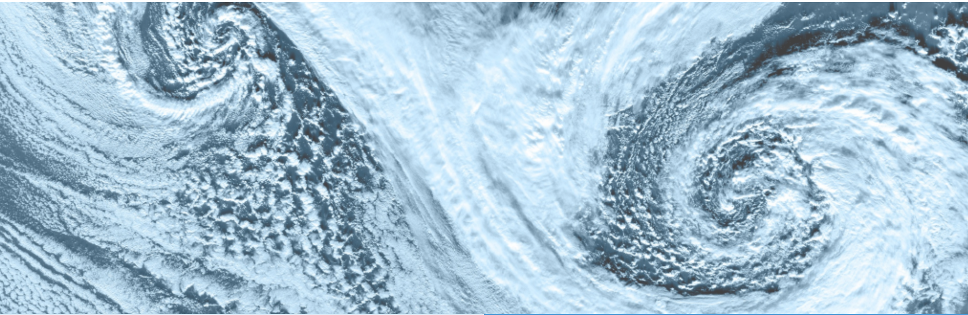
C2SM
Center for Climate
Systems Modeling

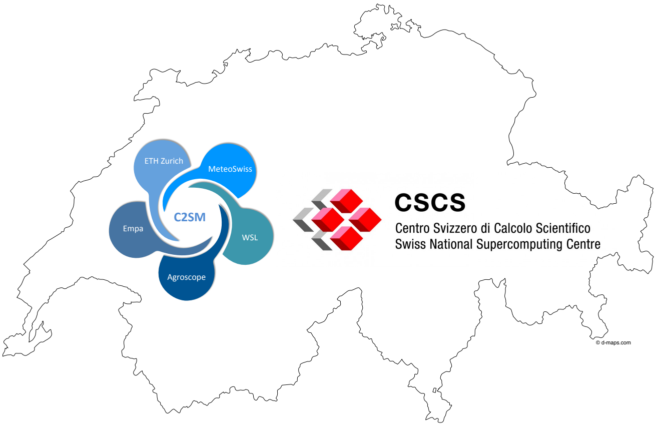
CLAW Provides High-Level Abstractions for Weather and Climate Models

ENES Workshop April 7, 2016

Jon Rood

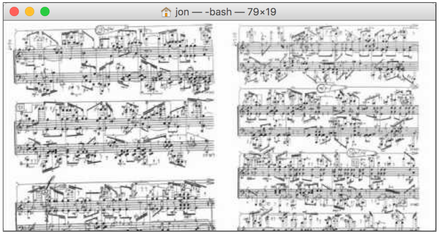


Who is Involved in CLAW?


ETH Zurich
MeteoSwiss
C2SM
Empa
WSL
Agroscope
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre
© d-maps.com

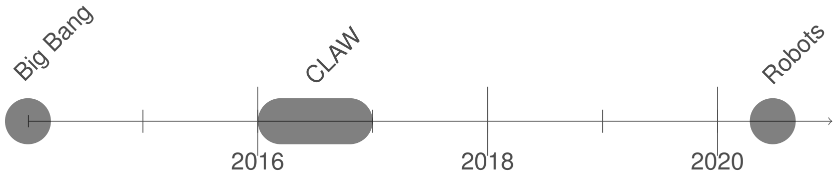
- People involved
  - Valentin Clément (C2SM), Sylvaine Ferrachat (ETH Zurich), Oliver Fuhrer (MeteoSwiss), Xavier Lapillonne (MeteoSwiss), Jon Rood (C2SM), Will Sawyer (CSCS)

What is CLAW?



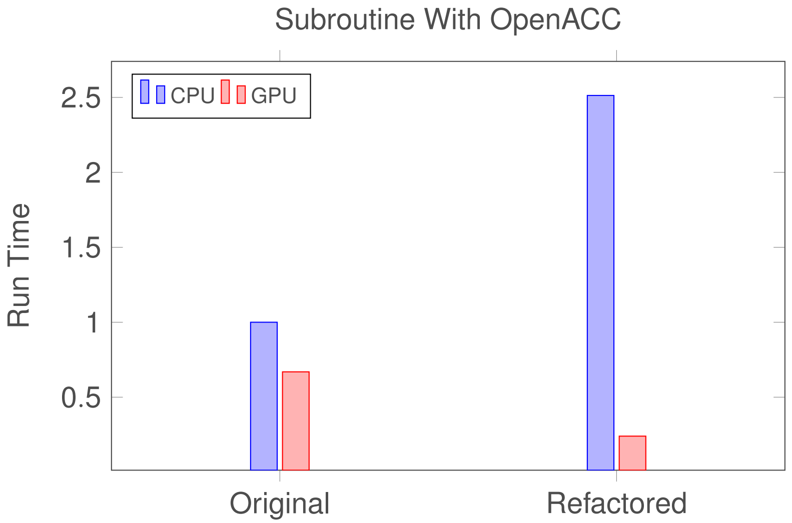
- Scientists must conduct a computational symphony of interdependent physical, virtual, and theoretical instruments of optimization
  - Machine configurations, software environments, programming languages, compilers, compiler flags, model configurations, domain-based optimizations, algorithmic optimizations, source code optimizations, etc.
- CLAW is an effort to increase the performance portability and maintainability of climate and weather codes (that are written in Fortran)

When is CLAW Happening?



- CLAW is happening right now through 2016 as a small project to develop a proof-of-concept
- Hopefully future funding possibilities will expand its functionality and abstraction

Why Develop CLAW?



Code Example Using #IFDEF

Original Subroutine

```
!$acc parallel loop gang
DO k=1,ntrac
  IF (trlist%ti(k)%nsedi==0) CYCLE
  imod=trlist%ti(k)%mode
  slinnfac=sigma(imod)**(2._dp*sigmaln(imod))
  IF (trlist%ti(k)%nphase==AEROSOLNUMBER) THEN
    !$acc parallel loop gang vector collapse(2)
    DO j=1,klev
      DO i=1,kproma
        zmd(i,j)=MIN(rwet(imod)%ptr(i,j,krow)*2._dp,50.E-6_dp)
      END DO
    END DO
  END IF
  !$acc parallel loop gang vector collapse(2)
  DO j=1,klev
    DO i=1,kproma
      zsediflux(i,j)=0._dp
      IF (zmd(i,j)>0._dp) THEN
        zxtp1=pxtm1(i,j,k)+pxtte(i,j,k)*time_step_len
        zvsedi=(2._dp/9._dp*&
          (densaer(imod)%ptr(i,j,krow)-zrho(i,j)) &
          *grav/zvis(i,j)*(zmd(i,j)/2._dp)**2._dp) &
          *(slinnfac+1.246_dp*2._dp*zlair(i,j) &
          /zmd(i,j)*exp((0.5_dp*sigmaln(imod)**2._dp)))
        zsedtend=zxtp1*zvsedi/zdz(i,j)
        pxtte(i,j,k)=pxtte(i,j,k)-zsedtend
        zsediflux(i,j)=zsedtend*zdpg(i,j)
      END IF
    END DO
  END DO
  !$acc parallel loop gang vector collapse(2)
  DO j=2,klev
    DO i=1,kproma
      pxtte(i,j,k)=pxtte(i,j,k)+(zsediflux(i,j-1)/zdpg(i,j))
    END DO
  END DO
END DO
!$acc end parallel loop
```

Refactored Subroutine

```
!$acc parallel loop gang vector collapse(2)
DO k=1,ntrac
  DO i=1,kproma
    DO j=1,klev
      IF (trlist%ti(k)%nsedi==0) CYCLE
      imod=trlist%ti(k)%mode
      slinnfac=sigma(imod)**(2._dp*sigmaln(imod))
      IF (trlist%ti(k)%nphase==AEROSOLNUMBER) THEN
        zmd=MIN(rwet(imod)%ptr(i,j,krow)*2._dp,50.E-6_dp)
      END IF
      zsediflux(j)=0._dp
      IF (zmd>0._dp) THEN
        zxtp1=pxtm1(i,j,k)+pxtte(i,j,k)*time_step_len
        zvsedi=(2._dp/9._dp*&
          (densaer(imod)%ptr(i,j,krow)-zrho(i,j)) &
          *grav/zvis(i,j)*(zmd/2._dp)**2._dp) &
          *(slinnfac+1.246_dp*2._dp*zlair(i,j) &
          /zmd*exp((0.5_dp*sigmaln(imod)**2._dp)))
        zsedtend=zxtp1*zvsedi/zdz(i,j)
        pxtte(i,j,k)=pxtte(i,j,k)-zsedtend
        zsediflux(j)=zsedtend*zdpg(i,j)
      END IF
      IF (j>1) THEN
        pxtte(i,j,k)=pxtte(i,j,k)+(zsediflux(j-1)/zdpg(i,j))
      END IF
    END DO
  END DO
END DO
!$acc end parallel loop
```

Plot generated from this example

Code Example With CLAW

Original Subroutine w/ CLAW

```
!$claw acc parallel loop gang vector collapse(2)
DO k=1,ntrac
  !$claw begin loop-hoist(j,i) loop-interchange &
  !$claw reshape(zmd(0),zsediflux(1,j))
  IF (trlist%ti(k)%nsedi==0) CYCLE
  imod=trlist%ti(k)%mode
  slinnfac=sigma(imod)**(2._dp*sigmaln(imod))
  IF (trlist%ti(k)%nphase==AEROSOLNUMBER) THEN
    DO j=1,klev
      DO i=1,kproma
        zmd(i,j)=MIN(rwet(imod)%ptr(i,j,krow)*2._dp,50.E-6_dp)
      END DO
    END DO
  END IF
  DO j=1,klev
    DO i=1,kproma
      zsediflux(i,j)=0._dp
      IF (zmd(i,j)>0._dp) THEN
        zxtp1=pxtm1(i,j,k)+pxtte(i,j,k)*time_step_len
        zvsedi=(2._dp/9._dp*&
          (densaer(imod)%ptr(i,j,krow)-zrho(i,j)) &
          *grav/zvis(i,j)*(zmd(i,j)/2._dp)**2._dp) &
          *(slinnfac+1.246_dp*2._dp*zlair(i,j) &
          /zmd(i,j)*exp((0.5_dp*sigmaln(imod)**2._dp)))
        zsedtend=zxtp1*zvsedi/zdz(i,j))
        pxtte(i,j,k)=pxtte(i,j,k)-zsedtend
        zsediflux(i,j)=zsedtend*zdpg(i,j)
      END IF
    END DO
  END DO
  DO j=2,klev
    DO i=1,kproma
      pxtte(i,j,k)=pxtte(i,j,k)+(zsediflux(i,j-1)/zdpg(i,j))
    END DO
  END DO
  !$claw end
END DO
!$claw acc end parallel loop
```

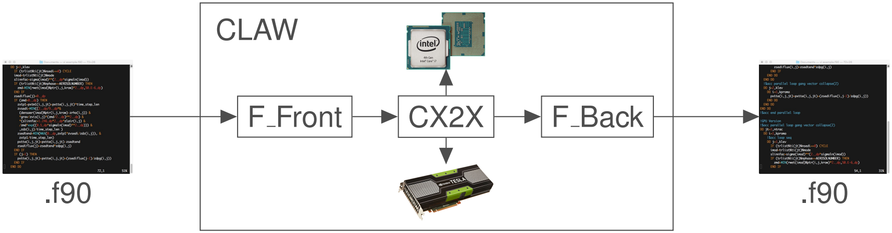
Generated Subroutine

```
!$acc parallel loop gang vector collapse(2)
DO k=1,ntrac
  DO i=1,kproma
    DO j=1,klev
      IF (trlist%ti(k)%nsedi==0) CYCLE
      imod=trlist%ti(k)%mode
      slinnfac=sigma(imod)**(2._dp*sigmaln(imod))
      IF (trlist%ti(k)%nphase==AEROSOLNUMBER) THEN
        zmd=MIN(rwet(imod)%ptr(i,j,krow)*2._dp,50.E-6_dp)
      END IF
      zsediflux(j)=0._dp
      IF (zmd>0._dp) THEN
        zxtp1=pxtm1(i,j,k)+pxtte(i,j,k)*time_step_len
        zvsedi=(2._dp/9._dp*&
          (densaer(imod)%ptr(i,j,krow)-zrho(i,j)) &
          *grav/zvis(i,j)*(zmd/2._dp)**2._dp) &
          *(slinnfac+1.246_dp*2._dp*zlair(i,j) &
          /zmd*exp((0.5_dp*sigmaln(imod)**2._dp)))
        zsedtend=zxtp1*zvsedi/zdz(i,j))
        pxtte(i,j,k)=pxtte(i,j,k)-zsedtend
        zsediflux(j)=zsedtend*zdpg(i,j)
      END IF
      IF (j>1) THEN
        pxtte(i,j,k)=pxtte(i,j,k)+(zsediflux(j-1)/zdpg(i,j))
      END IF
    END DO
  END DO
END DO
!$acc end parallel loop
```

1. Loops hoisted
2. Loop interchange
3. Arrays reshaped

How is CLAW Implemented?

- CLAW is developed by utilizing the OMNI source-to-source compiler from the Riken Institute
- Currently code exists in original CPU optimal state with CLAW directives for GPU-centric transformations
- OMNI compiler front-end generates an intermediate code representation with abstract syntax tree where transformations are performed in JAVA and the back-end generates transformed Fortran code

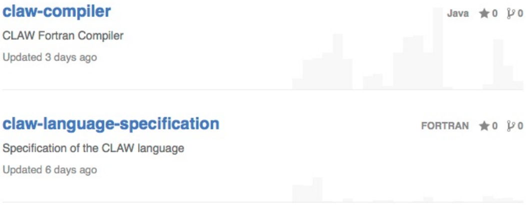


```
DO j=1,klev
  IF (trlist%ti(jt)%nsedi==0) CYCLE
  imod=trlist%ti(jt)%mode
  slinnfac=sigma(imod)**(2._dp*sigmaln(imod))
  IF (trlist%ti(jt)%nphase==AEROSOLNUMBER) THEN
    zmd=MIN(rwet(imod)%ptr(i,j,krow)*2._dp,50.E-6_dp)
  END IF
  zsediflux(j)=0._dp
  IF (zmd>0._dp) THEN
    zxtp1=pxtm1(i,j,jt)+pxtte(i,j,jt)*time_step_len
    zvsedi=MIN((2._dp/9._dp*&
     (densaer(imod)%ptr(i,j,krow)-zrho(i,j)) &
     *grav/zvis(i,j)*(zmd/2._dp)**2._dp) &
     *(slinnfac+1.246_dp*2._dp*zlair(i,j) &
     /zmd*exp((0.5_dp*sigmaln(imod)**2._dp))) &
     ,zdz(i,j)/time_step_len )
    zsedtend=MIN(MAX(0._dp,zxtp1*zvsedi/zdz(i,j)), &
     zxtp1/time_step_len)
    pxtte(i,j,jt)=pxtte(i,j,jt)-zsedtend
    zsediflux(j)=zsedtend*zdpg(i,j)
  END IF
  IF (j>1) THEN
    pxtte(i,j,jt)=pxtte(i,j,jt)+(zsediflux(j-1)/zdpg(i,j))
  END IF
END DO
                                                72,1          51%
```

```
        zsediflux(i,j)=zsedtend*zdpg(i,j)
      END IF
    END DO
  END DO
  !$acc parallel loop gang vector collapse(2)
  DO j=2,klev
    DO i=1,kproma
      pxtte(i,j,jt)=pxtte(i,j,jt)+(zsediflux(i,j-1)/zdpg(i,j))
    END DO
  END DO
END DO
!$acc end parallel loop

!GPU Version
!$acc parallel loop gang vector collapse(2)
DO jt=1,ntrac
  DO i=1,kproma
    !$acc loop seq
    DO j=1,klev
      IF (trlist%ti(jt)%nsedi==0) CYCLE
      imod=trlist%ti(jt)%mode
      slinnfac=sigma(imod)**(2._dp*sigmaln(imod))
      IF (trlist%ti(jt)%nphase==AEROSOLNUMBER) THEN
        zmd=MIN(rwet(imod)%ptr(i,j,krow)*2._dp,50.E-6_dp)
      END IF
                                                54,1          31%
```

Where is CLAW Available?



claw-compiler
CLAW Fortran Compiler
Updated 3 days ago
Java ★0 0

claw-language-specification
Specification of the CLAW language
Updated 6 days ago
FORTRAN ★0 0

https://github.com/C2SM-RCM

CLAW Progression

- CLAW development is iterating from the ‘bottom-up’
- Have begun developing most useful transformations for GPUs already seen in COSMO
- Investigating and verifying use of these transformations and searching for more in ECHAM/HAMMOZ and ICON, as well as COSMO
- Currently can perform loop reordering, loop fusion, array demotion to scalar, code removal, and vector notation to loops
- Working on loop hoisting/lowering and reshaping arrays

CLAW Progression

- After building up sufficient transformation capabilities, higher level abstractions should become possible and are desired
- Will want to expand beyond only CPU $\leftrightarrow$ GPU transformations into more architectures
- Interested in possible connection to other high level abstraction projects (DSLs, DSAs)
- Could CLAW ultimately become a transformation knowledgebase?

Questions?