Research

April 2020

oneAPI: Software Abstraction for a Heterogeneous Computing World

A J.Gold Associates Research Report

"This whitepaper will discuss what is needed for enterprises to move towards a new cross-architecture model, how it will benefit organizations in cost and time savings, and oneAPI's open approach to achieving this goal..."

J.Gold Associates

Contents

Introduction

Enterprises are becoming much more computing diverse as they attempt to cope with an ever increasing need to digitally transform. Indeed, many enterprises have 500-700 different apps running on their systems, and many of them are legacy and/or not optimized to today's advanced processing capabilities. The need to provide high performance apps that can run on premise, in the cloud, on mobile devices as well as on traditional PC endpoints, and do it while taking advantage of all the latest implementations of hardware-based accelerations is a continuing challenge.

> ***TREND**: Enterprise workloads are becoming more complex and more mission-critical as new technologies like data mining, Artificial Intelligence, high definition video, edge computing, hybrid clouds, etc. grow. The broadly defined "digital transformation" requires that organizations look at implementing a variety of new processes and workloads, often on new systems, to supplement and/or replace legacy solutions. In the next 3-4 years, we expect 65%-75% of corporate workloads to be new implementations built on an array of heterogeneous computing platforms, some of which are not yet available.*
> *J.Gold Associates LLC.*

Creating apps for only one general purpose computing platform will not be viable going forward, especially in situations where performance is critical. What's needed is a way to easily adapt any app to the changes in available accelerated processing solutions (e.g., advanced GPUs, specialized FPGAs, neural network accelerators, unstructured data (especially video) accelerators, etc.). Advanced acceleration capability is becoming broadly available and often provides a better performance-sensitive workload solution than just a standard compute engine. Being able to take advantage of workload-optimized solutions is the best way to maximize efficiency in operations while minimizing cost. But starting from scratch to rewrite/update apps specifically for each new acceleration platform results in a dramatic increase in development time and associated costs. It also limits the ability to rapidly deploy on newer platforms as they become available.

What's needed is an approach to application development that allows developers to program once, tune code as needed for any targeted architecture, and efficiently deploy that tuned code to the most appropriate processor. The approach must be open and supported by multiple vendors. It's an approach that importantly prevents lock-in to a vendor and/or architecture in a compute world that is changing rapidly, is much more specialized than the general purpose computers of the past, and is far more heterogeneous than ever before.

This whitepaper will discuss what is needed for enterprises to move towards a new cross-architecture model, how it will benefit organizations in cost and time savings, and oneAPI's open approach to achieving this goal.

A New Approach to Performance Sensitive Applications

An average corporate app remains in use for 5-7 years, with many still in use at double that timeframe. However, significant improvements in computing platforms occur every 2-3 years at least, with many "accelerators" that could significantly benefit performance-sensitive

workloads going unused or underutilized due to lack of application compatibility. The challenge is that once these app workload solutions are designed and tuned to a designated platform, the amount of effort to port those solutions to new operating platforms is often massive. Indeed, the number one reason most companies don't upgrade their software solutions is the amount of time and effort needed to do a port to take advantage of newer platforms with their specialized capabilities.

For maximum flexibility at minimal cost, software must be designed at an abstraction level above the hardware so that it can easily take advantage of any hardware improvements as they become available. Failure to do so means enterprises will be stuck with sub-optimal software solutions along with expensive single-vendor hardware assets and significant barriers to taking advantage of hardware innovations from across the industry. It's imperative that with the rapid change in technology, solutions must be created that run on the best computing platforms available, and are able to change when new and more appropriate (e.g., task-dedicated, energy-efficient, workload optimized) technologies become available.

There are several examples of approaches to portability that are basically "wrappers" to proprietary programming models, but these approaches will never achieve the same level of performance on alternative architectures (e.g., using AMD's HIP to run CUDA code that was originally optimized to run on Nvidia's GPU architecture). While available as a "quick porting" option, such codes will always achieve sub-optimal performance, and thus are of limited use when maximum performance is required.

What's needed is a way to ensure that the majority of work that goes into designing a solution for a specific workload can be easily reused and/or repurposed while still achieving maximum performance even as newer computing platforms arrive in the market. oneAPI enables such an approach.

What is oneAPI?

oneAPI is a cross-industry, open, standards-based, unified programming model that delivers a common developer experience across multiple architectures such as CPUs, GPUs, FPGAs, and specialized accelerators. The oneAPI specification (see Appendix) extends existing developer programming models to enable a diverse set of hardware through a direct programming language (Data Parallel C++), a set of APIs (domain-specific performance libraries), and a low level hardware interface (Level Zero) to support cross-architecture programming. To promote compatibility and enable developer productivity and innovation, the oneAPI specification builds upon industry standards and provides an open, cross-platform developer stack.

The oneAPI industry initiative encourages collaboration on the oneAPI specification and compatible oneAPI implementations across the ecosystem. There are currently two beta implementations of oneAPI that are already available – The Intel oneAPI product for Intel hardware and Codeplay's implementation for NVIDIA GPUs.

oneAPI represents a move to enable the next generation of application creation based on the need to support a rapidly evolving and heterogeneous computing environment.

Why Now?

We've reached an inflection point that is dramatically changing how compute is deployed. With so many different workloads that need to be accelerated, we often need to mix and match scalar (CPU), vector (GPU), matrix (AI/neural processor), and spatial (FPGA) computing architectures. Indeed, there were inflection points in the past – when we moved from single threaded to multi-threaded, when we moved from single core to multi-core, when we moved from CPU to include GPU, and now, we're moving yet again to include complex Video Processing, Machine Learning, Deep Learning and Artificial Intelligence workloads on hardware accelerators incorporated into mainstream computing systems.

All of these inflection points required a new way of interacting with the specialized hardware, often leading to either proprietary methods of programming (e.g., CUDA for Nvidia GPUs), or industry standards that traded off performance-optimization in a quest to be universal (e.g. OpenCL). But as the rate of technology innovation rapidly accelerates, it's no longer viable to take the time to invent a new programming model for each new architecture. What's needed is a durable abstraction layer that allows software developers to create a single set of application algorithms that can then be deployed on existing components, as well as those yet to be invented. That's the stated goal of the oneAPI initiative.

Why a Cross-Architecture Model is Important

The effort for completely re-architecting an application for each new architecture and subsequently maintaining and improving multiple code bases, generation to generation, can be very high. A considerable amount of work is also needed to ensure behavioral consistency across implementations, including on additional documentation and testing. We estimate that a typical application creator spends 90%-100% of the original amount of effort on an application that needs to be updated and moved to a new computing platform. Even with the increased performance obtained from such a move, the cost in dollars and available resources can be overwhelming and is often the reason why the move doesn't happen. Indeed, sometimes the personnel required simply aren't available, no matter what the cost. A cross-platform approach to writing performance-sensitive applications that offers investment protection against hardware obsolescence and/or future proofing the base level software investment can therefore provide a very significant ROI.

Need for a Common Programming Model

With specialized programmers in extremely short supply, any high portability app creation solution must be based on leveraging widely available personnel. A universal programming language that's based on a commonly understood model (e.g., C++) used by millions of developers, allows programmers to easily migrate their skills.

oneAPI has a heterogeneous programming environment through the implementation of the Data Parallel C++ (DPC++) language, with included extensions for the accelerating components of modern systems. DPC++ is based on C++ and presents a relatively low threshold of new learning for developers who already understand and use it. DPC++ also incorporates SYCL to support heterogeneous computing and includes extensions to simplify data parallel programming.

The shift to a heterogeneous programming methodology will be a continuous journey rather than a single moment and will require updates and changes over the years as new architectures become available. DPC++, being an open sourced development effort, will take advantage of feedback and contributions from the extended community during its evolution.

What's also needed as part of any abstraction layer is a simple way to accommodate new hardware interfaces as novel architectural components become available, along with required libraries, compilers, debuggers and analysis tools to simplify their use. This requires a cross architecture and cross vendor model that is open and extensible, while also preserving much of the front end development work potentially done before these new components and tools were available. The oneAPI initiative takes into account all of these requirements.

The High Cost of Single Use Code - Calculating the Advantage

Deciding whether the consolidated approach to application portability envisioned in this approach makes sense can best be determined by looking at the costs associated with the traditional single use approach and the cross-architecture oneAPI approach across hardware implementations.

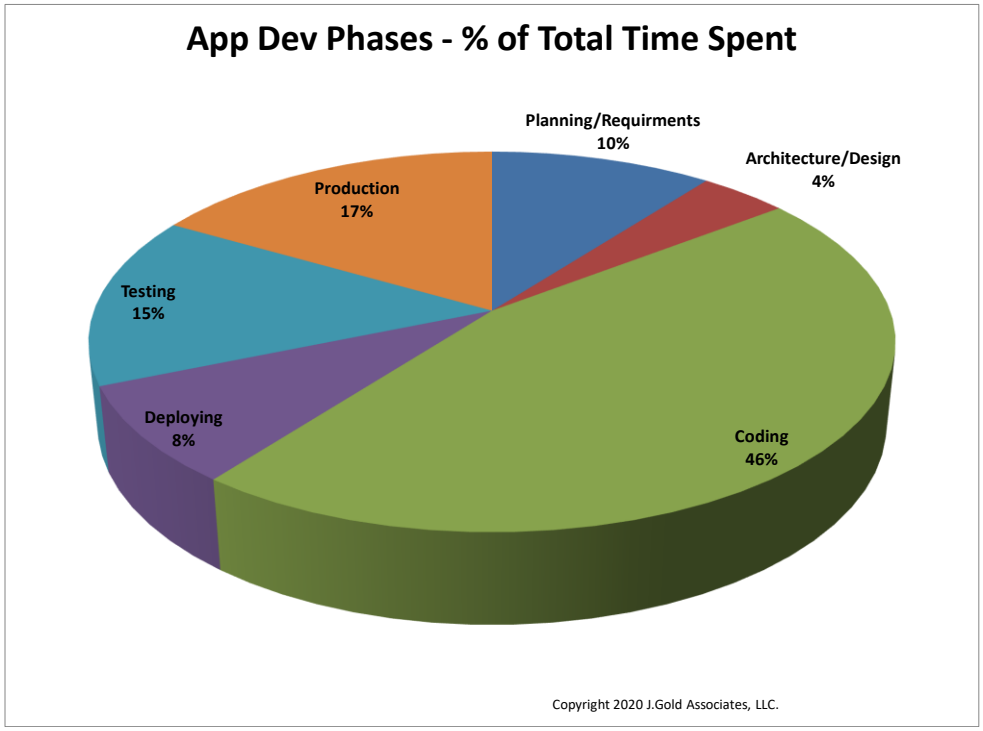
As an example of the costs involved in creating an application, Table 1 provides an overview of a typical project. For this example, we assume the project to be 1 year in duration and have 6 people assigned to its completion, with a yearly burdened salary of $120,000 for each participant.

Table 1: Component Phases of a Typical Enterprise Application Project and Their Costs

Phases of App Development	Months	Cost
Planning/Requirements	1.25	$75,000
Architecture/Design	0.5	$30,000
Coding	5.5	$330,000
Deploying	1	$60,000
Testing	1.75	$105,000
Production	2	$120,000
TOTAL	**12**	**$720,000**

Shown above is the cost for each phase and for the total project. Based on the above calculations, we can determine the percentages of total resources necessary for completion of each phase of the project. The result is shown in Figure 1 below.

Figure 1: Phases of a Typical Application Development Project, with Approximate Percentage of Time Spent in Each Phase



Finally, in evaluating the possibility of benefiting from a more reusable environment that doesn't require a complete rewrite for use of new technology solutions, we'll look at a predicted result of using a more portable approach to moving the given application workload to a new processing engine and/or accelerator platform.

When porting the performance-sensitive apps to a new technology platform such as a newly available accelerator, we estimate the following advantages of a cross-architecture model such as oneAPI:

- *Planning/Requirements* – we expect no significant saving here as the problem definition and algorithms will likely remain constant across all of the potential platform deployments.
- *Architecture/Design* – we expect 25%-75% savings for multi-architecture designs as new technology deployments won't require redesign. For a single application with no porting requirements, there will be no real savings.

- *Coding* – we expect 35%-85% savings as the majority of software created will not have to be reprogrammed when porting to a new solution.
- *Deploying* – we expect a modest 10%-30% savings over deployment of a completely new implementation as learning from earlier deployments will be leveraged.
- *Testing* – as in deploying above, we expect a relatively modest 15%-30% savings, based on learning from a previous deployment test process
- *Production* – we expect a 20%-80% savings from the increase in productivity and enhanced performance available on newer accelerated platforms, although this number can vary greatly depending on the workload and organization. Production effort, which includes bug fixes and maintenance of multiple code bases, is substantially reduced when operating with a single code base.

By taking a midpoint of our estimates detailed above, we can calculate the savings of using a cross-architecture programming model. The first two columns of Table 2 indicates our estimated percentage savings per phase of the project if using a more portable and reusable approach to developing an application, and then moving that application workload to a new processing engine.

Using those estimated savings per phase, we can calculate the total cost savings for each move of the application to a new computing platform or accelerator and the cost advantage that a cross-architecture model offers. We used the midpoint of average savings for each phase to complete the savings calculations provided in Table 2 below.

Table 2: Component Phase, Percent Savings and Cost Savings for Porting an Application when Employing a Reusable Approach

Phases of App Development	% Low	% High	Cost	% Saving	Savings
Planning/Requirements	0%	0%	$75,000	0%	$0
Architecture/Design	25%	75%	$30,000	50%	$15,000
Coding	35%	85%	$330,000	60%	$198,000
Deploying	10%	30%	$60,000	20%	$12,000
Testing	15%	30%	$105,000	22.5%	$23,625
Production	20%	80%	$120,000	50%	$60,000
TOTAL			**$720,000**		**$308,625**

Table 2 indicates that in our example, reusability enabled by a cross-architecture approach that doesn't require a total rewrite of the application to take advantage of new platforms can save $308,625 for each move to a new computing platform or accelerator. And it's likely that over time, there will be multiple moves, so the cost savings will increase accordingly.

But it's not only about cost savings in app creation that's important. Shortening the time to deployment of any improved app can have major benefits to an organization. Table 3 below quantifies this savings in time to app deployment.

Table 3: Component Phases and Time Savings for Porting an Application when Employing a Reusable Approach

Phases of App Development	Months	% Saving	Time Saving
Planning/Requirements	1.25	0%	0.00
Architecture/Design	0.5	50%	0.25
Coding	5.5	60%	3.30
Deploying	1	20%	0.20
Testing	1.75	22.5%	0.39
Production	2	50%	1.00
TOTAL	**12**		**5.14**

As shown above, using a high portability approach can save 5.14 months in time to deployment of an app. It's hard to generalize the cost savings associated with this, but we can imagine increased productivity of a workforce, or being able to process more calculations over a 5 month period that can substantially add to the bottom line of any organization.

While we believe these numbers to be a good indication of a typical performance-critical project and related costs and advantages, each project will be unique and some may be more advantageous, particularly if the application workload will be in use for a long period and multiple iterations will be moved to new computing platforms over the years.

Recommendations

We strongly recommend organizations take the following actions:

- Computing systems will require updates on a regular and decreasing-time basis as new purpose-driven acceleration becomes available. Enterprises and application developers must look at portable, cross-architecture code methodologies as a way to efficiently move apps to new technology accelerated computing platforms in order to maximize user productivity and performance, while also minimizing scarce developer resource requirements.
- Irrespective of whether performance-sensitive apps are run natively, in a fully cloud-based, or a hybrid environment, it's very likely that the ability to utilize advanced accelerators will become highly advantageous. Those enterprises utilizing cloud-based systems should transition to oneAPI even sooner as it's more likely that advanced acceleration will be available earlier in such systems, making a cross-architecture model essential for rapid utilization of new resources.

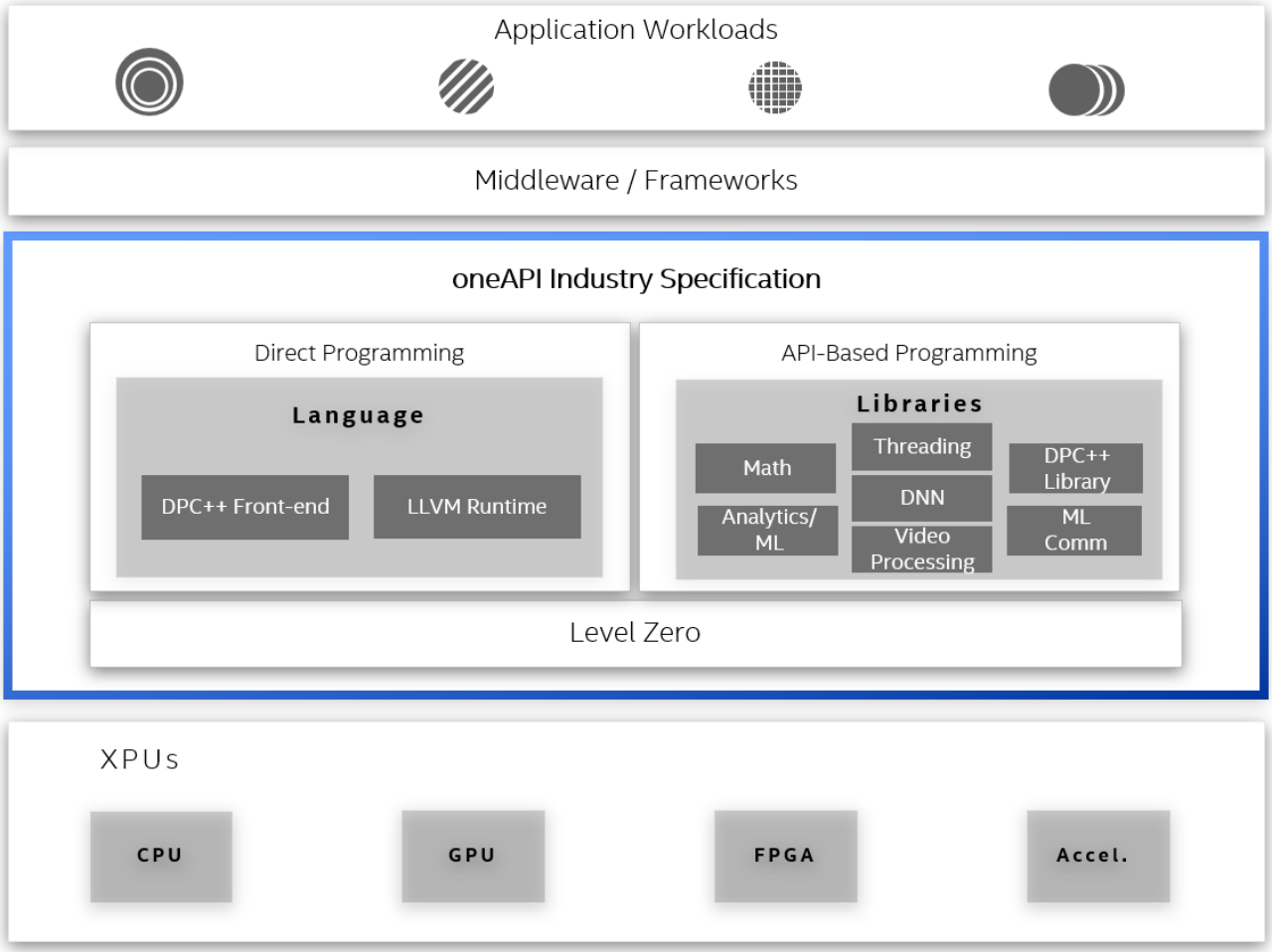
- Adopting architectures requiring proprietary programming models is a major inhibitor to being able to use cross platform technology, both from a performance and cost of operations scenario. Companies should avoid locking into specific architectures and vendors in order to realize significant savings in both hardware and software costs.
- Those organizations that fail to adopt a more portable model for deploying workloads will fall behind their competition within 1-3 years as the pace of technology change is great and a single port model does not provide any computing flexibility. Enterprises must therefore take maximum advantage of open, cross-architecture models of operation to avoid putting themselves at a major competitive disadvantage in the long term.

Conclusions

A wide array of accelerated processors are coming to market at an ever faster pace and more application workloads can take advantage of their capabilities, but only if the applications are built with a model that assures maximum compatibility and flexibility. A cross-architecture model like oneAPI goes a long way to assuring developers of performance-sensitive applications that portability across new and emerging acceleration platforms can take place without having to totally rewrite the app. Although not a “write once and run anywhere” model, many of which ultimately proved to have severe performance trade-offs, this new paradigm does provide an open community approach that can go a long way to helping organizations keep their software assets running at maximum effectiveness for a long time into the future.

Appendix

oneAPI Industry Specification:



About J.Gold Associates

J.Gold Associates provides insightful, meaningful and actionable analysis of trends and opportunities in the computer and technology industries. We offer a broad based knowledge of the technology landscape, and bring that expertise to bear in our work. J.Gold Associates provides strategic consulting, syndicated research and advisory services, and in-context analysis to help its clients make important technology choices and to enable improved product deployment decisions and go to market strategies.

J.Gold Associates, LLC
6 Valentine Road
Northborough, MA 01532 USA
+1 508 393 5294
www.jgoldassociates.com