SAND2005-3208C

Components, Software Scalability, and the CFRFS Problem Solving Environment

Benjamin Allan and Jaideep Ray

Sandia National Laboratories, Livermore

May 25, 2005

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.

Problem Statement

- Assumptions :
  - Writing software is a reality
  - Code complexity retards productivity – papers, timelines and proposal writing.
- Questions addressed :
  - Can a component-based software methodology lower code complexity? Provide numbers.
  - Is the administrative overhead (coding discipline, code design etc) worth it i.e. is productivity enhanced? Provide numbers.
  - What is the general structure of such a software development effort? Is there is a "hero programmer" who provides continuity?
- Questions not addressed :
  - Can a component-based software architecture be adopted for MPI-heavy scientific code ?
    - Yes it can; see CCA-related publications for examples.

Sandia National Laboratories

Outline of the talk

- Based on our experience while developing the Computational Facility for Reacting Flow Science toolkit
- What is the CFRFS toolkit ?
  - What does it do ? How general is it ?
  - What's so special about it ? How is it componentized ?
- How did componentization help tame software complexity ?
  - How granular is the componentization ? How complex are the interfaces between components ?
  - What was the make-up of the software team ?
- Did taming of complexity help ?
  - Did we publish ?
  - Did we trigger new fields of research ?

The CFRFS toolkit

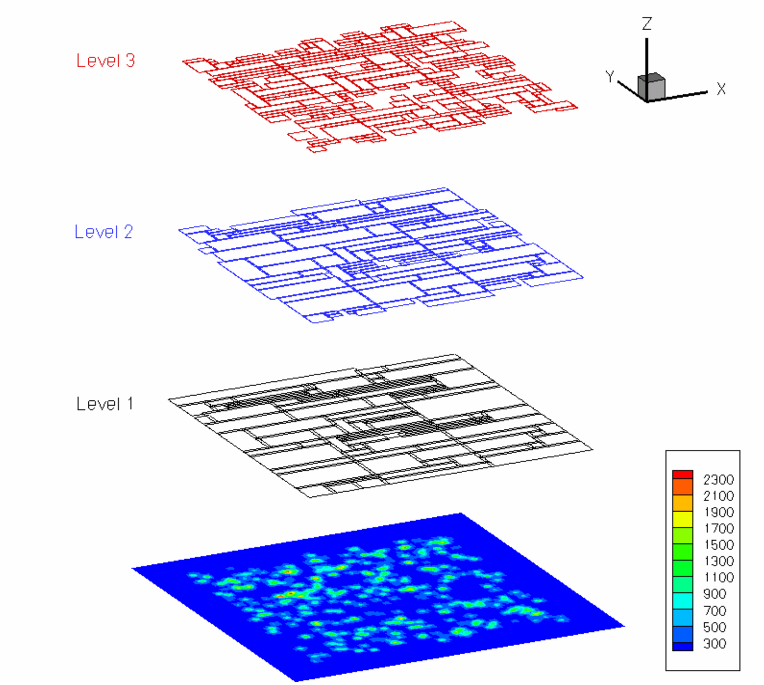
- A toolkit to perform simulations of lab-sized unsteady flames
- Solve the Navier-Stokes with detailed chemistry (~30 species, 200 reversible reactions)
- Consequently :
  - Disparity of length-scales :
    - use structured adaptively refined meshes
  - Disparity of timescales (transport v/s chemistry) :
    - use an operator-split construction and solve chemistry implicitly
    - Adaptive chemistry : use computational singular perturbation to identify low dimensional chemical manifolds and proceed along them

An example problem

- A coarse approx. to a flame.
- H_2-Air mixture; ignition via 3 hot-spots
- 9-species, 19 reactions, stiff chemistry

$$\frac{DY_i}{Dt} = \nabla.\alpha\nabla Y_i + \dot{w}_i$$

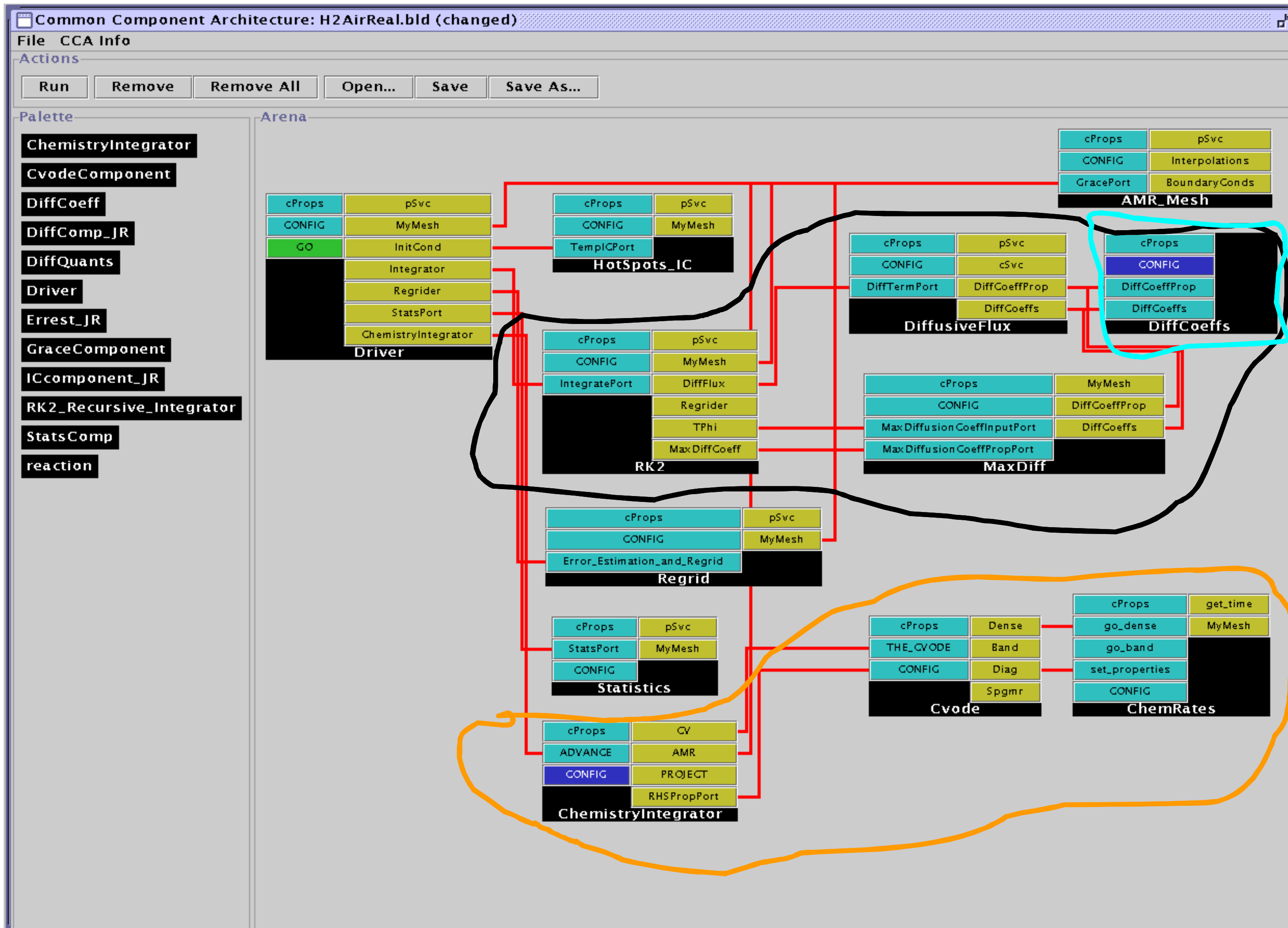
- 1cm X 1cm domain, 100x100 coarse mesh, finest mesh = 12.5 micron.
- Timescales : O(10ns) to O(10 microseconds)



What's so special about CFRFS ?

- Component-based, and still parallel.
  - Each functionality (e.g. integrator, mesh, diffusion-flux constructor etc) are implemented as peer components.
  - Completely independent of each other – can be mixed-and-matched
  - Components are compiled into dynamically loadable libraries
  - And are loaded into a framework at runtime and assembled into a working code
- Note :
  - There is *never* an **a.out**
  - Component A can be replaced by Component B in *the middle of a run*
  - And no, this does not involve a "*compile, link and keep the static executable in memory*" trick by the framework.
- Component architecture
  - Common Component Architecture
  - The framework used is CCAFFEINE (made in Sandia)

The code



Common Component Architecture: H2AirReal.bld (changed)
File CCA Info
Actions
Run
Remove
Remove All
Open...
Save
Save As...
Palette
ChemistryIntegrator
CvodeComponent
DiffCoeff
DiffComp_JR
DiffQuants
Driver
Errest_JR
GraceComponent
ICcomponent_JR
RK2_Recursive_Integrator
StatsComp
reaction
Arena
cProps pSvc CONFIG MyMesh GO InitCond Integrator Regrider StatsPort ChemistryIntegrator
Driver
cProps pSvc CONFIG MyMesh TemplCPort
HotSpots_IC
cProps pSvc CONFIG Interpolations GracePort BoundaryConds
AMR_Mesh
cProps pSvc CONFIG cSvc DiffTermPort DiffCoeffProp DiffCoeffs
DiffusiveFlux
cProps CONFIG DiffCoeffProp DiffCoeffs
DiffCoeffs
cProps pSvc CONFIG MyMesh IntegratePort DiffFlux Regrider TPhi MaxDiffCoeff
RK2
cProps MyMesh CONFIG DiffCoeffProp MaxDiffusionCoeffInputPort DiffCoeffs MaxDiffusionCoeffPropPort
MaxDiff
cProps pSvc CONFIG MyMesh Error_Estimation_and_Regrid
Regrid
cProps pSvc StatsPort MyMesh CONFIG
Statistics
cProps Dense THE_CVODE Band CONFIG Diag Spgmr
Cvode
cProps get_time go_dense MyMesh go_band set_properties CONFIG
ChemRates
cProps CV ADVANCE AMR CONFIG PROJECT RHSPropPort
ChemistryIntegrator

Research software scalability goals

- Generate scientific publications.
- Tame the complexity (cost!) of the software management:
  - Support new science and more complex models quickly.
  - Reuse verified parts of legacy codes.
  - Verify new codes (some from competitors).
  - Promote incremental and iterative development.
- Future-proofing:
  - Prevent sloppy, unmaintainable agglomerated code.
  - Cope with shifting HPC platforms regularly.
  - Enable lasting contributions from transient collaborators.

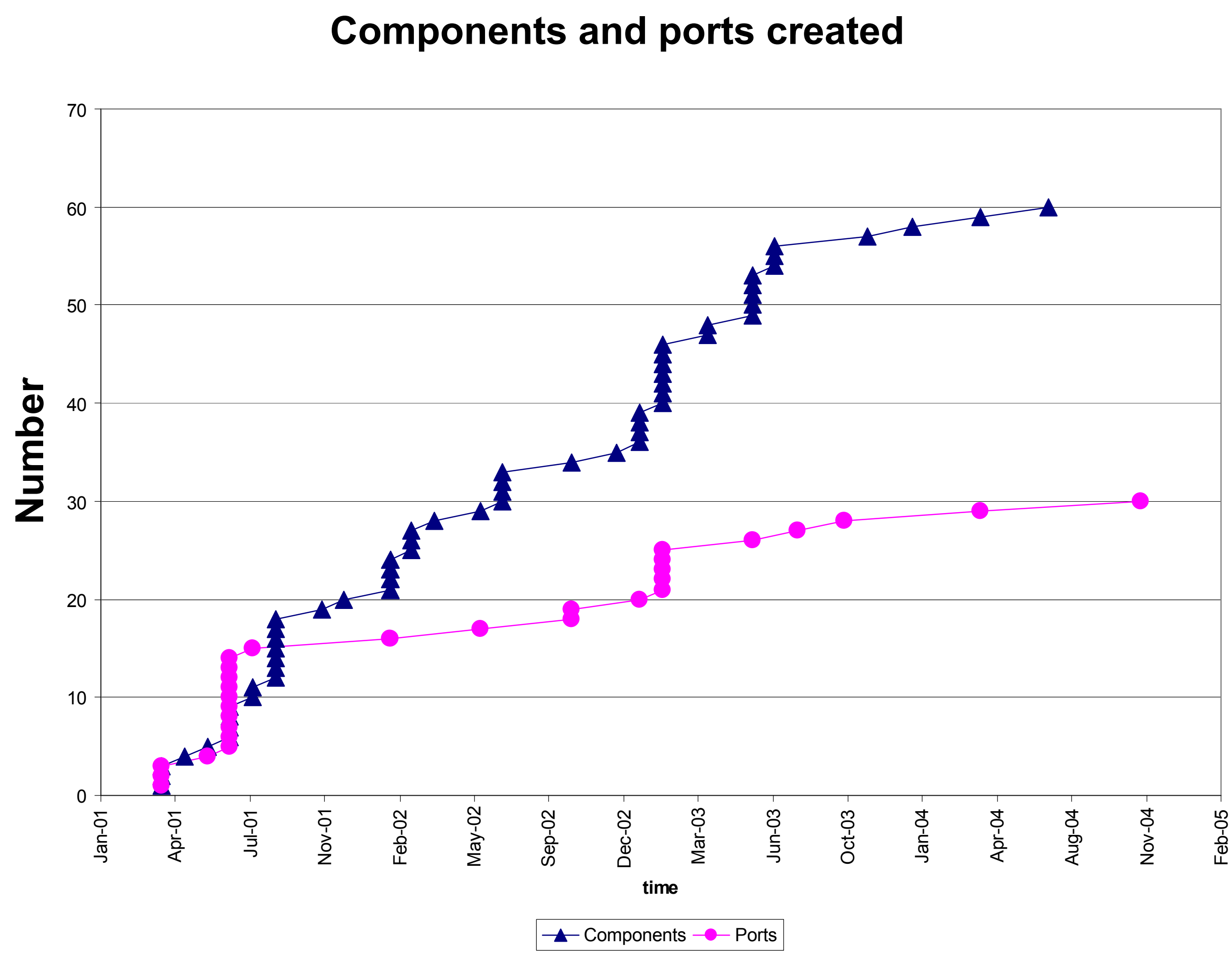
CFRFS PSE approach

- RFS applications are composed and parameterized in the generic CCA component management framework Ccaffeine using a GUI or scripting.
- Public function call interfaces (CCA Ports).
- Completely private implementation modules (CCA Components).
  - Parallel communication is a private implementation detail.
  - Each “global” variable or common block isolated in one component- no under the table accesses.
  - FORTRAN77 wrapped in C/C++.

Issues in CFD research software

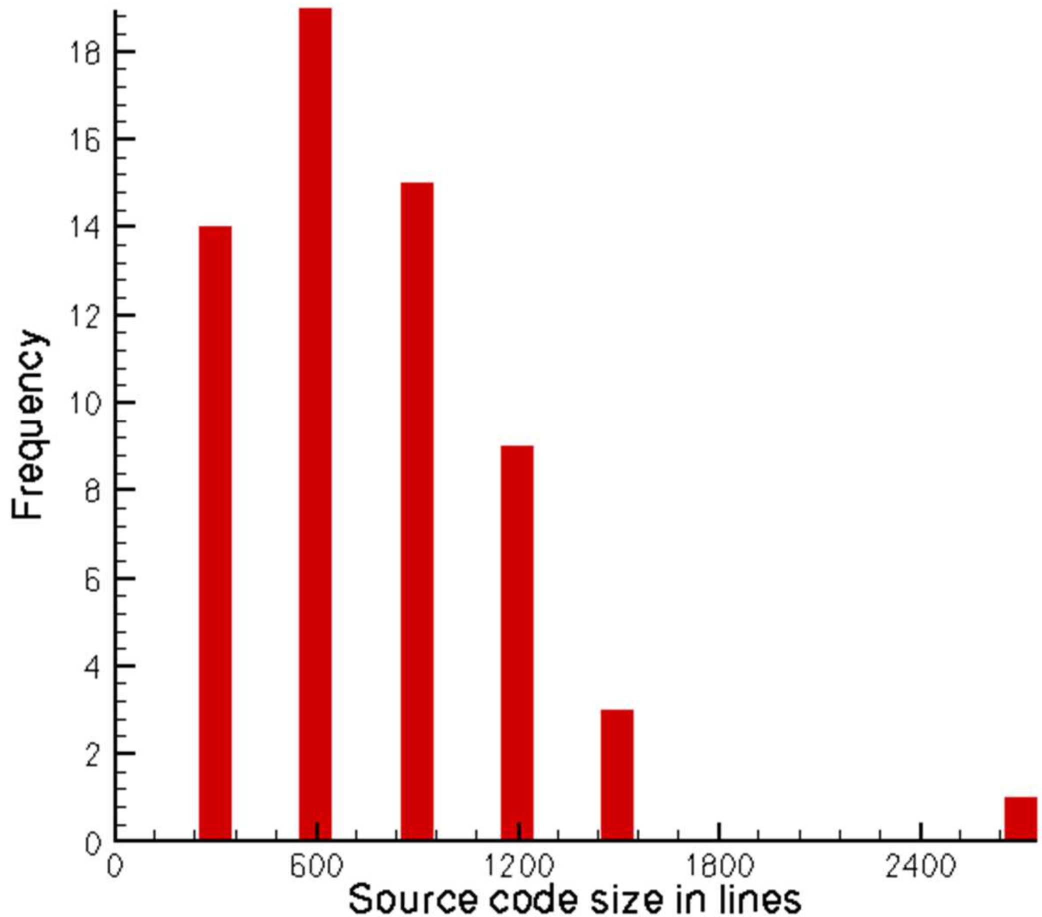
- Many external libraries must be used, not recreated.
- Diversity of development team skills and styles.
- Module boundary design and enforcement.
- Publication and funding activities compete with science and software for time.
- Software verification against other implementations.
- Changing needs and design requirements.
- Status :
  - Started in 2001
  - 61 components today, all peers, independent, mixed and matched for combustion and shock hydrodynamics
  - 7 external libraries
  - Diverse programming team skills and styles; 9 in all, including 3 summer students.

Scalability: PSE growth without rewrites



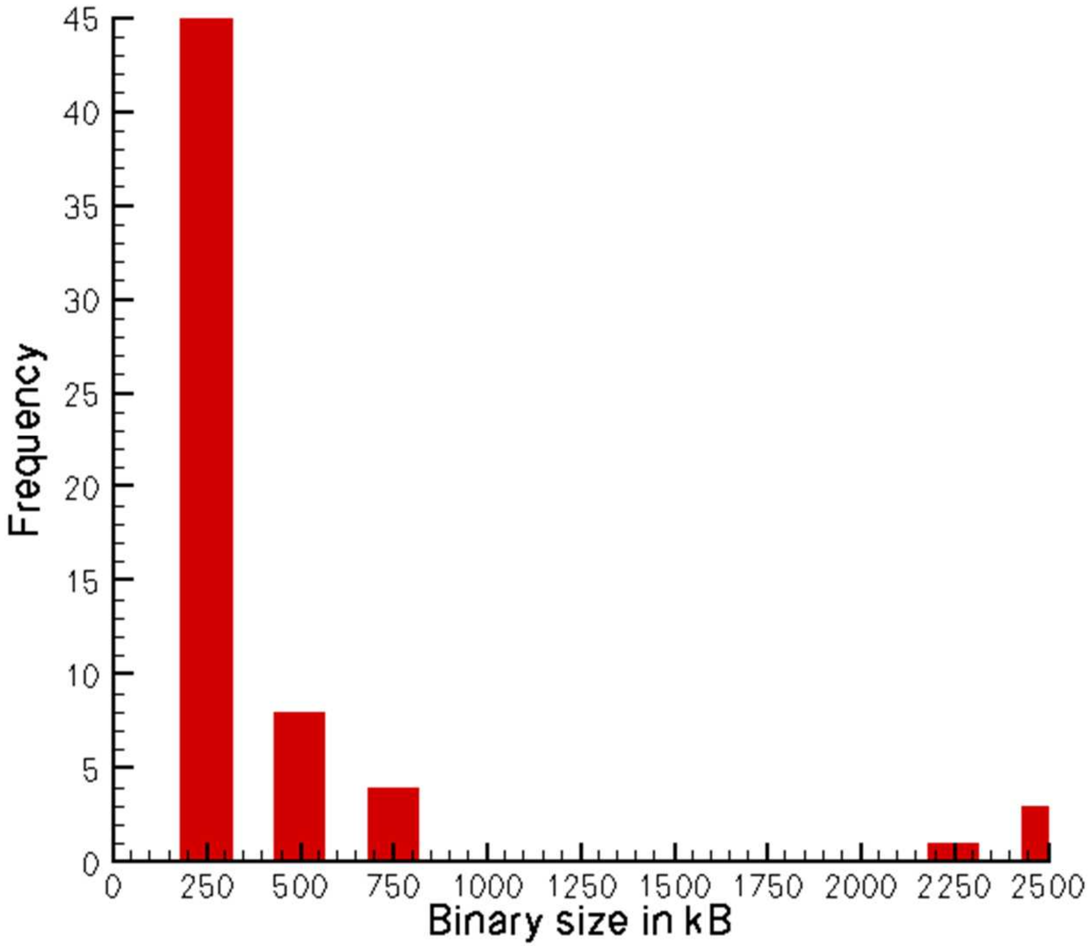
Sandia National Laboratories

Taming complexity: Component Source



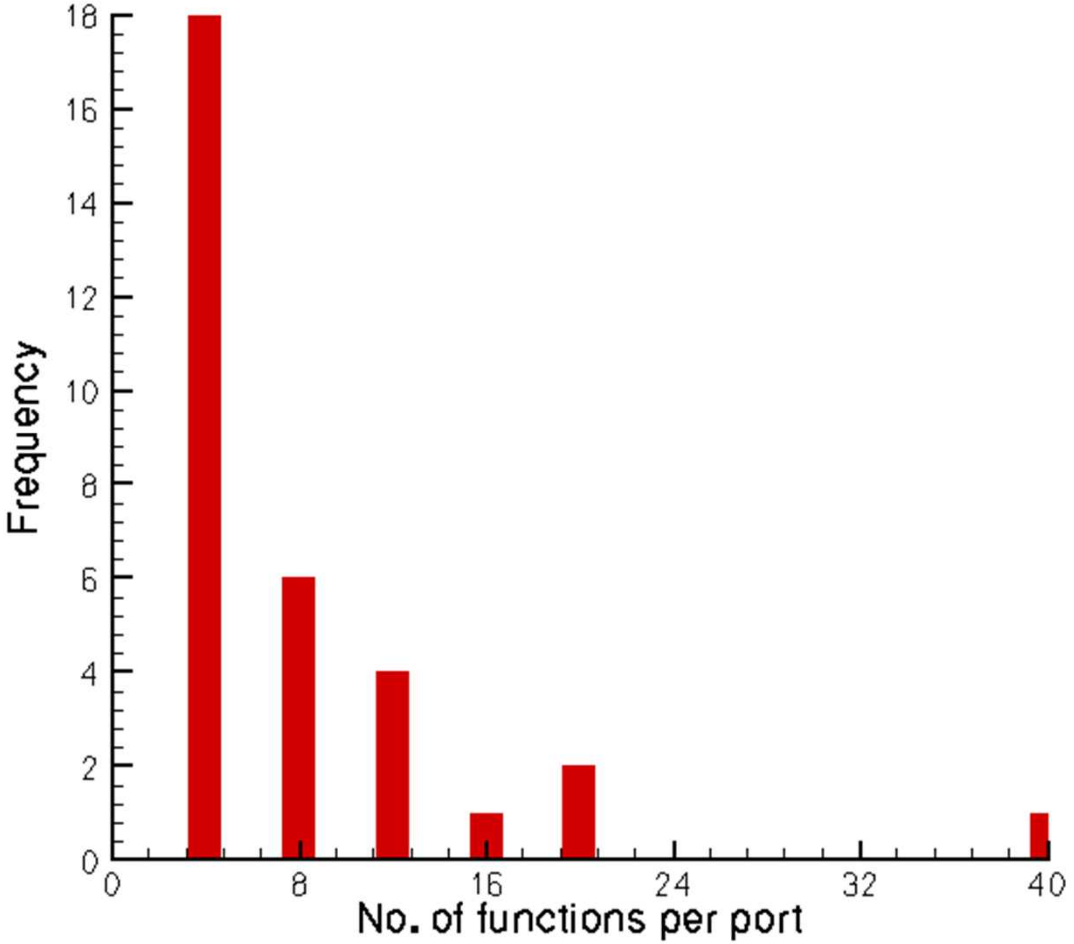
- ***Most components are < 1000 lines i.e they are easily maintainable***
- ***Grace, Chombo (parallel mesh libraries with load-balancers) are the largest.***

Taming complexity: Component sizes



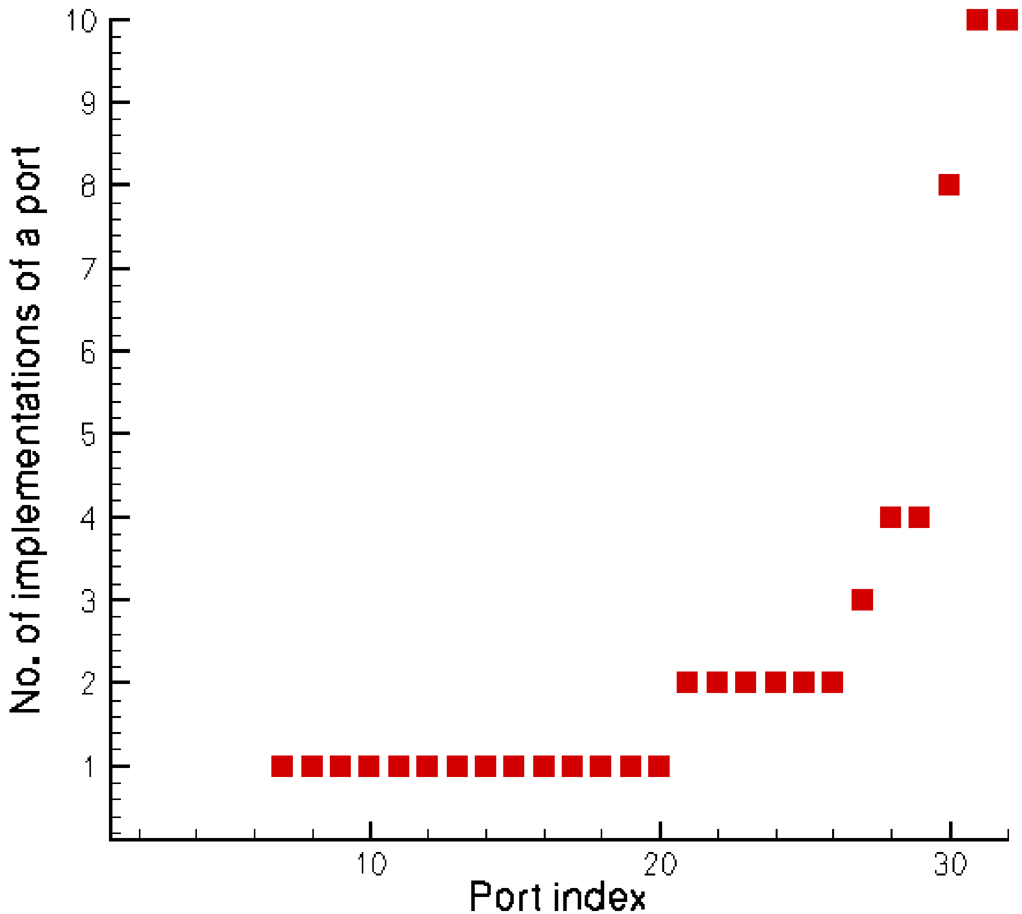
- ***Most components are < 250 kB.***
- ***The larger the binary, the more complexity is being hidden in underlying (externally contributed) libraries.***

Taming complexity: Interface size



- CCA Port is a unit of task exchange, and generally also a unit of thought.
- ***In our PSE, this is typically in the range of 5-10 functions***,
- **Exception : SAMR mesh data port.**

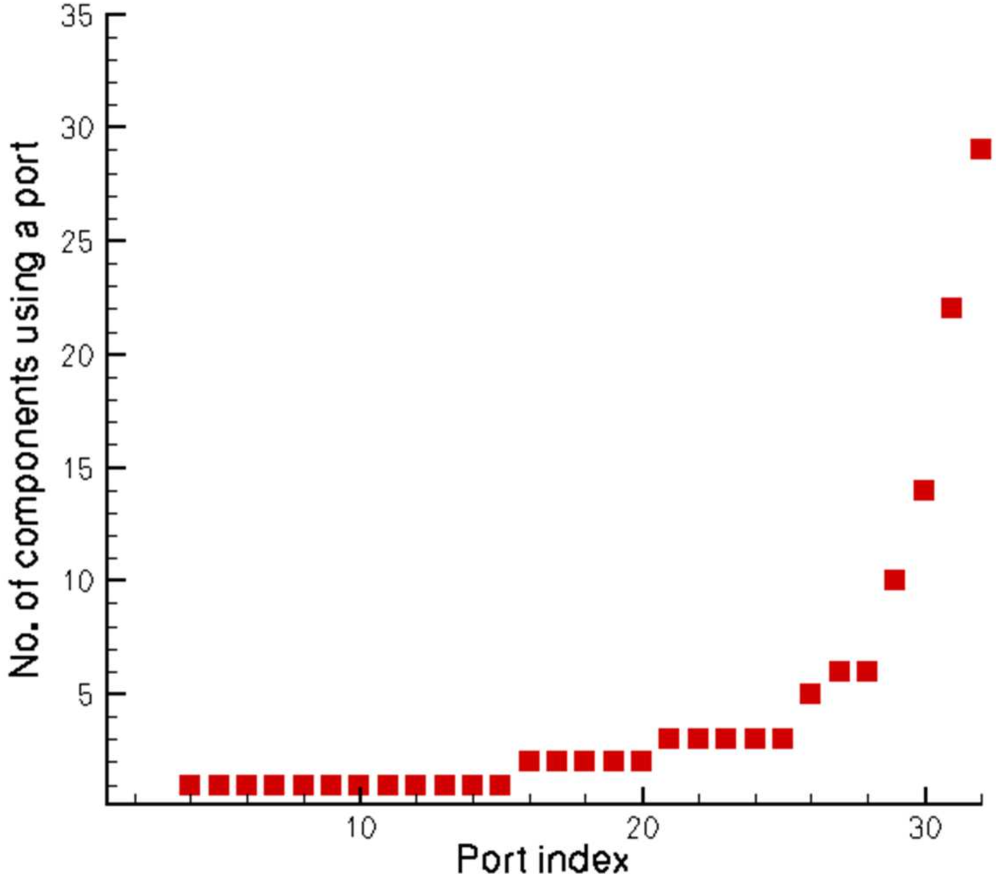
Taming complexity: Implementations



- **RFS ports may have just one or many implementations, as needed, but ..**
- ***Most Ports have 1 or 2 implementations***
- **But high-utility ports exists e.g. for exchanging a patch's worth of data.**

Sandia National Laboratories

Taming complexity: Callers



- ***Most RFS Ports are used by only a few clients, but ..***
- ***Key ports are used by many components.***

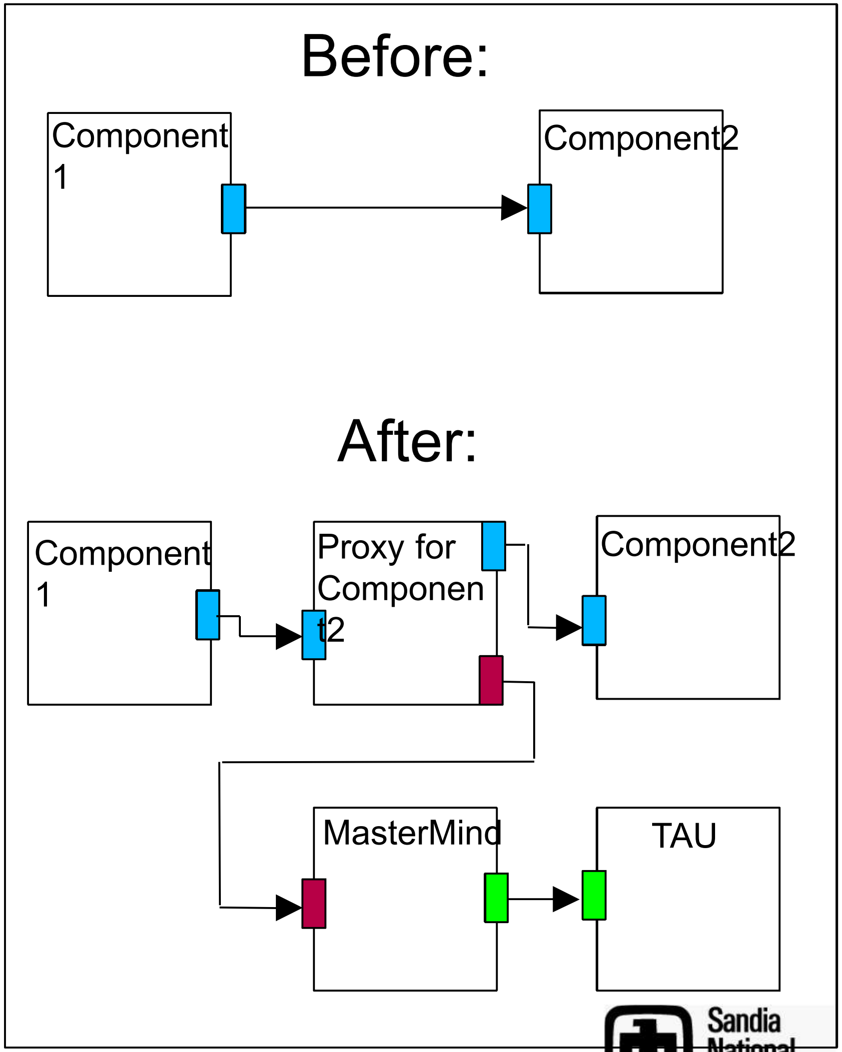
Performance Measurement In A Component World

- CCA poses a curious problem in profiling & modelling component performance
- In performance modelling one collects incoming inputs and match them up with the corresponding performance, by manually/automatically instrumenting the code, but ..
  - *What if the component is not yours ?*
  - *How does one non-intrusively instrument a code? And at what granularity ?*
- What kind of performance infrastructure can achieve this?
  - Previous research suggests proxies
    - Proxies serve to intercept and forward method calls

“Integrated” Performance Measurement Capability

Measurement infrastructure:

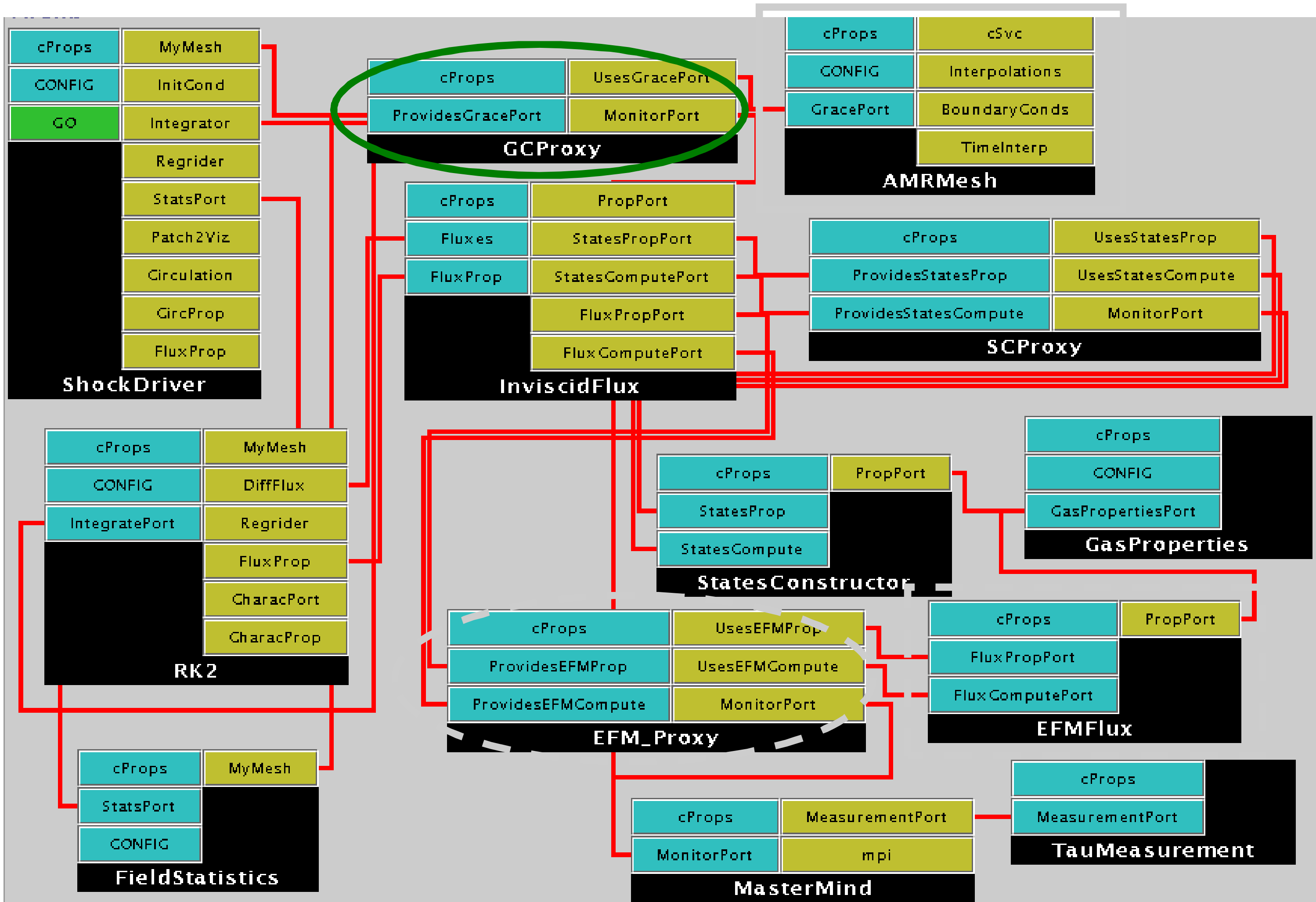
- **Proxy**
  - Notifies MasterMind of all method invocations of a given component, along with performance dependent inputs
  - Generated automatically using PDT
- **MasterMind**
  - Collects and stores all measurement data
- **TAU**
  - Makes all performance measurements
- **Work done at U. of Oregon by Prof. Malony and team**



Tuning and Analysis Utilities

Sandia National Laboratories

Component Application With Proxies



Productivity: Publication

- 6 test applications.
- 4 RFS journal papers, other software-oriented papers.
- ~11 conference papers, including best paper awards.
- Over 60 presentations.

Conclusions

- CCA has been an enabling technology
  - Enabled mathematicians to contribute new strategies, shrink-wrapped
- Enable the integrator (research scientist ?) to
  - Try unconventional approaches
- Dynamic codes a reality
  - A promising way to go petascale
  - But the devil's in the details.
- Contributions to research and codebase :
  - Sophia Lefantzi, Jeremiah Lee, Christopher Kennedy, W. Ashurst
  - K. Smith, M. Liu. N. Trebon
- Acknowledgements :
  - DoE's Office of Advanced Scientific Computing Research (MICS-funded CCTTSS SciDAC center) and Office of Basic Energy Sciences (Chemical Science's funded CFRFS SciDAC Center)

Sandia National Laboratories

Background

Reacting flows simulation research PSE

Put PSE category blocks here

Put amr movie here

BC	**IC**
Restrict	**Exp. ODE**
Prolong	**Imp. ODE**
Thermo	**Interpolation**
Kinetics	**Dif'n Models**
AMR Data	**Flux Models**
Parallel IO	**Error Estimators**

Sandia National Laboratories

Scalability: Contributors over time

Chart of this by quarters/years would be better:

Ray: PSE lead 2001-present

Allan: Ccaffeine support 2001-present

Lefantzi: 2001-2003

Smith:

Lee:

Kennedy:

Trebon:

Liu:

Ashurst:

Scalability: External libraries over time

Lots more of these, too. Order, dates probably wrong.

Grace v1, HDF, MPI – 2Q-01

Chemkin – 3Q01

KennedyLib – 2Q03

Chombo – 3Q03

CSP – 3Q04

Grace v2 – 2Q04

Tau – 1Q04