Maarten Breddels

- Ex: astronomer (working on software for big data and visualization: vaex)
Maarten Breddels

- Ex: astronomer (working on software for big data and visualization: vaex)
- Now: Freelancer / consultant / data scientist for Python / Jupyter
Maarten Breddels

- Ex: astronomer (working on software for big data and visualization: vaex)
- Now: Freelancer / consultant / data scientist for Python / Jupyter
- Core Jupyter-Widgets developer
Maarten Breddels

- Ex: astronomer (working on software for big data and visualization: vaex)
- Now: Freelancer / consultant / data scientist for Python / Jupyter
- Core Jupyter-Widgets developer
- Authors of vaex and ipyvolume
Maarten Breddels

- Ex: astronomer (working on software for big data and visualization: vaex)
- Now: Freelancer / consultant / data scientist for Python / Jupyter
- Core Jupyter-Widgets developer
- Authors of vaex and ipyvolume

I live on the internet at:
- @maartenbreddels
- maartenbreddels@gmail.com
- github.com/maartenbreddels
- www.maartenbreddels.com
Maarten Breddels

- Ex: astronomer (working on software for big data and visualization: vaex)
- Now: Freelancer / consultant / data scientist for Python / Jupyter
- Core Jupyter-Widgets developer
- Authors of vaex and ipyvolume

I live on the internet at:

- @maartenbreddels
- maartenbreddels@gmail.com
- github.com/maartenbreddels
- www.maartenbreddels.com

Jovan Veljanoski
Maarten Breddels

• Ex: astronomer (working on software for big data and visualization: vaex)
• Now: Freelancer / consultant / data scientist for Python / Jupyter
• Core Jupyter-Widgets developer
• Authors of vaex and ipyvolume

I live on the internet at:
@maartenbreddels
maartenbreddels@gmail.com
github.com/maartenbreddels
www.maartenbreddels.com

Jovan Veljanoski

• Ex- astronomer (big influence on vaex)
Maarten Breddels

- Ex: astronomer (working on software for big data and visualization: vaex)
- Now: Freelancer / consultant / data scientist for Python / Jupyter
- Core Jupyter-Widgets developer
- Authors of vaex and ipyvolume

I live on the internet at:
- @maartenbreddels
- maartenbreddels@gmail.com
- github.com/maartenbreddels
- www.maartenbreddels.com

Jovan Veljanoski

- Ex- astronomer (big influence on vaex)
- Data scientists at Xebia Labs
Maarten Breddels

- Ex: astronomer (working on software for big data and visualization: vaex)
- Now: Freelancer / consultant / data scientist for Python / Jupyter
- Core Jupyter-Widgets developer
- Authors of vaex and ipyvolume

I live on the internet at:
- @maartenbreddels
- maartenbreddels@gmail.com
- github.com/maartenbreddels
- www.maartenbreddels.com

Jovan Veljanoski

- Ex- astronomer (big influence on vaex)
- Data scientists at Xebia Labs
- vaex coauthor
Maarten Breddels

- Ex: astronomer (working on software for big data and visualization: vaex)
- Now: Freelancer / consultant / data scientist for Python / Jupyter
- Core Jupyter-Widgets developer
- Authors of vaex and ipyvolume

I live on the internet at:

- @maartenbreddels
- maartenbreddels@gmail.com
- github.com/maartenbreddels
- www.maartenbreddels.com

Jovan Veljanoski

- Ex- astronomer (big influence on vaex)
- Data scientists at Xebia Labs
- vaex coauthor

I live on the internet at:

- @N147185
- jovan.veljanoski@gmail.com
- github.com/JovanVeljanoski
- https://www.linkedin.com/in/jovanvel/
Agenda

- Why does vaex exist?
- What is vaex?
- Why is it so fast?
- Demos
- Summary
Motivation: Gaia
Motivation: Gaia

- > 1 billion stars
- Sky positions
- Distance
- Motions
- And many more
- Errors / Correlations
Motivation: Gaia

• > 1 billion stars
• Sky positions
• Distance
• Motions
• And many more
  • Errors / Correlations
• Latest data release
  • 1.7 billion rows
  • 1.2 TB
  • 94 columns/features
• How fast can it be done?
• How fast can it be done?

• $10^9 \times 2 \times 8$ bytes = 15 GiB (double is 8 bytes)
• How fast can it be done?

• $10^9 \times 2 \times 8$ bytes = 15 GiB (double is 8 bytes)

• Memory bandwidth: 10-50 GiB/s: ~1 second
• How fast can it be done?

• $10^9 \times 2 \times 8$ bytes = 15 GiB (double is 8 bytes)

• Memory bandwidth: 10-50 GiB/s: ~1 second

• CPU: 3 Ghz (but multicore, say 4-8): 12-24 cycles/second
• How fast can it be done?
  • $10^9 \times 2 \times 8$ bytes = 15 GiB (double is 8 bytes)
  • Memory bandwidth: 10-50 GiB/s: ~1 second
  • CPU: 3 Ghz (but multicore, say 4-8): 12-24 cycles/second
  • Few cycles per row/object, simple algorithm
• How fast can it be done?
• \(10^9 \times 2 \times 8 \text{ bytes} = 15 \text{ GiB (double is 8 bytes)}\)
• Memory bandwidth: 10-50 GiB/s: \(~1\) second
• CPU: 3 Ghz (but multicore, say 4-8): 12-24 cycles/second
• Few cycles per row/object, simple algorithm
• Histograms/Density/Statistics grids
How fast can it be done?

- 109 * 2 * 8 bytes = 15 GiB (double is 8 bytes)
- Memory bandwidth: 10-50 GiB/s: ~1 second
- CPU: 3 Ghz (but multicore, say 4-8): 12-24 cycles/second
- Few cycles per row/object, simple algorithm
- Histograms/Density/Statistics grids
0d  330,000 rows

1d

2d

3d
0d  330,000 rows

1d

2d

3d
0d  330,000 rows  mean: -0.083
0d  330,000 rows  mean: -0.083
vaex

• Python library (conda/pip installable)
vaex

- Python library (conda/pip installable)
- Pandas-like (familiar API)
  - Out-of-core, expression system
  - ApacheArrow / hdf5 + memory mapping
vaex

• Python library (conda/pip installable)
• Pandas-like (familiar API)
  • Out-of-core, expression system
  • ApacheArrow / hdf5 + memory mapping
• Strong focus on statistics on N-d grids (count/mean/max/std/…)
vaex

• Python library (conda/pip installable)
• Pandas-like (familiar API)
  • Out-of-core, expression system
  • ApacheArrow / hdf5 + memory mapping
• Strong focus on statistics on N-d grids (count/mean/max/std/…)
• >1 billion rows / sec on a desktop (quad core 3Gz)
  • >50x faster than scipy.stats.binned_statistic_2d
vaex

• Python library (conda/pip installable)
• Pandas-like (familiar API)
  • Out-of-core, expression system
  • ApacheArrow / hdf5 + memory mapping
• Strong focus on statistics on N-d grids (count/mean/max/std/…)
• >1 billion rows / sec on a desktop (quad core 3Gz)
  • >50x faster than scipy.stats.binned_statistic_2d
• Does visualisation / matplotlib / bqplot / ipyvolume / ipyleaflet
vaex

- Python library (conda/pip installable)
- Pandas-like (familiar API)
  - Out-of-core, expression system
  - ApacheArrow / hdf5 + memory mapping
- Strong focus on statistics on N-d grids (count/mean/max/std/…)
- >1 billion rows / sec on a desktop (quad core 3Gz)
  - >50x faster than scipy.stats.binned_statistic_2d
- Does visualisation / matplotlib / bqplot / ipyvolume / ipyleaflet
- More
  - Machine learning (Boosted Trees, K-means, PCA, ..)
  - Distributed computing (>10^{10} rows)
What kind of data?
What kind of data?
What kind of data?
What kind of data?
“Never do a live demo”
-Many people

Demo notebooks at:
https://github.com/maartenbreddels/talk-pyparis-2018
Takeaway
Takeaway

• Next generation data frame library (vaex?)
Takeaway

• Next generation data frame library (vaex?)

• Large datasets should be explored with statistics, not individual points
Takeaway

• Next generation data frame library (vaex?)
  • Large datasets should be explored with statistics, not individual points
  • Large datasets should be memory mapped: Apache Arrow / hdf5
Takeaway

• Next generation data frame library (vaex?)
  • Large datasets should be explored with statistics, not individual points
  • Large datasets should be memory mapped: Apache Arrow / hdf5
  • Should use expressions
Takeaway

• Next generation data frame library (vaex?)
  • Large datasets should be explored with statistics, not individual points
  • Large datasets should be memory mapped: Apache Arrow / hdf5
  • Should use expressions
    • No memory wasted
Takeaway

• Next generation data frame library (vaex?)
  • Large datasets should be explored with statistics, not individual points
  • Large datasets should be memory mapped: Apache Arrow / hdf5
• Should use expressions
  • No memory wasted
  • No information lost: JIT/derivatives
Takeaway

• Next generation data frame library (vaex?)
  • Large datasets should be explored with statistics, not individual points
  • Large datasets should be memory mapped: Apache Arrow / hdf5
  • Should use expressions
    • No memory wasted
    • No information lost: JIT/derivatives
  • ML pipelines are a byproduct
• vaex

• https://vaex.io

• https://github.com/maartenbreddels/vaex

• pip install —pre vaex

• conda install -c conda-forge vaex

• https://github.com/maartenbreddels/talk-pyparis-2018

• maartenbreddels@gmail.com

• jovan.veljanoski@gmail.com