THE GRAPHCORE SOFTWARE STACK: BUILT TO SCALE

INTRODUCTION

Software for new processor designs is critical to enabling application deployment and optimizing performance. UK-based startup Graphcore, a provider of silicon for application acceleration, places significant emphasis on software, dedicating roughly half its engineering staff to the challenge. Graphcore's Intelligence Processing Unit (IPU) utilizes the expression of an algorithm as a directed graph, and the company's Poplar software stack translates models and algorithms into those graphs for execution. The software simplifies adoption of the chip for AI and parallel computing, making it vital to the company's success. This paper explores the benefits provided by the company's software and discusses how these capabilities could speed development and deployment of applications that run on Graphcore IPUs.

A BRIEF OVERVIEW OF THE INTELLIGENCE PROCESSING UNIT

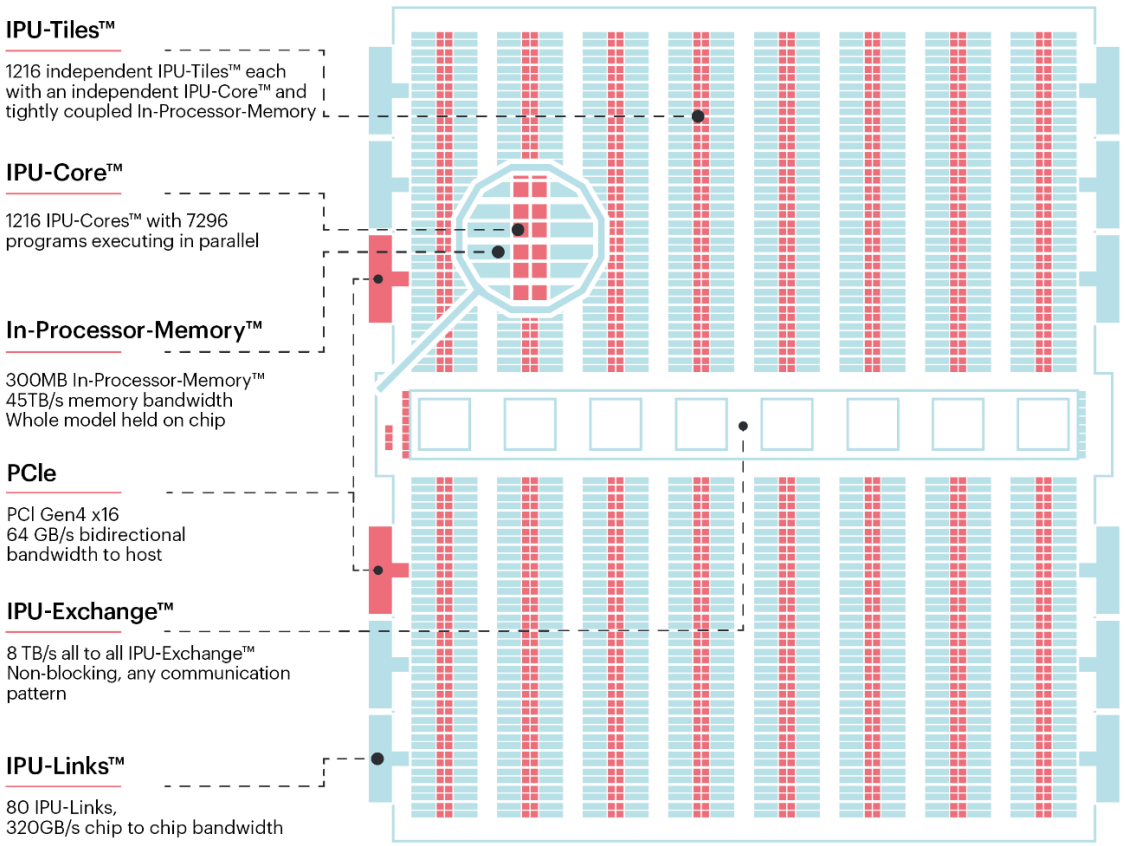
Before we dive into the software, a rudimentary understanding of the underlying hardware will be helpful. Graphcore's Intelligence Processing Unit (IPU) is fundamentally different from today's CPU, GPU, and other AI processors. The IPU and Poplar software combination is not strictly a machine-learning (ML) tool set. The IPU is a flexible, scalable, fine-grained parallel processor designed to deliver high performance for a wide range of computationally intensive algorithms. Together, the IPU and Poplar form a graph-programming platform that enables applications in finance, High Performance Computing (HPXC), robotics, and data science and supports machine-intelligence workloads.

The IPU design goal was to solve problems beyond the capabilities of current acceleration architectures found in most ASICs and GPUs. Typically, those chips are optimized for dense linear-algebra workloads which may be implemented in some convolutional neural networks (CNNs) and are less well-suited for applications with more varied computation, communication, or data access patterns.

The IPU is composed of 1,216 interconnected processing tiles. Each tile has its own core and local on-die SRAM memory to enable the model and data to reside on the IPU, thereby greatly improving memory bandwidth and latency. These tiles are

interconnected through an 8 TB/s on-die fabric (the “IPU-Exchange”), which also connects through “IPU-Links” running at 320 GB/s to create a chip-to-chip fabric.

FIGURE 1: THE GRAPHCORE INTELLIGENT PROCESSOR



The Graphcore Intelligent Processor has 1,216 "tiles" where the computation cores and memory all reside. These tiles are all connected by a high-speed, low-latency “Exchange” fabric, which also extends off the chip to enable multi-chip parallelism to thousands of chips.

Source: Graphcore

Essentially, the entire system (often composed of many IPUs) executes in two synchronous phases: computation and communication. Applications that target the IPU are expressed as computational graphs. Computations are performed at the vertices of the graph, and the results are communicated to adjacent vertices according to the edges interconnecting the graph. The communication phase is implemented as a Bulk Synchronous Parallel (BSP) operation, which efficiently transfers data from each tile’s on-die SRAM memory to connected tiles’ memory. In addition to computation instructions, each IPU core features a dedicated tile-level instruction set for communication phases of the BSP model.

The integrated exchange-communication fabric is designed to support BSP for both data and model parallelism — enabled by the graph compiler — potentially scaling to thousands of nodes. An important distinction for the IPU architecture, according to Graphcore, is the ability to efficiently process sparse data and graphs, which improves performance while reducing the total memory requirements.

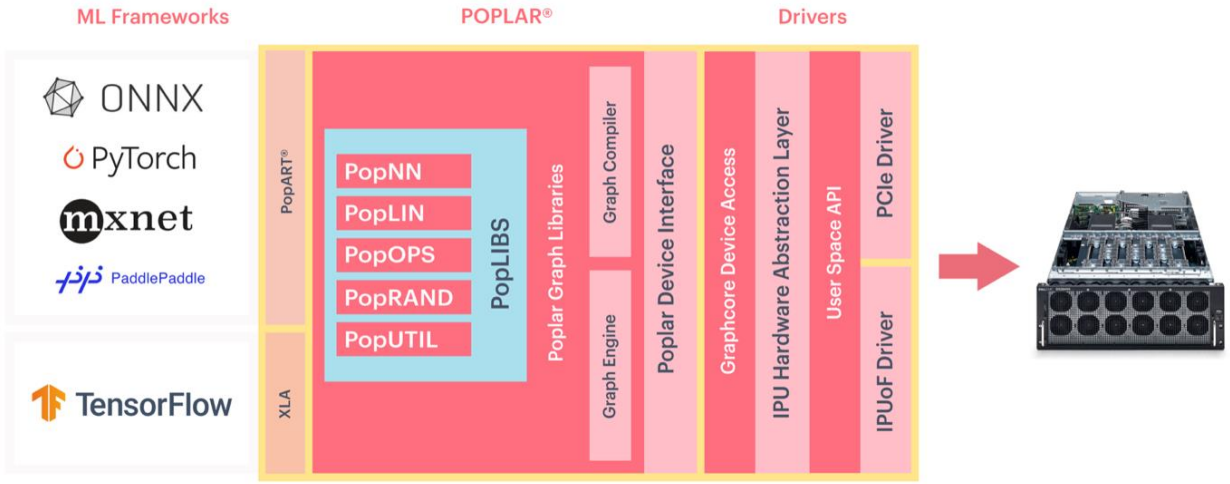
Graphcore's Poplar SDK addresses the programming challenges this unique architecture presents by automating the compilation and optimization workflow, therefore exploiting the fine-grained parallelism of a fabric of IPUs without requiring hand-tuned instruction-level programming. Direct hardware access is also supported.

THE GRAPHCORE SOFTWARE PLATFORM

OVERVIEW

Graphcore is addressing two challenges in software development for its chip: 1) make it easy to optimize and run existing ML software such as deep neural networks, or DNNs, expressed in high-level frameworks, and 2) enable research and development of entirely new fine-grained parallel workloads to run on an IPU infrastructure. The latter ability is central to the company's strategy and could enable Graphcore to address a much larger set of market segments.

FIGURE 2: THE GRAPHCORE SOFTWARE PLATFORM



The Graphcore software platform takes models from many popular AI frameworks and optimizes the code for execution on IPUs. It also supports building custom vertices, or applets, of optimized codes for developing new neural networks and other algorithms that can be expressed as graphs.

Source: Graphcore

As Figure 2 shows, Poplar consists of graph and element compilers, optimized libraries, and a graph engine for run-time management and scheduling. ML frameworks feed the Poplar stack through the Poplar Advanced Run Time (PopART) interface for the Open Neural Network eXchange (ONNX) as well as the Accelerated Linear Algebra (XLA) compiler for TensorFlow-based models. Graphcore says direct support will be provided for PyTorch by the end of 2020.

Let's look at how Graphcore's software platform facilitates application development for the IPU. Specifically, we will examine these major components and consider their potential to accelerate performance and adoption: frameworks, compilers, run-time support, and libraries.

Porting and Developing Models Using Open Frameworks

To evaluate a new AI accelerator, a deep-learning scientist would first port, train, and test existing DNNs using standard frameworks such as TensorFlow or PyTorch, and then compare the results (training time and accuracy) with known platforms. The software team at Graphcore has made it simple to train and run these neural nets on the IPU as well as to prototype new solutions. Developers can augment standard network layers with customized layer types and new library functions, which they can then include in open source frameworks. This capability can augment or enhance existing layers as well.

For new applications that are outside the scope of popular DNN frameworks, Graphcore has built a custom framework for the IPU. The Graph Framework is not intended to compete with existing frameworks. Conversely, it is available to enable completely new parallel workloads to run on their graph architecture.

Graphcore intends to release its software to open source over the coming year, and the IPU SDK downloads already include Poplar library source code. This is a critical direction for Graphcore, as an open-source strategy should help broaden active industry participation in research and development on the IPU platform, enabling the community to develop and upstream new libraries and layers.

Optimizing Execution: Poplar Compilers, Libraries, and Graph Engine

The Graph and Graph Element Compilers

The development process begins with the Graph Compiler, which builds the model for deployment and in turn pulls in Graph Elements (kernels or "codelets") that run on the

IPU. There are 2 phases here, compiling computation (graph vertices) and generating the code to implement BSP communications (the edges for graph).

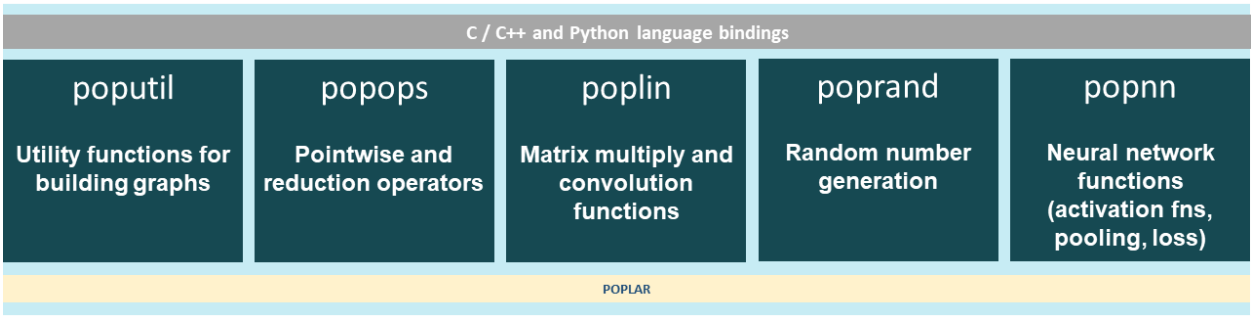
The Graph Compiler, which lies at the heart of run-time deployment and optimization, has been under development for over 5 years. It optimizes code reuse, minimizes data movement, and exploits data locality. This approach helps the IPU perform well in sparse models, which tax other architectures that are more suited to large, contiguous memory-access patterns. Graphcore designed the compiler to simplify programming the IPU, especially in cases where the work is deployed across many IPUs. Importantly, the compiler alleviates the burden on developers to manage data or model parallelism. Instead of programming to, say, 16 distinct ASICs or GPUs in a server, the Poplar Graph Compiler targets a single "Multi-IPU," enabling developers to focus on their data and algorithms. This is a distinct advantage over most other accelerators.

The Poplar Element Compiler acts as a back end, compiling code computation into elements to run at the vertices. So, the workflow looks like:

TensorFlow front end -> XLA -> Poplar Graph Compiler -> Poplar Element Compiler

The IPU has a full and flexible instruction set, allowing tiles to essentially run any code or algorithm, and Graph Elements can be written in C/C++ using the LLVM-based compiler or directly in IPU assembly. All the Poplar Libraries are built using this compilation tool set.

FIGURE 3: GRAPHCORE'S POPLAR LIBRARIES



Graphcore has built over 50 primitives for commonly required functions and operators. Users have the ability to add new libraries to support new workloads as they are developed and contribute them to the open source community.

Source: Graphcore

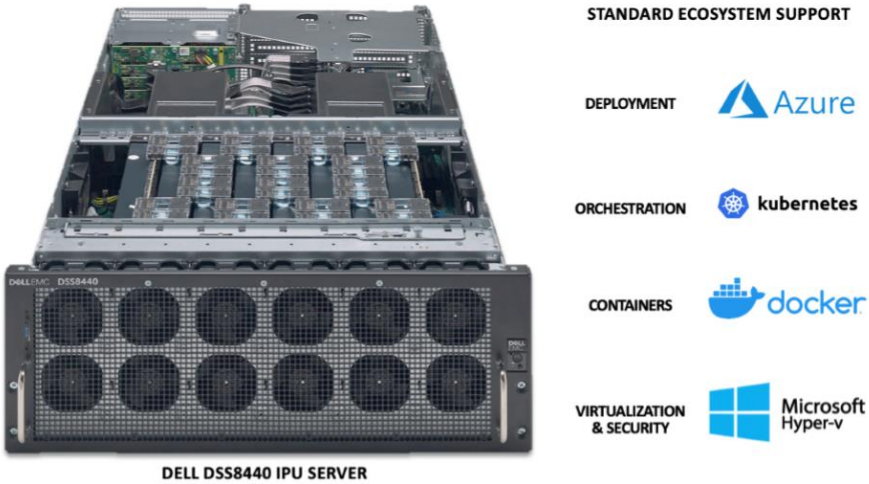
Poplar Libraries

The Poplar Libraries contain more than 50 highly optimized primitives and building blocks targeting common operations such as linear algebra, common neural-network functions, and other operations used in machine-intelligence models. Each function works by combining building blocks from an extensive library of more than 750 high-performance compute elements, or codelets. The Poplar Libraries consume a high-level graph description from a user application and build the massively parallel computational graph required to target the IPU. This process includes work and data partitioning to target the IPU's distributed processor and memory architecture. The Poplar Libraries build a custom layout for every operation that is highly tuned for execution on the IPU. Libraries also offer the possibility of publishing and running completely new kernels and DNN layers for new applications that can run on the IPU architecture.

The Graph Engine

The Poplar Graph Engine provides run-time support for the IPU. This support includes connecting software on the host CPU with software running on the IPU; managing data movement; and managing the IPU device itself for I/O, application loading, debugging, and profiling. Optimizing data flow from the CPU is critical to achieving maximum throughput, and the Graph Engine orchestrates the data I/O pipelines to enable continuous execution. The Graph Engine is supported by custom hardware on the IPU that facilitates debugging and profiling.

FIGURE 4: GRAPHCORE ECOSYSTEM SUPPORT



Graphcore includes support for deployment capabilities rarely found in startups.

Source: Graphcore

DEPLOYMENT: INDUSTRY STANDARD TOOLS AND PLATFORMS

Graphcore software extends to the efficient deployment of production workloads, both in the public cloud and in on-premises infrastructures. Specifically, Dell Technologies sells and supports the dual IPU Graphcore C2 PCIe cards for servers, while Microsoft Azure and Cirrascale support IPU cloud instances (in preview at Azure). For management and orchestration, Graphcore supports the virtualization, security, and orchestration tools that have become standard across the industry, including support for Kubernetes, Docker, and Microsoft Hyper-V. Graphcore is the only startup we know that has extended its offerings to include such a large suite of deployment software and infrastructure.

CONCLUSIONS: GRAPHCORE IS BUILT FOR PRODUCTION, AT SIGNIFICANT SCALE

When training neural networks, developers require fast, scalable accelerators that can handle the massive computational loads required by larger models such as natural language processing and conversational interfaces. More broadly, computationally intensive applications are emerging which require a high-speed parallel processor that goes beyond the matrix-multiplication operations common to neural networks. Graphcore appears to have developed just such a general-purpose, high-performance hardware and software platform to meet these needs.

Graphcore's software stack stands out in several areas:

1. The Poplar Graph Compiler enables "Multi-IPU" parallelism while implementing efficient memory use and data movement.
2. The Graph Framework enables new workloads, especially new algorithms (graphs) that emerge from domains outside the realm of traditional ML frameworks.
3. The full support and optimization of open-source ML frameworks can be extended by the user with custom "applets" that can implement new layers and kernels.
4. Graphcore's open-source strategy should help the company extend its reach across a wide community of researchers in multiple disciplines.

Graphcore has also taken the next step in management software, providing containerization, orchestration, security, and virtualization on which data centers have become reliant and comfortable. Combined with the company's hardware design and software stack, these steps will ease adoption as more applications are deployed on the Graphcore platform.

One of the challenges facing Graphcore is being able to compare its platform to alternative products. This is a natural consequence of targeting new workloads that are not yet widely adopted or for which other platforms are not well tuned. While Graphcore claims that the IPU can run perhaps 2-4 times faster on “simple” models such as Resnet, Graphcore has published results on their website that show an order-of-magnitude better performance in emerging workloads.[1] And that is where the magic begins.

[1] https://cdn2.hubspot.net/hubfs/729091/assets/pdf/Benchmarks_slides_May2020-comp.pdf

IMPORTANT INFORMATION ABOUT THIS PAPER

CONTRIBUTOR
Karl Freund, Senior Analyst at Moor Insights & Strategy

PUBLISHER
Patrick Moorhead, Founder, President, & Principal Analyst at Moor Insights & Strategy

INQUIRIES
Contact us if you would like to discuss this report, and Moor Insights & Strategy will respond promptly.

CITATIONS
This paper can be cited by accredited press and analysts but must be cited in-context, displaying author's name, author's title, and "Moor Insights & Strategy". Non-press and non-analysts must receive prior written permission by Moor Insights & Strategy for any citations.

LICENSING

DISCLOSURES
This paper was commissioned by Graphcore. Moor Insights & Strategy provides research, analysis, advising, and consulting to many high-tech companies mentioned in this paper. No employees at the firm hold any equity positions with any companies cited in this document.

DISCLAIMER
The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. Moor Insights & Strategy disclaims all warranties as to the accuracy, completeness, or adequacy of such information and shall have no liability for errors, omissions, or inadequacies in such information. This document consists of the opinions of Moor Insights & Strategy and should not be construed as statements of fact. The opinions expressed herein are subject to change without notice.

Moor Insights & Strategy provides forecasts and forward-looking statements as directional indicators and not as precise predictions of future events. While our forecasts and forward-looking statements represent our current judgment on what the future holds, they are subject to risks and uncertainties that could cause actual results to differ materially. You are cautioned not to place undue reliance on these forecasts and forward-looking statements, which reflect our opinions only as of the date of publication for this document. Please keep in mind that we are not obligating ourselves to revise or publicly release the results of any revision to these forecasts and forward-looking statements in light of new information or future events.