JavaScript Promise Practical Example

2014. 10. 08 @ <divtag> developer meetup

Hiun Kim (openhiun@divtag.sejong.edu)

node JS™

http://jsconf.eu/2009/assets_c/RDahl_sw.jpg evented I/O for v8 javascript

building fast, scalable network applications

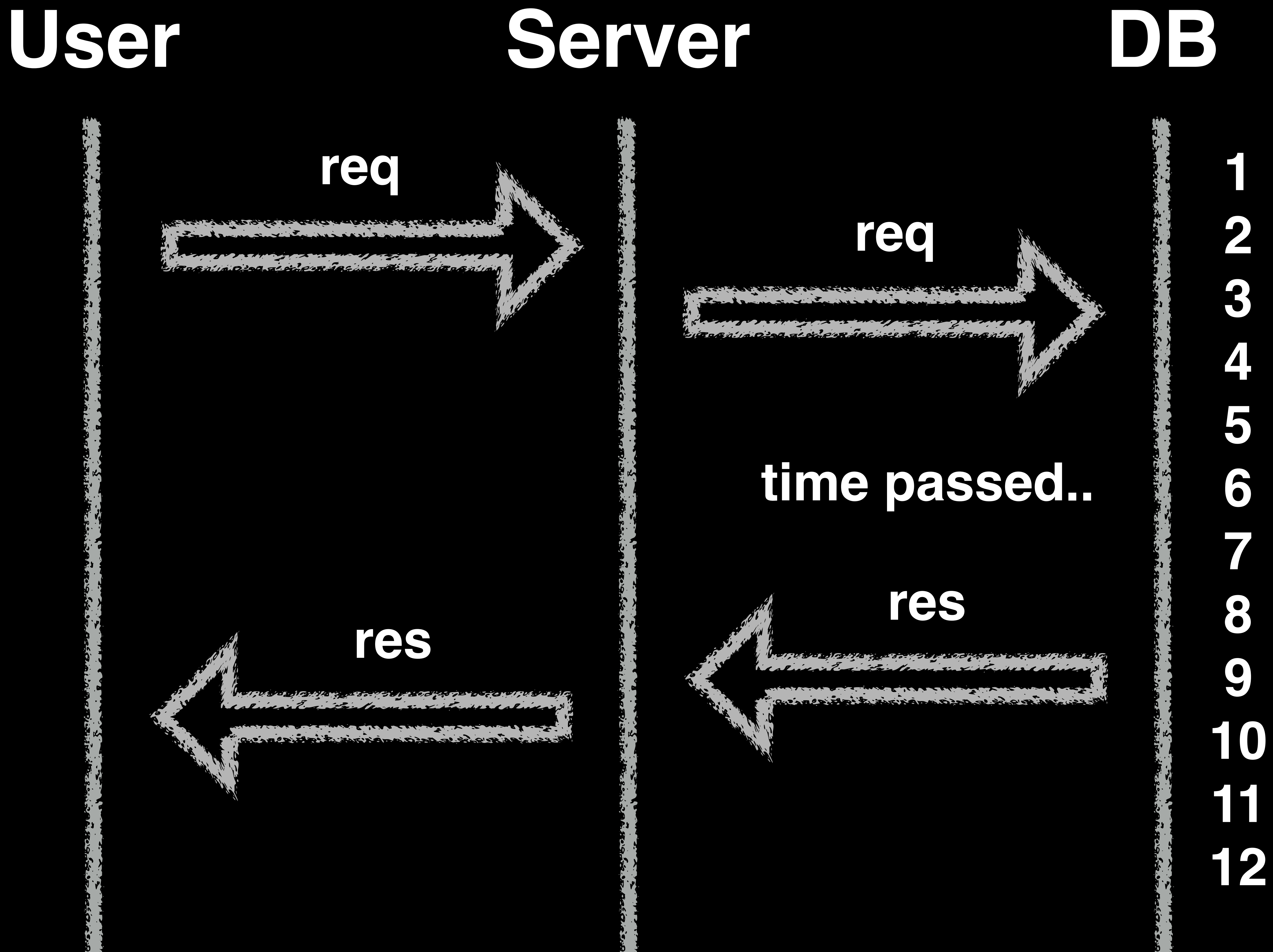
event-driven
non-blocking I/O

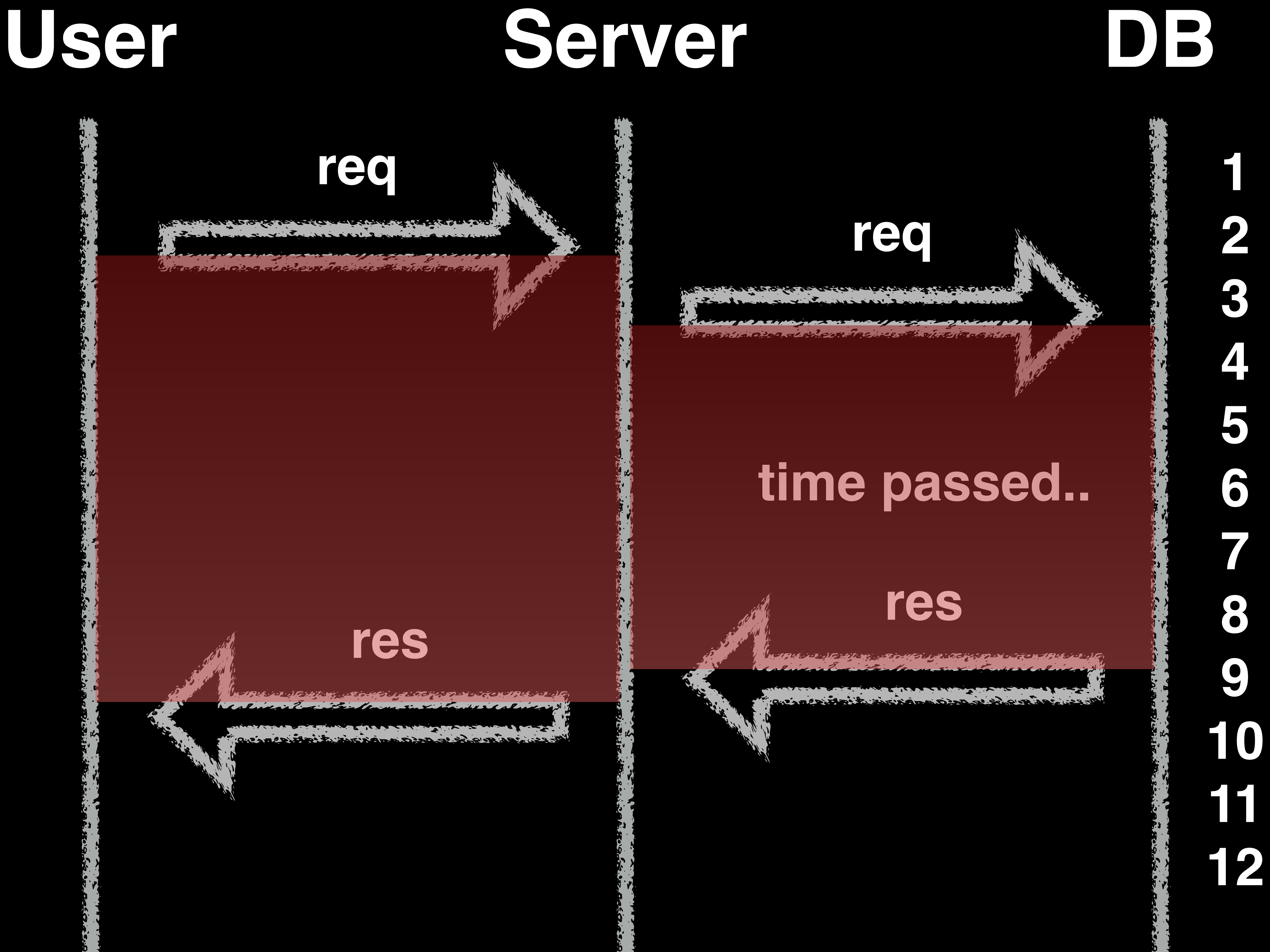
Usually..
blocking I/O

RAM - 1
DISK -100

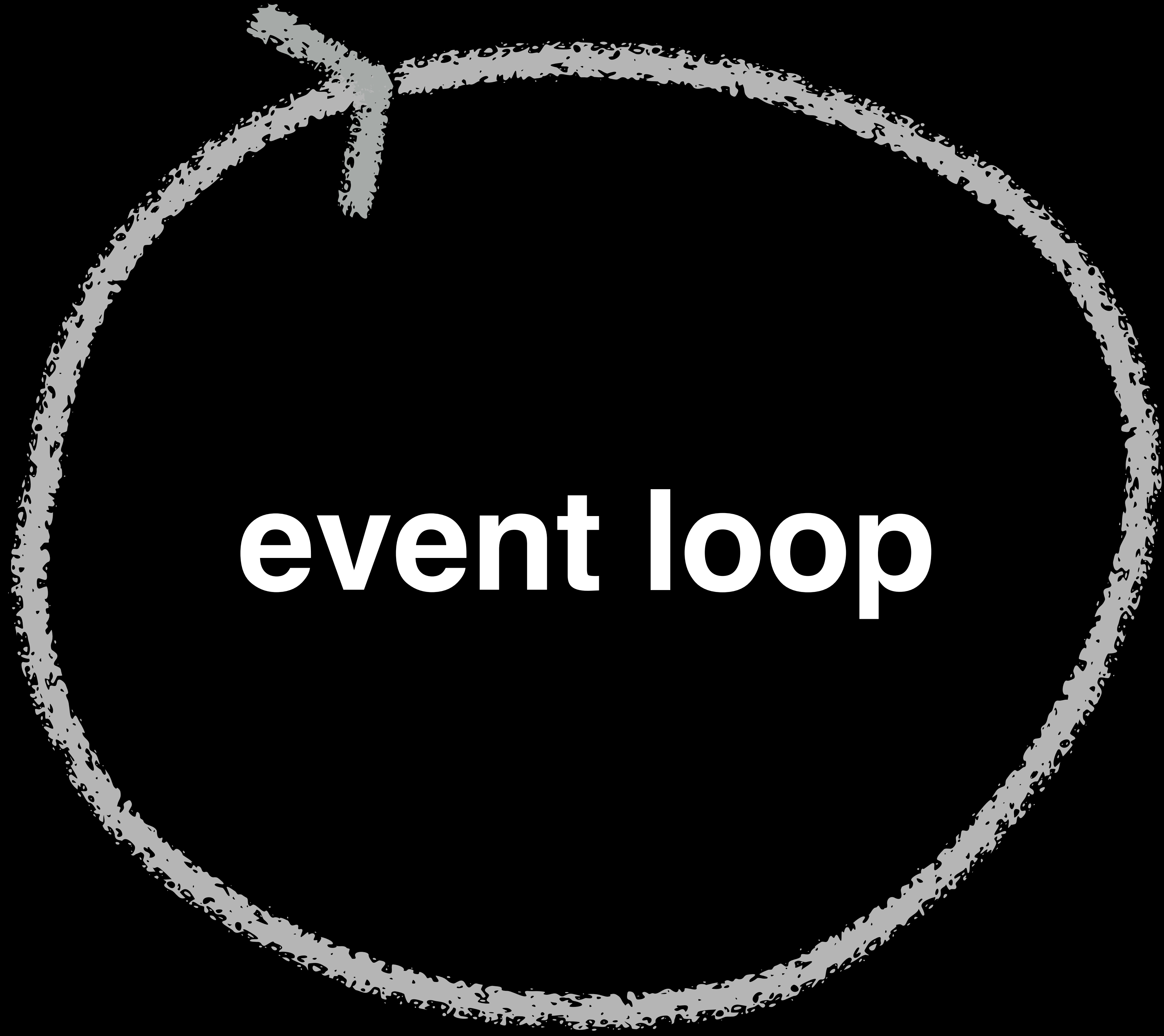
RAM - 1
DISK -100
Network - 5000

```
var result = db.query('SELECT * FROM Users');
```


User
Server
DB
req
req
time passed..
res
res
1
2
3
4
5
6
7
8
9
10
11
12

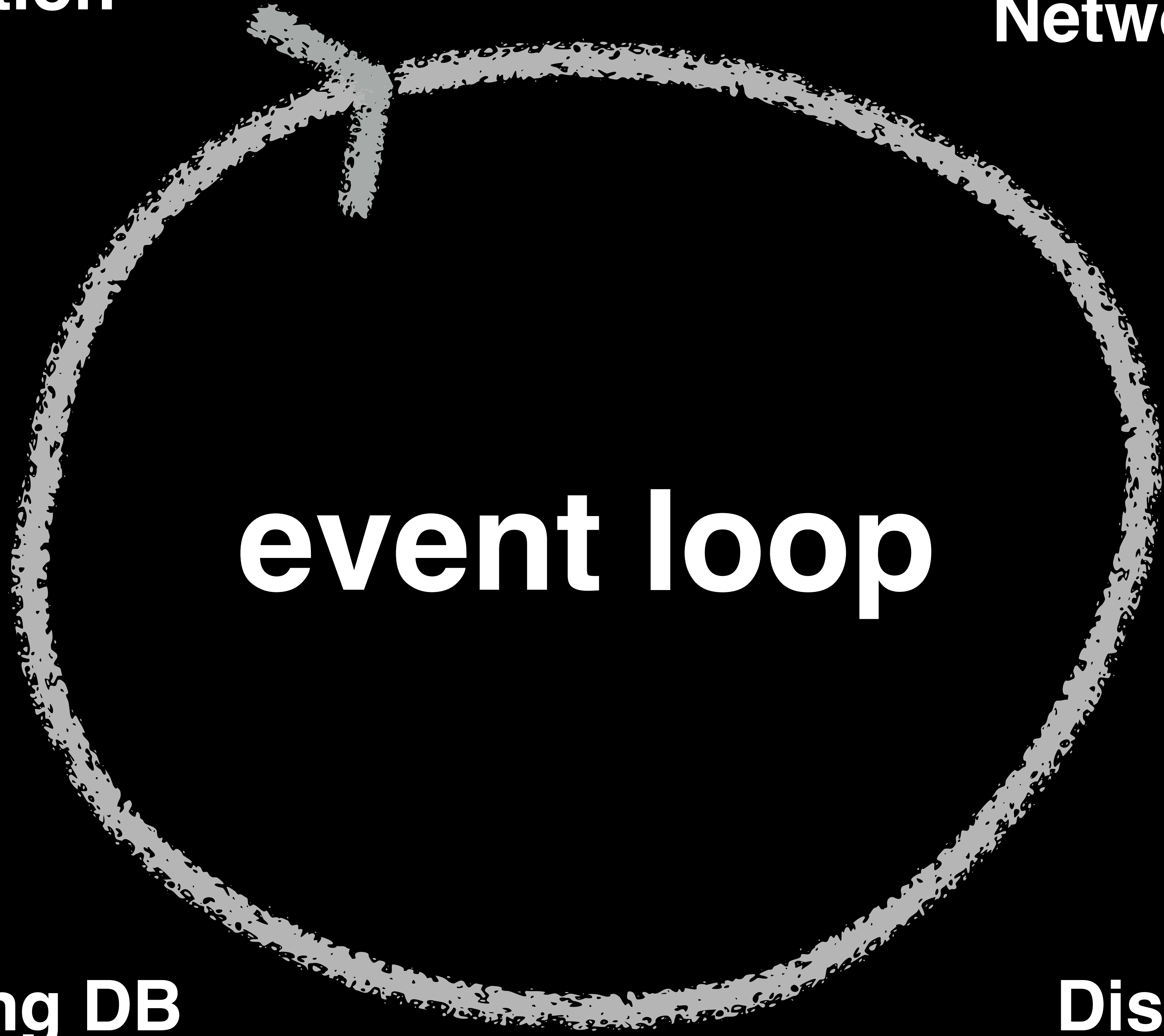

User
Server
DB
req
req
time passed..
res
res
1
2
3
4
5
6
7
8
9
10
11
12

http://www.ohmynews.com/NWS_Web/View/at_pg.aspx?CNTN_CD=A0001748826


event loop

Calculation

Network I/O

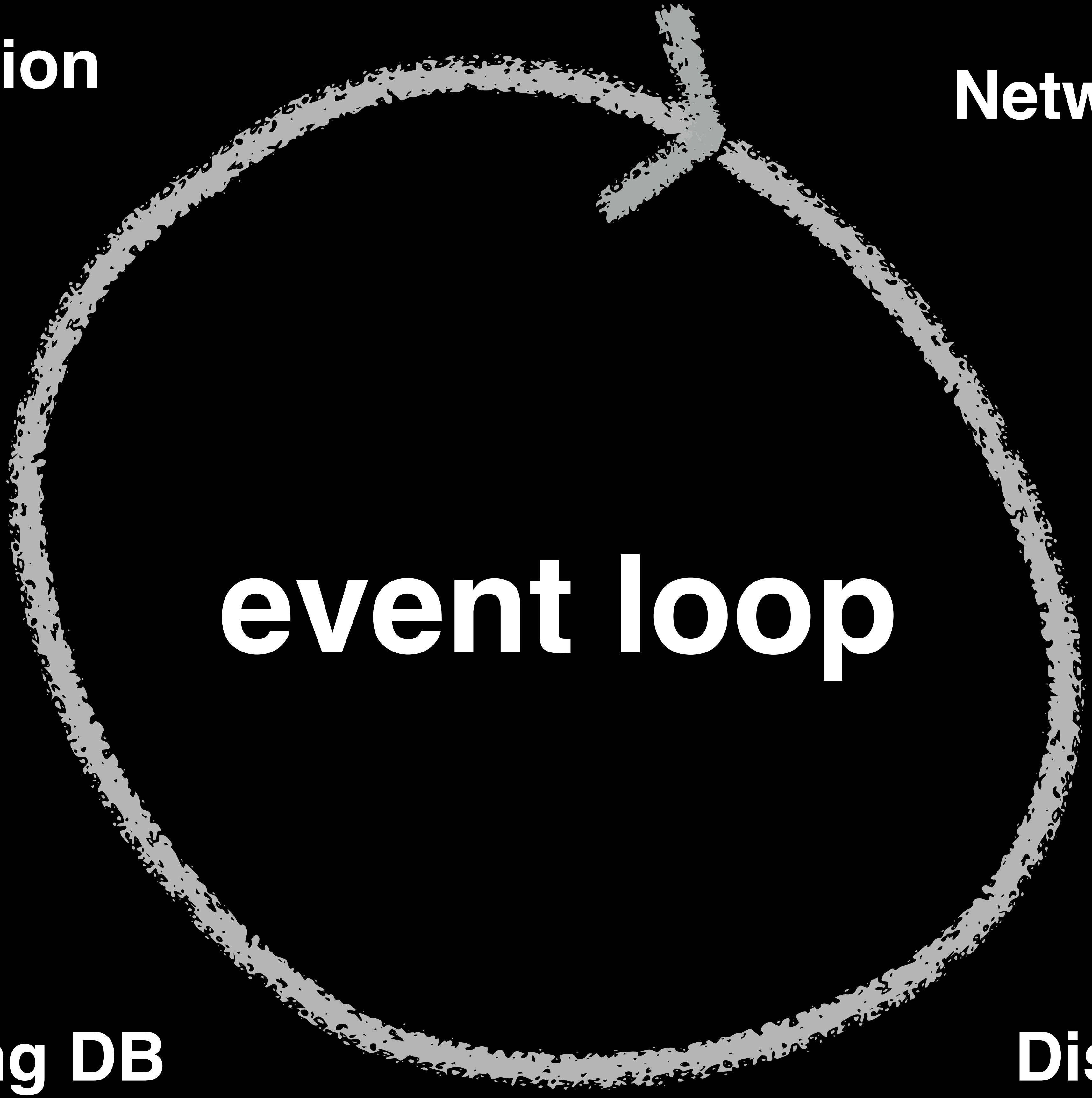


Querying DB

Disk I/O

Calculation

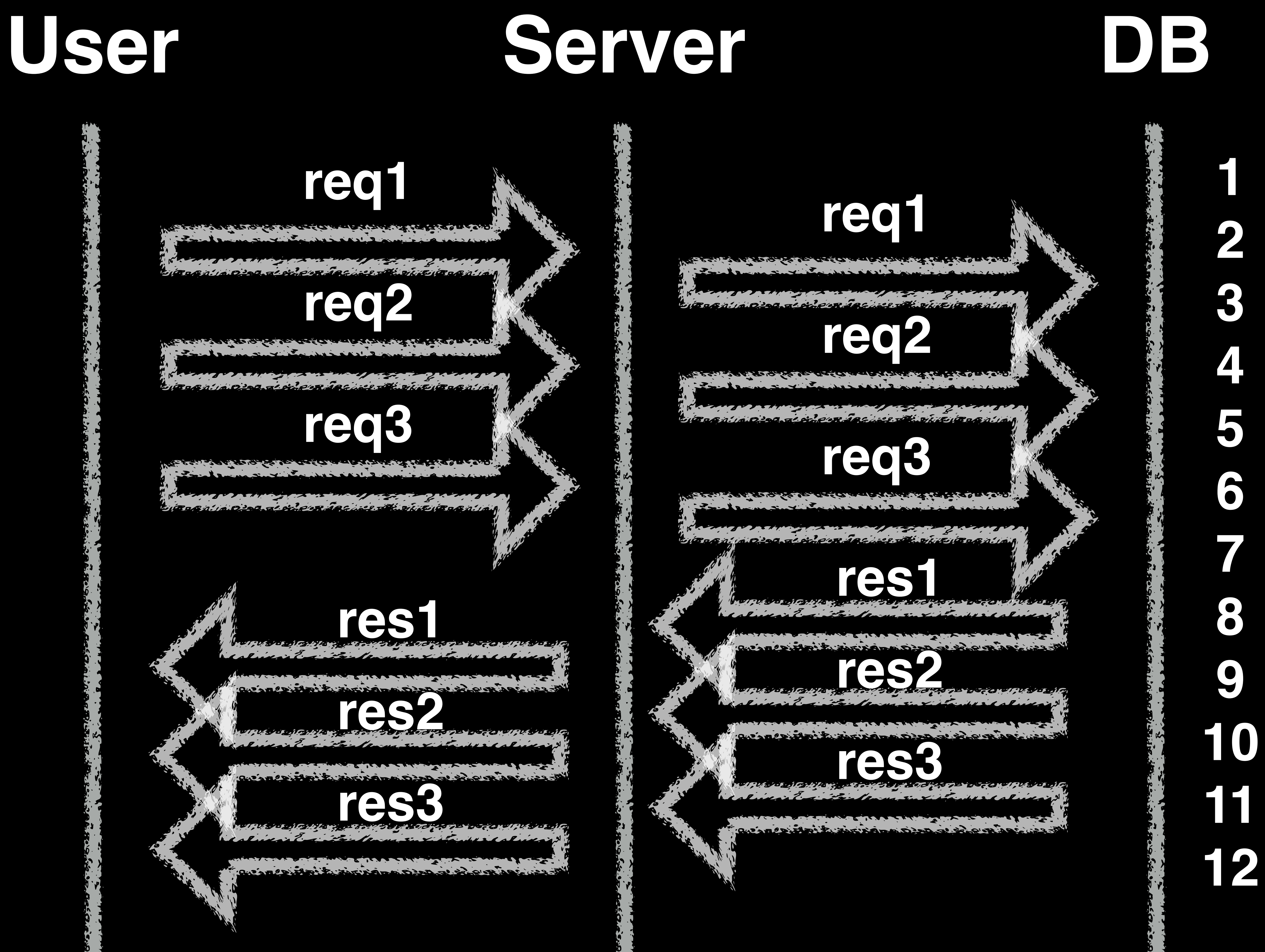
Network I/O

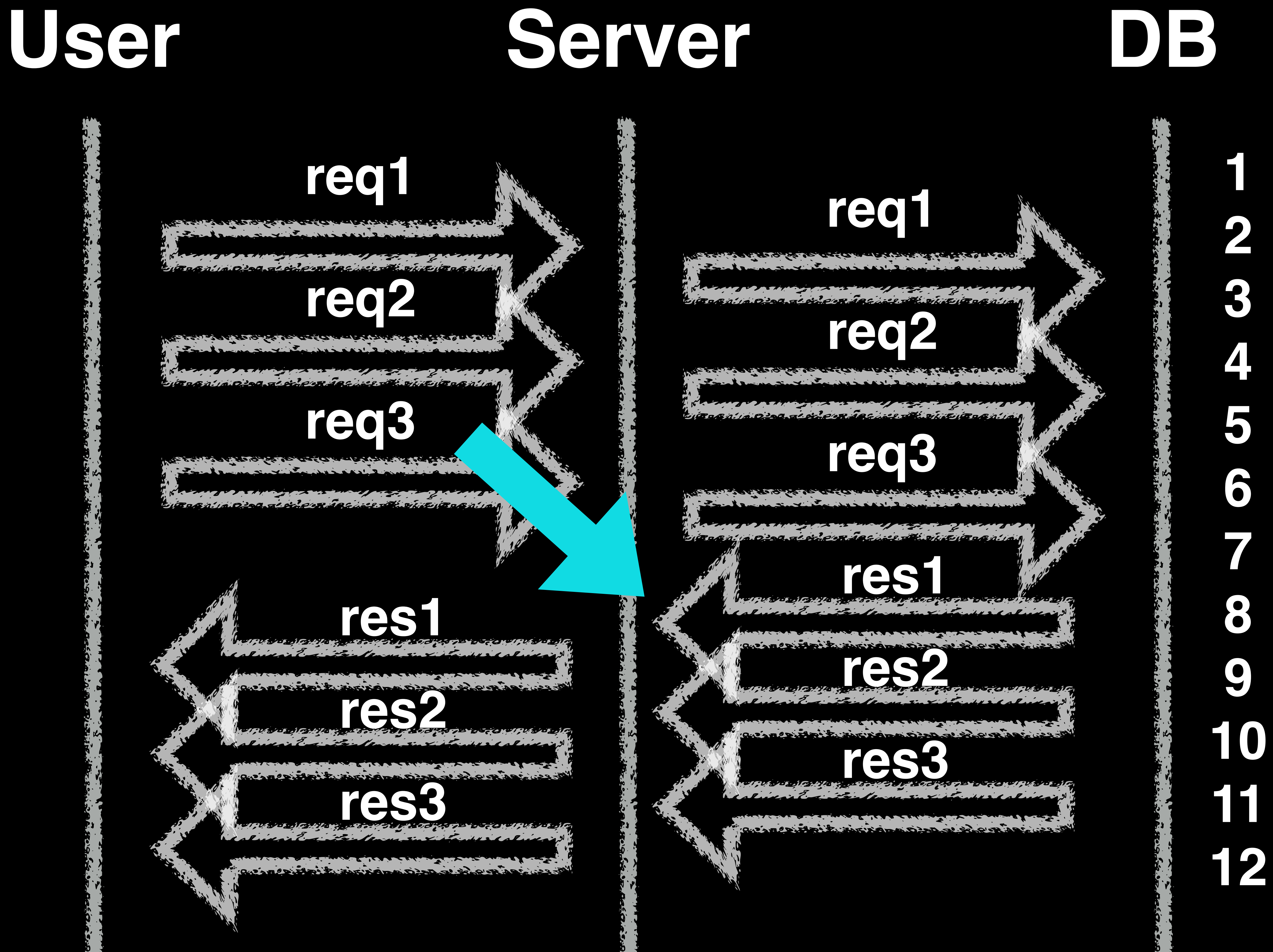


event loop

Querying DB

Disk I/O


User
Server
DB
req1
req2
req3
res1
res2
res3
req1
req2
req3
res1
res2
res3
1
2
3
4
5
6
7
8
9
10
11
12


User
Server
DB
req1
req2
req3
res1
res2
res3
req1
req2
req3
res1
res2
res3
1
2
3
4
5
6
7
8
9
10
11
12

Callback

```
var result = db.query('SELECT * FROM Users');

result.addListener('err', function (err) {
  //use err as a local variable
});

result.addListener('result', function (result) {
  //use result as a local variable
});
```

```
db.query('SELECT * FROM Users', function (err, result) {
  //use result as a local variable
});
```

Continous Passing Style

```
var fs = require('fs');
var data = 'a.txt';

fs.readFile(data, 'utf8', function (data1) {
  fs.readFile(data1, 'utf8', function (data2) {
    fs.readFile(data2, 'utf8', function (data3) {
      fs.readFile(data3, 'utf8', function (data4) {
        fs.readFile(data4, 'utf8', function (data5) {

          //do stuff with data5

        });
      });
    });
  });
});
```

Real World Scenario

Sign Up

sign up requires..

- receive data
- validate data*
- manipulate data
- insert data*

```
var email = req.body.email;
var password = req.body.email;

if (blanky(email, password) === true ||
   validator.isEmail(email) === false ||
   password.length < 8) {
   res.send(400).end();
} else {

   db.query('SELECT Email FROM Users WHERE ?',
      email, function (err, result) {
      if (err) {
         res.send(500).end();
      } else if (result === 0) {
         res.send(409).end();
      } else {

         db.query('INSERT INTO Users VALUES (?, ?)',
            [email, password], function (err, result) {
            if (err) {
               res.send(500).end();
            } else {
               req.session.email = email;
               res.send(200).end();
            }
         });
      }
   });
}
```

receive data

validate data

insert data

manipulate & insert data

```
var email = req.body.email;
var password = req.body.email;

if (blanky(email, password) === true ||
   validator.isEmail(email) === false ||
   password.length < 8) {
   res.send(400).end();
} else {

   db.query('SELECT Email FROM Users WHERE ?',
      email, function (err, result) {
      if (err) {
         res.send(500).end();
      } else if (result === 0) {
         res.send(409).end();
      } else {

         db.query('INSERT INTO Users VALUES (?, ?)',
            [email, password], function (err, result) {
            if (err) {
               res.send(500).end();
            } else {
               req.session.email = email;
               res.send(200).end();
            }
         });
      }
   });
}
```

receive data

validate data

insert data

manipulate & insert data

```
var email = req.body.email;
var password = req.body.email;

if (blanky(email, password) === true ||
   validator.isEmail(email) === false ||
   password.length < 8) {
   res.send(400).end();
} else {

   db.query('SELECT Email FROM Users WHERE ?', email)
      .then(function (result) {
         if (result === 0) {
            res.send(409).end();
         } else {
            return db.query('INSERT INTO Users VALUES (?, ?)', [email,
algo(password)]);
         }
      })
      .then(function () {
         req.session.email = email;
         res.send(200).end();
      })
      .catch(function () {
         res.send(500).end();
      })
      .done();
}
```

receive data

validate data

insert data

manipulate & insert data

promise provides..

- managing code complexity horizontally
- maintainable code
- error handling at once

How promise works??

promise is..

- returns a promise, not a callback
- almost every module in npm returns a callback ;-(
- you have to *promisify* or *denodeify* module to use in promise chain

Applied Examples

```javascript
var Promise = require('bluebird');

var fs = Promise.promisifyAll(require('fs'));

fs.readFileAsync('sejong.json')
  .then(JSON.parse)
  .then(console.log)
  .done();
```

```
//sejong.json
{
  "location": "seoul"
}
```

```
var Promise = require('bluebird');

var fs = Promise.promisifyAll(require('fs'));

fs.readFileAsync('sejong.json')
  .then(JSON.parse)
  .then(console.log)
  .done();
```

```
//sejong.json
{
  "location": "seoul"
}
```

```
//will print

{ location: 'seoul' }
```

```
var Promise = require('bluebird');

var fs = Promise.promisifyAll(require('fs'));

fs.readFileAsync('sejong.json')
  .then(JSON.parse)
  .then(function (data) {
    console.log(data.location);
  })
  .done();
```

```
//sejong.json
{
  "location": "seoul"
}
```

```
var Promise = require('bluebird');

var fs = Promise.promisifyAll(require('fs'));

fs.readFileAsync('sejong.json')
  .then(JSON.parse)
  .then(function (data) {
    console.log(data.location);
  })
  .done();
```

```
//sejong.json
{
  "location": "seoul"
}
```

```
//will print

seoul
```

```
var Promise = require('bluebird');

var fs = Promise.promisifyAll(require('fs'));

fs.readFileAsync('sejong.json')
   .then(JSON.parse)
   .then(function (data) {
      var location = data.location;
      return location;
   })
   .then(function (location) {
      return db.query('SELECT * FROM University WHERE ?', location);
   })
   .then(function (list) {
      return [makeHttpRequest(list), location];
   })
   .spread(function (result, location) {
      console.log('request ' + result + 'university' + 'which in' + location);
   })
   .catch(function () {
      console.log('Unknown error has occured!');
   })
   .catch(SyntaxError, function () {
      console.log('JSON syntax is not valid!');
   })
   .finally(function () {
      console.log(‘Program은 성공하거나 실패하거나.. 둘중 하나야');
   })
   .done();
```

```javascript
//module.js
module.exports = function (school) {
  return new Promise(function (reject, resolve) {
    db.query('SELECT Name FROM Users WHERE School = ?',
      school, function (err, result) {
      if (err) {
        reject(err);
      } else {
        resolve(result);
      }
    });
  });
};

//app.js
var module = require('module.js');

module('Sejong University')
  .then(console.log)
  .catch(console.log)
  .done();
```

ES6ROCKS

ECMAScript 6
June 2015

ES6ROCKS

QnA