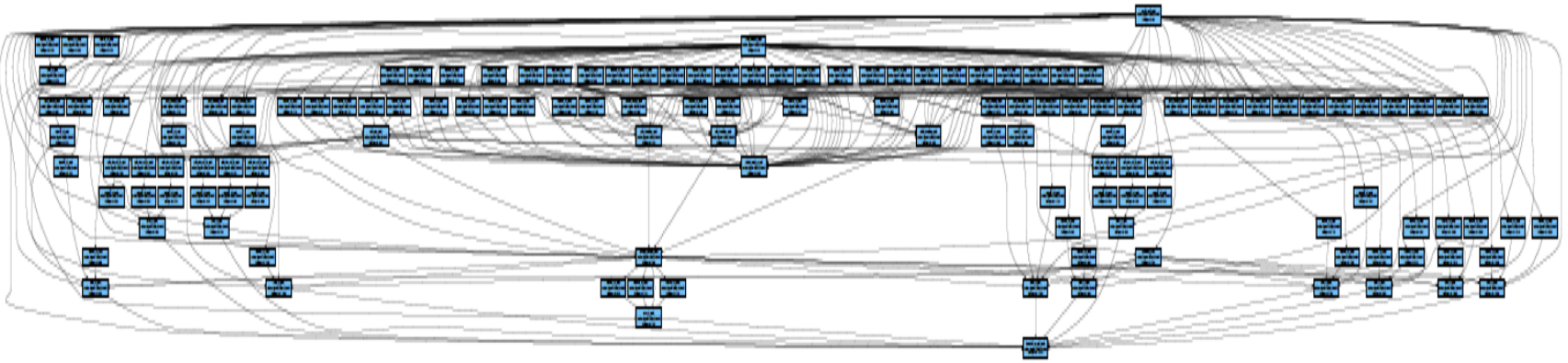
Automatic Parallelization and Transparent Fault Tolerance (project article)

Kei Davis, Dean Prichard, *David Ringo*, Loren Anderson, and Jacob Marks

Trends in Functional Programming, June 8-10, 2016

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

In Search of Automatic Parallelization
or at least automatic scheduling



1

S3D task dependency, combustion chemistry calculation

Simplest interesting chemistry, task graph much larger with more complex reactants. Schedule by hand?

[1] Courtesy Stanford Legion project

Scientific Computing in our Microcosm

Local evolution of scientific computing

- Serial Fortran programs
- MPI everywhere (inter- and intra-node)
- C, C++
- MPI+X, X is Pthreads, OpenMP, OpenCL, CUDA, etc.
- Parallel runtimes, e.g., Cilk++, Intel Threading Building Blocks, Stanford's Legion, etc.

We have a 'new' generation of scientific programmers, aka computational scientists, who have some understanding of meaning and virtue of *pure functional*, and even dabble in Haskell programming.

In a multi-100,000 line program, do not temporarily alter the global speed-of-light 'constant' variable.

Non-strictness/laziness Anathema to Parallelism

How to get around?

- Strictness analysis
- Bang patterns
- Par/pseq, other specifications
- Speculative evaluation
- ...
- *Strict(er) default semantics*

Here is a simple frame

- Here is the first bullet in a standard bulleted list, as is customary for a presentation like this
  - It includes some sub-bullets
  - Here they are
- Here is another bullet
- And here is another, with some math:

$$e^{i\theta} = \cos\theta + i\sin\theta$$

In Search of Automatic Parallelization
or at least automatic scheduling

- Here is a bulleted list that sits alongside a graphic
- With a second bullet item
- And a third
  - It can have sub-bullets too
  - Like this

insert caption here