Two-Tiered Component Design and Performance Analysis of Synergia2 Accelerator Simulations

Stefan Muszala (Tech-X Corporation), Jim Amundson (FNAL), Lois Curfman McInnes (ANL), Boyana Norris (ANL)

Phase-I and II DOE SBIR and ComPASS SAP Project

CCA BABEL

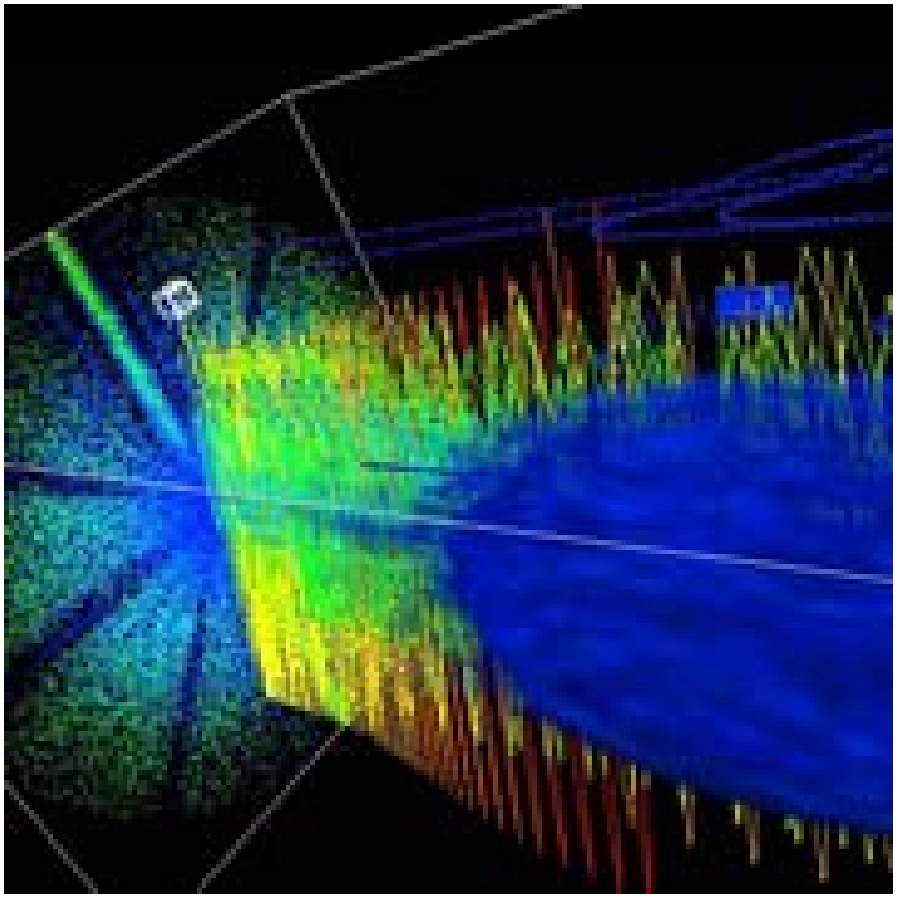
TECH-X CORPORATION

Accelerator simulations play vital near-, medium-, and long-term roles

- **Particle accelerator programs play a significant role in 14 out of 28 DOE facilities, which span a number of DOE offices such as the Offices of High Energy Physics (HEP), Nuclear Physics (NP) and Basic Energy Sciences (BES)** ***(Facilities for the Future of Science)***
- **Accelerator simulation is required throughout the life-cycle of accelerators in four areas**
  - Design
  - Analysis
  - Optimization
  - Upgrading

High-performance accelerator software should allow complex applications while promoting good software engineering practices

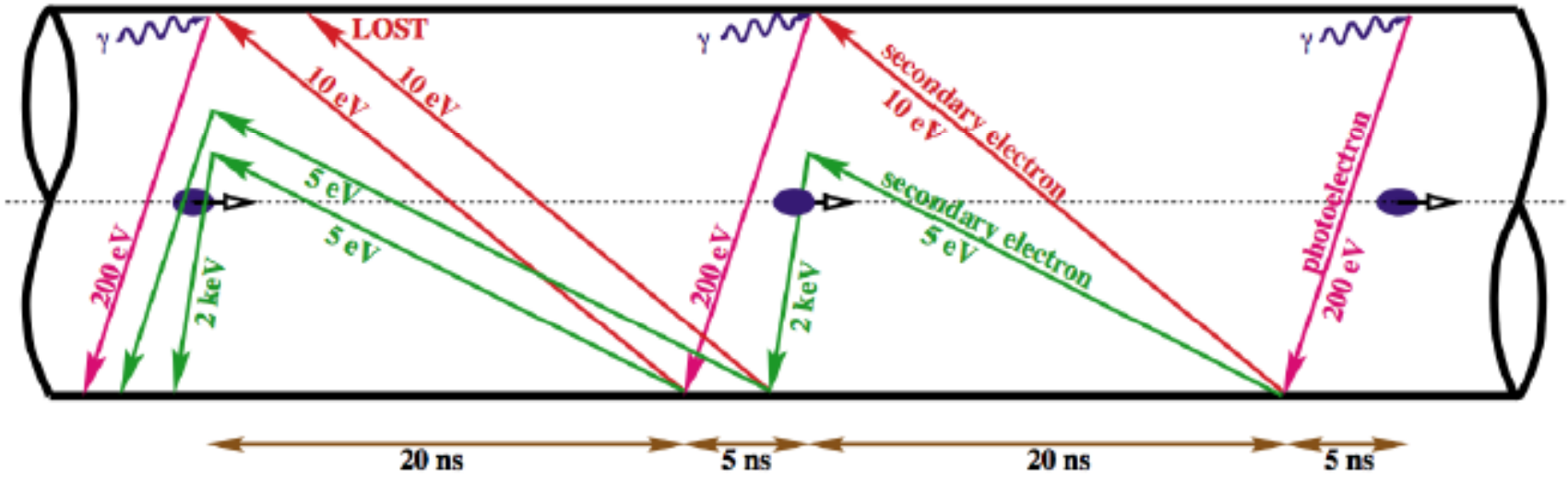
- Software reuse and common interfaces
- Ability to compose simulations
- Portability
- Mixed language programming interoperability
- Performance analysis of composed simulations



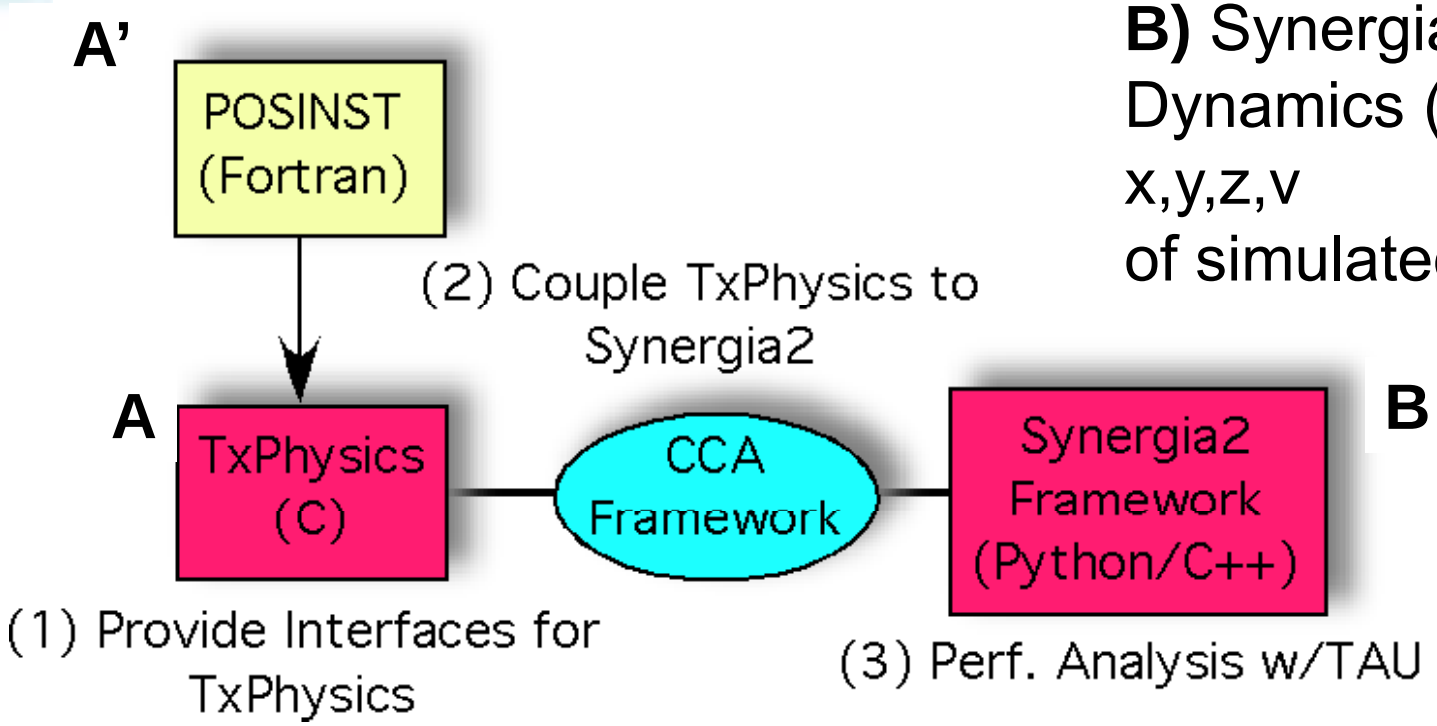
The Electron Cloud Effect (ECE) is one of the most pervasive issues in accelerator modeling and is part of the COMPASS SciDAC project

- **Community Petascale Project for Accelerator Science and Simulation (COMPASS)**
- **An ECE simulation combines beam dynamics with cloud generation codes**

  Electrons bouncing off of beam walls often emit more electrons due to secondary emission and eventually build into a cloud. The ECE is important to particle accelerator simulations since the cloud causes the proton beam to degrade



High level view of coupling TxPhysics and Synergia



B) Synergia2 for Beam Dynamics (calculates
x,y,z,v
of simulated particles)

A) TxPhysics for Electron Cloud Generation
a. Number of secondary electrons produced in each impact
b. Energy spectrum of those electrons

ECE simulations need to address coupling and language interoperability solutions

- **Software engineering hurdles specific to ECE simulations**
  - Beam dynamics (Synergia) and cloud generation (POSINST) codes are often written in different programming languages

    ==> *Need to solve language interoperability problem*
  - Timescales of particles in beam dynamics involve movement over km (10^5 cm) while those of cloud generation involve movement over cm

    ==> *Requires solution to software interoperation problem so codes may be easily coupled and composed*

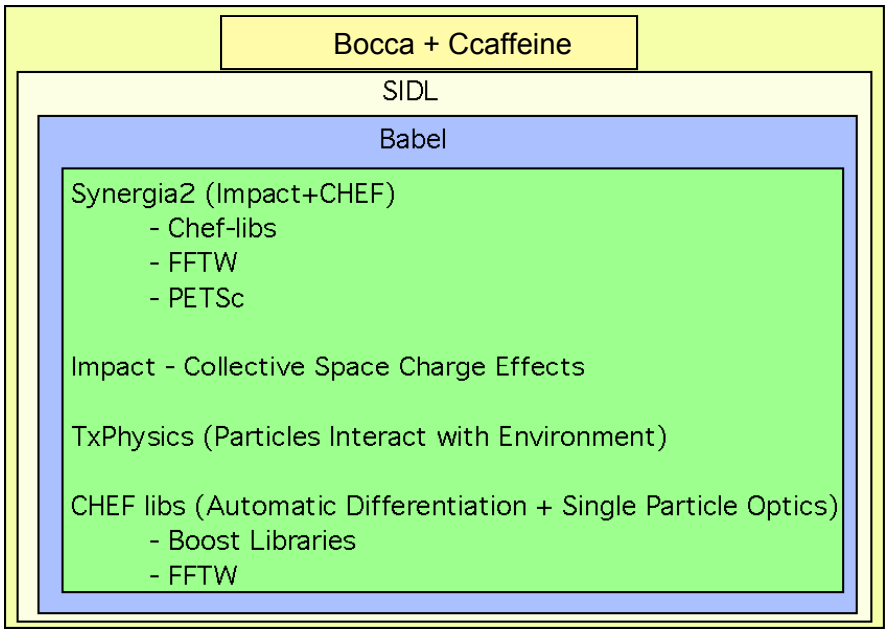
- **The Common Component Architecture (CCA) addresses these hurdles by providing a framework and language interoperability (Babel)**

- **TAU performance tools provide necessary performance monitoring**

New CCA Electron Cloud simulation

Beam Dynamics Toolkit

- F90-based beam **optics components** (quadrupoles and drifts) from the MaryLie/Impact application (LBNL)
- C++ and F90 **particle store components** from the Synergia2 framework (FNAL)
- A newly implemented C++-based **space charge solver**, Sphyraena, which uses Synergia2, PETSc (ANL), and FFTW
- C++ **ionization components** from TxPhysics (Tech-X)



Uses CCA tools:

- **Bocca**: Creates skeletal structure for a component and its interfaces, including the entire build system
- **SIDL/Babel:** Provides language interoperability

TxPhysics models the interaction of charged particles with environment

Port Name	Description
txrandPort	Custom random number generators
txegenelecPort	Algorithms for calculating electron-induced
	secondary electron emission
txionpackPort	Algorithms for calculating field and impact
	ionization rates
txstoppingPort	Algorithms for calculating ion stopping powers,
	including bound electronic, free electron, and
	nuclear stopping
txsputterPort	Algorithms for calculating ion-induced physical
	sputtering rates
txradiationPort	Algorithms for calculating impurity radiation rates
txigenelecPort	Algorithms for calculating ion-induced secondary
	electron emission

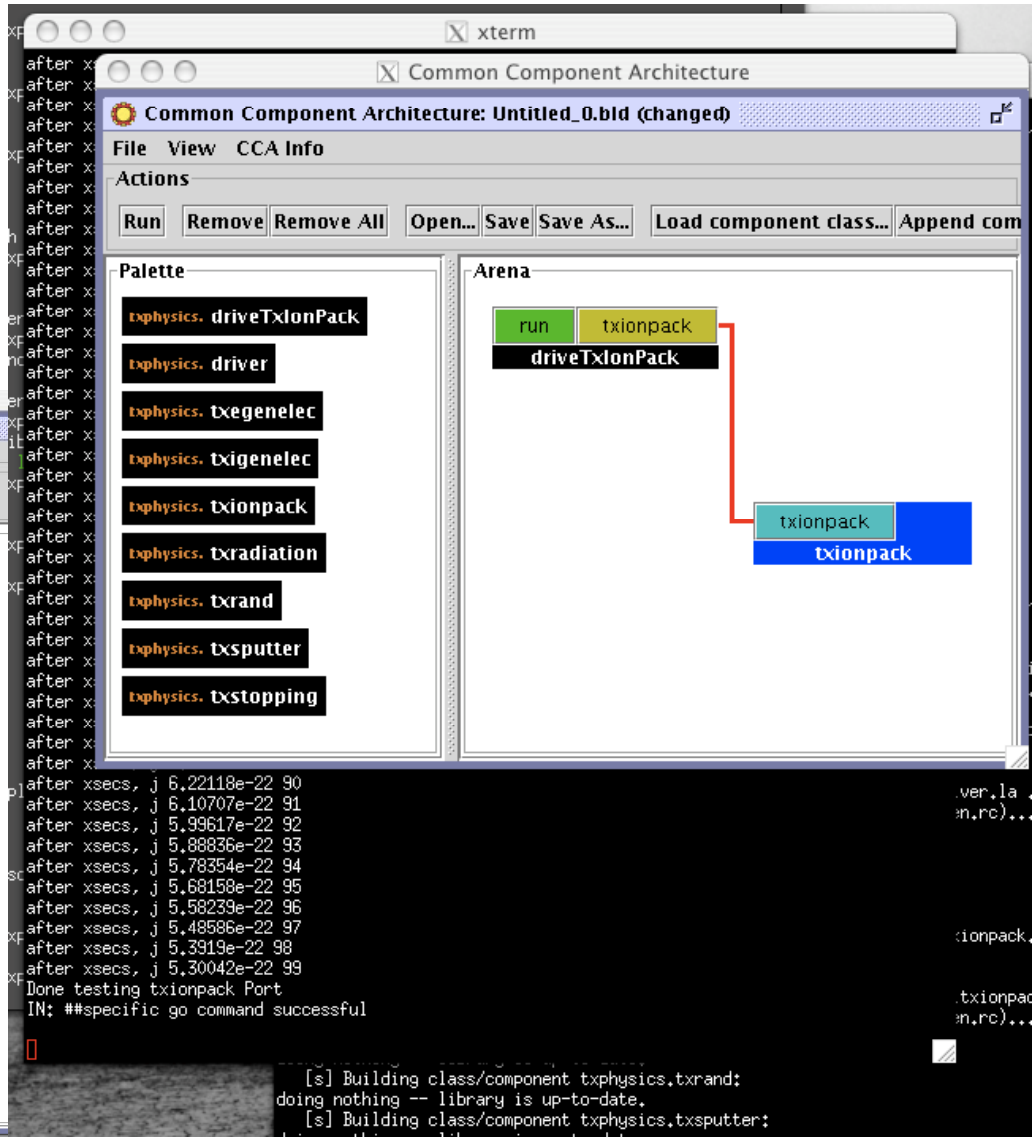
TxPhysics types are arrays and scalars and translate to SIDL well

```
package TxPhysicsCCA version 0.0 {
  interface txegenelecPort extends gov.cca.Port
  {
    void nsec_array(
      in      rarray<double,1>   E0_arr(n_inc_elecs),
      in      rarray<double,1>
costheta_arr(n_inc_elecs),
      in      int                n_inc_elecs,
      in      int                matnum,
      out     int                nstotal,
      inout   rarray<double,1>   beta_nor(maxelecs),
      inout   rarray<double,1>   beta_tan(maxelecs),
      inout   rarray<double,1>   beta_z(maxelecs),
      in      int                maxelecs);
  }
}
```

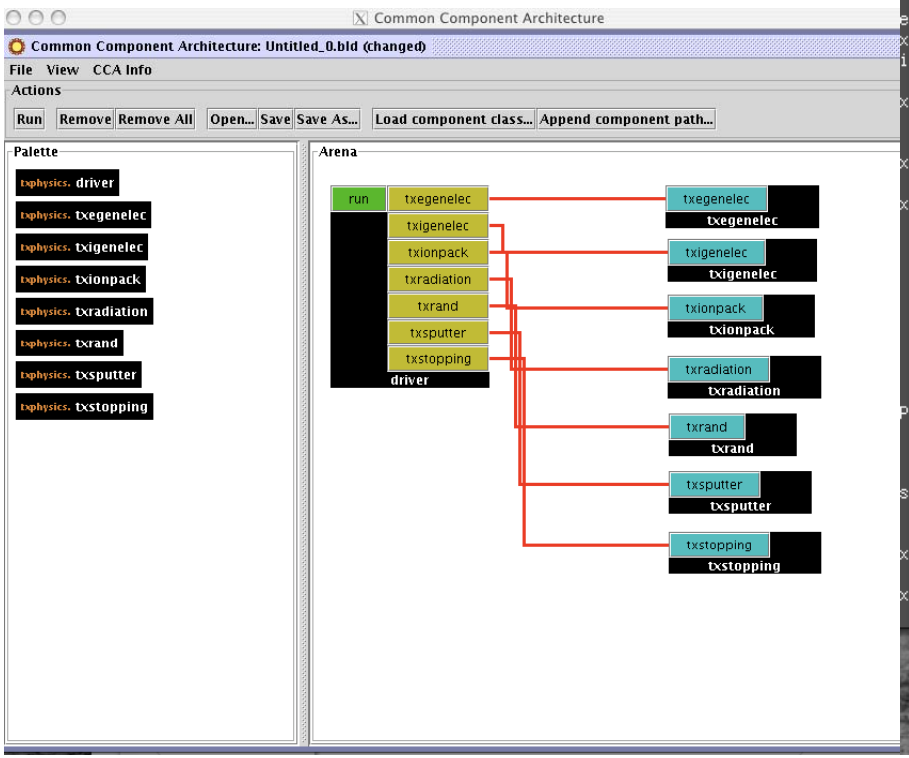
nsec_array method: calculates the number of electrons produced from secondary emission and their velocities due to a number incident electrons

We designed CCA components for TxPhysics and tested ports and interfaces

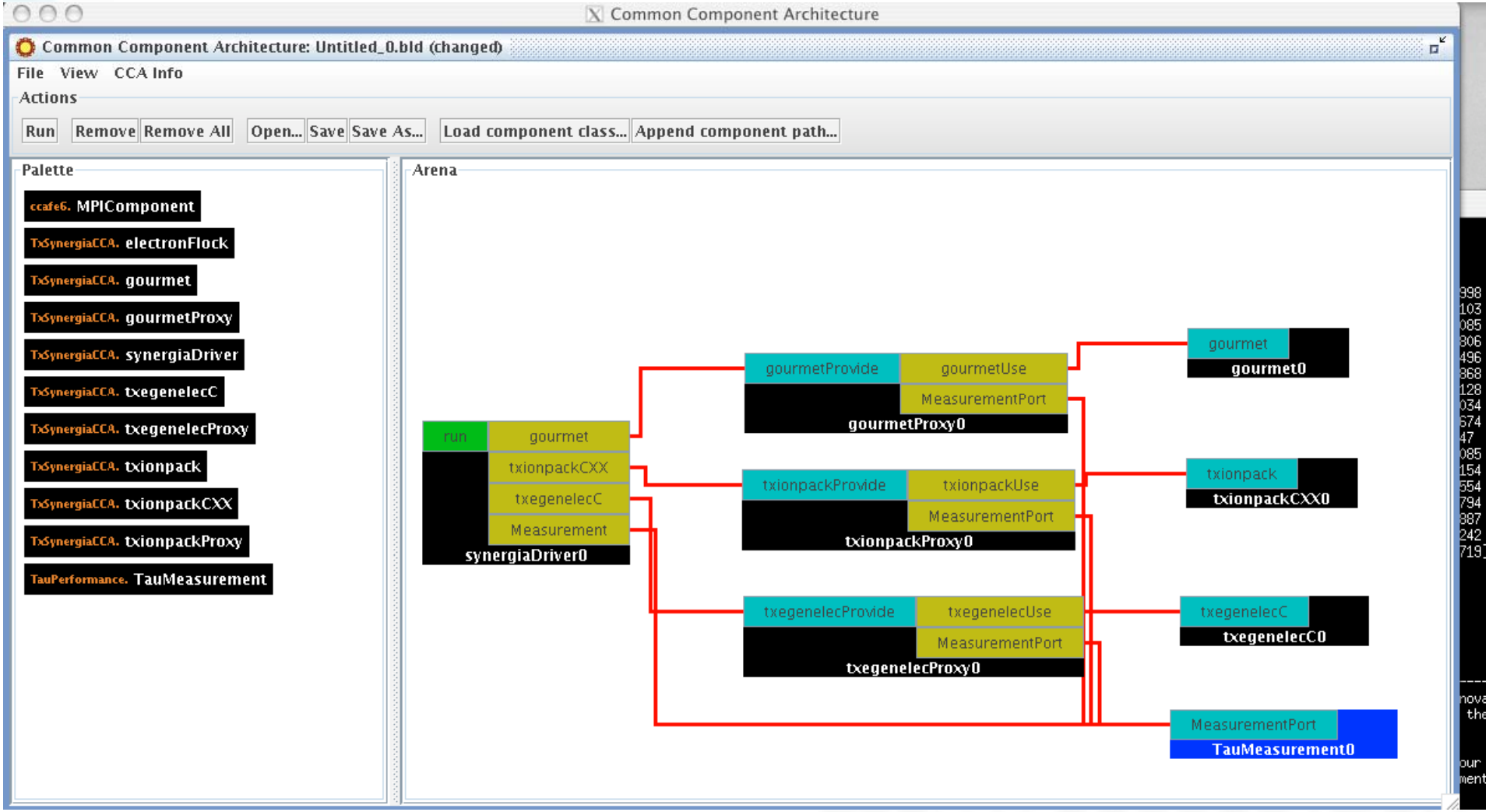
TxIonPack component testing



All CCA TxPhysics components

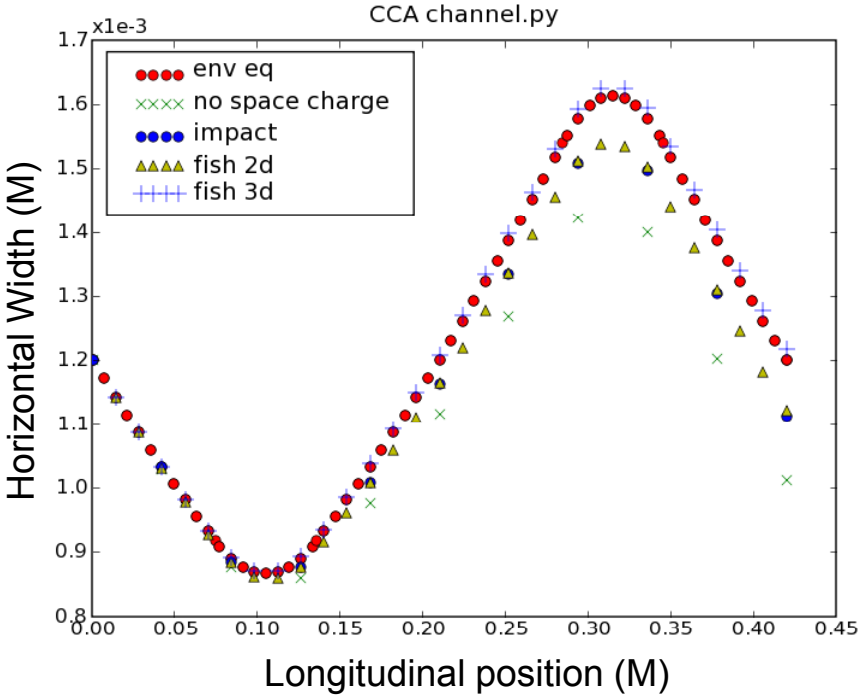


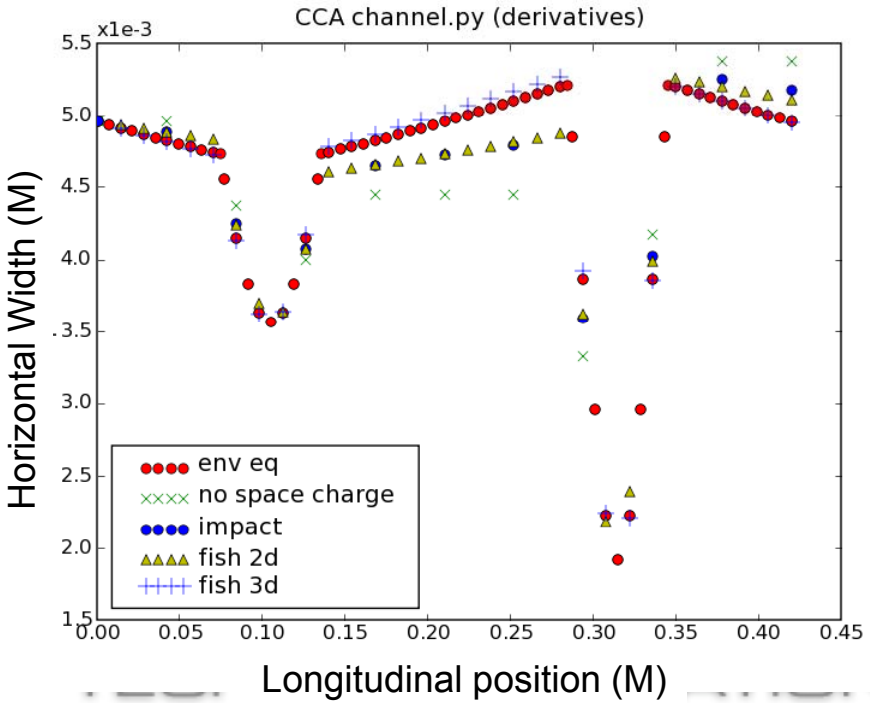
The full CCA Ecloud simulation uses TxPhysics, Synergia2 and TAU performance tools components.



The original Synergia2 channel driver exercised different space charge routines and provides a concise test-bed for a CCA implementation

Results are comparable



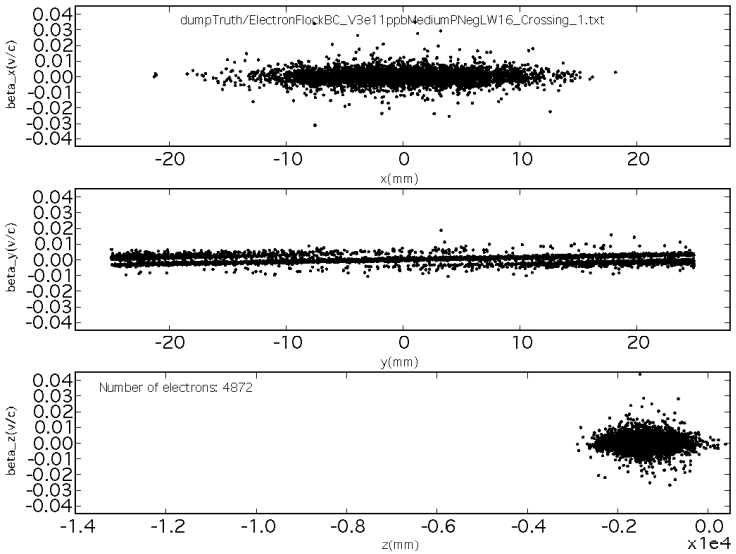


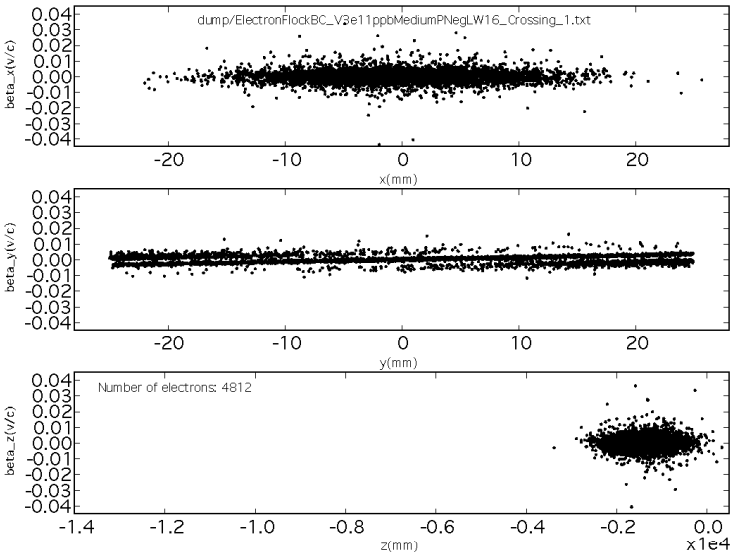
Comparison of original and CCA versions of beta_{x,y,z} vs. {x,y,z} show similarity after multiple bunch crossings.

ORIGINAL

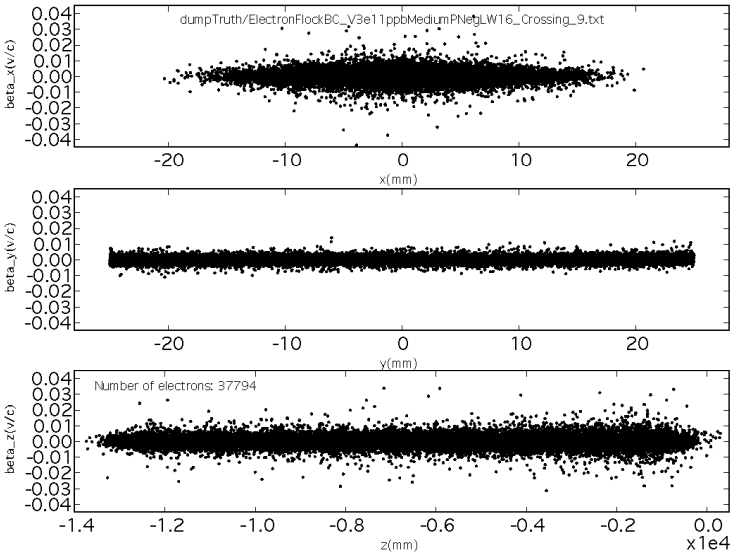
CCA

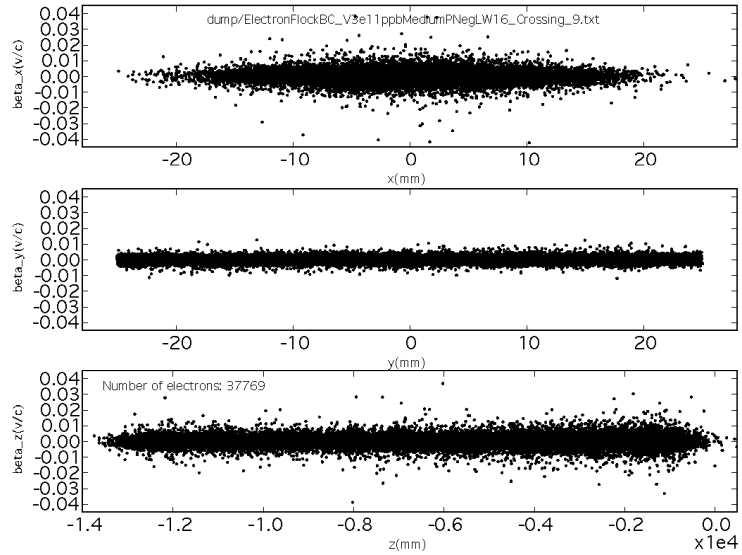
Cross=1



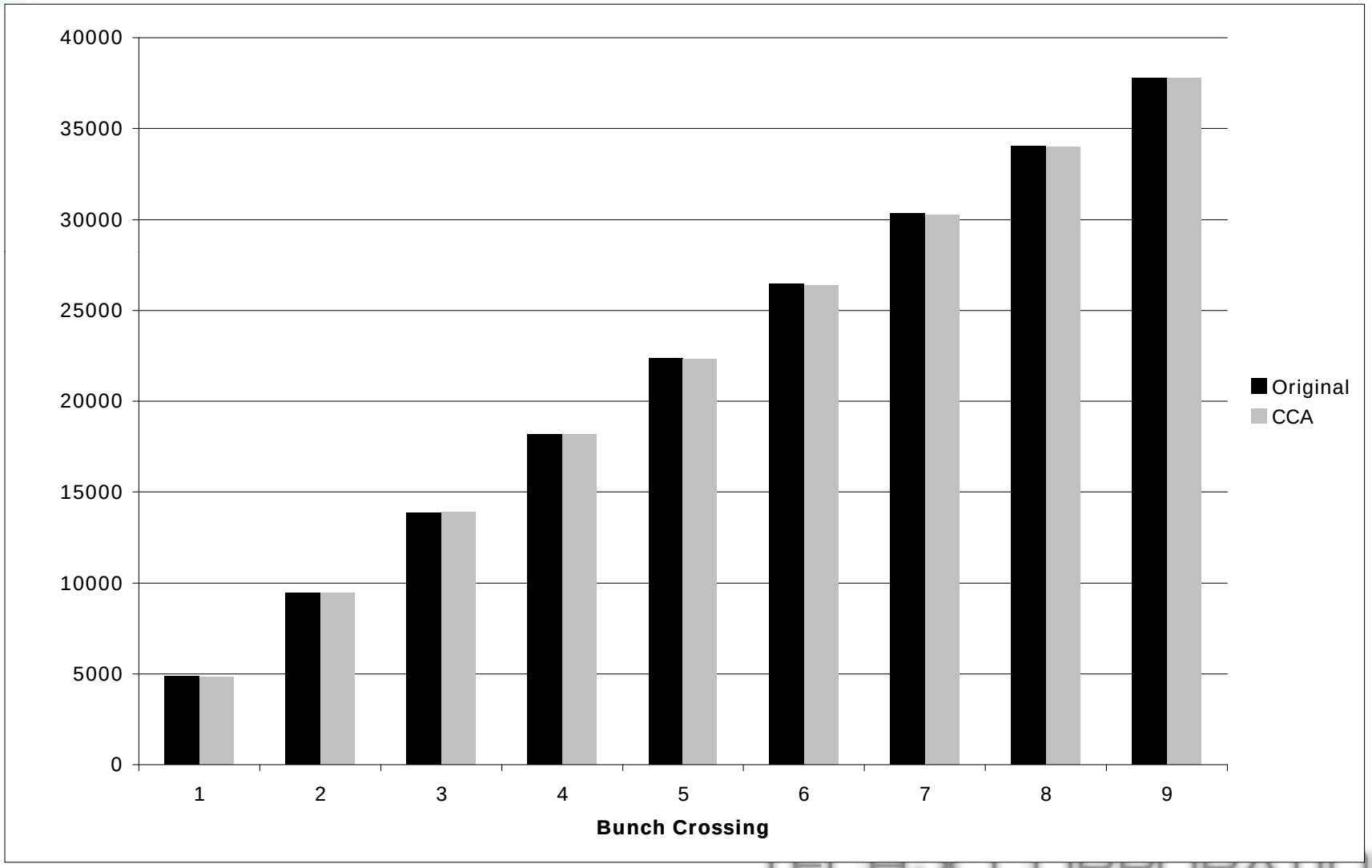


Cross=9





Comparison of original and CCA versions of the number of electrons produced after each bunch crossing shows similar behavior.



Initial profiles indicate that integrators and solvers are computationally most expensive

```
[quartic /home/research/muszala/svnStuff/ccaEcloud/src/TxSynergiaCCA  ]$
>>pprof perfProfiles/10Crossing/profile.0.0.0
Reading Profile files in profile.*

NODE 0;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0           1    58:54.016           1          33 3534016542 TxSynergiaCCA_synergiaDriver synergiaDriver::main
 96.5           1    56:49.436          10          10  340943617 ElectronFlockProp
 96.5       0.788    56:49.434          10          50  340943494 ElectronFlockInternal
 58.8    34:38.714    34:39.152          10       27268  207915242 ElectronFlockInternalPropNoBeam
 37.0    21:47.867    21:48.309          20       27775   65415464 ElectronFlockInternalPropWithBeam
  7.0     2:04.571     4:09.143           8           4   31142879 TxSynergiaCCA_gourmetProxy gourmet::gourmetCcaWrapPropagate3D
  3.5            1     2:04.395           4
  0.6       21.972       21.972          20
  0.0          880          880       55043
  0.0        0.246          178           1
  0.0            3            3           4
  0.0            1            2           2
  0.0        0.427        0.454           1
  0.0        0.226        0.226          12
  0.0         0.11        0.214           2
  0.0        0.193        0.193           1
  0.0        0.064        0.123           2
  0.0        0.027        0.027           1
---------------------------------------------------------------------------------------

USER EVENTS Profile :NODE 0, CONTEXT 0, THREA
---------------------------------------------------------------------------------------
NumSamples   MaxValue   MinValue  MeanValue
---------------------------------------------------------------------------------------
 5.506E+04  2.072E+05  5.084E+04  1.491E+05
```

TAU: ParaProf Manager

File Options Help

- Applications
  - Standard Applications
    - Default App
      - Default Exp
        - TxSynergiaCCA/src/ccaEcloud/svnStuff/muszala/research/home/
          - Time
  - ccaEcloud (jdbc:postgresql://quartic:5432/perfdmf)

TrialField	Value
Name	TxSynergiaCCA/src/ccaEcloud/svnStuff/...
Application ID	0
Experiment ID	0
Trial ID	0
CPU Cores	2
CPU MHz	2000.000
CPU Type	Dual Core AMD Opteron(tm) Processor 270
CPU Vendor	AuthenticAMD
CWD	/home/research/muszala/svnStuff/ccaEcl...
Cache Size	1024 KB
Executable	/home/research/muszala/ccaTutorial/qu...
Hostname	quartic.txcorp.com
Local Time	2009-01-20T13:46:02-07:00
Memory Size	4063204 kB
Node Name	quartic.txcorp.com
OS Machine	x86_64
OS Name	Linux
OS Release	2.6.25.14-108.fc9.x86_64
OS Version	#1 SMP Mon Aug 4 13:46:35 EDT 2008
Starting Timestamp	1232480828488343
TAU Architecture	x86_64
TAU Config	
TAU Version	2.17.2
Timestamp	1232484362520489
UTC Time	2009-01-20T20:46:02Z
pid	1238
username	muszala

TAU: ParaProf: Function Legend: TxSynergiaCCA/src/ccaEcloud/svn

File Filter Windows Help

- Bunch
- ElectronFlock
- ElectronFlockInternal
- ElectronFlockInternalIO
- ElectronFlockInternalPropNoBeam
- ElectronFlockInternalPropWithBeam
- ElectronFlockProp
- Gourmet
- Setup
- Synergia
- TxSynergiaCCA_gourmetProxy gourmet::gourmetCcaGetInitalU
- TxSynergiaCCA_gourmetProxy gourmet::gourmetCcaInsertSpaceChargeMarkers
- TxSynergiaCCA_gourmetProxy gourmet::gourmetCcaOrbitLength
- TxSynergiaCCA_gourmetProxy gourmet::gourmetCcaWrapPropagate3D
- TxSynergiaCCA_synergiaDriver synergiaDriver::main
- Txionpack
- txegenelcC::nsec_array
- txionpackCXX::get_sigma_impact_array

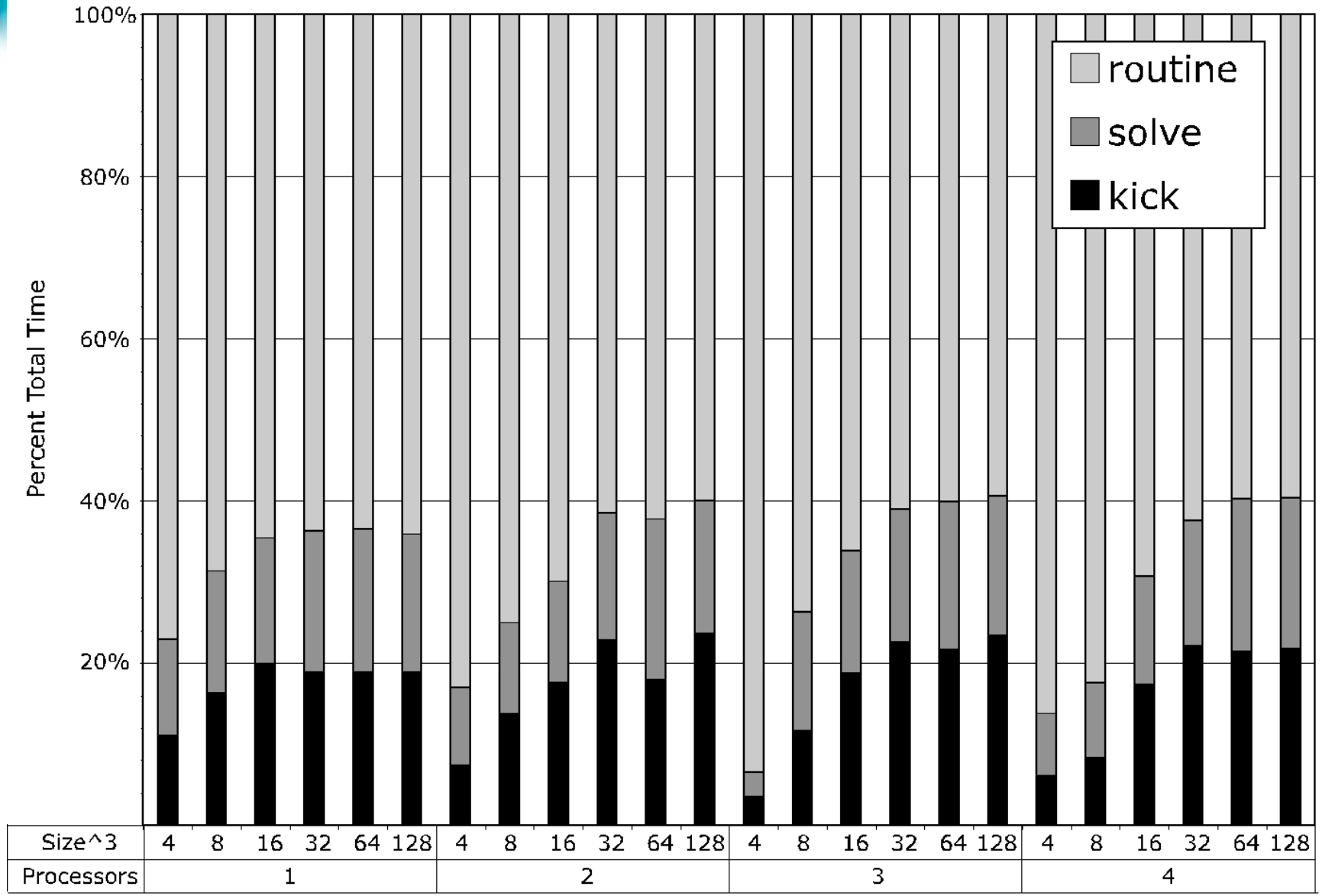
TAU: ParaProf: /home/research/muszala/svnStuff/ccaEcloud/src/TxSynergiaCCA

File Options Windows Help

Metric: Time
Value: Exclusive per Call

node 0

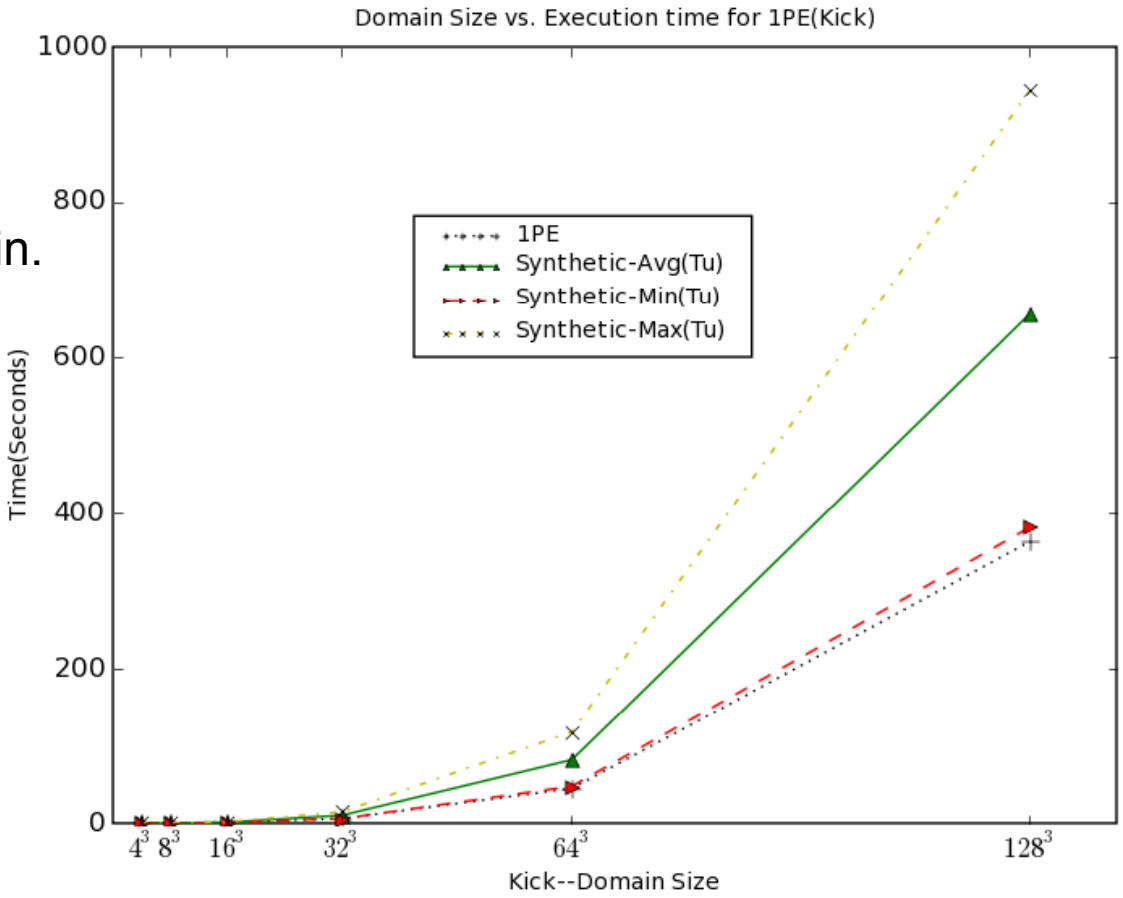
After instrumentation we can see Solve and Kick behavior for Synergia2



Single processor performance of the space charge kick goes as as N^3

- Number of Particles = N^3, N=grid size, particles/cell=1
- Core work consists of two triple-nested for loops over 3-dimensions: $6(N^3)$
- $T_1=T_u 6(N^3)\alpha+\beta$
- T_u = {min | max | average} time for cell update. We use min.
- need to study why $\alpha = 2.6$ and $\beta=\min(T_u)$

One processor run over increasing problem size using different values of the time for a single update (min, max, average)



Multi-Processor performance starts with Amdahl's law

- Start with Amdahl's law $T_P = S + Q/P$ but let $f = S/(S + Q)$ be the fraction of serial work (S=time for sequential code, Q=time for parallel part of code)
- Amdahl's law is now: $T_P = fT_1 + (1 - f)T_1/P$
- Account for communication for PEs > 0

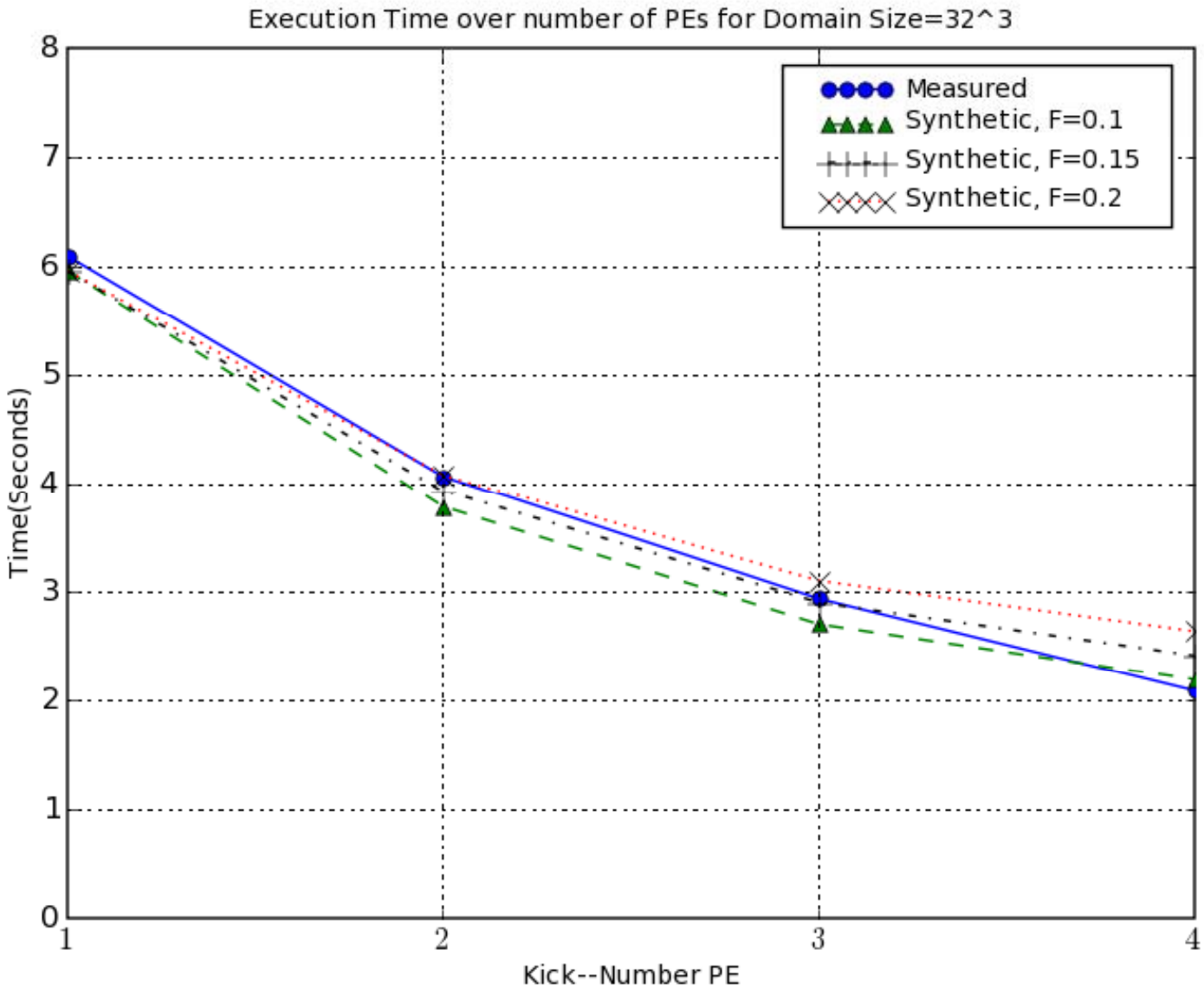
$$T_P = fT_1 + \frac{(1-f)T_1}{P} + T_{comm}$$

- Substitute for T_1 and T_{comm}

$$T_P = f(T_u 6N^3\alpha + \beta) + \frac{(1-f)(T_u 6N^3\alpha + \beta)}{P} + \frac{1}{log_2(P)}$$

- T_u = {min | max | average} for cell update. From serial model We use min. for now.
- Need to actually quantify f (cycle and instruction count), understand messaging better

Multi-Processor performance model matches real data for a 32^3 size problem (32,768 particles)



Current and out-year milestones that build toward ECE simulations with CQoS

CQoS (Computational Quality of Service) for accelerator simulations:
How, during runtime, can we make make sound choices for reliability, accuracy, and performance, taking into account the problem instance and computational environment?

- **Composition:** select initial component implementations and configuration parameters
- **Reconfiguration:** change parameters
- **Substitution:** change implementations

1) Design and implement CCA Synergia2 components for use on leadership-class supercomputers

- CCA tools infrastructure (getting this in place first)
- Interfaces and components

2) Explore performance models for parallel architectures networks and algorithms

- Performance analysis and modeling
- Build CQoS infrastructure

3) Demonstrate and compose various Synergia2 CCA electron cloud component use cases

- Compose ECE simulations
- Apply CQoS

This work is done in collaboration with multiple institutions and projects.

- Personnel
  - Jim Amundson (Fermi National Accelerator Laboratory)
  - Lois Curfman McInnes (Argonne National Laboratory)
  - Paul Lebrun (Fermi National Accelerator Laboratory)
  - Boyana Norris (Argonne National Laboratory)
  - Peter Stoltz (Tech-X)
  - Seth Veitzer (Tech-X)
- Institutions and URLs
  - COMPASS (Community Petascale Project for Accelerator Science and Simulation), SciDAC-2 program, https://compass.fnal.gov/
  - Common Component Architecture (CCA), http://www.cca-forum.org/
    - Bocca
    - Ccaffeine
    - Babel
    - CQoS (Computational Quality of Service),
  - Tuning and Analysis Utilities (TAU) for Performance Analysis and CQoS, http://www.cs.uoregon.edu/research/tau/home.php
  - TASCS (The Center for Technology for Advanced Scientific Component Software), http://tascs-scidac.org/

End of talk, extra slides follow

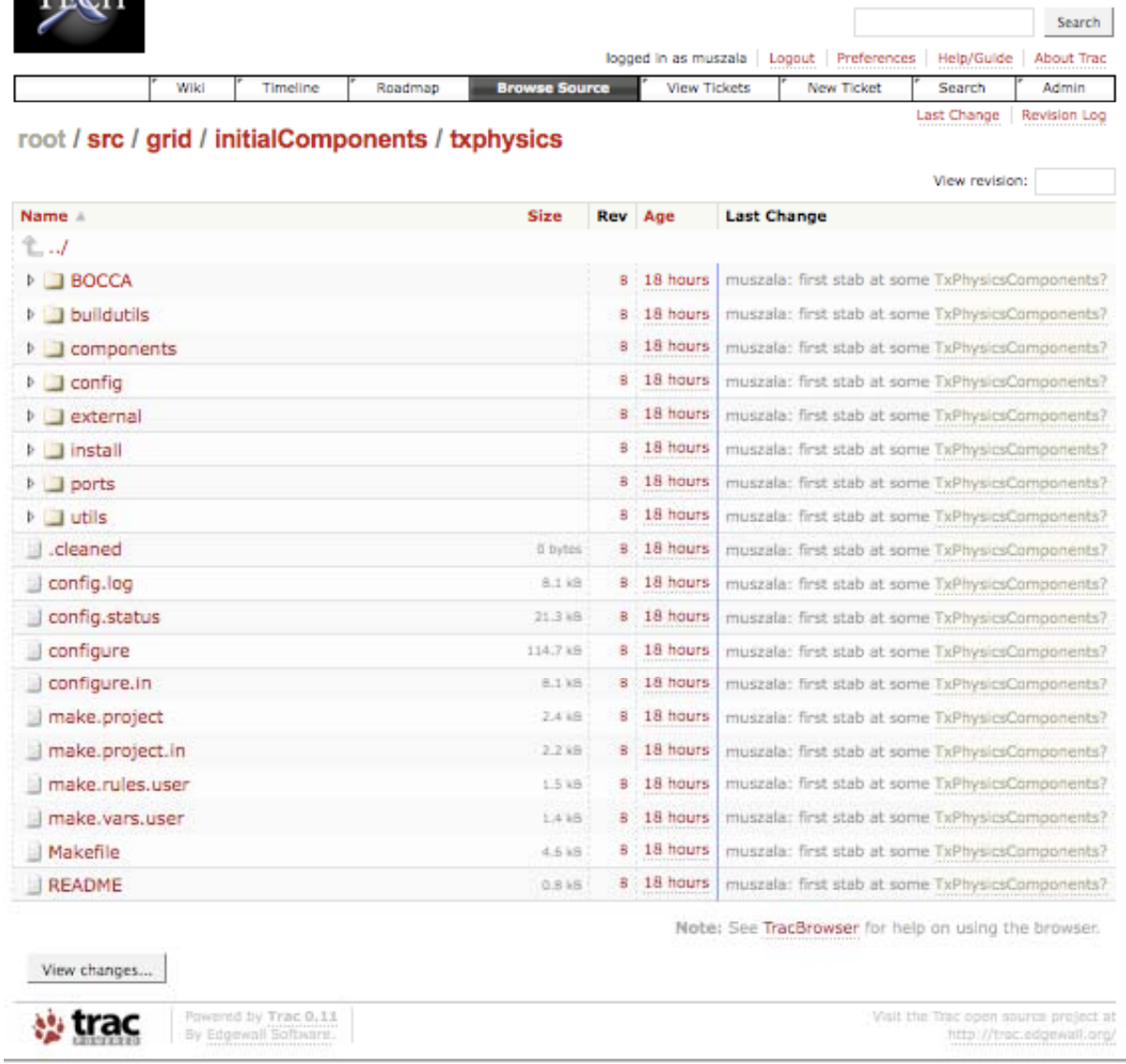
The source of overhead is due to existing Synergia2 method structure

```
#~ Python driver component call to current wrapPropagate component

synergiaGourmet.wrapPropagate(...)

#~ Python wrapPropagate Method of the synergiaGourmet Component only showing
#~ major logical branches

   for action in self.internalGourmet.get_actions():
     if action.is_mapping():
       for bunch in bunches:
          action.get_data().apply(...)
     elif action.is_synergia_action():
       if action.get_synergia_action() == "space_charge_endpoint":
       elif action.get_synergia_action() == "space_charge_kick":
         if use_s2_fish:
-->         s2_fish.apply_space_charge_kick(...)
         elif use_s2_fish_cylindrical:
         elif use_impact:
         elif use_gauss:
         elif use_none:
         elif action.get_synergia_action() == "rfcavity1" or \
            action.get_synergia_action() == "rfcavity2":
              chef_propagate.chef_propagate(...)

#~s2_fish.apply_space_charge_kick in turn has the following structure
#~
def apply_space_charge_kick(...)
   if impedance or space_charge:
   if impedance:

   if space_charge:
#~ Call to C++ implementation of FFT before space charge kick (SOLVE)
     phi = solver_fft_open(...)
#~ Call to C++ implementation of space charge kick (KICK)
-->   full_kick(...)
```

New Synergia2 call structure will be simplified

```
#~ Python driver component call to renamed propagate component

synergiaGourmet.propagate(...)

#~ Python propagate Method of the synergiaGourmet Component only showing
#~ major logical branches

        for bunch in bunches:
            action.get_data().apply(...)
#~ note that s2Fish is now it's own CCA Component
-->     synergiaS2Fish.apply_space_charge_kickSC(...)

#~ synergiaS2Fish.apply_space_charge_kickSC in turn has the following structure
#~
def apply_space_charge_kickSC(...)

#~ Call to C++ Component implementation of FFT before space charge kick (SOLVE)
      phi = synergiaS2FishSolver.solver_fft_open(...)
      phi = solver_fft_open(...)
#~ Call to C++ Component implementation of space charge kick (KICK)
-->   synergiaS2FishSolver.full_kick(...)
```

Bringing good computer science practices to physics applications: svn and trac



https://ice.txcorp.com/trac/ccaEcloud/

(Currently internal, soon will be external)

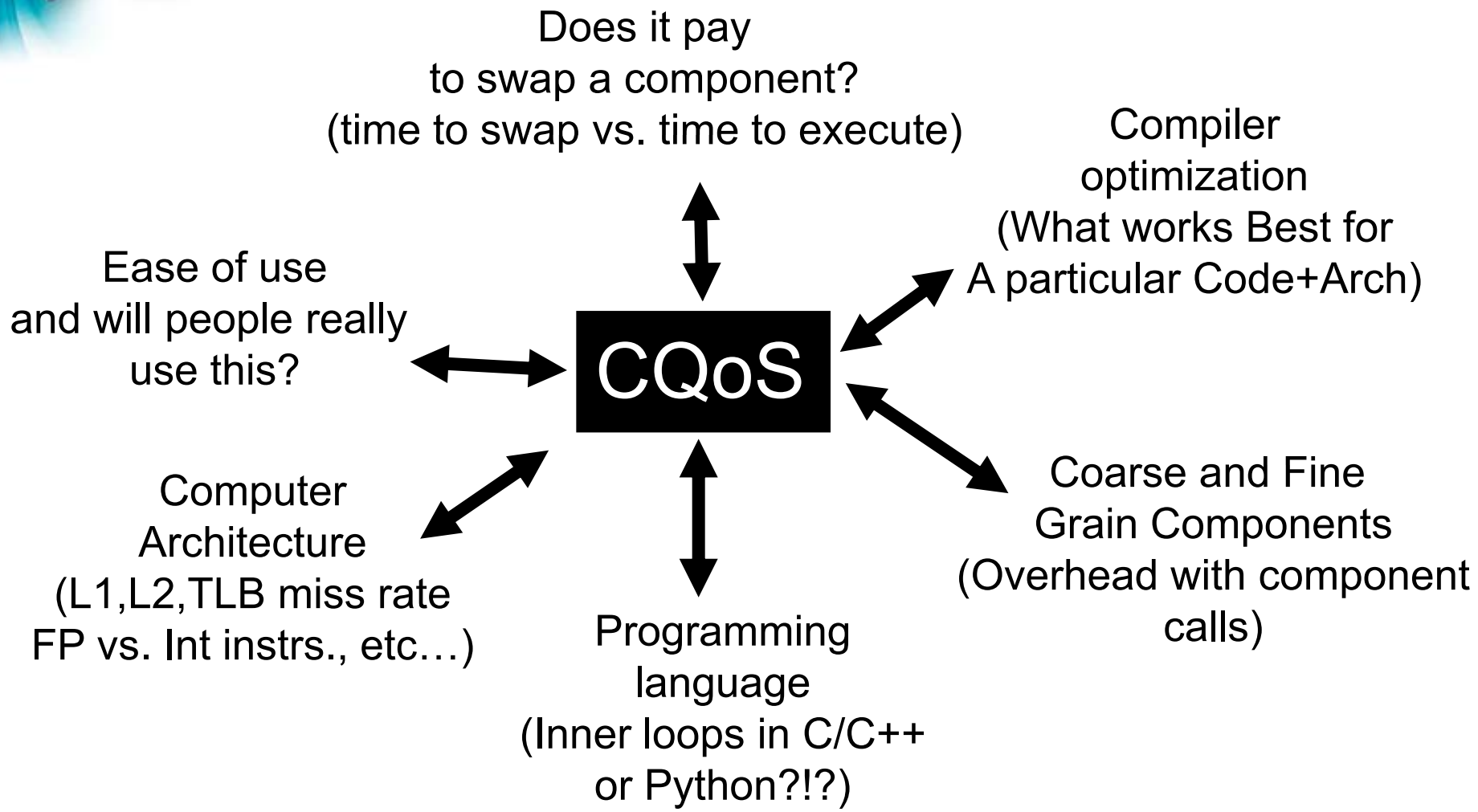
The Common Component Architecture addresses issues that are important to scientific computation

- Common Interfaces: CCA uses SIDL (Scientific Interface Definition Language) to define interfaces and add an OO structure to otherwise non-OO languages.
- OOP+: Allows OO programming between different languages using Babel (C,C++,Java,F90,F77,F03,Python).
- Composing Simulations: CCA components allow you to exchange components easily and build simulations using run files.
- HPC: CCA and Babel are written independent of type of parallelism (message passing, shared memory) and work with PVM, MPI, openMP, etc…

Opportunities for collaboration, future plans and design questions abound:

- Auto-generation of interfaces based on existing code
  - Tested CCA software On-Ramp to generate SIDL interfaces from TxPhysics source code
  - Conceivable use outside of CCA (just change code-gen for SIDL to something else--C interfaces)
- CQoS is the 'novel' computer engineering/science work. Plan to spend 75 percent of time on performance modeling and showing CQoS can work.
  - Show that it works with/without CCA as this will appeal to a broader ECE/CS audience.
- Plan to compare Synergia2 and Vorpal performance
- Possibility to offload selected Synergia2 work to GPUs, optimize for shared-memory
- Other Open Questions
  - How do we educate a user to want to build and use these tools?
  - Portability and testing of Synergia2 and Chef
    - Will run nightly tests at Tech-X
    - Add regression testing to Chef

CQoS requires understanding a number of parameters



The Electron Cloud Effect (ECE) is one of the most pervasive issues in accelerator modeling and is addressed as part of the COMPASS project

- **Community Petascale Project for Accelerator Science and Simulation (COMPASS)**
  - COMPASS goal :: developing *"a comprehensive set of interoperable components for beam dynamics, electromagnetics, electron cooling, and advanced accelerator modeling"* (Spentzouris, Community Petascale Project for Accelerator Science and Simulation (COMPASS). FNAL DOCDB, CD-doc-2098, version 1, 2007)
  - Is a SciDAC2 project funded through HEP, NP, BES and ASCR
    - Synergia2 provides the beam dynamics code (Python, F90, C++)
    - POSINST (TxPhysics) provides the cloud generation code (C)