ENSIGN: High-performance Data Analytics Tool

Scaling and Deepening Tensor Decompositions and Applications using ENSIGN

Muthu Baskaran
James Ezick
Aditya Gudibanda
Thomas Henretty
M. Harper Langston
Pierre-David Letourneau
Richard Lethin
Benoit Meister
Matt Robillard

Reservoir Labs, Inc.
New York, NY

26 January 2019
# Exascale NonStationary Graph Notation (ENSGN)
Driving Towards a Practical High-performance Data Analytics Tool

<table>
<thead>
<tr>
<th>Class</th>
<th>Differentiating Specifics</th>
<th>Benefit to Analyst</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeling (Capability)</td>
<td>First-order decomposition methods&lt;br&gt;Second-order decomposition methods&lt;br&gt;Joint tensor decompositions&lt;br&gt;Multiple data distribution models&lt;br&gt;Normalized decompositions&lt;br&gt;Streaming decompositions&lt;br&gt;... more coming</td>
<td>Breadth of models enabled&lt;br&gt;Framework for graph fusion&lt;br&gt;Platform for anomaly detection&lt;br&gt;Sparsity-maximizing approaches&lt;br&gt;Efficient update with arrival of new data&lt;br&gt;Discovery of new behaviors through new components</td>
</tr>
<tr>
<td>Performance</td>
<td>Optimized sparse tensor data structures&lt;br&gt;Mixed static/dynamic optimization&lt;br&gt;Memory-efficiency optimizations&lt;br&gt;Algorithmic improvements&lt;br&gt;Shared memory parallelism&lt;br&gt;Distributed memory parallelism&lt;br&gt;Cloud-based optimizations</td>
<td>Extend the range, scale, and scope of analysis&lt;br&gt;Analyze tensors of billion-scale and beyond&lt;br&gt;Enable large rank decompositions&lt;br&gt;Enable large number of mode decompositions&lt;br&gt;Leverage HPC Systems&lt;br&gt;Quick time-to-solution</td>
</tr>
<tr>
<td>Usability</td>
<td>GUI &amp; CLI&lt;br&gt;Python bindings&lt;br&gt;C bindings&lt;br&gt;QGIS support&lt;br&gt;Virtual machine distributions&lt;br&gt;Documented, Tested, Supported</td>
<td>Interactive large scale exploration&lt;br&gt;In standard environments (e.g., Jupyter notebooks)&lt;br&gt;Integration with existing corporate data lakes/pipelines&lt;br&gt;Visualization&lt;br&gt;Reliable install and operation&lt;br&gt;Training, Someone to Call</td>
</tr>
</tbody>
</table>
ENSIGN Application Areas

Cyber Security

Bioinformatics

GEOINT

Reservoir Labs  Jan 26, 2019  Invited Workshop on Compiler Techniques for Sparse Tensor Algebra
MODELING (CAPABILITY)
**Generalized CP Streaming Framework**

**Algorithm - Streaming CP update**

- **Input:** $[[\mathbf{A}_{old}^{(n)}]], \mathbf{X}_{new}, K_{new} > 0, 0 < \nu_{sim} \leq 1,$
  $\tau > 0, \tilde{K}$
- **Compute:** $[[\mathbf{A}_{new}^{(n)}]]$ (rank-$K_{new}$ decomp. of $\mathbf{X}_{new}$)
  $[[\mathbf{A}^{(n)}]], \tilde{\mathbf{A}}_{new}^{(N+1)} \leftarrow \text{MERGE} \left( [[\mathbf{A}_{old}^{(n)}]], [[\mathbf{A}_{new}^{(n)}]], \nu_{sim} \right)$
  $\mathbf{A}^{(N+1)} \leftarrow \text{UPDATE} \left( [[\mathbf{A}^{(n)}]], \tilde{\mathbf{A}}_{new}^{(N+1)} \right)$
  $\{C_1, C_2, C_3\} \leftarrow \text{CLASSIFY} \left( [[\mathbf{A}^{(n)}]], K, K_{old}, \tau \right)$
  $[[\mathbf{A}^{(n)}]], S_{trunc} \leftarrow \text{TRUNCATE} \left( [[\mathbf{A}^{(n)}]], K, \tilde{K} \right)$
- **Output:** $[[\mathbf{A}^{(n)}]], \{C_1, C_2, C_3\}, S_{trunc}$

---

**Highlights/Differentiators**
- **Low-cost** computations (of the order of size of streaming data streams)
- Extraction of “new information” entirely present in the new data streams
- Unified framework across different CP decompositions

---

Real-world Cyber Application

... Evolution of the attack seen with streaming decompositions

State of the activity at 9am

State of the activity at 11am

State of the activity at 1pm

State of the activity at 2pm
PERFORMANCE
ENSIGN Data Structures

Highlights
• Compressed sparse tensor storage
• Mode-generic and mode-specific formats*

Key differentiators
• Applies to all tensor decomposition methods
• Supports a spectrum of tensors within the formats
  – From extremely sparse to partially dense to fully dense tensors
• Enables computation and memory reduction (from compression)
• Enables improved parallelism (from data structure arrangement)

Performance Optimizations

Highlights

- Distributed-memory (MPI) optimizations
- Shared-memory (OpenMP) optimizations*
- Cloud-based (Spark) optimizations
- Memory- and operation-efficient tensor operations
  - Building blocks for newer capabilities

USABILITY
Python Bindings & Jupyter Notebook

```python
import ensign.cp.decomp as cpd
import ensign.sptensor as spt

# Parameters
rank = '100'
sptensor_file = 'tensor data.txt'

# Load tensor & decompose
the_tensor = spt.read_sptensor_file(sptensor_file)
als_decomp = cpd.cp.als(the_tensor, rank, "als_save_dir")
apr_decomp = cpd.cp.apr(the_tensor, rank, "apr_save_dir")
pdnr_decomp = cpd.cp.pdnr(the_tensor, rank, "pdnr_save_dir")

als_weights = pd.Series(als_decomp.weights)
apr_weights = pd.Series(apr_decomp.weights)
pdnr_weights = pd.Series(pdnr_decomp.weights)

ax = als_weights.plot(color='orange', logy=True, label='als', legend=True)
ax = apr_weights.plot(color='magenta', logy=True, label='apr', ax=ax, legend=True)
ax = pdnr_weights.plot(color='blue', logy=True, label='pdnr', ax=ax, legend=True)
```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x7f29bd28fe98>


