HiPAS
High Performance Adaptive Schema Migration with Minimum Downtime Option

pasolfora GmbH

An der Leiten 37
D-91177 Thalmässing

Web: www.pasolfora.com

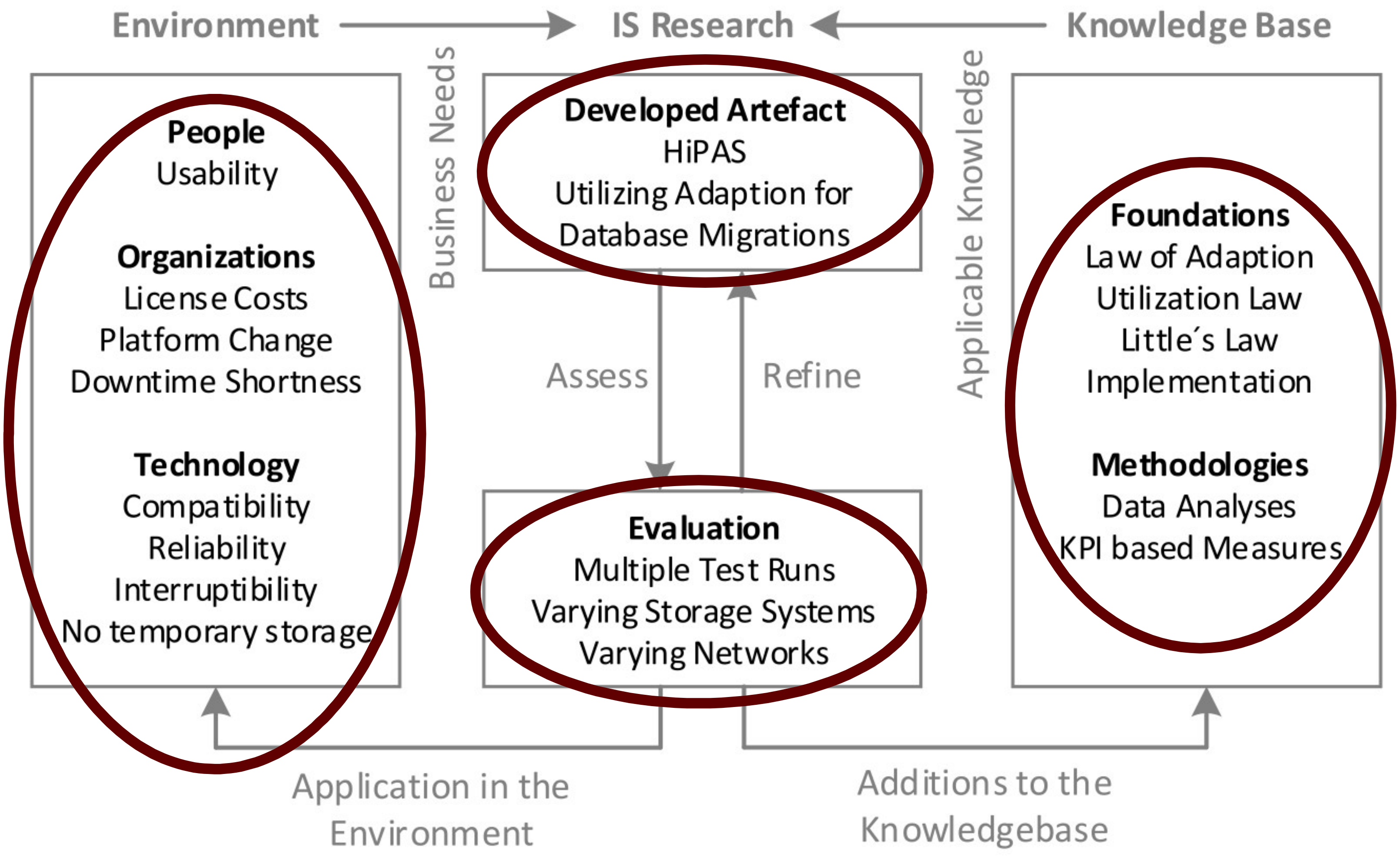
Andreas Prusch
Steffan Agel
Andreas.Prusch@pasolfora.com
Steffan.Agel@pasolfora.com

26/02/2016

Background

- Minimum Downtime Schema Migration and Continous Replication
  - needed very often
  - business and data critical
  - high demand of intensive planning
- Implemented completely in PL /SQL
  - adding up the best practices from Data Pump, O2O, Golden Gate
  - only one PL/SQL package on source and destination
- Academic approach
  - Self Adaptive (artificial intelligence)
  - Developed together with the University of Potsdam/Berlin

Agenda



Environment → IS Research ← Knowledge Base

Business Needs

Applicable Knowledge

People
Usability

Organizations
License Costs
Platform Change
Downtime Shortness

Technology
Compatibility
Reliability
Interruptibility
No temporary storage

Developed Artefact
HiPAS
Utilizing Adaption for
Database Migrations

Assess

Refine

Evaluation
Multiple Test Runs
Varying Storage Systems
Varying Networks

Foundations
Law of Adaption
Utilization Law
Little´s Law
Implementation

Methodologies
Data Analyses
KPI based Measures

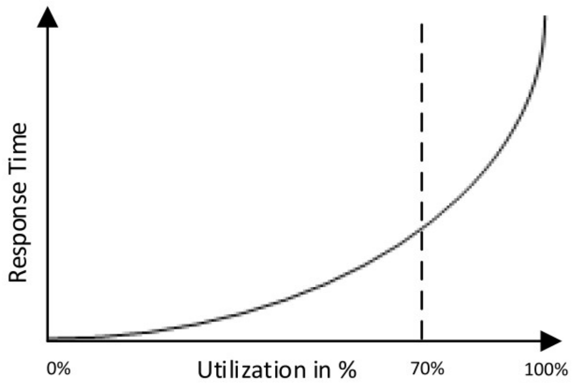
Application in the Environment

Additions to the Knowledgebase

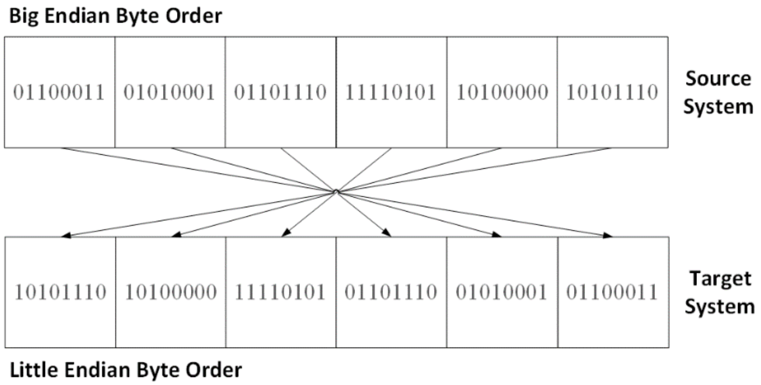
Adapted from Information Systems Research Framework [1]

Migration Challenges

- Short Downtime
  - expensive unavailability due to opportunity costs
- Storage I/O Controller Utilization
  - average utilization of 70% as optimal [2]
  - table diversity (empty, small, very large), up to 70,000 tables
- Endianness
  - byte order changes, e.g., from Solaris to Linux



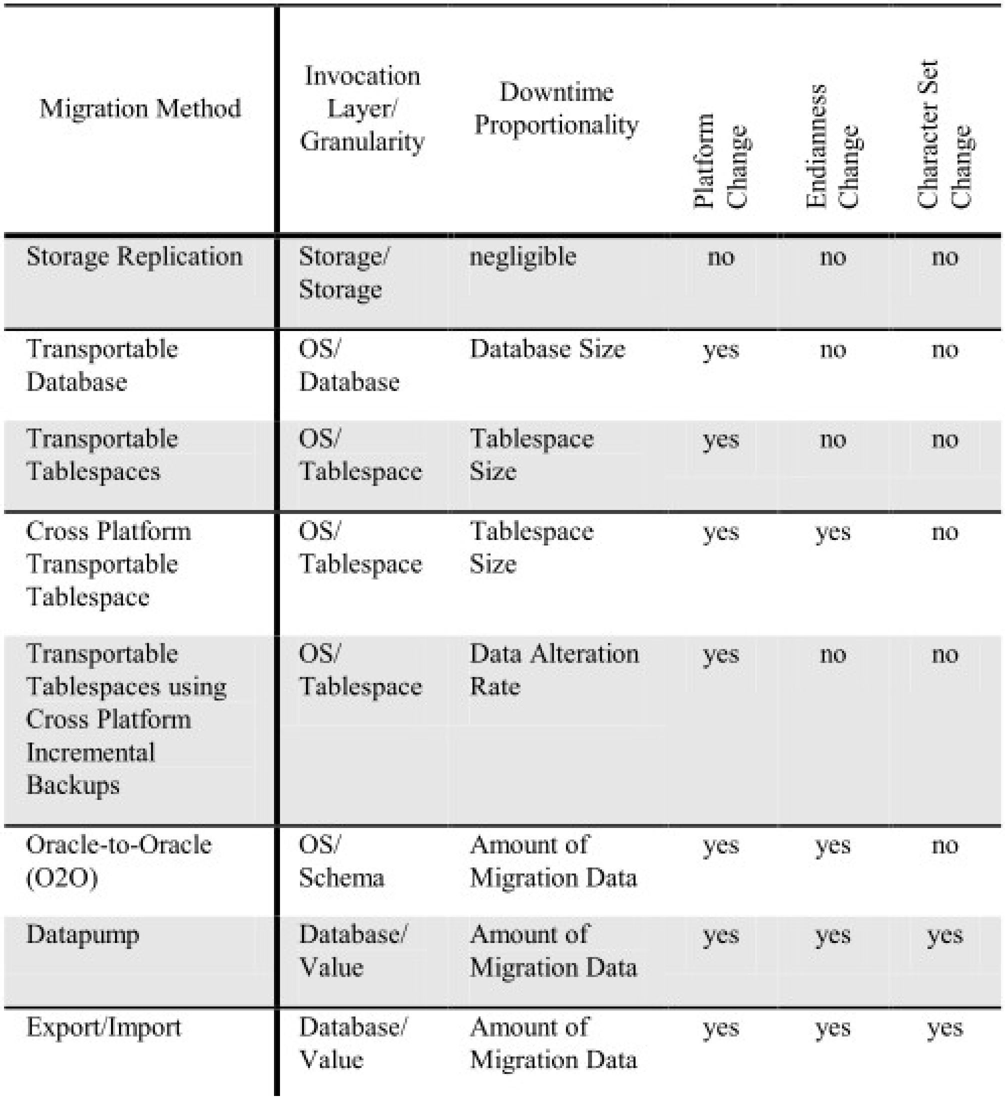
Adapted from [2]



Source System	01100011	01010001	01101110	11110101	10100000	10101110
Target System	10101110	10100000	11110101	01101110	01010001	01100011

Migration Approach Differentiation

- Invocation layer
  - Storage
  - OS
  - Database
- change of platform
- change of endianness
- change of character set
- Downtime proportionality
  - Size of migration data
  - Data alteration rate

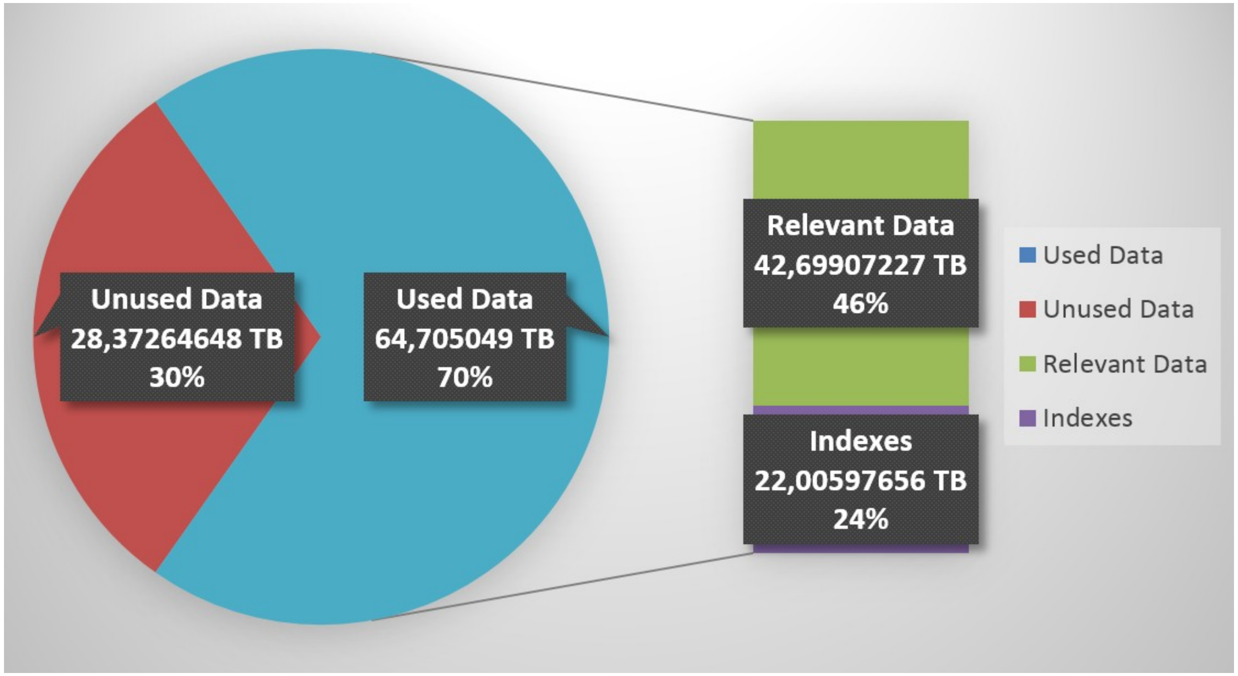


Migration Method	Invocation Layer/ Granularity	Downtime Proportionality	Platform Change	Endianness Change	Character Set Change
Storage Replication	Storage/ Storage	negligible	no	no	no
Transportable Database	OS/ Database	Database Size	yes	no	no
Transportable Tablespaces	OS/ Tablespace	Tablespace Size	yes	no	no
Cross Platform Transportable Tablespace	OS/ Tablespace	Tablespace Size	yes	yes	no
Transportable Tablespaces using Cross Platform Incremental Backups	OS/ Tablespace	Data Alteration Rate	yes	no	no
Oracle-to-Oracle (O2O)	OS/ Schema	Amount of Migration Data	yes	yes	no
Datapump	Database/ Value	Amount of Migration Data	yes	yes	yes
Export/Import	Database/ Value	Amount of Migration Data	yes	yes	yes

Prior Analysis

Average Structure of Allocated Data

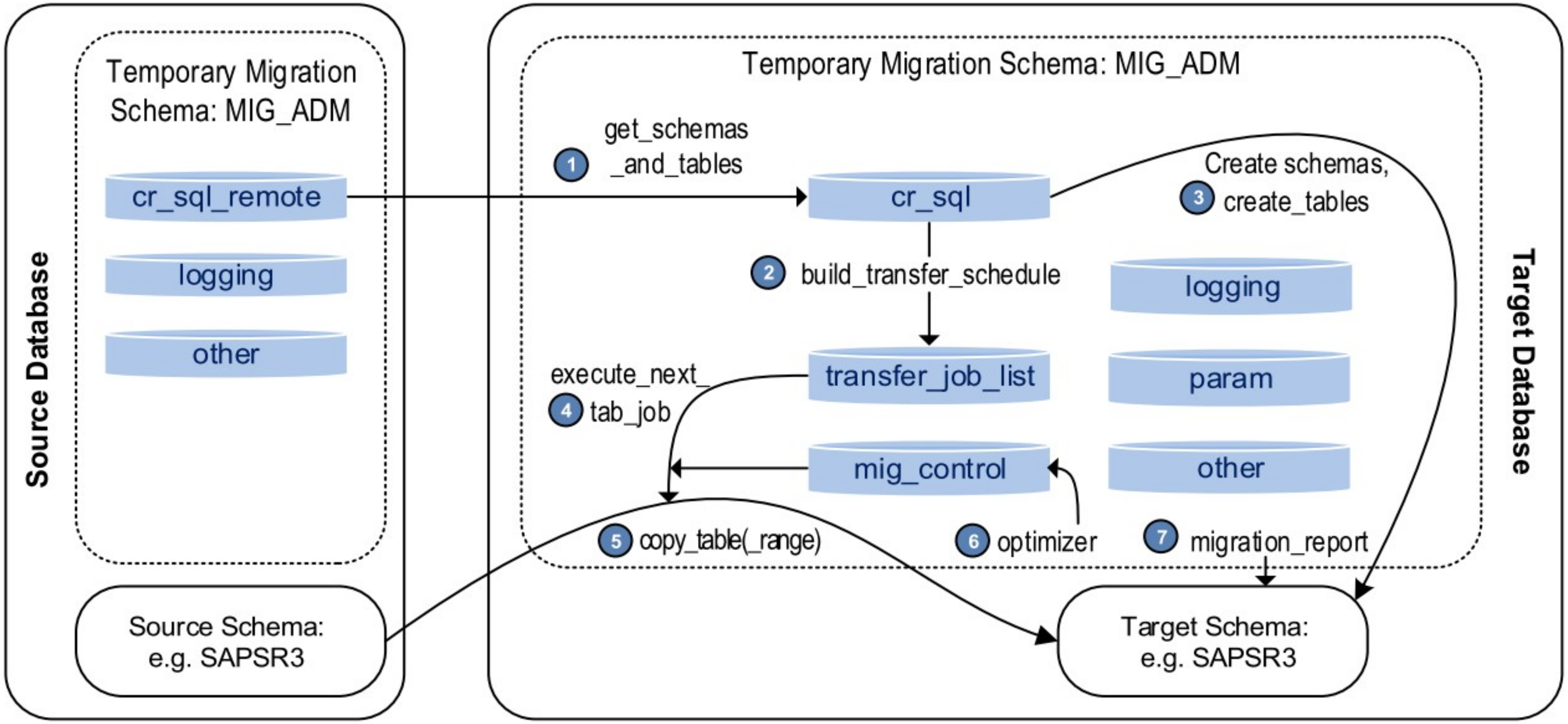
- (based on 41 productively running SAP Systems)



- irrelevant data can be excluded when migrating on **logical database layer**

HiPAS Architecture

- Everything is a tuple


Source Database
Temporary Migration Schema: MIG_ADM
cr_sql_remote
logging
other
Source Schema: e.g. SAPSR3
Temporary Migration Schema: MIG_ADM
1 get_schemas _and_tables
cr_sql
Create schemas, 3 create_tables
2 build_transfer_schedule
logging
transfer_job_list
param
execute_next_ 4 tab_job
mig_control
other
5 copy_table(_range)
6 optimizer
7 migration_report
Target Schema: e.g. SAPSR3
Target Database

Adaptive Data Transfer

- Enabling adaptive behavior during transfer phase
  - partitioning into equally sized transfer bundles
  - Number of running transfer jobs can be reduced or increased
- Two approaches were developed and evaluated
  - **Adaption**: based on an incremental adjustment process, until changes do not evoke further improvements, thus, reaching the state of an optimal parallelization degree
  - **Anticipation**: makes continuously new modification decisions independently of each other, based on knowledge about used and monitored resources

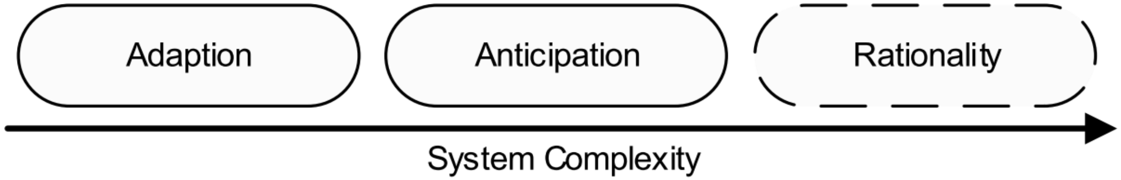


Figure adapted from [3]

Self-Adaptive Software

“Self-adaptive software evaluates its own behavior and **changes behavior when** the evaluation indicates that it is not accomplishing what the software is intended to do, or when **better** functionality or **performance is possible**.” [4]

“Self-adaptive software **modifies** its own **behavior in response to** changes in its operating **environment**. […]” [5]

- Self-Properties of self-adaptive software [6]
  - Self-configuring
  - **Self-healing**
  - **Self-optimizing**
  - Self-protecting

Design Space Dimensions

Observation

- Environment-Awareness
  - Storage System
  - CPU
  - Memory
- Self-Awareness
  - Number of running jobs

Presentation

- Concurrency events
- Average write time
- Average read time
- Redo log buffer size
- Available memory size
- etc.

Control

Master/slave control pattern in distributed

- system

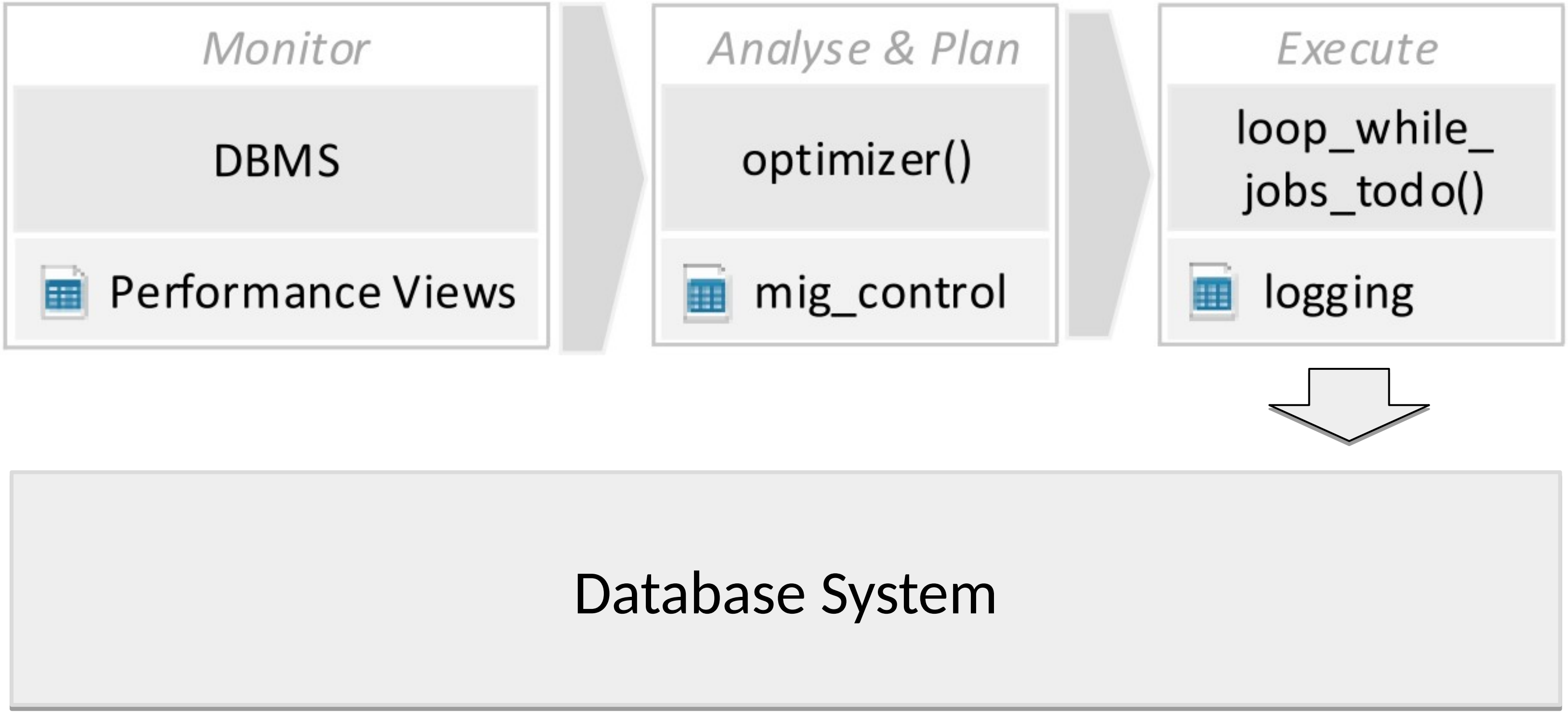
Identification and Enabling Adaption

- Plugin architecture
- Table MIG_Control as interface

[7]

Adaptive Capabilities of HiPAS

- „Optimizer“ plugin for data transfer phase
- acts according to MAPE feedback loop [6]



Monitor

DBMS

Performance Views

Analyse & Plan

optimizer()

mig_control

Execute

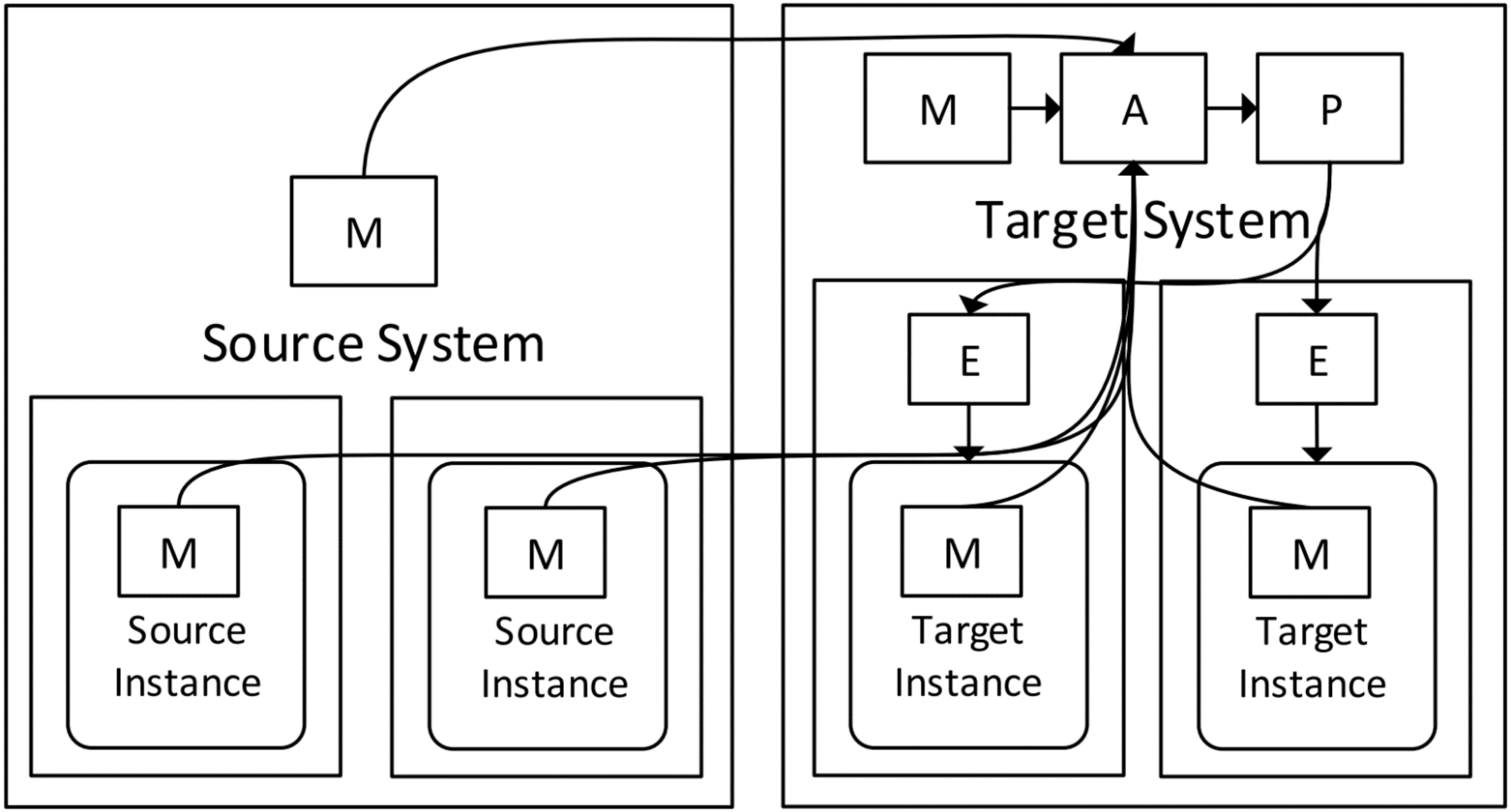
loop_while_ jobs_todo()

logging

Database System

Master/Slave Control Pattern

- Control Dimension

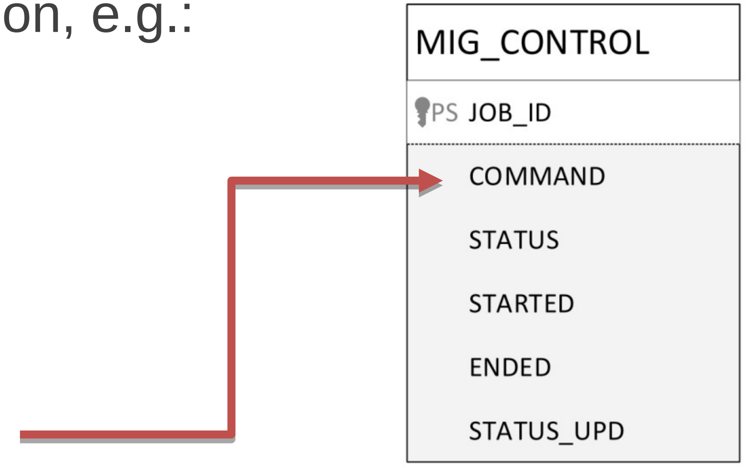


Adapted from [7]

Monitor, Analyse and Plan

PASOLFORA

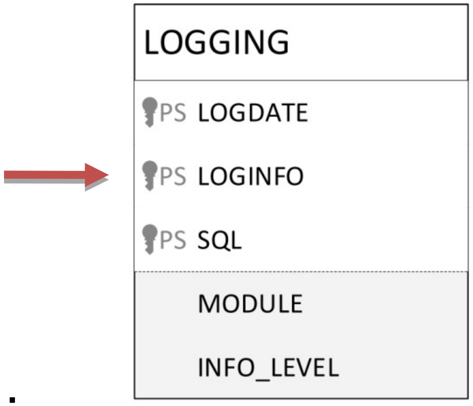
- Optimizer analyses system information, e.g.:
  - Concurrency events
  - Average write/read time
  - Redo log buffer size
  - Available memory size
- Optimizer plans:
  - writes “STOP“/“CONTINUE“-command
- Optimizer writes log string:



MIG_CONTROL

PS	JOB_ID
	COMMAND
	STATUS
	STARTED
	ENDED
	STATUS_UPD

```
“Prev  Jobs:  40/  Jobs:  40  Max  Jobs:  400  #
Read  Avg:  3.32(20-40) # Write Avg:  105.9(100-
200) # R_Read Avg:  .12(20-40) # R_Write Avg:  .
3(20-40) # R Fail Ind:  3 conc:3026(2607)
redo:5720763732(5776886904) r_conc:5157(5069) #
numjobs > 0 # Jobs being stopped: 0  #  (Resource
Overload)  and  numjobs  >  minjobs  and
jobs_being_stopped  =  0  #  Running:  20/Stopping:
 5  on inst:1 # Running: 20/Stopping: 5 on inst:2”
```



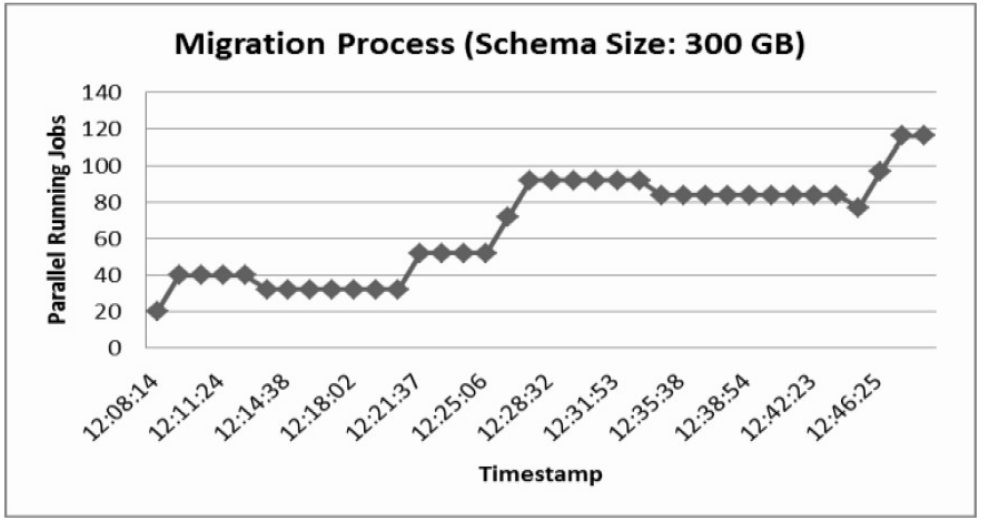
LOGGING

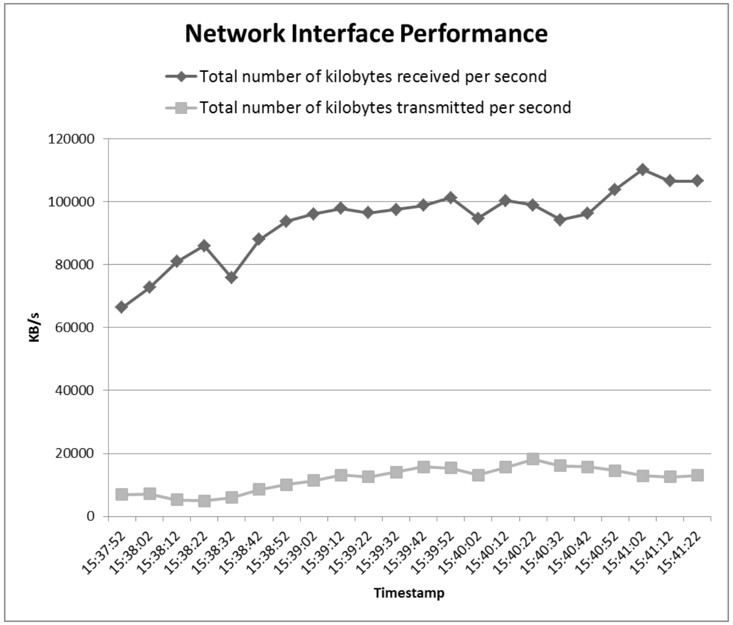
PS	LOGDATE
PS	LOGINFO
PS	SQL
	MODULE
	INFO_LEVEL

Evaluation

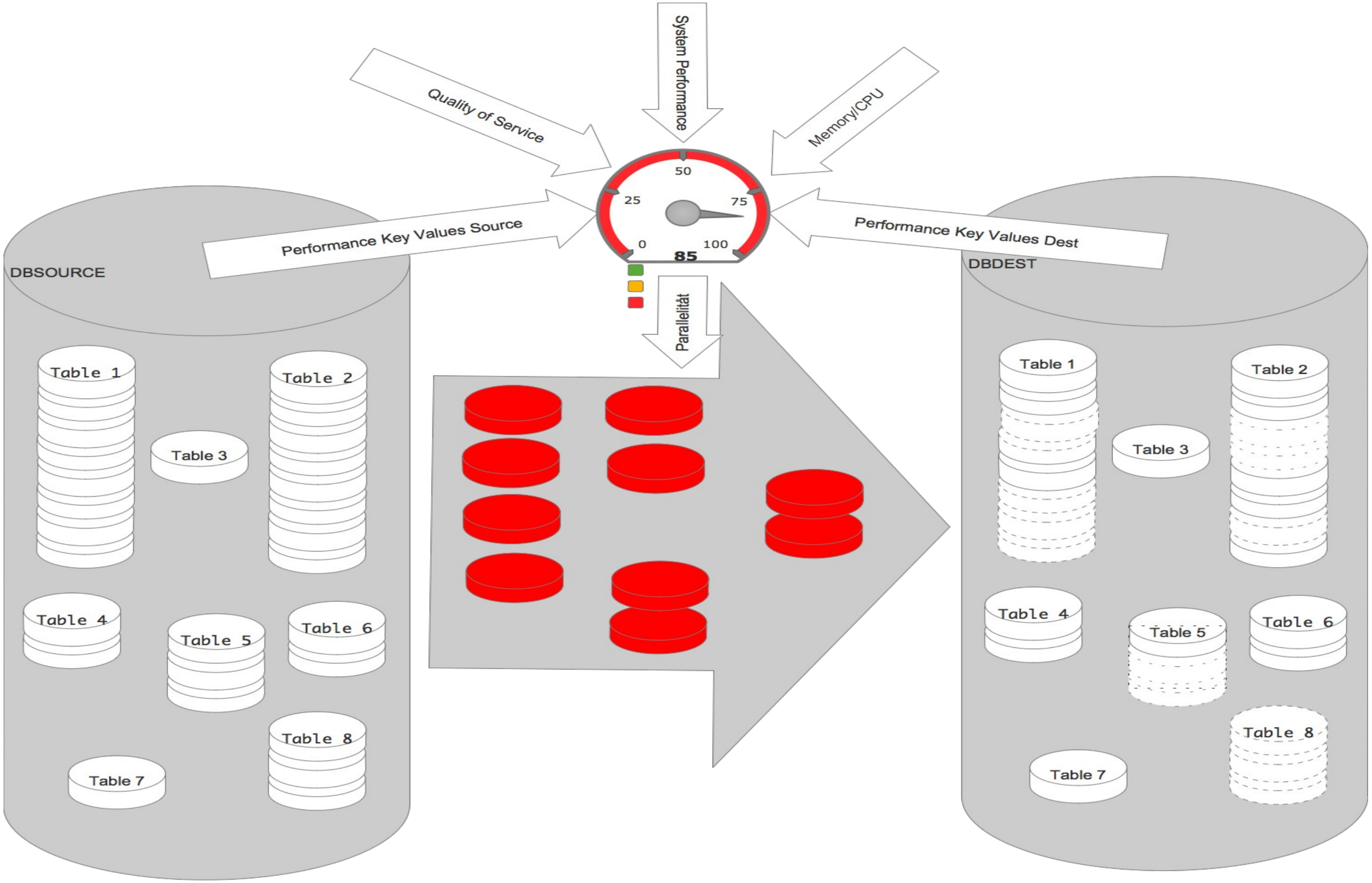
PASOLFORA

- Adaption of parallelization degree according to system environment and migration data
- ~ 123 MByte/s per 1 gbit network interface
- ~ 1 GByte/s per 10 gbit network interface





How does it work ?



How does it work ?

- PL/SQL only
- SQLNET only
  - no temporary Storage necessary
- Source and Destination RAC aware
  - automatic multi instance parallelization
- Everything protected by Oracle transactional integrity
  - no data loss possible
  - Restart after failure / server / network outage
    - automatically
    - no Re-copy of row sets
- Parallel Index Build

How does it work ?

- dbms_metadata on source
- Stats on source
- create table extents on dest
- PL/SQL Long to LOB conversion on source
- University Solution for transfer parallelization
- create dbms_scheduler jobs
- transfer table rows, LOBs
- calibrate IO / Auto DOP for indexing on dest
- Count rows and select „source“ minus „dest“
- generate compliance report

Conclusions

- non-adaptive and sequential migrations leave useful resources idle or need to be tuned manually
  - „self adaptive is always better“
- logical transfer
  - platform, version, endianess and character set independent
- Ultra Fast parallel LOB interface
- Copy Performance of 3 to 5 TByte per Hour
  - adequate Network and I/O Bandwith necessary
- Easy Fallback – source stays untouched

Conclusions

- Remap everything
  - User
  - Tablespaces
  - Table / Tablespace Mapping
  - create object attributes
  - Index table compression
- Compliance Check
- Diff Report for rows and metadata

Minimum Downtime Option

- works without EE or Partitioning
- provides same functionality and benefits
  - easy fallback
  - protected by oracle transactional integrity
  - Remap everything
  - Diff Report for all rows and metadata

Minimum Downtime - Capture

- Capture changes while transfer base data
  - List of Transactions
    - Trigger
      - generate list of changes SCN based
      - Old Value / New Value / SCN / ID
    - Uses Log Stream to doublecheck
- Generates List of Sqls
  - Capture / Apply to other DB Platforms possible
- Parallel Capture and Apply

Minimum Downtime – SCN Copy

- Dirty Read Option dismissed
  - „Dirty“ Reads (different SCNs per Rowset)
    - merge changes at the end of transfer
      - Row need apply / Row newer than change
    - like Oracle Recovery
- Select … as of ….. (same SCN for all Rowsets)
    - Undo Guarantee
    - generates insert sqls for multi DB Plattform
    - Trigger on Large Tables
    - Small Tables in switchover downtime
    - apply list of changes ordered

Replication

PASOLFORA

- Initial Load by Hipas Base Schema Transfer
- Replication based on Hipas capture
- Trigger based
  - thin and fast implementation (rac aware)
  - blacklist / whitelist
    - object / column
  - generates list of sqls
    - replications to other db platforms possible

Replication

- Self Repair / Healing after Outtages
  - log stream to extract / apply gaps
- Management by GUI
- CDC / Streams alternative
- Parallel Capture and Apply
- EE or Partitioning not necessary

Presentation References

1) A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design Science in Information Systems Research”. MIS Quarterly
2) Vol. 28 No.1., 2004, p. 80. G. Somasundarum and A. Shrivastava, Information Storage and Management - Storing, Managing, and Protecting Digital Information. EMC Education Services, Wiley Publishing Inc. Inianapolis 2009, p. 35.
3) J. A. Martin Hernandez, J. de Lope and D. Maravall, “Adaptation, anticipation and rationality in natural and artificial systems: computational paradigms mimicking nature.”, Natural Computing, Volume 8, Issue 4, Springer Netherlands, 2009, pp. 758-765.
4) R. Laddaga, Self-adaptive software. Tech. Rep. 98-12, DARPA BAA., 1997
5) P. Oreizy, M. M. Gorlick , R. N. Taylor, D. Heimbigner, G. Johnson, N. Medvidovic, A. Quilici,, D. S. Rosenblum, and A. L. Wolf, An architecture-based approach to self-adaptive software, IEEE Intel. Syst., 1999
6) IBM. An architectural blueprint for autonomic computing. Tech. rep., IBM. 2003.
7) Y. Brun, R. Desmarais, K Geihs, M. Litoiu, A. Lopes, M. Shaw, and M. Smit, A Design Space for Self-Adaptive Systems