Patterns and Computer Vision

Michael Anderson, Bryan Catanzaro, Jike Chong, Katya Gonina, Kurt Keutzer, Tim Mattson, Mark Murphy, David Sheffield, Bor-Yiing Su, Narayanan Sundaram

...and the rest of the PALLAS group
How NOT to write parallel programs

Initial Code

Profile

Add threads

Fast enough

Not fast enough

Ship it!

Parallel code is slower?!
What is this person thinking of?

Add threads

Threads, locks, semaphores, mutexes

Edward Lee, “The Problem with Threads”
Architecting parallel software

Specification → Architect SW for Parallelism using Patterns → Map SW Arch to HW Arch → Profile → Not fast enough → Fast enough → Ship it!
What is this person thinking of?

- Today’s code is serial, but the world is parallel
  - Find the inherent parallelism, express it in software, map it to parallel hardware

Re-architect with patterns

- Linear Algebra
- Graph algorithms
- Spectral
- FSMS
- Stencil
Applying this methodology

- We developed a pattern language to help people learn how to successfully architect parallel programs
- We applied this methodology in several areas

  - Computer Vision
    - Support Vector Machine Training & Classification
    - High-Quality Image Contour Detection
      - Object Recognition
      - Optical Flow & Point Tracking
      - Poselet based recognition and pose estimation
  
  - MRI Reconstruction for Compressed Sensing
  - Speech Recognition
  - Computational Finance
Parallelism in action

- Berkeley professor Jitendra Malik approached us:
  
  "My new contour detector is awesome, but people can’t use it: it’s too slow"
  
- A collaborative effort at the ParLab made it 100X faster
  - Rethink algorithms using patterns
  - Efficient parallel implementation

3.7 mins to 1.8 seconds
Image Contour Detection

- High quality contours
  - Provide high quality segments
  - And high quality classification
- But they are expensive to compute
Image Contours

- Contours are subjective – they depend on personal perspective
- Surprise: Humans agree (more or less)
- We can compare against human “ground truth”
**gPb Algorithm**

- **global Probability of boundary**
- Currently, the most accurate image contour detector
- 3.7 mins per small image (0.15 MP) limits applicability
  - N billion images on web
  - Large computer cluster would take several years to find their contours
  - By then we’d have another 3N images to process...

Maire, Arbelaez, Fowlkes, Malik, CVPR 2008
gPb Computation Outline

- Image
  - Convert Colorspace
    - Filter bank Convolution
      - K-means
        - Histogram
          - SGEMM
          - Relabel
    - Integral Images
      - Gradients
    - Filter bank Convolution
    - Combine
      - Non-max suppression
  - Intervening Contour
    - Generalized Eigensolver
      - SpMV
    - Form Eigenvectors
      - Filter bank Convolution
      - SGEMM
    - Combine, Normalize
      - Skeletonize
        - Contours
Textons: Kmeans

- Textures are analyzed in the image by finding textons
- The image is convolved with a filter bank
- Responses to the filter bank are clustered using k-means clustering

Parallel implementation: use BLAS, parallelize histogram construction
Gradients

- An edge detector designed to filter out noise
- For every pixel, and for a given scale and orientation, compare half-discs centered at that pixel
- If half-discs are different, response is high (edge detected)

- Algorithmic transformation: Use integral images to accelerate the analysis of all the half-discs
Integral Images

- Integral images are used to reduce algorithmic complexity

  Input image

  Integral image $O(n^2)$

  Input convolved with constant rectangle $O(n^2r^2) \Rightarrow O(n^2)$

- Saves redundant summation in gradient calculation
- Implement using parallel-prefix sums (scans)
Spectral Graph Partitioning

- Spectral methods avoid creating small, isolated regions
- An affinity matrix links each pixel to its local neighbors
- Interconnected local neighborhoods make a global system
- This step was the most computationally dominant for the serial implementation
Eigenproblem: we only need the smallest $k$ eigenvectors

Algorithmic transformation, suggested by ParLab faculty Jim Demmel:

- Lanczos algorithm with the Cullum-Willoughby test
## Performance Results

<table>
<thead>
<tr>
<th>Computation</th>
<th>Original C++</th>
<th>C + Pthreads (8 threads, 2 sockets)</th>
<th>Damascene (GTX280)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Textons</td>
<td>8.6</td>
<td>1.35</td>
<td>0.152</td>
</tr>
<tr>
<td>Gradients</td>
<td>53.8</td>
<td>12.92</td>
<td>0.75</td>
</tr>
<tr>
<td>Intervening Contour</td>
<td>6.3</td>
<td>1.21</td>
<td>0.03</td>
</tr>
<tr>
<td>Eigensolver</td>
<td>151.0</td>
<td>14.29</td>
<td>0.81</td>
</tr>
<tr>
<td>Overall</td>
<td>222 seconds</td>
<td>29.79 seconds</td>
<td>1.8 seconds</td>
</tr>
</tbody>
</table>

### gPb: CVPR 2008

- **Textons**: 8.6
- **Gradients**: 53.8
- **Intervening Contour**: 6.3
- **Eigensolver**: 151.0
- **Overall**: 222 seconds

### Pthreads

- **Textons**: 1.35
- **Gradients**: 12.92
- **Intervening Contour**: 1.21
- **Eigensolver**: 14.29
- **Overall**: 29.79 seconds

### GTX280

- **Textons**: 0.152
- **Gradients**: 0.75
- **Intervening Contour**: 0.03
- **Eigensolver**: 0.81
- **Overall**: 1.8 seconds
Our results just as accurate
1.8 seconds instead of 3.7 minutes
Speedup came from:
- Rethinking algorithms
- Parallel implementation
3500 downloads
Further work built on this to create image classifiers
Wrapping up

- Contour detection is only one example of the many computations we successfully parallelized.
- Today we’ll also mention:
  - compressed sensing MRI reconstruction
  - Gaussian mixture modeling (PyCASP)
  - Optical flow
- …and there are many more.
- Rearchitectureing applications using patterns helped us make parallelism practically useful.

Parallelism enables new applications.