Experimental results on Cartesius

Section 4.12 of Parallel Scientific Computation, 2nd edition

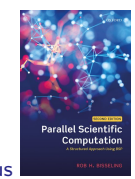
Rob H. Bisseling

Utrecht University

Test set of sparse matrices

Name	n	nz	c	Origin
`mip1`	66 463	10 352 819	155.8	mixed integer progr.
`in-2004`	1 382 908	16 917 053	12.2	web links India 2004
`asia_osm`	11 950 757	25 423 206	2.1	road network Asia
`cage14`	1 505 785	27 130 349	18.0	DNA electrophoresis
`rgg_n_2_21_s0`	2 097 152	28 975 990	13.8	random geometric graph
`laplace2D_4096`	16 777 216	83 869 696	5.0	2D Laplacian
`random100k`	100 000	100 004 125	1 000.0	random sparse matrix
`laplace3D_256`	16 777 216	117 047 296	7.0	3D Laplacian

- Random geometric graph with 2^{21} randomly placed points in a unit square interacting within a cut-off radius.
- Laplacian matrices: 4096×4096 2D grid and $256 \times 256 \times 256$ 3D grid, both with 2^{24} grid points.
- Random sparse matrix with density $d = 0.01$.

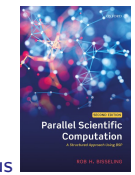


BSP cost for the smallest matrices

p	mip1	in-2004	asia_osm
1	20 705 638	33 834 106	50 846 412
2	$10\,635\,008 + 1\,087g$	$17\,402\,680 + 319g$	$25\,582\,806 + 62g$
4	$5\,321\,972 + 2\,152g$	$8\,619\,792 + 611g$	$12\,847\,854 + 62g$
8	$2\,665\,180 + 3\,539g$	$4\,354\,756 + 832g$	$6\,526\,308 + 82g$
16	$1\,332\,106 + 2\,603g$	$2\,178\,062 + 867g$	$3\,272\,994 + 97g$
32	$666\,428 + 2\,740g$	$1\,089\,034 + 832g$	$1\,636\,606 + 143g$
64	$333\,156 + 2\,000g$	$544\,514 + 756g$	$818\,266 + 96g$
128	$166\,614 + 1\,640g$	$272\,258 + 777g$	$409\,110 + 99g$
256	$83\,304 + 1\,380g$	$136\,128 + 588g$	$204\,564 + 132g$

- The matrices and vectors were partitioned by Mondriaan, version 4.2.1 with the default medium-grain method; $\epsilon = 3\%$.
- The fixed synchronization cost $4l$ is not shown. For $p = 2$ on Cartesius, this is of the order 100 000.
- To make parallelism worthwhile, $T_2 \leq T_{\text{seq}}$ should hold, and hence at least $2nz/2 + 4l \leq 2nz$, i.e.,

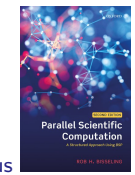
$$nz \geq 4l.$$



BSP cost for larger matrices

p	cage14	rgg_n_2_21_s0	laplace2D_4096
1	54 260 698	57 951 980	167 739 392
2	27 166 388 + 74 019g	29 832 972 + 1 578g	84 777 274 + 4 096g
4	13 565 176 + 99 763g	14 569 690 + 1 602g	43 065 966 + 4 634g
8	6 890 846 + 74 273g	7 440 620 + 1 784g	21 577 598 + 7 741g
16	3 391 294 + 49 716g	3 728 966 + 1 587g	10 797 944 + 5 956g
32	1 718 302 + 34 948g	1 864 464 + 1 409g	5 397 110 + 4 600g
64	859 872 + 19 192g	932 328 + 917g	2 699 434 + 3 339g
128	434 492 + 11 846g	465 786 + 830g	1 349 762 + 2 785g
256	215 784 + 9 402g	233 020 + 553g	674 844 + 1 869g

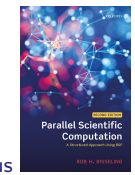
- The matrix cage14 is difficult to partition and it has a high communication cost.
- The matrix rgg_n_2_21_s0 with about the same n and nz has a far lower cost. Here, Mondriaan discovered the underlying geometric proximity in 2D.
- This also happened for the 2D Laplacian matrix.



BSP cost for the largest matrices

p	`random100k`	`laplace3D_256`
1	200 008 250	234 094 592
2	$100\,004\,126 + 50\,001g$	$117\,124\,986 + 65\,942g$
4	$51\,502\,050 + 77\,951g$	$59\,819\,416 + 72\,363g$
8	$25\,342\,198 + 70\,647g$	$30\,138\,652 + 78\,441g$
16	$12\,875\,064 + 60\,560g$	$15\,065\,066 + 59\,840g$
32	$6\,437\,730 + 42\,376g$	$7\,532\,394 + 44\,229g$
64	$3\,218\,834 + 39\,937g$	$3\,766\,718 + 30\,805g$
128	$1\,609\,420 + 30\,428g$	$1\,883\,288 + 18\,681g$
256	$804\,708 + 24\,039g$	$941\,610 + 12\,726g$

- The matrix `random100k` is difficult to partition and it has a high communication cost.
- Mondriaan also discovered the underlying geometric proximity for the 3D Laplacian matrix, but it still has a 6.8 times higher communication cost than the 2D Laplacian matrix.
- The computation cost is 1.4 times higher.



Execution time (in ms) using Multicore BSP for C

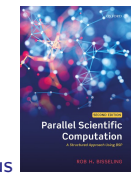
p		mip1	in-2004	asia_osm	cage14	rgg	lapl 2D	random 100k	lapl 3D
1	(seq)	13.9	32.1	150.2	46.9	59.0	213.6	379.5	255.5
1	(par)	14.9	35.4	181.2	50.9	63.6	241.2	390.3	285.7
2		7.5	18.6	91.4	31.6	32.9	122.7	188.8	150.2
4		4.1	10.0	46.6	21.4	16.5	63.7	115.9	104.4
8		2.3	5.7	24.9	13.8	11.0	52.2	71.6	72.0
16		2.3	4.8	30.1	10.4	11.2	53.1	38.9	72.6
32		1.6	5.4	29.7	12.4	5.3	43.6	20.8	41.0

- Results for `bspmv` executed on one Broadwell node of Cartesius, used as a shared-memory parallel computer.
- The overhead of the parallel program run for $p = 1$ is up to 20.6% (for `asia_osm`), compared to the sequential program.
- Only modest speedups were obtained, up to 18.2× on 32 processors for `random100k`.
- Higher speedups can be obtained by further optimization, e.g. by sending data in larger packets.

Execution time (in ms) using BSPonMPI

p		mip1	in-2004	asia_osm	cage14	rgg	lapl 2D	random 100k	lapl 3D
1	(seq)	13.8	31.6	150.0	46.5	56.5	213.6	380.0	256.3
1	(par)	14.7	34.6	178.6	50.0	61.5	239.0	386.7	283.8
2		7.9	19.0	95.8	40.8	32.8	125.9	190.4	176.3
4		4.5	9.7	44.9	33.2	16.4	66.2	126.8	93.2
8		3.0	6.0	26.0	22.3	12.8	52.5	84.8	80.9
16		3.9	7.7	35.0	23.0	12.7	61.8	58.3	85.8
32		2.4	8.3	34.2	22.5	5.7	52.7	23.9	38.8
64		2.2	2.8	7.9	8.4	3.3	13.2	15.9	21.0
128		2.9	2.1	5.3	7.5	2.3	8.2	13.5	13.1
256		3.9	3.0	3.0	6.1	2.3	5.1	15.4	8.7

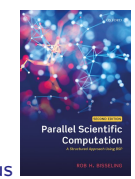
- Results for `bspmv` executed on up to 8 Broadwell nodes of Cartesius, used as a distributed-memory parallel computer.
- The highest speedup achieved is 50× on 256 processors for the matrix `asia_osm`, which has a low communication cost.
- The lowest maximum speedup achieved is 6.3× on 64 processors for the smallest matrix `mip1`.



Shared memory vs. distributed memory for $p = 32$

	mip1	in-2004	asia_osm	cage14	rgg	lapl 2D	random 100k	lapl 3D
shared	1.6	5.4	29.7	12.4	5.3	43.6	20.8	41.0
distributed	2.4	8.3	34.2	22.5	5.7	52.7	23.9	38.8

- Execution time (in ms) for `bspmv` executed on one Broadwell node of Cartesius.
- MulticoreBSP for C v.2.0.4 was used for shared memory and BSPonMPI v1.1 for distributed memory.
- In general, shared memory is faster. The largest gain is for cage14.
- Distributed memory can use more nodes.
- A hybrid shared/distributed-memory approach (possibly based on hybrid-BSP) can thus be beneficial.

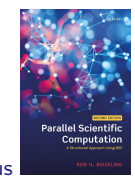


Did we gain something?

- We have shown that for large problem sizes the algorithm scales well.

Did we gain something?

- We have shown that for large problem sizes the algorithm scales well.
- If only all problems were large! More important than showing good speedups for large problems is understanding what happens for various problem sizes, small as well as large.
- The BSP cost can be used to explain timing results on a particular machine, or predict them, although agreement will never be perfect.



Summary

- The advantage of presenting BSP costs over presenting raw timings on a particular machine is that BSP cost results are future-proof.
- Raw timings, however, have a short lifespan.
- 20 years from now:
  - Cartesius, Cori, Summit, Fugaku, and their friends will all be dead.
  - I shall be older and perhaps wiser.
  - BSP costs like $215\,784 + 9\,402g + 4l$ for $p = 256$ and matrix cage14 will still be meaningful, perhaps as predictors for completely solar-powered zettaflop/s (10^{21} flop/s) machines.