HiPAS
High Performance Adaptive
Schema Migration with
Minimum Downtime Option
Background

- Minimum Downtime Schema Migration and Continuous 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

Adapted from Information Systems Research Framework [1]

- **People**
  - Usability

- **Organizations**
  - License Costs
  - Platform Change
  - Downtime Shortness

- **Technology**
  - Compatibility
  - Reliability
  - Interruptibility
  - No temporary storage

**Developed Artefact**
- HiPAS
- Utilizing Adaption for Database Migrations

**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

**Environment**

**IS Research**

**Knowledge Base**

**Business Needs**

**Applicable Knowledge**

**Additions to the Knowledgebase**

**Application in the Environment**

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]
## 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

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

- 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:

```
"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) #
umjobs > 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"
```
Evaluation

- 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

- 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


