Cybertron: Pushing the Limit on I/O Reduction in Data-Parallel Programs

Tian XIAO, Zhenyu GUO, Hucheng ZHOU, Jiaxing ZHANG, Xu ZHAO, Chencheng YE, Xi WANG, Wei LIN, Wenguang CHEN, Lidong ZHOU

Tsinghua University, Microsoft Research, Huazhong University of Science and Technology, MIT CSAIL
Problem: reduce network I/O
Approach I: data compression

Examples

- DEFLATE (gzip)
- PPMd (7-zip)
Approach II: unused data elimination

Example:

• Unused table column
• Unused table row (filter before send)
Push further reduction of network IO?

Is it always necessary for \( \mathbf{V} == \mathbf{V}' \) ?

⇒

What if \( \mathbf{V} != \mathbf{V}' \) AND size(\( \mathbf{V}' \)) < size(\( \mathbf{V} \))?
Cybertron: execution-equivalent data encoding

Correctness: $f(v) == f(v')$
Apply in map-reduce programs

• Observation
  • Dominant type of $\mathbf{V}$ is string
  • Dominant computation of $f$ is string operations
A Production Example

\[ f(v) = \]

```c
string flags = null;
string ip = null;
foreach (string token in v.Split(";")) {
    if (token.StartsWith("flags="))
        flags = token.Substring(8);
    if (token.StartsWith("ip="))
        ip = token.Substring(3);
}
```

\[ f(v) == f(v') \quad \{\text{flags = "4C0", ip = "192.168.0.1"}\} \]

**WHEN**  \( v = \)  
**AND**  \( v' = \)  

AND we easily get  \( \text{size}(v') < \text{size}(v) \)
Question

Given $f(v)$ and $v$, how to get $v'$ so that $f(v) = f(v')$ and $\text{size}(v') \leq \text{size}(v)$?

```
ip=192.168.0.1;scheduler=611;flags=0x4C0;action=b123
```

```
ip=192.168.0.1;$$$$$$$$$$$$$$$$;flags=$$4C0;$$$$$$$$$$$$$$$$
```
Intuition

\[ f(v) = \]

```csharp
string flags = null;
string ip = null;
foreach (string token in v.Split(";")) {
    if (token.StartsWith("flags="))
        flags = token.Substring(8);
    if (token.StartsWith("ip="))
        ip = token.Substring(3);
}
```

\( f(v) = f(v') \) holds when:

1. control flow remains the same
2. final data values (e.g., flags, ip) remain the same
   (intermediate data value can be different)
string flags = null;
string ip = null;

foreach (string token in V.Split(";")) {
    if (token.StartsWith("flags="))
        flags = token.Substring(8);
    if (token.StartsWith("ip="))
        ip = token.Substring(3);
}

V = ip=192.168.0.1; scheduler=611; flags=0x4C0; action=b123
From Range Constraints to V’

- Individual range constraint is easy to be solved
- Challenges
  - overlapping, e.g., \((1, 2)\)
  - conflict, e.g., \((1, 2, 4)\)

### Z3

Solver?

Too expensive!
Goal: transform them into non-overlapping and non-conflicting range constraints.

- Overlapping $\leftarrow$ decomposition
  - $(\text{offset}, \text{len}, \text{c}) := (\text{offset}, \text{len} – \text{len}1, \text{c}), (\text{offset}, \text{len}1, \text{c})$
  - Easy for \text{Any/Value/Const} (\text{2 3}), difficult for \text{Pred}(\text{1 4})

- Confliction $\leftarrow$ total ordering of constraints
  - \text{Any} $\leq$ \text{Pred} $\leq$ \text{Value} $\leq$ \text{Const} (\text{2 3} are selected)

  - Challenges around \text{Pred} constraint, solved by introducing \text{Special}
    - See paper for details

- Result: \text{Special, Value, Const} on non-overlapping ranges (\text{2 5 3})
  - Efficient encoder and decoder provided
Back to the example

```
string flags = null;
string ip = null;
foreach (string token in fields[7].Split(",")) {
    if (token.StartsWith("flags="))
        flags = token.Substring(8);
    if (token.StartsWith("ip="))
        ip = token.Substring(3);
}
```

```
ip=192.168.0.1;scheduler=611;flags=0x4C0;action=b123
```

![Diagram of the example with ENCODE and DECODE processes]
Cybertron Workflow

$f'(\text{Encoder}) \rightarrow \text{network} \rightarrow \text{Decoder} \rightarrow f(\text{Decoder})$
Data reduction and contribution from different constraint types
Network throughput impact with different network bandwidths
Conclusion

• Cybertron: execution-equivalent data encoding to reduce network I/O
  • Combine both static and dynamic methods
  • Trade computation for network I/O
  • Orthogonal to traditional data compression methods (e.g., DEFLATE)
• Applied in map-reduce programs and look for more scenarios where network I/O matters
Thanks!