Web-Lar

Matrix computation web-service

Luca Menichetti, Fabrizio Rebecca

(1)
DESCRIPTION & SUMMARY

DESCRIPTION & SUMMARY

Description

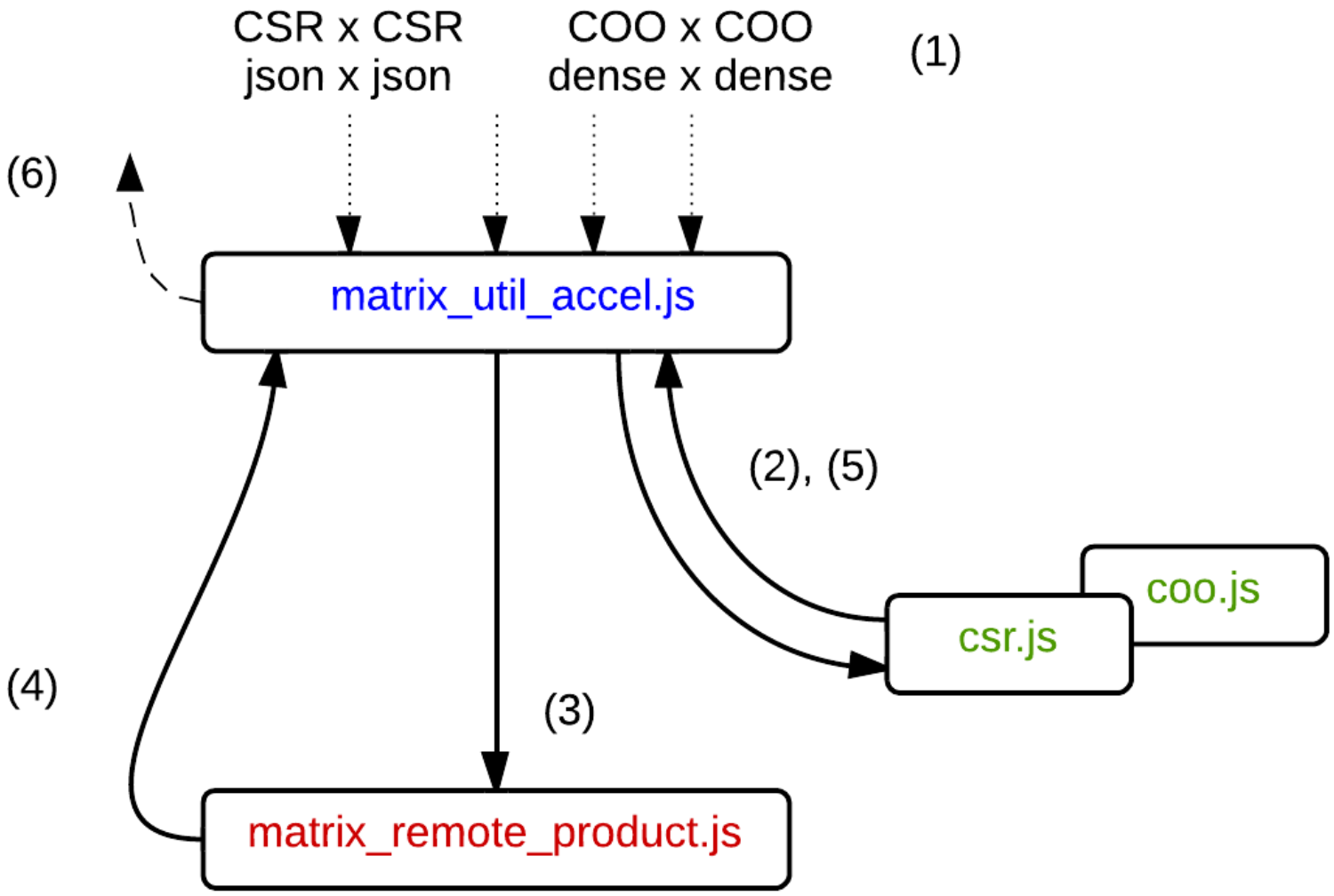
Create a webservice that provides fast matrix operations on the network

Summary

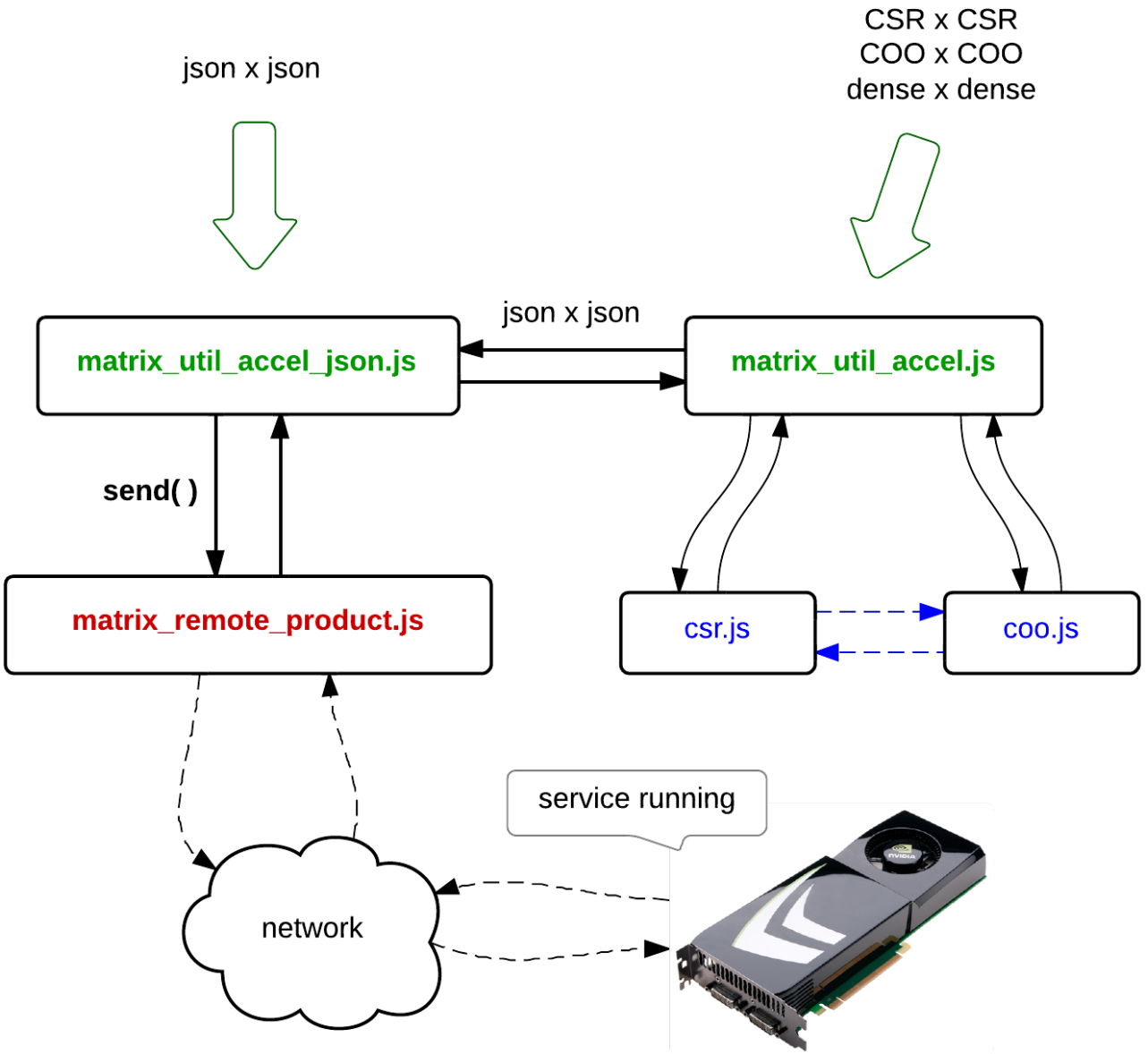
1. ***Localization of code's sections that use matrix operations***
   - Replacement with a call to an external procedure (*interface*)
2. ***Creation of a proper layer that implements the computation of matrix's operation hiding the logic implementation***
   - Definition of a **matrix_util_accel.js** layer (responsable to handle format or encoding, such as csr, coo, json)
   - Realization of a HttpRrequest with the matrix computation webservice using Representational state transfer – REST – in the **matrix_remote_product.js** layer (This layer is also responsable to manage the answer, with an opportune decoding in order to satisfy the spefics)
3. ***Setup a web service that offers such operation online***
   - available with node.js;
   - acquisition of the request using REST;
   - executing the computations (OpenCL) and forwarding the results;
   - create a *domain name* in which will be available such services online

(2)
CLASS DIAGRAM

CLASS DIAGRAM – PREVIOUS VERSION



CLASS DIAGRAM

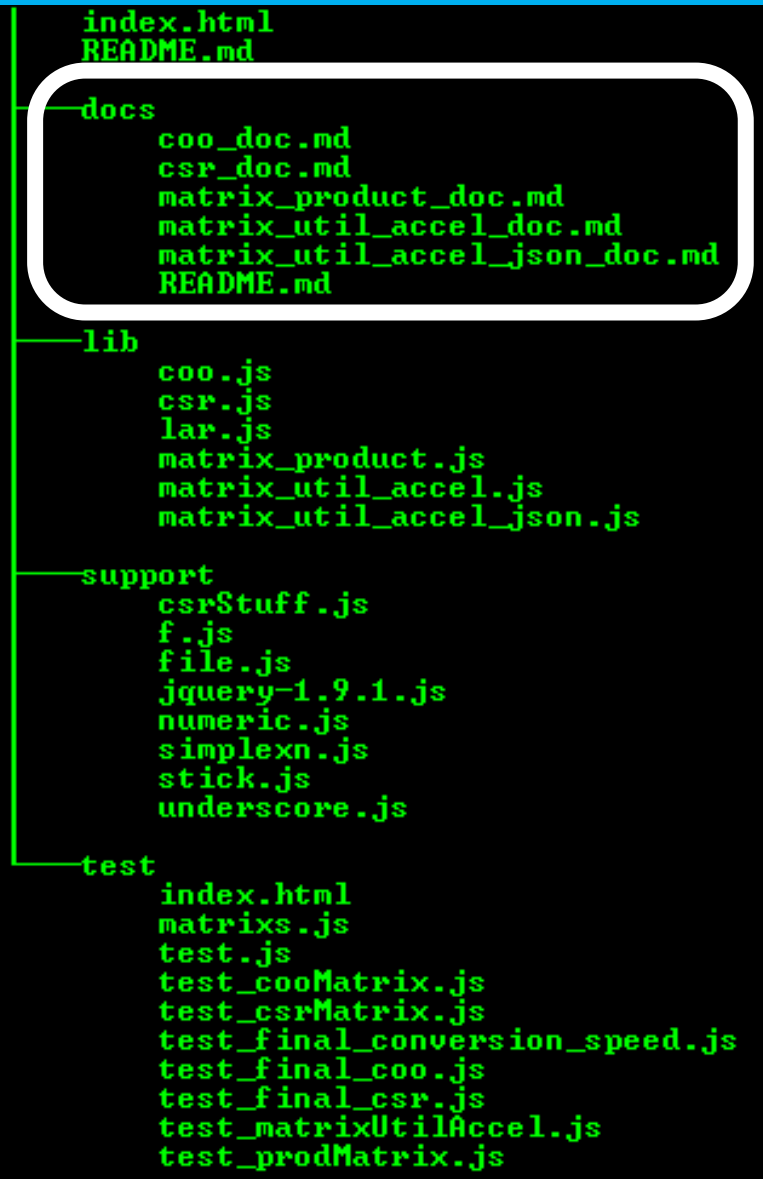


(3)
CODE TREE

CODE TREE

```
index.html
README.md

docs
    coo_doc.md
    csr_doc.md
    matrix_product_doc.md
    matrix_util_accel_doc.md
    matrix_util_accel_json_doc.md
    README.md

lib
    coo.js
    csr.js
    lar.js
    matrix_product.js
    matrix_util_accel.js
    matrix_util_accel_json.js

support
    csrStuff.js
    f.js
    file.js
    jquery-1.9.1.js
    numeric.js
    simplexn.js
    stick.js
    underscore.js

test
    index.html
    matrixs.js
    test.js
    test_cooMatrix.js
    test_csrMatrix.js
    test_final_conversion_speed.js
    test_final_coo.js
    test_final_csr.js
    test_matrixUtilAccel.js
    test_prodMatrix.js
```

CODE TREE



DOCUMENTATION FOLDER

CODE TREE

CLASS FOLDER

CODE TREE

SUPPORT CLASS FOLDER

CODE TREE

```
index.html
README.md

docs
    coo_doc.md
    csr_doc.md
    matrix_product_doc.md
    matrix_util_accel_doc.md
    matrix_util_accel_json_doc.md
    README.md

lib
    coo.js
    csr.js
    lar.js
    matrix_product.js
    matrix_util_accel.js
    matrix_util_accel_json.js

support
    csrStuff.js
    f.js
    file.js
    jquery-1.9.1.js
    numeric.js
    simplexn.js
    stick.js
    underscore.js

test
    index.html
    matrixs.js
    test.js
    test_cooMatrix.js
    test_csrMatrix.js
    test_final_conversion_speed.js
    test_final_coo.js
    test_final_csr.js
    test_matrixUtilAccel.js
    test_prodMatrix.js
```

TEST FOLDER

(4)
EXAMPLE TEST

HOW IT WORKS

Il servizio è disponibile presso l'indirizzo

http://cvd01.dia.uniroma3.it:3000/service/test/multiply

E' possibile avviarlo effettuando il log-in sul server

cd node-lar-multiply-rest
./startREST.sh --webcl

Per testare che il servizio sia in piedi

http://webpdb.dia.uniroma3.it/service/test/networktest

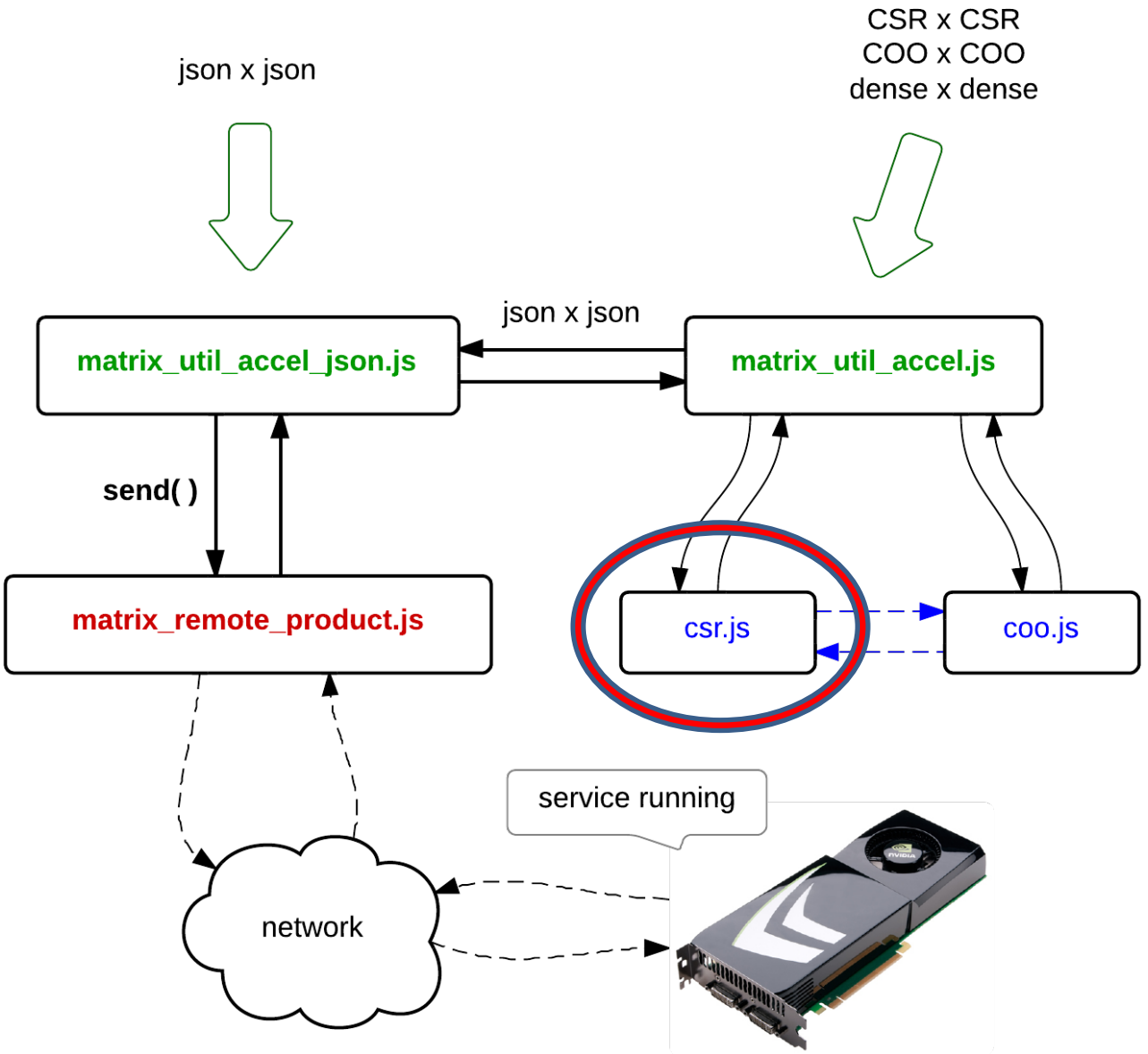
TEST SUMMARY

- Test of **CSR** matrix (**csr.js**)
- Test of **COO** matrix (**coo.js**)
- Test the **product** of matrix (**matrix_remote_product.js**)
- Test of the **conversion speed**
- **Performance test**

TEST SUMMARY

- Test of **CSR** matrix (**csr.js**)
- Test of **COO** matrix (**coo.js**)
- Test the **product** of matrix (**matrix_remote_product.js**)
- Test of the **conversion speed**
- **Performance test**

CLASS DIAGRAM



TEST – CSR MATRIX (csr.js)

Testing how to create a matrix from a dense representation

```
var dense_matrix = [[1, 0, 1],[0, 1, 0],[1, 0, 0]];

var matrix_from_dense =
        new csr_matrix_from_dense(dense_matrix);

log("This is his JSON : " +
        JSON.stringify( matrix_from_dense.toJSON() ));
```

TEST – CSR MATRIX (csr.js)

```
/**
   * Return a csr_matrix reference from a flat representation given in a
         JSON format.
   * @param {Array of Array} objargs An Array of Array containing the
         dense matrix
   * @return {csr_matrix}
   */

function csr_matrix_from_dense(denseMatrix) { .. }
```

```
var matrix_from_dense =
         new csr_matrix_from_dense(dense_matrix);
```

TEST – CSR MATRIX (csr.js)

```
/**
   * Print the JSON of the current CSR matrix
   * @return {JSON}
   */

csr_matrix.prototype.toJSON = function() { .. }
```

```
log("This is his JSON : " +
        JSON.stringify( matrix_from_dense.toJSON() ));
```

RUN TEST

TEST – CSR MATRIX (csr.js)

Testing how to create a matrix from JSON

```
var csr_json = { "ROW" : [0,2,3,4], "COL" : [0,2,1,0], "DATA" :
[1,1,1,1], "ROWCOUNT" : 3, "COLCOUNT" : 3 };

var matrixFromJson = new csr_matrix_from_json(csr_json);

print(matrixFromJson.toString());

print(matrixFromJson.toDense());
```

TEST – CSR MATRIX (csr.js)

```
/**
   * Print a string representazion of the current csr.
   * @return {String}
   */

csr_matrix.prototype.toString = function() { .. }
```

print(matrixFromJson.toString());

```
/**
   * Returns a dense representation of the matrix.
   * @return {Array{Array}}
   */

csr_matrix.prototype.toDense = function() { .. }
```

print(matrixFromJson.toDense());

TEST – CSR MATRIX (csr.js)

```
/**
   * Create an istance of csr_matrix using a spefic json format which is
         the following.
   * @param {JSON} objargs, Here there is the valid format { "ROW" : array,
         "COL" : array, "DATA" : array, "ROWCOUNT" : value,
         "COLCOUNT" : value }
   * @return {csr_matrix}
   */

function csr_matrix_from_json(objargs){ .. }
```

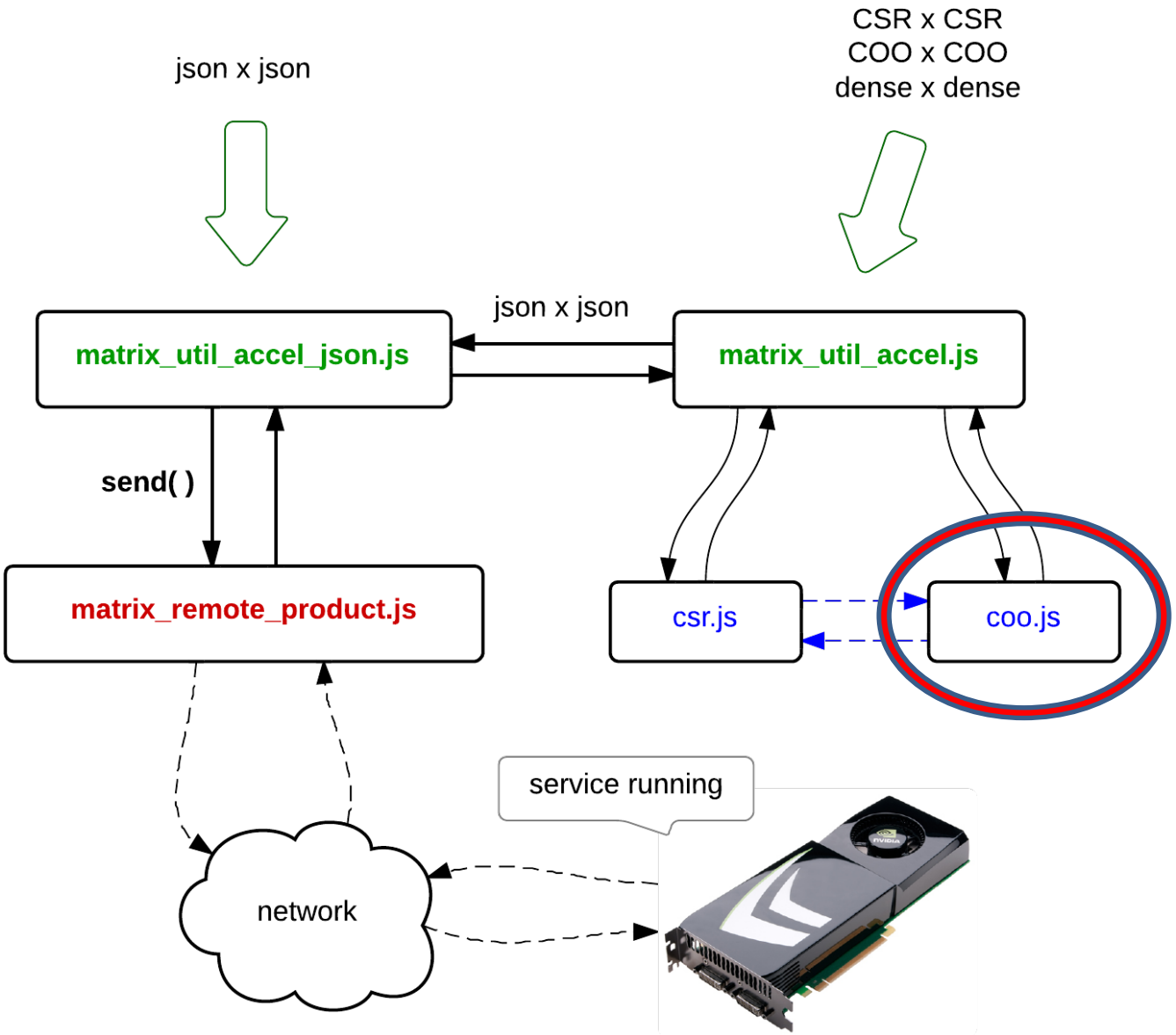
var matrixFromJson = new **csr_matrix_from_json(csr_json);**

RUN TEST

TEST SUMMARY

- Test of **CSR** matrix (**csr.js**)
- Test of **COO** matrix (**coo.js**)
- Test the **product** of matrix (**matrix_remote_product.js**)
- Test of the **conversion speed**
- **Performance test**

CLASS DIAGRAM



TEST – COO MATRIX (coo.js)

Testing how to create a matrix from a dense representation.

```
var dense_matrix = [[1, 0, 1, 0], [0, 2, 0, 1], [1, 3, 6, 0]];

var matrix_from_dense =
        new coo_matrix_from_dense(dense_matrix);

log("This is his JSON : " +
        JSON.stringify(matrix_from_dense.toJSON()));
```

TEST – COO MATRIX (coo.js)

```
/**
   * Returns a coo_matrix reference from a flat representation given in a
         JSON format.
   * @param {Array of Array} objargs An Array of Array containing the
         dense matrix
   * @return {coo_matrix}
   */

function coo_matrix_from_dense(denseMatrix) { ..  }
```

```
var matrix_from_dense =
         new coo_matrix_from_dense(dense_matrix);
```

RUN TEST

TEST – COO MATRIX (coo.js)

Testing how to create a matrix from json.

```
var coo_json = { "rowcount": 3,
                 "colcount": 4,
                 "row": [0,0,1,1,2,2,2],
                 "col": [0,2,1,3,0,1,2],
                 "val": [1,1,2,1,1,3,6]};

var matrixFromJson = new coo_matrix_from_json(coo_json);
```

TEST – COO MATRIX (coo.js)

```
/**
   * Create an istance of coo_matrix using a spefic json format which is
         the following.
   * @param {JSON} objargs, Here is the valid format { "row" : array, "col" : array,
         "val" : array, "rowcount" : value, "colcount" : value }
   * @return {coo_matrix}
   */

function coo_matrix_from_json(objargs){ .. }
```

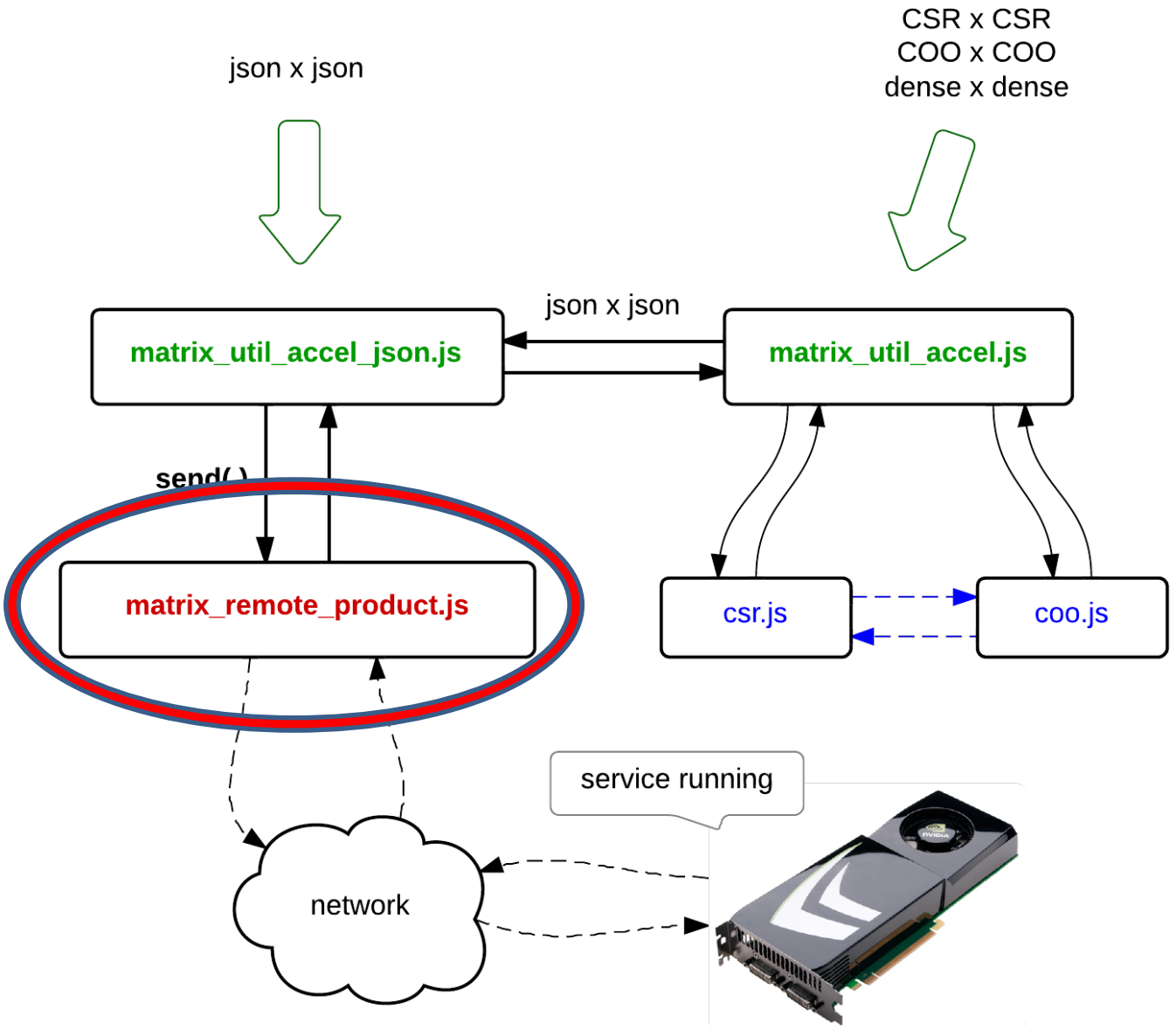
var matrixFromJson = new **coo_matrix_from_json(coo_json);**

RUN TEST

TEST SUMMARY

- Test of **CSR** matrix (**csr.js**)
- Test of **COO** matrix (**coo.js**)
- Test the **product** of matrix (**matrix_remote_product.js**)
- Test of the **conversion speed**
- **Performance test**

TEST SUMMARY



TEST – MATRIX PRODUCT (matrix_remote_product.js)

Testing prodMatrixAsync_log()

var Ajson = { "ROWCOUNT" : 3, "COLCOUNT" : 3, "ROW" : [0,1,3,5], "COL" : [2,0,2,0,1], "DATA" : [1,1,1,1,1] };
var Bjson = { "ROWCOUNT" : 3, "COLCOUNT" : 3, "ROW" : [0,1,3,5], "COL" : [2,0,2,0,1], "DATA" : [1,1,1,1,1] };

matrix_remote_product.prodMatrixAsync_log(Ajson,Bjson);

TEST – MATRIX PRODUCT (matrix_remote_product.js)

```
/**
   * Send an async request for a matrices product and print a log in the console.
   * @param {JSON} matrixA
   * @param {JSON} matrixB
   */

mrp.prodMatrixAsync_log = function (matrixA,matrixB) { .. }
```

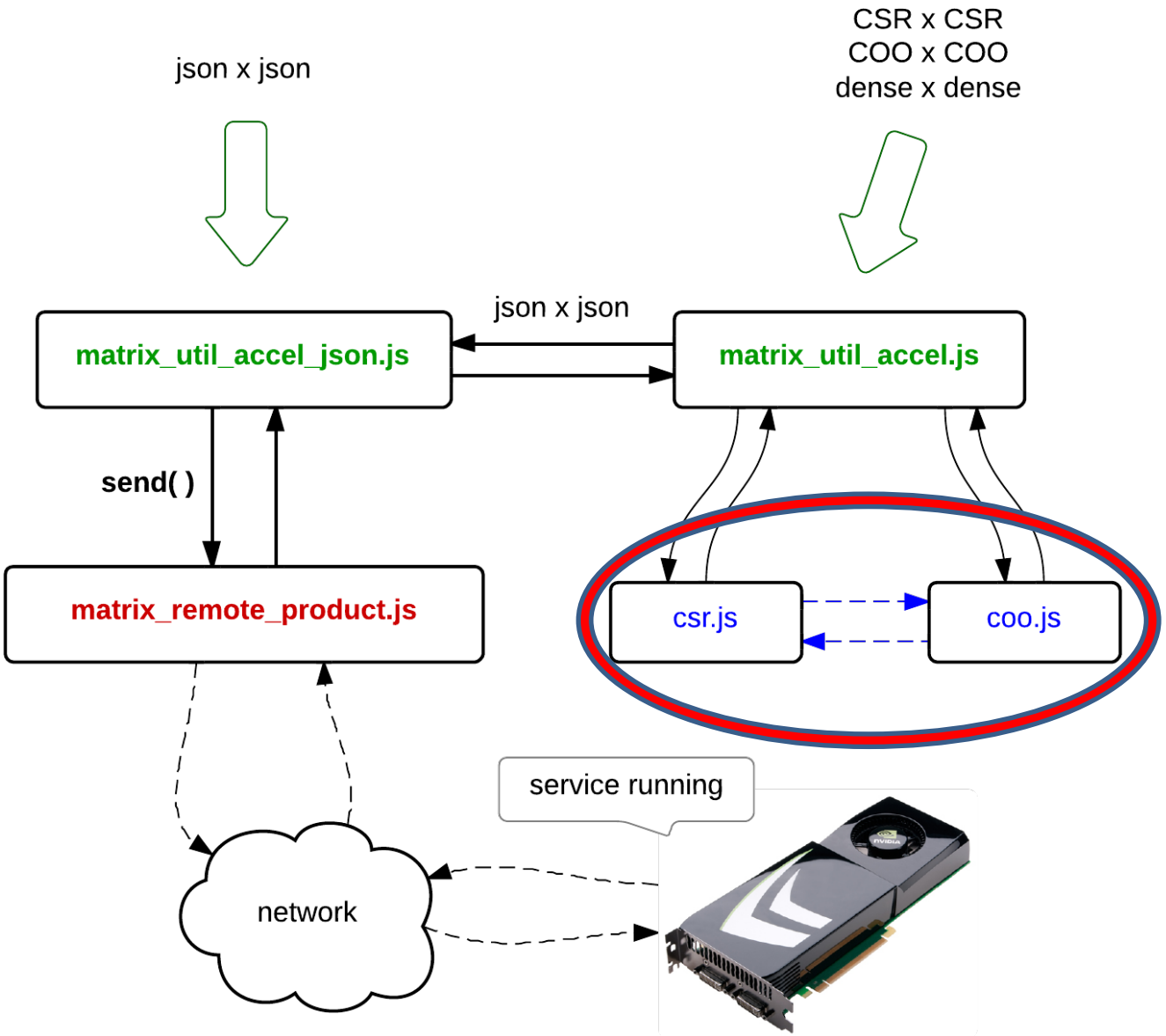
matrix_remote_product.prodMatrixAsync_log(Ajson,Bjson);

RUN TEST

TEST SUMMARY

- Test of **CSR** matrix (**csr.js**)
- Test of **COO** matrix (**coo.js**)
- Test the **product** of matrix (**matrix_remote_product.js**)
- Test of the **conversion speed**
- **Performance test**

CLASS DIAGRAM



TEST – CONVERSION SPEED

From dense to csr

var matrixA_csr_1000 =
new **csr_matrix_from_dense(matrix_a_1000);**

From csr to json

var matrixA_csr_json_1000 = **matrixA_csr_1000.toJSON();**

From csr to dense

var matrixA_dense_1000 = **matrixA_csr_1000.toDense();**

TEST – CONVERSION SPEED

From dense to coo

```
var matrixA_coo_1000 =
        new coo_matrix_from_dense(matrix_a_1000);
```

From coo to json

```
var matrixA_coo_json_1000 = matrixA_coo_1000.toJSON();
```

From coo to dense

```
var matrixA_dense_1000 = matrixA_coo_1000.toDense();
```

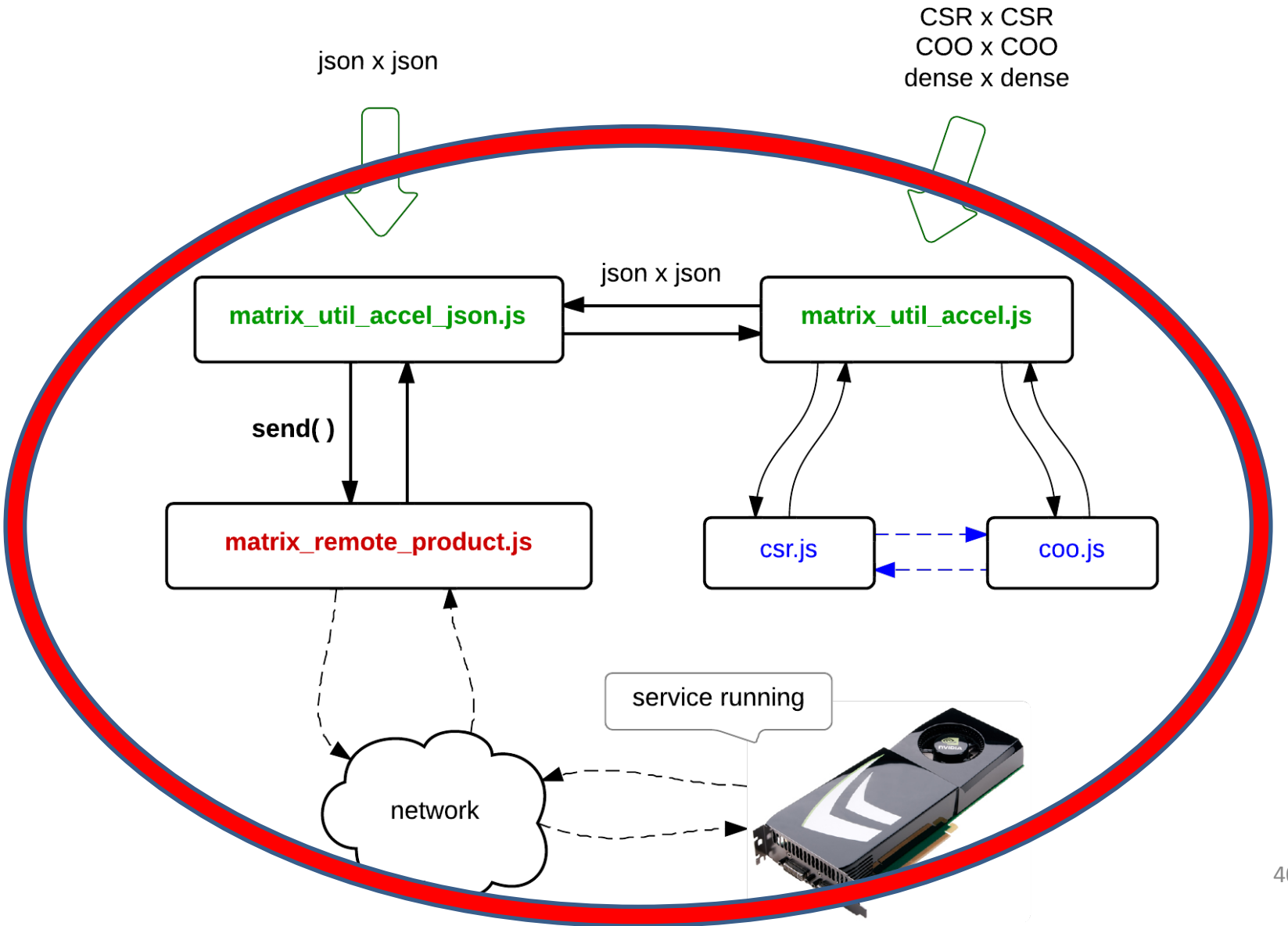
TEST – CONVERSION SPEED

RUN TEST

TEST SUMMARY

- Test of **CSR** matrix (**csr.js**)
- Test of **COO** matrix (**coo.js**)
- Test the **product** of matrix (**x.js**)
- Test of the **conversion speed**
- **Performance test**

CLASS DIAGRAM



TEST – PERFORMANCE TEST

100x100 Matrix Product

```
var matrixAB_csr_json_100 =
        matrix_util_accel_json.csr_json_product(
                matrixA_csr_json_100,
                matrixB_csr_json_100);

var matrixAB_coo_json_100 =
        matrix_util_accel_json.coo_json_product(
                matrixA_coo_json_100,
                matrixB_coo_json_100);
```

RUN TEST

TEST – PERFORMANCE TEST

1000x1000 Matrix Product

```
var matrixAB_csr_json_1000 =
        matrix_util_accel_json.csr_json_product(
                matrixA_csr_json_1000,
                matrixB_csr_json_1000);

var matrixAB_coo_json_1000 =
        matrix_util_accel_json.coo_json_product(
                matrixA_coo_json_1000,
                matrixB_coo_json_1000);
```

RUN TEST

(5)
REFERENCES

REFERENCES

Project main folder:

https://github.com/cvdlab-bio/weblar/tree/master/projects/Matrix%20computation%20web-service

Documentation:

https://github.com/cvdlab-bio/weblar/tree/master/projects/Matrix%20computation%20web-service/docs