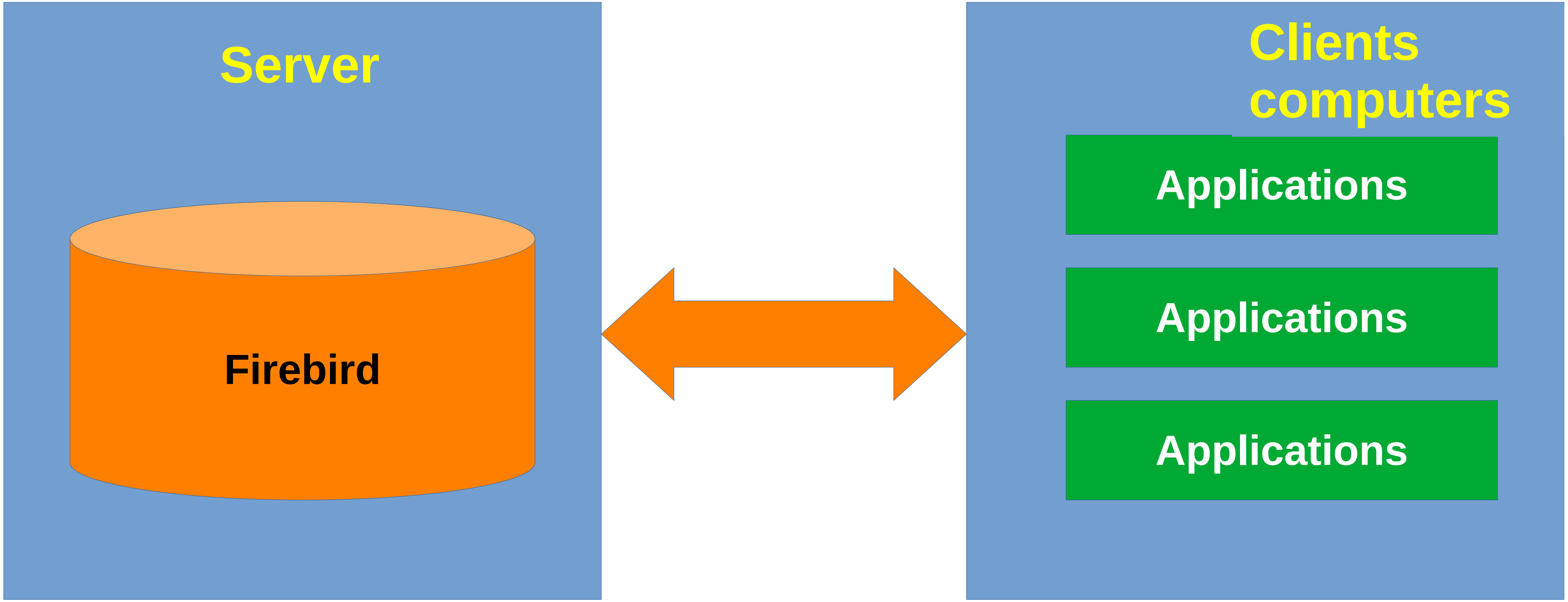
Firebird in the cloud: SaaS and more

Alexey Kovyazin, Firebird Foundation

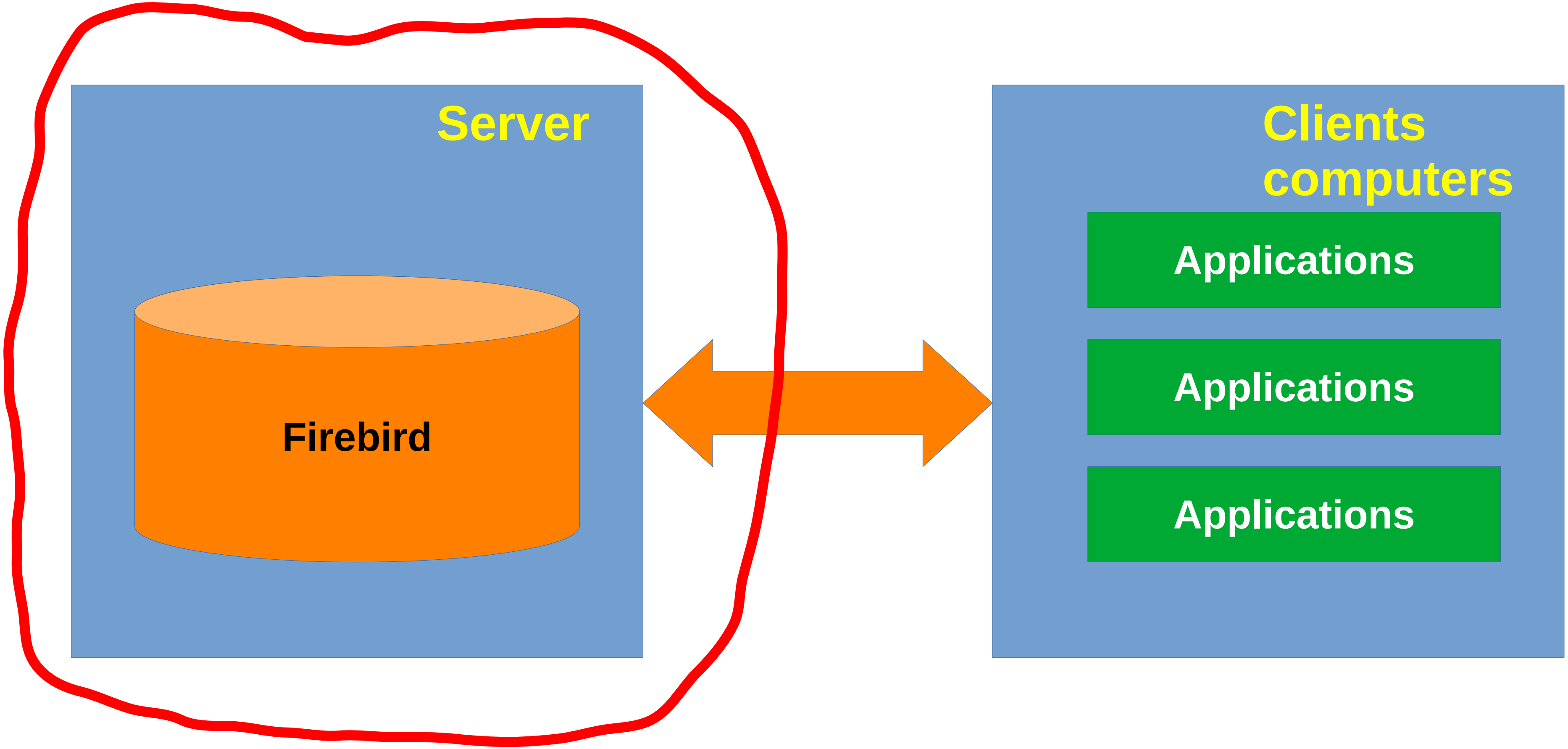
Agenda

- Scenarios
- Problems with cloud solutions:
  - Performance
  - Network
  - Balance
  - Security
- Testing clouds

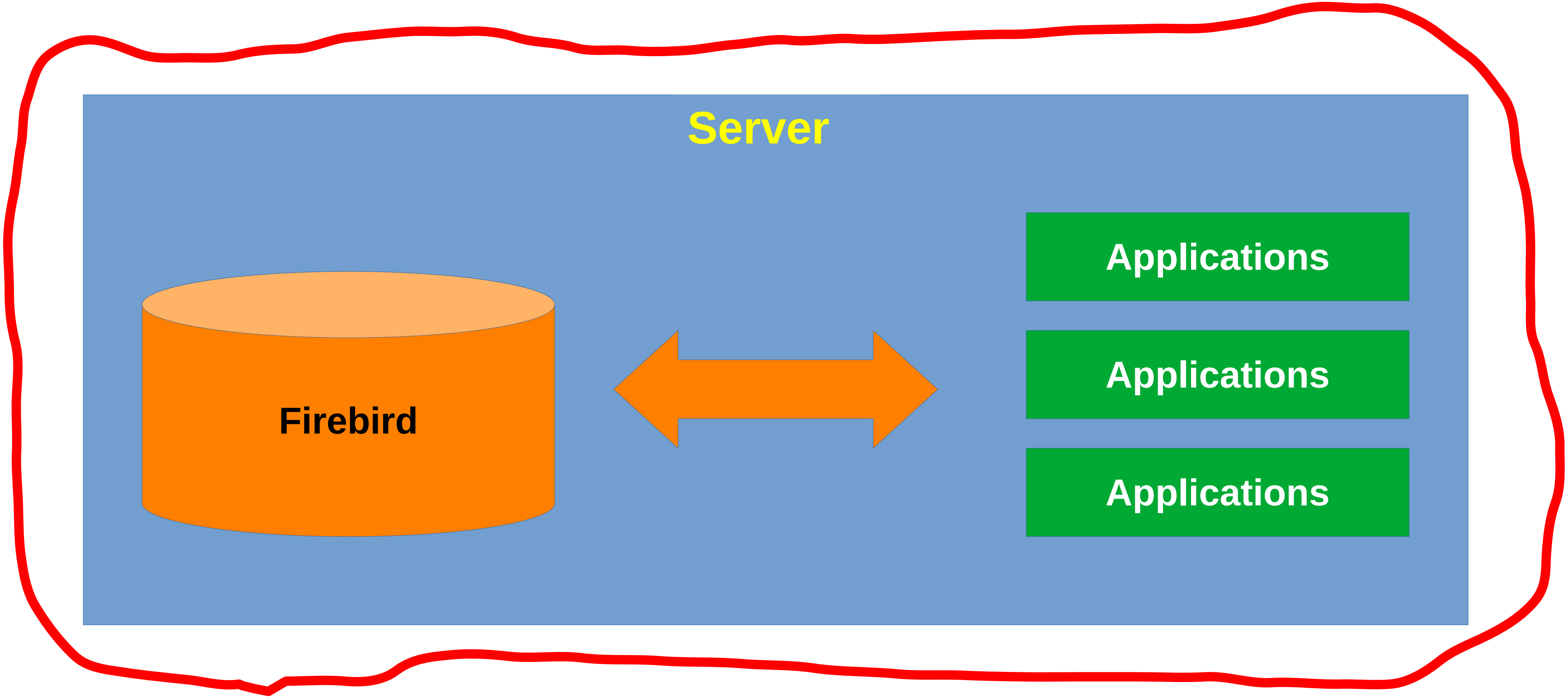
Scenario 1: traditional client-server



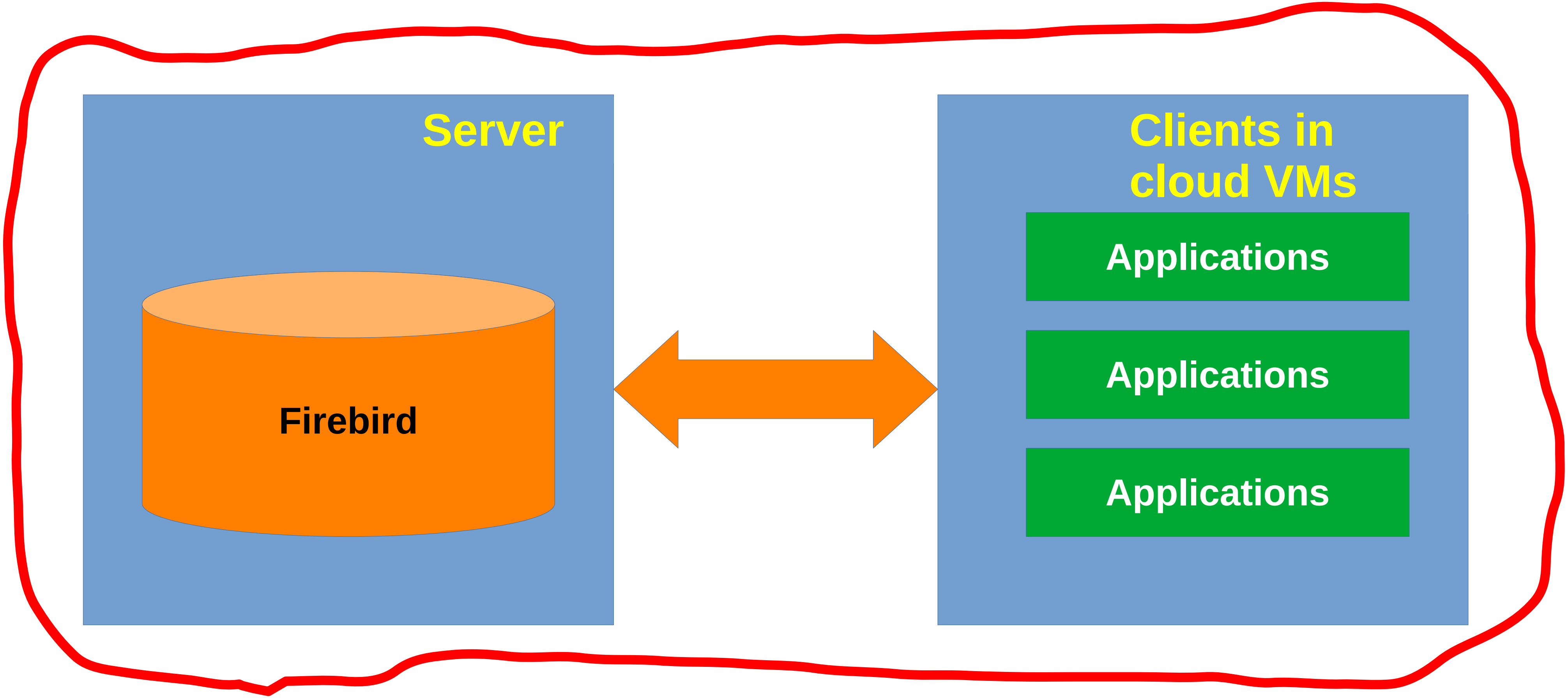
1.1. Only server resides in the cloud



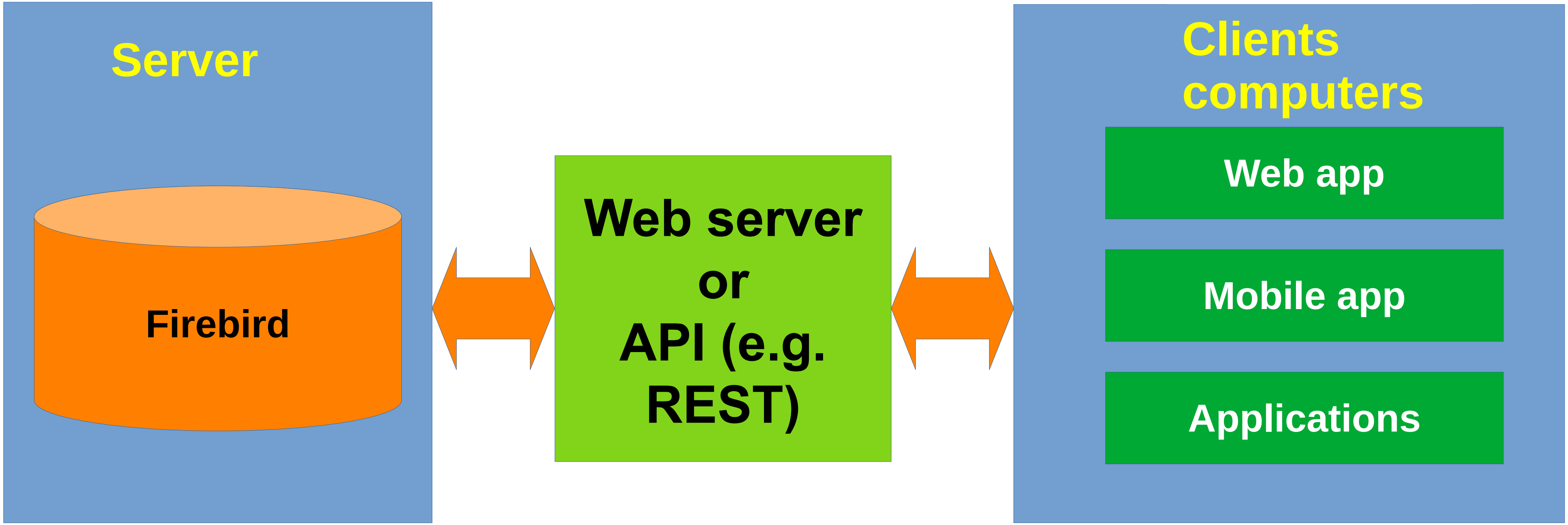
1.2. Server and clients are on the same cloud machine



1.3. Scenario: clients and server in the cloud



Scenario 2: middleware



Server

Firebird

Web server or API (e.g. REST)

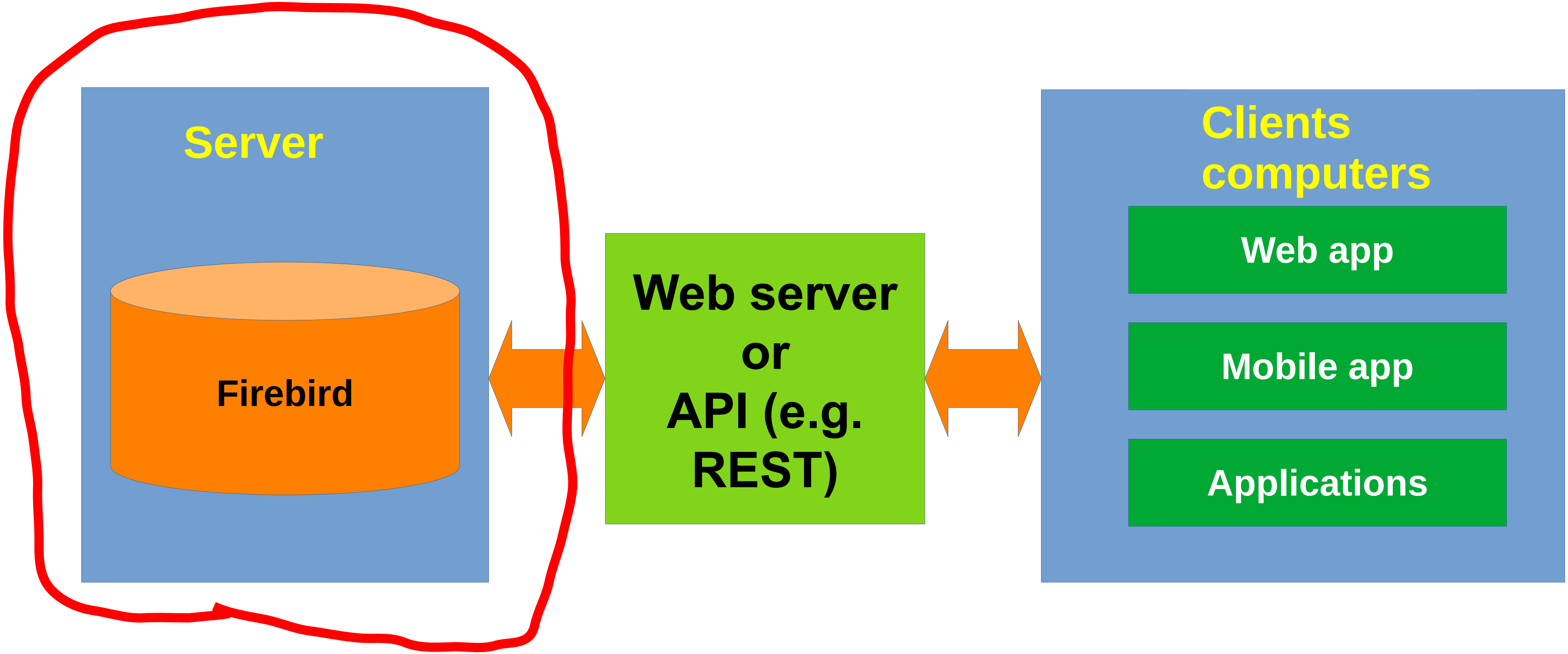
Clients computers

Web app

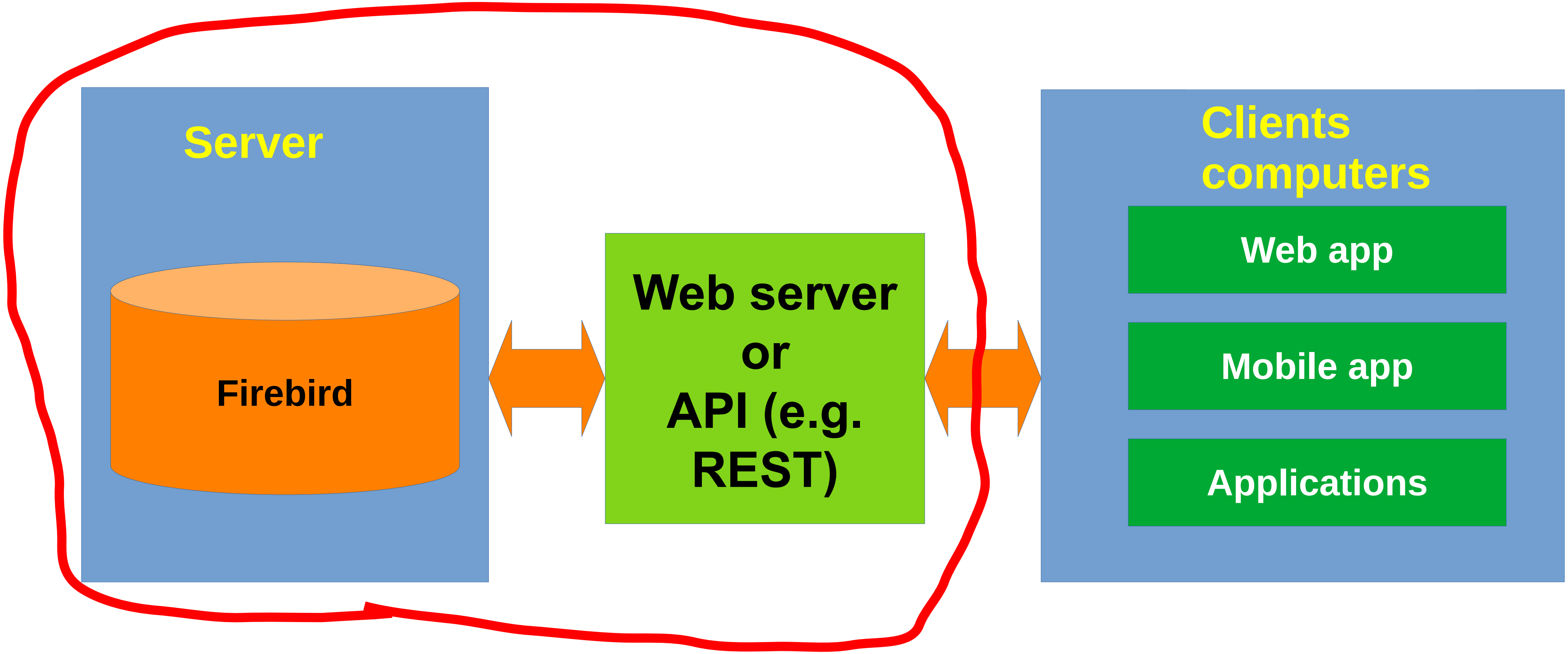
Mobile app

Applications

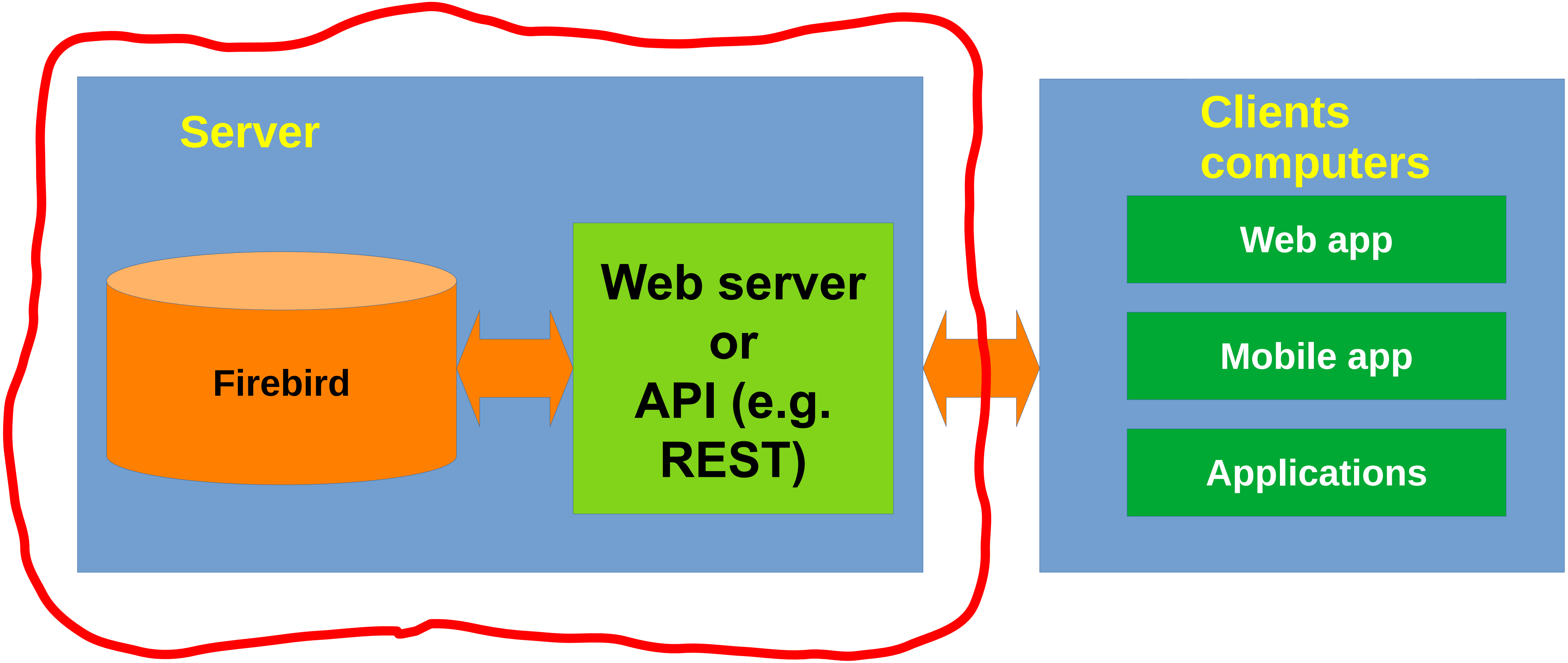
2.1: Only server in the cloud



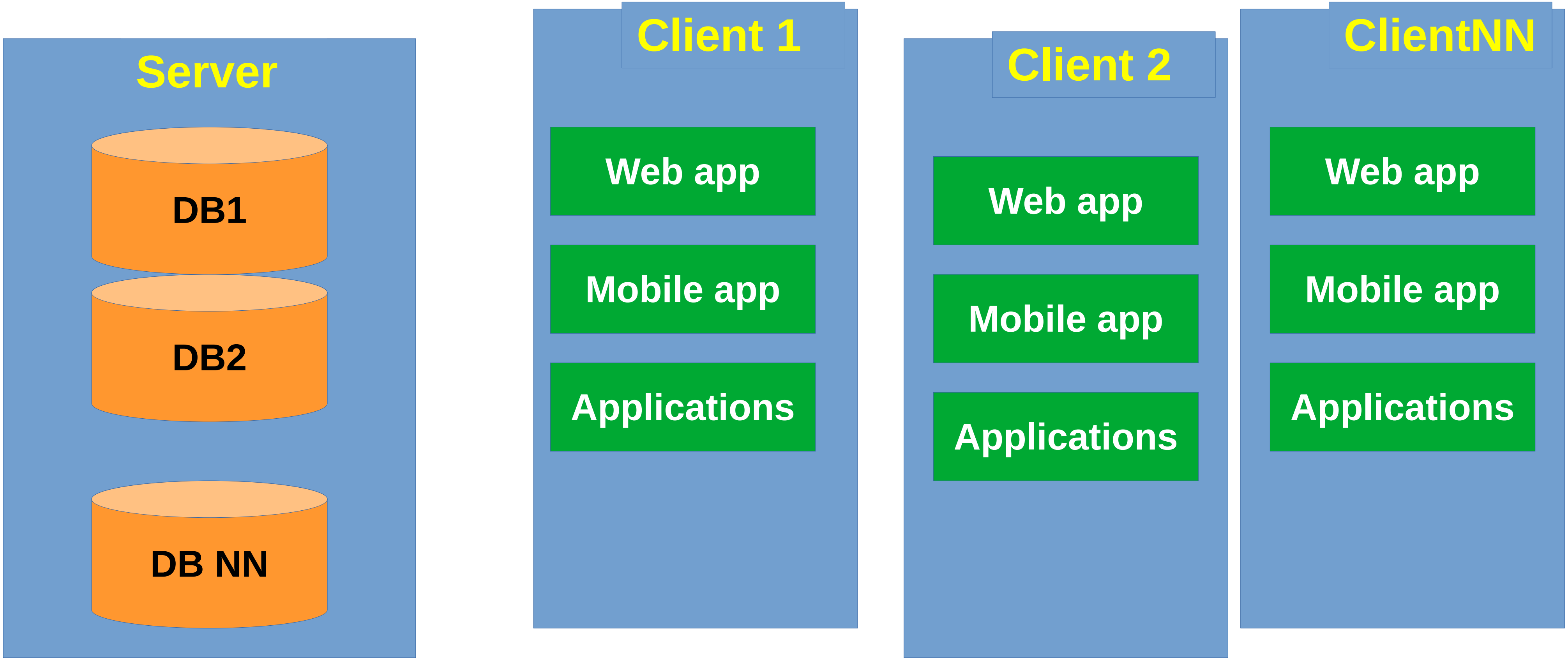
2.2. Database and middleware in the cloud



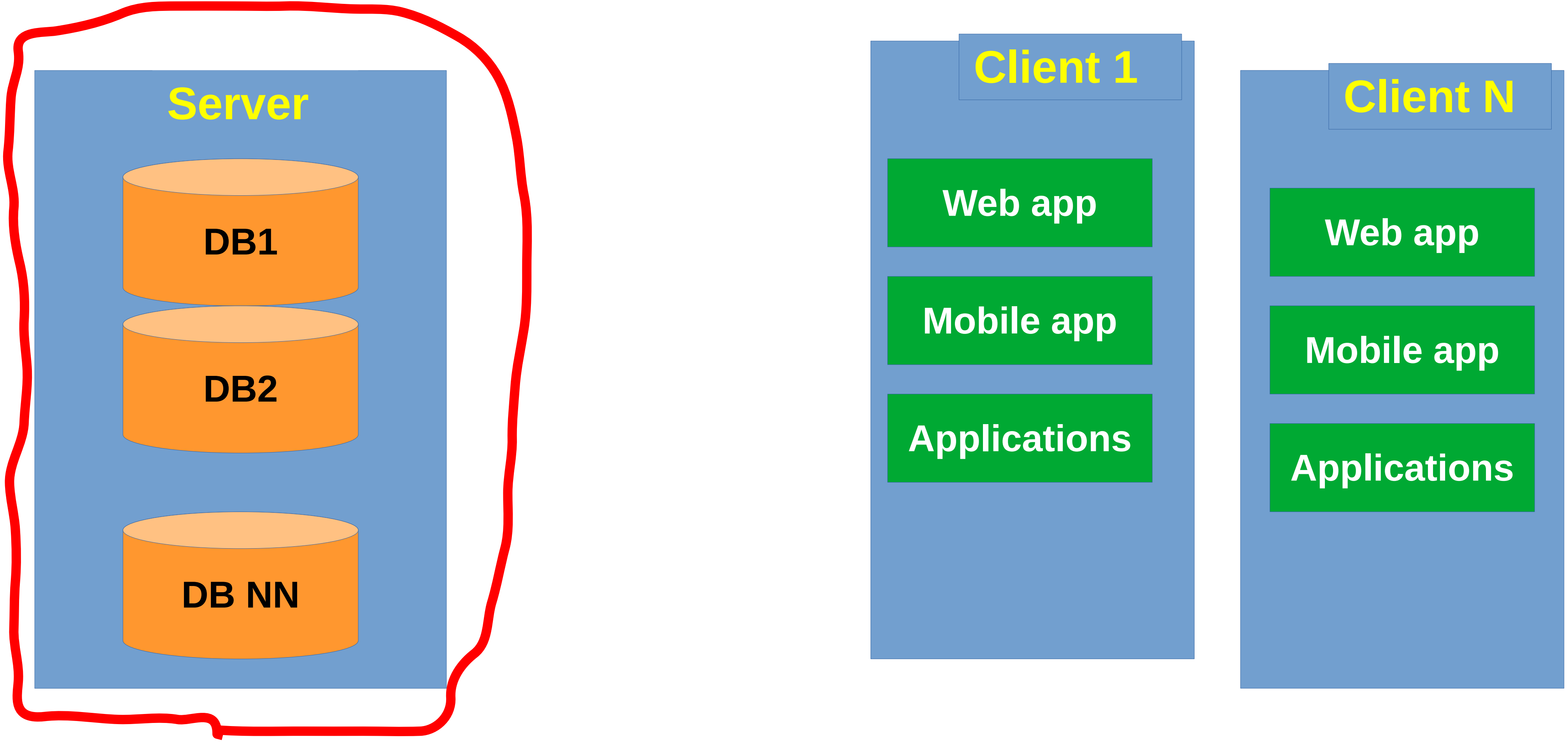
2.3. Database and middleware on the same server



Database per client (SaaS or hosting)



3.1.Database per client



Server

DB1

DB2

DB NN

Client 1

Web app

Mobile app

Applications

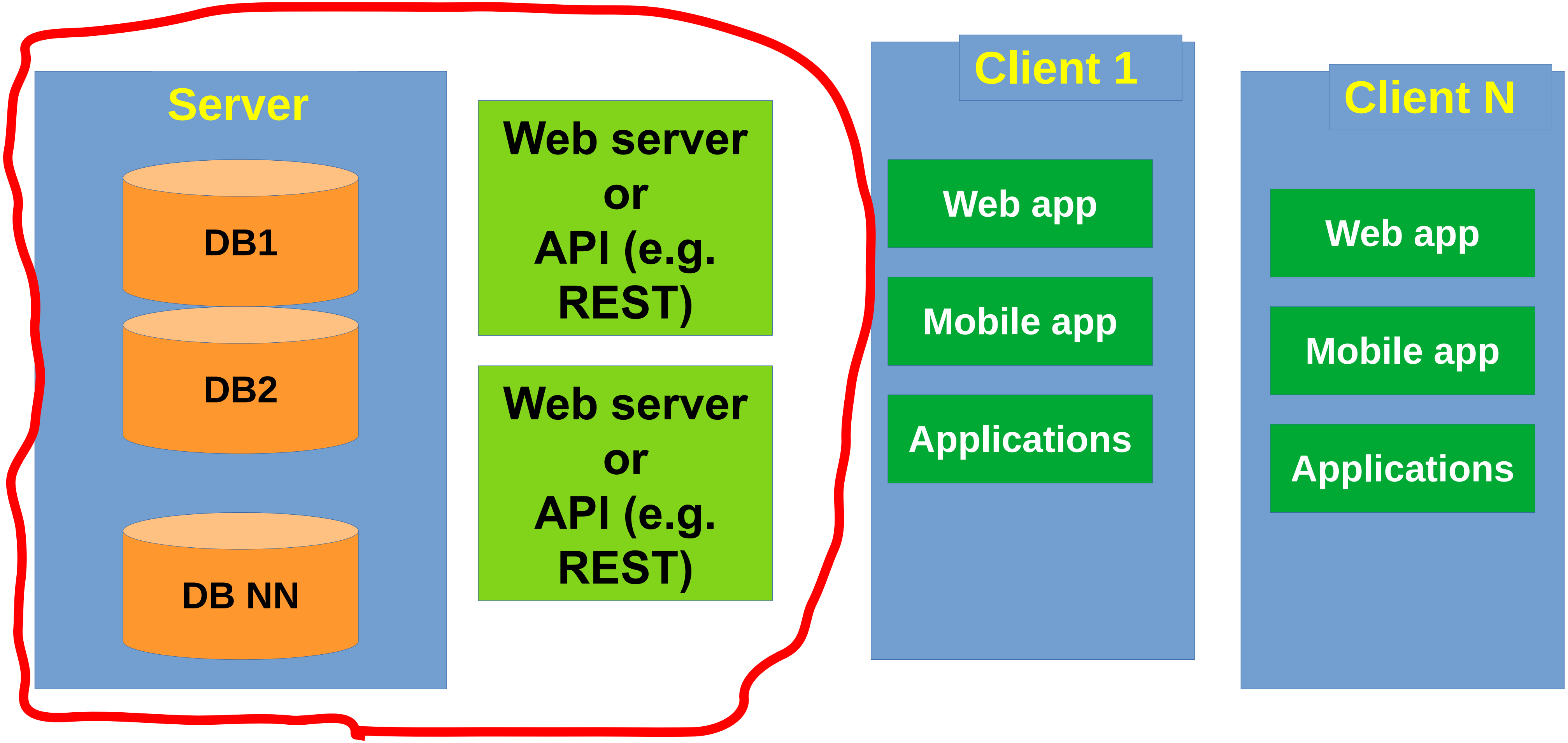
Client N

Web app

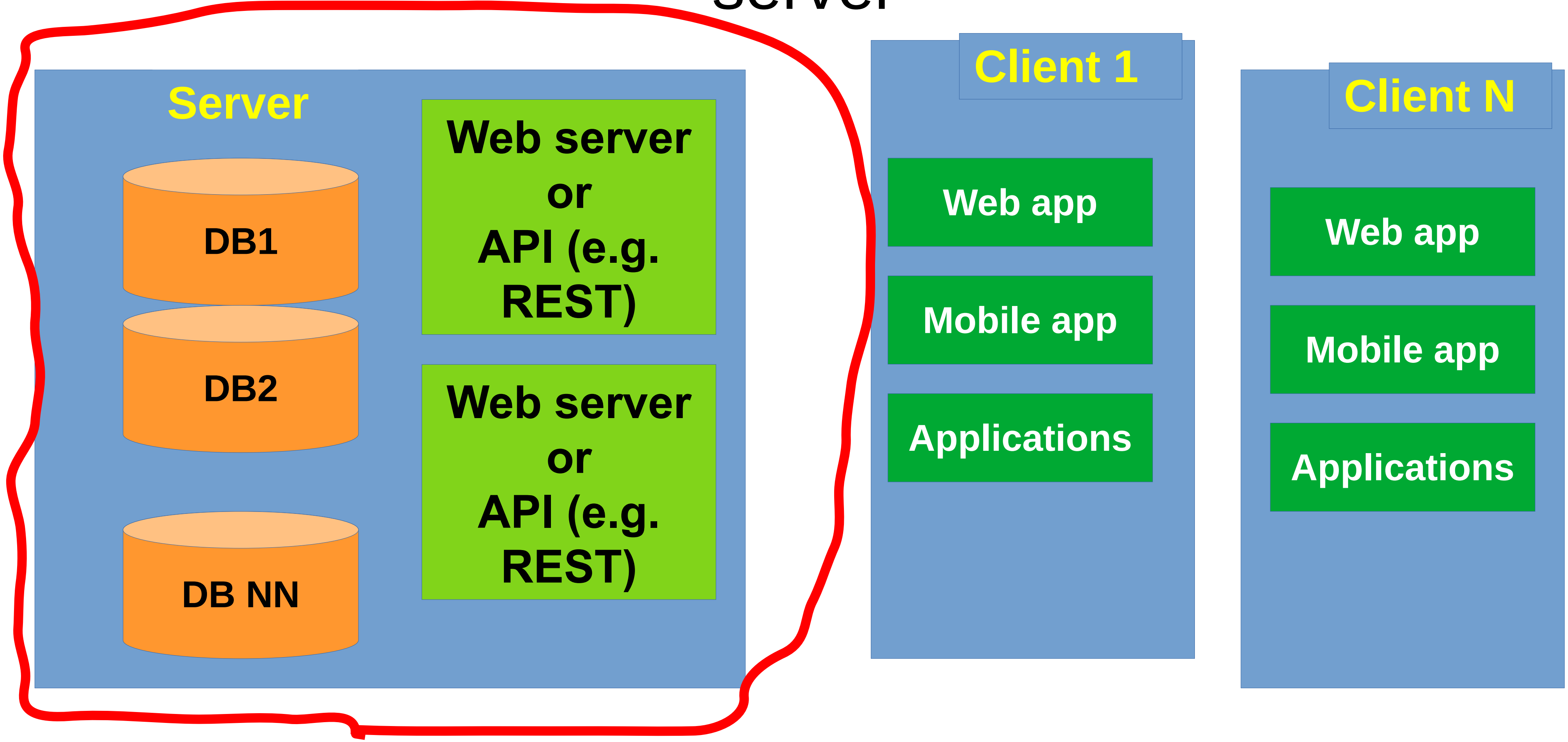
Mobile app

Applications

3.2: Database per client with middleware



3.3: Database per client with middleware on the same server



All scenarios

- Traditional Client-Server - few databases
  - Database in Cloud, Applications with direct access
  - Database in Cloud, Applications in cloud on RDP
  - Database and Applications together on the same server
- Client-Server with middleware – few databases
  - Database in Cloud, Middleware in another Cloud
  - Database and Middleware in the same cloud, different servers
  - Database and Middleware on the same server in cloud
- Many databases on the server
  - tighter resources, need to balance resources, more security problems

Cloud = VM + Network through Internet

Problems:

1) **VM** Level: Cloud and shared environment, there may be several VMs per host, VM always slower than host with physical server
    1) Virtual Machine overhead
    2) Shared storage systems
    3) Memory overcommit
2) **Network**: high latency, loss of packets, settings
3) **Balance**: how to divide resources between databases, applications, middleware, clients

Scenarios and their problems

- Traditional Client-Server - few databases
  - Database in Cloud, Applications with direct access — **VM**, **net**
  - Database in Cloud, Applications in cloud on RDP - **VM**
  - Database and Applications together on the same server - **VM, balance**
- Client-Server with middleware – few databases
  - Database in Cloud, Middleware in another Cloud — **VM, net**
  - Database and Middleware in the same cloud, different servers - VM
  - Database and Middleware on the same server in cloud — **VM. balance**
- Many databases on the server
  - tighter resources, need to balance resources, more security problems — **VM, net, balance**

Which scenario is better?

- Scenario depends on development history and client requirements
- If you plan to start from scratch
  - Database and Middleware on different servers

How to solve the problems

- VM
- Network
- Balance

VM level

- Choice of Firebird architecture
- Cloud-specific configurations
- Recommended Operating Systems
- File systems
- Query optimization: focus on reads/writes and fetches

Choice of Firebird architecture-1

- SuperServer - memory usage per database
  - great for few databases
  - Good option for <100 databases with 5+ connections per database
  - For Firebird version 3+
  - For environments with query monitoring

Choice of Firebird architecture-2

- SuperClassic – memory usage proportional to the number connection
  - The only option for v2.5 with good performance
  - For situations when number of connections is less than quantity of databases (3000 databases, but, on average 500 connections)

Choice of Firebird architecture-3

- Classic – like SuperClassic, separate processes, slightly slower
  - can use OS tools to kill connections with high resource consumption

Most common problems with architectures (beyond wrong configuration)

- SuperServer (v3+)
  - Firebird crash (due to wrong UDF, bugs) will stop all connections
  - without advanced monitoring difficult to identify which database uses many resources and disturbs others
- SuperClassic/Classic (v3+)
  - many connections will consume all memory. Generally needs more memory
  - lower performance than SuperServer

Cloud-specific configurations

- Deficient resources

  1) IO is almost always the first problem!

  2) Memory – second problem

  3) CPU – can be a problem in case of non-optimized queries and very frequent connections/transactions

How to create cloud configuration

- Start with Configuration Calculator https://cc.ib-aid.com
- Become Firebird Supporter to get access to Advanced Calculator
  - https://store.firebirdsql.org/

Operating Systems - Windows

- Windows Server
  - Use recent versions 2019+
  - keep drivers updated
- Power configurations
- - always maintain High performance
- Hibernate – prohibit!
- Windows - never disable swap!

Operating Systems — Linux - 1

- Linux – uname -a
  - core 5.+ minimum, 6+ recommended
- Power configurations
  - maintain High performance
  - Power saving also exists in Linux!

Operating Systems — Linux - 2

- Swapiness
  - If RAM>32Gb in /etc/sysctl.conf vm.swapiness = 1
- Swap - never disable swap!
- Max Open Files
  - set 50000 minimum, for SaaS with 50+ DB – 500000+
- vm.max_map
  - 250000 minimum, for 50+ DB - 1000000

File Systems

- Linux – ext4 no barrier
  - xfs, zfs shows lower performance than ext4
  - Details are in Firebird Linux webinar
- Windows – NTFS
  - Cluster size – can be default

Firebird Performance Webinars about Firebird and Linux

- Webinars with Firebird core developers for Firebird Supporters
  - Windows with Vlad Khorsun
  - Linux with Alex Peshkoff
- Recordings are available
- More to come!

SQLs

- Problematic SQLs for VMs
  - many reads and writes – problem with slow disks
  - many fetches - problem with shared CPU
- Less problematic – frequent SQLs
- Analyze with trace logs and advanced tools

Network

1) Firebird Version — 3+, 5.0.3 to work with BLOBs

2) Client versions – need to be updated, client versions need to equal server version

3) Use of BLOBs

4) Configurations in conf

6) Basic check

Firebird Version - 3+

1) In version 3 the network protocol was optimized
2) Even more optimized in 4
3) Breakthrough in version 5.0.3

Client versions

1) Client version = server version

2) To check (v3+):

```
SELECT DISTINCT MON$CLIENT_VERSION FROM
MON$ATTACHMENTS
```

Zoo example:

```
LI-V4.0.0.2496 Firebird 4.0
WI-V3.0.5.33220 Firebird 3.0
WI-V4.0.1.2692 Firebird 4.0
WI-V4.0.2.2816 Firebird 4.0
WI-V4.0.3.2975 Firebird 4.0
WI-V5.0.0.1306 Firebird 5.0
```

Use of BLOBs

- Blobs
  - problem with blobs in network protocol was resolved in v 5.0.3
    - See BLOB revolution article https://firebirdsql.org/en/community-news/blob-revolution
- Versions <5.0.3
  - cast to VARCHAR as a workaround
  - don't include BLOB for SELECT for grid, do separately
  - 1:1 tables with BLOB fields
  - store in varchar preview with part 1, remainder in blob of another table

Configurations in conf

- WireCrypt and WireCompress
  - Makes a difference!
  - Mandatory libraries on client side - from Firebird distributive

Basic Check

- ping server_ip -f -l 1472
- MTU 1500

Balance

1) How to measure resource usage between databases?

2) How to measure resource usage between applications and Firebird (on the same server)?

3) How to divide resource usage between databases?

4) How to divide resource usage between Firebird and applications?

How to measure resource usage between databases?

1) In vanilla Firebird with SuperServer/SuperClassic – difficult.

With Classic can identify processes and see trace log and MON$

2) In HQbird has resources in SuperServer to see load per database

How to measure resource usage between applications and Firebird (on the same server)?

1) Memory - RAMMap in Windows, top in Linux

2) Disk – Resource Monitor in Windows, iostat in Linux

3) CPU – Resource Monitor, top

How to divide resource usage between databases?

1) Different Firebird instances on different ports

- install.sh -path /opt/fbXXX
- RemoteServicePort and RemoteAuxPort

2) Can use Classic and SuperClassic on different ports to work with the same database

How to divide resource usage between Firebird and applications (same server)?

1) CPUAffinity (Windows)

2) nice in Linux

3) Other tools that manage affinity and process priority (not only for Firebird)

Security

- Don't use masterkey
- Don't use SYSDBA
- Encrypt databases
- Change the standard ports RemoteServicePort (3050), RemoteAuxPort
- Use Srp (strong passwords), don't use LegacyAuth
- Don't share folders on the server!

Testing clouds

- Don't believe what they say, do the test!

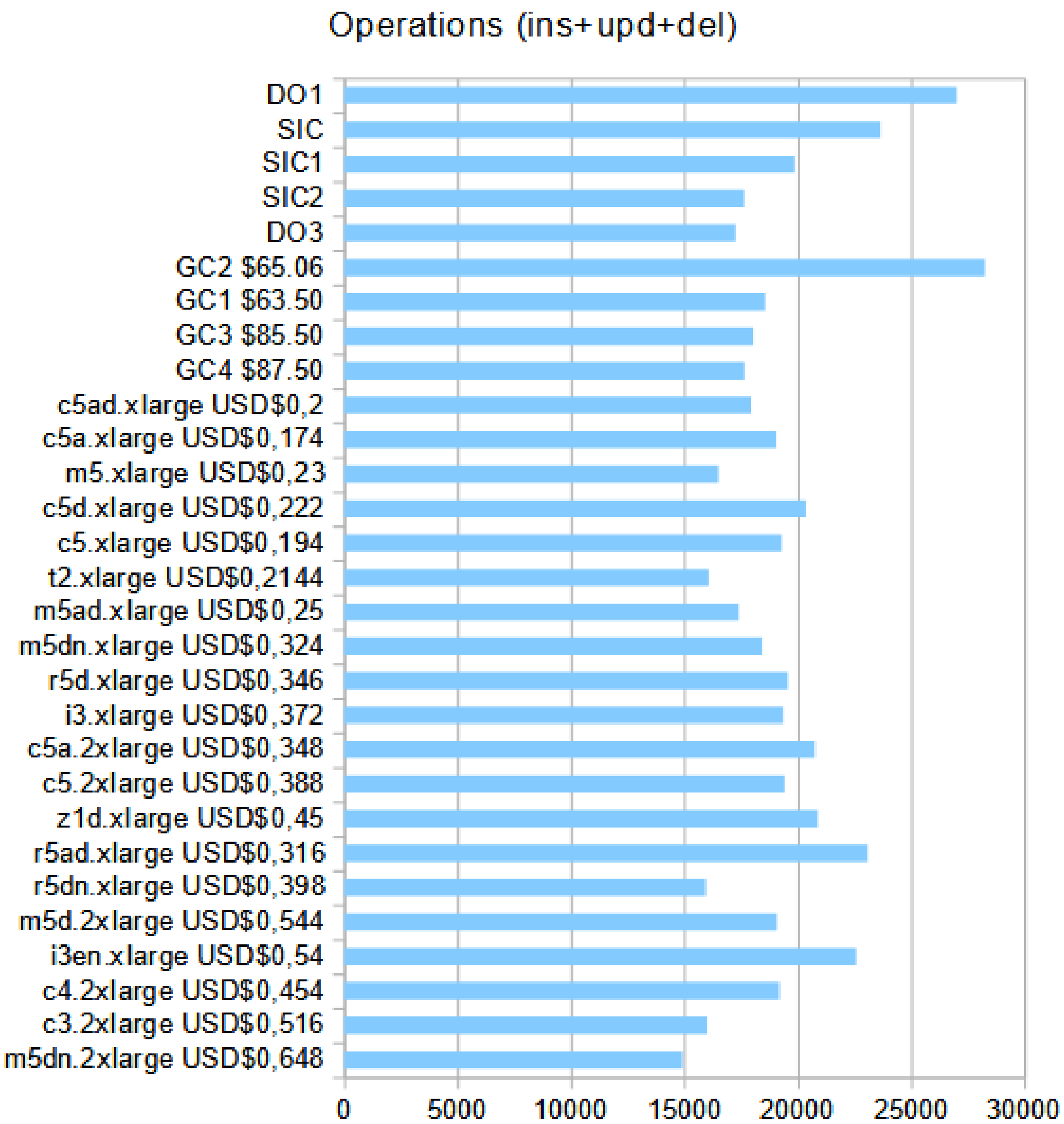
https://ib-aid.com/en/simple-insert-update-delete-test-for-firebird/

Testing clouds: Top 3:

1) Google Cloud

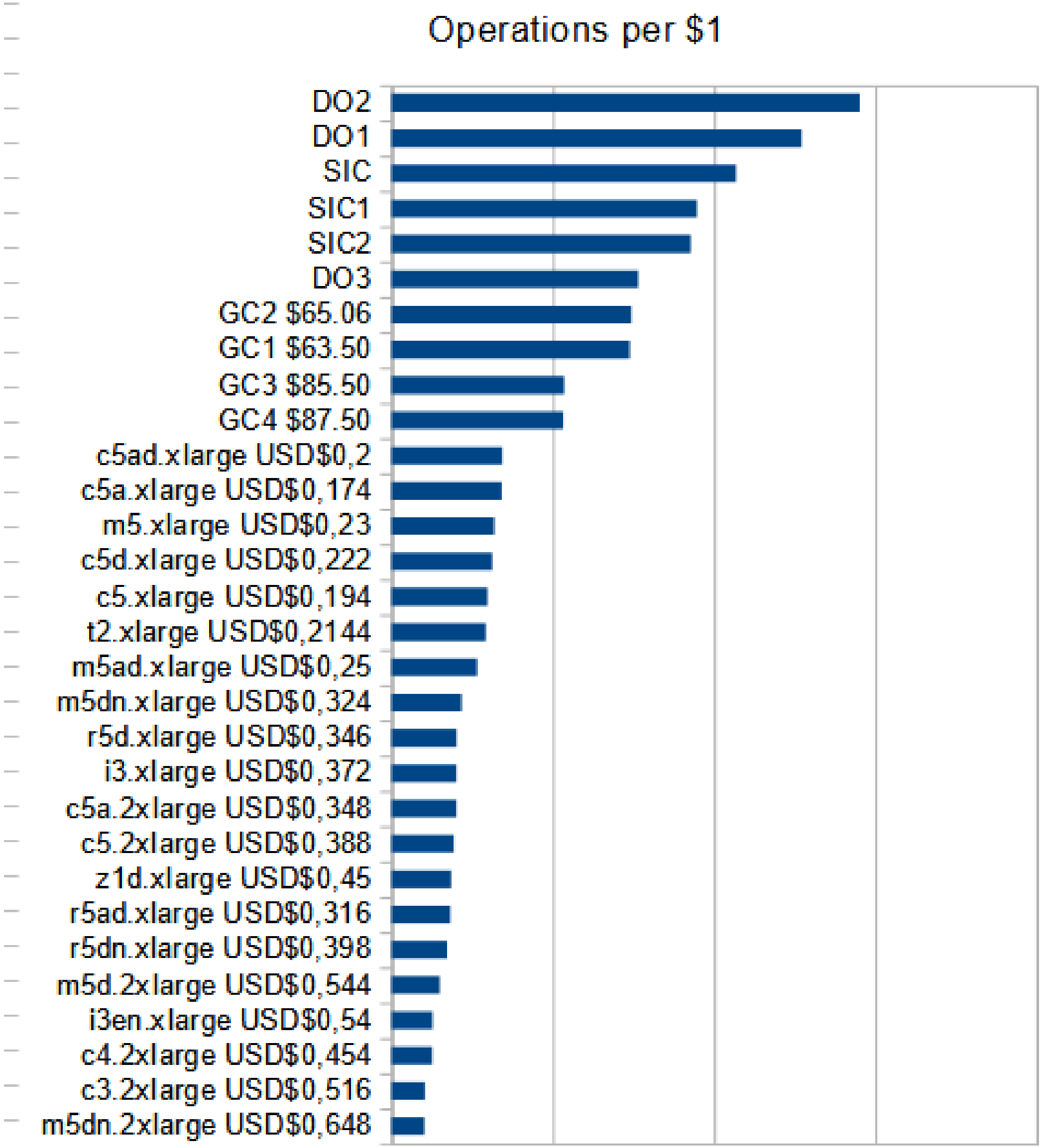
2) Digital Ocean

3) AWS i3en.xlarge



Testing clouds: value per $1

- Value per USD$1:
  1) Digital Ocean
  2) SaveInCloud
  3) Google Cloud
  4) AWS c5ad.xlarge



Thank you!

- Questions?
  - ak@firebirdsql.org
- Become a Firebird Supporter now:
  - https://store.firebirdsql.org/
  - Certification
  - Access to EmberWings magazine
  - Access to webinars with core developers (recording and new)
  - Discounts and special offers