# Hazelcast framework

Rades Ana-Stefania

ana.rades@gmail.com

# Contents

# Hazelcast framework -Introduction-

- Open source clustering and highly scalable data distribution platform
- It is pure Java
- uses multicast or TCP/IP
- backup so if one node fails, no data will be lost

# Hazelcast framework -features-

- Distributed implementations of java.util.{Queue, Set, List, Map}
- Distributed implementation of java.util.concurrent.ExecutorService
- Distributed implementation of java.util.concurrency.locks.Lock
- Distributed Topic for publish/subscribe messaging
- Distributed listeners and events

# Hazelcast framework
# -Distributed data structures-

- Data in the cluster is almost evenly distributed across all nodes.
- If a member goes down, its backup replica prevents data loss
- There is no single cluster master
- When a new node joins the cluster it will eventually become a new partition

# Hazelcast framework
# -distributed topic,backups-

- Messages are ordered
- Distributed maps have 1 backup
- Backup operations are *synchronous(map.put(key,value))*
- *One backup problem*

# Hazelcast framework -persistence and query-

- load and store the distributed map entries from/to a persistent datastore
- store the entries synchronously (write-through)
- store the entries asynchronously (write-behind)
- distributed queries on the distributed map

# -Cluster monitoring tool-

- monitor your running Hazelcast cluster
- Hazelcast as of 1.8.1 version
- Cluster members
- Size&Memory
- Displays how your entries are partitioned among member
- How many entries and backups each member owns

# Distributed Execution

- java.util.concurrent.ExecutorService

Execute your code in cluster:

- on a specific cluster member you choose.
- on the member owning the key you choose.
- on the member Hazelcast will pick.
- on all or subset of the cluster members.

# Key based Distributed Executions

- Usual execution involves 4 distribuited operations
Hazelcast reduces to only one

- Send a Callable task to the member owning the key, clusterId.

- Callable does the deletion of the order right there and returns with the remaining order count.

- Upon completion of the Callable task, return the result (remaining order count).

# Execution Cancellation
# Execution Callback

- ability to cancel any execution
- get notified when the execution is done

# Testing and comparing

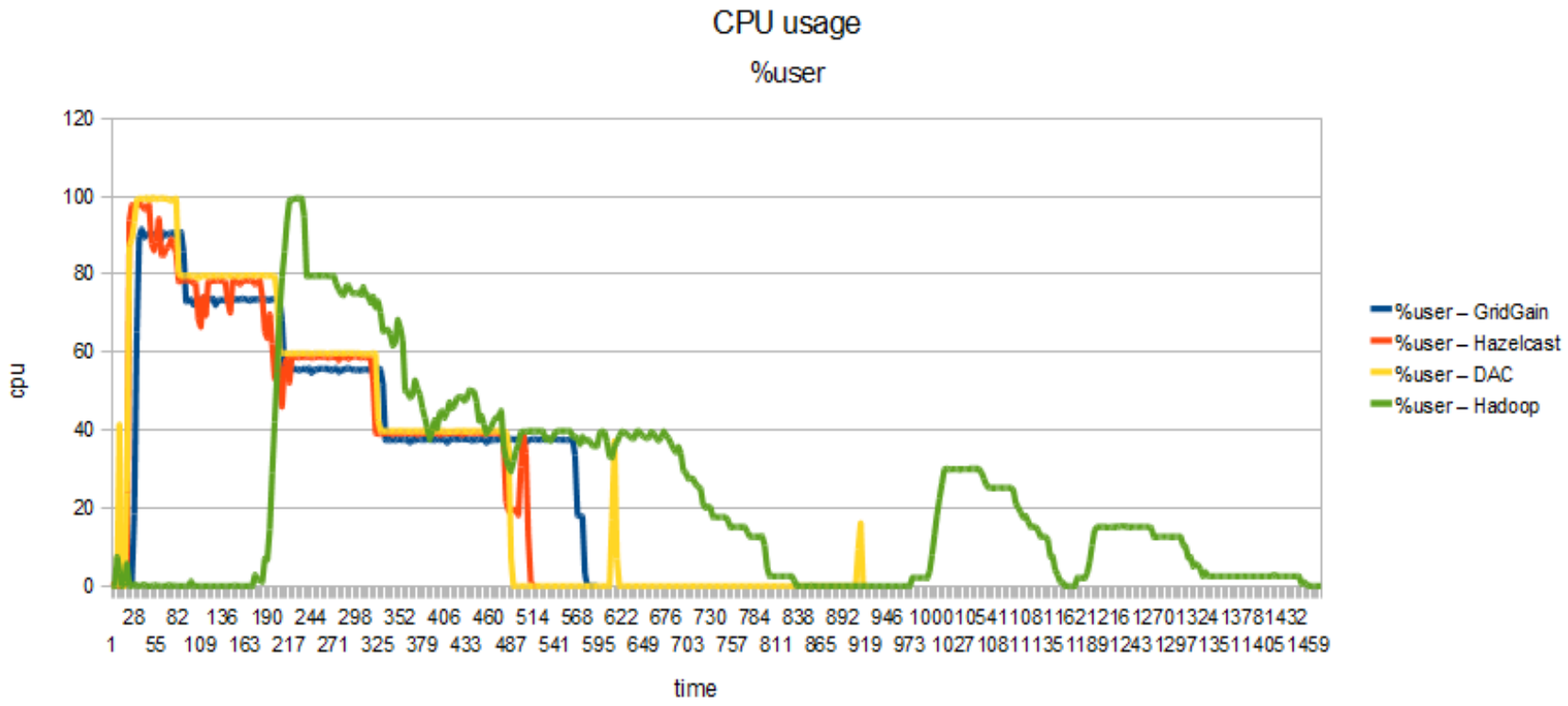
We have simulated node failures during our tests in the following order:

- intel4 went down after 60 seconds from the beginning of computations
- intel3 went down after 180 seconds from the beginning of computations
- intel2 went down after 300 seconds from the beginning of computations

All tests were repeated ten times in order to avoid measuring error.

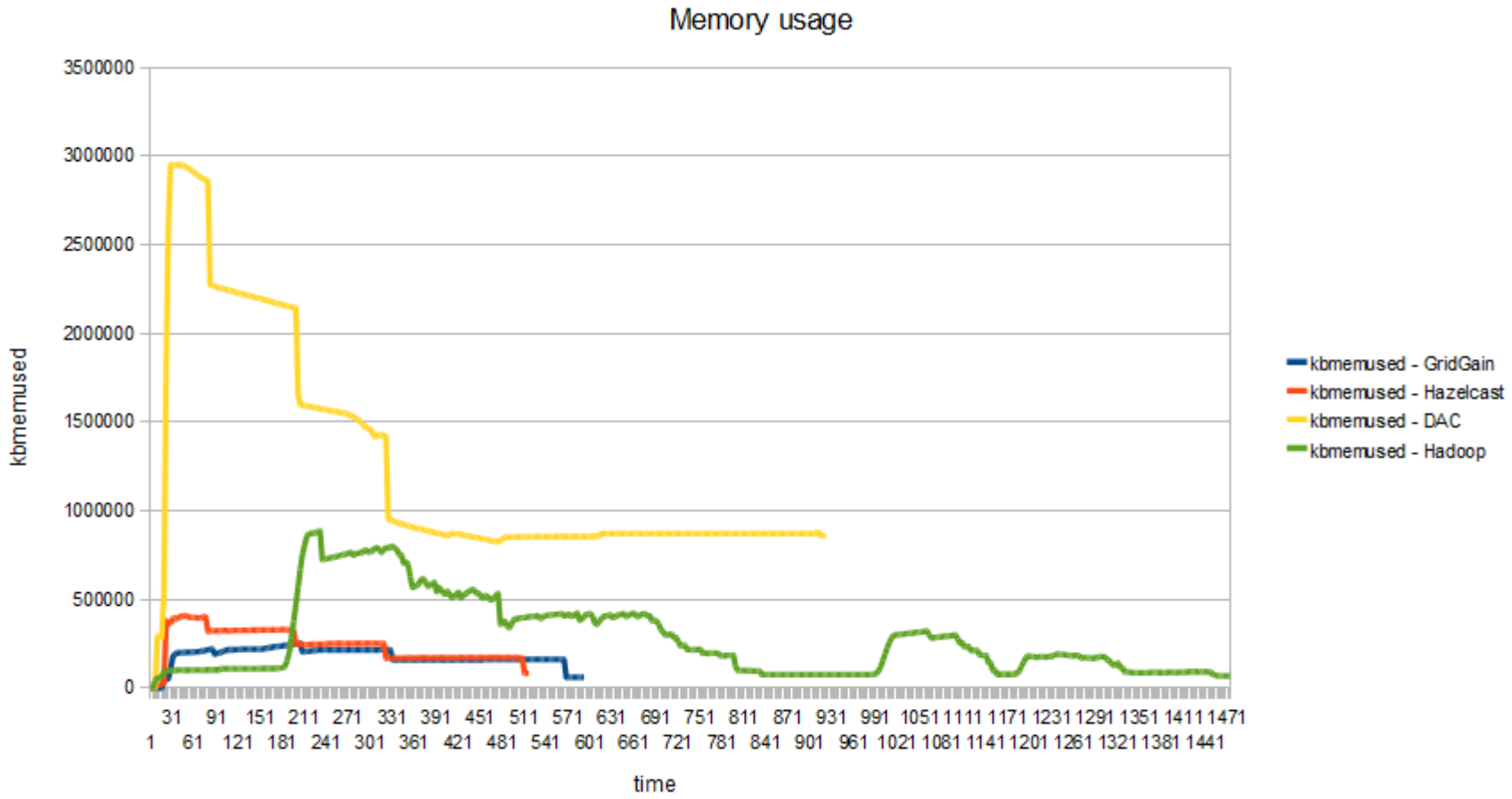
# CPU

Average CPU usage (%user) gathered on all machines:



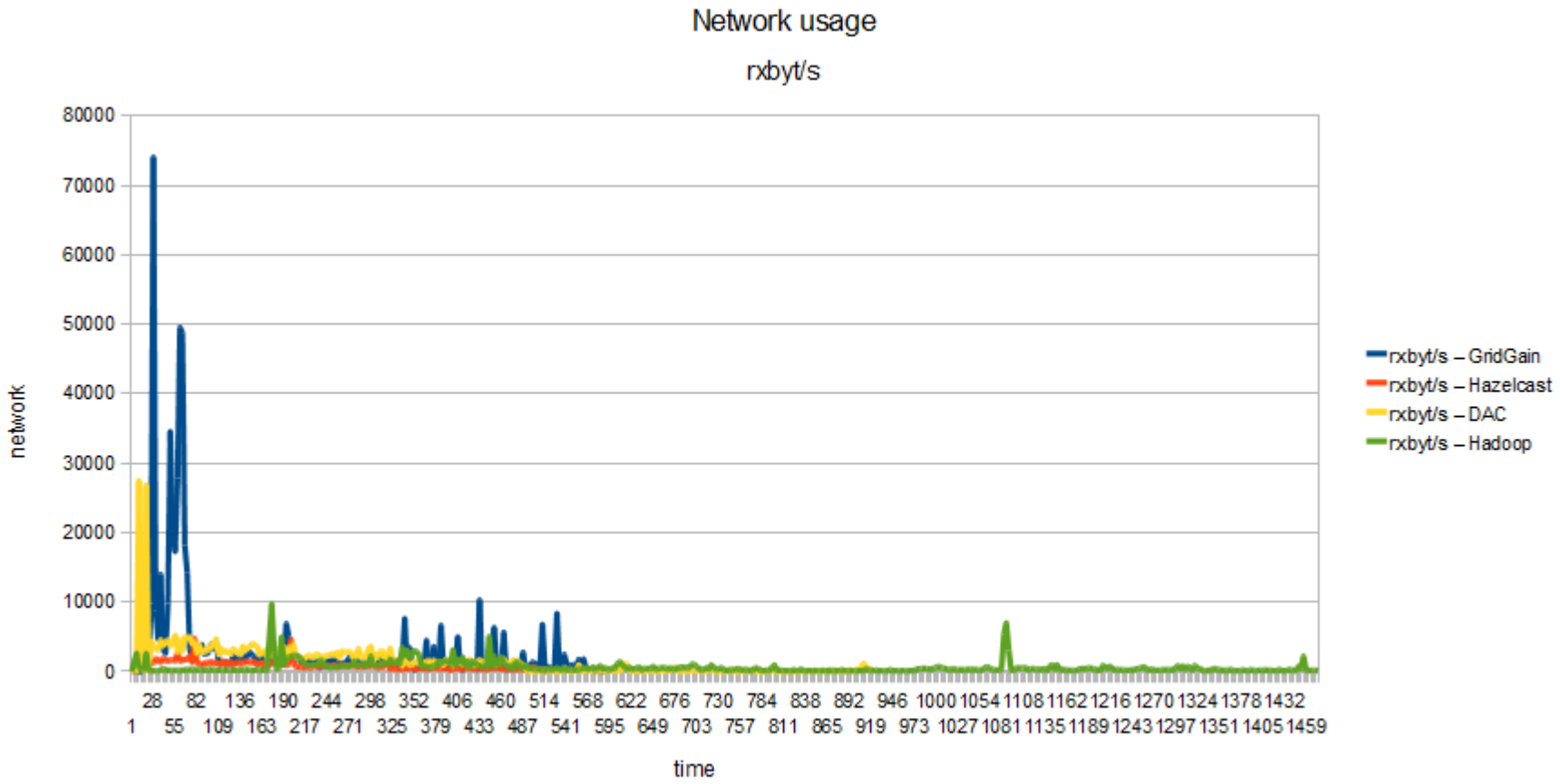
# Memory

Average memory usage gathered on all machines:



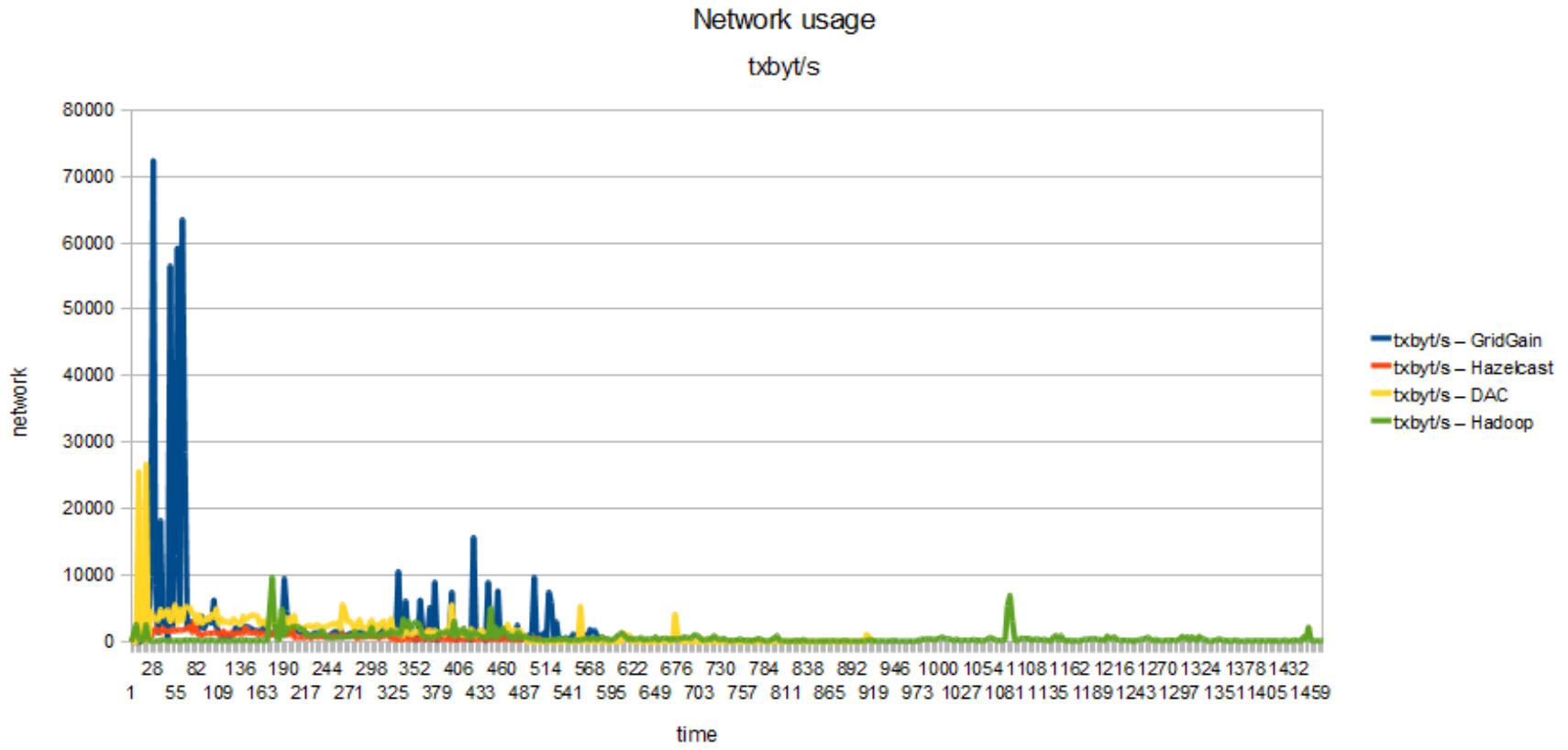
# Network

Average network usage (received bytes/s) gathered on all machines:



# Network

Average network usage (transmitted bytes/s) gathered on all machines:



# Conclusions

- Hazelcast and GridGain are the best choice for an easily-parallelized, low-data, CPU-intensive tasks.

- Hazelcast consumes the smallest amount of CPU and network bandwidth