Argonne NATIONAL LABORATORY

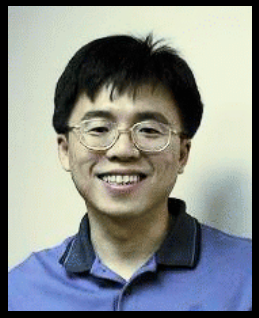
... for a brighter future

A Configurable Algorithm for Parallel Image-Compositing Applications

Tom Peterka
Argonne National Laboratory

Dave Goodell
Argonne

Rob Ross
Argonne



Han-Wei Shen
The Ohio State University

Rajeev Thakur
Argonne

UltraVis
SciDAC Institute for Ultrascale Visualization
www.ultravis.org

U.S. Department of Energy

UChicago Argonne LLC

Office of Science
U.S. DEPARTMENT OF ENERGY

A U.S. Department of Energy laboratory managed by UChicago Argonne, LLC

Tom Peterka
tpeterka@mcs.anl.gov
Mathematics and Computer Science Division

Definition of Image Compositing

Visualization definition: the "sort" in sort-last parallel rendering

The final stage in sort-last parallel visualization algorithms:

1. Partition data among processes
2. Visualize local data
3. Composite resulting images into one

Composition = communication + computation

The computation is usually an alpha-blend called "*Over*"

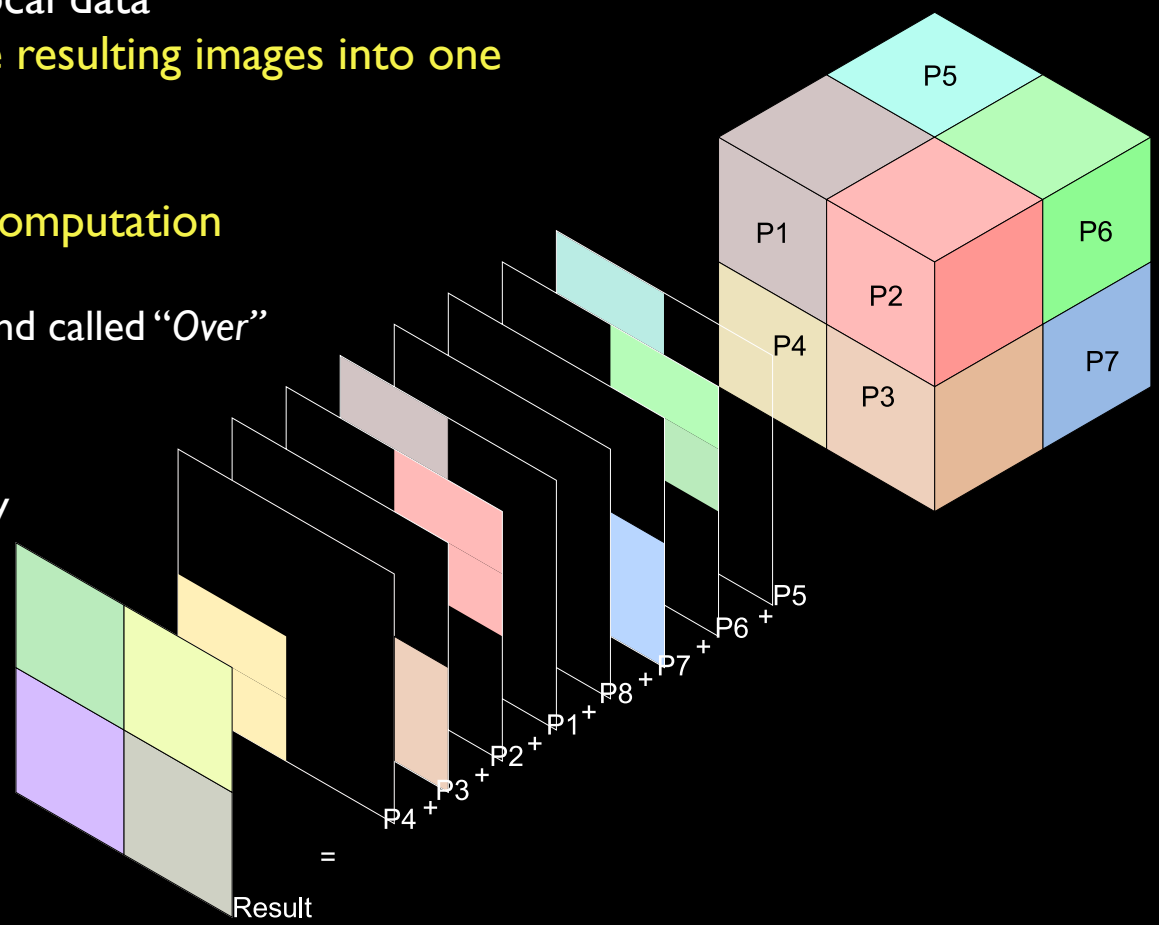
$$i = (1.0 - \alpha_{old}) * i_{new} + i_{old}$$

$$\alpha = (1.0 - \alpha_{old}) * \alpha_{new} + \alpha_{old}$$

where i = intensity (R,G,B), α = opacity

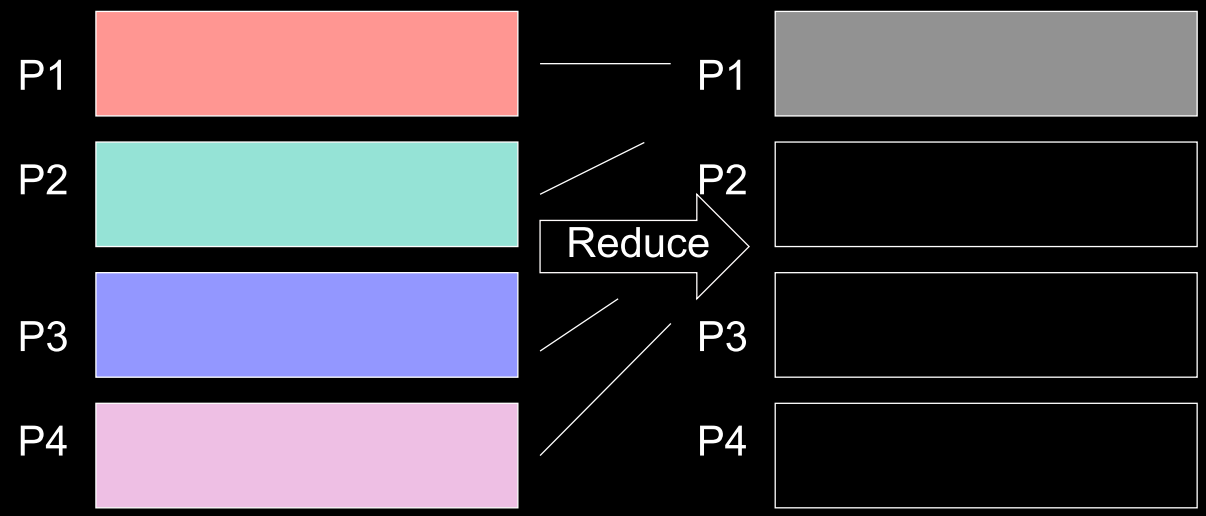
[Porter & Duff, Compositing Digital Images, 1984]

Communication is the subject of this paper



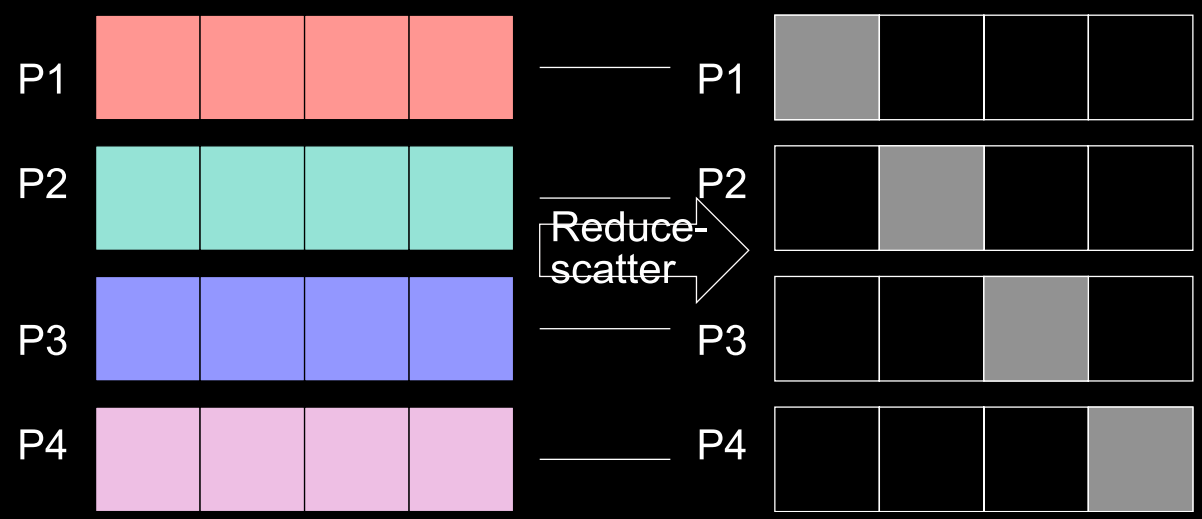
Abstraction of Image Compositing

The message-passing view: a reduction or reduce-scatter



Can be implemented as an MPI collective with user-defined noncommutative reduction operator.

Reduce-scatter is actually better. No need to gather at one node; output image can be written using collective I/O in parallel.



Formal Problem Definition

Three rules

1. P processes each own a vector x_p of length n. (Each element of n is one pixel)

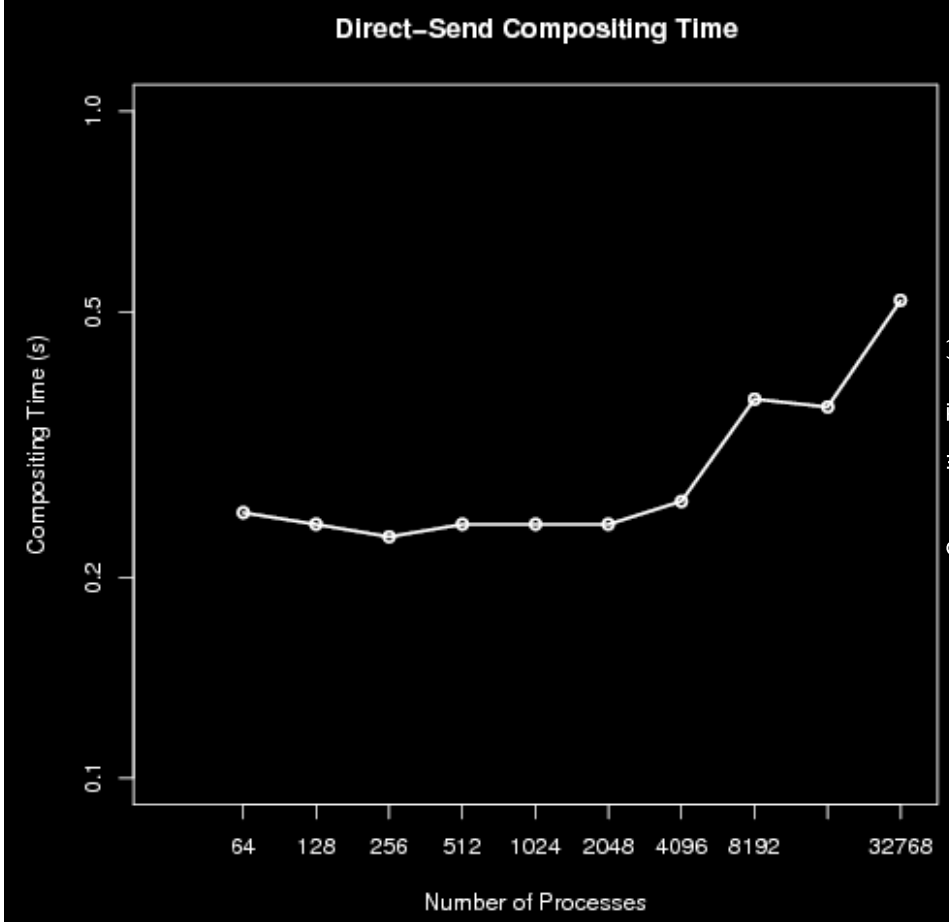
2. *Over* is a binary component-wise linear combination of two vectors. *Over* is associative and noncommutative. In our tests, the canonical order of compositing is p_1 over p_2 iff rank(p_1) < rank(p_2). *Under* is an equivalent operator, p_1 over p_2 $\Leftrightarrow$ p_2 *under* p_1

3. The algorithm terminates when every vector element has its final value. Not all elements need to reside at the same process.

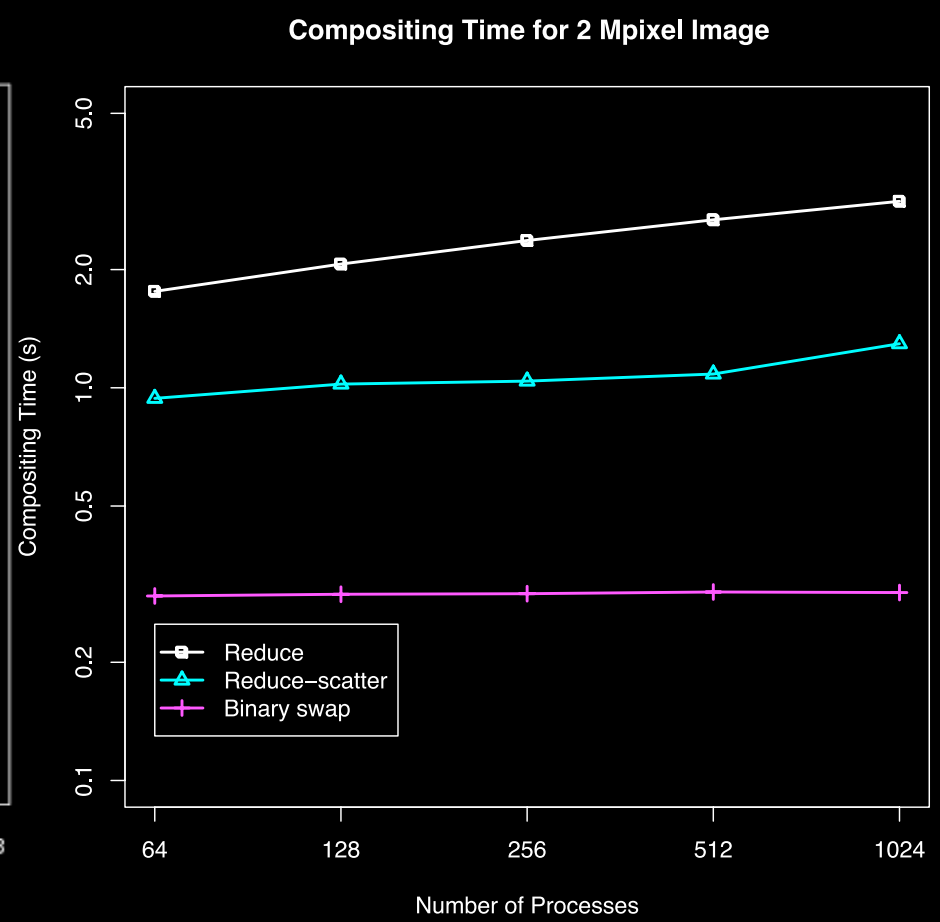
Tested at 1, 2, 4, and 8 Megapixels. 1 pixel = 4 floats (R,G,B,A) (16 bytes per pixel)
Vector lengths are 16MB, 32 MB, 64 MB, and 128 MB, respectively.

Background: Baseline Performance

MPI collectives, direct-send, and binary swap



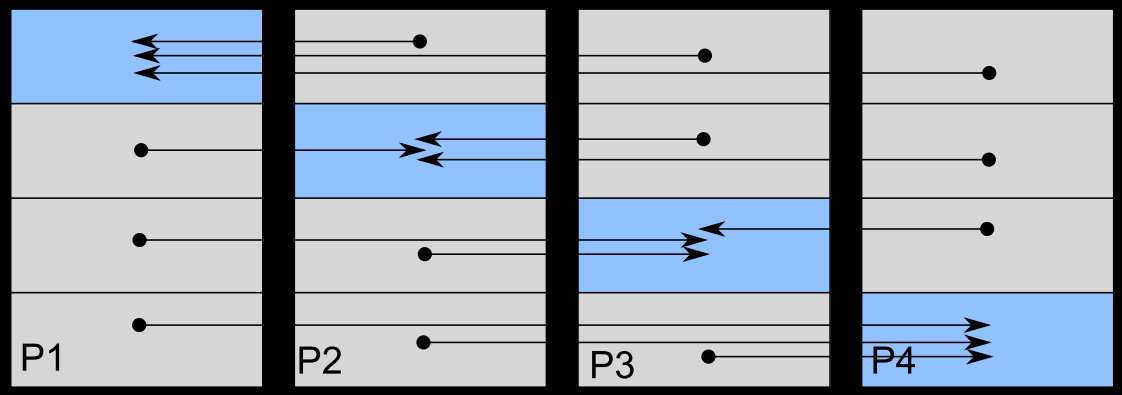
Performance of direct-send compositing for 2.5 Mpixel image degrades after 2048 processes due to contention from larger number of messages.



Performance of binary swap and MPI collectives for 2 Mpixel image. Binary swap performs 3X faster than reduce-scatter.

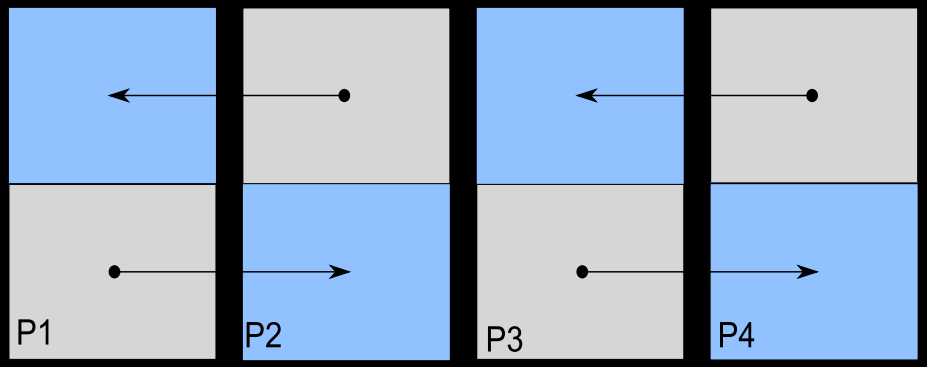
Direct-Send and Binary Swap Operation

Number of rounds, groups, number of participants in a group

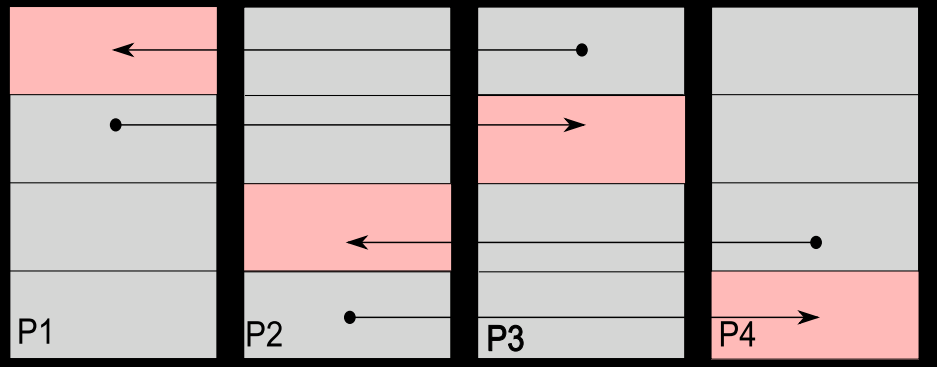


[Hsu, Segmented Ray Casting for DataParallel Volume Rendering, 1993]

Direct-send: maximum parallelism but high number of small messages results in network contention, al messages in one round, non-power-of-two processes ok



Round 1



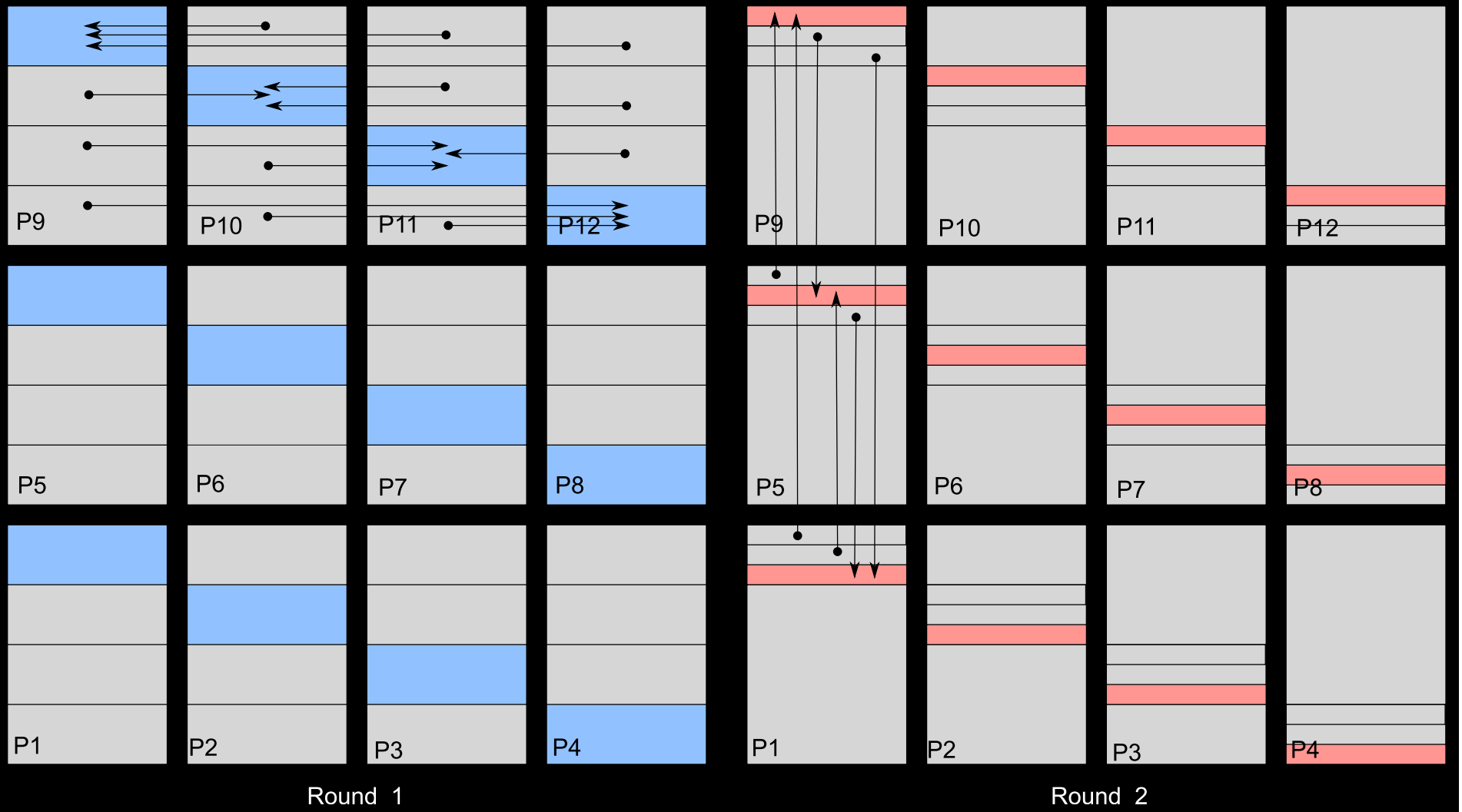
Round 2

Binary swap: fewer messages per round, $\log_2 p$ rounds, p = number of processes, power of 2

[Ma et al., Parallel Volume Rendering Using Binary-Swap Compositing, 1994]

Radix-k Compositing

A generalization of direct-send and binary swap



Radix-k: More parallel, managed contention, p does not need to be power of 2

Keys to Success

Increase message concurrency and overlap communication with computation

-More participants per group than binary swap ($k > 2$)

-Manage contention by limiting k value ($k < p$)

-Overlap communication with computation (nonblocking communication and careful order of operations)

-Can never do worse than binary swap or direct-send

-No penalty for non-powers-of two numbers of processes

Theoretical Complexity

Lower bounds on latency, bandwidth, and computation

Algorithm	Latency	Bandwidth	Computation
Reduce-scatter	$\alpha \log_2 p$	$n \beta (p-1) / p$	$n \gamma (p-1) / p$
Direct-send	$\alpha p / k$	$n \beta (p-1) / p$	$n \gamma (p-1) / p$
Binary swap	$\alpha \log_2 p$	$n \beta (p-1) / p$	$n \gamma (p-1) / p$
2-3 swap (nonpower-of-two case)*	$4 \alpha \log_2 p$	$4 / 3\, n \beta p$	$2 n \gamma p$
Radix-k	$\alpha \log_k p$	$n \beta (p-1) / p$	$n \gamma (p-1) / p$

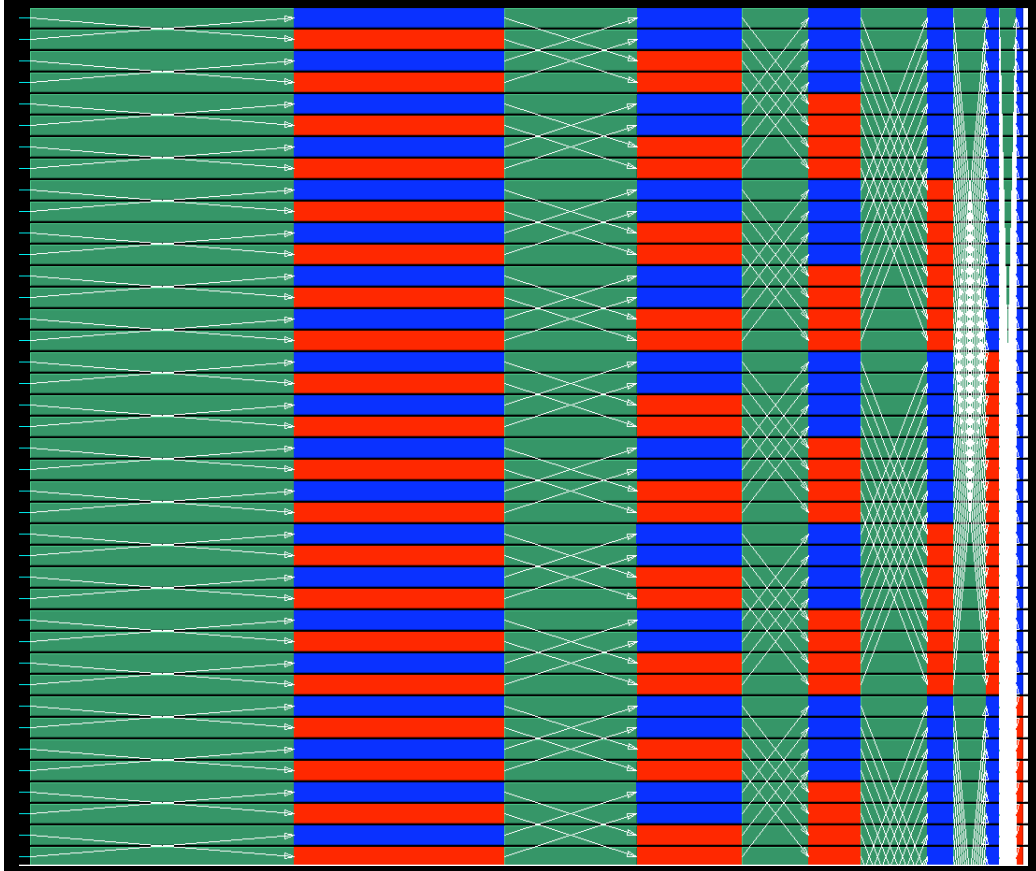
p = number of processes
k = number of participants with no contention
n = length of vector
α = latency per message
β = time to transmit one vector element
γ = time to compute (reduce) one vector element

*[Yu et al., Massively Parallel Volume Rendering Using 2-3 Swap Image Compositing, 2008]

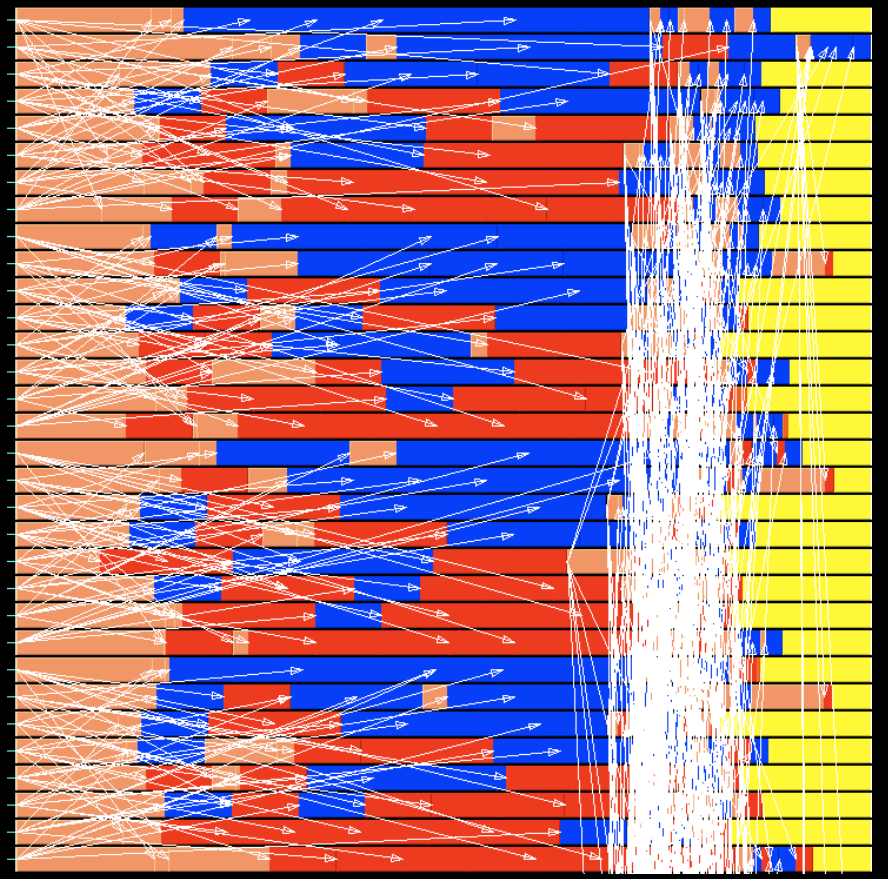
Standard model assuming fully connected network, nonoverlapping communication and computation, zero contention for k participants. Time to transmit one message consisting of n elements is $\alpha + n \beta + n \gamma$.

Profiling Actual Cost

MPE and Jumpshot



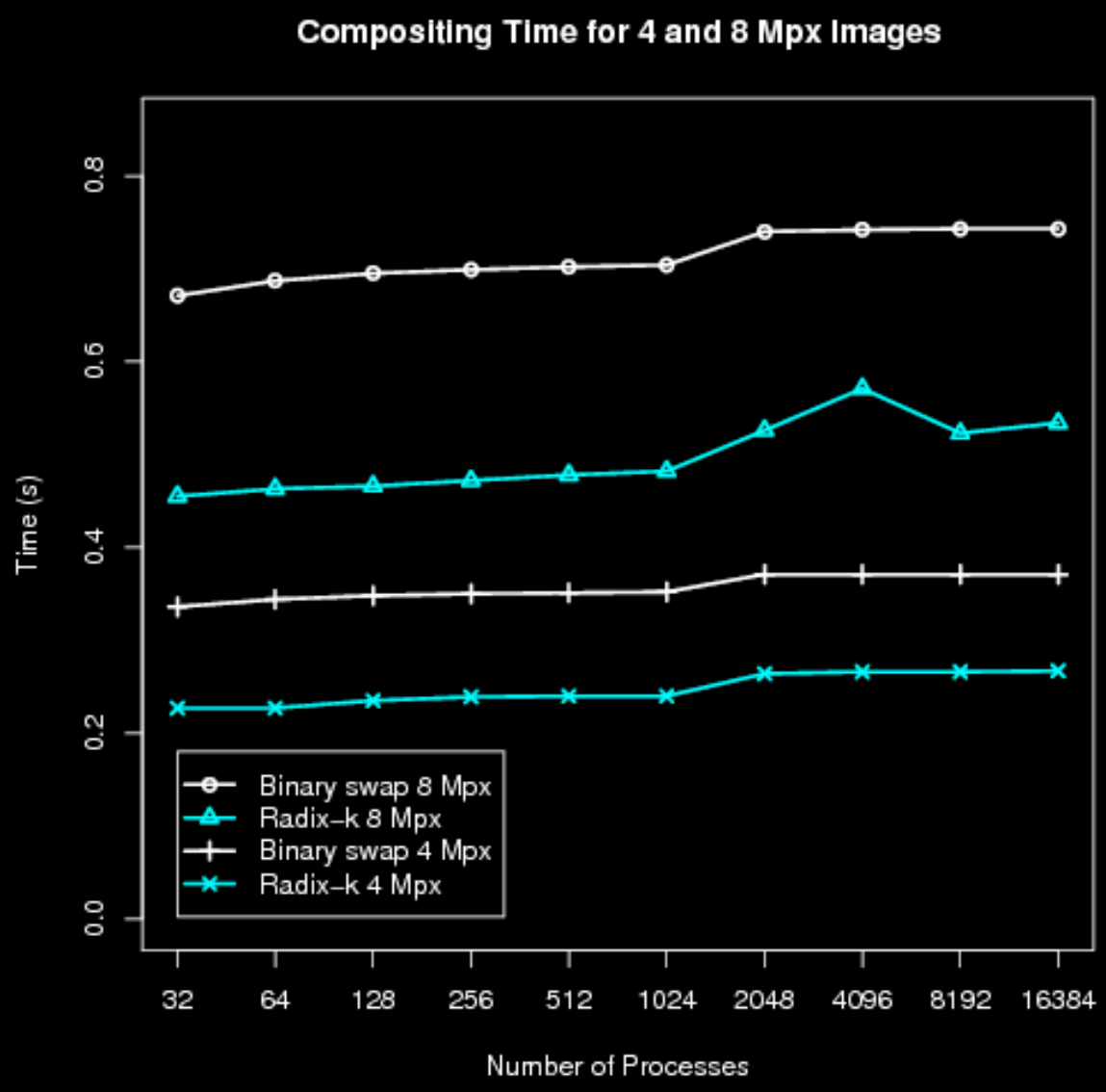
Jumpshot profile of binary swap for 64 processes is highly synchronized into 6 compute – communication rounds.



Radix-k for 64 processes factored into 2 rounds of k = [8, 8] overlaps communication with computation whenever possible.

Radix-k Performance

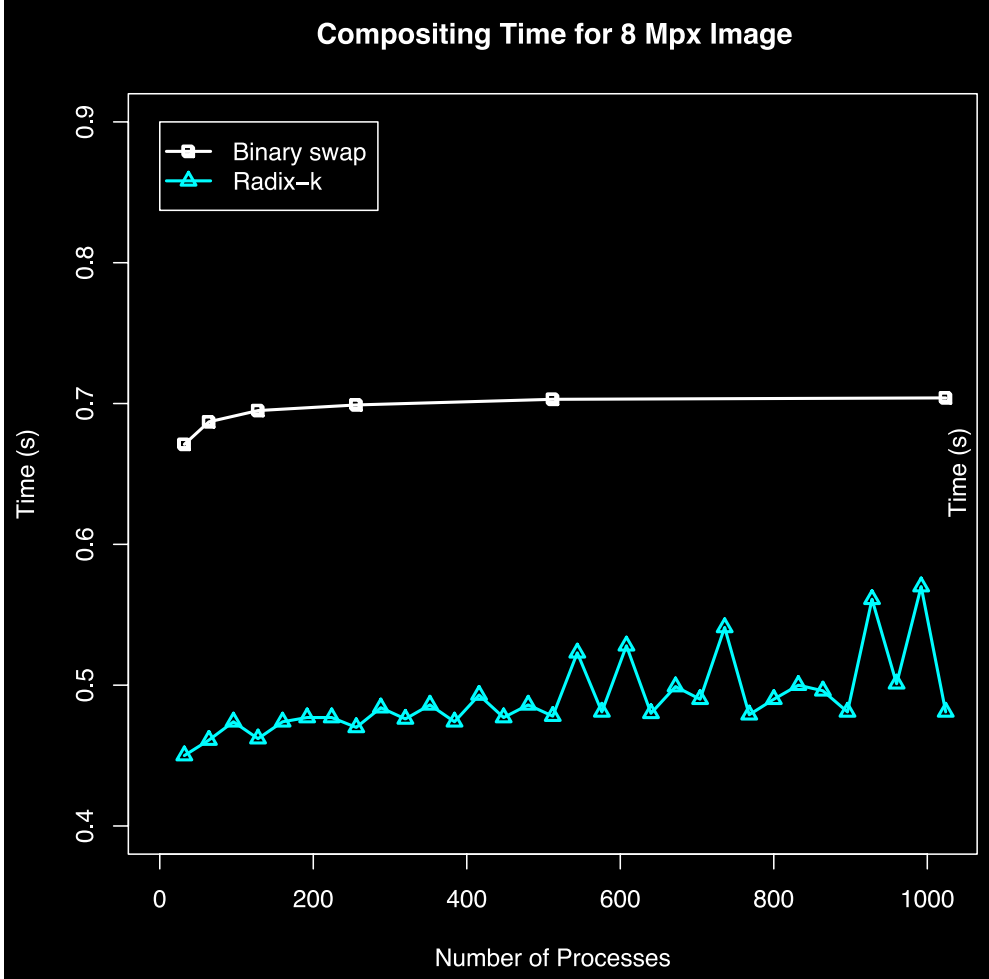
Powers of two process counts on Blue Gene/P Intrepid

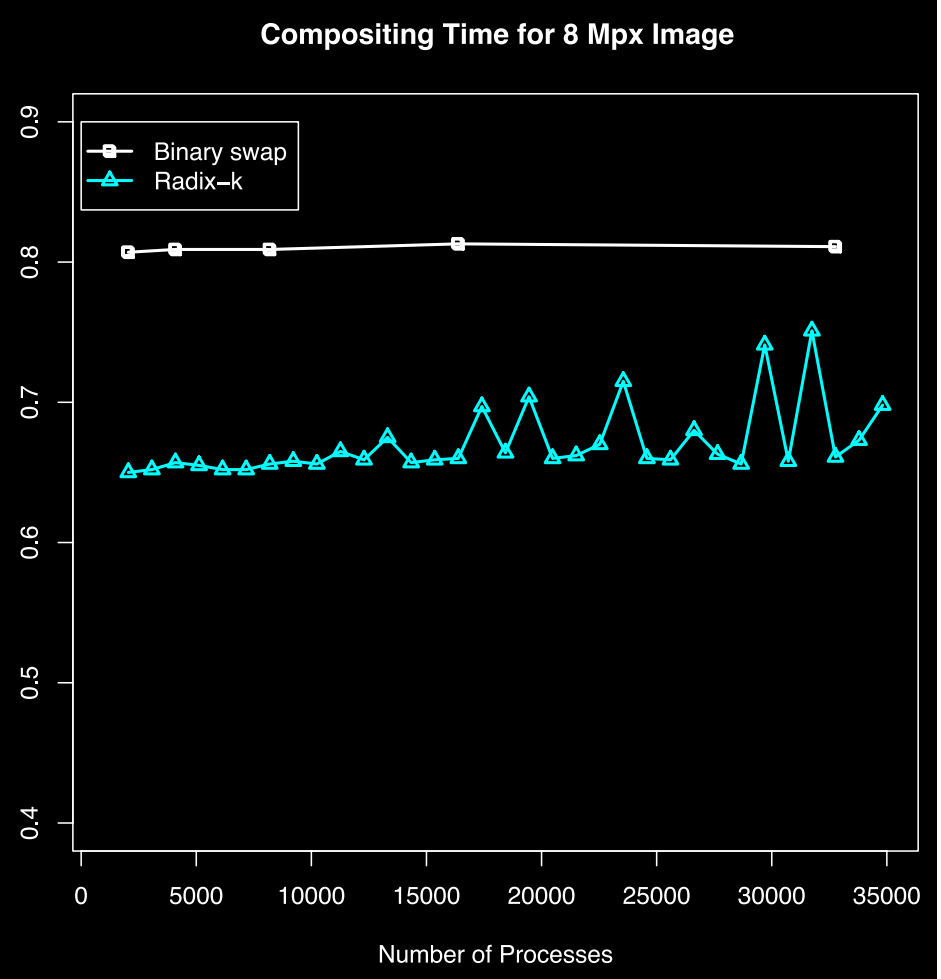


Scalability over a range of process counts and image sizes. Radix-k performance is 40% better than binary swap. The step at 1024 processes is due to moving beyond a single rack in the 3D torus of Blue Gene/P.

Radix-k Performance

From 32 to 35,000 processes including non-powers-of-two on Blue Gene/P Intrepid



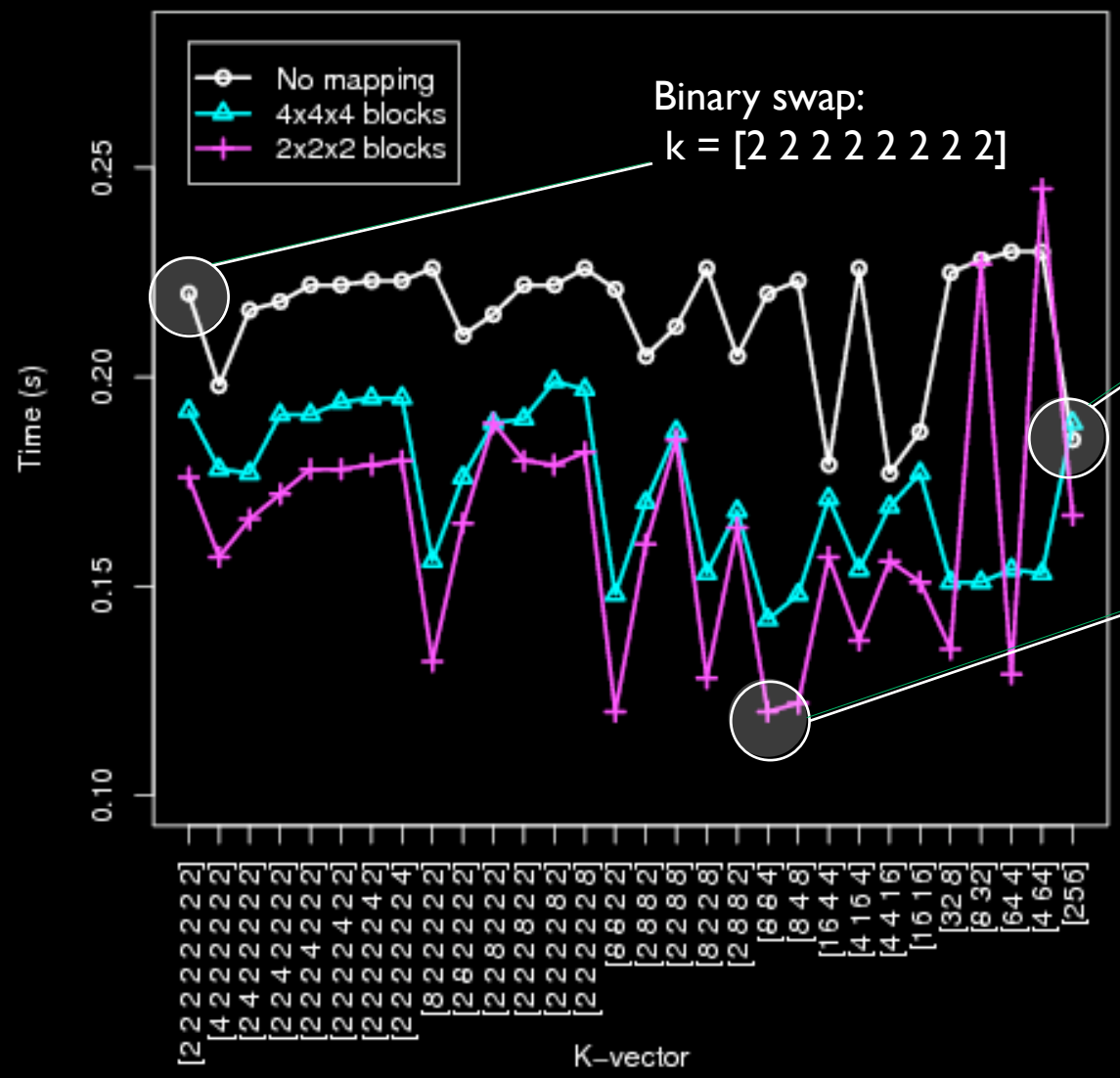


Radix-k continues to perform with a 40% improvement over binary swap at non-powers-of-two process counts. Left: p varies from 32 to 1024 in steps of 32. Right: p continues from 1024 to 35,000 in steps of 1024.

Selecting the K-vector

Factoring the number of processes and considering network topology

Evaluating K–vectors and Mapping Processes to Physical Locations

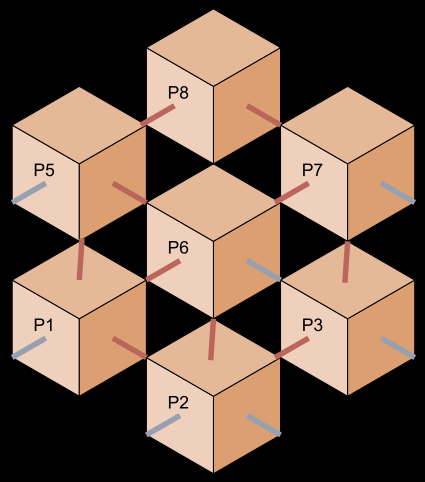


Binary swap:
k = [2 2 2 2 2 2 2 2]

Performance of different k-values for 256 processes, 2 megapixel image size.

Direct-send:
k = [256]

Sweet spot for this architecture:
k = [8 8 4] with mapping process ranks into physical blocks of 2x2x2.



Recap

Review and looking ahead

Contributions

-Unifies direct-send, binary swap and points between

- Configurable to architecture

- Non-powers-of-two number of processors

Ongoing and future work

- Optimizations: bounding boxes, load balancing

- Autotuning

- Implementation in visualization libraries and MPI

Argonne NATIONAL LABORATORY

... for a brighter future

A Configurable Algorithm for Parallel Image-Compositing Applications

Thank you

Acknowledgments:
Argonne Leadership Computing Facility
US DOE SciDAC UltraVis Institute

Tom Peterka
tpeterka@mcs.anl.gov
Mathematics and Computer Science Division

UltraVis
SciDAC Institute for Ultrascale Visualization
www.ultravis.org

U.S. Department of Energy

UChicago Argonne LLC

Office of Science
U.S. DEPARTMENT OF ENERGY

A U.S. Department of Energy laboratory managed by UChicago Argonne, LLC