Easily implement parallel processing in online and batch processing

From time to time I faced the problem of slow performance and/or too much use of temporarily needed memory storage space.

One possible solution is to split up the whole task into packages that may be processed online or in background. This will not necessarily save any time. Also we face the problem of collecting the results in one single list or whatever form of output we choose.

The next approach is parallel processing: If we create background jobs and let the operating system schedule the run as soon as possible, we have the administrative task of checking all Jobs have finished their run successfully. Then, as mentioned, the collecting and combining results store in spool lists or in other locations.

ABAP allows to use the of CALL FUNCTION func ...STARTING NEW TASK task name.

This allows to process several packages in parallel. The addition ... PERFORMING form ON END OF TASK allows to receive the results.

Unfortunately, the documentation on this is not much and a couple of questions and searches in the ABAP forums did not give sufficient answers.

It looks comparably easy to create an RFC function module to process packages of the whole workload and start it parallel in a task group using .. DESTINATION IN GROUP group name.

According to documentation, asynchronous RFC calls are processed by dialog tasks. This implies the same time limits (usually 5 minutes) apply here as in 'normal' online work. And, because the number of available tasks is limited, this time may be easily exceeded.

After doing some experimental work I found out that after dispatching packets to the available tasks, the program must execute the WAIT statement to enable the PERFORMING form ON END OF TASK take effect and the form being executed.

Finally I created a sample program including a small class for the task handling. It is just a test program using the BAPI function BAPI_BILLINGDOC_GETLIST retrieving packets determined by ranges of document numbers. It can be easily remodeled to handle anything.

Error rendering macro 'code': Invalid value specified for parameter 'lang'

*---------------------------------------------------------------------*
* Report  ZZZPARTEST                                                 *
*----------------------------------------------------------------------*
*----------------------------------------------------------------------*
*----------------------------------------------------------------------*
REPORT zzzpartest.
PARAMETERS:
p_dbcnt                         TYPE sydbcnt DEFAULT 1010,
p_pacsz                         TYPE sydbcnt DEFAULT 95.
CONSTANTS:
gc_function                     TYPE tfdir-funcname
    VALUE 'BAPI_BILLINGDOC_GETLIST'.
DATA:
gt_bapivbrksuccess             TYPE TABLE OF
    bapivbrksuccess,
gv_activ                       TYPE i,
gv_rcv                         TYPE i,
gv_snd                         TYPE i,
BEGIN OF ls_intval,
task                           TYPE numc4,
idxfr                          TYPE i,
idxto                          TYPE i,
activ                          TYPE flag,
fails                         TYPE i,
END OF ls_intval,
gt_intval                     LIKE TABLE OF ls_intval.
START-OF-SELECTION.
PERFORM paralleltest.

*---------------------------------------------------------------------*
*       CLASS task DEFINITION                                        *
*---------------------------------------------------------------------*

CLASS task DEFINITION.
PUBLIC SECTION.
CLASS-METHODS:
  provide
    RETURNING
    value(name)           TYPE numc4,
  return
    IMPORTING
    name                 TYPE numc4,
  initialize
    RETURNING
    value(group)         TYPE rzllitab-classname.
PRIVATE SECTION.
CLASS-DATA:
  gv_group                  TYPE rzllitab-classname,
  BEGIN OF ls_task,
    name                    TYPE numc4,
    used                    TYPE flag,
  END OF ls_task,
  gt_task                  LIKE TABLE OF ls_task.
ENDCLASS.  
***       CLASS itab IMPLEMENTATION ***
CLASS task IMPLEMENTATION.
METHOD initialize.
  DATA:
    lv_max                   TYPE i,
    lv_inc                   TYPE numc7,
    lv_free                  TYPE i.
  CLEAR gt_task.
  COMMIT WORK."start new LUW?
  SELECT classname
    INTO gv_group
    FROM rzllitab
    UP TO 1 ROWS
    WHERE grouptype = 'S'.
  ENDSSELECT.
  CALL FUNCTION 'SPBT_INITIALIZE'
    EXPORTING
      group_name       = gv_group
    IMPORTING
      max_pbt_wps     = lv_max
      free_pbt_wps    = lv_free
    EXCEPTIONS
      invalid_group_name             = 1
      internal_error                 = 2
      pbt_env_already_initialized    = 3
      currently_no_resources_avail   = 4
      no_pbt_resources_found         = 5
      cant_init_different_pbt_groups = 6
      others                        = 7.
  IF sy-subrc <> 0.
    MESSAGE ID sy-msgid
      TYPE sy-msgty
      NUMBER sy-msgno
    WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.
  SUBTRACT 2 FROM lv_free.
  IF lv_free >= 1.
    DO lv_free TIMES.
      ls_task-name = sy-index.
      APPEND ls_task TO gt_task.
    ENDDO.
  ELSE.
    MESSAGE e000(r1)
      WITH
      'Parallel processing canceled,'
  ENDCLASS.  
***       CLASS itab IMPLEMENTATION ***
CLASS task IMPLEMENTATION.
METHOD initialize.
  DATA:
    lv_max                   TYPE i,
    lv_inc                   TYPE numc7,
    lv_free                  TYPE i.
  CLEAR gt_task.
  COMMIT WORK."start new LUW?
  SELECT classname
    INTO gv_group
    FROM rzllitab
    UP TO 1 ROWS
    WHERE grouptype = 'S'.
  ENDSSELECT.
  CALL FUNCTION 'SPBT_INITIALIZE'
    EXPORTING
      group_name       = gv_group
    IMPORTING
      max_pbt_wps     = lv_max
      free_pbt_wps    = lv_free
    EXCEPTIONS
      invalid_group_name             = 1
      internal_error                 = 2
      pbt_env_already_initialized    = 3
      currently_no_resources_avail   = 4
      no_pbt_resources_found         = 5
      cant_init_different_pbt_groups = 6
      others                        = 7.
  IF sy-subrc <> 0.
    MESSAGE ID sy-msgid
      TYPE sy-msgty
      NUMBER sy-msgno
    WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.
  SUBTRACT 2 FROM lv_free.
  IF lv_free >= 1.
    DO lv_free TIMES.
      ls_task-name = sy-index.
      APPEND ls_task TO gt_task.
    ENDDO.
  ELSE.
    MESSAGE e000(r1)
      WITH
      'Parallel processing canceled,'
lv_free
'processes in group'
gv_group.
*
& & & &
ENDIF.
ENDMETHOD.  "initialize
METHOD provide.
FIELD-SYMBOLS:
  <task> LIKE ls_task.
IF  gv_group IS INITIAL.
MESSAGE e000(r1)
WITH 'Task group not initialized'.
ENDIF.
LOOP AT gt_task ASSIGNING <task>
  WHERE used IS initial.
  EXIT.
ENDLOOP.
CHECK sy-subrc = 0.
  <task>-used = 'X'.
  name = <task>-name.
ENDMETHOD.
METHOD return.
  LOOP AT gt_task INTO ls_task
  WHERE
    name = name
    AND used = 'X'.
  DELETE gt_task.
ENDLOOP.
  IF sy-subrc = 0.
    CLEAR ls_task-used.
    APPEND ls_task TO gt_task.
  ELSE.
  * fatal error
  ENDIF.
ENDMETHOD.
ENDCLASS.  "itab IMPLEMENTATION
*---------------------------------------------------------------------*
*&      Form  paralleltest
*---------------------------------------------------------------------*
FORM paralleltest.
DATA:
  ls_bapi_ref_doc_range TYPE bapi_ref_doc_range,
  lv_done TYPE flag,
  lv_group TYPE rzllitab-classname,
  lv_task TYPE numc4,
  lv_msd TYPE text255,
  lv_grid_title TYPE lvc_title,
  lv_tfill TYPE sytfill,
  lv_vbelv TYPE vbenv,
  lv_npacs TYPE i,
  lt_vbelv TYPE SORTED TABLE OF vbelv
  WITH UNIQUE KEY table_line,
  lv_mod TYPE i.
FIELD-SYMBOLS:
  <intval> LIKE LINE OF gt_intval.
*
build intervals
SELECT vbelv INTO lv_vbelv
  FROM vbfa.
INSERT lv_vbelv INTO TABLE lt_vbelv.
CHECK sy-subrc = 0.
ADD 1 TO lv_tfill.
CHECK:
  p_dbcnt > 0,
  lv_tfill >= p_dbcnt.
EXIT.
ENDSELECT.
DESCRIBE TABLE lt_vbelv LINES lv_tfill.
IF ( p_pacsz < p_dbcnt OR
  p_dbcnt = 0
) AND
  p_pacsz > 0.
  lv_npacs = lv_tfill DIV p_pacsz.
  lv_mod = lv_tfill MOD p_pacsz.
IF lv_mod <> 0.
    ADD 1 TO lv_npacs.
ENDIF.
DO lv_npacs TIMES.
    ls_intval-idxfr = ls_intval-idxto + 1.
    ls_intval-idxto = ls_intval-idxfr - 1 + p_pacsz.
    IF ls_intval-idxto > lv_tfill.
        ls_intval-idxto = lv_tfill.
    ENDIF.
    APPEND ls_intval TO gt_intval.
ENDDO.
ELSE.
    ls_intval-idxfr = 1.
    ls_intval-idxto = lv_tfill.
    APPEND ls_intval TO gt_intval.
ENDIF.
WHILE lv_done IS INITIAL.
    * find an interval to be processed
    LOOP AT gt_intval ASSIGNING <intval>
        WHERE activ = space
        AND fails BETWEEN 0 AND 5.
        EXIT.
    ENDDO.
    IF sy-subrc <> 0.
        * no inactive unprocessed interval. All complete or must wait?
        * check for intervals with unsuccessful tries
        LOOP AT gt_intval ASSIGNING <intval>
            WHERE fails BETWEEN 0 AND 5.
            EXIT.
        ENDDO.
        IF sy-subrc = 0.
            * wait until all started processes have been received.
            * Note: No receive is executed without WAIT
            WAIT UNTIL gv_activ IS INITIAL UP TO 600 SECONDS.
        ELSE.
            * all done
            lv_done = 'X'.
        ENDIF.
    ENDIF.
    * process interval if provided
    IF <intval> IS ASSIGNED.
        * init parallel processing
        lv_group = task=>initialize( ).
    ENDIF.
    * get unused task
    lv_task = task=>provide( ).
    CHECK lv_task IS INITIAL.
    * no unused task? wait for all started task are received
    WAIT UNTIL gv_activ IS INITIAL UP TO 600 SECONDS.
    ENDWHILE.
    * call if task assigned
    CHECK NOT lv_task IS INITIAL.
    * prepare function parameters
    ls_bapi_ref_doc_range = 'IBT'.
    READ TABLE lt_vbelv INTO ls_bapi_ref_doc_range-ref_doc_low INDEX <intval>-idxfr.
    READ TABLE lt_vbelv INTO ls_bapi_ref_doc_range-ref_doc_high INDEX <intval>-idxto.
    * mark interval as failed
    ADD 1 TO <intval>-fails.
    ADD 1 TO gv_snd.
    CALL FUNCTION gc_function
        STARTING NEW TASK lv_task
        DESTINATION IN GROUP lv_group
        PERFORMING bapi_receive ON END OF TASK
        EXPORTING
            refdocrange = ls_bapi_ref_doc_range
        EXCEPTIONS
            communication_failure = 1 MESSAGE lv_msg
            system_failure = 2 MESSAGE lv_msg
            RESOURCE_FAILURE = 3.
        IF sy-subrc = 0.
            <intval>-activ = 'X'.
            <intval>-task = lv_task.
ADD 1 TO gv_activ.
ELSE.
   CALL METHOD task=>return EXPORTING name = lv_task.
ENDIF.
CLEAR lv_task.
ENDIF.
ENDWHILE.
* wait for pending processes
MESSAGE s000(r1) WITH 'Wait for pending processes'.
WAIT UNTIL gv_activ IS INITIAL.
* report unfinished intervals
LOOP AT gt_intval ASSIGNING <intval>
   WHERE fails >= 0.
   READ TABLE lt_vbelv INTO la_bapi_ref_doc_range-ref_doc_low
      INDEX <intval>-idxfr.
   READ TABLE lt_vbelv INTO la_bapi_ref_doc_range-ref_doc_high
      INDEX <intval>-idxto.
   MESSAGE i000(r1) WITH 'unprocessed interval from'
   la_bapi_ref_doc_range-ref_doc_low
   'to'
   la_bapi_ref_doc_range-ref_doc_high.
ENDLOOP.
MESSAGE s000(r1) WITH 'start ALV'.
* transfer results to standard table
WRITE gv_rcv TO lv_grid_title LEFT-JUSTIFIED.
   lv_grid_title+40(1) = '+'.
WRITE gv_snd TO lv_grid_title+50 LEFT-JUSTIFIED.
REPLACE '+' WITH 'RCV/SND' INTO lv_grid_title.
CONDENSE lv_grid_title.
CALL FUNCTION 'REUSE_ALV_GRID_DISPLAY'
   EXPORTING
   i_structure_name = 'BAPIVBRKSUCCESS'
   i_grid_title     = lv_grid_title
   TABLES
   t_outtab         = gt_bapivbrksuccess.
ENDFORM.

*---------------------------------------------------------------------*
* &      Form  bapi_receive
*---------------------------------------------------------------------*
FORM bapi_receive USING pv_task TYPE any.
DATA:
   lv_task                               TYPE numc4,
   lt_bapivbrksuccess                    TYPE TABLE OF bapivbrksuccess,
   lv_msg                                TYPE text80,
   lv_subrc                              TYPE sy-subrc.
FIELD-SYMBOLS:
   <intval>                              LIKE LINE OF gt_intval.
CLEAR lt_bapivbrksuccess.
RECEIVE RESULTS FROM FUNCTION gc_function
   TABLES
   success = lt_bapivbrksuccess
   EXCEPTIONS
   communication_failure = 1 MESSAGE lv_msg
   system_failure = 2 MESSAGE lv_msg.
   lv_subrc = sy-subrc.
   lv_task = pv_task.
CALL METHOD task=>return EXPORTING name = lv_task.
LOOP AT gt_intval ASSIGNING <intval>
   WHERE task = lv_task
   AND fails <> -1.
   EXIT.
ENDLOOP.
IF lv_subrc <> 0.
   * fatal error
   MESSAGE e000(r1) WITH 'returned task' lv_task 'not in task table'.
ENDIF.
CLEAR <intval>-activ.
CASE lv_subrc.
   WHEN 0.
      <intval>-fails = -1.
APPEND LINES OF lt_bapivbrksuccess TO gt_bapivbrksuccess.
ADD 1 TO gv_rcv.
WHEN 1.
  ADD 1 TO <intval>-fails.
  WRITE: 'communication_failure for task', lv_task, lv_msg.
WHEN 2.
  WRITE: 'system_failure', lv_task, lv_msg.
  ADD 1 TO <intval>-fails.
ENDCASE.
SUBTRACT 1 FROM gv_activ.
ENDFORM.  
  " bapi_receive

Runtime errors in Parallel Processing:
COMMUNICATION_FAILURE: You will get this error when you try to release more than one task with same name.