# Current ChemApp Developments and Projects

Stephan Petersen, GTT-Technologies

GTT-Technologies' 10th Annual Workshop, June 4 - 6, 2008


# 0. Overview

- Recent ChemApp developments
- Parallelization of ChemApp with OpenMP
- Current ChemApp projects



# 1. Recent ChemApp developments

New subroutines to manipulate thermochemical data (read from a data-file in ASCII format):

| | |
|---|---|
| TQGDAT | Get selected thermodynamic data of a phase constituent |
| TQLPAR | List all excess Gibbs energy or all excess magnetic interactions of a phase |
| TQGPAR | Get selected excess Gibbs energy or excess magnetic parameters of a phase |
| TQCDAT | Change selected data of a specified phase constituent or of a specified phase |
| TQWASC | Write a thermodynamic data-file in ASCII format |



# Code example - data manipulation

```
C Retrieving and changing thermodynamic data

      PROGRAM CAF33
      IMPLICIT NONE

      INTEGER NOERR, INDEXP, NVALV, I, J, NOPAR, NOEXPR, NVALA
      INTEGER LGTPAR(1999)

      DOUBLE PRECISION VALV(25),VALA(20,18)

      CHARACTER*156 CHRPAR(1999)


C Initialise ChemApp
      CALL TQINI(NOERR)

C Open the thermochemical data-file fec.dat (system Fe-C)
C for reading
      CALL TQOPNA('fec.dat', 10, NOERR)

C Read data-file
      CALL TQRFIL(NOERR)

C NOTE: Do not yet close the data-file, otherwise tqwasc (see below)
C won't be able to access the commentary block in fec.dat in order
C to attach it to the new file.
```



# Code example - data manipulation

```
C Get the index number of phase 'C_GRAPHITE'
      CALL TQINP('C_GRAPHITE ', INDEXP, NOERR)

C Retrieve Cp coefficients for the first Cp-range of C_GRAPHITE
      CALL TQGDAT(INDEXP, 1, 'Cp ', 1, NVALV, VALV, NOERR)

C Print coefficients
      DO I=1, NVALV
         WRITE(UNIT=*,FMT=*) 'VALV(',I,') is ', VALV(I)
      ENDDO
```

**Output:**

```
VALV( 1) is   24.3
VALV( 2) is   0.0009446
VALV( 3) is   0.
VALV( 4) is  -5125200.
VALV( 5) is   1.5858E+09
VALV( 6) is  -3.
VALV( 7) is  -1.44E+11
VALV( 8) is  -4.
```



# Code example - data manipulation

**Options for use with TQGDAT**

| | |
|---|---|
| H | Enthalpy/J at 298.15 K |
| S | Entropy/J.K-1 at 298.15 K |
| Cp | Heat capacity expression/J.K-1 or Helgeson terms for the models PIHZ, HELZ, HTSZ, HTWZ or HTDZ |
| Tt | Upper temperature limit/K for a Cp range |
| Ht | Transformation enthalpy/J at the upper temperature limit for a Cp range |
| V | Molar volume data in input order of a condensed phase constituent |
| Gc | Critical properties in input order of a gas phase constituent |
| M | Curie/Neel temperature and average magnetic moment per atom for a magnetic phase constituent |
| Ch | Charge of an aqueous phase constituent |



# Code example - data manipulation

```
C Get the phase index number of phase 'Fe3C_CEMENTITE'
      CALL TQINP('Fe3C_CEMENTITE ', INDEXP, NOERR)

C Retrieve Enthalpy/J at 298.15 K of Fe3C_CEMENTITE
      CALL TQGDAT(INDEXP, 1, 'H ', 1, NVALV, VALV, NOERR)
      WRITE(UNIT=*,FMT=*) 'Enthalpy/J at 298.15 K of Fe3C_CEMENTITE: ',
     *     VALV(1)

C Set the enthalpy/J at 298.15 K of Fe3C_CEMENTITE to a new value
      CALL TQCDAT(1, 0, 0, 1, INDEXP, 20000.D0, noerr)

      CALL TQGDAT(INDEXP, 1, 'H ', 1, NVALV, VALV, NOERR)
      WRITE(UNIT=*,FMT=*)  'New enthalpy/J at 298.15 K of ' //
     *     'Fe3C_CEMENTITE: ', VALV(1)

C Write data to file
      WRITE(UNIT=*,FMT=*) 'Writing modified data-file to fec_mod.dat'
      CALL TQWASC('fec_mod.dat', noerr)
      WRITE(UNIT=*,FMT=*) 'Done'
```

**Output:**

```
Enthalpy/J at 298.15 K of Fe3C_CEMENTITE:   25211.89
New enthalpy/J at 298.15 K of Fe3C_CEMENTITE:   20000.
Writing modified data-file to fec_mod.dat
Done
```



# Code example - data manipulation

```
C NOW close the data-file fec.dat, AFTER tqwasc has been called
      CALL TQCLOS(10, NOERR)

C Get index number of phase FCC_A1
      CALL TQINP('FCC_A1 ', INDEXP, NOERR)

C Retrieve all excess Gibbs energy interactions in phase FCC_A1
      CALL TQLPAR(INDEXP, 'G ',NOPAR, CHRPAR, LGTPAR, NOERR)

      WRITE(UNIT=*,FMT=*) 'Excess Gibbs energy interactions ' //
     *     'in phase FCC_A1:'

C Print them
      DO I=1, NOPAR
         WRITE(UNIT=*,FMT=*) 'CHRPAR(',I,') is ',
     *        CHRPAR(I)(1:LGTPAR(I))
      ENDDO
```

**Output:**

```
Excess Gibbs energy interactions in phase FCC_A1:
CHRPAR( 1) is    1: *2 (  1)-(  2)
```



# Code example - data manipulation

```
C Retrieve the Gibbs energy parameters for the first (and only) interaction
C in phase FCC_A1
      CALL TQGPAR(INDEXP, 'G ', 1, noexpr, nvala, vala, noerr)

      WRITE(UNIT=*,FMT=*)  'Printing Gibbs energy parameters for ' //
     *     'this interaction:'
      WRITE(UNIT=*,FMT=*)  'noexpr is ', noexpr
      WRITE(UNIT=*,FMT=*)  'nvala is ', nvala

      do i=1, noexpr
        do j=1, nvala
           WRITE(UNIT=*,FMT=*) 'noexpr = ',i,', nvala = ',j, ': ',
     *          vala(i,j)
        enddo
      enddo
```

**Output:**

```
Printing Gibbs energy parameters for this interaction:
noexpr is  1
nvala is  4
noexpr =  1, nvala =  1:  -34671.
noexpr =  1, nvala =  2:   0.
noexpr =  1, nvala =  3:   0.
noexpr =  1, nvala =  4:   0.
```

```
      END
```

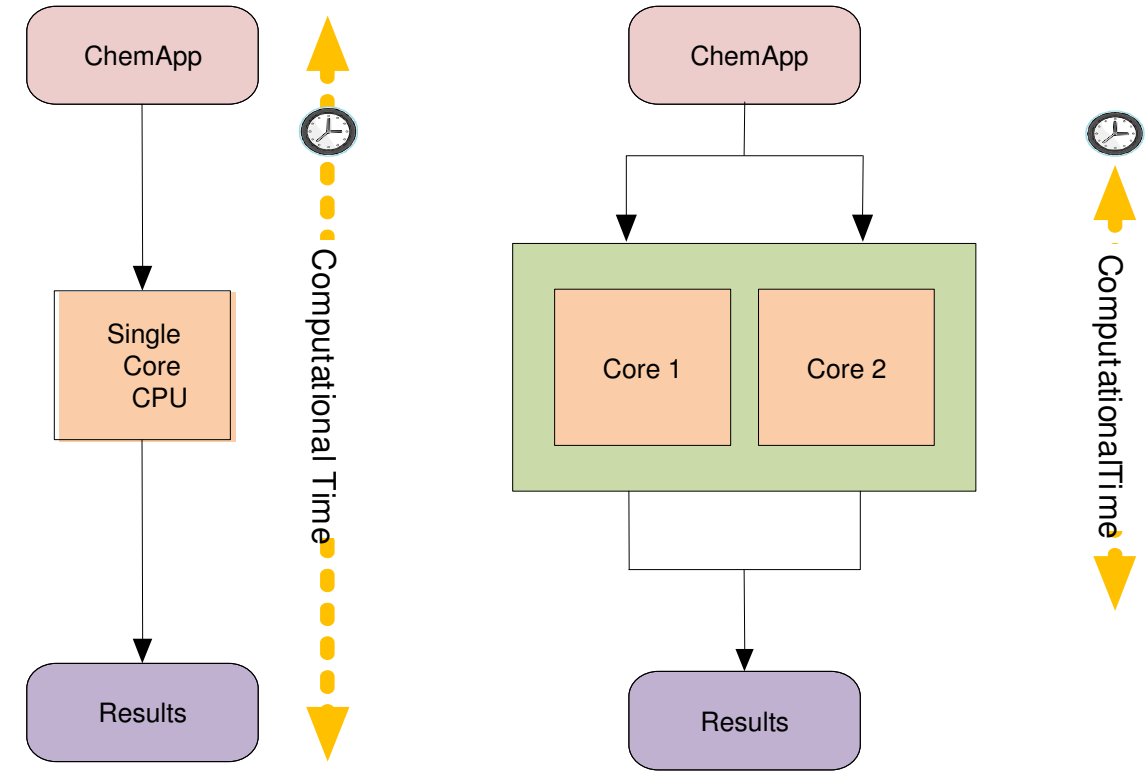


# 2. Parallelization of ChemApp with OpenMP

- *ChemApp* is being adapted to the current generation of multi-core processors by *parallelizing* it.
- *Parallelization* is an optimization technique to distribute work load efficiently on a multiple processor machine via multiple threads.
- Parallelization can reduce computation time significantly on a multi-core processor.
- **OpenMP** is used to efficiently parallelize *ChemApp* for multi-core processors.



# Parallelization of ChemApp




**Serial Execution** **Parallel Execution (shared memory)**

Schematic representation of Serial Execution vs. Parallel Execution of ChemApp on a shared memory architecture with significantly reduced computation time.



# Parallelization of ChemApp

- Parallelized version of ChemApp can run multiple threads depending upon the processor cores.
  - The number of threads are dynamically allotted but can be manually set via the environment variables.
- The parallelized version of *ChemApp* is designed with emphasis on faster calculation of thermochemical equilibria (subroutine TQCE).
  - Calculation of thermochemical equilibrium is typically the most computational intensive part of ChemApp code.



# Parallelization of ChemApp

- This initial parallelized version of *ChemApp* will be available only for *shared memory* architectures (multi-core CPUs + OpenMP) and not *distributed memory* architectures (Grids and Clusters + MPI/PVM).
- The current OpenMP parallelized version of ChemApp can still run on a cluster but needs additional MPI language bindings in the main program which calls ChemApp for maximum cluster performance.



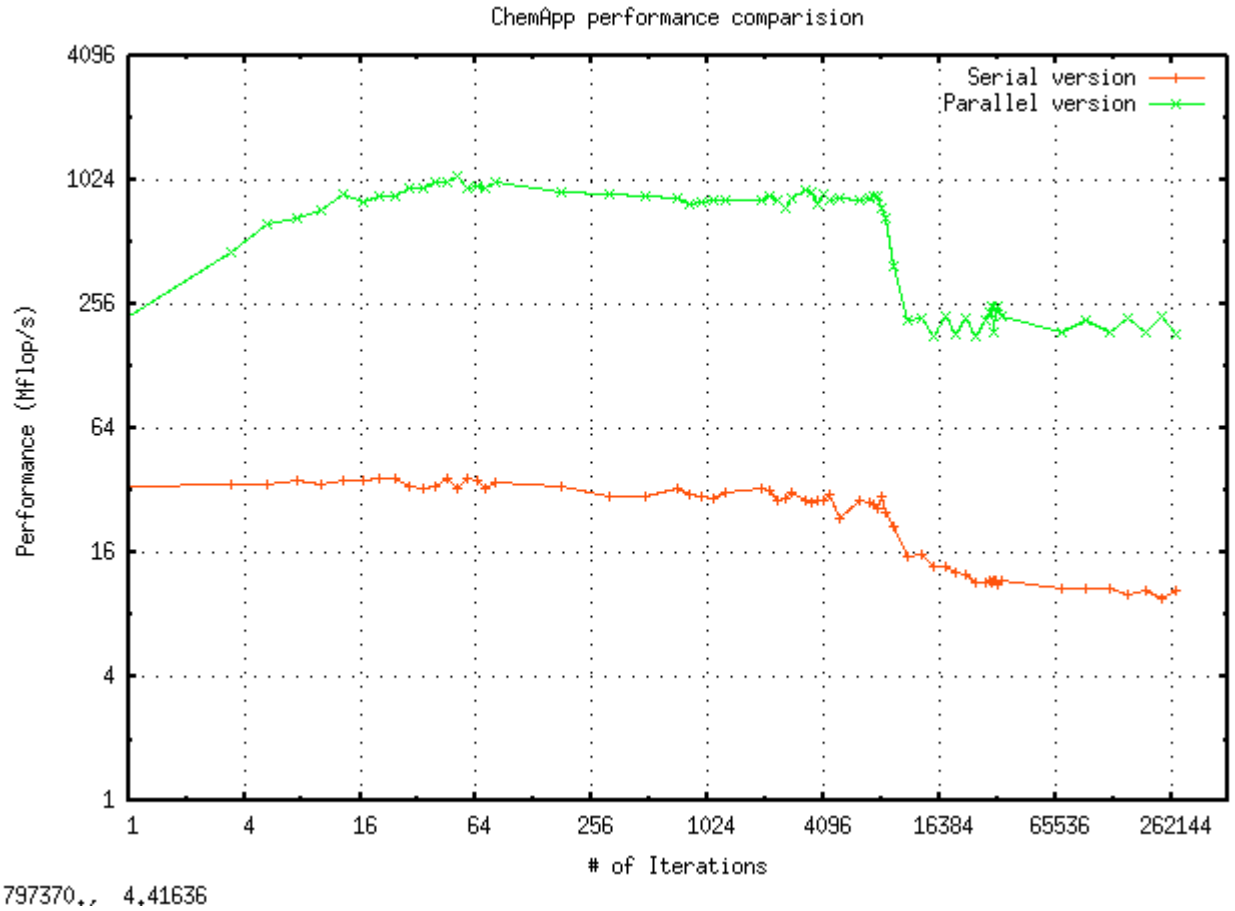
# Performance Overview

Parallel performance is largely dependent on:

- Parallel Computing Infrastructure
  - Number of cores in a processor
  - Speed of the individual cores
  - Memory capacity and latency
- Programming Style and Algorithm
  - Efficient memory handling and allocation
  - Performing more thermochemical equilibrium calculations can extract more parallel performance from ChemApp
- Amdahl's Law
  - Performance starts to degrade after a point
  - A program might run faster on a quad core or 8 core CPU than on a 16 core CPU (not necessarily always and depends on a number of factors)



# Parallel Performance






# Anticipated Parallel Performance

- On a typical dual-core processor, performance might scale between a factor of 1.2x and 1.75x of serial execution.
- On a higher number of processing cores and with a suitable parallelizable problem, performance can scale up to a generic factor of 0.75 x number of cores but is inhibited by *Amdahl's law*.



# 3. Current ChemApp projects

For examples of ChemApp applications, see Stephan Petersen and Klaus Hack: “The thermochemistry library ChemApp and its applications”, Int. J. Mat. Res., vol 98(10), 2007, p. 935

## Development of a new Nuclear Fuel Rod Model

Srdjan Simunovic, Larry J. Ott, Phani K. Nukala, Kevin T. Clarno, B. Radhakrishnan, Ted Besmann and Gorti Sarma, ORNL



# DEVELOPMENT OF A NEW NUCLEAR FUEL ROD MODEL

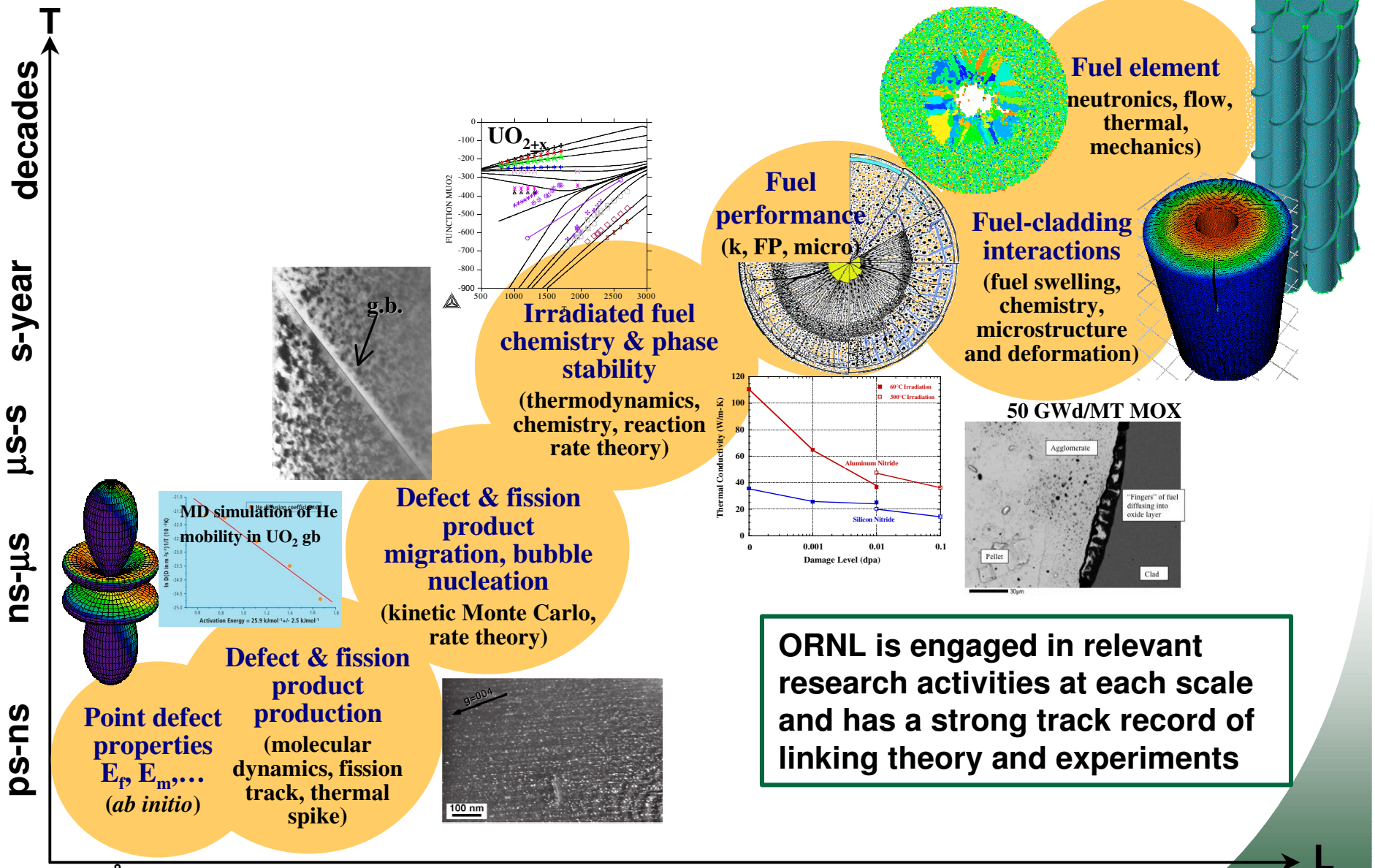
**Srdjan Simunovic[1], Larry J. Ott[2], Phani K. Nukala[1], Kevin T. Clarno[2], B. Radhakrishnan[1], Ted Besmann[3] and Gorti Sarma[1]**

[1]Computer Science and Mathematics Division
[2]Nuclear Science & Technology Division
[3]Materials Science & Technology Division

Managed by UT-Battelle
for the Department of Energy

OAK RIDGE National Laboratory

# Fuels Modeling is Inherently Multiscale




**ORNL is engaged in relevant research activities at each scale and has a strong track record of linking theory and experiments**

# ORNL Model Development Strategy

- **Top-down approach**
- **As modular as possible**
- **Assume two categories of phenomena:**
  - **Nuclear power generation, material structural changes, fission products, chemistry - *micro scale***
  - **Transport (power, species), mechanics - *macro scale***
- **Weak coupling between micro and macro phenomena**
  - **e.g. material structural changes are used as input data for thermo-mechanics loop**
- **Utilize current fuel codes as much possible**
  - **e.g. for initiation of 3D states for detailed analysis**
- **Combine elements of computer codes with proven parallel performance**

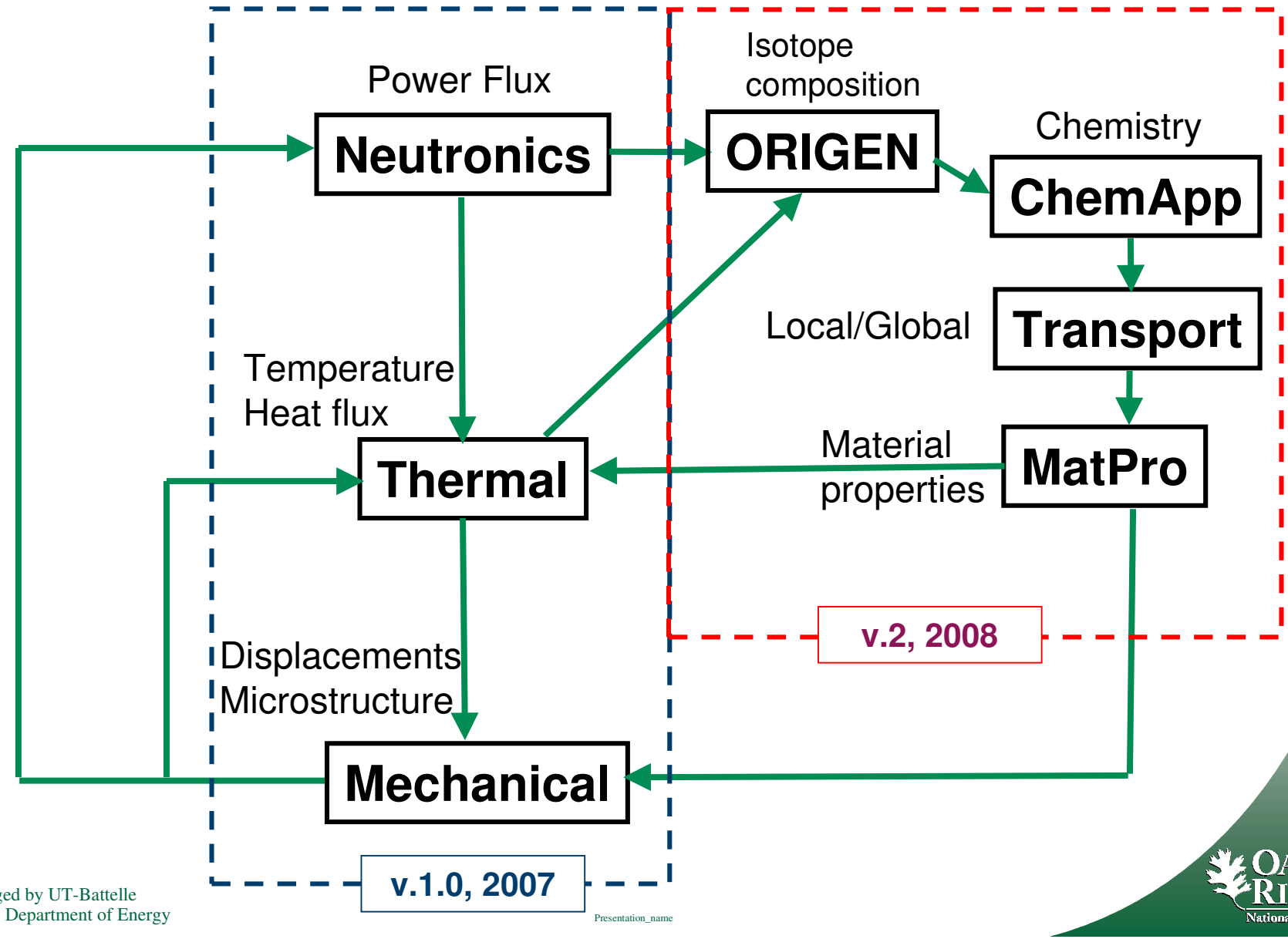
# Why High Performance Computing?

- **Spatial resolution:** Fundamental physics occurs at the grain size level (10-20 μm) in the fuel pellet
- **Multiple sub-models:** Each with high resolution
- **Coupling:** Increase in coupling increases problem size
- **Process time of interest:** Minutes, Hours and Days
- **Simple example:**
  - Mesh resolution (*h*) for crack modeling for CPI estimated from material toughness (energy dissipated for failure) and strength

$$G_{fr} = \frac{1}{2}\frac{(\sigma_R)^2}{E}h \quad , \quad \left.\begin{array}{l} G_{fr}(UO_2) = 3J/m^2 \\ \sigma_R(UO_2) = 136MPa \\ E(UO_2) = 160GPa \end{array}\right\} \quad h = 50\mu m$$

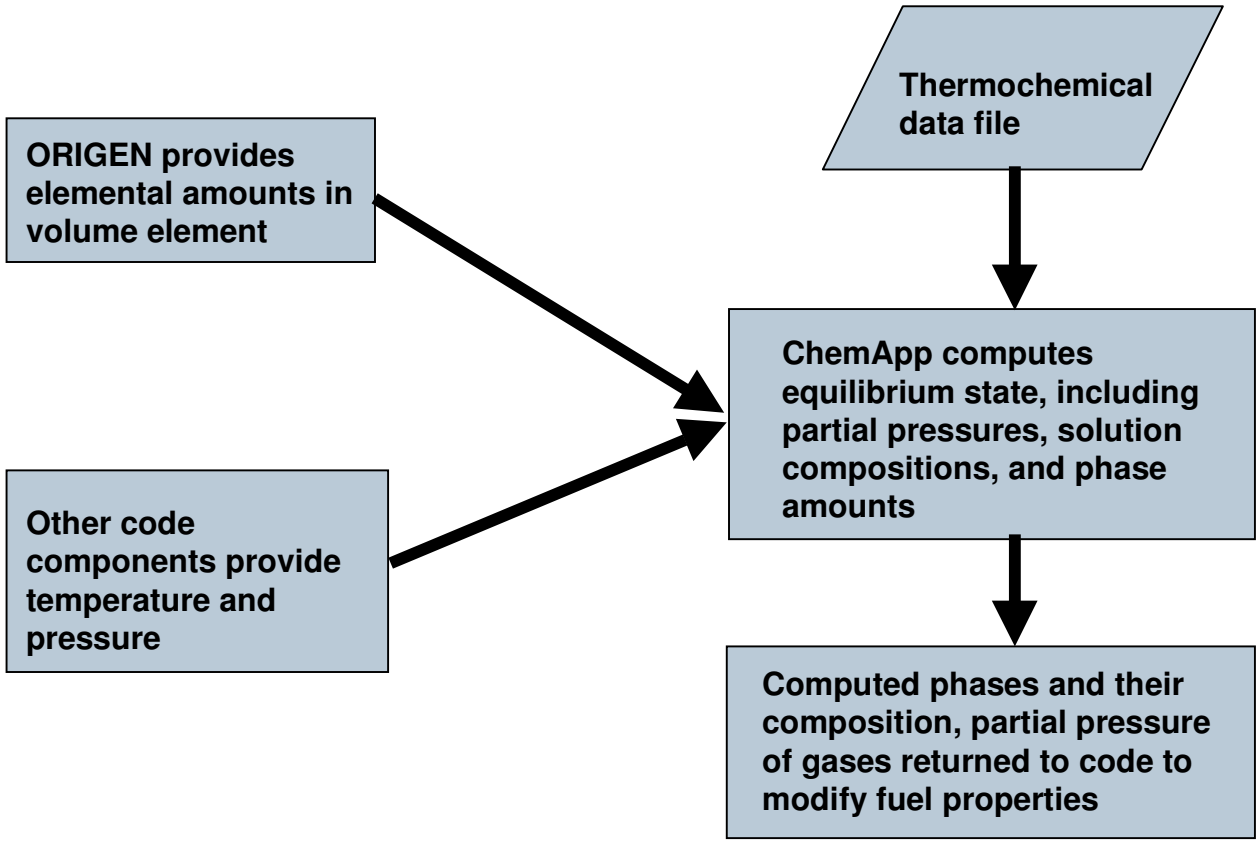
  - Adding microstructure and local transport models will require smaller *h*

# ORNL Code Integration for Nuclear Fuel Modeling



# Example: Chemistry Module

- **Chemistry Module computes equilibrium state from input elements and returns phases, activities and partial pressures**
- **Results are used in transport and material properties modules**




Solve for each species transport requires a global solve operation

Time increments are based on power generation conditions (e.g. hours, days)

# Example: Chemistry Module Calculation of MOX Fuel at 50 GWd/t

***Thermochemical calculations yields local gas phase composition, oxygen-to-metal ratio of the fuel, and secondary metal and oxide fission product phases***

**Fluorite Phase Oxide Fuel**

$(U, Pu, Am, Np, La, Ce, Nd, Pr)O_{2-x}$
4020 mol
O/M Ratio = 1.9914

**Metal Fission Product Phase**

(Mo, Pd, Rh, Ru, Tc) - 109 mol

**Oxide Fission Product Phases**

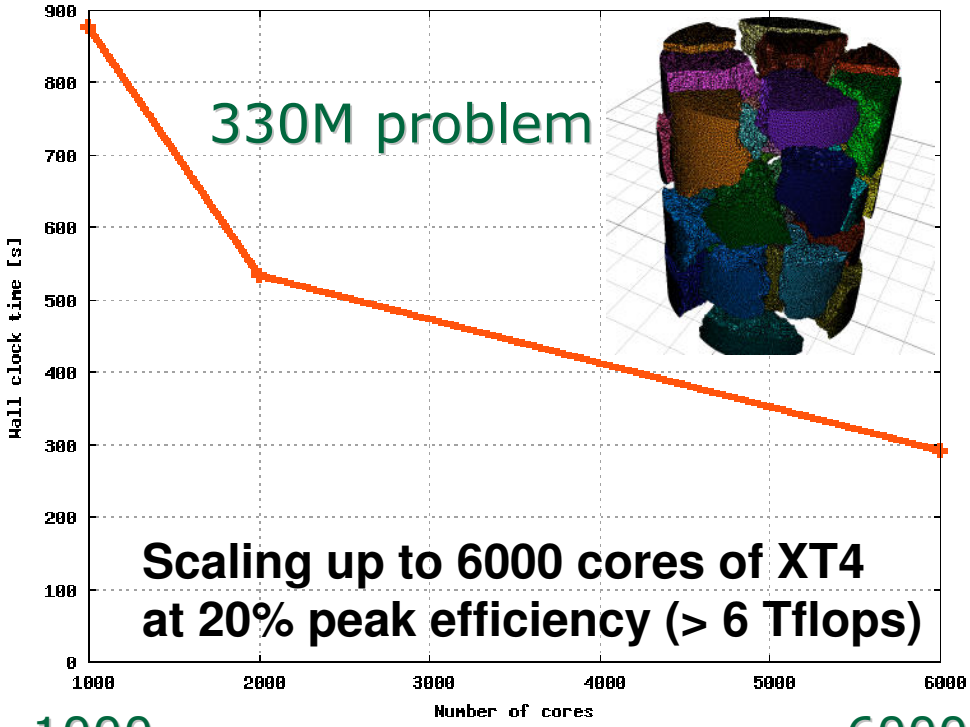
| | mol |
|---|---|
| $BaMoO_4$ | 1.4661E+01 |
| $Cs_2Zr_3O_7$ | 1.1123E+01 |
| SrUO4 | 8.4450E+00 |
| $Cs_sZrO_3$ | 6.0323E+00 |

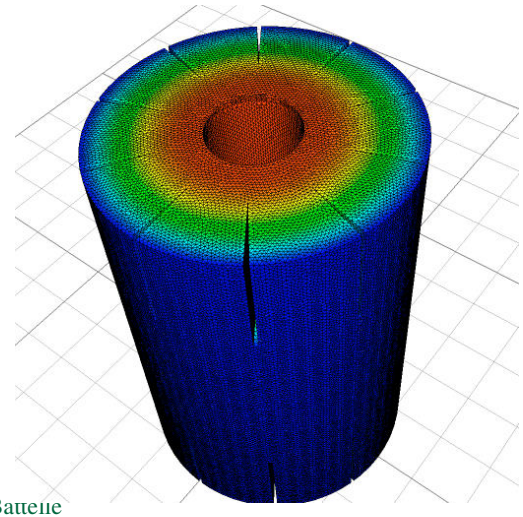
**Major Gas Phase Species**

| | EQUIL AMOUNT mol | MOLE FRACTION | FUGACITY bar |
|---|---|---|---|
| Xe | 5.3100E+01 | 7.9816E-01 | 7.9816E-01 |
| $Mo_3O_9$ | 4.6833E+00 | 7.0396E-02 | 7.0396E-02 |
| Kr | 3.9500E+00 | 5.9373E-02 | 5.9373E-02 |
| CsI | 2.8819E+00 | 4.3319E-02 | 4.3319E-02 |
| I | 5.3848E-01 | 8.0940E-03 | 8.0940E-03 |
| $Mo_2O_6$ | 4.5201E-01 | 6.7942E-03 | 6.7942E-03 |
| $Cs_2MoO_4$ | 3.9158E-01 | 5.8859E-03 | 5.8859E-03 |
| $Mo_4O_{12}$ | 3.8642E-01 | 5.8084E-03 | 5.8084E-03 |
| MoO3 | 8.2018E-02 | 1.2328E-03 | 1.2328E-03 |
| $BaMoO_4$ | 1.8703E-02 | 2.8114E-04 | 2.8114E-04 |
| Cs | 1.2275E-02 | 1.8451E-04 | 1.8451E-04 |
| Pd | 9.9805E-03 | 1.5002E-04 | 1.5002E-04 |
| $UO_3$ | 6.8465E-03 | 1.0291E-04 | 1.0291E-04 |

# HPC Simulations

- Coarse resolution (100 μm) equivalent to a fuel rod with 1 million variables per pellet
- Fine resolution (10 μm) simulation of a fuel pellet (1.1 Billion DOF)
- **Largest FEM mechanics system solved so far**



Models are developed for simulation of cracked fuel and microstructure resolution