ) .LT. 0.0) OUTMN=OUTMN-M(I-1,J  )
  331: |||     A                     IF(M(I  ,J  ) .GT. 0.0) OUTMN=OUTMN+M(I  ,J  )
  332: |||     A                     IF(N(I  ,J-1) .LT. 0.0) OUTMN=OUTMN-N(I  ,J-1)
  333: |||     A                     IF(N(I  ,J  ) .GT. 0.0) OUTMN=OUTMN+N(I  ,J  )
  334: |||
  335: |||     A                     OUTD_(I,J)=OUTMN*DT/DX/GV(I,J)
  336: |||
  337: |||                         ENDIF
  338: |||                       ENDIF
  339: ||V----                 ENDDO
  340: |V-----              ENDDO
  344: |V----->            DO J=J1,J2
  345: ||V---->              DO I=I1(J),I2(J)
  346: |||     A                IF(X(I,J).LT. 0.5 .OR. X(I+1,J) .LT. 0.5
  347: |||              &                     .OR. X(I,J+1) .LT. 0.5) THEN
  348: |||     A                  IF(Isl(I,J).EQ.T_ISL .OR. Isl(I+1,J) .EQ. T_ISL (計算領域を限定：北海道)
  349: |||              &                .OR. Isl(I,J+1) .EQ. T_ISL ) THEN
  350: |||
  351: |||     A                   IF(OUTD_(I,J) .GT. D(I,J)) THEN
  352: |||                           OUTD=D(I,J)/OUTD_(I,J)
  353: |||     A                     IF(M(I  ,J  ) .GT. 0.0) M(I  ,J  )=M(I  ,J  )*OUTD
  354: |||     A                     IF(M(I-1,J  ) .LT. 0.0) M(I-1,J  )=M(I-1,J  )*OUTD
  355: |||     A                     IF(N(I  ,J-1) .LT. 0.0) N(I  ,J-1)=N(I  ,J-1)*OUTD
  356: |||     A                     IF(N(I  ,J  ) .GT. 0.0) N(I  ,J  )=N(I  ,J  )*OUTD
  357: |||                         ENDIF
  358: |||
  359: |||                       ENDIF
  360: |||                     ENDIF
  361: ||V----                ENDDO
  362: |V-----              ENDDO

（コンパイル時最適化メッセージの抜粋）
  322  vec  (   1): ループ全体をベクトル化する。
  323  vec  (   1): ループ全体をベクトル化する。
  344  vec  (   1): ループ全体をベクトル化する。
  345  vec  (   1): ループ全体をベクトル化する。
```

図 5　ベクトル化後コード

表 3　初期コードの FTRACE 情報

【初期コード】

PROC.NAME	FREQUENCY	EXCLUSIVE TIME[sec](%)	AVER.TIME [msec]	MOPS	MFLOPS	V.OP RATIO	AVER. V.LEN	VECTOR TIME	I-CACHE MISS	O-CACHE MISS	BANK CONFLICT CPU PORT	BANK CONFLICT NETWORK	ADB HIT ELEM.%
flood100	1	3395.271(100.0)	3395270.83	14618.1	6206.6	97.83	198.2	1665.816	0.022	163.245	27.664	229.810	53.30
init	1	0.134(0.0)	134.439	29316.9	3580.7	99.75	255.9	0.134	0.000	0.000	0.004	0.004	0.01
output	3	0.000(0.0)	0.064	160.2	0.0	1.92	74.0	0.000	0.000	0.000	0.000	0.000	0.00
total	5	3395.405(100.0)	679081.09	14618.7	6206.5	97.83	198.2	1665.950	0.023	163.245	27.668	229.814	53.29

表 4　ベクトル化後コードの FTRACE 情報

【ベクトル化後コード】

PROC.NAME	FREQUENCY	EXCLUSIVE TIME[sec](%)	AVER.TIME [msec]	MOPS	MFLOPS	V.OP RATIO	AVER. V.LEN	VECTOR TIME	I-CACHE MISS	O-CACHE MISS	BANK CONFLICT CPU PORT	BANK CONFLICT NETWORK	ADB HIT ELEM.%
flood100	1	1850.736(100.0)	1850736.14	27222.3	11498.7	99.53	198.6	1706.915	0.013	0.211	26.722	243.202	52.91
init	1	0.135(0.0)	134.952	29205.6	3567.1	99.75	255.9	0.135	0.000	0.000	0.006	0.005	0.01
output	3	0.000(0.0)	0.065	158.2	0.0	1.92	74.0	0.000	0.000	0.000	0.000	0.000	0.00
total	5	1850.871(100.0)	370174.25	27222.5	11498.2	99.53	198.6	1707.049	0.014	0.211	26.729	243.207	52.91

表 5　IF 文消去後コードの FTRACE 情報

【IF文消去後コード】

PROC.NAME	FREQUENCY	EXCLUSIVE TIME[sec](%)	AVER.TIME [msec]	MOPS	MFLOPS	V.OP RATIO	AVER. V.LEN	VECTOR TIME	I-CACHE MISS	O-CACHE MISS	BANK CONFLICT CPU PORT	BANK CONFLICT NETWORK	ADB HIT ELEM.%
flood100	1	9946.668(100.0)	9946668.215	27367.2	12425.3	99.60	202.2	9232.460	0.225	0.547	327.191	932.144	55.50
init	1	0.273(0.0)	273.275	29229.4	1762.1	99.44	255.9	0.273	0.000	0.000	0.031	0.014	0.00
output	3	0.116(0.0)	38.760	26325.1	1.0	98.29	256.0	0.116	0.000	0.000	0.006	0.019	0.00
total	5	9947.058(100.0)	1989411.554	27367.2	12424.9	99.60	202.2	9232.849	0.225	0.547	327.227	932.178	55.50

表 6　性能比較（初期コード、ベクトル化の対象：北海道　IF 文消去後コードの対象：日本全土）

	演算時間 [sec]	演算性能 [GFlops]	実行効率 [%]	平均ベクトル化率 [%]	平均ベクトル長
初期コード	3,395	6.21	9.0	97.83	198.2
ベクトル化後コード	1,850	11.50	16.7	99.53	198.6
IF 文消去後コード	9,947	12.42	18.0	99.60	202.2

を示す記号 “S” が付加されている。最適化メッセージから 291 行目および 292 行目の処理に依存関係があるため、DO ループ全体のベクトル化が不可となっていることが分かる。

該当箇所のベクトル化を促進するために、1. 依存関係のある演算順序の変更、2. ループ分割と作業配列の導入、を行った。図 5 にベクトル化後コードの編集リストの一部を示す。

依存関係によりベクトル化が不可であった、291 行目の演算を後に、292 行目の演算を前に行うように演算順序を変更（それぞれ 354 行目と 353 行目に）した。次に、ループ分割を行うことで DO ループ中の IF 文による条件分岐の個数を減らし、コンパイラがベクトル化の判断を行うことが可能となった。ループ分割を行うために後半の DO ループで必要な演算結果を前半の DO ループで保存するための作業配列 OUTD_を導入した。

3.3　IF 文の消去による高速化

SX-ACE では DO ループ中の IF 文については、真偽に基づいてベクトルマスクを作成し、演算については本来の IF 文の真偽にかかわらず演算を行う。DO ループの演算終了後に IF 文を考慮したベクトルマスクにより、IF が真となる演算結果のみをストアすることでベクトル化を可能としている。このことから、図 5 において DO ループ中で対象とする計算領域を指定する IF 文（326 行目および 348 行目）を消去する、すなわち日本全土を対象として演算することで実効性能の改善が期待できる。

3.4　性能比較

ベクトル化後コードの FTRACE 情報を表 4 に、対象とする計算領域を指定する IF 文を消去し、日本全土について演算を行ったときの FTRACE 情報を表 5 に示す。また、初期コード、ベクトル化後コードおよび IF 文消去後のコードの性能比較を表 6 に示す。

初期コードでは平均ベクトル化率が 97.83% であっ

たが、ループ分割と演算順序の変更によるベクトル化により 99.53% に向上した。また、演算領域を限定する IF 文を消去したことで、対象範囲が北海道から日本全土になったために演算時間は増加しているが、実行効率は 16.7% から 18.0% に向上したことから、日本全土をシミュレーションする場合にはＩＦ文消去後のコードの方が高速であることが分かる。初期コードに対する性能向上比は演算性能値の比較より、それぞれ 1.8 倍、2.0 倍を達成することができた。

4　MPI ライブラリによる並列化

4.1　並列化方針

次に、ベクトル化および IF 文を消去した最適化コードの MPI ライブラリによる並列化を行った。分割方向は図 3 に示した J 方向について 1 次元分割を行い、各プロセスで分割された領域について演算を行った後に、袖領域のデータ通信を行う。データ通信には MPI_SENDRECV を使用した。演算結果は代表プロセスに各プロセスから MPI_GATHERV を用いて受信し、バイナリ形式のファイルとして書き出しを行う。

4.2　並列化コードの演算時間比較

MPI ライブラリによる並列化を行ったコードを実行した結果を表 7 に示す。

SX-ACE においては、128 コアを用いた並列実行は逐次実行と比較して演算時間比で約 78 倍の性能向上が得られた。並列コア数に対する並列性能は理想値の 68% 程度であるので、通信部分の最適化などによりさらなる性能向上を試みる予定である。

また LX406Re-2 において 120 コアを用いた並列実行は逐次実行と比較して、約 73 倍の性能向上がみられた。SX-ACE（128 コア）と LX406Re-2（120 コア）の理論演算性能比は約 3.8 倍であるが、演算時間比では約 5.8 倍と、SX-ACE の実効性能の高さが示された。

演算時間計測は観測期間が 1 時間分に相当する 14,400 ステップで行ったので、観測期間 4 日分に相当する 1,382,400 ステップの演算を行う場合の演算時間は、SX-ACE（128 コア）においては約 3.4 時間と推定される。

表 7　並列化コードの演算時間比較（対象：日本全土）

計算機	並列化	演算コア数	演算時間 [sec]
SX-ACE	なし	1	9,946
SX-ACE	Flat MPI	128(32 ノード)	128
LX406Re-2	なし	1	53,901
LX406Re-2	Flat MPI	120(6 ノード)	739

5　まとめ

本稿では『全国洪水氾濫被害額推定のための 2 次元氾濫計算』コードの SX-ACE 向け最適化および MPI 並列化の事例について述べた。

今回実施したベクトル化および IF 文を消去する最適化により、SX-ACE において約 9% であった実行効率を約 18% まで高速化することができた。また、MPI ライブラリによる並列化により 128 コアを用いた並列実行では、逐次実行の約 78 倍の性能向上が得られた。

従来の研究におけるグリッドセルサイズは約 1km×1km であったが、この解像度では、河川ごとに異なる治水施設の整備状況などを十分に反映することができなかった。今回、約 250m×250m の解像度が実用的な時間で実行できるようになったことで、被害額の推定結果の妥当性が向上すると同時に、局地的な現象についても検討することが可能となった。

計算速度の向上により様々な条件での計算を多数実行することが容易となったことから、今後は降水量の増加や海面上昇など地球温暖化による影響を考慮した解析 [3]、洪水に加え高潮や斜面崩壊、沿岸浸食などによる複合災害のリスクの検討 [4] を高解像度で実施する予定である。

今後は MPI 通信の最適化などについて高速化を行い、利用者の研究活動の加速を支援する予定である。

サイバーサイエンスセンターの大規模科学計算システムが、利用者の研究発展の一助になるために、今後も高速化推進研究活動に取り組む所存である。

参考文献

[1] 国土交通省河川局：治水経済調査マニュアル（案），2005.

[2] 手塚翔也，小野桂介，風間聡，小森大輔：極値降雨，流出量に基づく洪水被害推定およびその将来変化，土木学会論文集，Vol.70，No.4，pp.1501-1506，2014.

[3] 町田宗一郎，川越清樹，風間聡，沢本正樹，横木裕宗，安原一哉，地球温暖化に伴う全国の浸水被害額評価，地球環境シンポジウム講演論文集，vol.15，pp.155-160，2007.

[4] 秋間将宏，風間聡，峠嘉哉，小森大輔，川越清樹，多田毅，年最小気圧を用いた複合水災害潜在被害額の将来推定，土木学会論文集 B1（水工学），Vol.73，No.4，I_139-I_144，2017.