A Data Affinity & Reuse Model for High Performance on NUMA Multicores
Or
Can we Afford Weak Scaling at a Multicore Node?

Padma Raghavan
Vanderbilt University

Workshop on Clusters, Clouds & Data for Scientific Computing, Oct 3-6 2016
Chateteauform, Le Maison des Contes

VANDERBILT
UNIVERSITY

Presenting joint work with:

Guillaume Aupy (Vanderbilt)
Joshua Booth (Sandia)
Anne Benoit (ENS-Lyon)
Humayun Kabir (Penn State)
Yves Robert (ENS-Lyon)

Most of the results are from SC15 Paper:
STS-k: A Multilevel Sparse Triangular Solution Scheme for NUMA Multicores

Research Supported by NSF & Vanderbilt University

A Very Simple Example
Triangular Solution

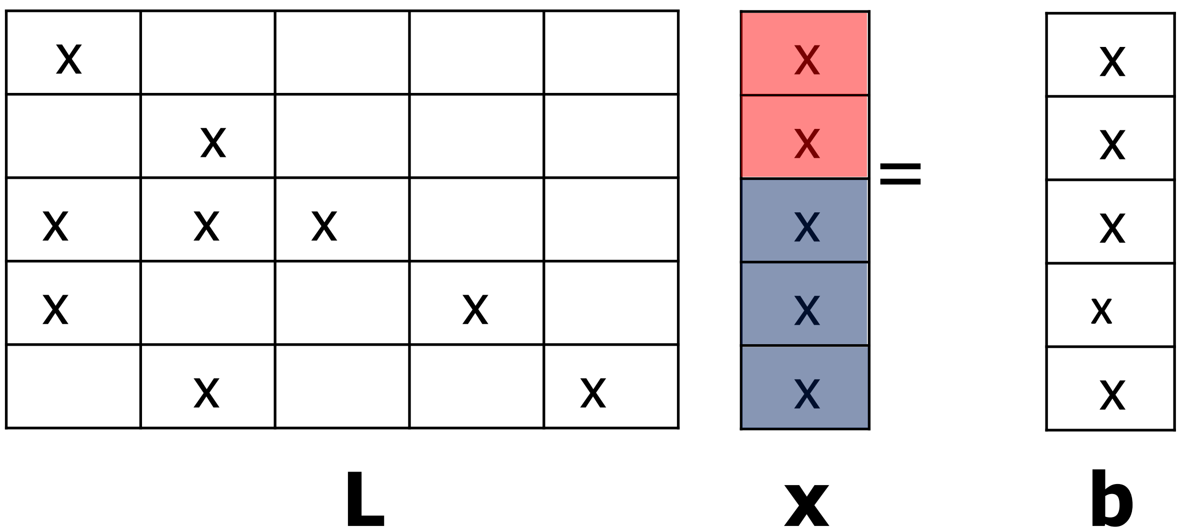
$Lx = b$; solve for x

L is a lower triangular matrix

L is sparse

Sparse- TS: Level Sets or Coloring for Parallel Computing

- Sparsity pattern permits parallel calculation of unknowns
- Example: 2 –color, each color is independent; level sets are the same for this example (not true in general)



1. Irregular to Regular: CSR to CSR-k: Rows to Super-Rows

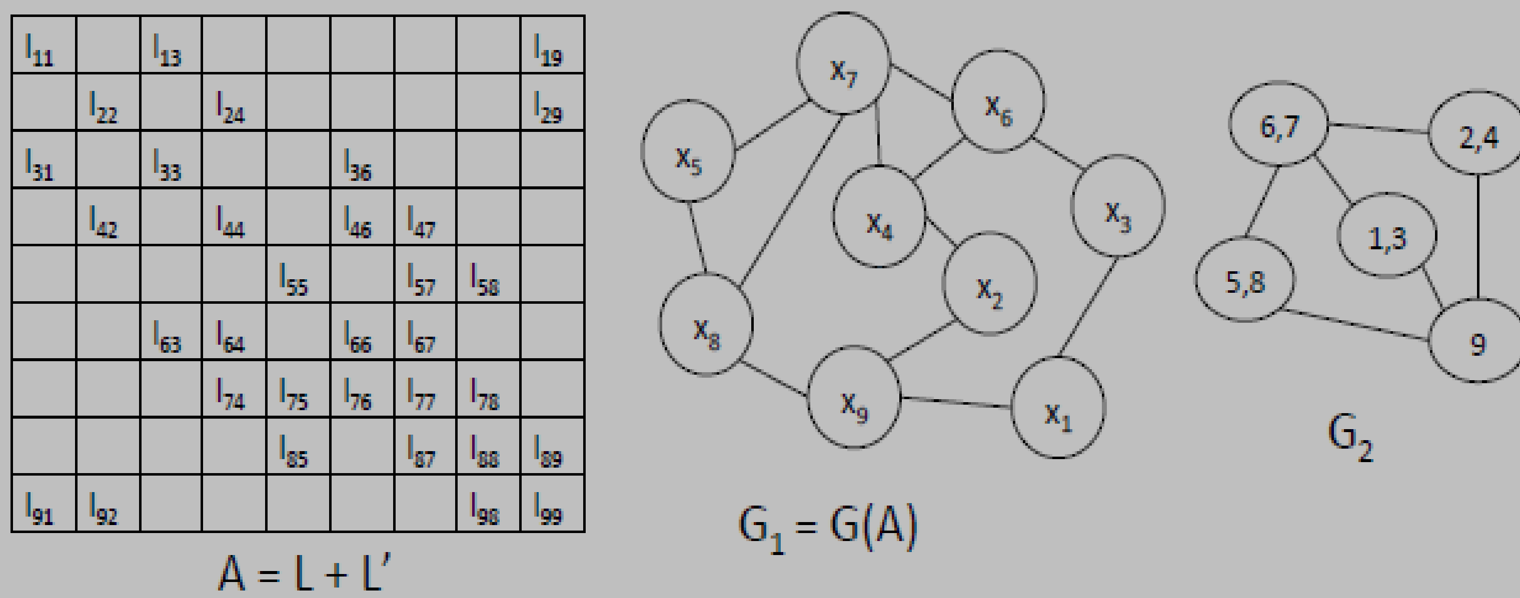


Figure 1: $A = L + L^T$ (left) and its graph G_1 (middle) transformed into G_2 (right) with super-rows through coarsening. A vertex of G_2 is formed by collapsing two connected vertices of G_1.

- Spatial locality in cache/memory
- Uniform length tasks at desired granularity

2. Parallelism: Level Sets or Coloring of Coarse Graph

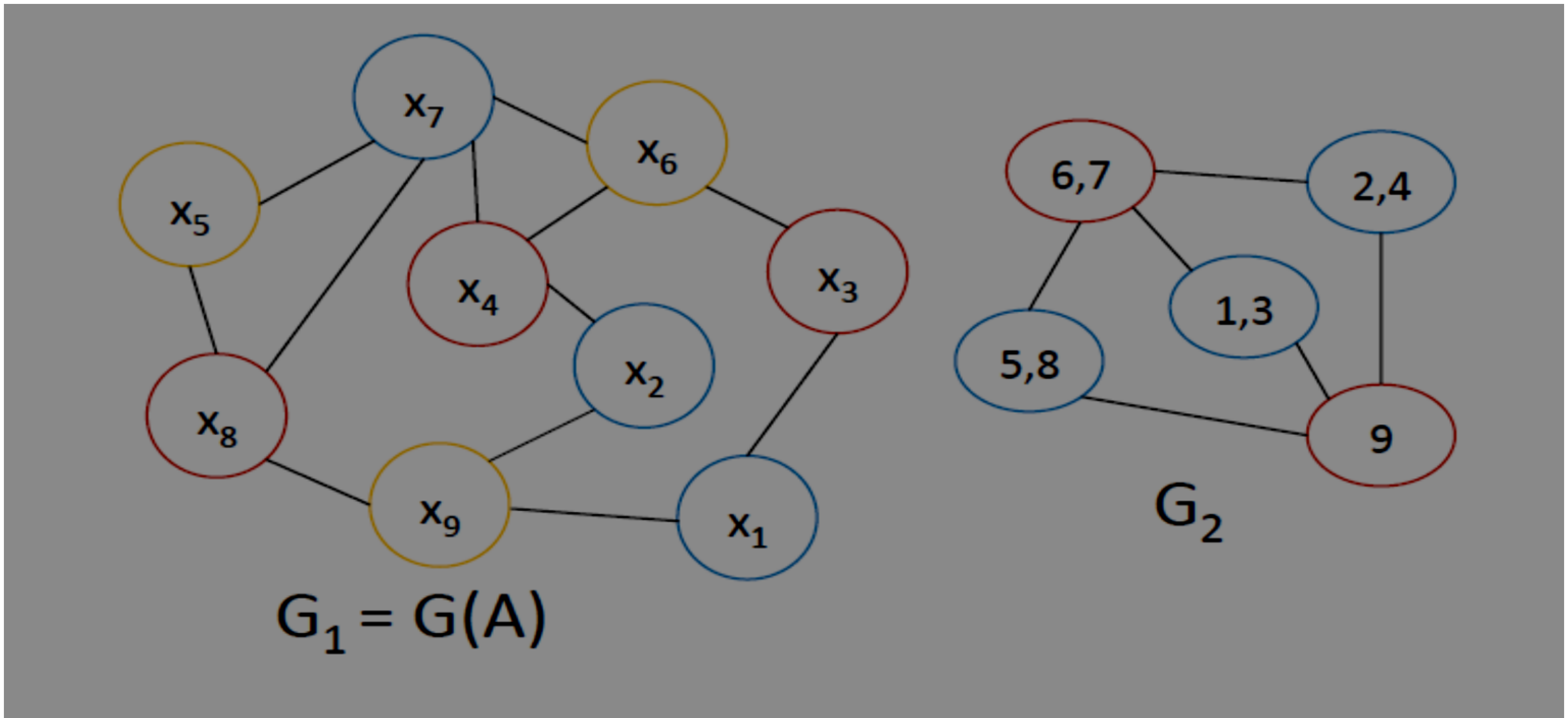
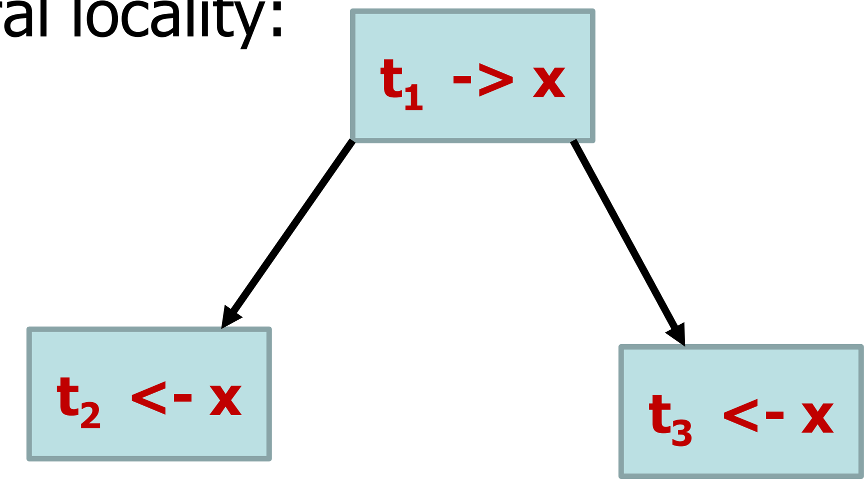


Figure 2: Independent sets or packs of rows obtained after coloring of G_1 (left) and packs of super-rows after coloring of G_2 of the example shown earlier in

- 2-coloring of CSR-2 representation
- Level sets can also be determined on CSR-2

From Spatial to Temporal Locality: Reuse of x

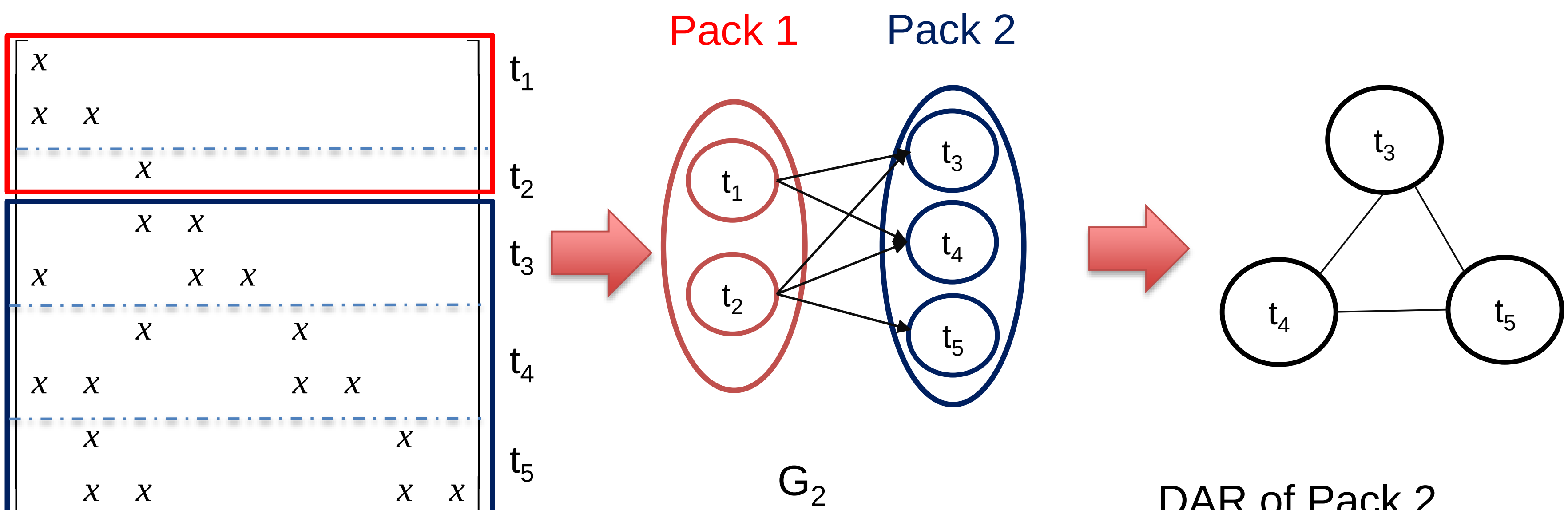
- Temporal locality:



- Pack: A set of tasks that can be solved in parallel
- Goal: Increase temporal locality between tasks in a pack

Temporal Locality: DAR graph of a Pack

- DAR (Data Affinity and Reuse) graph of a pack
  - Vertices are tasks
  - Edges are connection between tasks



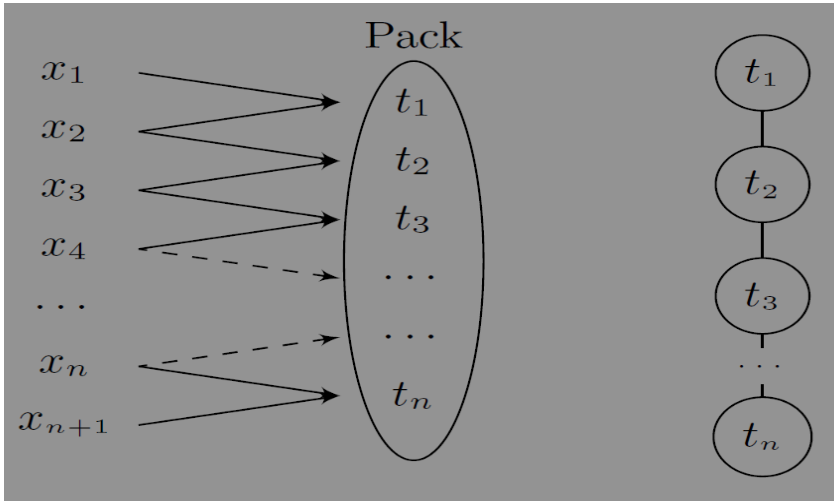
DAR of Pack 2
Tasks are connected if they share inputs

In-pack assignment problem for temporal locality

- In-Pack Assignment Problem (for reuse in x):
  - Input: a DAR graph of a pack
  - Output: Assignment of tasks to cores
  - Constraints:
    - Load is balanced across cores
    - Minimize data access cost

- NP-complete on a UMA (Uniform Memory Architecture) architecture (reduction from 3 Partition problem)

Insight into Solving In-Pack Assignment Problem



- If the DAR graph is a line, then an optimal schedule exists:
  - assign consecutive tasks of equal block size to cores
  - if there is q cores and n tasks: assign n/q consecutive tasks to a core
- Transform DAR graph in a near line form by doing a band-width reducing ordering

STS-K & Tests

- ➢Convert & store input matrix in CSR-k

Spatial locality

- ➢Find Packs in Graph of CSR-k

Extract parallelism: Use Level Sets or Coloring

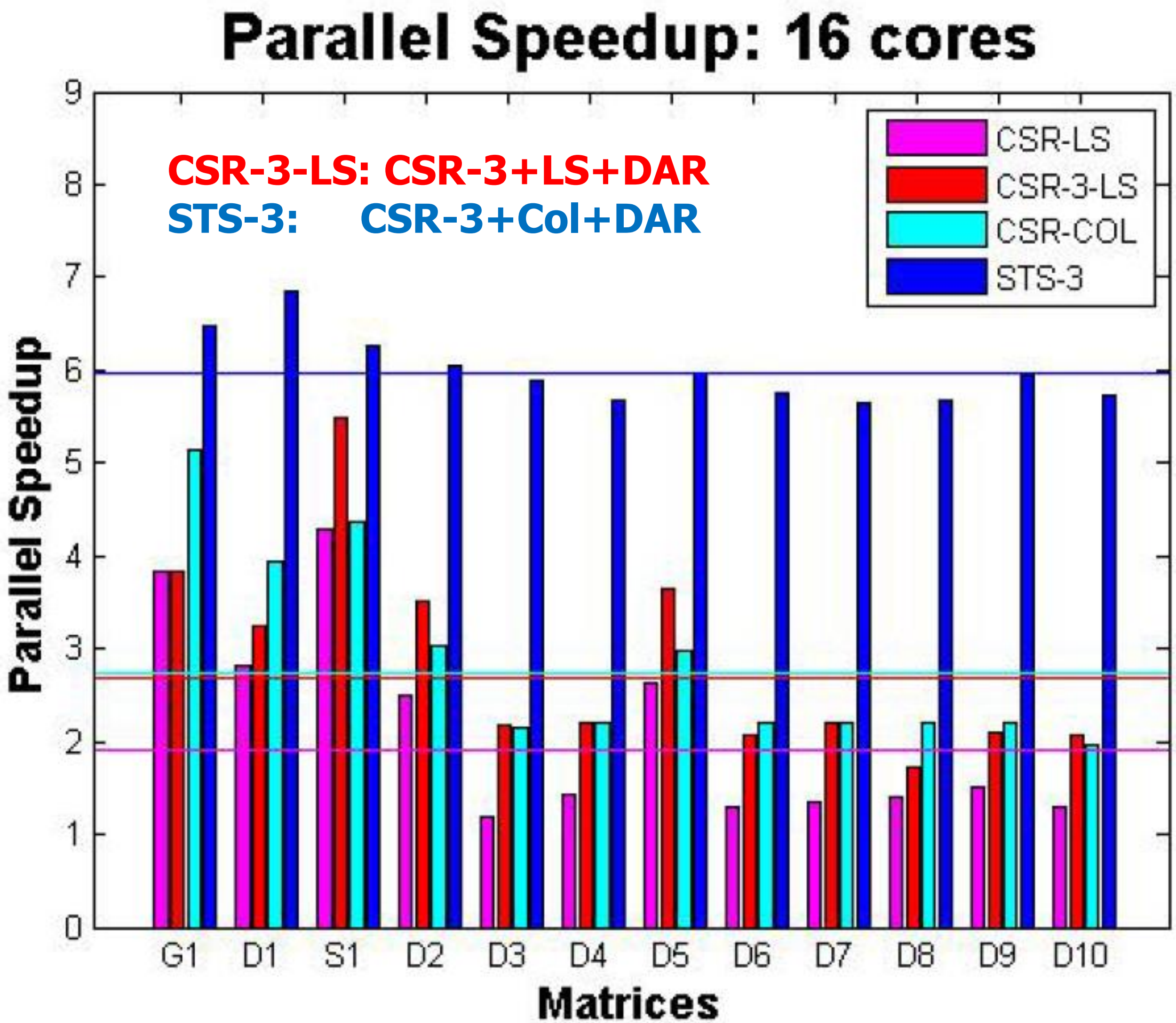
- ➢Make DAR graph of each Pack
- ➢Reorder DAR graph using band-width reducing ordering (near line form)

Temporal locality for reuse of x

Architecture	L1	L2	L3	#Cores
Intel	Private	Private	Shared	32
AMD	Private	Private	Shared	24

Intel Xeon-8837 & AMD-'Magny-Cours'

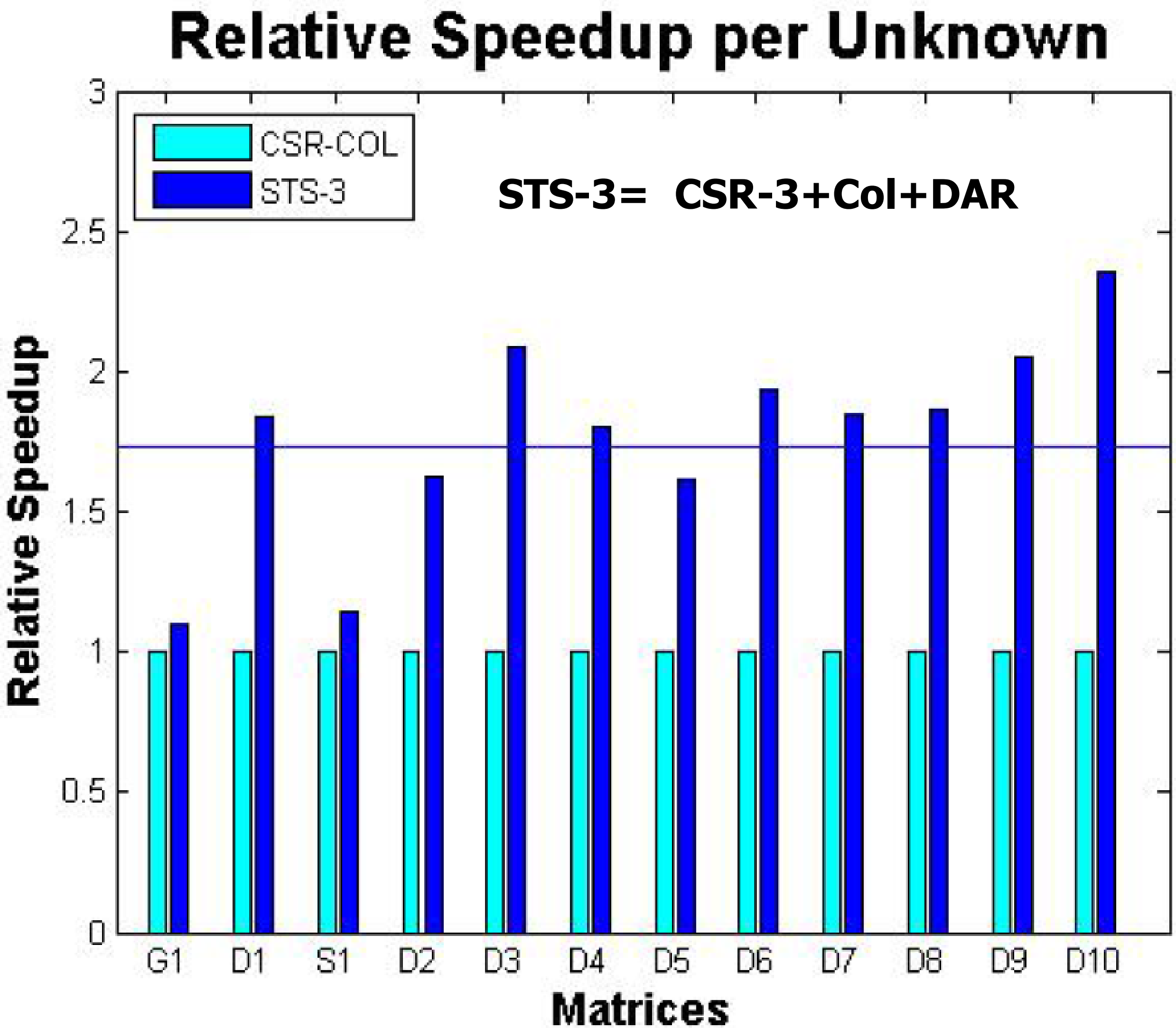
Parallel Speedup (Intel) vs CSR-LS



$$\frac{T(mat, \text{CSR-LS}, 1)}{T(mat, \text{method}, q)}$$

- STS-3 achieves 6x speedup compared to CSR-LS
- We observed similar results on AMD
- LS suffers from synchronization overheads; many packs of smaller size

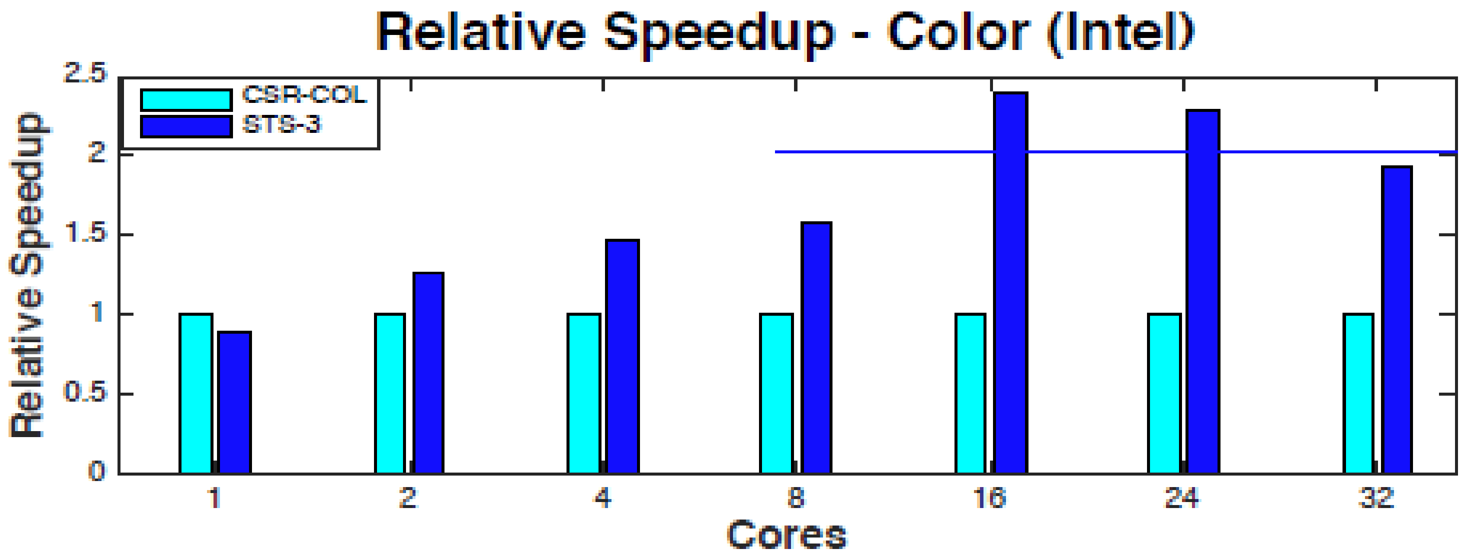
Effect of Data Locality in Largest Pack

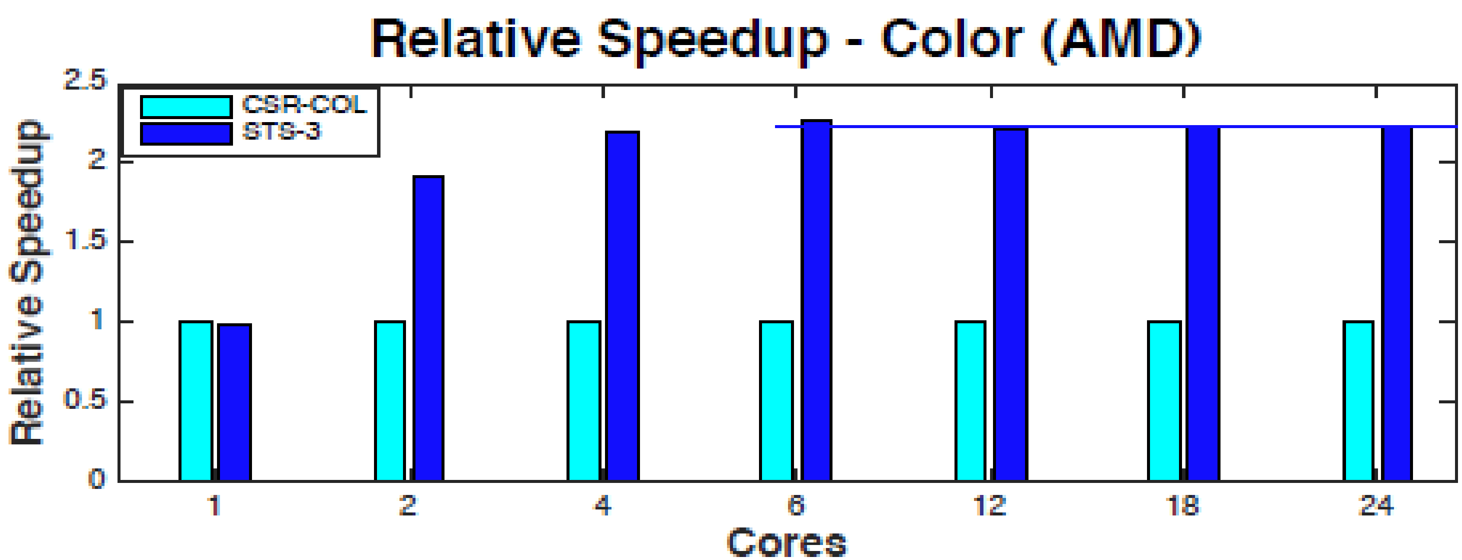


$$\frac{t(\text{CSR-COL}, q)}{t(\text{STS-3}, q)}$$

- q = 16 cores
- STS-3 achieves 1.75x speedup compared to CSR-COL
- Similar results hold on AMD

Effect of Data Locality for test suite 1-32/24 cores





So what?

- Dynamic task scheduling systems
- at multicore node could be very useful

Likely capture most of these types of performance advantages for many irregular applications

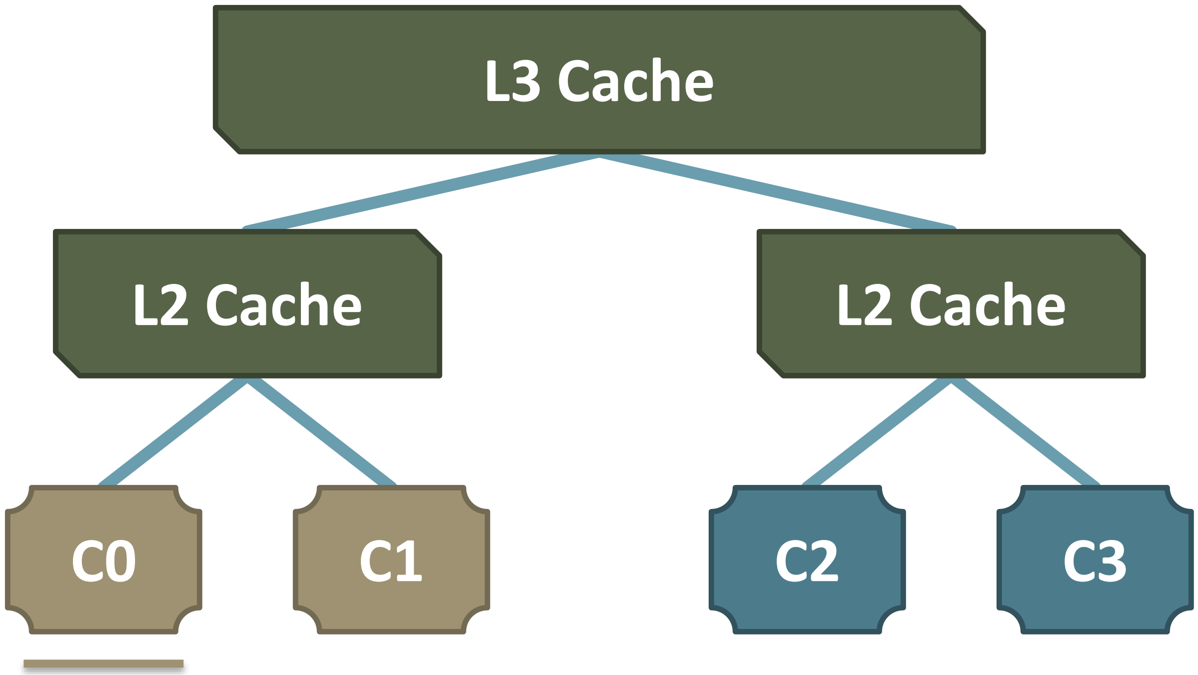
NUMA-Aware Temporal Reuse

- Pack n: Each task bi has been assigned to $core(bi)$
- Pack n+1: With tasks in $f1, f2, \ldots, fn$
- Let bi have data that can be reused by fi
- Probability of hit from reuse when fi is assigned $core(fi)$

$$P(hits, fi \mid core(bi)) \propto distance\ (core(fi), core(bi))$$

- If $fi\ \&\ fj\ \ have\ data\ affinity\ and\ reuse\ on\ same\ core\ \ or\ close\ core$

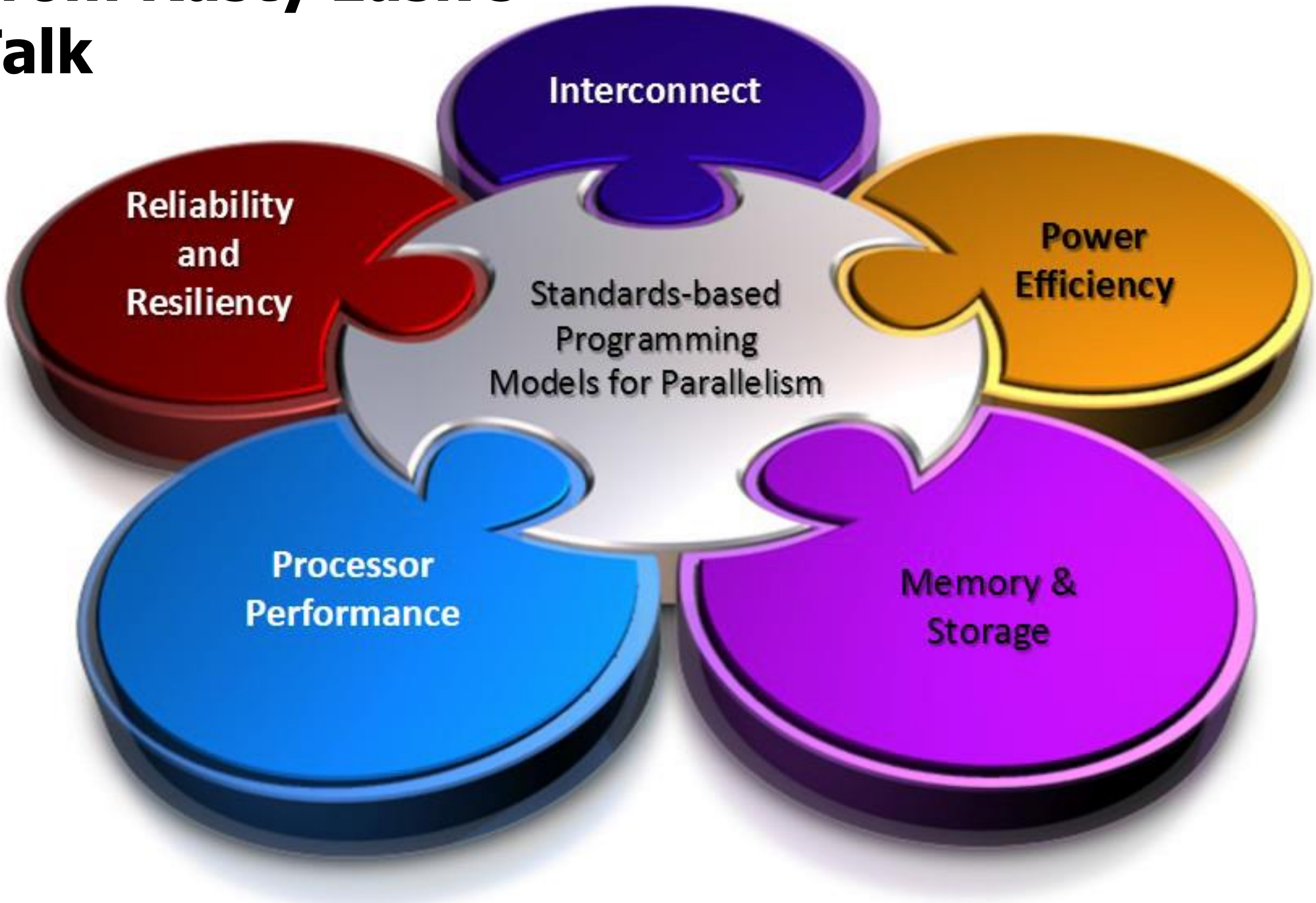
NUMA Distance Aware Dynamic Work Queues



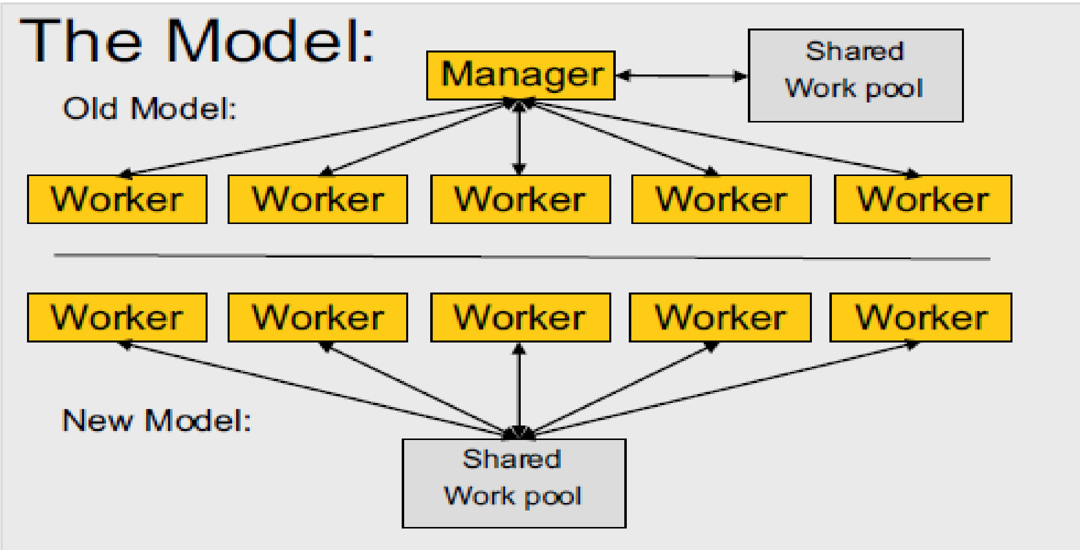
- C0: { C0, C1, C2, C3 }
- C1: { C1, C0, C3, C2 }
- C2: { C2, C3, C0, C1 }
- C3: { C3, C2, C1, C0 }

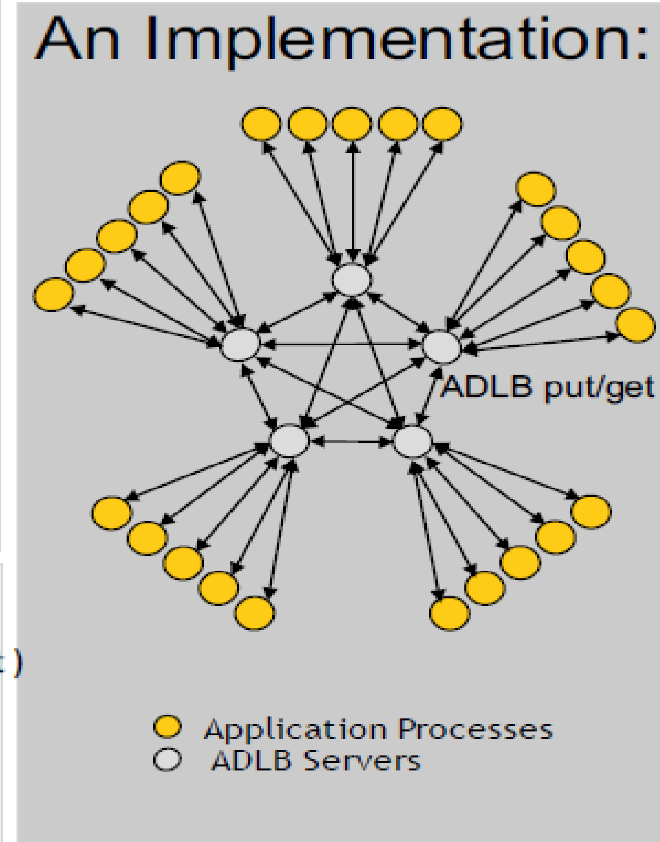
- Each core/thread has its own work queue; when out of work it traverses queues in order of NUMA-distance for work stealing
- It will likely provide most of the benefits when combined with useful abstractions get, put, affinity …

From Rusty Lusk's Talk



ADLB On One Slide





The API:

- ADLB_Put(type, priority, len, buf, target_rank, answer_dest)
- ADLB_Reserve(req_types, handle, len, type, prio, answer_dest)
- ADLB_Get_Reserved(handle, buffer)
- and a few housekeeping calls…

ADLB abstracts the idea of creating/acquiring work using put/get of work units into a work pool

Rusty Lusk: ADLP+ as DMEM for MPI, cross-node
Padma: Could be very useful for irregular computations at multicore node

Exascale

- Then, now and beyond
  - From fast, hot …to parallel, cooler
  - To billion-way parallel, heterogeneous, unreliable
- The action is at a node
  - Many cores, NUMA,NOCs, accelerators
- Can we afford weak scaling at a multicore node?