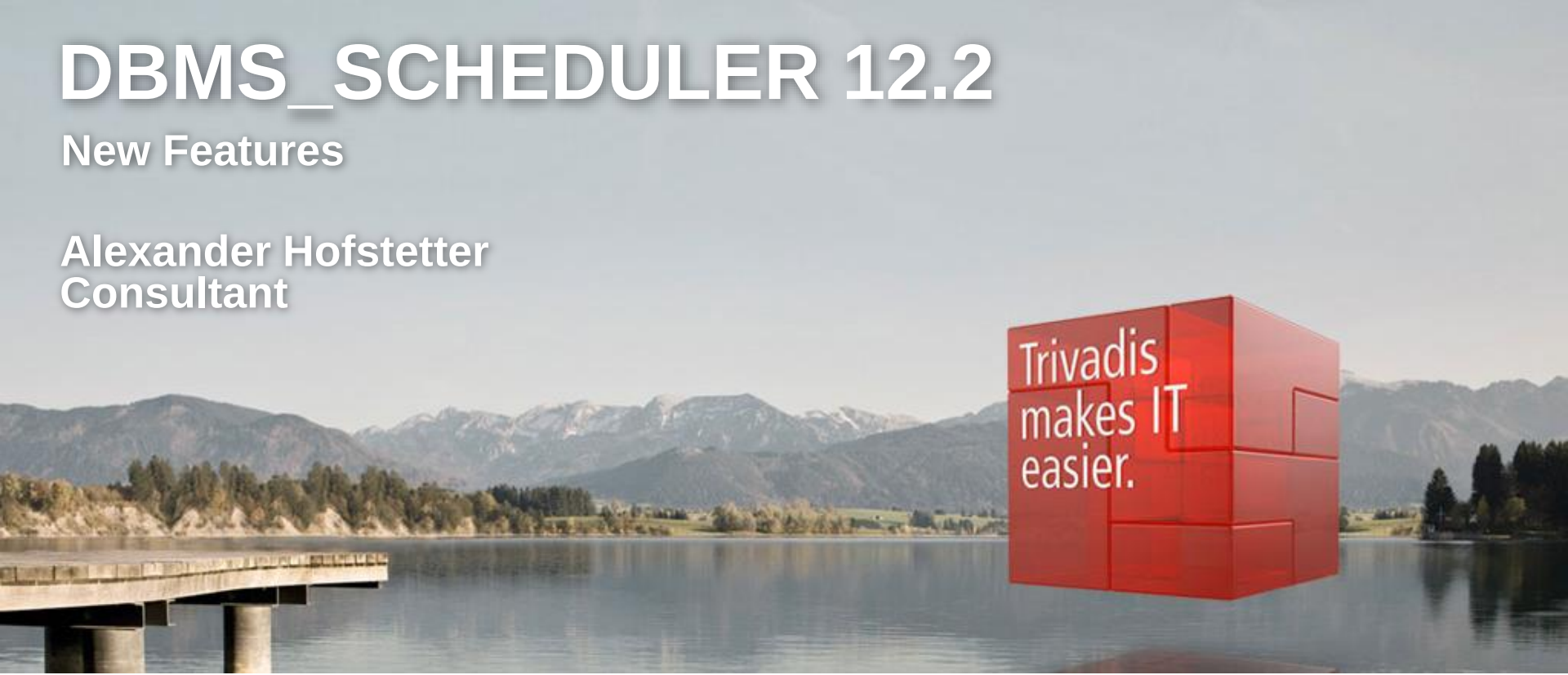
DBMS_SCHEDULER 12.2

New Features

Alexander Hofstetter
Consultant



About me

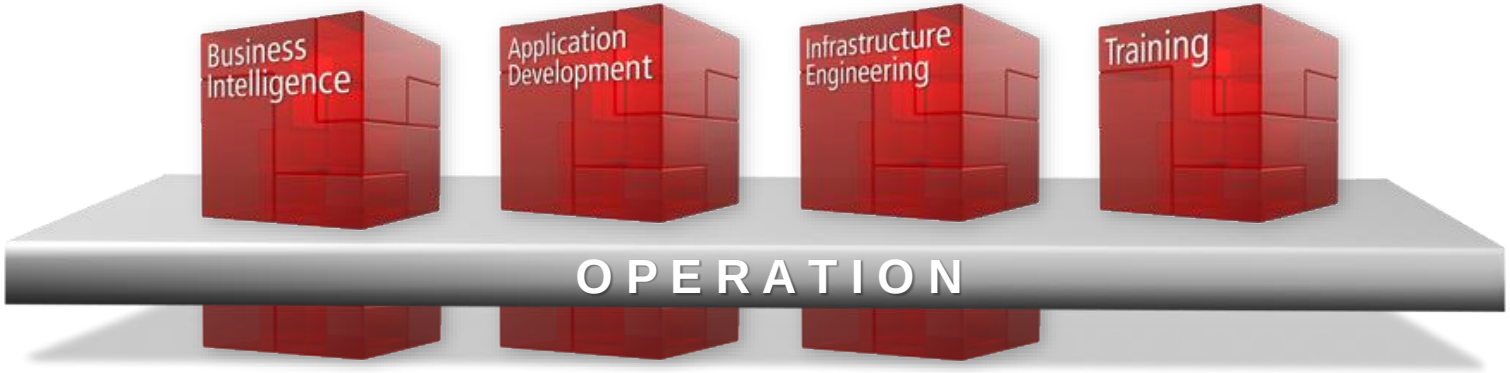
- Consultant, Trivadis GmbH, Munich
- Since 2008 Oracle DBA
- Working with Oracle since 2005
- Latest Projects
  - High Availability
  - Automation
  - Migration & Upgrades

ORACLE
Certified Professional
Oracle Database 11*g*
Administrator

trivadis
makes IT easier.

Our company.

Trivadis is a market leader in IT consulting, system integration, solution engineering and the provision of IT services focusing on ORACLE and Microsoft technologies
in Switzerland, Germany, Austria and Denmark. We offer our services in the following strategic business fields:



Trivadis Services takes over the interacting operation of your IT systems.

trivadis makes IT easier.

With over 600 specialists and IT experts in your region.

- 14 Trivadis branches and more than 600 employees
- 200 Service Level Agreements
- Over 4,000 training participants
- Research and development budget: CHF 5.0 million
- Financially self-supporting and sustainably profitable
- Experience from more than 1,900 projects per year at over 800 customers

trivadis
makes IT easier.

Agenda

1. Scheduler Basics
2. Resource Management
3. Scheduling
4. New Features 12.2
5. Conclusion

Scheduler Basics

Introduction

- Implemented in 10gR1
- Time- and event-based scheduling
- Monitoring, logging and reporting possibilities
- Resource management

Architecture

- sys.scheduler$_job Table

- The job coordinator (cjqNNN process) controls the jobs
  - Queries the job table
  - Spawns and cleanup slave processes if needed
  - Switches to sleep mode and wake up
  - There is one coordinator per instance in a RAC environment

Architecture II

- Job Execution by Job Slave
  - Gathers all required information about the job
  - Starts a database session
  - Opens a transaction and starts the job
  - After completion transaction is commited and closed
  - Closes the the database session

Architecture III

- After completion
  - Run Count
  - Logging
  - Rescheduling if required
  - Look for new work or go to sleep

Job Categories

- **Database Job**
  - PL/SQL Block
  - Stored Procedure
- **External Job**
  - Executable
- **Script Job**
  - SQL Script
  - External Script
  - Backup Script

Job Categories II

- Chain Job
- Lightweight Job
- Multiple-destination Job
- Detached Job
- In-Memory-Job

Resource Management

Resource management

- **Resource Manager**
  - Provides more control over hardware resources
  - Limits the amount of CPU usage
  - Limit the degree of parallelism and priority
  - Limit the amount of active sessions
  - Move session to another group

Resource management II

- **Job Classes**
  - Assign Consumer Group
  - Assign Service
  - Define logging level
  - Created in SYS schema
  - MANAGE SCHEDULER privilege is necessary

Resource management III

- Windows
  - Assign Resource Plan
  - Only one Window can be active
  - Overlapping is possible (not recommended)
  - Manage Scheduler privilege is necessary

Scheduling

Scheduling

- **Time-based scheduling**
  - Start at a particular date and time
  - Once or repeating, even complex intervals
- **Event-based scheduling**
  - Job starts based on an event
  - There a two kindes of event

Time-based Scheduling

- start_date, repeat_interval, end_date
- repeat_interval can be Calender Expression or PL/SQL Expression
  - Calendar: `FREQ=MINUTELY; INTERVAL=5;`
  - PL/SQL: `SYSTIMESTAMP + INTERVAL '5' MINUTE`
- **FREQ, INTERVAL**, BYMONTH, BYWEEKNO, BYYEARDAY, BYDATE,BYMONTHDAY,BYDAY,BYHOUR,BYMINUTE,BYSECOND,BYSETPOS, INCLUDE,EXCLUDE,INTERSECT,PERIODS,BYPERIOD

- Syntax available at:

http://tinyurl.com/jo4645n

Scheduling Examples

- Every 2nd Thursday

```
FREQ=Weekly; INTERVAL=2; BYDAY=THU
```

- Last workday of a month

```
FREQ=MONTHLY; BYDAY=MON,TUE,WED,THU,FRI; BYSETPOS=-1;
```

- Every 2nd hour every 5 minutes from Monday to Friday

```
FREQ=MINUTELY;INTERVAL=5;BYDAY=MON,TUE,WED,THU,FRI;
BYHOUR=0,2,4,6,8,10,12,14,16,18,20,22
```

New Features 12.2

Job Incompatibilities

- You can define an incompability between two or more jobs
- Only one of the group can be running at a time
- Incompatibility can be defined at job or program level

Job Incompatibilities – Job Level

- Only one Job in the definition can run at the time
- All other jobs in the same group have to wait until first jobs finishes

Example: you have got one Incompatible Group with JOB A, B und C

while JOB A runs, you can‘t start JOB B or C.

Job Incompatibilities – Program Level

- Only Jobs running the same Program can run at a time
- All other jobs, using other programs in the same group have to wait until all jobs finish.

Example: you have got one Incompatible Group with Program A, B und C

Job 1+2 run Program A ; Job 3+4 run Program B; Job 5+6+7+8 run Program C

While a Job 5 with Program C is running, only Job 6, 7 and 8 would run

Job 1,2,3 and 4 can‘t start as long a job with Program C is running

Job Incompatibilities Procedures

- CREATE_INCOMPATIBILITY
- ADD_TO_INCOMPATIBILITY
- REMOVE_FROM_INCOMPATIBILITY
- DROP_INCOMPATIBILITY

Job Incompatibilities: CREATE_INCOMPATIBILITY

```
DBMS_SCHEDULER.CREATE_INCOMPATIBILITY (
   incompatibility_name     IN VARCHAR2,
   object_name              IN VARCHAR2,
   constraint_level         IN VARCHAR2 DEFAULT 'JOB_LEVEL',
   enabled                  IN BOOLEAN DEFAULT TRUE,
   comments                 IN VARCHAR2 DEFAULT NULL);
```

```
BEGIN
dbms_scheduler.create_incompatibility(
  incompatibility_name => 'incompatible1',
  object_name => 'job1,job2,job3',
  enabled => true );
END;
/
```

Job Incompatibilities ADD_TO_INCOMPATIBILITY

```
DBMS_SCHEDULER.ADD_TO_INCOMPATIBILITY (
   incompatibility_name    IN VARCHAR2,
   object_name             IN VARCHAR2);
```

```
BEGIN
dbms_scheduler.add_to_incompatibility(
  incompatibility_name => 'incompatible1',
  object_name => 'job4');
END;
/
```

Job Incompatibilities REMOVE_FROM_INCOMPATIBILITY

```
DBMS_SCHEDULER.REMOVE_FROM_INCOMPATIBILITY (
   incompatibility_name    IN VARCHAR2,
   object_name             IN VARCHAR2);
```

```
BEGIN
dbms_scheduler.remove_from_incompatibility(
  incompatibility_name => 'incompatible1',
  object_name => 'job2');
END;
/
```

Job Incompatibilities DROP_INCOMPATIBILITY

```
DBMS_SCHEDULER.DROP_INCOMPATIBILITY (
   incompatibility_name     IN VARCHAR2);
```

```
BEGIN
dbms_scheduler.drop_incompatibility(
  incompatibility_name => 'incompatible1');
END;
/
```

Job Incompatibilities Views

- USER_SCHEDULER_INCOMPATS
- USER_SCHEDUER_INCOMPAT_MEMBER

```
USER_SCHEDULER_INCOMPATS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 INCOMPATIBILITY_NAME                      NOT NULL VARCHAR2(128)
 CONSTRAINT_LEVEL                                   VARCHAR2(13)
 ENABLED                                            VARCHAR2(5)
 JOBS_RUNNING_COUNT                                 NUMBER
 COMMENTS                                           VARCHAR2(256)
```

Job Incompatibilities Views

```
USER_SCHEDULER_INCOMPAT_MEMBER
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 INCOMPATIBILITY_OWNER                     NOT NULL VARCHAR2(128)
 INCOMPATIBILITY_NAME                      NOT NULL VARCHAR2(128)
 OBJECT_OWNER                              NOT NULL VARCHAR2(128)
 OBJECT_NAME                               NOT NULL VARCHAR2(128)
```

Resource Queues

Resource Queues

- Specifies how many resources are required to execute a Job
- Resource can be specified by name and count
- At execution time Scheduler ensures that available resources are not exceeded
- Jobs can‘t run until required resources are available

Resource Queues Procedures

- CREATE_RESOURCE
- SET_ATTRIBUTE / _NULL
- SET_RESOURCE_CONSTRAINT
- DROP_RESOURCE

Resource Queues CREATE_RESOURCE

```
DBMS_SCHEDULER.CREATE_RESOURCE (
  resource_name     IN VARCHAR2,
  units             IN PLS_INTEGER,
  status            IN VARCHAR2 DEFAULT 'ENFORCE_CONSTRAINTS',
  constraint_level  IN VARCHAR2 DEFAULT 'JOB_LEVEL',
  comments          IN VARCHAR2 DEFAULT NULL);
```

```
BEGIN
   DBMS_SCHEDULER.CREATE_RESOURCE(
      resource_name => 'app_resource1',
      units => 2,
      status => 'ENFORCE_CONSTRAINTS',
      comments => 'Resource1');
END;
/
```

Resource Queues SET_RESOURCE_CONSTRAINT

```
DBMS_SCHEDULER.SET_RESOURCE_CONSTRAINT (
   object_name         IN VARCHAR2,
   resource_name       IN VARCHAR2,
   units               IN NUMBER DEFAULT 1);
```

```
BEGIN
    DBME_SCHEDULER.SET_RESOURCE_CONSTRAINT(
         OBJECT_NAME   => 'job1',
         RESOURCE_NAME => 'app_resource1',
         UNITS         =>1);
END;
/
```

Resource Queues DROP_RESOURCE

```
BEGIN
   DBMS_SCHEDULER.DROP_RESOURCE(
      resource_name => 'app_resource1',
      force         => true
   )
END;
/
```

Resource Queues Views

- USER_SCHEDULER_RESOURCES

```
USER_SCHEDULER_RESOURCES
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 RESOURCE_NAME                             NOT NULL VARCHAR2(128)
 STATUS                                             VARCHAR2(19)
 RESOURCE_UNITS                                     NUMBER
 UNITS_USED                                         NUMBER
 JOBS_RUNNING_COUNT                                 NUMBER
 COMMENTS                                           VARCHAR2(256)
```

Resource Queues Views

- USER_SCHEDULER_RSC_CONSTRAINTS

```
USER_SCHEDULER_RSC_CONSTRAINTS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 OBJECT_OWNER                              NOT NULL VARCHAR2(128)
 OBJECT_NAME                               NOT NULL VARCHAR2(128)
 RESOURCE_OWNER                            NOT NULL VARCHAR2(128)
 RESOURCE_NAME                             NOT NULL VARCHAR2(128)
 UNITS_USED                                         NUMBER
```

In-Memory Jobs

In-Memory Jobs

- Should be created and run during a short period of time
- In-memory jobs have a larger memory footprint
- There are 2 types of in-memory jobs:
  - IN_MEMOR_RUNTIME
  - IN_MEMORY_FULL

IN_MEMORY_RUNTIME

- Based on Lightweight Jobs (persistent)
- Repeatable
- Run with DEFAULT_IN_MEMORY_JOB_CLASS
  - Logging level = none
  - No information in scheduler views

IN_MEMORY_FULL

- Not persistent, only in memory
- Must be associated to a program
- Not repeatable, supposed to run once
- Do not generate Redo at create or during runtime
- On RAC job is only present where created
- They are not propagated to physical or logical standby

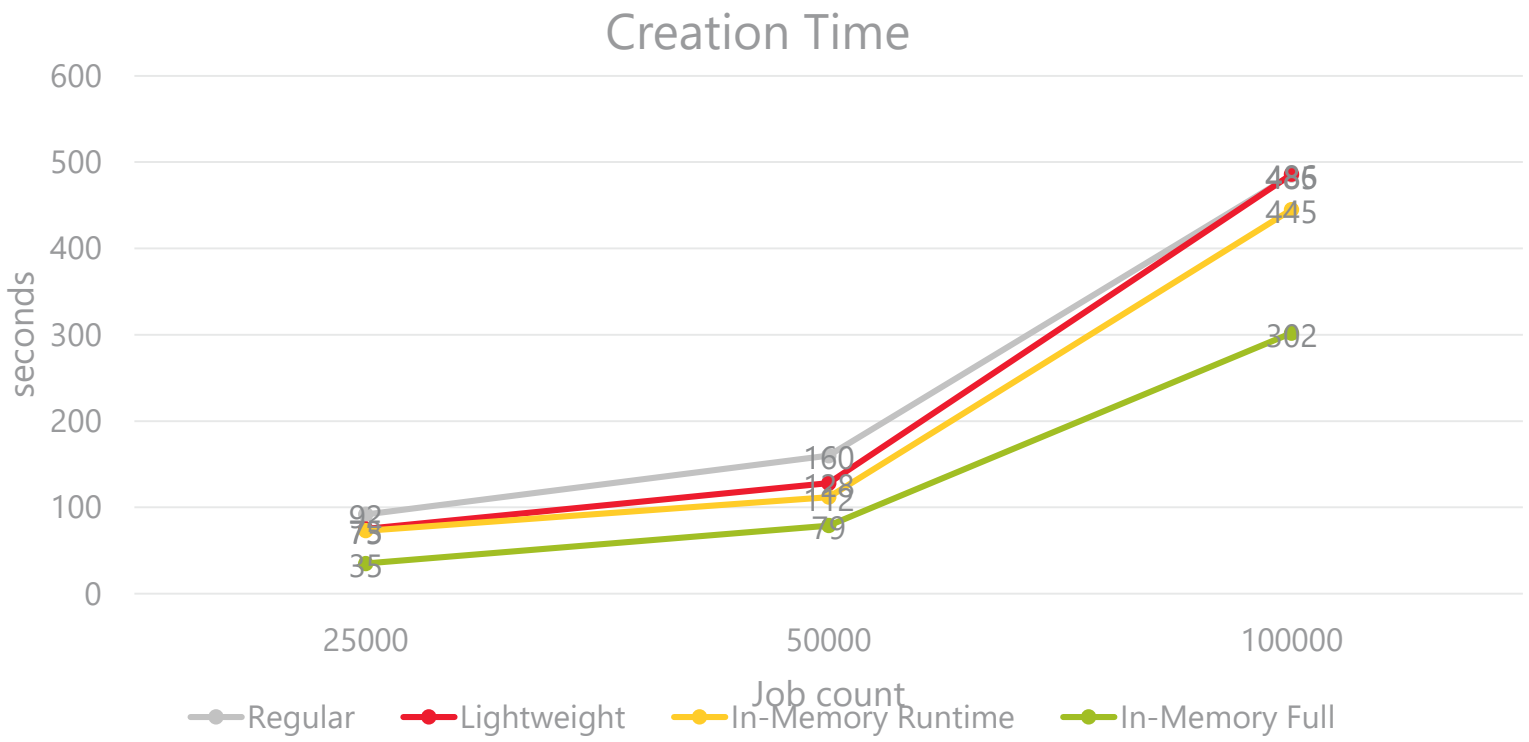
In-Memory Runtime Job Example

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
    job_name => 'my_repeat_job',
    program_name => 'repeat_prog',
    start_date => systimestamp,
    repeat_interval => 'freq=secondly;interval=10',
    job_style => 'IN_MEMORY_RUNTIME',
    enabled => true);
END;
/
```

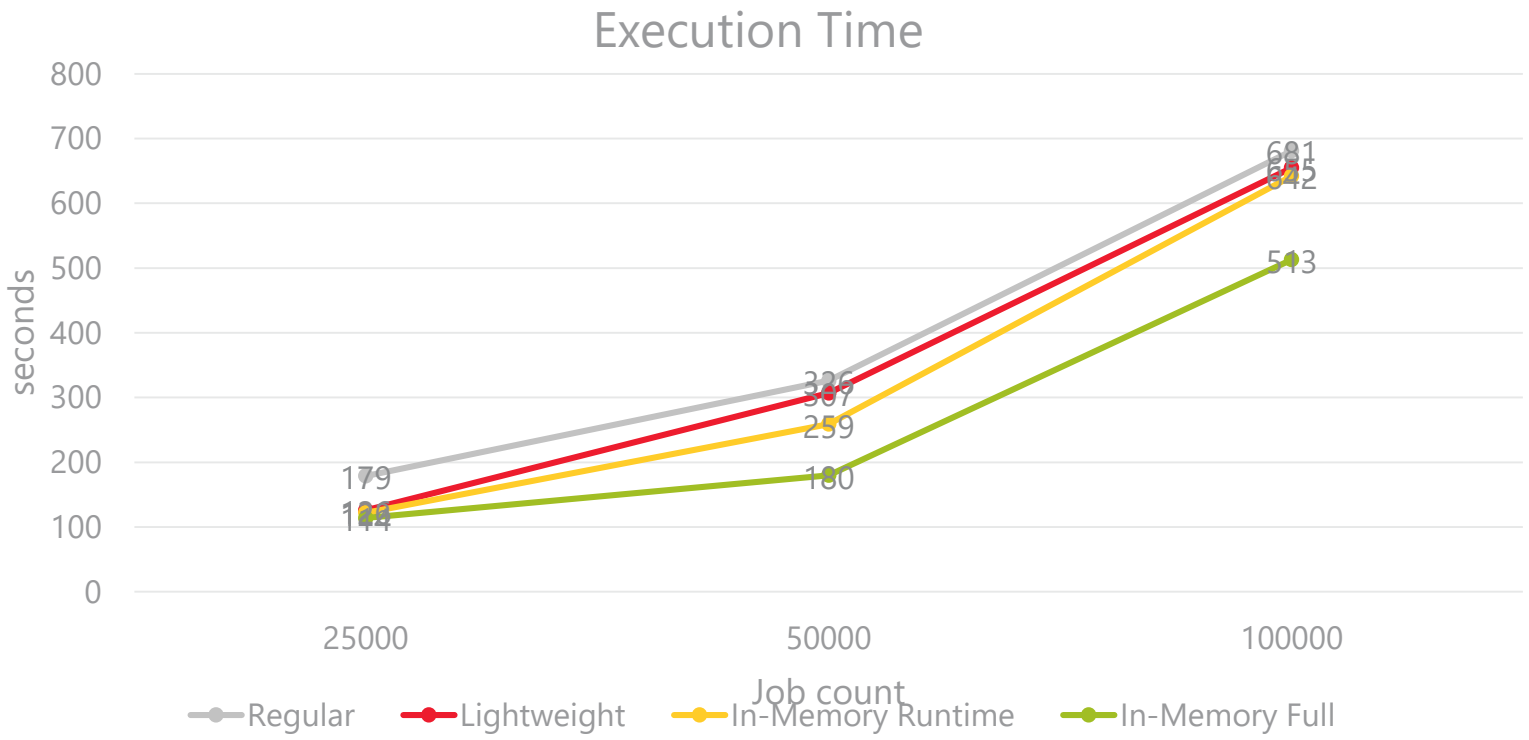
In-Memory Full Job Example

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
    job_name => 'my_immediate_job',
    program_name => 'fast_op',
    job_style => 'IN_MEMORY_FULL',
    enabled => true);
END;
/
```

Job Type Comparison



Job Type Comparison



In-Memory Views

- V$SCHEDULER_INMEM_MDINFO

Name	Null?	Type
OBJID		NUMBER
PRGOID		NUMBER
LAST_ENABLED_TIME		TIMESTAMP(3) WITH TIME ZONE
CLSOID		NUMBER
INSTANCE_ID		NUMBER
FLAGS		NUMBER
CREATOR		VARCHAR2(128)
CLIENT_ID		VARCHAR2(65)
GUID		VARCHAR2(33)
CON_ID		NUMBER

In-Memory Views

- V$SCHEDULER_INMEM_RTINFO

```
Name                                      Null?    Type
----------------------------------------- -------- --------------------
USERID                                             NUMBER
OBJID                                              NUMBER
ID_TYPE                                            NUMBER
NAME                                               VARCHAR2(128)
NEXT_RUN_DATE                                      TIMESTAMP(3) WITH TIME ZONE
LAST_START_DATE                                    TIMESTAMP(3) WITH TIME ZONE
LAST_END_DATE                                      TIMESTAMP(3) WITH TIME ZONE
RUN_COUNT                                          NUMBER
FAILURE_COUNT                                      NUMBER
RUNNING_INSTANCE                                   NUMBER
RUNNING_SLAVE                                      NUMBER
JOB_STATUS                                         NUMBER
CON_ID                                             NUMBER
```

Conclusion

Conclusion

- New Possibilites to regulate jobs
  - Job Incompatibilities defines jobs / programs which can‘t run parallel
  - Resource Queues defines a number of resource which can‘t be exceeded

- New Possibilities for extra fast jobs
  - In-Memory Runtime job for a persistent, repeatable jobs
  - In-Memory Full job for an extra fast one time executable job

Any questions…?

Alexander Hofstetter

Tel: +49 89 99 27 59 302
Alexander.Hofstetter@trivadis.com

trivadis
makes IT easier.