KolmoLD: Data Modelling for the Modern Internet*

Dmitry Borzov, Huawei Canada
Tim Tingqiu Yuan, Huawei
Mikhail Ignatovich, Huawei Canada
Jian Li, Futurewei

*work performed before May 2019
Challenges: Peak Traffic Composition

Content: sizable, fan-ended, static
- Streaming Services (Netflix, Hulu, YouTube, Spotify)
- Software distribution
- File storage services
- Everything else (Instant Messaging, VoIP, Social Media)

Most internet traffic will flow over content delivery networks (CDNs)

<table>
<thead>
<tr>
<th>Year</th>
<th>CDN</th>
<th>Exabytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016</td>
<td></td>
<td>38.4</td>
</tr>
<tr>
<td>2017</td>
<td></td>
<td>54</td>
</tr>
<tr>
<td>2018</td>
<td></td>
<td>73</td>
</tr>
<tr>
<td>2019</td>
<td></td>
<td>97.8</td>
</tr>
<tr>
<td>2020</td>
<td></td>
<td>127</td>
</tr>
<tr>
<td>2021</td>
<td></td>
<td>166.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Non-CDN</th>
<th>Exabytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>34.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>46.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>53.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>62.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>69.5</td>
</tr>
</tbody>
</table>

Technologies to define the revolution of the internet

**IPFS**
- Founded in 2014
- Content-addressable network protocol based on cryptohash naming scheme
- Founding company is a YCombinator graduate with backing of high profile SV investors

**Dat Project**
- Founded in 2016
- Content-addressable network protocol based on cryptohash naming scheme
- Open source project
- P2P project
- YCombinator graduate

**ChunkStream**
- Video codec
- Based on a 2014 MIT research paper
- Based on the cryptohash naming scheme

**WebAssembly**
- Browser-targeted Runtime
- Implemented and supported by all major browsers, an IETF standard
Our Proposal: A data model for interoperable protocols

Content addressing through hashes has become a widely-used means of connecting data in distributed systems, from the blockchains that run your favorite cryptocurrencies, to the commits that back your code, to the web’s content at large.

Yet, whilst all of these tools rely on some common primitives, their specific underlying data structures are not interoperable.
KolmoLD: Kolmogorov Linked Data

**Kolmogorov** Content-addressable
Users care about what they want, not who they get it from

**Data composability**
Send a way to reproduce data, not data itself

**Turing-complete programmability**
Sandboxed computability by treating data as code
## Content-addressable revolution in the making

<table>
<thead>
<tr>
<th>Content-addressable cryptohash naming</th>
<th>IPFS</th>
<th>Dat Project</th>
<th>ChunkStream</th>
<th>KolmoLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Composability</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Turing-complete Programmability</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
KolmoLD

Kolmogorov Content-addressable
Users care about what they want, not who they get it from

Data composability
send a way to reproduce data, not data itself

Turing-complete programmability
Sandboxed computability by treating data as code
Content-addressable networking:
Streaming an Olympics game (Before)
Content-addressable networking:
Streaming an Olympics game (After)
Host-addressable networks

GET /video1

kolmoblocks.org

Content-addressable networks

GET 42FBCC0D60EADA7

Announce interest

kolmoblocks.org
How to represent data content and DIKW in general?

Content-addressable networking

1) Consumers specify what they want, not who they need it from
2) The solution for content distribution
Kolmogorov complexity:
The shortest unambiguous algorithm, a computer program or code, that will output a given data string.

Shannon entropy = expected [ Kolmogorov complexity ]

Claude Shannon (1916-2001)
Andrei Kolmogorov (1903-1987)
Alan Turing (1912-1954)
Kolmogorov complexity metric of the given string of data is the size of the shortest algorithm that outputs that data

Kolmogorov complexity https://xkcd.com/1155/
As an algorithm

Calculate the first 100 terms of the series:

$$\sqrt{2} = \sum_{k=0}^{\infty} (-1)^{k+1} \frac{(2k - 3)!!}{(2k)!!} = 1 + \frac{1}{2} - \frac{1}{2 \cdot 4} + \frac{1 \cdot 3}{2 \cdot 4 \cdot 6} - \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6 \cdot 8} + \cdots.$$
Kolmogorov Content-addressable
Users care about what they want, not who they get it from

Data composability
Send a way to reproduce data, not data itself

Turing-complete programmability
Sandboxed computability by treating data as code
A new version of the OS is released

A 200Mb distro file is distributed across 20M devices across the world

A security patch is issued that flips a single byte at 0xfa24d4588

A patched 200Mb distro file is treated as completely new by the network

Dear phones,
The new version of the distro can be composed out of the original one by flipping the bit at 0xfa24d4588.

Love, the dev team
Data composability 2:
don’t send data, send a way to reproduce data

Algorithm
Concatenate images
A1EF919, FD62862
Data composability 3:
Reusing the encoding table

1Mb, book on potatoes  3Mb, book on cabbages  2Mb, book on tomatoes

\[ S_H(N, M) = 27^N + M \cdot \sum_i p(x_i) \log_2 p^{-1}(x_i) \]

\[ S = M \cdot \sum_i p'(x_i) \log_2 p^{-1}(x_i) \]
Data Composability 4: Quantifying the impact

Assumptions:

• Huffman encoding,

• Poisson distribution of the distinctly originating content blocks matching the statistical profile of the reported Internet traffic

• Entropy of the English language text

\[ R_H = \frac{M \cdot 8}{27^N + M \cdot G(N)} \]

\[ R_K(M) = \frac{M \cdot 8}{M \cdot H + \sqrt{M} \cdot \delta L} \]

\[ \frac{1}{R_H} - \frac{1}{R_KH} = \frac{27^3}{M} - \frac{1}{\sqrt{M} \delta L} \]
Data Composability 4: Quantifying the impact

\[ R_H = \frac{M \cdot 8}{27^N + M \cdot G(N)} \]

\[ R_K(M) = \frac{M \cdot 8}{M \cdot H + \sqrt{M} \cdot \delta L} \]
Data composability

1. Data blocks are identified by cryptographic hash functions
2. Global address space of data
3. Compose data blocks out of other data blocks

You don’t need to send it, just send a way to reproduce the data
KolmoLD

- Kolmogorov Content-addressable
  Users care about what they want, not who they get it from

- Data composability
  Send a way to reproduce data, not data itself

- Turing-complete programmability
  Sandboxed computability by treating data as code
Turing-complete programmability 1
Encode data as programs that output given data

ABABABBF

# SHA256: CC646
def render():
    return "ABABABBF"

# SHA256: CC646
def render():
    return "AB"*3+"F"
Turing-complete programmability 2
Referencing other kolmобlocks

# SHA256: F3025

def render():
    p = dep("CC646")
    return p+"-"+p[:1]
Turing-complete programmability 3
Reference other data chunks based on cryptohash naming

# lambdablock 77650
def f(huffman_tree, encoded_string):
    output = ''
    cur = 0
    while (cur < len(encoded_string)):
        node = huffman_tree
        while type(node) is list:
            bit = encoded_string[cur]
            cur += 1
            node = node[0] if bit else node[1]
        output += node
    return output

# kolmoblock
# target block: F3025
# dependancy blocks: 77650
def render():
    huffman_decode = eval(dep('77650'))
    huffman_tree = ['AB',
                     ['BA',
                      ['F', '-']
                     ]
    ]
    encoded = 0b00011011110101010
    return huffman_decode(huffman_tree, encoded)
## Turing-complete programmability 4

### Domain-specific video codecs

<table>
<thead>
<tr>
<th></th>
<th>Surveillance video</th>
<th>Self-driving car LIDAR footage</th>
<th>Academic videos (lecture videos, talks and tutorials)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application-specific requirements</td>
<td>Some objects (people/ car plates) are higher priority</td>
<td>E.g. depth resolution</td>
<td>Text needs to be readable</td>
</tr>
<tr>
<td>Probabilistic profile</td>
<td>Static scenery, Traffic &amp; weather</td>
<td>Surrounding traffic video</td>
<td>Emphasis on text, Illustrations, screencasting</td>
</tr>
<tr>
<td>Community / engineering resources to support it</td>
<td>Billion dollar industry</td>
<td>Self-driving car industry</td>
<td>Academic &amp; Huawei community</td>
</tr>
</tbody>
</table>
Turing-complete programmability 5

Timelines of adoption of new codecs is in decades
Turing-complete programmability

Vendoring software

Dynamic libraries
.dll / .so

Go compiler binaries:
Everything is statically included

Containerization:
Docker, Unikernels etc

# Fibonacci
```python
def fibonacci(n):
    fib = [0] * (n + 1)
    fib[0] = 0  fib[1] = 1
    for i in range(2, n + 1):
        fib[i] = fib[i - 1] + fib[i - 2]  return fib[n]
```

# SHA256: CC646:
Turing-complete programmability

1. Send data as programs that output the given data
2. Implement a sandboxed runtime that is secure and deterministic
3. Distribute the code along with the data

Data as code: distribute the data and the code that reads it along the same channel
More details and live demos @ https://kolmoblocks.org/

KolmoLD

enable a new kind of computer networks, networks that are content-addressable, feature composable data, and let you distribute the code to read the data along with the data itself.

VIDEO TOUR  LIVE EXAMPLES

KolmoLD: Data Modeling for the Modern Internet
Example:

Hello world!

Hello

world!

Example:
Example:

```plaintext
(match,
 (cid, "B5D40"),
  (exec
   (cid, "cat",
    type: "application/wasm+kolmold"
   ),
   (cid,
    type: "text/plain",
    data: "Hello",
   ),
   (cid,
    type: "text/plain",
    data: "world",
   ),
  ),
)

(match,
 (cid, "B5D40"),
  (exec
   (cid, "cat",
    type: "application/wasm+kolmold"
   ),
   (cid,
    type: "text/plain",
    data: "He",
   ),
   (cid,
    type: "text/plain",
    data: "llo world",
   ),
  )
)
def main():
    out.low = dep1.low
    out.high = dep2.high

Kolmoblock, cid:"cat"
type: application/wasm+kolmold

Globals of wasm module’s instance

    dep1.low=0
    dep1.high=6
    dep2.low=6
    dep2.high=12
    out.low
    out.high

Linear memory of the wasm module’s instance

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>e</td>
<td>l</td>
<td>l</td>
<td>o</td>
<td></td>
<td>w</td>
<td>o</td>
<td>r</td>
<td>l</td>
<td></td>
<td>d</td>
</tr>
</tbody>
</table>

Hello world!

dep1, dependency datablock 1
dep2, dependency datablock 2
KolmoLD: Kolmogorov Linked Data

Addressable: connecting layer, inspired by the principles of Kolmogorov complexity theory

Composable: sending data as code, where code efficiency is theoretically bounded by Kolmogorov complexity

Computable: sandboxed computability by treating data as code

Democratizing networking of generic ICT devices with a principled approach
Thanks!

KolmoLD: Data Modelling for the Modern Internet*

Dmitry Borzov, Huawei Canada
Tim Tingqiu Yuan, Huawei
Mikhail Ignatovich, Huawei Canada
Jian Li, Futurewei

*work performed before May 2019