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.
    - 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

\[
X = H \Sigma A^T
\]
\[
X = \sum_{r=1}^{R} \sigma_r h_r \circ a_r
\]
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

\[ \chi \approx \sum_{r=1}^{R} \lambda_r h_r \circ a_r \circ t_r \]
Multi-link Graphs or Hypergraphs and Multi-aspect Data

Real world Data
- Multi-dimensional
- Multi-aspect
- Large
- Sparse

Email Data
Sender x Receiver x Keyword x Time period

Network Traffic Data
Src x Dest x Time

Environmental Sensor Monitoring
Time x Location x Type

Bibliometric Data
Author x Keyword x Year

Social network graph
Person x Person x Relation
Multi-linear Algebra for Analyzing Multi-aspect Data

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

\[ \chi \approx \sum_{r=1}^{R} \lambda_r a_r \odot b_r \odot c_r \]

**CP decomposition algorithm (ALS method)**

```plaintext
repeat
  for n = 1, ..., N do
    V = \( A^{(1)} \odot A^{(1)} \odot \cdots \odot A^{(n-1)} \odot A^{(n-1)} \odot A^{(n+1)} \odot A^{(n+1)} \odot \cdots \odot A^{(N)} \odot A^{(N)} \)
    \( A^{(n)} \leftarrow X^{(n)}(A^{(N)} \odot \cdots \odot A^{(n+1)} \odot A^{(n-1)} \odot \cdots \odot A^{(1)})V \)
  normalize columns of \( A^{(n)} \) (storing norms as \( \lambda \))
end for
until fit ceases to improve or maximum iterations exhausted
```

\( \chi \approx \sum_{r=1}^{R} \lambda_r a_r \odot b_r \odot c_r \)
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)

\[ \mathbf{X} \approx \mathbf{G} x_1 A_1 x_2 A_2 x_3 A_3 \]

Tucker decomposition algorithm (HOOL method)

```
repeat
  for n = 1 ... N do
    \[ y = \mathbf{X} \times_1 A_1^{(1)T} \cdots \times_{n-1} A_{n-1}^{(n-1)T} \times_{n+1} A_{n+1}^{(n+1)T} \cdots \times_N A_N^{(N)T} \]
    \[ A_n = J_n \text{ leading left singular vectors of } Y_n \]
  end for
  \[ \mathbf{G} = y \times_N A_N^{(N)T} \]
until convergence
```
Presentation Roadmap

Background

Tensor Analysis Application

ENSIGN Techniques

Performance Evaluation

Summary & Forward Work
Reservoir Network Traffic Data

Internet → Gateway → Switch → Tap/Mirror → R-Scope → Network Log Data

- **Servers**
- **Users**

TensorStation
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:

<table>
<thead>
<tr>
<th>Time</th>
<th>IP</th>
<th>Protocol</th>
<th>Port</th>
<th>Length</th>
<th>Source</th>
<th>Destination</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1367954284.775933</td>
<td>10.0.2.15</td>
<td>41173</td>
<td>72.22.185.216</td>
<td>80</td>
<td>tcp</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1367954284.776223</td>
<td>10.0.2.15</td>
<td>35904</td>
<td>23.6.146.127</td>
<td>80</td>
<td>tcp</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1367954284.902059</td>
<td>10.0.2.15</td>
<td>57323</td>
<td>63.251.85.24</td>
<td>80</td>
<td>tcp</td>
<td>2294</td>
<td>797</td>
</tr>
<tr>
<td>367954285.233317</td>
<td>10.0.2.15</td>
<td>51934</td>
<td>54.243.131.12</td>
<td>80</td>
<td>tcp</td>
<td>578</td>
<td>184</td>
</tr>
<tr>
<td>3.0950611367954281.046495</td>
<td>10.0.2.15</td>
<td>51237</td>
<td>10.1.1.1</td>
<td>53</td>
<td>udp</td>
<td>32</td>
<td>200</td>
</tr>
<tr>
<td>1367954281.980046</td>
<td>10.0.2.15</td>
<td>61472</td>
<td>10.1.1.1</td>
<td>53</td>
<td>udp</td>
<td>33</td>
<td>128</td>
</tr>
<tr>
<td>1367954287.375995</td>
<td>10.0.2.15</td>
<td>55776</td>
<td>74.125.226.214</td>
<td>443</td>
<td>tcp</td>
<td>489</td>
<td>2658</td>
</tr>
<tr>
<td>1367954282.655601</td>
<td>10.0.2.15</td>
<td>52392</td>
<td>72.22.185.207</td>
<td>80</td>
<td>tcp</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1367954282.651031</td>
<td>10.0.2.15</td>
<td>45201</td>
<td>72.22.185.198</td>
<td>80</td>
<td>tcp</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1367954282.654099</td>
<td>10.0.2.15</td>
<td>52391</td>
<td>72.22.185.207</td>
<td>80</td>
<td>tcp</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1367954282.660982</td>
<td>10.0.2.15</td>
<td>34639</td>
<td>72.22.185.214</td>
<td>80</td>
<td>tcp</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1367954282.689652</td>
<td>10.0.2.15</td>
<td>58404</td>
<td>72.22.185.199</td>
<td>80</td>
<td>tcp</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1367954283.208290</td>
<td>10.0.2.15</td>
<td>34880</td>
<td>72.21.91.19</td>
<td>80</td>
<td>tcp</td>
<td>394</td>
<td>313</td>
</tr>
</tbody>
</table>

wt.o.nytimes.com
p.typekit.net
Components from Tensor Decomposition

DNS Traffic

Daily Web Browsing

Surprises?

Nightly Quality Tests

System Update Traffic

Reservoir Network Traffic Tensor

Security Threats?
Reservoir Network Data Analysis
One Factor: SSH Attacks (Component 71 of 120)

Time

12 am to 5 am

Origin IP

Chinese IP’s (all three blacklisted for SSH attacks)

Destination IP

Reservoir Code Repository

Destination port

SSH port

Response bytes

Every request was rejected
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

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

NNZ

Pi

Phi_local

Phi

Pi Computation

Partial Phi Computation

Sync

Final Phi Computation

P1  P2  P3  P4
Improved Parallelization

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

Need to balance the work load among the processors.
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)
Performance Evaluation

<table>
<thead>
<tr>
<th>Tensor</th>
<th>Size</th>
<th>Non-zeros</th>
<th>#iterations timed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook</td>
<td>63891 x 63891 x 1591</td>
<td>737934</td>
<td>190</td>
</tr>
</tbody>
</table>
Performance Evaluation

<table>
<thead>
<tr>
<th>Tensor</th>
<th>Size</th>
<th>Non-zeros</th>
<th>#iterations timed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyber</td>
<td>14811811 x1899 x1899 x 3 x 6067</td>
<td>2085108</td>
<td>50</td>
</tr>
</tbody>
</table>
Performance Evaluation

<table>
<thead>
<tr>
<th>Tensor</th>
<th>Size</th>
<th>Non-zeros</th>
<th>#iterations timed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enron</td>
<td>105 x 105 x 27</td>
<td>5418</td>
<td>180</td>
</tr>
</tbody>
</table>
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)