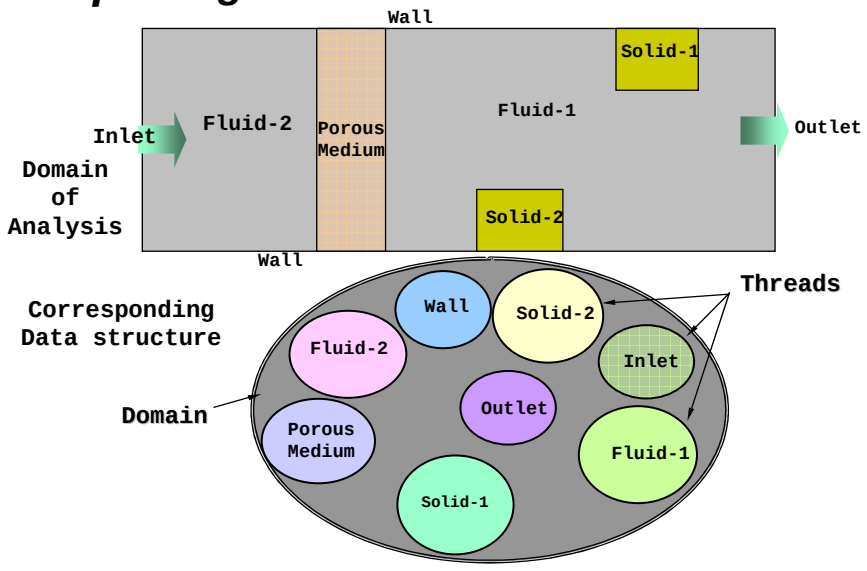
UDFs for Multiphase Flows

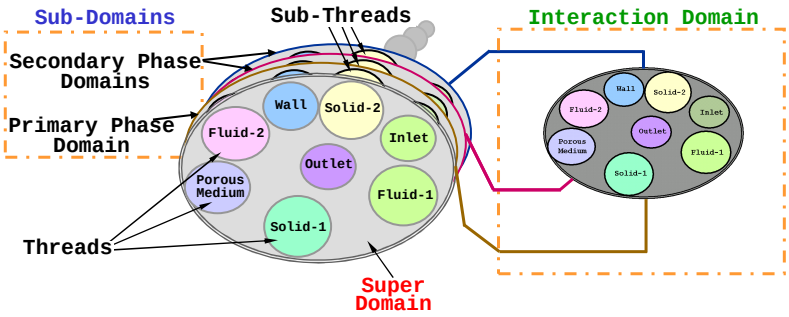
Advanced UDF Modeling Course

Recap: Single Phase Data Structure



Data Structure for Multiphase Models

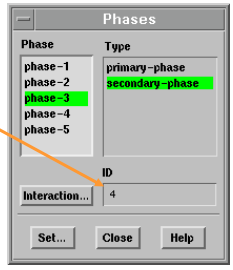
- Data Structure in multiphase models involve multiple domains:
  - Super Domain: This is the top-level domain contains all phase-independent and **mixture** data: geometry, connectivity, property
  - Sub-domains (phase domains): Each phase has a sub-domain that inherits the mixture-specific data and maintains the phase-specific data
  - Interaction Domain: To activate the phase interaction mechanisms



The Threads in Multi-Domains

- The mixture:
  - In single-phase, a mixture represents the sum over all the species
  - In multiphase it represents the sum over all the phases
- This distinction is important
  - Also, the code will later be extended to multiphase multi-component fluids (where, for example, *a phase could be a mixture of species*)
- Thread data structures:
  - Threads must be associated with the super domain and all sub-domains
    - For each cell (or face) thread of the super domain, there is a corresponding cell (or face) thread for each sub-domain
    - Some of the information defined in one thread of the super domain is shared with the corresponding threads of each of the sub-domains
    - Threads associated with the super domain are referred to as super-threads
    - Threads associated with the subdomain are referred to as phase-level threads or sub-threads

The Domain-Ids and The Thread-Ids



- For multiphase models, the domains need to be identified by unique Ids (including Interaction domain)
- **`Domain_ID`** of the super (mixture) domain is always '**`1`**'
- **`Domain_ID`**s are not necessarily ordered sequentially
- Therefore, to access the phase domains, each phase also has a **`phase_domain_index`**:
  - '**`0`**' for the primary phase
  - '**`N-1`**' for the last secondary phase
  - **`phase_domain_index`** is used in UDFs to retrieve phase thread pointers
    - Useful when you want to access data for another phase from an UDF for a particular phase

Domain Looping Macro

- **`sub_domain_loop(subdomain, mixture_domain, phase_index)`**
  - Loops over all phases (sub-domains) in a mixture
  - **`mixture_domain`** is already available
  - **`subdomain`**, **`phase_index`** are defined locally, initialized within the macro
- An Example for the loop, **`Domain_ID`** and the **`phase_domain_index`**:

```
DEFINE_ADJUST(print_id, mix_domain)
   {
   Domain *s_d;/*subdomain pointer, locally defined*/
   int p_d; /* loop counter for phase_domain_index, locally defined*/
   int p_d_id; /* mix_domain is available*/
   sub_domain_loop(s_d, mix_domain, p_d)
    {
      p_d_id = DOMAIN_ID(s_d);
      Message("the phase domain id = %d and the phase
               domain index = %d\n", p_d_id, p_d);
    }
   }
```

Thread Looping Macro

- **sub_thread_loop(subthread,mixture_thread,phase_index)**
  - Loops over all threads in a mixture
  - **mixture_thread** is already available
  - **subthread** & **phase_index** are defined locally, initialized within the macro

An Example:

```
/*compute bulk density of mixture and store it in a UDM*/
DEFINE_ADJUST(calc_den, mix_domain)
{ Thread *mix_thread;
  thread_loop_c(mix_thread,mix_domain)
  { cell_t c;
begin_c_loop(c,mix_thread)
    { Thread *s_t;
      int p_d_i;
      C_UDMI(c,mix_thread,0) = 0.;
      sub_thread_loop(s_t, mix_thread, p_d_i)
      C_UDMI(c,mix_thread,0) += C_VOF(c,s_t)*C_R(c,s_t);
    end_c_loop(c,mix_thread)
    }}}
```


Other Looping Macros

- **mp_thread_loop_c(cell_thread, mixture_domain, pt)**
  - **cell_thread** is a pointer to mixture thread in the **mixture_domain**
  - **mixture_domain** is already assumed to be available
  - **pt** is an array of thread pointers
- **mp_thread_loop_f(face_threads, mixture_domain, pt)**
  - **face_threads** is a pointer to face thread in the **mixture_domain**
  - **mixture_domain** is already assumed to be available
  - **pt** is an array of thread pointers pointing to the phase-level threads

An Example:

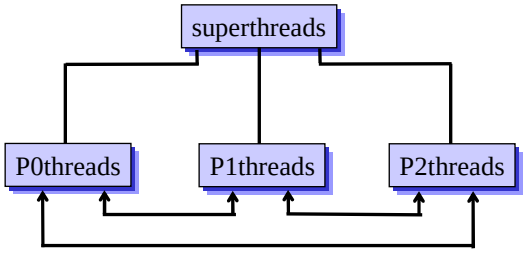
```
DEFINE_ADJUST(print_vof, mix_domain)
{ Thread *mix_thread;
  Thread **pt;
  mp_thread_loop_c(mix_thread, mix_domain, pt)
  {cell_t c;
   begin_c_loop(c, mix_thread)
       Message("cell volume fraction = %f\n",C_VOF(c,pt[0]));
   end_c_loop(c, mix_thread)}}
```

Access the Right Thread / Domain

- While writing UDF's, it is important that the right thread / domain is accessed
  - **`C_R(cell,thread)`** will return
    - The mixture density if **`thread`** is the mixture thread or
    - The phase densities if it is the **`phase thread`**
- In general the type of **`'DEFINE'`** macro determines which thread or domain (mixture or phase) gets passed to your UDF
  - **`DEFINE_INIT`** and **`DEFINE_ADJUST`** functions always get passed the domain structure associated with the super domain
  - **`DEFINE_ON_DEMAND`** functions are not passed any domain structures
  - If your UDF is not explicitly passed the pointer to the thread or domain required, then you can use a multiphase-specific utility macro to retrieve it

Superthreads and Phasethreads

- Each Thread is also in a hierarchy that matches that of the domains
- The "**superthreads**" are where the "mixture" of the phases is stored and so are often called the **"mixture threads"**
- Shared values such as the cell's geometry data are stored in the superthread
- Each Phase has its own set of threads known as a **"subthreads"** or "**phase threads**"



Phase Subthreads

Phase Thread Interactions

Access Variables External to a UDF

- `Get_Domain(Domain-ID)`
  - Usage: `Domain *domain=Get_Domain(n);`

    '**n**' is the **`Domain-ID`**, as appear in Define-Phase GUI. It is always '1' for the mixture domain
- `DOMAIN_ID(domain)`
  - Usage: `int domain_id = Domain_ID(subdomain);`

    '**`subdomain`**' is the pointer to a phase-level domain; **`domain_id`** upon return is the same integer ID displayed in the GUI under Define-Phases panel
- `DOMAIN_SUB_DOMAIN(mixture_domain,ph_domain_index)`
  - Usage:
    ```
    Domain *mixture_domain;
    Domain *subdomain = DOMAIN_SUB_DOMAIN
           (mixture_domain, phase_domain_index);
    ```
    returns the phase pointer subdomain for the **`phase_domain_index`**

Access Variables External to a UDF (2)

- `THREAD_SUB_THREAD(mixture_thread,ph_domain_index)`
  - Usage:
    ```
    int ph_d_index = 0; /* primary phase index is 0 */
    Thread *mix_th;/* mixture-level thread pointer */
    Thread *subth=THREAD_SUB_THREAD(mix_th, ph_d_index);
    ```
    returns the phase-level thread pointer for the given **`ph_d_index`**
- `THREAD_SUB_THREADS(mixture_thread)`
  - Usage:
    ```
    Thread *mixture_thread;
    Thread **pt; /* initialize pt: pointer array */
    pt = THREAD_SUB_THREADS(mixture_thread)
    ```
    returns the pointer array, **`pt`**, whose elements contain pointers to phase-level threads (subthreads)

Access Variables External to a UDF (3)

- **THREAD_SUPER_THREAD(subthread)**
  - Usage:

```
Thread *subthread; /*pointer to a phase thread within the mixture*/
Thread *mix_thread=THREAD_SUPER_THREAD(subthread)
```

    Given a phase thread pointer, it returns the super-thread (mixture-thread) pointer
- **DOMAIN_SUPER_DOMAIN(subdomain)**
  - Usage:

```
Domain *subdomain; /*pointer to a phase domain within the mixture*/
Domain *mixture_domain = DOMAIN_SUPER_DOMAIN(subdomain)
```

    It returns the mixture domain pointer

Access Variables External to a UDF (4)

- **PHASE_DOMAIN_INDEX(subdomain)**
  - Usage:

```
Domain *subdomain; /*points to a phase domain within the mixture*/
int phase_domain_index = PHASE_DOMAIN_INDEX(subdomain)
```

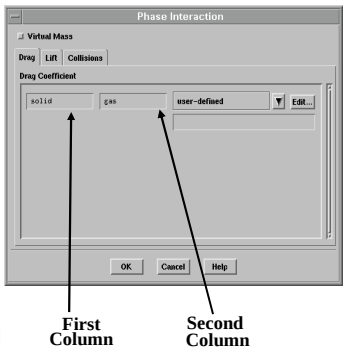
  returns the phase domain index for the phase domain (subdomain) pointer;
  It is an integer that starts with '0' for the primary phase and is incremented by one for each secondary phase

- **THREAD_DOMAIN(thread)**
  - Usage:

```
Thread *subthread; /*points to a phase thread within the mixture*/
Thread *mix_thread=THREAD_SUPER_THREAD(subthread)
Domain *subd=THREAD_DOMAIN(subthread); /*points to a phase domain*/
Domain *mixd=THREAD_DOMAIN(subthread); /*points to mixture domain*/
```

    returns domain pointer for the **thread**

Exchange Macros(1)



- **`DEFINE_EXCHANGE_PROPERTY ( name, c, mixture_thread, second_column_phase_index, first_column_phase_index)`**
  - This macro is used to specify custom drag & lift coefficients for the Eulerian multiphase model
  - **`mixture_thread`** points to the mixture thread
  - **`c`** is the index of a cell on the **`mixture_thread`**
  - **`first_column_phase_index`** and **`second_column_phase_index`** are integer identifiers corresponding to the pair of phases in your multiphase flow
  - The identifiers correspond to the phases that are selected in the Phase-Interaction panel in the GUI
  - The UDF returns the real value of the lift or drag coefficient

Exchange Macros(2)

- **`DEFINE_VECTOR_EXCHANGE_PROPERTY(name, c,mixture_thread, second_column_phase_index,first_column_phase_index, vector_result)`**
  - This macro is used to specify custom slip velocities for multiphase Mixture model
  - **`mixture_thread`** points to the mixture thread
  - **`c`** is the index of a cell on the **`mixture_thread`**
  - **`first_column_phase_index`** and **`second_column_phase_index`** are integer identifiers corresponding to the pair of phases in your multiphase flow
  - The identifiers correspond to the phases that are selected in the **`Phase-Interaction`** panel in the GUI
  - The UDF is passed the real pointer to the slip velocity vector **`vector_result`**, and it will need to set the components of the slip velocity vector

Exchange Macros(3)

An Example:

```
#include "udf.h"
#include "sg_mphase.h"
DEFINE_VECTOR_EXCHANGE_PROPERTY(custom_slip, c, mixture_thread,
second_column_phase_index, first_column_phase_index, vector_result)
{  real grav[2] = {0., -9.81};
   real K = 5.e4;
   real pgrad_x, pgrad_y;
   Thread *pt, *st;/* thread pointers for primary & secondary phases*/

   pt = THREAD_SUB_THREAD(mixture_thread, second_column_phase_index);
   st = THREAD_SUB_THREAD(mixture_thread, first_column_phase_index);
/* Now the threads are known for primary (0) & secondary(1) phases */
   pgrad_x = C_DP(c, mixture_thread)[0];
   pgrad_y = C_DP(c, mixture_thread)[1];
  vector_result[0] = -(pgrad_x/K)+(((C_R(c,st)-C_R(c,pt))/K)*grav[0]);
  vector_result[1] = -(pgrad_y/K)+(((C_R(c,st)-C_R(c,pt))/K)*grav[1]);
}
```


Cavitation Macros

- **`DEFINE_CAVITATION_RATE (name,c,t,p,rhoV,rhoL,vofV, p_v,n_b,mdot)`**
- You can use this macro to model the creation of vapor due to pressure tension in a multiphase flow
  - It is applied in the **`User-Defined-Function-Hooks`**→**`Cavitation-Mass-Rate-Function`** panel
  - **`t`** is a pointer to the mixture-level thread
  - **`c`** is the index of a cell on the thread pointed to by **`t`**
  - The remaining arguments are real pointers to the following data:
    - Shared pressure (**`p`**), vapor density (**`rhoV`**), liquid density (**`rhoL`**), vapor volume fraction (**`vofV`**), vaporization pressure (**`p_v`**), number of bubbles per unit volume (**`n_b`**), and rate of vapor formation (**`mdot`**)
  - The UDF sets the value referenced by the real pointer **`mdot`**, to the cavitation rate

Miscellaneous: Multiphase Macros

- Phase diameter
  - `C_PHASE_DIAMETER(c,phase_thread)`
- Phase Volume-fraction
  - `C_VOF(c,phase_thread)`
- Phase velocity gradients
  - `C_U_G(c, phase_thread)`
  - `C_V_G(c, phase_thread)`
  - `C_W_G(c, phase_thread)`

Miscellaneous : Multiphase Macros

- Phase volume fraction gradients: `C_VOF_G(c,phase_thread)`
  - Memory needs to be allocated and gradients need to be explicitly calculated

```
Domain   *pDomain = DOMAIN_SUB_DOMAIN(domain,P_PHASE);
Alloc_Storage_Vars   (pDomain,SV_VOF_RG,SV_VOF_G,SV_NULL);
Scalar_Reconstruction(pDomain,SV_VOF,-1,SV_VOF_RG,  NULL);
Scalar_Derivatives   (pDomain,SV_VOF,-1,SV_VOF_G,SV_VOF_RG,
                      Vof_Deriv_Accumulate);
```

Miscellaneous: Multiphase Macros

- Check if a given thread is a “super” or “sub” thread

`THREAD_SUPER_THREAD(thread)` is **`NULL`** for mixture thread, and not **`NULL`** for phase threads

  - mixture : **`if ( NULLP (THREAD_SUPER_THREAD(thread)))`**
  - phase : **`if (!NULLP (THREAD_SUPER_THREAD(thread)))`**