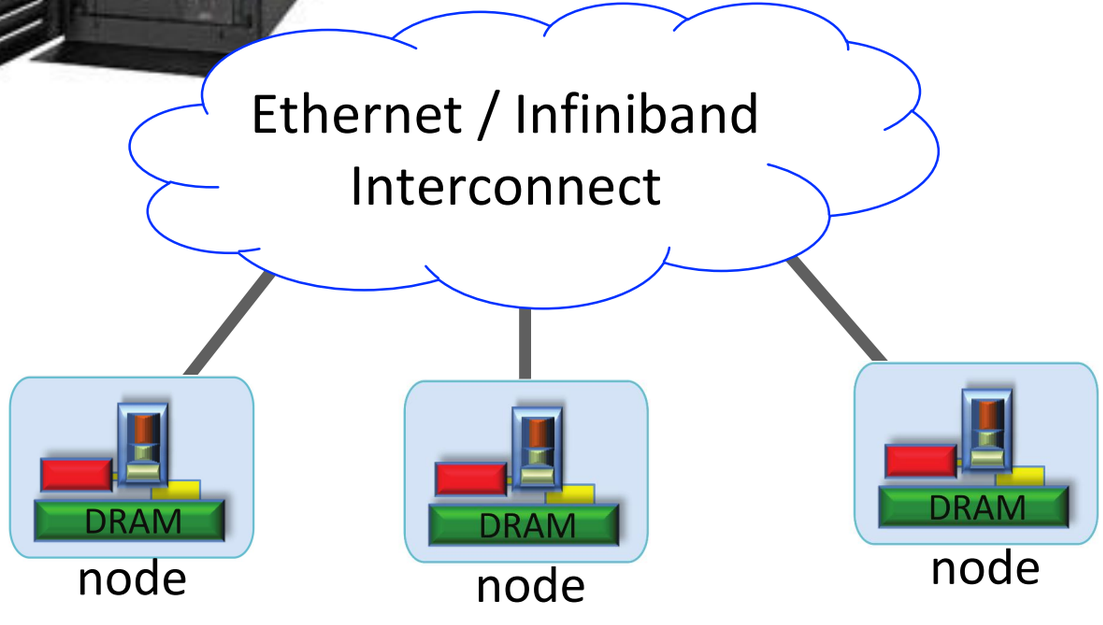
Distributed Data Structures in a Global Address Space

www.dash-project.org
Dr. Karl Fürlinger
Ludwig-Maximilians-Universität (LMU)
München

Motivation: A Typical Parallel Machine, early 2000s

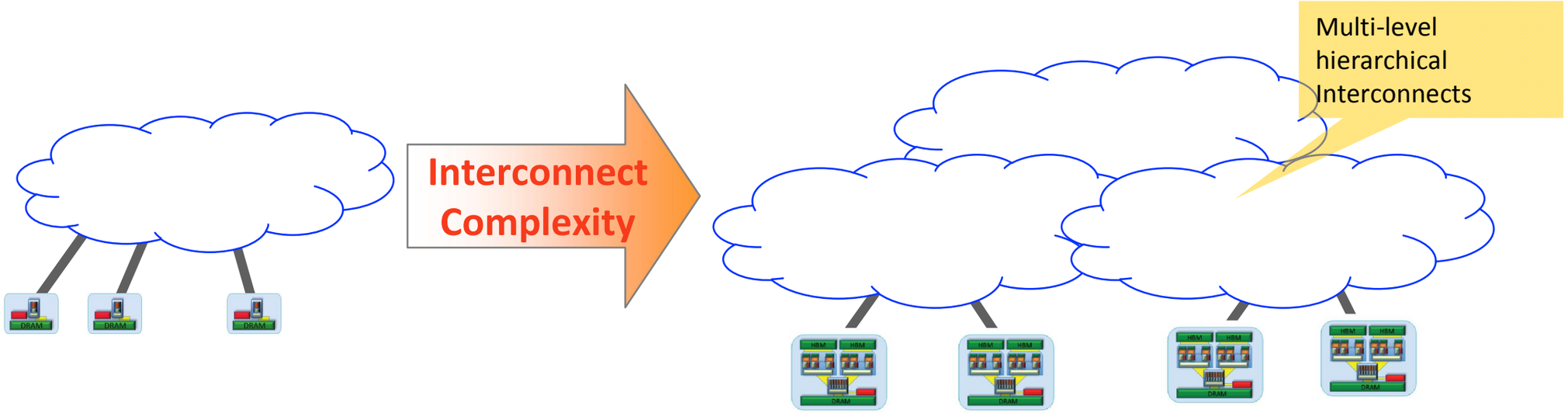
Image Source: IBM

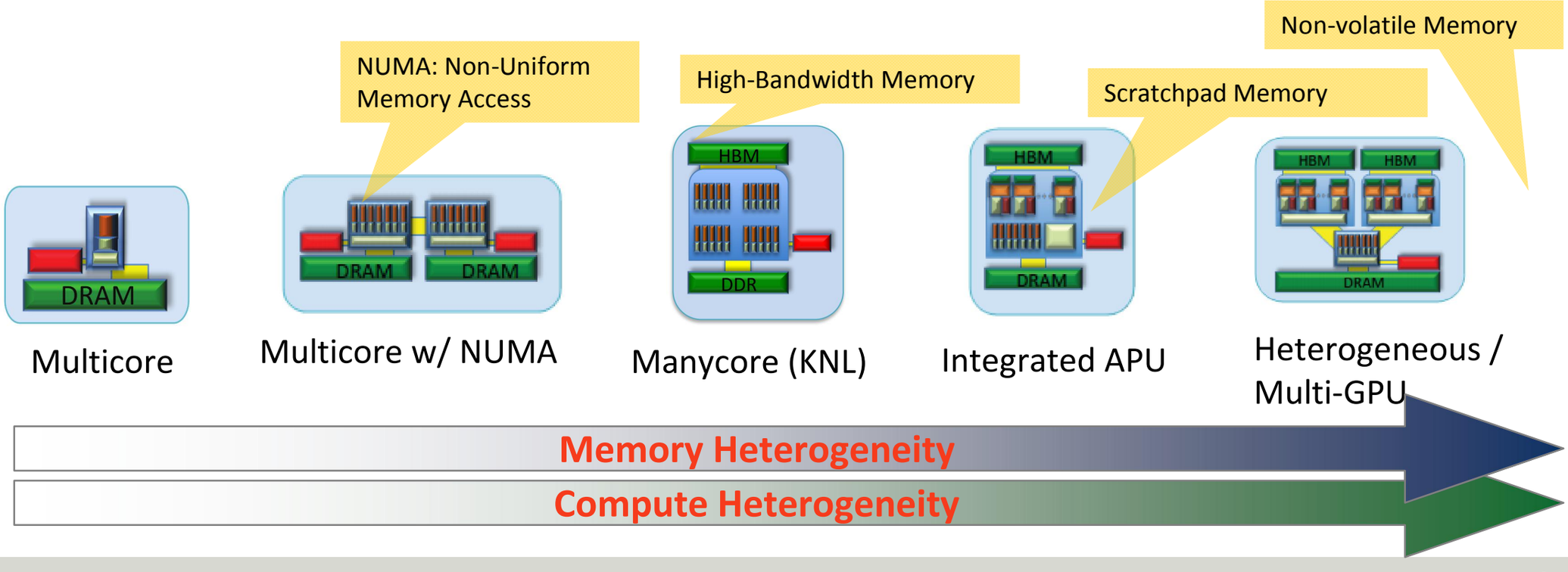
- IBM xSeries Cluster
  - Intel Xeon, Pentium 4, 3.06 GHz
  - Gigabit Ethernet interconnect
  - Scales to 1024 cores
  - Several entries in the Top500 list (2004) around rank 100



- 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





New Application Areas

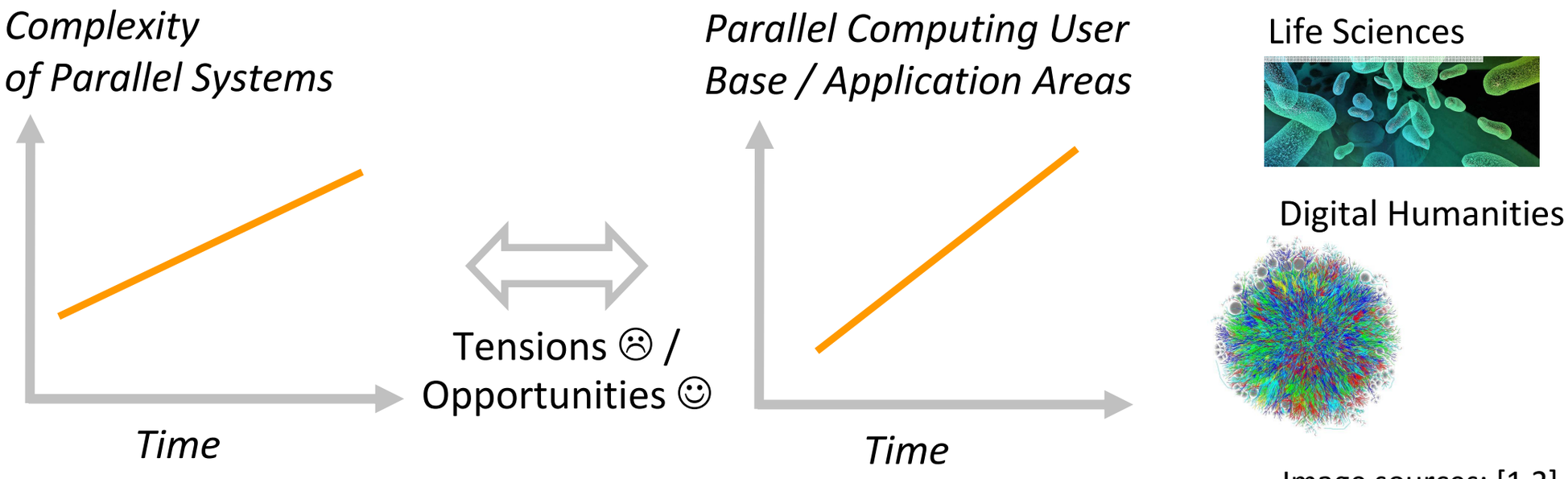


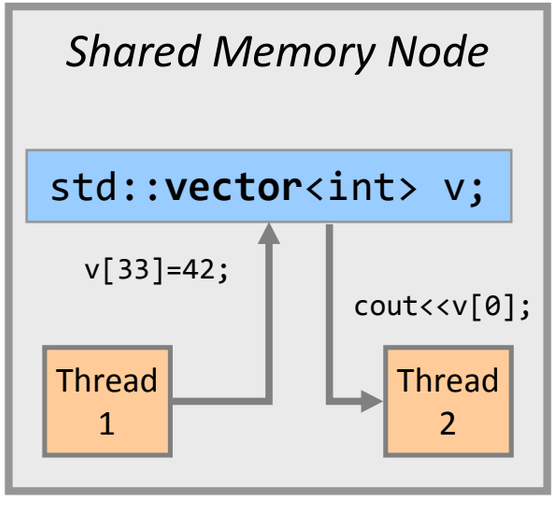
Image sources: [1,2]

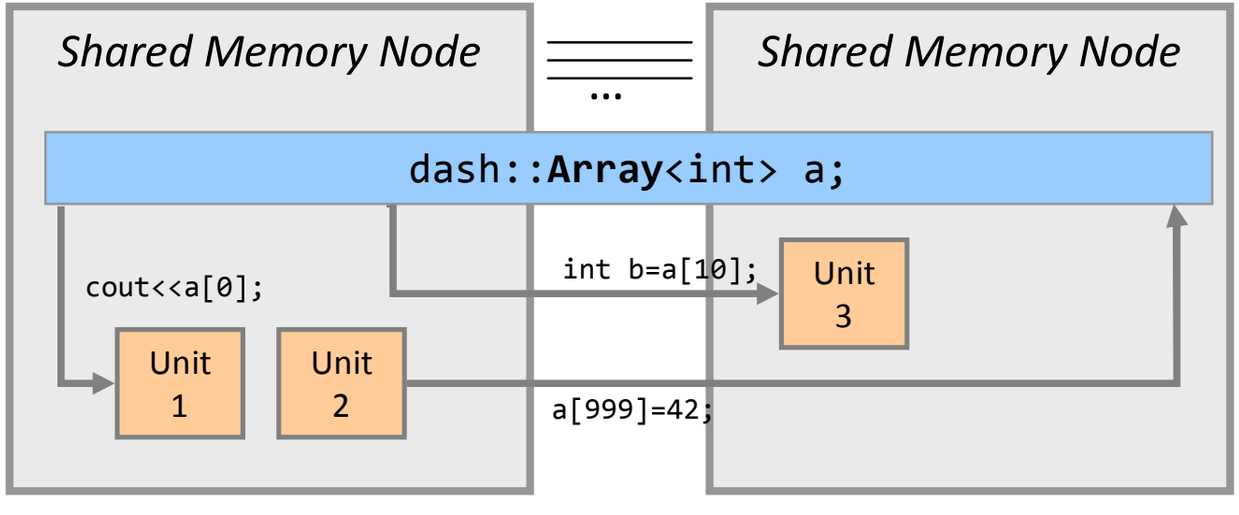
- Approaches
  - Specialized packages, libraries, frameworks,…
  - Domain specific languages (DSLs)
  - Still need **general parallel programming approaches** for the cases that are not supported well by the above
  - For these, **productivity** is as important as performance!

[1] http://www.medicalive.it/probiotici-aiuto-le-allergie-soprattutto-nei-bambini/
[2] http://www.kecl.ntt.co.jp/as/members/tatsushi/GALLERY/GRAPH/vsLGL.html

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:



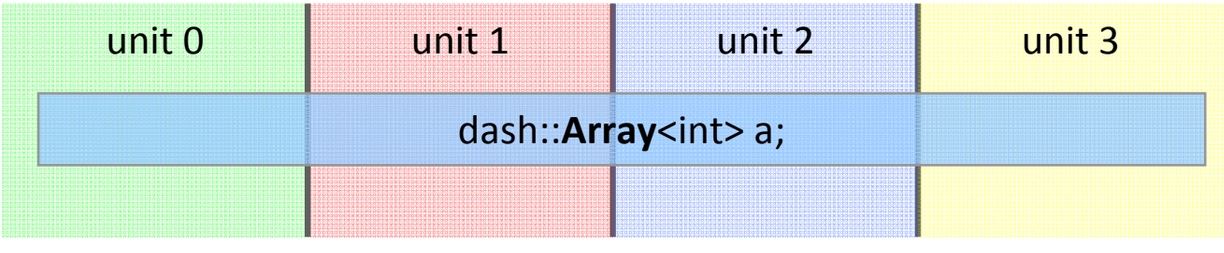


- Multiple threads access **physically** shared memory
- Multiple nodes connected by a high-speed network
- Multiple threads ("units") access **logically** shared memory

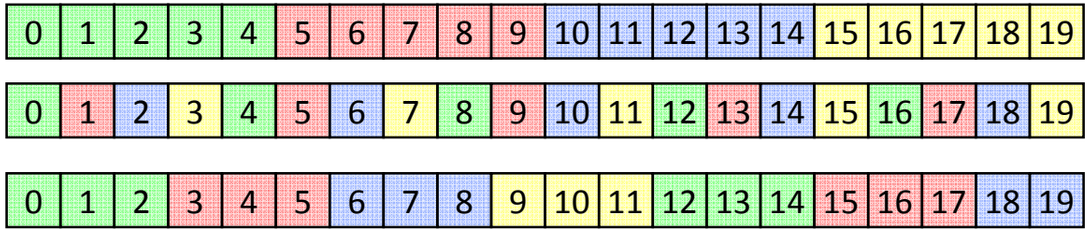
DASH - A PGAS Programming System

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

- Example



- Example Data Distribution Patterns (1D)

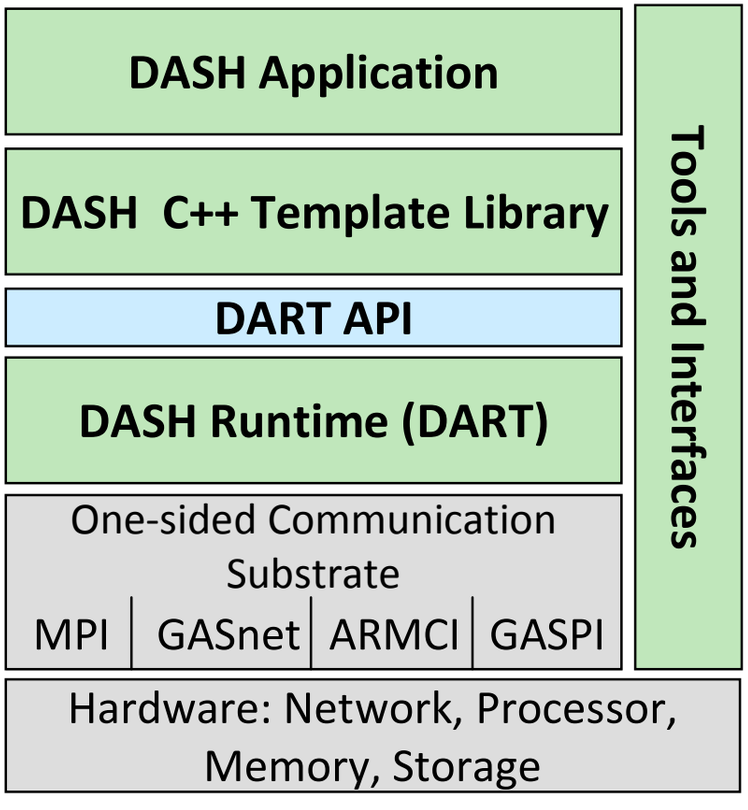


BLOCKED

CYCLIC

BLOCKCYCLIC(3)

DASH – Project Overview



	Phase I (2013-2015)	Phase II (2016-2018)
LMU Munich	Project management, C++ template library	Project management, C++ template library, DASH data dock
TU Dresden	Libraries and interfaces, tools support	Smart data structures, resilience
HLRS Stuttgart	DART runtime	DART runtime
KIT Karlsruhe	Application case studies	
IHR Stuttgart		Smart deployment, Application case studies

dash

www.dash-project.org

DFG

SPPEXA

DASH is one of 16 SPPEXA projects

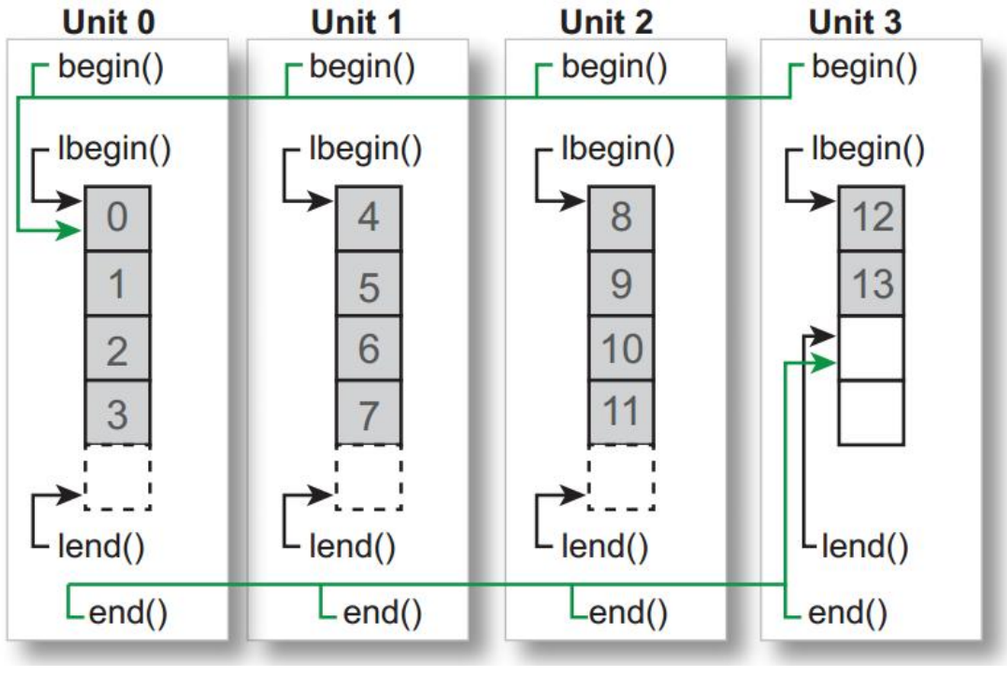
French/Japanese Collaborations

- 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)

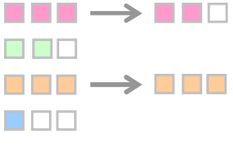
- Global data structures
  - Avoid the need for manual index calculations, especially for multidimensional data structures
  - Both **globalview** semantics and **localview** semantics supported

```
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

Container	Description	Data distribution
Shared<T>	Shared Scalar	
Array<T>	1D Dist. Array	
NArray<T, N>	N-dim. Dist. Array	
Coarray<T[R][S]>	CAF-like Coarray	
List(*)<T>, **Map**(*)<T>	Dynamic data structures (growing/shrinking)	

(*) Under Development

Performance and Productivity Features (2)

- STL compatibility and parallel algorithms

```
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

[1] Wilson, Gregory V., and R. Bruce Irvin. *"Assessing and comparing the usability of parallel programming systems."* University of Toronto. Computer Systems Research Institute, 1995.

[2] Nanz, Sebastian, Scott West, Kaue Soares Da Silveira, and Bertrand Meyer. *"Benchmarking usability and performance of multicore languages."* In Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on, pp. 183-192. IEEE, 2013.

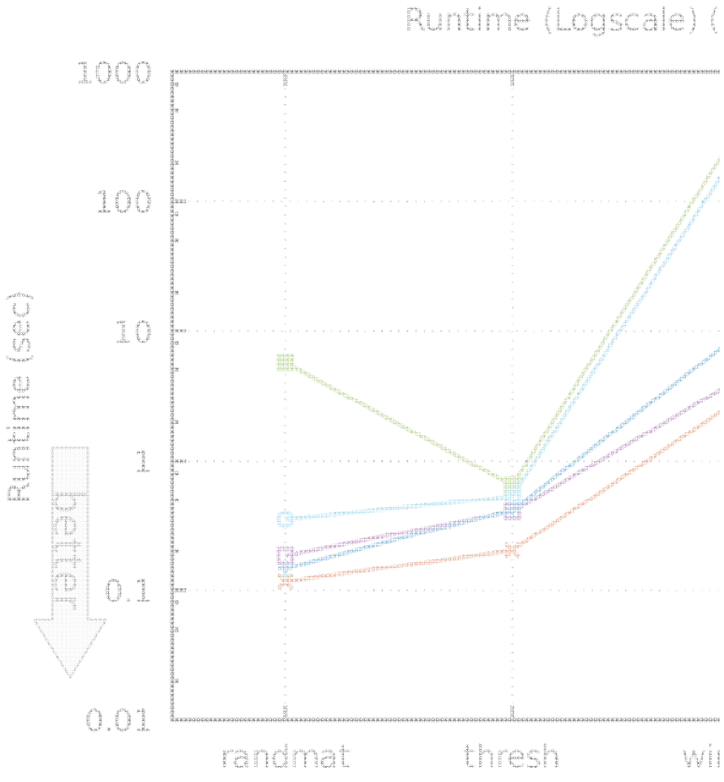
Cowichan Results – Lines of Code

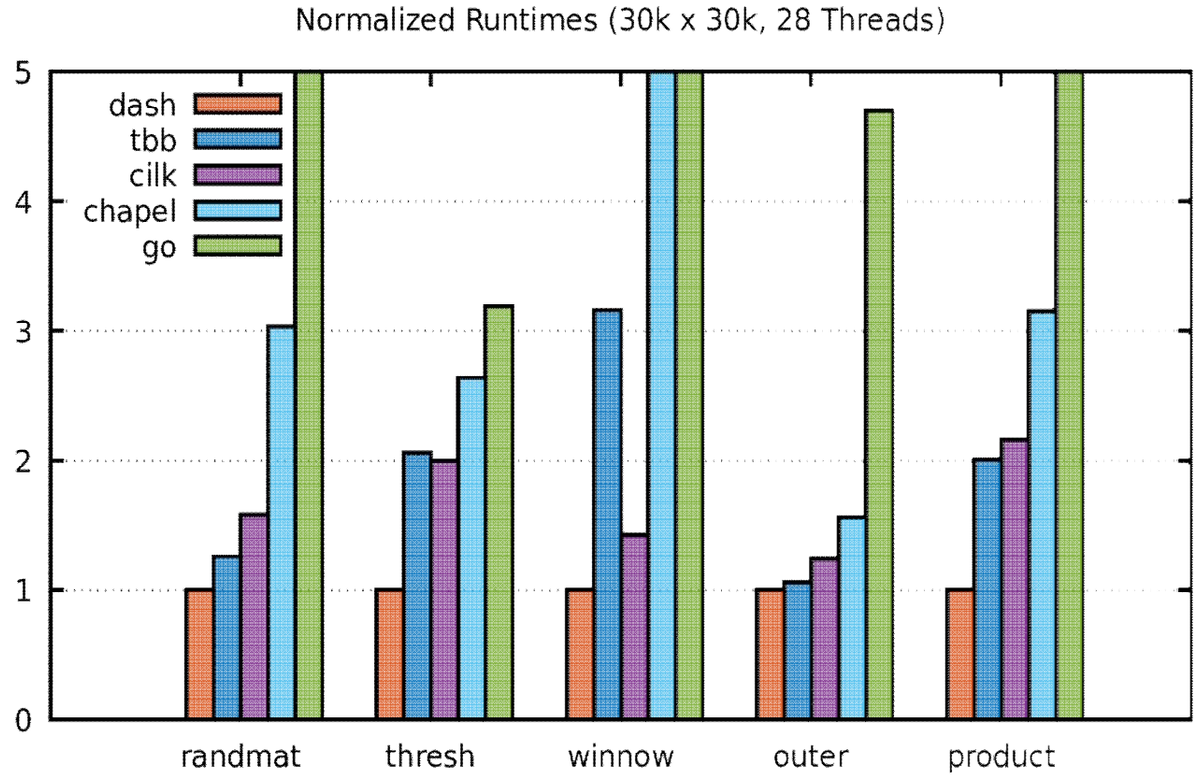
	DASH	go	Chapel	TBB	Cilk
randmat	16	29	14	15	12
thresh	31	63	30	56	52
winnow	53	94	31	74	78
outer	23	38	15	19	15
product	20	27	11	14	10

- 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!

Cowichan Results – Shared Memory (1)

- 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



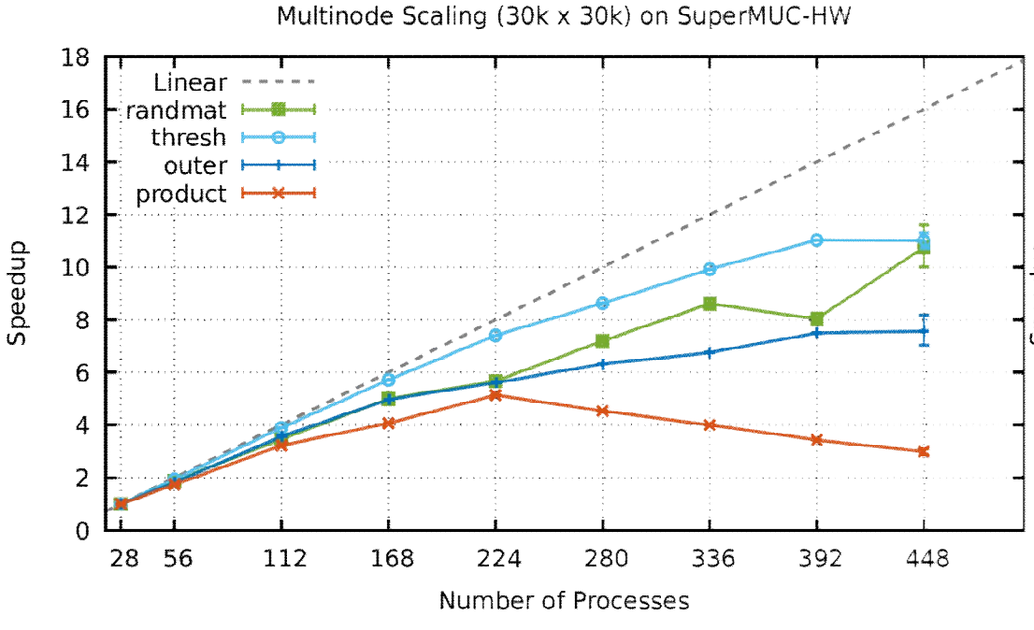


Absolute performance, using all 28 cores, 30k x 30k Matrix

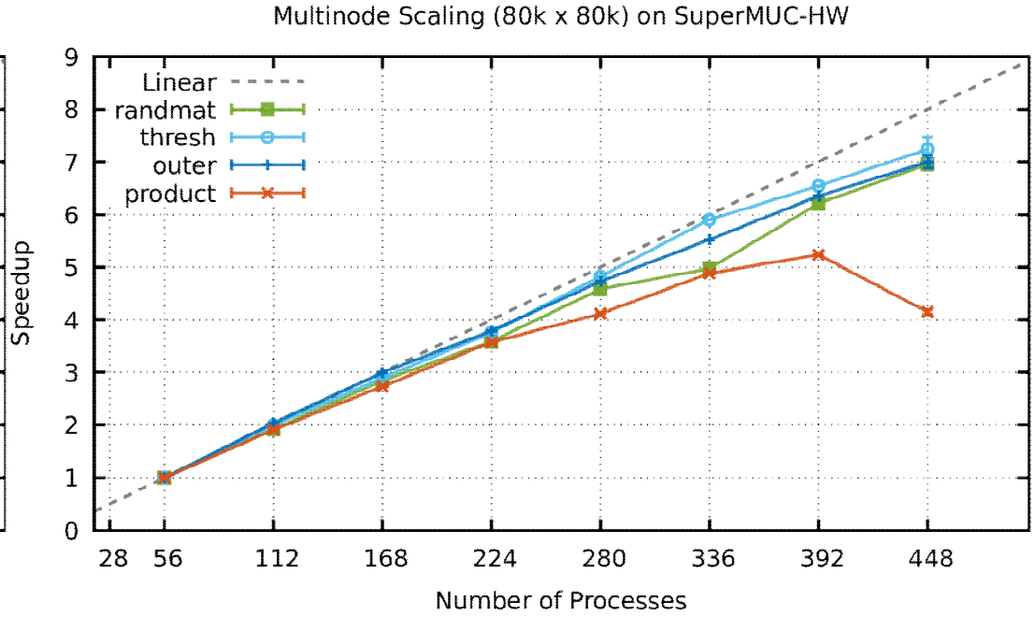
Perf. relative to DASH, using all 28 cores, 30k x 30k Matrix

Cowichan Multinode Scaling and Summary

- 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**)



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

Funding: Bundesministerium für Bildung und Forschung

DFG Deutsche Forschungsgemeinschaft

Github: https://github.com/dash-project/dash/
Web: http://www.dash-project.org

- 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)