Скалируемост

Скалируемост - съдържание (1 / 3)

- синхронен модел: клиент – сървър
- отчитане и проблеми
- разпределяне на товара

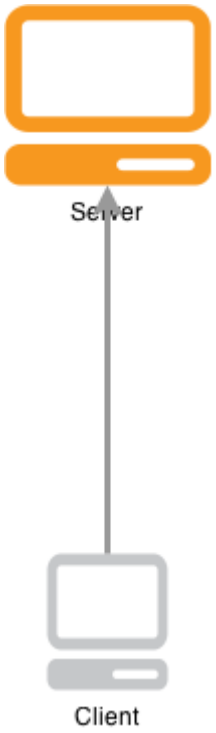
Скалируемост - съдържание (2 / 3)

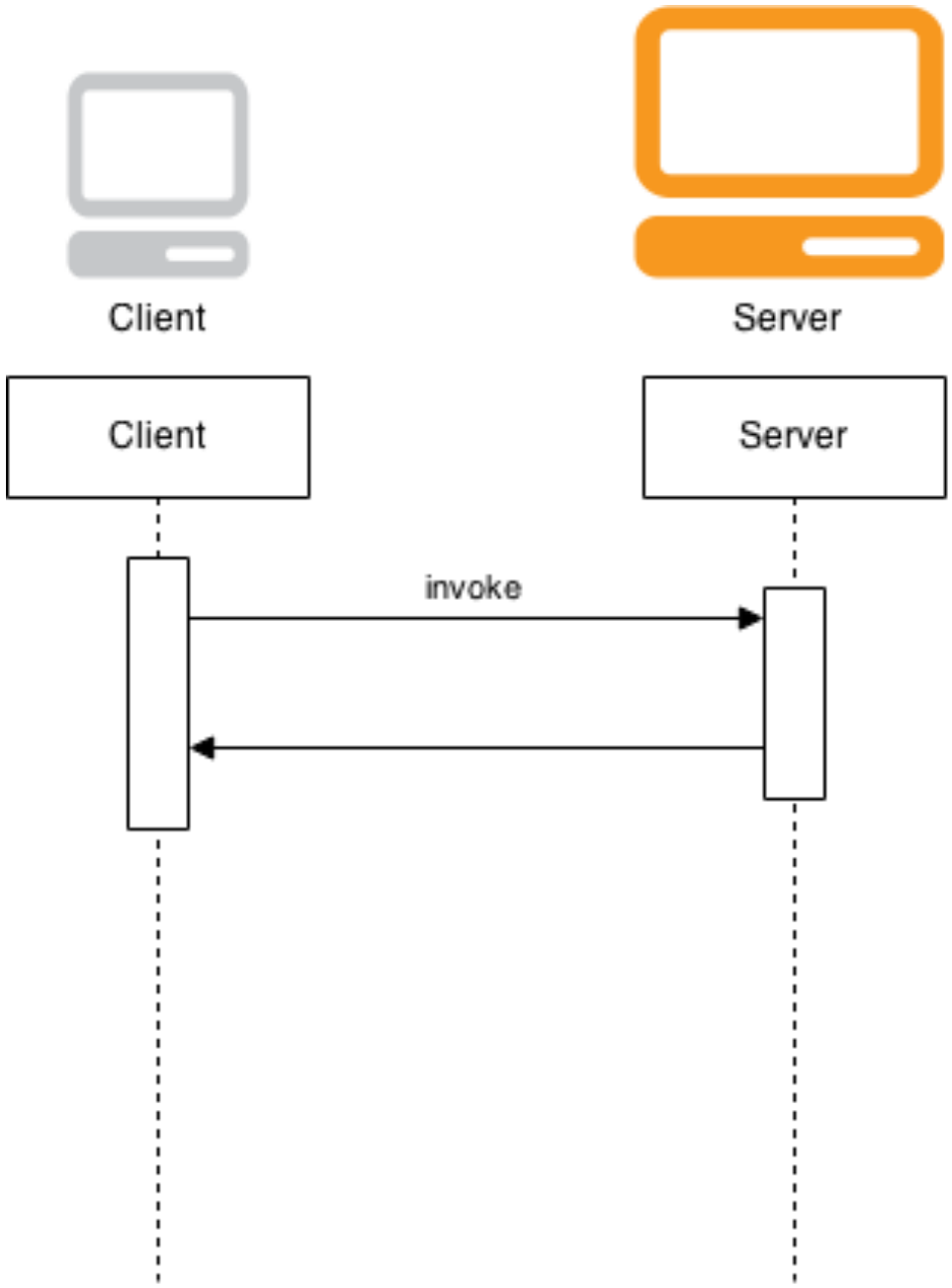
- асинхронно изпълнение
- задачи като абстракция
- примерни решения

Скалируемост - съдържание (3 / 3)

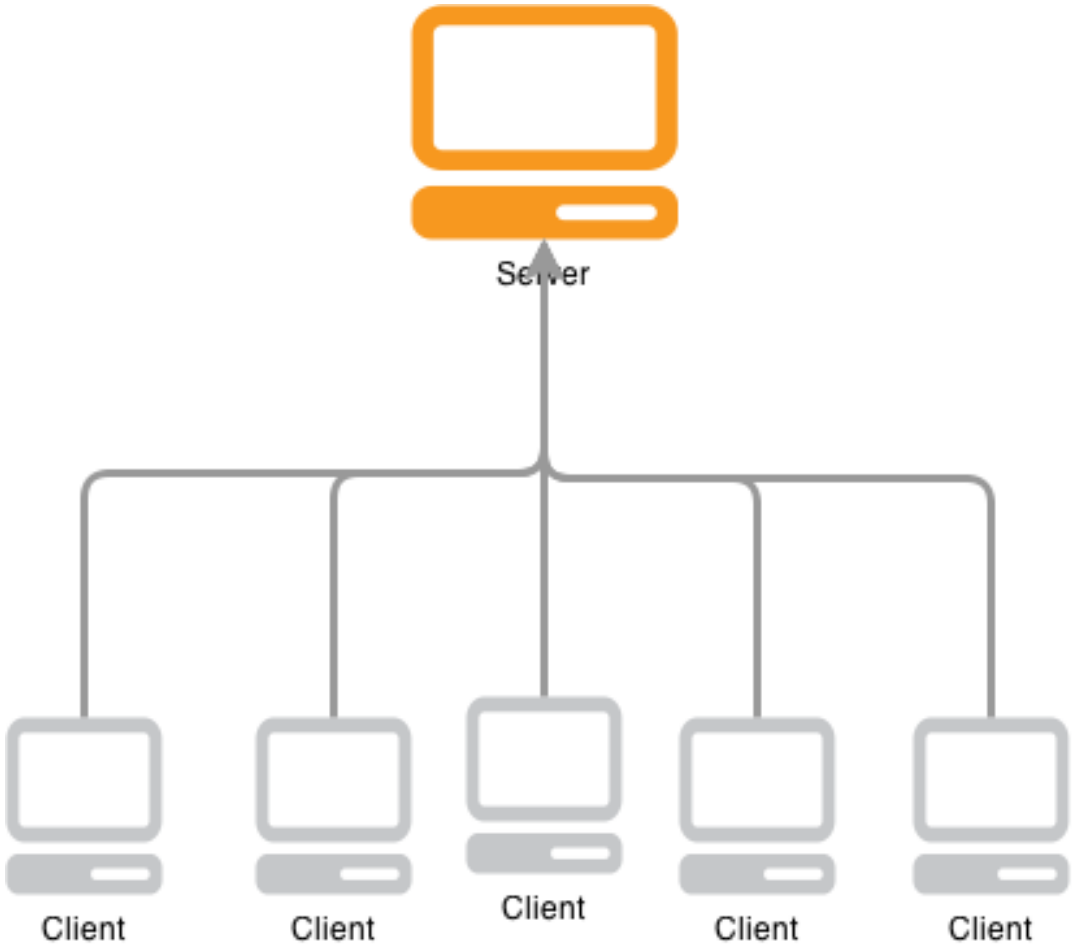
- големи данни
- MapReduce с Hadoop

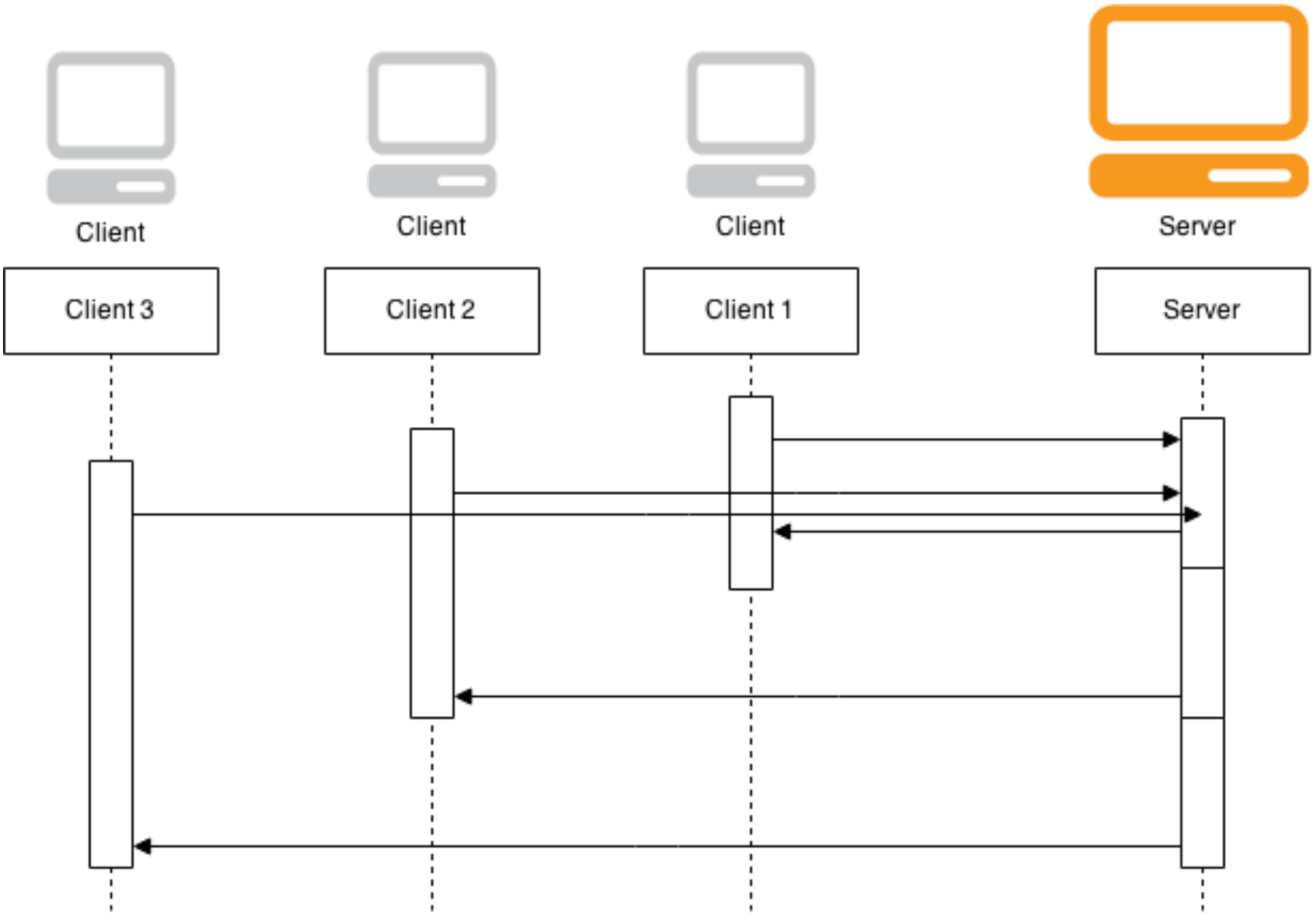
Класически модел




Client
Server
Client
Server
invoke

Класически модел




Client
Client
Client
Server
Client 3
Client 2
Client 1
Server

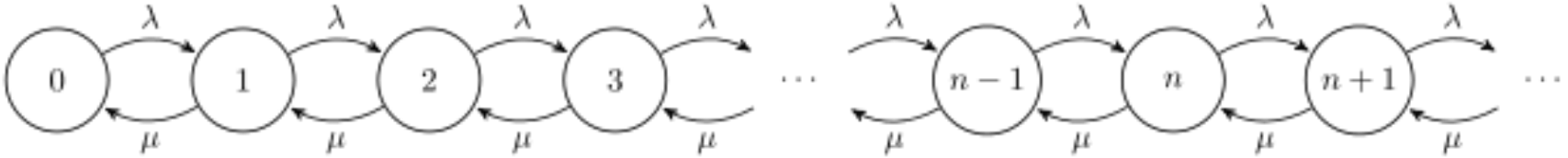
Метрика

- колко отнема всяка заявка
- колко отнемат заявките средно

M/M/1

Поасонов процес

$$P[N(t+\tau) - N(t) = k] = \frac{e^{-\lambda\tau}(\lambda\tau)^k}{k!} \qquad k = 0, 1, \ldots,$$



$$Q = \begin{pmatrix} -\lambda & \lambda & & & & \\ \mu & -(\mu+\lambda) & \lambda & & & \\ & \mu & -(\mu+\lambda) & \lambda & & \\ & & \mu & -(\mu+\lambda) & \lambda & \\ & & & & & \ddots \end{pmatrix}$$

Аналитично решение

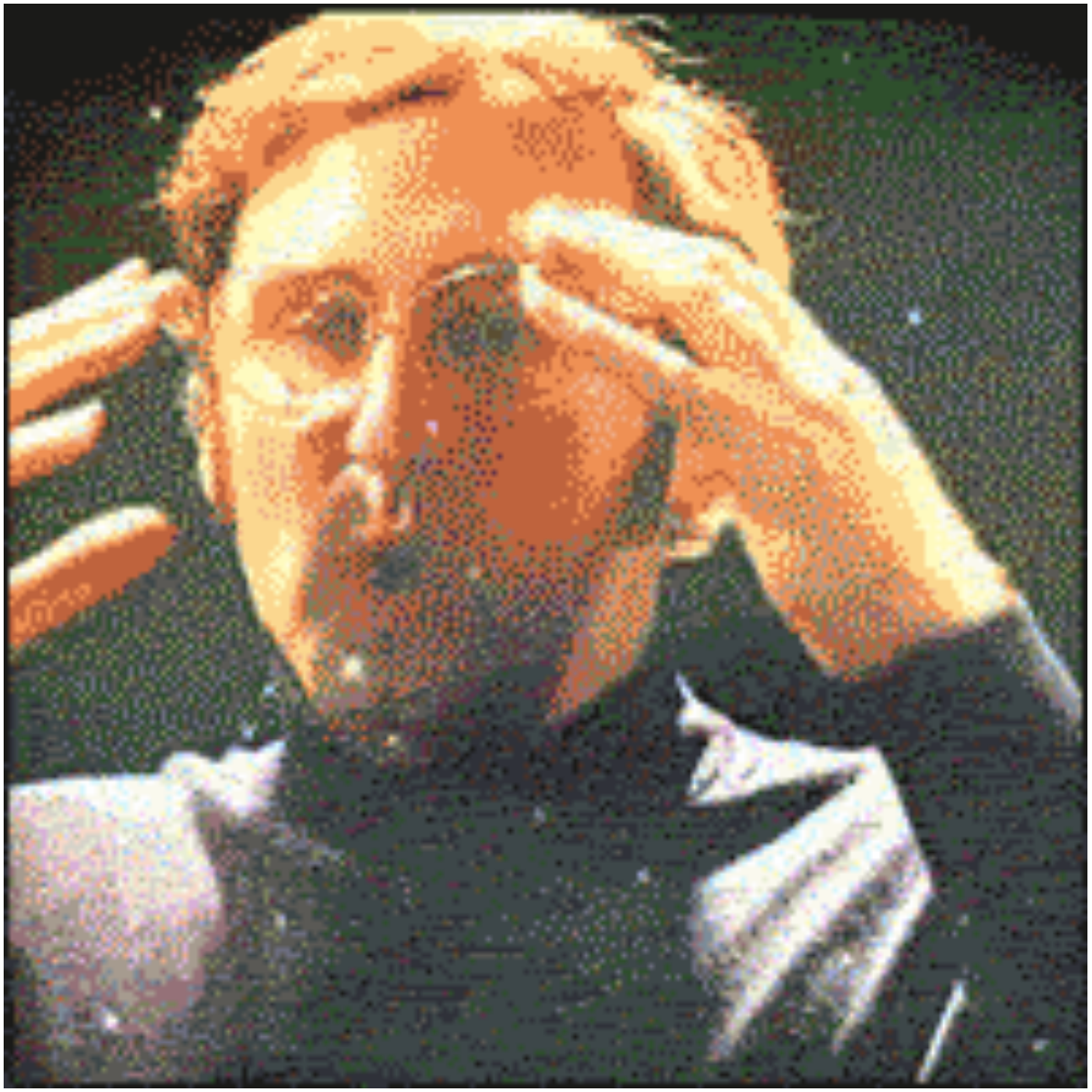
Пораждаща функция на чакането

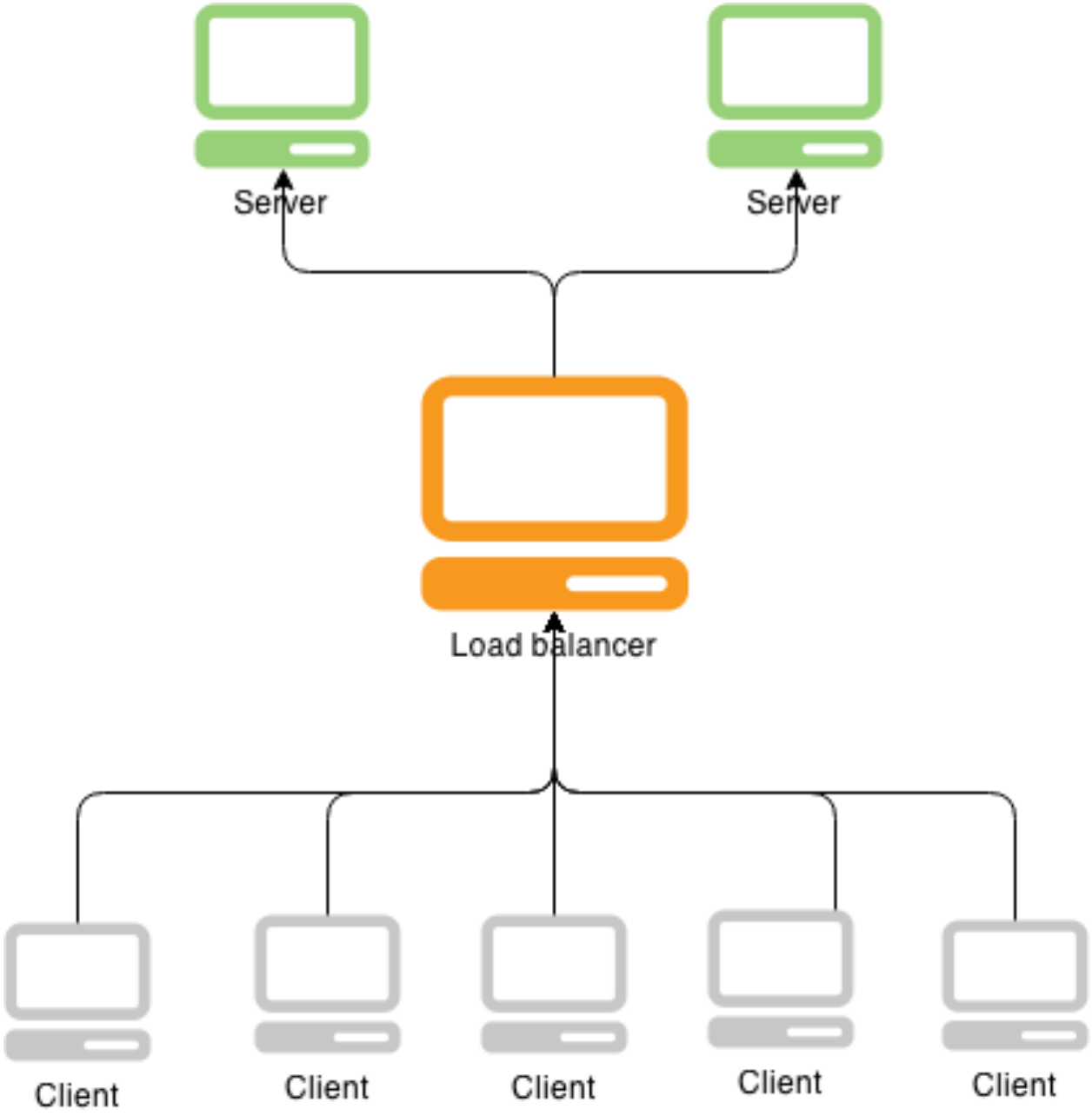
$$f(t) = \begin{cases} (\mu - \lambda)e^{-(\mu-\lambda)t} & t > 0 \\ 0 & \text{otherwise.} \end{cases}$$

M/D/1

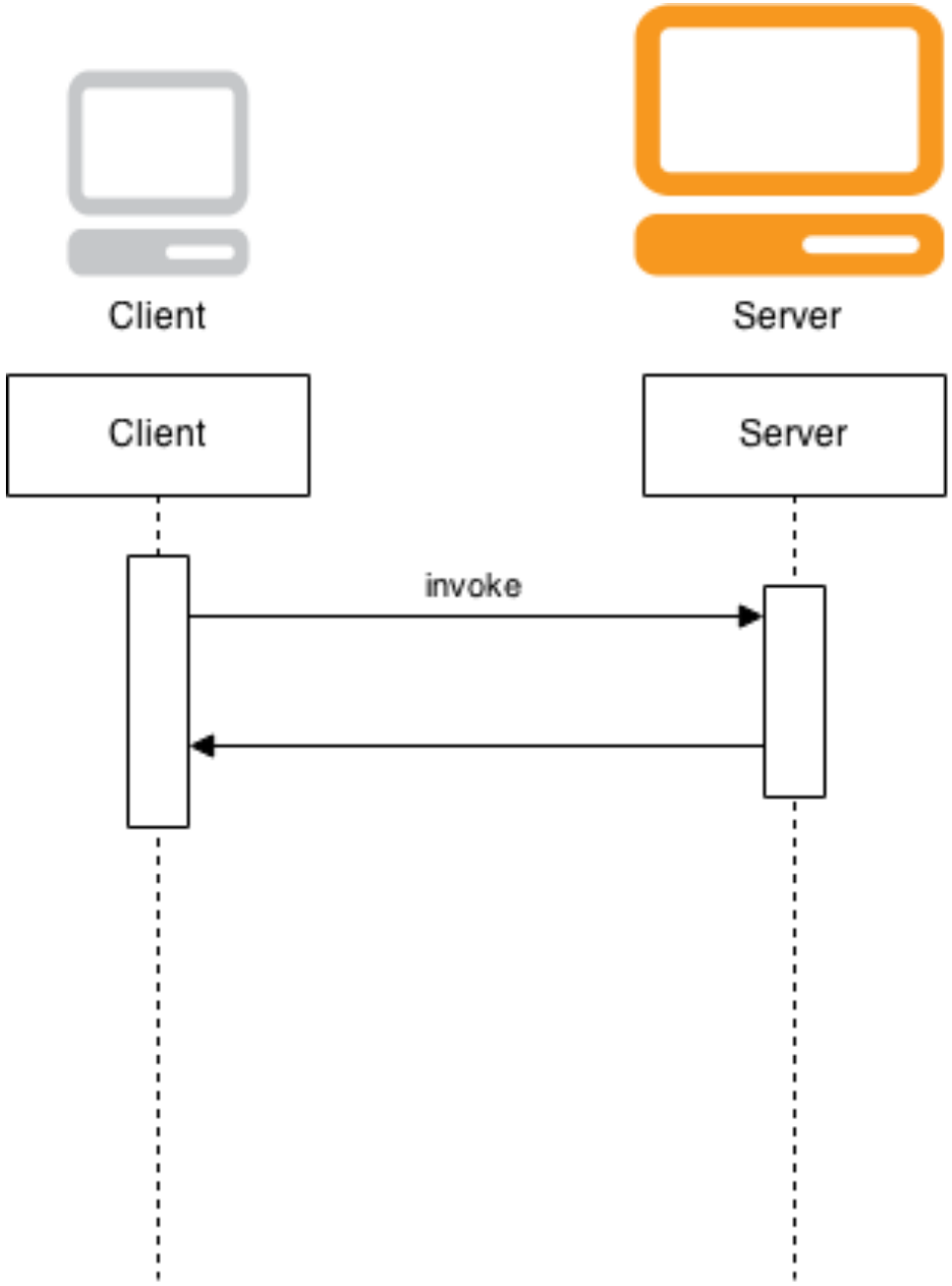
Erlang

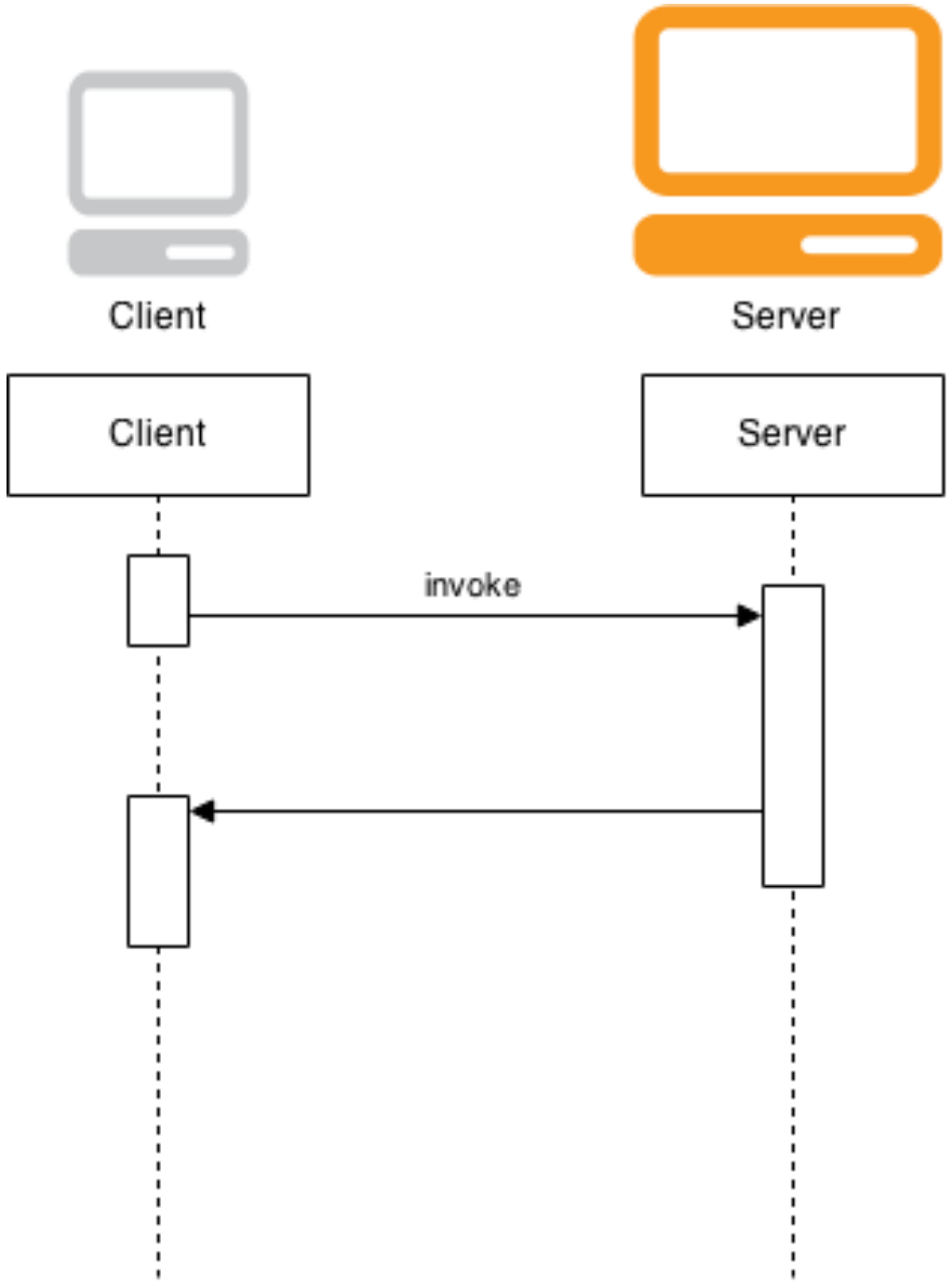
Agner Krarup



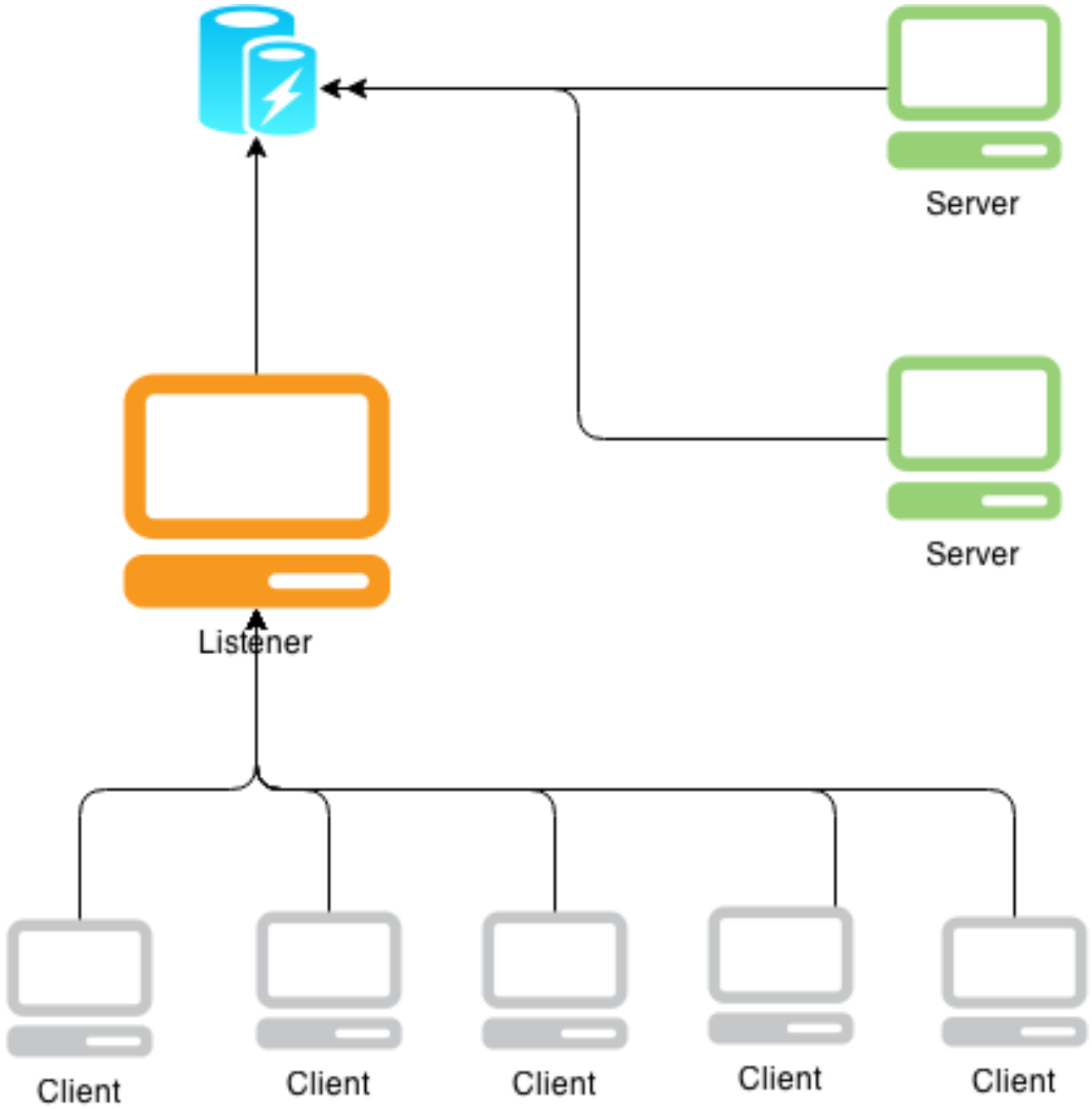

Server
Server
Load balancer
Client
Client
Client
Client
Client

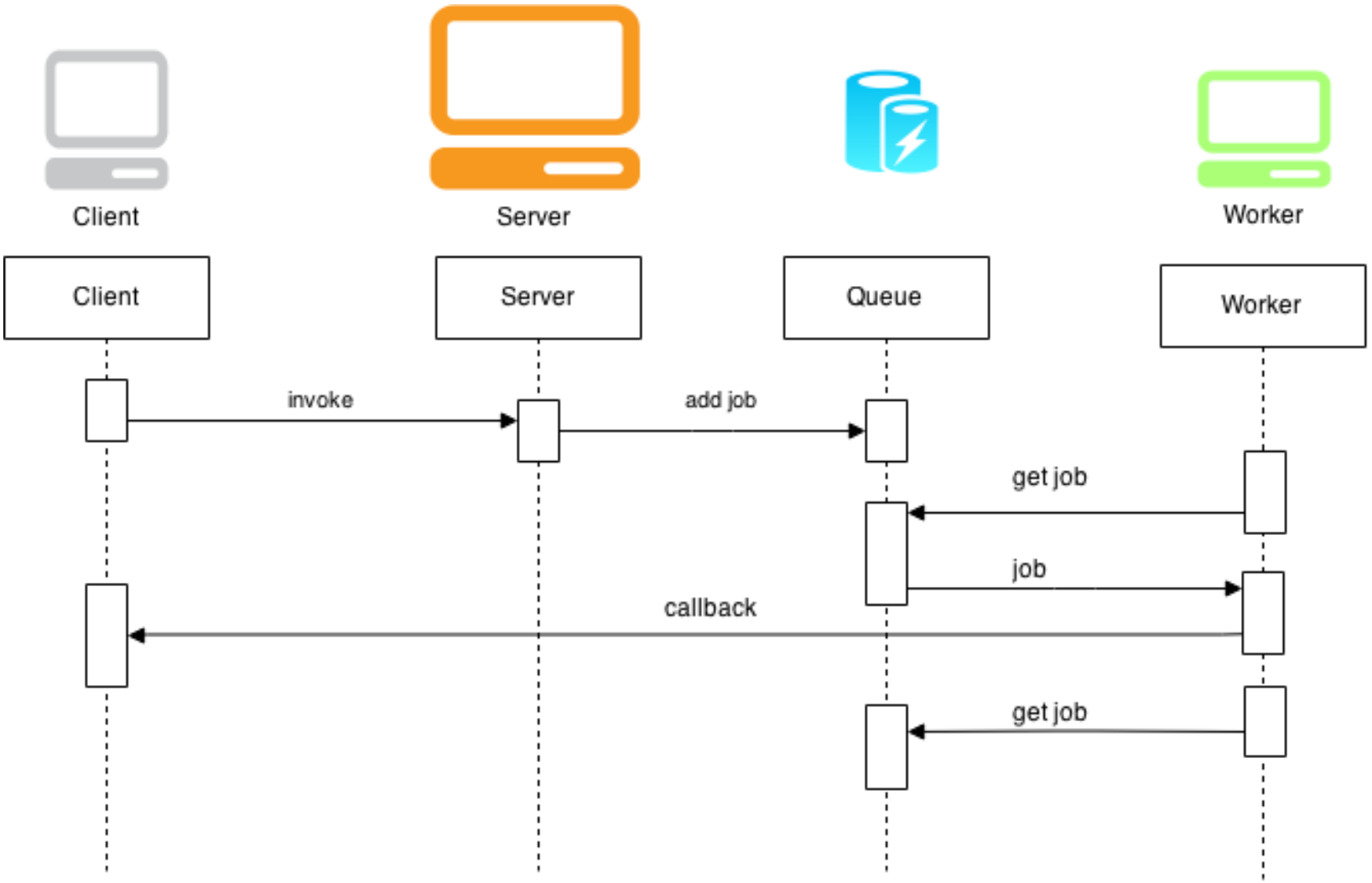
Асинхронни извиквания


Client
Server
Client
Server
invoke


Client
Server
Client
Server
invoke

M/M/k


Server
Server
Listener
Client
Client
Client
Client
Client


Client
Server
Worker
Client
Server
Queue
Worker
invoke
add job
get job
job
callback
get job

Видове задачи

- планирана/реактивна
- IO/CPU
- приложна/подръжна

Характеристики

- приоритет
- тип
- отнемат различно време

Екстри

- да ги разпределяме на различни машини
- повтаряме, рестартираме

Resque

- Jobs
- Redis
- json

Job - Покани приятел

```
class InviteFriendJob
  def self.perform(from_email, to_email)
    Mailer.invite(from_email, to_email)
  end
end
```

Опашка

```
class InviteFriendJob
  @queue = :mail
  ...
```

Нареждане

```
Resque.enqueue(
   InviteFriendJob,
   “niki@abv.bg”,
   “pe6o@mail.bg”)
```

Запис

```
{
    "class": "InviteFriendJob",
    "vars": {
        "from": "niki@abv.bg",
        "to": "pe6o@mail.bg"
    }
}
```

Друг запис - DSL

```
class Person
   def invite(other)
      Mailer.invite(self.email, other.email)
   end
end

niki = Person.find(5)
pesho = Person.find(100)

niki.async(:invite, pesho)
```

Алтернативи

DelayedJob

Goworker

Jesque

Функционално

отклонение

Imperative vs. Functional Separation of Concerns

```
List<String> errors = new ArrayList<>();
int errorCount = 0;
File file = new File(fileName);
String line = file.readLine();
while (errorCount < 40 && line != null) {
     if (line.startsWith("ERROR")) {
          errors.add(line);
          errorCount++;
     }
     line = file.readLine();
}
```

```
List<String> errors =
     Files.lines(Paths.get(fileName))
          .filter(l -> l.startsWith("ERROR")
          .limit(40)
          .collect(toList());
```

MapReduce

- Input
- Map
- Partition
- Sort
- Reduce
- Output

Игра

Apache Hadoop

- HDFS
- HBase
- Scheduler, Sorter
- ...

Пример - Map

```
public static class Map extends MapReduceBase
        implements Mapper<IntWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(IntWritable key, Text value,
        OutputCollector<Text, IntWritable> output,
        Reporter reporter) throws IOException {

        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            output.collect(word, one);
        }
    }
}
```

Пример - Reduce

```
public static class Reduce extends MapReduceBase
        implements Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterator<IntWritable> values,
        OutputCollector<Text, IntWritable> output,
        Reporter reporter) throws IOException {

        int sum = 0;
        while (values.hasNext()) {
            sum += values.next().get();
        }
        output.collect(key, new IntWritable(sum));
    }
}
```

Конфигурация

```
public static void main(String[] args) throws Exception {
        JobConf conf = new JobConf(WordCount.class);
        conf.setJobName("wordcount");
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        conf.setMapperClass(Map.class);

        conf.setReducerClass(Reduce.class);
        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);
        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));

        JobClient.runJob(conf);
    }
}
```

Облачни Hadoop-и

- Elastic Compute Cloud
- Azure

Кой го ползва?

Кой ли не!

- Амазон
- FB
- Yahoo

Powered by

Екстра

- Хадуп срещу Баш
- Полезни умения

Финал