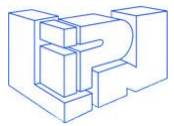
NEO Overview

This guide will teach you:

- What NEO is
- NEO vs ZEO
- NEO main concepts
- NEO main features

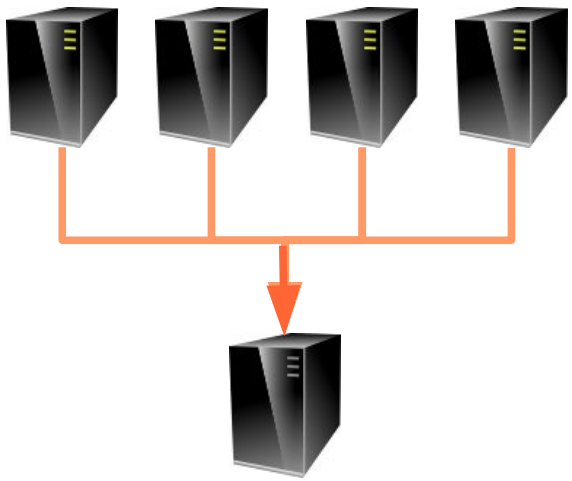


Support NEO !
www.neoppod.org

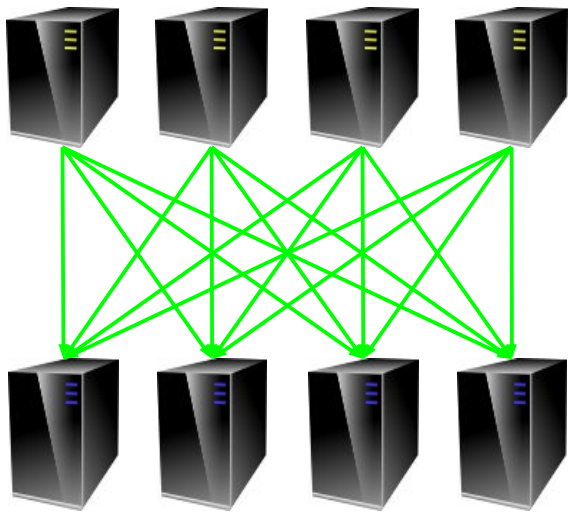
ERP5

NEO vs ZEO - Scalability

ZEO

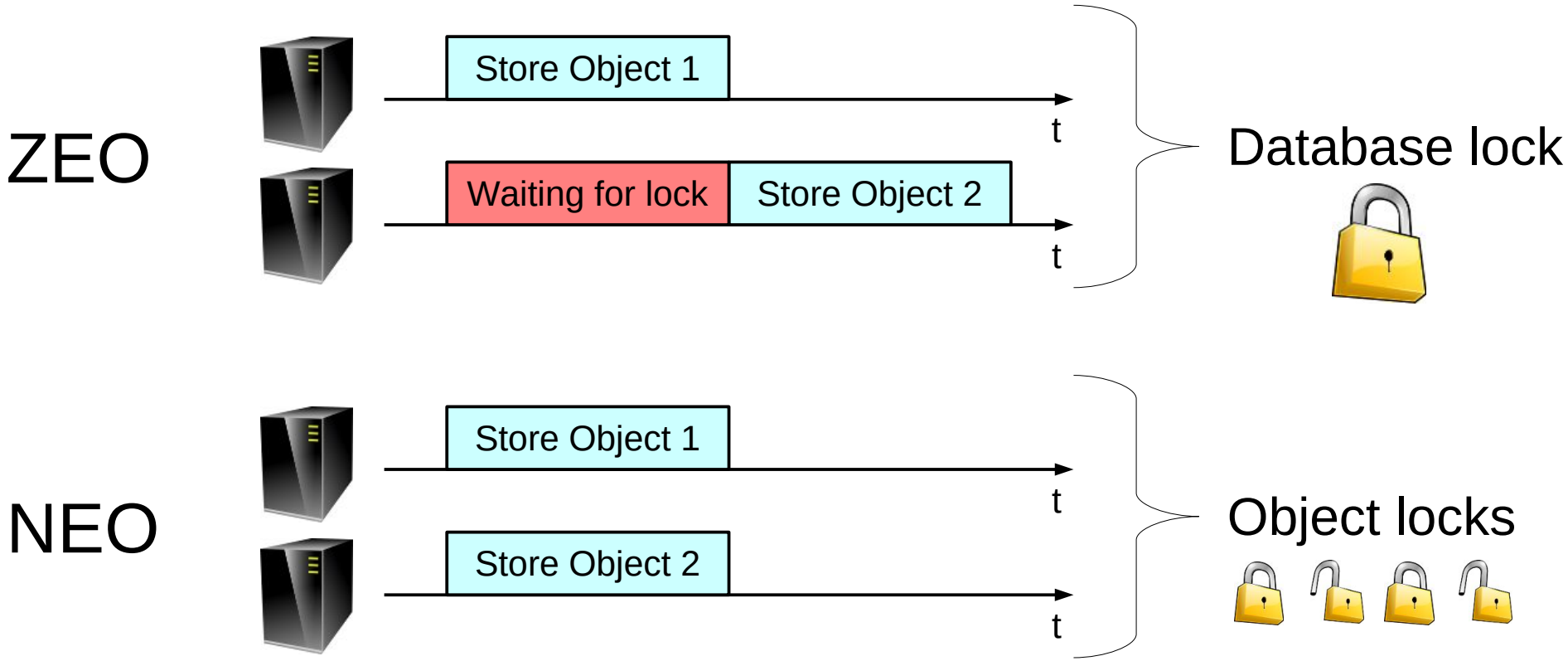


NEO



ERP5

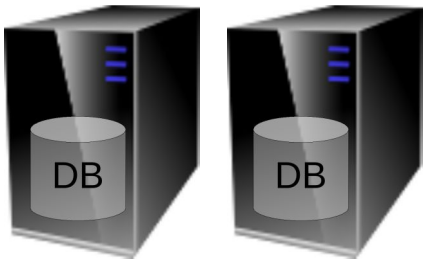
NEO vs ZEO – Locking Level



ERP5

Types of nodes

Masters



Storages

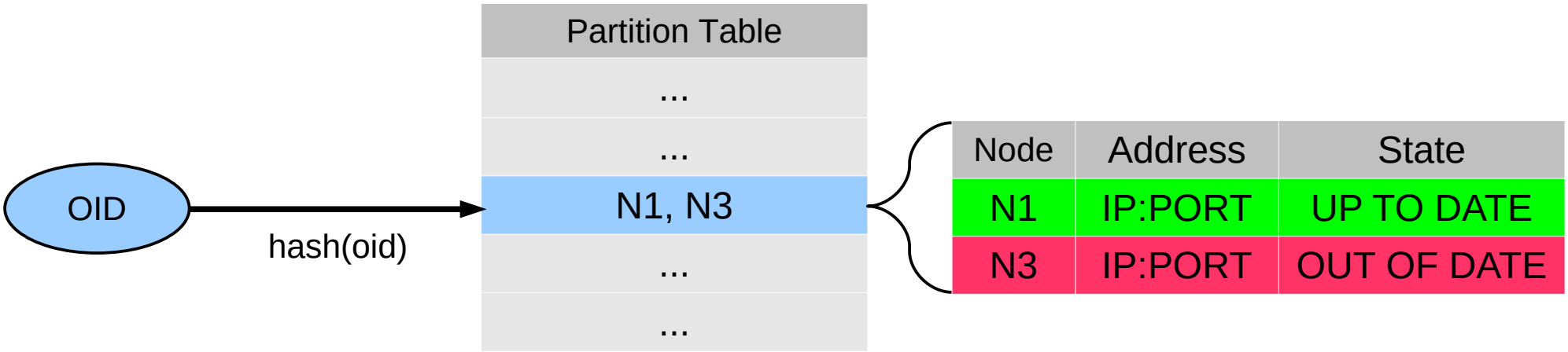
Clients

Admin

NeoCTL

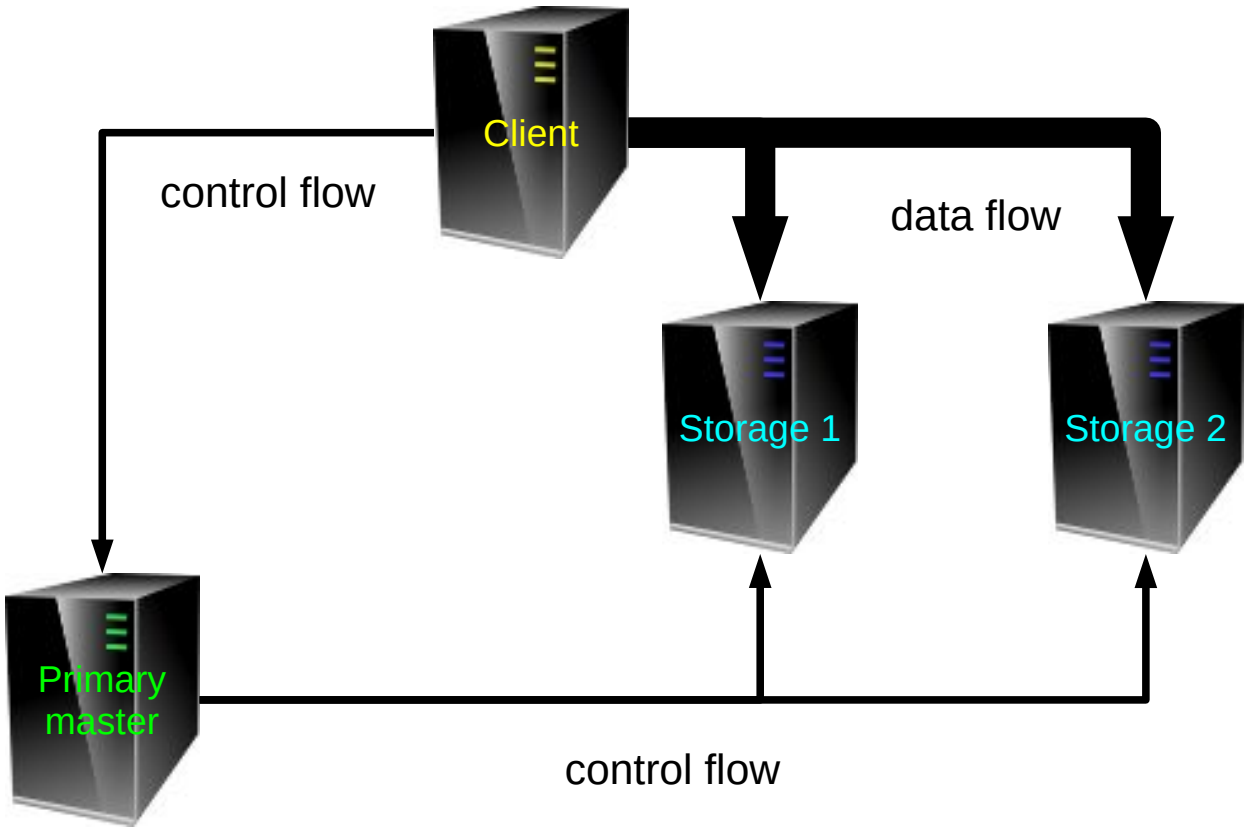
ERP5

Object access



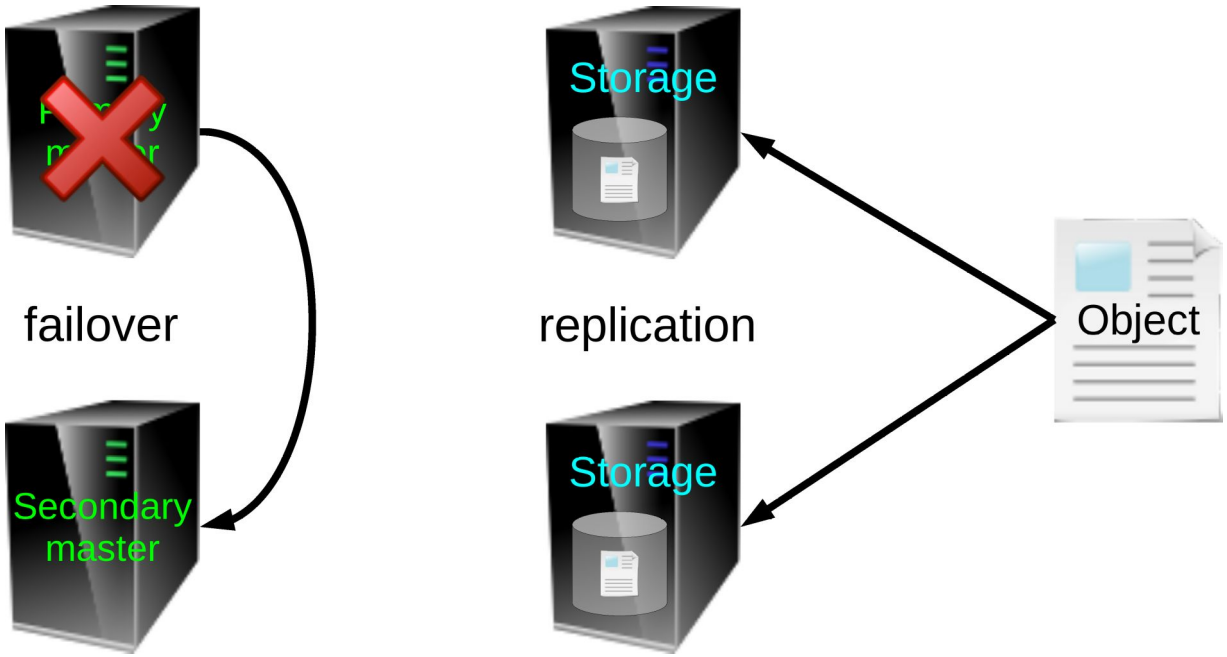
ERP5

Network communications



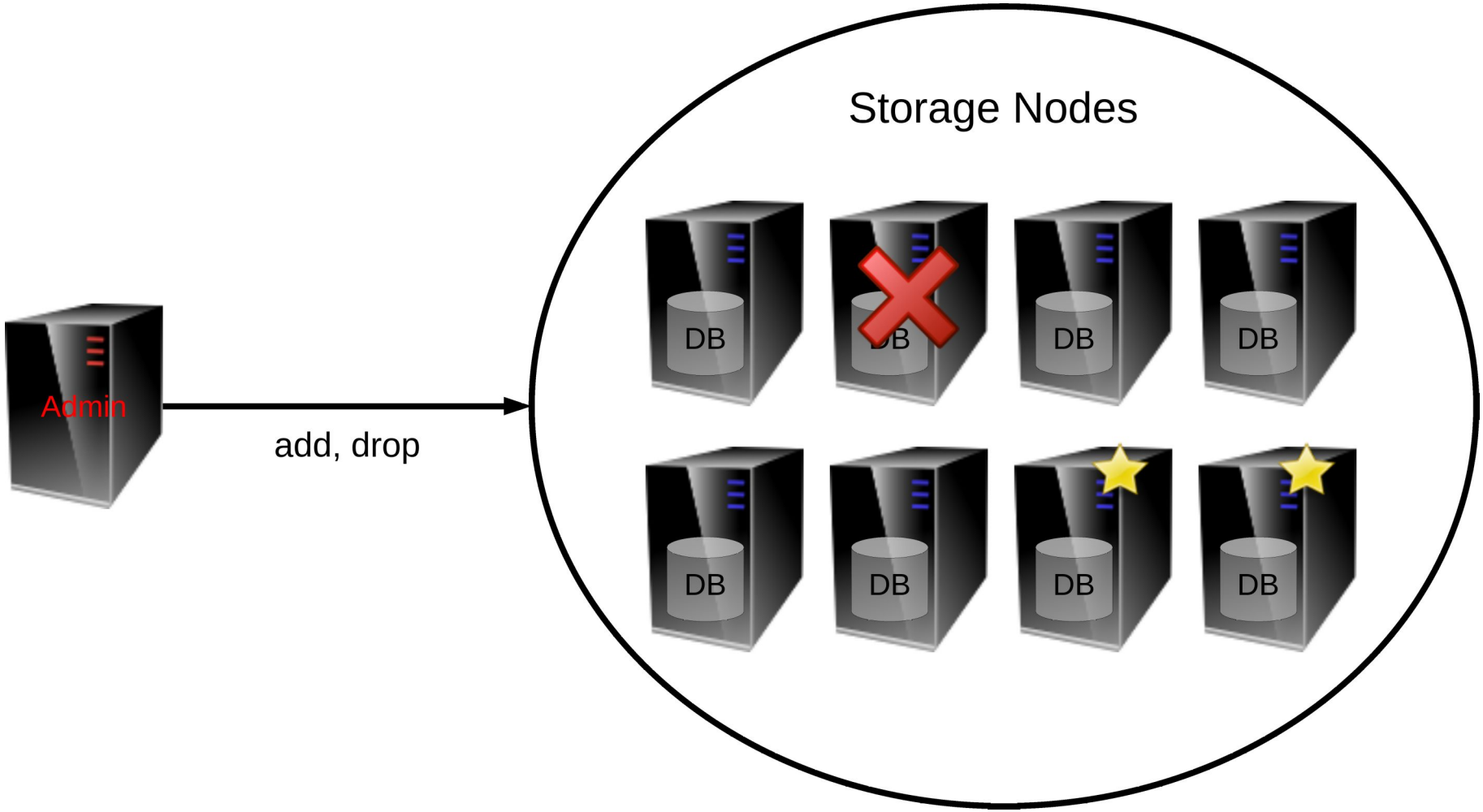
ERP5

Fault-tolerant design



ERP5

Dynamic configuration



ERP5

NEO Overview

This guide will teach you:

- What NEO is
- NEO vs ZEO
- NEO main concepts
- NEO main features

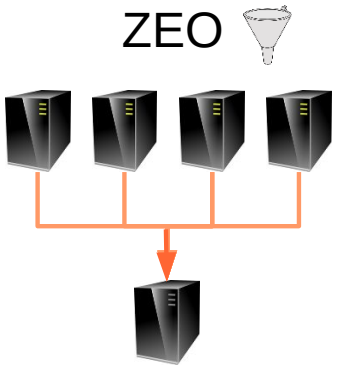
Support NEO !
www.neoppod.org

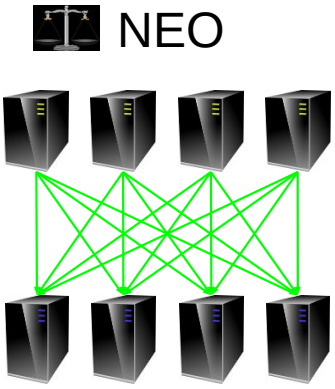
ERP5

Nexedi Enterprise Objects (NEO) is a project to make a robust and fast storage system for Zope Object Database (ZODB) in a distributed manner. The goals are to provide a reliable and scalable database that can overtake the limits reached by the ZEO implementation.

NEO project was initiated by Nexedi SA and is now endorsed by System@tic competitive cluster, by Paris Region and by FEDER programme of the European Union.

NEO vs ZEO - Scalability



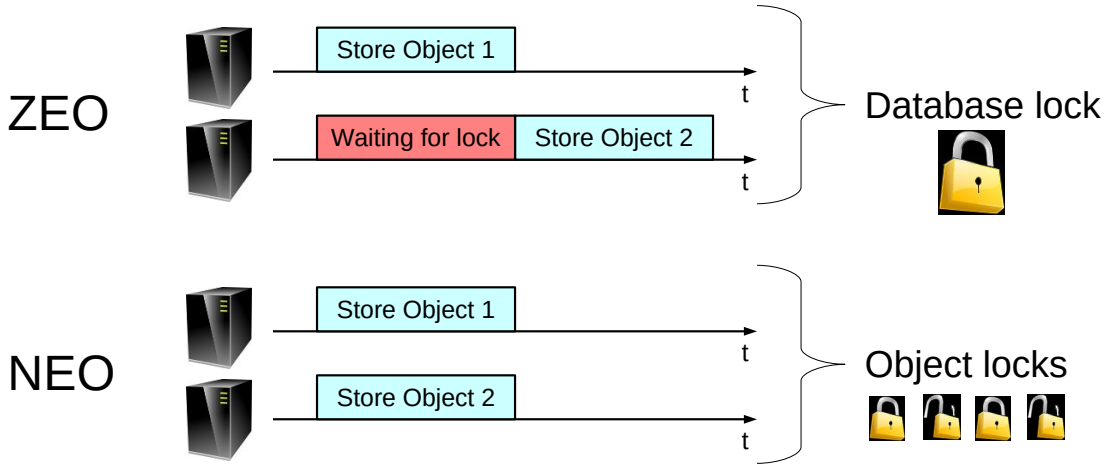


ERP5

ZEO is a centralized system, it exposed the FileStorage API and store all data in one place. As long as the load increase, the storage may become overloaded (disks, cpu...) which lead to a technical barrier against scalability.

NEO is a distributed system, all objects are distributed among multiple storage nodes so the load induced by the clients activity is also distributed. Client nodes deals with objects that are available on multiple storage nodes and connect to them on-demand, so each storage handle a subset of operations and a part of the load.

NEO vs ZEO – Locking Level

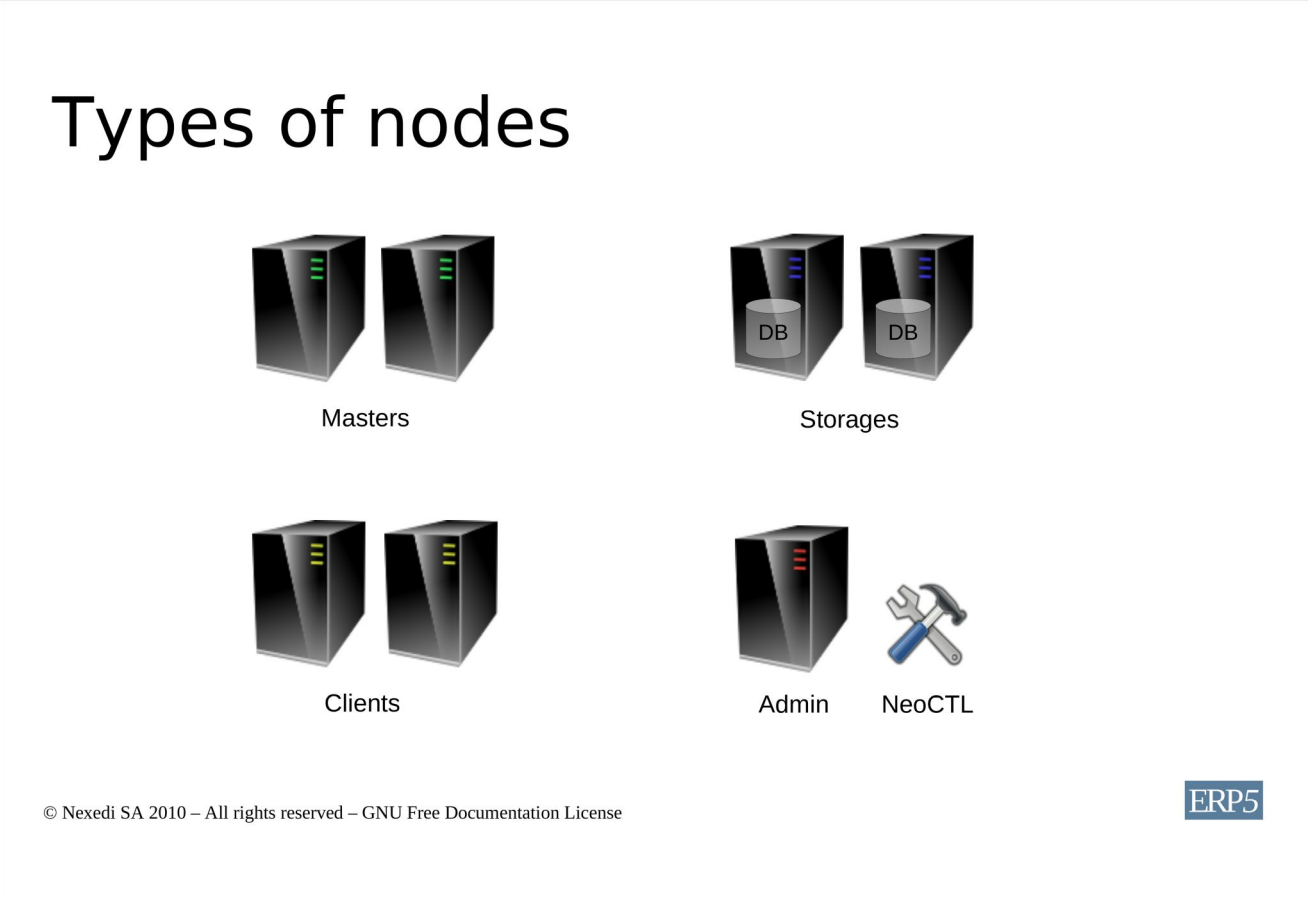


ERP5

One of the main advantages of NEO over ZEO is its locking level.

The ZEO rule is simple, the whole database is locked at commit to prevent concurrent transactions to modify the same object. This is a bottleneck when dealing with many concurrent requests as all commits are serialized.

NEO locking system is fine-grained, a lock is held at the object-level when a store is requested. Thanks to that approach, two transactions that doesn't modify the same objects can be committed in parallel. This lead to a reduced commit latency and increased bandwidth.



As NEO is distributed, different types of nodes are involved:

One of master nodes gets elected to become the "primary master node", other becoming "secondary master nodes". The primary handles all centralised tasks, such as OID/TID generation, cluster consistency checking and broadcasting cluster changes to all nodes. When a primary master falls, secondaries take over by restarting an election.

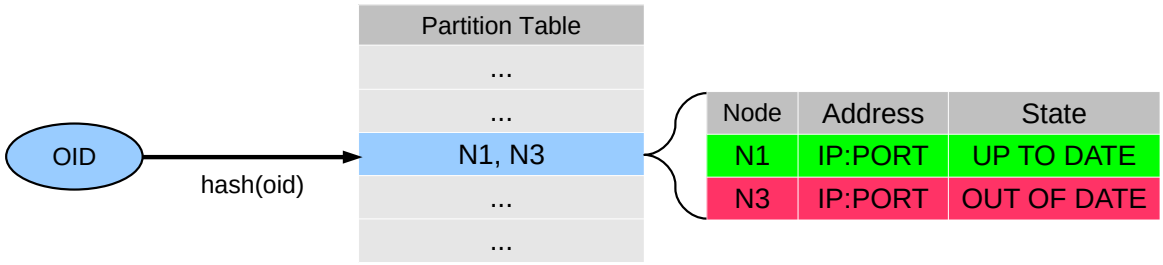
Storages nodes are responsible of load and store operations requested by client nodes. It contains replication facilities so they can mirror other storage nodes' cells. They embed a relational database that act as a key/value backend with transactional facilities.

Client nodes implements the Storage API and talk with the primary master and storage nodes on demand. Those nodes are multithreaded to fit with requirements of the ZODB API.

The admin node is the standard way to interact with NEO internals, it is for example required at bootstrap to decide which partition table should be used or collect statistical data.

NeoCTL is a command line utility to access admin node and control the cluster.

Object access



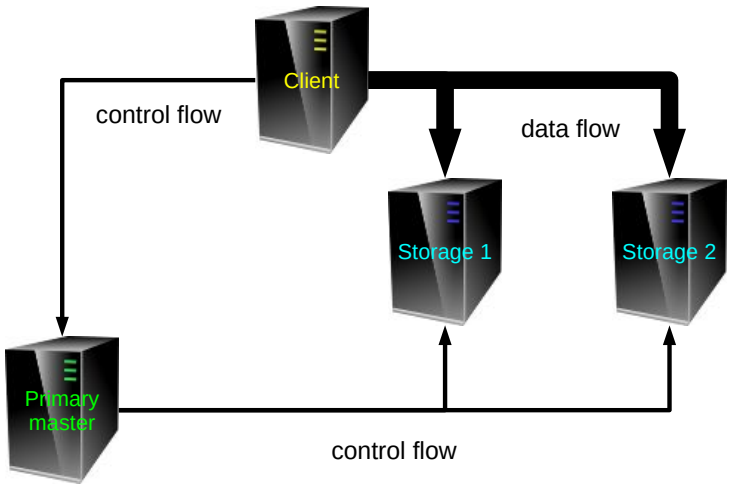
ERP5

In NEO the object identifier (OID) is used to compute the partition to which the object belongs. An object is always assigned to one and only one partition and cannot be moved to another, but a partition can be replicated multiple times among multiple nodes.

A partition is composed of a number of cells that are just a storage node with a known state. The cell state indicates if the node can be used for load and/or store operations, for example:

- UP TO DATE: The node has a full copy of the partition content and can be used for load and store operations, this is the stable state.
- OUT OF DATE: The node may have some missing data (or even nothing), it must not be used for loading objects but new objects must be written to. Any cell in this state means that the storage node is replicating missing data and will switch to the state above once it is finished.

Network communications

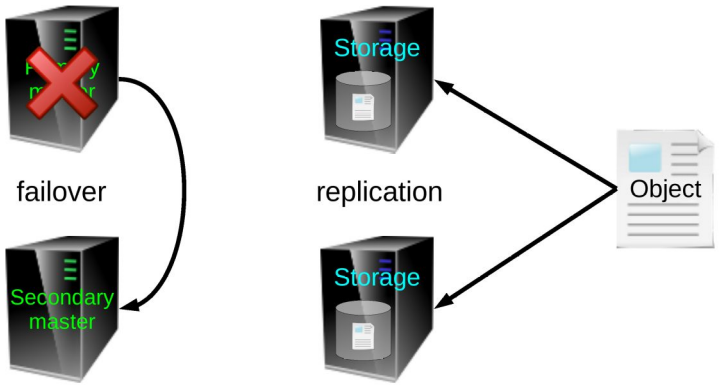


ERP5

As it is distributed, NEO is heavily based on network communications. Operations such as object load and store are done directly between client and storage node. The data flow is distributed among storage nodes to reduce the load and allow scalability.

Operations such as a transactional commit must be controlled by a central point, called the primary master node, as it guarantee unicity and atomicity. There is almost no cluster-wide operations, a transactions involves only a subset of all storage nodes, depending on the transaction content.

Fault-tolerant design

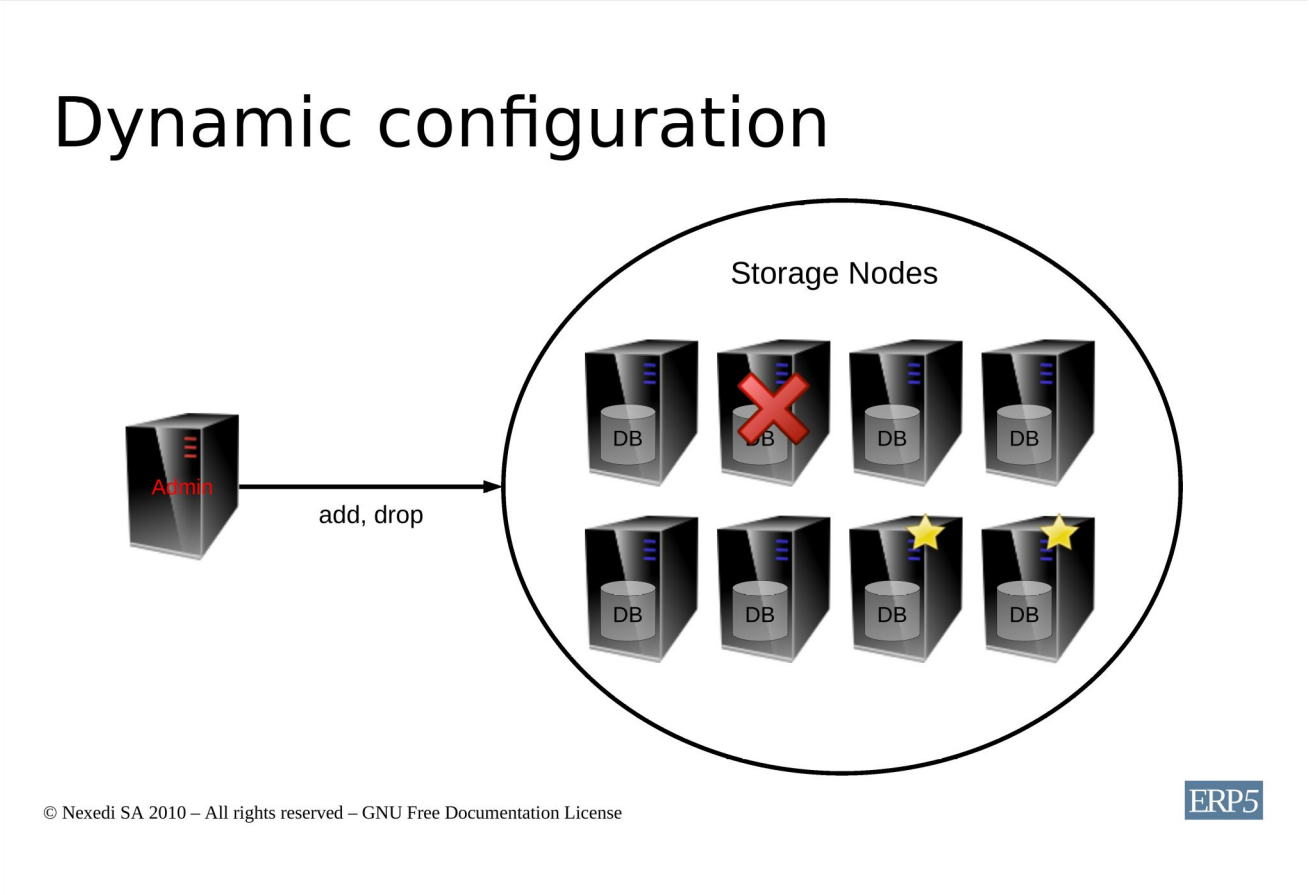


ERP5

NEO is fault-tolerant by design.

An object is written belongs to one partition but is written to multiple storage nodes, as much as the replica number requested if enough storage nodes are available. If a storage node crash, the others that handle the same partitions can take over the operations with no down time. A NEO cluster can face to storage crashes while there is at least one storage available per partition.

For the masters side, only one can be primary but secondaries are available to replace it in case of failure. Note that this require an administrator choice to validate the recovery informations found on the available storages nodes to prevent a restart from an old cluster state.



NEO is dynamically configurable, no down time is required to performs all administrative operations like:

- Adding new storage nodes when more space of performances are needed, the current partition table will be automatically updated and balanced. Replication processes will be triggered to initialise new nodes with objects that belongs to partitions assigned.
- Dropping existing nodes in case of hardware failure or an under-expected load.
- Changing replication rules like the number of object copies to improve reliability or reduce the required space.