Automatically Leveraging MapReduce Frameworks for Data-Intensive Applications

Maaz Bin Safeer Ahmad
(University of Washington)

Alvin Cheung
(University of Washington)
Why translate sequential code to MapReduce?
Optimizing Existing Applications

Sequential Code

Java
Optimizing Existing Applications

Sequential Code

Java

{Sequential Code}
Optimizing Existing Applications

Sequential Code
Optimizing Existing Applications

Sequential Code

Java

Flink  Spark  hadoop
Optimizing Existing Applications

Sequential Code

{MapReduce Code}

Framework API

Java

Flink  Spark  hadoop
Option 1: Manual Re-write

- Recurring
- Error-prone
- Tedious
- Requires Expertise

Java → Flink → Spark → Hadoop
Option 2: Build a Compiler
Why is the sequential to MapReduce re-write difficult to automate?
Syntax Directed Translation

- Traditionally compilers use pattern-matching rules to do code transformations.

```java
for (int i=0; i < $in.size(); ++i)
{
    if ($in.get(i) > $c)
        $out.add($in.get(i));
}
```
Syntax Directed Translation

- Traditionally compilers use pattern-matching rules to do code transformations.

```java
for (int i=0; i < $in.size(); ++i) {
    if ($in.get(i) > $c) {
        $out.add($in.get(i));
    }
$out.union($in.filter(e -> e > $c));
```
Syntax Directed Translation

• Traditionally compilers use pattern-matching rules to do code transformations.

```java
for (int i=0; i<N; i++) {
    HashMap<String, Double> contrib = new HashMap<>();
    for (Map.Entry<String, Double> r : ranks.entrySet()) {
        List<String> urls = grouped_links.get(r.getKey());
        if (urls != null) {
            int size = urls.size();
            urls.forEach(dst -> {
                if (!contrib.containsKey(dst))
                    contrib.put(dst, 0.0);
                contrib.put(dst, contrib.get(dst) +
                            (r.getValue() / size));
            });}
    for (String dst : contrib.keySet())
        ranks.put(dst, contrib.get(dst) * 0.85 + 0.15);
}
```
Syntax Directed Translation

- Traditionally compilers use pattern-matching rules to do code transformations.

```java
for (int i=0; i<N; i++) {
    HashMap<String,Double> contrib = new HashMap<>();
    for (Map.Entry<String,Double> r : ranks.entrySet()) {
        List<String> urls = grouped_links.get(r.getKey());
        if(urls != null) {
            int size = urls.size();
            urls.forEach(dst -> {
                if (!contrib.containsKey(dst))
                    contrib.put(dst, 0.0);
                contrib.put(dst, contrib.get(dst) + (r.getValue() / size));
            });
    }
    for (String dst : contrib.keySet())
        ranks.put(dst, contrib.get(dst) * 0.85 + 0.15);
}
```
A Synthesis Based Approach

```java
int sum = 0;
for (Integer val : data) {
    sum += val * val;
}
```

Re-write rules

```java
int sum = 0;
sum = data.map(val -> val*val)
  .reduce((v1,v2) -> v1+v2);
```
A Synthesis Based Approach

\[ \lambda_m(v) \rightarrow v \times v \]
\[ \lambda_r(v_1, v_2) \rightarrow v_1 + v_2 \]
\[ \text{sum} = \text{reduce}(\text{map}(\text{data}, \lambda_m), \lambda_r); \]

```
int sum = 0;
for (Integer val : data) {
    sum += val * val;
}
```
A Synthesis Based Approach

\[ \lambda_m(v) \rightarrow v \times v \]
\[ \lambda_r(v_1, v_2) \rightarrow v_1 + v_2 \]
\[ \text{sum} = \text{reduce} (\text{map} (\text{data}, \lambda_m), \lambda_r) ; \]

Java

```java
int sum = 0;
for (Integer val : data) {
    sum += val * val;
}
```

Re-write rules

```java
int sum = 0;
sum = data.map(val -> val*val)
    .reduce((v1,v2) -> v1+v2);
```

Codegen
A Synthesis Based Approach

\[ \lambda_m(v) \rightarrow v \ast v \]
\[ \lambda_r(v_1, v_2) \rightarrow v_1 + v_2 \]

\[ \text{sum} = \text{reduce}(\text{map}(\text{data}, \lambda_m, \lambda_r)); \]

```
int sum = 0;
for (Integer val : data) {
    sum += val * val;
}
```

```
int sum = 0;
sum = data.map(val -> val*val)
   .reduce((v1,v2) -> v1+v2);
```
for (int i=0; i<N; i++) {
    HashMap<String,Double> contrib = new HashMap<>();
    for (Map.Entry<String,Double> r : ranks.entrySet()) {
        List<String> urls = grouped_links.get(r.getKey());
        if(urls != null) {
            int size = urls.size();
            urls.forEach(dst -> {
                if (!contrib.containsKey(dst))
                    contrib.put(dst, 0.0);
                contrib.put(dst, contrib.get(dst) +
                    (r.getValue() / size));
            });
        }
    }
    for (String dst : contrib.keySet())
        ranks.put(dst, contrib.get(dst) * 0.85 + 0.15);
}
Program Synthesis

for (int i=0; i<N; i++) {
    HashMap<String,Double> contrib = new HashMap<>();
    for (Map.Entry<String,Double> r : ranks.entrySet()) {
        List<String> urls = grouped_links.get(r.getKey());
        if(urls != null) {
            int size = urls.size();
            urls.forEach(dst -> {
                if (!contrib.containsKey(dst))
                    contrib.put(dst, 0.0);
                contrib.put(dst, contrib.get(dst) +
                    (r.getValue() / size));
            });}
        for (String dst : contrib.keySet())
            ranks.put(dst, contrib.get(dst) * 0.85 + 0.15);
    }
}

Space of programs constructed using map and reduce operators
Program Synthesis

```java
for (int i=0; i<N; i++) {
    HashMap<String, Double> contrib = new HashMap<>();
    for (Map.Entry<String, Double> r : ranks.entrySet()) {
        List<String> urls = grouped_links.get(r.getKey());
        if (urls != null) {
            int size = urls.size();
            urls.forEach(dst -> {
                if (!contrib.containsKey(dst))
                    contrib.put(dst, 0.0);
                contrib.put(dst, contrib.get(dst) +
                    (r.getValue() / size));
            });}
    for (String dst : contrib.keySet())
        ranks.put(dst, contrib.get(dst) * 0.85 + 0.15);
}
```

Space of programs constructed using
**map** and **reduce** operators
for (int i=0; i<N; i++) {
    HashMap<String,Double> contrib = new HashMap<>();
    for (Map.Entry<String,Double> r : ranks.entrySet()) {
        List<String> urls = grouped_links.get(r.getKey());
        if(urls != null) {
            int size = urls.size();
            urls.forEach(dst -> {
                if (!contrib.containsKey(dst))
                    contrib.put(dst, 0.0);
                contrib.put(dst, contrib.get(dst) +
                    (r.getValue() / size));
            });}
    for (String dst : contrib.keySet())
        ranks.put(dst, contrib.get(dst) * 0.85 + 0.15);
}
for (int i=0; i<N; i++) {
    HashMap<String,Double> contrib = new HashMap<>();
    for (Map.Entry<String,Double> r : ranks.entrySet()) {
        List<String> urls = grouped_links.get(r.getKey());
        if(urls != null) {
            int size = urls.size();
            urls.forEach(dst -> {
                if (!contrib.containsKey(dst))
                    contrib.put(dst, 0.0);
                contrib.put(dst, contrib.get(dst) +
                        (r.getValue() / size));
            });
        }
    }
    for (String dst : contrib.keySet())
        ranks.put(dst, contrib.get(dst) * 0.85 + 0.15);
}
Making Search Manageable

- Design a concise API to express specification
- Use program analysis to specialize search
  - Ex: Only use specific operators
- Use incremental search
- Cost-based pruning
Introducing Casper

A compiler that automatically re-targets sequential Java applications to MapReduce frameworks.

**Input**
Un-annotated sequential Java application source code.

**Output**
An optimized version of the application that uses either Spark, Flink or Hadoop.
System Evaluation: Benchmarks

We used to optimize **55 benchmarks** collected from various sources.

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phoenix</td>
<td>Classical MapReduce problems</td>
</tr>
<tr>
<td>Fiji</td>
<td>Four open-sourced plugins implementing image processing algorithms</td>
</tr>
<tr>
<td>Bigλ</td>
<td>Big-Data analytics kernels</td>
</tr>
<tr>
<td>Ariths</td>
<td>Mathematical functions such as <em>sum</em>, <em>count</em>, <em>delta</em> etc.</td>
</tr>
<tr>
<td>Stats</td>
<td>Statistical functions such as <em>mean</em>, <em>variance</em>, <em>standard error</em> etc.</td>
</tr>
<tr>
<td>TPC-H</td>
<td>Java implementations for q1, q6, q15 and q17</td>
</tr>
<tr>
<td>Iterative</td>
<td>Page-rank and Logistic Regression based classification</td>
</tr>
</tbody>
</table>
Feasibility Analysis

Casper successfully translated 82 of the 101 identified code fragments across all benchmarks.

Causes of failures

• 3 caused by references to external library calls which were not currently supported
• 7 benchmarks could not be expressed in our intermediate language
• 9 benchmarks timed out (required more than 90 minutes)
Performance Analysis (Spark)

- Rule-based compiler (MOLD)
- Casper

<table>
<thead>
<tr>
<th>Task</th>
<th>Speedup over original</th>
</tr>
</thead>
<tbody>
<tr>
<td>String Match</td>
<td>8x</td>
</tr>
<tr>
<td>Word Count</td>
<td>13x</td>
</tr>
<tr>
<td>Linear Regression</td>
<td>35x</td>
</tr>
<tr>
<td>3D Histogram</td>
<td>5x</td>
</tr>
<tr>
<td>YelpKids</td>
<td>18x</td>
</tr>
<tr>
<td>PageCount</td>
<td>24x</td>
</tr>
<tr>
<td>Covariance</td>
<td>15x</td>
</tr>
<tr>
<td>Hadamard Product</td>
<td>17x</td>
</tr>
<tr>
<td>Database Select</td>
<td>22x</td>
</tr>
<tr>
<td>Anscombe Transform</td>
<td>6x</td>
</tr>
</tbody>
</table>

75GB data on a 10 node cluster (8 cores, 30gb ram)
Performance Analysis (Spark)

- Rule-based compiler (MOLD)
- Casper
- Hired Developers

75GB data on a 10 node cluster (8 cores, 30gb ram)
How long does Casper take?

Mean compilation time for one benchmark was **11.4 minutes**.

Median compilation time for one benchmark was **2.1 minutes**.
How long does Casper take?

Mean compilation time for one benchmark was **11.4 minutes**.

Median compilation time for one benchmark was **2.1 minutes**.

No managerial overhead!

---

Maaz Safeer

Hi

Can you update on your progress?

Maaz Safeer

Please update on progress, thank you.

---

Another point that I'm considering is time. I guess that it's difficult for me deliver the 40 algorithms in 4 days.
Take-aways

• Casper can automatically translate a wide array of sequential applications to MapReduce.

• With average speedups of 15.6x, Casper is competitive with hand written translations.
Take-aways

• Casper can automatically translate a wide array of sequential applications to MapReduce.

• With average speedups of 15.6x, Casper is competitive with hand written translations.

[Website Link]

casper.uwplse.org

Paper | Online Demo | Source Code