CERN IT Department

RAC scalability
PVSS example

Eric Grancher
eric.grancher@cern.ch
CERN IT

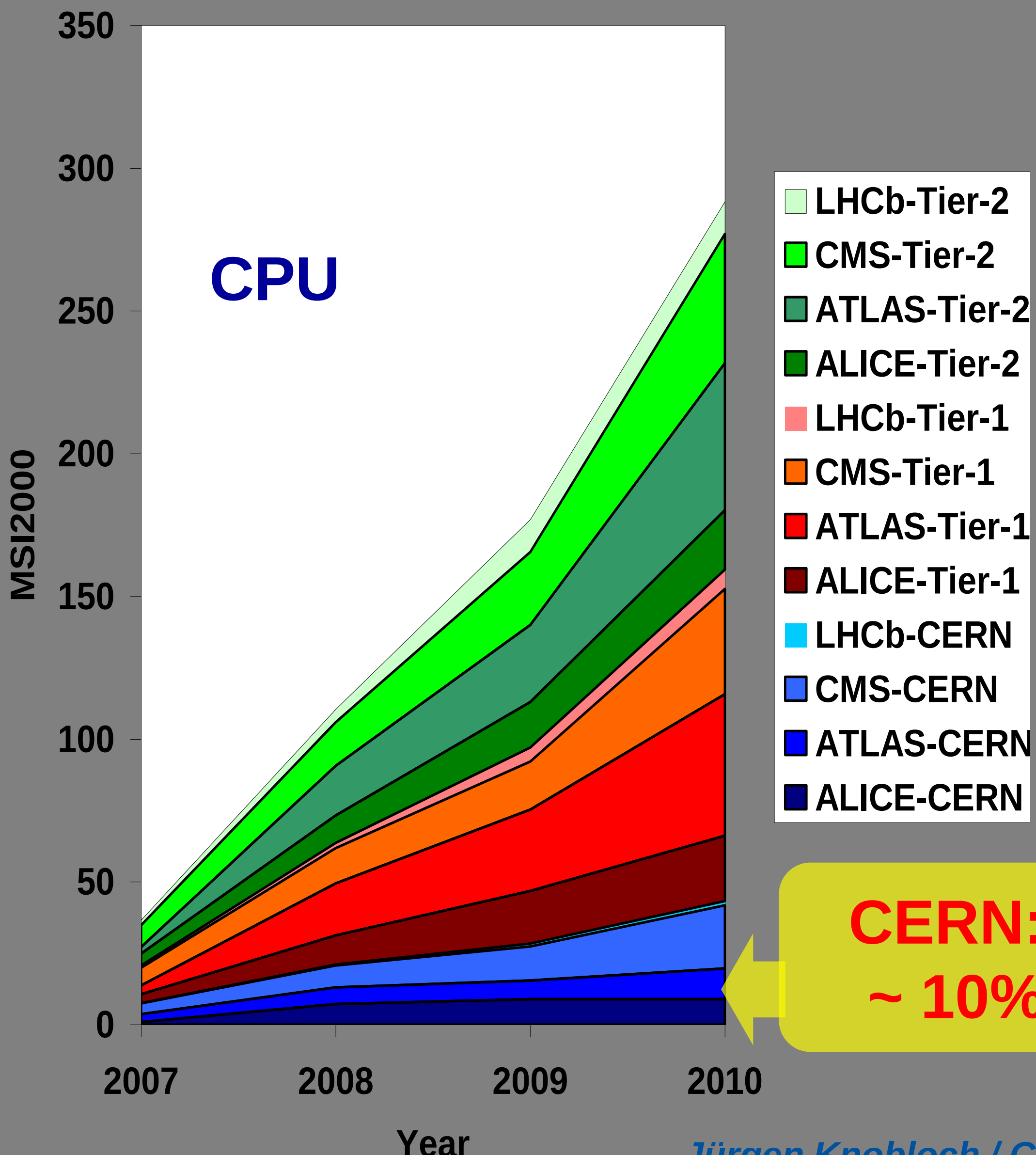
Anton Topurov
anton.topurov@cern.ch
openlab, CERN IT

Outline

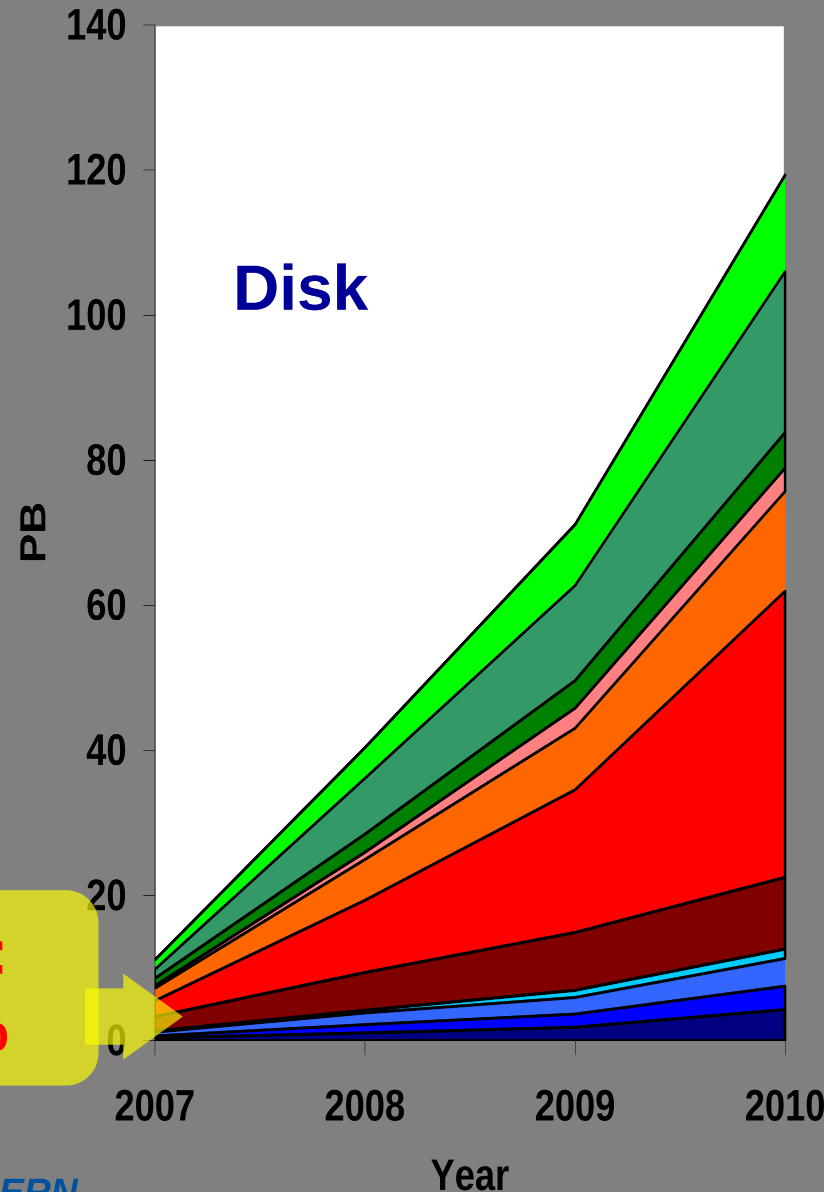
- CERN computing challenge
- Oracle RDBMS and Oracle RAC @ CERN
- RAC scalability – what, why and how?
- Real life scalability example
- Conclusions
- References

CPU & Disk Requirements 2006

CERN IT Department



CERN:
~ 10%



Jürgen Knobloch / CERN

Oracle databases at CERN

- **1982** : CERN starts using Oracle
- **1996**: OPS on Sun SPARC Solaris
- **2000**: Use of Linux x86 for Oracle RDBMS

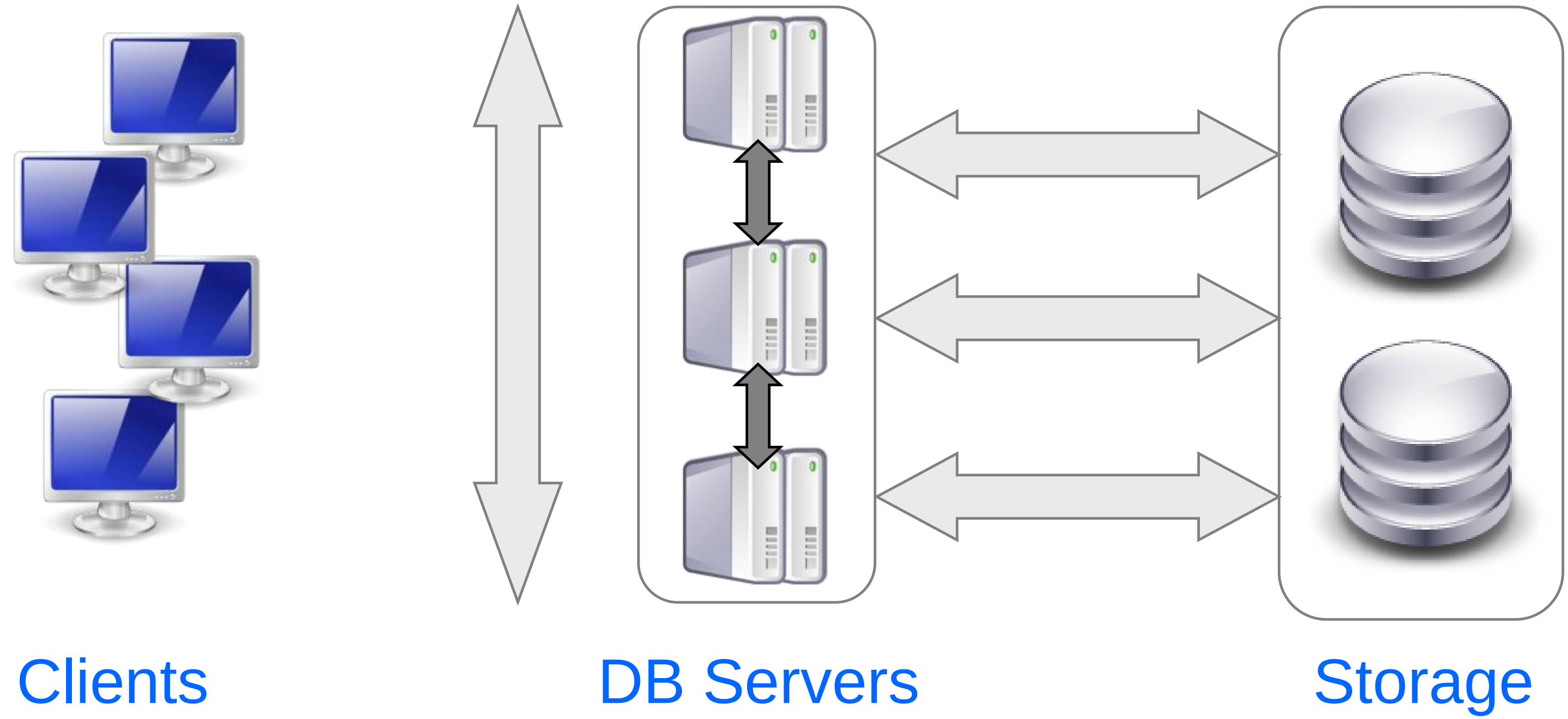
- **Today**:

  Oracle RAC for most demanding services:
  - CASTOR mass storage system (15 PB / year)
  - Administrative applications (AIS)
  - Accelerators and controls etc.
  - LHC Computing Grid (LCG)
  - Experiment databases

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

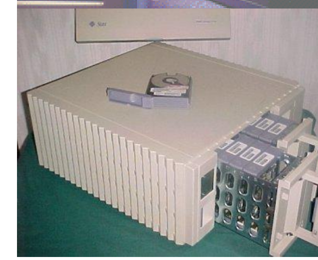
RAC basics

- Shared disk infrastructure, all disk devices have to be accessible from all servers
- Shared buffer cache (with coherency!)



RAC history at CERN (1/3)

- 1996 for consolidation,
  Solaris 2.5.1, Sun PDB 1.2,
  Oracle7 OPS,
  2 SPARCcenter 2000 (10 CPUs 40MHz)
  et 2 SPARCStorage FC (30 disks 1GB)
  « Merge » of two databases
  over a week-end
  HA capabilities were
  (already) working



- 1998, rejuvenation,
  2 nodes E450,
  Solaris 2.6, SunCluster 2.2
  Storedge D1000

C

RAC history at CERN (2/3)

CERN IT Department

- 2003, rejuvenation,
  3 nodes Sun Sparc V880 16GB 8CPU,
  Storedge T3 then 3510FC,
  Solaris 2.8, SunCluster3
  All 1Gbit/s Ethernet

- From Oracle7OPS to Oracle9iR2 RAC(9208).
- Impressive software stability
  (almost everything broke at some point:
  disks, power, CPUs, controllers, interconnect… even mother board!).
- But human errors / configuration problems lead to some downtime. Complexity. A lot of layers.

RAC history at CERN (3/3)

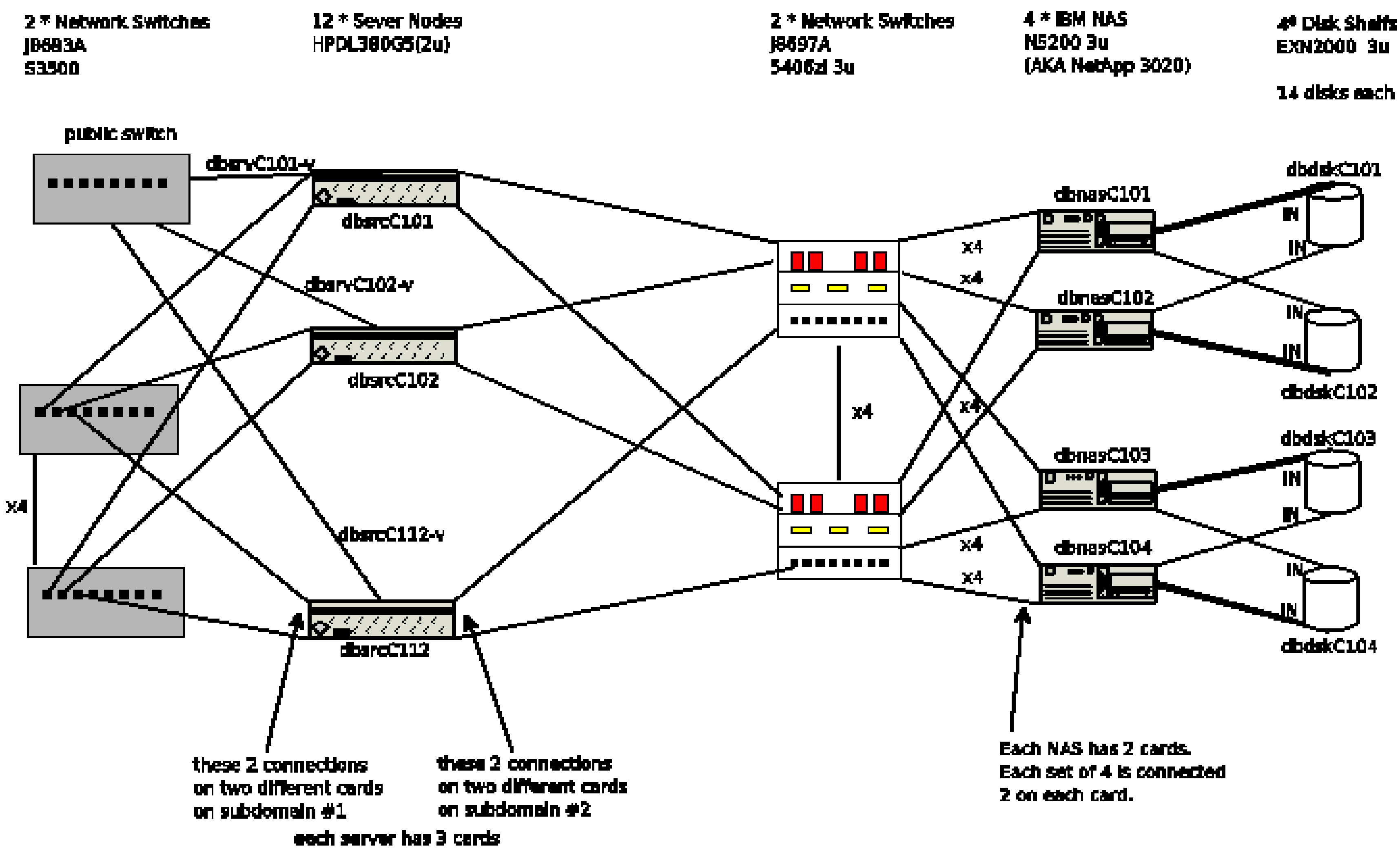
- We started in 2001 to investigate Linux and RAC.
- A lot of things have changed now:
  - ASM for storage / Network Attached Storage (and cluster filesystem).
  - Much simpler and more efficient usage of resources (example: GC_FILES_TO_LOCKS).
  - 10g database services, Virtual IPs

- Now global migration from Sparc Solaris to Linux x86/x86_64

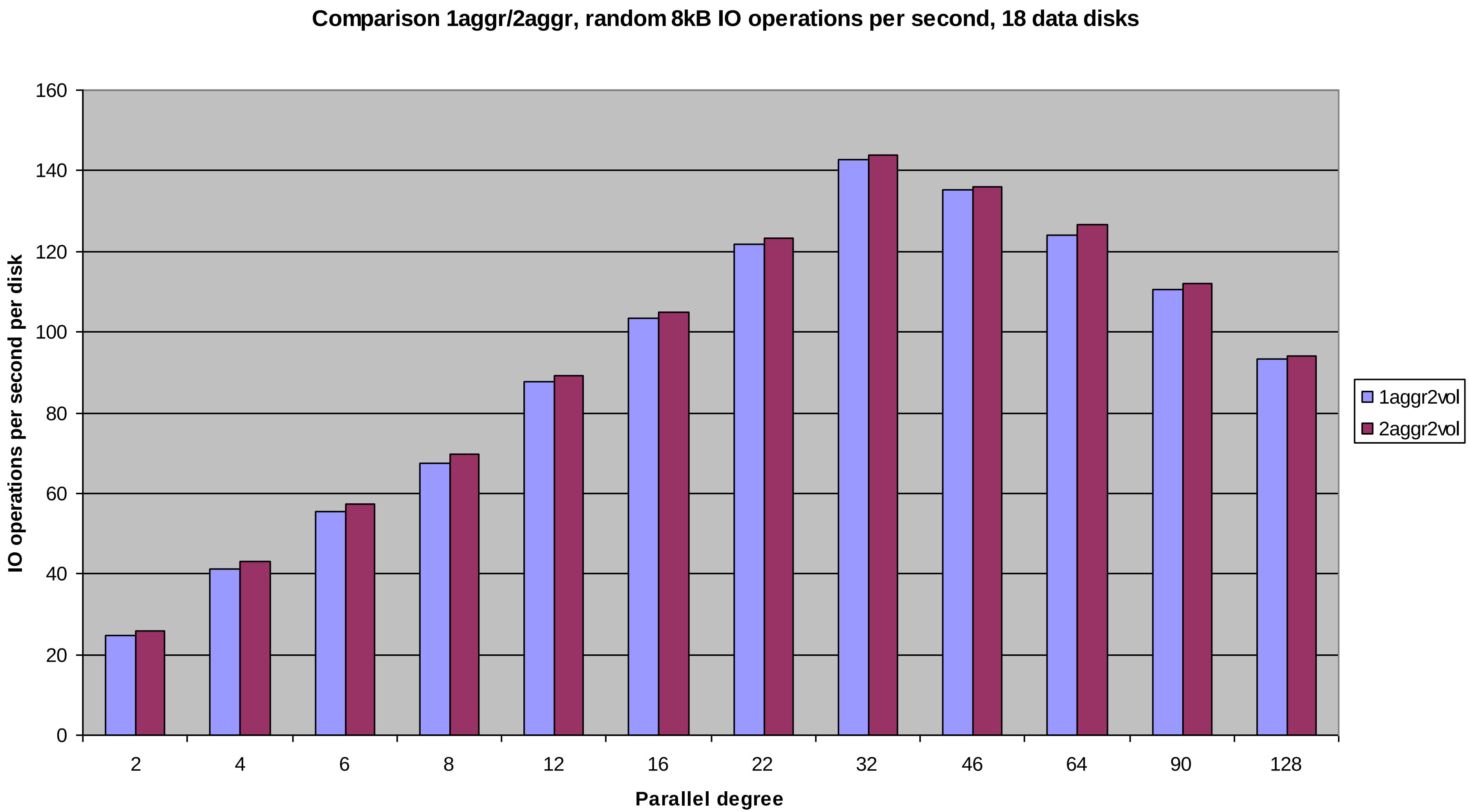
Platform

- Multi-core servers (Woodcrest, Harpertown to be installed very soon)
- NAS with GbitE trunking (12x GbitE for storage)
- Private network for storage, private network for interconnect
- Networking handled by the networking team

NAS Based Oracle DB Infrastructure for CastorV2.

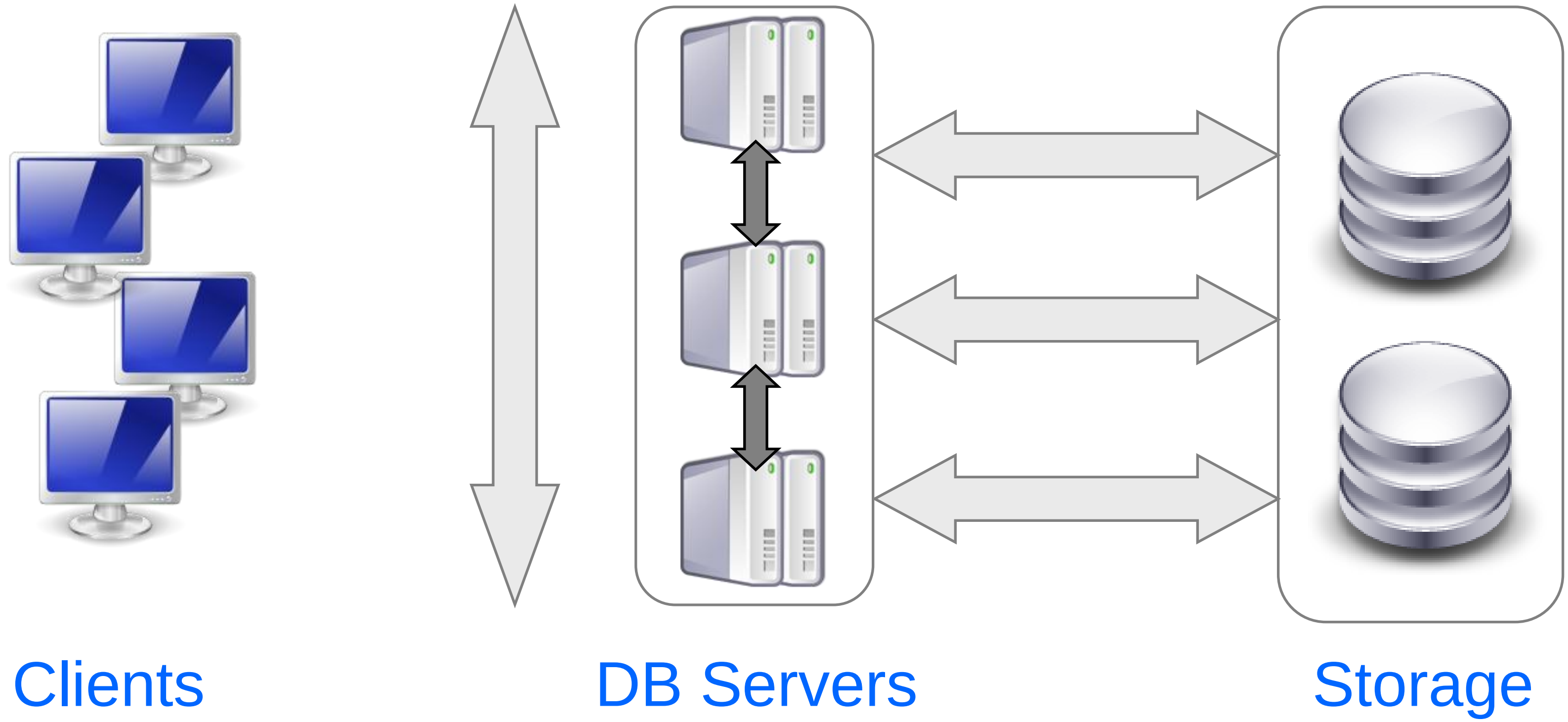


NAS performance



Real Application Cluster

- **Shared disk** infrastructure, all disk devices have to be accessible from all servers
- **Shared buffer** cache (with coherency!)



RAC messaging

PVSS II

- Commercial SCADA application
  - critical for LHC and experiments
- Archiving in Oracle Database
- Out of the box performance:

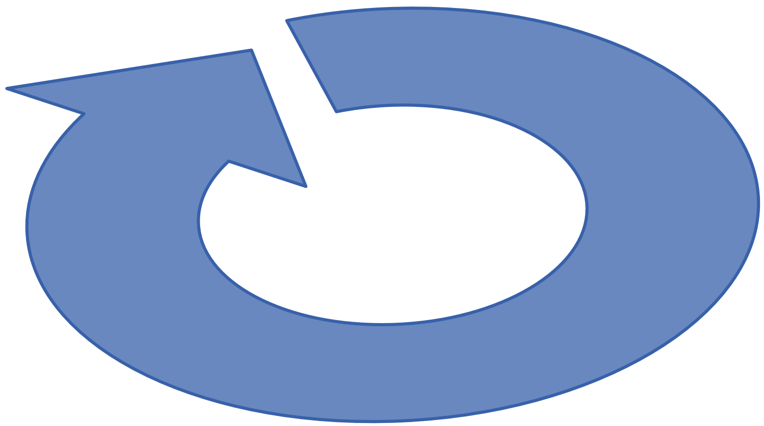
  100 "changes" per second
- CERN needs:

  150'000 changes per second = **1'500 times** faster!

The Tuning Process

1. run the workload, gather ASH/AWR information, 10046…

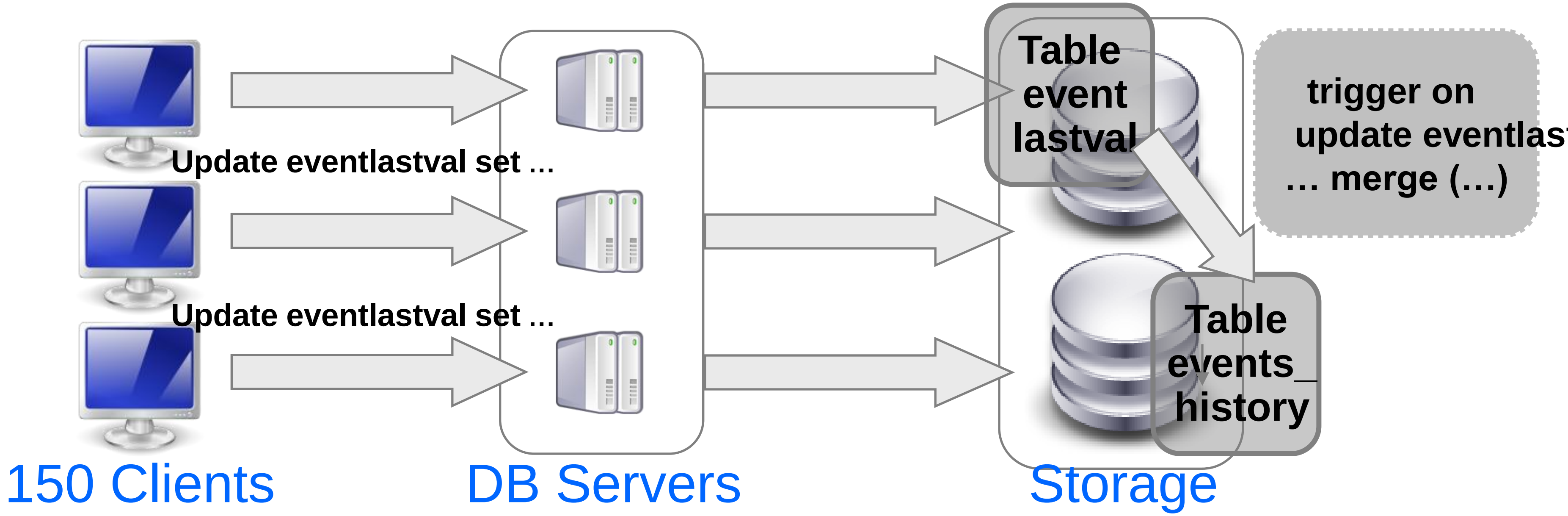
2. find the top event that slows down the processing



3. understand why time is spent on this event

4. modify client code, database schema, database code, hardware configuration

PVSS Tuning (1/10)



- **Shared resource**:
  EVENTS_HISTORY (ELEMENT_ID, VALUE…)
- Each client "measures" input and registers history with a "merge" operation in the EVENTS_HISTORY table

Performance:

- **100** "changes" per second

PVSS Tuning (2/10)

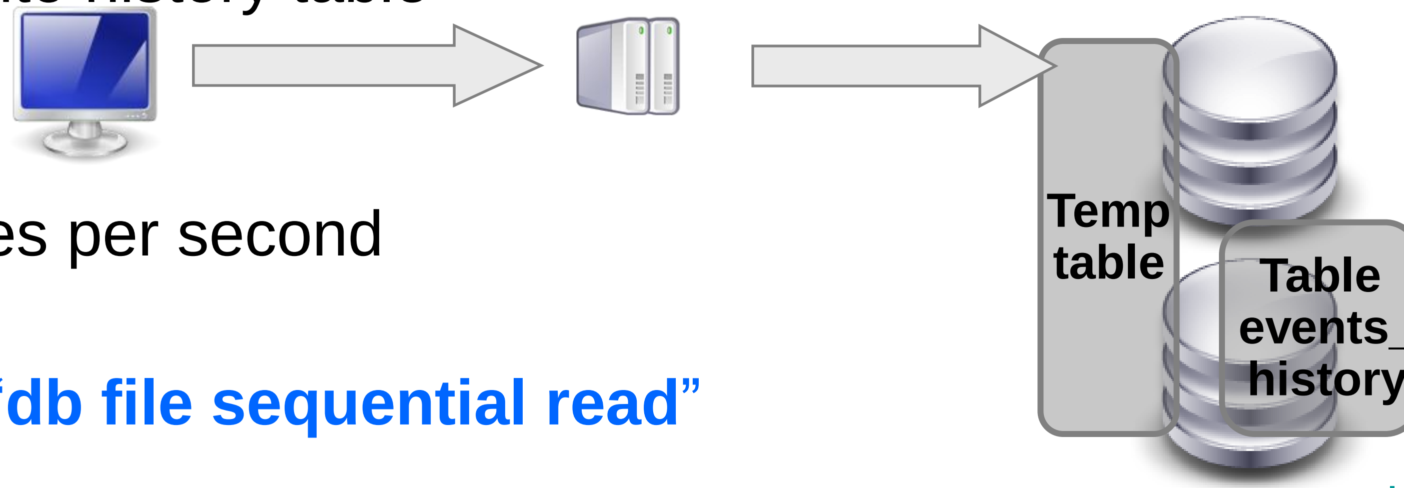
Initial state observation:

- database is waiting on the clients
  - “**SQL*Net message from client**”
- Use of a generic library C++/DB
- Individual insert (one statement per entry)
- Update of a table which keeps “latest state” through a trigger

PVSS Tuning (3/10)

Changes:

- bulk insert to a temporary table with OCCI, then call PL/SQL to load data into history table



Performance:

- **2'000** changes per second

Now top event: **"db file sequential read"**

awrrpt_1_5489_5490.html

Event	Waits	Time(s)	Percent Total DB Time	Wait Class
db file sequential read	29,242	137	42.56	User I/O
enq: TX - contention	41	120	37.22	Other
CPU time		61	18.88	
log file parallel write	1,133	19	5.81	System I/O
db file parallel write	3,951	12	3.73	System I/O

PVSS Tuning (4/10)

Changes:

- Index usage analysis and reduction
- Table structure changes. IOT.
- Replacement of merge by insert.
- Use of “direct path load” with ETL (handle errors efficiently)

- `DBMS_ERRLOG.CREATE_ERROR_LOG ( dml_table_name IN VARCHAR2, err_log_table_name IN VARCHAR2 := NULL, err_log_table_owner IN VARCHAR2 := NULL, err_log_table_space IN VARCHAR2 := NULL, skip_unsupported IN BOOLEAN := FALSE);`
- `LOG ERRORS [INTO [schema.]table] [ (simple_expression) ] [ REJECT LIMIT {integer|UNLIMITED} ]`

PVSS Tuning (5/10)

CERN IT Department

Performance:

- **16’000** “changes” per second
- Now top event: **cluster related wait event**

test5_rac_node1_8709_8710.html

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
gc buffer busy	27,883	728	26	31.6	Cluster
CPU time		369		16.0	
gc current block busy	6,818	255	37	11.1	Cluster
gc current grant busy	24,370	228	9	9.9	Cluster
gc current block 2-way	118,454	198	2	8.6	Cluster

PVSS Tuning (6/10)

CERN IT Department

Changes:

- Each “client” receives a unique number.
- Partitioned table.
- Use of “direct path load” to the partition, requires to specify the partition to avoid table lock
- LMODE = 3 (row-x SX), 6 (exclusive X) --- TYPE = TM (DML enqueue)

```
insert /*+ APPEND */ into ALERTHISTORYVALUES_00000001 select * from ALERTHISTORYVALUES_temp;
select l.type,l.id1,l.id2,l.lmode,l.request,o.object_name,o.subobject_name,o.object_type
from v$lock l,dba_objects o
where l.sid = ( select sid from v$mystat where rownum=1) and o.object_id=l.id1;
TYPE        ID1        ID2      LMODE    REQUEST OBJECT_NAME                    SUBOBJECT_NAME  OBJECT_TYPE
---- ---------- ---------- ---------- ---------- ------------------------------ --------------- ---------------
TM        86462          0          6          0 ALERTHISTORYVALUES_00000001                    TABLE
TM        86466          0          3          0 ALERTHISTORYVALUES_TEMP                        TABLE
TO        86466          1          3          0 ALERTHISTORYVALUES_TEMP                        TABLE


insert /*+ APPEND */ into ALERTHISTORYVALUES_00000001 partition ("DUMMY") select * from ALERTHISTORYVALUES_temp;
select l.type,l.id1,l.id2,l.lmode,l.request,o.object_name,o.subobject_name,o.object_type
from v$lock l,dba_objects o
where l.sid = ( select sid from v$mystat where rownum=1) and o.object_id=l.id1;
TYPE        ID1        ID2      LMODE    REQUEST OBJECT_NAME                    SUBOBJECT_NAME  OBJECT_TYPE
---- ---------- ---------- ---------- ---------- ------------------------------ --------------- ---------------
TM        86468          0          6          0 ALERTHISTORYVALUES_00000001    DUMMY           TABLE PARTITION
TM        86467          0          3          0 ALERTHISTORYVALUES_00000001                    TABLE
TM        86471          0          3          0 ALERTHISTORYVALUES_TEMP                        TABLE
TO        86471          1          3          0 ALERTHISTORYVALUES_TEMP                        TABLE
```

PVSS Tuning (7/10)

Performance:

- **150’000** changes per second
- Now top event : “**freezes**” once upon a while

rate75000_awrrpt_2_872_873.html

Event	**Waits**	**Time(s)**	**Avg Wait(ms)**	**% Total Call Time**	**Wait Class**
row cache lock	813	665	818	27.6	Concurrency
gc current multi block request	7,218	155	22	6.4	Cluster
CPU time		123		5.1	
log file parallel write	1,542	109	71	4.5	System I/O
undo segment extension	785,439	88	0	3.6	Configuration

PVSS Tuning (8/10)

Problem investigation:

- Link between foreground process and ASM processes
- Difficult to interpret, use of ASH report, 10046 trace

Problem identification:

- ASM space allocation is blocking some operations

Changes:

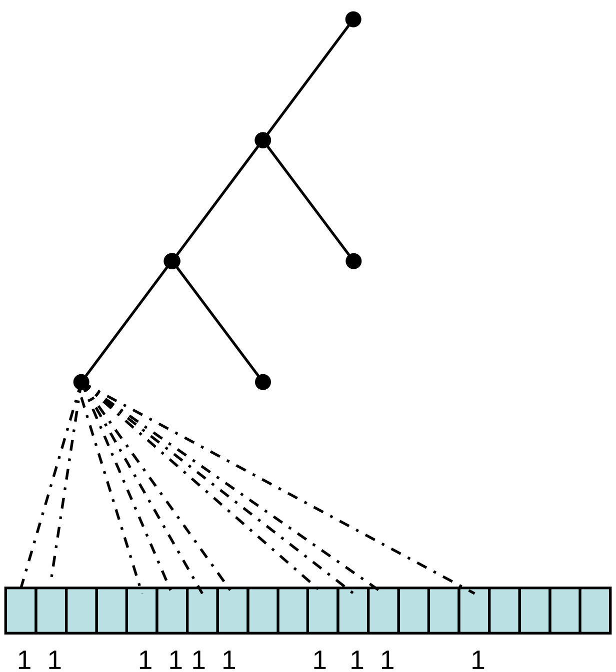
- Space pre-allocation, background task.

Result:

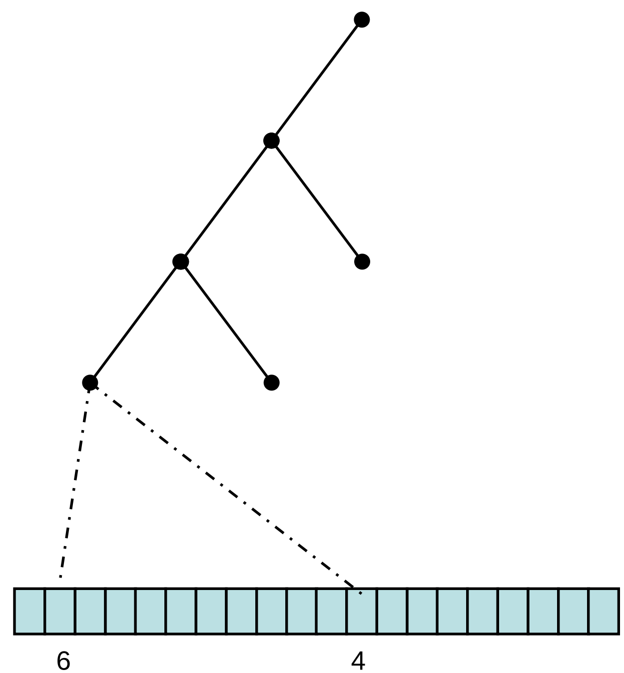
- Stable 150’000 “changes” per second.

PVSS Tuning (9/10)

- Typical queries are long
- “Index clustering factor” is not good…



1 CR/DiskRead in index ->
10 CR/DiskRead for data



1 CR/DiskRead in index ->
2 CR/DiskRead for data

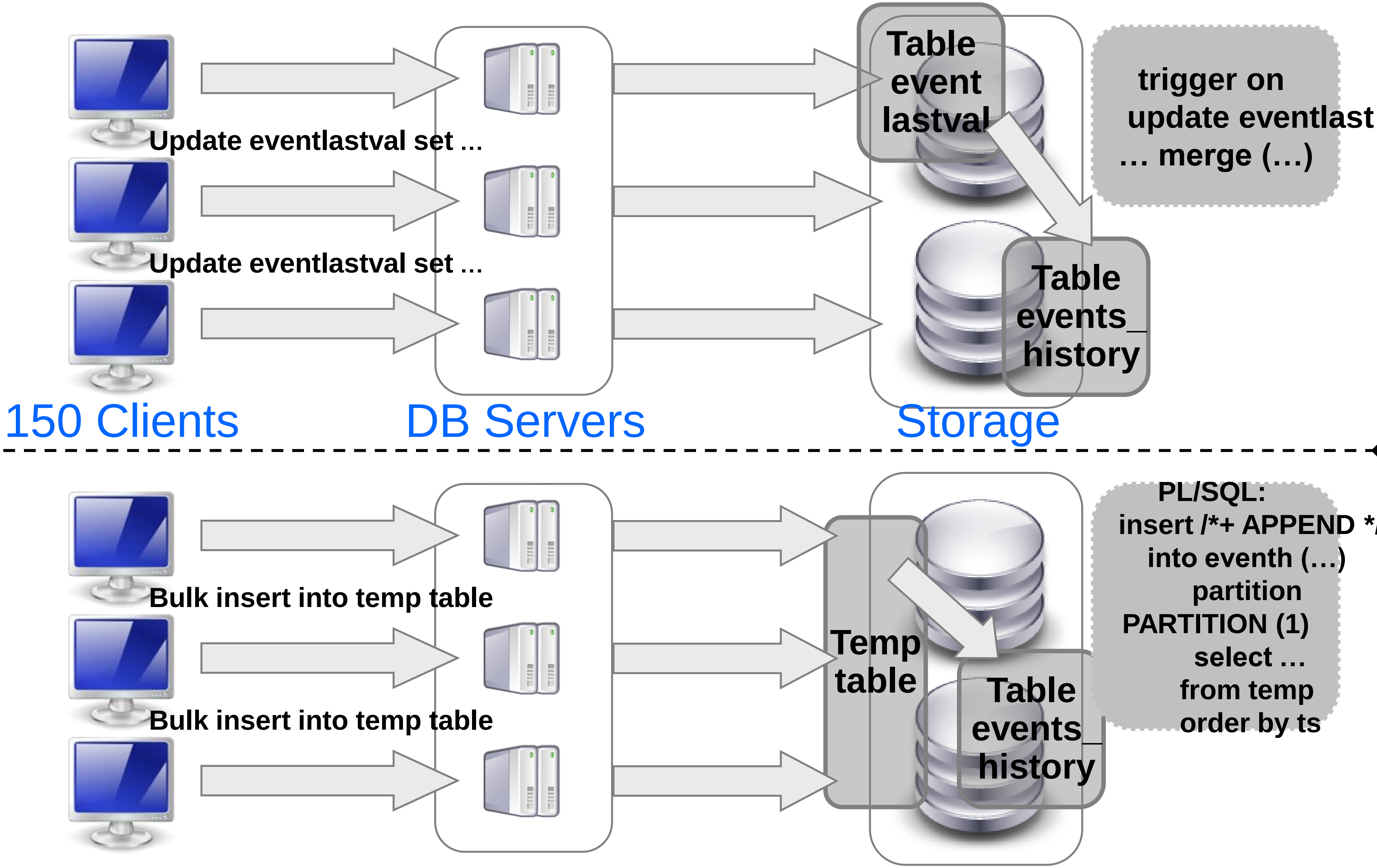
PVSS tuning (10/10)

- `insert /*+ APPEND */ into ALERTHISTORYVALUES_00000001 select * from ALERTHISTORYVALUES_temp`

- `insert /*+ APPEND */ into ALERTHISTORYVALUES_00000001 select * from ALERTHISTORYVALUES_temp` **`order by ts`**

- CR and DiskReads divided by 4

PVSS Tuning Schema



PVSS Tuning Summary

CERN IT Department

Conclusion:

- from **100** changes per second to **150’000** “changes” per second (9’000’000 per minute), simple transactions
- **6 nodes RAC** (dual CPU, 4GB RAM), 32 disks SATA with FCP link to host
- **4 months effort:**
  - Re-writing of part of the application with changes interface (C++ code)
  - Changes of the database code (PL/SQL)
  - Schema change
  - Numerous work sessions, joint work with other CERN IT groups

Scalability Conclusions

CERN IT Department

- RAC is a stable technology that can scale write intensive applications
- ASM / cluster filesystem / NAS allow
  - a much easier deployment
  - way less complexity and human error risk.
- 10g/11g RAC is much easier to work with and tune
- RAC can boost your application performance, but also disclose the weak design points and magnify their impact
- **Proper application design is the key** to almost linear scalability for a “non-read-only” application

Performance needs

- IO operations per second
- Low latency
- Simplicity
- And cheap

- Looking forward to SSD, 10GbitE

Q&A