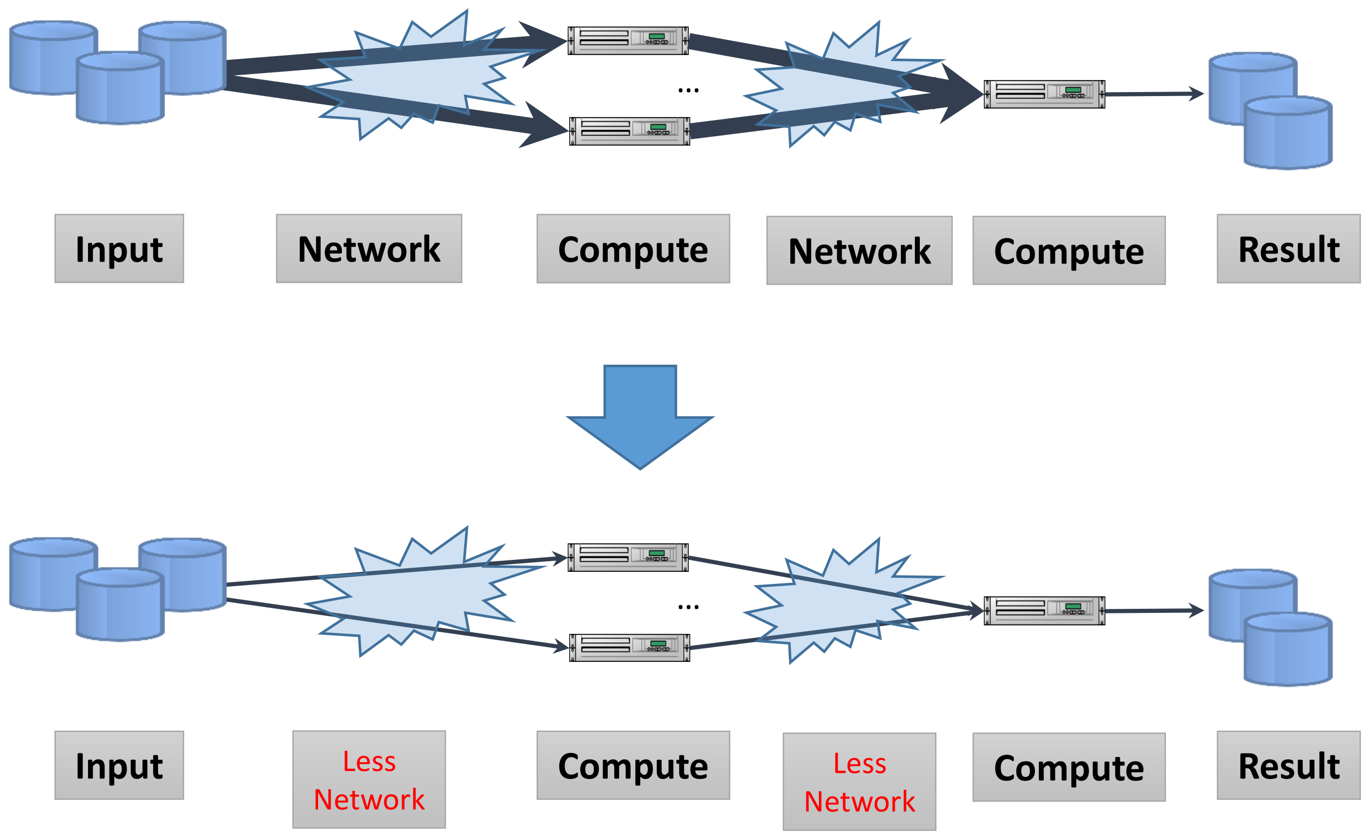
Cybertron: Pushing the Limit on I/O Reduction in Data-Parallel Programs

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

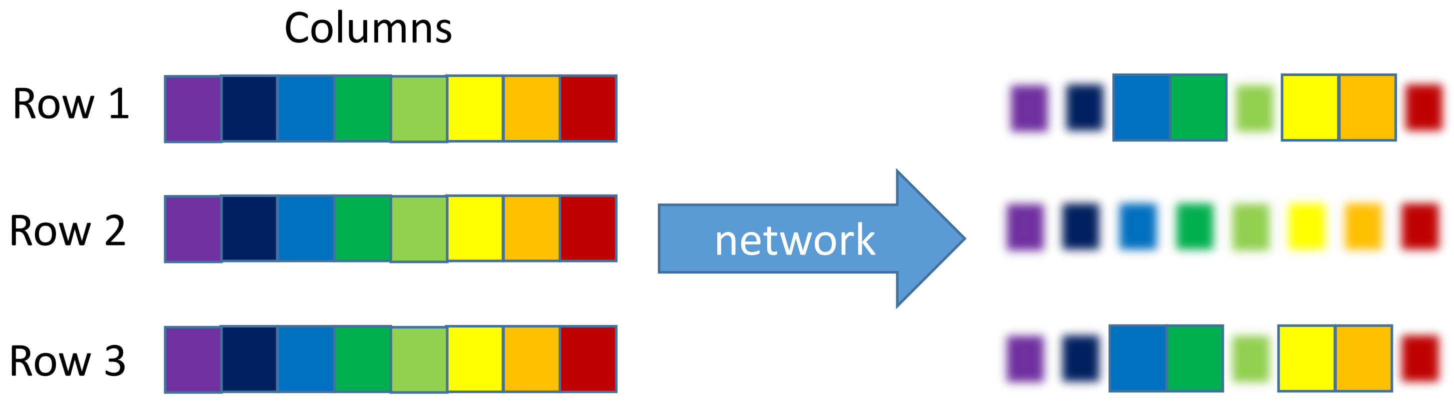
Examples

- DEFLATE (gzip)
- PPMd (7-zip)

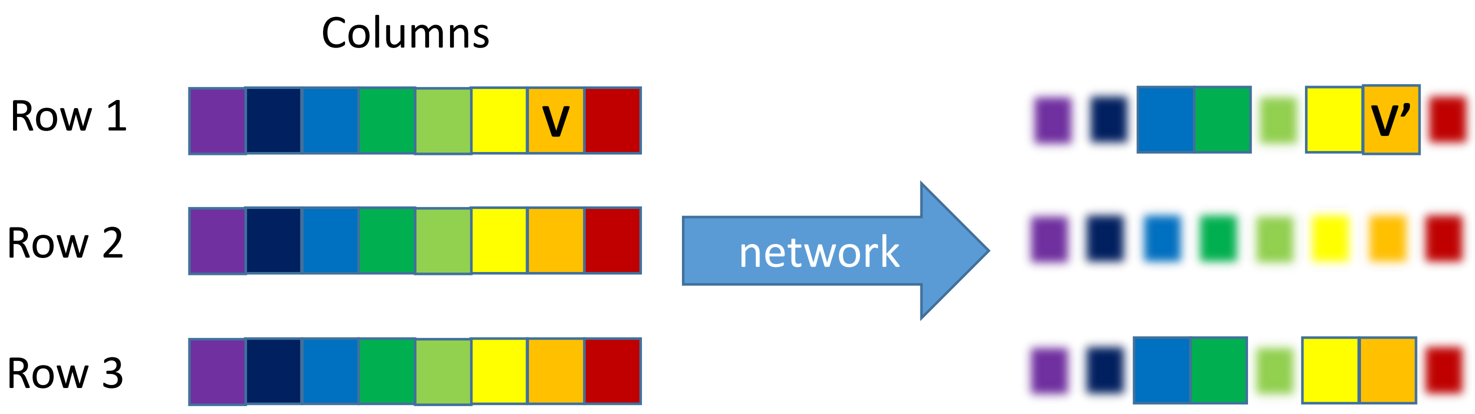
Approach II: unused data elimination

Example:

- Unused table column
- Unused table row (filter before send)



Push further reduction of network IO?

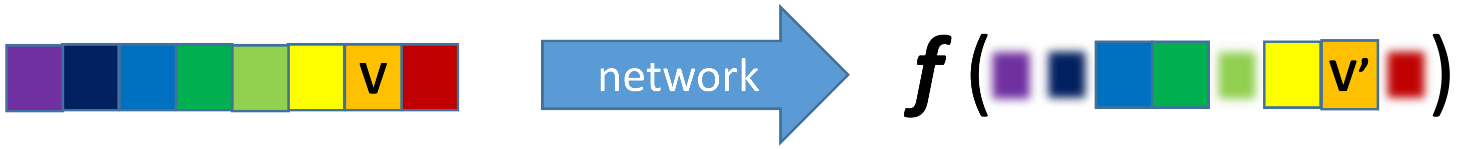


Is it always necessary for **V == V’** ?

➔

What if **V != V’** AND size(**V’**) < size(**V**) ?

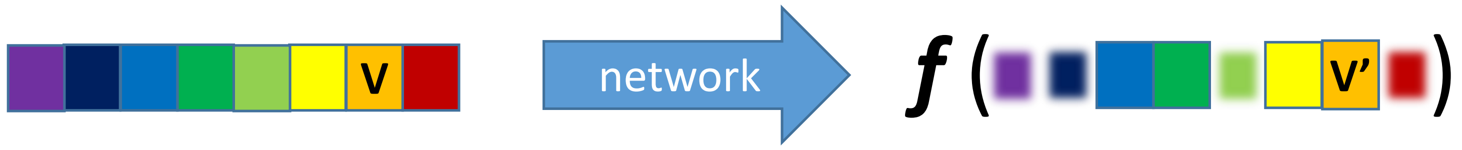
Cybertron: execution-equivalent data encoding



Correctness: $f(v) == f(v')$

Apply in map-reduce programs

- Observation
  - Dominant type of **V** is string
  - Dominant computation of ***f*** is string operations



A Production Example

f(v) =

```
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') {flags = “4C0”, ip = “192.168.0.1”}

WHEN v =

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

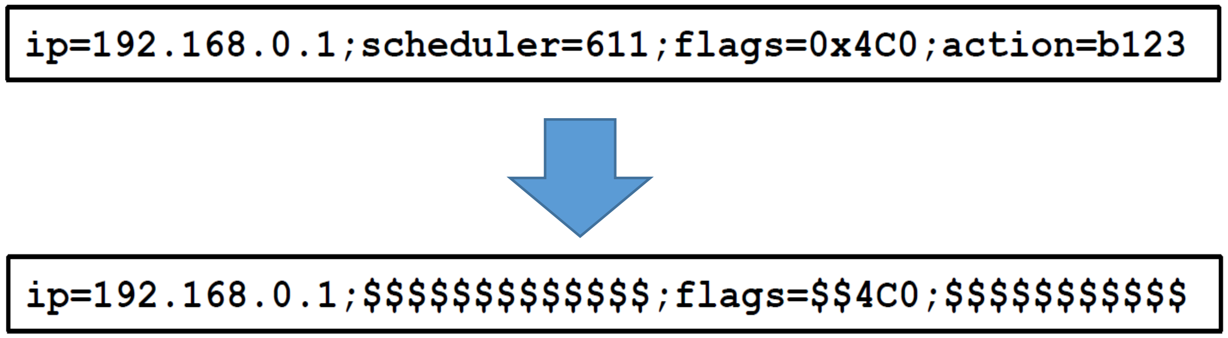
AND v' =

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

AND we easily get size(v') < size(v)

Question

Given *f*(v) and v, how to get v’ so that *f*(v) == *f*(v’) and size(v’) <= size(v)?



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

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

Intuition

f(v) =

```
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);
}
```

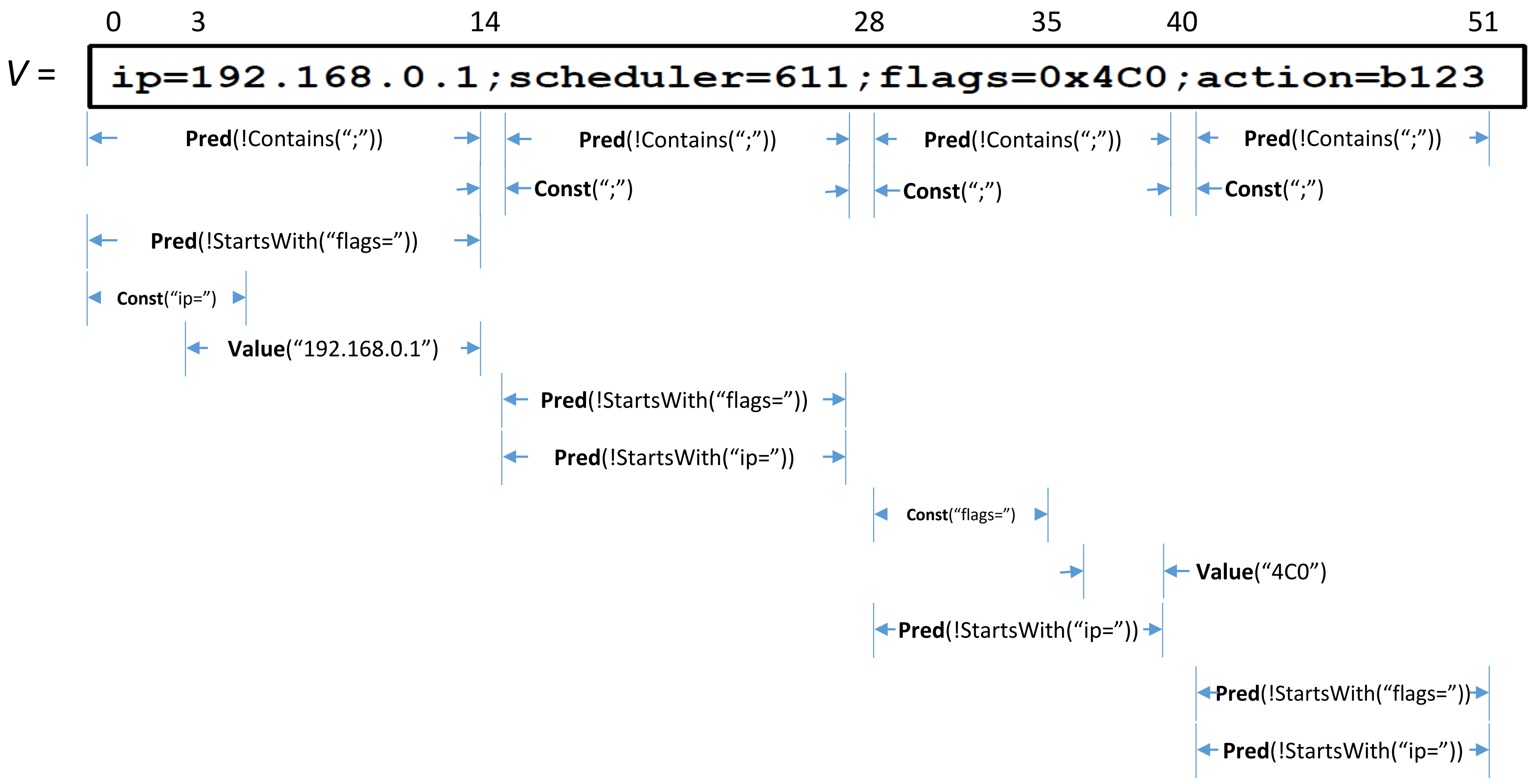
f(v) == f(v’) holds when:

(1) control flow remains the same

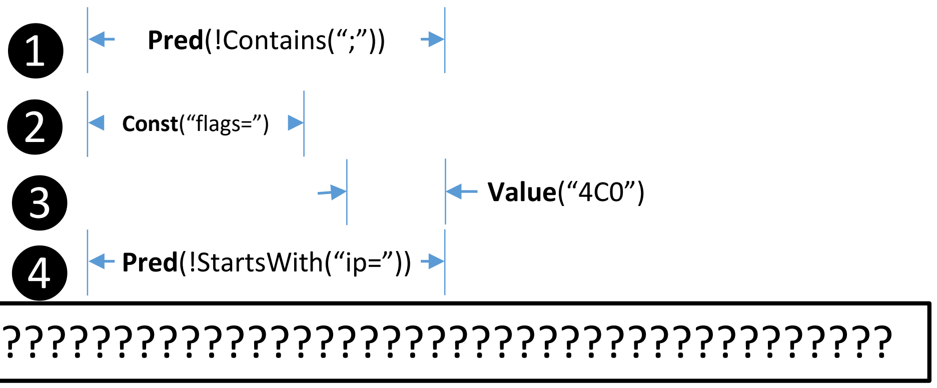
(2) final data values (e.g., flags, ip) remain the same
    (intermediate data value can be different)

From *V* To Range Constraints

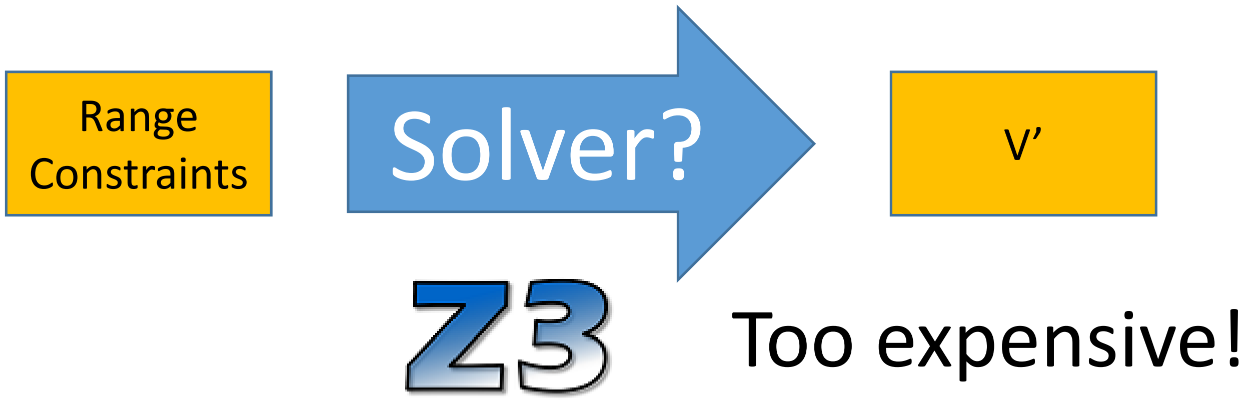
```
string flags = null;
string ip = null;
foreach (string token in           V.Split(";")) {
  if (token.StartsWith("flags="))
    flags = token.Substring(8);
  if (token.StartsWith("ip="))
    ip = token.Substring(3);
}
```



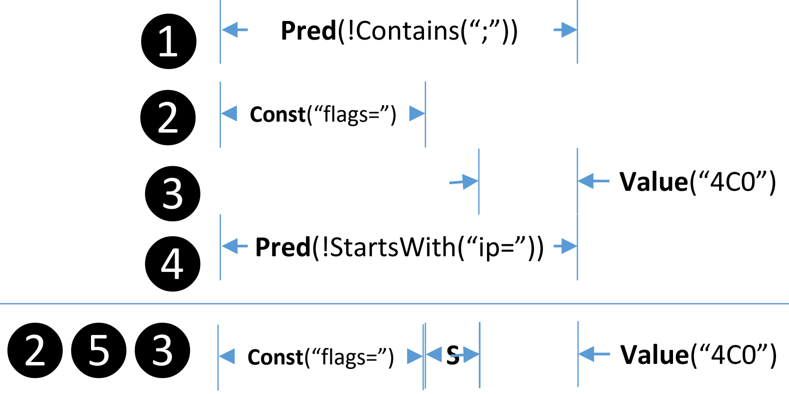
From Range Constraints to V’



- Individual range constraint is easy to be solved
- Challenges
  - overlapping, e.g., (❶, ❷)
  - conflict, e.g., (❶,❷, ❹)



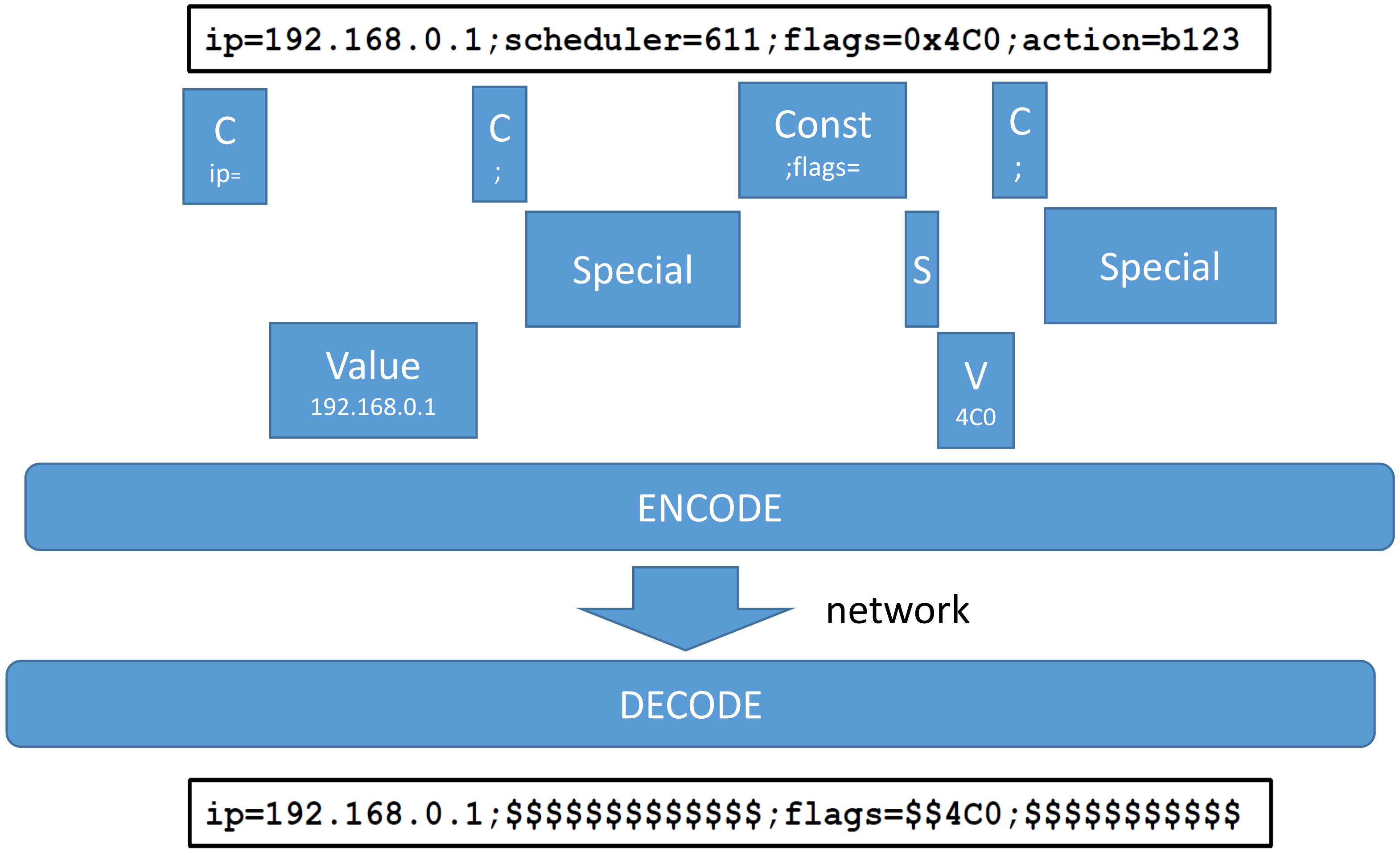
Constraints Concretization


❶ Pred(!Contains(";"))
❷ Const("flags=")
❸ Value("4C0")
❹ Pred(!StartsWith("ip="))
❷❺❸ Const("flags=") S Value("4C0")

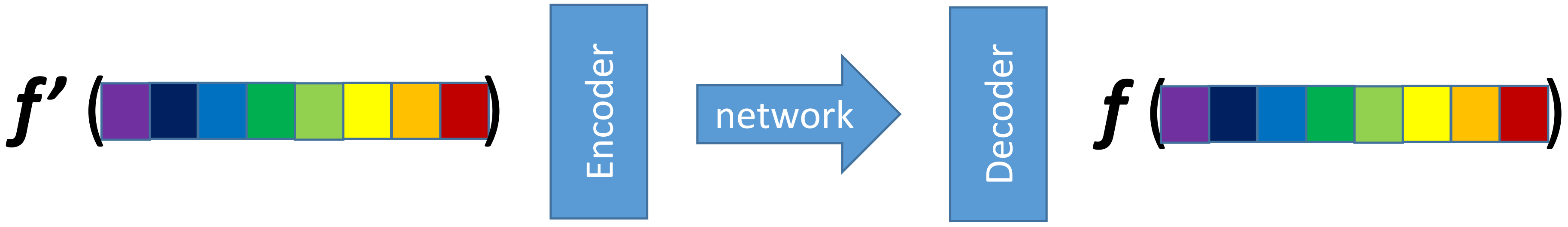
- Goal: transform them into non-overlapping and non-conflicting range constraints.
  - Overlapping 🡐 decomposition
    - (offset, len, c) := (offset, len – len1, c), (offset, len1, c)
    - Easy for *Any/Value/Const* (❷❸), difficult for *Pred*(❶❹)
  - Confliction 🡐 total ordering of constraints
    - *Any <= Pred <= Value <= Const (*❷❸ *are selected)*
  - Challenges around *Pred* constraint, solved by introducing *Special*
    - See paper for details
- Result: *Special*, *Value*, *Const* on non-overlapping ranges (❷❺❸)
  - Efficient encoder and decoder provided

Back to the example

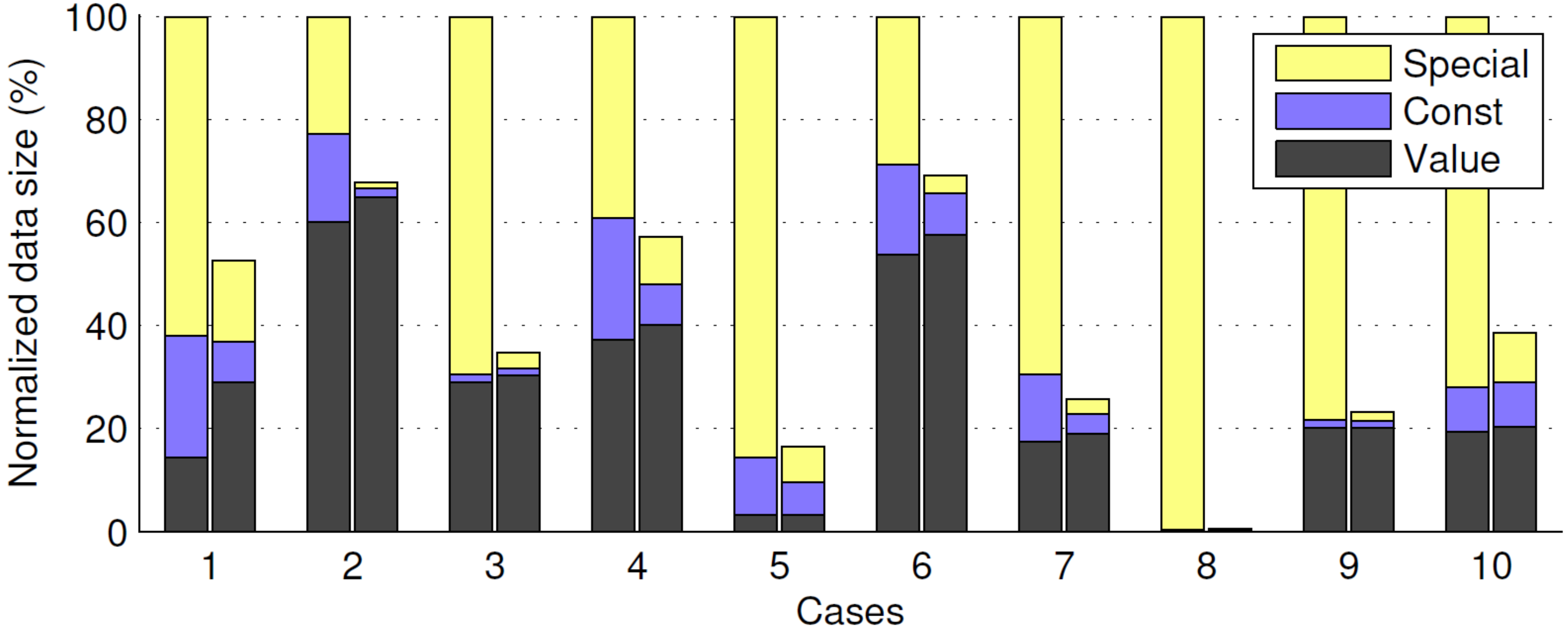
```
string flags = null;
string ip = null;
foreach (string token in fields[7].Split(";")) {
  if (token.StartsWith("flags="))
    flags = token.Substring(8);
  if (token.StartsWith("ip="))
    ip = token.Substring(3);
}
```


ip=192.168.0.1;scheduler=611;flags=0x4C0;action=b123
C
ip=
C
;
Const
;flags=
C
;
Special
S
Special
Value
192.168.0.1
V
4C0
ENCODE
network
DECODE
ip=192.168.0.1;$$$$$$$$$$$$$;flags=$$4C0;$$$$$$$$$$$

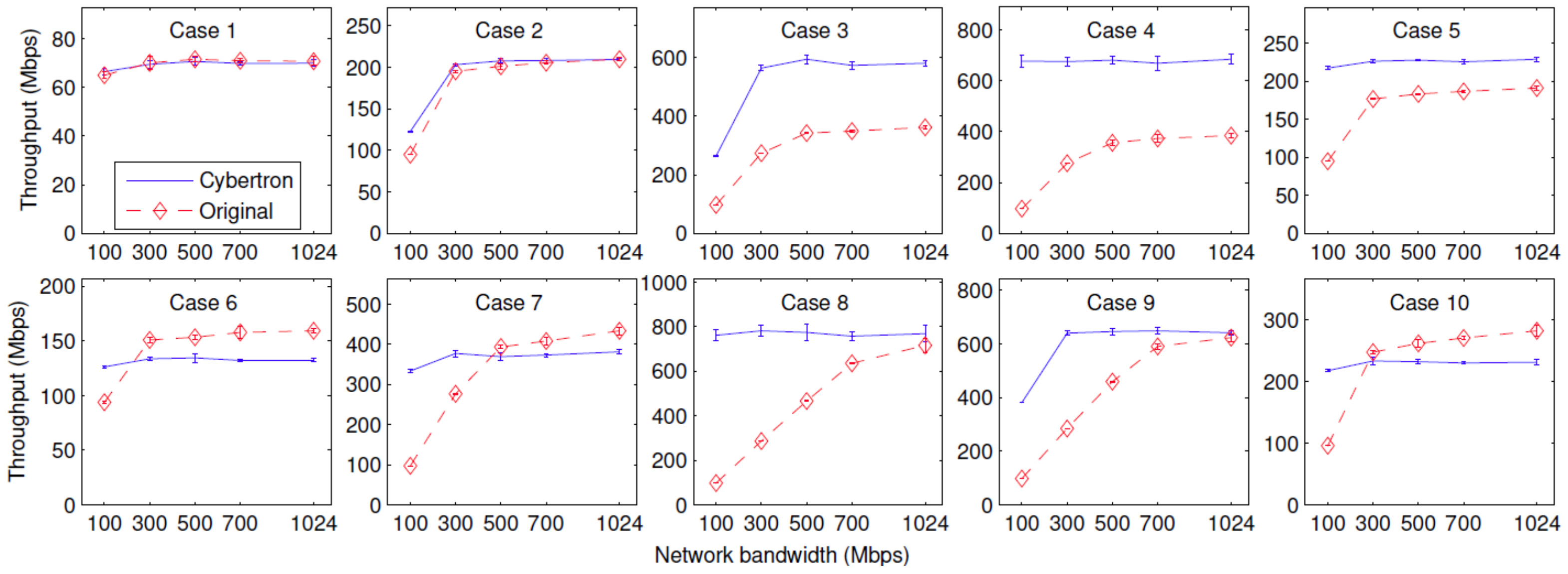
Cybertron Workflow



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!