High Performance Computing:
Tuning Guide for AMD EPYC™ 7002 Series Processors

Publication #	**56827**	Revision:	**1.0**
Issue Date:	**January 2020**		
Authors:	**Anre Kashyap**		

Advanced Micro Devices

Contents

Revision History

Date	Revision	Description
January, 2020	1.0	Initial Public Release

Purpose

This document is intended to provide general guidance getting started with HPC workloads on AMD EPYC™ 7002 Series Processor based systems. This is not meant to be an all-inclusive guide. This guide will provide a general starting point from which workloads can be tuned for each use case.

Chapter 1 Introduction

This tuning guide provides detailed descriptions of parameters that can optimize performance on servers with AMD EPYC™ 7002 Series processors in them. The default configurations on hardware and BIOS from different OEM vendors may not provide the best possible performance on all OS platforms and for all workloads. To enable optimization on a per platform and workload level, this guide calls out

- BIOS settings that can impact performance
- Hardware configuration best practices
- Supported versions of operating systems and optimization hooks on them
- Workload specific settings in BIOS and operating systems for a variety of workloads

1.1 Prerequisites

This document is intended for a technical audience with a background of configuring servers.

- Administrative access to the Server's Management Interface (BMC) as well as the operating system is required.
- Familiarity with OEMs Server's Management Interface (BMC) is strongly recommended.
- Familiarity with the OS specific tools for configuration, monitoring and troubleshooting is strongly recommended.

1.2 History

The AMD EPYC™ 7002 Series Processors are built with leading-edge 7nm technology, AMD ‘Zen 2’ core and microarchitecture. The AMD EPYC™ SoC offers a consistent set of features across 8 to 64 cores, including 128 lanes of PCIe® Gen 4, 8 memory channels and access to up to 4 TB of high-speed memory. AMD EPYC™ 7002 Series processors are built with the following specifications:

AMD EPYC™ 7002 Series	
Process technology	7nm
Max number of cores	64
Max memory speed	3200MHz
Max memory capacity	4TB
Peripheral Component Interconnect	128 lanes (max) PCIe®Gen4 per socket

Chapter 2 Microarchitecture Overview

2.1 Microarchitecture

Processor cores, memory controllers, I/O controllers, and security are incorporated into a Multi-Chip Module (MCM) of the AMD EPYC™ 7002 Series Processors.

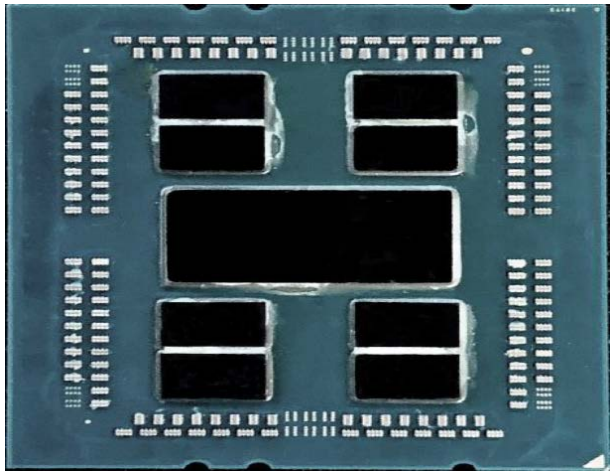


Figure 1 EPYC 7002 Configuration with 8 Core Complex Dies (CCDs) and central I/O Die (IOD)

2.2 Zen 2 core

The EPYC 7002 Series processor is based the new Zen2 processor core, that includes an L1 write-back cache. Each core can support Simultaneous Multi-threading (SMT), allowing 2 execution threads to execute simultaneously per core. Each core includes a private 512KB L2 cache.

2.3 Core Complex Die (CCD) and Core-Complex (CCX)

Up to four Zen2 cores share a 16MB (last level) L3 cache. While the two L3 Caches are on the same chiplet, they are separate. The 4 cores and their associated caches are referred to as a Core-Complex (CCX). Each Core Complex Die (CCD) contains 2 CCXs

Figure 2 Two Core Complexes (CCXs) on a Core Complex Die (CCD)

Two CCDs may be abstracted as a quadrant. The CCDs connect to memory, I/O, and each other through the I/O Die (IOD). There is support for up to 8 memory channels per socket.

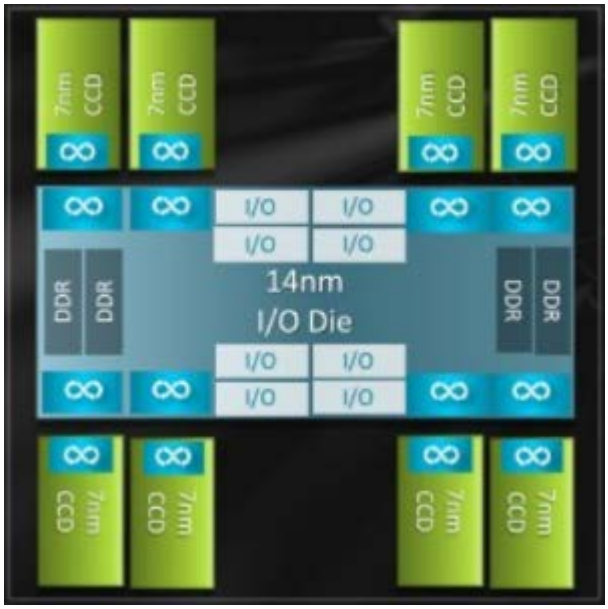


Figure 3 Single socket EPYC 7002 Processor internal connection between CCDs and Memory through memory IOD

2.4 Memory and I/O Layout

Each EPYC 7002 Series processor supports 8 memory channels. Each memory channel supports up to 2 DIMMs. Based upon BIOS settings these channels can be interleaved across a quadrant (2-way), all the way through 16-channel interleave, that is, across all memory channels of a 2-socket system. The system can have access to a maximum of 4TB of DDR4 memory at 3200MHz per processor.

The PCI subsystem provides up to 128 lanes of high speed I/O.

While all memory and I/O connect to the single I/O Die, they can be abstracted into separate quadrants each with 2 DIMM channels and 32 I/O lanes.

Two EPYC 7002 SoCs are interconnected via Socket to Socket Global Memory Interconnect (xGMI) links, part of the Infinity Fabric which connects all the components of the SoC together.

2.5 NUMA

The EPYC 7002 Series processors use a Non-Uniform Memory Access (NUMA) Micro-architecture. The four logical quadrants in an AMD EPYC 7002 Series processor (as described in *Core Complex Die (CCD) and Core-Complex (CCX)*) allow the processor to be partitioned into different NUMA domains. These domains are designated as NUMA per socket (NPS).

2.5.1 NPS1

The processor is a single NUMA domain, i.e. all the cores on the processor, all memory connected to it and all PCIe devices connected to the processor are in one NUMA domain. Memory is interleaved across the eight memory channels.

2.5.2 NPS2

The processor is partitioned into two NUMA domains. Half the cores and half the memory channels connected to the processor are grouped together into one NUMA domain. Memory is interleaved across the four memory channels in each NUMA domain.

2.5.3 NPS4

The processor is partitioned into four NUMA domains. Each logical quadrant of the processor is a NUMA domain. Memory is interleaved across the two memory channels in each quadrant. PCIe devices will be local to one of four NUMA domains on the processor depending on the quadrant of the IO die that has the PCIe root for that device.

2.5.4 L3 Cache as NUMA Domain

Each L3 Cache (as explained in *Core Complex Die (CCD) and Core-Complex (CCX)*) is exposed as a NUMA node. On a dual processor system, with up to 16 L3 Caches per processor, this setting will expose 32 NUMA domains.

Using BIOS settings, each server can be configured as NPS1, NPS2 or NPS4, with an additional option to configure L3 cache as NUMA nodes.

AMD EPYC 7002 Series processors are available in different core counts per processor and not all of them can support all NPS settings. See *https://developer.amd.com/wp-content/resources/56338_1.00_pub.pdf* for details on NUMA architecture and settings.

Chapter 3 Hardware Configuration Best Practices

3.1 Memory Configurations

For optimal performance, populate 8 DIMMs for 1 DIMMs per Channel (DPC) configuration, or 16 DIMMs for 2 DPC configuration, per processor. Other configurations, such as 12 DIMMs per processor, does not provide optimal performance. 1 DPC configuration enables the system to run the memory DIMMs at the highest possible speed, while 2 DPC will typically require slightly reduced memory speed.

3.1.1 Platforms that support previous generations of AMD EPYC

- Platforms designed for AMD EPYC 7001 Processors may be compatible with AMD EPYC 7002 Processors. Contact your OEM to inquire about support.
- Contact your OEM to determine the Maximum Memory Bus Frequency supported on your platform.
- For throughput sensitive applications, to obtain higher IO throughput, Maximum Memory Bus Frequency can be set to the maximum allowed, provided your Memory DIMM hardware supports it. However, in some cases, the Infinity Fabric Clock on these platforms may not synchronize with the maximum Memory Bus Frequency supported by the OEM. This unsynchronized behavior can lead to higher latency.
- For latency sensitive applications, better performance is obtained by setting the Maximum Memory Bus Frequency to 2667 MT/s, since this frequency synchronizes with the Infinity Fabric Clock.

3.1.2 Platforms specifically designed for AMD EPYC 7002

- Platforms specifically designed for AMD EPYC 7002 should support Maximum Memory Bus Frequency. Contact your OEM to determine the Maximum Memory Bus Frequency supported.
- The Maximum Memory Bus Frequency supported on these platforms should be 3200 MT/s.
- For throughput sensitive applications, to obtain higher IO throughput, Maximum Memory Bus Frequency can be set to the maximum allowed (3200 MT/s) provided your Memory DIMM hardware supports it. In this case, the Infinity Fabric Clock on these platforms will not be optimally synchronized with a Memory Bus Frequency of 3200 MT/s. This can a slight increase in memory access latency.
- For latency sensitive applications, memory access latency can be reduced by setting the Maximum Memory Bus Frequency to 2933 MT/s or 2667 MT/s, in order to synchronize with the Infinity Fabric Clock. The appropriate Memory Bus Frequency for synchronized mode will depend on the AMD EPYC 7002 product family.

3.1.3 PCI Subsystem

For I/O intensive workloads, performance can be improved by executing the workload on the same CPU socket that connects to the I/O device being used. For example, for a networking intensive operation, place the workload on the socket that the NIC is connected to. Tools, such as lstopo on Linux, help determine the connectivity between PCI devices and sockets.

Chapter 4 BIOS Settings

4.1 General Usage Recommended BIOS Settings

This section provides general guidelines on the recommended BIOS settings for BareMetal HPC workloads.

Note: The guidelines below are intended to get you started with the tuning. Make sure to evaluate your needs and apply the appropriate settings that are best for your requirements. Check your server manufacturer's guide if you cannot find the exact options below. These options represent changes from the default BIOS settings on AMD reference platforms, but not necessarily on every server platform.

- x2APIC -> Enabled
- SMT -> Disabled
- NPS -> 4
- (Set Memory Frequency Properly)
- APBDIS -> 1
- Fixed SOC Pstate -> P0
- Preferred IO

4.2 Explanation of BIOS Specific Settings

4.2.1 Simultaneous Multi-Threading (SMT)

In HPC workloads the SMT is typically turned off. If you are not in a compute bound scenario you may see some benefit from SMT. If your code is not licensed per core or if there is no monetary impact for enabling SMT you may want to experiment to see if it is beneficial for your workload.

- Enabled: This allows 1 core to execute 2 threads.
- Disabled: This allows 1 core to execute 1 thread.

4.2.2 CCD Control

This option allows you to modify the number of active CCDs in the processor. It can be used in combination with Core Control to change the effective layout of the part.

4.2.3 Core Control

This option allows you to modify the number of active cores in a CCX. The options are listed as (x+x) where x is the number of active cores per ccx.

For example: Setting this to (2+2) means there are 2 active cores per CCX and 4 active cores per CCD. If the part has 8 CCDs, then you have a total of 32 cores. 4 cores per CCD * 8 CCDs = 32 total cores.

Note: CCD and Core control options are typically used to simulate the behavior of different part configurations with your workload. This experiment can help you to identify the best part configuration for your workload and needs.

4.2.4 x2APIC

This option helps the operating system deal with interrupts more efficiently in high core count configurations. It is recommended to enable this option. This option must be enabled if using > 255 threads.

4.2.5 NPS

This option sets the NUMA nodes Per Socket. In many HPC applications, ranks and memory can be pinned to cores and NUMA nodes, and the typical recommendation in this case is to use the NPS4 option. If your workload is not very well NUMA aware or suffers when NUMA complexity increases, you can experiment with NPS1.

4.2.6 Memory Frequency, Infinity Fabric Frequency, and coupled vs uncoupled mode

The Memory clock and the Infinity Fabric clock can either run at synchronous frequencies, called coupled mode, or at asynchronous frequencies, called uncoupled mode.

AMD EPYC supports DDR4 frequencies up to 3200 MT/s, however the fabric clock can be synchronous to a maximum speed of 2933 MT/s (or 2667 MT/s, for lower-power Group B infrastructure parts).

If your memory is clocked at or lower than 2933 MT/s, the memory and fabric will always run in coupled mode which will provide the lowest memory latency.

If you are running DDR4 memory at 3200 MT/s, the memory and fabric clocks will run in uncoupled mode. This provides slightly higher bandwidth at the cost of increased memory latency.

If your system supports 3200 MT/s memory, you can experiment with coupled mode at 2933 MT/s and uncoupled mode at 3200 MT/s to determine which is best for your workload.

In the BIOS, set your memory frequency to the desired speed and make sure APBDIS is set to 1 and fixed SOC Pstate is set to P0.

4.2.7 Preferred IO

Preferred IO allows one PCIe device in the system to be configured in a preferred state. This device gets preferential treatment on the infinity fabric. This is typically enabled for fabric adapters that provide the interconnect between systems.

See section 6.4 on how to enable Preferred IO

4.2.8 Determinism Slider

This option exists due to the varying characteristics of the different pieces of silicon in a CPU that exist naturally due to the manufacturing process. There is always some variation from part to part even under the same SKU. The determinism slider has 2 options: Performance and Power.

- **Performance**: In this mode, the CPUs in the system perform at the minimum performance requirement in order to meet the capability of the part as specified. Select this option if you must have all nodes in a system deliver identical performance.
- **Power**: In this mode, the CPUs in the system perform at the maximum capability of the each silicon device in the SoC. Due to the natural variations that exist during the manufacturing process, some CPUs and/or nodes may be capable of increased performance over others. Performance of a device will never fall below what you would get in Performance determinism mode, but additional performance can be gained.

Chapter 5 Linux Tuning Options

5.1 Linux Kernel Versions

It is recommended to use the latest Linux kernel. The AMD Linux team is continually pushing performance updates and fixes to the Linux kernel, and running the latest kernel ensures that the latest fixes and updates are in place.

5.2 Useful Commands

Command	Description
`lscpu`	Prints out useful information about CPU and its configuration.
`hwloc-ls`	Prints out useful information about the NUMA locality of devices and general hardware locality information.
`cat /sys/devices/system/cpu/cpufreq/boost`	Print out if CPU boost is on or off.
`cpupower frequency-info`	Prints out useful information about CPU governor.

5.3 General OS Tuning

This section includes recommended OS tuning options for Linux. This is not mean to be an all-inclusive list. Please evaluate your environment and workload(s) to find the options that work best for you.

5.3.1 Turn off swap to prevent any accidental swapping

It is recommended to disable swap to prevent any unwanted swap usage. If you need to use swap you need to buy more memory capacity for your nodes.

Ensure your node have sufficient memory to work your workloads. Disabling swap without sufficient memory can have undesired effects.

```
swapoff -a
```

5.3.2 Turn off NUMA balancing

NUMA balancing can have undesired effects, and since it is possible to bind the ranks and memory in HPC, this setting is not needed.

```
echo 0 > /proc/sys/kernel/numa_balancing
```

5.3.3 Disable ASLR

ASLR (Address Space Layout Randomization) is a security feature used to prevent the exploitation of memory vulnerabilities.

This can have a slight performance hit and it is recommended to turn it off.

```
echo 0 > /proc/sys/kernel/randomize_va_space
```

5.3.4 Set CPU governor to performance and disable cc6

Setting the CPU performance to governor to performance ensures max performance at all time. Disabling cc6 ensures that deeper CPU sleep states are not entered.

```
cpupower frequency-set -g performance

cpupower idle-set -d 2
```

5.4 Tuning Before every run

5.4.1 Drop all caches

Use the following command to drop all caches from previous runs and to start clean with new runs. This helps ensure consistent performance from run to run.

```
sync; echo 3 > /proc/sys/vm/drop_caches
```

Chapter 6 Mellanox HCA Information

6.1 Make sure latest OFED is installed

Ensure the latest version of OFED is installed

6.2 Updating Mellanox HCA FW

How to Update Mellanox HCA FW

6.3 How to determine MLX id and NUMA node of HCA

First ensure the latest version of OFED is installed. Then run the following commands as root

```
mst start
mst status -v
```

You will see information printed like below.

For the device you use to use the string circled in blue is your MLX id and the NUMA node circled in green is the NUMA node locality of that device.

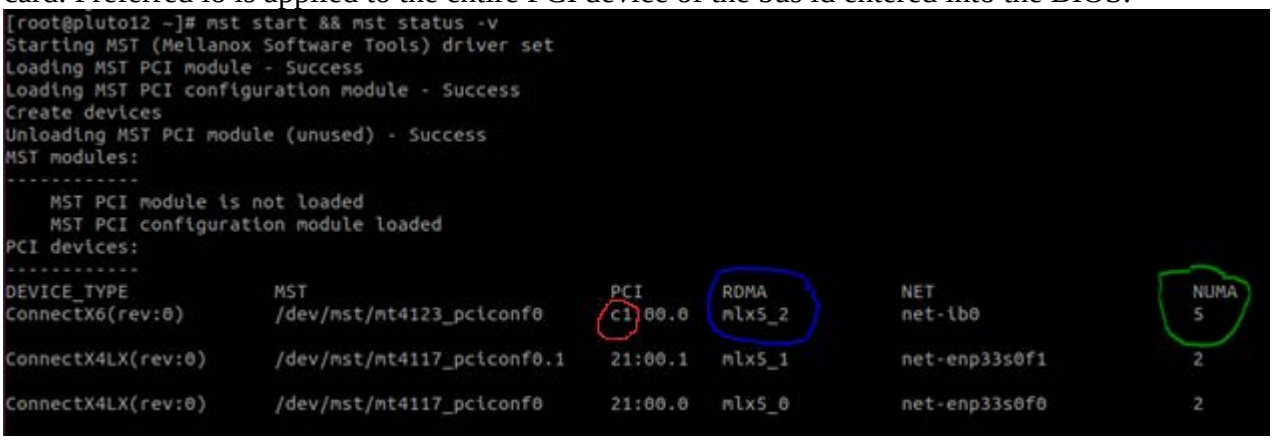
As an example, for the Connect X-6 device in the system below. The MLX id is mlx5_2 and the device is attached to NUMA node 5.

```
[root@pluto12 ~]# mst start && mst status -v
Starting MST (Mellanox Software Tools) driver set
Loading MST PCI module - Success
Loading MST PCI configuration module - Success
Create devices
Unloading MST PCI module (unused) - Success
MST modules:
------------
    MST PCI module is not loaded
    MST PCI configuration module loaded
PCI devices:
------------
DEVICE_TYPE             MST                           PCI       RDMA      NET                NUMA
ConnectX6(rev:0)        /dev/mst/mt4123_pciconf0      c1:00.0   mlx5_2    net-ib0            5

ConnectX4LX(rev:0)      /dev/mst/mt4117_pciconf0.1    21:00.1   mlx5_1    net-enp33s0f1      2

ConnectX4LX(rev:0)      /dev/mst/mt4117_pciconf0      21:00.0   mlx5_0    net-enp33s0f0      2
```

6.4 How to determine bus id for preferred io mode

In the same output you can also determine the bus id for enabling preferred io mode in the BIOS. The bus id to enter into the bios is circled in red. In this instance the bus id to enter into the BIOS

would be c1. Note: Due to the nature of how preferred io mode works it will be enabled on the full card on cards with more than 1 port. There is no way to enable preferred io on only 1 port of a card. Preferred io is applied to the entire PCI device of the bus id entered into the BIOS.



Chapter 7 Application Level Tuning

AMD Optimized Compilers and Libraries

Application level tuning is critical to achieving max performance on your workload. Please explore using AMD compilers and libraries to take advantage of optimizations our engineers have made. Please select your MPI for more details.

- **AOCC (AMD Compilers)** ***https://developer.amd.com/amd-aocc/***
- **AOCL (AMD Libraries)** ***https://developer.amd.com/amd-aocl/***

7.1 PLATFORM MPI

This section covers some of the common MPI tuning options for PlatformMPI. This page is not meant to be an all-inclusive list of necessary/recommended MPI options. Every workload should be tuned independently to achieve maximum performance.

Below is an example list of MPI options. Each option is detailed down below.

Platform MPI Options Example

```
-cpu_bind=rank,v -e MPI_FLUSH_FCACHE=1,full,v -prot -v -netaddr rank:11.11.11.0/24
```

7.1.1.1 Rank Binding

This option binds the ranks to cores. The v allows the binding to be printed out. It is highly recommended to review the pinning output to ensure it is pinning in the most efficient way possible.

PlatformMPI CPU Binding

```
-cpu_bind=rank,v
```

7.1.1.2 Flushing Caches

This option flushes various caches before each run.

PlatformMPI Flush all caches before run

```
-e MPI_FLUSH_FCACHE=1, full,v
```

Communication and Interconnect details

This option prints the communication method being used between different ranks. It can be reviewed to ensure the communication methods used are most optimal.

Debug information print transports

```
-prot -v
```

7.1.1.3 Interconnect / Network selection

This options selects a network for communication. Adjust the network to match your interconnect network.

PlatformMPI Select Network

```
-netaddr rank:11.11.11.0/24
```

7.2 OPENMPI

This purpose of this section is to show some of the common MPI tuning options for OpenMPI. This page is not meant to be an all inclusive list of necessary/recommended MPI options. Every workload should be tuned independently to achieve maximum performance.

Below is an example list of MPI options. There is an example for both openib and UCX.

Each option is detailed down below.

7.2.1 OpenMPI openib Options

--map-by core --report-bindings --mca btl_openib_allow_ib true -mca btl_openib_if_include mlx5_2:1 --mca pml ob1 --mca btl openib,self,vader

7.2.2 OpenMPI UCX Options

--map-by core --report-bindings --mca pml ucx --mca osc ucx --mca coll_hcoll_enable 1 -x UCX_NET_DEVICES=mlx5_2:1 -x HCOLL_MAIN_IB=mlx5_2:1

Selecting between UCX and OpenIB

There may be performance differences between UCX and OpenIB. It is recommended to test both and see which one performs better.

Explanation of Options

7.2.2.1 OpenMPI CPU Binding Options

This binding option is used in PureMPI scenarios where 1 rank is bound to 1 core. The report binding option prints binding information for each rank. More information about various map-by options can be found *HERE*.

```
--map-by core --report-bindings
```

7.2.2.2 Transport Selection openib

Use these options if you want to use openib.
replace mlx5_2 with the appropriate mlx* for your device. *Refer to section 6.3*

```
--mca btl_openib_allow_ib true -mca btl_openib_if_include mlx5_2:1 --mca pml ob1 --mca btl
openib,self,vader
```

7.2.2.3 Transport Selection UCX

Use these options if you want to use ucx.
replace mlx5_2 with the appropriate mlx* for your device. *Refer to section 6.3*

```
--mca pml ucx --mca osc ucx --mca coll_hcoll_enable 1  -x UCX_NET_DEVICES=mlx5_2:1 -x
HCOLL_MAIN_IB=mlx5_2:1
```

Hardware collective offload on UCX

Hardware collective offload can be built with OpenMPI or comes with Mellanox's HPCX. Some workloads benefit from this and others dont. Please test for your particular workload. This can be disabled by setting --mca coll_hcoll_enable to 0

7.3 INTELMPI

This purpose of this page is to show some of the common MPI tuning options for OpenMPI. This page is not meant to be an all-inclusive list of necessary/recommended MPI options. Every workload should be tuned independently to achieve maximum performance.

Below is an example list of MPI options. Each option is detailed down below.

```
-genv I_MPI_DEBUG=5 -genv I_MPI_PIN=1 -genv KMP_AFFINITY verbose,granularity=fine,compact
```

7.3.1 Explanation of Options

7.3.2 Print Debug Information

This option enables debug output to print transport and pinning information.

```
-genv I_MPI_DEBUG=5
```

7.3.3 Enable rank pinning

This option enables rank pinning. Use in conjunction with the previous option.

```
-genv I_MPI_PIN=1
```

7.3.4 Pinning methodology

This option sets the pinning methodology. *Please see the Intel MPI docs for more information.*

```
-genv KMP_AFFINITY verbose,granularity=fine,compact
```