TÜBİTAK

Sabancı Üniversitesi

EURO CC ^{2}

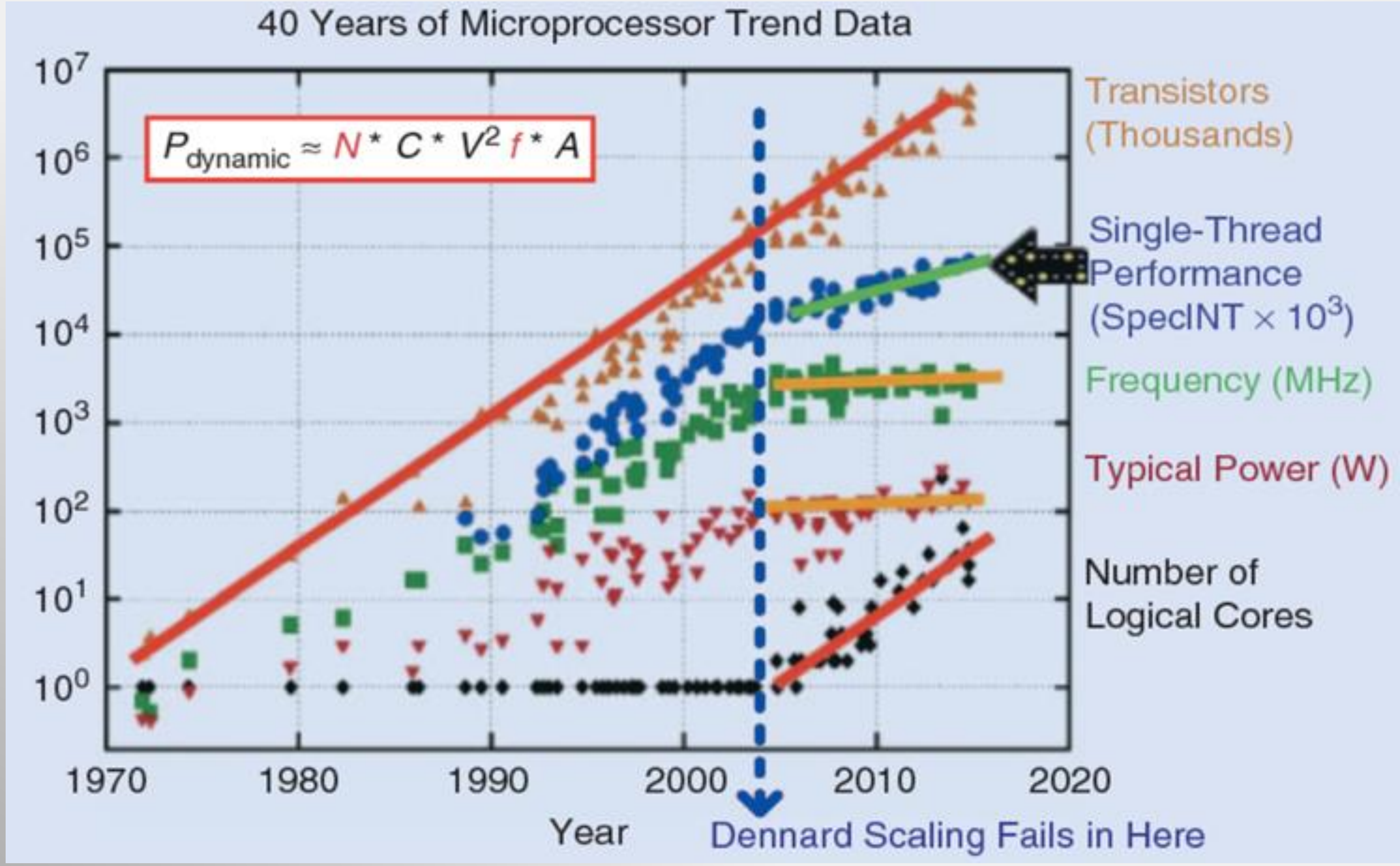
High Performance Computing with Sparse Data

Graphs, Matrices and Tensors

Kamer Kaya, Sabancı University

ncc@ulakbim.gov.tr

Parallelism:
CPUs; much faster/many cores.



https://vision.hipeac.net/new-hardware--heterogeneous-and-domain-specific-acceleration.html

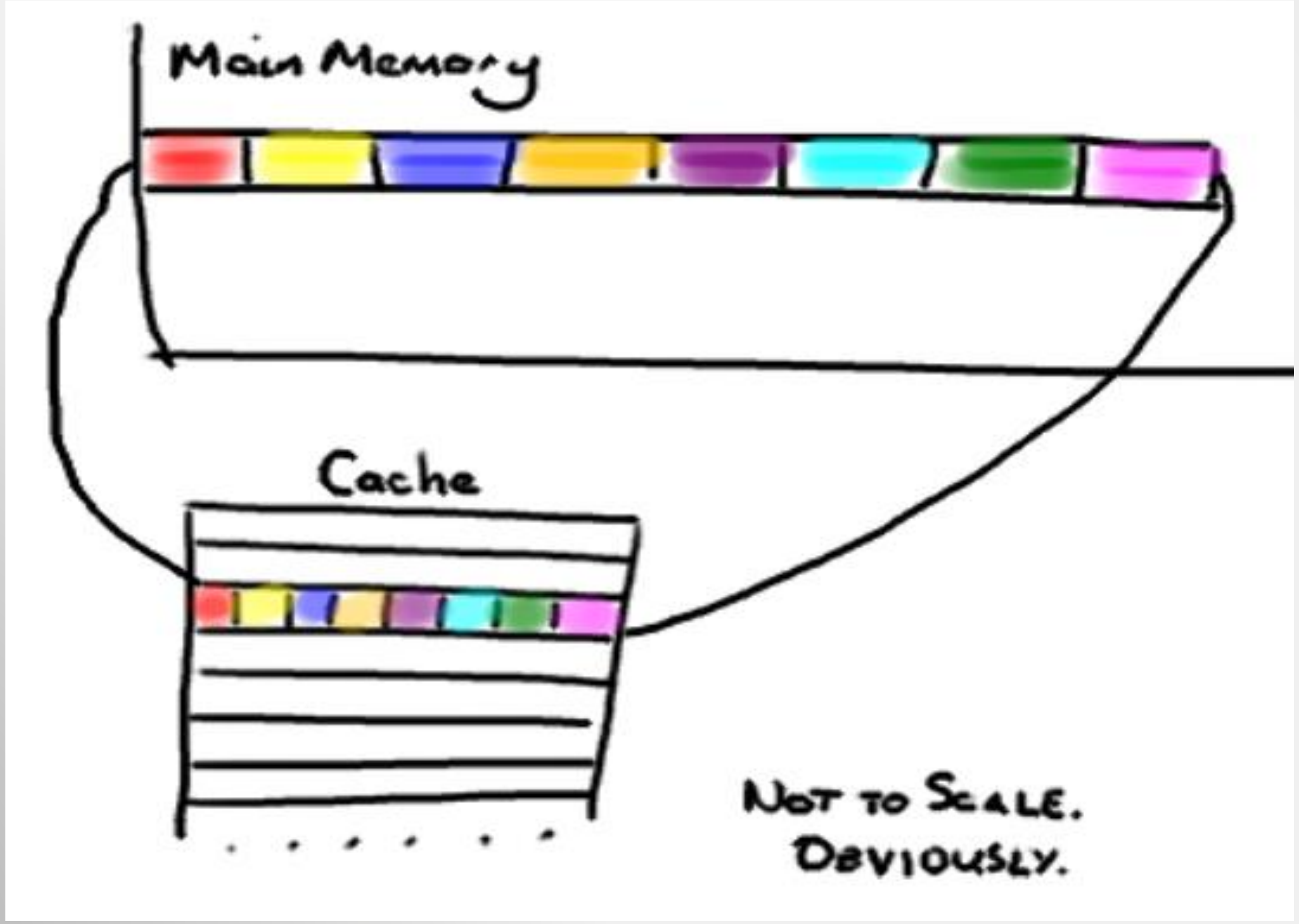
Processor frequencies do not accelerate as fast as they were used to be - instead, they're getting more and more multi-core.

Not all these cores work at full performance at the same time.

Time Moore: Exploiting Moore's Law From The Perspective of Time

Liming Xiu • Published 2019 • Computer Science • IEEE Solid-State Circuits Magazine

Memory remembers, memory forgets…



Spatial Locality

Temporal Locality

https://trishagee.com/2011/07/22/dissecting_the_disruptor_why_its_so_fast_part_two__magic_cache_line_padding/

Parallelism: Efficient if loads are equal

EURO^2

```
// Function to perform BFS
void BFS(int startNode, const vector<vector<int>>& adjacencyList) {
    vector<bool> visited(adjacencyList.size(), false); // Track visited nodes
    queue<int> q; // Queue for BFS

    // Start BFS from the given node
    visited[startNode] = true;
    q.push(startNode);

    cout << "BFS Traversal starting from node " << startNode << ": ";

    while (!q.empty()) {
        int currentNode = q.front();
        q.pop();

        // Process the current node
        cout << currentNode << " ";

        // Add all unvisited neighbors to the queue
        for (int neighbor : adjacencyList[currentNode]) {
            if (!visited[neighbor]) {
                visited[neighbor] = true;
                q.push(neighbor);
            }
        }
    }
```

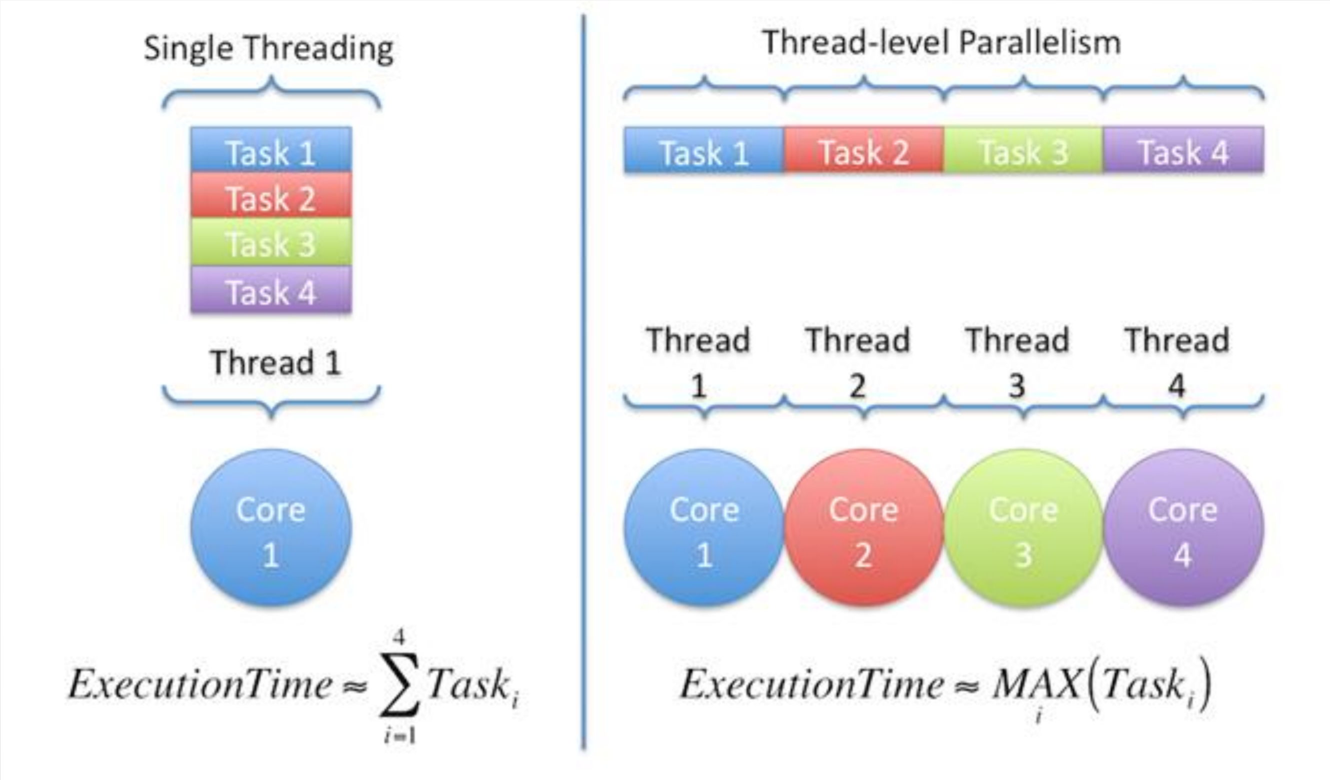
Sequential BFS

The cost of this loop deviates a lot! Hub-vertices will have high cost, low-degree vertices will have low cost.

Big problem for caches... Neighborhoods are not ordered. A probable cause of cache misses.

A problem for parallelism if this queue is used. Not trivial but doable.

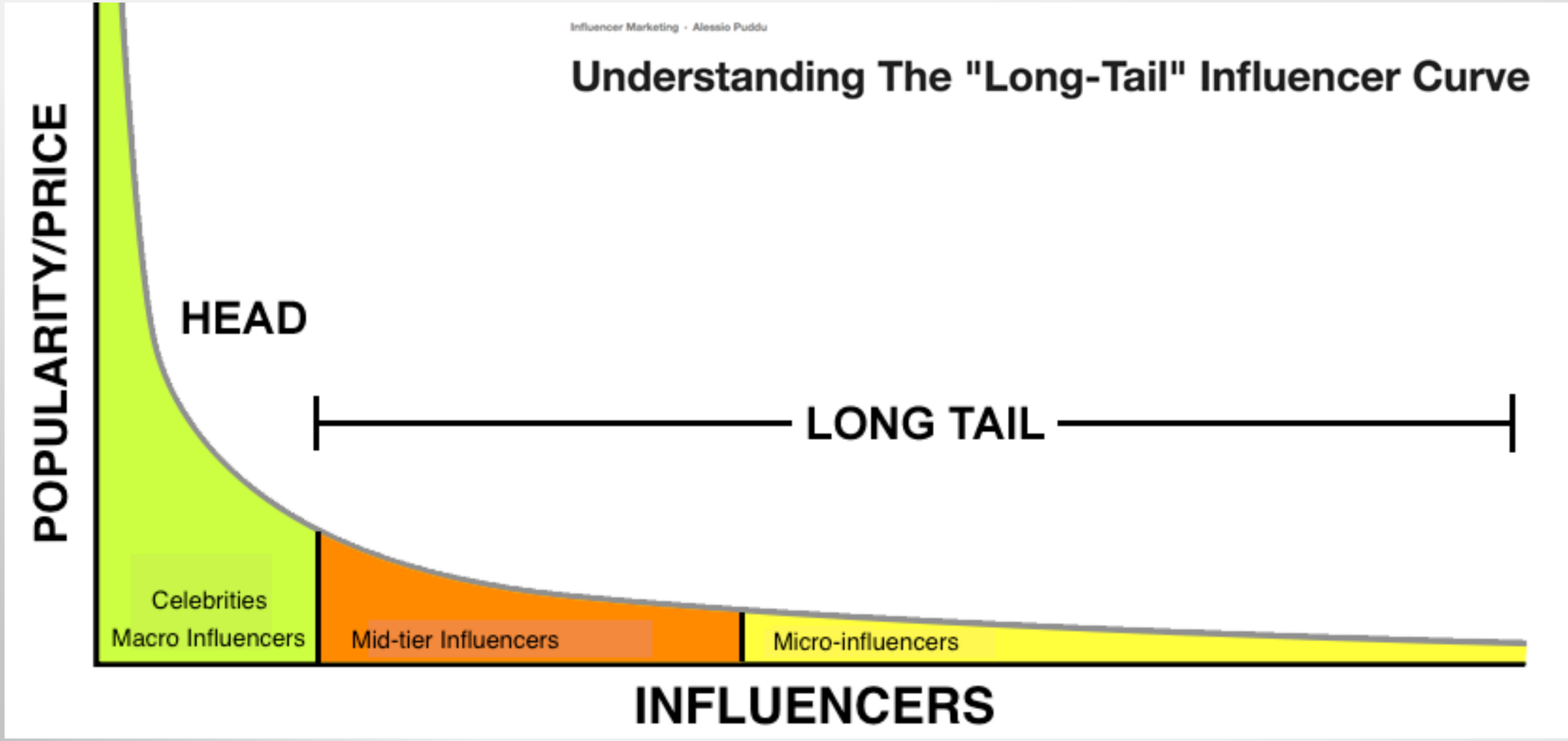
Parallelism: Efficient if loads are equal



Overview of Performance Measurement and Analytical Modeling Techniques for Multi-core Processors

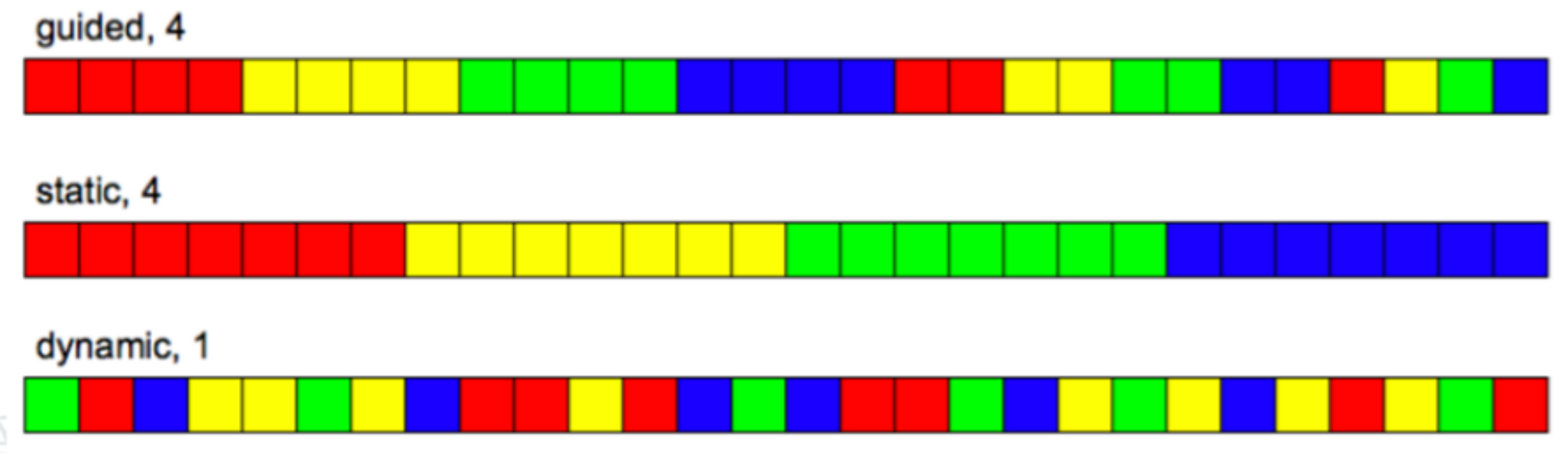
Garrison Prinslow, gprinslow@gmail.com

Parallelism: Think like a vertex and suffer



Parallelism:
Think like a set of vertices

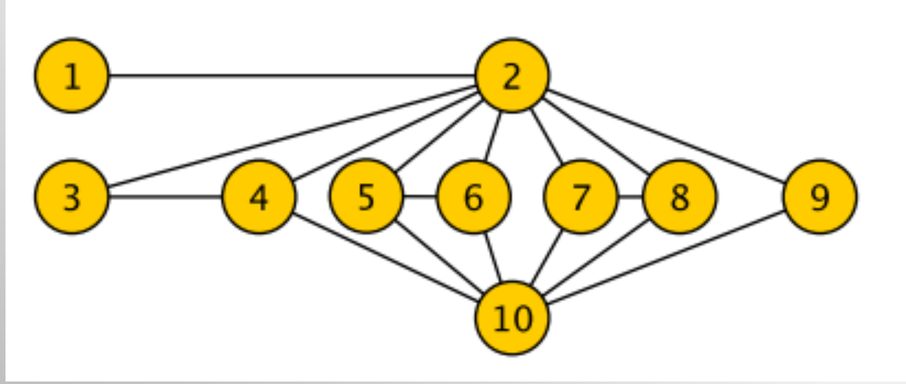
OpenMP Scheduling Policies
Vertices: colors are threads

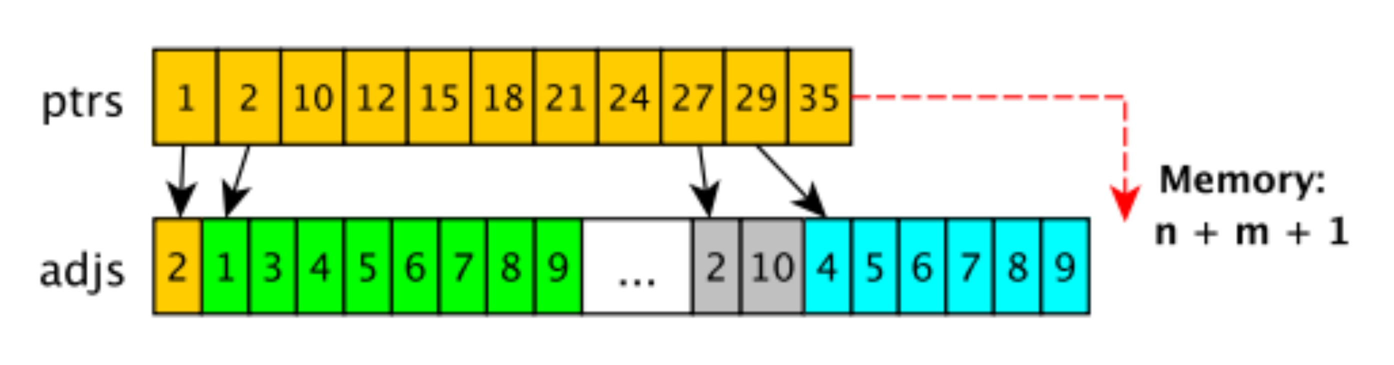


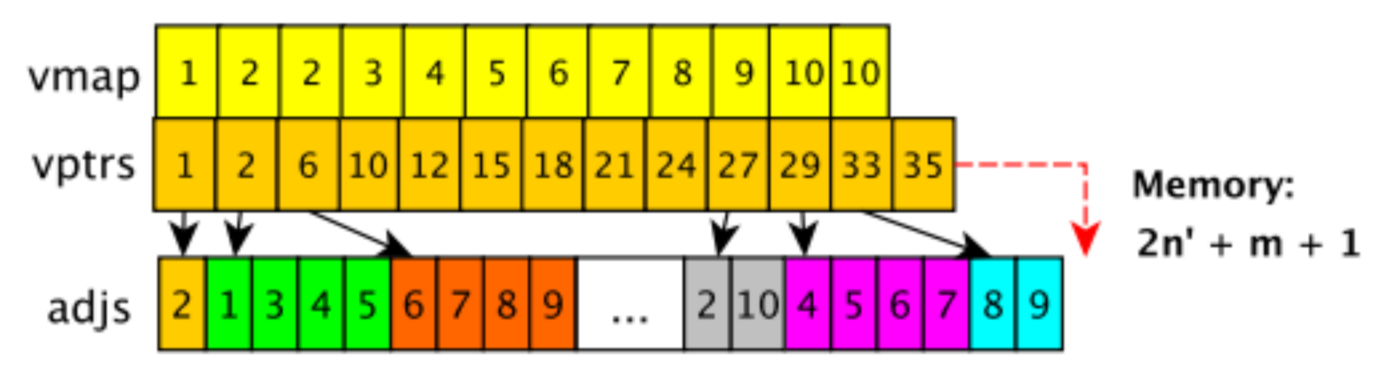
CPU	GPU
Usually OK	Not Trivial
Not Trivial	Not Trivial
Usually OK	Not Trivial

Handling Hub-vertices: A new data structure

EURO²

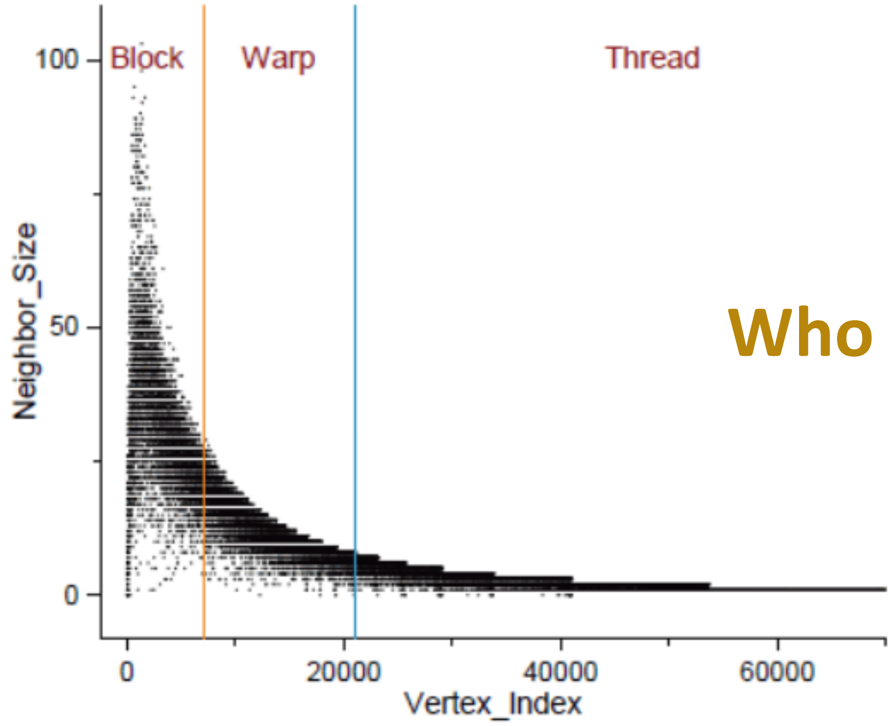






Virtual/ghost vertices

Handling Hub-vertices: Think like a vertex and do some more



Who is thinking?

Fig. 2.
The neighbor size distribution with vertex indexes of the graph soc-slashdot0902. The yellow line and the blue line divide the vertices into three scopes according to the average degrees. Kernels with different size are assigned for each scope.

C. Gui, L. Zheng, P. Yao, X. Liao and H. Jin, "Fast Triangle Counting on GPU," *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, Waltham, MA, USA, 2019, pp. 1-7, doi: 10.1109/HPEC.2019.8916216.

Handling Hub-Vertices: Reading hub data

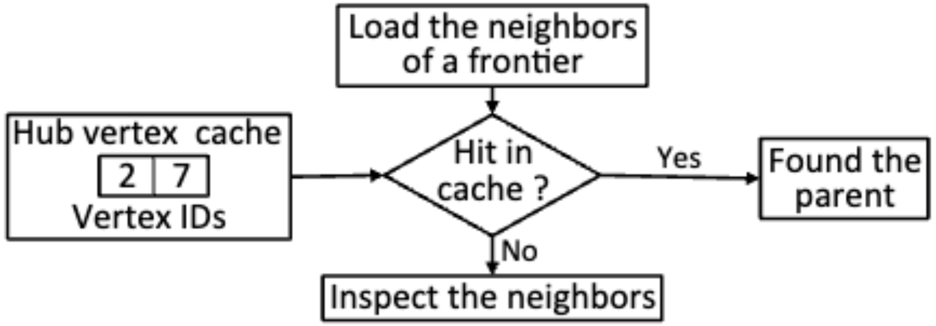
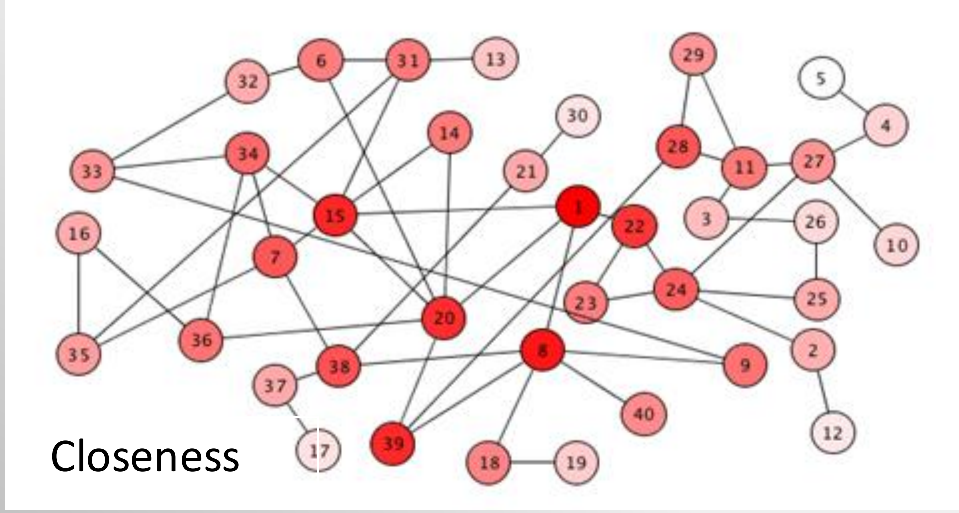
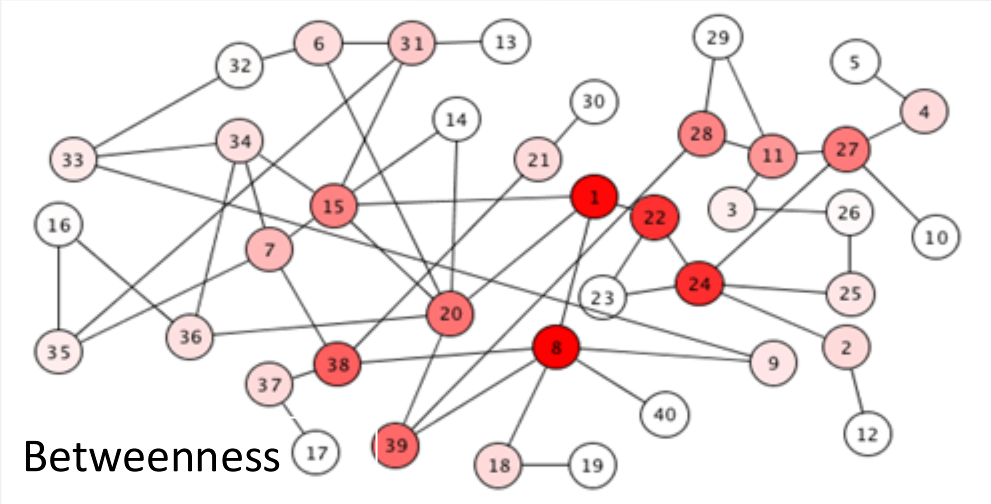


Figure 2.12: Hub vertex cache design, using the level 4 traversal in example graph from Figure 2.1.

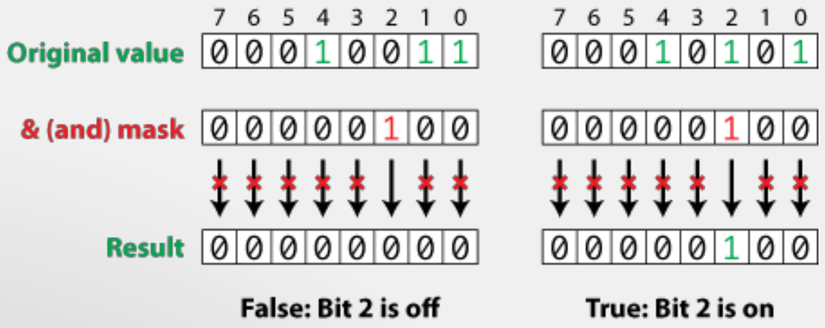
H. Liu and H. H. Huang, "Enterprise: breadth-first graph traversal on GPUs," *SC '15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Austin, TX, USA, 2015, pp. 1-12, doi: 10.1145/2807591.2807594.

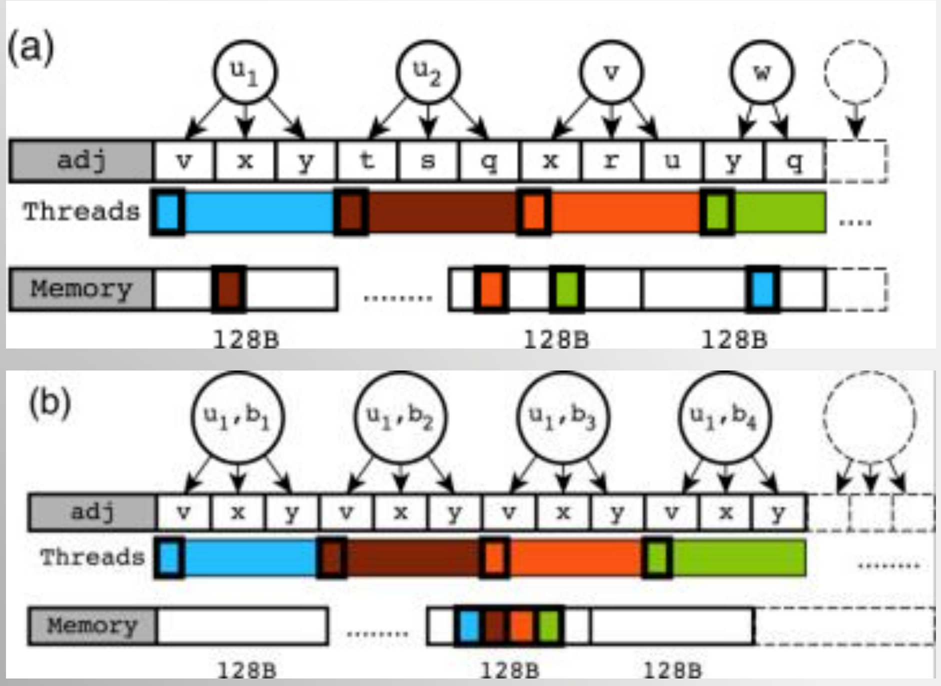
Multiple traversals: Graph Centrality





Multiple traversals: Graph Centrality

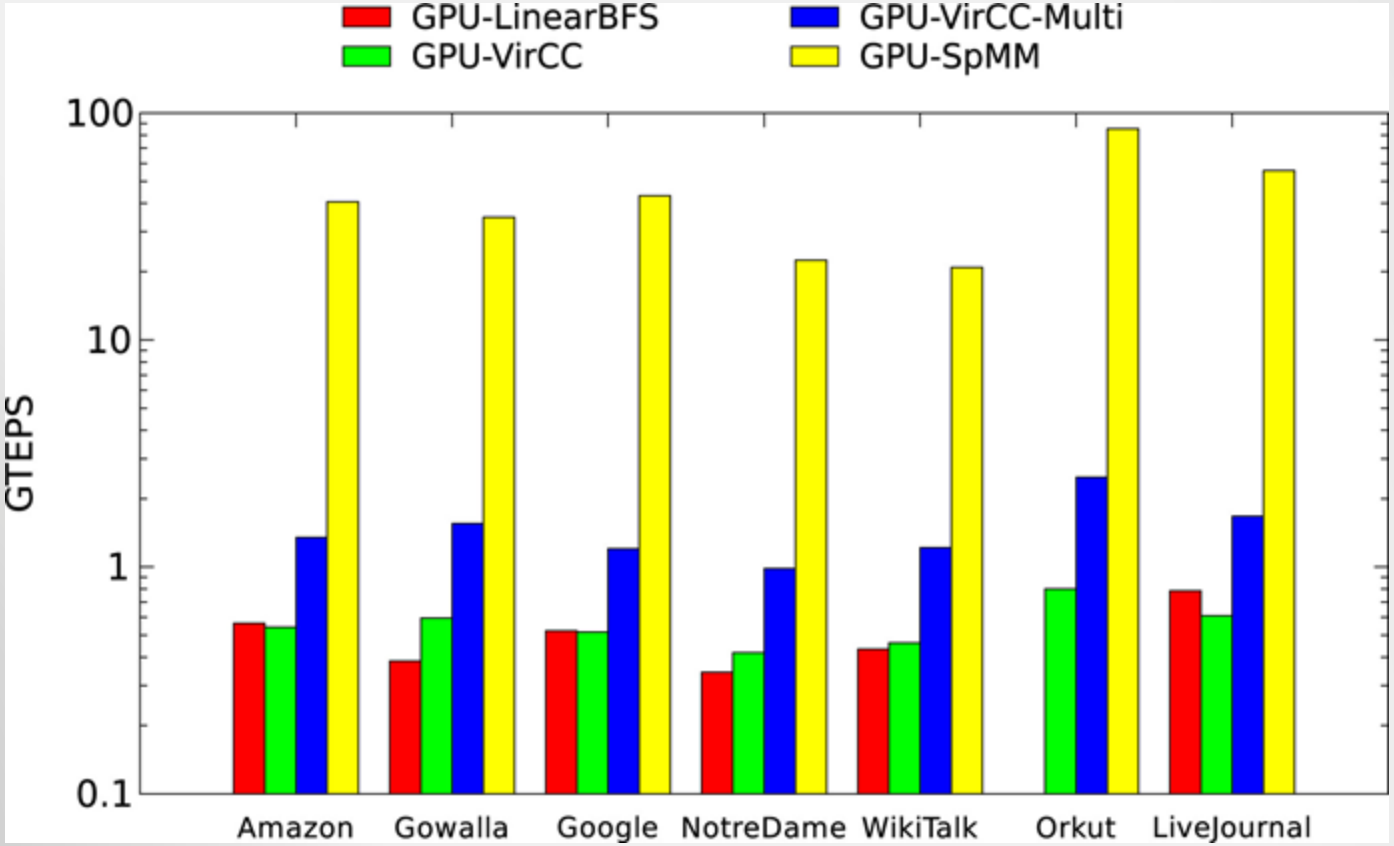




Algorithm 5: CC-SpMM: SpMM-based centrality computation

Data: $G = (V, E)$, $\mathcal{B}$
Output: ccent[.]
▷Init
ccent$[v] \leftarrow 0, \forall v \in V$
$\ell \leftarrow 0$
partition V into k batches $\Pi = \{V_1, V_2, \ldots, V_k\}$ of size $\mathcal{B}$
for each batch *of vertices* $V_p \in \Pi$ **do**
  $x^0_{s,s} \leftarrow 1$ if $s \in V_p$, 0 otherwise
  while $\sum_i \sum_s x^\ell_{i,s} > 0$ **do**
    ▷SpMM
    $y^{\ell+1}_{i,s} = \text{OR}_{j \in adj(i)} x^\ell_{j,s}, \forall s \in V_p, \forall i \in V$
    ▷Update
    $x^{\ell+1}_{i,s} = y^{\ell+1}_{i,s}$ AND not$(\text{OR}_{\ell' \leq \ell} x^{\ell'}_{i,s}), \forall s \in V_p, \forall i \in V$
    $\ell \leftarrow \ell + 1$
    for all $v \in V$ **do**
      ccent$[v] \leftarrow$ ccent$[v] + \frac{\sum_s x^\ell_{v,s}}{\ell}$
return ccent[.]

Multiple traversals: Graph Centrality



Multiple Traversals: Influence Maximization

IM wants to find K starting nodes in a given graph which maximizes the diffusion of information (i.e., influence).

- Monte Carlo simulations are widely used in the literature.
  - Simulate/sample + traverse
- For large-scale graphs and large-scale simulations, finding these K nodes can take hours or even days.

Multiple Traversals: Influence Maximization

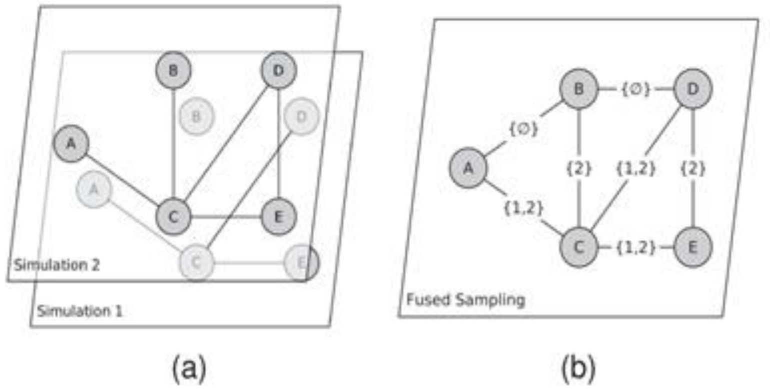


Fig. 3. (a) Two sampled subgraphs of the toy graph from Fig. 1 with 5 vertices and 3 and 5 edges, respectively. (b) The simulations are performed in a way to be fused with sampling. Each edge is labeled with the corresponding sample/simulation IDs.

Dataset	$p = 0.01$ IMM ($\epsilon = 0.13$)	$p = 0.01$ IMM ($\epsilon = 0.5$)	$p = 0.01$ INFUSER MG	$p = 0.1$ IMM ($\epsilon = 0.13$)	$p = 0.1$ IMM ($\epsilon = 0.5$)	$p = 0.1$ INFUSER MG
Amazon	62.67	4.95	**2.09**	24.80	**2.72**	9.99
DBLP	55.92	**4.02**	7.02	168.68	15.34	**11.83**
Epinions	72.39	7.55	**1.91**	86.10	7.82	**1.96**
LiveJournal	9078.34	860.38	**265.84**	-	1527.58	**153.46**
NetHEP	2.80	0.29	**0.08**	6.31	0.65	**0.18**
NetPhy	3.55	0.39	**0.36**	22.57	2.06	**0.73**
Slashdot0811	135.54	12.33	**2.69**	146.09	14.48	**2.04**
Slashdot0902	107.83	10.63	**3.11**	129.15	13.29	**1.81**
Orkut	24300.59	2279.10	**654.52**	-	1987.11	**195.60**
Pokec	2646.98	247.36	**227.24**	-	611.36	**74.38**
Twitter	298.97	26.70	**3.07**	261.94	23.70	**2.52**
Youtube	201.65	**19.42**	26.18	740.35	78.51	**26.31**

G. Göktürk and K. Kaya, "Boosting Parallel Influence-Maximization Kernels for Undirected Networks With Fusing and Vectorization," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1001-1013, 1 May 2021, doi: 10.1109/TPDS.2020.3038376.

Thanks

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia