Many-core optimization of the Community Earth System Model

John Dennis, Chris Kerr, Youngsung Kim, Raghu Kumar, Rashmi Oak, Amogh Simha

Application Scalability and Performance Group, CISL
Outline

• Motivation (See other presentations)
• Group/Team
• Three Optimization Tales
  – Eulerian Advection
  – MG2
  – RRTMG
• Lessons Learned
• Conclusions
Group/Team

- Rich Loft, Division Director (NCAR)
- John Dennis, Scientist (NCAR)
- Chris Kerr, Software Engineer, contractor
- Youngsung Kim, Software Engineer (NCAR) /Graduate Student (CU)
- Raghu Raj Prasanna Kumar, Associate Scientist (NCAR)
- Amogh Simha, Graduate Student (CU)
- Rashmi Oak, Graduate Student (CU)
- Nitin Bhat, Graduate Student (Indian Institute for Science [IISc])
- Ravi Nanjundiah, Professor (IISc)
Related/Collaborative Activities

- Funding from Intel Parallel Computing Center (IPCC-WACS)
- NESAP (NERSC Exascale Science Application Program)
  - Bi-weekly: NERSC-Cray-NCAR telecon on CESM & HOMME performance (Feb 2015)
- Weekly Intel-TACC-NREL-NERSC-NCAR telecon
  - Concalf focused on CESM/HOMME KNC performance
- Strategic Parallel Optimization of Computing
  - NCAR effort focused on Xeon architectures
  - SPOC currently focused on MPAS
Current optimization focus

Xeon and Xeon Phi based platforms

- Sandybridge (SNB) [i.e. Yellowstone]
- Ivybridge (IVB) [i.e. Edison]
- Haswell (HSW)
- Knights Corner (KNC) [i.e. Babbage]

Compilers

- Intel 15.0.1 or 15.0.3
- Cray 8.3 or 8.4
CAM5-SE+MG2
SNB, ne=16 (8x2:120x2)
Intel 14.0.2, OPT=02, sec/day = 26.7

60% of CAM
Three Optimization Tales
(1) HOMME: Eulerian Advection
HOMME overview

- High Order Methods Modeling Environment (HOMME)
  - CAM: 35% of time (plev=30, qsize=25)
  - WACCM: 88% of time (plev=70, qsize=135)

- Default advection algorithm in HOMME
  - 2\textsuperscript{nd} order Runge-Kutta
  - Memory hierarchy bandwidth limited
  - Expensive:
    - CAM: 23% of time
    - WACCM: 81% of time
HOMME advection (CAM-like): L3 miss rates

Why are we missing so badly in L3?

![Graph showing L3 miss rates over time]
Optimization steps

1. Threading memory copy in boundary exchange [Jamroz]
2. Restructure data-structures for vectorization [Vadlamani & Dennis]
3. Rewrite message passing library/ specialized comm ops [Dennis]
4. Rearrange calculations in euler_step for cache reuse [Dennis]
5. Reduced # of divides [Dennis]
6. Restructured/alignment for better vectorization [Kerr]
7. Rewrote and optimized limiter [Demeshko & Kerr]
HOMME (NE=6, PLEV=26, qsize=25) [single node]

25% reduction in cost!
HOMME (NE=6, PLEV=70, qsize=135)[single node]

44-50% reduction in cost!
(2) Morrison Gettelman
microphysics v2
Morrison Gettelman microphysics version 2 (MG2) kernel

- Need code to test optimization techniques
- Characteristics
  - Not too large/not too small
  - Non-trivial
  - Relevant to CESM science plan
  - Expensive (10-13% of CAM)
  - Easily accessible
  - Well written
- Engage vendor experts
- Refine optimization techniques
MG2 kernel (con’t)

• Need to optimize kernel rapidly to minimize version skew
  – Given subversion URL: Nov 2014
  – Generate kernel using KGEN: Dec 2014
  – Public release of kernel: Jan 2015
  – Optimized version reintegrated: May 2015

• Optimization work (Kerr & Santos)

• Used in NWSC-2 benchmark efforts
MG2 kernel ($\mu$sec) [single thread]

52-57% in cost!
How was MG2 optimized?

- Eliminated use of elemental functions
- Pushed vector loop into lower into call stack
- Eliminated divides where possible
- Avoided using slow vendor gamma function
- Rewrote code to minimized use of gamma functions (Santos)
- Rewrote the sedimentation loop (Santos)
- Remove initialization of variables that are overwritten
How was MG2 optimized (con’t)

• Eliminate use of automatic arrays
• Rearrange loops to allow alignment
• Use more aggressive compiler optimizations
(3) Rapid Radiation Transport Model for GCM (RRTMG)

CAM variant
Rapid Radiative Transport Model (RRTMG)

- Produced by Atmospheric & Environmental Research (AER)

- Multiple versions:
  - RRTMG-{WRF,MPAS}
  - RRTMG-fast-WRF [Michalakes]
  - RRTMG-CAM (PORT)
  - RRTMG-ECHAM (PSrad)

- Opportunity to examine different design choices!
Optimization strategy for RRTMG

- Push column loop down call tree (Kumar)
- Optimize random number generator (Kerr)
- Restructure branchy lookup table calls in gas-phase optics (Kumar & Loft)
- Align and vectorize solver (Kerr)
- Vectorize other code (Kumar, Kim, Oak, Dennis)
RRTMG-SW kernel (μsec) [single thread]

33-39% reduction in cost!
Reintegration Status

- **MG2**
  - Reintegrated into CESM as of [May 2015]

- **HOMME advection**
  - On the HOMME trunk
  - Not yet used in CAM [target: Oct 1, 2015]

- **RRTMG**
  - Optimized RNG [Status unknown]
  - Verification testing underway [target: Oct 1, 2015]
Lessons from Optimization

• Close vendor engagement is critical
  – Intel
    • 2+ years
    • Weekly or bi-weekly concalls
    • Easy access to large number compiler/performance experts
  – Cray
    • 7 months
    • Bi-weekly concalls
    • Focus on correctness/performance issues [Wagner]
  – PGI
    • 1+ years
    • Fortran standards compliance
Lessons from Optimization

• Need for multiple different performance analysis tools
  – General identification (GPTL)
  – Diagnose performance issue (extrae, vTune)
  – Optimization phase (gprof, vprof, system_clock)

• Optimizations involve many phases
  – Fixing one issue reveals another
  – Keep going until you are sick of it! 😊
Lessons from Optimization (con’t)

- Multiple “eyes” are most effective
  - MG2
  - HOMME advection

- Easily creating kernels is critical (KGEN)
  - Simplifies vendor collaboration
  - Parallelization of optimization effort
  - Helps with rapidly evolving code base

- Excessive amount of compiler hand-holding!
- Process not finished until reintegration complete!
Future work

- Continue Cori preparation
- Dissect KNL performance
- KGEN refinement/collaboration
- Need more parallelism from HOMME advection!
- More HOMME communication library optimization
- More code optimization (CLUBB)
Questions?

dennis@ucar.edu
## Long-wave rad. in PORT and PSrad
(sequential - one thread)

<table>
<thead>
<tr>
<th>PORT*</th>
<th>PSrad</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configuration</strong></td>
<td><strong>Configuration</strong></td>
</tr>
<tr>
<td>- NX=144, NY=96, NLEV=26, NCOL=16</td>
<td>- LAT=96, LON=192, NLEV=47, KPROMA=16</td>
</tr>
<tr>
<td><strong>SUBROUTINE</strong> rad_rrtmg_lw()</td>
<td><strong>SUBROUTINE</strong> lrtm()</td>
</tr>
<tr>
<td>call mcica_subcol_lw()</td>
<td>call sample_cld_state()</td>
</tr>
<tr>
<td>do iplon = 1, ncol</td>
<td>call lrtm_coeffs()</td>
</tr>
<tr>
<td>- call inatm()</td>
<td>do ig = 1, n_gpts_ts</td>
</tr>
<tr>
<td>- call cldprmc()</td>
<td>do jl = 1, kproma</td>
</tr>
<tr>
<td>- call setcoef()</td>
<td>- call gas_optics_lw()</td>
</tr>
<tr>
<td>- call taumol()</td>
<td>end do</td>
</tr>
<tr>
<td>- call rtrnmc()</td>
<td>end do</td>
</tr>
<tr>
<td>end do</td>
<td>do ig = 1, nbndlw</td>
</tr>
<tr>
<td></td>
<td>- planckFunction()</td>
</tr>
<tr>
<td></td>
<td>end do</td>
</tr>
<tr>
<td></td>
<td>do ig = 1, n_gpts_ts</td>
</tr>
<tr>
<td></td>
<td>- call lrtm_solver()</td>
</tr>
<tr>
<td></td>
<td>end do</td>
</tr>
</tbody>
</table>

* : Parallel Offline Radiative Transfer
What code look like before and after changes

After changes:

```fortran
function wv_sat_svp_to_qsat(es, p, mgncol) result(qs)
    integer, intent(in) :: mgncol
    real(r8), dimension(mgncol), intent(in) :: es  ! SVP
    real(r8), dimension(mgncol), intent(in) :: p   ! Current pressure.
    real(r8), dimension(mgncol) :: qs
    integer :: i
    do i=1,mgncol
        ! If pressure is less than SVP, set qs to maximum of 1.
        if ( (p(i) - es(i)) <= 0._r8 ) then
            qs(i) = 1.0_r8
        else
            qs(i) = epsilo*es(i) / (p(i) - omeps*es(i))
        end if
    enddo
end function wv_sat_svp_to_qsat
```
What code look like before and after changes

Before changes:

```fortran
 elemental function wv_sat_svp_to_qsat(es, p)
    result(qs)

    real(r8), intent(in) :: es  ! SVP
    real(r8), intent(in) :: p   ! Current pressure.
    real(r8) :: qs

    ! If pressure is less than SVP, set qs to maximum of 1.
    if ( (p - es) <= 0._r8 ) then
        qs = 1.0_r8
    else
        qs = epsilo*es / (p - omeps*es)
    end if

end function wv_sat_svp_to_qsat
```
MG2 kernel [single thread]
MPI task x OMP threads
HOMME (NE=6, PLEV=26, qsize=25) [single HSW2 node]