

Spark 平台实现 Smith-Waterman 算法

陈允文, 车丹丹

2018 年 9 月 6 日

1 概要

本报告主要讲了如何在 Spark 平台上实现 Smith-Waterman 算法. 首先, 介绍了基本的 SW 算法: SW 算法是对两条序列进行比对, 得到局部最优比对. 当查询某条序列与数据库中的哪些序列的相似度最大的时候 (1 vs n), 我们就需要做 n 次比对, 这是一个相当耗时的工作, 因此我们可以想到利用 Spark 平台, 将数据库中的序列分割到不同的结点, 并行的进行比对 (与 mapreduce 中的 map 过程相似), 最后再把所有比对的结果汇总 (reduce 过程), 这样就可以大幅度的减少比对时间.

2 Smith-Waterman

设要比对的两序列为 $A = a_1a_2 \dots a_n$ 和 $B = b_1b_2 \dots b_m$, 其中 n, m 分别代表 A, B 的长度.

2.1 确定置换矩阵和空位罚分方法

在本实验中, 我们采用 BLOSUM50 作为置换矩阵, 空位延伸罚分模型作为空位罚分的计算方式. 空位延伸罚分就是当存在连续的多个空位的时候, 我们加重开头的空位的罚分权重, 降低后续的空位的罚分的权重, 使得序列比对更具有生物学意义. 例如当有连续 k 个空位的时候罚分可以表示成 $W_k = u(k - 1) + v$, 本实验中 $u = 1, v = 5$.

- $s(a, b)$ 表示元素 a, b 之间的相似得分, 可以通过访问置换矩阵的 (a, b) 元素得到.

- 长度为 k 的空位罚分.

2.2 生成得分矩阵

1. 创建得分矩阵 $M_{(n+1,m+1)}$, 初始化其首行首列. $M_{k0} = M_{0l} = 0, (0 \leq k \leq n, 0 \leq l \leq m)$
2. 从左到右, 从上到下对矩阵的剩余部分按如下公式进行填充.

$$M_{i,j} = \max \begin{cases} M_{i-1,j-1} + s(a_i, b_j), \\ \max_{k \geq 1} \{M_{i-k,j} - W_k\}, \quad \text{其中 } (1 \leq i \leq n, 1 \leq j \leq m). \\ \max_{l \geq 1} \{M_{i,j-l} - W_l\}, \\ 0 \end{cases}$$

2.3 回溯

1. 找到得分矩阵中的最大元素 M_{i_0,j_0} ,
2. 然后找出 $\max\{M_{i_0,j_0-1}, M_{i_0-1,j_0}, M_{i_0-1,j_0-1}\}$, 并记录下来新的坐标为 (i_{new}, j_{new})
3. 对新的坐标, $i_0 := i_{new}, j_0 := j_{new}$, 执行上一步. 直到遇到矩阵中的 0 元素.
4. 根据回溯的坐标, 对应上序列, 就可以得到比对之后的两个序列的相似子序列.

3 Spark

3.1 安装

直接下载然后解压就可以了. 版本是 spark-2.3.1-bin-hadoop2.7. Download Spark 然后是要配置环境, Spark/conf/目录下新建一个 spark-env.sh 文件, 并写入以下内容:

```
SPARK_WORKER_CORE=5
SPARK_WORKER_INSTANCES=5
```

SPARK_WORKER_MEMORY=6g

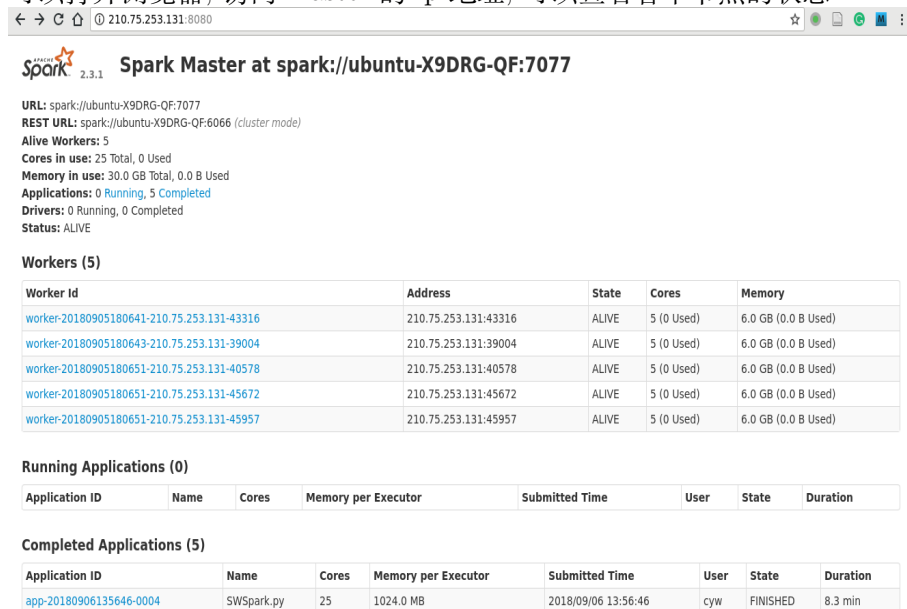
分别定义了每个 worker 的核数,worker 的数量, 以及每个 worker 的内存, 其中的数字可以依据实际情况进行修改.

3.2 使用

我们使用的是 standalone 模式来模拟一个集群. 在 Spark 的文件夹里输入以下命令即可:

```
./sbin/start-master.sh
./sbin/start-slaves.sh spark://127.0.1.1:7077
./bin/spark-submit --master spark://127.0.1.1:7077 SWSpark.py
```

可以打开浏览器, 访问 master 的 ip 地址, 可以查看各个节点的状态:



The screenshot shows the Spark Master web interface at the URL `spark://ubuntu-X9DRG-QF:7077`. The interface displays the following information:

- URL:** `spark://ubuntu-X9DRG-QF:7077`
- REST URL:** `spark://ubuntu-X9DRG-QF:6066 (cluster mode)`
- Alive Workers:** 5
- Cores in use:** 25 Total, 0 Used
- Memory in use:** 30.0 GB Total, 0.0 B Used
- Applications:** 0 Running, 5 Completed
- Drivers:** 0 Running, 0 Completed
- Status:** ALIVE

Workers (5)

Worker Id	Address	State	Cores	Memory
worker-20180905180641-210.75.253.131-43316	210.75.253.131:43316	ALIVE	5 (0 Used)	6.0 GB (0.0 B Used)
worker-20180905180643-210.75.253.131-39004	210.75.253.131:39004	ALIVE	5 (0 Used)	6.0 GB (0.0 B Used)
worker-20180905180651-210.75.253.131-40578	210.75.253.131:40578	ALIVE	5 (0 Used)	6.0 GB (0.0 B Used)
worker-20180905180651-210.75.253.131-45672	210.75.253.131:45672	ALIVE	5 (0 Used)	6.0 GB (0.0 B Used)
worker-20180905180651-210.75.253.131-45957	210.75.253.131:45957	ALIVE	5 (0 Used)	6.0 GB (0.0 B Used)

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

Completed Applications (5)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20180906135646-0004	SWSpark.py	25	1024.0 MB	2018/09/06 13:56:46	cyw	FINISHED	8.3 min

3.3 RDD

RDD(Resilient Distributed Dataset, 弹性式分布数据集) 是 Spark 的对数据的核心抽象. Spark 对数据的所有操作不外乎创建 RDD, 转化 RDD 以及调用 RDD 操作进行求值. 而在这一切的背后,Spark 会自动将 RDD 分发

到集群上, 并将操作并行化执行. 对于这次实验, 我们将 db.file 读入 Spark 作为一个 RDD 实例, 然后 Spark 就能自动将 db.file 中的文件分发到集群上, 并行化执行操作.

4 核心代码

下面是核心代码:

```
# python code
0. sub_mat = MatrixInfo.blosum50
1. db = sc.textFile('db.file')
2. db = db.map(lambda x: x.split(','))
3. query = sc.textFile('query.file')
4. query = query.collect()[0]
5. aligned = db.map(lambda x: alignment(dbs=x, query=query))
6. aligned = aligned.sortBy(lambda x: x[1], ascending=False)
```

第 0 行: 把 BLOSUM50 作为置换矩阵

第 1-4 行: 读入蛋白质数据库序列和查询序列

第 5 行: 使用 Smith-Waterman 算法对序列进行比对, 其中的 alignment 是在外部定义好的 SW 算法

第 6 行: 将序列按得分从高到低进行排序

5 实验结果

为了验证 spark 的准确性, 我们准备了两个实验, 一是利用前边提到的配置, 在 spark 平台运行 Smith-Waterman 算法; 另一个是则是完全单机, 使用纯粹的 python 编程语言, 运行 Smith-Waterman 算法, 最后我们都取了前 7 个结果 (3,5 都只是前 7 的一个真子集) 进行比较, 来确定 spark 平台的准确性.

5.1 集群

0	UniRef100_Q91G54	445.0
1	UniRef100_Q197F3	112.0
2	UniRef100_Q6GZX4	100.0
3	UniRef100_Q91G57	93.0
4	UniRef100_Q197D2	90.0
5	UniRef100_Q197E9	87.0
6	UniRef100_Q91G88	85.0

5.2 单机

0	'UniRef100_Q91G54'	445.0
1	'UniRef100_Q197F3'	112.0
2	'UniRef100_Q6GZX4'	100.0
3	'UniRef100_Q91G57'	93.0
4	'UniRef100_Q197D2'	90.0
5	'UniRef100_Q197E9'	87.0
6	'UniRef100_Q91G88'	85.0

通过对比我们发现,spark 平台的分布式运行出来的结果与单机情况下运行出来的结果一致.