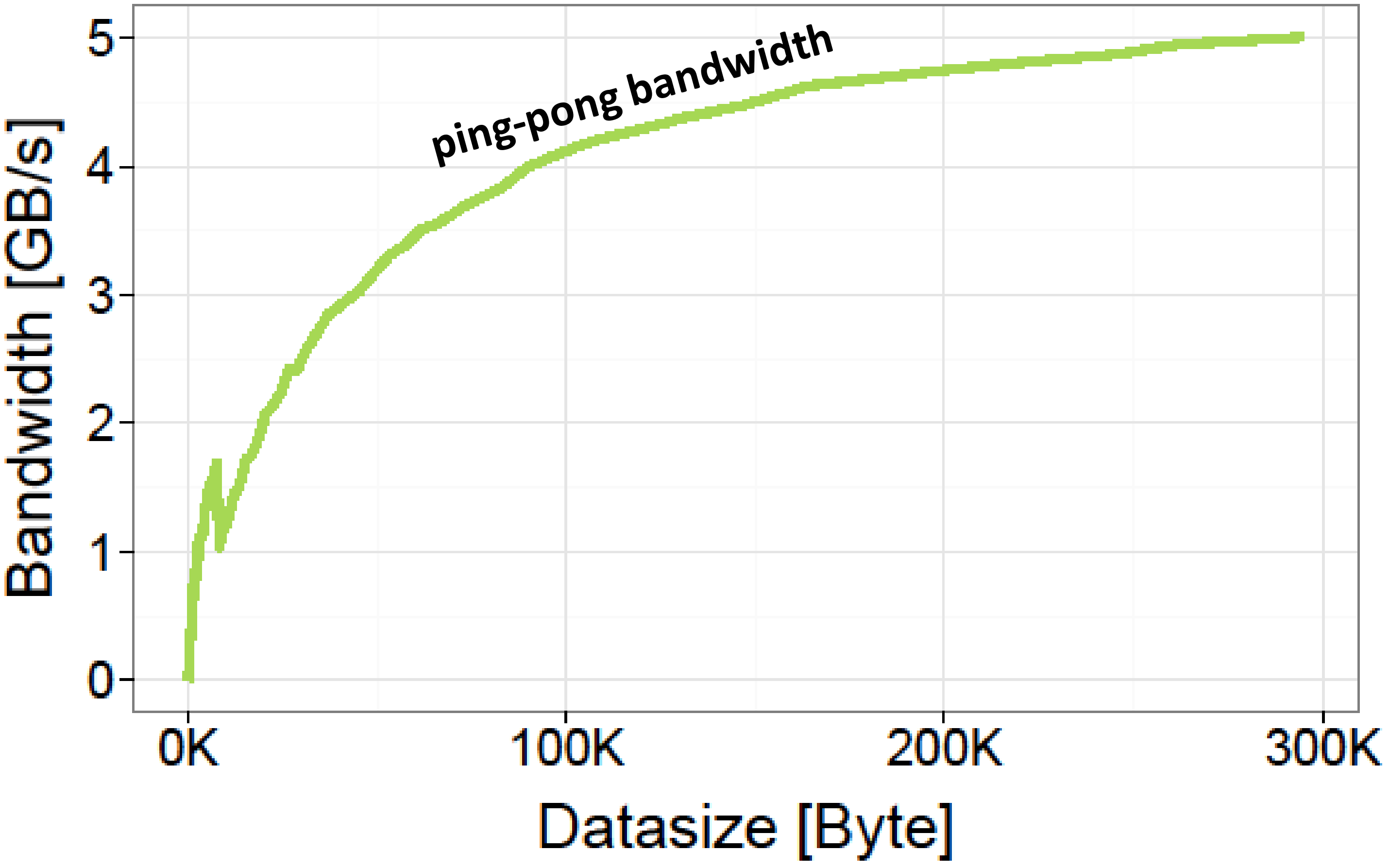
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

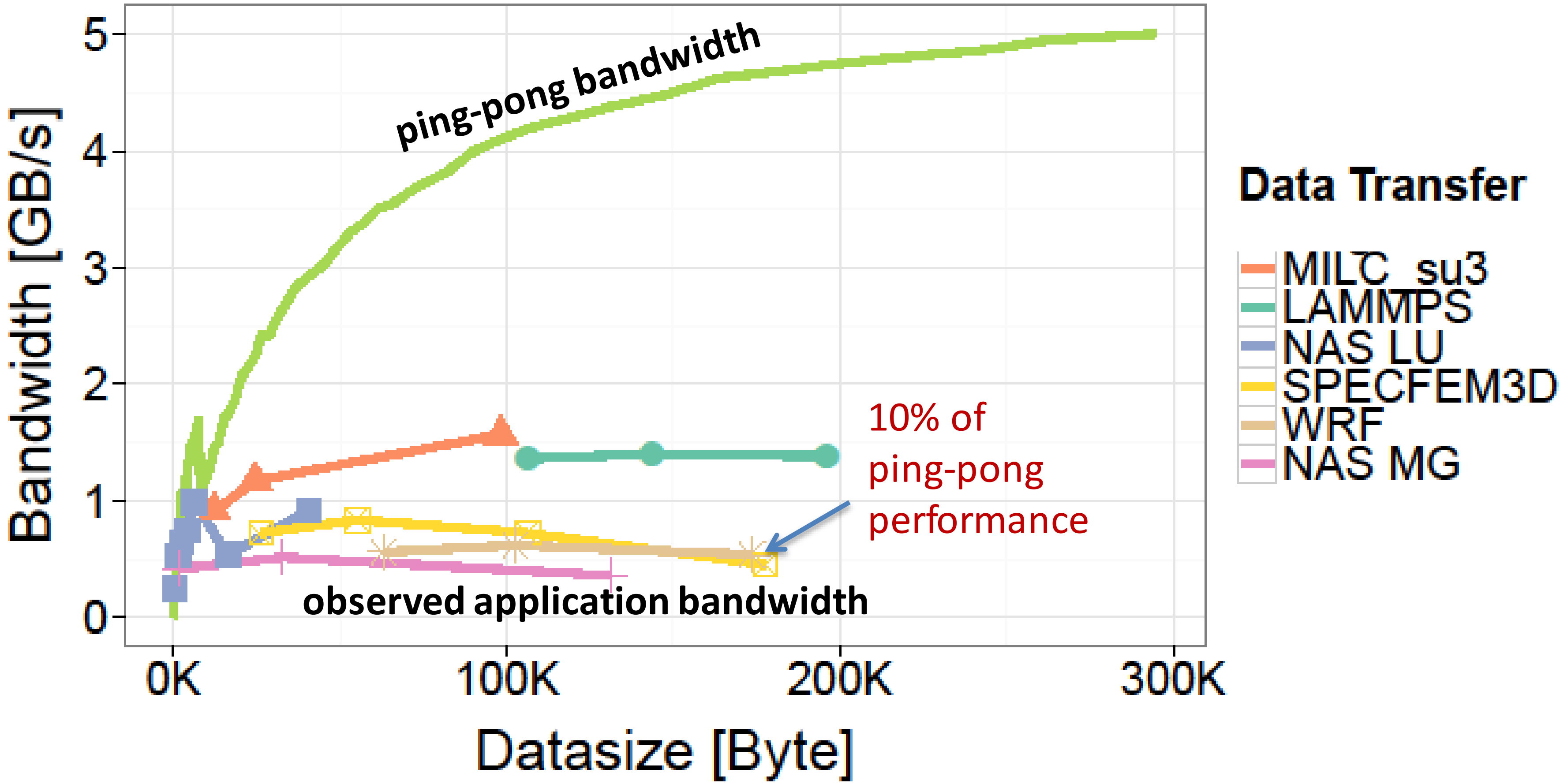
Compiler Optimizations for Non-Contiguous Remote Data Movement

Timo Schneider, Robert Gerstenberger, Torsten Hoefler

WHAT YOUR VENDOR SOLD YOU

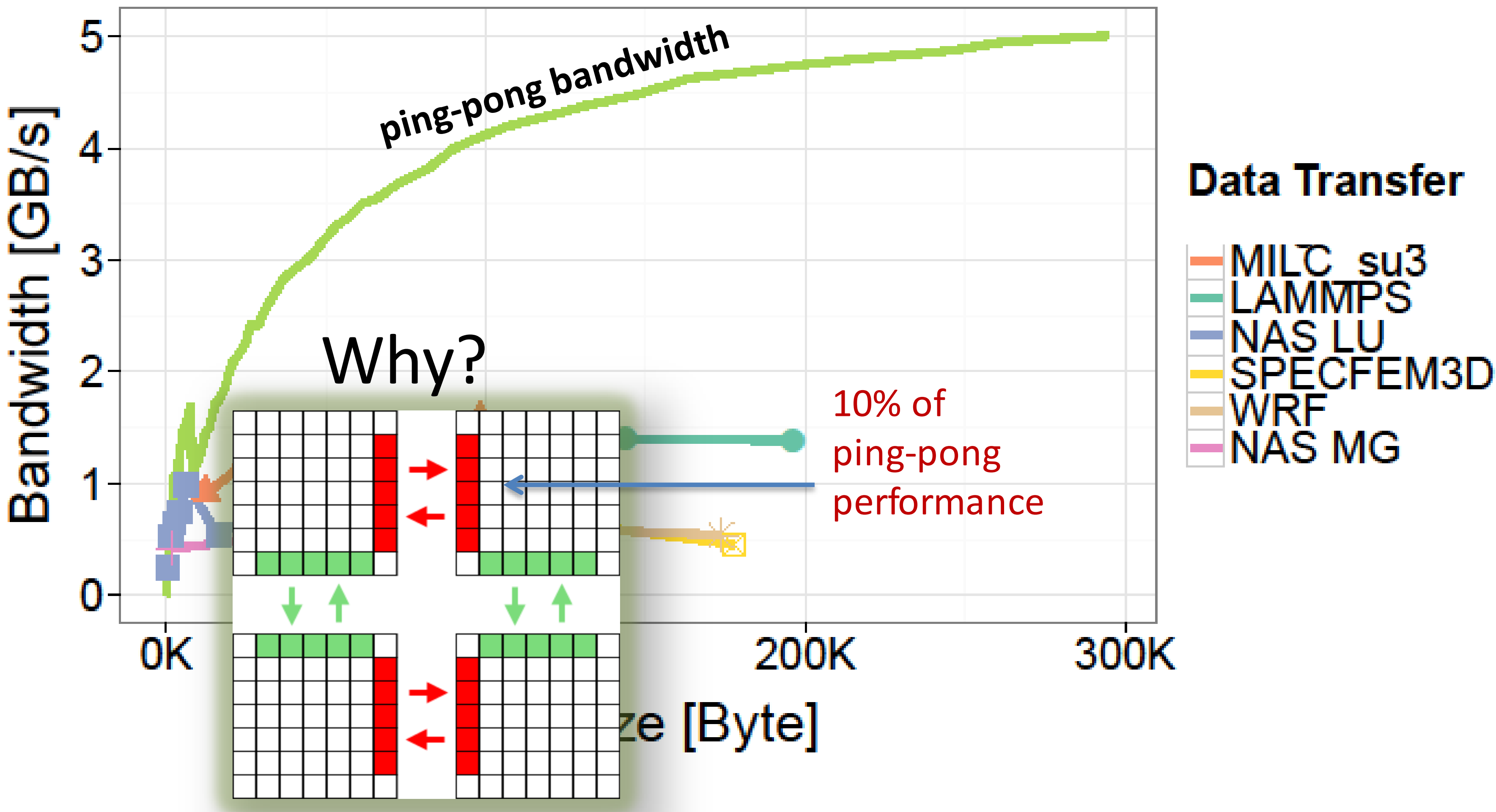


WHAT YOUR APPLICATIONS GET



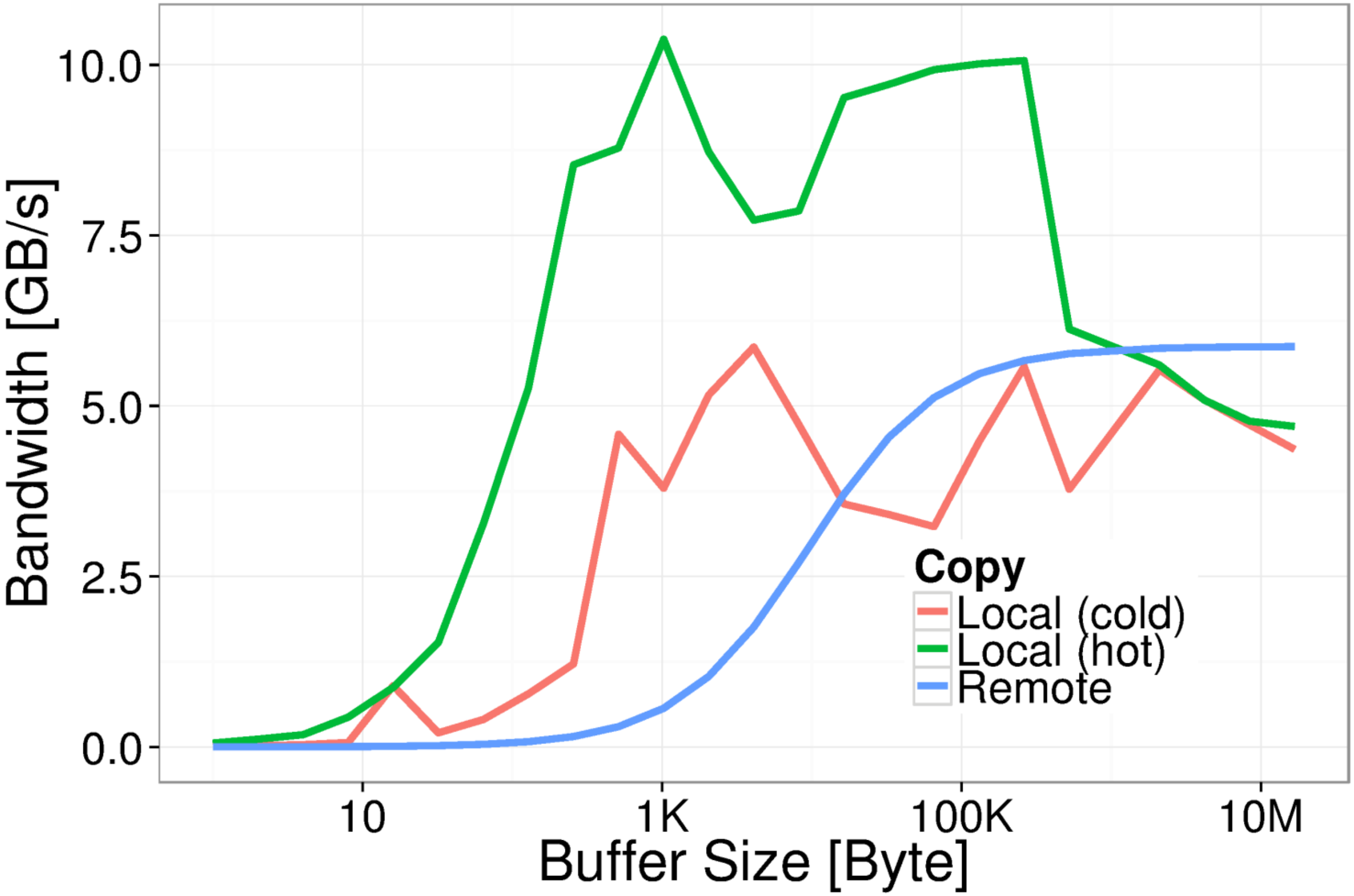
[1]: Schneider et al.: *"Application-oriented ping-pong benchmarking: how to assess the real communication overheads", Elsevier Computing*

WHAT YOUR APPLICATIONS GET



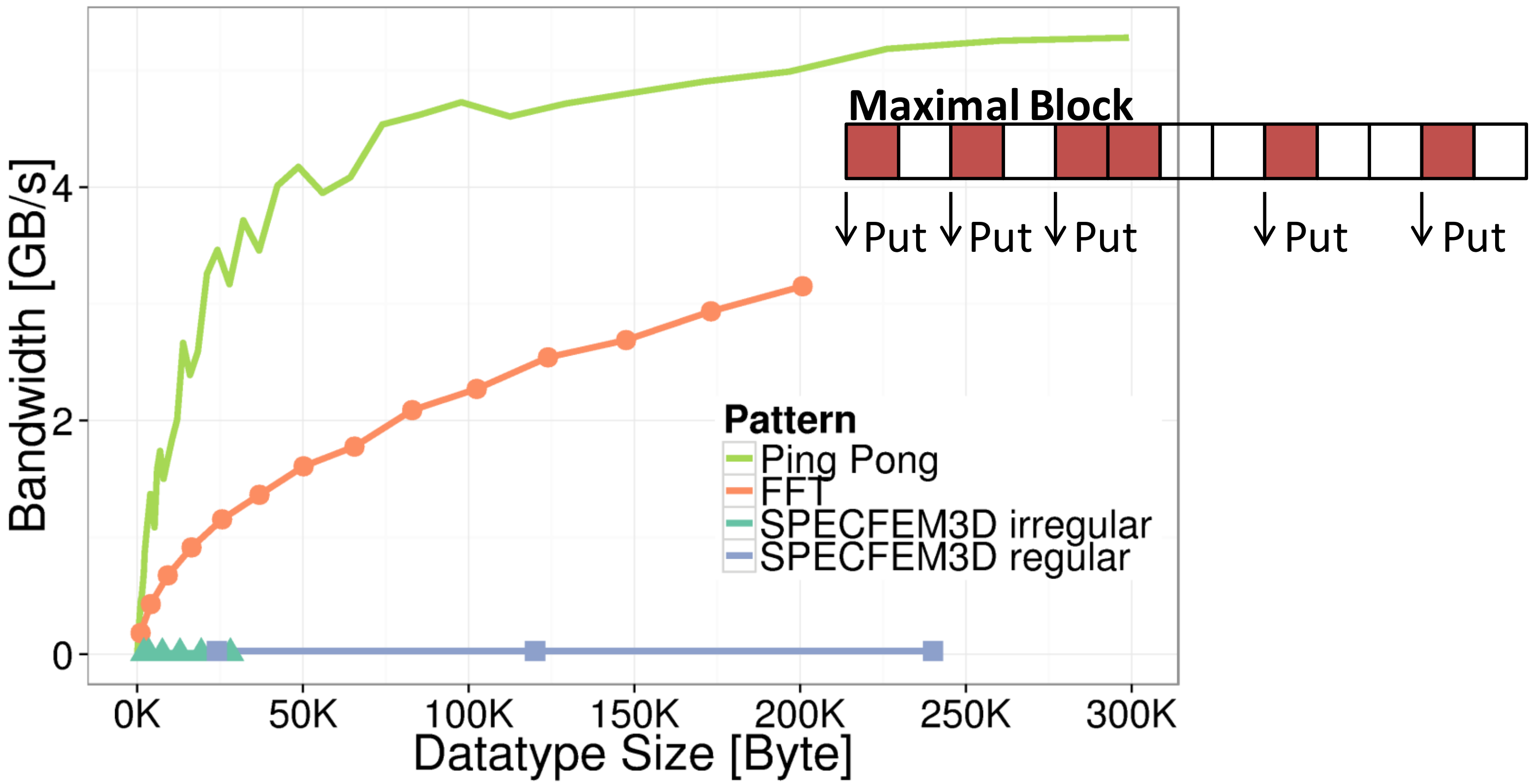
[1]: Schneider et al.: *"Application-oriented ping-pong benchmarking: how to assess the real communication overheads ", Elsevier Computing*

Local Copy vs. Remote Copy

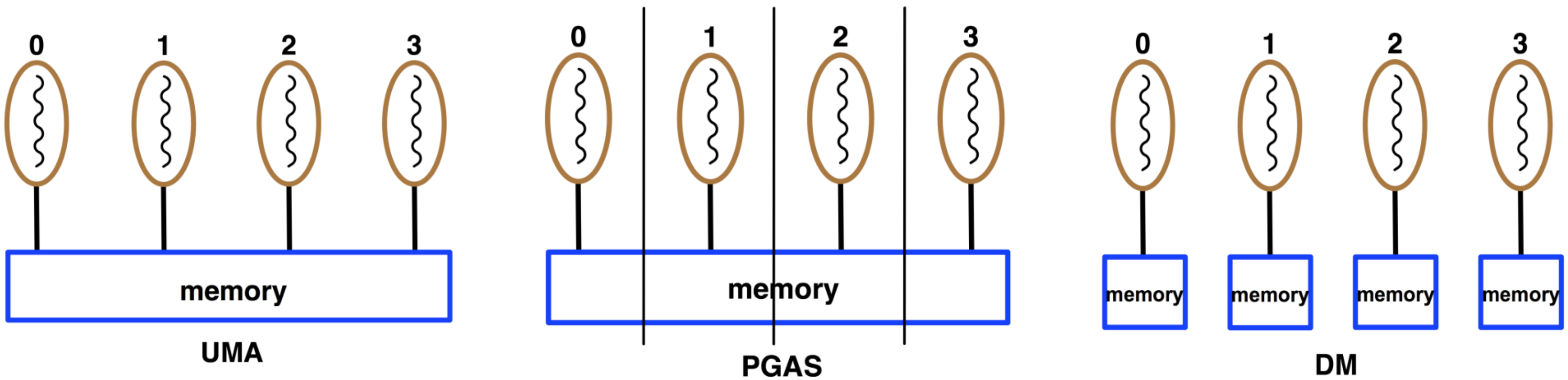


Put Maximum Contiguous Blocks?

- PGAS languages do not support datatypes
  - Send noncontiguous elements separately?



Shared Memory, PGAS, Message Passing



OpenMP, Pthreads, MPI-3, …	UPC, CAF, MPI One Sided, …	MPI Two Sided, PVM, …
origin specifies target address no involvement of target control flow		origin specifies channel target specifies address
simple compiler analysis and transformations		**static message matching**

→ One-Sided Control Flow!

Research Questions and Contributions

- Can we utilize the "one-sided control flow" to optimize non-contiguous communications?
  - During compilation, automatically?

Kjolstad et al. [2]: *"We have implemented the algorithm in a tool that transforms packing code to MPI Datatypes."*

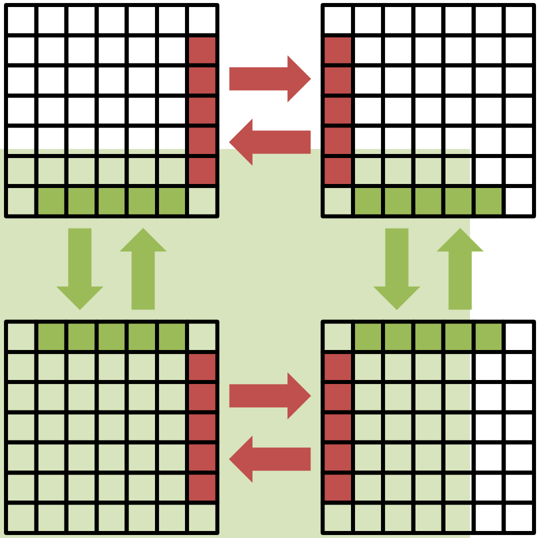
  - Backslice from communication to packing loop

    Or accumulate communications to the same target

  - Affine loops are easy to handle

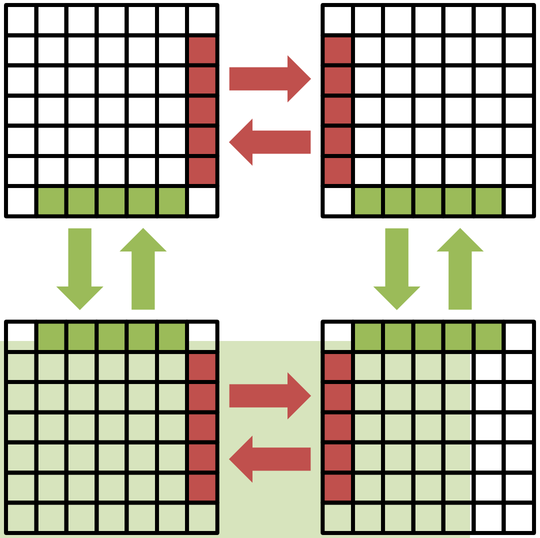
[2]: Kjolstad et al.: *"Automatic Datatype Generation and Optimization", PPoPP'12*

Example: Full Packing



```
for ( int iters=0; iters<niters; iters++) {
    compute_2d_stencil( array, ... );
    // swap arrays (omitted for brevity)
    for ( int i=0; i<bsize; ++i )
        sbufnorth[i] = array[i+1,1];
    // ... omitted south, east, and west pack loops
    RMA_Put( sbufnorth , rbufnorth , bsize , north );
    // ... omitted south , east , and west communications
    RMA_Fence();
    for ( int i =0; i<bsize; ++i ) array[i+1,0] = rbufnorth[i];
    // ... omitted south, east, and west unpack loops
}
```

Example: Maximal Blocks

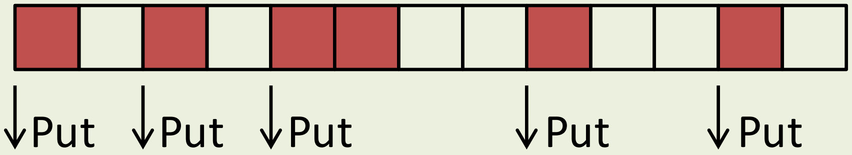


```
for ( int iter=0; iter<niters; ++iter ) {
    compute_2d_stencil( array, ... );
    // swap arrays ( omitted for brevity )
    for ( int i=0; i<bsize ; i++ ) {
        RMA_Put( array[i+1,1], array[i+1,0], size, north );
        // ... omitted south, east, and west communications
    }
    RMA_Fence( );
}
```

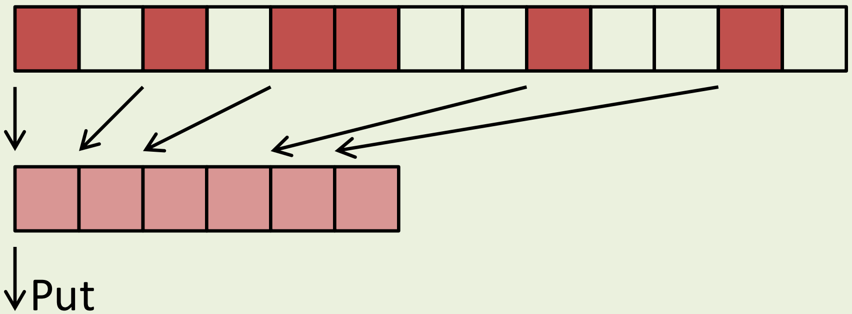
High-level Overview

Traditional Approaches

Maximal Block

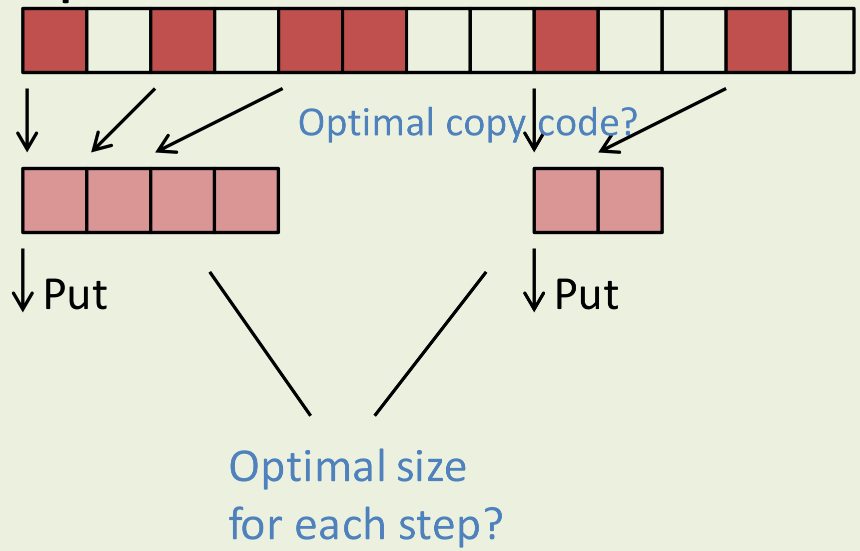


Pack First

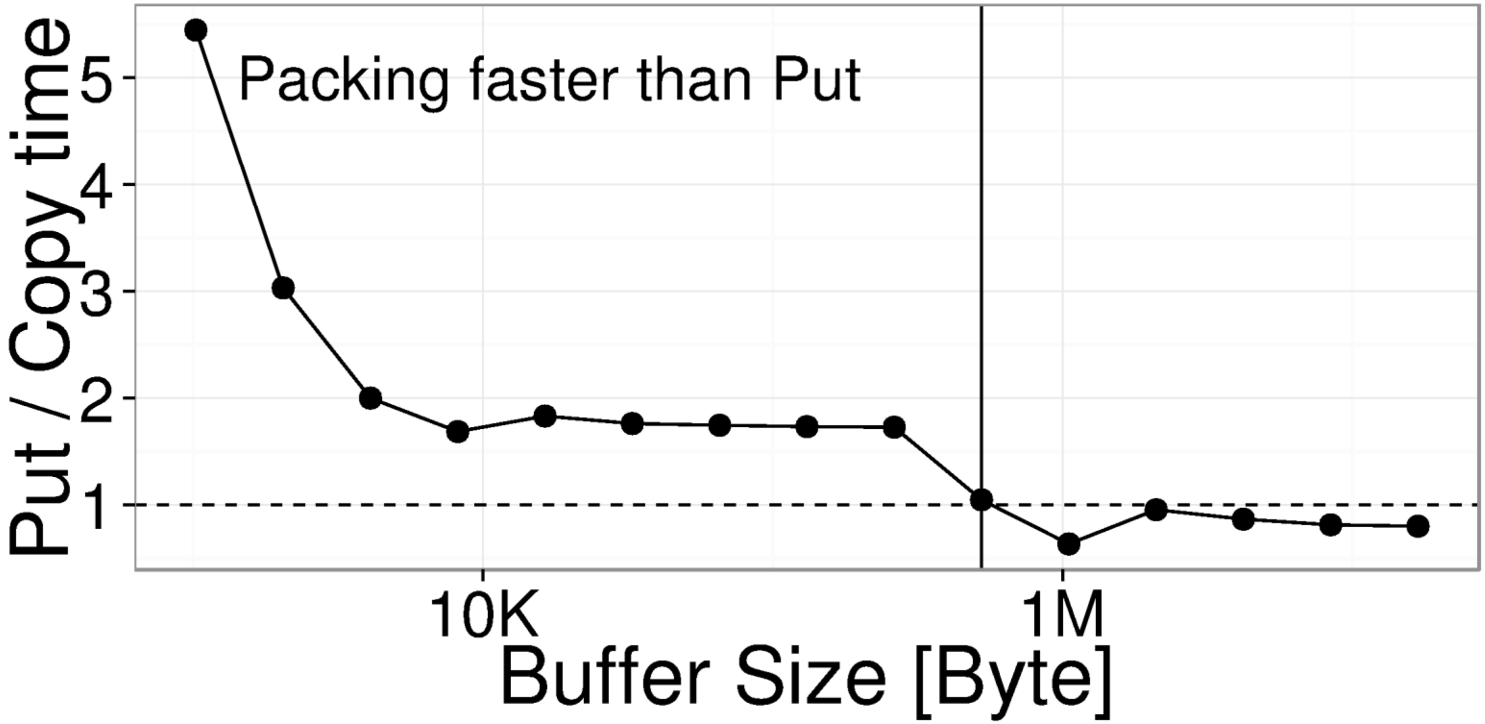


Pipelined Packing

Pipelined-Pack-Put

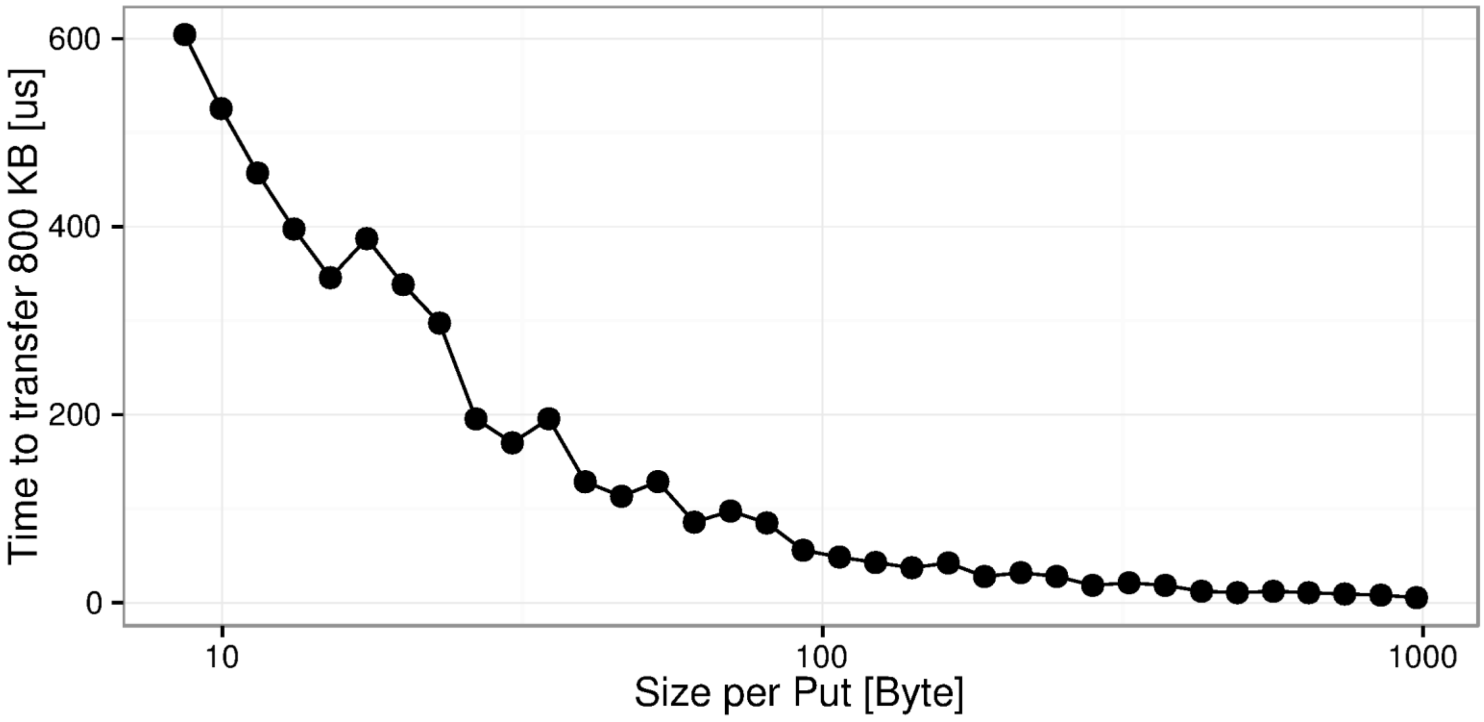


APPLICABILITY?



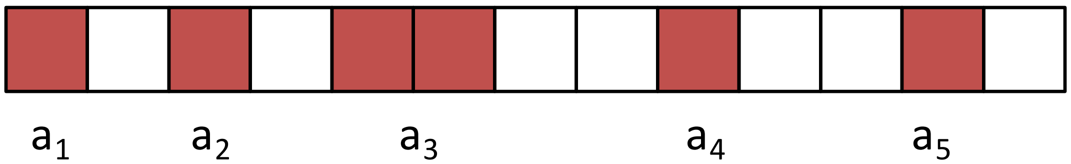
Observation I: If contiguous blocks > 512 kiB then put directly!

Bandwidth Considerations



Observation II: Larger transfers attain higher bandwidth (well known)

A Model for Noncontiguous Transfers



$S = \{a_1, a_2, a_3, a_4, a_5\}$

Block at target

Maximum Block Communication

$$T = \sum_{a \in S} T_{put}(a.l)$$

Fully Packed Communication

$$T = \sum_{a \in S} T_{copy}(a.l) + T_{put}\left(\sum_{b \in S} b.l\right)$$

MODELING NON-CONTIGUOUS PUTS

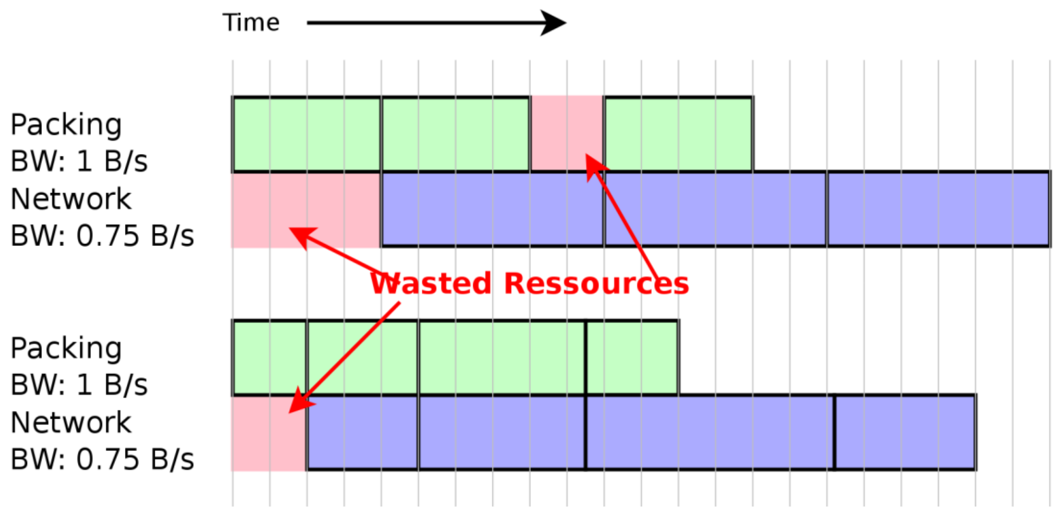
Pipeline packing and remote put

$$T = \sum_{a \in P_1} T_{copy}(a.l) + \sum_{i=2}^{n} max\left(T_{put}\left(\sum_{b \in P_{i-1}} b.l\right), \sum_{c \in P_i} T_{copy}(c.l)\right) + T_{put}\left(\sum_{d \in P_n} d.l\right)$$

Optimization Problem: find the n optimal partitions!

Strategy I: fixed partition size ("fixed pipeline")

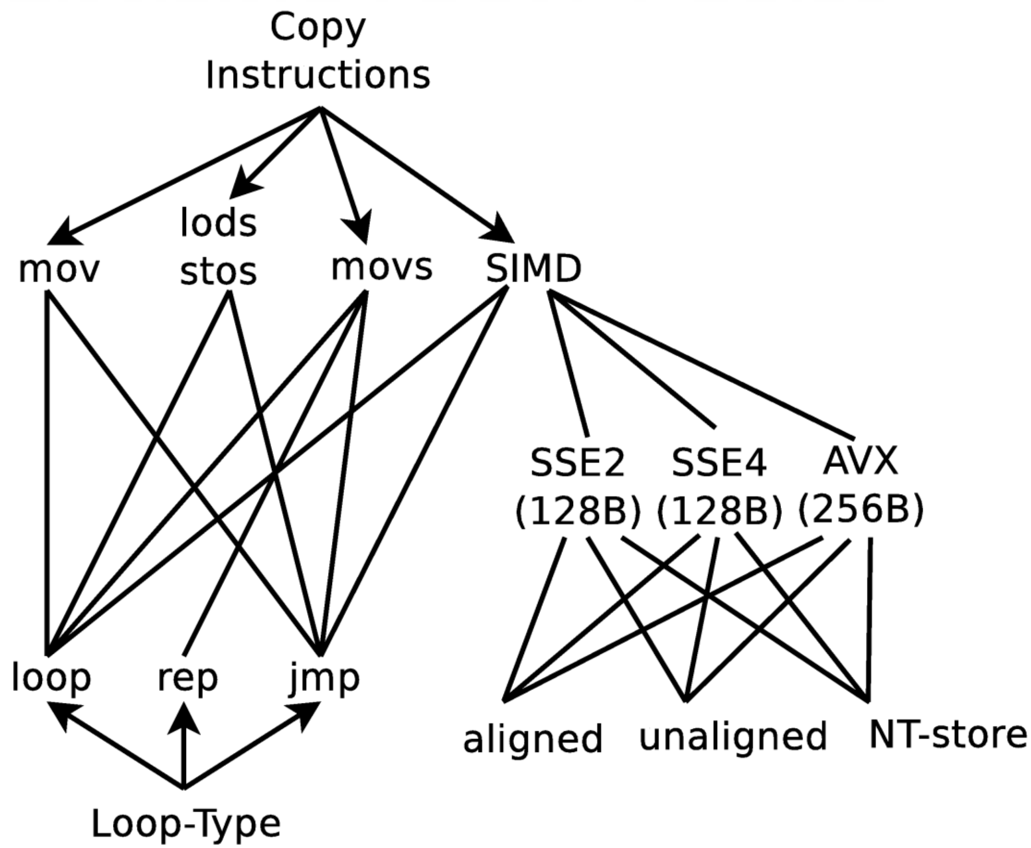
Strategy II: close-to-optimal partition size ("Superpipeline" [3])



[3]:A. Denis: "*A high performance superpipeline protocol for InfiniBand*", EuroPar 2011

Modeling and Optimizing Local Copies

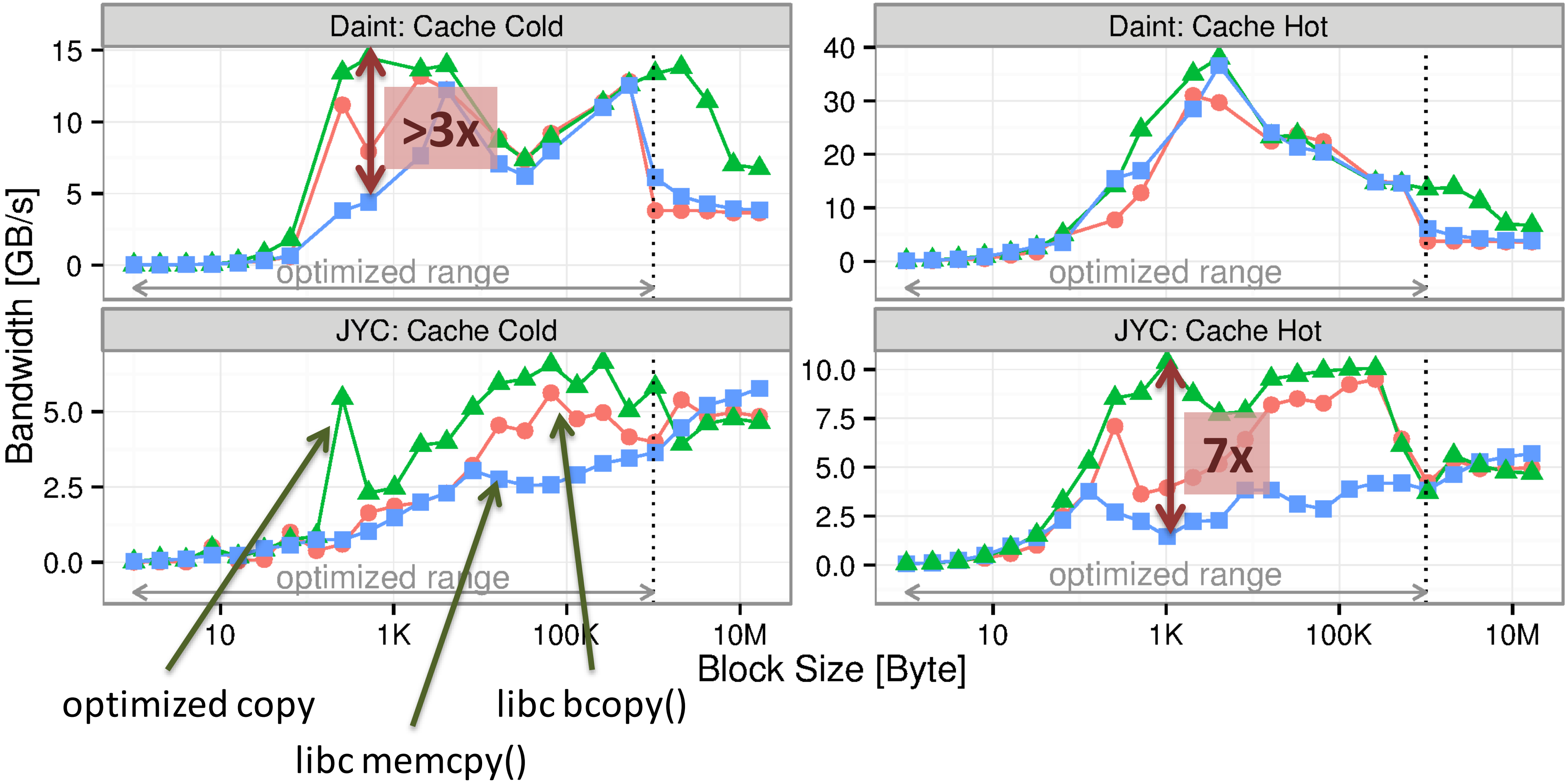
- Lots of choice to move data!
  - > 36 ways on x86
- Restricted semantics allow for Super-optimization [4]
  - Exhaustive search
  - Runs ~1 day
  - Generates close-to-optimal sequences



Overview of data movement and loop-forming instructions on x86-64.

[4]: S. Bansal and A. Aiken: *"Automatic generation of peephole superoptimizers", SIGPLAN Notices 2006*

OPTIMIZED LOCAL COPY SEQUENCE



Network Communication Model

per-put overhead (inverse message rate)

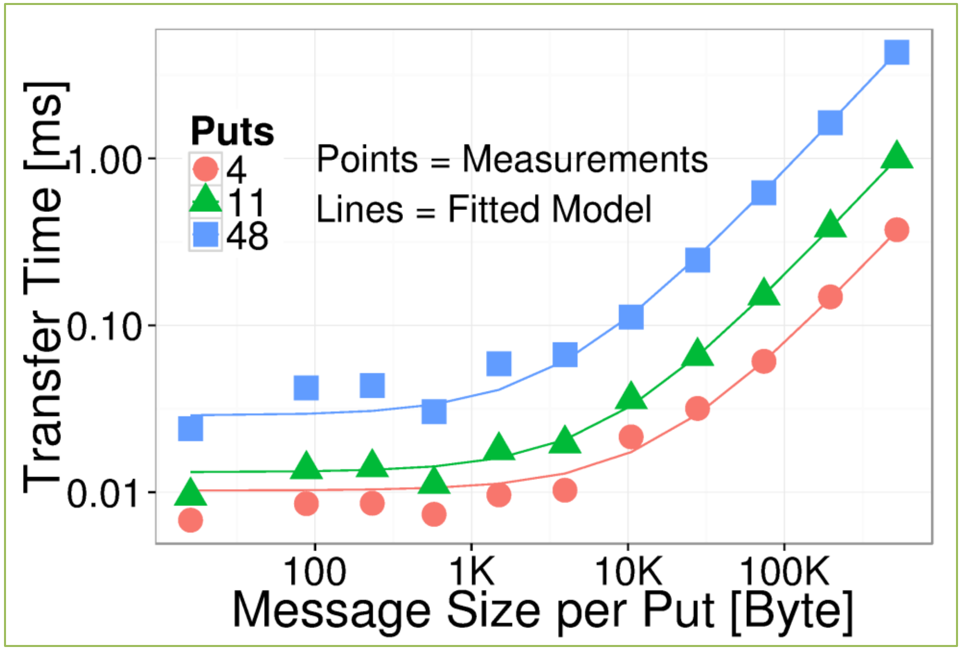
per-byte overhead (inverse bandwidth)

$$T = L + n \times o_{put} + G \sum_{i=1}^{n} s_i$$

latency + synchronization

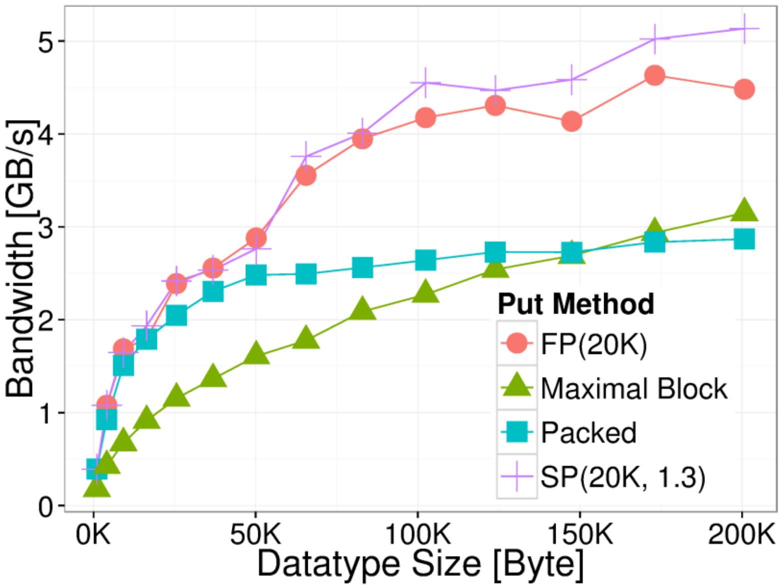
JYC (Gemini):
- L=1 us
- o=0.69 us
- 0.17 ns/B

Piz Daint (Aries):
- L=1 us
- o=0.66 us
- 0.06 ns/B

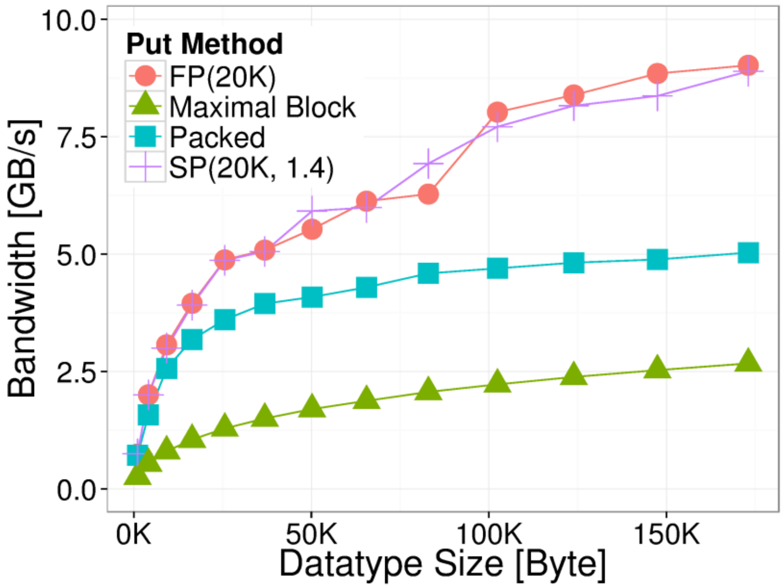


Inter-node communication performance on JYC (R^2=0.999)

Results I: FFT Parallel Transpose



JYC

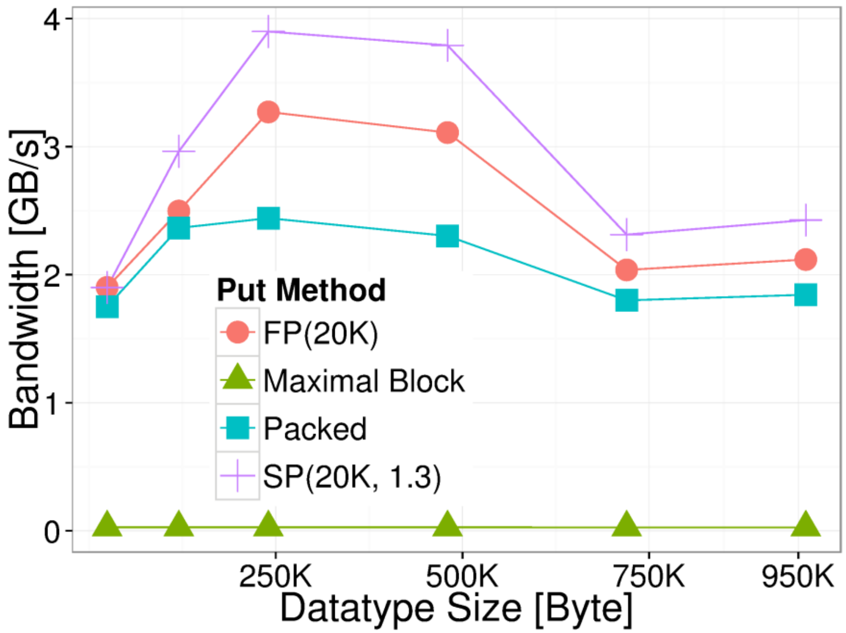


Piz Daint

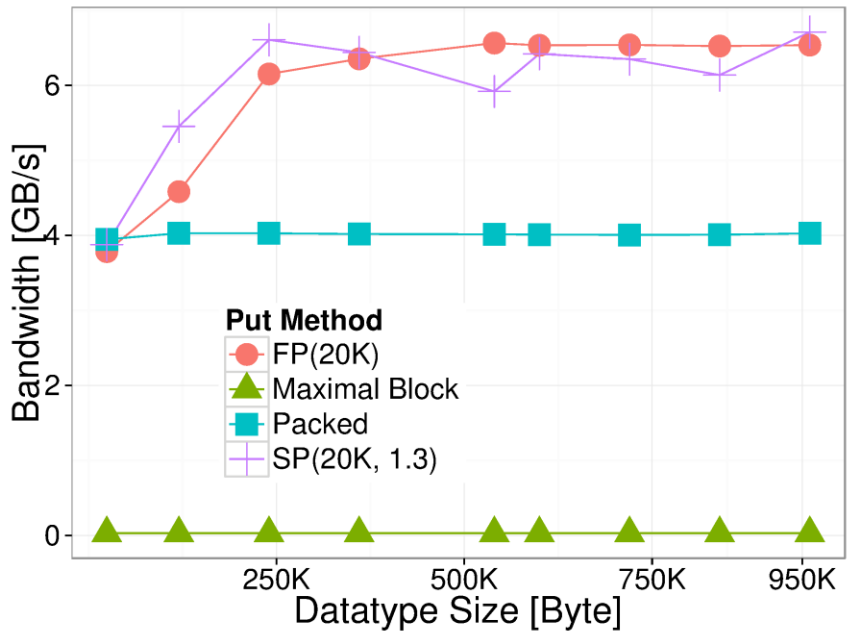
JYC (Gemini):
- L=1 us
- o=0.69 us
- 0.17 ns/B

Piz Daint (Aries):
- L=1 us
- o=0.66 us
- 0.06 ns/B

Results II: SPECFEM3D (12B blocks)



JYC



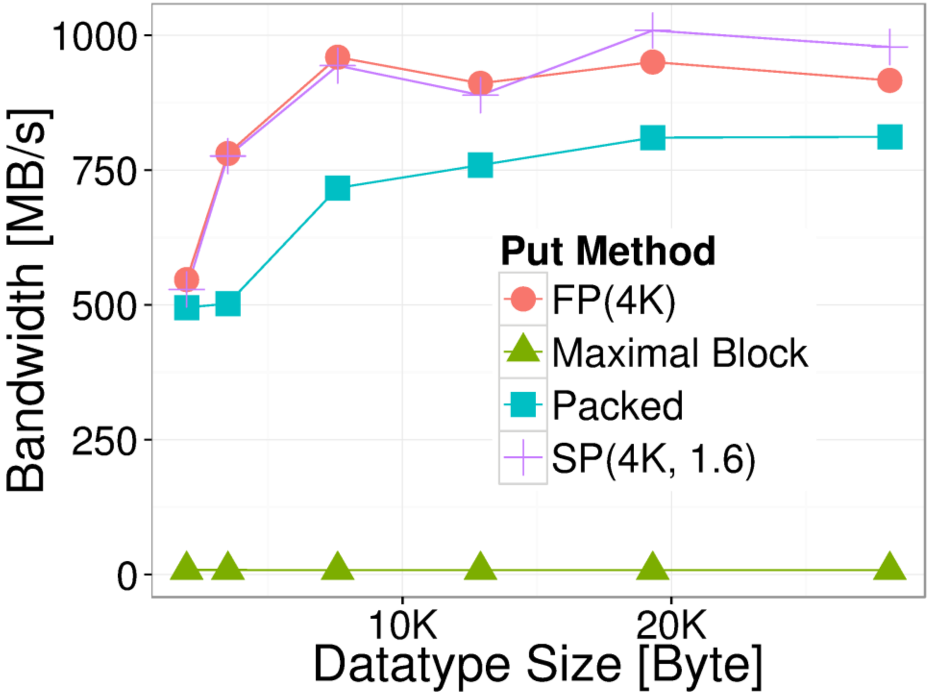
Piz Daint

JYC (Gemini):
- L=1 us
- o=0.69 us
- 0.17 ns/B

Piz Daint (Aries):
- L=1 us
- o=0.66 us
- 0.06 ns/B

Results III: Irregular Data Transfer

- Data layout from SPECFEM3D_GLOBE
- 4 Byte blocks with irregular displacements on sender, consecutive on receiver
- High copy overhead because of the small block size



JYC

Conclusions & Acknowledgments

- Process-local compiler transformations speed up communication >2x
- Analytic performance models work in practice
- Superoptimization for specialized domains

- Thanks to

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

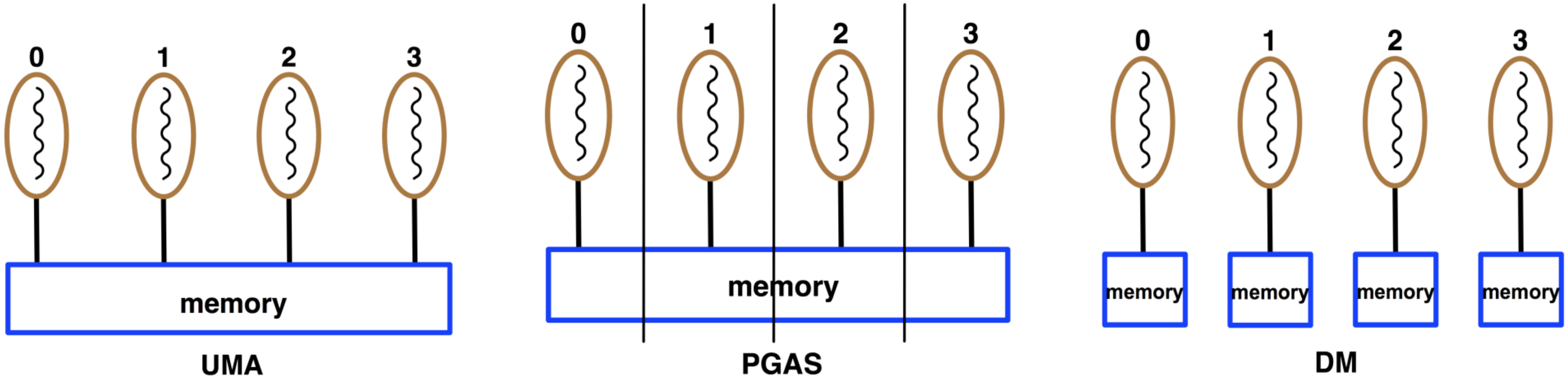
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

BLUE WATERS
SUSTAINED PETASCALE COMPUTING

  - the anonymous reviewers and Kimura-san

Backup Slides

Programming Models Overview



OpenMP, Pthreads, CUDA, …	UPC, CAF, MPI One Sided, …	MPI Two Sided, PVM, …
▪ races, deadlocks, livelocks ▪ hidden locality ▪ memory model issues ▪ scalability issues	▪ races, deadlocks, livelocks ▪ memory model issues ▪ no coherency	▪ deadlocks (rare) ▪ matching overheads
▪ coherency ▪ direct match to hardware	▪ explicit locality ▪ scalable ▪ direct match to hardware	▪ explicit locality ▪ scalable ▪ no races etc. ▪ ease of use