doitnow

ホワイトペーパー

AMD によるパフォーマンス向上

Amazon EC2 Hpc7a インスタンスのベンチマーク

Increasing performance with AMD

Amazon EC2 Hpc7a instances benchmarking

AMD

目次

1.	**はじめに**	3
1.1	目的	3
1.2	クラウドインスタンスについて	3
1.2.1	Amazon EC2 hpc6a.48xlarge	4
1.2.2	Amazon EC2 hpc7a.96xlarge	4
1.2.3	Amazon EC2 hpc7a.48xlarge	4
1.2.4	Amazon EC2 hpc7a.24xlarge	4
1.2.5	Amazon EC2 hpc7a.12xlarge	4
1.3	ソフトウェアについて	5
1.4	モデルについて	6
1.4.1	Altair® Radioss®	6
1.4.2	Altair® AcuSolve®	6
2.	**アーキテクチャ設計**	7
3.	**インストール**	8
3.1	AWS ParallelCluster	8
3.1.1	スケジューラ	8
3.1.2	ストレージ	8
3.2	Altair ソフトウェアとライセンスサービス	8
3.3	ユーザーデータリポジトリ	8
4.	**ベンチマーク実行**	8
4.1	ベースライン設定	8
4.2	MPIチューニング	9
4.2.1	Amazon EC2 Hpc6a MPI 設定	9
4.2.2	Amazon EC2 Hpc7a MPI 設定	9
4.3	スケーラビリティ	10
4.4	パフォーマンス結果	10
4.4.1	Altair® Radioss®	10
4.4.2	Altair® AcuSolve®	12
4.5	ジョブ価格結果	13
4.5.1	Altair® Radioss®	14
4.5.2	Altair® AcuSolve®	15
5.	**結論**	16
5.1	クレジットと謝辞	16
6.	**付録**	17
6.1	Slurm 用サブミッションコマンド例	17
6.2	ジョブスクリプト	17
6.2.1	Radioss	17
6.2.2	AcuSolve	18
6.3	ParallelCluster 設定ファイル	19
6.3.1	eu-north-1 の Amazon EC2 Hpc6a	19
6.3.2	eu-west-1 の Amazon EC2 Hpc7a	20

Index

1.	**Introduction**	3
1.1	Objective	3
1.2	About the cloud instances	3
1.2.1	Amazon EC2 hpc6a.48xlarge	4
1.2.2	Amazon EC2 hpc7a.96xlarge	4
1.2.3	Amazon EC2 hpc7a.48xlarge	4
1.2.4	Amazon EC2 hpc7a.24xlarge	4
1.2.5	Amazon EC2 hpc7a.12xlarge	4
1.3	About the software	5
1.4	About the models	6
1.4.1	Altair® Radioss®	6
1.4.2	Altair® AcuSolve®	6
2.	**Architecture design**	7
3.	**Installation**	8
3.1	AWS ParallelCluster	8
3.1.1	Scheduler	8
3.1.2	Storage	8
3.2	Altair software and licensing services	8
3.3	User Data Repository	8
4.	**Benchmark execution**	8
4.1	Baselining	8
4.2	MPI tuning	9
4.2.1	Amazon EC2 Hpc6a MPI settings	9
4.2.2	Amazon EC2 Hpc7a MPI settings	9
4.3	Scalability	10
4.4	Performance results	10
4.4.1	Altair® Radioss®	10
4.4.2	Altair® AcuSolve®	12
4.5	Job price results	13
4.5.1	Altair® Radioss®	14
4.5.2	Altair® AcuSolve®	15
5.	**Conclusions**	16
5.1	Credits and special thanks	16
6.	**Appendix**	17
6.1	Example submission command for Slurm	17
6.2	Job scripts	17
6.2.1	Radioss	17
6.2.2	AcuSolve	18
6.3	ParallelCluster configuration files	19
6.3.1	Amazon EC2 Hpc6a in eu-north-1	19
6.3.2	Amazon EC2 Hpc7a in eu-west-1	20

1. はじめに

Do IT Now は、AWS 上の AMD ベースの HPC 最適化インスタンスを使用してクラウドに HPC クラスターをデプロイし、インフラストラクチャに Altair アプリケーションをインストールし、参照モデルを用いて各アプリケーションごとにベンチマークを実施した。また、各アプリケーションとインスタンスの組み合わせについて、パフォーマンスとコストの観点から結果を比較した。

この分析において、Do IT Now はクラウド環境におけるアプリケーション、インスタンス、および HPC アーキテクチャ全体のチューニングに向けた最適化パターンとパラメータの可能性についても探求した。

1.1 目的

本プロジェクトの目的は、クラウド環境で実施したベンチマークの結果を提供し、ベースラインとなる性能結果と、Do IT Now チームがインフラストラクチャ、ソフトウェア、およびそのパラメータに適用した追加の最適化を示すことである。

追加目的として、Do IT Now はプロジェクト実行中に実施した様々な設計ステップ、発生した問題とその解決方法、得られた教訓について提示・説明する。

1.2 クラウドインスタンスについて

選定したクラウドインスタンスは、同一の AWS HPC インスタンスファミリー内で、2 世代の異なる AMD EPYC™ プロセッサをベースとしている：

- Amazon EC2 Hpc6a インスタンス。2 つの第 3 世代 AMD EPYC™ 7003 シリーズ 48 コアプロセッサを搭載し、全コアターボ周波数最大 3.6 GHz、100 Gb/s ネットワーク。
（https://aws.amazon.com/ec2/instance-types/hpc6a/）

- Amazon EC2 Hpc7a インスタンス。2 つの第 4 世代 AMD EPYC™ 9004 シリーズ 96 コアプロセッサを搭載し、全コアターボ周波数最大 3.7 GHz、DDR5 RAM、300 Gb/s ネットワーク。Hpc7a インスタンスでは、第 4 世代 AMD EPYC™ プロセッサの 24 コア、48 コア、96 コア構成も選択可能。
（https://aws.amazon.com/ec2/instance-types/hpc7a/）

Hpc7a の場合、AWS はより小さなインスタンスサイズを提供しており、他のリソースを一定に保ちつつ、ワークロード要件に合わせてアクティブ化する CPU コア数を選択できるようになっている。実際、より小さいサイズ（96 コア、48 コア、24 コア）ではコアあたりのメモリ容量とメモリ帯域幅が増加する。これはソルバーの性能に重大な影響を与え、コア単位でライセンスされる商用ソフトウェアの場合には顕著な効果を発揮する。さらに、最適なコアトポロジー構成が AWS によって自動的に選択されるため、性能が最大化され、実行時の複雑なコアピンニング設定を行う必要がなくなる。

Hpc6a およびすべての Hpc7a サイズでベンチマークを実施した。

1. Introduction

Do IT Now deployed an HPC cluster in the cloud, using AMD-based HPC-optimized instances on AWS, installed Altair applications on the infrastructure, and conducted benchmarks with reference models for each application. Do IT Now also compared results in terms of performance and costs for each application and instance combination.

During this analysis, Do IT Now also explored possible optimization patterns and parameters to tune the applications, the instances, and the overall HPC architecture in the cloud.

1.1 Objective

The objective of this project is to provide the results of the benchmarks conducted in the cloud environment, showing the baseline performance results and any additional optimizations that the Do IT Now team applied to the infrastructure, the software, and its parameters.

As an additional objective, Do IT Now will present and describe the various design steps, any issue and how it was solved, and the lessons learned during the project execution.

1.2 About the cloud instances

The selected cloud instances are based on AMD EPYC™ processors of two different generations, all within the same AWS HPC instance family:

- Amazon EC2 Hpc6a instances, featuring two 3rd Gen AMD EPYC™ 7003 series 48-core processors with up to 3.6 GHz all-core turbo frequency and 100 Gb/s networking. (https://aws.amazon.com/ec2/instance-types/hpc6a/)

- Amazon EC2 Hpc7a instances, featuring two 4th Gen AMD EPYC™ 9004 series 96-core processors with up to 3.7 GHz all-core turbo frequency, DDR5 RAM, and 300 Gb/s networking. Hpc7a instances also offer 24-, 48-, or 96-core 4th Gen AMD EPYC™ processors. (https://aws.amazon.com/ec2/instance-types/hpc7a/)

In case of Hpc7a, AWS is offering smaller instance sizes that make it easier for customers to pick a smaller number of CPU cores to activate while keeping all other resources constant based on their workload requirements. In fact, the smaller sizes (with 96, 48, and 24 cores) increase the memory per core and memory bandwidth per core. This can have a serious impact on solver performance and becomes tangible in the case of commercial software that's licensed on a per-core basis. Additionally, the best core topology configuration is automatically selected by AWS, maximizing performance and eliminating the need for complicated core pinning configurations at run time.

We ran the benchmarks on Hpc6a and all Hpc7a sizes.

1.2.1 Amazon EC2 hpc6a.48xlarge

詳細仕様:

- CPU: AMD EPYC™ 7R13 プロセッサ @ 3.6 GHz
- コア数: 48 x 2 ソケット、1コアあたり1スレッド、合計 96 物理コア
- RAM: 384 GiB
- ネットワーク: 1 x Elastic Fabric Adapter ネットワークインターフェイス、100 Gb/s 帯域幅

1.2.2 Amazon EC2 hpc7a.96xlarge

詳細仕様:

- CPU: AMD EPYC™ 9R14 プロセッサ @ 3.7 GHz
- コア数: 96 x 2 ソケット、1 コアあたり1スレッド、合計 192 物理コア
- RAM: 768 GiB
- ネットワーク: 2 x Elastic Fabric Adapter ネットワークインターフェイス、300 Gb/s 帯域幅

1.2.3 Amazon EC2 hpc7a.48xlarge

詳細仕様:

- CPU: AMD EPYC™ 9R14 プロセッサ @ 3.7 GHz
- コア数: 48 x 2 ソケット、1 コアあたり 1 スレッド、合計 96 物理コア
- RAM: 768 GiB
- ネットワーク: 2 x Elastic Fabric Adapter ネットワークインターフェイス、300 Gb/s 帯域幅

1.2.4 Amazon EC2 hpc7a.24xlarge

詳細仕様:

- CPU: AMD EPYC™ 9R14 プロセッサ @ 3.7 GHz
- コア数: 24 x 2 ソケット、1 コアあたり 1 スレッド、合計 48 物理コア
- RAM: 768 GiB
- ネットワーク: 2 x Elastic Fabric Adapter ネットワークインターフェイス、300 Gb/s 帯域幅

1.2.5 Amazon EC2 hpc7a.12xlarge

詳細仕様:

- CPU: AMD EPYC™ 9R14 プロセッサ @ 3.7 GHz
- コア数: 12 x 2 ソケット、1 コアあたり 1 スレッド、合計 24 物理コア
- RAM: 768 GiB
- ネットワーク: 2 x Elastic Fabric Adapter ネットワークインターフェイス、300 Gb/s 帯域幅

1.2.1 Amazon EC2 hpc6a.48xlarge

Full specifications:

- CPU: AMD EPYC™ 7R13 Processor @ 3.6 GHz
- Cores: 48 x 2 sockets, 1 thread per core, 96 total physical cores
- RAM: 384 GiB
- Network: 1 x Elastic Fabric Adapter network interface, 100 Gb/s bandwidth

1.2.2 Amazon EC2 hpc7a.96xlarge

Full specifications:

- CPU: AMD EPYC™ 9R14 Processor @ 3.7 GHz
- Cores: 96 x 2 sockets, 1 thread per core, 192 total physical cores
- RAM: 768 GiB
- Network: 2 x Elastic Fabric Adapter network interface, 300 Gb/s bandwidth

1.2.3 Amazon EC2 hpc7a.48xlarge

Full specifications:

- CPU: AMD EPYC™ 9R14 Processor @ 3.7 GHz
- Cores: 48 x 2 sockets, 1 thread per core, 96 total physical cores
- RAM: 768 GiB
- Network: 2 x Elastic Fabric Adapter network interface, 300 Gb/s bandwidth

1.2.4 Amazon EC2 hpc7a.24xlarge

Full specifications:

- CPU: AMD EPYC™ 9R14 Processor @ 3.7 GHz
- Cores: 24 x 2 sockets, 1 thread per core, 48 total physical cores
- RAM: 768 GiB
- Network: 2 x Elastic Fabric Adapter network interface, 300 Gb/s bandwidth

1.2.5 Amazon EC2 hpc7a.12xlarge

Full specifications:

- CPU: AMD EPYC™ 9R14 Processor @ 3.7 GHz
- Cores: 12 x 2 sockets, 1 thread per core, 24 total physical cores
- RAM: 768 GiB
- Network: 2 x Elastic Fabric Adapter network interface, 300 Gb/s bandwidth

1.3 ソフトウェアについて

ベンチマークで使用したソフトウェアは Altair の CAE 製品ラインの一部である：

- Altair® Radioss®：動的荷重下における高度な非線形問題の製品性能評価・最適化を実現する最先端解析ソリューション。あらゆる産業分野で世界的に採用され、複雑な設計の衝突安全性能、安全性、製造性を向上させる。

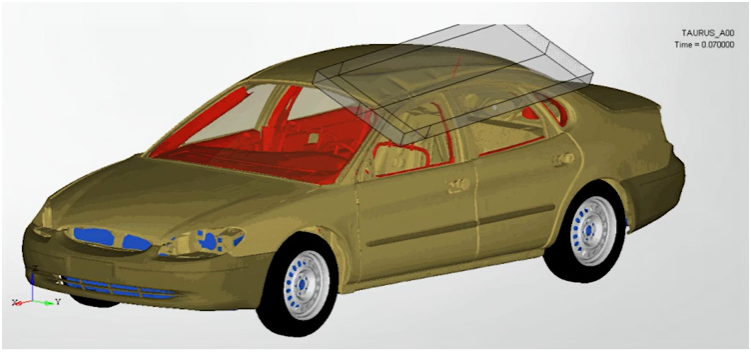


図1 – 当初開発された NCAC FE Taurus モデルから派生した Altair 改良版（画像提供：Altair）

- Altair® AcuSolve®：最も要求の厳しい産業・科学アプリケーションを解決可能な、最先端の汎用計算流体力学（CFD）ソルバー。堅牢でスケーラブルなソルバー技術により、ユーザーに比類のない精度を提供する。

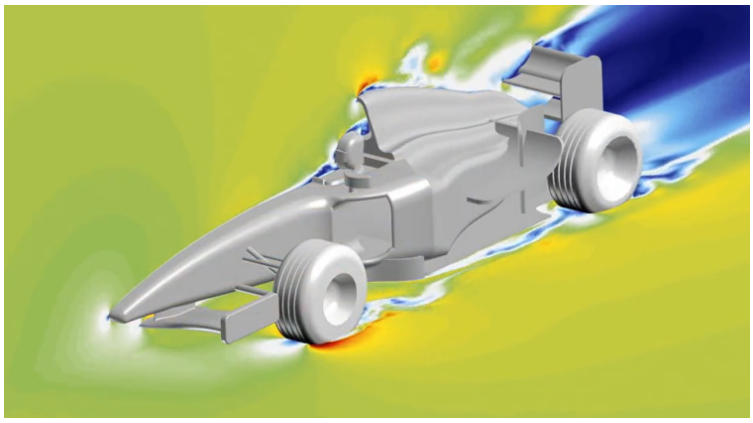


図2 – Altair® AcuSolve® 出力例（画像提供：Altair）

1.3 About the software

The software used for the benchmarks are part of the Altair CAE product line:

- Altair® Radioss®, a leading analysis solution to evaluate and optimize product performance for highly nonlinear problems under dynamic loadings. Used worldwide across all industry sectors, it improves the crashworthiness, safety, and manufacturability of complex designs.

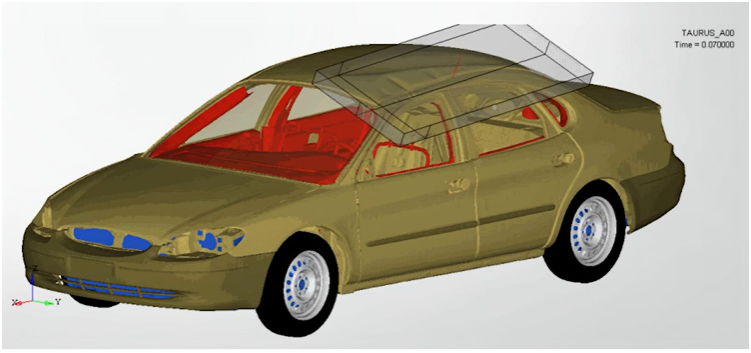


Figure 1 – Altair refined version derived from originally developed NCAC FE Taurus model (image/courtesy of Altair)

- Altair® AcuSolve®, a leading general-purpose Computational Fluid Dynamics (CFD) solver that is capable of solving the most demanding industrial and scientific applications. Robust and scalable solver technology empowers users by providing unparalleled accuracy.

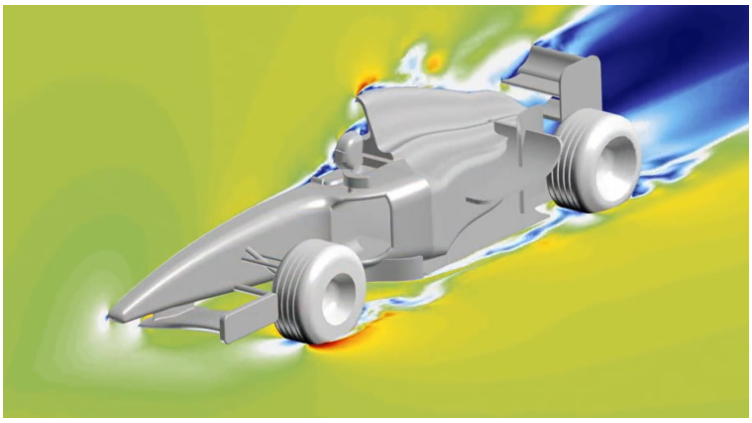


Figure 2 – Altair® AcuSolve® example output (image/courtesy of Altair)

1.4 モデルについて

1.4.1 Altair® Radioss®

このソフトウェアで使用されるモデルは、OpenRadioss™ の公開リポジトリから取得された公開モデル「Taurus 10 million finite elements」である。多くの CPU を持つ HPC クラスターでの HPC 性能のベンチマークに適している。このベンチマークは 1000 万個の有限要素からなる精緻なメッシュを有する。

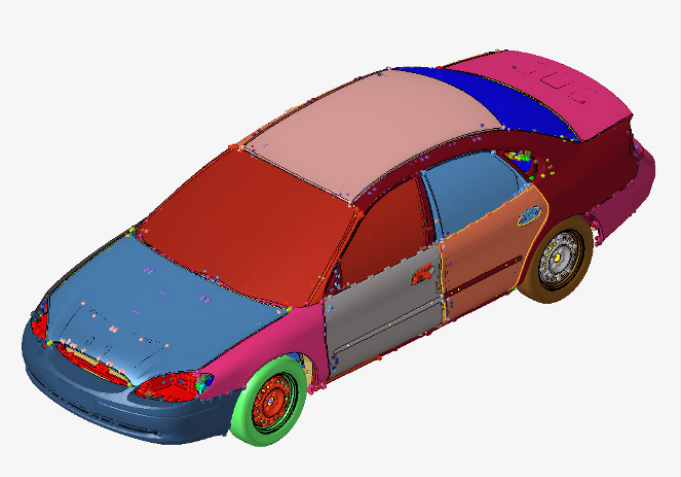


図3 – 当初開発された NCAC FE Taurus モデルから派生した Altair 改良版（画像提供：Altair）

このモデルには、シミュレーション時間に基づく 3 つの異なる設定がある：

- フルシミュレーション、120 ミリ秒。
- 短縮シミュレーション、10 ミリ秒: 多くのノード/ CPU での高速性能・スケーラビリティ試験に最適。
- 超短縮シミュレーション、2 ミリ秒: HPC クラスター機能のテストに有用。

ベンチマークでは短縮シミュレーション（シミュレーション時間 10 ミリ秒）を使用した。

1.4.2 Altair® AcuSolve®

AcuSolve 用モデルは、HPC 環境における高いスケーラビリティを実証する好例として、Altair より直接提供された。

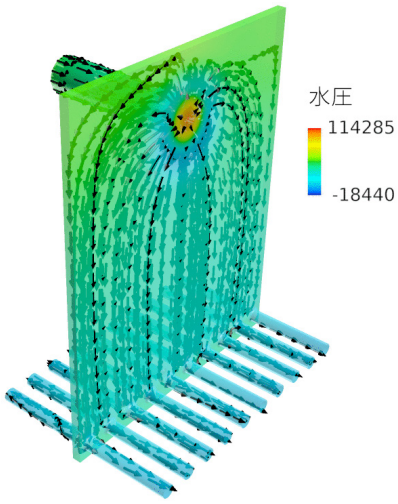


図4 –「Impinging Nozzle」モデル（画像提供：Altair）

1.4 About the models

1.4.1 Altair® Radioss®

The model used for this software is the publicly available “Taurus 10 million finite elements” model, recovered from the OpenRadioss™ public repository. It is well suited to benchmarking HPC performance in HPC clusters with many CPUs. This benchmark has a refined mesh with 10 million finite elements.

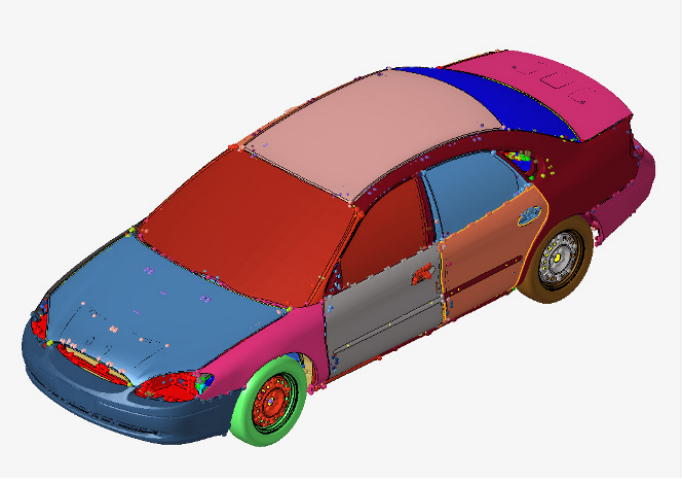


Figure 3 – Altair refined version derived from originally developed NCAC LS-DYNA FE Taurus model (image/courtesy of Altair)

The model has three different settings, based on the simulation time:

- Full simulation, 120 milliseconds.
- Shorter simulation, 10 milliseconds: best suited for fast performance and scalability testing on many nodes/CPUs.
- Very short simulation, 2 milliseconds: useful to test the HPC cluster functions.

For our benchmarks we use the “Shorter” simulation test case, with 10 milliseconds of simulation time.

1.4.2 Altair® AcuSolve®

The model used for AcuSolve has been sent to us directly from Altair, as they considered it a good example of a simulation that will demonstrate high scalability in an HPC environment.

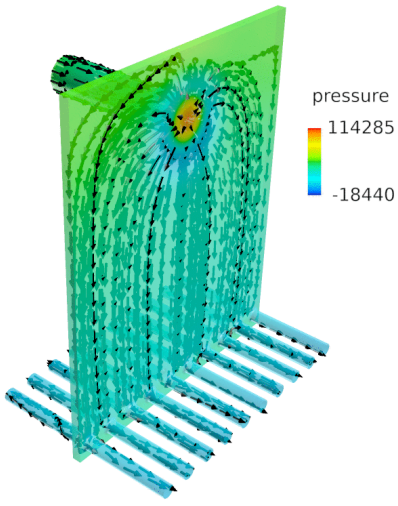


Figure 4 – “Impinging Nozzle” model (image/courtesy of Altair)

「Impinging Nozzle」(衝突ノズル) はかなり大きなモデル (780 万ノード、770 万要素) であり、ナビエ・ストークス方程式に基づくソルバーと Spalart– Allmaras 乱流モデルを用いて、ノズルを通る定常水流をシミュレートする。

本ベンチマークのシミュレーションは 200 タイムステップに制限されている。

2. アーキテクチャ設計

本アーキテクチャは AWS ParallelCluster の標準デプロイメントに基づいている。

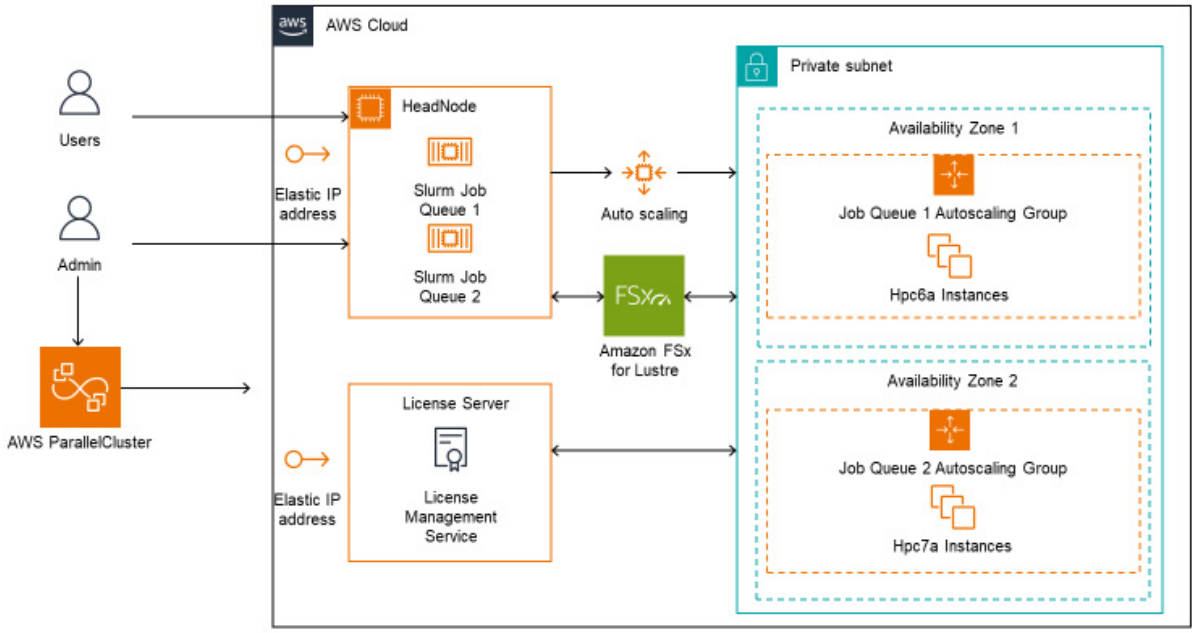


図5 – AWS アーキテクチャ図

インストールされたヘッドノードが、2 つのジョブキューに、一元化されたユーザー認証、ジョブスケジューリングサービス、およびオートスケーリング統合を提供する。各ジョブキューは、分離されたアベイラビリティゾーン内の特定インスタンスタイプにそれぞれ紐付けられている。

Amazon FSx for Lustre は、アプリケーション、ユーザーデータ、および出力データを格納する共有ファイルシステムを提供し、計算ジョブの高性能スクラッチファイルシステムとしても機能する。

全システムは Red Hat Enterprise Linux 8.8 を使用する。

HPC インスタンスでは、EFA ネットワーキング (https://aws.amazon.com/hpc/efa/) が有効化されている。

外部ライセンスサーバーが、シミュレーション実行に必要なネットワークライセンスをソフトウェアに提供する。

The “Impinging Nozzle” is a fairly substantial model (7.8 million nodes and 7.7 million elements) that simulates a steady water flow through a nozzle using a Navier–Stokes-based solver, and a Spalart–Allmaras turbulence model.

The simulation for this benchmark has been limited to 200 time-steps.

2. Architecture design

The architecture is based on an AWS ParallelCluster standard deployment.

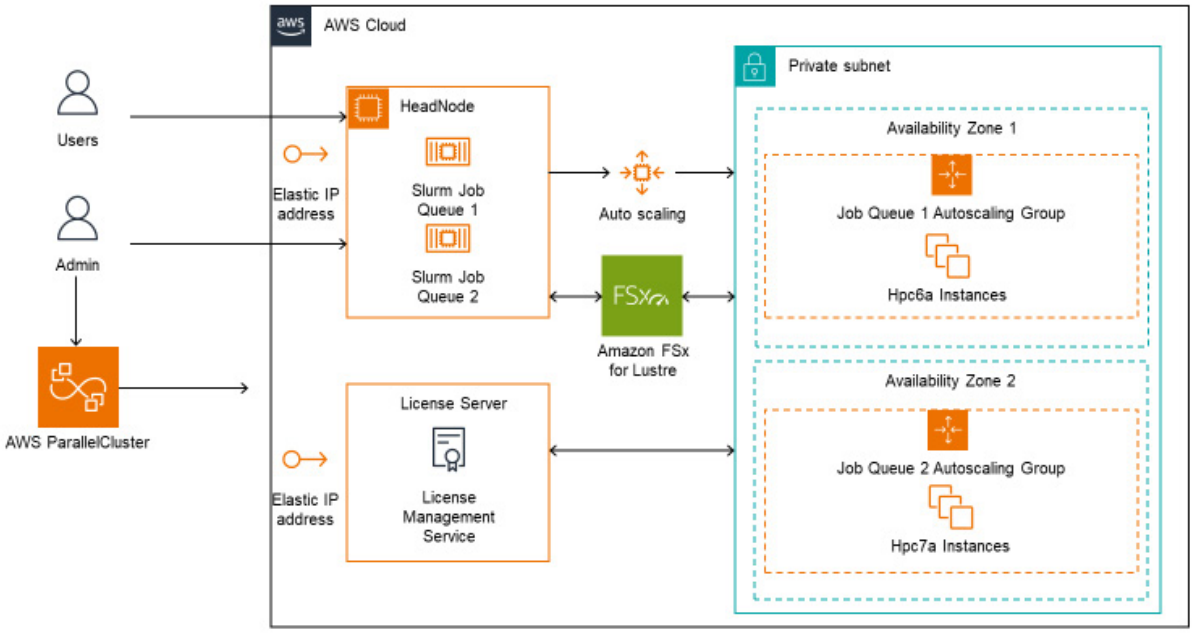


Figure 5 – AWS architecture diagram

A HeadNode is installed, providing centralized user authentication, job scheduling services, and auto-scaling integration for two job queues. Each job queue is associated with a specific instance type in a separated Availability Zone.

An Amazon FSx for Lustre deployment provides the shared filesystem that will contain the applications, the user input, and output data while also acting as a high-performance scratch filesystem for compute jobs.

All systems are using RedHat Enterprise Linux 8.8.

On HPC instances, EFA networking (https://aws.amazon.com/hpc/efa/) is enabled.

An external License Server provides the software with the networked licenses that are required to run the simulations.

3. インストール

3.1 AWS ParallelCluster

アーキテクチャのデプロイメントには、AWS ParallelCluster を使用し、インスタンス仕様に適した構成とベース AMI を選択して実施した。インスタンスが計算ジョブに使用されていない間のコスト発生を避けるために、コンピューティングインスタンスのタイムアウトを 10 分に設定した。

3.1.1 スケジューラ

Slurm スケジューラも AWS ParallelCluster によってデプロイされ、適切なパーティション（ジョブキュー）を構成し、それぞれに特定のインスタンスタイプの紐付けに役立った。全てのオートスケーリング操作は Slurm によって直接制御され、各ジョブのリソース要求に基づいてコンピューティングインスタンスのデプロイが管理される。

3.1.2 ストレージ

FSx for Lustre ストレージも AWS ParallelCluster によってデプロイされ、ヘッドノードと各コンピューティングインスタンスが同じデータを最高性能でアクセスできるようにした。ユーザーホームディレクトリ用に EBS と NFS の使用も検討したが、本番環境ではないため不要と判断し、非必須機能の削減による総コスト抑制を図った。

3.2 Altair ソフトウェアとライセンスサービス

Altair ライセンスソフトウェアは、HPC アーキテクチャ外部の別マシンにインストールされた。ライセンスサーバーも AWS 上にあり、固定のパブリック IP アドレスがあり、厳格なセキュリティアクセスルールにより、HPC インフラストラクチャ内のマシンからのみアクセス可能となっている。

全ての Altair CAE ソフトウェアは、FSx for Lustre ファイルシステム上に直接インストールされ、全てのコンピューティングインスタンスから一貫したファイルシステムパスでアクセス可能となっている。

3.3 ユーザーデータリポジトリ

ユーザーデータも FSx for Lustre ファイルシステム内に保存され、各ソルバーごとに「 models 」と「 runs 」で整理されている。

4. ベンチマーク実行

各ジョブディレクトリは、異なる設定でシミュレーションを実行するために必要な全データで慎重に準備された。シミュレーション結果は各ジョブのディレクトリ内で出力された。

4.1 ベースライン設定

パフォーマンスとスケーラビリティのベースラインを設定するため、本分析の全インスタンスタイプに対して、各シミュレーションを単一インスタンス上で実行した。

3. Installation

3.1 AWS ParallelCluster

The deployment of the architecture was performed using AWS ParallelCluster and choosing the appropriate dimensions and base AMI for the instances and instance specifications. The compute instance timeout was set to 10 minutes to avoid incurring costs while the instances are not actively used for compute jobs.

3.1.1 Scheduler

The Slurm scheduler was also deployed using AWS ParallelCluster, which helped configuring the proper partitions (aka job queues) with the association of a specific instance type to each partition. All auto-scaling operations are directly controlled by Slurm, which takes care of deploying compute instances based on each job's resource requests.

3.1.2 Storage

FSx for Lustre storage was also deployed via AWS ParallelCluster, to ensure that the HeadNode and each compute instance can access the same data with the highest possible performance. We considered the possibility of using EBS with NFS for the users' home directories, but we deemed it unnecessary as this is not a production-ready environment, and we wanted to contain overall costs by cutting what we considered to be non-essential features.

3.2 Altair software and licensing services

Altair licensing software was installed on a separate machine, outside of the HPC architecture. The license server is also on AWS, has a fixed public IP, and strict security access rules to be reachable only by the machines within the HPC infrastructure.

All Altair CAE software was installed directly on the FSx for Lustre filesystem, to be accessible from all the compute instances with a consistent filesystem path.

3.3 User Data Repository

User data was also stored in the FSx for Lustre filesystem, organized in "models" and "runs" for each solver.

4. Benchmark execution

The different job directories were carefully prepared with all the required data to run the simulations in different configurations. Simulation results were produced within each job's directory.

4.1 Baselining

To provide a baseline for performance and scalability, we ran each simulation on a single instance for each of the instance types covered in this study.

4.2 MPI チューニング

2 つのインスタンスタイプはそれぞれ異なる世代のプロセッサを搭載し、コア数とネットワーク構成が異なる。このため、各インスタンスタイプの特性に最適化するため、MPI ライブラリの微調整が必要であった。

両インスタンスタイプで使用した MPI ライブラリは、Intel® MPI Version 2021 Update 9 であった。Altair ソフトウェアと AWS ParallelCluster システムイメージには、同じバージョンで同一ライブラリが含まれており、後者では EFA サポート強化のため AWS がカスタマイズした外部 libfabric（version 1.17）も提供されている。

ベンチマークで使用した全ソフトウェアの MPI 設定は同一であった。

4.2.1 Amazon EC2 Hpc6a MPI 設定

各ジョブの実行時に以下の環境変数が設定された：

1. export FI_EFA_FORK_SAFE=1
2. export I_MPI_OFI_LIBRARY_INTERNAL=0
3. export I_MPI_FABRICS=shm:ofi
4. export I_MPI_OFI_PROVIDER=efa
5. export I_MPI_DEBUG=5

これは全ベンチマークで使用したベースライン設定であり、以下の目的を有する：

- EFA ネットワーク上での信頼性の高いプロセス起動を実現 (FI_EFA_FORK_SAFE)
- AWS 提供の libfabric を使用 (I_MPI_OFI_LIBRARY_INTERNAL)
- ノード内およびノード間通信における最適パフォーマンスのための適切なファブリックを選択 (I_MPI_FABRICS)
- AWS Elastic Fabric Adapter のための必要なファブリックプロバイダを選択 (I_MPI_OFI_PROVIDER)
- ファブリック設定を詳細に分析するのに充分な詳細を持つデバッグモードを有効化 (I_MPI_DEBUG)

4.2.2 Amazon EC2 Hpc7a MPI 設定

各ジョブの実行時に以下の環境変数が設定された：

1. export FI_EFA_FORK_SAFE=1
2. export I_MPI_OFI_LIBRARY_INTERNAL=0
3. export I_MPI_FABRICS=shm:ofi
4. export I_MPI_OFI_PROVIDER=efa
5. export I_MPI_DEBUG=5
6. export FI_EFA_SHM_AV_SIZE=256
7. export I_MPI_MULTIRAIL=1

Hpc6a からの違いは最後の 2 つの環境変数：

- FI_EFA_SHM_AV_SIZE は、Hpc7a が単一ノードで持つプロセッサ数の増加により、ParallelCluster 3.7.0 以前のバージョンで必要である。適切な設定なしではシミュレーションが MPI プロセスの生成時にクラッシュしていた。
- I_MPI_MULTIRAIL は、Hpc7a ノード上の 2 つの EFA インターフェースを最大限に活用し、シミュレーションの MPI プロセスが利用可能なネットワーク帯域幅全体を増加させるために必要である。

4.2 MPI tuning

The two instance types have processors from different generations with a different number of cores and a different network configuration. This required fine-tuning the MPI libraries to best suit the characteristics of each instance type.

The MPI library used for both instance types was Intel® MPI, version 2021 Update 9. In fact, the same library at the same version was included in the Altair software and in the AWS ParallelCluster system image, with the latter also providing an external libfabric (version 1.17) customized by AWS for better EFA support.

The MPI configurations were the same for all software used in our benchmarks.

4.2.1 Amazon EC2 Hpc6a MPI settings

The following environment variables were set at each job's run time:

1. export FI_EFA_FORK_SAFE=1
2. export I_MPI_OFI_LIBRARY_INTERNAL=0
3. export I_MPI_FABRICS=shm:ofi
4. export I_MPI_OFI_PROVIDER=efa
5. export I_MPI_DEBUG=5

This is the baseline configuration that we used for all tests to:

- provide reliable process spawn on EFA network (FI_EFA_FORK_SAFE)
- make use of the AWS-provided libfabric (I_MPI_OFI_LIBRARY_INTERNAL)
- select the correct fabrics for optimal performance within the nodes and for node-to-node communications (I_MPI_FABRICS)
- select the required fabric provider for AWS Elastic Fabric Adapter (I_MPI_OFI_PROVIDER)
- enable debug mode with enough details to analyze the fabric configuration in detail (I_MPI_DEBUG)

4.2.2 Amazon EC2 Hpc7a MPI settings

The following environment variables were set at each job's run time:

1. export FI_EFA_FORK_SAFE=1
2. export I_MPI_OFI_LIBRARY_INTERNAL=0
3. export I_MPI_FABRICS=shm:ofi
4. export I_MPI_OFI_PROVIDER=efa
5. export I_MPI_DEBUG=5
6. export FI_EFA_SHM_AV_SIZE=256
7. export I_MPI_MULTIRAIL=1

The key difference from Hpc6a are the last two environment variables:

- FI_EFA_SHM_AV_SIZE is required with ParallelCluster versions older than 3.7.0 because of the increased number of processors that Hpc7a has on a single node. Without the proper setting, the simulations were crashing at MPI process spawn.
- I_MPI_MULTIRAIL is required to fully take advantage of the two EFA interfaces on the Hpc7a nodes and increase the overall network bandwidth available to the MPI processes of the simulations.

4.3 スケーラビリティ

スケーラビリティを検証するため、異なるノード構成で同一シミュレーションを実施した。ベースラインから開始し、コア数を元の 2 倍、4 倍に増加させた。ただし、Hpc6a インスタンスは Hpc7a のノードあたりコア数の半分であるため、後者との適切なコア対コア比較を行うため、前者で 8 倍まで増加させた。これにより、シミュレーションあたりの最大コア数は 768 となった。

4.4 パフォーマンス結果

注記：環境上のインスタンス数上限およびライセンスコア数の制約により、Hpc7a インスタンスの低コアサイズにおけるベンチマークを全規模実行できなかったため、一部のデータポイントが欠落している。

4.4.1 Altair® Radioss®

各 Radioss ジョブの実行時間には、出力ファイルからエンジン・ソルバー自身が報告した時間を使用した。これは、ジョブ開始時に単一ノード上で実行されるスタータープロセスが、シミュレーションを実行する各エンジンプロセス用の入力ファイルを準備するためである。シミュレーション実行に使用するコア数が増加すると、スタータープロセスによる入力ファイル準備時間は線形に増加する。最終的に、異なる実行間の比較をより正確かつ公平に行うため、ベンチマーク経過時間からスタータープロセスの時間を除外することにした。

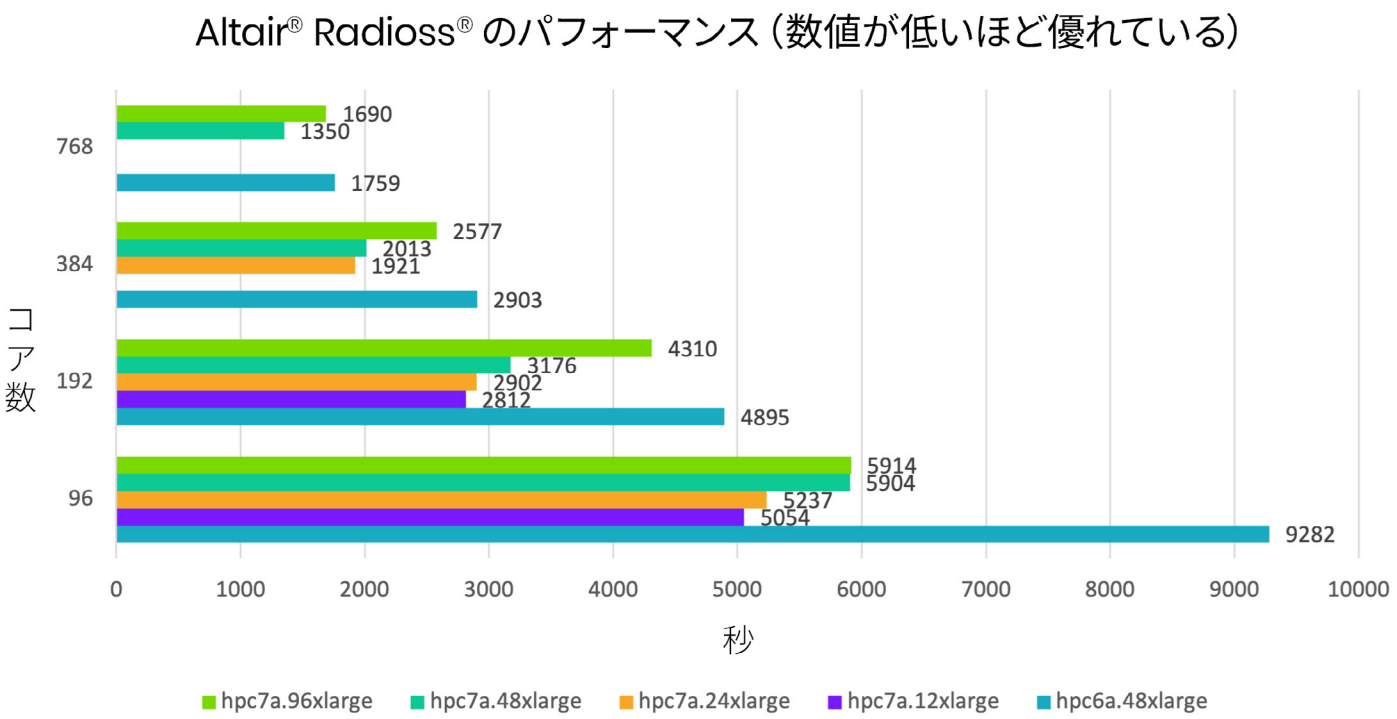


図6 – Hpc6a および Hpc7a 上の Altair® Radioss® のパフォーマンス比較図

4.3 Scalability

To test scalability, the same simulations were run on different node configurations: Starting from the baseline, the number of cores was increased to 2 times and 4 times the original amount. However, since Hpc6a instances have half the cores per node than Hpc7a, we went up to 8 times with the former to have a proper core-per-core comparison with the latter. This brought our maximum core number per simulation to 768.

4.4 Performance results

Note: We are missing some data points for low-core sizes of Hpc7a instances, as we could not run the full scale of benchmarks due to limitations to the maximum number of instances and licensed cores in our environment.

4.4.1 Altair® Radioss®

The execution time of each Radioss job was taken from the output files, using the reported time from the "engine" solver itself. This is because the "starter" process runs at the beginning of the job on a single node, preparing the input files for each single "engine" process that will run the simulation. The time for the "starter" process to prepare the input files increases linearly with the number of cores the simulation will run. In the end, we chose not to include "starter" timings in the benchmark elapsed time to provide a more precise, apples-to-apples comparison between different runs.

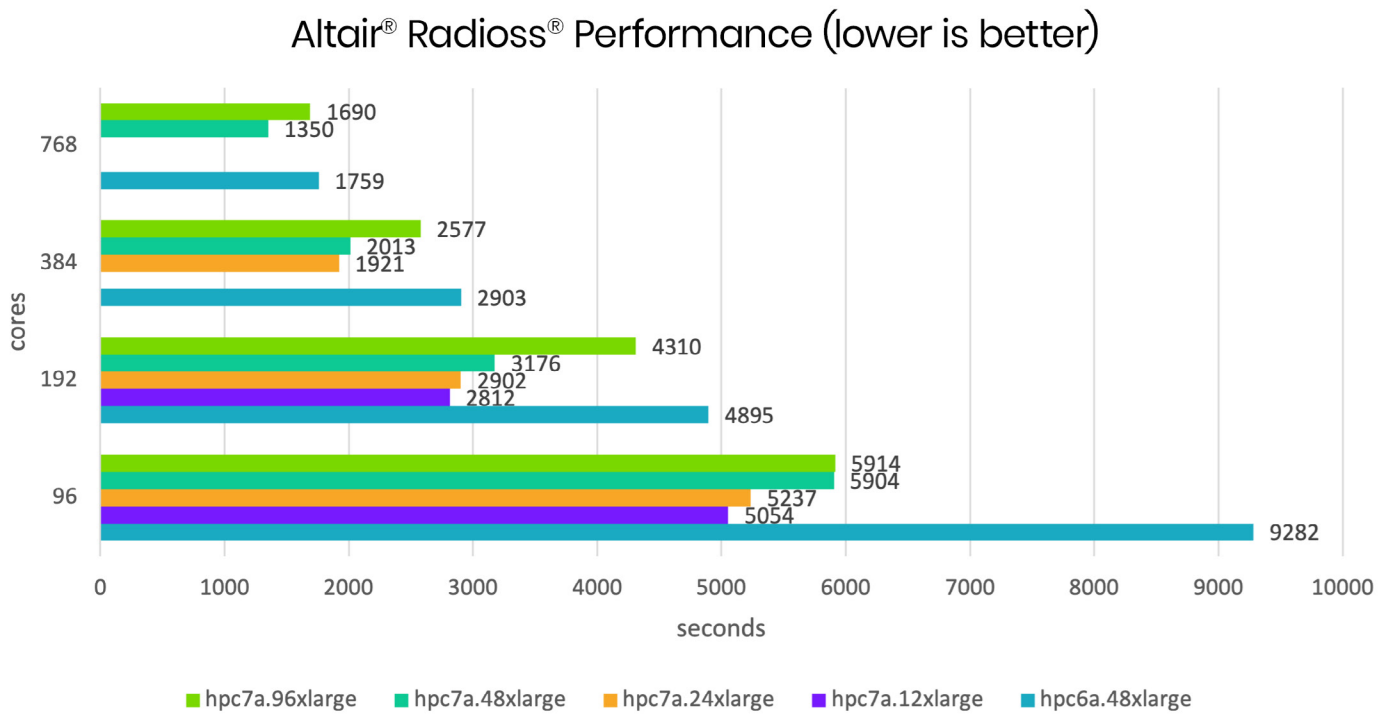


Figure 6 – Performance chart for Altair® Radioss® on Hpc6a and Hpc7a

hpc6a.48xlarge	96 コア	192 コア	384 コア	768 コア
ジョブ経過時間 (秒)	9282	4895	2903	1759
スケーラビリティ (96 コアを基準として)	100%	94.8%	79.9%	66.0%

図7 – hpc6a.48xlarge 上の Altair® Radioss® のスケーラビリティ

hpc7a.96xlarge	96 コア	192 コア	384 コア	768 コア
ジョブ経過時間 (秒)	5914	4310	2577	1690
スケーラビリティ (96 コアを基準として)	100%	68.6%	57.4%	43.7%

図8 – hpc7a.96xlarge 上の Altair® Radioss® のスケーラビリティ

Hpc6a 上で、Altair® Radioss® はコア数を 96 から 768 に増加させたときに非常に良いスケーラビリティ (66%) を示す。

Hpc7a では、インスタンスのサイズによって結果が異なる。

デフォルトサイズである hpc7a.96xlarge では、コア数を 96 から 768 に増加させた際のスケーラビリティが 43.7% とやや低くなる。

192 コアのデータポイントのみを考慮すると、複数の hpc7a.12xlarge ノードを使用した場合、単一の hpc7a.96xlarge インスタンスを使用する場合よりもジョブ経過時間が短縮される (2812 秒対 4310 秒)。

384 コアのデータポイントでも同様の傾向が見られ、hpc7a.24xlarge サイズを使用すると、hpc7a.96xlarge インスタンスの 2577 秒に対し、経過時間を 1921 秒に短縮できる。

一般的に、コア数の少ない Hpc7a インスタンスでのパフォーマンスが向上していることが確認できる。これは、コアあたりのメモリ量とプロセスあたりのメモリ帯域幅の増加がシミュレーション性能を向上させ、EFA ネットワーク機能を活用することでメモリボトルネックを克服している可能性を示唆している。

まとめ:

- Hpc7a はあらゆる状況で Hpc6a に対して明確な性能優位性を持つ。
- Hpc6a のスケーラビリティは Hpc7a の基本インスタンスサイズに対して効率的だが、Hpc7a の低コアサイズを活用することで、同コア数環境下では Hpc7a が明らかに Hpc6a を上回る。
- ソフトウェアライセンスが総コア数に基づく場合、Hpc7a の低コアのインスタンスサイズは、総コア使用量を維持しながら性能を大幅に向上させる可能性を持つ。

hpc6a.48xlarge	96 cores	192 cores	384 cores	768 cores
Job Elapsed Time (seconds)	9282	4895	2903	1759
Scalability (using 96 cores as the reference)	100%	94.8%	79.9%	66.0%

Figure 7 – Scalability for Altair® Radioss® on hpc6a.48xlarge

hpc7a.96xlarge	96 cores	192 cores	384 cores	768 cores
Job Elapsed Time (seconds)	5914	4310	2577	1690
Scalability (using 96 cores as the reference)	100%	68.6%	57.4%	43.7%

Figure 8 – Scalability for Altair® Radioss® on hpc7a.96xlarge

On Hpc6a, Altair® Radioss® has very good scalability (66%) when increasing the number of cores from 96 to 768.

On Hpc7a, the picture is a bit different, depending on the actual size of the instance.

The default size, hpc7a.96xlarge, is a bit less efficient with a scalability of 43.7% when increasing the number of cores from 96 to 768.

Considering just the 192 cores data point, using multiple hpc7a.12xlarge nodes gives a shorter job elapsed time than using just one hpc7a.96xlarge instance (2812 seconds vs 4310 seconds).

We can see a similar pattern at the 384 cores data point, where using the hpc7a.24xlarge size can reduce the elapsed time to 1921 seconds compared to 2577 seconds obtained with hpc7a.96xlarge instances.

In general, we can see that the test case performed better on lower-core Hpc7a instances. This means that having more memory per core and more memory bandwidth per process is enhancing the simulation performance, potentially overcoming a memory bottleneck by leveraging EFA networking features.

Summing it up:

- Hpc7a has a definite performance advantage vs. Hpc6a in all situations.
- While Hpc6a scalability is more efficient vs. the base instance shape of Hpc7a, the latter clearly overcomes the former in all iso-core situations by making use of Hpc7a lower-core sizes.
- If your software license is based on the total number of cores, Hpc7a low-core shapes could provide a significant opportunity to enhance performance with the same overall core usage.

4.4.2 Altair® AcuSolve®

AcuSolve の実行時間として考慮されたのは、計算全体の開始から終了までの経過時間（秒単位）である。Radioss と同様に、AcuSolve には各ソルバープロセス向けのシミュレーションデータを分割する準備段階が存在する。ただし、分散計算において AcuSolve は Radioss とは異なる手法を採用するため、この準備フェーズの所要時間はシミュレーション全体の所要時間において無視できる程度（0.1%～0.5%）である。

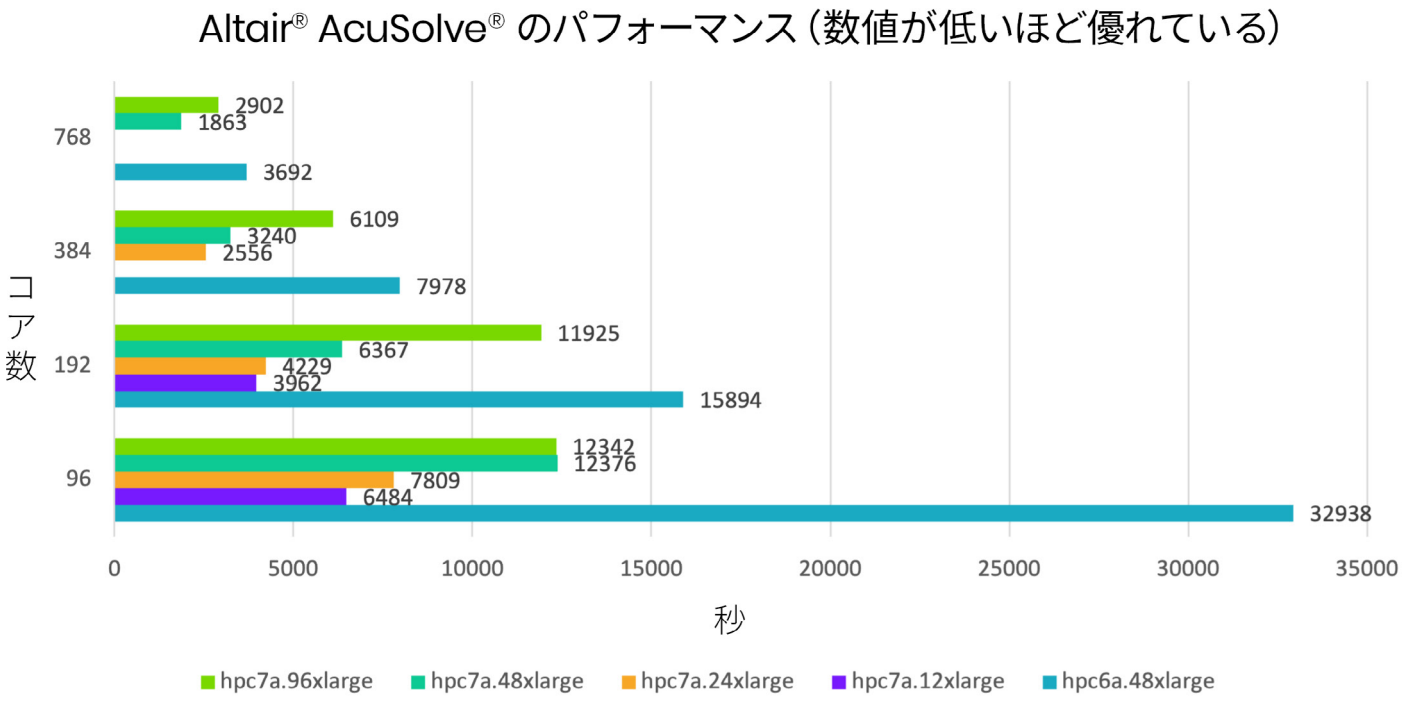


図9 – Hpc6a および Hpc7a 上の Altair® AcuSolve® のパフォーマンス比較図

hpc6a.48xlarge	96 コア	192 コア	384 コア	768 コア
ジョブ経過時間 (秒)	32938	15894	7978	3692
スケーラビリティ (96 コアを基準として)	100%	103.7%	103.3%	111.6%

図10 – hpc6a.48xlarge 上の Altair® AcuSolve® のスケーラビリティ

hpc7a.96xlarge	96 コア	192 コア	384 コア	768 コア
ジョブ経過時間 (秒)	12342	11925	6109	2902
スケーラビリティ (96 コアを基準として)	100%	51.7%	50.5%	53.2%

図11 – hpc7a.96xlarge 上の Altair® AcuSolve® のスケーラビリティ

hpc7a.48xlarge	96 コア	192 コア	384 コア	768 コア
ジョブ経過時間 (秒)	12376	6367	3240	1863
スケーラビリティ (96 コアを基準として)	100%	97.1%	95.5%	83.0%

図12 – hpc7a.48xlarge 上の Altair® AcuSolve® のスケーラビリティ

4.4.2 Altair® AcuSolve®

The execution time taken into consideration for AcuSolve was the overall elapsed time, in seconds, between the start and the end of the whole computation. Like Radioss, AcuSolve has a preparation phase to subdivide the simulation data for each solver process. However, since AcuSolve automatically uses a different approach than Radioss for distributed computations, the duration of this preparation phase is negligible in the context of the overall duration of the simulation (0.1% to 0.5%).

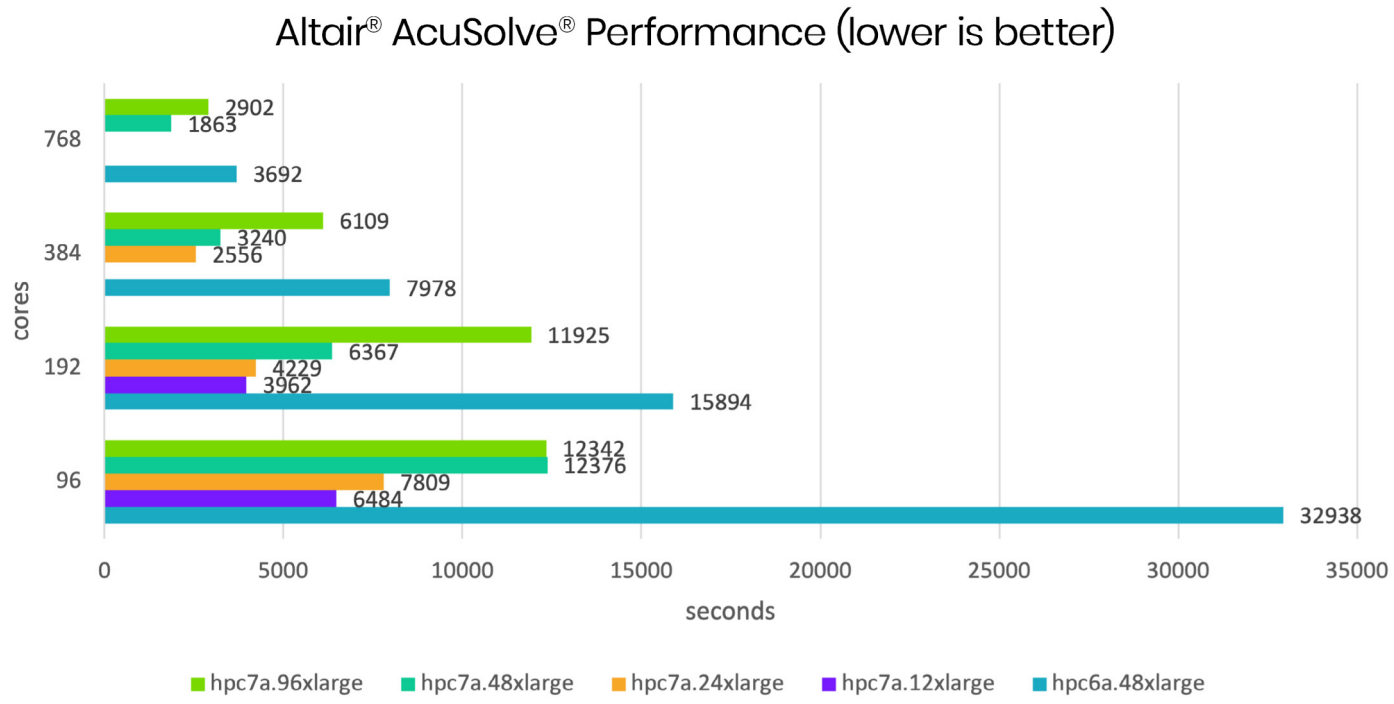


Figure 9 – Performance chart for Altair® AcuSolve® on Hpc6a and Hpc7a

hpc6a.48xlarge	96 cores	192 cores	384 cores	768 cores
Job Elapsed Time (seconds)	32938	15894	7978	3692
Scalability (using 96 cores as the reference)	100%	103.7%	103.3%	111.6%

Figure 10 – Scalability for Altair® AcuSolve® on hpc6a.48xlarge

hpc7a.96xlarge	96 cores	192 cores	384 cores	768 cores
Job Elapsed Time (seconds)	12342	11925	6109	2902
Scalability (using 96 cores as the reference)	100%	51.7%	50.5%	53.2%

Figure 11 – Scalability for Altair® AcuSolve® on hpc7a.96xlarge

hpc7a.48xlarge	96 cores	192 cores	384 cores	768 cores
Job Elapsed Time (seconds)	12376	6367	3240	1863
Scalability (using 96 cores as the reference)	100%	97.1%	95.5%	83.0%

Figure 12 – Scalability for Altair® AcuSolve® on hpc7a.48xlarge

Hpc6a では、Altair® AcuSolve® はコア数を 96 から 768 に増加させた場合において優れたスケーラビリティ（111%）を示す。

Hpc7a のデフォルトサイズである hpc7a.96xlarge は、Hpc6a ノードと比較して非常に良好なパフォーマンスを示す。96 コアの場合、経過時間は 32,938 秒から 12,342 秒に短縮される。ノード数を増やすとスケーラビリティは約 50% だが安定しており、ノード追加数に比例して経過時間が減少する。

低コア Hpc7a インスタンスがより良いパフォーマンスを示した。hpc7a.48xlarge ノードではスケーラビリティが大幅に向上した（768 コアで 83.0%）。コア数が少ないインスタンスを使用する方が一般的に優れており、ジョブ経過時間が若干短縮する。

まとめ：

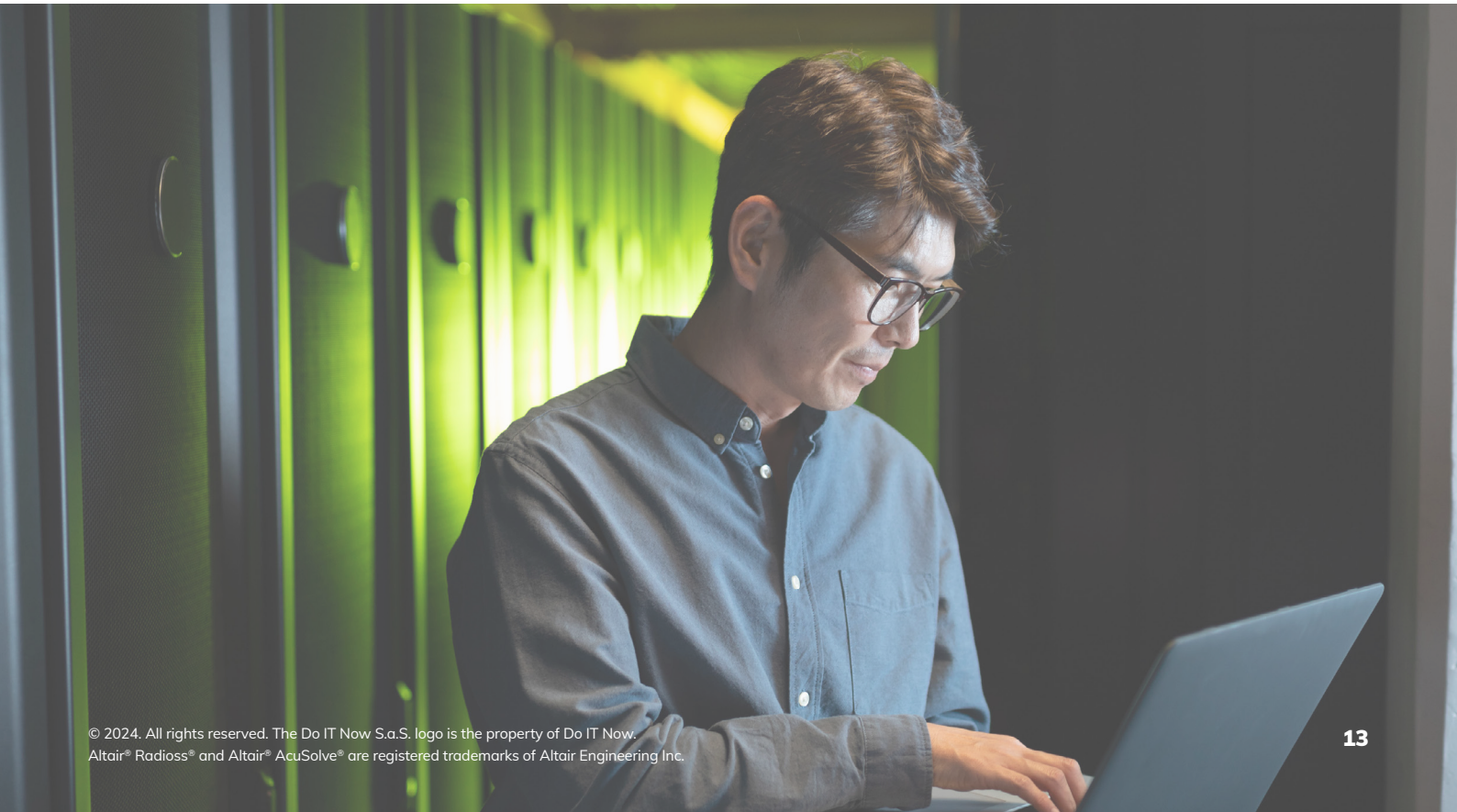
- 全体的なパフォーマンスは Hpc7a が圧倒的に優れており、Hpc6a と比較して約 3 倍の性能向上となった。
- Hpc7a ではコア数が少ないサイズほど良いパフォーマンスが得られ、hpc7a.12xlarge が最も効率的なソリューションを提供する。
- ソフトウェアライセンスが総コア数に基づく場合、Hpc7a の低コアのインスタンスサイズは、総コア使用量を維持しながら性能を大幅に向上させる可能性を持つ。

4.5 ジョブ価格結果

各ジョブ価格を計算し適切に比較するために、以下の式を使用した：

JobPrice = (JobInstanceNum * HourlyInstancePrice * JobExecutionTimeInSecs / 3600)

なお、ストレージは考慮対象外とした。すべてのインスタンスが同一の FSx for Lustre ファイルシステムをストレージとして使用しているためである。各インスタンスタイプの時間単価は、2023 年 11 月 6 日時点の公開データから取得している。



On Hpc6a, Altair® AcuSolve® has excellent scalability (111%) with the test case when increasing the number of cores from 96 to 768.

The default size of Hpc7a, hpc7a.96xlarge, shows a very good performance compared to Hpc6a nodes. The elapsed time goes from 32,938 seconds to 12,342 seconds for 96 cores. When using more nodes, the scalability is around 50% but stable: adding more nodes will decrease the elapsed time in proportion to the number of nodes added.

The test case performed better on lower-core Hpc7a instances. With hpc7a.48xlarge nodes, scalability is much better (83.0% for 768 cores). Using instances with a lower core count is usually better and will slightly decrease the job elapsed time.

Summing it up:

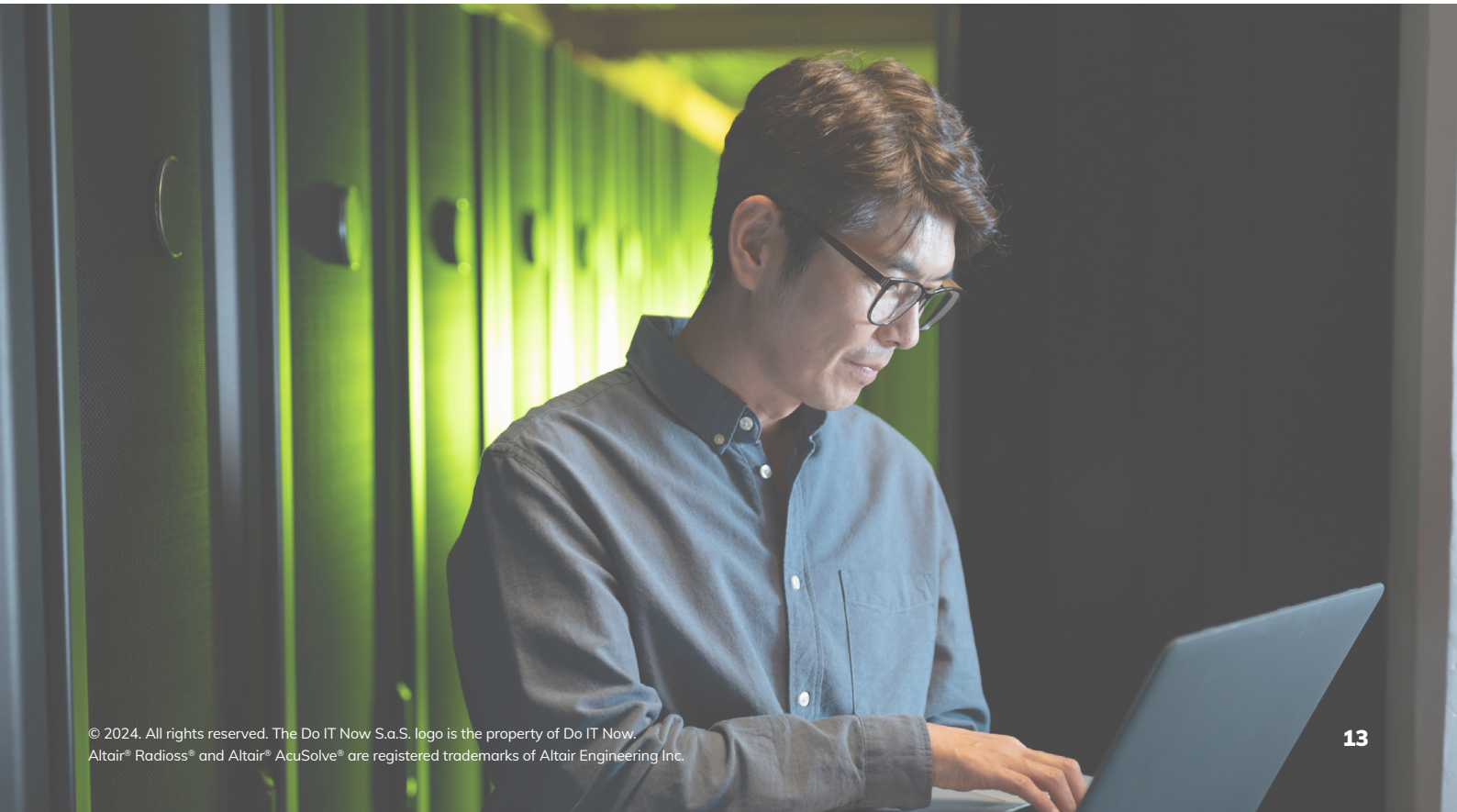
- Overall performance heavily favors Hpc7a with a performance uplift of ~3X compared to Hpc6a.
- The best performance is obtained with lower core count sizes of Hpc7a, with hpc7a.12xlarge providing the most efficient solution.
- In case your software license is based on the total number of cores, Hpc7a low-core shapes could provide a significant opportunity to enhance performance with the same overall core usage.

4.5 Job price results

To calculate each job price and compare it properly, the following formula was used:

JobPrice = (JobInstanceNum * HourlyInstancePrice * JobExecutionTimeInSecs / 3600)

Please note that we took storage out of the equation, since all instances are using the same FSx for Lustre filesystem for their storage needs. The hourly rates for each instance type have been retrieved from the publicly available listings on November 6th, 2023.



4.5.1 Altair® Radioss®

以下の図は、Radioss の各ジョブにおけるコア数に対するジョブ価格の比較を示している：

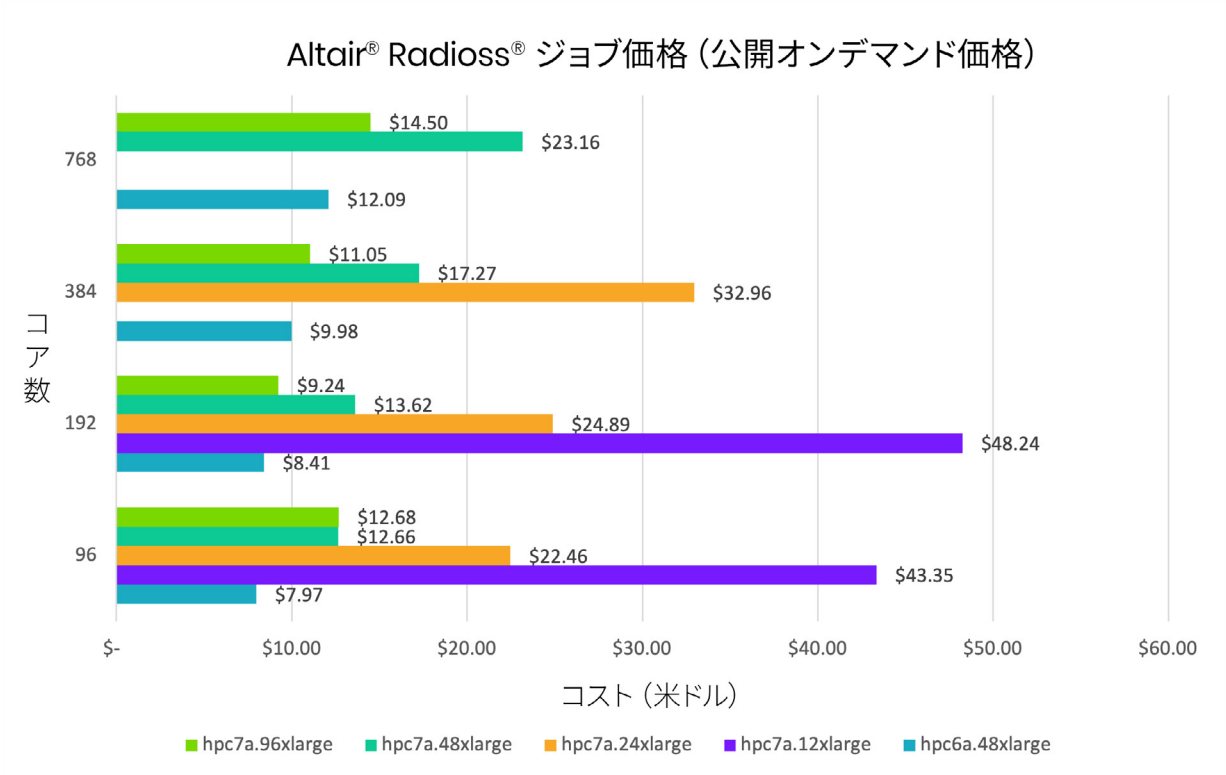


図13 – Hpc6a および Hpc7a 上の Altair® Radioss® のジョブ価格比較図

384 コアのデータポイントを対象としたパフォーマンス数値：

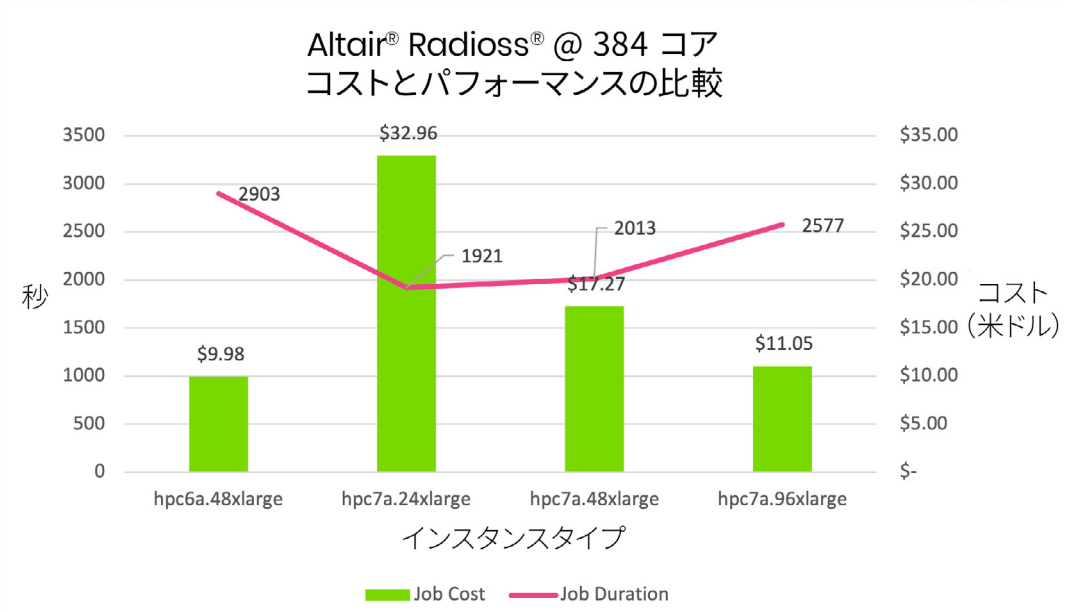


図14 – 様々なインスタンスタイプ/サイズ上の 384 コアの Altair® Radioss® のジョブ価格と性能比較図

最も安価な選択肢は 96 コアの Hpc6a だが、Hpc7a はジョブコストをわずかに増加させながら (シミュレーション秒あたり $0.001 未満)、解決時間の大幅な短縮を実現する。

4.5.1 Altair® Radioss®

The following charts plot the comparative job price against the core count for each job of Radioss:

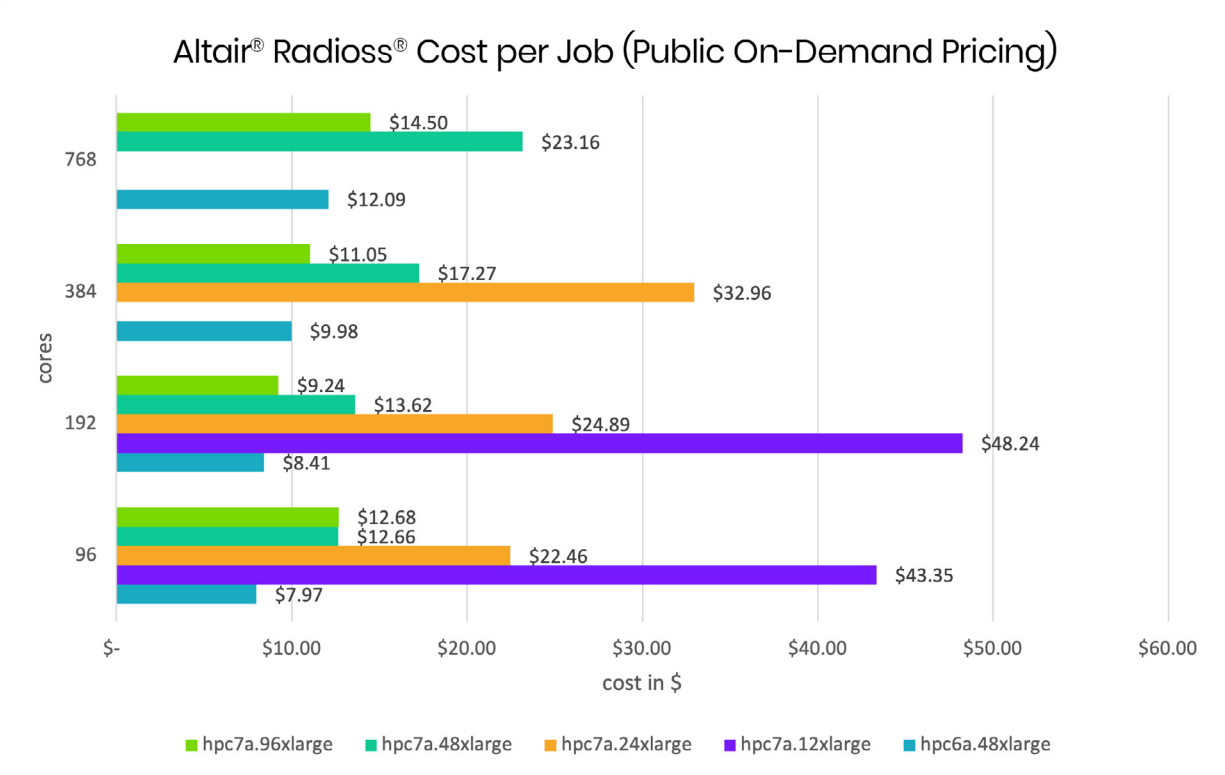


Figure 13 – Cost per job chart for Altair® Radioss® on Hpc6a and Hpc7a

Considering the 384 cores data point and including performance numbers:

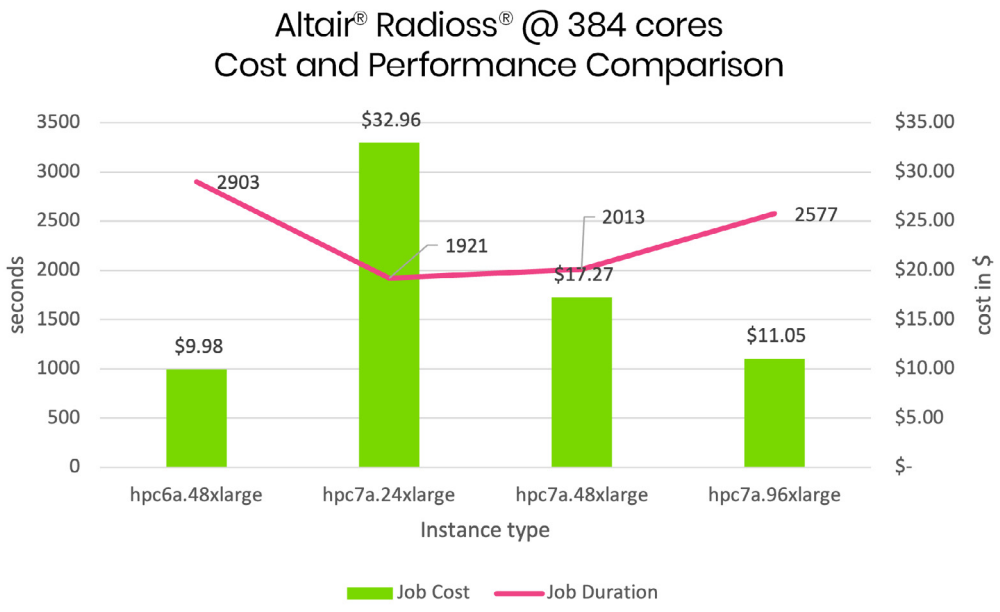


Figure 14 – Cost per job and performance chart for Altair® Radioss® @ 384 cores on various instance types/sizes

While the cheapest option is the Hpc6a with 96 cores, Hpc7a provides a superior reduction in time-to-solution while increasing the job cost by a small amount (less than $0.001 per simulation second).

4.5.2 Altair® AcuSolve®

以下の図は、AcuSolve の各ジョブにおけるコア数に対するジョブ価格の比較を示している：

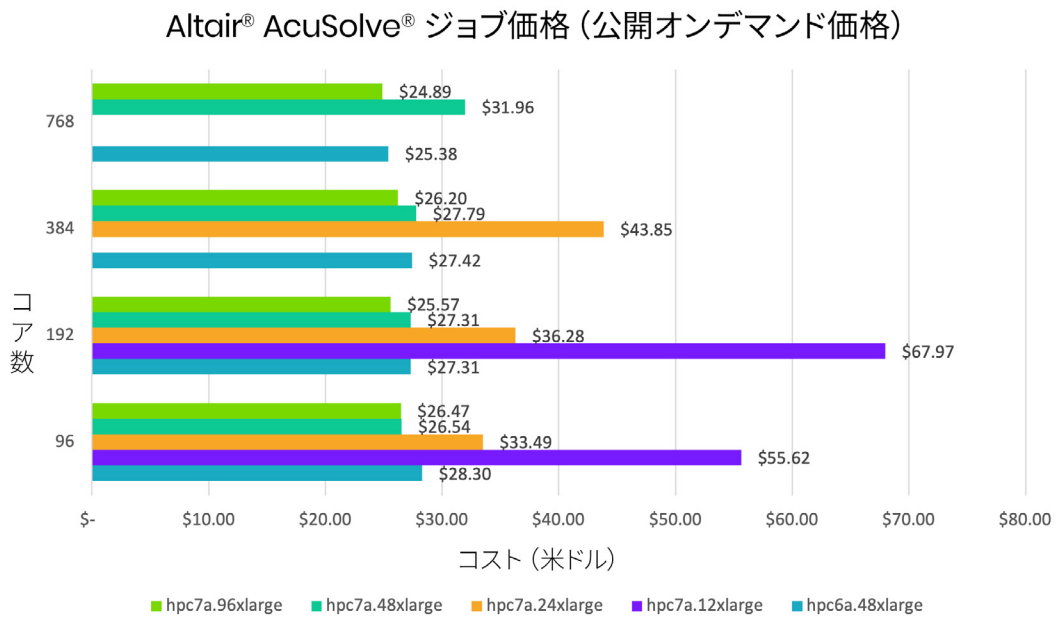


図15 – Hpc6a および Hpc7a 上の Altair® AcuSolve® のジョブ価格比較図

Altair Radioss の図とは大きく異なる状態である：スケーラビリティ効率が限界に近づいたものを除き、ほとんどのジョブ価格は同水準である。

384 コアのデータポイントを対象としたパフォーマンス数値：

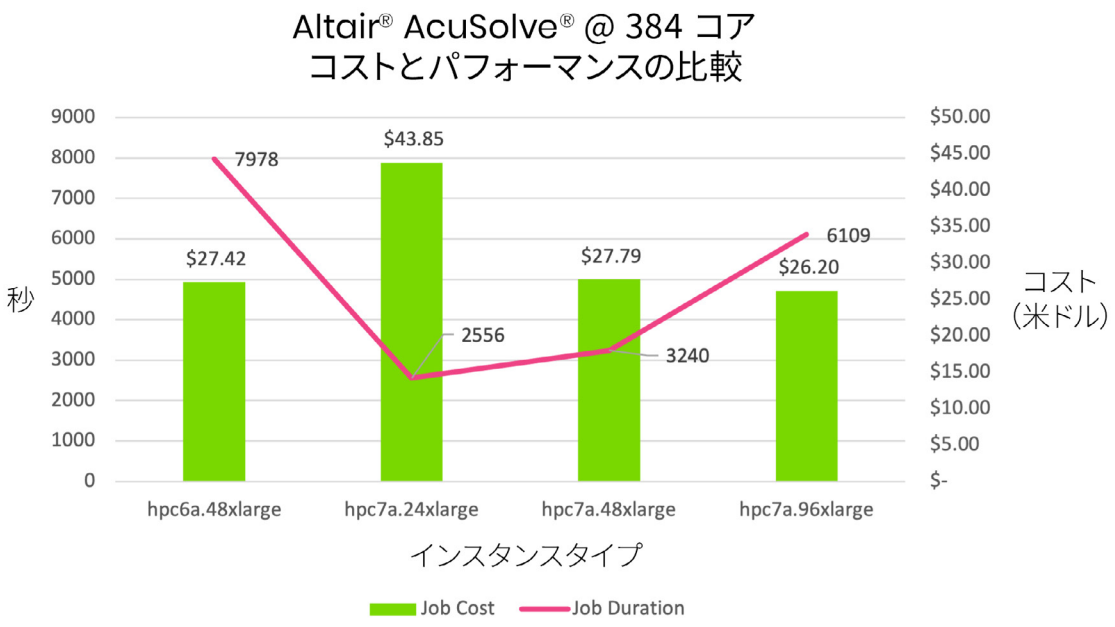


図16 – 様々なインスタンスタイプ/サイズ上の 384 コアの Altair® AcuSolve® のジョブ価格と性能比較図

hpc7a.48xlarge はコスト面で競争力を保ちつつ優れたパフォーマンスを提供しており、一方、最も安価な選択肢である hpc7a.96xlarge では解決までの経過時間がほぼ2倍となっていることが明らかである。

4.5.2 Altair® AcuSolve®

The following charts plot the comparative job price against the core count for each job of AcuSolve:

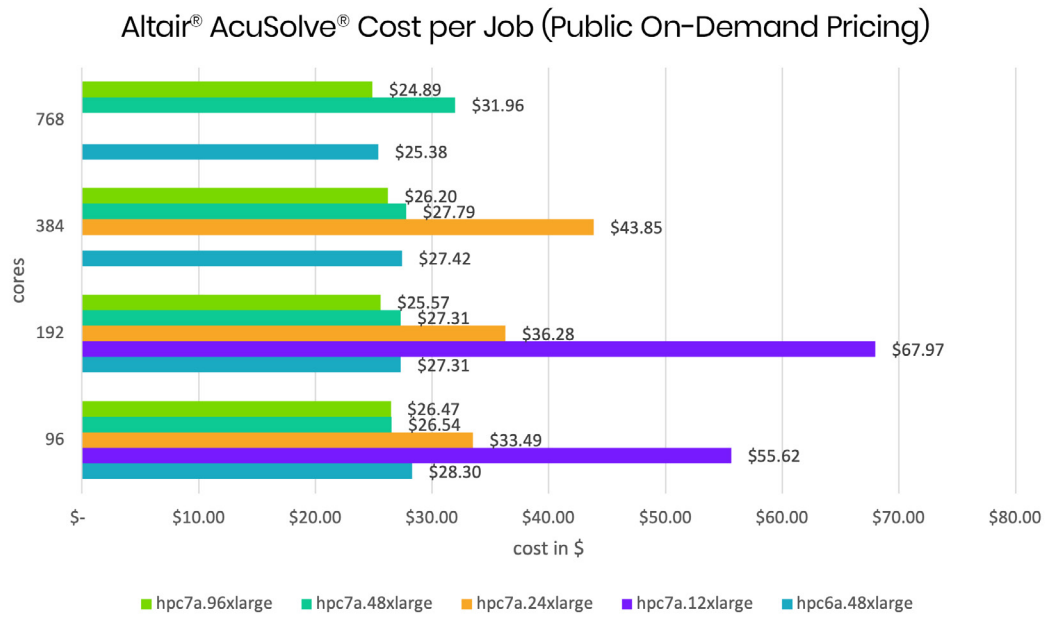


Figure 15 – Cost per job chart for Altair® AcuSolve® on Hpc6a and Hpc7a

The situation is much different than the one we saw on the Altair Radioss chart: Most of the job prices are in the same range, with an exception made for the ones where the scalability efficiency was approaching the limit.

Looking again at the 384 cores data point and including performance data:

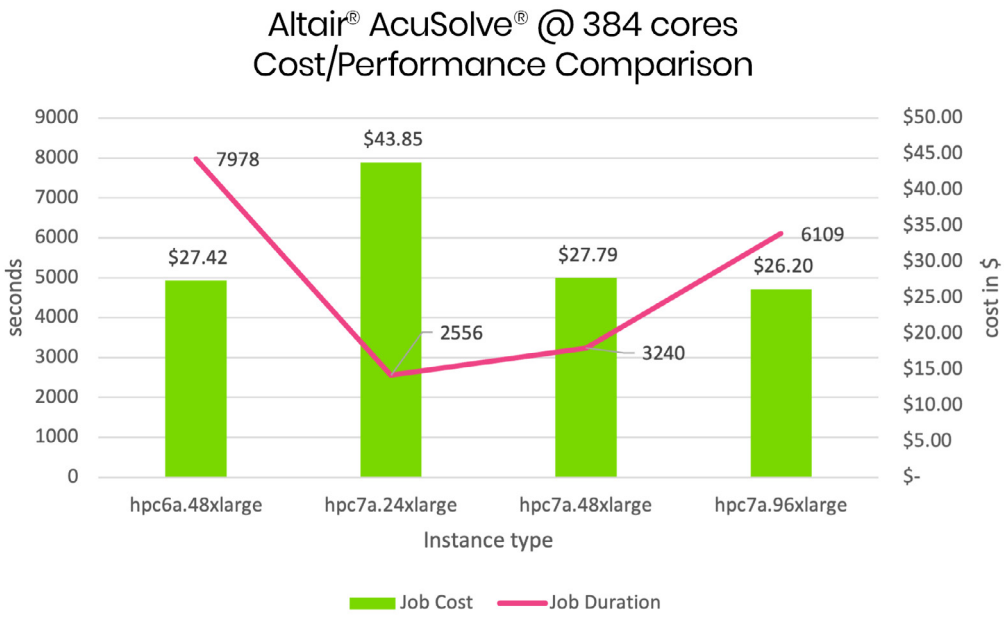


Figure 16 – Cost per job and performance chart for Altair® AcuSolve® @ 384 cores on various instance types/sizes

It's clear that hpc7a.48xlarge provides excellent performance while being aligned in terms of cost, while the cheapest option is provided by hpc7a.96xlarge with almost double the elapsed time-to-solution.

5. 結論

純粋なパフォーマンスの観点では、Hpc7a インスタンスは依然として優れた Hpc6a に対して常に優位性を持っている。インスタンスあたりのコア数増加と高速相互接続の 2 倍以上の帯域幅は、マルチスレッドおよびマルチプロセスのソルバーにとって高いパフォーマンスを実現するための強力なツールである。

価格面では、ジョブ価格の増加率が性能の増加率より小さい点に注意が必要である。つまり、性能向上のためにホストを追加しても、そのインスタンスの時間単価分ではなく、その一部のみがジョブ価格に上乗せされる。この割合はスケーラビリティ曲線に従う。

これは、Altair ソルバーのスケーラビリティ特性、AMD プロセッサの先進的なアーキテクチャ、HPC シリーズインスタンスと EFA ネットワークインターフェースの AWS 最適化によって実現されている。

価格比較では、Altair® Radioss® では旧世代の AWS HPC インスタンスが有利だが、Altair® AcuSolve® では逆である。いずれの場合も、Hpc7a の性能が明らかに優れているにもかかわらず、価格差はそれほど大きくない。

さらに、Hpc7a インスタンスの異なるサイズは、より高いパフォーマンスを実現しつつ、よりコスト効率の良い選択肢を提供できる。

5.1 クレジットと謝辞

本分析をスポンサーした AMD および AWS に深く感謝の意を表する。

ソルバーのライセンスを提供いただいた Altair に対し特別に感謝する。

5. Conclusions

In terms of pure performance, Hpc7a instances always have the edge against the still excellent Hpc6a. The higher cores-per-instance count and the more than double bandwidth for the high-speed interconnect are powerful tools that both multi-thread and multi-process solvers can use to achieve greater performance.

In terms of price, it's important to note that the per-price job scales less than performance, meaning that adding a host to the computation to get better performance will not actually increase the price of the job by that instance's hourly price, but for a fraction. This fraction will follow the scalability curve.

This can be achieved thanks to the scalability properties of the Altair solvers, the advanced architecture of AMD processors, and the AWS optimizations of the HPC-series instances and EFA network interfaces.

The price comparison favors the older generation of the AWS HPC instances for Altair® Radioss®, while the opposite is true for Altair® AcuSolve®. In both cases, prices are not that far from each other even though performance on the Hpc7a is clearly better.

Additionally, the different sizes of Hpc7a instances help in achieving even faster performance and can provide more cost-efficient options.

5.1 Credits and special thanks

Many thanks go to AMD and AWS, for sponsoring this study.

A special thanks to Altair, for providing the licenses for the solvers.

6. 付録

6.1 Slurm 用サブミッションコマンド例

```
sbatch -N<nodeNum> -n<coreNum> --exclusive --mem=0 <jobscript>
```

6.2 ジョブスクリプト

6.2.1 Radioss

```
#!/bin/bash

RADIOSS_DIR="/shared/apps/altair/2023/altair/scripts/"
RADIOSS_INP="TAURUS_A05_FFB50_0000.rad"
NDOMAINS=0

ulimit -s unlimited

cd ${SLURM_SUBMIT_DIR}

NCPUS=${SLURM_NTASKS}

if [ $NDOMAINS -lt 1 ]; then
    NDOMAINS=${NCPUS}
    NTHREADS=1
else
    [ $((NCPUS % NDOMAINS)) -ne 0 ] && exit 1
    NTHREADS=$((NCPUS / NDOMAINS))
fi

NHOSTS=${SLURM_NNODES}

[ $((NDOMAINS % NHOSTS)) -ne 0 ] && exit 1
NDOMAINS_PER_NODE=$((NDOMAINS / NHOSTS))

NODEFILE=nodefile.txt

scontrol show hostnames "$SLURM_JOB_NODELIST" > ./slurm_hosts.${SLURM_JOBID}
for host in $(cat slurm_hosts.${SLURM_JOBID}); do
    printf $host'\n%.0s' {1..${SLURM_NTASKS}} >> ./${NODEFILE}.${SLURM_JOBID}
done
```

6. Appendix

6.1 Example submission command for Slurm

```
sbatch -N<nodeNum> -n<coreNum> --exclusive --mem=0 <jobscript>
```

6.2 Job scripts

6.2.1 Radioss

```
#!/bin/bash

RADIOSS_DIR="/shared/apps/altair/2023/altair/scripts/"
RADIOSS_INP="TAURUS_A05_FFB50_0000.rad"
NDOMAINS=0

ulimit -s unlimited

cd ${SLURM_SUBMIT_DIR}

NCPUS=${SLURM_NTASKS}

if [ $NDOMAINS -lt 1 ]; then
    NDOMAINS=${NCPUS}
    NTHREADS=1
else
    [ $((NCPUS % NDOMAINS)) -ne 0 ] && exit 1
    NTHREADS=$((NCPUS / NDOMAINS))
fi

NHOSTS=${SLURM_NNODES}

[ $((NDOMAINS % NHOSTS)) -ne 0 ] && exit 1
NDOMAINS_PER_NODE=$((NDOMAINS / NHOSTS))

NODEFILE=nodefile.txt

scontrol show hostnames "$SLURM_JOB_NODELIST" > ./slurm_hosts.${SLURM_JOBID}
for host in $(cat slurm_hosts.${SLURM_JOBID}); do
    printf $host'\n%.0s' {1..${SLURM_NTASKS}} >> ./${NODEFILE}.${SLURM_JOBID}
done
```

```
export ALTAIR_LICENSE_PATH="6200@10.0.0.1"

export FI_EFA_FORK_SAFE=1
module load intelmpi
module load libfabric-aws/1.17.1
export I_MPI_OFI_LIBRARY_INTERNAL=0
export I_MPI_FABRICS=shm:ofi
export I_MPI_OFI_PROVIDER=efa
export I_MPI_DEBUG=5

#Uncomment the following on Hpc7a
#export FI_EFA_SHM_AV_SIZE=256
#export I_MPI_MULTIRAIL=1

# Common starter command
${RADIOSS_DIR}/radioss -starter ${RADIOSS_INP} -np ${NDOMAINS} -nt ${NTHREADS} -hostfile
${NODEFILE}.${SLURM_JOBID} -mpi i -mpipath ${I_MPI_ROOT}/bin | tee
starter.${SLURM_JOBID}.out
# Hpc6a engine command
${RADIOSS_DIR}/radioss -engine ${RADIOSS_INP//0000/0001} -np ${NDOMAINS} -nt ${NTHREADS} -
hostfile ${NODEFILE}.${SLURM_JOBID} -mpi i -mpipath ${I_MPI_ROOT}/bin -mpiargs '-genv
I_MPI_FABRICS shm:ofi -genv I_MPI_DEBUG=5 -genv I_MPI_OFI_LIBRARY_INTERNAL=0 -genv
I_MPI_OFI_PROVIDER=efa' | tee engine.${SLURM_JOBID}.out
# Hpc7a engine command
#${RADIOSS_DIR}/radioss -engine ${RADIOSS_INP//0000/0001} -np ${NDOMAINS} -nt ${NTHREADS} -
hostfile ${NODEFILE}.${SLURM_JOBID} -mpi i -mpipath ${I_MPI_ROOT}/bin -mpiargs '-genv
I_MPI_FABRICS=ofi -genv I_MPI_DEBUG=5 -genv I_MPI_OFI_LIBRARY_INTERNAL=0 -genv
I_MPI_OFI_PROVIDER=efa -genv FI_EFA_SHM_AV_SIZE=256 -genv I_MPI_MULTIRAIL=1 -genv
FI_MR_CACHE_MONITOR=memhooks -genv FI_EFA_FORK_SAFE=1' | tee engine.${SLURM_JOBID}.out
```

6.2.2 AcuSolve

```
#!/bin/bash

source /shared/apps/altair/2023/altair/hwcfdsolvers/acusolve/linux64/script/acusim.sh


ulimit -s unlimited

cd ${SLURM_SUBMIT_DIR}
```

```
export ALTAIR_LICENSE_PATH="6200@10.0.0.1"

# Acusolve recognizes the platform automatically and takes care of MPI settings, even
between Hpc6a and Hpc7a
#export FI_EFA_FORK_SAFE=1
#export I_MPI_OFI_LIBRARY_INTERNAL=0
#export I_MPI_FABRICS=shm:ofi
#export I_MPI_OFI_PROVIDER=efa
#export I_MPI_DEBUG=5
#export FI_EFA_SHM_AV_SIZE=256
#export I_MPI_MULTIRAIL=1

acuRun -slurm | tee acuRun.${SLURM_JOBID}.out
```

```
export ALTAIR_LICENSE_PATH="6200@10.0.0.1"

export FI_EFA_FORK_SAFE=1
module load intelmpi
module load libfabric-aws/1.17.1
export I_MPI_OFI_LIBRARY_INTERNAL=0
export I_MPI_FABRICS=shm:ofi
export I_MPI_OFI_PROVIDER=efa
export I_MPI_DEBUG=5

#Uncomment the following on Hpc7a
#export FI_EFA_SHM_AV_SIZE=256
#export I_MPI_MULTIRAIL=1

# Common starter command
${RADIOSS_DIR}/radioss -starter ${RADIOSS_INP} -np ${NDOMAINS} -nt ${NTHREADS} -hostfile
${NODEFILE}.${SLURM_JOBID} -mpi i -mpipath ${I_MPI_ROOT}/bin | tee
starter.${SLURM_JOBID}.out
# Hpc6a engine command
${RADIOSS_DIR}/radioss -engine ${RADIOSS_INP//0000/0001} -np ${NDOMAINS} -nt ${NTHREADS} -
hostfile ${NODEFILE}.${SLURM_JOBID} -mpi i -mpipath ${I_MPI_ROOT}/bin -mpiargs '-genv
I_MPI_FABRICS shm:ofi -genv I_MPI_DEBUG=5 -genv I_MPI_OFI_LIBRARY_INTERNAL=0 -genv
I_MPI_OFI_PROVIDER=efa' | tee engine.${SLURM_JOBID}.out
# Hpc7a engine command
#${RADIOSS_DIR}/radioss -engine ${RADIOSS_INP//0000/0001} -np ${NDOMAINS} -nt ${NTHREADS} -
hostfile ${NODEFILE}.${SLURM_JOBID} -mpi i -mpipath ${I_MPI_ROOT}/bin -mpiargs '-genv
I_MPI_FABRICS=ofi -genv I_MPI_DEBUG=5 -genv I_MPI_OFI_LIBRARY_INTERNAL=0 -genv
I_MPI_OFI_PROVIDER=efa -genv FI_EFA_SHM_AV_SIZE=256 -genv I_MPI_MULTIRAIL=1 -genv
FI_MR_CACHE_MONITOR=memhooks -genv FI_EFA_FORK_SAFE=1' | tee engine.${SLURM_JOBID}.out
```

6.2.2 AcuSolve

```
#!/bin/bash

source /shared/apps/altair/2023/altair/hwcfdsolvers/acusolve/linux64/script/acusim.sh


ulimit -s unlimited

cd ${SLURM_SUBMIT_DIR}
```

```
export ALTAIR_LICENSE_PATH="6200@10.0.0.1"

# Acusolve recognizes the platform automatically and takes care of MPI settings, even
between Hpc6a and Hpc7a
#export FI_EFA_FORK_SAFE=1
#export I_MPI_OFI_LIBRARY_INTERNAL=0
#export I_MPI_FABRICS=shm:ofi
#export I_MPI_OFI_PROVIDER=efa
#export I_MPI_DEBUG=5
#export FI_EFA_SHM_AV_SIZE=256
#export I_MPI_MULTIRAIL=1

acuRun -slurm | tee acuRun.${SLURM_JOBID}.out
```

6.3 ParallelCluster 設定ファイル

6.3.1 eu-north-1 の Amazon EC2 Hpc6a

```
Region: eu-north-1
Image:
  Os: rhel8
HeadNode:
  InstanceType: t3.medium
  Networking:
    SubnetId: subnet-0ffffffffffffffff
  Ssh:
    KeyName: amd-benchmark
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: hpc6a
    ComputeResources:
    - Name: hpc6a48xlarge
      Instances:
      - InstanceType: hpc6a.48xlarge
      MinCount: 0
      MaxCount: 10
      Efa:
        Enabled: true
    Networking:
```

```
        Enabled: true
    Networking:
      PlacementGroup:
        Enabled: true
      SubnetIds:
      - subnet-02222222222222222
    CustomActions:
      OnNodeStart:
        Script: s3://amd-benchmarking/create-users.sh
SharedStorage:
  - MountDir: /shared
    Name: AmdBenchmarkingSharedLustre
    StorageType: FsxLustre
    FsxLustreSettings:
      StorageCapacity: 1200
      DeploymentType: PERSISTENT_2
      PerUnitStorageThroughput: 125
Tags:
  - Key: Project
    Value: AMD
```

6.3 ParallelCluster configuration files

6.3.1 Amazon EC2 Hpc6a in eu-north-1

```
Region: eu-north-1
Image:
  Os: rhel8
HeadNode:
  InstanceType: t3.medium
  Networking:
    SubnetId: subnet-0ffffffffffffffff
  Ssh:
    KeyName: amd-benchmark
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: hpc6a
    ComputeResources:
    - Name: hpc6a48xlarge
      Instances:
      - InstanceType: hpc6a.48xlarge
      MinCount: 0
      MaxCount: 10
      Efa:
        Enabled: true
    Networking:
```

```
        Enabled: true
    Networking:
      PlacementGroup:
        Enabled: true
      SubnetIds:
      - subnet-02222222222222222
    CustomActions:
      OnNodeStart:
        Script: s3://amd-benchmarking/create-users.sh
SharedStorage:
  - MountDir: /shared
    Name: AmdBenchmarkingSharedLustre
    StorageType: FsxLustre
    FsxLustreSettings:
      StorageCapacity: 1200
      DeploymentType: PERSISTENT_2
      PerUnitStorageThroughput: 125
Tags:
  - Key: Project
    Value: AMD
```

6.3.2 eu-west-1 の Amazon EC2 Hpc7a

```
Region: eu-west-1
Image:
  Os: rhel8
HeadNode:
  InstanceType: t3.medium
  Networking:
    SubnetId: subnet-0eeeeeeeeeeeeeeee
  Ssh:
    KeyName: amd-benchmark-2
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: hpc7a
    ComputeResources:
    - Name: hpc7a96xlarge
      Instances:
      - InstanceType: hpc7a.96xlarge
      MinCount: 0
      MaxCount: 10
      Efa:
```

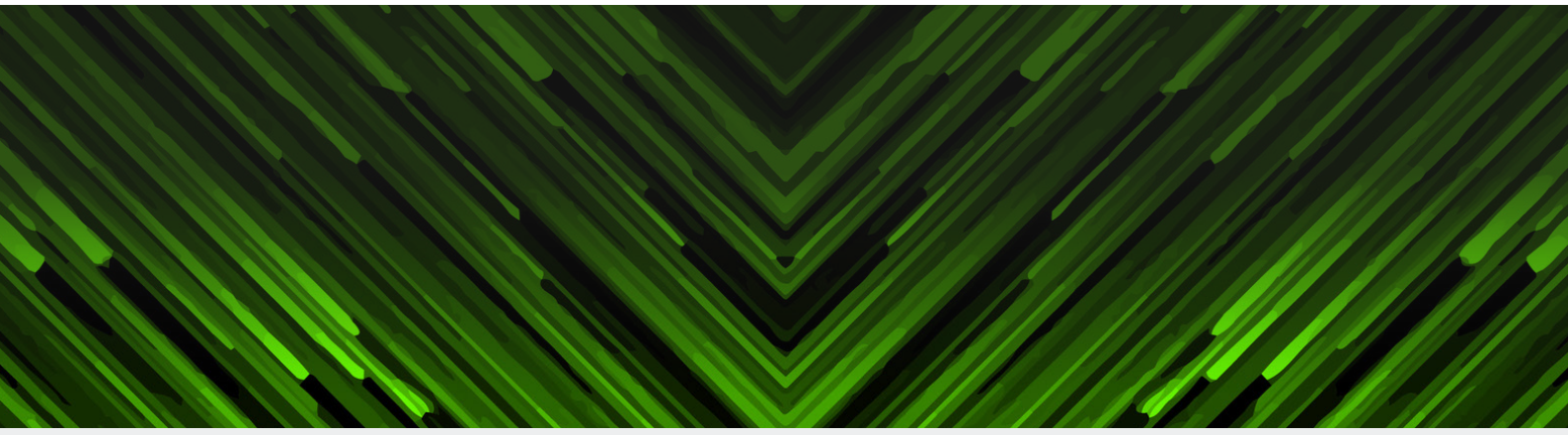
スポンサー: AMD, AWS
バージョン: 8 – 2023 年 11 月 30 日

6.3.2 Amazon EC2 Hpc7a in eu-west-1

```
Region: eu-west-1
Image:
  Os: rhel8
HeadNode:
  InstanceType: t3.medium
  Networking:
    SubnetId: subnet-0eeeeeeeeeeeeeee
  Ssh:
    KeyName: amd-benchmark-2
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: hpc7a
    ComputeResources:
    - Name: hpc7a96xlarge
      Instances:
      - InstanceType: hpc7a.96xlarge
      MinCount: 0
      MaxCount: 10
      Efa:
```

doitnow

AMD



HPC に関するご相談は、是非ご連絡ください

お問い合わせは info@doitnowgroup.com まで

Discuss your HPC needs today

Contact us at info@doitnowgroup.com

www.doitnowgroup.com