レポートの背景情報

このセクションでは、詳細な結果を記載し、テスト対象のソリューションとテスト方法について説明します。

実地テストは 2024 年 4 月 9 日に終了しました。テスト中に、適切なハードウェアとソフトウェアの構成を決定し、利用可能になった更新を適用しました。このレポートの結果には、2024 年 3 月 11 日以前に確定された構成が反映されています。やむを得ず、これらの構成が、このレポートの公開時点で使用可能な最新のバージョンではない可能性があります。

チャート

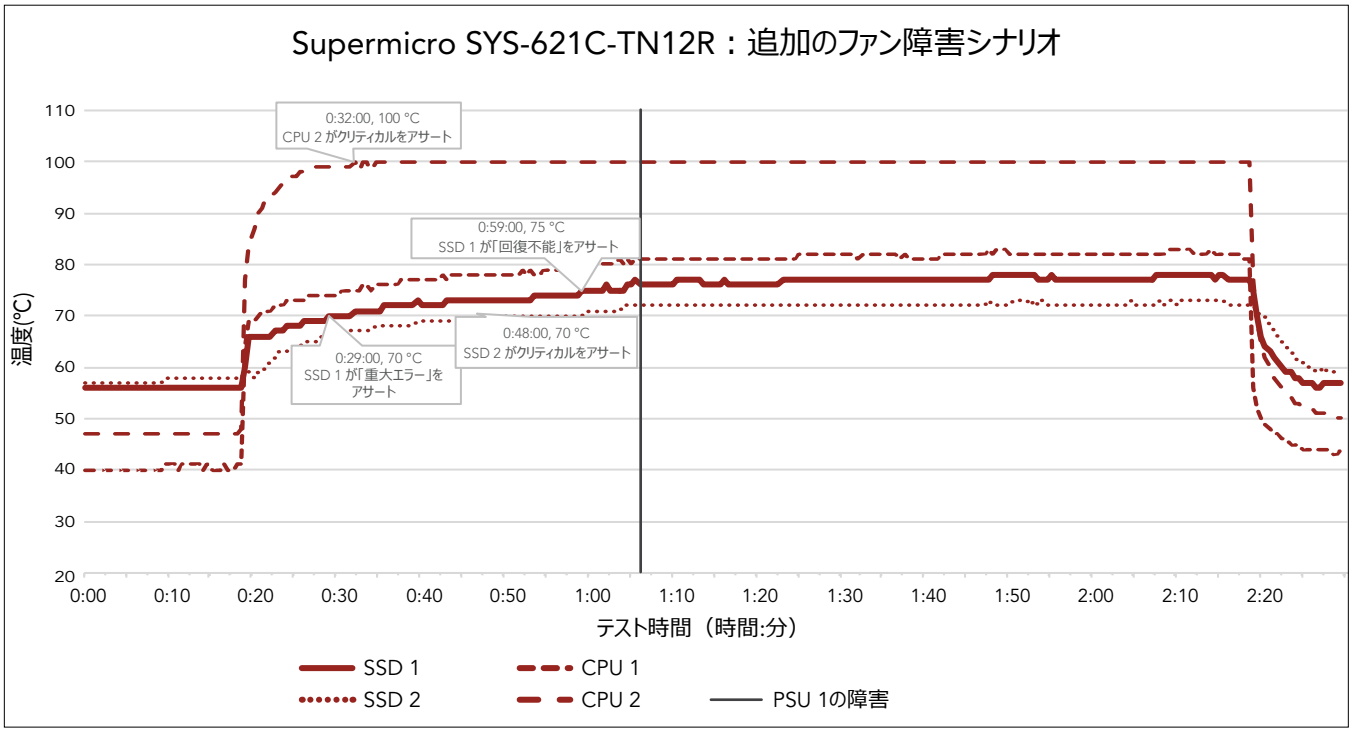

図 1：ファン 3 を無効にした状態でサーバーが浮動小数点ワークロードを実行した追加のファン障害シナリオにおける、Supermicro® SYS-621C-TN12 の SSD とプロセッサーの温度。ワークロードは 0:15 に開始し、2:15 に終了。SSD 1 で OS を実行し、SSD 2 はアイドル状態。出典：Principled Technologies。

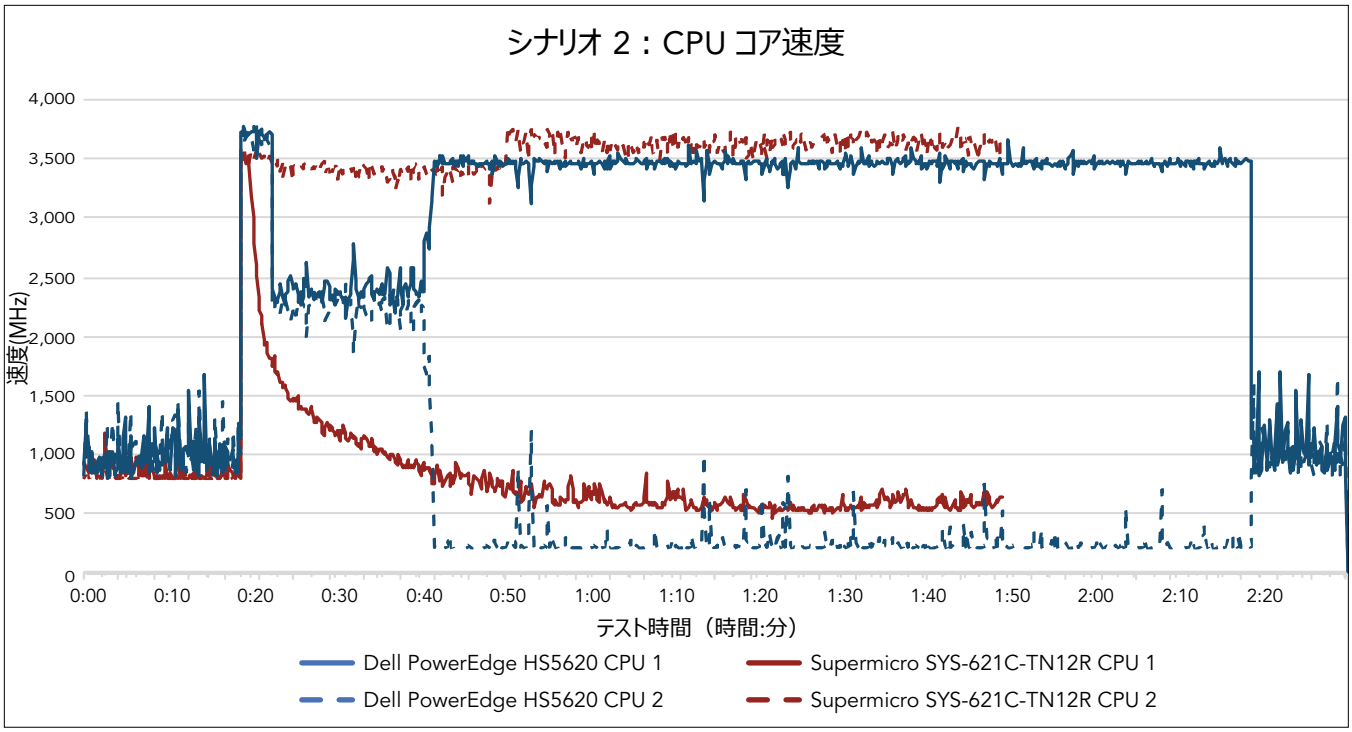

図 2：1 つ目のファン障害シナリオにおける Dell™ PowerEdge™ HS5620 と Supermicro SYS-621C-TN12 のプロセッサー コア速度。浮動小数点ワークロードは 0:15 に開始し、2:15 に終了。出典：Principled Technologies。

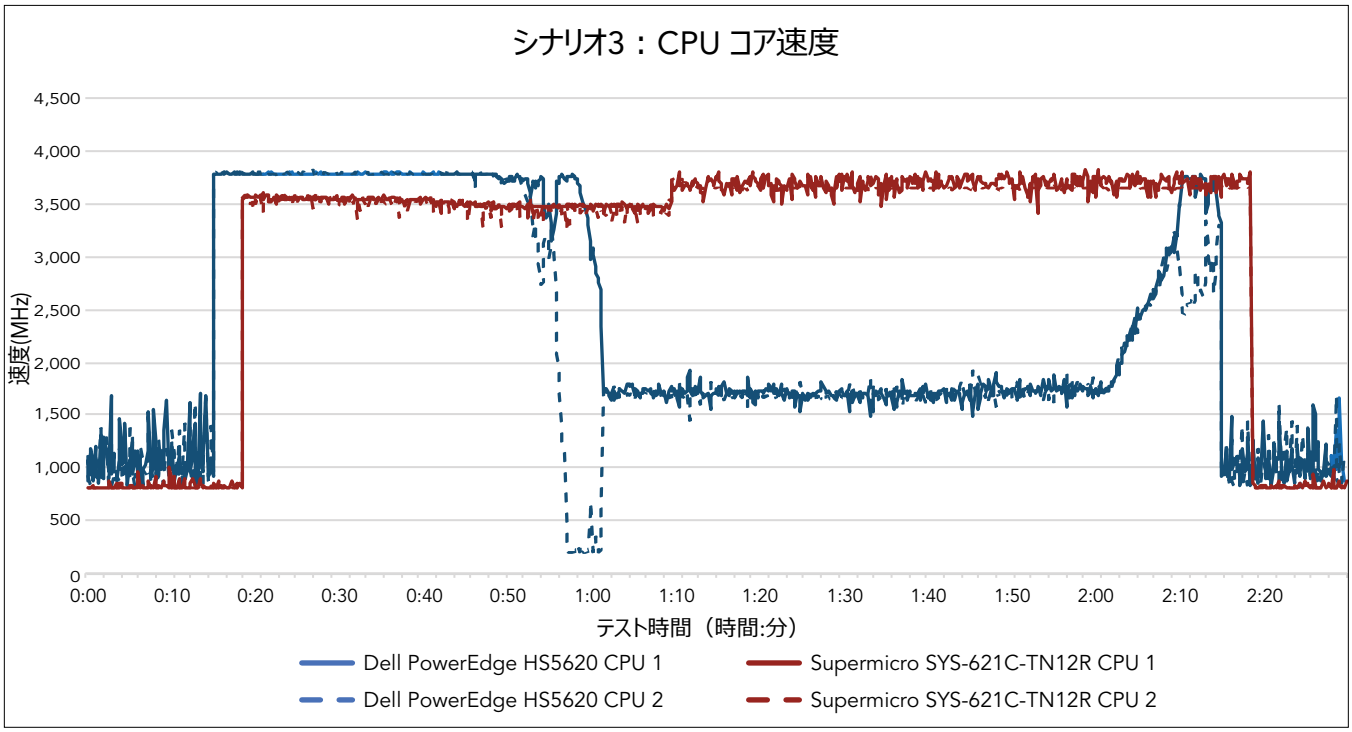

図 3：HVAC 誤動作シナリオにおける Dell PowerEdge HS5620 と Supermicro SYS-621C-TN12 のプロセッサー コア速度。浮動小数点ワークロードは 0:15 に開始し、2:15 に終了。出典：Principled Technologies。

システム構成情報

表 1：今回のテストで使用したシステムの詳細情報。

システム構成情報	Dell PowerEdge HS5620	Supermicro SYS-621C-TN12R
BIOS 名とバージョン	Dell 2.1.3	Supermicro 2.1
デフォルト以外の BIOS 設定	ワットあたりのパフォーマンス (OS)	該当なし
オペレーティング システムの名前とバージョン / ビルド番号	Ubuntu 22.04.3	Ubuntu 22.04.3
OS 更新プログラム / パッチの最終適用日	2024 年 3 月 11 日	2024 年 1 月 28 日
電源管理ポリシー	ワットあたりのパフォーマンス (OS)	バランスのとれたパフォーマンス
プロセッサー		
プロセッサーの数	2	2
ベンダーとモデル	インテル ® Xeon® Gold 6444Y	インテル Xeon Gold 6444Y
コア数（プロセッサーあたり）	16	16
コア周波数 (GHz)	3.60（4.0 ターボ）	3.60（4.0 ターボ）
ステッピング	8	8
メモリー モジュール		
システム メモリーの合計 (GB)	1,024	1,024
メモリー モジュールの数	16	16
ベンダーとモデル	Hynix® HMCG94AEBRA109N	SK Hynix HMCG94MEBRA123N
サイズ (GB)	64	64
タイプ	DDR5 DIMM	DDR5
速度 (MHz)	4,800	4,800
サーバーでの実行速度 (MHz)	4,800	4,800
ストレージ コントローラー（前面ストレージ）		
ベンダーとモデル	Dell HBA355i アダプター	Supermicro MegaRAID AOC-S3916L-H16iR-32DD-P
キャッシュ サイズ (GB)	0	8
ファームウェアのバージョン	24.15.14.00	5.240.02-3768
ドライバーのバージョン	該当なし	52.24.0-4766
ストレージ コントローラー (NVMe® M.2)		
ベンダーとモデル	Dell BOSS-N1 モノリス型	該当なし
キャッシュ サイズ (GB)	0	該当なし
ファームウェアのバージョン	2.1.13.2025	該当なし
ローカル ストレージ (OS)		
ドライブ数	2	2
ドライブのベンダーとモデル	Dell NVMe PE8010 RI M.2 960GB	Micron® 7450 MTFDKBA960TFR

システム構成情報	Dell PowerEdge HS5620	Supermicro SYS-621C-TN12R
ドライブ サイズ (GB)	960	960
ドライブ情報（速度、インターフェイス、タイプ）	8GT/s M.2 SSD	PCIe® M.2 NVMe
ローカル ストレージ（データ）		
ドライブ数	12	12
ドライブのベンダーとモデル	HGST HUH721212AL5200	WDC WUH721814ALE6L4
ドライブ サイズ (GB)	120,000	1,400
ドライブ情報（速度、インターフェイス、タイプ）	12 Gbps SAS 3.5 インチ HDD	6Gb SATA 3.5 インチ HDD
ネットワーク アダプター A		
ベンダーとモデル	4x インテル 25G 2P E810-XXV	3x インテル E810-XXVAM2 (AOC-S25GC-i2S)
ポートの数とタイプ	2x 25Gb	2x 25Gb
ドライバーのバージョン	22.5.7	4.20 (0x800177B4)
ネットワーク アダプター B		
ベンダーとモデル	1x Broadcom® NetXtreme ギガビット Ethernet (BCM5720)	1x インテル E810-XXVAM2 (AOC-A25G-i2SM)
ポートの数とタイプ	2x 1Gb	2x 25Gb
ドライバーのバージョン	22.71.3	4.30 (0x800177B4)
冷却ファン		
番号、ベンダー、モデル	1x Dell HPR ゴールド 5x Dell HPR シルバー	3x Supermicro 製ミドル ファン FAN-0206L4
電源装置		
ベンダーとモデル	Dell 05222NA00	Supermicro PWS-1K23A-1R
電源装置の数	2	2
個々のワット数 (W)	1,800	1,200

テスト方法

温度の制御と測定を行える環境を作るために、フル装備の 42U サーバー ラックの周囲にはカスタム エンクロージャを構築しました。Dell システムと Supermicro システムはどちらもラック内の同じ位置でテストし、3 つのシナリオでサーバーの内部温度を収集しました。ラック内のサーバーに対して 4 つのウェーブで、それぞれ 1 分 10 秒の間隔をあけて stress-ng ベンチマークを実行しました。今回テストした Dell と Supermicro のシステムでは、最初のサーバーがワークロードを開始してから 3 分 30 秒後、4 つ目のウェーブでワークロードを起動しました。テストのセットアップと実行のために取った手順を以下にまとめます。

Ubuntu 22.04.3 のインストールと設定

1. Ubuntu 22.04.3 メディアから起動する。
2. [Try or Install Ubuntu Server] を選択する。
3. 言語メニューでは、デフォルトのまま [Done] を選択する。
4. [Update to the new installer] を選択する。
5. [Keyboard configuration] では、デフォルトのまま [Done] を選択する。
6. [Installation type] では、デフォルトのまま [Done] をクリックする。
7. [Network connections] メニューでは、デフォルトのまま [Done] を選択する。
8. [Configure proxy] 画面では、デフォルトのまま [Done] を選択する。
9. [Configure Ubuntu archive mirror] 画面では、テストに合格するのを待ってから、[Done] を選択する。
10. [Guided storage configuration] 画面では、デフォルトのまま [Done] を選択する。
11. [Storage configuration] 画面のサマリーでは、デフォルトのまま [Done] を選択する。
12. 破壊アクションを確認するために、[Continue] を選択する。
13. [Profile setup] 画面の [Your name] と [Username] に「ptuser」と入力する。[Your servers name] に名前を入力し、パスワードを確認する。
14. [Done] を選択する。
15. Ubuntu Pro へのアップグレード画面では、デフォルトのまま [Continue] を選択する。
16. [SSH Setup] 画面で [Install OpenSSH server] を選択し、[Done] を選択する。
17. [Featured Server Snaps] 画面では、デフォルトのまま [Done] を選択する。
18. インストールが完了したら、[Reboot now] を選択する。
19. 上記で作成した認証情報を使用して Ubuntu にログインする。
20. アップデートを処理する。

```
sudo apt update
sudo apt upgrade
```

21. CIFS ユーティリティーをインストールし、PT 共有をマッピングする。

```
sudo apt install cifs-utils
sudo mkdir /mnt/pt-data01
sudo mount -t cifs //10.41.1.21/pt /mnt/pt-data01/ -o “rw,user=<useraccount>,pass=<password>”
```

22. ネットワーキングを構成する。

```
sudo cp /etc/netplan/*.yaml /etc/netplan/00-installer-config.yaml.bak
sudo nano /etc/netplan/*.yaml
```

23. 目的のネットワーク アダプターを特定し、次の調整を行う。

```
addresses:
  - <IP_Address>/<CIDR>
routes:
  - to: default
    via: <Default_Gateway>
nameservers:
    search: [<NameServer1>, <NameServer2>]
    addresses: [<DNS_IP1>, <DNS_IP2>, <DNS_IP3>]
```

24. 変更されたファイルをテストして適用する。

```
sudo netplan try
sudo netplan apply
```

25. ホスト名を設定する。

```
sudo hostnamectl set-hostname <NewHostname>
```

26. ホストを再起動する。

```
sudo shutdown -r now
```

パスワード不要 sudo のデプロイ

クライアント側でのデプロイ

1. sudoers ファイルを編集する。

```
sudo visudo /etc/sudoers
```

2. ファイルの末尾に次の行を追加する。

```
ptuser ALL=(ALL:ALL) NOPASSWD: ALL
```

コントローラー側でのデプロイ

1. SSH キーペアを生成する。

```
ssh-keygen -t rsa -b 4096 -N "" -f "$HOME/.ssh/id_rsa.pub"
```

2. SSH 公開キーを各リモート サーバーにコピーする。

```
ssh-copy-id ptuser@<remote_server_IP>
```

データ コレクションのための TIG-P スタックの実装

Docker の構成

1. ptuser としてロギング マシンにログインする。
2. Docker のインストールの準備をする。

```
sudo apt update
sudo apt install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

3. Apt ソースにリポジトリーを追加し、インストールする。

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.
```

```
docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Huntabyte TIG スタックの構成

1. ロギング マシンで tig-stack リポジトリーのクローンを作成する。

```
git clone https://github.com/huntabyte/tig-stack.git
```

2. デプロイ用に .env ファイルを編集する。

```
sudo nano tig-stack/.env
```

3. FluxDB について、次のようにユーザー名、パスワード、組織、バケット、保持期間を入力する。

```
DOCKER_INFLUXDB_INIT_USERNAME: admin
DOCKER_INFLUXDB_INIT_Password: <PasswordHere>
DOCKER_INFLUXDB_INIT_ORG: PT
DOCKER_INFLUXDB_INIT_BUCKET: <BucketName>
DOCKER_INFLUXDB_INIT_RETENTION: 52w
```

4. 次のコマンドを使用してランダムな 32 文字の 16 進文字列を生成し、管理者トークンの結果を .env ファイルに入力する。

```
openssl rand -hex 32
```

5. 保存して終了する。
6. telegraf.conf を編集する。

```
sudo nano tig-stack/telegraf/telegraf.conf
```

7. 次の値を設定する。

```
services:
  influxdb:
    image: influxdb
  telegraf:
    image: gibletron/telegraf-ipmitool
  grafana:
    image: grafana/grafana-oss
    links:
      - prometheus
```

8. 保存して終了する。
9. Docker Compose を起動する（ヘッドレス、デタッチ モード）。

```
sudo docker-compose up -d
```

10. 監視する各サーバーで、次のコマンドを実行する。

```
sudo apt install telegraf
```

11. InfluxDB 管理インターフェイスを開くために、ポート 8086 で InfluxDB ホストの IP アドレスをブラウズする。

12. 必要に応じて API トークンを作成し、ウィンドウを閉じる前に必ず記録する。
13. ［Load Data］で［API Tokens］をクリックし、［Generate API Token］をクリックする。
14. 監視する各サーバーで、/etc/telegraf/telegraf.conf を次のように編集する。

```
[[outputs.influxdb_v2]]
  urls = ["<influxDB_IP>:8086"]
  token = "<API_token>"
  organization = "PT"
  bucket = "<bucket_name>"
```

15. テスト対象の各システムで次の行を追加する。

```
[[inputs.intel_powerstat]]
  cpu_metrics = ["cpu_frequency"]
```

16. 保存して終了する。
17. Telegraf を再起動する。

```
sudo systemctl restart telegraf
```

Prometheus の構成

1. 次の行を /home/ptuser/tig-stack/docker-compose.yml に追加する。

```
prometheus:
  image: prom/prometheus:latest
  volumes:
    - ${PROM_CFG_PATH}:/etc/prometheus/prometheus.yml
    - prom-storage:/prometheus
  ports:
    - 9090:9090
volumes:
  prom-storage:
```

2. 保存して終了する。
3. .env ファイルを編集し、次の行を追加する。

```
PROM_CFG_PATH=./prometheus/prometheus.yml
```

4. 保存して終了する。
5. 監視する各サーバーで、次のコマンドを実行する。

```
sudo apt install dbus prometheus-node-exporter prometheus-node-exporter-collectors -y
```

6. テスト対象の各システムで次のコマンドを実行する。

```
sudo apt install prometheus -y
```

7. Prometheus で監視ジョブを作成するために、次の行を /home/ptuser/tig-stack/prometheus/prometheus.yml に追加する。

```
- job_name: "<custom_name>"
  static_configs:
    - targets: ["<target_IP:9090>"]
```

8. ステップ 7 と同様に、追加のエントリーを作成して、ジョブやターゲットを追加する。その他のターゲットも、同じジョブに対して別のターゲット行として追加できる。
9. 保存して終了する。

stress-ng によるテスト

各テスト シナリオでは、次の手順に従って stress-ng 浮動小数点ワークロードを実行しました。

1. 各サーバーで次のコマンドを実行する。

```
sudo apt install stress-ng linux-tools-generic -y
```

2. テスト対象の各サーバーで次のコマンドを実行する。

```
sudo modprobe rapl
sudo modprobe intel_rapl_common
sudo modprobe intel_rapl_msr
sudo modprobe msr
sudo modprobe intel-uncore-frequency
sudo setcap cap_sys_rawio,cap_dac_read_search,cap_sys_admin+ep /usr/bin/telegraf
sudo chmod -R a+rx /sys/devices/virtual/powercap/intel-rapl/
```

3. テスト対象の各サーバーで *https://github.com/andikleen/pmu-tools/blob/master/event_download.py* に移動し、未加工ファイルをダウンロードして実行する。

```
sudo chmod +x event_download.py
./event_download.py
```

4. コントローラーで PSSH をインストールする。

```
sudo apt install pssh -y
```

5. コントローラーで、stress-ng の実行中に使用するファイルを作成する。

```
sudo touch ~/.pssh_hosts_file
sudo touch ~/.pssh_hosts_file_wave1
sudo touch ~/.pssh_hosts_file_wave2
sudo touch ~/.pssh_hosts_file_wave3
sudo touch ~/.pssh_hosts_file_wave4
```

6. ~/.pssh_hosts_file ファイルを編集し、すべてのサーバー IP アドレスを各行に 1 つずつ入力する。
7. ~/.pssh_hosts_file_wave1 から ~/.pssh_hosts_file_wave4 までのファイルを編集し、各ファイルの IP アドレスの 4 分の 1 を適切に入力する。
8. すべてのサーバーがネットワークに接続されており、リモート コマンドに応答していることをテストする。

```
sudo pssh -i -h ~/.pssh_hosts_file uptime
```

9. コントローラーで stress-ng テスト用のログ フォルダーを作成する。

```
sudo mkdir /var/log/stress-ng
sudo chmod 777 /var/log/stress-ng
```

10. 「wave1」を適切なウェーブ番号に編集して、次のコマンドでテストを実行する。

```
pssh -i -h ~/.pssh_hosts_file_wave1 sudo stress-ng --cpu 4 --matrix 0 --cpu-method matrixprod --mq
4 --hdd 6 --tz --metrics --perf --times --aggressive -t 2h --log-file /var/log/stress-ng/$(date
+'%Y%m%d_%H%M%S').log
```

▶ オリジナルの英語版のレポートは
https://facts.pt/gPS09my からご覧いただけます

このプロジェクトは、デル・テクノロジーズの委託を受けて作成されています。

Principled Technologies®

Facts matter.®