



採用**Run:AI**的**NetApp**協調解決方案 NetApp Solutions

NetApp
April 12, 2024

This PDF was generated from https://docs.netapp.com/zh-tw/netapp-solutions/ai/osrunai_solution_overview.html on April 12, 2024. Always check docs.netapp.com for the latest.

目錄

TR-4858：NetApp協調解決方案搭配Run:AI	1
解決方案總覽	1
解決方案技術	2
最佳叢集與GPU使用率搭配Run:AI	3
結論	15
第4.8節的測試詳細資料	15
第4.9節的測試詳細資料	16
第4.10節的測試詳細資料	17
何處可找到其他資訊	18

TR-4858：NetApp協調解決方案搭配Run:AI

Rick Huang、David Arignette、Sung-HAN LIN、NetApp Yaron Goldberg、Run：AI

NetApp AFF 供應極致效能與領先業界的混合雲資料管理功能。NetApp與Run：AI合作展示NetApp ONTAP SAI解決方案的獨特功能、適用於人工智慧（AI）與機器學習（ML）工作負載、提供企業級效能、可靠性與支援。執行：AI工作負載的AI協調功能可新增Kubernetes型排程與資源使用平台、協助研究人員管理及最佳化GPU使用率。NetApp、NVIDIA和Run:AI的結合解決方案結合NVIDIA DGX系統、提供專為企業AI工作負載所打造的基礎架構堆疊。本技術報告提供指引、協助客戶建立對話式AI系統、以支援各種使用案例和產業垂直市場。其中包含有關Run：AI和NetApp AFF Sof A800儲存系統部署的資訊、並可做為參考架構、以提供最簡單的方法、協助您快速成功地部署AI方案。

解決方案的目標對象包括下列群組：

- 企業架構設計師設計解決方案、以開發適用於Kubernetes型使用案例的AI模型和軟體、例如容器化微服務
- 資料科學家正在尋求高效率的方法、以便在具有多個團隊和專案的叢集環境中達成有效的模型開發目標
- 負責維護和執行正式作業模式的資料工程師
- 想要創造最佳Kubernetes叢集資源使用體驗、並從AI計畫中獲得最快上市時間的高階主管與IT決策者及企業領導者

解決方案總覽

NetApp ONTAP 的AI與AI控制平面

NetApp ONTAP 與NVIDIA開發並驗證的NetApp AI架構、採用NVIDIA DGX系統與NetApp雲端連線儲存系統。此參考架構可為IT組織提供下列優勢：

- 消除設計複雜性
- 可獨立擴充運算與儲存設備
- 讓客戶從小規模開始、並無縫擴充
- 提供多種儲存選項、可滿足各種效能與成本效益

NetApp ONTAP W人工 智慧將DGX系統與NetApp AFF 的Arde-A800儲存系統與先進的網路技術緊密整合。NetApp ONTAP 採用AI和DGX系統、可免除設計複雜度和猜測、簡化AI部署。客戶可以從小規模開始、以不中斷的方式擴充系統、同時以智慧方式管理從邊緣到核心到雲端及後端的資料。

NetApp AI Control Plane是適用於資料科學家和資料工程師的完整堆疊AI、ML和深度學習（DL）資料與實驗管理解決方案。隨著企業組織增加AI使用率、他們面臨許多挑戰、包括工作負載擴充性和資料可用度。NetApp AI Control Plane透過各種功能來解決這些挑戰、例如像Git repo一樣快速複製資料命名空間、以及定義及實作AI訓練工作流程、將近乎即時的資料建立與模型基準、以利追蹤及版本管理。有了NetApp AI Control Plane、您就能在不同站台和區域之間順暢複寫資料、並迅速配置Jupyter筆記型電腦工作區、存取大量資料集。

Run：AI工作負載協調的AI平台

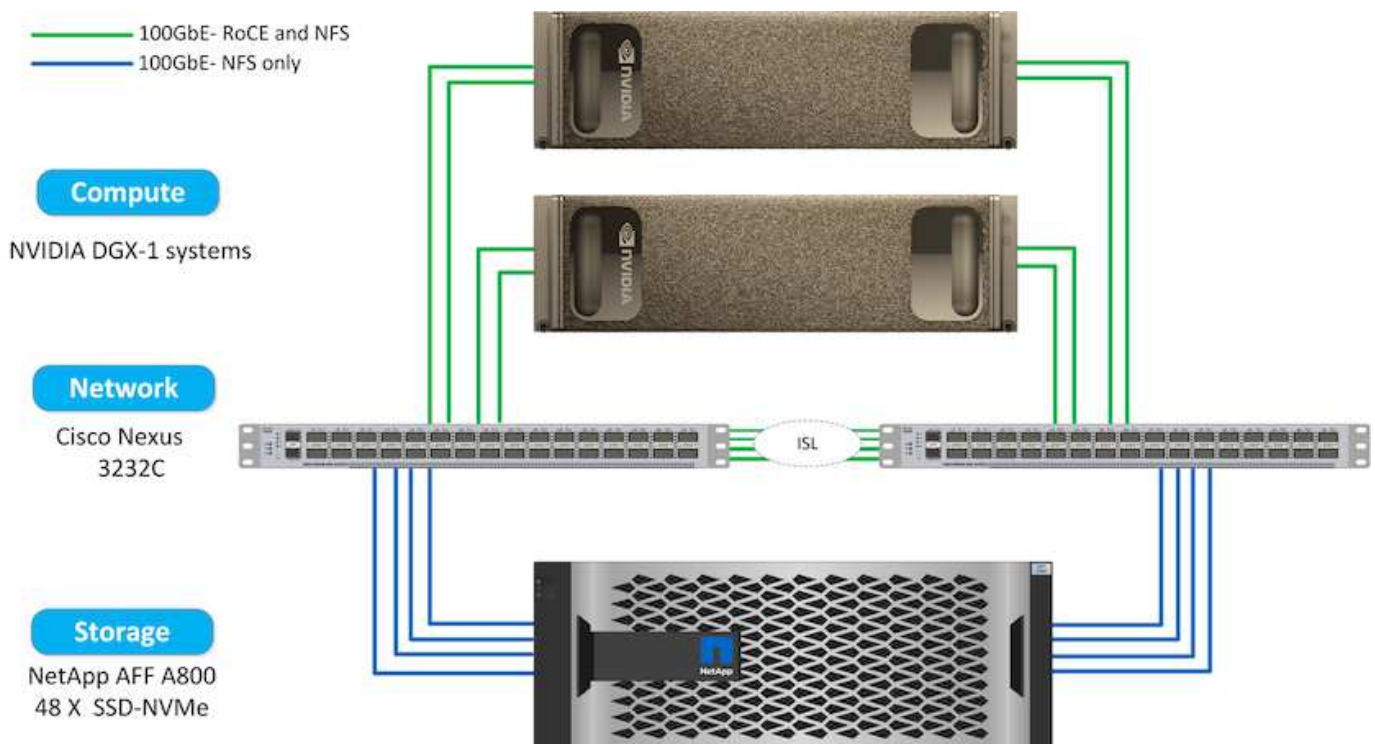
RUN：AI為AI基礎架構建置了全球第一個協調與虛擬化平台。執行：AI將工作負載從基礎硬體中抽離、建立可動態配置的GPU資源共享資源池、有效協調AI工作負載、並最佳化GPU的使用。資料科學家可以無縫地消耗大量GPU電力來改善及加速研究、同時IT團隊也能在資源配置、佇列及使用率方面、維持集中化的跨站台控制及即時可見度。Run：AI平台建置於Kubernetes之上、可與現有的IT和資料科學工作流程輕鬆整合。

Run：AI平台提供下列優點：

- *縮短創新時間。*藉由搭配使用Run:AI資源池、佇列及優先順序機制、再搭配NetApp儲存系統、研究人員將不再受基礎架構管理的困擾、而且只能專注於資料科學。RUN：AI與NetApp客戶可在不產生運算或資料傳輸途徑瓶頸的情況下、視需要執行任意數量的工作負載、進而提高生產力。
- *提高團隊生產力。*執行：AI公平演算法可確保所有使用者和團隊都能獲得公平的資源共享。您可以預設優先專案的相關原則、而平台可將資源從一個使用者或團隊動態分配給另一個使用者或團隊、協助使用者及時存取令人夢寐以求的GPU資源。
- 提升**GPU**使用率。Run:AI排程器可讓使用者輕鬆使用部分GPU、整型GPU及GPU的多個節點、進行Kubernetes的分散式訓練。如此一來、AI工作負載就會根據您的需求而非容量來執行。資料科學團隊能夠在相同的基礎架構上執行更多AI實驗。

解決方案技術

此解決方案是以一AFF 部NetApp Efora800系統、兩部DGX-1伺服器、以及兩部Cisco Nexus 3232C 100Gbe-switches來實作。每部DGX-1伺服器均連接至Nexus交換器、並具有四個100GbE連線、可透過融合式乙太網路（RoCE）使用遠端直接記憶體存取（RDMA）進行GPU間通訊。NFS儲存存取的傳統IP通訊也會發生在這些連結上。每個儲存控制器都使用四個100GbE連結連接至網路交換器。下圖顯示ONTAP 本技術報告中針對所有測試案例所使用的AI解決方案架構。



本解決方案所使用的硬體

此解決方案已通過ONTAP 驗證、採用了一套支援整個AI參考架構、其中兩個DGX-1節點和一個AFF 支援完善的資料中心A800儲存系統。請參閱 ["NVA-1121."](#) 如需此驗證所使用基礎架構的詳細資訊、請參閱。

下表列出實作已測試解決方案所需的硬體元件。

硬體	數量
DGX-1系統	2.
解答800 AFF	1.
Nexus 3232C交換器	2.

軟體需求

此解決方案已通過基本Kubernetes部署驗證、並已安裝Run:AI操作員。Kubernetes是使用部署的 ["NVIDIA DeepOps"](#) 部署引擎、可部署正式作業就緒環境所需的所有元件。DeepOps會自動部署 ["NetApp Trident"](#) 為了持續與k8s環境整合儲存設備、我們建立了預設的儲存類別、讓容器能夠利用AFF 來自於該系統的儲存設備。如需ONTAP 有關Trident with Kubernetes on AI的詳細資訊、請參閱 ["TR-4798"](#)。

下表列出實作已測試解決方案所需的軟體元件。

軟體	版本或其他資訊
NetApp ONTAP 資料管理軟體	9.6p4.
Cisco NX-OS交換器韌體	7.0 (3) I6 (1)
NVIDIA DGX OS	4.0.4 - Ubuntu 18.04 LTS
Kubernetes版本	1.17
Trident版本	20.04.0
執行：AI CLI	v2.1.13
RUN：AI Orchestration Kubernetes運算子版本	1.0.39
Docker Container平台	18.06.1-CE [e68fc7a]

有關Run:AI的其他軟體需求、請參閱 ["執行：AI GPU叢集先決條件"](#)。

最佳叢集與GPU使用率搭配Run:AI

下列各節提供執行：AI安裝、測試案例及此驗證所執行結果的詳細資料。

我們使用業界標準基準測試工具（包括TensorFlow基準測試）來驗證此系統的運作和效能。ImageNet資料集用於訓練ResNet-50、這是著名的Convolutional Neuriency Network（CNN/有線新聞網路）DL影像分類模式。ResNet-50提供準確的訓練結果、並加快處理時間、讓我們能夠為儲存設備帶來足夠的需求。

執行：AI安裝

若要安裝Run：AI、請完成下列步驟：

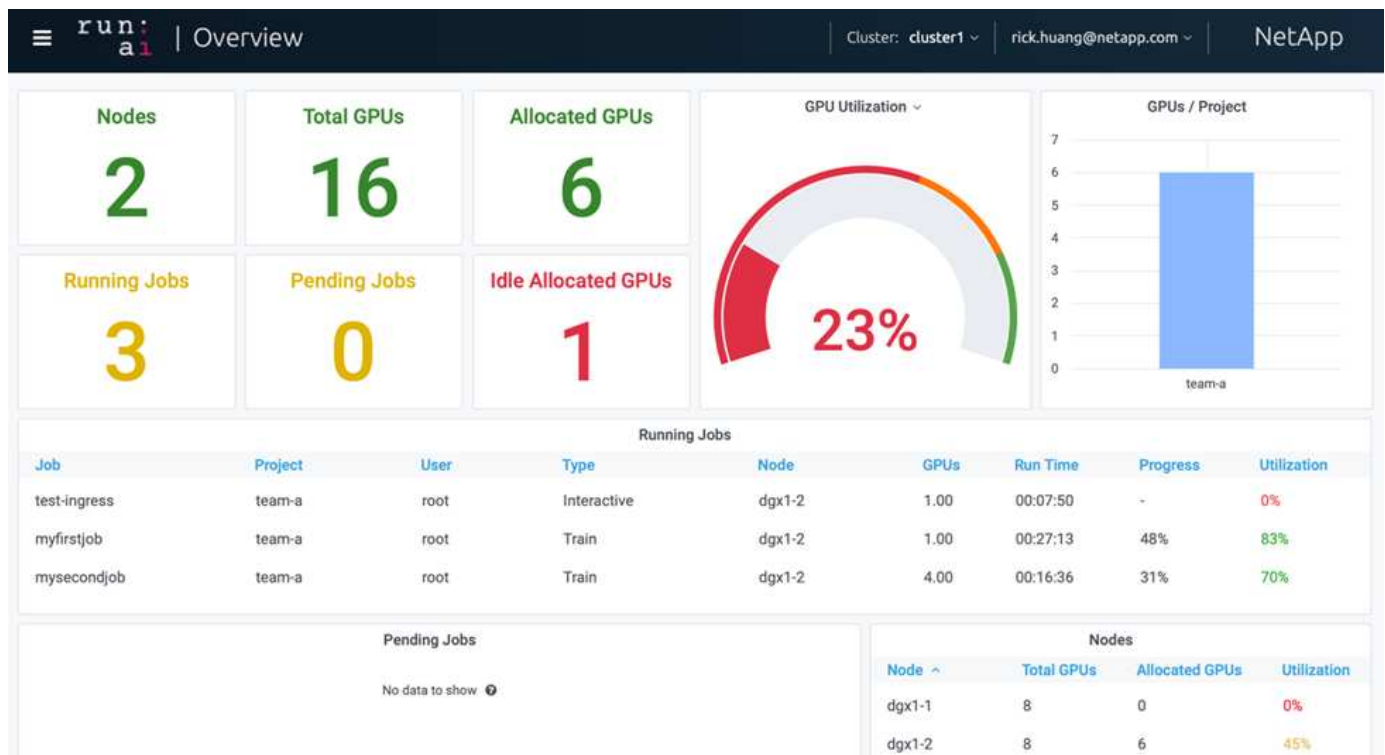
1. 使用DeepOps安裝Kubernetes叢集、並設定NetApp預設儲存類別。
2. 準備GPU節點：
 - a. 確認GPU節點上已安裝NVIDIA驅動程式。
 - b. 確認「nvidia-Docker」已安裝並設定為預設的泊塢視窗執行時間。
3. 安裝執行：AI：
 - a. 登入 "執行：AI管理UI" 以建立叢集。
 - b. 下載建立的「runai-oper-<clustername>.yaml」檔案。
 - c. 將操作員組態套用至Kubernetes叢集。

```
kubectl apply -f runai-operator-<clustername>.yaml
```

4. 驗證安裝：
 - a. 前往 "<https://app.run.ai/>"。
 - b. 前往「總覽」儀表板。
 - c. 確認右上角的GPU數量反映出GPU的預期數量、而GPU節點全都列在伺服器清單中。如需執行：AI部署的詳細資訊、請參閱 "[在內部部署Kubernetes叢集上安裝Run:AI](#)" 和 "[安裝Run:AI CLI](#)"。

執行：AI儀表板和檢視

在Kubernetes叢集上安裝Run:AI並正確設定容器之後、您會在上看到下列儀表板和檢視 "<https://app.run.ai/>" 下圖所示。



叢集中共有16個GPU、由兩個DGX-1節點提供。您可以查看節點數量、可用的GPU總數、指派給工作負載的已分配GPU、執行中工作的總數、擱置中工作、以及閒置配置的GPU。右側的長條圖顯示每個專案的GPU、其中摘要說明不同的團隊如何使用叢集資源。中間是目前執行中工作的清單、其中包含工作詳細資料、包括工作名稱、專案、使用者、工作類型、每個工作所在的節點、為該工作分配的GPU數量、工作目前的執行時間、工作進度百分比、以及該工作的GPU使用率。請注意、叢集使用率偏低（GPU使用率為23%）、因為單一團隊（「team A」）只提交三個執行中工作。

在下一節中、我們將示範如何在「專案」索引標籤中建立多個團隊、並為每個團隊分配GPU、以便在每個叢集有許多使用者時、將叢集使用率最大化並管理資源。測試案例模擬企業環境、在這些環境中、訓練、推斷及互動式工作負載之間共享記憶體與GPU資源。

為Data科學團隊建立專案並分配GPU

研究人員可以透過Run:AI CLI、Kubeflow或類似程序來提交工作負載。為了簡化資源配置並建立優先順序、Run:AI引進專案概念。專案是配額實體、可將專案名稱與GPU配置和偏好設定建立關聯。這是管理多個資料科學團隊的簡易方法。

提交工作負載的研究人員必須將專案與工作負載要求建立關聯。Run:AI排程器會將要求與目前的配置和專案進行比較、並決定是否可以分配資源給工作負載、或是否應保持擱置狀態。

身為系統管理員、您可以在「執行：AI專案」索引標籤中設定下列參數：

- ***模型專案。***設定每個使用者的專案、設定每個使用者小組的專案、以及針對真正的組織專案設定專案。
- ***專案配額。***每個專案都會與GPU配額相關聯、以便同時分配給此專案。這是一個保證配額、因為無論叢集的狀態為何、使用此專案的研究人員都能獲得此數量的GPU。一般而言、專案配置的總和應等於叢集中的GPU數量。除此之外、此專案的使用者可能會收到過度配額。只要未使用GPU、使用此專案的研究人員就能取得更多GPU。我們在中示範過配額測試案例和公平考量 ["透過過度配額GPU配置、實現高叢集使用率"](#)、["基本資源配置公平性"](#)和 ["過度配額的公平性"](#)。
- 建立新專案、更新現有專案、以及刪除現有專案。
- 限制在特定節點群組上執行工作。您可以指派特定專案、僅在特定節點上執行。當專案團隊需要專用的硬體（例如有足夠的記憶體）時、此功能非常實用。或者、專案團隊可能是以專業預算購買的特定硬體擁有者、或是當您需要將建置或互動式工作負載導向較弱的硬體、並將較長的訓練或無人管理的工作負載導向較快的節點時。如需群組節點及設定特定專案關聯性的命令、請參閱 ["執行：AI文件"](#)。
- 限制互動工作的持續時間。研究人員經常忘了關閉互動工作。這可能會導致資源浪費。有些組織偏好限制互動工作的持續時間、並自動關閉。

下圖顯示建立了四個團隊的專案檢視。每個團隊都會指派不同數量的GPU來處理不同的工作負載、GPU總數等於由兩個DGX-1組成的叢集中可用GPU總數。

run:ai Projects		Cluster: cluster1		rick.huang@netapp.com		NetApp
Filter and Search						+ Add New project
Project Name ↓	Assigned GPUs	Created	Training Node Affinity	Interactive Node Affinity ⚙		
team-a	2	07/27/20, 9:28AM	none	none		
team-b	4	07/28/20, 7:50AM	none	none		
team-c	2	07/28/20, 7:50AM	none	none		
team-d	8	07/28/20, 7:51AM	none	none		

在Run:AI CLI中提交工作

本節提供基本Run:AI命令的詳細資訊、可用於執行任何Kubernetes工作。它根據工作負載類型分為三個部分。AI / ML / DL工作負載可分為兩種一般類型：

- 無人蔘與的訓練課程。有了這些類型的工作負載、資料科學家就能準備好自行執行工作負載、然後將其傳送執行。執行期間、客戶可以檢查結果。這類工作負載通常用於正式作業、或是在不需要人為介入的階段進行模型開發。
- 互動式建置工作階段。有了這些類型的工作負載、資料科學家就能開啟與Bash、Jupyter Notebook、遠端PyCharm或類似的IDE互動式工作階段、並直接存取GPU資源。我們在第三個案例中、使用連接埠執行互動式工作負載、以向容器使用者展示內部連接埠。

無人管理的訓練工作負載

設定專案並分配GPU之後、您可以在命令列使用下列命令來執行任何Kubernetes工作負載：

```
$ runai project set team-a runai submit hyper1 -i gcr.io/run-ai-demo/quickstart -g 1
```

此命令會為團隊A開始無人值守的訓練工作、並分配單一GPU。工作是以「GCR.IO/RUN -AI DEMO / quickstart」樣本泊塢視窗影像為基礎。我們將工作命名為「hyper1」。然後您可以執行下列命令來監控工作的進度：

```
$ runai list
```

下圖顯示「Runai list」命令的結果。您可能會看到下列一般狀態：

- 「ContainerCreating」。Docker容器正在從雲端儲存庫下載。
- 「待處理」。工作正在等待排程。
- 《執行中》。工作正在執行中。


```

You can run 'runai get hyper1' to check the job status
~> runai list
Showing jobs for project team-a
NAME    STATUS  AGE  NODE                                     IMAGE                                     TYPE    PROJECT  USER  GPUS
hyper1  Running  11s  gke-dev-yaron1-gpu-4-pool-154f511d-5nk5  gcr.io/run-ai-demo/quickstart          Train   team-a   yaron  1

```

若要取得工作的其他狀態、請執行下列命令：

```
$ runai get hyper1
```

若要檢視工作記錄、請執行「Runai logs <job-name>」命令：

```
$ runai logs hyper1
```

在此範例中、您應該會看到正在執行的DL工作階段記錄、包括目前的訓練時期、ETA、Loss Function Value、Accuracy、以及每個步驟所經過的時間。

您可以在的Run:AI UI上檢視叢集狀態 "<https://app.run.ai/>"。在「儀表板」>「總覽」下、您可以監控GPU使用率。

若要停止此工作負載、請執行下列命令：

```
$ runai delete hyper1
```

此命令可停止訓練工作負載。您可以再次執行「Runai list」（Runai清單）來驗證此動作。如需詳細資訊、請參閱 "[啟動無人管理的訓練工作負載](#)"。

互動式建置工作負載

設定專案並分配GPU之後、您可以在命令列使用下列命令來執行互動式建置工作負載：

```
$ runai submit build1 -i python -g 1 --interactive --command sleep --args infinity
```

工作是以範例Docker影像python為基礎。我們將工作建置命名為「1」。



「互動」旗標表示工作沒有開始或結束研究人員有責任完成這項工作。系統管理員可以定義互動工作的時間限制、之後系統會終止這些工作。

「-g 1」旗標會將單一GPU配置給此工作。提供的命令與引數是「-command sleep- args infinity」。您必須提供命令、否則容器會立即啟動並結束。

下列命令的運作方式與中所述的命令類似 [\[無人管理的訓練工作負載\]](#)：

- 「Runai list」：顯示名稱、狀態、年齡、節點、映像、專案、使用者及GPU的工作。
- 「Runai Get build1」：在工作建置1上顯示其他狀態。

- "Runai DELETE build1"：停止互動式工作負載建置1。若要將Bash Shell移至容器、請執行下列命令：

```
$ runai bash build1
```

這可直接在電腦中提供Shell。然後、資料科學家可以在容器內開發或微調模型。

您可以在的Run:AI UI上檢視叢集狀態 "<https://app.run.ai>"。如需詳細資訊、請參閱 "[啟動及使用互動式建置工作負載](#)"。

互動式工作負載與連接的連接埠

作為互動式建置工作負載的延伸、您可以在使用Run:AI CLI啟動容器時、向容器使用者顯示內部連接埠。這對於雲端環境、使用Jupyter筆記型電腦或連線至其他微服務都很有用。"[入侵](#)" 允許從Kubernetes叢集外部存取Kubernetes服務。您可以建立規則集合來定義哪些傳入連線可到達哪些服務、藉此設定存取。

為了更妥善管理叢集中的外部服務存取、建議叢集管理員安裝 "[入侵](#)" 並設定負載平衡器。

若要使用Ingress做為服務類型、請執行下列命令、在提交工作負載時設定方法類型和連接埠：

```
$ runai submit test-ingress -i jupyter/base-notebook -g 1 \
--interactive --service-type=ingress --port 8888 \
--args="--NotebookApp.base_url=test-ingress" --command=start-notebook.sh
```

容器成功啟動後、執行「Runai list」（執行清單）以查看用來存取Jupyter Notebook的「服務URL（S）」。URL由入口端點、工作名稱和連接埠組成。例如、請參閱 <https://10.255.174.13/test-ingress-8888>。

如需詳細資料、請參閱 "[使用連接的連接埠啟動互動式建置工作負載](#)"。

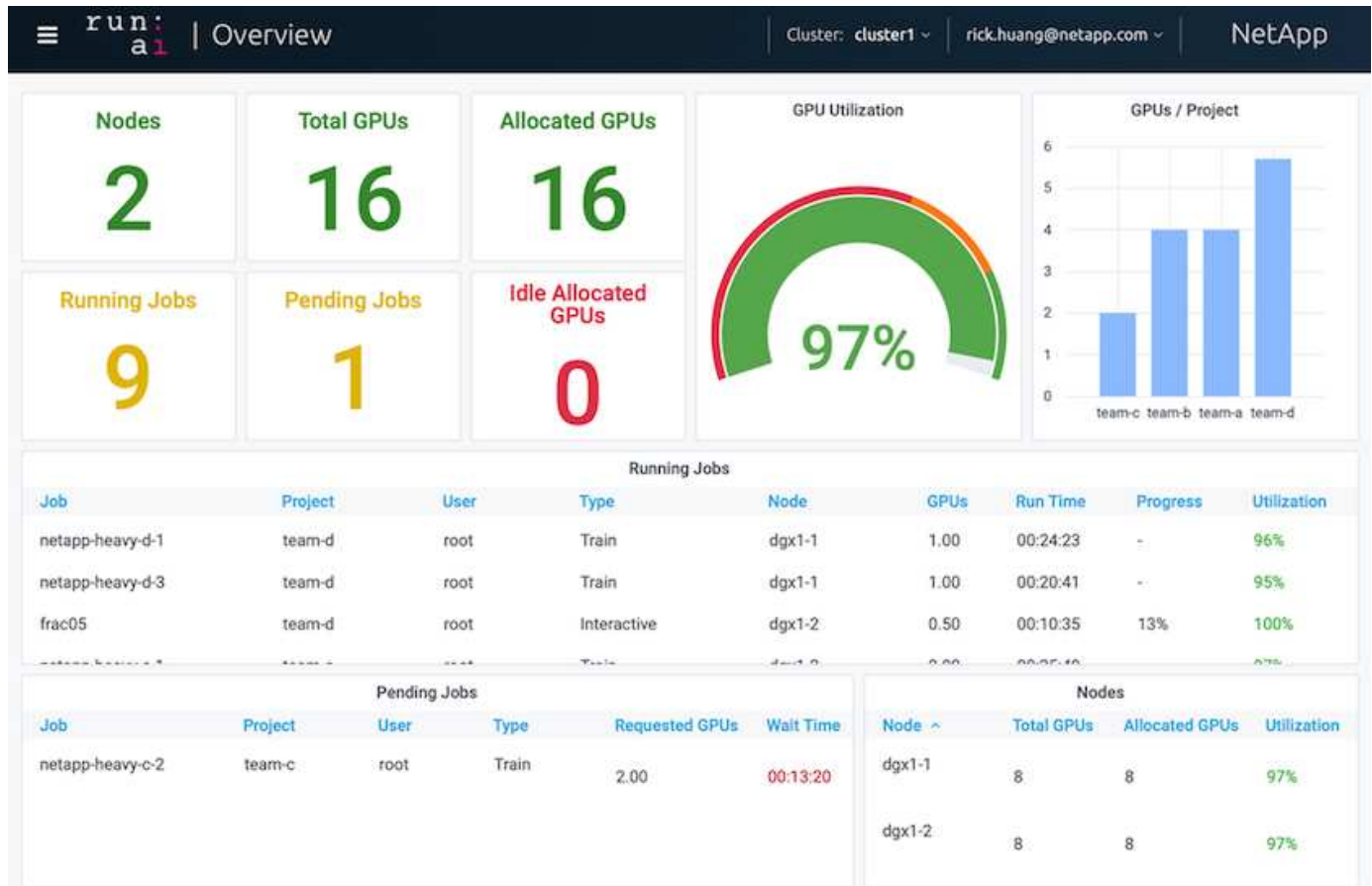
實現高叢集使用率

在本節中、我們模擬一個實際案例、其中四個資料科學團隊各自提交自己的工作負載、以展示Run:AI協調解決方案、在維持優先順序和平衡GPU資源的同時、達到高叢集使用率。我們首先使用一節中所述的ResNet-50基準測試 "[ResNet-50搭配ImageNet資料集基準測試摘要](#)"：

```
$ runai submit netapp1 -i netapp/tensorflow-tfl-py3:20.01.0 --local-image
--large-shm -v /mnt:/mnt -v /tmp:/tmp --command python --args
"/netapp/scripts/run.py" --args "--
dataset_dir=/mnt/mount_0/dataset/imagenet/imagenet_original/" --args "--
num_mounts=2" --args "--dgx_version=dgx1" --args "--num_devices=1" -g 1
```

我們執行的ResNet-50基準測試與中相同 "[NVA-1121](#)"。我們使用旗標「-本機映像」來表示未駐留在公用泊塢視窗儲存庫中的容器。我們將主機DGX-1節點上的目錄「/mnt」和「/tmp」分別掛載到容器上。資料集位於NetApp AFFA800、其中「dataset_dir」引數指向目錄。無論是「-num_devices=1」或「-g 1」、我們都會為此工作分配一個GPU。前者是所謂的「run.py」指令碼、後者則是「runai submit」指令的旗標。

下圖顯示系統總覽儀表板、其中GPU使用率達97%、並已配置全部16個可用GPU。您可以在GPU /專案長條圖中輕鬆查看每個團隊分配的GPU數量。「執行中工作」窗格會顯示目前執行中的工作名稱、專案、使用者、類型、節點、GPU使用量、執行時間、進度和使用率詳細資料。佇列中的工作負載與其等待時間清單、會顯示在「擱置工作」中。最後、節點方塊會針對叢集中的個別DGX-1節點、提供GPU數量和使用率。



部分GPU配置、適用於要求較低或互動性較差的工作負載

當研究人員和開發人員在開發、超參數調校或偵錯階段、都在研究他們的模型時、這類工作負載通常需要較少的運算資源。因此、配置部分GPU和記憶體的效率較高、因此同一個GPU可同時分配給其他工作負載。RUN：AI的協調化解決方案為Kubernetes上的容器化工作負載提供部分GPU共享系統。系統支援執行CUDA程式的工作負載、特別適合輕量化AI工作、例如推斷和建構模型。部分GPU系統可讓資料科學和AI工程團隊在單一GPU上同時執行多個工作負載。如此一來、公司就能在同一個硬體上執行更多工作負載、例如電腦視覺、語音辨識和自然語言處理、進而降低成本。

RUN：AI的部分GPU系統可有效建立虛擬化邏輯GPU、並提供其專屬的記憶體與運算空間、讓容器如同獨立的處理器一樣使用及存取。如此一來、多個工作負載就能在同一個GPU的容器中並排執行、而不會互相干擾。此解決方案透明、簡單且可攜、不需變更容器本身。

典型的usecase可以看到在同一個GPU上執行兩到八個工作、也就是說、您可以使用相同的硬體來執行八倍的工作。

從下圖中的項目"team d (團隊d)"的"Fract05 (分裂05)"工作中可以看出、分配給GPU的數量是0.5。「nvidia-smi」命令可進一步驗證、顯示容器可用的GPU記憶體為16255MB：DGX-1節點每V100 GPU 32GB的一半。

```

root@run-deploy:~# runai bash frac05 -p team-d
root@frac05-0:/workload# nvidia-smi
Tue Jul 28 15:17:03 2020
+-----+
| NVIDIA-SMI 450.51.05      Driver Version: 450.51.05      CUDA Version: 11.0      |
+-----+-----+-----+-----+-----+-----+
| GPU   Name                Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan   Temp   Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|                                           MIG M.         |
+-----+-----+-----+-----+-----+-----+
|  0    Tesla V100-SXM2...    On      | 00000000:07:00.0 Off |                    0 |
| N/A   57C    P0     240W / 300W | 15525MiB / 16255MiB |   100%    Default   |
|                                           N/A              |
+-----+-----+-----+-----+-----+-----+
+-----+
| Processes:                                     |
|  GPU   GI    CI          PID    Type   Process name                      GPU Memory |
|          ID    ID                                   Usage      |
+-----+-----+-----+-----+-----+-----+
|  0     N/A  N/A         156      C      python3                          15525MiB |
+-----+

```

透過過度配額GPU配置、實現高叢集使用率

在本節和各節中 "[基本資源配置公平性](#)"和 "[過度配額的公平性](#)"我們設計了進階測試案例、以展示Run:AI協調功能、適用於複雜的工作負載管理、自動優先排程、以及過度配額GPU資源配置。我們這樣做是為了在ONTAP 整個AI環境中、達到高叢集資源使用率、並最佳化企業級資料科學團隊的生產力。

在這三個區段中、設定下列專案和配額：

專案	配額
團隊A	4.
團隊b	2.
團隊	2.
團隊	8.

此外、這三個區段還使用下列容器：

- Jupyter Notebook：《[jupyter/base筆記型電腦](#)》
- RUN：AI quickstart：「[GCR.IO/RUN-AI-DEMO / quickstart](#)」

我們為此測試案例設定下列目標：

- 展現資源配置的簡易性、以及如何從使用者中抽取資源

- 示範使用者如何輕鬆配置GPU的一部分和GPU的整數數目
- 示範系統如何排除運算瓶頸、讓團隊或使用者在叢集中有可用的GPU時、可以跳過資源配額
- 示範如何在執行運算密集工作（例如NetApp容器）時、使用NetApp解決方案來消除資料管線瓶頸
- 示範多種容器如何使用系統執行
 - Jupyter筆記型電腦
 - 執行：AI Container
- 叢集已滿時顯示高使用率

如需測試期間實際執行命令順序的詳細資料、請參閱 ["第4.8節的測試詳細資料"](#)。

提交所有13項工作負載時、您會看到已分配的容器名稱和GPU清單、如下圖所示。我們有七項訓練和六項互動工作、模擬四個資料科學團隊、每個團隊都有自己的執行或開發模式。對於互動工作、個別開發人員使用Jupyter Notebooks來撰寫或偵錯其程式碼。因此、它適合在不使用太多叢集資源的情況下配置GPU分數。

```
root@run-deploy:~# kubectl get pods -A
```

NAME	STATUS	AGE	NODE	IMAGE	TYPE	PROJECT	USER	GPUs	CREATED BY	CLI	SERVICE URL(S)
b-4-gg	Running	2m	dgx1-2	gcr.io/run-ai-demo/quickstart	Train	team-b	root	2	true		
c-5-g	Running	2m	dgx1-2	gcr.io/run-ai-demo/quickstart	Train	team-c	root	1	true		
c-4-gg	Running	2m	dgx1-1	gcr.io/run-ai-demo/quickstart	Train	team-c	root	2	true		
b-3-g	Running	2m	dgx1-1	gcr.io/run-ai-demo/quickstart	Train	team-b	root	1	true		
c-3-g02	Running	2m	dgx1-1	gcr.io/run-ai-demo/quickstart	Interactive	team-c	root	0.2	true		
d-1-gggg	Running	2m	dgx1-2	gcr.io/run-ai-demo/quickstart	Train	team-d	root	4	true		
c-2-g03	Running	2m	dgx1-1	gcr.io/run-ai-demo/quickstart	Interactive	team-c	root	0.3	true		
c-1-g05	Running	2m	dgx1-1	gcr.io/run-ai-demo/quickstart	Interactive	team-c	root	0.5	true		
a-2-gg	Running	3m	dgx1-1	gcr.io/run-ai-demo/quickstart	Train	team-a	root	2	true		
b-2-g04	Running	3m	dgx1-2	gcr.io/run-ai-demo/quickstart	Interactive	team-b	root	0.4	true		
a-1-g	Running	3m	dgx1-1	gcr.io/run-ai-demo/quickstart	Train	team-a	root	1	true		
b-1-g06	Running	3m	dgx1-2	gcr.io/run-ai-demo/quickstart	Interactive	team-b	root	0.6	true		
a-1-1-jupyter	Running	3m	dgx1-1	jupyter/base-notebook	Interactive	team-a	root	1	true		http://10.61.218.134/a-1-1-jupyter

此測試案例的結果顯示下列項目：

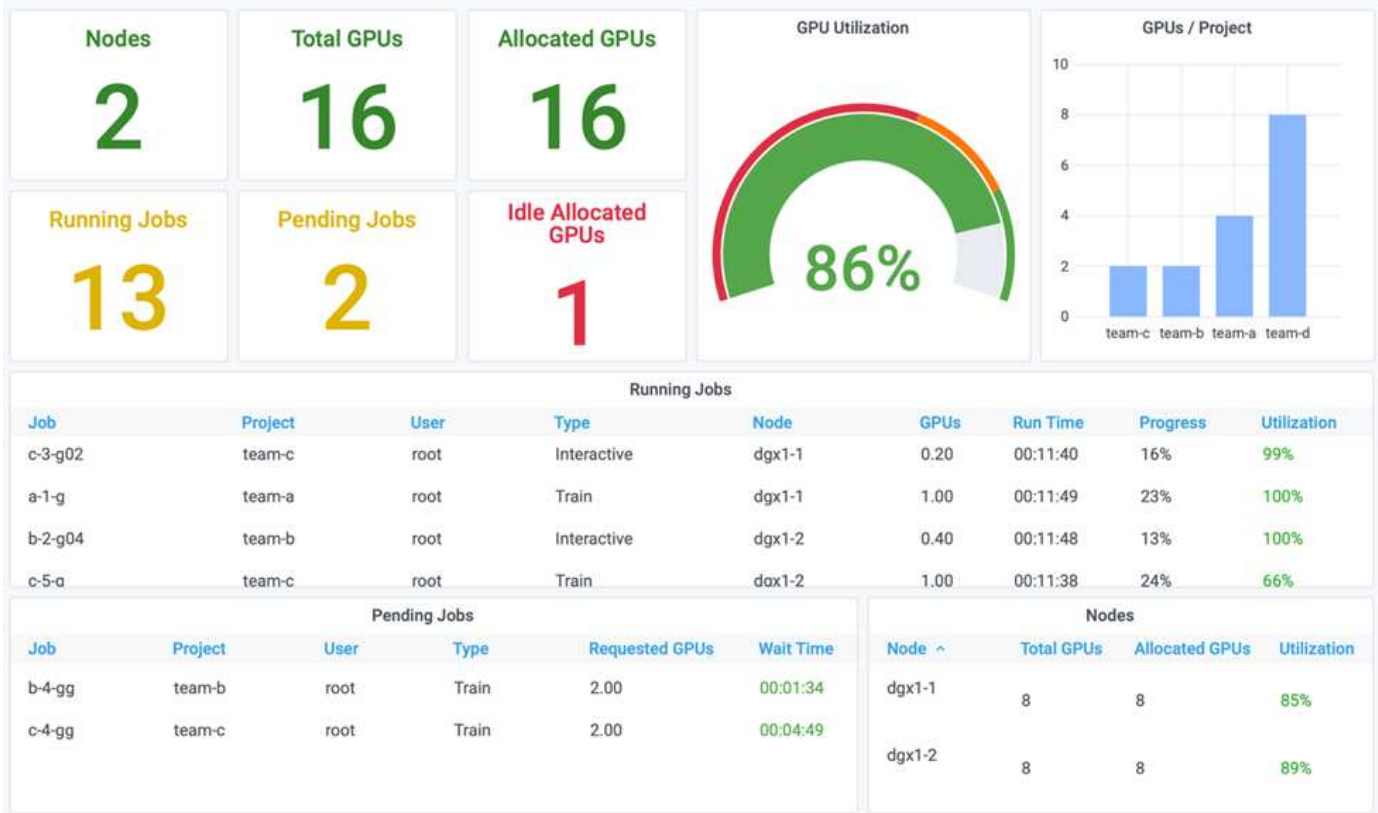
- 叢集應已滿：使用16/16 GPU。
- 叢集使用率高。
- 由於分數分配、實驗量比GPU多。
- 因為，“Team d”並沒有全部使用配額，所以“team-b”和“team-c”可以使用額外的GPU進行實驗，從而縮短創新時間。

基本資源配置公平性

在本節中、我們顯示、當「team d」要求更多GPU（以配額為限）時、系統會暫停「team b」和「team c」的工作負載、並以公平的共享方式將其移至待處理狀態。

如需工作提交、使用的容器映像及執行命令順序等詳細資料、請參閱一節 ["第4.9節的測試詳細資料"](#)。

下圖顯示所產生的叢集使用率、每個群組分配的GPU、以及由於自動負載平衡和優先排程而產生的擱置工作。我們可以觀察到、當所有團隊工作負載所要求的GPU總數超過叢集中可用的GPU總數時、Run:AI的內部公平演算法會因為達到專案配額、而在「team b」和「team -c」中各暫停一項工作。這可提供整體高叢集使用率、而資料科學團隊仍在系統管理員設定的資源限制下工作。



此測試案例的結果顯示下列項目：

- *自動負載平衡。*系統會自動平衡GPU的配額、使每個團隊現在都使用配額。暫停的工作負載屬於超出配額的團隊。
- *公平共用暫停。*系統會選擇停止某個團隊的工作負載、使其超出配額、然後停止另一個團隊的工作負載。RUN：AI具有內部公平演算法。

過度配額的公平性

在本節中、我們將展開多個團隊提交工作負載並超過其配額的案例。如此一來、我們就能示範Run：AI的公平演算法如何根據預設配額的比率來配置叢集資源。

本測試案例的目標：

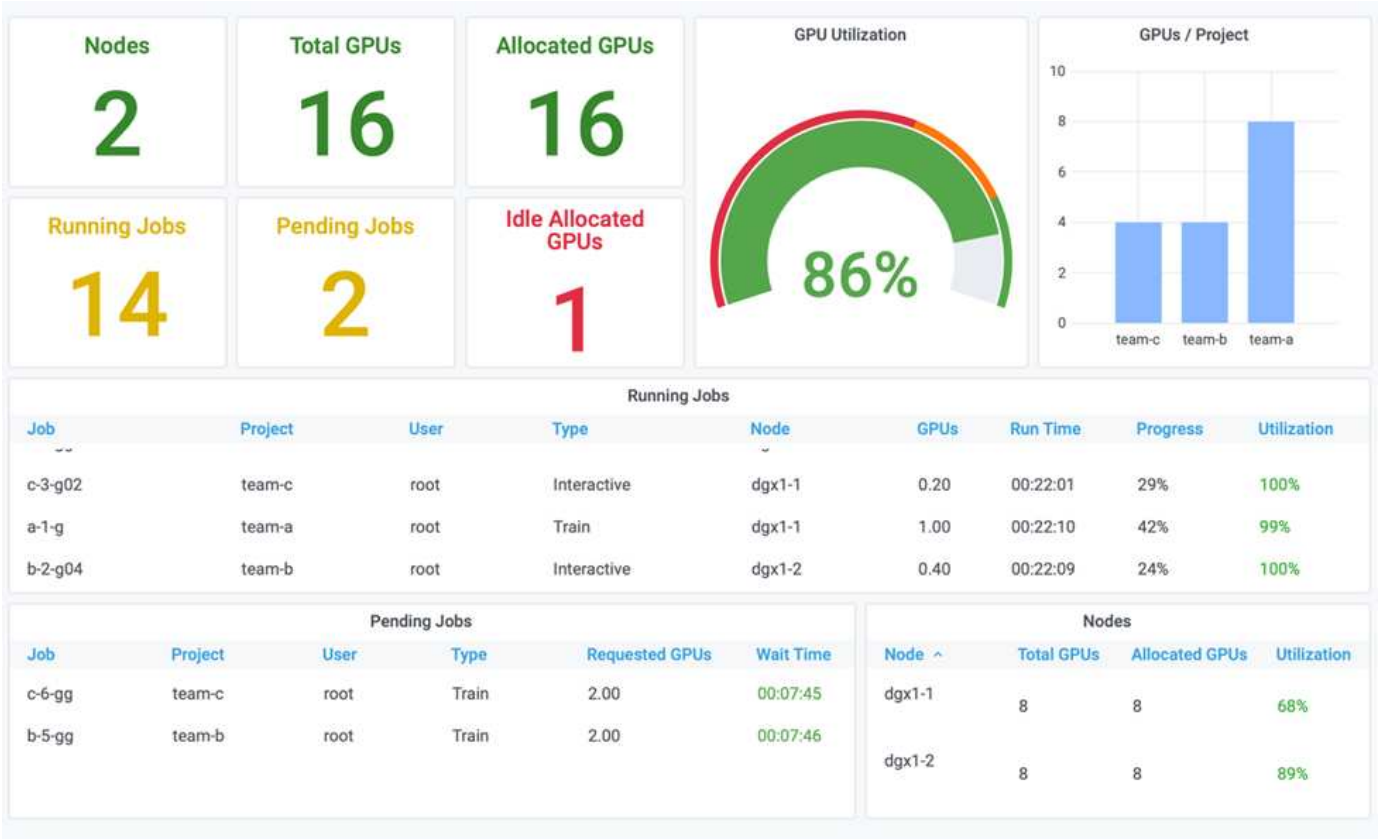
- 當多個團隊要求GPU超出配額時、顯示佇列機制。
- 示範系統如何根據配額之間的比率、在多個超出配額的團隊之間分配公平的叢集共用區、讓具有較大配額的團隊獲得較大的備用容量份額。

結束時 "[基本資源配置公平性](#)"有兩個工作負載排入佇列：一個用於「team b」、一個用於「team c」。在本節中、我們會排入其他工作負載的佇列。

如需工作提交、使用的容器映像及執行命令順序等詳細資料、請參閱 "[第4.10節的測試詳細資料](#)"。

當所有工作都根據區段提交時 "[第4.10節的測試詳細資料](#)"系統儀表板顯示：“team-a（團隊A）”、“team-b（團隊b）”和“team-c（團隊c）”都比預設配額多出GPU。與預設的軟配額（4個）相比、「團隊A」（team A）佔據4個GPU、而「團隊b」（team b）和「團隊c」（team c）則各佔2個GPU、而非軟配額（2個）。分配的

配額過多GPU比例等於預設配額的比例。這是因為當多個團隊要求更多GPU、超過配額時、系統會使用預設配額作為優先順序的參考資料、並據此進行資源配置。當企業資料科學團隊積極參與AI模式的開發與正式作業時、這種自動負載平衡可提供公平性和優先順序。



此測試案例的結果顯示下列項目：

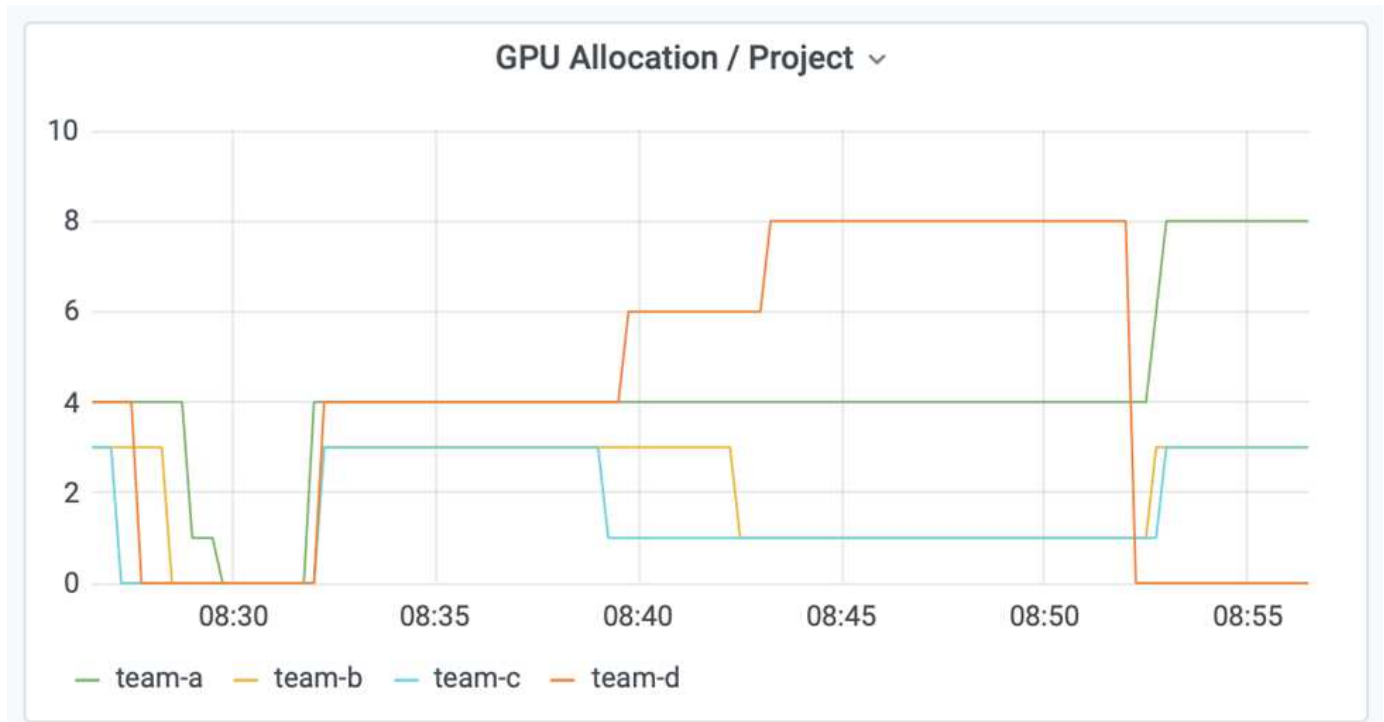
- 系統開始將其他團隊的工作負載取消佇列。
- 根據公平演算法決定取消佇列的順序、例如「team-b」和「team-c」獲得相同數量的配額GPU（因為配額相似）、而A組的配額比B組和C組的配額高出兩倍、因此獲得兩倍的GPU數量。
- 所有分配都會自動完成。

因此、系統應穩定在下列狀態：

專案	已分配的GPU	留言
團隊A	8/4.	超過配額的四個GPU。空佇列。
團隊b	4/2.	超過配額的兩個GPU。一個工作負載已排入佇列。
團隊	4/2.	超過配額的兩個GPU。一個工作負載已排入佇列。
團隊	0/8.	完全不使用GPU、沒有排入佇列的工作負載。

下圖顯示各區段的「執行：AI分析」儀表板中、隨著時間推移、每個專案的GPU配置 "透過過度配額GPU配置、實現高叢集使用率"、"基本資源配置公平性"和 "過度配額的公平性"。圖中的每一行都會指出任何時間為特定資料科學團隊配置的GPU數量。我們可以看到、系統會根據提交的工作負載動態配置GPU。如此一來、當叢集中

有可用的GPU時、團隊就能跳過配額、然後根據公平原則預先部署工作、最後才會達到四個團隊的穩定狀態。



將資料儲存至Trident佈建的PersistentVolume

NetApp Trident是完全受支援的開放原始碼專案、旨在協助您滿足容器化應用程式的複雜持續需求。您可以將資料讀寫至Trident佈建的Kubernetes PersistentVolume (PV)、並享有NetApp ONTAP 供應的資料分層、加密、NetApp Snapshot技術、法規遵循及高效能等額外優點。

重新使用現有命名空間中的PVCS

對於較大型的AI專案、不同的容器可能會更有效率地讀取資料並將資料寫入相同的Kubernetes PV。若要重複使用Kubernetes持續Volume宣告 (PVC)、使用者必須已建立一個PVC。請參閱 ["NetApp Trident文件"](#) 以取得建立永久虛擬基礎資料的詳細資料。以下是重複使用現有的永久虛擬資料的範例：

```
$ runai submit pvc-test -p team-a --pvc test:/tmp/pvc1mount -i gcr.io/run-ai-demo/quickstart -g 1
```

執行下列命令、查看專案「團隊A」的「PVC-TEST」工作狀態：

```
$ runai get pvc-test -p team-a
```

您應該會看到PV /tmp/pvc1mount掛載到「team A」工作「PVC-test」。如此一來、多個容器就能從同一個磁碟區讀取、當開發或正式作業中有多個相競的模型時、此功能非常實用。資料科學家可以建立一組模型、然後以多數投票或其他技術來結合預測結果。

使用下列項目存取Container Shell：

```
$ runai bash pvc-test -p team-a
```

然後、您可以檢查掛載的Volume、並存取容器內的資料。

這項可重複使用PVCS的功能可搭配NetApp FlexVol 的功能、以及NetApp ONTAP FlexGroup 的NetApp功能、讓資料工程師擁有更靈活、更健全的資料管理選項、以充分運用NetApp技術所提供的資料架構。

結論

NetApp與Run：AI已在本技術報告中合作、展示NetApp ONTAP AI解決方案的獨特功能、並搭配Run:AI平台、以簡化AI工作負載的協調作業。上述步驟提供參考架構、可簡化資料管線和工作負載協調流程、以利深度學習。我們鼓勵想要實作這些解決方案的客戶聯絡NetApp、並聯絡Run：AI以取得更多資訊。

第4.8節的測試詳細資料

本節包含本節的測試詳細資料 ["透過過度配額GPU配置、實現高叢集使用率"](#)。

依下列順序提交工作：

專案	映像	# GPU	總計	留言
團隊A	Jupyter	1.	1/4	—
團隊A	NetApp	1.	2/4.	—
團隊A	執行：AI	2.	4/4.	使用所有配額
團隊b	執行：AI	0.6	0.6/2.	部分GPU
團隊b	執行：AI	0.4	1/2	部分GPU
團隊b	NetApp	1.	2/2.	—
團隊b	NetApp	2.	4/2.	超過配額2個
團隊	執行：AI	0.5	0.5/2.	部分GPU
團隊	執行：AI	0.3	0.8/2.	部分GPU
團隊	執行：AI	0.2	1/2	部分GPU
團隊	NetApp	2.	3/2.	一個超出配額
團隊	NetApp	1.	4/2.	超過配額2個
團隊	NetApp	4.	4/8.	使用一半配額

命令結構：

```
$ runai submit <job-name> -p <project-name> -g <#GPUs> -i <image-name>
```

測試中使用的實際命令順序：

```
$ runai submit a-1-1-jupyter -i jupyter/base-notebook -g 1 \
--interactive --service-type=ingress --port 8888 \
--args="--NotebookApp.base_url=team-a-test-ingress" --command=start
-notebook.sh -p team-a
$ runai submit a-1-g -i gcr.io/run-ai-demo/quickstart -g 1 -p team-a
$ runai submit a-2-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-a
$ runai submit b-1-g06 -i gcr.io/run-ai-demo/quickstart -g 0.6
--interactive -p team-b
$ runai submit b-2-g04 -i gcr.io/run-ai-demo/quickstart -g 0.4
--interactive -p team-b
$ runai submit b-3-g -i gcr.io/run-ai-demo/quickstart -g 1 -p team-b
$ runai submit b-4-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-b
$ runai submit c-1-g05 -i gcr.io/run-ai-demo/quickstart -g 0.5
--interactive -p team-c
$ runai submit c-2-g03 -i gcr.io/run-ai-demo/quickstart -g 0.3
--interactive -p team-c
$ runai submit c-3-g02 -i gcr.io/run-ai-demo/quickstart -g 0.2
--interactive -p team-c
$ runai submit c-4-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-c
$ runai submit c-5-g -i gcr.io/run-ai-demo/quickstart -g 1 -p team-c
$ runai submit d-1-gggg -i gcr.io/run-ai-demo/quickstart -g 4 -p team-d
```

此時、您應具備下列狀態：

專案	已分配的GPU	已排入佇列的工作負載
團隊A	4/4（軟配額/實際分配）	無
團隊b	4/2.	無
團隊	4/2.	無
團隊	4/8.	無

請參閱一節 ["透過過度配置GPU來實現高叢集使用率"](#) 以討論後續的測試案例。

第4.9節的測試詳細資料

本節包含本節的測試詳細資料 ["基本資源配置公平性"](#)。

依下列順序提交工作：

專案	# GPU	總計	留言
團隊	2.	6/8.	team b/c/ c工作負載暫停並移至「待處理」。

專案	# GPU	總計	留言
團隊	2.	8/8.	其他團隊 (b / c) 工作負載暫停、移至「待處理」。

請參閱下列執行的命令順序：

```
$ runai submit d-2-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-d$
runai submit d-3-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-d
```

此時、您應具備下列狀態：

專案	已分配的GPU	已排入佇列的工作負載
團隊A	4/4.	無
團隊b	2/2.	無
團隊	2/2.	無
團隊	8/8.	無

請參閱一節 "[基本資源配置公平性](#)" 以討論後續測試案例。

第4.10節的測試詳細資料

本節包含本節的測試詳細資料 "[過度配額的公平性](#)"。

請依下列順序提交工作：「team a」、「team-b」和「team-c」：

專案	# GPU	總計	留言
團隊A	2.	4/4.	1個工作負載排入佇列
團隊A	2.	4/4.	2個工作負載排入佇列
團隊b	2.	2/2.	2個工作負載排入佇列
團隊	2.	2/2.	2個工作負載排入佇列

請參閱下列執行的命令順序：

```
$ runai submit a-3-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-a$
runai submit a-4-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-a$ runai
submit b-5-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-b$ runai
submit c-6-gg -i gcr.io/run-ai-demo/quickstart -g 2 -p team-c
```

此時、您應具備下列狀態：

專案	已分配的GPU	已排入佇列的工作負載
團隊A	4/4.	兩個工作負載分別要求GPU
團隊b	2/2.	兩個工作負載分別要求兩個GPU
團隊	2/2.	兩個工作負載分別要求兩個GPU
團隊	8/8.	無

接下來、刪除「team d」的所有工作負載：

```
$ runai delete -p team-d d-1-gggg d-2-gg d-3-gg
```

請參閱一節 "[過度配額的公平性](#)"、以討論後續的測試案例。

何處可找到其他資訊

若要深入瞭解本文所述資訊、請參閱下列資源：

- NVIDIA DGX系統
 - NVIDIA DGX-1系統<https://www.nvidia.com/en-us/data-center/dgx-1/>
 - NVIDIA V100 Tensor Core GPU<https://www.nvidia.com/en-us/data-center/tesla-v100/>
 - NVIDIA NGC<https://www.nvidia.com/en-us/gpu-cloud/>
- RUN：AI Container協調解決方案
 - RUN：AI產品簡介<https://docs.run.ai/home/components/>
 - 執行：AI安裝文件<https://docs.run.ai/Administrator/Cluster-Setup/Installing-Run-AI-on-an-on-premise-Kubernetes-Cluster/>
<https://docs.run.ai/Administrator/Researcher-Setup/Installing-the-Run-AI-Command-Line-Interface/>
 - 在Run:AI CLI中提交工作<https://docs.run.ai/Researcher/Walkthroughs/Walkthrough-Launch-Unattended-Training-Workloads-/>
<https://docs.run.ai/Researcher/Walkthroughs/Walkthrough-Start-and-Use-Interactive-Build-Workloads-/>
 - 在Run:AI CLI中分配GPU分數<https://docs.run.ai/Researcher/Walkthroughs/Walkthrough-Using-GPU-Fractions/>
- NetApp AI Control Plane
 - 技術報告<https://www.netapp.com/us/media/tr-4798.pdf>
 - 簡短示範https://youtu.be/gfr_sO27Rvo
 - GitHub儲存庫https://github.com/NetApp/kubeflow_jupyter_pipeline
- NetApp AFF 產品
 - NetApp AFF A系列產品型錄<https://www.netapp.com/us/media/ds-3582.pdf>
 - 適用於All Flash FAS 的NetApp Flash優勢<https://www.netapp.com/us/media/ds-3733.pdf>
 - 資訊庫ONTAP<http://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286>

- NetApp ONTAP FlexGroup 《銷售區》技術報告<https://www.netapp.com/us/media/tr-4557.pdf>
- NetApp ONTAP AI
 - 《採用DGX-1與Cisco Networking設計指南的AI》 ONTAP<https://www.netapp.com/us/media/nva-1121-design.pdf>
 - 《採用DGX-1和Cisco Networking部署指南的AI》 ONTAP<https://www.netapp.com/us/media/nva-1121-deploy.pdf>
 - 《採用DGX-1和Mellanox網路設計指南的AI》 ONTAP<http://www.netapp.com/us/media/nva-1138-design.pdf>
 - 《採用DGX-2設計指南的AI》 ONTAP<https://www.netapp.com/us/media/nva-1135-design.pdf>

版權資訊

Copyright © 2024 NetApp, Inc. 版權所有。台灣印製。非經版權所有人事先書面同意，不得將本受版權保護文件的任何部分以任何形式或任何方法（圖形、電子或機械）重製，包括影印、錄影、錄音或儲存至電子檢索系統中。

由 NetApp 版權資料衍伸之軟體必須遵守下列授權和免責聲明：

此軟體以 NETAPP「原樣」提供，不含任何明示或暗示的擔保，包括但不限於有關適售性或特定目的適用性之擔保，特此聲明。於任何情況下，就任何已造成或基於任何理論上責任之直接性、間接性、附隨性、特殊性、懲罰性或衍生性損害（包括但不限於替代商品或服務之採購；使用、資料或利潤上的損失；或企業營運中斷），無論是在使用此軟體時以任何方式所產生的契約、嚴格責任或侵權行為（包括疏忽或其他）等方面，NetApp 概不負責，即使已被告知有前述損害存在之可能性亦然。

NetApp 保留隨時變本文所述之任何產品的權利，恕不另行通知。NetApp 不承擔因使用本文所述之產品而產生的責任或義務，除非明確經過 NetApp 書面同意。使用或購買此產品並不會在依據任何專利權、商標權或任何其他 NetApp 智慧財產權的情況下轉讓授權。

本手冊所述之產品受到一項（含）以上的美國專利、國外專利或申請中專利所保障。

有限權利說明：政府機關的使用、複製或公開揭露須受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中的「技術資料權利 - 非商業項目」條款 (b)(3) 小段所述之限制。

此處所含屬於商業產品和 / 或商業服務（如 FAR 2.101 所定義）的資料均為 NetApp, Inc. 所有。根據本協議提供的所有 NetApp 技術資料和電腦軟體皆屬於商業性質，並且完全由私人出資開發。美國政府對於該資料具有非專屬、非轉讓、非轉授權、全球性、有限且不可撤銷的使用權限，僅限於美國政府為傳輸此資料所訂合約所允許之範圍，並基於履行該合約之目的方可使用。除非本文另有規定，否則未經 NetApp Inc. 事前書面許可，不得逕行使用、揭露、重製、修改、履行或展示該資料。美國政府授予國防部之許可權利，僅適用於 DFARS 條款 252.227-7015(b)（2014 年 2 月）所述權利。

商標資訊

NETAPP、NETAPP 標誌及 <http://www.netapp.com/TM> 所列之標章均為 NetApp, Inc. 的商標。文中所涉及的所有其他公司或產品名稱，均為其各自所有者的商標，不得侵犯。