GRAPHCORE

Getting Started with IPU-POD Systems

Version latest

Graphcore Ltd

Aug 25, 2021

CONTENTS

CHAPTER

ONE

IPU-POD OVERVIEW

The IPU-POD™ is designed to make both training and inference of very large and demanding machine-learning models faster, more efficient, and more scalable. This enables very large and emergent models to be run most effectively.

The IPU-POD is constructed from a number of IPU-M2000s, each containing four IPUs. For example, the IPU-POD_{16} has four IPU-M2000s (16 IPUs), and the IPU-POD_{64} is built from 16 IPU-M2000s (64 IPUs). The number of IPUs in an IPU-POD must be a power of 2, and greater than or equal to 16.

IPU-Links provide communication between the IPUs in an IPU-M2000 and also between the IPU-M2000s in an IPU-POD. A gateway controller in the IPU-M2000 provides links (GW-Links) for high-speed, low-latency communication between IPU-POD racks.

1.1 V-IPU software

The Virtual-IPU™ (V-IPU™) IPU management software is used for allocating and configuring IPUs in the IPU-POD. It has command line support for both the Poplar user role and IPU admin role.

It consists of the following components:

- **V-IPU agents**: An agent resides on each IPU-M2000 in an IPU system and manages the IPU-M2000 hardware.
- **V-IPU controller**: The V-IPU controller runs on a management node. It is responsible for managing V-IPU agents.
- **V-IPU command-line interface**: Command line tools provide access to the administration and user functions of the V-IPU controller.

This document assumes you have access to an IPU-POD that has the V-IPU agents and controller installed and configured.

The V-IPU software should already be installed on the IPU-POD. You can check this with your system administrator. You will also need to ask them for the information you need to connect to the IPU-POD (see Section 3, Getting started with V-IPU).

If you are the system administrator, refer to the V-IPU Admin Guide for information on installing and using the V-IPU software.

1.2 Poplar SDK

The IPU-POD is fully supported by Graphcore's Poplar SDK to provide a complete, scalable platform for accelerated machine intelligence development.

The Poplar SDK contains tools for creating and running programs on IPU hardware using standard machine-learning frameworks such as PyTorch and TensorFlow. It also includes command line tools for managing IPU hardware.

CHAPTER

TWO

SOFTWARE INSTALLATION

In order to run models on an IPU-POD you will need to download and install some software packages. The software can be downloaded from the Graphcore software download portal.

You need to install the Poplar SDK, which includes the development tools and also command line tools for managing the IPU hardware.

You also need to install the user-level V-IPU software.

2.1 SDK installation

Download the SDK tarball and unpack it using the following command:

```
$ tar -xvzf poplar_sdk-[os]-[ver].tar.gz
```

Where `[os]` is the host OS and `[ver]` is the software version number of the package.

The main components under the SDK installation directory are shown in Table 2.1.

Table 2.1: SDK contents

File or directory	Description
gc_kernel-module-[os]-[ver]/	Directory containing the Graphcore drivers and utilities.
popart-[os]-[ver]/	Directory containing the PopART framework.
poplar-[os]-[ver]/	Directory containing the Poplar graph programming framework and PopLibs libraries
docs/	HTML and PDF documentation for the SDK tools and libraries. The documentation on the Graphcore documentation portal may contain updates made after the SDK release and includes other documents not packaged with the SDK.
tensorflow-1.15.4+[ver]-[arch].whl	File to install the Graphcore port of TensorFlow v1.15 for Python 3.
tensorflow-2.4.1+[ver]-[arch].whl	File to install the Graphcore port of TensorFlow v2.4 for Python 3.
poptorch-[ver]-[arch].whl	File to install PopTorch (to run PyTorch models on the IPU).
horovod-[ver].whl	File to install Horovod to support distributed training in PopART (see Distributed training with Horovod for more information).
poplibs-source-code-[ver].zip	Source code for the PopLibs libraries (also available on GitHub).
license.html	The Graphcore end user license agreement (EULA).
release_notes.*	Release notes for this version of the SDK, in HTML and XML (Docbook) format.

Note: There are two versions of each of the TensorFlow wheel files, optimised for Intel and AMD processors respectively. These are indicated by the `arch` component of the filename. You must install the correct wheel file for your host processor type.

2.2 Setting up the SDK environment

To use the Graphcore tools and Poplar libraries, several environment variables (such as library and binary paths) need to be set up, as shown below:

```
$ cd poplar_sdk-[os]-[ver]
$ source poplar-[os]-[ver]/enable.sh
$ source popart-[os]-[ver]/enable.sh
```

Where `[os]` is the host OS and `[ver]` is the current software version number of each package.

You will need to source both the Poplar and the PopART enable scripts if you are using PyTorch or PopART.

Note: Each of these scripts must be sourced every time the Bash shell is reset. If you attempt to run any Poplar software without having first enabled these scripts, you'll get an error from the C++ compiler similar to the following (the exact message will depend on your code).

```
fatal error: 'poplar/Engine.hpp' file not found
```

You can verify that Poplar has been successfully set up by running:

```
$ popc --version
```

This will display the version number of the installed software.

PopTorch and TensorFlow for the IPU are provided as Python wheel files that can be installed using `pip` as described in the following sections.

2.3 Setting up PyTorch for the IPU

PopTorch is part of the Poplar SDK. It provides functions to allow PyTorch models to run on the IPU.

Before running PopTorch, you must source the `enable.sh` scripts for Poplar and PopART as described in Section 2.2, Setting up the SDK environment.

PopTorch is packaged as a Python wheel file that can be installed using `pip`.

Note: PopTorch requires `pip` version 18.1 or later, so it important to make sure you have the latest version before installing PopTorch.

We recommend creating a virtual environment, using `virtualenv`, to isolate your PopTorch environment from the system Python environment. You can create a virtual environment in a `workspace` directory and install PopTorch as shown below:

```
$ virtualenv -p python3 ~/workspace/poptorch_env
$ source poptorch_env/bin/activate
$ pip3 install -U pip
$ pip3 install poptorch-[ver].whl
```

Where `[ver]` is the SDK version version.

To confirm that PopTorch has been installed, you can use `pip list`, which should include the `poptorch` package in the output.

You can also test that the module has been installed correctly by attempting to import it in Python, for example:

```
$ python3 -c "import poptorch; print(poptorch.__version__)"
```

For more information, refer to PyTorch for the IPU: User Guide.

2.4 Setting up TensorFlow for the IPU

Before running TensorFlow, you must source the `enable.sh` scripts for Poplar as described in Section 2.2, Setting up the SDK environment.

To use the Graphcore port of TensorFlow, you must set up a Python virtual environment.

You can create a virtual environment in a `workspace` directory and install TensorFlow as shown below:

```
$ virtualenv -p python3.6 ~/workspace/tensorflow_env
```

Then activate it.

```
$ source tensorflow_env/bin/activate
```

Now all installations will be local to that virtual environment.

We support TensorFlow 1 and TensorFlow 2. There are versions of these compiled for Intel and AMD processors to provide the best performance on those hosts. As a result, there are four Python wheel files that can be installed with `pip`.

Warning: You must install the correct wheel file for your host CPU. You can use the command `lscpu` to determine the CPU type, if you are not sure.

For example, to install Graphcore's TensorFlow distribution, compatible with v2.1.0 of TensorFlow, you would use a command similar to the following:

```
$ pip install tensorflow-2.1.0+[ver]+[arch].whl
```

Where `[ver]` is the TensorFlow version you have downloaded, and `[arch]` is the host CPU architecture (Intel or AMD).

To confirm that `tensorflow` has been installed, you can use `pip list`, which should include the `tensorflow` package in the output, for example:

```
(tensorflow_env) jsp$ pip list
Package        Version
-------------  ----------
future         0.18.2
numpy          1.19.5
pip            20.3.3
pkg-resources  0.0.0
tensorflow_env 2.1.0
setuptools     51.1.2
torch          1.6.0+cpu
wheel          0.36.2
```

You can also test that the module has been installed correctly by importing it in Python, for example:

```
$ python -c "from tensorflow.python import ipu"
```

For the next steps with TensorFlow, refer to the appropriate user guide:

- Targeting the IPU from TensorFlow 1.
- Targeting the IPU from TensorFlow 2.

2.5 Installing the V-IPU command-line tools

Note: You can omit this step if `vipu` is already installed on the system. You can check if it is installed by running `vipu --version`.

You can download the V-IPU client software from the Graphcore software download portal. It can be installed on any computer that can communicate with the V-IPU controller.

Extract the contents of the downloaded archive:

```
$ tar xzvf vipuusertools-$VERSION.tar.gz
```

Where `$VERSION` is the version of the software.

Add the directory containing the extracted package to the `PATH` environment variable:

```
$ export PATH=$PWD/vipu-$VERSION:$PATH
```

Now, confirm that the `vipu` executable is found and that it reports the expected version:

```
$ vipu --version
```

CHAPTER

THREE

GETTING STARTED WITH V-IPU

After installing the `vipu` tool, the next step is to establish connectivity with the V-IPU controller. This assumes that the system administrator has created a user account for you and allocated some IPUs to that user ID.

You will need the following information from the V-IPU administrator:

- V-IPU controller host name or IP address
- V-IPU controller port number
- User ID and API access key

Run the following command to make the server report its version:

```
$ vipu --server-version --api-host 10.1.2.3 --api-port 9090 --api-user-id alice --api-access-key 685XK4uCzN
```

This example assumes a V-IPU controller running on host 10.1.2.3 with port 9090. You should use the details provided by your system administrator.

To avoid having to add options to each command, you can specify the server details in environment variables:

```
$ export VIPU_CLI_API_HOST=10.1.2.3
$ export VIPU_CLI_API_PORT=9090
$ export VIPU_CLI_API_USER_ID=alice
$ export VIPU_CLI_API_ACCESS_KEY=685XK4uCzN
$ vipu --server-version
```

Alternatively, you can add them to a configuration file:

```
$ cat ~/.vipu.hcl
api-host=10.1.2.3
api-port=9090
api-user-id=alice
api-access-key=685XK4uCzN

$ vipu --server-version
```

The next step is to allocate some IPUs to run your software on.

3.1 Creating a partition

This section explains how to create a usable partition on the IPU system. A partition defines a set of IPUs used for running end-user applications.

The simplest way to get started is to create a “reconfigurable” partition. This makes a set of IPUs available to users in a flexible way as a number of single-IPU device IDs and a set of multi-IPU device IDs if the partition is larger than a single IPU.

You can do this with a command such as the following:

```
vipu create partition pt --size 4 --reconfigurable
```

This allocates four IPUs to a partition called “pt”.

You can see the IPUs that are now available using the `gc-info` command that was installed with the SDK:

```
$ gc-info -l
-+- Id: [0], type:    [Fabric], PCI Domain: [3]
-+- Id: [1], type:    [Fabric], PCI Domain: [2]
-+- Id: [2], type:    [Fabric], PCI Domain: [1]
-+- Id: [3], type:    [Fabric], PCI Domain: [0]
-+- Id: [4], type: [Multi IPU]
 |--- PCIe Id: [0], DNC Id: [0], PCI Domain: [3]
 |--- PCIe Id: [1], DNC Id: [1], PCI Domain: [2]
-+- Id: [5], type: [Multi IPU]
 |--- PCIe Id: [2], DNC Id: [2], PCI Domain: [1]
 |--- PCIe Id: [3], DNC Id: [3], PCI Domain: [0]
-+- Id: [6], type: [Multi IPU]
 |--- PCIe Id: [0], DNC Id: [0], PCI Domain: [3]
 |--- PCIe Id: [1], DNC Id: [1], PCI Domain: [2]
 |--- PCIe Id: [2], DNC Id: [2], PCI Domain: [1]
 |--- PCIe Id: [3], DNC Id: [3], PCI Domain: [0]
```

Here, the four individual IPUs have the IDs 0 to 3.

The multi-IPU devices are listed below that. A multi-IPU device always represents a number of IPUs which is a power of two. Here there are three multi-IPU devices:

- ID 4 contains IPUs 0 and 1
- ID 5 contains IPUs 2 and 3
- ID 6 contains all four IPUs

Different Poplar programs can make use of these devices concurrently. A device ID that is attached to by a Poplar program is removed from the list of available device IDs while that Poplar program is running.

When you create a partition, a file is created in the directory `$HOME/.ipuof.conf.d/`. This file contains information needed by Poplar to connect to the IPUs. Note that there should only be one file in the directory, so you should delete the partition (with the `vipu remove partition partition-name` command) before creating another one.

In a fully deployed system, you may want to define partitions containing sets of IPUs configured in specific ways.

See the V-IPU User Guide for full details of the V-IPU command-line software, allocating IPUs for your program, and running code on the IPU-POD.

3.2 Running a program on the IPU-POD

You can now run one of the example programs from the SDK. The program `adder_ipu.cpp` builds a simple graph to add two vectors together and return the result.

1. Make a copy of the `poplar/examples/adder` directory in your working directory
2. Compile the program:

   ```
   $ make
   ```

3. Run the program.

   It will produce the following output:

   ```
   $ ./adder_ipu
   Creating graph
   Building engine
   v1 sum = 10
   v2 sum = 65
   ```

This simple example just runs on one IPU. Further examples and tutorials can be found in the Graphcore GitHub repositories examples and tutorials, such as a simple pipelining example using TensorFlow.

See Section 4, Next steps for more information.

3.3 Hardware and software monitoring

The IPU hardware, and any programs running on it, can be monitored and analysed in various ways.

3.3.1 Command line tools

The IPU driver software includes a number of software tools that provide information on the current status of the connected hardware. These include:

- `gc-info`: determine which IPU devices are present in the system.
- `gc-monitor`: passively monitor IPU activity and telemetry such as power draw, temperature and clock speed.
- `gc-reset`: reset one or more IPU devices.

These are in the directory `poplar-os-version/bin` under the Poplar SDK directory (where `os` is the host operating system, and `version` is the current software version number).

The use of these tools is described in the IPU Command Line Tools document.

3.3.2 Trace output

When running a program, you have the option to output a trace from the running code which allows you to see the phases that the graph compiler uses when preparing and compiling the graph processes to execute on the IPU. Going back to the simple adder example in Section 3.2, Running a program on the IPU-POD, try running:

```
$ POPLAR_LOG_LEVEL=DEBUG ./adder_ipu
```

You will see each stage of the process listed before the program output appears. The logging options are documented in the Poplar & PopLibs User Guide.

3.3.3 Execution profiling

The PopVision Graph Analyser tool is available on the Graphcore software download portal.

This provides a graphical view of the graph execution trace, showing memory use, tile mapping and other vital information to help you optimise your application. For more information see the PopVision Graph Analyser User Guide.

CHAPTER

FOUR

NEXT STEPS

Full documentation for the Poplar software is available on the Graphcore documentation site: https://docs.graphcore.ai/. More information can be found on the Graphcore developer pages: https://www.graphcore.ai/developer.

For an overview of the Poplar SDK and development tools, see the SDK Overview.

The IPU Programmer's Guide provides an introduction to the IPU architecture, programming model and tools available.

If you are interested in running TensorFlow on the IPU, there are user guides and API references for the IPU implementation of TensorFlow 1 and TensorFlow 2.

You can also run PyTorch on the IPU.

Graphcore also has GitHub repositories with further examples:

- https://github.com/graphcore/examples:
  - TensorFlow, PyTorch and PopART versions of commonly used machine learning models for training and inference, including CNNs such as ResNet, ResNeXt and EfficientNet
  - Training data for the above models
- https://github.com/graphcore/tutorials:
  - Tutorials
  - Examples of using Poplar and IPU features
  - Examples of simple models
  - Source code from videos, blogs and other documents
  - Benchmarks for performance testing of layer types on your IPU system

You can use the tags "ipu", "poplar" and "popart" when asking questions or looking for answers on StackOverflow.

Support is available from the Graphcore customer engineering team via the Graphcore support portal.

CHAPTER

FIVE

TRADEMARKS & COPYRIGHT