Orchestrating a brighter world NEC

Machine Learning and data analytics MW for SX-Aurora TSUBASA

NEC Data Science Research Laboratories
Takeo Hosomi
June. 25, 2018

Machine Leaning in Big Data Analytics

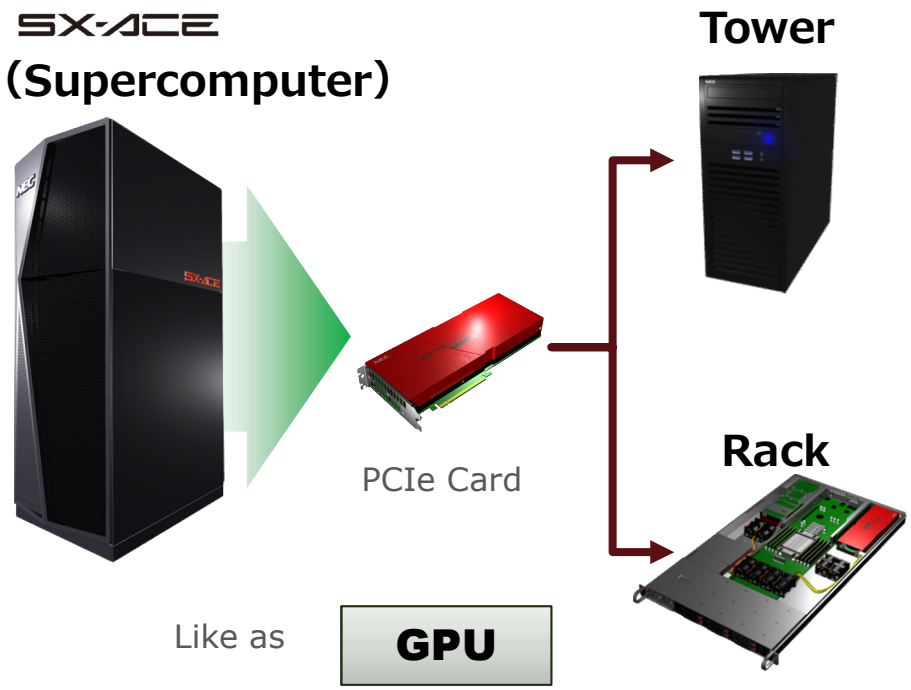
- Recently, machine learning (ML) is becoming important in Big Data analytics

- Most ML algorithms can be written as “matrix operation”; Large scale ML tends to use “sparse matrix”, which is memory intensive
  - Vector architecture is promising

- We created middleware for ML that runs on vector architecture
- In addition, we made the middleware seamlessly callable from Apache Spark and Python
  - More than 50x performance improvement
  - Users of Spark/Python can easily utilize high performance of vector without special programming

New Generation AI/BD Accelerator !!
～SX-Aurora TSUBASA

SX-Aurora TSUBASA



Downsizing a super computer as an accelerator for BigData and AI

POINT 1 High Performance

Enables to achieve a high performance on memory intensive applications

POINT 2 Easy to use

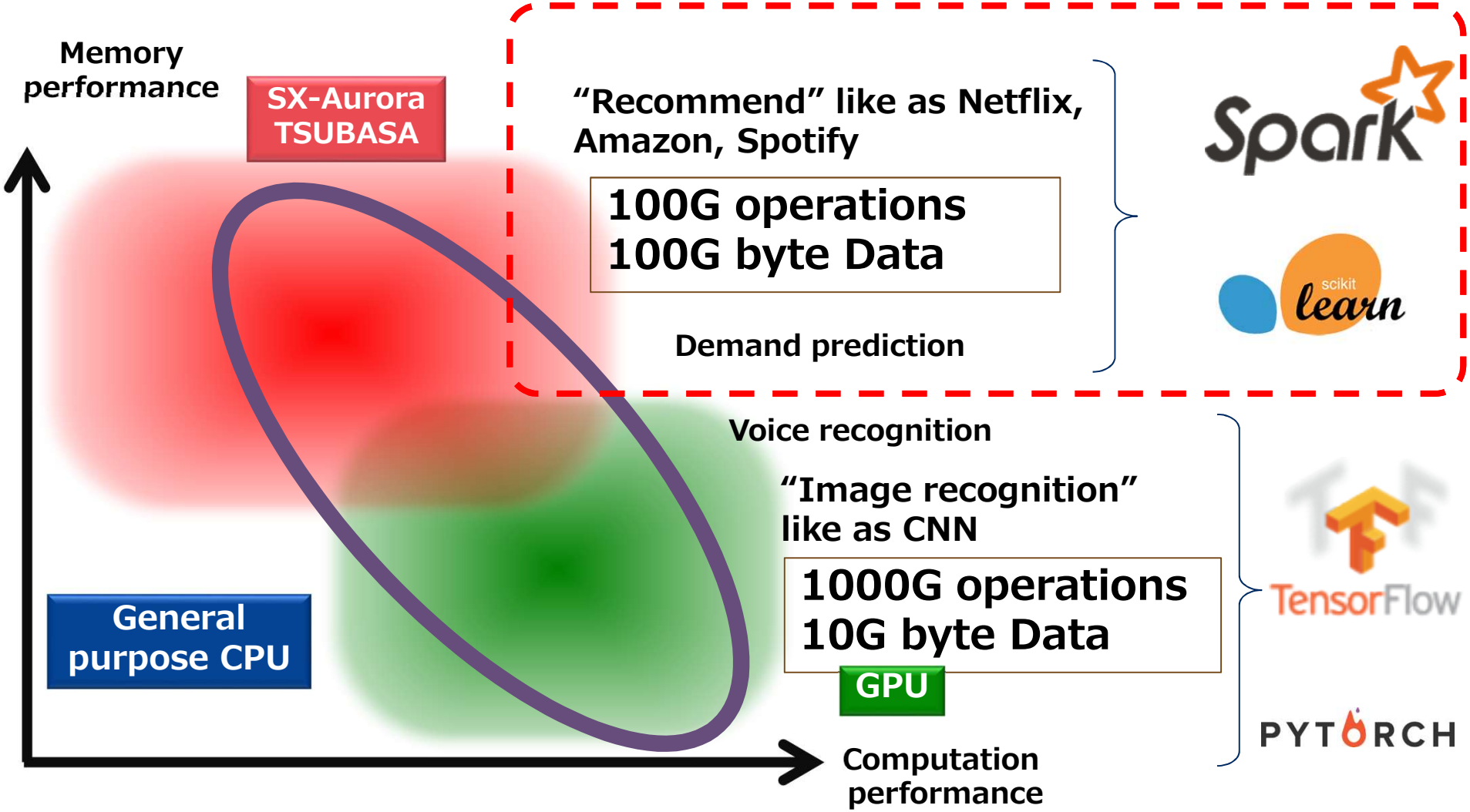
Can program with standard C and C++, and our compiler generate an optimized code.

POINT 3 Flexible

Support several form factor; tower, rack, and HPC models.

Position of SX-Aurora TSUBASA

■ We target accelerating memory intensive workloads for Bigdata/AI



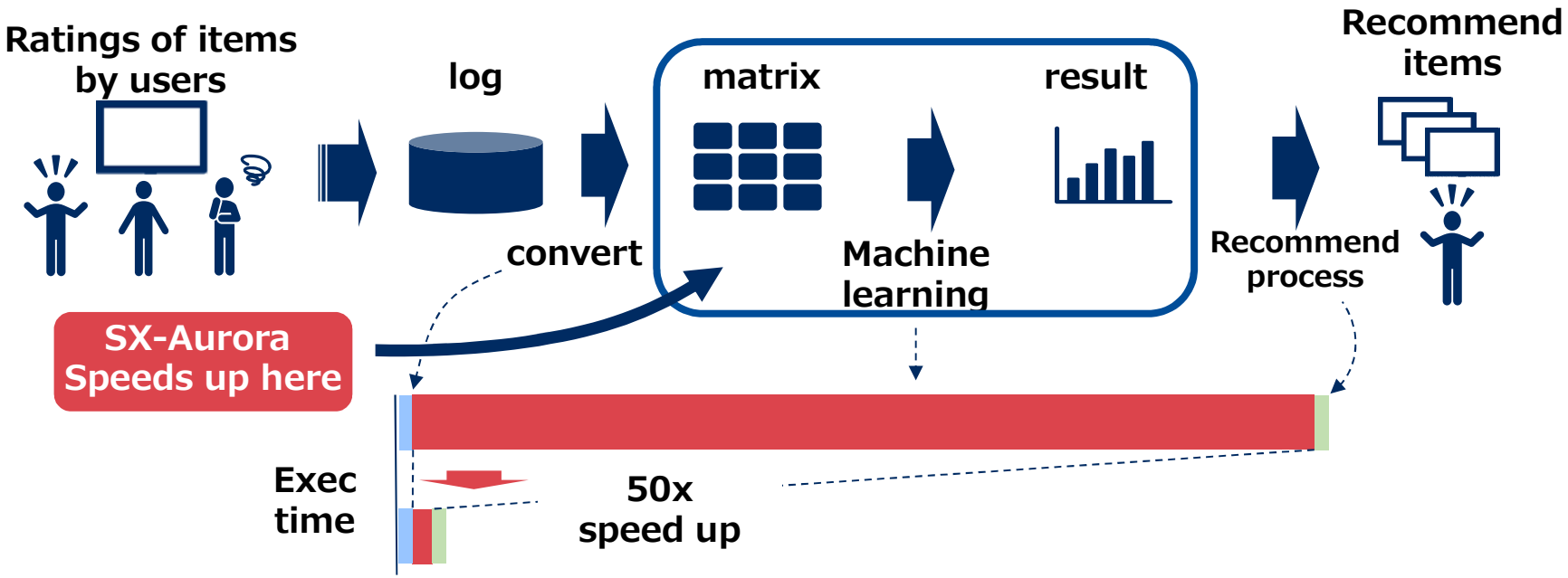
Application: Recommendation System

SX-Aurora TSUBASA can reduce the number of servers by 1/50

- 35% sales of Amazon, 75% sales of Netflix is from recommendation
- More than 95% of the execution time is spent on machine learning
  - SX-Aurora TSUBASA showed 50x speed up with small data

50X

1/50 computing power consumption
Every 30 min updates from 24hours updates



Apache Spark for big data analytics

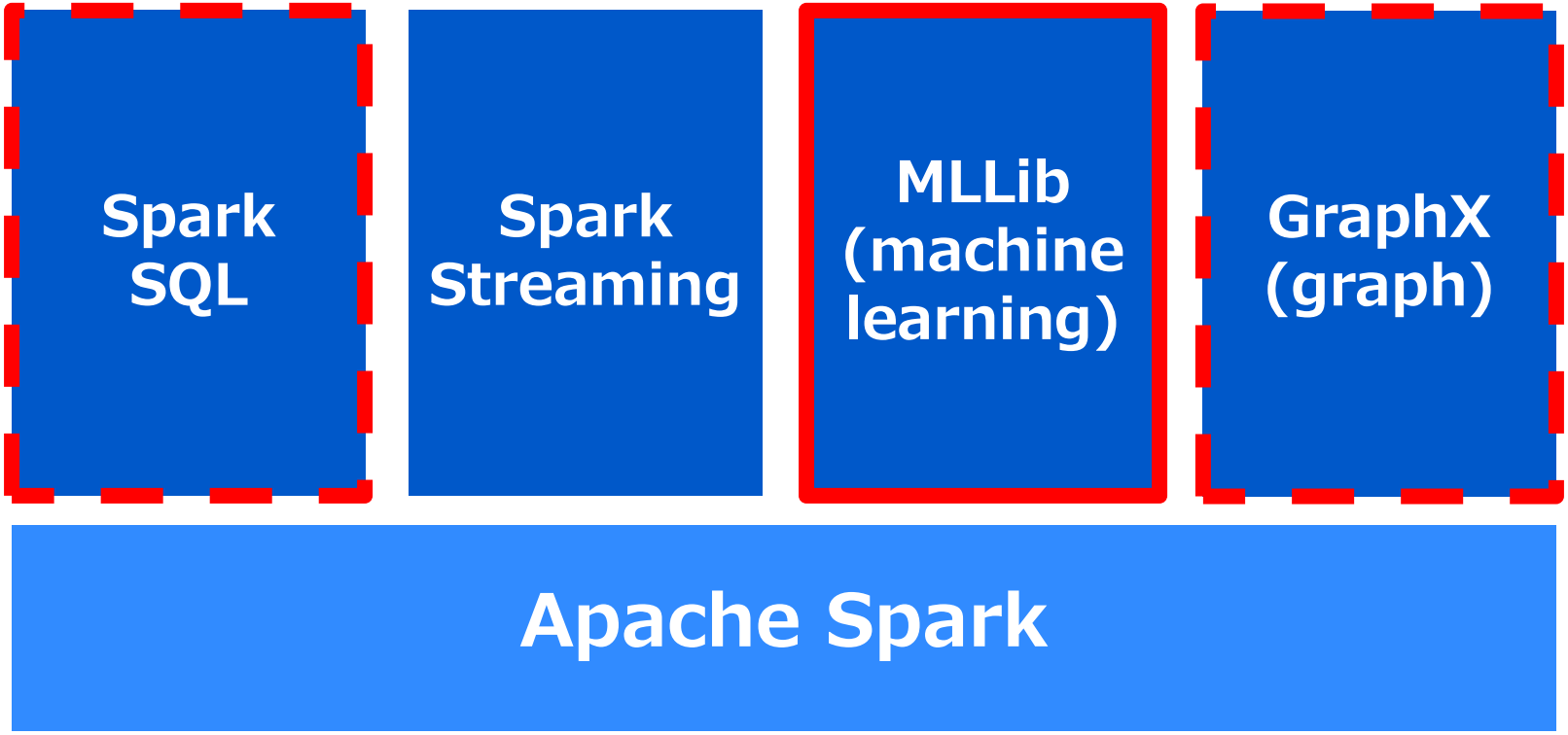
Spark is de facto standard of statistical machine learning middleware

Only for Presentation

Apache Spark and its components

Apache Spark is a unified analytics platform for large-scale data processing. It supports SQL processing, stream processing, machine learning, and graph processing.

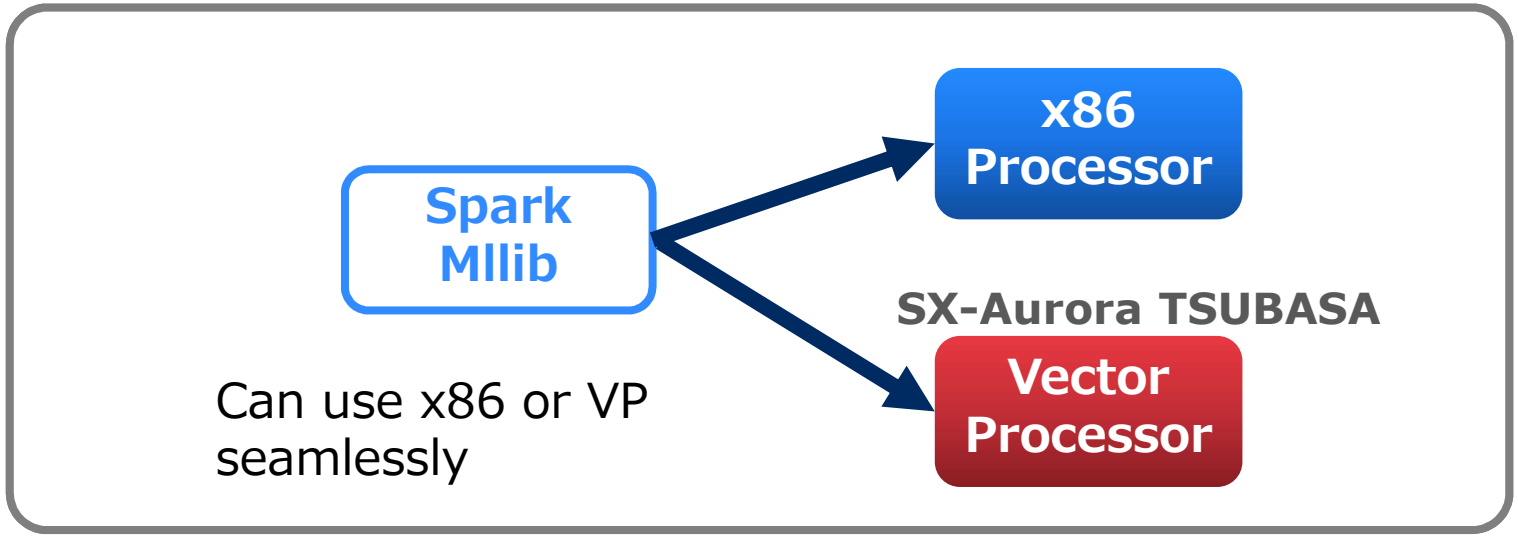
- We have supported MLlib (>50% of functions have been supported), and start to investigate Spark SQL and GraphX.



Machine Learning MW on SX-Aurora TSUBASA

Accelerate Spark machine learning library (Spark Mllib) by replacing our optimized library "Frovedis" for SX-Aurora TSUBASA

- Application programmer on Spark can use SX-Aurora TSUBASA seamlessly
- Our library is fully optimized for SX-Aurora TSUBASA, and also support to use multiple cards.



Prototype implementation: “Frovedis”

Replacing Spark Mllib to our original library “Frovedis” for SX-Aurora TSUBASA. “Frovedis” provides the same interface to Spark Mllib.

Original Spark program: logistic regression

```
…
import org.apache.spark.mllib.classification.LogisticRegressionWithSGD
…                                                          // preprocessing
val model = LogisticRegressionWithSGD.train(data)
…
```

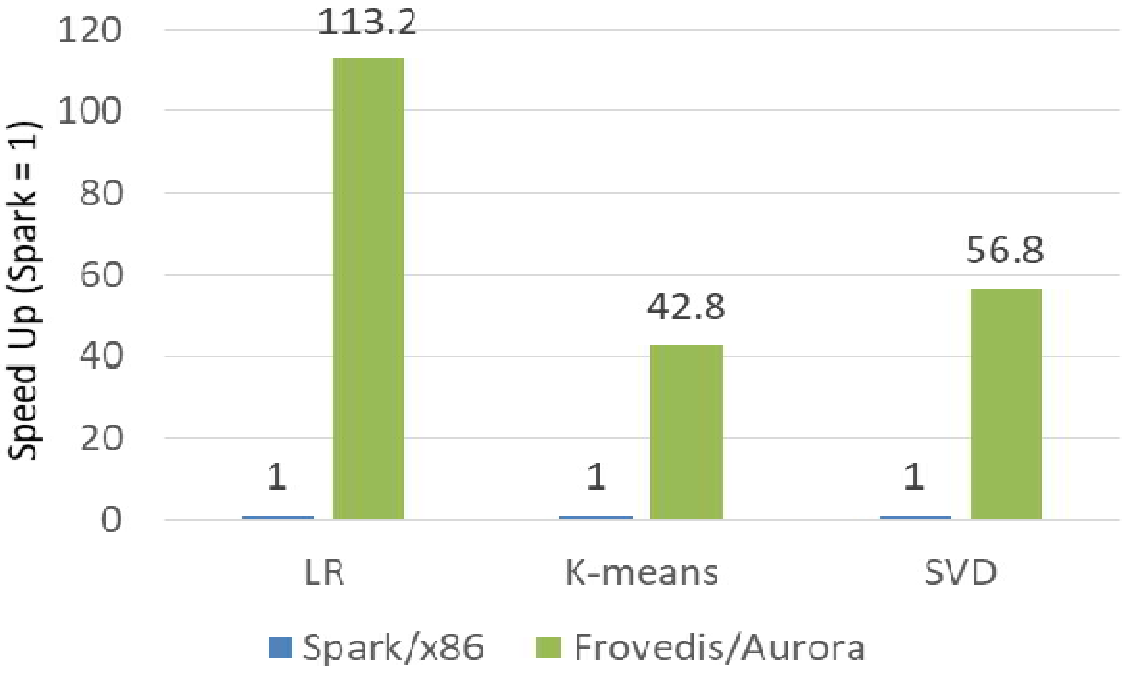
Change to call Frovedis implementation

```
…
import main.scala.mllib.glm.LogisticRegressionWithSGD  // change import
…
FrovedisServer.initialize(…)                            // invoke Frovedis Server
val model = LogisticRegressionWithSGD.train(data)      // no change: same API
FrovedisServer.shut_down()                             // stop Frovedis Server
…
```

Specify command to invoke Frovedis

Performance evaluation (1) Machine Learning

"Frovedis" on SX-Aurora TSUBASA shows 42x to 113x performance improvement from Spark Mllib.

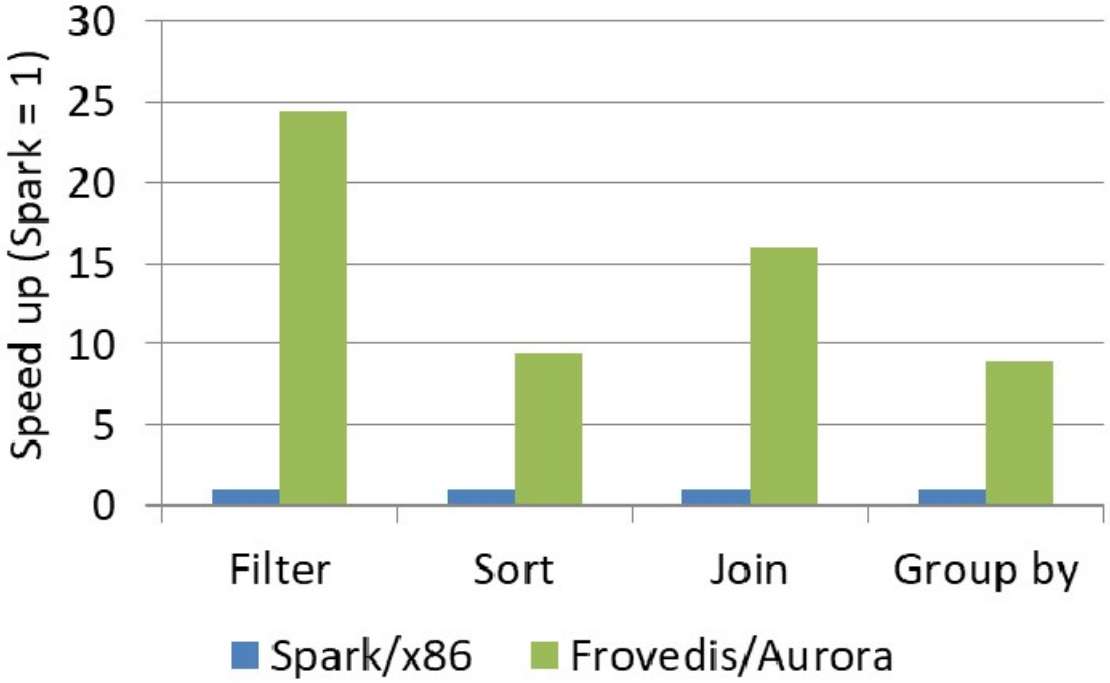


Xeon (Gold 6126) 1 socket vs 1 VE, with sparse data (w/o I/O)

- LR uses CTR data provided by Criteo (1/4 of the original, 6GB)
- K-means and SVD used Wikipedia doc-term matrix (10GB)

Performance Evaluation (2) DataFrame Processing

"Frovedis" on SX-Aurora TSUBASA shows 9x to 24x performance improvement from Spark DataFrame.



- x86: Xeon Gold 6126 CPU @ 2.60GHz (Skylake)
- Spark: 2.2.1
- Aurora model 10b

Froveids (1/2)

Middleware that provides interface like Spark

- Written in C++
- Internally uses MPI to implement distributed processing

Users need not be aware of MPI to write distributed processing

- Write functions in C++
- Provide the functions to the middleware to run them in parallel

Example: double each element of distributed variable

```
int two_times(int i) {return i * 2;}
int main(...) {
  ...
  dvector<int> r = v.map(two_times);
}
```

distributed variable

run "two_times" in parallel

Frovedis (2/2)

Provides dense/sparse matrix library

- Including basic matrix operations and linear algebra
  - e.g. matrix multiplication, singular value decomposition, solving linear equations
- Backed by existing libraries
  - e.g. ScaLAPACK/PBLAS, Parallel ARPACK

Provides machine learning (ML) algorithms

- Utilizing the above matrix library
- Especially on <u>sparse datasets</u>

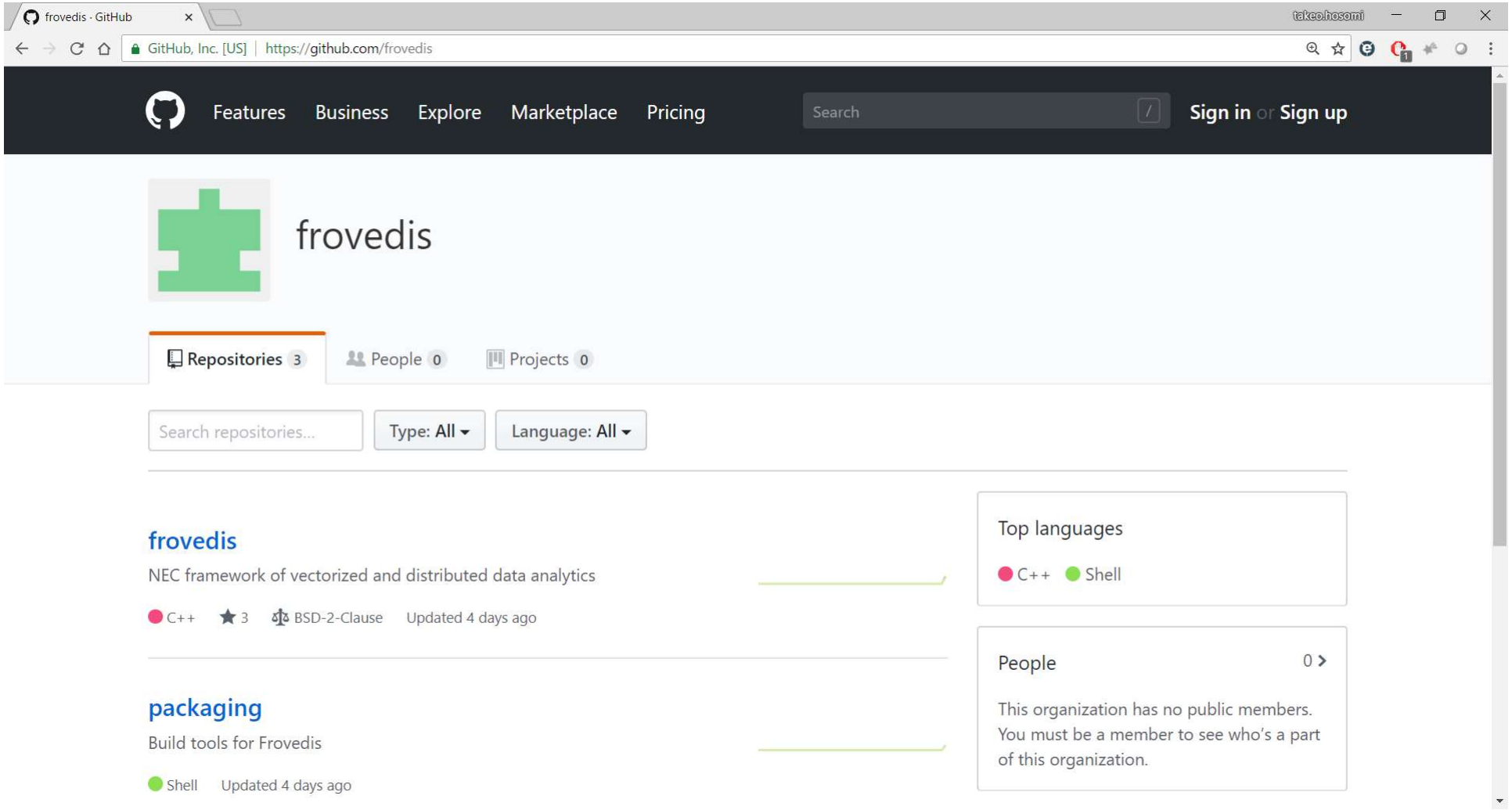
⬆

- Large scale ML tends to use sparse matrix
- Vector architecture is very good at sparse matrix, because sparse matrix operations require large memory bandwidth

Frovedis

Open Source at github

- https://github.com/frovedis/frovedis



Orchestrating a brighter world

NEC