BIG DATA

Project Presentation on Crystal Ball Event Prediction

Sachin Kushwaha (984681)

Sujan Thapa (984477)

Pseudo code - PAIR approach Mapper

```
class Mapper
        method Initialize
                H = new associative array

        method Map (docid a, doc d)
                for all term w in doc d do
                        for all term u in Neighbours(w) do
                                H(pair(w,u)) = H(pair(w,u))+1;
                                H(pair(w,*)) = H(pair(w,*))+1;
        method Close
                for all items in H do
                        Emit (pair p, count c)
```

Java code - PAIR approach Mapper

```
@Override
protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
    String[] allTerms = value.toString().split("\\s+");

    for (int i = 0; i < allTerms.length; i++) {
        String term = allTerms[i];
        for (int j = i + 1; j < allTerms.length; j++) {
            if (term.equals(allTerms[j]))
                break;

            Pair keyTerm = new Pair(term, allTerms[j]);

            Pair starTerm = new Pair(term, "*");

            if (!map.containsKey(starTerm)) {
                map.put(starTerm, 1);
            } else {
                map.put(starTerm, map.get(starTerm) + 1);
            }

            if (!map.containsKey(keyTerm)) {
                map.put(keyTerm, 1);
            } else {
                map.put(keyTerm, map.get(keyTerm) + 1);
            }
        }
    }
}
```

Pseudo code - PAIR approach Reducer

```
class Redcuer
        method Reduce(pair p, counts [c1; c2; c3; :::]
                total = 0;
                s = 0;
                if (pair p = p(w,*))
                        for all c in counts[c1; c2; c3; :::]
                                total += c;
                else
                        for all c in counts[c1; c2; c3; :::]
                                s += c;
                        Emit(pair p, s/total)
```

Java code - PAIR approach Reducer

```
int totalCount = 0;

@Override
protected void reduce(Pair pair, Iterable<IntWritable> values, Context context)
        throws IOException, InterruptedException {
    DecimalFormat decimalFormat = new DecimalFormat("0.000");
    if (pair.getValue().toString().equals("*")) {
        totalCount = sumAllValues(values);
    } else {
        int sum = sumAllValues(values);
        String value = decimalFormat.format((double) sum / (double) totalCount);
        context.write(pair, new DoubleWritable(Double.parseDouble(value)));
    }
}

private int sumAllValues(Iterable<IntWritable> values) {
    int sum = 0;
    for (IntWritable v : values) {
        sum += v.get();
    }
    return sum;
}
```

Input

34 56 29 12 34 56 92 29 34 12

92 29 12 34 79 29 56 12 34 18

Result - PAIR approach

(12, 18) 0.091

(12, 29) 0.182

(12, 34) 0.364

(12, 56) 0.182

(12, 79) 0.091

(12, 92) 0.091

(29, 12) 0.308

(29, 18) 0.077

(29, 34) 0.308

(29, 56) 0.154

(29, 79) 0.077

(29, 92) 0.077

(34, 12) 0.25

(34, 18) 0.083

(34, 29) 0.25

(34, 56) 0.25

(34, 79) 0.083

(34, 92) 0.083

(56, 12) 0.3

(56, 18) 0.1

(56, 29) 0.2

(56, 34) 0.3

(56, 92) 0.1

(79, 12) 0.2

(79, 18) 0.2

(79, 29) 0.2

(79, 34) 0.2

(79, 56) 0.2

(92, 12) 0.25

(92, 18) 0.083

(92, 29) 0.25

(92, 34) 0.25

(92, 56) 0.083

(92, 79) 0.083

Pseudo code – STRIPE approach Mapper

```
class Mapper {
        AssociativeArray H1;
        method Initialize() {
                H1 = new AssociativeArray();
        }
        method Map(docid a; doc d) {
                for all term w in doc d do
                        H2 = new AssociativeArray();
                        for all term u in Neighbors(w) do
                        if(H1[w] == null) {
                                H2[u] = 0;
                                H1.add(w, H2);
                        }
                        H1[w].H2[u].count++;
        }
        method Close() {
                Foreach(element e in H1)
                        Emit(e.Key, e.Value); // Emit(term w, stripe H2)
        }
}
```

Java code – STRIPE approach Mapper

```
@Override
protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
    String[] allTerms = value.toString().split("\\s+");

    for (int i = 0; i < allTerms.length; i++) {
        Text term = new Text(allTerms[i]);

        if (!map.containsKey(term)) {
            if(i == allTerms.length - 1)
                break;
            map.put(term, new MapWritable());
        }
        MapWritable mp = map.get(term);
        for (int j = i + 1; j < allTerms.length; j++) {
            if (term.toString().equals(allTerms[j]))
                break;

            Text u = new Text(allTerms[j]);
            if (!mp.containsKey(u))
                mp.put(u, ZERO);

            IntWritable val = new IntWritable(Integer.parseInt(mp.get(u).toString()) + 1);
            mp.put(u, val);
        }
    }

}
```

Pseudo code – STRIPE approach Reducer

```
class Reducer {
        method Reduce(term w, stripes [H1, H2, H3, ...]) {
                H_f = new AssociativeArray();
                total = 0
                for all stripe H in stripes [H1, H2, H3, ...] do
                        total += Sum(H_f, H); // Element-wise sum
                for all term u in H_f do
                        H_f(u) = H_f(u) / total;
                Emit(term w, stripe H_f);
        }
}
```

Java code – STRIPE approach Reducer 1

```
@Override
protected void reduce(Text key, Iterable<MapWritable> values, Context context)
        throws IOException, InterruptedException {
    sumOfAllStripes.clear();

    for (MapWritable value : values) {
        addStripeToFinalStripe(value);
    }
    int totalCount = 0;

    for (Entry<Writable, Writable> entry : sumOfAllStripes.entrySet()) {
        totalCount += ((IntWritable) entry.getValue()).get();
    }
    DecimalFormat decimalFormat = new DecimalFormat("0.000");
    StringBuilder stringBuilder = new StringBuilder();

    for (Entry<Writable, Writable> entry : sumOfAllStripes.entrySet()) {
        if (stringBuilder.length() > 0)
            stringBuilder.append(",");
        stringBuilder.append(" (").append(entry.getKey().toString()).append(",")
                .append(decimalFormat.format(
                        Integer.parseInt(entry.getValue().toString()) / (double) totalCount))
                .append(")");
    }
    context.write(key, new Text("[" + stringBuilder.toString() + " ]"));
}
```

Java code – STRIPE approach Reducer 2

```
private void addStripeToFinalStripe(MapWritable value) {

    Set<Writable> allKeys = value.keySet();
    for (Writable key : allKeys) {
        IntWritable keyCountInCurrentStripe = (IntWritable) value.get(key);
        if (sumOfAllStripes.containsKey(key)) {
            IntWritable keyCountInResultantStripe = (IntWritable) sumOfAllStripes.get(key);
            keyCountInResultantStripe.set(keyCountInCurrentStripe.get() + keyCountInResultantStripe.get());
        } else {
            sumOfAllStripes.put(key, keyCountInCurrentStripe);
        }
    }

}
```

Result – STRIPE approach

12 [(34,0.364), (56,0.182), (79,0.091), (29,0.182), (18,0.091), (92,0.091)]

29 [(12,0.308), (34,0.308), (56,0.154), (79,0.077), (18,0.077), (92,0.077)]

34 [(56,0.250), (12,0.250), (79,0.083), (29,0.250), (18,0.083), (92,0.083)]

56 [(12,0.300), (34,0.300), (29,0.200), (18,0.100), (92,0.100)]

79 [(56,0.200), (12,0.200), (34,0.200), (29,0.200), (18,0.200)]

92 [(34,0.250), (12,0.250), (56,0.083), (79,0.083), (29,0.250), (18,0.083)]

Pseudo code – HYBRID approach Mapper

```
class Mapper {
        AssociativeArray H;
        method Initialize() {
                H = new AssociativeArray();
        }
        method Map(docid a; doc d) {
                for all term w in doc d do
                        for all term u in Neighbors(w) do
                                if(H[pair(w, u)] == null)
                                        H.add(pair(w, u), 0);
                                H[pair(w, u)].count++;
        }
        method Close() {
                Foreach(element e in H)
                        Emit(e.Key, e.Value);
        }
}
```

Java code – HYBRID approach Mapper

```
@Override
protected void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {
    String[] allTerms = value.toString().split("\\s+");

    for (int i = 0; i < allTerms.length; i++) {
        String term = allTerms[i];
        for (int j = i + 1; j < allTerms.length; j++) {
            if (term.equals(allTerms[j]))
                break;

            Pair keyTerm = new Pair(term, allTerms[j]);

            if (!map.containsKey(keyTerm)) {
                map.put(keyTerm, 1);
            } else {
                map.put(keyTerm, map.get(keyTerm) + 1);
            }
        }
    }
}
```

Pseudo code – HYBRID approach Reducer

```
class Reducer
        method Initialize
                H1 = new associative array

        method Reduce (Pair p, Count [c1;c2;c3; :::])
                for all count c in Count [c1;c2;c3; :::] do
                        H{Pair p} += c;

        method Close
                H_f = new associative array
                term cW

                for all (pair(w, u), integer i) in H do
                        if cW != w and H_f.size() != 0 then
                                total = 0;
                                for all pair(w, u) in H_f do
                                        total += H_f{u}
                                for all pair(w, u) in H_f do
                                        H_f{u} = H_f{u} / total
                                Emit(term currentW, stripe H_f)
                                clear(H_f)
                        else
                                H_f{u} = H_f{u} + i
                        cW = w
```

Java code – HYBRID approach Reducer 1

```
@Override
protected void reduce(Pair key, Iterable<IntWritable> values, Context context)
        throws IOException, InterruptedException {
    total = sumAllValues(values);
    String kS = key.getKey().toString();
    String vS = key.getValue().toString();
    Pair p = new Pair(kS, vS);
    map.put(p, total);
}

@Override
protected void cleanup(Context context) throws IOException, InterruptedException {
    Map<String, Integer> mp = new HashMap<>();

    String wPrev = "";
    total = 0;
    for (Map.Entry<Pair, Integer> entry : map.entrySet()) {
        String w = entry.getKey().getKey().toString();

        if (!w.equals(wPrev) && mp.size() != 0) {

            StringBuilder stringBuilder = getStripe(mp);
            context.write(new Text(wPrev), new Text("[" + stringBuilder.toString() + " ]"));

            mp.clear();
            total = 0;
        }
        accumulate(mp, entry);
        wPrev = w;
    }
    StringBuilder stringBuilder = getStripe(mp);
    context.write(new Text(wPrev), new Text("[" + stringBuilder.toString() + " ]"));
}
```

Java code – STRIPE approach Reducer 2

```
private void accumulate(Map<String, Integer> mp, Map.Entry<Pair, Integer> entry) {
    String u = entry.getKey().getValue().toString();
    mp.put(u, entry.getValue());
    total += entry.getValue();
}

private StringBuilder getStripe(Map<String, Integer> mp) {
    DecimalFormat decimalFormat = new DecimalFormat("0.000");
    StringBuilder stringBuilder = new StringBuilder();

    for (Map.Entry<String, Integer> e : mp.entrySet()) {
        if (stringBuilder.length() > 0)
            stringBuilder.append(",");
        stringBuilder.append(" (").append(e.getKey()).append(",")
                .append(decimalFormat.format(e.getValue() / (double) total)).append(")");
    }
    return stringBuilder;
}

private int sumAllValues(Iterable<IntWritable> values) {
    int sum = 0;
    for (IntWritable v : values) {
        sum += v.get();
    }
    return sum;
}
```

Result – HYBRID approach

12 [(34,0.364), (56,0.182), (79,0.091), (18,0.091), (29,0.182), (92,0.091)]

29 [(12,0.308), (34,0.308), (56,0.154), (79,0.077), (18,0.077), (92,0.077)]

34 [(12,0.250), (56,0.250), (79,0.083), (18,0.083), (29,0.250), (92,0.083)]

56 [(12,0.300), (34,0.300), (18,0.100), (29,0.200), (92,0.100)]

79 [(12,0.200), (34,0.200), (56,0.200), (18,0.200), (29,0.200)]

92 [(12,0.250), (34,0.250), (56,0.083), (79,0.083), (18,0.083), (29,0.250)]

Comparison

Parameters	Pair Approach	Stripe Approach	Hybrid Approach
Map input records	2	2	2
Map output records	40	6	34
Map output bytes	394	354	340
Map output materialized bytes	480	372	414
Input split bytes	121	121	121
Combine input records	0	0	0
Combine output records	0	0	0
Reduce input groups	40	6	34
Reduce shuffle bytes	480	372	414
Reduce input records	40	6	34
Reduce output records	34	6	6
Spilled Records	80	12	68
Shuffled Maps	1	1	1
Failed Shuffles	0	0	0
Merged Map outputs	1	1	1
GC time elapsed (ms)	110	145	113
CPU time spent (ms)	0	0	0
Physical memory (bytes) snapshot	0	0	0
Virtual memory (bytes) snapshot	0	0	0
Total committed heap usage (bytes)	345505792	407371776	408420352