

Summary of *etcd* benchmarks

*Exploring etcd as a potential back-end
for the O² Configuration module*

Pascal Boeschoten

Outline

- Objective
- Test setup
- Results
- Next steps
- Notes

Objective

- Evaluate the performance of *etcd* as a store for configuration parameters
 - Using representative workloads
 - In “single-server” and “cluster” modes
 - With multiple parameter data structures

Test setup – Workload

- At start of run, thousands of processes retrieve parameters
 - Large burst of GET requests from clients
 - High load on server(s) for a short period

Metric	Value
Number of client processes	7k to 70k
Parameters per client process	100
Average size per parameter	100 bytes
Total configuration data volume	700 MB

Key workload estimations

Test setup – Parameter organization

- The structure of the data will vary per client process, thus various approaches were explored

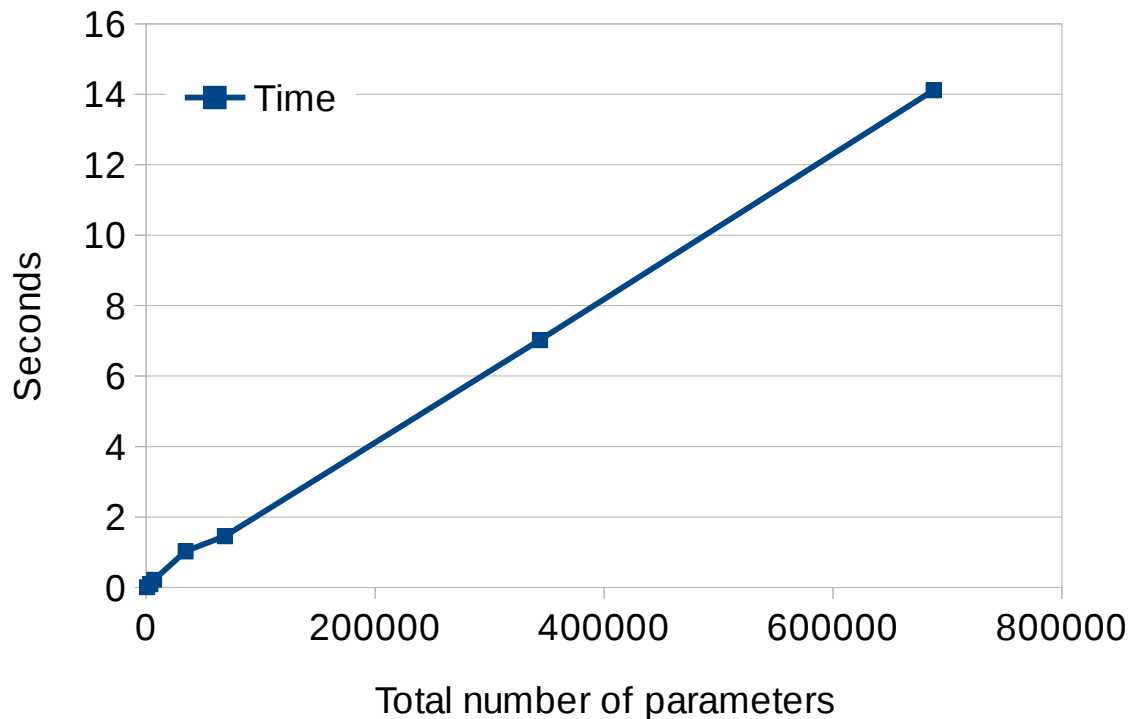
Name	Description	Query
Fragmented	Separate key-value pairs	One GET per parameter
Blob	Key-value pairs stored in single blob	One GET for whole blob
Flat	Key-value pairs grouped in one directory	Recursive GET
Tree	Key-value pairs grouped in nested directories, structured like a binary tree, with five parameters per directory	Recursive GET

Results – Single server

- Fragmented structure: low performance
- Others: much better

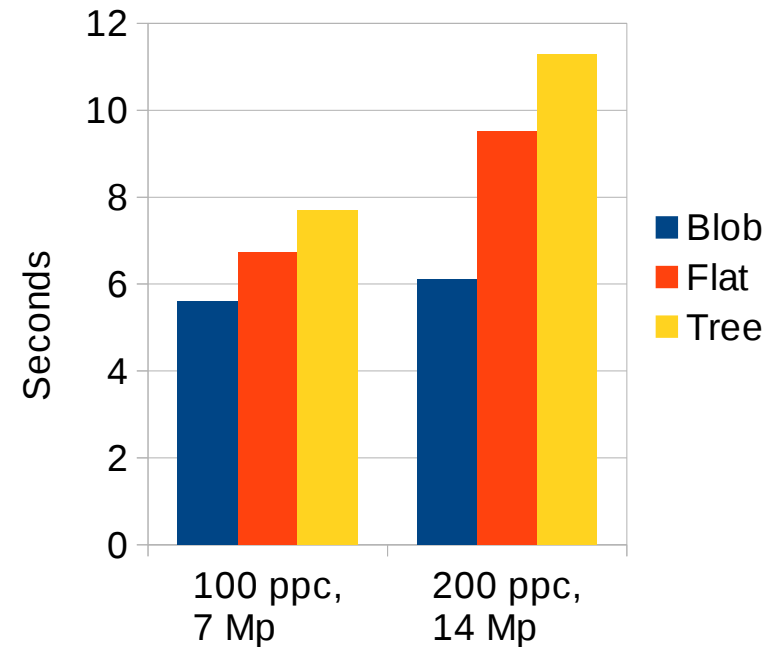
Time to pull parameters, fragmented structure

1 server, 43 client hosts, 16 processes per client



Time to pull parameters, varying structures

1 server, 42 client hosts



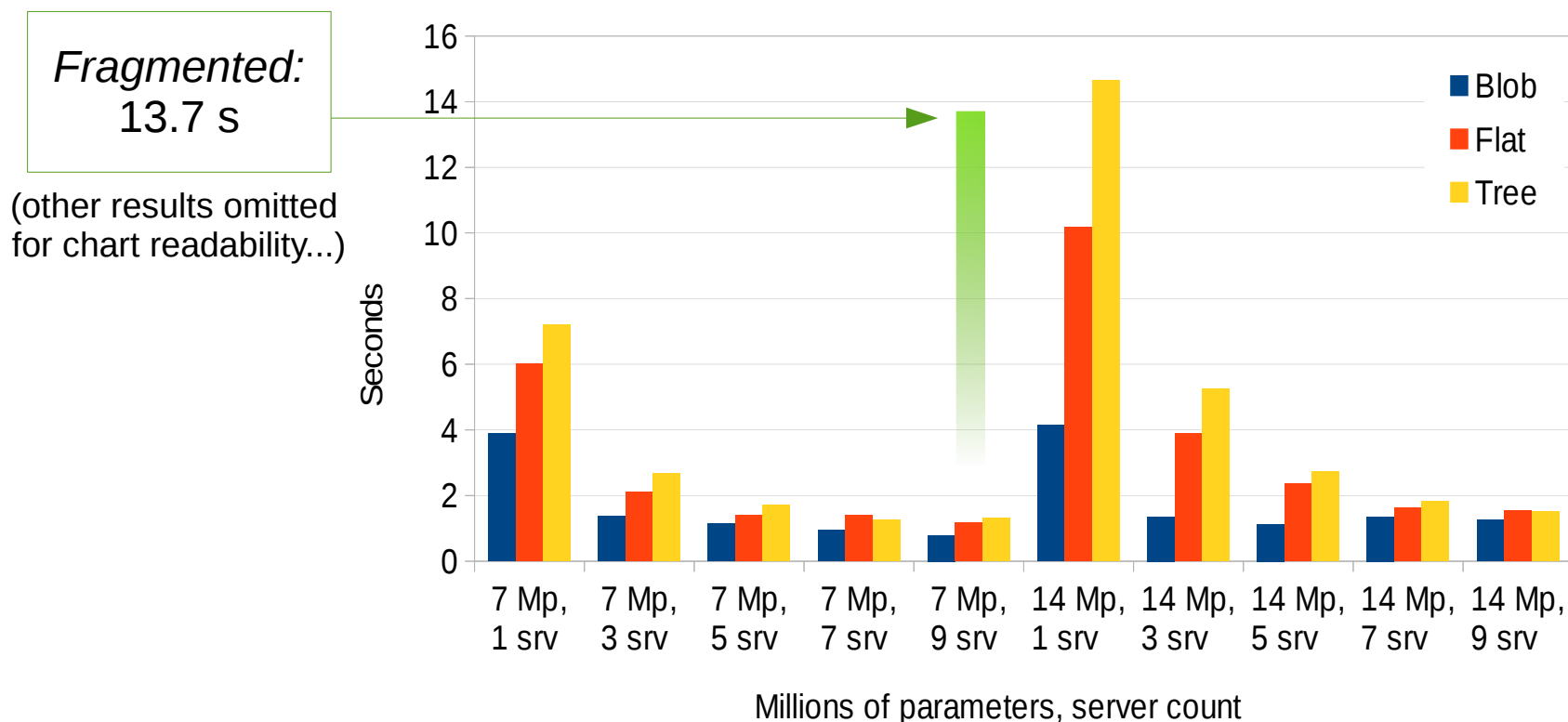
Parameters per process, total parameters

Results – Cluster

- With higher server counts, performance quickly improves to acceptable levels (except for *fragmented*)

Time to pull parameters, various data organizations and cluster sizes

70k client processes across 42 hosts

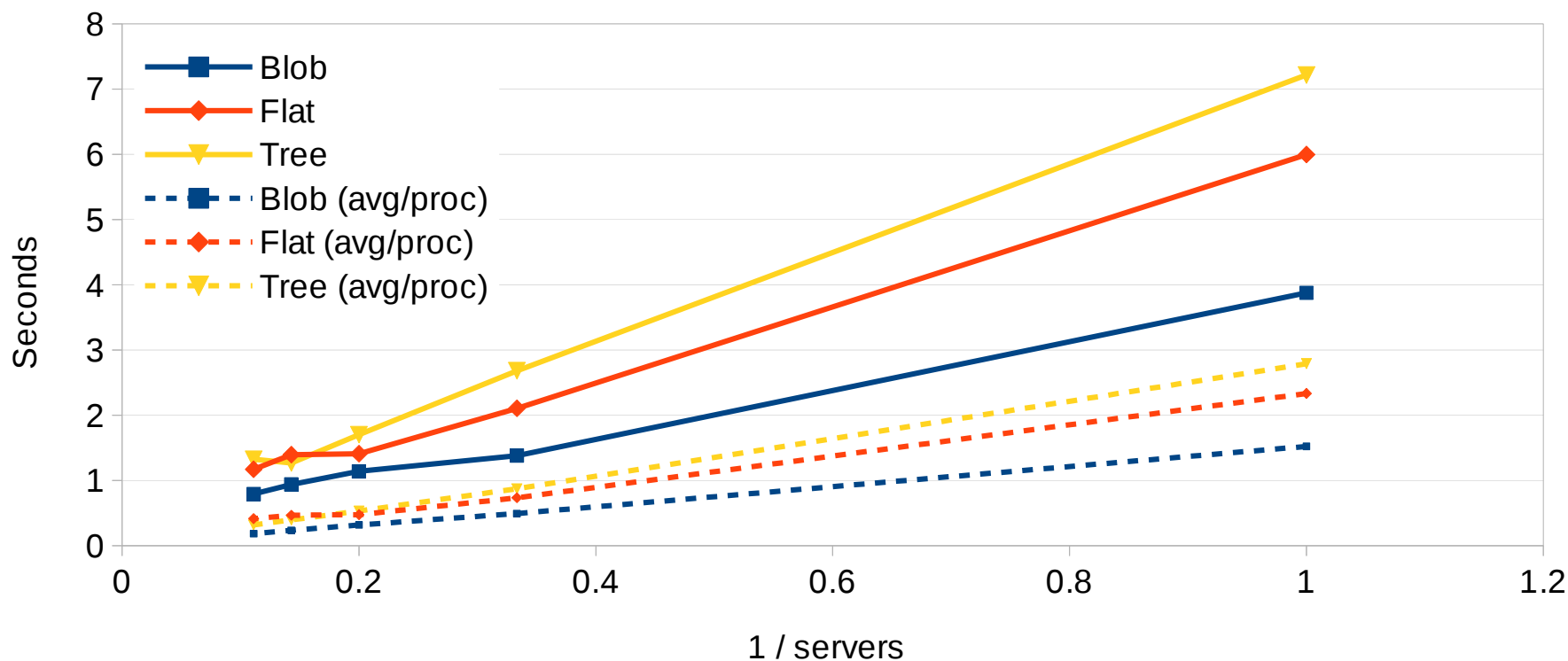


Results – Cluster

- Roughly linear relationship between server count and performance → good!

Time to pull 7M parameters as a function of $1/\text{servers}$

70k client processes across 42 hosts



Results – JSON overhead

- Significant JSON response size overhead for *flat* and *tree*
 - *Tree* overhead is “exponential” ●
- A newer (not yet stable) *etcd* API uses Google Protocol Buffers for serialization
 - The binary format might reduce overhead

Parameter structure	Usable size (bytes)	JSON response size (bytes)	Overhead	
100 blob	10000	10892	8.92 %	
100 flat	10000	18098	81.0 %	
100 tree	10000	21214	112 %	←
200 blob	20000	21796	8.92 %	
200 flat	20000	36198	81.0 %	
200 tree	20000	44214	121 %	←

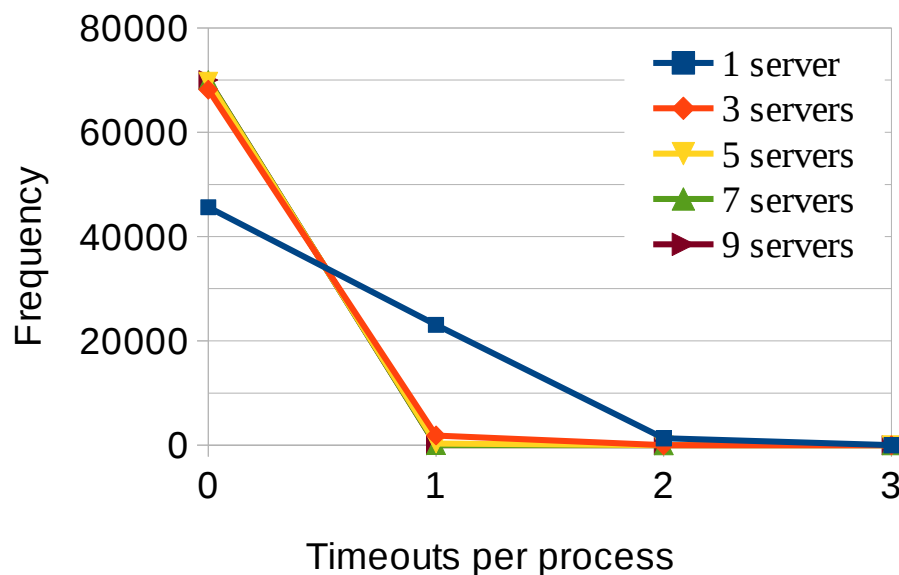
Results – Timeouts

- Connection timeouts occur when server is overloaded
 - Harmful to performance with 1 server
 - Practically disappear with ≥ 3 servers

Of tested structures,
100 blob is “best case”,
200 tree “worst case”
(not counting *fragmented*)

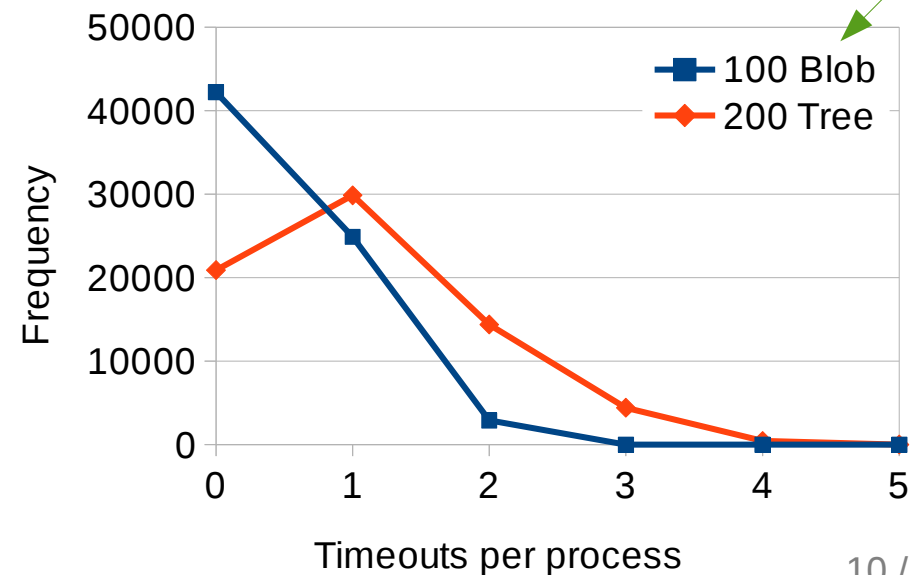
Frequency of amount of timeouts per process

7M parameters in "200 tree" structure, 42 clients, 70k processes, 10s timeout



Frequency of amount of timeouts per process

9 servers, 42 clients, 70k processes, 250ms timeout

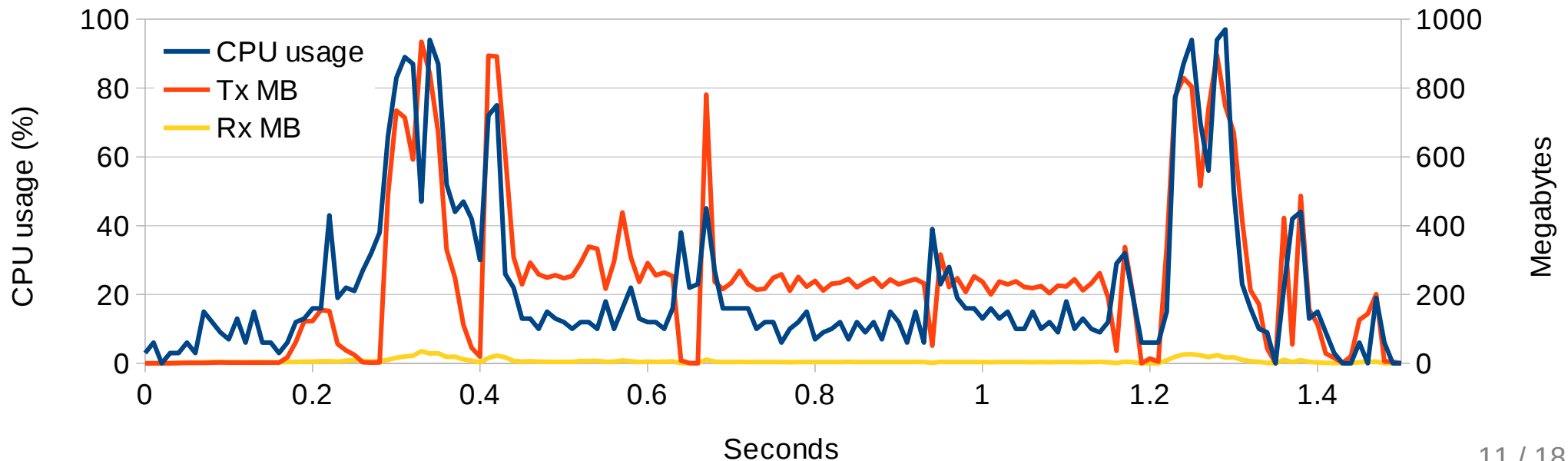


Results – Resource usage

- Not ideal, but OK
 - Large spikes
 - Overall underutilization

Etcd server resource usage with "200 tree" structure

9 servers, 42 client hosts, 70k processes, 14M parameters

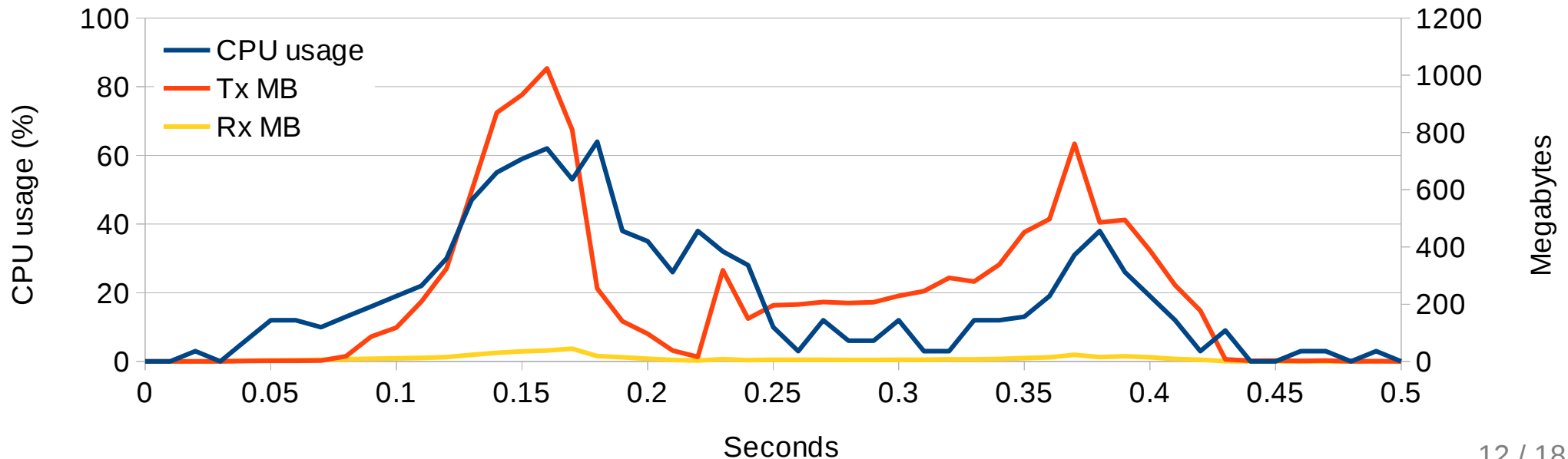


Results – Resource usage (2)

- Spiking apparent even during very short runs
- Cause unknown
- But is tuning necessary? “Good enough” already?

Etcd server resource usage with "200 blob" structure

9 servers, 42 client hosts, 70k processes, 14M parameters

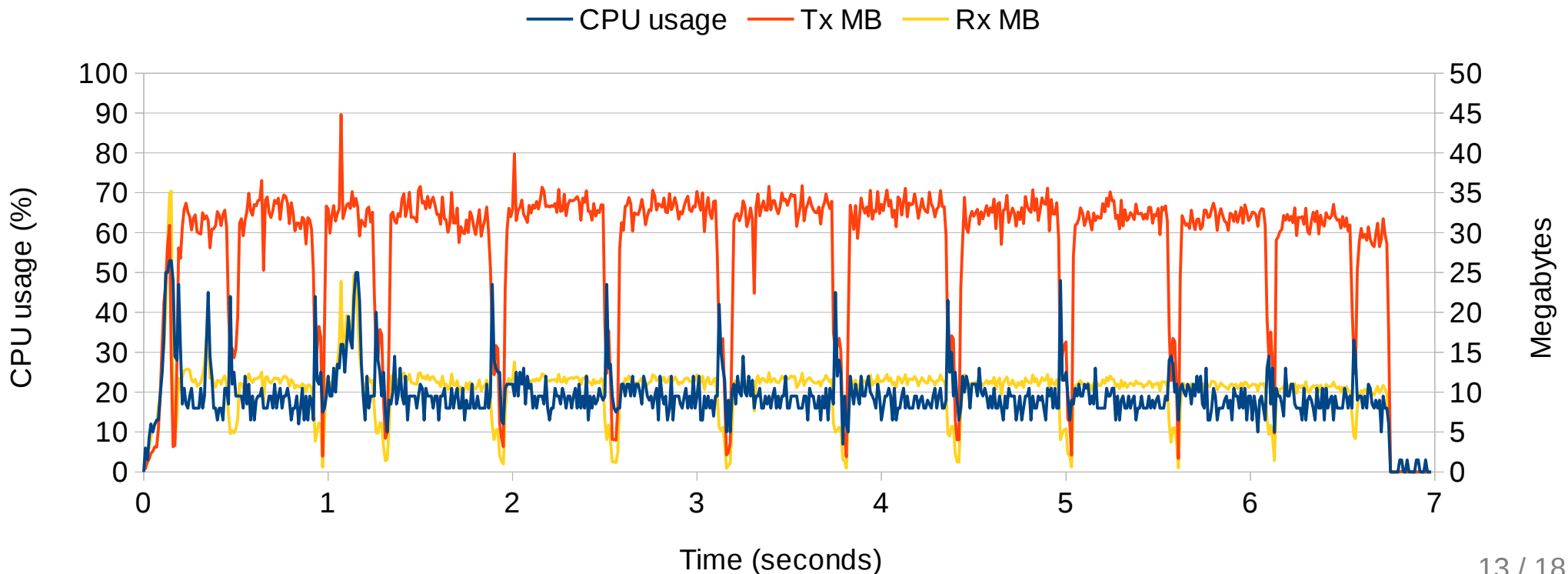


Results – Resource usage (3)

- Resource usage with *fragmented* structure:
 - Similar CPU, much lower transmission.

Etcd server resource usage with "50 fragmented" structure

9 servers, 42 client hosts, 70k processes, 3.5M parameters



Next steps

- Investigate a structure in between *fragmented* and *tree*
 - Multiple recursive GETs per process
- Multiple *etcd* instances per physical server
 - May improve resource utilization
- Compare with other back-ends
 - Consul, Zookeeper?

Notes

- More details available attached to JIRA ticket:
<https://alice.its.cern.ch/jira/browse/OCONF-3>

Reference slides

Machine specifications

Hostname	Specifications
aidrefpc001/45	2x E5520, 8 threads, 2.26 GHz
aidrefsrv10	2x E5-2650 v2, 16 threads, 2.6 GHz
aido2qc10/13	2x E5-2640 v3, 16 threads, 2.6 GHz
aido2qc40/43	
aido2web1	

JSON snippet

Formatted snippet of a “200 tree” JSON response

```
{
  "action": "get",
  "node": {
    "key": "/test/tree100",
    "dir": true,
    "nodes": [
      {
        "key": "/test/tree100/key3",
        "value": "value00...03",
        "modifiedIndex": 103,
        "createdIndex": 103
      },
      {
        "key": "/test/tree100/key4",
        "value": "value00...04",
        "modifiedIndex": 104,
        "createdIndex": 104
      },
      {
        "key": "/test/tree100/dirA",
        "dir": true,
        "nodes": [
          {
            "key": "/test/tree100/dirA/key8",
            "value": "value00...08",
            "modifiedIndex": 78,
            "createdIndex": 78
          },
          ...
        ]
      }
    ]
  }
}
```