GPU-accelerated interstellar chemistry with WIND, a new general ODE solver

Alex Gurvich
Blue Waters Graduate Fellow
Northwestern University
June 4th, 2019 - Blue Waters Symposium
I use BW to:
develop a new
GPU-accelerated
ODE solver
Galaxy formation and interstellar chemistry (on FIRE)

CURRENT:
- Gravity
- Hydrodynamics
- Star formation
  - feedback

NEW: time-dep. chemistry
- Informs gas cooling rate
- Predicts observations
  - abundances/masses
  - emission/absorption spectra
The CHIMES time-dependent chemistry network

- Reaction network of 157 coupled "stiff" ODEs
  - includes metal ions & molecules
  - CO, H₂, OVI, etc...
  - many different timescales
- Prohibitively expensive, up to ~90% of work
How to integrate ODEs
Solving differential equations numerically both explicitly and implicitly

**Explicit: Runge Kutta (RK2)**

\[ y_{n+1} = y_n + hf\left(y_n + \frac{h}{2}f(y_n)\right) \]

must resolve short timescales or diverges

**Implicit: Semi-Implicit Euler (SIE)**

\[ y_{n+1} = y_n + hf(y_{n+1}) \]
\[ y_{n+1} \approx y_n + h \left(1 - h\frac{\partial f}{\partial y}\right)^{-1} f(y_n) \]

converges to answer at late times
Solving differential equations numerically both explicitly and implicitly

**Explicit: Runge Kutta (RK2)**

\[ y_{n+1} = y_n + hf\left( y_n + \frac{h}{2}f(y_n) \right) \]

must resolve short timescales or diverges

**Implicit: Semi-Implicit Euler (SIE)**

\[ y_{n+1} = y_n + hf(y_{n+1}) \]
\[ y_{n+1} \approx y_n + h \left( 1 - h \frac{\partial f}{\partial y} \right)^{-1} f(y_n) \]

canverges to answer at late times

Goal is to find final value
Solving (coupled) ODEs simultaneously with linear algebra
Why GPUs?
How are GPUs and CPUs different?

- Many threads that operate concurrently
- Good for vector operations & linear algebra

- Memory bandwidth
- Typically requires substantial code/algorithm rewrite
- Optimal configuration hardware dependent

CPU
~ 10s of cores

GPU
~1000s of cores
GPUs aren’t the next big thing, they are the current thing

- Modern HPC resources are GPU powered
- ~95% peak flops are on GPUs
- more power efficient

credit: NVIDIA
GPUs aren’t the next big thing, they are the current thing

- Modern HPC resources are GPU powered
- ~95% peak flops are on GPUs
- more power efficient

<table>
<thead>
<tr>
<th>#1 on top 500</th>
<th>Summit, Peak Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>CPU</td>
</tr>
<tr>
<td>Type</td>
<td>POWER9</td>
</tr>
<tr>
<td>Count</td>
<td>9,216</td>
</tr>
<tr>
<td></td>
<td>2 \times 18 \times 256</td>
</tr>
<tr>
<td>Peak FLOPS</td>
<td>9.96 PF</td>
</tr>
<tr>
<td>Peak AI FLOPS</td>
<td>3.456 EF</td>
</tr>
<tr>
<td></td>
<td>V100</td>
</tr>
<tr>
<td></td>
<td>27,648</td>
</tr>
<tr>
<td></td>
<td>6 \times 18 \times 256</td>
</tr>
<tr>
<td></td>
<td>215.7 PF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#2 on top 500</th>
<th>Peak TFLOPS (CPUs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sierra Peak Performance</td>
<td>4,666</td>
</tr>
<tr>
<td>Peak TFLOPs (GPUs)</td>
<td>120,960</td>
</tr>
<tr>
<td>Peak TFLOPs (CPUs+GPUs)</td>
<td>125,626</td>
</tr>
</tbody>
</table>

credit: NVIDIA

WIND

Already implemented:
2 solvers, RK2 and SIE on both the GPU and CPU
How does it perform?
Using a five species H-He chemical network as a test case

\[ R^a = \kappa_{\mu \nu} n^\mu n^\nu \]

\[ n^\mu = \begin{bmatrix} n_{\text{HI}} \\ n_{\text{HII}} \\ n_{\text{HeI}} \\ n_{\text{HeII}} \\ n_{\text{HeIII}} \end{bmatrix} \]

\[ n_e = n_{\text{HI}} + n_{\text{HII}} + 2n_{\text{HeIII}} \]

\[ R^{\text{HI}} = \alpha_{\text{HI}} n_e n_{\text{H}} - \left( \alpha_{\text{HI}} + \Gamma_{e,\text{HI}} + \Gamma_{\gamma,\text{HI}} / n_e \right) n_e n_{\text{HI}} \]

\[ R^{\text{HII}} = - R^{\text{HI}} \]

\[ R^{\text{HeI}} = (\alpha_{\text{HeI}} + \alpha_d) n_e n_{\text{HeI}} - \left( \Gamma_{e,\text{HeI}} + \Gamma_{\gamma,\text{HeI}} / n_e \right) n_e n_{\text{HeI}} \]

\[ R^{\text{HeII}} = - (R^{\text{HeI}} + R^{\text{HeIII}}) \]

\[ R^{\text{HeIII}} = \left( \Gamma_{e,\text{HeII}} + \Gamma_{\gamma,\text{HeII}} / n_e \right) n_e n_{\text{HeII}} - \alpha_{\text{HeIII}} n_e n_{\text{HeIII}} \]

assume temperature & density is fixed \hspace{1cm} \textbf{evolve to equilibrium}
The graph shows the fractional abundance of different species over time in a fixed tolerance scenario. The x-axis represents time in years, ranging from 0 to 200, and the y-axis represents the fractional abundance ranging from 0.0 to 1.0. The equations for the scenario are 

\[ \log(nH) = 2.0 - \log(T) = 2.0 \]
Mimic the computational challenge of the full CHIMES chemistry network

total size: 150x150
~$10^6$ systems
Changing the number of systems

The graph shows the time to solution (in seconds) as a function of the number of systems ($N_{\text{systems}}$). The data points are fitted with linear equations:

- **SIE-cpu**: $37 \text{ s} + 1.8 \text{ s/sys}$
- **SIE-gpu**: $10 \text{ s} + 170.7 \text{ ms/sys}$
- **RK2-cpu**: $206 \text{ ms} + 189.7 \mu\text{s/sys}$
- **RK2-gpu**: $5 \text{ ms} + 276.5 \mu\text{s/sys}$

The graph is labeled as follows:

- **Kalar96 on BW: K20x - 150 equations**
Changing the number of systems

![Graph showing time to solution vs. number of systems for SIE-cpu, RK2-cpu, and SIE-gpu.](image)
Changing the size of each system

- **SIE-cpu**: $6\text{s} + 269.0\ \mu\text{s}/\text{eqn}^3$
- **SIE-gpu**: $862\ \text{ms} + 40.8\ \mu\text{s}/\text{eqn}^3$
- **RK2-gpu**: $208\ \text{ms} + 288.7\ \mu\text{s}/\text{eqn}$
- **RK2-cpu**: $31\ \text{ms} + 824.4\ \mu\text{s}/\text{eqn}$

*Ka:296 on BW: K20x - 500 systems*
Changing the size of each system
Takeaways

- We have to adapt to a new computing paradigm to take advantage of the most powerful HPC resources.
- WIND is a new general, multi-method, GPU-accelerated ODE solver.
  - Explicit methods are efficient on smooth problems.
  - Implicit methods are advantageous for stiff systems such as a more complete chemistry network like CHIMES.
Next steps

• Implement optimized solvers for sparse systems
  • cost scales as $N^3 \rightarrow N$
  • reduces memory footprint from $N^2 \rightarrow N$

• Testing on more recent GPUs
  • Pascal & Volta

• Encode CHIMES network into WIND and attach to FIRE to run on galaxy simulations
Thanks to Blue Waters Fellowship