All-Active High Performance CIFS Clustering

Ronnie Sahlberg, Samba Team

- What do you do if you have tens of thousands of users, PBytes of data, hundreds of servers, thousands of shares.

- You use AllActive CIFS clustering and consolidate all your hundreds of servers and thousands of shares into one single server and one single share!

Who am I

- Ronnie Sahlberg, CTDB Developer
- Member of Samba Team
- Working for IBM OzLabs

What is CTDB?

What is CTDB

- An All-Active cluster database
- Clustered CIFS (when used with Samba)
- A framework for All-Active clustering
- Very Low-overhead
- Very scalable
- Very high redundancy
- **It provides 100% correct CIFS semantics!**

- It is fast. Very fast!

Video I

CTDB history

History

- Clustering CIFS has been the holy-grail for CIFS for a long time.
- Samba Team and Andrew Tridgell has had plans and ideas on how to cluster CIFS.
- Different prototypes has been built, tested and evaluated over the years.

- A year and a half ago, we decided to start building a proper highly scalable framework for CIFS clustering.

CTDB current status

Current Status

- We have come very far in 18 months.
- CTDB is no longer just a clustering database. It is now a full-blown All-Active cluster framework.
- CIFS, FTP, iSCSI, NFS, ...
- It is used in production use serving critical data.

Main Components

Main components

- CTDB
- Samba
- RHEL5
- A cluster filesystem

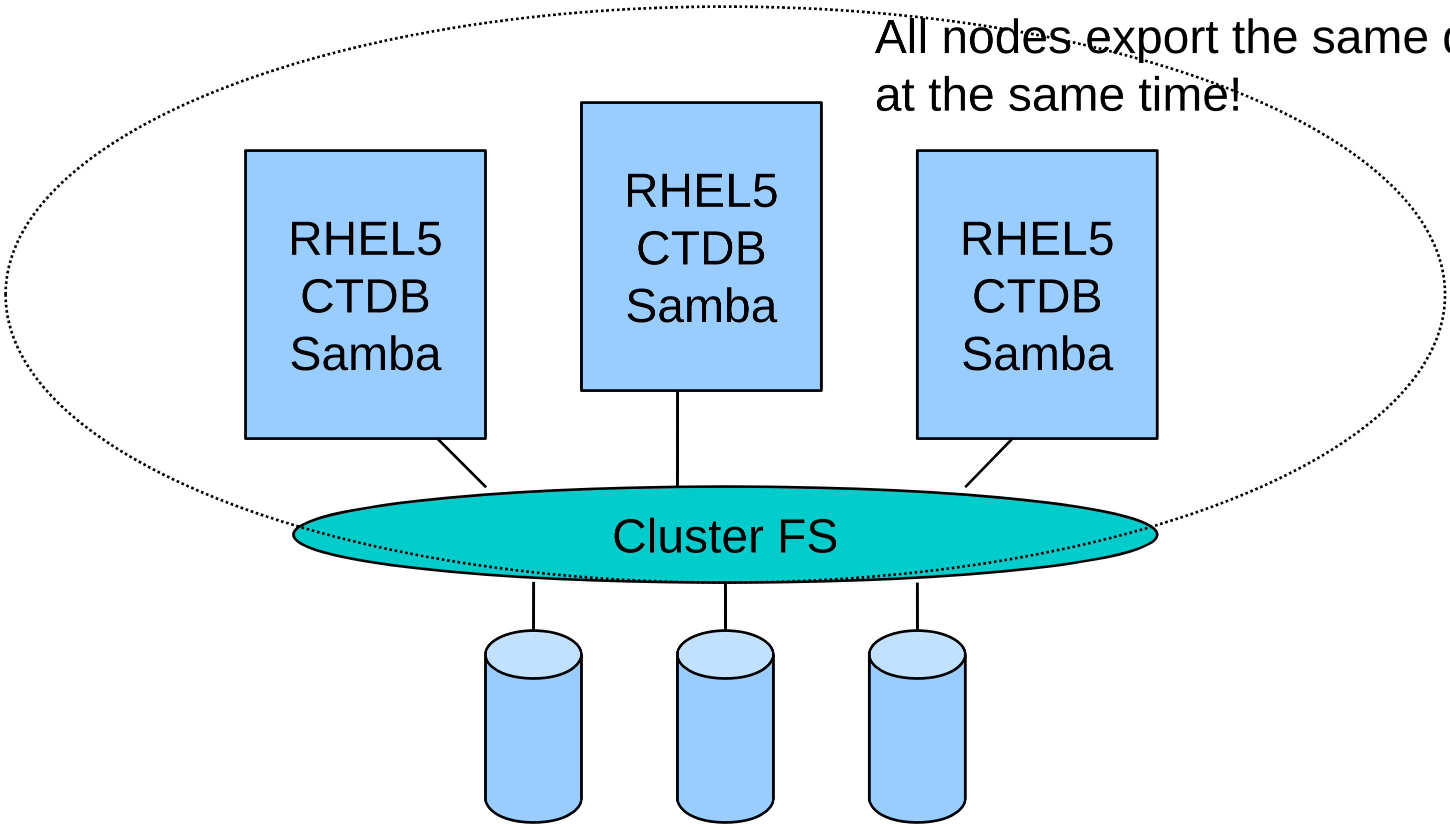
- ... and other components

Conceptual Overview

Overview

Many nodes but only one machine account.
One “instance” of Samba.
Sometimes even just one single ip address!

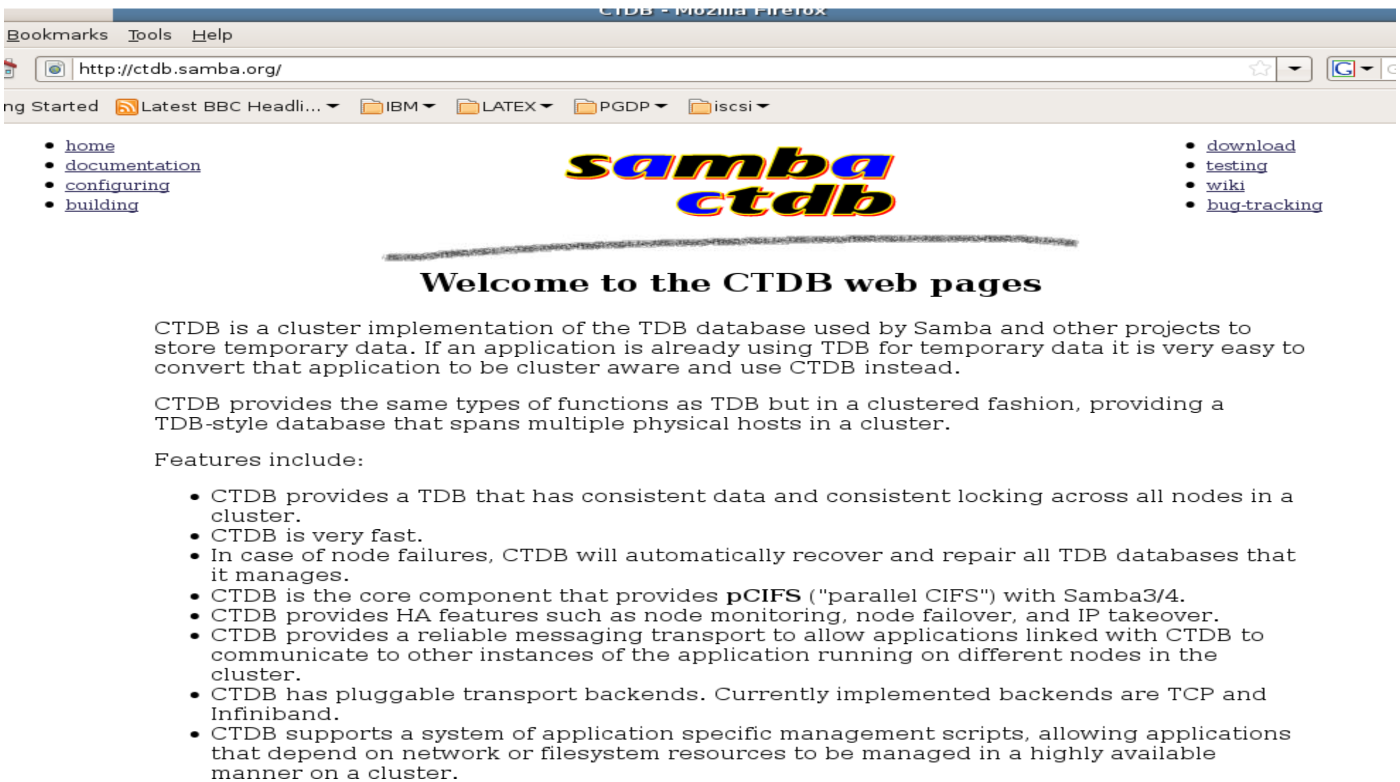
All nodes export the same data read/write at the same time!



Technology

Technology

- CTDB is open source and available at ctdb.samba.org



- home
- documentation
- configuring
- building

samba ctdb

- download
- testing
- wiki
- bug-tracking

Welcome to the CTDB web pages

CTDB is a cluster implementation of the TDB database used by Samba and other projects to store temporary data. If an application is already using TDB for temporary data it is very easy to convert that application to be cluster aware and use CTDB instead.

CTDB provides the same types of functions as TDB but in a clustered fashion, providing a TDB-style database that spans multiple physical hosts in a cluster.

Features include:

- CTDB provides a TDB that has consistent data and consistent locking across all nodes in a cluster.
- CTDB is very fast.
- In case of node failures, CTDB will automatically recover and repair all TDB databases that it manages.
- CTDB is the core component that provides **pCIFS** ("parallel CIFS") with Samba3/4.
- CTDB provides HA features such as node monitoring, node failover, and IP takeover.
- CTDB provides a reliable messaging transport to allow applications linked with CTDB to communicate to other instances of the application running on different nodes in the cluster.
- CTDB has pluggable transport backends. Currently implemented backends are TCP and Infiniband.
- CTDB supports a system of application specific management scripts, allowing applications that depend on network or filesystem resources to be managed in a highly available manner on a cluster.

Technology

- We currently need a special version of samba with clustering support built in : http://ctdb.samba.org/packages/redhat/RHEL5

- In the future clustering will be built into all versions of samba by default.

- /etc/samba/smb.conf:
    clustering = yes
  to activate

Scalability
and
clustering overhead

Scalability

- What are the goals we want to achieve?

Scalability

- Low overhead: A one node cluster should perform no worse than a normal non-clustered samba install.
- Positive scaling: A N+1 node cluster should perform better than a N node cluster.

- (We initially wanted to be able to scale to hundreds of nodes, but given the very good scaling we see this would be ridiculous. No one would ever need a system that fast.)

Scalability

- Samba itself is (for many reasons) a single- threaded application. Each client connections results in a separate process.
- Thus samba already was well abstracted between the handling of data and metadata.

- Samba used TDB databases to synchronize metadata across the processes.

Scalability

- These TDB databases are memory mapped and very very fast.
- Example: byte range locking database!

- Our challenge was to design a distributed version of TDB that both preserved the semantics of CIFS and also was virtually as fast as memory a mapped database.

Scalability

- Why not just put these TDB databases inside the clustered filesystem?

- That would be easy! But it wouldnt work and it would be SLOW.

Scalability

- How slow? This slow:
  1 node : 30.0
  2 nodes : 2.1
  3 nodes : 1.8
  4 nodes : 1.8

- We have not gained anything. We only shifted all data shuffling down into the cluster fs layer, thinking it would magically be able to do inter-node communications infinitely fast and with 0 latency.

Scalability

- We have to avoid network traffic.
  We may also need to allow the databases to become out of sync across the nodes!

- This makes things more “interesting” because we now need to research areas such as “safe-metadata-dataloss” :-)

- Be very careful. This is tricky. But the pay-off would be enormous!

DMASTER/LMASTER

- LMASTER : LocationMaster. Each record has an Lmaster.
- Lmaster roles:
  Create the record
  Destroy the record
  Keep track of onto which node the record is currently migrated to/hosted
- All records are evenly distributed across the LMasters.

LMASTER/DMASTER concept

- DMASTER : DataMaster. This is the node which currently holds the current version of the record.
- In order to WRITE to a record, a node must first become DMASTER, i.e. cause the record to be migrated onto the local node.
- Each record has a CTDB header prepended to it which contains among other things a serial number for the record and also a hint which tells the node "This node is DMASTER for this record: yes/no".

LMASTER/DMASTER concept

- Thus, the process to write to a record would thus be (simplified):

  Samba grabs the record from the memory mapped database. If DMASTER == local node then we can access the record directly just as we do in the non-CTDB case. This case does not involve any CTDB involvement or any network I/O at all.
  This is the average case.

LMASTER/DMASTER concept

- Othervise, Samba detects that “We are not DMASTER for the record, thus the record might be 'old'” and would invoke the local CTDB daemon to locate the record in the cluster and pull it/migrate it onto the local node.
- This involves network I/O but is far from the average case.
- In the average case the overhead is 0.

LMASTER/DMASTER

- We need special care when nodes start/stop/crashes to recover the databases.

LMASTER/DMASTER

- This approach give us a 1-node cluster with almost identical performance as a normal non-clustered samba.
- It gives us virtually linear scaling in many use cases.
- 1 node : ~600MByte/sec
  2 nodes : ~1GByte/sec
  4 nodes : ~1.7GByte/sec

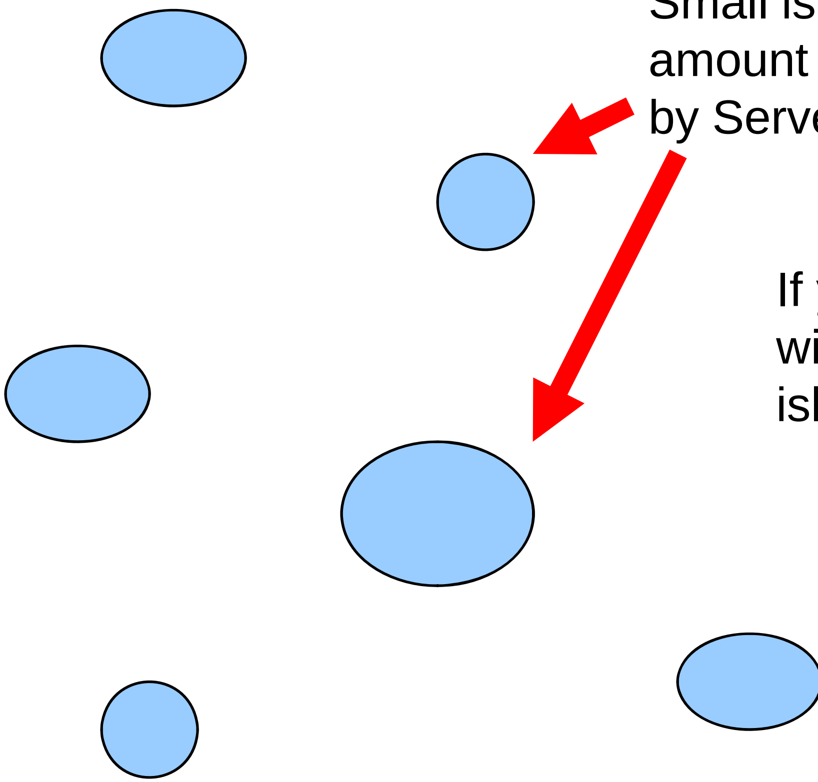
Single Namespace

Single Namespace

- In traditional Active/Passive failover NAS you can only export/share the same data from one node at a time, the Active node.
- This together with contraints in CPU, I/O and amount of data that can be hosted on the Active node means you will have to partition your data into small disjoint islands.
- A lot of small isolated data islands.

not-SingleNamespace

Small islands of ~10TB each hosting a small amount of your data. Each island is keyed by Server/Share.

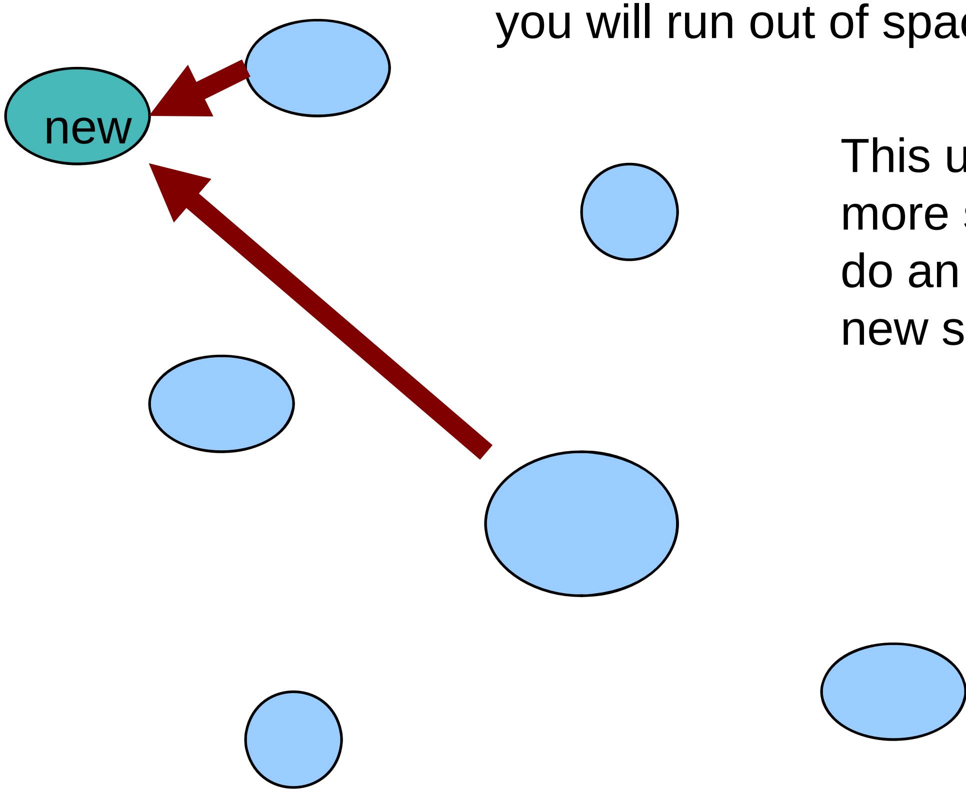


If you have a few Pbyte of data you will end up with many many small islands

Anyone here have >100 servers to host their data?
How much does that cost in maintenance?

not-SingleNamespace

Even worse. Your data is not static and demand grows at ~25% per year? That means of your 100 Server/Shares you will run out of space on 25 shares per year.



This usually means that you have to add some more servers and storage, then you need to do an online migration of some data onto the new server/share.

But now your data has changed name so you must update all your apps.

Thats pretty expensive.

SingleNamespace

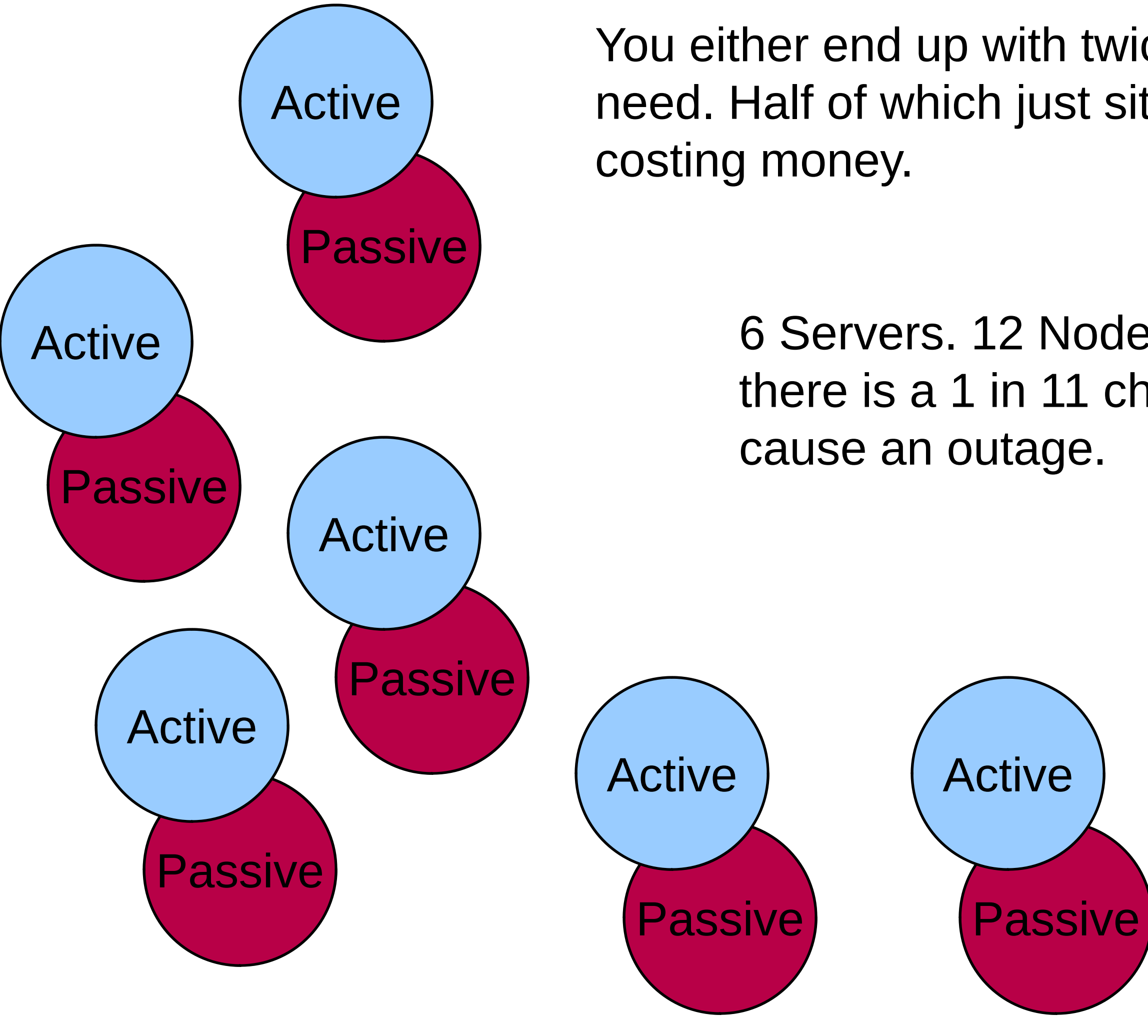
- Using an All-Active cluster like CTDB there is never any need for any migration. All servers are always hosting/serving all of the data, all of the time.
- Thus you just add more nodes to your cluster when you run out of processing power, or more storage to your cluster filesystem.

Reliability and better use of capital resources

Active/Passive

- Classic Active/Passive failover.
- Twice the capital cost.
- Half your equipment just sits idle.
- Savings possible by reducing reliability (3 active nodes per passive)
- Very nasty failure-modes.

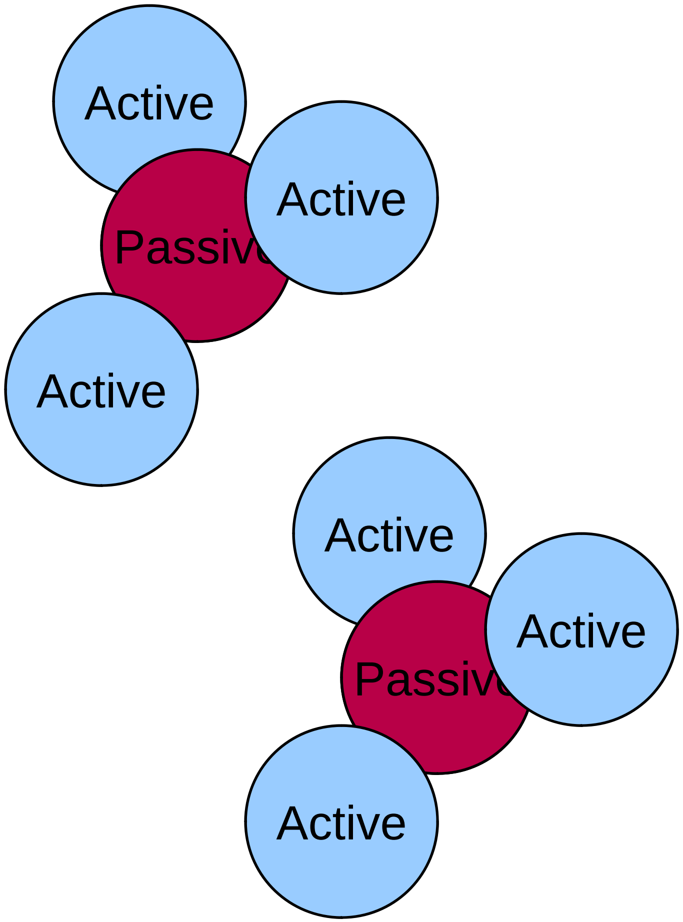
Active/Passive



You either end up with twice as many nodes as you need. Half of which just sits idle wasting energy and costing money.

6 Servers. 12 Nodes. After the first node failure there is a 1 in 11 chance the next failure will cause an outage.

Active/Passive



Or you can choose to losen your redundancy requirements a bit. Two node failures in the same group means DataUnavailable.

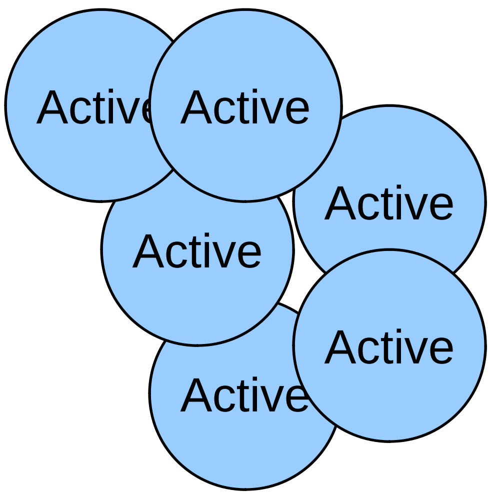
6 Servers, 8 Nodes. After the first failure there is a 3 in 7 chance the second failure causes an outage.

Maybe you dont really care much about redundancy and thats why this is acceptable?

AllActive

No redundant hardware just wasting space, energy and money

Vs an All-Active cluster. All 6 servers consolidated onto 1 server spread across 6 nodes.



First node failure just means there are only 5 nodes left to loadbalance across.

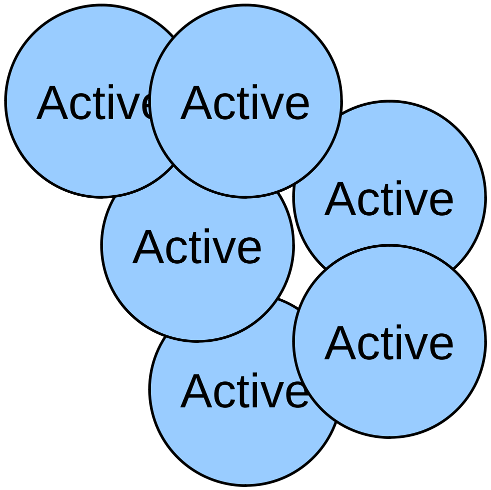
Second nodefailure never causes an outage. Only means you now only have 4 nodes to loadbalance across.

Third failure, again no outage.

Only when all 6 nodes are all out at the same time will you have an outage!

6 AllActive nodes provide vastly higher redundancy than even the most expensive 12 node Active/Passive solution. For half the cost.

AllActive



Adding nodes implicitely means two things :
Higher aggregated performance (thanks to the scaling)
Higher redundancy.

No waste and eco-unfriendly row-upon-rows of powered on but passive systems in your datacentre.

Active/Passive vs AllActive

Failure modes are very nasty in Active/Passive systems

Active/Passive

- In Active/Passive you can only run one node at a time. In a two node configuration this would look something like this :

In a perfect world you would have these two possible states :

Node 1	Node 2
ON	OFF
OFF	ON

Active/Passive

- In the real-world you rather have 4 states :

Node 1	Node 2
OFF	OFF
ON	OFF
OFF	ON
ON	ON

These two are the two states when the system is operational and serves data. Transition between the two states always go via the OFF/OFF state.

Active/Passive

- In the real-world you rather have 4 states :

Node 1	Node 2
OFF	OFF
ON	OFF
OFF	ON
ON	ON

This state occurs during the failover. Sometimes during failover the system will get stuck in this state and you have a long DataUnavailable outage.

This happens for example if the dormant passive node has developed a fault and you wont find out until you need the failover to work and it failed. :-(

That is actually not a big deal. DataUnavailable for a long time is a pretty minor inconvenience.

Active/Passive

- In the real-world you rather have 4 states :

Node 1	Node 2
OFF	OFF
ON	OFF
OFF	ON
ON	ON

DataUnavailable for a long time is a minor inconvenience? Am I insane?

Hmmm. Consider the fourth state.

Once this state is activated it will usually transition to the OFF/OFF state very quickly and permanently.

Anyone have first-hand experience with the ON/ON state?

AllActive

- For AllActive

Node 1	Node 2
OFF	OFF
ON	OFF
OFF	ON
ON	ON

This is the normal state.

AllActive

- Or rather THIS is the normal state :

Node 1	Node 2	...		Node N
ON	ON	ON	...	ON

Active/Passive

- You do test failover/failback once a week to check that everything is ok, right?
- Why not? Too disruptive?
- Small risk that failover will not work?
- There is always a certain amount of uncertainty whether the failover will actually work.

- How long/how disruptive is a failover/failback in your environment?

AllActive

- You dont need to test failover in AllActive since it doesnt do failover. All nodes are ALWAYS active and thus you know that they work!

AllActive

- Just check the node status :

```
[root@c2n1 SOFS]# ctdb status
Number of nodes:4
pnn:0 10.0.0.21        OK (THIS NODE)
pnn:1 10.0.0.22        OK
pnn:2 10.0.0.23        OK
pnn:3 10.0.0.24        OK
Generation:993786231
Size:4
hash:0 lmaster:0
hash:1 lmaster:1
hash:2 lmaster:2
hash:3 lmaster:3
Recovery mode:NORMAL (0)
Recovery master:0
[root@c2n1 SOFS]#
```

Video 2

Client Access

Client Access

- Three modes :

- IP address failover
- Layer-4 switch with single ip for cluster
- SW single ip for cluster

Command overview

CTDB

- The CTDB command tool is an interface to view and manage the CTDB cluster.
- Man ctdb
- Man ctdbd

Video 3/Demo

Resources

Resources

- ctdb.samba.org
- Samba-technical mailing list
- ctdb.samba.org/~tridge (presentations)
- ctdb.samba.org/~tridge/ctdb_movies

Questions?