MVAPICH2 Meets Python: Enhancing OMB with Python Benchmarks
Presentation at MUG ‘21

Nawras Alnaasan
Network-Based Computing Laboratory
Dept. of Computer Science and Engineering, The Ohio State University
alnaasan.1@osu.edu
Outline

- OSU Micro Benchmarks (OMB)
- Why Python?
- MPI for Python
- Introducing OMB-Py
- Initial Results
- Conclusion
OSU Micro Benchmarks (OMB)

- OMB is a widely used package to measure performance of MPI operations.
- It helps in optimizing MPI applications on different HPC systems.
- Offers a series of benchmarks including but not limited to point-to-point, blocking and non-blocking collectives, and one-sided tests.
- A large set of options and flags for users to create customizable tests.
- Supports different platforms like ROCm/CUDA to run on ARM/NVIDIA GPUs.
- Written in C
Why Python?

- Second most popular programming language according to the TIOBE index as of August 2021.

<table>
<thead>
<tr>
<th>Aug 2021</th>
<th>Aug 2020</th>
<th>Change</th>
<th>Programming Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>^</td>
<td>Python</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>↓</td>
<td>Java</td>
</tr>
</tbody>
</table>

Adopted from: https://www.tiobe.com/tiobe-index/

- Great community and support around Machine Learning, Big Data, and Cloud Computing
  - Many deep learning frameworks like PyTorch, Tensorflow, Keras, etc.
  - One of the most important tools for data science and analytics.
- Great Potential to use MPI and distributed computing techniques.
- Many other Python libraries and frameworks like SciPy, Numpy, Django, etc.
Why Python?

Very flexible and simplified syntax.

Adopted from:
https://www.hardikp.com/2017/12/30/python-cpp/
MPI for Python

• MPI support in Python is needed for multiple purposes especially for distributed computing applications (DL, data science, etc.)
• Python needs a wrapper to use MPI functionalities.
• mpi4py is the most widely used wrapper to provide Python bindings for MPI
• Other wrappers exist like the torch.distributed package from PyTorch
For OMB-Py, we mainly use mpi4py to provide Python bindings for MPI.

- Supports communication for selected Python objects as memory buffers (bytearrays, Numpy, etc.)
- Capable of serializing general Python objects using built-in “Pickle” module.
  - MPI.Send() uses supported data structures as memory buffers.
  - MPI.send() serializes objects before sending data.
- Recent releases +v3.1 supports GPU-aware MPI.
Introducing OMB-Py

Python Applications

OMP-Py Benchmarks

Python

mpi4py

MPI Implementation for CPU or GPU

Hardware components
What does OMB-Py offer?

• Follows a similar design to the original OMB and brings it into Python.
• Support for point-to-point and collective MPI operations.
• Mimics behavior of real python applications that use MPI.
• Support for both CPU and GPU tests using CUDA-aware Python data structures as buffers like CuPy, Numba, and PyCUDA.
• A wide range of command-line arguments to run customizable tests.
### Sample Latency Output

<table>
<thead>
<tr>
<th>Size</th>
<th>Latency (us)</th>
<th>Size</th>
<th>Latency (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.12</td>
<td>0</td>
<td>1.55</td>
</tr>
<tr>
<td>1</td>
<td>1.14</td>
<td>1</td>
<td>1.58</td>
</tr>
<tr>
<td>2</td>
<td>1.14</td>
<td>2</td>
<td>1.58</td>
</tr>
<tr>
<td>4</td>
<td>1.15</td>
<td>4</td>
<td>1.58</td>
</tr>
<tr>
<td>8</td>
<td>1.14</td>
<td>8</td>
<td>1.58</td>
</tr>
<tr>
<td>16</td>
<td>1.18</td>
<td>16</td>
<td>1.62</td>
</tr>
<tr>
<td>32</td>
<td>1.18</td>
<td>32</td>
<td>1.62</td>
</tr>
<tr>
<td>64</td>
<td>1.19</td>
<td>64</td>
<td>1.63</td>
</tr>
<tr>
<td>128</td>
<td>1.25</td>
<td>128</td>
<td>1.63</td>
</tr>
<tr>
<td>256</td>
<td>1.66</td>
<td>256</td>
<td>1.68</td>
</tr>
<tr>
<td>512</td>
<td>1.74</td>
<td>512</td>
<td>2.10</td>
</tr>
<tr>
<td>1024</td>
<td>1.93</td>
<td>1024</td>
<td>2.18</td>
</tr>
<tr>
<td>2048</td>
<td>2.29</td>
<td>2048</td>
<td>2.36</td>
</tr>
<tr>
<td>4096</td>
<td>3.19</td>
<td>4096</td>
<td>2.73</td>
</tr>
<tr>
<td>8192</td>
<td>4.59</td>
<td>8192</td>
<td>3.63</td>
</tr>
<tr>
<td>16384</td>
<td>6.89</td>
<td>16384</td>
<td>5.03</td>
</tr>
<tr>
<td>32768</td>
<td>9.18</td>
<td>32768</td>
<td>7.24</td>
</tr>
<tr>
<td>65536</td>
<td>12.06</td>
<td>65536</td>
<td>9.79</td>
</tr>
<tr>
<td>131072</td>
<td>21.29</td>
<td>131072</td>
<td>12.73</td>
</tr>
<tr>
<td>262144</td>
<td>32.07</td>
<td>262144</td>
<td>22.17</td>
</tr>
<tr>
<td>524288</td>
<td>54.40</td>
<td>524288</td>
<td>33.01</td>
</tr>
<tr>
<td>1048576</td>
<td>97.02</td>
<td>1048576</td>
<td>55.19</td>
</tr>
<tr>
<td>2097152</td>
<td>181.26</td>
<td>2097152</td>
<td>97.53</td>
</tr>
<tr>
<td>4194304</td>
<td>351.69</td>
<td>4194304</td>
<td>186.07</td>
</tr>
</tbody>
</table>

**OMB**

**OMB-Py**

Similar CLI options
OMB-Py: Initial CPU Results

osu_latency inter-node initial results

- Small Python overhead around 0.5 microseconds.
- Overhead only apparent in smaller message sizes

Small message size

Large message size
OMB-Py: Initial GPU Results

Small message size

- Different overheads depending on the CUDA-aware data structure
  - Around 3 microseconds for CuPy and PyCUDA
  - Around 5 microseconds for Numba
- Overhead is only apparent in smaller message sizes
Conclusion

• MPI support for Python is crucially needed in the Python community.
• MPI benchmarks are needed to measure and optimize MPI performance.
• mpi4py provides Python bindings for MPI and is used with OMB-Py.
• OMB-Py provides Python MPI benchmarking similar to OMB in C.
• Initial results show a small overhead due to Python-MPI bindings layer and the use of Python objects as buffers for communication.
Thank You!

alnaasan.1@osu.edu

Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

High Performance Deep Learning
http://hidl.cse.ohio-state.edu/

The High-Performance MPI/PGAS Project
http://mvapich.cse.ohio-state.edu/

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/