SMAT: An Input Adaptive Auto-Tuner for Sparse Matrix-Vector Multiplication

Jiajia Li, Guangming Tan, Mingyu Chen, Ninghui Sun

Institute of Computing Technology,
Chinese Academy of Sciences

6/17/2013
High Performance Computational Software Development

“2–8” Principle

Application

Productivity Layer (20%)

Efficiency Layer (80%)

Scientific Applications (Hypre, PETSc, Trilinos, etc.)

High Performance Math Lib. (Intel MKL, AMD ACML, etc.)

80% Programmers

High Productivity

20% Programmers

High Performance

High Usability

Performance Portability

Percentage of execution time
High Performance Computational Software Development

Hand-tuning

Application

Algorithm

Data Structure

Program

Architecture

Autotuning

Yesterday

Today

Tomorrow

Co-Autotuning
Killer Applications

Computational Science
ITER
Climate
Oil

Exascale

Big Data

Data Science
Social network
National Security
System Biology

Sparse Linear System

Sparse Matrices
Protein
FEM / Spheres
FEM / Cantilever
Wind Tunnel
FEM / Harbor
QCD
FEM / Ship
Economics
Epidemiology
FEM / Accelerator
Circuit
webbase

ITER
Climate
Oil

Computational Science

Exascale

Big Data

Data Science
Social network
National Security
System Biology

Sparse Linear System

Sparse Matrices
Protein
FEM / Spheres
FEM / Cantilever
Wind Tunnel
FEM / Harbor
QCD
FEM / Ship
Economics
Epidemiology
FEM / Accelerator
Circuit
webbase
Sparse Matrix

- Diverse Application Background
- Different Solving Methods

### Diff. Nonzero Distribution Structure

#### Kinds of Sparse Matrices

- Diagonal Matrix
- “Slim” Matrix
- “Fat” Matrix
- Power-Law Matrix
- Matrix with Dense Blocks
- ...

#### Storage Formats

**SpMV:** solve \( Y = AX + Y \), where \( A \) is a sparse matrix, \( X \) and \( Y \) are dense vectors.

- **CSR SpMV**
  - `row`: \[ [0 \, 0 \, 1 \, 1 \, 2 \, 2 \, 3 \, 3] \]
  - `col`: \[ [0 \, 1 \, 1 \, 2 \, 0 \, 2 \, 3 \, 1] \]
  - `data`: \[ [1 \, 5 \, 0 \, 0 \, 0 \, 2 \, 6 \, 0] \]
  - `sum x[indices] + data[i]`, \( y[i] = \text{sum} \)

- **COO SpMV**
  - `offsets`: \[-2 \, 0 \, 1\]
  - `data`: \[ [1 \, 5 \, 0 \, 0 \, 0 \, 2 \, 6 \, 0] \]
  - `row[i] = data[i+rowstrides[i]]` for \( i = 0 \) to \( N \)

- **DIA SpMV**
  - `indices`: \[ [0 \, 1 \, 1 \, 0 \, 1 \, 2 \, 1 \, 3 \, 1 \, 5 \, 2 \, 6 \, 8 \, 3 \, 7 \, 9 \, 4 \] \]
  - `data`: \[ [1 \, 5 \, 0 \, 0 \, 0 \, 2 \, 6 \, 0] \]
  - `y[i] = data[indices[i]*stride + i] * x[indices[i]*stride + i]` for \( n = 0 \) to \( N \)

- **ELL SpMV**
  - `rows`: 
  - `cols`: 
  - `data`: 

---

Diagonal
“pcrystk02”

“Slim”
“Bfly”

“Fat”
“crankseg_2”

Power-Law
“roadNet-CA”

number of nonzeros per row
Sparse solvers mainly use ONE storage format CSR
- Hypre (LLNL), PETSc (ANL), Trilinos (SNL)

Libraries provide complicated interfaces to users
- MKL (Intel), OSKI (UCB), SpBLAS (UTK)

Low Performance
- Low Productivity

GAP!

High Performance
- High Productivity

\[(\text{csr is the only one exposed to users})\]

\[
\text{smat}_\text{\langle \text{precision} \rangle}_\text{\langle \text{csr} \rangle}_\text{\langle \text{operation} \rangle}(())
\]

\[
\text{mkl}_\text{\langle \text{precision} \rangle}_\text{\langle \text{format} \rangle}_\text{\langle \text{operation} \rangle}(())
\]

(variants of format)
Motivation cont.

Observation 1: The optimal formats are diverse for sparse matrices from different application areas.
Observation 2: Different formats are needed in different stages of one application during runtime.

Performance Gap: 10x!
Application-Architecture Aware Auto-tuner Design

**Offline**
- Extract application features
- Determine feature values for learning set
- Build feature database (including the optimal algorithm and implementation)
- Summarize the rules and build a model
- Choose the optimal implementation with architecture characteristics

**Online**
- Extract parameter values of the input matrix
- Predict the optimal algo. & impl. based on model

---

**Sparse App.**
- Matrix Dimension
- Diagonal Situation
- Nonzero Distribution
- Nonzero Fill Ratio

**Graph App.**
- Dimension
- Degree Distribution
- Diameter
- Un-/directed
- Power-Law
- Connectivity

---

**Diagram**
- Offline
  - Application
  - Feature Database
  - Model
- Online
  - Architecture
  - Various Implementations
  - Optimal Implementation
- Thin arrows represent data flow between application and architecture.
SMAT Framework

Example
- TLB size
- Cache size
- Reg. Size
- Prefetch
- SIMDization
- Branch
- Multi-thread

Example
- Matrix Dim.
- Diagonal Situation
- Nonzero Ratio
- Nonzero Dist.
- Power-Law

Offline
- Kernel Library
- Search
- Optimal Kernel
- Exec. & Measure
- Feature Extraction
- Feature Database
- Determine Parm. Value
- Data Mining (C5.0)
- Link
- < TH
- Optimal SpMV
- SMAT_xCSR_SpMV

Online

Optimal Kernel Learning Model

SMAT_xCSR_SpMV
Intel MKL

- mkl_xcsrgemv
- mkl_xxiagemv
- mkl_xbsrgemv
- mkl_xcscmv
- mkl_xcoogemv
- mkl_xskymv

SMAT

- SMAT_xCSR_SpMV

So Easy!
Matrix Feature Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Formula</th>
<th>DIA</th>
<th>ELL</th>
<th>CSR</th>
<th>COO</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Matrix Dimension</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>the number of rows</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>the number of columns</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ndiags</td>
<td>the number of diagonals</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NTdiags_ratio</td>
<td>the ratio of “true” diagonals to total diagonals</td>
<td>$NTdiags_ratio = \frac{\text{number of “true diagonals”}}{Ndiags}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Diagonal Situation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NNZ</td>
<td>the number of nonzeros</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aver_RD</td>
<td>the number of nonzeros per row</td>
<td>$\text{aver}_RD = \frac{NNZ}{M}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>max_RD</td>
<td>the maximum number of nonzeros per row</td>
<td>$\text{max}_RD = \max_1^M {\text{number of nonzeros per row}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>var_RD</td>
<td>the variation of the number of nonzeros per row</td>
<td>$\text{var}_RD = \frac{\sum_1^M \left(\text{row degree} - \text{aver}_RD\right)^2}{M}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Nonzero Distribution</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ER_DIA</td>
<td>the ratio of nonzeros in DIA data structure</td>
<td>$ER_DIA = \frac{NNZ}{N\times M}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ER_ELL</td>
<td>the ratio of nonzeros in ELL data structure</td>
<td>$ER_EELL = \frac{NNZ}{\text{max}_RD\times M}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Power-Law</strong></td>
<td>a factor of power-law distribution</td>
<td>$P(k) \sim k^{-R}$</td>
<td></td>
<td></td>
<td></td>
<td>1.4</td>
</tr>
</tbody>
</table>

“√” shows the parameter is useful for all formats.
“↑/↓” indicates the larger (smaller) the parameter value is, the format shows more benefit.
Divide the matrix set based on the optimal storage format, and measure SpMV performance on them.

Draw the value distribution graph to each parameter, and find the regulations.

### Matrix Partition

<table>
<thead>
<tr>
<th></th>
<th>DIA_best</th>
<th>ELL_best</th>
<th>CSR_best</th>
<th>COO_best</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of Sub-set</td>
<td>206</td>
<td>169</td>
<td>1496</td>
<td>507</td>
</tr>
</tbody>
</table>

**Perf. Beneficial values**

- Ndiags with small value
- DIA format earns most benefit

- "DIA_best" Matrix Set

- [1,4]
SMAT—Choosing Optimal Implementation

**Scoreboard Strategy**

- Choose typical matrix set to test each algo. and impl.
- Record the performance value on scoreboard
- The optimal impl. for each algo. are recorded in index table

---

**Diagram:**

1. **SpMV Optimized Kernels**
2. **Run**
3. **Impl.**
4. **Perf. Value**
5. **Choose the Highest Perf.**
6. **Scoreboard**
7. **Index of Optimal Impl.**
Belong to data mining problems

- Classification problem
- Target Attribute: “Best_format”

\[
f(\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n, \overline{TH}) \rightarrow C_n(DIA, ELL, CSR, COO)
\]

\(\bar{x}_i\): value of each record, \(\overline{TH}\): threshold of each parameter, 
\(C_n(DIA, ELL, CSR, COO)\): one of the four formats.

Build model

- Use ruleset to represent model
- Add rule confidence
SMAT—Data Mining

Data Mining Process

Classification Algorithm

Rules

IF Ndiags <= 39
AND ER_DIA > 0.3
AND ER_ELL <= 0.9
THEN
Best_format = DIA

<table>
<thead>
<tr>
<th>M</th>
<th>N</th>
<th>Ndiags</th>
<th>Ntdiags_ratio</th>
<th>...</th>
<th>Best_format</th>
</tr>
</thead>
<tbody>
<tr>
<td>36476</td>
<td>36476</td>
<td>1328</td>
<td>0.001</td>
<td></td>
<td>ELL</td>
</tr>
<tr>
<td>23560</td>
<td>23560</td>
<td>33</td>
<td>0.515</td>
<td></td>
<td>DIA</td>
</tr>
<tr>
<td>12M</td>
<td>12M</td>
<td>1.8M</td>
<td>~0</td>
<td></td>
<td>COO</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
SMAT -- Online Procedure

Choose Format

- DIA ruleset
- ELL ruleset
- CSR ruleset
- COO ruleset

Format Confidence

0~1

Rule Confidence

Format Confidence

> TH

Exec. & Measure

Optimal SpMV

SMAT_xCSR_SpMV

Extract Feature

Learning Model
Platform and Matrix Set

Platforms:
- Intel Xeon X5680
- AMD Opteron 6168

Matrix set: The University of Florida Sparse Matrix Collection
- Learning Set: 2055
- Testing Set: 331, represented by 16 matrices
Performance

◆ SMAT Auto-tuning
◆ Optimized SpMV kernels
  ▪ Assembling opt.
  ▪ Loop unrolling
  ▪ SIMDization
  ▪ Multi-threading on task level
    ▪ Allocate a sub-block to each thread
    ▪ Independently choose the optimal algo. & impl. on each sub-block

Compared with MKL

~3X Speedup on average
Analyze the prediction procedure and accuracy on 16 representative matrices

Overhead
- When the model predicts right, small overhead (about 2 CSR-SpMV)
- Wrong predict, execute & measure mudule used, the overhead is more than 10 CSR-SpMV (OSKI:40+; clSpMV: ~10)
- When a matrix is used hundreds of times in an iterative method, the overhead can be overlapped.

<table>
<thead>
<tr>
<th>Matrix Number</th>
<th>Matrix Name</th>
<th>Model Prediction Format</th>
<th>Execution</th>
<th>SMAT Prediction Format</th>
<th>Actual Best Format</th>
<th>Model Accuracy</th>
<th>SMAT Overhead (times of CSR-SpMV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pcrystk02</td>
<td>DIA</td>
<td>-</td>
<td>DIA</td>
<td>DIA</td>
<td>R</td>
<td>2.28</td>
</tr>
<tr>
<td>2</td>
<td>denormal</td>
<td>DIA</td>
<td>-</td>
<td>DIA</td>
<td>DIA</td>
<td>R</td>
<td>2.09</td>
</tr>
<tr>
<td>3</td>
<td>cryg10000</td>
<td>DIA</td>
<td>-</td>
<td>DIA</td>
<td>DIA</td>
<td>R</td>
<td>2.11</td>
</tr>
<tr>
<td>4</td>
<td>apache1</td>
<td>DIA</td>
<td>-</td>
<td>DIA</td>
<td>DIA</td>
<td>R</td>
<td>1.94</td>
</tr>
<tr>
<td>5</td>
<td>bfly</td>
<td>ELL</td>
<td>-</td>
<td>ELL</td>
<td>ELL</td>
<td>R</td>
<td>1.18</td>
</tr>
<tr>
<td>6</td>
<td>whitaker3_dual</td>
<td>ELL</td>
<td>-</td>
<td>ELL</td>
<td>ELL</td>
<td>R</td>
<td>4.89</td>
</tr>
<tr>
<td>7</td>
<td>ch7-9-b3</td>
<td>ELL</td>
<td>-</td>
<td>ELL</td>
<td>ELL</td>
<td>R</td>
<td>2.25</td>
</tr>
<tr>
<td>8</td>
<td>shar_te2-b2</td>
<td>ELL</td>
<td>-</td>
<td>ELL</td>
<td>ELL</td>
<td>R</td>
<td>2.24</td>
</tr>
<tr>
<td>9</td>
<td>pkustk14</td>
<td><strong>confidence &lt; TH</strong></td>
<td>CSR+COO</td>
<td>CSR</td>
<td>CSR</td>
<td>W</td>
<td>16.39</td>
</tr>
<tr>
<td>10</td>
<td>crankseg_2</td>
<td><strong>confidence &lt; TH</strong></td>
<td>CSR+COO</td>
<td>CSR</td>
<td>CSR</td>
<td>W</td>
<td>16.28</td>
</tr>
<tr>
<td>11</td>
<td>GaAs3H12</td>
<td><strong>confidence &lt; TH</strong></td>
<td>CSR+COO</td>
<td>CSR</td>
<td>CSR</td>
<td>W</td>
<td>16.2</td>
</tr>
<tr>
<td>12</td>
<td>HV15R</td>
<td><strong>confidence &lt; TH</strong></td>
<td>CSR+COO</td>
<td>CSR</td>
<td>CSR</td>
<td>W</td>
<td>15.43</td>
</tr>
<tr>
<td>13</td>
<td>europe_osm</td>
<td>COO</td>
<td>-</td>
<td>COO</td>
<td>COO</td>
<td>R</td>
<td>2.3</td>
</tr>
<tr>
<td>14</td>
<td>D6-6</td>
<td>COO</td>
<td>-</td>
<td>COO</td>
<td>COO</td>
<td>R</td>
<td>5.79</td>
</tr>
<tr>
<td>15</td>
<td>dictionary28</td>
<td>COO</td>
<td>-</td>
<td>COO</td>
<td>COO</td>
<td>R</td>
<td>2.05</td>
</tr>
<tr>
<td>16</td>
<td>roadNet-CA</td>
<td>COO</td>
<td>-</td>
<td>COO</td>
<td>COO</td>
<td>R</td>
<td>2.38</td>
</tr>
</tbody>
</table>

“R” and “W” represent Right and Wrong prediction respectively.
Algebraic Multi-grid Algorithm

- An iterative algorithm to solve linear equations $Au=f$, where $A$ is a sparse matrix, $u$, $f$ are dense vectors.
- As a pre-conditioner applied in applications such as laser fusion and climate modeling.

SpMV the critical operation of AMG, takes 90% execution time.

<table>
<thead>
<tr>
<th>Coarsen</th>
<th>Rows</th>
<th>Hypre AMG</th>
<th>SMAT AMG</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>cljp_7pt_50</td>
<td>125k</td>
<td>3034</td>
<td>2487</td>
<td>1.22</td>
</tr>
<tr>
<td>rugel_9pt_500</td>
<td>250k</td>
<td>388</td>
<td>300</td>
<td>1.29</td>
</tr>
</tbody>
</table>

- Relax is a relaxation algorithm, such as Jacobi, G-S iterative methods.
- $A$, $P$ are sparse matrices.
- $U$, $f$, $e$ are dense vectors.
SMAT Many-core

Performance

**Accuracy**
- For 289 testing matrices: 89% (single), 95% (double)
SMAT Summary

◆ Develop auto-tuning method
  ■ Design application-architecture aware SpMV auto-tuner.
  ■ Develop auto-tuning method to algorithm level

◆ Introduce data mining to auto-tuning method
  ■ Reinforce its usability and extensibility

◆ API Easy-to-use
  ■ Unify the interface

◆ Increase SpMV performance using SMAT

◆ Apply SMAT to AMG, and extend it to many-core architecture
Future Work

- Extend storage formats
- Combine other auto-tuners
Thank You!

Question?


Jiajia Li, Guangming Tan, Mingyu Chen, Ninghui Sun