TOHOKU
UNIVERSITY

ISSN 0286-7419

東 北 大 学
サイバーサイエンスセンター

大規模科学計算システム広報

SENAC

Vol.45 No.1 2012−1

Supercomputing System
Cyberscience Center
Tohoku University
www.ss.isc.tohoku.ac.jp

大規模科学計算システム関連案内

＜大規模科学計算システム関連業務は、サイバーサイエンスセンター本館内の情報部情報基盤課が担当しています。＞

http://www.ss.isc.tohoku.ac.jp/

階	係・室名	電話番号(内線) e-mail	主なサービス内容	サービス時間 月～金
一階	共同利用支援係 （受　付）	022-795-3406(3406) FAX:022-795-6099 uketuke@isc.tohoku.ac.jp	各種申請書、講習会、利用相談、広報、センターへの質問や要望の受付等	8:30～17:15
	利用相談室	022-795-6153(6153) sodan05@isc.tohoku.ac.jp 相談員不在時 022-795-3406(3406)	計算機利用全般に関する相談 大判プリンタ、利用者端末等の利用	8:30～17:15 8:30～21:00
	利用者談話室	(3444)	各センター広報の閲覧	8:30～21:00
	展示室（分散コンピュータ博物館）		歴代の大型計算機等の展示	9:00～17:00
三階	庶務係	022-795-3407(3407) syomu@isc.tohoku.ac.jp	庶務に関すること	8:30～17:15
	会計係	022-795-3405(3405) kaikei@isc.tohoku.ac.jp	会計に関すること、負担金の請求に関すること	8:30～17:15
	共同研究支援係	022-795-6252(6252) rs-sec@isc.tohoku.ac.jp	共同研究、計算機システムに関すること	8:30～17:15
	共同利用支援係	022-795-6251(6251) uketuke@isc.tohoku.ac.jp	ライブラリ、アプリケーションに関すること	8:30～17:15
	ネットワーク係	022-795-6253(6253) net-sec@isc.tohoku.ac.jp	ネットワークに関すること	8:30～17:15
四階	研究開発部	022-795-6095(6095)		
五階	端末機室	(3445)	PC端末機(X端末)	8:30～21:00

(　)内は東北大学内のみの内線電話番号です。青葉山・川内地区以外からは頭に92を加えます。

本誌の名前「SENAC」の由来

昭和33年に東北地区の最初の電子計算機として、東北大学電気通信研究所において完成されたパラメトロン式計算機の名前でSENAC-1(SENdai Automatic Computer-1)からとって命名された。

［巻頭言］

次世代津波警報システムの開発

理学研究科 地震・噴火予知研究観測センター
長谷川　昭

昨年3月11日に発生した東北地方太平洋沖地震は、我が国観測史上最大のマグニチュード（M）9という超巨大地震で、東北地方太平洋岸を中心に各地に甚大な被害を及ぼした。犠牲になった方々の数は2011年12月22日現在で19,312名である。その殆どは津波によるもので、1923年関東地震、1896年明治三陸地震津波に次ぐ、明治以降で3番目に大きな被害となった。被害の規模は想像を絶するものであり、地震学者の一人として、このような事態に至ったことは誠に無念で、言いようもない無力感に打ちのめされている。

今回の地震により、津波による犠牲者の数をどのようにしたら減らすことができるかが、大きな課題として浮かび上がった。地震後の種々の調査で明らかになったことの一つに、海岸近くに住んでいた人達で、あれほど大きな揺れであっても、地震後すぐに避難しなかった人の割合が多かったことがあげられる。津波常襲地帯であり、1896年明治三陸地震、1933年昭和三陸地震、1960年チリ地震と過去100年ほどの間に3度も大きな津波被害を受けた三陸沿岸の地域であってさえも、そうであった。避難が迅速に行われていればこれほどまでにならなかったのにと、悔やまれてならない。

その原因としては、1)ここまで津波は来ないという思い込み、2)防潮堤への過信、3)津波警報の予測波高が低かったこと、4)地震調査研究推進本部の長期予測でM9を想定していなかったこと、5)中央防災会議でも防災対策の検討対象としていなかったことなど、いろいろ考えられる。おそらく原因は一つだけでなく、複合的なものであろう。今後は、系統的な調査により原因の一つ一つを特定した上で、それら全てを取り除いていくための対策をとることが求められる。2度とこのようなことが起こらないように。

上記のように、考えられる全ての対策をとる必要があるが、その中でとりわけ重要な対策の一つに、津波警報の格段の高精度化がある。現在の津波警報は、陸上の地震計網で得られた地震波データのみを用いて予測を行い、警報として発信している。これでは、精度に自ずと限界がある。そのことが、避難する人の割合を減らす原因の一つになっていることを如実に示したのが、2007年1月13日に千島列島で発生したM8.2の地震である。地震後、北海道の太平洋沿岸東部とオホーツク海沿岸に津波警報が発令されたが、かなり多くの人々が避難しなかった。実はその2か月前の2006年11月15日に同じく千島列島でM8.1の地震が発生し、その際にも津波警報が出されたが大きな津波は来なかった。このことがあって、2007年の地震の際には避難した人の割合が激減した。発信された情報が信頼され

なかったのである。幸い、このときも大きな津波にならなかったので良かったが。

津波警報が出ても避難する人の割合が極めて少ないという現象は、2003 年 5 月 26 日宮城県沖地震の際にも既にみられていた。このことは、次に本当に大きな津波が来襲したら、とんでもなく大きな被害になってしまうであろうことを予感させていた。地震調査研究推進本部や中央防災会議、宮城県の委員会などで度々指摘してきたが、有効な対策がとれないうちに今回の地震になってしまったことは誠に残念である。

では、どうすれば津波警報の予測精度を格段に上げることができるのであろうか？その答えは、1994 年 12 月 28 日三陸はるか沖地震後に岩手県釜石沖に設置されたケーブル式地震・津波計のデータが既に示している。すなわち、陸上の地震計のデータに頼るだけでなく、津波波源に近い沖合の津波計のデータを使って予測し、まだ津波が到達していない沿岸各地にその情報を伝えれば良い。これは、震源に近い地点の地震計のデータを使って予測し、地震波のまだ到来していない地点にその情報を早期に伝える「緊急地震速報」と同じ原理である。事実、釜石沖のケーブル式地震・津波計は、今回の地震でも、震源から少し離れていたにも拘らず、地震発生から 7 分後には約２ｍ、14 分後には５ｍの高さの津波を観測している。津波は、沖合の水深の深い場所ほど速く伝わるからである。従って、沖合に津波計の観測網があれば、各地点に来襲する津波の到達時刻・波高などの情報を、いち早くかつ高精度に予測することが可能となる。

今回の地震後、地震調査研究推進本部では、北海道の根室沖から関東の房総沖に至る海域に、ケーブル式の稠密地震・津波計網を設置する計画を立てた。幸い、今年度補正予算と来年度予算で経費の一部が認められた。念願がやっとかなったが、これが東北沖地震の前であったらと思うと残念でならない。とは言え、津波常襲地帯の三陸沿岸を始め、海岸地域の今後の津波対策を考える上で、朗報である。格段に予測精度の高い「次世代津波警報システム」の開発に向けて、大きな一歩を踏み出すことができたからである。これからは、陸上の観測網のデータだけでなく、沖合の稠密津波観測網のデータを用いて津波伝搬の様子を逐次に予測するシステムの開発を急がねばならない。稠密観測網からデータは時々刻々と送られてくる。従って、津波波源と周囲への津波伝搬の推定は、陸域と海域両方の観測網からのデータを時々刻々と取り込みながら、時間を追うごとに精度を上げて推定を行うこととなり、そのための逐次推定システムの開発を目指すこととなる。

この推定には膨大な計算時間が必要となり、極めて高速の計算機の助けがなくては不可能である。警報システムなので、実時間で推定しなければならないからである。現在の計算機の能力では津波伝搬の計算は実時間で行えないので、あらかじめグリーン関数を計算しておきそれをファイルに確保しておくなど、種々の工夫をしながらの開発となろう。これまで地震および地震災害研究分野では、地震発生予測、地震波動伝搬、津波伝搬、および地震動や津波による被害予測の研究が、計算機科学・計算科学の力を借りて精力的に進められてきた。今後組織的に推進されることになるこの「次世代津波警報システム」の開発にも、これまでと同様、計算機科学・計算科学の助力を必要としている。連携がうまくいき、この次世代システムが早期に開発されること、そして大津波への備えが整うことを期待している。

［共同研究成果］

革新的な航空機開発のための次世代 CFD の研究

中橋和博、佐々木大輔
東北大学大学院工学研究科航空宇宙工学専攻

サイバーサイエンスセンター共同研究制度のサポートによりBuilding-Cube Methodと名付けた直交格子ベースの CFD（Computational Fluid Dynamics）の研究を進めている。その進捗状況の報告を兼ねて、航空機開発における CFD の現状と今後についての私見を述べる。

1. 航空機開発における風洞と数値風洞（CFD）

航空宇宙技術研究所（現 JAXA）にスーパーコンピュータが導入された 80 年代初めの頃、“CFD が発展すると風洞がいらなくなる”という意見を述べた人がいて風洞運転を専門としている人たちを不安がらせたことがある。1993 年に航技研に導入された当時世界最高性能スパコンは数値風洞（Numerical Wind Tunnel）と名付けられた。風洞を CFD で置き換えようとの当時の意気込みが分かる。

現在の航空機開発に CFD は不可欠なツールとなっている。NASA CP-2004-213028[1]によると、80 年代に開発されたボーイング 767 では 77 個もの主翼モデルが風洞で試験されたのに対し、最新のボーイング 787 では 5 個で済まされている。CFD による最適化と解析が風洞の試験回数を大幅に減らした。スーパーコンピュータの性能改善はこの流れを加速するだろう。

では、航空機設計に風洞試験は全く不要になるかと問われると Yes とは応えられない。風洞は無くなるどころか海外では更に大型の設備も作られている。CFD が未だに風洞に取って代われないのは計算結果の信頼性に未だに不安があるためである。

ジェット旅客機は高度約 1 万メートルをマッハ 0.8（音速の 0.8 倍）前後で飛ぶ。この巡航時の空力性能が燃費や航続距離に直結するため、風洞試験と CFD には共に高い精度が要求される。例えば胴体に主翼を付けたような旅客機の基本形状モデルでは、空気抵抗係数で C_D=0.0001（これを 1 カウントと呼ぶ）の精度が求められる。これは飛行機にかかる力（揚力、抗力）の 0.02%程度であるが、数千kmを飛ぶ旅客機には 1 カウントでも燃費にはね返る重要な数値である。

10 年程前に米国航空宇宙学会(AIAA)で行われた Drag Prediction Workshop では、20 人程の研究者がそれぞれのソルバー、計算格子、物理モデル（乱流モデル）で結果を持ち寄ったが、それら算出 C_D 値に数十カウントのバラツキがあることが分かり、CFD は風洞試験にはまだまだ敵わないとの思いを強くした。結果のバラツキに大きく影響を与えていた要因の一つは計算格子である。ワークショップで提供された格子と独自の格子では、格子点数がほぼ同じでも空間の格子サイズ分布の違いで算出 C_D 値に 10 カウント以上もの違いが出ることを学んだ。

今日、航空 CFD の計算アルゴリズムはかなり成熟し、RANS 計算（レイノルズ平均ナビエ・ストークス方程式の計算）に必要な乱流モデルも剥離の無い流れではかなり良く乱流境界層を再現する。その意味で、旅客機巡航時の流れは十分に細かな格子を用いればかなりの計算精度が得られる。スーパーコンピュータの性能が 10 年で 1000 倍近く改善されていることを考えると、その圧倒的な性能の計算機では過度とも言える高密度な格子を使うことで計算結果の格子依存を無くし、CFD の信頼性を高めることができよう。CFD が風洞試験に取って代わるのもそう遠く無いのかと思わせる。

２．数値飛行試験

風洞を数値風洞(CFD)で置き換えることができたとしても、"航空機開発のためのCFDは万全"、とは言えない。

旅客機の巡航時の絶対精度は高密度な計算格子を使うことで改善する。しかしながら飛行機は巡航の他に、離陸と着陸、旋回や突風対応等の飛行がある。これらは飛行機の安全性に大きく関係する。離着陸では、最近は脚等からの風切り音の解析が空港騒音問題から重要になってきている。舵の効きも設計者が欲しがるデータである。これら巡航以外の飛行条件は風洞試験では難しく実機の飛行試験に頼っているところも多い。この飛行試験をCFDで代用することが期待されているが、未だに手強い。もちろん計算は可能で流れの様子も観察できるが、非巡航時では、流れの剥離や境界層遷移を伴うため物理モデルの信頼性が絡む。

航空で出くわす通常の流れ（10の7乗のオーダーの高レイノルズ数流れ）では、細かく不規則な運動を含んだ乱流境界層を伴う。この微細な非定常運動までを全機規模の形状モデルで解像することは世界最高性能の京コンピュータを持ってしても未だに困難である。それゆえ航空機の設計現場では、微細な流れの不規則変動を平均操作してRANSなりLES（Large Eddy Simulation）で解く手法が一般的に用いられている。そこに用いる物理モデルが剥離等に伴う複雑な乱流を記述するのに万能ではないために定量的な信頼が得られない。飛行機の失速速度は航空機設計のもっとも重要なパラメータの一つであるが、それすら未だに精度良く予測できないのはこのためである。つまり、剥離を伴った乱流や遷移の物理モデルの不完全性が計算の信頼性の限界を作っており、それが解決しない限りCFDの適用範囲も限られる。

物理モデルを使わずに流れの最小スケールまでを解像するような計算、いわゆるDNS（Direct Numerical Simulation）が飛行機全機の空力設計に実用化されるのは数十年先になると言われているが、そこまで待つ訳にはいかない。そろそろ物理モデルに頼るRANSやLESから抜けなくては、いつまで経ってもCFDは風洞実験や飛行試験に頼らざるを得ないことになる。少なくとも物理モデルへの依存を完全に無くすのは無理にしても、細かな格子でできるだけ微細な渦までを直接計算して物理モデル依存性を極力少なくすることで精度改善を目指すべきである。

スーパーコンピュータの性能改善の勢いがまだ続くことを見込み、これまでのCFDの目標であった数値風洞試験の次の、数値飛行試験を目指した次世代CFDの研究を加速すべき時期である。現在、旅客機にもCO_2排出削減や空港騒音の低減が必須になってきている。そのためには機体形状の革新が必要であるが、それには過去の機体の経験値に頼ってきた従来の設計法は通用しない。数値飛行試験に対する期待は大きい。

３．CFD革新の必要性

私の研究室では非構造格子を用いたCFDソルバー（TAS Code）[2～4]を10年以上前に開発し、航空関連の空力解析・設計に活用している。三菱航空機が開発中のジェット旅客機MRJの空力設計にも多用されて性能改善に役立っている。では、今後の高性能コンピュータ上で数億、数十億点の大規模格子で計算すれば、もっと精度良く、あるいは舵角応答や空力騒音まで解析できるのだろうか？

既に格子移動・変形法を開発して舵角応答や、数千万格子点（セル数では１億以上）を使っての空力騒音源の探査等を試みたが、計算規模が大きくなるにつれ計算の前処理の手間暇が増えた。まず航空機全機の緻密なモデル周りに格子を生成しなくてはならず、飛行機周りの大きな空間に何億もの四面体やプリズムなどのセルを生成させるには演算時間だけでも数時間を要する。流れ場の並列計算には格子をCPU数に応じた数の小領域に分割するが、大規模格子では分割アルゴリズムの演算負荷や信頼性に起因する手間も増加した。舵などの稼働部が入ってくる場合、流れ場の計算中に格子を変形される部分がベクトル演算に乗らないため全体の計算時間が大幅に増えた。非定常流れの計算では出力量も膨大となり計算後処理の負荷増大も厄介である。

計算機性能が向上しても、それに伴いユーザーの作業量も増やすことになるのでは高性能計算機の産業利用は増えない。その意味で、非構造格子での大規模計算は技術的には可能であるものの、工学的な実用ツールとするには多くの課題が残っている。

航空 CFD のアルゴリズム研究は 70 年代から急速に進んだ（図１）。Thompson が提案した境界適合格子（boundary-fitted mesh または構造格子）や、その格子上での効率良い計算アルゴリズム（Steger ら）等で、翼周りの解析が実用化され現在のジェット旅客機の性能向上に寄与した。80 年代後半、計算機性能が更に改善されると翼だけでなく、航空機全機周りの計算がターゲットになる。しかし、それまでの構造格子では 3 次元の機体形状周りに格子を生成するだけで何ヶ月もの時間を要することになった。それを抜本的に解決したのが非構造格子（unstructured mesh）で、90 年代に格子生成とソルバーの両面で飛躍的に研究が進んだ。このことで航空機全機周りの流れの解析から最適設計までを可能にしている。

その非構造格子 CFD も、今後の計算機性能の向上を考えると、先に述べたような人的負担が目立ってくる。新たな発想の計算アルゴリズム開発が必要である。

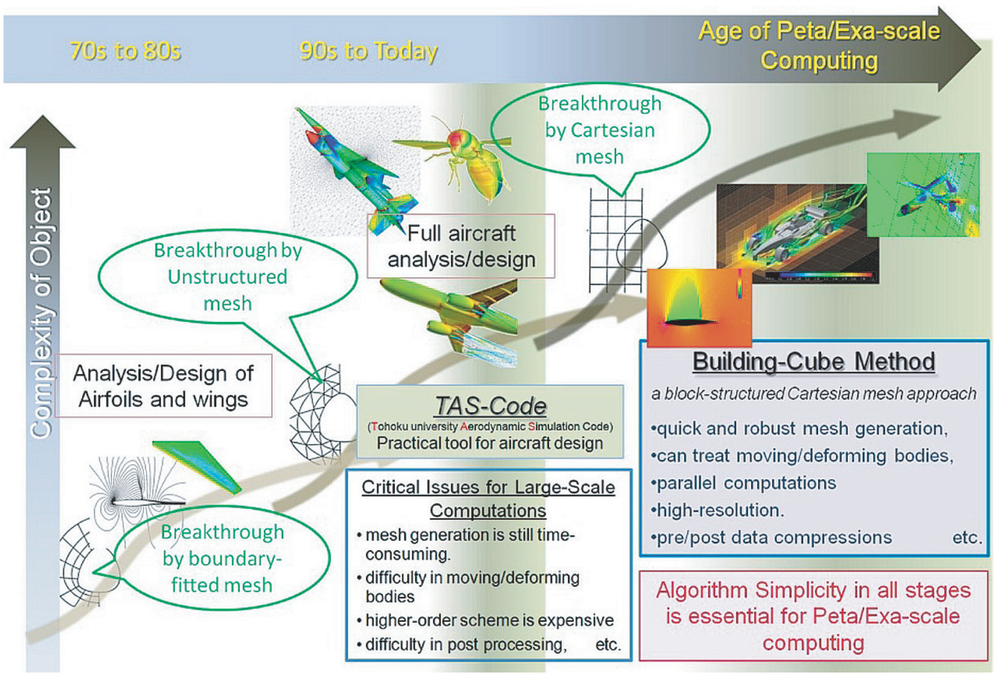


図１．航空 CFD の発展

４．次世代 CFD を目指した Building-Cube Method の開発状況

TAS コード（非構造格子 CFD）の開発を通じて出くわした様々な課題の抜本的な解決のため、Building-Cube Method（BCM）と名付けた直交格子（Cartesian mesh）を用いたソルバー開発をサイバーサイエンスセンターの協力の下で進めている[5,6]。

直交格子は、過去に物体境界での精度等の理由で使われなくなった方法であるが、計算機の高性能化で新たに注目されている。何よりも私が気に入っているのは、非構造格子に比べ圧倒的にシンプルに格子生成もソルバーも構築できることである。出力も直交格子だと大規模データを扱い易い。直交格子 CFD は、計算が大規模化すればするほどアルゴリズムの単純さの点で有利になろう。

BCM では、直交格子の格子幅を局所的に可変にするために Adaptive Mesh Refinement（AMR）を、格子でなく小計算領域に適用するアプローチである。また、図２のように小計算領域を立方体

（cube）で統一し、演算量も総ての cube で同じにすることで、アルゴリズムの単純さを保ちつつ大規模並列計算への対応の容易さを狙っている。

BCM の最終目標は、航空分野 CFD が非設計点でも信頼性を得て真に風洞試験に取って代われる、さらに計算機内での飛行試験も可能にすることである。しかし、その実現には計算機の性能向上をもう暫く待つ必要がある。一方、BCM の利点を活かした実用ツール化の要望も強くなってきた。そのため、現在の計算機環境で工学的な応用を視野に研究開発を進めている。以下に本研究の進捗状況と課題をまとめる。

4.1　BCM 格子生成の高速化と GUI 装備

直交格子法の利点の一つは高速な格子生成である。BCM では、cube 分割による並列化も活用することで数千万の格子を完全自動でパソコン上で１分以内に生成が可能であり[7]、図２のような複雑な形状周りの流れ解析[8]も容易にしている。BCM のこの利点は様々な空力形状を試行錯誤する形状最適設計の効率化にとって重要である。

現在、さらに高速化と信頼性向上のための改善を進めている。また、格子生成条件の設定やユーザーが希望する領域の格子細分化を行うための GUI を開発することで、一般ユーザーが３次元 CAD データを下に計算の前処理を容易に行えるツール構築を進めている。

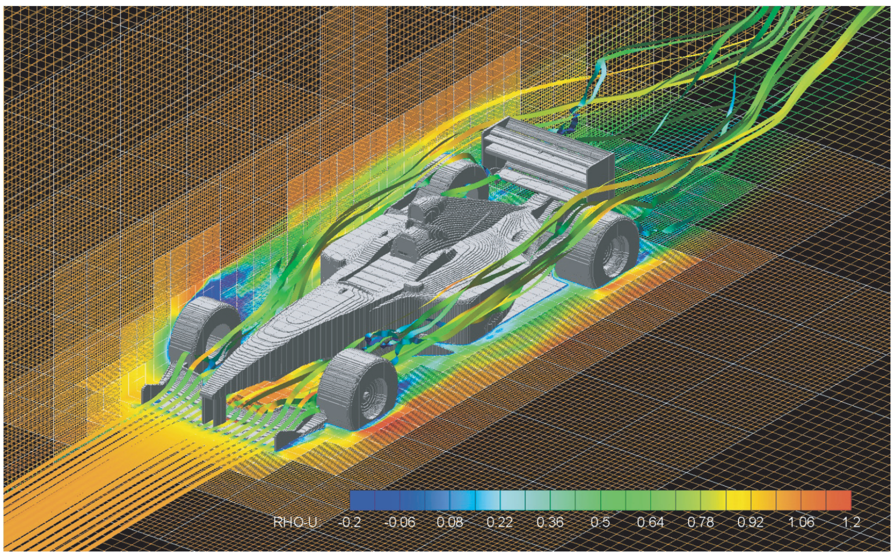


図２．レーシングカー周りの BCM 格子と流れ計算例（計算セル数は約２億）[8]

4.2　非圧縮性流体に対する BCM ソルバー

BCM の複雑形状への対応の容易さを活かし、航空機の脚周りの流れおよび空力騒音の解析を行って定性的に良好な結果を得ている[9]。図３に示すように、タイヤホイールに開けられた涙状の穴を通して流れが漏れ出ている様子も捉えられている。このような複雑な形状は、従来の非構造格子を用いた TAS-Code での計算も不可能では無いが、格子生成等に要する手間暇や大規模計算からはき出されるデータの処理等でユーザーに大きな負担を強いることになる。

しかしながら、物体のボクセル表現に起因する誤差により定量的な精度に課題を残している。それを解決するために埋め込み境界法（Immersed Boundary Method, IBM）や乱流モデルの導入などの研究を進めている。

図３．航空機の脚周りの流れの計算例

4.3　圧縮性流体に対する BCM ソルバー

流線型をした航空機では粘性の影響は比較的小さく、そのために CFD を用いた空力最適化では Navier-Stokes 方程式ではなく Euler 方程式（非粘性流体方程式）の数値解が空力解析時間を短縮するために多用される。一方、従来の非構造格子 CFD を用いた場合、最適化の過程での形状変形などに対して格子の再生成が全体の計算時間を増大させ信頼性も低下させている。BCM では、格子生成は 1､2 桁早い時間で行え且つロバストであることから､空力形状最適化に有効であろう。そのため、BCM に非粘性壁条件のための Immersed Boundary Method[10]を開発するとともに、図４に示すような航空機周り流れ解析や翼の構造と空力とが連成して生じるフラッター（翼振動現象）での移動壁問題への BCM の応用[11]を進めており、その結果として高いスループット性能（格子生成から結果を得るまでの時間の短さ）を持つ圧縮性非粘性流体ソルバーが構築されつつある。このソルバーのベクトル化や MPI 並列による高速チューニングが進めば、非構造格子 CFD ソルバーである TAS の非粘性流版よりも汎用性・頑丈性の高い且つ高速なソルバーが構築されるであろう。

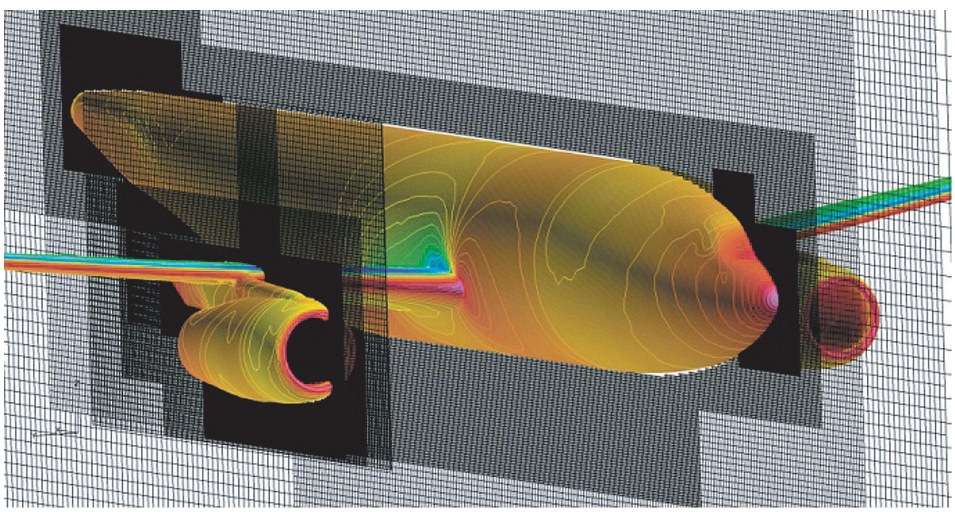


図４．Building-Cube Method による航空機周りの圧縮性非粘性流れ計算[11]

圧縮性Navier-Stokes方程式に対しては、航空機のような高レイノルズ数流れでは物体壁に沿って発達する境界層をいかに精度良く解くかが課題である。そのためのいくつかのアプローチを研究している。境界層部分に非構造プリズム格子を組み合わせるもの[12]、境界層部分にグリッドレス法を組み合わせるもの[13, 14]、そして壁関数等の境界層近似モデルを導入する方法などである。物体に沿って薄く非構造格子を用いる方法は高い精度が得られることを確認しているが、その非構造格子の生成をいかに自動化するかが課題である。

4.4 音の伝播解析のためのLEEソルバー

航空機の低騒音化のための研究の一つとして、圧縮性オイラー方程式を音波成分に対して線形化した線形オイラー方程式(Linearized Euler Equation, LEE)をBuilding-Cube Methodのフレームで解くコード開発を進めている。音波を扱うために空間高次精度な計算法、そして高密度な格子が必要であることから直交格子ベースのソルバーは有利である。また、BCMのアプローチにより複雑な形状にも容易に対応できる。2次元場でのLEEコードの検証[15]を済ませ、さらに図5に示すような、エンジンファンノイズの空間伝播シミュレーションへと拡張している[16]。

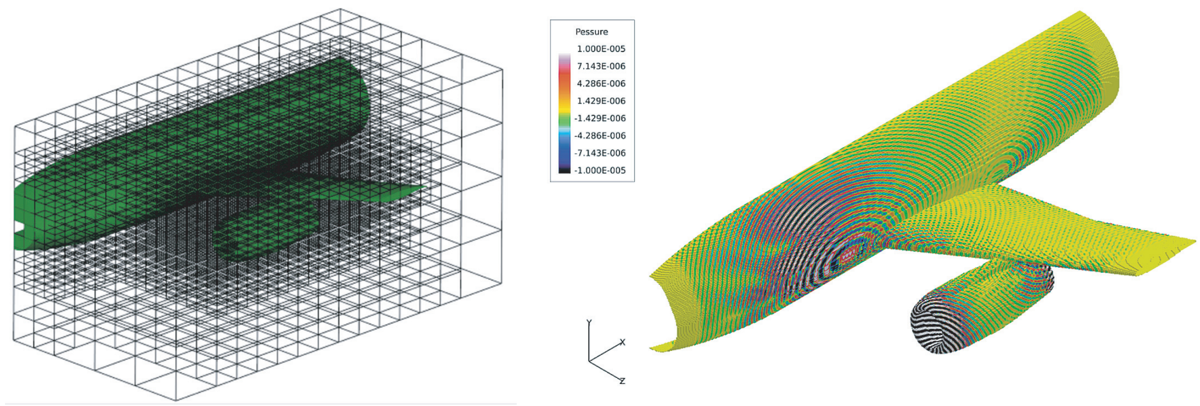


図５．LEEによるエンジンファンノイズの伝播解析

4.5 データ圧縮

大規模計算に伴いCFD解析の出力量も膨大になる。たとえば図６の計算では格子点数が2.4億点であり、その計算では時間ステップ１回につき3.88GBのデータを出力する。流れは非定常であり、何百、何千ものステップのデータ出力がなされ、トータルとしてテラバイトのデータが１計算で出力される。直交格子ベースのBCMでは領域をサブ領域に分けることも相まってデータ圧縮を効率よく行える[17]。その圧縮法をさらに進化させ、図６のテスト例ではオリジナルデータの30分の１近くまでに圧縮する結果を得ている。圧縮したデータからの可視化図（右）はオリジナルデータ（左）とほとんど区別がつかない。

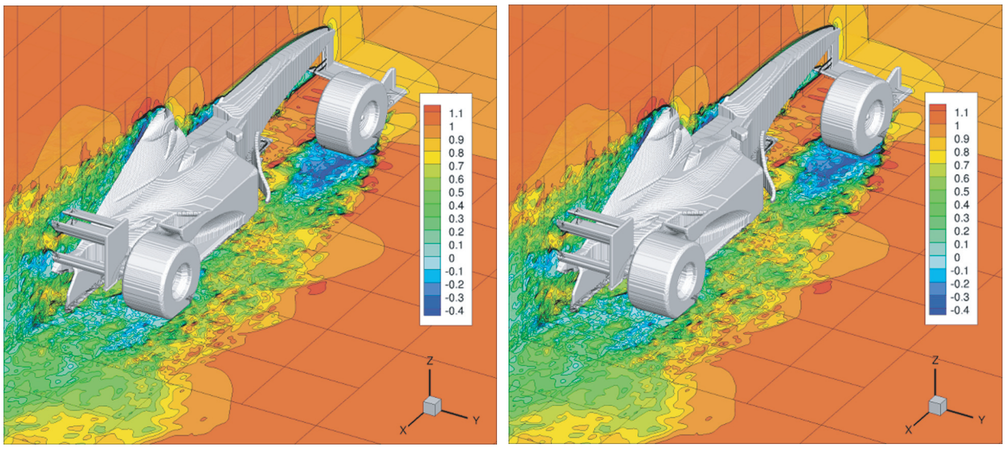


図６．大規模 CFD 出力のデータ圧縮（左：オリジナルデータ可視化図、右：圧縮データからの可視化図）

５．次世代 CFD で我が国の航空機開発に革新を

2001 年、欧州の航空業界は“A Vision for 2020”を発表し、2020 年までに事故率を現在の 1/5 に、CO2 排出を 1/2、そして空港騒音を現在の 1/2 にして欧州の航空機産業を世界一にすることを宣言した。2008 年から始めた、“CLEAN SKY Joint Technology Initiative”も、その具体策の一つである。米国も 2006 年に“National Aeronautics Research and Development Policy” を大統領令として発布し、その施策を着実に進めている。欧米ともに政治レベルで航空産業を強く支援しているのはエアバスとボーイングの競争が背景にあるが、航空技術が国の安全保障に重要であること、さらには他の産業への波及効果として航空技術が未だに頂点の一つであることを強く認識していることの現れである。飛行機が日常生活に溶け込んだ今日でも、それを実際に造り飛ばすには大変な技術が必要である。その技術を国として持つことが技術先進国の発展を牽引するのに不可欠である。

航空機の低騒音化、低 CO_2 排出、そして安全性向上への要求が益々高まる今日、大型風洞設備や飛行試験設備では欧米に遅れている我が国にとって、計算科学への期待は大きい。高性能計算機の開発が進むなか、それを用いて何ができるかが問われており、その一つとして環境と安全で世界を牽引するような革新的な航空機開発を目指すべく、次世代 CFD の研究開発を加速したい。

謝辞

本研究は、東北大学サイバーサイエンスセンターのスーパーコンピュータを利用することで実現することができた。研究にあたっては同センター関係各位に有益なご指導とご協力をいただいた。また、科学研究費 S（課題番号 21226018）の支援をいただいている。

参考文献

[1] “COMSAC: Computational Methods for Stability and Control”, NASA CP-2004-213028/PT1
[2] 中橋和博，藤田健，“非構造格子に基づく流体数値計算法の並列化”，SENAC, Vol.35, No.2, 2002-7.

[3] Nakahashi, K., Ito, Y., Togashi, F., “Some challenges of realistic flow simulations by unstructured grid CFD,” Int. J. for Numerical Methods in Fluids, Vol. 43, pp.769-783, 2003.

[4] 中橋和博，中山亮，渡辺匠，“航空機の空力解析・設計における CFD の活用”, SENAC, Vol.41, No.1, pp.21-28, 2008-1.

[5] Nakahashi, K. “High-Density Mesh Flow Computations with Pre-/Post-Data Compressions,” AIAA 2005-4876, June 2005.

[6] 中橋和博，“CFD におけるペタフロップス級計算機への期待”, SENAC, Vol.38, No.4, 2005-10.

[7] Ishida, T., Takahashi, S., Nakahashi, K., “Efficient and Robust Cartesian Mesh Generation for Building-Cube Method”, Journal of Computational Science and Technology, Vol. 2, No. 4, pp.435-446, 2008.

[8] 高橋俊，石田崇，中橋和博，小林広明，岡部公起，下村陽一,曽我隆,撫佐昭裕,“大規模非圧縮性流体シミュレーションの工学問題への応用”、SENAC Vol.42, No.1, pp.107-114, 2009-1.

[9] 佐々木大輔,恩田博,中橋和博，“Building-Cube 法による非定常流体解析と空力音推定”、SENAC Vol.44, No.1 , pp.33-45 , 2011.1.

[10] Nakahashi, K., “Immersed Boundary Method for Compressible Euler Equations in the Building-Cube Method,” AIAA 2011-3386, June 2011.

[11] 西村康孝，佐々木大輔，中橋和博，“圧縮性 Building-Cube Method を用いた翼周りの非定常非粘性流れ解析”，第 25 回数値流体力学シンポジウム講演集, 2011 年 12 月.

[12] 橋場道太郎，佐々木大輔，中橋和博，“直交・非構造格子重ね合わせによる 3 次元流体ソルバーの構築”，第 49 回飛行機シンポジウム講演集, 2011 年 10 月.

[13] Ishida, T., Kawai, S., Nakahashi, K., “A High-Resolution Method for Flow Simulations with Block-Structured Cartesian Grid Approach”, AIAA 2011-3380, June 2011.

[14] Su, X., Sasaki, D., Nakahashi, K., “A Hybrid Scheme for the Near Wall Treatment of Building Cube Method”, Proc. 8th International Conference on Flow Dynamics, Nov. 2011.

[15] 福島裕馬，佐々木大輔，中橋和博,“ブロック構造型直交格子と IB 法を用いた LEE コードの構築”，日本航空宇宙学会論文集（印刷中）.

[16] 福島裕馬，佐々木大輔，中橋和博,“ブロック構造型直交格子を用いた LEE コードによる主翼騒音遮蔽効果の推定”，第 25 回数値流体力学シンポジウム講演集, 2011 年 12 月.

[17] Sakai, R., Sasaki, D., Nakahashi, K., "Parallel Implementation of Large-Scale CFD Data Compression towards Aeroacoustic Analysis," The 23rd International Conference on Parallel Computational Fluid Dynamics, Barcelona, Spain, 2011.

［共同研究成果］

ランダムな位置に破損素子を有する超大規模アレーアンテナの IEM/LAC による数値解析

今野佳祐, 陳 強, 澤谷 邦男
東北大学大学院工学研究科 電気・通信工学専攻

1　まえがき

近年, 化石燃料に代わる新たなエネルギー源として, 宇宙太陽光利用システム (Space Solar Power Systems, SSPS) が注目を集めている [1], [2]. SSPS は, 宇宙空間に配置したソーラーパネルによって発電した電力をマイクロ波で地上に送電するというシステムである. SSPS の送電部には数億素子規模のアレーアンテナが使用される予定であり, SSPS の実現に先立ってそのような億単位の素子を有する超大規模アレーアンテナの特性を明らかにする必要がある.

さて, アレーアンテナの特性を解析する手法としてモーメント法 (Method of Moments, MoM) が挙げられる [3], [4]. MoM は行列方程式を解くことによってアンテナ上の電流分布を得るという手法であり, 素子間相互結合あるいは運用中に生じた破損素子の影響を考慮した上で電流分布を求められるという利点がある. しかしながら, 未知の電流係数の数を N とすると, インピーダンス行列を保存するメモリが $O(N^2)$, 逆行列を求めるのに必要な計算時間が $O(N^3)$ となってしまうため, MoM で億単位の素子を有する超大規模アレーアンテナを解析することは不可能である.

そこで我々の研究グループでは, SSPS 用超大規模アレーアンテナの特性を明らかにするために, インピーダンス拡張法 (Impedance Extension Method, IEM) をこれまでに提案した [5]. IEM は, 周期的アレーアンテナ中にある各素子の動作インピーダンスの性質に基づく解析法であり, 億単位の素子を有する超大規模アレーアンテナの数値解析が可能である. また, 局所アドミタンス補償 (Local Admittance Compensation, LAC) を提案し, IEM と組み合わせた IEM/LAC を用いることによりランダムな位置に破損素子を含む超大規模アレーの解析も可能にした [6]. 本報告では, IEM/LAC を用いて億単位の素子を有する超大規模アレーアンテナを解析し, 素子間相互結合の利得への影響, 破損素子がメインローブ及びサイドローブに与える影響を検討する.

2　IEM/LAC の原理

図 1 に, IEM の原理を示す. IEM では, 比較的小規模なアレーアンテナ (Small array) を MoM で解析し, 得られた各素子の動作インピーダンスを超大規模なアレーアンテナ (Huge array) の対応する素子にコピーする. そして, 動作インピーダンスと給電電圧から素子の電流分布を求める. IEM を用いることで, 超大規模アレーアンテナにおける素子間相互結合や端効果が動作利得に及ぼす影響を定量的に評価することができる. また, 電流分布を拡張する方法と違い, IEM はテーパ付き給電を与えたアレーアンテナの数値解析を行うことも可能である.

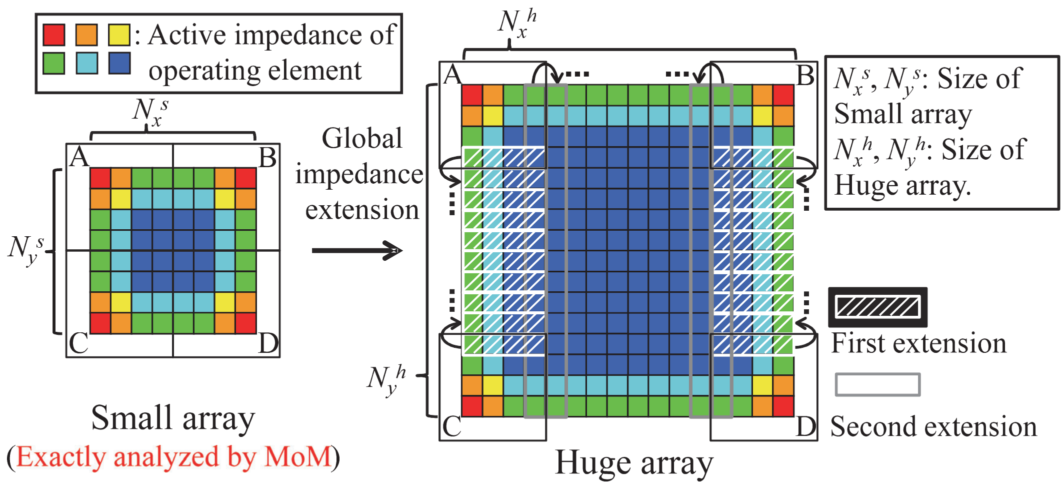


図 1: 8 × 8 Small array から 16 × 16 Huge array への IEM.

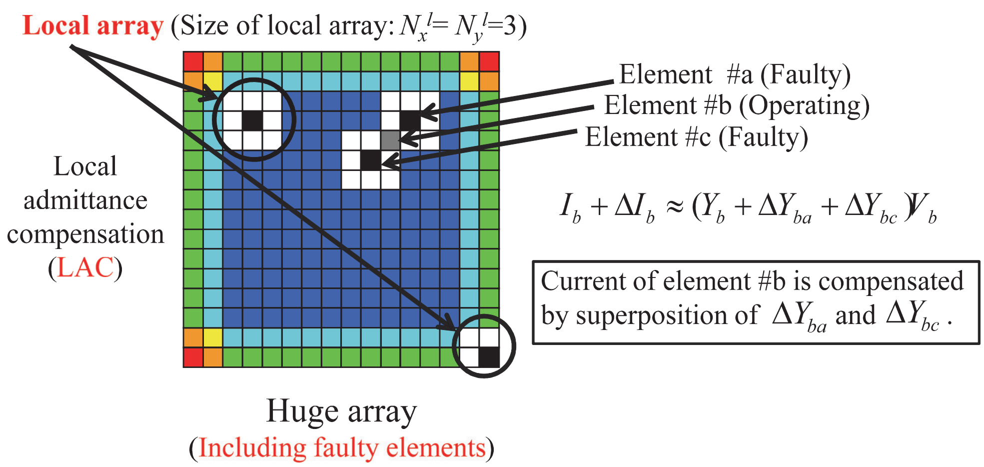


図 2: 8 × 8 Small array から 16 × 16 Huge array への IEM/LAC.

運用中に破損素子が生じた Huge array を解析する際には，図 2 に示すような LAC を実行し，破損素子の影響を組み入れることができる．LAC では，予め 1 つだけ破損素子を含む Small array を全パターン (図の場合は $8 \times 8 = 64$ 通り) 解析し，全く破損素子を含まない Small array の動作アドミタンスをその動作アドミタンスから引いて保存しておく．そして，図 2 に示すように，破損素子の周囲にある素子の動作アドミタンスの変動を組み入れる．LAC では，破損素子の影響を受ける範囲にある素子のことを Local array と呼び，図 2 では $N_x^l = N_y^l = 3$ である．LAC は重畳の理に基づいているため，複数の破損素子から影響を受けた素子の動作アドミタンスの変動も加算によって容易に組み込むことができる．以上のように，IEM と LAC を組み合わせた IEM/LAC を利用することで，ランダムな位置に破損素子を含む超大規模アレーアンテナの数値解析が可能になる．

3　数値解析

3.1　解析モデル

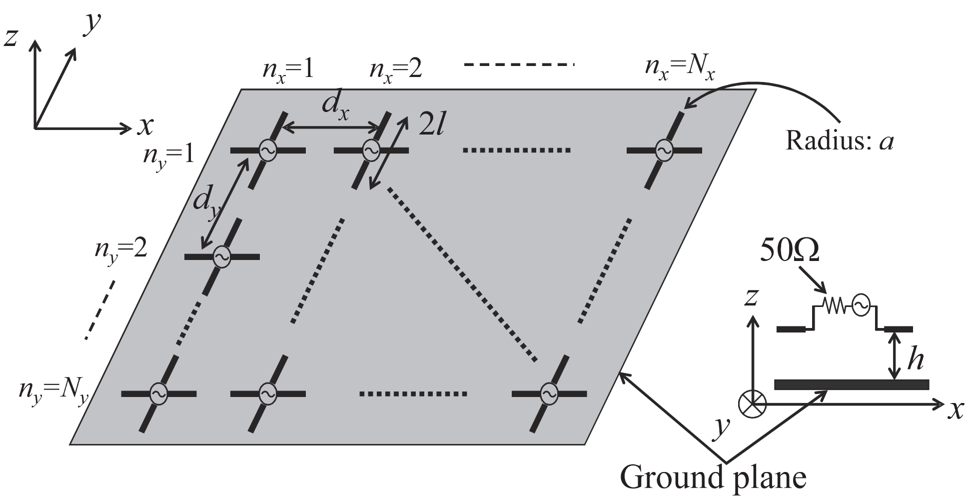


図 3: 反射板付き 2 次元クロスダイポールアレーアンテナ.

図 3 に, 解析モデルである反射板付きの 2 次元クロスダイポールアレーアンテナを示す. 反射板の影響は鏡像法を用いて再現し, Small array の解析にはスーパーコンピュータ向けにベクトル化・並列化チューニングを施した MoM を用いる. 各素子は 3 つのセグメントに分割するものとし, 給電分布はガウス分布に従う 10dB テーパとする. また, 給電位相を変えることによってビーム方向も $(\theta_{main}, \phi_{main})$ へと制御できる.

3.2　超大規模アレーアンテナの IEM による数値解析

表 1: $10,000 \times 10,000$ の Huge array の動作利得における素子間相互結合の影響.

	Actual Gain [dBi]	
	IEM	w/o mutual coupling
Mainlobe ($\theta = 10°$)	82.0	81.7
Sidelobe ($\theta \approx 40°$)	-3.9	-2.6

まず, IEM を用いて超大規模アレーを数値解析し, 動作利得における素子間相互結合の影響を明らかにする. 表 1 に, 1 億素子 (i.e. $N_x^h = N_y^h = 10,000$) の Huge array を IEM で解析して得た動作利得を示す. なお,　mall array のサイズは $N_x^s = N_y^s = 50$, Small array 及び Huge array の素子長 $2l = 0.5\lambda$, 素子半径 $a = 0.00825\lambda$, 素子-反射板間距離 $h = 0.25\lambda$ とし, ビーム方向は $(\theta_{main}, \phi_{main}) = (10°, 0°)$ とした. この Huge array を厳密に解析するのに必要な計算機メモリは約 11520PB に達するため, 厳密解を得るのは不可能である. しかしながら, IEM を用いることで約 19GB 程度まで必要なメモリが削減され, 近似解を得ることができる. 表 1 から, メインローブ

レベルが素子間相互結合の影響によって 0.3dB 増加している一方で, 40° 方向のサイドローブレベルは 1.3dB 低下していることが分かる. SSPS 用アレーアンテナの規模は非常に大きく, 送信電力も大きいため, 素子間相互結合による動作利得の増減は電力効率や EMC の観点から決して無視できない. 従って, SSPS 用超大規模アレーアンテナを設計する際に, 素子間相互結合の影響を考慮する必要があると言える.

3.3　破損素子を含む超大規模アレーアンテナの IEM/LAC による数値解析

表 2: メインローブレベルと破損素子数との関係.

	$\lvert\mathbf{E}_R\rvert^2$ [dB]	
	IEM/LAC	$(1-P)^2$
P =0.1 %	$-0.0096 \sim -0.0094$	-0.0087
P =1 %	$-0.0958 \sim -0.0952$	-0.087
P =10 %	$-0.998 \sim -0.996$	-0.92

破損素子数が動作利得に及ぼす影響を統計的に評価するために, 相対利得 “Relative gain” を以下のように定義する.

$$\text{Relative gain} = \lvert\mathbf{E}_R\rvert^2 = \frac{\lvert\mathbf{E}(P,\theta,\phi)\rvert^2}{\lvert\mathbf{E}(0,\theta,\phi)\rvert^2}. \tag{1}$$

ここで, $\mathbf{E}$ は (θ,ϕ) 方向の遠方界, P は Huge array に占める破損素子の割合である. 本節では, 破損素子がメインローブ及びサイドローブに及ぼす影響を相対利得によって評価する.

前節と同様に, $N_x^h = N_y^h = 10,000$ の Huge array を IEM/LAC によって解析し, 破損素子は割合 P で Huge array 中のランダムな位置に分布しているものとする. Huge array の素子数以外のパラメータは前節までと同じであるが, IEM/LAC における Local array サイズは数値解析に基づいて $N_x^l = N_y^l = 3$ (i.e., $1.5\lambda \times 1.5\lambda$) と最適化してある. 前節で説明したように, この Huge array を厳密に解くのに莫大な計算機メモリを要するため, 厳密解を求めることは不可能であるが, 厳密解が得られる規模の Huge array を解析するのに要した時間から推定すると, この規模の Huge array を 1 度解析するには最低でも 2,400 日以上かかることになる. 一方で, IEM/LAC による数値解析に要した計算時間は 1 回当たり約 15 秒であり, 非常に短時間である. 本節では統計的な解析を行うため, 試行回数 M_t を 2,000 回とした. Huge array における破損素子数と相対利得との関係を表 2 に示す. 表 2 から, P を固定した場合に IEM/LAC によって得られたメインローブの相対利得はほぼ一定値となっていることが分かる. これは, メインローブでは各素子からの電界が位相差のない状態で足し合わせられており, 生じた破損素子の数に応じて一定程度必ず相対利得が低下するからである. なお, 文献 [7] では, そのときの相対利得が $(1-P)^2$ となることが示されているが, IEM/LAC によって得た結果とよく一致していることが分かる. また, IEM/LAC によって得られた結果と $(1-P)^2$ との間に若干の差が生じているのは, 素子間相互結合の影響や給電振幅分布にテーパをかけた分布を与えたことによるものだと考えられる.

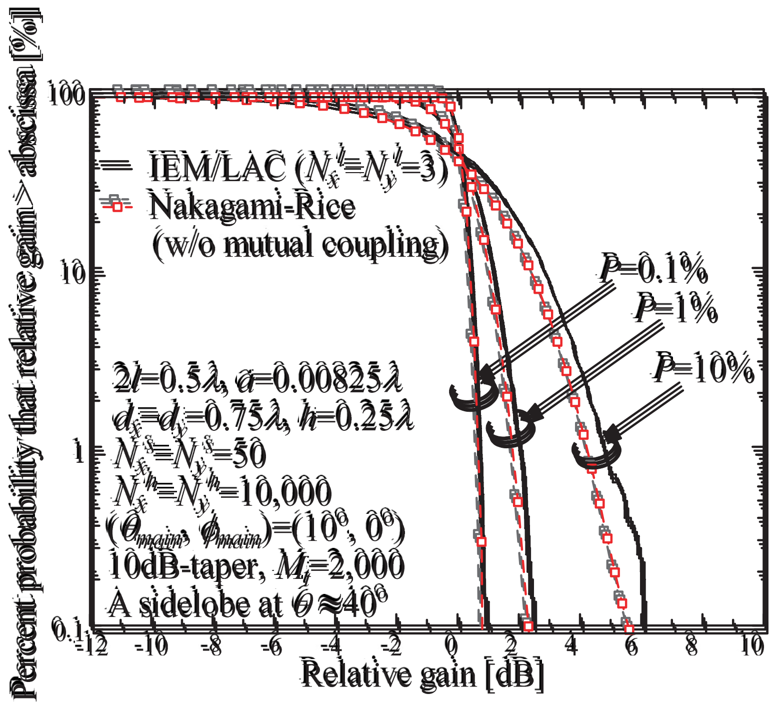


図 4: $\theta \approx 40°$ 方向のサイドローブの相対利得が横軸を超えるパーセント確率.

同じ Huge array において, $\theta \approx 40°$ 方向へのサイドローブの相対利得が横軸の値を超えるパーセント確率を求め, 結果を図 4 に示す. なお, 素子の電流振幅や位相にランダムな誤差を含むアレーアンテナに関する過去の研究では, 相対振幅値の確率密度関数が仲上–ライス分布であるということが報告されてきたため, 仲上–ライス分布による曲線も比較のために示す [8], [9]. (1) 式に示した相対利得の定義から, 図 4 において 0dB を超える, または下回る値の相対利得は, 破損素子によるサイドローブレベルの上昇または低下をそれぞれ示している. 図 4 から, ランダムに破損素子が分布した Huge array においては 10% の確率で, 0.4 dB (@$P = 0.1\%$), 1.1 dB (@$P = 1\%$), そして 2.8 dB (@$P = 10\%$) ほどサイドローブレベルが上昇することが分かる. このことは, サイドローブでは各素子からの電界が位相差を持って足し合わせられているため, メインローブと違い破損素子の数だけではなくその分布によって上昇・低下が決まることを意味している. また, 素子間相互結合を無視した仲上–ライス分布の結果と比較すると IEM/LAC によって得られた結果は高い相対利得を示しており, 破損素子の影響は素子間相互結合を考慮して評価すべきであることも分かる. 超大規模なアレーアンテナの場合, サイドローブレベルも非常に大きくなるため, このようなサイドローブレベルの上昇は EMC の観点から非常に重大な問題である.

4　むすび

本報告では, 周期的アレーアンテナの性質に基づいて SSPS 用超大規模アレーアンテナを解析する手法である IEM/LAC について概説した. 次に, 破損素子のない 1 億素子の超大規模アレーアン

テナをIEMによって数値解析し，素子間相互結合が動作利得に及ぼす影響を定量的に明らかにした．そして，破損素子を含む1億素子の超大規模アレーアンテナをIEM/LACによって数値解析し，破損素子の数及び分布が動作利得に及ぼす影響を統計的に明らかにした．その結果，メインローブは破損素子の数が増加すると単調に低下する一方，サイドローブは破損素子の分布によっては上昇し得ることが分かった．

謝辞

本研究は，東北大学サイバーサイエンスセンターのスーパーコンピュータSX-9を利用して行われたものである．関係各位に深く感謝する．

参考文献

[1] P.E. Glaser, "Power from the sun: Its future," Science, vol.162, pp.857-861, Nov. 1968.

[2] P.E. Glaser, "An overview of the solar power satellite option," IEEE Trans. Microw. Theory Tech., vol.40, no.6, pp.1230-1238, June 1992.

[3] R.F. Harrington, Field Computation by Moment Methods, Macmillan, New York, 1968.

[4] J.H. Richmond and N.H. Geary, "Mutual impedance of nonplanar-skew sinusoidal dipoles," IEEE Trans. Antennas Propag., vol.AP-23, no.3, pp.412-414, May 1975.

[5] K. Konno, Q. Chen, K. Sawaya, and T. Sezai, "Analysis of huge-scale periodic array antenna using impedance extension method," IEICE Trans. Commun., vol.E92-B, no.12, pp.3869-3874, Dec. 2009.

[6] K. Konno, Q. Chen, K. Sawaya, and T. Sezai, "Statistical analysis of huge-scale periodic array antenna including randomly distributed faulty elements," IEICE Trans. Commun., vol.E94-C, no.10, pp.1611-1617, Oct. 2011.

[7] M.I. Skolnik, J.W. Sherman, III, and F.C. OGG, Jr., "Statistically designed density-tapered arrays," IEEE Trans. Antennas Propag., vol.12, no.4, pp.408-417, July 1964.

[8] J.K. Hsiao, "Normalized relationship among errors and sidelobe levels," Radio Sci., vol.19, no.1, pp.292-302, Janu.-Feb. 1984.

[9] R.J. Mailloux, Phased Array Antenna Handbook, Artech House, Boston, London, 1994.

［大規模科学計算システム］高速化推進研究活動報告第５号より転載

並　列　処　理

スーパーコンピューティング研究部　江川隆輔　岡部公起

情報部情報基盤課　伊藤英一　小野敏　山下毅

日本電気株式会社　撫佐昭裕　神山典　小久保達信　金野浩伸

NEC システムテクノロジー株式会社　曽我隆　塩田和永

6.1 並列処理の概要

並列処理とは，並列に実行可能な処理を複数のプロセッサ（あるいはコア）を用いて実行することである．単一のプロセッサでのみ実行可能なプログラムを逐次処理プログラムと呼び，逐次処理プログラムを並列処理プログラムに書き換えることを並列化と言う．並列化を行うことにより，プログラムの実行時間を短縮することができる．

並列処理には，コンピュータアーキテクチャに応じた以下の方式がある．

(1)　共有メモリ型並列処理

共有メモリ型並列処理とは，図 6.1-1 のように複数のプロセッサが単一のメモリ空間を共有しながら並列処理を行うものである．これは，コンパイラの自動並列化機能やコンパイル指示行の挿入により並列化を行うものである．東北大学サイバーサイエンスセンターにおける SX-9 では 16 並列，Express5800 では 32 並列まで実行することができる．

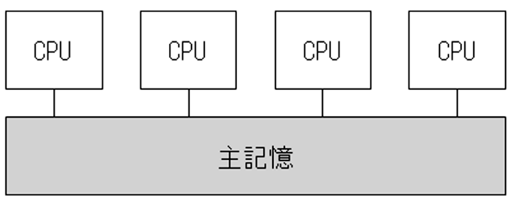


図 6.1-1　共有メモリマシンのアーキテクチャ

(2)　分散メモリ型並列処理

分散メモリ型並列処理とは，図 6.1-2 のようにネットワークを介して接続されている，複数の独立したメモリ空間を持つコンピュータで行われる並列処理のことである．この並列処理では，処理を分割し異なるプロセッサで別々に実行する．このとき，プロセッサ間で情報交換が必要な場合，MPI（Message Passing Interface）等の通信ライブラリを用いてデータのやり取りを行う．東北大学サイバーサイエンスセンターにおける SX-9，Express5800 では共に 64 並列まで実行することができる．

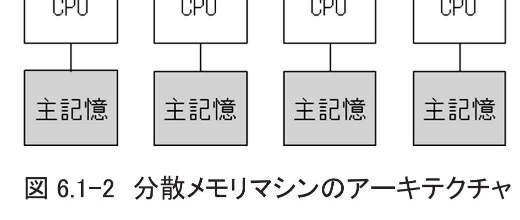


図 6.1-2　分散メモリマシンのアーキテクチャ

6.2 MPI プログラムの概要

6.2.1 MPI とは

MPI とは，MPI フォーラム(注 1)により開発・規格化された分散メモリ型並列処理におけるデータ通信のための標準規格であり，複数のプロセス(注 2)間でデータをやり取りするために用いるメッセージ通信操作のサブプログラム仕様の規格である．

主な特徴を以下に示す．

- FORTRAN，C から呼び出す通信操作のサブプログラムであり，ライブラリで提供される．
- 様々なコンピュータシステムの MPI 実装環境において同じソースコードで利用できる．
- 共有メモリ型並列処理に比べてプログラマの負担が大きいが，大規模計算が行える．

(注　1)MPI フォーラムの公式 Web ページは http://www.mpi-forum.org である．
(注　2)プロセス：コンピュータのプロセッサ上で自律的に動作するプログラムの実体

6.2.2 MPI のプログラム例

ここでは，総和計算を行う FORTRAN プログラムを例に MPI プログラムの説明を行う．

(1)　総和計算のプログラム

図 6.2-1 は，1 から 100 までの整数の和を求めるコードである．

```
program main
parameter(N=100,i1=1,i2=N)
isum = 0
do i = i1, i2
  isum = isum + i
enddo
write(6,*) "sum=",isum
stop
end program
```

図 6.2-1　総和計算のコード例

図 6.2-1 に示した総和計算の例では，1 から 100 まで順に和を求めている．この総和計算では，DO ループを複数に分割し，分割したループごとに部分和を取り，それぞれの部分和を集計して求めることができる．このように DO ループを分割した部分について，部分和を並列に実行することで並列処理することができる．

(2)　処理の分割

図 6.2-2 は，DO ループを均等に 4 分割し，4 プロセスで実行する場合の概念図である．各プロセスがそれぞれ DO ループのどの部分を担当するか，プロセス識別番号(0～3)で表している．MPI ではプロセスの識別番号のことを rank 番号と呼ぶ．各プロセスは担当する DO ループの部分和を求める計算を行う．

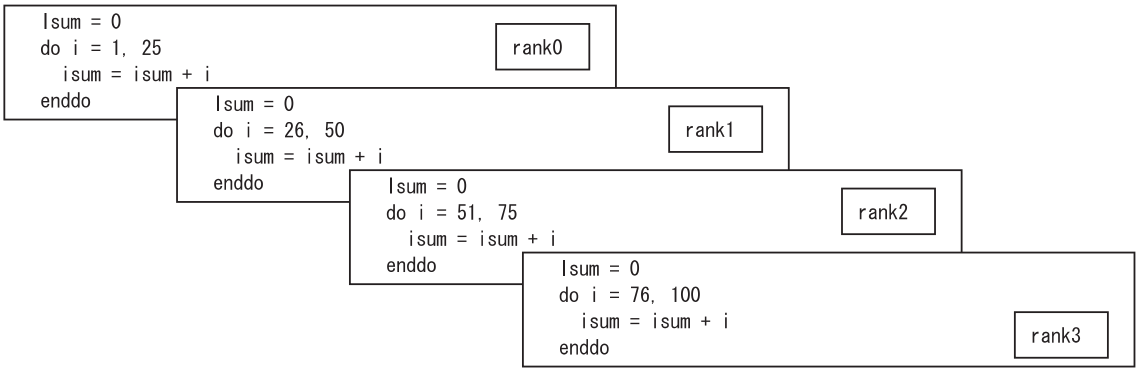


図 6.2-2　DO ループ分割の概念図

図 6.2-2 は並列数 4 の例であるが，プロセス数に応じて，各プロセスが分担する領域の始点と終点を下式①，②のように rank 番号から求める必要がある．ここで nprocs が全プロセス数を，myrank が各プロセスの rank 番号を表している．ただし，各プロセスでループを均等に分割できることを前提にしている．

①　始点　ist = ((i2−i1+1)/nprocs)*myrank+1
②　終点　ied = ((i2−i1+1)/nprocs)*(myrank+1)

この式によって求められる ist，ied は表 6.2-1 のとおりである．nprocs，myrank を使用することにより，図 6.2-3 のように各プロセスの処理を共通のコードで表すことができる．

表 6.2-1　rank 番号ごとの ist，ied の値

nprocs=4	ist	ied
myrank=0	1	25
myrank=1	26	50
myrank=2	51	75
myrank=3	76	100

```
ist = ((i2-i1+1)/nprocs)*myrank+1
ied = ((i2-i1+1)/nprocs)*(myrank+1)
isum = 0
do i = ist, ied
  isum = isum + i
enddo
```

図 6.2-3　部分和のコード例

(3)　プロセス間通信

全体の総和は，各プロセスで求められた部分和を代表プロセスで集計して求める．集計には MPI による通信ライブラリを使用する．図 6.2-4 は，プロセス間の通信と部分和を集計する例であり，データの送受信を行う MPI の一対一通信サブルーチン（MPI_RECV，MPI_SEND）を用いている．rank 番号が 0 以外のプロセスは，それぞれの部分和を rank 番号 0 に送信する．rank 番号 0 のプロセスは，自プロセス以外のすべてのプロセスから送られる部分和を（nprocs−1）回受信し，受信した部分和 itmp を isum に集計する．

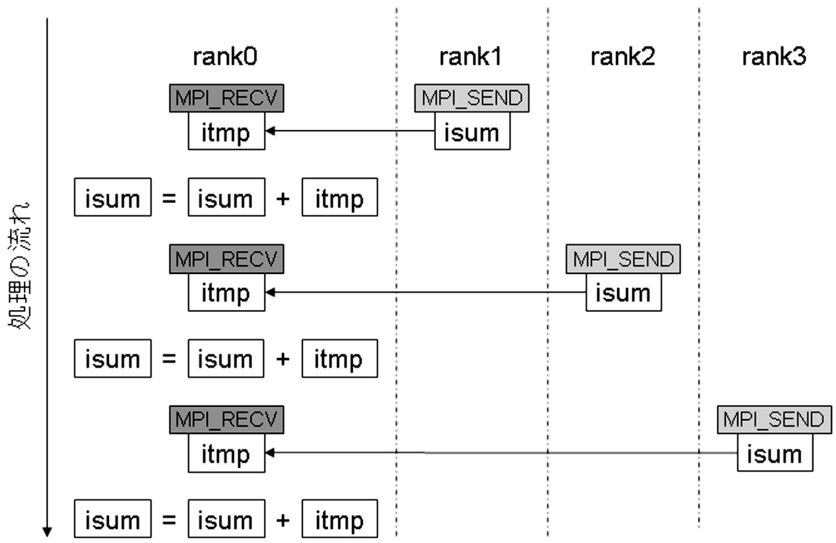


図 6.2-4　総和処理の流れ

(4)　総和計算の MPI コード例

以上を総合して，図 6.2-1 のプログラムを並列化したコードを図 6.2-5 に示す.

```
program example1
include 'mpif.h'                                                            !①
integer status(MPI_STATUS_SIZE)                                             !②
parameter(N=100,i1=1,i2=N)
integer nprocs,myrank,source,tag,ierr,ist,ied,i,isum,itmp
call MPI_INIT(ierr)                                                         !③
call MPI_COMM_SIZE(MPI_COMM_WORLD,nprocs,ierr)                              !④
call MPI_COMM_RANK(MPI_COMM_WORLD,myrank,ierr)                              !⑤

ist = ((i2-i1+1)/nprocs)*myrank+1
ied = ((i2-i1+1)/nprocs)*(myrank+1)
isum = 0
do i = ist, ied
  isum = isum + i
enddo

if(myrank .eq. 0) then
  do i = 1, nprocs-1
    source = i
    tag = i
    call MPI_RECV(itmp,1,MPI_INTEGER,source,tag,MPI_COMM_WORLD, status,ierr)  !⑥
    isum = isum + itmp
  enddo
else
  tag = myrank
  call MPI_SEND(isum,1,MPI_INTEGER,0,tag,MPI_COMM_WORLD,ierr)               !⑦
endif
if(myrank .eq. 0) write(6,*) "sum=",isum
call MPI_FINALIZE(ierr)                                                     !⑧
stop
end
```

図 6.2-5　並列化コード例（総和計算）

この例で使用したMPIの宣言部分とサブルーチンについて以下に示す.

① mpif.h
MPIで予約されている定数が定義されている. MPIサブルーチンを利用するすべてのサブルーチン, 関数においてinclude文で引用する必要がある.

② status(MPI_STATUS_SIZE)
下記MPI_RECVの第七引数で使用されるメッセージ情報.

③ MPI_INIT
MPIの実行開始を宣言するサブルーチンであり, すべてのMPIサブルーチンに先立ち, 呼び出される必要がある. 引数(ierr)に実行結果の状態を受け取る(以下同様).

④ MPI_COMM_SIZE
MPIのプロセス管理を行うサブルーチンであり, 総プロセス数の問い合わせを行う. 第一引数はMPI通信のためのコミュニケータ(注1)“MPI_COMM_WORLD”を指定する. 第二引数(nprocs)に総プロセス数を受け取る.

⑤ MPI_COMM_RANK
MPIのプロセス管理を行うサブルーチンであり, 自プロセスのrank番号の問い合わせを行う. 第二引数(myrank)に自rank番号を受け取る.

⑥ MPI_RECV
MPIの通信処理を行うサブルーチンであり, データの受信をブロッキング(同期)モード(注2)で行う. このサブルーチンは, 一対一通信と呼ばれる分類に属し, 通信相手が必要になる. 送信側が送ったデータを受信するために呼び出される. 第一引数(itmp)は受信データの開始アドレス, 第二引数(1)は受信データの要素数, 第三引数(MPI_INTEGER)は受信データのタイプ, 第四引数(source)は通信相手の rank 番号, 第五引数(tag)はタグ(送受信は同じタグ間で行われる), 第六引数(MPI_COMM_WORLD)はコミュニケータ, 第七引数(status)はメッセージ情報を受け取る.

⑦ MPI_SEND
MPIの通信処理を行うサブルーチンであり, データの送信をブロッキング(同期)モードで行う. このサブルーチンは, 一対一通信と呼ばれる分類に属し, 通信相手が必要になる. 受信側にデータを送信するために呼び出される. 第一引数(isum)は送信データの開始アドレス, 第二引数(1)は送信データの要素数, 第三引数(MPI_INTEGER)は送信データのタイプを示す. 以降の引数はMPI_RECVと同じである.

⑧ MPI_FINALIZE
MPIの終了処理を行うものであり, すべてのMPIサブルーチンが呼ばれた最後で呼び出す必要がある.

(注 1)コミュニケータ:通信に参加する MPI プロセスを集めてできたグループにつけられる一種の名前のようなものである. この識別は通信相手のプロセスを特定する場合に必要となる. 利用者は, プログラムにおいて複数のコミュニケータを定義することもできる.

(注 2)ブロッキング(同期)モード:対応する送信手続きと受信手続きのどちらかを先に呼び出した場合に, 両方の呼び出しが揃うまで先行した側の MPI プロセスは待ち合わせを行う処理モードである.

6.3 領域分割法

6.3.1 領域分割法の例

領域分割法とは，図 6.3-1 のようにシミュレーションの対象となる空間を分割し，複数のプロセスで処理を分担して実行するものである．分割した空間の間で，それぞれのプロセスがデータの参照を必要とする場合，MPI を用いてデータの転送を行う．

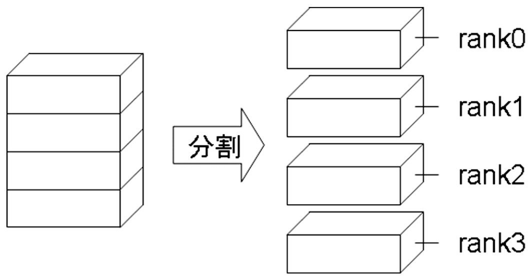


図 6.3-1 領域分割

図 6.3-2 に，差分コードの例を示す．この差分コードの領域を均等に 2 分割し，2 プロセスで実行する場合のコードを図 6.3-3 に示す．

```
 do i = 1, 10
   a(i) = i
enddo
 do i = 2, 10
   b(i) = a(i) - a(i-1)
enddo
```

図 6.3-2 差分コード

```
 do i = 1, 5              rank0
   a(i) = i
enddo
 do i = 2, 5
   b(i) = a(i) - a(i-1)
enddo
```

```
 do i = 6, 10             rank1
   a(i) = i
enddo
 do i  = 6, 10
   b(i) = a(i) - a(i-1)
enddo
```

図 6.3-3 領域分割のコード

図 6.3-3 のように分割する場合，rank 番号 1 は i=6 のときの計算に必要な a(5)のデータが領域外となる．このとき，図 6.3-4 で示されるように a(5)のデータは rank 番号 0 の領域内にあり，rank 番号 1 は rank 番号 0 から a(5)のデータを受け取る必要がある．

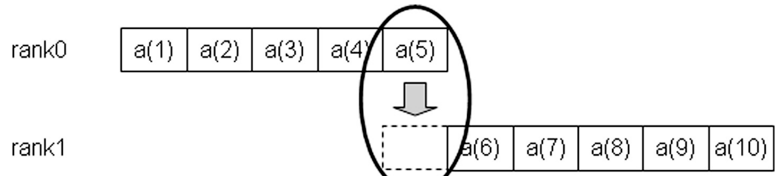


図 6.3-4　境界面で不足するデータのイメージ

6.3.2　領域分割法におけるデータ転送

領域分割法では，自プロセスで演算に必要となるデータが担当する領域の外にある場合，そのデータが含まれる領域を担当するプロセスから，データを受け取る必要がある．このデータ転送がオーバヘッドとなり，並列性能を低下させる要因となる場合がある．

図 6.3-5 はポアソン方程式をヤコビ法で解くコードである．ここでは i, j, k の三次元空間を扱っている．これを用いて領域分割の違いによるデータ転送のコストについて説明する．

```
30: +------>         do loop =1,n
31: |+----->           do k = 2,kmax-1
32: ||+---->             do j = 2,jmax-1
33: |||V--->               do i = 2,imax-1
34: ||||                     u(i,j,k)=s*(uu(i+1,j,k)+uu(i-1,j,k)
35: ||||           &                    +uu(i,j+1,k)+uu(i,j-1,k)
36: ||||           &                    +uu(i,j,k+1)+uu(i,j,k-1)
37: ||||           &                    +a(i,j,k))
38: |||V---                enddo
39: ||+----              enddo
40: |+-----            enddo
41: |                  err=0.0d0
42: |+----->           do k = 2,kmax-1
43: ||+---->             do j = 2,jmax-1
44: |||V--->               do i = 2,imax-1
45: ||||                     err=err+(uu(i,j,k)-u(i,j,k))*(uu(i,j,k)-u(i,j,k))
46: |||V---                enddo
47: ||+----              enddo
48: |+-----            enddo
49: |                  if(err .lt. eps) go to 100
50: +------            end do
51: +------>           do k = 2,kmax-1
52: |+----->             do j = 2,jmax-1
53: ||V---->               do i = 2,imax-1
54: |||                      uu(i,j,k)=u(i,j,k)
55: ||V----                enddo
56: |+-----              enddo
57: +------            enddo
58:             100  continue
```

図 6.3-5　ポアソン方程式をヤコビ法で解くコード

三次元空間を j 方向または k 方向で 4 分割することを考える．ただし，配列 uu はサイズ(imax,n,n)の倍精度実数とする．図 6.3-5 の 34-37 行目の配列 uu は 3 次元のそれぞれの方向に差分式となっている．どの方向に分割しても隣接しているデータ(境界面)を参照しているため，MPI を用いてデータ転送する必要

がある．以下に k 方向，j 方向に分割したときの通信量，通信回数について説明する．

(1)　k 方向に分割したとき

図 6.3-5 の 34-37 行目の配列 uu は分割した空間の境界面のデータを参照（3 次元目が k+1, k-1）しているため，MPI を用いて転送する必要がある．この例では，MPI_SEND，MPI_RECV を用いて転送する．k 方向に分割した場合，境界は ij 平面となりメモリ上に連続に配置されているので，1 つの境界面あたり 1 回の通信でデータを送受信することができる．図 6.3-6 に rank 番号 1 におけるデータ転送を示す．1 度の通信で（imax×n×8）Byte のデータを転送し，必要な転送は 2 回のみとなる．

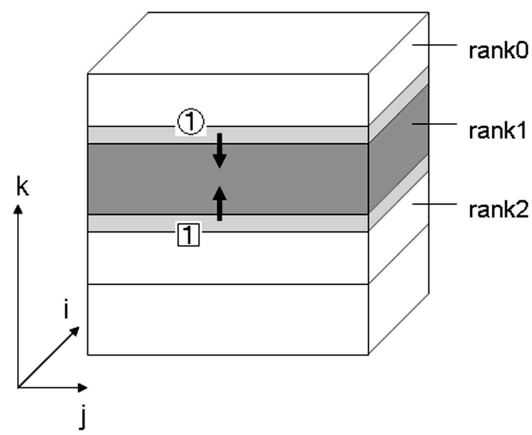


図 6.3-6　k 方向で分割したときのデータ転送

図 6.3-7 は上に示したデータ転送の通信例である（数字は図 6.3-6 と対応）．rank 番号 1 は，rank 番号 0，2 から境界面のデータを受信する必要がある．

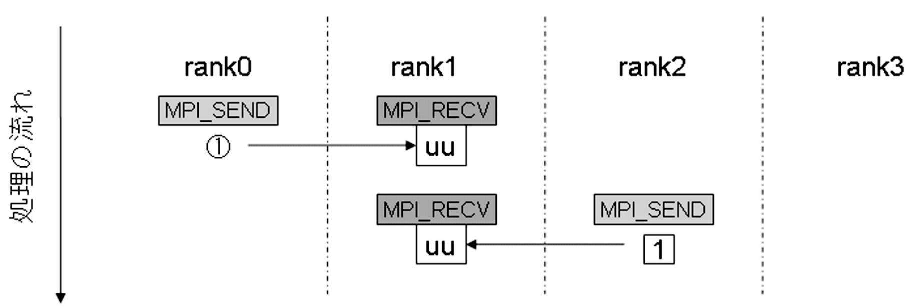


図 6.3-7　k 方向で分割したときの通信

(2)　j 方向に分割したとき

配列 uu は分割した空間の境界面のデータを参照（2 次元目が j+1, j-1）しているため，MPI を用いて転送する必要がある．図 6.3-8 に MPI_SEND，MPI_RECV を使用した場合の rank 番号 1 におけるデータ転送を示す．メモリ上に連続に配置されているデータは i 方向のみであるため，1 度の通信で転送できるデータは（imax×8）Byte であり，（n×2）回の転送が必要となる．

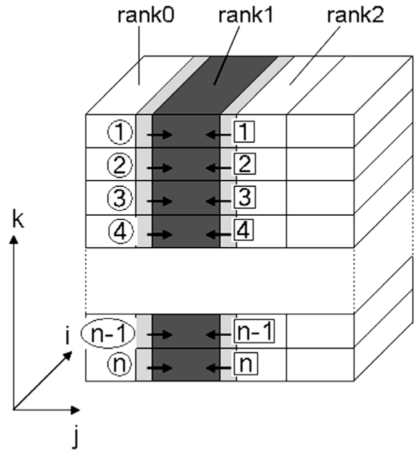


図 6.3-8　j 方向で分割したときのデータ転送

図 6.3-9 は上に示したデータ転送のプロセス間の通信例である（数字は図 6.3-8 と対応）．rank 番号 1 は，rank 番号 0，2 から境界面のデータを受信する必要がある．

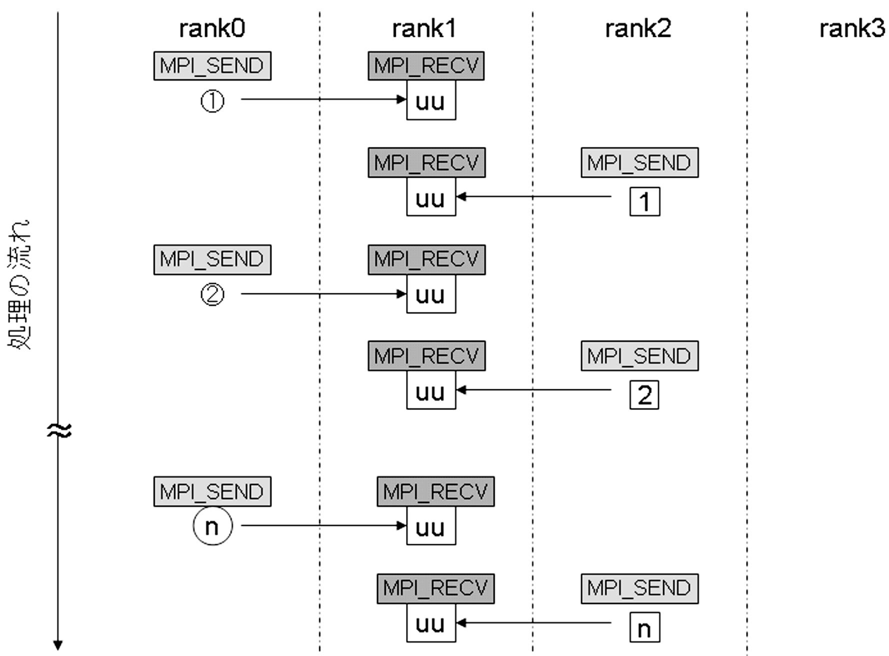


図 6.3-9　j 方向で分割したときの通信例

6.3.3 領域分割法による通信性能

図 6.3-5 にあるポアソン方程式をヤコビ法で解くコードを，問題サイズ「imax=1280，jmax=128，kmax=128」として性能評価を行う．このコードを MPI で分散並列化し，16 プロセスで並列実行するときの通信量，呼び出し回数，総通信量を表 6.3-1 に示す．jmax=kmax であるため，j，k の分割方向によらず総通信量は同じである．

表 6.3-1　通信量，呼び出し回数，総通信量

	1 回あたりの MPI 通信量	呼び出し回数	総通信量
k 方向に分割	約 1.25MB	20,000	約 25,000MB
i 方向に分割	約 10KB	2,560,000	約 25,000MB

上記条件で，SX-9 と Express5800 それぞれで実行する．図 6.3-10 に分割方向の違いによる通信時間のグラフを示す．総通信量は同じであるが，SX-9，Express5800 ともに，通信回数の少ない k 方向の分割の方が通信時間は短いことがわかる．このように並列性能の向上のためには，通信回数が少なく，また 1 度の通信で多くのデータを送受信できる適切な分割方法を検討することが重要になる．

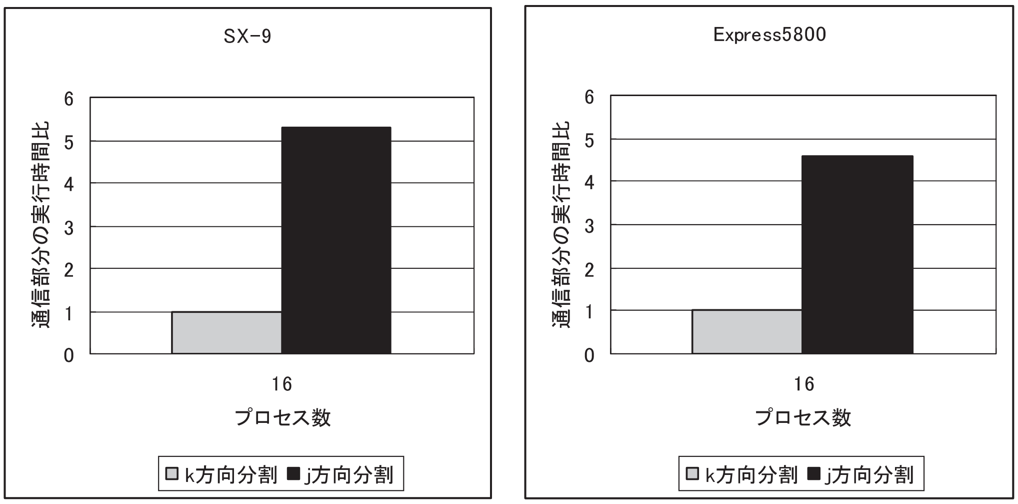


図 6.3-10　分割方向の違いによる通信時間

6.4　領域分割法における MPI コード例

本節では，6.3 節のポアソン方程式を領域分割法によって MPI 化した実際のコードについて説明する．

6.4.1　MPI による隣接領域間の転送

6.3 節でも述べたが，図 6.4-1 の 34-37 行目の配列 uu は分割した空間の境界面のデータを参照している差分式であるため，隣接領域間の転送が必要である．

```
30: +------>         do loop =1,n
31: |+----->           do k = 2,kmax-1
32: ||+---->             do j = 2,jmax-1
33: |||V--->               do i = 2,imax-1
34: ||||                     u(i,j,k)=s*(uu(i+1,j,k)+uu(i-1,j,k)
35: ||||            &                   +uu(i,j+1,k)+uu(i,j-1,k)
36: ||||            &                   +uu(i,j,k+1)+uu(i,j,k-1)
37: ||||            &                   +a(i,j,k))
38: |||V---                enddo
39: ||+----              enddo
40: |+-----            enddo
 :
57: +------            enddo
```

図 6.4-1　ポアソン方程式における差分式（逐次処理コード）

図 6.4-2 に領域を分割したとき，差分式の演算で必要な境界面のデータ通信を示す．k 方向に領域を 4 分割すると，①～④の切り出した領域を各プロセスが分担する．分割した領域には隣接した空間の境界面のデータをそれぞれ冗長に持つ必要がある．具体的には，①は B，②は A と D，③は C と F，④は E の境界面のデータを保有する．ただし，これら境界面は隣接したプロセスが演算結果を更新するため，自プロセスでは更新前の古いデータを保持することなる．そのため，各プロセスは隣り合う rank 番号に対して更新されたデータの送受信を行う必要がある．

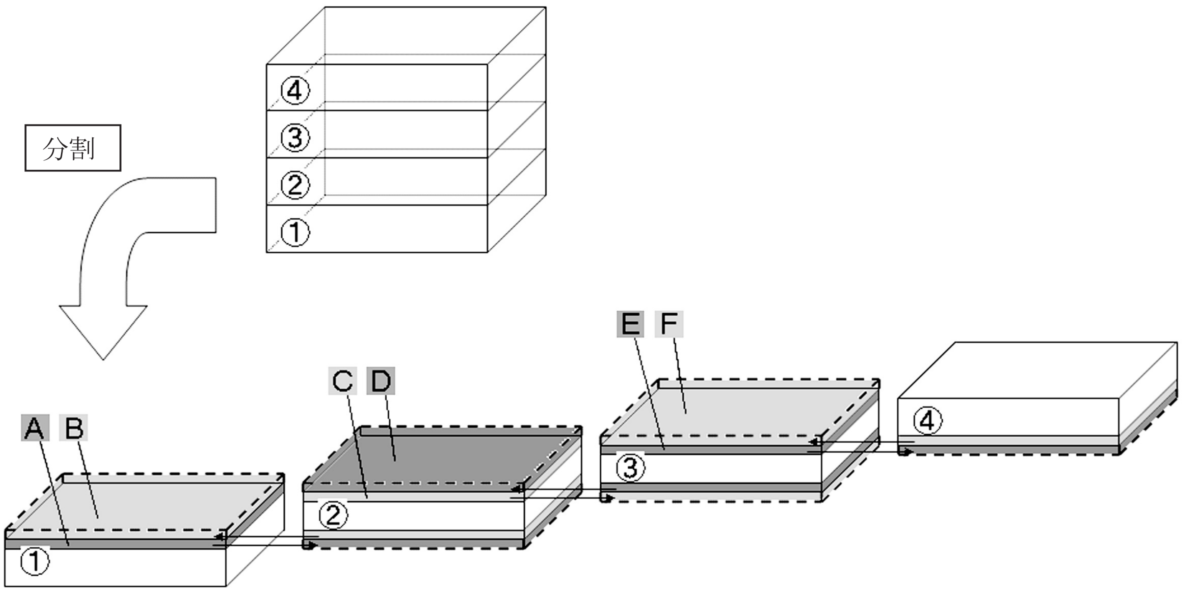


図 6.4-2　領域分割とデータ通信

図 6.4-3 に MPI_SEND，MPI_RECV を用いたコード例を示す．この例では分割する 3 次元目のサイズは実行するプロセス数で割り切れるものとする．21-59 行目では，k 方向に分割したときの各プロセスが必要とするデータ(作業配列，ループ長とループの始点，終点)を作成し，71-80 行目では，分割された領域内でポアソン方程式を解く．94-117 行目で更新された境界面のデータを転送する．

```
 !各プロセスが担当する３次元目の大きさを計算
                       kmax = kmax/nprocs
 !各プロセスにおける配列ｕの３次元目の始点を計算
                         ukst =myrank*kmax
 !境界面のデータを格納する領域を持つ配列 uu,各プロセスの配列ｕの値を保持する作業配列 wu の確保
                       allocate(uu(imax,jmax,0:kmax+1),wu(imax,jmax,kmax))
 !各プロセス毎の始点, 終点の定義
                       if(myrank .eq. 0)then
                         kst=2
                       else
                         kst=1
                       endif
                       if(myrank .eq. nprocs-1)then
                         ked=kmax-1
                       else
                         ked=kmax
                       endif
 :
 +------>              do loop =1,n
 |+----->                do k = kst,ked
 ||+---->                  do j = 2,jmax-1
 |||V--->                    do i = 2,imax-1
 ||||                          wu(i,j,k)=s*(uu(i+1,j,k)+uu(i-1,j,k)
 ||||                 &                     +uu(i,j+1,k)+uu(i,j-1,k)
 ||||                 &                     +uu(i,j,k+1)+uu(i,j,k-1)+a(i,j,k))
 |||V---                     enddo
 ||+----                   enddo
 |+-----                 enddo
 !隣接領域間の転送
 |                     if(myrank.ne.0)then
 |                       dest=myrank-1
 |                       tag=myrank
 |                       call MPI_SEND(wu(1,1,kst),imax*jmax,MPI_DOUBLE_PRECISION,
 |                    &                dest,tag,MPI_COMM_WORLD,IERR)
 |                     endif
 |                     if(myrank.lt.nprocs-1)then
 |                       source=myrank+1
 |                       tag=myrank+1
 |                       call MPI_RECV(uu(1,1,ked+1),imax*jmax,MPI_DOUBLE_PRECISION,
 |                    &                source,tag,MPI_COMM_WORLD,status,IERR)
 |                     endif
 |                     if(myrank.ne.0)then
 |                       source=myrank-1
 |                       tag=myrank-1
 |                       call MPI_RECV(uu(1,1,kst-1),imax*jmax,MPI_DOUBLE_PRECISION,
 |                    &                source,tag,MPI_COMM_WORLD,status,IERR)
 |                     endif
 |                     if(myrank.lt.nprocs-1)then
 |                       dest=myrank+1
 |                       tag=myrank
 |                       call MPI_SEND(wu(1,1,ked),imax*jmax,MPI_DOUBLE_PRECISION,
 |                    &                dest,tag,MPI_COMM_WORLD,IERR)
 |                     endif

 +------               enddo
```

図 6.4-3　ポアソン方程式における差分式（MPI コード）

6.4.2 収束判定のための総和演算

図 6.4-4 は，ヤコビ法の収束判定を行っている箇所を抜き出したものである．49 行目の変数 err の値によって収束判定を行っている．

```
30: +------>        do loop =1,n
 :
41: |                 err=0.0d0
42: |+----->          do k = 2,kmax-1
43: ||+---->            do j = 2,jmax-1
44: |||V--->              do i = 2,imax-1
45: ||||                    err=err+(uu(i,j,k)-u(i,j,k))*(uu(i,j,k)-u(i,j,k))
46: |||V---               enddo
47: ||+----             enddo
48: |+-----           enddo
49: |                 if(err .lt. eps) go to 100
 :
57: +------           enddo
58:            100  continue
```

図 6.4-4　ヤコビ法の収束判定（逐次処理コード）

MPI コードでは図 6.4-4 の 42-48 行目のループは分割されているため，変数 err は部分和となる．各プロセスの部分和を集計し，総和を求める必要がある．6.2 節の総和計算では，一対一通信の MPI_RECV と MPI_SEND のサブルーチンを使用して集計したが，ここでは総和演算を MPI の集団通信のサブルーチンで集計する例を図 6.4-5 に示す．

```
 71: +------>        do loop =1,n
  :
 81: |                 err=0.0d0
 82: |+----->          do k = kst,ked
 83: ||+---->            do j = 2,jmax-1
 84: |||V--->              do i = 2,imax-1
 85: ||||                    err=err+(uu(i,j,k)-wu(i,j,k))
 86: ||||           &                *(uu(i,j,k)-wu(i,j,k))
 87: |||V---               enddo
 88: ||+----             enddo
 89: |+-----           enddo
 90: |                   CALL MPI_ALLREDUCE(err,all_err,1,MPI_DOUBLE_PRECISION,    !①
 91: |              &                          MPI_SUM,MPI_COMM_WORLD,IERR)
 92: |                   if (myrank.eq.0) write(*,*) all_err,loop
 93: |                 if(all_err .lt. eps) go to 100
  :
130: +------         enddo
131:
132:             100 continue
```

図 6.4-5　ヤコビ法の収束判定（MPI コード）

この例で使用した MPI サブルーチンについて以下に示す．

① MPI_ALLREDUCE

MPI の集団通信を行うサブルーチンであり，複数のプロセスからそれぞれのデータを受信し，リダクション演算を行う．集団通信は，同一コミュニケータの全プロセスが対象になる．第五引数に MPI_SUM を与えると，第一引数で与えられた各プロセスのデータの総和を第二引数で受け取る．

6.5　デッドロック

MPI プログラムの実行時には，プログラムの書き方によって各プロセスが送受信待ち状態となり次の処理へ進めなくなる場合がある．この状態をデッドロックと言う．図 6.5-1 はデッドロックが発生する場合のコードである．

```
do i = 0,nprocs-1
  if(myrank==i)then
    k=mod(i+1,nprocs)
    itag=k
    call MPI_SEND(b(1+i*it),it,MPI_INTEGER,k,itag,MPI_COMM_WORLD,ierr)
  endif
enddo

do i = 0,nprocs-1
  if(myrank==i)then
    k=mod(i-1+nprocs,nprocs)
    itag=i
    call MPI_RECV(b(1+i*it),it,MPI_INTEGER,k,itag,MPI_COMM_WORLD,status,ierr)
    endif
enddo
```

図 6.5-1　デッドロックのコード例

MPI_SEND は 6.2 節で述べたようにブロッキングモードで通信を行うため，データの受信が完了しないと次の処理に進めない．つまり，MPI_SEND に対して，受信側プロセスで MPI_RECV が実行されるまで処理は中断される．図 6.5-2 は図 6.5-1 を 2 並列で実行したときのイメージである．この例では，rank 番号 0 は rank 番号 1 が MPI_RECV を呼ぶまで MPI_SEND で待ち続け，rank 番号 1 は rank 番号 0 が MPI_RECV を呼ぶまで MPI_SEND で待ち続けることになり処理が進まなくなる．

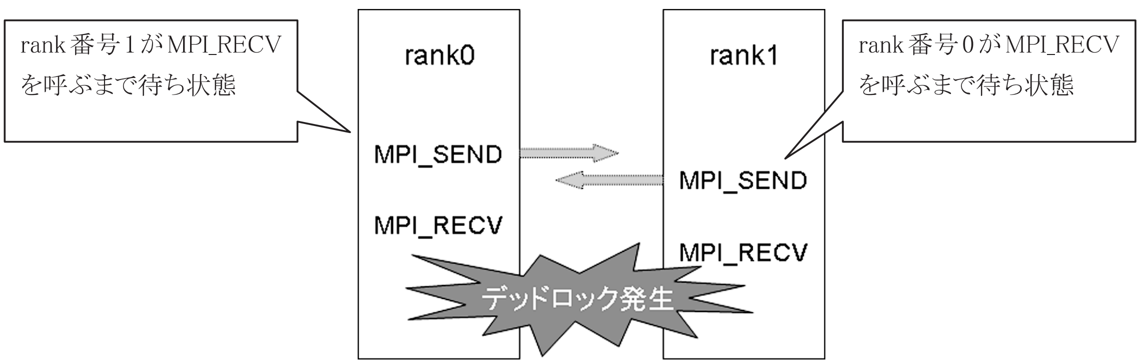


図 6.5-2　デッドロック発生のイメージ

```
do i = 0,nprocs-1
  if(myrank==i)then
    k=mod(i+1,nprocs)
    itag=k
    call MPI_ISEND(b(1+i*it),it,MPI_INTEGER,k,itag,MPI_COMM_WORLD,request,ierr)       !①
  endif
enddo

do i = 0,nprocs-1
  if(myrank==i)then
    k=mod(i-1+nprocs,nprocs)
    itag=i
    call MPI_RECV(b(1+i*it),it,MPI_INTEGER,k,itag,MPI_COMM_WORLD,status,ierr)
  endif
enddo

call MPI_WAIT(request,status,ierr)                                                      !②
```

図 6.5-3 正常動作(デッドロックが発生しない)例

図 6.5-3 にデッドロックを発生させずに実行するコードを示す．図 6.5-1 では MPI_SEND を使用していたのに対し，図 6.5-3 では MPI_ISEND を使用している．MPI_ISEND はノンブロッキングモード（注）で通信を行うため，送信バッファが再利用可能になるのを待たずに次の処理へ進むことができる．ただし，MPI_WAIT を指定し，通信が完了したことを待ち合わせる必要がある．図 6.5-4 は図 6.5-3 を 2 並列で実行したときのイメージである．

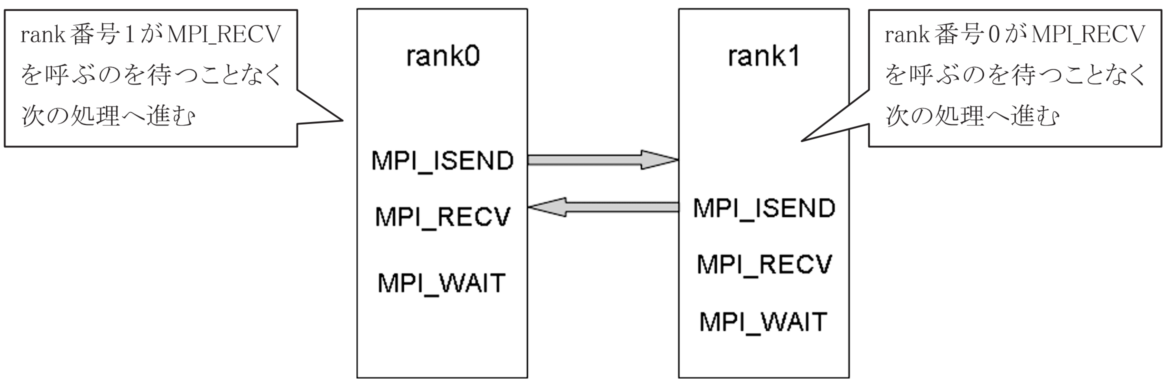


図 6.5-4 正常動作のイメージ

図 6.5-3 で使用した MPI サブルーチンについて以下に示す．

① MPI_ISEND

MPI の通信処理を行うサブルーチンであり，データの送信をノンブロッキング（非同期）モードで行う．このサブルーチンは一対一通信と呼ばれる分類に属し，通信相手が必要になる．受信側にデータを送信するために呼び出される．MPI_WAIT とペアで使用する．第一引数から第六引数は，MPI_SEND と同じである．第七引数（request）は通信識別子を示す．

② MPI_WAIT

ノンブロッキング通信の完了の待ち合わせを行う．一対一ノンブロッキング通信を使用した場合にペアで使用する．第一引数（request）に通信識別子，第二引数（status）にメッセージ情報を示す．

（注）ノンブロッキング（非同期）モード：送信または受信の操作が完了する前に手続きから戻る．ノンブロッキング通信を使用する場合は，送受信操作が完了する前に送信または受信領域の内容を変更してはならない．

6.6 SX-9 の最適化事例

6.6.1 通信方法の見直し

適切な通信サブルーチンを選択することにより，通信時間を短縮することができる．本節では，集団通信である MPI_SCATTERV と MPI_GATHERV を単方向通信である MPI_GET と MPI_PUT に置き換える例を示す．MPI_GET，MPI_PUT は，MPI プロセスが単独で他の MPI プロセスに対してデータの送受信を行うことができる．このため，すべてのプロセスが同時に通信を行う集団通信 MPI_SCATTERV，MPI_GATHERV に比べて実行時間を短縮することができる．

図 6.6-1 は rank 番号 0 のプロセスが持つデータを他の rank 番号のプロセスへ送り，処理を行った後にそれらのデータを rank 番号 0 のプロセスへ送り返すコードである．図 6.6-1 の通信のイメージを図 6.6-2 に示す．

```
   if(myrank.eq.0) then
     call MPI_SCATTERV(data,iscnt,idisp,MPI_REAL8,                  !①
  &                MPI_IN_PLACE,idummy,idummy,0,
  &                MPI_COMM_WORLD,ierr)
   else
     im=nz/nprocs*myrank+1
     call MPI_SCATTERV(data,iscnt,idisp,MPI_REAL8,                  !①
  &                data(1,1,im),isize,MPI_REAL8,0,
  &                MPI_COMM_WORLD,ierr)
   endif
    :
   if(myrank.eq.0) then
     call MPI_GATHERV(MPI_IN_PLACE,idummy,idummy,                   !②
  &                data,ircnt,idisp,MPI_REAL8,0,
  &                MPI_COMM_WORLD,ierr)
   else
     call MPI_GATHERV(data(1,1,im),iscnt,MPI_REAL8,                 !②
  &                data,ircnt,idisp,MPI_REAL8,0,
  &                MPI_COMM_WORLD,ierr)
   Endif
```

図 6.6-1　MPI_SCATTERV と MPI_GATHERV を用いた通信コード

この例で使用した MPI サブルーチンについて以下に示す．

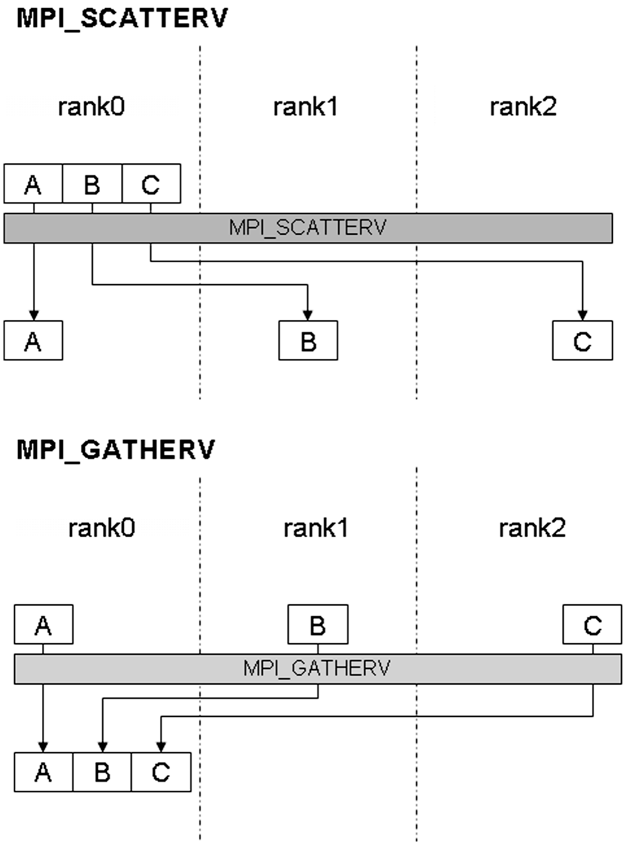
① MPI_SCATTERV

MPI の集団通信を行うサブルーチンであり，同一コミュニケータ内の 1 つのプロセスがデータを送信し，全プロセスが受信を行う．第一引数は送信データの先頭アドレス，第二引数は送信するデータ要素数（プロセス毎），第三引数は送信データの先頭後レスからの変位（プロセス毎），第四引数（MPI_REAL8）は送信するデータの型，第四引数は受信データの先頭アドレス，第五引数（MPI_REAL8）は受信するデータの要素数，第六引数は受信するデータの型，第七引数（0）は送信元プロセスの rank 番号を示す．

② MPI_GATHERV

MPI の集団通信を行うサブルーチンのであり，同一コミュニケータ内のすべてのプロセスがデータを送信し，1 つのプロセスがそれらのデータの受信を行う．第一引数は送信データの先頭アドレス，第二引

数は送信するデータの要素数，第三引数（MPI_REAL8）は送信するデータの型，第四引数は受信データの先頭アドレス，第五引数は受信するデータの要素数（プロセス毎），第六引数は受信データの先頭アドレスからの変位（プロセス毎），第七引数（MPI_REAL8）は受信するデータの型，第七引数（0）は送信先プロセスの rank 番号を示す．



MPI_SCATTERV,MPI_GATHERV は集団通信のため，全プロセスで呼び出される必要がある．

図 6.6-2 MPI_SCATTERV と MPI_GATHERV を用いた通信

次に MPI_SCATTERV，MPI_GATHERV を MPI_GET，MPI_PUT に置き換える．MPI_GET，MPI_PUT を用いた通信イメージを図 6.6-3 に，コードを図 6.6-4 に示す．

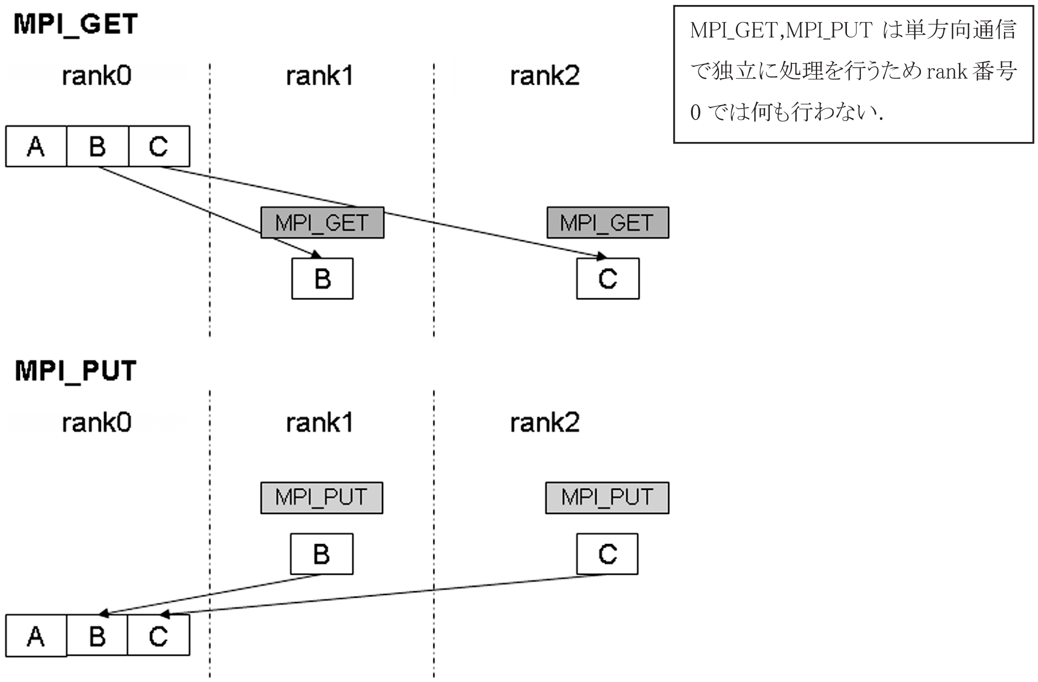


図 6.6-3　MPI_GET と MPI_PUT を用いた通信

```
      integer(kind=MPI_ADDRESS_KIND) idisp                               !①

      isize=8*nx*ny*nz
      call MPI_WIN_CREATE(data,isize,8,MPI_INFO_NULL,                    !②
     &                   MPI_COMM_WORLD,win,ierr)
      call MPI_WIN_FENCE(0,win,ierr)                                     !③

      idisp=nx*ny*nz/nprocs*myrank
      call MPI_WIN_FENCE(0,win,ierr)                                     !③
      if(myrank.ne.0) then
        call MPI_GET(data(1,1,nz/nprocs*myrank+1),nx*ny*nz/nprocs,       !④
     &               MPI_REAL8,0,idisp,nx*ny*nz/nprocs,
     &               MPI_REAL8,win,ierr)
      endif
      call MPI_WIN_FENCE(0,win,ierr)                                     !③
       :
      call MPI_WIN_FENCE(0,win,ierr)                                     !③
      if(myrank.ne.0) then
        call MPI_PUT(data(1,1,nz/nprocs*myrank+1),nx*ny*nz/nprocs,       !⑤
     &               MPI_REAL8,0,idisp,nx*ny*nz/nprocs,
     &               MPI_REAL8,win,ierr)
      endif
      call MPI_WIN_FENCE(0,win,ierr)                                     !③

      call MPI_WIN_FREE(win,ierr)                                        !⑥
```

図 6.6-4　MPI_PUT と MPI_GET を用いた通信コード

この例で使用した MPI の宣言部分とサブルーチンについて以下に示す.

①　integer(kind＝MPI_ADDRESS_KIND)

下記, MPI_GET, MPI_PUT の第五引数で使用する変数は, この型を宣言する.

②　MPI_WIN_CREATE

MPI の単方向通信は，許可したメモリ領域以外，他プロセスからのアクセスを禁止している．この許可したメモリ領域のことをウィンドウと言い，MPI_WIN_CREATE はこのウィンドウの登録を行うサブルーチンである．第一引数はウィンドウに登録するデータの先頭アドレス，第二引数はウィンドウのサイズ（単位は byte），第三引数はウィンドウ内の 1 つのデータのサイズ（単位は byte），第四引数（MPI_INFO_NULL）は info 引数，第六引数は登録されるウィンドウを示す．

③　MPI_WIN_FENCE

単方向通信 MPI の通信処理の同期をとるためのサブルーチンである．ウィンドウをアクセスする可能性のあるすべてのプロセスで呼び出される必要がある．

④　MPI_GET

MPI の単方向通信を行うサブルーチンであり，転送対象プロセスのウィンドウ領域からデータの読込みを行う．第一引数は受信先の先頭アドレス，第二引数は受信するデータ数，第三引数は受信するデータの型，第四引数は送信元の rank 番号，第五引数は登録したウィンドウの先頭からの変位，第六引数は送信するデータ数，第七引数は送信するデータの型を示す．

⑤　MPI_PUT

MPI の単方向通信を行うサブルーチンであり，転送対象プロセスのウィンドウ領域に対してデータの書込みを行う．第一引数は送信するデータの先頭アドレス，第二引数は送信するデータ数，第三引数は送信するデータの型，第四引数は受信先の rank 番号，第五引数は登録したウィンドウの先頭からの変位，第六引数は受信するデータ数，第七引数は受信するデータの型を示す．

⑥　MPI_WIN_FREE

ウィンドウの登録を抹消するためのサブルーチンである．第一引数は登録を抹消するウィンドウを示す．

SX-9 におけるそれぞれの通信時間は図 6.6-5 のとおりである．なお通信に使用する配列 data のサイズは約 8GB，プロセス数は 4 プロセスである．MPI の集団通信から単方向通信へ変更することで 1.27 倍の性能向上が得られる．

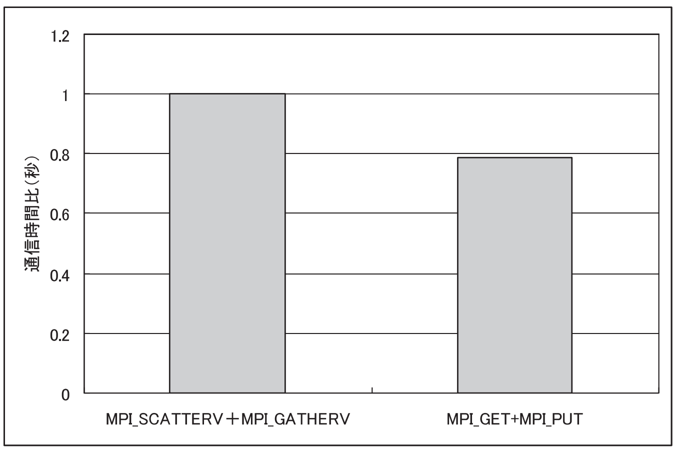


図 6.6-5　通信の違いによる性能比

6.6.2 SX-9 のグローバルメモリ機能

SX-9 では通常のユーザプログラムが使用するメモリ空間はローカルメモリにある. MPI がデータ通信をする際には, 図 6.6-6 に示すように, ローカルメモリ上にある転送データをグローバルメモリの通信バッファへコピーし, MPI システム通信バッファ間で転送が行われる. 受信側で MPI システム通信バッファからローカルメモリのユーザ領域へコピーが完了した時点でデータ通信が終了する. SX-9 では, グローバルメモリ機能を提供しており, 図 6.6-7 に示すようにユーザプログラムが使用するメモリ空間をグローバルメモリ上へ割り付けておくことができる. このことにより, ローカルメモリとグローバルメモリ間のメモリコピーを行うことなく, グローバルメモリ空間にあるユーザ領域のデータを直接転送することができる.

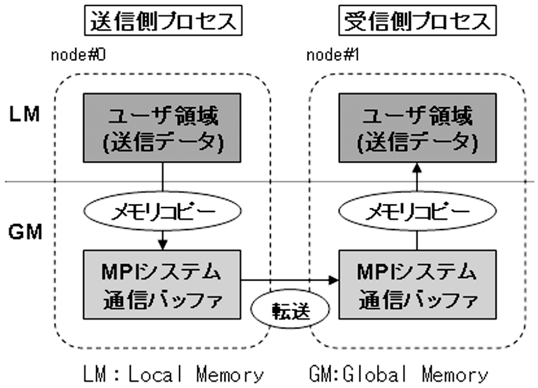


図 6.6-6　通常の MPI データ転送手順

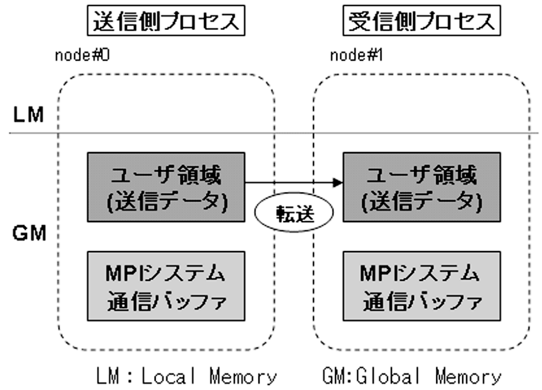


図 6.6-7　グローバルメモリ領域を使用した MPI データ転送手順

グローバルメモリ機能を使用するためには, 通信するデータを格納する配列を動的(allocatable 属性)に確保し, コンパイルオプション「-gmalloc」を指定する. ただし, 配列を通信するごとに動的に確保するような場合, 配列の割当/解放の処理がオーバヘッドとなりグローバルメモリを用いた効果を打ち消す場合がある. このためグローバルメモリの動的な割当/解放は最小限に止めることが必要である. 表 6.6-1 に, SX-9 においてグローバルメモリ機能を使用した場合としない場合のデータ転送時間を示す. これは, 約 100MB のデータを MPI_ALLGATHER でデータ転送した例である. グローバルメモリ機能を使用することで, 2 ノード(32 プロセス)使用時に 2.77 倍, 4 ノード(64 プロセス)使用時に 2.67 倍の性能向上が得られることがわかる.

表 6.6-1　グローバルメモリによる実行時間の比較

	2 ノード(32 プロセス)	4 ノード(64 プロセス)
グローバルメモリ機能なし	162.031sec	327.985sec
グローバルメモリ機能あり	58.441sec	122.542sec

6.7 SX-9 における MPI 性能情報採取方法

本節では SX-9 における MPI 性能情報の採取方法について説明する.

(1) MPIPROGINF

MPI/SX では各 MPI プロセスの性能情報を採取する機能がある. この機能によって MPI プロセス毎, または全 MPI プロセスの情報を集計編集して表示させることができる. 性能情報の表示は, MPI_COMM_WORLD(MPI_UNIVERSE=0)の rank 番号 0 における MPI プロセスの標準エラー出力に対して行われる. 使用するにはプログラム実行時に環境変数 MPIPROGINF を指定する. MPIPROGINF の値とその動作は以下, 表 6.7-1 のとおりである.

表 6.7-1 MPIPROGINF の値

NO	性能情報を出力しない(既定値).
YES	基本情報を集約形式で出力する.
DETAIL	詳細情報を集約形式で出力する.
ALL	基本情報を拡張形式で出力する.
ALL_DETAIL	詳細情報を拡張形式で出力する.

```
MPI Program Information:
========================
Note: It is measured from MPI_Init till MPI_Finalize.
      [U,R] specifies the Universe and the Process Rank in the Universe.

                                        (a)                (b)            (c)
Global Data of 16 processes :          Min [U,R]          Max [U,R]      Average
===========================

Real   Time (sec)           :       1.350 [0,15]       1.577 [0,0]        1.464
User   Time (sec)           :       1.267 [0,13]       1.324 [0,8]        1.308
System Time (sec)           :       0.008 [0,2]        0.012 [0,13]       0.008
Vector Time (sec)           :       0.735 [0,0]        0.766 [0,8]        0.752
Instruction Count           :   192390742 [0,0]    200269828 [0,1]    197702659
Vector Instruction Count    :     4788036 [0,0]      5041807 [0,1]      4979275
Vector Element Count        :    75304093 [0,8]     78786404 [0,1]     77034023
FLOP Count                  :      200402 [0,2]       200411 [0,0]       200403
MOPS                        :     199.609 [0,0]      210.707 [0,15]     206.323
MFLOPS                      :       0.151 [0,8]        0.158 [0,13]       0.153
Average Vector Length       :      14.988 [0,8]       15.854 [0,15]      15.472
Vector Operation Ratio (%)  :      28.193 [0,8]       28.841 [0,15]      28.555
Memory size used (MB)       :    1095.745 [0,0]     1095.745 [0,0]     1095.745
Global Memory size used (MB):      64.000 [0,0]       64.000 [0,0]       64.000
MIPS                        :     146.014 [0,0]      153.770 [0,15]     151.210
Instruction Cache miss (sec):       0.080 [0,3]        0.093 [0,0]        0.085
Operand     Cache miss (sec):       0.012 [0,8]        0.030 [0,0]        0.013
Bank Conflict Time
  CPU Port Conf.       (sec):       0.001 [0,7]        0.001 [0,13]       0.001
  Memory Net. Conf.    (sec):       0.578 [0,0]        0.608 [0,8]        0.595
```

図 6.7-1 MPIPROGINF の出力例(DETAIL 指定時)

表示される項目の意味は以下のとおりである. その他の項目はプログラム実行解析情報(PROGINF)と同じとなる.

(a):すべての MPI プロセスを対象に, 基本情報または詳細情報について集計した最小値と rank 番号

（b）:すべてのMPIプロセスを対象に，基本情報または詳細情報について集計した最大値とrank番号
（c）:すべてのMPIプロセスを対象に，基本情報または詳細情報について集計した平均値

MPIPROGINF の各項目で最大値と最小値の差が大きい場合，処理のインバランス（注）が発生している可能性がある．領域が均等に分割されていてもプロセスごとの処理量が異なるとインバランスが発生する．領域の分割方法の変更などの検討が必要である．

（注）インバランス:各プロセスに割り当てる処理が不均衡な状態を言う．インバランスの場合，プロセス間で同期を取るタイミングで，処理の長いプロセスを待ち合わせする必要があり，先に処理を終えたプロセスに待ち時間が発生する．

(2) MPICOMMINF

MPI/SX では各 MPI プロセスの MPI 通信情報を採取する機能がある．この機能によって全 MPI 手続き実行所要時間，MPI 通信待ち合わせ時間，送受信データ総量，および主要 MPI 手続き呼び出し回数を表示することができる．表示は，MPI_COMM_WORLD（ MPI_UNIVERSE=0 ）の rank 番号 0 における MPI プロセスの標準エラー出力に対して行われる．使用するにはプログラム実行時に環境変数 MPICOMMINF を指定する．MPICOMMINF の値とその動作は以下，表 6.7-2 のとおりである．

表 6.7-2 MPICOMMINF の値

NO	通信情報を出力しない(既定値).
YES	最小値，最大値，および平均値を表示する．
ALL	最小値，最大値，平均値，および各プロセス毎の値を表示する．

```
MPI Communication Information:
------------------------------

Real MPI Idle Time (sec)     :        0.092 [0,8]         0.098 [0,1]         0.095    (a)
User MPI Idle Time (sec)     :        0.089 [0,13]        0.098 [0,1]         0.095    (b)
Total real MPI Time (sec)    :        1.302 [0,15]        1.532 [0,0]         1.419    (c)
Send       count             :            0 [0,0]             0 [0,0]             0
Recv       count             :            0 [0,0]             0 [0,0]             0
Barrier    count             :            0 [0,0]             0 [0,0]             0
Bcast      count             :            0 [0,0]             0 [0,0]             0
Reduce     count             :            0 [0,0]             0 [0,0]             0
Allreduce  count             :            0 [0,0]             0 [0,0]             0
Scan       count             :            0 [0,0]             0 [0,0]             0
Exscan     count             :            0 [0,0]             0 [0,0]             0
Redscat    count             :            0 [0,0]             0 [0,0]             0
Gather     count             :            0 [0,0]             0 [0,0]             0
Gatherv    count             :            0 [0,0]             0 [0,0]             0
Allgather  count             :       100000 [0,0]        100000 [0,0]        100000
Allgatherv count             :            0 [0,0]             0 [0,0]             0    (d)
Scatter    count             :            0 [0,0]             0 [0,0]             0
Scatterv   count             :            0 [0,0]             0 [0,0]             0
Alltoall   count             :            0 [0,0]             0 [0,0]             0
Alltoallv  count             :            0 [0,0]             0 [0,0]             0
Alltoallw  count             :            0 [0,0]             0 [0,0]             0
Number of bytes sent         :            0 [0,0]             0 [0,0]             0
Number of bytes recv         :            0 [0,0]             0 [0,0]             0
Put        count             :            0 [0,0]             0 [0,0]             0
Get        count             :            0 [0,0]             0 [0,0]             0
Accumulate count             :            0 [0,0]             0 [0,0]             0
Number of bytes put          :            0 [0,0]             0 [0,0]             0
Number of bytes got          :            0 [0,0]             0 [0,0]             0
Number of bytes accum        :            0 [0,0]             0 [0,0]             0
```

図 6.7-2 MPICOMMINF の出力例（YES 指定時）

表示される各項目の意味は以下のとおりである．

(a)：プログラム実行に要した実 MPI 通信待ち合わせ時間

(b)：プログラム実行に要した CPU 時間の内，ユーザルーチン実行に要した MPI 通信待ち合わせ時間

(c)：全 MPI サブルーチン実行時間

(d)：主要 MPI サブルーチン呼び出し回数

ただし，本機能を利用する場合には，プロファイル版 MPI ライブラリをリンクする必要がある．プロファイル版 MPI ライブラリは，MPI プログラムのコンパイル/リンク用シェルスクリプト（mpif90）の-mpitrace，-mpiprof，-ftrace いずれかのオプション指定によりリンクされる．

主要 MPI サブルーチン呼び出し回数と次節 6.7(3)の FTRACE とを合わせて，通信性能を分析することができる．詳しくは 6.7(3)を参照のこと．

(3) FTRACE

4.3.1(4)にて，逐次処理プログラムにおける FTRACE について説明した．MPI プログラムで本機能を使用すると，逐次処理プログラムと同様の情報に加え，手続きごと，あるいはユーザ指定リージョンの MPI 通信性能情報を採取することができる．使用するには，測定対象のソースプログラムをオプション -ftrace を指定してコンパイルする．環境変数 F_FTRACE の値とその動作は以下，表 6.7-3 のとおりである．

表 6.7-3 F_FTRACE の値

YES	プログラム終了直後に解析リストを標準エラー出力ファイルに形式2で出力することを指定する(FMT2 と同じ).
FMT0	プログラム終了直後に解析リストを標準エラー出力ファイルに形式0（注1）で出力することを指定する.
FMT1	プログラム終了直後に解析リストを標準エラー出力ファイルに形式1（注2）で出力することを指定する.
FMT2	プログラム終了直後に解析リストを標準エラー出力ファイルに形式2（注3）で出力することを指定する.
NO	プログラム終了直後に解析リストを標準エラー出力ファイルに出力しないことを指定する(既定値).

```
PROC.NAME  FREQUENCY  EXCLUSIVE       AVER.TIME     MOPS   MFLOPS V.OP  AVER.     VECTOR I-CACHE O-CACHE    BANK CONFLICT
                      TIME[sec]( % )    [msec]                    RATIO V.LEN       TIME    MISS    MISS CPU PORT  NETWORK

test               1      1.320(100.0)  1320.361    199.8      0.2 28.75  15.8     0.735   0.094   0.031    0.001    0.578
------------------------------------------------------------------------------------------------------------------------
total              1      1.320(100.0)  1320.361    199.8      0.2 28.75  15.8     0.735   0.094   0.031    0.001    0.578

                (a)           (b)        (c)        (d)        (e)       (f)        (g)       (h)

PROC.NAME      ELAPSED      COMM.TIME  COMM.TIME   IDLE TIME  IDLE TIME  AVER.LEN     COUNT  TOTAL LEN
               TIME[sec]        [sec]  / ELAPSED       [sec]  / ELAPSED    [byte]                [byte]

test               1.615        1.572      0.973       0.093      0.057      68.0    100000       6.5M
---------------------------------------------------------------------------------------------------------
```

図 6.7-3 FTRACE の出力例（YES 指定時）

表示される各項目の意味は以下のとおりである.
(a)～(h)以外の項目は逐次処理プログラムの FTRACE と同じである.

(ａ):経過時間
(ｂ):MPI 通信の送受信に要した経過時間（MPI 手続きの実行に要した時間を含む）
(ｃ):関数の MPI 通信の送受信に要した経過時間（MPI 手続きの実行に要した時間を含む）と, 経過時間に対する比率
(ｄ):MPI 通信を行うまでの待ち時間, および同期待ちに要した経過時間
(ｅ):関数の MPI 通信を行うまでの待ち時間, および同期待ちに要した経過時間と, 経過時間に対する比率
(ｆ):MPI 通信 1 回当たりの平均通信量
(ｇ):MPI 通信回数
(ｈ):MPI 通信の通信量

FTRACE を用いることで, 手続き（サブルーチンや関数）単位の通信性能を知ることができる. しかしながら FTRACE では, どの MPI サブルーチンが何回呼び出されたかを知ることができない. 6.7(2)節の MPICOMMINF で得た情報と組み合わせることで, 使用された MPI サブルーチンと通信性能(通信時間, 通信量など)を分析することが可能になる. また, FTRACE の region 機能を使用することで, 特定の MPI ライブラリの通信性能情報を得ることができる. この場合, MPI_BARRIER（注 4）で同期を取ることで, 演算処理の待ち時間を含まない通信に要する時間のみを採取することができる.

（注 1）形式 0. プログラム単位名を左端に表示する.

（注 2）形式 1. プログラム単位名を右端に表示する.

（注 3）形式 2. プログラム単位名を左端に表示する. ただし, プログラム単位名の長さが 10 文字を超えるものについては, 名前の表示の直後で改行して表示する.

（注 4）MPI_BARRIER：コミュニケータ内の全プロセスで同期を取る場合に使用する. 全プロセスで MPI_BARRIER が呼ばれるまで, どのプロセスも次の処理へ進まない.

[全国共同利用情報基盤センター研究開発論文集 No.33より]

ベクトル型スーパーコンピュータ広域連携基盤の性能評価

†山下　毅　‡村田善智　‡江川隆輔
†小野　敏　†大泉健治　‡小林広明

†東北大学情報部情報基盤課
‡東北大学サイバーサイエンスセンタースーパーコンピューティング研究部

1　はじめに

平成 23 年 3 月 11 日 14 時 46 分 18 秒，宮城県石巻市牡鹿半島の東南約 130km 地点（三陸沖）の深さ約 24km を震源とする東北地方太平洋沖地震が発生した．本報告では，震災後の厳しい電力制限の下で，如何に現状の大規模科学計算システムを通常運用に復旧させたかについての経緯を述べる．また，震災等により限られた計算資源下においても安定した大規模計算サービスを提供するために，計算資源の運用形態の一つとして期待されるハイパフォーマンスコンピューティング (HPC) クラウドの開発状況と性能評価について報告する．

2　東北地方太平洋沖地震発生からスーパーコンピュータシステム復旧まで

3 月 11 日に発生した M9.0 の本震により，東北電力管内では青森県・岩手県・秋田県の全域，山形県・宮城県のほぼ全域と，福島県の一部で停電した．これに伴いセンターの全スーパーコンピュータシステムが停止した．居室の物品が床に散乱するなどはあったが，幸いなことにセンター関係者の人的被害は無く，建物の損傷も軽微なものであった．またスーパーコンピュータシステムも，今後発生するとされていた宮城県沖地震の対策として行っていた耐震施工が功を奏し，筐体の転倒や空調機の落下などは無かった．その後の点検で CPU・メモリ交換で対応可能な故障はあったが，運用に支障がない軽微なものであった．表 1 に 3 月 11 日の地震発生から 5 月 9 日の全スーパーコンピュータシステム復旧までの経緯を示す．

サイバーサイエンスセンターのある宮城県仙台市青葉区では 13 日の 14 時頃までに停電が解消し，運用再開に向けての作業を開始した．15 日には全スーパーコンピュータシステムの起動が確認されたが，震災直後の利用者の需要低下と節電対応のため部分運転を行っていた．その後，利用者のジョブの増加に対応するために 4 月 4 日までにはスーパーコンピュータの 55%，並列コンピュータの 33% が運用再開していたが，7 日夜に発生した余震による瞬停で，再度スーパーコンピュータの停止を余儀なくされた．

4 月 19 日にはスーパーコンピュータの稼働率が未だ 77% ではあるものの，センターの社会貢献の一環として，震災にかかわる復興研究を支援するために，災害・防災，安全技術，環境エネルギーに関する研究分野を対象として，演算負担額の全額または一部を免除する課題の募集を行った．5 月 9 日までに全スーパーコンピュータシステムが通常運用となり，現在に至っている．

表１　災害・復旧状況の経緯

日時	災害・復旧状況など
3 月 11 日 14 時 46 分	地震発生 M9.0 仙台市青葉区で震度 6 弱を観測
14 時 48 分	停電により全スーパーコンピュータシステム停止
13 日 14 時頃	センターの電気復旧
15 日	全スーパーコンピュータシステムの起動を確認 節電対応のためログインサーバおよびファイルサーバのみの運用開始
22 日	飲料用給水タンクが空になる 空調用の給水を井戸水に切り替える
24 日	スーパーコンピュータ 2 ノード，並列コンピュータ 2 ノードの運用開始
28 日	センターの水道復旧 空調用の給水を水道水に切り替える
29 日	スーパーコンピュータ 4 ノードを追加運用開始
4 月　4 日	スーパーコンピュータ 4 ノードを追加運用開始
7 日 23 時 30 分	余震（M 7.4 震度 5 弱）により運用中のスーパーコンピュータ 10 ノードが停止
8 日	スーパーコンピュータ 10 ノード運用開始
15 日	並列コンピュータ 2 ノードを追加運用開始
18 日	スーパーコンピュータ 4 ノードを追加運用開始
19 日	特別復興研究支援課題の募集開始
5 月　9 日	スーパーコンピュータ，並列コンピュータ全ノードの運用開始

災害や電力供給力の問題により本センターのシステムが縮小運転される状況下において，北海道大学，名古屋大学，京都大学，大阪大学，九州大学の各情報基盤センターから可能な範囲での計算資源提供の申し出を頂いた．この支援計画は「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）」構築の準備段階の活動として実施されたものである．この活動の一つとして，本センターの利用者が本センターの利用者 ID をそのまま使用して，大阪大学サイバーメディアセンターの大規模計算システムでのジョブの実行を可能とする，アカウント連携の導入を行った．

3　広域連携高性能基盤の概要

東北大学サイバーサイエンスセンターと大阪大学サイバーメディアセンターはかねてより，計算資源を仮想化することで，ユーザが使用する計算資源数等を意識することなくジョブの投入が可能なサービスの提供，かつユーザジョブのターンアラウンドタイムの短縮と両センターが有する資源の高効率利用を目的に，ベクトルコンピューティングクラウドと呼ばれる広域連携高性能計算基盤の研究開発に取り組んできた．今回の被災を受けて，当センターではノード数を減らし

た部分運転を強いられたため，大規模なジョブクラスのサービスを制限せざるを得なかった．このような視点からみると，これまで取り組んできた広域連携による計算資源のシームレスな利用と，物理的に離れた計算資源の有機的連携による大規模計算サービスの提供は，サービスの向上や，資源の高効率利用，ジョブのターンアラウンドタイムの短縮ばかりでなく，災害時にも利用可能なディペンダブルな計算環境として活用する事が期待できる．

これまでのクラウドコンピューティングでは，ユーザジョブの実行に際して，計算資源のみを仮想化し，ユーザがクラウドの中にある計算資源を意識する事無く利用できる環境を提供するものであった．これは，クラウドを構成する計算資源は大規模データセンター等に集約されている場合が多く，計算資源間を結ぶネットワークの遅延をそれほど考慮する必要がないためである．しかし，高性能計算のアプリケーションを対象とするハイパフォーマンスコンピューティングクラウド (以下：HPC クラウド) では，現存するスーパーコンピュータの計算資源が限られていること，分散する各計算資源の規模を超えた計算の実行が期待されていることから，物理的に分散した計算資源を高速なネットワーク網で結合し有機的に連携させることで，クラウドコンピューティングサービスを提供する必要がある．

このため，HPC クラウドには，ユーザが複数の計算資源をシングルサインオンで，かつ物理的に計算資源を意識することなく利用可能な環境が求められている．このような状況下で，我々は大阪大学サイバーメディアセンターと共同で”ベクトルコンピューティングクラウド”と呼ばれる計算基盤の構築に取り組み，プロトタイプシステムを構築している．このプロトタイプシステムでは，ユーザは Web インターフェースからシングルサインオンでシステムを利用可能で遠隔にあるベクトルコンピュータ双方の計算資源を利用する MPI プログラムの実行を可能にしている．クラウド基盤として，高性能計算をサービスとして提供できることは確認できたものの，東北大学，大阪大学間を結ぶ 800 ｋｍにもおよぶネットワーク性能・帯域が性能に与える影響は大きい．

現在の IP ネットワークでは，ネットワークに点在するルータが経路を決定し，かつその経路は１つに定まっている．このため，ネットワークが有する通信性能 (帯域) を，ネットワークを利用している全てのサービスで共有することになる．また，ネットワーク内部の接続構成は実質，動的に制御不能で信頼性・安全性を保証することができず，ベストエフォートで粗い経路制御しかできない．加えて，複雑な制御機能を有するルータが数多く点在し， IP ネットワークにルータの対故障性や，消費電力の増加が問題視され始めているにも関わらず，これらの管理維持コスト，電力コスト等が非常に高いものとなっている．

これらの既存の数々の問題を解決する新世代ネットワーク基盤技術として，近年 Open Flow Network が注目を集めている．Open Flow Network では既存の IP ネットワークにおいて，ルータが持つ複雑な制御機能をネットワークの外に分離して，制御サーバとして集約することで，ネットワーク全体の制御を集中的に管理することが可能になる．これにより，制御サーバがネットワークを瞬時に構築し，素早くかつ動的に最適化することが可能になるばかりでなく，ネットワークの管理維持コスト，消費電力等を大幅に削減することが期待できる．このような状況下で，現在我々は，計算資源だけではなく，計算資源間を結ぶネットワークも同時に仮想化し，ユーザのアプリケーションが求める計算資源・ネットワークをテーラーメードで提供できる技術として Open Flow Network 技術に着目し，図１に示す HPC クラウドの開発に取り組んでいる．

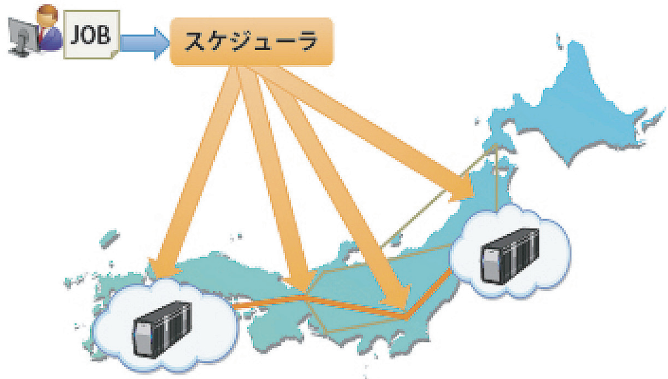


図 1　最適経路選択による HPC クラウドの実現

我々が目指す HPC クラウドでは，実行されるアプリケーションが求めるネットワーク性能を適宜判断し，必要なネットワークをアプリケーションに応じて割り当てる事が可能になる．これらの取り組みとサイバーサイエンスセンタースーパーコンピューティング研究部が開発している実行履歴に基づくジョブスケジューリング機構と連動させることで，高効率かつ，短いターンアラウンドタイムを実現する広域連携計算基盤の構築が可能であると考えている．

次節では，これらの前述の高度化の足がかりとして，HPL（High Performance Linpack）による広域連携計算基盤のネットワーク性能評価と，実アプリケーションを用いた広域連携基盤の基本性能評価について述べる．

4　性能評価と今後の課題

4.1　HPL ベンチマーク

本節では，高性能計算資源の計算性能を評価するために HPL を用いた，広域連携環境の性能評価を示す．評価では，単一拠点の計算資源を用いた環境 (inhouse) と，複数拠点の計算資源を広域連携させた環境 (wide area) において評価を行った．表 2 に示す 6 通りの環境において，inhouse 環境は IXS および 1GbE を用いたネットワーク環境，wide area 環境は SINET3 および JGN2Plus を用いたネットワーク環境である．また MPI 実装は評価に用いた MPI の環境であり，MPI/SX はベンダー提供の MPI 環境，GridMPI はベンダー提供の MPI 環境をラッピング関数を用いて仮想化した環境，GridMPI/IMPI は GridMPI と遠隔 2 拠点間での MPI 通信を実現する IMPI の 2 つを組み合わせた環境である．

図 2 に，それぞれの環境における HPL の性能評価の結果を示す．図 2 より，高速なネットワーク性能を持つ MPI/SX(IXS) と GridMPI(IXS) の 2 つの環境で高い性能を示している．一方，ネットワーク性能が限られる，GridMPI(SINET3) と GridMPI(JGN2Plus) の広域連携環境でも，問題サイズに比例した性能の向上が確認できる．特に，10Gb の転送性能を持つ JGN2Plus を用いた GridMPI (JGN2Plus) の環境では，実行効率が 51.3% を達成しており，ネットワーク性能が限定される広域連携環境においても高い計算性能が得られることが示された．

4.2　実アプリケーションを用いた基本性能評価

実アプリケーションとして理化学研究所・計算科学研究機構 松岡浩教授が開発した 2 次元格子ガス法コードを用い，本計算基盤の評価を行った．本計算基盤を用いて 2 次元格子ガス法コードを実行した時の計算結果を可視化したものを図 3 に示す．図 3 から確認できるように，本計算基盤を用いた場合でも，流体の挙動を正確に計算することができている．本評価では，本計算基盤においてもローカル環境と同様に大規模 MPI プログラムを実行可能であることを確認すると共に，遠隔地にある計算機間のデータ通信量が実行性能に与える影響を明らかにし，通信量を削減する最適化を施すことによって実行効率を向上できることを確認した．

表 2　HPL 評価環境

名称	MPI 実装	ネットワーク
MPI/SX (IXS)	MPI/SX	IXS(64GB/s)
GridMPI (IXS)	GridMPI	IXS(64GB/s)
GridMPI (JF)	GridMPI	1GbE(Jumbo Frame)
GridMPI (NF)	GridMPI	1GbE(Normal Frame)
GridMPI (SINET3)	GridMPI/IMPI	SINET3(1GbE)
GridMPI (JGN2Plus)	GridMPI/IMPI	JGN2Plus(10GbE)

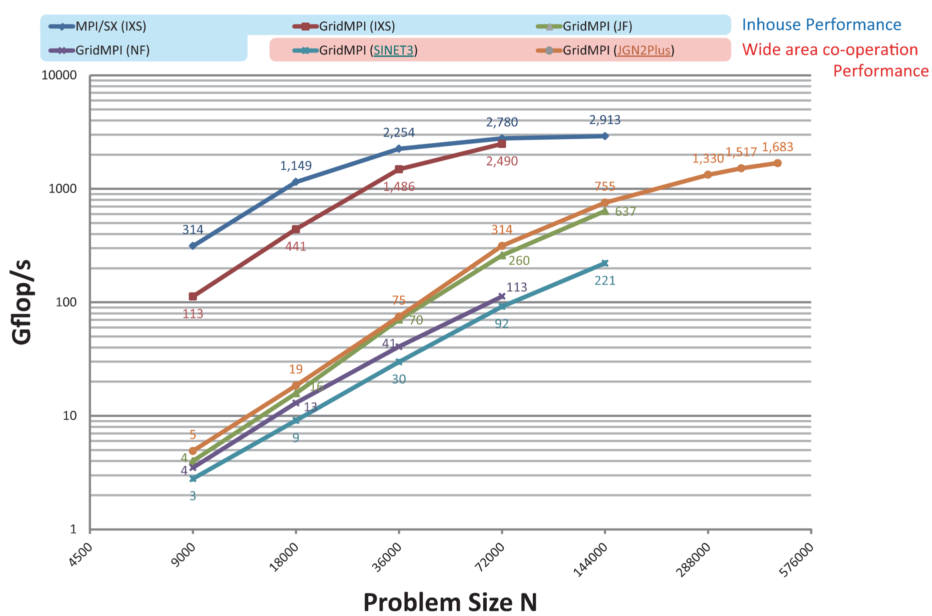


図 2　HPL 評価結果

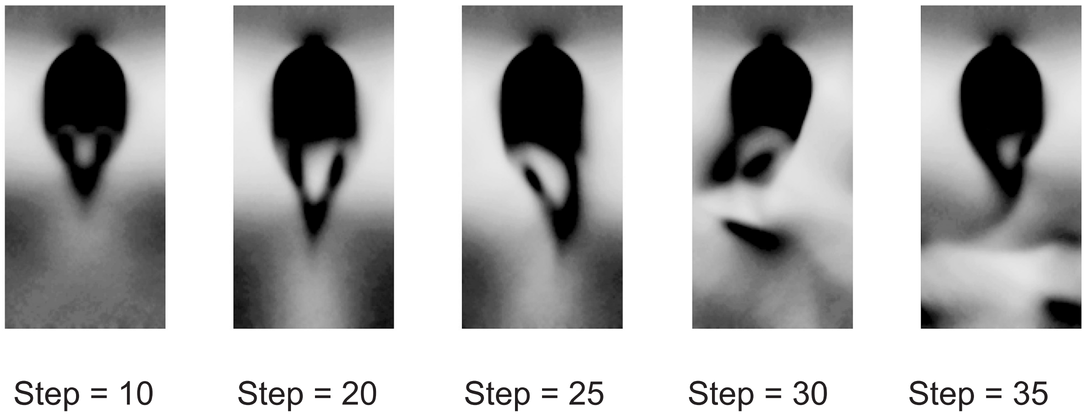


図３　本計算基盤を用いて得られた円柱周りの流体の挙動

5　おわりに

本稿では，震災後から現在に至るまでの当センターの大規模科学計算システムの運用状況と，HPC クラウド構築に向けたスーパーコンピュータ広域連携の性能評価について述べた．評価の結果，800km 離れたスーパーコンピュータを連携させることで，51.3% と高い実行効率を実現できることが明らかになった．今後の課題としては，アプリケーションが必要とする計算性能，ネットワーク性能の明確化と，アプリケーションの要求する計算・ネットワーク資源を仮想化してユーザに提供するシステムの構築が挙げられる．

謝辞

震災直後いち早く支援のお申し出をいただいた各情報基盤センターの関係者各位に深く感謝する．大阪大学との連携実験において支援頂いた大阪大学サイバーメディアセンター 東田学助教，技術的支援を頂いた日本電気株式会社の皆様，2 次元格子ガス法コードを提供して頂いた理化学研究所・計算科学研究機構 松岡浩教授，余震の続く中スーパーコンピュータシステムの復旧に尽力して頂いた NEC フィールディング株式会社の皆様，日本電気株式会社の皆様に深く謝意を表する．また広域連携高性能計算基盤の研究は，国立情報学研究所受託事業「最先端学術情報基盤の構築を推進する事業」の成果である．

[全国共同利用情報基盤センター研究開発論文集 No.33より]

東北大学の学内ネットワークの東日本大震災による被災状況報告と復旧への取り組み

森　倫子

東北大学情報部情報基盤課

1. はじめに

平成 23 年 3 月 11 日に発生した東日本大震災により東北大学の学内ネットワーク TAINS も 2 日程度の停止に追い込まれた。

本稿では、ネットワークを運用する技術職員の立場から体験を交えながら被災状況と復旧について報告するとともに、そこから見えてきた課題についても報告する。

2. 被災状況

2.1. 東北大学の被害状況

平成 23 年 3 月 11 日（金）14 時 46 分、M9.0、最大震度 7 の地震が発生し、その後大津波が発生した。幸い東北大学の主要キャンパスは内陸部に位置するため津波による被害はなかったものの、学生 3 名が学外で津波被災により死亡、建物については危険判定 28 棟、要注意 48 棟の被害を受け、建替え・改修等で約 448 億円 （概算）の損害、研究機器の損害が 352 億円（概算）に及んだ他、停電によって生物系の研究室で多くの貴重な細胞・試料が喪失するなど甚大な被害を受けた。

2.2. 発災から復旧までのサイバーサイエンスセンターの足取り

2.2.1. 当日の状況

その日サイバーサイエンスセンター（以下センター）職員は偶然にも出張中の教職員が一人もいなく、全員キャンパス内で勤務していた。地震が発生し、数分間に及ぶ長い激しい揺れがあり、その間に停電となった。職員と学生はヘルメットを被り全員建物の外の安全な場所へと避難し、怪我人もなく全員無事であった。

すぐにセンター長、副センター長、情報基盤課長、課長補佐、各係長によるセンター災害対策本部を設置し、その他の職員は帰宅させることとした。

ネットワーク系の職員で建物内の機器室に立ち入り簡単な見回りを行い、ラックの倒壊や機器の損傷がなさそうなこと、ネットワーク機器とサーバが UPS で稼働していることを確認した。並行してしばらく情報収集等を行っていたが、発災から 3 時間弱程度で最後の UPS のバッテリがなくなり、すべての機器が停止した。

センター災害対策本部で協議し、復電するまではどうしようもないため、翌日の集合時間を決め、帰宅不能の職員を残して全員帰宅することとした。

2.2.2. センターおよび学内ネットワークの被災状況

学内ネットワークの詳細な停止状況は次の通りである。

地震発生の約2分後	センター停電、UPS給電（4系統）開始
停電の10分後	TAINSメール、リモートアクセスサーバ等自動シャットダウン
〃　28分後	SINET停止
〃　2時間30分後	TOPIC（地域ネットワーク）停止
〃　2時間50分後	TAINS幹線停止

2.2.3. 復旧進捗

翌3月12日、出勤した職員で被害状況の確認を行った。建物については、外壁の一部が損傷し、建物内壁にクラックが多数散見されたが、使用に問題はないようであった。また、ラックの倒壊はなく、サーバ、パソコン等情報機器及びネットワーク機器の損傷もないことを確認した。

その後東北大学災害対策本部へ前述の状況報告を行い、電気が復旧すれば即運用可能であることを伝え、電気の復旧を待った。なお、事務系のサーバを設置している情報推進課（大学本部棟）も建物やラック、機器の損傷はなかったが、サーバ室のフリーアクセスフロア床下が損傷していた。

3月13日、仙台市全域が停電のため情報が入りにくい状況であったが、仙台市中心部から徐々に電気が復旧しているようであった。13日午前、大学本部から復電可能な状態になったのでセンター職員の立会いのもと復電作業を行うとの連絡があり、センター職員立ち会いのもと電気が復旧した。電気の復旧後順次ネットワーク機器・サーバを立ち上げ、動作確認を行い、学内ネットワークTAINS、地域ネットワークTOPICおよびSINETは無事復旧した。

サービスの復旧の詳細は以下に示す通りである。

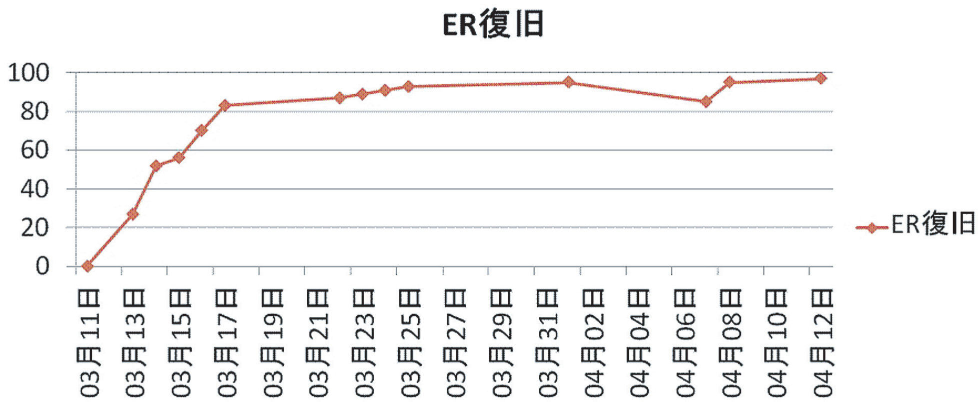
- TAINSネットワーク
  - TAINS基幹ネットワーク　→　復旧：3/13　14:10
  - TAINSメールサービス（@m.tohoku）　→　復旧：3/13　17:28
  - 大学ホームページ　→　復旧：3/13　14:10（通電）3/14（緊急連絡用開設）
- スパコン　→　復旧：3/15（通電）、5/9（通常運用）
- 業務システム
  - 事務用ネットワーク　→　復旧：3/14
  - 事務用メールシステム（@bureau）　→　復旧：3/13
  - 統合電子認証システム　→　復旧：3/13
  - 教職員グループウェア　→　復旧：3/13
  - 東北大学ポータルシステム　→　復旧：4/12（通電）、5/11（通常運用）
  - 各種業務システム（財務会計、教務情報、人事給与システム等）
    →　復旧：3/13（通電）、3/14（通常運用）

2.3. 部局ネットワーク

2.3.1. 部局ネットワークの復旧状況

TAINS は基幹ルータと部局に設置したエッジルータ（以下 ER）にて構成されている。部局ネットワークの復旧状況は次の通りである。

3/13（日）　基幹ネットワークを復旧
3/13（日）　ER 27%（13/48）復旧
4/ 1（金）　ER 95%（46/48）復旧
4/ 7（木）　余震発生　停電のため ER 5 台が一時停止、すぐに復旧
4/12（火）　ER 97%（47/48）復旧



2.3.2. 部局ネットワークへの緊急支援

被災した部局に対し、以下の緊急支援を行った。

- ハウジングサービス
  - 部局サーバをサイバーサイエンスセンター内に移設して仮復旧
- 部局ドメイン代行サービス
  - 代行メール転送の機能を提供
  - ウェブサイト公開の機能を提供
  - DNS の機能を提供
- ネットワークインフラ
  - ネットワーク障害復旧作業
  - 無線 LAN システム と有線ネットワークを緊急増設

3. 災害対策の効果と課題

3.1. 事前の地震対策の効果

事前に行っていた地震対策で効果があったものについて述べる。

まず第一にフリーアクセスフロアおよびラックの耐震補強をしていたことが非常に功を奏した。また、一般に建物の上の階になるほど被害が大きいが、本センターでは機器室を低層階に設置し

たことも効果があった。ラックの固定をせず、機器室を上の階に設置したある部局では、機器および電源が甚大な被害を受けたため復旧に大変な時間を要したようである。

居室の対策としては、本棚の耐震等対策や、棚からの落下防止を行っていたことで、怪我人もなく、居室の被害も少なく、電気復旧後即通常業務が可能となった。

震災直後から仙台市内では食料・物品の入手が大変困難な状態となった。センターでは非常食料、懐中電灯、乾電池等の備蓄が潤沢であったので大変役に立った。また，センターの屋上等に設置してある受水槽のおかげで，仙台市水の復旧にかなりの時間を要した状況下であっても，最低限の水の確保が可能であった。

3.2.　実際経験してわかった事

事前に対策をしていたつもりでも不足していたこともあれば、想像もできなかったこと、経験してみて初めてわかったこともあった。

まず困ったのは、職員間の連絡が困難なことであった。緊急連絡網は整備していたものの、ほとんどの携帯電話・自宅電話が不通となり、地震発生の翌々日頃まで多くの職員と連絡が取れなかった。また、公共の交通機関の停止、ガソリンの入手困難等により長い間職員の出勤に支障が生じた。

情報サービス運用の非常時体制（BCP）についても不十分なところがあった。情報システムをほぼすべて学内に配置していたため、停電になったことですべてのシステムが停止した。情報伝達用のウェブサーバ、メールサービスも利用できず、DNS サーバの停止により tohoku.ac.jp が引けない状況になってしまった。（ただし，震災直後のあの状況下では，たとえ tohoku.ac.jp が引けたとしても，あまり大勢に影響はなかったのではないかという見方もある。）また、センターで運用しているものではないが，安否確認システムも停止してしまった。計画的な停電時には発電機を用意しているが，緊急時に対する非常用電源の平時からの準備はなかった。運用体制については、例えば、東北大学のトップページが２日目にサーバが再開したものの広報担当の職員がいなかったため更新開始が３日目になる等、事前に運用体制を検討しておくべきだったことがわかった。また、非常時の部局との連絡網を整備していなかったため、ネットワークの状況を知らせたり問い合わせを受けることもできなかったことも反省点としてあげられる。（だが、個人のプライベートな携帯電話の番号を集めてもいいものか検討が必要かもしれない。）

4.　他大学との連携

東北地区では、東北学術研究インターネットコミュニティ TOPIC が今も活発に活動を続けている。6 月末に開催した TOPIC 総会時の講演会および 9 月末開催のネットワーク担当職員研修会において、参加大学の情報・経験・ノウハウを共有できたことは今後の在り方の検討に向けて大きな糧となった。

5.　学内情報基盤の今後の課題

前述の問題点から学内情報基盤の今後の課題が見えてきた。

設備の防災対策面では、ラックの耐震が一部未施工であったのですべてのラックの耐震補強を実施したい。電源確保のためには非常用電源の整備が理想であるが、費用や施設のスペース等から早急な整備は難しいかもしれない。

大学の情報システム全体を考えると、現在部局でバラバラに運用しているサーバやセンターで預っている部局サーバをホスティングに集約することで、システム最適化と防災対策との両立または相乗効果が期待できるものと思われる。ただし、UPS による稼働時間の延長のためには収容機器はごく限られたものとなるため検討の必要がある。

また、システムだけでなく非常時の組織の運用体制の整備も重要である。今後、業務継続計画（BCP）の策定においてインフラ整備の情報基盤に関る部分について、情報シナジー機構が中心となり考えていく必要があると思う。即効的な対策としては、運用継続のためのサーバの学外設置、特に DNS サーバについては早期に実現したい。

最後に、学内・学外への避難研究室のネットワーク利用には学術認証連携が必須であり、学内への普及および TOPIC を通じて東北地域の大学への普及に務めたい。

6.　まとめ

思えば数年前から宮城県沖地震は必ずくるという警告が発せられ続けてきた。情報基盤の話からは少し逸れるが、国や宮城県・仙台市が建物の耐震補強や電気・水道・ガス・道路他のインフラを我々の知らず知らずのうちに着々と進めてくれていたこと、そして東北大学やセンターも対策を怠らなかったことが被害を最小限に留め、早期の復旧に繋ったことと思う。また、復旧にあたっては数多くの遠方の地方自治体の職員や他大学の（特に施設系の）職員のみなさまが年度末の忙しいところお手伝いに来てくださったことに改めて感謝申し上げたい。

そして、耐震補強や災害用備蓄等に予算を割り当て、実施してくれた会計担当職員や、仕様書を策定したネットワーク研究部の教員、ネットワーク WG の委員、実施してくれた保守運用業者の方々、ネットワーク係の職員を誇りに思うと同時にこれからも直面する課題に力を合わせて取り組んでいきたい。

以上、当時のことを思い出しながら、被災状況や復旧の様子などを率直に綴ってみた。拙いところもあったかと思うが、被災した方々に心よりお見舞いを申し上げつつ、本稿が大学の情報基盤を支える職員の方々に少しでもお役に立てれば幸いである。

［全国共同利用情報基盤センター研究開発論文集 No.33より］

キャンパス無線 eduroam 導入のメリットと国内外の動向

後藤 英昭　　曽根 秀昭

東北大学サイバーサイエンスセンター

1　はじめに

国際的な学術無線 LAN ローミング基盤である eduroam (エデュローム)[1]は，2006 年に日本が参加して以来，国内でも次第に参加機関が増え，2011 年 9 月時点で 24 機関が利用している[2]. この数は前年同月比で約 7 割増であり，順調な増加傾向である. また，情報教育研究集会[3]を始め，各種研究会や講習会，学会などにおける広報により，eduroam の知名度も高くなってきており，参加に向けて準備中あるいは検討中の機関も少なくない. 国際的には，北米(カナダ，アメリカ合衆国)における普及ペースが速いこともあって，国際的な無線 LAN ローミング基盤としての地位は確実なものとなった.

日本国内には 1,200 を超える高等教育機関があり，普及率では 2%とまだまだ低い. eduroam の導入を検討している幾つかの機関からは，学内において eduroam 導入のメリットを理解してもらうのが難しいといった声も聞かれる. 広報において国際性を前面に出していたことが影響してか，「海外との交流が少ないのでメリットがない」と思われているケースも見られた. しかし，eduroam の導入は，国内や学内といった範囲でも様々なメリットがあり，さらには，今後来るであろうキャンパスユビキタス時代に適した学術ネットワークアクセス環境を提供し，教育・研究環境の改善や開拓が期待される.

本稿では，各機関における導入説明の一助となるように，eduroam 導入の様々なメリットについてまとめる.

2　キャンパス無線 eduroam 導入のメリット

eduroam は，以下に示すような特徴を有する．これらを踏まえた上で，eduroam 導入の様々なメリットについて述べる.

(1) 商用無線 LAN でも利用されている，標準的な IEEE802.1X 方式を採用し，安全なユーザ認証を実現．(技術的には何ら特殊なところは無い．)

(2) Windows PC や Mac はもちろん，iPhone や Android などのスマートフォン，タブレット，Linux など，幅広い端末および環境で利用可能.

(3) 学術系の無線 LAN システムとして，国際的なデファクトスタンダード.

(4) エンドユーザに対して課金しない．ローミングしている機関どうしや，国の間でも，利用料金のバランス(支払い)などは行わず，サービスの無償相互提供が原則.

(5) 全世界どこでも，ほぼ同じ接続手順．端末の多くは，一度の初期設定のみで良く，サービスエリア内に入ると自動的にネットワークに接続される.

(6) 不正利用時のインシデント対応(端末の追跡・特定など)が可能.

(7) 認証 VLAN の機能によって，学内 LAN とゲストネットワークの安全な分離が可能.
(来訪者の端末をゲストネットワークに収容．)

2.1　キャンパスの仮想的拡大

現在，無線 LAN システムをキャンパスに導入済みの教育・研究機関は少なくない. 大学等では，講師がプレゼンテーション用の PC を学内 LAN に接続したり，学生が持ち込みの PC を使って演

習や自習を行なったりするなどの利用形態があり，キャンパスネットワークはこのような新しい授業方法を支援していく必要がある．これらのことは従来の無線 LAN システムでも実現可能に見えるが，実際は利便性に問題を抱えている機関も少なくない．例えば，大きな大学では部局ごとに無線 LAN システムを整備することがあるが，全学で相互利用できるようなローミング対応のシステムをわざわざ設計・構築しない限り，無線 LAN システムは部局ごとに閉じたものになってしまう．その結果，教員や学生にとっては他学部での講義・演習に支障があり，教職員にとっては移動先の会議室において不便を強いられる．全学で共通の無線 LAN システムを構築すれば，このような問題は解決できるだろうが，機関内に閉じていることに変わりはない．

近年では，教職員・学生の大学間の移動にも対応できるシステムが求められている．単位互換制度により他校の講義や演習に出席する学生にとっては，現地でのネットワーク接続が必要になる．大学間の協定が進んで，各大学に閉じたシステムでは教育・研究を十分にサポートできなくなってくることは想像に難くない．

eduroam は国際標準である IEEE802.1X に基づいたネットワークローミングシステムであり，学内の部局間での無線 LAN ローミングにも応用できるうえに，国立情報学研究所にあるサーバに登録するだけで，他機関との認証連携が有効になり，無線 LAN の相互利用環境を実現できる．

教員や学生が国際会議などで海外渡航した際は，インターネットに接続するのが難しいことが多い．もし現地の教育・研究機関で自由にネットワーク接続が可能ならば，利用者にとって非常に便利なことはもちろん，利用コストの面でもメリットが非常に大きい．例えばヨーロッパにおいて，公衆無線 LAN が利用できたとしても，一日あたりの利用料金は 20〜30 ユーロといった相場であり，相当な負担となる．日本のプロバイダ(ISP)のアカウントを利用したローミング利用も考えられるが，料金的には同様に高額である．eduroam を利用すれば，現地で端末の電源を入れるだけで(多くの場合は)自動的にネットワークに接続され，無料で高速なインターネット利用が可能である．

以上のように，eduroam では部局間から大学間，さらに国をまたいでも，シームレスでスケーラブルな無線 LAN 相互利用環境が実現できる．言い換えれば，従来は学内に閉じていたキャンパスネットワークが，仮想的に拡大するとも言える．電子化された教材はもちろんのこと，国内外を問わずに電子ジャーナルや学術クラウドなどへのアクセスが可能となることから，ネットワークへの依存度が高まった現代の教育・研究環境では，「キャンパスの仮想的拡大」と見ることもできるだろう(図 1).

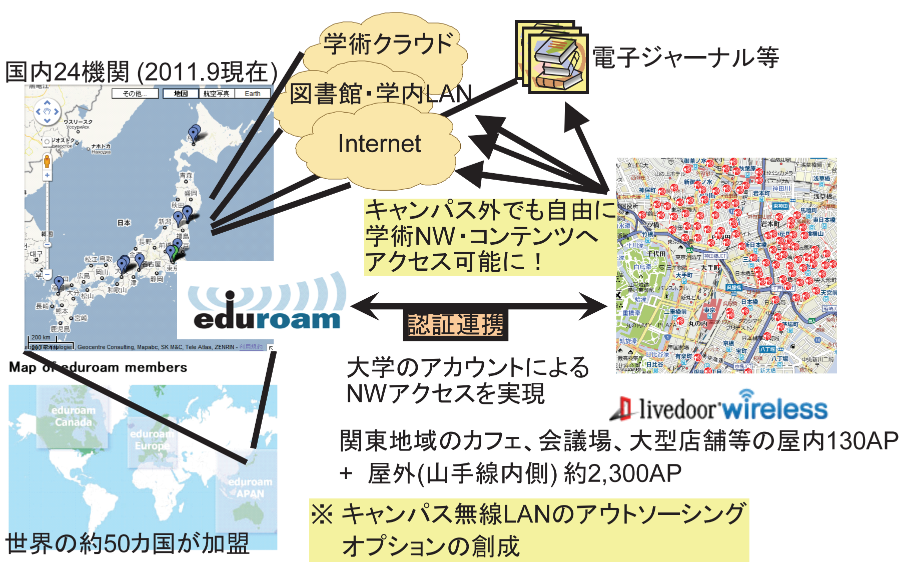


図 1. キャンパスの仮想的拡大

さらに近年では，公衆無線 LAN の提供業者や自治体と協力して，教育研究機関以外でも eduroam 対応のアクセスポイント(AP)が利用できるような環境の構築が進められている．例えば欧州では，

大学の近所のパブや，街なかのカフェ等においても，eduroam 対応の AP を設置する動きがある．ルクセンブルクでは，地方自治体が運営する市街地無線 LAN サービス HotCity の AP で eduroam が利用可能となっている．日本においては，今のところ関東地区限定ではあるが，ライブドア社が市街地やカフェ，大型店舗で運用している公衆無線 LAN サービスにおいて，約 2,500 基の AP で eduroam が利用可能となっている[4]．このように，キャンパス無線 LAN システムに eduroam を採用するだけで，他に特別な契約もなく，キャンパスを大幅に拡大できることになる．

2.2　安全なユーザ認証

現在，キャンパス無線 LAN でも商用の公衆無線 LAN サービスでも，ウェブブラウザの画面に ID・パスワードを入力する，いわゆる「ウェブ認証」と呼ばれる方式が広く用いられている．しかし，付録 A に示すように，ウェブ認証は偽 AP 問題に対処できず，ID・パスワードの盗難に関して大変危険であることが知られている．もし，教職員の業務や学生の履修登録などの重要な用途で使われる ID・パスワードと同一のものがウェブ認証でも用いられているならば，それはセキュリティ上の大変な脅威である．

eduroam で利用されている 1X 認証では，偽 AP 問題への対策が可能であり，端末の初期設定で正しい認証方式(MS-CHAPv2 など)が選択されていれば，ID・パスワード盗難のリスクは非常に低くなる．また，本章の冒頭でも挙げたように，一度設定を行なっておけば，サービスエリア内に入ると端末は自動的にネットワークに接続される．1X 認証では，正しいかどうかわからない認証画面の指示に利用者が従うような危険性はない．また，eduroam の端末設定は一部の値を除いて世界共通なので，周囲の人の援助を受けやすいという利点もある．

日本の eduroam では，「学術認証フェデレーション(学認)」を利用してエンドユーザが eduroam 用のアカウントを取得できる「仮名アカウント発行システム」が提供されている．もし機関が学認に参加し，このシステムを利用すれば，eduroam 用のユーザデータベースを別途構築しなくて済む．また，配布されるのは一時アカウントであるので，万一の盗難の際にも被害範囲を小さく抑えることができる．

2.3　安全なネットワーク構成

キャンパス無線 LAN では，学内利用者の端末は学内 LAN に接続させ，高度なネットワークサービスを提供し，一方で来訪者の端末はゲスト用のネットワークに収容し，学内 LAN への干渉を避けたいというニーズがある．ウェブ認証方式で，このような利用者 ID に基づくネットワークの分離を行うことも不可能ではないが，セッションハイジャックにも耐性のある仕組みを作り込むことは難しい．eduroam では，1X 認証において認証 VLAN を利用することで，安全なネットワーク分離が可能である．

ローミング環境では，大学が契約している電子ジャーナルなどに来訪者がアクセスできるようでは，ライセンス違反となる．認証 VLAN を利用することで，このような問題にも対処が可能である．

2.4　導入と運用の簡略化，低コスト化

従来のキャンパス無線 LAN システム構築では，大学とシステムインテグレータ(SIer)が，認証方式の選定からシステム設計を行うケースも少なくなかった．1X 認証がまだあまり普及しておらず，Windows のサポートも弱く，実績が少なかったことなども影響していたと思われる．

eduroam は世界標準の 1X 認証方式を採用しており，近年では様々なオペレーティングシステムで 1X のサポートが強化されたこともあり，無線 LAN の安全な認証方式としてはごく標準的なものになっている．このため，キャンパス無線 LAN に eduroam を採用する場合，極端に言えば調達における文書に「eduroam に対応すること」の一文を入れることで，仕様策定の大幅な簡略化が期待できる．AP まわりもごくありふれた機器構成・ネットワーク構成になるので，経験のある SIer やプロバイダ(ISP)では，設計が容易になり，初期導入コストの低減が期待される．

従来はスモールオフィス向けの安価なスタンドアロン型の AP を並べることも多かったが，キャンパスに数十～数百といった数の AP を設置すると，その管理・運用コストが意外に高くなる．コントローラ型の AP システムは，初期導入費用が割高になるものの，死活監視や設定変更など

の手間が大幅に減るので，少ない人数で運用でき，長期運用ではトータルコストの低減につながると考えられる．また，ハードウェアの高い安定性も期待できる．

eduroam は，ネットワーク管理者の運用負担の観点でもメリットがある．大学で学会などが開催される場合，従来は既設 AP の設定変更やゲスト用 AP の一時的設置などの作業が必要だった．独自の認証方式を採用している場合は，接続のサポートも必要である．eduroam を導入すれば，eduroam のアカウントを持っている来訪者はそのまま無線 LAN が利用でき，アカウントのない人にはゲスト ID を配るだけで済む．小規模な会議で従来はゲスト用 AP の設置を諦めていたケースでも，容易に無線 LAN サービスを提供できるようになると思われる．

2.5　海外の研究機関との交流促進・支援

2.1 で述べたように，日本の教員や研究者が海外の教育・研究機関を訪れる際に，eduroam が利用できることは非常に大きなメリットである．当然ながら，海外から見れば，日本でも各所で eduroam が利用できることが期待されている．日本の公衆無線 LAN サービスが欧米に比べて安いといっても，契約が必要になるか，割高な一日利用権を購入する手間がかかる．さらに，多くの大学等のキャンパスには公衆無線 LAN が導入されていないため，商用サービスが利用できないという問題がある．

eduroam を導入することによって，海外の研究者に無償でネットワーク利用環境を提供することができ，会議も誘致しやすくなると考えられる．小規模で頻繁な打ち合わせでは特に，ゲスト ID の取得なしに自由にネットワークが利用できることの恩恵は大きい．eduroam サービスの提供は，相互に恩恵を与えあうという観点では義務であるが，海外機関にとっての「おもてなし」の意思の表れでもある．国際的な教育・研究環境をうたう機関であればなおさら，交流促進・支援の観点で eduroam の存在意義は大きいものとなるはずである．

3　むすび

本稿では，eduroam 導入が機関にもたらす様々なメリットについてまとめた．各機関における eduroam 導入説明の一助となれば幸いである．

参考文献

[1] L. Florio, K. Wierenga, ``Eduroam, providing mobility for roaming users,” EUNIS 2005, June 2005.

[2] eduroam JP ウェブサイト: `http://www.eduroam.jp/`

[3] 後藤英昭, 曽根秀昭, ``大学間無線 LAN ローミング基盤 eduroam の動向と容易な導入方法, ” 平成 22 年度 情報教育研究集会講演論文集, 2010.

[4] プレスリリース「ライブドアと国立情報学研究所(NII)国際学術無線 LAN ローミング基盤 eduroam の共同実証実験を livedoor Wireless アクセスポイントにて開始」
`http://corp.livedoor.com/press/2010/0308376`

付録A. ウェブ認証の危険性について

無線 LAN では, 利用者が期待する正規のアクセスポイント(AP)に常に端末が接続されるとは限らない. 悪意を持った者が ID とパスワードを盗み出すなどの目的で偽の AP を立てることも考えられる. このような「偽アクセスポイント(AP)問題」への対策がない認証方式では, 攻撃者に ID・パスワードなどを盗られることがある.

現在, キャンパス無線 LAN でも商用の公衆無線 LAN サービスでも, 国内外を問わず, 端末をアクセスポイントに接続するとウェブブラウザの画面に ID・パスワード入力画面が強制的に表示される, いわゆる「ウェブ認証」と呼ばれる方式が広く用いられている. しかし, ウェブ認証で偽 AP 問題に対処することは困

難であり，ウェブ認証は大変危険であることが知られている．

ログイン画面で HTTP（HyperText Transfer Protocol）が使われている場合はもちろんであるが，例え HTTPS（HyperText Transfer Protocol over Secure Socket Layer）が使われていても，AP の背後にある認証サーバが正規のものである保障はなく，入力した ID・パスワードは暗号化無し(平文)で攻撃者に伝わってしまう．ウェブブラウザが警告を出さないような証明書は，条件さえ満たせば誰でも入手できることに注意が必要である．しかも，ユーザ認証がまだ済んでいない状態なので，端末はインターネットにアクセスできず，信頼できる機関を通してサーバ証明書の検証を行うこともできない．事前にウェブブラウザに機関独自のサーバ証明書を仕込んでおく手法も考えられるが，証明書の配布と導入サポートの手間がかかるという問題がある上に，ID 発行者と AP の管理者が異なるローミング環境では利用できない．

さらには，例えサーバ証明書を用いていたとしても，端末接続のたびに手動で入力操作を行う場合は，利用者が HTTPS と HTTP を意識せずに，攻撃者が用意した HTTP のページにうっかり ID・パスワードを入力してしまう危険性が排除できない．この時，当然ながらブラウザは一切警告を出さない．また，利用者の判断に頼るシステムである以上，攻撃者は偽の利用案内をブラウザに表示して，利用者を欺き，より重要な情報を聞き出すことも可能だろう．商用サービスがウェブ認証を提供し続ける理由として，歴史的経緯とサービスの継続性，利用者にとって直観的で簡便な利用方法，低機能デバイスのサポートなどが考えられるが，ウェブブラウザを開くという簡単な操作で広告効果が得られる点も無視できないと思われる．一方で，1X 認証を利用した高セキュリティなプランを提供する公衆無線 LAN の業者も増えてきている．

［報　告］

平成23年度全国共同利用情報基盤センター顕彰

平成 23 年度顕彰授賞式が行われました。

全国共同利用情報基盤センター及び国立情報学研究所の運用に非常に顕著な貢献があった方々に授与される平成 23 年度全国共同利用情報基盤センター顕彰功績賞候補者として、先に本センターから推薦していた情報部情報基盤課高橋洋一技術専門職員の授賞式が行われました。

授賞式は、昨年 11 月 21 日（月）午前 10 時から東京大学の当番により開催された、全国共同利用情報基盤センター研究開発連合発表講演会の冒頭で行われました。授賞式では、第 17 回全国共同利用情報基盤センター長会議議長である国立情報学研究所長名の表彰状と副賞（楯）が、当番校の東京大学情報基盤センターの石川裕センター長から、高橋洋一技術専門職員を含め、4 名の受賞者の方々に手渡されました。

今年度顕彰を受けられた高橋技術専門職員は、永年に亘り東北大学サイバーサイエンスセンター（旧情報シナジーセンターと大型計算機センターを含む。）において、全国共同利用の大規模科学計算システムの運用への貢献が極めて大きいことが表彰の対象となったものです。

高橋技術専門職員は、昭和 48 年大型計算機センターに採用となり、データベース、ネットワーク等の整備や TSS（Time Sharing System）の普及において、中心的な活動でセンター初期の計算サービス基盤を築かれ、後年もその見識を持ってスーパーコンピュータ、並列コンピュータシステムの構築に大いに貢献されました。また、計算機システム構築以外にも、運用監視システムによる空調機管理体制を整備したことにより、システムの安定性が飛躍的に高まりました。この設計は現在も受け継がれ活用されています。

このように、38 年の永きに亘り全国共同利用としての大規模科学計算システムの環境整備と運用管理に対して多大な貢献をされたことが今回の受賞に至った理由となっています。

『受賞者のコメント』

東北大学情報部情報基盤課　　高橋　洋一

このたび名誉ある賞をいただきありがとうございました。これもひとえに理解のある上司、優れた同僚に恵まれたおかげと心より感謝しております。昭和 48 年センターに採用されて以来、大規模科学計算システムの運用管理と利用者支援の業務に携わってまいりました。この間利用者の皆様にも、運用上の様々な場面でご指導・ご支援を頂きました。この場を借りて御礼申し上げます。

これからもより良いセンターを目指し、サービス向上に努めてまいりますのでよろしくお願いいたします。

［報　告］

後藤英昭准教授と曽根秀昭教授の研究グループが 情報教育研究集会・最優秀論文賞を受賞しました

サイバーサイエンスセンターの後藤英昭，曽根秀昭が，情報教育研究集会・最優秀論文賞を受賞しました。

本受賞は，平成22年度情報教育研究集会における下記の発表論文が高く評価されたものです。

対象論文：

「大学間無線LANローミング基盤eduroamの動向と容易な導入方法」　後藤英昭，曽根秀昭

（この賞は，情報教育に関する日本で最大の研究集会である情報教育研究集会において発表された論文・ポスターの中から，情報教育及び関連分野の教育研究を進める上で特に優れた成果と認められる若干の件数を選考し，表彰するもののうち，最も優秀な論文に与えられるものです。）

受賞理由：

情報教育および関連分野の教育・研究への大きな示唆と貢献。

研究背景：

サイバーサイエンスセンター・CSI研究室は、国際的な学術系無線LANローミング基盤であるeduroam(エデュローム)を2006年に日本に初導入し、国立情報学研究所と共同で、国内の基盤であるeduroam JPを立ち上げ、その運用および研究開発を行なってきました。欧州諸国と比較して桁違いに多い1,200以上もの高等教育機関を有する日本において、eduroam導入の技術面および運用面のコスト低減は必須であり、これらを実現するための技術開発と評価運用を行なうとともに、キャンパスネットワークのユビキタス化に向けての研究開発などを推進しています。これらの一連の活動は、研究成果から実運用への適用と、多数機関のeduroam参加促進の効果が実証されています。

[報　告]

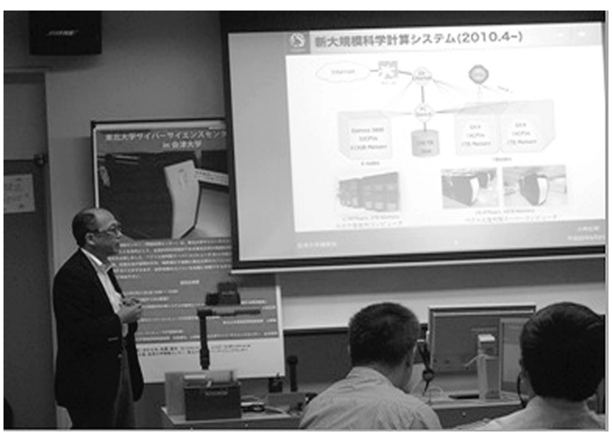
東北大学サイバーサイエンスセンター講習会 in 会津大学

秋葉　真宏
会津大学情報センター（情報処理センター）

平成23年9月21日に会津大学CALL教室1を会場として、東北大学サイバーサイエンスセンターの講習会が開催されました。同講習会は、東北大学サイバーサイエンスセンターと会津大学情報センター（情報処理センター）の共催として開催されましたので、会場の様子などをここでご紹介したいと思います。

講習会は三部構成で、先ず東北大学サイバーサイエンスセンター長の小林先生から「東北大学大規模科学計算システムの運用とベクトルコンピューティングに関する研究開発」に関してご講演がありました。サイバーサイエンスセンターの歴史及びその役割のご説明から始まり、現在のシステムのご紹介がありました。国内近距離航空機・次世代航空機設計技術から地震解析や局地気象監視・予測システムまで非常に幅広い分野での研究で活用されていることが判りました。また、全国共同利用施設に設置されているスーパーコンピュータは学術利用のみに活用されていると思っておりましたが、東北大学のスーパーコンピュータは企業との産学連携利用も多いことが判りました。更に政府の事業仕分けを踏まえた、今後の予定もご教授して頂きました。参加者から何点か質問が出ましたが、東日本大震災での被害状況やその復旧プロセスを始め非常に丁寧にご回答していただきました。次に東北大学情報部情報基盤課の小野様による「東北大学のスーパーコンピュータの利用手続きや負担金等について」のご説明がありました。利用負担金の確認方法や負担金の割引制度及び民間企業利用に関してのご説明がありました。最後に「スーパーコンピュータの体験利用」として東北大学情報部情報基盤課の大泉様、小野様、サイバーサイエンスセンターの水木先生により、CALL1 教室の端末から東北大学のスーパーコンピュータへ接続することで、実際にスーパーコンピュータを体験することができました。非常に丁寧に操作方法を説明していただけたこともあり、問題なく操作できたようです。また、実際にスーパーコンピュータ上でプログラムを走らせることで、東北大学のスーパーコンピュータの高速性能を実感することができました。

最後になりましたが、「東北大学サイバーサイエンスセンター講習会 in 会津大学」の開催にあたり、東北大学サイバーサイエンスセンターのセンター長の小林先生を始め、水木先生、東北大学情報部情報基盤課の大泉様、小野様には大変お世話になりました。この場を借りて厚く御礼申し上げます。



［報　告］

SC11報告

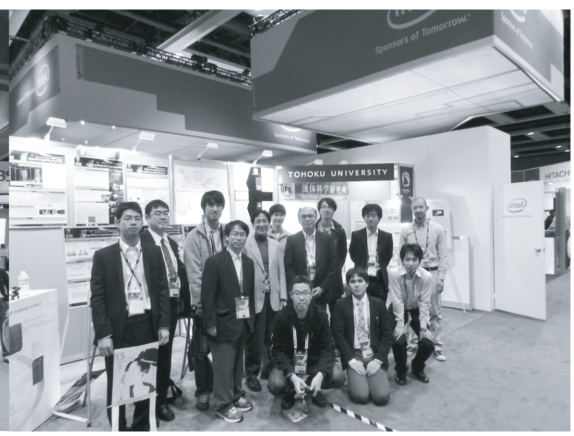
東北大学サイバーサイエンスセンター　小林広明

2011 年 11 月 12 日ー18 日に米国ワシントン州シアトル市において International Conference for High Performance Computing, Networking, Storage and Analysis (通称 SC)が開催されました。SC はスーパーコンピューティングとその関連技術に関する世界最大の国際会議であり、参加者は１万人を超え、その年の最新の研究成果・製品が学界、産業界から発表されてきました。今回、SC11(11 は開催年を意味する)に参加し、サイバーサイエンスセンターの研究活動の展示を行ってきましたので、その報告をします。

SC11は技術論文が発表される本会議と大学・研究所・企業等が最新の成果・製品を発表する展示会から構成され、サイバーサイエンスセンターは流体科学研究所と金属材料研究所と連携し、東北大学としてスーパーコンピューティングに関する活動についての展示を行いました。東北大学展示ブースでは，サイバーサイエンスセンターのスパコンシステムの紹介をはじめ、ベクトルクラウドや３次元積層型ベクトルプロセッサなど次世代高性能計算基盤技術に関する最新の研究成果を紹介しました。また、スパコン研究部に所属する大学院学生５名も帯同させ、各自が取り組む高性能計算技術に関する成果発表も行いました。14～17日の４日間の展示会期間中、東北大学展示ブースへの訪問者は300名以上にものぼり、活発な情報交換を行うことができました。また、訪問者にはセンター紹介パンフレット、手ぬぐい、SX-9折り紙などをくばり、特に折り紙は外国の方々に非常に好評でした。(作り方はyoutubeでも見られます。http://www.youtube.com/watch?v=J60eh_5fnPY)

今年の SC12 は 11月10日～16日にユタ州ソルトレイク市で行われる予定です。

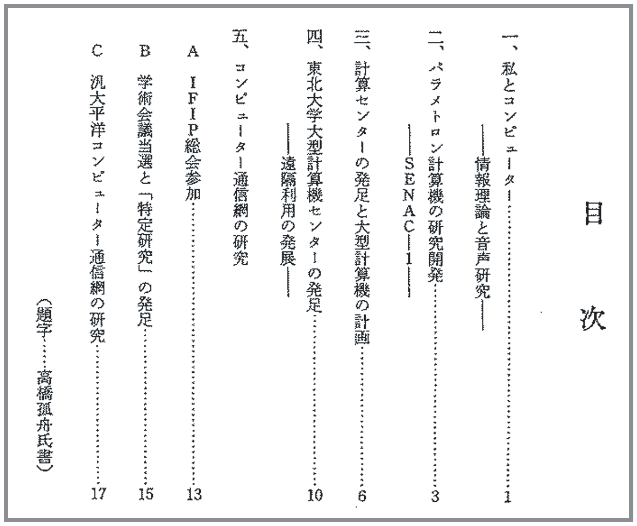
展示の様子



東北大学展示スタッフ

［展示室便り③］

大泉充郎先生



目次

（題字……高橋孤舟氏書）

展示品１　コンピューター　二十年・冊子と目次

展示品２　「JIS に準拠した FORTRAN」（基礎コース、拡充コース、演習コース）と「新 JIS に準拠した FORTRAN」

今回は、初代の大型計算機センター長もされた大泉充郎先生に関する展示品です。

展示品の一つに、東北大学名誉教授である大泉充郎先生が停年退職時(1971 年、昭和 46 年)に発行した「コンピューター二十年」という冊子があります。展示室便り①では、この中から SENAC 開発秘話を紹介いたしました。展示品１には、目次も掲載しましたのでご覧ください。この内容から SENAC-1 の研究開発、学内の計算センター、大型計算機センターの発足、そしてコンピュータ通信網の研究に尽力を尽くされたことが分かります。ま

た、この冊子から大泉先生は1970年にはすでにコンピュータとネットワークの関係をかなり具体的に意識していたことが伺えます。冊子からの文章を紹介いたします。

私が、大型計算機のシステム設計に当って考えたのは、バッチ（一括）処理と並行して会話型の計算サービスを提供したいということであった。バッチ処理はセンター側が利用者からプログラム（カード）を受け取ってまとめて計算機にかける方式で計算機の利用効率を限度迄高めることが出来る。之に対し会話型は利用者が計算機を通信回線で結ばれたタイプライターを通じ、計算機と直接相対して質問、応答の形で計算を進める方式で、接続されるタイプライターが多い場合、計算機本体の利用率の下がることが欠点とされる。しかし、利用者としてはプログラムの誤り（虫）が容易に見つかること、疑問の点は計算機に質問出来ること、計算が自分の意図したものとは違っていることを発見した時は途中で中止させることが出来るなど優れた特長を持っている。大型機の出現によって計算速度は SENAC-1 の千倍以上速くなったが、バッチ処理方式では利用者は全く計算機と隔離されており、計算機に対する親しみが持てないのに対し、会話型では SENAC-1 時代と同じような親近感を持てるのである。決定された機種は NEAC2200 モデル 500 が四台であったので、内三台をバッチ、一台を会話型で用いることにした。

世の中では、これからバッチ処理のサービスを計画しているときに、すでに会話処理による遠隔利用も考えていたことが分かります。

また、「C　汎太平洋コンピューター通信網の研究」の中には、つぎの文章があります。

衛星コンピューター通信網の将来

私はこの段階までで東北大を去る事となったが、国際コンピューター通信網は今後日本の主導の下に国連又は国連大学等の協力を得て、東南アジア、開発途上国援助の一環として活躍する事も考えられ、その将来が期待されている。

尚、米国における衛星を使用する国内コンピューター通信網の発展は極めて著しく、これは近い将来我が国に大きな影響を及ぼすものと考えられるので、我々は真剣にその対策を立てる必要に迫られている。

この文章を書かれてから40年経ちましたが、大泉先生の予測はどうだったでしょう。

展示品２は、大泉先生が監修された「JISに準拠したFORTRAN」（基礎コース、拡充コース、演習コースの３冊で構成されています）と「新JISに準拠したFORTRAN」という書籍です。これらの本を教科書にFORTRANを学び、各地の大型計算機センターを使用された研究者の方も大勢いるのではないでしょうか。

[共同利用支援係]

［Web 版大規模科学計算システムニュースより］

大規模科学計算システムニュースに掲載された記事の一部を転載しています。　http://www.ss.isc.tohoku.ac.jp/tayori/

利用相談室のご案内(No.125)

サイバーサイエンスセンターでは、本館、弘前大学、秋田大学、山形大学に利用相談室を設けています。テクニカルアシスタントの担当分野、日程等詳しくは以下の Web ページをご覧ください。

http://www.ss.isc.tohoku.ac.jp/tayori/tayori/119.html#3

平日 8:30～17:15 の間、電話、本館 1 階共同利用支援係窓口で利用相談を常時受け付けております。相談室にテクニカルアシスタントが不在の時間帯は、センター職員が対応いたします。メールは 24 時間受け付けますが、回答は上記の時間帯になりますのであらかじめご了承ください。利用申請、利用負担金、プログラムの並列化、高速化、ベクトル化やアプリケーションに関することなど、センター利用に関する質問を受け付けます。また、相談室には各種マニュアルや書籍も揃えています。閲覧のみも可能ですのでお気軽にご利用ください。

サイバーサイエンスセンター本館　利用相談室
E-mail：sodan05@isc.tohoku.ac.jp
Tel.　：022-795-6153

（共同利用支援係）

― SENAC 執筆要項 ―

１. お寄せいただきたい投稿内容

次のような内容の投稿のうち、当センターで適当と判定したものを掲載します。その際に原稿の修正をお願いすることもありますのであらかじめご了承ください。

・一般利用者の方々が関心をもたれる事項に関する論説
・センターの計算機を利用して行った研究論文の概要
・プログラミングの実例と解説
・センターに対する意見、要望
・利用者相互の情報交換

２. 執筆にあたってご注意いただく事項

(1)原稿は横書きです。
(2)術語以外は、「常用漢字」を用い、かなは「現代かなづかい」を用いるものとします。
(3)学術あるいは技術に関する原稿の場合､200 字～400 字程度のアブストラクトをつけてください。
(4)参考文献は通し番号を付し末尾に一括記載し、本文中の該当箇所に引用番号を記入ください。
　・雑誌：著者, タイトル, 雑誌名, 巻, 号, ページ, 発行年
　・書籍：著者, 書名, ページ, 発行所, 発行年

３. 原稿の提出方法

原稿のファイル形式は Word を標準としますが、PDF での提出も可能です。サイズ*は以下を参照してください。ファイルは電子メールで提出してください。

－Word の場合－
　・用紙サイズ：A4
　・余白：上＝30mm　下＝25mm　左右＝25mm　綴じ代＝0
　・標準の文字数（45 文字 47 行）

＜文字サイズ等の目安＞
　・表題＝ゴシック体 14pt 中央　・副題＝明朝体 12pt 中央
　・氏名＝明朝体 10.5pt 中央
　・所属＝明朝体 10.5pt 中央
　・本文＝明朝体 10.5pt
　・章・見出し番号＝ゴシック体 11pt～12pt
　　*余白サイズ、文字数、文字サイズは目安とお考えください。

４. その他

(1)執筆者には、希望があれば別刷 50 部を進呈します。50 部を超える分については、著者の実費負担とします。別刷の希望部数等は投稿の際に申し出てください。
(2)投稿予定の原稿が 15 ページを超す場合は以下まで前もってご連絡ください。
(3)初回の校正は、執筆者が行って、誤植の防止をはかるものとします。
(4)原稿の提出先は次のとおりです。

東北大学サイバーサイエンスセンター内　情報部情報基盤課共同利用支援係
e-mail　uketuke@isc.tohoku.ac.jp
TEL　022-795-3406

編 集 後 記

寒い日がつづきます。早いもので2012年を迎えました。昨年は震災にはじまり、辛く、悲しいことが続きました。ただ、今号の巻頭言にもありますとおり、より安全で豊かな生活の実現にむけて、計算科学の先生方と我々スパコンセンターが協調しながら、できること一つ一つに前向きに取り組んで行きたいとの思いを新たにしました。

元旦に、昨年生まれたばかりの坊主と初詣に行きました。家族三人大吉。今年なにか良いことあるでしょうか！？ (R.E)

1年前、現在のような社会状況を誰が想像していたでしょうか。未曾有の東日本大震災による被災は言葉に表せないくらい大変な思いを多くの人がしています。約33万5000人の人が避難･転居先で正月を迎えたと報道されています。1日も早い復旧､復興が望まれます。その道のりを今後も見守っていきたいと思います。

これから、ますます寒くなってきます。風邪など、特にインフルエンザには十分な注意が必要です。

今号の「展示室便り」には、大型計算機センター（サイバーサイエンスセンターの前身）の初代センター長の記事が掲載されています。ぜひ展示室にも足を運んでください。”温故知新”も良いものです。(Y.T)

サイバーサイエンスセンターより東へ 2.5 キロにある青葉城址公園内の伊達政宗像（2012.1.1撮影）

SENAC編集部会

小林広明　曽根秀昭　水木敬明　後藤英昭
江川隆輔　早坂哲夫　大泉健治　小野　敏
高橋洋一　斉藤くみ子

平成24年1月発行
編集・発行　東北大学
サイバーサイエンスセンター
仙台市青葉区荒巻字青葉6-3
郵便番号　980-8578
印　刷　大成印刷株式会社

索　　引

SENAC Vol.44 (No.1～No.4)　　　　2011.1～2011.10

システム一覧

計算機システム	ホスト名	機　種
スーパーコンピュータ	super.isc.tohoku.ac.jp	SX-9
並列コンピュータ	gen.isc.tohoku.ac.jp	Express5800

サービス時間

利用システム名	利用時間帯
スーパーコンピュータ	連　続　運　転
並列コンピュータ	連　続　運　転
館 内 利 用	月曜日～金曜日は 8:30～21:00、 土・日・祝日は閉館

ジョブクラスと制限値

計算機システム	処　理	ジョブクラス	ＣＰＵ時間	メモリ容量
スーパーコンピュータ	会話型	(4cpu)	1 時間	8GB
	バッチ処理	ss　(4cpu)	1 時間	256GB
		s　(4cpu)	無制限	32GB
		p8　(8cpu)	〃	512GB
		p16　(16cpu)	〃	1024GB
		p32　(32cpu)	〃	1024GB×2
		p64　(64cpu)	〃	1024GB×4
並列コンピュータ	会話型	(4 並列)	1 時間	8GB
	バッチ処理	as　(非並列)	無制限	16GB
		am　(Marc 専用)	〃	16GB
		am2　(Marc 専用)	〃	128GB
		a8　(8 並列)	〃	128GB
		a16　(16 並列)	〃	256GB
		a32　(32 並列)	〃	512GB

目次

東北大学サイバーサイエンスセンター
大規模科学計算システム広報　Vol.45 No.1 2012−1