Parallel Space-Time Kernel Density Estimation

Erik Saule[†], Dinesh Panchananam[†],
Alexander Hohl[‡], Wenwu Tang[‡], Eric Delmelle[‡]

[†] Dept. of Computer Science
[‡]Dept. of Geography and Earth Sciences
UNC Charlotte
Email: {esaule,dpanchan,ahohl,wtang4,eric.delmelle}@uncc.edu

Scheduling in Knoxville
May 26th, 2017

Outline

Space Time Kernel Density

What is it?

- Common way of visualizing events with time and place information
- Basically voxelize the space
- Give a value to each voxel that depends on the number of *neighboring* event to the voxel (with some kind of decay).
- Essentially a generalization of density maps (e.g., population density)

What is it useful for?

- Monitoring disease outbreak
- Political analysis
- Social media analysis
- Ornithology

Space-Time Kernel Density Estimate Formally

For a voxel x, y, t

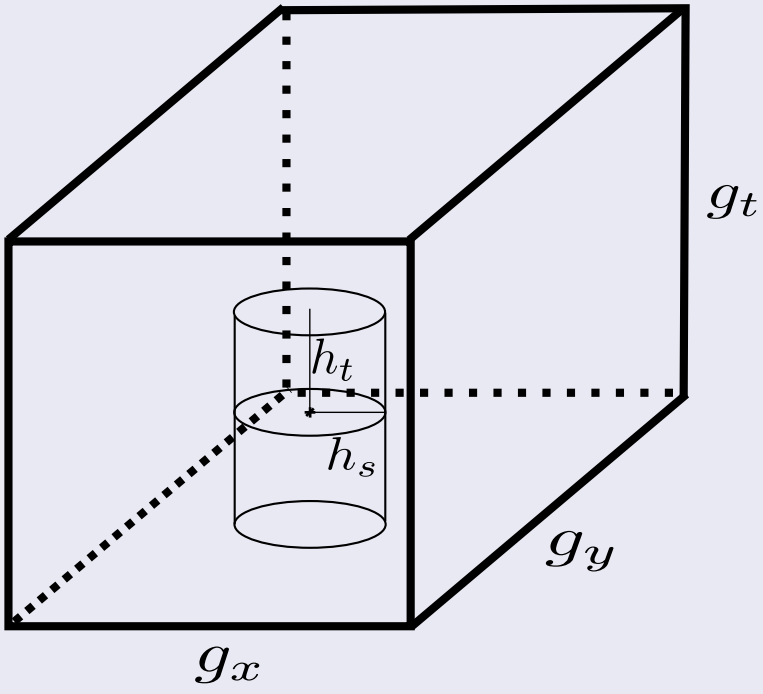
$$\hat{f}(x,y,t) = \frac{1}{nh_s^2 h_t} \sum_{i|d_i<h_s, t_i<h_t} k_s\left(\frac{x-x_i}{h_s}, \frac{y-y_i}{h_s}\right) k_t\left(\frac{t-t_i}{h_t}\right)$$

$$k_s(u,v) = \frac{\pi}{2}(1-u)^2(1-v)^2$$

$$k_t(w) = \frac{3}{4}(1-w)^2$$

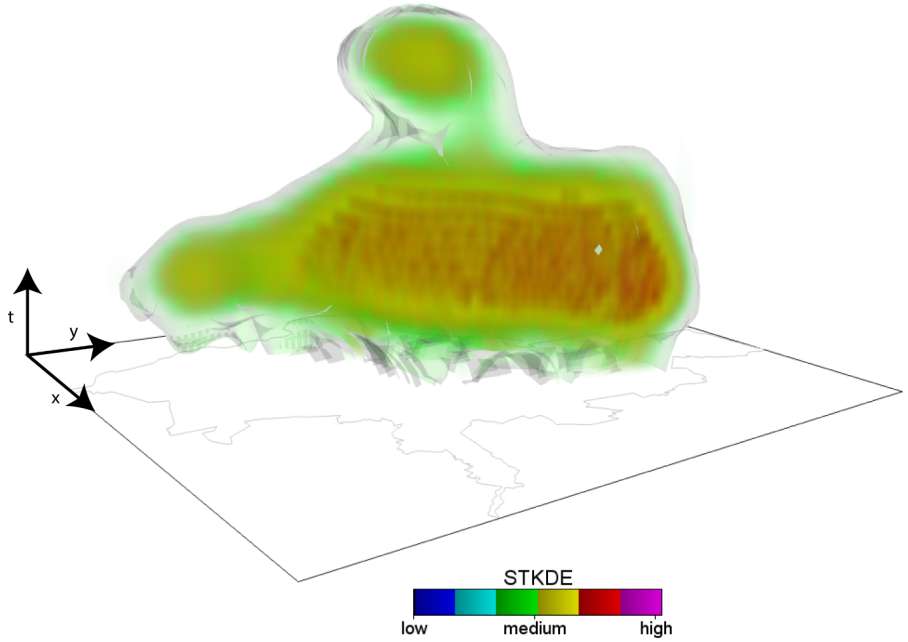
h_S is the spatial bandwidth
h_t is the temporal bandwidth
n is the number of points (events)

Each event radiates density

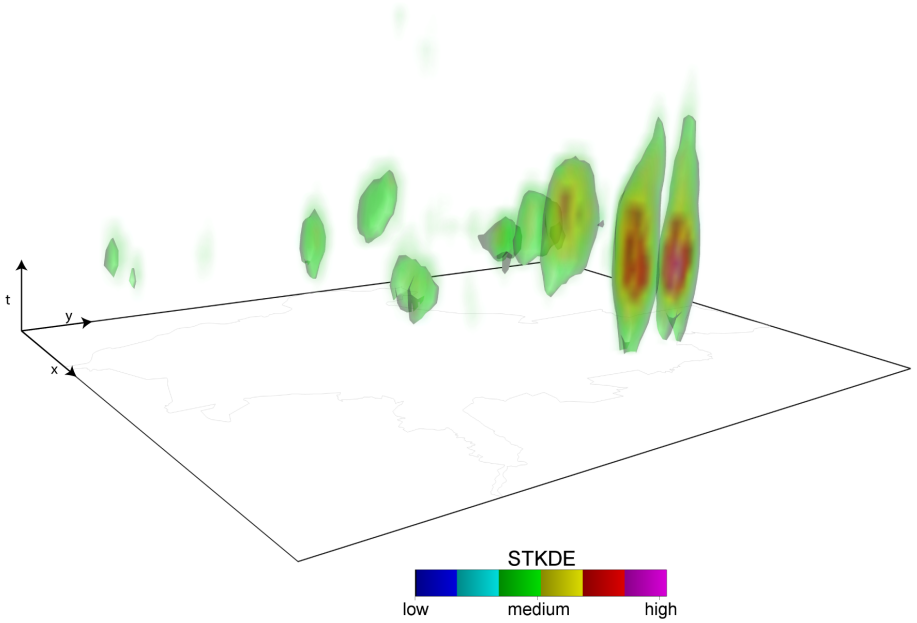


Similar to computing sums of radial basis functions from physics.

Dengue Fever in Cali, Colombia



$h_s = 2500m,\ h_t = 14days$



$h_s = 500m,\ h_t = 7days$

Outline

Voxel Based Algorithm VB

Algorithm

for all voxels $s = (x, y, t)$ **do**
  $sum = 0$
  for all points i at x_i, y_i, t_i **do**
    if $\sqrt{(x_i - x)^2 + (y_i - y)^2} < h_s$ and $|t_i - t| \leq h_t$ **then**
      $sum+ = k_s(\frac{x-x_i}{h_s}, \frac{y-y_i}{h_s})k_t(\frac{t-t_i}{h_t})$
  $stkde[X][Y][T] = \frac{sum}{nh_s^2 h_t}$

- $\theta(G_x G_y G_t n)$ distance tests
- $\theta(nH_s^2 H_t)$ density values
- Complexity: $\theta(G_x G_y G_t n)$

But pleasingly parallel.

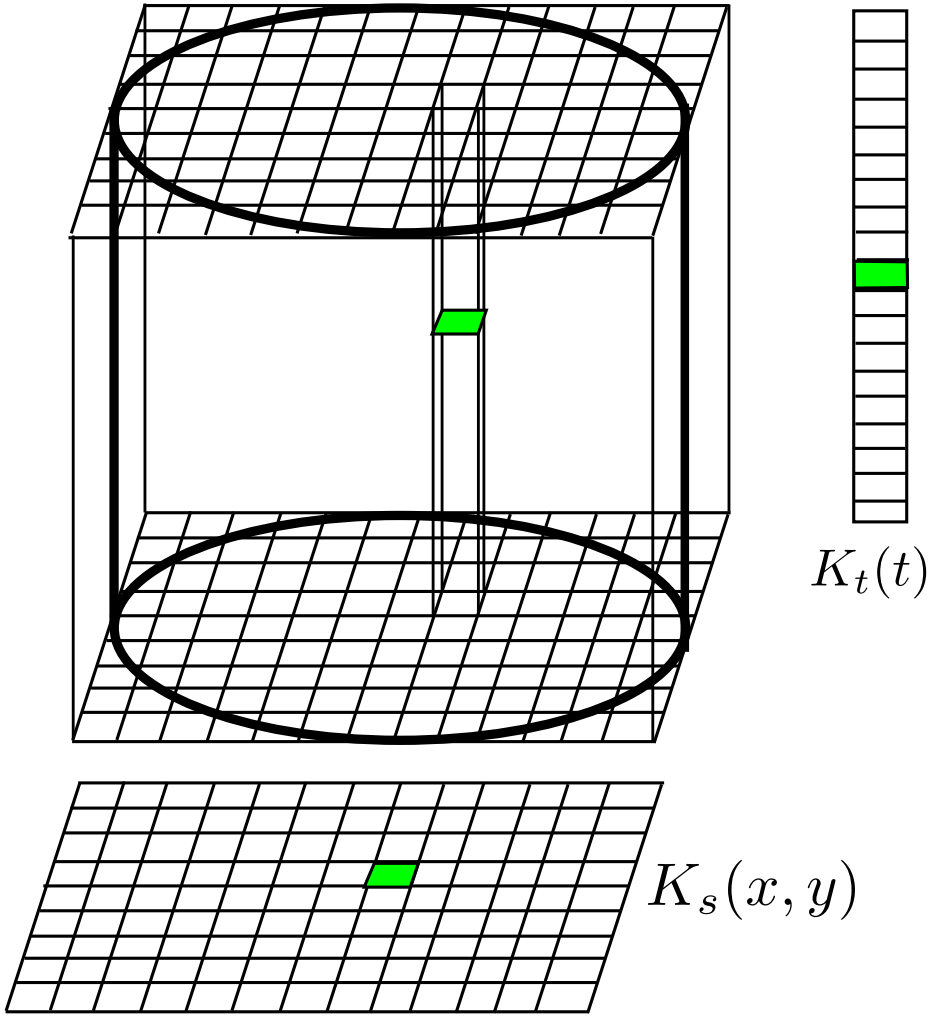
Point Based Algorithm PB

Algorithm

for all voxels $s = (x, y, t)$ **do**
  $stkde[X][Y][T] = 0$
for each points i at x_i, y_i, t_i **do**
  for $X_i - H_s \leq X \leq X_i + H_s$ **do**
    for $Y_i - H_s \leq Y \leq Y_i + H_s$ **do**
      for $T_i - T_s \leq T \leq T_i + H_s$ **do**
        if $\sqrt{(x_i - x)^2 + (y_i - y)^2} < h_s$ and $|t_i - t| \leq h_t$ **then**
          $stkde[X][Y][T] += \frac{k_s(\frac{x-x_i}{h_s}, \frac{y-y_i}{h_s}) k_t(\frac{t-t_i}{h_t})}{nh_s^2 h_t}$

- $\Theta(G_x G_y G_t)$ for memory initialization
- $\Theta(nH_s^2 H_t)$ density computations
- Complexity: $\Theta(G_x G_y G_t + nH_s^2 H_t)$
- (Gain the $\theta(G_x G_y G_t n)$ distance tests)

Exploiting Symmetries PB-SYM



For each point:

- Compute the K_t
- Compute the K_s
- Cross product

Complexity is the same, but saves computation in practice

Experimental settings

Instance	n	G_x x G_y x G_t	Size	H_s	H_t
Dengue_Lr-Lb	11056	148x194x728	79MB	3	1
Dengue_Lr-Hb	11056	148x194x728	79MB	25	1
Dengue_Hr-Lb	11056	294x386x728	315MB	2	1
Dengue_Hr-Hb	11056	294x386x728	315MB	50	1
Dengue_Hr-VHb	11056	294x386x728	315MB	50	14
PollenUS_Lr-Lb	588189	131x61x84	2MB	2	3
PollenUS_Hr-Lb	588189	651x301x84	62MB	10	3
PollenUS_Hr-Mb	588189	651x301x84	62MB	25	7
PollenUS_Hr-Hb	588189	651x301x84	62MB	50	14
PollenUS_VHr-Lb	588189	6501x3001x84	6252MB	100	3
PollenUS_VHr-VLb	588189	6501x3001x84	6252MB	50	3
Flu_Lr-Lb	31478	117x308x851	117MB	1	1
Flu_Lr-Hb	31478	117x308x851	117MB	2	3
Flu_Mr-Lb	31478	233x615x1985	1085MB	2	3
Flu_Mr-Hb	31478	233x615x1985	1085MB	4	7
Flu_Hr-Lb	31478	581x1536x5951	20260MB	5	7
Flu_Hr-Hb	31478	581x1536x5951	20260MB	10	21
eBird_Lr-Lb	291990435	357x721x2435	2391MB	2	3
eBird_Lr-Hb	291990435	357x721x2435	2391MB	6	5
eBird_Hr-Lb	291990435	1781x3601x2435	59570MB	10	3
eBird_Hr-Hb	291990435	1781x3601x2435	59570MB	30	5

Shared memory machine:

- 2 Intel Xeon E5-2667 v3 (2 times 8 cores)
- 128GB of DRAM
- G++ 5.3 (with OpenMP 4.0)

In practice

Instance	Time (in seconds) VB	VB-DEC	PB	PB-DISK	PB-BAR	PB-SYM	speedup PB-SYM
Dengue_Lr-Lb	219.163	2.283	0.040	0.029	0.035	0.028	1.429
Dengue_Lr-Hb	220.591	13.878	1.298	0.564	1.152	0.499	2.601
Dengue_Hr-Lb	866.445	9.522	0.089	0.082	0.085	0.084	1.060
Dengue_Hr-Hb	871.774	55.206	5.169	2.272	4.563	2.074	2.492
Dengue_Hr-VHb	1056.172	404.845	51.885	11.478	42.994	7.431	6.982
PollenUS_Lr-Lb	518.859	7.639	1.106	0.347	0.922	0.256	4.320
PollenUS_Hr-Lb	12721.001	189.337	23.539	7.700	18.527	4.708	5.000
PollenUS_Hr-Mb	17179.482	3126.947	357.743	86.129	295.791	57.528	6.219
PollenUS_Hr-Hb			2666.104	583.175	2212.626	382.566	6.969
PollenUS_VHr-Lb			2428.126	1004.174	1949.988	759.722	3.196
PollenUS_VHr-VLb			603.789	240.236	488.388	179.834	3.357
Flu_Lr-Lb	926.360	3.691	0.035	0.032	0.034	0.032	1.094
Flu_Lr-Hb	966.328	3.797	0.081	0.046	0.070	0.042	1.929
Flu_Mr-Lb	8591.165	30.355	0.305	0.278	0.298	0.277	1.101
Flu_Mr-Hb	8957.175	32.018	0.714	0.384	0.608	0.323	2.211
Flu_Hr-Lb		536.091	5.702	5.089	5.454	5.059	1.127
Flu_Hr-Hb		591.955	12.795	6.822	10.992	7.072	1.809
eBird_Lr-Lb			396.811	147.951	322.580	125.248	3.168
eBird_Lr-Hb			6969.187	1897.051	5611.158	1067.395	6.529
eBird_Hr-Lb			8373.273	3226.016	6470.764	2229.460	3.756
eBird_Hr-Hb						34577.745	

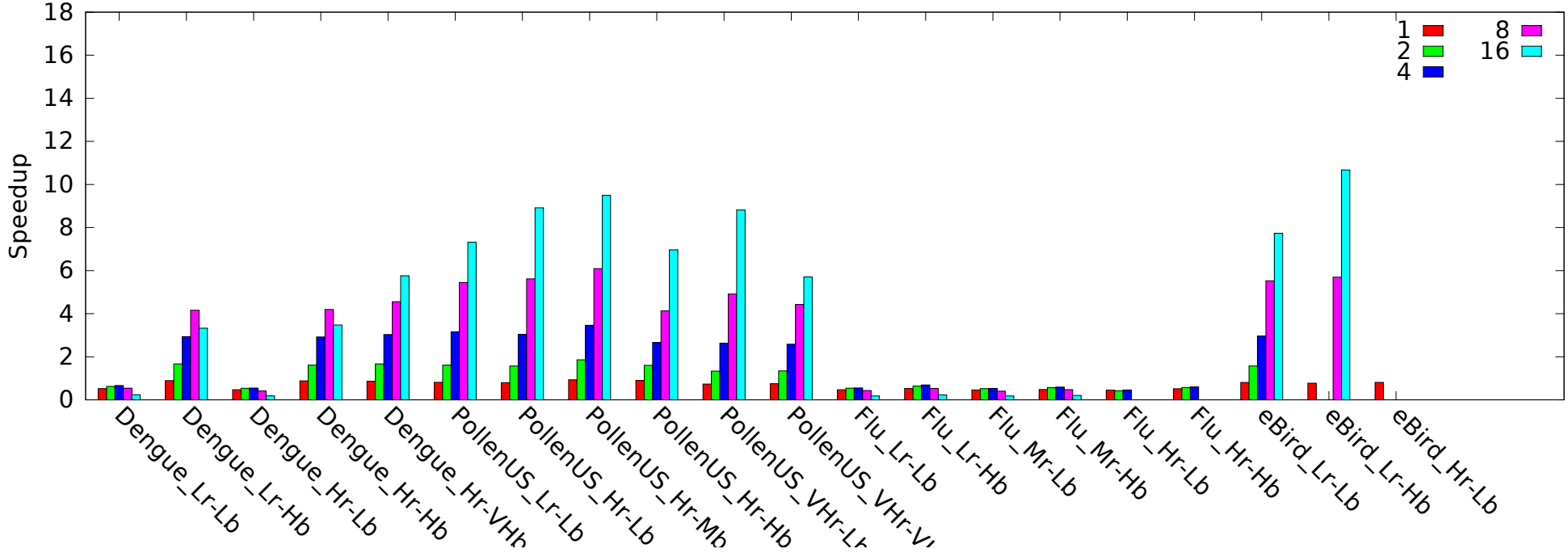
Clearly, PB-SYM is the algorithm to make parallel.

Outline

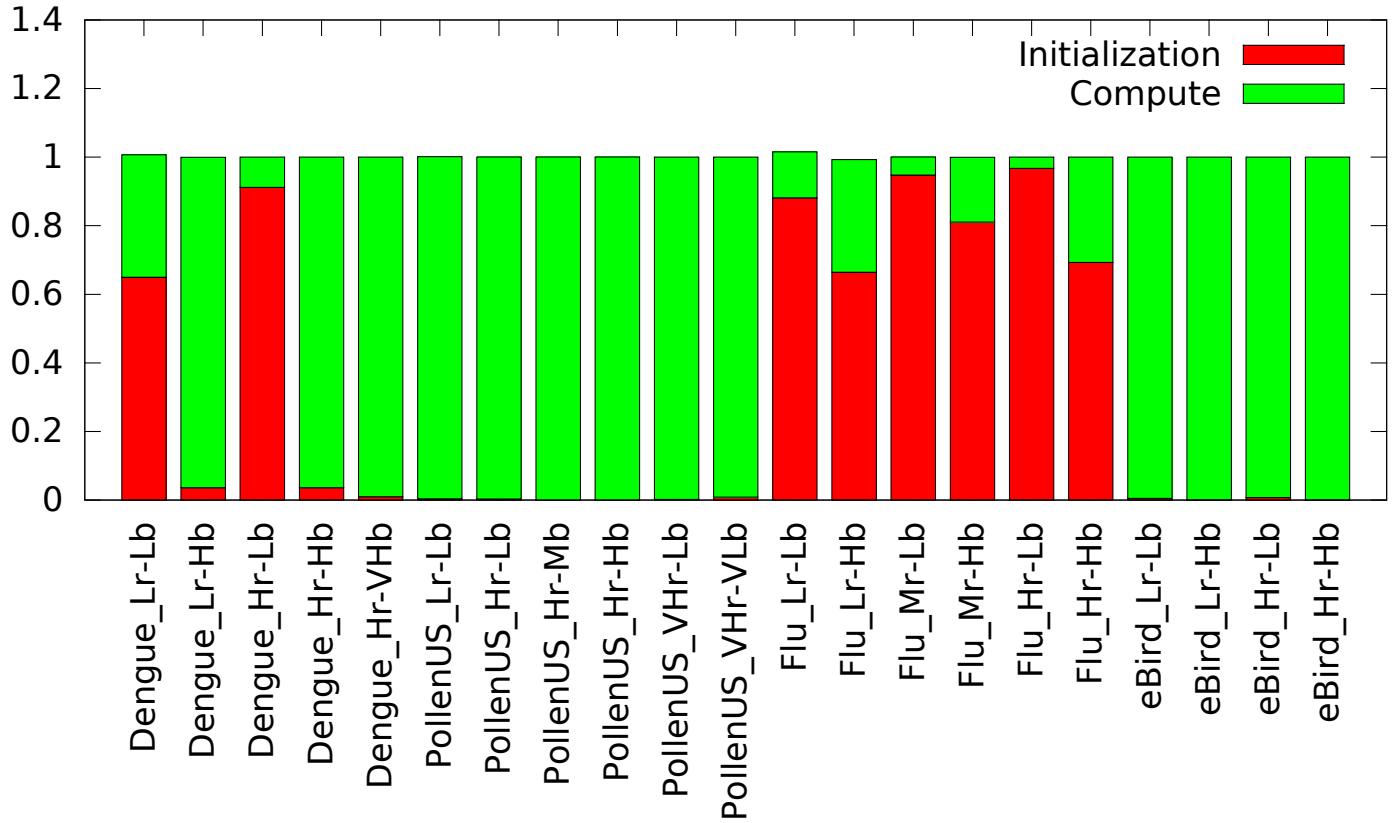
Domain Replication PB-SYM-DR

Each worker:

- Initialize its own memory buffer
- Process some points in its own buffer (with load balancing)
- Participate in reducing the result



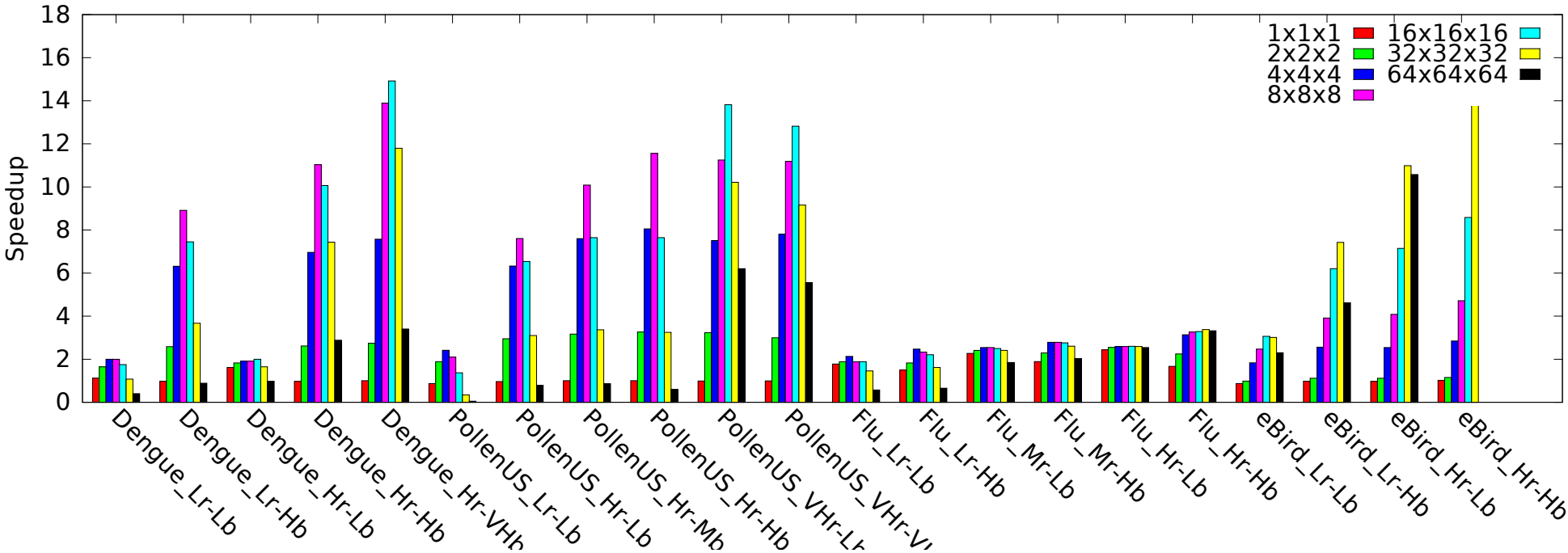
Why is DR bad? Some instances have low computation!



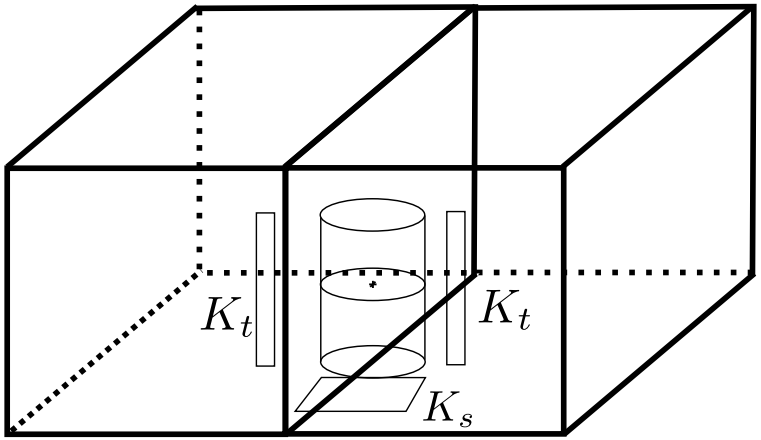
(and some run out of memory)

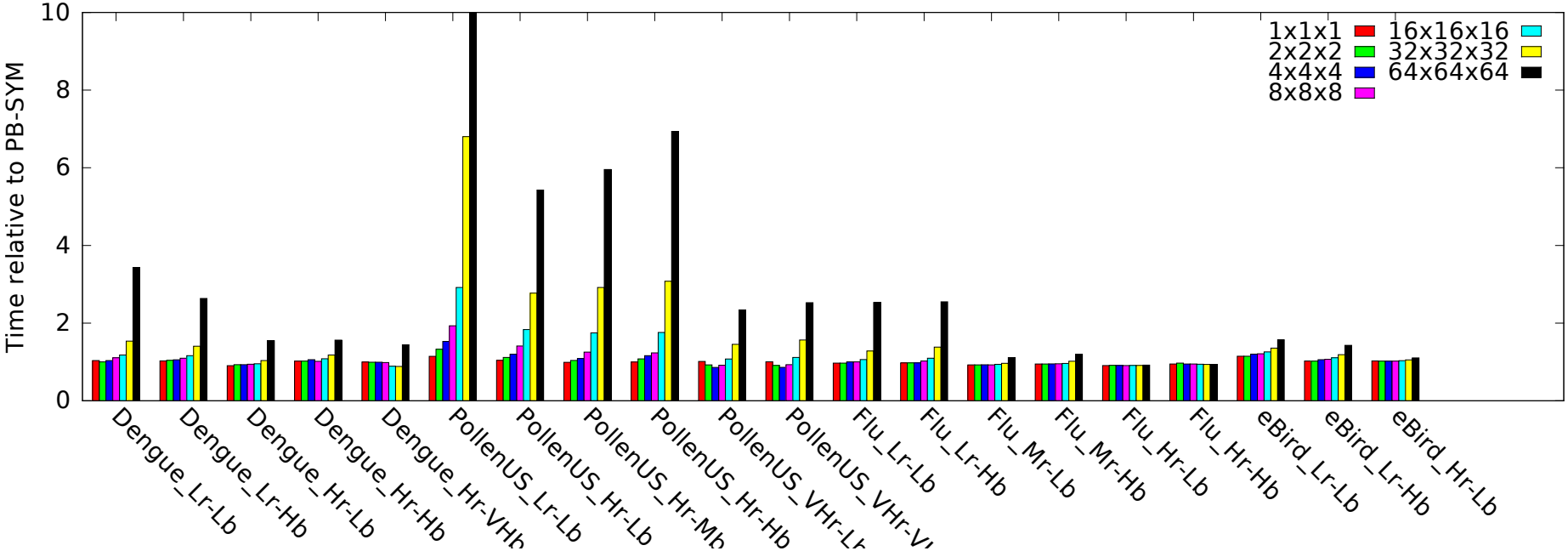
Domain Decomposition PB-SYM-DD

- Decompose the domain in $K \times K$ subdomains
- Each worker process different subdomains (load balanced on the subdomains)



Why is DD bad? Work overhead. Some cylinders are cut!



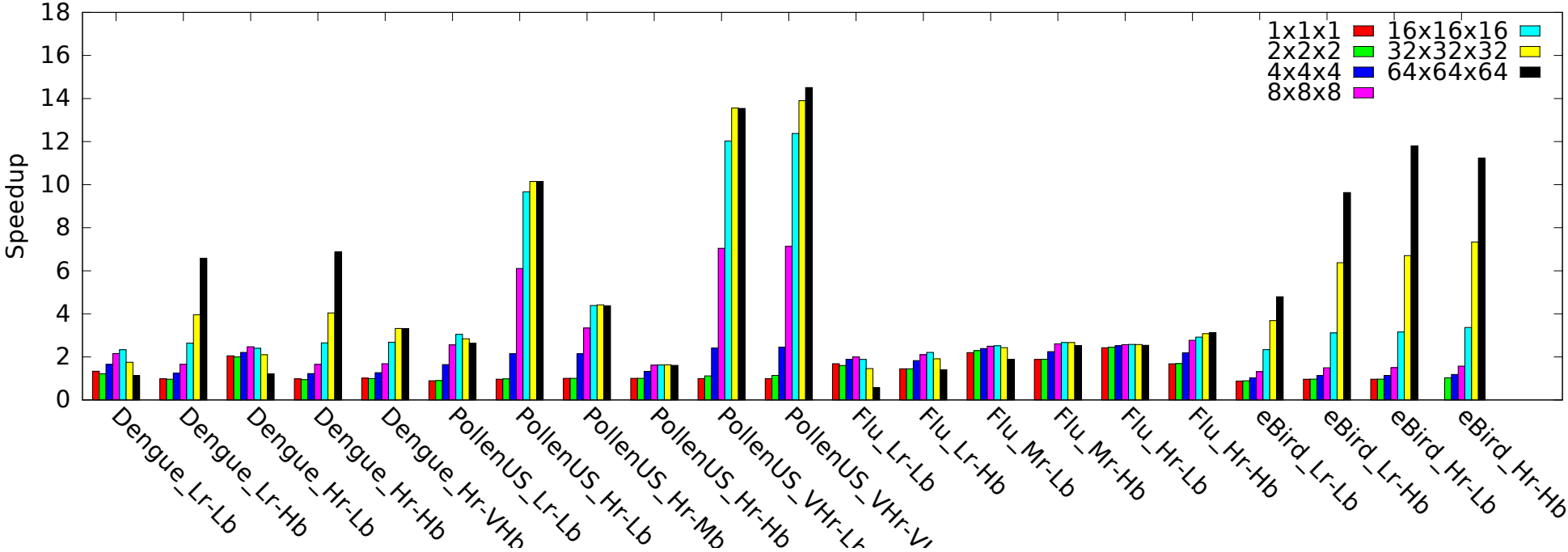


Does anyone know a cheap way to partition better? Some structures admit dynamic programming.

Outline

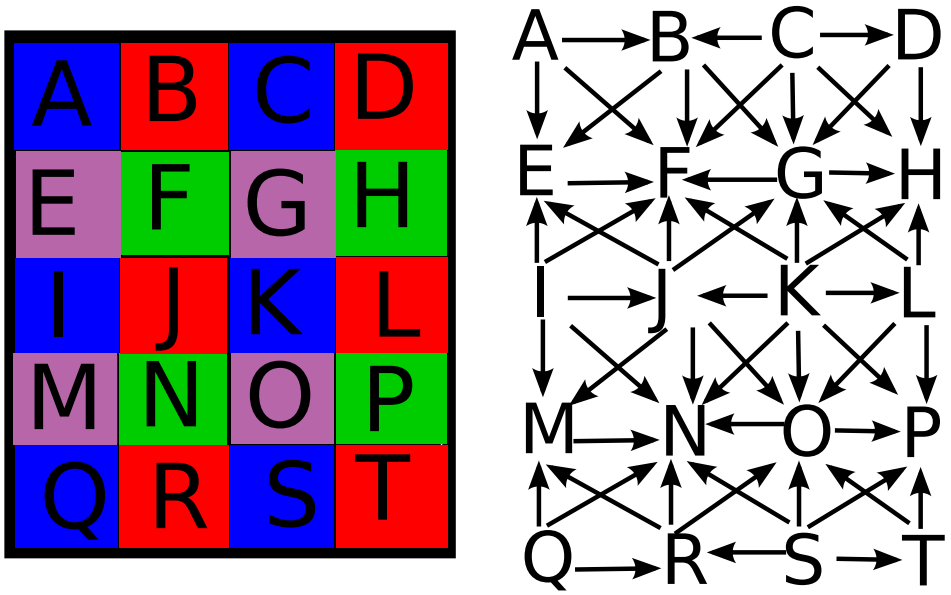
Point decomposition PB-SYM-PD

- Partition the points in a regular AxBxC grid such that each dimension is bigger than the bandwidth.
- For $(a, b, c) \in \{0, 1\}^3$
  - Do in parallel grids $2i + a, 2j + b, 2k + c, \forall i, j, k$



Why is this bad? Too many dependencies?

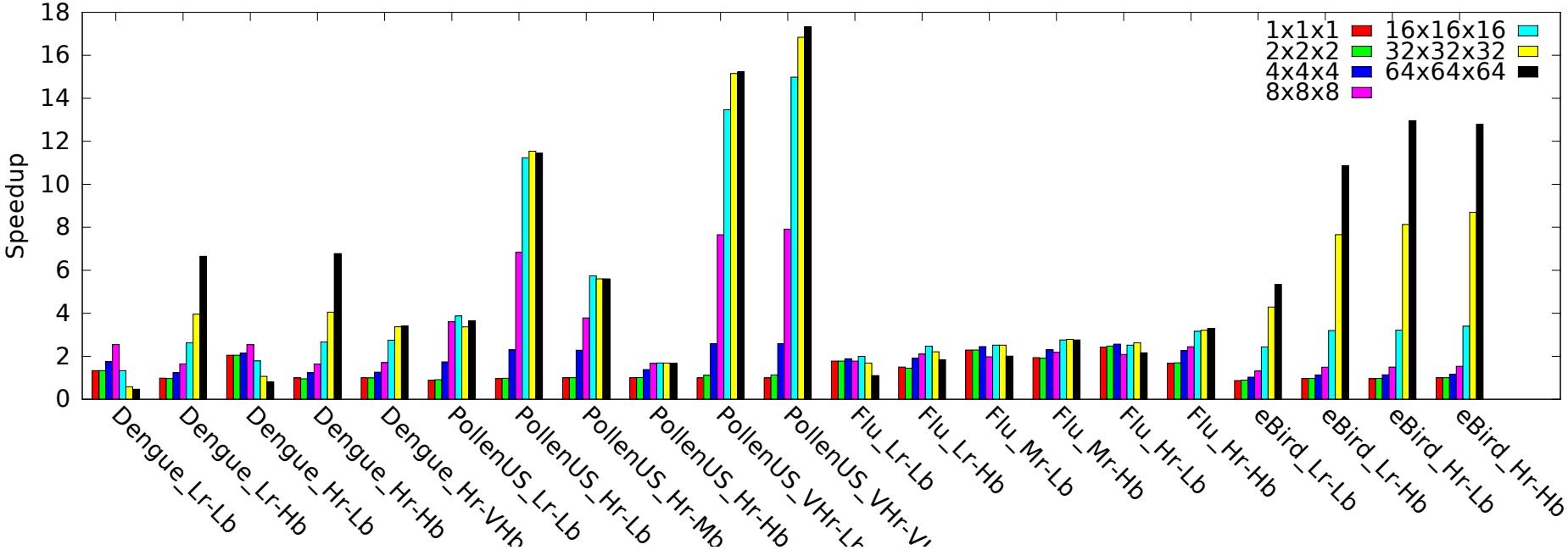
Since all $2i, 2j, 2k$ are done before any $2i+1, 2j, 2k$, there is a forced precedence of $0, 0, 0$ over $3, 0, 0$. But they are not dependent.
This does coloring, instead of doing scheduling.



Building the graph from a coloring is simple (and easily expressed in OpenMP 4.0).

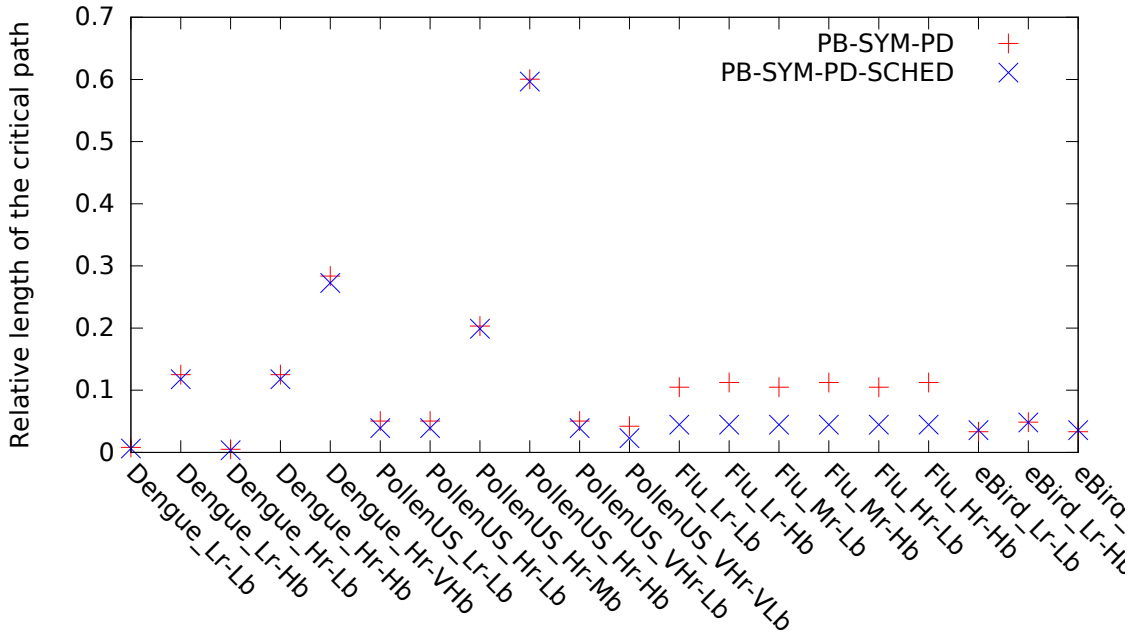
A better coloring with PD-SYM-PD-SCHED

We don't need to color subdomains to minimize the number of colors.
We need a coloring that minimizes the longest chain in the implied graph.
Heuristic: greedily color subdomains in highest number of points first.



(If you don't have a good eye, it is a bit better than before)

Why is it bad? Still too long critical path!



How hard is the coloring/edge-orientation problem to minimize critical path?

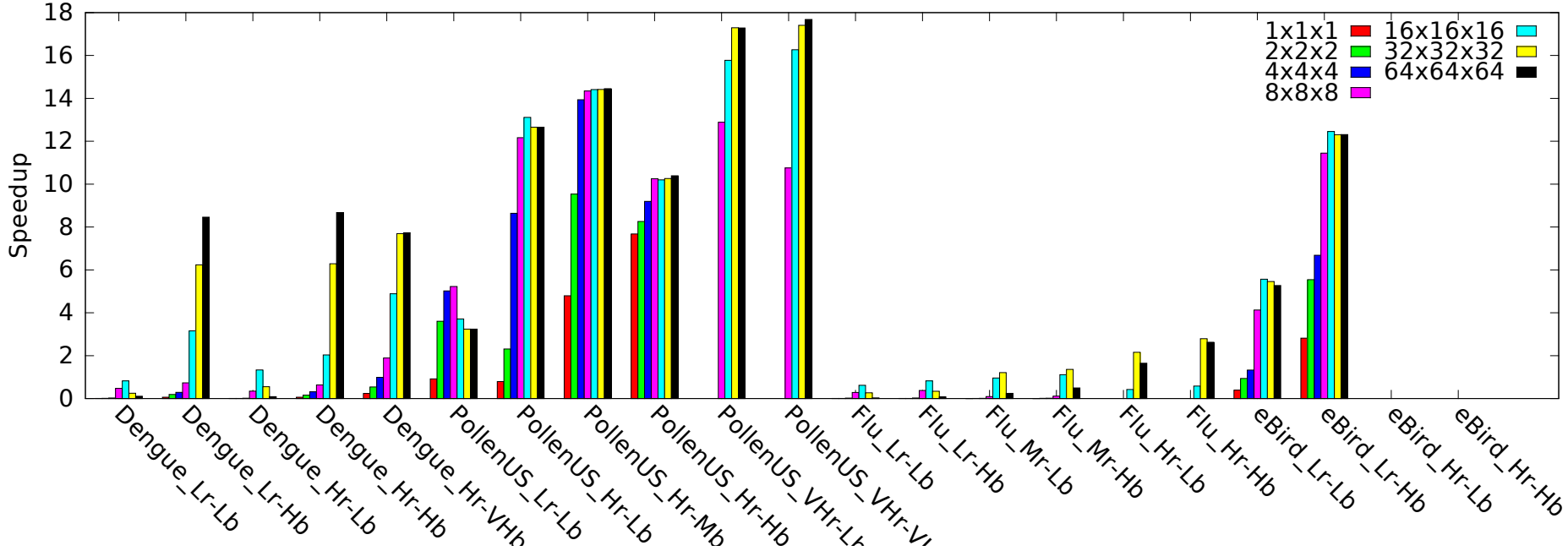
- NP-hard in general graph (harder than coloring)
- Trivial on chains
- Other?

How good is the heuristic?

I need to think more about this...

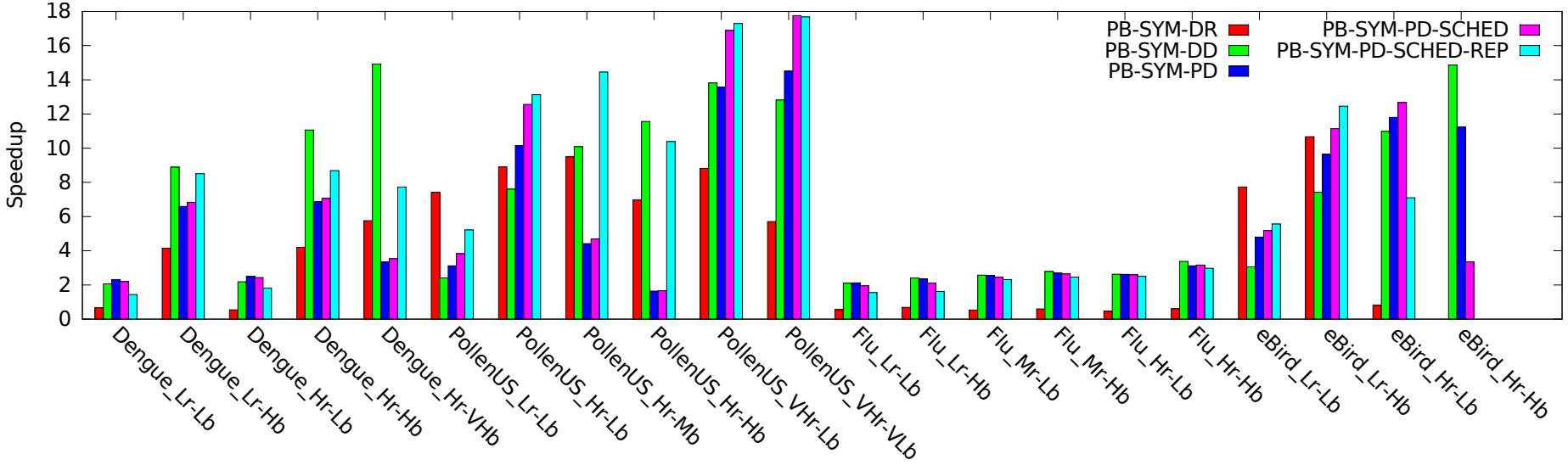
Parallelize long path PB-SYM-PD-SCHED-REP

One can replicate a subdomain and get perfect work parallelism. (at the expense of some memory initialization and reduction.)
Does that sound like moldable DAG scheduling? (Actually a bit more complicated than that, but close enough.)
Heuristic: for all path longer than $\frac{n}{2P}$, add one copy to all tasks on the path



Outline

All methods


Speedup
0
2
4
6
8
10
12
14
16
18
PB-SYM-DR
PB-SYM-DD
PB-SYM-PD
PB-SYM-PD-SCHED
PB-SYM-PD-SCHED-REP
Dengue_Lr-Lb
Dengue_Lr-Hb
Dengue_Hr-Lb
Dengue_Hr-Hb
Dengue_Hr-VHb
PollenUS_Lr-Lb
PollenUS_Hr-Lb
PollenUS_Hr-Mb
PollenUS_Hr-Hb
PollenUS_VHr-Lb
PollenUS_VHr-VLb
Flu_Lr-Lb
Flu_Lr-Hb
Flu_Mr-Lb
Flu_Mr-Hb
Flu_Hr-Lb
Flu_Hr-Hb
eBird_Lr-Lb
eBird_Lr-Hb
eBird_Hr-Lb
eBird_Hr-Hb

Future Works

Other platforms:

- GPU (not quite sure how to approach it)
- KNL
- Distributed Memory

Some algorithmic problems:

- Better way to decompose for PB-SYM-DD
- Formally study the edge orientation problem for PB-SYM-DD-SCHED
- Look deeper into the moldable scheduling connection for PB-SYM-PD-SCHED-REP
- Model everything and derive analytical bounds on performance

Food for thoughts!

- Scheduling makes me think about parallel problems differently.
- Scheduling helps me understand performance.
- Maybe the real problem is to figure which instance of the problem to give to the scheduler.
- Maybe the scheduling algorithm is not always what is most important.

Thank you!

And thanks:

- Dan Janies for pointing out the Flu dataset
- Bora Uçar for pointing out the Gallai-Hasse-Roy-Vitaver theorem
- The US tax payer
  - Support from US NSF XSEDE Supercomputing Resource Allocation (SES170007) is acknowledged.
  - This material is based upon work supported by the National Science Foundation under Grant No. 1652442.

Want to know more?

Contact: `esaule@uncc.edu`
Visit: `http://webpages.uncc.edu/~esaule`