ISSN 1344—9567

スーパーコンピューティングニュース

Vol.22 No.5, 2020.9

東京大学情報基盤センター
INFORMATION TECHNOLOGY CENTER, THE UNIVERSITY OF TOKYO

スーパーコンピュータシステム 利用負担金表

Oakbridge-CX スーパーコンピュータシステム 利用負担金表（2020 年 4 月 1 日）

コース		負担金額（税込）		ディスク容量	備考
		大学・公共機関等	企業		
パーソナルコース		申込 1 セット当り， 最大 3 セットまで 100,000 円 （8,640 ノード時間）		申込 1 セット当り /work 4TB 利用者当り /home 50GB	最大ノード数 256 ノード
グループコース	一般申込	申込 1 セット当り 100,000 円 （8,640 ノード時間）	申込 1 セット当り 120,000 円 （8,640 ノード時間）	グループ 1 セット当り /work 4TB 利用者当り /home 50GB	最大ノード数 256 ノード
	ノード固定	申込 1 セット当り 150,000 円 （8,640 ノード時間）	申込 1 セット当り 180,000 円 （8,640 ノード時間）		
トークン追加		8,400 円 （720 ノード時間）	10,000 円 （720 ノード時間）		
ディスク追加		6,480 円／（1TB*年）			1TB 単位で申込可 （/work のみ）

※トークン消費係数は 1.00 である。
※括弧内のノード時間は付与するトークン量。実行したジョブのノード時間積と消費係数に応じてトークンが消費される。
付与したトークンは，利用期間内に全量が使用できることを保証するものではない。
トークンは利用期間内に限り有効とし，利用終了後に残量がある場合でも繰越や利用負担金の返還は行わない。
トークンの他のシステムへの移行については，「トークン移行におけるノード時間積の換算表」を参照。
※ノード固定の申し込みには審査を要する。
※/home のディスク容量はパーソナルコースやグループコースに複数所属していても利用者当り 50GB 固定。

Oakforest-PACS スーパーコンピュータシステム 利用負担金表（2020 年 4 月 1 日）

コース	負担金額（税込）		ディスク容量	備考
	大学・公共機関等	企業		
パーソナルコース	申込 1 セット当り， 最大 6 セットまで 50,000 円 （8,640 ノード時間）		申込 1 セット当り /work 1TB 利用者当り /home 50GB	最大ノード数 2,048 ノード
グループコース	申込 1 セット当り 50,000 円 （8,640 ノード時間）	申込 1 セット当り 60,000 円 （8,640 ノード時間）	グループ 1 セット当り /work 1TB 利用者当り /home 50GB	最大ノード数 2,048 ノード
トークン追加	4,200 円 （720 ノード時間）	5,000 円 （720 ノード時間）		
ディスク追加	6,480 円／（1TB*年）			1TB 単位で申込可 （/work のみ）

※トークン消費係数は 1.00 である。
※括弧内のノード時間は付与するトークン量。実行したジョブのノード時間積と消費係数に応じてトークンが消費される。
付与したトークンは，利用期間内に全量が使用できることを保証するものではない。
トークンは利用期間内に限り有効とし，利用終了後に残量がある場合でも繰越や利用負担金の返還は行わない。
トークンの他のシステムへの移行については，「トークン移行におけるノード時間積の換算表」を参照。
※/home のディスク容量はパーソナルコースやグループコースに複数所属していても利用者当り 50GB 固定。

Reedbush スーパーコンピュータシステム(Reedbush-H/L) 利用負担金表(2020 年 4 月 1 日)

コース		負担金額(税込)		ディスク容量	備考
		大学・公共機関等	企業		
パーソナルコース		申込 1 セット当り, 最大 2 セットまで 75,000 円 (8,640 ノード時間)		申込 1 セット当り /lustre 1TB 利用者当り /home 2GB	Reedbush-H 最大ノード数 32 ノード Reedbush-L 最大ノード数 16 ノード
グループコース	一般申込	申込 1 セット当り 75,000 円 (8,640 ノード時間)		グループ 1 セット当り /lustre 1TB 利用者当り /home 2GB	Reedbush-H 最大ノード数 32 ノード Reedbush-L 最大ノード数 16 ノード
	公募制度 Reedbush-H	申込 1 セット当り 公募制度 180,000 円 (21,600 ノード時間)	申込 1 セット当り 公募制度 216,000 円 (21,600 ノード時間)	グループ 1 セット当り /lustre 4TB 利用者当り /home 2GB	最大ノード数 32 ノード
	公募制度・ ノード固定 Reedbush-L	申込 1 セット当り 公募制度 300,000 円 ノード固定 450,000 円 (34,560 ノード時間)	申込 1 セット当り 公募制度 360,000 円 ノード固定 540,000 円 (34,560 ノード時間)	グループ 1 セット当り /lustre 4TB 利用者当り /home 2GB	最大ノード数 16 ノード
トークン追加		6,300 円 (720 ノード時間)	7,500 円 (720 ノード時間)		
ディスク追加		6,480 円／(1TB*年)			1TB 単位で申込可 (/lustre のみ)

※トークン消費係数は下記の通りである。

Reedbush-H: 2.50, Reedbush-L: 4.00

※括弧内のノード時間は付与するトークン量。実行したジョブのノード時間積と消費係数に応じてトークンが消費される。

付与したトークンは，利用期間内に全量が使用できることを保証するものではない。

トークンは利用期間内に限り有効とし，利用終了後に残量がある場合でも繰越や利用負担金の返還は行わない。

トークンの他のシステムへの移行については，「トークン移行におけるノード時間積の換算表」を参照。

※公募制度・ノード固定の申し込みには審査を要する。

※/home のディスク容量はパーソナルコースやグループコースに複数所属していても利用者当り 2GB 固定。

トークン移行におけるノード時間積の換算表

移行元＼移行先	Reedbush-H/L システム	Oakforest-PACS システム	Oakbridge-CX システム
Reedbush-H/L システム	―	1.5	0.75
Oakforest-PACS システム	0.6	―	0.5
Oakbridge-CX システム	1.3	2	―

移行先に追加されるトークン量（ノード時間）＝移行トークン量×係数

注意事項（Oakbridge-CX, Oakforest-PACS, Reedbush,スーパーコンピュータシステム 共通）

- 「大学・公共機関等」は大学，高等専門学校及び大学共同利用機関，文部科学省所管の独立行政法人，学術研究及び学術振興を目的とする国又は地方公共団体が所管する機関，並びに文部科学省科学研究費補助金の交付を受けて学術研究を行う者に適用する。
- 「企業」の申し込みには，企業利用申込書添付書類の提出および審査を要する。
- 利用期間は，利用開始月から終了月の末日またはサービス休止前までとする。利用期間内に計算機利用を中止した場合であっても利用負担金額の変更は行わない。年度の途中で利用開始または終了する場合の負担金額は月数別利用負担金表（Web ページ）を参照すること。
- 前掲の利用負担金表は基本セットの内容であり，最小セットについては Web ページを参照すること。
- パーソナルコース（ただし，本センターのスーパーコンピュータシステムに初めて登録された利用者）においてのみ，利用開始月の翌月末日までに利用を中止することができる。利用負担金はパーソナルコースの利用期間1ヶ月の金額を適用し，請求する。
- 利用負担金は，原則として利用開始月に応じ，以下の月の初旬に一括して請求する。
  - 利用開始月が 4 月から 7 月までは 10 月，8 月から 9 月までは 12 月，10 月から 12 月までは 3 月，1 月から 3 月までは 3 月末。
  - 前年度内に事前申込をした分については，利用開始月に関わらず，7 月初旬の請求となる。
- 利用負担金額が減額となる変更はできない。
- コース間の変更については，利用負担金が増額になる場合のみ別途相談に応じる。（ただし，利用者番号変更の場合がある。）
- グループコースのディスク量は，グループ全体の上限値である。

スーパーコンピュータシステム ジョブクラス制限値

Oakbridge-CX スーパーコンピュータシステム ジョブクラス制限値(2019 年 7 月 1 日)

キュー名※1	ノード数※2（最大コア数）	制限時間（経過時間）	メモリー容量（GB）※3	パーソナルコース	グループコース 申込	グループコース 一般	グループコース 固定ノード
debug	1 ～ 16 (896)	30 分	168	○	○	○
short	1 ～ 8 (448)	8 時間	168	○	○	○
(regular)						
small	1 ～ 16 (896)	48 時間	168	○	○	○
medium	17 ～ 64 (3584)	〃	〃	○	○	○
large	65 ～ 128 (7168)	〃	〃	○	○	○
x-large	129 ～ 256 (14336)	24 時間	〃	○	○	○
challenge	1 ～ 1368 (76608)	24 時間	168	★	★	★
任意	申込数	任意 ※4	168	×	×	○
(interactive) ※5						
interactive_n1	1 (56)	2 時間	168	○	○	○
interactive_n8	2 ～ 8 (448)	10 分	〃	○	○	○

※1 キューの指定(“#PJM -L“rscgrp=キュー名””)は, regular, debug, short を小文字で指定する
regular キューはノード数の指定(“#PJM -L“node=ノード数””)でノード数別のキューに投入される
※2 トークンの消費係数は 1.00
※3 1ノード当りの利用者が利用可能なメモリー容量
※4 申込ノード数の合計以内ならば, キュー名・制限時間(原則 48 時間以内)は相談の上, 任意に設定可能
※5 インタラクティブジョブの起動は次のとおり (トークン消費なし)
pjsub --interact -g グループ名 -L“rscgrp=interactive,node=ノード数”

Oakforest-PACS スーパーコンピュータシステム ジョブクラス制限値(2018 年 4 月 1 日)

キュー名※1	ノード数※2（最大コア数）	制限時間（経過時間）	メモリー容量（GB）※3	パーソナルコース	グループコース
debug-cache	1 ～ 128 (8704)	30 分	82	○	○
debug-flat	1 ～ 128 (8704)	30 分	96	○	○
(regular-cache)					
small-cache	1 ～ 128 (8704)	48 時間	82	○	○
medium-cache	129 ～ 512 (34816)	〃	〃	○	○
large-cache	513 ～ 1024 (69632)	〃	〃	○	○
x-large-cache	1025 ～ 2048 (139264)	24 時間	〃	○	○
(regular-flat)					
small-flat	1 ～ 128 (8704)	48 時間	96	○	○
medium-flat	129 ～ 512 (34816)	〃	〃	○	○
large-flat	513 ～ 1024 (69632)	〃	〃	○	○
x-large-flat	1025 ～ 2048 (139264)	24 時間	〃	○	○
challenge	1 ～ 8208 (558144)	24 時間	82 / 96	★	★
(interactive-cache) ※4					
interactive_n1-cache	1 (68)	2 時間	82	○	○
interactive_n16-cache	2 ～ 16 (1088)	10 分	〃	○	○
(interactive-flat) ※4					
interactive_n1-flat	1 (68)	2 時間	96	○	○
interactive_n16-flat	2 ～ 16 (1088)	10 分	〃	○	○
prepost	1 (28)	6 時間	222	○	○

★ 審査による課題選定の上, 月1回の一定期間のみ利用可能(原則として月末処理日前日の朝～翌日朝)
※1 キューの指定(“#PJM -L“rscgrp=キュー名””)は, regular-cache/flat, debug-cache/flat を小文字で指定する
regular-cache/flat キューはノード数の指定(“#PJM -L“node=ノード数””)でノード数別のキューに投入される
※2 トークンの消費係数は 1.00
※3 1ノード当りの利用者が利用可能なメモリー容量
※4 インタラクティブジョブの起動は, pjsub --interact -g グループ名 -L“rscgrp=キュー名,node=ノード数”
(キュー名は interactive-cache/flat, トークン消費なし)

Reedbush スーパーコンピュータシステム（Reedbush-H） ジョブクラス制限値（2018 年 9 月 27 日）

キュー名※1	ノード数※2（最大 GPU 数）	制限時間（経過時間）	メモリー容量（GB）※3	パーソナルコース	グループコース	
					一般申込	公募制度
h-debug	1 ～ 4 (8)	30 分	244	○	○	○
h-short	1 ～ 4 (8)	2 時間	244	○	○	○
(h-regular)						
h-small	1 ～ 4 (8)	48 時間	244	○	○	○
h-medium	5 ～ 8 (16)	〃	〃	○	○	○
h-large	9 ～ 16 (32)	〃	〃	○	○	○
h-x-large	17 ～ 32 (64)	24 時間	〃	○	○	○
h-challenge	1 ～ 120 (240)	24 時間	244	★	★	★
(h-interactive) ※4						
h-interactive_1	1 (2)	2 時間	244	○	○	○
h-interactive_2	2 (4)	30 分	〃	○	○	○
(h-regular-low) ※5						
h-small-low	1 ～ 4 (8)	12 時間	244	▲	▲	▲
h-medium-low	5 ～ 8 (16)	〃	〃	×	▲	▲
h-large-low	9 ～ 16 (32)	〃	〃	×	▲	▲
h-x-large-low	17 ～ 32 (64)	6 時間	〃	×	▲	▲

▲　パーソナルコースは 最大 1 ノード，グループコースは申込ノード数の 4 分の 1 まで実行可（公募制度による申し込みの場合は申込ノード数まで実行可）
★　審査による課題選定の上，月1回の一定期間のみ利用可能（原則として月末処理日前日の朝～翌日朝）
※1 キューの指定（〝#PBS -q キュー名〞）は，h-short, h-regular, h-debug を小文字で指定する
h-regular キューはノード数の指定（〝#PBS -l select=ノード数〞）でノード数別のキューに投入される
※2 トークンの消費係数は 2.50
※3 1ノード当りの利用者が利用可能なメモリー容量
※4 インタラクティブジョブの起動は次のとおり（トークン消費なし）
qsub -I -q h-interactive -l select=ノード数 -l walltime=XX:XX -W group_list=グループ名
※5 非優先ジョブクラス（低プライオリティキュー）は，トークンの追加申込が締め切られ，ジョブ実行に必要なトークン残量が不足した場合のみ実行可

Reedbush スーパーコンピュータシステム（Reedbush-L） ジョブクラス制限値（2020 年 4 月 1 日）

キュー名※1	ノード数※2（最大 GPU 数）	制限時間（経過時間）	メモリー容量（GB）※3	パーソナルコース	グループコース		
					一般申込	公募制度	ノード固定
l-debug	1 ～ 4 (16)	30 分	244	○	○	○	○
(l-regular)							
l-small	1 ～ 4 (16)	168 時間	244	○	○	○	○
l-medium	5 ～ 8 (32)	〃	〃	○	○	○	○
l-large	9 ～ 16 (64)	〃	〃	○	○	○	○
任意	申込数	任意 ※4	244	×	×	×	○
(l-interactive) ※5							
l-interactive_1	1 (4)	24 時間	244	○	○	○	○
l-interactive_2	2 (8)	6 時間	〃	○	○	○	○
l-interactive_4	3 ～ 4 (16)	1 時間	〃	○	○	○	○
(l-regular-low) ※6							
l-small-low	1 ～ 4 (16)	12 時間	244	▲	▲	▲	▲
l-medium-low	5 ～ 8 (32)	〃	〃	×	▲	▲	▲
l-large-low	9 ～ 16 (64)	〃	〃	×	▲	▲	▲

▲　パーソナルコースは 最大 1 ノード，グループコースは申込ノード数の 4 分の 1 まで実行可（公募制度・ノード固定による申し込みの場合は申込ノード数まで実行可）
※1 キューの指定（〝#PBS -q キュー名〞）は，l-debug, l-regular を小文字で指定する
l-regular キューはノード数の指定（〝#PBS -l select=ノード数〞）でノード数別のキューに投入される
※2 トークンの消費係数は 4.00
※3 1ノード当りの利用者が利用可能なメモリー容量
※4 申込ノード数の合計以内ならば，キュー名・制限時間（原則 168 時間以内）は相談の上，任意に設定可能
※5 インタラクティブジョブの起動は次のとおり（トークン消費あり）
qsub -I -q l-interactive -l select=ノード数 -l walltime=XX:XX -W group_list=グループ名
※6 非優先ジョブクラス（低プライオリティキュー）は，トークンの追加申込が締め切られ，ジョブ実行に必要なトークン残量が不足した場合のみ実行可

センターから

サービス休止等のお知らせ

2020 年 9 月下旬からの計算機サービス予定は以下のとおりです。

Oakbridge-CX スーパーコンピュータシステム

○ Oakbridge-CX スーパーコンピュータシステム サービス休止のお知らせ

日　付	利用者サービス	センター内作業
9 月 25 日（金）～ 9 月 28 日（月）	9 月 25 日（金） 9:00 ～ 9 月 28 日（月）17:00 までサービス休止	月末処理，空調機メンテナンス，柏キャンパスにおける特別高圧受変電設備点検・二次変電設備点検
10 月 30 日（金）	9:00 ～ 20:00 までサービス休止	月末処理
11 月 27 日（金）	9:00 ～ 20:00 までサービス休止	月末処理

・ Oakbridge-CX システムは，原則 24 時間サービスを行っています。
ただし，月末処理日 （原則として毎月最終金曜日） はサービスを停止します。

○ Oakbridge-CX スーパーコンピュータシステム 大規模 HPC チャレンジ のお知らせ

大規模 HPC チャレンジ 実施期間
9 月 24 日（木）9:00 ～ 9 月 25 日（金）9:00 まで 10 月 29 日（木）9:00 ～ 10 月 30 日（金）9:00 まで 11 月 26 日（木）9:00 ～ 11 月 27 日（金）9:00 まで

・上記期間中，Oakbridge-CX の debug，short，regular，interactive，prepost，ノード固定 および 講義用キューのサービスを休止します。
ログインノードは通常どおり利用できます。

Oakforest-PACS スーパーコンピュータシステム

○ Oakforest-PACS スーパーコンピュータシステム サービス休止のお知らせ

日　付	利用者サービス	センター内作業
9 月 25 日（金）～ 9 月 28 日（月）	9 月 25 日（金） 9:00 ～ 9 月 28 日（月）17:00 までサービス休止	月末処理，空調機メンテナンス，柏キャンパスにおける特別高圧受変電設備点検・二次変電設備点検
10 月 30 日（金）	9:00 ～ 22:00 までサービス休止	月末処理
11 月 27 日（金）	9:00 ～ 22:00 までサービス休止	月末処理

・ Oakforest-PACS スーパーコンピュータシステムは，原則 24 時間サービスを行っています。
ただし，月末処理日（原則として毎月最終金曜日）はサービスを停止します。

○ Oakforest-PACS スーパーコンピュータシステム 大規模 HPC チャレンジ のお知らせ

大規模 HPC チャレンジ 実施期間
9 月 24 日（木）9:00 ～ 9 月 25 日（金）9:00 まで 10 月 29 日（木）9:00 ～ 10 月 30 日（金）9:00 まで 11 月 26 日（木）9:00 ～ 11 月 27 日（金）9:00 まで

・上記期間中，Oakforest-PACS の regular-flat，regular-cache，debug-flat，debug-cache，interactive-flat，interactive-cache および 講義用キューのサービスを休止します。ログインノード，prepost キューは通常どおり利用できます。

Reedbush スーパーコンピュータシステム

○ Reedbush スーパーコンピュータシステム（Reedbush-H, Reedbush-L）サービス休止のお知らせ

日　付	利用者サービス	センター内作業
9 月 25 日（金）～ 9 月 28 日（月）	9 月 25 日（金） 9:00 ～ 9 月 28 日（月）17:00 までサービス休止	月末処理，本郷キャンパスにおける二次変電設備定期点検
10 月 23 日（金）～ 10 月 26 日（月）	10 月 23 日（金） 9:00 ～ 10 月 26 日（月）17:00 までサービス休止	月末処理，空調機メンテナンス，本郷キャンパスにおける特別高圧受変電施設定期点検のためサービス休止
11 月 27 日（金）	9:00 ～ 17:00 までサービス休止	月末処理

・Reedbush-H, Reedbush-L スーパーコンピュータシステムは，原則 24 時間サービスを行っています。
ただし，月末処理日（原則として毎月最終金曜日）はサービスを停止します。

○ Reedbush スーパーコンピュータシステム（Reedbush-H）大規模 HPC チャレンジ のお知らせ

大規模 HPC チャレンジ 実施期間
9 月 24 日（木）9:00 ～ 9 月 25 日（金）9:00 まで 10 月 22 日（木）9:00 ～ 10 月 23 日（金）9:00 まで 11 月 26 日（木）9:00 ～ 11 月 27 日（金）9:00 まで

・上記期間中， Reedbush-H の h-debug, h-short, h-regular, h-interactive および 講義用キューのサービスを休止します。
Reedbush-L, ログインノード および prepost キューは利用可能です。

【注意事項】

- サービス休止等の計画は原稿作成時の予定です。やむを得ずサービスを変更したり，休止したりする場合がありますので，最新の情報は login 時のメッセージ及びスーパーコンピューティング部門の Web ページの運用スケジュール (https://www.cc.u-tokyo.ac.jp/supercomputer/schedule.php) をご確認ください。
- 平日の 9:00～17:00 以外，休日（土・日・祝日等）は，システム障害等でサービスが停止した場合，運転を継続できない場合があります。その場合は，その時間をもってサービスを中止しますのでご了承ください。

システム変更等のお知らせ

(2020.7.7 － 2020.9.2 変更)

1. ハードウェア

1.1 Oakbridge-CX スーパーコンピュータシステム … なし

1.2 Oakforest-PACS スーパーコンピュータシステム … なし

1.3 Reedbush スーパーコンピュータシステム (Reedbush-H/L) … なし

2. ソフトウェア

2.1 Red Hat Enterprise Linux 7, CentOS 7 (Oakbridge-CX)

Intel 開発環境 2020 (update1)		(2020.07.31)
Intel Compiler	2020.1.217	
Intel MPI	2019.7.217	
TeX Live	2020	(2020.08.28)

インストールを実施しました。利用方法については、利用支援ポータルのお知らせ、または
ドキュメント閲覧より利用手引書をご覧ください。

2.2 RedHat Enterprise Linux 7, CentOS 7 (Oakforest-PACS) … なし

2.3 RedHat Enterprise Linux 7 (Reedbush-H/L)

MVAPICH2 (CUDA 10.0.130, PGI 18.4, GPUDirect 対応版)	2.3.4	(2020.07.31)
TeX Live	2020	(2020.08.28)

インストールを実施しました。利用方法については、利用支援ポータルのドキュメント閲覧より
利用手引書または各資料をご覧ください。

Lustre ファイルシステム	(2020.08.21)

Lustre ファイルシステムの不良対策を実施しました(Lustre ファイルシステムのバージョンアップ
を実施)。

3. その他

3.1 Oakbridge-CX における show_module コマンド導入について

利用可能なアプリケーション/ライブラリを一覧表示する show_module コマンドが使用可能になりました。利用方法の詳細は、Oakbridge-CX 利用支援ポータルの Oakbridge-CX 利用手引書をご参照ください。

※実行例(ヘルプの表示)

```
$ show_module --help
show_module: 1.0 2020/07/31

Usage: show_module [-a]

Options:
  -a                        Show all available module list
```

スーパーコンピュータシステム「大規模 HPC チャレンジ」課題募集のお知らせ

Oakbridge-CX、Oakforest-PACS、Reedbush-H スーパーコンピュータシステムでは、「大規模 HPC チャレンジ」を実施しています。「大規模 HPC チャレンジ」は、スーパーコンピュータシステムがもつ最大規模のノード数を、最大 24 時間・1 研究グループで計算資源の占有利用ができる公募型プロジェクトです。採択条件等については、以下をご覧ください。皆様からの課題応募をお待ちしております。

1. 提供資源

以下のスーパーコンピュータシステムの計算ノードを最大 24 時間占有利用することができます。

・Oakbridge-CX スーパーコンピュータシステムの計算ノード 1,280 ノード（内 SSD 搭載 112 ノード）
・Oakforest-PACS スーパーコンピュータシステムの計算ノード 8,208 ノード
・Reedbush-H スーパーコンピュータシステムの計算ノード 120 ノード（GPU 240 基）

2. 利用案内

・1 ヶ月に 1 回、原則として月末処理前の木曜日 9:00～金曜日の 9:00 までの最大 24 時間、提供資源を占有利用することが可能です。
（Oakbridge-CX につきましては、大規模 HPC チャレンジの当日に、チャレンジ用の設定に変更いたします。この変更は 1 時間程度の作業時間が見込まれるため、大規模 HPC チャレンジのご利用時間は 10 時頃開始、翌日 9 時終了となります。）
（Oakforest-PACS のメモリモードにつきましては、通常サービス中は、全計算ノードの約半分を flat モード、もう半分を cache モードに設定して、利用者の皆様にご利用いただいていますが、大規模 HPC チャレンジの当日に、チャレンジ実施者の希望に沿って全計算ノードをどちらかのモードに変更いたします。この変更は 3 時間程度の作業時間が見込まれるため、大規模 HPC チャレンジのご利用時間は 12 時頃開始、翌日 9 時終了となります。）
・課題は公募制とし、現ユーザーに限定せず、広く課題を募集します。個人、およびグループによる応募が可能ですが、各月に 1 グループの採用を原則とします。
・**本制度により得られた成果については公開して頂きます。**成果公開には東京大学情報基盤センターまたは最先端共同 HPC 基盤施設のスーパーコンピュータシステムを利用し、「大規模 HPC チャレンジ」制度によって実施した旨を明記していただきます。また、「スーパーコンピューティングニュース」や広報誌等への成果報告記事の執筆などを行っていただきます。
・センターまたは最先端共同 HPC 基盤施設の主催、共催するセミナー、ワークショップ等でご発表いただく場合があります。
・利用料金は無料です。

3. 実施日程

2020 年度の今後の「大規模 HPC チャレンジ」実施日程は表 1～3 のとおりです。

表 1. 2020 年度 Oakbridge-CX 大規模 HPC チャレンジ実施日程

実施日時	募集締切	審査	採択通知
2020 年 12 月 17 日(木) 10:00 ～ 12 月 18 日(金) 9:00 2021 年 1 月 28 日(木) 10:00 ～ 1 月 29 日(金) 9:00 2021 年 2 月 25 日(木) 10:00 ～ 2 月 26 日(金) 9:00 2021 年 3 月 30 日(火) 10:00 ～ 3 月 31 日(水) 9:00	2020 年 10 月 26 日（月） 17:00【締切】	11 月上旬	11 月中旬

表 2. 2020 年度 Oakforest-PACS 大規模 HPC チャレンジ実施日程

実施日時	募集締切	審査	採択通知
2020 年 12 月 17 日(木) 12:00 ～ 12 月 18 日(金) 9:00 2021 年 1 月 28 日(木) 12:00 ～ 1 月 29 日(金) 9:00 2021 年 2 月 25 日(木) 12:00 ～ 2 月 26 日(金) 9:00 2021 年 3 月 30 日(火) 12:00 ～ 3 月 31 日(水) 9:00	2020 年 10 月 26 日（月） 17:00【締切】	11 月上旬	11 月中旬

表 3. 2020 年度 Reedbush-H 大規模 HPC チャレンジ実施日程

実施日時	募集締切	審査	採択通知
2020 年 12 月 17 日(木) 9:00 ～ 12 月 18 日(金) 9:00 2021 年 1 月 28 日(木) 9:00 ～ 1 月 29 日(金) 9:00 2021 年 2 月 25 日(木) 9:00 ～ 2 月 26 日(金) 9:00 2021 年 3 月 30 日(火) 9:00 ～ 3 月 31 日(水) 9:00	2020 年 10 月 26 日（月） 17:00【締切】	11 月上旬	11 月中旬

・メンテナンス等の都合により募集スケジュールが変更となることがあります。最新情報は Web Page[1]をご覧ください。

・年複数回を申し込むことも可能ですが、申込状況によりご希望に添えない場合もありますのであらかじめご了承ください。また、一回の申し込みで利用可能なのは一回のみです。

・表に掲載されている以外の日程でも募集を行うことがあります。最新情報は Web Page[1] をご覧ください。

[1]「大規模 HPC チャレンジ」
https://www.cc.u-tokyo.ac.jp/guide/hpc/

4. 研究課題

「大規模 HPC チャレンジ」では、提供する最大ノードを使用する大規模計算を実施する研究に限定します。申込者及び研究グループのメンバーは、国内外の並列計算機を利用した大規模計算の実績があることを前提とし、以下のような「High-Performance Computing」に関連した幅広い分野の研究を対象としています。

・大規模シミュレーション
・大規模データ処理
・大規模ベンチマーク、演算・通信システム性能評価
・その他、大規模計算に関係するソフトウェア実行

5. 利用資格

利用資格は、申込書を基に、東京大学情報基盤センタースーパーコンピューティング研究部門の教職員（Oakforest-PACS については筑波大学計算科学研究センターの教職員も含む）、および、外部委員により構成される審査委員会において審査されます。現ユーザーである必要はありません。

応募課題は、審査委員会により採択課題を選考し、できるだけ速やかに公表を行う予定です。

なお、申込者は「国内の大学、公共機関に所属する研究者、および民間企業に所属する者」とします。また、研究グループのメンバー又は申込者が企業の方の場合は、以下の書類のいずれかを提出していただく必要があります。

・「**共同研究契約書の写し**」
（申込者の所属機関と共同研究契約を結んでいる研究組織に所属する者）
・「**適切に監督を行うことを記した誓約書及び請負契約書の写し**」
（申込者の所属機関と請負契約を結んでいる企業の従業員）
・「**利用規定にある利用目的を遵守することを記した誓約書**」
（民間企業に所属している者）

6. 採択基準、審査方法

応募課題は、以下の基準により、東京大学情報基盤センタースーパーコンピューティング研究部門の教職員（Oakforest-PACS については筑波大学計算科学研究センターの教職員も含む）、および、外部委員より構成される審査委員会により採択課題を選考し、できるだけ速やかに公表を行う予定です。

主な採択基準

A) 計算・結果の詳細を論文等も含めて公表できること。
B) 計算結果が科学的に有用、あるいは社会的なインパクトがあると考えられること。
C) 提供する最大ノード数の利用を目標としていること。
D) 計画に実現性があり、短期間で効果を示すことが可能であること（一回の使用時間は最大 24 時間です）。
E) 本システムの運用、ユーザーにとって有用な情報を提供すること。

※ 項目 A) ～ D)は、必須となります。項目 E)は必須ではありませんが、申込書に該当する記述がある場合、加点評価される場合があります。

7. 利用申込

募集要項、スーパーコンピューターシステム利用規程等をよくお読みの上、利用申込書に必要事項をご記入ください。ご記入頂いた利用申込書は以下の提出先まで、電子メールに添付してお送りください。

申込書類に必要な項目は以下の通りです。

1. 申込年月日
2. 利用希望時期
3. 申込者情報（氏名、所属、職名、連絡先住所、E-mail、電話）
4. 研究課題名（和文、英文）、概要
5. 研究課題の内容、目標
6. 申込者、研究グループメンバーの当該分野における研究業績のうち、大規模計算機利用の実績として代表的な論文１編の別刷り（メール添付または郵送）

   ※別刷りを郵送される場合は、「大規模 HPC チャレンジ 別刷」と朱記し、申込書の 1 ページ目を印刷して同封ください。提出先は「問い合わせ先」をご覧ください。
7. プログラム情報、利用スケジュール等
8. 要望事項、特記事項
9. 研究グループメンバーの情報（氏名、所属、職名、研究課題における役割）

8. 問い合わせ先

利用申込等ご不明な点は、電子メールでお問い合わせください（電話でのお問い合わせはご遠慮ください。なお、詳細は本センターWeb Page[1] でもご案内しておりますので、あわせてご覧ください）。

申込書提出先：

E-Mail ： koubo@cc.u-tokyo.ac.jp

〒 113-8658

東京都文京区弥生 2-11-16

東京大学　情報システム部　情報戦略課　研究支援チーム

問い合わせ先：

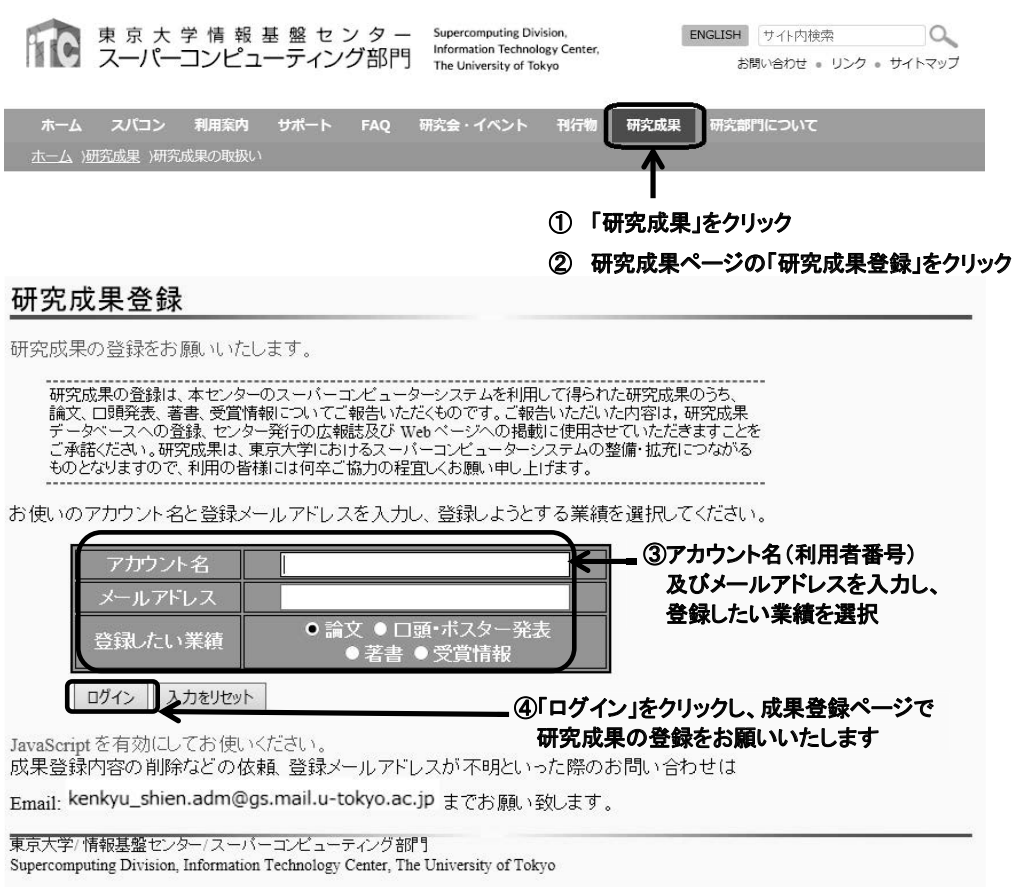
E-Mail ：uketsuke@cc.u-tokyo.ac.jp

研究成果の登録のお願い

情報戦略課研究支援チーム
情報基盤センタースーパーコンピューティング部門

研究成果の登録は、本センターのスーパーコンピューターシステムを利用して得られた研究成果のうち、論文、口頭発表、著書、受賞情報についてご報告いただくものです。研究成果の登録は、本センタースーパーコンピューティング部門のWebサイト（https://www.cc.u-tokyo.ac.jp/）から「研究成果登録」に進んでください。なお、ご報告いただいた内容は、研究成果データベースへの登録、本センター発行の広報誌及びWebページに掲載させていただきますので、ご了解ください。

研究成果は、東京大学におけるスーパーコンピューターシステムの整備・拡充につながるものとなりますので、利用者の皆様には何卒ご協力くださいますようお願い申し上げます。



6・7月のジョブ統計

1. Oakbridge-CX スーパーコンピュータシステムジョブ処理状況（Red Hat Enterprise Linux 7、CentOS 7）

年月	登録者数	実利用者数	処理件数				接続時間 [時間]	ファイル使用量 [GiB]		ログイン（実CPU）	演算時間 [ノード時間]（経過時間）			平均ノード利用数（ノード）	ノード利用率（%）
			ログイン	プリポスト	インタラクティブジョブ	バッチジョブ		/home	/lustre		プリポスト	インタラクティブジョブ	バッチジョブ		
2020年4月	1,092	147	5,802	20	789	16,434	24,146	647	16,881,212	1,745	20	354	233,086	379.9	27.9
5月	1,127	215	7,937	71	1,324	39,551	30,946	684	17,588,622	1,190	725	709	496,108	709.9	52.2
6月	1,298	217	8,669	156	1,242	132,825	39,950	1,000	24,529,271	2,674	627	936	607,896	898.9	66.1
7月	1,378	347	18,263	118	1,760	80,506	53,532	1,340	23,582,558	5,753	519	1,179	668,610	957.8	70.4
2019年8月	361	79	3,377	−	361	59,048	16,728	153	2,390,289	922.77	−	153	433,971	784.0	57.6
9月	337	83	4,634	14	301	91,667	25,096	99	4,135,511	1,591.27	12	83	488,727	843.3	62.0
10月	445	70	4,603	0	302	22,159	24,772	198	5,102,808	1,436.98	0	129	195,762	273.4	20.1
11月	480	95	4,759	2	379	20,085	34,108	185	5,891,232	3,037.14	1	162	261,169	372.9	27.4
12月	529	114	5,019	1	276	26,914	35,531	393	14,788,166	2,489.42	0	138	371,203	511.4	37.6
2020年1月	562	117	5,831	0	377	32,698	43,594	495	15,790,227	3,237.89	0	141	294,689	410.2	30.2
2月	587	115	5,299	0	326	20,493	32,540	610	17,504,878	1,158.40	0	181	414,952	620.0	45.6
3月	588	98	4,785	1	464	26,064	28,456	586	16,978,662	1,903.42	11	317	680,984	961.4	70.7
合計			78,978	383	7,901	568,444	389,399			27,139.93	1,915	4,482	5,147,157		

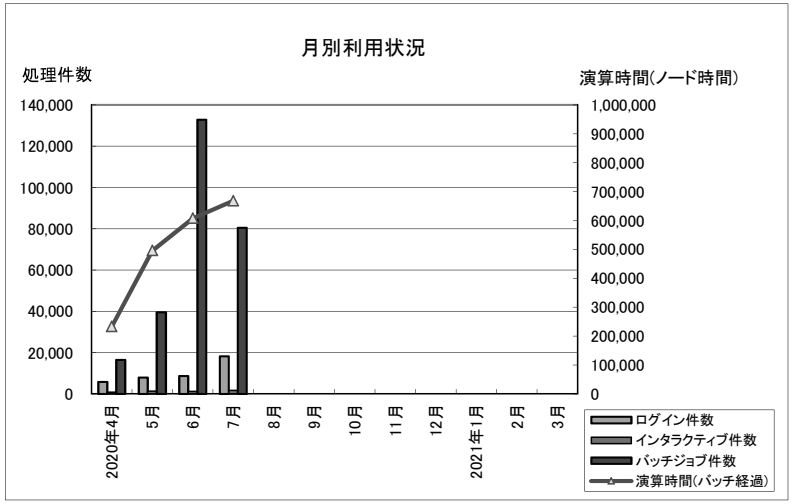
・試験運転は、2019年7月1日より開始。正式サービスは、2019年10月1日より開始

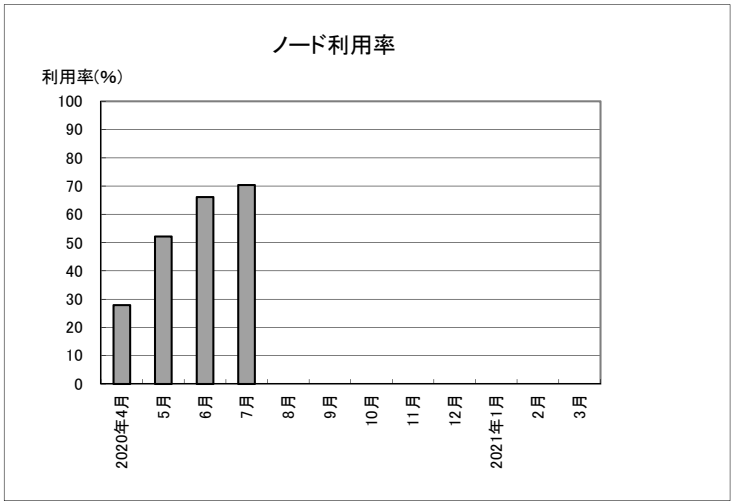
・接続時間： ログイン時間の累計

・ログイン（実CPU）： コア時間単位

・ノード利用数： インタラクティブおよびバッチジョブの経過時間を1ノードが100%動作したと仮定した場合の利用ノード数。
計算式＝1ヶ月のインタラクティブおよびバッチジョブ経過時間合計÷1ヶ月の稼動時間

・ノード利用率： サービスノードに対する利用比率。 計算式＝ノード利用数÷サービスノード数×100





2. Oakforest-PACSスーパーコンピュータシステムジョブ処理状況 （RedHat Enterprise Linux 7、CentOS 7）

年月	登録者数	実利用者数	処理件数				接続時間[時間]	ファイル使用量 [GiB]		ログイン（実CPU）	演算時間 [ノード時間]（経過時間）			平均ノード利用数（ノード）	ノード利用率（%）
			ログイン	プリポスト	インタラクティブジョブ	バッチジョブ		/home	/work		プリポスト	インタラクティブジョブ	バッチジョブ		
2020年4月	1,885	464	9,576	461	253	38,627	54,218	3,927	6,326,936	1,829.07	709	214	2,852,608	4,441.8	54.1
5月	1,802	446	9,554	247	254	59,638	64,493	3,220	5,455,455	2,698.20	502	152	3,577,511	4,894.1	59.6
6月	1,611	455	11,840	151	358	47,236	67,932	3,361	5,858,067	1,957.58	342	161	3,429,085	4,850.4	59.1
7月	1,622	410	12,928	343	327	52,585	64,608	3,446	5,996,751	2,262.33	492	163	4,121,263	5,909.8	72.0
2019年7月	1,576	423	13,070	321	411	83,528	89,282	3,043	4,437,448	22,625.24	496	131	3,674,404	5,025.8	61.2
8月	1,604	395	8,227	95	290	47,167	47,333	3,232	4,639,620	10,901.40	179	72	2,756,332	5,457.7	66.5
9月	1,666	468	10,804	249	1,338	46,191	60,950	3,377	4,839,917	15,410.42	341	215	4,112,815	6,426.3	78.3
10月	1,734	483	13,519	341	468	63,874	76,261	3,460	5,035,415	4,566.66	625	320	3,580,492	4,898.2	59.7
11月	1,616	456	13,103	457	208	51,790	103,507	3,683	5,277,714	9,000.08	956	151	4,248,874	6,009.4	73.2
12月	1,618	454	14,180	470	437	72,460	89,295	3,659	5,615,226	5,920.77	976	214	4,913,167	6,721.0	81.9
2020年1月	1,598	445	11,965	408	378	48,545	118,573	3,725	6,034,310	9,427.65	666	217	5,198,931	7,111.9	86.6
2月	1,629	424	10,772	359	401	47,509	78,624	3,850	6,253,958	12,649.16	768	192	4,977,581	7,321.7	89.2
3月	1,598	401	11,791	536	533	64,010	80,219	3,761	6,408,220	9,302.12	1,062	196	5,581,379	7,713.3	94.0
合計			138,259	4,117	5,245	639,632	906,013			85,925.44	7,618	2,267	49,350,038		

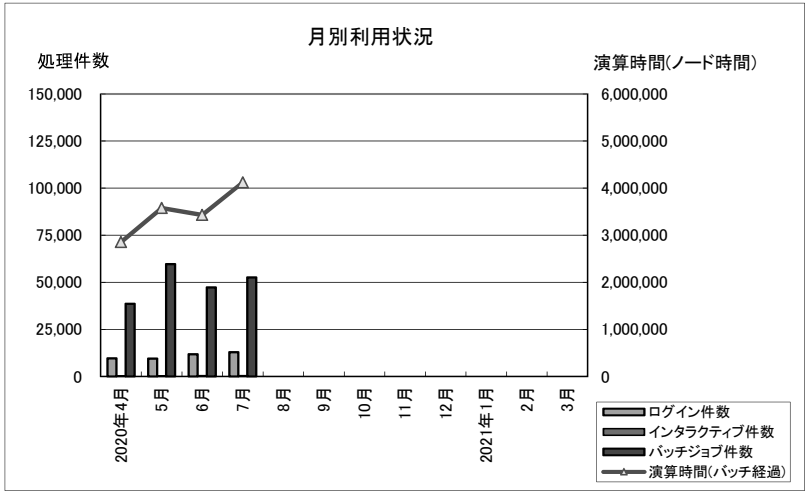
・接続時間： ログイン時間の累計

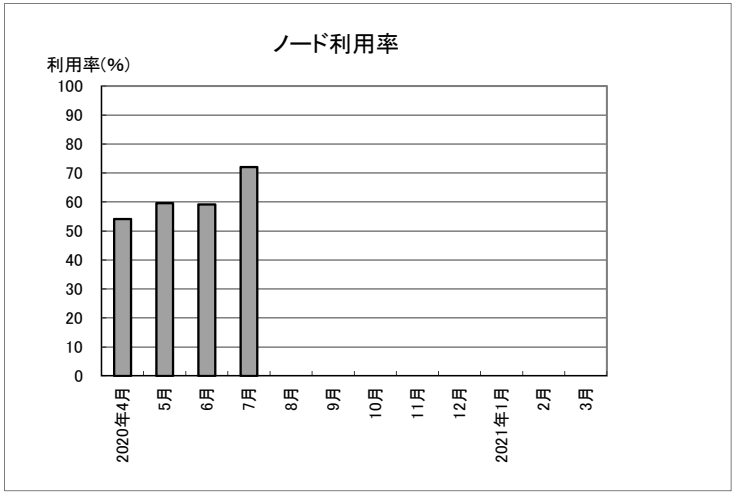
・ログイン（実CPU）： コア時間単位

・2019年7月分は合計に含まない

・ノード利用数： インタラクティブおよびバッチジョブの経過時間を1ノードが100%動作したと仮定した場合の利用ノード数。
計算式＝ノード利用率×総ノード数(8208)

・ノード利用率： サービスノードに対する利用比率。 計算式＝利用ノード時間÷サービスノード時間×100



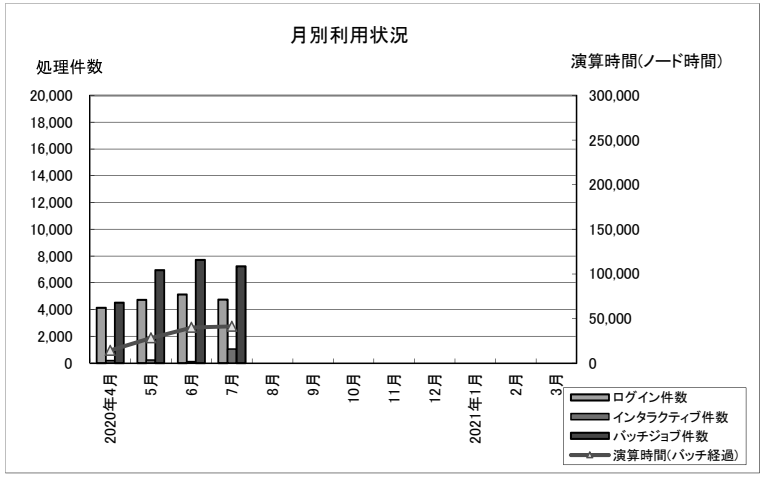


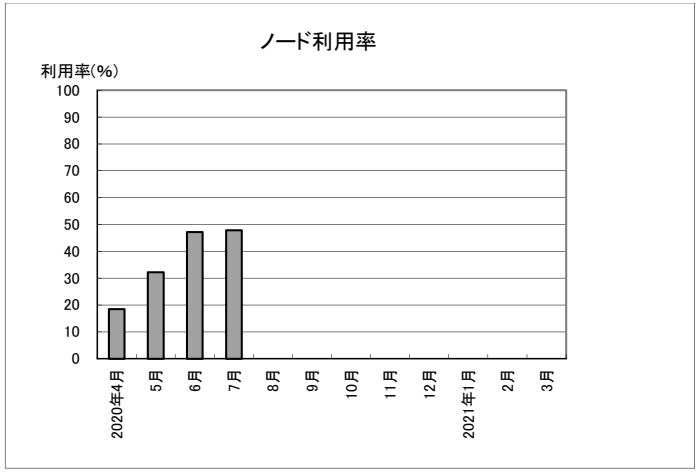
3. Reedbushスーパーコンピュータシステム（Reedbush-H）ジョブ処理状況 （RedHat Enterprise Linux 7）

22巻9月号よりReedbush-H/L/Uのジョブ処理状況の掲載順を変更しています。

年月	登録者数	実利用者数	処理件数				接続時間 [時間]	ファイル使用量 [GiB]		ログイン（実CPU）	演算時間 [ノード時間]（経過時間）			平均ノード利用数（ノード）	ノード利用率（%）
			ログイン	プリポスト	インタラクティブジョブ	バッチジョブ		/home	/lustre		プリポスト	インタラクティブジョブ	バッチジョブ		
2020年4月	1,287	239	4,124	1	181	4,512	5,558	169	435,177	6.74	0	165	14,192	22.1	18.5
5月	1,287	204	4,724	1	230	6,950	7,699	133	300,576	9.43	0	293	28,184	38.6	32.2
6月	1,032	235	5,136	2	110	7,720	7,073	145	306,587	5.06	0	29	39,952	56.2	47.2
7月	1,069	188	4,736	8	1,053	7,234	7,653	148	299,291	6.06	5	1,024	41,261	57.4	47.8
2019年7月	1,377	448	8,803	39	253	5,579	12,456	131	335,883	12.72	20	239	41,317	56.4	47.0
8月	1,356	262	5,850	0	72	5,576	8,173	138	361,872	6.43	0	59	31,020	55.4	46.2
9月	1,378	258	5,870	1	114	9,254	8,238	145	404,288	7.97	0	55	35,234	57.2	47.7
10月	1,256	312	7,778	0	661	15,397	11,357	153	423,068	12.62	0	493	46,355	70.5	58.8
11月	1,294	333	8,566	21	589	9,721	11,653	168	460,973	17.58	17	247	51,108	72.1	60.1
12月	1,254	324	8,563	0	472	6,545	13,520	176	520,955	18.00	0	386	41,445	56.8	47.4
2020年1月	1,241	316	9,461	1	787	15,949	15,181	186	541,491	13.84	0	537	57,031	78.2	65.2
2月	1,237	273	5,826	0	515	25,272	9,731	183	616,806	11.83	0	323	42,821	62.7	52.2
3月	1,217	227	5,185	0	158	3,650	8,216	184	604,794	30.69	0	138	51,619	71.0	59.2
合計			75,819	35	4,942	117,780	114,052			146.25	22	3,749	480,222		

・接続時間： ログイン時間の累計
・ログイン（実CPU）： コア時間単位
・2019年7月分は合計に含まない
・ノード利用数： インタラクティブおよびバッチジョブの経過時間を1ノードが100%動作したと仮定した場合の利用ノード数。
計算式＝1ヶ月のインタラクティブおよびバッチジョブ経過時間合計÷1ヶ月の稼動時間
・ノード利用率： サービスノードに対する利用比率。 計算式＝ノード利用数÷サービスノード数×100





4. Reedbushスーパーコンピュータシステム（Reedbush-L）ジョブ処理状況 （RedHat Enterprise Linux 7）

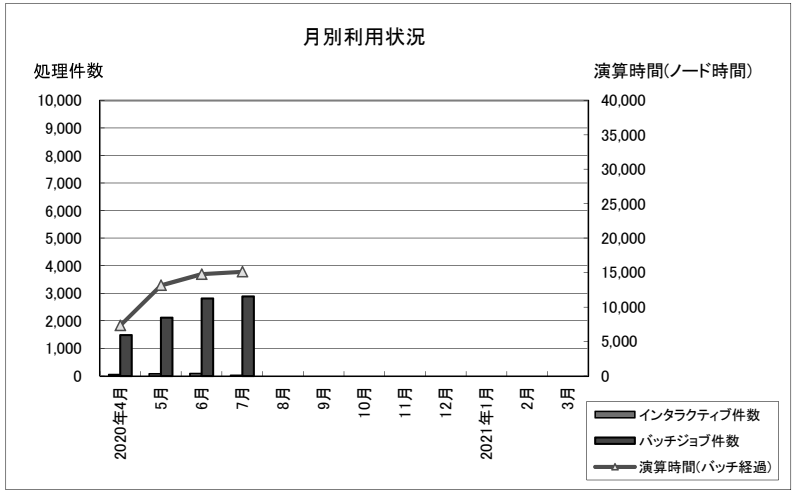
年月	処理件数		演算時間 [ノード時間](経過時間)		平均ノード利用数（ノード）	ノード利用率（%）
	インタラクティブジョブ	バッチジョブ	インタラクティブジョブ	バッチジョブ		
2020年4月	54	1,484	37	7,332	11.4	21.0
5月	80	2,115	74	13,149	17.9	33.2
6月	91	2,807	79	14,764	20.8	38.6
7月	21	2,883	27	15,123	20.6	38.1
2019年7月	99	1,625	82	7,517	10.3	18.8
8月	158	1,738	95	8,553	15.4	28.0
9月	55	1,474	49	5,542	9.1	16.5
10月	61	1,847	90	7,640	11.6	21.2
11月	118	2,083	91	21,247	30.0	54.5
12月	93	2,313	80	26,748	36.5	66.3
2020年1月	133	2,560	158	32,196	44.0	79.9
2月	56	2,657	211	28,072	41.1	74.7
3月	16	727	24	34,927	47.9	87.2
合計	936	24,688	1,015	215,293		

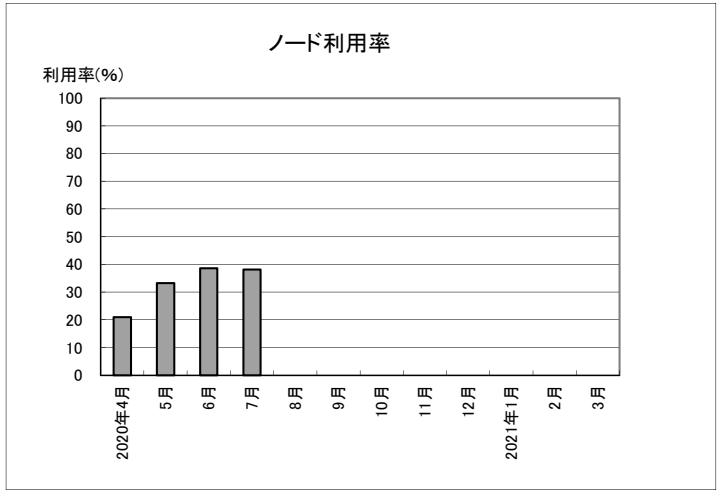
・登録者数、実利用者数、ログイン件数、接続時間、ファイル使用量、ログイン（実CPU）はReedbush-Uと共通。

・2019年7月分は合計に含まない

・ノード利用数： インタラクティブおよびバッチジョブの経過時間を1ノードが100%動作したと仮定した場合の利用ノード数。
計算式＝1ヶ月のインタラクティブおよびバッチジョブ経過時間合計÷1ヶ月の稼動時間

・ノード利用率： サービスノードに対する利用比率。 計算式＝ノード利用数÷サービスノード数×100





5. Reedbushスーパーコンピュータシステム（Reedbush-U） ジョブ処理状況 （RedHat Enterprise Linux 7）

2020年6月末を持ちまして、サービスを終了しました。

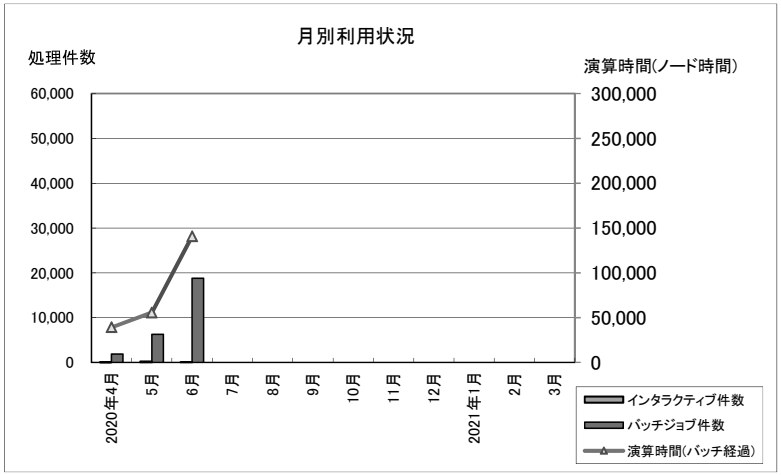
年月	処理件数		演算時間 [ノード時間]（経過時間）		平均ノード利用数（ノード）	ノード利用率（%）
	インタラクティブジョブ	バッチジョブ	インタラクティブジョブ	バッチジョブ		
2020年4月	60	1,890	22	39,032	60.0	14.0
5月	208	6,243	84	55,498	75.0	18.0
6月	14	18,806	3	140,699	199.6	47.5
7月	–	–	–	–	–	–
2019年7月	207	18,790	40	174,021	236.0	64.0
8月	221	8,738	61	121,728	217.1	59.0
9月	135	8,078	23	183,405	297.5	80.9
10月	305	10,210	82	184,602	277.9	75.5
11月	321	10,305	44	178,090	250.2	68.0
12月	240	12,750	36	203,904	277.1	75.3
2020年1月	325	12,649	23	227,303	308.9	83.9
2月	91	9,846	26	209,897	304.9	82.9
3月	131	11,844	45	216,745	297.4	80.8
合計	2,051	111,359	449	1,760,903		

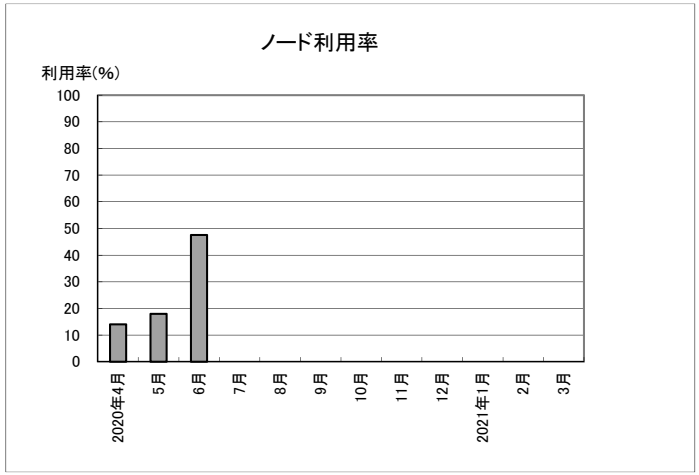
・登録者数、実利用者数、ログイン件数、接続時間、ファイル使用量、ログイン（実CPU）はReedbush-Uと共通。

・2019年7月分は合計に含まない

・ノード利用数： インタラクティブおよびバッチジョブの経過時間を1ノードが100%動作したと仮定した場合の利用ノード数。
計算式＝1ヶ月のインタラクティブおよびバッチジョブ経過時間合計÷1ヶ月の稼動時間

・ノード利用率： サービスノードに対する利用比率。 計算式＝ノード利用数÷サービスノード数×100





ゲリラ豪雨予報のリアルタイム実証実験

三 好 建 正

理化学研究所

１.「ゲリラ豪雨予測手法」の開発

8 月 25 日から 9 月 5 日にかけて、ゲリラ豪雨予報のリアルタイム実証実験を行っている。このために Oakforest-PACS の 1200 ノードを占有利用させていただき、ユーザーの皆様のご理解に心より感謝している。天気予報はスーパーコンピュータによる社会貢献の一つだが、先端研究では、計算機性能の向上によってどのような天気予報が可能となり、新たな価値を生むことができるかを切り拓く。今回のゲリラ豪雨予報実験は、未来の天気予報を切り拓く取り組みである。

今回の実験システムは、2013 年 10 月から開始した CREST ビッグデータ応用領域（領域総括：田中譲・北海道大学）の研究課題「「ビッグデータ同化」の技術革新の創出によるゲリラ豪雨予測の実証」（研究代表者：三好建正）にて着想し、開発に着手したものである。当時、2012 年夏に新たに開発された最新鋭のフェーズドアレイ気象レーダ（PAWR）[1]と 2012 年 9 月から共用を開始したスーパーコンピュータ「京」、さらに 2014 年に打ち上げが予定されていた新しい気象衛星ひまわり 8 号という複数の先端技術が幸運にも揃い、これらを組み合わせて天気予報に革新をもたらす「ビッグデータ同化」の技術革新を着想した。「ビッグデータ」が急速に興隆し席巻しているところでもあった。

「データ同化」とは、シミュレーション計算に実測データを取り込み同化させ、シミュレーションを現実世界と同期させる方法で、シミュレーションによる天気予報の根幹を成す。「ビッグデータ同化」システムは、30 秒ごとに雨雲を隈なくスキャンする PAWR のビッグデータを余すことなく活用するため、100m 四方のメッシュの高精細気象シミュレーションを行う。気象庁が運用する最も高解像度な局地モデル（LSM）は 2km 四方のメッシュのシミュレーションに 1 時間ごとに集められる観測データを同化する。これは世界の現業数値天気予報システムとして決して引けを取らないトップクラスのシステムである。また世界の先端研究で用いられるシステムでは、大まかに言って、1km 四方のメッシュのシミュレーションに最高で 15 分程度の頻度でレーダのデータを同化する。これらと比べ、構想した「ビッグデータ同化」システムは、100m 四方のメッシュで 30 秒ごとのデータ同化という、桁違いなものである。

この桁違いの 30 秒ごとの超高頻度データ同化は、これまでの経験が効かない新たなチャレンジで、様々な困難に直面した。実験をしてみると、通常雨雲が発達することのない成層圏に雨雲を作るなど、非現実的なシミュレーション結果に悩まされた。調べると、30 秒ごとに雨雲のデータが入ってくることで、シミュレーションの気象学的な状態を不安定化していた。データ同化における局所化手法の改善など、様々な工夫を重ね、開発を始めてから約 3 年後の 2016 年 8 月に「ゲリラ豪雨予測手法」[2]の開発を完了した（Miyoshi et al. 2016a; 2016b）。計算時間も「京」

---

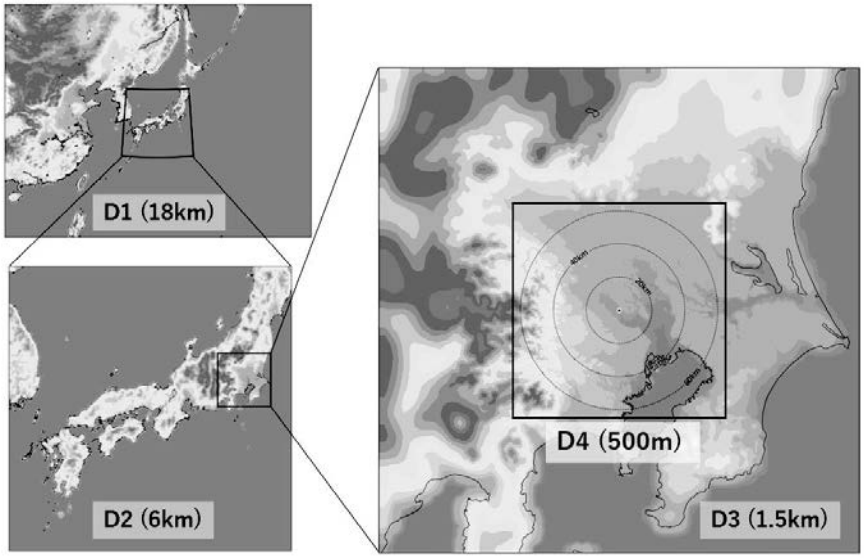
[1] 2012 年 8 月 31 日情報通信研究機構プレスリリース
(https://www.nict.go.jp/press/2012/08/31-1.html)

[2] 2016 年 8 月 9 日理化学研究所プレスリリース
(https://www.riken.jp/press/2016/20160809_1/)

コンピュータを使って当初 1 時間以上かかっていたところから、10 分までに短縮した。しかし、30 秒ごとのデータ同化を行うには、この計算時間を 30 秒以内に縮める必要があった。

２．今回のリアルタイム実証実験

前節で述べた CREST 研究課題は 2019 年 3 月に終了し、その後続課題「ビッグデータ同化と AI によるリアルタイム気象予測の新展開」（研究代表者：三好建正）が AIP 加速課題として採択された。2019 年 3 月までに、10 分かかっていた計算を 30 秒以内に縮めることに成功していたが、すべてのインプットデータが事前に用意されている条件下であった。真にリアルタイムの予報を行うには、観測データや、側面境界として与える気象状態（側面境界値）といったインプットデータをリアルタイムに作成する必要がある。2019 年 4 月以降これらの課題に取り組み、埼玉大学に設置されている新型のマルチパラメータ・フェーズドアレイ気象レーダ（MP-PAWR）[3]のデータをリアルタイムに Oakforest-PACS へ転送するネットワーク及び転送ソフトウェア JIT-DT を開発した。また、4 重に入れ子にした領域（第 1 図）を設定して、米国環境予測センターから入手する全球予報データを最も外側の側面境界値として与え、最も内側の 500m 四方のメッシュで計算するゲリラ豪雨予報の領域へとリアルタイムに実行するシステムを構築した。500m としたのは、占有利用するノード数を抑えることを意識しつつ、今の気象モデル技術、特に降水プロセ



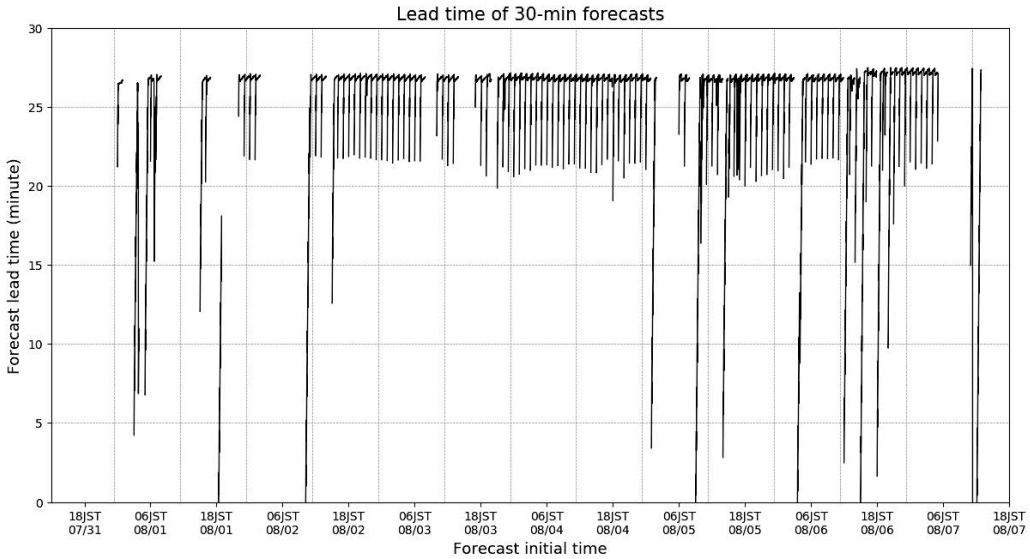
第 1 図: 4 重の入れ子に設定したシミュレーション計算領域。

左上図の領域 D1（解像度 18km）の内側に左下図の領域 D2（解像度 6km）、その内側に右図の領域 D3（解像度 1.5km）、さらにその内側に右図内枠の領域 D4（解像度 500m）を入れ子に設定し、最も内側の領域 D4 で 30 秒ごとに更新する予報を行う。マルチパラメータ・フェーズドアレイ気象レーダ（MP-PAWR）の設置場所（黒点、円の中心）と探知範囲 60km の円、またその内側に 40km、20km の円を示している。2020 年 8 月 21 日理化学研究所お知らせ[4]の図 1 より転載。

[3] 2017 年 11 月 29 日情報通信研究機構プレスリリース
（https://www.nict.go.jp/press/2017/11/29-1.html）
[4] https://www.riken.jp/pr/news/2020/20200821_1/index.html

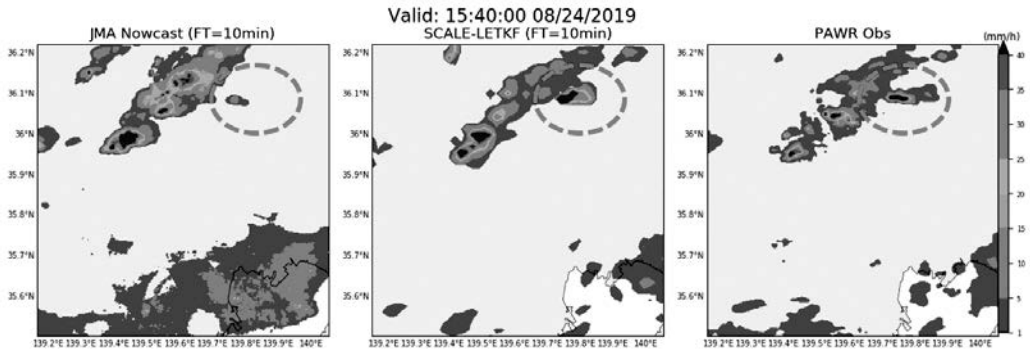
スに関わるモデリングの限界により、メッシュを細かくすることによる予報精度向上が計算コストに見合わないことを考慮した。このデータ同化にかかる計算は、Oakforest-PACS の 992 ノードを用いて 20 秒程度となった。同じ Oakforest-PACS の 992 ノードの中で、30 分後までの予報計算を行っている。今回の実証実験に先立って 7 月 31 日から 8 月 7 日にかけて実施したリアルタイムの準備実験で計ったところ、30 分後の予報結果の計算終了に約 3 分程度を要する（第 2 図)。つまり、30 分後の予報を計算するのに、約 3 分の処理時間を要し、実時間で約 27 分のリードタイムが取れる。

今回のシステムは、5 分、10 分といった短時間で急激に発達するゲリラ豪雨が予測できる。従



第 2 図: 7 月 31 日から 8 月 7 日の各予報初期時刻における予報リードタイムの時系列。

7 月 31 日 11:20 から 8 月 7 日 14:00（日本時間）の間の 13426 回の予報（4 日 15 時間 53 分稼働）のうち、ほとんどの場合で観測終了後約 3 分で 30 分予報の計算が終了している。



第 3 図: 2019 年 8 月 24 日 15:40 UTC（日本時間 25 日午前 0 時 40 分）における降水強度分布

(左)気象庁高解像度降水ナウキャストの 10 分後予測。(中)本研究の予報システムの 10 分後予測。(右)さいたま市での MP-PAWR 観測。2020 年 8 月 21 日理化学研究所お知らせの図 2 より転載。

来の予測手法では、これが予測できず、ゲリラ豪雨と呼ばれてきた。事前に行った過去事例（2019年8月25日未明）の実験では、第3図点線内にある10分以内に急激に発達した雨雲を、今回のシステムは良く予測した。第3図点線領域の左側にある雨雲は急速に弱まっているが、このような雨雲の弱まりもよく予測する。第3図には比較のため気象庁高解像度降水ナウキャストの予測も示しているが、ナウキャスト手法は基本的に雨雲の移動をトラッキングする手法であり、その性質上、雨雲の発生、発達、衰弱、消滅は予測しない。気象シミュレーションは、気象学的なメカニズムを考慮するため、移動だけではない変化を予測する。

３．まとめと展望

今回、30秒ごとに更新する30分後までの数値天気予報という、世界でも例がない唯一のリアルタイム天気予報を実施している。ここに至るには、2013年10月から継続してきた数値天気予報システム開発のほか、Oakforest-PACSの占有利用や、内閣府のSIP「レジリエントな防災・減災機能の強化」の施策として情報通信研究機構をはじめとする研究グループが世界で初めて開発したMP-PAWRの利用など、多方面にわたるご協力が必須であった。心より感謝申し上げたい。

今回の実証実験は、0を1にするという意味で、一つのマイルストーンを築いた。しかし、予測精度や安定的な実行などの面で課題がある。本実験終了後、まずは実験結果を十分に検証し、実用化に向けた課題を洗い出すことが重要である。これにより、1を100にしていく改善、高度化へと進んでいく。また経費をどう賄うのか、ビジネスモデルも開拓の必要がある。経費削減の観点では、近年急速に発達する機械学習の技術を応用することで、スーパーコンピュータの高解像度気象シミュレーションが生み出すビッグデータを教師データとするなど、HPCとビッグデータ、AI 技術の連携を深め、気象シミュレーションやデータ同化にかかる計算を削減することが考えられる。本研究を進めるAIP加速課題では、このような観点も含め、AIとビッグデータ同化の協奏を探っていく。

参　考　文　献

Miyoshi, T., M. Kunii, J. Ruiz, G.-Y. Lien, S. Satoh, T. Ushio, K. Bessho, H. Seko, H. Tomita and Y. Ishikawa, 2016: "Big Data Assimilation" Revolutionizing Severe Weather Prediction. Bull. Amer. Meteor. Soc., 97, 1347-1354. doi:10.1175/BAMS-D-15-00144.1

Miyoshi, T., G.-Y. Lien, S. Satoh, T. Ushio, K. Bessho, H. Tomita, S. Nishizawa, R. Yoshida, S. A. Adachi, J. Liao, B. Gerofi, Y. Ishikawa, M. Kunii, J. Ruiz, Y. Maejima, S. Otsuka, M. Otsuka, K. Okamoto and H. Seko, 2016: "Big Data Assimilation" toward Post-peta-scale Severe Weather Prediction: An Overview and Progress. Proc. of the IEEE, 104, 2155-2179. doi:10.1109/JPROC.2016.2602560

大規模多相流体解析向け省通信型マルチグリッド前処理付き共役勾配法

井戸村泰宏、小野寺直幸、山田進
日本原子力研究開発機構システム計算科学センター
山下晋
日本原子力研究開発機構原子力基礎工学研究センター
伊奈拓哉、今村俊幸
理化学研究所計算科学研究センター

1　はじめに

反復行列解法は大規模疎行列の連立一次方程式の解法として広く用いられており、そのスケーラビリティはエクサスケール計算における重要課題の一つとなっている。原子力分野では、原子炉における多相流体問題の数値流体力学（CFD: Computational Fluid Dynamics）解析にエクサスケール計算が必要となっている。このような大規模 CFD 解析では、反復行列解法に基づく陰解法ソルバが主要なコストを占め、コード全体のスケーラビリティと性能を左右する。クリロフ部分空間法 [1] は様々な分野で使用されている最も有力な反復行列解法の一つである。しかしながら、大規模クリロフソルバには大域的同期処理と収束特性劣化という 2 つの大きな課題が存在する。

クリロフ部分空間法では、各反復において基底ベクトルの生成と直交化が行われる。後者は内積処理を必要とし、大域的縮約通信とそれに伴う同期処理が発生する。最先端メニーコア環境では計算性能と通信性能のギャップが拡大しており、大域的同期処理の遅延が重要なボトルネックとなっている。この通信ボトルネックを緩和するために、省通信型（CA: Communication Avoiding）クリロフ部分空間法 [2–4] やパイプライン型クリロフ部分空間法 [5,6] といった通信削減手法が提案されてきた。前者は複数の基底ベクトルをまとめて計算することによって大域的同期処理を削減するのに対し、後者は非同期大域的通信によって同期処理のコストを隠蔽する。本研究では、CA クリロフ部分空間法を用いて大規模原子力 CFD 解析の強スケーリングの向上に取り組んできた [7–12]。

一方、近年の大規模 CFD 解析では問題のマルチスケール性が顕在化しており、行列の条件数が増大している。そのような悪条件問題では、クリロフソルバの反復回数が問題規模とともに増大する傾向にあり、前処理の高度化が必須となっている。マルチグリッド（MG: Multi-Grid）法はこのようなマルチスケール問題に対して最も有効な手法の一つであり、約 1 兆自由度に達する最近の大規模問題 [13,14] でも使われている。MG 前処理を用いた収束特性の向上により、同期処理を伴う反復計算の回数が大幅に削減される。しかしながら、この収束特性を維持しつつ MG 前処理における通信処理を削減することが残された課題となっている。

本稿では、2018 年度「大規模 HPC チャレンジ」において上記 2 つの課題解決に取り組んだ成果 [15] について概説する。この研究では 3 次元多相多成分熱流動 CFD コード JUPITER [16] 向けに省通信型マルチグリッド（CAMG）前処理を新たに開発し、従来の前処理付き共役勾配（P-CG: Preconditioned Conjugate Gradient）法やより進んだ CA クリロフ部分空間法との性能比較を Oakforest-PACS 上で行った。この結果、以下の成果が得られた。

- 前処理付きチェビシェフ反復（P-CI: Preconditioned Chebyshev Iteration）スムーザを用いて幾何的マルチグリッド（GMG: Geometric Multi-Grid）法に基づく新しい CAMG 前処理を開発した。P-CI スムーザにおいて、袖通信に混合精度処理を適用し、固有値計算に CA ランチョス法を用いることで CAMG 処理の省通信化を実現した。

- P-CG 法、前処理付き省通信型共役勾配（P-CACG）法、前処理付きチェビシェフ基底省通信型共役勾配（P-CBCG: Preconditioned Chebyshev Basis CACG）法、CAMG 前処理付き共役勾配（CAMGCG）法の計算性能と収束特性の系統的な比較結果が得られた。

- CAMGCG ソルバを約 900 億自由度の大規模多相流体解析に適用した。この結果、P-CG ソルバに比べて反復回数を約 1/800 に削減し、8,000 台の KNL に至る良好な強スケーリングを維持しつつ約 11.6 倍の高速化を達成した。

2　JUPITER コードにおけるクリロフソルバ

2.1　コード概要

JUPITER コード [16] は、非圧縮粘性流体を仮定した連続の式、ナビエ・ストークス方程式、エネルギー方程式によって原子炉内の溶融物の熱流動現象を記述する。燃料ペレット、燃料被覆管、チャンネルボックス、制御棒、炉内構造材から構成される多成分の気相、液相、固相の挙動を Volume of Fluid（VOF）関数の移流によって表現する。密度によって与えられる圧力ポアソン方程式の行列係数は気相と固相の密度比によって $\sim 10^7$ という極端なコントラストを示すために悪条件問題となる。このため、主要な計算コスト（約 90%以上）をポアソンソルバが占める。ポアソン方程式は直交座標系 (x, y, z) における 2 次精度中心差分（7 ステンシル）によって離散化される。この対称ブロック対角行列が与える連立一次方程式をクリロフソルバによって計算する。本研究のクリロフソルバは圧縮対角格納（CDS: Compressed Diagonal Storage）形式を採用している。メモリの間接参照を必要とする圧縮行格納（CSR: Compressed Sparse Row）形式と比較して、CDS 形式はブロック対角行列に対する効率的な直接メモリ参照を可能とする。並列化は (x, y, z) 方向の粗い 3 次元領域分割に MPI を適用し、各分割領域内における z 方向の細かい 1 次元領域分割に OpenMP を用いるハイブリッド並列モデルを採用する。この細かい分割領域の単位でスレッド並列処理を行えるように不完全 LU 分解に基づくブロックヤコビ（BJ: Block Jacobi）前処理 [1] を適用する。

2.2　P-CG 法

Algorithm 1 前処理付き共役勾配（P-CG）法

Input: $Ax = b$, Initial guess x_1

Output: Approximate solution x_i

1: $r_1 := b - Ax_1, z_1 = M^{-1}r_1, p_1 := z_1$
2: **for** $j = 1, 2, ...$ until convergence **do**
3:　Compute $w := Ap_j$
4:　$\alpha_j := \langle r_j, z_j \rangle / \langle w, p_j \rangle$
5:　$x_{j+1} := x_j + \alpha_j p_j$
6:　$r_{j+1} := r_j - \alpha_j w$
7:　$z_{j+1} := M^{-1} r_{j+1}$
8:　$\beta_j := \langle r_{j+1}, z_{j+1} \rangle / \langle r_j, z_j \rangle$
9:　$p_{j+1} := z_{j+1} + \beta_j p_j$
10: **end for**

オリジナルの JUPITER コードでは圧力ポアソン方程式を BJ 前処理に基づく P-CG 法で計算していた。Algorithm 1 に示す P-CG 法では 1 回の反復処理が疎行列ベクトル積（SpMV）、BJ 前処理、2 回の内積処理、3 回のベクトル処理（AXPY）によって構成される。ここで、反復毎に SpMV は袖通信を、内積処理は 2 回の大域的縮約通信（All_reduce）をそれぞれ必要とする。line 4 の All_reduce は残差ベクトルのノルムを含めて 2 要素を転送し、line 8 の All_reduce は 1 要素を転送する。

2.3　P-CACG 法

Algorithm 2 は P-CACG 法 [2, 7] を示す。P-CACG 法では主要な計算は外部ループにおける省通信ステップ数 s 回の SpMV（line 5）と BJ 前処理（line 6）、グラム行列を構築する GEMM 処理（line 7）、および、内部ループの 3 項間漸化式の計算に必要な 3 つのベクトル処理となる。ここで、GEMM 処理のサイズは s に依存するため、演算強度も s とともに増大する。キャッシュブロッキングの最適化を適用すると 3 項間漸化式の係数も s 回再利用できるため、3 つのベクトル処理の演算強度も s の拡大によって向上する。SpMV は P-CG 法と同様に内部反復毎に袖通信が必要となるが、グラム行列計算は外部反復に

Algorithm 2 前処理付き省通信型共役勾配（P-CACG）法

Input: $Ax = b$, Initial guess x_1

Output: Approximate solution x_i

1: $z_0 := 0, z_1 := b - Ax_1$
2: $q_0 := 0, q_1 := M^{-1}z_1$
3: **for** $k = 0, 1, 2, ...$ until convergence **do**
4: $\quad v_{sk+1} := z_{sk+1}$
5: $\quad$ Compute $\underline{V}_k$ $(v_{sk+1}, M^{-1}Av_{sk+1}, ..., (M^{-1}A)^s v_{sk+1})$
6: $\quad$ Compute $\underline{W}_k$ $(M^{-1}\underline{W}_k = \underline{V}_k)$
7: $\quad G_{k,k-1} := \underline{V}_k^* Z_{k-1}, G_{kk} := \underline{V}_k^* \underline{W}_k$
8: $\quad G_k = \begin{pmatrix} D_{k-1} & G_{k,k-1}^* \\ G_{k,k-1} & G_{kk} \end{pmatrix}$
9: $\quad$ **for** $j = 1$ to s **do**
10: $\quad\quad$ Compute d_{sk+j} that satisfies $Aq_{sk+j} = [Z_{k-1}, \underline{W}_k]d_{sk+j}$, $M^{-1}Aq_{sk+j} = [Q_{k-1}, \underline{V}_k]d_{sk+j}$
11: $\quad\quad$ Compute g_{sk+j} that satisfies $z_{sk+j} = [Z_{k-1}, \underline{W}_k]g_{sk+j}$, $q_{sk+j} = [Q_{k-1}, \underline{V}_k]g_{sk+j}$
12: $\quad\quad \mu_{sk+j} := g_{sk+j}^* G_k g_{sk+j}$
13: $\quad\quad \nu_{sk+j} := g_{sk+j}^* G_k d_{sk+j}$
14: $\quad\quad \gamma_{sk+j} := \mu_{sk+j}/\nu_{sk+j}$
15: $\quad\quad$ **if** $sk + j = 1$ **then**
16: $\quad\quad\quad \rho_{sk+j} := 1$
17: $\quad\quad$ **else**
18: $\quad\quad\quad \rho_{sk+j} := (1 - \frac{\gamma_{sk+j}}{\gamma_{sk+j-1}} \cdot \frac{\mu_{sk+j}}{\mu_{sk+j-1}} \cdot \frac{1}{\rho_{sk+j-1}})^{-1}$
19: $\quad\quad$ **end if**
20: $\quad\quad u_{sk+j} := [Q_{k-1}, \underline{V}_k]d_{sk+j}$
21: $\quad\quad y_{sk+j} := [Z_{k-1}, \underline{W}_k]d_{sk+j}$
22: $\quad\quad x_{sk+j+1} := \rho_{sk+j}(x_{sk+j} + \gamma_{sk+j}q_{sk+j}) + (1 - \rho_{sk+j})x_{sk+j-1}$
23: $\quad\quad q_{sk+j+1} := \rho_{sk+j}(q_{sk+j} + \gamma_{sk+j}u_{sk+j}) + (1 - \rho_{sk+j})q_{sk+j-1}$
24: $\quad\quad z_{sk+j+1} := \rho_{sk+j}(z_{sk+j} + \gamma_{sk+j}y_{sk+j}) + (1 - \rho_{sk+j})z_{sk+j-1}$
25: $\quad$ **end for**
26: **end for**

つき 1 回だけの All_reduce を行う。この All_reduce は $s(s+1)$ 要素の $G_{k,k-1}$ と $(s+2)(s+1)/2$ 要素の $G_{k,k}$ の上三角行列を転送する。これに加えて、収束確認のための残差ベクトル $r_{sk} = b - Ax_{sk+1}$ のノルム計算に外部反復あたり 1 回の All_reduce を必要とする。このため、P-CACG 法は外部反復あたり 2 回の All_reduce を必要とする。

2.4 P-CBCG 法

Algorithm 3 は P-CBCG 法 [4, 17] を示す。P-CBCG 法の主な計算コストは SpMV と BJ 前処理を含むチェビシェフ基底ベクトル生成（line 10）と残りの行列計算により与えられる。line 10 の SpMV は s 回の袖通信を必要とするのに対し、line 5、6、11 の行列計算では大域的縮約通信が行われる。このため、P-CBCG 法では s ステップにつき 2 回の All_reduce を必要とする。line 5、6 の All_reduce は $s(s+1)/2$ 要素の $Q_k^* A Q_k$ の上三角行列、s 要素の $Q_k^* r_{sk}$、および、1 要素の残差ベクトルノルムを転送し、line 11 の All_reduce は s^2 要素の $Q_k^* A S_{k+1}$ を転送する。

2.5 CAMGCG 法

Algorithm 1 の P-CG 法において BJ 前処理を CAMG 前処理で置き換える。ここで、JUPITER コードは直交格子系に基づくため、標準的な V サイクルの GMG 法 [1] を用いる（図 1）。スムーザとして BJ 前

Algorithm 3 前処理付きチェビシェフ基底省通信型共役勾配（P-CBCG）法

Input: $Ax = b$, Initial guess x_0

Output: Approximate solution x_i

$r_0 := b - Ax_0$
Compute S_0 $(T_0(AM^{-1})r_0, ..., T_{s-1}(AM^{-1})r_0\ )$
$Q_0 = S_0$
for $k = 0, 1, 2, ...$ until convergence **do**
  Compute $Q_k^* A Q_k$
  Compute $Q_k^* r_{sk}$
  $a_k := (Q_k^* A Q_k)^{-1} Q_k^* r_{sk}$
  $x_{s(k+1)} := x_{sk} + Q_k a_k$
  $r_{s(k+1)} := r_{sk} - AQ_k a_k$
  S_{k+1} $(T_0(AM^{-1})r_{s(k+1)}, ..., T_{s-1}(AM^{-1})r_{s(k+1)})$
  Compute $Q_k^* A S_{k+1}$
  $B_k := (Q_k^* A Q_k)^{-1} Q_k^* A S_{k+1}$
  $Q_{k+1} := S_{k+1} - Q_k B_k$
  $AQ_{k+1} := AS_{k+1} + AQ_k B_k$
end for

Algorithm 4 前処理付きチェビシェフ反復（P-CI）法

Input: $Ax = b$, Initial guess x_0, Approximate minimum/maximum eigenvalues of $A\tilde{M}^{-1}$, $\lambda_{\min}, \lambda_{\max}$

Output: Approximate solution x_i

$d := (\lambda_{\max} + \lambda_{\min})/2, c := (\lambda_{\max} - \lambda_{\min})/2$
$r_0 := b - Ax_0, z_0 := \tilde{M}^{-1} r_0, p_0 := z_0, \alpha_0 := 1/d$
for $i = 1, 2, ...$ until convergence **do**
  $x_i := x_{i-1} + \alpha_{i-1} p_{i-1}$
  $r_i := b - Ax_i$
  $z_i := \tilde{M}^{-1} r_i$
  $\beta_i := (\alpha_{i-1} c/2)^2$
  $\alpha_i := 1/(d - \beta_i/\alpha_{i-1})$
  $p_i := z_i + \beta_i p_{i-1}$
end for

処理を用いた P-CI 法 [18] を実装するが、この手法は内積処理を全く含まない。P-CI 法ではチェビシェフ多項式を用いて解ベクトルを向上することにより、反復法の収束特性を加速するが、チェビシェフ多項式の計算に前処理行列 $A\tilde{M}^{-1}$ の最小・最大固有値の計算が必要となる（Algorithm 4）。本研究ではこの固有値計算にニュートン基底を用いた前処理付き CA ランチョス法 [2] を用いた。また、収束確認のための大域的縮約通信を回避するためにスムーザの反復回数は初期の収束特性スキャン結果に基づき固定値を設定した。

CAMGCG 法において、計算強度の向上と袖通信の削減を図るために混合精度処理を採用した。ここで、精度の選択は各アルゴリズムの数値的特性に応じて最適化した。CG 法では収束極限近傍での残差ベクトル計算に倍精度が必要となる。CA ランチョス法では CA クリロフ部分空間法と同様に複数基底ベクトルの生成と直交化をまとめて処理するが、これらの基底ベクトルの線形独立性は丸め誤差に敏感であるため倍精度を必要とする。一方、P-CI スムーザは典型的に数 10 回の反復で得られる近似解を計算するため、単精度を利用可能である。このため GMG 前処理を単精度で処理し、CA ランチョス法と CG 法を倍精度で計算した。

CAMGCG 法の主要な計算コストは SpMV、BJ 前処理、および、3 回の AXPY で構成される P-CI 法が与える。これらは All_reduce を必要とする内積処理以外は P-CG 法の計算カーネルとほぼ同様である。このため、P-CI スムーザは P-CG ソルバをわずかに修正するだけで容易に実装できる。

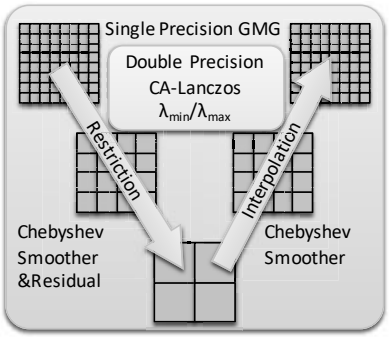


図 1: V サイクルの前処理付きチェビシェフ反復スムーザに基づく混合精度幾何的マルチグリッド前処理。固有値計算は省通信型ランチョス法によって行う。

3 カーネル性能評価

本節では P-CG、P-CACG、P-CBCG、および、CAMGCG ソルバの計算カーネル性能を分析する。ここでは小規模テスト問題サイズ $N = 104 \times 104 \times 265$ を KNL1 ノードで処理する。この問題規模は大規模問題におけるプロセッサあたりの典型的な問題規模に相当する。コンパイラやスレッドの設定は 4 節に示す数値実験と同じ条件を用いている。KNL における各カーネルの実行性能を修正ルーフラインモデル [19] と比較する。このモデルでは各カーネルの理論的な処理時間を浮動小数点演算とメモリアクセスの処理時間の和 $t_R = f/F + b/B$ によって評価する。ここで、f と b はカーネル中の浮動小数点演算数とメモリアクセス数を示し、F と B はプロセッサのピーク演算性能と STREAM メモリバンド幅を示す。

3.1 P-CG ソルバ

P-CG 法の主要計算カーネルは SpMV（line 3、4）、BJ（line 7、8）、および、Vector（AXPY、line 5、6、9）によって与えられる。ここで、SpMV と BJ に内積処理が含まれる。各カーネルのソースコードからカウントした浮動小数点演算数 f、メモリアクセス数 b、および、演算強度 f/b を表 1 にまとめる。圧力ポアソン方程式は 2 次精度中心差分を用いて計算するため、SpMV と BJ は比較的低い演算強度 $f/b < 0.2$ となる。さらに、残りの AXPY は $f/b = 0.1$ というメモリアクセスが支配的なカーネルとなる。このため、KNL の高いメモリバンド幅は P-CG ソルバの性能向上に大きく貢献する。Vector における AXPY はルーフラインモデルとの性能比が $t_R/t \sim 0.84$ というほぼ理想的な実行性能を達成しているが、SpMV と BJ におけるステンシル計算は $t_R/t \sim 0.6$ という性能劣化を示す。

3.2 P-CACG ソルバ

P-CACG 法の主要計算カーネルは SpMV（line 5、6）、BJ（line 5、6）、Gram（line 7、8）、および、3-term（line 20-24）となる。Gram と 3-term の演算強度は s に比例するため、P-CACG 法の演算強度は s に依存して変化する。ここで、Gram は $f = 2(s+1)(2s+1)/s$ および $b = 8(3s+2)/s$ とスケールし、3-term はキャッシュブロッキング最適化によって $f = (8s^2 + 12s + 2)/s$ および $b = 48(s+2)/s$ と変化する。表 1 では $s = 3$ のカーネル性能をまとめている。この省通信ステップ数は 4.2 節のベンチマーク問題が収束する上限値となっている。P-CACG 法における SpMV と BJ は内積処理を含まないため、P-CG 法よりも低い f と b を与える。Gram と 3-term では f が大きく増大するが、高い演算強度 $f/b > 0.5$ のために、P-CG 法からの処理時間の増大（約 1.53 倍）は f の増大（約 2.24 倍）に比べて小さい。

3.3 P-CBCG ソルバ

P-CBCG 法の主要計算カーネルはチェビシェフ基底計算 CB（line 10）と残りの行列計算 Matrix となる。P-CACG 法と同様に P-CBCG 法の演算強度も s に依存する。CB は $f = 2(9s+4)/s$ および $b = 8(4s+35)/s$

表 1: KNL1 ノードを用いたカーネル性能評価。ここで、各変数の定義は浮動小数点演算数 f [Flop/grid]、メモリアクセス数 b [Byte/grid]、演算強度 f/b、ピーク演算性能 F [Flops]、STREAM メモリバンド幅 B [Byte/s]、ルーフライン時間 $t_R = f/F + b/B$ [ns/grid]、処理時間 t [ns/grid]、実効性能 P [GFlops] となる。単精度（SP: Single Precision）の場合には F は倍精度（DP: Double Precision）の 2 倍となる。

Solver	Kernel	f	b	f/b	t	P	t_R/t
P-CG	SpMV	15.0	80.0	0.19	0.28	52.8	0.60
	BJ	20.0	128.0	0.16	0.46	43.3	0.59
	Vector	4.0	40.0	0.10	0.10	39.7	0.84
	Total	39.0	248.0	0.16	0.85	46.0	0.62
P-CACG ($s = 3$)	SpMV	13.0	80.0	0.16	0.24	54.6	0.72
	BJ	14.0	120.0	0.12	0.44	31.9	0.58
	Gram	18.7	29.3	0.64	0.13	142.9	0.51
	3term	41.7	80.0	0.52	0.49	84.2	0.36
	Total	87.3	309.3	0.28	1.30	67.0	0.52
P-CBCG ($s = 12$)	CB	30.6	228.7	0.13	0.89	34.5	0.55
	Matrix	93.2	83.3	1.12	0.63	147.1	0.32
	Total	123.8	312.0	0.40	1.52	81.4	0.45
P-CI (DP)	SpMV	14.0	88.0	0.16	0.33	42.0	0.56
	BJ	14.0	104.0	0.13	0.44	31.7	0.50
	Vector	4.0	40.0	0.10	0.10	38.4	0.81
	Total	32.0	232.0	0.14	0.88	36.4	0.56
P-CI (SP)	SpMV	14.0	44.0	0.32	0.24	57.5	0.39
	BJ	14.0	52.0	0.27	0.36	38.8	0.31
	Vector	4.0	20.0	0.20	0.05	75.4	0.80
	Total	32.0	116.0	0.28	0.66	48.7	0.38

とスケールし、Matrix は $f = (7s + 2)(s + 1)/s$ および $b = 40(2s + 1)/s$ と変化する。表 1 では $s = 12$ のカーネル性能をまとめている。この省通信ステップ数は 4.2 節のベンチマーク問題で用いられている。P-CBCG 法の浮動小数点演算数 f はさらに P-CG 法の約 3.17 倍まで増大するが、演算強度の向上によって処理時間の増大は約 1.79 倍に抑制されている。

3.4 CAMGCG ソルバ

CAMGCG 法の主要計算カーネルは SpMV（line 5）、BJ（line 6）、および、Vector（AXPY、line 4、9）を含む P-CI スムーザとなる。CAMGCG 法の数値特性は P-CG 法とほぼ同じとなり、処理時間も同程度となる。しかしながら、P-CI スムーザに単精度処理を適用するとメモリアクセス数 b が半分となり、ピーク演算性能 F が 2 倍となる。このため、ルーフラインモデルによると約 2 倍の性能向上が期待される。表 1 では Vector は約 2 倍の性能向上を示すが、SpMV と BJ の性能向上はそれを下回る。この結果、倍精度演算と比較した処理速度の向上は約 1.33 倍にとどまる。

4 数値実験

Oakforest-PACS において数値実験を実施した。Oakforest-PACS は 8,208 台の Intel Xeon Phi 7250 プロセッサ（KNL、68 コア、F=3046GFlops、B=480GB/s）および OmniPath（fat tree トポロジ、12.5GB/s）から構成される。コンパイラは Intel compiler 17.0.4（-O3 -qopenmp -xMIC-AVX512）および Intel MPI library 2017 を使用した。並列処理は 1 ノードあたり 1MPI プロセス、64 スレッドで行い、ハイパース

表 2: P-CG、P-CACG（$s = 3$）、P-CBCG（$s = 12$）、CAMGCG（SSOR/P-CI スムーザ）ソルバの収束特性。問題サイズ $N = 800 \times 500 \times 3,540 \sim 1.4 \times 10^9$ に対する収束特性と 500KNL を用いた時間ステップあたりの処理時間を示す。ここで、s は省通信ステップ数、ω は SSOR スムーザの加速パラメータを示す。CAMGCG ソルバは 5 レベルの V サイクルを使用し、SSOR スムーザと P-CI スムーザはそれぞれ各レベルで 30 回の内部反復を行った。

Solver	iterations/step	sec/step
P-CG	7063	34.1
P-CACG($s = 3$)	7065	32.8
P-CBCG($s = 12$)	7080	39.6
CAMGCG(SSOR, $\omega = 1.2$)	143	110.3
CAMGCG(SSOR, $\omega = 1.0$)	133	104.3
CAMGCG(SSOR, $\omega = 0.8$)	146	116.8
CAMGCG(P-CI, double precision)	33	14.5
CAMGCG(P-CI, mixed precision)	33	9.8

レッドは使用しなかった。また、MCDRAM（16GByte）と DDR4（96GByte）から構成される階層的なメモリをキャッシュモードで使用した。

4.1 収束特性

この数値実験では原子炉の燃料集合体 1 体における溶融デブリの非線形発展を計算した。問題サイズは $N = 800 \times 500 \times 3,540 \sim 1.4 \times 10^9$ とした。この問題サイズは先行研究 [7, 9, 16] でも使われた。P-CG、P-CACG、P-CBCG、および、CAMGCG ソルバの収束特性を表 2 にまとめる。ここで、収束条件は相対残差 $|b - Ax|/|b| < 10^{-8}$ によって与えた。文献 [7, 9] に従い、P-CACG および P-CBCG ソルバの省通信ステップ数はそれぞれ $s = 3$ および $s = 12$ とした。P-CACG は丸め誤差の影響により $s > 3$ で破綻するが、P-CBCG ではチェビシェフ基底を用いることでこのような数値不安定性が解決されており、$s > 12$ でも安定して収束する。ここでは、基底ベクトルを格納するメモリ使用量の制限によって $s = 12$ と選んだ。CAMGCG ソルバでは P-CI 法および赤黒オーダリングに基づく対称逐次加速緩和（SSOR: Symmetric Successive Over-Relaxation）法 [1] という 2 つの異なるスムーザアルゴリズムを用いて GMG 前処理を計算した。レベル数は最も粗いレベルの格子数が 10^6 以下となるように選んだ。各レベルのスムーザの並列化には最も細かい元のグリッドにおける (x, y, z) 方向の 3 次元領域分割モデルと同じ領域分割モデルおよび袖通信を用いた。この場合、レベルを粗くする毎に計算と袖通信はそれぞれ $\sim 1/2^3$ および $\sim 1/2^2$ に減少する。上記問題サイズに対し、 CAMGCG ソルバは 5 レベルの V サイクルを使用し、 各レベルの P-CI スムーザおよび SSOR スムーザの内部反復数は 30 回に固定した。

表 2 において、P-CACG ソルバと P-CBCG ソルバは P-CG ソルバとほぼ同じ反復回数となっている。これは、これらのアルゴリズムは数学的に等価であり、丸め誤差の影響による収束特性劣化が生じないように省通信ステップ数を選べば、同様の収束特性となることが期待されるためである。一方、CAMGCG ソルバは収束特性を大幅に向上する。 SSOR スムーザおよび P-CI スムーザを用いた CAMGCG ソルバは反復回数をそれぞれ $\sim 1/50$ および $\sim 1/200$ に削減する。これにより All_reduce の回数も大幅に削減される。しかしながら、SSOR スムーザを用いた CAMGCG ソルバは非効率なストライドメモリアクセスと赤黒で 2 回の袖通信が必要となるために全体処理時間が P-CG ソルバよりも長くなる。P-CI スムーザを用いた CAMGCG ソルバでは SSOR スムーザと比較して反復回数が $\sim 1/4$ となり、計算と通信の両方が削減される。これに加え、混合精度処理を適用することでさらに処理性能が加速される。ここで、単精度処理の P-CI スムーザによる収束特性劣化は全く見られない。

JUPITER コードにおいては、ポアソンソルバの行列係数が密度分布に依存して時間変化する。このため、P-CI スムーザは時間ステップ毎に CA ランチョス法による固有値計算を必要とする。固有値ソルバにおいて、最大・最小固有値は典型的には 10 回程度の反復で収束するが、今回の実装ではノルムの計算

による All_reduce を必要とする収束条件を検証せずに十分な精度を保証するために反復回数を 100 回（省通信ステップ数 $s = 5$）に固定した。この結果、固有値ソルバの処理時間は約 0.7 秒となったが、これは全体コストの 10%以下である。ここで、省通信型でないランチョス法を用いた場合には、固有値ソルバの処理時間は約 2 倍になる。以降のスケーリングテストでは P-CI スムーザに基づく混合精度 CAMGCG ソルバを使用する。

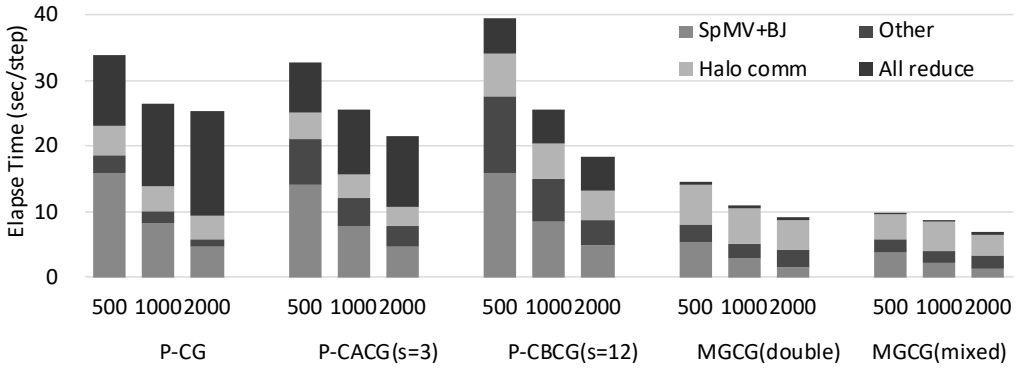


図 2: 500、1,000、2,000KNL を用いた P-CG、P-CACG（$s = 3$）、P-CBCG(s=12)、および、CAMGCG（倍精度/混合精度）ソルバ の強スケーリング。問題サイズ $N = 800 \times 500 \times 3,540 \sim 1.4 \times 10^9$ に対する 1 時間ステップあたりの処理時間のコスト分布を示す。

表 3: 図 2 におけるコスト分布（sec/step）。

Solver	Node	SpMV +BJ	Other	Halo Comm.	All Reduce	Total
P-CG	500	16.0	2.7	4.6	10.8	34.0
	1000	8.3	1.7	4.0	12.6	26.5
	2000	4.6	1.2	3.5	16.1	25.4
P-CACG ($s = 3$)	500	14.1	6.9	4.2	7.6	32.8
	1000	7.7	4.2	3.6	9.9	25.6
	2000	4.7	3.0	3.1	10.6	21.4
P-CBCG ($s = 12$)	500	15.8	11.7	6.7	5.3	39.6
	1000	8.4	6.6	5.4	5.2	25.7
	2000	4.8	4.0	4.4	5.2	18.4
MGCG (double)	500	5.3	2.8	6.1	0.3	14.5
	1000	2.9	2.2	5.4	0.4	11.0
	2000	1.6	2.7	4.4	0.5	9.2
MGCG (mixed)	500	3.8	2.0	3.9	0.1	9.8
	1000	2.2	1.8	4.3	0.3	8.8
	2000	1.2	2.2	3.2	0.3	6.9

4.2 14億自由度の強スケーリングテスト

P-CG、P-CACG、P-CBCG、および、CAMGCG ソルバの強スケーリングテスト結果を図 2 と表 3 にまとめる。この強スケーリングテストでは 500、1,000、2,000KNL を用いた。P-CG ソルバは All_reduce の影響による強スケーリングの大きな劣化を示す。P-CACG ソルバ（$s = 3$）は All_reduce のコストを削減し、強スケーリングを向上させる。しかしながら、 All_Reduce のコスト削減率は理論的評価 $1/s$ を大幅に下回り、2,000KNL における P-CG ソルバからの性能向上は約 1.18 倍にとどまる。 P-CBCG ソルバ（$s = 12$）は P-CACG ソルバより良い強スケーリングを示す。しかしながら、All_Reduce のコスト削減率はやはり $1/s$ を大幅に下回り、計算カーネルのコストも P-CACG ソルバより大きい。この結果、2,000KNL における P-CG ソルバからの性能向上は約 1.38 倍となる。CAMGCG ソルバの計算コストは最も細かいレベルの P-CI スムーザによって占められる。これは $30 \times 2 \times 33 = 1,980$ 回呼ばれるため、SpMV のコストは反復回数が約 7,000 回の P-CG 法に比べて $\sim 1/3$ となる。All_reduce の回数が約 7,000 回から 33 回に減少するため、主要な通信コストは P-CI スムーザの袖通信によって与えられる。今回の強スケーリングテストでは分割領域サイズを z 方向に 1/2 および 1/4 と細分化したため、MPI プロセスあたりの袖通信データサイズは 500KNL からそれぞれ 2/3 および 1/2 に減少する。しかしながら、袖通信のスケーリングはこの評価を下回る。この特徴は全てのソルバで見られるが、CAMGCG ソルバでは袖通信の影響が相対的に大きいため、強スケーリングが他のソルバより悪化する。ただし、全体処理時間自体は他のソルバに比べて大幅に短縮されている。混合精度処理を適用することで計算と通信の両方が加速され、倍精度処理と比べて約 1.34 倍の性能向上が得られる。この結果、2,000KNL における P-CG ソルバと混合精度 CAMGCG ソルバの全体処理性能比は約 3.7 倍となる。

4.3 900億自由度の強スケーリングテスト

本節では前節で使用したものと同じ多相流体問題において全方向に解像度を 4 倍した大規模問題 $N = 3,200 \times 2,000 \times 14,160 \sim 9.06 \times 10^{10}$ の数値実験を示す。この問題では適合型時間ステップ幅も自動的に 1/4 となる。密度分布はより一層シャープなコントラストを示すため、行列の条件数が増大する。これにより P-CG ソルバの収束特性は大幅に劣化し、反復回数は 26,475 回に達する。一方、CAMGCG ソルバは混合精度処理、倍精度処理の両方の場合において 32 回の反復で収束し、All_reduce の回数が P-CG ソルバの $\sim 1/800$ に減少する。ここで、CAMGCG ソルバは 7 レベル（各方向 4 倍のグリッド数のため 2 レベル増加）の V サイクルを使用し、各レベルで 50 内部反復を処理する。

強スケーリングテストにおいて、Oakforest-PACS 全系規模まで P-CG ソルバと混合精度 CAMGCG ソルバのテストを実施した。図 3 および表 4 に 2,000、4,000、8,000KNL における 2 つのソルバのコスト分布を示す。 2,000KNL における P-CG ソルバのコスト分布は、14 億自由度の問題と比較するとプロセスあたりの問題サイズが 64 倍になるため、より計算コストの比率が高くなり、強スケーリングが向上する。しかしながら、8,000KNL においては、コスト分布の約半分が All_reduce で占められる。CAMGCG ソルバでは最も細かいグリッドレベルにおける P-CI スムーザの SpMV は $50 \times 2 \times 32 = 3,200$ 回呼ばれる。この数字は P-CG ソルバの反復回数より一桁小さく、SpMV のコストは $\sim 1/10$ に減少する。しかしながら、2,000KNL では全てのレベルを含めた袖通信コストは P-CG ソルバの約 2 倍となる。このコスト増大は 2,000KNL のみで発生し、4,000KNL および 8,000KNL における袖通信コストは P-CG ソルバより小さくなる。このような通信性能劣化はメモリ使用量が MCDRAM のサイズを超える場合にしばしば見られ、その原因としてはキャッシュモードにおけるダイレクトマップキャッシュでのライン競合の可能性が考えられる。CAMGCG ソルバも計算量の増大によって 14 億自由度の問題よりも良い強スケーリングを示し、4,000KNL および 8,000KNL において、それぞれ 2,000KNL の約 2.6 倍および約 4.2 倍の性能向上を示す。P-CG ソルバと CAMGCG ソルバの性能差は 14 億自由度の問題よりも大幅に拡大し、8,000KNL で約 11.6 倍の性能向上を示す。

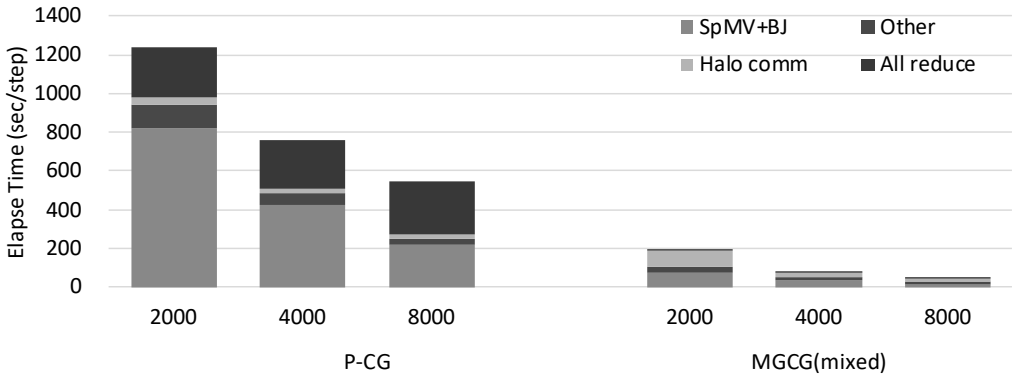


図 3: 2,000、4,000、8,000KNL を用いた P-CG ソルバと混合精度 CAMGCG ソルバの強スケーリング。問題サイズ $N = 3,200 \times 2,000 \times 14,160 \sim 9.06 \times 10^{10}$ に対して 1 時間ステップあたりの処理時間のコスト分布を示す。

表 4: 図 3 におけるコスト分布（sec/step）。

Solver	Node	SpMV +BJ	Other	Halo Comm.	All Reduce	Total
P-CG	2000	821.1	117.1	37.9	265.4	1241.6
	4000	424.8	63.0	22.9	247.3	758.1
	8000	218.3	32.6	21.4	270.8	543.1
MGCG (mixed)	2000	77.3	25.2	89.9	4.6	197.0
	4000	39.7	13.7	20.5	2.1	76.0
	8000	17.9	9.6	17.2	1.9	46.7

5　まとめ

本稿では JUPITER コードにおける圧力ポアソン方程式用に開発された新しい CAMGCG ソルバを紹介した。CAMGCG ソルバは P-CI スムーザと CA ランチョス法を用いて大域的縮約通信を回避するように設計されている。これに加えて、混合精度処理を最適化することにより、収束特性を劣化させることなく計算と通信の両方をさらに削減することに成功した。

CAMGCG ソルバの処理性能と収束特性を CA クリロフソルバと比較した。省通信ステップ数 $s > 10$ で動作する P-CBCG 法によって CA クリロフソルバのロバースト性が向上したが、14 億自由度の問題における P-CG ソルバと比べた性能向上は約 1.34 倍にとどまる。これは CA クリロフ部分空間法は元のクリロフ部分空間法と数学的に等価であり、収束特性の向上が期待できないためである。一方、CAMGCG ソルバは通信処理の削減と収束特性の向上を両立する。ここで、後者が前者に大きく影響する。今回の多相流体問題では二桁以上という劇的な収束特性の向上を示し、大規模問題になるほどより大きな性能向上が得られた。このような収束特性向上、混合精度処理、および、省通信設計の効果により、CAMGCG ソルバは 14 億自由度および 900 億自由度の問題において、P-CG 法に比べてそれぞれ約 3.7 倍および約 11.6 倍という大幅な性能向上を達成し、Oakforest-PACS 全系規模の 8,000KNL まで良好な強スケーリングを示した。

JUPITER コードにおいて、CAMGCG ソルバは問題サイズによらずほぼ同様の収束特性を示し、900 億自由度の大規模多相流体問題をリーズナブルな計算コストで処理することができた。この結果から、提案手法は将来のエクサスケール CFD シミュレーションにおいて有望な手法であることが示された。

謝辞

本研究は文部科学省（ポスト京重点課題6：革新的クリーンエネルギーの実用化）および最先端共同HPC基盤施設（JCAHPC）「大規模HPCチャレンジ」の支援を受けた。本研究の計算の一部はOakforest-PACS（JCAHPC）およびICEX（JAEA）において実施された。

参考文献

[1] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003.

[2] M. Hoemmen. *Communication-avoiding Krylov subspace methods*. PhD thesis, University of California, Berkeley, 2010.

[3] E. C. Carson. *Communication-Avoiding Krylov Subspace Methods in Theory and Practice*. PhD thesis, University of California, Berkeley, 2015.

[4] Reiji Suda, Li Cong, Daichi Watanabe, et al. Communication-avoiding CG method: New direction of krylov subspace methods towards exa-scale computing. *RIMS Kôkyûroku*, Vol. 1995, pp. 102–111, 2016.

[5] P. Ghysels, T. Ashby, K. Meerbergen, and W. Vanroose. Hiding global communication latency in the GMRES algorithm on massively parallel machines. *SIAM Journal on Scientific Computing*, Vol. 35, No. 1, pp. C48–C71, 2013.

[6] P. Ghysels and W. Vanroose. Hiding global synchronization latency in the preconditioned conjugate gradient algorithm. *Parallel Computing*, Vol. 40, No. 7, pp. 224–238, 2014.

[7] A. Mayumi, Y. Idomura, T.Ina, et al. Left-preconditioned communication-avoiding conjugate gradient methods for multiphase CFD simulations on the K computer. In *Proceedings of the 7th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, ScalA '16, pp. 17–24, Piscataway, NJ, USA, 2016. IEEE Press.

[8] Y. Idomura, T. Ina, A. Mayumi, et al. Application of a communication-avoiding generalized minimal residual method to a gyrokinetic five dimensional eulerian code on many core platforms. In *Proceedings of the 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, ScalA '17, pp. 7:1–7:8, New York, NY, USA, 2017. ACM.

[9] Y. Idomura, T .Ina, A. Mayumi, et al. Application of a preconditioned chebyshev basis communication-avoiding conjugate gradient method to a multiphase thermal-hydraulic CFD code. *Lecture Notes in Computer Science*, Vol. 10776, pp. 257–273, 2018.

[10] Kazuya Matsumoto, Yasuhiro Idomura, Takuya Ina, Akie Mayumi, and Susumu Yamada. Implementation and performance evaluation of a communication-avoiding gmres method for stencil-based code on gpu cluster. *The Journal of Supercomputing*, Vol. 75, No. 12, pp. 8115–8146, 2019.

[11] Y. Ali, N. Onodera, Y. Idomura, T. Ina, and T. Imamura. Gpu acceleration of communication avoiding chebyshev basis conjugate gradient solver for multiphase cfd simulations. In *2019 IEEE/ACM 10th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA)*, pp. 1–8, Nov 2019.

[12] Y. Idomura, T. Ina, Y. Ali, and T. Imamura. Acceleration of fusion plasma turbulence simulations using the mixed-precision communication-avoiding krylov method. *accepted for publication in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'20)*, 2020.

[13] Tsuyoshi Ichimura, Kohei Fujita, Pher Errol Balde Quinay, et al. Implicit nonlinear wave simulation with 1.08T DOF and 0.270T unstructured finite elements to enhance comprehensive earthquake simulation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, pp. 4:1–4:12, New York, NY, USA, 2015. ACM.

[14] Chao Yang, Wei Xue, Haohuan Fu, et al. 10M-core scalable fully-implicit solver for nonhydrostatic atmospheric dynamics. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '16, pp. 6:1–6:12, Piscataway, NJ, USA, 2016. IEEE Press.

[15] Y. Idomura, T. Ina, S. Yamashita, N. Onodera, S. Yamada, and T. Imamura. Communication avoiding multigrid preconditioned conjugate gradient method for extreme scale multiphase cfd simulations. In *2018 IEEE/ACM 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (scalA)*, pp. 17–24, 2018.

[16] Susumu Yamashita, Takuya Ina, Yasuhiro Idomura, and Hiroyuki Yoshida. A numerical simulation method for molten material behavior in nuclear reactors. *Nuclear Engineering and Design*, Vol. 322, No. Supplement C, pp. 301 – 312, 2017.

[17] Yosuke Kumagai, Akihiro Fujii, Teruo Tanaka, et al. Performance analysis of the chebyshev basis conjugate gradient method on the K computer. *Lecture Notes in Computer Science*, Vol. 9537, pp. 74–85, 2016.

[18] Martin H. Gutknecht and Stefan Röllin. The chebyshev iteration revisited. *Parallel Computing*, Vol. 28, No. 2, pp. 263 – 283, 2002.

[19] T. Shimokawabe, T. Aoki, C. Muroi, et al. An 80-fold speedup, 15.0 TFlops full GPU acceleration of non-hydrostatic weather model ASUCA production code. In *2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–11, Nov 2010.

メニーコア型スーパーコンピュータにおける
大規模電子動力学シミュレーションの実現

廣川　祐太、矢花　一浩、山田　篤史

野田　真史(*)、山田　俊介、朴　泰佑

筑波大学計算科学研究センター

(*)現在、株式会社アカデメイア

１．はじめに

筑波大学計算科学研究センターを主な実施機関として、我々は光科学シミュレータ SALMON (Scalable Ab-initio Light-Matter simulator for Optics and Nanoscience) の研究開発を行っている [1]。SALMON は古典的電磁気学 (Maxwell 方程式) と、第一原理電子状態計算 (Time-Dependent Kohn-Sham, TDKS 方程式) を組み合わせた異なるスケールの計算を同時に解く電子動力学シミュレーション (以下、マルチスケール計算) を実現する世界初のアプリケーションである。我々はこれまでに、マルチスケール計算について Oakforest-PACS 全系を用いた性能評価を実施し、良好なスケーリング性能を示した [2]。

マルチスケール計算は巨視的・微視的挙動を同時に解くことで従来法では表現不可能な電子と物質の相互作用をシミュレート可能となっただけでなく、HPC 分野においては、袖領域通信や大規模 IO 処理といった多くの実アプリケーションが大規模実行時の課題として挙げる諸問題を解決可能なスケーラブルなアプリケーションを実装できる。現在、我々は大規模 TDKS 方程式と Maxwell 方程式を組み合わせた電子動力学シミュレーション (以下、シングルスケール計算) の実現を目指している。シングルスケール計算では、先に掲げた諸問題の解決が極めて大きな課題となる。

我々は JCAHPC のご協力の下、シングルスケール計算を対象とした電子動力学シミュレーションの性能評価を大規模 HPC チャレンジにて遂行したが、Oakforest-PACS の 50%程度のノード規模になって初めて下記の複数の問題が発覚し、全系でのシミュレーションを達成することができなかった。

1. 初期波動関数生成時に NaN が発生
   - 初期値生成に使ったガウス関数と使用した擬似乱数の偏りによって、グラムシュミットによる直交化でゼロ除算が発生
2. メモリ消費量の課題
   - 全系計算であっても、MCDRAM (16 GiB) に問題なく収まると考えていたが、メモリ不足が多発した
3. 波動関数の IO 処理コストの爆発的増加
   - 計算対象となる波動関数配列が巨大化するにつれ、IO 処理時間が支配的となった

1 は当時の実装では数学的に問題があったこと、2 は問題規模に対し線形増加で確保されてしまう配列が複数あることが後日の調査で判明し、改修することができた。また、特に 3 はスーパー

コンピュータ「富岳」での実行を見据えた場合、数万ノード規模での IO 処理が必要となるため、大幅な高速化が必要であると考えた。

本稿では、シングルスケール計算におけるアプリケーション固有の課題である大規模 IO 処理の高速化とスーパーコンピュータ「富岳」のネットワークに起因する最適な MPI プロセスマッピングについて、それぞれの実装を紹介する。それらを踏まえ、我々がスーパーコンピュータ「富岳」の試行的利用（富岳共用前評価環境）にて実施した 27,648 ノードまでの弱スケーリング性能を紹介し、現在の課題を述べ結びとする。

２．光科学シミュレータ SALMON

SALMON（Scalable Ab-initio Light-Matter simulator for Optics and Nanoscience）は、光と物質の相互作用の第一原理計算を目的とした光科学シミュレータである。SALMON は、前進としてマルチスケール計算を実現する ARTED［3］と派生ソフトウェア GCEED［4］を統合し、異なる複数のスケールでのシミュレーションを 1 つのアプリケーションとして計算・実現可能である。2 つは TDKS 方程式の規模、それにともない並列分散方法が異なるが、計算内容はほぼ同一で、［2］で最適化したコードを活用できる。本稿で紹介する最適化も、すべて SALMON の最新版に取り込まれている[1]。

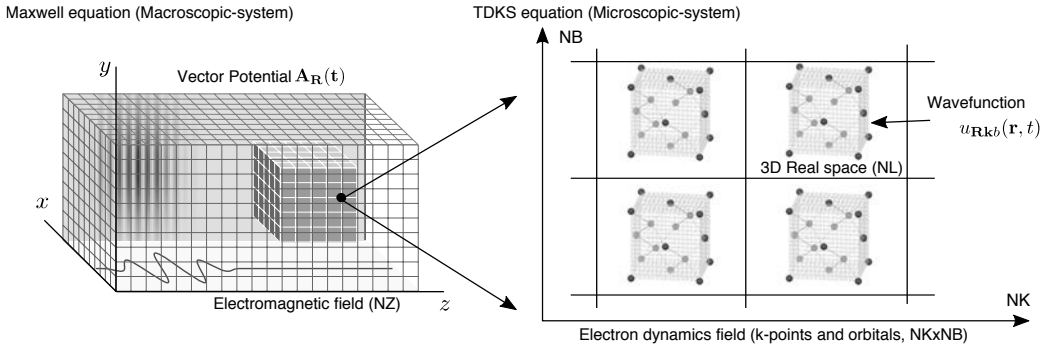


図 1: TDKS+Maxwell マルチスケールシミュレーションのイメージ図

3 次元 Maxwell 方程式で示された巨視的空間上に物質があり、超短波光を照射したときに物質（差分格子）内で発生する微視的現象を TDKS 方程式で解く。

電子動力学計算は、時刻ゼロの状態を決定するために電子の波動関数の基底状態を求める必要がある。これを基底状態計算と呼ぶ。同計算は、岩田らによる電子状態計算ソフトウェア RSDFT と同様のアルゴリズムを採用している［5］。同計算も第一原理計算に基づくため、計算コストが極めて高く、SALMON が対象とする問題規模では基底状態を求めるだけで数時間を要する場合がある。ただし、基底状態計算はグラムシュミットの正規直交化や部分対角化といった $O(N^3)$ の計算が支配的であるのに対し、電子動力学計算はステンシル計算や FFT などメモリ・ネットワークバンド幅に律速されるシミュレーションで、SALMON はどちらも高速に計算可能でなければならない。

[1] http://salmon-tddft.jp

第一原理計算は極めて多くの計算量が必要であることが知られており、マルチスケール計算も例に漏れず TDKS 方程式の求解時間が支配的となる。しかしながら、一般的な TDKS 方程式に対し実空間格子の規模は 16^3 といった一般的な CPU の L2 キャッシュメモリである 512 KiB 未満で、それぞれを独立に並列計算可能な波数空間（バンド計算、k 点計算）が大規模となるような、**MPI による分散並列化が容易に行える**シミュレーションが中心であった。加えて、Maxwell 方程式の差分計算（Finite-Difference Time-Domain method, FDTD 法）と各式の電流項にて結合することにより、FDTD 法の求解で必要とする格子点数の数だけ、TDKS 方程式を解く必要がある。このとき、1 つの TDKS 方程式の問題サイズは比較的少ないプロセッサ（16 台程度まで）で計算可能な規模にとどまり、TDKS 方程式は MPI のサブコミュニケータに閉じて計算が行われる。サブコミュニケータ**内**は TDKS 方程式の通信（波数空間を束ねるための Collective 通信）、サブコミュニケータ**間**は Maxwell 方程式（FDTD 法における格子点の袖交換）の通信に該当する。Maxwell 方程式の格子点数は TDKS 方程式の規模とのバランスから、Oakforest-PACS を用いても 32^3 程度と小さく、Maxwell 方程式の計算コストは無視できるレベルに小さい。

対してシングルスケール計算は Maxwell 方程式を TDKS 方程式と同じスケール、電子間相互作用を記述する。この場合の TDKS 方程式の実空間格子点は 256^3 程度と一般的な差分計算で採用される水準となるため、実空間の分散並列化が必要かつマルチスケール計算に比べて Maxwell 方程式の計算コストが高い。そこで、我々は TDKS 方程式の大規模化に伴う IO データ量の増加と、局所的通信の最適化を行う必要がある。

３．大規模電子動力学計算における大規模 IO 処理の効率化

SALMON では基底状態計算と電子動力学計算をそれぞれ別のバッチジョブとして実行可能なように、2 つの計算をバイナリデータファイルの受け渡しにより接続している。例えば気象・海洋学のアプリケーションでは netCDF や HDF5 が用いられるが、物性物理学では一般的なフォーマットが定義されておらず、また様々なシステムでの動作・利用を目的とする SALMON の性質上、依存するソフトウェアパッケージを削減するため、MPI-IO ネイティブ、および Fortran I/O ルーチン群で実装している。

バイナリデータファイルはほぼすべて電子の波動関数で、大規模実行においては**数 TB から数百 TB 程度のバイナリデータの IO 処理が必要**となる。加えて両計算はどちらも同水準に計算コストが高いため、システムエラーやソフトウェアのクラッシュなどの障害から復帰可能なようにチェックポイントの保存も必要である。以上の要求を満足するために、十分に高速な IO 処理を実現する必要がある。Oakforest-PACS 全系で計算可能な波動関数の規模を見積もると全体で約 4 TB で、同システムが提供するファイルシステムの理論データ転送速度（500 GB/s on Lustre, 1560 GB/s on BurstBuffer）では数秒で読み書き可能なサイズではあるが、MPI-IO を用いた全プロセスで 1 個のファイルとして読み書きする方式（以下、SSF: Single Shared File）ではこの性能を達成することは極めて困難である。

並列ファイルシステムの性能を最大限活用するためには各 MPI プロセスが独立に自身の担当領域を読み書きする方式（以下、FPP: File Per Process）が最適と考えられるが、SALMON では各計算において最適な性能を実現できる MPI プロセス割当ルールが異なる。したがって、計算ごとにプロセス割当を容易に変更できる柔軟性を持ち、かつ高速な IO 処理を提供可能なファイル構造が要求されている。

また、並列ファイルシステムの輻輳対策も必要となる。Lustre 上のファイルにアクセスする場合、実際にデータが保存されているオブジェクトストレージサーバ（OSS）のアドレスやデータサイズといったメタデータが保存されているメタデータサーバ（MDS）へアクセスし、メタデータを参照して対象ファイルが保存されている OSS へアクセス、IO 処理を行う。MDS および OSS は複数サーバによって構成されるが、対象ファイルのメタデータ保存先である MDS のアドレスは通常アクセス先のディレクトリによって決定される。つまり、複数のプロセスが同一ディレクトリ上の単一または複数ファイルへ同時にアクセスすると、特定の MDS へ問い合わせが集中し輻輳が発生、IO 性能の低下だけでなくファイルシステムの障害を引き起こしかねないため、ディレクトリを分散することで各 MDS の負荷を軽減しなければならない。FPP の場合、各 MPI プロセス専用のディレクトリを準備することで、全 MDS への負荷を均等にすることが可能であると考えられる。

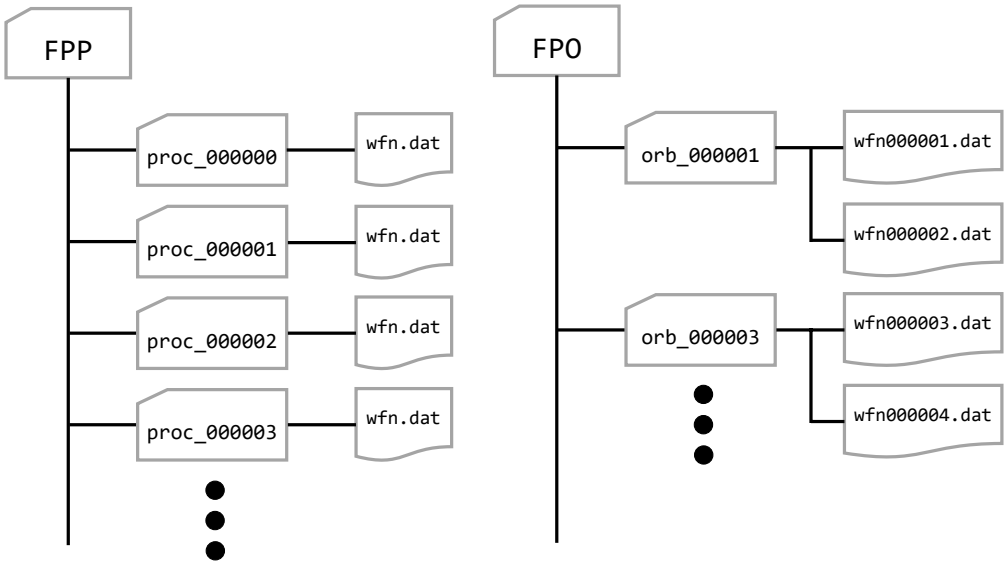


図 2: FPP および FPO の書き込みパターン

FPP（File Per Process）および FPO（File Per Orbital）の書き込みパターン、FPP では各プロセスが独立に専用のディレクトリにファイルの読み書きを行うのに対し、FPO は各軌道関数を計算するプロセス群が MPI-IO で読み書きを行う。その際のディレクトリ内の軌道関数ファイルの数は任意に決定可能。図の場合、2 を設定している。

アプリケーション特有の情報として、電子の波動関数として定義される配列は（X, Y, Z）の 3 次元実空間と、バンド理論によるバンド数および k 点数の 5 次元空間として定義される。このうち k 点は、非専門分野では各バンドのレプリカデータと解釈して良い。バンドそれぞれのデータについて、一般的に「軌道関数（Orbital Function)」と定義される。軌道関数単位でファイルを保存することで、MPI-IO 通信の局所化、ファイルシステムの輻輳軽減、バンド幅の有効活用を両立できるものと考える（以下、FPO: File Per Orbital)。軌道関数単位で取り回すことで、データの読み書きそのものもかなり単純化され、軌道関数配列の MPI_Datatype の定義、データタイプに基づいた Collective IO（MPI_File_read_all/MPI_File_write_all）通信を実行、

Collective IO を必要な軌道関数分だけ行うだけで良い。図 2 に FPP および FPO の模式図を示す。SALMON では FPP は障害復帰用のチェックポイント作成手段として実装し、通常利用時には SSF、大規模実行向けに FPO を実装している。FPO は、実空間の分割プロセス数によりアクセス負荷が変わることから、各ディレクトリ内に保存可能な軌道関数の数を任意に設定できる。ただし、我々の実装はディレクトリ生成に POSIX 定義の関数を利用しているため、POSIX を利用できない環境では動作しない。

この機能の実装により、富岳共用前評価環境上において、最高 12.3 TB の読み書き処理を 10 分以内に完了できることを確認した[2]。

４．Tofu-D ネットワークにおける MPI プロセス割当の最適化

Oakforest-PACS や筑波大学の Cygnus などは Intel Omni-Path や Mellanox InfiniBand を用いた Full-bisection Fat-tree ネットワークで構築されているため、ノードあたり 1 MPI プロセスを割り当てる場合、各 MPI プロセスが物理的な計算ノード群のどの位置にあり、通信相手となる計算ノードの距離を考慮することなく通信が可能であった。しかし富岳や FX1000 などの超大規模スーパーコンピュータでは Fat-tree の構築に必要なネットワークスイッチやケーブルの数といった実装コストの観点から、多次元メッシュトーラスネットワークが採用される。そのため、適切な MPI プロセスマッピングを考慮せずに実行すると、長い距離（ホップ数）となる通信が多発し、大幅な性能低下を招く可能性がある。富岳が採用する Tofu-D ネットワークは 6 次元メッシュトーラス形状で、一般ユーザからは 3 次元のノード集合として扱うことができる。

SALMON は、主に以下の 2 つの通信パターンに分けられる。

(1) 軌道関数**間**の通信: Collective (MPI_Bcast, MPI_Allreduce)

(2) 軌道関数**内**の通信: 袖領域交換、FFT 計算時の MPI_Alltoall

ここで重要なのは、基底状態計算では (1) が、実時間発展計算では (2) の通信が支配的で、他方の通信コストは negligible になる点である。SALMON は波動関数配列のすべての次元を MPI でプロセス分割できるため、支配的な通信を行うプロセス群が可能な限り連続した塊（以下、プロセスブロック）として計算ノードに割り当てられるようにすることで通信性能を最大化する必要がある。また富岳に搭載されている A64FX プロセッサは、12 コアを 1 NUMA 構成として 4 NUMA 構成を取り、1 NUMA ノードあたり 1 MPI プロセスでの実行が推奨されている。Tofu-D インターコネクトに比べて高速な NUMA ノード間通信を活用するためにも、プロセスの割当方法を工夫する必要がある。このとき k 点はレプリカのため、波動関数配列は 3 次元実空間 + 1 次元波数空間（バンド + k 点、電子軌道と呼ぶ）の 4 次元空間配列 (X, Y, Z, W) と解釈できる。対して、Tofu-D で構成されたネットワークは 3 次元 MPI プロセス空間として扱われるため、4 次元配列を 3 次元 MPI プロセス空間にマッピングする必要がある。(1) の場合は W 方向に対する通信が、(2) の場合は (X, Y, Z) 領域内で発生する通信コストが最小となるようにマッピングしなければならない。

---

[2] 我々が実験した際の富岳共用前評価環境において、並列ファイルシステムの整備状況に起因し、富岳が本来提供する計算ノードあたりの IO のバンド幅に比べ大幅に低い性能しか利用できなかったため、この結果でも良好と考えられる。

表 1: 各変数の説明

変数名	説明
$T_{x,y,z}$	Tofu-D ネットワーク上の各方向のノード数
P_{all}	全 MPI プロセス数
$P_{rx,ry,rz}$	実空間方向の MPI プロセス数
$P_{orbital}$	波数空間方向の MPI プロセス数
$P_{ox,oy,oz}$	$P_{orbital}$の因数
P_{node}	ノードあたりプロセス数（4 で固定）
$L_{x,y,z}$	実空間格子点数
N_{orb}	電子軌道数

ここで、(2) の場合のマッピング方法について考える。(2) の場合、(X, Y, Z) 領域を計算する 3 次元 MPI プロセスブロックを、合計で W 領域の MPI 並列数となるように 3 次元的に並べることにより、**擬似的な 4 次元 MPI プロセス空間を表現**し、(X, Y, Z) 領域内における通信コストを最小化する。以下、表 1 に示す各変数を用いて詳細を説明するが、各変数はそれぞれ以下の関係性を持つ。

$$(T_x \times P_{node}) \times T_y \times T_z = P_{all}$$
$$(T_x \times P_{node}) \div P_{rx} = P_{ox}$$
$$T_y \div P_{ry} = P_{oy}$$
$$T_z \div P_{rz} = P_{oz}$$
$$product(P_{ox}, P_{oy}, P_{oz}) = P_{orbital}$$
$$(T_x \times P_{node})\ mod\ P_{rx} = 0$$
$$T_y\ mod\ P_{ry} = 0$$
$$T_z\ mod\ P_{rz} = 0$$

また、一部の計算で FFTE[3]を利用し (X, Y, Z) 領域内で閉じた MPI 版 3 次元 FFT を利用するため、下記の条件も満たす必要がある。

$$L_x\ mod\ P_{ry} = 0$$
$$L_y\ mod\ P_{ry} = 0$$
$$L_y\ mod\ P_{rz} = 0$$
$$L_z\ mod\ P_{rz} = 0$$

まず、プロセス分割数を決定するユーザは、$P_{rx,ry,rz,orbital}$の 4 つを決定しなければならない。これにより必要な$T_{x,y,z}$および$P_{ox,oy,oz}$を上記式より決定できるが、$max(P_{ox,oy,oz})$が W 方向における最大の通信ホップ数となるため、これを最小に設定することが望ましい。ただしここで注意したいのは、導入されたシステムによって提供される 3 次元ノード形状の制約を受けるため、適切なパラメータの設定はシステムや問題規模に強く影響される。また$P_{rx} < P_{node}$の場合は、P_{ry}をノード内で連続に割り当てるといった設定も可能である。

[3] http://www.ffte.jp/

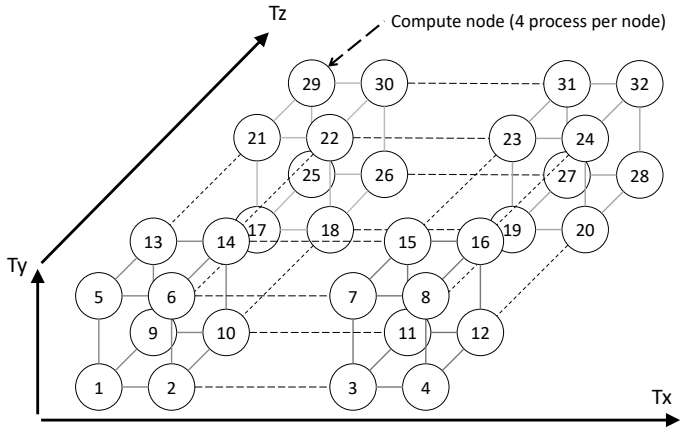


図 3: 3 次元ノード空間に対する 4 次元 MPI プロセスのマッピング

同図では合計 32 ノード 128 MPI プロセスにおけるマッピング例を示している。このとき、同色線で接続された 8 ノードが同じ（X, Y, Z）領域を計算する 3 次元ノードブロックで、同ブロックが 4 つ配置されている。すなわち、W 領域は 4 MPI プロセスで、(X, Y, Z) 領域は 32 MPI プロセスで並列化されている。

ここで、マッピングイメージを図 3 に示すが、このとき、以下通り各変数が設定されている。実際にどのように並列化されているかは図 3 の説明に示す通り。

$$(T_x, T_y, T_z) = (4,2,4)$$
$$(P_{rx}, P_{ry}, P_{rz}) = (8,2,2)$$
$$(P_{ox}, P_{oy}, P_{oz}) = (2,1,2)$$
$$P_{orbital} = 4$$

以上の通り 4 次元 MPI プロセス空間を擬似的に作成することで、通信コストを最小化することが可能となる。今回、本節の冒頭で説明した（2）におけるマッピング方法を紹介したが、(1) の場合は条件式のP_{rx}と$P_{orbital}$を入れ替えた式に変形することで、最適な MPI プロセスマッピングを実現できる。

５．スーパーコンピュータ「富岳」での性能評価と現在の課題

我々の現在のシミュレーション目標は、電子動力学計算の時間ステップあたりの実計算時間を 1 秒以内（1 second/iteration）で実行することである。この目標値は、電子動力学計算で一般的に必要な時間ステップから、1 実験あたり 24 時間程度でシミュレーションを完了するために必要な計算速度として算出している。本節では弱スケーリング性能を紹介し、現在の課題について述べる。

表 2: 弱スケーリングにおける問題設定

# of node	$T_{x,y,z}$	$P_{rx,ry,rz,orbital}$	$L_{x,y,z}, N_{orb}$
432	(3, 6, 24)	(2, 2, 12, 36)	(108, 96, 480, 5328)
1,728	(6, 12, 24)	(2, 4, 12, 72)	(108, 192, 480, 10656)
6,912	(24, 12, 24)	(4, 4, 12, 144)	(216, 192, 480, 21312)
27,648	(48, 12, 48)	(4, 8, 12, 288)	(216, 384, 480, 42624)

今回、我々が対象とした問題はスーパーセル計算という複数のシングルセル（レプリカ）を結合した物理シミュレーションとなっている。同計算は、対象となる波動関数の実空間を 2 倍にしたとき、同時に電子軌道も 2 倍にする必要があり、Weak scaling では 4 倍の計算リソースでスケールしなければ正当な評価ができない。表 2 に、27,648 ノードまでの弱スケーリングにおける問題サイズ、3 次元ノード形状およびプロセス分割ルールを示す。Z 方向から長短パルスが入力されるため、Z 次元の格子点数は問題の性質上固定される。

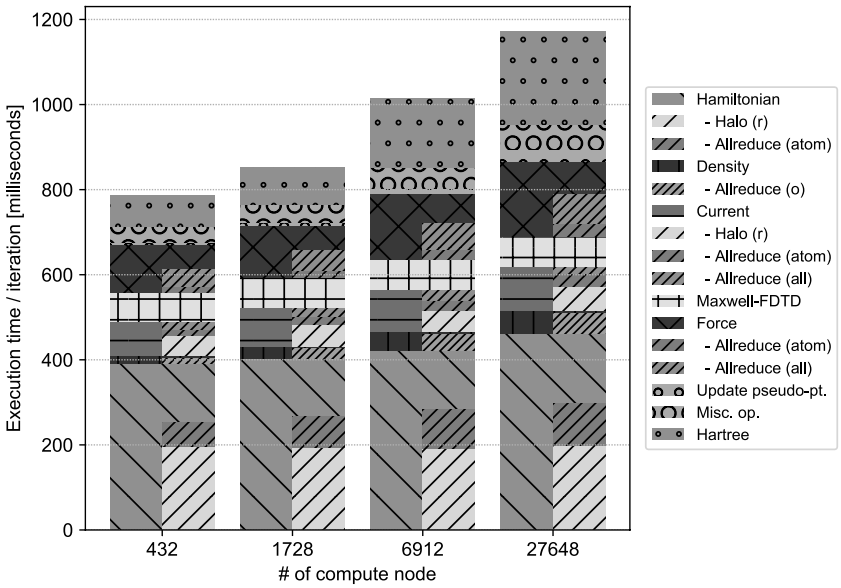


図 4: 弱スケーリング性能評価結果

縦軸は時間ステップあたりの計算時間［ミリ秒］で、横軸は計算ノード数である。図中の幅が半分になっている棒グラフは、各実行時間の中での通信種別および通信時間を示している。すなわち、通常幅の棒グラフから通信時間を差し引くと正味の計算時間と解釈できる。

図 4 に富岳共用前評価環境における弱スケーリングの結果を示す。この中で最下段グラフは支配的な計算である波動関数のハミルトニアンの実行時間で、通信時間が半分以上を占めているが MPI_Allreduce のみ増加が見られる。これは 3 次元空間上に存在する全原子に対する通信だが、原子数は弱スケーリング時に各ステップで 2 倍に増加し、通信時間も増加する傾向にある。対してハミルトニアンの実行時間の半分を占める袖領域交換は、ほぼ完全な状態でスケールしており、本稿で紹介した MPI プロセスマッピング手法が効果的に動作したことを示している。

ここで注目したいのは最上段グラフで、これは Hartree potential を 3 次元実空間から得た電子密度から計算する工程である。Hartree potential は陰的に解く必要があるが、SALMON では 3 次元実空間内での局所的 3 次元 FFT で解を求めている。FFT は MPI_Alltoall が必要な計算パターンのため、富岳のようなネットワークバンド幅が相対的に低いシステムでは性能を出しにくい欠点がある。ただし、SALMON における FFT 計算は、表 2 を参照すると$product(P_{rx,ry,rz})/P_{node}$が FFT の計算で使われるノード数となり、27,648 ノード構成であっても 96 ノードといった小規模

ノード群で閉じた並列計算である。もしこの性能劣化特性がキープされるとするならば、全系計算を行った場合、432 ノード実行に対して約 55%の実行効率、約 1.5 seconds/iteration となり当初の目標値を達成できない。したがって、FFT の代わりに前処理付き CG 法で解くなど、別の手法が必要と考えられる。

６．おわりに

本稿では、我々が開発する SALMON を用いた大規模電子動力学シミュレーションの実現のために、IO 処理の高速化と安定化、メッシュネットワークに対する多次元 MPI プロセスマッピング手法について紹介し、スーパーコンピュータ「富岳」の 1/6 に相当する 27,648 ノードまでの計算性能を紹介した。

計算性能については、Oakforest-PACS（Intel Xeon Phi）プロセッサにおける知見、最適化の成果を十分に活用でき、また Oakforest-PACS 全系を用いた実験により発覚した諸問題の解決、富岳共用前評価環境にて IO 性能と MPI プロセスマッピングを最適化することにより、富岳全系を用いた大規模電子動力学シミュレーションの実現に大幅に近づけたと考えられる。

今後の課題は、大規模化につれ局所的な FFT の MPI_Alltoall 通信が全体性能を押し下げているため FFT フリー計算の実現が挙げられる。そして、富岳の供用開始後にはその計算リソースを十分に活かし光科学の発展に寄与できるものと期待される。

謝　辞

本研究の内容の一部は、JCAHPC の運用する Oakforest-PACS を用いた大規模 HPC チャレンジの結果から得られた。同チャレンジがなければ富岳共用前評価環境における成果を得ることは困難であり、今回の機会をくださった JCAHPC に感謝する。

免　責　事　項

本研究の富岳共用前評価環境の利用における性能評価は、2020 年 4 月末までの利用結果であり、スーパーコンピュータ『富岳』の共用開始時の性能・電力等の結果を保証するものではない。

参　考　文　献

[1] M. Noda, *et.al*: SALMON: Scalable Ab-initio Light-Matter simulator for Optics and Nanoscience, Computer Physics Communications. Volume 235, 356 (2019). https://doi.org/10.1016/j.cpc.2018.09.018

[2] Y. Hirokawa, *et.al*: Performance Optimization and Evaluation of Scalable Optoelectronics Application on Large Scale KNL Cluster, Proceedings of ISC2018 (2018). https://doi.org/10.1007/978-3-319-92040-5_11

[3] S.A. Sato and K. Yabana: Maxwell + TDDFT multiscale simulation for laser-matter interactions, J. Adv. Simulat. Sci. Eng., Vol. 1, No. 1, pp. 98-110 (2014). https://doi.org/10.15748/jasse.1.98

[4] M. Noda, *et.al*: Massively-Parallel Electron Dynamics Calculations in Real-time and Real-Space: Toward Applications to Nanostructures of more than Ten-Nanometers in

Size, Journal of Computational Physics, Vol. 265, No. 14, pp. 145-155 (2014). https://doi.org/10.1016/j.jcp.2014.02.006

[5] Y. Hasegawa, *et.al*: Firstprinciples Calculations of Electron States of a Silicon Nanowire with 100,000 Atoms on the K Computer, Proceedings of SC11 (2011). https://doi.org/10.1145/2063384.2063386

学際大規模情報基盤共同利用・共同研究拠点

公募型共同研究　2020 年度採択課題

飯 野 孝 浩

東京大学情報基盤センター

１．今年度の課題応募・採択状況について

「学際大規模情報基盤共同利用・共同研究拠点」は，8 大学（北海道大学，東北大学，東京大学，東京工業大学，名古屋大学，京都大学，大阪大学，九州大学）の計算機関連共同利用施設を構成拠点とし，当センターを中核拠点とする「ネットワーク型」共同利用・共同研究拠点として文部科学大臣の認定を受け，2010 年 4 月から本格的に活動を開始している。

2020 年度の一般・国際研究課題には 65 課題の応募があり，2020 年 2 月に行われた課題審査委員会による厳正な審査により，うち 53 課題が採択された。なお，萌芽課題については 8 月現在で 43 件が採択されている。採択された一般・国際研究課題のうち，東京大学の資源を用いる課題の代表者・所属・課題名について表 1 に示す。 なお，各課題の詳細については JHPCN ウェブサイト[1]を参照されたい。

表 1：2020 年度 JHPCN 採択課題（一般・国際研究課題，東大資源利用分のみ）

代表者名	代表者所属	課題名
三浦英昭	核融合科学研究所	電磁流体力学乱流の高精度・高並列 LES シミュレーションコード開発研究
柏原賢二	東京大学大学院総合文化研究科	大規模並列計算による格子の最短ベクトル探索の効率化に関する研究
片桐孝洋	名古屋大学情報基盤センター	Developing Accuracy Assured High Performance Numerical Libraries for Eigenproblems
天本義史	九州大学先導物質化学研究所	データサイエンスに基づく高分子材料の構造物性相関
齊木吉隆	一橋大学大学院経営管理研究科	機械学習に基づくマクロ経済変動の数理モデリング
村田忠彦	関西大学総合情報学部	社会の分析とシミュレーションのための合成人口データ提供システム
横田理央	東京工業大学学術国際情報センター	Hierarchical low-rank approximation methods on distributed memory and GPUs
田仲正弘	国立研究開発法人情報通信研究機構ユニバーサルコミュニケーション研究所データ駆動知能システム研究センター	超巨大ニューラルネットワークのための分散深層学習フレームワークの開発とスケーラビリティの評価

[1] https://jhpcn-kyoten.itc.u-tokyo.ac.jp/ja/

安藤亮輔	東京大学理学系研究科地球惑星科学専攻	時空間領域境界積分方程式法の高速解法の開発と巨大地震シミュレーションへの応用
中島研吾	東京大学情報基盤センター	High resolution simulation of cardiac electrophysiology on realistic whole-heart geometries
中島研吾	東京大学情報基盤センター	高性能・変動精度・高信頼性数値解析手法とその応用
村田健史	国立研究開発法人情報通信研究機構ソーシャルイノベーションユニット総合テストベッド研究開発推進センター	HPC と高速通信技術の融合による大規模データの拠点間転送技術開発と実データを用いたシステム実証試験
藤井昭宏	工学院大学情報学部	Innovative Multigrid Methods II
佐藤一誠	東京大学情報理工学系研究科	Deep Learning を用いた医用画像診断支援に関する研究
下川辺隆史	東京大学情報基盤センター	Development of Fast Surrogate for Approximating Large-scale 3D Blood Flow Simulation
深谷猛	北海道大学情報基盤センター	エクサスケール時代の数値計算手法に対する性能予測技術
大島聡史	名古屋大学情報基盤センター	分散機械学習技術を用いた大規模医用画像処理の実現に向けた研究
長﨑正朗	京都大学学際融合教育研究推進センタースーパーグローバルコース医学生命系ユニット	ハイブリッドクラウド構築とゲノム情報解析の効率的な運用に関した研究
小野寺直幸	国立研究開発法人日本原子力研究開発機構システム計算科学センター	Scalable Multigrid Poisson solver for AMR-based CFD applications in Nuclear Engineering
安藤嘉倫	名古屋大学工学研究科附属計算科学連携教育研究センター	階層的に並列化された分子動力学計算ソフトウェアの最適な実行条件探索方法の確立
三好建正	理化学研究所計算科学研究センター	ゲリラ豪雨予測のリアルタイム実証実験
高木洋平	横浜国立大学大学院工学研究院	機械学習を用いた風環境予測精度の向上と防災技術への応用
地道正行	関西学院大学	財務ビッグデータの可視化と統計モデリング

学際大規模情報基盤共同利用・共同研究拠点
第 12 回シンポジウム開催報告

飯 野 孝 浩

東京大学情報基盤センター

１．ねらいと当日の運用

2020 年 7 月 9 日，第 12 回となる 2020 年度 JHPCN 拠点シンポジウムがオンラインにて開催された。事前に行われた課題審査委員会での議論により，今年度は規模を縮小し， 1）オーラル発表は数を減らして実施， 2）ポスター発表は，従来はオーラル発表であった課題と従来のポスター課題に拡張して実施，という形式で行われた。今年度からの新たなポスター発表の枠として，来年度に各拠点から提供される資源や取り組みについての発表，採択課題以外からの一般発表も公募された。さらにオーラル発表として，来年度以降の稼働が予定されているデータプラットフォーム(mdx)構想の発表がされた。それぞれの発表枠での発表数は以下の通りである。

・2019 年度実施課題の成果報告：オーラル 15 件，ポスター43 件
・2020 年度採択課題の内容紹介：ポスター52 件
・2019 年度年度実施萌芽課題の成果報告：ポスター3 件
・2020 年度採択萌芽課題の内容紹介：ポスター9 件
・2021 年度提供予定資源の紹介：8 センター
・その他：オーラル 1 件，ポスター2 件

オーラルセッションは Zoom を用いた同期型の実施とし，課題審査委員会が選定した 15 件について，午前と午後それぞれのセッションに 7 件，8 件と割り振った。その後にポスターセッションコアタイムを 1 時間 50 分設定するというスケジュールであった。

事前に強く懸念されたことは，ポスターセッションにおいてどのように議論の場を設定するかということであった。ポスターセッションの持つ，ポスターをザッピングしながら会場を歩き，参加者間でフランクなコミュニケーションを行うという形式をオンラインでいかに実現するかは難しい課題であり，知見が十分にあるとは言い難い。凝ったシステムを導入したものの不具合が続発してしまった学会，ポスターセッションの目的を果たせないとしてポスターセッション自体を中止してしまう学会もあり，事務局を悩ませることになった。

紆余曲折の末，広く用いられている業務用 SNS である Slack を用い，非同期・同期それぞれで質疑応答を行う，という仕様とした。課題ごとに 1 つのスレッドを作り，ポスターの PDF や関連動画ファイル等をアップロード[1]し，質疑応答は当該スレッド内で行うという形式にした。Slack へのログインは前日までにほとんどの方が済ませてくれていたが，質問を該当するスレッドにて

[1] Slack へのファイルのアップロードや更新，発表者名・課題名の更新を人海戦術の手作業でこなしてくださった，研究支援チームの杉田さん，伊藤さんのご尽力のおかげで，本形式での実施が可能になった。当日もタイムキーピングなどの慣れない業務をこなしてくださったお二人なしに，今回のシンポジウムは実施不可能であった。改めて御礼を申し上げる。

行うという方式は一部で徹底できなかった。

オーラルセッションは Zoom のウェビナーモードで実施したため，運営・座長に大きな権限がある形式であった。質疑応答はすべて Slack の該当スレッドに事前に寄せてもらい，座長が選定の上で読み上げるという形式にした。Zoom にも発表意思を示す機能はあるものの，オーラルセッションのうちに Slack 上への質問投稿に慣れてもらいたいという意図であった。

シンポジウム参加登録者は 247 名であり，オーラルセッション中に観覧していた人数は定常的に 150 名ほどであった。登録者のほぼ全員が Slack にアカウントを作成していた。

２．アンケート抜粋

参加登録者を対象としてアンケート調査を実施したので，その結果の一部をここに紹介する。結果全体の紹介は今後のセンター広報誌 Digital Life（現在リニューアル中）を参照されたい。回答者数は 59 であった。

口頭発表の件数を大幅に絞った件については，昨年度までのように全課題にすべきという回答は 13.6%にすぎなかった。また，会期を 2 日から 1 日に短縮したことについても，否定的な意見は 32.2%と少数派であった。また，パラレルセッションの実施についても否定的な意見は 33.9%に過ぎなかった。

冒頭で触れたように，採択課題数の増大は，シングルトラックで全課題代表者に口頭発表をしてもらうという形式を不可能にしつつある。たとえ来年度以降に疫学的状況が劇的に改善していたとしても，シンポジウムの新たな形式を再び模索すべきときに来ていると言えよう。

科学技術計算Ｉ／計算科学アライアンス特別講義Ｉ／スレッド並列コンピューティング　「科学技術計算のためのマルチコアプログラミング入門」（オンライン）

中 島 研 吾

東京大学情報基盤センター

本稿では，2020 年度 S1・S2 学期に実施した，科学技術計算 I（大学院情報理工学系研究科数理情報学専攻）／計算科学アライアンス特別講義 I（同 コンピュータ科学専攻）／スレッド並列コンピューティング（大学院工学系研究科電気系工学専攻）「科学技術計算のためのマルチコアプログラミング入門」[1]について紹介する。

近年マイクロプロセッサのマルチコア化が進み，様々なプログラミングモデルが提案されている。中でも OpenMP は指示行（ディレクティヴ）を挿入するだけで手軽に「並列化」ができるため，広く使用されており，様々な解説書も出版されている。メモリへの書き込みと参照が同時に起こるような「データ依存性（data dependency）」が生じる場合に並列化を実施するには，適切なデータの並べ替えを施す必要があるが，このような対策は OpenMP 向けの解説書でも詳しく取り上げられることは余り無い。本講義では，「有限体積法から導かれる疎行列を対象とした ICCG 法」を題材として，科学技術計算のためのマルチコアプログラミングにおいて重要なデータ配置，reordering などのアルゴリズムについての講義，スパコン（大規模超並列スーパーコンピュータシステム（Oakbridge-CX，OBCX）[2]）を使用した実習を実施した。

講義内容の詳細については，ウェブページから資料をダウンロードできるのでそちらを参照いただきたい。本講義では，受講者の多様なバックグラウンドを考慮して，ほぼ全講義内容について Fortran，C 両方による教材を準備している。

さて，今年は「新型コロナウイルス感染症」のため，全ての講義を Zoom によるオンラインで実施した。個人的には，既に 3 月初め頃からオンライン講義に向けた準備は始めていたが，もともと，Microsoft PowerPoint を使って講義を実施し，Web から資料を公開していたので，それほど特別な準備をする必要はなかった。従来は，情報基盤センターの演習室で教育用計算機システム（Educational Campus-wide Computing System, ECCS）[3]の端末を使用して講義・演習を実施していたが，オンライン講義では各自の PC を使用するため，Windows，macOS，Unix/Linux に対応した必要ソフトウェア類のインストールのためのマニュアル[4]は今回のために特別に準備した。また，ハンズオン演習で最も障壁となるのがスパコンへの SSH ログインである。普段は問題がある人は一人ずつ見回って相談しながら解決するのだが，今回は可能な限り詳細な資料[5]を用意することを心がけた（図 1）。**オンライン講義を体験した人の多くが感じていることであろうが，オンラインでは対面講義と同じことは決してできない。一定の制約の下でオンラインの特性を生かした講義・演習を心がける必要がある。対面であれば，学生の反応を見なが**

[1] http://nkl.cc.u-tokyo.ac.jp/20s/
[2] https://www.cc.u-tokyo.ac.jp/en/supercomputer/obcx/system.php
[3] https://www.ecc.u-tokyo.ac.jp/
[4] http://nkl.cc.u-tokyo.ac.jp/20s/OnlineClass.pdf
[5] http://nkl.cc.u-tokyo.ac.jp/20s/OBCXlogin.pdf

ら，理解が不足している項目に関しては，より詳しく説明するということになるのだが，オンラインではそれが難しいため，とにかく可能な限り詳細な資料を用意するしかない。

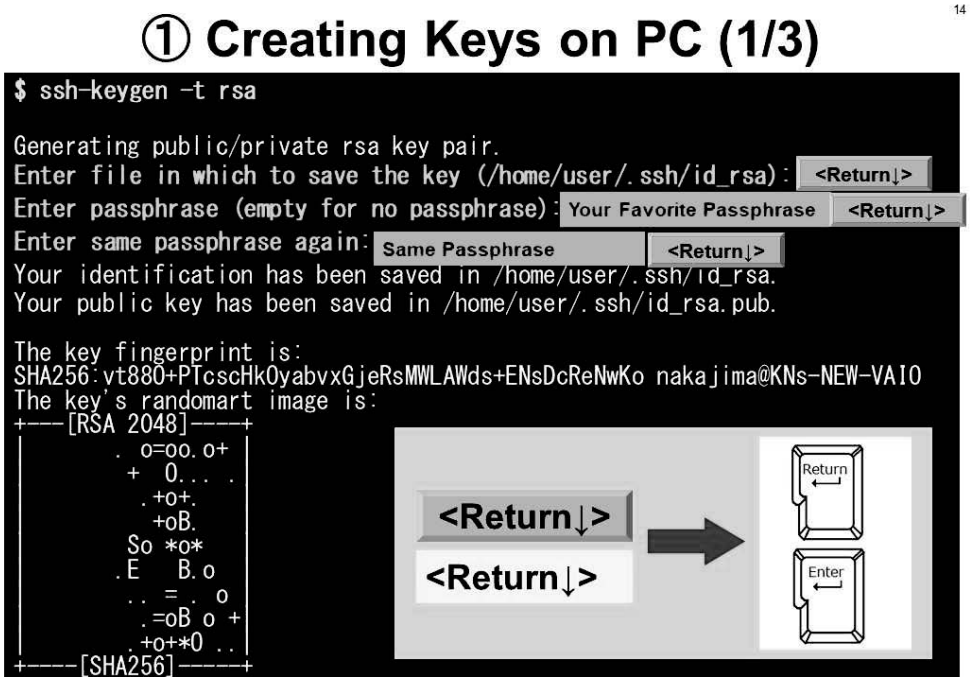


図 1　PC 上での SSH 鍵生成の手順説明

Zoom のための URL，スパコンの利用者 ID，初期パスワード等の受講者への連絡については UTAS（UTokyo Academic Affairs System），ITC-LMS（Learning Management System）等の学務システムを駆使して比較的安全と思われる方法を選択した。実はこの部分にはそれなりの時間を費やしたのであるが，諸般の理由により，ここでは詳しくは書かない。興味がある人は個別に連絡してほしい。

本講義は 4 月 8 日に開講したが，情報理工学系研究科から「学生の通信環境が整っていない場合があるので，最初の 2 回程度は準備にあて，4 月 20 日以降本格的な講義を開始する」という通達があったので，4 月 8 日と 15 日は同じ講義を実施した（図 2）。

登録者は 54 名（科学技術計算 I ：14 名，計算科学アライアンス特別講義 I ：11 名，スレッド並列コンピューティング：29 名）であった。例年は登録者 35 名程度，出席者は 20 名を切る程度なのであるが，今年はオンラインになったということもあり，08:30 開始の 1 限にもかかわらず，7 月の最後の講義まで常時 35～40 名程度の出席者があった。毎回の講義は録画して，クラウド上の動画のありかを ITC-LMS 経由で受講者に連絡していた。どうやらこれを使って復習している学生も少なからず居るらしく，連絡を忘れると催促が来たりした。**このようにビデオ動画等を使ってオンデマンドで受講できるというのはオンライン講義の一つの利点と言えるだろう，ということもオンライン講義をやってみて得られた知見である。国際会議でもオンデマンド（発表聴講）とリアルタイム（質疑・議論）とを組み合わせる方式は，定着しつつある。更に Slack のようなツールを組み合わせることによって，実は従来よりも効果的なコミュニケーションを効率良く実現できる可能性もある，ということを感じている人も多いであろう。**

一方で，各講義で Slack を立ち上げるため，学生にも「Slack 疲れ」というような状況に陥っている，という話も聞いたので，今学期は自分の講義用の Slack のワークスペースは作ったが結局使わなかった。来学期以降は利用を検討してみたい。

通常も同じだが，15 分に一回くらい小休止して Zoom のチャット機能による質問を受け付けることにしたところ，普段よりも活発に質問が出ていたようである。

何よりも大変だったのは学生諸君だったと思うが，各自色々と工夫しながらこの状況に適応していこうという姿勢が伝わってきた。また，6 月 24 日は，学期前は海外出張で休講の予定だったが，この時間は演習として，講義は実施せず 105 分全てを復習と質疑に当てた（これと言った質問はなかったが）。

2018・2019 年度に引き続きやや難しいプログラミング（sequential reordering の実装と評価）をレポート課題としたが，単位を取得したのは 13 名であった。出席者数は減少しないものの，実際にどれくらい理解できているのか，やや不安な面もあったが，例年よりもレポート提出者，単位取得者は若干増加した（単位取得者数は 7 名（2018 年），10 名（2019 年））。理解度をモニターするために，小テスト的な課題を課することも考えたが，オンラインになって課題が増えて学生の負担になっている，というようなことも聞いていたので実施はしなかった。

2013 年度以降，資料は英語版のみ用意していたが，講義そのものは日本語で実施していた。2017 年度から英語で実施することとしたため，留学生の受講は増加しており，2020 年度は登録者の半数をやや上回っている。

以上，ほとんどがオンライン講義に関する感想のようになってしまった。A1・A2 学期も基本的にはオンライン講義継続となった。様々な模索も継続すると考えられるが，より受講者の負担を減らすような方法が何よりも重要である。

Date	ID	Title
Apr-08 (W)	CS-01a	Introduction-a
Apr-15 (W)	CS-01b	Introduction-b (Introduction-a and –b are same)
Apr-22 (W)	CS-02	FVM (1/3)
Apr-29 (W)	(no class)	(National Holiday)
May-06 (W)	(no class)	(National Holiday)
May-13 (W)	CS-03	FVM (2/3)
May-20 (W)	CS-04	FVM (3/3) , OpenMP (1/3)
May-27 (W)	CS-05	OpenMP (2/3), Login to OBCX
Jun-03 (W)	CS-06	OpenMP (3/3)
Jun-10 (W)	CS-07	Reordering (1/2)
Jun-17 (W)	CS-08	Reordering (2/2)
Jun-24 (W)	-	Exercise, Q/A (Optional)
Jul-01 (W)	CS-09	Tuning
Jul-08 (W)	CS-10	Parallel Code by OpenMP (1/2)
Jul-15 (W)	CS-11	Parallel Code by OpenMP (2/2)
Jul-22 (W)	CS-12	Advanced Topics, Q/A

図 2　講義スケジュール：学期前は 6 月 24 日は海外出張で休講の予定だったが，105 分まるごと質疑にあてることとした

第135回お試しアカウント付き並列プログラミング講習会「Oakforest-PACS実践」実施報告

塙　敏博
東京大学情報基盤センター

2020年6月17日（水）の午後、第135回お試しアカウント付き並列プログラミング講習会「Oakforest-PACS実践」が開催されました。例年は東京大学情報基盤センターにおいて開催されている本講習会ですが、今回は新型コロナウイルス感染症対策のためにZoomを用いたオンライン講習会として実施されました。

本講習会は、東京大学内および学外における当センターのスーパーコンピュータの利用を考えているユーザに加え、社会貢献の一環として、高性能計算や並列処理の技術習得を目的にした企業に所属する研究者、技術者の方が参加可能になっております。

受講者は、学部学生：2名、大学院学生(修士)：7名、大学院学生(博士)：2名、教授：1名、准教授：1名、講師：2名、企業の方：1名、計16名の方にご参加いただきました。

1ヶ月有効となるお試しアカウントが与えられ、Oakforest-PACSスーパーコンピュータシステムの利用方法、OpenMPおよびMPI(Message Passing Interface)を用いたプログラミングに関する実行方法についての演習が、終日の日程で行われました。

当日のプログラムを、以下に載せます。

- 6月17日（水）

  13：00 － 14：15　Oakforest-PACSシステム紹介、KNL概要

  14：30 － 16：15　KNLにおけるOpenMP最適化、Oakforest-PACSでのMPI並列化(講義＋演習)

  16：30 － 17：45　OpenMP+MPIハイブリッド並列化と性能分析（講義+演習）

14名の参加者について、講習会に関するアンケートをご提出いただきました。主要な項目の集計結果を以下に示します。

プログラミング経験については、5年未満が半数、20年を超える型も3名いらっしゃいました。並列プログラミングについては、知識を前提にしていましたが、経験なしの方が4名いらっしゃいました。使用しているプログラミング言語については、FortranとPythonが同数、CとC++（複数回答可）となり、Pythonのユーザが増えてきています。

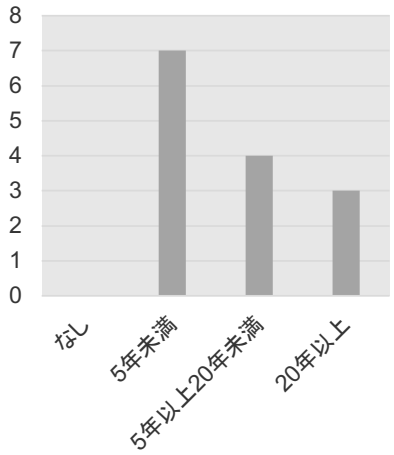


図 1 プログラミング経験

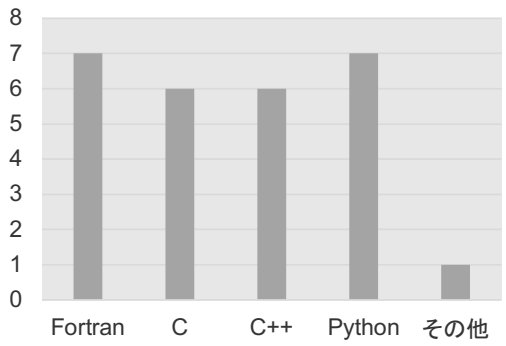


図 2 普段使用するプログラミング言語

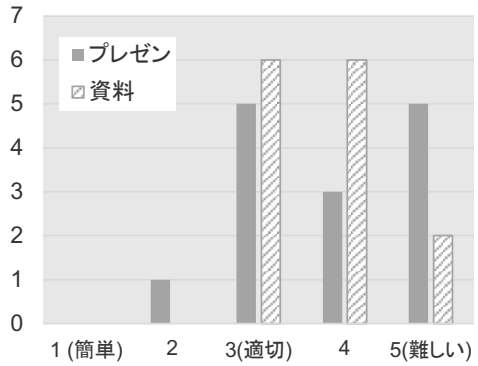


図 3 講習会の内容

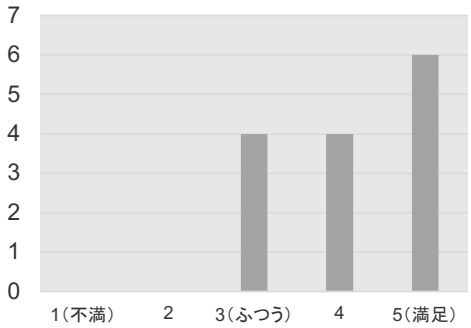


図 4 講習会参加の満足度

講習会の満足度の平均値は 4.1 であり、図 4 に示すようにおおむね満足度が高かったようです。ただ、図 3 に示すように、内容が難しいと感じた方もいらっしゃったようです。

今回は Zoom を用いての完全オンライン開催であったので、オンライン開催に関する回答をいただきました。オンライン開催で良かったことについての主な回答は

- 講師の画面を見ながら演習などを行うことができ，やりやすかった.
- 現地に行かなくても、講習が受けられたこと。気軽に申し込めた。
- 大きなモニターを繋げて参加でき、必要な情報がすべて手元で見れる。
- 移動にかかる費用や時間が不要なこと
- いつも利用している自分の環境からスパコンにログインしながら実習を受けられたことがよかった。
- サンプルプログラムの実行が出来たこと。移動時間が節約出来たこと。

- 関西に住んでいるので，自宅で受講することができたことが良かったです.
- 参加に関わる移動時間がなくなり、非常に合理的な開催方法だと思います。

一方悪かったことについては、

- 全員の進捗状況が把握しにくいこと。通信環境に依存してしまうこと。
- 講師側でミュートになっていて、音声が途切れたところがあった。

との意見をいただきました。遠隔地からの参加、使いやすい環境で講義を受けられることで、概ね好評だったようです。

また、以下のような感想をいただきました。

- この度は、オンラインで講習会を開いて下さりありがとうございました。本日学んだことを、今後の研究に活かしていきます。
- 最後の Scorep と Scalasca を用いた可視化の部分が良くわからなかった。
- 引き続きオンラインでも参加できる枠があるとよいと感じました。
- HT, SIMD, NUMA などの知識を前提に講義が進んでいたため、講義後半の様々な数値設定パラメタが、何を設定しようとした数値なのか、わかりにくかった。OpenMP+MPI では設定が複雑なことがわかった。
- 具体的な科学問題のサンプルでの並列効率化が見えるとありがたいです。
- 将来的には、参加者が録画データを見直せるようにしていただければ幸いです。

同様の講習会があれば、「また受けたい」という回答が 8 名、「どちらともいえない」が 6 名で、感想からもその他の講習会にも期待されていることが伺えます。

「KNL 実践」講習会、「OFP 実践」講習会合わせて今回で 8 回目になりました。参加者からは非常にためになったとの声も多く聞かれており、Oakforest-PACS 運用から 3 年半が経過していますが、富岳が運用を始めるまでの間、国内最大規模の計算資源である Oakforest-PACS の重要性が増しています。

特に今回は、オンライン開催への移行に伴って遠隔地からの参加が容易になったこともあり、今までは参加できなかった方々に対しても参加が可能になったということもありそうです。また、録画データの公開によって、復習に役立てたり、自習に使っていただく例もあるかもしれません。

しばらくは新型コロナウイルス感染症対策でオンラインのみの開催が続きます。オンライン講習会にはオンサイト講習会にない利点があることも分かってきたので、今後オンサイト開催が可能になってもオンラインを考慮しながら内容を検討していく予定です。

以上

原　稿　募　集

本誌では利用者の皆様からの原稿を募集しています。以下の執筆要項に基づいて投稿してください。

執　筆　要　項

1　内容は，本センターのスーパーコンピューターシステムの利用者にとって有意義な情報の提供となる原稿とします。

2　掲載可否については当編集委員会で決定させていただきます。

3　掲載可とした投稿原稿に対して，加除訂正を行うことがあります。

4　原稿枚数には特に制限はありませんが，シリーズに分割することもあります。

5　プログラムの実例が大量になる場合(概ね１頁を超える)は，本文には一部のみを記述し，投稿者の Web ページ等に全体を掲載し，その URL を引用するようにしてください。

6　原稿は横書きにしてください。

7　原稿は，印刷出来上がり寸法がＢ５判で文字の大きさ９ポイントを標準とし，印字部分は必ず左端を 2.5cm 以上空けて，縦 21cm，横 14cm になるようにしてください。Ａ４サイズの場合はＢ５サイズに縮小した場合に上記のサイズになるようにしてください。

　併せて，PDF 形式(フォント埋め込み)の完全原稿を電子メールにて uketsuke@cc.u-tokyo.ac.jp まで提出願います。

8　投稿原稿は返却しません。

9　採用された原稿は，本センターの Web ページ上でも掲載させていただきます。
希望がある場合は，１タイトルにつき 50 部の別刷を差し上げます。

【スーパーコンピュータシステム利用案内】

お知らせ	Web ページ
サービス案内、運転状況など	https://www.cc.u-tokyo.ac.jp/
公開鍵登録、マニュアル閲覧など	https://obcx-www.cc.u-tokyo.ac.jp/ (Oakbridge-CX) https://ofp-www.jcahpc.jp/ (Oakforest-PACS) https://reedbush-www.cc.u-tokyo.ac.jp/ (Reedbush)

お問い合わせ内容	お問い合わせ先
利用申込関係	電子メール： uketsuke@cc.u-tokyo.ac.jp スーパーコンピュータシステム利用申込書提出先 〒113-8658 東京都文京区弥生 2-11-16 東京大学情報システム部 情報戦略課研究支援チーム
プログラム相談・システム利用に関する質問	https://www.cc.u-tokyo.ac.jp/supports/contact/#SOUDAN
システムに関する要望・提案	voice@cc.u-tokyo.ac.jp

【IP ネットワーク経由時のホスト名】

シ ス テ ム	ホ ス ト 名
Oakbridge-CX スーパーコンピュータシステム	obcx.cc.u-tokyo.ac.jp 以下のホストの何れかに接続します※ obcx0{1-6}.cc.u-tokyo.ac.jp
Oakforest-PACS スーパーコンピュータシステム	ofp.jcahpc.jp 以下のホストの何れかに接続します※ ofp0{1-6}.jcahpc.jp
Reedbush スーパーコンピュータシステム (Reedbush-H/L)	reedbush.cc.u-tokyo.ac.jp 以下のホストの何れかに接続します※ reedbush-{u{1-4},h1}.cc.u-tokyo.ac.jp

※どのホストに接続しても同じです。

【編集】
東京大学情報基盤センタースーパーコンピューティング研究部門
東京大学情報システム部情報基盤課スーパーコンピューティングチーム
〃 情報戦略課研究支援チーム
【発行】
東京大学情報基盤センター
〒113-8658 東京都文京区弥生2-11-16
(電話) 03-5841-2717 (ダイヤルイン)
(FAX) 03-5841-2708

東京大学情報基盤センター ・ スーパーコンピューティングニュース
Vol.22 No.5(2020.9)

目　　次