Chemical speciation in GPU for the parallel resolution of reactive transport problems

Pablo Gamazo[1], *Lucas Bessone*[1], *Julián Ramos*[1], *Elena Alvareda*[1] *and Pablo Ezzatti*[2]
(1)Universidad de la República, Departamento del Agua (Water Department), Salto, Uruguay
(2)Universidad de la República, Facultad de Ingeniería, Montevideo, Uruguay

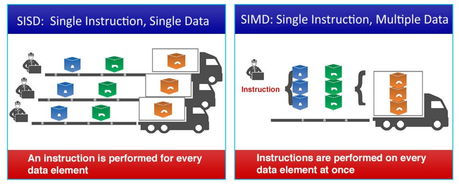
EGU General Assembly 2020
Online | 4–8 May 2020
EGU2020-1631

i Why Chemical Specialization on GPU?

- Reactive Transport modeling (RTM) can be very computationally expensive and therefor several efforts have been made in the last decade in order to parallelize it.
- Despite the recent advances on GPU, only a few works explore this architecture for RTM, and they mainly focused on the implementation of parallel sparse matrix solvers for the component transport.
- Solving the component transport consumes an important amount of time during simulation, but another time-consuming part of RTM is the chemical speciation, a process that has to be performed multiple times during the resolution of each time step over all nodes (or discrete elements of the mesh).
- In this work, we show the performance of a speciation code implemented in CUDA and we compare running times with an OpenMP code that uses IPhreeqc.

ii Chemical Specialization with Single Instruction Multiple Data

- Since speciation involves local calculations, it is a priory a very attractive process to parallelize. But, to the author's knowledge, no work on literature explores chemical speciation parallelization on GPU.
- One of the reasons behind this might be the fact that the unknowns and the number of chemical equations that act over each node might be different and can dynamically change in time, due to mineral precipitation or dissolution. This can be a drawback for the single instruction multiple data paradigm since it might lead to the resolution of several systems with potentially different sizes all over the domain. This may cause thread divergence, which penalizes the performance of the GPU code.



iii Solving Chemical Specialization

- Chemical speciation implies solving the distribution of all elements among defined chemical species in a system.
- A component is one collection of chemically independent constituents of a system. The number of components represents the minimum number of independent species necessary to define the composition of all phases of the system (plus pressure and temperature).
- One of the most common ways to perform speciation is from the concentration of primary species (which are equal in number to the number of system components). From the concentrations of the primary species, a system of equations is solved that involves the calculation of the activities of all spices and meet all laws of mass action.
- Let's consider a system with 3 species and one equilibrium reaction:

Reaction: $aA + bB \leftrightarrow cC$
Species: $A, B \; and \; C$

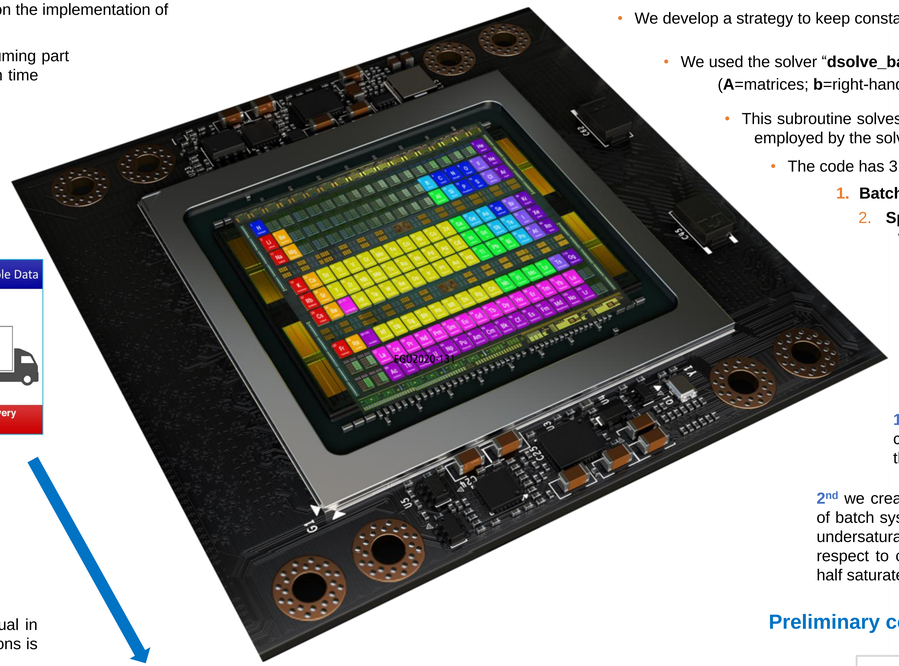
Unknowns
species concentration: $[A], [B] \; and \; [C]$
species activities: $\{A\}, \{B\} \; and \; \{C\}$

Equations
Activity model: $\{A\} = f_A([A], [B], [C])$
$\{B\} = f_B([A], [B], [C])$
$\{C\} = f_C([A], [B], [C])$

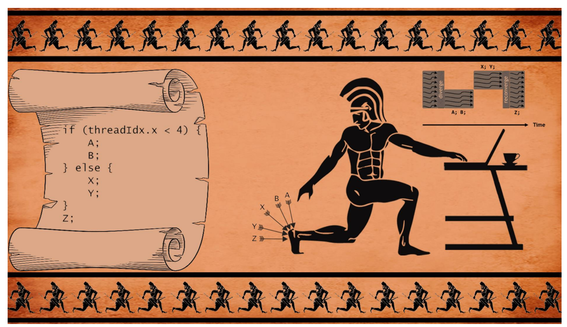
Law of mass action: $\frac{\{C\}^c}{\{A\}^a \{B\}^b} = K$

- The system has 6 unknowns and 4 equations.
- But if the reaction involves a mineral, it has to be omitted when the mineral is undersaturated.
- If the mineral activity is considered as constant (as in most cases when solving groundwater problems), the number of primary species is reduced by 1.

This might lead to systems of equations with a different size on each batch system, which may cause thread divergence



If not treated properly thread divergence can be the Achilles' heels of any CUDA code



iv Implementation

- We develop a strategy to keep constant the size of the system to be solved during speciation that minimizes thread divergence.
- We used the solver "**dsolve_batch(A, b, x, n, batch)"** available at NVIDIA DEVELOPER site: *https://developer.nvidia.com*
  (**A**=matrices; **b**=right-hand sides; **n**=systems dimension, **batch**=number of batches).
- This subroutine solves many double-precision systems of linear equations, each with a single right-hand side. Partial pivoting is employed by the solver algorithm for increased numerical stability.
- The code has 3 main subroutines:
  1. **Batch_Activity**: calculates aqueous species activity coefficients (and its derivatives) for a given composition.
  2. **Speciation_From_Primary**: calculates secondary species concentration from primary species concentration while meeting all active mass action laws.
  3. **Speciation_From_Components**: calculates all species concentration form the components values.
- Each batch system is processed by a single thread, for the calculation of activities coefficients and its derivatives, saturations indexes and the contributions to the matrix **A** and the vector **b**.

v application

1st we solve the following chemical system considering the B-dot activity model:

$$HCO_3^- + Ca^{2+} \leftrightarrow CaCO_{3(aq)} + H^+$$
$$H_2O + Ca^{2+} \leftrightarrow CaOH^+ + H^+$$
$$HCO_3^- \leftrightarrow CO_3^{2-} + H^+$$
$$H_2O \leftrightarrow OH^- + H^+$$

$$HCO_3^- + Ca^{2+} \leftrightarrow CaHCO_3^+$$
$$HCO_3^- + H^+ \leftrightarrow CO_2 + H_2O$$
$$HCO_3^- + Ca^{2+} \leftrightarrow CaCO_{3(m)} + H^+$$

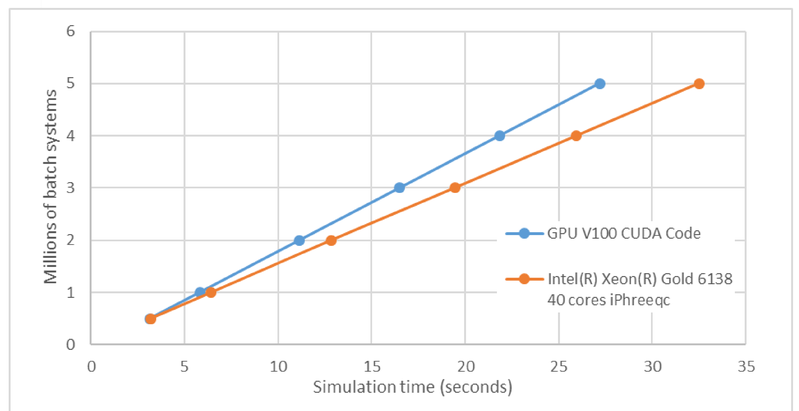
2nd we create millions of batch systems, half undersaturated with respect to calcite and half saturated:

Without calcite
```
pH = 7
Ca = 1 mmol/kgw
C  = 1 mmol/kgw
```

With calcite
```
Ca = 1 mmol/kgw
C  = 1 mmol/kgw
```

3rd we change randomly 5% of component concentration and we run the speciation.

Preliminary comparison with IPhreeqc:



- For the studied system, the CUDA code runs speciation 18% faster than the CPU OpenMP code.