Kppa: A High Performance Source Code Generator for Chemical Kinetics

John C. Linford

jlinford@paratools.com

ParaTools, Inc.

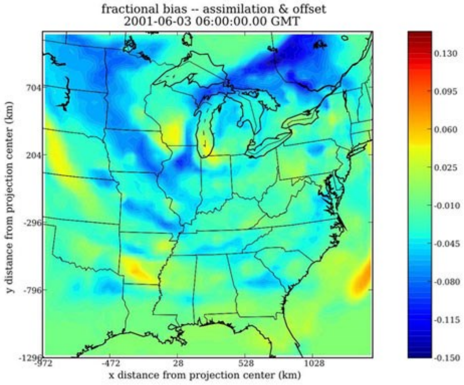
EMiT'15, Manchester UK

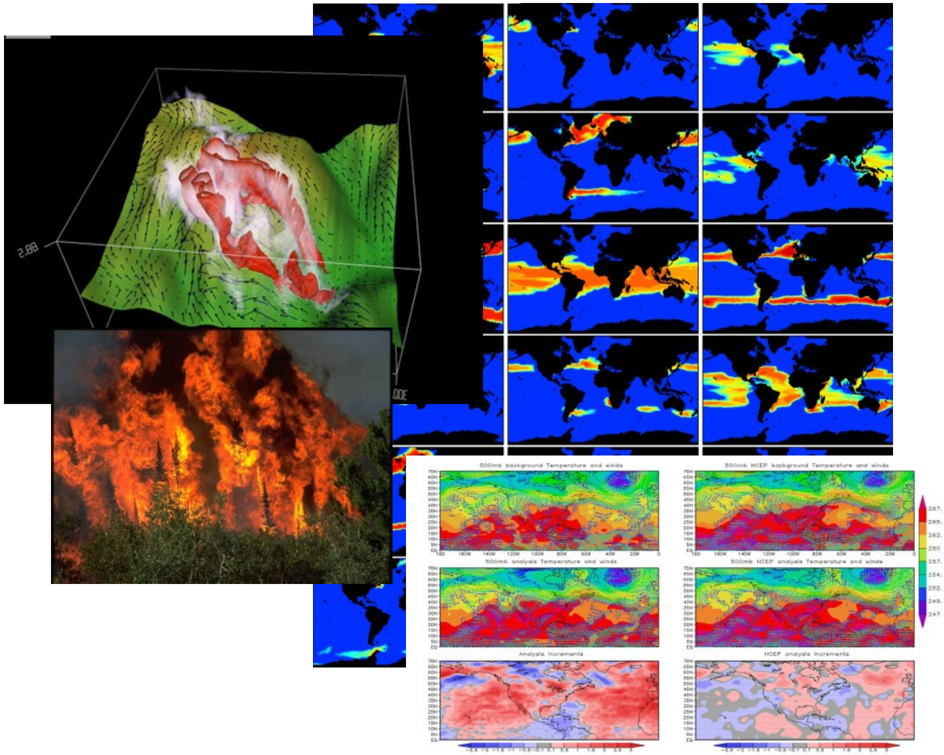
1 July 2015

Numerical Simulation of Chemical Kinetics

CLIMATE & ATMOSPHERE

- Air and water quality
- Climate change
- Wildfire tracking
- Volcanic eruptions

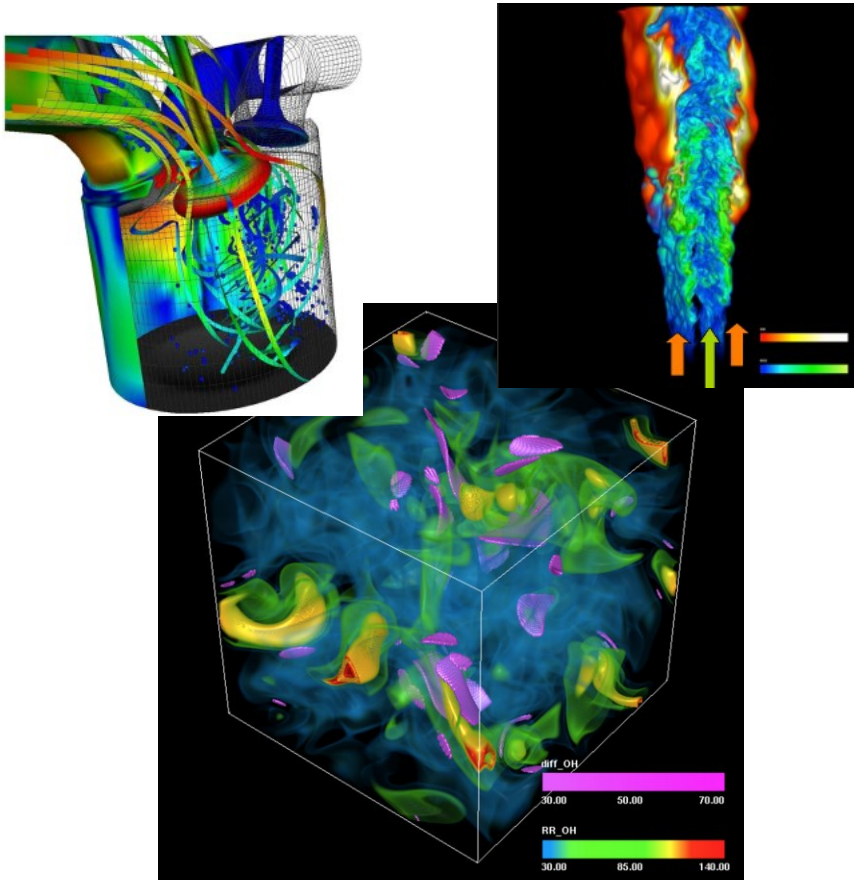

fractional bias -- assimilation & offset
2001-06-03 06:00:00.00 GMT
y distance from projection center (km)
704
204
-296
-796
-1296
-972
-472
28
528
1028
x distance from projection center (km)
0.130
0.095
0.060
0.025
-0.010
-0.045
-0.080
-0.115
-0.150



Numerical Simulation of Chemical Kinetics

ENERGY

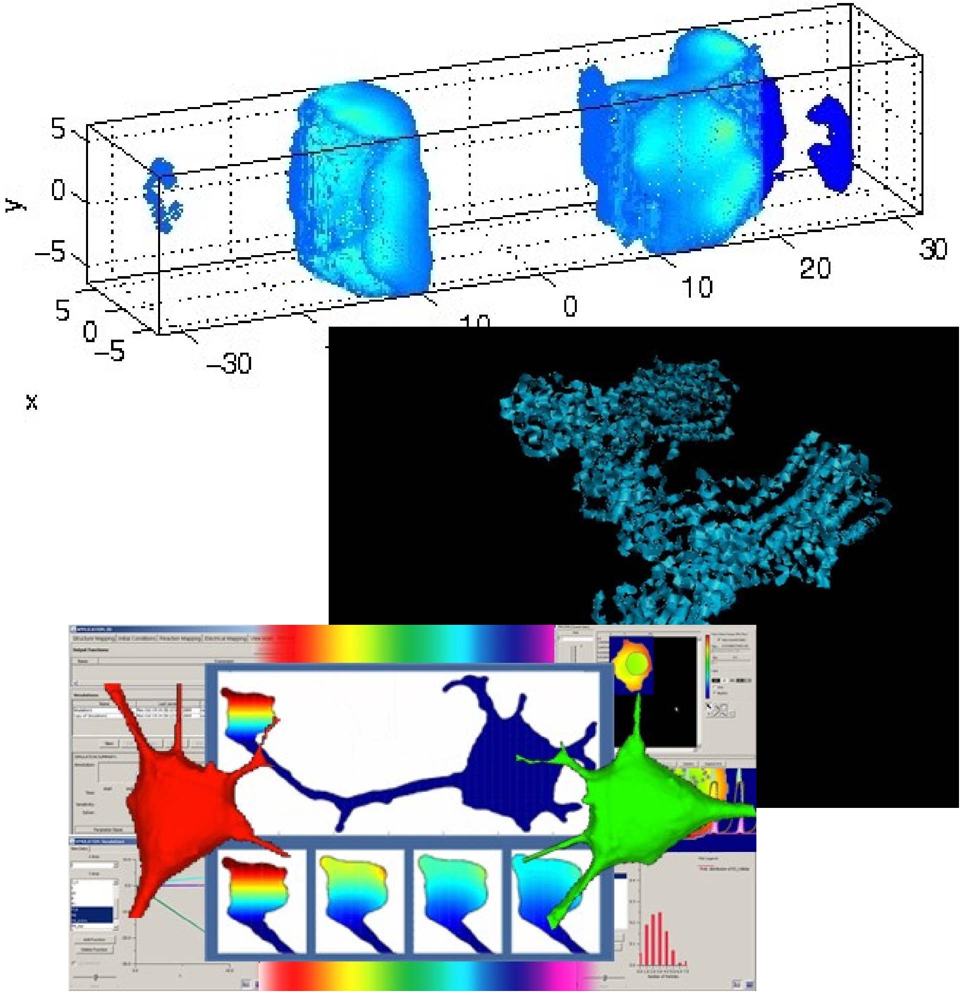
- Low emissions aircraft
- Alternate fuels
- High efficiency ICE



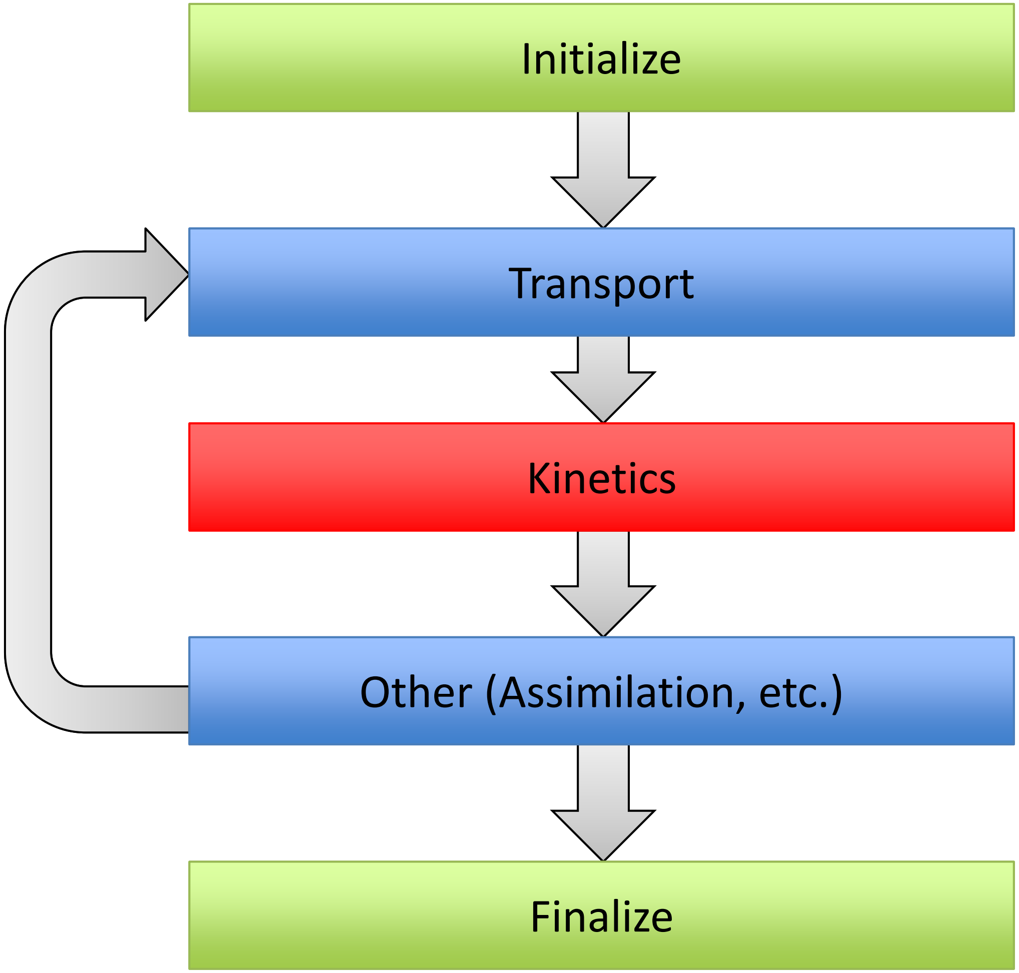
Numerical Simulation of Chemical Kinetics

Medical Research

- Microorganism growth
- Cell biology
- Cancer growth & treatment



Whole System Model Outline


Initialize

Transport

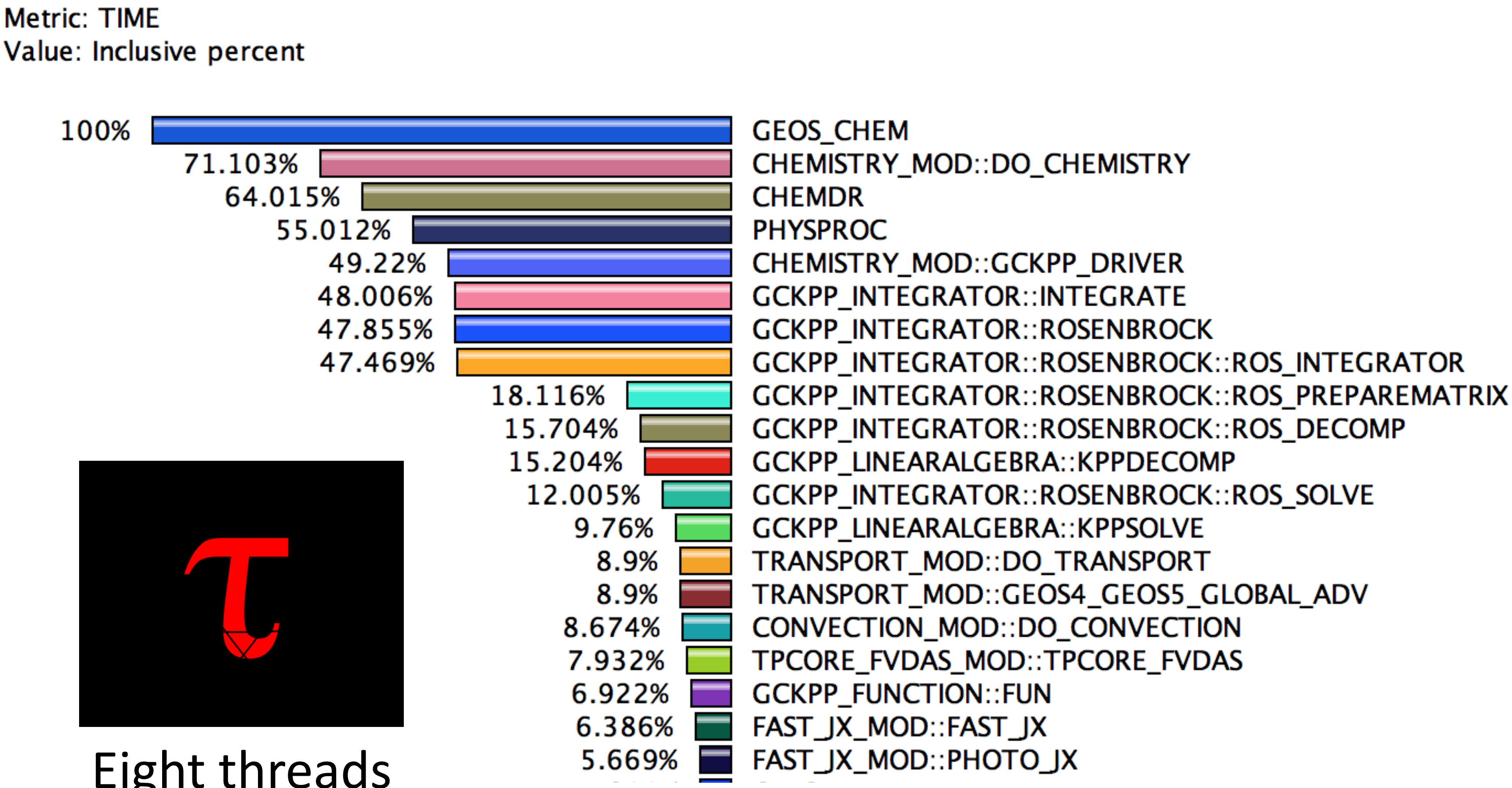
Kinetics

Other (Assimilation, etc.)

Finalize

60-99% of runtime

70% of GEOS-Chem Runtime is Chemistry

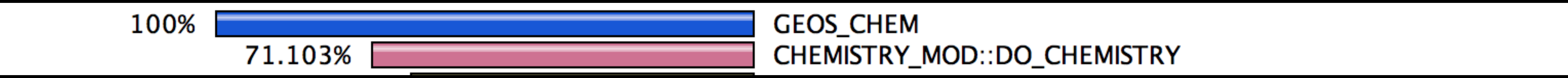


Metric: TIME
Value: Inclusive percent

100%	GEOS_CHEM
71.103%	CHEMISTRY_MOD::DO_CHEMISTRY
64.015%	CHEMDR
55.012%	PHYSPROC
49.22%	CHEMISTRY_MOD::GCKPP_DRIVER
48.006%	GCKPP_INTEGRATOR::INTEGRATE
47.855%	GCKPP_INTEGRATOR::ROSENBROCK
47.469%	GCKPP_INTEGRATOR::ROSENBROCK::ROS_INTEGRATOR
18.116%	GCKPP_INTEGRATOR::ROSENBROCK::ROS_PREPAREMATRIX
15.704%	GCKPP_INTEGRATOR::ROSENBROCK::ROS_DECOMP
15.204%	GCKPP_LINEARALGEBRA::KPPDECOMP
12.005%	GCKPP_INTEGRATOR::ROSENBROCK::ROS_SOLVE
9.76%	GCKPP_LINEARALGEBRA::KPPSOLVE
8.9%	TRANSPORT_MOD::DO_TRANSPORT
8.9%	TRANSPORT_MOD::GEOS4_GEOS5_GLOBAL_ADV
8.674%	CONVECTION_MOD::DO_CONVECTION
7.932%	TPCORE_FVDAS_MOD::TPCORE_FVDAS
6.922%	GCKPP_FUNCTION::FUN
6.386%	FAST_JX_MOD::FAST_JX
5.669%	FAST_JX_MOD::PHOTO_JX

Eight threads
Intel Core i7-4820K CPU
3.70GHz

70% of GEOS-Chem Runtime is Chemistry

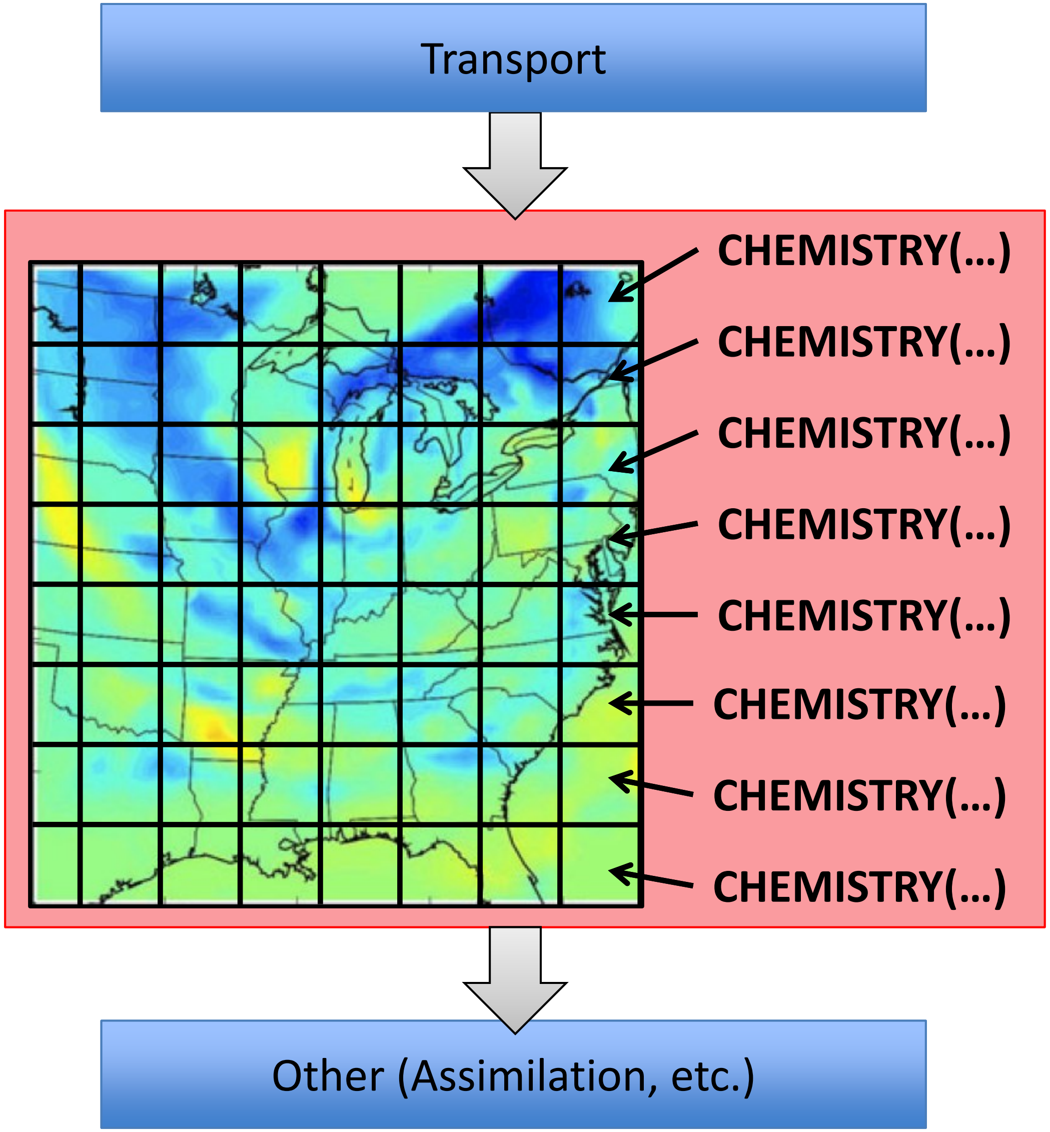


Solver is applied over domain "grid"

- Reactions in a "grid cell" depend on concentrations in that cell only
  - Increasing resolution greatly increases computational cost
  - Embarrassingly parallel

- Low computational intensity, e.g. 0.08 operations per byte
  - **Not** well suited to GPUs
  - Need large, low latency cache-per-thread

- ***Cost limits capability***



Transport

CHEMISTRY(…)

CHEMISTRY(…)

CHEMISTRY(…)

CHEMISTRY(…)

CHEMISTRY(…)

CHEMISTRY(…)

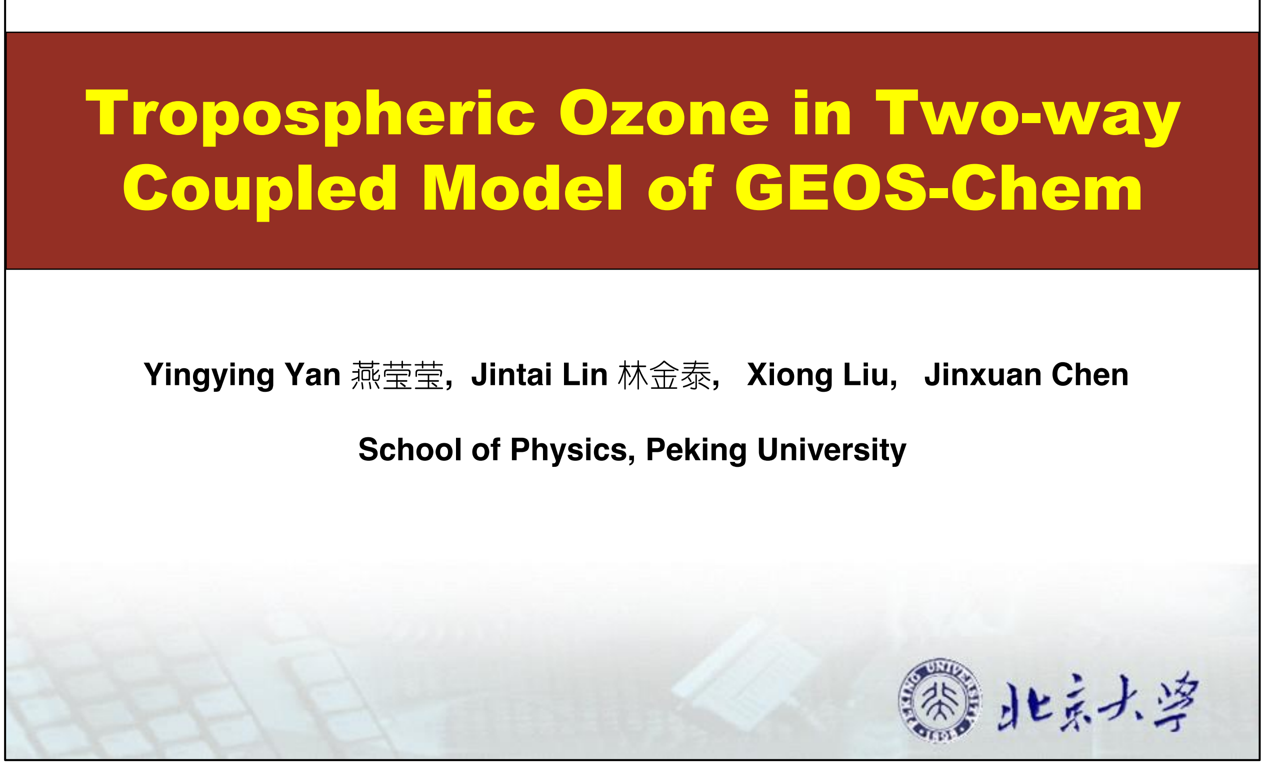
CHEMISTRY(…)

CHEMISTRY(…)

Other (Assimilation, etc.)

Ozone Simulation with GEOS-Chem

The 7th International GEOS-Chem Meeting (IGC7)



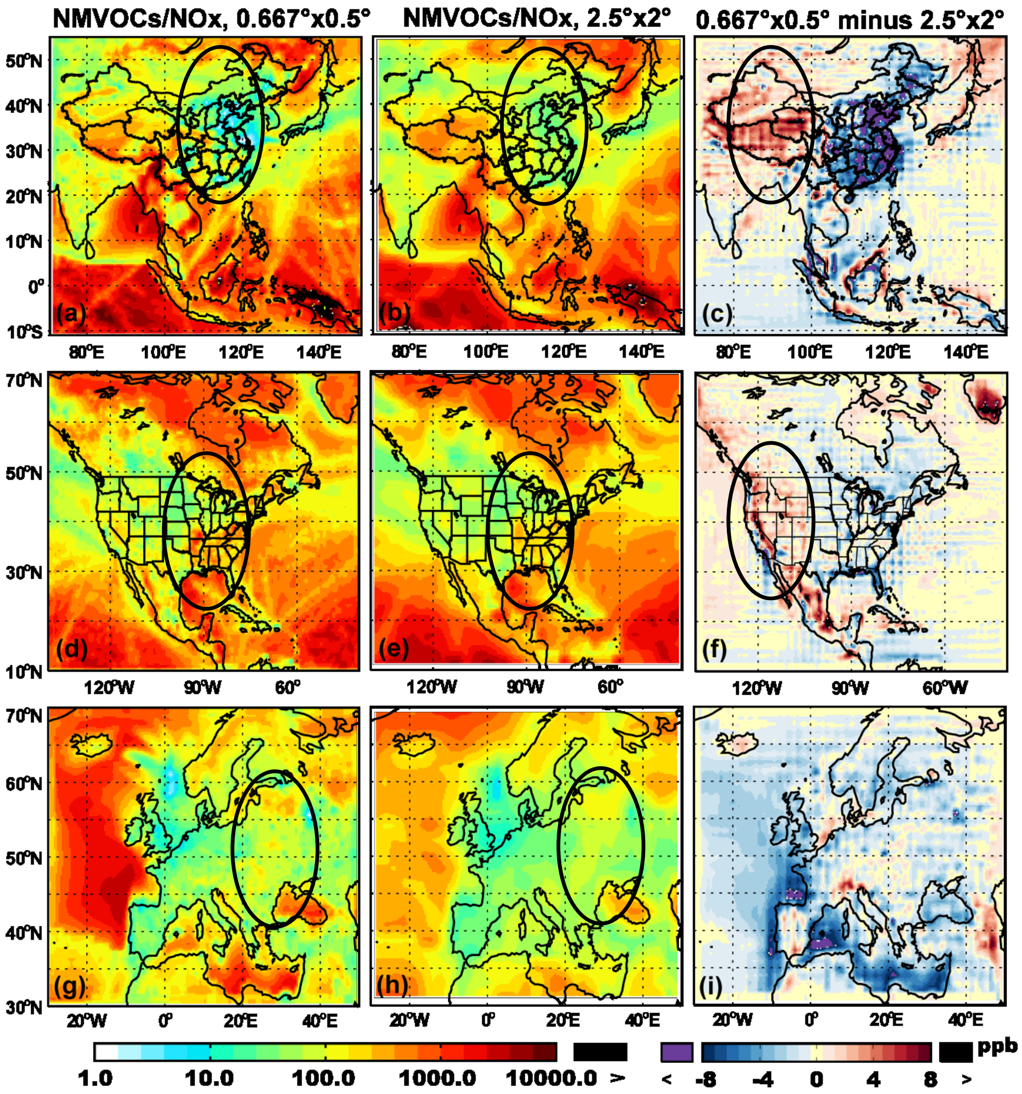
Tropospheric Ozone in Two-way Coupled Model of GEOS-Chem

Yingying Yan 燕莹莹，Jintai Lin 林金泰， Xiong Liu， Jinxuan Chen

School of Physics, Peking University

北京大学

Tropospheric Ozone

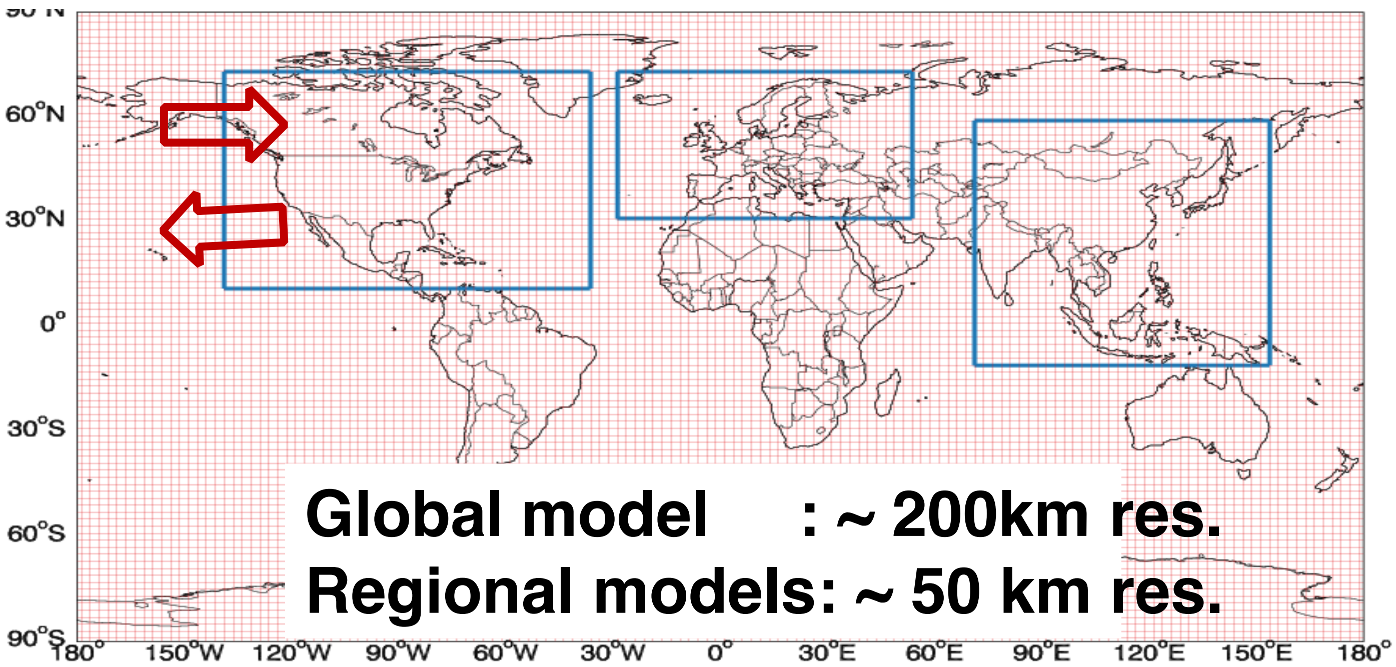


- Computational cost limits resolution.
- Limited resolution misrepresents small scale processes.
- Small scale variations in chemistry and emissions cause large errors.

Yingying Yan et al.

Two-way Coupled Model

- High resolution regional nested simulations.
- Differences can be transported outside nested domains and accumulate over species lifetime.



Global model : ~ 200km res.
Regional models: ~ 50 km res.

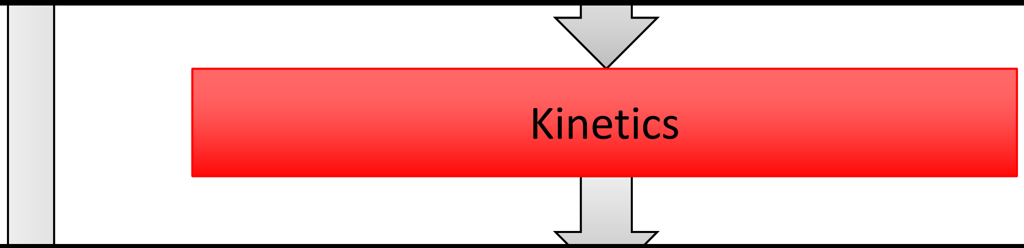
Yingying Yan et al.

Improvements in Tropospheric Simulation

	Global Model	Two-way Model	‘Observation’
OH (10^5 cm^{-3})	11.8	11.2 (– 5% *)	10.4 – 10.9
MCF lifetime (yr)	5.58	5.87 (+ 5.2%)	6.0 – 6.3
CH_4 lifetime (yr)	9.63	10.12 (+ 5.1%)	10.2 – 11.2
O_3 (DU)	34.5	31.5 (– 8.7%)	31.1 ± 3 (OMI/MLS)
O_3 (Tg)	384	348 (– 9.4% #)	
NOx (TgN)	0.169	0.176 (+ 4.1%)	
CO (Tg)	359	398 (+ 10.8% &)	
NMVOC (TgC)	10.1	10.2	

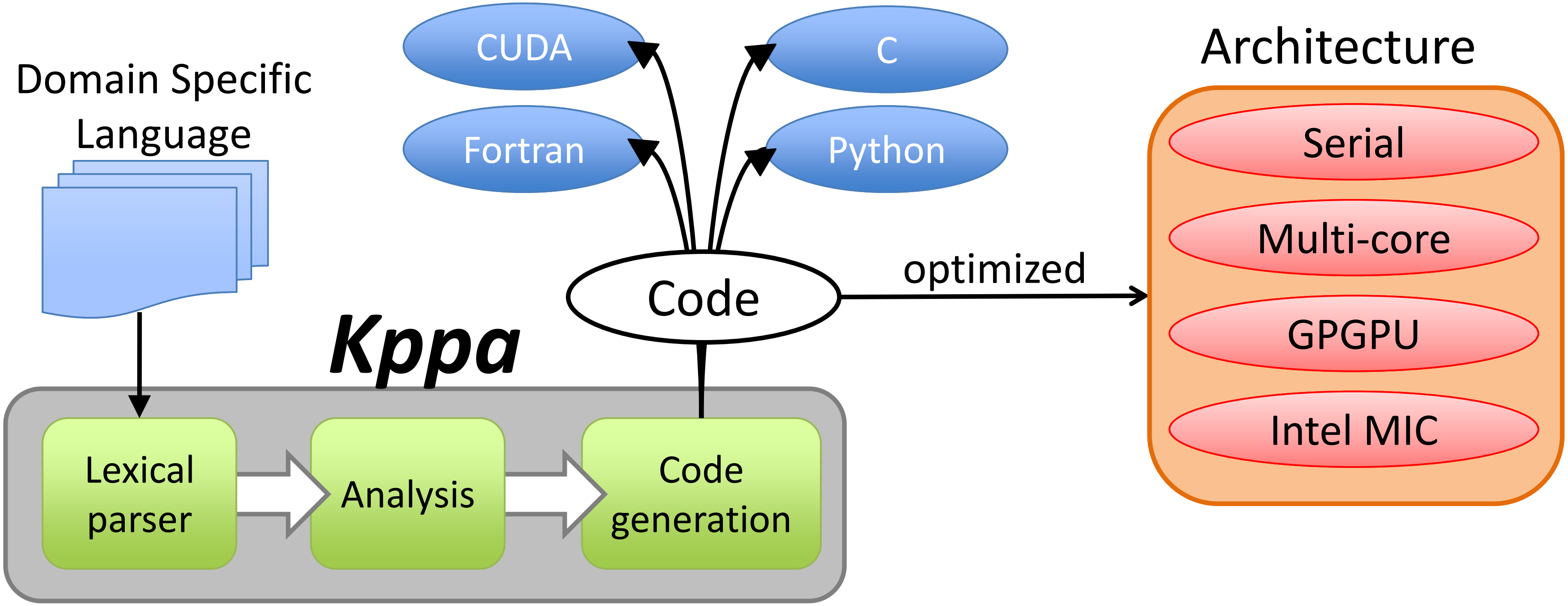
Yingying Yan et al.

How do we accelerate kinetics?



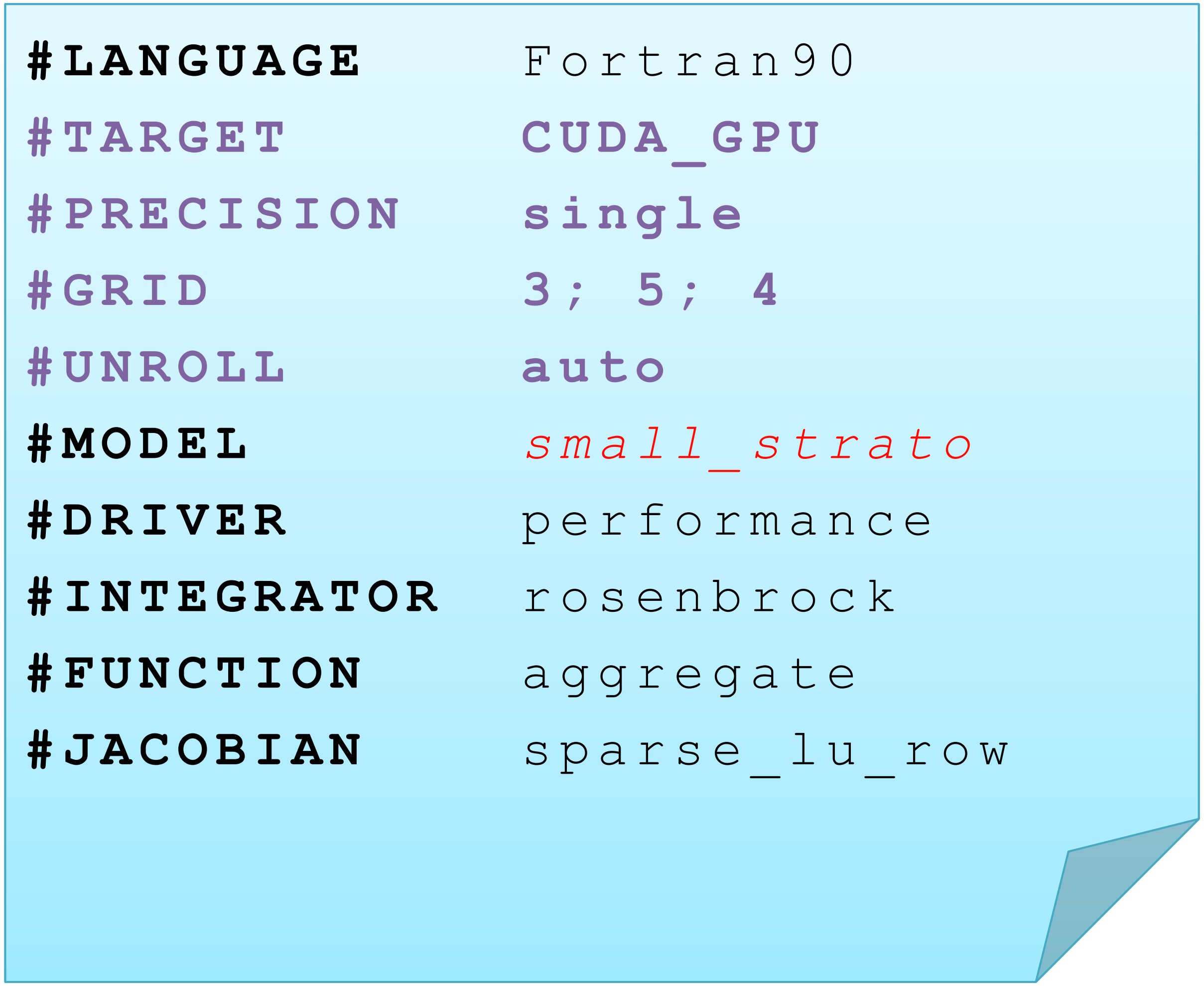
Kinetics

Kppa: The Kinetic PreProcessor Accelerated



Kppa’s Domain Specific Language

KPP Language[*] with extensions for target hardware, optimization parameters, precision, etc.



```
#LANGUAGE       Fortran90
#TARGET         CUDA_GPU
#PRECISION      single
#GRID           3; 5; 4
#UNROLL         auto
#MODEL          small_strato
#DRIVER         performance
#INTEGRATOR     rosenbrock
#FUNCTION       aggregate
#JACOBIAN       sparse_lu_row
```

small.kppa

[*] V. Damian, A. Sandu, M. Damian, F. Potra, and G.R. Carmichael: *The Kinetic PreProcessor KPP -- A Software Environment for Solving Chemical Kinetics*, Computers and Chemical Engineering, Vol. 26, No. 11, p. 1567-1579, 2002.

Kppa’s Domain Specific Language

Mechanism definition is **pure KPP Language*** for backwards compatibility

```
#DEFVAR
    O   = O;   // Oxygen atomic ground state
    O1D = O;   // Oxygen atomic excited state
    O3  = O + O + O;   // Ozone
    NO  = N + O;       // Nitric oxide
    NO2 = N + O + O;   // Nitrogen dioxide
#DEFFIX
    M   = O + O + N + N;   // Generic molecule
    O2  = O + O;           // Molecular oxygen
#EQUATIONS
  O2    + hv = 2O          : 2.643E-10f *SUN*SUN*SUN;
  O     + O2 = O3          : 8.018E-17;
  O3    + hv = O    + O2   : 6.120E-04f * SUN;
  O     + O3 = 2O2         : (1.576E-15);
  O3    + hv = O1D + O2    : (1.070E-03f) * SUN*SUN;
  O1D   + M  = O    + M    : (7.110E-11);
  O1D   + O3 = 2O2         : (1.200E-10);
  NO    + O3 = NO2 + O2    : (6.062E-15);
  NO2   + O  = NO   + O2   : (1.069E-11);
  NO2   + hv = NO   + O    : (1.289E-02f) * SUN;
```

small_strato.def

* V. Damian, A. Sandu, M. Damian, F. Potra, and G.R. Carmichael: *The Kinetic PreProcessor KPP -- A Software Environment for Solving Chemical Kinetics*, Computers and Chemical Engineering, Vol. 26, No. 11, p. 1567-1579, 2002.

Mass Action Kinetics

n concentrations: $y_i \in \vec{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$

R reaction rates: $k_j \in \vec{k} = \begin{pmatrix} k_1 \\ \vdots \\ k_R \end{pmatrix}$

Stoichiometric coefficients: $s^{-}{}_{i,j}$ and $s^{+}{}_{i,j}$

The jth reaction (r_j): $$\sum s^{-}_{i,j} y_i \xrightarrow{k_j(t)} \sum s^{+}_{i,j} y_i$$

Reaction velocity: $$\omega_j(t,y) = k_j(t) \square_{i=1}^{n} \; y_i^{s^{-}{}_{i,j}}$$

Coupled stiff ODE system

Time evolution of y: $$\frac{d}{dt}\vec{y} = (S^{+} - S^{-})\vec{\omega}(t,y) = S\vec{\omega}(t,\vec{y}) = f(t,\vec{y})$$

Large sparse matrices

N-stage Rosenbrock Solver

- Outperforms backwards differentiation formulas (GEAR)
- Jacobian generally inseparable
  - Solver cannot be parallelized
  - BLAS operations within solver can be parallelized

```
Initialize $k(t,y)$ from starting concentrations and meteorology ($\rho$, $t$, $q$, $p$)
Initialize time variables $t \leftarrow t_{start}$, $h \leftarrow 0.1 \times (t_{end} - t_{start})$
While $t \le t_{end}$
    $Fcn_0 \leftarrow Fcn \leftarrow f(t,y)$
    $Jac_0 \leftarrow J(t,y)$
    $G \leftarrow$ LU_DECOMP$(\frac{1}{h\gamma} - Jac_0)$
    For $s \leftarrow 1,2,\ldots,n$
        Compute $Stage_s$ from $Fcn$ and $Stage_{1 \ldots (s\text{-}1)}$
        Solve for $Stage_s$ implicitly using $G$
        Update $k$(t,y) with meteorology ($\rho$, $t$, $q$, $p$)
        Update $Fcn$ from $Stage_{1 \ldots s}$
    Compute $Y_{new}$ from $Stage_{1 \ldots s}$
    Compute error term $E$
    If $E > \delta$ then discard iteration, reduce $h$, restart
    Otherwise, $t \leftarrow t + h$ and proceed to next step
Finish : Result in $Y_{new}$
```

N-stage Rosenbrock Solver

```
Initialize $k(t,y)$ from starting concentrations and meteorology $(\rho, t, q, p)$
Initialize time variables $t$ □ $t_{start}$, $h$ □ $0.1$ □ $(t_{end} - t_{start})$
While $t \le t_{end}$
    $Fcn_0$ □ $Fcn$ □ $f(t,y)$
    $Jac_0$ □ $J(t,y)$
    $G$ □ LU_DECOMP$(\frac{1}{h\gamma} - Jac_0)$
    For $s$ □ $1,2,$□$,n$
        Compute $Stage_s$ from $Fcn$ and $Stage_{1□(s-1)}$
        Solve for $Stage_s$ implicitly using $G$
        Update $k(t,y)$ with meteorology $(\rho, t, q, p)$
        Update $Fcn$ from $Stage_{1□s}$
    Compute $Y_{new}$ from $Stage_{1□s}$
    Compute error term $E$
    If $E$ □ $\delta$ then discard iteration, reduce $h$, restart
    Otherwise, $t$ □ $t + h$ and proceed to next step
Finish : Result in $Y_{new}$
```

Initial values of the function and its derivatives

Sparse LU decomposition

3 to 6 stage vector calculations

New solution and solution error

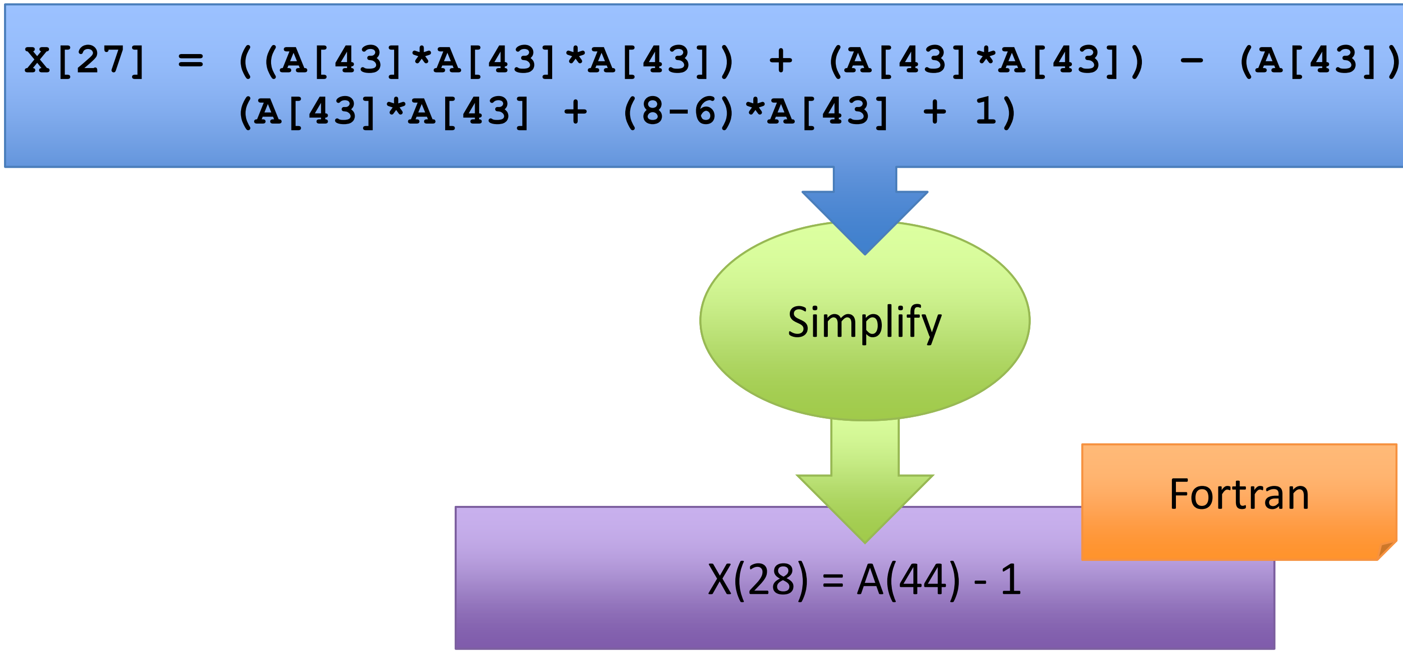
Sparse Operation Optimization

$$\begin{bmatrix} 4 & & A13 \\ & A22 & \\ & A32 & 0.75 \end{bmatrix} \begin{bmatrix} 1.5 \\ \\ B3 \end{bmatrix}$$

```
X[1] = 6 + A[13]*B[3];
X[2] = 0
X[3] = 0.75*B[3];
```

Simplify Before Generating Code

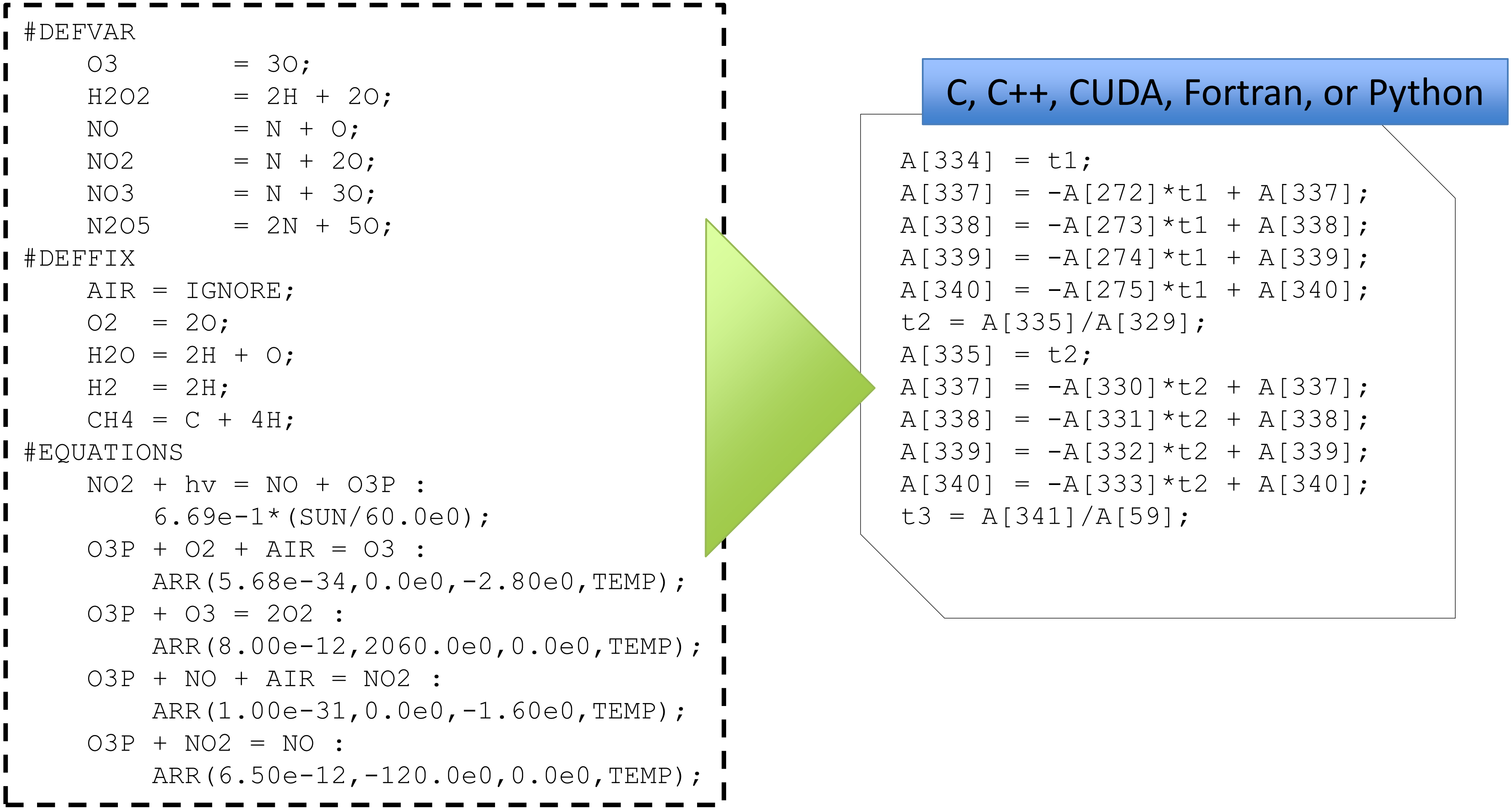
$$\frac{x^3 + x^2 - x - 1}{x^2 + 2x + 1} \square\, x - 1$$



```
X[27] = ((A[43]*A[43]*A[43]) + (A[43]*A[43]) - (A[43]) - (5+4)) /
        (A[43]*A[43] + (8-6)*A[43] + 1)
```

```
X(28) = A(44) - 1
```

Sparse Matrix/Vector Code Generation



```
#DEFVAR
    O3       = 3O;
    H2O2     = 2H + 2O;
    NO       = N + O;
    NO2      = N + 2O;
    NO3      = N + 3O;
    N2O5     = 2N + 5O;
#DEFFIX
    AIR = IGNORE;
    O2  = 2O;
    H2O = 2H + O;
    H2  = 2H;
    CH4 = C + 4H;
#EQUATIONS
    NO2 + hv = NO + O3P :
        6.69e-1*(SUN/60.0e0);
    O3P + O2 + AIR = O3 :
        ARR(5.68e-34,0.0e0,-2.80e0,TEMP);
    O3P + O3 = 2O2 :
        ARR(8.00e-12,2060.0e0,0.0e0,TEMP);
    O3P + NO + AIR = NO2 :
        ARR(1.00e-31,0.0e0,-1.60e0,TEMP);
    O3P + NO2 = NO :
        ARR(6.50e-12,-120.0e0,0.0e0,TEMP);
```

C, C++, CUDA, Fortran, or Python

```
A[334] = t1;
A[337] = -A[272]*t1 + A[337];
A[338] = -A[273]*t1 + A[338];
A[339] = -A[274]*t1 + A[339];
A[340] = -A[275]*t1 + A[340];
t2 = A[335]/A[329];
A[335] = t2;
A[337] = -A[330]*t2 + A[337];
A[338] = -A[331]*t2 + A[338];
A[339] = -A[332]*t2 + A[339];
A[340] = -A[333]*t2 + A[340];
t3 = A[341]/A[59];
```

Where can we parallelize?

```
Initialize $k(t,y)$ from starting concentrations and meteorology $(\rho, t, q, p)$
Initialize time variables $t$ □ $t_{start}$, $h$□ $0.1$□$(t_{end} - t_{start})$
While $t \le t_{end}$
    $Fcn_0$ □ $Fcn$ □ $f(t,y)$
    $Jac_0$ □ $J(t,y)$
    $G$□ LU_DECOMP$(\frac{1}{h\gamma} - Jac_0)$
    For $s$ □ $1,2,$□ $,n$
        Compute $Stage_s$ from $Fcn$ and $Stage_{1}$□ $_{(s\text{-}1)}$
        Solve for $Stage_s$ implicitly using $G$
        Update $k$(t,y) with meteorology $(\rho, t, q, p)$
        Update $Fcn$ from $Stage_{1}$□ $_s$
    Compute $Y_{new}$ from $Stage_{1}$□ $_s$
    Compute error term $E$
    If $E$ □$\delta$ then discard iteration, reduce $h$, restart
    Otherwise, $t$□ $t + h$ and proceed to next step
Finish : Result in $Y_{new}$
```

In general, the solver cannot be parallelized

BLAS operations can be parallelized

Many solver instances on the whole system model grid

Vectorized n-stage Rosenbrock solver

Vector element 1

```
Initialize $k(t,y)$ from starting concentrations and meteorology ($\rho$, $t$, $q$, $p$)
Initialize time variables $t$ □ $t_{start}$, $h$ □ $0.1$□$(t_{end} - t_{start})$
While $t \le t_{end}$
    $Fcn_0$ □ $Fcn$ □ $f(t,y)$
    $\text{Jac}_0$ □ $J(t,y)$
    $G$ □ LU_DECOMP($\frac{1}{h\gamma} - Jac_0$)
    For $s$ □ $1,2,$□$,n$
        Compute $Stage_s$ from $Fcn$ and $Stage_{1□(s\text{-}1)}$
        Solve for $Stage_s$ implicitly using $G$
        Update $k$(t,y) with meteorology ($\rho$, $t$, $q$, $p$)
        Update $Fcn$ from $Stage_{1□s}$
    Compute $Y_{new}$ from $Stage_{1□s}$
    Compute error term $E$
    If $E$ □ $\delta$ then discard iteration, reduce $h$, restart
    Otherwise, $t$ □ $t+h$ and proceed to next step
Finish : Result in $Y_{new}$
```

…

Vector element N

```
Initialize $k(t,y)$ from starting concentrations and meteorology ($\rho$, $t$, $q$, $p$)
Initialize time variables $t$ □ $t_{start}$, $h$ □ $0.1$□$(t_{end} - t_{start})$
While $t \le t_{end}$
    $Fcn_0$ □ $Fcn$ □ $f(t,y)$
    $\text{Jac}_0$ □ $J(t,y)$
    $G$ □ LU_DECOMP($\frac{1}{h\gamma} - Jac_0$)
    For $s$ □ $1,2,$□$,n$
        Compute $Stage_s$ from $Fcn$ and $Stage_{1□(s\text{-}1)}$
        Solve for $Stage_s$ implicitly using $G$
        Update $k$(t,y) with meteorology ($\rho$, $t$, $q$, $p$)
        Update $Fcn$ from $Stage_{1□s}$
    Compute $Y_{new}$ from $Stage_{1□s}$
    Compute error term $E$
    If $E$ □ $\delta$ then discard iteration, reduce $h$, restart
    Otherwise, $t$ □ $t+h$ and proceed to next step
Finish : Result in $Y_{new}$
```

Vectorized n-stage Rosenbrock solver

```
Initialize $k(t,y)$ from starting concentrations and meteorology $(\rho,\ t,\ q,\ p)$
Initialize time variables $t \square\ t_{start},\ h \square\ 0.1 \square (t_{end} - t_{start})$
While $t \leq t_{end}$
    $Fcn_0 \square\ Fcn \square\ f(t,y)$
    $Jac_0 \square\ J(t,y)$
    $G \square$ LU_DECOMP$(\frac{1}{h\gamma} - Jac_0)$
    For $s \square\ 1,2,\square\ ,n$
        Compute $Stage_s$ from $Fcn$ and $Stage_{1\square\ (s\text{-}1)}$
        Solve for $Stage_s$ implicitly using $G$
        Update $k$(t,y) with meteorology $(\rho,\ t,\ q,\ p)$
        Update $Fcn$ from $Stage_{1\square\ s}$
    Compute $Y_{new}$ from $Stage_{1\square\ s}$
    Compute error term $E = \max(E_1, E_2, \square\ , E_{vn})$
    If $E \square \delta$ then discard iteration, reduce $h$, restart
    Otherwise, $t \square\ t + h$ and proceed to next step
Finish : Result in $Y_{new}$
```

Kppa Benefits

Performance

- Parallelize across multiple “grid cells”
- Simplify the code so fewer instructions are required
- Parallel BLAS operations
- Use all levels of memory
- Optimize for emerging architectures

Productivity

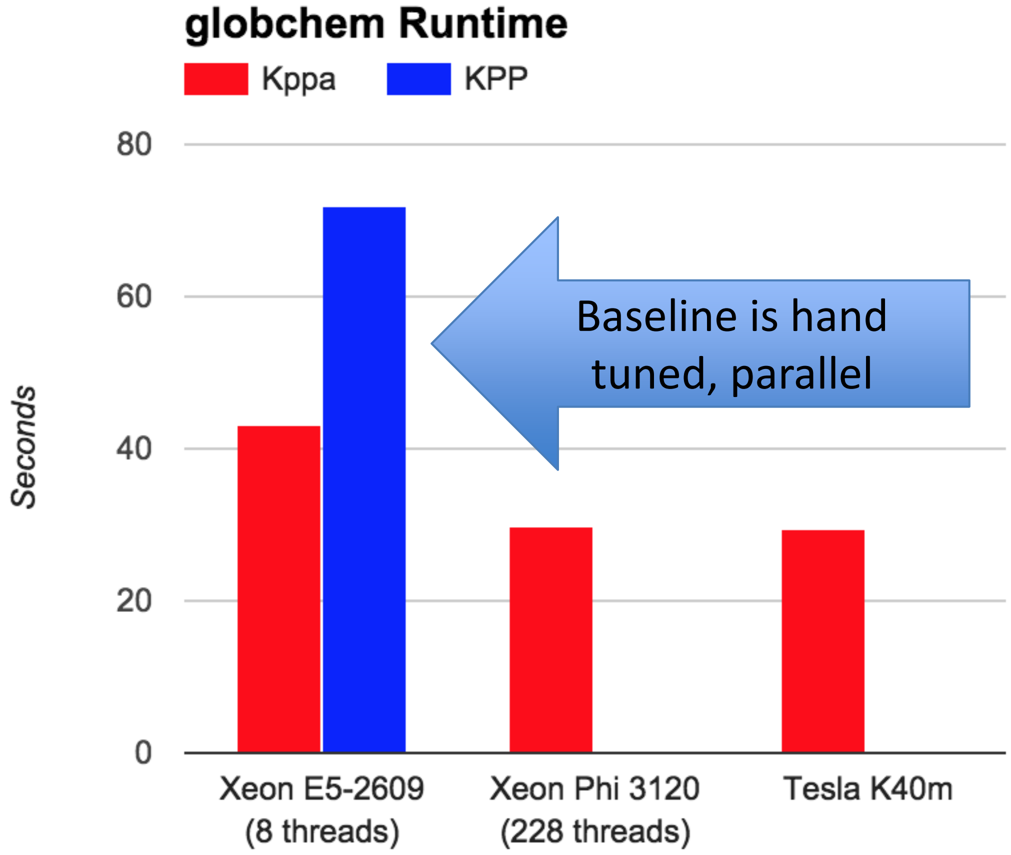
- High-level domain specific language as input
- Output in the most familiar or convenient language
- Regenerate mechanism code to target new hardware
- Extend and update mechanisms easily

Performance Results

- **Baseline:** hand-tuned KPP-generated code
  1. Use KPP to generate a serial code
  2. A skilled programmer parallelizes the code

- **Comparison**: unmodified Kppa-generated code
  - Same input file format as KPP
  - Minimal source code modifications

Kppa vs. Hand Tuned KPP



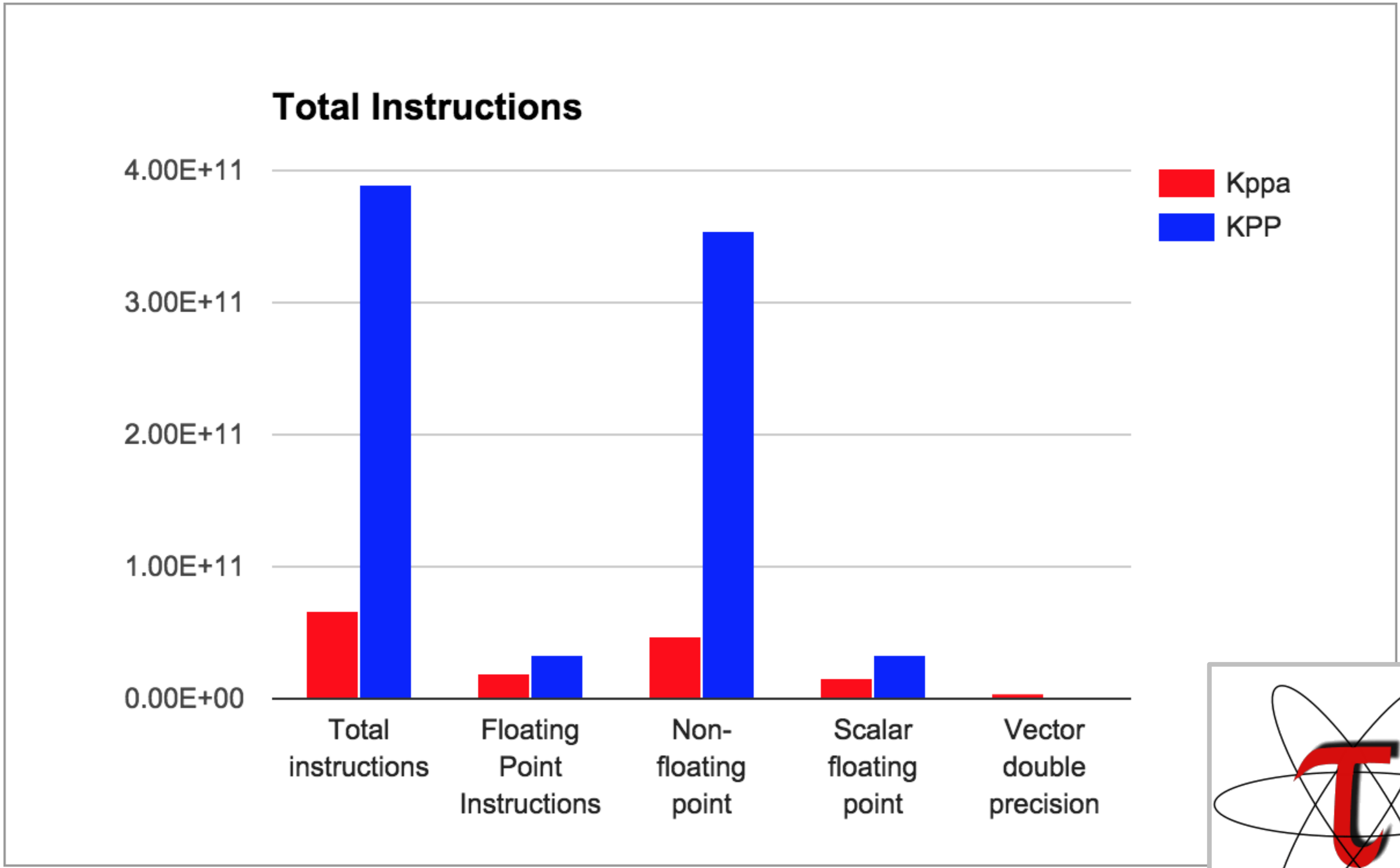
Baseline is hand tuned, parallel

Speedup:

- **Xeon Phi: 2.42**
- **NVIDIA K40: 2.45**
- **Xeon with OpenMP: 1.66**

Exact same hardware

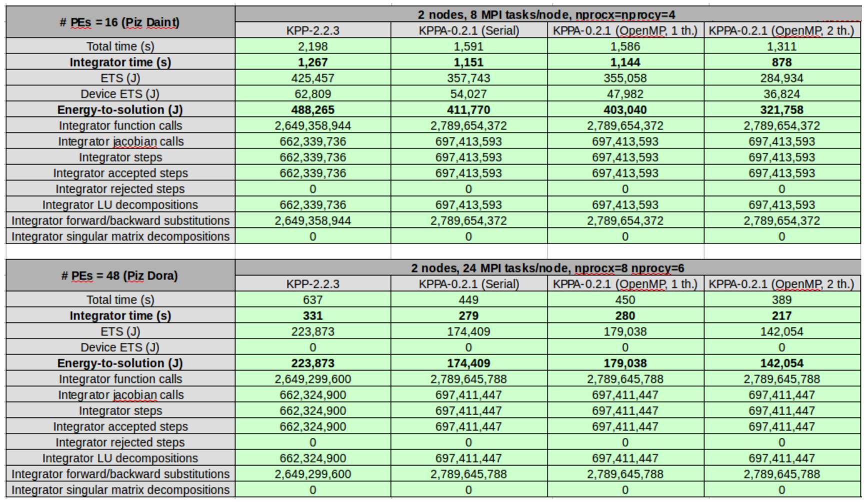
Fewer Operations, Faster Runtimes



TAU configured with PAPI and PDT

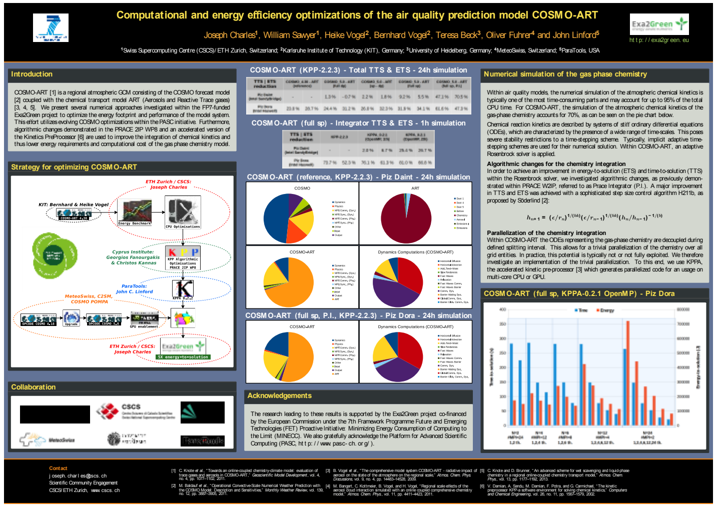
Ongoing: COSMO-ART

Joseph Charles, William Sawyer, Heike Vogel, Bernhard Vogel, Teresa Beck, Oliver Fuhrer, and John Linford. *Computational and Energy Efficiency Optimizations of the Air Quality Prediction Model COSMO-ART.* Poster, PASC'15, 1-3 June 2015.



# PEs = 16 (Piz Daint)	2 nodes, 8 MPI tasks/node, nprocx=nprocy=4			
	KPP-2.2.3	KPPA-0.2.1 (Serial)	KPPA-0.2.1 (OpenMP, 1 th.)	KPPA-0.2.1 (OpenMP, 2 th.)
Total time (s)	2,198	1,591	1,586	1,311
Integrator time (s)	**1,267**	**1,151**	**1,144**	**878**
ETS (J)	425,457	357,743	355,058	284,934
Device ETS (J)	62,809	54,027	47,982	36,824
Energy-to-solution (J)	**488,265**	**411,770**	**403,040**	**321,758**
Integrator function calls	2,649,358,944	2,789,654,372	2,789,654,372	2,789,654,372
Integrator jacobian calls	662,339,736	697,413,593	697,413,593	697,413,593
Integrator steps	662,339,736	697,413,593	697,413,593	697,413,593
Integrator accepted steps	662,339,736	697,413,593	697,413,593	697,413,593
Integrator rejected steps	0	0	0	0
Integrator LU decompositions	662,339,736	697,413,593	697,413,593	697,413,593
Integrator forward/backward substitutions	2,649,358,944	2,789,654,372	2,789,654,372	2,789,654,372
Integrator singular matrix decompositions	0	0	0	0

# PEs = 48 (Piz Dora)	2 nodes, 24 MPI tasks/node, nprocx=8 nprocy=6			
	KPP-2.2.3	KPPA-0.2.1 (Serial)	KPPA-0.2.1 (OpenMP, 1 th.)	KPPA-0.2.1 (OpenMP, 2 th.)
Total time (s)	637	449	450	389
Integrator time (s)	**331**	**279**	**280**	**217**
ETS (J)	223,873	174,409	179,038	142,054
Device ETS (J)	0	0	0	0
Energy-to-solution (J)	**223,873**	**174,409**	**179,038**	**142,054**
Integrator function calls	2,649,299,600	2,789,645,788	2,789,645,788	2,789,645,788
Integrator jacobian calls	662,324,900	697,411,447	697,411,447	697,411,447
Integrator steps	662,324,900	697,411,447	697,411,447	697,411,447
Integrator accepted steps	662,324,900	697,411,447	697,411,447	697,411,447
Integrator rejected steps	0	0	0	0
Integrator LU decompositions	662,324,900	697,411,447	697,411,447	697,411,447
Integrator forward/backward substitutions	2,649,299,600	2,789,645,788	2,789,645,788	2,789,645,788
Integrator singular matrix decompositions	0	0	0	0



Computational and energy efficiency optimizations of the air quality prediction model COSMO-ART

Joseph Charles[1], William Sawyer[1], Heike Vogel[2], Bernhard Vogel[2], Teresa Beck[3], Oliver Fuhrer[4] and John Linford[5]

[1]Swiss Supercomputing Centre (CSCS)/ETH Zurich, Switzerland; [2]Karlsruhe Institute of Technology (KIT), Germany; [3]University of Heidelberg, Germany; [4]MeteoSwiss, Switzerland; [5]ParaTools, USA

Exa2Green
http://exa2green.eu

Introduction

COSMO-ART [1] is a regional atmospheric GCM consisting of the COSMO forecast model [2] coupled with the chemical transport model ART (Aerosols and Reactive Trace gases) [3, 4, 5]. We present several numerical approaches investigated within the FP7-funded Exa2Green project to optimize the energy footprint and performance of the model system. This effort utilizes evolving COSMO optimizations within the PASC initiative. Furthermore, algorithmic changes demonstrated in the PRACE 2IP WP8 and an accelerated version of the Kinetics PreProcessor [6] are used to improve the integration of chemical kinetics and thus lower energy requirements and computational cost of the gas phase chemistry model.

Strategy for optimizing COSMO-ART

ETH Zurich / CSCS: Joseph Charles
KIT: Bernhard & Heike Vogel
COSMO-ART 4.30
Energy Benchmark
CPU Optimisations
Cyprus Institute: Georgios Fanourgakis & Christos Kannas
KPP
KPP Algorithmic Optimisations PRACE 2IP WP8
ParaTools: John C. Linford
KPPA 0.2.2
MeteoSwiss, C2SM, COSMO POMPA
OPCODE COSMO 4.18
Upgrade
OPCODE COSMO 5.0
GPU enablement
ETH Zurich / CSCS: Joseph Charles
Exa2Green
5X energy-to-solution

Collaboration

COSMO-ART (KPP-2.2.3) - Total TTS & ETS - 24h simulation

TTS | ETS reduction	COSMO_4.30_ART (reference)		COSMO_5.0_ART [illegible]		COSMO_5.0_ART [illegible]		COSMO_5.0_ART [illegible]		COSMO_5.0_ART (full sp, P.I.)	
Piz Daint (Intel SandyBridge)	-	-	1.3 %	-0.7 %	2.2 %	1.8 %	9.2 %	5.5 %	47.1 %	70.5 %
Piz Dora (Intel Haswell)	23.8 %	28.7 %	24.4 %	31.2 %	26.8 %	32.3 %	31.8 %	34.1 %	61.6 %	47.3 %

COSMO-ART (full sp) - Integrator TTS & ETS - 1h simulation

TTS | ETS reduction	KPP-2.2.3		KPPA 0.2.1 (OpenMP, 1th)		KPPA 0.2.1 (OpenMP, 2th)	
Piz Daint (Intel SandyBridge)	-	-	2.0 %	6.7 %	25.6 %	28.7 %
Piz Dora (Intel Haswell)	73.7 %	52.3 %	76.1 %	61.3 %	66.0 %	66.6 %

COSMO-ART (reference, KPP-2.2.3) - Piz Daint - 24h simulation

COSMO-ART (full sp, P.I., KPP-2.2.3) - Piz Dora - 24h simulation

Acknowledgements

The research leading to these results is supported by the Exa2Green project co-financed by the European Commission under the 7th Framework Programme Future and Emerging Technologies (FET) Proactive Initiative: Minimizing Energy Consumption of Computing to the Limit (MINECC). We also gratefully acknowledge the Platform for Advanced Scientific Computing (PASC, http://www.pasc-ch.org/).

Numerical simulation of the gas phase chemistry

Within air quality models, the numerical simulation of the atmospheric chemical kinetics is typically one of the most time-consuming parts and may account for up to 95% of the total CPU time. For COSMO-ART, the simulation of the atmospheric chemical kinetics of the gas-phase chemistry accounts for 70%, as can be seen on the pie chart below.

Chemical reaction kinetics are described by systems of stiff ordinary differential equations (ODEs), which are characterized by the presence of a wide range of time-scales. This poses severe stability restrictions to a time-stepping scheme. Typically, implicit adaptive time-stepping schemes are used for their numerical solution. Within COSMO-ART, an adaptive Rosenbrock solver is applied.

Algorithmic changes for the chemistry integration
In order to achieve an improvement in energy-to-solution (ETS) and time-to-solution (TTS) within the Rosenbrock solver, we investigated algorithmic changes, as previously demonstrated within PRACE W2IP, referred to as Prace Integrator (P.I.). A major improvement in TTS and ETS was achieved with a sophisticated step size control algorithm H211b, as proposed by Söderlind [2]:

$$h_{n+1} = (\epsilon/r_n)^{1/(bk)}(\epsilon/r_{n-1})^{1/(bk)}(h_n/h_{n-1})^{-1/(b)}$$

Parallelization of the chemistry integration
Within COSMO-ART the ODEs representing the gas-phase chemistry are decoupled during defined splitting interval. This allows for a trivial parallelization of the chemistry over all grid entities. In practice, this potential is typically not or not fully exploited. We therefore investigate an implementation of the trivial parallelization. To this end, we use KPPA, the accelerated kinetic pre-processor [3] which generates parallelized code for an usage on multi-core CPU or GPU.

COSMO-ART (full sp, KPPA-0.2.1 OpenMP) - Piz Dora

Contact
joseph.charles@scs.ch
Scientific Community Engagement
CSCS/ETH Zurich, www.cscs.ch

[1] C. Knote *et al.*, "Towards an online-coupled chemistry-climate model: evaluation of trace gases and aerosols in COSMO-ART," *Geoscientific Model Development*, vol. 4, no. 4, pp. 1077–1102, 2011.

[2] M. Baldauf *et al.*, "Operational Convective-Scale Numerical Weather Prediction with the COSMO Model: Description and Sensitivities," *Monthly Weather Review*, vol. 139, no. 12, pp. 3887–3905, 2011.

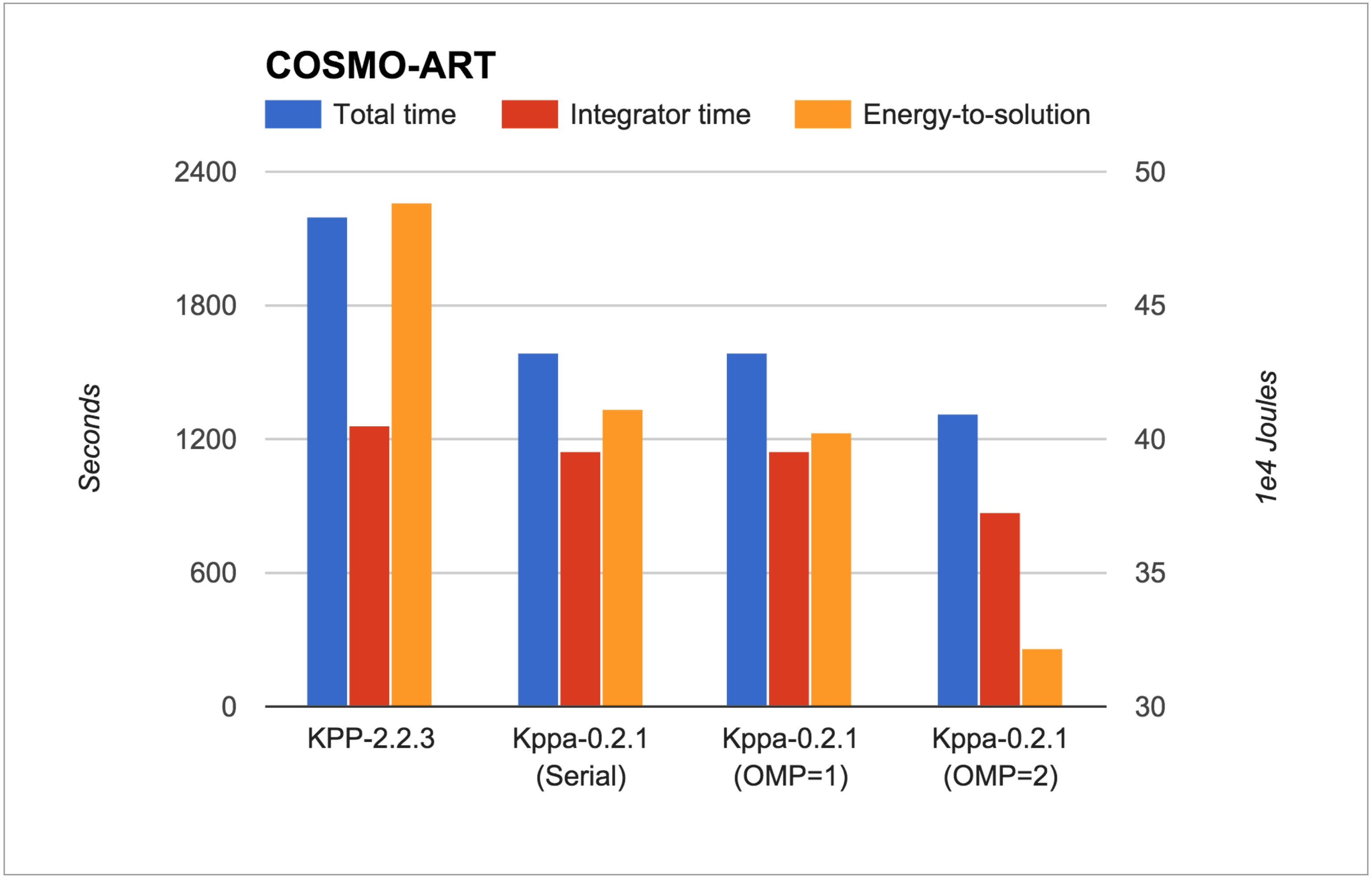
[3] B. Vogel *et al.*, "The comprehensive model system COSMO-ART - radiative impact of aerosol on the state of the atmosphere on the regional scale," *Atmos. Chem. Phys. Discussions*, vol. 9, no. 4, pp. 14483–14528, 2009.

[4] M. Bangert, C. Kottmeier, B. Vogel, and H. Vogel, "Regional scale effects of the aerosol cloud interaction simulated with an online coupled comprehensive chemistry model," *Atmos. Chem. Phys.*, vol. 11, pp. 4411–4423, 2011.

[5] C. Knote and D. Brunner, "An advanced scheme for wet scavenging and liquid-phase chemistry in a regional online-coupled chemistry transport model," *Atmos. Chem. Phys.*, vol. 13, pp. 1177–1192, 2013.

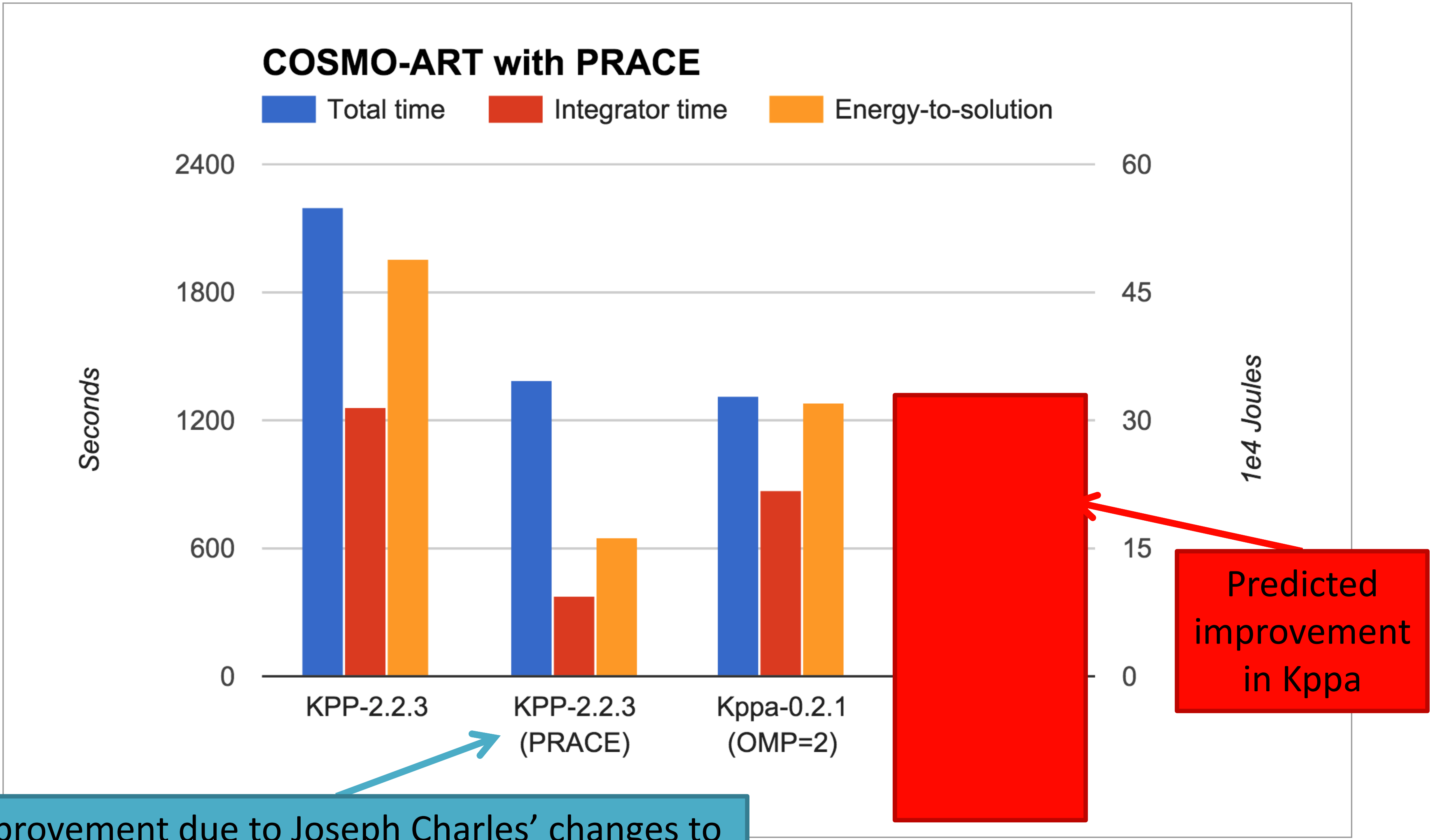
[6] V. Damian, A. Sandu, M. Damian, F. Potra, and G. Carmichael, "The kinetic preprocessor KPP-a software environment for solving chemical kinetics," *Computers and Chemical Engineering*, vol. 26, no. 11, pp. 1567–1579, 2002.

COSMO-ART Benchmarks



Joseph Charles, et al.

Next Steps: PRACE W2IP and H211b



Predicted improvement in Kppa

Improvement due to Joseph Charles' changes to integrator timestep calculation (H211b)

Joseph Charles, et al.

Kppa Performance Overview

- Automatically-generated code is 1.7 – 2.5x faster than hand-optimized parallel code (minutes vs. months)
- 22-30x faster than code from by competing tools (KPP)
- GEOS-Chem runtime reduced ~20%
  - Exact same hardware
  - No loss of precision or stability
- COSMO-ART runtime reduced ~30%
  - Exact same hardware
  - No loss of precision or stability

Next Steps

- Aerosols
- Master Chemical Mechanism (MCM)
  - Large mechanisms
- PRACE integrator for timestep adjustment
  - About 4x faster in COSMO-ART
- Apply Kppa code generation to new domains
  - Coupled PDT systems
  - Signal processing
  - Graph analysis (cyber security)

http://www.paratools.com/kppa

Downloads, tutorials, resources