

Using Intel Advanced Matrix Extensions with VMware ESXi

Planning / Implementation

Artificial Intelligence is transforming how we work and live every day, and it is evolving rapidly, continues to make groundbreaking advancements, drawing interest from business, research and consumer. Business can benefit from applying AI in various scenarios, that range from image processing, recommendation systems, natural language processing and machine learning.

At its essence, AI is the process of converting raw data into actionable insights. The typical AI workflow occurs in three stages: data engineering, AI training, and inference. Each stage has different memory, compute, and latency requirements. GPUs were originally designed for rendering graphics but have become crucial for AI due to their ability to handle parallel processing tasks efficiently. However the latest GPUs are more expensive and hard-to-procure, organizations need significantly more processing power to increase the performance of AI/ML workloads.

The 4th generation and the 5th generation Intel Xeon Scalable Processors are the unique, scalable platform optimized for different workloads accelerations including HPC, AI, BigData, Networking etc, with higher performance and lower total cost of ownership. The new built-in AI acceleration engine Intel Advanced Matrix Extensions (Intel AMX), are extensions to the x86 instruction set architecture for microprocessors from Intel designed to work on matrices to accelerate AI and machine learning workloads.

The AMX supports 16-bit brain floating point (BF16) and 8-bit integer (INT8) data types of low-precision matrix operations to boost the instructions per cycle for AI applications. With the high-speed matrix multiplication enabled by Intel AMX, the Intel Xeon Scalable processor can quickly pivot between optimizing general computing and AI workloads.

Intel AMX introduces the new extensions to the x86 Instruction Set Architecture (ISA) to work on matrices and which accelerate matrix multiplication in AI workloads.

Intel AMX consists of two components as the Figure 1 shows:

- A set of 2-dimensional registers (tiles) representing sub-arrays from a larger 2-dimensional memory image
- An accelerator able to operate on tiles

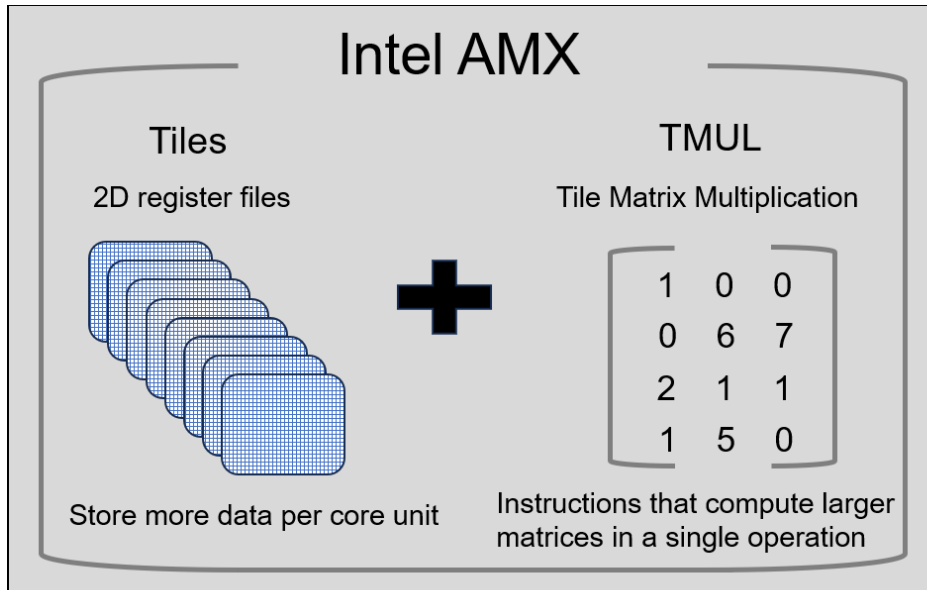


Figure 1. Intel AMX

Intel AMX will store larger chunks of data in each core and then compute larger matrices in a single operation. The first implementation is called TMUL (tile matrix multiply unit) that comprises a grid of fused multiply-add units capable of operating on tiles. In its initial form, it implements a set of up to eight tiles (named TMM0 ... TMM7), which are arrays with 16 rows of size of 64 bytes. Load a tile representing a small section from a larger image in memory, operate on that tile, and repeat with the next tile that represents the next portion of the image. When done, store the resultant tile to memory.

Figure 2 shows a conceptual diagram of the Intel AMX architecture. The Intel architecture host drives the algorithm, the memory blocking, loop indices and pointer arithmetic. Intel AMX instructions are synchronous in the Intel architecture instruction stream and the memory loaded and stored by the tile instruction is coherent with respect to the host's memory accesses.

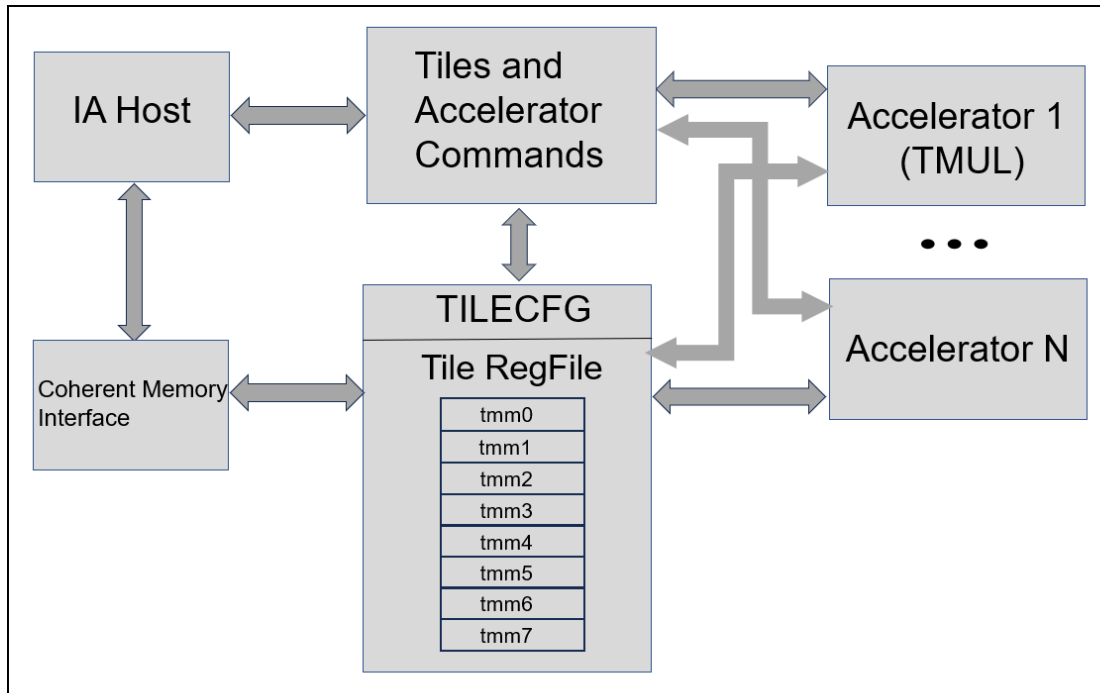


Figure 2. Intel AMX architecture

The AMX introduces 12 new instructions and comprises of three sub-extensions:

- AMX-TILE
- AMX-INT8
- AMX-BF16

The 12 instructions are listed in the following table.

Table 1. Intel AMX instructions

Instruction	Sub-Extension	Description
LDTILECFG	AMX-TILE	Load tile configuration
STTILECFG	AMX-TILE	Store tile configuration
TILELOADD	AMX-TILE	Load data into tile
TILELOADDT1	AMX-TILE	Load data into tile with hint to optimize data caching
TILESTORED	AMX-TILE	Store tile
TILERELEASE	AMX-TILE	Release tile
TILEZERO	AMX-TILE	Zero tile
TDPBSSD	AMX-INT8	Dot product of signed bytes with dword accumulation
TDPBSUD	AMX-INT8	Dot product of signed/unsigned bytes with dword accumulation
TDPBUSD	AMX-INT8	Dot product of unsigned/signed bytes with dword accumulation
TDPBUUD	AMX-INT8	Dot product of unsigned bytes with dword accumulation
TDPBF16PS	AMX-BF16	Dot product of BF16 tiles accumulated into packed single precision tile

The default floating point precision of a CPU primitive is single-precision floating point (float 32 or FP32). FP32 is a standard 32-bit floating-point data type used to train deep learning models and for inferencing. This data type is more computationally demanding than other data types but typically achieves high accuracies.

By reducing the size of data during computation will achieve better performance. Intel AMX supports BF16 and INT8 data types, augmenting the optimizations from Intel AVX512 and Intel deep learning boost to enable fast and efficient AI and deep learning across various industries and use cases. BF16 is a truncated version of FP32 used for training and inference, and it offers similar accuracy but faster computation. INT8 offers higher performance and is the least computationally demanding data type, ideal for real-time application and matrix multiplication tasks where speed and efficiency are a priority. It has the minimal impact on accuracy. The comparison of different data types is shown in Figure 3.

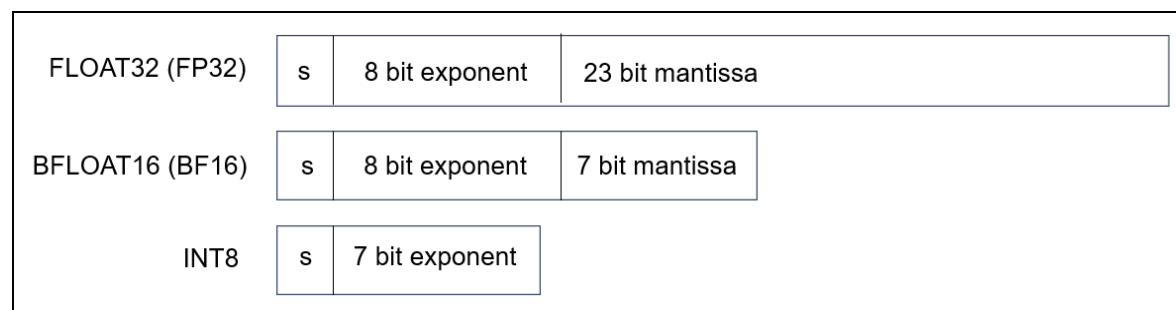


Figure 3. Different data type formats

Test configuration

The test bed configuration of ThinkSystem ST650 V3 is listed in the following table. Intel AMX instructions are supported on ESXi 8.0 U1 and above with VMs using virtual HW version 20 and above. The guest OS running Linux should use kernel version 5.16 or later. Kernel version 5.19 or later is recommended.

Table 2. ThinkSystem ST650 V3 server configuration

Component	Configuration
Server	ThinkSystem ST650 V3 Server
CPU	2x Intel Xeon Gold 6534 Processors (Emerald Rapids)
Memory	2x DDR5 5600MHz 16GB DIMMs
HDD	240GB SATA SSD
Host OS	ESXi 8.0 U3 Custom Image for Lenovo ThinkSystem
Guest VM OS	Ubuntu Server LTS 24.04.1

Implementing Intel AMX

This section describes the steps we took to configure and use Intel AMX in VMware ESXi 8.0 Update 3.

The steps to implement Intel AMX are as follows:

1. Power on the server and install VMware ESXi 8.0U3
2. Connect to the vCenter Server 8.0U3 by using the vSphere Client and add the host to the data center.
3. Create a virtual machine and install a guest OS (e.g., Ubuntu server 24.04.1) with kernel version 5.19 or later.
4. Find out the virtual HW version a VM is using by one of the two ways:
 - o In the vCenter Server, go to the Updates tab of the VM, and click the CHECK STATUS button, the result is displayed at the right-hand side, as figure 4 shows, the virtual HW version is 21.
 - o The other way is Check VMware products and virtual HW version on Broadcom web: <https://knowledge.broadcom.com/external/article?articleNumber=315655>

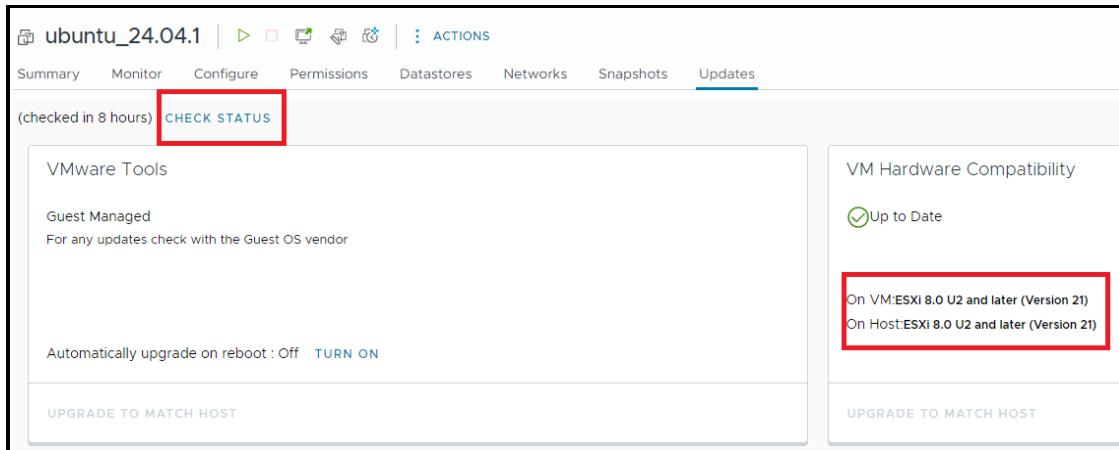


Figure 4. Display VM Virtual HW Version

5. Login to the guest VM, use the following command on the terminal and search for AMX in the flags section, and make sure the AMX flags (amx_bf16, amx_tile and amx_int8) are supported by the CPU.

```
~# cat /proc/cpuinfo | grep -i amx
```

```
occhen@occhen:~$ cat /proc/cpuinfo |grep -i amx
flags       : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall
tsc arch_perfmon rep_good nopl xtopology tsc_reliable nonstop_tsc cpuid tsc_known_freq pni pclmulqdq sse3 fma cx16 pcid sse4_1
eadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch ssbd ibrs ibpb stibp ibrs_enhanced fsgsbase tsc_adi
pcid avx512f avx512dq rdseed adx smap avx512ifma clflushopt clwb avx512cd sha_ni avx512bw avx512vl xsaveopt xsavec xgetbv1 xsae
6_wbnoinvd arat avx512vbmi umip pku ospke avx512_vbmi2 gfni vaes vpclmulqdq avx512_vnni avx512_bitalg avx512_vpopcntdq rdpid c
clear serialize amx_bf16 avx512_fp16 amx_tile amx_int8 flush_l1d arch_capabilities
```

Figure 5. AMX Flags from CPU Info

We can also use cpuid command to dump CPUID information and ensure that the VM can access the hardware's AMX instructions.

```
~# cpuid -1 -1 7 -s 0 | grep -i amx
```

```
occhen@occhen:~$ cpuid -1 -1 7 -s 0 |grep -i amx
  AMX-BF16: tile bfloat16 support           = true
  AMX-TILE: tile architecture support      = true
  AMX-INT8: tile 8-bit integer support     = true
occhen@occhen:~$ _
```

Figure 6. AMX Flags from cpuid Command

Intel provides a sample code to demonstrate testing the new instructions using intrinsic function. The test code will multiply matrices A and B of size 16x64 containing INT8 value and accumulate the result to a 16x16 matrix C containing INT32 value. It's simplified to highlight use of Intel AMX instruction to configure the tiles, load data from memory into tiles, perform INT8 matrix multiplication on tile data and copy the result from tiles to memory. We will explain some code snippets below.

1. Declare the tile config data structure that will hold the control register for tile configuration and format defined as a 64-byte memory location.

```
typedef struct __tile_config
{
    uint8_t palette_id;
    uint8_t start_row;
    uint8_t reserved_0[14];
    uint16_t colsb[16];
    uint8_t rows[16];
} __tilecfg;
```

Figure 7. Struct Tile Config Declaration

2. Initialize the tile config variable with the specific information given by the matrices and the `_tile_loadconfig()` intrinsic function is used to load the tile configuration metadata from the 64-byte memory location specified by `tileinfo`.

```
static void init_tile_config (__tilecfg *tileinfo)
{
    int i;
    tileinfo->palette_id = 1;
    tileinfo->start_row = 0;

    for (i = 0; i < 1; ++i)
    {
        tileinfo->colsb[i] = MAX_ROWS;
        tileinfo->rows[i] = MAX_ROWS;
    }

    for (i = 1; i < 4; ++i)
    {
        tileinfo->colsb[i] = MAX_COLS;
        tileinfo->rows[i] = MAX_ROWS;
    }

    _tile_loadconfig (tileinfo);
}
```

Figure 8. Tile Config Variable Initialization

3. The application will need to invoke a system call to request access to Intel AMX feature. This is performed using the `arch_prctl()` command (`ARCH_REQ_XCOMP_PERM`) to request permission to use Intel AMX and can require XSTATE component to be enabled. Intel AMX is an XSAVE-enabled feature, meaning that it requires use of the XSAVE feature set for their enabling.

```
static bool set_tiledata_use()
{
    if (syscall(SYS_arch_prctl, ARCH_REQ_XCOMP_PERM, XFEATURE_XTILEDATA))
    {
        printf("\n Fail to do XFEATURE_XTILEDATA \n\n");
        return false;
    }
    else
    {
        printf("\n TILE DATA USE SET - OK \n\n");
        return true;
    }

    return true;
}
```

Figure 9. Application Request to Access to Intel AMX

Summary

As part of Intel AI Engines, Intel AMX was designed to balance inference, the prominent use case for a CPU in AI applications, with more capabilities for training. Selecting the Intel Xeon processors with Intel AMX for new AI deployment is an efficient and cost-effective approach to accelerating AI workloads.

VMware announced a collaboration with Intel to help customers accelerate the adoption of AI and enable private AI everywhere – across data centers, public cloud and edge environments, develop and deploy classical machine learning models and generative AI applications on the infrastructure with built-in AI acceleration and managed by VMware Cloud Foundation.

The Intel AI software suite and VMware Cloud Foundation are validated on Lenovo ThinkSystem and ThinkAgile servers with 4th and 5th Gen Intel Xeon Scalable processors. For more information, see the Solution Brief, [VMware Private AI with Intel on Lenovo ThinkAgile VX V3 and ThinkSystem V3](#) .

Reference

For more information, see these resources:

- Intel Advanced Matrix Extensions Overview
<https://www.intel.com/content/www/us/en/products/docs/accelerator-engines/advanced-matrix-extensions/overview.html>
- Developer Resources from Intel and VMware by Broadcom
<https://www.intel.com/content/www/us/en/developer/ecosystem/vmware.html>
- Intel Intrinsic Guide
<https://www.intel.com/content/www/us/en/docs/intrinsic-guide/index.html>
- AMX Sample code
<https://github.com/intel/AMX-TMUL-Code-Samples/tree/main>

Author

Alpus Chen is an OS Engineer at the Lenovo Infrastructure Solutions Group in Taipei, Taiwan. As a specialist in Linux and VMware technical support for several years, he is interested in operating system operation and recently focuses on VMware OS.

Thanks to the following specialists for their contributions and suggestions:

- Skyler Zhang, Lenovo VMware Engineer
- Gary Cudak, Lenovo OS Architect
- David Watts, Lenovo Press

Related product families

Product families related to this document are the following:

- [Processors](#)
- [VMware vSphere](#)

Notices

Lenovo may not offer the products, services, or features discussed in this document in all countries. Consult your local Lenovo representative for information on the products and services currently available in your area. Any reference to a Lenovo product, program, or service is not intended to state or imply that only that Lenovo product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any Lenovo intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any other product, program, or service. Lenovo may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

Lenovo (United States), Inc.
8001 Development Drive
Morrisville, NC 27560
U.S.A.
Attention: Lenovo Director of Licensing

LENOVO PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Lenovo may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

The products described in this document are not intended for use in implantation or other life support applications where malfunction may result in injury or death to persons. The information contained in this document does not affect or change Lenovo product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Lenovo or third parties. All information contained in this document was obtained in specific environments and is presented as an illustration. The result obtained in other operating environments may vary. Lenovo may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any references in this publication to non-Lenovo Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this Lenovo product, and use of those Web sites is at your own risk. Any performance data contained herein was determined in a controlled environment. Therefore, the result obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

© Copyright Lenovo 2024. All rights reserved.

This document, LP2067, was created or updated on November 27, 2024.

Send us your comments in one of the following ways:

- Use the online Contact us review form found at:
<https://lenovopress.lenovo.com/LP2067>
- Send your comments in an e-mail to:
comments@lenovopress.com

This document is available online at <https://lenovopress.lenovo.com/LP2067>.

Trademarks

Lenovo and the Lenovo logo are trademarks or registered trademarks of Lenovo in the United States, other countries, or both. A current list of Lenovo trademarks is available on the Web at <https://www.lenovo.com/us/en/legal/copytrade/>.

The following terms are trademarks of Lenovo in the United States, other countries, or both:

Lenovo®

ThinkAgile®

ThinkSystem®

The following terms are trademarks of other companies:

Intel® and Xeon® are trademarks of Intel Corporation or its subsidiaries.

Linux® is the trademark of Linus Torvalds in the U.S. and other countries.

Other company, product, or service names may be trademarks or service marks of others.