

The Rule of “N”

Clive Longbottom,
Service Director, Quocirca Ltd

- Rationalisation
 - Bring various versions of the same application to the same release version
- Consolidation
 - Bring different instances of the same application on to one server
- Virtualisation
 - Bring different workloads on to the same server through different virtual servers





- Make many into one
 - Take all the server assets and make them into one logical resource pool
- Make one into many
 - Take resource as it is required from the pool as discrete servers
- But:
 - Not all workloads are well suited to virtualisation as yet
 - Virtualisation means that servers must be architected correctly

One app per server?

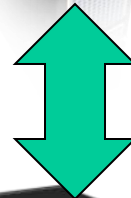
ERP

CRM

eMail

HR

etc.



ERP
CRM
eMail
HR
Etc...

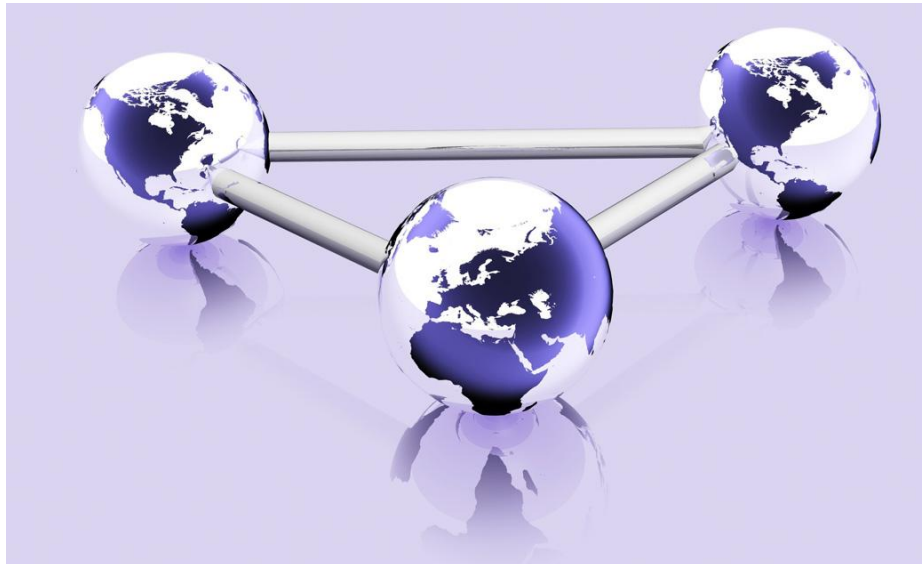
- Each application needs to have a server that is architected for it:
 - Peak usage
 - Cyclical usage
 - Leads to less than 10% utilisation overall
- Also – problems with availability





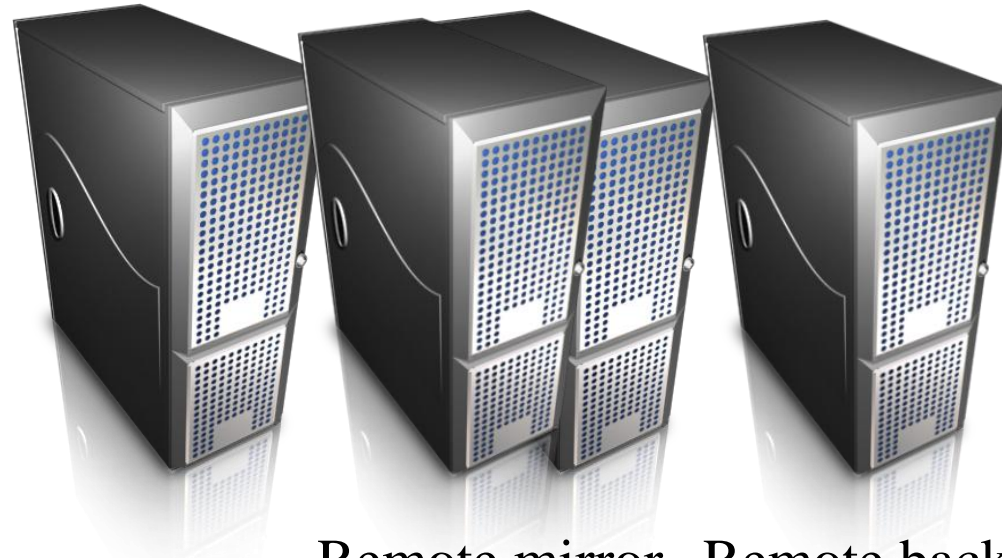
- Oops! It's gone wrong...
 - Is 99.995% uptime good enough?
 - Nearly 30 minutes per annum
 - How about planned downtime?
- Single server approaches lead to the need for massive DR policies
 - But, you're still going out of business every second you don't have access to your systems

- We can't afford for anything going wrong to impact the business
 - If it goes wrong, we need to make sure things still work
 - Provide spare capability to cover anything going wrong



Application

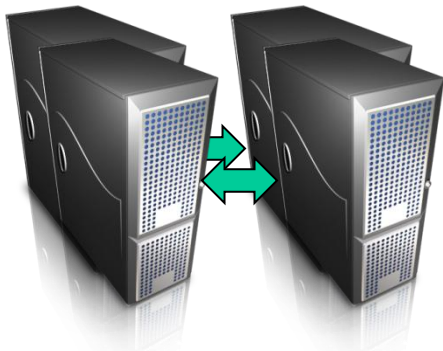
Application



Remote mirror Remote backup

Which leads to:

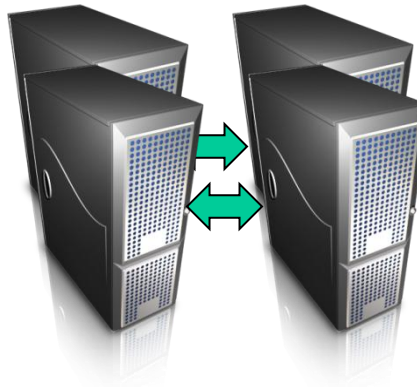
ERP



HR



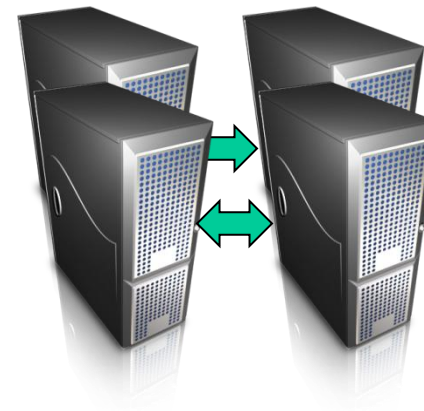
CRM



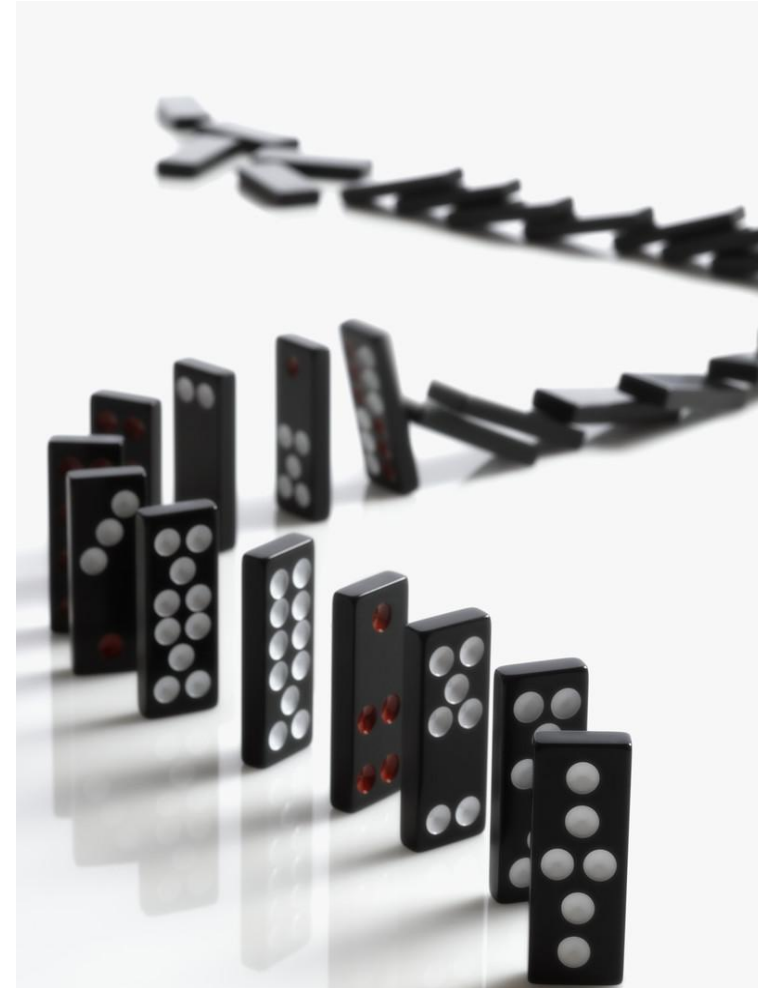
eMail



Etc...



- If utilisation on a one app per server basis is $<10\%$...
 - Utilisation on a “N+1” system is $<5\%$
- This is still a local BC solution
 - If the data centre is taken out, local “N+1” has done nothing
- If this is replicated over a distance, utilisation becomes even worse
 - $<4\%$ for an “N+1+1” approach
 - $<2.5\%$ for an “2(N+1)” approach



- Take the one app per server starting point
- Use a shared BC server approach
 - At a local level, highly unlikely that many apps will fail at the same time
 - At a remote level, retaining some level of capability is key
- Use virtualisation to provide flexible capabilities
 - Needs can be resized and shared



ERP



CRM



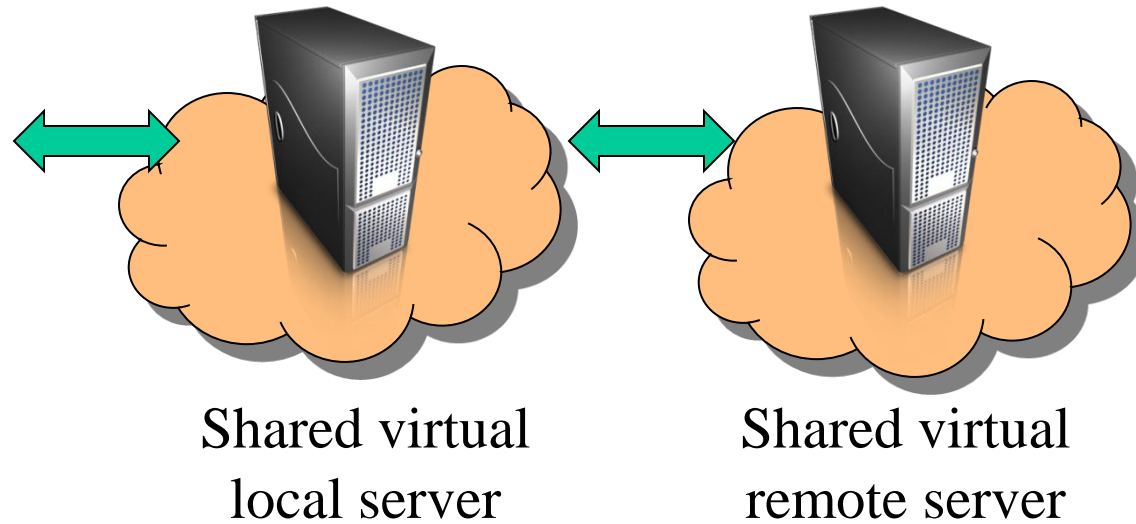
eMail



HR



Etc...





- Improvement in BC utilisation rates
- Local and long distance BC provided
- When combined with a “many apps per server” approach, utilisation rates can be driven up
 - But – only virtualise what is suitable for virtualising – and what you are happy with

- Maintaining state
 - Any failure must be essentially transparent to the user
 - Knowledge of where the user's compute state is
 - Speed of failover – seconds, not minutes/hours
- Maintaining transaction
 - Loss of transaction state must be avoided
 - Long distance, latency issues
- Knowledge of architecture is required
 - Context of environment, capability to work across mixed server assets

- BC is far more important than DR
 - BC keeps the business going – DR tries to stop it from going out of business
- A full “N+M” approach is overly expensive
 - And hits any green credentials the business is aiming for
- Fractional “N+M” provides the most effective, efficient approach
 - Optimises utilisation, provides the greatest continuity
- The technology underpinning this has to be solid
 - Maintenance of state