RC25008 (W1006-029) June 9, 2010
Computer Science

IBM Research Report

An Evaluation of Parallel Graph Partitioning and Ordering Software on a Massively Parallel Computer

Anshul Gupta
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

An Evaluation of Parallel Graph Partitioning and Ordering Softwares on a Massively Parallel Computer

Anshul Gupta
Mathematical Sciences Department
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA.
anshul@watson.ibm.com

June 8, 2010

Abstract

We empirically study state-of-the-art parallel graph partitioning and sparse matrix ordering software packages. We compare their speed, quality, and robustness. For a model case, in which good partitionings (even optimal partitionings, in some cases) can be constructed manually, we compare the size of the edge cuts of the manual partitions with that of the partitions generated by the multilevel heuristics that are at the heart of modern graph partitioning software packages. We show that the quality of the partitions generated by the software is only slightly worse than that of the manual partition for this class of model graphs. We discuss the shortcomings of the current ordering software and argue that there is an urgent need for more robust, scalable, and high-quality software for sparse matrix ordering to support scalable solution of sparse linear systems by direct methods on massively parallel computers.

1 Introduction

This report contains the results of a concise experimental study of two state-of-the-art parallel graph partitioning software packages and three parallel sparse matrix ordering packages on the Blue Gene/P [8]. For our study, we use a set of a dozen graphs derived from symmetric sparse matrices of moderate sizes. These matrices, listed in Table 1, were obtained from the University of Florida collection [4]. We are aware of three distributed-memory parallel software packages for graph partitioning, namely ParMETIS [7], PT-SCOTCH [3], and JOSTLE [10]. Since neither the source code, nor binaries compiled for our target machine, BG/P, are available for JOSTLE, we restricted the study to ParMETIS and PT-SCOTCH. We are aware of another parallel sparse matrix ordering software known as PORD [9], but did not include it in the study. PORD does not contain graph partitioning software, and based on our earlier experience with MUMPS [1, 2] (a parallel sparse linear system solver package that uses PORD ordering as the default), the quality of the orderings produced by Parallel Watson Sparse Matrix Package (WSMP) [6] is generally better. Therefore, we have included ordering results from the latter for comparison with ParMETIS and PT-SCOTCH. We use Version 9.9 of WSMP, Version 3.2 of ParMETIS, and Version 5.2.9.

Matrix	N	NNZ	Application/Origin
Algor-big	1,073,724	84,317,460	Static stress analysis
Alsim-b	5,978,665	29,640,547	Power network analysis
Audikw_1	943,695	76,651,847	Automotive crankshaft modeling
G3_circuit	1,585,478	7,660,826	Circuit simulation
Lmco	665,017	107,514,163	Structural analysis
Mstamp-2c	902,289	70,925,391	Metal stamping
Nastran-b	1,508,088	111,614,436	Structural analysis
Nd24k	72,000	28,715,634	3D mesh problem
Sgi_1M	1,522,431	125,755,875	Structural analysis
Ten-b	1,371,166	108,009,680	3-D metal forming
Thermal2	1,2281,045	8,580,313	Steady state thermal problem
Torso	201,142	3,161,120	Human torso modeling

Table 1: SPD test matrices with their order (N) and number of non-zeros (NNZ).

2 Graph Partitioning Results

In this section, we compare ParMETIS and PT-SCOTCH for graph partitioning. Both these packages employ multi-level heuristics. In multi-level partitioning, a graph is first coarsened by successively finding matchings, and then eliminating the matched edge by coalescing the vertices that it connects. After sufficient number of coarsening steps, when the graphs has been reduced to a size below a predetermined threshold, it is partioned. Finally, the coarsening is undone one level at a time, while refining the partitioning at each step. When the graph is fully uncoarsened to its original state, the partitioning is complete after the refinement step. More detailed descriptions of the parallel multilevel partitioning heuristics can be found in the relevant literature [3, 7]. A key difference between the partitioning strategies of ParMETIS and PT-SCOTCH is that ParMETIS performs only one cycle of coarsening and refining irrespective of the number of partitions desired, whereas PT-SCOTCH performs $\lceil \log_2(P) \rceil$ cycles of coarsening and refining to partition a graph into P parts. This is because ParMETIS partitions the coarsest graph into P parts and refines the complete partitioning at each uncoarsening step. PT-SCOTCH computes only a bisection or a 2-way partition at a time. It then bisects the partitions recursively until the desired number of partitions is computed. Clearly, this entails $\lceil \log_2(P) \rceil$ steps.

2.1 Comparison of partitioning quality and speed

Tables 2–5 show the results of partitioning the graphs in our test suite on 16, 64, 256, and 1024 BG/P nodes. In each table, we report the partitioning time and quality of ParMETIS and PT-SCOTCH for partitioning the graphs into 2, P, and $2P$ parts, where P is the number of nodes. We measure and report two metrics for quality of the partitioning. These are the total edge cut or the number of edges that connect vertices belonging to different partitions, and the maximum imbalance or the percentage by which the size of the largest partition exceeds the average partition size. Both partitioner accept a use defined threshold of acceptable maximum imbalance. We used a threshold of 5% in all our experiments. Default values are used for all other parameters for both the packages. A hyphen in the tables indicates a failure of the configuration corresponding to that entry, either because of the partitioner crashing or running out of memory. For a ready comparison of the two packages, the

Number of Partitions → / Matrices ↓	ParMETIS 2	ParMETIS P	ParMETIS $2 \times P$	PT-SCOTCH 2	PT-SCOTCH P	PT-SCOTCH $2 \times P$	
Algor-big	425039	1984596	2720681	–	–	–	Edges cut
	4.47%	4.39%	4.78%	–	–	–	Max. imbalance
	2.94	3.03	3.29	–	–	–	Time (seconds)
Alsim-b	4081	26618	38260	2680	15916	24537	Edges cut
	0.14%	2.16%	3.92%	0.00%	2.54%	2.91%	Max. imbalance
	6.67	6.81	6.86	5.86	16.0	19.8	Time (seconds)
Audikw_1	106853	1357326	2027681	107217	1367289	2072169	Edges cut
	1.03%	4.66%	4.98%	0.85%	4.39%	5.17%	Max. imbalance
	9.56	10.4	10.9	9.24	30.0	40.7	Time (seconds)
G3_circuit	1994	14039	19295	1244	9914	15152	Edges cut
	0.00%	3.68%	4.88%	1.80%	3.44%	3.88%	Max. imbalance
	1.56	1.60	1.66	1.30	3.80	4.76	Time (seconds)
Lmco	154301	2481785	3962014	145761	2298571	3975585	Edges cut
	0.99%	5.00%	5.03%	1.46%	5.25%	6.30%	Max. imbalance
	3.00	3.87	4.26	4.20	18.1	34.0	Time (seconds)
Mstamp-2c	382815	1834896	2458087	379260	1761246	2446541	Edges cut
	0.00%	4.78%	4.82%	0.11%	4.86%	3.84%	Max. imbalance
	2.46	2.63	2.89	4.53	14.6	21.3	Time (seconds)
Nastran-b	103517	973744	1499380	86274	899577	1491046	Edges cut
	0.63%	4.99%	4.99%	1.27%	4.90%	5.95%	Max. imbalance
	16.1	16.9	17.4	14.5	43.5	56.8	Time (seconds)
Nd24k	795640	3061013	3899140	701355	3019536	4067957	Edges cut
	2.37%	5.02%	5.33%	2.00%	6.49%	7.56%	Max. imbalance
	3.02	4.21	4.64	4.88	13.4	14.3	Time (seconds)
Sgi_1M	134640	1591912	2558298	126252	1509588	2466981	Edges cut
	1.60%	4.99%	4.87%	1.98%	5.92%	6.96%	Max. imbalance
	22.0	23.2	24.0	19.1	54.9	70.7	Time (seconds)
Ten-b	348777	2175602	3182763	355428	2140359	3096725	Edges cut
	2.76%	4.57%	4.87%	1.09%	4.61%	5.36%	Max. imbalance
	4.19	4.56	4.82	5.41	20.7	34.1	Time (seconds)
Thermal2	1087	13995	22310	1054	13080	20408	Edges cut
	0.01%	3.54%	3.99%	1.19%	4.05%	4.30%	Max. imbalance
	1.42	1.51	1.55	0.98	2.81	3.34	Time (seconds)
Torso	10440	54662	80629	9772	52758	79912	Edges cut
	4.43%	4.85%	5.00%	1.99%	5.73%	6.80%	Max. imbalance
	0.38	0.41	0.44	0.51	1.62	2.43	Time (seconds)
Geometric mean	**46608**	**370839**	**544540**	**41016**	**331369**	**506780**	**Edges cut**
Average	**1.27%**	**4.39%**	**4.79%**	**1.25%**	**4.74%**	**5.37%**	**Max. imbalance**
Geometric mean	**3.70**	**4.09**	**4.32**	**4.06**	**12.8**	**17.4**	**Time (seconds)**

Table 2: Partitioning statistics for ParMETIS and PT-SCOTCH on 16 processes ($P = 16$).

bottom of each table contains geometric means of the edge cut, maximum imbalance, and partitioning time for those graphs for which none of the partitioners failed.

The comparison in Table 2 shows that the quality of PT-SCOTCH partitions is better than those of ParMETIS on 16 nodes. As expected, PT-SCOTCH takes longer to compute 16 or 32 partitions because of its recursive bisection strategy, unlike the single step multiway partitioning strategy used in ParMETIS. The results in Table 3 for 64 nodes are similar to those in Table 2. A notable difference, however, is that the maximum imbalance in PT-SCOTCH partitioning into 64 and 128 parts exceeds the 5% threshold. The cause of growing imbalance in PT-SCOTCH lies in its recursive bisection strategy. PT-SCOTCH treats each bisection as an independent partitioning problem. A given bisectioning problem does not take into account whether it is partitioning the smaller or the larger partition from the previous bisection. Ideally, the balancing requirements on the larger partition need to be more stringent in order to guarantee that the imbalance in the final partitioning does not exceed the user defined threshold. In the absence of an adaptive management of imbalance in each bisection computation, the imbalance gets compounded at each level of bisection. Tables 4 and 5 show that this phenomenon gets more pronounced as the number of nodes and partitions increases. Although the size of the edge cuts produced by PT-SCOTCH remains smaller than those produced by ParMETIS, the high imbalance renders PT-SCOTCH partitioning into more than 64 parts impractical for real applications.

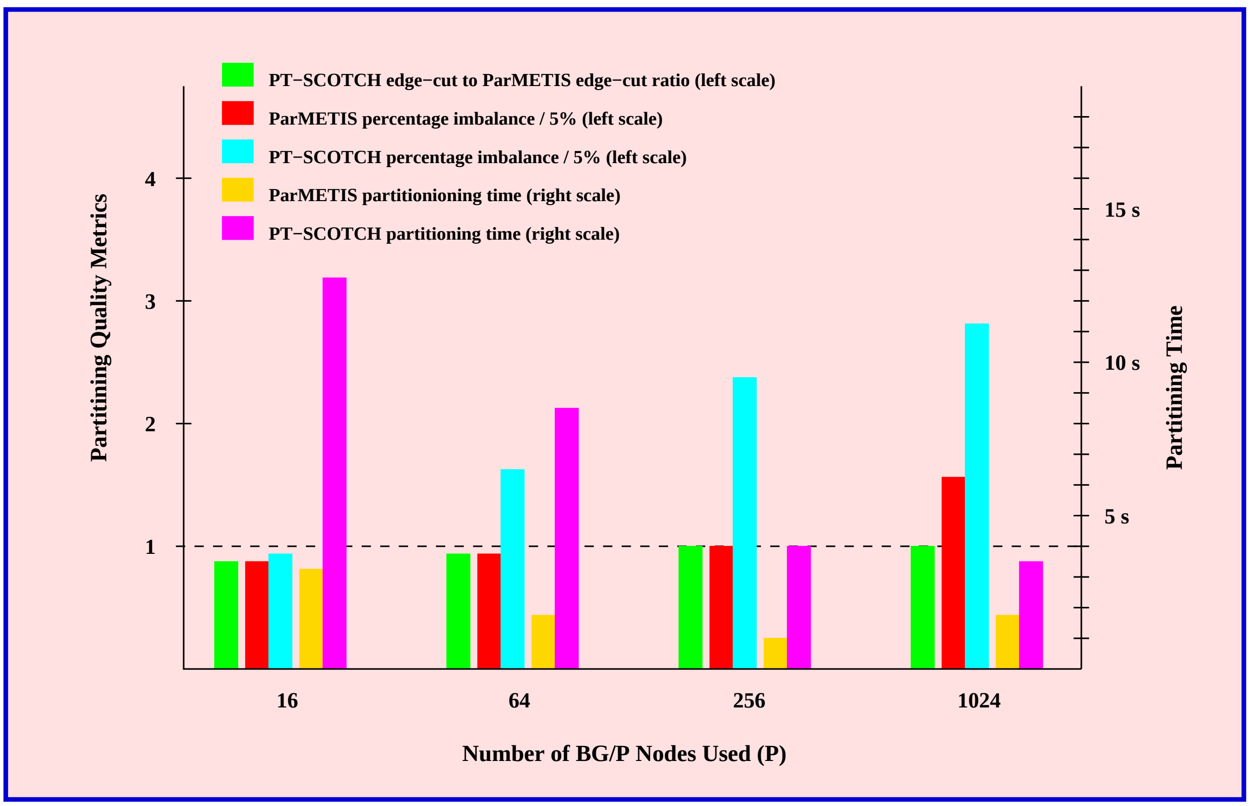

Figure 1: ParMETIS and PT-SCOTCH comparison when the number of partitions is the same as the number of MPI processes.

Figure 1 contains a graphical depiction of the geometric mean data from Tables 2–5. Note that for 1024 partitions on 1024 MPI processes, the geometric mean of the imbalance for ParMETIS too seems to go beyond

Number of Partitions → / Matrices ↓	ParMETIS 2	ParMETIS P	ParMETIS $2 \times P$	PT-SCOTCH 2	PT-SCOTCH P	PT-SCOTCH $2 \times P$	
Algor-big	446459	3665634	4802692	427831	3660252	4996580	Edges cut
	1.84%	4.94%	4.87%	0.42%	10.1%	11.1%	Max. imbalance
	1.18	1.38	1.48	3.39	13.3	16.4	Time (seconds)
Alsim-b	4036	48122	64533	2414	32241	47685	Edges cut
	0.03%	3.51%	4.53%	0.90%	4.70%	5.68%	Max. imbalance
	2.05	2.22	2.29	2.61	10.4	11.3	Time (seconds)
Audikw_1	109180	2965230	4162268	108981	2986228	4241224	Edges cut
	0.50%	4.99%	5.00%	1.03%	8.85%	9.46%	Max. imbalance
	2.58	3.13	3.38	3.46	16.1	18.9	Time (seconds)
G3_circuit	1874	33245	53040	1333	22183	36881	Edges cut
	2.27%	4.54%	4.28%	0.05%	5.47%	5.51%	Max. imbalance
	0.61	0.67	0.70	0.50	2.16	2.58	Time (seconds)
Lmco	135529	5751471	8079937	129529	5780260	8195493	Edges cut
	0.43%	5.00%	5.27%	0.96%	8.59%	9.69%	Max. imbalance
	1.28	1.82	2.08	2.59	15.9	18.8	Time (seconds)
Mstamp-2c	384021	3244471	4313942	376713	3195067	4453477	Edges cut
	0.85%	4.98%	4.99%	0.31%	7.40%	7.97%	Max. imbalance
	1.08	1.20	1.29	3.00	11.4	14.0	Time (seconds)
Nastran-b	88479	2225284	3263959	90180	2253226	3255261	Edges cut
	0.76%	4.99%	5.01%	1.47%	7.99%	9.07%	Max. imbalance
	6.37	7.11	7.50	5.52	23.3	28.0	Time (seconds)
Nd24k	695354	5303531	6993334	689704	5404439	6979632	Edges cut
	1.60%	5.42%	7.56%	2.00%	11.6%	12.7%	Max. imbalance
	6.37	9.87	12.1	2.79	7.93	8.17	Time (seconds)
Sgi_1M	131417	3984616	5717904	133236	3887558	5747958	Edges cut
	1.46%	5.00%	5.00%	0.33%	8.74%	9.83%	Max. imbalance
	7.05	7.99	8.42	6.10	25.2	30.8	Time (seconds)
Ten-b	365490	4377504	5672840	354393	4337095	5725184	Edges cut
	0.54%	4.65%	4.99%	0.63%	7.15%	7.75%	Max. imbalance
	1.64	1.90	2.01	3.32	17.1	20.8	Time (seconds)
Thermal2	1290	32837	49497	1077	32545	47185	Edges cut
	0.00%	4.53%	4.46%	0.46%	6.93%	7.59%	Max. imbalance
	0.50	0.56	0.60	0.34	1.62	1.81	Time (seconds)
Torso	11320	114436	156181	10126	113099	157246	Edges cut
	2.83%	4.61%	4.94%	1.14%	8.37%	9.51%	Max. imbalance
	0.20	0.24	0.25	0.31	1.25	1.44	Time (seconds)
Geometric mean	**55357**	**891198**	**1243358**	**49832**	**831423**	**1183434**	**Edges cut**
Average	**1.09%**	**4.76%**	**5.08%**	**0.81%**	**7.99%**	**8.82%**	**Max. imbalance**
Geometric mean	**1.58**	**1.88**	**2.03**	**1.99**	**8.49**	**9.92**	**Time (seconds)**

Table 3: Partitioning statistics for ParMETIS and PT-SCOTCH on 64 processes ($P = 64$).

Number of Partitions → / Matrices ↓	ParMETIS 2	ParMETIS P	ParMETIS $2 \times P$	PT-SCOTCH 2	PT-SCOTCH P	PT-SCOTCH $2 \times P$	
Algor-big	447753	6252550	7927361	425736	6435014	8085763	Edges cut
	0.95%	5.00%	5.00%	0.82%	13.2%	14.0%	Max. imbalance
	0.41	0.56	0.68	1.98	7.62	8.30	Time (seconds)
Alsim-b	3145	84722	118210	–	–	–	Edges cut
	0.00%	4.55%	4.86%	–	–	–	Max. imbalance
	1.17	1.38	1.48	–	–	–	Time (seconds)
Audikw_1	112311	5710116	7524868	104535	5790781	7707901	Edges cut
	0.26%	5.31%	5.31%	0.84%	12.6%	12.6%	Max. imbalance
	1.08	1.40	1.57	1.58	8.98	9.56	Time (seconds)
G3_circuit	1878	78502	105678	1332	55433	80242	Edges cut
	0.00%	3.74%	3.43%	1.92%	9.05%	9.57%	Max. imbalance
	0.60	0.69	0.70	0.30	1.49	1.62	Time (seconds)
Lmco	144141	10920589	14247071	–	–	–	Edges cut
	0.11%	5.25%	5.26%	–	–	–	Max. imbalance
	0.50	0.85	1.07	–	–	–	Time (seconds)
Mstamp-2c	412146	5508116	7063156	384759	5700756	7317206	Edges cut
	0.29%	5.12%	5.20%	1.02%	10.6%	10.7%	Max. imbalance
	0.34	0.49	0.60	1.86	8.54	7.55	Time (seconds)
Nastran-b	96904	4773917	6611421	92358	4669752	6598890	Edges cut
	0.14%	5.16%	5.28%	0.29%	10.1%	11.1%	Max. imbalance
	2.32	2.28	3.06	2.11	11.4	12.4	Time (seconds)
Nd24k	730119	8684320	11333968	–	–	–	Edges cut
	1.29%	10.9%	27.3%	–	–	–	Max. imbalance
	6.33	10.4	11.1	–	–	–	Time (seconds)
Sgi_1M	135873	7651743	10200306	128457	7729759	10351402	Edges cut
	1.21%	5.03%	5.70%	2.00%	15.7%	16.9%	Max. imbalance
	2.80	3.38	3.72	2.47	12.5	13.4	Time (seconds)
Ten-b	366462	7319344	9400789	346923	7489539	9753346	Edges cut
	1.07%	5.02%	5.19%	1.37%	15.0%	15.0%	Max. imbalance
	0.53	0.72	0.89	2.11	10.3	11.3	Time (seconds)
Thermal2	1129	70447	100101	1126	71706	99916	Edges cut
	0.00%	4.69%	4.43%	2.00%	10.3%	10.9%	Max. imbalance
	0.59	0.71	0.71	0.21	1.10	1.17	Time (seconds)
Torso	13509	213203	281944	9941	214744	285206	Edges cut
	1.57%	4.87%	5.13%	0.21%	10.1%	11.2%	Max. imbalance
	0.29	0.35	0.43	0.25	1.15	1.19	Time (seconds)
Geometric mean	**52173**	**1581794**	**2100704**	**46672**	**1541061**	**2068988**	**Edges cut**
Average	**0.61%**	**4.88%**	**4.96%**	**1.16%**	**11.8%**	**12.4%**	**Max. imbalance**
Geometric mean	**0.72**	**0.89**	**1.04**	**1.00**	**4.90**	**5.15**	**Time (seconds)**

Table 4: Partitioning statistics for ParMETIS and PT-SCOTCH on 256 processes ($P = 256$).

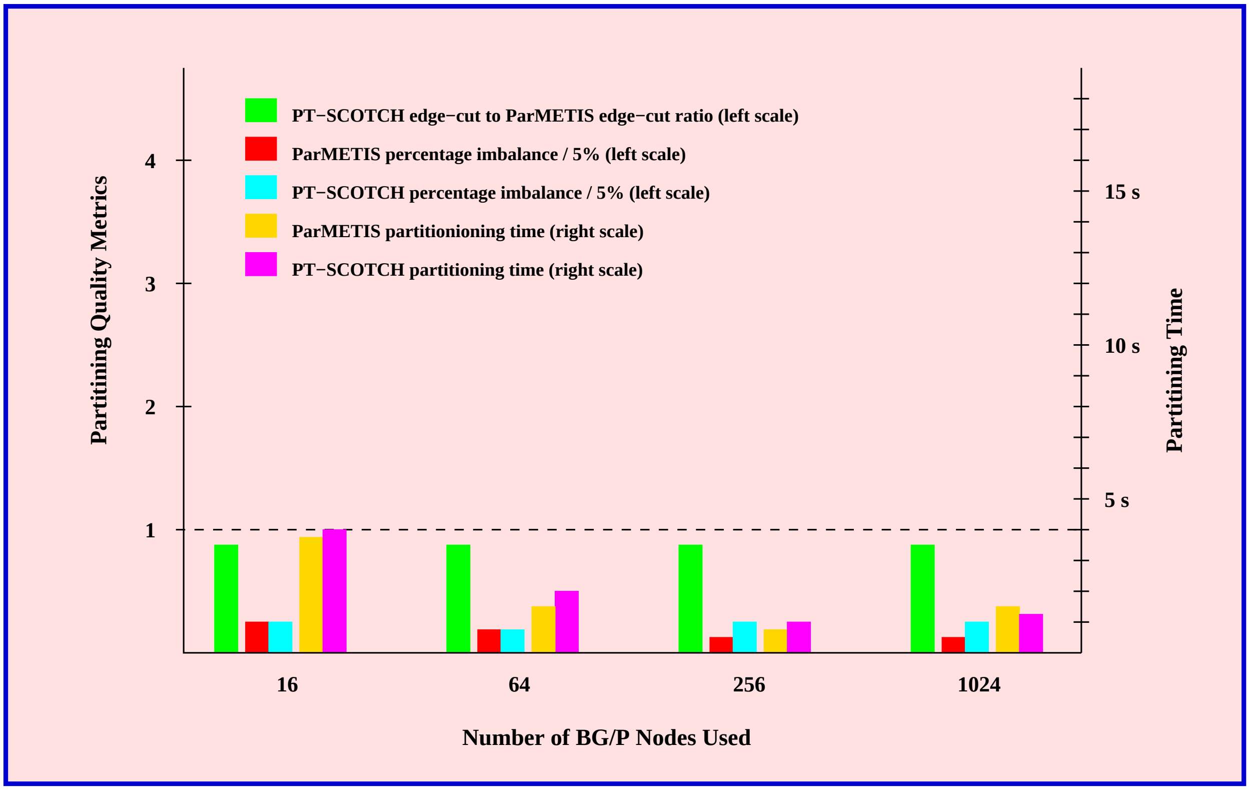

Figure 2: ParMETIS and PT-SCOTCH comparison for graph bisection.

5%. However, a closer look at Table 5 would reveal that the statistics are skewed by a single bad case: graph *nd24k*, which has only 72000 vertices and would have very few vertices per partition in a 1024-way partitioning. Figure 2 shows similar statistics for graph bisection (2-way partition) on number of MPI processes increasing from 16 to 1024. This figure, along with Table 2, shows that partition quality obtained by PT-SCOTCH is slightly better than that obtained from ParMETIS for small number of partitions (irrespective of the number of MPI processes).

2.2 Heuristic versus manual partitions for a model problem

In this section, we consider partitioning regular unweighted 3-D graphs with a 7-point cubic stencil, where each vertex is connected to its immediate neighbors in both directions along x, y, and z axes. Tables 6 and 7 compare the edge-cuts of partitions generated by ParMETIS with balanced manual partitions created simply by introducing cutting planes along each dimension to maintain load balance and to maintain the best possible aspect ratio of of the dimensions of the partitions.

It can be observed that the ratio of the sizes of cuts produced by ParMETIS to those generated manually is close to 1.5 for a wide range of graph sizes, partition sizes, and number of MPI processes. The ratio deteriorates slightly as the size of the graph increases, and improves very slightly as the number of partitions increases. In fact, a close observation reveals that the ratio is almost constant for a given average number of vertices per partition. Table 7 shows that when the number of MPI processes is kept constant at 32 and the number of

Number of Partitions → / Matrices ↓	ParMETIS 2	ParMETIS P	ParMETIS $2 \times P$	PT-SCOTCH 2	PT-SCOTCH P	PT-SCOTCH $2 \times P$	
Algor-big	475753	10074683	12667459	419560	10490912	13142764	Edges cut
	0.80%	5.29%	5.86%	1.56%	14.4%	15.8%	Max. imbalance
	0.32	0.57	0.96	1.80	6.16	6.20	Time (seconds)
Alsim-b	3773	171165	239776	2458	143195	206839	Edges cut
	0.55%	4.80%	4.68%	0.49%	9.34%	9.24%	Max. imbalance
	2.30	2.59	2.87	1.81	5.73	5.85	Time (seconds)
Audikw_1	109746	9745613	12534227	113211	9954688	12761406	Edges cut
	0.11%	5.58%	7.21%	1.46%	14.8%	16.1%	Max. imbalance
	0.81	1.15	2.36	1.26	6.32	6.36	Time (seconds)
G3_circuit	2156	140353	188705	1483	114184	161962	Edges cut
	0.00%	4.24%	5.92%	2.00%	9.47%	9.93%	Max. imbalance
	1.26	1.04	1.49	0.58	1.85	1.89	Time (seconds)
Lmco	136665	18519743	23575523	130270	18840372	24082589	Edges cut
	0.48%	7.02%	7.48%	1.00%	16.4%	17.6%	Max. imbalance
	1.77	1.50	2.84	1.15	7.11	7.17	Time (seconds)
Mstamp-2c	413838	8987618	11280209	371628	9277671	11628226	Edges cut
	0.72%	5.54%	7.36%	0.89%	14.1%	16.9%	Max. imbalance
	0.45	0.78	1.07	1.73	5.47	5.47	Time (seconds)
Nastran-b	98559	9004369	12076573	88893	8962402	12093880	Edges cut
	0.04%	5.65%	6.47%	0.78%	16.0%	17.2%	Max. imbalance
	7.69	8.06	8.11	1.44	7.31	7.42	Time (seconds)
Nd24k	723707	12348960	13154160	694100	11994364	13097836	Edges cut
	1.34%	39.4%	39.4%	2.06%	18.0%	22.3%	Max. imbalance
	6.55	10.9	11.0	2.70	6.40	6.42	Time (seconds)
Sgi_1M	145908	13429147	17272737	132912	13698648	17561488	Edges cut
	1.24%	5.87%	6.41%	0.59%	13.6%	14.9%	Max. imbalance
	9.44	8.80	10.1	1.71	7.86	7.95	Time (seconds)
Ten-b	380961	11941106	15030032	359883	12312997	15521890	Edges cut
	0.03%	5.37%	5.75%	1.48%	14.5%	15.2%	Max. imbalance
	0.46	0.84	1.38	1.70	7.47	7.61	Time (seconds)
Thermal2	1212	141401	197978	1102	148107	203343	Edges cut
	0.01%	4.31%	4.06%	0.93%	11.8%	11.9%	Max. imbalance
	2.05	2.32	2.42	0.51	1.44	1.47	Time (seconds)
Torso	12347	365929	469777	10310	369946	475324	Edges cut
	0.71%	5.38%	5.89%	0.39%	14.0%	15.1%	Max. imbalance
	0.52	0.72	1.01	0.42	1.47	1.47	Time (seconds)
Geometric mean	**57919**	**2905685**	**3733212**	**50590**	**2857623**	**3699231**	**Edges cut**
Average	**0.50%**	**8.20%**	**8.87%**	**1.14%**	**13.9%**	**15.2%**	**Max. imbalance**
Geometric mean	**1.51**	**1.90**	**2.59**	**1.23**	**4.61**	**4.67**	**Time (seconds)**

Table 5: Partitioning statistics for ParMETIS and PT-SCOTCH on 1024 processes ($P = 1024$).

partitions is increased, then the ratio improves somewhat faster than what we observe in Table 6. This shows that the partition quality declines very slightly with increasing number of MPI processes.

Cube (n), Graph (n^3) Dimensions	32-way	64-way	128-way	256-way	512-way	1024-way	
$n = 128$	168722	232199	305206	396226	508356	635988	Metis
	114688	*147456	212992	278528	*344064	475136	Manual
$n^3 = 2097152$	**1.47**	**1.57**	**1.43**	**1.42**	**1.48**	**1.34**	Ratio
$n = 256$	692838	941707	1264127	1665213	2137405	2722783	Metis
	458752	*589824	851968	1114112	*1376256	1900544	Manual
$n^3 = 16777216$	**1.51**	**1.60**	**1.48**	**1.49**	**1.55**	**1.43**	Ratio
$n = 448$	2119976	2951874	3982187	5207300	6758690	8601565	Metis
	1404928	*1806336	2609152	3411968	*4214784	5820416	Manual
$n^3 = 89915392$	**1.51**	**1.63**	**1.53**	**1.53**	**1.60**	**1.48**	Ratio
$n = 576$	3543317	4946173	6743418	8787701	11267256	14477239	Metis
	2322432	*2985984	4313088	5640192	*6967296	9621504	Manual
$n^3 = 191102976$	**1.53**	**1.66**	**1.56**	**1.56**	**1.62**	**1.50**	Ratio

Table 6: A comparison of the best (smallest possible) and ParMETIS edge-cuts for unweighted 3-D graphs with a 7-point cubic stencil. The number of BG/P nodes used is the same as the number of partitions. A '*' indicates an optimal partition.

Cube (n), Graph (n^3) Dimensions	32-way	64-way	128-way	256-way	512-way	1024-way	
$n = 256$	692838	918806	1216633	1589311	2036719	2577758	Metis
	458752	*589824	851968	1114112	*1376256	1900544	Manual
$n^3 = 16777216$	**1.51**	**1.56**	**1.43**	**1.43**	**1.48**	**1.36**	Ratio

Table 7: A comparison of the best (smallest possible) and ParMETIS edge-cuts for unweighted 3-D graphs with a 7-point cubic stencil. 32 BG/P nodes are used for each partition. A '*' indicates an optimal partition.

2.3 Conclusions from partitioning experiments

The results in Sections 2.1 indicate that PT-SCOTCH outperforms ParMETIS in terms of partition quality for a small number of partitions. However, for a large number of partitions, ParMETIS is the only practical option.

The results in Section 2.2 indicate that, at least for a 3-D model problem, the quality of the partitions generated by the parallel multilevel heuristics is worse than the optimal partitioning by only a small factor, and that this factor stays more or less constant over a wide range of graph sizes, MPI processes, and number of partitions.

Matrices	ParMETIS		PT-SCOTCH		PWSSMP uncompressed		PWSSMP compressed	
	Time	Opcount	Time	Opcount	Time	Opcount	Time	Opcount
Algor-big	32.7	4.21e+13	-	-	228.	2.34e+13	83.4	2.31e+13
Alsim-b	36.8	1.31e+11	33.0	2.43e+11	68.9	1.52e+11	72.0	1.57e+11
Audikw_1	44.8	6.02e+12	43.2	5.99e+12	68.6	5.05e+12	19.6	5.20e+12
G3_circuit	10.1	6.46e+10	10.2	6.84e+10	17.7	6.94e+10	18.9	6.95e+10
Lmco	41.7	4.09e+12	34.3	4.05e+12	88.1	3.11e+12	24.4	3.27e+12
Mstamp-2c	30.4	2.80e+13	25.2	1.98e+13	154.	1.62e+13	27.8	1.62e+13
Nastran-b	59.7	3.20e+12	66.3	3.16e+12	87.6	2.70e+12	29.5	3.76e+12
Nd24k	71.2	2.04e+12	14.1	2.11e+12	131.	2.06e+12	143.	1.97e+12
Sgi_1M	77.1	8.98e+12	80.7	8.71e+12	107.	7.40e+12	34.1	7.84e+12
Ten-b	42.7	4.98e+13	36.3	3.59e+13	179.	2.91e+13	41.0	2.90e+13
Thermal2	8.64	1.64e+10	5.61	1.77e+10	12.8	1.75e+10	14.4	1.76e+10
Torso	6.34	1.40e+11	2.45	1.44e+11	5.50	1.27e+11	6.44	1.27e+11
Geom. mean	**29.8**	**1.40e+12**	**21.6**	**1.41e+12**	**56.2**	**1.12e+12**	**28.4**	**1.25e+12**

Table 8: Ordering statistics for ParMETIS, PT-SCOTCH, and PWSSMP on 16 processes ($P = 16$).

3 Parallel Sparse Matrix Ordering

In this section, we compare the time and quality of parallel fill-reducing ordering produced by ParMETIS, PT-SCOTCH, and parallel WSMP on the set of matrices described in Table 1. ParMETIS and PT-SCOTCH use a distributed multilevel strategy to compute vertex separators of the graph of the matrix to generate a nested-dissection [5] type permutation of the vertices. On the contrary, parallel WSMP generates a vertex separator sequentially. It first computes a node bisector of the entire graph on a single MPI process. The two subgraphs are subsequently processed independently in parallel and the process continues recursively. Note that there are two potential disadvantages of this approach. First, only limited speedup from parallelism can be expected because a significant amount of computation that goes into finding the first separator is performed sequentially, and the parallelism increases gradually as the size of the subgraphs decreases. The second problem is that this approach requires the entire graph to be stored on a single process to compute the first separator, and hence is not scalable with respect to memory, although this may not be a problem for highly compressible graphs (see next paragraph).

Some of the matrices in our test suite have graphs with multiple degrees of freedom per node. Parallel WSMP ordering can automatically detect this and take this into account to reduce memory and time requirements of ordering. On the other hand, the user must explicitly supply compressed graph information to ParMETIS and PT-SCOTCH. In our experiments, we have used ParMETIS and PT-SCOTCH with the original uncompressed graphs, and Parallel WSMP with the graph compression option turned both off and on. Default values for all other parameters were used with the exception of matching type for ParMETIS, which was set to PARMETIS_MTYPE_GLOBAL. Setting this to the default value of PARMETIS_MTYPE_LOCAL resulted in some improvement in runtime and some deterioration in the quality of ordering. Tables 8–11 show the parallel ordering results for the four cases on 16 to 1024 BG/P nodes. The last row of each table shows that geometric means of the column values for all those matrices that all the packages were able to reorder. Figure 3

Matrices	ParMETIS		PT-SCOTCH		PWSSMP uncompressed		PWSSMP compressed	
	Time	Opcount	Time	Opcount	Time	Opcount	Time	Opcount
Algor-big	44.4	3.47e+13	17.5	4.08e+13	140.	2.29e+13	76.4	2.28e+13
Alsim-b	20.3	1.24e+11	-	-	58.0	1.46e+11	61.3	1.46e+11
Audikw_1	66.6	5.55e+12	17.1	5.86e+12	92.3	5.07e+12	18.8	5.17e+12
G3_circuit	16.0	6.21e+10	7.18	6.71e+10	14.4	7.10e+10	15.7	7.09e+10
Lmco	80.1	3.96e+12	17.2	4.08e+12	77.4	3.23e+12	23.3	3.19e+12
Mstamp-2c	51.6	2.21e+13	14.6	1.93e+13	141.	1.61e+13	27.2	1.62e+13
Nastran-b	58.9	2.89e+12	27.2	3.27e+12	84.3	2.78e+12	27.2	3.68e+12
Nd24k	99.0	2.03e+12	8.52	2.10e+12	130.	2.03e+12	129.	2.10e+12
Sgi_1M	71.8	7.91e+12	30.2	8.37e+12	97.8	7.33e+12	31.4	7.62e+12
Ten-b	47.6	4.40e+13	20.5	4.99e+13	209.	2.91e+13	39.6	2.88e+13
Thermal2	10.5	1.71e+10	2.44	1.82e+10	11.5	1.72e+10	13.0	1.74e+10
Torso	16.1	1.19e+11	1.48	1.49e+11	4.93	1.33e+11	5.87	1.33e+11
Geom. mean	**41.7**	**2.14e+12**	**11.0**	**2.31e+12**	**58.2**	**1.90e+12**	**26.6**	**1.97e+12**

Table 9: Ordering statistics for ParMETIS, PT-SCOTCH, and PWSSMP on 64 processes ($P = 64$).

Matrices	ParMETIS		PT-SCOTCH		PWSSMP uncompressed		PWSSMP compressed	
	Time	Opcount	Time	Opcount	Time	Opcount	Time	Opcount
Algor-big	189.	2.67e+13	8.93	2.57e+13	212.	2.34e+13	72.2	2.33e+13
Alsim-b	45.6	1.07e+11	-	-	58.4	1.35e+11	61.4	1.36e+11
Audikw_1	207.	5.06e+12	8.29	5.75e+12	90.0	4.94e+12	18.7	5.17e+12
G3_circuit	39.6	6.16e+10	5.03	7.17e+10	14.1	6.85e+10	15.4	6.83e+10
Lmco	272.	3.06e+12	8.24	3.79e+12	73.2	3.08e+12	23.1	3.25e+12
Mstamp-2c	175.	1.61e+13	7.84	2.74e+13	139.	1.62e+13	26.4	1.62e+13
Nastran-b	173.	2.61e+12	11.1	3.48e+12	83.8	2.78e+12	27.4	3.61e+12
Nd24k	103.	2.04e+12	6.91	2.10e+12	140.	2.10e+12	150.	2.10e+12
Sgi_1M	215.	7.10e+12	13.3	8.81e+12	132.	7.54e+12	31.9	7.59e+12
Ten-b	169.	3.96e+13	10.8	3.60e+13	202.	2.88e+13	40.1	2.87e+13
Thermal2	28.8	1.66e+10	1.38	1.64e+10	11.1	1.73e+10	12.6	1.68e+10
Torso	18.3	1.20e+11	1.30	1.38e+11	4.88	1.29e+11	5.82	1.29e+11
Geom. mean	**109.**	**1.91e+12**	**6.15**	**2.20e+12**	**61.4**	**1.89e+12**	**26.7**	**1.95e+12**

Table 10: Ordering statistics for ParMETIS, PT-SCOTCH, and PWSSMP on 256 processes ($P = 256$).

Matrices	ParMETIS		PT-SCOTCH		PWSSMP uncompressed		PWSSMP compressed	
	Time	Opcount	Time	Opcount	Time	Opcount	Time	Opcount
Algor-big	255.	2.26e+13	7.23	3.38e+13	169.	2.29e+13	79.5	2.31e+13
Alsim-b	142.	1.03e+11	10.2	1.81e+11	59.6	1.34e+11	62.6	1.34e+11
Audikw_1	248.	4.93e+12	5.33	5.54e+12	64.6	4.97e+12	19.8	5.11e+12
G3_circuit	62.5	5.15e+10	4.65	8.41e+10	14.7	6.57e+10	16.1	6.54e+10
Lmco	-	-	5.91	3.67e+12	74.8	3.00e+12	23.2	3.19e+12
Mstamp-2c	212.	1.60e+13	6.69	1.74e+13	141.	1.61e+13	27.5	1.62e+13
Nastran-b	327.	2.54e+12	6.67	3.27e+12	75.9	2.64e+12	27.9	3.48e+12
Nd24k	-	-	-	-	139.	2.03e+12	144.	2.10e+12
Sgi_1M	389.	7.13e+12	7.76	9.25e+12	95.5	7.16e+12	32.5	7.54e+12
Ten-b	298.	2.91e+13	7.83	3.27e+13	148.	2.86e+13	41.1	2.86e+13
Thermal2	60.3	1.32e+10	1.65	1.78e+10	11.4	1.71e+10	12.9	1.68e+10
Torso	18.7	1.20e+11	1.58	1.33e+11	5.16	1.28e+11	6.13	1.28e+11
Geom. mean	**148.**	**1.23e+12**	**5.14**	**1.61e+12**	**49.3**	**1.34e+12**	**25.8**	**1.39e+12**

Table 11: Ordering statistics for ParMETIS, PT-SCOTCH, and PWSSMP on 1024 processes ($P = 1024$).

summarizes these results graphically.

The results in Tables 8–11 and Figure 3 show that all ordering packages have limitations. PWSMP is robust (no failures in any of the tests), and generates good quality orderings in reasonable time; however, as stated earlier, its parallel ordering strategy is not scalable in terms of either ordering time or memory. Having said that, it seems to have a fairly memory-frugal implementation relative to PT-SCOTCH (which runs out of memory for one test case on 16 nodes) and ParMETIS (which runs out of memory for two test cases on 1024 nodes). In terms of ordering quality, PWSMP without graph compression generates orderings with the least operation count on 16 and 64 nodes. ParMETIS catches up as the number of nodes increase, matching PWSMP's quality on 256 nodes and exceeding it on 1024. However, this improvement in quality of ParMETIS comes at the cost of runtime that increases with the number of nodes. PT-SCOTCH ordering is the fastest and is the only one that gets meaningful speedups. However, The factorization operation count of ordering generated by PT-SCOTCH is consistently worse than that of PWSMP by roughly 20%.

Acknowledgements

This work was supported by King Abdullah University of Science and Technology (KAUST).

References

[1] Patrick R. Amestoy, Iain S. Duff, Jacko Koster, and J. Y. L'Excellent. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[2] Patrick R. Amestoy, Iain S. Duff, and J. Y. L'Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computational Methods in Applied Mechanical Engineering*, 184:501–520, 2000.

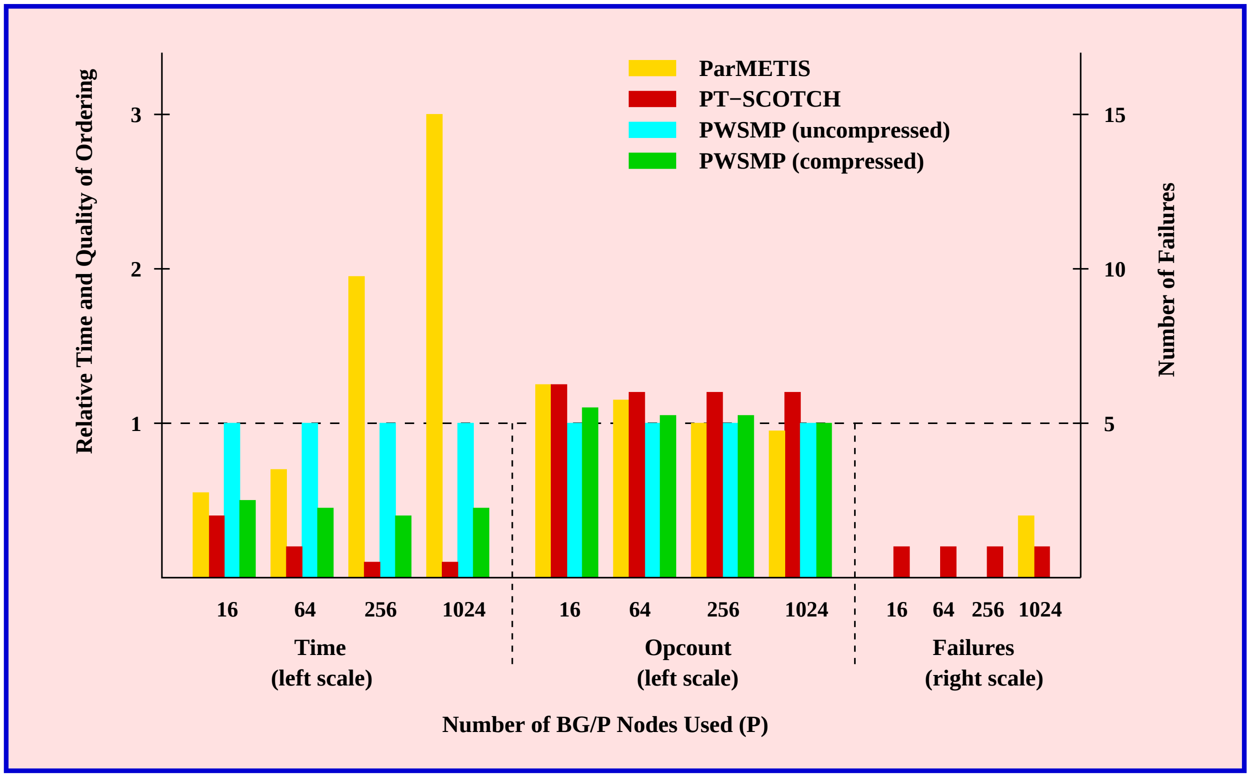

Figure 3: A comparison of the speed, quality, and robustness of ParMETIS, PT-SCOTCH, and PWSMP orderings. The time and operation count of factorization is relative to that of uncompressed PWSMP ordering.

[3] C. Chevalier and F. Pellegrini. PT-Scotch: A tool for efficient parallel graph ordering. *Parallel Computing*, 34(6-8):318–331, 2008.

[4] Timothy A. Davis. The university of Florida sparse matrix collection. Technical report, Department of Computer Science, University of Florida, Jan 2007.

[5] Alan George. Nested dissection of a regular finite-element mesh. *SIAM Journal on Numerical Analysis*, 10:345–363, 1973.

[6] Anshul Gupta. WSMP: Watson sparse matrix package (Part-I: Direct solution of symmetric sparse systems). Technical Report RC 21886, IBM T. J. Watson Research Center, Yorktown Heights, NY, November 2000. *http://www.cs.umn.edu/˜agupta/wsmp*.

[7] George Karypis and Vipin Kumar. ParMETIS: Parallel graph partitioning and sparse matrix ordering library. Technical Report TR 97-060, Department of Computer Science, University of Minnesota, 1997.

[8] Gary Lakner and Carlos P. Sosa. *IBM System Blue Gene Solution: Blue Gene/P Application Development.* IBM, 2008. *http://www.redbooks.ibm.com/abstracts/sg247287.html*.

[9] Jürgen Schulze. Towards a tighter coupling of bottom-up and top-down sparse matrix ordering methods. *Bit Numerical Mathematics*, 41(4):800–841, 2001.

[10] Chris Walshaw and M. Cross. JOSTLE: Parallel multilevel graph-partitioning software—an overview. In F. Magoules, editor, *Mesh Partitioning Techniques and Domain Decomposition Techniques*, pages 27–58. Civil-Comp Ltd., 2007.