LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Smart-DASH Update and Status

SPPEXA Annual Plenary Meeting 2019
22-23 January 2019, Garching

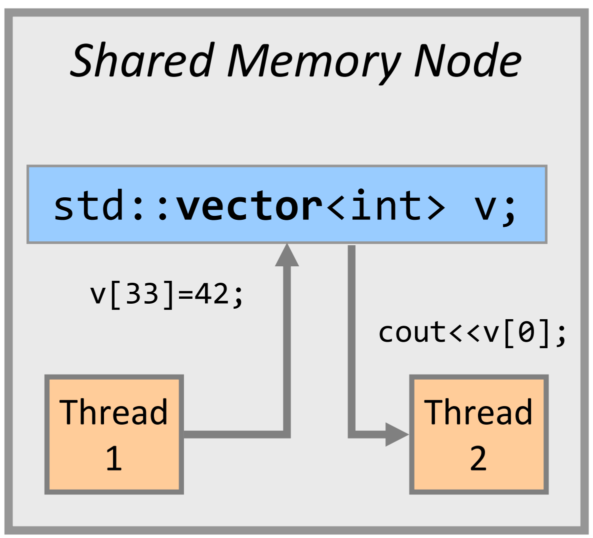
dash

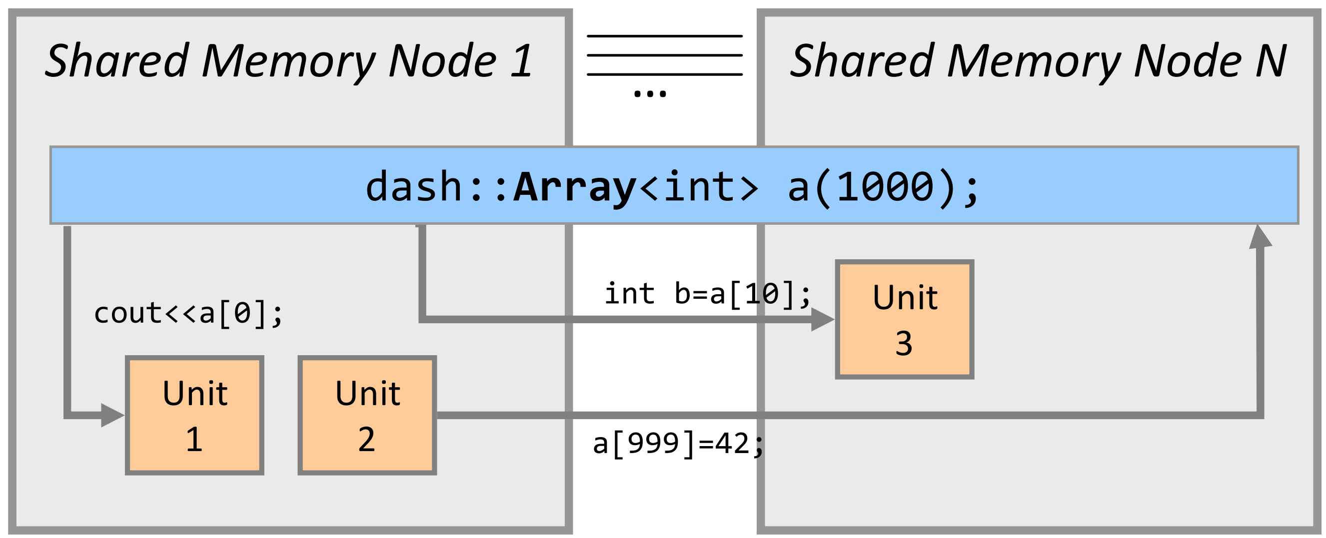
www.dash-project.org

Karl Fürlinger
Ludwig-Maximilians-Universität München

DASH - Overview

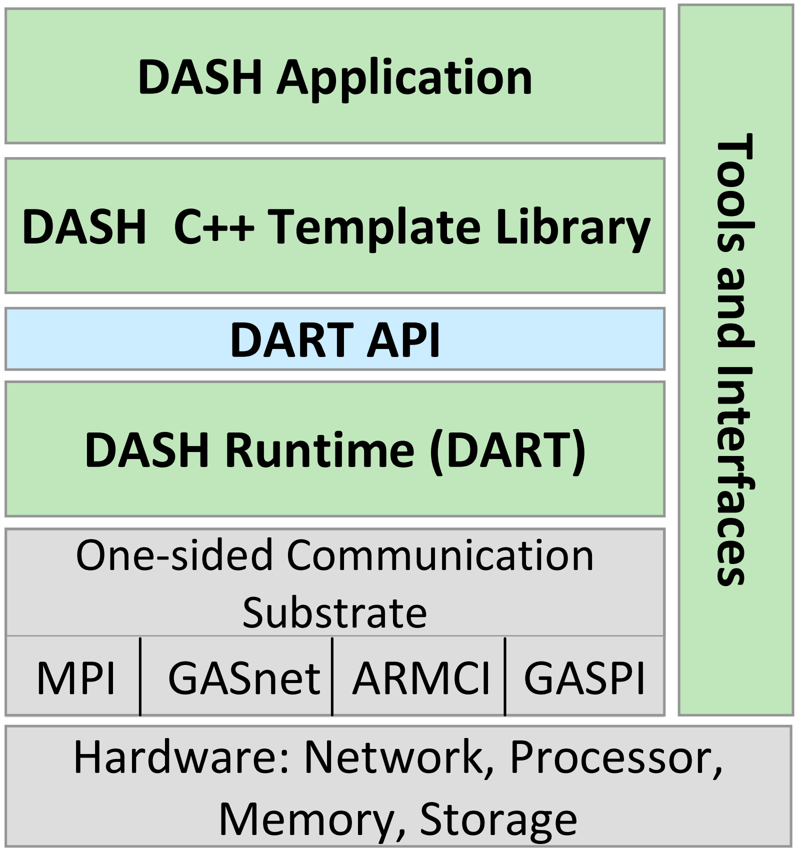
- DASH is a C++ template library, that offers
  - Distributed data structures, e.g., dash::**Array**<int>
  - Parallel algorithms, e.g., dash::**sort**()
- Generalizes shared memory programming to distributed memory systems:





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

DASH – 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, Tasking
KIT Karlsruhe	Application case studies	
IHR Stuttgart		Smart deployment, Application case studies

dash

www.dash-project.org

DASH is one of 16 SPPEXA projects

Smart-DASH – Recent Activities

- Performance and Productivity Evaluation
- DASH Algorithms – Distributed Parallel Sort
- DASH and task-based execution

Performance and Productivity

Case Study: The Cowichan Problem Suite

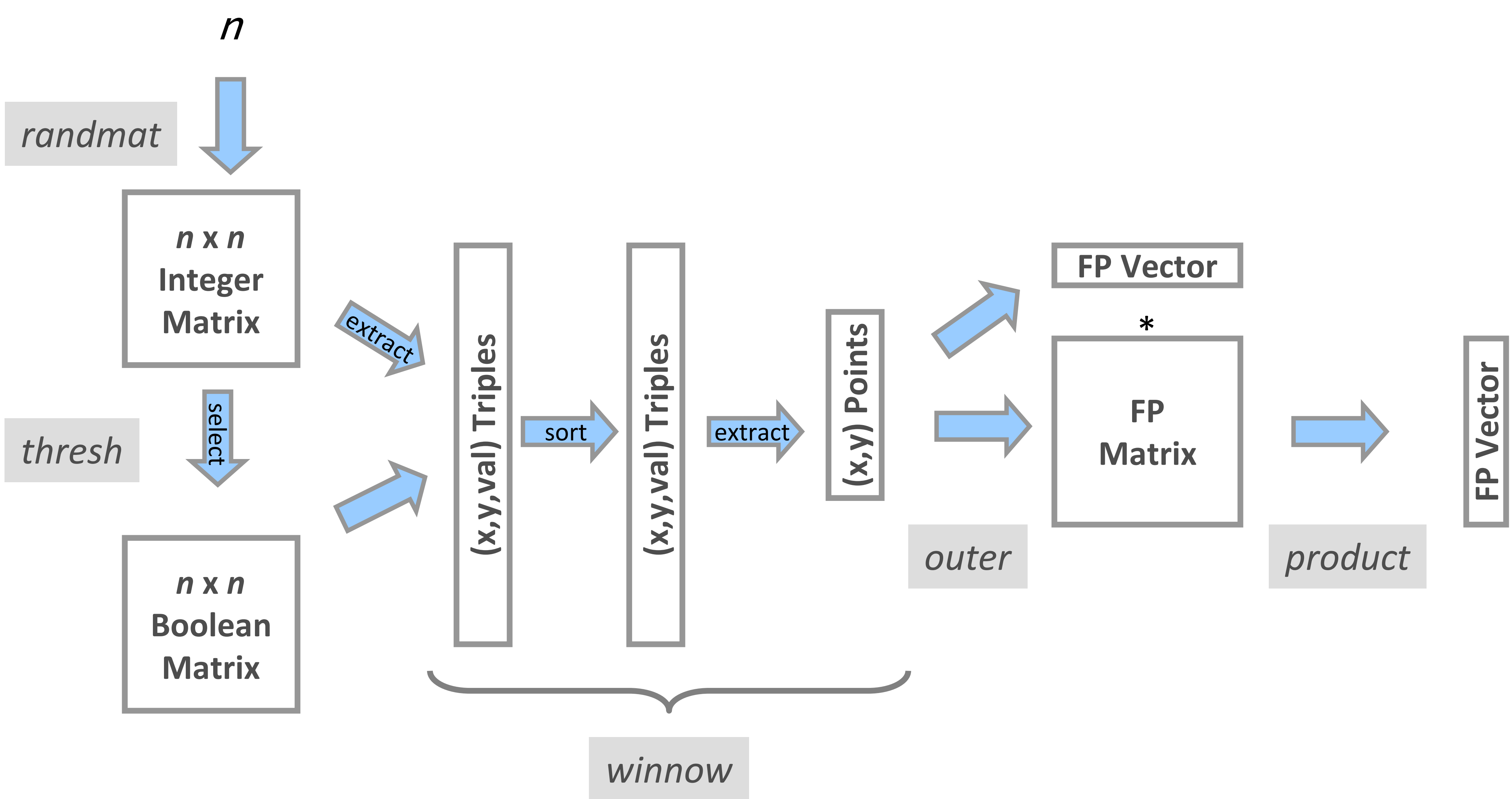
- Cowichan problems
  - A **benchmark suite** designed to investigate the usability of parallel programming systems (1990s)
  - 13 “toy” problems, quick implementation, composable by chaining [1]

- 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.

Case Study: Data Structures and Algorithms



Cowichan Results – Lines of Code

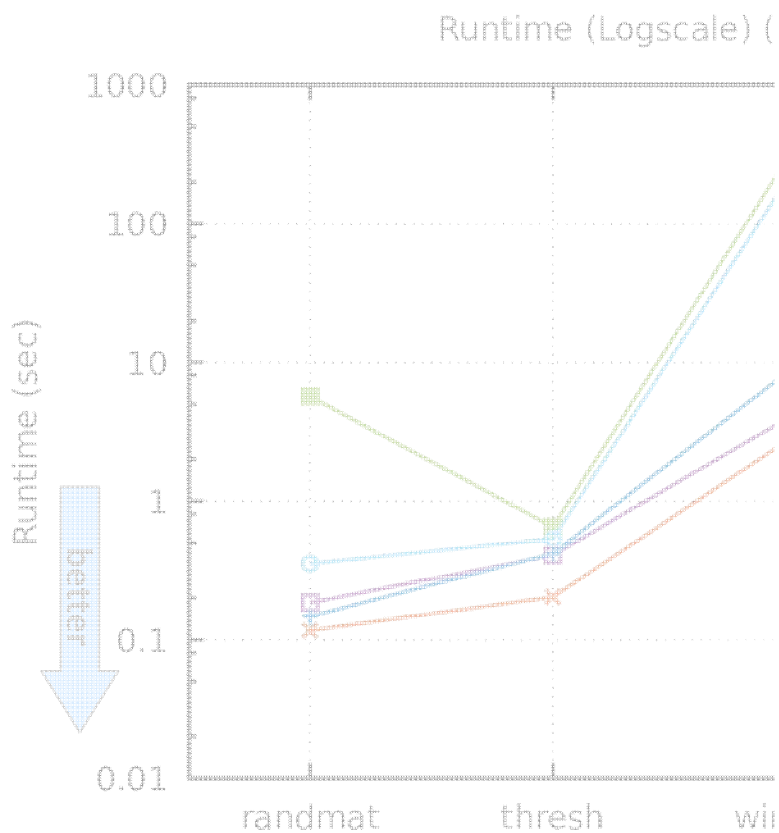
	DASH	go	Chapel	TBB	Cilk
randmat	16	29	14	15	12
thresh	31	63	30	56	52
winnow	53	94	31	74	78
outer	23	38	15	19	15
product	20	27	11	14	10

- 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!

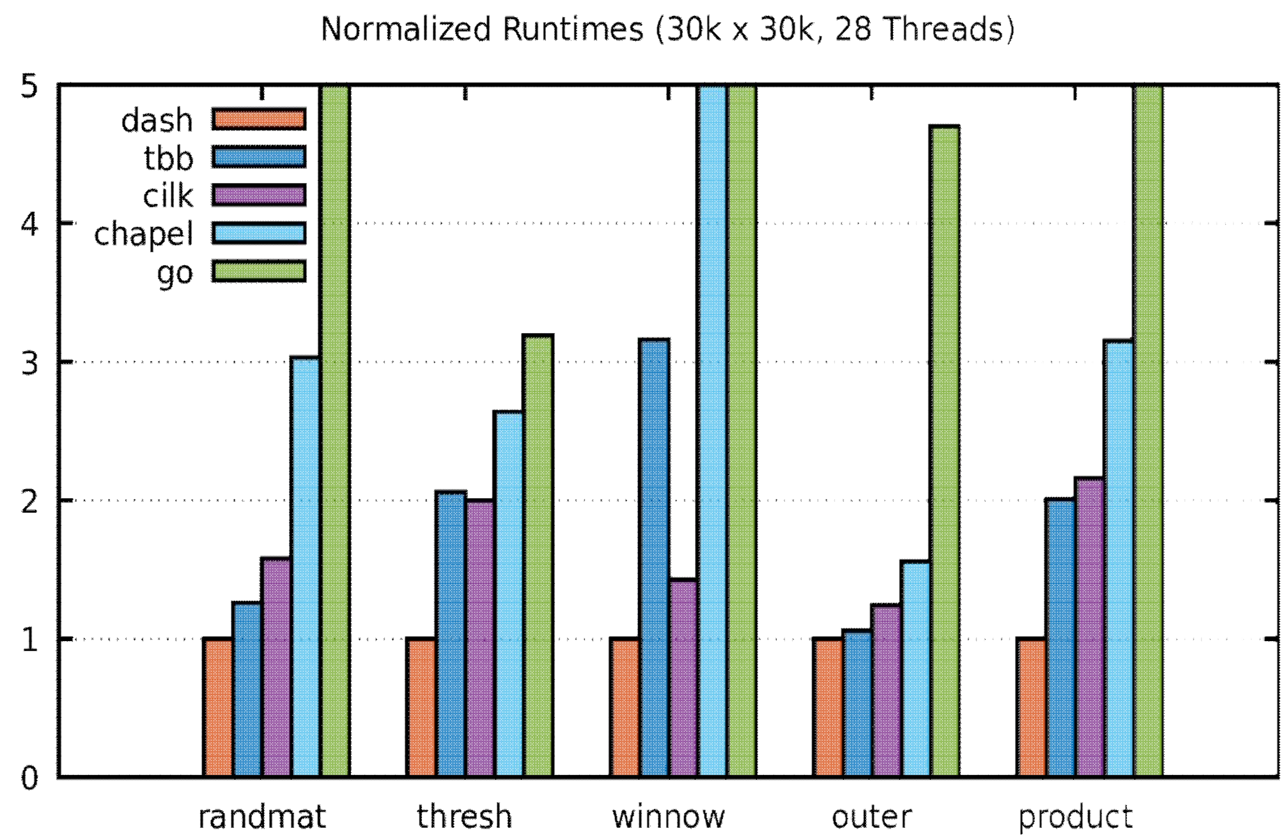
"Investigating the Performance and Productivity of DASH Using the Cowichan Problems", K. Fürlinger, R. Kowalewski, T. Fuchs, and B. Lehmann; Proc. of the International Conference on High Performance Computing in Asia-Pacific Region, Tokyo Jan. 2018

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 mem
  - 30k x 30k matrix
  - Intel Compiler (icc) v. 18.0.2 used for all programming systems



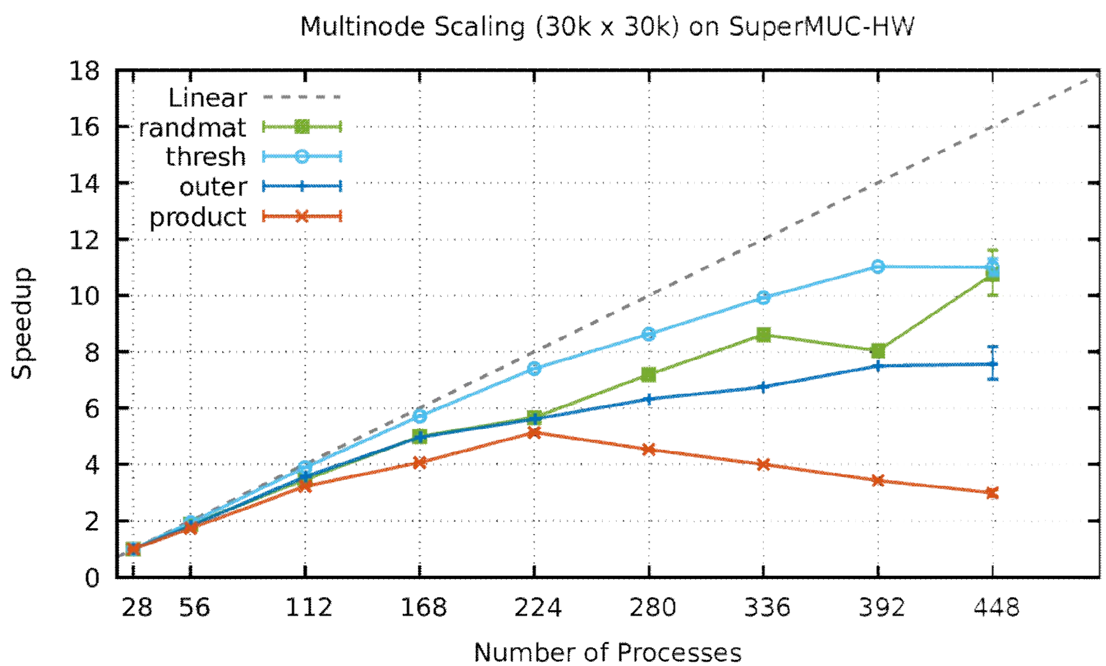
Absolute performance, using all 28 cores, 30k x 30k Matrix



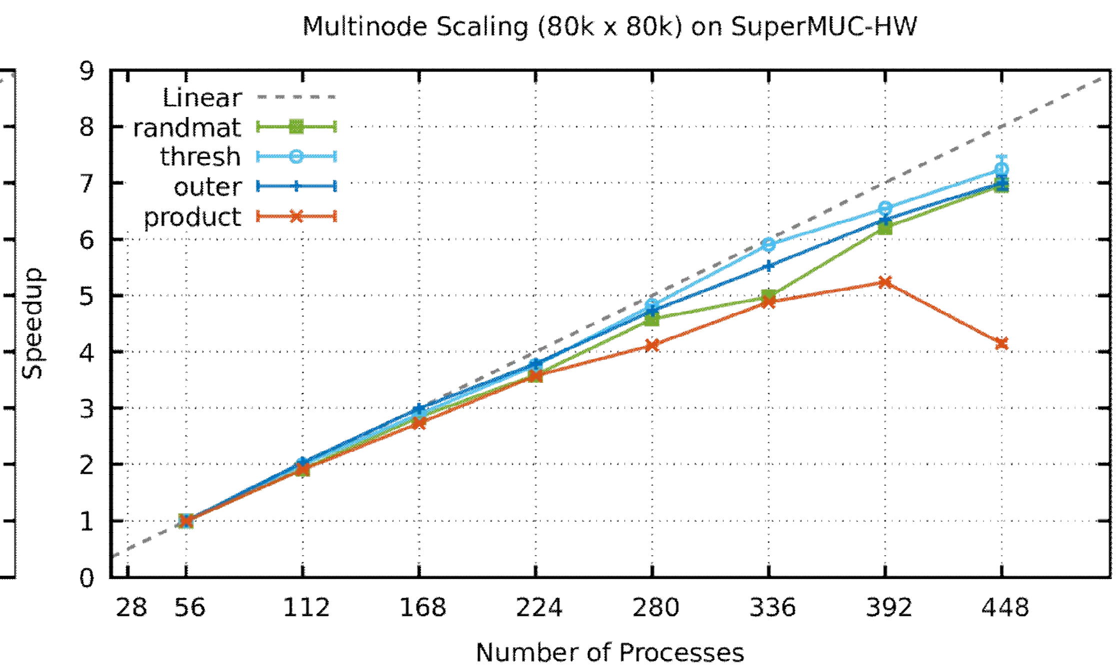
Performance relative to DASH, using all 28 cores, 30k x 30k Matrix

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**)



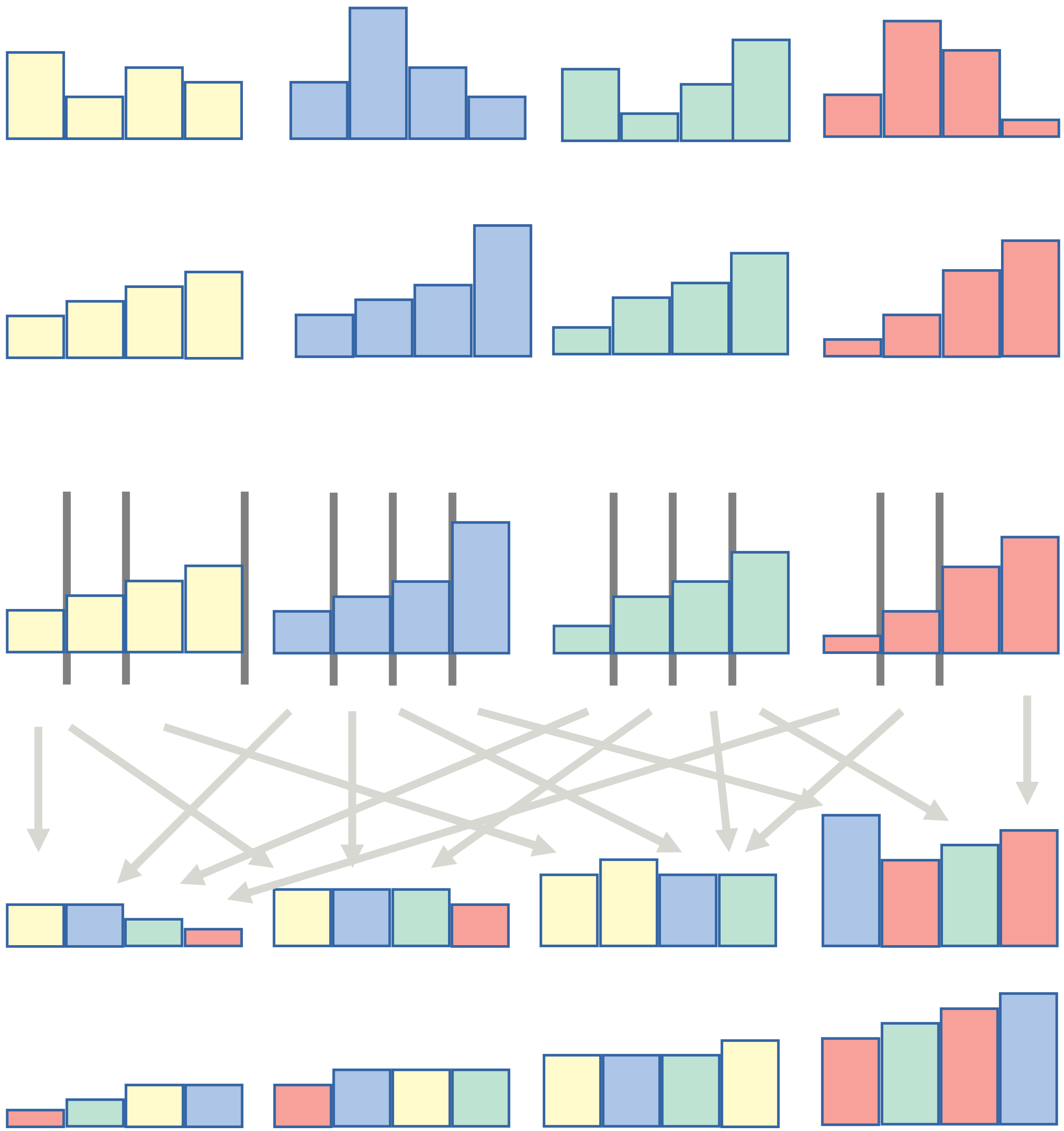
30k x 30k, Speedup vs 1 node



80k x 80k, Speedup vs 2 nodes

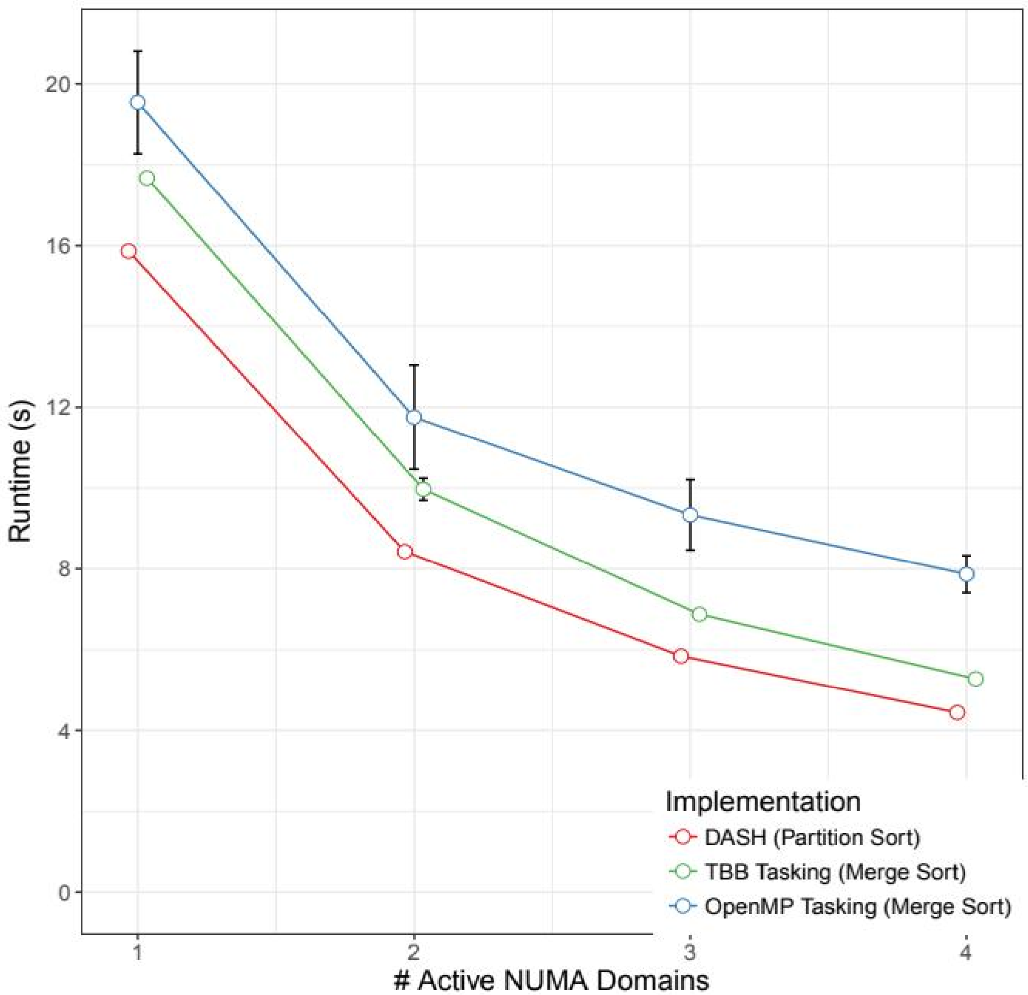
Sort

Partition-based Sort



Sorting – Shared Memory

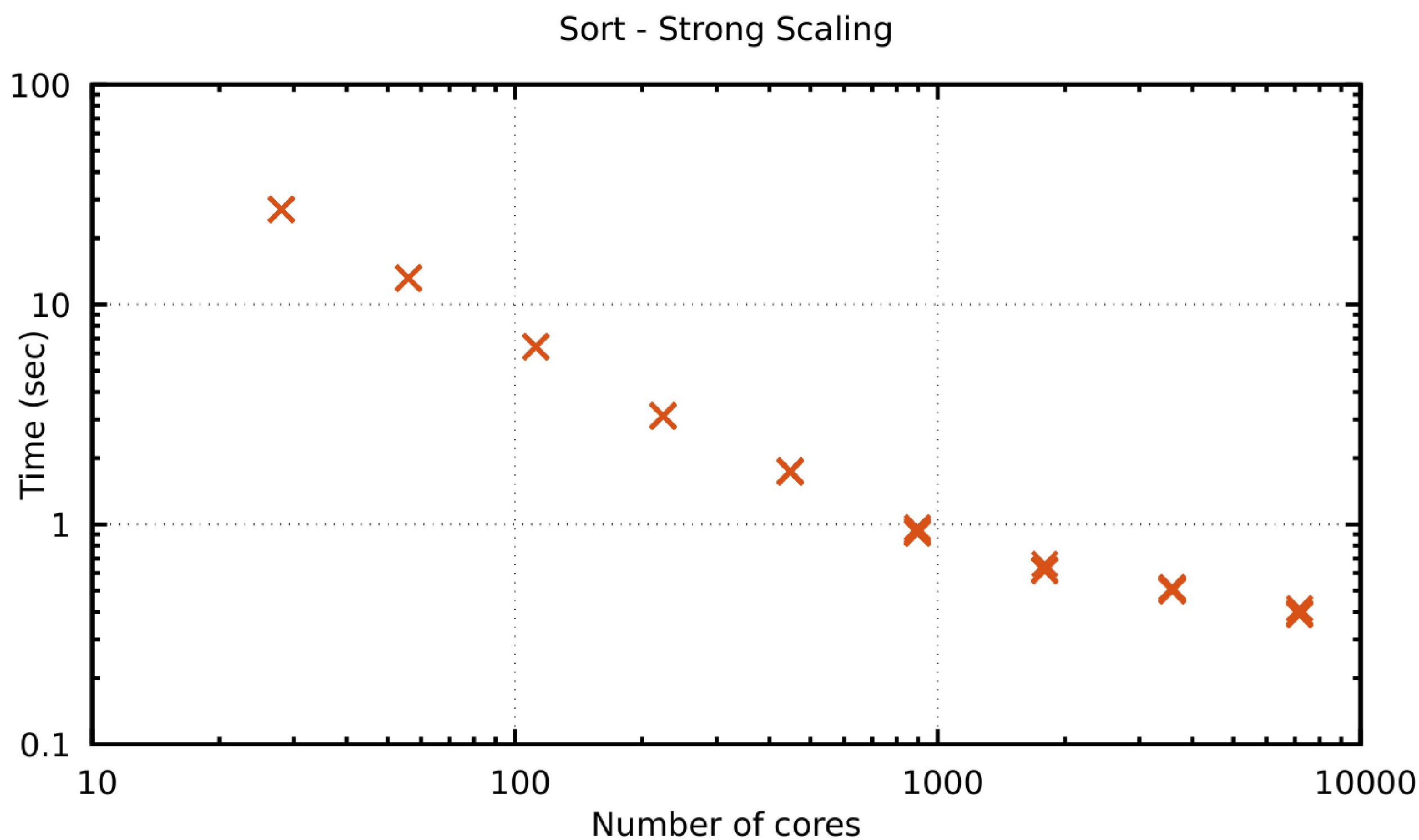
- Platform: 1 Node of SuperMUC Phase 2
  - 2 sockets, 28 cores, 4 NUMA domains
  - Sorting 4900 MB of data (int)



- PGAS approach beneficial because data locality is primary concern
- Partition-based sort algorithm moves data only once

Sorting – Distributed Memory

- Strong scaling on SuperMUC (HW)
  - Sorting ~19 GB data (DP numbers)



"Engineering a Scalable Histogram Sort", R. Kowalewski, P. Jungblut, K. Fürlinger, upcoming publication.

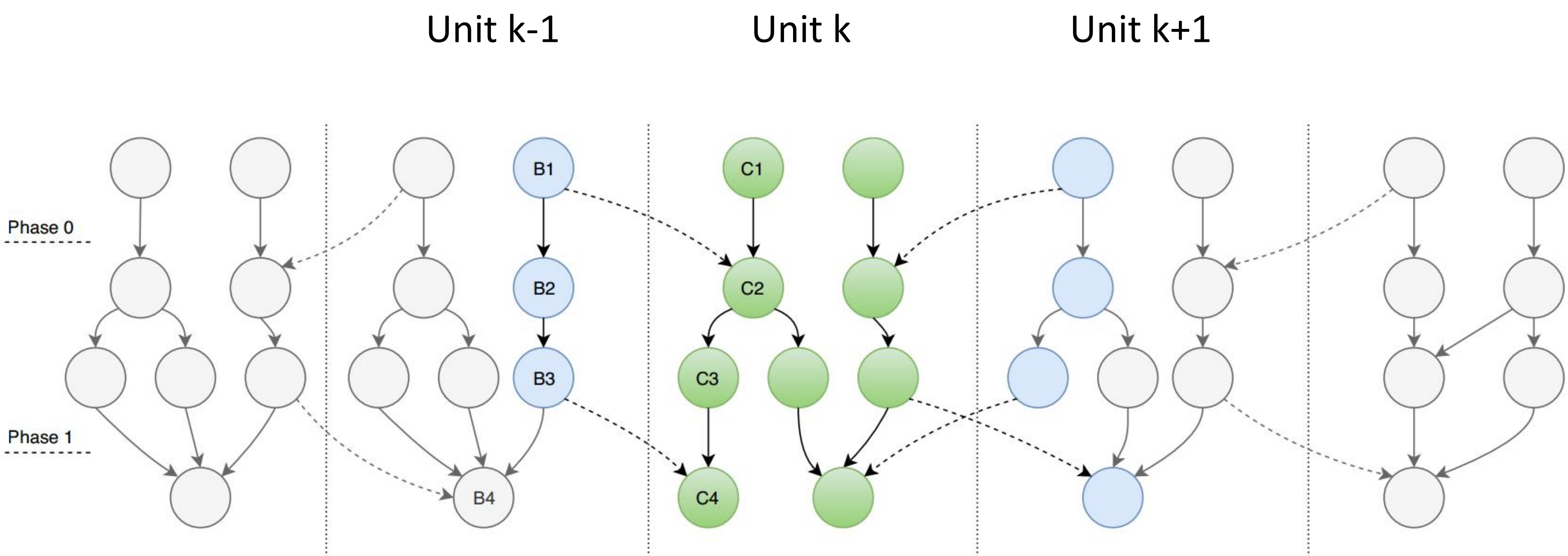
Tasking

Tasking in DASH

- The DASH tasking model is inspired by OpenMP, but can express dependencies in the global address space

OpenMP Tasks	DASH Tasks
#pragma omp task	dash::async(…) w/ lambda and dependency spec.
Data dependencies in **node-local memory only**	Data dependencies can **span global address space** (multiple nodes)
One scheduler instance	N distributed scheduler instances
Dependencies between **previously generated** sibling tasks	Dependencies between sibling tasks in **previous phase** on any node
in / out / inout dependencies	in / out / copyin dependencies

DASH Tasks – Phase Model



DASH Tasking - Cholesky Example

```
for (int k = 0; k < num_blocks; ++k) {
  if (mat.block(k,k).is_local()) {
    dash::async([&](){ potrf(matrix.block(k,k)); },
      dash::out(mat.block(k,k)));
  }

  // advance to next phase
  dash::async_barrier();
  for (int i = k+1; i < num_blocks; ++i)
    if (mat.block(k,i).is_local())
      dash::async([&](){
          trsm(cache[k][k], matrix.block(k,i)); },
        dash::copyin(mat.block(k,k), cache[k][k]),
        dash::out(mat.block(k,i)));

  // advance to next phase
  dash::async_barrier();
  for (int i = k+1; i < num_blocks; ++i) {
    for (int j = k+1; j < i; ++j) {
      if (mat.block(j,i).is_local()) {
        dash::async([&](){
            gemm(cache[k][i],
                 cache[k][j], mat.block(j,i)); },
          dash::copyin(mat.block(k,i), cache[k][i]),
          dash::copyin(mat.block(k,j), cache[k][j]),
          dash::out(mat.block(j,i)));
      }
    }

    if (mat.block(i,i).is_local()) {
      dash::async([&](){
          syrk(cache[k][i], mat.block(i,i)); },
        dash::copyin(mat.block(k,i), cache[k][i]),
        dash::out(mat.block(i,i)));
    }
  }
  // advance to next phase
  dash::async_barrier();
}
// wait for all tasks to execute
dash::complete();
```

potrf

sh::async() de
sk using C++ l
expressions

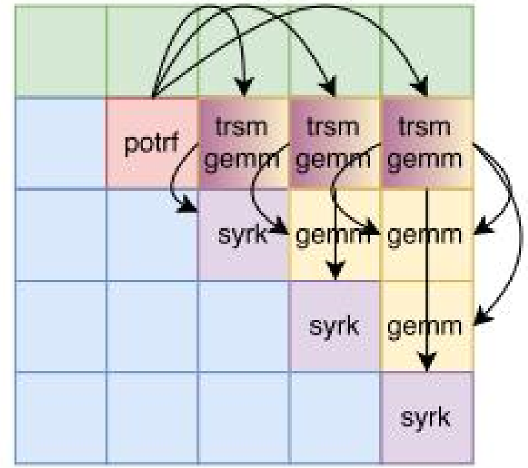
trsm

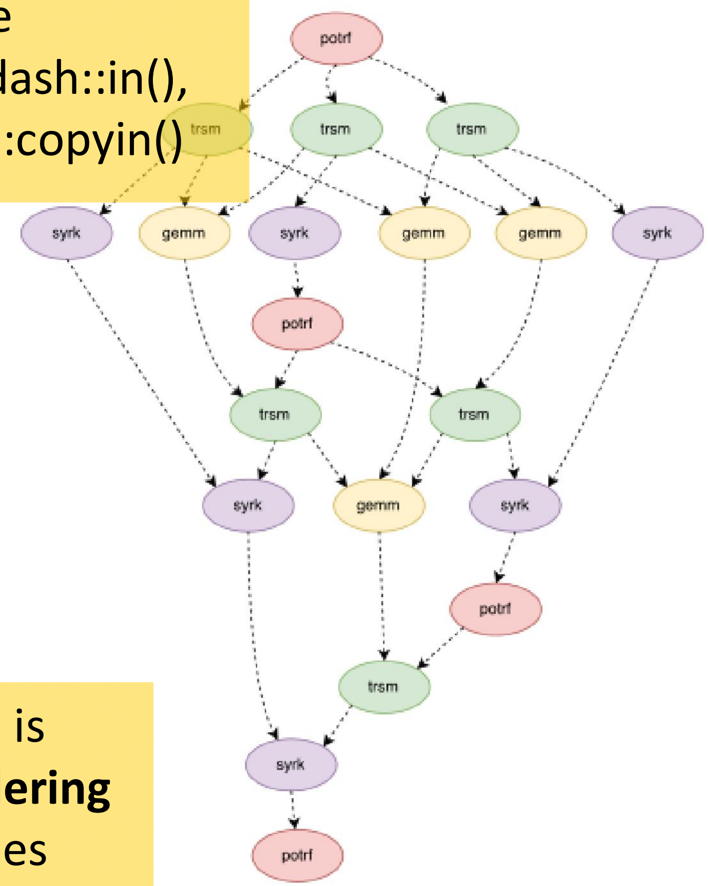
dependencies are expressed using dash::in(), dash::out(), dash::copyin()

gemm

syrk

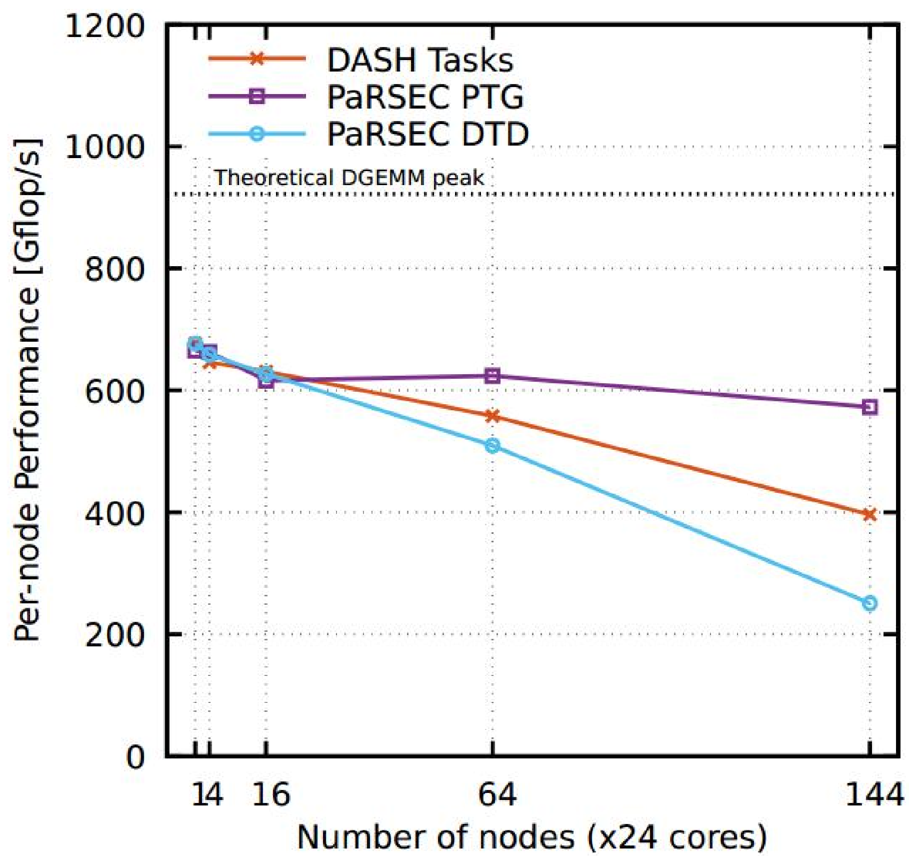
dash::async_barrier() is used to **establish ordering** between dependencies



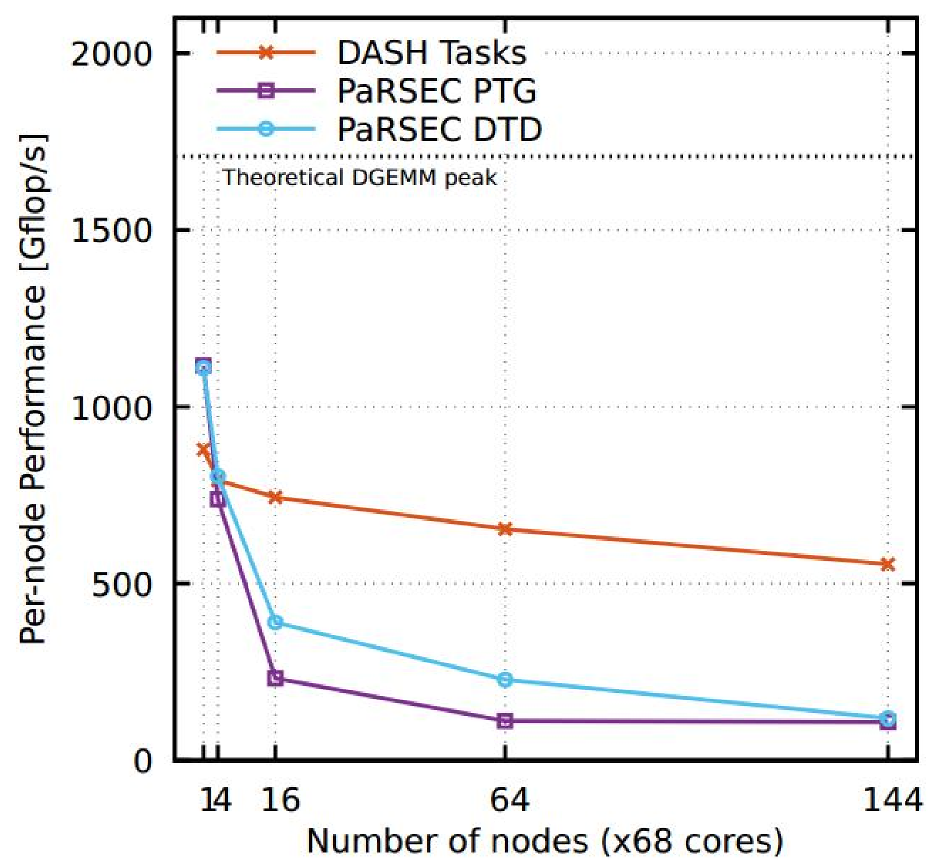


Tasking - Cholesky Results

- Weak scaling study (20k x 20k matrix per node)
  - Compares with PaRSEC (http://icl.utk.edu/parsec/)



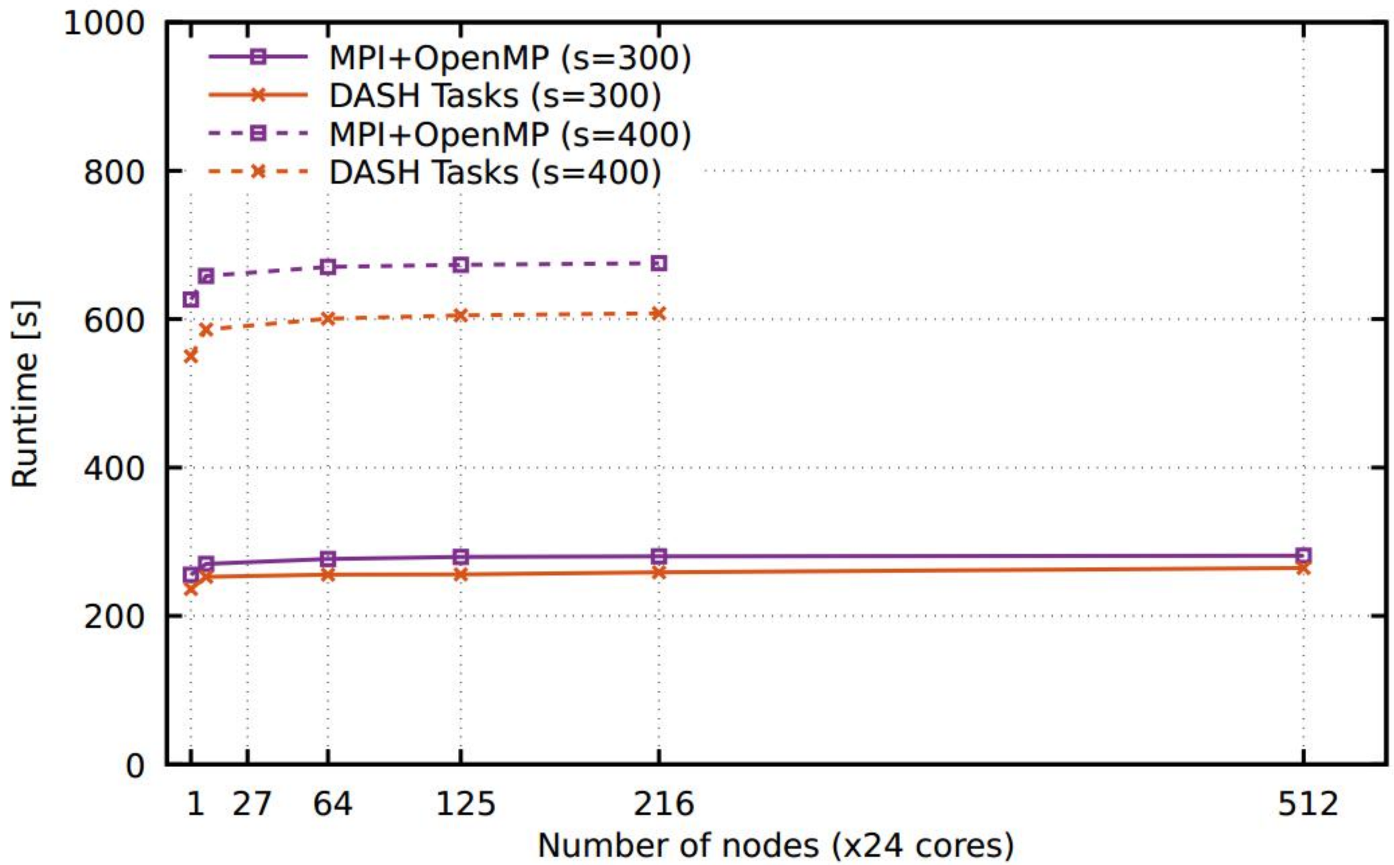
Cray XC40 (Intel SandyBridge)



Oakforest-PACS (KNL)

Tasking - LULESH Results

- Weak scaling on Cray XC40 (Intel SandyBridge)



"Global Task Data-Dependencies in PGAS Applications", Joseph Schuchart and Jose Gracia, upcoming publication.

Collaborations (Inside and Outside of SPPEXA)

- Collaboration with the **MYX** and **ESSEX-II** projects: Workshops on "*Parallel Programming Models - Productivity and Applications*" and "*Programming and Computing for Exascale and Beyond*"
  - Aachen (15 March, 2018) and Tokyo (30 October, 2018)
  - Plans to use **DART** as a use-case for the **MUST** correctness checking tool

- BMBF Project **MEPHISTO** (02/2017 – 01/2020)
  - TU Dresden (lead) + HZDR + LMU Munich
  - Integration of ALPAKA (portable compute kernel abstraction) with DASH (data structure abstraction)

Acknowledgements

- Funding

DFG Deutsche Forschungsgemeinschaft

Bundesministerium für Bildung und Forschung

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

- DASH is on GitHub
  - https://github.com/dash-project/dash/