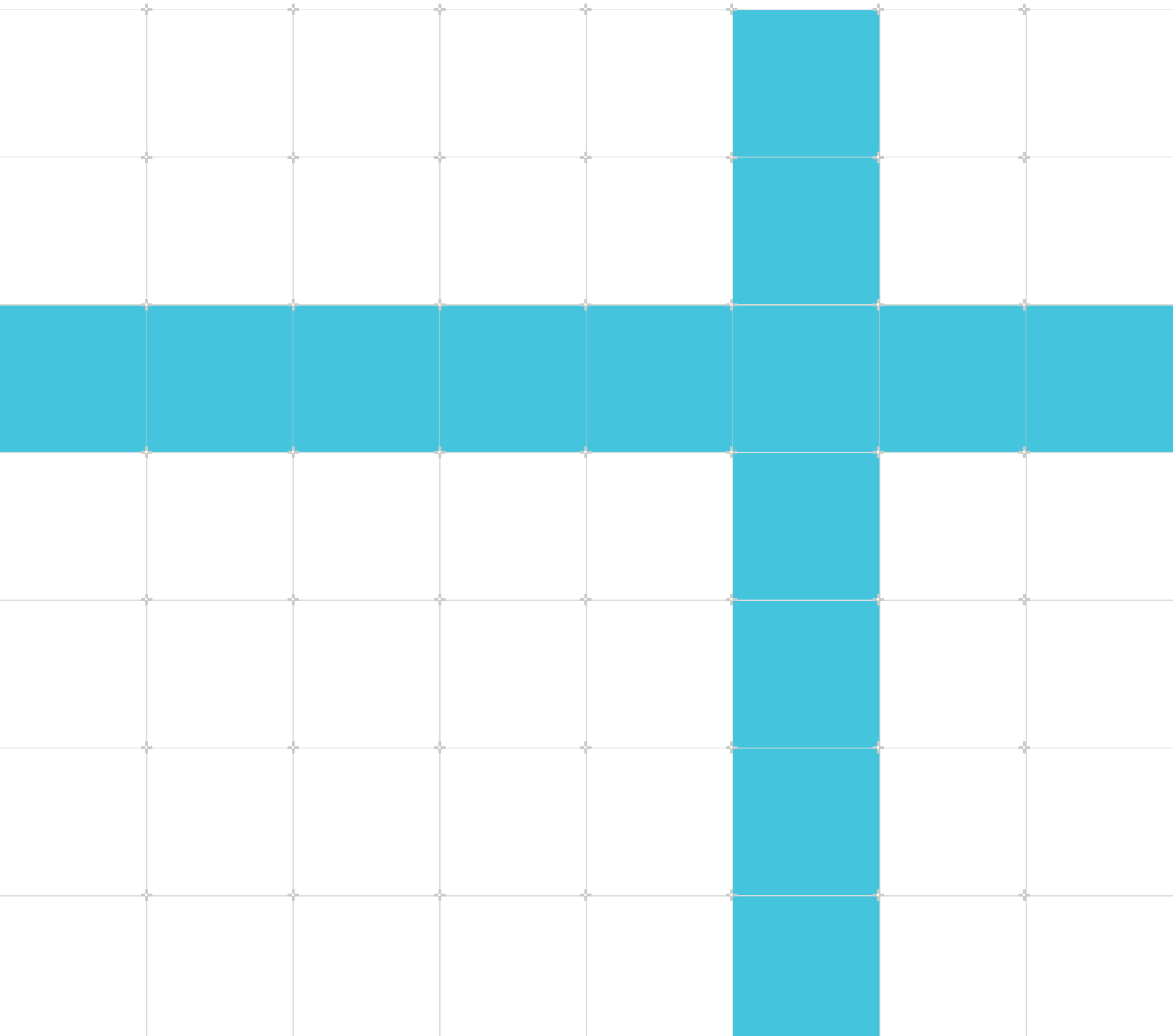
arm

DPDK Tuning Guide

Version 1.0

Non-Confidential
Copyright © 2024 Arm Limited (or its affiliates).
All rights reserved.

Issue
109701_1.0_en



DPDK Tuning Guide

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.

Release information

Document history

Issue	Date	Confidentiality	Change
1.01	29 May 2024	Non-Confidential	Minor update
1.0	10 March 2024	Non-Confidential	Initial release

Proprietary Notice

This document is protected by copyright and other related rights and the use or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm Limited ("Arm"). No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm

makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, any patents, copyrights, trade secrets, trademarks, or other rights.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted use, duplication, or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. Please follow Arm's trademark usage guidelines at https://www.arm.com/company/policies/trademarks. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Contents

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

1. Introduction

DPDK is the Data Plane Development Kit DPDK Project Home that consists of libraries to transmit packets directly to/from applications. These libraries run on a wide variety of CPU architectures and accelerate packet processing workloads by allowing incoming network packets transmitted to user space with no overhead for memory copying.

Though DPDK uses several techniques to optimize packet throughput, how it works (and the key to its performance) is based on bypassing kernel and Poll Mode Driver (PMD).

- Kernel bypass: Create a path from the NIC to the application within user space, in other words, bypass the kernel. This eliminates context switching when moving the frame between user space and kernel space. Additionally, further gains are obtained by negating the NIC driver, kernel network stack, and performance penalties they introduce.
- Poll Mode Driver (PMD): Speed up the packet pipeline by allocating a CPU core to constantly poll for new packets instead of relying on asynchronous, interrupt-based signaling mechanisms.

The guide will focus on the configuration about platform and I/O to boost throughput for DPDK, including system check, NUMA setting, BIOS setting, Linux configuration, NIC setting, compiling setting, and tuning parameters for L3fwd. Finally, the guide will talk about one example of multi-core scenarios. The guide is intended for engineers implementing applications with DPDK to achieve optimal performance on Arm platforms.

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

2. Quick Start Guide

A system health check is always the first thing to do before any testing. It helps discover those issues in the system which might slow down the performance. Use common utilities like dmidecode, lscpu, lshw, lspci, etc. to check the BIOS version, CPU frequency, memory DIMM population, ethernet link speed, etc. Make sure all the components of the system are running in a functional and performing manner.

System Health Check

A system health check is always the first thing to do before any testing. It helps discover those issues in the system which might slow down the performance. Use common utilities like dmidecode, lscpu, lshw, lspci, etc. to check the BIOS version, CPU frequency, memory DIMM population, ethernet link speed, etc. Make sure all the components of the system are running in a functional and performing manner.

NUMA setting

NUMA balancing

Automatic NUMA balancing is enabled when both of the following conditions are met:

- `numactl --hardware` shows multiple nodes.
- `cat /proc/sys/kernel/numa_balancing` shows 1.

Manual NUMA tuning of applications will override automatic NUMA balancing. In some cases, system-wide manual NUMA tuning is preferred.

To disable automatic NUMA balancing, use the following command: `echo 0 > /proc/sys/kernel/numa_balancing`

To enable automatic NUMA balancing, use the following command: `echo 1 > /proc/sys/kernel/numa_balancing`

On some Neoverse-N2 platforms with two NUMA nodes, the performance for l3fwd is better when enabling automatic NUMA balancing.

NUMA Affinity

Use the following command to check the PCI device related NUMA node id and ensure the running core and the PCI device are on the same NUMA node. `cat /sys/bus/pci/devices/0000\ :xx\ :00.x/numa_node`

Check the memory for each NUMA node and ensure the memory on the expected node.

```
numastat -m
```

Tools such as the Linux lstopo command can also help determine the connectivity between PCI devices and sockets by displaying how the OS sees NUMA nodes, caches, cores, and PCI devices.

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

BIOS Performance Setting

BIOS settings are dependent on the platform you are using. Here is a list of some generic options for Arm platforms recommended for DPDK. For others, you must investigate more.

Write allocation to SLC

Write allocation to system level cache (SLC) allows the PCIe device to write to SLC directly instead of writing to memory.

For Ex: for Ampere Altra, use `lspci` to find the PCI bridge of the NIC. Then write `78007800/00000000` to the offset PCI address 8e8 to enable/disable cache stashing.

```
$ lscpi

0000:00:00.0 Host bridge: Ampere Computing, LLC Device e100 0000:00:01.0 PCI bridge:
Ampere Computing, LLC Device e101 (rev 04) # This is the PCI bridge we're looking for
0000:01:00.0 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE
QSFP+ (rev 02) 0000:01:00.1 Ethernet controller: Intel Corporation Ethernet Controller
XL710 for 40GbE QSFP+ (rev 02)

# tips: setpci needs sudo privilege $ setpci -s 0000:00:01.0 8e8.l=78007800 # enable
cache stashing $ setpci -s 0000:00:01.0 8e8.l=00000000 # disable cache stashing
```

Power Management

Enable ACPI and CPPC at the same time to run the power management feature of DPDK. Otherwise, suggest disabling the feature as it may result in performance degradation.

```
Advanced->ACPI Settings->Enable ACPI Auto Configuration [Disabled] Advanced->ACPI
Settings->Enable CPPC [Disabled]
```

Overwrite the following parameters in `GRUB_CMDLINE_LINUX` in `/etc/default/grub to disable CPPC`, if there is not above setting in BIOS.

```
cpufreq.off = 1 cpuidle.off = 1
```

PCIe Relaxed Ordering

Relaxed PCIe ordering helps maximum throughput performance, especially with high-speed PCIe 4.0 NICs when targeting memory.

- Use NIC vendor-provided communication libraries that enable relaxed ordering by a new API. This is the preferred option. Mellanox NIC can be configured via `mlxconfig -d <PCIe Address>` set `PCI_WR_ORDERING=1`.
- Force all traffic to use relaxed ordering. This can break some optimized communications that rely on memory being written and visible in a given order in memory.

Some vendor specific settings also influence the performance, below lists some examples from Ampere Altra processors. Contact different vendors for different platforms for more details.

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Advanced->ACPI Settings->Enable LPI [Disabled] Chipset->CPU Configuration->ANC mode [Monolithic] Chipset->CPU Configuration-> SLC Replacement Policy [Enhanced Least Recently Used] Chipset->CPU Configuration->L1/L2 Prefetch [Enabled] Chipset->CPU Configuration->SLC as L3$ [Disabled]

Other Settings for Ampere

Some vendor specific settings also influence the performance, below lists some examples from Ampere Altra processors. Contact different vendors for different platforms for more details.

```
Advanced->ACPI Settings->Enable LPI [Disabled]

Chipset->CPU Configuration->ANC mode [Monolithic]

Chipset->CPU Configuration-> SLC Replacement Policy [Enhanced Least Recently Used]

Chipset->CPU Configuration->L1/L2 Prefetch [Enabled]

Chipset->CPU Configuration->SLC as L3$ [Disabled]
```

Linux OS Configuration

This part provides the commandline to set Linux parameters. Overwrite them in GRUB_CMDLINE_LINUX in /etc/default/grub.

Hugepage Support

Larger hugepages can cover larger memory areas without TLB misses and are recommended. For example, if the hugepage size is 1G and there are 32 hugepages, add the following to the kernel parameter list:

```
hugepagesz=1G hugepages=32
```

Verify 32*1GB hugepages are evenly distributed across all NUMA nodes.

```
cat /sys/devices/system/node/node$N/hugepages/hugepages-1048576kB/nr_hugepages cat /
sys/devices/system/node/node*/meminfo
```

Core Isolation

While the threads used by a DPDK application are pinned to logical cores on the system, it is possible for the Linux scheduler to run other tasks on those cores. To help prevent additional workloads, timers, RCU processing and IRQs from running on those cores, it is possible to use the Linux kernel parameters `isolcpus, nohz_full, irqaffinity` to isolate them from the general Linux scheduler tasks to reduce context switch.

For example, if a platform has 0-7 cores and DPDK applications are to run on logical cores 2,4 and 6, add the following to the kernel parameter list:

```
isolcpus=2,4,6 nohz_full=2,4,6 irqaffinity=0,1,3,5,7
```

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

IOMMU

`iommu.passthrough=1` bypasses the IOMMU translation for DMA. This will expose the platform's RAM to PCIe devices and reduce extra memory cost by IOMMU.

Please refer to the kernel document for more details of these options.

NIC setting

Different NIC may have different optimal settings, please refer to the user manual of the device for details.

Compiler Version

The Arm compiler team has done a lot of optimizations focused on Arm platforms and support better for the newer GCC version. That is why we recommend using the newest version, which can give better performance. On some Neoverse-N2 platforms, the l3fwd performance compiled by GCC12 is better than GCC11.

Compiler Options

DPDK has a few options that can be adjusted as part of the build configuration process. The options can be listed by running meson configure inside a configured build folder. The "platform" option specifies a set of configuration parameters that will be used, such as generic, native, `<SoC>`.

The "WFE" option is configured in `config/arm/meson.build` (flags_common). SoCs can build with this option enabled/disabled, and the ability is available for either `native`, `generic`, or `cross build`.

For example, run the following command to configure `aarch32` DPDK on an `aarch64` platform.

```
meson cross-build -Dplatform=generic_aarch32
```

Tuning Parameters for L3fwd

Refer to these performance reports from Broadcom, NVIDIA, and Intel, three parameters are important to been tuned for l3fwd according to different NICs and platforms. Modify them in examples/l3fwd/l3fwd.h.

- RX_DESC_DEFAULT
- TX_DESC_DEFAULT
- MAX_PKT_BURST

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

3. User Guide

A system health check is always the first thing to do before any testing. It helps discover those issues in the system which might slow down the performance. Use common utilities like `dmidecode`, `lscpu`, `lshw`, `lspci`, etc. to check the BIOS version, CPU frequency, memory DIMM population, ethernet link speed, etc. Make sure all the components of the system are running in a functional and performing manner.

System Health Check

A system health check is always the first thing to do before any testing. It helps discover those issues in the system which might slow down the performance. Use common utilities like `dmidecode`, `lscpu`, `lshw`, `lspci`, etc. to check the BIOS version, CPU frequency, memory DIMM population, ethernet link speed, etc. Make sure all the components of the system are running in a functional and performing manner.

`dmicode` is a tool for dumping a platform's DMI (also SMBIOS) table contents in a human-readable format. This table contains a description of the system's hardware components, as well as other useful pieces of information such as serial numbers and BIOS revision.

`lscpu` gathers CPU architecture information from sysfs and /proc/cpuinfo. The information includes, for example, the number of CPUs, CPU frequency, threads, cores, sockets, and Non-Uniform Memory Access (NUMA) nodes. Make sure that all the showed information matches the expected content.

`lshw` is a small tool to extract detailed information on the hardware configuration of the machine. It can report exact memory configuration, firmware version, mainboard configuration, CPU version and speed, cache configuration, bus speed, etc.

NUMA Setting

Automatic NUMA Balancing

Automatic NUMA balancing improves the performance of applications running on NUMA hardware systems. An application will generally perform the best when the threads of its processes are accessing memory on the same NUMA node as the threads are scheduled. Automatic NUMA balancing moves tasks (which can be threads or processes) closer to the memory they are accessing. It also moves application data to the memory closer to the tasks that reference it. This is all done automatically by the kernel when automatic NUMA balancing is active.

Automatic NUMA balancing is enabled when both of the following conditions are met:

- `numactl` –hardware shows multiple nodes.
- `cat /proc/sys/kernel/numa_balancing` shows 1.

Manual NUMA tuning of applications will override automatic NUMA balancing. In some cases, system-wide manual NUMA tuning is preferred.

To disable automatic NUMA balancing, use the following command:

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

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

To enable automatic NUMA balancing, use the following command:

```
echo 1 > /proc/sys/kernel/numa_balancing
```

On some Neoverse-N2 platforms with two NUMA nodes, the performance for l3fwd is better when enabling automatic NUMA balancing.

NUMA Affinity

I/O intensive workloads can perform better when placed on the same NUMA node that connects to the I/O device used. Use the following command to check the PCI device related NUMA node id and ensure the running core and the PCI device are on the same NUMA node.

```
cat /sys/bus/pci/devices/0000\ :xx\ :00.x/numa_node
```

Check the memory for each NUMA node and ensure the memory on the expected node.

```
numastat -m
```

Tools such as the Linux lstopo command can also help determine the connectivity between PCI devices and sockets by displaying how the OS sees NUMA nodes, caches, cores, and PCI devices.

BIOS Performance Setting

BIOS settings are dependent on the platform you are using. Here is a list of some generic options for Arm platforms recommended for DPDK. For others, you must investigate more.

Write allocation to SLC

Write allocation to system level cache (SLC) allows the PCIe device to write to SLC directly instead of writing to memory. Depending on the use cases, it can improve the throughput and latency of the applications. One should approach the platform provider to understand how to enable/disable this feature.

For Ex: for Ampere Altra, use `lspci` to find the PCI bridge of the NIC. Then write `78007800/00000000` to the offset PCI address 8e8 to enable/disable cache stashing.

```
$ lscpi

0000:00:00.0 Host bridge: Ampere Computing, LLC Device e100 0000:00:01.0 PCI bridge:
Ampere Computing, LLC Device e101 (rev 04) # This is the PCI bridge we're looking for
0000:01:00.0 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE
QSFP+ (rev 02) 0000:01:00.1 Ethernet controller: Intel Corporation Ethernet Controller
XL710 for 40GbE QSFP+ (rev 02)

# tips: setpci needs sudo privilege $ setpci -s 0000:00:01.0 8e8.l=78007800 # enable
cache stashing $ setpci -s 0000:00:01.0 8e8.l=00000000 # disable cache stashing
```

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Power Management

Advanced Configuration and Power Interface (ACPI) has been supported by Armv8 and higher. Collaborative Processor Performance Control (CPPC) is an ACPI table provided by BIOS which is used for scaling CPU frequency.

Enable ACPI and CPPC at the same time to run the power management feature of DPDK. Otherwise, suggest disabling the feature as it may result in performance degradation.

```
Advanced->ACPI Settings->Enable ACPI Auto Configuration [Disabled] Advanced->ACPI
Settings->Enable CPPC [Disabled]
```

Overwrite the following parameters in GRUB_CMDLINE_LINUX in /etc/default/grub to disable CPPC, if there is not above setting in BIOS.

```
cpufreq.off = 1 cpuidle.off = 1
```

PCIe

In PCIe protocol there are some transaction ordering rules to avoid deadlocks and producer-consumer problems. PCIe specification provides a way to change the ordering rules by using Relaxed Ordering (RO), ID-Based Ordering (IDO) attributes present in the TLP.

Relaxed PCIe ordering helps maximum throughput performance, especially with high-speed PCIe 4.0 NICs when targeting memory.

- Use NIC vendor-provided communication libraries that enable relaxed ordering by a new API. This is the preferred option. Mellanox NIC can be configured via mlxconfig -d `<PCIe Address>` set PCI_WR_ORDERING=1.
- Force all traffic to use relaxed ordering. This can break some optimized communications that rely on memory being written and visible in a given order in memory.

Other Settings

Some vendor specific settings also influence the performance, below lists some examples from Ampere Altra processors. Contact different vendors for different platforms for more details.
`Advanced->ACPI Settings->Enable LPI [Disabled] Chipset->CPU Configuration->ANC mode [Monolithic] Chipset->CPU Configuration-> SLC Replacement Policy [Enhanced Least Recently Used] Chipset->CPU Configuration->L1/L2 Prefetch [Enabled] Chipset->CPU Configuration->SLC as L3$ [Disabled]`

Linux OS Configuration

This part provides the commandline to set Linux parameters. Overwrite them in `GRUB_CMDLINE_LINUX` in `/etc/default/grub`.

Hugepage Support

The supported hugepage size is different with the base kernel page size config. If the kernel page size is configured to 4K, the supported hugepage sizes are 64K, 2M, 32M, and 1G (1G hugepage is

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

recommended). If the kernel page size is configured to 64K, the supported hugepage sizes are 2M, 512M, and 16G (512M Sor 16G hugepage is recommended). Larger hugepages can cover larger memory areas without TLB misses.

For example, if the hugepage size is 1G and there are 32 hugepages, add the following to the kernel parameter list:

```
hugepagesz=1G hugepages=32
```

Verify 32*1GB hugepages are evenly distributed across all NUMA nodes.

```
cat /sys/devices/system/node/node$N/hugepages/hugepages-1048576kB/nr_hugepages cat /
sys/devices/system/node/node*/meminfo
```

Core Isolation

While the threads used by a DPDK application are pinned to logical cores on the system, it is possible for the Linux scheduler to run other tasks on those cores. To help prevent additional workloads, timers, RCU processing and IRQs from running on those cores, it is possible to use the Linux kernel parameters `isolcpus`, `nohz_full`, `irqaffinity` to isolate them from the general Linux scheduler tasks to reduce context switch.

`isolcpus` partitions the CPUs between isolated tasks and the rest of the system.

`nohz_full` stops the ticks for the specified CPUs while running a single task. Most kernel unbound load is migrated and most RCU processing is offloaded to the CPUs outside the isolated range.

The CPUs set as nohz_full will run in NOCB (no callback) mode, which means the RCU callbacks queued on these CPUs are executed from unbound kthreads running on non-isolated CPUs.

`irqaffinity` fires hardware interruption to the non-isolated CPUs to avoid disturbing the exclusive load on isolated CPUs.

For example, if a platform has 0-7 cores and DPDK applications are to run on logical cores 2,4 and 6, add the following to the kernel parameter list:

```
isolcpus=2,4,6 nohz_full=2,4,6 irqaffinity=0,1,3,5,7
```

IOMMU

`iommu.passthrough=1` bypasses the IOMMU translation for DMA. This will expose the platform's RAM to PCIe devices and reduce extra memory cost by IOMMU.

Please refer to the kernel document for more details of these options.

NIC Setting

Use Mellanox NIC as an example in this part.

Mellanox NIC Setup

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Download the OFED driver and install it.

```
mount -o ro,loop MLNX_OFED_LINUX-5.4-3.1.0.0-ubuntu20.04-aarch64.iso /mnt
/mnt/mlnxofedinstall --upstream-libs -dpdk
```

Install Mellanox Firmware Tools (MFT).

```
wget https://www.mellanox.com/downloads/MFT/mft-4.20.0-34-arm64-deb.tgz
tar xvf mft-4.20.0-34-arm64-deb.tgz
cd mft-4.20.0-34-arm64-deb/
./install.sh
mst start
```

Install latest firmware for the NIC.

```
mlxfwmanager -online -u -d <device PCIe address>
Download a specific firmware version from Mellanox Firmware Downloads. To find the
 proper firmaware, first obtain its OPN and PSID via sudo mlxfwmanager -d <device
 PCIe address> --query. The output looks like:
Querying Mellanox devices firmware ...
Device #1:
----------
  Device Type:      ConnectX5
  Part Number:      MCX516A-CDA_Ax_Bx
  Description:      ConnectX-5 Ex EN network interface card; 100GbE dual-port
 QSFP28; PCIe4.0 x16; tall bracket; ROHS R6
  PSID:             MT_0000000013
  PCI Device Name:  0000:01:00.0
```

The PSID is listed in the output. The OPN will resemble the part number but may not match exactly. As an example, the OPN for the sample output is MCX516A-CDA. After downloading the desired firmware, flash that firmware to the NIC: `mlxfwmanager -d <device PCIe address> -i <path to unzipped FW download>`

Optimal Setting for the Mellanox NIC

Disable the pause frame of the ethernet interface for flow control. Flow control is intended to handle the situation where a transmitter is sending data faster than the receiver can deal with. The IEEE 802.3x standard specifies PAUSE on the above situation which may result in throughput loss.

```
ethtool -A <interface> rx off tx off
```

Set the MaxReadReq of the PCIe device to 4096 bytes. This is just an example for Mellanox NIC.

```
setpci -s <PCIe Address> 68.w=5fff
```

Enable relaxed ordering and CQE Compression of the Mellanox NIC:

```
mst start mst status mlxconfig -d <PCIe Address> query |grep -e PCI_WR_ORDERING -e
CQE_COMPRESSION mlxconfig -d <PCIe Address> set CQE_COMPRESSION=1 mlxconfig -d <PCIe
Address> set PCI_WR_ORDERING=1
```

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Different NIC may have different best settings, please refer to the user manual of the device for details.

Compiler Setting

Complier Version

DPDK has a few options that can be adjusted as part of the build configuration process. The options can be listed by running meson configure inside a configured build folder. Many of these options come from the "meson" tool itself and can be seen documented on the Meson Website.

The "platform" option specifies a set of configuration parameters that will be used.

`-Dplatform=generic` is for non-unique SoCs, it uses configuration that works on all machines of the same architecture as the build machine.

`-Dplatform=native` is for the unique SoC, it tailors the configuration to the build machine.

`-Dplatform=<SoC>` is for cross build, it uses configuration optimized for a particular SoC. Consult the "soc" dictionary in config/arm/meson.build to see which SoCs are supported. The "WFE" option is configured in config/arm/meson.build (flags_common). SoCs can build with this option enabled/disabled, and the ability is available for either `native`, `generic`, or `cross build`.

For example, run the following command to configure `aarch32` DPDK on an `aarch64` platform.

```
meson cross-build -Dplatform=generic_aarch32.
```

Tuning Parameters for L3fwd

The dpdk-l3fwd sample application demonstrates the use of the hash, LPM and FIB based lookup methods provided in DPDK to implement packet forwarding using poll mode or event mode PMDs for packet I/O. This part will introduce two examples about tuning l3fwd sample.

Tunable Parameters for DPDK

Topology

Refer to these performance reports from Broadcom, NVIDIA, and Intel, three parameters are important to been tuned for l3fwd according to different NICs and platforms. Modify them in examples/l3fwd/l3fwd.h.

- RX_DESC_DEFAULT
- TX_DESC_DEFAULT

The two parameters describe the queue length which are allocated equal-sized resources for Rx and Tx. They can be modified in.

- MAX_PKT_BURST

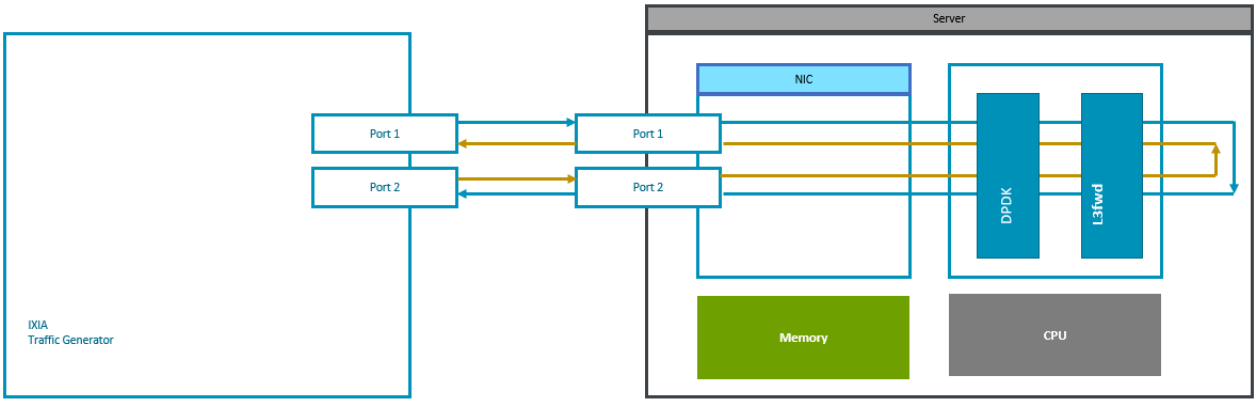
DPDK utilizes burst process to decompose transmitting and receiving into multiple stages, and deal with several packets at a time, usually 8, 16, or 32 described by MAX_PKT_BURST.

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

DUT: N1SDP (Neoverse-N1)

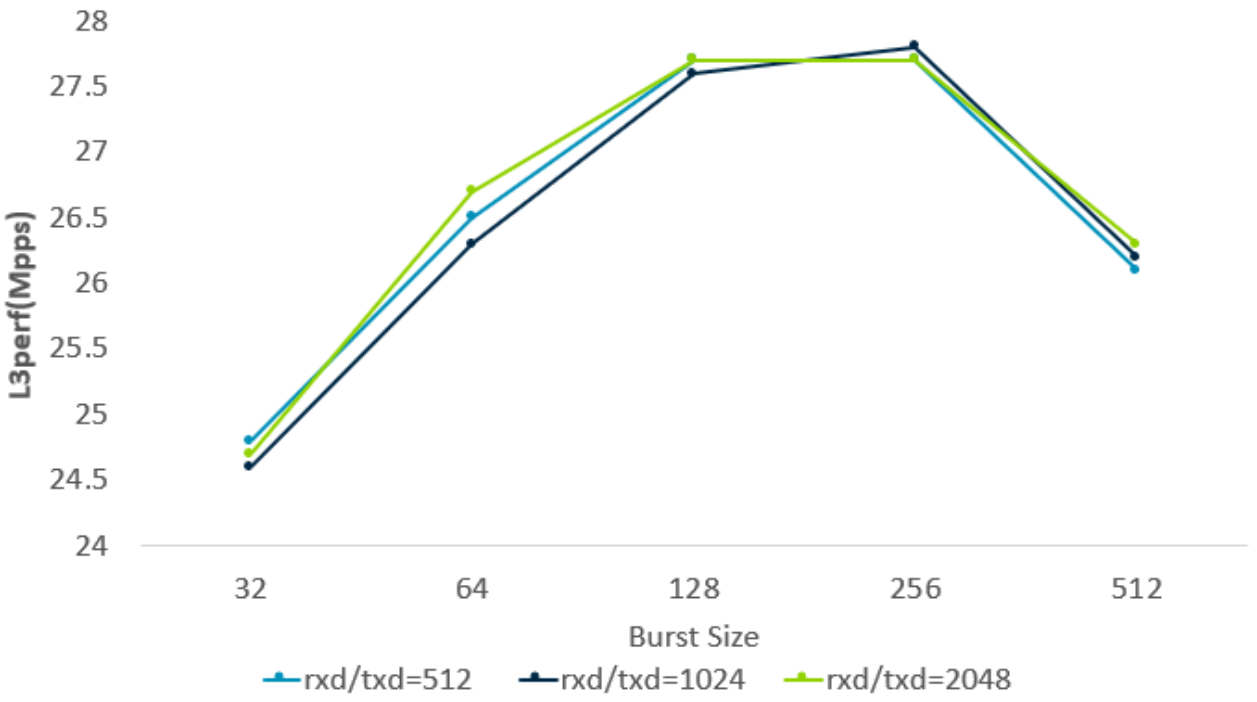
NIC: Mellanox ConnectX-5

Figure 3-1: DUT connected to one Mellanox NIC



As shown, the Device Under Test (DUT) are connected to one Mellanox NIC with two ports.

Results



The throughput changes with different Rx/Tx length and burst size for MLX5 NIC and N1SDP platform. It shows the best performance when setting as below. `#define RX_DESC_DEFAULT 1024 #define TX_DESC_DEFAULT 1024 #define MAX_PKT_BURST 256`

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

For RX/TX_DESC_DEFAULT, the shorter length for Rx/Tx means the NIC holds less packets, which may lead to packet loss in high traffic scenarios. In turn, the longer queue length holds more packets with more memory space, which may lead to resource cost.

For MAX_PKT_BURST, the burst mode processes adjacent data access and similar data calculation together. This mode minimizes memory access for multiple data read/write and make the maximize cache benefits. Therefore, it saves time for processing packets and achieves better performance.

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

4. Example for Multi-core scenario

This part shows the configuration from part-3 for multi-core scenario and hope users can get some inspiration.

Topology

The multi-core testing case is tested on the following platforms.

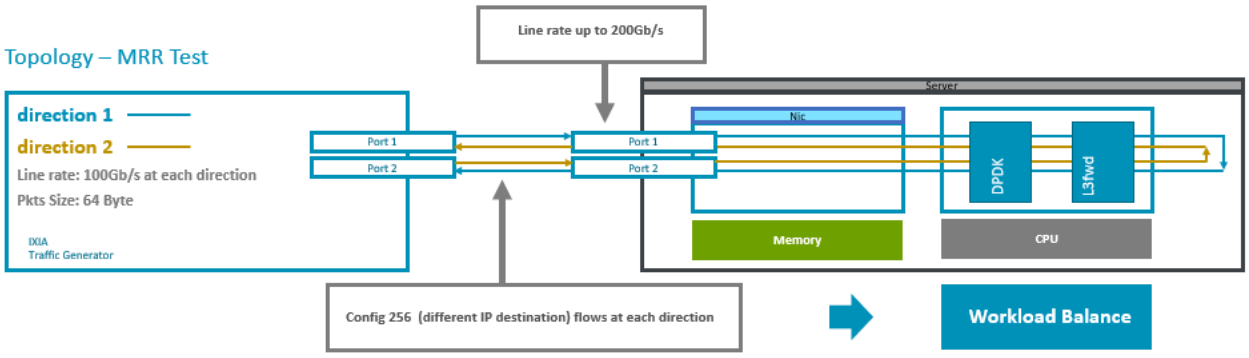
DUT: Ampere Altra (Neoverse-N1)

- Ubuntu 20.04.1
- Kernel 5.4.0-65-generic

NIC: Mellanox ConnectX-5 100G

- OFED driver: 5.7-1.0.2
- Firmware version: 16.33.1048

Figure 4-1: DUT connected to one NIC running MMR test



As shown, the Device Under Test (DUT) connects to one NIC and two ports and runs the MRR test. Enable 256 flows at each direction to force NIC to enable the efficient distribution of network receive processing across multiple CPUs with Receive Side Scaling (RSS) RSS.

NIC and DPDK Parameters Setting

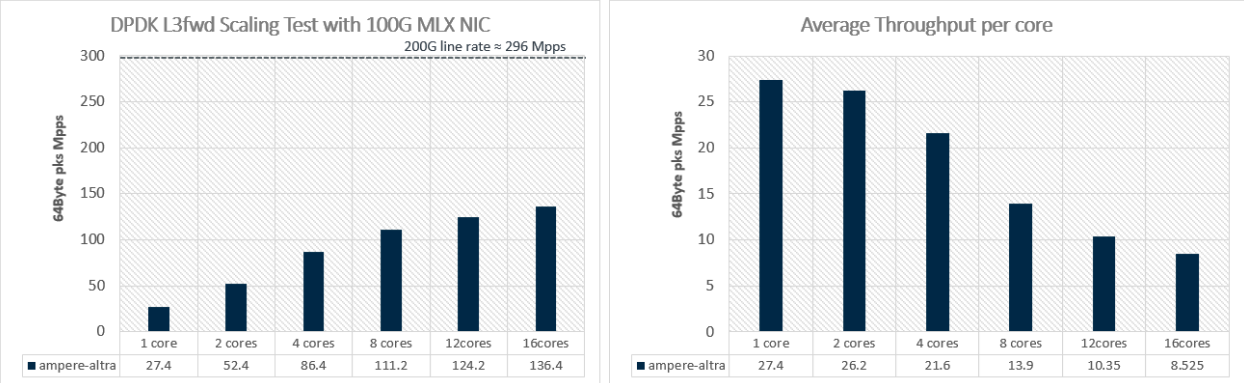
Configure Mellanox NIC according to 3.5-part.

The 3.7.1-part talks about `RX_DESC_DEFAULT`, `TX_DESC_DEFAULT`, and `MAX_PKT_BURST`. Configure these parameters as below in this case. `#define RX_DESC_DEFAULT 4096 #define TX_DESC_DEFAULT 4096 #define MAX_PKT_BURST 64`

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

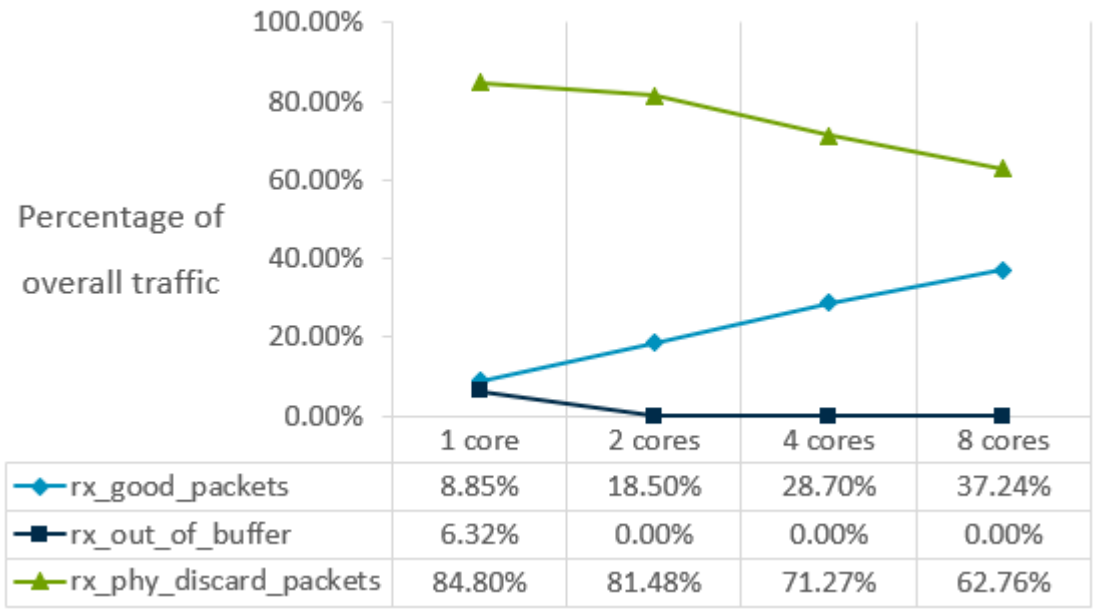
Results

Figure 4-2: L3fwd Scaling test



If the scalar performance increases linearly, the 16-core throughput should be 432 Mpps (27.7 * 16). As the upper bound traffic rate for 200G NIC is 296 Mpps, the 16-core throughput is 296 Mpps in theory. As the left figure shows, the scaling performance does not linearly increase with core numbers and there is a large gap between the 16-core throughput and the upper bound. Further analyze the performance from 1 core to 8 cores as below.

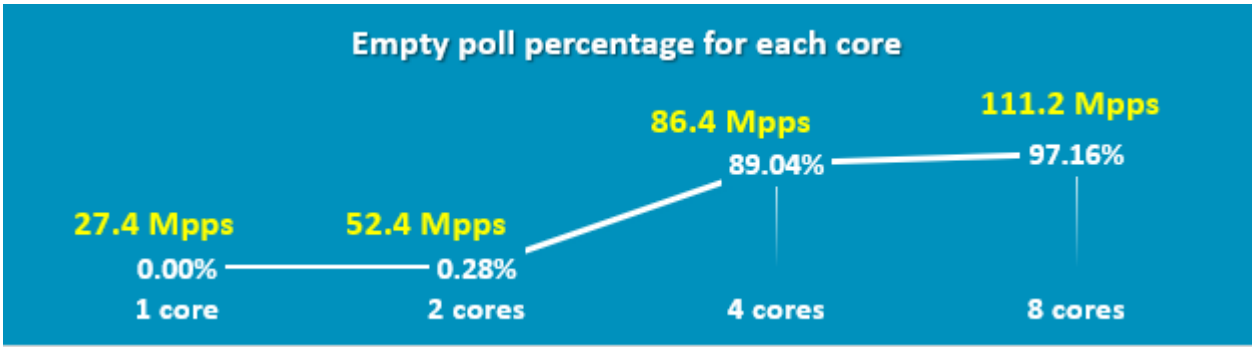
Figure 4-3: Core performance analysis



`Rx_good_packets` mean the packets received by DPDK successfully. `Rx_out_of_buffers` mean the packets dropped due to none allocated software buffers. `Rx_phy_discard_packets` mean the packets dropped due to lack of buffers on a physical port.

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 4-4: Empty poll percentage each core



DPDK would keep polling to request packets from memory when transmitting them. If there are no packets in the memory area, it results in one empty poll. The higher percentage for empty roll, the more idle the CPU would be.

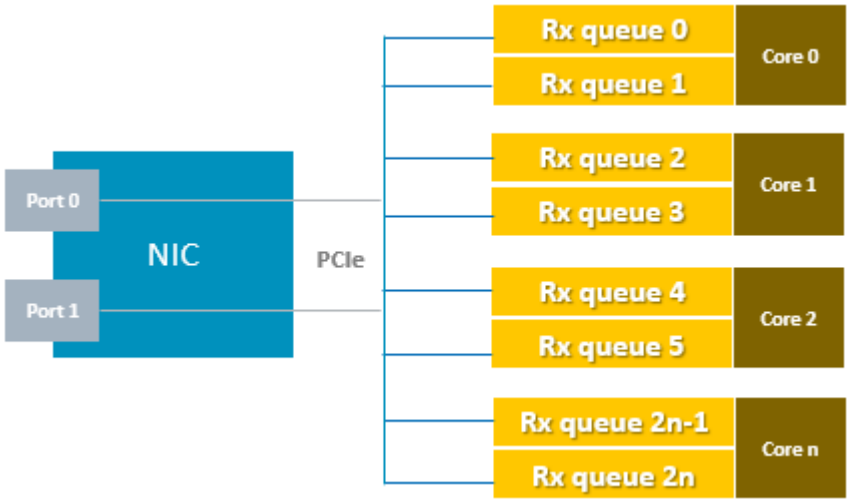
As the above figures show, for 8-core case, most packets are dropped on the NIC side and the CPU is idle, which shows the bottleneck is in hardware RX path.

Optimization

The following lists several suggestions for better scaling results for I/O.

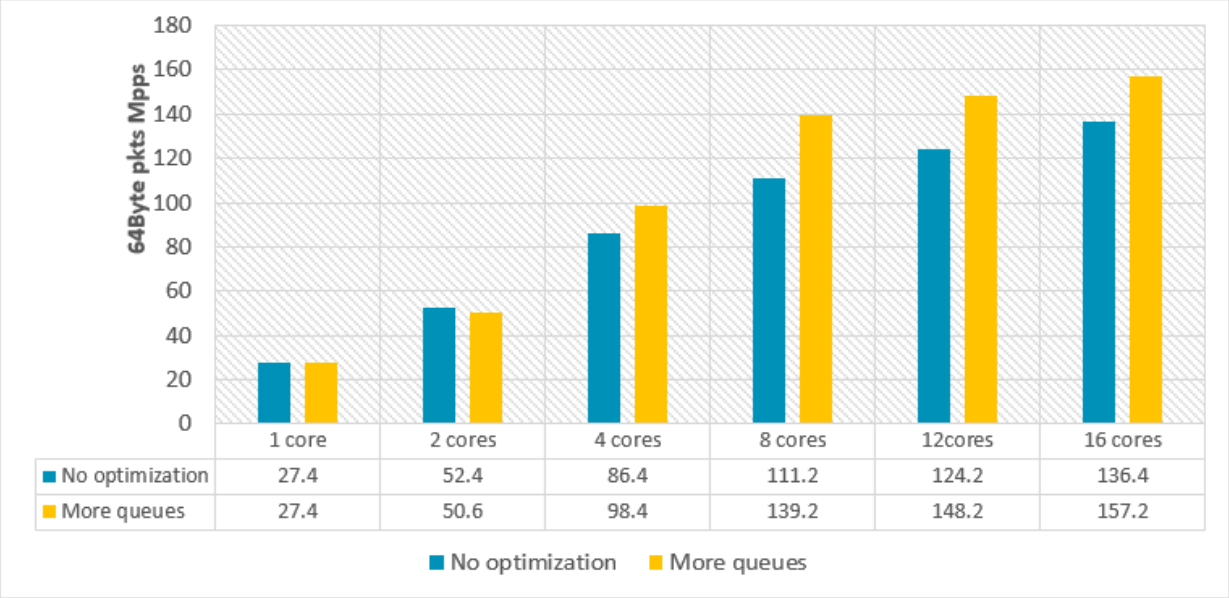
1. Increase the RX queue number for each core. Note that performance and the RX queue number do not grow linearly, so it is important to schedule the suitable number of configured queues for each core during the scaling test.

Figure 4-5: NIC test diagram four cores



Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

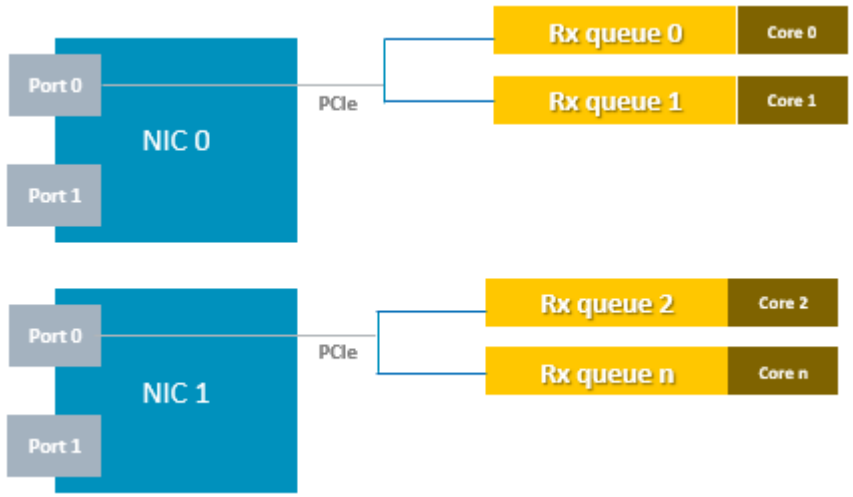
Figure 4-6: Performance comparison table for four cores



The performance increases almost 13.8% ~ 25% for more than four cores.

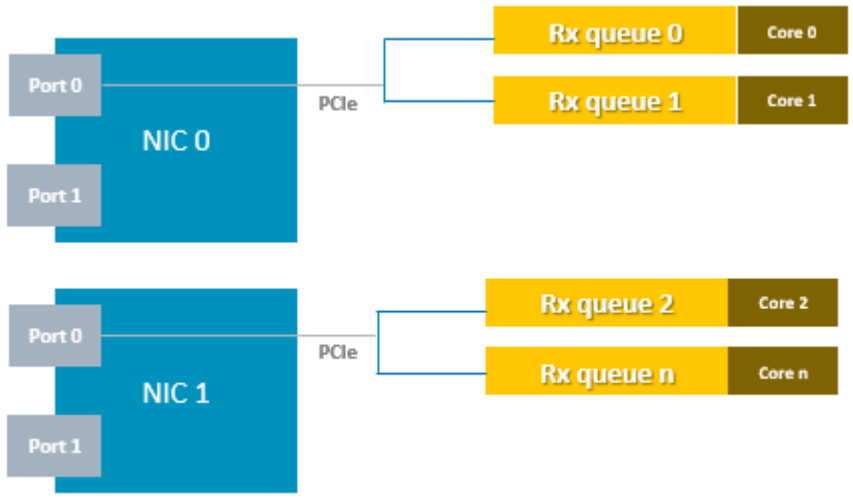
2. Use two NICs, and each NIC provides one port to increase the PCIe bandwidth and I/O ability.

Figure 4-7: PCIe bandwidth diagram



Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

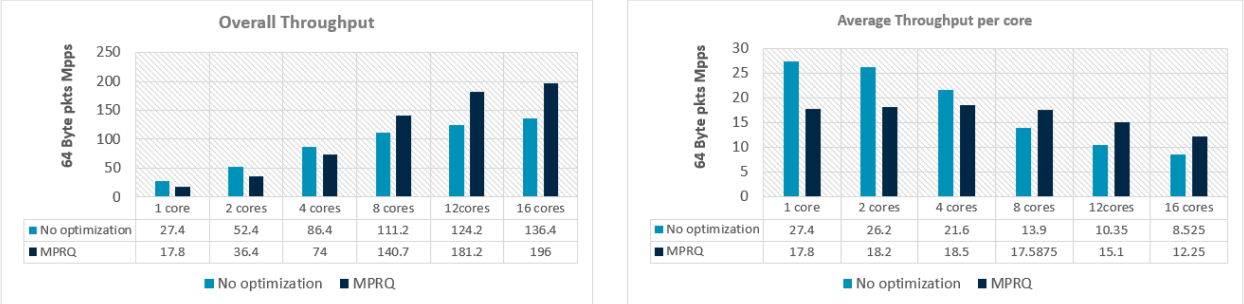
Figure 4-8: Performance comparison no optimization and PCIe



The performance increases almost 18.5% ~ 62% for more than four cores.

3. Multi-Packet Rx Queue (MPRQ) is as known as Striding RQ, it can further save PCIe bandwidth by posting single large buffer for multiple packets. Instead of posing a buffer per packet, one large buffer is posted to receive multiple packets on the buffer. A MPRQ buffer consists of multiple fixed-size strides and each stride receives one packet. MPRQ can improve throughput for small-packet traffic.

Figure 4-9: Throughput comparisons



DPDK implements extra memory copy moving MPRQ buffer to the usual buffer and costs more CPU cycles during the copy. This may result in the performance loss for the single core when enabling MPRQ. For more than eight cores, the advantage of enabling MPRQ can be revealed by providing enough CPU capability and increasing the performance throughput.

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential