Performance evaluation of CRDTs algorithms for document editing

Why Evaluate ?

- So many collaborative applications/features
  - Real-time collaborative editors (GDocs, Etherpad, CoWord, …)
  - Wikis (Wikipedia, Uniwiki, wiki using paxos…)
  - (D)VCS (svn, git, darcs, Hg, …)
  - Cloud sharing tools (Dropbox, GDrive, Evernote, …)
  - …
- Different architectures : From centralized to P2P
- Different requirements
  - From single user (multiple devices) to massive
  - From “real-time” to controlled push/pull communication
  - From operation based to state based
  - From mobile devices to personal computers

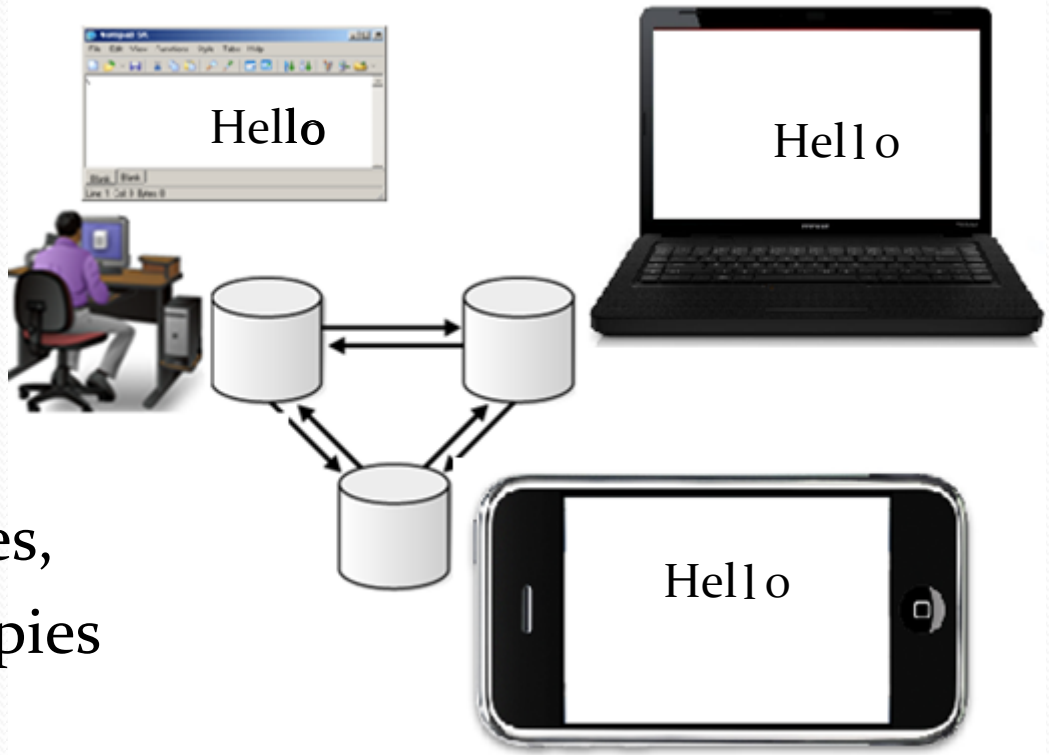
Objective

- Help in designing collaborative editing systems
  - Symmetric (Completely decentralized)
    - *Google don't need us, in the hope that we do not need Google ;)*
  - Ensuring causality
    - Required by users (?), required by (almost all) algorithms
- Select and evaluate potential algorithms
  - Execution time (must be before 50 ms)
  - Memory (lower is better)
  - Merge result (user satisfaction)
- Framework
  - Simulate real-time collaboration traces based on [1]
  - Retrieve real collaboration traces from Git
  - Execute all algorithms in same condition

[1]: M. Ahmed-Nacer, C.-L. Ignat, G. Oster, H.-G. Roh, and P. Urso, "Evaluating crdts for real-time document editing," in ACM Symposium on Document Engineering, ACM, Ed., San Francisco, CA, USA, september 2011, p. 10 pages

Optimistic Replication

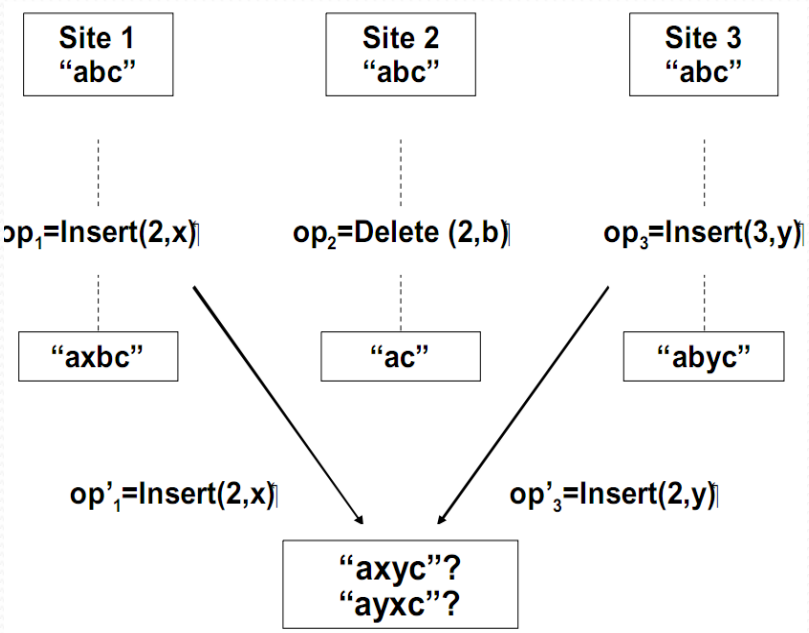
- A copy per site
- An operation is :
  - Locally executed,
  - Sent to remote sites,
  - Received by remote sites,
  - Executed on remote copies



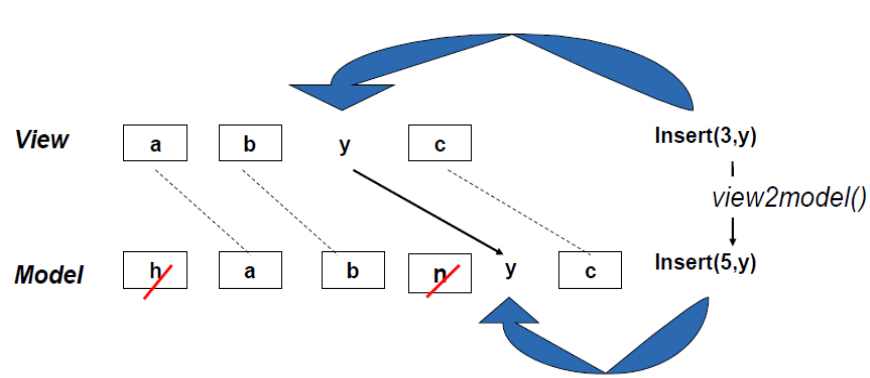
Algorithms Evaluated :
Operation Transformation (OT)

- An operation is :
  - Locally executed,
  - Sent to other sites,
  - Received by a site,
  - *Transformed according to concurrent operations*,
  - Executed on local copy
- 2 components :
  - An integration algorithm : diffusion, integration (*SOCT2*)
  - Some transformation functions (TTF)

Algorithms Evaluated :
Operation Transformation (OT)



TTF



Algorithms Evaluated :
Commutative Replicated Data Types (CRDT)

- Concurrent operations natively commutative
- Document = linear sequence of elements
- Unique position identifiers
  - Each element has a unique identifier
  - Constant in the lifetime of the document
  - Total order of identifiers consistent with element order
- Different approaches for generating identifiers:
  - Logoots , RGA, WOOTs, Treedoc

Algorithms Evaluated :

Commutative Replicated Data Types (CRDT)

- Logoot : *id* = list <*id* , *num* , *h*>.

Document	Identifiants
	<0, NA, NA>
A	<5, 1, 30>
B	<6, 1, 50>
C	<20, 3, 100>
	<Max, NA, NA>

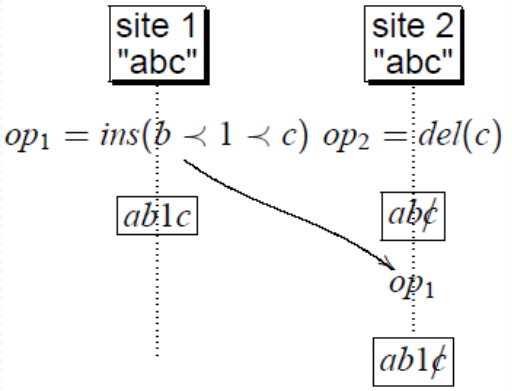
Replica2 insert("XY", 2)

Document	Identifiants
	<0, NA, NA>
A	<5, 1, 30>
X	<5, 1, 30><2, 1, 60>
Y	<5, 1, 30><4, 1, 61>
B	<6, 1, 31>
C	<20, 3, 100>
	<Max, NA, NA>

Replica3 insert("KMN", 2)

Document	Identifiants
	<0, NA, NA>
A	<5, 1, 30>
X	<5, 1, 30><2, 1, 60>
K	<5, 1, 30><2, 1, 60><3, 3, 20>
M	<5, 1, 30><2, 1, 60><5, 3, 21>
N	<5, 1, 30><2, 1, 60><10, 3, 22>
Y	<5, 1, 30><4, 1, 61>
B	<6, 1, 31>
C	<20, 3, 100>
	<Max, NA, NA>

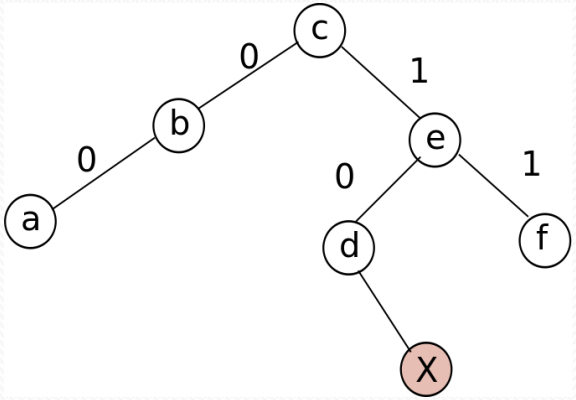
- WOOTO-WOOTH : Ins(p $\prec$ c $\prec$ n)

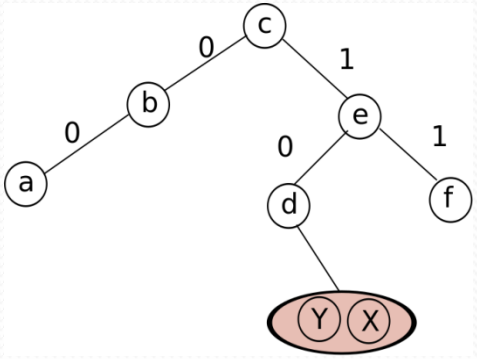


• Replace the deleted characters by a tombstone

Algorithms Evaluated :
Commutative Replicated Data Types (CRDT)

- Treedoc
  - Document = Binary tree structure
  - PosID = path
  - Lookup : walking the tree in infix order : *"abcdxef"*
  - Balanced sub-trees
  - Concurrent insertion at the same position→ Major-node

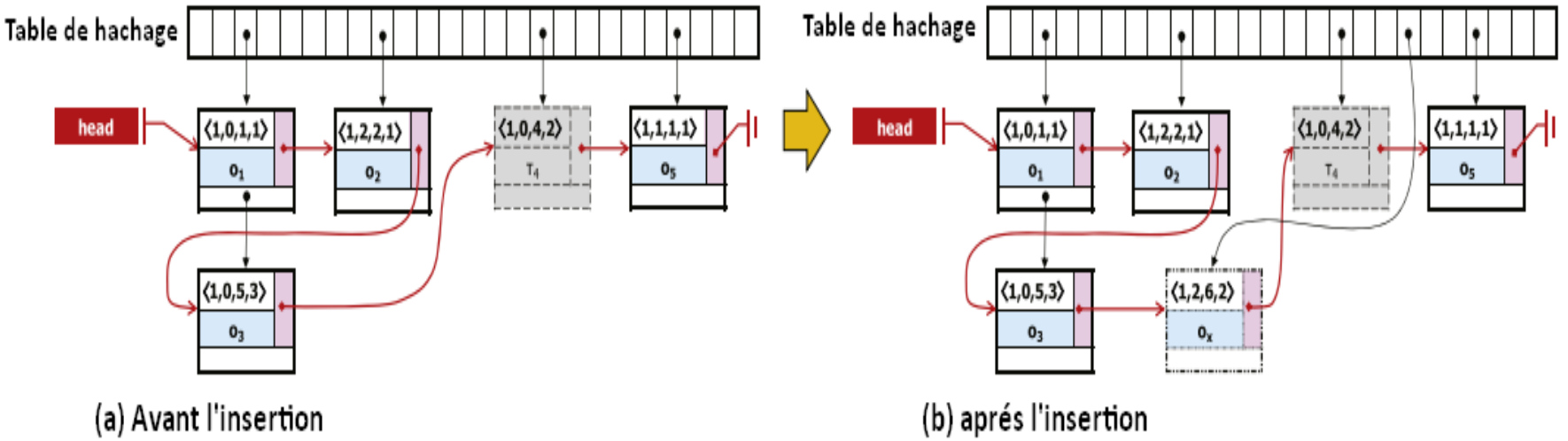




Algorithms Evaluated :

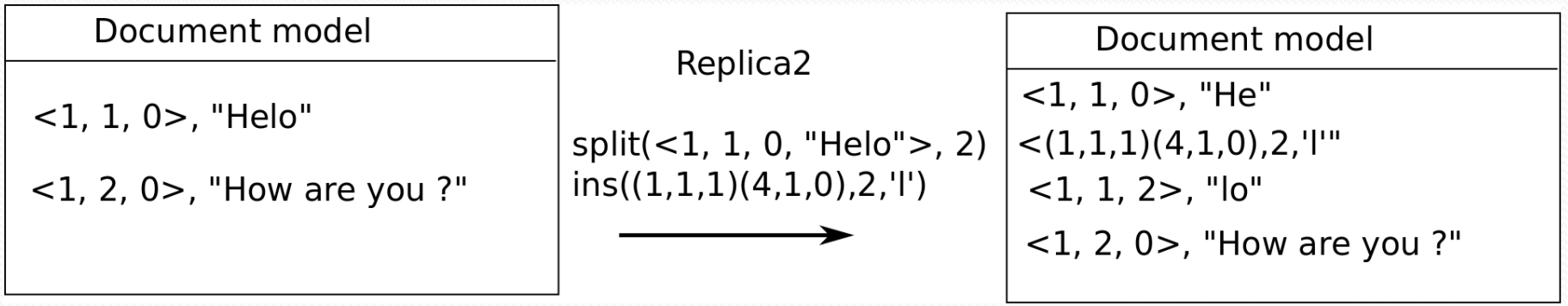
Commutative Replicated Data Types (CRDT)

- RGA :
  - Unique identifier to each character : *s4vector*
  - Represent an element by a node of a linked list



Algorithms Evaluated :
Commutative Replicated Data Types (CRDT)

- Logoot Split
- Based on Logoot algorithm but granularity is variable
  - Insertion of string
  - Deletion of strings
  - Possibility to split a string into two sequences
  - Id = [<Pos, Site, OffSet>, Clock, String]



Algorithms Evaluated :

Commutative Replicated Data Types (CRDT)

- Logoot List
- Based on Logoot algorithm but *length(id)* = 1
- Identifier must be dense and unique:
  - Concatenation : *Pos.Replica.Clock.DigRep.DigClock*
- Example :

Logoot identifier : <12, 14, 100>

(14 in 2 digits and 100 in 3 digits)

Id : 121410023

- LogootList Identifier as byte[]

Document	Identifiers
A	[-128,-127,1,17]
B	[-128,-127,2,17]
C	[-128,-127,3,17]

Replica2
ins(XY, 2)

Document	Identifiers
A	[-128,-127,1,17]
X	[-128,-127,1,17,1]
Y	[-128,-127,1,17,2]
B	[-128,-127,2,17]
C	[-128,-127,3,17]

Algorithms Evaluated :
Commutative Replicated Data Types (CRDT)

Document	Logoot Identifiers	Logoot List Identifiers
...	...	...
A	<5, 1, 30>	<513012>
B	<6, 1, 50>	<615012>
C	<20, 3, 100>	<20310013>
...	...	...

Replica2 : ins(XY, 2)

Document	Logoot Identifiers	Logoot List Identifiers
A	<5, 1, 30>	<513012>
X	<5, 1, 30><2, 1, 60>	<522111>
Y	<5, 1, 30><4, 1, 61>	<532111>
B	<6, 1, 50>	<615012>
C	<20, 3, 100>	<20310013>

Theoretical comparison

ALGORITHMS	LOCAL		REMOTE	
	INS	DEL	INS	DEL
WOOTO	$O(N.d^2)$	$O(N)$	$O(N.d^2)$	$O(N)$
WOOTH	$O(N+d^2)$	$O(N)$	$O(d^2)$	$O(1)$
Logoot	$O(k)$	$O(1)$	$O(k.\log(n))$	$O(k.\log(n))$
Logoot Split	$O(n/l+l)$	$O(n/l+l)$	$O(k'.\log(n/l)+l)$	$O(k'.\log(n/l)+l))$
Logoot List	$O(b)$	$O(1)$	$O(b.\log(n)$	$O(b.\log(n))$
TreeDoc	$O(S.\log(N))$	$O(S.\log(N))$	$O(S.\log(N))$	$O(S.\log(N))$
RGA	$O(N)$	$O(N)$	$O(1+c/n)$	$O(1)$
SOCT2/TTF	$O(N + R)$	$O(N + R)$	$O(H.c)$	$O(H.c)$

R : number of replicas
H : number of operations
N : the total number of elements
n : the size of document(non deleted characters)
k (k') : the average size of Logoot(Split) identifiers

b : the average size of LogootList identifiers
l: length of the strings
c: the average number of operations concurrent to a given one.
d: the average number of element found between two successive elements.
S : the average size of major-node

Theoretical comparison

ALGORITHMES	Space Complexity	
	PIRE CAS	MOYENNE
Logoot	$O(H^2)$	O(k . n)
Logoot Split	$O(H^2)$	O(k'. n/l)
Logoot List	$O(H^2)$	O(b . n)
WOOTO-WOOTH	O(H)	O(N)
Treedoc	$O(H^2)$	O(N)
RGA	O(H)	O(N)
SOCT2/TTF	O(H . R)	O(H . R)

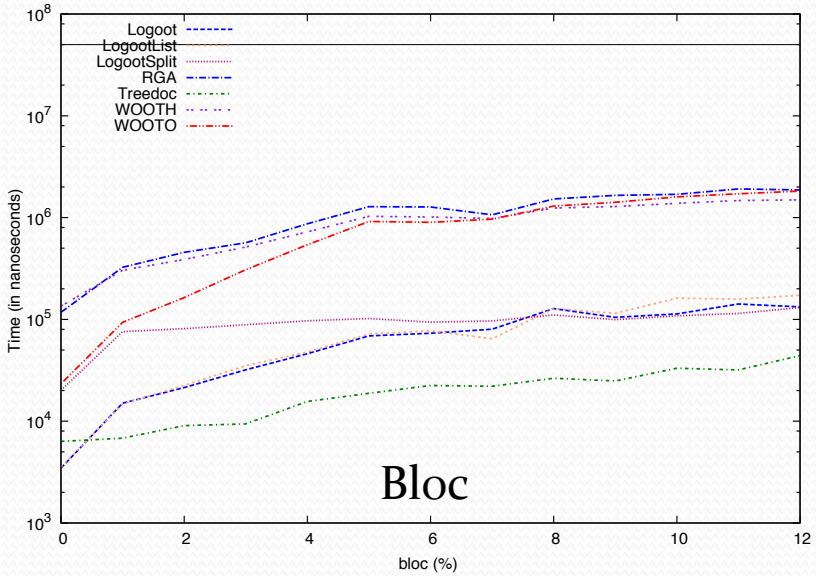
H : number of operations
R : number of replica
n : the size of document(non deleted characters)
k : the average size of Logoot identifiers
b : the size of array byte that represent LogootList identifiers
l: number of blocks of characters

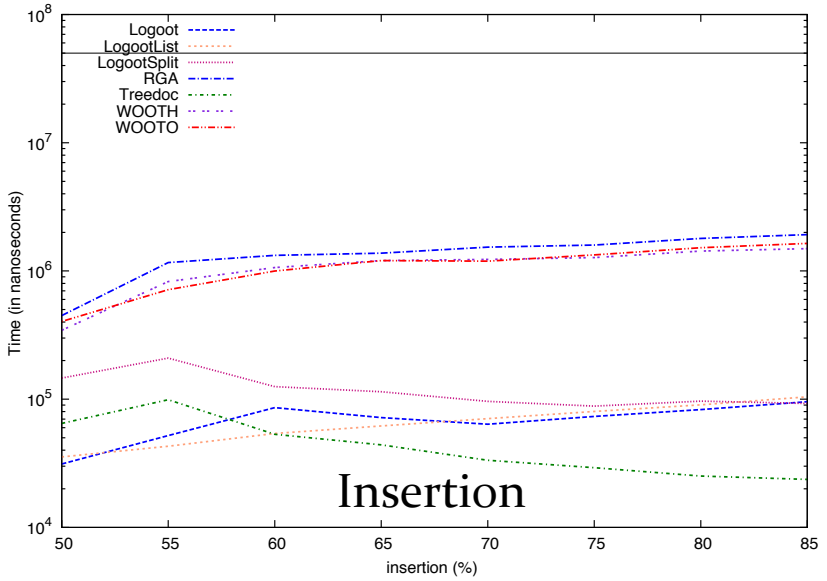
Experiment 1 : Simulation

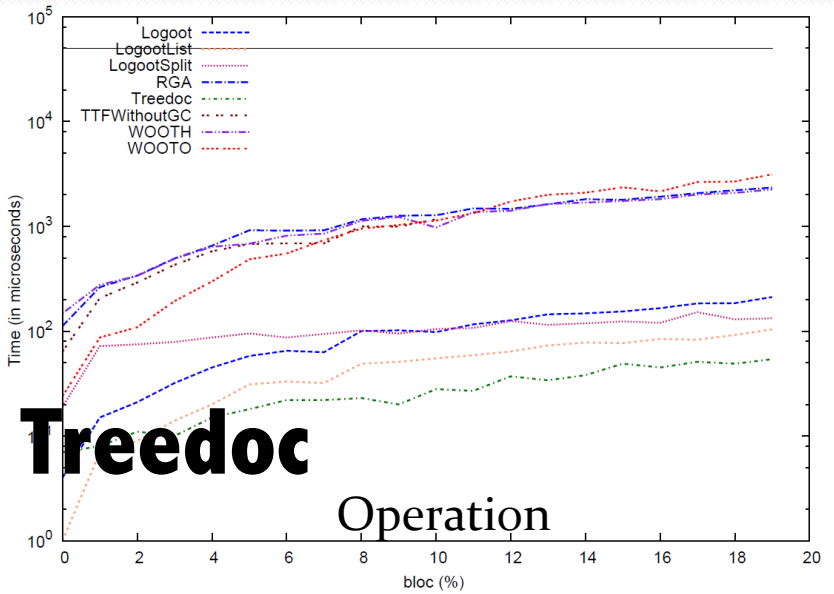
	experiment	Operation	Insertion
FEATURES	Nbr of operations	10 000 $\xrightarrow{+5000}$ 50 000	10 000
	Nbr of replica	10	10
	percentage blocks	15%	15%
	percentage insertion	80%	50% $\xrightarrow{+5\%}$ 100%
	AvgSizeBlock	100	100

	experiment	Blocks	Replica
FEATURES	Nbr of operations	10000	10 000
	Nbr of replica	10	10 $\xrightarrow{+5}$ 50
	percentage blocks	0% $\xrightarrow{+1\%}$ 20%	15%
	percentage insertion	80%	80%
	AvgSizeBlock	100	100

Local operations execution

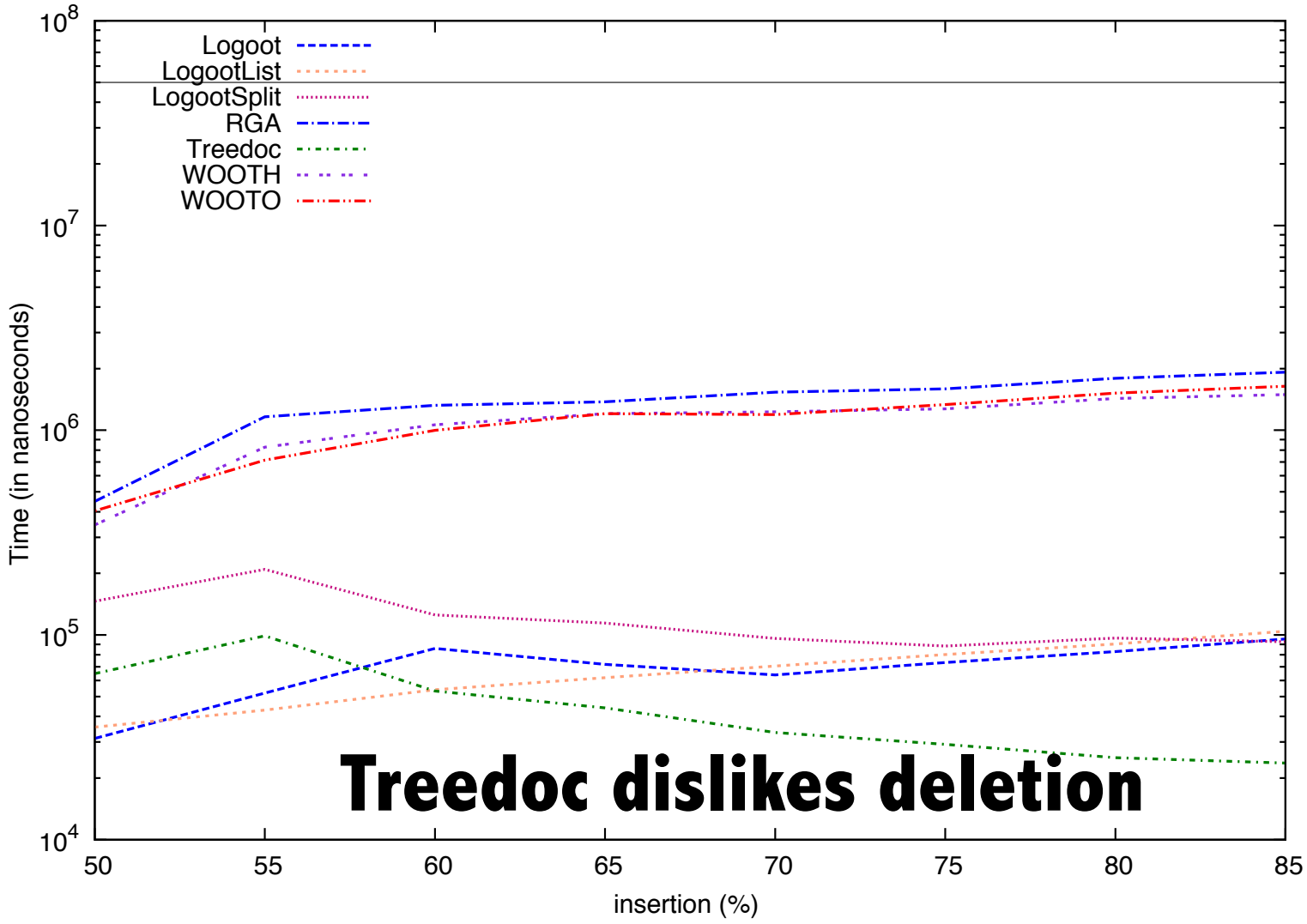




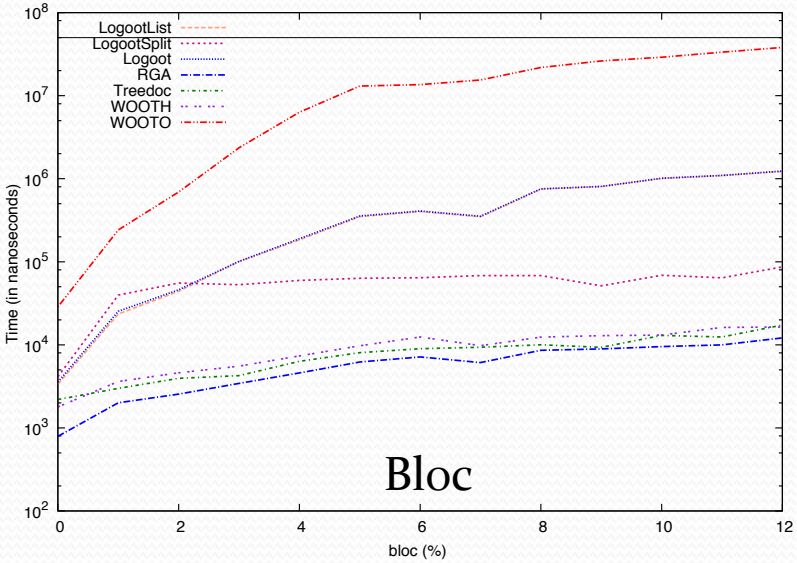


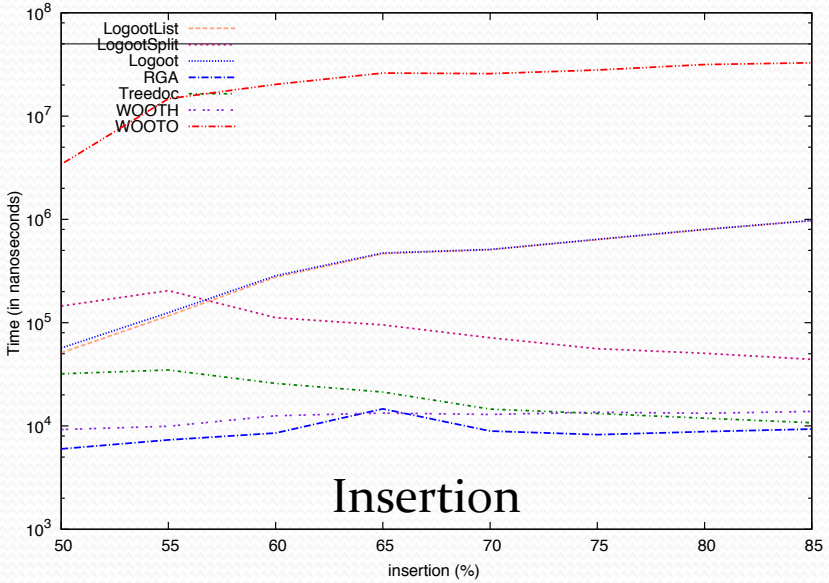
Winner is : Treedoc

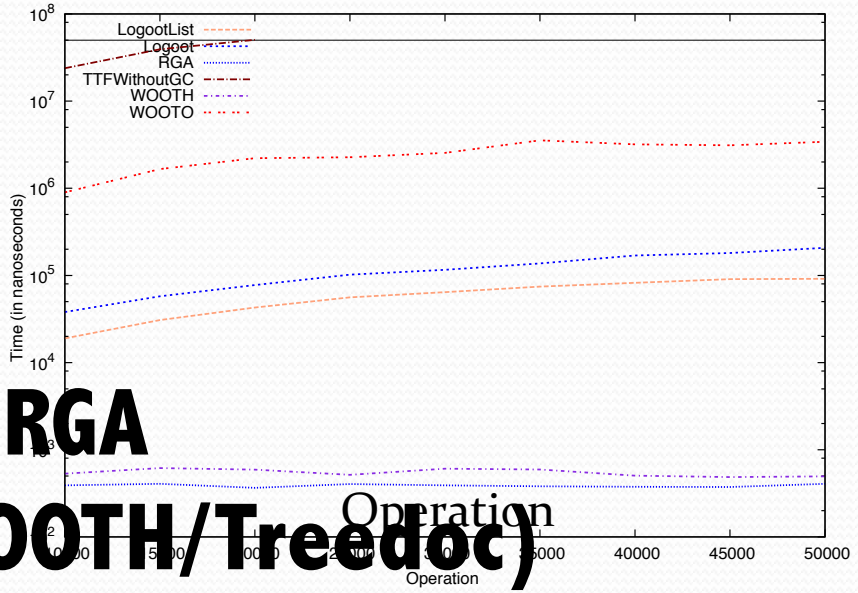
Local operations execution : Insertions



Remote operations execution

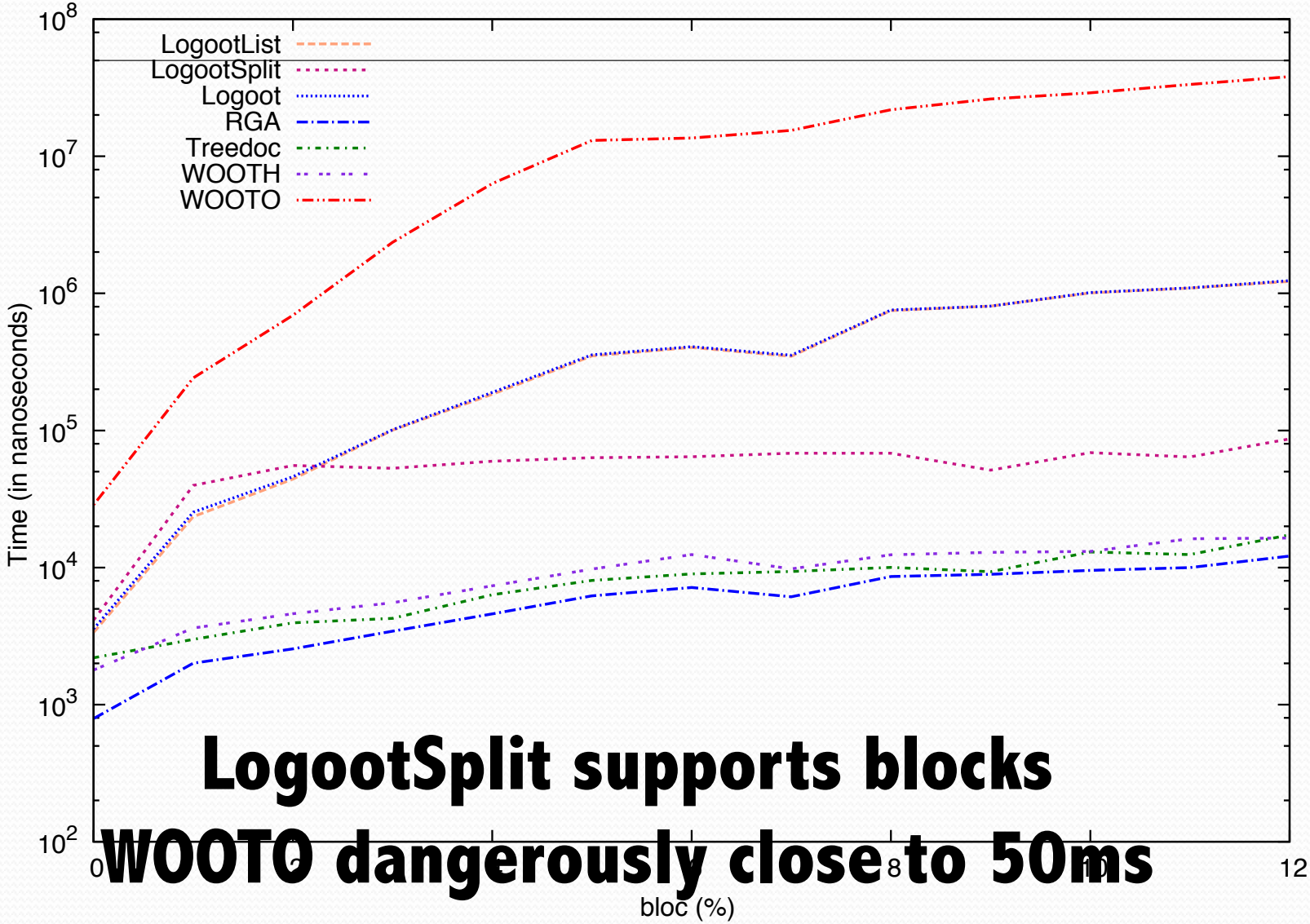






Winner is RGA
(close finish with WOOTH/Treedoc)

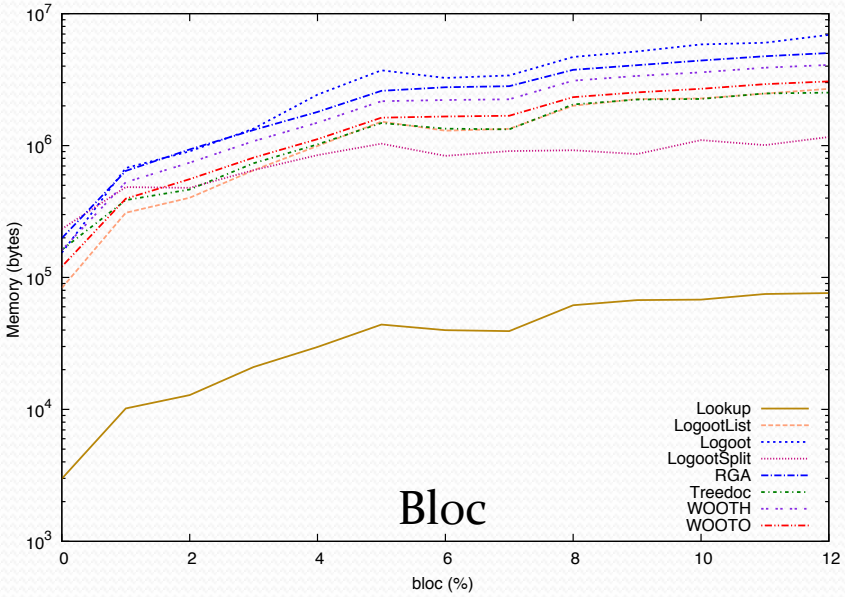
Remote operations execution : Blocks

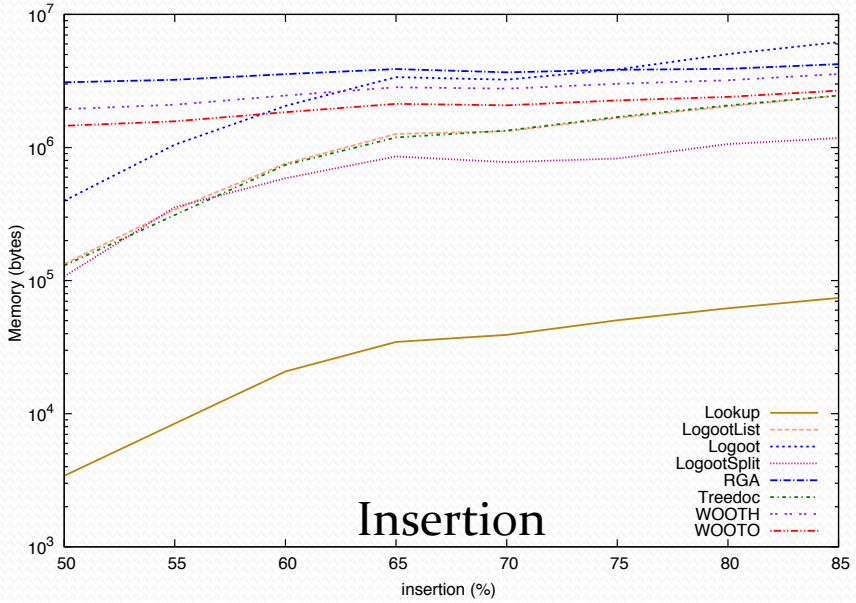


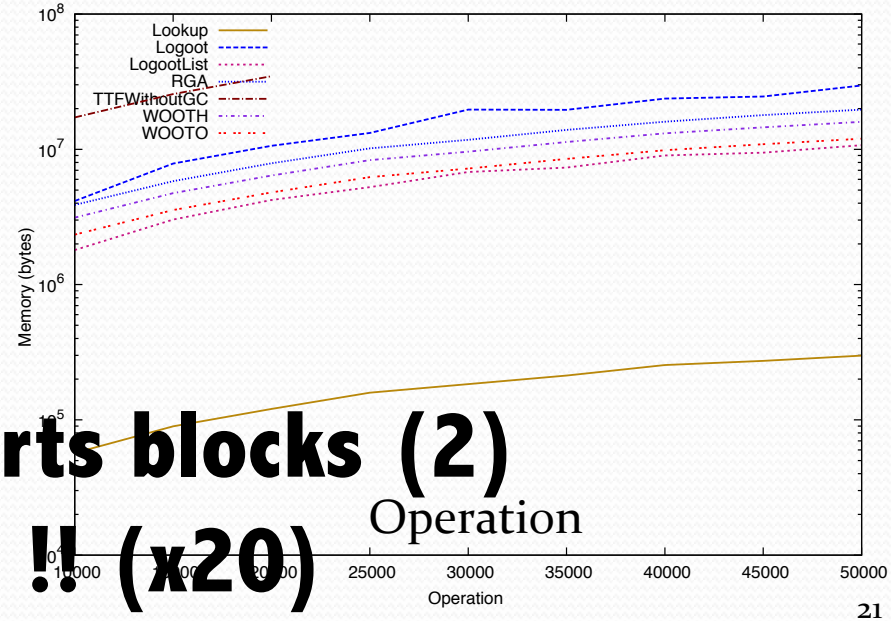
LogootSplit supports blocks

WOOTO dangerously close to 50ms

Memory



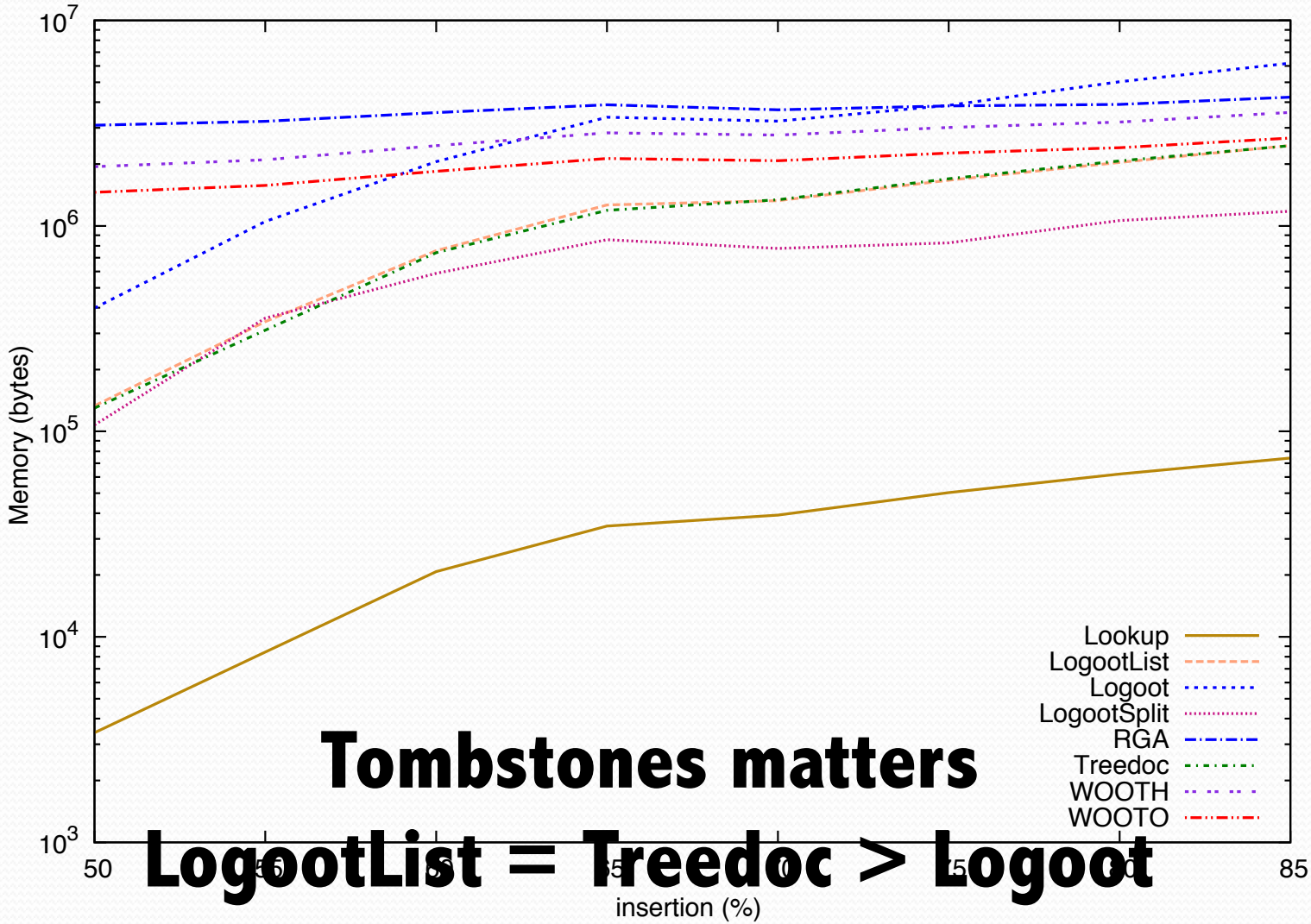




LogootSplit supports blocks (2)

Big overhead !! (x20)

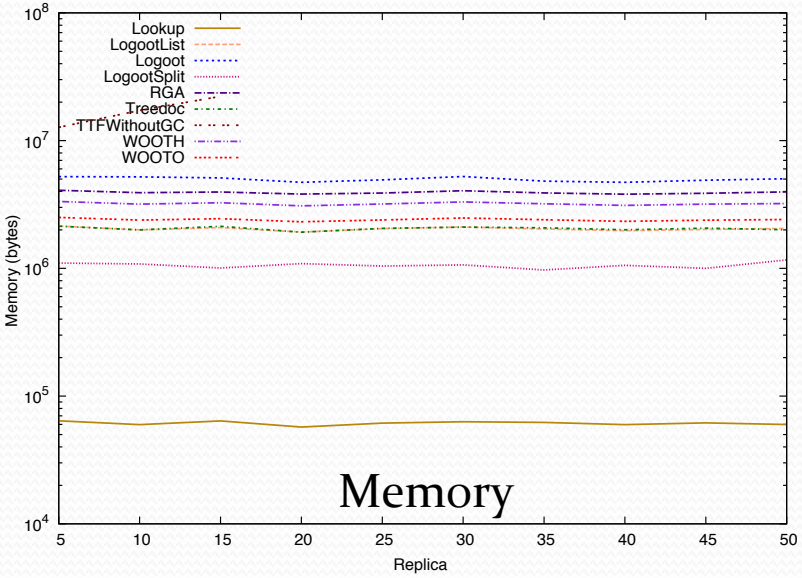
Memory : Insertion

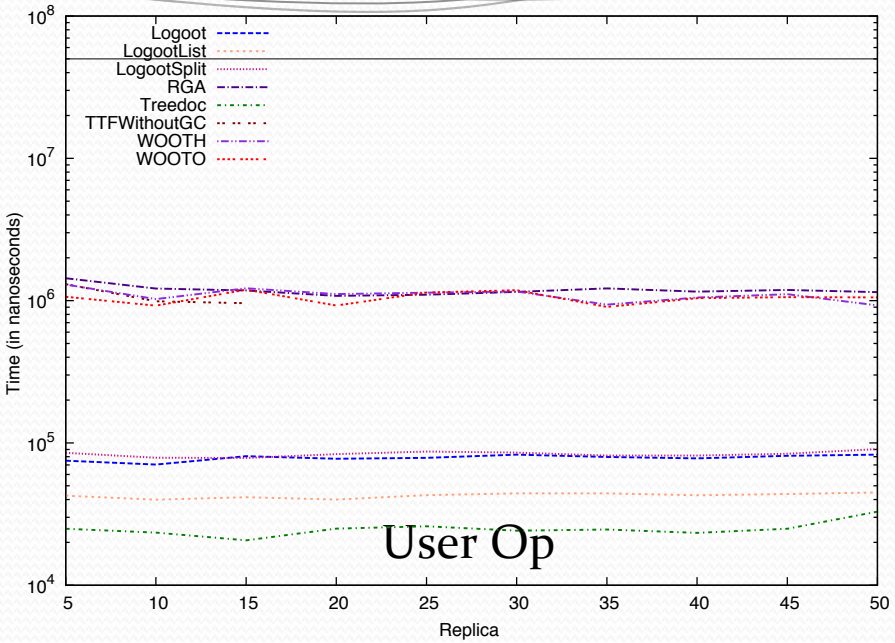


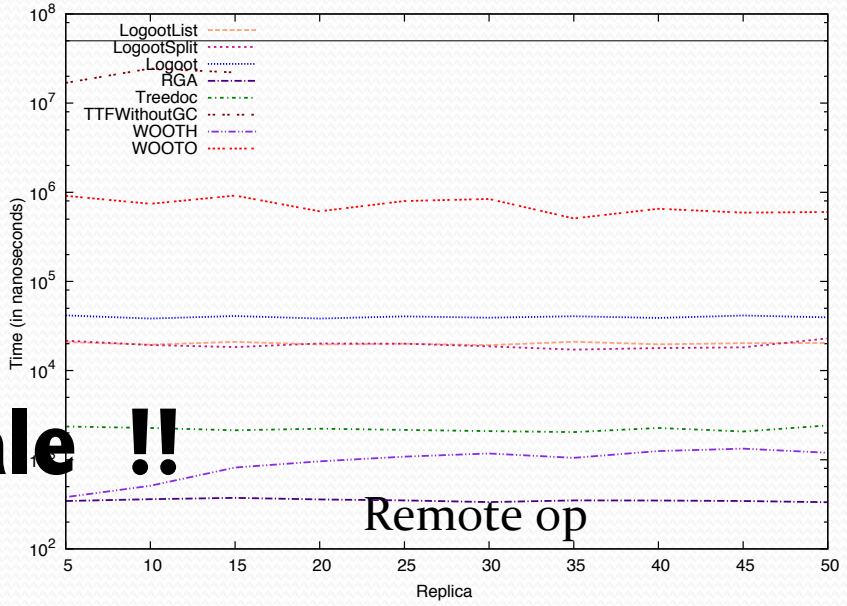
Tombstones matters

LogootList = Treedoc > Logoot

Replicas







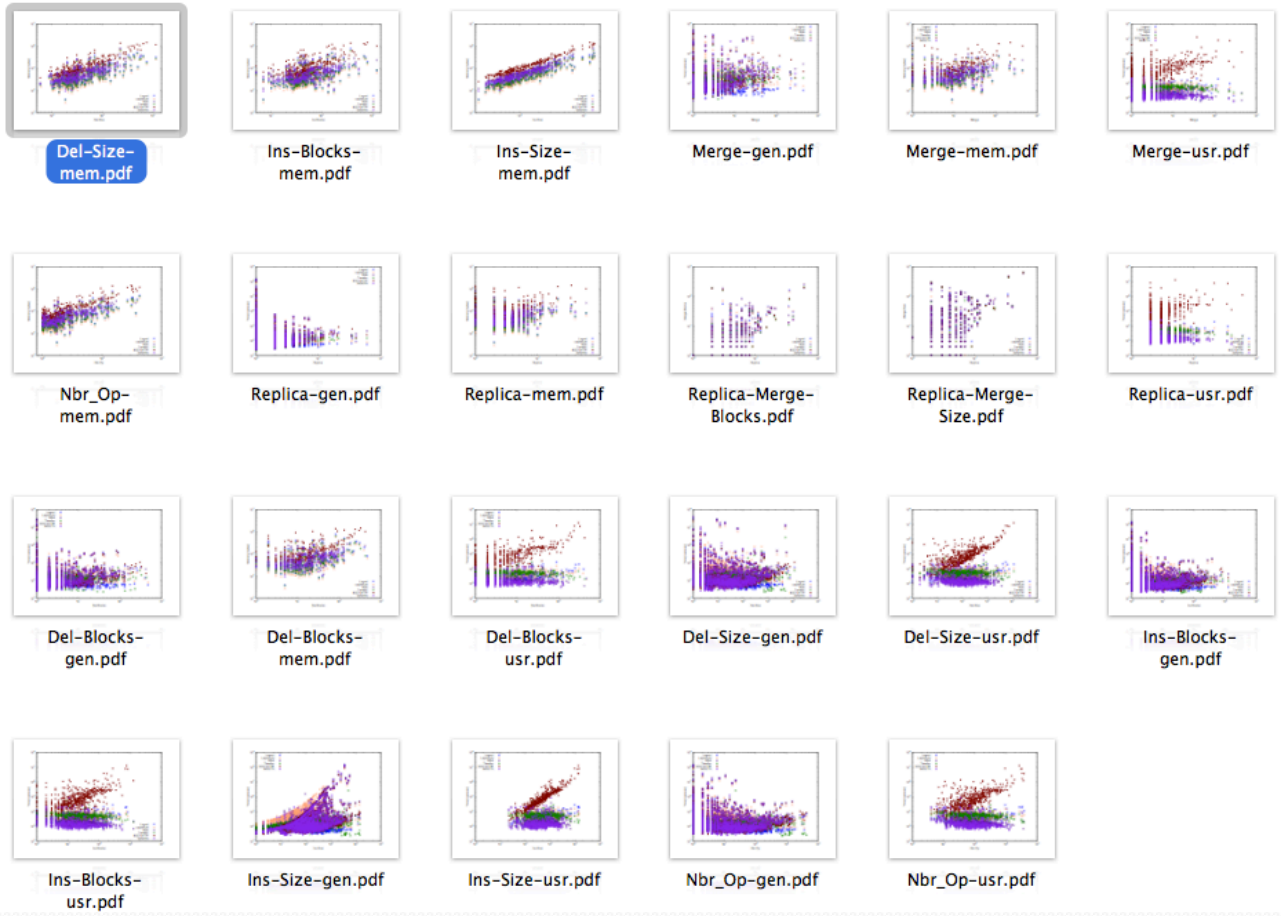
CRDTs scale !!

Experiment 2 : Git traces

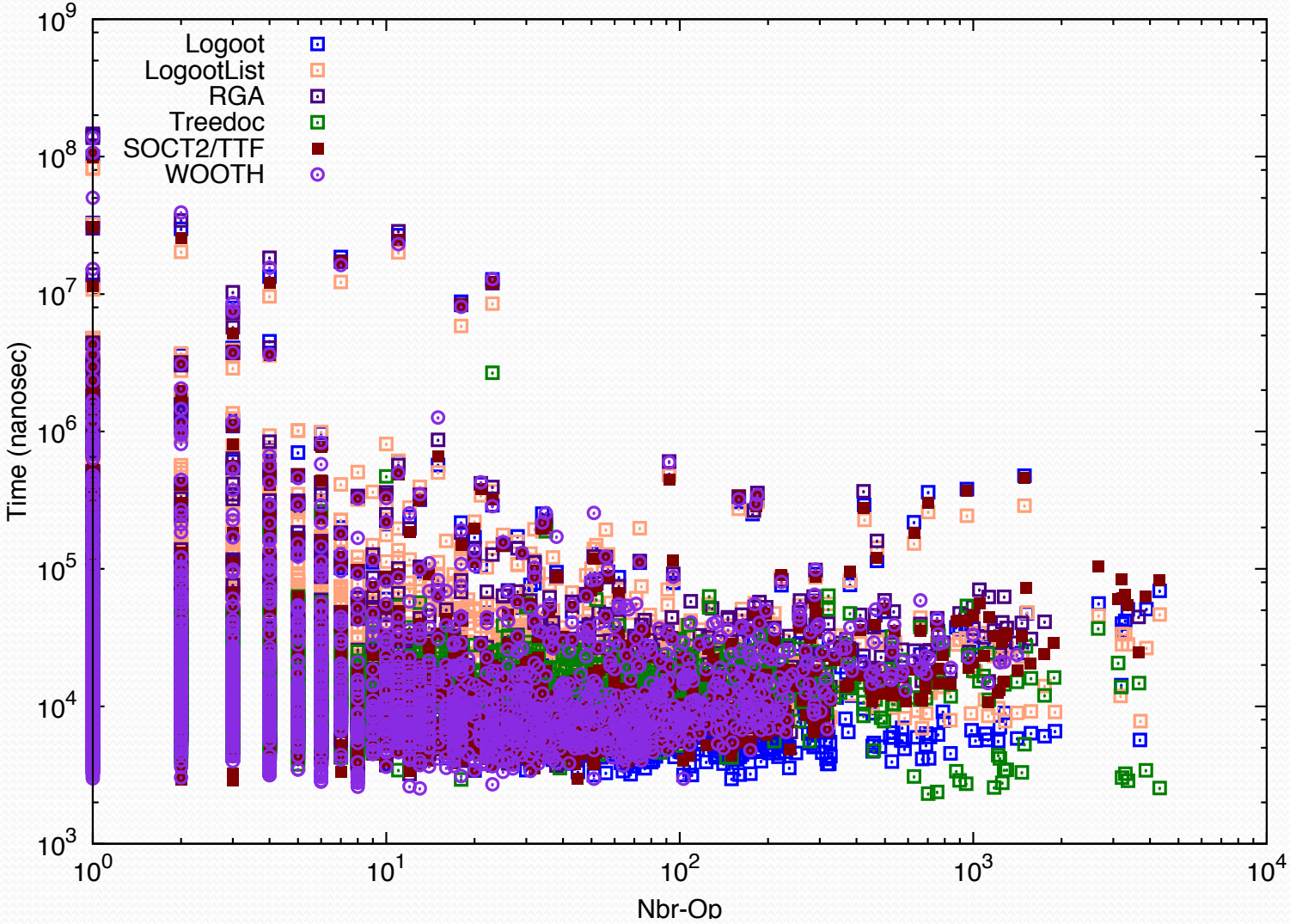
- Retreive traces of Git project from GitHub
- Total number of files : 2495

Characteristics	max	min	avg
Commit	1742	1	16
Deletion	11599	0	101
insertion	14815	47	337
update	4019	0	40
replica	60	1	2
bloc Del	432	0	5
bloc Ins	1263	0	13
operations	13959	1	440

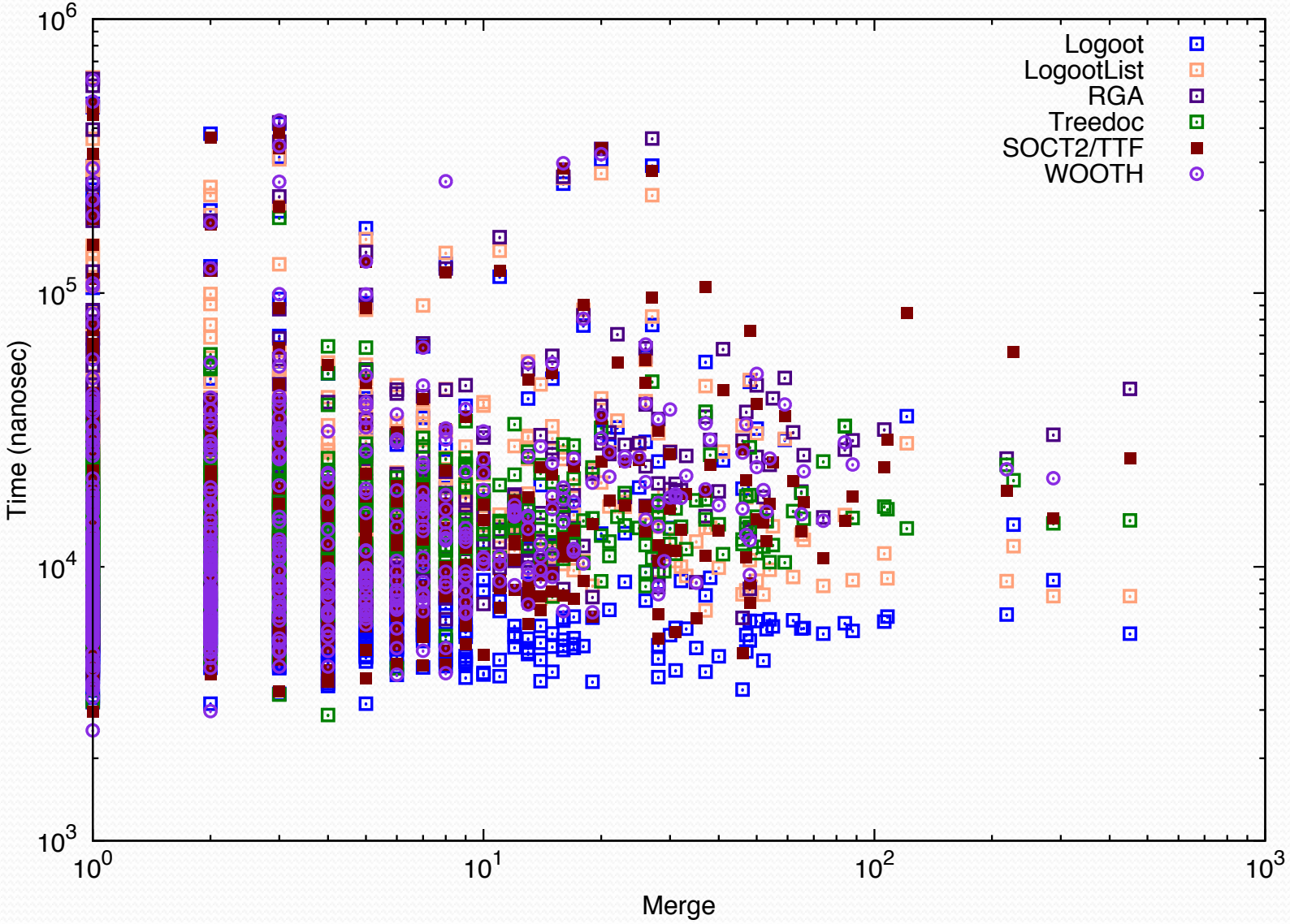
Many many results



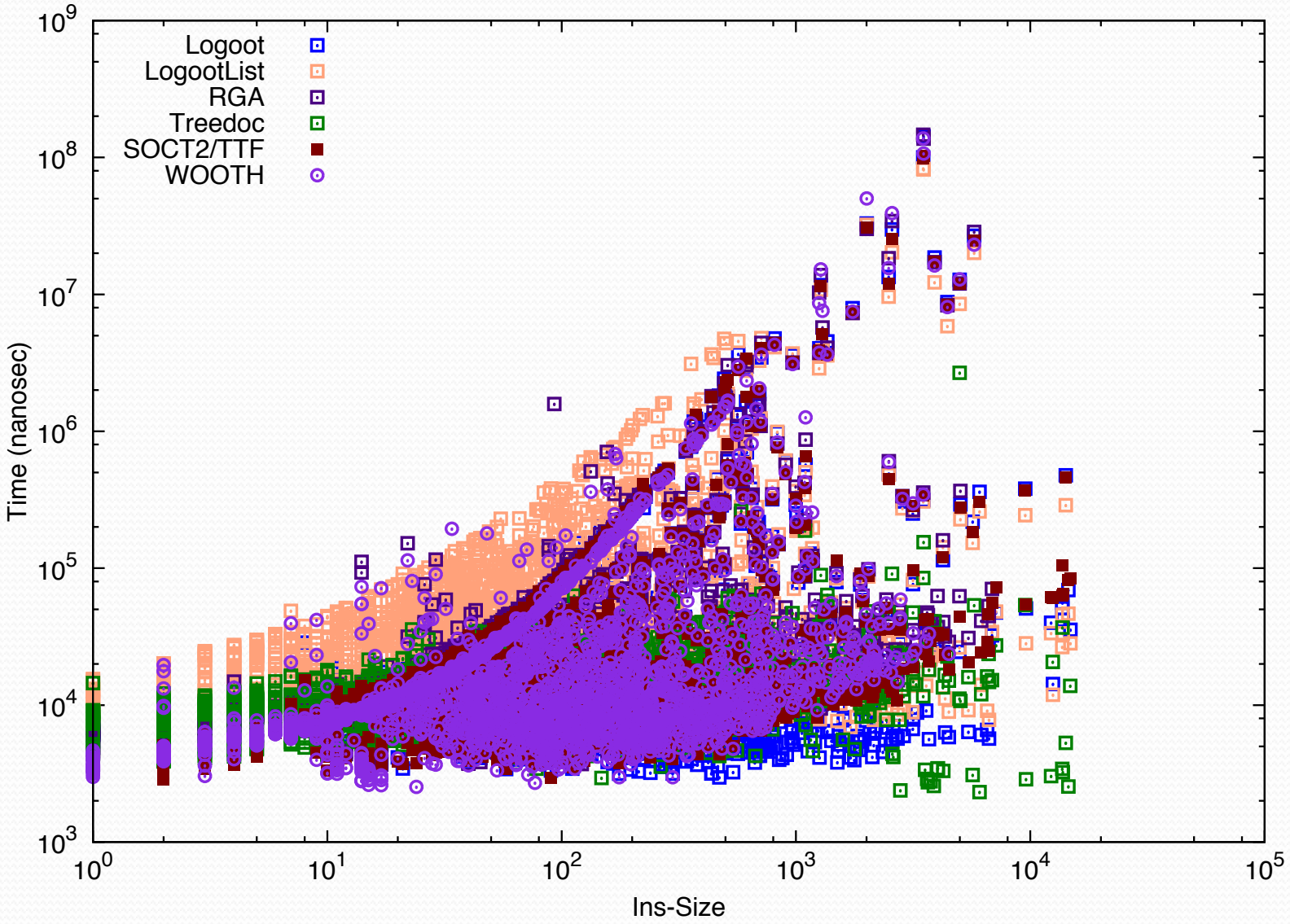
Local operations : op number



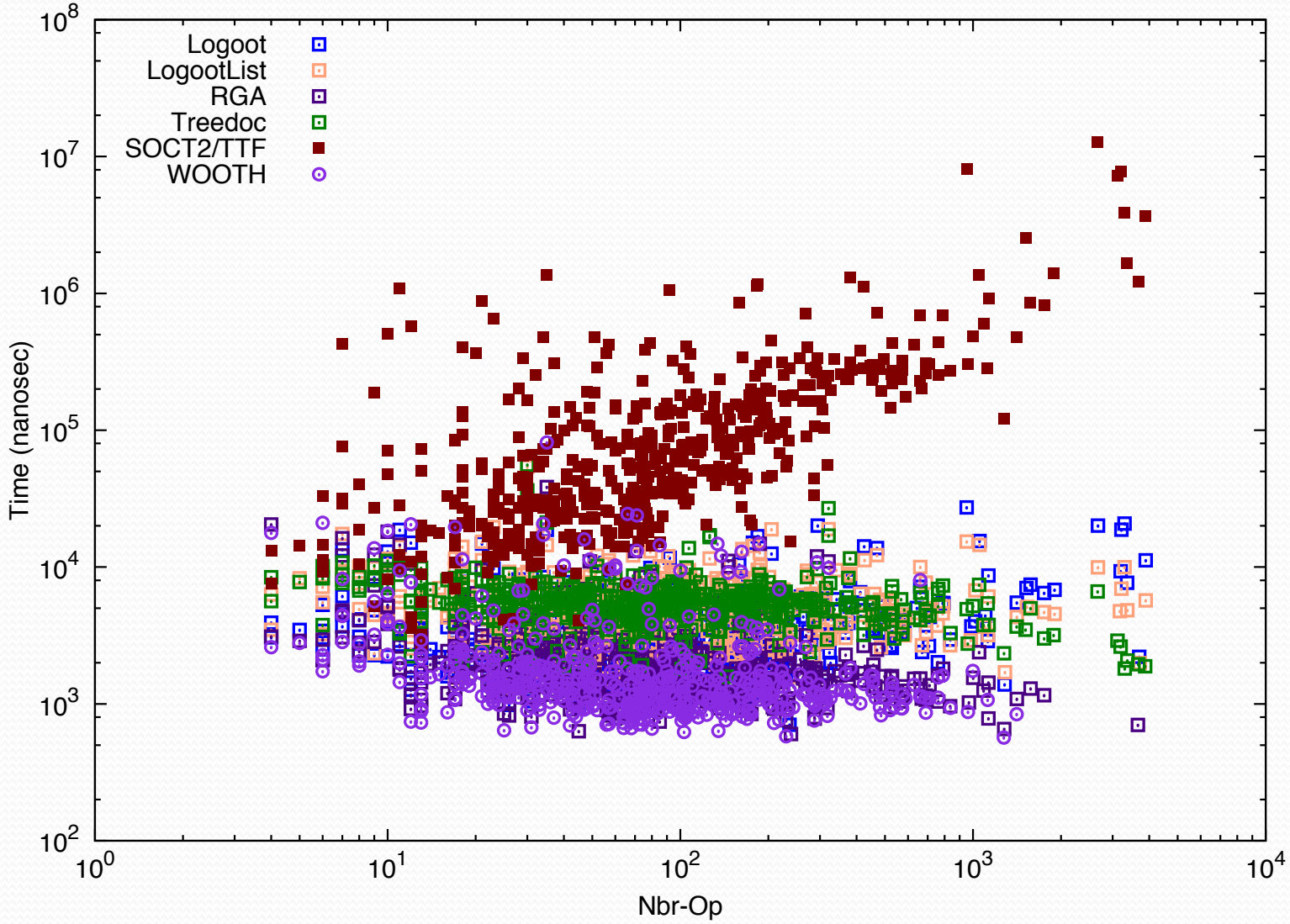
Local operations : merge



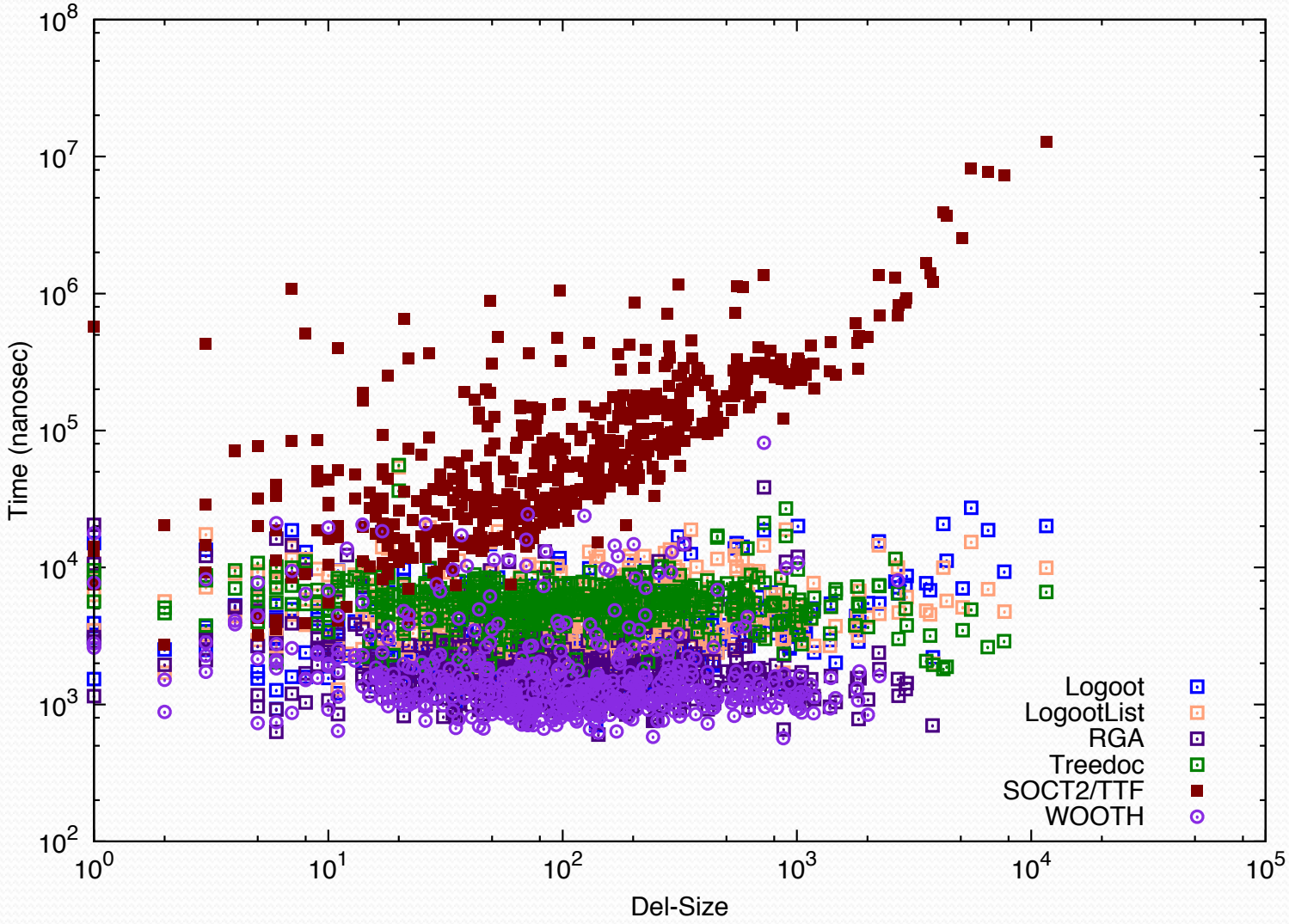
Local operations : Insert size



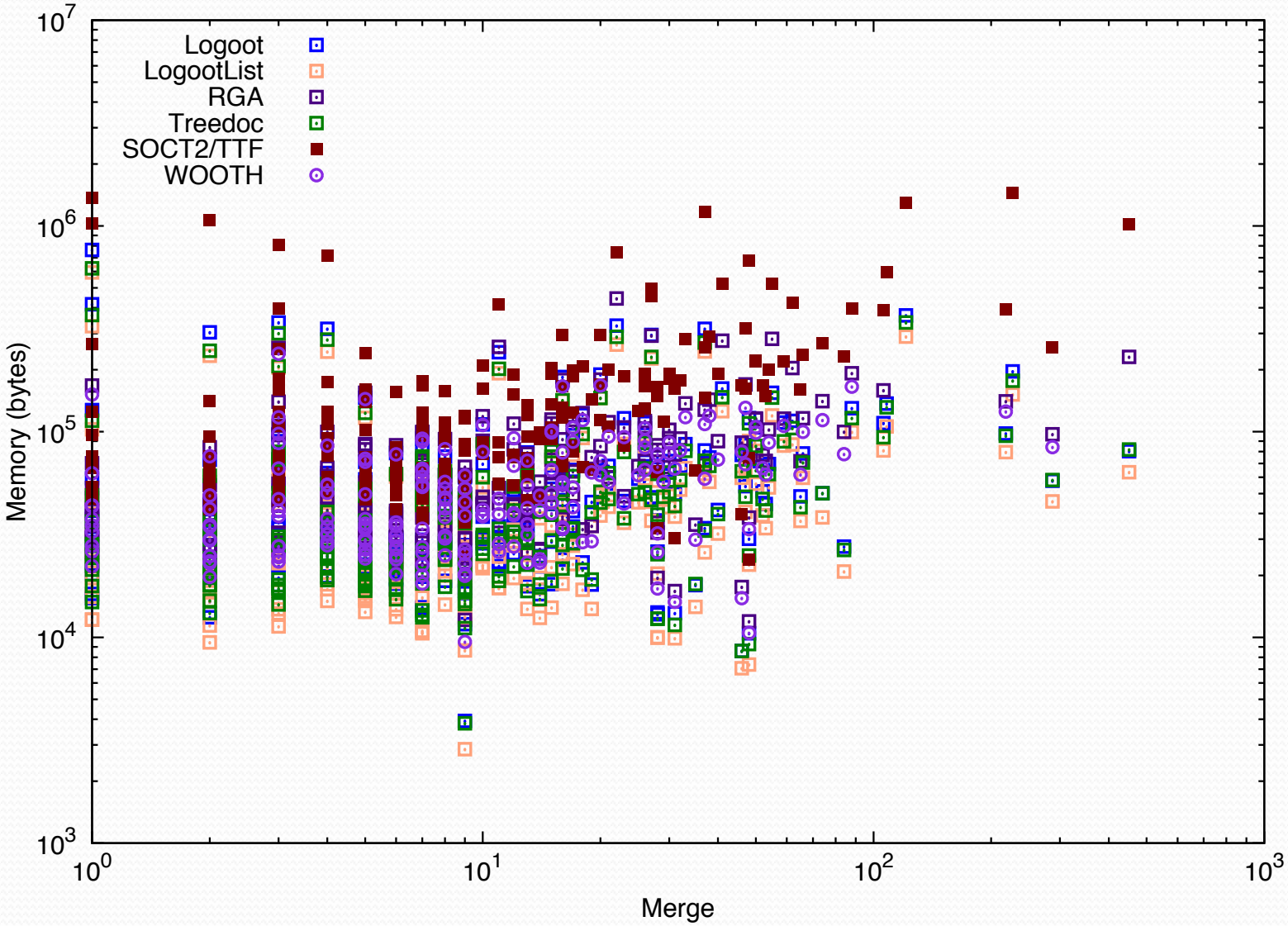
Remote operations : op. number



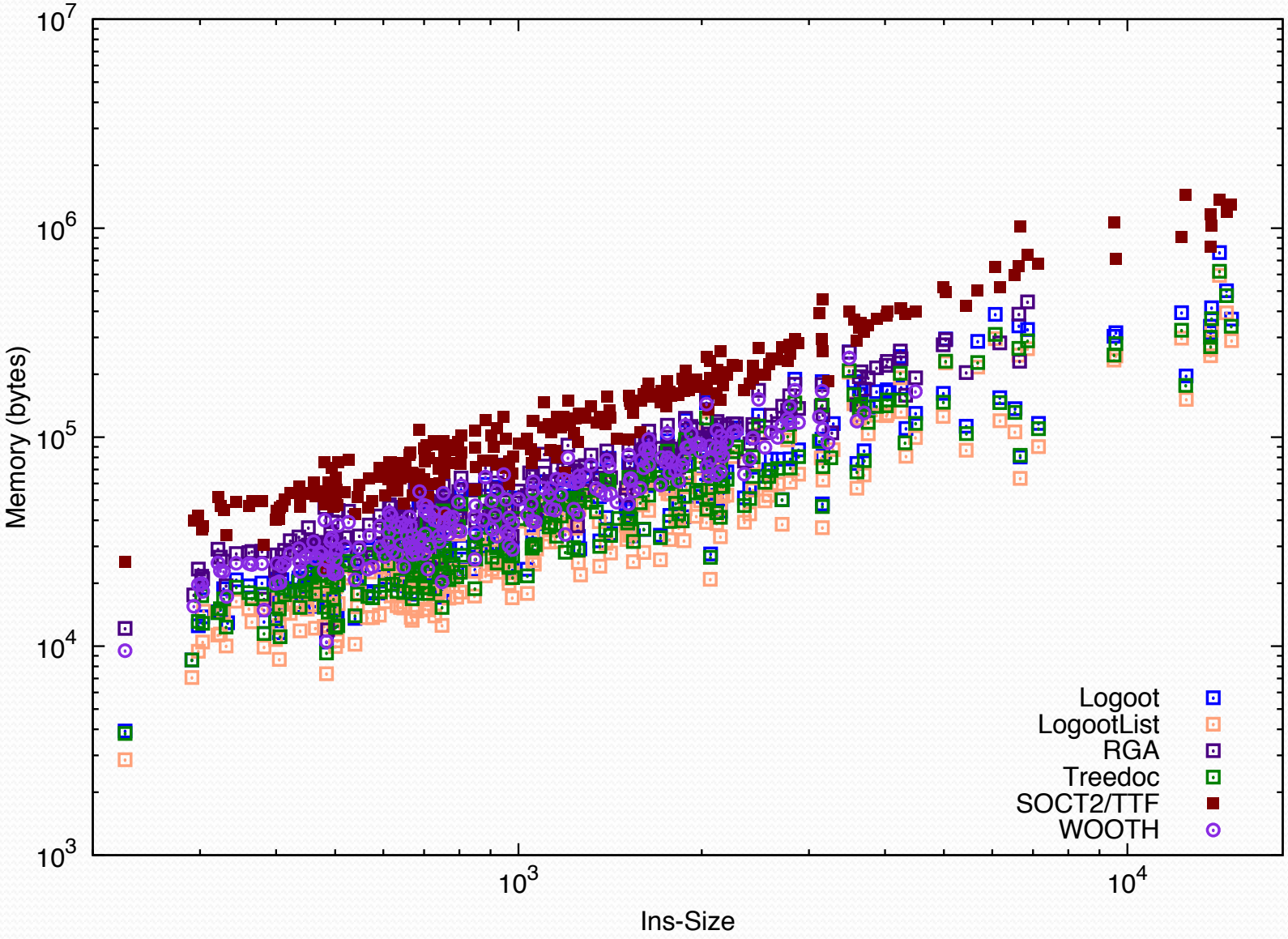
Remote operations : del size



Memory : merge number



Memory : merge number



Two experiments lead to two different results (to be refined)

Algorithms	Sim local	Sim remote	Sim mem	Git local	Git remote	Git mem
Treedoc Vs. Logoot	Treedoc	Treedoc	draw	Logoot	draw	draw
Logoot Vs. Logoot List	draw	draw	List	Logoot	draw	List
WOOTH Vs. RGA	WOOTH	RGA	WOOTH	?	WOOTH	?
…	…	…	…	…	…	…

Conclusions (to be refined)

- CRDTs scale !
- Applications with lot of remote operations
  - RGA and WOOTH
- Memory criteria and small copy/past
  - LogootList
- Large copy/past
  - LogootSplit
  - Treedoc with block ?
- Other case
  - Real-time : TreeDoc
  - Asychronous : Logoot