Low-overhead Load-balanced Scheduling for Sparse Tensor Computations

Muthu Baskaran, Benoit Meister, Richard Lethin

Introduction

ENSIGN (Exascale Non-Stationary Graph Notation)

- Tool for hypergraph (or multi-link graph) analysis as tensor decompositions
  - Automatic source-to-source optimization tool
    - High-level specification → fast parallel target code
- Solve important graph and data analytic problems
- Offers performance and productivity benefits
  - Compiler technologies to improve and scale existing key tensor computations
    - Inter-operate with Reservoir Labs' auto-parallelizing compiler R-Stream
  - Quick turnaround time from prototyping to developing high-performance codes

Presentation Roadmap

Background

Tensor Analysis Application

ENSIGN Techniques

Performance Evaluation

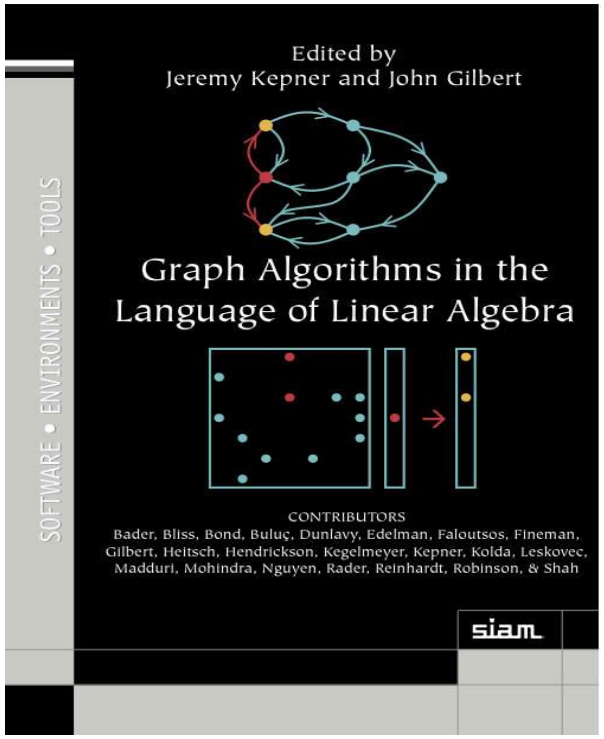
Summary & Forward Work

Linear Algebra and Graph Analysis

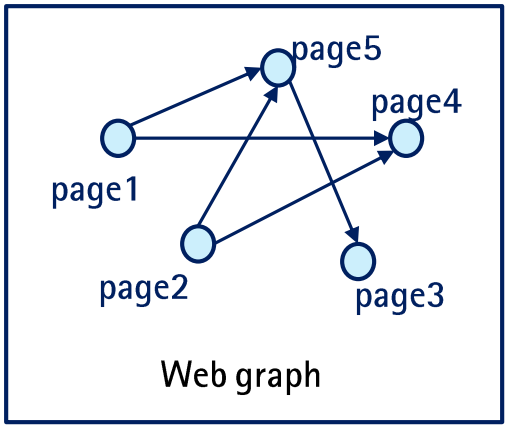
Popular graph problems using linear algebra formulations

- PageRank [Brin & Page, 1998]
  - Ranking web pages based on importance
  - Finding dominant Eigen vectors
- HITS [Kleinberg, 1998]
  - Finding authoritative web pages
  - Finding principal singular vectors
- Common graph computations
  - Let A be the adjacency matrix,

    D the degree matrix.
    - Connected components: Zero eigenvectors of D – A.
    - Community detection: Least non-zero eigenvector of D – A.



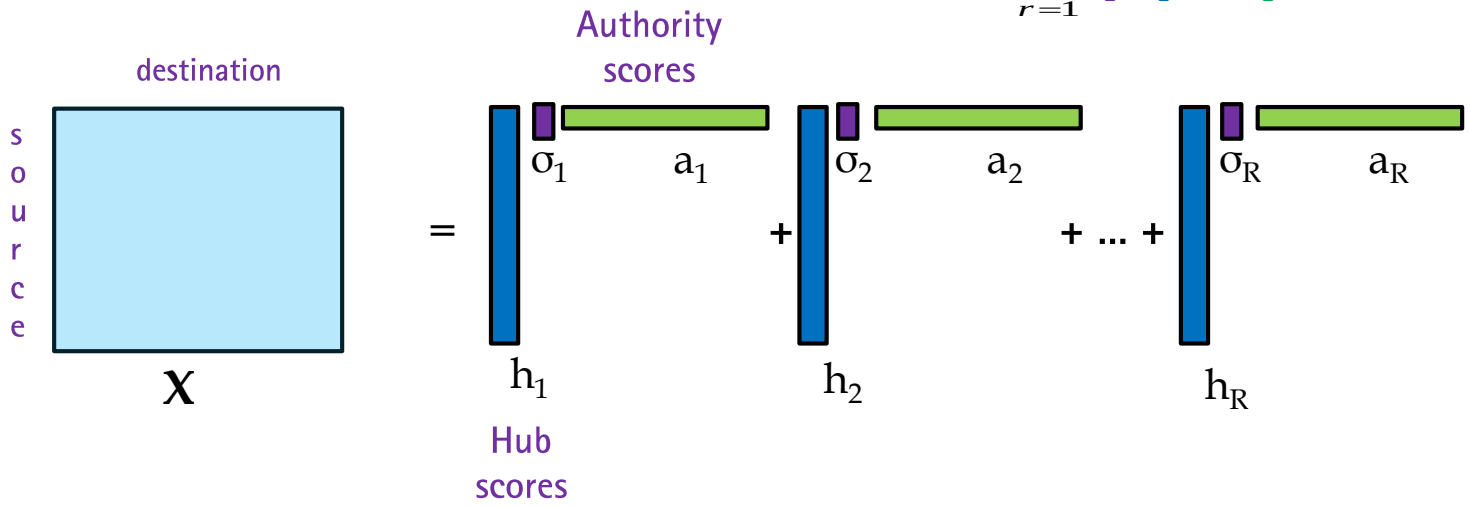
Web graph analysis using SVD



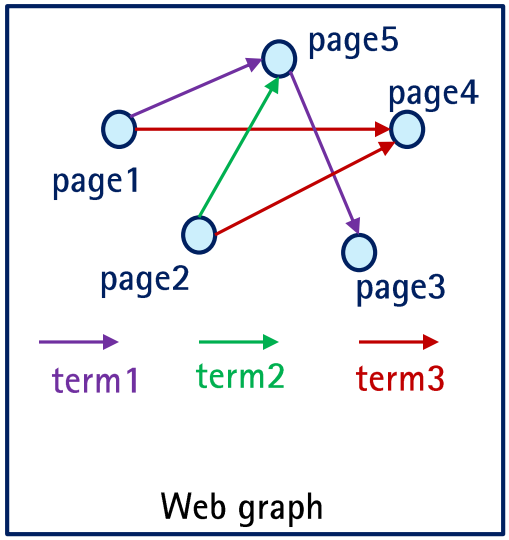
- Find authorities and hubs
  - Authorities: Pages that many important pages point to
  - Hubs: Pages that point to good authorities
- SVD on the "sparse" adjacency matrix of web graph

$$\mathbf{X} = \mathrm{H}\Sigma\mathrm{A}^{\mathrm{T}}$$

$$\mathbf{X} = \sum_{r=1}^{R} \sigma_r \, h_r \circ a_r$$

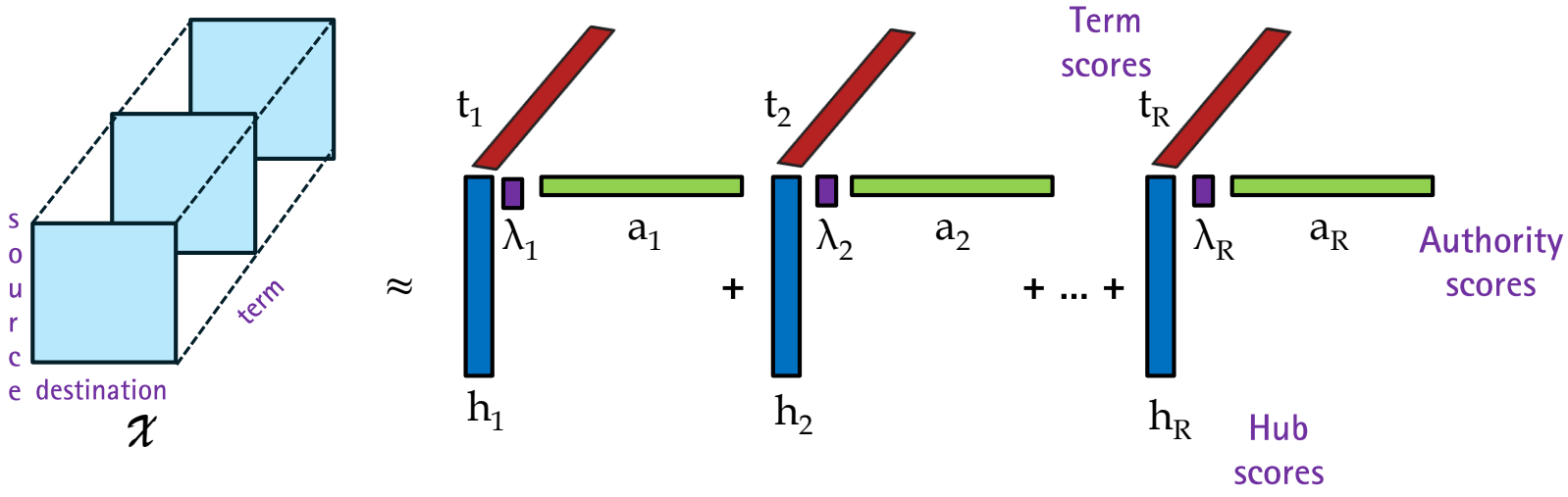


Tensor analogous of web graph analysis



- 3D view of the web graph with topical information added
  - **Very sparse** 3D adjacency "tensor"
- TOPHITS [Kolda et al., 2005]
- Tensor decomposition of the adjacency tensor

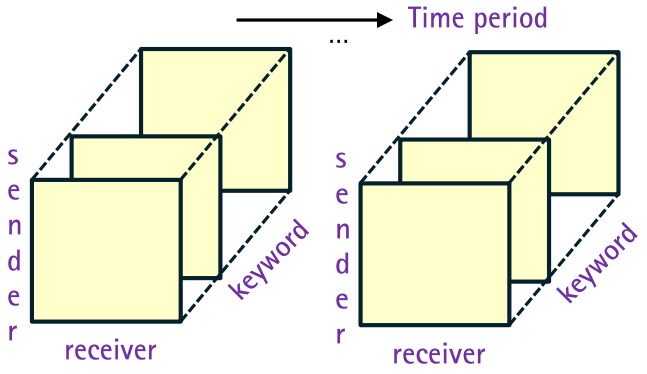
$$\mathcal{X} \approx \sum_{r=1}^{R} \lambda_r \, h_r \circ a_r \circ t_r$$



Multi-link Graphs or Hypergraphs and Multi-aspect Data

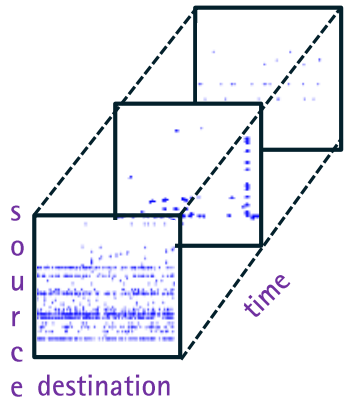
Real world Data

- Multi-dimensional
- Multi-aspect
- Large
- Sparse



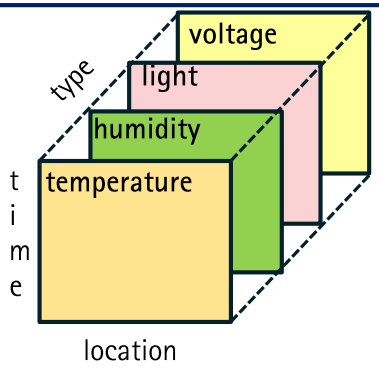
Sender x Receiver x Keyword x Time period

Email Data



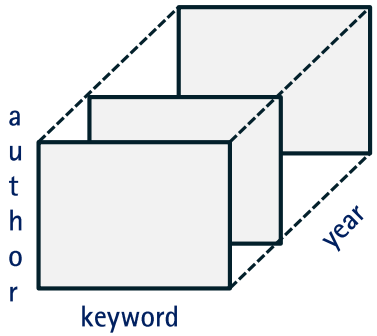
Src x Dest x Time

Network Traffic Data



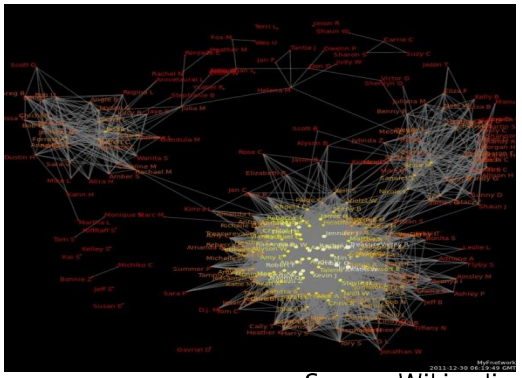
Time x Location x Type

Environmental Sensor Monitoring



Author x Keyword x Year

Bibiliometric Data



Source: Wikipedia

Person x Person x Relation

Social network graph

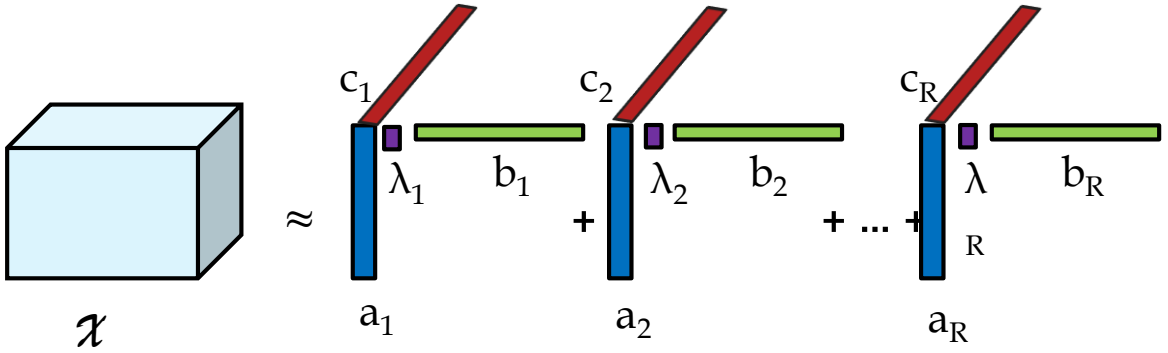
Multi-linear Algebra for Analyzing Multi-aspect Data

- CANDECOMP/PARAFAC (CP) tensor decomposition
  - Factorizes a tensor into a sum of component rank-one tensors

$$\mathcal{X} \approx \sum_{r=1}^{R} \lambda_r \, a_r \circ b_r \circ c_r$$

CP decomposition algorithm (ALS method)

repeat
  for $n = 1, \ldots, N$ do
    $\mathbf{V} \leftarrow \mathbf{A}^{(1)\mathsf{T}}\mathbf{A}^{(1)} * \cdots * \mathbf{A}^{(n-1)\mathsf{T}}\mathbf{A}^{(n-1)} * \mathbf{A}^{(n+1)\mathsf{T}}\mathbf{A}^{(n+1)} * \cdots * \mathbf{A}^{(N)\mathsf{T}}\mathbf{A}^{(N)}$
    $\mathbf{A}^{(n)} \leftarrow \mathbf{X}_{(n)}(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)})\mathbf{V}^{\dagger}$
    normalize columns of $\mathbf{A}^{(n)}$ (storing norms as $\boldsymbol{\lambda}$)
  end for
until fit ceases to improve or maximum iterations exhausted



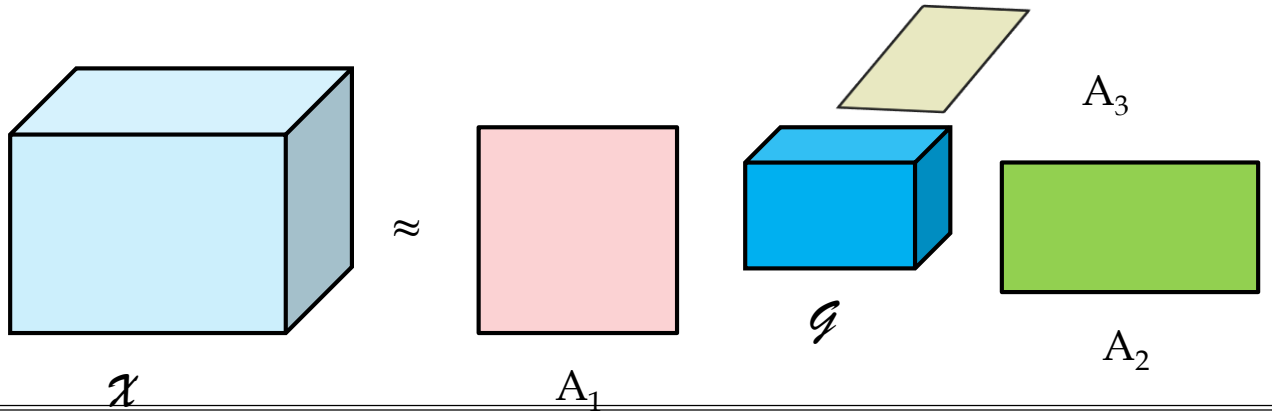
Multi-linear Algebra for Analyzing Multi-aspect Data

- Tucker tensor decomposition
  - Factorizes a tensor into a core tensor and a set of factor matrices (one along each mode)

$$\mathcal{X} \approx \mathcal{G} \times_1 A_1 \times_2 A_2 \times_3 A_3$$

Tucker decomposition algorithm (HOOI method)

repeat
  for $n = 1 \ldots N$ **do**
    $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{A}^{(1)T} \cdots \times_{n-1} \mathbf{A}^{(n-1)T} \times_{n+1} \mathbf{A}^{(n+1)T} \cdots \times_N \mathbf{A}^{(N)T}$
    $\mathbf{A}_n = J_n$ leading left singular vectors of $\mathbf{Y}_n$
  end for
  $\mathcal{G} = \mathcal{Y} \times_N \mathbf{A}^{(N)T}$
until convergence



Presentation Roadmap

Background

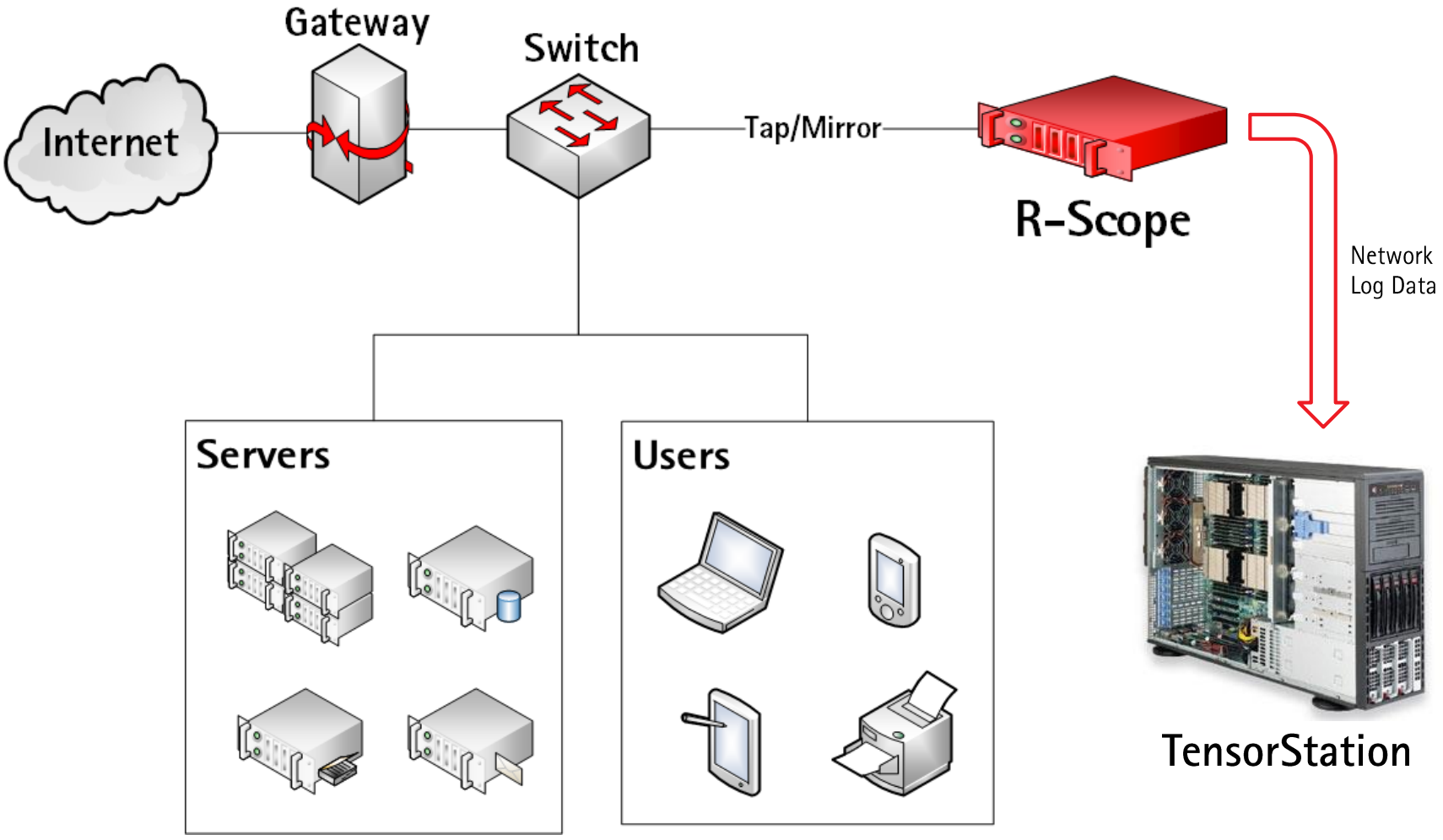
Tensor Analysis Application

ENSIGN Techniques

Performance Evaluation

Summary & Forward Work

Reservoir Network Traffic Data



Reservoir Network Traffic Data

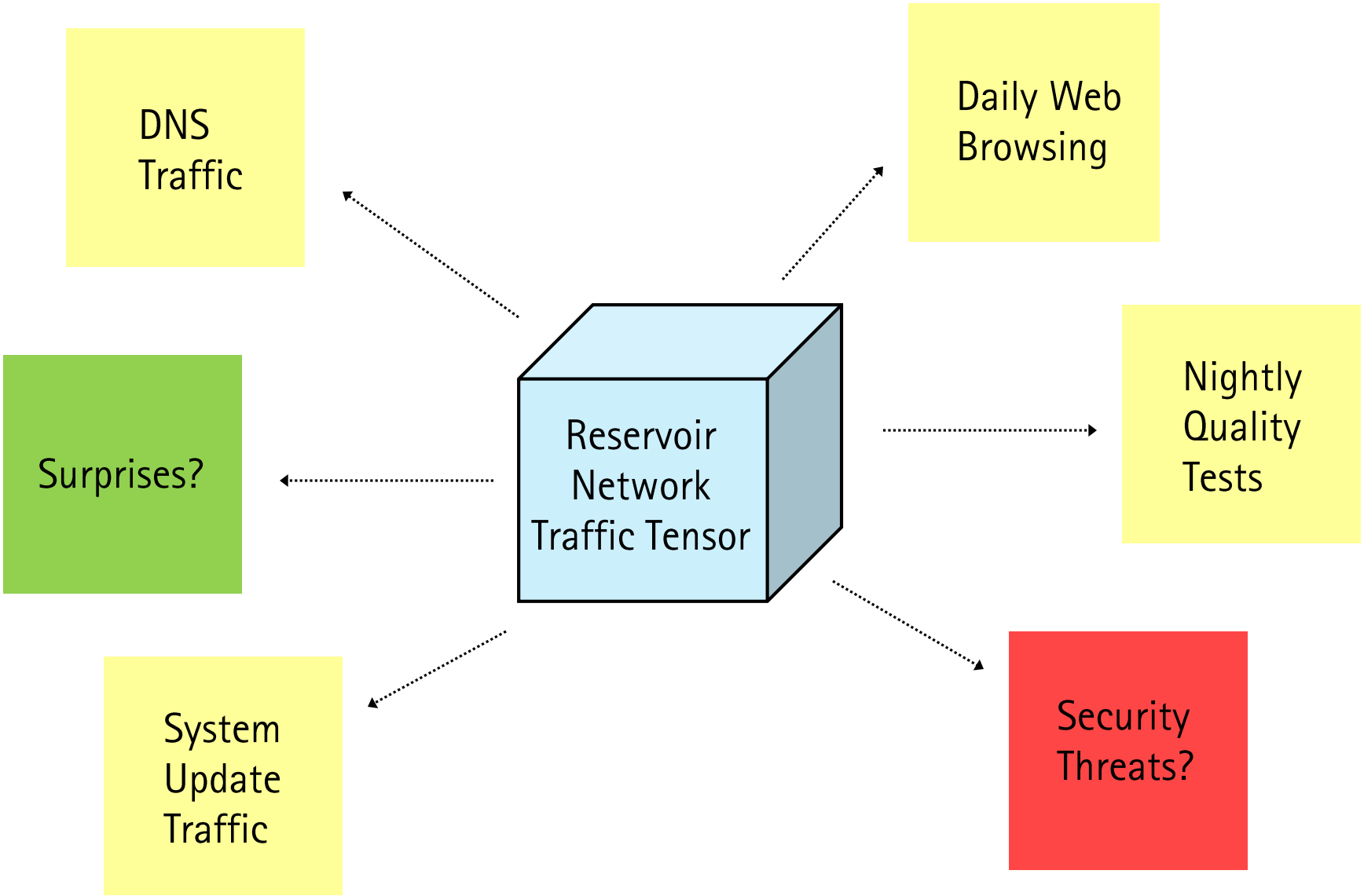
Reservoir Network Traffic Tensor

- Multiple attributes
  - (e.g. 9 dimensional : Time, IP (sender, receiver), port (sender, receiver), protocol, # bytes (sender, receiver), URL)
- Very larger tensor
  - (e.g. 1.5 M x 3664 x 47890 x 3664 x 47869 x 3 x 20175 x 20175 x 2343 from one day)
- Millions of messages (e.g. 2,298,967 from one day)

Excerpt:

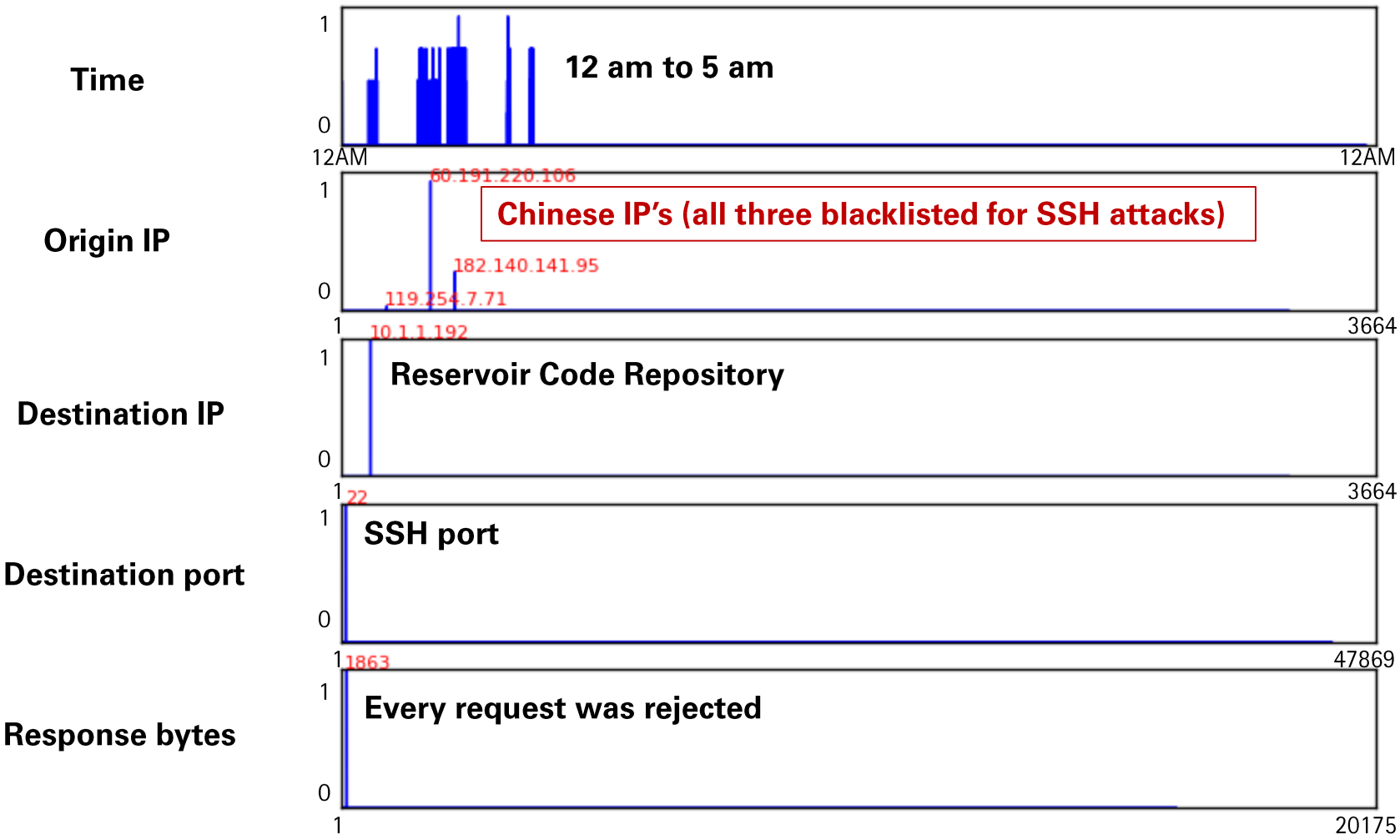
1367954284.775933	10.0.2.15	41173	72.22.185.216	80	tcp	0	0	-
1367954284.776223	10.0.2.15	35904	23.6.146.127	80	tcp	0	0	-
1367954284.902059	10.0.2.15	57323	63.251.85.24	80	tcp	2294	797	wt.o.nytimes.com
367954285.233317	10.0.2.15	51934	54.243.131.12	80	tcp	578	184	
3.0950611367954281.046495	10.0.2.15	51237	10.1.1.1	53	udp	32	200	-
1367954281.980046	10.0.2.15	61472	10.1.1.1	53	udp	33	128	-
1367954287.375995	10.0.2.15	55776	74.125.226.214	443	tcp	489	2658	-
1367954282.655601	10.0.2.15	52392	72.22.185.207	80	tcp	0	0	-
1367954282.651031	10.0.2.15	45201	72.22.185.198	80	tcp	0	0	-
1367954282.654099	10.0.2.15	52391	72.22.185.207	80	tcp	0	0	-
1367954282.660982	10.0.2.15	34639	72.22.185.214	80	tcp	0	0	-
1367954282.689652	10.0.2.15	58404	72.22.185.199	80	tcp	0	0	-
1367954283.208290	10.0.2.15	34980	72.21.91.19	80	tcp	394	313	p.typekit.net

Components from Tensor Decomposition



Reservoir Network Data Analysis

One Factor: SSH Attacks (Component 71 of 120)



Presentation Roadmap

Background

Tensor Analysis Application

ENSIGN Techniques

Performance Evaluation

Summary & Forward Work

Our Earlier Work

Our earlier work on optimizing sparse tensor computations*

- New efficient sparse tensor formats
  - Efficiently handle the sparseness in input data
- High-level algorithmic modifications
  - Avoid unnecessary computations
  - Improve data reuse
- Techniques for handling large sparse data sets
  - Avoid memory blowup in storing tensors

* Baskaran et al., "Efficient and Scalable Computations with Sparse Tensors", IEEE HPEC 2012.

Focus of this Work

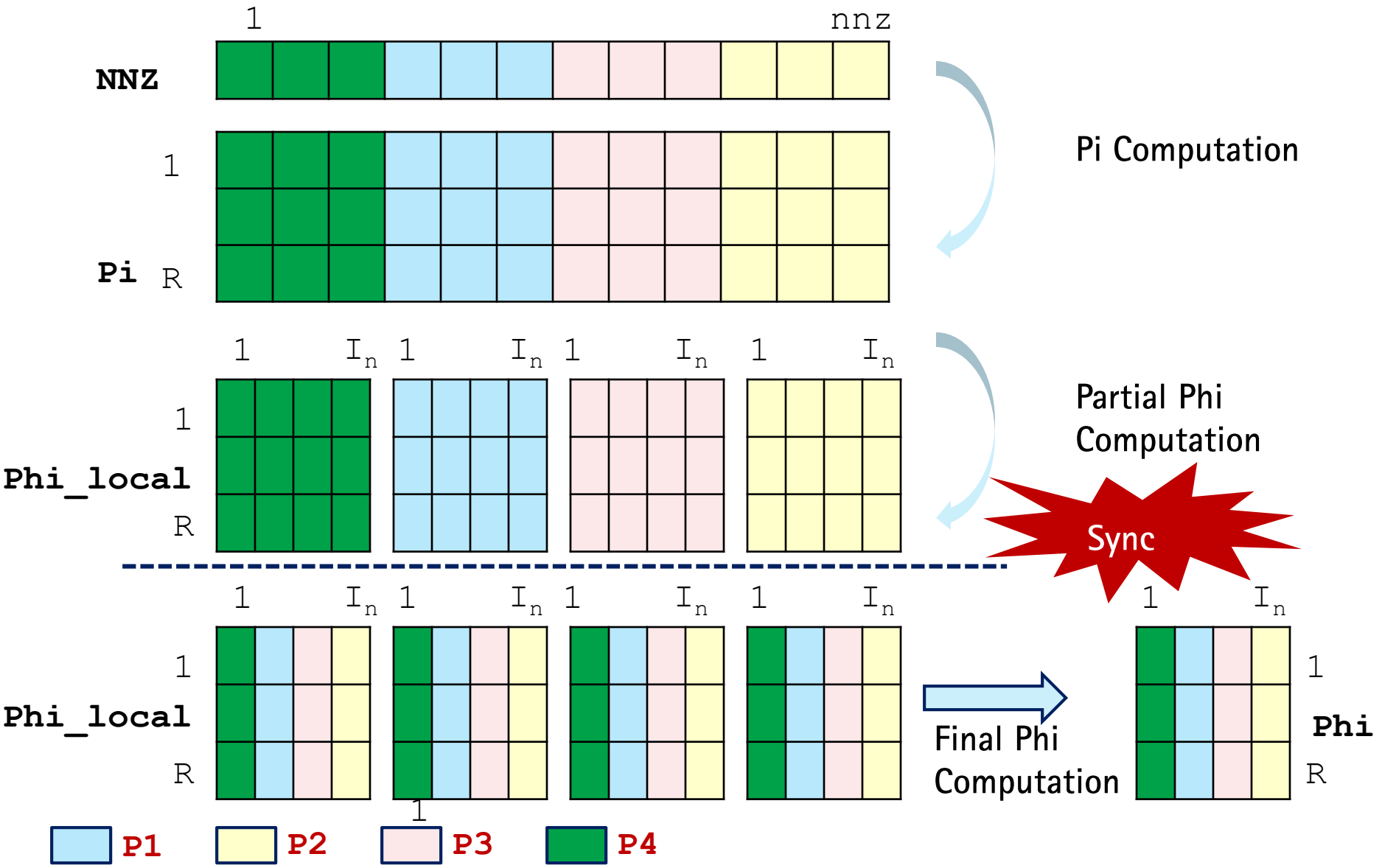
Challenge: Optimizing "sparse" computations

- A "data-driven" scheduling problem
- Need to efficiently handle irregular memory accesses
- **R-Stream** approach for optimizing sparse computations
  - Hiding irregularity in "black-boxes"
  - Parallelism and locality addressed
- Current parallelization efforts (including **R-Stream**) have scope for improvement
  - More parallelism
  - Reduced synchronization
  - Improved data locality

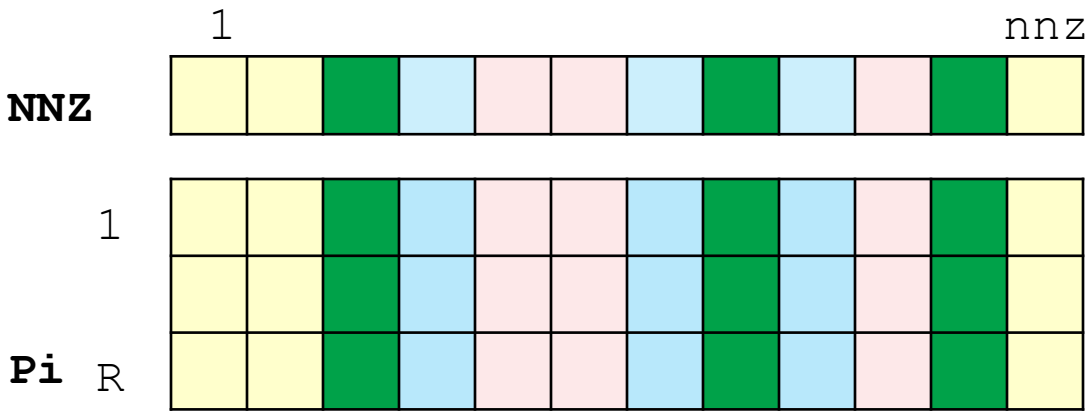
Focus of this Work

- Goals of ENSIGN improvisation techniques
  - Uncover more concurrency
  - Reduce synchronization
  - Improve data locality
  - Achieve load balance
  - Reduce scheduling overhead

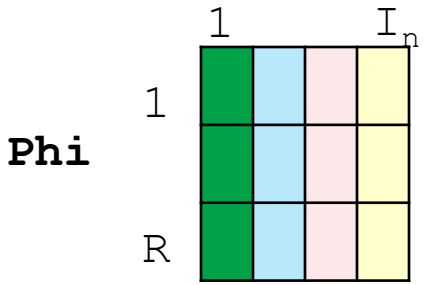
Improved Parallelization



Improved Parallelization



Pi Computation



Phi Computation

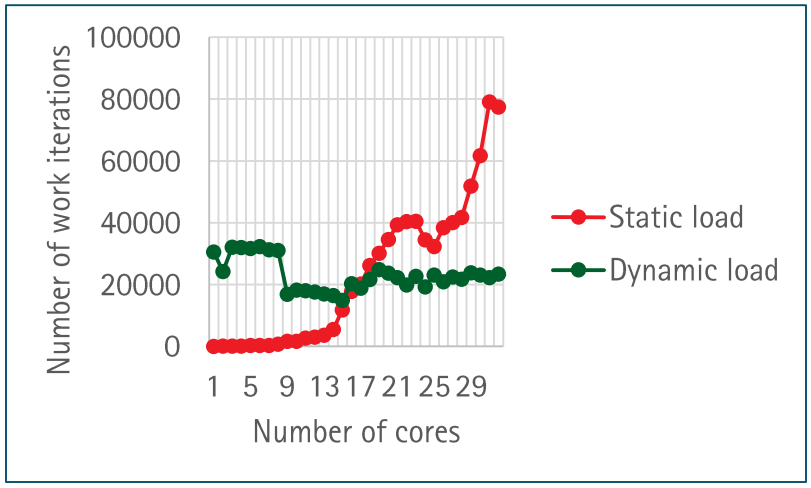
Proper partitioning of non-zeros to partition the computation among processors without synchronization

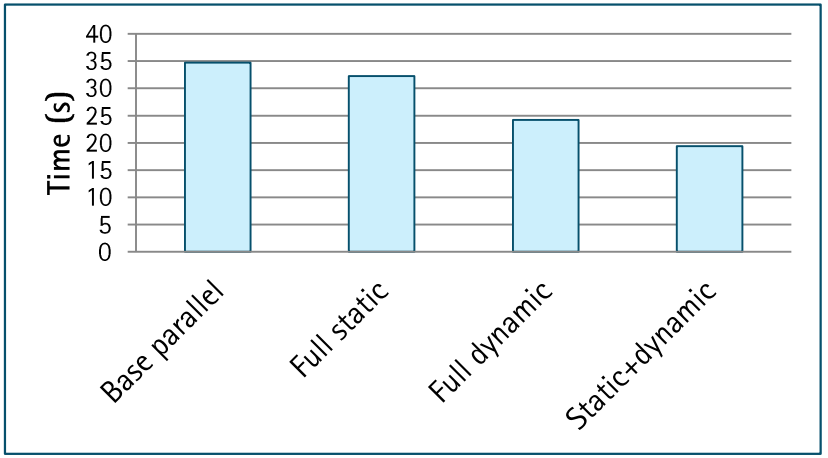
Need to balance the work load among the processors

P1 P2 P3 P4

Mixed static and dynamic runtime scheduling

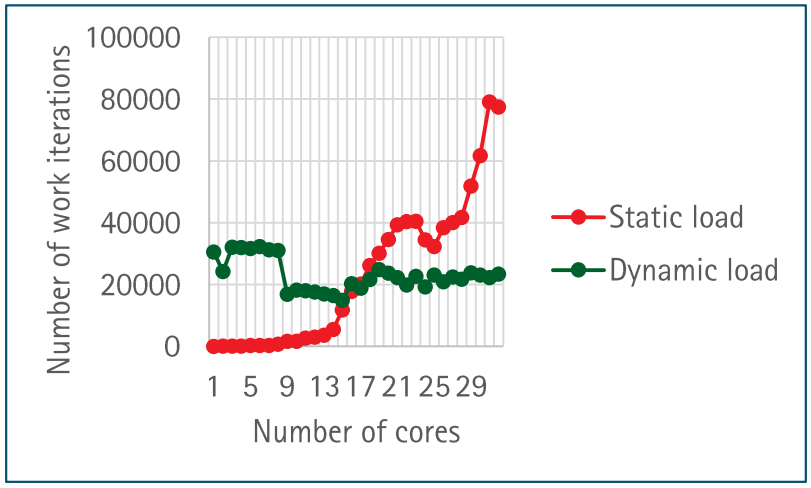
- Static scheduling - poor load balance, low scheduling overhead
- Dynamic scheduling - good load balance, high scheduling overhead
- Our Approach – Achieves the pros of both schemes
  - One dynamic scheduling iteration to get a load balanced pattern
  - Static scheduling using the pattern for later iterations
  - good load balance, low scheduling overhead

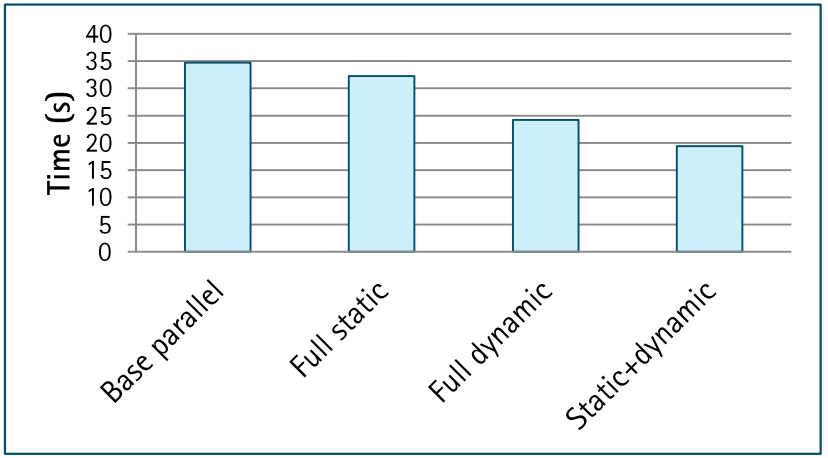




Mixed static and dynamic runtime scheduling

- Currently use OpenMP runtime for the dynamically scheduled iteration
- Additional memory for storing the schedule information
- Initial investigation in progress using Open Community Runtime (OCR)
  - Use OCR for all iterations
  - Use OCR for one iteration and use the schedule for later iterations





Improved Data Locality

- Memory-hierarchy aware approach
  - Task distribution across processor cores in the dynamic scheduling iteration governed by
    - Data touched by them
    - Memory in which data resides
  - Over-loaded cores "steal" tasks from "topologically" closer neighbors that are under-loaded
    - NUMA topology in shared memory systems
  - Facilitate data sharing across cores

Presentation Roadmap

Background

Tensor Analysis Application

ENSIGN Techniques

Performance Evaluation

Summary & Forward Work

Performance Evaluation

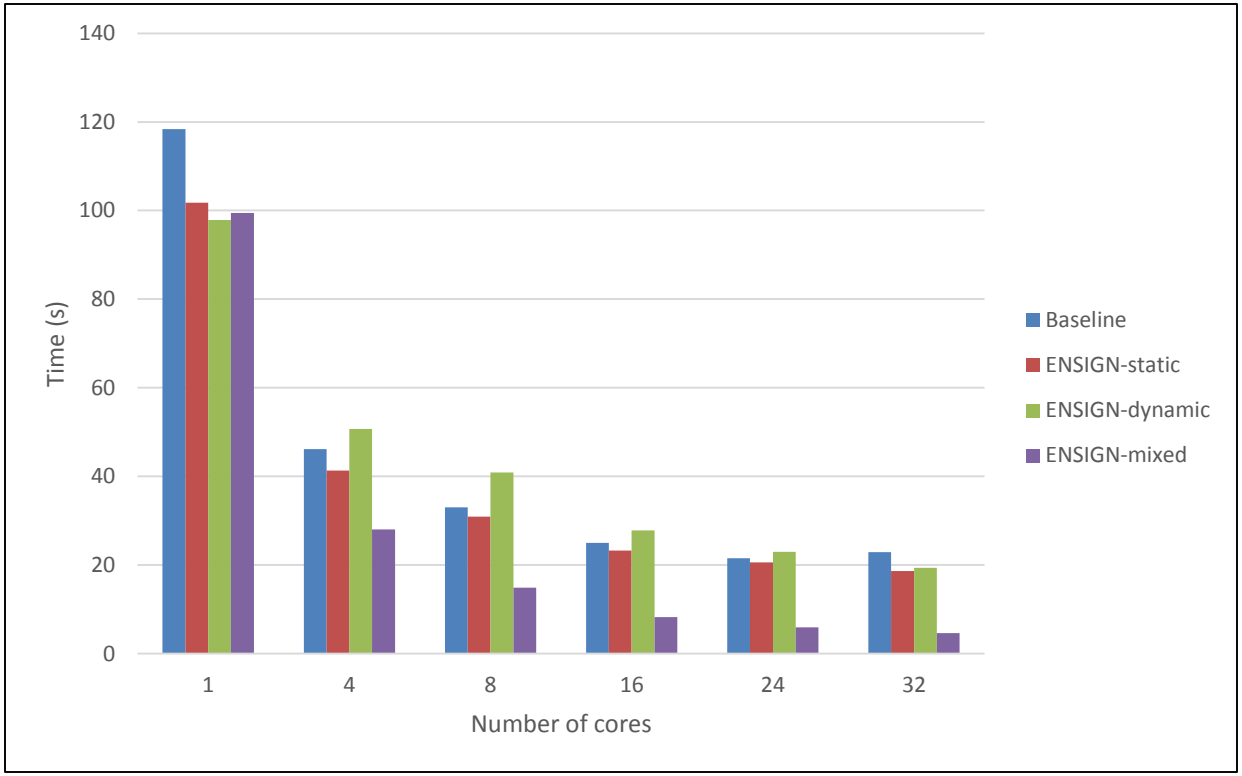
Benchmarked using CP-APR (Alternating Poisson regression) method

- Evaluated using
  - Three different real tensor data
  - Intel Xeon E5-4620 2.2 GHz (Quad socket 8-core)

Reservoir "Tensorstation"

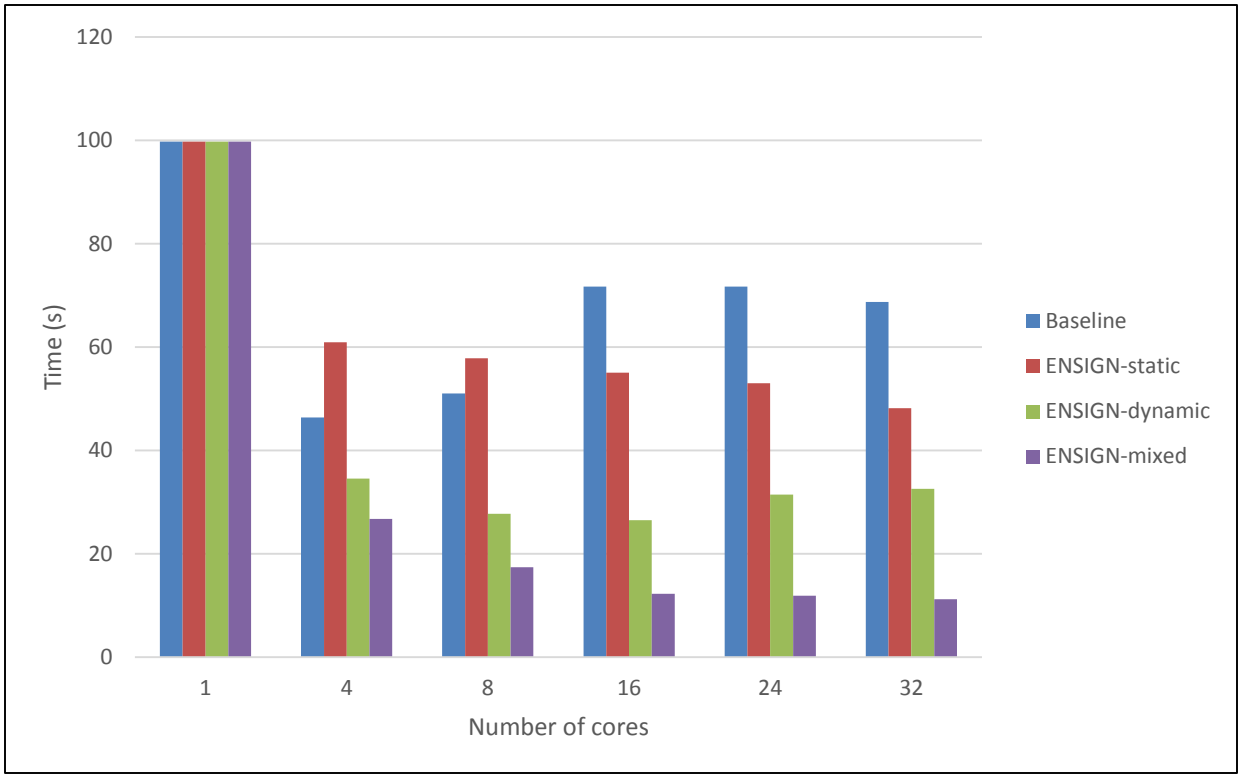
Performance Evaluation

Tensor	Size	Non-zeros	#iterations timed
Facebook	63891 x 63891 x 1591	737934	190



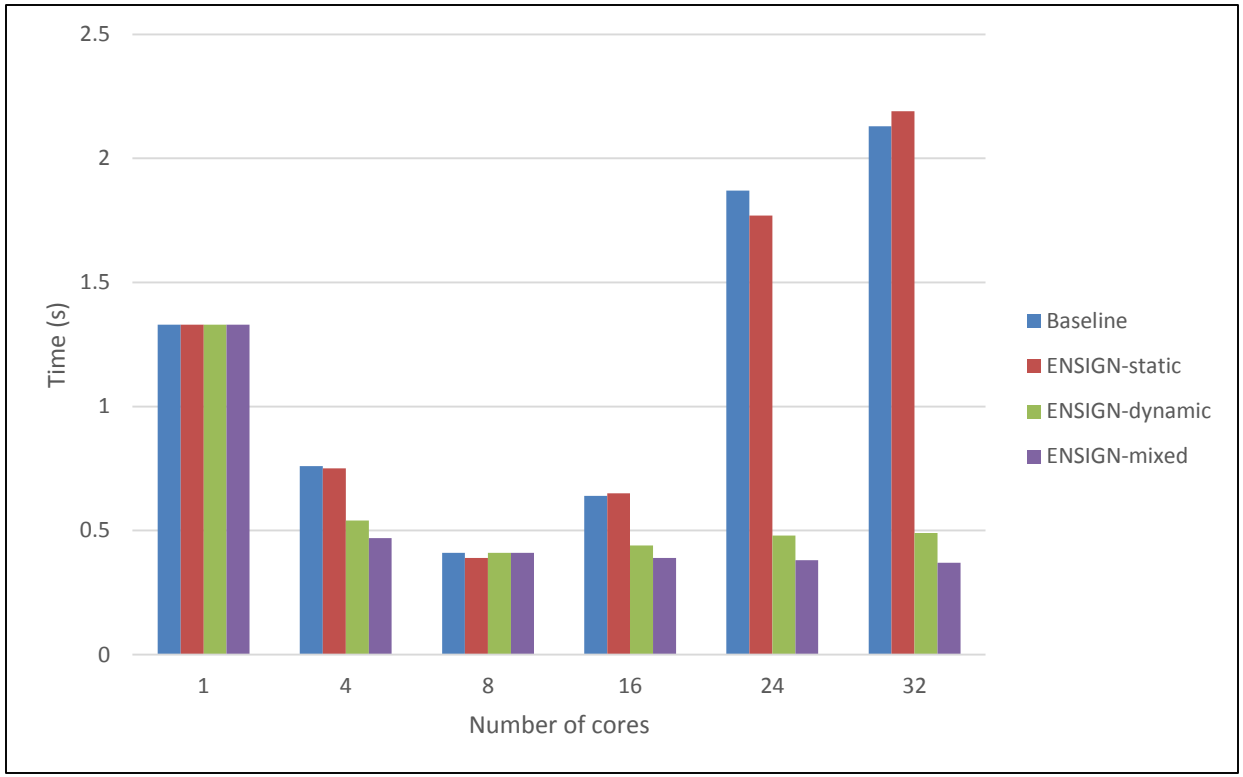
Performance Evaluation

Tensor	Size	Non-zeros	#iterations timed
Cyber	14811811 x1899 x1899 x 3 x 6067	2085108	50



Performance Evaluation

Tensor	Size	Non-zeros	#iterations timed
Enron	105 x 105 x 27	5418	180



Presentation Roadmap

Background

Tensor Analysis Application

ENSIGN Techniques

Performance Evaluation

Summary & Forward Work

Summary & Forward Work

What we do

- Developed techniques to effectively parallelize and scale large sparse tensor computations
  - Low scheduling overhead
  - Good load balance
  - Reduced synchronization
  - Improved data locality

What we plan to do

- Integration with scalable runtime systems such as Open Community Runtime (OCR)
- More scaling to larger parallel computers / larger problems
  - Distributed systems
  - Large shared memory systems (e.g. SGI UV)