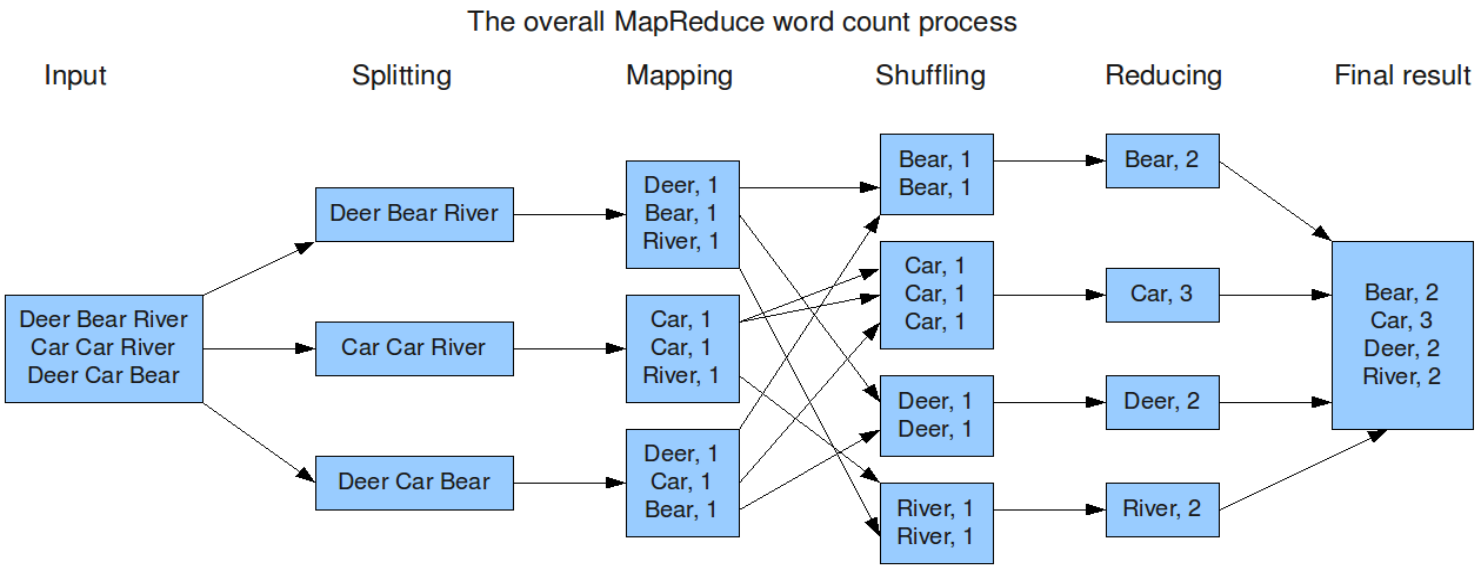
PiEstimator

Grupo 11: Alexandre Baruque ra104520
Carlos Gregorek ra104721
Marcelo Santos ra106140
Yuri Brito ra107001

Map Reduce

- Largamente utilizado em computação distribuída
- Características:
  - Paralelização e distribuição de tasks
  - Tolerância a falhas

Map Reduce



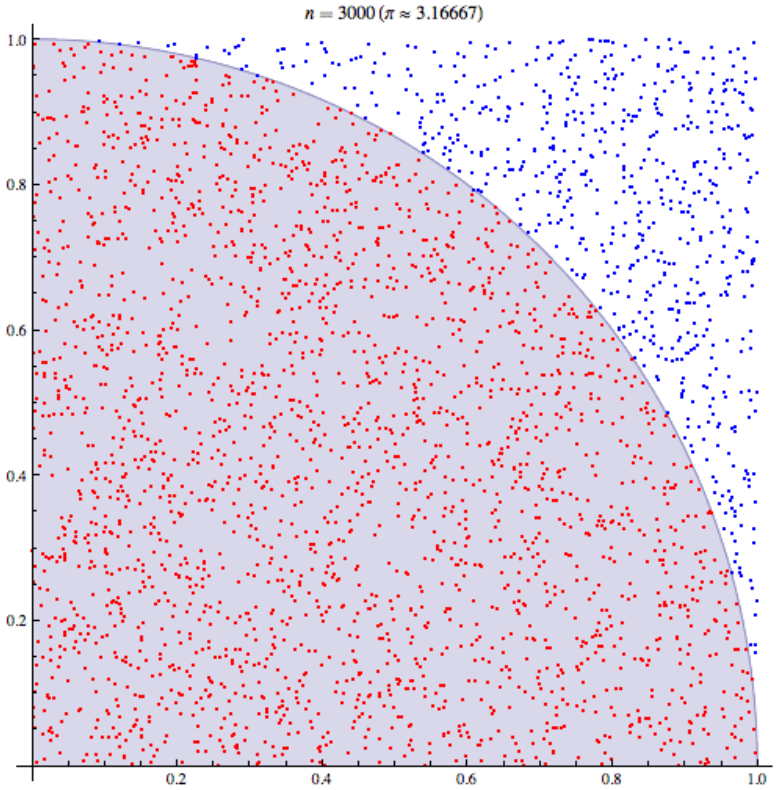
fonte: http://www.cs.uml.edu/~jlu1/doc/source/report/MapReduce.html

Método de Monte Carlo

- Utiliza-se de números pseudo-aleatórios para calcular numericamente a integral de funções.

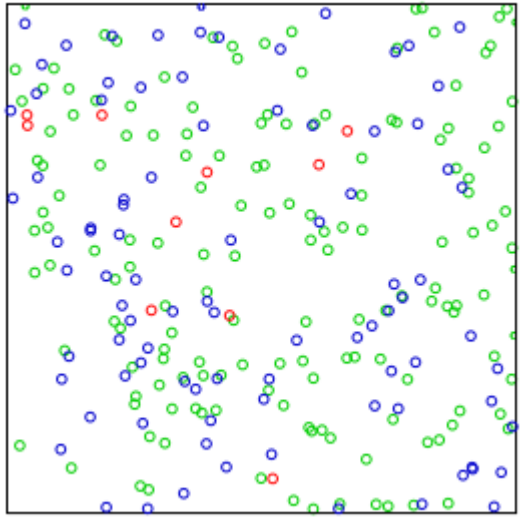
- Quanto mais tempo ele é executado maior sua precisão.

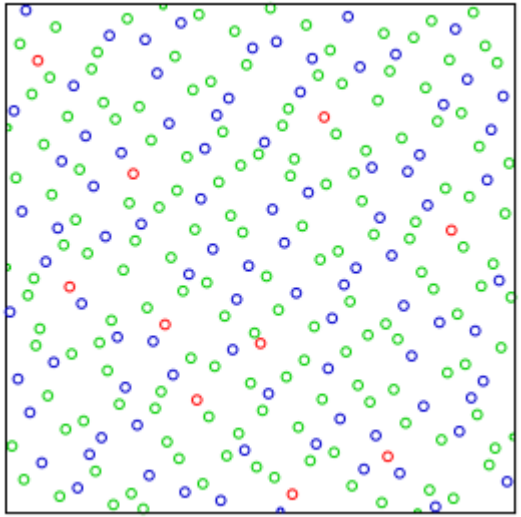
- Facilmente paralelizável.



Método Quasi-Monte Carlo

- Faz uso de uma sequência de baixa discrepância ao invés de números aleatórios e por isso converge mais rápido que somente o método de monte carlo.



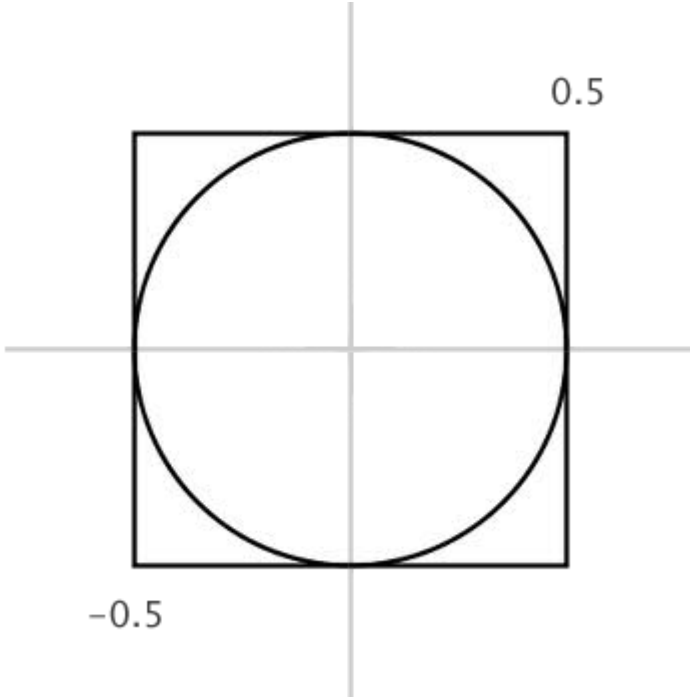


Sequência de Halton

```
FUNCTION (index, base)
  BEGIN
    result = 0;
    f = 1 / base;
    i = index;
    WHILE (i > 0)
    BEGIN
      result = result + f * (i % base);
      i = FLOOR(i / base);
      f = f / base;
    END
    RETURN result;
  END
```

```
1 =   1.0 => 0.1   = 1/2
2 =  10.0 => 0.01  = 1/4
3 =  11.0 => 0.11  = 3/4
4 = 100.0 => 0.001 = 1/8
5 = 101.0 => 0.101 = 5/8
6 = 110.0 => 0.011 = 3/8
7 = 111.0 => 0.111 = 7/8
```

PiEstimator- Map



Map method.

Parameters:

offset samples starting from the (offset+1)th sample.
size the number of samples for this map
out output {ture->numInside, false->numOutside}
reporter

```
148
149     public void  map(LongWritable offset,
150                      LongWritable size,
151                      OutputCollector<BooleanWritable, LongWritable> out,
152                      Reporter reporter) throws IOException {
153
154       final HaltonSequence haltonsequence = new HaltonSequence(offset.get());
155       long numInside = 0L;
156       long numOutside = 0L;
157
158       for(long i = 0; i < size.get(); ) {
159         //generate points in a unit square
160         final double[] point = haltonsequence.nextPoint();
161
162         //count points inside/outside of the inscribed circle of the square
163         final double x = point[0] - 0.5;
164         final double y = point[1] - 0.5;
165         if (x*x + y*y > 0.25) {
166           numOutside++;
167         } else {
168           numInside++;
169         }
170
171         //report status
172         i++;
173         if (i % 1000 == 0) {
174           reporter.setStatus("Generated " + i + " samples.");
175         }
176       }
177
178       //output map results
179       out.collect(new BooleanWritable(true), new LongWritable(numInside));
180       out.collect(new BooleanWritable(false), new LongWritable(numOutside));
181     }
182   }
```

PiEstimator- Reduce

Accumulate number of points inside/outside results from the mappers.

Parameters:

`isInside` Is the points inside?
`values` An iterator to a list of point counts
`output` dummy, not used here.
`reporter`

```
207
208     public void reduce(BooleanWritable isInside,
209                        Iterator<LongWritable> values,
210                        OutputCollector<WritableComparable<?>, Writable> output,
211                        Reporter reporter) throws IOException {
212       if (isInside.get()) {
213         for(; values.hasNext(); numInside += values.next().get());
214       } else {
215         for(; values.hasNext(); numOutside += values.next().get());
216       }
217     }
```

PiEstimator

Grupo 11: Alexandre Baruque ra104520
Carlos Gregorek ra104721
Marcelo Santos ra106140
Yuri Brito ra107001