INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

acm In-Cooperation

sighpc

HiPa: Hierarchical Partitioning for Fast PageRank on NUMA Multicore Systems

YuAng Chen, Yeh-ching Chung
The Chinese University of Hong Kong, Shenzhen

Content

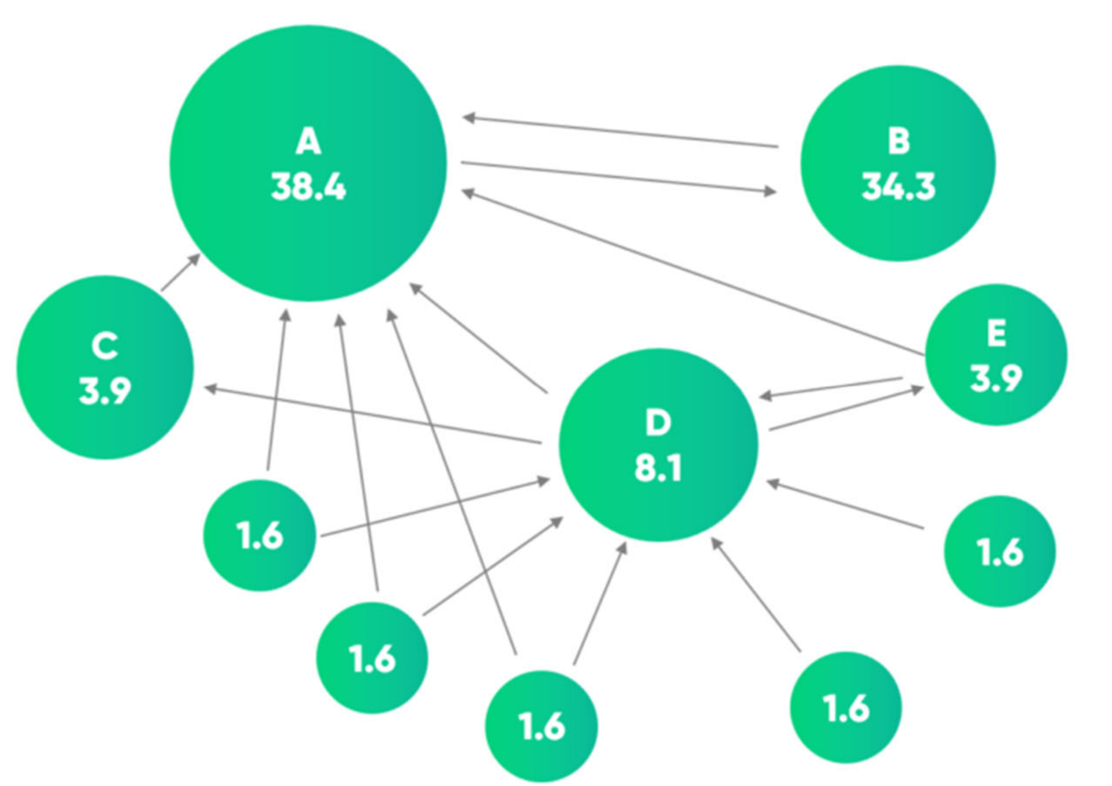
1 Introduction

- PageRank
- Issues

1 Introduction

(a) PageRank

PageRank



PageRank:
Larry Page
- or webpage

1 Introduction

(a) PageRank

$$PR_{new}(v) = 1 - d|V| + d \times \Sigma_{u \in Ein(v)} PR_{old}(u) \, |E_{out}(u)|$$

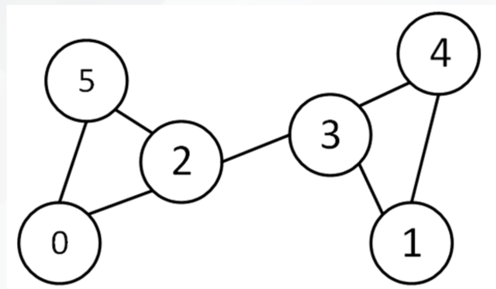
parameter	description
d	damping factor
$\lvert V \rvert$	the total number of vertices in a graph
u	in-neighbors of v
$\lvert E_{out}(u) \rvert$	the number of outgoing edges of u

the workload of PageRank mainly depends on the edges of the graph

1 Introduction

(b) Issue of graph processing on multicore systems

Pointer-based Data Structure



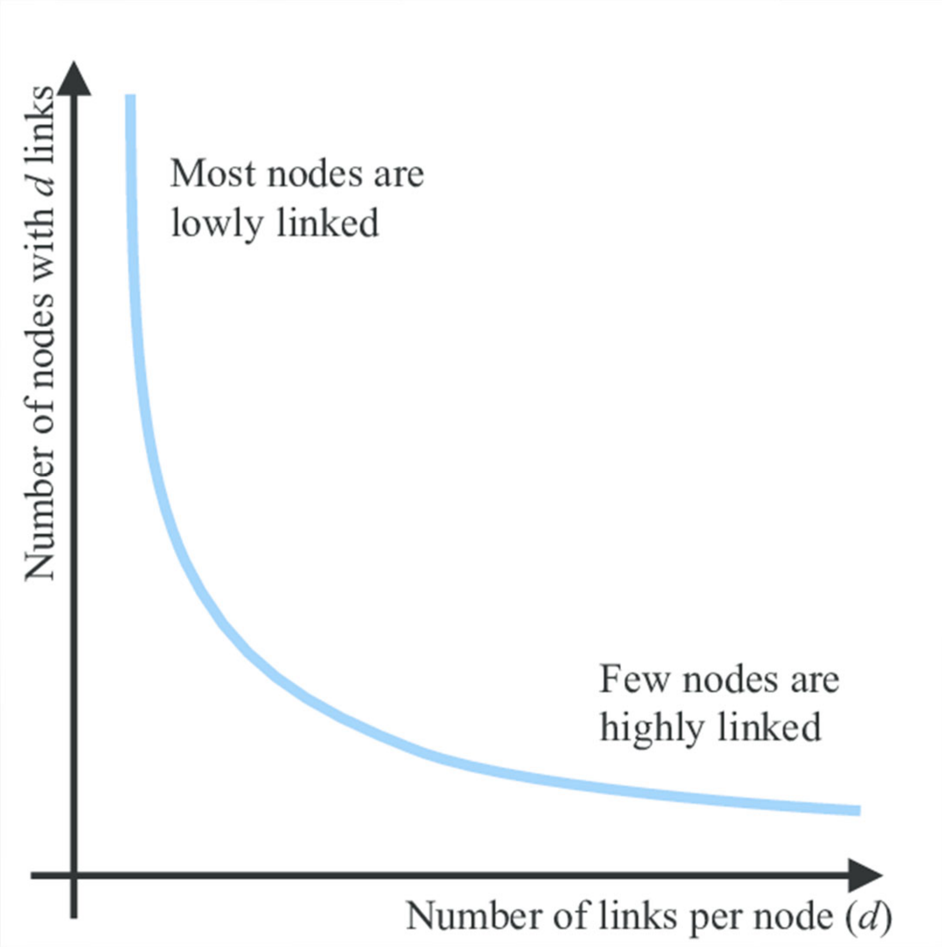
Random graph

	0	1	2	3	4	5
0			1			1
1				1	1	
2	1			1		1
3		1			1	
4		1		1		
5	1		1			

1 Introduction

(b) Issue of graph processing on multicore systems

Skewed power-law degree distribution

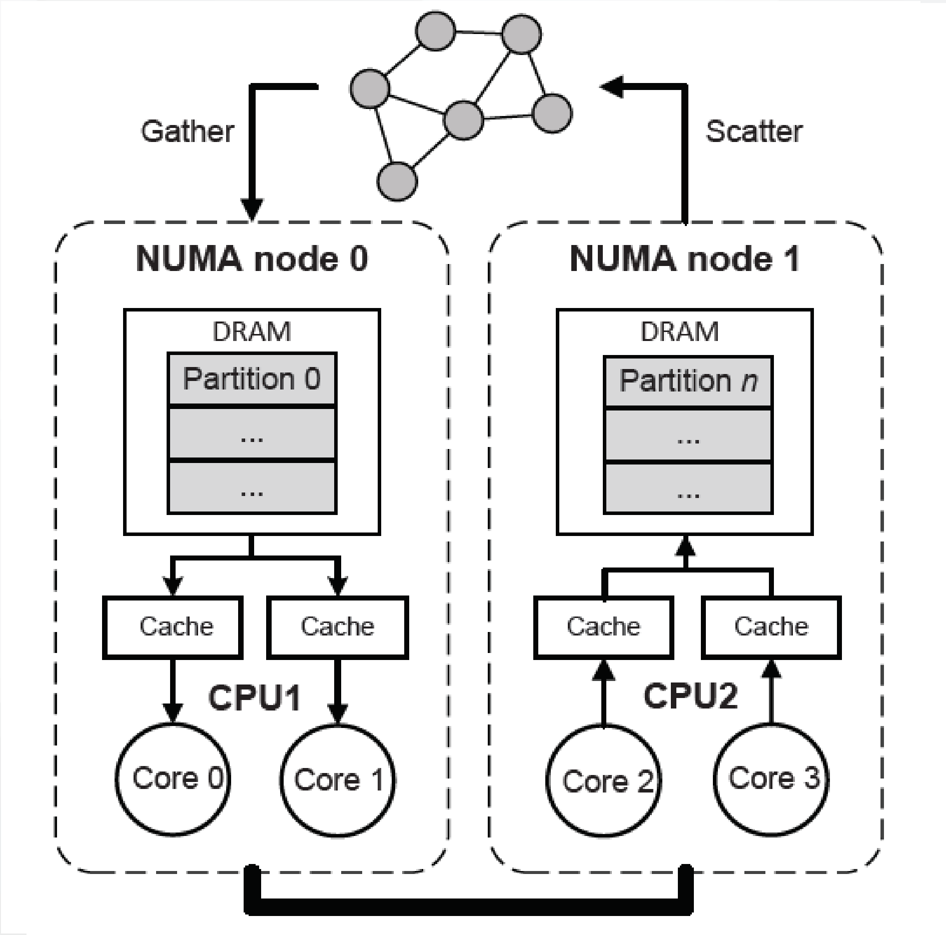


2 Hierarchical Partitioning

- Memory
- Cache
- Thread

2 Design of HiPa

(x) Abstraction



2 Design of HiPa

(a) NUMA-aware Partitioning

Goal:
co-locate the computation and data within the same NUMA node.

Step 1 - Intuition:
Each NUMA node *i* is allocated with the same number of edges $|E|/N$.

$$|E_i| = \frac{|E|}{N}$$

$$V_i = \{v \in V | \sum_{v \in V} D(v) = \frac{|E|}{N}\}$$

Step 2 - Roundup:

- The number of vertices allocated to a NUMA node must be a multiple of L2-partitions;
- The size of a L2-partition P is fixed to $|P|$= {L2 cache size} / {single vertex size}.

$$|\tilde{V}_i| = ceil(\frac{|V_i|}{|P|}) \cdot |P| = (\frac{|V_i| - 1}{|P|} + 1) \cdot |P| = n_i \cdot |P|$$

$$|\tilde{E}_i| = \sum_{v \in \tilde{V}_i} D(v)$$

2 Design of HiPa

(b) Cache-aware Partitioning

Goal:
promote high cache **locality**

Step 3 – Distribution of partitions

- These L-2 partitions are organized in groups G and then distributed to cores C.
- Each group G of Core j $(1 < j < C)$ contains (roughly) the same no. of edges

$$n_i = \sum_{j=1}^{C} m_j$$

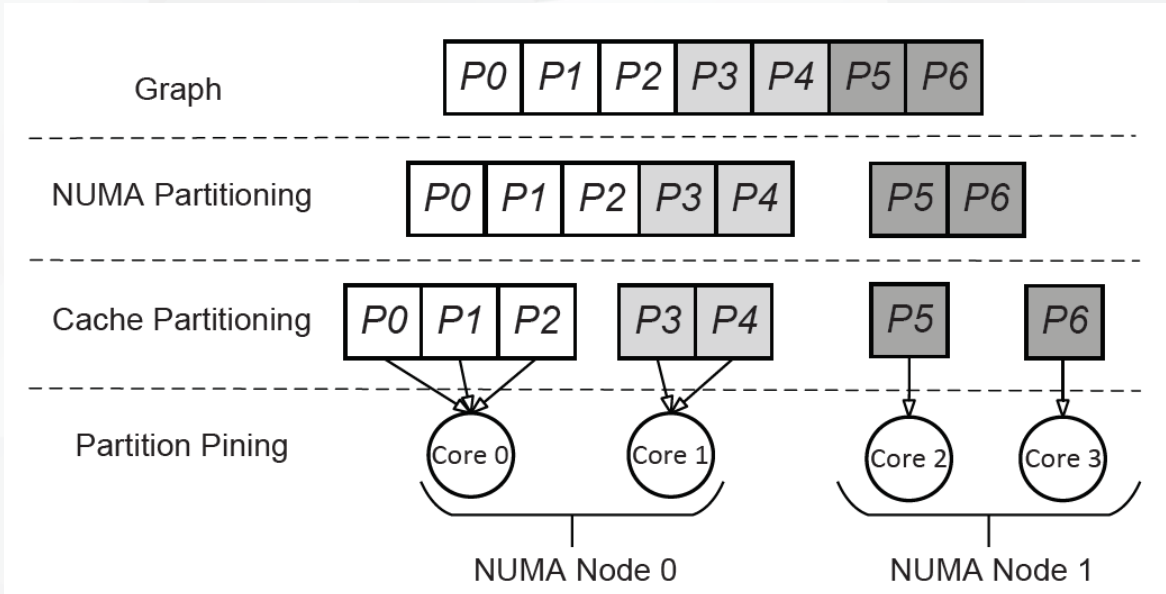
$$|G_j| = m_j \cdot |P|$$

$$\frac{|\tilde{E}_i|}{C} = \sum_{v \in G_j} D(v)$$

Why partitioning by L2 cache?
Cache-able disjoint partitions of a graph, limits the vertex access within L2 cache for high cache locality
The optimal partition size is to be discussed later.

2 Design of HiPa

(c) Partitioning Result



1. The boxes represent cache-able partitions of the graph data.
2. P0-2 hold 10 edges, P3-4 hold 15 edges, and P5-6 hold 30 edges.
3. The processor cores are allocated with *unequal* numbers of partitions but *equal* number of edges.

2 Design of HiPa

(d) Thread Management

Algorithm 2: Numa-aware scatter-gather model

Input: *numa_Partitions* → numa-ly allocated partitions
Input: *numa_Threads* → numa-ly bound threads

```
Function Th_Func(numa_part)
    for i ← 0 to iter do
        scatter(numa_part);
        synchronize with other threads;
        gather(numa_part);
for th ∈ numa_Threads do in parallel        ▷ parallel region
    match th with p ∈ numa_Partitions;
    th calls Th_Func(p);
Graph ← concatenate (numa_Partitions)
```

3 Evaluation

- Execution Time
- Memory Access
- Sensitivity

3 Evaluation

(x) Experiment Setup

Machine	Configurations
Intel Xeon Silver 4210 processors	2
Physical, Virtual Cores	20, 40
L1, L2, LLC Caches	64KB, **1MB**, 13.75MB.

Contemporary works	Descriptions
P-PR	Hand-optimized code, partition-centric
V-PR	Hand-optimized code, vertex-centric
GPOP	Framework, partition-centric
Polymer	Framework, vertex-centric, NUMA-aware

3 Evaluation

(a) Execution Time

Execution time (in seconds) of 20-iteration PageRank with various implementations.

	HiPa	p-PR	v-PR	GPOP	Polymer
journal	**0.31**	0.41	0.54	1.14	1.72
pld	**2.43**	3.37	8.44	4.18	22.27
wiki	**1.74**	1.80	1.96	3.90	4.63
kron	**7.20**	10.06	32.82	11.29	76.62
twitter	**8.43**	9.83	12.09	14.91	41.06
mpi	**13.93**	17.54	24.41	33.90	64.00

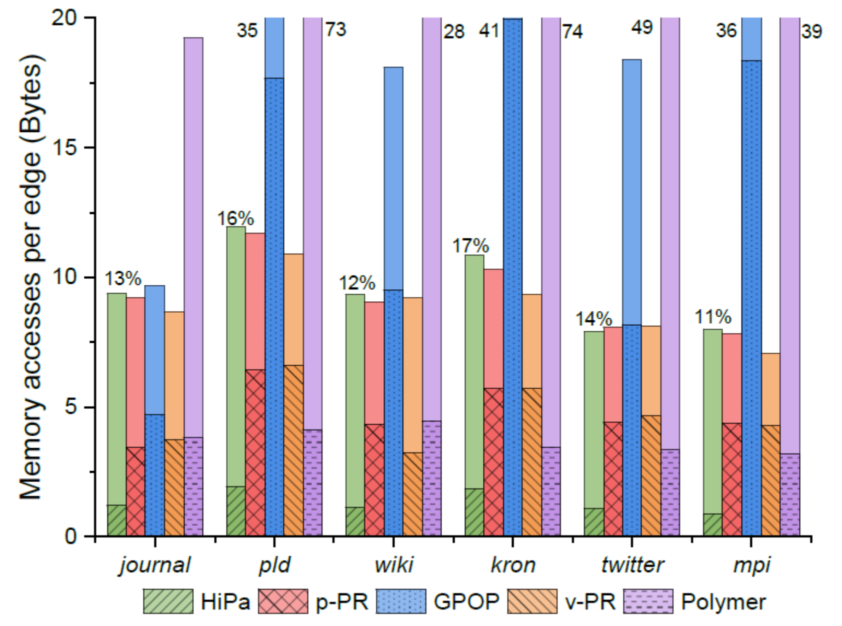
HiPa > others

Hand-coded (HiPa, p-PR, v-PR) > framework-based (GPOP, Polyer)

Partition-centric (p-PR, GPOP) > vertex-centric (v-PR, Polymer)

3 Evaluation

(b) Memory Accesses



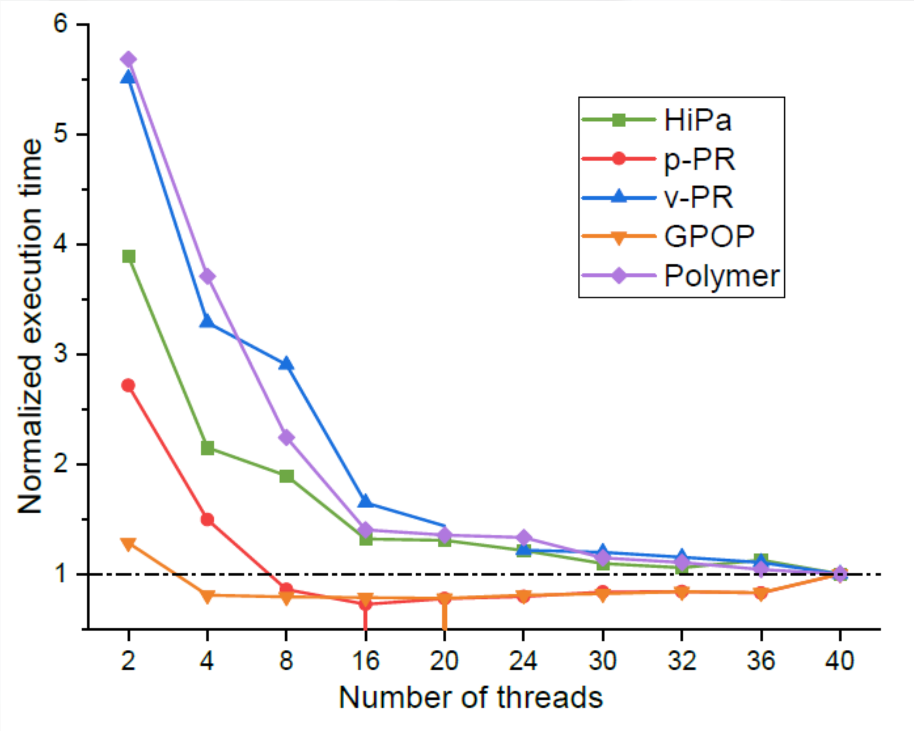
The total bar is the total memory accesses: remote + local memory accesses.

The lower, shadowed bar segment: the remote memory accesses.

HiPa achieves the least remote memory access, which is the key reason for the performance gain

3 Evaluation

(c) Scalability

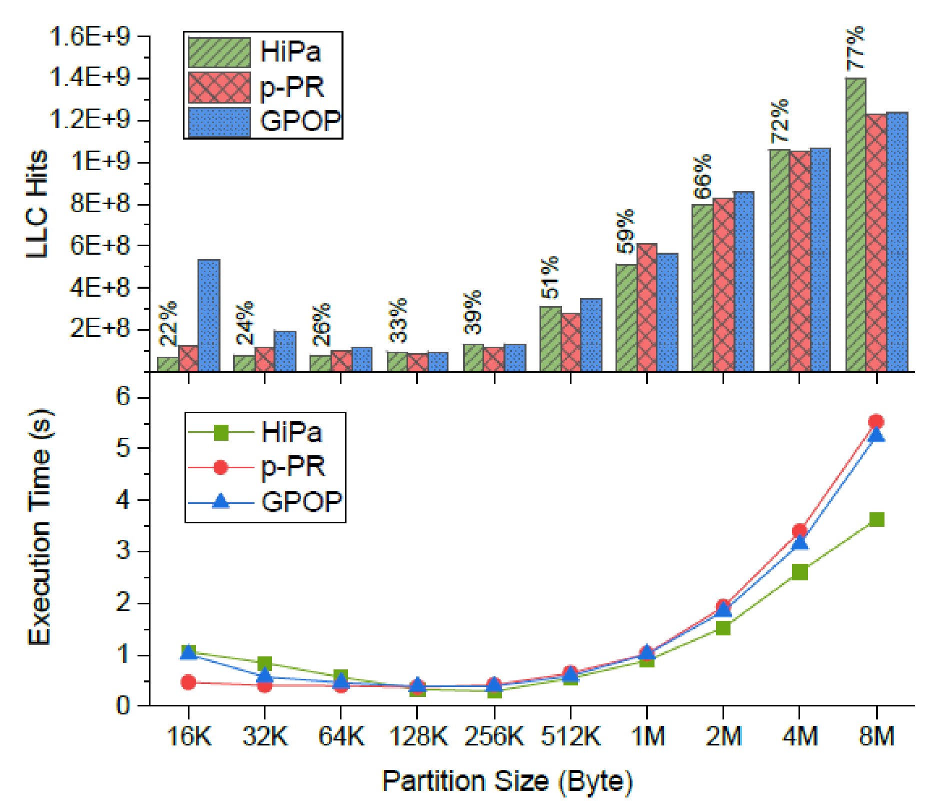


The lowest point means the best performance

- p-PR and GPOP @ 20 threads, and then decay as the #thread grows
- HiPa, v-PR and Polymer @ 40 threads exhibits higher scalability
    - Thread-data pinning of HiPa: thread contention ↓, scalability ↑

3 Evaluation

(d) Sensitivity



The optimal partition size = ¼ * L2 cache size on Skylake = 256KB

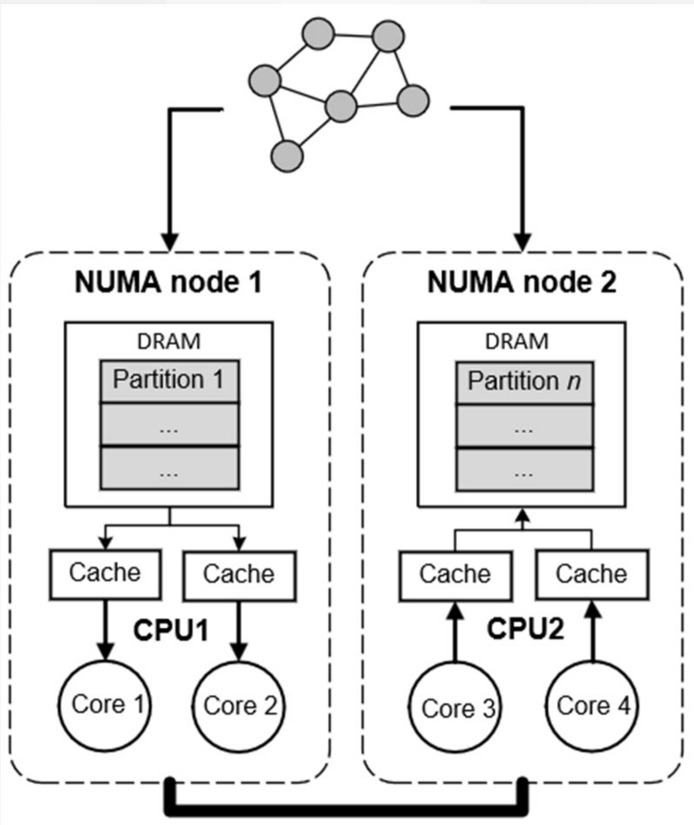
= ½ * L2 cache size on Haswell = 128KB

4 Conclusion

- Key Features

4 Conclusion

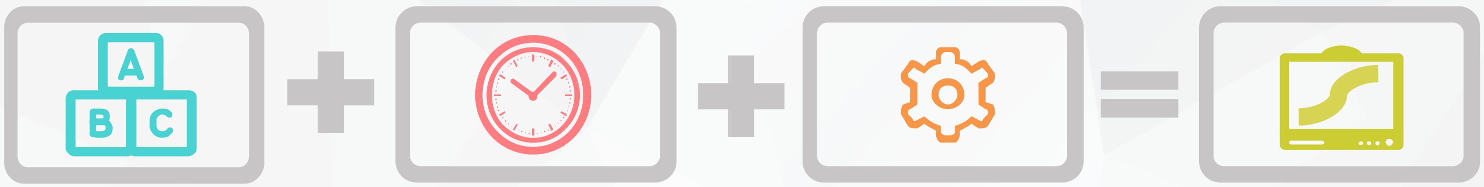
(a) Key Features



- NUMA Partitioning
- Cache Partitioning
- Thread-data binding
- Data Layout

4 Conclusion

(c) Main Achievement



Execution Speedup	Reduced remote memory access	High Scalability	Performance Gain

Thank

you