NETINT

AMPERE.

Building a 300-Channel Video Encoding Server

NETINT VPU Technology with Ampere® Altra® Max Processors set new operational cost and efficiency standards

Increasing costs and advanced video format computing requirements necessitate the reinvention of the video transcoding server to meet OPEX and performance conditions of hyperscale video platforms.

Demand for live video streaming has exploded worldwide. In both developed and emerging markets, operational costs are under pressure while user expectations are expanding, leading to NETINT reimagining the video transcoding server. A new live video server architecture opens new video processing capabilities created in collaboration with Supermicro and Ampere Computing.

The intensive video encoding and transcoding process is offloaded to NETINT VPUs. This allows the host CPU to perform deinterlacing, software decoding, and media processing functions not natively supported in the VPU. The powerful host CPU enables AI inferencing with an industry-first automated subtitling using OpenAI's Whisper model.

Powered by the Ampere® Altra® Max Processor, the server performs video processing and transcoding tasks in a dense, high-performance, and cost-effective manner not possible with x86 processors. Video engineers control the server via FFmpeg, GStreamer, SDK, or NETINT's Bitstreams Edge application interface, making it accessible for deploying and replacing existing transcoding resources or in greenfield installations.

This case study presents how NETINT, Supermicro, and Ampere engineers optimized the system to deliver a reimagined video server to transcode one of 95x 1080i30 streams, 195x 720i30 streams, 365x 576i30 streams, or a combined 100x 576i, 100x 720i,10x 1080i, 40x 1080p30, 40x 720p30 and 10x 576p streams in a single Supermicro MegaDC SuperServer ARS-110M-NR 1U server.

The NETINT Quadra Video Server with Ampere CPU expands system functionality by enabling video formats not natively supported in the NETINT Quadra T1U VPU. Strengthened by the 96-core Altra Max CPU, the server will decode 96 1080i30 H.264 or HEVC streams or up to 320 1080i MPEG-2 streams.

"The punchline is that with an Ampere Altra Max Processor and NETINT VPU, a Supermicro 1U server unlocks a whole new world of value," Alex Liu, Co-founder, NETINT.

NETINT's Vision

Responding to customers' lamenting limited CPU processing and skyrocketing power costs, NETINT built a custom ASIC for one purpose: highest-quality, lowest cost video processing and encoding.

NETINT reinvented the live video transcoding server by combining NETINT Quadra VPUs with Ampere's Altra Max Processor to create a smaller and faster server that costs 80% less to operate and increases throughput by 20x compared to software on x86.

NETINT's Story

Founded in 2015, NETINT's big dream of combining the benefits of silicon with the quality and flexibility of software for video encoding using proprietary ASICs is now a reality. As the first commercial vendor for video processing-specific silicon, NETINT pioneered category development of the video processing unit (VPU).

Why NETINT chose Supermicro

Supermicro is a global technology leader committed to delivering first-to-market innovation for Enterprise, Cloud, AI, Metaverse, and 5G Telco/Edge IT Infrastructure, with a focus on environmentally-friendly products. NETINT selected Supermicro based on their expertly engineered building block approach which allows maximum flexibility for different form factors to accomodate all customer needs. This flexibility enables key cost-optimization through different configurations of memory, hard drives and CPUs.

Requirements to reinvent the Video Server:

- **Engineer smaller and faster**
- **Make it cost 80% less to operate**
- **Increase throughput by 20x**

Why NETINT chose Ampere Processors

NETINT was already familiar with Ampere Computing's high-performance and low-power processors, which perfectly complement NETINT's Quadra VPUs. The Ampere Altra Max cloud native processor is designed for a new era of computing and an energy constrained world, delivering unprecedented efficiency and performance.

From web and video service infrastructure to CDNs to demanding AI inference, Ampere products are the most efficient dense computing platforms on the market. The benefits of using a cloud-native processor like Ampere Altra Max processor include improved efficiency and scalability, have great synergy with NETINT high performance and energy efficient VPU's.

Problem:

Could Ampere Altra Max deinterlace 100 576i, 100 720i, and 10 1080i simultaneous video streams that legacy x86 processors couldn't in a cost-effective 1RU form factor?

Here's how Ampere Responded

Engineers from NETINT, Supermicro & Ampere unlocked the high performance available with NETINT's Quadra VPU and Ampere Altra Max 96-core processor to redefine the live stream video server. Initial results with Ampere Altra Max using FFmpeg 5.0 were encouraging compared to legacy x86 processors but did not meet NETINT's goal to increase throughput 20x while reducing costs by 80%.

Ampere engineers studied different deinterlacing filters available in FFmpeg and investigated recent Arm64 optimizations available in updated FFmpeg releases. An FFmpeg avfilter patch that provides optimized assembly implementation using Arm64 NEON SIMD instructions showed a significant performance increase in video deinterlacing with up to 2.9x speedup using FFmpeg 6.0 compared to FFmpeg 5.0, shown in Figure 1.

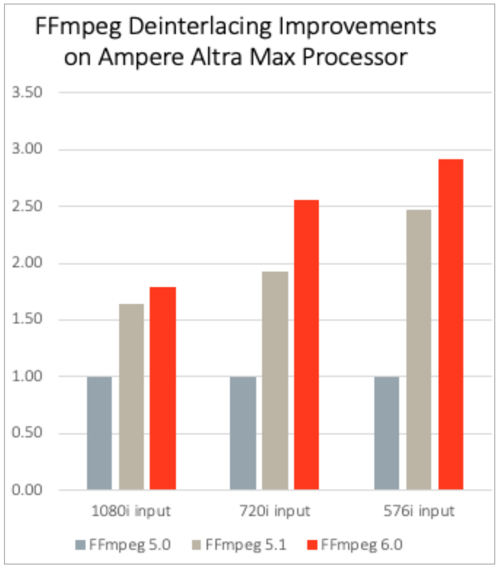


Figure 1: FFmpeg Arm64 Deinterlacing Optimizations improved performance by up to 2.9x faster on the Ampere Altra Max processor when switching from FFmpeg 5.0 to 6.0.

Performance Challenges

NETINT, Supermicro, and Ampere engineers went to work on the full video workload, combining CPU-based video deinterlacing and transcoding using NETINT's Quadra VPUs. Though it demonstrated outstanding results running the deinterlacing jobs with full video workload, it fell short of our performance target. Combining our expertise in hardware and software, our team analyzed the root cause and delivered the aggressive requirements of using just 50-60% of Ampere processor's CPU utilization, allowing headroom for future features.

Averages	
CPU	all
%usr	36.29
%nice	0
%sys	15.29
%iowait	10.48
%irq	0
%soft	9.3
%steal	0
%guest	0
%gnice	0
%idle	28.64
CPU Utilization	71.36

Initial results didn't meet the target of simultaneously transcoding 100x 576i, 100x 720i , 10x 1080i , 40x 1080p30, 40x 720p30 and 10x 576p inputs videos. Investigating closer, we discovered performance was close to the goal, though unexpectedly slowed down over time.

Following Ampere's Performance Analysis Methodology for Optimizing Altra Family CPUs, we first characterized platform-level performance metrics - Figure 2 shows the mpstat utility data: initially, the system was running within ~4% of the performance target yet was only running at ~71% overall CPU utilization, with ~36% in user space (mpstat %usr), and ~35% is system related tasks – kernel time (mpstat %sys), waiting for IO (mpstat's %iowait) and soft interrupts (mpstat %soft).

The fact that the system was idle ~29% confirmed something was blocking performance.

Figure 2. This mpstat utility output shows the system is idle 100.0 - 71.4 = 28.6% of the time during initial performance analysis when the system wasn't meeting the performance target. This revealed the limited system performance.

With the large percentage of software interrupts and IO wait time, we investigated interrupts using the soft irq tool in BCC, which provides BPF-based Linux IO analysis, networking, monitoring, and more. The soft irq tool traces the Linux kernel calls to measure the latency for all the software interrupts on the system, outputting a histogram graph showing the latency distribution.

The BCC tools are very powerful and easy to run. It showed ~20 microsecond average latency in the driver used by NETINT's VPU while handling ~40K interrupts/s. As our performance problem was of the order of milliseconds, the BCC soft irq tool showed that software interrupts weren't limiting performance, so we continued to investigate what was limiting performance.

Figure 3
Data shows the BCC softirq tool measuring software interrupt latency. The softirq block device output shows block IRQ average latency of ~12 usecs, which is thus not critical for overall performance when running at 30 FPS or 33 milliseconds per frame.

```
softirq = block
     usecs               : count     distribution
         0 -> 1          : 222      |                                        |
         2 -> 3          : 118363   |*****************                       |
         4 -> 7          : 276249   |****************************************|
         8 -> 15         : 164502   |***********************                 |
        16 -> 31         : 49870    |*******                                 |
        32 -> 63         : 69452    |**********                              |
        64 -> 127        : 34325    |****                                    |
       128 -> 255        : 9134     |*                                       |
       256 -> 511        : 730      |                                        |
       512 -> 1023       : 284      |                                        |
      1024 -> 2047       : 714      |                                        |
      2048 -> 4095       : 5646     |                                        |
      4096 -> 8191       : 2171     |                                        |
      8192 -> 16383      : 7        |                                        |
```

Next, we used the perf record/perf report utilities to measure various Performance Measurement Unit (PMU) counters to characterize the low-level details of how the application was running on the CPU, looking to pinpoint performance bottleneck(s).

Since the team didn't know the source of the limiting performance, we collected a PMU counter data to measure CPU utilization (CPU cycles, CPU instructions, Instructions per Clock, frontend, and backend stalls), cache and memory access, memory bandwidth, and TLB access.

The system, after reboot, reached ~96% of the performance target and degraded to ~60% after running many jobs; we collected perf data after reboot and when the performance was poor.

Analyzing the PMU data to look for the most significant differences in the good and poor performance cases, the kernel function alloc_and_insert_iova_range stood out by taking 40x more CPU cycles in the poor performance case.

Searching Linux kernel source code via the very powerful live grep website showed this function is related to IOMMU. Rebooting the kernel with the iommu.passthrough=1 option resolved the performance degradation overtime issue by reducing the TLB miss rate, putting us at ~96% of the performance target but still short of the target.

NETINT engineers made final performance increases when they saw additional Arm64 deinterlacing optimizations available in FFmpeg mainline. This delivered on the performance goals while reduced the overall CPU utilization to 50-60%, down from 70%.

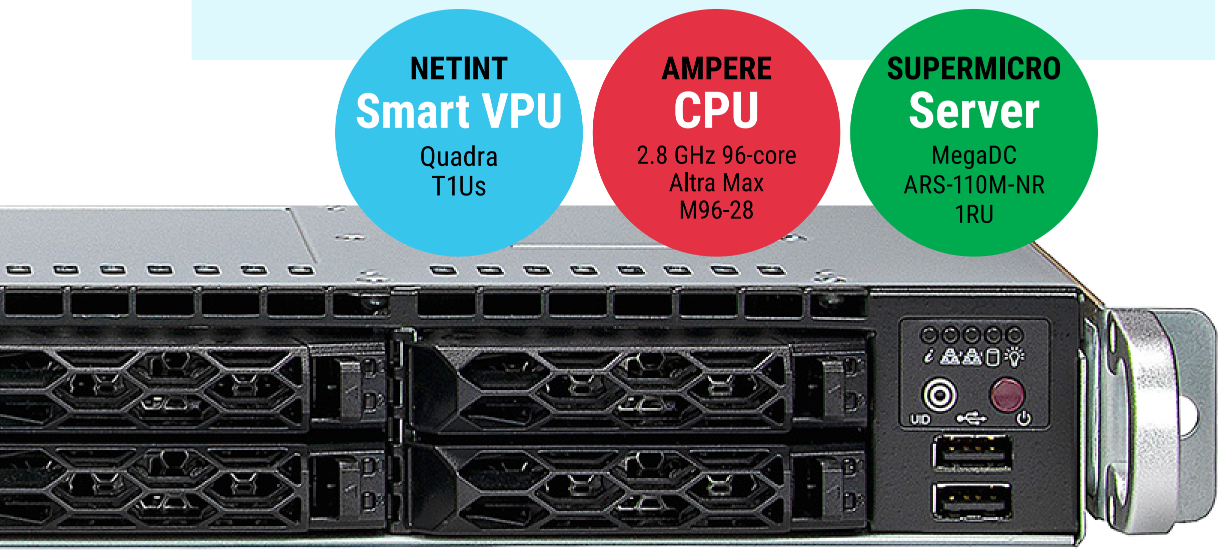
Figure 4
Perf utility output showing performance critical functions when the system was running slow and fast. The function __alloc_and_insert_iova_range shows a large increase in the CPU cycles and Stall Frontend. This led us to solve the performance degradation over time using the Linux kernel boot option iommu.passthrough=1.

Function	**Stall Frontend (%)**			**CPU Cycles**			**CPU Instructions**		
	Slow	**Fast**	**% Diff**	**Slow**	**Fast**	**% Diff**	**Slow**	**Fast**	**% Diff**
ffmpeg_ffmpeg_filter_line_c	6.0%	7.0%	**86.2%**	49.4%	49.0%	**100.8%**	69.0%	69.7%	**98.9%**
[kernel.kallsyms]_swapper_cpuidle_enter_state	24.0%	18.6%	**129.0%**	2.6%	1.9%	**131.4%**	0.2%	0.2%	**104.8%**
[kernel.kallsyms]_ffmpeg_arch_local_irq_enable	6.7%	7.0%	**96.7%**	2.5%	2.0%	**120.6%**	1.0%	0.6%	**154.0%**
[kernel.kallsyms]_swapper_arch_local_irq_enable	2.3%	1.5%	**159.2%**	1.5%	0.5%	**288.7%**	0.8%	0.2%	**405.0%**
libc.so.6_ffmpeg_0x0000000000097ecc	0.1%	0.1%	**75.0%**	1.1%	1.5%	**76.9%**	0.2%	0.2%	**110.0%**
[kernel.kallsyms]_ffmpeg_alloc_and_insert_iova_range	1.9%	0.0%	**18900.0%**	0.8%	0.0%	**4000.0%**	0.2%	#N/A	**#N/A**

The Results

The result is the NETINT 300 Channel Live Stream Video Server based on a collaboration of NETINT, Supermicro and Ampere which can simultaneously transcode 95x 1080i30 streams, 195x 720i30 streams, 365x 576i30 streams or a combined 100x 576i, 100x 720i , 10x 1080i , 40x 1080p30, 40x 720p30 and 10x 576p streams in a Supermicro MegaDC SuperServer ARS-110M-NR 1U server.

This server expands the system functionality to enable running video workloads that require high-performance CPU performance in a dense, power, and cost-effective 1RU server.



Call to Action

NETINT's vision to reimagine the live video server based on customer demands resulted in the NETINT Quadra Video Server Ampere Edition in a Supermicro 1U server chassis, unlocking a whole new world of value for customers who need to run video workloads that require high-performance CPU processing in addition to video transcoding with NETINT's VPUs. Read here >

The Alex Liu and Mark Donnigan from NETINT, Sean Varley from Ampere Computing, and Ben Lee from Supermicro webinar available to watch on NETINT's YouTube channel, "How to Build a Live Streaming Server that delivers 300 HD interlaced channels" provides additional information. Watch now >

Other video workloads that are excellent to run on this server include AI inference processing, which NETINT recently announced and demonstrated at NAB 2024 - NETINT unveiled the Industry-First Automated Subtitling Feature With OpenAI Whisper running on Ampere Altra Max processor. Read here >

amperecomputing.com

netint.com

Smart VPUs are engineered for high-volume, live video streaming and interactive applications.

We want to hear what you're building.
Allow us to share our industry insights
to improve your outcome.

sales@netint.com
netint.com