

スーパーコンピューティング ニュース

Vol.22 No.6, 2020.11



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值である。

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 一 般	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 ナ ル	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 -l -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 ナ ル	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 -l -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 11 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
11 月 27 日 (金)	9:00 ～ 20:00 までサービス休止	月末処理
12 月 18 日 (金)	9:00 ～ 20:00 までサービス休止	月末処理
1 月 29 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
11 月 26 日 (木) 9:00 ～ 11 月 27 日 (金) 9:00 まで 12 月 17 日 (木) 9:00 ～ 12 月 18 日 (金) 9:00 まで 1 月 28 日 (木) 9:00 ～ 1 月 29 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
11 月 27 日 (金)	9:00 ～ 22:00 までサービス休止	月末処理
12 月 18 日 (金)	9:00 ～ 22:00 までサービス休止	月末処理
1 月 29 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
11 月 26 日 (木) 9:00 ～ 11 月 27 日 (金) 9:00 まで 12 月 17 日 (木) 9:00 ～ 12 月 18 日 (金) 9:00 まで 1 月 28 日 (木) 9:00 ～ 1 月 29 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
11 月 27 日 (金)	9:00 ～ 17:00 までサービス休止	月末処理
12 月 18 日 (金)	9:00 ～ 17:00 までサービス休止	月末処理
1 月 29 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
11 月 26 日 (木) 9:00 ～ 11 月 27 日 (金) 9:00 まで 12 月 17 日 (木) 9:00 ～ 12 月 18 日 (金) 9:00 まで 1 月 28 日 (木) 9:00 ～ 1 月 29 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。
- * 新型コロナにかかわる現状に鑑みて、当面の間大規模 HPC チャレンジを中止いたします。再開、再募集等の際は Web ページ (<https://www.cc.u-tokyo.ac.jp/guide/hpc/>) で適宜お知らせいたします。

システム変更等のお知らせ

(2020.9.3 – 2020.11.4 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)

nano	2.3.1	(2020.09.11)
CP2K	8.0	(2020.09.25)
LAMMPS	3Mar2020	(2020.09.25)
Intel 開発環境 2020 (update2)		(2020.10.30)
Intel Compiler	2020.2.254	
Intel MPI	2019.8.254	
Gromacs (Intel 開発環境 2019 Update5 向け)	2020.4	(2020.10.30)

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)

TeX Live	2020	(2020.09.25)
LAMMPS	3Mar2020	(2020.09.25)
Intel 開発環境 2020 (update2)		(2020.10.30)
Intel Compiler	2020.2.254	
Intel MPI	2019.8.254	
Gromacs (Intel 開発環境 2019 Update5 向け)	2020.4	(2020.10.30)

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。

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

Open MPI (CUDA 10.2.89, GPUDirect 対応版)	4.0.5	(2020.09.28)
CUDA 開発環境	10.2.89	(2020.09.28)
Anaconda3	2020.07	(2020.10.26)

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。

3. その他

3.1 Oakforest-PACS 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。

た。利用方法の詳細は、Oakforest-PACS 利用支援ポータルの Oakforest-PACS 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 2020/09/25
```

```
Usage: show_module [-a]
```

Options:

-a

Show all available module list

2021 年度の利用申込（新規・継続）について

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

2021年度のスーパコンピュータシステム利用申込（新規・継続）は下記のとおり取り扱い
ます¹。利用申込の内容によって、利用申込期限が異なりますのでご注意ください。

なお、この情報は本原稿作成時点での予定ですので、変更する場合があります。最新の情報
については、スーパーコンピューティング部門Webページ（<https://www.cc.u-tokyo.ac.jp/>）
でご確認ください。

また、2021年4月からの利用に関する「利用登録のお知らせ」の送付については、2021年3月
末を予定しておりますので、予めご了解ください。

1. 新規利用申込

1-1. Oakbridge-CX、Oakforest-PACS、Reedbushのパーソナルコース

2021年2月上旬にスーパーコンピューティング部門Webページ

（<https://www.cc.u-tokyo.ac.jp/>）に2021年度版の利用申込サイトを公開します。スーパー
コンピュータシステム利用規程等をよくお読みになり、利用申込を行ってください。

申込は随時受け付けますが、2021年4月初めからのご利用を希望される場合は、利用申込期限
までに手続きを行ってください。

■利用申込期限：2021年2月26日（金）（早めに手続きを行ってください）

1-2. Oakbridge-CX、Oakforest-PACS、Reedbush のグループコース

2021年1月下旬にスーパーコンピューティング部門Webページ

（<https://www.cc.u-tokyo.ac.jp/>）に2021年度版の利用申込サイトを公開します。スーパー
コンピュータシステム利用規程等をよくお読みになり、利用申込を行ってください。

申込は随時受け付けますが、2021年4月初めからのご利用を希望される場合は、利用申込期限
までに手続きを行ってください。

なお、申込状況により利用のお断りもしくは希望セット数どおりの提供ができない場合があります。

■利用申込期限：2021年2月12日（金）（早めに手続きを行ってください）

¹ 企業、若手・女性、大規模 HPC チャレンジ、学際大規模情報基盤共同利用・共同研究拠点及
び HPCI 等の公募制度による利用、講義・講習会等の教育利用及びトライアルユースを除きま
す。

2. 継続利用申込

2-1. Oakbridge-CX、Oakforest-PACS、Reedbushのパーソナルコース

2021年2月上旬に2020年度の登録内容を記載した「継続手続きの案内」を利用者の方に送付します。2021年度も継続利用される場合は、案内に従い、利用申込期限までに手続きを行ってください。

■利用申込期限：2021年2月26日（金）（早めに手続きを行ってください）

2-2. Oakbridge-CX、Oakforest-PACS、Reedbushのグループコース

2021年1月下旬に代表者の方に2020年度の登録内容を記載した「継続手続きの案内」を送付します。2021年度も継続利用される場合は、案内に従い、利用申込期限までに手続きを行ってください。

なお、申込状況により利用のお断りもしくは希望セット数どおりの提供ができない場合があります。

また、提供セット数の決定にあたっては、2020年度利用実績、研究成果登録状況等を参考にさせていただく場合があります。

■利用申込期限：2021年2月12日（金）（早めに手続きを行ってください）

3. 問い合わせ先

〒113-8658

東京都文京区弥生2-11-16（東京大学情報基盤センター内）

東京大学情報システム部

情報戦略課研究支援チーム

E-mail : uketsuke@cc.u-tokyo.ac.jp

東京大学柏キャンパス一般公開 2020

スーパーコンピューティング部門

2020 年度の東京大学柏キャンパス一般公開は、新型コロナウイルスの感染拡大に配慮しオンラインにて 10 月 17 日(土)～31 日(土)に開催されました。情報基盤センターでは「人類と地球を護るスーパーコンピューティング」をテーマに、スーパーコンピュータの入門編から、COVID19 感染症対策などスーパーコンピュータを利用した最新の研究まで、スーパーコンピュータに関する様々な話題を動画により紹介しました。10 月 23 日(金)と 24 日(土)にはウェブ会議システム(Zoom)を利用して参加者の質問にお答えしました。

■ 柏キャンパス一般公開 2020 情報基盤センター

https://www.cc.u-tokyo.ac.jp/kashiwa_open/

■ 東京大学柏キャンパス一般公開 2020(オンライン開催)

<https://park-ssl.itc.u-tokyo.ac.jp/utkk/opc2020/>

■ 東京大学柏キャンパス一般公開 2020 参加部局イベント案内

<https://park-ssl.itc.u-tokyo.ac.jp/utkk/opc2020/oc15.html#b06>

【開催期間中ののべ数】

柏キャンパス一般公開 2020 情報基盤センターウェブ アクセス数 3,166 件

動画 視聴回数 1,002 回

「Q&A」(ウェブ会議システム) 参加者 8 名

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部門

Supercomputing Division,
Information Technology Center,
The University of Tokyo

ENGLISH | サイト内検索

お問い合わせ | リンク | サイトマップ

ホーム | スパコン | 利用案内 | サポート | FAQ | 研究会・イベント | 広報・刊行物 | 研究成果 | 研究部門について

ホーム > 柏キャンパス一般公開2020

柏キャンパス一般公開2020

スーパーコンピュータとは？

計算による科学

計算の科学
計算のための科学

将来のスーパーコンピューティング

中高生のためのスーパーコンピューティング

柏キャンパス一般公開2020
東京大学情報基盤センター
～人類と地球を護るスーパーコンピューティング～

大規模災害や新型感染症、人類の直面する様々な問題を、
スーパーコンピュータが解明します！

スパコンって何？から、COVID19感染症対策などスパコンを利用した最新の研究まで、スパコンに関する様々な話題を動画によりご紹介します！ご質問のある方は、YouTubeの各動画コメント欄よりお願いします！

スーパーコンピュータとは？(入門～やや専門家向け)

2020年度 インターン・後期 東京大学情報基盤センター「若手・女性利用者推薦」採択課題

スーパーコンピューティングチーム

東京大学情報基盤センター（以下、センター）では、若手研究者（2020年4月1日時点において40歳以下、学生を含む）及び女性研究者（年齢は問わない）による、スーパーコンピュータ、大規模ネットワーク機器などの大型計算資源を使用した研究を対象とした公募型プロジェクトを実施しています。センターの教員による審査の上、年間で数十件の優れた研究提案課題を採択する予定です。採択された課題では申請した計算資源を無料で使用することができます。

前期・後期に募集を行う一般枠と、学部学生・大学院生を対象とし、主に夏期における利用を想定したインターン制度があります。一般枠（前期・後期）では、1人で行う研究課題を募集します。一般枠の課題は、1年または半年単位（後期は半年のみ）の実施となります。インターン制度では、1人で行う研究課題または2人以上のメンバーで構成された研究グループで行う研究課題を募集します。

一般枠で新規に採択された課題のうち、特に優れた課題で「学際大規模情報基盤共同利用・共同研究拠点（JHPCN）」の萌芽型共同研究課題の条件を満たすものについては、本センターより同拠点萌芽型共同研究課題として推薦する予定です。同拠点共同研究課題審査委員会で審査の上、JHPCNの萌芽型共同研究課題としても採択された場合、毎年7月に開催されるJHPCNのシンポジウムにて発表の機会が与えられる場合があります。本制度に採択された課題は終了後、得られた成果をもとに、「学際大規模情報基盤共同利用・共同研究拠点（JHPCN）」の公募型共同研究（一般課題、国際課題、企業課題）等へと進展することが大いに期待されます。なお、インターン制度で採択された課題はJHPCN萌芽型共同研究課題としての推薦は行いません。

このたび、以下の基準による厳正な審査のうえ、2020年度インターンは5件の課題を、後期は4件の課題を採択いたしました。

- 本制度が提供する計算機システムを利用することで、学術的にインパクトがある成果を創出できると期待される提案を積極的に採択します。
- スーパーコンピュータの利用環境の改善に寄与すると期待されるソフトウェア開発に関する提案も歓迎します。
- 現状の環境にとどまらず、メニーコア、10万コア超える超並列環境など、将来の先端的なスーパーコンピュータ環境を目指した提案は特に歓迎します。

本制度の詳細は、以下のHPをご覧ください。

<https://www.cc.u-tokyo.ac.jp/guide/young/>

■ 2020年度 インターン 採択課題

課題名	単眼カメラによる距離推定技術に関する研究
代表者名（所属）	牛 昭峰（奈良先端科学技術大学院大学 情報科学研究科）
利用システム名	Reedbush-H
実施期間	半年
Depth estimation plays a significant role in 3D reconstruction and scene understanding. However, depth cameras or depth sensors have lots of limitations, such as high power consumption and restricted operating ranges. These limitations will be solved if depth estimation can be achieved based on RGB images only. The main purpose of my research is to improve the performance of depth estimation and obtain highly accurate depth information based on RGB images.	

課題名	繰り込み群による冷却極性分子系での量子スピン液体実現可能性の探索
代表者名（所属）	福井 毅勇（東京大学大学院 理学系研究科）
利用システム名	Oakforest-PACS Oakbridge-CX
実施期間	半年
本課題は、光格子中に閉じ込めた冷却極性分子系において量子スピン液体状態が実現するか否かを汎関数繰り込み群の大規模並列計算により数値的に明らかにするものである。冷却分子系は分子間の双極子相互作用のために、最近接相互作用のみを考えた固体物理における量子スピン液体模型から基底状態が大きく変わりうる。双極子型の相互作用をする量子スピン液体の模型（Kitaev 模型）がスピン液体状態を取るのか否か調べることは、模型の非可解性と相互作用の長距離性から困難であった。本課題は、汎関数繰り込み群法によりこの問題を解き明かし、実験の提案を目指すものである。	

課題名	極低温推進薬の軌道上貯蔵・輸送に向けた減圧沸騰現象の解明と予測手法の開発
代表者名（所属）	谷 和磨（東京大学大学院 工学系研究科）
利用システム名	Oakbridge-CX
実施期間	半年
軌道上での極低温推進薬管理技術の確立に向けた、大型タンク内における沸騰凝縮を伴う気液二相熱流動現象を解明する。界面張力が重力を卓越する宇宙空間では相変化による液体挙動や圧力変動の経験的な予測が難しく、適切に流体機器を運用できない危険性がある。そのため、昨今計画されている長期の宇宙ミッションの実現には、こうした相変化現象を予測する必要がある。しかし、実機寸法の大型タンクでの沸騰試験データが不足している上、極低温推進薬の熱流動特性に基づいた定量的な大型タンク内現象予測手法は、沸騰現象（小寸法）とタンク（大寸法）のスケールの違いから計算コストも高く未だ確立されていない。本利用課題では、VOF法をベースにした自由表面流用数値流体解法であるCIP-LSMを改良し、大型タンク内での相変化現象を予測する手法を開発する。地上での極低温流体用の大型タンクを用いた減圧沸騰試験の結果と比較することで予測手法を評価する。こうして構築された数値流体解法を用いて、微小重力環境での相変化を伴う熱流動現象を明らかにする。	

課題名	電氣的・熱的效果によるキャビティ音制御の数値解析
代表者名（所属）	大竹 克也（豊橋技術科学大学 大学院 工学研究科）
利用システム名	Oakbridge-CX
実施期間	半年
新幹線の台車部などはキャビティと呼ばれる窪み部を有し、窪み部入口より発生した渦により自励振動が発生し、強い騒音が発生する場合があります、低減手法の確立が課題となっている。近年、キャビティ音の制御にプラズマアクチュエータ（PA）が注目されている。また、PA のオン・オフを切り替える間欠制御により、キャビティ音の低減に加え消費電力の削減も期待できる。PA をオン・オフに切り替えた際の流れ場に注目し、発生音のメカニズムも説明出来る。また、騒音の低減に加え、キャビティ音の音響エネルギーを有効利用する方法の一つとしては、熱音響ヒートポンプ現象の利用が有益であると考えられる。熱音響ヒートポンプ現象の利用により、騒音低減と狭い流路を有する多孔体や平板列（スタック）に生じた温度差を用いた発電が可能であると考えられる。本課題では、空力音直接数値計算を用いて、PA によるキャビティ音のフィードバック制御を行い、PA の適切な制御条件を明らかにすることを目的とする。また、キャビティまわりの流体音響解析とスタックの熱伝導解析を連成した解析を実施し、キャビティ音により駆動される熱音響ヒートポンプ現象の解明を目的とする。	

課題名	階層型直交格子法と壁面モデルを用いた LBM による航空機高揚力装置の空力音響解析
代表者名（所属）	前山 大貴（東京大学大学院 工学系研究科）
利用システム名	Oakbridge-CX
実施期間	半年
近年の環境問題に対する意識の高まりから、より低騒音・低燃費な航空機の実現が望まれている。航空機の空力設計では、主翼コード長ベースで 10 の 7 乗から 10 の 8 乗にもなる高レイノルズ数条件での空力解析が必要となり、流れ場から発生する空力騒音の予測には、非定常な境界層剥離や再付着現象が支配的な流れ場を高精度に解析することが要求される。音の直接計算が可能な非定常流体解析手法として、格子ボルツマン法 (LBM) が注目されている。LBM は計算アルゴリズムが単純で、大規模並列計算に適しているという特徴をもつ一方で、直交格子法を用いた数値解析手法であるため、壁面近傍に発達する乱流境界層の解像が困難である。航空機実機レベルの高レイノルズ数乱流境界層流れを高精度に解析するためには、境界層内層域の乱流をモデル化することが必須である。本研究では、任意形状に対して適用可能な LBM の壁面モデルを新たに開発・実装し、航空機機体空力騒音の主要な発生源である高揚力装置の空力音響解析を行う。複雑な形状に対しても完全自動で格子を生成可能な階層型直交格子法と組み合わせることで、LBM を用いた航空機空力騒音の予測手法の確立を目指す。	

■ 2020年度 後期 採択課題

課題名	PaCS-MD に基づく効率的リガンド結合経路探索法の開発
代表者名 (所属)	原田 隆平 (筑波大学 計算科学研究センター)
利用システム名	Reedbush-H
実施期間	半年
本研究では、リガンドと標的タンパク質の結合過程を効率的に抽出する計算手法を開発する。具体的には、タンパク質の遷移経路をレアイベントとして効率的に抽出する目的で開発した「Parallel Cascade Selection Molecular Dynamics (PaCS-MD)」をリガンド結合経路探索に拡張する。PaCS-MD は、遷移確率が高い分子構造を選択し、短時間 MD により構造リサンプリングを繰り返すことで、効率的に遷移経路を探索するレアイベントサンプリング法である。PaCSMD を標的タンパク質とリガンドの複合系に適用可能なように拡張する。拡張にあたり、熱揺らぎの範囲において結合経路探索を加速するため、リガンドが複数存在する高密度環境下で PaCSMD を実行する。最終的に、リガンド結合経路を効率的に探索する計算手法として PaCS-MD を拡張し、Ligand-Docking-PaCS-MD (ld-PaCS-MD) を開発する。	

課題名	Coarse-graining Model for Monte Carlo Simulation of Spin Configurations in Ferromagnets
代表者名 (所属)	李 其放 (東京大学 新領域創成科学研究科)
利用システム名	Oakbridge-CX
実施期間	半年
Spin configuration affects the the magnetic moment and related physical properties in solid-state devices. Markov chain Monte Carlo (MCMC) simulation is a common approach to study the dynamic behaviors of spins. In this research, long-ranged interactions are simplified by stochastic cutoff algorithm. A new algorithm of coarse-graining will be developed to reduce the degree of freedoms of spins in the entire system. This algorithm will also be applied for design of ferromagnet devices.	

課題名	降着円盤における微小スケール乱流の特性解明
代表者名 (所属)	川面 洋平 (東北大学 学際フロンティア研究所)
利用システム名	Oakforest-PACS
実施期間	半年
降着円盤は天体の周りを回転しながら落下するプラズマの流れである。特にブラックホールにおける降着円盤は、直接観測することが困難なブラックホールの情報を間接的に教えてくれる重要な存在である。降着円盤は磁気回転不安定性 (MRI) によって駆動される乱流状態になっていると考えられているが、この乱流の持つ特性にはまだ未解明な点が多く存在する。我々の最近の研究によって、降着円盤の観測結果を解釈するためには、乱流における圧縮的な揺動と非圧縮的な揺動の比を知る必要があることが明らかになっている。本研究の目標はこの比を数値計算によって求めることである。しかし圧縮的な揺動と非圧縮的な揺動の比を求めるためには、これまで行われてきた MRI 乱流のシミュレーションを遥かに上回る解像度のシミュレーションが必要となる。そこで本研究では、我々が開発した超高並列性能を持つ電磁流体力学コードを用いて圧縮的な揺動と非圧縮的な揺動の比を得ることに挑戦する。	

課題名	Large-eddy simulations of nearshore offshore wind farms and their interactions with atmospheric boundary layer
代表者名 (所属)	Goit Jay Prakash (近畿大学 工学部)
利用システム名	Oakbridge-CX
実施期間	半年
Offshore wind energy has received larger interest, motivated by the fact that offshore sites are generally characterized by higher and uniform wind speeds, thus, leading to higher capacity factors and lower fatigue loads experienced by turbines. The current study employs large-eddy simulations to investigate the effect of onshore terrain on the evolution of ABL—when it transitions from land to sea—and evaluate how that influence the performance of offshore wind farms in the nearshore region.	

以上

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。



① 「研究成果」をクリック

② 研究成果ページの「研究成果登録」をクリック

研究成果登録

研究成果の登録をお願いいたします。

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。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、センター発行の広報誌及びWebページへの掲載に使用させていただきますことをご承諾ください。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、利用の皆様には何卒ご協力の程宜しくお願い申し上げます。

お使いのアカウント名と登録メールアドレスを入力し、登録しようとする業績を選択してください。

アカウント名	
メールアドレス	
登録したい業績	☐ 論文 ☐ 口頭・ポスター発表 ☐ 著書 ☐ 受賞情報

③ アカウント名(利用者番号)及びメールアドレスを入力し、登録したい業績を選択

④ 「ログイン」をクリックし、成果登録ページで研究成果の登録をお願いいたします

JavaScriptを有効にしてお使いください。

成果登録内容の削除などの依頼、登録メールアドレスが不明といった際のお問い合わせは

Email: kenkyu_shien.adm@gs.mail.u-tokyo.ac.jp までお願い致します。

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部門
Supercomputing Division, Information Technology Center, The University of Tokyo

8・9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
8月	1,371	295	8,780	67	1,450	63,090	27,508	1,350	25,301,247	9,449	136	949	498,413	938.4	69.0
9月	1,535	299	11,804	48	1,152	80,846	39,017	1,399	26,814,117	3,073.18	119	740	630,436	1,030.7	75.8
2019年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			91,551	484	9,841	561,665	414,100			37,147.86	2,158	5,935	5,353,308		

・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

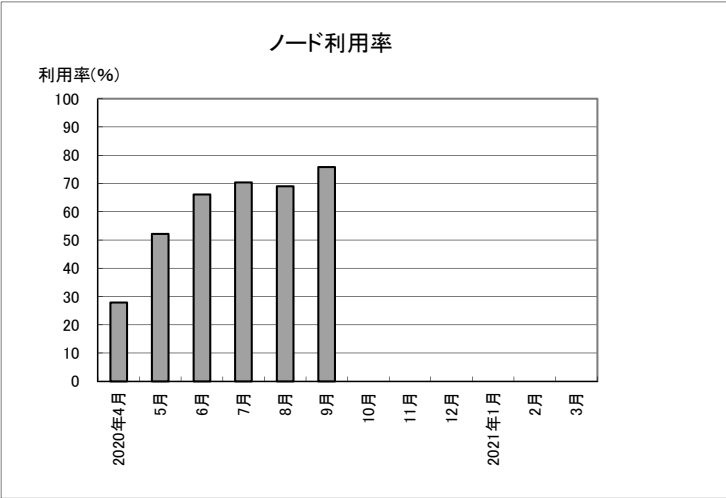
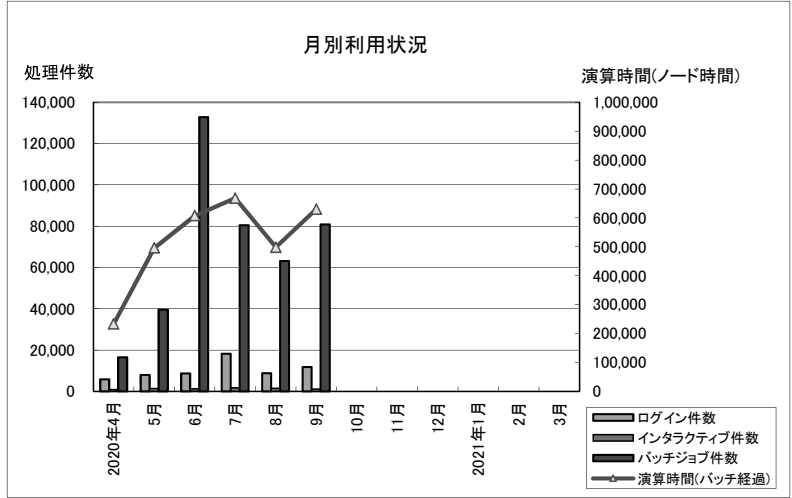
・ログイン(実CPU)： コア時間単位

・2019年9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	38,464	54,218	3,927	6,326,936	1,829.07	709	214	2,840,983	4,423.7	53.9
5月	1,802	446	9,554	247	254	59,496	64,493	3,220	5,455,455	2,698.20	502	152	3,571,465	4,885.8	59.5
6月	1,611	455	11,840	151	358	47,010	67,932	3,361	5,858,067	1,957.58	342	161	3,415,672	4,831.4	58.9
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
8月	1,617	401	12,139	315	618	83,908	70,019	3,854	6,332,050	11,191.93	685	514	4,726,977	6,611.0	80.5
9月	1,732	420	9,811	293	795	52,222	56,925	3,964	6,299,074	3,237.18	667	1,390	4,125,115	6,447.3	78.5
2019年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			141,178	4,381	5,030	681,873	924,674			74,042.73	8,450	3,884	51,301,899		

・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

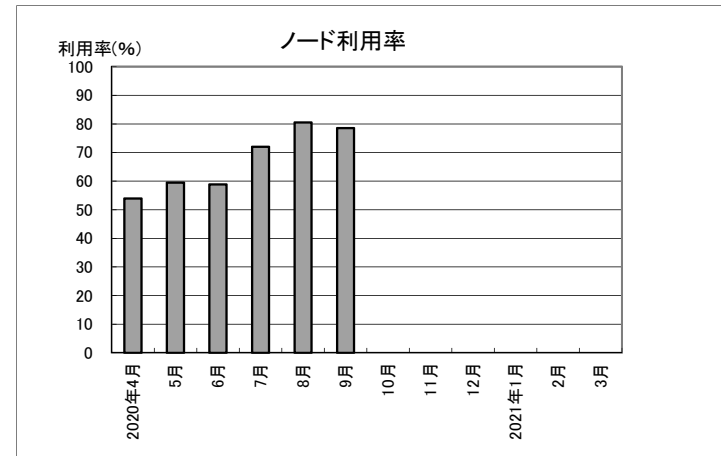
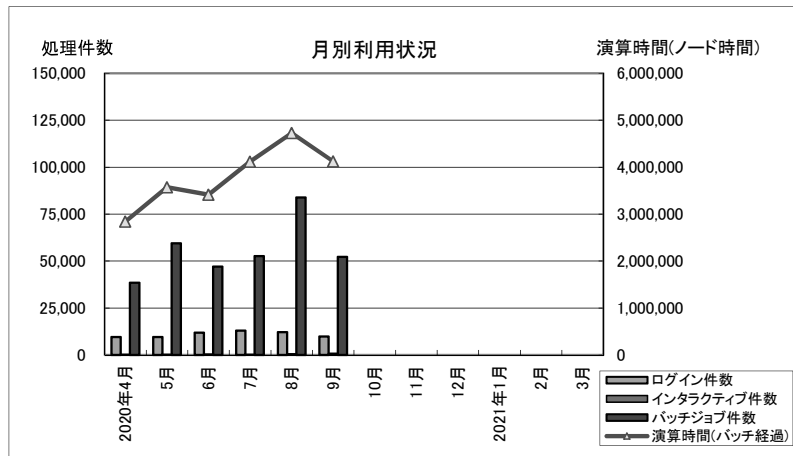
・2019年9分は合計に含まない

・ノード利用率：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率。計算式＝利用ノード時間÷サービスノード時間×100

・2020年4月～6月、下線部分の値に誤りがございました。お詫びして訂正いたします。



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
8月	999	171	4,032	0	897	5,165	6,535	157	298,649	7	0	1,213	34,188	72.2	60.2
9月	986	161	4,364	0	742	6,403	8,054	159	321,098	12.18	0	1,077	26,655	43.3	36.1
2019年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	8,526	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			72,495	34	6,395	114,518	112,230			150.74	22	5,925	474,811		

・接続時間: ログイン時間の累計

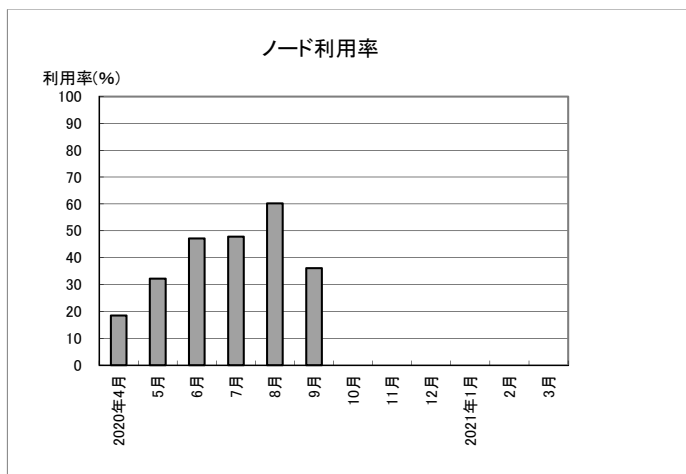
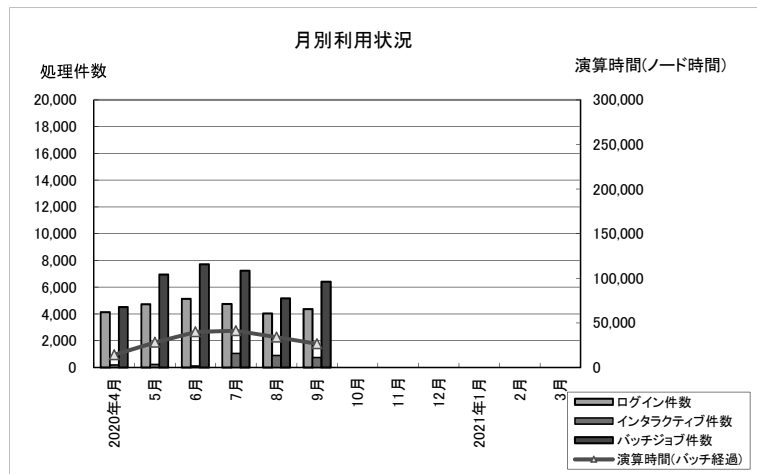
・ログイン(実CPU): コア時間単位

・2019年9月分は合計に含まない

・ノード利用率: 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ヶ月の稼働時間

・ノード利用率: サービスノードに対する利用率。 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
8月	58	1,551	64	10,739	25.6	47.4
9月	57	1,757	49	20,902	32.7	60.6
2019年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	838	24,784	984	232,839		

・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、

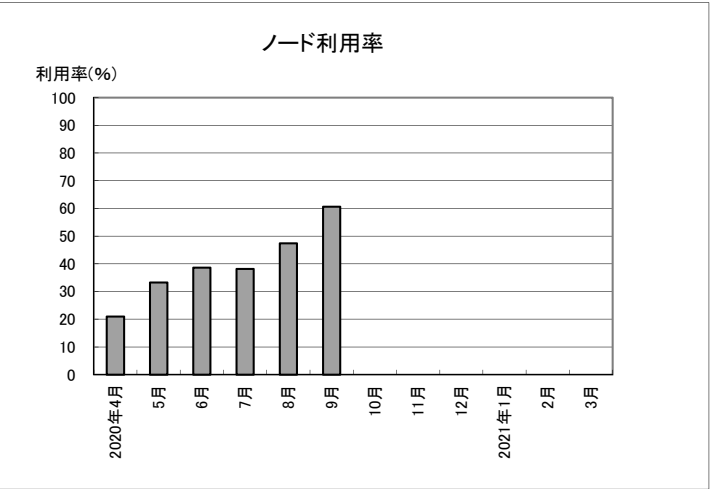
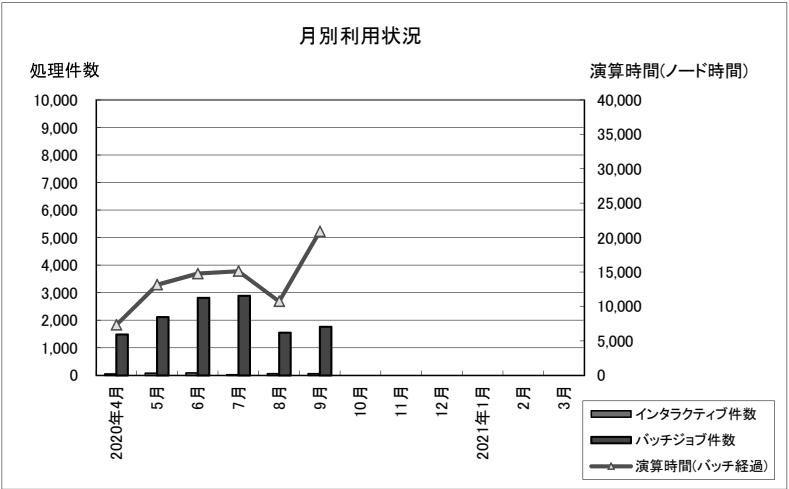
ログイン(実CPU)はReedbush-Uと共通。

・2019年9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ヶ月の稼働時間

・ノード利用率: サービスノードに対する利用率。 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		

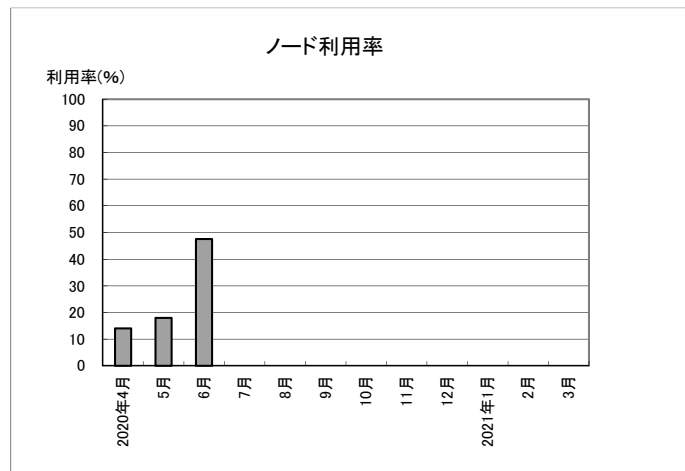
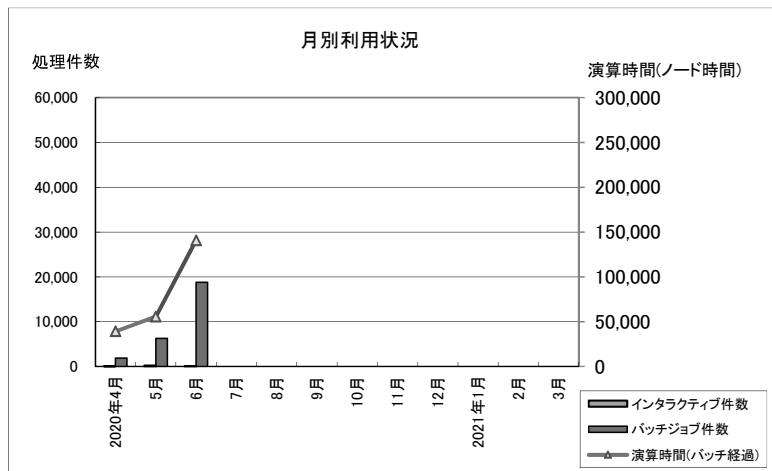
・登録者数、実利用者数、ログイン件数、接続時間、ファイル使用量、
ログイン(実CPU)はReedbush-Uと共通。

・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



温水冷却の効率に関する定量的評価の試み

庄司 文由、野中 丈士

理化学研究所計算科学研究センター

塙 敏博

東京大学情報基盤センター/最先端共同 HPC 基盤施設

1. はじめに

近年、温水冷却技術は、HPC システムの冷却にかかるエネルギー効率を改善するために、多くの HPC センターおよびデータセンターで採用されている。温水冷却においては、CPU 等の冷水温度を高く設定することで、外気による自然な冷却が促されるため、冷凍機等を駆動するための電力が削減できる[1][2]。その一方で、CPU 等のシリコンから構成される半導体は、高温で動作させればさせるほど、漏れ電流の増加により、消費電力が増加することが知られている。また、近年の CPU は、駆動温度が一定の水準を超えると、上記の消費電力の増加や故障率の上昇を抑制するために自動的にクロック周波数や電圧を下げる機構 (Dynamic Voltage and Frequency Scaling, DVFS) を備えている[3][4][5]。

温水冷却の効果を正しく評価するためには、冷却のための電力を削減できるというポジティブな点に加え、上記のような消費電力の増加や、クロック周波数の低下による演算性能の悪化のようなネガティブな点も等しく考慮に入れる必要がある。演算性能が低下すれば、その分より長く計算することにより、HPC システムが消費するエネルギーの総量が増加する可能性があるためである。特にクロック周波数の低下による演算性能への影響は、大規模なジョブでより深刻になると予想される。クロック周波数は CPU 毎に独立にコントロールされるが、CPU (プロセス) 間で同期を取る場合、最も遅い CPU (プロセス) に律速されるからである。

以上を踏まえ、本研究では、単体および複数の CPU を使う様々なジョブを、異なる冷水温度で実行し、冷却設備および HPC システムの消費電力の増加と演算性能の低下を定量的に分析する。左記の分析に基づき、効率的な施設運用の実現に向けた運用手順の確立を目指す。

本稿では、今回の大規模 HPC チャレンジによりこれまでに得られた成果の概要を紹介する。なお、本研究の成果の一部は、ISC2020 Research Poster[6]および IEEE Cluster2020 paper[7]として採録されているので、詳細についてはそれらを参照いただきたい。

2. 実験方法

冷水温度の変更には施設側のオペレーションが必要で、温度が安定するまでに時間がかかるため、今回の実験では、それぞれの冷水温度において、同じベンチマークプログラムのセットを実行する方法を採用した。ベンチマークプログラムとしては、演算インテンシブなものとして LINPACK[8]、メモリアクセスインテンシブなものとして HPCG[9]、ウィークスケーリングを評価するものとして GeoFEM をそれぞれ用いた。LINPACK (シングルノード及びマルチノード版) と HPCG は Intel Parallel Studio XE (Ver. 2019.5.075) に配布されてるものを用いた。

最初に計算ノード毎の演算性能や消費電力等の特性のばらつきを評価するためにシングルノ

ードの LINPACK を実行した。その後、それぞれのベンチマークプログラムについて、複数のサイズのマルチノードジョブを実行した。それぞれのケースで、演算性能、クロック周波数、消費電力等を測定した。用いた冷水温度は、9℃、12℃、18℃である。ちなみに、25℃のケースも試みたが、大多数ノードでの LINPACK 実行中に冷却システムが追いつかなくなる事象に遭遇したため測定を断念した。通常運用時の設定温度は 12℃とのことである。測定した冷水温度、ノードサイズ、測定回数を表 1 に示す。

	ノードサイズ	9℃	12℃	18℃
LINPACK	1	5120	4096	4097
MP-LINPACK/HPCG/GeoFEM	128	5/6/6	7/7/7	5/6/6
	256	5/5/5	6/6/6	5/5/5
	512	5/5/5	6/6/6	5/5/5
	1024	5/5/5	6/6/6	5/5/5
	2048	5/5/5	6/6/6	5/5/5
	4096	5/5/5	6/6/6	5/5/7

表 1 ベンチマーク毎の各冷水温度におけるノードサイズと測定回数

3. 結果

最初に冷水温度の変化に対する、冷却設備の効率の変化を見る。

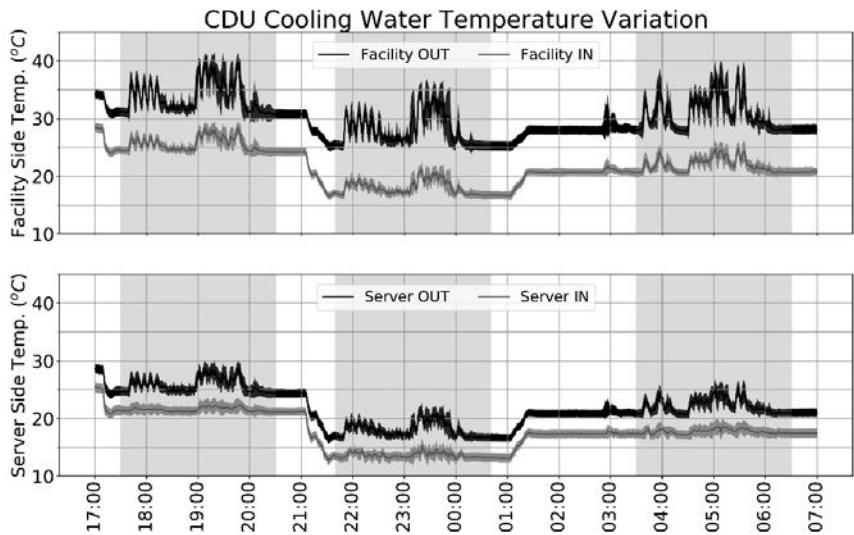


図 1 冷却設備側（上）と HPC システム側（下）の冷水温度の変化

図 1 の上段は冷却設備側、下段は HPC システム側の、inlet と outlet の冷水温度の推移を示している。それぞれの温度のケースについて、同じベンチマークプログラムのセットを実行したが、時間にすると概ね 3 時間であった。図 1 の濃い色の背景になっている部分はその時間帯を示している。したがって、この 3 時間において、冷却設備および HPC システムが消費したエネルギーを用いて冷却設備の効率の変化を評価する。

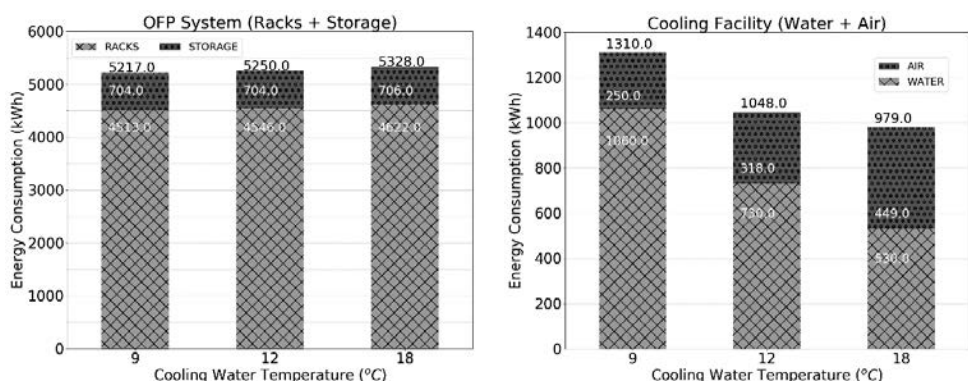


図 2 冷水温度の変化に対する HPC システム (左) および冷却システム (右) の消費エネルギーの変化

図 2 に冷水温度の変化に対する HPC システム (左) および冷却システム (右) の消費エネルギーの変化を示す。冷水温度が高くなれば、より外気によって自然に冷却されやすくなるので、冷却にかかるエネルギーは減少する。その一方で、HPC システムは冷水温度が上昇するとリーク電流が増加するため、消費電力および消費エネルギーが増加すると見込まれるが、今回の実験においては、3つの温度において、消費エネルギーはほぼ変わらなかった。これは後述する DVFS によって、消費電力および消費エネルギーの増加が抑制されたためと考えられる。

今回は、冷却システムのエネルギー効率を評価するための指標として、COP (Coefficient of Performance) と PUE (Power Usage Effectiveness) を用いた。本稿における COP は、冷却に消費したエネルギーに対し、どれだけの熱エネルギーを除去できたかを示すものであり、大きいほど効率が良いことを示す。OakForestPACS が消費したエネルギーがすべて熱に変換され、それが冷却システムですべて冷却されたと仮定し、以下の式で COP を定義する。

$$COP = \frac{OFP\ System\ (Racks)}{Cooling\ Facility\ (Water)}$$

一方、PUE は冷却システムも含めたサイト全体の消費エネルギーに対する IT システムの消費エネルギーの比である。冷却等にかかるエネルギーがゼロのような理想的なケースでは 1 となり、大きくなるほど効率が悪いことを示す。本稿では以下の式で定義する。

$$PUE = \frac{OFP\ Sytem + Cooling\ Facility}{OFP\ System}$$

それぞれの指標について、図 2 に示した数値で計算したものが、図 3 である。

PUE および COP 共に、冷水温度が高いほど効率が良くなることを示している。特に COP については、9°C と 18°C のケースを比較すると 2 倍以上効率が改善されたことがわかる。PUE も 18°C のケースでは 1.2 を下回るスコアになっており、9°C のケースと比較して大きく改善されることがわかった。これらの成果の詳細については[6]を参照されたい。

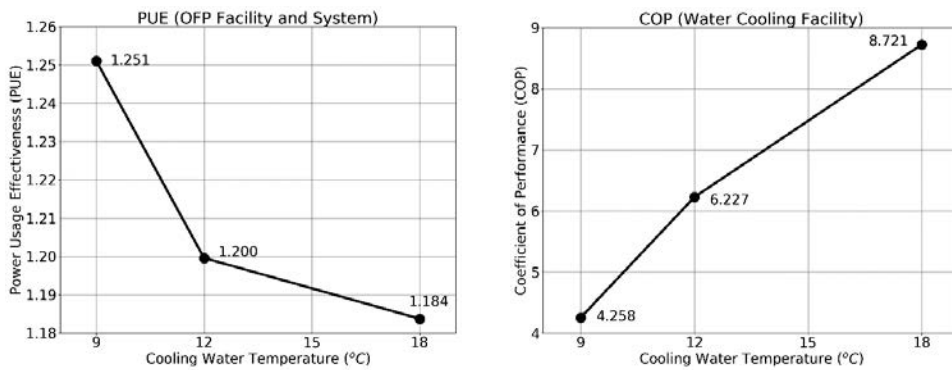


図 3 冷水温度の変化に対する PUE および COP の変化

次に、冷水温度の変化によるシングルノード性能への影響を見る。

まず 12°C のケースで LINPACK を実行した際の実行性能の分布(図 4)と演算性能と CPU の温度の相関(図 5)を見る。

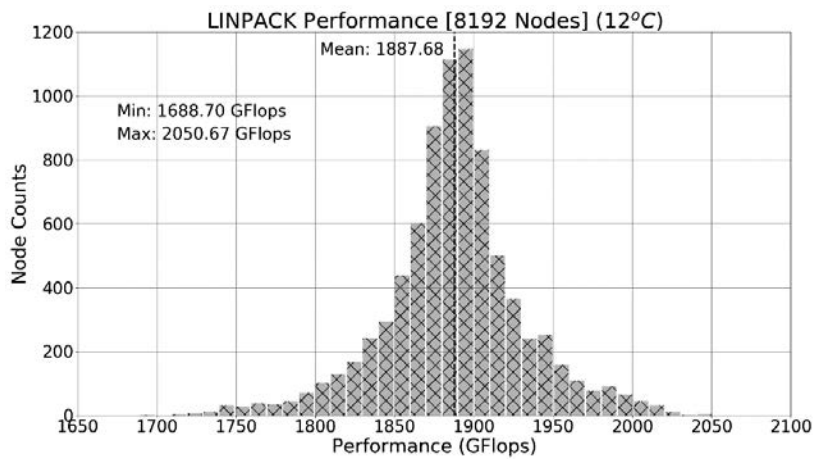


図 4 冷水温度 12°C のケースの LINPACK 性能の分布

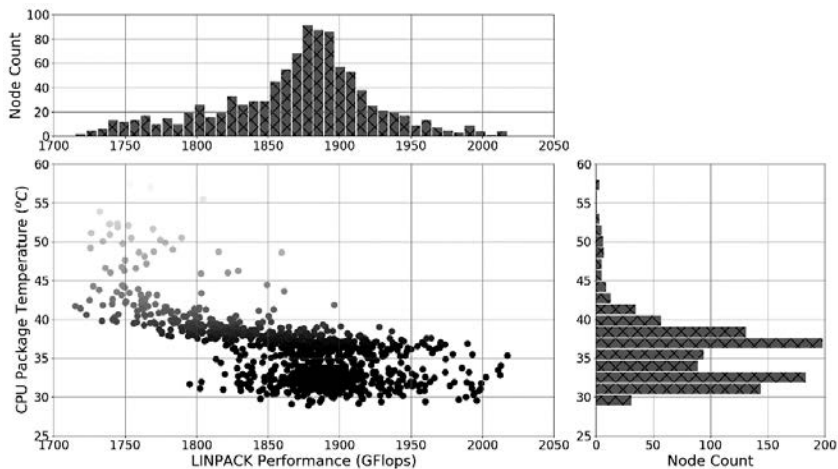


図 5 冷水温度 12°C のケースの LINPACK 性能と CPU の温度の分布

図 4 に示す LINPACK 性能の分布については、通常運用時の 12℃ のケースで既に平均値に対し 20% 近くの幅で性能のばらつきがあることがわかる。図 5 からは、CPU 毎に LINPACK 性能に加えて、温度も広い範囲で分布していることがわかる。全体としては温度が低い CPU ほど高い演算性能を出す傾向にあるが、少数ながらも 45℃ 以上で性能が低いグループも存在しており、マルチノードジョブの際にこれらのノードが含まれると全体の性能に影響が出ることが懸念される。

次に、9℃ と 18℃ のケースを LINPACK 性能とエネルギー消費の観点で比較する。

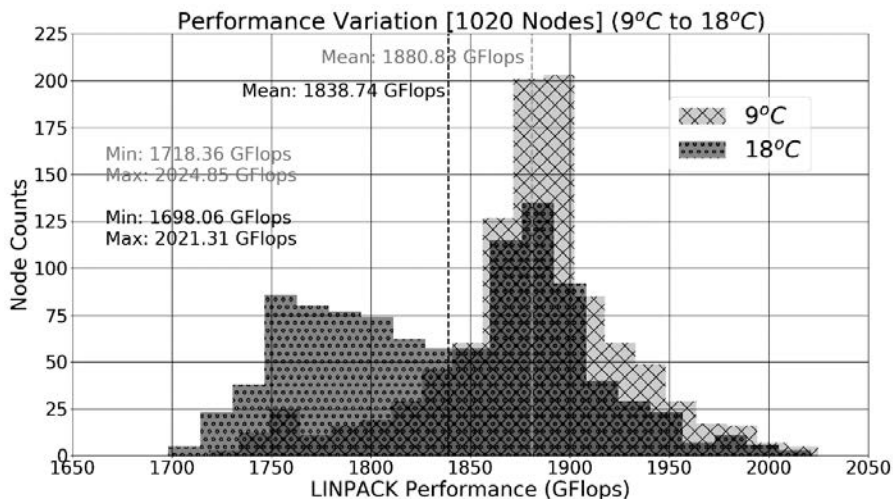


図 6 9℃と18℃のケースの比較 (LINPACK 性能)

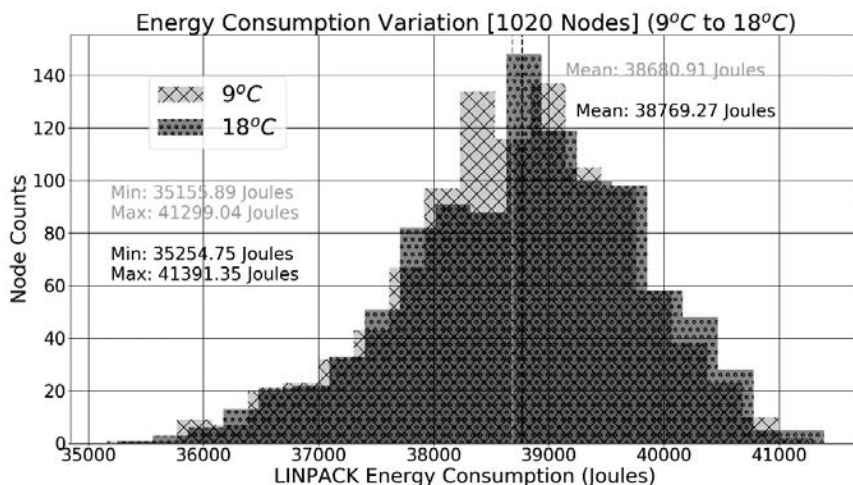


図 7 9℃と18℃のケースの比較 (エネルギー消費)

図 6 は冷水温度 9℃ と 18℃ のケースを LINPACK 性能の観点で比較したグラフである。18℃ のケースでは、9℃ のケースに比べ、明らかにパフォーマンスが低い CPU が増えており、平均値および最低値共に 9℃ のケースよりも低くなっている。この結果は冷水温度に対する感受性が CPU 毎に大きく異なることを示しており、特に最低値が下がることにより、マルチノードジョブの性

能に対するインパクトはより大きくなることを示唆する。同期を取るケースなど、一番遅いノードの性能に律速されるからである。一方で、エネルギー消費の分布は9℃と18℃でほとんど変わらない。これは DVFS がうまく機能し、エネルギー消費を抑えた結果だと考えることができる。つまり、高い冷水温度では、DVFS によりクロック周波数や駆動電圧が抑制されやすくなることで、演算性能は悪化（つまり演算時間は増加）するものの、エネルギー消費の増加は抑制できるということだと考えられる。これらの成果の詳細については[7]を参照されたい。

最後に、マルチノードでの性能の評価を行う予定だったが、実行時の設定が不十分だったため、特定のノード数を用いるジョブに対し、同じノード群を用いた実行ができておらず、異なる冷水温度の間で比較できるデータが十分に得られていなかった。上述のとおり、今回の LINPACK のシングルノード性能の評価結果は、冷水温度を上げることによる性能低下の影響は、マルチノードジョブでより深刻になることを示唆しており、今後機会があればマルチノードジョブに対する詳細かつ定量的な評価を行いたい。

5. まとめ

温水冷却技術を正しく評価するためのひとつの方法として、冷水温度を変化させつつ、特定のベンチマークを実行し、その際の HPC システムの消費エネルギーおよび演算性能と冷却システムの消費エネルギーから、全体のエネルギー効率を評価する方法を提案し、OakForestPACS を用いて実証実験を行った。

冷却システムのエネルギー効率については、冷水温度が高い方がより高い効率で冷却できること、特に9℃と18℃のケースを比較すると、倍以上の効率が実現できることがわかった。また PUE についても 5%以上改善された。

シングルノードにおける LINPACK の実行により、各 CPU の演算性能と駆動温度の分布を調査した。通常運用時の冷水温度 12℃のケースでも、演算性能、駆動温度共に広く分布しており、演算性能については平均値に対し 20%近くの幅でばらつきがあることがわかった。冷水温度 9℃と18℃のケースを比較すると、演算性能が劣化する CPU が増加し、18℃のケースでは平均値、最低値共に低下することがわかった。この結果は、冷水温度を上げることによるマルチノードジョブの性能に対するインパクトはより大きくなることを示唆している。

今回の実験により、冷水温度を上げることにより、冷却システムの消費エネルギーを抑制できるというメリットがある一方で、演算性能の低下というデメリットがあることについて、定量的に評価することができた。今後、評価の範囲をマルチノードジョブにも拡大し、冷水温度の変化に対する、性能の劣化（とそれを補完するための追加のエネルギー消費）を考慮したエネルギー効率の評価手法を確立したい。それができれば、冷水温度をパラメータとして、エネルギーやターンアラウンドタイム等それぞれの運用者が着目する指標を最適化した運用が可能となる。今後もその目標に向けて研究を進めていきたい。

6. 謝辞

本研究は、大規模 HPC チャレンジという制度がなければ到底実現できなかった。このような機会を与えていただいたことに深く感謝致します。また、今回の実証実験を行う上で、JCAHPC の関係者、運用主体の富士通株式会社の皆様に大変なご苦勞をおかけしました。お詫びと共に深く感

謝申し上げます。GeoFEM のコードの提供および OFP 上での実行に関するサポートについて、東京大学情報基盤センター 中島教授および Intel 株式会社 堀越氏に多大なご協力をいただきました。

参 考 文 献

- [1] S. Zimmermann, et.al., “Aquasar: A hot water cooled data center with direct energy reuse,” Energy Volume 43, Issue 1, July 2012, Pages 237-245.
- [2] H. Huber et al., “Case Study: LRZ Liquid Cooling, Energy Management, Contract Specialities,” 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, Salt Lake City, UT, 2012, pp. 962-992, doi: 10.1109/SC.Companion.2012.123.
- [3] Intel, “Intel Xeon Phi processor: Your path to deeper insight.” [Online]. Available: <https://www.intel.com/content/www/us/en/processors/xeon/xeon-phi-processor-product-brief.html>
- [4] K. D. Stroup and P. Peltz, “Measuring and mitigating processor performance inconsistencies,” in Cray User Group Conference (CUG 2019), 2019. [Online]. Available: <https://cug.org/proceedings/cug2019proceedings/includes/files/pap124s2-file1.pdf>
- [5] R.Schöne, T.Ilsche, M.Bielert, A.Gocht and D.Hackenberg, “Energy efficiency features of the Intel Skylake-SP processor and their impact on performance,” arXiv preprint arXiv:1905.12468, 2019.
- [6] J. Nonaka, T. Hanawa and F. Shoji, “Analysis on the Impact of the Cooling Water Temperature on the HPC System and Facility - Case Study: Oakforest-PACS (OFP) System and Facility,” ISC2020, Frankfurt, Germany, 2020. Available: <https://2020.isc-program.com/presentation/?id=post131&sess=sess325>.
- [7] J. Nonaka, T. Hanawa and F. Shoji, “Analysis of Cooling Water Temperature Impact on Computing Performance and Energy Consumption,” 2020 IEEE International Conference on Cluster Computing (CLUSTER), Kobe, Japan, 2020, pp. 169-175, doi: 10.1109/CLUSTER49012.2020.00027.
- [8] <http://www.netlib.org/benchmark/hpl/>
- [9] <https://www.hpcg-benchmark.org/>

縦長行列の QR 分解に対する各種アルゴリズムの比較：

Oakforest-PACS 上での性能評価

深谷 猛

北海道大学 情報基盤センター

1. はじめに

A を $m \times n$ 列フルランク実行列で、縦長 ($m \gg n$) であるとする。本稿では、このような行列の QR 分解 (Reduced QR 分解, Thin QR 分解) [1]

$$A = QR$$

の計算を考える。ただし、 Q は $m \times n$ 列直交行列 ($Q^T Q = I_n$ を満たす)、 R は $n \times n$ 上三角行列である。縦長行列の QR 分解は、最小二乗問題の求解[2]や特異値分解の前処理[3]に加え、線形方程式や固有値問題に対する様々なアルゴリズムにおけるベクトルの直交化としての応用を持つ[4]。

行列の QR 分解を計算するためのアルゴリズムとしては、線形代数の講義で習う Gram-Schmidt の直交化のアルゴリズム[5]と、一般的な数値計算の教科書に記載されている Householder QR 分解のアルゴリズム[5]が有名である。一方、大規模分散並列環境の普及とともに、線形計算アルゴリズム中の内積計算などで必要となる集団通信 (例: MPI_Allreduce) のコスト (特に、レイテンシに相当するコスト) が強スケーリング時の大きなボトルネックとなり、通信回避 (Communication-Avoiding) や通信隠蔽 (Communication-Hiding) の重要性が増した[6, 7]。そのような背景の下、縦長行列の QR 分解に対して、TSQR アルゴリズムが提案され[8]、その有効性が示された。また、近年、Cholesky QR 分解に基づくアルゴリズム[3]の研究も活発に行われている。

このように、現在では、縦長行列の QR 分解を計算するためのアルゴリズムとして、様々な選択肢が存在する。それぞれのアルゴリズムは、計算精度と計算時間において、異なる特徴を持っている。そこで、本稿では、縦長行列の QR 分解を計算する各種アルゴリズムを概説するとともに、Oakforest-PACS 上で性能評価を行った結果を報告する。なお、本稿で紹介する性能評価結果は、2020 年 1 月 30 日～31 日に実施した「大規模 HPC チャレンジ (Oakforest-PACS)」で得られた結果の一部である。

以下では、分散並列環境における縦長行列の QR 分解を議論するが、計算対象の行列 A は、

$$A = \begin{pmatrix} A_1 \\ \vdots \\ A_p \\ \vdots \\ A_p \end{pmatrix}$$

のように、行方向に 1 次元ブロック分割され、 p 番目のプロセス (プロセス数は P) が A_p を保持する状況を想定する。また、計算結果の Q についても、同様の分散を想定する。一方、 R については、少なくとも一つのプロセスが R 全体を保持している (全てのプロセスが保持する必要はない) とする。なお、アルゴリズムの演算コストや通信コストの議論では、クリティカルパス上のコストをカウントするものとする。また、2 ノルムによる行列の条件数を

$$\kappa_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$$

と定義する[9]。ただし、 $\sigma_{\max}(A)$ と $\sigma_{\min}(A)$ はそれぞれ A の最大特異値と最小特異値である。 A の列ベクトルの線形従属性が強くなるほど、 $\kappa_2(A)$ は大きくなり、一般的に数値的な QR 分解が難しくなる（数値計算の精度が低下したり、アルゴリズムの破綻が生じたりする可能性が高まる）。なお、以下の議論や数値実験では、倍精度浮動小数点（FP64）を用いる場合を想定しており、 A が列フルランクであることを仮定しているため、 $\kappa_2(A) \leq 10^{16}$ 程度となる。

2. 様々な QR 分解アルゴリズム

様々な QR 分解アルゴリズムが存在するが、それらは、Orthogonal Triangularization 型と Triangular Orthogonalization 型に大別される[1]。Orthogonal Triangularization 型のアルゴリズムでは、

$$\hat{Q}_k \cdots \hat{Q}_2 \hat{Q}_1 A \rightarrow \begin{pmatrix} R \\ 0 \end{pmatrix}$$

と直交変換 $\hat{Q}_1, \dots, \hat{Q}_k \in \mathbb{R}^{m \times m}$ を A に（左側から）作用させて、 A を上三角行列に変形する。さらに、

$$(\hat{Q}_k \cdots \hat{Q}_2 \hat{Q}_1)^{-1} \begin{pmatrix} I_n \\ 0 \end{pmatrix} \rightarrow Q$$

とすることで Q を得る。Orthogonal Triangularization 型アルゴリズムの代表例は、Householder 変換を用いたアルゴリズム（Householder QR）やGivens 回転を用いたアルゴリズムである。

一方、Triangular Orthogonalization 型のアルゴリズムでは、

$$A \hat{R}_1 \hat{R}_2 \cdots \hat{R}_k \rightarrow Q$$

と上三角行列 $\hat{R}_1, \hat{R}_2, \dots, \hat{R}_k$ を A に（右側から）作用させて、 A を列直交行列に変形する。さらに、

$$(\hat{R}_1 \hat{R}_2 \cdots \hat{R}_k)^{-1} \rightarrow R$$

とすることで R を得る。Triangular Orthogonalization 型アルゴリズムの代表例は、Gram-Schmidt の直交化によるアルゴリズムである。

以下では、いくつかの主要なアルゴリズムを概説する。ただし、それぞれの特徴を中心に説明するため、アルゴリズムの具体的な中身等については、示している参考文献等を参照されたい。

2. 1. Householder QR 分解アルゴリズム

Householder QR 分解アルゴリズムは、直交変換として Householder 変換[5]

$$H = I - tuu^T, \quad t = \frac{2}{u^T u}$$

を用いた Orthogonal Triangularization 型アルゴリズムである。直交変換に基づく数値計算アルゴリズムは一般的に安定であることが多く、Householder QR アルゴリズムは、 $\kappa_2(A)$ に関わらず、計算精度が良いことが知られている。また、一般的な数値計算の教科書のアルゴリズムを素直に実装すると、行列ベクトル積相当の Level-2 BLAS による計算となるため、LAPACK 等では、Householder 変換のブロック化（Compact WY representation）[10]を用いた、行列積（Level-3 BLAS）中心のアルゴリズムが実装されている。基本的に、行列の列方向に計算の逐次性があり、各列に対応する計算で、列ベクトルに関する内積（や部分的な行列ベクトル積）が必要となり、分散並列計算の場合、 $O(n)$ 回の集団通信が必要となる。そのため、大規模な分散並列計算におけ

る強スケーリングには限界がある。

2. 2. TSQR アルゴリズム

TSQR アルゴリズムは、Demmel らにより提案された[8], Orthogonal Triangularization かつ通信回避型のアルゴリズムである。アルゴリズムの主たるアイディアは、

$$\hat{Q}^{(3)} \begin{pmatrix} \hat{Q}_1^{(2)} & & \\ & \hat{Q}_2^{(2)} & \\ & & \hat{Q}_3^{(2)} \end{pmatrix} \begin{pmatrix} \hat{Q}_1^{(1)} & & & \\ & \hat{Q}_2^{(1)} & & \\ & & \hat{Q}_3^{(1)} & \\ & & & \hat{Q}_4^{(1)} \end{pmatrix} \begin{pmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{pmatrix} \rightarrow R$$

のように、ブロック対角構造を持つ直交行列を段階的に作用させる点である。このような手順で計算する場合、最初の段階の

$$\hat{Q}_i^{(1)} A_i \rightarrow \begin{pmatrix} R_i^{(1)} \\ 0 \end{pmatrix}$$

は、お互いに一切の通信を必要とせず、並列に実行可能となる。その後、隣同士で得られた三角行列を（一対一通信により）送受信し、次の段階の

$$\hat{Q}_i^{(2)} \begin{pmatrix} R_i^{(1)} \\ 0 \\ R_{i+1}^{(1)} \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} R_i^{(2)} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

を計算する。ここでも、最初の段階と同様に、通信を必要とせず、並列に実行可能である。また、実際の計算では、上三角行列 2 個分に相当する $2n \times n$ の密行列の QR 分解相当の演算のみで実装可能であり、さらに、三角行列の構造を配慮した実装も可能である。以下、三角行列の送受信と、三角行列の縮約計算を繰り返すことで、最終的に、元々の計算対象の行列の QR 分解の R を得ることができる。また、直交行列（の転置）を逆の順番で作用させることで、 Q が得られる。なお、各段階で用いる直交行列は、一般的に Householder 変換である。したがって、TSQR アルゴリズムは、局所的な Householder QR 分解を繰り返している、と解釈することが可能である。

TSQR アルゴリズムも、直交変換を利用したアルゴリズムであるため、数値安定性に優れ、Householder QR アルゴリズムと同様に、 $\kappa_2(A)$ に関わらず、計算精度が良いことが知られている（より詳細には、Householder QR と異なる特徴を持つことも示されている）[11]。一方、分散並列計算において必要となる通信は、各段階の間の上三角行列の送受信であり、合計で $O(\log_2 P)$ 回程度の一対一通信となる（簡単のため、二分木に沿った通信を想定する）。より直感的には、TSQR アルゴリズムにおける通信は、三角行列を 2 個縦に並べた行列の QR 分解を縮約演算とする、三角行列に対する $O(1)$ 回の集団通信と解釈できる。これは、集団通信の回数において、Householder QR 分解アルゴリズムの $O(n)$ よりも少なく（通信されるデータ量では両者は同程度）、TSQR アルゴリズムが通信回避型アルゴリズムである所以である。

2. 3. Gram-Schmidt の直交化に基づくアルゴリズム

Gram-Schmidt の直交化は、非常に長い歴史を持つ（1907 年出版の論文で提唱）数値計算アルゴリズムの一つである[12]。アルゴリズムの基本的な構造は、 A の列ベクトルに対するスケーリング（規格化）と、計算済み（直交化済み）のベクトルとの直交化の繰り返しである。ある列ベ

クトルに対するスケーリングは

$$A \begin{pmatrix} I_{i-1} & & \\ & s_i & \\ & & I_{n-i} \end{pmatrix}$$

と書くことができ、一方、直交化（直交補空間への射影）は

$$A \begin{pmatrix} I_{i-1} & \mathbf{r}_i & \\ & 1 & \\ & & I_{n-i} \end{pmatrix}$$

と書くことができる。そのため、Gram-Schmidt の直交化に基づく QR 分解アルゴリズムは、Triangular Orthogonalization 型と解釈できる。

具体的な数値計算アルゴリズムとしては、Classical Gram-Schmidt (CGS) と modified Gram-Schmidt (MGS) が有名である。両者は、数学的には同じであるが、演算順序が異なるため、数値計算的にはアルゴリズムの挙動が異なる。CGS も MGS も $\kappa_2(A)$ が大きくなるにつれて、計算結果の精度 (Q の直交性) が悪化することが知られており、CGS の方が MGS に比べて、精度低下の度合いが大きい。この問題を解決する手段として、再直交化 (Reorthogonalization) があり、再直交化を付与した CGS (通称: CGS2) などがある。なお、再直交化は 1 回 (全体で直交化は 2 回) で十分であることが示されており、「twice is enough」という有名なフレーズがある [13]。

CGS や MGS のアルゴリズムも、教科書に記載の手順を素直に実装すると、Level-2 BLAS による計算となる。ただし、CGS (や CGS2) については、ブロック化により、Level-3 BLAS 中心のアルゴリズムが構築可能である (MGS は演算順序の制約から不可能) [14]。また、分散並列計算では、Householder QR 分解と同様に、基本構造が列ごとの逐次的な処理であり、それに付随する列ベクトルの内積相当の計算のために、 $O(n)$ 回の集団通信が必要となる。そのため、強スケーリングには不向きとなる。

2. 4. Cholesky QR 分解に基づくアルゴリズム

計算対象の行列 A に対して、

$$A^T A \rightarrow W, \quad W \rightarrow R^T R, \quad AR^{-1} \rightarrow Q$$

とすることで、 A の QR 分解を (数学的には) 得ることが可能である [3]。これを Cholesky QR 分解と呼ぶ。なお、2 番目の計算は W に対するコレスキー分解である。このアルゴリズムは、 A が縦長である場合、計算コストの主要部が、1 番目と 3 番目の計算となり、どちらも Level-3 BLAS で実行可能である。また、分散並列計算で必要となる通信は、グラム行列 W を得るための、集団通信 1 回のみである (TSQR と同様に、通信回避型アルゴリズムとみなせる)。したがって、実装が非常にシンプルである点も含めて、Cholesky QR 分解は、高性能計算に非常に適したアルゴリズムである。なお、上記の構造から、Cholesky QR 分解が Triangular Orthogonalization 型であることは明らかである。

しかし、数値計算の教科書等において、 $A^T A$ のような行列を陽的に構成するアルゴリズムはタブー視されている (例: 正規方程式を陽的に構成して、最小二乗問題を解く)。これは、行列の条件数が二乗になり、計算精度の悪化やアルゴリズムの破綻等につながるためである。Cholesky QR 分解でも同様であり、 $\kappa_2(A)$ に応じて、計算結果の精度 (Q の直交性) が悪化し、実用的でないことが知られていた。

この問題に対して、近年、いくつかの進展があった。以前から、Cholesky QR 分解のアルゴリズムを繰り返すことで、 Q の直交性が改善することが示唆されていた[15]が、実際に、 $\kappa_2(A)$ に関する現実的な条件 ($\kappa_2(A) \leq O(10^8)$) の下で、1 回の再直交化を付与 (CholeskyQR2 アルゴリズム) すれば、十分な精度で QR 分解が計算できる (条件付きではあるが、「twice is enough」である) ことが示され[16]、その有効性が確認された[17]。また、さらに $\kappa_2(A)$ が大きい場合に対して、部分的に高精度 (擬似 4 倍精度) 演算を用いる混合精度型の Cholesky QR 分解アルゴリズム[18]、計算されたグラム行列 W に対するシフトを導入した Shifted Cholesky QR 分解を前処理として用いる Shifted-CholeskyQR3 アルゴリズム[19]、 A に対する部分ピボット選択付き LU 分解を前処理とする、LU-CholeskyQR 型アルゴリズム[20]などが提案されている。これらのアルゴリズムは、主要部が Cholesky QR 分解の構造となっているため、演算と通信の両面において、依然として利点を維持している。

3. 各種アルゴリズムの特徴

前節で紹介した、縦長行列の QR 分解を計算する様々なアルゴリズムの特徴を整理する。具体的には、計算時間に関する項目として、演算量 (#Flops)、通信回数 (#Msgs)、通信データ量 (#Words) をまとめる。なお、通信回数と通信データ量については、集団通信を単位とする ($\log_2 P$ 回の一回対一通信を 1 回の集団通信相当としてカウントする)。また、計算精度に関する項目として、計算された Q の直交性 ($\|Q^T Q - I_n\|_2$) を比較する。

各アルゴリズムについて整理した結果を表 1 に示す。なお、CholQR は (再直交化なしの) Cholesky QR、CholQR2 は CholeskyQR2、S-CholQR3 は Shifted-CholeskyQR3、LU-CholQR は LU-CholeskyQR、LU-CholQR2 は LU-CholeskyQR2 (LU-CholeskyQR に再直交化を付与したもの)、HQR は Householder QR である (以降、同様の略記を用いる)。 ϵ は unit roundoff であり、本稿では、倍精度を仮定しているので、 $\epsilon = 2^{-53} \approx 10^{-16}$ である。ただし、 $\kappa_2(A) \leq \epsilon$ は仮定している。また、 Q の直交性に関して、解析的な結果が示されているケースについては、参考文献を付与している (参考文献の記載がないケースは、経験的に知られた情報である)。

表 1 から分かるように、いずれのアルゴリズムも演算量は $O(mn^2/P)$ であるが、TSQR のみ P に応じて増加する項 (上三角行列の縮約のための演算) があることに注意が必要である。一方、通信回数は、非通信回避型が $O(n)$ 、通信回避型が $O(1)$ である。LU-CholQR と LU-CholQR2 が $O(n)$ なのは、部分ピボット選択付き LU 分解におけるピボット操作のためである。通信データ量については、どのアルゴリズムも $O(n^2)$ である。これは、非通信回避型の場合是一度の通信量が少なく (スカラーもしくはベクトル)、通信回避型の場合是一度の通信が行列であるためである。なお、通信回数が $O(1)$ のアルゴリズムに関して、通信回数 (あるいは通信データ量) の比は概ね

$$\text{CholQR} : \text{CholQR2} : \text{S-CholQR3} : \text{TSQR} = 1 : 2 : 3 : 1$$

となっている。

計算精度 (Q の直交性) については、CGS と CholQR が $\kappa_2^2(A)$ に応じて (かつ、CholQR は途中でアルゴリズムが破綻)、MGS と LU-CholQR が $\kappa_2(A)$ に応じて、それぞれ低下する。一方、それ以外のアルゴリズムは、($\kappa_2(A) \leq \epsilon$ の仮定の下では) 問題ないが、CholQR2 については、 $\kappa_2(A)$ が大きくなると、アルゴリズムが破綻するため、注意が必要である。なお、余談であるが、CGS については、解析的には $O(\epsilon \cdot \kappa_2^{n-1}(A))$ が上界となることが示されている[21]が、経験的には $O(\epsilon \cdot \kappa_2^2(A))$ 程度であることが知られている (MGS は $O(\epsilon \cdot \kappa_2(A))$ が上界であることが示されている)。

表 1：各種 QR 分解アルゴリズムの特徴。

Algorithm	#Flops	#Msgs	#Words	Orthogonality of Q
CGS	$\frac{2mn^2}{P} + O(n^3)$	$O(n)$	$O(n^2)$	$O(\epsilon \cdot \kappa_2^2(A))$ ※ $O(\epsilon \cdot \kappa_2^{n-1}(A))$ [21]
CGS2	$\frac{4mn^2}{P} + O(n^3)$	$O(n)$	$O(n^2)$	$O(\epsilon)$
MGS	$\frac{2mn^2}{P} + O(n^3)$	$O(n)$	$O(n^2)$	$O(\epsilon \cdot \kappa_2(A))$ [22]
CholQR	$\frac{2mn^2}{P} + O(n^3)$	$O(1)$	$O(n^2)$	$O(\epsilon \cdot \kappa_2^2(A))$ if $\kappa_2(A) \leq \frac{1}{\sqrt{\epsilon}}$ [16]
CholQR2	$\frac{4mn^2}{P} + O(n^3)$	$O(1)$	$O(n^2)$	$O(\epsilon)$ if $\kappa_2(A) \leq \frac{1}{\sqrt{\epsilon}}$ [16]
S-CholQR3	$\frac{6mn^2}{P} + O(n^3)$	$O(1)$	$O(n^2)$	$O(\epsilon)$ [19]
LU-CholQR	$\frac{3mn^2}{P} + O(n^3)$	$O(n)$	$O(n^2)$	$O(\epsilon \cdot \kappa_2(A))$
LU-CholQR2	$\frac{5mn^2}{P} + O(n^3)$	$O(n)$	$O(n^2)$	$O(\epsilon)$ [20]
HQR	$\frac{4mn^2}{P} + O(n^3)$	$O(n)$	$O(n^2)$	$O(\epsilon)$ [9]
TSQR	$\frac{4mn^2}{P} + O(n^3 \log_2 P)$	$O(1)$	$O(n^2)$	$O(\epsilon)$ [11]

4. Oakforest-PACS 上での性能評価

Oakforest-PACS を用いて、各種 QR 分解アルゴリズムの性能を評価した結果を報告する。評価環境は表 2 の通りである。プログラムは Fortran90 で、BLAS や LAPACK の関数を必要に応じて利用する形で実装した。なお、ScaLAPACK のルーチンは使用しておらず、MPI を用いて、独自に分散並列化している。LU-CholQR 型アルゴリズムの LU 分解部分については、(本来の LU 分解の意味では必要であるが) 今回のケースでは不要となるデータ通信を削除した実装を用いている。また、S-CholQR3 のシフト量は $\sqrt{m}\|A\|_F^2 \times 10^{-16}$ とした。

表 2：性能評価環境。

システム名	Oakforest-PACS (OFP)
CPU	Intel Xeon Phi 7250 (KNL), 1.40 GHz, 68 cores
メモリモード	Flat (MCDRAM のみ使用)
インターコネクト	Intel Omni-Path
コンパイラ&オプション	Intel ifort (ver. 18.0.1) -O3 -align array64byte -qopenmp-ipo-xMIC-AVX512
数値計算ライブラリ	Intel MKL (ver. 2018.0.1) by -mkl=parallel

4. 1. 計算精度に関する評価

まず、各アルゴリズムの計算精度に関する評価結果を示す。テスト行列は、意図した条件数に對して、

$$\Sigma := \text{diag}\left(1, \sigma^{\frac{1}{n-1}}, \dots, \sigma^{\frac{n-2}{n-1}}, \sigma\right), \quad \sigma := \frac{1}{\kappa_2(A)}$$

とした上で、

$$A := U\Sigma V^T$$

と生成した。なお、 $U \in R^{m \times n}, V \in R^{n \times n}$ はランダムに生成した直交行列である。

評価結果を表3に示す。概ね、表1の内容と一致していることが確認できる。なお、S-CholQR3が、 $\kappa_2(A) = 10^{15}$ の場合に破綻しているが、Shifted Cholesky QRによる前処理が1回では不十分であったことを意味しており、2回前処理を行うことで回避可能である(Shifted Cholesky QRを2回行った後に、CholeskyQR2を行う)。

表3：計算精度の評価 ($m = 16777216, n = 128, P = 4096$)。

(a) 直交性： $\|Q^T Q - I\|_F / \sqrt{n}$

$\kappa_2(A)$	1.00E+00	1.0E+03	1.0E+06	1.0E+09	1.0E+12	1.0E+15
CGS	2.06E-16	7.24E-13	1.05E-06	1.17E+00	3.39E+00	4.75E+00
CGS2	1.83E-16	1.83E-16	1.69E-16	1.60E-16	1.60E-16	1.57E-16
MGS	2.02E-16	1.01E-14	4.17E-12	2.94E-09	2.29E-06	2.34E-03
CholQR	1.67E-16	2.73E-12	1.69E-06	破綻	破綻	破綻
CholQR2	1.71E-16	2.06E-16	2.09E-16	破綻	破綻	破綻
S-CholQR3	1.63E-16	2.19E-16	2.00E-16	2.21E-16	2.08E-16	破綻
LU-CholQR	2.69E-15	2.75E-14	1.37E-11	1.06E-08	5.89E-06	6.12E-03
LU-CholQR2	1.76E-16	2.14E-16	1.91E-16	1.92E-16	2.44E-16	2.52E-16
HQR	2.92E-16	2.77E-16	2.72E-16	2.75E-16	2.71E-16	2.74E-16
TSQR	3.00E-16	2.69E-16	2.76E-16	2.54E-16	2.88E-16	2.82E-16

(b) 残差： $\|A - QR\|_F / \|A\|_F$

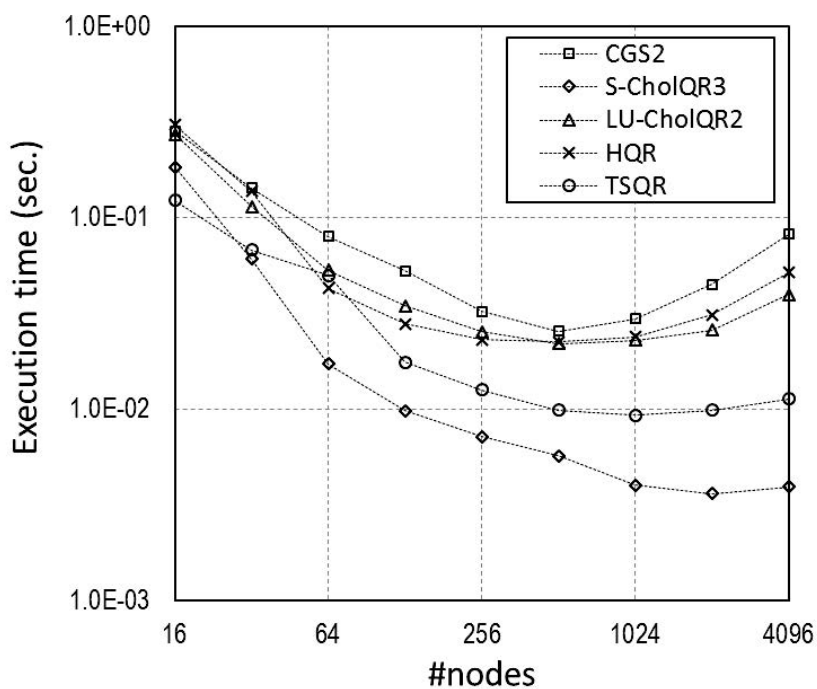
$\kappa_2(A)$	1.00E+00	1.0E+03	1.0E+06	1.0E+09	1.0E+12	1.0E+15
CGS	1.84E-16	3.75E-16	3.77E-16	3.79E-16	3.81E-16	3.81E-16
CGS2	2.33E-16	3.82E-16	3.82E-16	3.83E-16	3.84E-16	3.85E-16
MGS	1.84E-16	3.75E-16	3.77E-16	3.79E-16	3.81E-16	3.81E-16
CholQR	1.08E-16	1.74E-16	1.73E-16	破綻	破綻	破綻
CholQR2	1.23E-16	4.87E-16	5.32E-16	破綻	破綻	破綻
S-CholQR3	1.46E-16	5.31E-16	5.57E-16	5.81E-16	5.91E-16	破綻
LU-CholQR	1.59E-16	1.74E-16	1.72E-16	1.72E-16	1.72E-16	1.72E-16
LU-CholQR2	1.64E-16	4.59E-16	4.93E-16	5.24E-16	5.32E-16	5.52E-16
HQR	2.30E-16	4.88E-16	5.27E-16	5.14E-16	4.90E-16	6.45E-16
TSQR	4.05E-16	5.12E-16	5.12E-16	4.96E-16	4.66E-16	6.25E-16

4. 2. 計算時間に関する評価

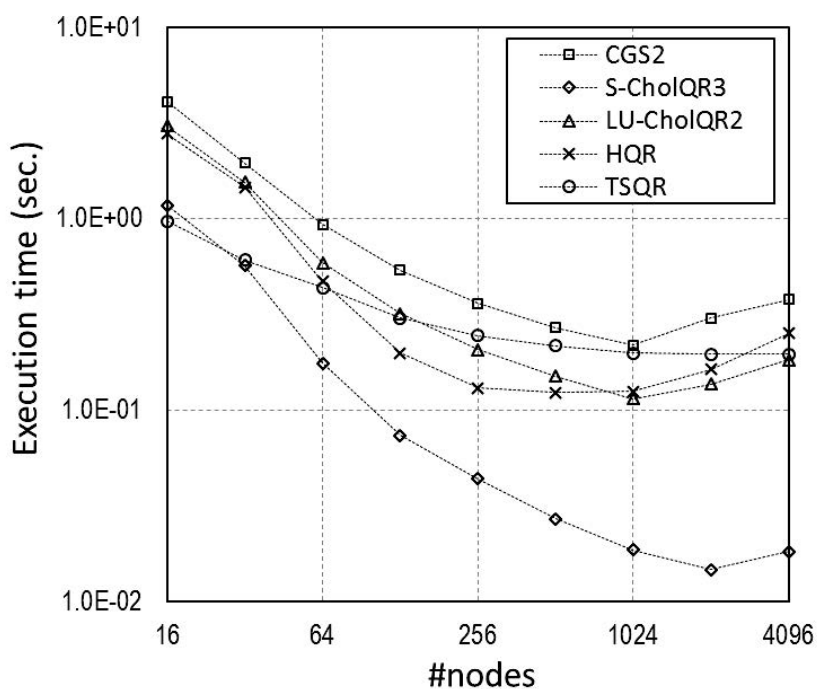
次に、各アルゴリズムの計算時間を評価した結果を報告する。以下では、 $\kappa_2(A)$ がそれなりに大きい場合でも精度良く計算可能な、CGS2, S-CholQR3, LU-CholQR2, HQR, TSQR を比較する。計算ノード数を 16 から 4096 まで変化（強スケーリング）させて、計算時間を測定した。なお、ノード上のプロセス数とスレッド数については、1 プロセス×64 スレッド、4 プロセス×16 スレッド、16 プロセス×4 スレッド、64 プロセス×1 スレッド、の 4 通りを試し（各設定で 5 回計測）、各アルゴリズム（の各ノード数）で最も計算時間が短かった場合を評価対象としている。

$n = 64, 256$ の場合の結果を図 1 に示す。図 1(a), (b) から分かるように、どちらの場合でも、S-CholQR3 が最も高速で、良好な強スケーリングを示している。4096 ノードを使用した場合において、 $n = 64$ では、2 番目に高速である TSQR の 2.9 倍、 $n = 256$ では、LU-CholQR2 の 10 倍、S-CholQR3 は高速であった。また、どちらの場合においても、16 ノードの場合に対して、4096 ノードにおいて、47 倍の高速化となっている。

次に、4096 ノードの場合における、各アルゴリズムの計算時間の内訳を図 2 に示す。なお、「Reduction Comput.」は TSQR における三角行列を 2 個並べた行列の QR 分解（縮約処理）の演算時間である。図 2 から分かるように、通信回数が $O(n)$ である CGS2, LU-CholQR2, HQR では、通信時間（MPI Comm.）が大部分を占めており、これが強スケーリングを阻害する原因となっている。一方、TSQR については、通信時間は少ないが、Reduction Comput. が目立っており、特に、 $n = 256$ において、大きなボトルネックとなっている。この部分は、 $O(n^3)$ で増加するため、 n が大きくなると、途端にボトルネックとなってしまう [23, 24]。これらのアルゴリズムに対して、S-CholQR3 は、通信時間と演算時間（Others）の両方が少なく、その結果、良好な性能が得られている。なお、演算時間も目立っていないのは、前処理部分の Shifted Cholesky QR 部分を含めて、Level-3 BLAS で大部分が計算可能であるという、Cholesky QR 元々の利点が維持されているからである。表 3 に関連して、 $\kappa_2(A)$ が非常に大きい場合、前処理（Shifted Cholesky QR）の追加が必要であると述べたが、その場合、計算時間は約 4/3 倍となる。しかし、図 1 の結果を踏まえると、今回の状況では、依然として計算時間の面で優位であることが推測できる。

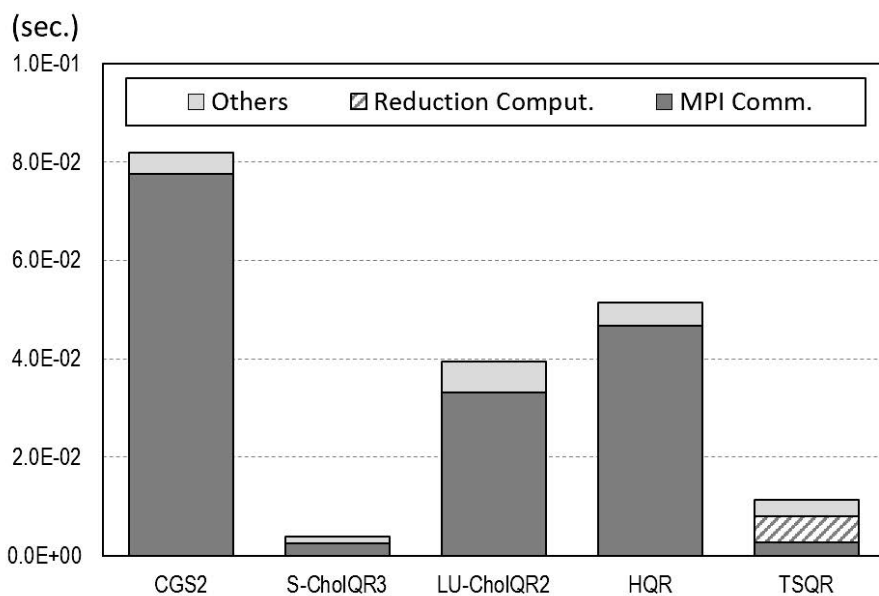


(a) $n = 64$

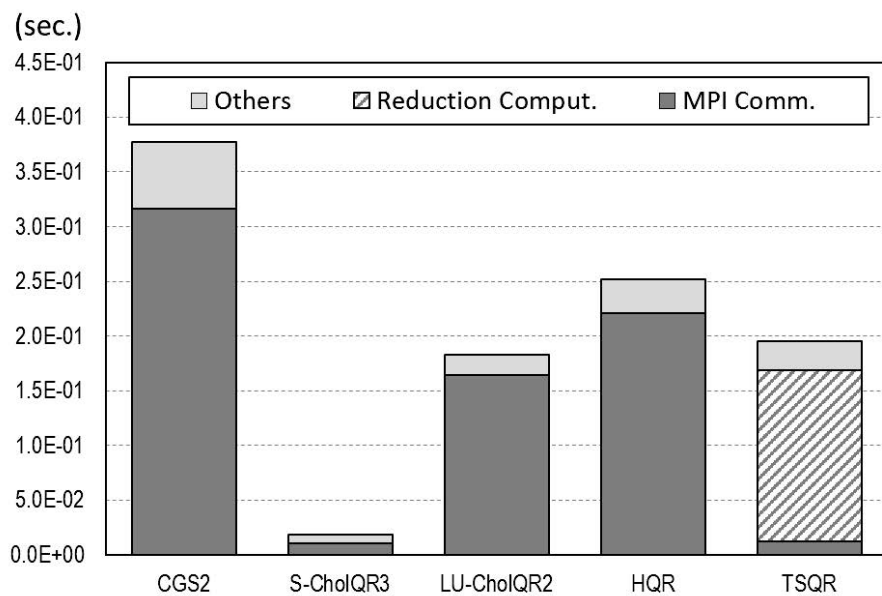


(b) $n = 256$

図 1 : 計算時間の評価結果 (強スケーリング, $m = 16777216$)。



(a) $n = 64$



(b) $n = 256$

図 2 : 計算時間の内訳 ($P = 4096$, $m = 16777216$)。

5. おわりに

本稿では、縦長行列の QR 分解を計算するための各種アルゴリズムを概説し、Oakforest-PACS 上で実施した性能評価結果を示しながら、アルゴリズムを比較した。計算対象の行列が比較的難しい ($\kappa_2(A)$ が大きい) 場合において、精度良く計算が可能なアルゴリズムの計算時間を、最大 4096 ノードまでを用いた強スケーリングで評価した結果、今回の状況では、S-CholQR3 が最も優れていることが確認された。S-CholQR3 は、演算量の面では他のアルゴリズムに劣るが、通信回避型 (集団通信の回数が $O(1)$) であり、演算は行列積等の Level-3 BLAS が中心であるため、最近の高性能計算環境に非常に適している。この点が今回の評価結果につながったと考えられる。

最後に、本稿では紹介できなかったが、関連する研究をいくつか紹介する。まず、アルゴリズムの安定性や計算精度に関しては、最近の文献[25]で詳しい議論がなされている。次に、Cholesky QR 型のアルゴリズムについて、行列が縦長ではない場合を想定した、2.5 次元実装が提案されている[26]。その他にも、同期を削減した QR 分解アルゴリズムとして、タイル QR 分解アルゴリズム[27]が知られている。

謝 辞

大規模 HPC チャレンジの実施に関してお世話になりました、東京大学情報基盤センターの関係者の皆様、及び、当日の不具合対応でお世話になりました、富士通株式会社の関係者の皆様に深く感謝いたします。また、文献[19]をはじめとする、Cholesky QR 分解関連の共同研究においてお世話になっている、山本有作 教授 (電気通信大学)、中務佑治 准教授 (University of Oxford)、柳澤優香 博士 (早稲田大学、文献[19]投稿時)、Dr. Ramaseshan Kannan (Arup, UK) に深く感謝いたします。本研究の一部は、JSPS 科研費 (課題番号: 18K18058) の支援を受けています。

参 考 文 献

- [1] L. N. Trefethen and I. David Bau, Numerical Linear Algebra, SIAM, 1997.
- [2] Å. Björck, Numerical Methods for Least Squares Problems, SIAM, 1996.
- [3] G. H. Golub and C. F. Van Loan, Matrix Computations, 3rd ed., Johns Hopkins University Press, 1996.
- [4] 櫻井鉄也, 松尾宇泰, 片桐孝洋, 数値線形代数の数理と HPC, 共立出版, 2018.
- [5] 杉原正顕, 室田一雄, 線形計算の数理, 岩波書店, 2009.
- [6] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, Minimizing communication in numerical linear algebra, SIAM J. Matrix Anal. Appl., Vol. 32 (2011), pp. 866-901.
- [7] J. Demmel, Communication-Avoiding Algorithms for Linear Algebra, Machine Learning, and Beyond, E-NLA seminar, 2020. <https://www.youtube.com/watch?v=42f0n0w2Nlg>
- [8] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, Communication optimal parallel and sequential QR and LU factorizations, SIAM J. Sci. Comput., Vol. 34 (2012), pp. 206-239.
- [9] N. J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, 2002.
- [10] R. Schreiber and C. Van Loan, A storage-efficient WY representation for products of Householder transformations, SIAM J. Sci. Stat. Comput., Vol. 10 (1989), pp. 53-57.

- [11] D. Mori, Y. Yamamoto, and S.-L. Zhang, Backward error analysis of the Allreduce algorithm for Householder QR decomposition, *Japan J. Indust. Appl. Math.*, Vol.29 (2012), pp.111–130.
- [12] S. J. Leon, Å. Björck, and W. Gander, Gram-Schmidt orthogonalization: 100 years and more, *Numer. Linear Algebra Appl.*, Vol.20 (2013), pp.492–532.
- [13] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, 1980.
- [14] G. W. Stewart, Block Gram-Schmidt orthogonalization, *SIAM J. Sci. Comput.*, Vol.31 (2008), pp.761–775.
- [15] A. Stathopoulos and K. Wu, A block orthogonalization procedure with constant synchronization requirements, *SIAM J. Sci. Comput.*, Vol.23 (2002), pp.2165–2182.
- [16] Y. Yamamoto, Y. Nakatsukasa, Y. Yanagisawa, and T. Fukaya, Roundoff error analysis of the CholeskyQR2 algorithm, *Electron. Trans. Numer. Anal.*, Vol.44 (2015), pp.306–326.
- [17] T. Fukaya, Y. Nakatsukasa, Y. Yanagisawa, and Y. Yamamoto, CholeskyQR2; A simple and communication-avoiding algorithm for computing a tall-skinny QR factorization on a large-scale parallel system, *Proc. Scala’ 14*, pp.31–38, 2014.
- [18] I. Yamazaki, S. Tomov, and J. Dongarra, Mixed-precision Cholesky QR factorization and its case studies on multicore CPU with multiple GPUs, *SIAM J. Sci. Comput.*, Vol.37 (2015), pp.C307–C330.
- [19] T. Fukaya, R. Kannan, Y. Nakatsukasa, Y. Yamamoto, and Y. Yanagisawa, Shifted Cholesky QR for computing the QR factorization of ill-conditioned matrices, *SIAM J. Sci. Comput.*, Vol.42 (2020), pp.A477–A503.
- [20] T. Terao, K. Ozaki, and T. Ogita, LU-Cholesky QR algorithms for thin QR decomposition, *Parallel Comput.*, Vol.92 (2020), pp.102571.
- [21] A. Kiełbasiński, Analiza numeryczna algorytmu ortogonalizacji Grama-Schmidta, *Ser. III Mat. Stosow. II*, (1974), pp.15–35 (in Polish).
- [22] Å. Björck, Solving linear least squares problems by Gram-Schmidt orthogonalization, *BIT*, Col.7 (1967), pp.1–21.
- [23] T. Fukaya, T. Imamura, and Y. Yamamoto, Performance analysis of the Householder-Type Parallel tall-skinny QR factorizations toward automatic algorithm selection, *LNCS*, Vol.8969 (2015), pp.269–283.
- [24] T. Fukaya, An investigation into the impact of the structured QR kernel on the overall performance of the TSQR algorithm, *Proc. HPC Asia 2019*, pp.81–90, 2019.
- [25] E. Carson, K. Lund, M. Rozložník, and S. Thomas, An overview of block Gram-Schmidt methods and their stability properties, *arXiv:2010.12058v1*, 2020.
- [26] E. Hutter and E. Solomonik, Communication-avoiding Cholesky-QR2 for rectangular matrices, *Proc. IPDPS 2019*, pp.89–100, 2019.
- [27] B. Hadri, H. Ltaief, E. Agullo, J. Dongarra, Tile QR factorization with parallel panel processing for multicore architectures, *Proc. IPDPS 2010*, pp.1–10, 2010.

第 136 回お試しアカウント付き並列プログラミング講習会

「OpenFOAM 入門」実施報告

今野 雅

東京大学情報基盤センター客員研究員

2020 年 7 月 21 日 (火), PC クラスタコンソーシアム (実用アプリケーション部会・HPC オープンソースソフトウェア普及部会)、オープン CAE 学会との共催で、第 136 回お試しアカウント付き並列プログラミング講習会「OpenFOAM 入門」がオンラインで開催された。本講習会は、センターに設置されたスーパーコンピュータ(以降、スパコン)の利用促進とスパコンを用いた数値流体解析の普及を目的として実施されたものである。なお、本講習会はセンターのお試しアカウント付き並列プログラミング講習会として行われた 16 回目の OpenFOAM の講習会である。受講者は、学部学生 3 名、大学院学生(修士)5 名、大学院学生(博士)2 名、助教 2 名、准教授 2 名、教授 1 名、技術職員 1 名、研究機関研究員 2 名、企業の方 7 名であり、事前申込者 30 名、受講者合計は 25 名であった。センターが運営するスパコン Oakbridge-CX を用い、Oakbridge-CX の概要、利用方法、OpenFOAM の演習が 1 日終日の日程で行われた。当日のプログラムを表 1 に掲載する。なお、講習会終了後約 1 ヶ月有効なお試しアカウント(Oakbridge-CX, 最大ノード数 8, 最大実行時間 15 分)が受講者に与えられた。

表 1 講習会プログラム

【2020 年 7 月 21 日 (火)】

- 10:00 - 10:30 Oakbridge-CX 概要
 - 10:30 - 11:30 OpenFOAM 概要
 - 12:30 - 14:00 キャビティ流れ演習 I
 - ・ blockMesh による格子生成
 - ・ ParaView による格子可視化
 - ・ icoFoam による流れ解析
 - 14:15 - 15:45 キャビティ流れ演習 II
 - ・ 解析結果可視化
 - ・ 解析結果サンプリング
 - ・ 解析結果プロット
 - 16:00 - 18:00 キャビティ流れ演習 III
 - ・ 並列計算
 - ・ snappyHexMesh による格子生成基礎
 - ・ 質疑
-

講習会終了後のアンケート集計結果(回答数 20)を表 2 に示すが、参加した満足度の平均は 5 点満点中、4.25 と高かった。また、参加者から表 3 に示すご意見を頂いた。今後の講習会の参考にしたい。

表 2 アンケート集計結果

評 点	講習会の時間		講習会の講義内容 (プレゼン)		配布資料の内容		サンプルプログラム 内容		参加した満足度	
1	短い	0	簡単	0	簡単	1	簡単	1	不満	0
2		3		4		2		2		0
3	適切	13	適切	10	適切	13	適切	13	普通	5
4		4		5		3		3		5
5	長い	0	難	1	難	1	難	1	満足	10
	平均	3.05	平均	3.15	平均	3.05	平均	3.05	平均	4.25

表 3 講習会に対するご意見(原文ママ)

- 卒業研究に活かせたらと思い、受講しました。スパコン未経験の私には大変難しい講習でしたが、質問にも丁寧に対応して下さいありがたかったです。もうすこし勉強してまたこちらの講習会に参加したいと思います。長時間ありがとうございました。
- 実際に利用を考えた場合、実行速度、ソルバーのカスタマイズ環境や、計算時間、メモリ使用量の予測など必要なことが多いと思います。そのあたりの世間一般のコンピューター環境との比較データもあると良かった。
- 講習会の開催、ありがとうございました。OpenFOAM 学会様が行っていますが、OpenFOAM の操作等に重きを置いた講習会を開催いただければ、初心者としては助かります。
- OpenFOAM の使い方のイメージがつかしました。また独学で linux を弄っていたので、コマンドの由来や alias の便利な使い方など、細かいテクニックも学ぶことができ、受講して良かったです。ありがとうございました。
- とてもわかりやすい OpenFOAM の講義だったと思います。ありがとうございました。ZOOM 用の端末と操作用の端末を分けたのですが、操作用の端末にもテキスト標示用のサブディスプレイがあったほうがよかったと思いました。エディターの操作を練習しておくべきでした。

第 137 回お試しアカウント付き並列プログラミング講習会

「第 2 回 GPU ミニキャンプ～GPU 化にチャレンジする会～」

下川辺 隆史

東京大学情報基盤センター

2020 年 8 月 3 日（月）、4 日（火）の 2 日間にわたり、Zoom と Slack を用いてオンラインにて、第 137 回お試しアカウント付き並列プログラミング講習会「第 2 回 GPU ミニキャンプ～GPU 化にチャレンジする会～」が開催されました。

本講習会では、これから GPU を利用される方またはすでに利用されているが効率化を進めたい方、スパコンの GPU を利用したい方、を対象に、情報基盤センターに設置されたスーパーコンピュータ Reedbush (Reedbush-H) を活用した実践を行いました。ミニキャンプは参加者がコードやデータセットを持ち込み、各自のペースで GPU 化や GPU 利用効率向上などを実践する形で進みます。情報基盤センター教員に加えて、GPU のスペシャリストがメンターとして参加し、受講者はコードの GPU 化や利用率向上の作業を進めるにあたり随時相談することができます。

本講習会は、東京大学情報基盤センター、エヌビディア合同会社、PC クラスタコンソーシアム（実用アプリケーション部会）の共催で開催され、下記の皆様にメンターとしてご協力いただきました（敬称略）。

- 成瀬 彰：エヌビディア合同会社 シニアデベロッパーテクノロジーエンジニア
- 村上真奈：エヌビディア合同会社 シニアソリューションアーキテクト
- 丹 愛彦：エヌビディア合同会社 HPC ソリューションアーキテクト
- 阮 佩穎：エヌビディア合同会社 ディープラーニングソリューションアーキテクト
- 大友広幸：東京工業大学 横田理央研究室 博士課程 1 年

本講習会のスケジュールは表 1 の通りです。本講習会では、OpenACC、CUDA、深層学習などに取り組む受講者が集まり、オンラインでメンターと相談をしながら各自の作業を進めていきました。図 1 は講習会最後に撮影した集合写真です。受講者には実習で使用した Reedbush-H を受講後 1 ヶ月間利用できるお試しアカウントが与えられます。

表1 スケジュール

日付	時間	内容
8月3日	10:00 - 10:20	Reedbush-Hの使い方講座
	10:20 - 10:50	自己紹介と目標設定など
	10:50 - 13:00	実践（適宜自由に休憩）
	13:00 - 13:30	お役立ち情報
	13:30 - 16:50	実施（適宜自由に休憩）
	16:50 - 17:00	事務連絡・終了
8月4日	10:00 - 10:20	昨日の結果と本日の目標設定
	10:20 - 16:00	実施（適宜自由に休憩）
	16:00 - 16:50	実践（適宜自由に休憩）
	16:50 - 17:00	アンケート記入・終了

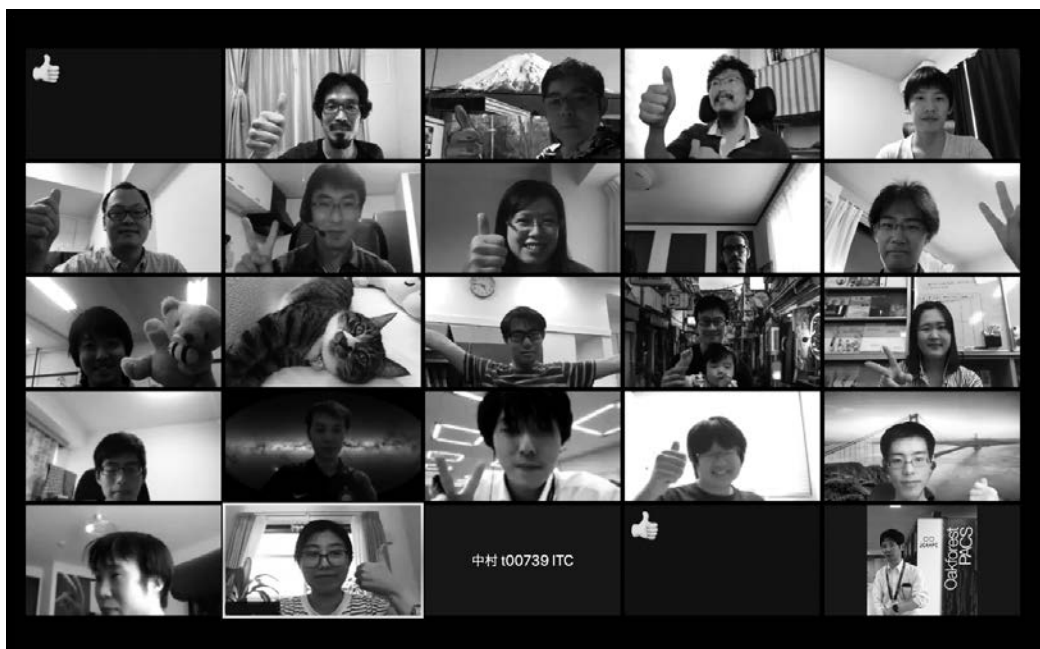


図1 講習会の集合写真

今回の講習会では、合計 19 名の事前申込者があり、そのうち 13 名が受講しました。受講者の内訳は、大学院（修士）学生：4 名、大学院（博士）学生：1 名、助教：3 名、研究機関研究員：2 名、企業の方：3 名でした。講習会終了後に実施したアンケートの質問項目と回答の人数分布は表 2 の通りです。

オンライン開催について、下記の自由回答をいただきました。

オンラインの方が良いと回答した参加者の理由：

- 対面よりも気軽に質問をすること、また一つの質問に多くのメンターの方から意見をいただくことができました。
- わざわざ現地に行かなくても十分質問できるため。
- 研究室から参加できるのが魅力です。メンターの方が slack で非常に丁寧に対応してくれたことも大きいと思います。敢えてデメリットを挙げるとするならば疲れてきたときに少しだれてしまったことくらいです。

現地開催の方が好ましいと回答した参加者の理由：

- zoom では回線が不安定になる等、教えてもらう時に不便があったため。
- メンターの方々に code を直接、見ていただきながらコメントをいただくと作業効率がよさそうな気がします。
- 対面に比べて質問をしにくい。対話に比べてレスポンスが悪化する。

下記の自由回答をいただきました。

- Reedbush の queue 使用開始がミニキャンプ初日でしたが、もう少し前から使えるとトライできてよいかなと思いました。
- 初歩的な質問にも丁寧にメンターの方々に回答していただくことができて大変勉強になりました。二日間ありがとうございました。
- もう少し OpenACC を使い込んでから、アドバンスドコースを受講したいと思いますので、今後、そのような講習会の企画をお願いします。いろいろとありがとうございました。
- 初心者にも分かりやすく色々教えていただき、メンターの方々には大変お世話になりました。技術的な面では非常に満足するやり取りができたと思います。一方、メンターや講習会参加者間での交流ができればさらに良かったと思いますが、オンライン開催だと中々難しいですね。いずれ、現地開催の交流会にも参加してみたいと思っています。いつになるか分かりませんが...
- 初めてスパコンを利用しましたが、親切にご指導いただき助かりました。
- お世辞ではなく、めちゃくちゃ勉強になりました、ネットの文言だけではわからないことを色々教えてもらえて非常に役立ちました。ありがとうございました。
- 2 日間はやはり短く、進めて行くうちに課題を発見すると思います。とはいえ、長時間参加し続けることはできません。同様の講習会を数回して頂けると嬉しいです。毎回参加という訳には行きませんが、可能な時には参加したいと思います。
- 大変勉強になりました。筑波大学内にあるスパコンを使ってみないかというお話がマからあったのですが、前向きに検討しようと思いました。

アンケート集計結果を見ると、多くの方に満足してもらえたようで、改善点を踏まえて、次回以降の開催を検討したいと考えております。今回は、新型コロナウイルス感染症対策のため、オンライン開催となりましたが、オンライン開催のメリットも確認できましたので、今後は状況をみながら、オンライン開催と現地開催の両方のメリットを生かした開催形態も考えていきたいと思ひます。

表2 アンケート集計結果の人数分布と平均

	あり				なし			
GPU プログラミング経験	5				7			
	オンラインが良い		現地開催が良い		どちらでも良い			
オンラインと現地開催	4		3		5			
	評点	1	2	3	4	5	平均	
講習会時間	短い <-> 長い	5	2	4	1		2.1	
講習会講義内容（プレゼン）	簡単 <-> 難	1	3	6	2		2.8	
配布資料内容	簡単 <-> 難	1	2	7	2		2.8	
満足度	不満 <-> 満足			1	3	8	4.6	

第 138 回お試しアカウント付き並列プログラミング講習会

「スーパーコンピューター超入門」

芝 隼人 河合 直聡

東京大学情報基盤センター

2020 年 9 月 18 日に、第 138 回お試しアカウント付き並列プログラミング講習会「スーパーコンピューター超入門」（共催：東京大学情報基盤センター、PC クラスタコンソーシアム（実用アプリケーション部会・HPC オープンソースソフトウェア普及部会））を開催した。本稿はその実施報告である。

この講習会はタイトルの通り「スーパーコンピューターとは何か、あまり知らない」「スーパーコンピューターの最低限の操作方法を知りたい」といった希望を持つ受講者を対象として開催したものであり、これまでに開催されてきた本センター主催の同シリーズの講習会とは趣を異にしている。このような講習会を開くことになった背景には、部門教員の間でミーティングをしている際に「学生および企業利用ユーザーなどを含め、最初のスパコン利用のためのとっかかりとなるような講習会が必要ではないか」といった議論があったことがある。スーパーコンピューターを初めて使用する前に最初に勉強する必要があることとして、Linux の簡単な操作・ログインの仕方・ジョブ投入の仕方、といったようなことが想定される。これらの内容を取り扱うべく、当初 2020 年 4 月の開催を目標に準備を始めようとしたところ、周知の COVID-19 の感染拡大となり一度開催を見合わせることにした。そこで Zoom を用いて、4 月以降に実施された他の一連のオンライン講習会の様子を観察し、この講習会の実施可能性について見極めを行った上で冒頭に述べた日程で実施することとした。8 月 18 日にアナウンスを実施したが、8 月中に申込数が定員 30 名に到達し申込受付を停止する盛況ぶりであった。

もとより、担当講師は 2 名とも当センターへの着任し業務にあたるようになったのが今年に入ってからである。そういうわけで受講者のみならず担当講師も「超入門」で担当している異例の講習会となった。

本講習会の構成は、次の通りの 3 部からなっている。

13:00 – 14:00 第 1 部 講義

14:15 – 16:15 第 2 部 Oakbridge-CX システムを用いた実習

16:30 – 17:00 第 3 部 センター提供のスーパーコンピューター紹介、利用申込の手引き

第 1 部ではスーパーコンピューターについて、その特徴や用途、そして基本的なコマンド操作やプログラム作成、Environment Module の利用方法、コンパイル、ジョブ投入など、基本的な事項について 1 時間かけて座学で紹介した。第 2 部では Oakbridge-CX システムを利用して 2 時間ほどにわたる実習を行った。公開鍵認証によるログイン、サンプル並列プログラムのコンパイル、そしてジョブ投入を行っていただき、並列化によって簡単な積分計算の実行時間を短縮できることを体感してもらう内容とした。そして、最後に第 3 部として、当センターにおけるスーパーコンピューター利用申込や申請にあたっての注意点を解説した。

第 1 部の講義では、オンライン講義という形態そのものについて巷間議論されていることでも

あるが、受講者の表情・反応が見えないため、理解度、進行度が妥当かどうかを判断するのが難しく感じた。講義の内容には Zoom や Slack の使用・質問方法も含めており、講師側が考える質問をしやすい環境を整えていたが、受講者からの反応はなかった。今後講義資料をより一層洗練させるとともに、講義中に質問しやすい環境を整えることにも注力していく。

第2部は2時間の、全般的にスケジュール通りに進んだと感じている。実習の際、受講者に所定の作業を端末上で実施してもらって、完了後には Zoom において最近利用可能になった “non-verbal feedback” の機能を利用して、作業完了・未完了の意思表示をしてもらったところ、80% 以上の方がすぐに反応してくださり、円滑な進行ができた。受講者のレベルはもとよりばらつきがあったため、相当の暇を持て余した参加者もいたと思われる。

実習の際のサンプルプログラムについては、C, Fortran, Python の3パターンを用意した。Python を取り上げたのは今回の講習会においてとりわけ特徴的なことであり、当センターの主催する講習会においても初めてではないかと考えている。Python の並列化は mpi4py のモジュールを利用して行うこととしたが、Python のパッケージ管理システムにおけるインストール先を変更するための環境設定についてサンプルとして受講者に配布したファイルに誤りがあり、Python を選択された受講者は受講時間中に作業をしていただくことができないこととなった。担当講師の側で確認が不足していたために発生したことであり、該当の受講者にはこの場を借りて改めてお詫びを申し上げる次第である。そんな中にもあっても、受講者の方々から Python を利用した講習に対して好評をいただいたことを、たいへん喜ばしく思っているところである。また資料にも誤植などが多く混乱を招いたことも申し訳なく、反省を次回以降に活かすこととしたい。

第3部については、内容をもう少し担当講師の側で吟味しておけば良かったと思っている。というのは、担当講師が想定していた参加者層はどちらかといえば、学位取得後の若手あるいは企業研究者だったのであるが、講義前日になってから参加者リストを見たところ大学院生の受講者が半分以上を占めていた。一方、実際に話した内容は、若手・女性利用制度の解説こそ大学院生に一応関係するものの、一般利用のコースの選択基準、利用料金などは大学院生自身にとって極めて重要な情報ではない。むしろ、ユーザー自身が自分のプログラムを動作させる際に、どのくらいの量の資源が必要となるかを見積もる方法、システム選択はどう行うべきか、などの情報を提供した方が有益でなかったかと反省している。今後、同様の講習会の実施の際には、この点をアップデートすることとしたい。

講習会を実施する講師にとっても参加者にとっても、オンライン講習会における講師2名体制での対応は非常に有益であったと思われる。というのも実習時のトラブルに直面した参加者に対する個別対応を行うことが可能だからである。個別対応は、Slack 上でチャットを行うことを基本とし、場合によっては講習会に使用している Zoom 会議室内にブレイクアウトルームに質問者および1名の講師(またはサポートスタッフ)を退避させて個別にやりとりすることで実施した。

講習会後に実施したアンケートには、18名からの回答を戴くことができた。表1は質問項目と回答内容の人数分布である。講習内容上、受講者のスキルのばらつきが避けられなかったわけであるが、その中にもあっても比較的好評をいただけたと思い、うれしく思っている。

オンライン講習会については、良かったこととして「気軽に参加できる」「画像などを後から追って確認でき、間違いが少ない」「ブレイクアウトルームを利用したトラブルシューティングが有効である」などの意見があった。また欠点としては「声が聞き取りづらいときがあった」「回線が悪いときがあった」「周囲の進捗度・理解度がわからない」などの意見を寄せていただいた。

表 1 アンケート集計結果

	評点	1	2	3	4	5
(a) 講習会時間	短い⇔長い		2	10	3	3
(b) 講習会講義内容	簡単⇔難	3	5	10		
(c) 配布資料内容	簡単⇔難	2	4	11	1	
(d) サンプルプログラム内容	簡単⇔難	4	5	8	1	
(e) 参加した満足度	不満⇔満足	1	1	4	4	8

また、アンケートの自由記述欄からの抜粋として、下記のようなコメントがあった。

- スパコンで `python` を利用に興味がある。初心者として必要な情報が何かもわからないため、スパコン向けの `python` 情報のマニュアルを整備していただくことを要望したい。
- `Python` の `JupyterLab` などを書いたスクリプトをスパコンで使用する場合の方法などを講習してほしい。
- `C, FORTRAN` による並列プログラミング解説に加えて `Python` による並列プログラミングの解説は目新しいものでした。
- データ分析のための利用を過去にしていたことに動機づけられ、受講した。
- サンプルプログラムをその場で編集しようとしたため気を取られ、そのあとの説明についていけず躓いた。自分が `CUI` に不慣れなためですが、ここは、用意されたものを初めから使うことで進めれば良いと思います。

これらの回答において、先述した `python` など、スーパーコンピューティングにおいては新しいプログラミング言語、またシミュレーションのみならずデータ分析用途でのスーパーコンピューター利用の需要があると示唆されている。今後、我々としても `python` によるスーパーコンピューター利用に特化した講習、あるいは、`python` を利用した機械学習ツールの利用方法についての講習の実施を検討していくことが必要であると考え次第である。

第 1、2 部については、講義資料および実習資料を当センターの第 138 回講習会ホームページ（<https://www.cc.u-tokyo.ac.jp/events/lectures/138/>）において公開している。また同じ場所からは、当日の第 1 部の講義動画を視聴することができる。

最後に本講習会の実施にあたっては情報基盤課スーパーコンピューティングチームの担当者の方々、とりわけ当日講習会の実施中、ずっとオンライン上で張り付いてアシストいただいた中張遼太郎、山田新の両氏には大変お世話になった。この場を借りてお礼を申し上げる。

第 139 回お試しアカウント付き並列プログラミング講習会

「OpenFOAM 初級」実施報告

今野 雅

東京大学情報基盤センター客員研究員

2020 年 9 月 29 日 (火), PC クラスタコンソーシアム (実用アプリケーション部会・HPC オープンソースソフトウェア普及部会)、オープン CAE 学会との共催で、第 139 回お試しアカウント付き並列プログラミング講習会「OpenFOAM 初級」がオンラインで開催された。本講習会は、センターに設置されたスーパーコンピュータ (以降, スパコン) の利用促進とスパコンを用いた数値流体解析の普及を目的として実施されたものである。なお、本講習会はセンターのお試しアカウント付き並列プログラミング講習会として行われた 17 回目の OpenFOAM の講習会である。受講者は、学部学生 6 名、大学院学生 (修士) 6 名、大学院学生 (博士) 1 名、講師 1 名、教授 1 名、企業の方 5 名であり、事前申込者 22 名、受講者合計は 20 名であった。センターが運営するスパコン Oakbridge-CX を用い、Oakbridge-CX の概要、利用方法、OpenFOAM の演習が 1 日終日の日程で行われた。当日のプログラムを表 1 に掲載する。なお、講習会終了後約 1 ヶ月有効なお試しアカウント (Oakbridge-CX, 最大ノード数 8, 最大実行時間 15 分) が受講者に与えられた。

表 1 講習会プログラム

【2020 年 9 月 29 日 (火)】

- 10 : 00 - 10 : 15 イン트로ダクション
- 10 : 15 - 10 : 45 Oakbridge-CX 概要
- 10 : 45 - 12 : 00 自動車空力解析演習 I
 - ・ Oakbridge-CX へのログイン・講習会の準備
 - ・ 解析モデル
 - ・ 解析手順
 - ・ バッチジョブスクリプト
- 13 : 00 - 15 : 00 自動車空力解析演習 II
 - ・ snappyHexMesh による並列格子生成
 - ・ ParaView による格子可視化
 - ・ 格子生成演習
 - ・ simpleFoam による並列定常乱流流体解析
- 15 : 15 - 16 : 45 自動車空力解析演習 III
 - ・ simpleFoam による並列定常乱流流体解析 (続き)
 - ・ 空力係数のモニター
 - ・ ParaView による解析結果可視化
 - ・ 解析結果のサンプリング
 - ・ 速度分布のプロットと実測値との比較

- ・性能プロファイリング
- ・並列計算演習
- ・空力解析演習

17 : 00 - 18 : 00 自動車空力解析演習 IV

- ・質疑
- ・関連話題紹介

講習会終了後のアンケート集計結果(回答数 17)を表 2 に示すが、参加した満足度の平均は 5 点満点中、3.88 と比較的高かった。また、参加者から表 3 に示すご意見を頂いた。今後の講習会の参考にしたい。

表 2 アンケート集計結果

評 点	講習会の時間		講習会の講義内容 (プレゼン)		配布資料の内容		サンプルプログラム 内容		参加した満足度	
1	短い	0	簡単	0	簡単	0	簡単	1	不満	0
2		1		1		1		4		1
3	適切	12	適切	5	適切	5	適切	10	普通	2
4		3		10		10		2		12
5	長い	1	難	1	難	1	難	0	満足	2
	平均	3.24	平均	3.65	平均	3.65	平均	2.76	平均	3.88

表 3 講習会に対するご意見(原文ママ)

- 本日はありがとうございました。
- 後半が駆け足すぎて、あまり理解できなかった。
- 中級も参加したいです。ダムブレイクだとうれしいです。
- 流体のことはあまり知らない想定でレクチャーしていただいているように感じるが、通信関係（ssh がどうか）は基礎知識が足りずついていけない場面があった。通信のことはどこで勉強してよいかわからないので、OpenFOAM 入門のように事前に目を通しておけばいい資料を指定してほしい。
- 大変勉強になりました。ありがとうございました。
- ベンチマークではあまり効果が無いようでしたが、メッシュの設定や OpenFoam の設定による（超）並列化による効果とハードウェアリソースを最大限に使うノウハウが知りたいです。
- OpenFOAM 講習会は希望者が多く、参加できない場合が多々あるので、録画データを希望者に公開していただけないでしょうか。入門編に参加できず、初級の内容は少々レベルが高かった印象を受けました（初級に参加できたことは幸いでした）。

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

目 次

センターから

サービス休止等のお知らせ	1
システム変更等のお知らせ	3
2021年度の利用申込（新規・継続）について	5
東京大学柏キャンパス一般公開2020	7
2020年度インターン・後期	
東京大学情報基盤センター「若手・女性利用者推薦」採択課題	8
研究成果登録のお願い	14
部門の人事異動	15
8月・9月のジョブ統計	16

ユーザーから

温水冷却の効率に関する定量的評価の試み	21
縦長行列のQR分解に対する各種アルゴリズムの比較： Oakforest-PACS上での性能評価	28

教育活動報告

第136回お試しアカウント付き並列プログラミング講習会 「OpenFOAM入門」（オンライン）実施報告	40
第137回お試しアカウント付き並列プログラミング講習会 「第2回GPUミニキャンプ～GPU化にチャレンジする会～」（オンライン） 実施報告	42
第138回お試しアカウント付き並列プログラミング講習会 「スーパーコンピューター超入門」（オンライン）実施報告	46
第139回お試しアカウント付き並列プログラミング講習会 「OpenFOAM初級」（オンライン）実施報告	49

原稿募集	51
------	----