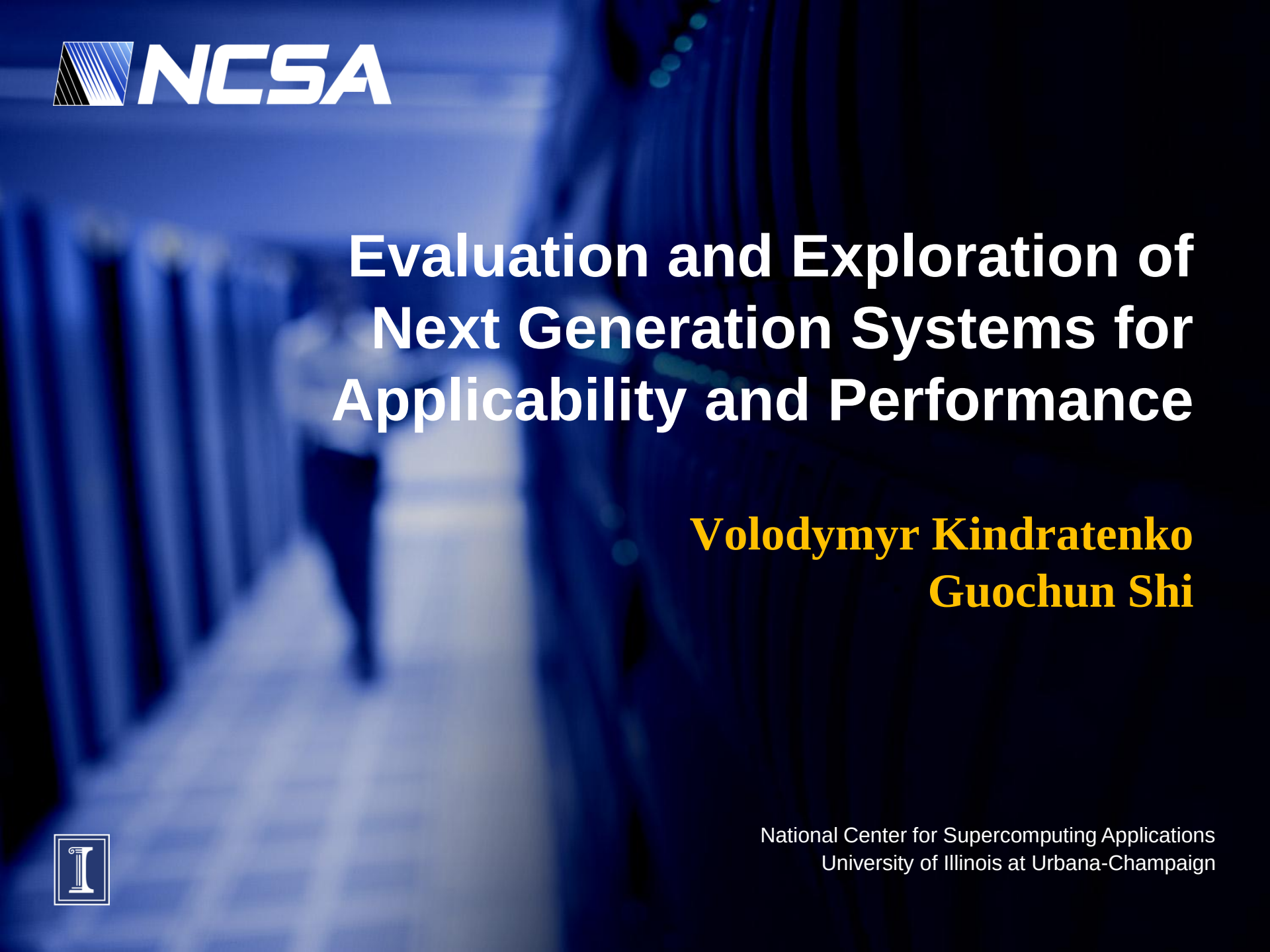


Evaluation and Exploration of Next Generation Systems for Applicability and Performance

Volodymyr Kindratenko
Guochun Shi

National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign

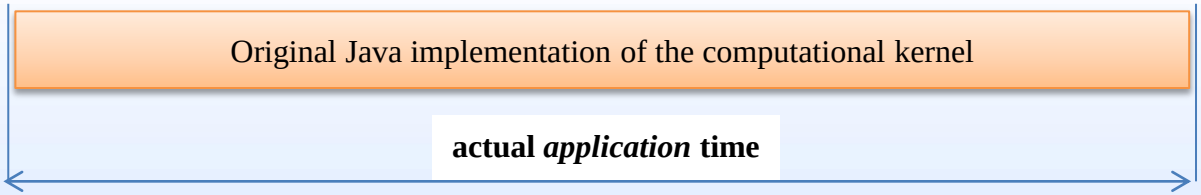
Plan of work for Q3

- **Within the doc2learn framework**
  - re-implement image processing algorithm for execution on the host CPU and GPUs in C, CUDA C, and OpenCL
    - Q: Why only image processing?
    - A: this is deemed to be the most compute- and data-intensive task in doc2learn
  - study performance
  - quantify potential benefits
  - develop plans for the follow-up work

doc2learn image characterization algorithm

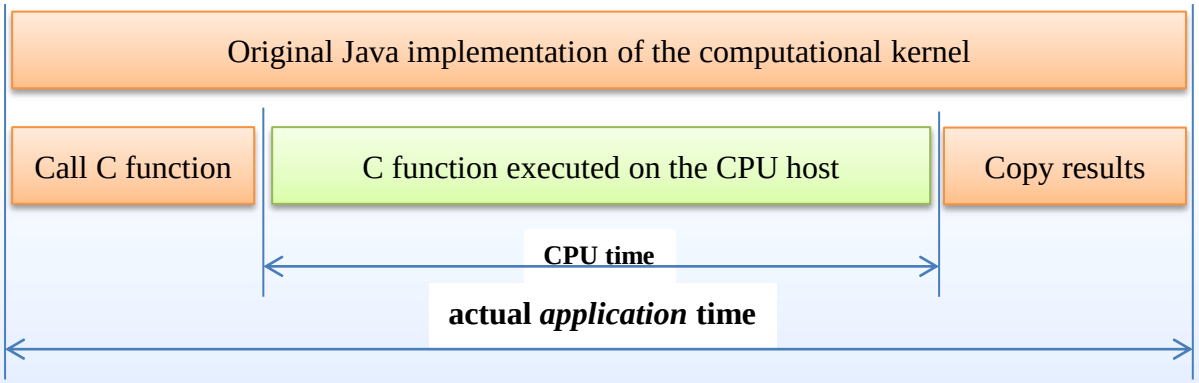
- **Computes probability density function (basically histogram) for images**
  - (Optimized) Java implementation

```
int red, green, blue;
byte[] data = ((DataBufferByte)bi.getRaster().getDataBuffer()).getData();
for (int i =0; i < data.length; i+=3){
    red =(data[i] & 0xff) / size;
    green = (data[i+1] & 0xff)/ size;
    blue = (data[i+2] & 0xff) / size;
    histogram[red][green][blue]++;
}
```



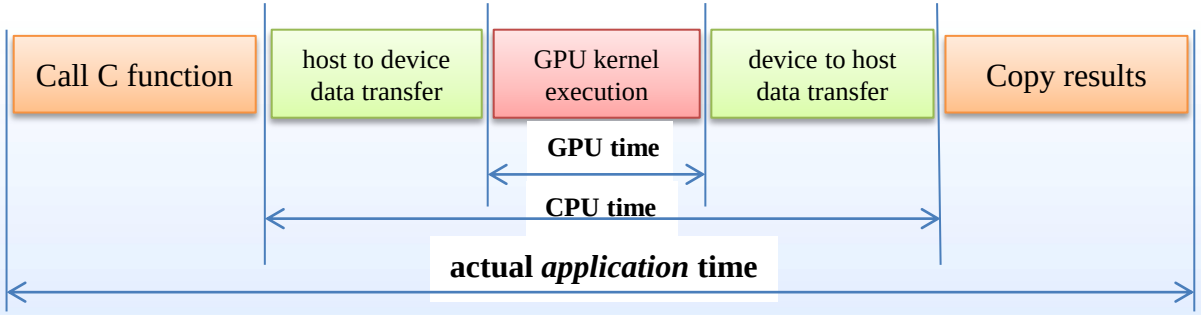
C implementation for Intel/AMD host

- **Java calls C function (using JNI interface)**
  - C function copies data and does the work
- **Java function collects results afterwards**

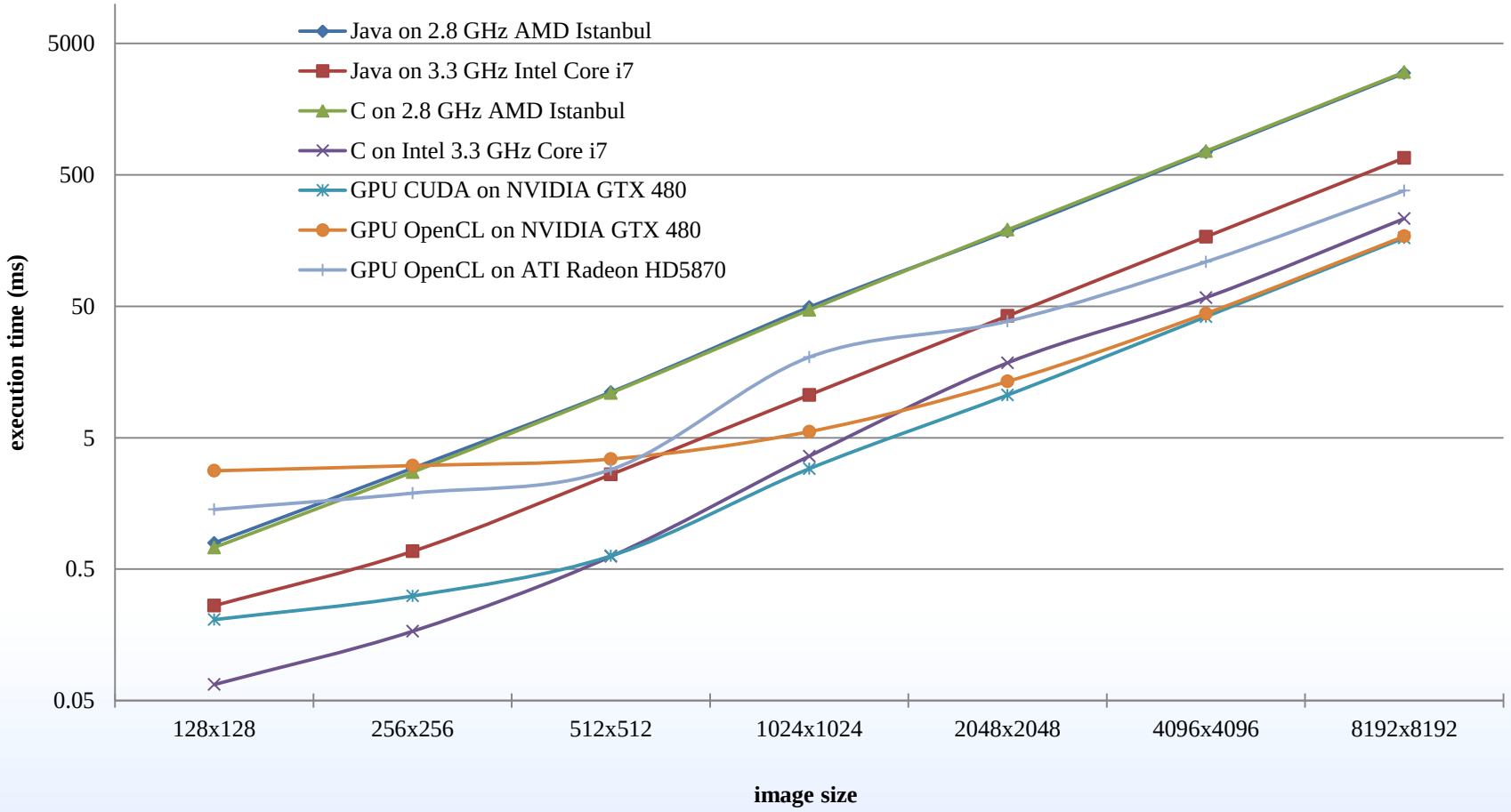


CUDA C/OpenCL Implementation for NVIDIA and AMD GPUs

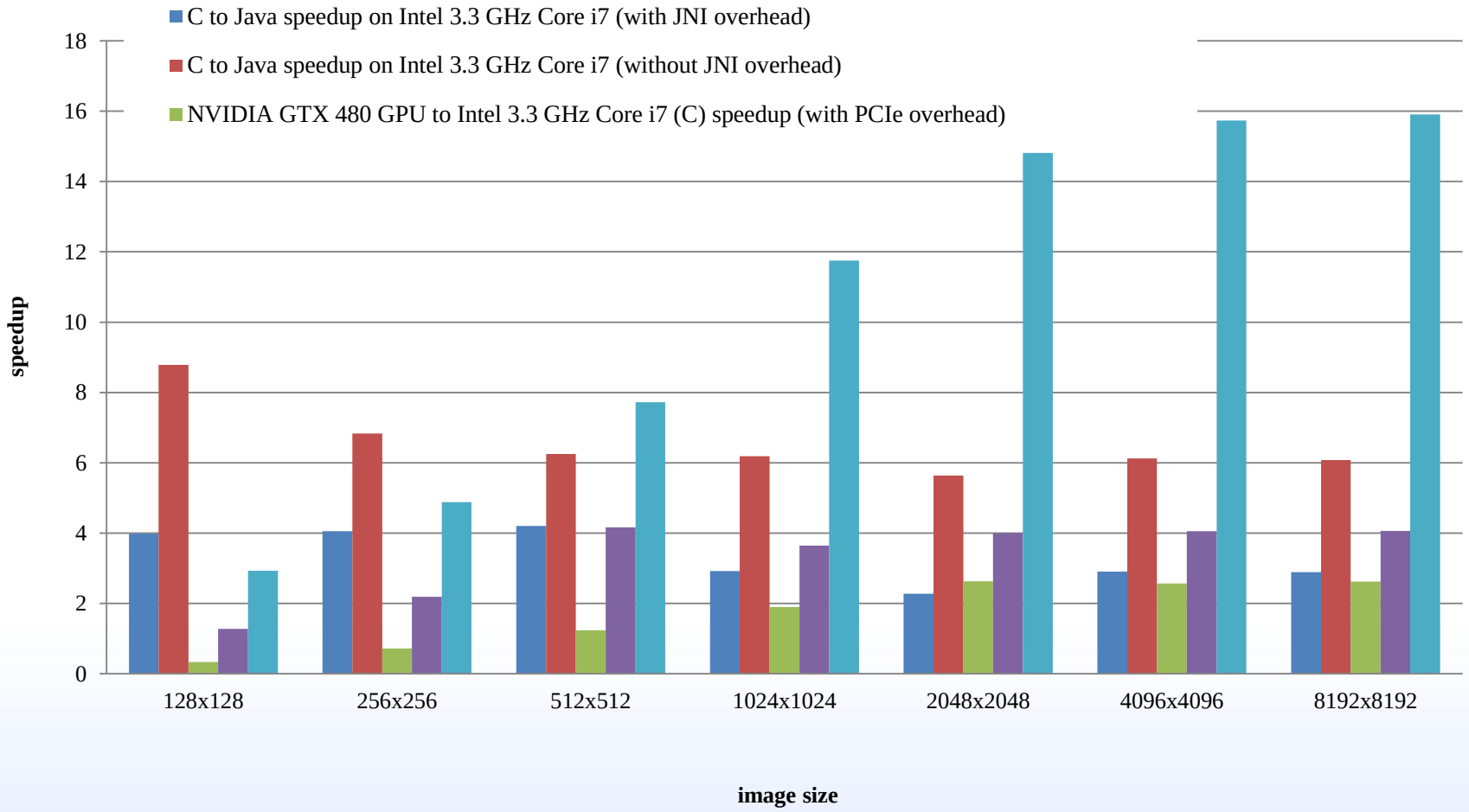
- **Java calls C function**
  - C function copies data from Java VM
  - C function copies data to the GPU memory and calls a GPU kernel
    - GPU computes
  - C function copies data from GPU memory
- **Java function collects results afterwards**



Stand-alone test for varying image size (all overheads included)

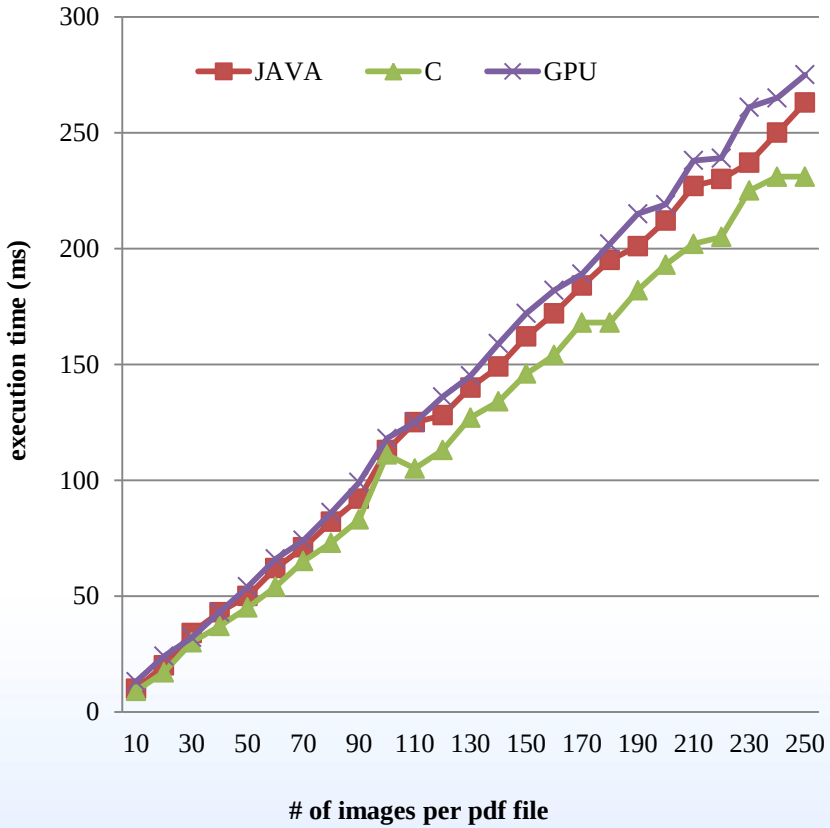


Speedups
(with and without overheads)

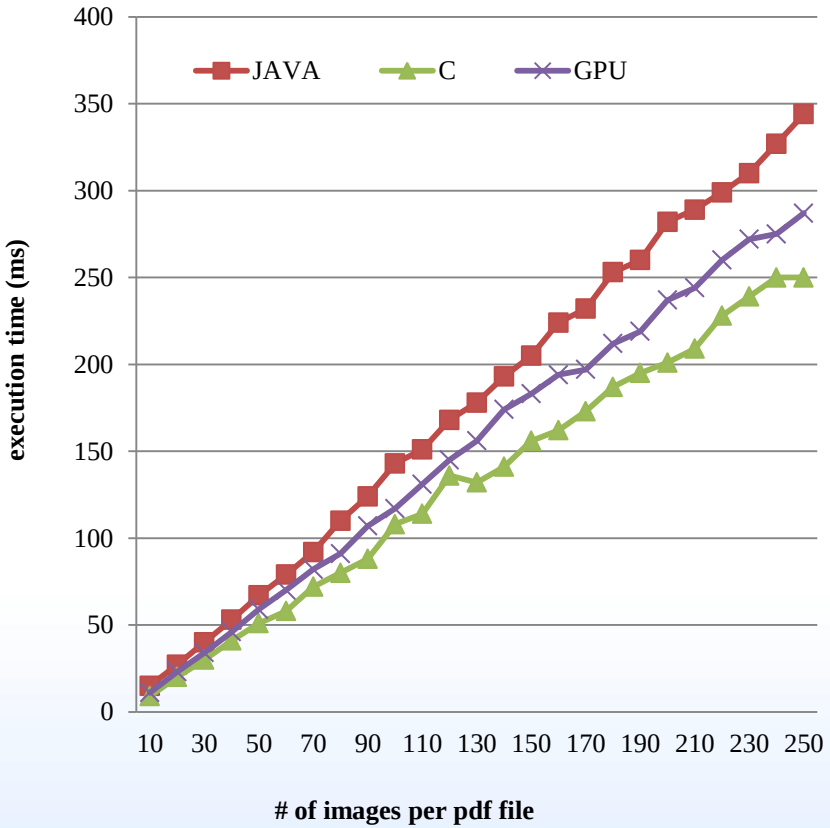


Synthetic dataset test

100x100 pixels images

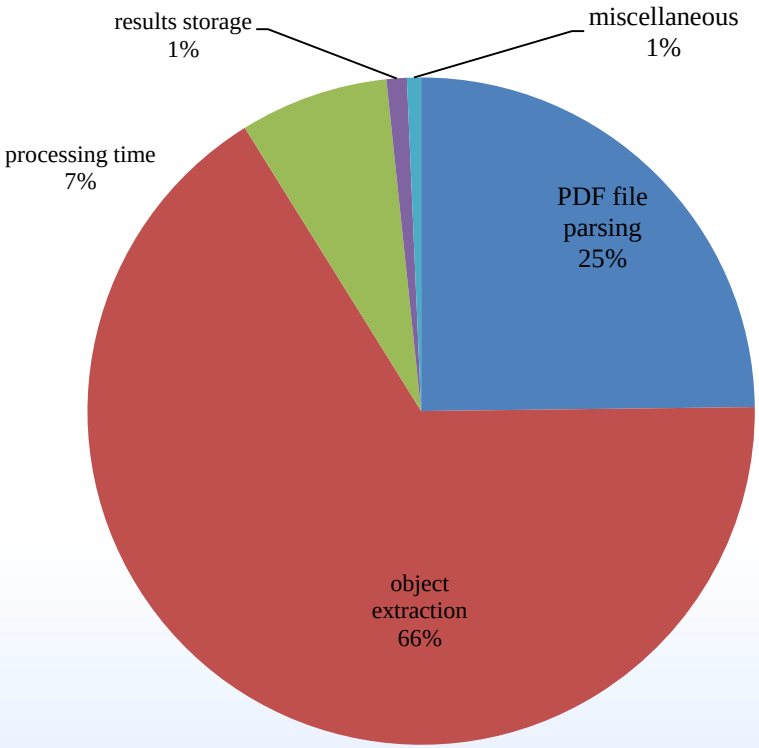


200x200 pixels images

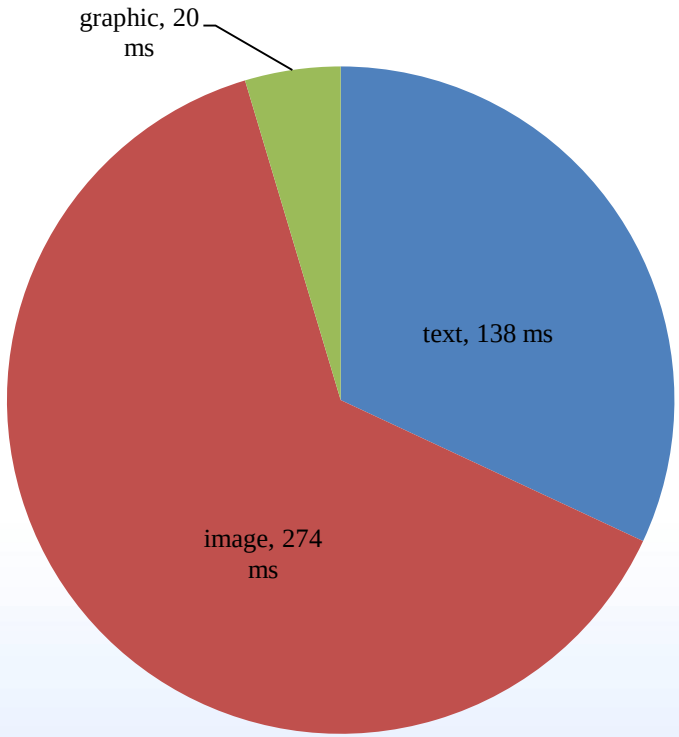


Does it really matter? (Java only example run)

Entire application profile

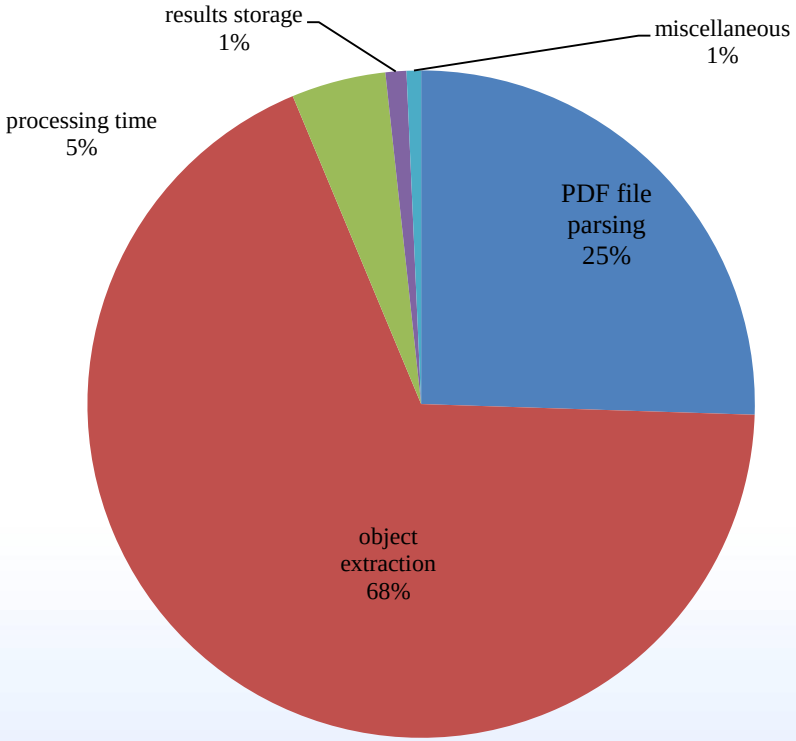


Data analysis profile

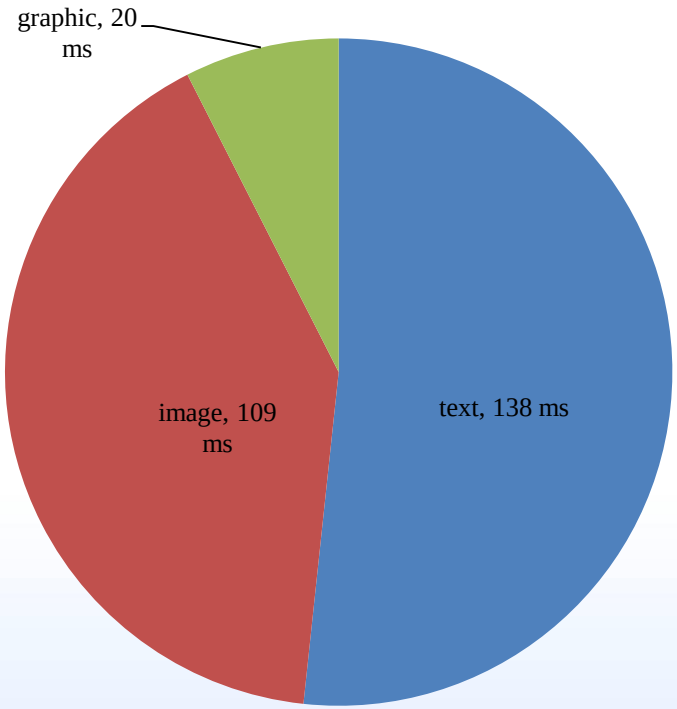


Does it really matter? (Java + C example run)

Entire application profile

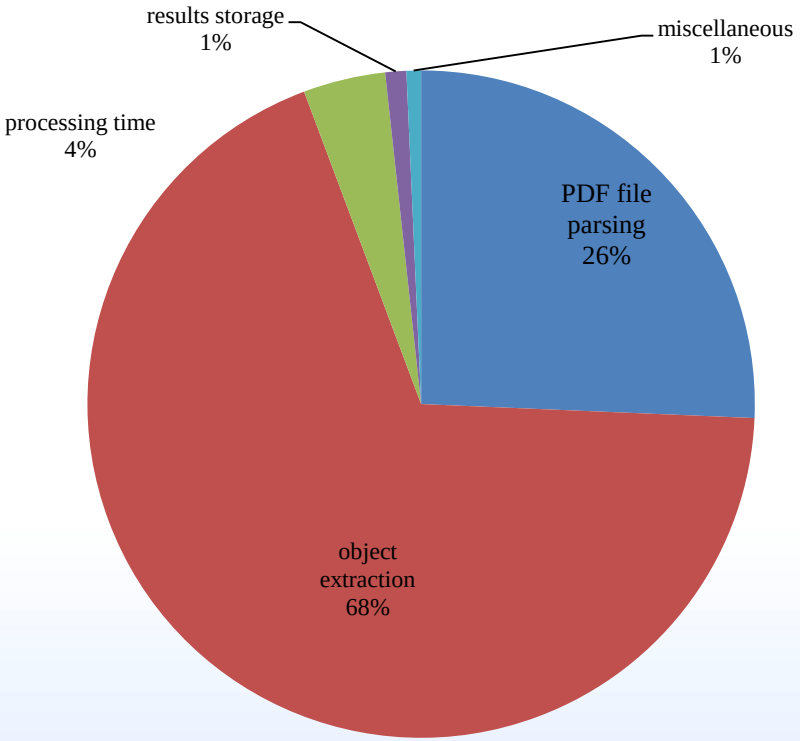


Data analysis profile

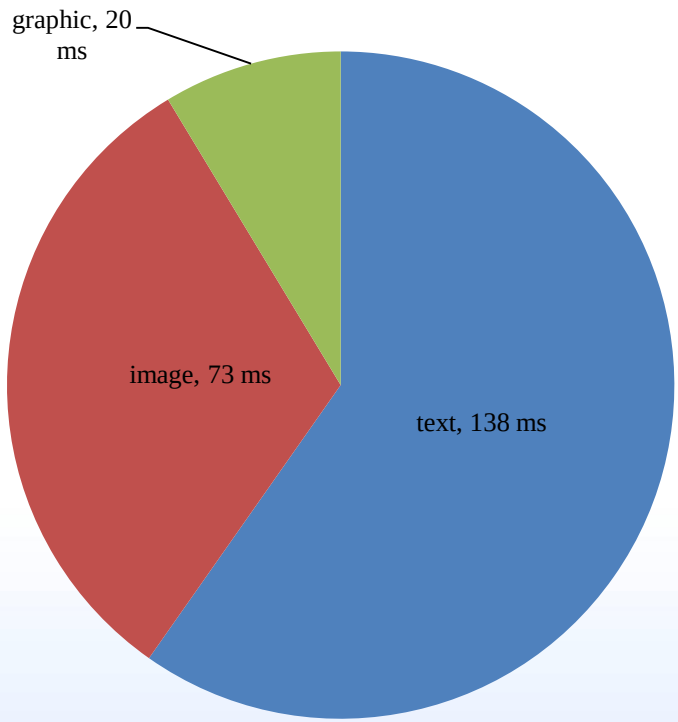


Does it really matter? (Java + C + NVIDIA GPU example run)

Entire application profile



Data analysis profile



Conclusions

- **Implications for doc2learn image analysis algorithm**
  - The image probability density function computation algorithm implemented in Java in doc2learn software can be accelerated by a factor of 6x if the entire doc2learn image analysis software is re-implemented in C,
  - Or by a factor of almost 16x if it also uses an NVIDIA GTX 480 GPU.
  - Actual GPU speedup largely depends on the image size; for images less than 512x512 a properly done CPU implementation will outperform a GPU implementation.
  - Calling a GPU-based implementation from the existing doc2learn Java-based code is still beneficial as it provides up to 4x speedup for sufficiently large images.
  - But another factor of 4x speedup can be achieved by porting the entire image analysis software suite to C and using GPU kernels within the C-based code. Java is not really a high-performance platform for this sort of computations.

Conclusions

- **Implications for doc2learn application**
  - Doc2learn execution profile indicates that only about 4% of the overall execution time for the given pdf file example is spent on the image processing part. Speeding it up by any factor will not make much of a difference for the entire application.
    - Said that, GPU acceleration may be still beneficial for pdf files containing very large images or embedded videos.
  - Doc2learn also implements probability density function computation algorithms for text and vector graphics. These data types exhibit less regular memory access patterns and require much large histograms to be stored. Because of this, they are less suitable for GPU implementation as compared to image histograms.

Conclusions

- **CUDA vs OpenCL**
  - At this point, CUDA-based implementation outperforms the OpenCL based implementation, but it does not provide portability across GPU platforms.
  - We have not investigated OpenCL implementation for a multi-core architecture, but from our prior experience we know that platform-specific tuning will be required to achieve good performance with OpenCL on any architecture. The OpenCL code written for one architecture will execute on another architecture, but typically not at its full potential.
  - Thus, in light of
    - poorer performance of OpenCL implementation
    - immaturity of the OpenCL tools
    - need for architecture-specific code tuning, and
    - overall impact on the doc2learn application performance
  - benefits of OpenCL implementation of the probability density function are minor.

Work in progress

- **Develop a stand-alone C test-bed of the image extraction component of doc2learn**
  - integrate the developed image probability density function computation algorithm (both the CPU and GPU implementations)
  - investigate how to extend the CPU implementation of the histogram computation to the multi-core architecture of modern CPUs
  - conduct a study how the stand-alone implementation compares to the original doc2learn Java-based implementation
  - use the stand-alone framework to analyze power consumption of the CPU and GPU implementations
- **Investigate other image comparison algorithms and their suitability for GPU acceleration**
- **Investigate pros and cons of extending Versus framework to use GPU-based image processing algorithms**