November 16-21, 2014

A Tool for Chemical Kinetics Simulation on Accelerated Architectures

John C. Linford

ParaTools, Inc.

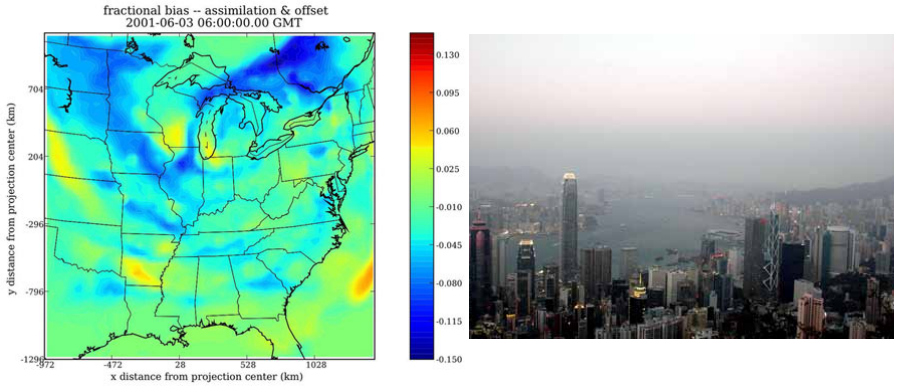
NASA Booth

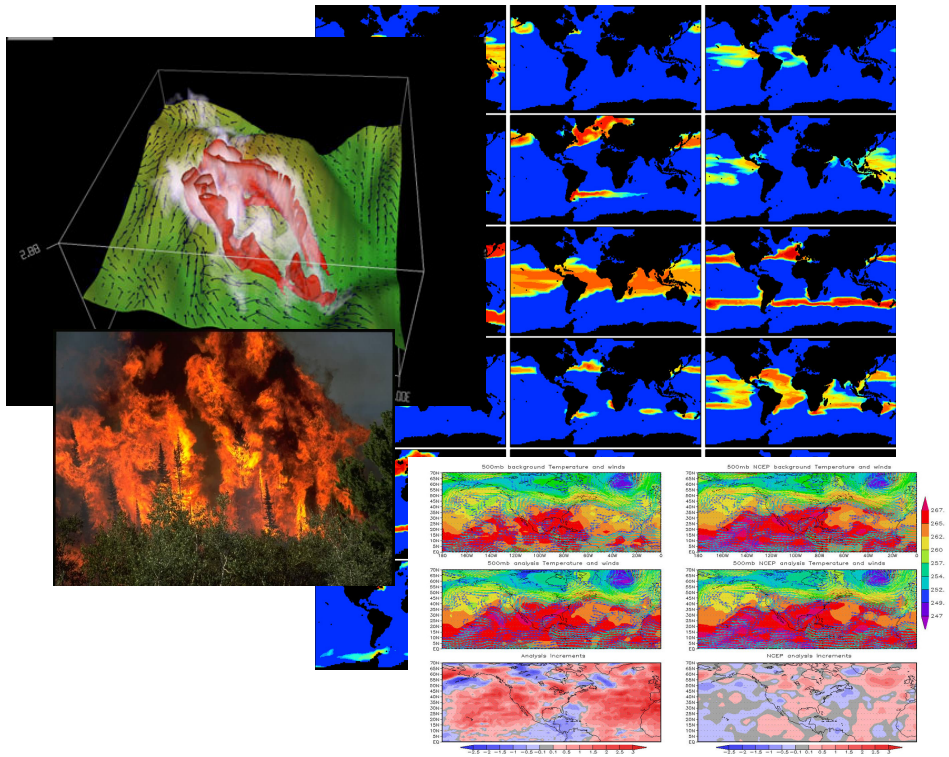
17 November 2014, SC'14

ParaTools

Chemical Kinetic Simulation

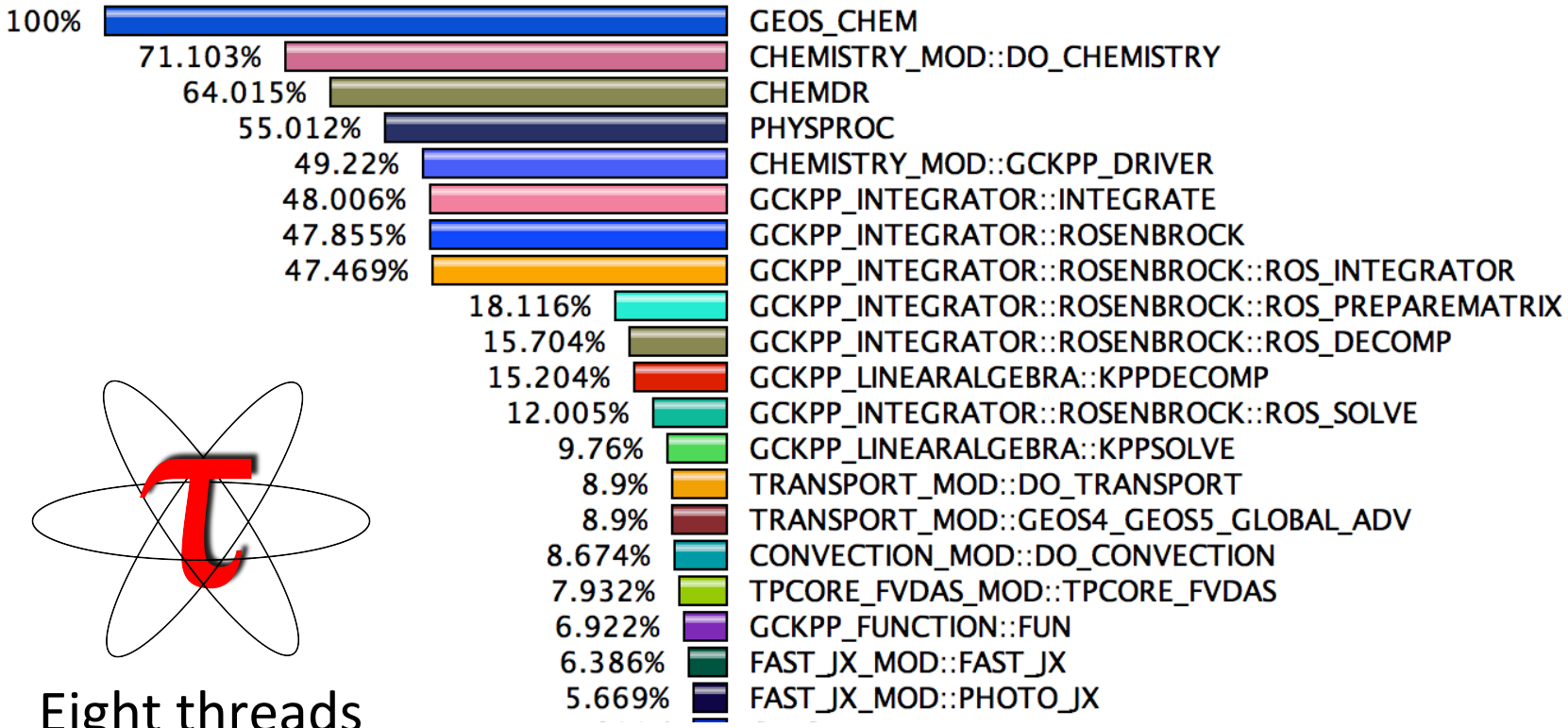
- Combustion
- Air and Water Quality
- Climate Change
- Wildfires
- Volcanic Eruptions
- Plastics Devolatilzation
- Microorganism Growth
- Cell Biology





70% of GEOS-Chem Runtime is Chemistry

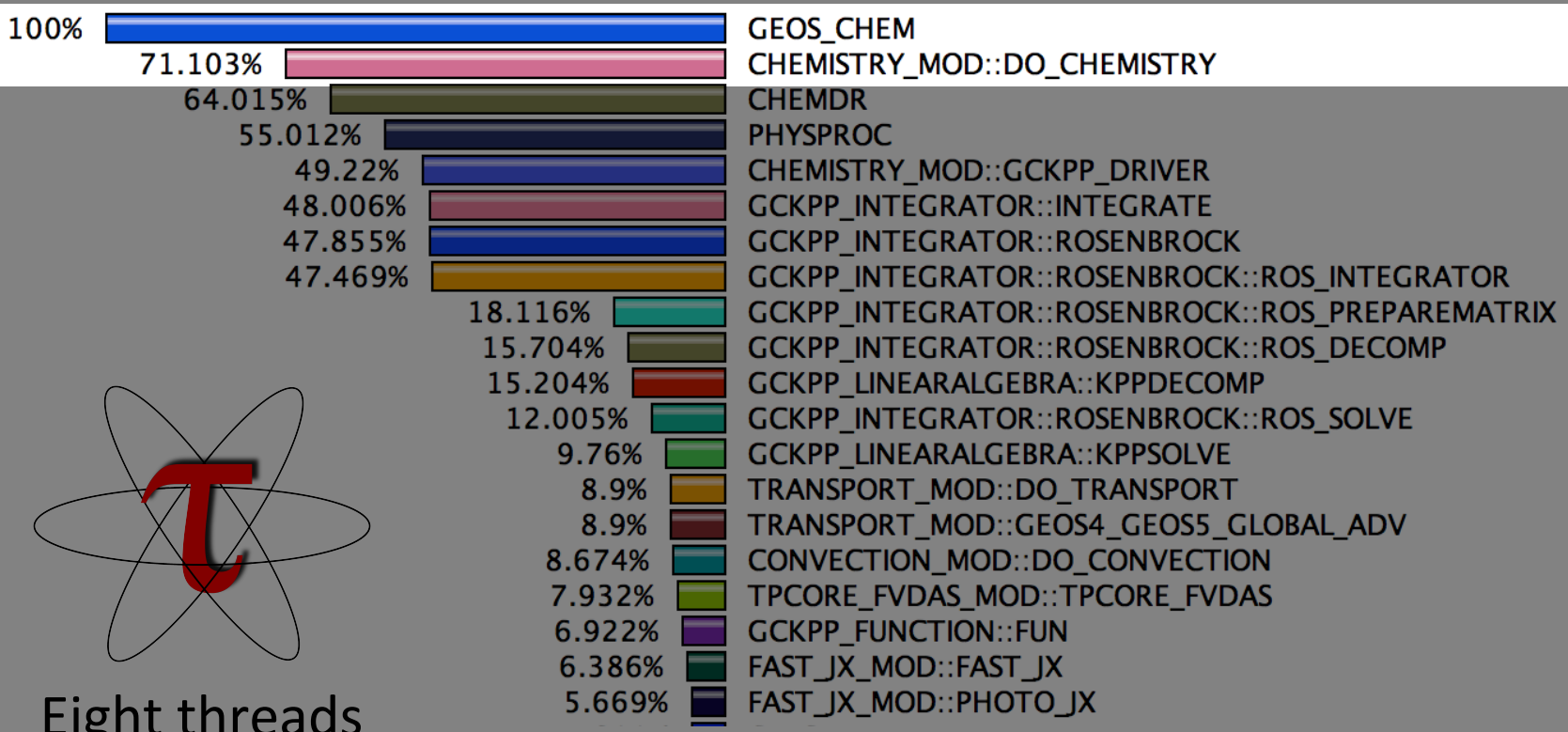
Metric: TIME
Value: Inclusive percent



Eight threads
Intel Core i7-4820K CPU
3.70GHz

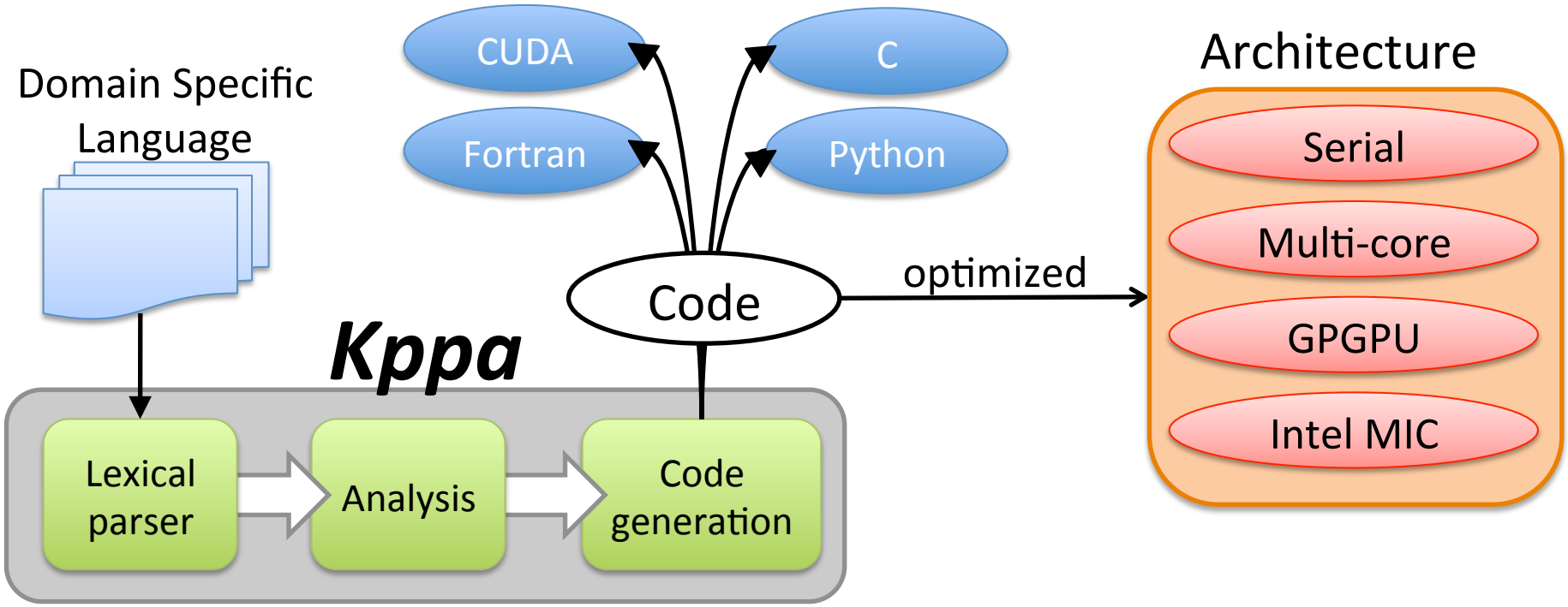
70% of GEOS-Chem Runtime is Chemistry

Metric: TIME
Value: Inclusive percent



Eight threads
Intel Core i7-4820K CPU
3.70GHz

Kppa: The Kinetic PreProcessor Accelerated



OpenMP

intel inside Xeon Phi

NVIDIA CUDA

Sparse Matrix/Vector Code Generation

```
<%
d_Decomp.begin()

piv = lang.Variable('piv', REAL,
                    'Row element divided by diagonal')
piv.declare()
%>
    ${size_t} idx = blockDim.x*blockIdx.x+threadIdx.x;
    if(idx < ${ncells32}) {
        ${A} += idx;
<%
lang.upindent()
for i in xrange(1, nvar):
    for j in xrange(crow[i], diag[i]):
        c = icol[j]
        d = diag[c]
        piv.assign(A[j*ncells32] / A[d*ncells32])
        A[j*ncells32].assign(piv)
...
%>
    }
<% d_Decomp.end() %>
```

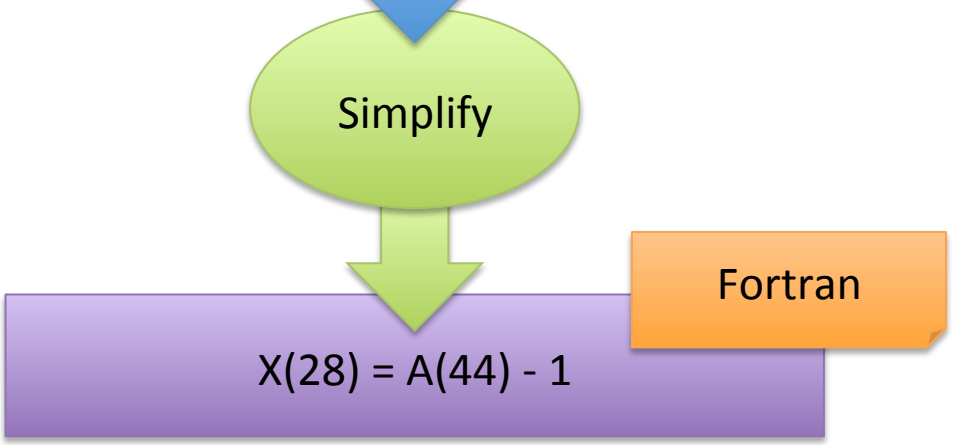
Generates C, C++, CUDA, Fortran, or Python

```
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];
```

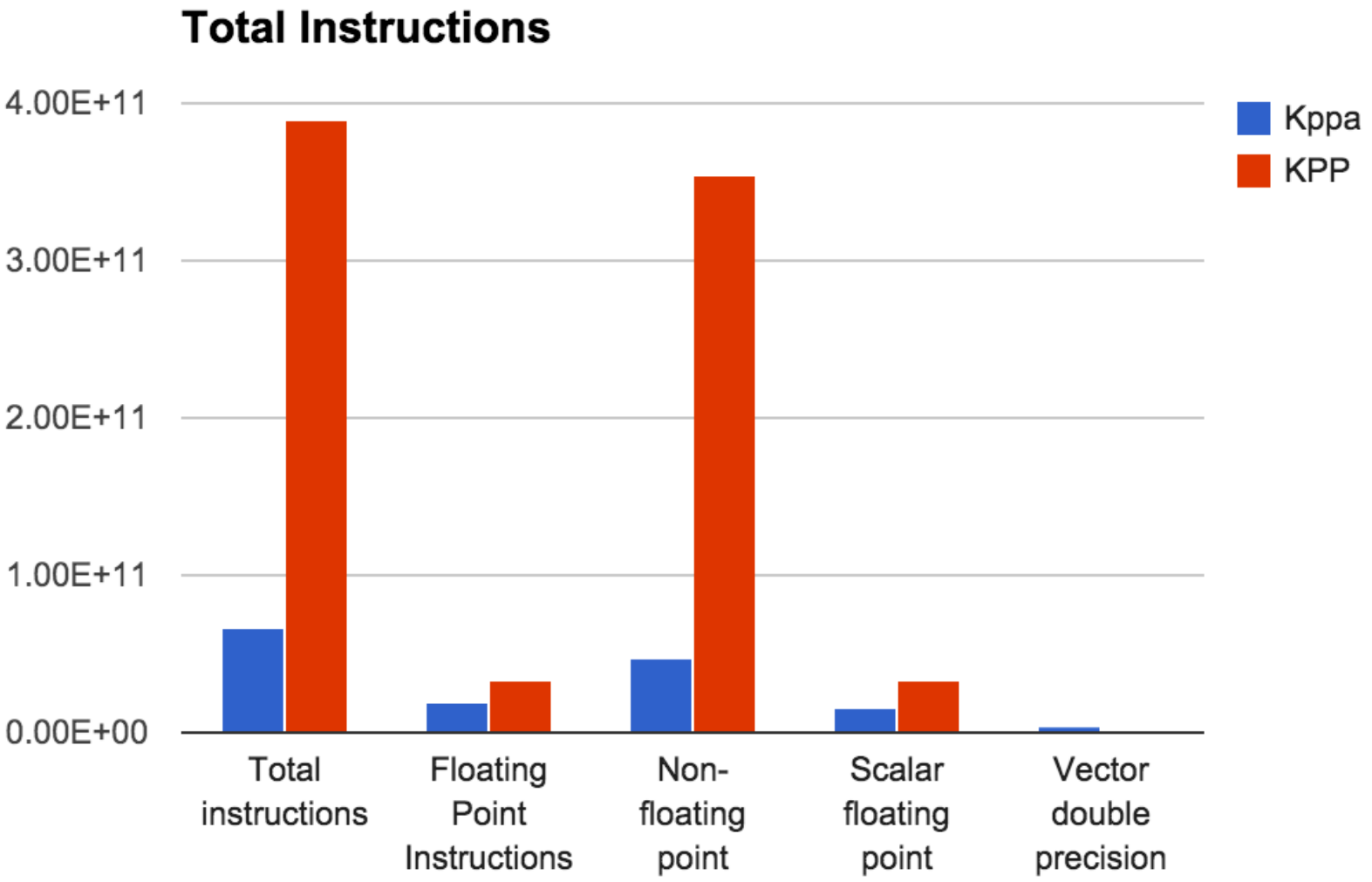
Simplify During Code Generation

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

```
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)
```



Fewer Operations, Faster Runtimes

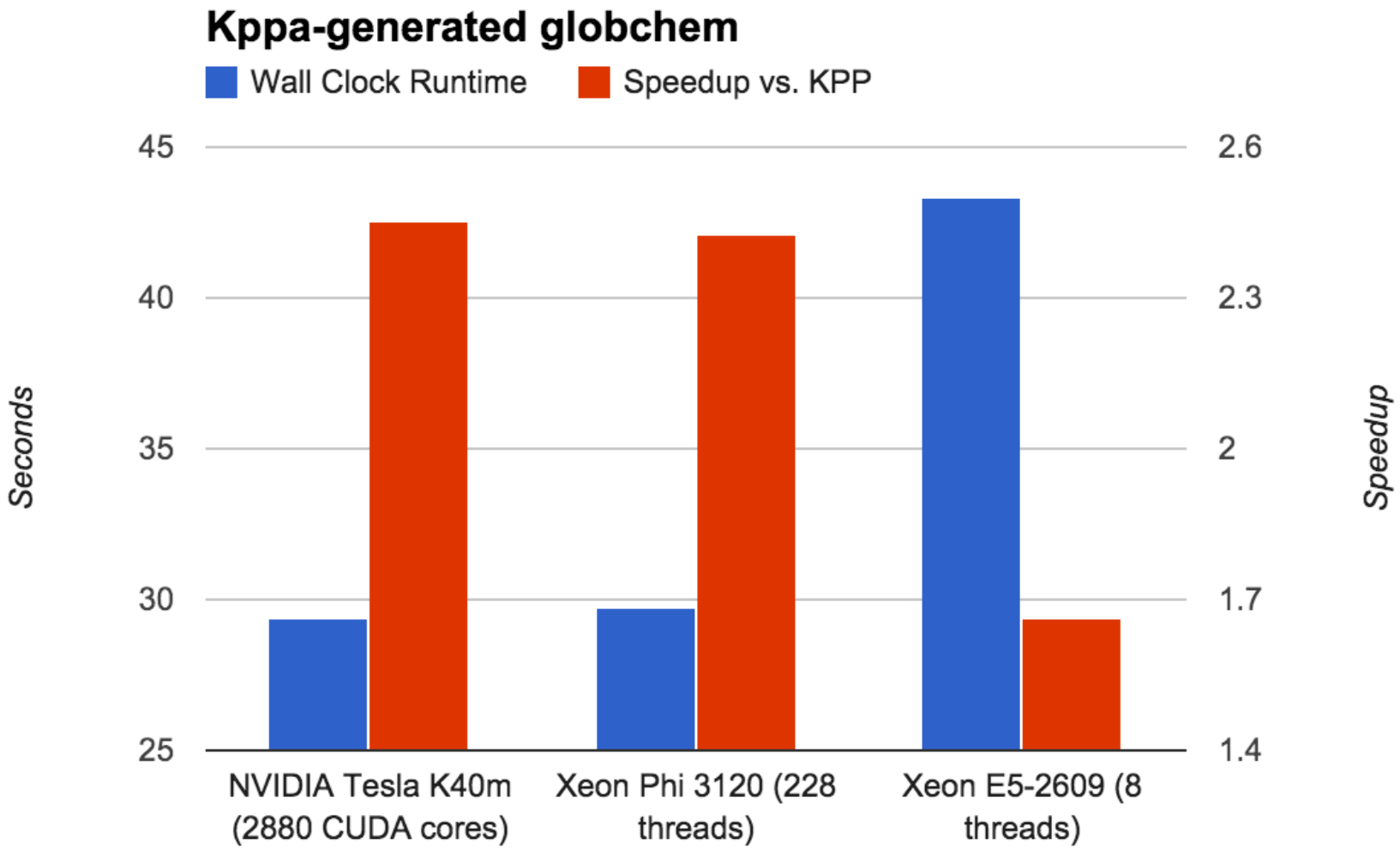


Performance Results

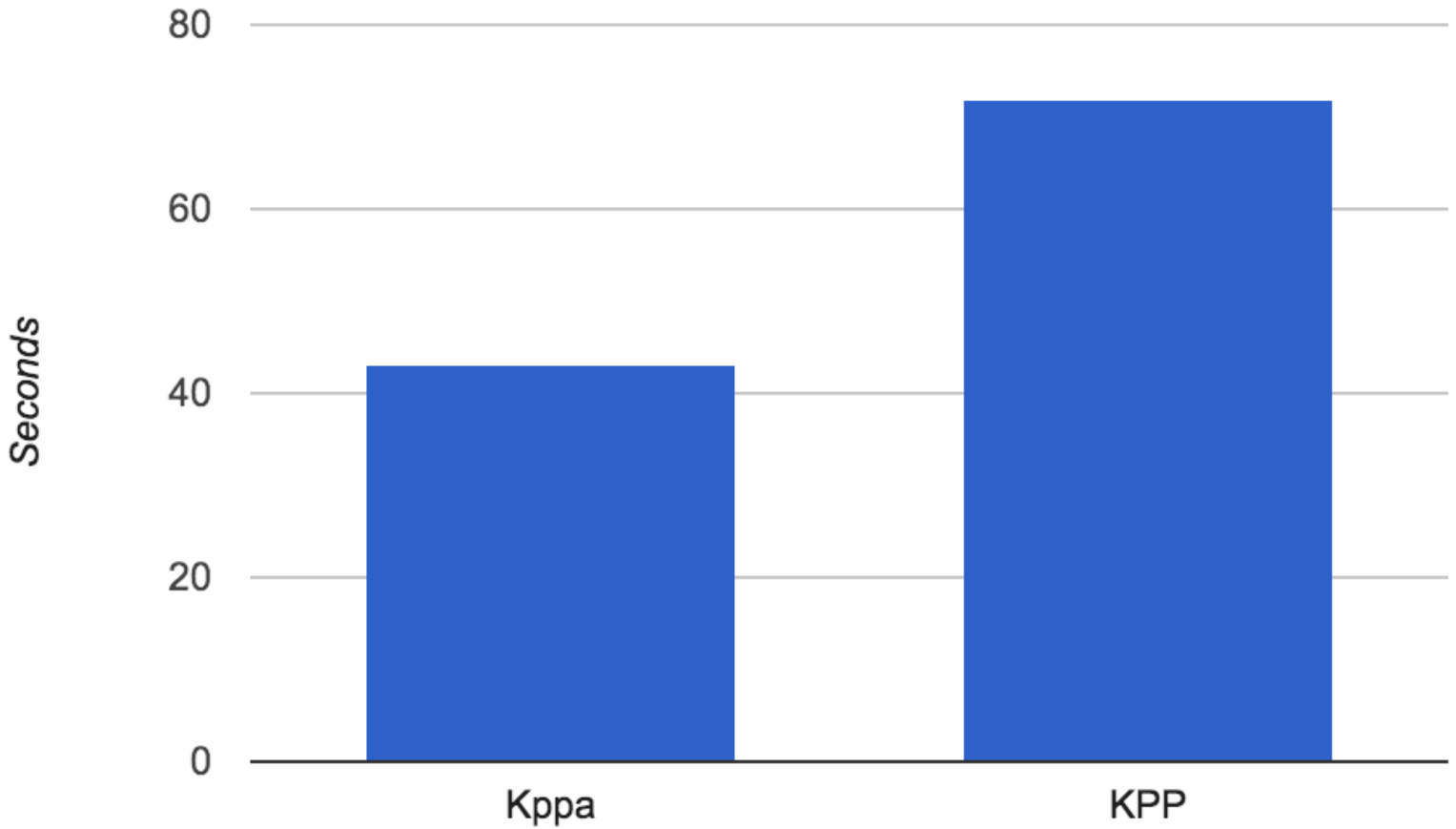
- **Baseline:** hand-tuned KPP-generated code
  1. Use KPP to generate a serial code
  2. A skilled programmer parallelizes the code
- **Comparison**: unmodified Kppa-generated code
  - Same input files as KPP
  - No modification to source
- **Results:**
  - **Xeon Phi: 140%** faster
  - **NVIDIA K40: 160%** faster
  - **Xeon with OpenMP: 60%** faster

Exact same hardware

Performance Results



Chemical Kernel Runtime



SBIR Project Status

- NASA SBIR, Phase I successful
- Phase II to implement:
  - Advanced reaction rate functions
  - Knight's Landing support
  - FPGA support
  - Auto-tuning source for hardware parameters
  - User interface and prototyping environment

http://www.paratools.com/kppa

Downloads, tutorials, resources