Comparative Evaluation of Big-Data Systems on Scientific Image Analytics Workloads

Parmita Mehta, Sven Dorkenwald, Dongfang Zhao, Tomer Kaftan,
Alvin Cheung, Magdalena Balazinska, Ariel Rokem,
Andrew Connolly, Jake VanderPlaas, Yusra AlSayyad

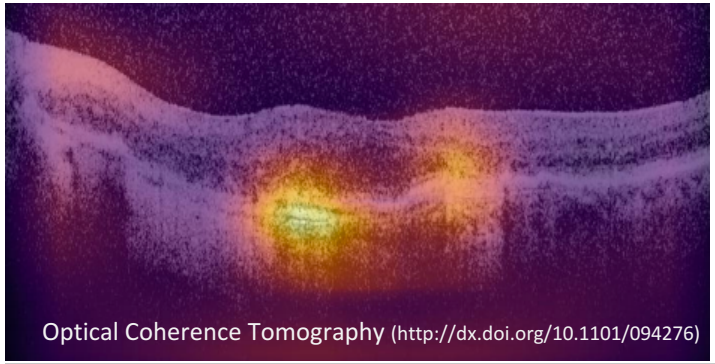
University of Washington

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

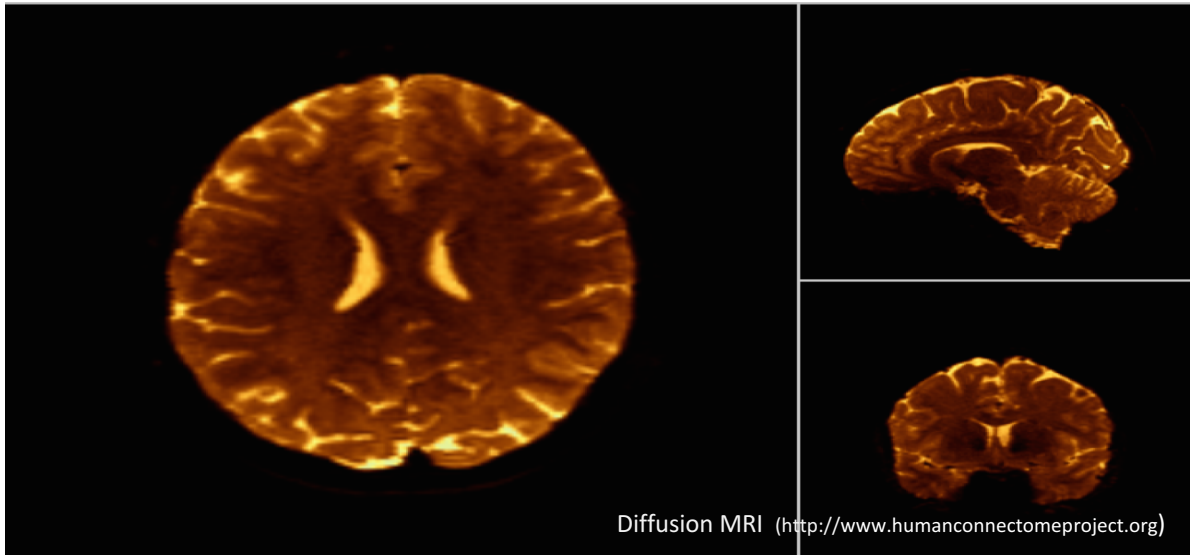
eScience Institute
ADVANCING DATA-INTENSIVE DISCOVERY IN ALL FIELDS

Scientific Images

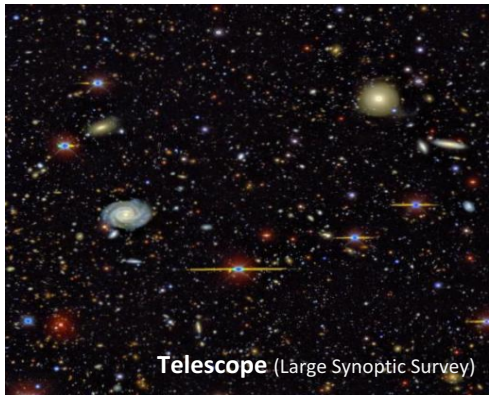
Telescope (Large Synoptic Survey)



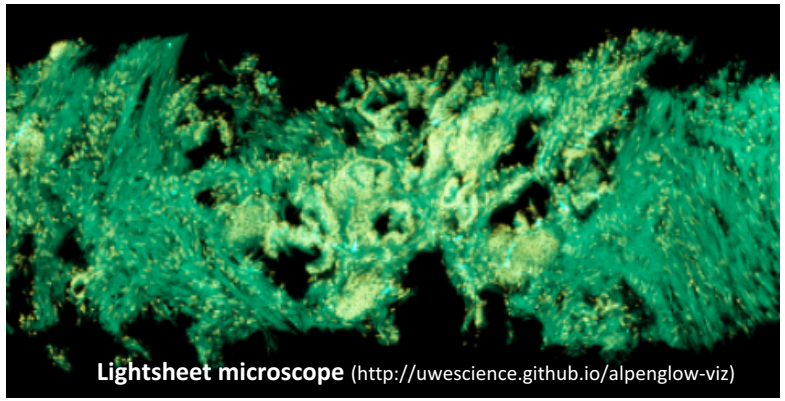
Optical Coherence Tomography (http://dx.doi.org/10.1101/094276)



Diffusion MRI (http://www.humanconnectomeproject.org)



Telescope (Large Synoptic Survey)

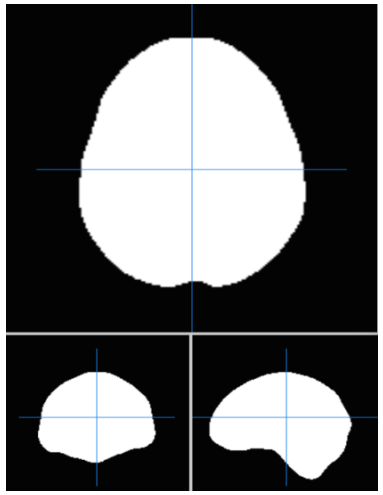


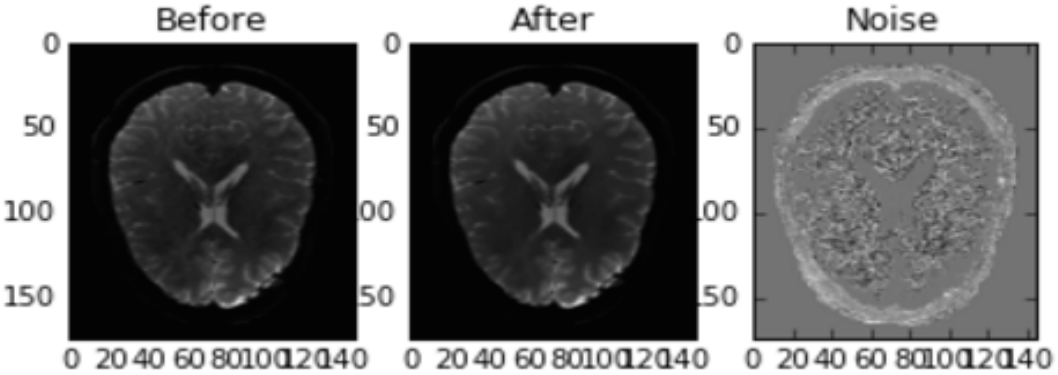
Lightsheet microscope (http://uwescience.github.io/alpenglow-viz)

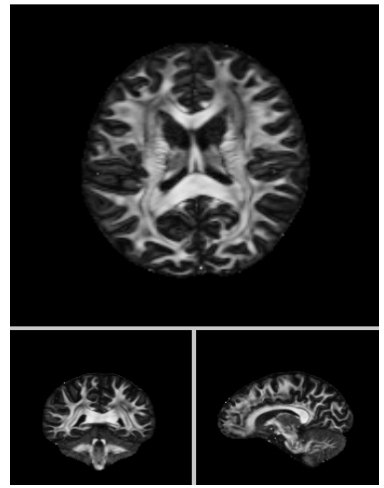
http://www.humanconnectomeproject.org

Neuroscience: Use case

1. Mask-creation
2. Denoising
3. Model fitting







Scientific Image Analysis Today

- Domain specific libraries (Python, Matlab, etc.).
- Driver code is Python e.g. Jupyter notebook.
- Which runs on a laptop.

How do we scale the computation?

Experiments and Analyses

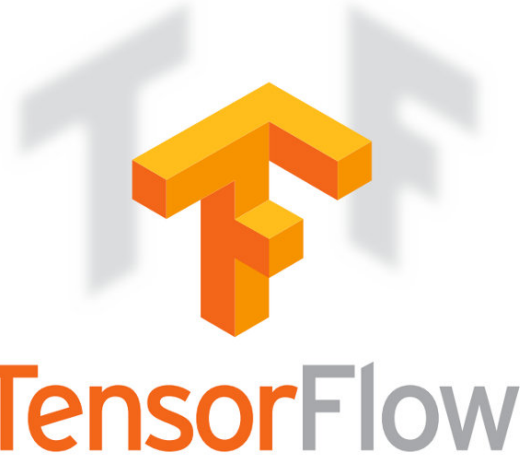
How well do existing systems support scientific image analytics?

Outline

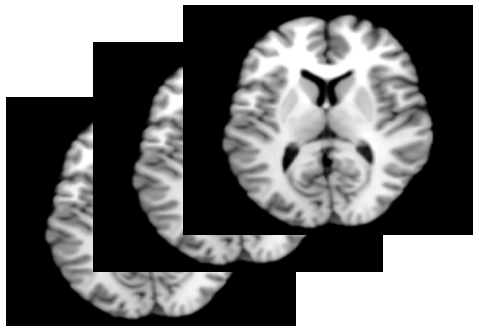
- Systems
- Use case
- Implementation
- Evaluation & Results
- Areas of research
- Conclusion

Systems

- **Dask**: Parallel Python library
- **Myria**: Shared-nothing, relational DBMS
- **SciDB**: Multidimensional array DBMS
- **Spark**: In-memory big data system
- **TensorFlow**: Machine learning library

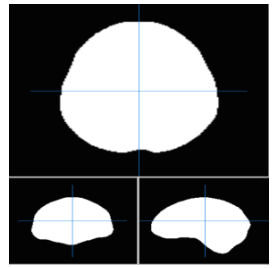


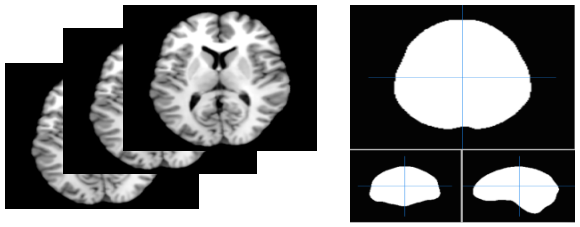
Neuroscience: Query Plan



MASK-CREATION

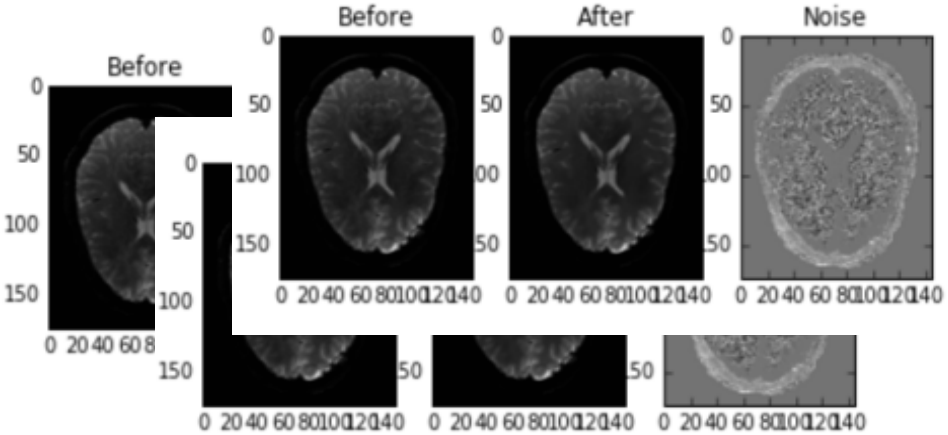
$$\sigma_{\text{imageid}} \rightarrow \Upsilon_{\text{subject}} \rightarrow G_{\text{Avg}} \rightarrow \text{UDF}_{\text{Median\_OTSU}}$$

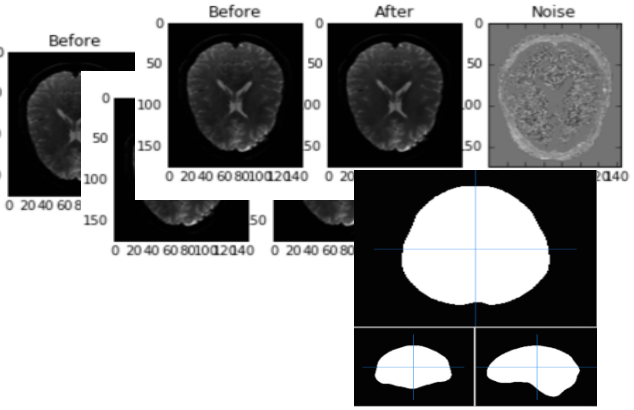




DENOISING

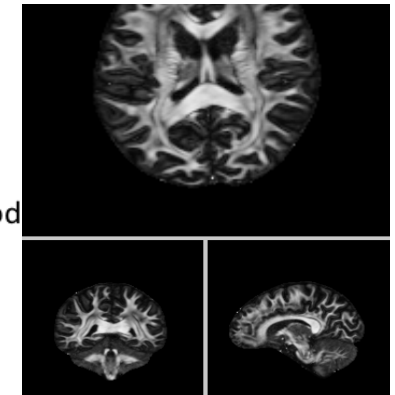
$$\bowtie_{\text{subject}} \rightarrow \text{UDF}_{\text{Denoise}}$$





MODEL FITTING

$$\text{Flatmap}_{\text{image->voxel}} \rightarrow \bowtie_{\text{subject}} \rightarrow \Upsilon_{\text{voxel, subject}} \rightarrow \text{UDA}_{\text{TensorMod}}$$



Complex User Defined Functions

```
    if len(input_volume.shape) == 4:
        if vol_idx is not None:
            b0vol = np.mean(input_volume[..., tuple(vol_idx)], axis=3)
        else:
            b0vol = input_volume[..., 0].copy()
    else:
        b0vol = input_volume.copy()
    # Make a mask using a multiple pass median filter and histogram
    # thresholding.
    mask = multi_median(b0vol, median_radius, numpass)
    thresh = otsu(mask)
    mask = mask > thresh

    if dilate is not None:
        cross = generate_binary_structure(3, 1)
        mask = binary_dilation(mask, cross, iterations=dilate)

    # Auto crop the volumes using the mask as input_volume for bounding box
    # computing.
    if autocrop:
        mins, maxs = bounding_box(mask)
        mask = crop(mask, mins, maxs)
        croppedvolume = crop(input_volume, mins, maxs)
        maskedvolume = applymask(croppedvolume, mask)
    else:
        maskedvolume = applymask(input_volume, mask)
    return maskedvolume, mask
```

[Diffusion MR Imaging in Python http://dipy.org]

"A threshold selection method from gray-level histograms", Nobuyuki Otsu IEEE Transactions on Systems, Man, and Cybernetics ,Jan. 1979

Implementation

- **Dask**: Task graph with Python data structures and functions.
- **Myria**: MyriaL and user-defined Python functions.
- **SciDB**:
  - *Implementation 1: Pure AQL .*
  - *Implementation 2:* AQL and Python function using *stream().*
- **Spark**: PySpark with Python lambdas.
- **TensorFlow**: Compute graph in Python with *tensors*.

Evaluation

- Qualitative
  - Can existing pipelines run on the systems?
  - How much effort is required to implement scientific pipelines?
  - How much technical expertise is required to optimize the system?
- Quantitative
  - How do systems scale with increasing data size?
  - What is the speed-up with cluster size increase?
  - Do systems show similar per-step performance?
  - System Tuning

Can we implement & execute end-to-end pipelines?

With great difficulty!

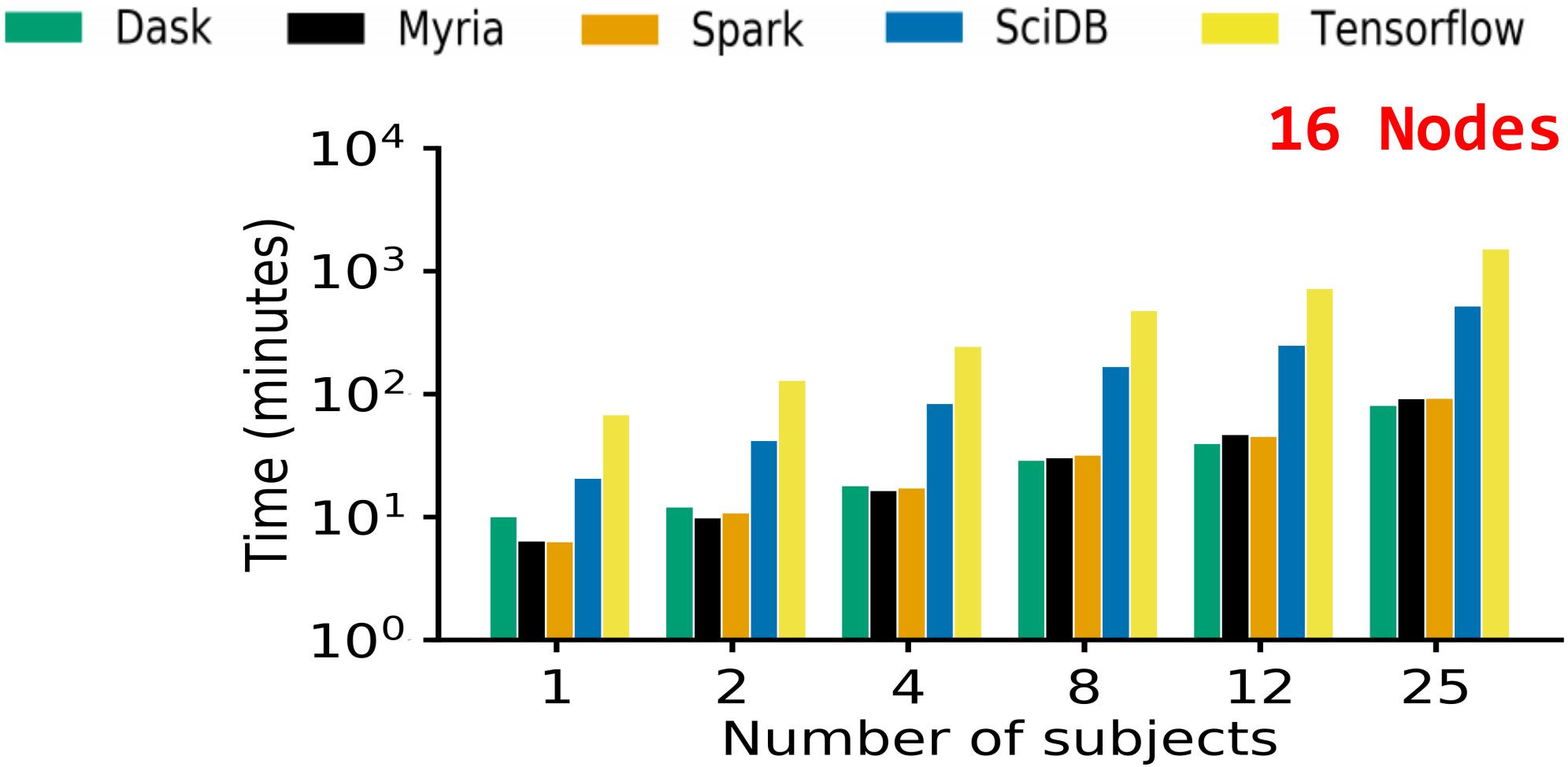
Qualitative Evaluation

How much effort is required to implement scientific pipelines?

	Dask	Myria	SciDB	Spark	TensorFlow
Reuse and (total) code*	20 (120)	16 (75)	16 (315)	17 (114)	0 (500+)
Streamlined installation	✓	✓	✗	✓	–
Flexible data formats	✓	–	✗	–	✗
Debuggability	✗	✓	–	✓	✓

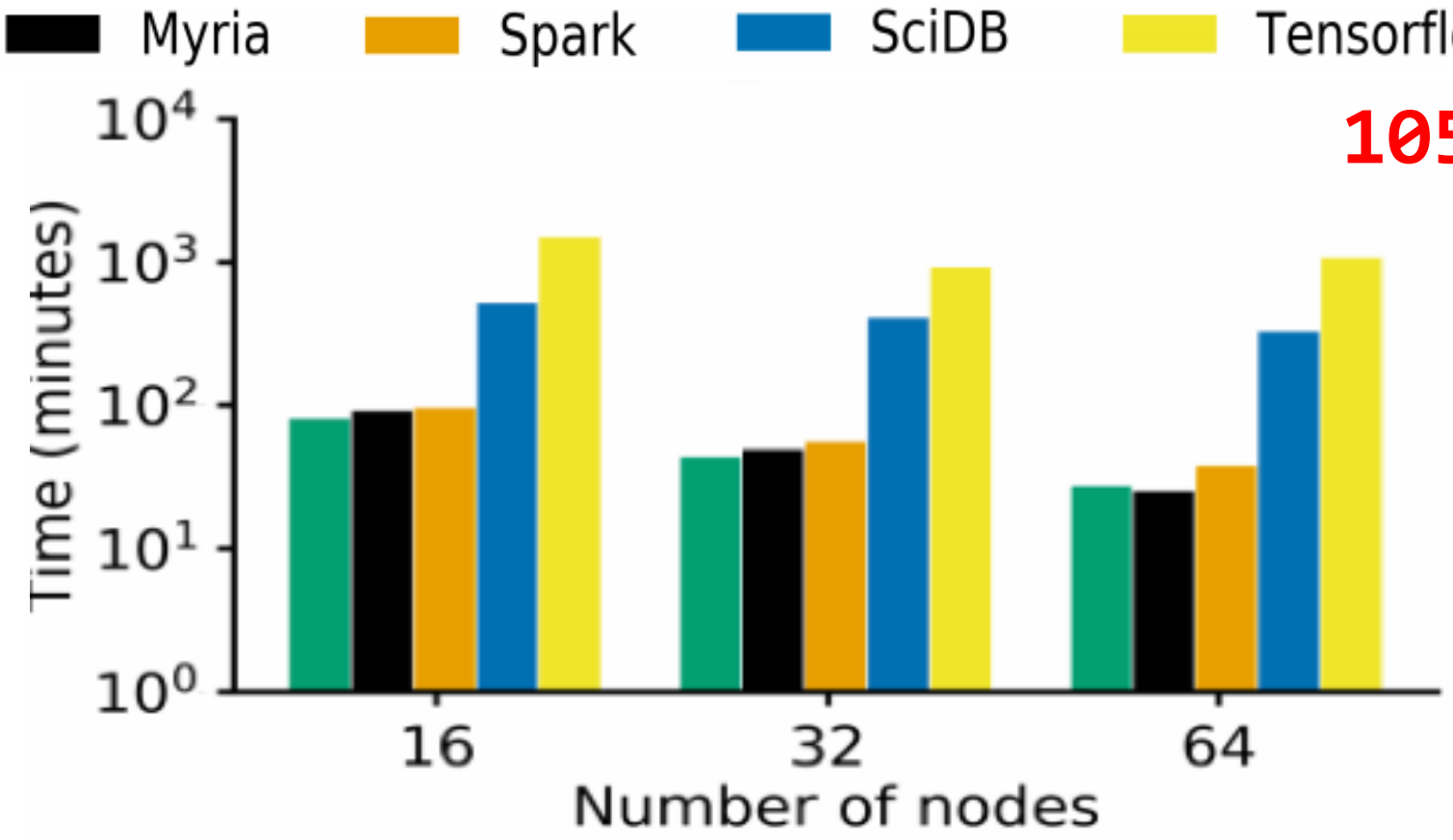
*Reference implementation for neuroscience use case was 26 lines of Python driver code calling into DiPy library.

End-to-end runtime



Subjects	1	2	4	8	12	25
Input	4.2GB	8.4GB	16.8GB	33.6GB	50.4GB	105GB

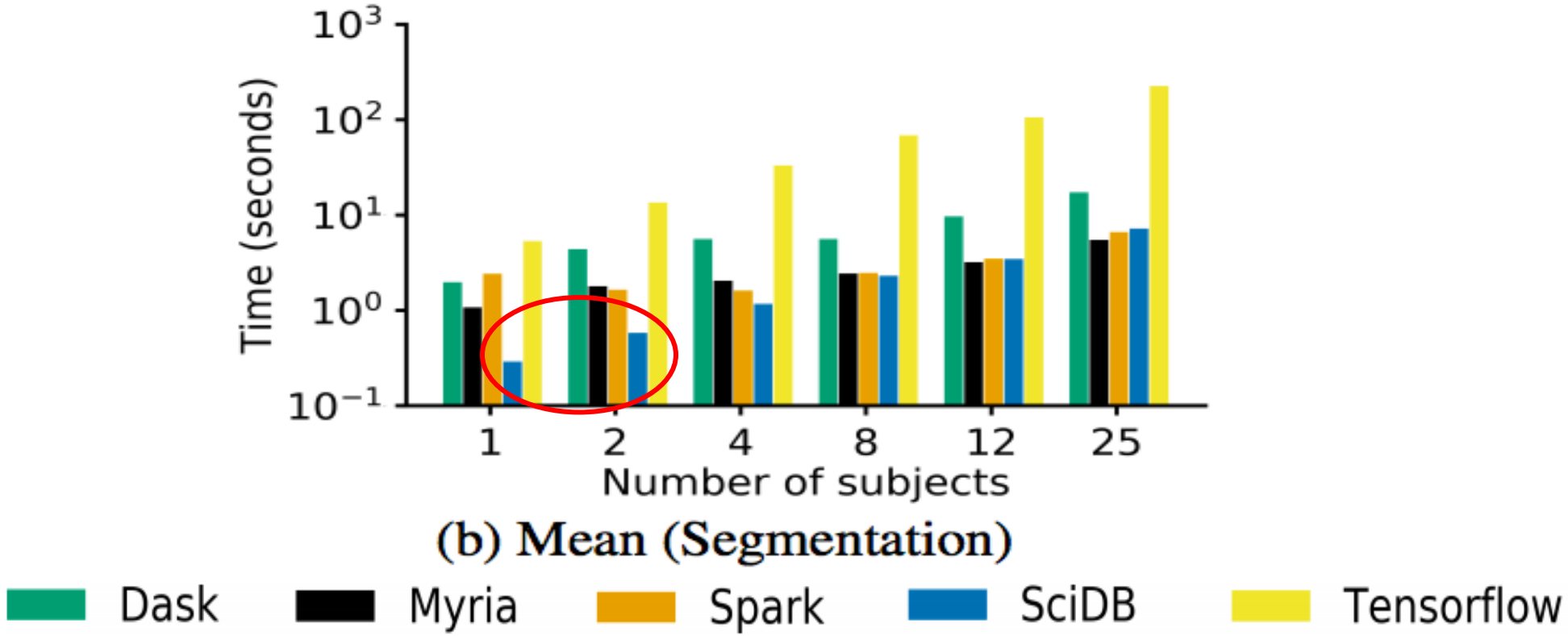
End-to-end: Speed-up



(d) Neuroscience: End-to-end runtime with varying cluster size

Re-write vs. Re-use

Re-write takes more time, but might be more efficient!



System Tuning

- None of the systems perform well with default settings
- All require manual tuning
- Two areas that require tuning in all systems:
  - Degree of parallelism
  - Memory management

Results Summary

	Dask	Myria	SciDB	Spark	TensorFlow
Code reuse, reference impl. (26)	20 (120)	16 (75)	16 (315)	17 (114)	0 (500+)
Streamlined installation	✓	✓	✗	✓	–
Flexible data formats	✓	–	✗	–	✗
Debbugability	✗	✓	–	✓	✓
Out of box performance	✗	✗	✗	✗	✗

Areas of Research

- Need to efficiently support pipelines with UDFs
- Large tuple sizes
- Memory management
- Self-tuning & robust systems

Conclusion

- Scientific image analytics is increasingly important
- Big Data systems are still too difficult to use and …
  - need to provide streamlined installation
  - need improvement in overall robustness and usability
  - out of box settings are the only ones that matter

Thank You!

TensorFlow: Sven Dorkenwald

SciDB: Dongfang Zhao

Dask: Alvin Cheung & Parmita Mehta

Myria: Parmita Mehta

Spark: Parmita Mehta & Tomer Kaftan

Neuroscience Domain Expert: Ariel Rokem

Astronomy Domain Expert: Jake VanderPlaas, Andrew Connolly, Yusra Alsayyad