



COSC 6365
Lecture 19
2008-03-25

CS@UH

Introduction to HPC

Lecture 19

Lennart Johnsson
Dept of Computer Science
Director TLC²



COSC 6365
Lecture 19
2008-03-25

CS@UH

Histogramming

- Each processor (core) has a copy of every bucket (embarrassingly parallel)
- For each bucket obtain a global count (reduction)
- Each summation can be parallelized (tree)
- Different bucket summations can be performed concurrently (all-to-all reduction)



COSC 6365
Lecture 19
2008-03-25

CS@UH

Performance

Complexity for P elements on N processors (cores), L buckets

Local histogram: P/N

Global Count: Arithmetic and data volume: $L - L/N$ for $L > N$

Steps: $\log_2 N$



Histogramming

Buckets ↑

0	8	16	24	32	40	48	56
1	9	17	25	33	41	49	57
2	10	18	26	34	42	50	58
3	11	19	27	35	43	51	59
4	12	20	28	36	44	52	60
5	13	21	29	37	45	53	61
6	14	22	30	38	46	54	62
7	15	23	31	39	47	55	63

→ Processors



Histogramming

0 ← 8 8	16 ← 24 40	32 ← 40 72	48 ← 56 104
1 → 9 10	17 → 25 42	33 → 41 74	49 → 57 106
2 ← 10 12	18 ← 26 44	34 ← 42 76	50 ← 58 108
3 → 11 14	19 → 27 46	35 → 43 78	51 → 59 110
4 ← 12 16	20 ← 28 48	36 ← 44 80	52 ← 60 112
5 → 13 18	21 → 29 50	37 → 45 82	53 → 61 114
6 ← 14 20	22 ← 30 52	38 ← 46 84	54 ← 62 116
7 → 15 22	23 → 31 54	39 → 47 86	55 → 63 118

Global counting – Step 1



Histogramming

0	8	16	24	32	40	48	56
48 ←				176 ←			
1	9	17	25	33	41	49	57
	52 ←				180 ←		
2	10	18	26	34	42	50	58
		56 →				184 →	
3	11	19	27	35	43	51	59
			60 →				188 →
4	12	20	28	36	44	52	60
64 ←				192 ←			
5	13	21	29	37	45	53	61
	68 ←				196 ←		
6	14	22	30	38	46	54	62
		72 →				200 →	
7	15	23	31	39	47	55	63
			76 →				204 →

Global counting – Step 2



Histogramming

0	8	16	24	32	40	48	56
224							
1	9	17	25	33	41	49	57
232							
2	10	18	26	34	42	50	58
240							
3	11	19	27	35	43	51	59
248							
4	12	20	28	36	44	52	60
256							
5	13	21	29	37	45	53	61
264							
6	14	22	30	38	46	54	62
272							
7	15	23	31	39	47	55	63
280							

Global counting – Step 3



COSC 6365
Lecture 19
2008-03-25

CS@UH

Bucket sort

1. Local assignment of elements to buckets
2. Global count of the number of elements in a bucket
3. Perform an accumulation of the counts for all preceding buckets (parallel prefix)
4. Rank assignment to each copy of each bucket
5. Local rank assignment
6. Permutation



Distribution (Bucket) Sort

Elements: 5, 0, 5, 0, 9, 1, 8, 2, 6, 4, 1, 5, 6, 6, 7, 7

Bucket	0	1	2	3	4	5	6	7	8	9
Counts	2	2	1	0	1	3	3	2	1	1
Accumulation	0	2	4	5	5	6	9	12	14	15

Elements: 5, 0, 5, 0, 9, 1, 8, 2, 6, 4, 1, 5, 6, 6, 7, 7

Rank: 6, 0, 7, 1, 15, 2, 14, 4, 9, 5, 3, 8, 10, 11, 12, 13



Distribution (Bucket) Sort

0	8	16	24	32	40	48	56
224							
1	9	17	25	33	41	49	57
	232						
2	10	18	26	34	42	50	58
		240					
3	11	19	27	35	43	51	59
			248				
4	12	20	28	36	44	52	60
				256			
5	13	21	29	37	45	53	61
					264		
6	14	22	30	38	46	54	62
						272	
7	15	23	31	39	47	55	63
							280

Result of Bucket count

0	8	16	24	32	40	48	56
1	9	17	25	33	41	49	57
	224						
2	10	18	26	34	42	50	58
		456					
3	11	19	27	35	43	51	59
			696				
4	12	20	28	36	44	52	60
				944			
5	13	21	29	37	45	53	61
					1200		
6	14	22	30	38	46	54	62
						1464	
7	15	23	31	39	47	55	63
							1736

Result of global accumulation



Distribution (Bucket) Sort

48	8	16	24	176	40	48	56
224							
1	52	17	25	33	180	49	57
	224						
2	10	56	26	34	42	184	58
		456					
3	11	19	60	35	43	51	188
			696				
64	12	20	28	192	44	52	60
			944				
5	68	21	29	37	196	53	61
				1200			
6	14	72	30	38	46	200	62
					1464		
7	15	23	76	39	47	55	204
							1736

Result of global accumulation

48	8	16	24	176	40	48	56	
0	→							48
1	52	17	25	33	180	49	57	
	224	→						276
2	10	56	26	34	42	184	58	
		456	→					512
3	11	19	60	35	43	51	188	
			696	→				756
64	12	20	28	192	44	52	60	
944	←							1008
5	68	21	29	37	196	53	61	
	1200	←						1268
6	14	72	30	38	46	200	62	
		1464	←					1536
7	15	23	76	39	47	55	204	
			1736	←				1812

Rank assignment – Step 1



Distribution (Bucket) Sort

Comparison of global counting and rank assignment steps

0	8	16	24	32	40	48	56
224							
1	9	17	25	33	41	49	57
232							
2	10	18	26	34	42	50	58
240							
3	11	19	27	35	43	51	59
248							
4	12	20	28	36	44	52	60
				256			
5	13	21	29	37	45	53	61
					264		
6	14	22	30	38	46	54	62
						272	
7	15	23	31	39	47	55	63
							280

Global counting – Step 3

48	8	16	24	176	40	48	56
0				48			
1	52	17	25	33	180	49	57
	224				276		
2	10	56	26	34	42	184	58
		456				512	
3	11	19	60	35	43	51	188
			696				756
64	12	20	28	192	44	52	60
944				1008			
5	68	21	29	37	196	53	61
	1200				1268		
6	14	72	30	38	46	200	62
		1464				1536	
7	15	23	76	39	47	55	204
			1736				1812

Rank assignment – Step 1



Distribution (Bucket) Sort

8	8	40	24	72	40	104	56
0				48			
1	10	17	42	33	74	49	106
	224				276		
12	10	44	26	76	42	108	58
		456				512	
3	14	19	46	35	78	51	110
			696				756
16	12	48	28	80	44	112	60
944				1008			
5	18	21	50	37	82	53	114
	1200				1268		
20	14	52	30	84	46	116	62
		1464				1536	
7	22	23	54	39	86	55	118
			1736				1812

Rank assignment – Step 1

8	8	40	24	72	40	104	56
0	→ 8			48	→ 120		
1	10	17	42	33	74	49	106
	224	→ 234			276	→ 350	
12	10	44	26	76	42	108	58
456	← 468			512	← 588		
3	14	19	46	35	78	51	110
	696	← 710			756	← 834	
16	12	48	28	80	44	112	60
944	→ 960			1008	→ 1088		
5	18	21	50	37	82	53	114
	1200	→ 1218			1268	→ 1350	
20	14	52	30	84	46	116	62
1464	← 1484			1536	← 1620		
7	22	23	54	39	86	55	118
	1735	← 1757			1312	← 1898	

Rank assignment – Step 2



Distribution (Bucket) Sort

Comparison of global counting and rank assignment steps

0	8	16	24	32	40	48	56
48	←			176	←		
1	9	17	25	33	41	49	57
	52	←			180	←	
2	10	18	26	34	42	50	58
	→ 56				→ 184		
3	11	19	27	35	43	51	59
	→ 60				→ 188		
4	12	20	28	36	44	52	60
64	←			192	←		
5	13	21	29	37	45	53	61
	68	←			196	←	
6	14	22	30	38	46	54	62
	→ 72				→ 200		
7	15	23	31	39	47	55	63
	→ 76				→ 204		

Global counting – Step 2

8	8	40	24	72	40	104	56
0	→ 8			48	→ 120		
1	10	17	42	33	74	49	106
	224	→ 234			276	→ 350	
12	10	44	26	76	42	108	58
456	← 468			512	← 588		
3	14	19	46	35	78	51	110
	696	← 710			756	← 834	
16	12	48	28	80	44	112	60
944	→ 960			1008	→ 1088		
5	18	21	50	37	82	53	114
	1200	→ 1218			1268	→ 1350	
20	14	52	30	84	46	116	62
1464	← 1484			1536	← 1620		
7	22	23	54	39	86	55	118
	1736	← 1758			1812	← 1898	

Rank assignment – Step 2



Distribution (Bucket) Sort

0 ← 8 8	16 ← 24 40	32 ← 40 72	48 ← 56 104
1 → 9 10	17 → 25 42	33 → 41 74	49 → 57 106
2 ← 10 12	18 ← 26 44	34 ← 42 76	50 ← 58 108
3 → 11 14	19 → 27 46	35 → 43 78	51 → 59 110
4 ← 12 16	20 ← 28 48	36 ← 44 80	52 ← 60 112
5 → 13 18	21 → 29 50	37 → 45 82	53 → 61 114
6 ← 14 20	22 ← 30 52	38 ← 46 84	54 ← 62 116
7 → 15 22	23 → 31 54	39 → 47 86	55 → 63 118

Global counting – Step 1

0 → 0 0	8 → 24 8	16 → 40 16	24 → 56 24	32 → 80 32	40 → 112 40	48 → 144 48	56 → 176 56
1	9	17	25	33	41	49	57
224 ← 225	234 ← 251	276 ← 309	350 ← 399				
2	10	18	26	34	42	50	58
456 → 458	468 → 486	512 → 546	588 → 638				
3	11	19	27	35	43	51	59
696 ← 699	710 ← 729	756 ← 791	834 ← 885				
4	12	20	28	36	44	52	60
944 → 948	960 → 980	1008 → 1044	1088 → 1140				
5	13	21	29	37	45	53	61
1200 ← 1205	1218 ← 1239	1268 ← 1305	1350 ← 1403				
6	14	22	30	38	46	54	62
1464 → 1470	1484 → 1506	1536 → 1574	1620 → 1674				
7	15	23	31	39	47	55	63
1736 ← 1743	1758 ← 1781	1812 ← 1851	1898 ← 1953				

Rank assignment – Step 3



Distribution (Bucket) Sort

Complexity (approximate) for $L > N$:

Step 1: P/N

Step 2: $L - L/N$

Step 3: $2L/N$

Step 4: $L - L/N$

Step 5: P/N

Total: $2P/N + 2(L - L/N) + 2L/N$

Sequential time: $2P + L - 1$

Speed-up:
$$\frac{2P + L - 1}{2P/N + 2(L - L/N) + 2L/N}$$

For $L \gg N$ and $P/N \gg L$ Speedup is $O(N)$

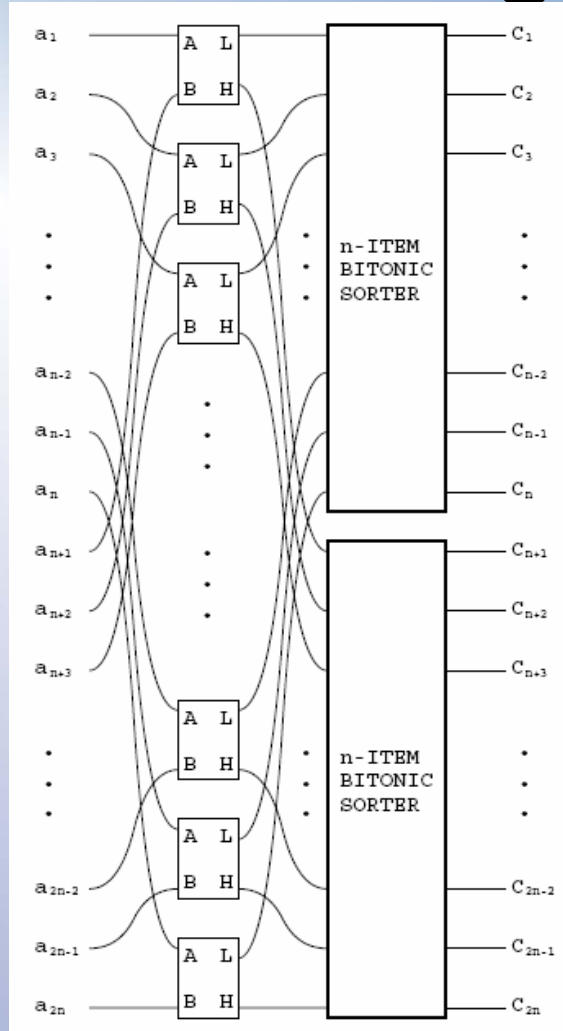
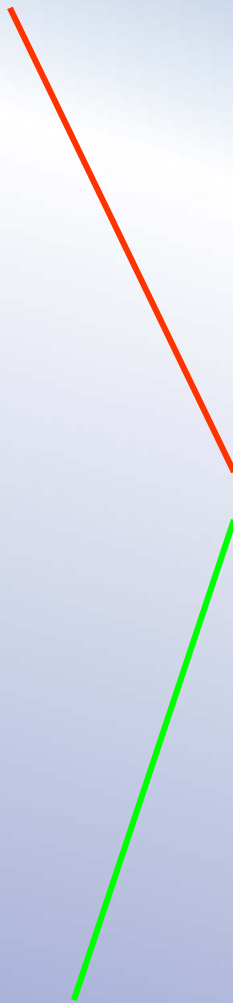


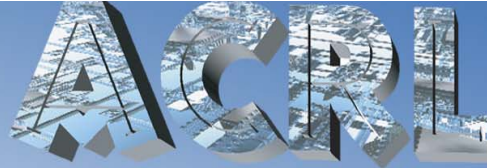
ADVANCED COMPUTING RESEARCH LABORATORY

COSC 6365
Lecture 19
2008-03-25

CS@UH

Bitonic merge

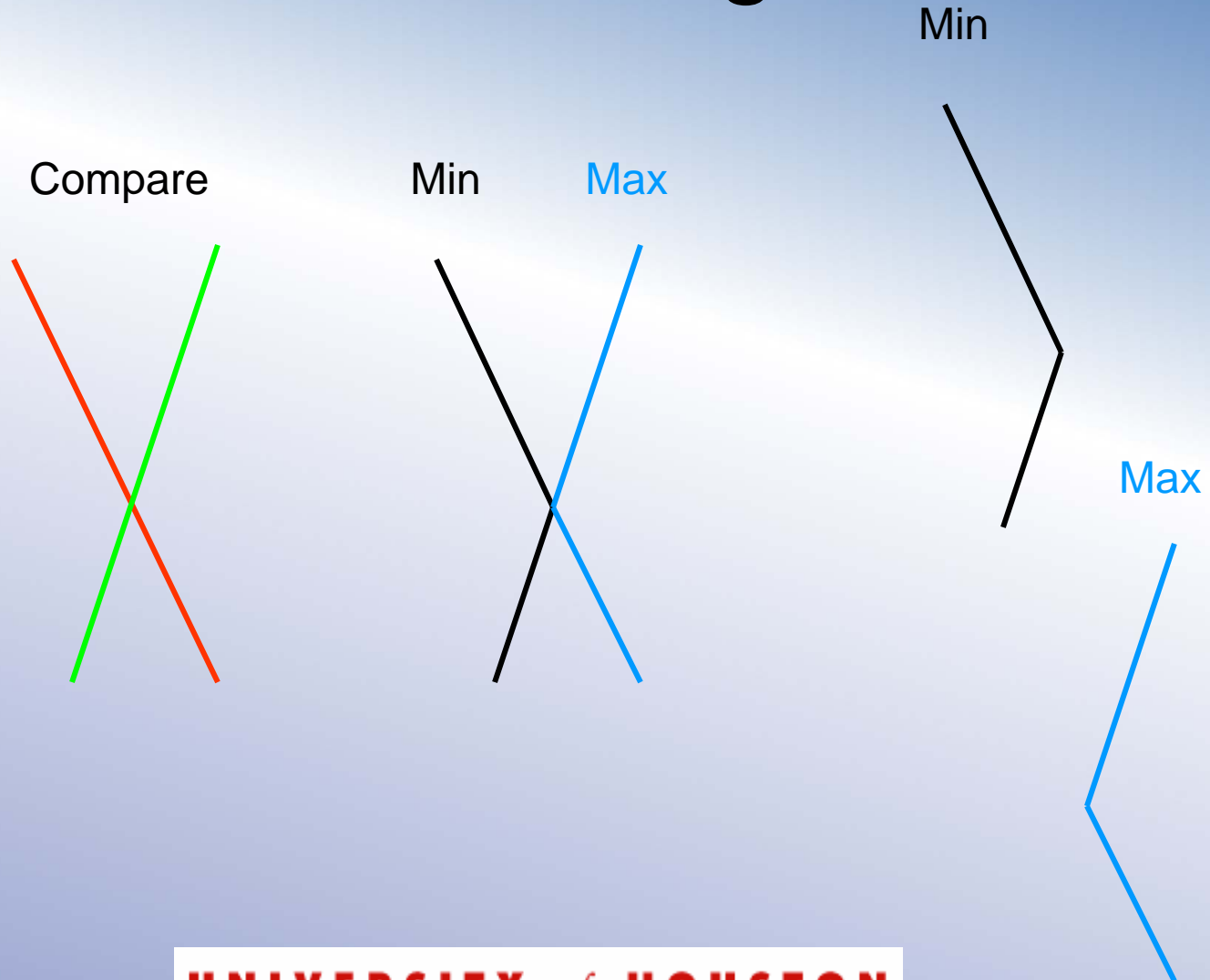




COSC 6365
Lecture 19
2008-03-25

CS@UH

Bitonic Merge



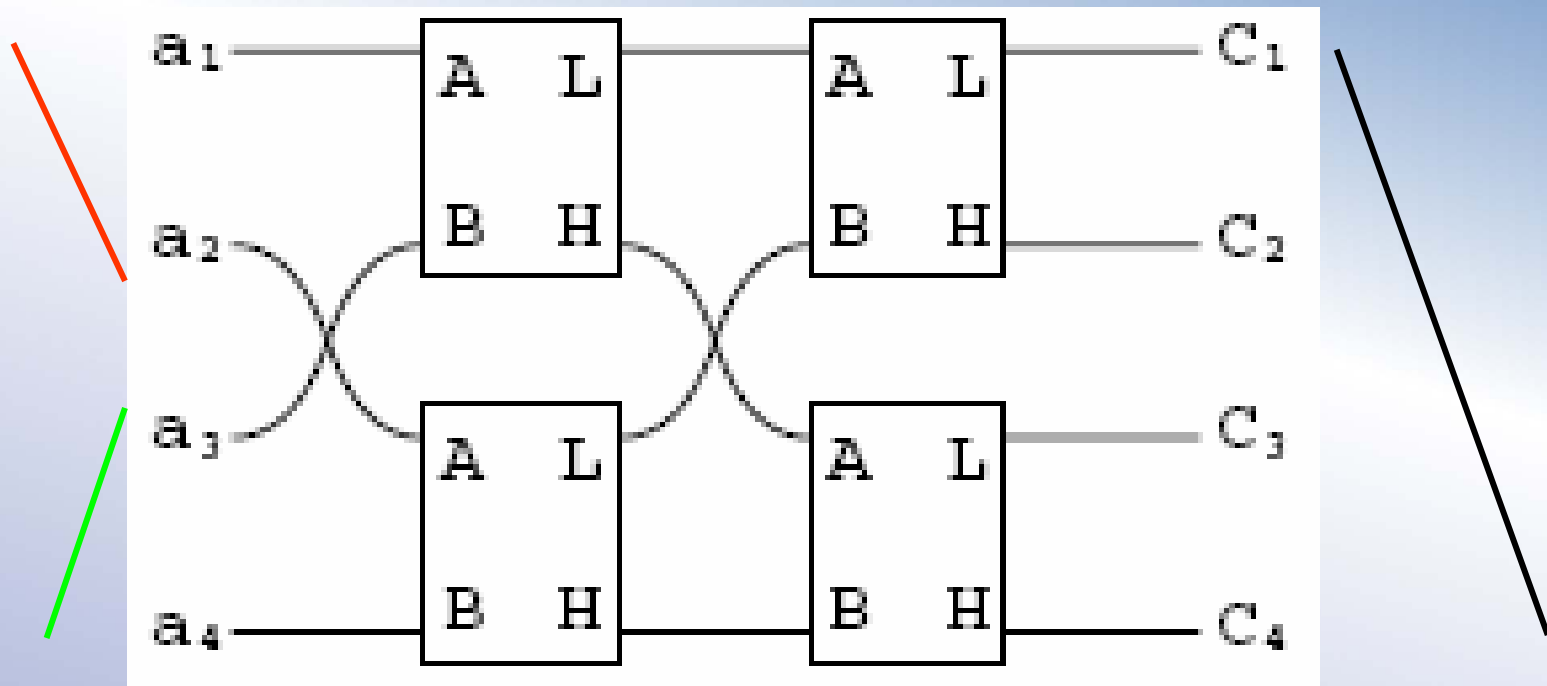


ADVANCED COMPUTING RESEARCH LABORATORY

COSC 6365
Lecture 19
2008-03-25

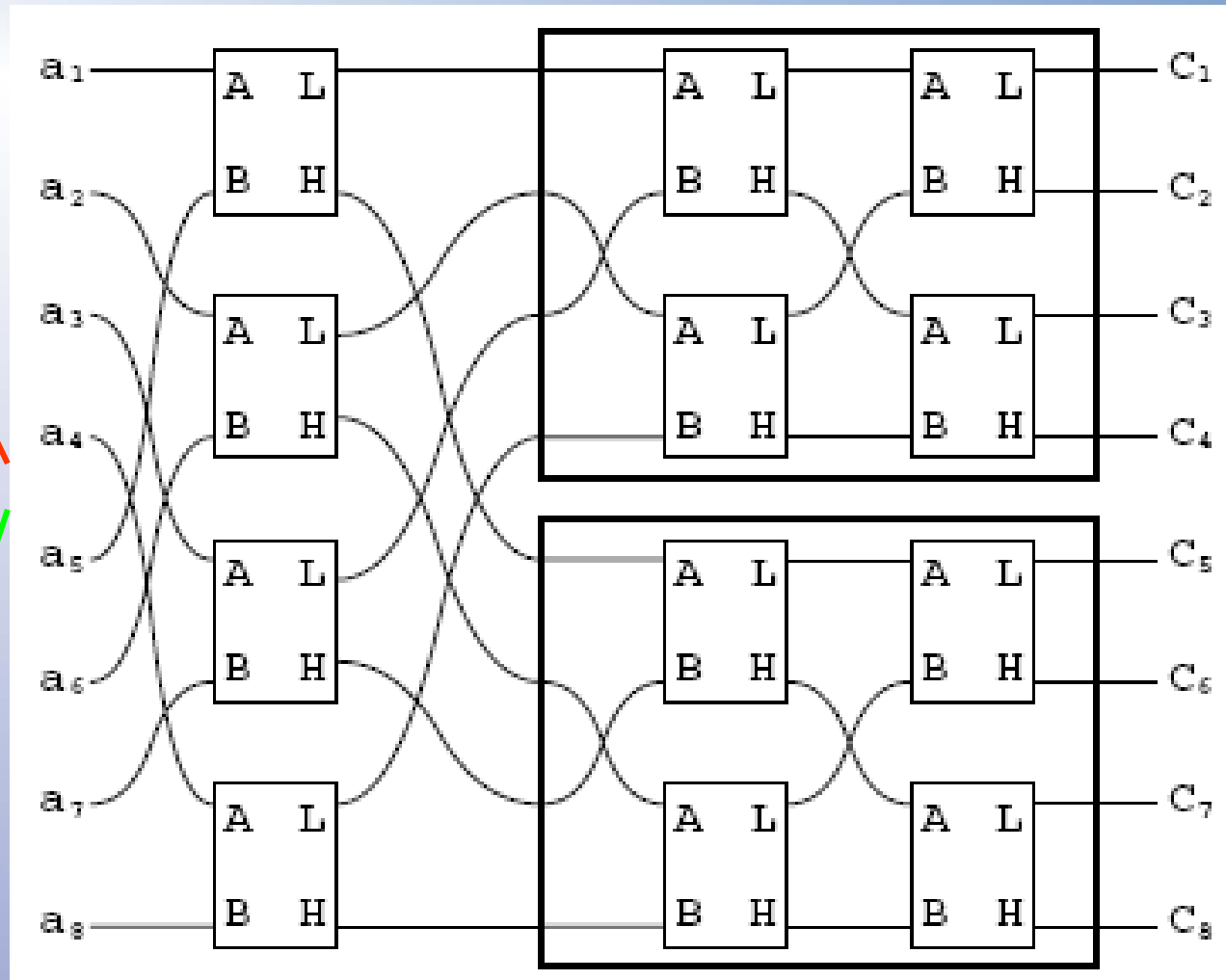
CS@UH

Bitonic Merge





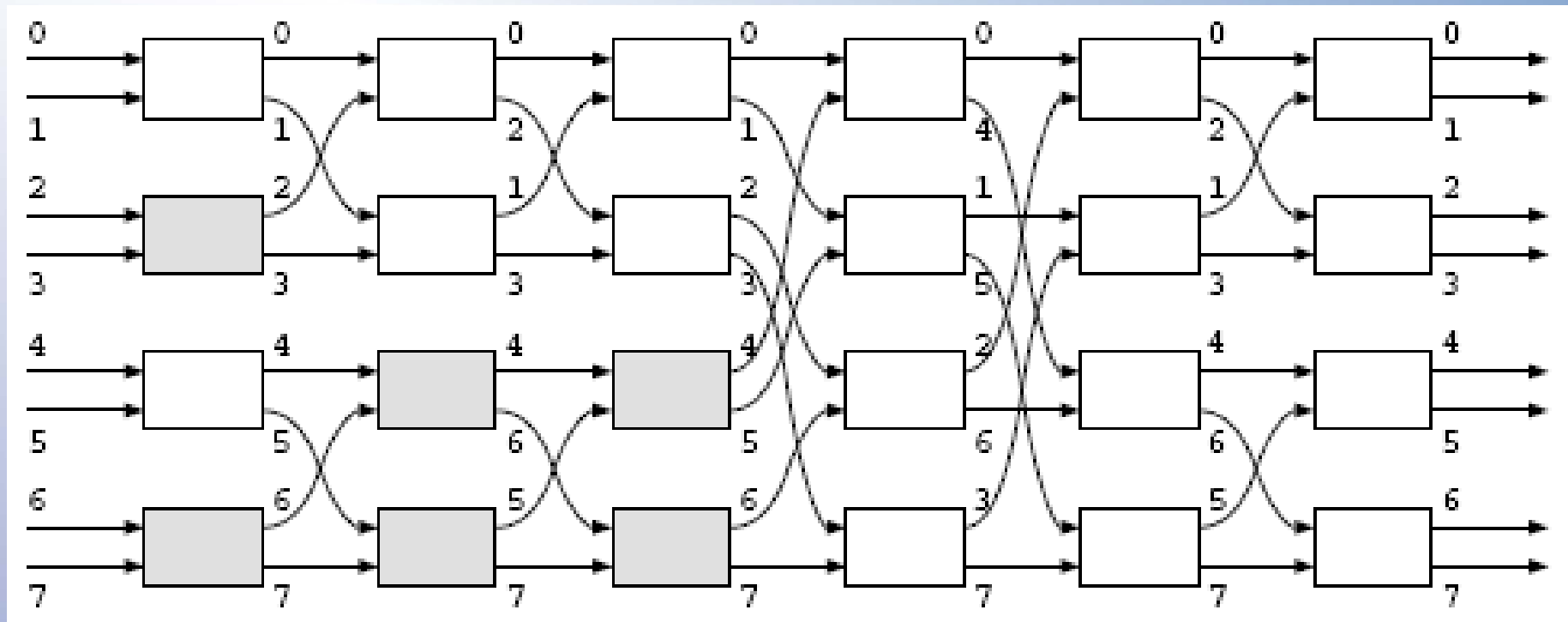
Bitonic recursive merge







Bitonic sort

(based on recursive application of bitonic merge)



 Sorts in opposite order compared to 



COSC 6365
Lecture 19
2008-03-25

CS@UH

Bitonic Sort

Complexity:

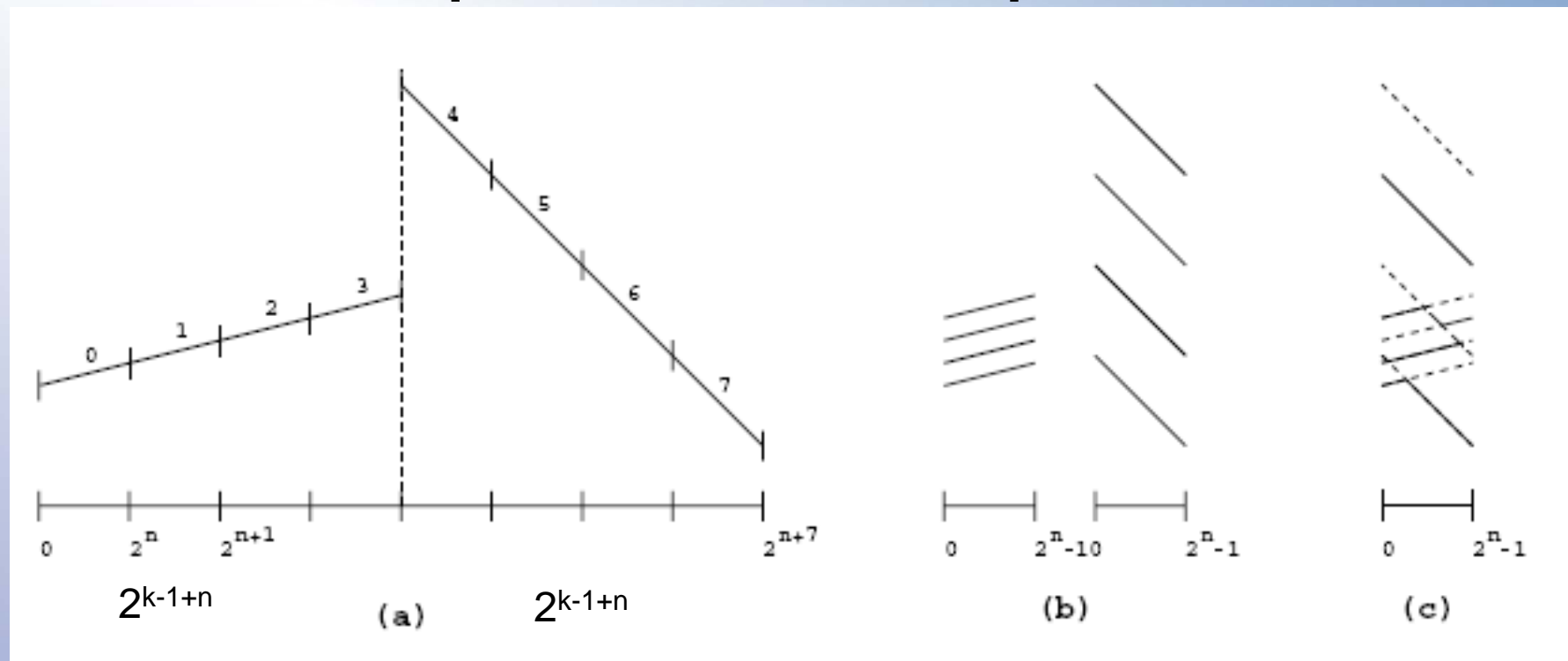
Bitonic merge: $N \log_2 N$

Bitonic Sort: $N(\log_2 N)^2$



Bitonic Sort

Multiple elements per core



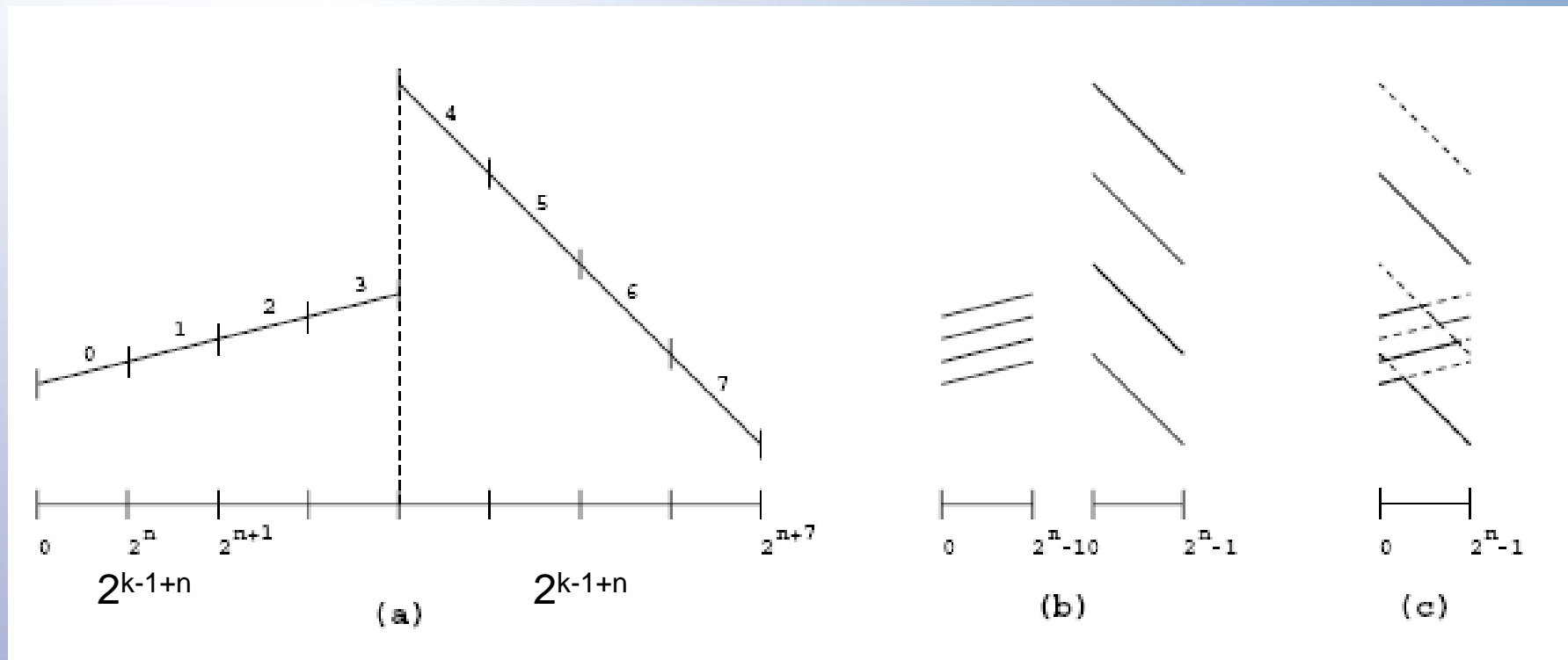
First k steps local: 2^{k-1} comparisons per core per step.

Last n steps local: 2^k bitonic sequences each with 1 elem. per core.



Bitonic Sort

Multiple elements per core



Time: $k2^{k-1} + 2^k n$ (or $\sim P/N \log_2(P/N) + P/N \log_2 N = P/N \log_2 P$).



COSC 6365
Lecture 19
2008-03-25

CS@UH

Sequential Merge

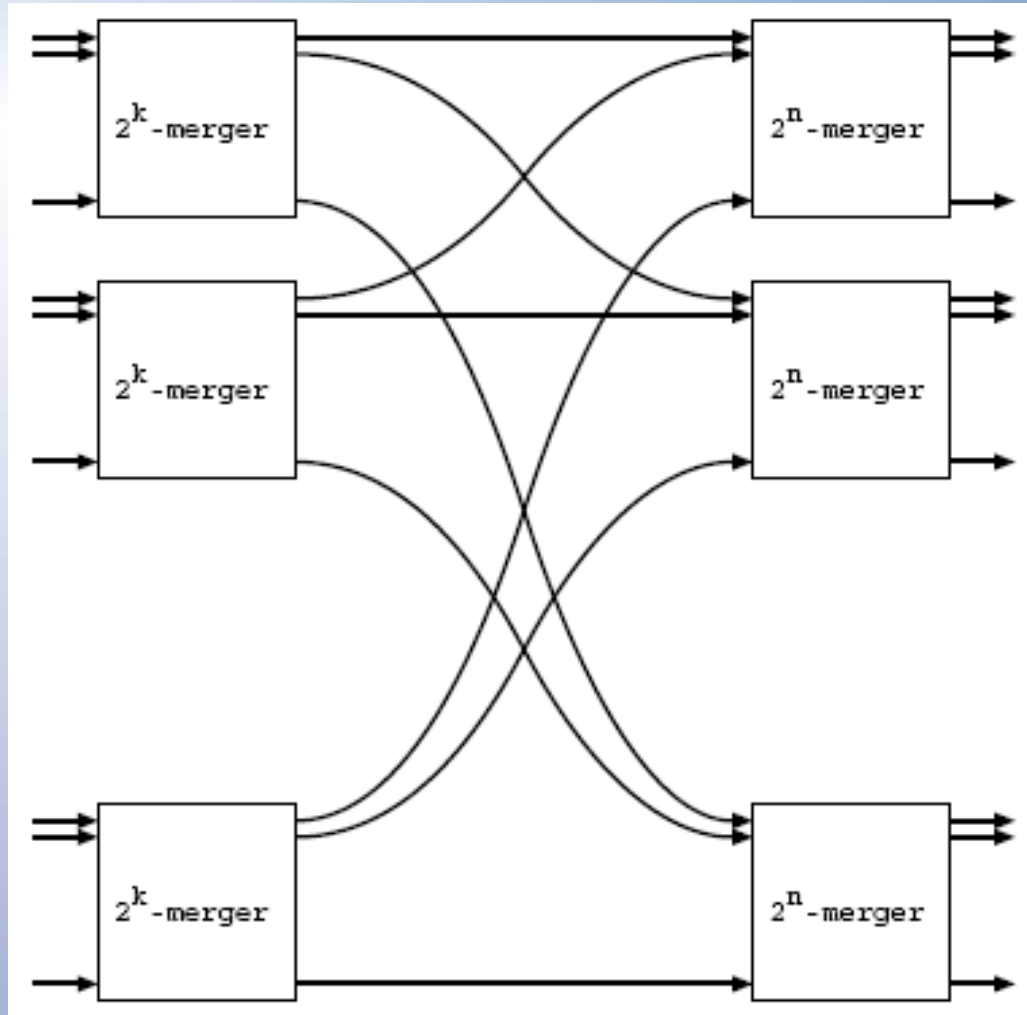
Complexity: P for P elements

Bitonic Merge: $P \log_2 P$

Inefficient by a factor of $\log_2 P$



Hybrid Merge



2^n independent
mergers of sets
of 2^k elements
each

Each set local
to a core

2^k independent
mergers of sets
of 2^n elements
each

Each set
distributed with
one element per
core



Radix Sort

For bits in the key in ascending order, $i = 0$ to $p-1$
rank with respect to bit i
permute according to rank on bit i
Endfor

Initial order		Step 1 (lsb)		Step 2		Step 3 (msb)	
Decimal	Binary	Rank	Permute	Rank	Permute	Rank	Permute
5	101						
6	110						
4	100						
7	111						
6	110						
2	010						
1	001						
0	000						



Radix Sort

For bits in the key in ascending order, $i = 0$ to $p-1$
 rank with respect to bit i
 permute according to rank on bit i
Endfor

Initial order		Step 1 (lsb)		Step 2		Step 3 (msb)	
Decimal	Binary	Rank	Permute	Rank	Permute	Rank	Permute
5	101	6					
6	110	1					
4	100	2					
7	111	7					
6	110	3					
2	010	4					
1	001	8					
0	000	5					



Radix Sort

For bits in the key in ascending order, $i = 0$ to $p-1$
rank with respect to bit i
permute according to rank on bit i
Endfor

Initial order		Step 1 (lsb)		Step 2		Step 3 (msb)	
Decimal	Binary	Rank	Permute	Rank	Permute	Rank	Permute
5	101	6	110				
6	110	1	100				
4	100	2	110				
7	111	7	010				
6	110	3	000				
2	010	4	101				
1	001	8	111				
0	000	5	001				



Radix Sort

For bits in the key in ascending order, $i = 0$ to $p-1$
rank with respect to bit i
permute according to rank on bit i
Endfor

Initial order		Step 1 (lsb)		Step 2		Step 3 (msb)	
Decimal	Binary	Rank	Permute	Rank	Permute	Rank	Permute
5	101	6	110	5			
6	110	1	100	1			
4	100	2	110	6			
7	111	7	010	7			
6	110	3	000	2			
2	010	4	101	3			
1	001	8	111	8			
0	000	5	001	4			



Radix Sort

For bits in the key in ascending order, $i = 0$ to $p-1$
 rank with respect to bit i
 permute according to rank on bit i
Endfor

Initial order		Step 1 (lsb)		Step 2		Step 3 (msb)	
Decimal	Binary	Rank	Permute	Rank	Permute	Rank	Permute
5	101	6	110	5	100		
6	110	1	100	1	000		
4	100	2	110	6	101		
7	111	7	010	7	001		
6	110	3	000	2	110		
2	010	4	101	3	110		
1	001	8	111	8	010		
0	000	5	001	4	111		



Radix Sort

For bits in the key in ascending order, $i = 0$ to $p-1$
 rank with respect to bit i
 permute according to rank on bit i
Endfor

Initial order		Step 1 (lsb)		Step 2		Step 3 (msb)	
Decimal	Binary	Rank	Permute	Rank	Permute	Rank	Permute
5	101	6	110	5	100	4	
6	110	1	100	1	000	1	
4	100	2	110	6	101	5	
7	111	7	010	7	001	2	
6	110	3	000	2	110	6	
2	010	4	101	3	110	7	
1	001	8	111	8	010	3	
0	000	5	001	4	111	8	



Radix Sort

For bits in the key in ascending order, $i = 0$ to $p-1$
rank with respect to bit i
permute according to rank on bit i
Endfor

Initial order		Step 1 (lsb)		Step 2		Step 3 (msb)	
Decimal	Binary	Rank	Permute	Rank	Permute	Rank	Permute
5	101	6	110	5	100	4	000
6	110	1	100	1	000	1	001
4	100	2	110	6	101	5	010
7	111	7	010	7	001	2	100
6	110	3	000	2	110	6	101
2	010	4	101	3	110	7	110
1	001	8	111	8	010	3	110
0	000	5	001	4	111	8	111



Radix Sort

Cycles per element on a CM-5 with 64k elements per vector unit.

Key distribution:
Uniform, Random

Radix	Each instance		All instances		Total
	Rank	Permute	Rank	Permute	
4	7.63	6.71	61.02	53.69	114.71
5	7.63	6.71	53.39	46.98	100.37
6	7.63	6.71	45.76	40.27	86.03
7	7.63	6.71	38.14	33.55	71.69
8	10.91	6.71	43.65	26.84	70.49
9	12.38	6.71	44.75	26.84	71.59
10	13.09	6.71	46.90	26.84	73.74
11	13.39	6.71	39.86	20.13	59.99
12	13.48	6.71	37.87	20.13	58.00
13	13.48	6.71	34.59	20.13	54.72
14	13.48	6.71	34.58	20.13	54.71

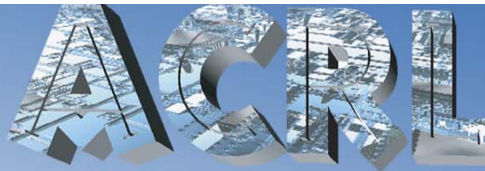


COSC 6365
Lecture 19
2008-03-25

CS@UH

Sample Sort

- Select a set of Splitters (at least $N-1$ for N processors)
- Sort the splitters
- Assign elements to buckets
- Permute
- Local sort



ADVANCED COMPUTING RESEARCH LABORATORY

COSC 6365
Lecture 19
2008-03-25

CS@UH

Sample Sort

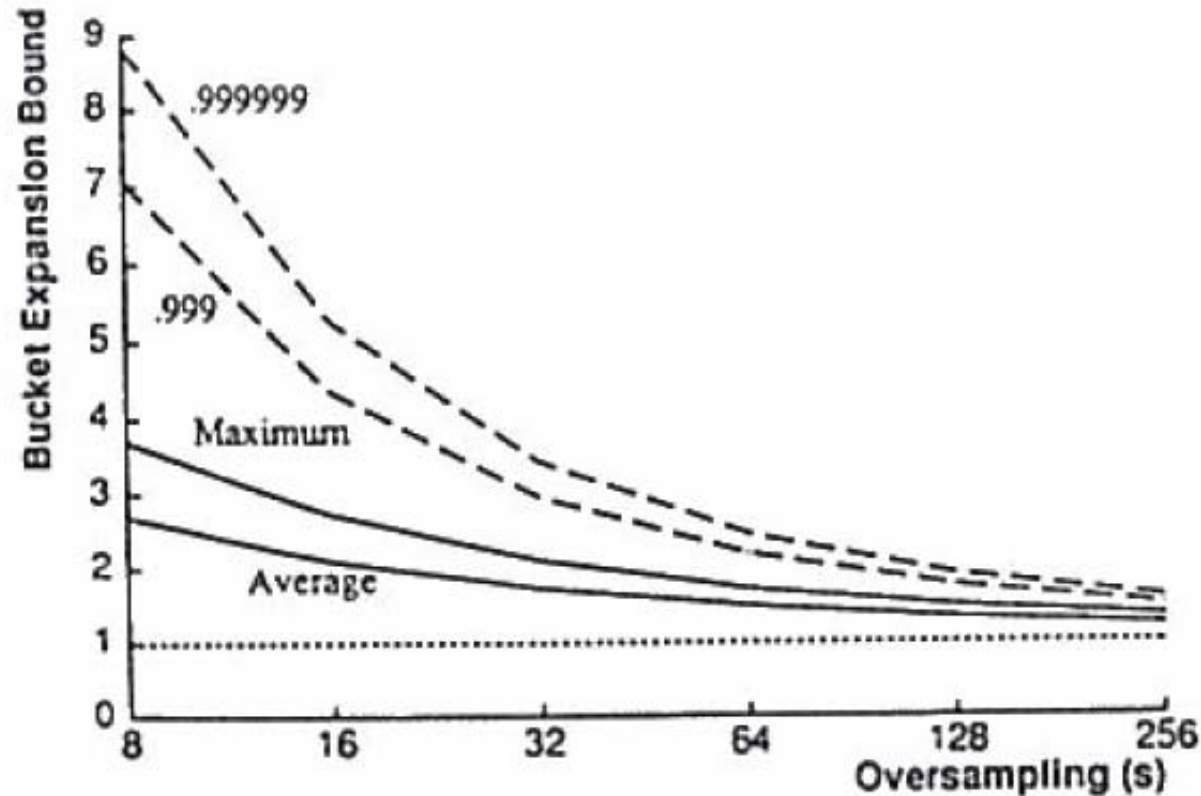
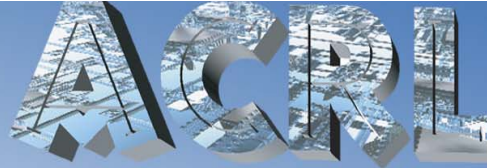


Figure 10: The relationship between bucket expansion and oversampling. The two upper curves shows the bucket expansions that are not exceeded with a probability of of 0.999 and 0.999999 respectively for one million keys and 1024 nodes. The two lower curves shows the observed maximum and average expansions for a 1000 trials.



ADVANCED COMPUTING RESEARCH LABORATORY

COSC 6365
Lecture 19
2008-03-25

CS@UH

Sample Sort

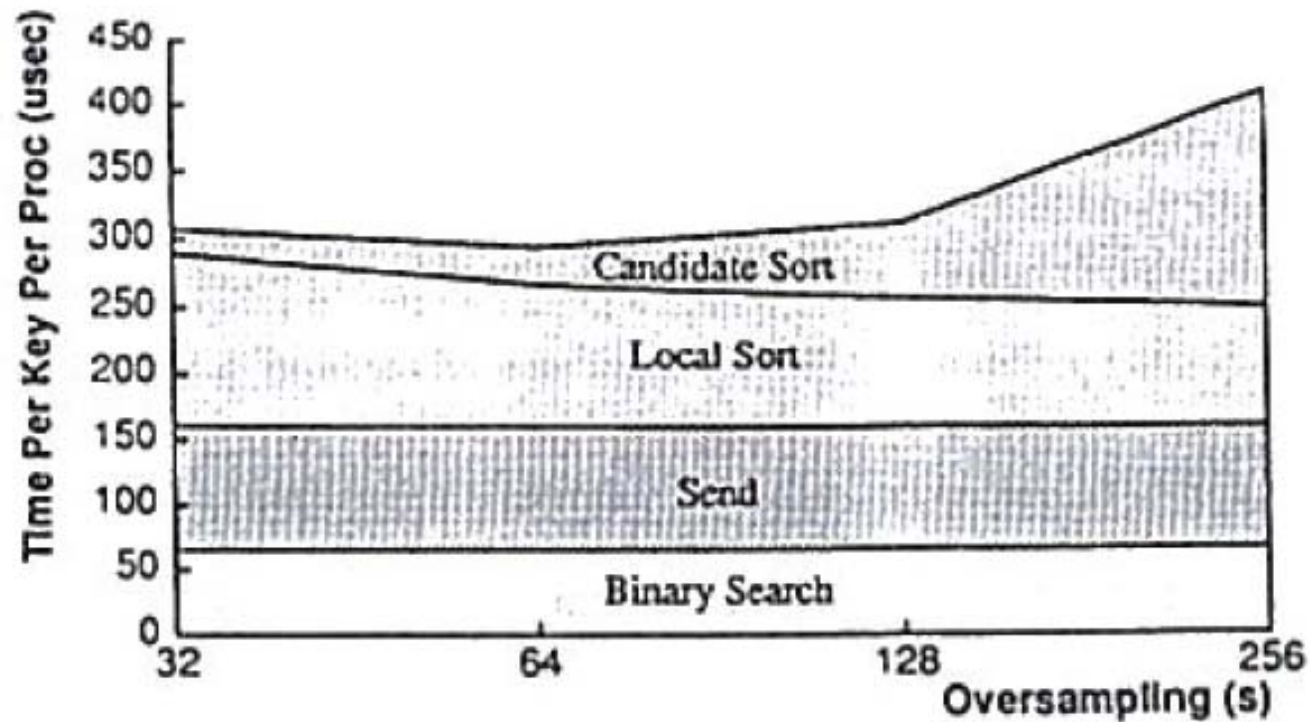
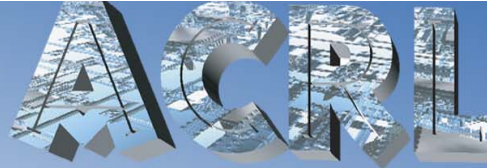


Figure 11: Sample sort time as a function of the oversampling ratio for 16384 keys per node of a 1024 node CM-2.



ADVANCED COMPUTING RESEARCH LABORATORY

COSC 6365
Lecture 19
2008-03-25

CS@UH

Sample Sort

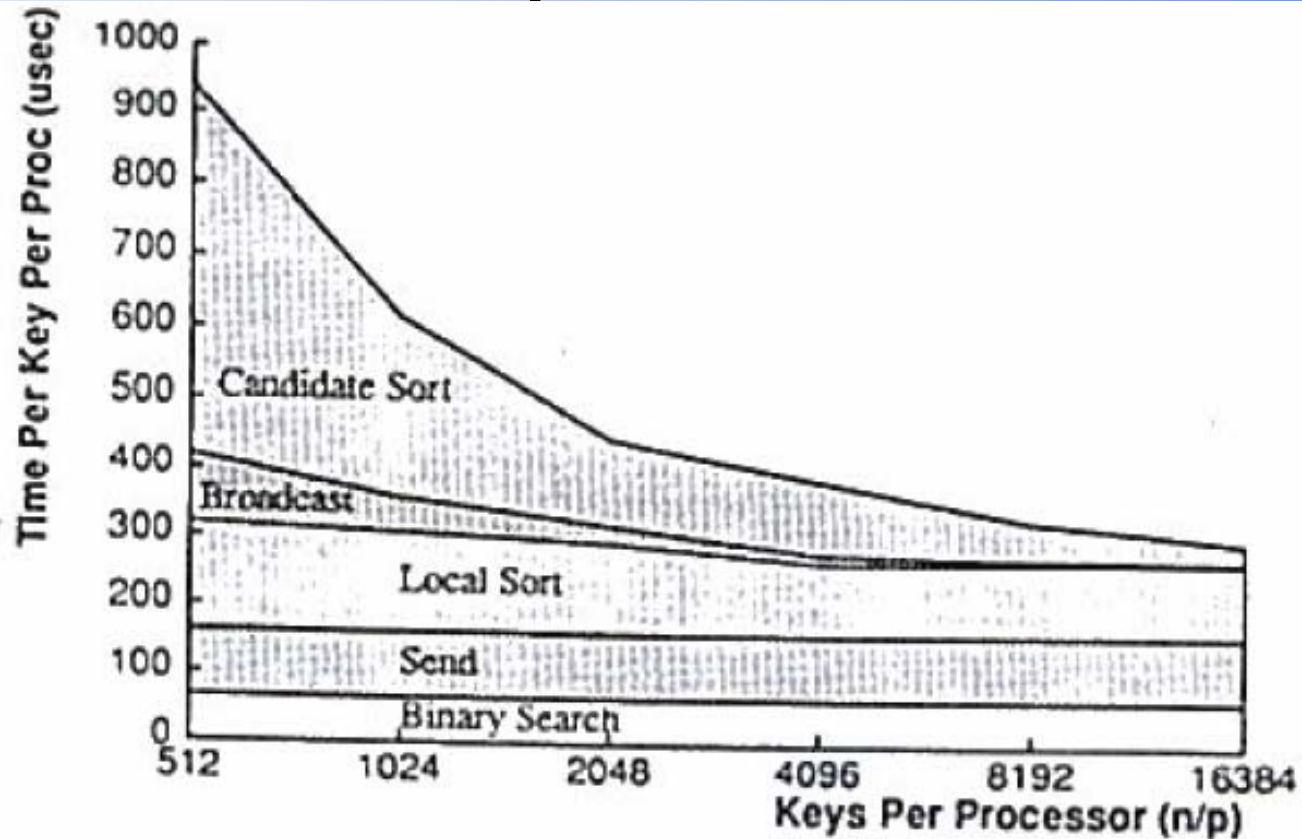


Figure 12: Sample sort time as a function of the number of keys per node for a 1024 node CM-2. Oversampling ratio 32 for up to 4096 keys per node. Oversampling ratio 64 for more than 4096 keys per node.



Sample Sort

Method	Stable	Load Balance	Time/key/node (msec)		Memory	Rank
			P/N=64	P/N=16,384		
Bitonic	No	Yes	1.60	2.20	1.0	1.5
Radix	Yes	Yes	2.40	0.95	2.1	1.0
Sample	No	No	5.50	0.33	3.2	1.5

Sorting times on a 1024 node CM-5



COSC 6365
Lecture 19
2008-03-25

CS@UH

Valiant's Parallel Merge Sort

- Merge two sorted sequences P and R in parallel by splitting each sequence evenly in $\sqrt{\text{size}}$ chunks.
- Sequence P : chunk size \sqrt{P} , \sqrt{P} chunks
- Sequence R : chunk size \sqrt{R} , \sqrt{R} chunks
- Merge the \sqrt{P} and \sqrt{R} splitters
- Insert the \sqrt{P} splitters into the proper chunk of R
- Repeat the process recursively for each sublist created by the insertion



COSC 6365
Lecture 19
2008-03-25

CS@UH

Valiant's Parallel Merge sort

- Merge the $\sqrt{P} + \sqrt{R}$ splitters requires $\sqrt{P} \times \sqrt{R}$ comparisons which can be carried out on $\sqrt{P} \times \sqrt{R}$ cores in unit time.
- Second step the same ...
- And the third see notes