



XPU CASE STUDY

Cognitive Neuroscience Research & AI Center gets to manage GPU resources and speed up work with an AI Platform.

OBJECTIVES

- A simple & effective AI Platform
- Dramatically improve current GPUs' utilization.
- Run many workloads on a GPU simultaneously.
- Allocate resources dynamically.
- Manage resources and schedule workloads automatically.



"XPU and OpenPAI combination is our lifesaver!"

Two powerhouses combined gave us a simple and user-friendly AI platform that greatly simplifies management, democratizes access to GPU resources, and significantly utilizes our GPUs."

DR. JUN LIEU

Chief Scientist at Beijing Cognitive Neuroscience Research & AI Center

CHALLENGES

Beijing cognitive neuroscience research & AI center combines the study of brain and cognitive science and artificial intelligence to solve a broad range of problems. It does so by developing AI models to analyze big data in neuroscience.

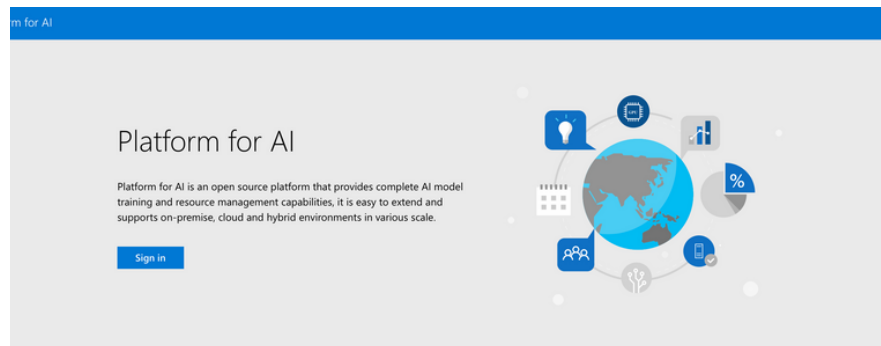
Lacking an effective AI infrastructure can be costly, and because of that, the Center has been struggling with very low productivity; teams can't access resources when they want to, results take "forever" to produce, and they even had to manage an Excel spreadsheet to track who gets to use the GPUs next...

As Dr. Lieu puts it: "As researchers, our goal is to make breakthroughs in scientific studies, not to develop or deal with IT. We want something that simply does what we want – submit jobs, see the results, and tune the parameters, that's it."

SOLUTIONS

The Research Center's challenges share among many organizations and research institutions. There is no easy-to-use AI platform to help them run the jobs effectively and manage the computing resources automatically.

XPU integrated with an open AI platform developed by Microsoft – OpenPAI, and met the challenges.



Submit a hello-world job

With submitting a hello-world job, this section introduces more knowledge about job, so that you can write your own job configuration easily.

[Learn more >](#)

Understand Job

The job of OpenPAI defines how to execute command(s) in specified environment(s). A job can be model training, other kinds of commands, or distributed on multiple servers.

[Learn more >](#)

Use VS Code Extension

OpenPAI Client is a VS Code extension to connect PAI clusters, submit AI jobs, and manage files on HDFS, etc. You need to install the extension in VS code before using it.

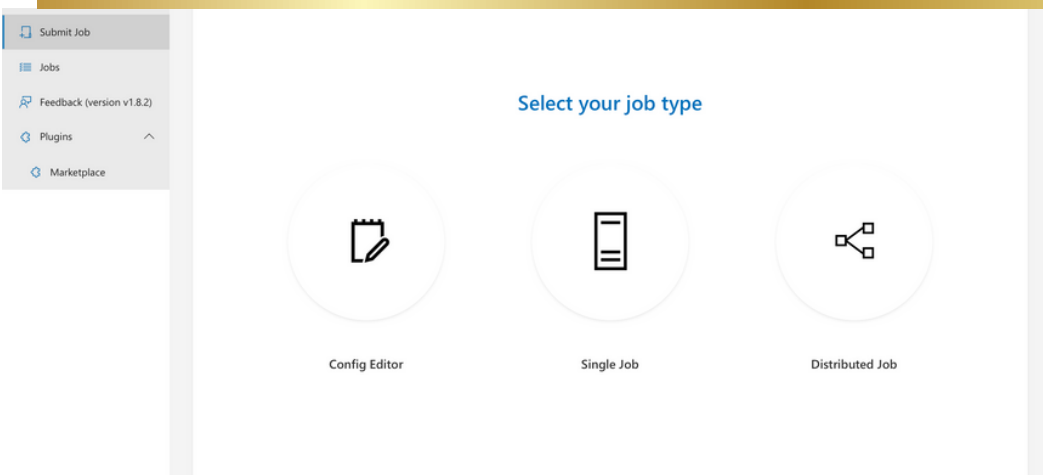
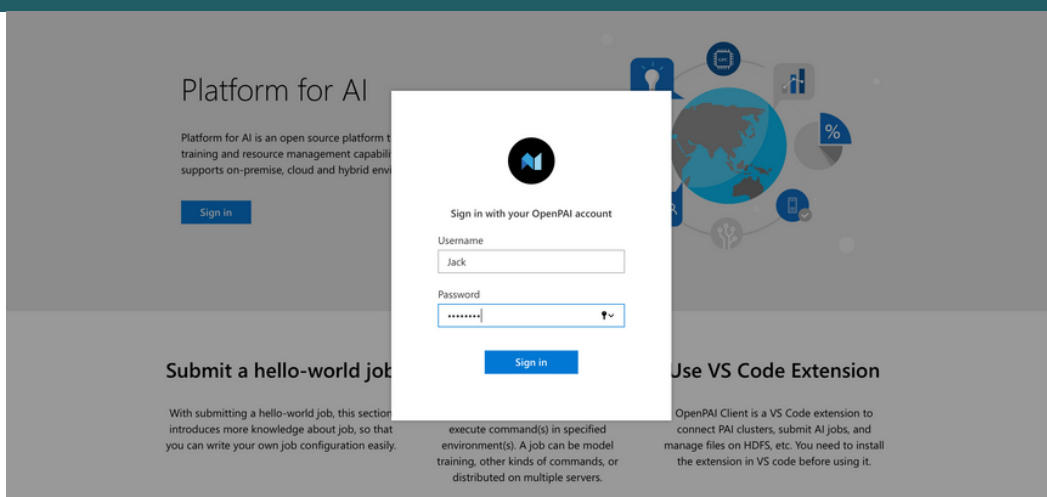
[Learn more >](#)



XPU CASE STUDY

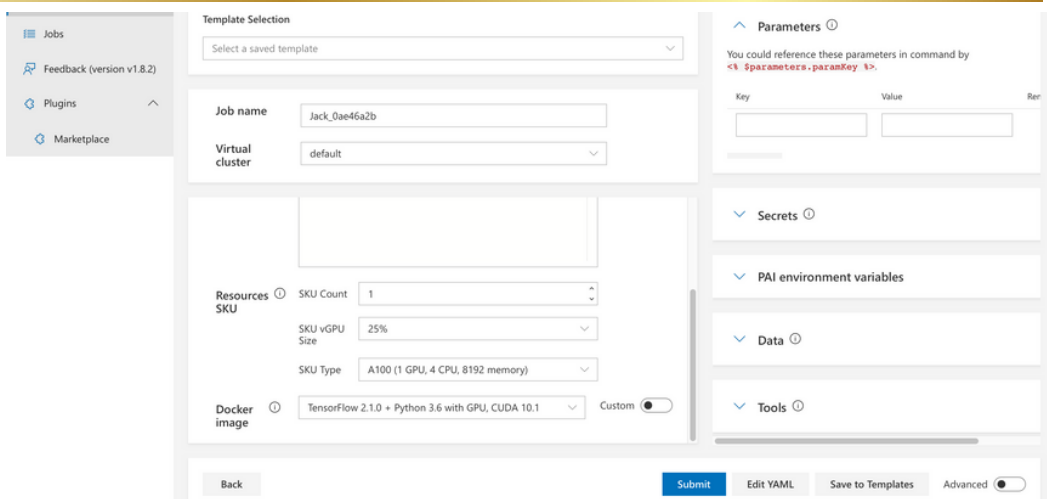
Cognitive Neuroscience Research & AI Center gets to manage GPU resources and speed up work with an AI Platform.

First, OpenPAI provides a simple and user-friendly interface; you can remotely login from anywhere and submit your jobs.



Next, submitting jobs is only a click away; you can choose either "Single Job" or "Distributed Job" you want to run.

You can then choose a complete suite of options, including selecting a Docker image, specifying computing resources, and many more.





XPU CASE STUDY

Cognitive Neuroscience Research & AI Center gets to manage GPU resources and speed up work with an AI Platform.

XPU redefined how GPU resources are accessed by integrating impeccably with OpenPAI; when you select the computing resources, you can choose only a portion of the GPU for the job. XPU takes care of the "splitting - virtualization" for them automatically, dynamically, and on demand. GPU is fully utilized in this fashion, and resource scheduling is automatically handled by OpenPAI's powerful "HiveD" scheduler; you don't have to worry about anything.

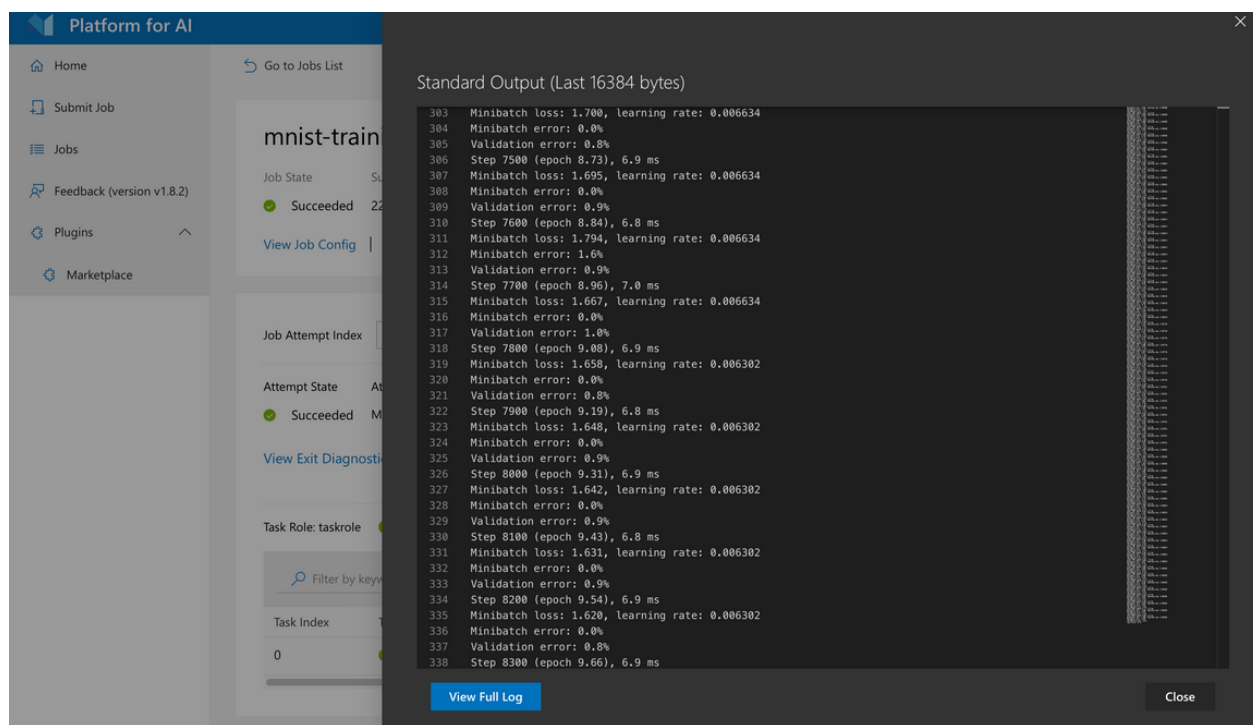
Resources ⓘ **SKU Count**

SKU **SKU vGPU Size**

SKU Type

Docker image ⓘ **Custom** ☐

When the job finishes, a simple click gets the results. You can either export it or tune the parameters as you wish. And the GPU resources will be returned to the pool automatically as well.



Platform for AI

Home | Go to Jobs List

Submit Job

Jobs

Feedback (version v1.8.2)

Plugins

Marketplace

mnist-train

Job State: Succeeded

View Job Config

Job Attempt Index

Attempt State: Succeeded

View Exit Diagnostic

Task Role: taskrole

Filter by key

Task Index: 0

Standard Output (Last 16384 bytes)

```
303 Minibatch loss: 1.700, learning rate: 0.006634
304 Minibatch error: 0.0%
305 Validation error: 0.8%
306 Step 7500 (epoch 8.73), 6.9 ms
307 Minibatch loss: 1.695, learning rate: 0.006634
308 Minibatch error: 0.0%
309 Validation error: 0.9%
310 Step 7600 (epoch 8.84), 6.8 ms
311 Minibatch loss: 1.794, learning rate: 0.006634
312 Minibatch error: 1.6%
313 Validation error: 0.9%
314 Step 7700 (epoch 8.96), 7.0 ms
315 Minibatch loss: 1.667, learning rate: 0.006634
316 Minibatch error: 0.0%
317 Validation error: 1.0%
318 Step 7800 (epoch 9.08), 6.9 ms
319 Minibatch loss: 1.658, learning rate: 0.006302
320 Minibatch error: 0.0%
321 Validation error: 0.8%
322 Step 7900 (epoch 9.19), 6.8 ms
323 Minibatch loss: 1.648, learning rate: 0.006302
324 Minibatch error: 0.0%
325 Validation error: 0.9%
326 Step 8000 (epoch 9.31), 6.9 ms
327 Minibatch loss: 1.642, learning rate: 0.006302
328 Minibatch error: 0.0%
329 Validation error: 0.9%
330 Step 8100 (epoch 9.43), 6.8 ms
331 Minibatch loss: 1.631, learning rate: 0.006302
332 Minibatch error: 0.0%
333 Validation error: 0.9%
334 Step 8200 (epoch 9.54), 6.9 ms
335 Minibatch loss: 1.620, learning rate: 0.006302
336 Minibatch error: 0.0%
337 Validation error: 0.8%
338 Step 8300 (epoch 9.66), 6.9 ms
```

View Full Log | Close