Disclaimer
The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED ‘AS IS.” AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

© 2020 Advanced Micro Devices, Inc. All rights reserved.
AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.
Features added/in-progress

- ROCm memory support for perf tools
- Remove dependency on GDRCopy module
- SSE based memcpy for small messages
- Improved agent selection for IPC transfers
- Staged D2D transfers for inter-node
- Added Hardware Tag Matching support
**ROCm memory support for perf tools**

```bash
$ ./bin/ucx_perftest localhost -f -c 2 -t tag_lat -m rocm
```

<table>
<thead>
<tr>
<th></th>
<th>latency (usec)</th>
<th>bandwidth (MB/s)</th>
<th>message rate (msg/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td># iterations</td>
<td>typical</td>
<td>average</td>
<td>overall</td>
</tr>
<tr>
<td>1000000</td>
<td>1.433</td>
<td>7.848</td>
<td>11.025</td>
</tr>
</tbody>
</table>

```bash
$ ./bin/ucx_perftest localhost -f -c 2 -t tag_lat -m host,rocm
```

<table>
<thead>
<tr>
<th></th>
<th>latency (usec)</th>
<th>bandwidth (MB/s)</th>
<th>message rate (msg/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td># iterations</td>
<td>typical</td>
<td>average</td>
<td>overall</td>
</tr>
<tr>
<td>1000000</td>
<td>0.792</td>
<td>2.302</td>
<td>2.338</td>
</tr>
</tbody>
</table>

```bash
$ ./bin/ucx_perftest localhost -f -c 2 -t tag_lat -m rocm,host
```

<table>
<thead>
<tr>
<th></th>
<th>latency (usec)</th>
<th>bandwidth (MB/s)</th>
<th>message rate (msg/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td># iterations</td>
<td>typical</td>
<td>average</td>
<td>overall</td>
</tr>
<tr>
<td>1000000</td>
<td>0.775</td>
<td>2.284</td>
<td>2.292</td>
</tr>
</tbody>
</table>

- UCX perftest now supports ROCm memory (#4587)
- Added support for measuring D2H/H2D transfers (#4607)
Remove dependency on GDRCopy

- No need for GDRCopy module to be installed
- Integrated into rocm_copy module (#4532)

export GDR_DIR=$INSTALL_DIR/gdr export LD_LIBRARY_PATH=$GDR_DIR/lib64:$LD_LIBRARY_PATH
git clone https://github.com/NVIDIA/gdrcopy.git
cd gdrcopy
git checkout -b v1.3 tags/v1.3
mkdir -p $GDR_DIR/lib64 $GDR_DIR/include
make PREFIX=$GDR_DIR lib install
configure --with-gdrcopy=$GDR_DIR …
With no degradation in performance!

**Latency (D-D)**

**BW (D-D)**
SSE based memcpy for small messages

- glibc memcpy is slow for D2H
- Did exhaustive study of the performance of vectorized copy intrinsics
- Added Non-temporal instruction based memcpy (#4532)
- Up to 50x faster than baseline
SSE based memcpy for small messages

MPI D2D Latency

MPI D2H Latency

LATENCY (US)

BYTES

LATENCY (US)

BYTES

old  new
Improved agent selection for IPC transfers

Local Read Remote Write (LRRW)

Remote Read Local Write (LRRW)
Read / Write bandwidth over PCIe 4.0

ROCm Bandwidth Test

• Read performs worse than Write
• get_zcopy performance is worse than put_zcopy
• How to make get_zcopy better?
• Only PCIe systems are affected
Improved agent selection for IPC transfers

- Solution: issue write from the source GPU using ROCR/HSA
- Limitation: Doesn’t work if both devices are not visible to both processes
D2D transfers for inter-node using GDR

Node #0
- GPU #0
- CPU #0

Node #1
- GPU #1
- CPU #1

NIC #0
- NIC #1
GDR Read/Write performance

- GDR Writes can saturate line rate (~200 Gb/s)
- GDR Reads are slow due to PCIe limitations (~60/120 Gb/s)
- Performance depends on PCIe root complex sharing between GPU and NIC
- `put_zcopy` doesn’t help here because the NIC still needs to read from the source GPU#0
Staged D2D transfers for inter-node

Node #0
- GPU #0
- CPU #0

Node #1
- GPU #1
- CPU #1

Must be done by the GPU

Initial Support: @bureddy et al
Add ROCm support: #5742
Staged D2D transfers for inter-node

**D2D Latency**

- 
- MSG SIZE
- 128 KB | 256 KB | 512 KB | 1 MB | 2 MB | 4 MB | 8 MB | 16 MB | 32 MB | 64 MB
- LATENCY (US)
- 0 | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 | 8000 | 9000 | 10000
- GDR | Staged

**D2D Bandwidth**

- 
- MSG SIZE
- 128 KB | 256 KB | 512 KB | 1 MB | 2 MB | 4 MB | 8 MB | 16 MB | 32 MB | 64 MB
- BANDWIDTH (GB/S)
- 0 | 5 | 10 | 15 | 20 | 25
- GDR | Staged
What’s next?
- Support for MPICH

What we are looking forward to?
- Heterogenous topology support
  - How to detect and encode link type? (PCle/Infinity Fabric/NVLink)
  - How to encode distance between PCle devices?