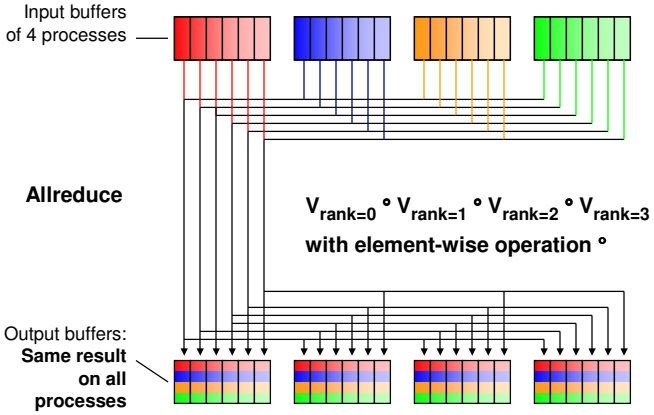
Optimization of MPI_Allreduce and MPI_Reduce

Rolf Rabenseifner, Panos Adamidis
{rabenseifner, adamidis}@hlrs.de

University of Stuttgart
High-Performance Computing-Center Stuttgart (HLRS)
www.hlrs.de

What do we want to do



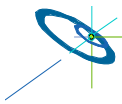
© Rolf Rabenseifner: **Collective Reduction Operation on Cray X1 and Other Platforms**.
CUG 2004, Knoxville, Tennessee, May 17-21 – www.cug.org – Page 1

Basic Principles

Principle I

- Different optimizations for latency and bandwidth
- Latency optimization, e.g.,
  - sending the full input buffers to all processors
  - executing the reduction on all processors
- Bandwidth optimization:
  - splitting the input buffers
  - transferring cross-wise between processes
  - reduction operation only on partial buffers
  - allgather step at the end

} i.e., reduce_scattter



H L R S

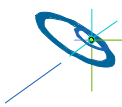
Basic Principles

Principle II

- In case where the number of processors is a power-of-two, then optimization is possible by buffer halving and distance doubling
- In case where the number of processors is non-power-of-two, various algorithms are shown.

Backround

- **37%** of MPI time in **MPI_Allreduce**
- **25%** of user time with **non-power-of-two** number of processes
  - data from automatic profiling of all customers on HLRS CRAY T3E

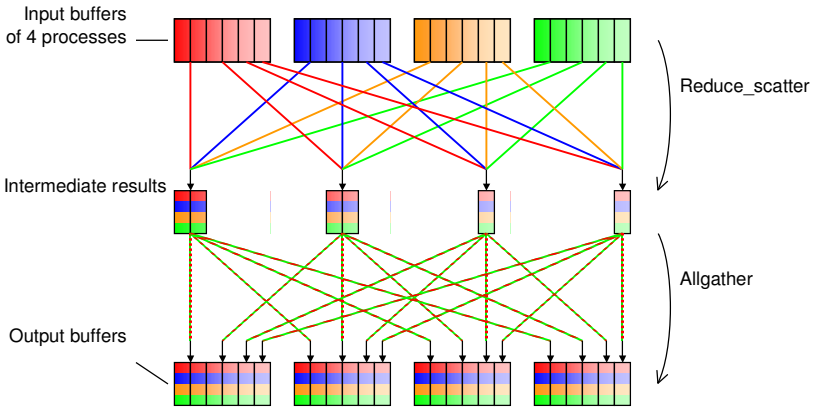


H L R S

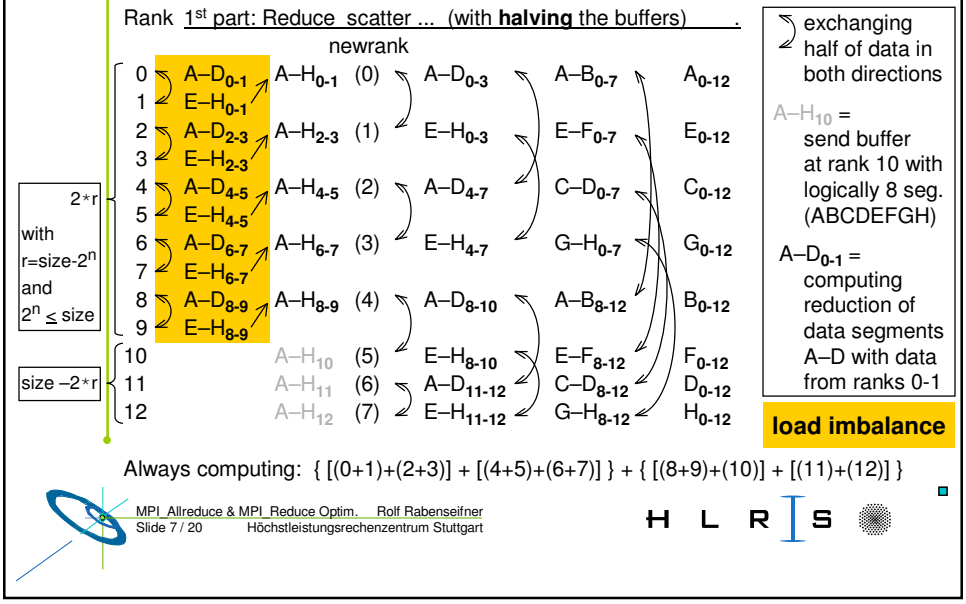
Rabenseifner's Algo., Nov. 1997

- Standard algorithm used in mpich1:
  - MPI_Reduce = binomial tree
  - MPI_Allreduce = binomial tree + MPI_Bcast
  - Binomial tree is inefficient
    - logarithmic behavior but in each iteration, **half** of the processes **gets inactive** ➔ **bad load balancing**
- Better algorithms (butterfly-algorithms):
  - MPI_Reduce = Reduce_scatter + Gather
  - MPI_Allreduce = Reduce_scatter + Allgather

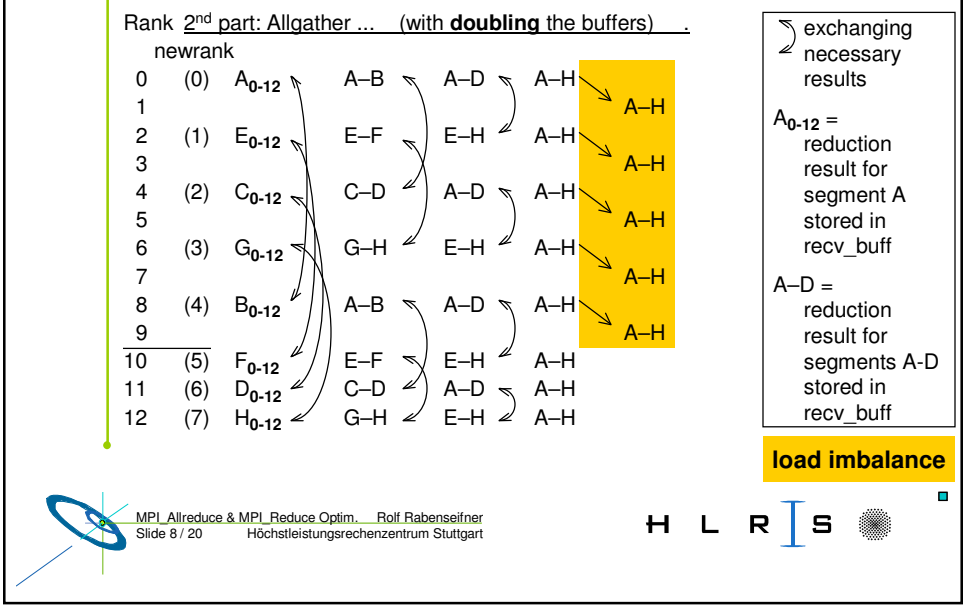
Reduce_scatter and Allgather



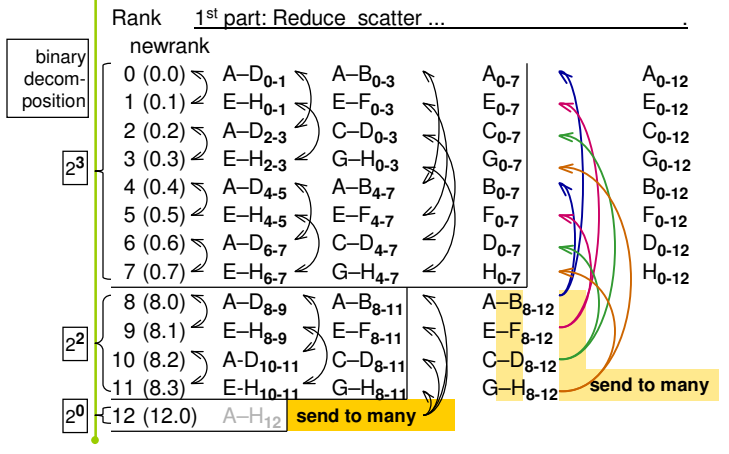
Scheme with Rabenseifner's Algo., Nov. 1997 (1st part)



Scheme with Rabenseifner's Algo., Nov. 1997 (2nd part)



New Binary Blocks Halving+Doubling, July 2003 (1st part)



exchanging half of data in both directions

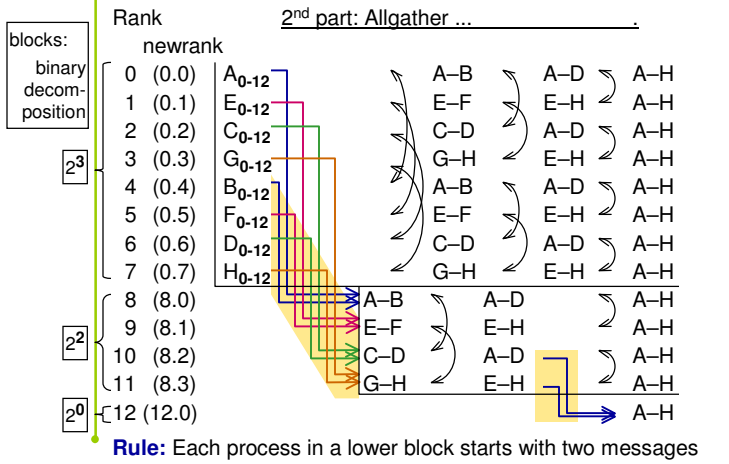
$A–H_{10}$ = send buffer at rank 10 with logically 8 seg. (ABCDEFGH)

$A–D_{0-1}$ = computing reduction of data segments A–D with data from ranks 0-1

load imbalance

Always computing: { [(0+1)+(2+3)] + [(4+5)+(6+7)] } + { [(8+9)+(10+11)] + [(12)] }

New Binary Blocks Halving+Doubling, July 2003 (2nd part)



exchanging necessary results

A_{0-12} = reduction result for segment A stored in recv_buff

A–D = reduction result for segments A-D stored in recv_buff

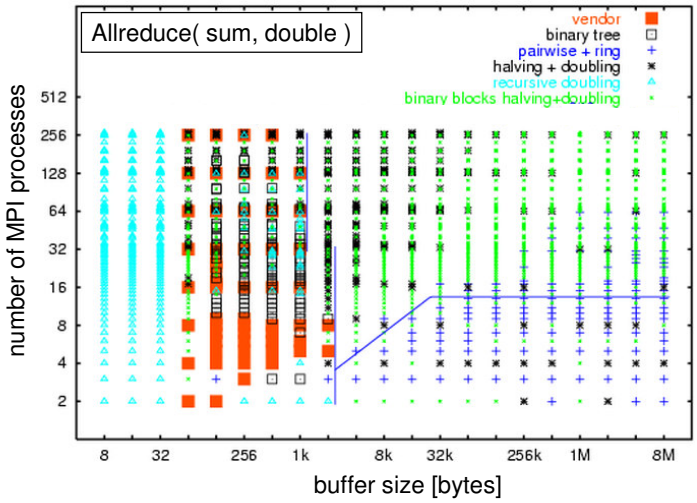
load imbalance

Rule: Each process in a lower block starts with two messages from the next higher block. The messages contain both halves.

Compared Protocols

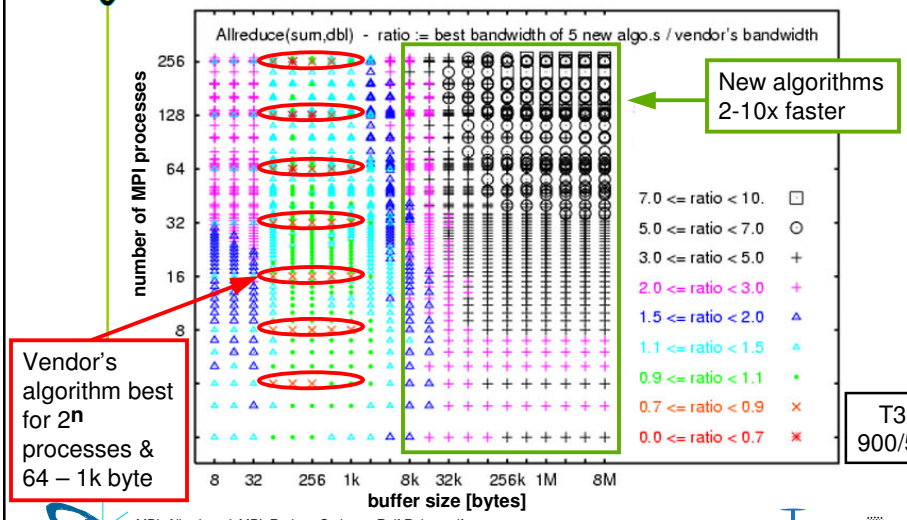
- **Vendor** (MPI_Allreduce and MPI_Reduce of the used MPI library)
- **Binomial tree** + Bcast (i.e., without latency optimization)
- **Recursive doubling** with full buffers (i.e., with latency optimization)
- *Reduce_scatter + Allgather (or Gather)*
  - **Pairwise & Ring**
    - input buffer is devided into (#proc.) pieces of same size
    - optimal load balance but high latency
      O(2x #processes) + O(2x vector size)
  - **Halving & Doubling**
    - O(2x ⌈log(#processes)⌉) + O(4x vector size)
  - **Binary Blocks Based Halving & Doubling**
    - normally better than halving & doubling
    - except for special #processes, e.g. 17, 33, 65,....

Comparison: Fastest Protocol on T3E 900/512

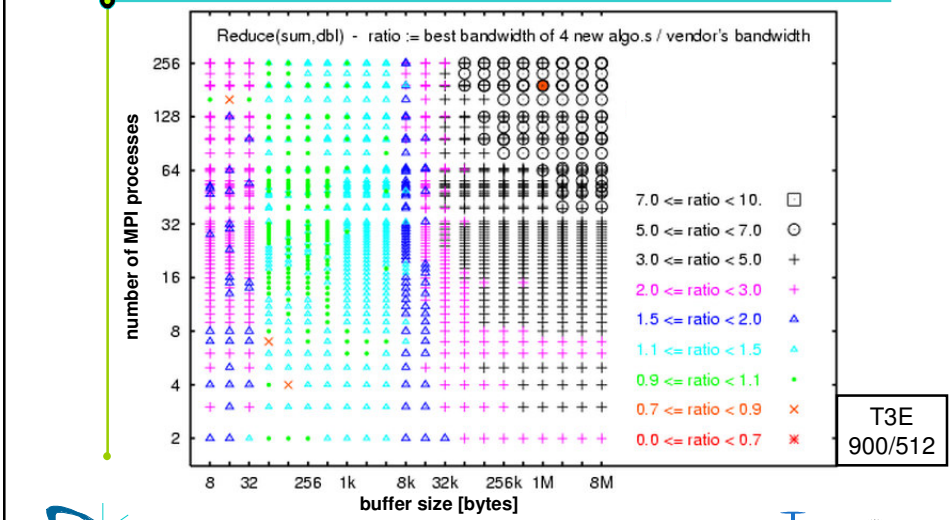


Benchmarks on T3E 900/512, sum of doubles, bandwidth := buffersize / wallclock time

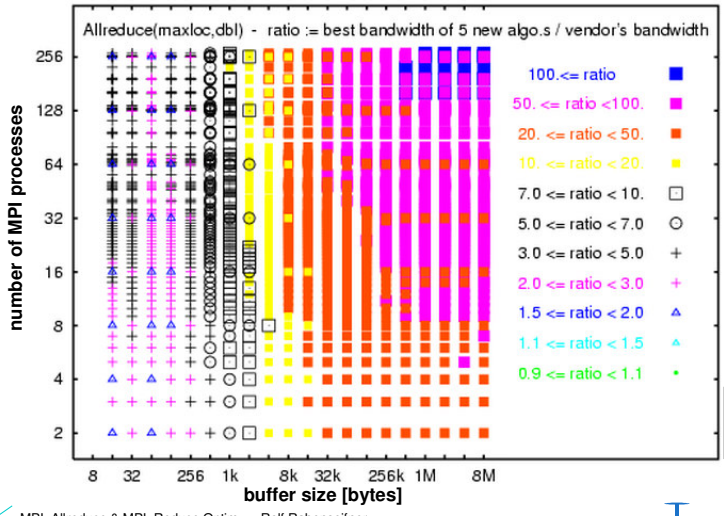
Best algorithm more than x7 faster than vendor's



Reduce: One of the 4 algorithms is always the best

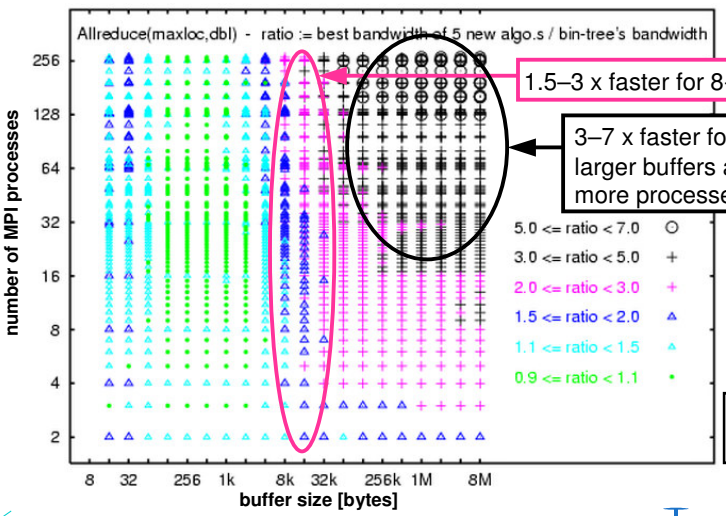


MAXLOC: Vendor's MPI_DOUBLE_INT is extreme slow



T3E 900/512

MAXLOC: Comparing with optimized binomial-tree

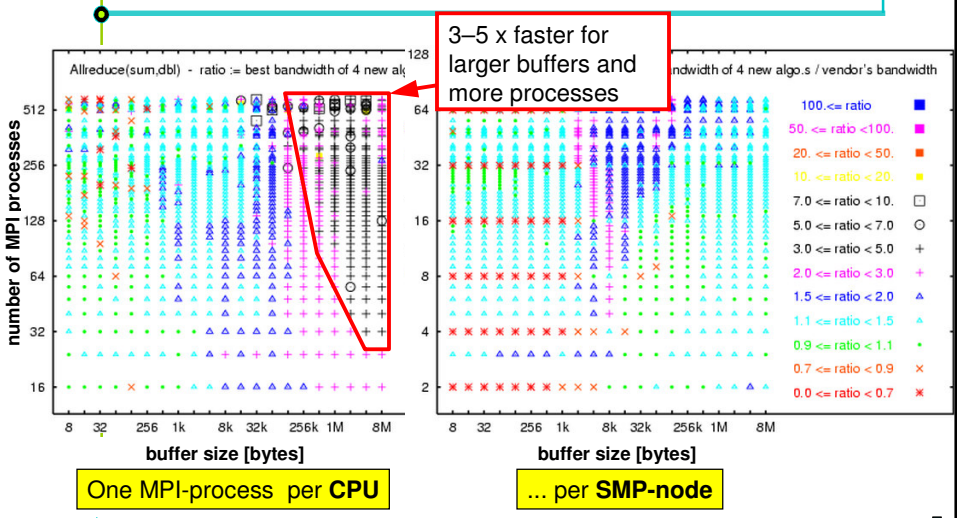


1.5–3 x faster for 8-32 kB

3–7 x faster for larger buffers and more processes

T3E 900/512

MPI_Allreduce on IBM SP at SDSC



One MPI-process per **CPU**

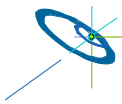
... per **SMP-node**

Real Benefit

- depends on real usage-pattern
  - # calls
  - # bytes (bandwidth vs. latency)
  - # processes (power-of-two?)
  - operation (MPI_SUM, MPI_MAX, MPI_MAXLOC, user-defined, …)
  - Allreduce / Reduce
- Cray X1
  - code is developed on T3E
  - code is tested on CRAY Opteron clusters, Linux clusters, IA32/IA64, IBM
  - code has a problem on Cray X1, need more time for debugging
  - first result on X1:
    - MAXLOC problem with Cray-MPI: **same as on T3E**

Acknowledgments

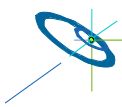
- Thanks for helpful discussions
  - Rajeev Thakur (Argonne)
  - Jesper Larsson Träff (NEC)
- Thanks for help with benchmarking
  - Gerhard Wellein (Uni. Erlangen)
  - Thomas Ludwig, Ana Kovatcheva (Uni. Heidelberg)
  - Rajeev Thakur (Argonne)
  - Monika Wierse, Andy Mason (Cray)
  - Patrick H. Worley (ORNL)
  - Terry Hewitt, Mike Pettipher, Adrian Tate (Uni. Manchester)



H L R S

Conclusion & Future Work

- Latency & Bandwidth optimization of MPI_Allreduce and MPI_Reduce is
  - possible
  - important
  - the '97 algorithm is now part of mpich
- Future work:
  Integrated algorithm under construction
  - smooth optimization for any vector size
  - nearly optimal for any # processes
  - again significantly better bandwidth for non-power-of-two



H L R S