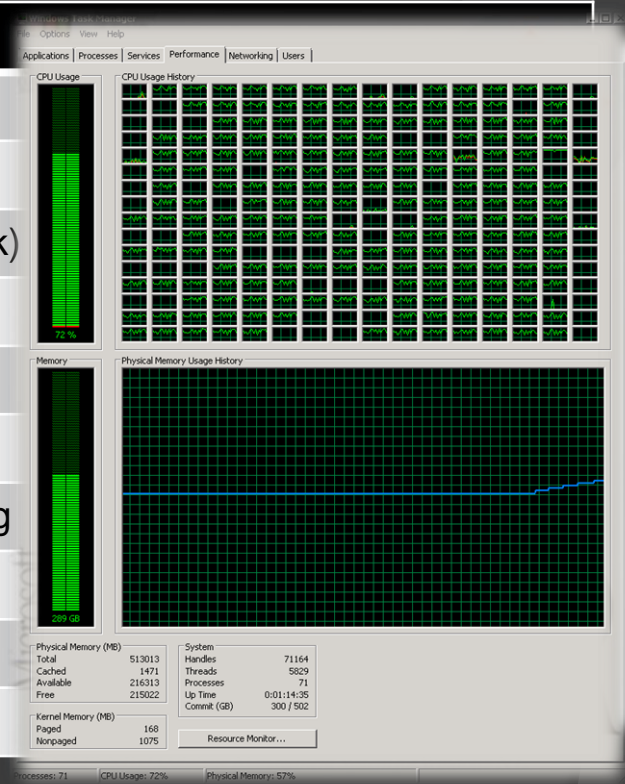


=tg= Thomas H. Grohser, bwin

# HOW TO GAIN PERFORMANCE ON SQL SERVER BY MAKING NUMA YOUR FRIEND

# 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	Can't wait to push the limits even further



## 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)

Close relationship with Hardware Vendors (Focus IA64)

Active **PASS** member and **PASS Summit Speaker**



# Agenda

- ⦿ What is NUMA?
  - Why is Memory Important?
  - How does NUMA work
- ⦿ How to Use It for SQL Server
  - Usage Examples
  - How to configure NUMA
  - How to avoid Context Switches
- ⦿ Wrap Up
- ⦿ Q&A
- ⦿ Drinks and Food

# What is NUMA?

- ⦿ **None**
  - ⦿ **Uniform**
  - ⦿ **Memory**
  - ⦿ **Access**
- 
- ⦿ **OK - but what the heck does that mean?**

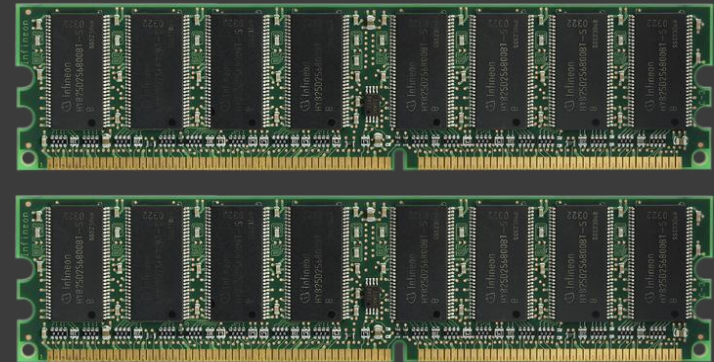
# Why is Memory Important?

- ◎ Memory is fast
  - Disk access is measured in ms
  - Memory access in ns
  - If snapping your finger once a second is memory access
  - Doing it every 11 days is disk access

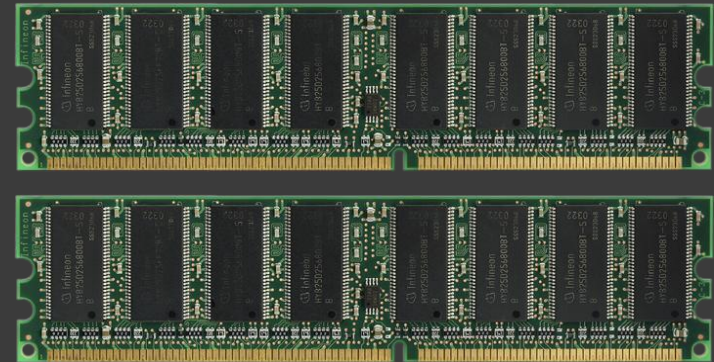
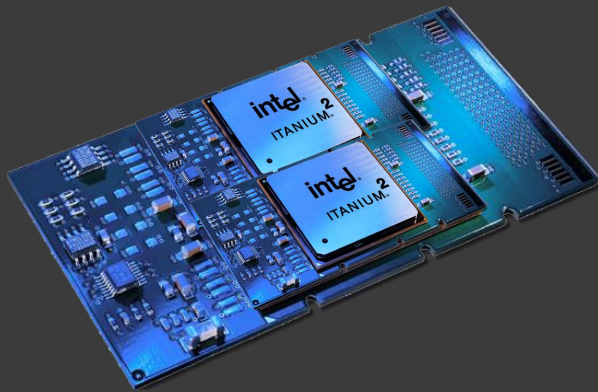
# Why is Memory Important?

- SQL Server is a in memory database
- All Query Requests and DML statements and even all DLL statements work only on data in memory
- If data required for the operation is not in memory it is loaded from disk into memory first
- Data is written back to disk at a later time
- Only the transaction log is persisted immediately to disk

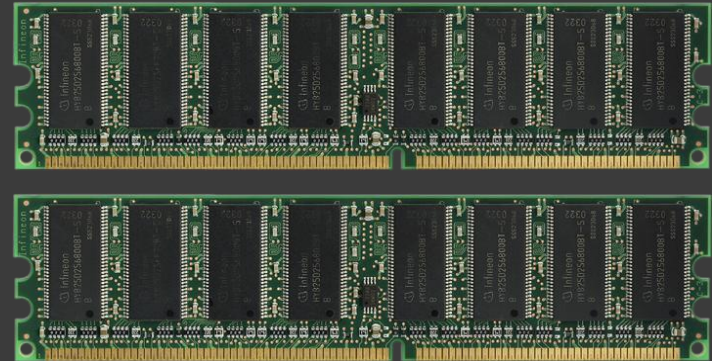
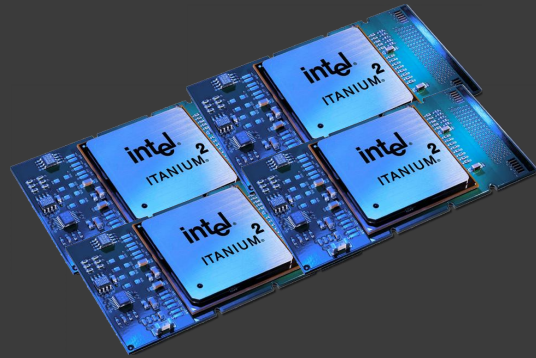
# Single CPU



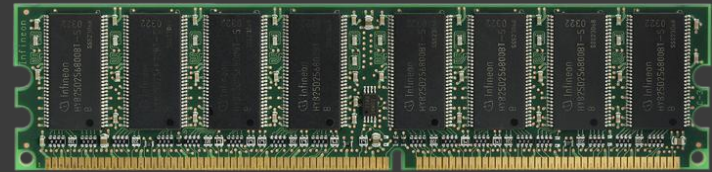
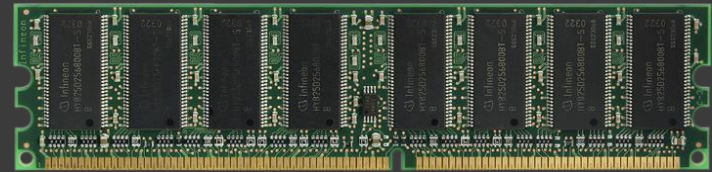
# Single CPU / Dual Core



# Single CPU / Multi Core



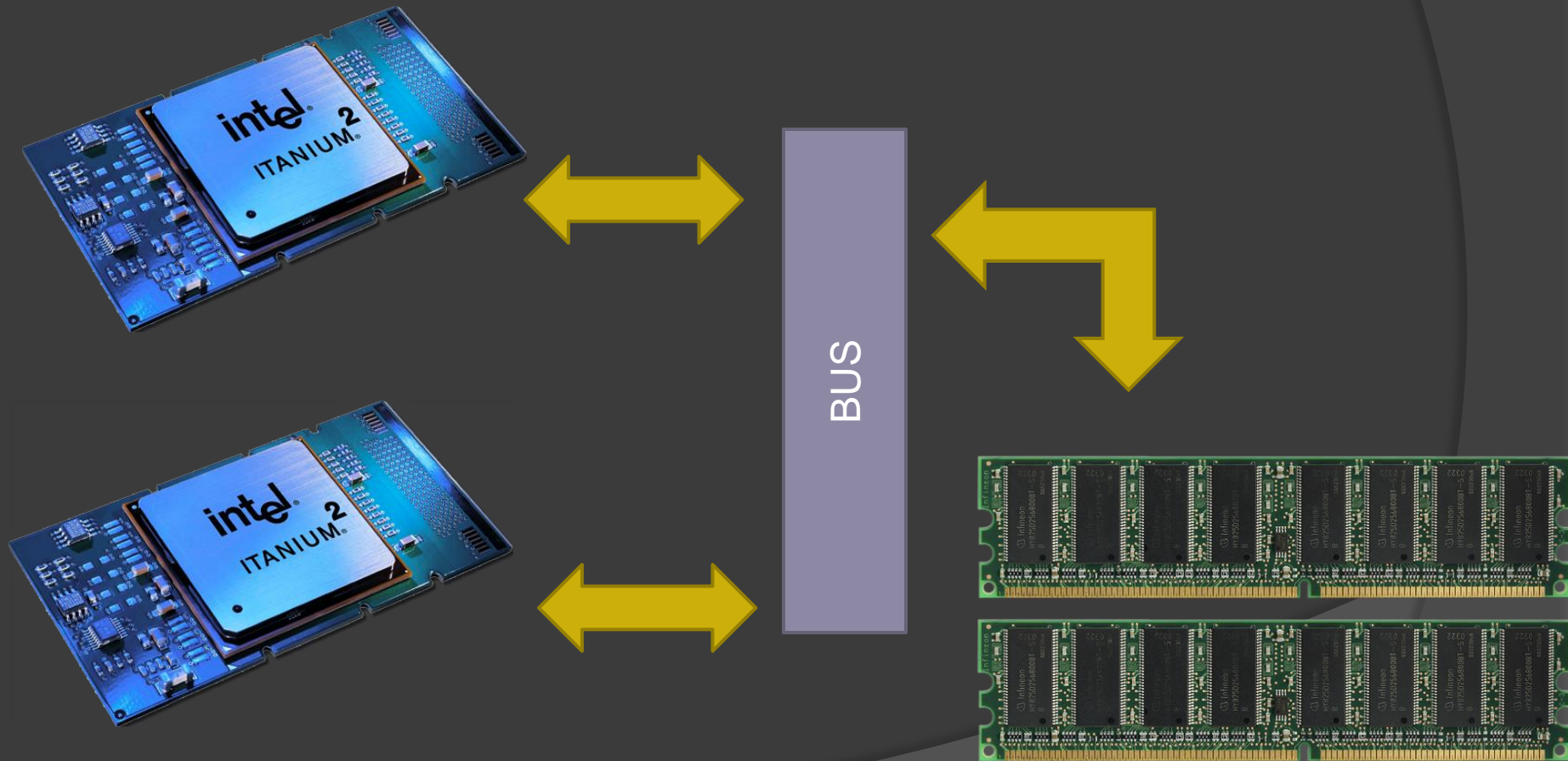
# Dual CPU



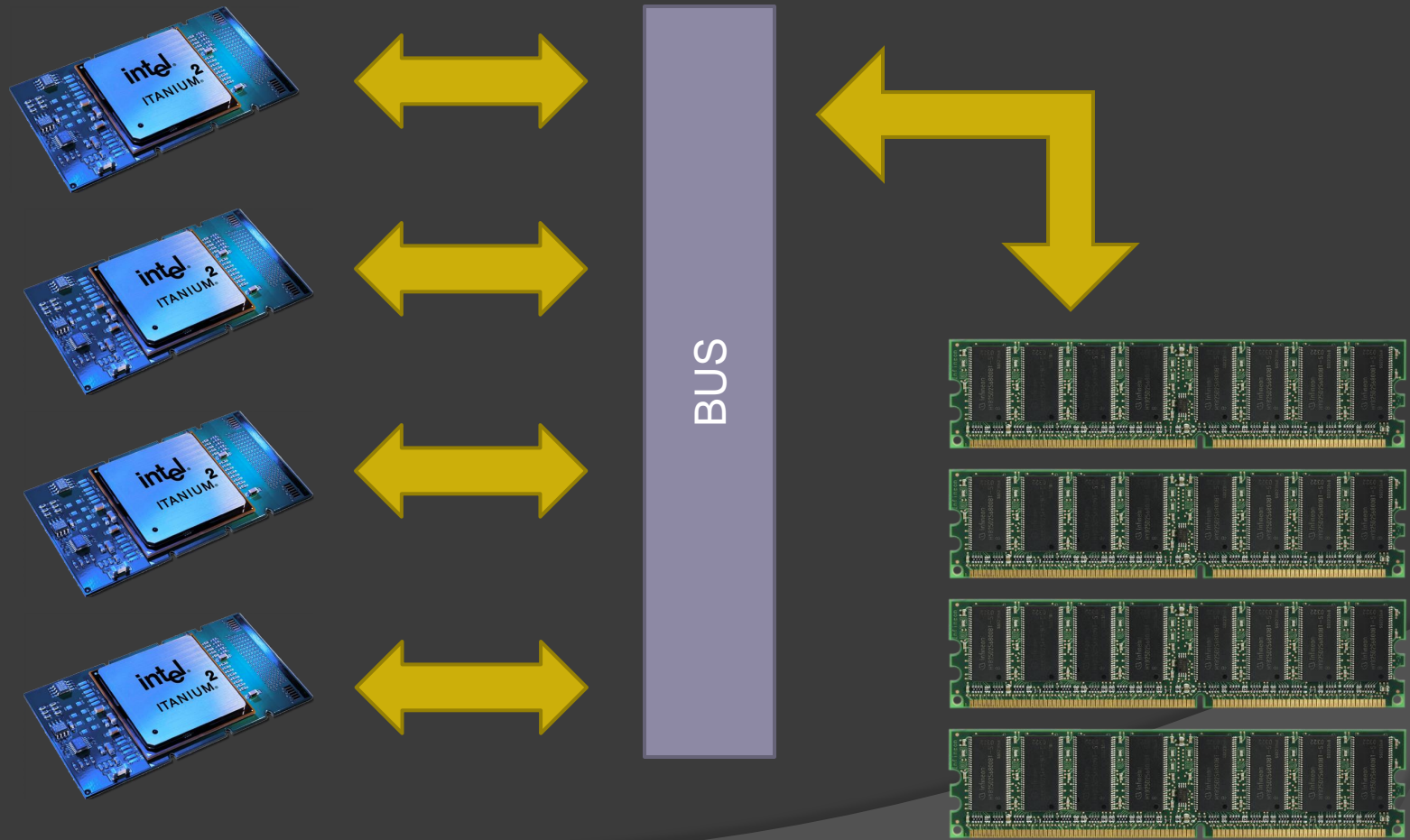
???



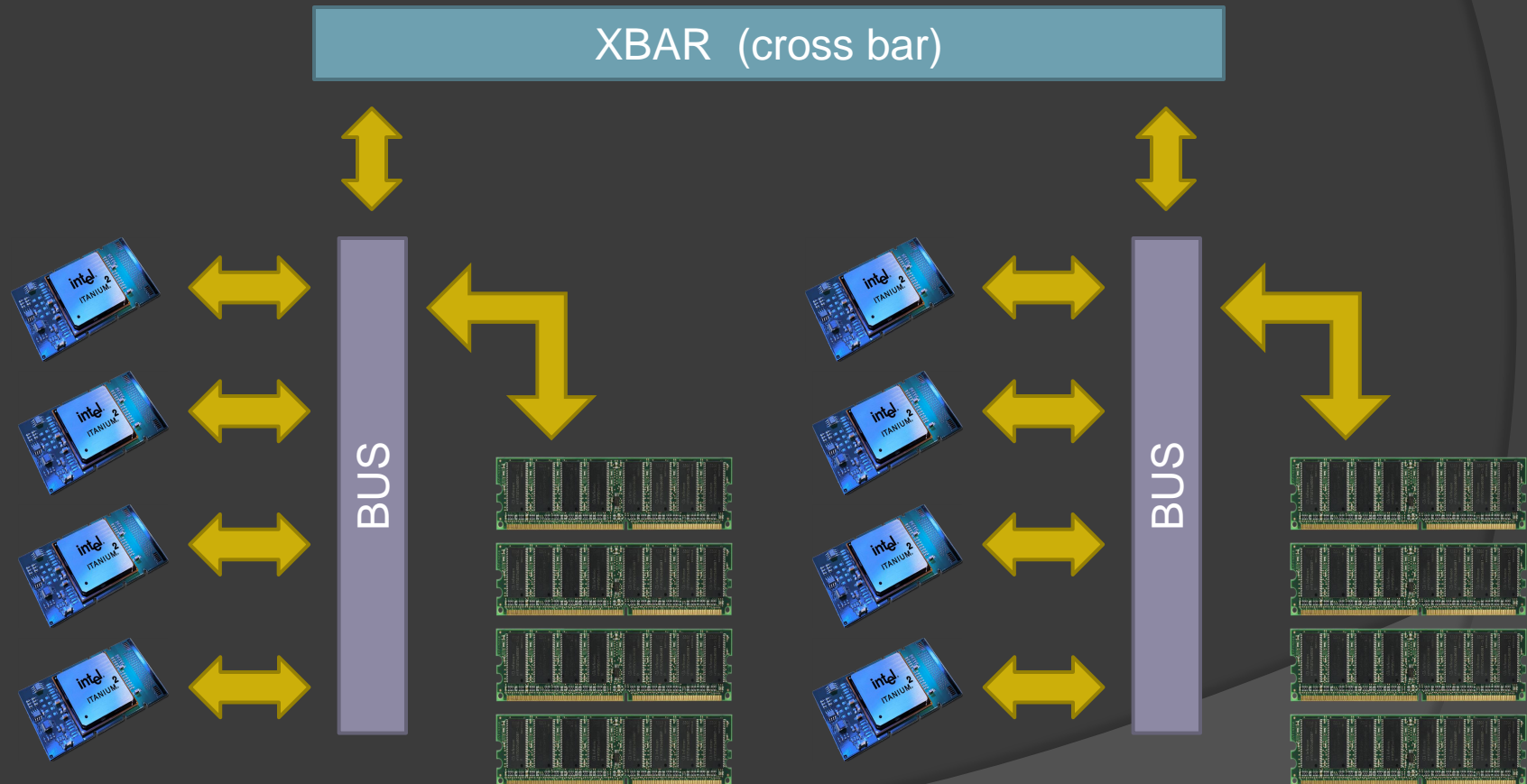
# Dual CPU / Solution



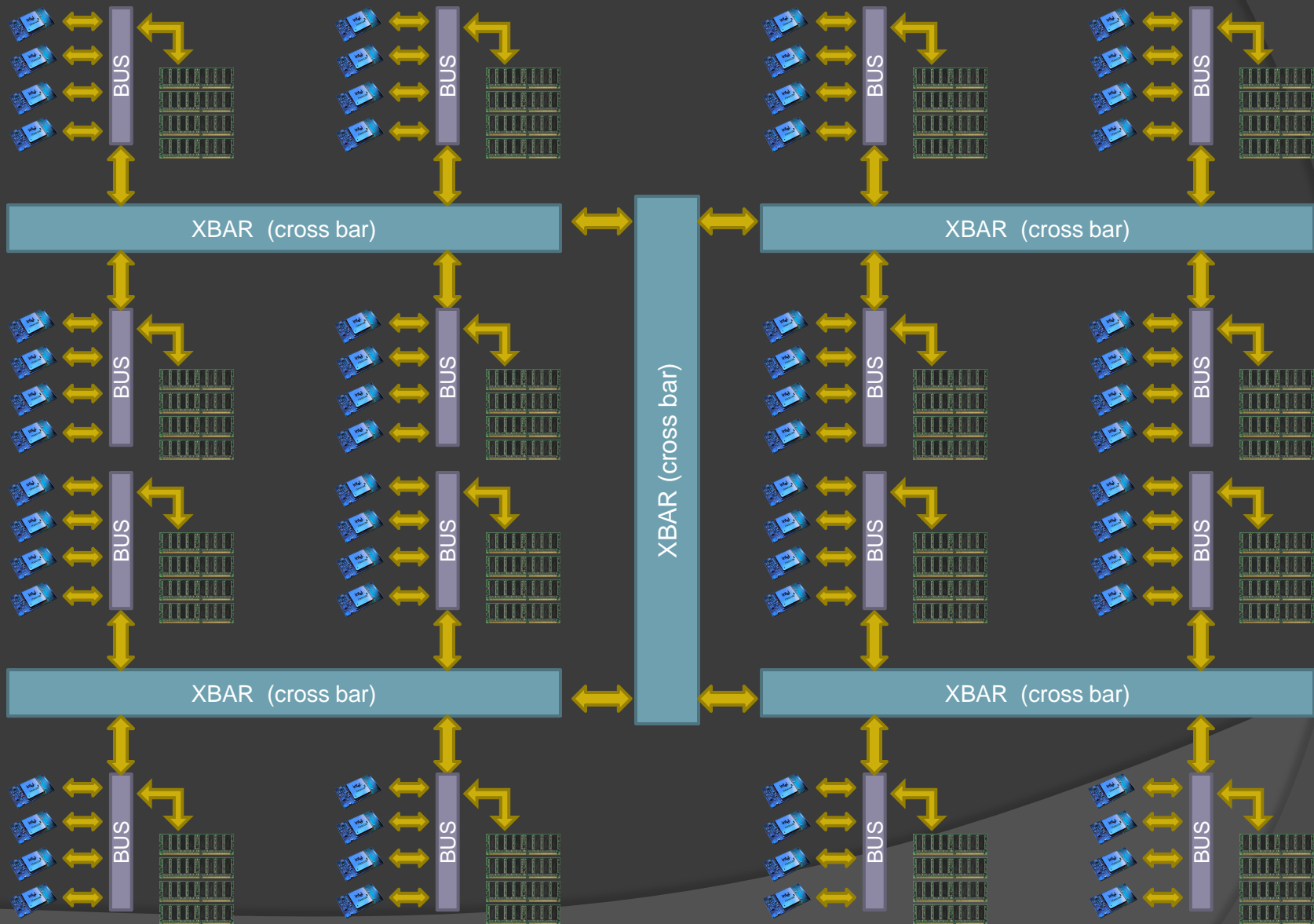
# Quad CPU Solution



# Octal CPU Solution



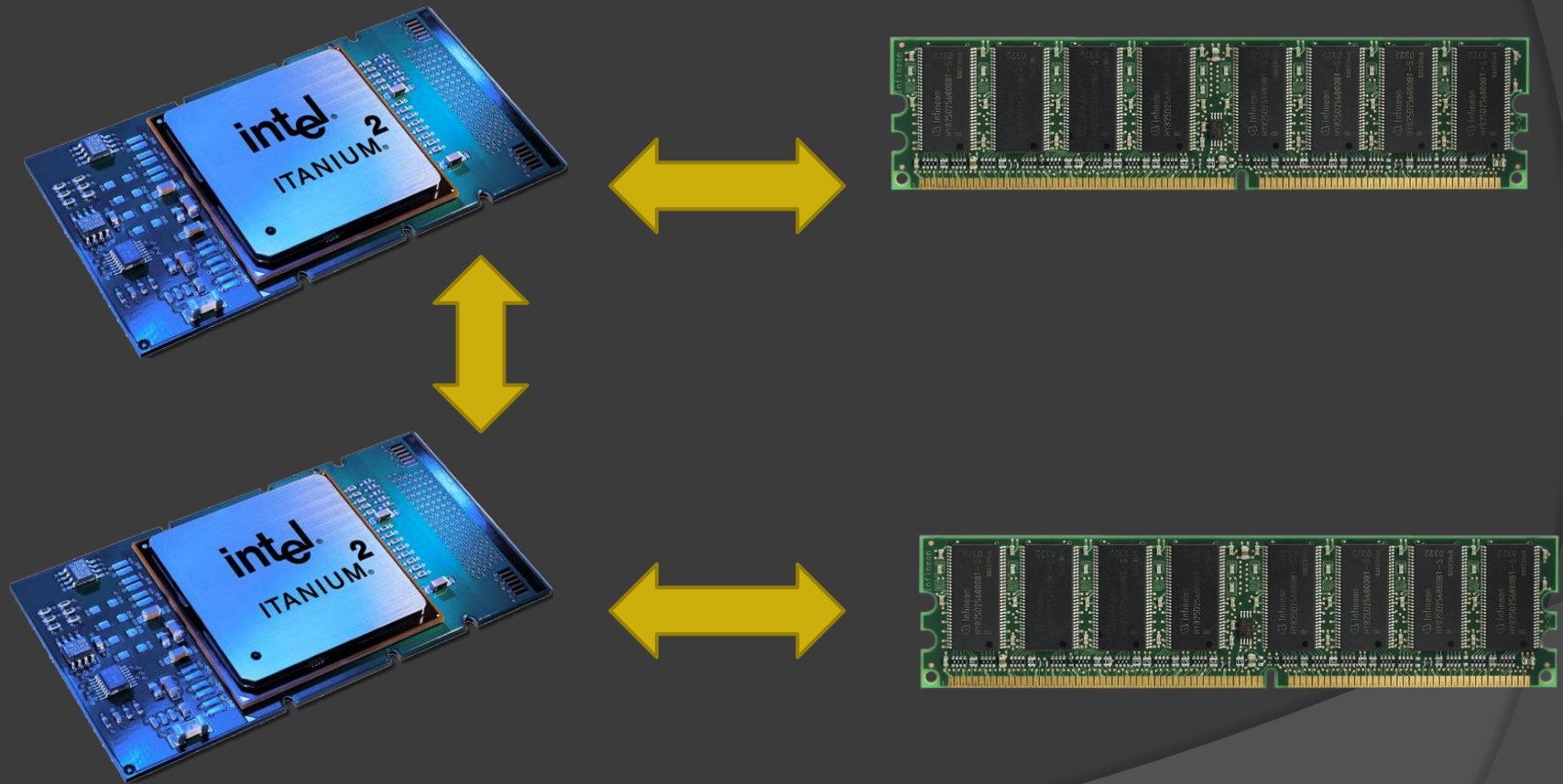
# >8 CPU Solution



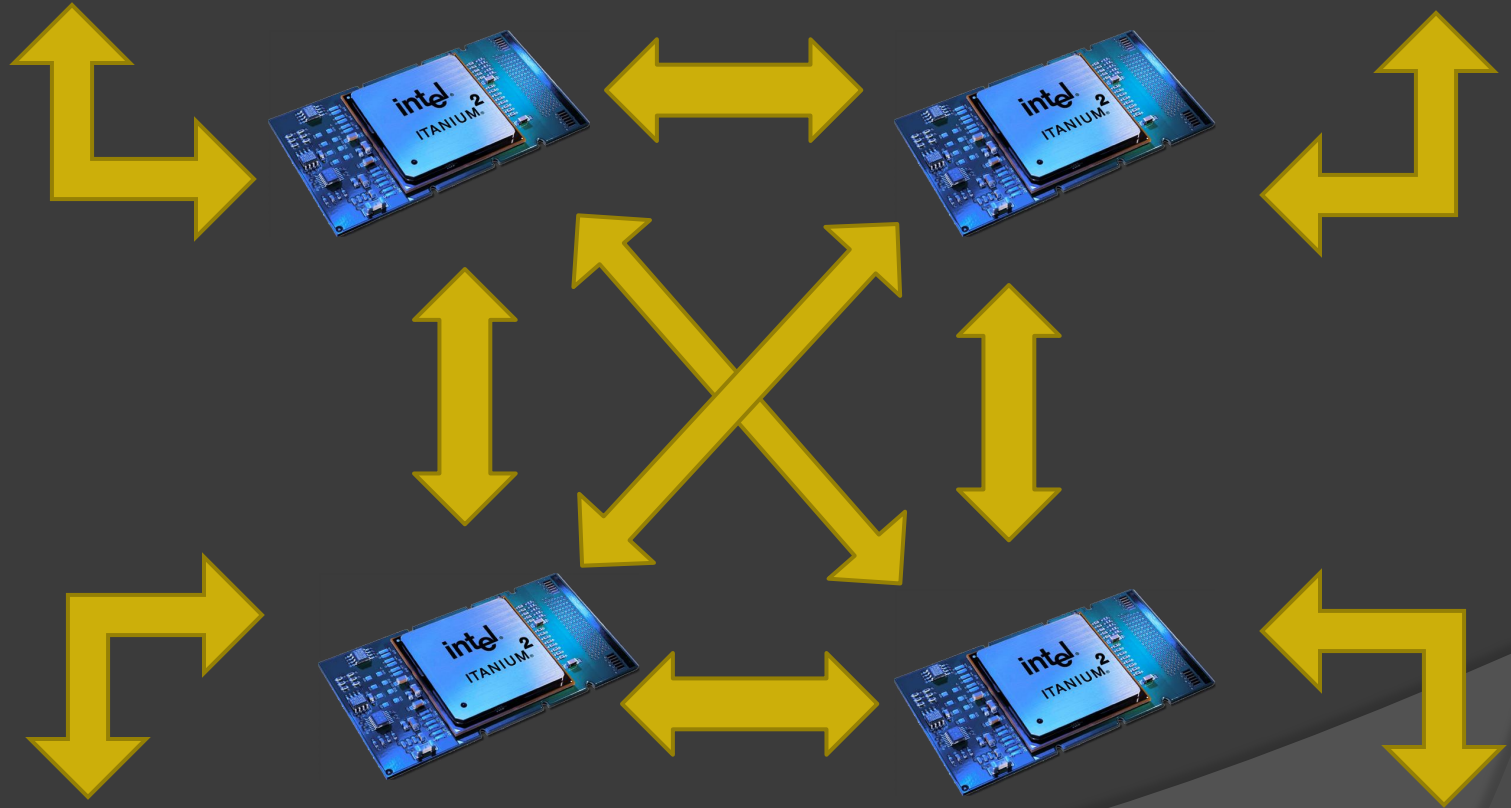
# NUMA

- ⦿ No longer a high end server feature
- ⦿ Almost every  $\geq 2$  CPU server is now a NUMA system.

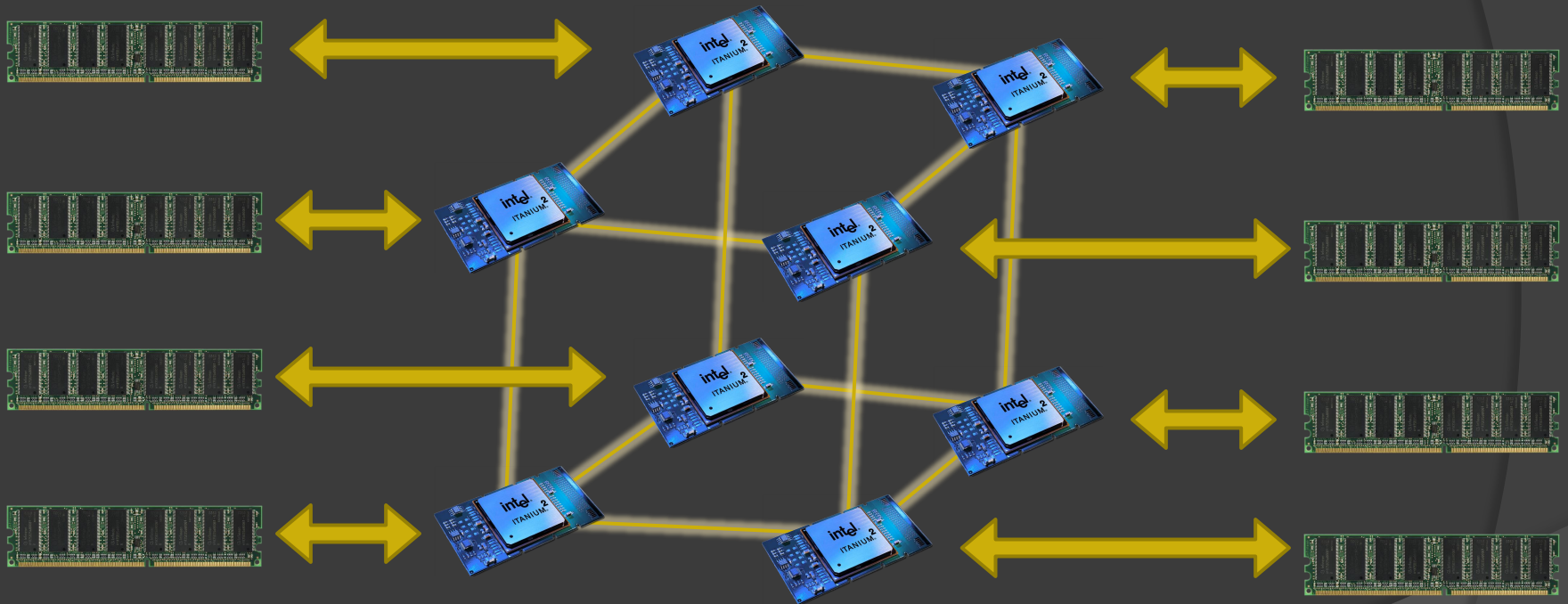
# Dual CPU / Solution



# Quad CPU Solution



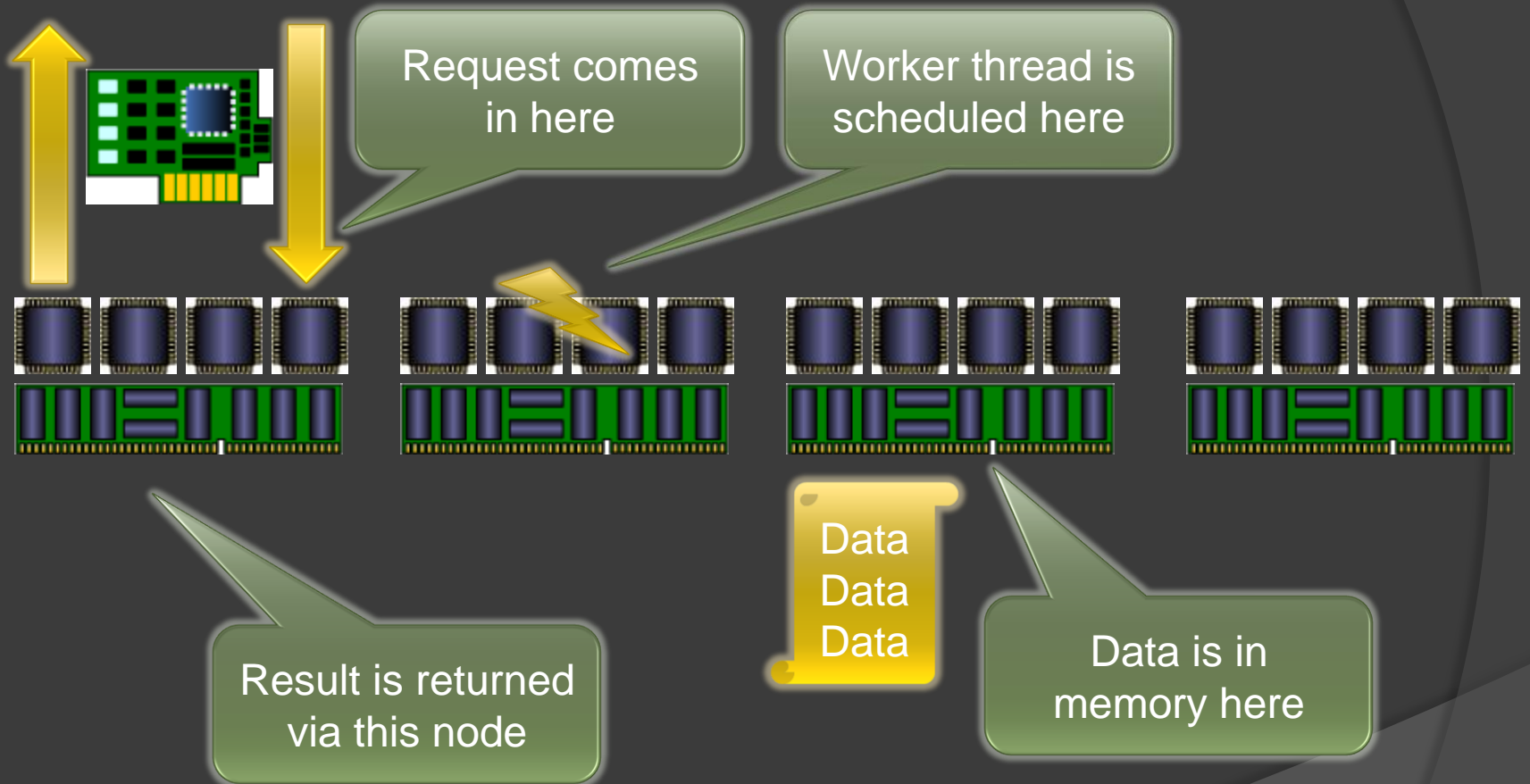
# Octal CPU Solution



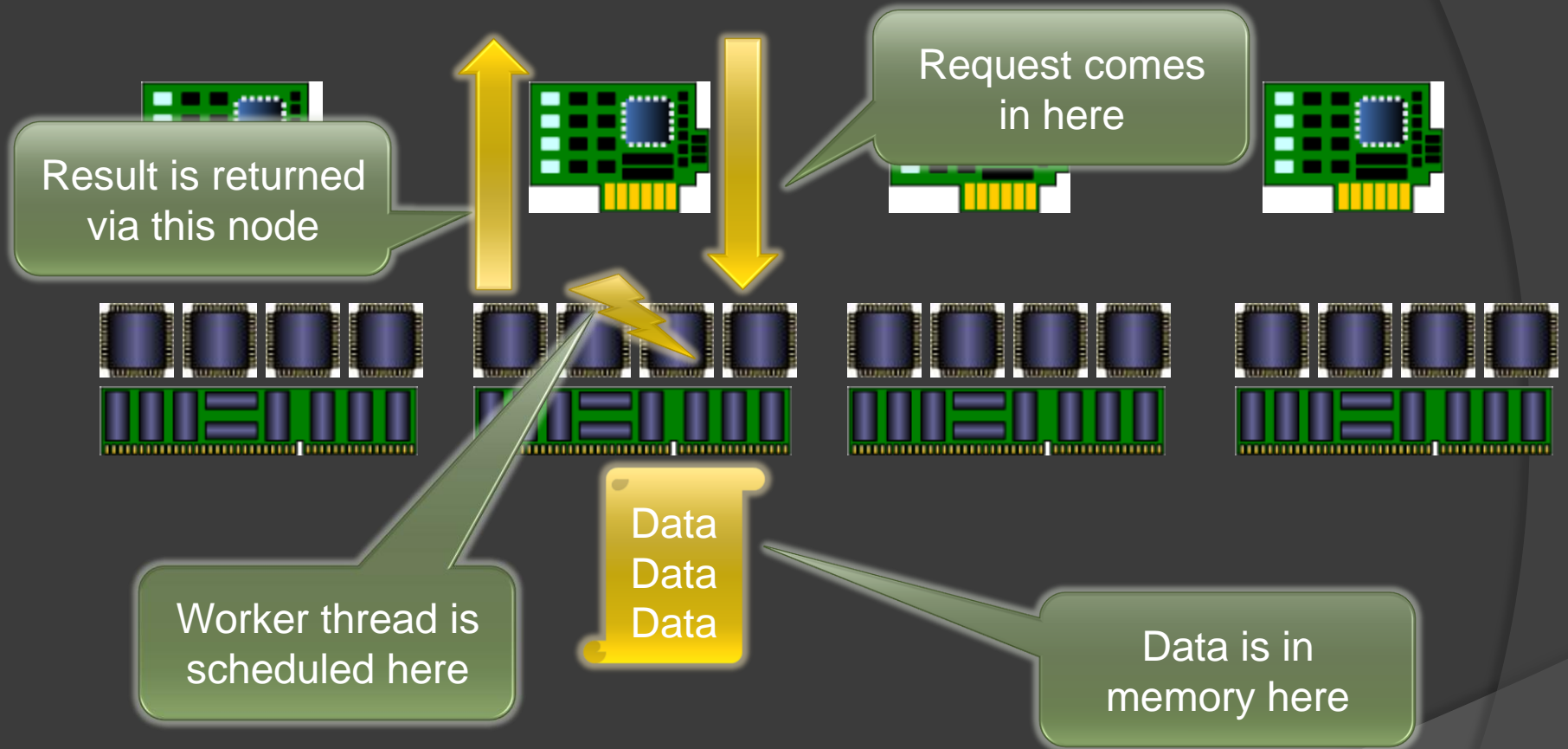
# How To Use it for SQL Server

- ① How SQL Server uses Memory
  - Allocation from the Buffer Pool
  - Allocation from Server Memory
- ① How SQL Server uses Memory on a NUMA System
  - Allocation from the local Buffer Pool
  - Allocation from local NUMA Node memory

# Not NUMA friendly



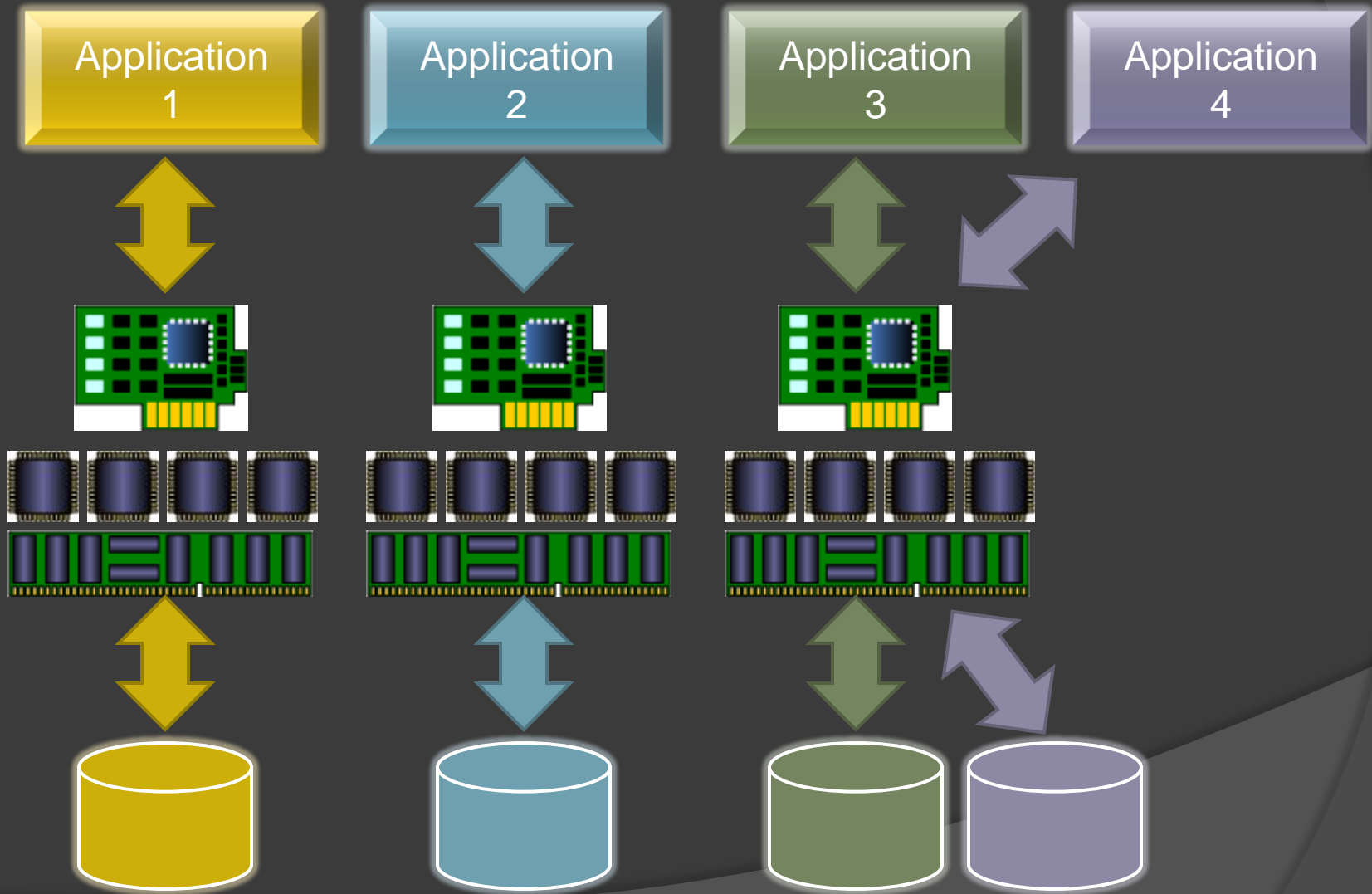
# NUMA Friendly way



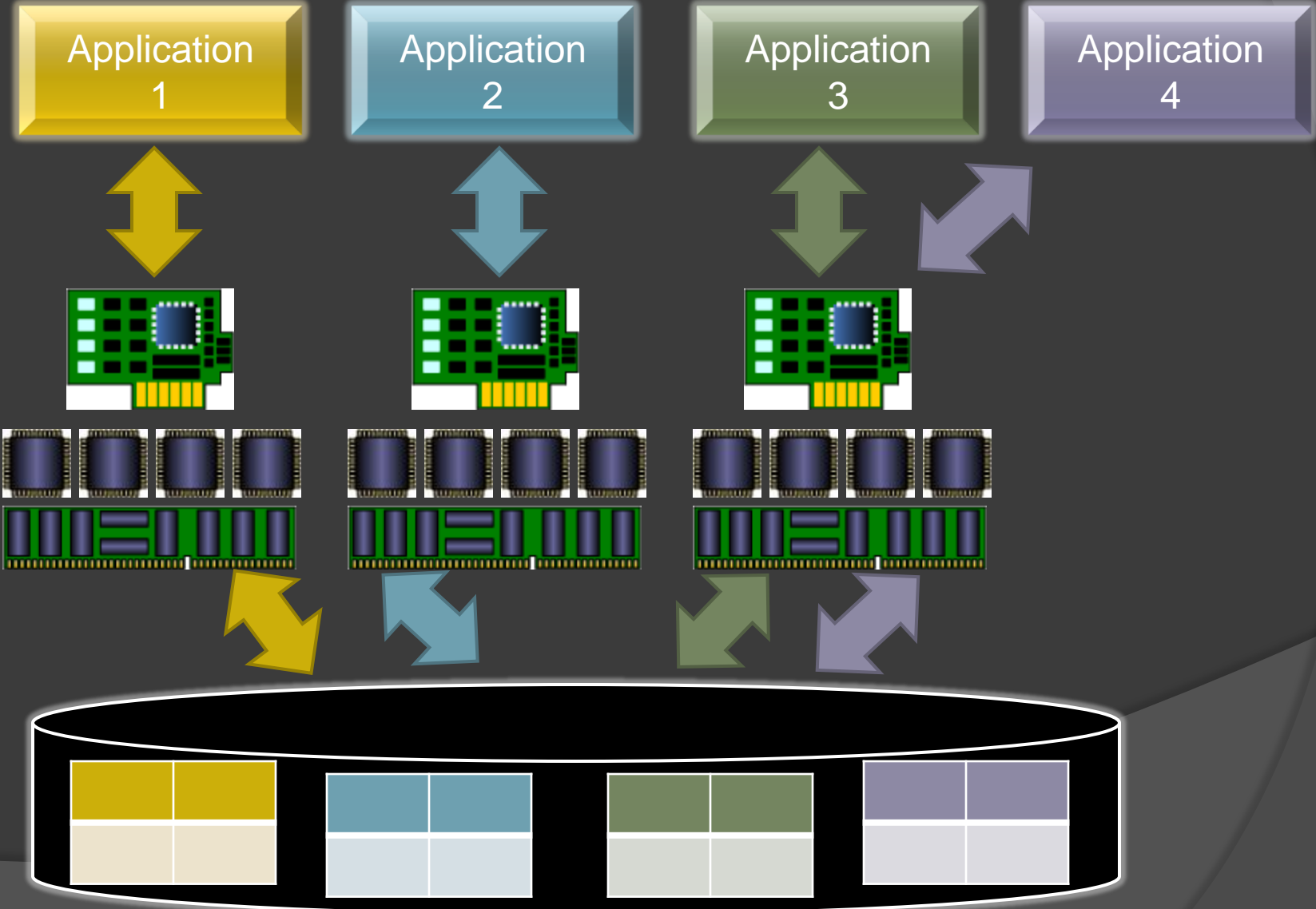
# What is easy to affinitize

- ◎ Fully Transparent to the Application
  - Multiple databases on the same server used by different application
  - Same database but different applications accessing different parts of the database
- ◎ With Application Support
  - Partition the data within one application and process each partition on a separate node

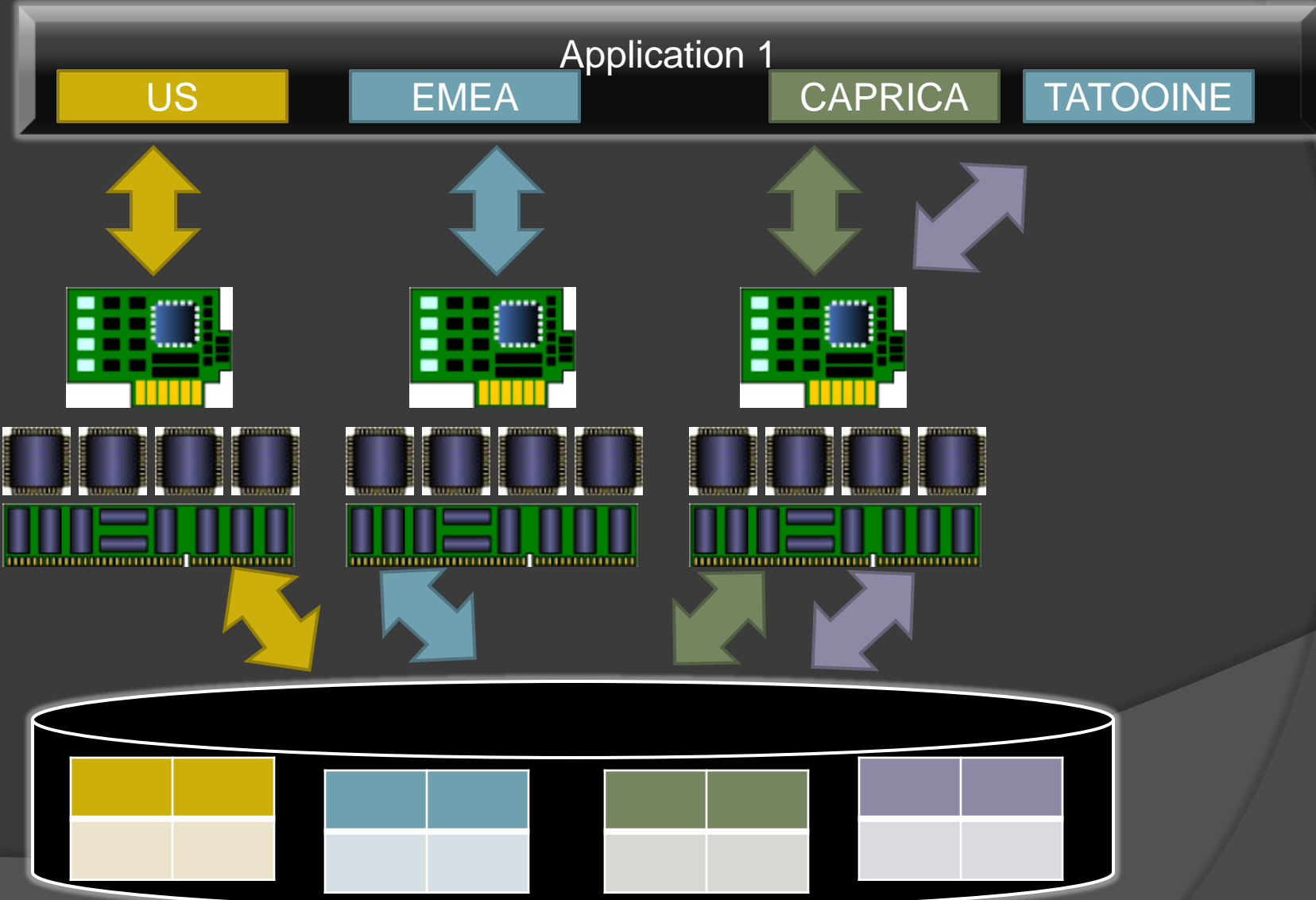
# Multiple Databases and Applications



# Same Database / Multiple Applications

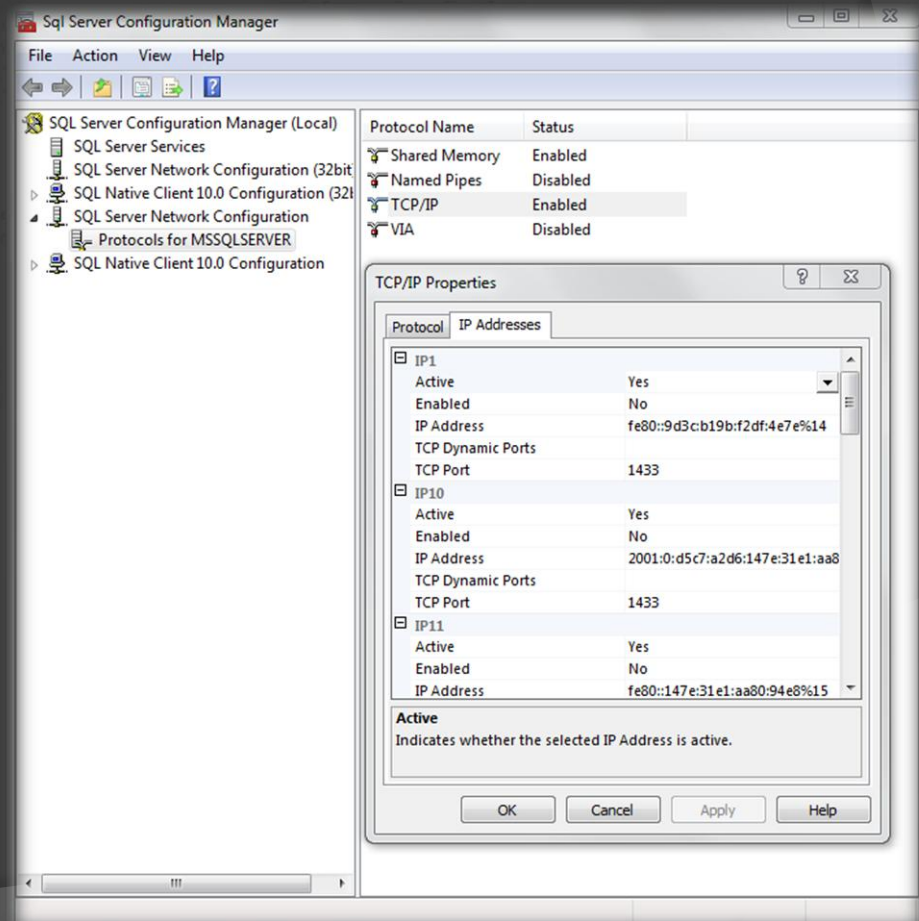


# Same Database and Application



# How to Affinitize

- You can map a NUMA node to an IP Address and Port



# How to Affinitize

⦿ <port number>[<affinity bit mask>]

⦿ Single Port Examples

- 1500[0x1] → NUMA Node 0
- 1600[0x2] → NUMA Node 1
- 1700[0x4] → NUMA Node 2
- 1800[0x7] → NUMA Node 0,1,2

# How to Affinitize

- You can also combine the settings
- 1500[0x1],1501[0x2],1502[0x3],1433[0xf]

1433 uses all Nodes, while 1500 to 1502 use one node each

# How to connect from an Application

```
string generalConnectionString =  
"Server=MyServer;Database=MyDB;...";
```

```
string node1ConnectionString =  
"Server=MyServer,1500;Database=MyDB;..."
```

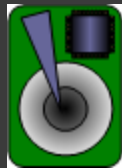
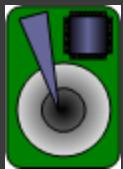
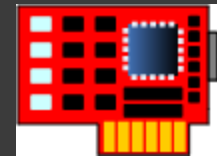
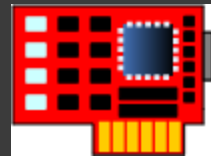
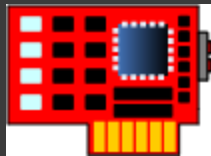
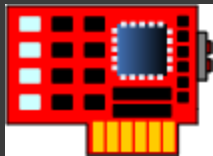
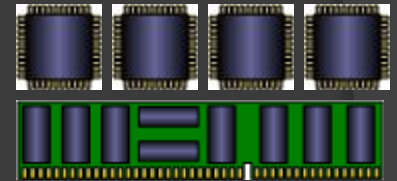
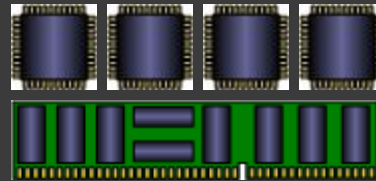
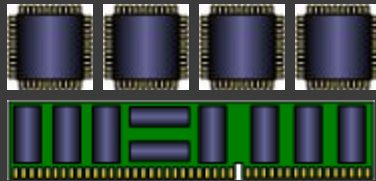
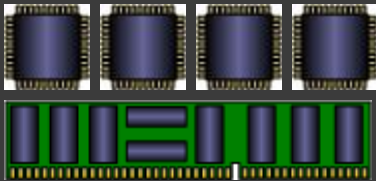
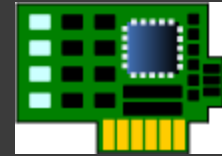
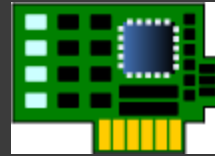
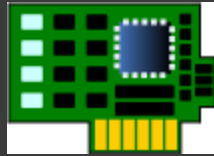
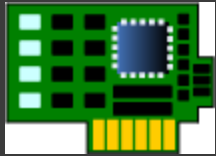
```
string node2ConnectionString =  
"Server=MyServer,1501;Database=MyDB;..."
```

# Good vs Bad Server design

- ⦿ All I/O in one node
- ⦿ Put OS Disks in the OS node
  - Node 0 on most systems, depends on CPU and Server manufacturer (on some system OS is in last node)
- ⦿ Enable NUMA
  - Setting often called Interleaved memory On / Off
  - Tricky setting:
    - On** = NUMA OFF or Disabled
    - Off** = NUMA ON or Enabled (the good stuff)



# Good vs Bad Server design



# SQL Server Error log

- On a NUMA enabled system you will find the following entries in the error log

<date> <time> Server Node configuration:

```
node 0: CPU mask: 0x00000000000000f0 Active CPU mask: 0x00000000000000f0
node 1: CPU mask: 0x000000000000000f Active CPU mask: 0x000000000000000f
node 2: CPU mask: 0x0000000000000f00 Active CPU mask: 0x0000000000000f00
node 3: CPU mask: 0x000000000000f000 Active CPU mask: 0x000000000000f000
node 4: CPU mask: 0x00000000000f0000 Active CPU mask: 0x00000000000f0000
node 5: CPU mask: 0x000000000f000000 Active CPU mask: 0x000000000f000000
node 6: CPU mask: 0x00000000f0000000 Active CPU mask: 0x00000000f0000000
node 7: CPU mask: 0x0000000f00000000 Active CPU mask: 0x0000000f00000000
```

This message provides a description of the NUMA configuration for this computer.  
This is an informational message only. No user action is required.

# Avoiding Context Switches

```
exec sp_configure 'show advanced options', 1  
reconfigure
```

```
exec sp_configure 'affinity mask', 0x0002  
reconfigure
```

HEX Value is CPU mask; One bit per CPU

Bit =1 CPU is used and a Worker Thread will not switch once started

Bit = 0 CPU is not used

All Bits = 0 (default) all CPU's are used but worker threads can switch from CPU to CPU

(this costs time only good on mixed OLTP/DSS/BI systems)

# Dedicating CPU's for I/O

- `exec sp_configure 'affinity I/O mask', 0x0002`
- If a CPU is dedicated to I/O corresponding bits in the affinity mask must be 0

Affinity Mask	0	0	1
Affinity I/O Mask	0	1	0



## >32 CPUs

For the CPUs 33 to 64 there is a second pair of configuration values:

```
exec sp_configure 'affinity64 mask', 0x0000
```

```
exec sp_configure 'affinity64 I/O mask', 0x0000
```

# >64 CPU's: Windows 2008 R2 and SQL Server 2008 R2

- Unisys ES7000 96 Cores Xeon
- HP Superdome 128/256 Cores Itanium
- More CPU's and cores to come in the future

ALTER SERVER CONFIGURATION

SET PROCESS AFFINITY

CPU = { AUTO | <range\_spec> } |  
NUMANODE = <range\_spec>

# >64 CPU's: Windows 2008 R2 and SQL Server 2008 R2

```
ALTER SERVER CONFIGURATION SET PROCESS  
AFFINITY CPU = AUTO
```

```
ALTER SERVER CONFIGURATION SET PROCESS  
AFFINITY CPU = 0
```

```
ALTER SERVER CONFIGURATION SET PROCESS  
AFFINITY CPU = 2 to 8
```

```
ALTER SERVER CONFIGURATION SET PROCESS  
AFFINITY CPU = 2 to 8, 12 to 100, 107
```

```
ALTER SERVER CONFIGURATION SET PROCESS  
AFFINITY NUMANODE = 0, 2 TO 4, 7, 8
```

# Side effect: Save licensing costs

- You have a lot of CPU's its very likely that one or more CPU for I/O might make sense
- Example 32 CPU's and we dedicate one per NUMA node for I/O

```
exec sp_configure 'affinity mask', 0x77777777
```

```
exec sp_configure 'affinity I/O mask', 0x88888888
```

→ Only 24 CPU licenses are required for SQL Server

# Wrap Up

- ① NUMA has arrived or will arrive at your datacenter soon
- ① NUMA makes local memory access faster not remote memory access slower, without NUMA all memory access would be like remote!
- ① Make NUMA your friend by actively supporting it



Thank you!

Questions ?

Email: [tg@grohser.com](mailto:tg@grohser.com)