PASS

PROFESSIONAL ASSOCIATION
FOR SQL SERVER

PASS Deutschland e.V.

100% Verfügbarkeit bei 24 Stunden Wartungsfenster pro Tag

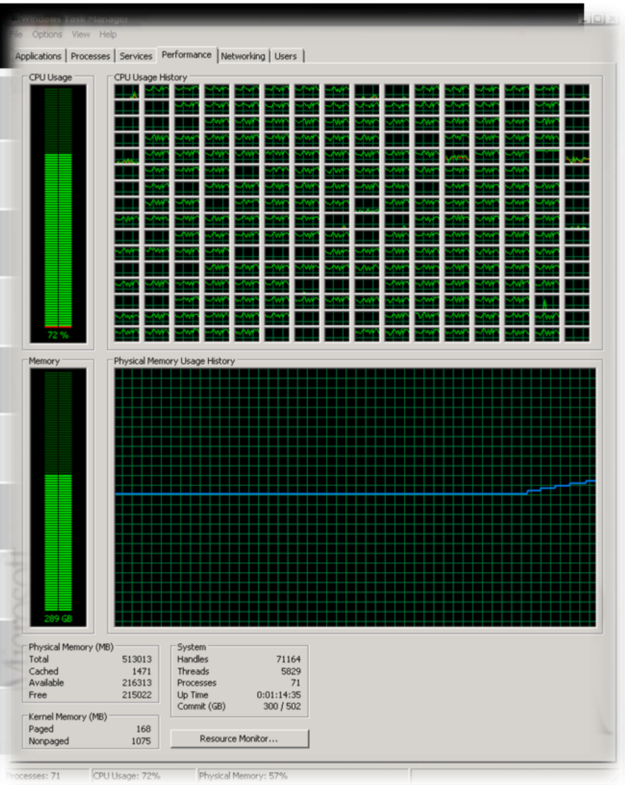
Original Title

100% Availability without 100% Uptime

=tg= Thomas Grohser

select * from =tg=

@@Version	Remark
SQL 4.21	First SQL Server ever used (1994)
SQL 6.0	First Log Shipping with failover
SQL 6.5	First SQL Server Cluster (NT4.0 + Wolfpack)
SQL 7.0	2+ billion rows / month in a single Table
SQL 2000	938 days with 100% availability
SQL 2000 IA64	First SQL Server on Itanium IA64
SQL 2005 IA64	First OLTP long distance database mirroring
SQL 2008 IA64	First Replication into mirrored databases
SQL 2008R2 IA64	First 256 CPUs & >500.000 STMT/sec
SQL 11 (Denali)	Can't wait to push the limits even further



Physical Memory (MB)	
Total	513013
Cached	1471
Available	216313
Free	215022

Kernel Memory (MB)	
Paged	168
Nonpaged	1075

System	
Handles	71164
Threads	5829
Processes	71
Up Time	0:01:14:35
Commit (GB)	300 / 502

Processes: 71 | CPU Usage: 72% | Physical Memory: 57%

Thomas Grohser, SQL Server MVP, bwin Interactive Entertainment AG

http://www.grohser.com/

Focus on SQL Server Infrastructure Architecture and Implementation

Close Relationship with Microsoft

- SQLCAT (SQL Server Customer Advisory Team)
- SCAN (SQL Server Customer Advisory Network)
- TAP (Technology Adoption Program SQL2008R2 and SQL11)

Active PASS member and PASS Summit Speaker

MVP Microsoft® Most Valuable Professional

SQL CAT

PASS PROFESSIONAL ASSOCIATION FOR SQL SERVER

World’s biggest publicly listed online gaming platform

World’s **leading provider** of online Sports Betting

One of the largest **Poker networks**

Comprehensive range of **Payment Service Providing**

Integrated gaming portal - **22 languages,**
25 core markets

Gross gaming revenues 2008 (GGR):
EUR **421 million**

More than **20 million registered customers**

1,500 employees

bwin builds on the strengths of the web in order to **tie up responsibility and gaming**

15 million page views and up to **980,000 users a day**

Agenda

- **Introduction to the challenge**
- **Theoretical solution**
- **Practical implementation**
- **Other problems we solved too**
- **Q+A**

ATTENTION: Important Information may be displayed at any slide at any time!

The Challenge

- **Globally operating companies**
  - Global websites
  - Offices around the globe
- **They all face the challenge that there is no time left to perform service and maintenance on the database systems.**
  - More and more patches to apply
  - Less and less time to maintain

The Challenge

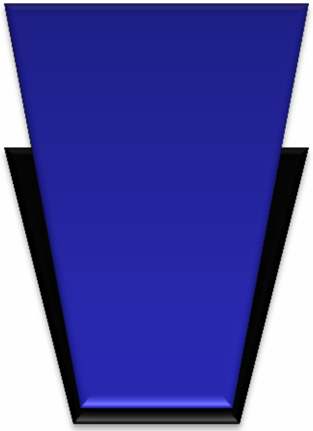
- **Every day of the year we must operate 24h per day without interruptions and do so faster and faster**

- **Every day of the year we would like to maintain our systems 24h per day.**

Non working ideas

- **Scale up**
  - – Solves partially performance
- **Scale out by hash(UserID or UserName)**
  - – Solves only availability and performance

We need a glass that can hold twice as much content than it has volume…



Solution

- **Impossible to do with a technical solution?**
- **So we need a political solution**

Yes we can

$^{\text{©}}$ Copyright and $^{\text{tm}}$ Trademark by some politician from Hawaii … mahalo

Solution

- **What does a good politician do**
  - – Bend the truth

    and
  - – Let the people solve the problem.

- **And this is exactly what we will do…**

Solution

- **We promised a solution that is available 24 hours per day**

- **What does available mean?**
  - A system is considered available if **all users** of the system at any time they desire to use the system can do so successfully
  - We split the rule
    - A system is considered available by **a user** when the user of the system at any time he or she desires to use the system can do so successfully
    - If **each** individual **user** considers the system available the whole system is considered available

Solution

- **Facts**
  - Our users are human beings
    (People on the web or in an office)
  - And all people have some basic needs
    - Eating
    - Sleeping
  - The first on is not helping us much but the second …
  - Humans need at least 4 hours of sleep and most sleep more like 6 to 10 hours a day.
  - And the greatest fact is that even if they travel the globe they somehow keep some of the hours of sleep and inactivity constant.

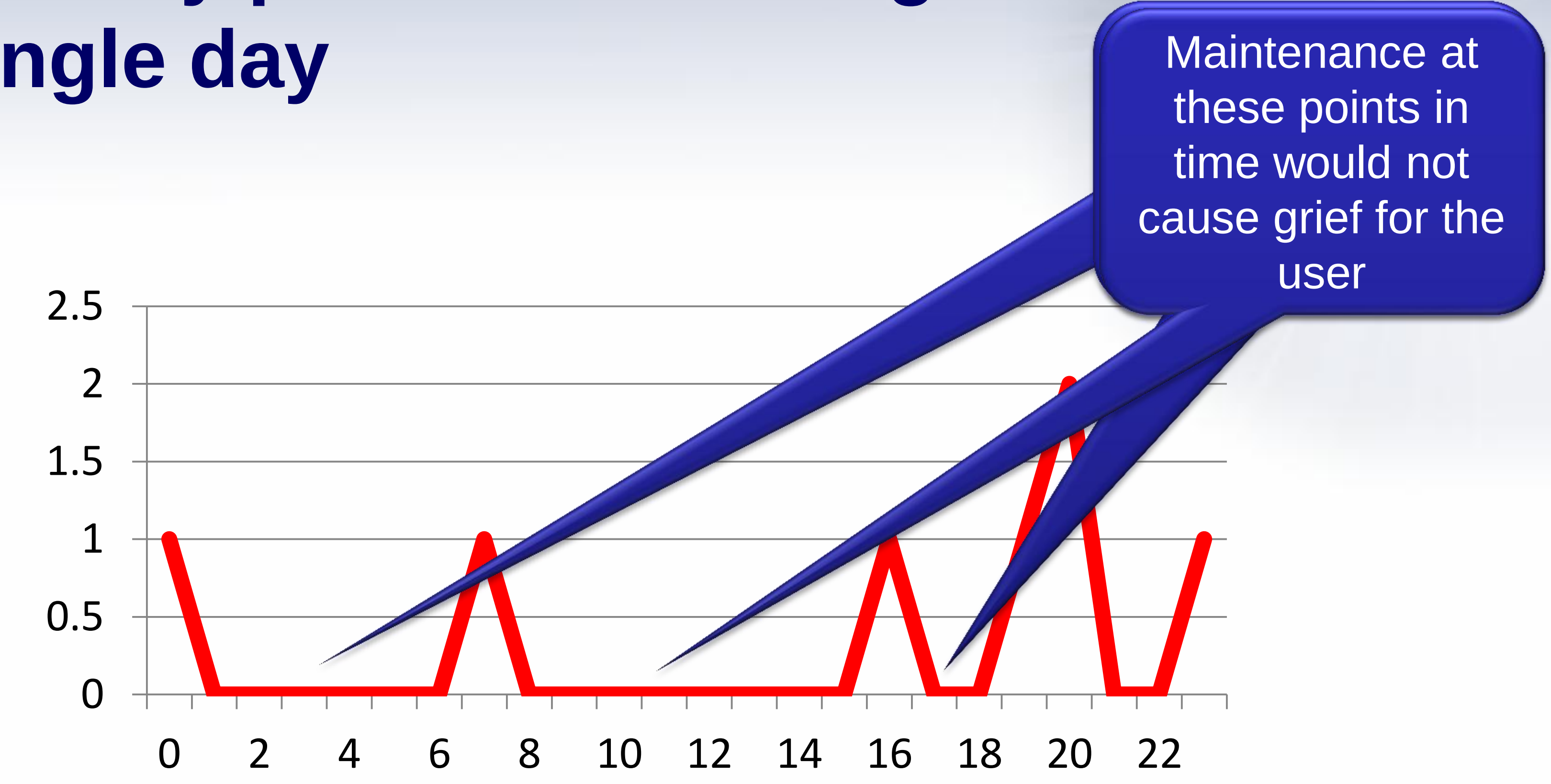
Solution

- So lets bend the facts into the rules

- If we group our users in a way that all people with the same inactivity pattern are in one group
- And place each group on its own server
- We just solved the problem…
- Easy isn’t it?

Theoretical Solution

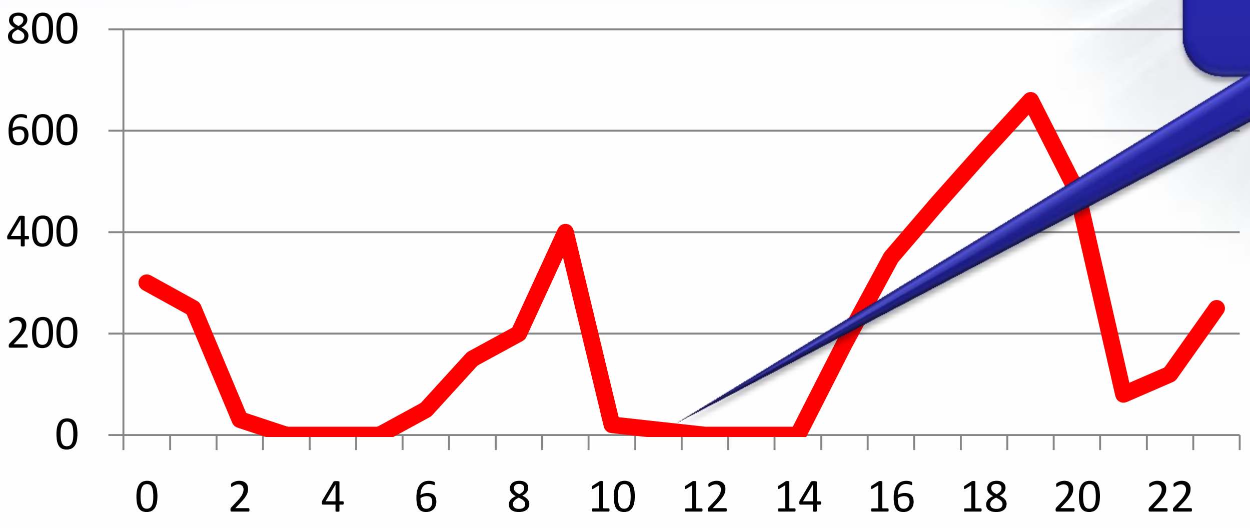
- **Activity per hour of a single user on a single day**



Maintenance at these points in time would not cause grief for the user

Theoretical Solution

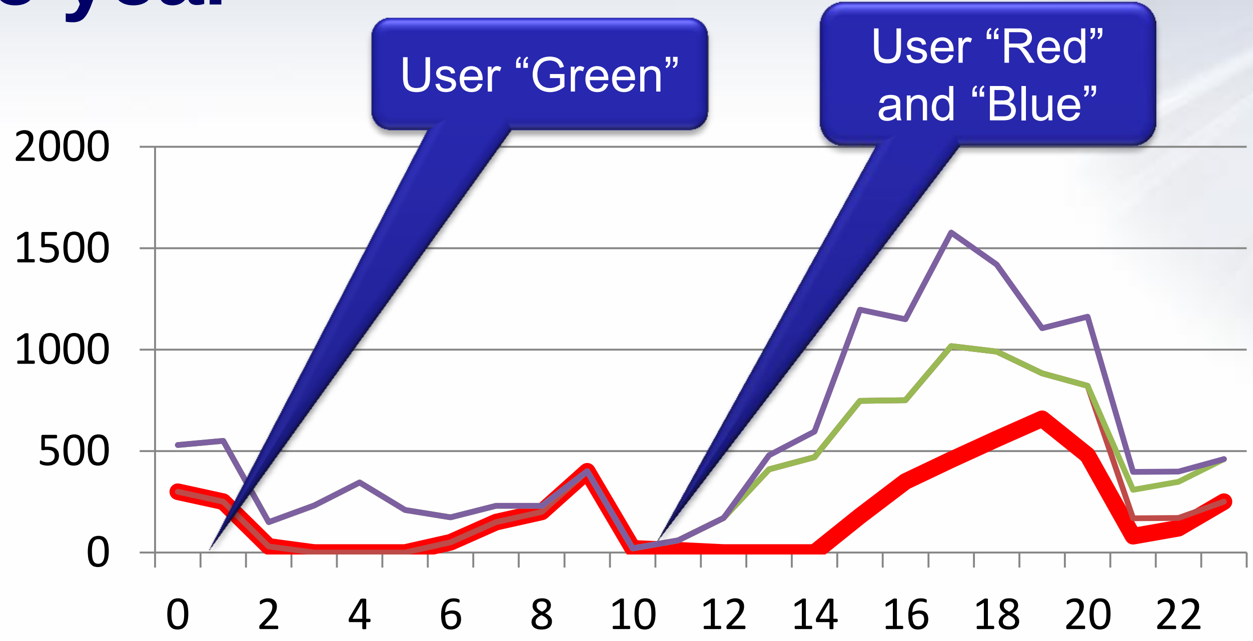
- **Activity per hour of a single user in one year**



Still time for maintenance available

Theoretical Solution

- **Activity per hour of multiple users in one year**



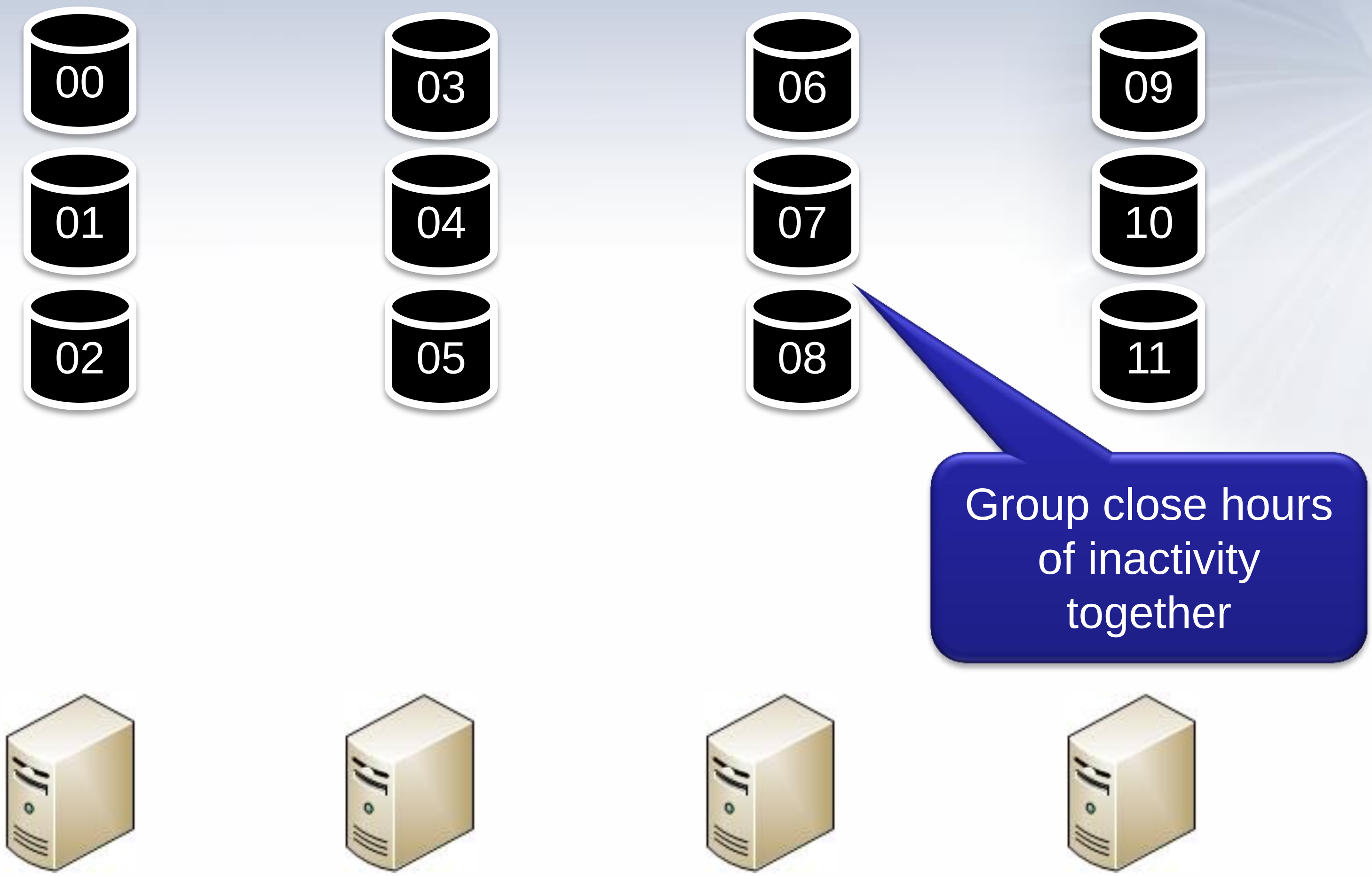
Some Statistics

- **Large globally operating website with millions of users and several years of operational data to analyze**
- **99.99% of the users have at least one constant inactive hour over all the years**
- **99.99% of the users are have more than one constant inactive hours over the last 2 years**
- **100% of the users have at least one constant inactive hour over the last year**

Practical Solution

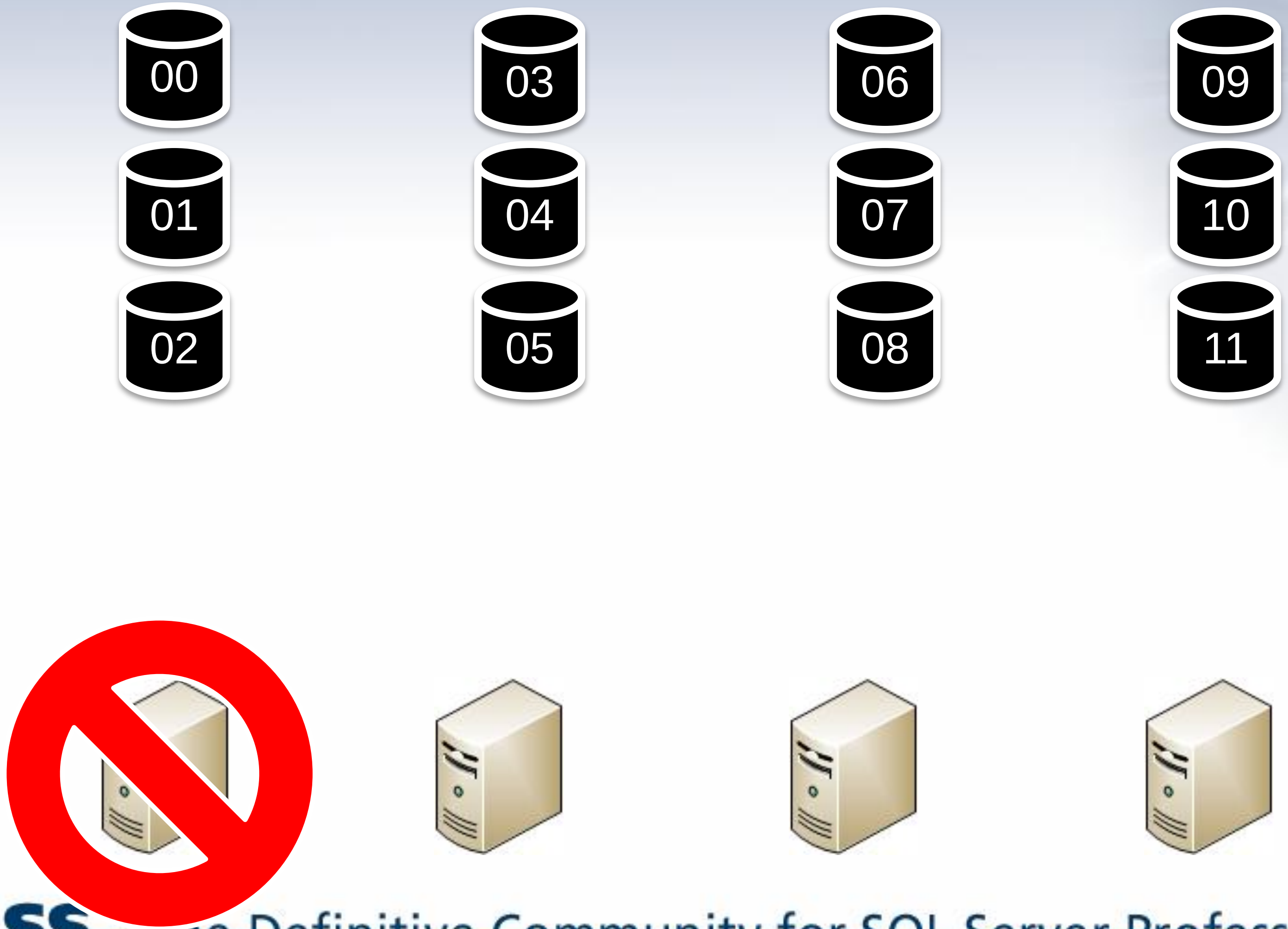
- **24 Partitions**
  - – One for each hour of the day
  - – 2 brick groups with 12 Partitions each
  - – 4 bricks with 3 Partitions each

Why 12 Partitions on 4 Servers

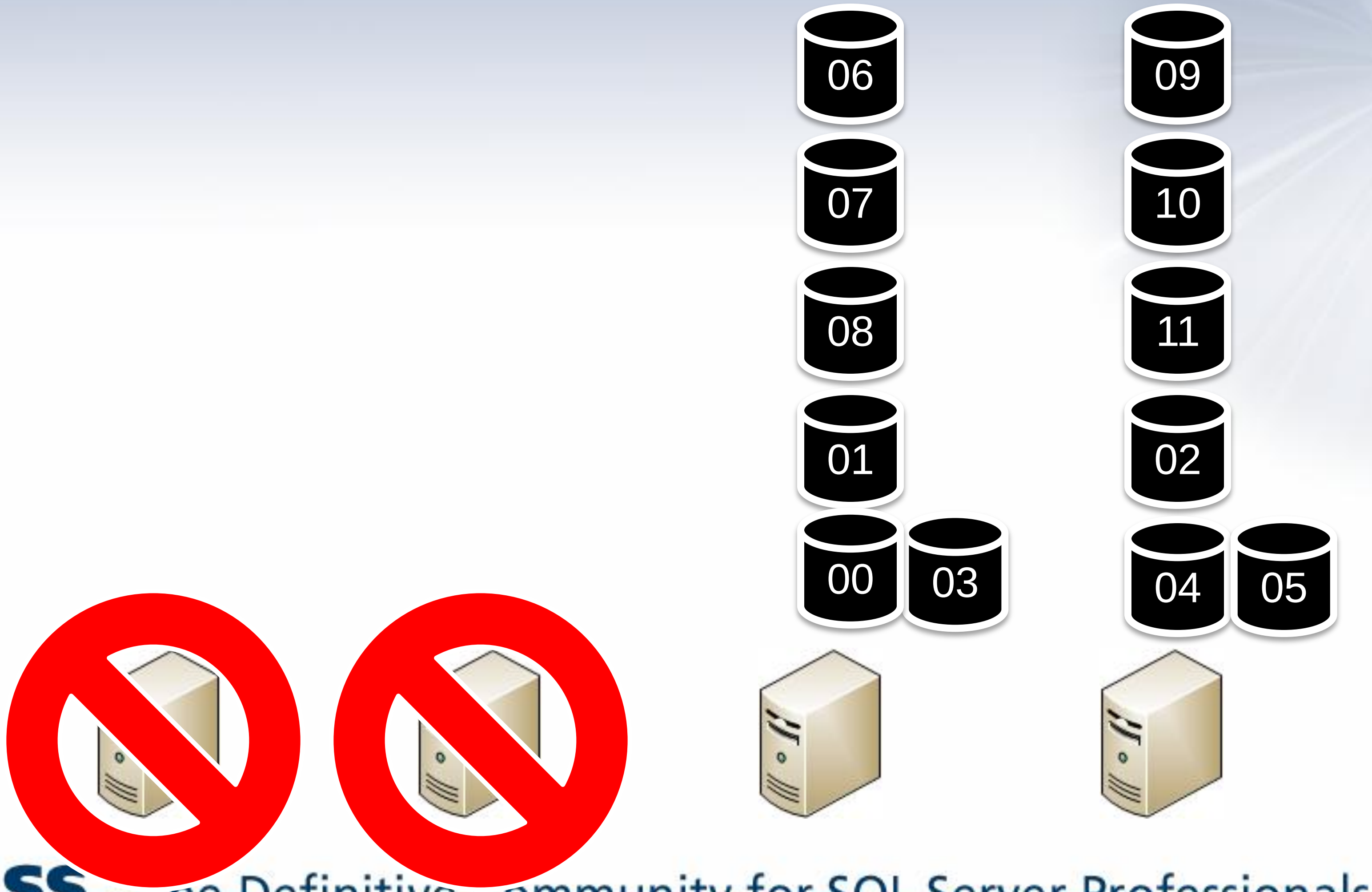


Group close hours of inactivity together

Why 12 Partitions on 4 Servers



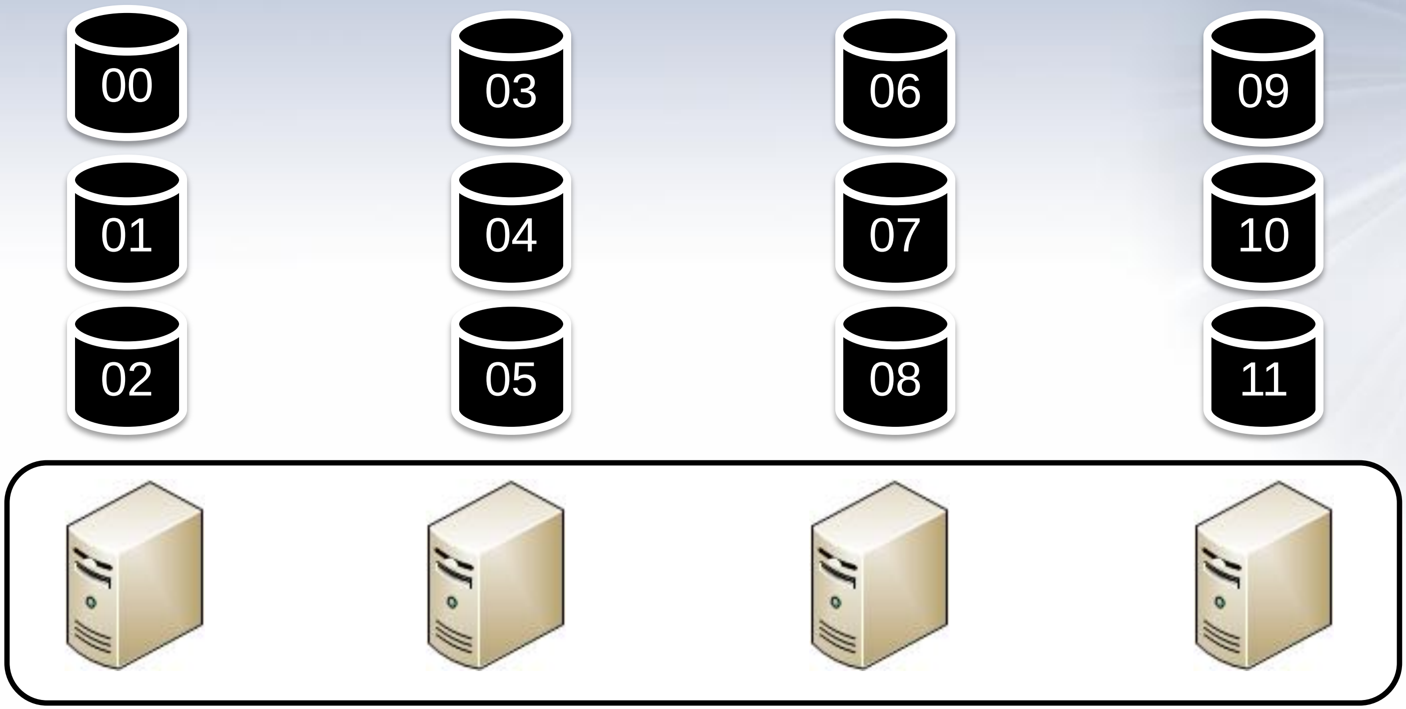
Why 12 Partitions on 4 Servers



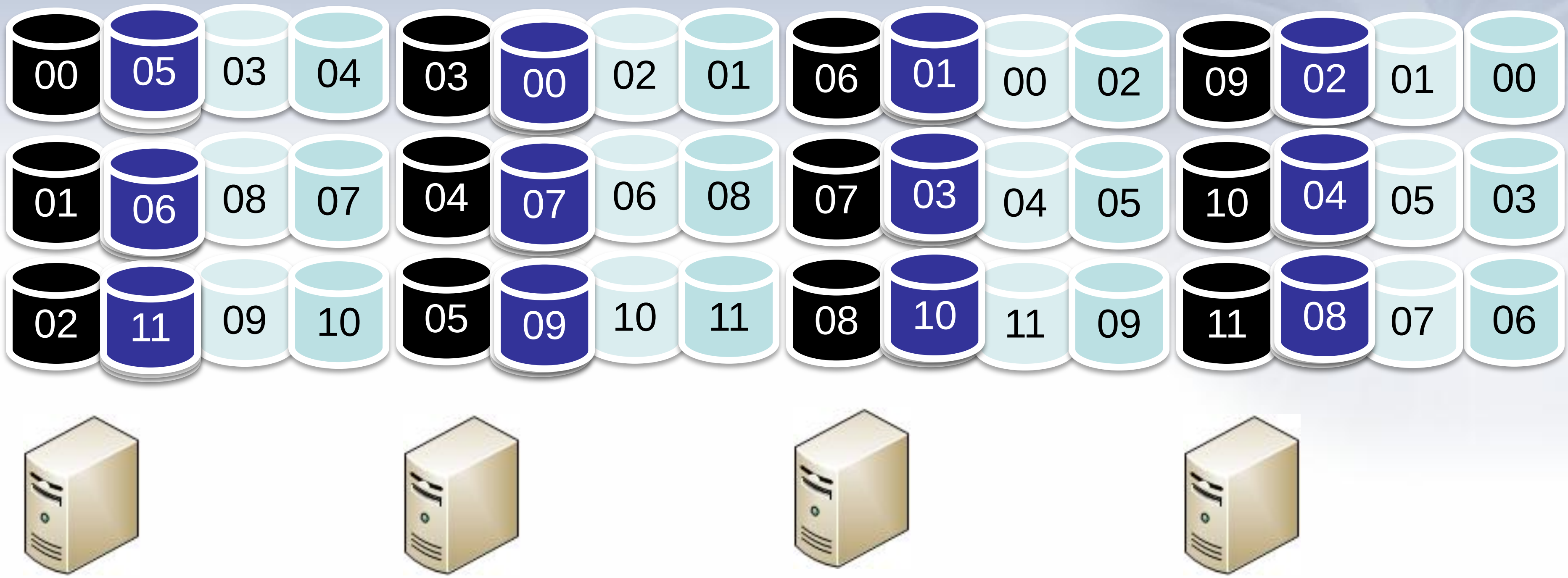
Clustering or Mirroring/Log Shipping

- **Clustering**
  - – Data only available once = Single point of failure
- **Mirroring/Log Shipping**
  - – Out of the box not as automatic as clustering
  - – Limited support for Replication

Cluster



Mirror & Log Shipping



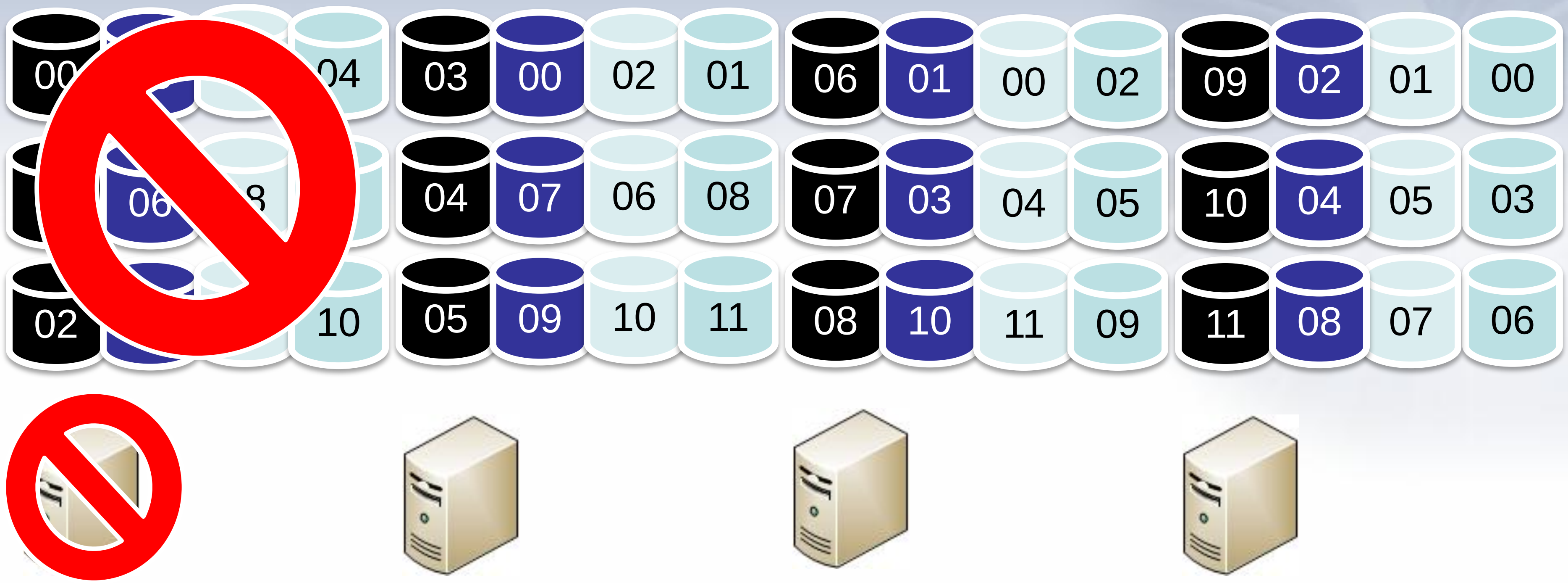
nn Principal database

nn Mirrored database

nn Log shipped database

nn Log shipped database 2nd copy

Mirror & Log Shipping



00			04	03	00	02	01	06	01	00	02	09	02	01	00
	06	8		04	07	06	08	07	03	04	05	10	04	05	03
02			10	05	09	10	11	08	10	11	09	11	08	07	06

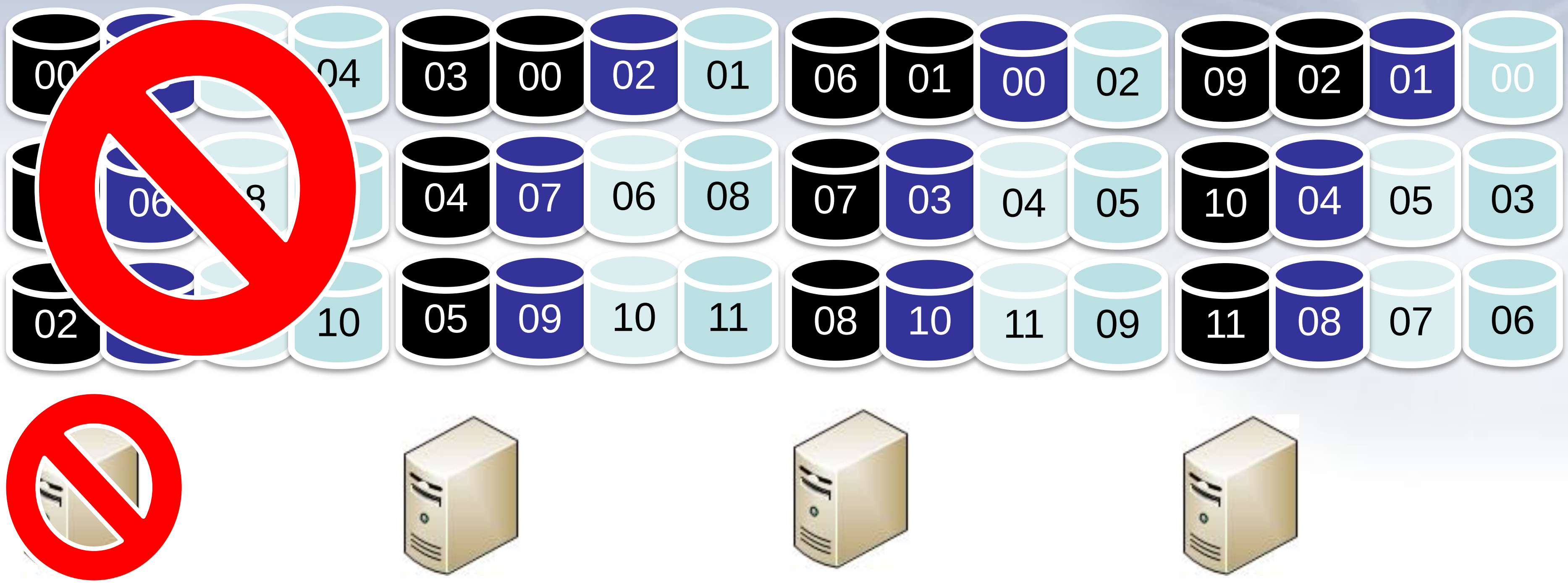
nn Principal database

nn Mirrored database

nn Log shipped database

nn Log shipped database 2nd copy

Mirror & Log Shipping



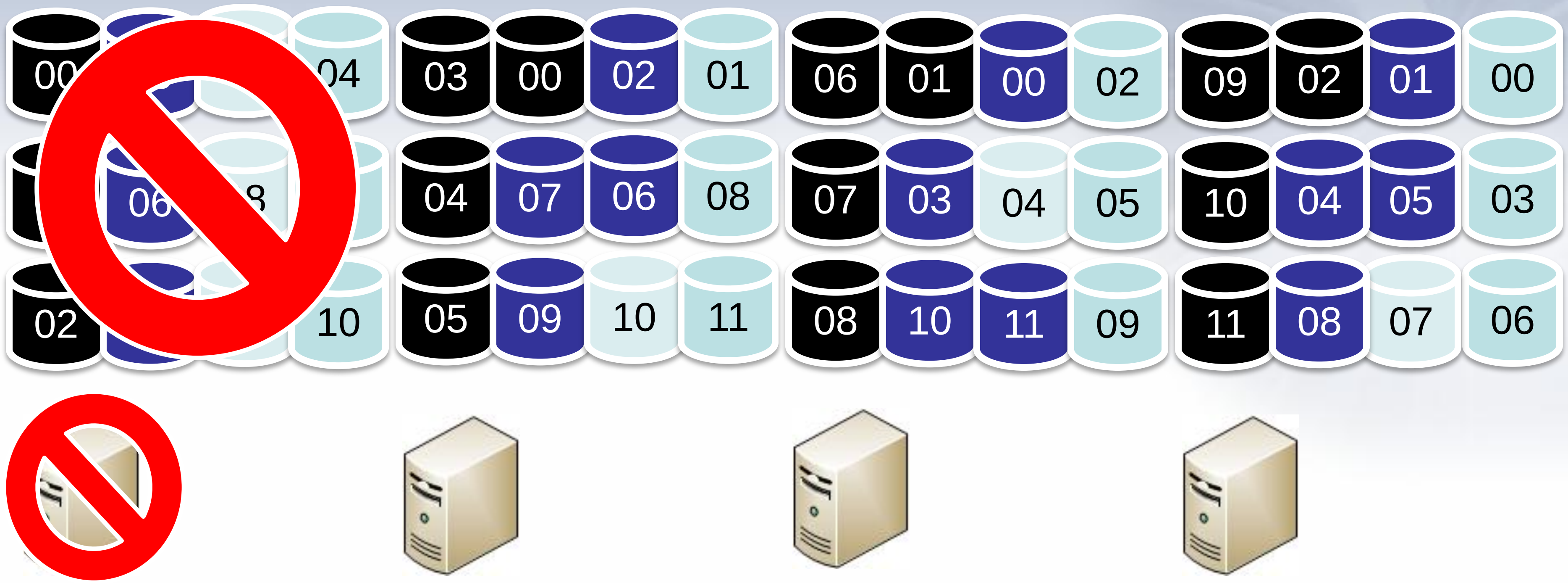
nn Principal database

nn Mirrored database

nn Log shipped database

nn Log shipped database 2nd copy

Mirror & Log Shipping



nn Principal database

nn Mirrored database

nn Log shipped database

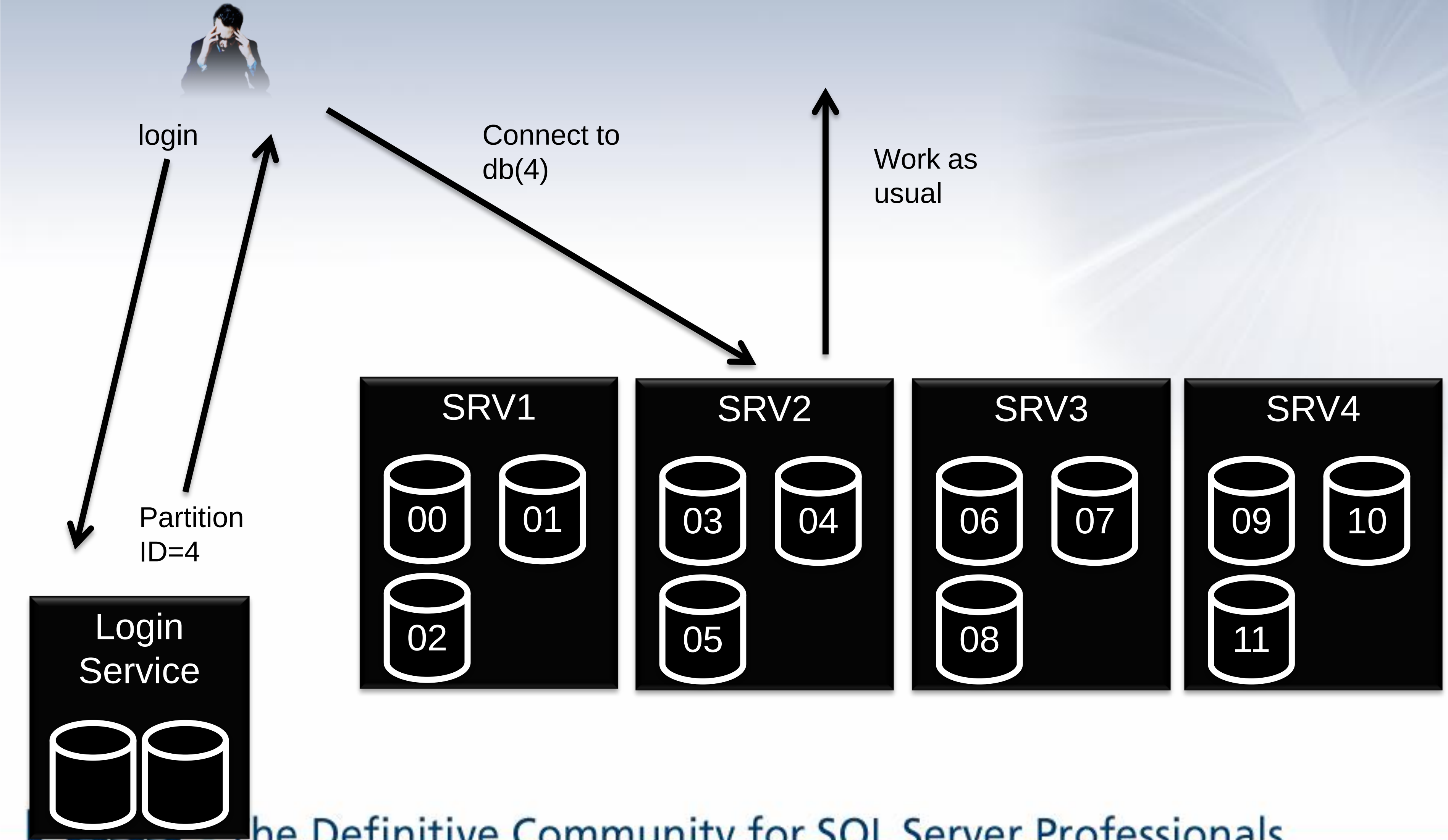
nn Log shipped database 2nd copy

Mirroring - Dual datacenter

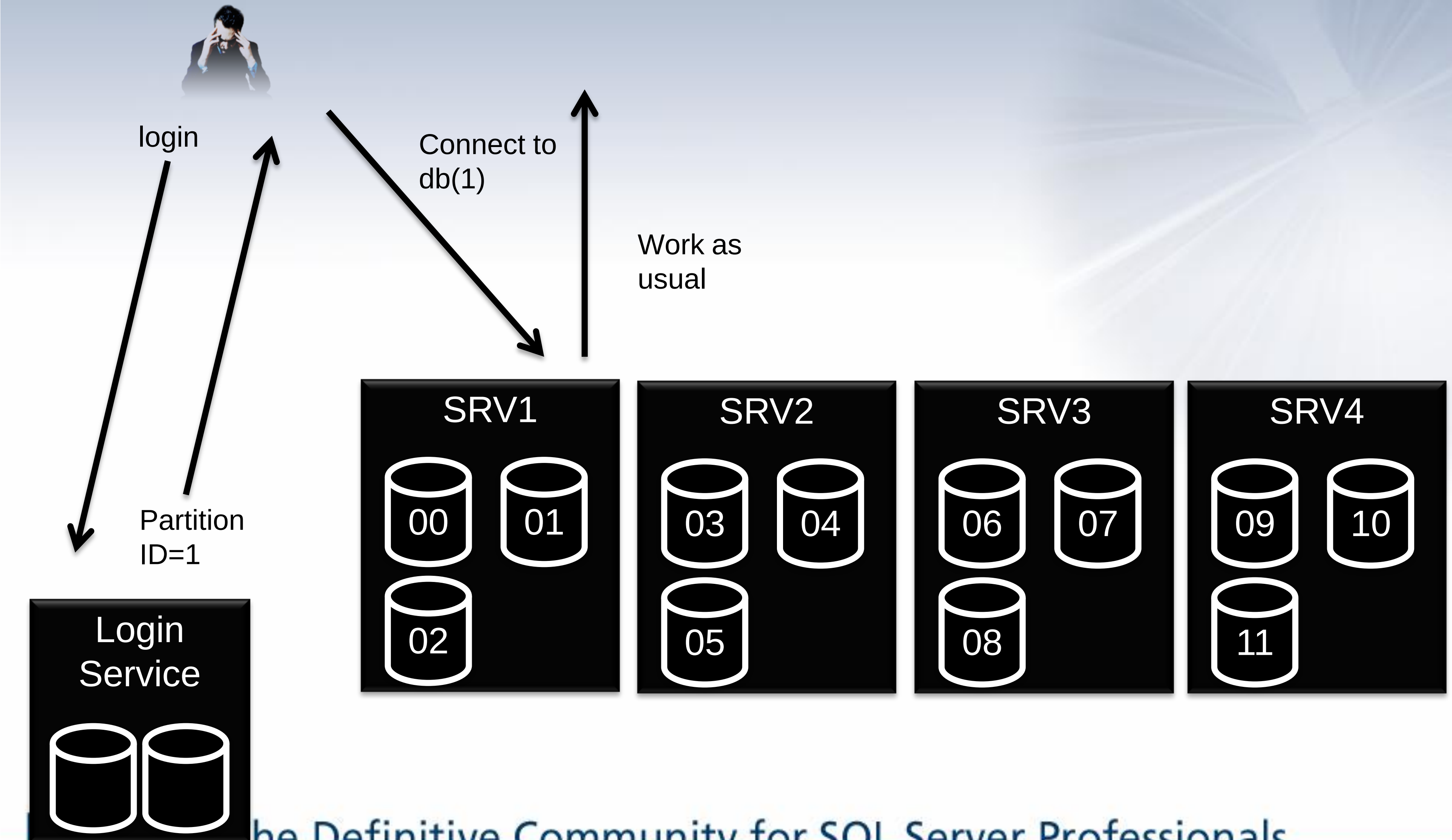
- **If you want the synchronous mirror to be always in the other datacenter it does not work with 4 servers and 12 partitions but works great with 8 Servers and 24 partitions**

- **Have fun figuring out the distribution matrix**

How does it work



Other user



login

Connect to db(1)

Work as usual

Partition ID=1

Login Service

SRV1

00 01 02

SRV2

03 04 05

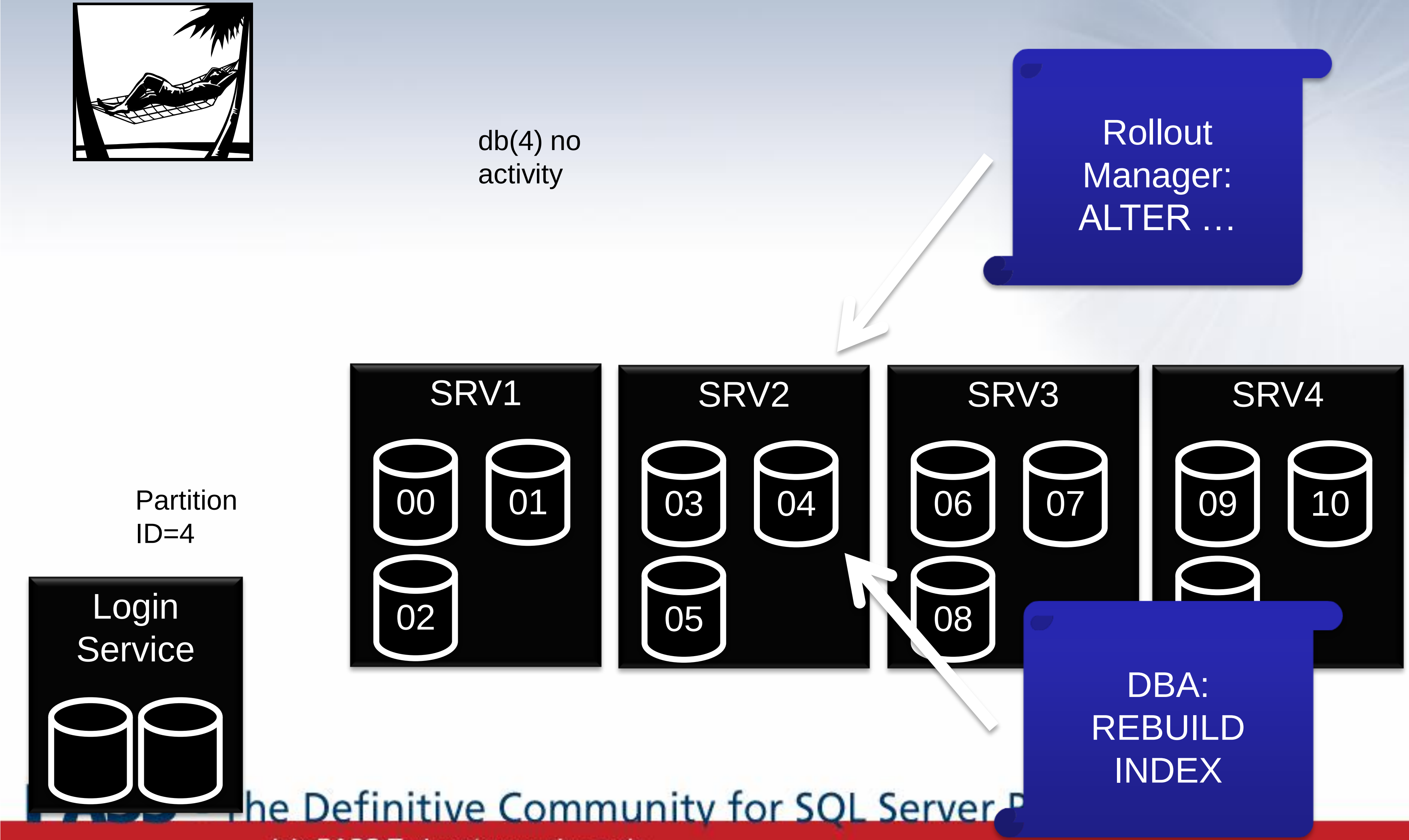
SRV3

06 07 08

SRV4

09 10 11

User sleeps…



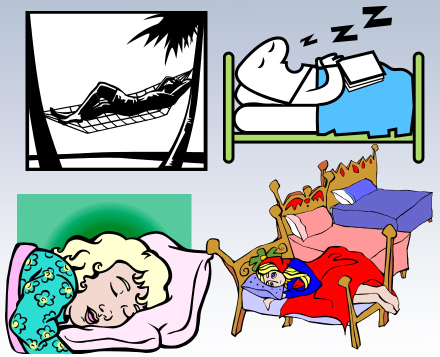
db(4) no activity

Rollout Manager: ALTER …

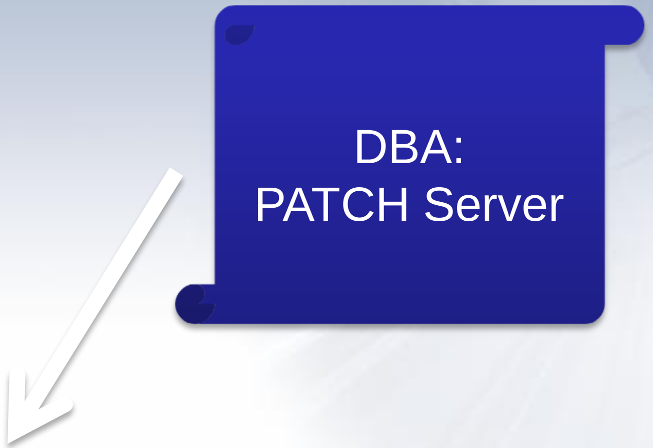
Partition ID=4

DBA: REBUILD INDEX

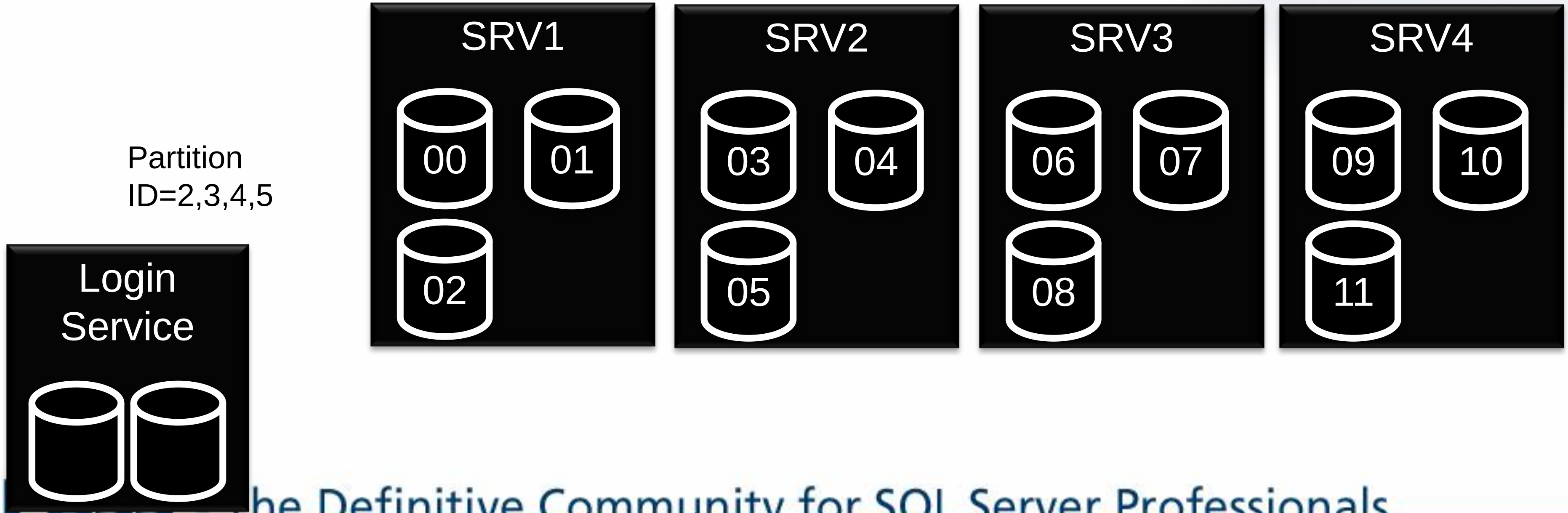
Many users sleep…



db(3 & 4 & 5) no activity



DBA:
PATCH Server



Partition ID=2,3,4,5

SRV1

00 01 02

SRV2

03 04 05

SRV3

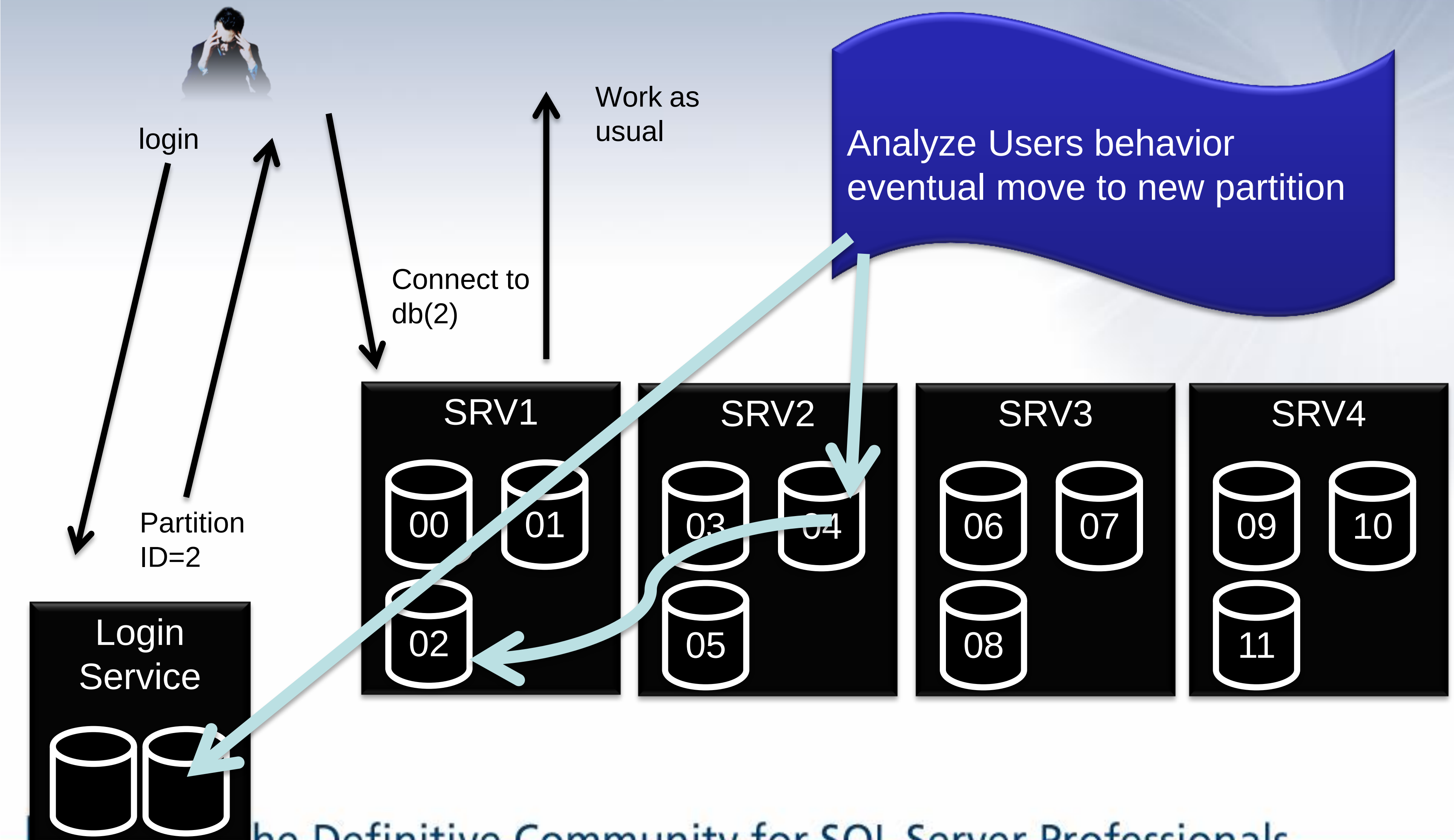
06 07 08

SRV4

09 10 11

Login Service

Repartition Users



Implementation Details

- **Identity Columns**
  - GUID
    - Large, very bad for indexes
  - Ranges per partition
    - 100.000.000 to 199.999.999 partition 1
    - 200.000.000 to 299.999.999 partition 2
    - … …
    - Large numbers from the beginning
  - Renumber every time you move a customer
    - Your DWH people will hate you…

Implementation Details

- **IDENTITY Columns**
  – Reverse Ranges
  - Partition 1

```
CREATE TABLE MyData
(ID bigint IDENTITY(1, 65536))
```

  - Partition 2

```
CREATE TABLE MyData
(ID bigint IDENTITY(2, 65536))
```

  - …

Implementation Details

- **Global static data (Countries, ZipCodes, …)**
  - Replicate/copy from one ”master” source to each database or keep on separate server
- **Global dynamic data (ProductCatalog, …)**
  - Keep on separate server or replicate/copy from one ”master” source to each database
- **Customer data (Transactions, Invoices, …)**
  - Partition with customer
- **Relationship data (FriendsOfFriends)**
  - Store on both ends

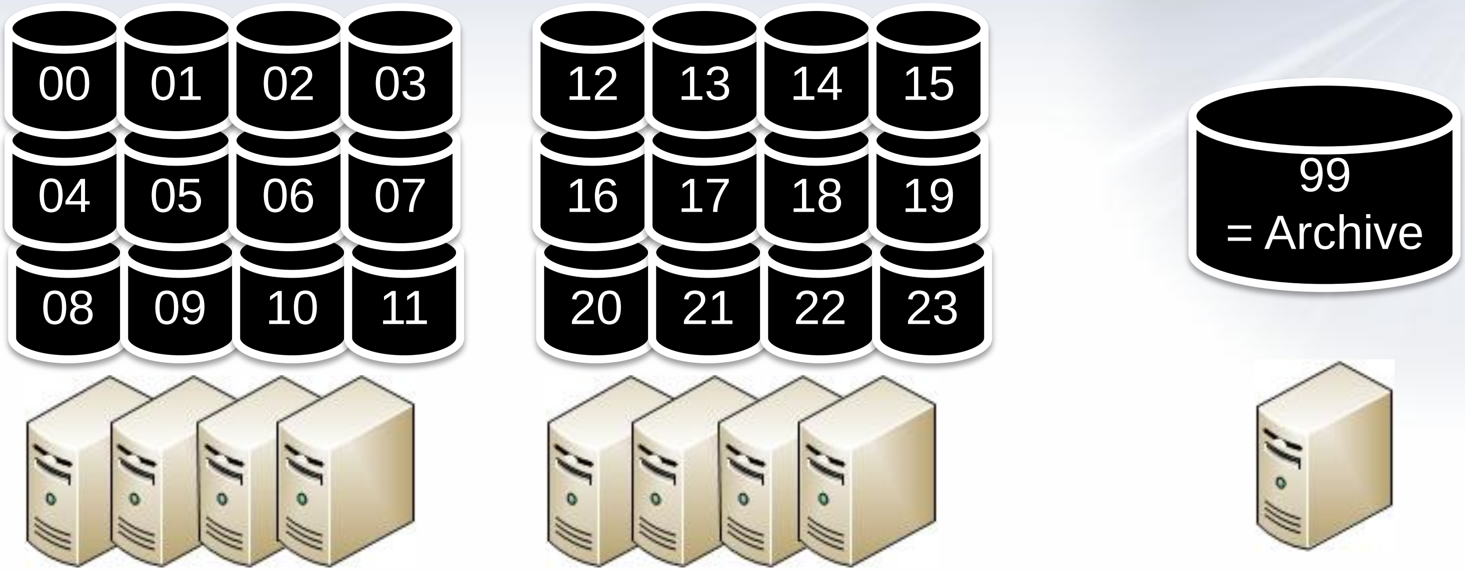
Other problems we solved too

- **Archiving**
- **Geographical placement of data**

Archiving

- Archiving OLTP data can be tricky and painful but management likes it because it saves money
- Create an extra Partition (the archive)
- Place the partition on a small server with large and inexpensive storage
- Move all inactive customers to this partition
- If the customer becomes active again move him back

Archiving



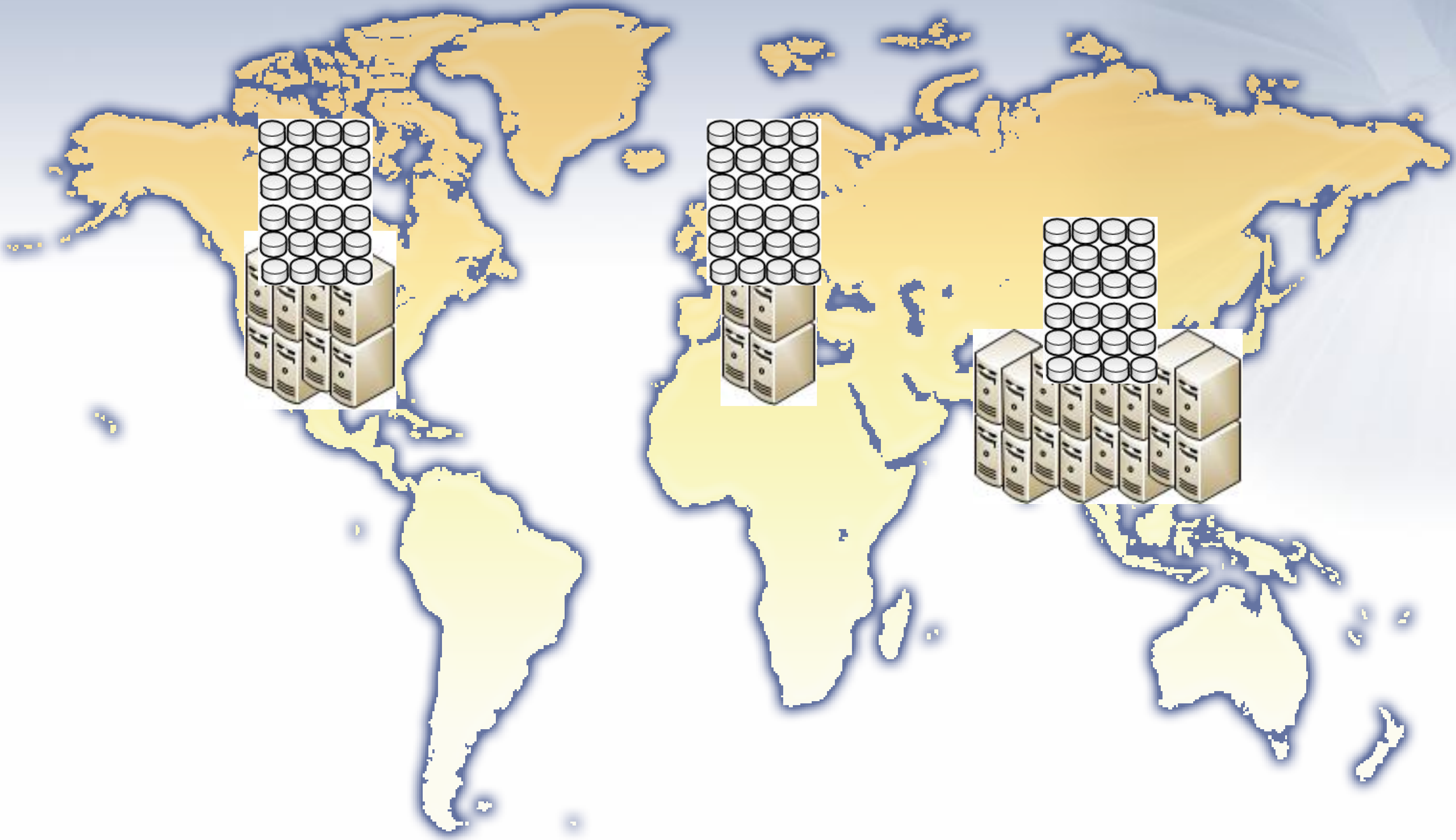
- High performance
- High availability
- “Low“ Capacity

- **Low performance**
- **Low availability**
- **High Capacity**

Geographical placement

- **Sometimes the requirement to place data at a specific location comes up.**
- **You can easily create extra partition(s) and place them on servers at a different location.**
- **For example: 24 Partitions on 4 Servers in Europe, 24 Partitions on 8 Servers in the US and 24 Partitions on 16 Servers in Asia**

Geographical placement



Just some Ideas

- **Have a friendly user partition for new releases and features**
- **Let the user choose its hour of inactivity in a configuration page/dialog**

Wrap Up

- **Find a way to group your data in active/inactive portions over time**
- **Separate the data physically by this grouping**
- **Scale Out**
- **Gain manageability**
- **Gain availability**

Questions?

=tg= Thomas Grohser

tg@grohser.com

http://www.grohser.com/