

技術報 ビッグデータをインプットとした演算処理高速化の研究

HPC を用いた演算処理高速化研究グループ

最新 GPU(演算器)を用いたプログラムの高速化と、1 システムを複数環境で実行可能な手法の実装化及び演算アルゴリズムの最適化について、東北大学と共同研究を行った。HPC を活用して、実運用を見据えた課題整理とその解決策に取り組むことで、当社独自のノウハウを確立し、将来的な業務受注や海外展開に繋げることを本研究の目的とする。本稿では、これらの取り組みの成果を報告する。

キーワード：HPC、複数環境、最新 GPU(演算器)、高速化、演算アルゴリズム

1. はじめに

建設コンサルタント業界におけるビッグデータの新たな利用価値を生み出すため、HPC（ハイ・パフォーマンス・コンピューティング）の高速化技術（ベクトル化技術、並列化等）を、東北大学滝沢研究室との共同研究により進めてきた。

高性能計算（HPC: High Performance Computing）技術を活用し、従来の計算資源に依存しない柔軟な運用システムへの転換を図るため、東北大学サイバーサイエンスセンターとの共同研究により、演算アルゴリズムの最適化、1 システムを複数環境で実行可能な手法の実装化及び最新 GPU(演算器)を用いたプログラムの高速化を行った。

2. 研究の背景

当社ではこれまで、RRI を用いた洪水予測システムの高速化を目的として、NEC 製のベクトル型スーパーコンピュータ「SX-Aurora TSUBASA」を活用してきた。しかし、同クラウドサービスの提供終了が発表されたこと、ならびにコスト面の課題（1 ベクトルエンジンあたり月額 66 万円）を受けて、東北大学サイバーサイエンスセンターの HPC 基盤「AOBA」への移行に着手した。AOBA では 20 円/時間程度の利用料金で高性能な演算資源を確保できるため、コスト面での大幅な削減が可能となった。AOBA への移行により、汎用的かつ低コストでの運用形態への転換が期待されている。

一方で、AOBA 特有の制約もあり、運用面での新たな課題が明らかになり、その解決に向けての取り組みを進めている。また、AOBA 移行後もさらなる冗長性の向上が必要となるため、Amazon EC2 をはじめとしたクラウドサービス上での計算実行も行ったが、演算速度の面で目標とする「10 分以内のリアルタイム洪水予測」には届かなかった。こうした経緯を踏まえ、現在では GPU を用いた演算高速化や、異なる実行環境間でのシステム共通化（ワークフローエンジンや PSI/J の導入）も視野に入

れて模索を進めている。



図-1 本研究における計算基盤の変遷と検証対象の位置づけ

次章では、SX-Aurora TSUBASA から AOBA への移行時に生じた具体的な課題とその解決策について詳述する。

3. 課題と解決策

1) 最新 GPU(演算器)を用いたプログラムの高速化

昨年度に実施したベクトル化、並列化の高速化コードをベースに、ディレクティブ（指示文）のみ付ける形で GPU 化可能な OpenACC を適用した。

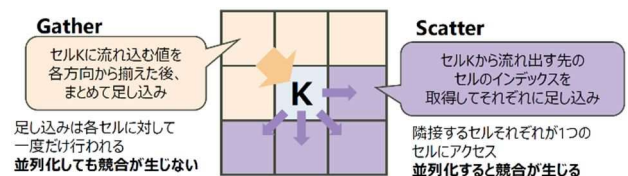


図-2 セル間における水のやり取り

RRI で斜面の2次元計算をする際に、メモリアクセス時の Gather（集約）/Scatter（分散）が繰り返されるが、メモリアクセスが細切れに発生するため、連続したアドレスに一斉にアクセスしようとしても最大の帯域幅（スループット）を確保できないのが、GPU 高速化を阻む最大原因である。

そのため、メモリを連続で確保することができるよう、メモリ管理の仕組みを図3,4のように新たに追加した。

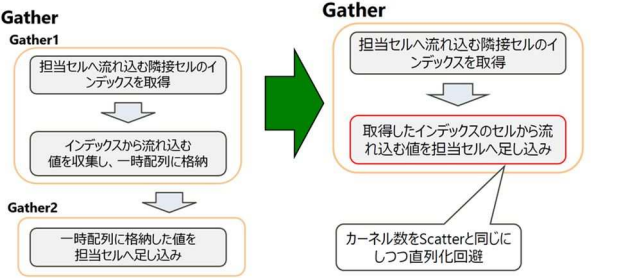


図-3 メモリアクセス時の Gather (集約) の改良前後

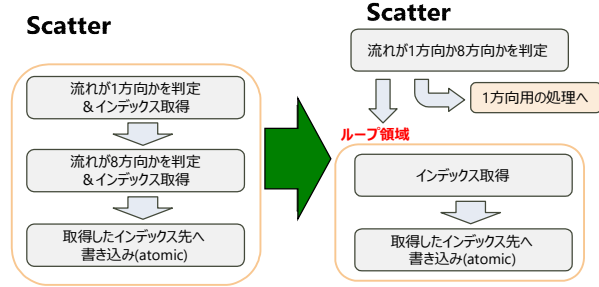


図-4 メモリアクセス時の Scatter (分散) の改良前後

表-1 GPU 改良前後の計算結果の比較

GPU (A100 80GB)	AOBA-S (type30A)
743. 616s	2149. 580s

表-1 で示すように、100mm/h 降雨時、全国版 RRI 東北モデルの 6 時間先まで予測計算において、SX-Aurora TSUBASA (VE) の AOBA-S (type30A) よりも、GPU (A100 80GB) のほうが、約 3 倍速くなった。

また、図-5 で示すように、RRI の各サブルーチンの中で、funcs 及び funcsr 部分が GPU 高速化に効いている。

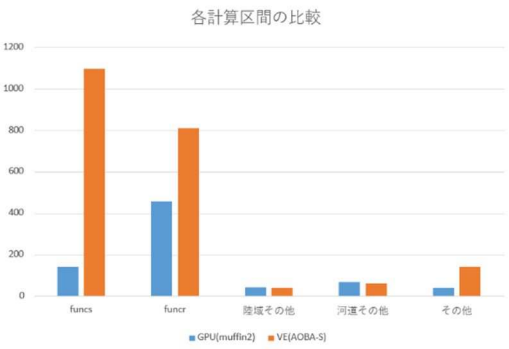


図-5 GPU 化による RRI 各サブルーチンの計算時間比較

高速化計算環境として、SX-Aurora TSUBASA (VE) 以外にクラウド環境の GPU 活用の選択肢が増えた。

ベクトルマシン環境だけでなく、AWS や Azure 等の GPU クラウド活用の可能性が広がった。RRIver2 における GPU の活用など、他社との差別化に優位であるが、GPU は高額という側面もある。

2) マルチプラットフォームで同じシステムが実行可能な手法の実装

課題①: マルチプラットフォームへの冗長化

AOBA (東北大学サイバーサイエンスセンター) の HPC を商用利用する方針であるが、年間 2 回の定期保守のための計画停止が予定されている。

計画停止期間中に別の環境に処理を切り替えて継続実行する仕組みを開発することで、マルチプラットフォームへの冗長化が実現できる。

それにより計画停止以外の理由で AOBA が停止した際にも、AWS 環境で処理を継続することができるため、システムの信頼性が向上する。

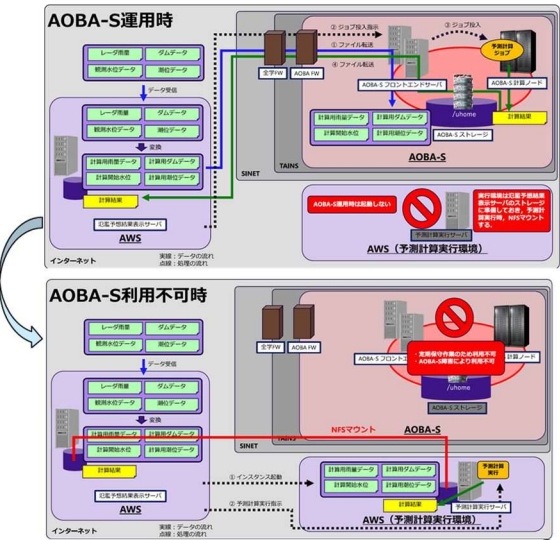


図-6 AOBA-S 運用時、利用不可時の作業フロー①

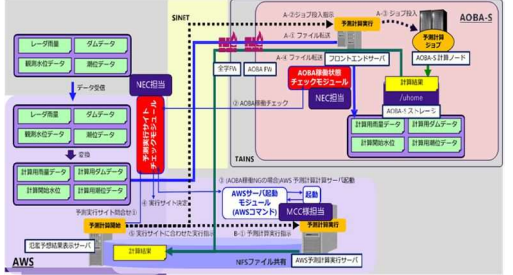


図-7 AOBA-S 運用時、利用不可時の作業フロー②

図-7 で示すようなチェックモジュールの開発により、双方向に冗長化が可能となった。チェックモジュールは、図-8 のような処理フローである。

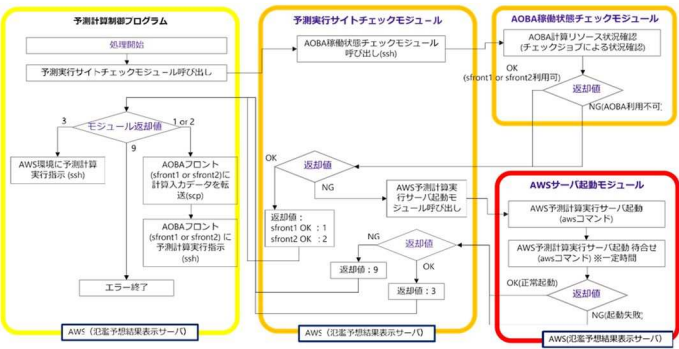


図-8 マルチプラットフォームの自動切り替えの処理フロー

The diagram illustrates the process of receiving a job completion notification from AWS to AOBA-S. It consists of two main panels: 'AWS (仮番予想結果表示サーバ)' on the left and 'AOBA-S' on the right. The workflow is as follows:

- ① AOBA-Sへ ssh によるジョブ投入指示 (Job input instruction to AOBA-S via ssh)
- ② AOBA-Sへのジョブ投入 (ジョブ終了後にジョブ終了フラグを作成) (Job input to AOBA-S (Create job completion flag after job completion))
- ③ ジョブ終了フラグチェックスクリプトをバックグラウンドで実行 (Execute job completion flag check script in the background)
- ④ AWS上でジョブ終了フラグの存在を定期的にチェック (Check for the existence of the job completion flag on AWS periodically)
- ⑤ フロントエンド上でジョブ終了フラグの存在を定期的にチェックし、存在を確認したらAWSへscpによりコピー (ジョブ終了フラグチェックスクリプト内でのバックグラウンド処理) (Check for the existence of the job completion flag on the front end periodically, and if confirmed, copy to AWS by scp (Background processing in the job completion flag check script))
- ⑥ AWS上でジョブ終了フラグの存在を確認したら、次の処理へ進む (If the existence of the job completion flag is confirmed on AWS, proceed to the next process)

表-2 で示すように、AOBA-S でのジョブ終了を判定するための方法として、案 1,2 が考えられたが、それぞれ問題が存在した。

No.	内容	課題
1	qwait コマンド (ジョブの終了を待ち合わせるコマンド) を利用する.	ジョブが完了するまで AWS と AOBA-S の間のセッションが張り続けられるため, 一時的なネットワーク断などの影響を受けてしまう
2	AOBA-S でのジョブ終了時にジョブ終了フラグを作成し, AWS から定期的にフラグの存在確認を行う.	AWS から定期的に AOBA-S 上のフラグの存在確認を行うため, AWS-AOBA 間の ssh による通信が多発する.

図-12 CWL を利用したワークフロー制御

3) 演算アルゴリズムの最適化

ビッグデータをインプットとした洪水予測等のリアルタイム性が必要な演算処理について、演算アルゴリズムの最適化による高速化が必須である。

本研究では、HPC のベクトル化技術、並列化技術を用いて分析、予測処理を高速化するためのプログラム開発手法について研究を行う。

1) ベクトル化

ベクトル命令は、同じ演算を行う演算器を複数持っており、複数データに対して同時に処理(読み込み、演算)を実行するため、1 度に多くの演算を実行することが可能となる。

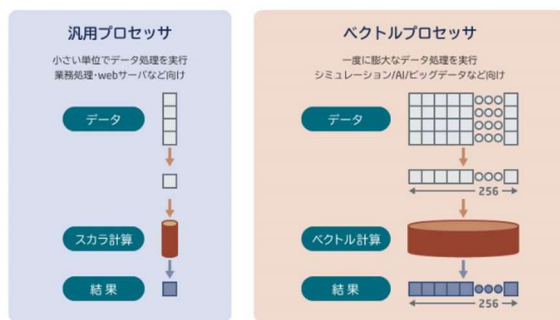


図-13 ベクトル化のイメージ

2) 並列化

公開されている RRI モデルは OpenMP による並列化は実装されているが、MPI による分散並列化は実装されていない。そのため、最大並列数は共有メモリノード内のコア数により制限される。そこで、RRI モデルの MPI 化を検討した。

ファイル I/O はランク 0 が代表して行っており、実行時間全体に占める I/O 時間の割合は高くなる。Hybrid MPI でファイル I/O の最適化を行った。

Hybrid MPI の `mpi_comm_split` 関数を使用して全プロセスが参加するコミュニケーター (`mpi_comm_world`) を、演算処理を行うコミュニケーター (`mpi_comm_rri`) と、I/O 処理を行うコミュニケーター (`mpi_comm_io`) に分割した。

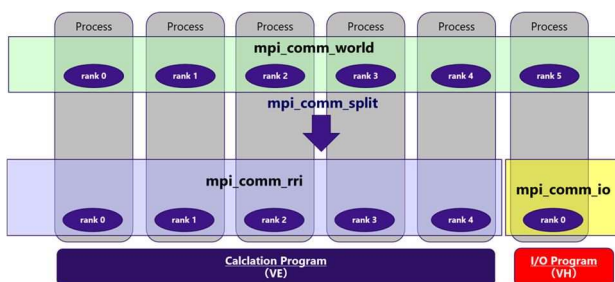


図-14 Hybrid MPI によるコミュニケーターの分割

○ 初期性能

実行時間	MFLOPS値	ベクトル演算率	平均ベクトル長
2,289.4	1,170.6	15.4	71.9

○ 最適化後

実行時間	MFLOPS値	ベクトル演算率	平均ベクトル長
796.3	8,705.0	75.0	167.8

約 2.9 倍

図-15 プログラムの高速化指標

ベクトル化の改善(平均ベクトル長、ベクトル演算率)に加えて、MPI 化(ランクの均一、ファイル I/O)により、東北モデルにおける実況 1 時間、予測 6 時間の計算において、4 ノードを使用すれば、計算時間が約 20 分まで短縮され、実用レベルまで達することを確認した。

4. まとめ

これまでの共同研究により、スパコンを用いた処理の高速化は実現することができたが、スパコンの処理結果を公開用 WEB サーバのクラウド環境に受け渡すデータ同期の高速化やエラー処理対策等のシステム全体としての最適化、高速化、安定化が課題として残っている。

公開が予定されている RRIver2 について他社に先駆けて高速化を実現する。

高速化、システム最適化の成果を踏まえ、AOBA (東北大学サイバーサイエンスセンター) の HPC を活用して業務システムの実装を進める。

当社が RRI モデルを構築した兵庫県、京都府、鳥取県向けのクラウドサービスを NEC クラウドから AOBA 移植するための実装を行う予定である。

参考文献

- 1) OpenACC : <https://www.openacc.org/>
- 2) RRI (Rainfall-Runoff-Inundation) : https://www.pwri.go.jp/icharm/research/rri/index_j.html.
- 3) GPU : <https://ja.wikipedia.org/wiki/GPU>
- 4) SX-Aurora TSUBASA : <https://jpn.nec.com/hpc/sxauroratsubasa/index.html>
- 5) AOBA: <https://www.cc.tohoku.ac.jp/service/supercomputer/>

謝辞: 本研究に際し、東北大学 滝沢 寛之教授、下村 陽一特任研究員、撫佐 昭裕客員教授にご支援及び協力を得た。