Support in OpenMP for Multi-GPU Parallelism

Raul Torres[1], Vivek Kale[2], Abid Malik[2], Tom Scogland[3], Roger Ferrer[1], Barbara M. Chapman[2]

[1] Barcelona Supercomputing Center, Barcelona, Spain
[2] Brookhaven National Laboratory, Upton, NY, USA
[3] Lawrence Livermore National Laboratory, Livermore, CA, USA

Motivation for OpenMP Support for Multi-GPUs

- Multi-GPU: a collection of GPUs on a node.
- MPI is often used by applications to parallelize their work across GPUs of a multi-GPU [4].
- A more performant as well as a portable solution for such parallelism is needed.
- OpenMP [1] is used to parallelize computation within a multi-core or within each GPU, but it could also be employed to achieve parallelism across the GPUs of a multi-GPU because:
  - OpenMP provides for incrementally introducing parallelism in code.
  - One programming model is within the entire node rather than two.
  - OpenMP has a reduced memory footprint compared to MPI.
  - OpenMP provides low-cost load balancing support through its scheduling and tasking features.
- In this work, we present an approach and a reference implementation in the widely-used open-source LLVM OpenMP implementation [2] to support OpenMP parallelization of an application's work across GPUs of a multi-GPU.

Implementation details

Using Dependencies with the target spread Directive

```
#pragma omp parallel
#pragma omp single
{
#pragma omp target spread \
  nowait \
  devices(2,0,1) \
  spread_schedule(static, 4) \
  map(to:    A[omp_spread_start:omp_spread_size]) \
  map(from:  B[omp_spread_start:omp_spread_size]) \
  depend(out:B[omp_spread_start:omp_spread_size])
for (int i=0; i < N; i++){
  B[i] = A[i] + 10;
}
#pragma omp target spread \
  nowait \
  devices(3,5,4) \
  spread_schedule(static, 4) \
  map(to:   B[omp_spread_start:omp_spread_size]) \
  map(from: A[omp_spread_start:omp_spread_size]) \
  depend(in:B[omp_spread_start:omp_spread_size])
 for (int i=0; i < N; i++){
    A[i] = B[i] + 10;
 }
}
```

- The **spread_schedule** clause supports only static schedule at the moment.
- It assigns the generated chunks, dictated by the chunk size, to each device specified in the **devices** clause in a round-robin fashion.
- If the chunk size is not specified, our scheduler will try to provide each device a single chunk of similar size .
- OpenMP does not allow explicit overlap and extension of array sections in a device.
- The static schedule automatically complies with this restriction by providing each device a set of non-contiguous chunks.
- A dynamic schedule is more complex, because it would have to add a verification step for making sure that a new array section does not overlap and extend any of the previously mapped ones.

Language Extension with Compiler Support

We have extended LLVM's OpenMP language and implementation by adding a new directive, **target spread,** which offloads chunks of a **for** loop into multiple devices in a distributed way:

- The clause **devices** specifies the accelerators to be sharing the workload.
- The clause **spread_schedule** sets the distribution strategy and the chunk size.
- The delimiter **omp_spread_start** refers to the start of each chunk at execution time.
- The delimiter **omp_spread_size** refers to the actual chunk size at execution time.

Syntax for the target spread directive

```
#pragma omp target spread \
 devices(<list of devices>) \
 spread_schedule(<strategy>, <chunk size>) \
 map(to:    A[omp_spread_start:omp_spread_size]) \
 map(from:  B[omp_spread_start:omp_spread_size]) \
for (int i=0; i < N; i++){
  …
}
```

Using the target spread Directive with a Stencil Sum

```
#pragma omp target spread \
 devices(2,0,1) \
 spread_schedule(static, 4) \
 map(to:  A[omp_spread_start-1:omp_spread_size+2]) \
 map(from:B[omp_spread_start  :omp_spread_size  ])
for (int i=1; i < N-1; i++){
  B[i]=A[i-1]+A[i]+A[i+1];
}
```

Evaluation

Performance of **target** vs **target spread**

Kernel	Measure	target (1 GPU)	target spread (1 GPU)	**target spread (4 GPUs)**	Gain
Stream	Time (s)	0.880737	0.925076	**0.145715**	6.38x

Impact of chunk size variation in **target spread** (max_streams=40)

Chunk size	25000	**62500**	250000	2500000
Time (s)	0.246214	**0.145715**	0.187575	0.324199
Kernels per GPU	100	**40**	10	1
Streams per GPU	40	**40**	10	1
Occupancy(streams/max_streams)	1	**1**	0.25	0.025
Load Balance(kernels/max_streams)	2.5	**1**	0.25	0.25

Discussion

- With the standalone **target spread** directive, the distribution of data will be tied to the loop distribution (execution-centric) and must perform data movements between the devices and the host.
- The **target data spread** directive is under development. Its purpose is to create a data region for several devices at the same time, where variables can be mapped individually with different strategies (data-centric).

Conclusions

- We have presented in this work our design choices, implementation and early results of the features for OpenMP to parallelize across GPUs of a multi-GPU, and more broadly a set of heterogeneous devices.
- Our implementation and results show promise for use in applications and over the conventional method of MPI parallelism on multi-GPUs.

Future Work

- Further versions of the **target spread** directive might allow the specification of device classes, e.g., devices(gpu), devices(cpu), devices(fpga), which are beneficial for future architectures with extremely heterogeneous processors.
- We plan to (1) experiment with the compiler support of target spread directive with the runtime systems support of task-to-GPU scheduling and (2) a more sophisticated set of task-to-GPU scheduling strategies that are more adaptive based on historical information about the GPUs.

REFERENCES

[1] OpenMP 5.0 Reference Guide. https://www.openmp.org/wp-content/uploads/OpenMPRef-5.0-1119-01-TSK-web.pdf .

[2] The LLVM Compiler Infrastructure.. http://llvm.org/.

[3] Leopold Grinberg, Carlo Bertolli, and Riyaz Haque. Hands on with OpenMP 4.5 and Unified Memory: Developing Applications for IBM's Hybrid CPU + GPU Systems (Part II), 2017.

[4] Vivek Kale, Wenbin Lu, Anthony Curtis, Abid Muslim Malik, Barbara M. Chapman,and Oscar R. Hernandez. Toward Supporting Multi-GPU Targets via Taskloop and User-defined Schedules. In Kent Milfeld, Bronis R. de Supinski, Lars Koesterke, andJannis Klinkenberg, editors,OpenMP: Portable Multi-Level Parallelism on ModernSystems - 16th International Workshop on OpenMP, IWOMP 2020, Austin, TX, USA,September 22-24, 2020, Proceedings, volume 12295 ofLecture Notes in ComputerScience, pages 295–309. Springer, 2020.

[5] Rengan Xu, Sunita Chandrasekaran, and Barbara Chapman. Exploring Programming Multi-GPUs Using OpenMP and OpenACC-based Hybrid Model. IPDPS 2013 Workshop and PhD Forum. pp. 1169-1176. May 2013.

Acknowledgment

This research was supported in part by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Ad-ministration. This work was supported in part by MEEP project, which has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No946002. The JU receives support from the European Union's Hori-on 2020 research and innovation programme and Spain, Croatia,Turkey.

Contact: Raul Torres (raul.torres1@bsc.es) and Vivek Kale (vkale@bnl.gov)