From AAPC to
Random Permuters and Sorters

Thomas M. Stricker and Jonathan C. Hardwick

School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213

Talk presented by
Thomas Gross

Carnegie Mellon

SPAA96, Padova, Veneto, Italia
June 25, 1996

Outline

AAPC: All to All Personalized Communication

- Congestion controlled AAPC routing
    - Quantify benefits and practical aspects (T3D)
- Memory operations in element permutations
    - Quantify difference between AAPC of blocks and true permutation of elements
- Sorting performance
    - Derive performance from an architectural model instead of measured implementation

AAPC Algorithms

- Conventional AAPC
  - Independent loop through all destinations
  - Router resolves congestion dynamically
  - Performance of AAPC: average over routes
- Phased AAPC
  - Synchronized loop through *k* phases/patterns
  - Congestion resolved statically
  - Performance of AAPC: slowest route determines phase; after that average over phases

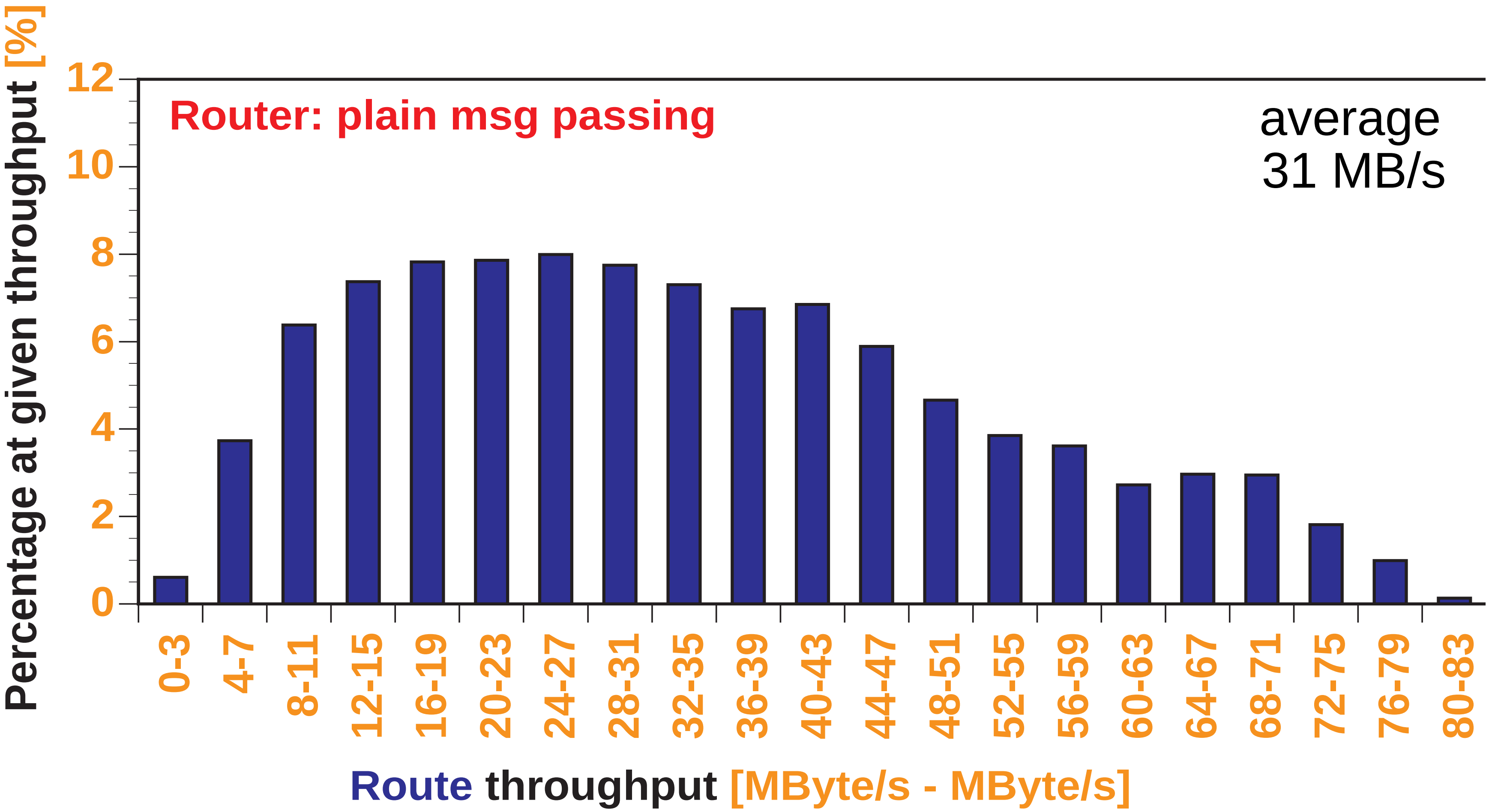
Target Architecture

Cray T3D with 512 nodes at the PSC:

- Topology 8x4x8x2 torus
- Non adaptive, dimension order routing on a 3-dimensional torus
- Short messages: 32 bytes pay load
- Fast links (2*75 MB/s per link raw speed)

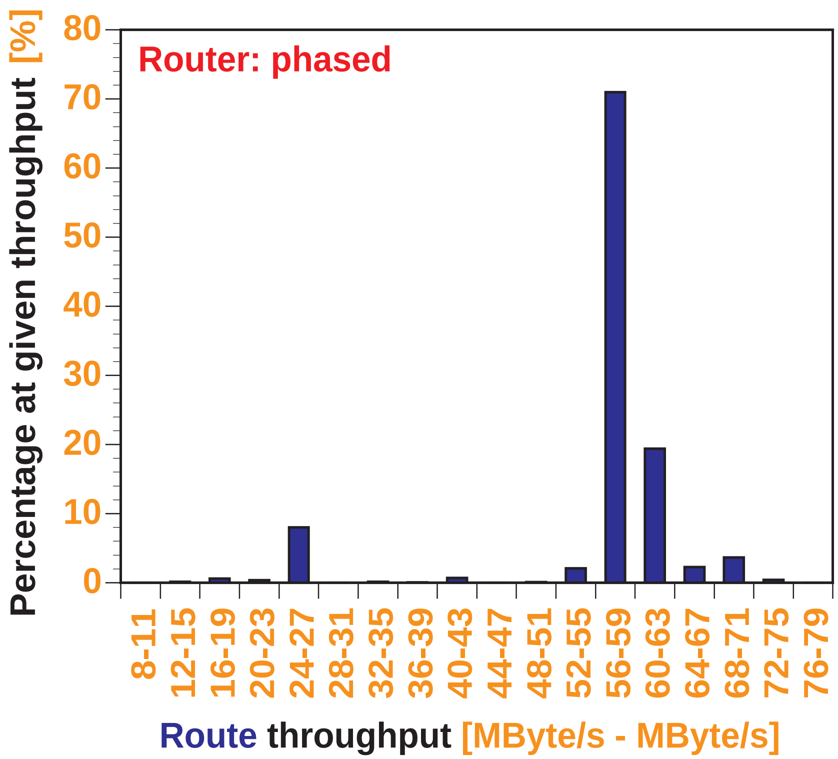
AAPC in Hell

(default routing - no phases)

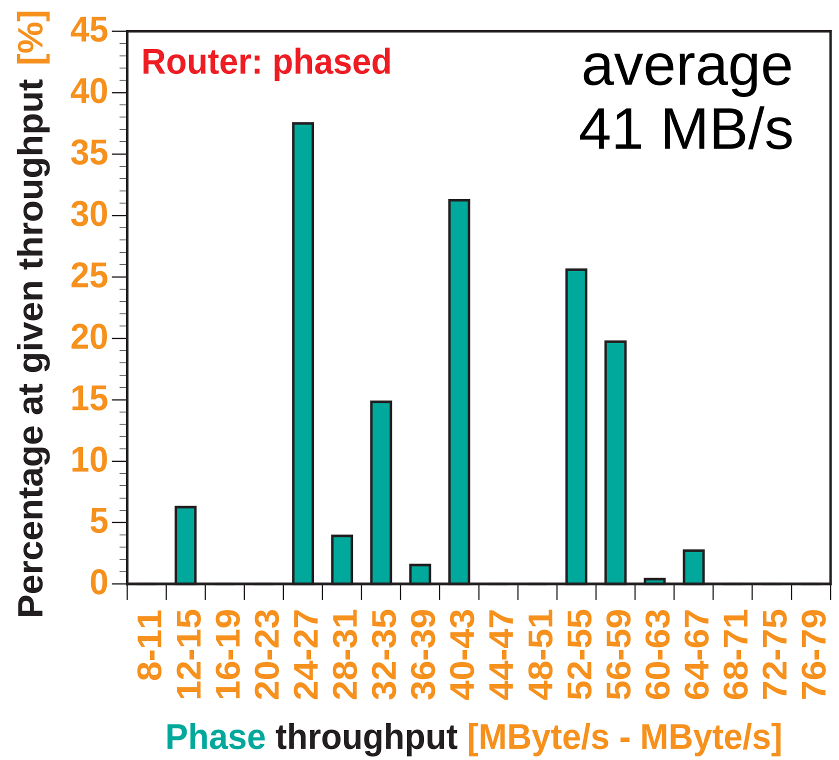


AAPC in Purgatory:

(phased routing with simple 3-dimensional patterns)



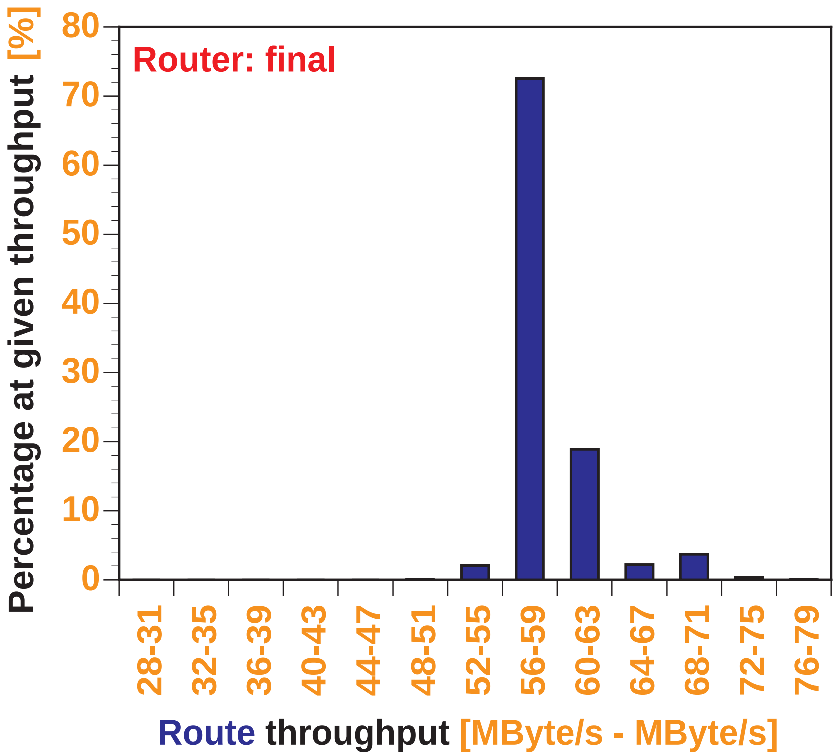
Throughput distribution over 262k routes



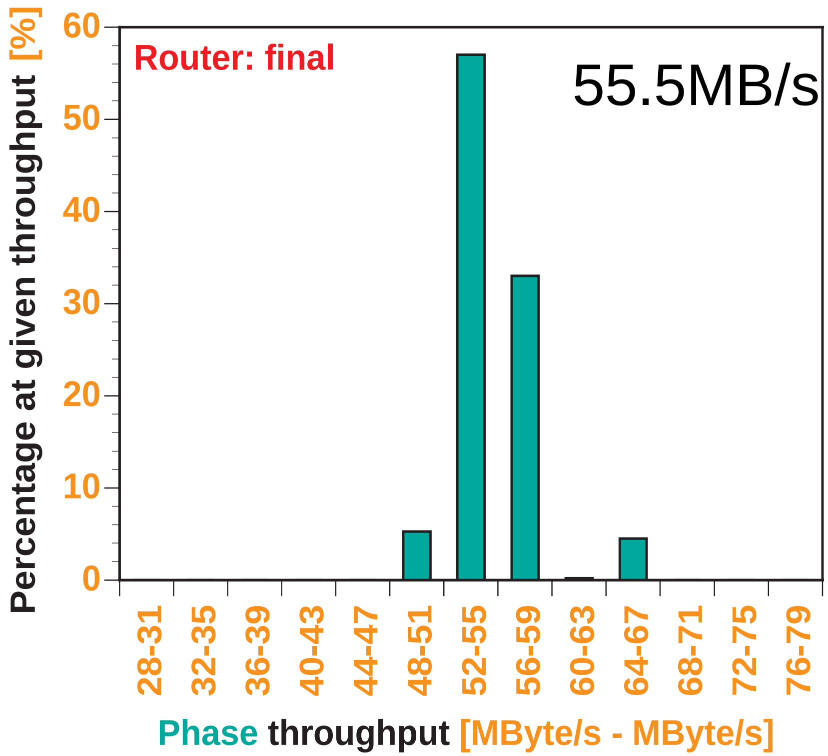
Throughput distribution over 512 phases

AAPC in Heaven:

(congestion free - after modified pattern and router)



Throughput distribution
over 262k routes



Throughput distribution
over 512 phases

Performance Impact of Congestion Controlled AAPC Routing

Balanced AAPC on 512 node T3D			
Algorithm	MBytes/s per node	MByte/s aggregate	Percent of Limit
No phases	31.1	15,945	41%
Phased	41.4	21,200	55%
Pattern fixed	46.7	23,910	62%
Router fixed	54.4	27,852	73%
Final	55.5	28,416	74%
Hardware	75.0	38,400	100%

Options for Element Permutations

- **Buffer-packing transfers**
  - Pack buffer (gather)
  - Transfer data across network, AAPC of blocks
  - Unpack buffer (scatter)
- **Chained transfers**
  - Pack - transfer - unpack

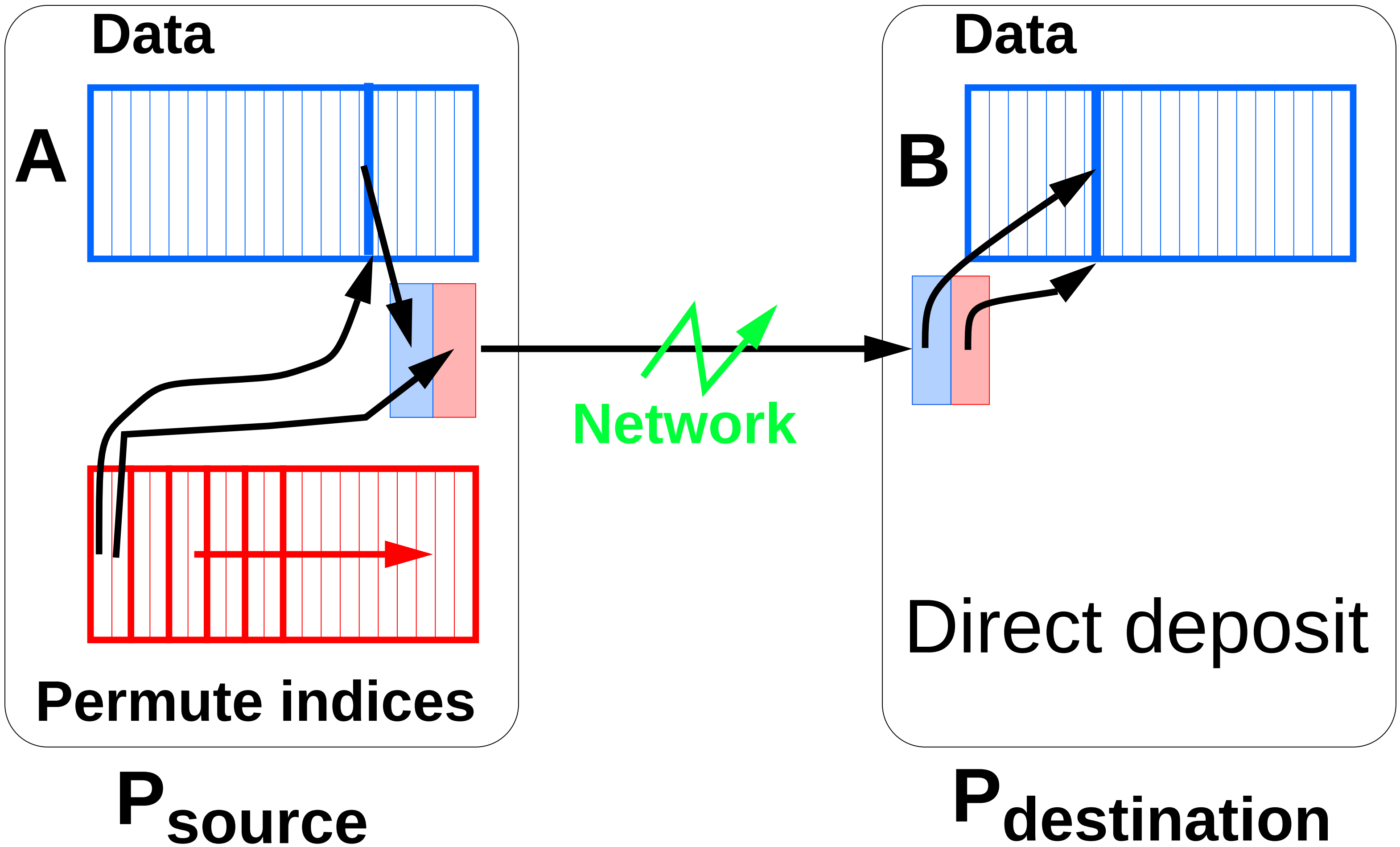
On a Cray T3D:

Buffer packing: 14 MB/s - Chained: 32 MB/s

Trade-offs described in Stricker/Gross ISCA95

Element Permutations (B=Perm[A])

(routed on a message passing machine)



Performance of Element Permutations

(including memory operations)

	Model	Measured on 512 node T3D		
AAPC type	MBytes/s per proc.	MBytes/s per proc.	MByte/s aggregate	Percent of Limit
Blocks	69	55.5	28,416	74%
Transpose	38	29.5	15,104	39%
Random Permute	32	22.0	11,264	29%

Motivation for Modelling Steps of Sorters

- Recent algorithmic work:
  - Bucket work vs. key work [Blelloch, Zagha]
  - Balanced transposes vs. unbalanced permutes [Bader, Helman, Jájá]
- Previous implementation work:
  - Measured implementations
  - Indirect connection to node architecture
- Copy transfer model:
  - Analytic model
  - Direct connection to architectural features

Modelling Radix Sort Performance

Primitive	Model work	Model MByte/s	Measured MByte/s
Bucket Counting	key	21	14
Local Permutation	key	21	8
Global Permutation	key	32	20
Bucket Transpose	bucket	38	20
Bucket Reduction	bucket	220	125
Bucket Scan	bucket	90	60
Bucket Untranspose	bucket	38	20

Model: 7.88 MB/s per processor

Measured: 3.99 MB/s per processor, 2042 MB/s total, 255 million keys/s (16bit key, 48bit data)

Conclusions

- Difficult to achieve hardware performance on AAPC routing of blocks.
  - Increase from 41% of nominal peak to 74%
  - Hiding net topology details is not a good idea!
- Cost of memory operations in permutations.
  - Element permutations at 29% of peak
  - Memory operations are significant!
- Estimated sorting performance with copy transfer model
  - Local memory operations are dominant costs!

Simple Radix Sorter

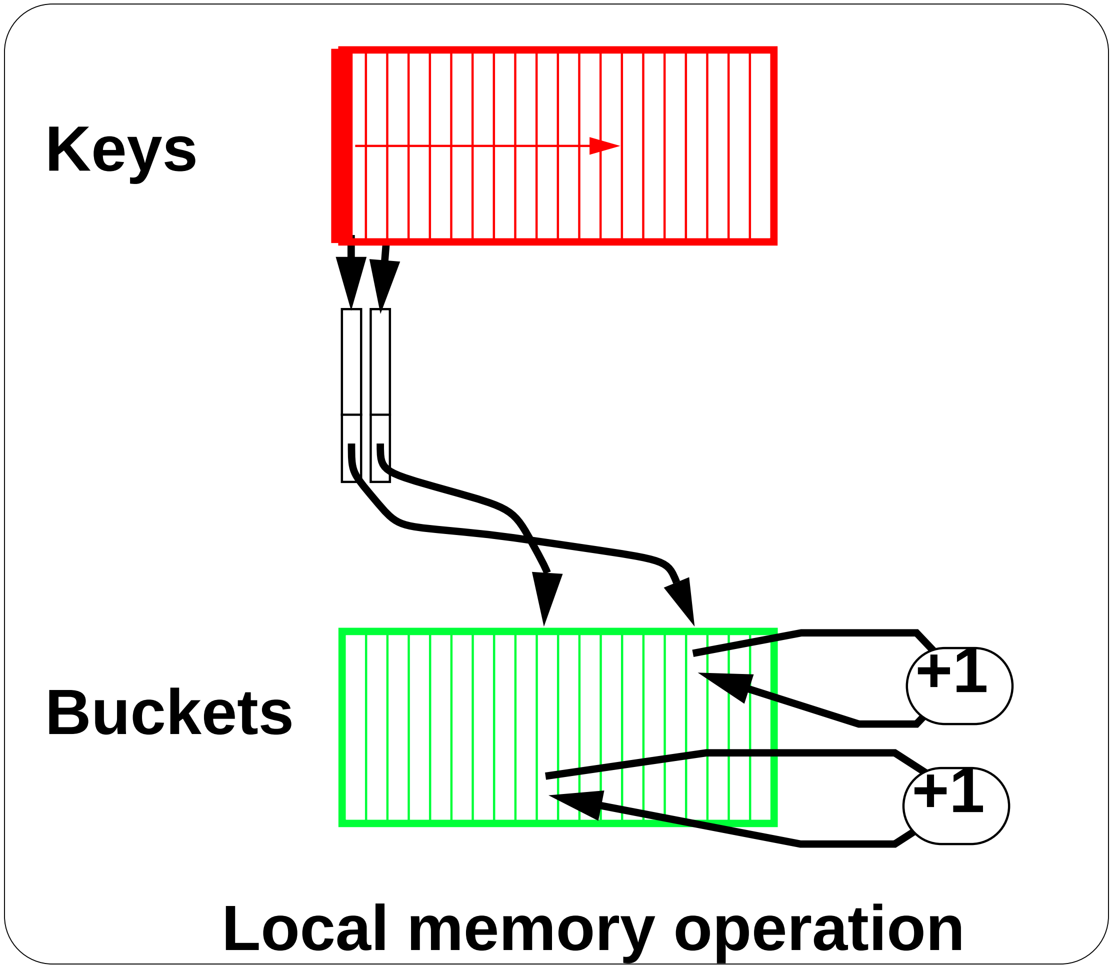
- for (i=0; i<radix_steps; i++)

  1: B=bucket_counts(A)
  2: transpose(B)
  3: B'=prefix_scan(B)
  4: transpose(B')
  5: A'=presort_copy(A,B')
  6: A=grouped_permute(A')

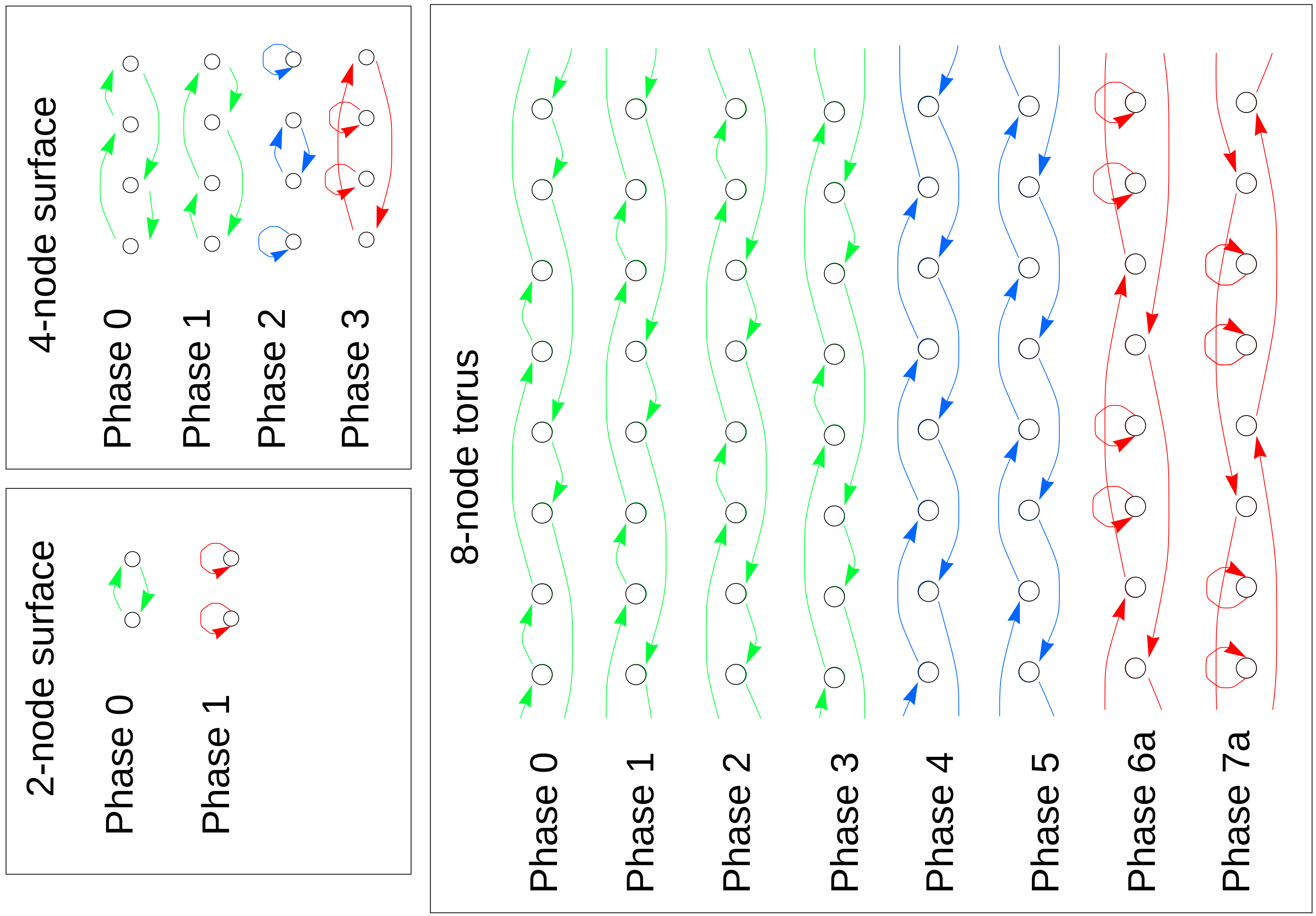
  Bucket work: steps 2,3,4
  Key work: steps 1,5,6

Analysis of sample sort:
Future work!

Index counting:
Bucket[Radix(a)]++



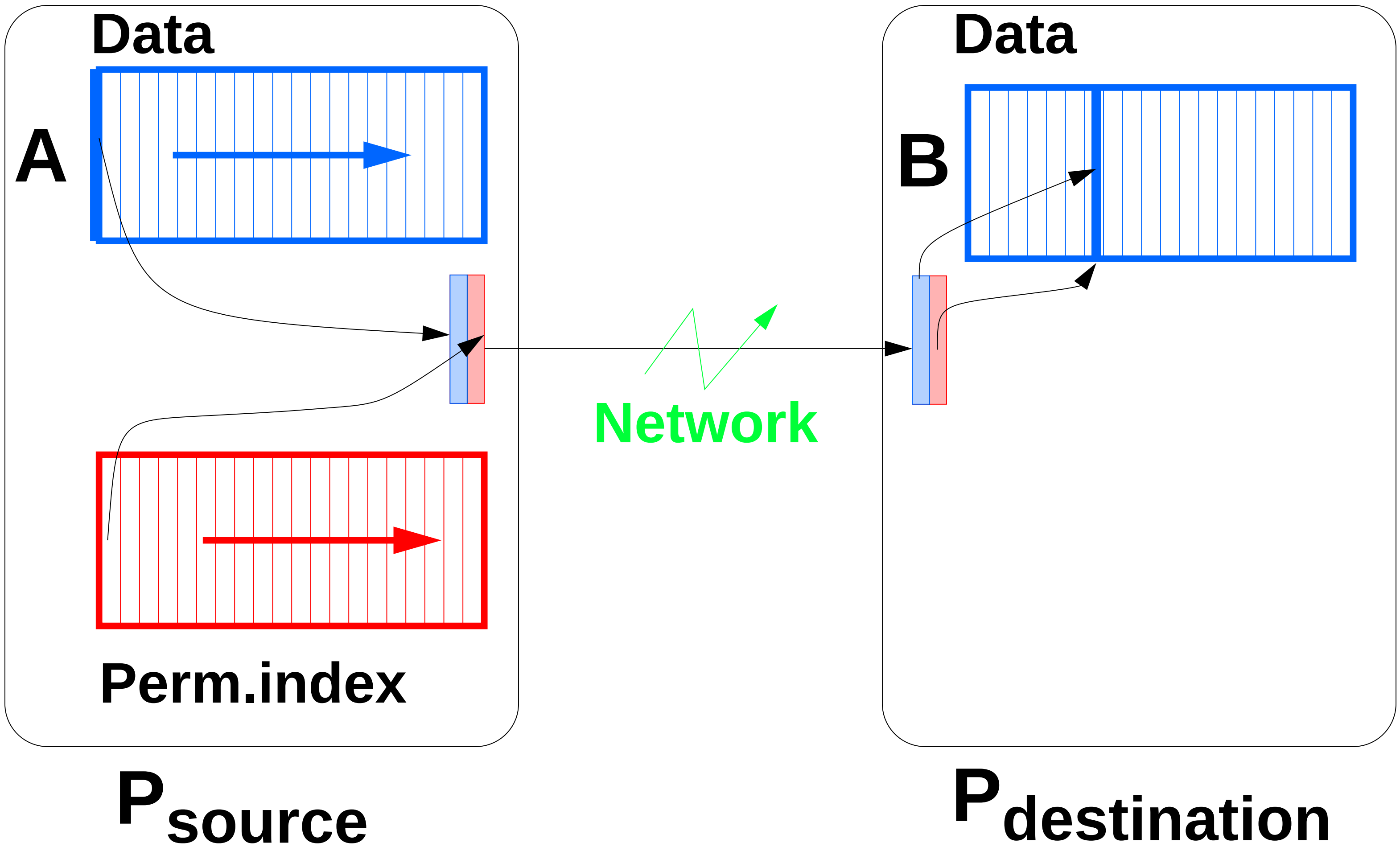
Patterns



Ref: Hinrichs et al. SPAA94

Grouped Permutation (B=Perm*[A])

(prearranged for a message passing machine)



AAPC: Algorithm - Router Interaction