CORPORATE PARTICIPANTS

**Xiaojun (Shawn) Li**

*Intel – Sales Director, Next Wave OEM & eODM*

**Abhinav Singh**

*Intel – Software Technical Consulting Engineer*

**Ashish Gupta**

*Intel – Business Development Manager*

................................................................................................................................................................................................................................

PRESENTATION

**Shawn Li**

Welcome, everyone to the Intel Network Builders Insights Series. I'm Shawn Li, Sales Director, Next Wave OEM & eODM at Network and Communication Sales Organization at Intel Corporation. I'm your host for today's webinar. Thank you for taking the time to join us today for our webinar titled “Analyze & Optimize FlexRAN, DPDK, and the Other Network Workloads Using Intel oneAPI”.

Before we get started, I want to point out some of the features of the BrightTALK tool that may improve your experience. There is a Questions tab below your viewer. I encourage our live audience to please ask questions at any time. Our presenters will hold answering them until the end of the presentation. Below your viewer screen, you will also find the Attachments tab with additional documents and reference materials which pertain to this presentation. Finally, at the end of the presentation, please take the time to provide feedback using the Rating tab. We value your thoughts and we will use the information to improve our future webinars.

Intel Network Builders Insights Series take place live every month, so please check the channel to see what is coming and access our growing libraries of recorded content.

In addition to the resources you see here, we also offer a comprehensive NFV and 5G training program through Intel Network Builders University. You can find the link to this program in the Attachments tab, as well as the link to the Intel Network Builders Newsletter.

Today, we are pleased to welcome Ashish Gupta and Abhinav Singh from Intel. Ashish Gupta has spent over 25 years at Intel in product management, and business development. He currently helps our networking, storage, and automotive customers discover Intel’s incredible portfolio of performance optimization tools.

Abhinav Singh is a software technical consulting engineer at Intel, where he enables products for customers and the software developers through technical support, trainings, and hands-on assistance in area of code development, debugging, tuning and scaling of software applications.

Welcome, Ashish and Abhinav, and thank you for taking the time to join us today. Ashish, I will hand it over to you to start off. Thank you.

**Ashish Gupta**

Well, thank you, Shawn, and hello, everybody. Thank you very much for taking the time to listen to our webinar. We are here to share with you some information about how to optimize your workloads.

Abhinav and I, what we do is we work with dozens of networking, dozens of storage customers, and our job is to basically help them discover Intel's amazing portfolio of tools, and frankly, my biggest roadblock is to get people to, once you discover it, actually try it. So, our plan for today is we're going to give you a high-level overview of oneAPI. oneAPI is the name of our new tool chain that Intel's providing. As you know, if you're a FlexRAN customer, FlexRAN requires Intel Compiler, it requires a very specific version. It requires 2019.3, and I'm going to share with you how to obtain the specific compiler, because that's a very commonly asked question we get. Abhinav will talk a little bit about IPP. IPP is Intel's Integrated Performance Primitives tool, and this is a library of functions which we find our FlexRAN customers, once they discover, really enjoy, and find useful. The bulk of our webinar today is going to be spent on a VTune overview. So, this is going to be a live VTune overview and demo. Most of this content is fairly technical. So, take notes, pay attention, and then as Shawn said, ask questions. This is your time. If you’ve dialed in live, ask questions, because it's the best way for you to interact.

All right, so let's get started. We've got a product called oneAPI, and this is part of Intel's legacy of tools. I've been with Intel for, gosh, almost 30 years, and we've had a legacy of tools where folks who would buy supercomputers and these massive data centers from us, they knew that they required tools from Intel to really help optimize the performance. We used to call these products under the name of Parallel Studio and Intel System Studio. Well, that's changed. What you see on the screen now is we see a whole host of blue icons here, and that is part of the Intel oneAPI Base Toolkit, the light blue. The dark blue is the oneAPI IoT Toolkit, which is an add-on to the Base, and what's not pictured here is an HPC Toolkit. So, what we did was, we've got well over a thousand developers creating tools for over 20 years, and that buys you a lot of tools, and what we've done is taken the tools and bundled them into packages, which we believe make the most sense for our customers. So, for the light blue, for the Base Toolkit, we've got our new DPC++ Compiler, which replaces Intel Compiler Classic. It includes compatibility tools, Python tools, FPGA tools. We've got a whole host of libraries. One of these libraries alone, IPP, we're going to talk about today. A few years ago, we added up how many man years or human years of effort has gone in, and IPP alone has over a thousand man years of effort that's gone in. So, these are enormous libraries that are available to you, and it's something which I think you should definitely discover.

And then for analysis, debug, our focus today is VTune, but there's also Advisor. Advisor is the vectorization modeling tool which basically shows you where your code is taking advantage of Intel's wide vectors, and then it also has a roofline analysis, which says, if you were to optimize this particular area of source code, you would get this much improvement in performance. So, as a developer, it's easy for me to come in and say, oh, you've got to optimize your code, but where do you even start, and that's where these tools really help.

Now, it ends up being an iterative process, and what I mean by that is, you take VTune, and Abhinav and I, when we talk to developers, developers are like, hey, I wrote the code, I know what's in it, and they're often almost always surprised when they look at VTune and see what their code is actually doing. So, the iterative process I'm mentioning is, you go to VTune and VTune will show you the line of source code that's using CPU, the most CPU, the most memory, the most I/O. It shows you cache misses, things which you wouldn't have thought about, and then you would take something like Intel Compiler and ICC, and if you're already using GCC, what you may not know is that ICC and GCC are completely compatible. In fact, Intel is one of the largest contributors for GCC. So, a trick our customers use is compile a portion of their code using ICC. You don't have to change the whole portion if you don't want to, the whole code, and then you go back to VTune. And now you find functions that are slow, and then you say, well, maybe IPP has a function that I can replace, or maybe MKL has a function that I can replace that are already written. These are basically written by our engineers who sit down with our CP architects, and as you know, every couple of years we expand our instruction set, we've got wider vectors, we've got more features, and these functions are really built to take advantage of all of those different Intel features.

So, that's the oneAPI Base Toolkit. The IoT toolkit includes a very important component for FlexRAN, which is the ICC Compiler, and I mentioned earlier, you need a specific version of ICC Classic. That's why you need the Intel oneAPI Base and IoT toolkit. So, the fourth tools you've seen highlighted over here are the ones we, in our experience, find most of our FlexRAN customers discover and then use. Certainly, there's others as well. So, those tools are ICC Classic. This is required. There's IPP, MKL, and then VTune.

So, let's go to the next slide. So, I mentioned FlexRAN requires ICC. The current version for FlexRAN 21.07 requires not just ICC, the Intel C++ Classic, but requires a specific version of it, and that version is 2019.3. Now, this is-- I think we've probably shipped seven or eight versions since then. So, although you can download oneAPI online at no cost, the no-cost version does not include access to older versions, and it does not include access to confidential support. So, in order to get this particular version, your best bet is to ask the Intel account manager. So, hopefully, all of you already know who your Intel account manager is. Alternatively, if you don't know your account manager, you can send an email to intel.software.sales@intel.com. Chances are whoever responds to this will probably ask you for your account manager, but regardless, we will try to help you out, but this is your best way to get a commercial license for oneAPI Base and IoT Toolkit, which is required for you to access this older version.

Now, once you get that serial number, you register on Registration Center and then intel.com, and then what you'll see is a box that says, “Pick your download”. So, you download a product called Intel System Studio Ultimate. This is a legacy product, and you actually need version 2019 update three with the offline installer, and then on the right-hand side, definitely not going to go through this right now, but this is basically for your reference. So, this is how you would go about installing the package, and it gives you two options here. If you want to install the entire 5-gig package, or if you want to do a silent install for just Intel Compiler. FlexRAN, we're now in November, at the end of this month, we expect to ship FlexRAN 21.11, which will require either the latest version of ICX, which is 2021.4, or it will also compile with ICC 2019.3. So, the good news is at the end of this month, you will have an option of using a more modern compiler, the ISC 2021.4, which is also C line based.

Please have any questions and I'm going to hand this over to Abhinav.

**Abhinav Singh**

Thank you, Ashish. So, let me first introduce you to IPP. So, IPP is a solution which has been used in the industry for about two decades now. Intel IPP provides developers with ready-to-use processor optimized function, which can accelerate image processing, signal processing, data compression, and cryptography domains. All these APIs are hand optimized for every Intel platform. So, a question comes in, why do we even need the performance library? So, every time a new platform comes in, we basically introduce new instructions, we enhance our features, enhance our codes, and code becomes difficult to maintain, and it's a complex process to hand optimize the software for every release or with every new generation of the hardware. So, Intel’s solution to this problem is to basically introduce Intel Performance Libraries, and we have basically shortlisted all the most common domains which are being used by our customers, and we have hand optimized and provided functions to our customers to just go ahead and use it, and get the performance out of it.

So, IPP is a solution which basically supports four different domains, as I talked about, image, signal, data compression, and cryptography. Image processing function itself supports more than 2,000 basic primitive-level APIs for resizing, rotation, remapping of an image, maintaining the quality of the image in case of resizing, if you need. We have image statistics function, image arithmetic functions, and logical operations, and when I say the word primitive, it basically means you are defining the data types, which are based on IPP, and you are also assigning memory which are based on IPP. So, it supports from your initialization to freeing of your memory, along with all the functions supported within IPP. Intel IPP image processing is also integrated with an OpenCV, so if you're using OpenCV in any form, IPP is available without any source changes. So, you just need to go, enable a flag saying with IPP, and just recompile your binaries against IPP.

Then we have signal processing, which is basically heavily used within different telecommunication customers, and it basically accelerates operations like DFTs, FFTs, filtering, CRCs, convolution, and if you look at the box here, where it says telecommunication and vector math, that's where signal processing is accelerating FlexRAN. So, we basically accelerate functions, which does a heavy compute on CRCs, DFTs, FFTs, and even vector math functions for comparing vectors or moving packs of vectors. At the very end, we have two different domains, supporting data and that is data compression and cryptography. So, the whole idea of introducing this domain into IPP and accelerating these domains is to accelerate open-source libraries. So, basically, if you're talking about data compression, IPP supports pretty much everything in the LZ family, including ZLIB, LZO, BZIP, LZ4, ZFP and basically, the whole idea is that the customer does not need to do any source changes. So, we provide patches on an open-source-specific version of data compression, and you can just take our patches, apply them on top of your open source libraries, and recompile it against IPP libraries, and you're set to go.

For cryptography, it's completely open source. So, Intel IPP Crypto is available on GitHub. It supports pretty much everything which is supported on crypto, including public key cryptography, RSA, HMAC, CMAC, data integrity, and authentication hash, which basically includes SHA, MD5, SM3, symmetric/asymmetric algorithms. It also supports a multi-buffer library for RSA, ECDSA, SM3, which are basically highly optimized for the new instruction sets which were introduced with the Intel 3rd Generation Xeon family.

Ashish, can we move to the next slide? All right, so let's have a look where basically IPP signal processing accelerates FlexRAN or any 5G function. These functions are available individually for use as well, and for your easiness, and for the developer to do less work, we have already identified all the functions which can accelerate FlexRAN SDK and have already integrated within FlexRAN. So, you just need to enable IPP from FlexRAN. So, basically, we have the copy function, which copies the contents of one vector into another vector.

We have CRCs, which basically computes up to 32-bit of CRC checksum from the source data buffer, and IPP supports all the CRCs which meet the 5G specification. These CRCs are held in data redundancy check for transport block, and within the data channel, as well as the control channels.

We have FFT function, which performs basic fast Fourier transform and discrete fast Fourier transform, which is DFT on signal samples. They can speed up the up-sampling and the down-sampling on a physical layer.

We have another function from vector math, which is MaxIndx, which returns the maximum value of the index, and it's being very heavily used when you're doing a physical random-access channel, while transmitting the data for peak detection.

We have sorting functions, which sorts all the elements of a vector using radix algorithms. In general, it helps in beamforming, which is a particular processing technique for signals that allow for a directional transmission or reception to make 5G connections more focused towards receiving devices, and that's the whole idea of making 5G faster. Can we move to the next slide?

So, this is a very new feature which we integrated into Intel IPP and will be available with IPP in 2020-- It's already available with IPP in 2021.4 release. So, if you look on the left-hand side of the box, there is a single float FFT and DFT call within 5G and that is supported within the FlexRAN SDK. You typically do steps which includes five steps in getting the size of the FFT. So, basically, get the compute size supported with the IPP API, you allocate your memory based on your needs, with the size of the specification structure, again an IPP API, you initialize the specific agent structure, which contains data, which you want to do the operation on. On the fourth step, you perform the specific length complexity and DFT, either way of the operation what you want to do, and once you're done, once you receive a status message from IPP, where it's successful, then you can free up the memory. So, these are five steps for a 5G single call of FFT and DFT which was float32.

And with FP16, which is half precision binary floating pointer, which is copying 16 bits or two bytes of data, we have introduced a single API to do all your operations. So, you just need to specify source destination and the length to a direct DFT or an FFT call with IPP. So, all your compute for these five steps will just go on into one step and it will do the work for you automatically. FP16 is a new feature which will be available with the next generation of Xeon. It has a lower precision, so the speed of communication can be improved with less memory and less time. So, it cuts off a very complex overhead from your application. Can we move to the next slide?

So, moving on. Basically, we want to introduce you to VTune Profiler, which is our profiler tool, and Ashish talked about it briefly, that how it helps the customer to optimize not only in the networking workload, but all over the industry. So, we have customers actually who are using VTune from embedded systems where even a microsecond is important for them to come. So, it scales up from a microsecond analysis to over hours and hours of analyses. So, it's a tool that supports CPU, GPU, FPGA, mostly which are supported by Intel. It supports pretty much every high-level language including C, C++, DPC++, Java, Python. It's supported on every operating system, which includes Linux, Windows, FreeBSD, Android, Yocto. You can analyze your data through a GUI, which is provided by region. You can collect your data using command line interface, and you can always hook up your command line within scripts or… You can always hook up your command line within scripts and collect data periodically if you want. Can we move to the next slide?

So, I'm going to give you a live demo of VTune, which will be walking through some data sets, and looking at different configurations of VTune within the GUI itself. So, if you're looking at my screen, I would recommend to please switch it to a full screen so you can look at the data collection or the different options available within VTune very clearly.

So, this is a standalone installation of VTune, which is on my Windows machine, and this is the GUI interface of VTune, with the welcome screen of VTune. Basically, what you will see is your different configuration types, your recent projects. On the left-hand side, you will see the project navigation. So, basically, if you have collected data sets on different machines, you can always sort it out and you can always switch it back and forth from different machines.

On the very bottom, you will see the small feature content, which is explaining to you how to configure VTune for different configurations, if you have any specific configuration on virtual machines, if you want to use any IDE, which VTune supports. Also, you will see different types of analysis, which gives you a very good overview of how to solve a specific problem. So, in general, we call them cookbooks within VTune, so we identify the common problems faced by our customers, and then we write an article about it, which can help you or walk you through the steps and give you very good tips on how you optimize your application.

So, I'll start with configuring a basic analysis and showing your different options which VTune supports in profiling an application or a system. So, once you have your project ready, once you create a new project, you just need to click on this “Play” button, which is “Configure Analysis”. Once you click on this button, there are three different boxes.

On the very first box, which says “Where”, you will see-- which means where do you want to profile your applications. So, your application can be locally available. It might be running on a remote system. And there are different ways of configuring.

Local Host means everything which is sitting on your local host. So, your application, your system, you have installed locally, and you are just profiling everything, which is locally aware.

You can always target an Android device by ADB or by using a USB. You can always target a remote Linux machine, and that’s one of the most popular ways they are used in VTune to target your machine, whether you are sitting on your laptop at your home, your machine is somewhere in a server… your machine or a server is somewhere in a lab around the world, and you can just target your machine sitting from your home.

You have TCP/IP, which helps you communicate through an agent, and you can always communicate over an IP connection. And you can-- at the very end, you can always generate command line for different operating systems.

For remote Linux SSH, you just need to provide the SSH destination with the username, then you need to provide where VTune will copy a collector package. So, a collector package is a package what VTune copies it over to your remote machine, extracts the drivers, extracts the services, which are needed to collect the data from your machine. You can always specify your custom path, which is writable with your username. By default, it will always choose DMP.

The second box is what you want to profile. So, you can either profile your whole system, attach it to a process, or launch an application. When you are attaching it to a process, you can attach it to a process name or a process PID, whichever is available. You can always profile the whole system. If you are running multiple applications, multiple services, and you just want to see how your application, or the system, or the I/O behaves with the internals, you can always profile the full system, and later you can filter it out. You can always launch an application, in case if you know where your application is sitting with additional parameters. And you can always specify to directly profile that application and see how it behaves with your system.

We have a couple of other options here within the box, which helps you manage the collection. So, you can automatically resume and stop collection in case if your application is doing an environment initialization, or a pre-fetching, which you don’t want to be profiled. You can always select this option and say, “Resume Collection”, which will give you a delta, which will pause your collection, and it will only collect you the data for the right time.

You can always stop your collection if your application is running for hours and hours, and if you just want to collect a couple of minutes, or a couple of seconds, and you think that’s enough for you to visualize the performance. You can always stop the collection, and VTune will automatically stop the collection after the time.

A couple of other metrics here is you can always specify the CPU Mask. So, if you have pinned your application to run on a specific core or your network workload to run on a specific core, you can always specify within this mask and VTune will just profile the data for you.

You can always provide a Wrapper Script, which is any wrapper around your application, which needs to be done before the collection.

The third box and the very most important box within VTune is how you want to profile your data. And this is something very important, which gives you an idea or the key metrics to optimize your application or the network workload.

So, we start with performance snapshots. So, that’s an idea of getting a coffee and a quick snapshot on your application performance. So, what you do is you do all the configuration, turn on Performance Snapshot, and just run VTune. It will give you a very good overview of what your application is bound on based on the suggestions, if it’s thread-bound, memory-bound, it will give you suggestions to run different analysis types here, which you see in this box.

We always recommend our users to start from hotspots, which is a high level overview on your application based on software and hardware event-based sampling. And when I say the word “software and hardware”, it means VTune has the capability of collecting data in two different ways. Either you can use the software collector, or you can use the hardware event-based collector.

When you use the software collector, you don’t need the Linux VTune drivers, which are installed for Perf collection. It will just use the system Perf, which is supported on Windows or Linux.

You can always use the hardware event-based sampling, but in that case, you'll be needing the VTune drivers and the SEP drivers which are available with the VTune package for a deeper dive analysis, which will give you more hardware-based sampling events.

The second option is anomaly detections. It’s mostly targeted for anomalies for profiling a critical code on a microsecond level. By default, VTune will profile your application on a millisecond level. You can always configure it to collect it on a microsecond level.

Microarchitecture Exploration is a superset of all these collections, and it gives us hardware event-based sampling for collecting the data. So, you will see every instruction set, which went from the top of your application to the bottom of your application, from the start time to the end of your application, how your instruction set came into the system, when your application or the workload started, and when it retired. So, it will give you information on every instruction, whether it went bad, whether it was memory-bound, whether it was code-bound, or whether it went very well.

You can always do memory access and memory consumption analysis, which will give you a very good overview how you are using your memory, and memory access-related issues, for example NUMA architecture, core, or what is the scaling of your memory with direct memory object and that allocation stacks.

If you want to do a specific threading analysis, we also support threading, which helps you figure out if your application is paralyzed. You can always do checks on times which will help you figure out context, which is very important in the case of FlexRAN SDK, because everything is optimized to run very fast, and that’s what we-- what our focus is to basically, you know, finish the job as soon as possible.

You can always do HPC performance, if you are doing an HPC performance characterization. And basically, it helps you identify the compute intensive application or the bottlenecks within CPU and GPU.

We have I/O analysis, which is an input/output subsystem on CPU processor-based, and it is already integrated within DPDK/SPDK open source libraries, which are publicly available. So, if you are using a DPDK-based library, you don’t need to do any specific prework. Intel VTune is already integrated within DPDK, and how it’s integrated is using Intel VTune instrumentation and trace technology APIs, or as we call it ITT APIs. These are just a set of simple APIs which can be hooked up within any part of your core to collect traces.

So, let’s say, for example, you have millions of lines of codes and you just want to profile a specific part of your network workload, or a specific part of your code, so you can just hook up these APIs between those specific functions, and VTune will collect traces for you. So, if you have a DPDK-based API or an application, you just need to click this option, provide all your applications. If it’s running on a specific core, you provide your specific core. If it’s in process for an application, you're going to attach to a process and just click on this “Play” button to start the profiling, and it will start all the profiling for DPDK.

We’ll go through a sample dataset of DPDK after this-- at the very end of this presentation.

Intel supports various accelerators, including GPUs, FPGAs, and if you are using any of these accelerators, VTune will help you identify the… the offloading and even the CPU or GPU interaction along with FPGAs.

We have platform level analysis as well, which does not include a lot of code coverage, but it gives you a very good overview of how your system is laid out, how your system is being used by your application if you are using every resource of the system, which is available on your application.

And Platform Profiler is the best tool, what I recommend for any platform engineers. So, it’s a tool which can collect data up to hours and hours and it will give you a very good overview of your data. Basically, if you are running a workload for, let’s say six hours, it will give you a whole timeline of six hours, how your CPU was used, how your storage devices were used, how your-- each individual RAMs were used, how your networks were used. If you have different network devices, what was the throughput? And at any point of the time, you can always, you know, zoom into a timeline, and see what exactly happened at a particular time.

At the very bottom, you see these four buttons. Basically, one is to start the collection, start and stop. You can always pause the collection and resume the collection. If you are running an application and you don’t want a part of your application timeline to be profiled, so you can always start and pause. There are two small different buttons. The first is for Binary/Symbol and Search. So, if you know the application source, if you know the release binaries or the debug binaries, you can always specify the path of these binaries here. It can be a network path, it can be a local path depending upon how you want to configure it.

Having said that, I just want to clarify, when you talk about binaries and symbols and sources, VTune does not require your application to be compiled or prechecked in a specific way. So, if you have your release binaries with GCC or ICC, you can always use VTune on top of it. If you have debug binaries, you can always use, which will give you better information. So, you don’t need any special pre-compiles with VTune.

At the very end, whatever you have configured over your GUI is all available through Command Line. Every option which is on the GUI is available through Command Line. So, even if you don’t have GUI installed on your server, you can always use the Command Line to collect data. And once you have collected data, you can copy back the data to your local machine and visualize using the GUI.

OK, having said that, let’s jump into a very basic dataset, which is a hotspots collection with a Matrix Multiplication example.

This hotspots collection was done locally and using Hotspots with user mode sampling and tracing. So, if you look on a very high level, it will give you information on timing. Basically, what was the CPU time, how much was the effective time, spin time, overhead. It will give you the top hotspots, which includes the top five functions which were being used from your application. So, if anything which you feel is taking a lot of time or any function which needs to be optimized can be just looked into here from this table.

It gives you a very good effective CPU utilization histogram, so individually, how every cores were used, the number of cores present on the machine, and it will give you cumulative results, how all the cores were used and how the parallelization was done. At the very end, it will give you collection platform info which will include information where you collected the data, what was the environment, if it was user mode sampling, or a hardware event-based sampling, and what was the processor name. So, in case of automation, if you are running VTune in Command Line, and you are running it on multiple machines, this is a very good way to figure out where the dataset was run and how it was run.

If you go up to the Bottom-Up view, you can do a deeper dive in your function, in your call stack, in your threading. There are different ways of grouping. You can always specify the grouping time, how you need, either you want it to be on a process-based, module-based, source file-based, or task-based in case if you are using the ITT APIs. You can always take a deeper look within every function by opening its call stack, and it will give you the call stack information.

At the very end, you see this timeline, which is the timeline of each and every individual thread which was created during the runtime of this application. You can always filter it out by a process, by a specific thread, by a specific module or any utilization, which can be like-- which VTune suggests as Poor, Ok, Ideal or Idle. .

If you hover over the timeline over here, you can always see the CPU time and the spin time in case of hotspots.

So, let’s say for example you want your application to be running 100% - utilizing 100% of the CPU time. And at any point of the time, you just see, “OK, there is a small dip in my performance and I just want to analyze what happened”. So, what you can do is you can always zoom in, into the part of the timeline, and VTune will show you what exactly happened during that timeline.

So, for example, you just take Thread 23244 and you see like, “OK, my utilization was 90%, but it just fell down to 40 at a particular time”. And so, what you can do is select that particular timeline and say, “Filtering By Collection”. What VTune will do is it will update your database just for that specific timeline.

So, you will just see the functions which were used or not used when your performance went down. At any point of time, you can just double-click on this function, and if you have your sources and binaries set in a proper way, or configured properly, you will see how much time you have utilized on each individual line of the code.

So, your main bottlenecks, if you're entrusted in your source, or if you're entrusted in your SMB language, whatever optimization you can do on your application to gain performance.

A very good feature of VTune is to collect data and once you optimize data, you can rerun VTune against your application or the collection again. And what it does-- what you can do is right click on two different collections and say, “Compare Results”. So, VTune will give you a very good overview how much you optimized, how much you added an overhead on a similar type of collection, what you collected using VTune.

Now, let’s jump into a-- very quickly jump into a Microarchitecture Exploration results, which is a deeper dive with a hardware event-based sampling collection.

So, this gives you a very good overview on the lapse time, the top tasks, effective CPU utilization, the collection platform. And when you see in the lapse time, it gives you a lot-- a whole lot of metrics. At any point of time, you can always hover over the screen and see what metrics is being used, and what metrics are being flagged out. So, anything which is red is being flagged out by VTune, and it complains to you about it.

So, you can see the CPI rate, which is Cycles Per Instruction, it’s a fundamental performance like how every instructions behaved with every cycle. On this new uPipe background, you see how much your instruction were frontend-bound, memory-bound. Green is everything which is good, how it should act properly, how much was code-bound, and how much was being lost. So, Bad Speculation is an instruction set came into the memory but it never retired. You can always see memory depreciation with L1, L2, and L3. And you can always see how everything was utilized.

Top Tasks is introducing ITT APIs. So, these all five functions were traced using the Intel ITT APIs, and it’s showing up how much task time was spent, what was the count and what was the average task time for each of these tasks.

It complains about CPU utilization, because it was run on an Intel Xeon Processor, and which has 56 cores. So, you can clearly see in this timeline that half of the cores are not being used during the full runtime at all. So, it complains that only an effective utilization of 28.5% of your cores, and that is only 15.9 out of 56. So, there is a high chance for parallelization or if you just want to run it on a lower core down.

So, that’s hardware event-based sampling data. Let’s jump into a DPDK dataset and what a DPDK dataset looks like.

So, this was done for-- to capture PCI change or PCI bandwidth. So, once you run a DPDK-based dataset, you will see the CPU time, effective time, which is very similar to collecting hotspots or a microarchitecture exploration. You'll see all the instruction rate, all the CPI, anything which is marked in red is flagged out, which is not in this case. You will see the inbound PCI read and outbound PCI read, so an average of the read and write from your system memory. You can always see where your fast packet processing is required. So, your polling thread utilization of fetching packets and what was the statistics domains, you know, or what were the core-- basically, the number of packets fetched on a single routine of DPDK. And you can basically, you know, scale it up depending upon how you have configured, and what's your configuration available on your machine.

You can always see a histogram, based on the PCI bandwidth or any other bandwidth, what you want to utilize. Again, at the very end, you'll see the top hotspots and the effective physical core utilization for your whole run.

A very interesting view we share is the Platform View, and which I want to focus and put some time in explaining it. So, on a very high level, you will see your thread, so how many threads were created and what was the timeline. You will see the inbound PCI bandwidth. So, during the whole run, what was the PCI bandwidth, what was the read and write, and the total bandwidth of your PCI. So, basically-- which is very, very important, because you want to achieve the maximum performance using VTune… using your application and VTune can help you optimize that. You can see the CPU frequency.

Basically, you can also see the QPI bandwidth. You can also see the QPI bandwidth and also… and thread platform basically how many threads were created along with the CPU frequency. If you have persistent memory, you can see your persistent memory graph. If you have… if you have any additional memory on your timeline, everything will just show up here. So, in terms of you are using two CPUs, what was the cross-traffic between CPUs, how much was the UPI utilization, how much was the memory utilization, DRAM bandwidth, and basically, all the thread counts.

So, at any point of time, again like how I said earlier, you can always go and check, hover over and zoom in the selection. So, you can see what exactly happened at that particular point of time. And if you want to go deeper, you can always use VTune to configure it for a lesser time collection, or lesser time… lesser time circling and you can always collect better or a deeper dive data.

This particular dataset was collected on runs with FlexRAN by using DFT, FFT, and the physical uplink share channel.

With showing all these datasets and with the time limit, I can show you more and more, but I think this should be enough for everyone to get started with VTune. So, I'm going to stop my live demo and go into the Q&A session.

**Shawn Li**

Thank you, Ashish and Abhinav, it was a great demo and great presentation. We have some questions and I will read through for you.

Question one, “How do I get a copy of oneAPI and is there a cost?”

**Ashish Gupta**

Yes, let me take that one. So, oneAPI is actually available online. If you just type in “Intel oneAPI download”, you can go to software.intel.com and download it. So, technically, there is no cost, right. You can get a one-year free license online. However, that license does not give you access to the older versions, which FlexRAN requires, and it also doesn’t give you access to confidential customer support, which is why you need a commercial license, right.

So, the short answer is, yes, you can get in online, right. The long answer is if you need it for FlexRAN, talk to your Intel account manager.

**Shawn Li**

Thank you. And the next question, “Does VTune support virtual machines?”

**Abhinav Singh**

Yes, VTune does support virtual machines, so we have a handful of virtual machines what VTune supports. So, it depends upon the virtual machine’s visibility how much VTune can look into the Perf of the system. Based upon that, you can always profile an application which is locally present on a virtual machine. Or like, you can always pin it if you have the visibility of the processors which is running in terms of Docker or container within the virtual machine.

**Shawn Li**

Great. Thank you. The next one, “Can you pin a core for collection with VTune?”

**Abhinav Singh**

Yes. The answer to that question is yes, you can always pin a core with a collection. So, you'll have to go into the second box of the Configuration Analyst, which is 'WHAT' tab screen. And at the very end, you will see just like, I think it’s the third last option or the fourth last option in the box which says, “CPU Mask”, and you can always specify the mask of the CPU cores, which you have pinned for your application to be done on.

**Shawn Li**

Thank you. The next one, “How can you collect trace for specific network functions?”

**Abhinav Singh**

That’s actually a very good question, and that’s something how we have integrated DPDK within, and that’s how I demonstrated the different tasks what we saw when we were profiling the DPDK application is through ITT APIs, which the hard part we have already done from VTune engineering side. So, if you have an application, let’s say, based on DPDK or FlexRAN which has millions of lines of code and you just want to profile a specific network workload or a specific network function, and you want to collect traces using VTune, so we have APIs which are called Instrumentation and Trace Technology APIs available with VTune. These are a set of very simple APIs, which basically you create a task, you create a domain, you basically define everything, and you just start and stop a collection within the code.

So, it’s just four hooks between the codes, and you can basically start and stop the collection. And once you run a VTune collection along with the ITT APIs, it will just collect traces for that particular function. So, you can just focus what happened along with the system or your resources available with that particular function.

**Shawn Li**

Great. Thank you. My next question, “My FlexRAN server does not have a GUI, can I still run VTune?”

**Abhinav Singh**

Yes, you can still run VTune if you don’t have a GUI. VTune completely supports every command line option, which is available in the GUI. There is a very good way of generating a command line, which I basically went through my demo and showed with a very small button at the very end. Once you have configured all of your configuration, what options you want, what flags you want, what you need to be turned on and off along with timings, you can just click on the command line clipboard, copy your command line and basically, you know, you can hook it up within your script or you can just use it as a command line option when the GUI is not available.

Having said that, there is a small-- a catch here is basically once you collect your data with the command line, it will give you just some basic information. You can always export to a CSV file for visualization, but our team highly recommends that you copy back the data where a GUI is available, and then you can visualize the data in a much better form zooming into timelines, filtering out the databases and, you know, pinpointing to the exact bottleneck what you have within your application.

**Shawn Li**

Yes, we have another three questions, OK. Following, next one, “Does VTune work on Mac?”

**Abhinav Singh**

So, VTune basically supports collection-- remote collection on Mac, it does not support local collection on a Mac. So, it has capability of visualizing your data and doing remote collection.

**Shawn Li**

Good. And the next one, “The VTune profilers only give the metrics and resource utilizations for hardware and infrastructure only, or it can provide application-level metrics?”

**Abhinav Singh**

So, VTune can provide application-level metrics as well, and that’s what we talked about. There are two different ways of collection software and hardware event-based sampling. So, if you are running a hotspot, you can basically collect every metric of your application, what you are running across-- everything what your application is basically interacting with the CPU or the GPU, in the case of a GPU.

**Shawn Li**

Good. The next one, “Can we use it for network and storage incentive… intention… intensive workload as well?”

**Abhinav Singh**

Yes, so VTune can be used for storage and network intensive workload as well. Depending upon the time of the collection what you want basically. So, VTune collects a lot of data. So, this is for our audience that it collects a lot of data, so it depends how much time you basically want to collect the data to understand what is enough for you to understand. So, if you want to go on a platform level and do a network or a storage compute intensive workload profiling, you use Platform Profiler. And the whole idea is you run your network workload or storage workload for hours and hours, and then you use Platform Profiler to basically view everything on a platform level, you know, and pinpoint your main bottlenecks, whether it’s memory-bound you're not getting enough read and writes or, you know, your network is not scaling up so you're not getting the maximum throughput of the NIC.

So, you can pinpoint your issues and then run VTune Profiler on those specific intervals or your specific collection type and it will help you visualize or optimize your application in a better way.

**Shawn Li**

Great. Thank you. The last one, “For FlexRAN SDK debugging, does it need AVX512 peripheral on host machine in which you're running the VTune program?”

**Abhinav Singh**

So, no. So, you don’t need an AVX512 support on the host machine from where you're running. So, I was running it from my laptop, which basically just supports AVX2. So, my laptop just supports AVX2 and I was basically remoting into a machine using Remote Linux SSH connection, and that was a Xeon-based server which supports AVX512, so you don’t need… the whole collection will be done remotely. Only the visualization will be done on your laptop, so you don’t need a machine where AVX512 is supported.

**Shawn Li**

Great. Great, thank you. This is a great question and answer session and appreciate the audience providing so many questions. Thank you.

And thank you for joining us today. If you have any further questions, please do not hesitate to contact us. Please do not forget giving us a rating for the live recording, so that we can continue to improve our quality of the webinars. And be sure to join us next time, Wednesday, December 1st at 8 a.m. Pacific Time for the Intelligence, Performance, Visibility with Intel Intelligent Fabric.

Thank you again for joining us today. This concludes our webinar.