Apache SystemML - Declarative Large-Scale Machine Learning

Romeo Kienzler (IBM Waston IoT)
Berthold Reinwald (IBM Almaden Research Center)
Frederick R. Reiss (IBM Almaden Research Center)
Matthias Rieke (IBM Analytics)

Swiss Data Science Conference 16 - ZHAW - Winterthur

“High-level programming”

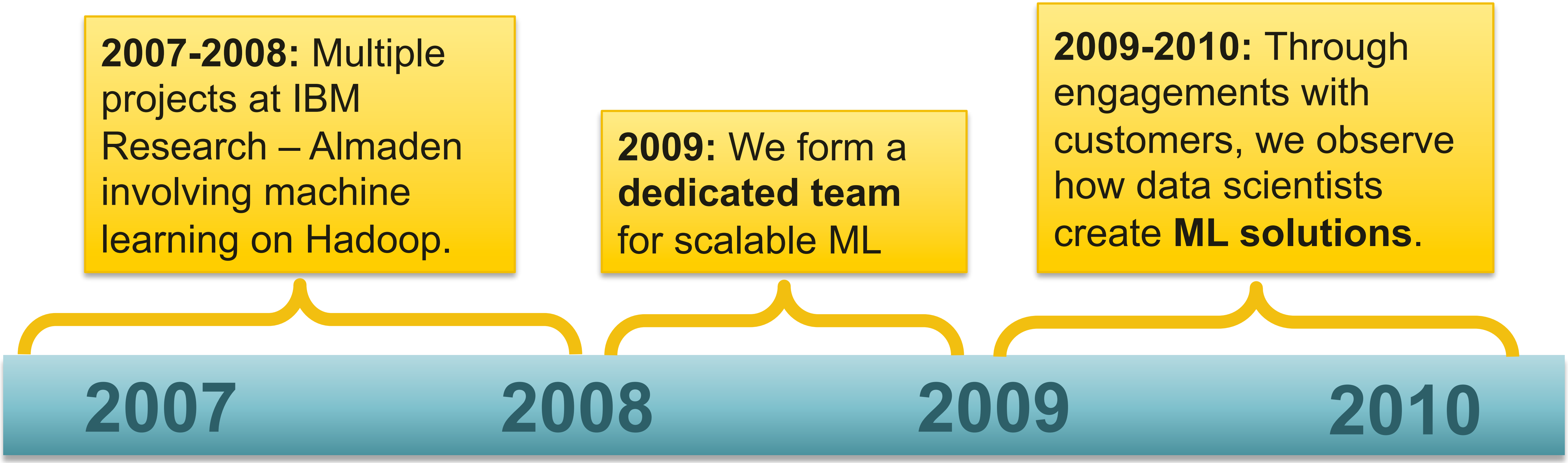
–Assembler vs. Python?

Why another lib?

- Custom machine learning algorithms
- Declarative ML
- Transparent distribution on data-parallel framework
  - Scale-up
  - Scale-out
- Cost-based optimiser generates low level execution plans

Why on Spark?

- Unification of SQL, Graph, Stream, ML
- Common RDD structure
- General DAG execution engine
    - lazy evaluation
    - distributed in-memory caching



2007-2008: Multiple projects at IBM Research – Almaden involving machine learning on Hadoop.

2009: We form a **dedicated team** for scalable ML

2009-2010: Through engagements with customers, we observe how data scientists create **ML solutions**.

2007

2008

2009

2010

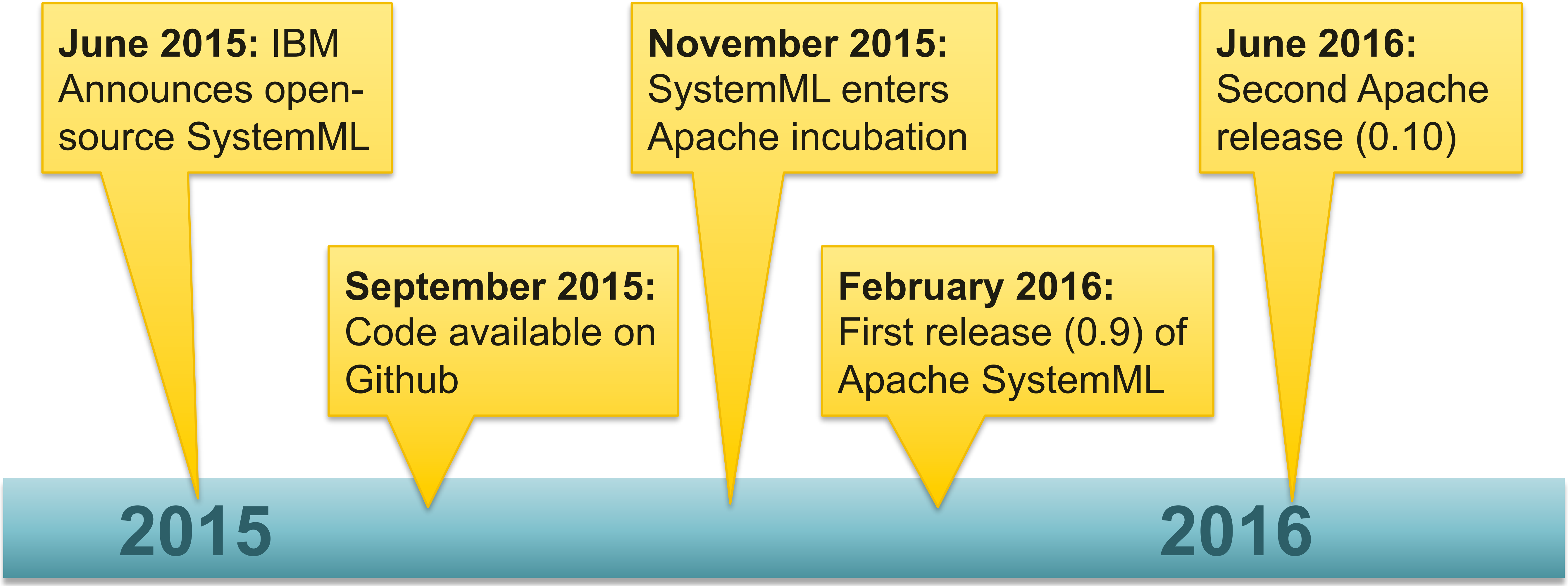
Research

2011

2012

2013

2014



June 2015: IBM Announces open-source SystemML

September 2015: Code available on Github

November 2015: SystemML enters Apache incubation

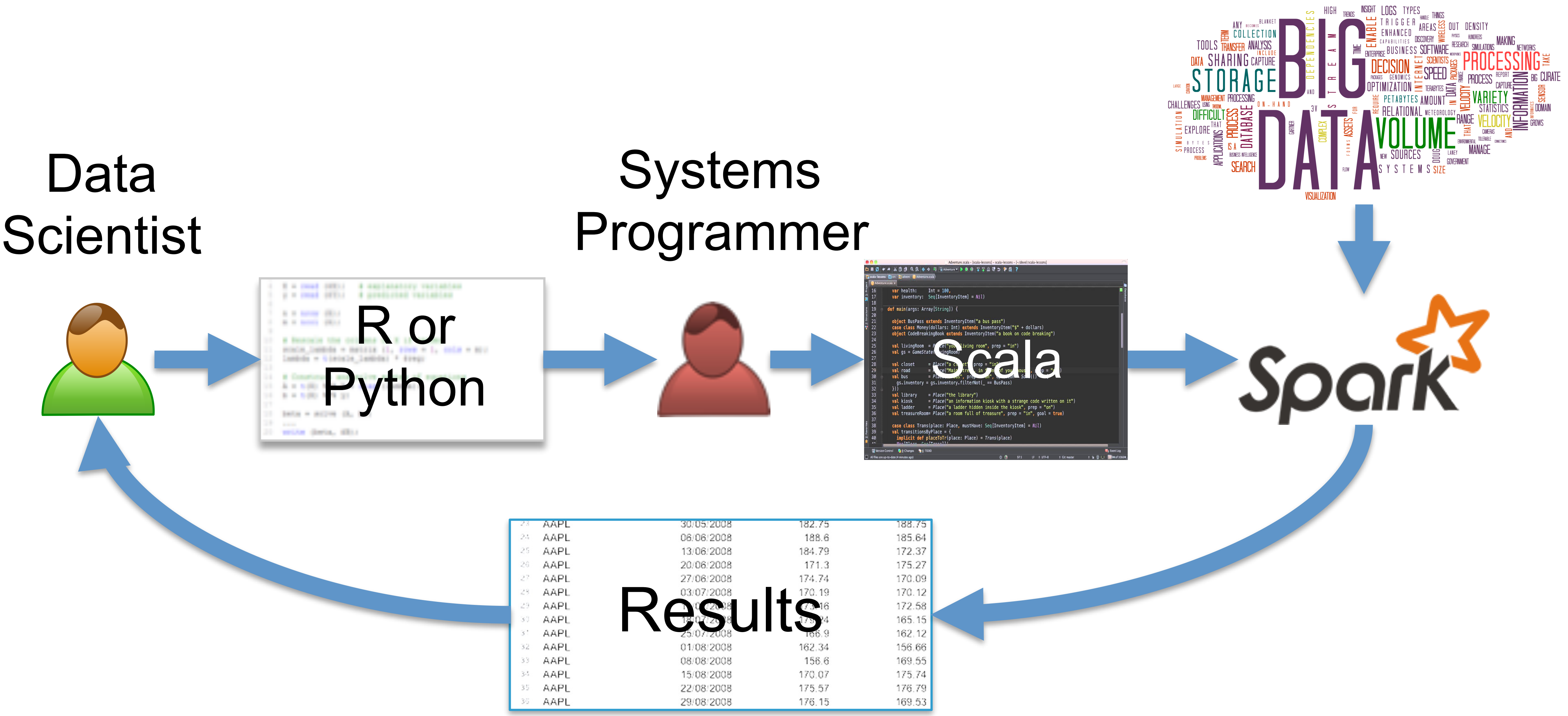
February 2016: First release (0.9) of Apache SystemML

June 2016: Second Apache release (0.10)

SystemML at IBM Watson Health

Moved from Hadoop MapReduce to Spark

- SystemML supports both frameworks
- **Exact same code**
- **300X faster** on 1/40th as many nodes



Data Scientist

R or Python

Systems Programmer

Scala

Results

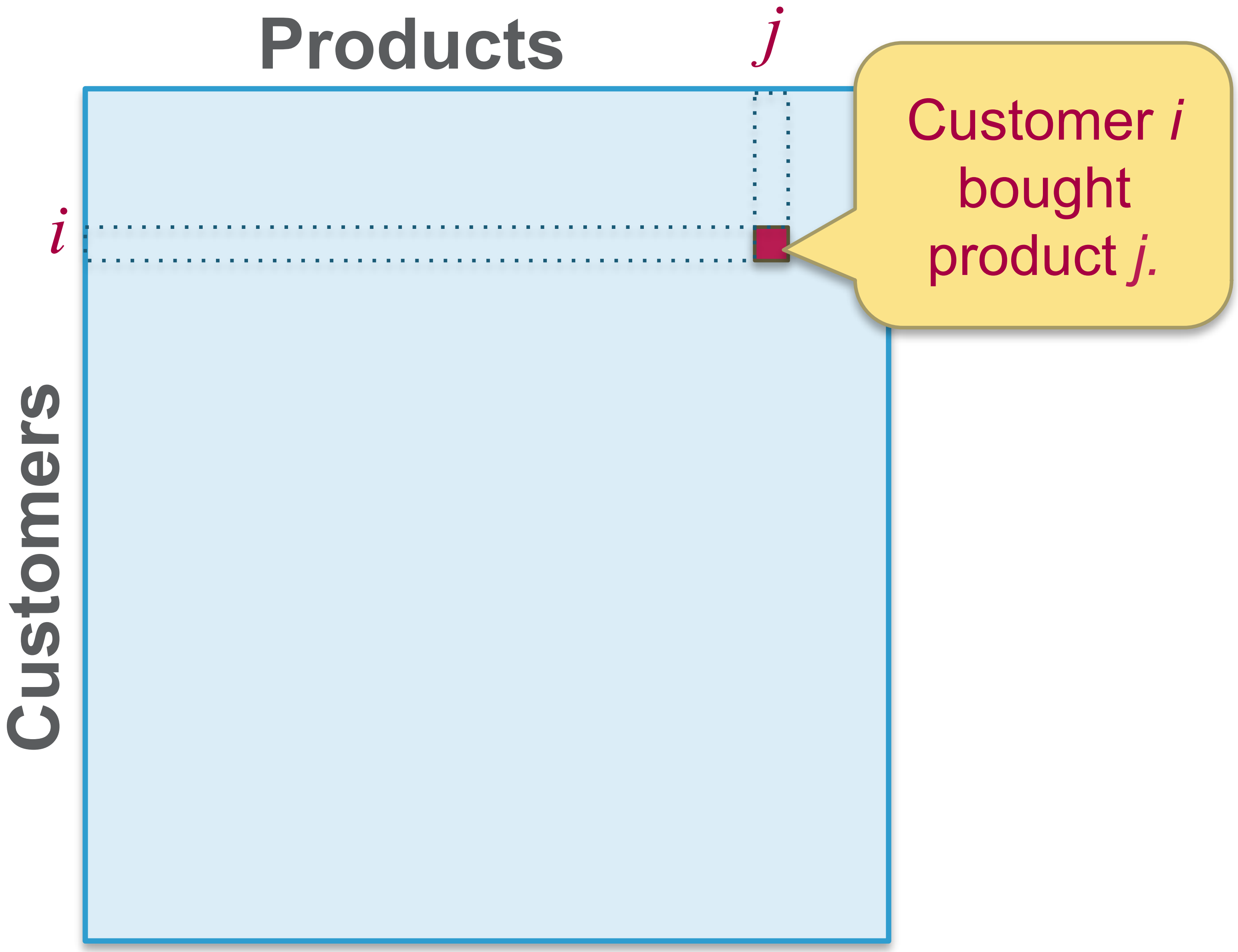
Adventure.scala - [scala-lessons] - scala-lessons - [~/devel/scala-lessons]

```
    var health:     Int = 100,
    var inventory:  Seq[InventoryItem] = Nil)

  def main(args: Array[String]) {

    object BusPass extends InventoryItem("a bus pass")
    case class Money(dollars: Int) extends InventoryItem("$" + dollars)
    object CodeBreakingBook extends InventoryItem("a book on code breaking")

    val livingRoom  = Place("your living room", prep = "in")
    val gs = GameState(livingRoom)

    val closet      = Place("a closet", prep = "in")
    val road        = Place("Main street, in front of your house", prep = "on")
    val bus         = Place("a bus", prep = "on", operation = Some(() => {
      gs.inventory = gs.inventory.filterNot(_ == BusPass)
    }))
    val library     = Place("the library")
    val kiosk       = Place("an information kiosk with a strange code written on it")
    val ladder      = Place("a ladder hidden inside the kiosk", prep = "on")
    val treasureRoom= Place("a room full of treasure", prep = "in", goal = true)

    case class Trans(place: Place, mustHave: Seq[InventoryItem] = Nil)
    val transitionsByPlace = {
      implicit def placeToTr(place: Place) = Trans(place)
```

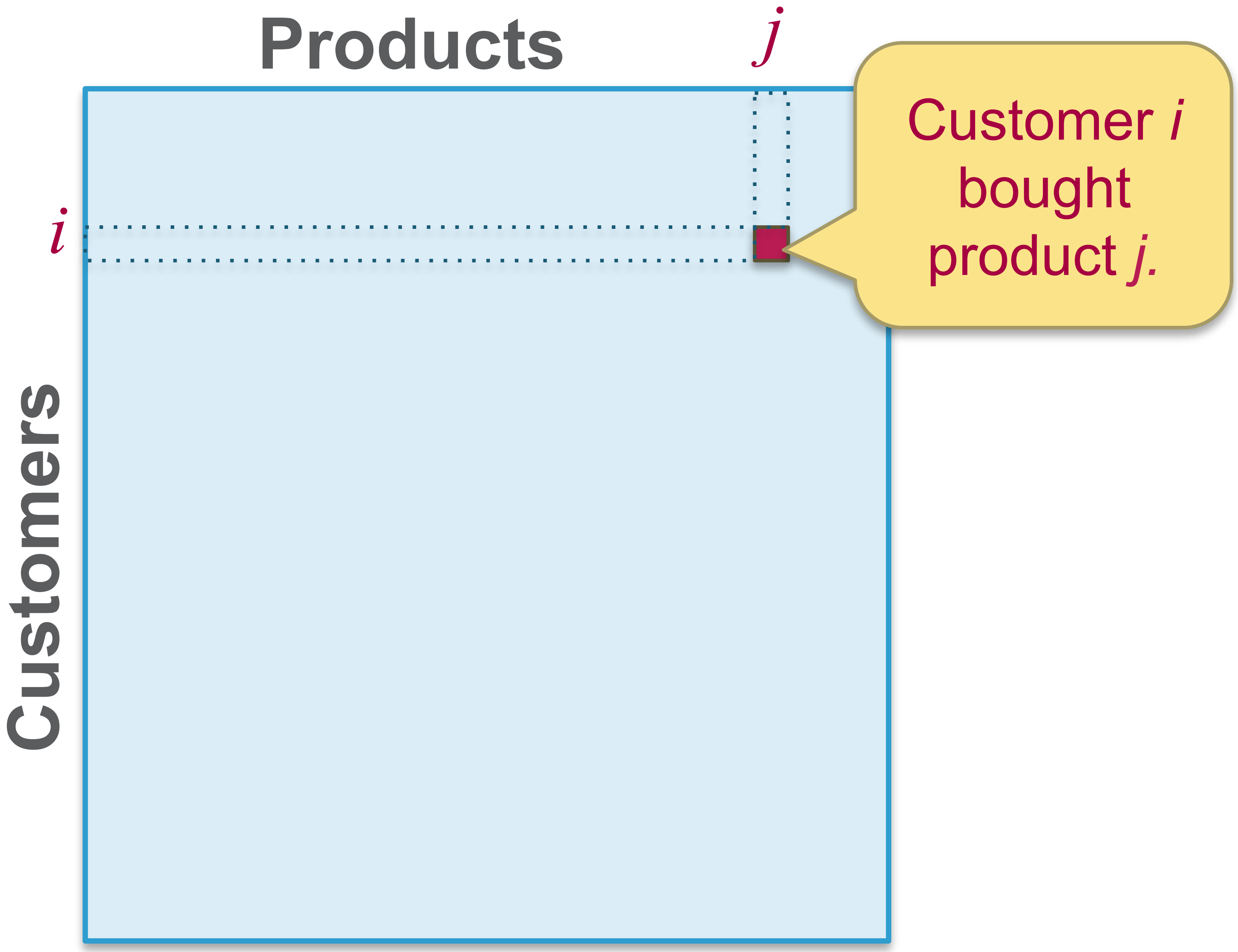
All files are up-to-date (4 minutes ago)

23	AAPL	30/05/2008	182.75	188.75
24	AAPL	06/06/2008	188.6	185.64
25	AAPL	13/06/2008	184.79	172.37
26	AAPL	20/06/2008	171.3	175.27
27	AAPL	27/06/2008	174.74	170.09
28	AAPL	03/07/2008	170.19	170.12
29	AAPL	11/07/2008	173.16	172.58
30	AAPL	18/07/2008	[illegible]	165.15
31	AAPL	25/07/2008	166.9	162.12
32	AAPL	01/08/2008	162.34	156.66
33	AAPL	08/08/2008	156.6	169.55
34	AAPL	15/08/2008	170.07	175.74
35	AAPL	22/08/2008	175.57	176.79
36	AAPL	29/08/2008	176.15	169.53

Alternating Least Squares



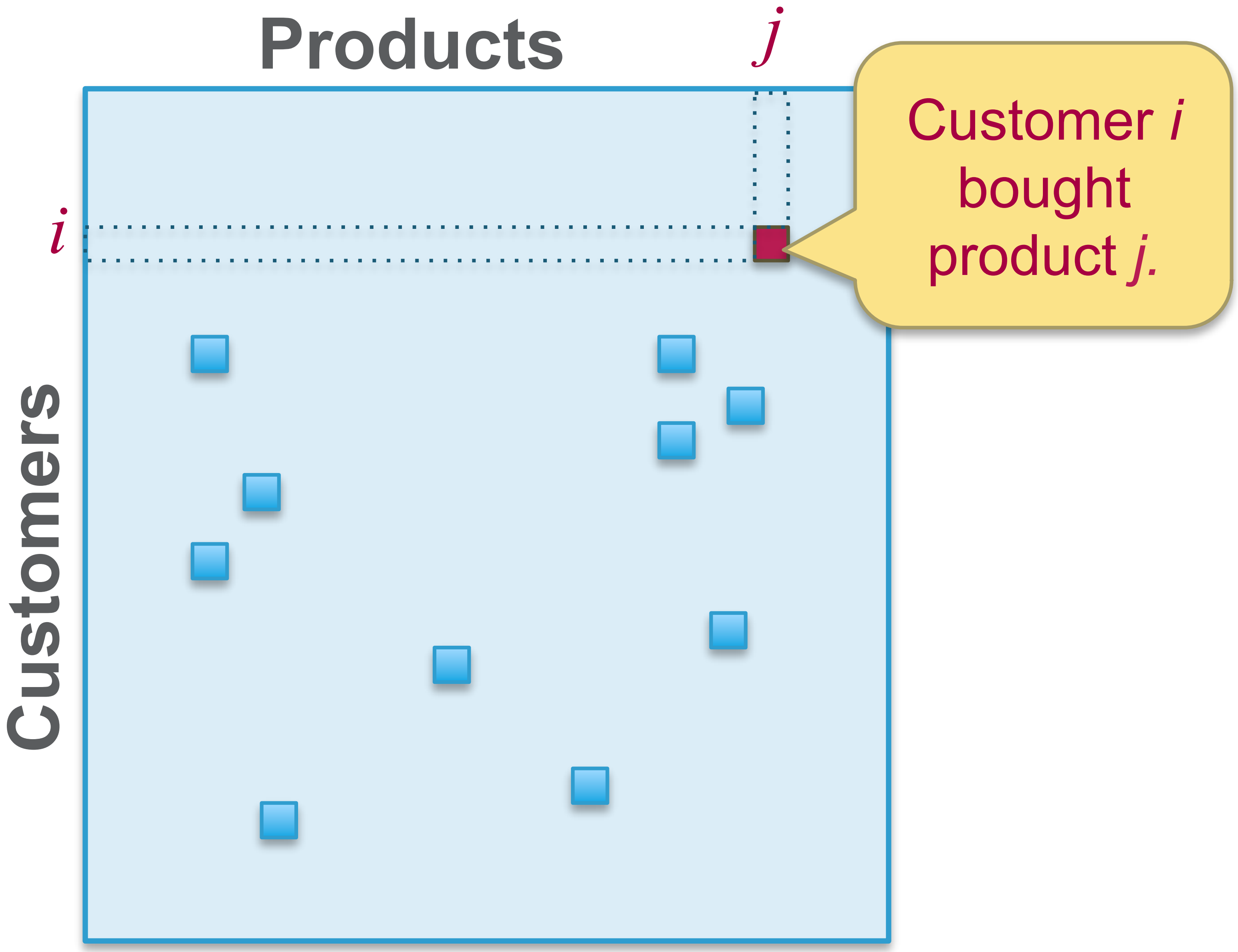
Alternating Least Squares



Customer *i* bought product *j*.

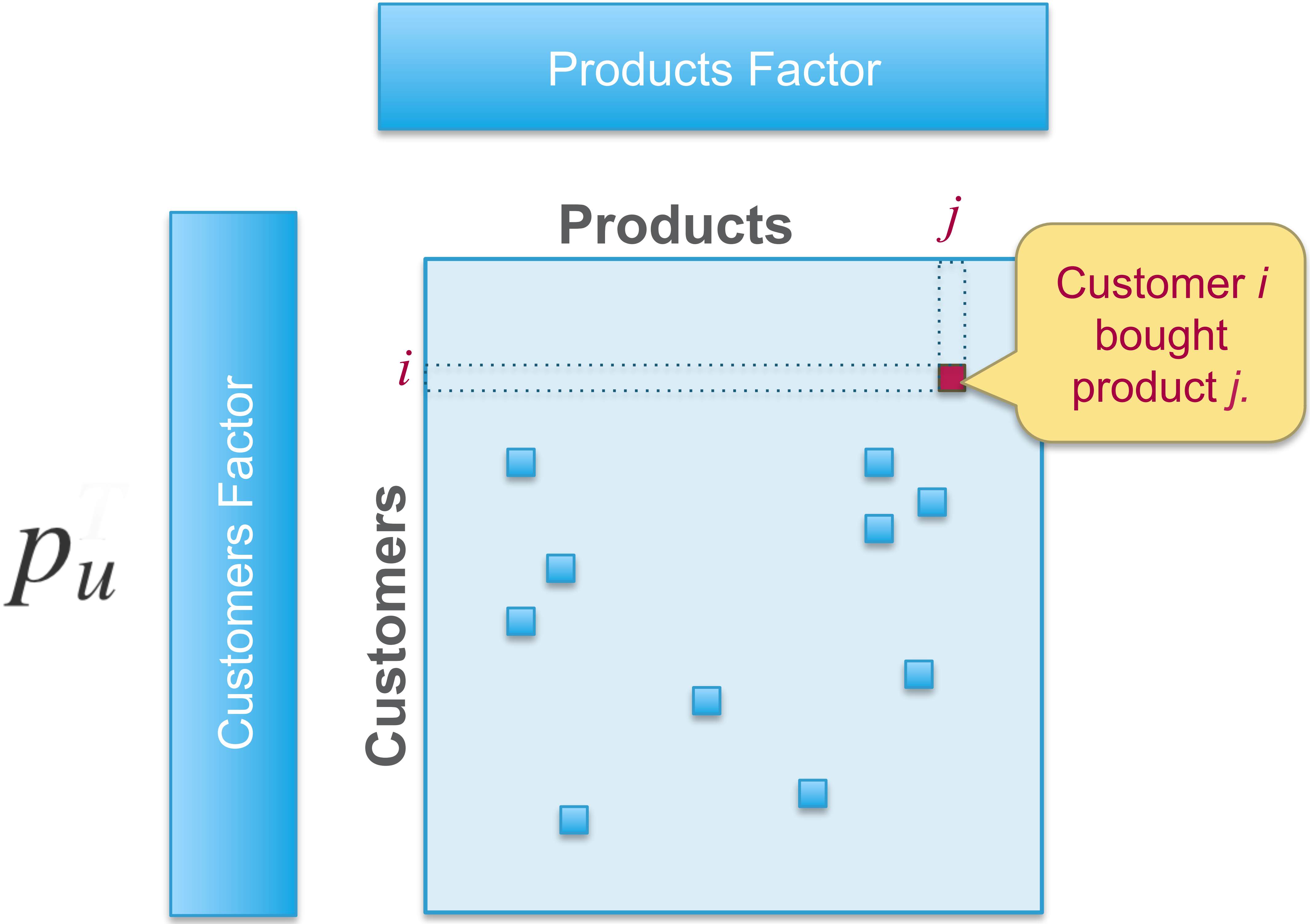
$$r_{ui}$$

Alternating Least Squares



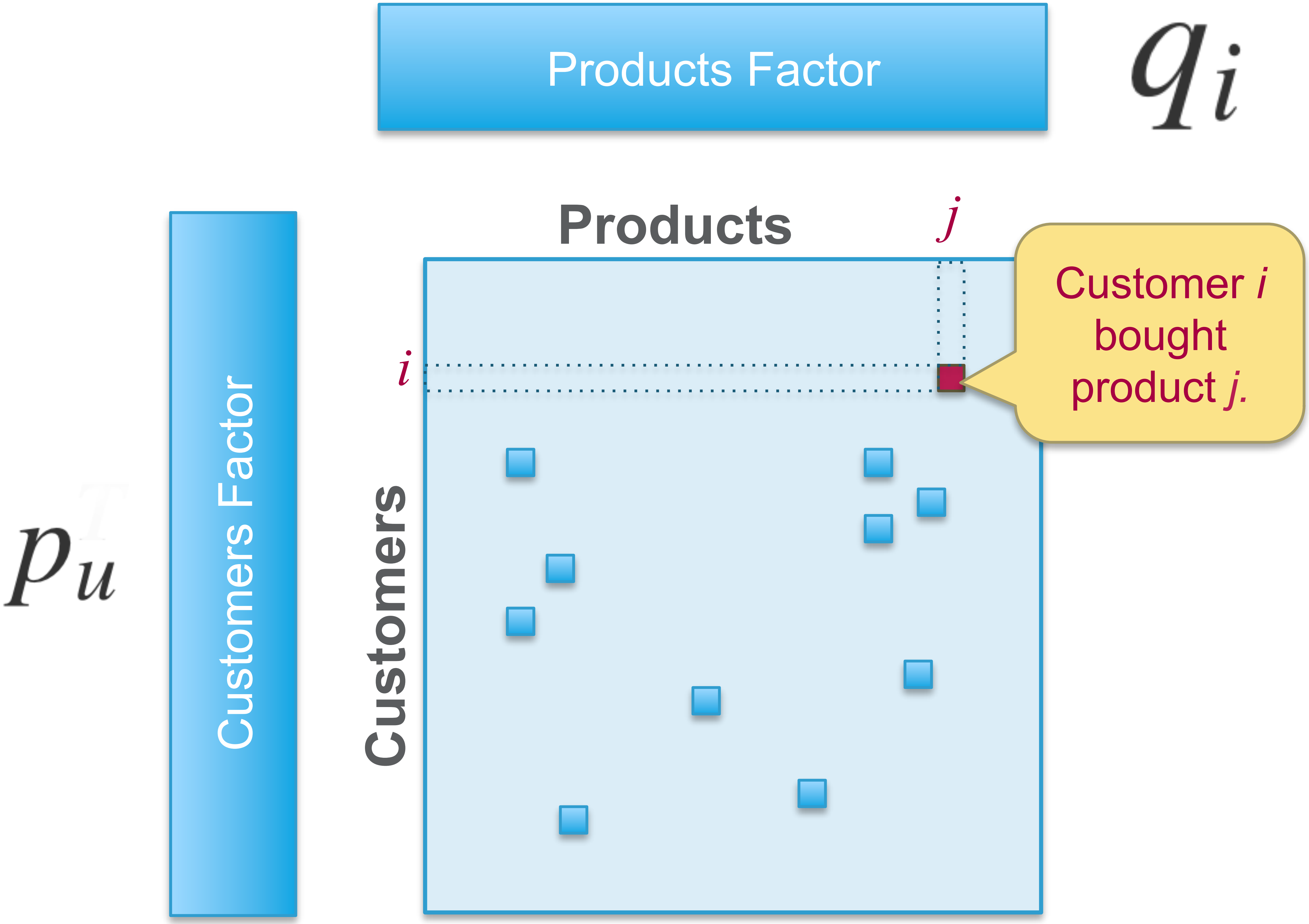
Customer *i* bought product *j*.

$$r_{ui}$$

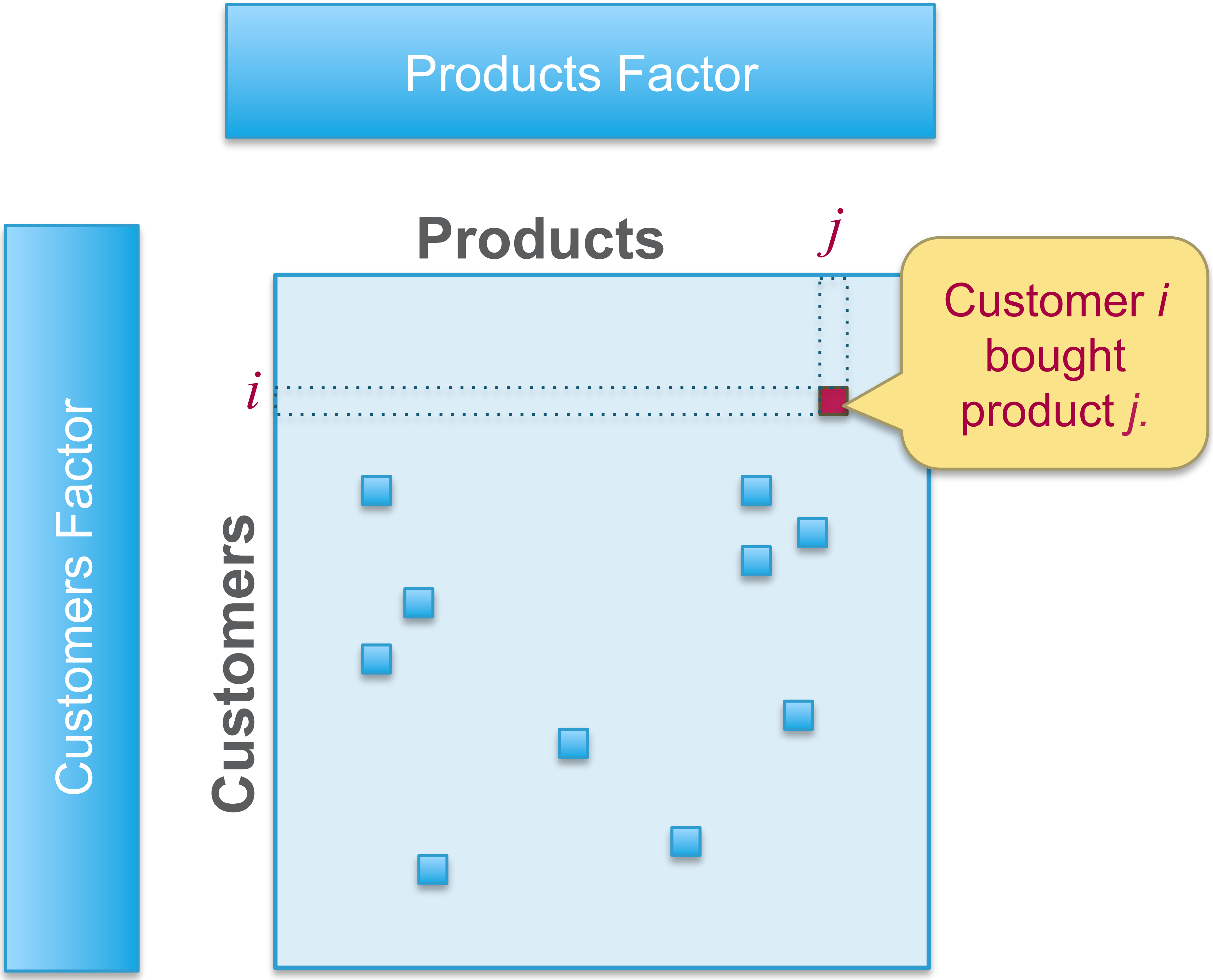


p_u^T

Customer *i* bought product *j*.



Customer *i* bought product *j.*



Products Factor

Customers Factor

Products

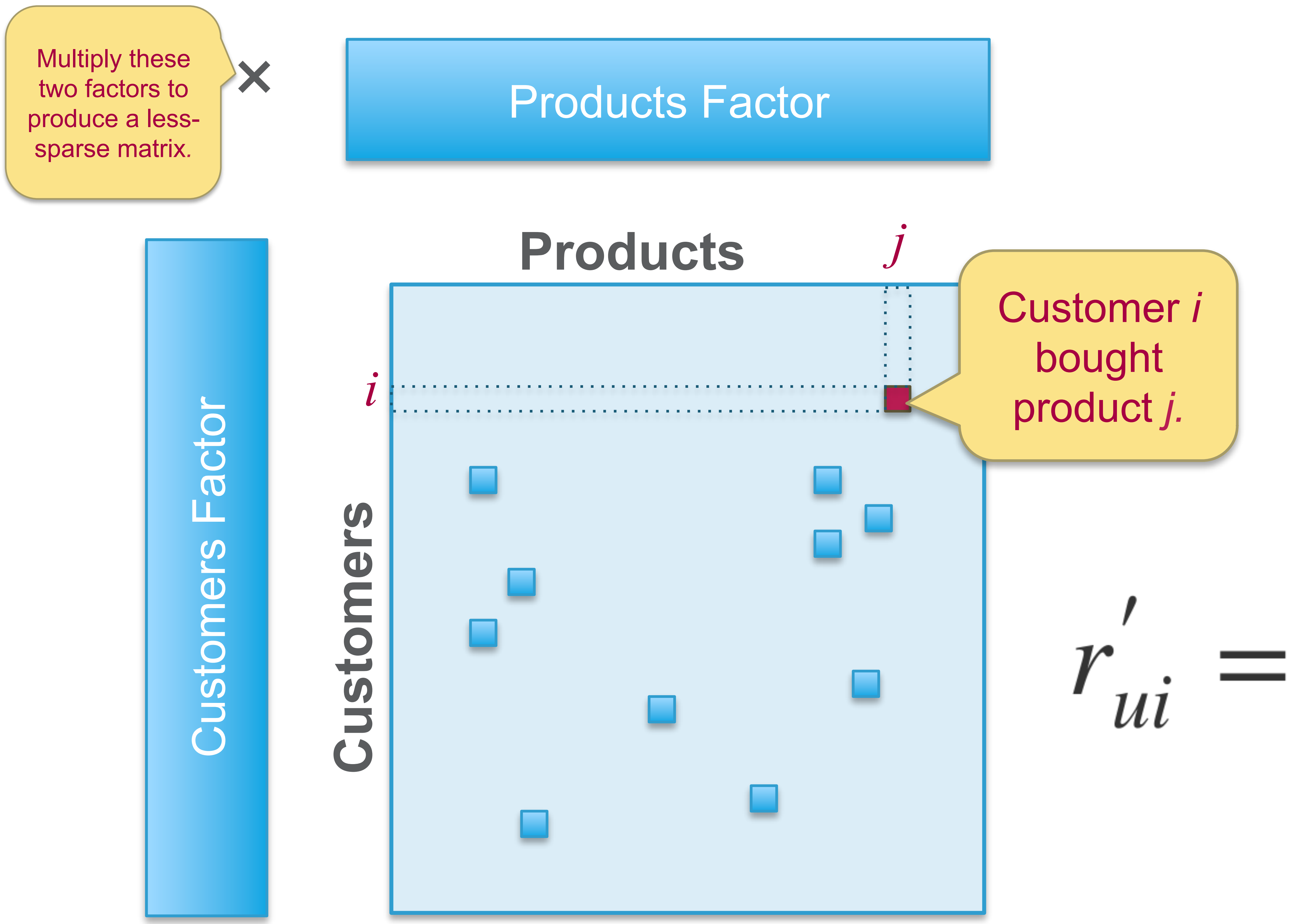
Customers

j

i

Customer *i* bought product *j*.

$$min_{q,p} \sum_{u,i} (r_{ui} - p_u^T q_i)^2$$



Multiply these two factors to produce a less-sparse matrix.

×

Products Factor

Customers Factor

Products

Customers

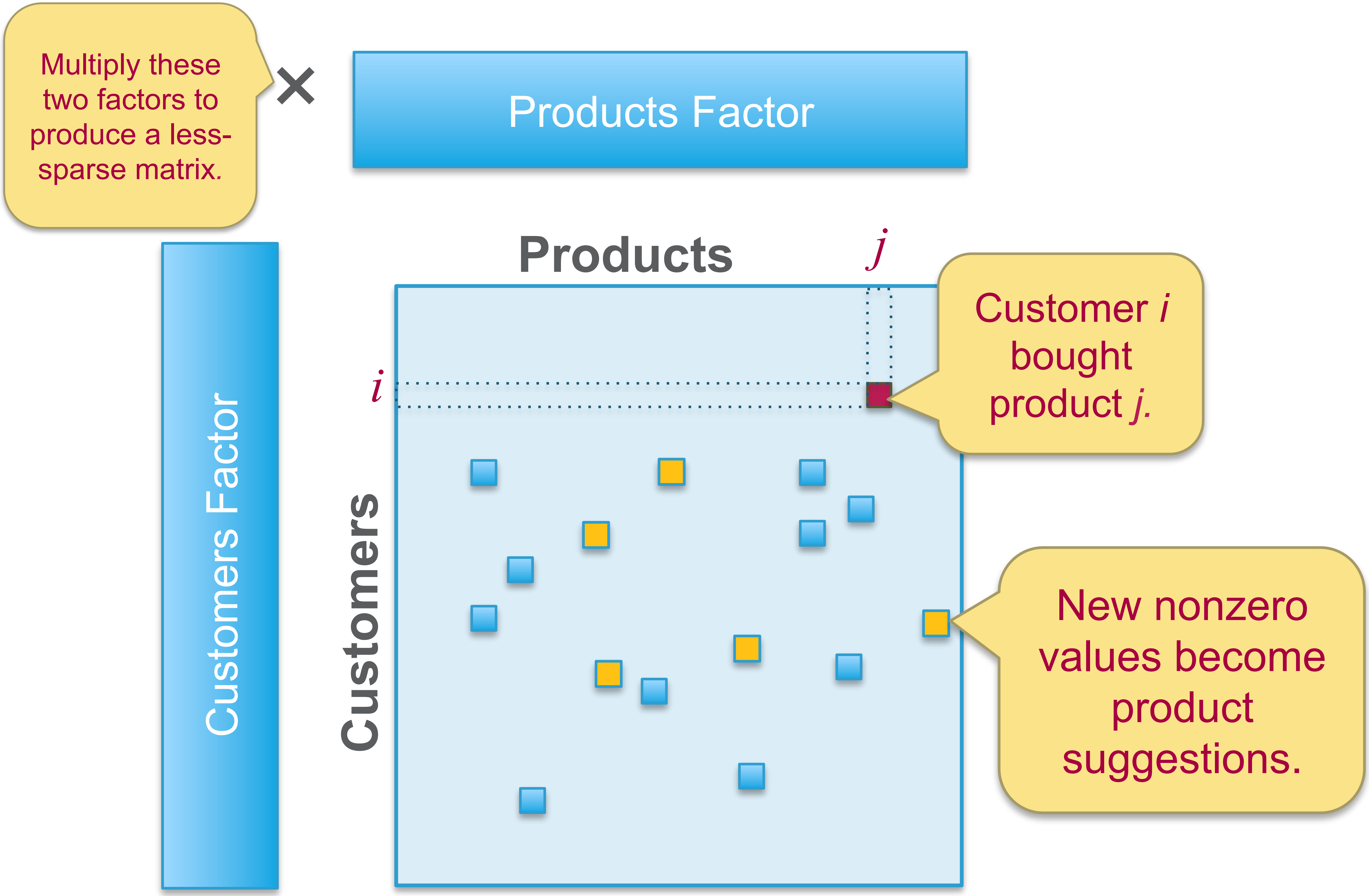
j

i

Customer *i* bought product *j.*

$$r'_{ui} = p_u^T q_i$$

$$min_{q,p} \sum_{u,i} (r_{ui} - p_u^T q_i)^2$$

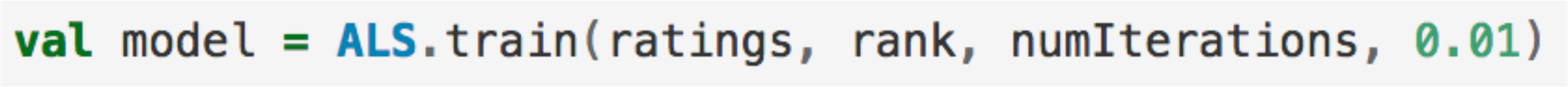


Multiply these two factors to produce a less-sparse matrix.

Customer *i* bought product *j*.

New nonzero values become product suggestions.

$$r'_{ui} = p_u^T q_i$$



```
val model = ALS.train(ratings, rank, numIterations, 0.01)
```

```
U = rand(nrow(X), r, min = -1.0, max = 1.0);
V = rand(r, ncol(X), min = -1.0, max = 1.0);
while(i < mi) {
   i = i + 1; ii = 1;
   if (is_U)
      G = (W * (U %*% V - X)) %*% t(V) + lambda * U;
   else
      G = t(U) %*% (W * (U %*% V - X)) + lambda * V;
   norm_G2 = sum(G ^ 2); norm_R2 = norm_G2;
   R = -G; S = R;
   while(norm_R2 > 10E-9 * norm_G2 & ii <= mii) {
     if (is_U) {
       HS = (W * (S %*% V)) %*% t(V) + lambda * S;
       alpha = norm_R2 / sum (S * HS);
       U = U + alpha * S;
     } else {
       HS = t(U) %*% (W * (U %*% S)) + lambda * S;
       alpha = norm_R2 / sum (S * HS);
       V = V + alpha * S;
     }
     R = R - alpha * HS;
     old_norm_R2 = norm_R2; norm_R2 = sum(R ^ 2);
     S = R + (norm_R2 / old_norm_R2) * S;
     ii = ii + 1;
   }
   is_U = ! is_U;
}
```

```
U = rand(nrow(X), r, min = -1.0, max = 1.0);
V = rand(r, ncol(X), min = -1.0, max = 1.0);
while(i < mi) {
   i = i + 1; ii = 1;
   if (is_U)
      G = (W * (U %*% V - X)) %*% t(V) + lambda * U;
   else
      G = t(U) %*% (W * (U %*% V - X)) + lambda * V;
   norm_G2 = sum(G ^ 2); norm_R2 = norm_G2;
   R = -G; S = R;
   while(norm_R2 > 10E-9 * norm_G2 & ii <= mii) {
     if (is_U) {
       HS = (W * (S %*% V)) %*% t(V) + lambda * S;
       alpha = norm_R2 / sum (S * HS);
       U = U + alpha * S;
     } else {
       HS = t(U) %*% (W * (U %*% S)) + lambda * S;
       alpha = norm_R2 / sum (S * HS);
       V = V + alpha * S;
     }
     R = R - alpha * HS;
     old_norm_R2 = norm_R2; norm_R2 = sum(R ^ 2);
     S = R + (norm_R2 / old_norm_R2) * S;
     ii = ii + 1;
   }
   is_U = ! is_U;
}
```

```
U = rand(nrow(X), r, min = -1.0, max = 1.0);
V = rand(r, ncol(X), min = -1.0, max = 1.0);
while(i < mi) {
   i = i + 1; ii = 1;
   if (is_U)
      G = (W * (U %*% V - X)) %*% t(V) + lambda * U;
   else
      G = t(U) %*% (W * (U %*% V - X)) + lambda * V;
   norm_G2 = sum(G ^ 2); norm_R2 = norm_G2;
   R = -G; S = R;
   while(norm_R2 > 10E-9 * norm_G2 & ii <= mii) {
     if (is_U) {
       HS = (W * (S %*% V)) %*% t(V) + lambda * S;
       alpha = norm_R2 / sum (S * HS);
       U = U + alpha * S;
     } else {
       HS = t(U) %*% (W * (U %*% S)) + lambda * S;
       alpha = norm_R2 / sum (S * HS);
       V = V + alpha * S;
     }
     R = R - alpha * HS;
     old_norm_R2 = norm_R2; norm_R2 = sum(R ^ 2);
     S = R + (norm_R2 / old_norm_R2) * S;
     ii = ii + 1;
   }
   is_U = ! is_U;
}
```

```
U = rand(nrow(X), r, min = -1.0, max = 1.0);
V = rand(r, ncol(X), min = -1.0, max = 1.0);
while(i < mi) {
   i = i + 1; ii = 1;
   if (is_U)
      G = (W * (U %*% V - X)) %*% t(V) + lambda * U;
   else
      G = t(U) %*% (W * (U %*% V - X)) + lambda * V;
   norm_G2 = sum(G ^ 2); norm_R2 = norm_G2;
   R = -G; S = R;
   while(norm_R2 > 10E-9 * norm_G2 & ii <= mii) {
     if (is_U) {
       HS = (W * (S %*% V)) %*% t(V) + lambda * S;
       alpha = norm_R2 / sum (S * HS);
       U = U + alpha * S;
     } else {
       HS = t(U) %*% (W * (U %*% S)) + lambda * S;
       alpha = norm_R2 / sum (S * HS);
       V = V + alpha * S;
     }
     R = R - alpha * HS;
     old_norm_R2 = norm_R2; norm_R2 = sum(R ^ 2);
     S = R + (norm_R2 / old_norm_R2) * S;
     ii = ii + 1;
   }
   is_U = ! is_U;
}
```

```
U = rand(nrow(X), r, min = -1.0, max = 1.0);
V = rand(r, ncol(X), min = -1.0, max = 1.0);
while(i < mi) {
   i = i + 1; ii = 1;
   if (is_U)
      G = (W * (U %*% V - X)) %*% t(V) + lambda * U;
   else
      G = t(U) %*% (W * (U %*% V - X)) + lambda * V;
   norm_G2 = sum(G ^ 2); norm_R2 = norm_G2;
   R = -G; S = R;
   while(norm_R2 > 10E-9 * norm_G2 & ii <= mii) {
     if (is_U) {
       HS = (W * (S %*% V)) %*% t(V) + lambda * S;
       alpha = norm_R2 / sum (S * HS);
       U = U + alpha * S;
     } else {
       HS = t(U) %*% (W * (U %*% S)) + lambda * S;
       alpha = norm_R2 / sum (S * HS);
       V = V + alpha * S;
     }
     R = R - alpha * HS;
     old_norm_R2 = norm_R2; norm_R2 = sum(R ^ 2);
     S = R + (norm_R2 / old_norm_R2) * S;
     ii = ii + 1;
   }
   is_U = ! is_U;
}
```

Every line has a clear purpose!

https://github.com/apache/spark/blob/master/mllib/src/main/scala/org/apache/spark/mllib/recommendation/ALS.scala https://github.com/apache/spark/blob/master/mllib/src/main/scala/org/apache/spark/mllib/recommendation/ALS.scala

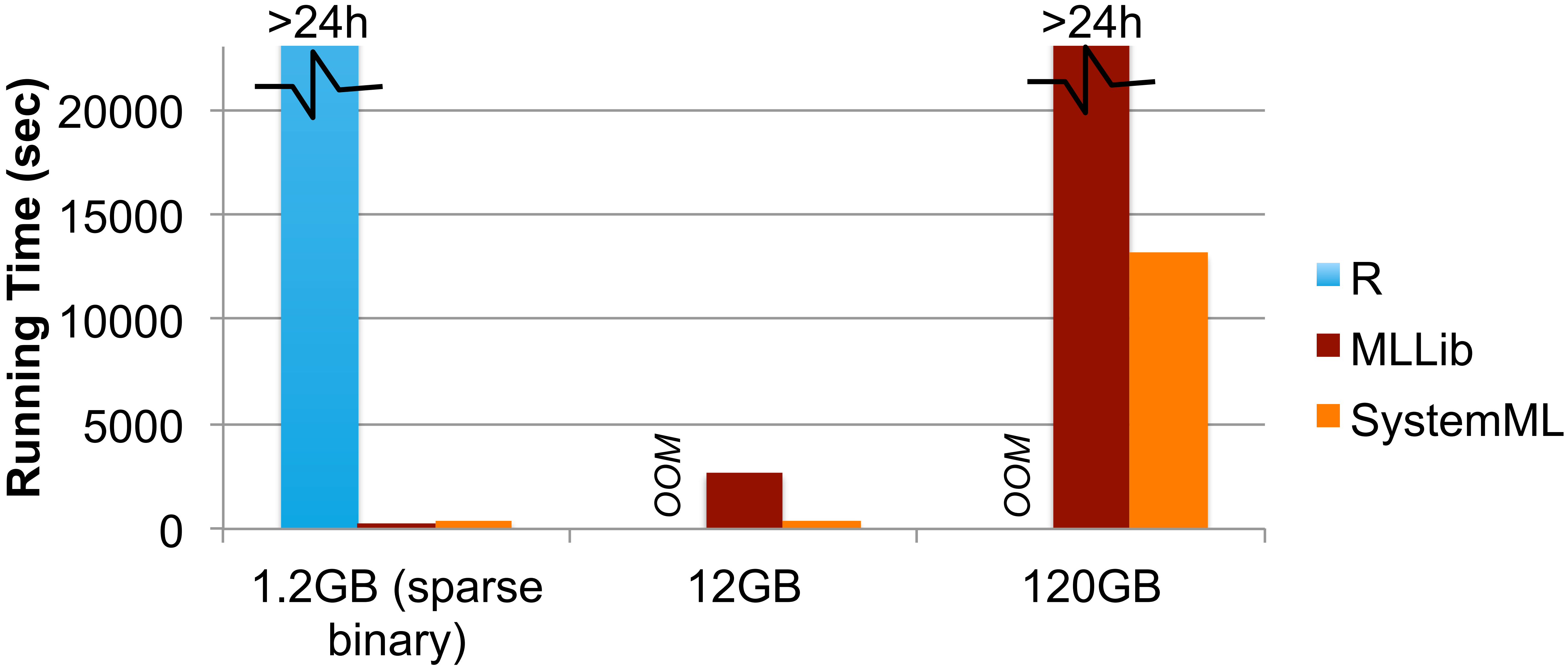
25 lines’ worth of algorithm…

…mixed with 800 lines of performance code

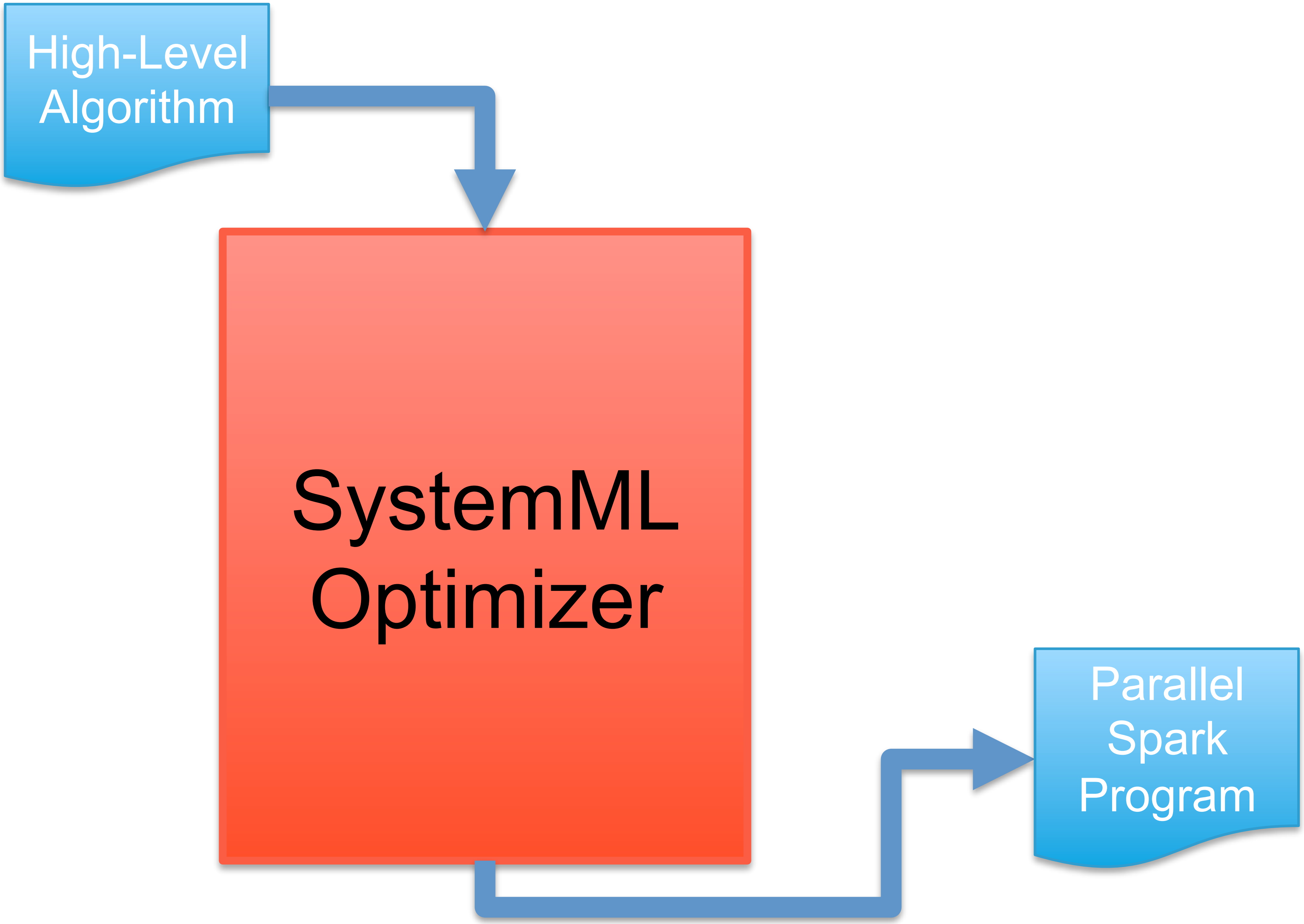
```
U = rand(nrow(X), r, min = -1.0, max = 1.0);
V = rand(r, ncol(X), min = -1.0, max = 1.0);
while(i < mi) {
   i = i + 1; ii = 1;
   if (is_U)
      G = (W * (U %*% V - X)) %*% t(V) + lambda * U;
   else
      G = t(U) %*% (W * (U %*% V - X)) + lambda * V;
   norm_G2 = sum(G ^ 2); norm_R2 = norm_G2;
   R = -G; S = R;
   while(norm_R2 > 10E-9 * norm_G2 & ii <= mii) {
     if (is_U) {
       HS = (W * (S %*% V)) %*% t(V) + lambda * S;
       alpha = norm_R2 / sum (S * HS);
       U = U + alpha * S;
     } else {
       HS = t(U) %*% (W * (U %*% S)) + lambda * S;
       alpha = norm_R2 / sum (S * HS);
       V = V + alpha * S;
     }
     R = R - alpha * HS;
     old_norm_R2 = norm_R2; norm_R2 = sum(R ^ 2);
     S = R + (norm_R2 / old_norm_R2) * S;
     ii = ii + 1;
   }
   is_U = ! is_U;
}
```

```
U = rand(nrow(X), r, min = -1.0, max = 1.0);
V = rand(r, ncol(X), min = -1.0, max = 1.0);
while(i < mi) {
   i = i + 1; ii = 1;
   if (is_U)
      G = (W * (U %*% V - X)) %*% t(V) + lambda * U;
   else
      G = t(U) %*% (W * (U %*% V - X)) + lambda * V;
   norm_G2 = sum(G ^ 2); norm_R2 = norm_G2;
   R = -G; S = R;
   while(norm_R2 > 10E-9 * norm_G2 & ii <= mii) {
     if (is_U) {
       HS = (W * (S %*% V)) %*% t(V) + lambda * S;
       alpha = norm_R2 / sum (S * HS);
       U = U + alpha * S;
     } else {
       HS = t(U) %*% (W * (U %*% S)) + lambda * S;
       alpha = norm_R2 / sum (S * HS);
       V = V + alpha * S;
     }
     R = R - alpha * HS;
     old_norm_R2 = norm_R2; norm_R2 = sum(R ^ 2);
     S = R + (norm_R2 / old_norm_R2) * S;
     ii = ii + 1;
   }
   is_U = ! is_U;
}
```

SystemML:
compile and run at scale
no performance code needed!


>24h
>24h
Running Time (sec)
20000
15000
10000
5000
0
OOM
OOM
1.2GB (sparse binary)
12GB
120GB
R
MLLib
SystemML

Architecture

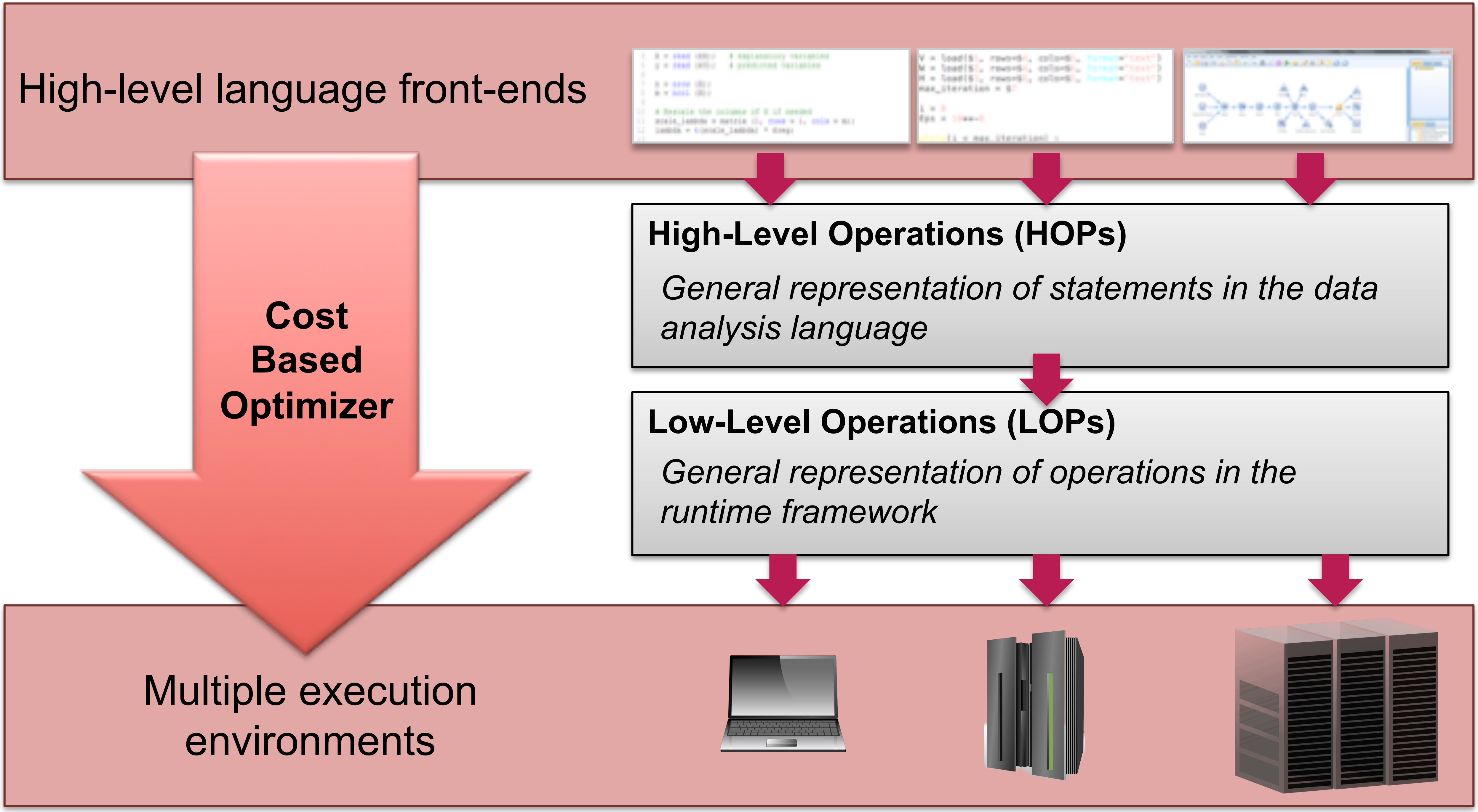


High-Level Algorithm

SystemML Optimizer

Parallel Spark Program

Architecture



High-level language front-ends

Cost Based Optimizer

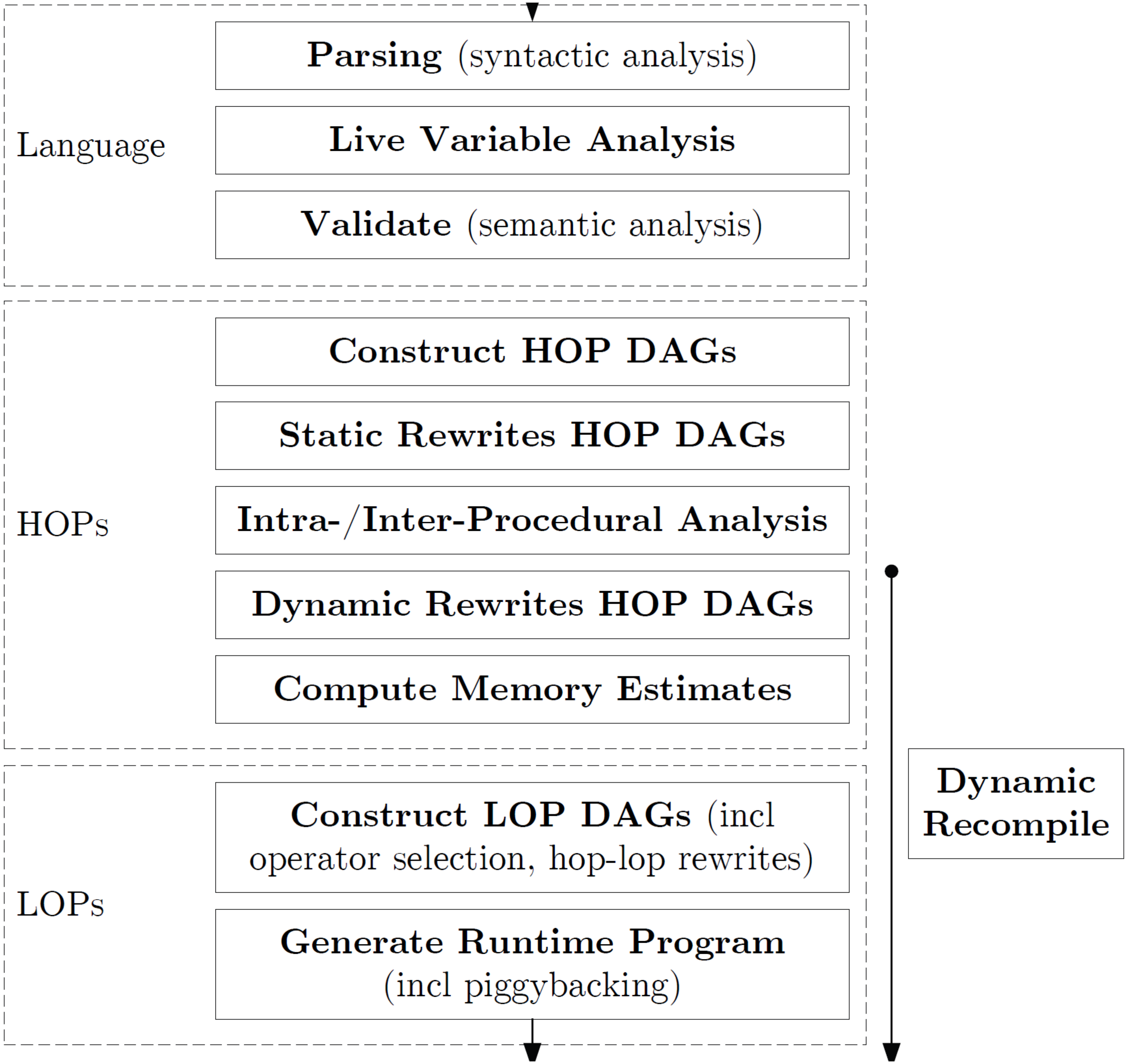
High-Level Operations (HOPs)

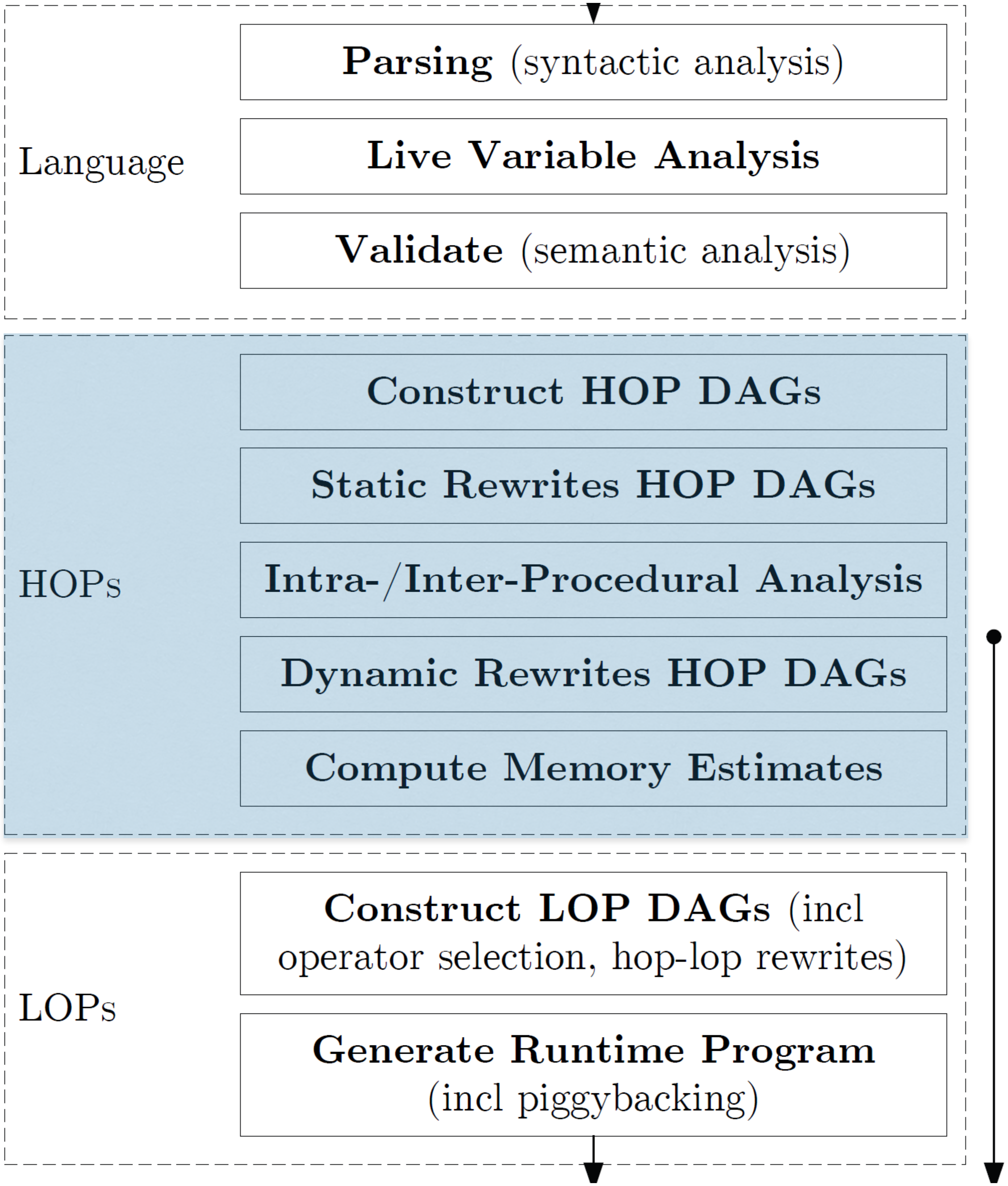
General representation of statements in the data analysis language

Low-Level Operations (LOPs)

General representation of operations in the runtime framework

Multiple execution environments


Language
Parsing (syntactic analysis)
Live Variable Analysis
Validate (semantic analysis)
HOPs
Construct HOP DAGs
Static Rewrites HOP DAGs
Intra-/Inter-Procedural Analysis
Dynamic Rewrites HOP DAGs
Compute Memory Estimates
LOPs
Construct LOP DAGs (incl operator selection, hop-lop rewrites)
Generate Runtime Program (incl piggybacking)
Dynamic Recompile



Language

Parsing (syntactic analysis)

Live Variable Analysis

Validate (semantic analysis)

HOPs

Construct HOP DAGs

Static Rewrites HOP DAGs

Intra-/Inter-Procedural Analysis

Dynamic Rewrites HOP DAGs

Compute Memory Estimates

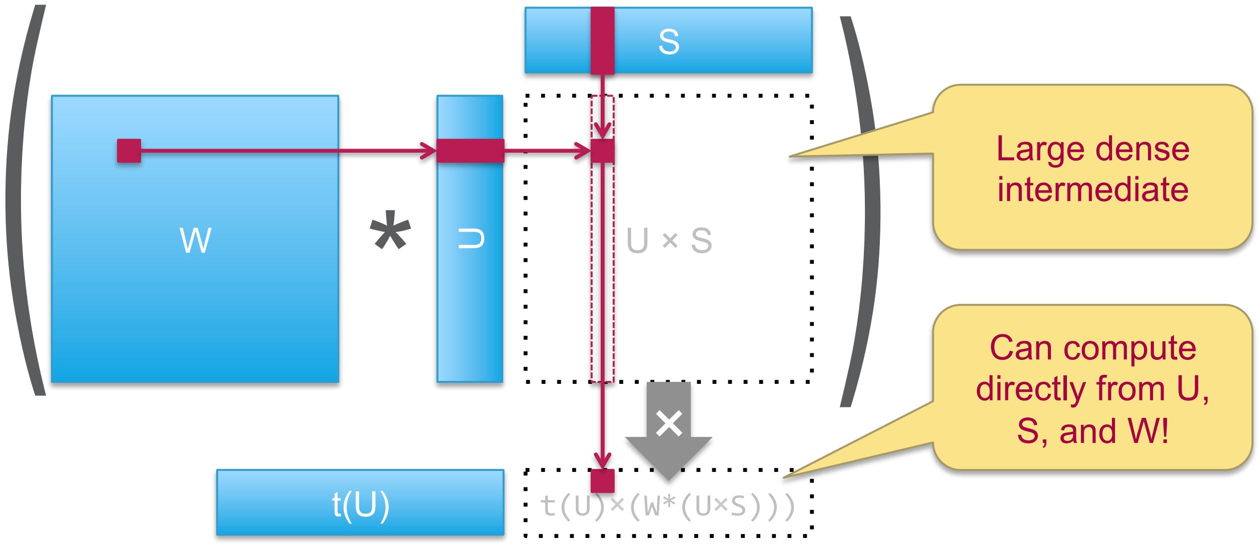
LOPs

Construct LOP DAGs (incl operator selection, hop-lop rewrites)

Generate Runtime Program (incl piggybacking)

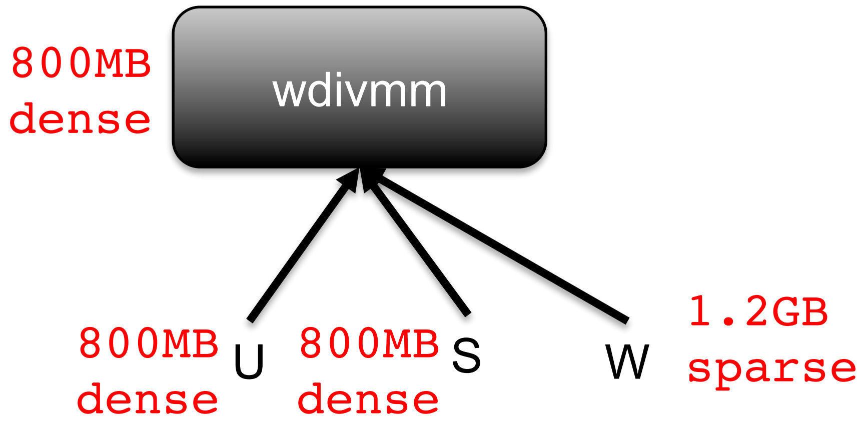
Dynamic Recompile

`t(U) %*% (W * (U %*% S))`

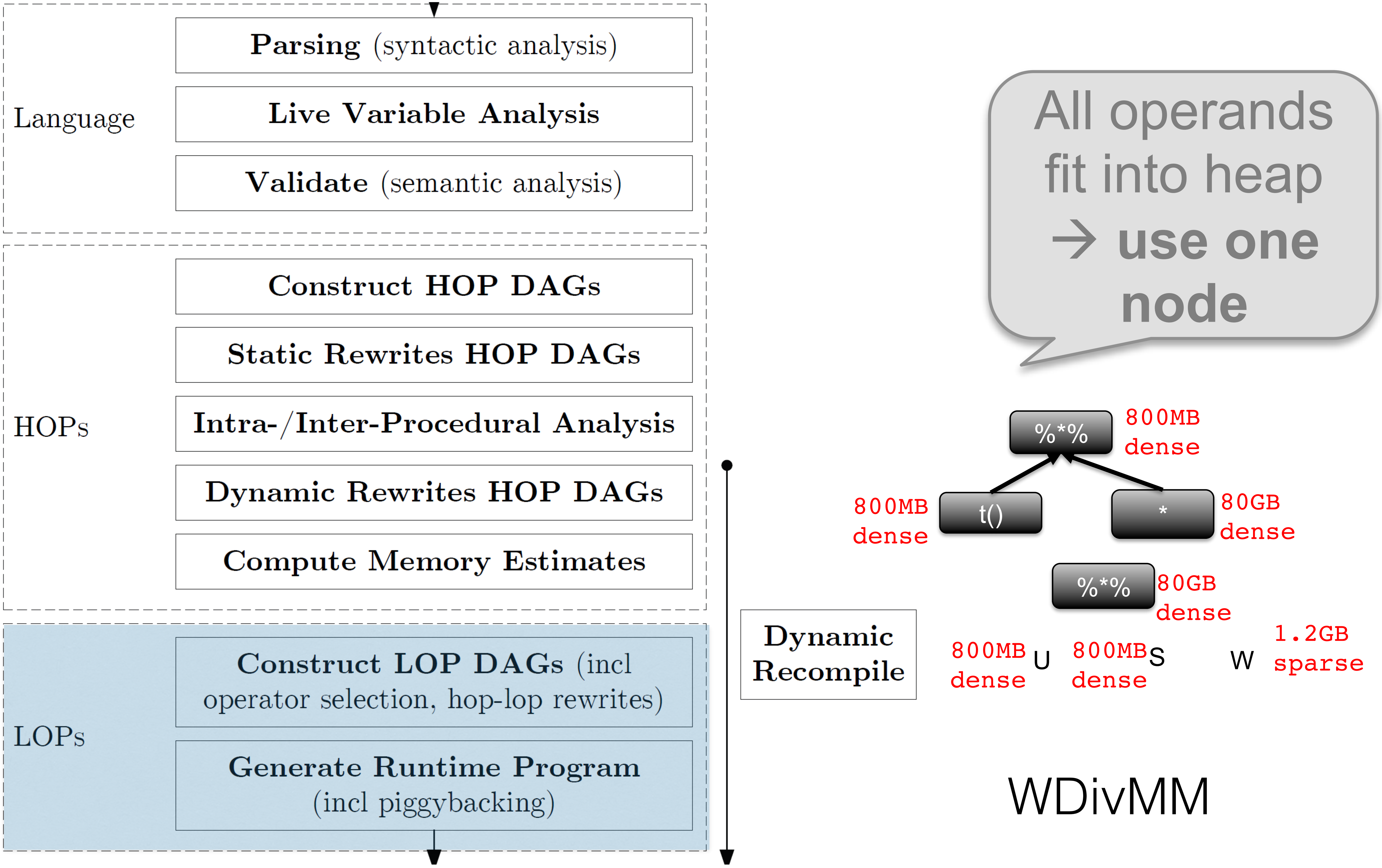


Large dense intermediate

Can compute directly from U, S, and W!



(weighted divide matrix multiplication)



Language

Parsing (syntactic analysis)

Live Variable Analysis

Validate (semantic analysis)

HOPs

Construct HOP DAGs

Static Rewrites HOP DAGs

Intra-/Inter-Procedural Analysis

Dynamic Rewrites HOP DAGs

Compute Memory Estimates

LOPs

Construct LOP DAGs (incl operator selection, hop-lop rewrites)

Generate Runtime Program (incl piggybacking)

Dynamic Recompile

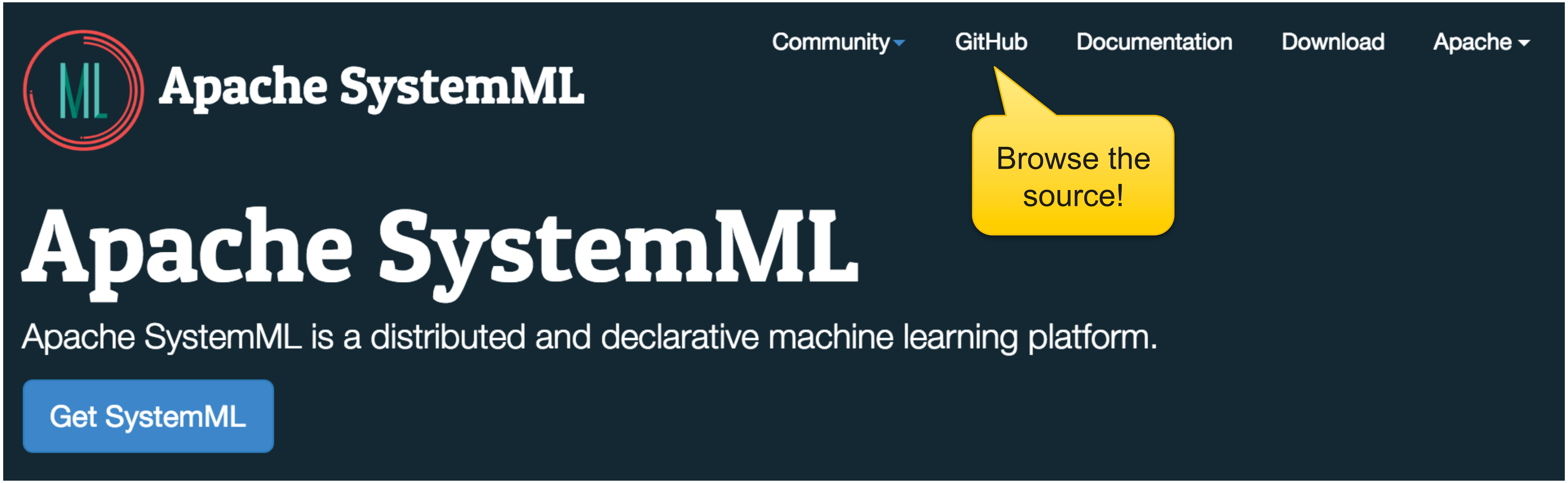
All operands fit into heap → **use one node**

WDivMM

(MapWDivMM)

Apache SystemML

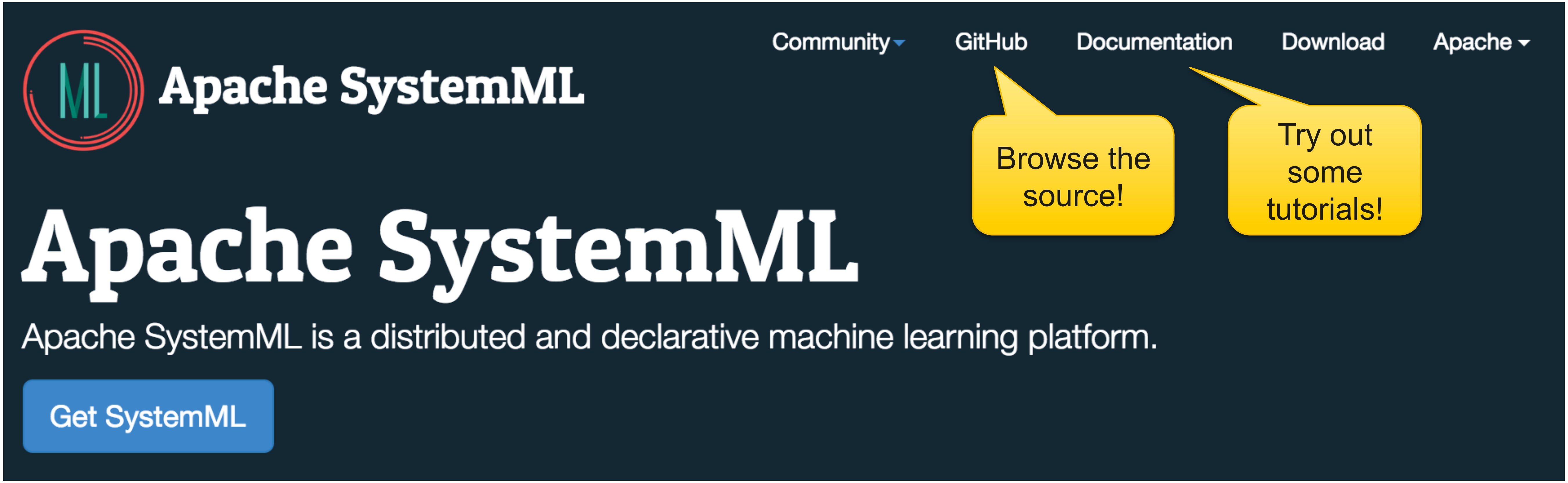
Apache SystemML is a distributed and declarative machine learning platform.



Apache SystemML

Apache SystemML is a distributed and declarative machine learning platform.

Browse the source!

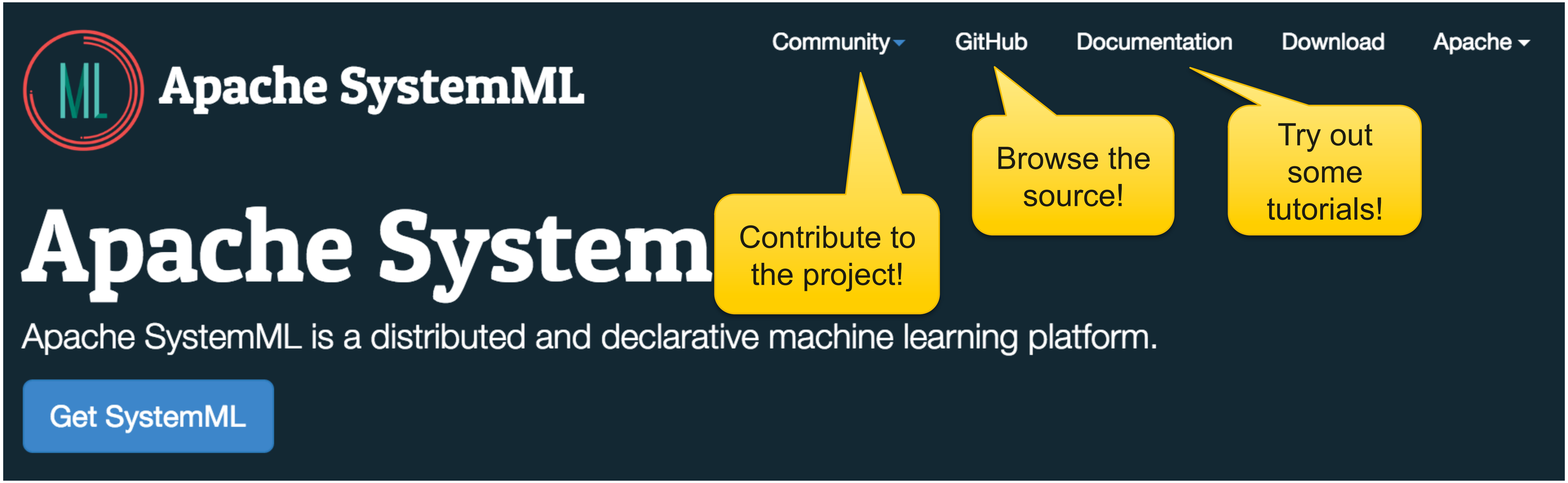


Apache SystemML

Apache SystemML is a distributed and declarative machine learning platform.

Browse the source!

Try out some tutorials!



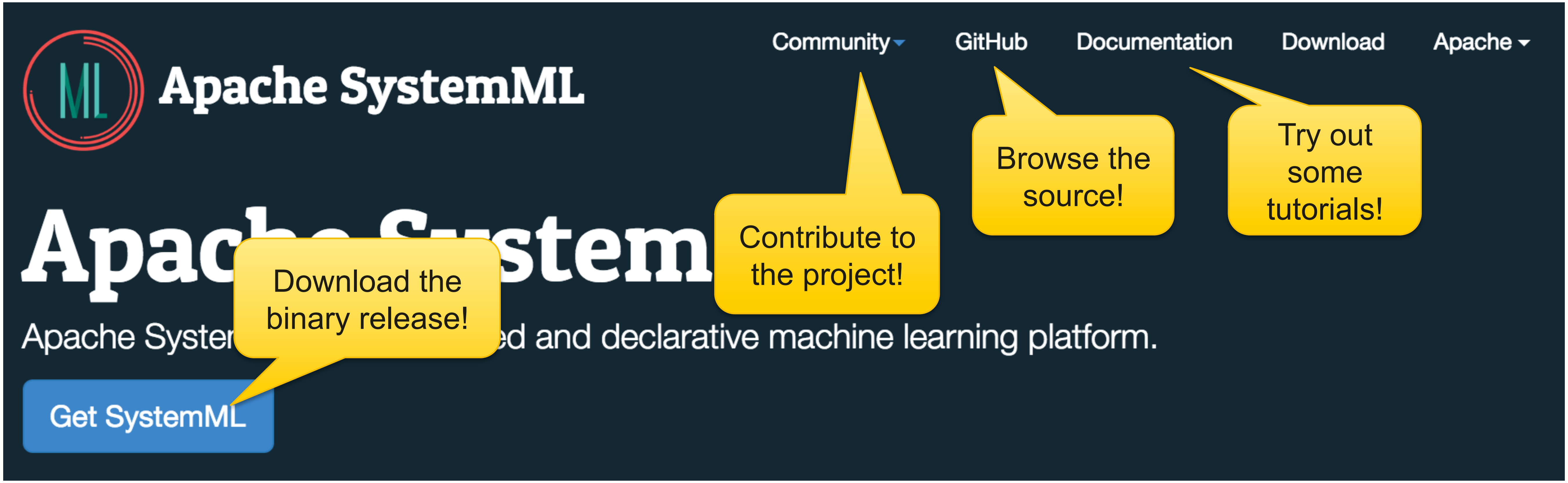
Apache System

Apache SystemML is a distributed and declarative machine learning platform.

Contribute to the project!

Browse the source!

Try out some tutorials!



Apache SystemML

Apach System

Apache Syster ed and declarative machine learning platform.

Contribute to the project!

Browse the source!

Try out some tutorials!

Download the binary release!

Demo