Global Address Space Programming Interface

Modell, Erfahrungen, Skalierbarkeit

ZKI-Arbeitskreis Supercomputing
Zeuthen, 19. Mai 2011

Dr. Mirko Rahn
Fraunhofer ITWM
Competence Center for HPC and Visualization
GPI – Global address space programming interface

PGAS – Partitioned global address space

- read/write global data, one-sided, non-blocking, asynchronous, zero-copy
- send/recv messages, passive
- barrier, allReduce
- atomic counters
- ranks
- queues
- fault tolerance

- daemon concept
- NUMA support
- auto detect and auto select network
GPI – Example code: Alltoall in C

```
#include <GPI.h>

extern void dump (const char *, const int *);

int main (int argc, char *argv[]) {
    startGPI (argc, argv, "", (1UL << 30));       // 1 GiB DMA enabled memory per node

    const int iProc = getRankGPI ();
    const int nProc = getNodeCountGPI ();

    int *mem = (int *)getDmaMemPtrGPI ();        // begin of DMA enabled memory
    int *src = mem;               // offset 0
    int *dst = mem + nProc;       // offset nProc * sizeof(int)

    for (int p = 0; p < nProc; ++p) {
        src[p] = iProc * nProc + p;

        const unsigned long locOff = p * sizeof (int);
        const unsigned long remOff = (nProc + iProc) * sizeof (int);

        writeDmaGPI (locOff, remOff, sizeof (int), p, GPIQueue0);
    }

    waitDmaGPI (GPIQueue0);
    barrierGPI ();

    dump ("src", src);
    dump ("dst", dst);

    shutdownGPI ();
}
```

> gcc alltoall.c -I $GPI_HOME/include -L $GPI_HOME/lib64 -lGPI -libverbs15 -o $GPI_HOME/bin/alltoall

> getnode -n 4

> $GPI_HOME/bin/alltoall

 [...] collected and sorted output...

<table>
<thead>
<tr>
<th>src 0:</th>
<th>0 1 2 3</th>
<th>dst 0:</th>
<th>0 4 8 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>src 1:</td>
<td>4 5 6 7</td>
<td>dst 1:</td>
<td>1 5 9 13</td>
</tr>
<tr>
<td>src 2:</td>
<td>8 9 10 11</td>
<td>dst 2:</td>
<td>2 6 10 14</td>
</tr>
<tr>
<td>src 3:</td>
<td>12 13 14 15</td>
<td>dst 3:</td>
<td>3 7 11 15</td>
</tr>
</tbody>
</table>

© Fraunhofer ITWM
GPI – BQCD

- Berlin Quantum Chromodynamics (Stüben, Nakamura)
- QCD = fundamental theory of strong interaction, e.g. prediction of particle masses
- BQCD: typical supercomputer application, benchmark at LRZ and ZIB
- Written in Fortran90, communication by MPI or shmem

- hybrid MPI/OMP implementation, master communicates
- domain decomposition of the 4D volume across the processes
- 50% of runtime spent in evaluation of the co-variante derivate \( d \)
- 4D nearest neighbour stencil → boundary exchange problem
GPI – BQCD with GPI

- Naïve conversion: Write shmem interface for GPI, still no overlap

- A simple substitution of MPI-calls by GPI-calls is not the solution!
GPI - BQCD with GPI and overlapping

- Overlap communication and computation, *not GPI specific!*

**MPI:**
- `Isend`, `Irecv`, `Testall`

**GPI:**
- `writeDMA`, synchronization protocol
GPI - BQCD with GPI and overlapping, speedup

BQCD (24 × 24 × 24 × 48): speedup
(HLRS, Xeon E5440, 16 GiB, DD Infiniband)

Speedup normalized to 16 core MPI/OMP nonoverlapping performance

<table>
<thead>
<tr>
<th></th>
<th>speedup @512 cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI/OMP sequential</td>
<td>30.24</td>
</tr>
<tr>
<td>MPI/OMP overlap</td>
<td>42.77</td>
</tr>
<tr>
<td>GPI/OMP overlap</td>
<td>48.37</td>
</tr>
</tbody>
</table>
GPI - Summary

GPI: powerful tool for the development of parallel applications
- fast: wirespeed, zero-copy
- scalable: perfect overlap, no cycles for communication
- easy to use: with MPI experience easy and intuitive to program
- (single programming model: no hybrid implementation anymore)

There is no free lunch:
- conceptual algorithmic changes might be required
- however, these changes have usually nothing to do with GPI itself