DASH Status Update
SPPEXA Annual Plenary Meeting 2017
Garching, March 20-22, 2017

www.dash-project.org

Karl Fürlinger
Ludwig-Maximilians-Universität München
DASH Overview

- DASH is a C++ template library that offers
  1. Distributed data structures and parallel algorithms that can be integrated in existing applications
  2. A complete PGAS (part. global address space) programming system without requiring a custom compiler

**Units**: The individual participants in a DASH program, usually full OS processes.

**Shared data**: managed by DASH in a virtual global address space

**Private data**: managed by regular C/C++ mechanisms

```
int a;
int b;
int c;
```

```
dash::Array<int> a(1000);
dash::Shared<int> s;
```

```
0..9
```

```
10..19
```

```
..999
```

```
int a;
... int c;
```
DASH Project Structure

<table>
<thead>
<tr>
<th>Tools and Interfaces</th>
<th>DASH Application</th>
<th>DASH C++ Template Library</th>
<th>DART API</th>
<th>DASH Runtime (DART)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware: Network, Processor, Memory, Storage</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LMU Munich</td>
<td>Project management, C++ template library</td>
</tr>
<tr>
<td></td>
<td>Project management, C++ template library, DASH data dock</td>
</tr>
<tr>
<td>TU Dresden</td>
<td>Libraries and interfaces, tools support</td>
</tr>
<tr>
<td></td>
<td>Smart data structures, resilience</td>
</tr>
<tr>
<td>HLRS Stuttgart</td>
<td>DASH runtime (DART)</td>
</tr>
<tr>
<td></td>
<td>DASH runtime (DART)</td>
</tr>
<tr>
<td>KIT Karlsruhe</td>
<td>Application case studies</td>
</tr>
<tr>
<td></td>
<td>Smart deployment, Application case studies</td>
</tr>
<tr>
<td>IHR Stuttgart</td>
<td></td>
</tr>
</tbody>
</table>

www.dash-project.org
Recent Activities

Features:

- DASH multi-dimensional array (dash::NArray) and data distribution patterns [1],[6]
- Application engagement: LULESH, others [2],[5],[7]
- Support for performance tools and debuggers with DASH [3]
- Correctness tool: Nasty-MPI [4]
- DART progress engine [8]

Project management and outreach

- Release of DASH v0.2.0 and move to GitHub
- DASH Tutorial @HiPEAC in Stockholm
- DASH presentations at LLNL, LBNL, ORNL, TU Wien, KTH Stockholm
- DASH v0.3.0 release imminent
The data distribution pattern is configurable

```
dash::Array<int> arr1(20); // default: BLOCKED

dash::Array<int> arr2(20, dash::BLOCKED)
dash::Array<int> arr3(20, dash::CYCLIC)
dash::Array<int> arr4(20, dash::BLOCKCYCLIC(3))
```

Assume 4 units

<table>
<thead>
<tr>
<th>arr1, arr2</th>
<th>BLOCKED</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>arr3</th>
<th>CYCLIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>arr4</th>
<th>BLOCKCYCLIC(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19</td>
<td></td>
</tr>
</tbody>
</table>
dash::Pattern<N> specifies N-dim data distribution
- Blocked, cyclic, and block-cyclic in multiple dimensions

Pattern<2>(20, 15)
Multi-dimensional Data Distribution (2)

- Tiled data distribution and tile-shifted distribution

TilePattern<2>(20, 15)

(TILE(2), TILE(5))

ShiftTilePattern<2>(32, 24)

(TILE(4), TILE(3))
Multi-dimensional Data Distribution (3)

- Row-major and column-major storage

Pattern\(<2, \text{COL\_MAJOR}>(20, 15)\)

TilePattern\(<2, \text{COL\_MAJOR}>(20, 15)\)

(BLOCKED, NONE)

(TILE(5), TILE(5))
**The Multi-Dimensional Array**

- `dash::NArray` (`.dash::Matrix`) offers a distributed multi-dimensional array abstraction
  - Dimension is a template parameter
  - Element access using coordinates or linear index
  - Support for custom index types
  - Support for row-major and column-major storage

```cpp
dash::NArray<int, 2> mat(40, 30); // 1200 elements

int a = mat(i,j);    // Fortran style access
int b = mat[i][j];   // chained subscripts

auto loc = mat.local;

int c = mat.local[i][j];
int d = *(mat.local.begin()); // local iterator
```
Local view works similar to 1D array

dash::NArray<char, 2> mat(7, 4);
cout << mat(2, 1) << endl; // prints 'j'

if(dash::myid()==0) {
    cout << mat.local(2, 1) << endl; // prints 'z'
}
Application Example: S(R)UMMA Algorithm

- Block matrix-matrix multiplication algorithm with block prefetching

while(!done) {
    blk_a = ...
    blk_b = ...
    // prefetch
    auto get_a = dash::copy_async(blk_a.begin(), blk_a.end(), lblk_a_get);
    auto get_b = dash::copy_async(blk_b.begin(), blk_b.end(), lblk_b_get);
    // local DGEMM
    dash::multiply(lblk_a_comp, lblk_b_comp, lblk_c_comp);
    // wait for transfer to finish
    get_a.wait(); get_b.wait();
    // swap buffers
    swap(lblk_a_get, lblk_a_comp); swap(lblk_b_get, lblk_b_comp);
}
DGEMM on a Single Shared Memory Node

- LRZ SuperMUC, phase 2: Haswell EP, 1.16 Tflop/sec peak

- DASH: Multi-process, using one-sided communication and single-threaded MKL
- MKL: Multithreaded, using OpenMP
- PLASMA: Multithreaded tile-algorithms on top of sequential BLAS

![Image source: [1]]
PDGEMM: DASH vs. ScaLAPACK Multinode

- Strong scaling on SuperMUC (57344 × 57344 matrix)

- Trace: Overlapping communication and computation

Image source: [1]
Porting LULESH to DASH (ongoing work)

- LULESH (Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics code)
  - Widely used mini-app/proxy application

Goals of the DASH port

- Remove limitations of MPI domain decomposition approach
- Avoid replication, manual index calculation, bookkeeping

```c
if (rowMin | rowMax) {
    /* ASSUMING ONE DOMAIN PER RANK, CONSTANT BLOCK SIZE HERE */
    int sendCount = dx * dz;

    if (rowMin) {
        destAddr = &domain.commDataSend[pmsg * maxPlaneComm];
        for (Index t fi=0; fi<xferFields; ++fi) {
            Domain_member src = fieldData[fi];
            for (Index t i=0; i<dz; ++i) {
                for (Index t j=0; j<dx; ++j) {
                    destAddr[i*dx+j] = (domain.*src)(i*dx*dy + j);
                }
            }
        }
        destAddr += sendCount;
    }
    destAddr -= xferFields*sendCount;

    MPI_Isend(destAddr, xferFields*sendCount, baseType, myRank - domain.tp(), msgType, MPI_COMM_WORLD, &domain.sendRequest[pmsg])
    ++pmsg;
```
DASH can be integrated in existing applications and allows for incremental porting

Porting options:
1. Port data structures, but keep communication as-is (using MPI two-sided)
   - Can use HDF5 writer for checkpointing
2. Keep explicit packing code but use one-sided put instead of MPI_Irecv/MPI_Isend
   - Potential performance benefit from one-sided communication
3. Use DASH for communication directly
   - auto halo = ...; dash::swap(halo) ...
   - less replicated code, more flexibility
4. Use dash::HaloNArray
   - Multi-dimensional array with built-in halo areas
Performance and scalability (weak scaling) of LULESH, implemented in MPI and DASH

Scaling of LULESH on SuperMUC-HW

Image source: [2]
DASH on GitHub

- https://github.com/dash-project/dash

DASH

A C++ Template Library for Distributed Data Structures with Support for Hierarchical Locality for HPC and Data-Driven Science.

Summary

Exascale systems are scheduled to become available in 2018-2020 and will be characterized by extreme scale and a multilevel hierarchical organization.
DASH Tutorial

Available at http://www.dash-project.org/tutorial/tutorial.html
DASH On-going Work

- **Memory Spaces**
  - Support for NVRAM, high-bandwidth memory, replica mgmt

- **Task-Based Execution**
  - Compute kernels and dependency mgmt

- **Dynamic Data Structures (growing/shrinking)**
  - List and hashmap as first goals

- **Hierarchical Locality Information System**
  - Make working with locality easier

- **Halo Matrix Smart Data Structure**
  - NArray with built-in support for halo exchange
Summary

- DASH is
  - A complete data-oriented PGAS programming system (i.e., entire applications can be written in DASH),
  - A library that provides distributed data structures (i.e., DASH can be integrated into existing MPI applications)

- The DASH Team
  T. Fuchs (LMU), R. Kowalewski (LMU), D. Hünich (TUD), A. Knüpfer (TUD), J. Gracia (HLRS), C. Glass (HLRS), H. Zhou (HLRS), K. Idrees (HLRS), J. Schuchart (HLRS), F. Mößbauer (LMU), K. Fürlinger (LMU)

- More information
  - http://www.dash-project.org/
  - https://github.com/dash-project/dash/
References (1)


References (2)

