\Orchestrating a brighter world NEC

国立大学法人　大阪大学
サイバーメディアセンター 御中

高性能計算・データ分析基盤システム SQUID 利用者説明会

2021年04月26日

日本電気株式会社
第一官公ソリューション事業部

\Orchestrating a brighter world

NECは、安全・安心・公平・効率という
社会価値を創造し、
誰もが人間性を十分に発揮できる
持続可能な社会の実現を目指します。

目　次

1. SQUIDの概要

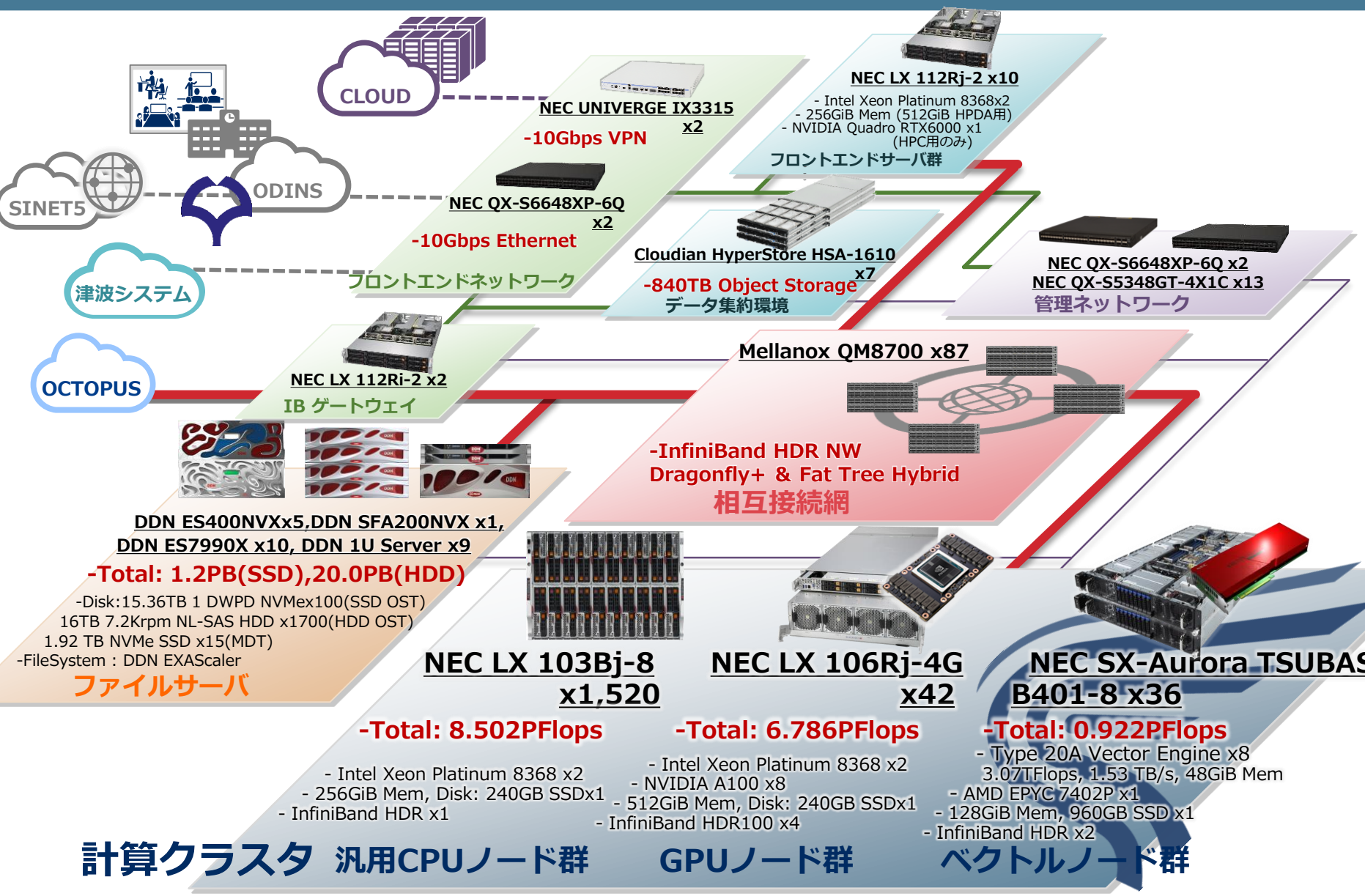
1. SQUIDの概要

SQUID

Supercomputer for Quest to Unsolved Interdisciplinary Datascience

本プロジェクトは、全国の国・公・私立大学等の研究者の学術研究のための利用を目的とし、計算機科学を応用した研究領域において、特に大規模な科学技術計算の需要に対応する、最高水準の計算およびデータ分析基盤を提供するためのスーパーコンピュータシステムを導入するものである。

1. SQUIDの概要



1. SQUIDの概要

SQUIDの特徴

Intel 最新鋭プロセッサ(IceLake)を含む総演算理論性能
16.21PFLOPS となる計算資源の拡充

高性能計算環境を支えるストレージ領域の拡充(21.2PB)
相互接続として、Infiniband HDRによるDragonfly+トポロジー

HPCの利便性向上に向けた、
コンテナサポート、及びモジュール環境の整備

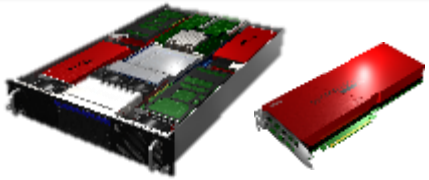
HPDAの利用者向けの対話型環境の強化
データ転送を用意にする多様なデータアクセス手段のサポート

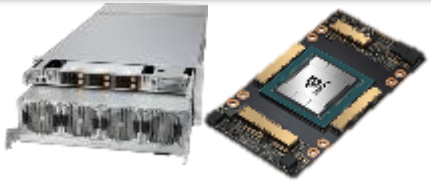
２段階認証をはじめとするセキュリティの強化

2. SQUIDの構成

2.1. 3つの計算環境

汎用CPU/ベクトル/GPGPUの3種の計算環境をご利用いただけます。





	汎用CPU計算環境	ベクトル計算環境	GPGPU計算環境
用途	一般的な計算用途。 大規模並列実行用	ベクトルエンジンによる 高メモリ帯域ベクトル演算用	GPGPU加速器による 高速演算計算用
モデル名	NEC LX 103Bj-8	NEC SX-Aurora TSUBASA B401-8	NEC LX 106Rj-4G
ノード数	1,520ノード	288 VE	42ノード
演算器	Intel Xeon Platinum 8368 (2.4GHz/28Core)x2	Type 20A Vector Engine (3.07 TFlops,1.53TB/s) x8	Intel Xeon Platinum 8368 (2.4GHz/28Core)x2
加速器	-	-	NVIDIA A100 x8
メモリ容量	256GiB	48GiB(HBM2)	512GiB
相互結合網	InfiniBand HDRx1	Infiniband HDR x2	InfiniBand HDR100x4

2.1. 3つの計算環境 - 汎用CPU計算環境

汎用CPU計算環境の CPUノードは、ブレードシステムを採用し、1,520ノードが、ラックあたり4シャーシ、シャーシあたり19ノードで構成されます。

汎用CPU計算環境
1,520ノード (20Rack)
80シャーシ

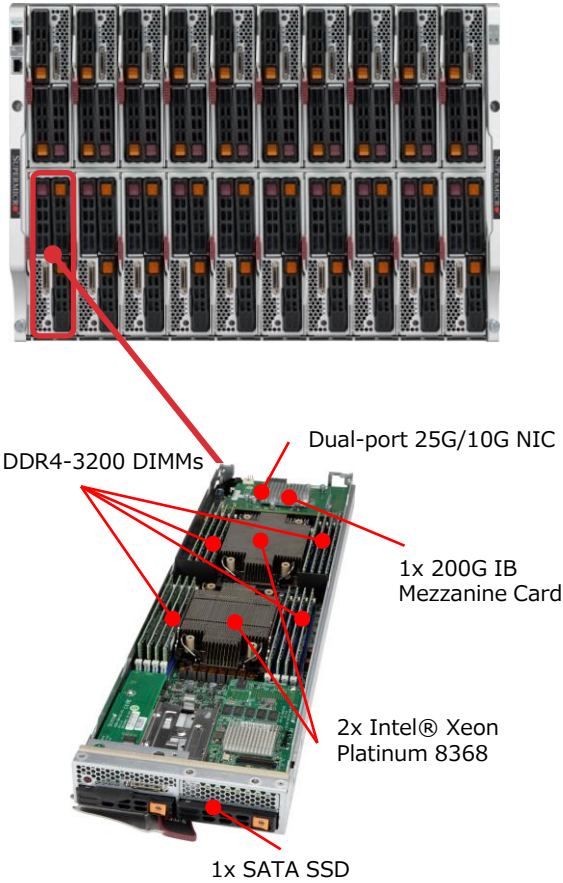
LX B2000E Enclosure
19ノード/シャーシ
IB HDR SW, 10GbE SW

CPUノード
LX 103Bj-8
CPU 2.4GHz/38core x2

2.1. 3つの計算環境 - 汎用CPU計算環境

汎用CPU計算環境 - CPUノード スペック

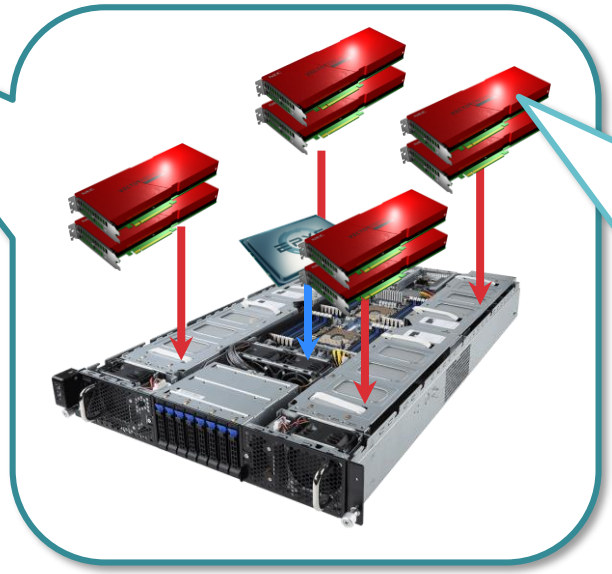
項目		構成
総ノード数		1,520台
サーバ構成	プロセッサ	Intel Xeon Platinum 8368 (2.4 GHz/38core) x2
	メモリ構成	256 GiB (16GiB DDR4-3200 RDIMM x16)
	ハードディスク	240GB SATA SSD x1
	インタフェース	InfiniBand HDR x1、25/10GbE x2、BMCx1
ソフトウェア環境	OS	CentOS 8.2 (64bit)
	コンパイラ	Intel Compiler
	MPI	Intel MPI



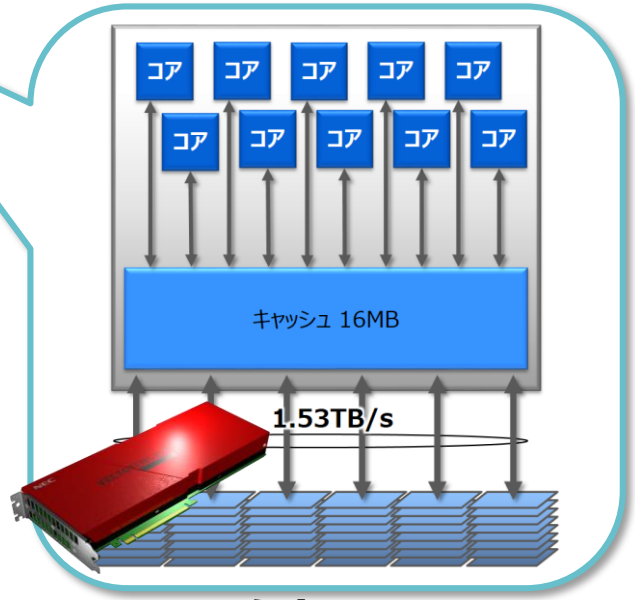
2.1. 3つの計算環境 - ベクトル計算環境

ベクトル計算環境は、旧システムの後継製品である NEC SX-Aurora TSUBASA を採用し、1VH(Vector Host)あたり8VE(Vector Engine)で、288 VEの構成です。

ベクトルノード群
36VH (2Rack)



VH(ベクトルホスト)
8VE + x86_64 サーバ
(288 VE)



VE(ベクトルエンジン)
10core / VE
(2,880 core)

並列処理

分散メモリ(MPI)並列で利用

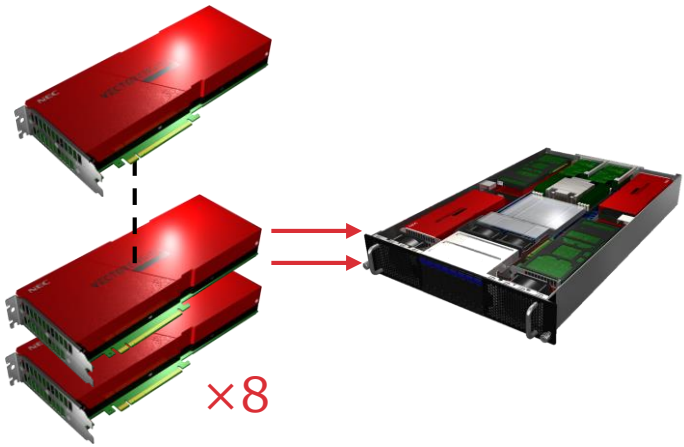
共有メモリ(スレッド)並列で利用

2.1. 3つの計算環境 - ベクトル計算環境

ベクトル計算環境 - ベクトルノード スペック

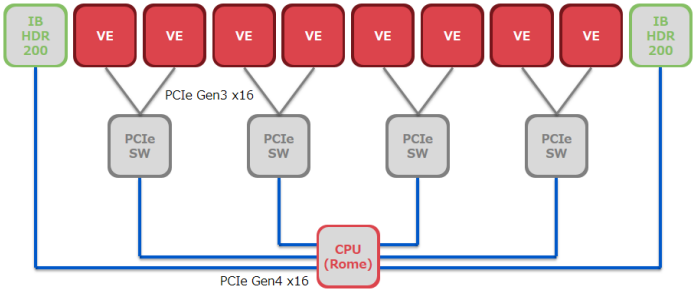
ベクトルエンジン(VE)

項目		構成
総VE数		288 VE (1VH当たり8VE)
モデル名		Type 20A
VE構成	演算性能(倍精度)	307 GFlops / 10core
	メモリ構成	48 GiB (HBM2)
	メモリ帯域	1.53 TB/s
ソフトウェア環境	OS	VEOS 2.5.0
	コンパイラ	NEC SDK for Vector Engine
	MPI	NEC MPI



ベクトルホスト(VH)

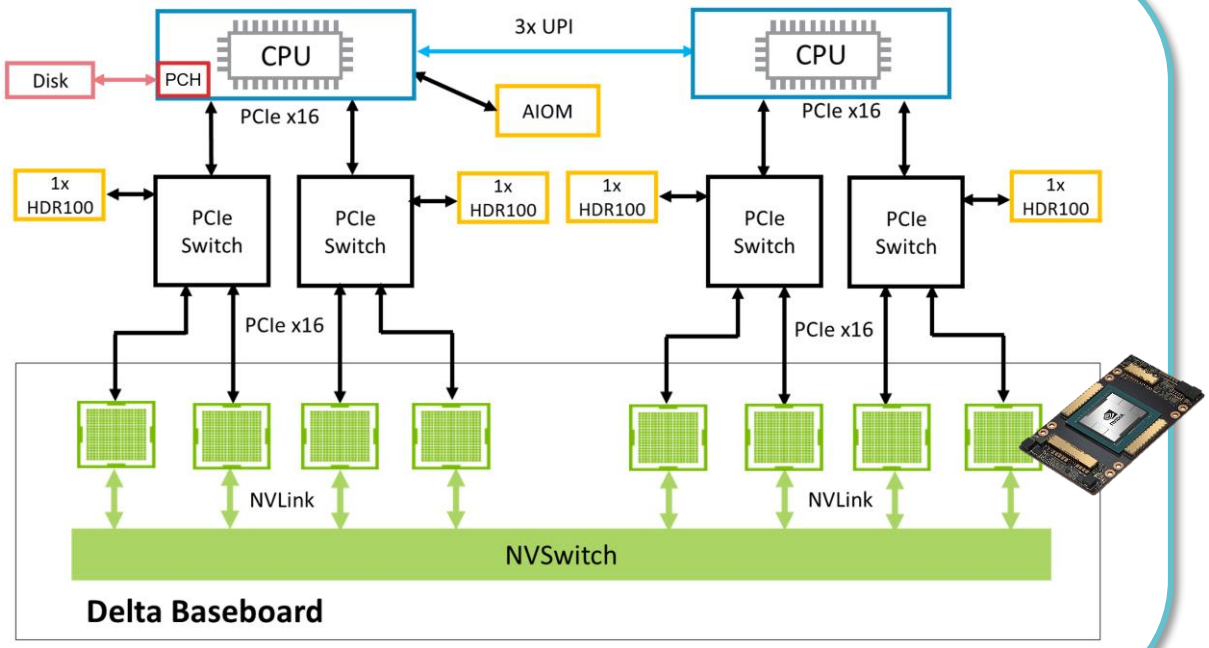
項目		構成
VH数		36VH
VH諸元	プロセッサ	AMD EPYC 7402P Processor(2.8GHz/24core) x1
	メモリ構成	128GiB (DDR4-3200 16GiB x8)
	ストレージ	960GB SATA SSD x1
	インタフェース	InfiniBand HDR x2, 1000Base-T x1, BMC
ソフトウェア環境	OS	CentOS 8.2 (64bit)



2.1. 3つの計算環境 - GPGPU計算環境

GPGPU計算環境は、NVIDIA A100 GPUを搭載したGPUノード、42ノードで構成されます。1ノードあたり、NVIDIA A100 GPUが8基搭載され、NVSwitch 経由で相互接続されます。

GPUノード群
42ノード（7Rack）



GPUノード
CPU 2.4GHz/38core x2
GPU 9.7TFlops x8 + NVSwitch

2.1. 3つの計算環境 - GPGPU計算環境

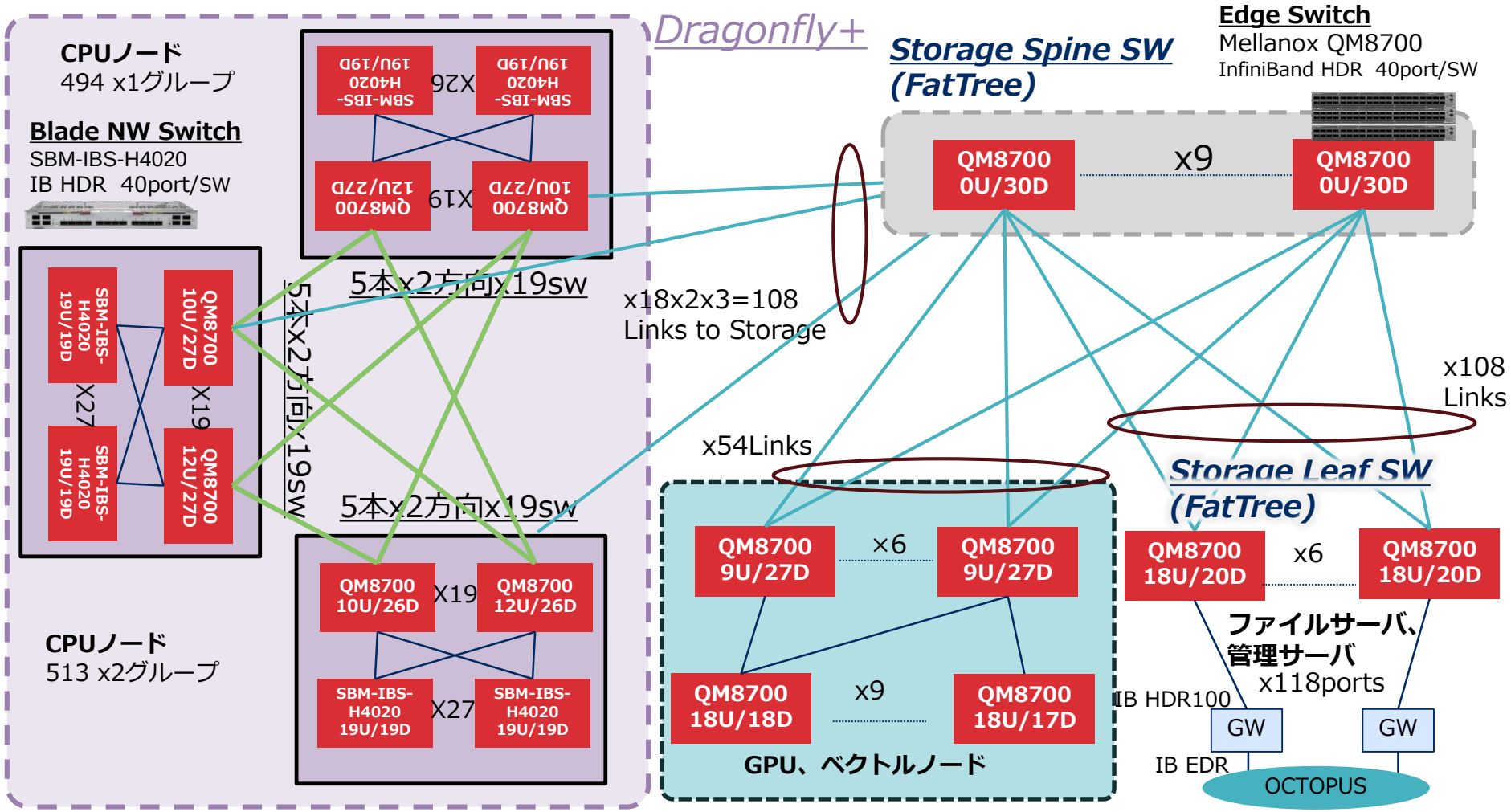
GPGPU計算環境 - GPUノード スペック

項目		構成
総ノード数		42ノード
サーバ構成	プロセッサ	Intel Xeon Platinum 8368 (2.4 GHz/38core) x2
	メモリ構成	512 GiB (32GiB DDR4-3200 ECC RDIMM x16)
	ハードディスク	240GB SATA SSD x1
	インタフェース	InfiniBand HDR100x4, 1000BASE-T x1, BMC x1
	GPGPU	NVIDIA A100 (SXM4) x8
ソフトウェア環境	OS	CentOS 8.2 (64bit)
	コンパイラ	NVIDIA HPC SDK
	MPI	OpenMPI

2.1. 3つの計算環境 - 相互接続網

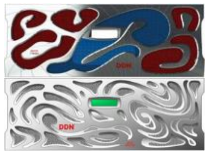
計算環境はInfiniband HDRの相互接続網で接続されます

- 汎用CPU計算環境：3グループによるDragonfly+ トポロジー
- ベクトル/GPGPU計算環境：Fat Tree トポロジー



2.2. 3つのストレージ領域

3つのストレージ領域がご利用いただけます。



	SSDストレージ	HDDストレージ	アーカイブストレージ
特徴	All Flashの環境による超高速IO領域	高速かつ大容量のデータ格納領域	複製や暗号化機能を有する柔軟なオブジェクトストレージ領域
容量	(追加購入)	home：10GiB work：5TiB + 追加購入	別途利用申請
ファイルシステム	DDN ExaScaler (Lustre)	DDN ExaScaler (Lustre)	Cloudian HyperStore
総容量	1.2 PB	20 PB	840 TB(物理容量)
ディスク装置	15.36TB NVMe SSD	16TB 7,200rpm NL-SAS	960GB SSD(メタデータ) 10TB SAS
ハードウェア	DDN ES400NVX x5	DDN ES7990X x10	Cloudian HyperStore Appliance 1610 x7

※計算環境から直接利用が可能なのは、SSDならびにHDD です。

2.3. 3つのストレージ領域

3つのフロントエンドがご利用いただけます。

	HPCフロントエンド	HPDAフロントエンド	セキュアフロントエンド
特徴	HPCアプリケーション開発向け環境 可視化処理環境、バッチジョブ投入環境	HPDA向けNoteBook環境 バッチジョブ投入環境	仮想マシンによる専用フロントエンド環境 バッチジョブ投入環境
ノード数	4 ノード	4 ノード	仮想マシン
アプリケーション	NICE DCV Server	Jupyter NoteBook	-
その他	NVIDIA Quadro RTX6000x1	512GiB メモリ(2倍)	別途利用申請

3. SQUID の 利用方法

3.1. ログイン方法

ログインはSSHによる2段階認証が必要です

ログイン先

サーバ	ホスト名
HPCフロントエンド	squidhpc.hpc.cmc.osaka-u.ac.jp
HPDAフロントエンド	squidhpda.hpc.cmc.osaka-u.ac.jp

2段階認証アプリ

※ 2段階認証には**事前にアプリのインストールが必要**です

OS	アプリ	入手元
Android	Google Authenticator	Google Play Store
iOS	Google Authenticator	Apple App Store
Windows	WinAuth	https://winauth.github.io/winauth/download.html
macOS	Step Two	Apple App Store

※利用可能アプリケーションの一例です

3.1. ログイン方法

初回ログイン

① 2段階認証アプリの入手
お持ちのスマートフォンやPCで、入手可能なアプリをインストールください。説明はGoogle Authenticator の例です。

② SSHアクセス
お手元のターミナルソフトからSSHアクセスします。
例：HPCフロントエンドの場合

```
$ ssh -l (利用者番号) squidhpc.hpc.cmc.osaka-u.ac.jp
Password: (パスワード)
```

✓ ユーザ名とパスワードは、申請後に通知される利用者番号と
利用者管理システムのパスワードを入力します。

3.1. ログイン方法

初回ログイン

③ 2段階認証アプリの入手
初回ログイン時、2段階認証のセットアップ画面が表示されます。

```
Initiallize google-authenticator
Warning: pasting the following URL info your browser exposes the OTP secret to Google:

https://www.google.com/chart?chs=200*200&chld=M|0&cht=qr&otpauth://totp/user1@squidhpc.hpc.cmc.osaka-u.ac.jp%3Fsecret%3DDXXXXXXXXCLI%26issuer%3Dsquidhpc.hpc.cmc.osaka-u.ac.jp
```

※ QRコード

```
Your new secret key is: XXXXXXXXXXX
Enter code from app (-1 to skip):
```

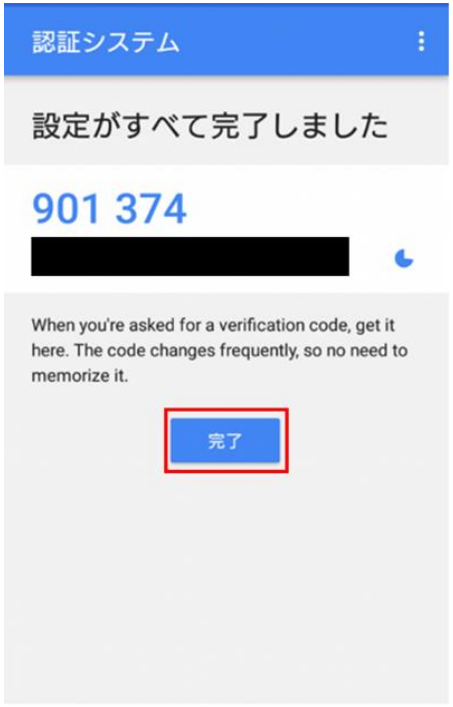
✓ ウィンドウサイズによってはQRコードが崩れます。フォントサイズを調整するか、記載のURL、シークレットキーをお使いください。

3.1. ログイン方法

初回ログイン

④ 認証コードの読込
手元のアプリを起動し、QRコードまたはシークレットキーを読み込みます。

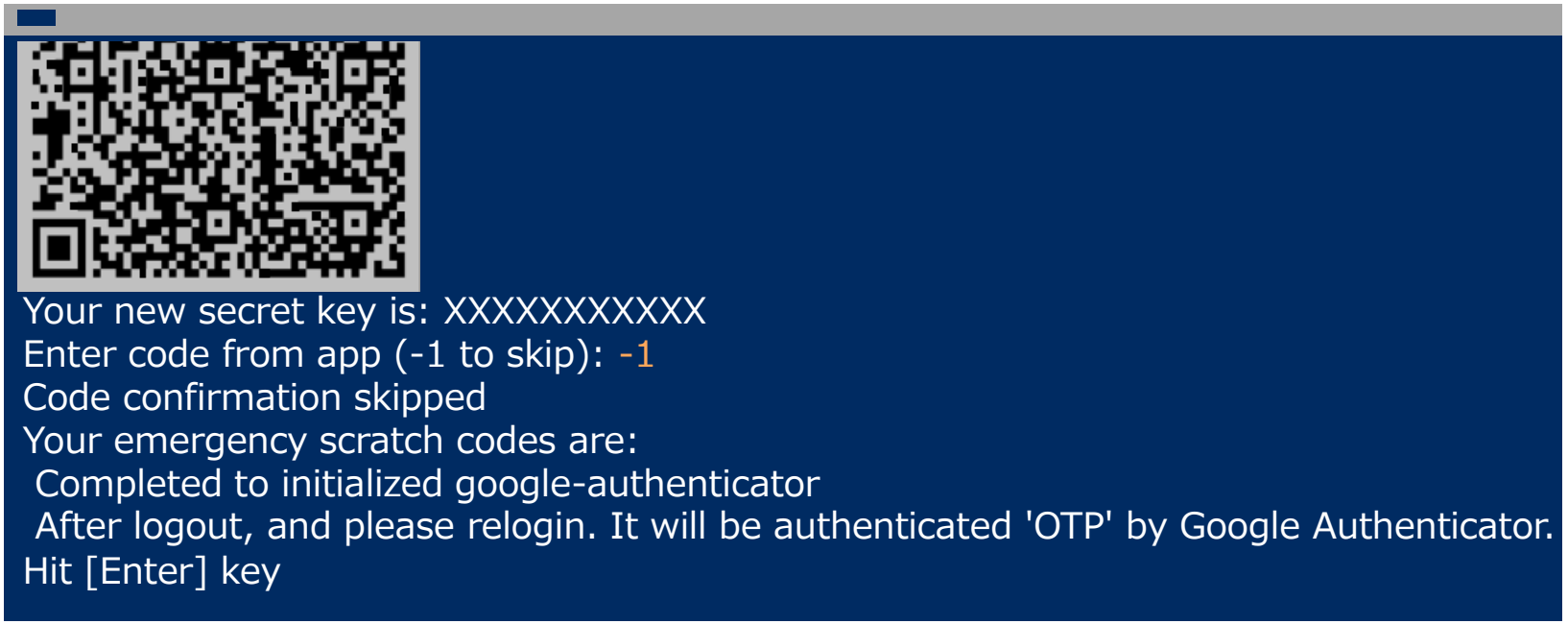
⑤ 読込完了
完了するとGoogle Authenticatorに、登録され、ワンタイムパスワードが発行されます。



3.1. ログイン方法

初回ログイン

⑥ ログアウト
ターミナルソフト側は、「Enter code from app」と聞かれていますので、「-1」を入力し、一度ターミナルを終了します。



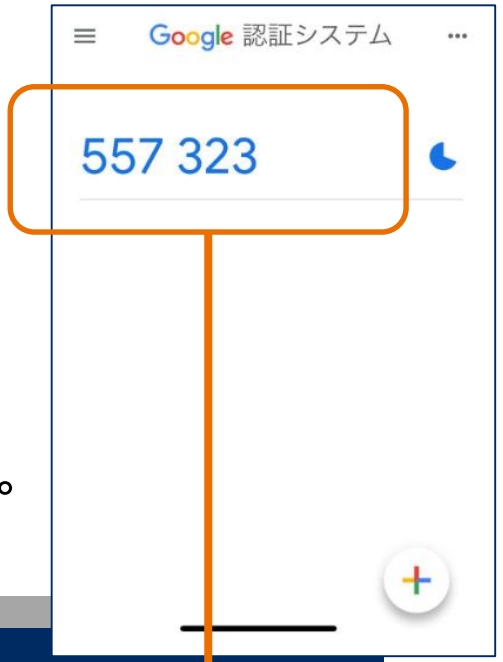
✓ 2段階認証の初期登録方法は以上です。

3.1. ログイン方法

2回目以降のログイン

① 2段階認証アプリの起動
お持ちのスマートフォンやPCで、アプリを起動し、ワンタイムパスワードを確認します。

✓ ワンタイムパスワードは、時間経過で変化します。



② SSHアクセス
お手元のターミナルソフトからSSHアクセスします。
例：HPCフロントエンドの場合

```
$ ssh -l （利用者番号） squidhpc.hpc.cmc.osaka-u.ac.jp
Password: （パスワード）
Verification code: （ワンタイムパスワード）
```

✓ ユーザ名とパスワードは、申請後に通知される利用者番号と利用者管理システムのパスワードを入力します。
✓ アプリ上で確認したワンタイムパスワードをターミナルに入力します。

ログイン手順は以上です。

3.2. フロントエンドの利用

ファイルシステム

ファイルシステムとして、SSDストレージおよびHDDストレージに直接アクセス可能です。ご利用可能なディスク領域およびそのクォータ制限は以下の通りです。

ストレージ	ファイルシステム	領域名	パス	サイズクォータ	備考
HDD	EXAScaler (Lustre)	ホーム領域	/sqfs/home/(利用者番号)	10GiB	ブラウザアクセス領域を含む
		拡張領域	/sqfs/work/(グループ名)/(利用者番号)	5TiB(+)	追加購入可
			/sqfs/s3/(UUID)		S3アクセス用
SSD		高速領域	/sqfs/ssd/(グループ名)/(利用者番号)	0B(+)	追加購入可

3.2. フロントエンドの利用

ファイルシステムの特徴

ホーム領域

- ユーザ登録時に与えられる最低限の領域です。初期容量として、10GiBが割り当てられます。
- WEBブラウザから、アクセスするための領域を含みます。

拡張領域

- 初期容量として5TiBが割り当てられており、申請によって所属グループ毎に容量を追加購入可能です。
- SQUID外部からS3 APIでアクセスするための領域が、別パスで用意されます。

高速領域

- SSDストレージによる超高速I/Oが可能です。
- 初期容量の割り当てはありませんが、申請によって所属グループ毎に容量を追加購入が可能です。

3.2. フロントエンドの利用

利用状況の確認 (usage_view)

計算環境は実行時間と消費係数に基づくポイントを消費して利用します。
ストレージは購入額に応じたクォータ制限の範囲で利用します。
これらの利用状況を確認するには、usage_viewコマンドを利用します。

```
$ usage_view
---------------+ Group: G012345 +---------------

[Group summary]

          SQUID points    HDD(GiB)    SSD(GiB)
------- -------------- ---------- ----------
usage              0.0        87.9        0.0
limit          11000.0     46080.0     20480.0
remain         11000.0     45992.1     20480.0
rate(%)            0.0         0.2         0.0

[Detail]

          SQUID points              Home(GiB)    HDD(GiB)    SSD(GiB)
------- -------------- -------------------- ---------- ----------
```

グループの利用状況
・ポイント消費
・HDD領域の利用量
・SSD領域の利用量

3.2. フロントエンドの利用

利用状況の確認 (usage_view)

```
[DETAIL]
```

	SQUID POINTS		HOME(GIB)	HDD(GIB)	SSD(GIB)
USER001	0.0	0.0 / 10.0 / 10.0		0.0	0.0
USER002	0.0	0.0 / 10.0 / 10.0		0.0	0.0
USER003	0.0	0.0 / 10.0 / 10.0		0.0	0.0

グループ利用のユーザ毎内訳

```
[NODE-HOURS]
```

			USAGE	AVAILABLE*
CPU		NODE	0.00	9090.90
GPU		NODE	0.00	8184.52
VECTOR		NODE	0.00	8461.53
AZURE	CPU	NODE	0.00	6547.61
	GPU	NODE	0.00	0.00
OCI	CPU	NODE	0.00	6547.61
	GPU	NODE	0.00	0.00
SECURE	CPU	NODE	0.00	6666.66
	GPU	NODE	0.00	6666.66
	VECTOR	NODE	0.00	6666.66

計算ノード毎のポイント利用状況
及び残りの見込み使用可能時間

3.2. フロントエンドの利用

環境設定（ Environment Module ）

- コンパイラやライブラリの環境変数は、Environment Moduleで制御します。module コマンドを使用して、開発環境を準備します

- 基本オプション

コマンド	説明
module avail	利用可能な開発環境/アプリの一覧表示
module list	読み込み済みのモジュールの一覧表示
module switch [file1] [file2]	モジュールの入れ替え（file1 → file2)
module load [file]	モジュールの読み込み
module unload [file]	モジュールの読み込み解除
module purge	ロード済みの全モジュールの解除
module show [file]	モジュールの詳細表示

3.2. フロントエンドの利用

基本環境と推奨環境

モジュールは標準的な開発環境をまとめた基本環境を用意しています。基本環境の中には、各計算環境の利用に適した推奨環境を用意しています。

基本環境

種別	モジュール名	推奨環境	内容
コンパイラ＋MPI ＋ライブラリ	BaseCPU/2021	汎用CPU	汎用CPUノード向けプログラム開発の推奨環境
	BaseVEC/2021	ベクトル	ベクトルノード向けプログラム開発の推奨環境
	BaseGPU/2021	GPGPU	GPUノード向けプログラム開発の推奨環境
	BaseGCC/2021		GCCを利用する際の開発環境
言語環境＋モジュール	BasePy2/2021		Python2 向けのプログラム開発環境
	BasePy3/2021		Python3 向けのプログラム開発環境
	BaseR/2021		R言語向けのプログラム開発環境
	BaseJulia/2021		Julia言語向けのプログラム開発環境
アプリケーション	BaseApp/2021		ISV及びOSSアプリケーション向けのベース環境

具体的な利用方法は、次節にてご説明します。

4. プログラムの開発、実行

コンパイラ、MPI

標準的な開発環境は、計算環境別にモジュールが用意されています。

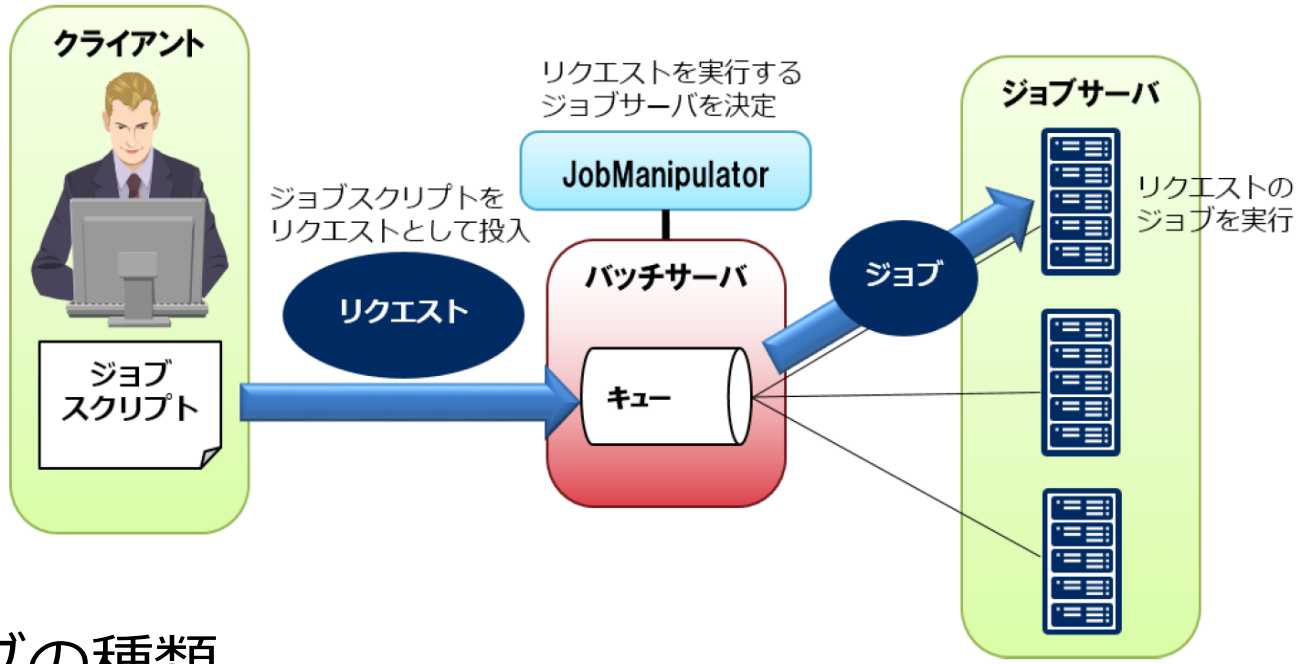
計算環境	推奨 モジュール名	コンパイラ	MPI
汎用CPU	BaseCPU	Intel Parallell Studio	Intel MPI
ベクトル	BaseVEC	NEC SDK for VE	NEC MPI
GPGPU	BaseGPU	NVIDIA HPC SDK CUDA	OpenMPI

```
$ module avail
--------------------------- /system/apps/env/base -------------------------------
BaseCPU/2021(default)     BaseVEC/2021     BaseGPU/2021     BaseGCC/2021
BasePy2/2021              BasePy3/2021     BaseApp/2021

$ module load BaseCPU/2021                  汎用CPU向け推奨環境を読み込み

$ ifort -o sample.out  sample.f90           Intel Compilerによるコンパイル

$ mpiifort -o sample_mpi sample_mpi.f90     Intel MPIを用いたコンパイル
```

ジョブの実行

ジョブ管理システム (NQSV)

計算環境の利用は、ジョブ管理システムにジョブを投入して利用します。
混雑状況をみて、システム側で実行時刻とサーバを決定します。



ジョブの種類

- バッチジョブ：実行内容を記述したジョブスクリプトを投入します。非対話的に実行されます。
- インタラクティブジョブ：デバッグ用などに、対話的な実行環境を要求します。

バッチリクエストの実行

qsub コマンドにて、バッチジョブを投入します。
投入にはグループの指定が必要です。

```
$ qsub　[オプション]　[ジョブスクリプトファイル名]
```

qsubコマンドを実行すると、リクエストIDが採番され、下記のように標準出力に表示されます。

Request **1182.sqd** submitted to queue: small

ジョブスクリプト例

```
1   #!/bin/bash
2   #------- qsub option -----------
3   #PBS -q SQUID
4   #PBS --group=G01234
5   #PBS -l elapstim_req=01:00:00
6   #PBS -l cpunum_job=76
7   #PBS -b 1
8   #PBS -v OMP_NUM_THREADS=76
9
10  #------- Program execution -----------
11  module load BaseCPU/2021
12  cd ${PBS_O_WORKDIR}
13  ./a.out
```

必須オプションとして、利用申請時に登録される**グループ名が必要**です。

ジョブの実行

ジョブクラスと投入キュー

- ジョブはいくつかジョブクラスに分類されます。
  ジョブクラスは、ジョブ管理システム上のキューに対応しており、利用者はキューにジョブを投入することで計算環境の利用が可能です。
- 一部のキューを除いて基本的には、汎用CPU/GPGPU計算環境はノード単位、ベクトル計算環境はVE単位の利用となります。

投入キュー名	種類	用途	備考
SQUID	バッチ	標準利用のためのジョブクラス	1ノード(VE)あたり1ジョブ利用
SQUID-S	バッチ	他のジョブとのノード内共有を許容し、ポイント消費を抑えたい方向け	
SQUID-H	バッチ	ポイント消費を多くし**高優先度ジョブを投入**し、待ち時間を短縮したい方向け	
SQUID-R	バッチ	NW帯域が狭い経路の利用を許容して、実行待ち時間を短縮したい方向け	
DBG	バッチ	デバッグ用の短時間の利用向け	キュー優先度高
INTX	インタラクティブ	デバッグ用の対話型利用向け	キュー優先度高

ジョブの実行

ジョブクラス (汎用CPU計算環境向け)

種別	ジョブクラス	利用可能経過時間	利用可能最大Core数	利用可能メモリ	同時利用可能ノード数	備考
共有利用	SQUID	24時間	38,912core (76c/ノード)	124TiB (248GB/ノード)	512	ノード内は占有利用
	SQUID-R	24時間	38,912core (76c/ノード)	124TiB (248GB/ノード)	512	※1
	SQUID-H	24時間	38,912core (76c/ノード)	124TiB (248GB/ノード)	512	※2
	SQUID-S	24時間	38core	124GiB	1	※3
	DBG	10分	152core (76c/ノード)	496GiB (248GB/ノード)	2	デバッグ用
	INTC	10分	152core (76c/ノード)	496GiB (248GB/ノード)	2	インタラクティブ利用
占有利用	mySQUID	無制限	76core × 占有ノード数	248GiB × 占有ノード数	占有数	別途利用申請

ジョブの実行

ジョブクラス (ベクトル計算環境向け)

種別	ジョブクラス	利用可能経過時間	利用可能最大Core数	利用可能メモリ	同時利用可能VE数	備考
共有利用	SQUID	24時間	2,560core (10c/VE)	12TiB (48GB/VE)	256	
	SQUID-H	24時間	2,560core (10c/VE)	12TiB (48GB/VE)	256	※1
	SQUID-S	24時間	40core	192GiB	4	※2
	DBG	10分	20core (10c/VE)	96GiB (48GB/VE)	2	デバッグ用
	INTV	10分	20core (10c/VE)	96GiB (48GB/VE)	2	インタラクティブ利用
占有利用	mySQUID	無制限	10core × 占有VE数	48GiB × 占有VE数	占有数	別途利用申請

ジョブの実行

ジョブクラス (GPGPU計算環境向け)

種別	ジョブクラス	利用可能経過時間	利用可能最大Core数	利用可能メモリ	同時利用可能ノード数	備考
共有利用	SQUID	24時間	2,432core (76c/ノード)	15.75TiB (504GB/ノード)	32	
	SQUID-H	24時間	2,432core (76c/ノード)	15.75TiB (504GB/ノード)	32	※1
	SQUID-S	24時間	38core	252GiB	1	※2
	DBG	10分	152core (76c/ノード)	1,008GiB (504GB/VE)	2	デバッグ用
	INTG	10分	152core (76c/ノード)	1,008GiB (504GB/VE)	2	インタラクティブ利用
占有利用	mySQUID	無制限	76core × 占有ノード数	504GiB × 占有ノード数	占有数	別途利用申請

CASE1：汎用CPU計算環境でスレッド並列ジョブを実行する

スレッド並列　1ノード で76スレッド実行の例

青字はコメントです

```
1   #!/bin/bash
2   #------- qsub option -----------
3   #PBS -q SQUID                    # 投入キューに　SQUID を指定
4   #PBS --group=G01234              # ポイント消費先のグループ
5   #PBS -l elapstim_req=00:30:00    # 実行時間を指定
6   #PBS -v OMP_NUM_THREADS=76       # スレッド数にCPUコア数を指定
7
8   #------- Program execution -----------
9
10  module load BaseCPU/2021         # コンパイル時に module load していたものを記載
11  module load xxx/xxx
12
13  cd $PBS_O_WORKDIR                # qsub したディレクトリに移動
14  ./a.out                          # 1proc 76thread の実行
15
16
17
18
19
```

CASE2：汎用CPU計算環境でMPI並列ジョブを実行する

MPI並列(Flat MPI) 4ノード304プロセスの実行例

青字はコメントです

```
1   #!/bin/bash
2   #------- qsub option -----------
3   #PBS -q SQUID                          # 投入キューに  SQUID を指定
4   #PBS --group=G01234                    # ポイント消費先のグループ
5   #PBS -l elapstim_req=00:30:00          # 実行時間を指定
6   #PBS -b 4                              # 実行ノード数を指定
7   #PBS -T intmpi                         # MPIに Intel MPI を指定
8
9   #------- Program execution -----------
10
11  module load BaseCPU/2021               # コンパイル時に module load していたものを記載
12  module load xxx/xxx
13
14  cd $PBS_O_WORKDIR                      # qsub したディレクトリに移動
15  mpirun ${NQSV_MPIOPTS} -np 304 ./a.out # Intel MPI の引数に NQSV_MPIOPTSを引き渡し
16
17
18
19
```

CASE3：ベクトル計算環境でスレッド並列ジョブを実行する

スレッド並列　1VEで10スレッド実行の例

青字はコメントです

```
1   #!/bin/bash
2   #------- qsub option -----------
3   #PBS -q SQUID-S                  # 投入キューにSQUID-Sを指定 ※SQUIDは8VEからのため
4   #PBS --group=G01234              # ポイント消費先のグループ
5   #PBS -l elapstim_req=00:30:00    # 実行時間を指定
6   #PBS --venode=1                  # VE数を指定
7   #PBS -v OMP_NUM_THREADS=10       # VEのコア数 10 を指定
8
9   #------- Program execution -----------
10
11  module load BaseVEC/2021         # コンパイル時に module load していたものを記載
12  module load xxx/xxx
13
14  cd $PBS_O_WORKDIR                # qsub したディレクトリに移動
15  ./a.out                          # 1proc 10thread/VE  x 1VEの実行
16
17
18
19
```

CASE4：ベクトル計算環境でMPI並列ジョブを実行する

MPI並列(Flat MPI) 40VE 400プロセスの実行例

青字はコメントです

```
1   #!/bin/bash
2   #------- qsub option -----------
3   #PBS -q SQUID                       # 投入キューにSQUIDを指定
4   #PBS --group=G01234                 # ポイント消費先のグループ
5   #PBS -l elapstim_req=00:30:00       # 実行時間を指定
6   #PBS --venode=40                    # 総VE数を指定    ※ 8VE/VH x 5VH の 指定
7   #PBS -T necmpi                      # MPIに NEC MPI を指定
8
9   #------- Program execution -----------
10
11  module load BaseVEC/2021            # コンパイル時に module load していたものを記載
12  module load xxx/xxx
13
14  cd $PBS_O_WORKDIR                   # qsub したディレクトリに移動
15  mpirun -np 400 ./a.out              # 10proc/VE x 40VE の実行
16
17
18
19
```

CASE5：GPGPU計算環境でスレッド並列ジョブを実行する

スレッド並列　1ノード で48スレッド実行の例

青字はコメントです

```
1   #!/bin/bash
2   #------- qsub option -----------
3   #PBS -q SQUID                          # 投入キューにSQUIDを指定
4   #PBS --group=G01234                    # ポイント消費先のグループ
5   #PBS -l elapstim_req=00:30:00          # 実行時間を指定
6   #PBS -l gpunum_job=8                   # GPU数の指定
7   #PBS -v OMP_NUM_THREADS=48             # スレッド数の指定
8
9   #------- Program execution -----------
10
11  module load BaseGPU/2021               # コンパイル時に module load していたものを記載
12  module load xxx/xxx
13
14  cd $PBS_O_WORKDIR                      # qsub したディレクトリに移動
15  ./a.out
16
17
18
19
```

CASE6：GPGPU計算環境でMPI並列ジョブを実行する

MPI並列　2ノード で4MPIプロセスの例

青字はコメントです

```
1   #!/bin/bash
2   #------- qsub option -----------
3   #PBS -q SQUID                          # 投入キューにSQUIDを指定
4   #PBS --group=G01234                    # ポイント消費先のグループ
5   #PBS -l elapstim_req=00:30:00          # 実行時間を指定
6   #PBS -b 2                              # 実行ノード数を指定
7   #PBS -l gpunum_job=8                   # 1ノードあたりのGPU数を指定
8   #PBS -T openmpi                        # MPIに OpenMPI を指定
9   #PBS -v NQSV_MPI_MODULE=BaseGPU/2021   # MPIを読み込むモジュール名を指定
10
11  #------- Program execution -----------
12
13  module load BaseGPU/2021               # コンパイル時に module load していたものを記載
14  module load xxx/xxx
15
16  cd ${PBS_O_WORKDIR}                    # qsub したディレクトリに移動
17  mpirun ${NQSV_MPIOPTS} -np 4 -npernode 2  ./mpi_prog
18                                         # OpenMPI の引数に NQSV_MPIOPTSを引き渡し
19
```

CASE7：GPGPU計算環境でコンテナジョブを実行する

MPI並列　2ノード で 16MPIプロセスのコンテナジョブの例

青字はコメントです

```
1   #!/bin/bash
2   #------- qsub option -----------
3   #PBS -q SQUID                          # 投入キューにSQUIDを指定
4   #PBS --group=G01234                    # ポイント消費先のグループ
5   #PBS -l elapstim_req=00:30:00          # 実行時間を指定
6   #PBS -b 2                              # 実行ノード数を指定
7   #PBS -l gpunum_job=8                   # 1ノードあたりのGPU数を指定
8   #PBS -T openmpi                        # MPIに OpenMPI を指定
9   #PBS -v NQSV_MPI_MODULE=BaseGPU/2021   # MPIを読み込むモジュール名を指定
10
11  #------- Program execution -----------
12
13  module load BaseGPU/2021               # コンパイル時に module load していたものを記載
14  module load xxx/xxx
15
16  cd ${PBS_O_WORKDIR}                    # qsub したディレクトリに移動
17  mpirun ${NQSV_MPIOPTS} -np 16 -npernode 8  ¥
18            singularity exec --nv （コンテナ名）（コンテナ内プログラム名）
19                                         # singularity の引数に --nv を指定する
```

5. データアクセス

WEBブラウザでのデータアクセス

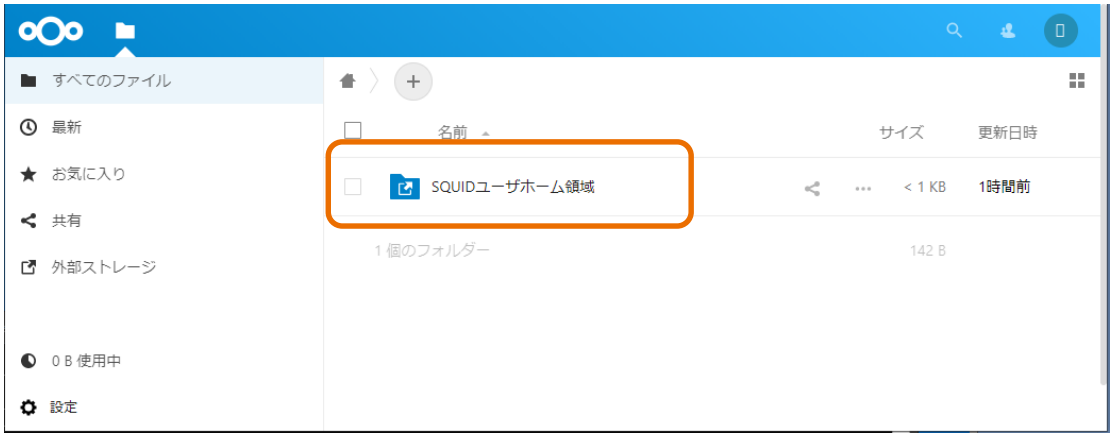
WEBブラウザを利用したグラフィカルなデータ転送手段

ログイン先

サーバ	URL
データ転送ゲートウェイ	https://onionweb.hpc.cmc.osaka-u.ac.jp

ブラウザ操作

① ユーザ名とパスワードを入力しログイン



② SQUIDホーム領域 にドラッグアンドドロップ等でデータ転送

SQUID内の保存先

領域	パス
ホーム領域	/sqfs/home/(利用者番号)/OnionWeb/

S3 API でのデータアクセス

S3 APIを利用したアプリケーションからのデータ転送手段

エンドポイント名

サーバ	エンドポイント名
S3DS	https://squidgw.hpc.cmc.Osaka-u.ac.jp’

アクセスキーの発行

フロントエンド上で、s3dskey コマンドによりキーを発行します

```
$ s3dskey create --group=(グループ名)
+-----------+-----------------------------------------+
| accesskey | AKIA7X1KXXXYYYZZZ000                    |
| enabled   | True                                    |
| fspaths   | None                                    |
| fsuid     | 60101:10                                |
| secretkey | hagwyutos8TThj3SOv9ahPJMF8TTK2dYFcV9Qv7T |
| tag       | (連番):(ユーザ名):(グループ名)                  |
| uuid      | ea60f00a60hufaweofapo12813nfawe9506e1216 |
+-----------+-----------------------------------------+
```

アクセスキー

シークレットキー

UUID

S3 API でのデータアクセス

外部からのS3 APIでのデータ操作

s3curl を例としてご説明します。設定ファイル(~/.s3curl)

```
%awsSecretAccessKeys = (
   username => {
         id => 'AKIA7X1KXXXYYYZZZ000',
         key => 'hagwyutos8TThj3SOv9ahPJMF8TTK2dYFcV9Qv7T',
   },
);
push(@endpoints, 'squidgw.hpc.cmc.osaka-u.ac.jp')
```

設定名
アクセスキー
シークレットキー
エンドポイント名

バケット作成 + データアップロード操作例

```
$ LANG=C
$ s3curl.pl --id username --createBucket -- ¥
                          -s https://squidgw.hpc.cmc.osaka-u.ac.jp/usernametestbucket

$ s3curl.pl --id username --put (ローカルファイル名) -- -s ¥
              -v https://squidgw.hpc.cmc.osaka-u.ac.jp/usernametestbucket/(オブジェクト名)
```

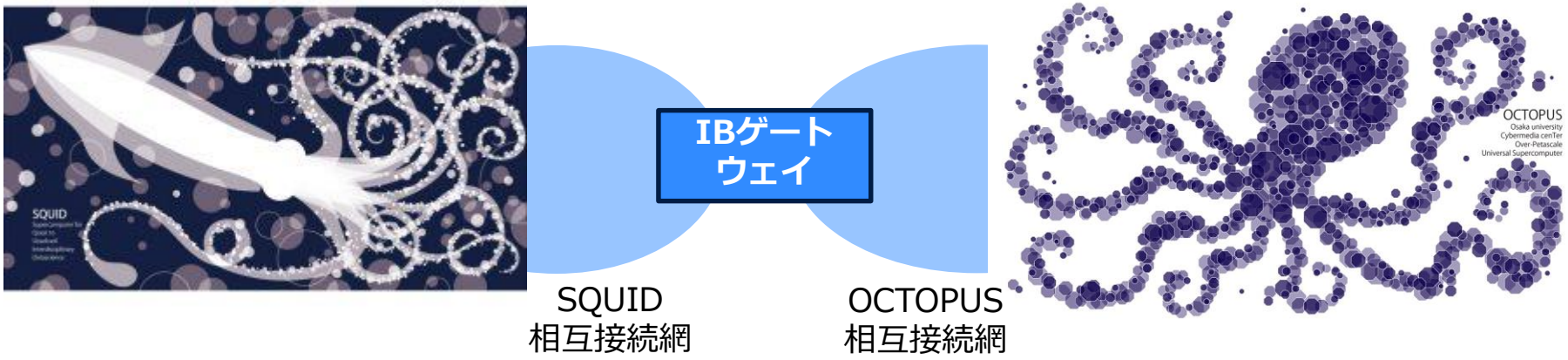
バケット名

SQUID内の保存先

領域	パス
拡張領域	/sqfs/s3/(UUID)/(バケット名)/(オブジェクト名)

OCTOPUSからのデータアクセス

OCTOPUS - SQUID 間は、Infiniband 経由でデータ転送可能です



- データ保存領域は下記で参照可能なため、フロントエンド上で相互にデータ転送が可能です

ファイルシステム	ファイルパス
高速領域	/sqfs/ssd
ホーム領域	/sqfs/home
拡張領域	/sqfs/work

ファイルシステム	ファイルパス
ホーム領域	/octfs/home
拡張領域	/octfs/work


Orchestrating a brighter world
NEC