Algorithms for Dispersed Processing

Josef Spillner, Alexander Schill

cmailto:josef.spillner@tu-dresden.de
xmpp:josef.spillner@jabber.org

Motivation

»Don't trust the cloud.«

Risks:
• temporary unavailability
• permanent unavailability
  • of data [loss]
  • of service [bankruptcy]
• arbitrary slowness
• unauthorised access [honest-but-curious] || 3rd-party spying
• malicious modification
• refusal to delete
• lock-in, transfer costs
• ... etc. pp.

Insufficient but essential protection:
Information dispersal over multiple clouds. => IDA
Definition

Popular definition:

Information dispersal algorithms are used to separate data packets into slices so that they are unrecognizable as they sit in storage arrays or traverse the network. Data can be reassembled at the receiving device.

But: Why just sit or traverse?

Storage  « √
Networking « √
Processing « ???
Background: Information Dispersal

Generalised Dispersed Processing

Storage

Network (Communication)

Processing (Computation)

data locality
(a), (c): central processing
(b), (d): map-reduce

IDA of choice: bitsplitting; e.g. 50% redundancy: k=2, m=1, n=k+m

Property: structure-preserving
Dispersed Processing: Summation Algorithm

Addition:
requires \( n=2 \)

Multiplication:
requires \( n=4 \)
(unless \( a, b \neq 0 \))

Method:
- map: sum/mult
- reduce: sum

\[(a) \text{ summation } 23 + 24 = 3 \text{ summations}\]

\[(b) \text{ multiplication } 23 \times 24 = 4 \text{ multiplications} + 1 \text{ summation with bitshifts}\]

\[(c) \text{ summation } 23 + 24 = 3 \text{ summations}\]
Dispersed Processing: Search Algorithm

Pattern search without index

Method:
- split pattern, too
- map: perform $k$ partial searches
- reduce: filter out false positives

Naïve search over full text

text: [1:hi lo] [2:hi lo] [3:hi lo] [4:hi lo]

pattern: [1:hi lo] [2:hi lo] ...

Naïve search over split data

text 1: [1:hi 2:hi] [3:hi 4:hi] [5:hi 6:hi] ...

pattern 1: [1:hi 2:hi] [3:hi 4:hi] ...

pos n/2

text 2: [1:lo 2:lo] [3:lo 4:lo] [5:lo 6:lo] ...

pattern 2: [1:lo 2:lo] [3:lo 4:lo] ...

pos n/2

pattern

pos n
Limits of parallelisation
- fixed-size integer shifts → carry bits
- filter operations → position data

Security effect: Information leakage

\[ \alpha := \{(\text{pos1}, \text{frag1}), (\text{pos2}, \text{frag2}), \ldots\} \]
Dispersed Processing: Encryption

Processing dispersed + encrypted data blocks

Property: *structure-preserving*

Algorithms:
- homomorphic encryption → arithmetics
- order-preserving encryption → sorting
- convergent encryption → deduplication
Dispersed Processing: Encryption Example

Travel distance:
71 km
+ 19 km
----------
= 90 km

generate-keypair \( \text{bits}=16 \) \( \Rightarrow \{k_{\text{priv}}, k_{\text{pub}}\} \)

71 km \( k_{\text{pub}} \) \( \rightarrow \) 283060154

19 km \( k_{\text{pub}} \) \( \rightarrow \) 630596813

Arithmetic: \( a \times b \mod \text{bits} \)
283060154 “km”
+ 630596813 “km”
----------
= 540987952 “km”

\( k_{\text{pub}}, k_{\text{priv}} \)

Travel distance:
540987952 „km“ = 90 km
If a cloud fails...

... repair and continue processing?
... or, expect degraded results?

Application-specific decision.
Example: floating point with (i)nteger, (f)ractional, (r)edundant parts.
Redundant data is not generally processable. Distribution matters.
Classification of algorithms (pending)
Performance Evaluation

Splitting Performance

- Bitsplitter/C
- Python*
- JeraisueCG
- JeraisueRSV

Handover protocol performance

- min-sr
- max-sr
- median-sr
- min-Lr
- max-Lr
- median-Lr

1st CLASP, 11.12.2014
Dispersed Computing

Storage, networking: Much existing research
Processing: need for special algorithms

Evaluation: slower processing vs. (often) less transmission

Code: git://nubisave.org/git/dispersedalgorithms

Stealth Computing

Combination of dispersion x encryption and further quality measures
Enabler for native cloud applications