Gliederung

July 30, 2015

1 Einleitung

1.1 „Aktueller Stand der Forschung“

- NGS
- Massen an Daten, Größe der Dateien
- Schwierigkeiten der Speicherung und Analyse

1.2 Motivation/Ziel

- Verringerung von Speicherplatz sowohl auf Dateiebene als auch (bzw. primär) für den Arbeitsspeicher
- Funktionalität in und zur Benutzung von SeqAn (DeltaMap)

1.3 Zusammenfassung aller anderen Bereiche

- Wie wurde das Ziel umgesetzt?
- Was kam heraus?
- Bewertung der Lösung
- Was lässt sich noch Verbessern?

2 Hintergrund

2.1 Speicheranalyse genomischer Daten

- Wie sieht genomische DNA grob aus
- Wie wird genomische DNA gespeichert (FASTA, VCF)
- Beispiele
- Ansprüche an einen Computer (bei bestimmten Algorithmen) bezogen auf Geschwindigkeit und vor allem Speicherplatz

2.2 Lösungsansätze der Literatur: Referentielle Kompression

- Kurz ansprechen was es sonst gibt: Lempel-Ziv...
- Was ist referentielle Kompression?
- (Was sind Delta-Events)
- Welchen Ansätze wurden schon nachgegangen, welche Algorithmen gibt es bereits für Kompression von Dateien?
- welche Algorithmen gibt es bereits die auf komprimierten Daten Algorithmen laufen lassen kann?
- SeqAn (Delta Map)

3 Methoden

- Grundidee des Algorithmus'
- Input, Output
- Defitionen: Seed, SearchField, DeltaEvent, DependantRegion, JournaledString/Entrie, DeltaMap

3.1 Variant Calling

3.1.1 Finde Seeds

- Q-gram Index (1. Sequenz des Inputs, Openadressing())
- Jede weitere Sequenz wird gegen den Index nach Hits durchsucht -> Hit = Seed
- Problem: O(n) sehr langsam vor allem bei Repeats
- Lösung für Repeats (Definition, Einschränkung)
- (Lösung für Schnelligkeit im nächsten Absatz)

3.1.2 Idee der Funktion (parallel)fastFirstSeeding()

- Hits(Seeds) werden wenn sie gefunden werden erweitert (extend)
- (Maximale Seed Größe, Grenze um Chaining Problem zu beschleunigen)[vielleicht erst bei Chaining-section]
- Scoring dieser Seeds (nach Länge und Lage zur Hauptdiagonalen)
- Offset = Länge des Seeds mit besten Score
- Anmerkung: Parallelisierung durch Aufteilen in Abschnitte

3.1.3 Chaining Algorithmus

- Seeds werden zu einem aligniert
- kurze Beschreibung des Algorithmus'
- Beschleunigung durch Maximale Seed Größe

3.1.4 Iterativer Schritt

- fastFirstSeeding ist sehr grob
- um trotzdem sensitivität zu erhalten wird ein iterativer Schritt durchgeführt
- N's werden zugelassen was sich aber oft negativ auf die Laufzeit auswirkt -> ignorieren der N-qgramme beim seeding (sollte durch extendSeeds eh berücksichtigt werden)

3.1.5 Überführen von Seeds in Delta Events

- Kette von Seeds bilden ein Alignment
- Alignment-Repräsentation mit Delta-Events
- Überführung von Seeds in Delta-Events -> Manhattan Distanz

3.2 Event Processing (on reference)

- Prozessieren um zu komprimieren. Kurze Zusammenfassung:
- Zwei verschiedene Ansätze:
- 1. Kombinieren von gleichen Events -> "Multiples Alignment"; Anzahl von DeltaEvents wird verringert, was bei nachfolgenden Schritt besser für die Laufzeit und den Speicher ist und falls man die Records an sich speichert.
- 2. Änderung der Referenz Sequenz -> um Anzahl JournalEntries zu minimieren (eigentliches Ziel dieser Arbeit um Arbeitsspeicher Ansprüche zu verringern wenn nachfolgend Algorithmen angewendet werden wollen)

3.2.1 Prä-Prozessierung der Records(DeltaEvents)

- Sortierung der Events nach Position, Typ und Value
- Kombinieren von gleichen Events (Update des packed String)
- Gruppieren der Events zu abhängigen Regionen (Definition?, Grund/Erklärung von sonstigen Schwierigkeiten)
- jede abhängige Region wird dann unabhängig prozessiert

3.2.2 Prozessierung: Scoring

- Kurz nochmal: Was wird optimiert: JournalEntires
- Score = Anzahl von JournalEntries die addiert oder abgezogen werden würden FALLS das entsprechende Event in die Referenz eingebaut wird
- Beschreibung für das Scoring von SNP, DEL, INS, SV mit Beispielen
- Anmerkung: nicht optimal, da verschiedene Kombinationen von Events auf derselben Sequenz nicht beachtet werden. Wäre außerdem zu Laufzeit aufwändig.

3.2.3 Prozessierubg: Veränderung der Referenz

- Der beste Score (am meisten negativ) wird in die Referenz eingebaut
- Alle Sequenzen müssen daraufhin geupdated werden -> wie
- Unterscheidung zwischen abhängigen und unabhängigen Events
- Beschreibung der generischen Veränderung von Events mit Beispielen
- offset beachten
- Wiederholen des Scorings und der Veränderung der Referenz solange es event mit negativen Score gibt

3.3 Überführung von prozessierten Events in eine DeltaMap

- Wozu? -> Anwendbarkeit von Algorithmen
- Wie ....
- Speicherformat/Ausgabeformat

4 Resultate

- auf welchem Server wurde getestet
- auf welchen Daten wurde getestet, wo kommen sie her und wie wurden sie eventuell prozessiert

4.1 Laufzeit und Speicherplatz

- auf Seeding vs. Prozessieren von Events achten
- Abhängigkeit von Parametern?

4.2 Variant Calling/Seeding

- wie viele Seeds bei wie vielen vcf-entries
- Event verteilung auf dem Chromosom
- Abhängigkeit von Parametern?

4.3 Komprimierung

- Anzahl Records
- Anzahl JournalEntries
- Anzahl Bytes im Arbeitsspeicher

5 Diskussion

- Auswertung aller Punkte in results.... was fällt auf? wie kann man das Erklären?
- Bewertung...

6 Outline

- Zusammenfassung der Ergebnisse und Bewertungen
- Anwendungsmöglichkeiten
- Verbesserungsmöglichkeiten