GPU を搭載した計算機の効率的な運用方法の検討と実装

當山 達也 [1)], 尾形 幸亮 [1)], 疋田 淳一 [1)]

1) 京都大学 企画・情報部

tohyama.tatsuya.6x@kyoto-u.ac.jp

Design and implementation for efficient operation of the GPU servers

Tatsuya Tohyama[1)], Kosuke Ogata[1)], Junichi Hikita[1)]

1) Planning and Information Management Department, Kyoto Univ.

概要

スーパーコンピュータは，様々な研究分野において用いられており，計算処理だけではなく，数値計算モデルの生成や計算結果の可視化といった処理にも多く利用されている．そのため，スーパーコンピュータを運用する多くの機関では，計算用途の計算機に加えて，画像処理装置 (GPU) を搭載した計算機を導入している．京都大学 学術情報メディアセンターでも，可視化の需要は増えてきており，GPU を搭載した計算機 (以下，GPU サーバ) の供用を 2018 年 12 月から開始した．しかし，需要に対して用意した GPU サーバは少なく，需要と供給のバランスを平準化させるためには，効率的な運用を行う必要がある．本稿では，GPU サーバを導入するにあたり，効率的な運用の実現に向けて検討した内容を述べ，検討結果を元に実装した予約システムを紹介する．

1 はじめに

スーパーコンピュータを利用する研究分野は，近年大幅に増加しており，従来のように高速に計算を行い計算結果を出力すれば良いといった単純なものから，数値計算モデルの生成 (プリプロセッサ) を行い，計算結果を可視化 (ポストプロセッサ) するアプリケーションの利用まで，多種多様なニーズが存在する．

そこで，京都大学 学術情報メディアセンター (以下，本センター) では，GPU を搭載した計算機（以下，GPU サーバ) を現行システムに追加で用意し，可視化用途として 2018 年 12 月から供用を開始した．しかし，プリポスト処理を行う利用者数に対して，導入した GPU サーバは 2 台であり，1 台の機器を 1 利用者に割り当てると機器が早々に枯渇する状況が考えられ，多くの利用者が利用できるよう，効率的な運用が求められる．

本センターでは，スーパーコンピュータ利用者向けにサービス申請などを行うことが可能なポータルサイト (以下，利用者ポータル)[1] を運用している．そこで，GPU サーバの予約機能を利用者ポータルに追加構築し，利用者が自由に GPU サーバを予約可能な仕組みを整えることで，効率的な運用を実現した．

本稿では，GPU サーバの効率的な運用手法について検討した内容ならびに結果を述べ，本センターで構築した効率的な運用を支援する予約システムについて紹介する．

2 スーパーコンピュータシステムの概要

2.1 構成

図 1 に，本センターが所有するスーパーコンピュータのシステム構成を示す．

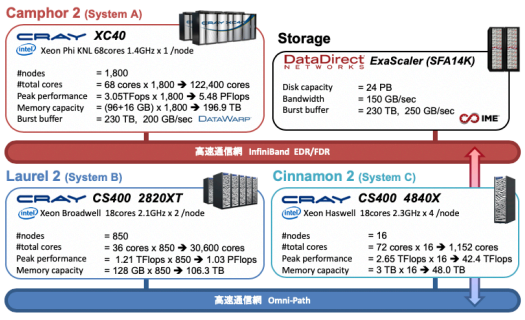

図 1 スーパーコンピュータシステムの構成図

すべてのサブシステムには，Intel 社製の CPU を搭載しており，GPU は搭載していない．また，各サブシステムのログインノードも同様の構造となっている．

2.2 新たに導入した GPU サーバの諸元

本センターでは，可視化需要の増加に対応するべく，2018 年 12 月ならびに 2019 年 4 月に 1 台ずつ GPU サーバの供用を開始した．GPU サーバの諸元は，表 1 ならびに 表 2 のとおりである．

表 1　1 号機の諸元

OS	RedHat Enterprise Linux7
CPU	Intel(R) Xeon(R) Gold 6140 2.30GHz (18 コア) x 2
メモリ	512GB
GPU	NVIDIA Quadro P4000 (8 GB GDDR5) x 2

表 2　2 号機の諸元

OS	RedHat Enterprise Linux7
CPU	Intel(R) Xeon(R) Silver 4110 2.10GHz (8 コア) x 2
メモリ	512GB
GPU	NVIDIA Quadro GV100 (32 GB GDDR5) x 2

GPU サーバは，Laurel2(System B) や，Cinnamon 2(System C) と同じ OS ならびに互換のあるプロセッサを採用することによって，従来動作していた GUI アプリケーションが，GPU サーバでも問題なく動作するように配慮した．

また，Camphor2(System A) については，従来から GUI アプリケーションの利用をサポートしていないため，当該機器に搭載されている OS ならびにプロセッサに類似した GPU サーバの導入は見送った．

2.3 現行システムの可視化手法

プリポスト処理を行う GUI アプリケーションを起動する際には，ジョブ投入用のコマンドではなく，”xrun”と呼ばれるコマンドを用意している．当該コマンドが実行された場合は，ログインノードを通して X Window が利用者の端末まで転送されるような実装となっている．

また，Linux の GUI を利用者の端末に転送するには，X Window System が必要となるため，本センターでは，OpenText 社の Exceed onDemand[2] を利用者に提供している．その他の方法として，Windows であれば，Xming[3]，MacOS であれば，XQuarz[4] といった X window アプリケーションの利用や，利用者が計算ノードで VNC サーバを立ち上げ，リモートデスクトップ機能を用いた操作を行うことも可能である．

3 GPU サーバの運用手法の検討

3.1 GPU サーバの運用設計

プリポスト処理を行う GUI アプリケーションは比較的多くのメモリを消費する傾向にある．このため，利用者自身が利用する GUI アプリケーションに応じてメモリを確保することが好ましく，単純に搭載メモリ量からシステムで用いるメモリ量を減算し，1 人あたりに割り当てる固定のメモリ量で按分して，最大利用者数を求めれば良いというわけにはいかない．このため，利用者が確保したいメモリ量を自由に設定することが可能で，搭載メモリ量を上回らないように利用者の受入判断を行う仕組みが必要である．

そこで，利用者ポータルに，GPU サーバの予約システムを追加実装することで解決できると考える．また，利用者が予約処理を始めた段階で計算機の現在の利用者数，ならびにメモリの利用量，事前に予約済みのリソースを取得できるように予約システムを設計することで，現時点のリソースに空きがあり，受入が可能であれば，現時点からの予約を受け入れる事が可能となるのが望ましい．

これに加えて，GPU に強く依存するような GUI アプリケーションを実行する利用者にむけて，GPU を専有できるような仕組みが必要であると考える．ただし，GPU のすべてを専有可能な設計とした場合，当該計算機の最大利用者数は GPU の数となる．効率的な運用という観点から考えると，最低 1 つの GPU は共有利用が可能となるように実装するのが望ましい．

3.2 効率的な運用に向けた検討

3.1 節で述べた運用設計をもとに，多くの利用者が GPU サーバを効率良く利用することが可能な手法を検討する必要がある．

3.2.1 制御可能なリソースの調査

GPU サーバに複数の利用者を収容してリソースを共有するにあたり，他の利用者の影響を受けないよう，論理的に GPU サーバのもつリソースを分離して提供する必要がある．リソースの利用を制限・隔離する機能として Linux OS 環境で広く利用されている cgroup[5] を調査したところ，CPU コア，メモリについては，制御可能であることを確認した．

次に，GPU に関する制御機能について調査を行った．cgroup を用いて制御可能な項目としては，デバ

イスの利用可否のみとなっており，GPU のコア単位での利用制御や，GPU メモリの利用制限は困難であることを確認した．また，GUI アプリケーションを GPU サーバで実行するにあたり，VirtualGL[6] を用いて描画処理を行うことで，可視化環境が大幅に改善されることを確認している [7]．そこで，VirtualGL によるリソース制御の可否についても調査を実施した．VirtualGL の実行パラメータを用いることで，利用する GPU の制御を行うことは可能だが，cgroup と同様に GPU のコア単位での利用制御や GPU メモリの利用制限は，困難であることを確認した．

以上の結果から，システムとして制御可能なリソースは，下記の 3 点であると考える．

- CPU コア
- メモリ
- GPU(デバイス単位)

3.2.2 最大同時利用者数の検討

通常，Linux OS でアプリケーションを実行する際には，CPU，メモリ，GPU，GPU メモリをそれぞれ使用する．これらのリソースのうち，3.2.1 節での調査において，システムとして制御可能と判定された CPU，メモリ，GPU のいずれかを用いることで，状況に応じた最大同時利用者数を算出することが可能か検討した．

まずはじめに，CPU ならびに GPU について検討を行った．CPU や GPU については，コア数よりも多数のスレッドが存在した場合は細かく時間を区切って順番に実行させることから，多数の利用者がアプリケーションを同時に実行したとしても，動作速度の低下は見られるが，強制終了されることはない．このため，CPU や GPU については最大同時利用者数を判断するにあたって，一定の目安を設定する必要はあるが，最大同時利用者数を算出するにあたって重要であるとは考えられない．

次に，メモリについて検討を行った．Linux OS の場合，メモリが不足してシステムの維持が困難になるとシステムが判断した場合には，メモリリソースを多く消費しているプロセスを強制的に終了させる機能 (OOM Killer[8]) が搭載されている．このため，システムに搭載しているメモリ量を超過してメモリを確保すると，アプリケーションの実行に多大な影響を及ぼす可能性が高く，最大同時利用者数を算出するにあたって適していると考えられる．

以上の検討結果から，最大同時利用者数は計算機に搭載しているメモリ量ならびに，現在のメモリの利用状況から算出することが望ましい．

3.2.3 利用者に割り当てる CPU コア数の検討

本センターでよく利用されている GUI アプリケーションの推奨値は，表 3 のとおりである．

表 3　プリポスト処理を行うアプリケーションの動作推奨値 (CPU)

アプリケーション名	推奨値
ANSYS [9]	4-12 コア
AVS/Express	記載なし
ENVI/IDL [10]	複数コア推奨
MATLAB [11]	4 コア
GaussView	記載なし

表 3 の結果から，4 コア程度の CPU コアを利用者に割り当てることで，プリポスト処理を行う GUI アプリケーションは問題なく動作するものと考えられる．

3.2.4 予約を受け付ける際に制限すべき項目の検討

関連文献 [12] を参考に検討した結果から，予約を受け付ける際に制限すべき項目として，以下の 3 点が考えられる．

- 予約受付期間
- 最大連続利用時間
- 同時予約数

まずはじめに，予約受付期間ならびに最大連続利用時間について検討を行った．本センターが提供するスーパーコンピューターのサービスコースを利用し，従来の計算ノードでプリポスト処理を行う GUI アプリケーションを実行した際の最大連続利用時間は，336 時間 (14 日) となっている [13]．しかし，GPU サーバを用いることによって，従来よりも計算時間は短縮されると考えられ，公平性の観点から最大連続利用期間を 72 時間 (3 日) として設定し，利用状況に応じて再度検討するといった対応を行うこととした．

次に，同時予約数について検討を行った．複数の予約を許可すると，一人の利用者が機器を専有利用する可能性が考えられ，多くの利用者へ利用機会を提供するといった観点から，好ましくないと考える．そこで，予約については最大 1 件とし，当該予約の期間満了または，利用者がキャンセル処理を行うと，改めて予約が可能となるように処理するのが望ましい．

以上の結果から，予約を受け付ける際には，表 4 に挙げた項目を満たす必要があると考えられる．

表 4　予約を受け付ける際に制限すべき項目

予約受付期間	2 週間先まで
最大連続利用時間	3 日間
同時予約数	1 件

4　GPU サーバ予約システムの概要

第 3 章での検討結果から，GPU サーバを効率的に運用する手法として，GPU サーバの予約システムを利用者ポータルに組み込むことにより，利用者が必要とするリソースを確保する仕組みの構築が必要と考えられる．そこで，実際に 3.1 節で検討した運用設計を元に，予約システムの実装を行った．

本章では，実際に実装した予約システムの特徴について説明を行い，実際に利用者に提供する予約画面の一部を紹介する．

4.1　GPU サーバ予約システムの実装

予約システムを構築するにあたっては，REST API を持った Web アプリケーションを，利用者ポータルならびに GPU サーバに実装・配備することで，情報の円滑なやりとりを実現した．また，システム間での情報のやり取りは，基本的に json 形式を用いて行い，多種の情報を一度に送受信するような実装とした．

次に，GPU サーバを運用するにあたって，一番重要と考えたメモリ管理について紹介する．本実装では，GUI アプリケーションの実行リクエストを利用者が GPU サーバへ投入した際に，予約状況ならびに予約として確保したメモリ量を，利用者ポータルから REST API を通して取得する仕組みとした．利用者ポータルから取得した利用可能なメモリ量は，cgroup の管理コマンドを用いて，アプリケーションの起動前に GPU サーバに設定することで，柔軟かつ確実なメモリ割当を実現した．また，利用時間のコントロールについては，Linux OS で利用可能な timeout コマンドを利用し，引数に利用可能時間 (秒) を渡すことで，アプリケーションが利用予定時間を越えて実行されない仕組みを取り入れた．

次に，予約について紹介する．本実装では，利用者が予約システムを用いて予約処理を実行する際に，REST API を用いて GPU サーバに現状の利用者数ならびに，メモリの利用状況を確認する．取得した結果と利用者ポータルに登録されている予約内容を突き合わせ，時間ごとに最大となる利用者数ならびにメモリの利用量を計算した上で，新規予約希望者が希望するメモリ量の受入可否を時間毎に計算し，表形式で表示する仕組みを整えた．これによって，GPU サーバのリソースに空きがあれば，利用者が必要な時に必要なリソースを最大限確保可能となる．

最後に，GPU の専有利用について紹介する．3.1 節での検討結果から，GPU のうち，最低 1GPU は共有利用として確保しておく必要がある．本実装では，GPU サーバごとに利用可能な GPU 数を管理するデータベースを構築し，利用可能な GPU 数が閾値以上であれば，専有利用することができるように実装した．具体的には，GPU を専有利用する予約処理を利用者が選択した場合は，データベースの利用可能な GPU 数が 2 以上となっている期間のみ，予約可能とすることで，共有利用で使用する GPU を確保する仕組みを整えた．

4.2　予約の流れ

利用者は利用者ポータルにログインし，サービス申請欄から GPU サーバの予約ページへ遷移することで，GPU サーバの予約を行うことができる．

図 2　リソース予約画面 (リソース指定欄)

予約ページは，図 2 のようになっており，利用者は利用したい GPU サーバを選択し，確保するメモリ量を入力する．また，GPU の専有利用を希望する場合は，GPU 欄のドロップダウンリストを専有に変更することで，専有予約が可能な日程を表示することが可能となる．

リソースの予約が完了すると，日時を選択する画面に遷移する．日時予約画面を表示する前に，利用者ポータルと GPU サーバとの間で，現在使われているリソース量の突き合わせが行われる．その結果とすでに予約されているリソースを鑑み，利用者が希望するリソースが利用可能な日時には”○”を，利用不可能な日時には”×”を表示した日程表を 14 日分生成し表示

する．生成された日程表の例を，図 3 に示す．

図 3　リソース予約画面 (日時指定選択前)

表示された日程表から，開始日時と終了日時を選択する仕様とすることで，予約期間外の日時を指定できないように工夫した．これに加えて，開始日時と終了日時を選択すると，予約期間が図 4 のように青く塗られ，予約した期間を利用者が直感的に認識できるように実装した．

図 4　リソース予約画面 (日時指定選択後)

予約開始ならびに終了日時を選択し，利用者が予約確定ボタンを押下すると，予約処理が実行される．予約処理が正常に終了すると，利用者ポータルの GPU サーバ予約ページに自動的に遷移し，予約完了の旨と予約状況が 図 5 のように表示される．

また，利用者ポータルの GPU サーバ予約ページは，表示時間以降の予約がない場合はリソース予約画面として機能し，表示時間以降の予約がある場合は，予約情報表示ならびに予約取消を行う画面として機能するようになっている．これによって，一人の利用者が重複予約を行うことができないような仕組みとなっている．

図 5　リソース予約完了画面

5　まとめ

本稿では，GPU サーバの効率的な運用にむけた検討ならびに，予約システムの実装について紹介した．実際に GPU サーバの運用を開始すると，現時点では意図しない問題が多く発生すると考えられる．これらの問題の解消は当然のことながら，利用者の要望ならびに利用実態を元に積極的な改善を進めることで，よりよい可視化環境を利用者に提供できるよう進めていく．

参考文献

[1] 京都大学学術情報メディアセンター スーパーコンピュータシステム 利用者ポータル
https://web.kudpc.kyoto-u.ac.jp/portal/
(2019 年 8 月 9 日閲覧)

[2] OpenText Exceed onDemand
https://www.macnica.net/opentext/eod.html
(2019 年 8 月 9 日閲覧)

[3] Xming
http://www.straightrunning.com/XmingNotes/
(2019 年 8 月 9 日閲覧)

[4] XQuartz
https://www.xquartz.org/
(2019 年 8 月 9 日閲覧)

[5] CGROUPS
https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt
(2019 年 8 月 9 日閲覧)

[6] VirtualGL
https://www.virtualgl.org
(2019 年 8 月 9 日閲覧)

[7] 疋田淳一，當山達也，尾形幸亮. VirtualGL を用

いたリモート可視化環境の改善，AXIES2018, 2018.

[8] Out Of Memory Management
https://www.kernel.org/doc/gorman/html/understand/understand016.html
(2019 年 8 月 9 日閲覧)

[9] ANSYS System Hardware Requirements
https://www.ozeninc.com/ansys-system-hardware-requirements/
(2019 年 8 月 9 日閲覧)

[10] ENVI System Requirements
https://www.harrisgeospatial.com/docs/using_envi_PlatformSupportTable.html
(2019 年 8 月 9 日閲覧)

[11] System Requirements for MATLAB R2019a
https://jp.mathworks.com/support/requirements/matlab-system-requirements.html
(2019 年 8 月 9 日閲覧)

[12] 平島智将, 原田浩睦, 小野真, 上田将嗣, 南里豪志. 可視化サーバ予約システムの導入と運用，AXIES2015, 2015.

[13] 京都大学情報環境機構 コンピューティングサービス サービスコース選択ガイド
http://www.iimc.kyoto-u.ac.jp/ja/services/comp/apply/guide.html
(2019 年 8 月 9 日閲覧)