SUPPLEMENTARY INFORMATION

2. GENOME ASSEMBLY

2.1 MARVEL assembly pipeline

MARVEL 软件组装包含多个步骤，在组装的过程中可以进行轻微的调整，但是分析流程是一样的。在组装蝾螈的过程中，由于较大的基因组和重复率，步骤被调整来减少计算时间和储存空间。最初计算的对于一个 daligner 比对所需的计算时间和储存预估超过了数百万的 CPU 时间和大于 2PB 的储存，通过使用动态重复 masking 服务器，完整的组装时间减少到了 150,000 个 CPU 时间和 120TB 的储存。

MARVLE 组装主要包括三个主要的阶段：准备阶段（setup phase），纠错阶段（patch phase），组装阶段（assembly phase）。首先，pacbio 数据从测序数据中提取出来，保存在内部数据库中。纠错阶段，检测和纠正 reads 中的错误，包括去除接头，聚合链转移，连接嵌合体，它是组装的主要障碍，纠错后的 reads 被用于最后的组装。在组装阶段，在重叠 read 中，由于不好的测序导致的短序列比对（一般是 50bp）被修补。接下来是重复注释和 overlap 图的生成，最后找出最好的路线来生成组装 contigs。

2.1.1 Overview

Reads 被重叠，然后纠错，再次重叠。最初的 overlap 不是所有的 block 都进行一对一的比对，但是部分比对需要完成，一个 read 只要比对总共 15X 覆盖度的 reads 就足够了，这样可以得到足够的信息用来纠错。当第二次比对结束，reads 质量，reads 修补信息和重复区域信息被注释。这些信息被用来，过滤因为重复造成的比对，检测嵌合体和其他会影响后面组装的错误。对比对信息过滤后，真正的 overlap 信息留了下来，基于比对信息，我们来构建 overlap 图然后提取 contigs。

2.1.2 Requirements

2.1.3 Major features

2.1.4 Detailed description of the three phases in MARVEL

- **准备阶段**

每一个 SMRTcell 上的每一个 ZMW 中得分最高的 read 被使用 H5dextract 提取出来。下面的公式被用来计算 reads 的得分：

Read_score= 0.2 * read_length + 0.4 * read_quality + 0.4 * slow_polymerase_events

Fasta 格式的文件被转换为一个 daligner 数据集提供了一个 4X 的压缩因子。通过使用 FA2db 来实现此步骤，在这一过程中，reads 的情况被统计，可以通过 DBstats 来查看。因为一些基因组的 read 数据集可能超过数百 G，直接的完全的比对是不可行的，因此有必要分块进行比较。为此，数据集被分成 400M 大小的块或者更小。

如果比对在一个好性能的集群上跑的话，我们使用 HPCaligner 来制定 daligner 数据集的所有块进行比对的策略。同样的合并所有重叠文件的策略也被制定。HPCaligner 仅仅对块矩阵的上三角线进行创建任务。这是因为 daligner 产生均匀的重叠文件，也就是说，当 daliger 计算块 i

和块 j 之间的重叠文件 ij 时，对应的重叠文件 ji 也在计算。为了减少硬盘上比对文件的大小，仅仅对于 readi 和 readj 之间比对的锚定位置（也叫 trace points）进行储存。然而，为了快速计算 readi 和 readj 之间的比对，基于 readj 的锚定位置也要保存。每 100bp 设置一个记录点。

- **修补阶段**

局部比对计算是组装过程中最消耗时间的一步，特别是对于超过 1G 或高重复的基因组来说，可导致 CPU 计算时间达到几个月。因此，比对计算可以下面 5 种模式来实现（对于蝾螈模式 4 被使用）：

模式 1：计算所有局部比对
模式 2：动态 mask 重复区域
模式 3：动态 mask 包含的 reads（也就是说，完全包含在其他 reads 中的 reads）
模式 4：动态 mask 重复区域和包含的 reads
模式 5：动态 mask 重复区域和包含的 reads，排除 flanking 区域的 mask

动态 mask 使用一个 mask 服务器来管理，这个服务器可以追踪每条 read 的局部比对数量和它们在 read 中的精确位置。先前计算的 mask 轨迹，如果有效，同样可以被用来替代 masking server 。另外，masking server 可以用先前的计算记录来初始化。这样做可以使第一次 daligner 的计算重复记录为组装阶段的第二次 daligner 做准备。然而，这需要对先前的重复注释进行相应的调节，因为重复注释可能被改变在修补 reads 的过程中。

在 daligner 比对 blocki 和 blockj 之前，它从 masking 服务器这些 block 的 mask 记录。网络故障几乎可以忽略，因为 block 的特定记录相对较小（<1M）。一旦 daligner 完成比对，mask 服务器就会分析比对，更新 read 的统计信息和 masking 信息。

对于每一个 block 数据集，含有这一 block 重叠信息的合并文件被创建，此文件叫作 DB.i.las，用 LAmerge 来创建。

基于先前的比对计算，从原始的 reads 中导出一系列新的 reads，它们的局部比对和质量计算被称为修补，由 LAfix 来实现。在 LAfix 之前，read 的质量（每 100bp 大小的分辨率）和修补信息被计算使用 LAq。

然后 LAfix 排除最糟糕的测序错误，这些错误被其他的 reads 来验证。这些是基于局部比对和先前产生的修补记录。错误被修补，包括大于 500bp 的 gap 被填补。为了填补 gap，它们必须被至少 3 条高质量 reads 所支持。同样坏的 read 片段被好的片段代替。包含修补过 reads 的新的数据集被创建，被用来下面的比对。

- **组装阶段**

首先，组装阶段，先前修补过 reads 的比对被计算。对于组装阶段的第二次 daligner，第一次 daligner 的重复注释可以被再次利用。下面的步骤是基于产生的重叠的。

首先，成对比对中存留的少于 50bp 的 gap 可以使用 LAstich 进行缝合。然后，使用 LAq 所有 reads 的质量分数被重新计算，修补记录生成，然后用 LAtrim 来进行修补。

紧随其后的是 LArepeat 的重复注释。重复区域被认定是基于重叠区域的覆盖度。如果一个潜在的重复区域的覆盖度是基因组期望覆盖度的二倍，这个区域被注释为一个重复。如果所有的碱基是期望覆盖度的至少 1.5 倍，这些碱基会被添加到重复区域，这是为了补偿重复区域的波动性。一旦覆盖度降到期望覆盖度的 1.5 倍以下，这个重复区域将会被关闭。这之后，使用

LAgap 来扫描 reads 来补 gap。Gap 存在这一阶段，是因为，一个 read 是嵌合体或者在 reads 中存在先前没有检测到的 left-overs of the “weak”区域。为了解决一个 gap，来自较短一边的重叠将会被丢弃。Gap 的解决是通过一系列的修剪实现的。

现在对于没有大规模错误的适当修剪过的 reads，我们可以使用 LAfilter 对重复进行一个最后的过滤，去除局部比对和因重复造成的比对。另外，MARVEL 试图去捕获可以跨域完整重复分子的重复区域的重叠。这样做是试图去分解大的重复区域为一系列独一无二的模块。

MARVEL 软件对蝾螈的组装

一个标准的组装由一下这些步骤组成：

1.H5dextract--从测序仪产生的测序数据中提取 reads
2.DBprepare—为重叠创建数据集，将其分成块，创建一个计划（一系列计算重叠和合并的命令）
3.Daligner—用 MARVEL 软件进行重叠计算
4.LAq—计算 read 的质量
5.LAfix—修复 reads 中大规模的错误
6.DEprepare—为重叠创建数据集，将其分块，创建一个计划（一系列计算重叠和合并的命令）。这一步是基于修补过的 reads 的。
7.LAstitch—修复已经被打断的比对，一个损坏（left-over bad）的区域
8.LArepeat—创建 reads 的重复注释
9.TKhomogenize—用局部比对转移（transfer）重复注释
10.LAq—计算 read 质量和修补（trim）信息
11.LAgap—寻找比对破裂的边缘，丢弃掉最短 reads。如果 gap 在重复区域中间，就予以忽略。
12.LAq—更新修补记录，仅仅对于因 LAgap 造成的改变。
13.LAfilter—清除所有局部和因重复引起的比对
14.OGbuild—建立重叠图
15.OGtour—寻找重叠图，导出 contigs

各模块的功能

1.TKmerge -- 合并每一个 block 产生的单个注释记录 -d 合并之后删除原始记录
2.LAq --创建注释记录，包括 read 的质量记录和 trim 信息
3.LAfix--基于比对信息，修补（patch）reads 中较大的测序错误。错误包括聚合链的变化，接头的缺失，缺失的序列，噪音过大的序列，随机噪音的插入
4.LAstitch—缝合（stitch）比对。如果这个比对不在一个或两个比对的噪音区，或会分成两个或更多的比对，我们将对它们进行缝合。
5.TKhomogenize --通过转移 A read 的注释到 B read 的注释来创建一个新的注释记录
6.LAtrim --根据修复（trim）记录对输入文件（*.las）进行修复并相应的更新比对信息。
7.LArepeat --基于异常现象的覆盖度检测重复元素并用它们创建一个注释记录
8.LAfilter --根据各种规范过滤输入文件（*.las）
9.LAgap --检测没有被任何 reads 跨域的 gaps 或区域，丢掉所有在 gap 一侧的比对信息。
10.OGbuild --基于输入文件（*.las）的比对信息创建重叠图