An Optimization Experiment with the Community Land Model on the Cray X1

James B. White III (Trey)
whitejbiii@ornl.gov

Acknowledgement

Research sponsored by the Mathematical, Information, and Computational Sciences Division, Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

Thanks to OSC for SV1 access.
Love those loopmarks!

Fun with CLM on X1

- Community Land Model
- Modifications
- Results
- Conclusions and Prospects

Community Land Model (CLM)

- Component of Community Climate System Model (CCSM)
- Part of Community Atmospheric Model (CAM)
- Can run stand-alone
- Fortran with MPI and OpenMP

CLM 2.1 nested types

- Grid cells
  - Same horizontal grid as CAM
- Land units
  - Percentages of grid cell, not contiguous region
  - Soil properties, landcover types (lakes, glaciers)
- Columns
  - Soil states, fluxes with atmosphere
- Plant functional types (PFTs)
  - Compete for column resources

CLM 2.1 data structures

- Nested user-defined types
  - Embedded pointers to sub-types
  - Pointer arrays to sub-levels
  - Pointers up to parent level
  - All defined in "`clmtype.F90`"
- "Point" types
  - One-dimensional arrays for each level
  - Nested types point into subsections of these
  - All defined in "`clmpoint.F90`"

CLM 2.1 column type

```
type column_type
   ! pointer to the next level in hierarchy
   type(pft_type), dimension(:),pointer:: p
   …
   ! pointers to higher level in hierarchy
   …
   ! conservation check structures for the column level
   type(energy_balance_type) :: cebal   !energy balance structure
   …
   ! state variables defined at the column level
   type(column_pstate_type) :: cps   !column physical state
   variables
   …
   ! flux variables defined at the column level
   type(column_eflux_type) :: cef    !column energy flux
   …
end type column_type
```

CLM 2.1 column type

```
type column_pstate_type
   type(landunit_pstate_type), pointer:: lps
      !pointer to higher level ps struct
   type(pft_pstate_type):: pps_a
      !pft-level pstate variables averaged to the column
   …
   real(r8):: albgrd(numrad)          !ground albedo (direct)
   real(r8):: albgri(numrad)          !ground albedo (diffuse)
   …
   real(r8):: wt
      !weight (relative to landunit) for this column (0-1)
   real(r8):: wtxy
      !weight (relative to gridcell) for this column (0-1)
   integer :: index1d          !index into 1d global column array
end type column_pstate_type
```

CLM 2.1 column type

```
type col_single_pointer
   type(column_type), pointer :: c
end type col_single_pointer

type col_array_pointer
   type (col_single_pointer), dimension(:), pointer :: col
end type col_array_pointer

type (col_array_pointer) :: cpoint
```

So…

```
c <=> cpoint%col(c%cps%index1d)%c
```

CLM driver

```
!$OMP PARALLEL DO PRIVATE (ci,c)
  do ci = cols1d%beg,cols1d%end
     c => cpoint%col(ci)%c
     if (.not. c%cps%lps%lakpoi) then
        …
        call Biogeophysics1(c)
     else if (c%cps%lps%lakpoi) then
        …
     endif
     …
  end do ! end column loop
```

Subroutine structure

- Pass single column argument, "c"
- Declare (many!) local pointers
- Associate local pointers with contents of "c"
- Compute with local pointers as shorter aliases

Using a variable

```
type (column_type),target,intent(inout):: c
type(column_pstate_type),pointer :: cps
type(landunit_pstate_type),pointer :: lps
real(r8),dimension(:),pointer:: sucsat
…
cps => c%cps
lps => cps%lps
sucsat => lps%sucsat
…
psit = -sucsat(1) * fac ** (-bsw(1))
```

Experiment

- Current code vectorizes very little
- Can we?
  - Modify data structures
  - Keep maintainability and extensibility
  - Keep superscalar performance
  - Get vectorization and multistreaming
- Try this:
  - Modify "`Biogeophysics1`" tree
    - Top "`driver`" call in profile on IBM p690 (Power4)
  - Compare performance of old and new "`Biogeophysics1`"

Modified Data Structures

- No user-defined types
- Major types become modules
  - Grid cells, land units, columns, PFTs
- "Flatten" remaining types
  - Scalar variables become 1D arrays
  - Arrays add a leading dimension

Modified Data Structures

- Add single-character prefix to variables
  - "g", "l", "c", "p"
  - Fixes name conflicts from flattening
  - Avoids indexing errors (more later)
- Relationships between types implemented as index arrays
  - Example: Integer array "pcolumn" gives column index associated with each PFT

Data Layout

- "`if`" tests in "`driver`" for lake and non-lake points
  - Most subroutines use only non-lake points or only lake points
  - Column and PFT arrays sorted with non-lake points all first
  - Pick lake or non-lake points simply with array bounds

Data Layout

- Column reductions of PFT variables
  - Sums or averages for all PFTs in a given column
  - Currently one PFT per column
  - Multiple PFTs per column in near future (competition among PFTs for column resources)
- Arrange PFTs in independent-column blocks
  - Collect each column's first PFT into a block
  - Then a block of each column's second PFT, *etc*.
  - Vectorize & multistream column updates over each block

Modified Subroutines

- `Biogeophysics1`
  - `SurfaceRadiation`
  - `BareGroundFluxes`
    - `FrictionVelocity`
      - `StabilityFunc` (scalar optimizations)
  - `CanopyFluxes`
    - `FrictionVelocity`
      - `StabilityFunc` (scalar optimizations)

Biogeophysics1: Original call

```
do ci = cols1d%beg,cols1d%end
   c => cpoint%col(ci)%c
   if (.not. c%cps%lps%lakpoi) then
      …
      call Biogeophysics1(c)
   else if (c%cps%lps%lakpoi) then
      …
   endif
   …
end do ! end column loop
```

Initial modification

- Move column loop inside
- Pass non-lake bounds
- "Pass" other variables through modules

```
call biogeophysics1v( &
  clb_nl, cub_nl, plb_nl, pub_nl)
```

Refined modification

- Loop over blocks
- "`bn`" becomes tuning parameter

```
do b = clb_nl, cub_nl, bn
   call biogeophysics1v( &
     b, min(b+bn-1,cub_nl), &
     b, min(b+bn-1,pub_nl))
end do
```

Tuning Parameter

- Stride == vector size
- Small stride
  - Cache-dependent superscalar systems
- Full-size stride - one loop iteration
  - Vector-only systems
- Large stride - four loop iterations
  - Vector systems with multistreaming, OpenMP
  - Stream and/or thread over outer loop

Biogeophysics1

- Column loop over input bounds
- Reads variables from grid cell and land unit
  - Uses index arrays to find right ones
- Indices follow prefix convention
  - Variable and index prefixes must match
  - Avoids indexing variables with wrong indices
  - ***Not enforced by the compiler***

Biogeophysics1 loop

```
do ci = clb, cub
   li = clandunit(ci)
   gi = cgridcell(ci)
   …
   cthm(ci) = gforc_t(gi) + &
        0.0098*gforc_hgt_t(gi)
```

SurfaceRadiation

- Originally a loop over column PFTs
- Easily modified to loop over PFT bounds provided as arguments
- Imperfectly nested inner loop over radiation bands (just 2 bands)
  - Vectorization of outer loop required unrolling inner loop
    `!dir$ unroll(nband)`
  - "`nband`" argument required

To if or not to if?

- "`Biogeophysics1`" calls "`BareGroundFluxes`" ***-or-*** "`CanopyFluxes`" for each PFT
- Move "`if`" test inside each subroutine
  - One test switched to "`.not. …`"
- Implement "`if`" around large blocks of code?
  - Standard "`if`" generates masked ops
  - Many elements are false, much time is wasted
- ***Use a filter!***

Filter

- Construct filter index array using "`if`" test
  - Vectorizes! (But doesn't multistream.)
- Iterate over elements of filter
  - Use filter to set PFT index
- Tell compiler the filter array has no dependencies
  - Declare filter (just once) as "`!dir$ permutation`" - vectorizes
  - Declare each loop as "`!dir$ concurrent`" - multistreams

Filter

```
fn = 0
do pi = plb, pub
   if (<test>) then
      fn = fn+1
      filterp(fn) = pi
   end if
end do
do fi = 1, fn
   pi = filterp(fi)
   ci = pcolumn(pi)
   gi = pgridcell(pi)
   …
```

BareGroundFluxes

- Filtered PFT loop
  - Inner iteration loop with static bounds
    - Call "FrictionVelocity"
      - Doesn't inline (Thanks to "if-else-else" trees in "FrictionVelocity"?)

BareGroundFluxes

- Push PFT loop down into "FrictionVelocity"
- PFT loop in "BareGroundFluxes" must be split
- Scalar temporaries become arrays
  - Automatic arrays using input bounds
  - Performance issue? Memory issue? (Not so far)

FrictionVelocity

- Original "FrictionVelocity" has three of these:

```
if (zeta < -zetam) then
  ustar = …
else if (zeta < 0.) then
  ustar = …
else if (zeta <= 1.) then
  ustar = …
else
  ustar = …
endif
```

`FrictionVelocity`

- Each calculation is expensive
  - Logs, fractional powers
- Masked operations would be inefficient
- Use filters!
  - Initial filter passed as argument
  - Local filters implement "`if`" series
    - Vectorizes but doesn't multistream
  - Computation loop over each filter
    - Vectorizes and multistreams

```
flnzn = 0
fl0n = 0
fle1n = 0
felsen = 0
do fi = 1, fn
   pi = filterp(fi)
   if (pzeta(pi) < -zetam) then
      flnzn = flnzn + 1
      flnz(flnzn) = pi
   else if (pzeta(pi) < 0.) then
      fl0n = fl0n + 1
      fl0(fl0n) = pi
   else if (pzeta(pi) <= 1.) then
      fle1n = fle1n + 1
      fle1(fle1n) = pi
   else
      felsen = felsen + 1
      felse(felsen) = pi
   end if
end do
```

`CanopyFluxes`

- Filtered PFT loop
- Also an iteration loop
  - "`do while`"
  - Different number of iterations for each PFT
  - ???
- Solution: ***Another filter!***

Irregular iteration

- Initialize iteration filter to input filter
- Perform iteration
- Reconstruct iteration filter
  - Use test from original "`do while`"
  - Weed out "`.false.`" elements
- Iterate until filter has no elements
- Restore input filter

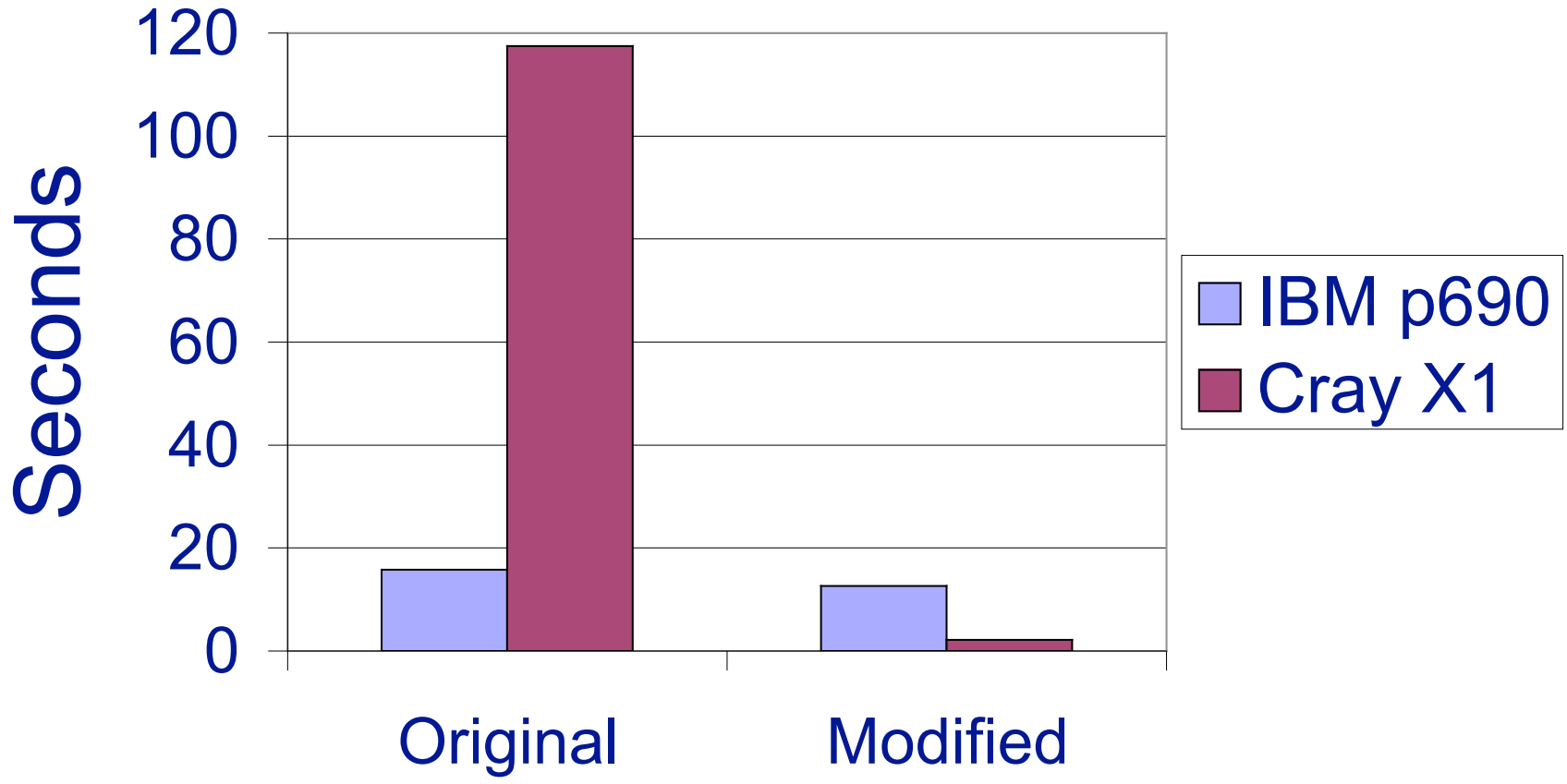
Performance Tradeoffs

- Original code has temporal locality
- Modified code has spatial locality
  - Enhance temporal locality using small blocks
- Filters, filters, filters
  - Critical for vectorization
  - Unnecessary overhead for superscalar

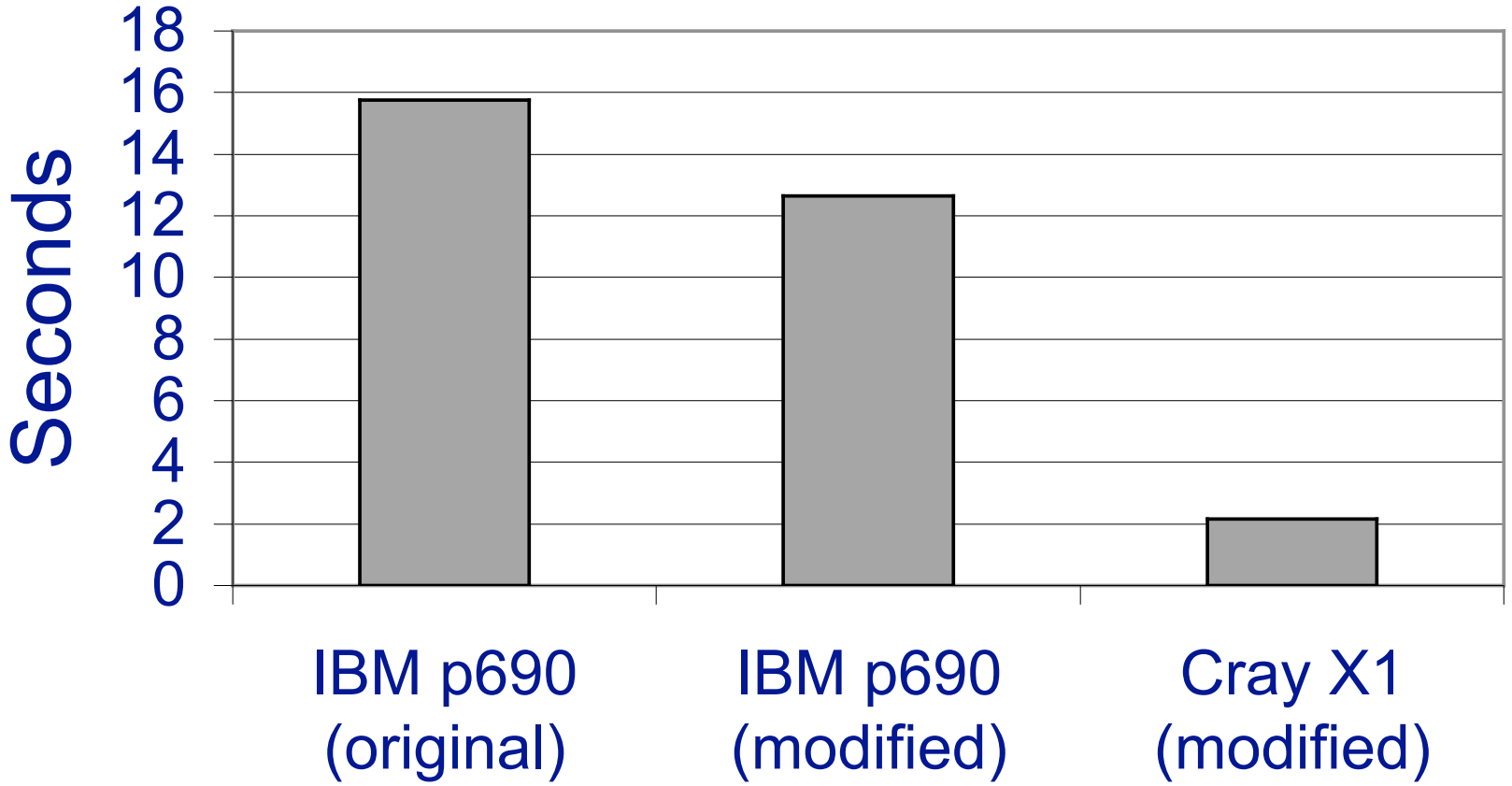
Experiment

- Stand-alone CLM
  - 48 time steps, standard input data
- IBM p690
  - One Power4 processor, 1.3 GHz
  - 16-element outer-loop stride (same as cache line)
- Cray X1
  - One MSP, 800 MHz
  - 4-iteration outer loop (for multistreaming)

Biogeophysics1 Runtime (48 timesteps)



Biogeophysics1 Runtime (48 timesteps)



Maintainability? Extensibility?

- New sources of error
  - Index prefix must match variable prefix
  - Indices at other levels must be set each time
    - Don't use "`gi`" without first reading it from "`pgridcell(pi)`"
- Some errors eliminated
  - Same local pointer associated with variables at different levels in the same subroutine
  - What if you associate it to the wrong one?
  - What if you didn't notice it was re-associated?

Maintainability? Extensibility?

Before

```
type (column_type),target,intent(inout):: c
type(column_pstate_type),pointer :: cps
type(landunit_pstate_type),pointer :: lps
real(r8),dimension(:),pointer:: sucsat
…
cps => c%cps
lps => cps%lps
sucsat => lps%sucsat
…
psit = -sucsat(1) * fac ** (-bsw(1))
```

Maintainability? Extensibility?

After

```
psit = -lsucsat(li,1) * fac ** (-lbsw(li,1))
```

Size of code
(minus comments and blanks)

	Lines			Characters		
	Original	Modified	Ratio	Original	Modified	Ratio
`Biogeophysics1`	225	98	0.44	4783	2234	0.47
`SurfaceRadiation`	119	61	0.51	2527	1344	0.53
`BareGroundFluxes`	205	138	0.67	4512	3122	0.69
`CanopyFluxes`	505	360	0.71	11179	9689	0.87
`FrictionVelocity`	120	248	2.07	2862	4943	1.73
Total	**1174**	**905**	**0.77**	**25863**	**21332**	**0.82**

Conclusions

- Modifications improve performance of "`Biogeophysics1`" tree on IBM p690 by 20%
- Modified tree on Cray X1 runs:
  - 5.86x faster than modified on IBM p690
  - 7.29x faster than original on IBM p690
- Modified subroutines have:
  - 23% fewer lines of code
  - 18% fewer characters

Prospects

- Forrest Hoffman at ORNL performing complementary experiments that keep original data structure
  - Pushing limits of the Cray compiler
  - Little success so far
- May have strong argument for modifying full CLM

An Optimization Experiment with the Community Land Model on the Cray X1

James B. White III (Trey)
whitejbiii@ornl.gov