Toward Programming Models for Parallel Processing of Sparse Data Sets

Padma Raghavan

Computer Science and Engineering
The Pennsylvania State University

Languages and Compilers
for Parallel Computing, 2015

PENNSTATE
1 8 5 5

Research Supported by NSF

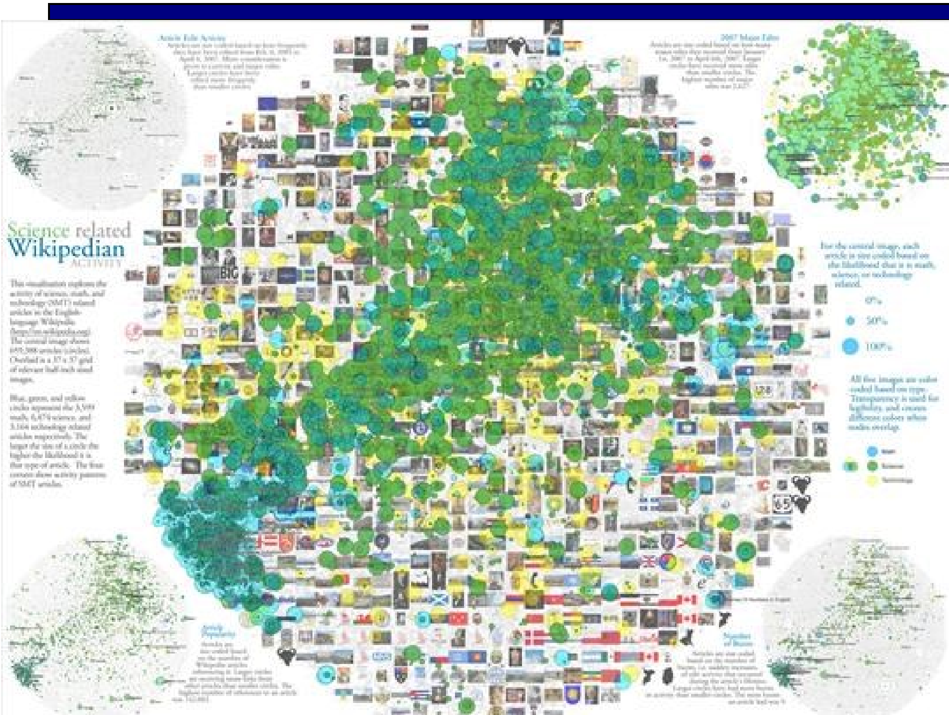
Outline

- Background on sparse data processing
- k-level representations of sparse data and computations
  - achieving high performance on NUMA multicores
  - as an abstraction for domain specific programming models and optimizations
- Looking ahead: the increasing role of compiler technologies

Background

- ➢Where do sparse data come from?
- ➢Why exploit sparsity?
- ➢How to exploit sparsity?

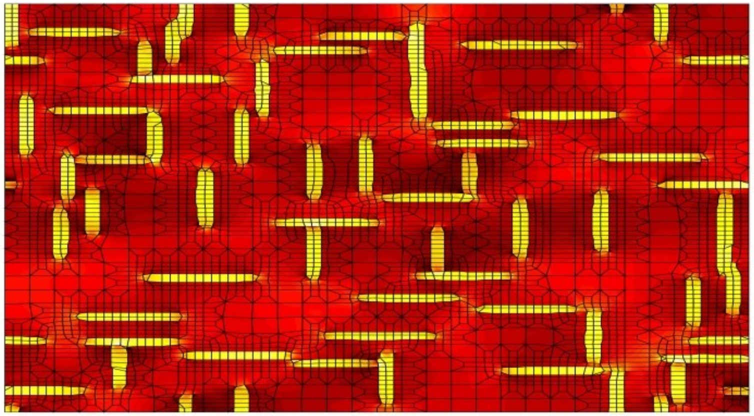
From …. Data, Text and Image mining



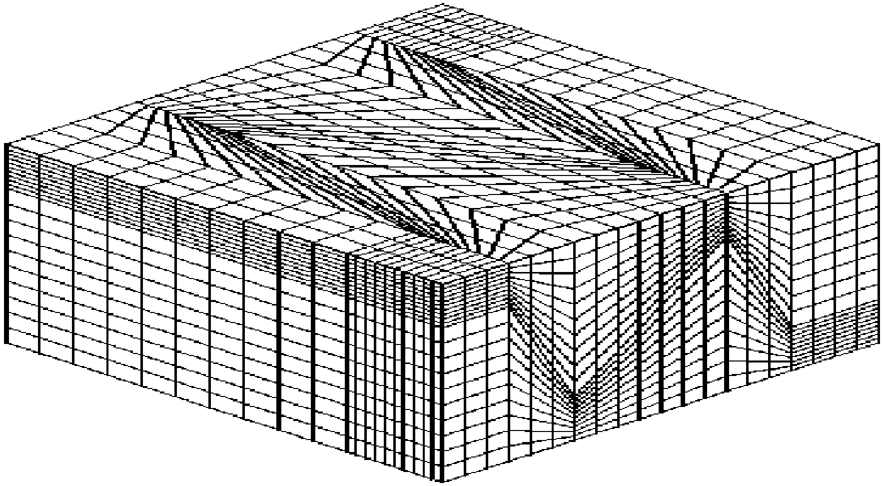
High Dimensional Hyper-Sparse Data

Wikipedia mining
Disease networks
Gene networks (images are from google)

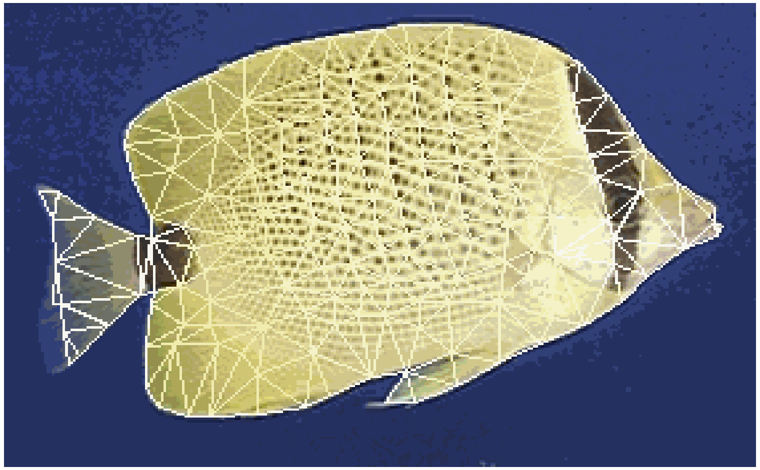
From …. Discretizing space + local interactions to model global transforms



Micron scale: MATCASE@PSU



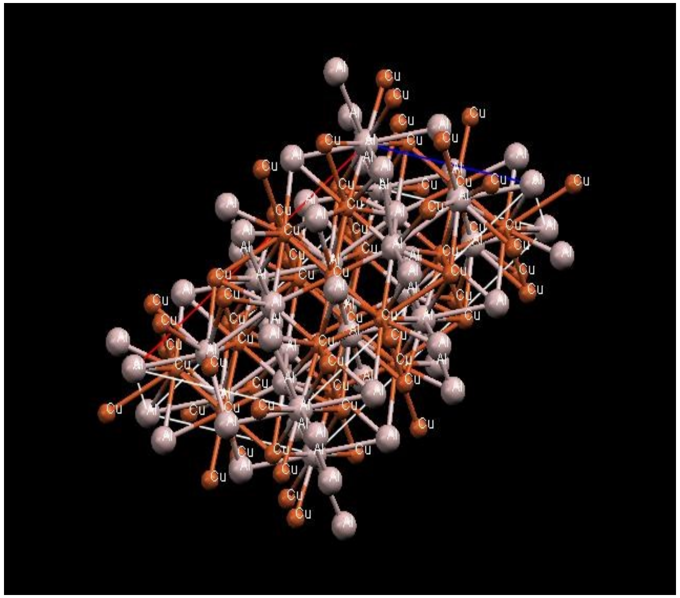
cm^3: Mechanics, Kershaw@ LLNL

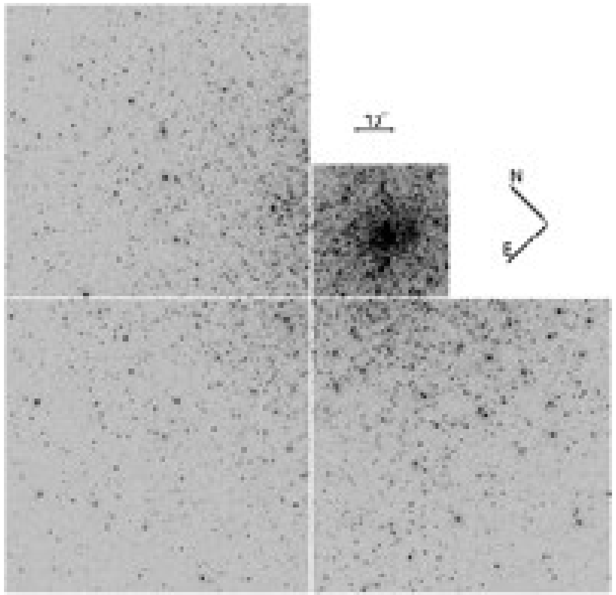


Mesh Based Animation: Tekalp & Osterman@U. Rochester

From Approximations

- Approximate N-body interactions

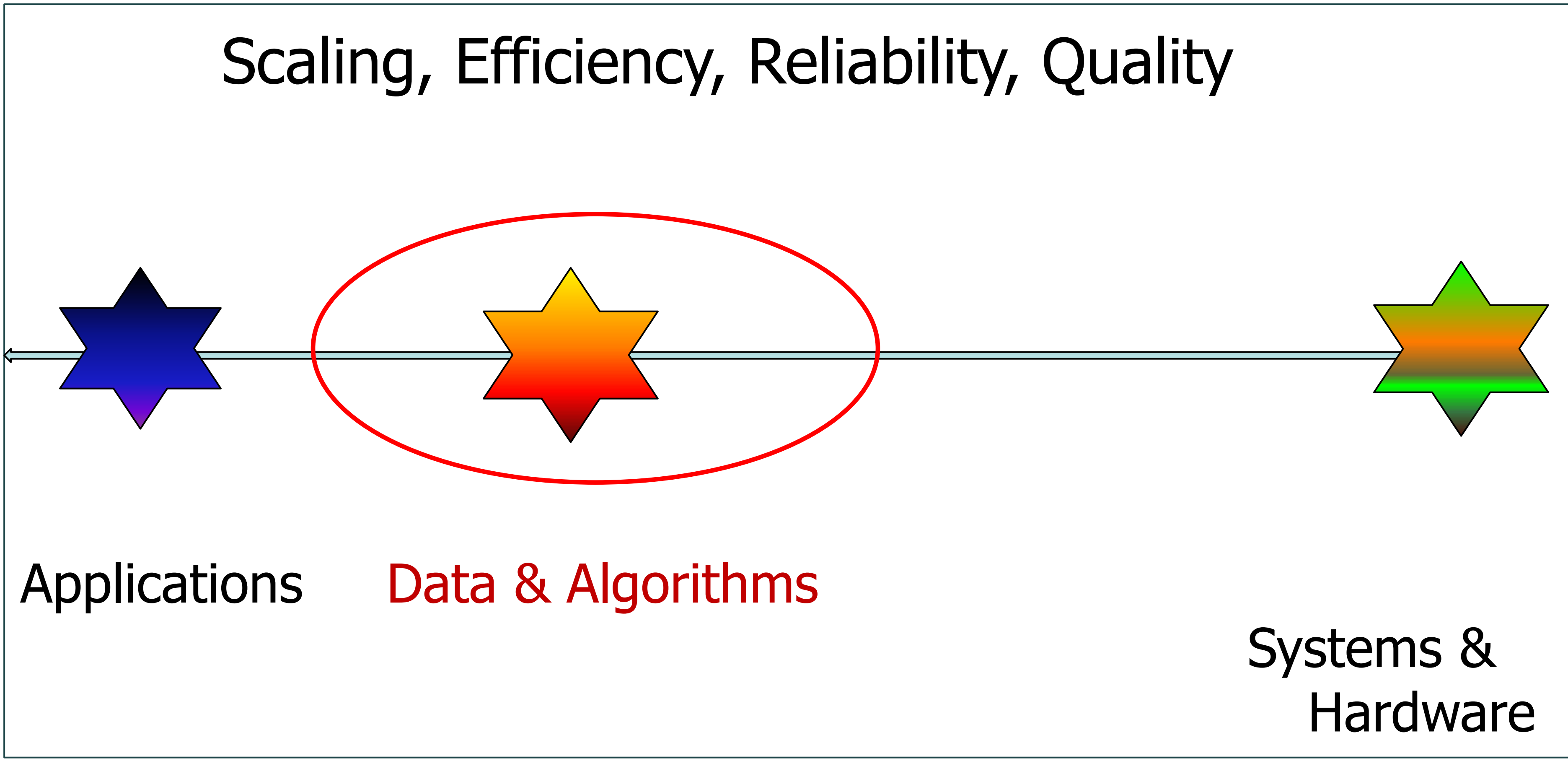




Atomistic: MATCASE@PSU

Astrophysics: P. Hut et al, Hubble Telescope

Performance Challenges



Why exploit Sparsity?

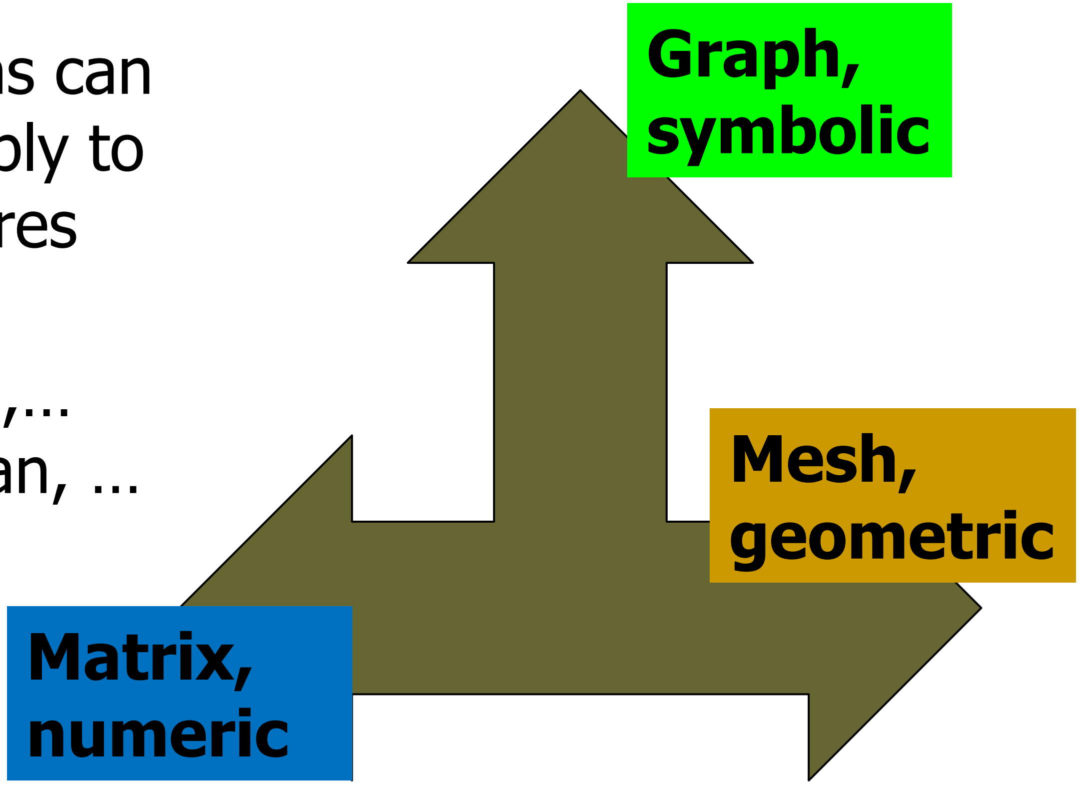
- Sparsity= Enables "Big Data" and "Big Simulation" based science that is otherwise beyond reach

- Opportunity=Order(s) of magnitude reduction in Memory and Computational costs

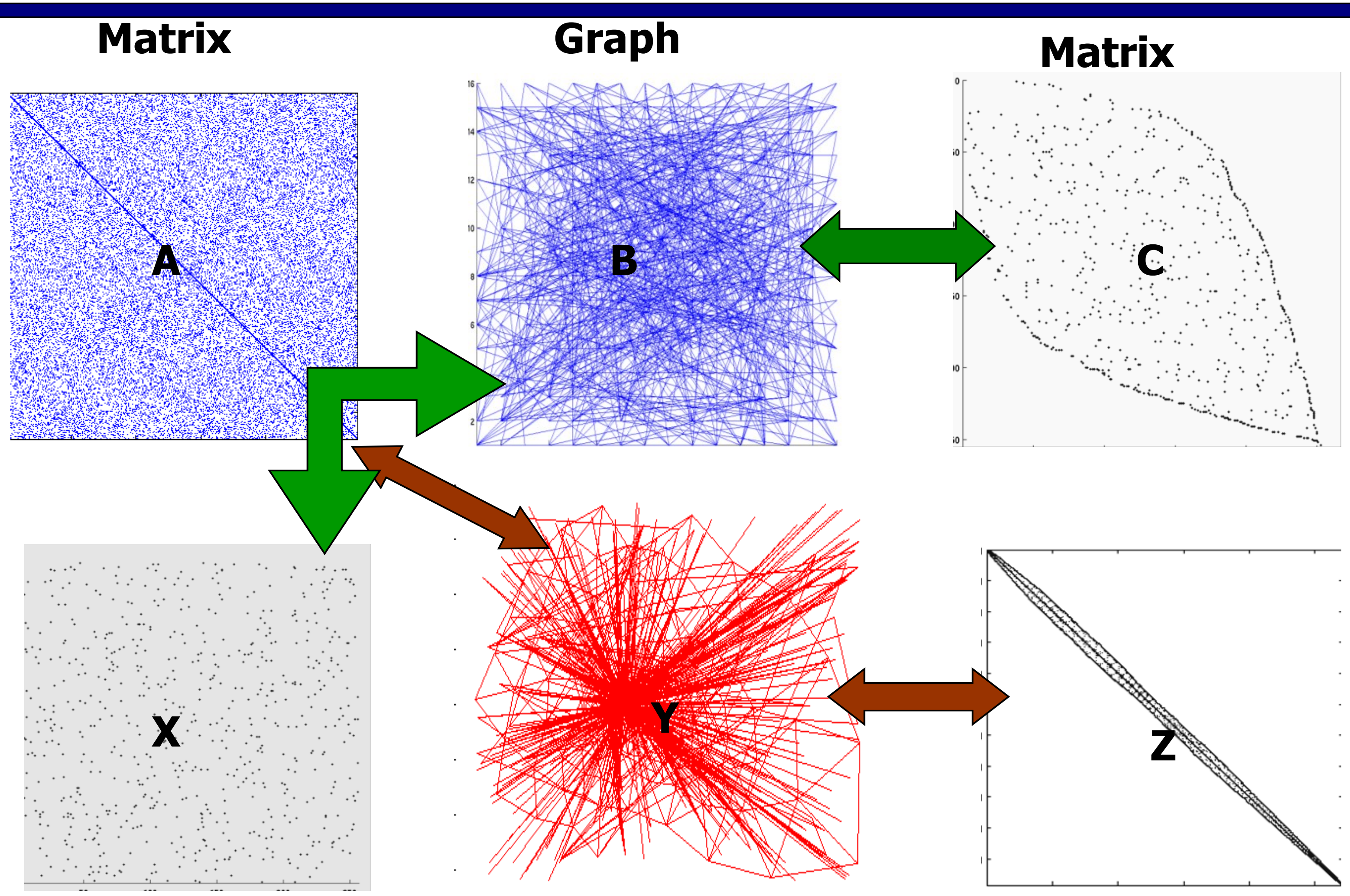
- Increasingly important – as shown by recent development of Sparse HPCG Type-2 benchmarks vs Dense Linpack Type-1

What are general approaches ?

- Identify structure that is latent in sparse data
- Multiple representations can be used interchangeably to find useful sub-structures
- Contributions by many Catalyurek, … Demmel,… Gilbert, … Ng, Raghavan, … Ucar, Yelick …

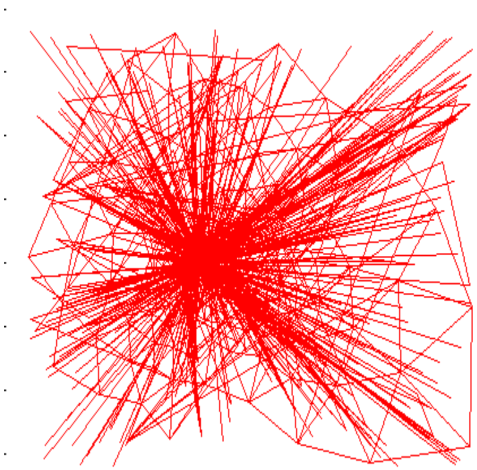


Match: Matrix to Graph to Matrix ?

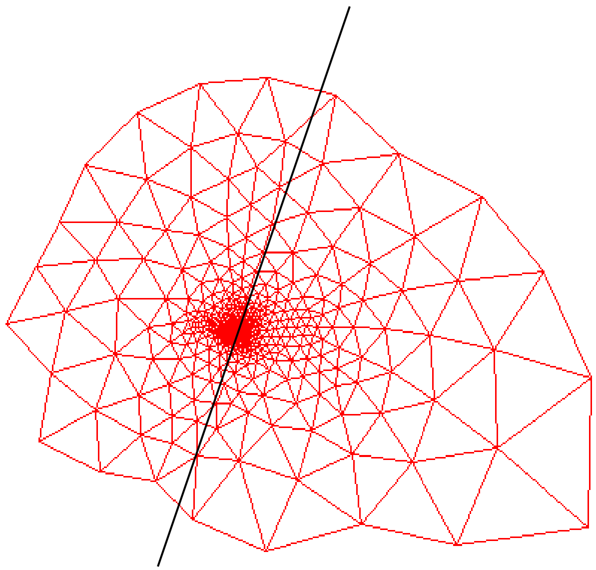


Why? Geometry Counts!

- Graph maps to clean geometric embedding
  =Planar =Separable!

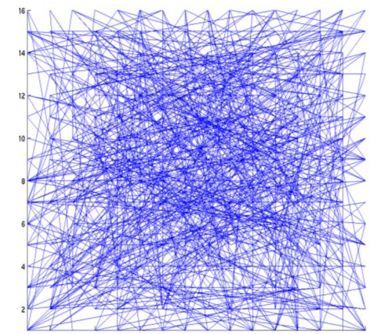


=



Geometric

- N vertices= 2 halves + $\sqrt{N}$ separator
- Elegant theory



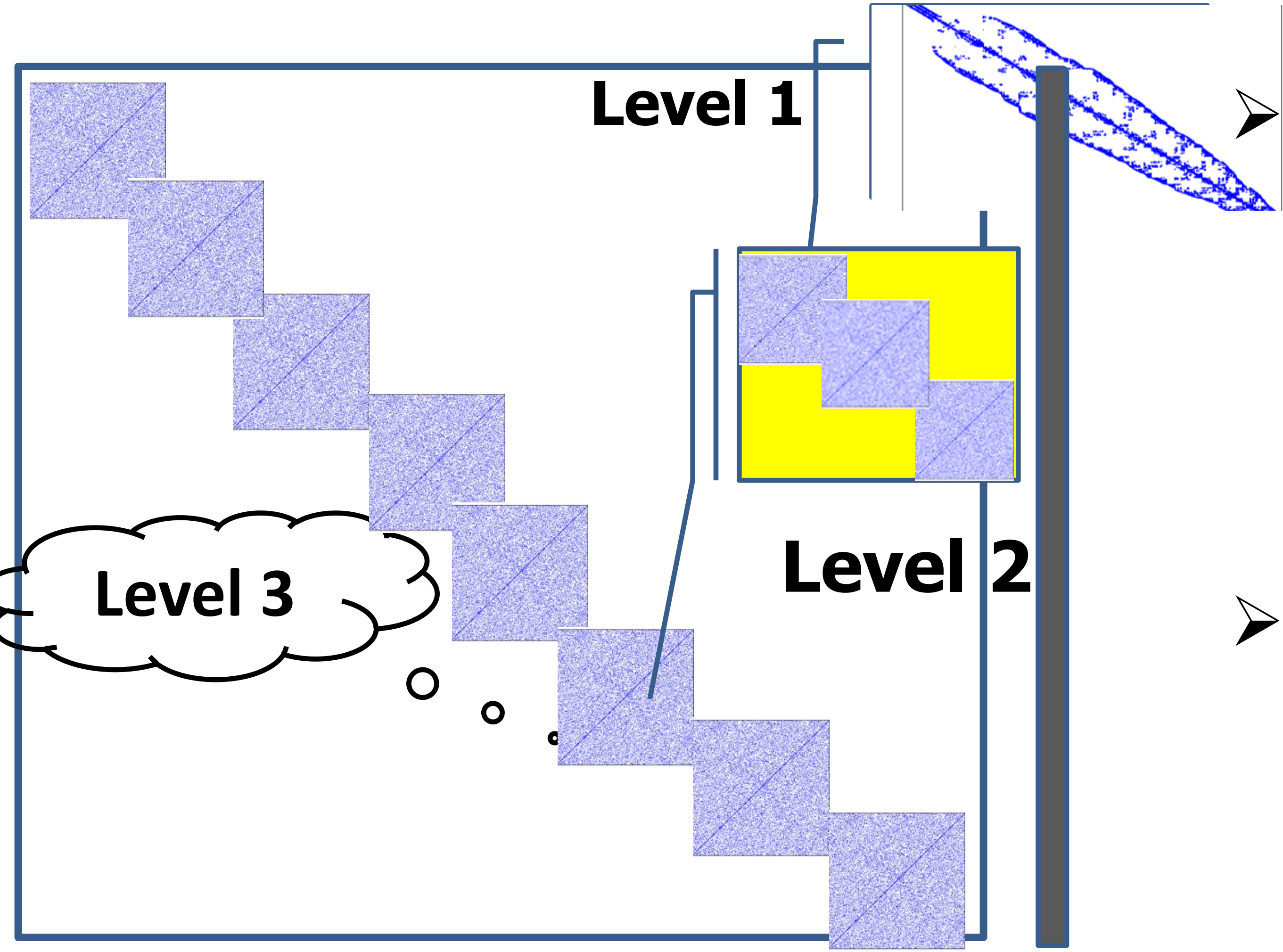
- Not separable even when sparser
- Almost all random matrices are not separable (Szermedi)

Some abstractions for High Performance on NUMA Multicores

- Large packs of independent equal-length tasks
  - Multilevel data sub-structuring
    - Data reuse aware task scheduling

- Sparse matrix examples ---approach extends to graphs and meshes

Multilevel Sub-structuring of Data



- Spatial locality in matrix or graph
- Temporal locality in vector (reuse)

CSR-k Sparse Mat-Vec

- In parallel for each super-super-row l = 1….r
  - for each super-row j in super-row l

for each row k in super-row j

y(i) = 0

for each nonzero in row i

Load subscript k, A[i][k], x(k)

y(i) = y(i) + A[i][k] *x(k)

end for

……………………………

Represents higher levels beyond traditional CSR

Speeding-up Sparse Computations on Multicores: What techniques are known to work?

- Ordering of matrix affects data locality & reuse in **x**
  - Profile reducing orderings (e.g. RCM) are generally good for mat-vec
  - Level set and coloring are generally good for triangular solve

- Utilizing dense sub-blocks to reduce loading of nonzero subscripts can help
  - Tradeoffs between # of loads & # of operation
  - Dense blocks can be artificially created by adding fill or dense blocks that exist naturally can be exploited

CSR-k: A multilevel form of CSR
Example: for K=3, symmetric A

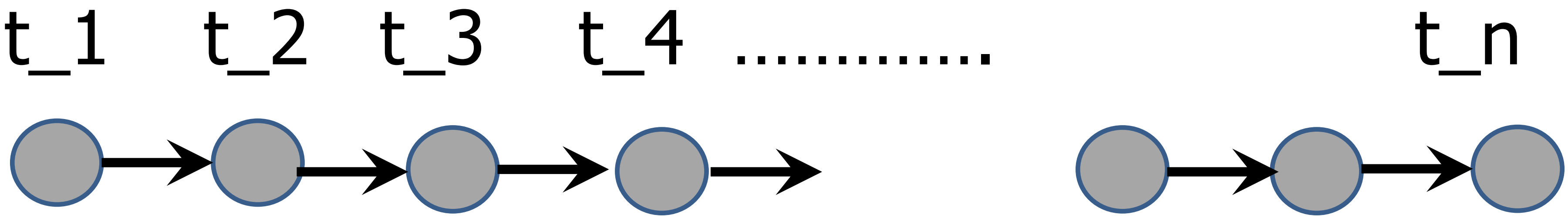
- Start with A1 =A and G1= graph of A1
- Coarsen G1 to get G2 (with super-rows); Order G2
- Coarsen G2 to get G3 (with super-super-rows); Order G3
- Expand G3 to G2; refine ordering in each super-super-row
- Expand G2 to G1; refine ordering in each super-row

- <u>Motivation:</u> To get packs of uniform length, independent tasks at a desired granularity, with spatial locality in A, and options for temporal locality/reuse in x through scheduling

CSR-k + Scheduling: 2 examples

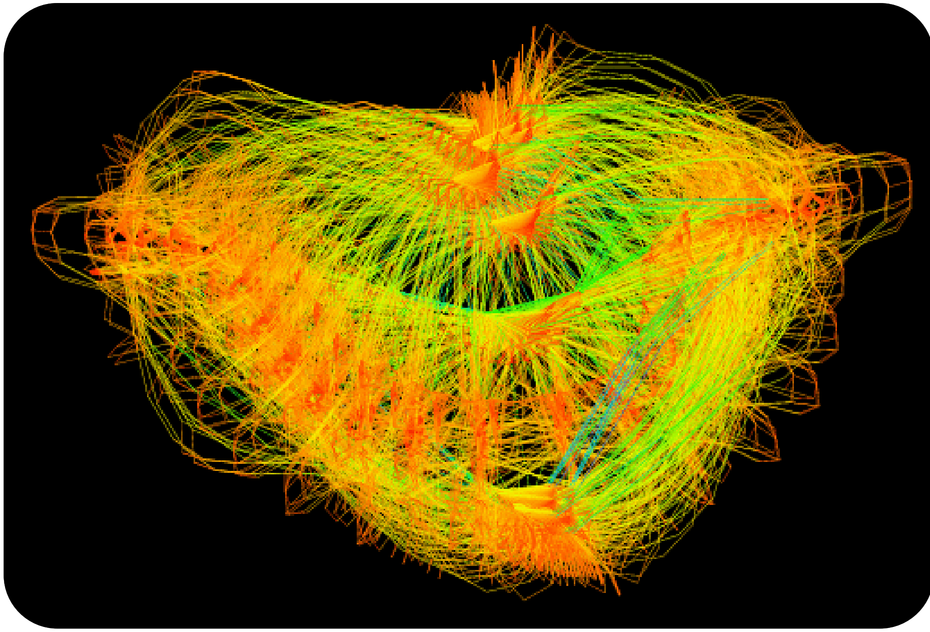
- Mat-Vec (Ax=y)
  - Coarsen: heavy edge matching or consecutive rows of a band ordering
  - Ordering of G2, G3 : a weighted form of band ordering
  - Published, HiPC 2014, Kabir, Booth, R.
- Tri- Solve (Ly = b)
  - Coarsen : same as above
  - Ordering of G2, G3: Coloring (serialization is removed)
  - To appear, SC15, Kabir, Booth, Aupy, Benoit, Robert, R.
- <u>Data affinity and reuse graph model of scheduling:</u> To utilize temporal locality in vector and promote reuse

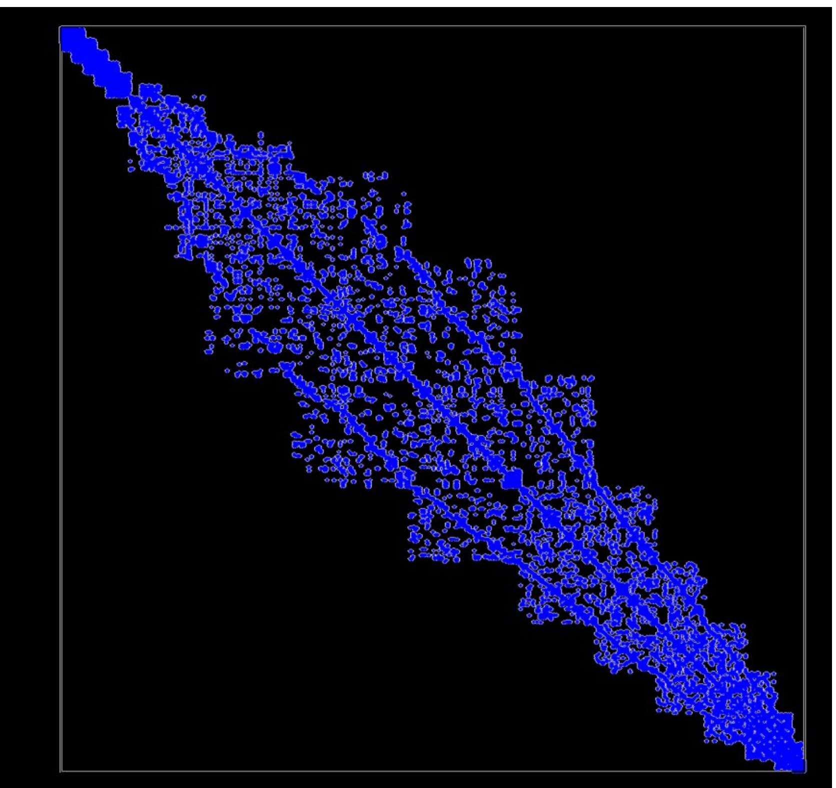
Data affinity and Reuse Graph for Scheduling



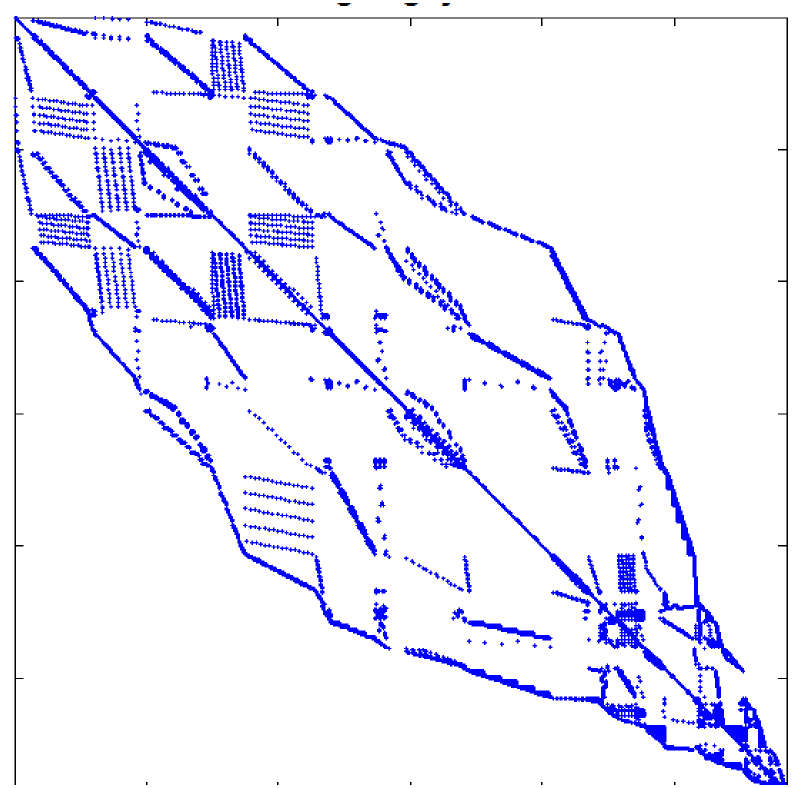
- n uniform length tasks, n is large
- Simplified model: edge between t_i and t_(i+1), i= 1, ..., n-1 if vector elements are shared

- If cores are identical, equal partition is optimal schedule (Aupy, Benoit, Robert)
- Actual graphs are not chains but we use ordering so that the "chain" reflects main reuse component

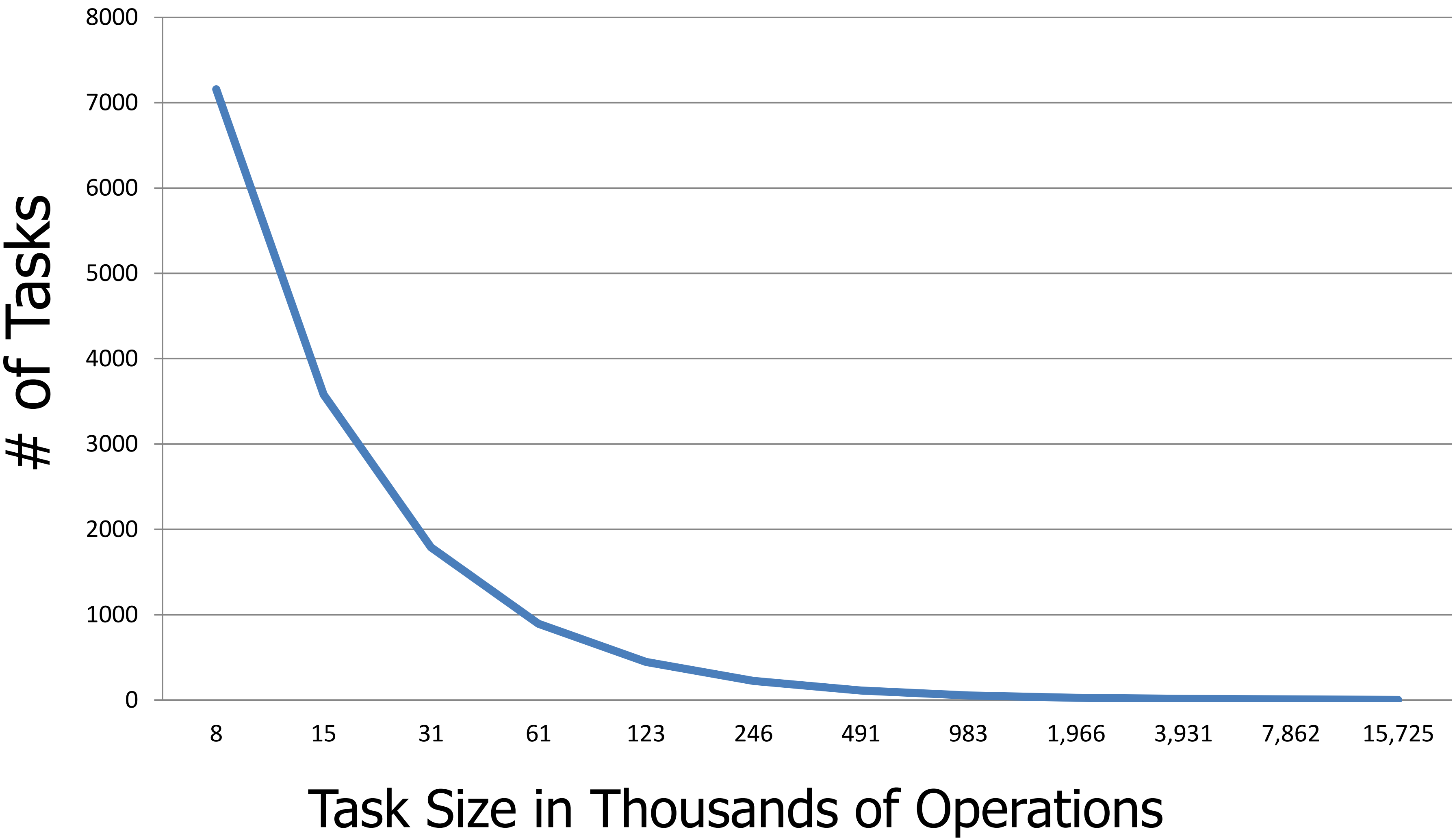




CSR-3: Tri-Solve



Tunable Degree of Parallelism:
Tasks vs Task Granularity



CSR-K:

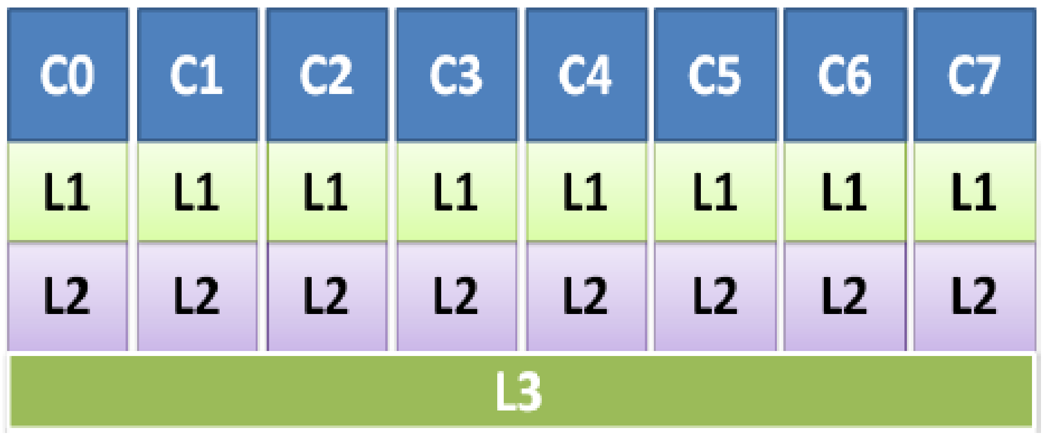
➢ How does it perform?

- CSR-2 Sparse Mat Vec (HiPC 2014, Kabir, Booth, R.)
- CSR-3 Tri-Solve (SC 2015, Kabir, Booth, Aupy, Benoit, Robert, R.)

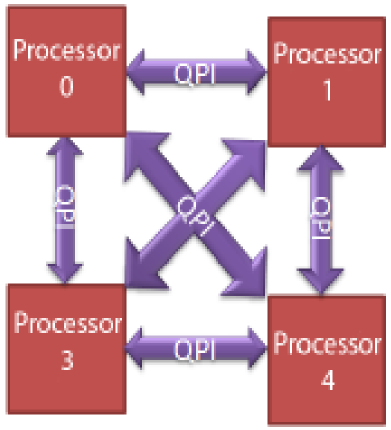
Sparse Matrix Suite for Tests

Matrix	# of Rows	# of Nonzeroes	Row Density
G1: ldoor	952,203	42,493,817	44.63
D1: rgg_n_2_21_s0	2,097,152	31,073,142	14.82
S1: nlpkkt160	8,345,600	225,422,112	27.01
D2: delaunay_n23	8,388,608	58,720,176	7.00
D3: road_central	14,081,816	47,948,642	3.41
D4: hugetrace-20	16,002,413	64,000,039	4.00
D5: delaunay_n24	16,777,216	117,440,418	7.00
D6: hugebubbles-0	18,318,143	73,258,305	4.00
D7: hugebubbles-10	19,458,087	77,817,615	4.00
D8: hugebubbles-20	21,198,119	84,778,477	4.00
D9: road_usa	23,947,347	81,655,971	3.41
D10: europe_osm	50,912,018	159,021,338	3.12

Intel Westmere NUMA



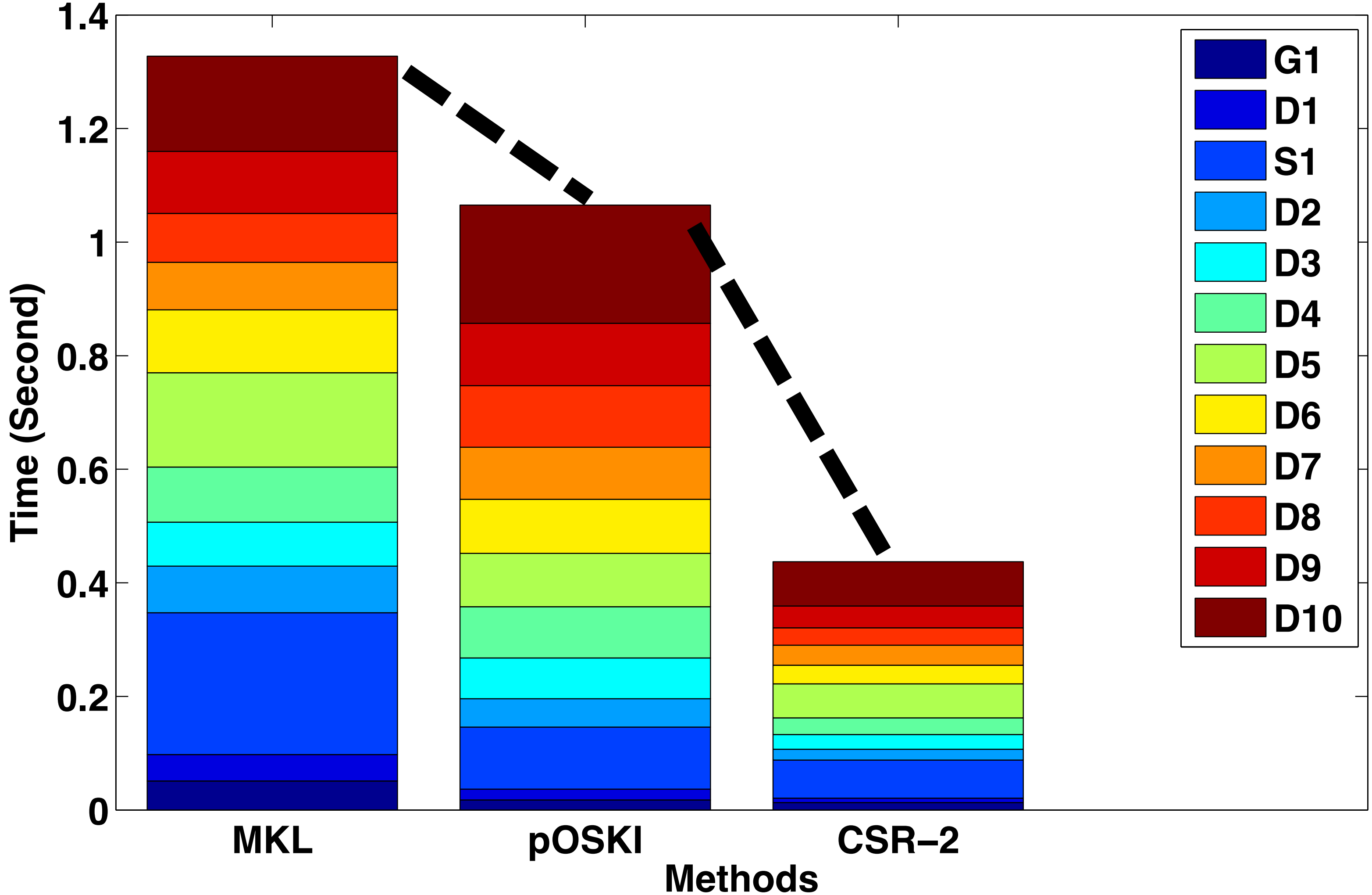
(a) 8-core processor.



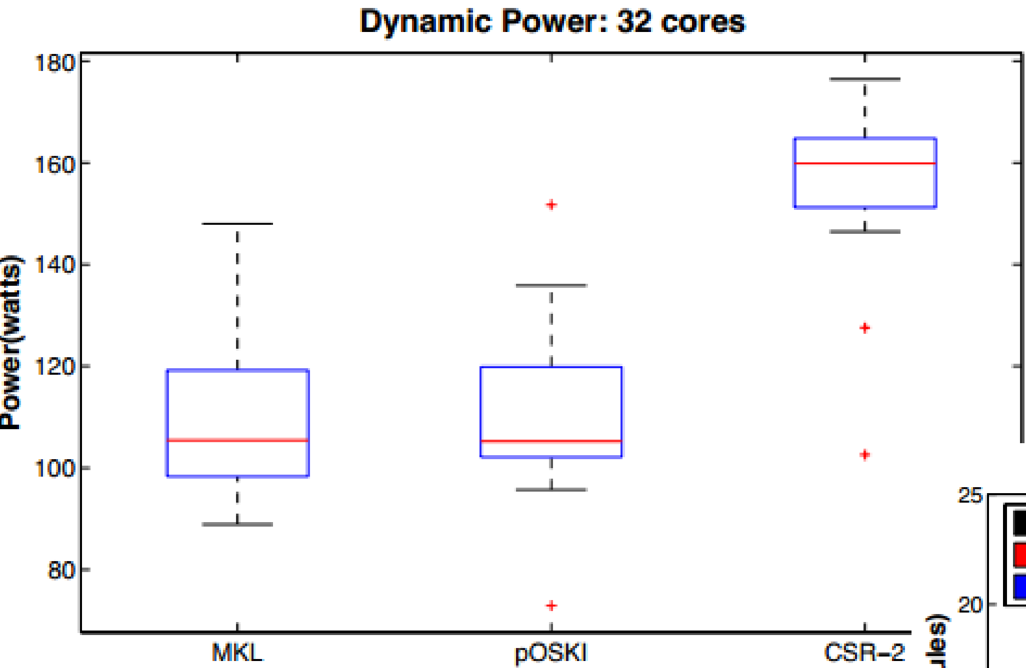
(b) 4-processors configuration with QPI.

- L1 access: **4** cycles (private)
- L2 access:**10** cycles (private)
- L3 access: **38-170** cycles (shared)
- Memory access: **175– 290** cycles (shared)

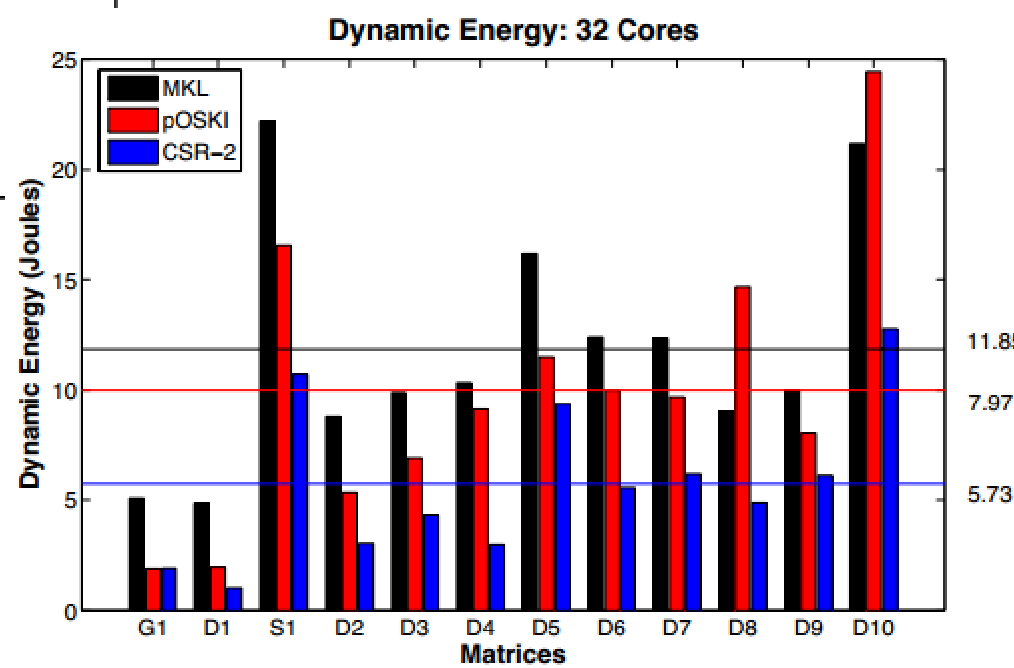
Mat-Vec: MKL, pOSKI, and CSR-2 on 32 cores



Dynamic Power and Energy: 32 cores

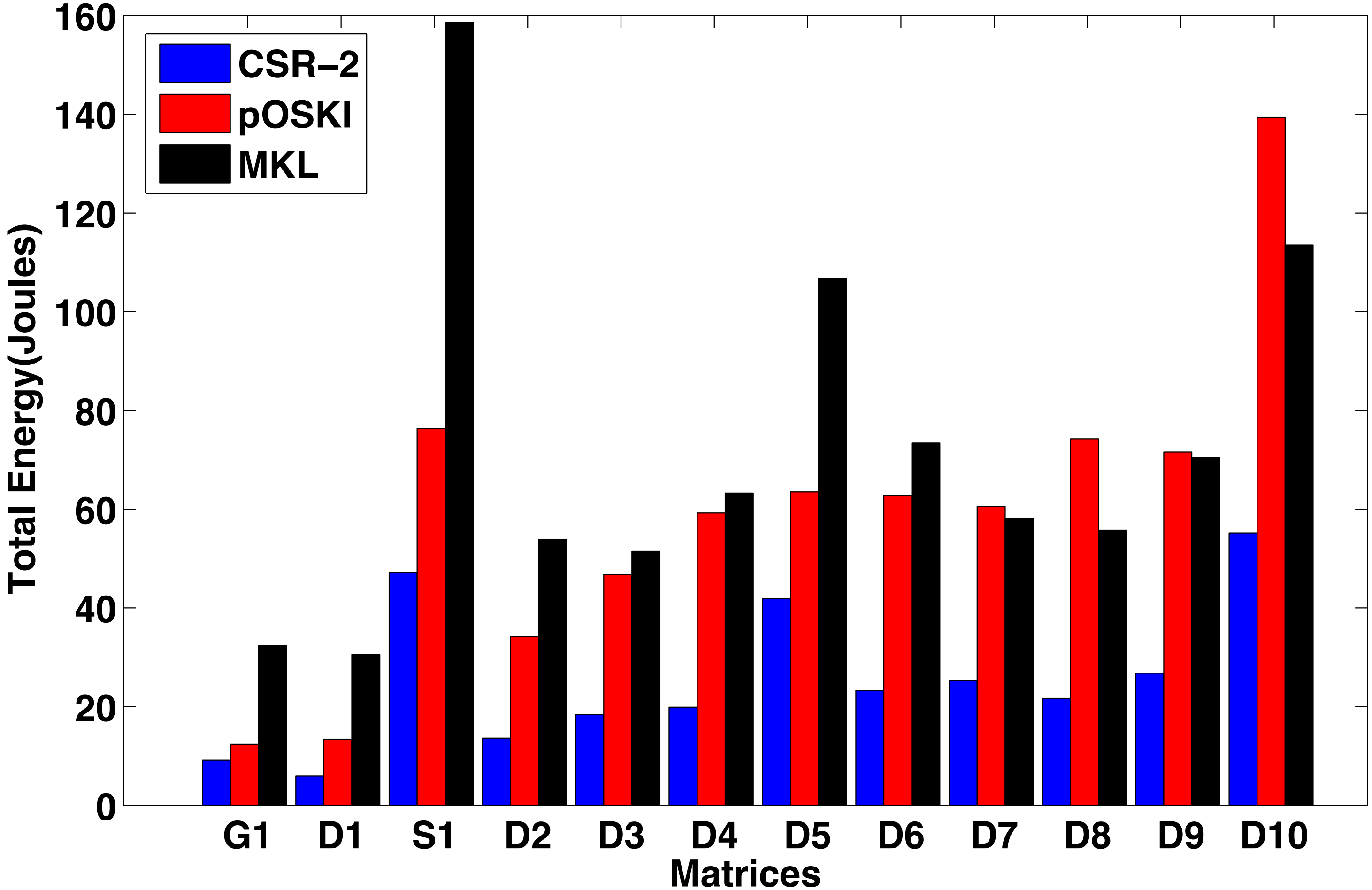


- CSR-2: Highest Dynamic Power !!



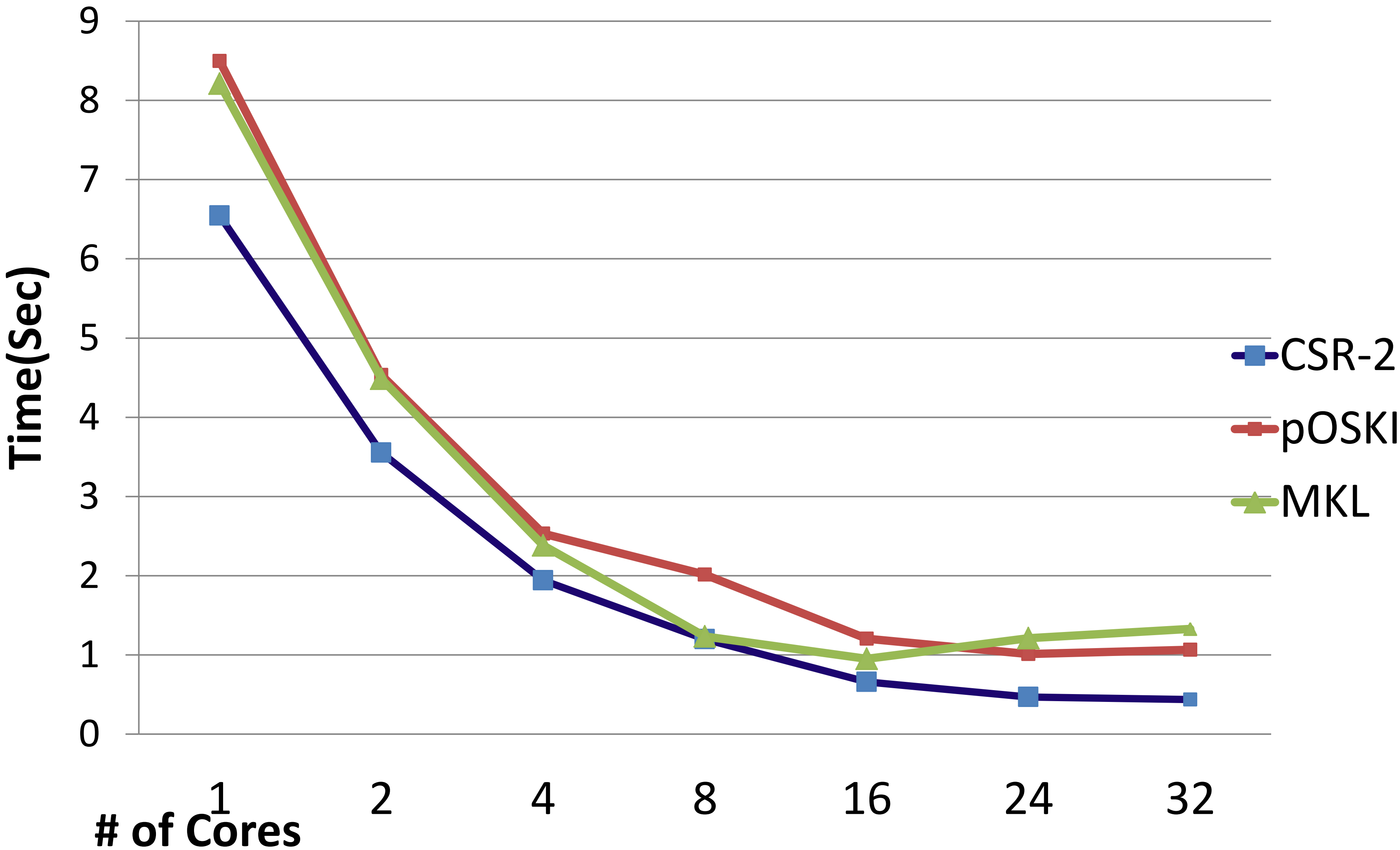
- Lowest Dynamic Energy;
- High H/W Power leads to faster execution

Total Energy: Static + Dynamic 32 cores

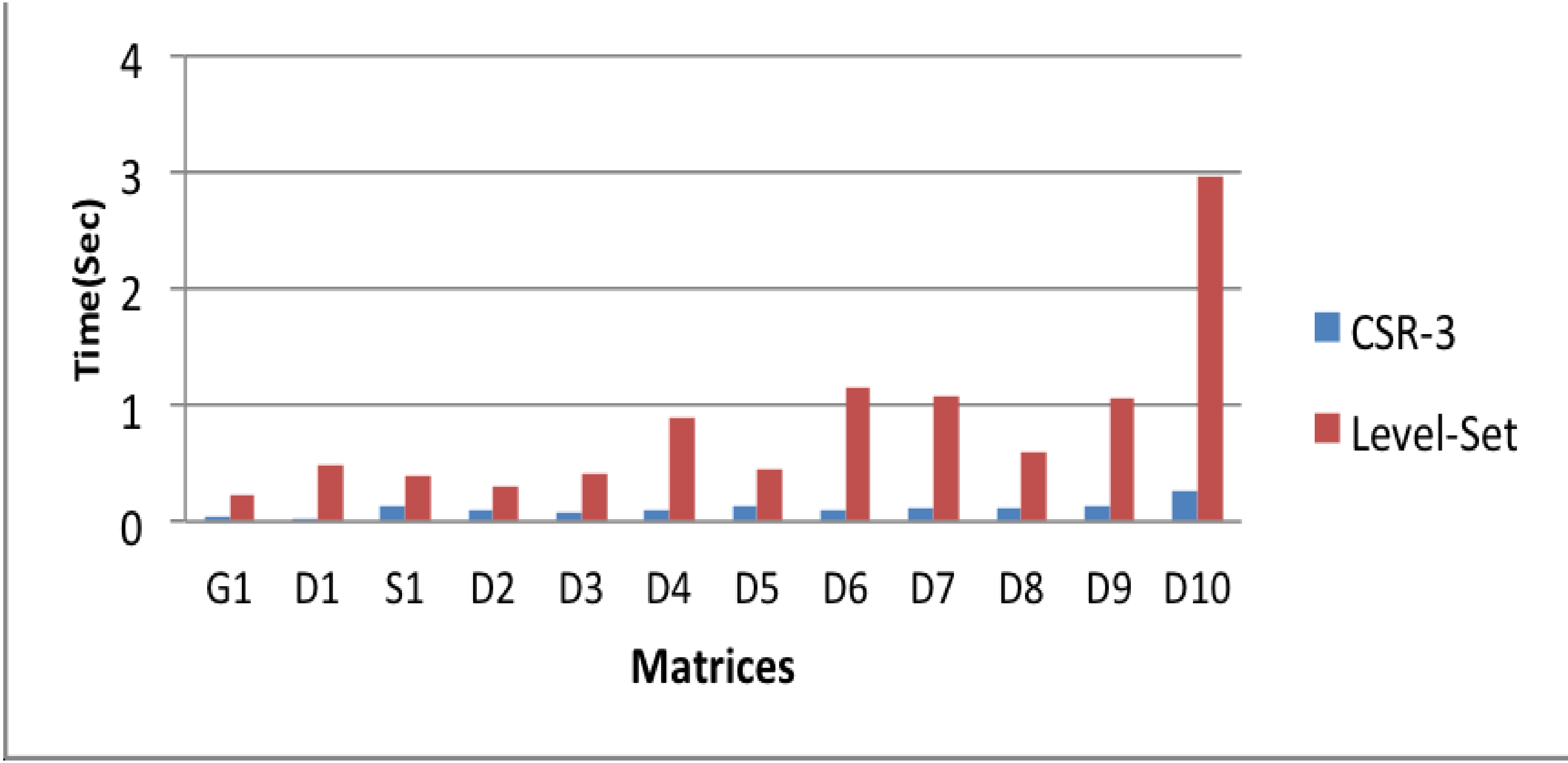


- CSR-2 uses only 44% of energy of pOSKI
- MKL uses the most energy

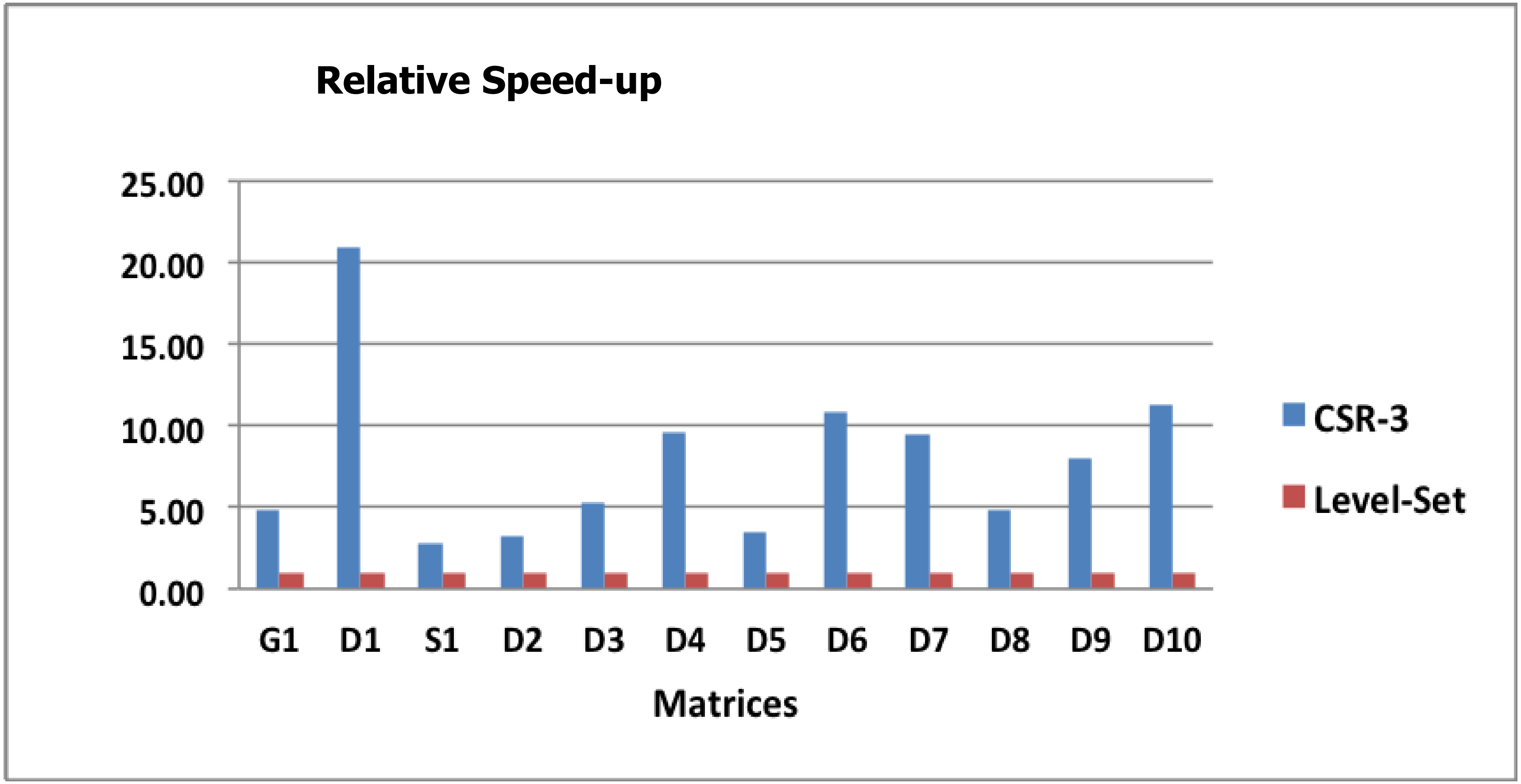
Mat-Vec: MKL, pOSKI and CSR-2: Total Time Over All Matrices, 1-32 Cores



Tri-Solve : Level- Set and CSR-3 with coloring, 32 cores

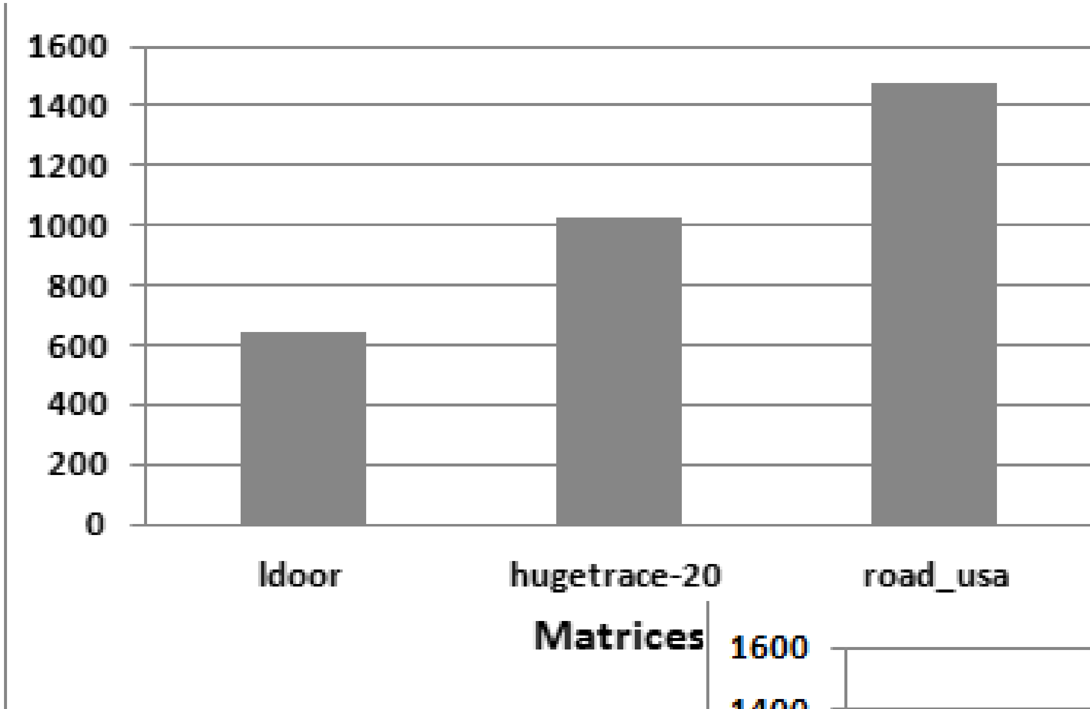


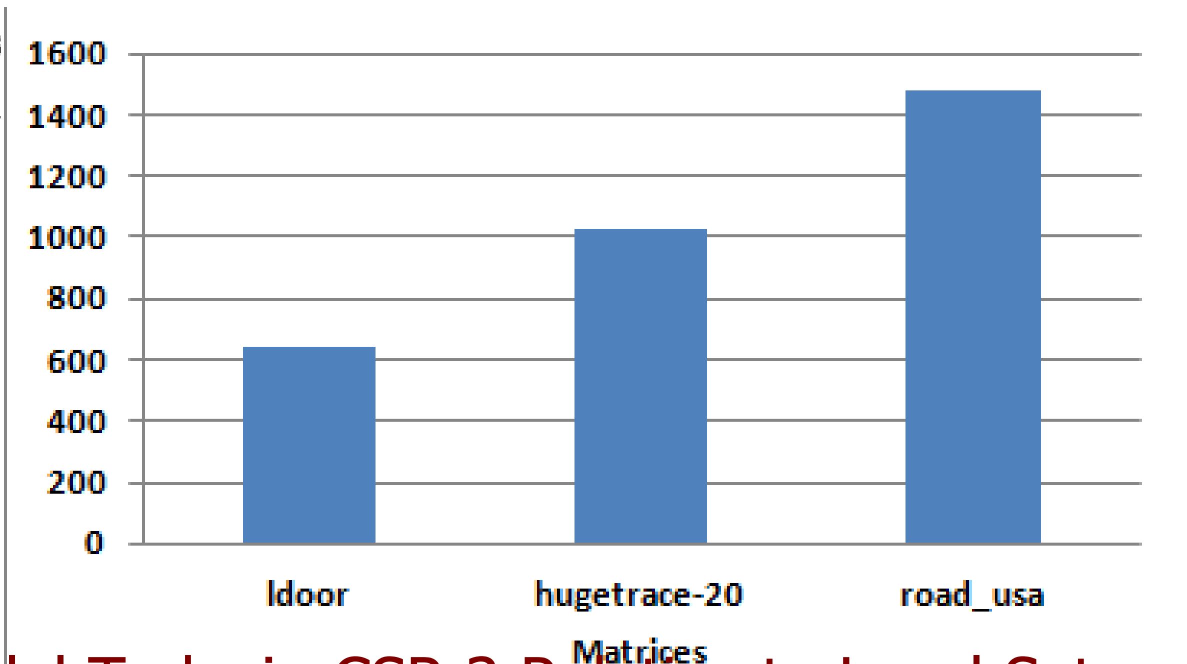
Tri-Solve: Speed-ups Relative to Level Set



- ➢ CSR-3 is 20x faster than level-set

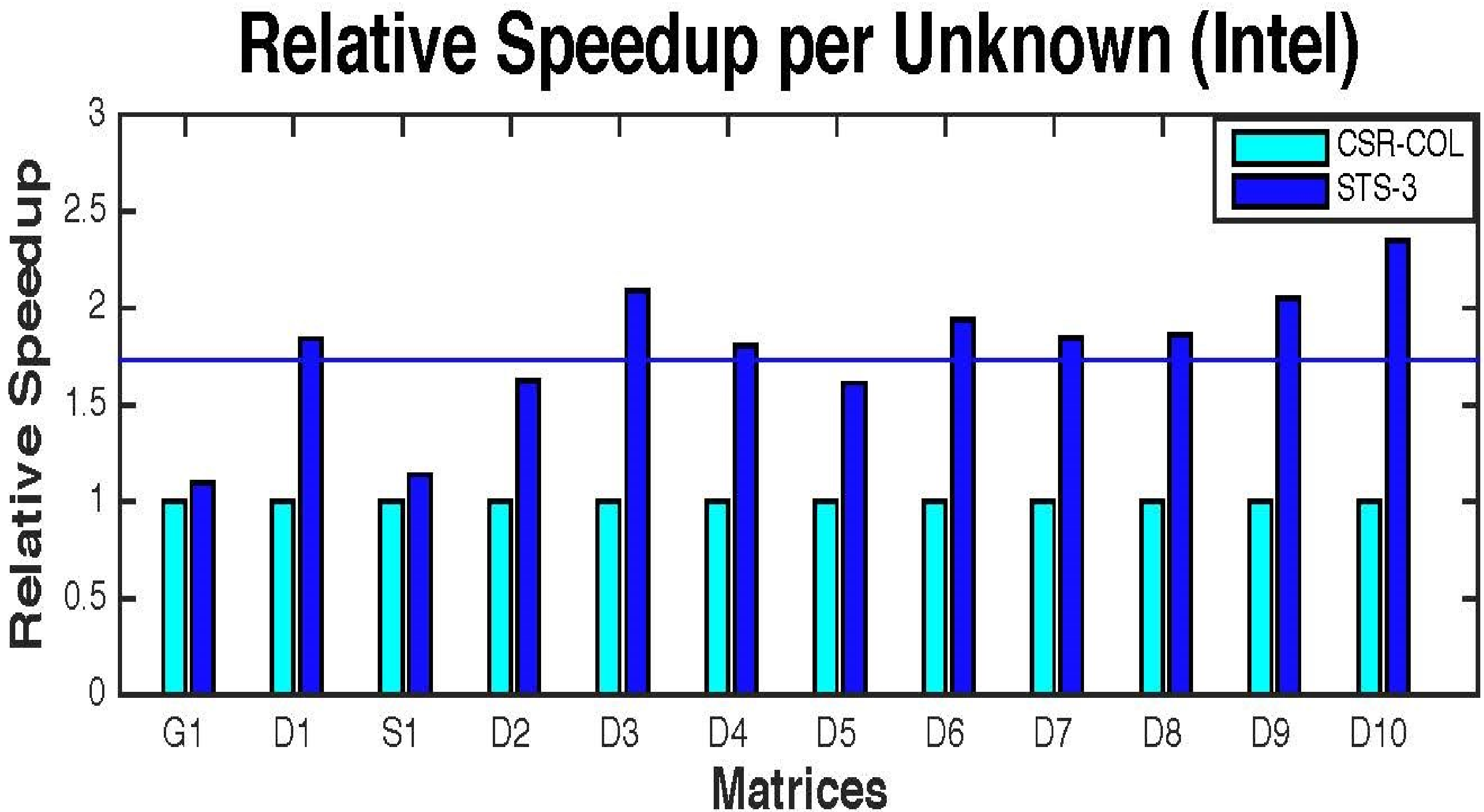
Serial Steps in Level Set Relative to CSR-3 with coloring





Average Number of Parallel Tasks in CSR-3 Relative to Level Set

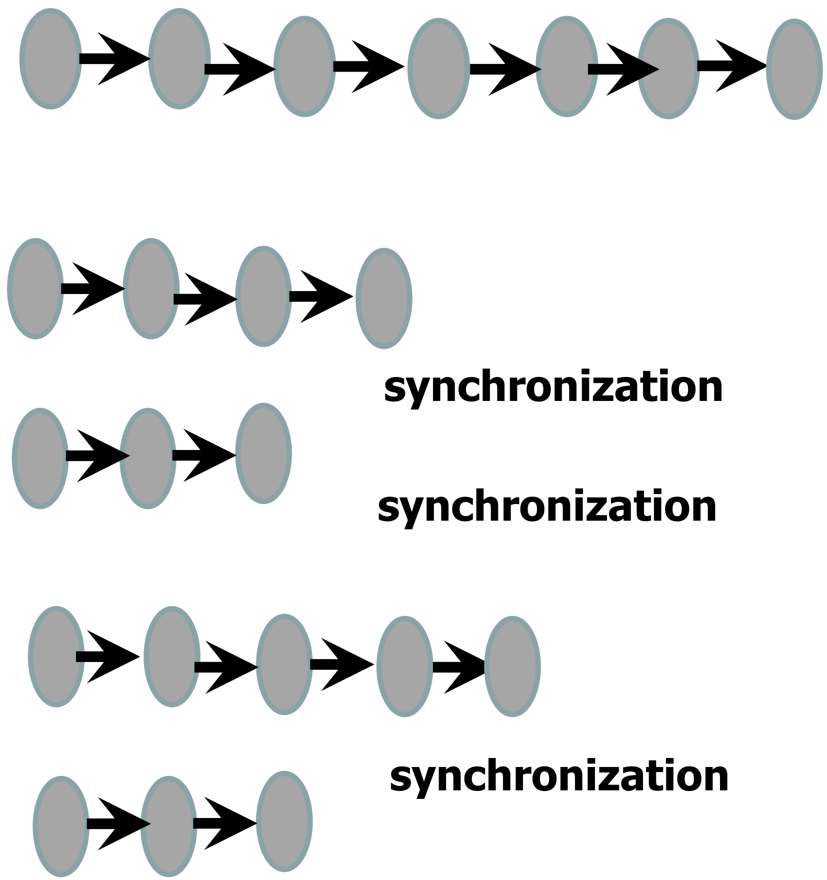
Tri-Solve : Effect of Reuse Aware Scheduling within 1 pack



CSR-k: A simple model for performance tuning

- Large packs of equal length independent tasks through ordering and sub-structuring
  - can tune granularity using k
  - enables spatial locality of accesses in matrix
- Data affinity/reuse graph model for scheduling can increase temporal locality in accesses to vector
- For graphs and meshes, there are k-level counterparts that are similar to csr-k for matrices
- Algorithm based fault tolerance can be wrapped in

CSR-k Patterns for Domain Specific Optimizations?



(a) Data affinity/reuse chain graph of independent equal length tasks (mat-vec, on multicore)

(b) Packs of (a) with synchronization between packs (tri-solve or bfs on multicore)

(c) Packs of (a) with partial synchronization between packs, e.g. tree-like, for cross multicore nodes, accelerators

The need for domain specific optimizations

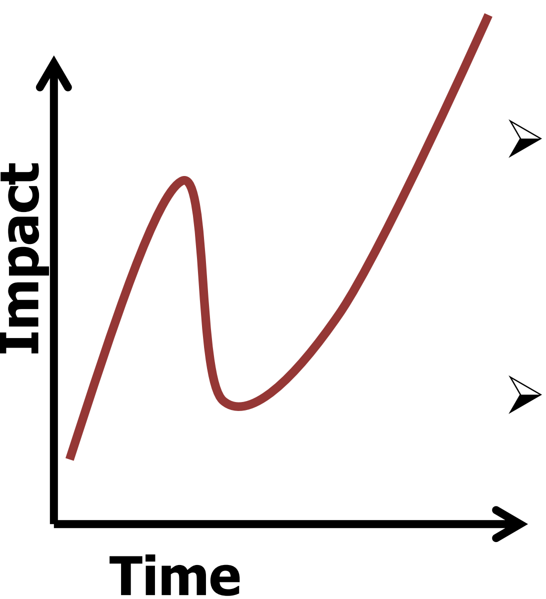
- CSR-k is a simplified abstraction for high performance sparse computations on NUMA multicores
- Needs a domain language/domain specific optimization approach to be widely usable and adaptable to new hardware
- Run-time optimizations and dynamic approaches to scheduling will be needed in practical applications to leverage tunable parameters of CSR-k and to variations in hardware

Looking ahead

- Parallel computing for all --- programming language and compilation technologies can be the driver!

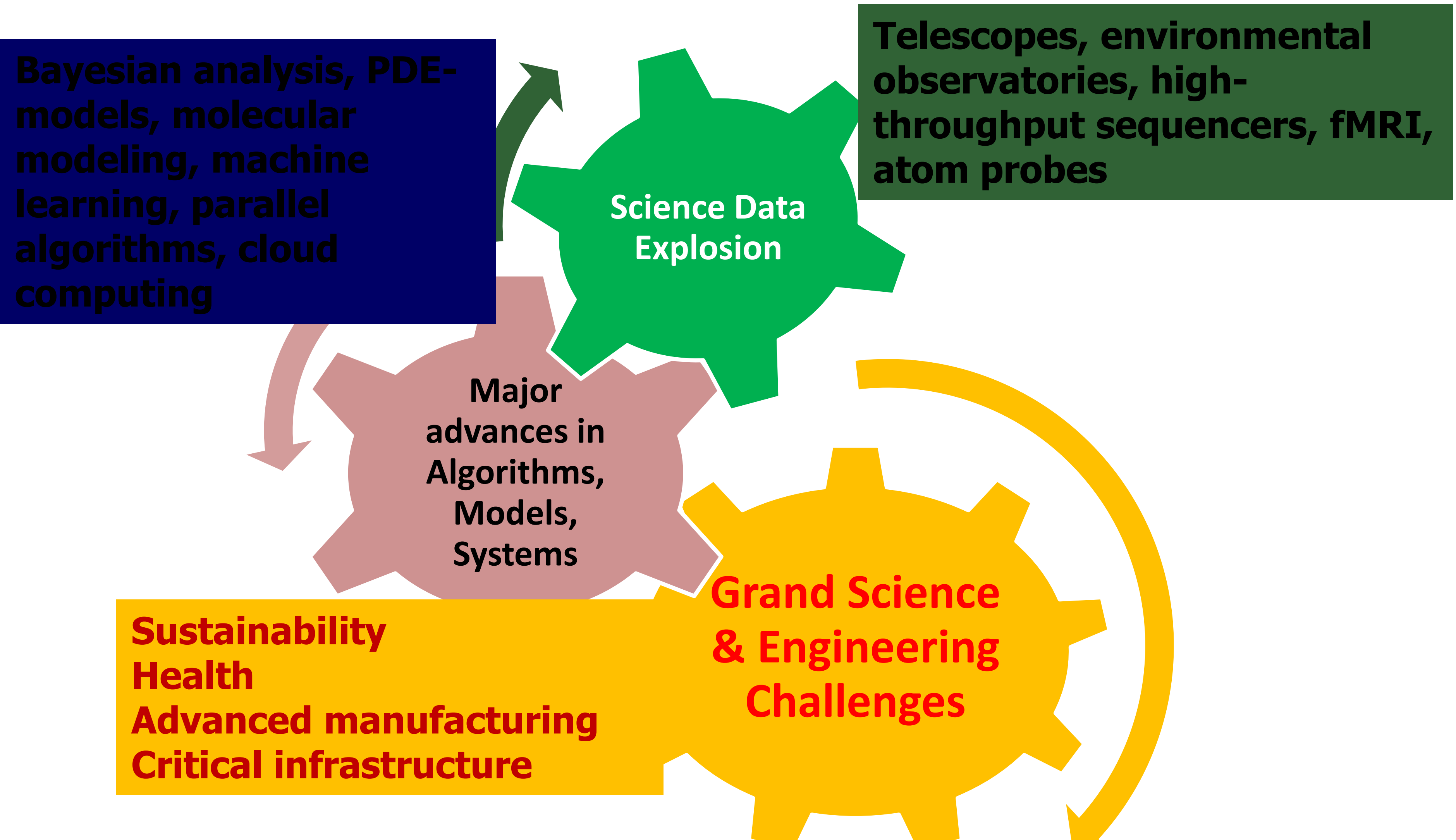
Parallel Computing: A Second Sustainable Surge?

"A truly transformational technology will always have its immediate consequences overestimated and its long-term consequences underestimated" *Francis Collins, 2010*



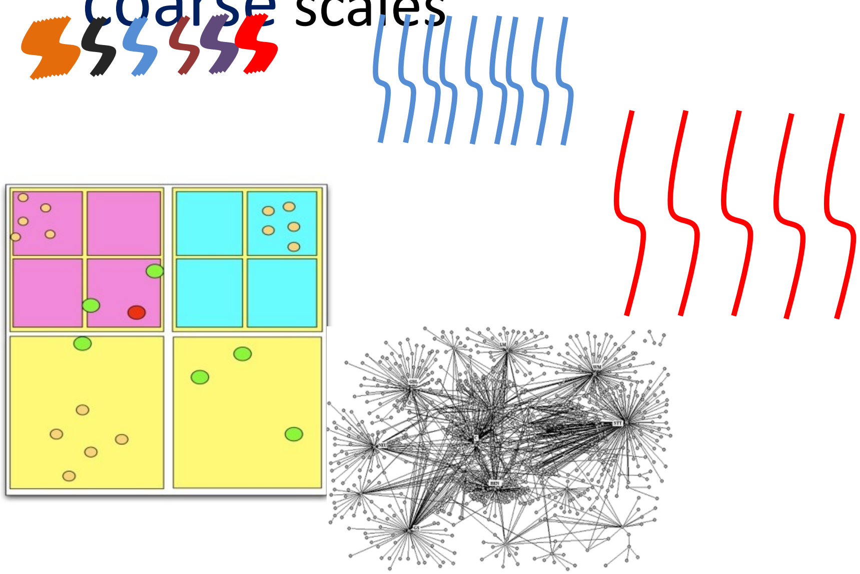
- 1990-2000:
  - NSF launches supercomputer centers program
  - Early breakthroughs in physics based modeling & simulation through parallel "capability" computing
- 2000-2010:
  - 1000x growth in peak computing rates
  - Big growth in modeling & simulation for science
  - High throughput science data generation
- 2010-
  - 1000x growth in peak computing rates
  - Multicore revolution: abundant TeraOps
  - Data deluge: 1000x faster growth than computing

Expanding Opportunities: Large Sparse Data Sets

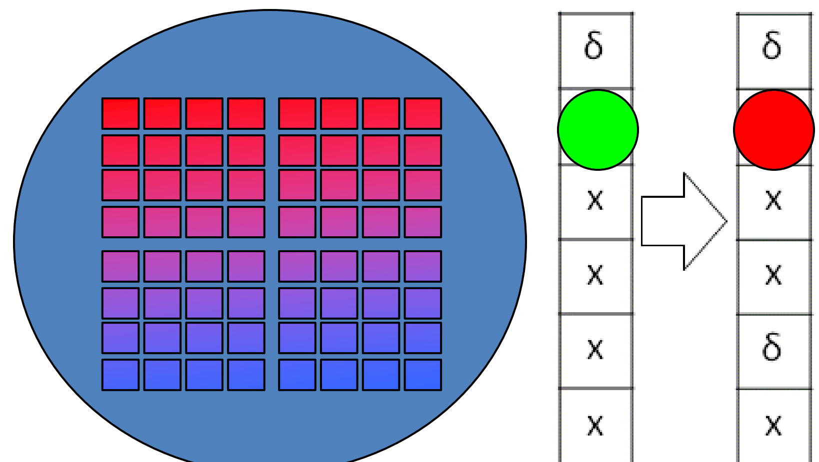


Where are the Opportunities?

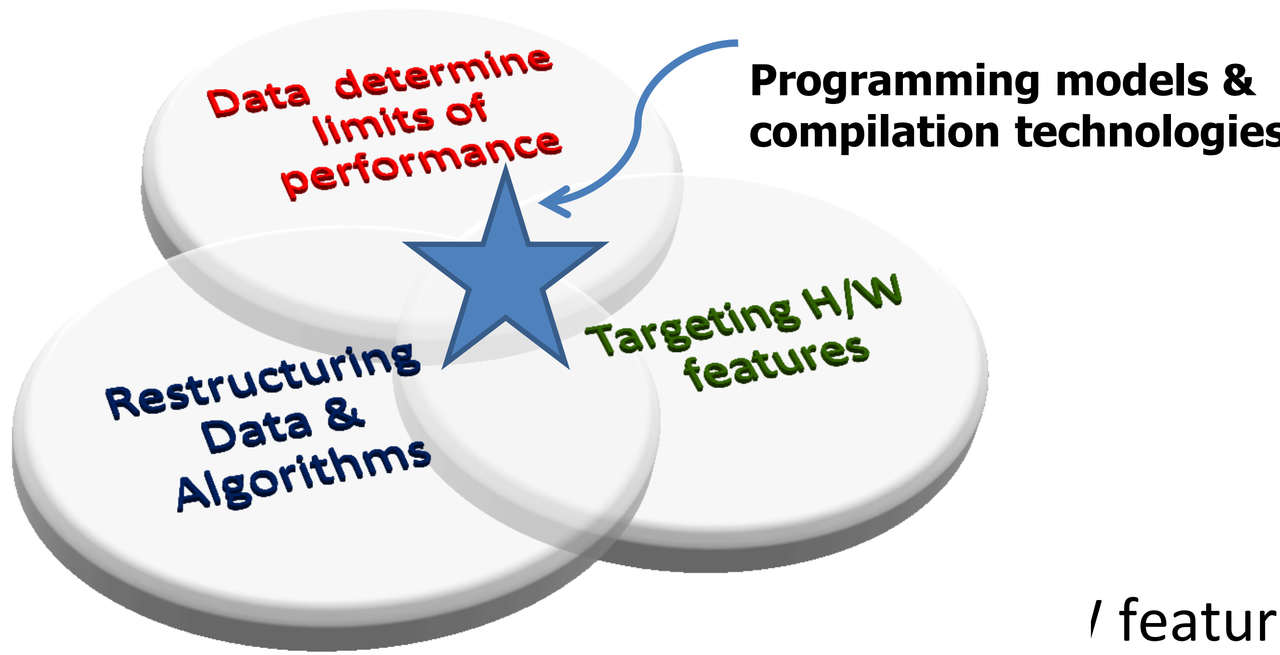
- Sparse data processing
  - Few ops-per-data
    - O(N) – or lower “sublinear” data accesses define performance efficiencies
  - Parallelism at fine, medium, coarse scales



- H/W trends
  - Fast, hot to slower, cooler
  - 2x cores/threads@18months
  - Heterogeneous
    - Process variability, GPUs
  - ~~Unreliable~~ Soft Errors

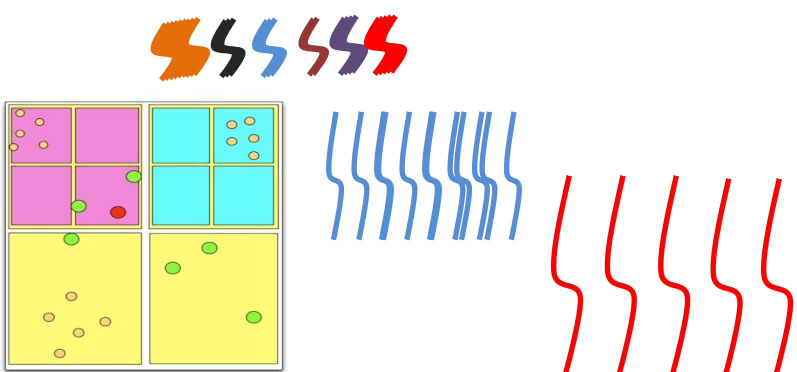


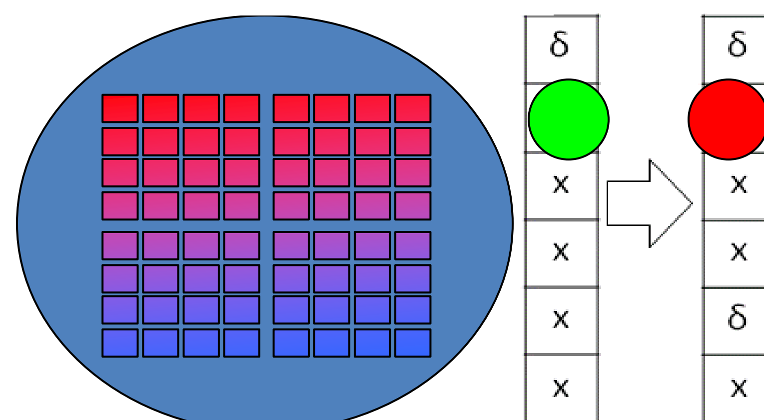
Increasing Need for Programming Models/ Run-Time Approaches



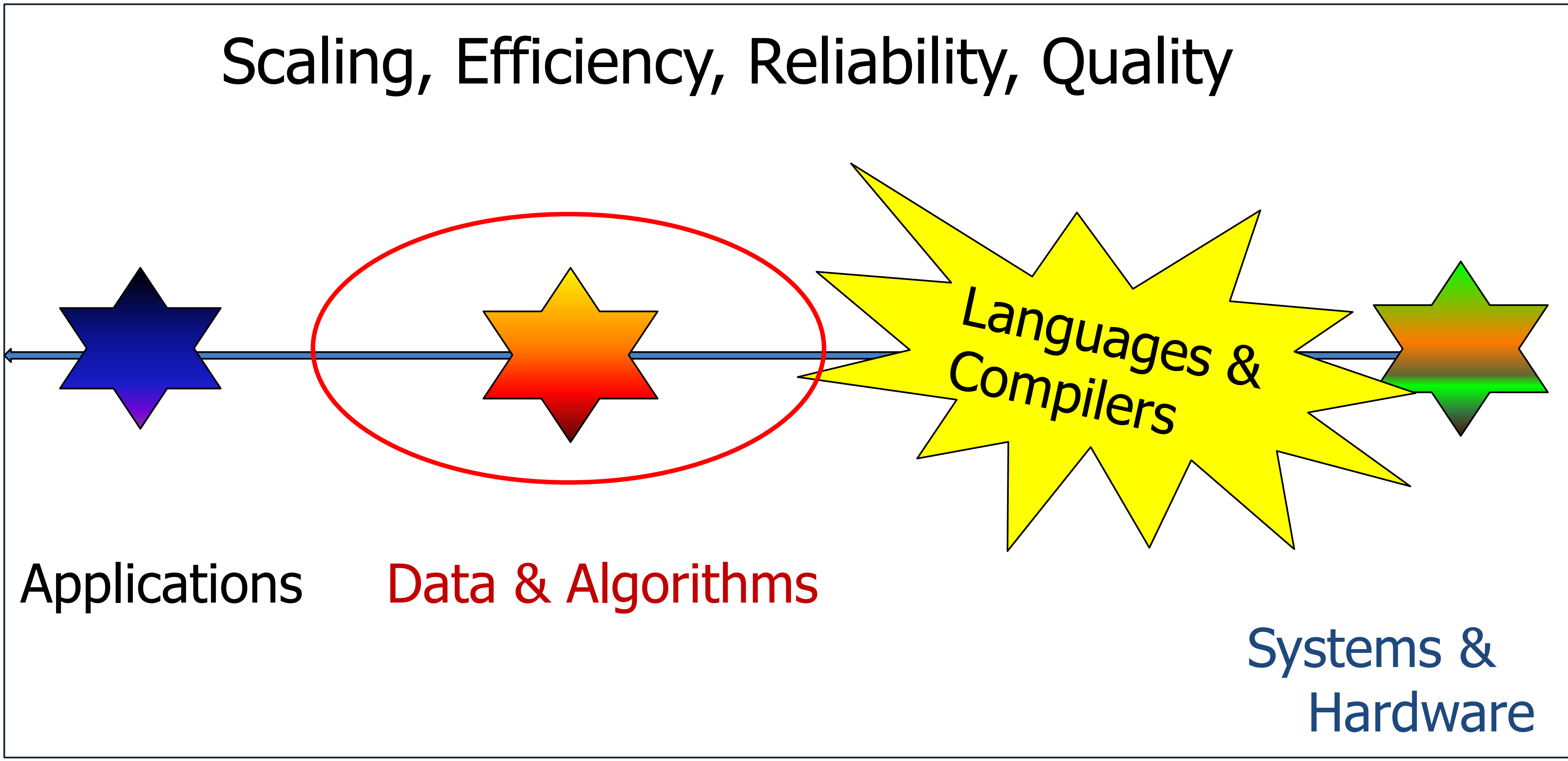
/ features

- Sparse Algorith





Addressing Performance Challenges



Acnowledgements

- Joint work with:
  - Guillaume Aupy, Josh Booth, Humayun Kabir (Penn State)
  - Anne Benoit (LIP, École Normale Supérieure de Lyon, France), and Yves Robert (LIP, École Normale Supérieure de Lyon, France & Univ of Tennessee)
- Thanks to
  - National Science Foundation & Penn State
  - LCPC organizers and attendees