UEXTERNAL/VEXTERNALDB subroutines

User subroutine that gives control to the user at key moments of the analysis

(V)UEXTERNALDB subroutine

UEXTERNALDB: User subroutine to manage user-defined external databases and calculate model-independent history information.

VEXTERNALDB: User subroutine that gives control to the user at key moments of the analysis so that data can be exchanged dynamically among Abaqus user subroutines and with external programs or files.

(V)UEXTERNALDB subroutine

From the ABAQUS documentation:

ABAQUS/standard

Overview

User subroutine **`UEXTERNALDB`**:

- is called once each at the beginning of the analysis, at the beginning of each increment, at the end of each increment, and at the end of the analysis (in addition, the user subroutine is also called once at the beginning of a restart analysis);
- can be used to communicate between other software and user subroutines within Abaqus/Standard;
- can be used to open external files needed for other user subroutines at the beginning of the analysis and to close those files at the end of the analysis;
- can be used to calculate or read history information at the beginning of each increment. This information can be written to user-defined COMMON block variables or external files for use during the analysis by other user subroutines; and
- can be used to write the current values of the user-calculated history information to external files.

ABAQUS/explicit

Overview

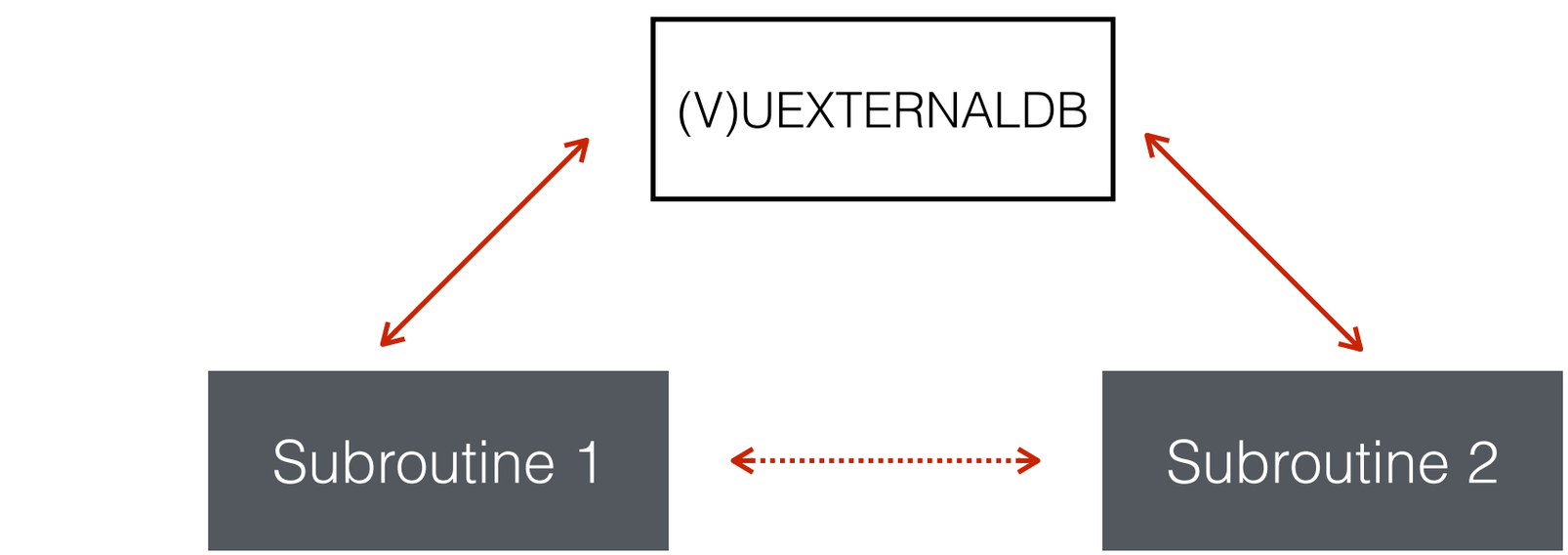
User subroutine **`VEXTERNALDB`**:

- is called once at the beginning of the analysis, at the beginning of each step, before each increment, at the start of each increment, at the end of each increment, at the end of each step, and finally at the end of the analysis;
- can be used to communicate data between external programs and user subroutines within Abaqus/Explicit;
- can be used to control the time incrementation of the Abaqus/Explicit analysis;
- can be used to control the output of the restart data for the analysis;
- can be used either to skip the remainder of an Abaqus step or to terminate the analysis;
- can be used to open and close external files as needed for exchange of data with the Abaqus analysis;
- can be used to exchange data with other user subroutines via user-allocated global and thread-local arrays (see “Allocatable arrays,” Section 2.1.23) and;
- can be used to exchange data with other Abaqus processes via an MPI mechanism (see “Obtaining parallel processes information,” Section 2.1.4) in domain-parallel analyses.

(V)UEXTERNALDB subroutine

(V)UEXTERNALDB subroutines can be used to:

- Access external databases/files,
- Pass informations through the memory,
- Make computations which requires access to all integration points at once.



User subroutine **`UEXTERNALDB`**:

- is called once each at the beginning of the analysis, at the beginning of each increment, at the end of each increment, and at the end of the analysis (in addition, the user subroutine is also called once at the beginning of a restart analysis);

User subroutine **`VEXTERNALDB`**:

- is called once at the beginning of the analysis, at the beginning of each step, before each increment, at the start of each increment, at the end of each increment, at the end of each step, and finally at the end of the analysis;

(V)UEXTERNALDB subroutine

```
 Possible values for the LOP argument
 parameter(j_int_StartAnalysis  = 0,
*          j_int_StartIncrement = 1,
*          j_int_EndIncrement   = 2,
*          j_int_EndAnalysis    = 3,
*          j_int_Restart        = 4,
*          j_int_StartStep      = 5,
*          j_int_EndStep        = 6)
```

Restart file

Time and timestep

Step number

Increment number

```
SUBROUTINE UEXTERNALDB(LOP,LRESTART,TIME,DTIME,KSTEP,KINC)
```

LOP variable indicates at which stage the EXTERNALDB subroutine is called.

```
SUBROUTINE VEXTERNALDB(LOP,I_ARRAY,NIARRAY,R_ARRAY,NRARRAY)
```

```
 Possible values for the lOp argument
 parameter(j_int_StartAnalysis  = 0,
*          j_int_StartStep      = 1,
*          j_int_SetupIncrement = 2,
*          j_int_StartIncrement = 3,
*          j_int_EndIncrement   = 4,
*          j_int_EndStep        = 5,
*          j_int_EndAnalysis    = 6 )
```

```
 dimension i_Array(niArray)
 Contents of i_Array
 parameter(i_int_nTotalNodes    = 1,
*          i_int_nTotalElements = 2,
*          i_int_kStep          = 3,
*          i_int_kInc           = 4,
*          i_int_iStatus        = 5,
*          i_int_lWriteRestart  = 6)
 Possible values for i_Array(i_int_iStatus)
 parameter(j_int_Continue          = 0,
*          j_int_TerminateStep     = 1,
*          j_int_TerminateAnalysis = 2)
```

```
 dimension r_Array(nrArray)
 Contents of r_Array
 parameter(i_flt_TotalTime = 1,
*          i_flt_StepTime  = 2,
*          i_flt_dTime     = 3)
```

(V)UEXTERNALDB subroutine

ABAQUS/standard

```
!-----------------------------------------------
!     Start of the analysis
!-----------------------------------------------
      if(LOP.eq.j_int_StartAnalysis)then
!-----------------------------------------------
!     Start of the increment
!-----------------------------------------------
      elseif(LOP.eq.j_int_StartIncrement)then
!-----------------------------------------------
!     End of the increment
!-----------------------------------------------
      elseif(LOP.eq.j_int_EndIncrement)then
!-----------------------------------------------
!     End of the Analysis
!-----------------------------------------------
      elseif(LOP.eq.j_int_EndAnalysis)then
      endif
!-----------------------------------------------
!     End of subroutine
!-----------------------------------------------
      RETURN
      END
```

ABAQUS/explicit

```
!-----------------------------------------------
!     Start of the analysis
!-----------------------------------------------
    if(lOp.eq.j_int_StartAnalysis)then
!-----------------------------------------------
!     Setup of the increment
!-----------------------------------------------
    elseif(lOp.eq.j_int_SetupIncrement)then
!-----------------------------------------------
!     Start of the increment
!-----------------------------------------------
    elseif(lOp.eq.j_int_StartIncrement)then
!-----------------------------------------------
!     End of the increment
!-----------------------------------------------
    elseif(lOp.eq.j_int_EndIncrement)then
!-----------------------------------------------
!     End of the analysis
!-----------------------------------------------
    elseif(lOp.eq.j_int_EndAnalysis)then
    endif
!-----------------------------------------------
!     End of the subroutine
!-----------------------------------------------
    return
    end
```

(V)UEXTERNALDB subroutine

In this example, we want to read the parameters an elasto-plastic model with a ductile fracture model from an additional text file.

V_UEXTERNALDB > example_UEXTERNALDB.k

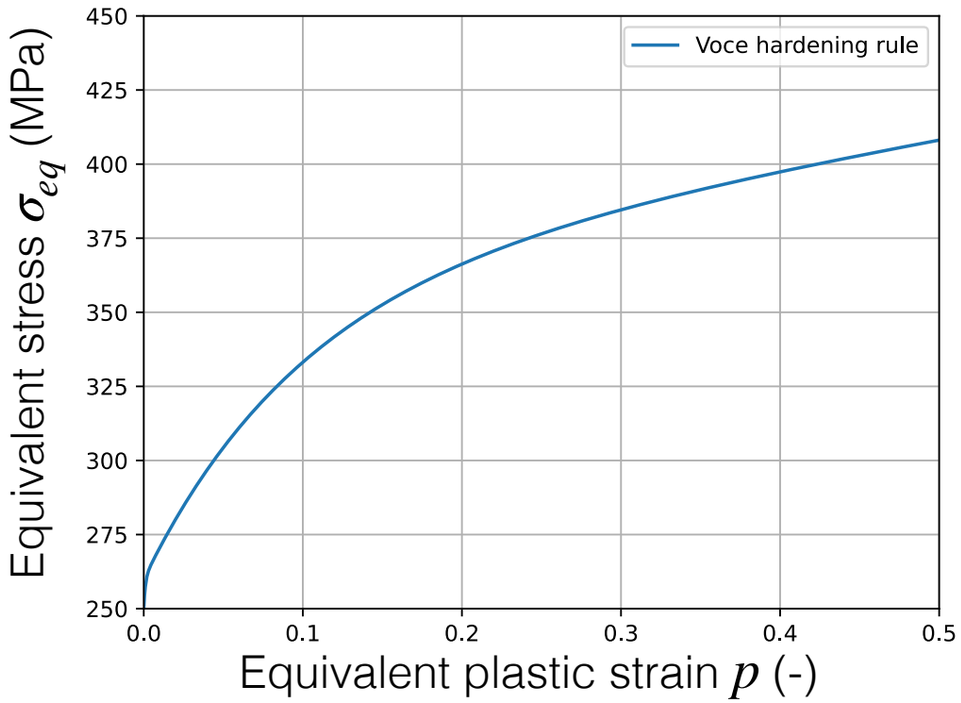
```
1   ** SIGMA0,      T1,    Q1,     T2,    Q2,    T3,     Q3, BLANK
2      250.0, 10000.0,  10.0, 1000.0, 100.0, 100.0, 1000.0,   0.0
3   **     WC,   BLANK, BLANK,  BLANK, BLANK, BLANK,  BLANK, BLANK
4      200.0,     0.0,   0.0,    0.0,   0.0,   0.0,    0.0,   0.0
```

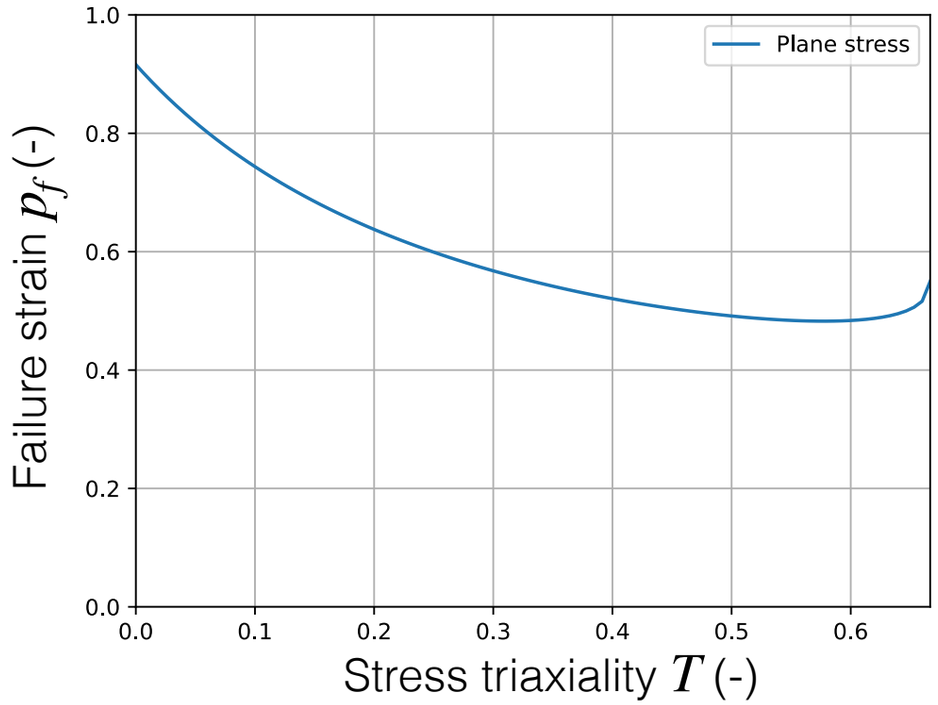
Isotropic hardening:

$$\sigma_y = \sigma_0 + \sum_{i=1}^{3} Q_i \left(1 - \exp\left(-\frac{\theta_i}{Q_i} p\right)\right)$$

Damage indicator model:

$$D = \int_0^{p_f} \frac{\langle \sigma_1 \rangle}{W_c} \dot{p} \leq D_c$$





(V)UEXTERNALDB subroutine

ABAQUS/standard

```
!-----------------------------------------------------------------
!     Start of the analysis
!-----------------------------------------------------------------
      if(LOP.eq.j_int_StartAnalysis)then
        IF((kInc.eq.0).and.(myThreadID.eq.0))THEN
!-----------------------------------------------------------------
!          CREATE GLOBAL ARRAYS
!-----------------------------------------------------------------
          ptr_REAL = SMAFloatArrayCreate(ID_REAL, 24, 0.0)
!         FETCH ABAQUS JOBNAME
          call GETJOBNAME(JOBNAME,LENJOBNAME)
!         FETCH ABAQUS JOBNAME
          call GETOUTDIR(OUTDIR,LENOUTDIR)
!         Try to open jobname.k for extra-parameters
          OPEN(unit=15,file=trim(OUTDIR)//'/'//trim(JOBNAME)//'.k',
     +     STATUS='OLD',IOSTAT=ios)
!         Read MPC parameters from the jobname.k file
          if(ios.gt.0)then
            WRITE(6,*) 'No *.k file submitted'
          else
           k = 0
           do while (ios == 0)
!             Read line
              READ(15,FMT='(A)',end=77) line
!             Read properties from line
              if(line(1:2).ne.'**')then
                  READ(line,*,end=78) (PROPS_REAL(k+i),i=1,8)
                  k = k+8
              endif
   78 CONTINUE
           enddo
   77 CONTINUE
           CLOSE(unit=15)
          endif
        ENDIF
!-----------------------------------------------------------------
```

ABAQUS/explicit

```
!-----------------------------------------------------------------
!     Start of the analysis
!-----------------------------------------------------------------
      if(lOp.eq.j_int_StartAnalysis)then
        IF(kInc.eq.0)THEN
!-----------------------------------------------------------------
!          CREATE GLOBAL ARRAYS
!-----------------------------------------------------------------
          ptr_REAL = SMAFloatArrayCreate(ID_REAL, 24, 0.0)
!         FETCH ABAQUS JOBNAME
          call VGETJOBNAME(JOBNAME,LENJOBNAME)
!         FETCH ABAQUS JOBNAME
          call VGETOUTDIR(OUTDIR,LENOUTDIR)
!         Try to open jobname.k for extra-parameters
          OPEN(unit=15,file=trim(OUTDIR)//'/'//trim(JOBNAME)//'.k',
     +     STATUS='OLD',IOSTAT=ios)
!         Read MPC parameters from the jobname.k file
          if(ios.gt.0)then
            WRITE(6,*) 'No *.k file submitted'
          else
           k = 0
           do while (ios == 0)
!             Read line
              READ(15,FMT='(A)',end=77) line
!             Read properties from line
              if(line(1:2).ne.'**')then
                  READ(line,*,end=78) (PROPS_REAL(k+i),i=1,8)
                  k = k+8
              endif
   78 CONTINUE
           enddo
   77 CONTINUE
           CLOSE(unit=15)
          endif
        ENDIF
!-----------------------------------------------------------------
```

(V)UEXTERNALDB subroutine

ABAQUS/standard

```
!------------------------------------------------------------------
!     Start of the analysis
!------------------------------------------------------------------
      if(LOP.eq.j_int_StartAnalysis)then
        IF((kInc.eq.0).and.(myThreadID.eq.0))THEN
!------------------------------------------------------------------
!          CREATE GLOBAL ARRAYS
!------------------------------------------------------------------
          ptr_REAL = SMAFloatArrayCreate(ID_REAL, 24, 0.0)
```

Pointer

Pointer ID (unique number)

Size

Initial value

ABAQUS/explicit

```
!------------------------------------------------------------------
!     Start of the analysis
!------------------------------------------------------------------
      if(lOp.eq.j_int_StartAnalysis)then
        IF(kInc.eq.0)THEN
!------------------------------------------------------------------
!          CREATE GLOBAL ARRAYS
!------------------------------------------------------------------
          ptr_REAL = SMAFloatArrayCreate(ID_REAL, 24, 0.0)
```

Pointer

Pointer ID (unique number)

Size

Initial value

1 Create float (real) array in memory

Define array size

Link pointer

```
!-----Declare global variables
      REAL*8 PROPS_REAL(24)          ! ARRAY WITH REAL PROPERTIES
      integer ID_REAL                ! ID for pointers
      parameter(ID_REAL = 1)         ! ID for pointers
      pointer(ptr_REAL, PROPS_REAL) ! pointer link
!------------------------------------------------------------------
```

Pointer ID

```
        ENDIF
!------------------------------------------------------------------
```

```
          ENDIF
!------------------------------------------------------------------
```

Interlude

When the size of the array is known before starting the analysis:

Set in the declaration

```
!-----Declare global variables
      REAL*8 PROPS_REAL(24)          ! ARRAY WITH REAL PROPERTIES
      integer ID_REAL                ! ID for pointers
      parameter(ID_REAL = 1)         ! ID for pointers
      pointer(ptr_REAL, PROPS_REAL) ! pointer link
!---------------------------------------------------------------
```

Set in array creation

```
!-----------------------------------------------------------------
!           CREATE GLOBAL ARRAYS
!-----------------------------------------------------------------
            ptr_REAL = SMAFloatArrayCreate(ID_REAL, 24, 0.0)
```

When the size of the array is <u>not</u> known before starting the analysis:

Indicate that it is a vector

```
!-----Declare global variables
      REAL*8 PROPS_REAL(1)          ! ARRAY WITH REAL PROPERTIES
      integer ID_REAL                ! ID for pointers
      parameter(ID_REAL = 1)         ! ID for pointers
      pointer(ptr_REAL, PROPS_REAL) ! pointer link
!---------------------------------------------------------------
```

Set size in array creation

```
!-----------------------------------------------------------------
!      CREATE GLOBAL ARRAYS
!-----------------------------------------------------------------
      ptr_REAL = SMAFloatArrayCreate(ID_REAL, 24, 0.0)
```

Use the same variable declaration in other subroutines where the global array is used.

Interlude

```
!-----Declare global variables
      REAL*8 PROPS_REAL(1)                ! ARRAY WITH REAL PROPERTIES
      integer PROPS_INT(1)                ! ARRAY WITH REAL PROPERTIES
      integer ID_REAL,ID_INT              ! ID for pointers
      parameter(ID_REAL = 1,ID_INT = 2) ! ID for pointers
      pointer(ptr_REAL, PROPS_REAL)       ! pointer link
      pointer(ptr_INT, PROPS_INT)         ! pointer link
!-----------------------------------------------------------------------
!     CREATE GLOBAL ARRAYS
!-----------------------------------------------------------------------
      ptr_REAL = SMAFloatArrayCreate(ID_REAL, 24, 0.0)
      ptr_INT  = SMAIntArrayCreate( ID_INT, 24, 0)
```

Create an integer array

There are no utility routines to create matrices, only vectors

A matrix can still be defined thanks to FORTRAN memory management

Interlude

Example of matrix in vector storage:

a 10 lines vector

```
      PROGRAM ARRAY_SIZE
      implicit none
      integer i
      integer array(10)
!-----------------------------
!     START PROGRAM
!-----------------------------
      do i=1,10
         array(i) = i
      enddo
      call TEST_ARRAY(array)
!-----------------------------
!     END PROGRAM
!-----------------------------
      end program ARRAY_SIZE
```

a 5 lines and 2 columns matrix

```
SUBROUTINE TEST_ARRAY(array)
integer array(5,2)
integer i
do i=1,5
   print*,'array', array(i,1), array(i,2)
enddo
RETURN
END
```

Output:

```
(base) davidmorin@davids-M1-MacBook-Pro ABAQUS_subroutines % ./test_array
 array           1           6
 array           2           7
 array           3           8
 array           4           9
 array           5          10
```

Interlude

Example of matrix in vector storage:

a 10 lines vector

```
      PROGRAM ARRAY_SIZE
      implicit none
      integer i
      integer array(10)
!-----------------------------
!     START PROGRAM
!-----------------------------
      do i=1,10
         array(i) = i
      enddo
      call TEST_ARRAY(array)
!-----------------------------
!     END PROGRAM
!-----------------------------
      end program ARRAY_SIZE
```

a 2 lines and 5 columns matrix

```
      SUBROUTINE TEST_ARRAY(array)
      integer array(2,5)
      integer i
      do i=1,2
         print*,'array', array(i,1), array(i,2),
     +   array(i,3), array(i,4), array(i,5)
      enddo
      RETURN
      END
```

Output:

```
(base) davidmorin@davids-M1-MacBook-Pro ABAQUS_subroutines % ./test_array
 array           1           3           5           7           9
 array           2           4           6           8          10
```

(V)UEXTERNALDB subroutine

ABAQUS/standard

```
!------------------------------------------------------------------
!     Start of the analysis
!------------------------------------------------------------------
      if(LOP.eq.j_int_StartAnalysis)then
        IF((kInc.eq.0).and.(myThreadID.eq.0))THEN
```

Name of the job — Length of string

```
!         FETCH ABAQUS JOBNAME
          call GETJOBNAME(JOBNAME,LENJOBNAME)
!         FETCH ABAQUS JOBNAME
          call GETOUTDIR(OUTDIR,LENOUTDIR)
```

Name of the directory — Length of string

```
        ENDIF
!------------------------------------------------------------------
```

ABAQUS/explicit

```
!------------------------------------------------------------------
!     Start of the analysis
!------------------------------------------------------------------
      if(lOp.eq.j_int_StartAnalysis)then
        IF(kInc.eq.0)THEN
```

Name of the job — Length of string

```
!         FETCH ABAQUS JOBNAME
          call VGETJOBNAME(JOBNAME,LENJOBNAME)
!         FETCH ABAQUS JOBNAME
          call VGETOUTDIR(OUTDIR,LENOUTDIR)
```

Name of the directory — Length of string

```
        ENDIF
!------------------------------------------------------------------
```

2. Get Jobname*
3. Get ABAQUS job directory*

```
integer LENJOBNAME,LENOUTDIR
character*256 JOBNAME,OUTDIR
```

* the utility routines are slightly different between ABAQUS/Standard and ABAQUS/Explicit

(V)UEXTERNALDB subroutine

ABAQUS/standard

```
!-----------------------------------------------------------------
!     Start of the analysis
!-----------------------------------------------------------------
      if(LOP.eq.j_int_StartAnalysis)then
        IF((kInc.eq.0).and.(myThreadID.eq.0))THEN
!-----------------------------------------------------------------
!           CREATE GLOBAL ARRAYS
!-----------------------------------------------------------------
            ptr_REAL = SMAFloatArrayCreate(ID_REAL, 24, 0.0)
!           FETCH ABAQUS JOBNAME
            call GETJOBNAME(JOBNAME,LENJOBNAME)
!           FETCH ABAQUS JOBNAME
            call GETOUTDIR(OUTDIR,LENOUTDIR)
!           Try to open jobname.k for extra-parameters
            OPEN(unit=15,file=trim(OUTDIR)//'/'//trim(JOBNAME)//'.k',
     +      STATUS='OLD',IOSTAT=ios)
```

ABAQUS/explicit

```
!-----------------------------------------------------------------
!     Start of the analysis
!-----------------------------------------------------------------
      if(lOp.eq.j_int_StartAnalysis)then
        IF(kInc.eq.0)THEN
!-----------------------------------------------------------------
!           CREATE GLOBAL ARRAYS
!-----------------------------------------------------------------
            ptr_REAL = SMAFloatArrayCreate(ID_REAL, 24, 0.0)
!           FETCH ABAQUS JOBNAME
            call VGETJOBNAME(JOBNAME,LENJOBNAME)
!           FETCH ABAQUS JOBNAME
            call VGETOUTDIR(OUTDIR,LENOUTDIR)
!           Try to open jobname.k for extra-parameters
            OPEN(unit=15,file=trim(OUTDIR)//'/'//trim(JOBNAME)//'.k',
     +      STATUS='OLD',IOSTAT=ios)
```

4 Combine directory and job name
+ the file extension (avoid ABAQUS extension)

```
      ENDIF
!-----------------------------------------------------------------
```

```
      ENDIF
!-----------------------------------------------------------------
```

(V)UEXTERNALDB subroutine

ABAQUS/standard

```
!-----------------------------------------------------------------
!     Start of the analysis
!-----------------------------------------------------------------
      if(LOP.eq.j_int_StartAnalysis)then
        IF((kInc.eq.0).and.(myThreadID.eq.0))THEN
!-----------------------------------------------------------------
!           CREATE GLOBAL ARRAYS
!-----------------------------------------------------------------
            ptr_REAL = SMAFloatArrayCreate(ID_REAL, 24, 0.0)
!           FETCH ABAQUS JOBNAME
            call GETJOBNAME(JOBNAME,LENJOBNAME)
!           FETCH ABAQUS JOBNAME
            call GETOUTDIR(OUTDIR,LENOUTDIR)
!           Try to open jobname.k for extra-parameters
            OPEN(unit=15,file=trim(OUTDIR)//'/'//trim(JOBNAME)//'.k',
     +      STATUS='OLD',IOSTAT=ios)
!           Read MPC parameters from the jobname.k file
            if(ios.gt.0)then
              WRITE(6,*) 'No *.k file submitted'
            else
             k = 0
             do while (ios == 0)
!               Read line
                READ(15,FMT='(A)',end=77) line
!               Read properties from line
                if(line(1:2).ne.'**')then
                    READ(line,*,end=78) (PROPS_REAL(k+i),i=1,8)
                    k = k+8
                endif
 78 CONTINUE
             enddo
 77 CONTINUE
             CLOSE(unit=15)
            endif
        ENDIF
!-----------------------------------------------------------------
```

Read text file

ABAQUS/explicit

```
!-----------------------------------------------------------------
!     Start of the analysis
!-----------------------------------------------------------------
      if(lOp.eq.j_int_StartAnalysis)then
        IF(kInc.eq.0)THEN
!-----------------------------------------------------------------
!           CREATE GLOBAL ARRAYS
!-----------------------------------------------------------------
            ptr_REAL = SMAFloatArrayCreate(ID_REAL, 24, 0.0)
!           FETCH ABAQUS JOBNAME
            call VGETJOBNAME(JOBNAME,LENJOBNAME)
!           FETCH ABAQUS JOBNAME
            call VGETOUTDIR(OUTDIR,LENOUTDIR)
!           Try to open jobname.k for extra-parameters
            OPEN(unit=15,file=trim(OUTDIR)//'/'//trim(JOBNAME)//'.k',
     +      STATUS='OLD',IOSTAT=ios)
!           Read MPC parameters from the jobname.k file
            if(ios.gt.0)then
              WRITE(6,*) 'No *.k file submitted'
            else
             k = 0
             do while (ios == 0)
!               Read line
                READ(15,FMT='(A)',end=77) line
!               Read properties from line
                if(line(1:2).ne.'**')then
                    READ(line,*,end=78) (PROPS_REAL(k+i),i=1,8)
                    k = k+8
                endif
 78 CONTINUE
             enddo
 77 CONTINUE
             CLOSE(unit=15)
            endif
        ENDIF
!-----------------------------------------------------------------
```

(V)UEXTERNALDB subroutine

ABAQUS/standard

UHARD subroutine

```
!-----------------------------------------------------------------
!-----Declaration material parameters
!-----------------------------------------------------------------
      real*8 SIGMA0,T1,Q1,T2,Q2,T3,Q3
!-----------------------------------------------------------------
!-----Declaration internal variables
!-----------------------------------------------------------------
      real*8 T1oQ1,T2oQ2,T3oQ3
!-----Declare global variables
      REAL*8 PROPS_REAL(24)          ! ARRAY WITH REAL PROPERTIES
      integer ID_REAL                ! ID for pointers
      parameter(ID_REAL = 1)         ! ID for pointers
      pointer(ptr_REAL, PROPS_REAL) ! pointer link
!-----------------------------------------------------------------
!     Beginning of subroutine
!-----------------------------------------------------------------
c     Access array in the memory
      ptr_REAL = SMAFloatArrayAccess(ID_REAL)
c     Isotropic work-hardening
      SIGMA0 = PROPS_REAL(1)
      T1     = PROPS_REAL(2)
      Q1     = PROPS_REAL(3)
      T2     = PROPS_REAL(4)
      Q2     = PROPS_REAL(5)
      T3     = PROPS_REAL(6)
      Q3     = PROPS_REAL(7)
!-----------------------------------------------------------------
```

Declare variables

Access memory

ABAQUS/explicit

VUHARD subroutine

```
!-----------------------------------------------------------------
!-----Declaration material parameters
!-----------------------------------------------------------------
      real*8 SIGMA0,T1,Q1,T2,Q2,T3,Q3
!-----------------------------------------------------------------
!-----Declaration internal variables
!-----------------------------------------------------------------
      integer i
      real*8 T1oQ1,T2oQ2,T3oQ3
!-----Declare global variables
      REAL*8 PROPS_REAL(24)          ! ARRAY WITH REAL PROPERTIES
      integer ID_REAL                ! ID for pointers
      parameter(ID_REAL = 1)         ! ID for pointers
      pointer(ptr_REAL, PROPS_REAL) ! pointer link
!-----------------------------------------------------------------
!     Read material properties
!-----------------------------------------------------------------
      if((steptime.eq.totaltime).and.(steptime.eq.zero))then
         SIGMA0 = 1e12
         T1     = 0.0
         Q1     = 0.0
         T2     = 0.0
         Q2     = 0.0
         T3     = 0.0
         Q3     = 0.0
      else
c     Access array in the memory
         ptr_REAL = SMAFloatArrayAccess(ID_REAL)
c     Isotropic work-hardening
         SIGMA0 = PROPS_REAL(1)
         T1     = PROPS_REAL(2)
         Q1     = PROPS_REAL(3)
         T2     = PROPS_REAL(4)
         Q2     = PROPS_REAL(5)
         T3     = PROPS_REAL(6)
         Q3     = PROPS_REAL(7)
      endif
!-----------------------------------------------------------------
```

(V)UEXTERNALDB subroutine

ABAQUS/standard — USDFLD subroutine

ABAQUS/explicit — VUSDFLD subroutine

Declare variables

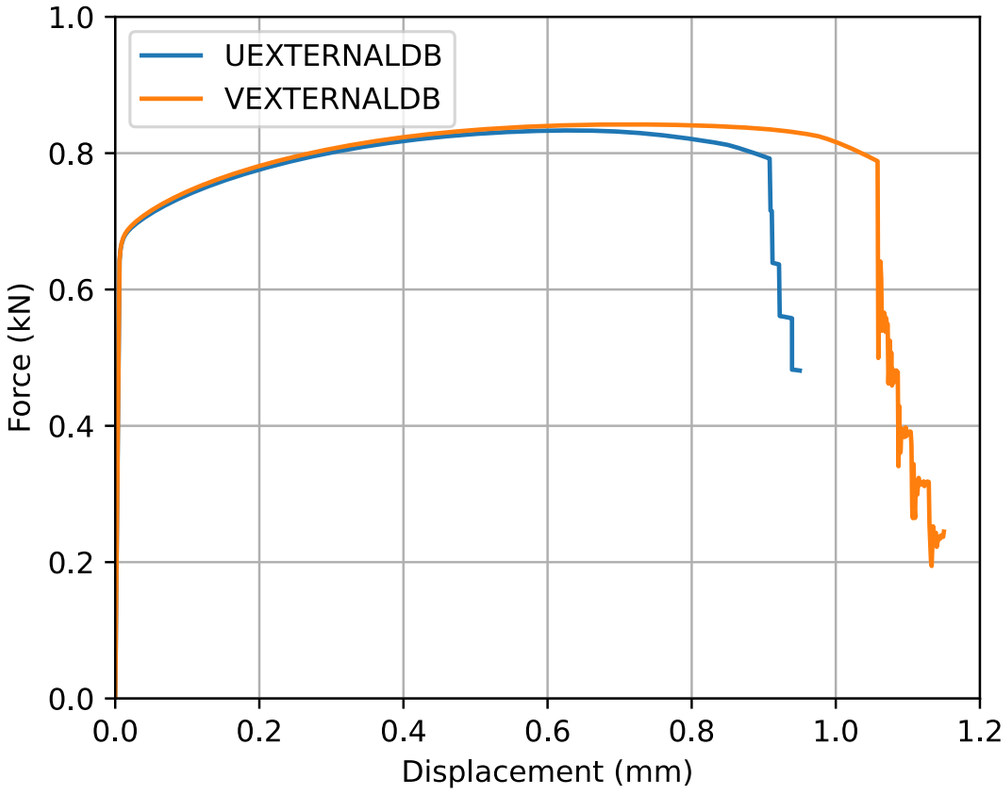
```
!-----------------------------------------------------------------
!-----Declaration material parameters
!-----------------------------------------------------------------
      real*8 Wc
!-----Declare global variables
      REAL*8 PROPS_REAL(24)          ! ARRAY WITH REAL PROPERTIES
      integer ID_REAL                ! ID for pointers
      parameter(ID_REAL = 1)         ! ID for pointers
      pointer(ptr_REAL, PROPS_REAL) ! pointer link
!-----------------------------------------------------------------
!     Read material properties
!-----------------------------------------------------------------
c     Access array in the memory
      ptr_REAL = SMAFloatArrayAccess(ID_REAL)
c     Fracture parameter
      Wc = PROPS_REAL(9)
!-----------------------------------------------------------------
```

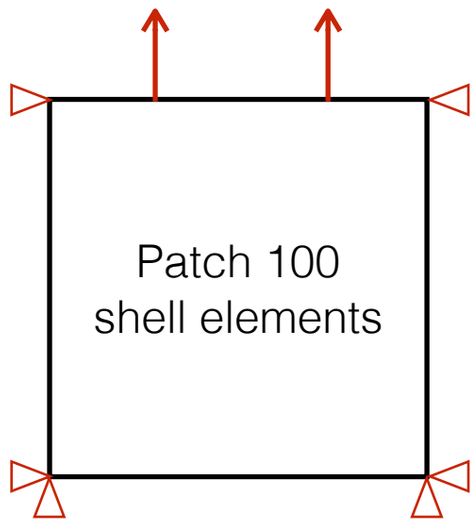
```
!-----------------------------------------------------------------
!-----Declaration material parameters
!-----------------------------------------------------------------
      real*8 Wc
!-----Declare global variables
      REAL*8 PROPS_REAL(24)          ! ARRAY WITH REAL PROPERTIES
      integer ID_REAL                ! ID for pointers
      parameter(ID_REAL = 1)         ! ID for pointers
      pointer(ptr_REAL, PROPS_REAL) ! pointer link
!-----------------------------------------------------------------
!     Read material properties
!-----------------------------------------------------------------
c     Access array in the memory
      ptr_REAL = SMAFloatArrayAccess(ID_REAL)
c     Fracture parameter
      Wc = PROPS_REAL(9)
!-----------------------------------------------------------------
```

Access memory

(V)UEXTERNALDB subroutine

Results from the UEXTERNALDB and VEXTERNALDB subroutines





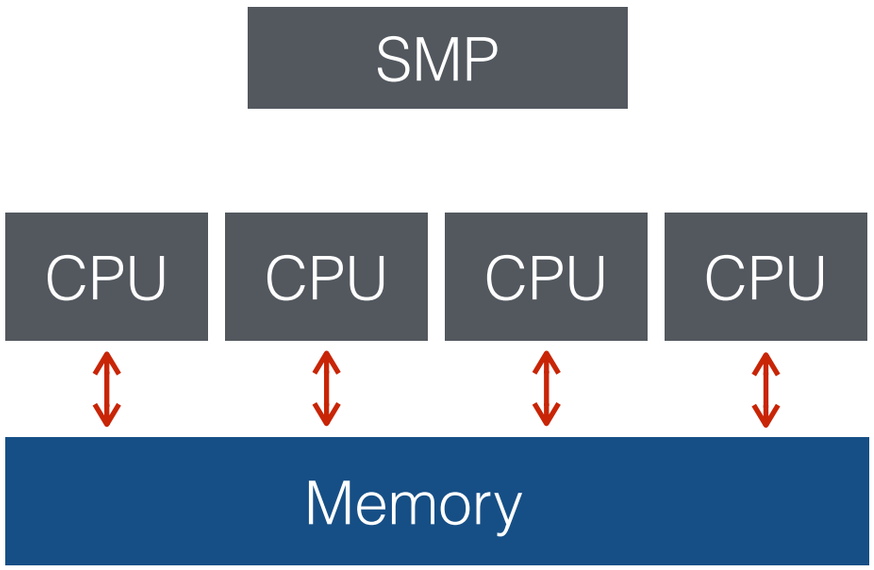
Interlude

SMP: Symmetric Multi Processing — Threads

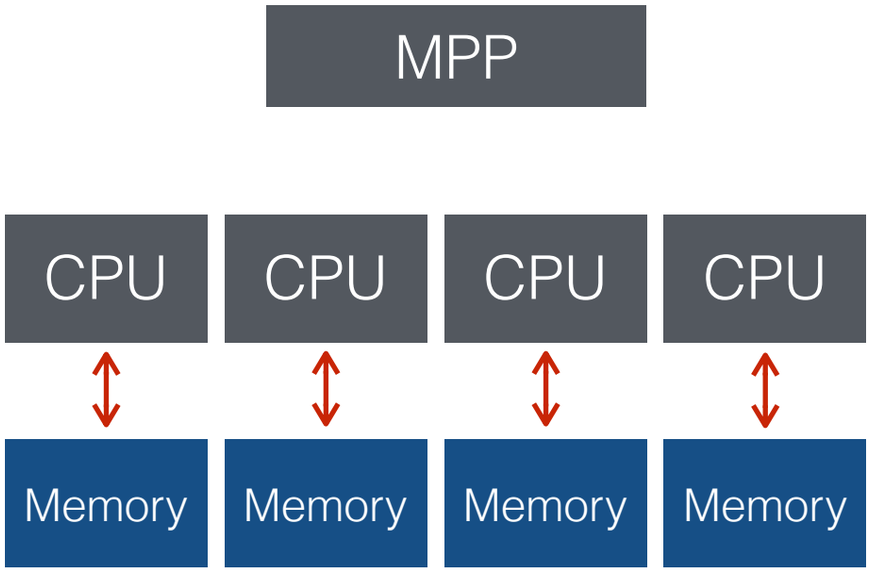
MPP: Massive Multi Processing — MPI

In ABAQUS:

- ABAQUS/Standard: SMP only
- ABAQUS/Explicit: MPP by default, SMP possible

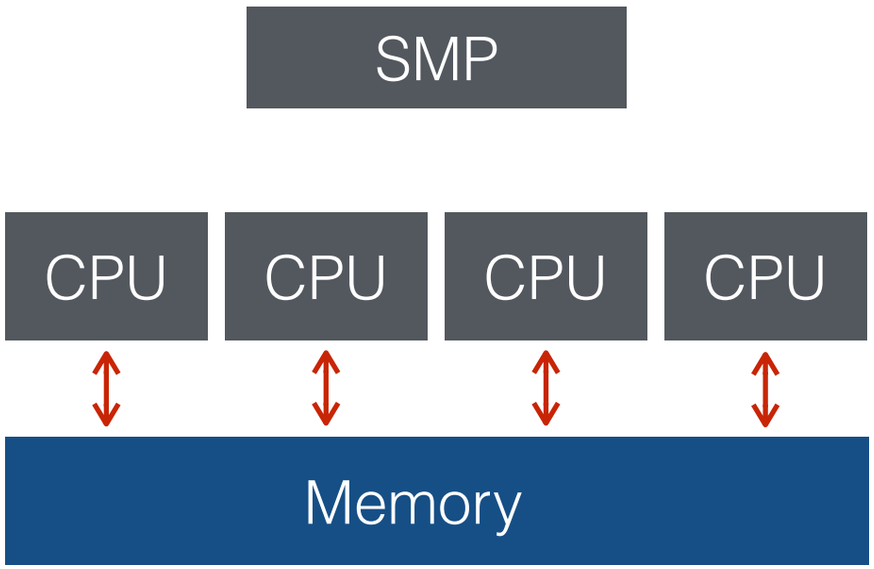


CPUs share the same memory

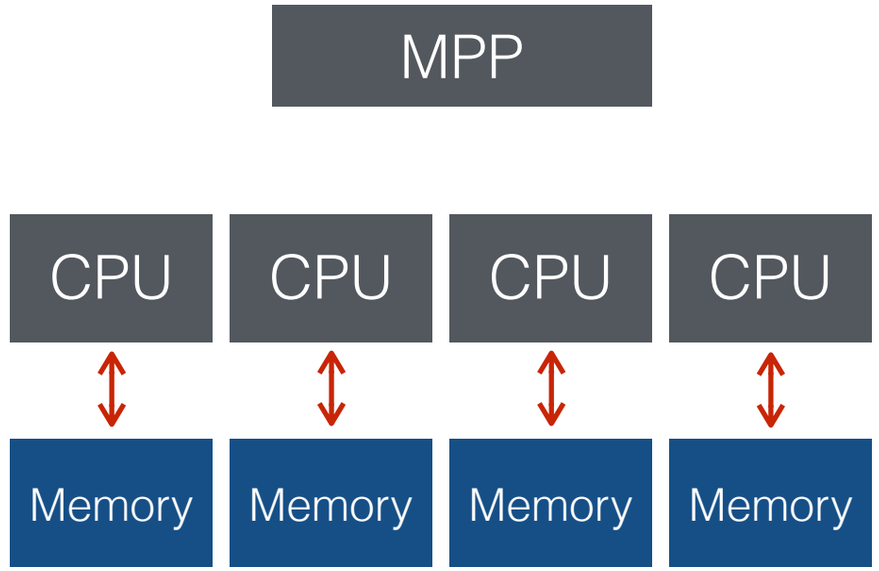


CPUs have their own memory

Interlude



CPUs share the same memory



CPUs have their own memory

CPUs have access to the same arrays	CPUs write in different arrays
CPUs have access to the same arrays	CPUs write in different arrays

Interlude

Should your code share the same arrays (for example non-local computations)?

	YES	NO

SMP	It will natively do it, global arrays will work,
	You still need to prevent the code to write at the same place, check UEXTERNALDB.f for example.
	`!------------------------------`
	`!  Get threadID`
	`!------------------------------`
	`myThreadID = get_thread_id()`
	`IF((kInc.eq.0).and.(myThreadID.eq.0))THEN`
	Filter out threads
		You need to define local arrays via:
		`!-----------------------------------------------------`
		`!  CREATE LOCAL ARRAYS`
		`!-----------------------------------------------------`
		`ptr_REAL = SMALocalFloatArrayCreate(ID_REAL, 24, 0.0)`
		`ptr_INT  = SMALocalIntArrayCreate( ID_INT, 24, 0)`
		Local arrays

MPP	You need to use MPI (Message Passing Interface) commands
	
	Example:
	`call MPI_Bcast( N,size, MPI_INT,0, ABA_COMM_WORLD,ierr)`
	`call MPI_Bcast(W,size,MPI_DOUBLE_PRECISION,0, ABA_COMM_WORLD,ierr)`
		It will natively do it, global arrays will work