Augmenting Hypergraph Models with Message Nets to Reduce Bandwidth and Latency Costs Simultaneously

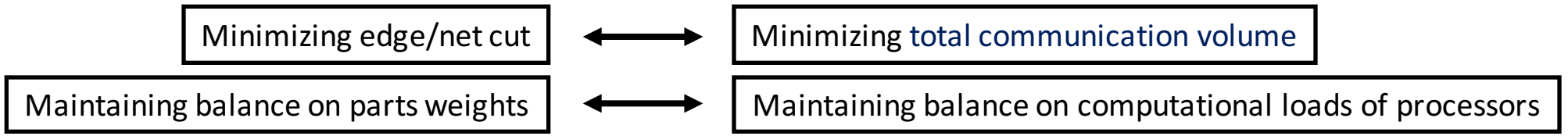
Oguz Selvitopi, **Seher Acer**, and Cevdet Aykanat
Bilkent University, Ankara, Turkey

CSC16, Albuquerque, NM, USA
October 10-12, 2016

To appear in *IEEE TPDS* with DOI: 10.1109/TPDS.2016.2577024 as
O. Selvitopi, S. Acer, C. Aykanat, "A Recursive Hypergraph Bipartitioning Framework for Reducing Bandwidth and Latency Costs Simultaneously"

Introduction

- Our goal
  - Efficient parallelization of irregular applications for distributed-memory systems
  - Optimization of communication costs
- Communication costs = bandwidth cost + latency cost
  - Bandwidth cost ≈ volume of data communicated
  - Latency cost ≈ number of messages
- Models for reducing bandwidth cost are abundant
  - Graph and hypergraph models
    - Vertices = computational tasks
    - Edges or nets = computational dependencies between tasks
    - Cut edges and nets incur communication (in terms of volume)



Related Work

- Works that minimize latency cost
  - A two-phase method: communication hypergraph [1, 2, 3] for sparse matrices
    - *1st phase*: a partition Π on computational tasks obtained, usually by a model addressing bandwidth cost
    - *2nd phase*: communication hypergraph model applied on Π to distribute communication tasks for minimizing latency cost
    - An objective optimized in one phase can be degraded in the other phase
  - A one-phase method: UMPa [4]
    - Can address bandwidth metrics (max/avg volume) and latency metrics (max/avg message count), together or separately
    - Contains specific refinement procedures for each of these metrics
    - Introduces an additional partitioning time of $O(VK^2)$ to each refinement pass
- Works that provide an upper bound on latency cost
  - 2D Cartesian models [5, 6, 7] with maximum message count of $2(\sqrt{K}-1)$

[1] Uçar and Aykanat, *SIAM SISC 2004*

[2] Uçar and Aykanat, *LNCS 2003*

[3] Selvitopi and Aykanat, *PARCO 2016*

[4] Deveci et al., *JPDC 2015*

[5] Hendrickson et al., *IJHSC 1995*

[6] Çatalyürek and Aykanat, *SC 2001*

[7] Boman et al., *SC 2013*

Proposed Model: Message Nets

Basics

- We augment standard hypergraph model with *message nets*
  - The nets in the standard models: *volume nets*
- Our model relies on *recursive hypergraph bipartitioning*

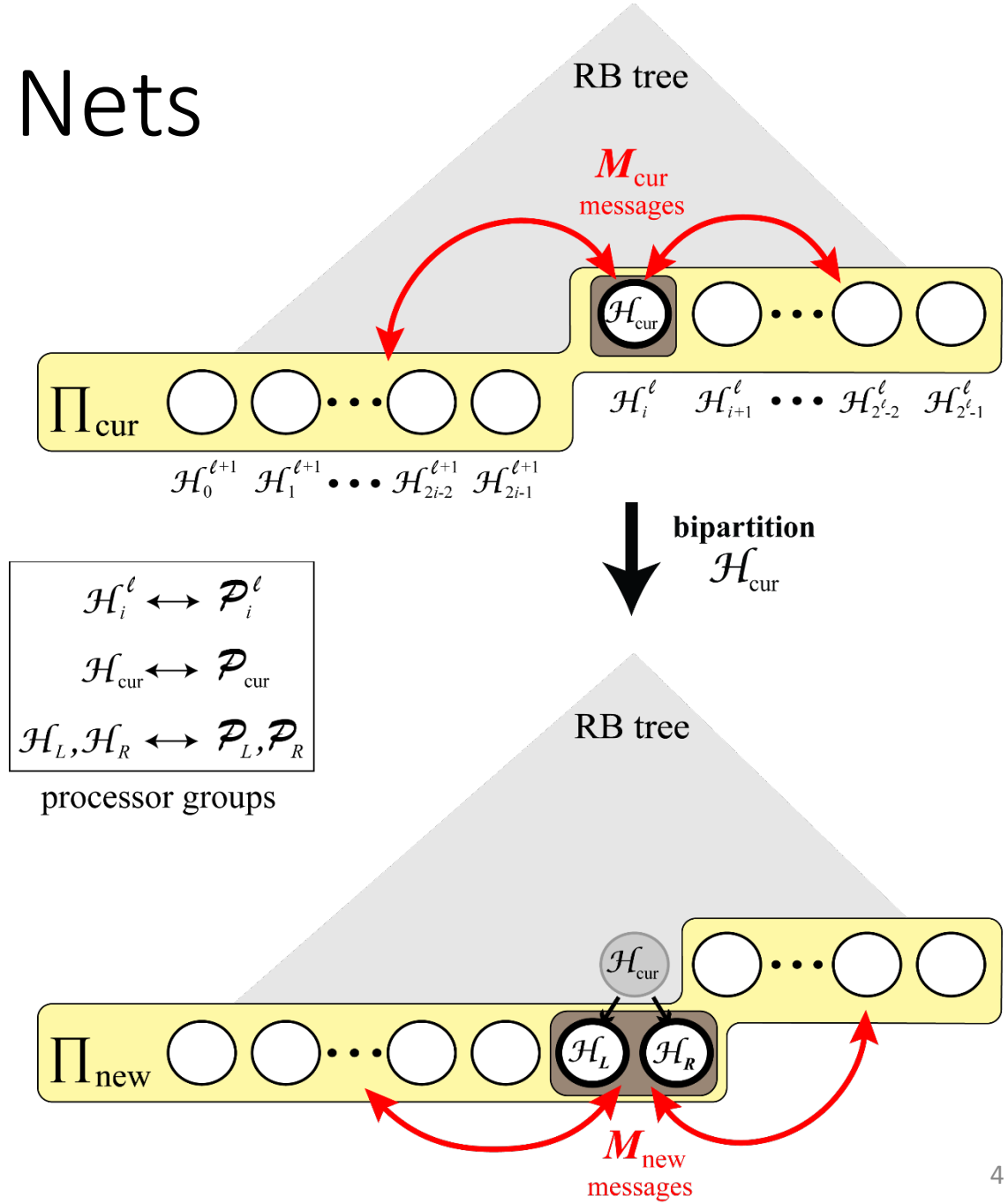
Nets

- Volume nets: maintained via net-splitting
- Message nets: added to the current hypergraph to be bipartitioned

Having **both** net types in bipartitions ⟷ **Simultaneous** reduction of bandwidth and latency costs

Message nets

- A message net connects vertices representing items/tasks that necessitate a message together
  - Such items/tasks are encouraged to be together either in P_L or P_R
- A send net s_k added for each P_k which P_{cur} sends a message to
  - Connects vertices representing input items sent to P_k
- A receive net r_k added for each P_k which P_{cur} receives a message from
  - Connects vertices representing tasks that need input items received from P_k



Proposed Model: Partitioning

Number of cut message nets = $M_{new} - M_{cur}$

Increase in number of messages that P_{cur} communicates with others

Correctness

Message nets and volume nets with respective costs of t_s and t_w

- Minimizing cutsize ≈ minimizing the increase in communication cost
- Provides a more **accurate** communication cost **representation**

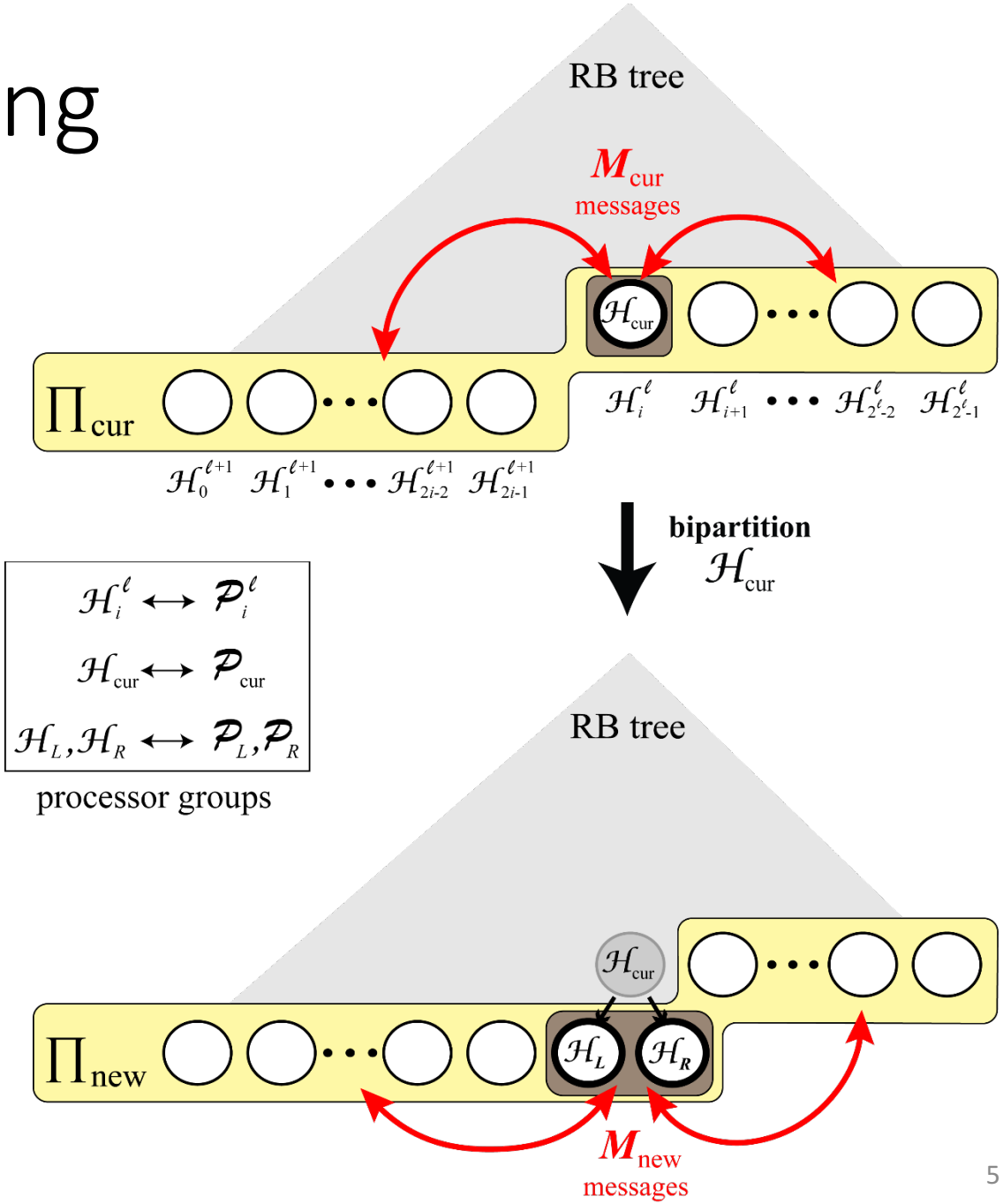
*It is **flexible***

Can be realized by using any hypergraph partitioning tool

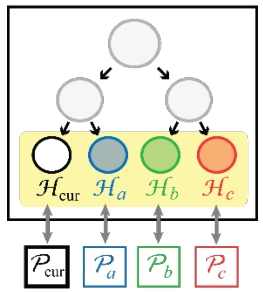
*It is **cheap***

$Cost$(our model) = $Cost$(standard model) + $O(p \log_2 K)$

- Our model traverses each pin once for each RB tree level

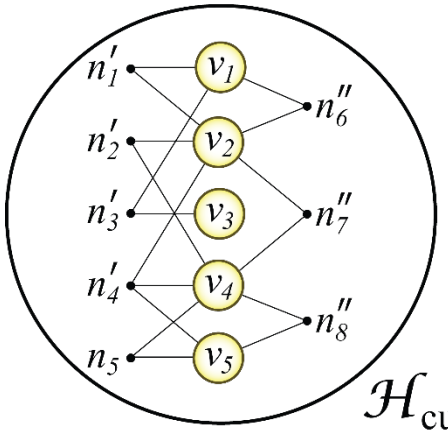


Recursive Bipartitioning Tree Status

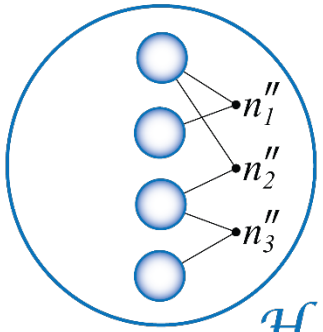


associated processor groups

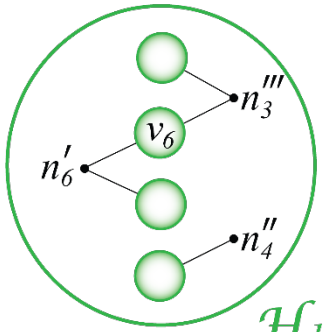
Hypergraphs in the Leaf Nodes of the Recursive Bipartitioning Tree



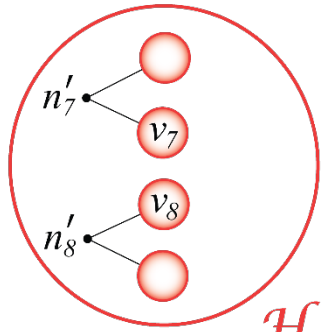
$\mathcal{H}_{\text{cur}}$



$\mathcal{H}_a$



$\mathcal{H}_b$

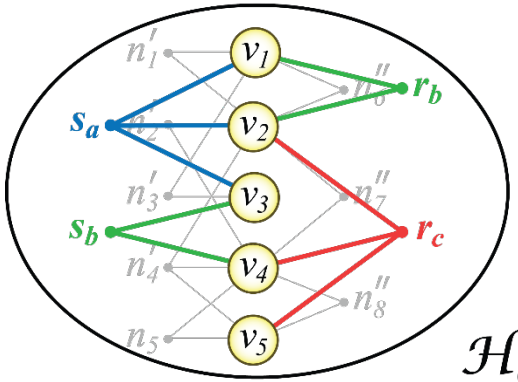


$\mathcal{H}_c$

addition of message nets

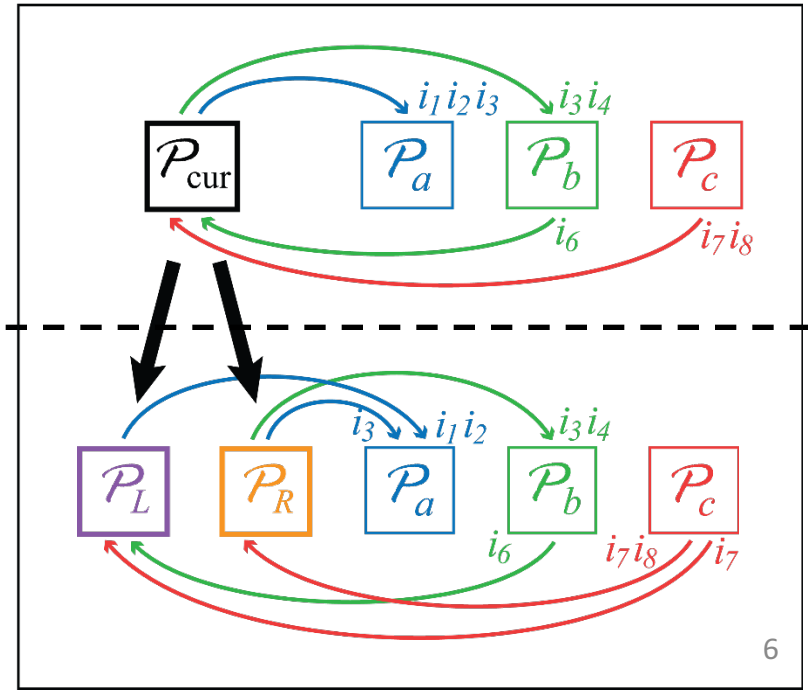
s_a, s_b : send nets

r_b, r_c : receive nets

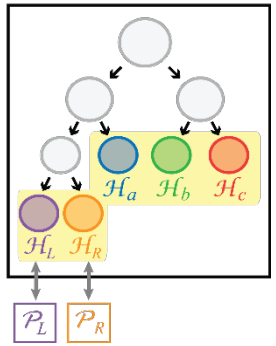


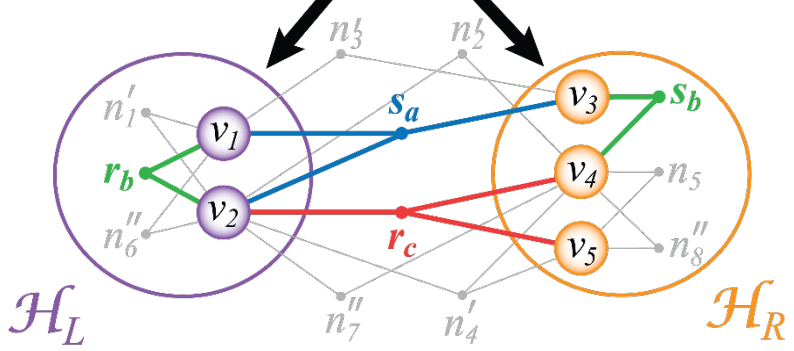
$\mathcal{H}_{\text{cur}}^M$

messages communicated by $\mathcal{P}_{\text{cur}}$



bipartitioning $\mathcal{H}_{\text{cur}}^M$





$\mathcal{H}_L$ $\mathcal{H}_R$

Experiments - 1

- An A_{PRE} application: 1D row-parallel SpMV
- Compared against: Standard column-net HP model
- Bipartitioning tool: PaToH with default setting
- Number of processors (K): $\{128, 256, 512, 1024, 2048\}$
- Message net cost (t_s/t_w): $\{10, 50, 100, 200\}$
- Dataset: 30 matrices from UFL
- Compared partitioning metrics
  - Total/maximum number of messages
  - Total/maximum volume of processors
  - Partitioning time
- Parallel SpMV
  - PETSc toolkit
  - Blue Gene/Q system

Dataset

name	problem kind	#rows/cols	#nonzeros
d_pretok	2D/3D	183K	1.6M
turon_m	2D/3D	190K	1.7M
cop20k_A	2D/3D	121K	2.7M
torso3	2D/3D	259K	4.4M
mono_500Hz	acoustics	169K	5.0M
memchip	circuit simulation	2.7M	14.8M
Freescale1	circuit simulation	3.4M	18.9M
circuit5M_dc	circuit simulation	3.5M	19.2M
rajat31	circuit simulation	4.7M	20.3M
laminar_duct3D	comp. fluid dynamics	67K	3.8M
StocF-1465	comp. fluid dynamics	1.5M	21.0M
web-Google	directed graph	916K	5.1M
in-2004	directed graph	1.4M	16.9M
eu-2005	directed graph	863K	19.2M
cage14	directed graph	1.5M	27.1M
mac_econ_fwd500	economic	207K	1.3M
gsm_106857	electromagnetics	589K	21.8M
pre2	freq. simulation	659K	6.0M
kkt_power	optimization	2.1M	14.6M
bcsstk31	structural	36K	1.2M
engine	structural	144K	4.7M
shipsec8	structural	115K	6.7M
Transport	structural	1.6M	23.5M
CO	theor./quan chemistry	221K	7.7M
598a	undirected graph	111K	1.5M
m14b	undirected graph	215K	3.4M
roadNet-CA	undirected graph	2.0M	5.5M
great-britain_osm	undirected graph	7.7M	16.3M
germany_osm	undirected graph	11.5M	24.7M
debr	undirected graph sequence	1.0M	4.2M

Experiments - 1

For message net cots of 50:

- Total number of messages: $\mathbf{35\% - 44\%}$ improvement
- Maximum number of messages: $20\% - 31\%$ improvement
- Total volume: $17\% - 48\%$ degradation
- Maximum volume: $25\% - 85\%$ degradation
- Partitioning time: $8\% - 33\%$ degradation
- Parallel SpMV time: $\mathbf{8\% - 29\%}$ improvement

↑ message net cost ⟷ ↑ improvements in latency metrics

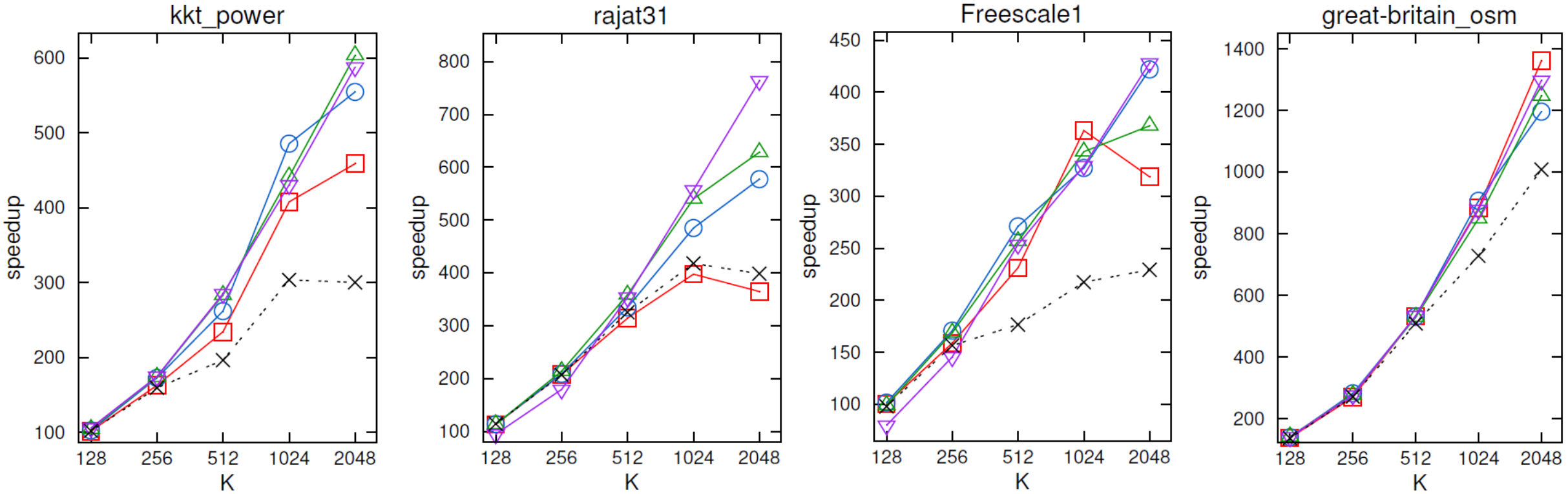
↑ number of processors ⟷ ↑ improvements in parallel SpMV time

↑ improvement rate in latency metrics ⟷ ↑ degradation rates in bandwidth metrics

Average results normalized with respect to standard model

message net cost	K	number of messages		volume		partitioning time	parallel SpMV time
		tot	max	tot	max		
10	128	0.82	0.87	1.08	1.11	1.07	**0.956**
	256	0.78	0.83	1.10	1.16	1.13	**0.904**
	512	0.75	0.83	1.12	1.22	1.13	**0.838**
	1024	0.73	0.84	1.16	1.29	1.25	**0.792**
	2048	0.71	0.88	1.20	1.37	1.28	**0.774**
50	128	0.65	0.76	1.17	1.25	1.08	**0.924**
	256	0.59	0.70	1.25	1.44	1.14	**0.846**
	512	0.56	0.69	1.33	1.57	1.21	**0.760**
	1024	0.57	0.74	1.41	1.69	1.24	**0.715**
	2048	0.59	0.80	1.48	1.85	1.33	**0.708**
100	128	0.59	0.73	1.24	1.43	1.09	**0.954**
	256	0.53	0.68	1.35	1.66	1.17	**0.858**
	512	0.51	0.68	1.45	1.86	1.19	**0.768**
	1024	0.53	0.71	1.54	1.92	1.31	**0.706**
	2048	0.57	0.80	1.61	2.06	1.41	**0.707**
200	128	0.54	0.72	1.33	1.60	1.15	**1.031**
	256	0.48	0.67	1.46	1.87	1.19	**0.872**
	512	0.49	0.67	1.57	2.02	1.25	**0.778**
	1024	0.52	0.72	1.65	2.09	1.37	**0.722**
	2048	0.57	0.79	1.70	2.17	1.48	**0.712**

Experiments - 1



Experiments - 2

- Same application (1D row-parallel SpMV)
- Compared against: **UMPa**
- Number of processors: 512
- Message net cost: 50
- Dataset: 38 matrices from 10th DIMACS Implementation Challenge
- Compared partitioning metrics
  - Total number of messages
  - Partitioning time
- Total number of messages:
  - **21**% improvement
- Partitioning time:
  - **37**% improvement

Proposed model vs UMPa

Name	#rows /cols	#nonzeros	Class	total number of messages			partitioning time		
				standard model	UMPa	proposed model	standard model	UMPa	proposed model
citationCiteseer	268K	2313K	Citation	53,824	0.64	0.65	24.17	4.11	1.57
coAuthorsCiteseer	227K	1628K	Citation	41,486	0.63	0.60	7.23	3.91	1.78
coAuthorsDBLP	299K	1955K	Citation	78,531	0.61	0.65	10.87	4.98	2.30
coPapersCiteseer	434K	32073K	Citation	53,467	0.69	0.68	246.29	0.36	1.10
coPapersDBLP	540K	30491K	Citation	106,073	0.71	0.74	157.64	1.03	2.12
caidaRouterLevel	192K	1218K	Clustering	20,491	0.74	1.34	8.39	3.78	1.92
cnr-2000	326K	3216K	Clustering	6,411	0.75	0.68	28.14	3.90	1.26
eu-2005	863K	19235K	Clustering	22,600	0.68	0.42	215.40	3.09	1.14
G_n_pin_pout	100K	1002K	Clustering	230,910	0.62	0.79	12.15	6.32	1.74
in-2004	1383K	16917K	Clustering	10,898	0.59	0.55	160.83	1.46	1.16
preferentialAttachment	100K	1000K	Clustering	190,519	0.67	1.05	16.91	6.34	2.04
smallworld	100K	1000K	Clustering	137,196	0.55	0.56	5.29	6.73	2.82
1280000	1279K	2570K	CompTask	3,033	0.57	0.35	14.72	2.65	1.15
320000	320K	645K	CompTask	2,481	0.72	0.43	9.30	2.01	1.23
delaunay_n17	131K	786K	Delaunay	3,028	1.00	0.61	3.50	3.37	1.38
delaunay_n18	262K	1573K	Delaunay	3,032	1.00	0.61	6.28	3.42	1.32
delaunay_n19	524K	3146K	Delaunay	3,035	1.00	0.66	11.24	2.11	1.22
delaunay_n20	1049K	6291K	Delaunay	3,026	1.00	0.74	20.60	1.99	1.23
rgg_n_2_17_s0	131K	1458K	RandGeom	2,864	0.91	0.59	4.26	2.70	1.28
rgg_n_2_18_s0	262K	3095K	RandGeom	2,885	0.92	0.59	8.37	2.26	1.35
rgg_n_2_19_s0	524K	6540K	RandGeom	2,898	0.96	0.59	16.95	1.85	1.36
rgg_n_2_20_s0	1049K	13783K	RandGeom	2,896	0.96	0.65	35.78	1.04	1.36
af_shell10	1508K	52672K	Sparse	2,886	1.00	0.99	103.09	0.40	1.13
af_shell9	505K	17589K	Sparse	2,804	1.00	0.96	35.56	0.60	1.17
audikw_1	944K	77652K	Sparse	5,284	0.95	0.83	357.19	0.35	1.14
ecology1	1000K	4996K	Sparse	2,895	0.99	0.68	13.50	2.76	1.22
ecology2	1000K	4996K	Sparse	2,895	0.99	0.65	13.46	2.82	1.28
G3_circuit	1585K	7661K	Sparse	2,975	0.98	0.64	26.09	2.28	1.35
ldoor	952K	46522K	Sparse	3,008	1.00	0.93	121.70	0.30	1.10
thermal2	1228K	8580K	Sparse	2,831	1.00	0.87	25.88	1.34	1.15
belgium_osm	1441K	3100K	Street	3,000	0.82	0.35	10.01	1.59	1.16
luxembourg_osm	115K	239K	Street	2,144	0.81	0.48	0.85	3.82	1.35
144	145K	2149K	Walshaw	5,665	0.93	0.65	11.75	3.20	1.15
598a	111K	1484K	Walshaw	5,514	0.93	0.61	8.52	3.67	1.23
auto	449K	6629K	Walshaw	5,598	0.94	0.73	34.37	2.44	1.24
fe_ocean	143K	819K	Walshaw	5,284	0.90	0.51	5.51	4.18	1.41
m14b	215K	3358K	Walshaw	5,347	0.94	0.62	16.82	3.39	1.19
wave	156K	2119K	Walshaw	6,758	0.92	0.67	10.94	3.13	1.20
			Geomean		**0.83**	**0.65**		**2.18**	**1.36**

Conclusion

- We proposed a one-phase hypergraph partitioning model for 1D partitioning
  - Our model simultaneously reduces total volume and total number of messages
- We augmented standard model with message nets
  - Our model is implementable by *any* hypergraph partitioning tool
- In comparison to standard model, we obtained average improvements of
  - up to **%52** in total number of messages
  - up to **%30** in parallel SpMV runtime
- In comparison to UMPa, we obtained average improvements of
  - up to **%21** in total number of messages
  - up to **%37** in partitioning time