

# HPCネットワークの多様化に挑む マルチベンダー×マルチOSで支えるHPCネットワーク運用の実際

さくらインターネット株式会社

黒澤 潔裕

2026/2/11



はじめに

アーキテクチャの試行

ホワイトボックスへの挑戦

自動化への取り組み

まとめ

はじめに

さくらインターネット株式会社  
クラウド事業本部 クラウドサービス部

## 黒澤 潔裕 / KUROSAWA Kiyohiro

### 業務内容：

- ・ GPU基盤関連のネットワーク設計
- ・ 上記に付随した自動化

### 経歴：

- ・ 2024/9 さくらインターネット 入社
- ・ 某通信事業者にてCGN/N6,SGI用 Clos, Routerの設計
- ・ 某コンテンツ事業者で3000台のサーバーを7人で運用するお仕事

### JANOG歴：

- ・ 登壇2回目（初一人登壇）



今日の話は、  
こんな人に向けています

Closを「理想的な構成」だと思っている人

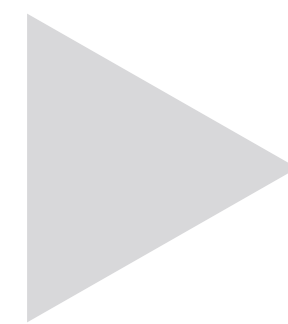
新たなベンダーへの挑戦はコスト削減策だと思っている人

自動化は“楽をするため”だと思っている人

さくらインターネットが挑戦した結果、皆さんがどう思うかを議論したい

## 構築済みHPCクラスタをマネージドサービスとして提供

- DC基盤
  - 電力（MW級）、冷却（空冷/液冷）、PUE最適化
- 計算リソース
  - 計算用のGPU（B200 / H200 / H100）
- ストレージ
  - Lustre
- ソフトウェア基盤
  - Slurm / MPI / CUDA・ROCm / NCCL
- ネットワーク
  - Multi-Tenancy/Traffic/Lossless Ethernet/Cabling

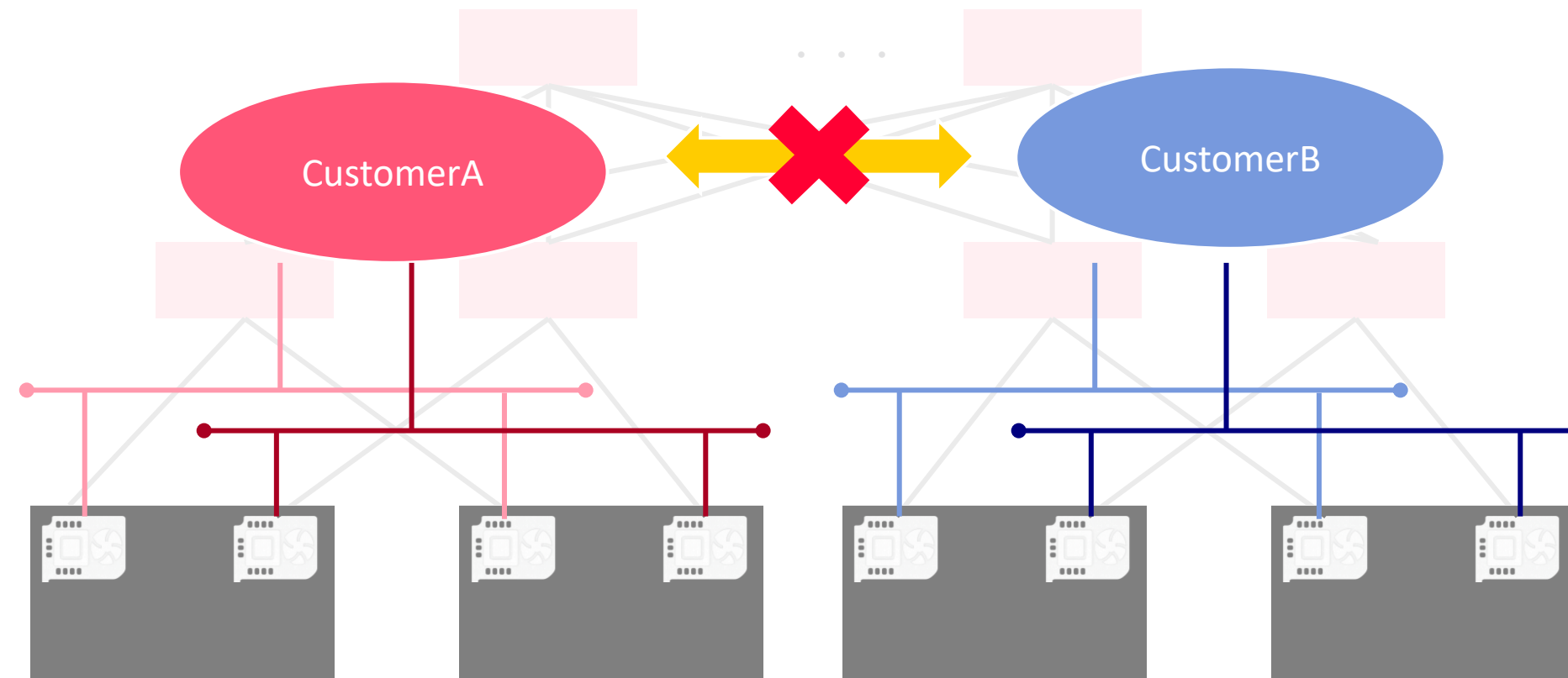


透明性の高い技術による実装を重視

# GPU クラスタにおける、高帯域かつlosslessの実現

## Multi-Tenancy

VRF  
VLAN  
EVPN/VXLAN

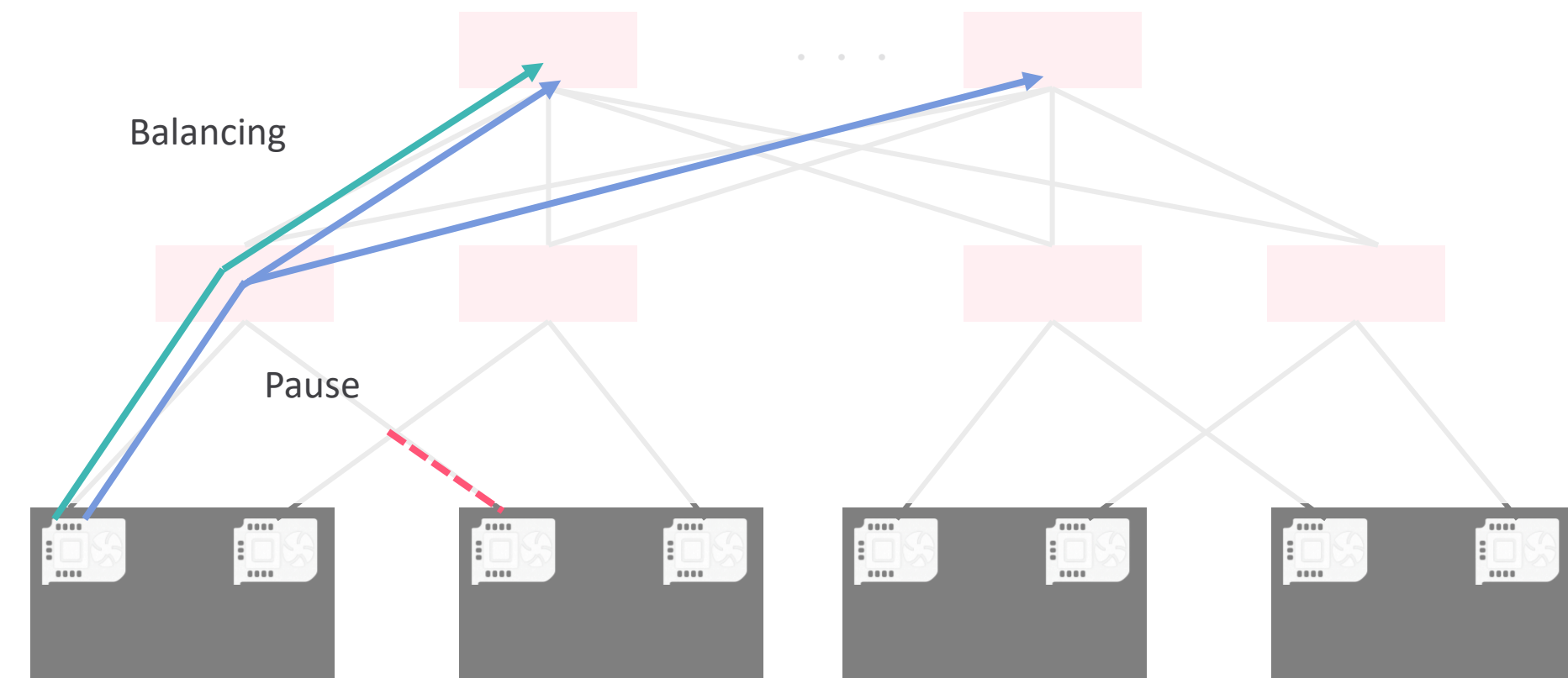


## High Traffic

Switch間接続 800Gbps  
GPU接続 400Gbps

## Lossless Ethernet

RoCEv2(ECN/PFC/CNP)  
Dynamic Load Balancing



## Cabling

高密度収容を実現するケーブル実装  
NW機器間: 800GBASE-SR8 572本  
GPU接続: 400GBASE-DR4 1400本



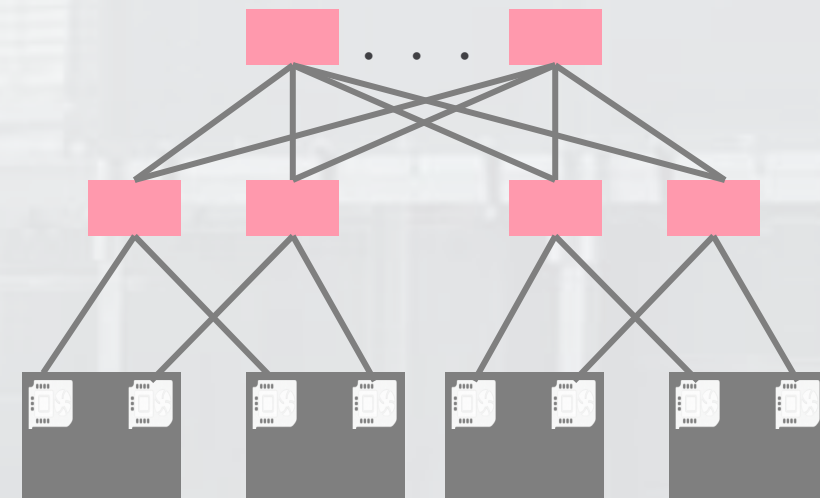
# クラスタ規模に合わせた最適なアーキテクチャを選定



さくら ONE

H100 GPU クラスタ  
2024/12 S-in

100 Servers



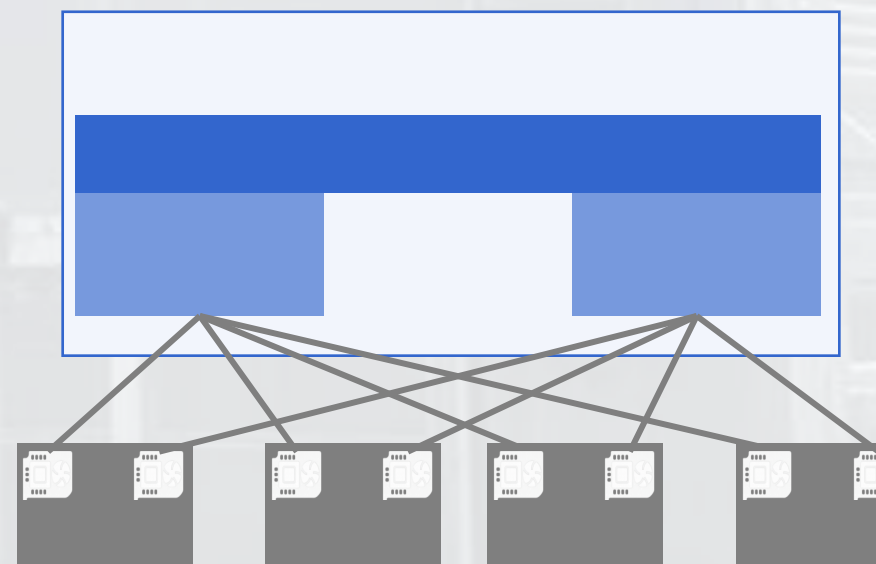
Clos Topology

2025/6 「ISC2025」  
処理性能ランキング  
TOP500にて世界49位

H200/B200 GPU クラスタ  
2025/6-8 S-in

50-60 Servers

ARISTA

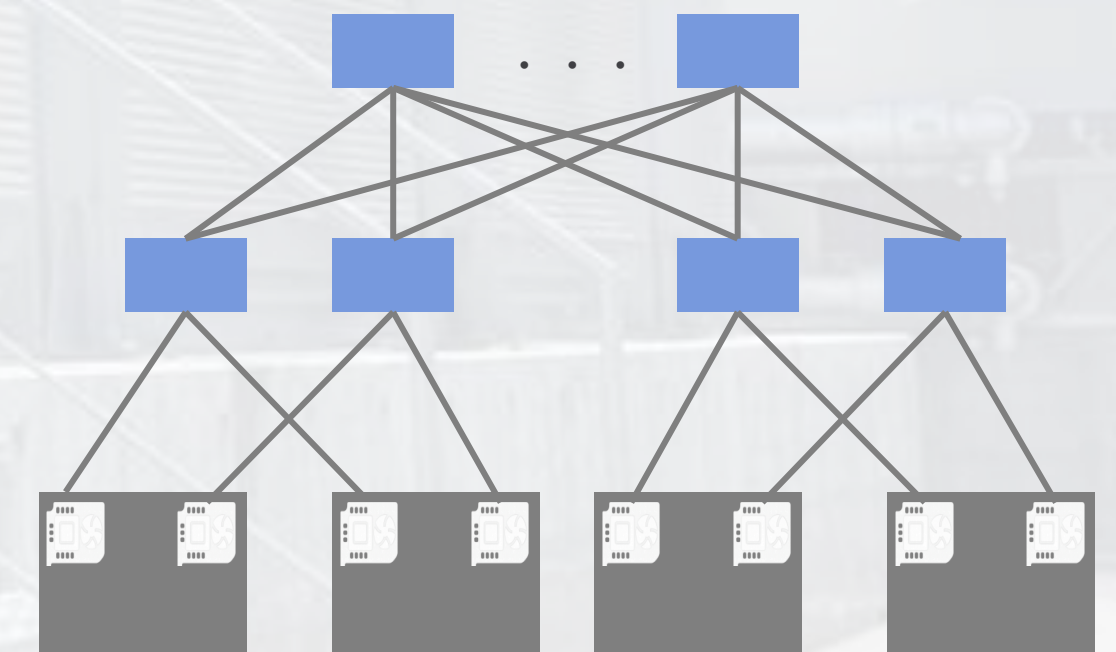


Chassis Switch

B200 GPU クラスタ  
Next Cluster

約140 Servers

ARISTA



Clos Topology

# Broadcom Tomahawk5スイッチを中心とした高密度GPUクラスタを実装

物理構成

論理構成

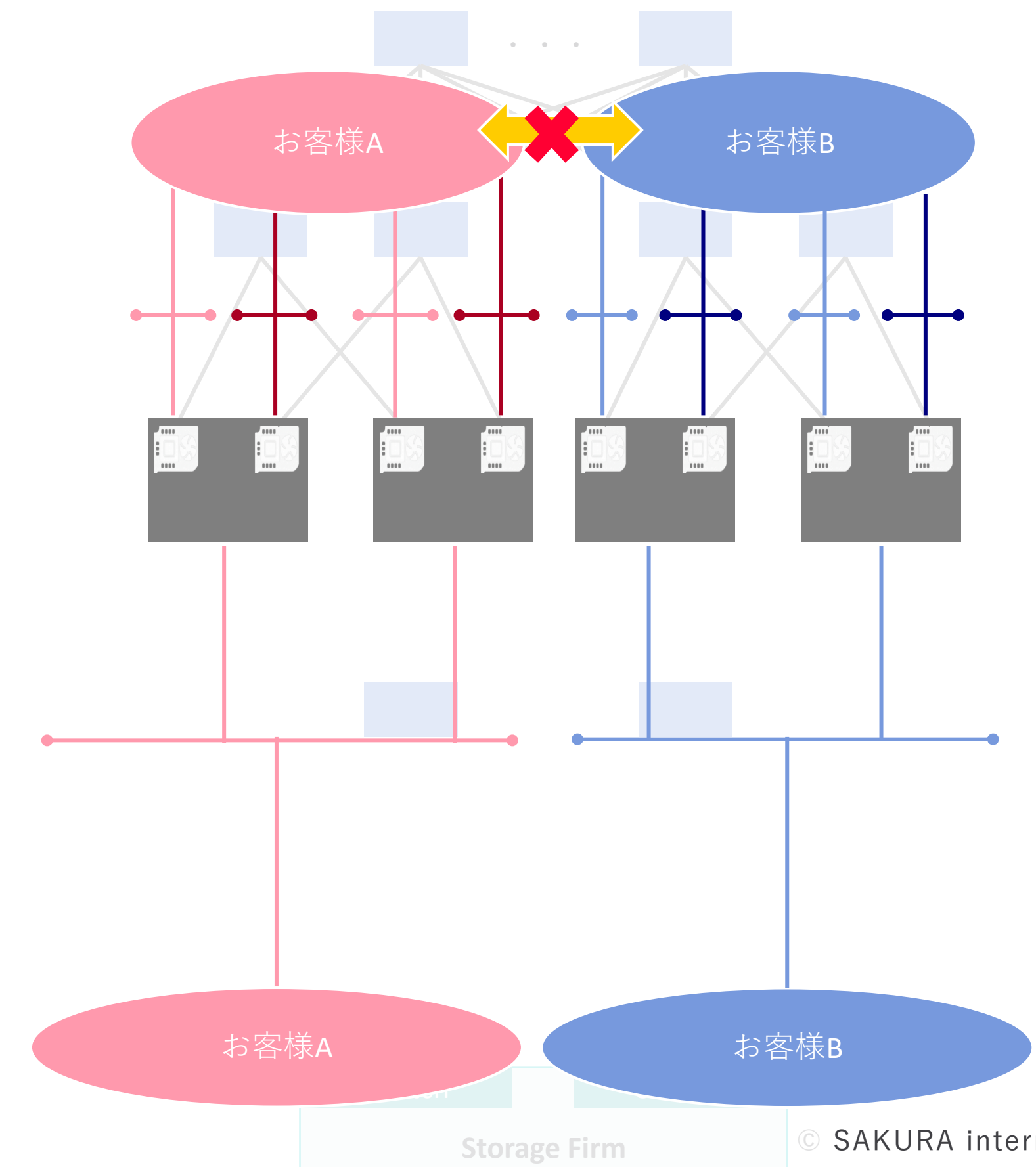
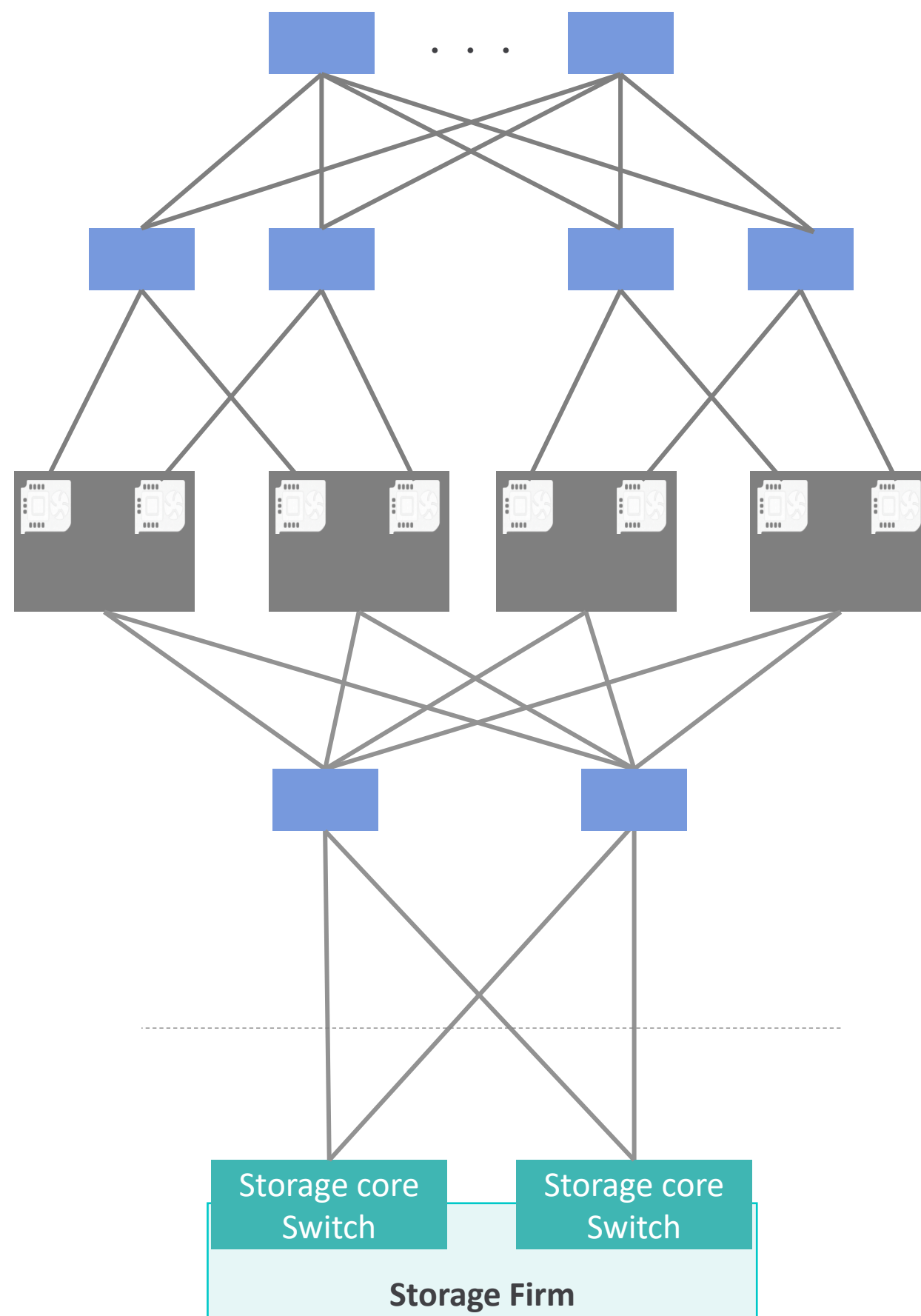
Spine 10 switch

Leaf 20 switch

HGX B200  
140 servers

Storage Edge  
8 switch

Storage Core  
2 switch



# Broadcom Tomahawk5スイッチを中心とした高密度GPUクラスタを実装

物理構成

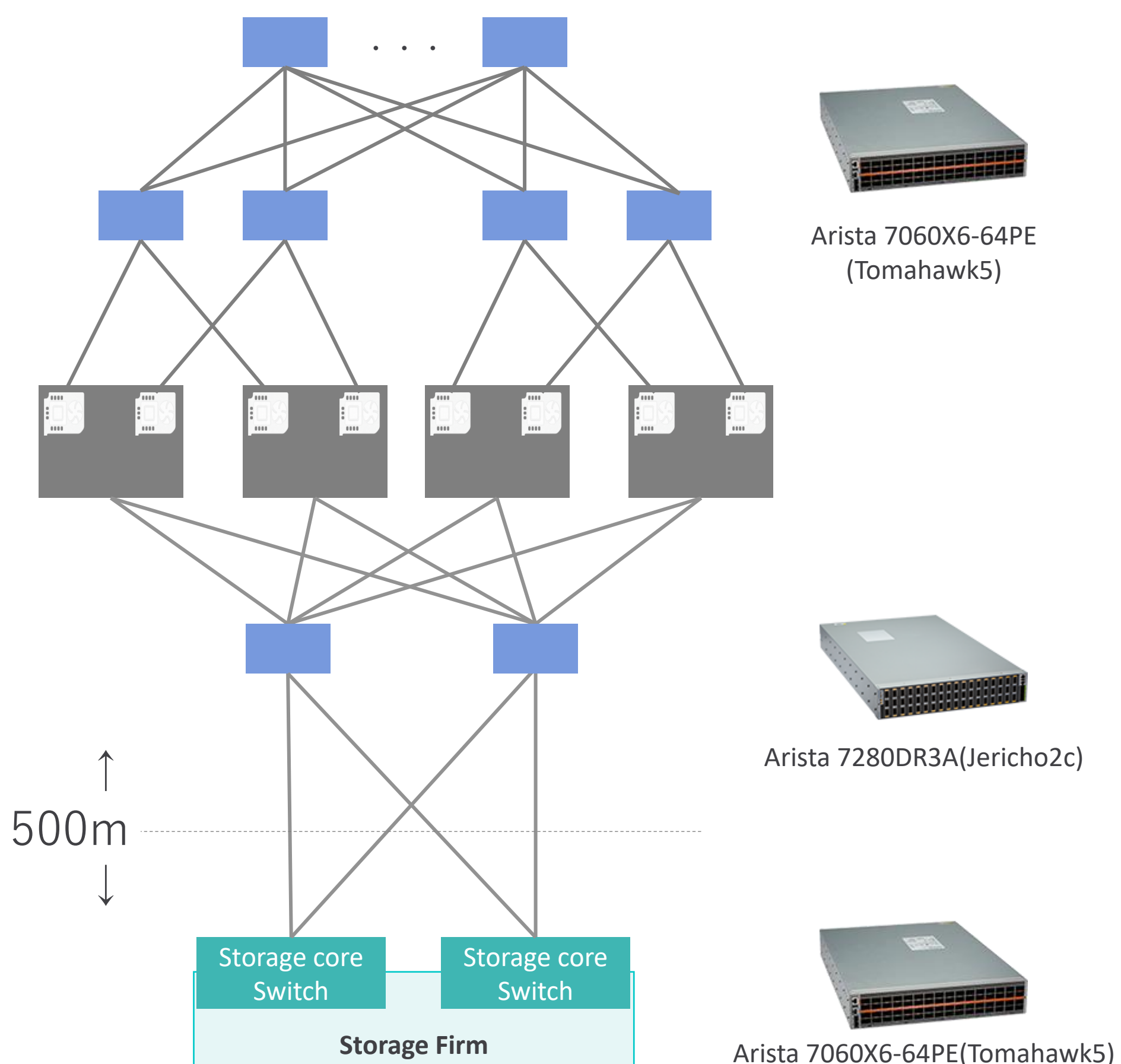
Spine 10 switch

Leaf 20 switch

HGX B200  
140 servers

Storage Edge  
8 switch

Storage Core  
2 switch

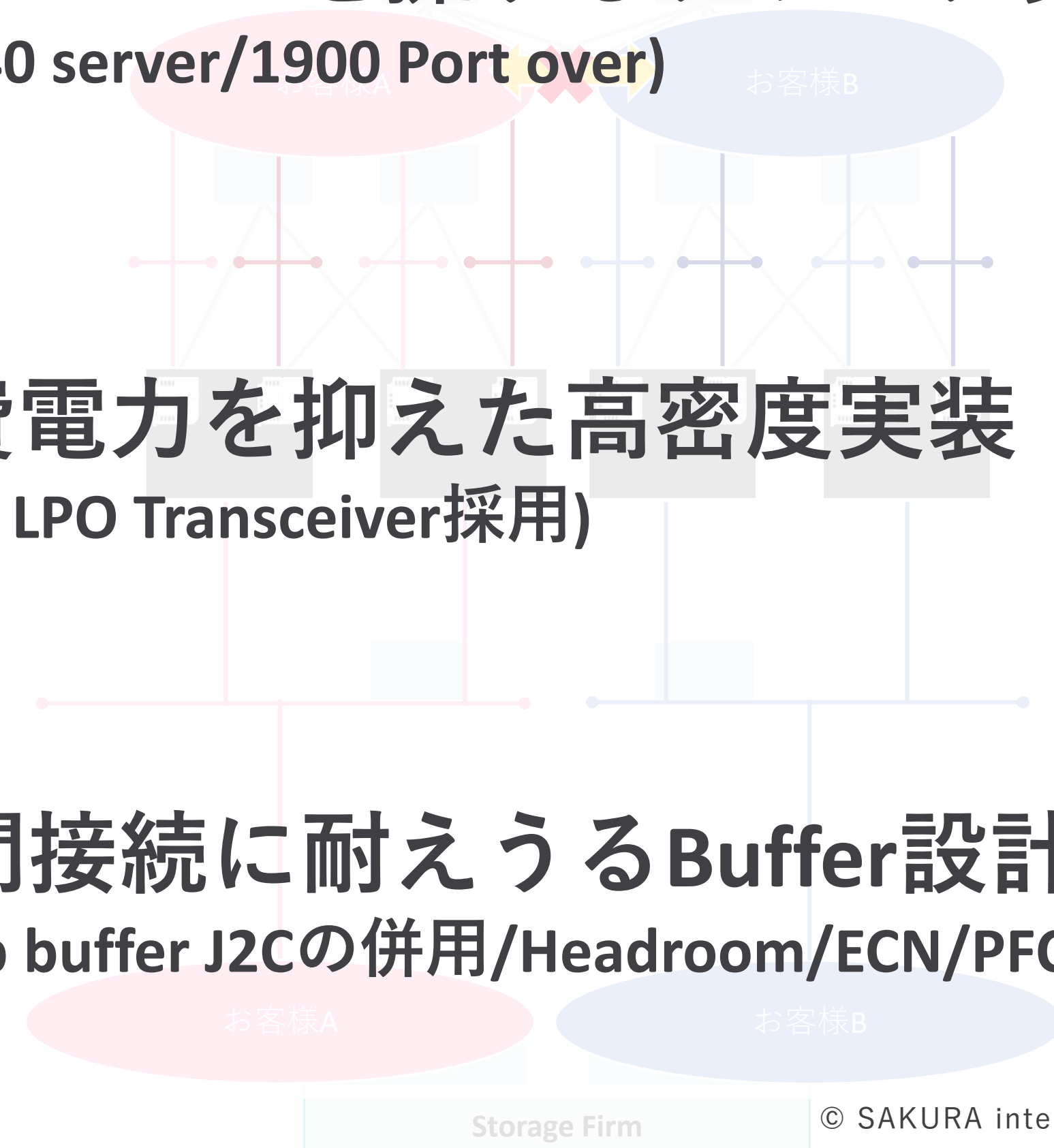


論理構成

約1100GPUを擁するクラスタ実装  
(約140 server/1900 Port over)

消費電力を抑えた高密度実装  
(800G LPO Transceiver採用)

棟間接続に耐えうるBuffer設計  
(Deep buffer J2Cの併用/Headroom/ECN/PFC)



# Broadcom Tomahawk5スイッチを中心とした高密度GPUクラスターを実装

物理構成

論理構成

無数のお客様同士の分離  
VRF,EVPN/VXLANによるマルチテナンシー

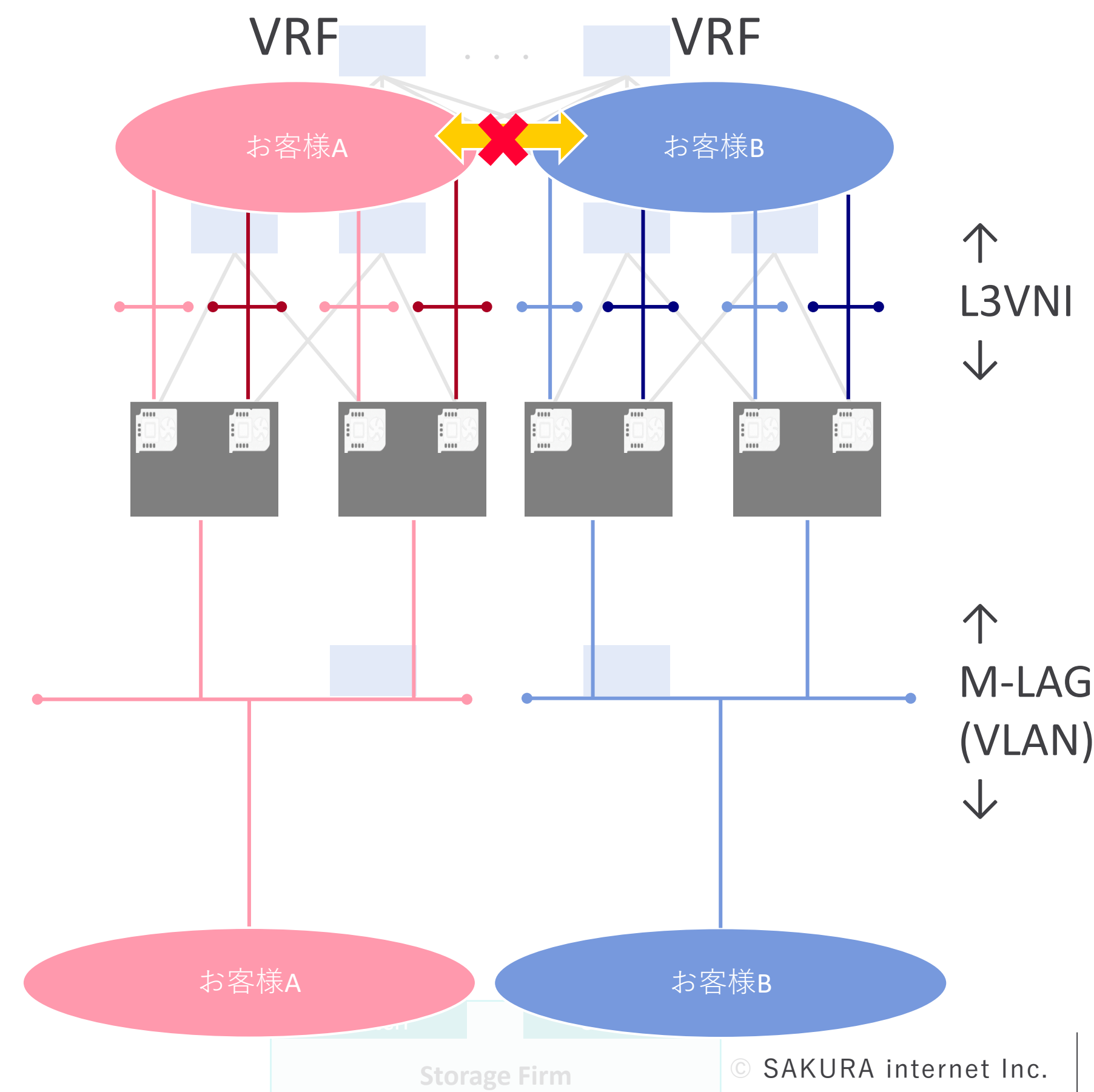
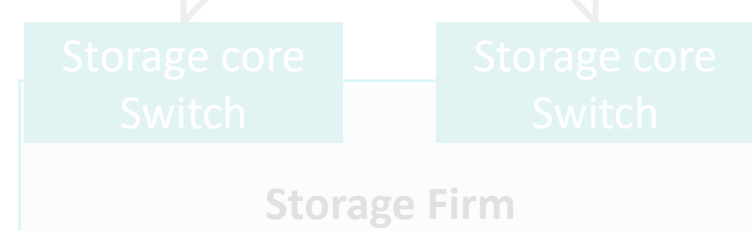
Leaf 20 switch

複雑なVLAN延伸を回避  
L3VNIを用いたシンプルなL3VPN化

Storage Edge  
8 switch

アプライアンス要件との両立  
L2/L3を問わないM-LAG構成

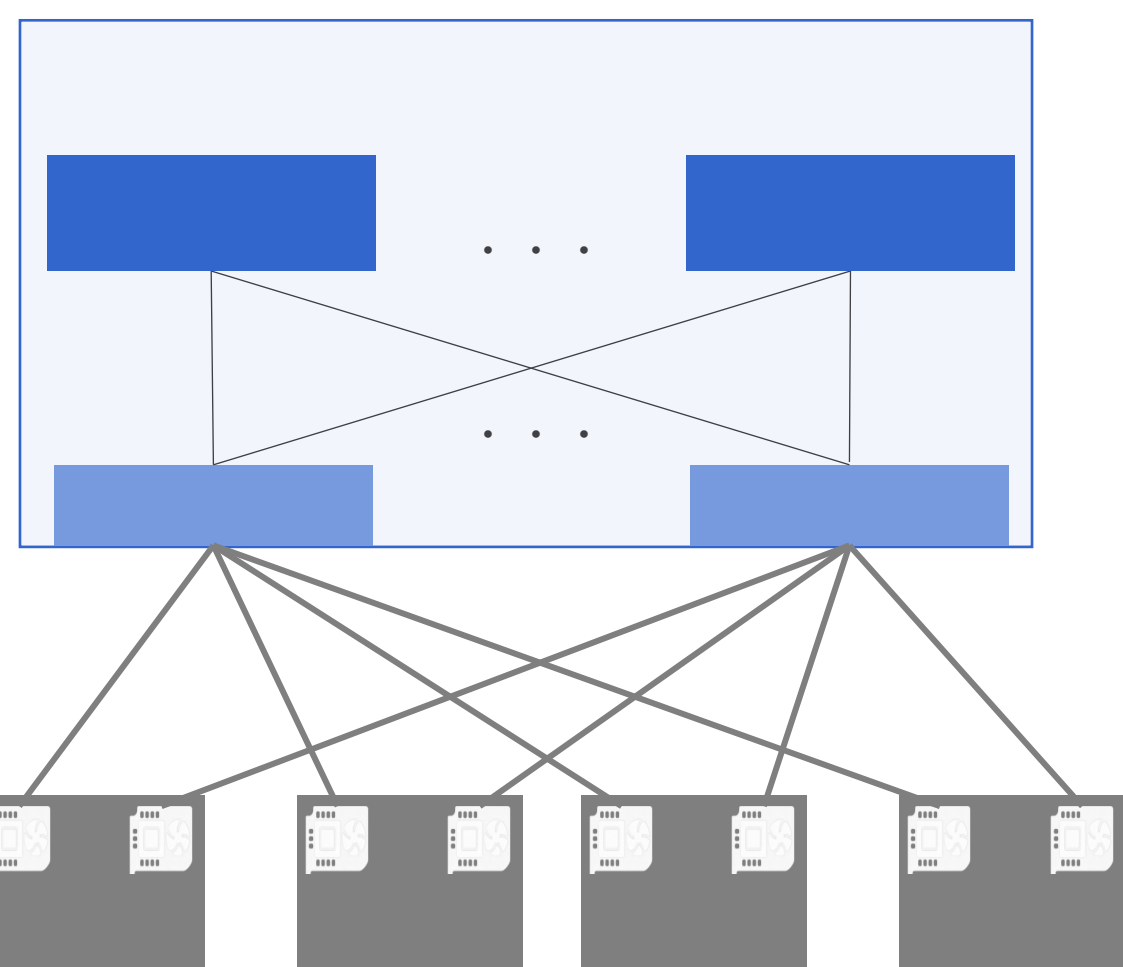
Storage Core  
2 switch



# TopologyとNOSの二軸で新規設計を訴求

## Architecture

収容上限突破を目的としたClos Topology検討



Chassis Switch

Clos Topology

## Network OS

デリバリー速度と中立的な技術の採用



ARISTA



SONiC  
Edge-core  
NETWORKS

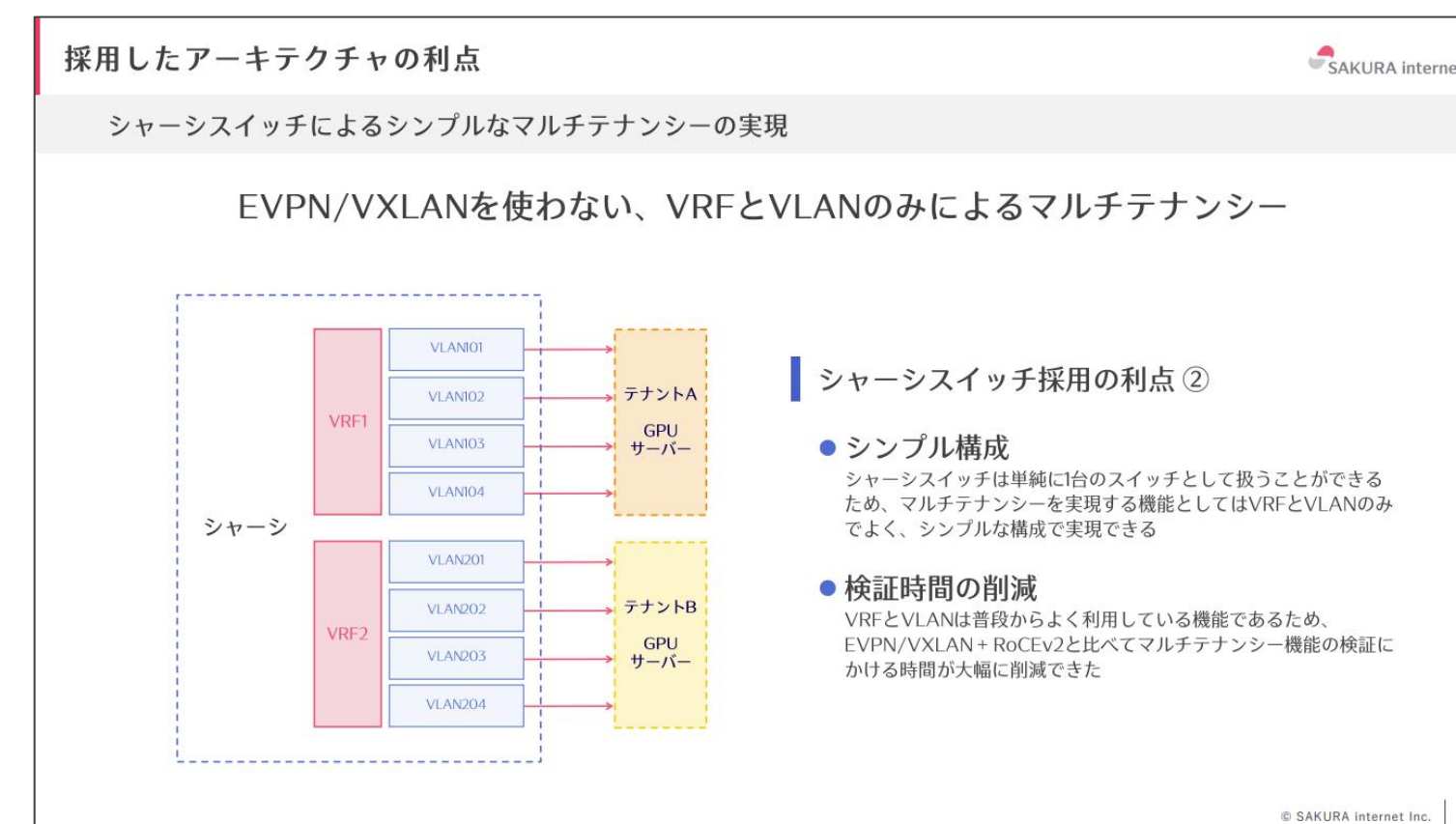
# 新たなアーキテクチャの試行

## 柔軟なデリバリーを実現するためのClos Topology採用

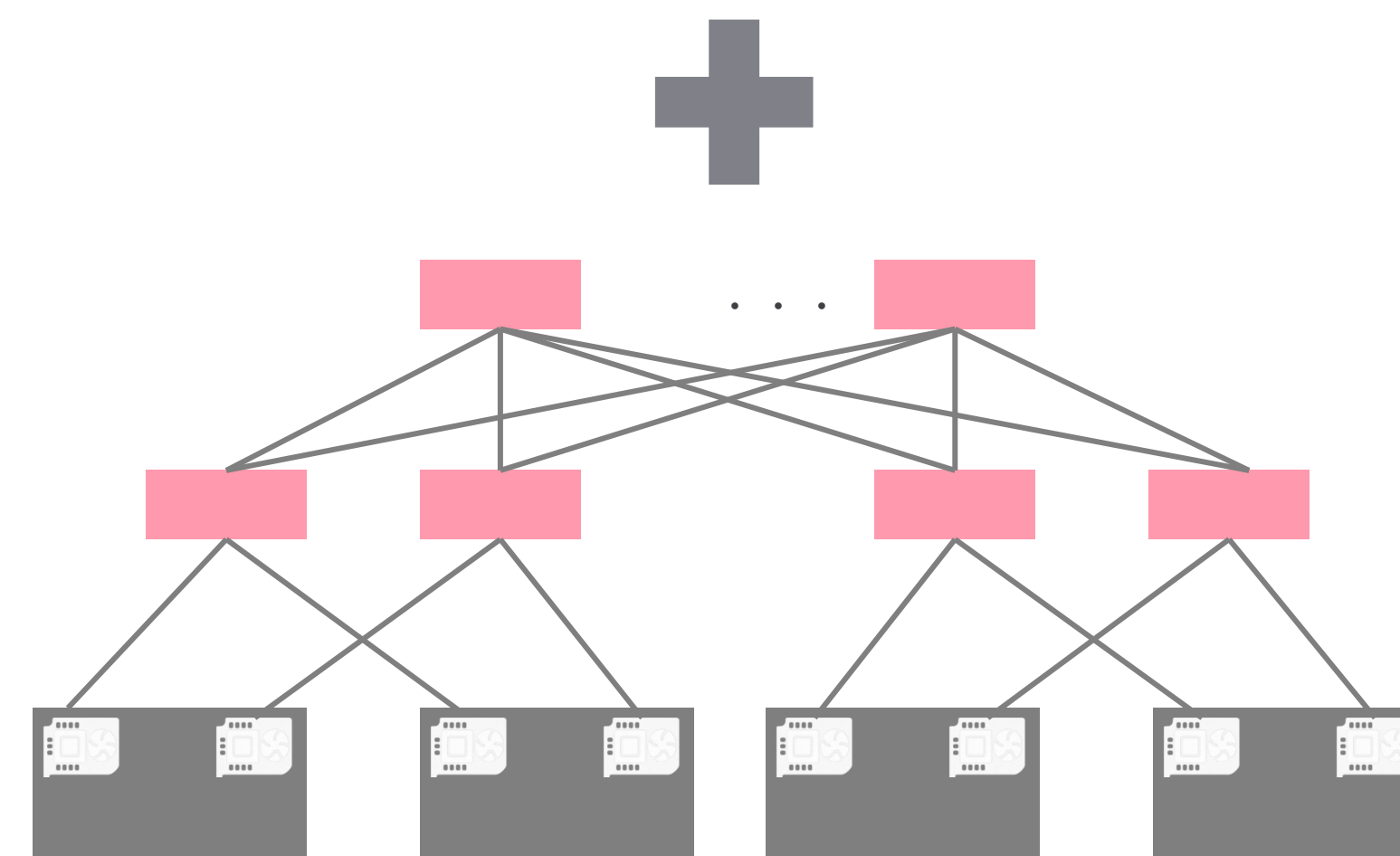
- クラスタ規模の増大  
→ 1台のシャーシに**収容するにはポート不足**
- GPUの提供タイミングに確実に調達できる必要  
→ シャーシ型スイッチは**選択肢が少ない**
- GPUの供給に合わせたネットワーク検討が必要  
→ ポート数の変化に**柔軟に対応**したい



- GPUクラスタの規模に合わせた**使い分け**ができる状態を目指す  
72台以下の小規模環境 → シャーシ  
それ以上の大規模環境 → Clos Topologyによる収容増

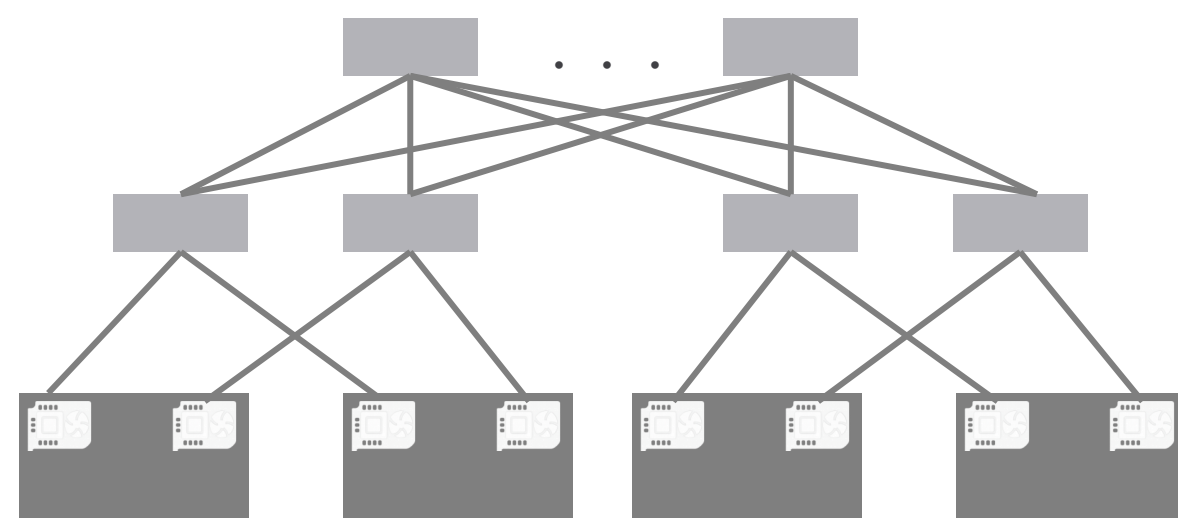


JANOG54 生成AI向けパブリッククラウドサービスをつくってみた話

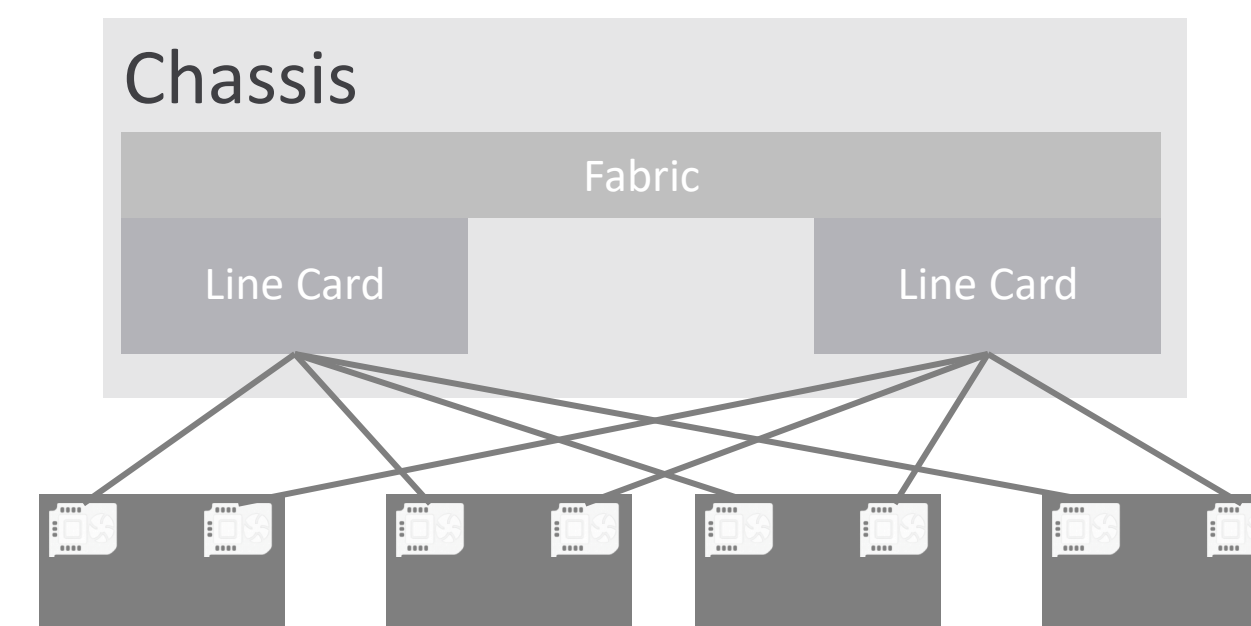


## 収容効率と柔軟性を目的としたClos Topology導入

Clos Topology



Chassis Switch



選択肢の多さ	多い	少ない
収容台数	Spine追加で <b>100台以上収容可能</b> ※ ケーブル・ユニット設計がシビアになるため、一定規模で上限あり	Line cardの枚数により <b>サーバー72台程度が上限</b>
管理対象	Spine/Leafを個々に管理	運用機器が1台だけ
スペース効率	低い（ケーブルやSpineによるoverhead）	<b>意外と省スペース</b>
構成技術	複数の技術を組み合わせる必要 BGP, EVPN/VXLAN(L3VNI or L2VNI)	VRF/VLANだけのシンプル構成
部品点数	<b>数百本単位</b> の スイッチ間トランシーバーが必要	筐体（Line CardとFabric）だけ

## GPUでは、“スケールしにくい”前提が揃いがち

### 一般的なClos Topology

- 柔軟なスケラビリティ
- Control Planeの分離による、障害・作業時影響の低減
- N+1台構成によるコスト低減

### GPU基盤における現実

- 余剰ポートの用意が物理的、経済的に困難  
→ **スケールしない**前提で設計（作り直し）
- 単一機器の瞬断でも、RDMAはFailする  
- Rolling updateによるSLA維持は許されない
- GPU/400G Switchは高価なため、**N+0で使い切る**必要  
- N+1を用意してhot standbyするのは採算性が...
- Full bisection(uplink:GPU/1:1)のLossless構成  
- **スイッチ間トランシーバーとケーブルが倍増**

それでもなお  
**収容効率の向上が最大のモチベーションとなった**

# ホワイトボックスへの挑戦

## 従来のスイッチとは全く異なる期待感を持った導入

### Arista EOS

実績豊富な一体型スイッチ

HW/SW一貫による安定性

操作性の高さ

手厚いサポート実績

### SONiC(Whitebox)

サーバー運用のノウハウを活用



出典: SONiCで構築・運用する生成AI向けパブリッククラウドネットワーク @ SONiC workshop 2025

迅速なデリバリー

Linux由来の自由度

OSSを活用した実装

## 自由度の反面「実装力」「OS内部の理解」が求められる

### Arista EOS

自動化に必要な機能がOSSで公開

AristaによるOSS Libraryが公開



HW/SW一貫による安定性



操作性の高さ



手厚いサポート実績



### SONiC(Whitebox)

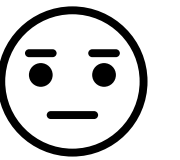
機能を自分たちで作り上げる重要性

GitHubでソースを読める



(人を選ぶ)

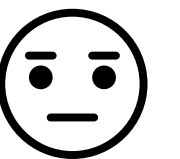
運用に必要なコマンドが未熟



迅速なデリバリー



Linux由来の自由度



OSSを活用した実装



# SONiCの運用に至るまで、内製で下記の機能追加を実施

## 運用コマンドの拡充



未成熟機能はLinux/Pythonを用いて自社スクリプト開発

隣接機器の確認

```
admin@gspine-008:~$ show lldp table
Capability codes: * Router, (B) Bridge, (O) Other
LocalPort RemoteDevice RemotePortID Capability RemotePortDescr
-----
Ethernet0 gleaf-201 Eth24(Port24) BR To_gspine-008
Ethernet8 gleaf-202 Eth24(Port24) BR To_gspine-008
Ethernet16 gleaf-203 Eth24(Port24) BR To_gspine-008
Ethernet24 gleaf-204 Eth24(Port24) BR To_gspine-008
```

Port? Interface? = 混乱  
Descriptionは要らない

configの差分確認

```
admin@gspine-001:~$ sudo diff startup-config.log running-config.log
(Sort違いにより大量の差分)
```

本当の差分はどこにある？

© SAKURA internet Inc.

LLDP/BGPの情報を整理

```
admin@gspine-008:~$ sakura-neighbor
LocalIF Port Remote hostname RemoteIF Port BGP state BFD state
-----
Ethernet0 Port1 gleaf-201 Ethernet184 Port24 Established Up
Ethernet8 Port2 gleaf-202 Ethernet184 Port24 Established Up
Ethernet16 Port3 gleaf-203 Ethernet184 Port24 Established Up
Ethernet24 Port4 gleaf-204 Ethernet184 Port24 Established Up
```

SONiCの設定を横串で差分比較

```
admin@gspine-008:~$ sakura-config-compare
SUCCESS: SONiC Running Configuration の取得が完了しました。
SUCCESS: SONiC Startup Configuration の読み込みが完了しました。
ALERT: SONiC Config に差分が検出されました！

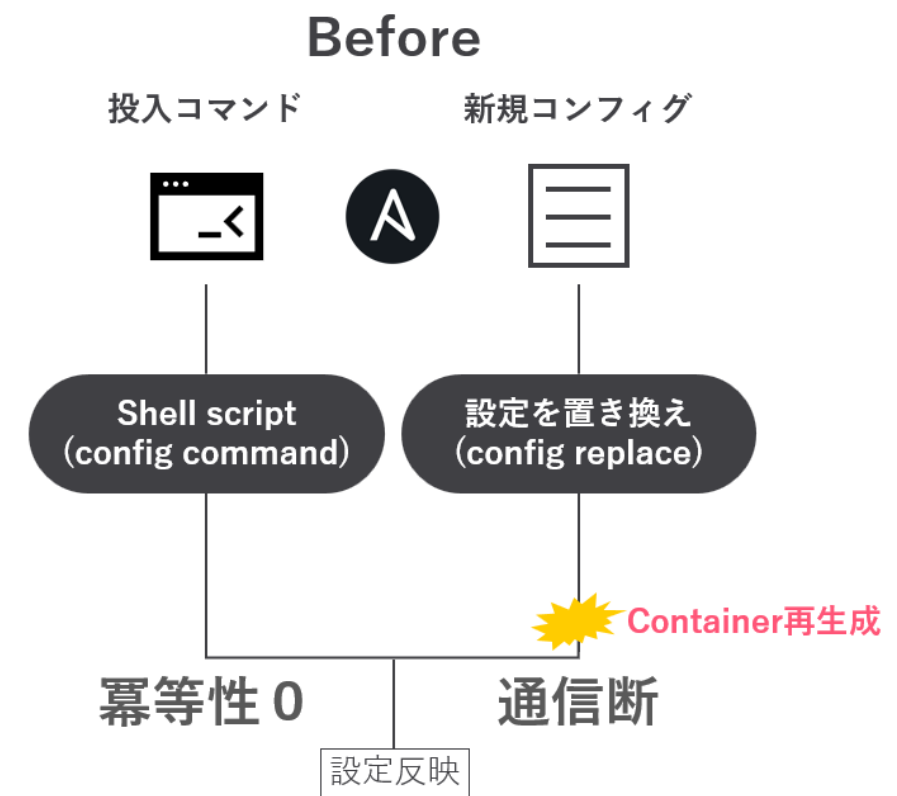
--- SONiC Config Running Config
+++ SONiC Config Startup Config
@@ -3860,7 +3860,6 @@
    "FLEX_COUNTER_STATUS": "enable"
  },
  "WRED_ECN_QUEUE": {
    "FLEX_COUNTER_DELAY_STATUS": "false",
    "FLEX_COUNTER_STATUS": "enable"
  },
}
```

© SAKURA internet Inc.

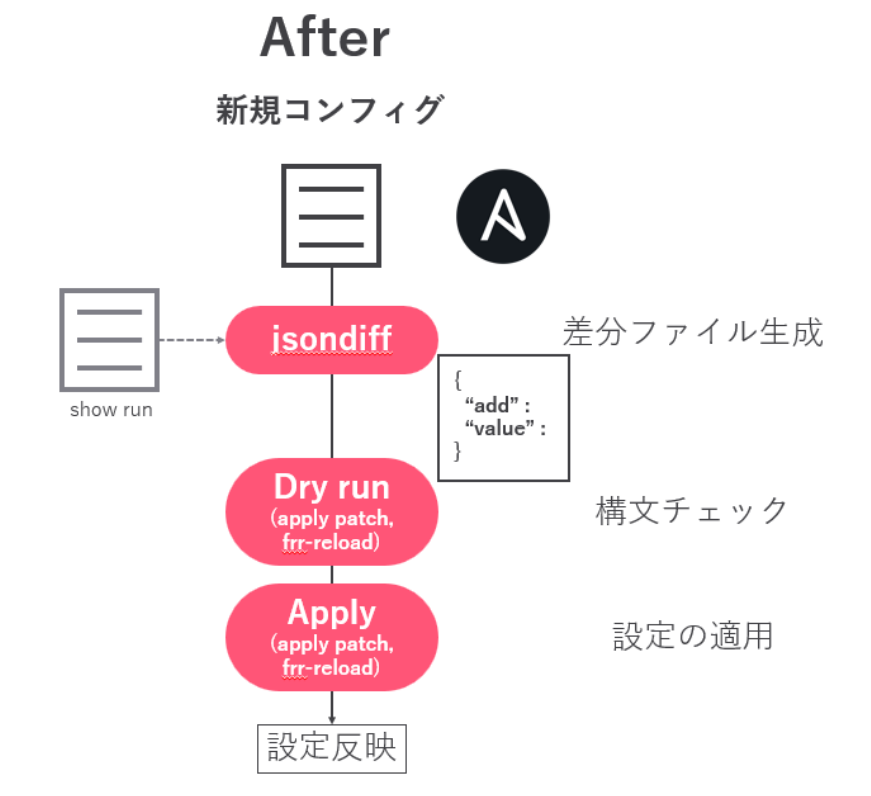
## 冪等性の実現



新規OS導入を機に、冪等性を担保した自動化ツール導入



© SAKURA internet Inc.



参考: SONiCのconfig apply-patchを試してみた | APRESIA Technical Blog

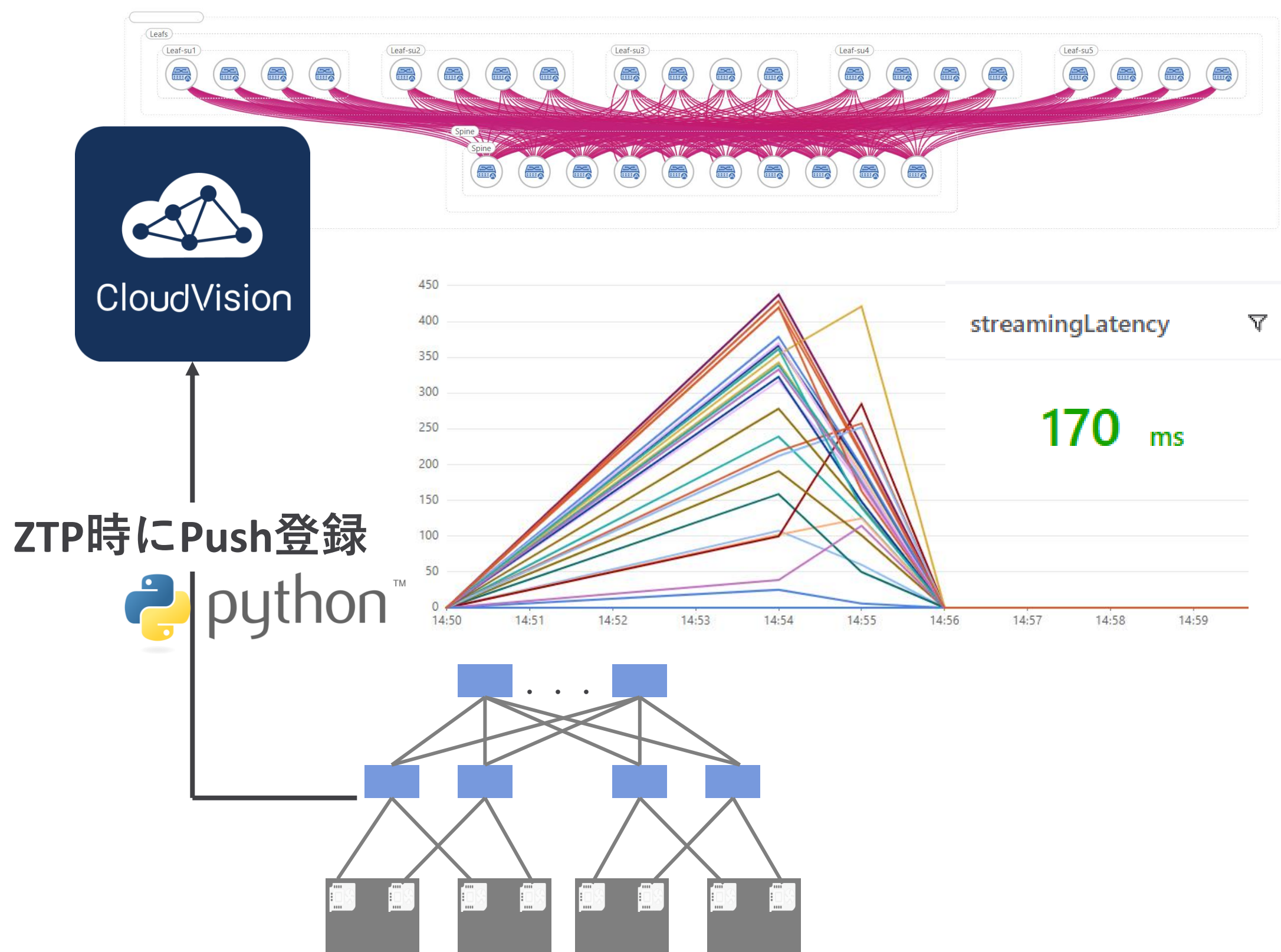
21

概要	内容
コンフィグの差分チェック	show   compare Likeな稼働中コンフィグとの差分確認を実装
LLDP neighbor情報の整理	show lldp neighbor likeなコマンドを自作した可視性向上
トランシーバー情報取得	TX/RXを一目で見るためのサマリチェック
ARP学習チェック	Rail単位でARP学習の正常性チェック
冪等性確保の対応	Dry-run実装/Configの冪等性対応 (Apply patch/frr-update.pyを改修)

# ネットワークOSの選択が、運用と監視の前提を変える

## Arista EOS

CloudVisionによる超高精度の監視

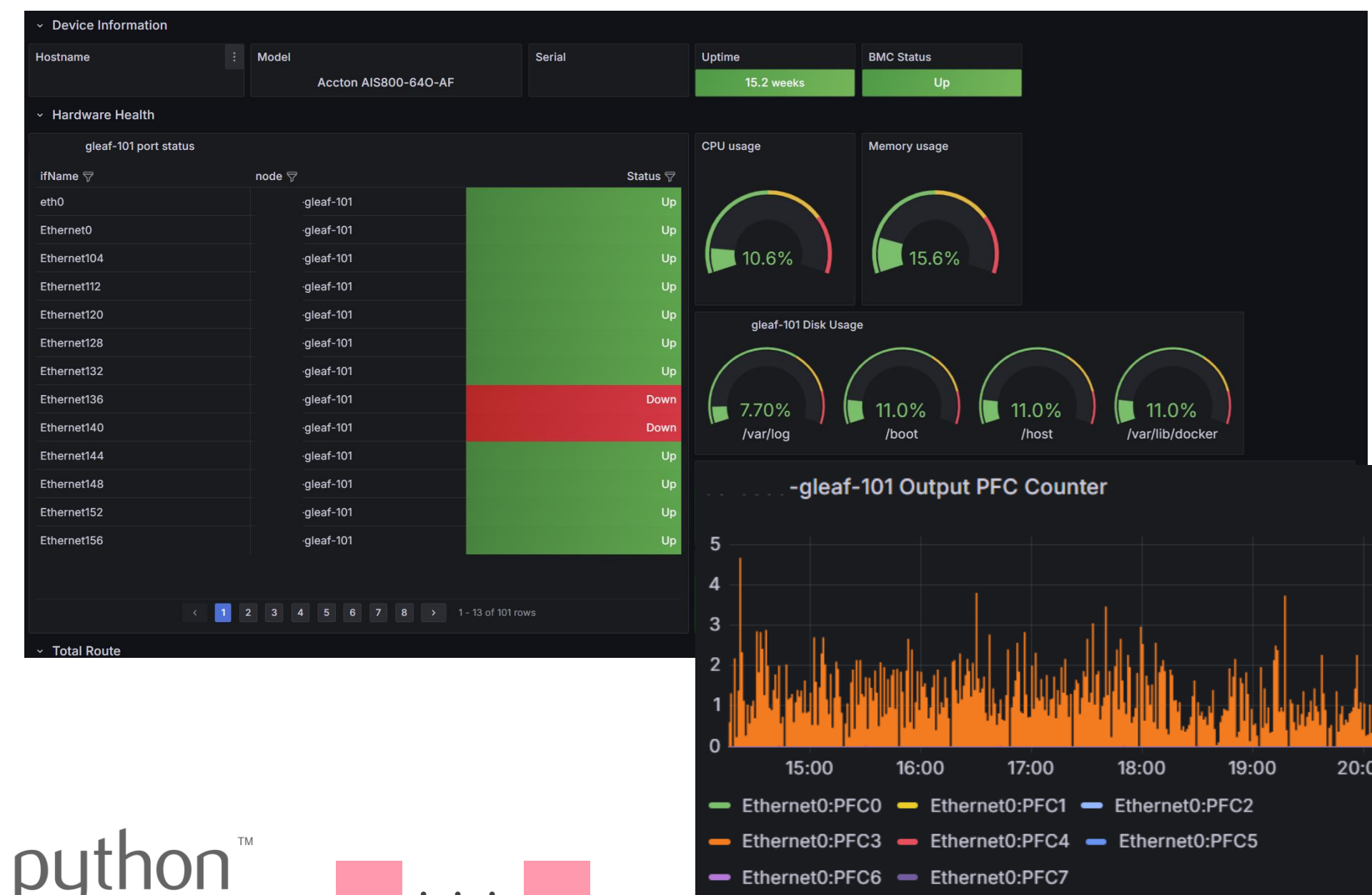


## SONiC(Whitebox)

OSSにより自分たちで作上げた監視基盤



AnsibleによるInventory管理

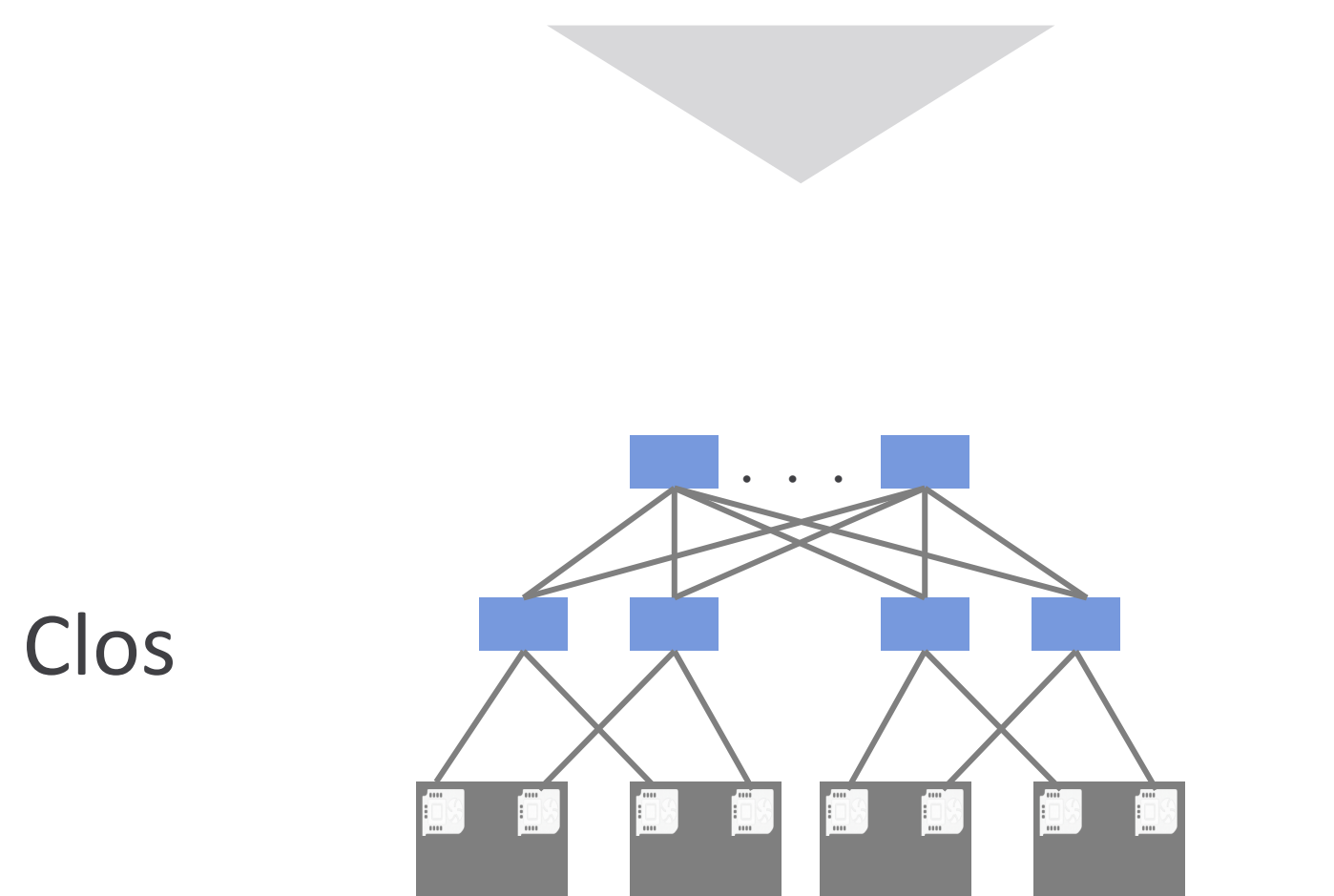
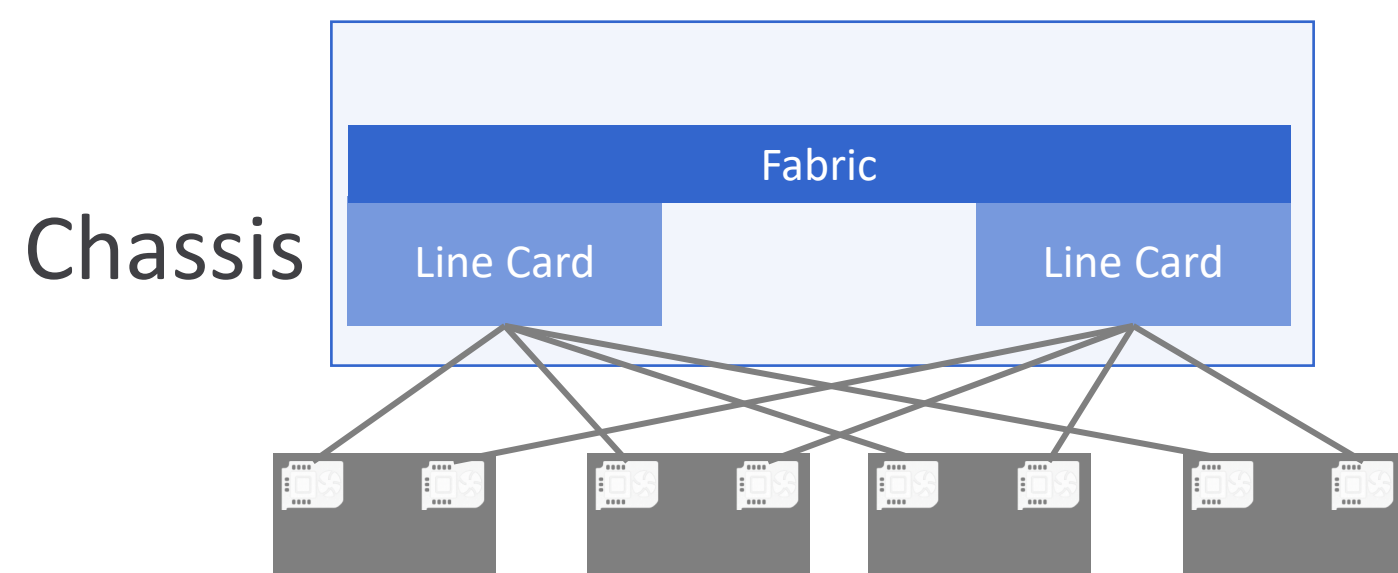


Telemetryへの本格対応が残課題

# 自動化への取り組み

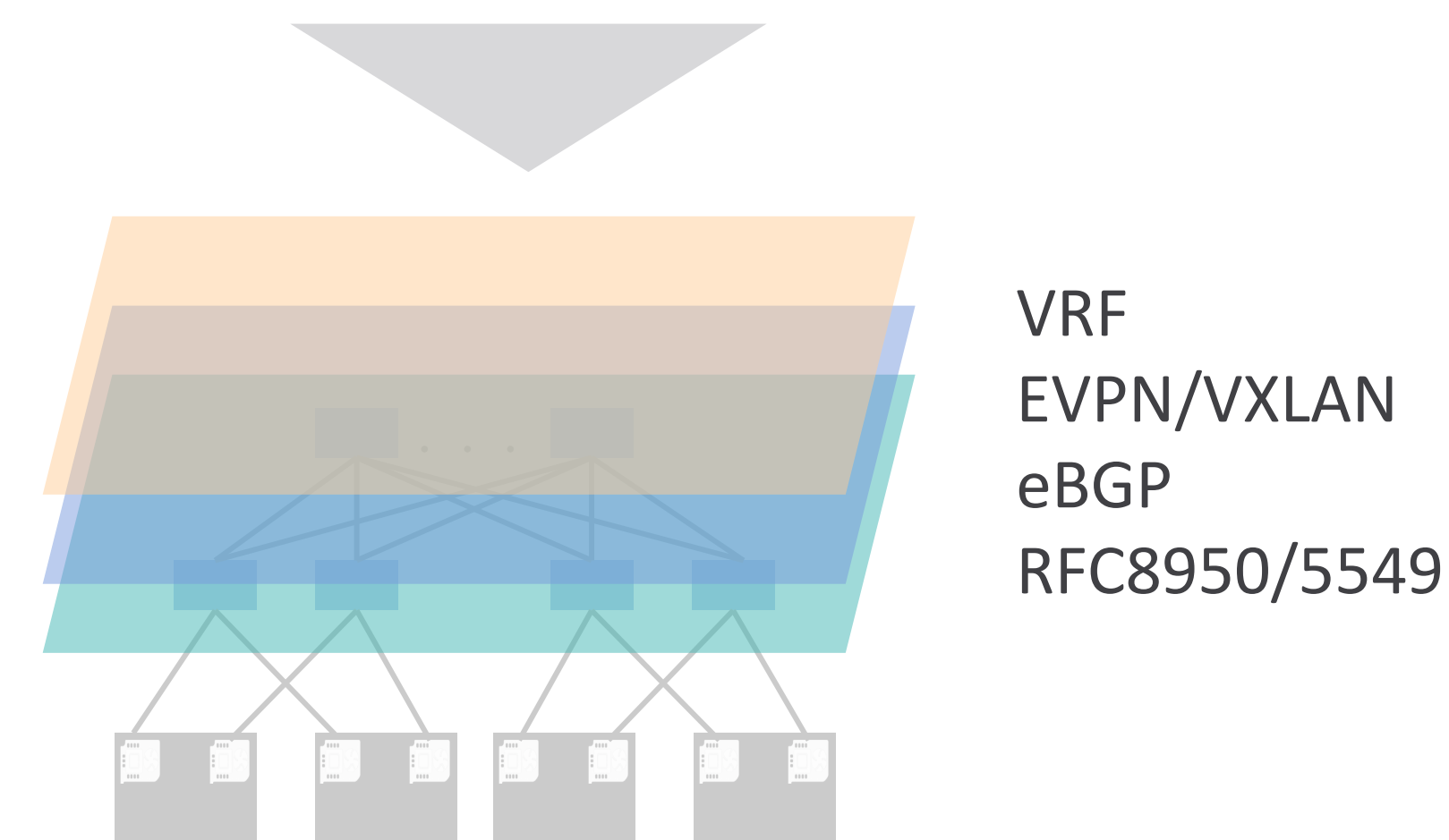
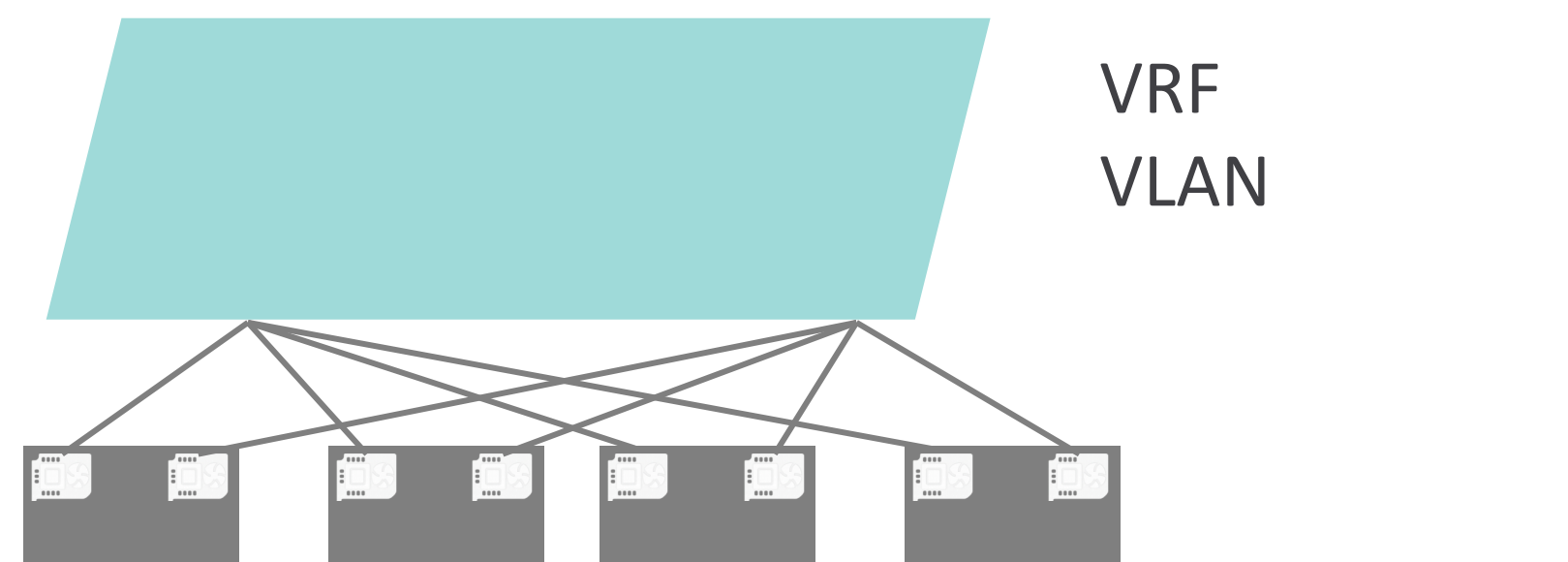
# 大規模クラスタにおいて“人が設定する”運用が現実的ではない

## スイッチの台数増加



クラスタにつき1台  
→ クラスタ当たり**30台**

## マルチテナンシーの複雑化



単一の構成  
→ **構成技術・パラメータの増大**

## 規模感の増加

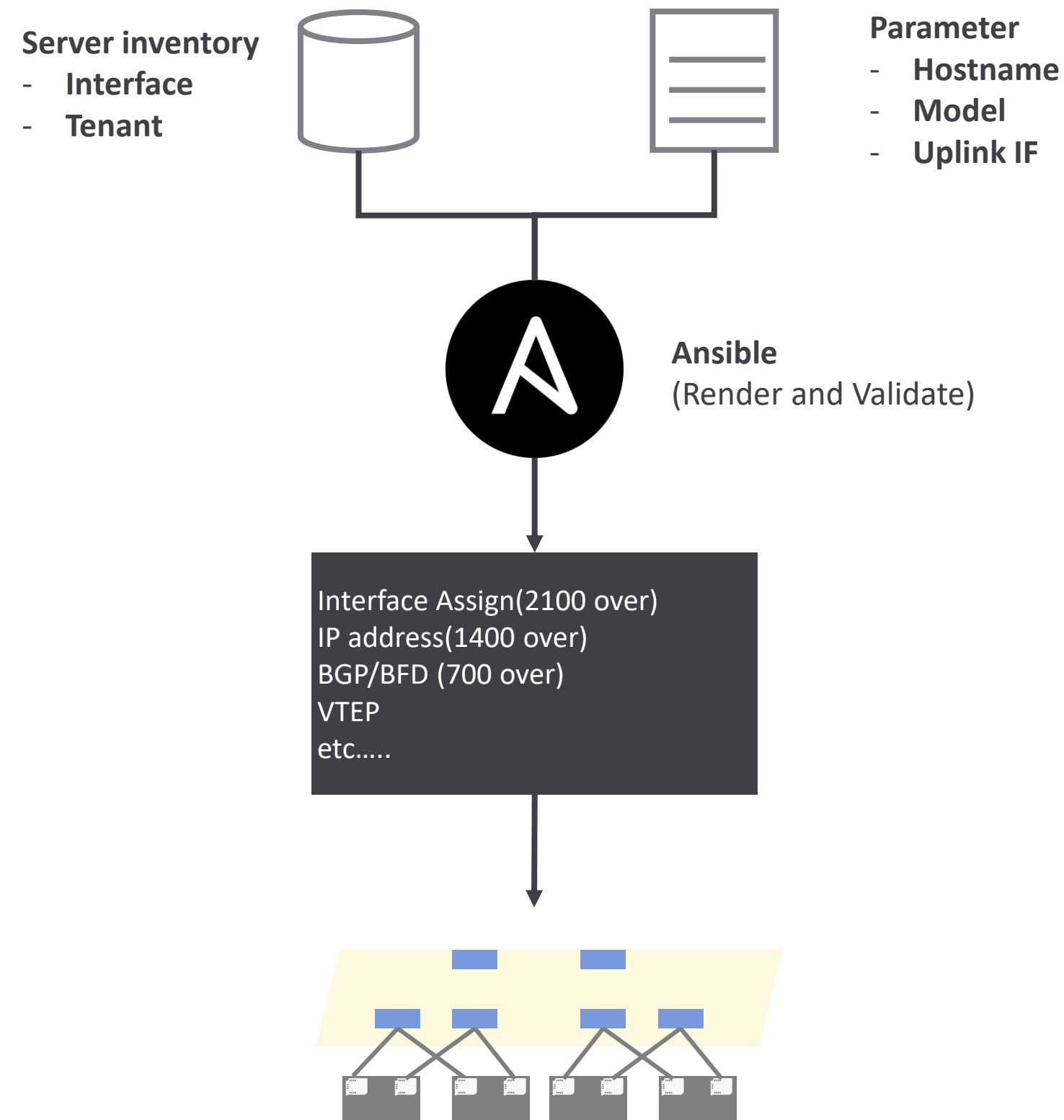
50 servers  
400G 400 Interface

約**140 Servers**  
400G 1400 Interface  
800G 572 Interface

スイッチ間IFの増加  
→ **Interface数1900over**

## Ansibleによる自動化

GPUサーバーの情報から設定を生成する仕組み



全台セットアップを**30分**で実現

## Git中心の構成管理

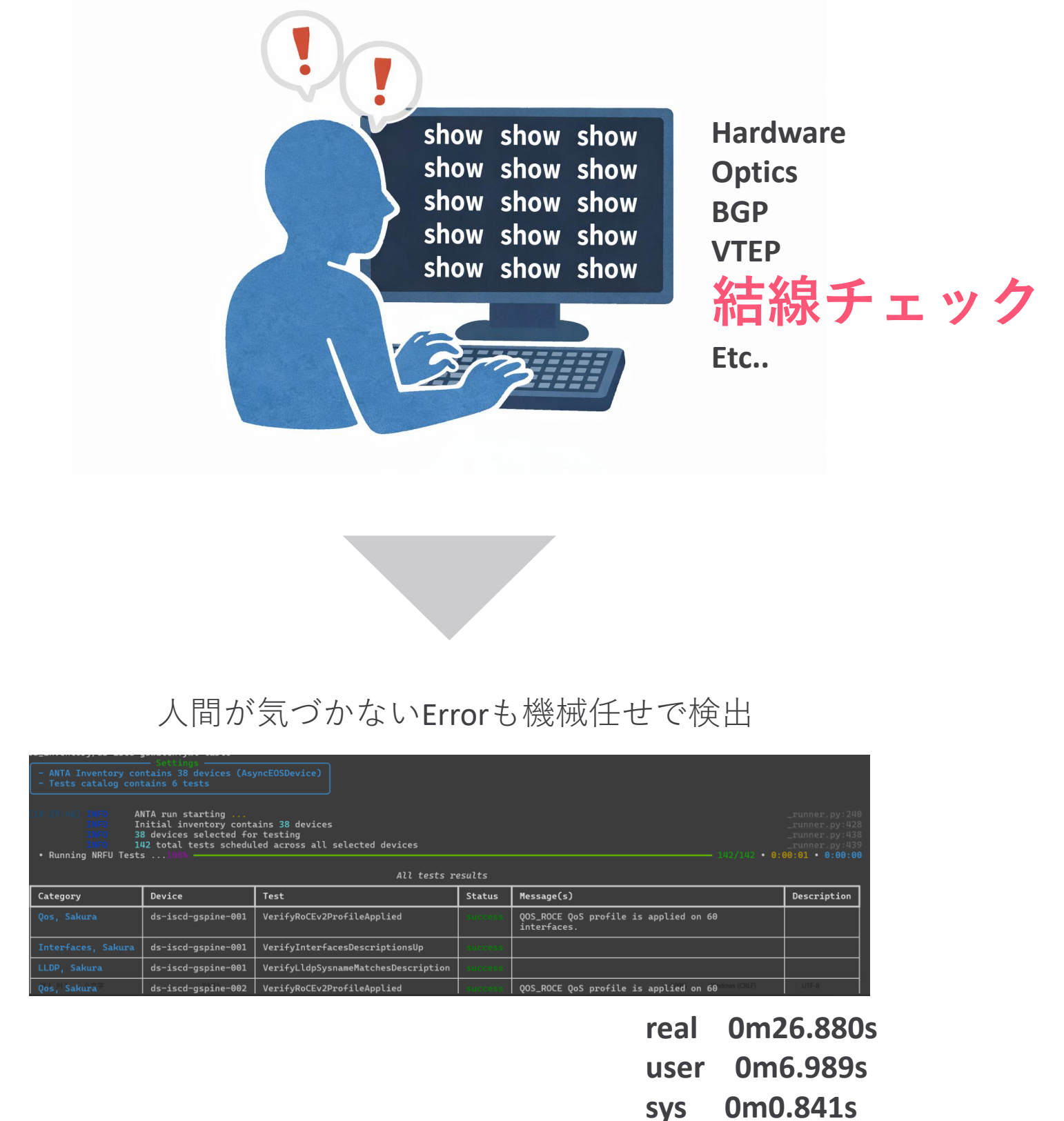
コードベースの構成管理をNWに導入



Bugの混入タイミングを可視化

## テストの自動化

ANTAを元にした自作テスト開発※



**1000項目**の試験を**30秒**で完了

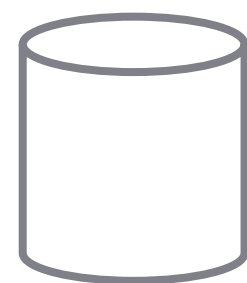
## サーバー情報と最低限のパラメータからconfigを生成

### Data source



#### Switch Parameter

- Hostname
- Model
- OS Version
- Uplink Interface



#### Server inventory

- NIC layout
- Tenant assign

```
ServerA:  
nics:  
- name: NIC1  
  alias: ens1f0  
  peer_device: gleaf-101  
  peer_interface: Ethernet1/1
```

```
Tenant_A:  
vni: 101  
servers:  
- ServerA  
- ServerB
```

### Transform

#### BGP

- Hostnameを元にしたASN算出
- Uplink interfaceのneighbor設定

#### Switch単位で情報を再構築

- VRF/VNIの自動割当
- 全区間InterfaceのIP割り当て

```
interface Ethernet X/Y  
ip address A.X.Y.Z
```

X : Switch hostname  
    gleaf-101 → 11

Y : Interface

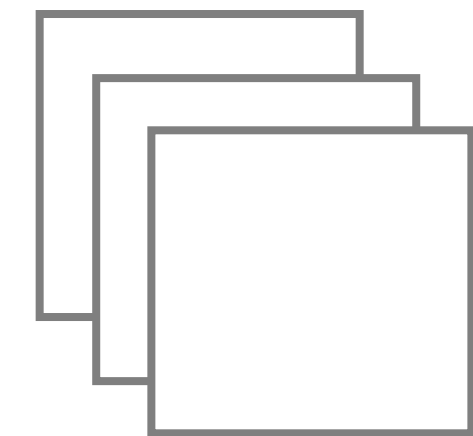
Z : SubInterface

Eth1/1 → 1.1  
Eth1/2 → 1.129

### Output

#### スイッチ 38台のconfigを生成

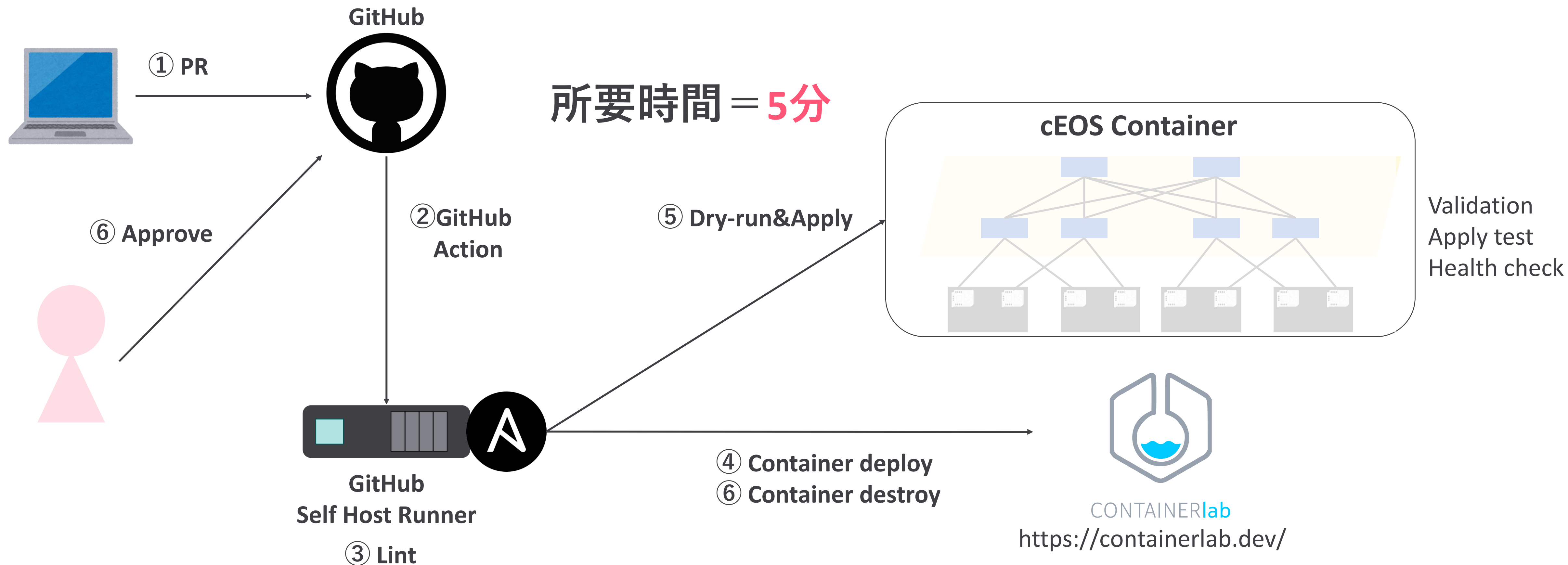
- Spine
- Leaf
- Storage Edge



```
$ wc -l * | grep total  
39640 total
```

# 約40,000行のConfigを自動生成

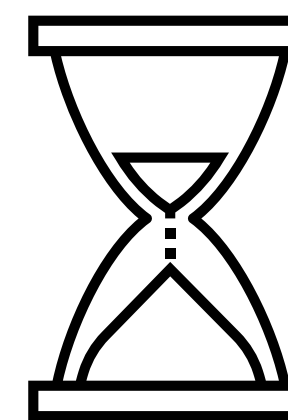
## 変更の都度、自動的にテストする仕組みを導入



検証サイクルの超高速化、短期間の検知を実現

これを人手でやったら、  
皆さんなら何日かかりますか？

私たちはOS導入から正常確認まで2週間かかりました



# まとめ

## さくらONEを支えるインフラ

複数の構成を使い分けることにより  
効率的なインフラ提供を実現

- Chassis Switch / Clos Topology
- EOS / SONiC

## 新たなアーキテクチャの試行

規模感に応じた  
Clos Topology/Chassis Switchの使い分け

GPUインフラ特有の課題にも直面

- **一般的なClos topologyとは異なる**性質
- Full bisectionに必要な**リンク数の増大**、  
管理コスト増加

## ホワイトボックスへの挑戦

SONiCの活用により、迅速なデリバリーや  
透明性の向上に対し、大きく貢献

- ハードウェア理解、時にはOSの  
ソースコードレベルの**内部理解**が必要
- **運用コストや、必要なスキルセット**  
も踏まえた選定が、重要となる

## 自動化の取り組み

構成要素の増加により、  
自動化は**楽するものから不可避な存在**に変化

結果として、迅速なデリバリーを実現

- OSSを活用した自動化の推進
- GitHub/Ansibleを中心にした設定自動化
- CIの導入による検証サイクルの短縮

Closを「理想的な構成」だと思っている人

→ 規模感に応じた使い分けが必要

新たなベンダーへの挑戦はコスト削減策だと思っている人

→ 自社の開発コストやあるべき姿も踏まえた選択

自動化は“楽をするため”だと思っている人

→ 自動化しなければ生き残れない

**大切なのは“自分たちで選び続ける”こと**

## アーキテクチャの選定基準

- ・ Closを採用することによって「大変になること」をどのように感じましたか？
- ・ 皆さんならシャーシとClosどちらを選択しますか？
- ・ どのような判断基準で選びますか？

## GPU基盤におけるマルチベンダーの検討

- ・ 今回はArista/SONiCの一例でしたが、皆さんはどのようなマルチベンダーを検討していますか？
- ・ マルチベンダーで運用するため、どのような工夫をしていますか？
- ・ SONiCを「自分たちで育てた」物語はどのように映りましたか？

## 増大する運用コストへのアプローチ

- ・ 自分たちで内製、自動化を突き詰める以外の解決アプローチはありますか？
- ・ 大量のInterfaceと、どのように戦っていますか？

**皆さんが同じ立場なら、どう“判断”しますか？**



「やりたいこと」を「できる」に変える

# Appendix

# さくらONEのGPUインフラ全体

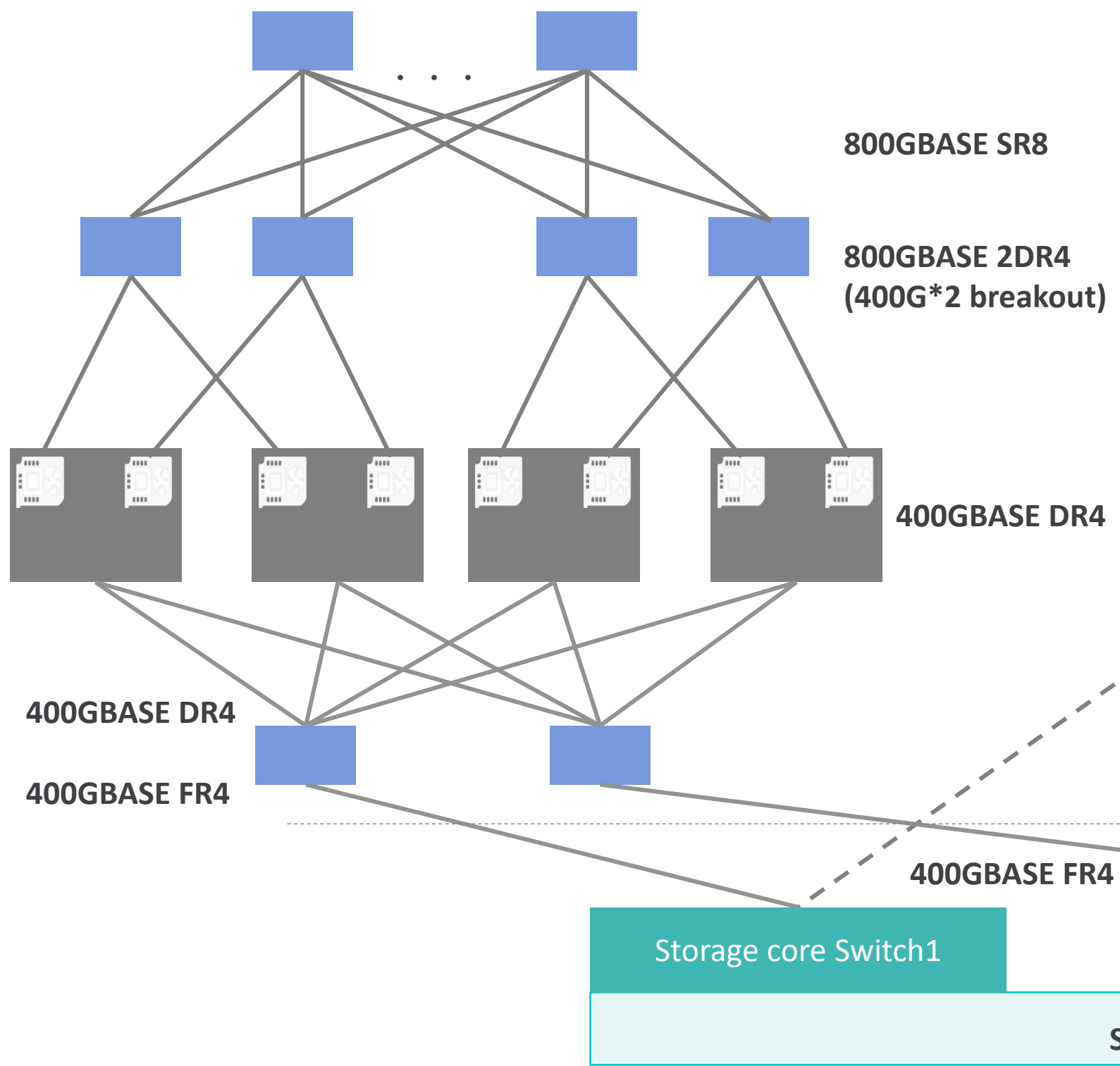
## B200 GPU クラスタ



Spine/Leaf :  
7060X6-64PE(Tomahawk5)



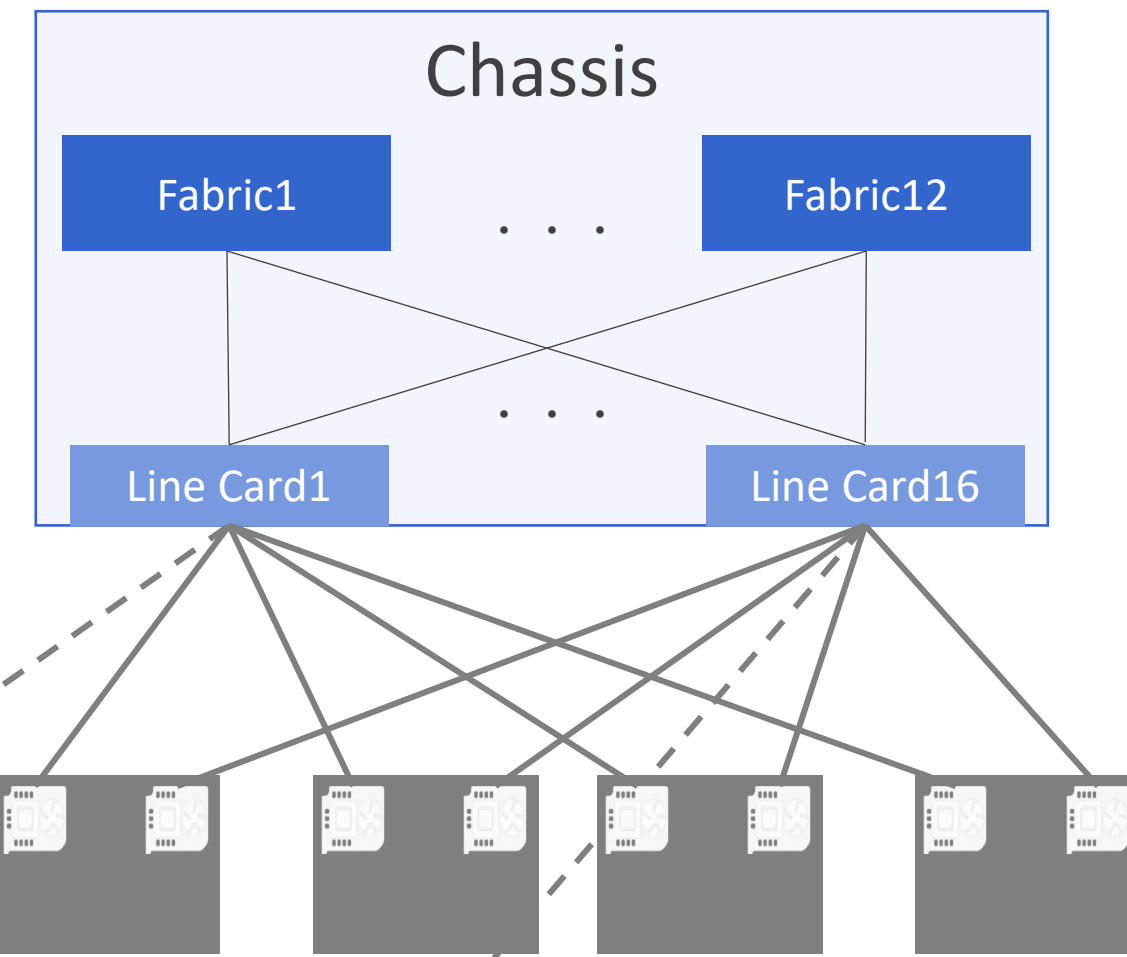
Storage:  
7280DR3A(Jericho 2c+)



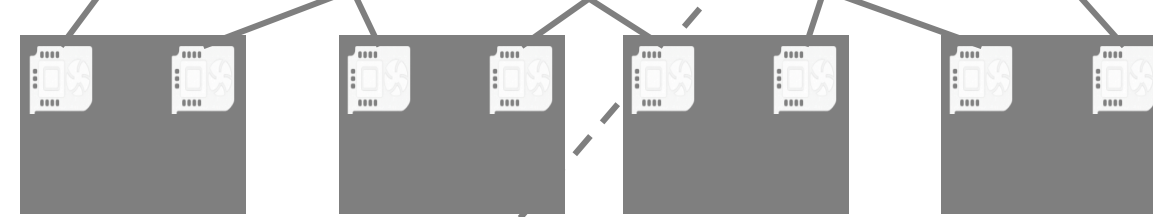
## H200/B200 GPU クラスタ



Chassis Switch:  
DCS-7816L-CH(Jericho 2c+)



400GBASE FR4

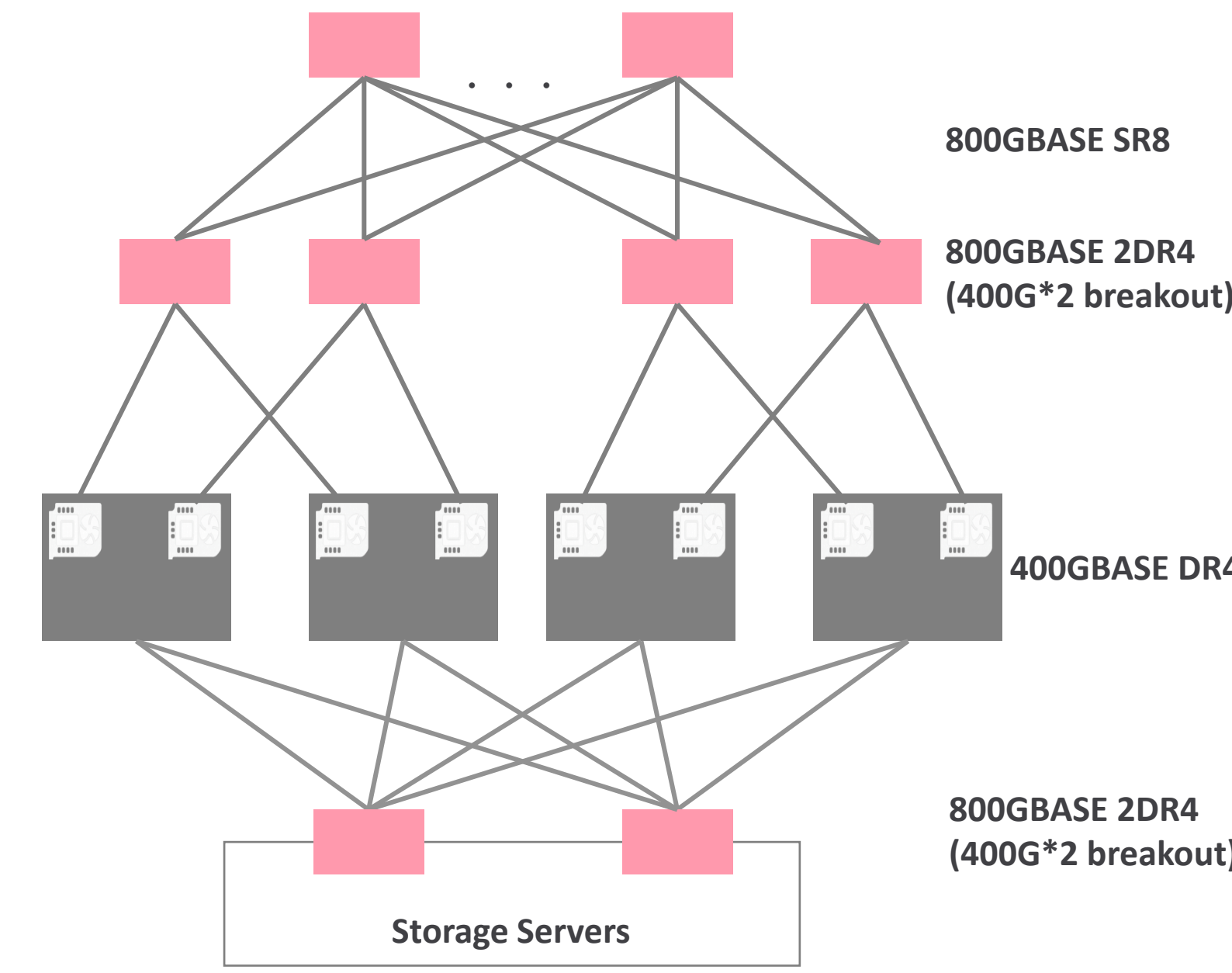


↑  
500m程度の  
棟間接続  
↓

## H100 GPU クラスタ



Spine/Leaf/Storage :  
Edgecore AIS800(Tomahawk5)  
+ SONiC



# Aristaシャーシを用いたGPUクラスタ設計

今回紹介しきれなかった構成は過去資料で紹介されています

## JANOG54 meeting

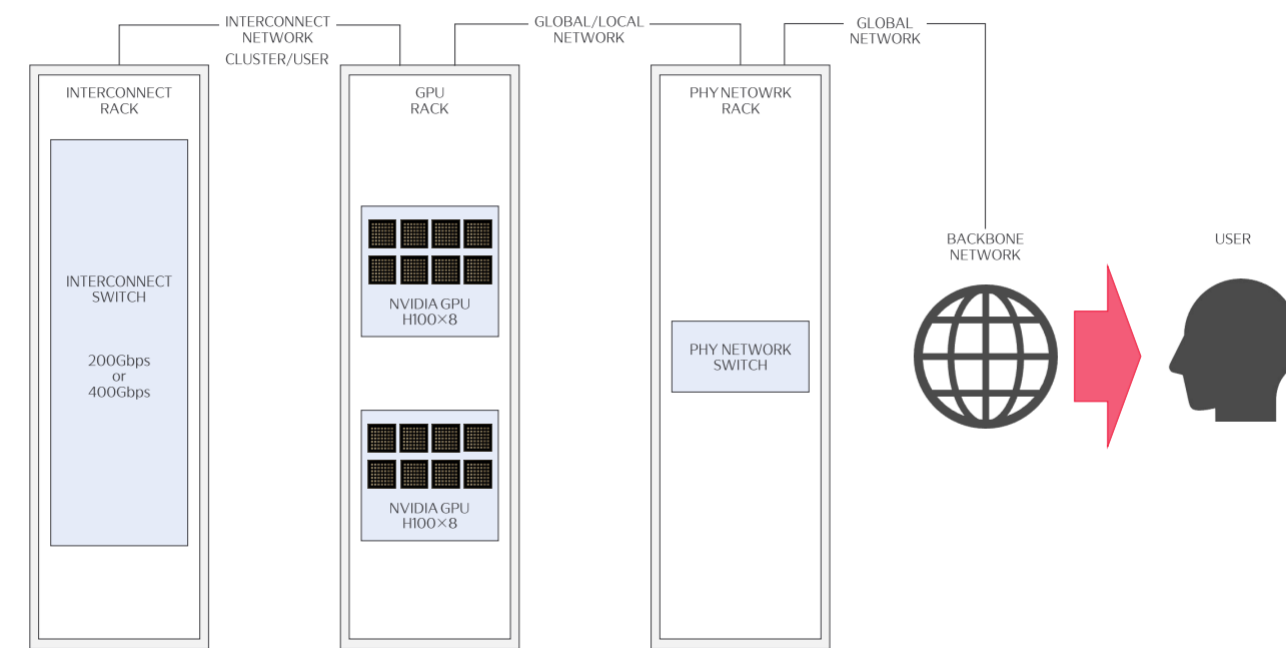
生成AI向けパブリッククラウドサービスをつくってみた話

### JANOG54 Meeting 生成AI向けパブリッククラウドサービス をつくってみた話

2024年7月4日  
さくらインターネット株式会社  
井上 壽祝  
高峯 誠  
平田 大祐



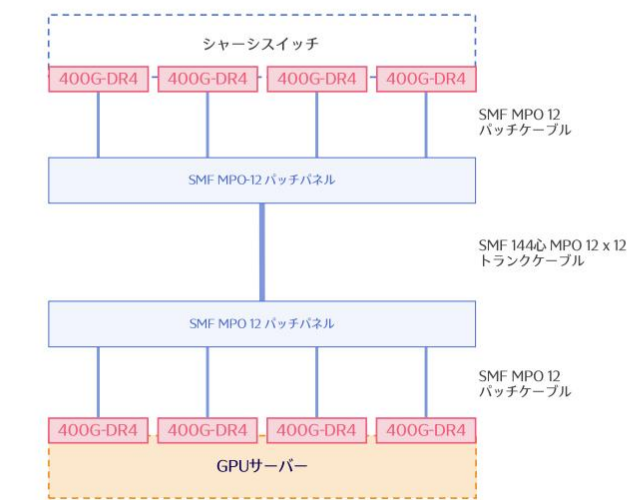
#### 全体像



© SAKURA internet Inc. | 11

#### 配線の設計

追加整備分では1サーバーあたり 400Gbs x 4 のネットワークに変更



#### 配線のポイント

- SMF MPO-12配線に変更
  - ・ 初回構築時より半年ほど経過した結果、サードパーティ品のOSFP-RHSがいろいろ選べるようになった
  - ・ 各種部材の納期が大幅に改善した



- いまのところ問題なし  
配線もシンプルになり、400G-DR4も安定している



© SAKURA internet Inc. | 35

#### スイッチの運搬と設置

Arista 7816R3のラック設置について

ハードウェアのインストール方法は動画などもあり詳細にまとめられていた



<https://www.arista.com/assets/data/pdf/Dataheets/7800R3-Quick-Look.pdf>

#### ラックマウントについて

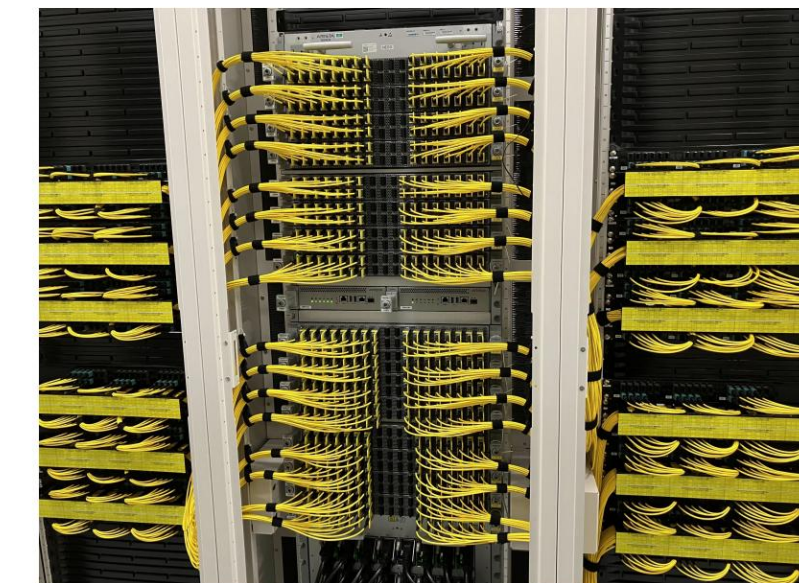
- ラック設置する際は推奨のツールがある  
ベンダー推奨のツールを探してみたが、国内では取り扱っている業者が無い様子 (スペックダウンモデルであれば取扱い有り)
- 自力でのラック搭載は不可と判断  
メーカー推奨の方法が取れないことから自社でのラックマウント作業は困難であると判断、専門の業者に依頼することに



© SAKURA internet Inc. | 28

#### 配線の設計

両隣のラックをパッチパネル専用ラックとして利用して配線する設計に



© SAKURA internet Inc. | 35

# SONiCを用いたGPUクラスタの構成

今回紹介しきれなかった構成は過去資料で紹介されています

## Arxiv

SAKURAONE: Empowering Transparent and Open AI Platforms through Private-Sector HPC Investment in Japan  
Fumikazu Konishi

## The Linux Foundation User stories

Open Networking at Scale: How SAKURA internet Deployed a TOP500 GPU Supercomputer with SONiC

## SONiC Workshop Japan 2025

SONiCで構築・運用する生成AI向けパブリッククラウドネットワーク

### SAKURAONE: EMPOWERING TRANSPARENT AND OPEN AI PLATFORMS THROUGH PRIVATE-SECTOR HPC INVESTMENT IN JAPAN

Fumikazu Konishi  
Research Center  
SAKURA internet Inc.  
Japan

SAKURAONE

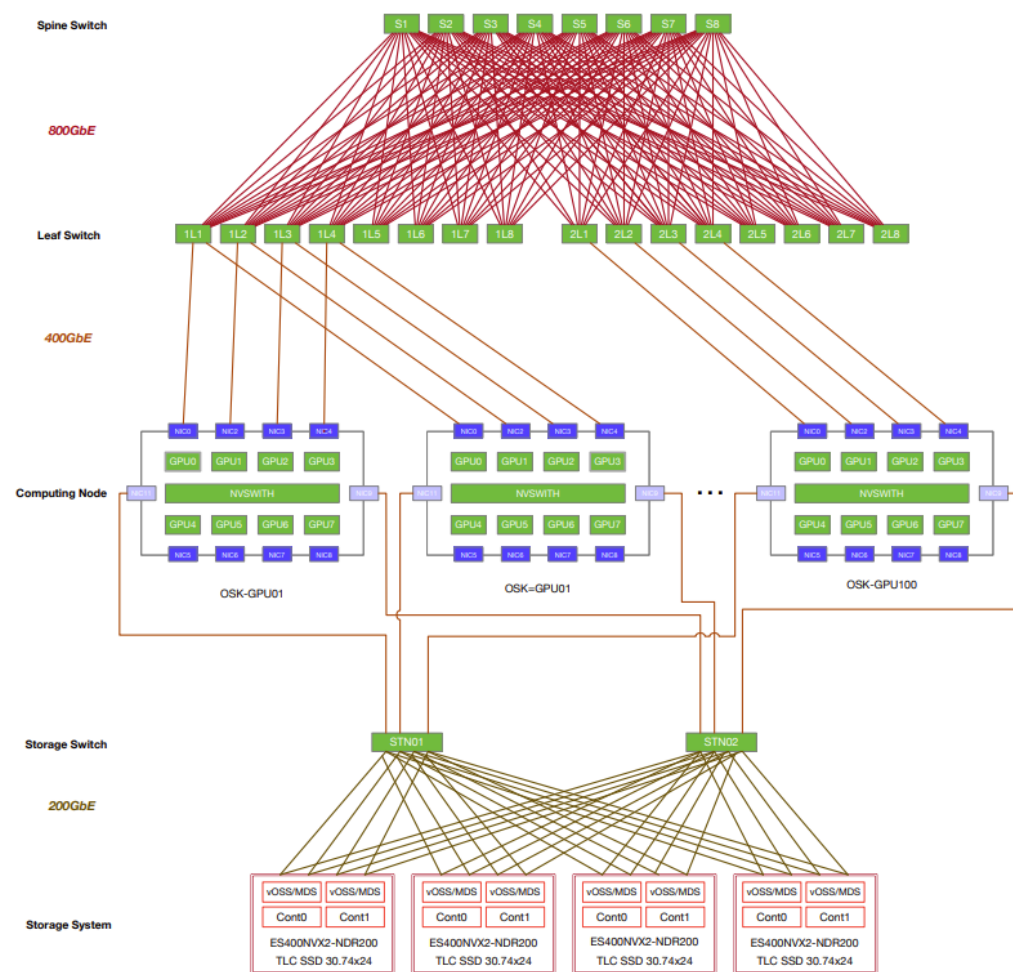


Figure 2: SAKURAONE Network Overview



USER STORY

### Open Networking at Scale: How SAKURA internet Deployed a TOP500 GPU Supercomputer with SONiC

#### Organization

SAKURA internet Inc. is an internet company founded in 1996. Under the corporate philosophy of "Turning 'what you want to do' into 'what you can do,'" we develop a variety of services to meet customer needs and propose DX solutions that cater to various industries.

Since our founding, which began with the provision of shared server services, we have expanded to offer services such as "Koukaryoku" to support generative AI, and "SAKURA Cloud," which has been conditionally certified for use in government cloud systems. A key feature of our company is that we handle everything from development to operations in-house.

#### Overview

SAKURA internet is responding to the growing demand for computational infrastructure driven by the rapid adoption of generative AI by continuously procuring next-generation GPUs and strengthening reliable operational systems in our own data centers. As a digital infrastructure company contributing to the sustainable development of the digital society, our mission is to provide cloud services for generative AI.

To continuously meet the increasing demand, we recognized the necessity of resolving the following challenges:

- Ensuring vendor-neutral supply to mitigate risks
- Adopting technologies with high neutrality
- Accelerating delivery speed

#### Why SONiC?

We selected SONiC for our new 800-GPU cluster because it directly addressed these needs. SONiC provided:

- High transparency, as it is implemented on a Debian/Linux platform
- Active development of new features supported by the global community
- The ability to streamline operations by leveraging the same technologies used for Linux servers

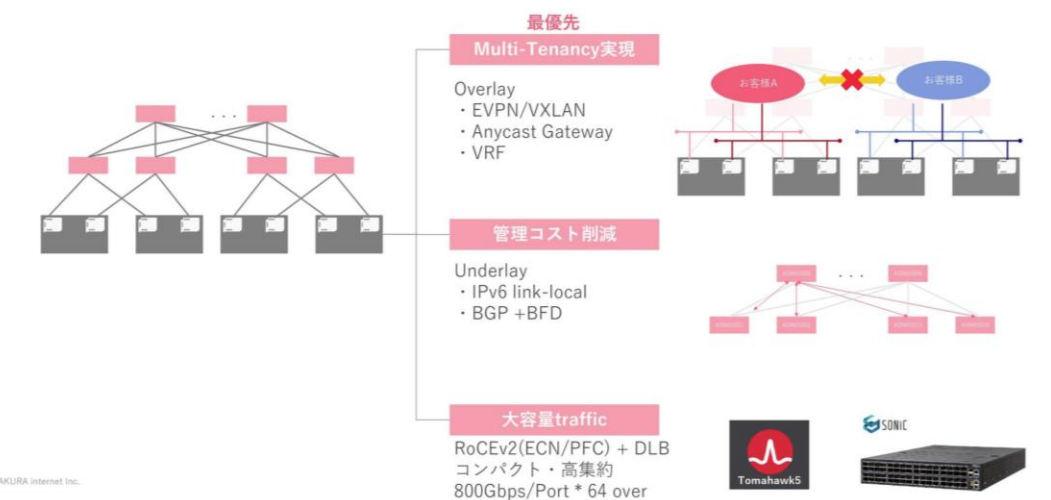
In addition, SAKURA internet has a corporate culture of leveraging OSS and bottom-up initiatives. This culture supported our adoption of SONiC and enabled us to launch a GPU cloud infrastructure in a very short period of time.

### SONiCで構築・運用する生成AI向けパブリッククラウドネットワーク

さくらインターネット株式会社  
黒澤 潔裕



構成要素の検討



運用コマンドの拡充

#### 未成熟機能はLinux/Pythonを用いて自社スクリプト開発

隣接機器の確認

```
admin@gpurne-008:~$ show ip neighbor
Capability codes: R Router, B Bridge, O Other
LocalPort RemoteDevice RemotePort Capability RemotePortDesc
Ethernet100 10-100-3-gleaf-201 Eth24(Port24) BR To 10-100-3-gpurne-008
Ethernet101 10-100-3-gleaf-202 Eth24(Port24) BR To 10-100-3-gpurne-008
Ethernet102 10-100-3-gleaf-203 Eth24(Port24) BR To 10-100-3-gpurne-008
Ethernet103 10-100-3-gleaf-204 Eth24(Port24) BR To 10-100-3-gpurne-008
```

Port? Interface? = 混乱  
Descriptionは要らない

configの差分確認  
admin@gpurne-001:~\$ sudo diff startup-config.log running-config.log  
(Sort違いにより大量の差分)

本当の差分はどこにある?

LLDP/BGPの情報を整理

```
admin@gpurne-008:~$ show lldp neighbor
LocalPort Port RemotePort RemotePortDesc BGP state BFD state
Ethernet100 Port1 10-100-3-gleaf-201 Ethernet104 Port24 Established Up
Ethernet101 Port2 10-100-3-gleaf-202 Ethernet104 Port24 Established Up
Ethernet102 Port3 10-100-3-gleaf-203 Ethernet104 Port24 Established Up
Ethernet103 Port4 10-100-3-gleaf-204 Ethernet104 Port24 Established Up
```

SONiCの設定を横断で差分比較

```
admin@gpurne-008:~$ show config diff
SONiC Running Configurationの取得が完了しました。
SONiC Startup Configurationの読み込みが完了しました。
ALERT: SONiC Configは差分が検出されました。
-- SONiC Config Running Config
+++ SONiC Config Startup Config
diff: 30852, 2, 69965, 8, 888
"TELEX_COUNTER_STATUS": "enable"
"WRRED_CON_QUEUE":
"TELEX_COUNTER_DELAY_STATUS": "false",
"TELEX_COUNTER_STATUS": "enable"
}
SONiC2205: FRB Configに差分はありません。
```