Distributed Data Structures in a Global Address Space

dash

www.dash-project.org
Dr. Karl Fürlinger
Ludwig-Maximilians-Universität (LMU)
München
Motivation: A Typical Parallel Machine, early 2000s

- IBM xSeries Cluster
  - Intel Xeon, Pentium 4, 3.06 GHz
  - Gigabit Ethernet interconnect
  - Scales to 1024 cores
  - Several entries in the Top500 list (2004) around rank 100

- Ethernet / Infiniband Interconnect
  - Single / dual core
  - Moderate scale
  - No heterogeneity
  - No NUMA effects
  - Fairly straightforward programming: message passing in the form of MPI
Increasing HW Complexity over Time

- **Interconnect Complexity**
- **NUMA: Non-Uniform Memory Access**
- **High-Bandwidth Memory**
- **Scratchpad Memory**
- **Non-volatile Memory**
- **Multilevel hierarchical interconnects**

**Memory Heterogeneity**

**Compute Heterogeneity**

- Multicore
- Multicore w/ NUMA
- Manycore (KNL)
- Integrated APU
- Heterogeneous / Multi-GPU

Tokyo, Oct. 30, 2018 | 3
New Application Areas

Complexity of Parallel Systems

Parallel Computing User Base / Application Areas

Life Sciences

Digital Humanities

Image sources: [1,2]

Approaches

- Specialized packages, libraries, frameworks,...
- Domain specific languages (DSLs)
- Still need **general parallel programming approaches** for the cases that are not supported well by the above
- For these, **productivity** is as important as performance!

DASH - Overview

- DASH is a C++ template library, that offers
  - Distributed data structures, e.g., `dash::Array<int>`
  - Parallel algorithms, e.g., `dash::sort()`

- Generalizes shared memory programming to distributed memory systems:

  - Multiple threads access **physically** shared memory
  - Multiple nodes connected by a high-speed network
  - Multiple threads (“units”) access **logically** shared memory
DASH - A PGAS Programming System

- DASH realizes a PGAS (Partitioned Global Address Space) abstraction
  - Global address space: accessible from everywhere
  - Partitioned: data distribution is **configurable** and **not hidden**

- Example

```
unit 0   unit 1   unit 2   unit 3
dash::Array<int> a;
```

- Example Data Distribution Patterns (1D)

```
<table>
<thead>
<tr>
<th>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCKED</td>
</tr>
<tr>
<td>CYCLIC</td>
</tr>
<tr>
<td>BLOCKCYCLIC(3)</td>
</tr>
</tbody>
</table>
```
DASH – Project Overview

<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></td>
<td></td>
<td></td>
<td></td>
<td>One-sided Communication Substrate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MPI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Hardware: Network, Processor, Memory, Storage</td>
</tr>
</tbody>
</table>

**DASH Run-time (DART)**

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

www.dash-project.org

DASH is one of 16 SPPEXA projects
French/Japanese Collaborations

- No official (funded) collaboration with French / Japanese partners at the moment

- Emmanuel Jeannot, Brice Goglin (INRIA Bordeaux): collaboration on data locality
  - COLOC workshop and poster session at EuroPar
  - hwloc / dyloc integration

- Mitsuhisa Sato (RIKEN): Collaboration on task-based execution
  - Joseph Schuchart (HLRS): visit / stay at Riken in 2017

- Planned: Camille Coti (Univ. Paris 13): Collaboration on numerical linear algebra in PGAS
Performance and Productivity Features (1)

- **Global data structures**
  - Avoid the need for manual index calculations, especially for multidimensional data structures
  - Both **globalview** semantics and **localview** semantics supported

```cpp
dash::Array<int> a(14);
```

- **Globalview** supports access by global indices, global pointers, global iterators, etc.
- **Localview** supports access by local indices, local pointers, etc.
- Access to local data is exposed for performance
## Global Data Structures Overview

<table>
<thead>
<tr>
<th>Container</th>
<th>Description</th>
<th>Data distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared(&lt;T&gt;)</td>
<td>Shared Scalar</td>
<td></td>
</tr>
<tr>
<td>Array(&lt;T&gt;)</td>
<td>1D Dist. Array</td>
<td></td>
</tr>
<tr>
<td>NArray(&lt;T, N&gt;)</td>
<td>N-dim. Dist. Array</td>
<td></td>
</tr>
<tr>
<td>Coarray(&lt;T[R][S]&gt;)</td>
<td>CAF-like Coarray</td>
<td></td>
</tr>
<tr>
<td>List(\textasterisk)&lt;(T), Map(\textasterisk)&lt;(T)&gt;</td>
<td>Dynamic data structures (growing/shrinking)</td>
<td></td>
</tr>
</tbody>
</table>

(*) Under Development
STL compatibility and parallel algorithms

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

if (dash::myid()==0 ) {
  // (sequential!) sort
  // using glob. iterators with std. algorithms
  std::sort(a.begin(), a.end());
}

// parallel DASH algorithm using global iterators
auto min = dash::min_element(a.begin(), a.end());

// range-based for loops on local and global ranges
for( auto& el : a.local ) {
  el = 23;
}
```
Case Study: The Cowichan Problem Suite

Cowichan problems
- A **benchmark suite** designed to investigate the usability of parallel programming systems (1990s)
- 13 small “toy” problems, quick to implement, composable by chaining [1]

Previous work by Nanz et al. [2] selected **five benchmarks** to evaluate the usability of multicore languages
- Four programming systems compared:
  - Go, Cilk, TBB, Chapel
- Metrics:
  - **Usability**: LOC, development time
  - **Performance**: execution time and scalability


DASH is not the most concise approach, but not much worse than the best solution

- DASH is the only case where the same code can be run on shared memory and distributed memory systems!
Platform: **Single node** of SuperMUC Phase 2 (Haswell)
- Haswell Xeon E5-2697, 2.6 GHz, 28 cores per node
- 64 GB of main memory
- Intel Compiler (icc) v. 18.0.2 used for all programming systems

Absolute performance, using all 28 cores, 30k x 30k Matrix

Performance relative to DASH, using all 28 cores, 30k x 30k Matrix
Platform: **Up to 16 nodes** of SuperMUC
- Haswell Xeon E5-2697, 2.6 GHz, 28 cores per node
- 64 GB of main memory
- DASH is the **only** approach that can also use distributed memory machines (**the same source code**)

**Multinode Scaling (30k x 30k) on SuperMUC-HW**

**Multinode Scaling (80k x 80k) on SuperMUC-HW**

30k x 30k, Speedup vs 1 node

80k x 80k, Speedup vs 2 nodes
Summary

- **DASH**
  - Is complete data-oriented PGAS programming system (i.e., entire applications can be written in DASH),
  - Is a library that provides distributed data structures (i.e., DASH can be integrated into existing MPI applications)
  - Aims to balance performance and productivity features

- **Productivity**
  - Data-structure oriented
  - Global-view access
  - Parallel algorithms
  - Memory space abstraction (HBM, NVRAM)
  - Integrated parallel I/O

- **Performance**
  - Local-view access
  - Asynchronous comm. (*)
  - Hierarchical teams (*)
  - Tasks (*)

(*) Not Discussed Here
Acknowledgements

The DASH Team

- T. Fuchs (LMU)
- R. Kowalewski (LMU)
- D. Hünich (TUD)
- A. Knüpfer (TUD)
- J. Gracia (HLRS)
- C. Glass (HLRS)
- K. Fürlinger (LMU)
- P. Jungblut (LMU)
- B. Wesarg (TUD)
- D. Bonilla (HLRS)
- H. Zhou (HLRS)
- J. Schuchart (HLRS)
- F. Mößbauer (LMU)

Github: [https://github.com/dash-project/dash/](https://github.com/dash-project/dash/)
Web: [http://www.dash-project.org](http://www.dash-project.org)