The Netflix Prize

Carl Nygaard
Netflix Rating System

Rate Movies

Rate More Movies in All Genres!

Keep rating movies to get recommendations. You can rate movies you've seen in the theater as well as movies you've added from Netflix. Click the star that matches your opinion.

Ratings Legend:
- ★★★★★: Loved it
- ★★★★: Really Liked it
- ★★★: Liked it
- ★★: Didn't Like it
- ★: Hated it

Movies:
- The Longest Yard: Not Interested
- Van Helsing: Not Interested
- John Q: Not Interested
- Taxi Driver: Not Interested
- Collateral Damage: Not Interested
- Citizen Kane: Not Interested
- Big Momma's House: Not Interested
- Broken Arrow: Not Interested

Add
<table>
<thead>
<tr>
<th></th>
<th><em>Psycho</em></th>
<th><em>Manos: The hands of Fate</em></th>
<th><em>Vertigo</em></th>
<th><em>Jaws</em></th>
<th><em>Night of the Living Dead</em></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ann</strong></td>
<td>4</td>
<td>1</td>
<td></td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td><strong>Bob</strong></td>
<td>2</td>
<td>5</td>
<td>1</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td><strong>Carl</strong></td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>?</td>
</tr>
</tbody>
</table>
My Approach – Neural Networks

<table>
<thead>
<tr>
<th>Training Input Vectors</th>
<th>Ann</th>
<th>4</th>
<th>1</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Operating Input</td>
<td>Carl</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

Training Output Ratings
- Ann: 5
- Bob: 2

Operating Output
- Carl: $\sim5$
**Implementation**

- **Challenges**
  - **Volume of Training Data**
    - 500k users
    - 18k movies
    - 100 million ratings
  - **Required Predictions:** 2.8 million

- **Used C++ for**
  - Speed & memory management
  - Data load time (memory-mapped binary data)

- **Used MPI for parallelization**
Parallelization

2.8 million independent subtasks run on n slaves

Master Node
Data indices
Task list

Slave Node
“Brain” generator

Single rating
learning data

Predicted Rating
Load Balancing

- **Problems**
  - Computation time varies between tasks
  - We cannot just give n/p tasks to each processor

- **Solution**
  - Clients request tasks
  - Tasks are assigned a number
  - Master saves the predictions to disk
  - Priority queue assures results stay in order
  - Results are stored directly to disk so crashed runs can be easily recovered
Speedup

Computation cost $>>$ data lookup cost

Computation cost $\approx$ data lookup cost
Results – Leaders as of 12/3/2006

Leaderboard shows top 40 submitters

Used paper on SVD with “substantial re-derivation”

Benign Solution: 1.0540

Two week ago, my week-long computation gave only 1.32

Carl

1.0932

-16.22
Questions?