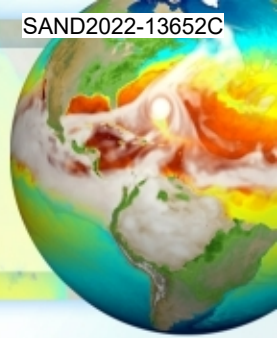
SAND2022-13652C



RRTMGPxx: a portable radiation code for ultra-high-resolution atmosphere modeling

Benjamin R. Hillman, Matthew Norman, Luca Bertagna, Aaron Donahue, Peter Caldwell

E3SM Energy Exascale Earth System Model

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

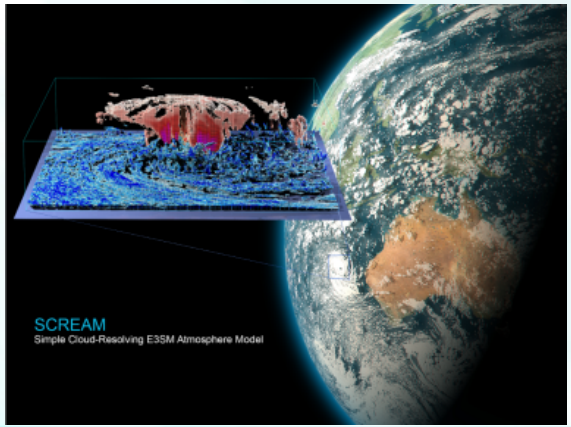
U.S. DEPARTMENT OF ENERGY

Acknowledgements

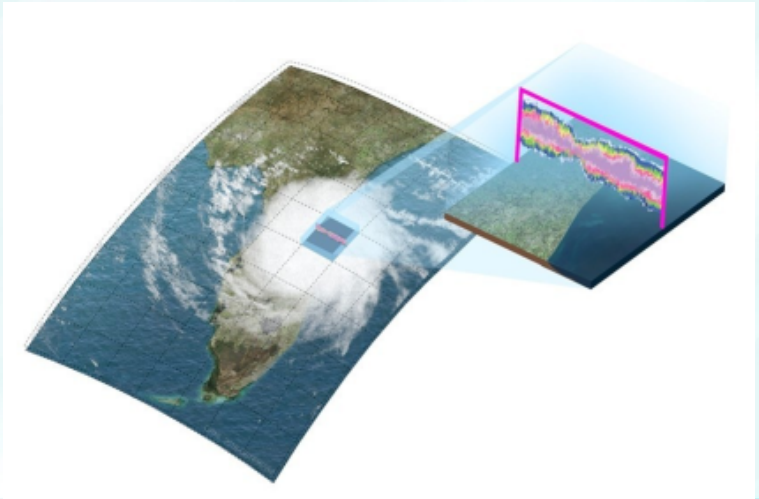
- This research was supported as part of the Energy Exascale Earth System Model (E3SM) project, funded by the U.S. Department of Energy, Office of Science, Office of Biological and Environmental Research
- This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.
- This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

High resolution atmosphere modeling in DOE

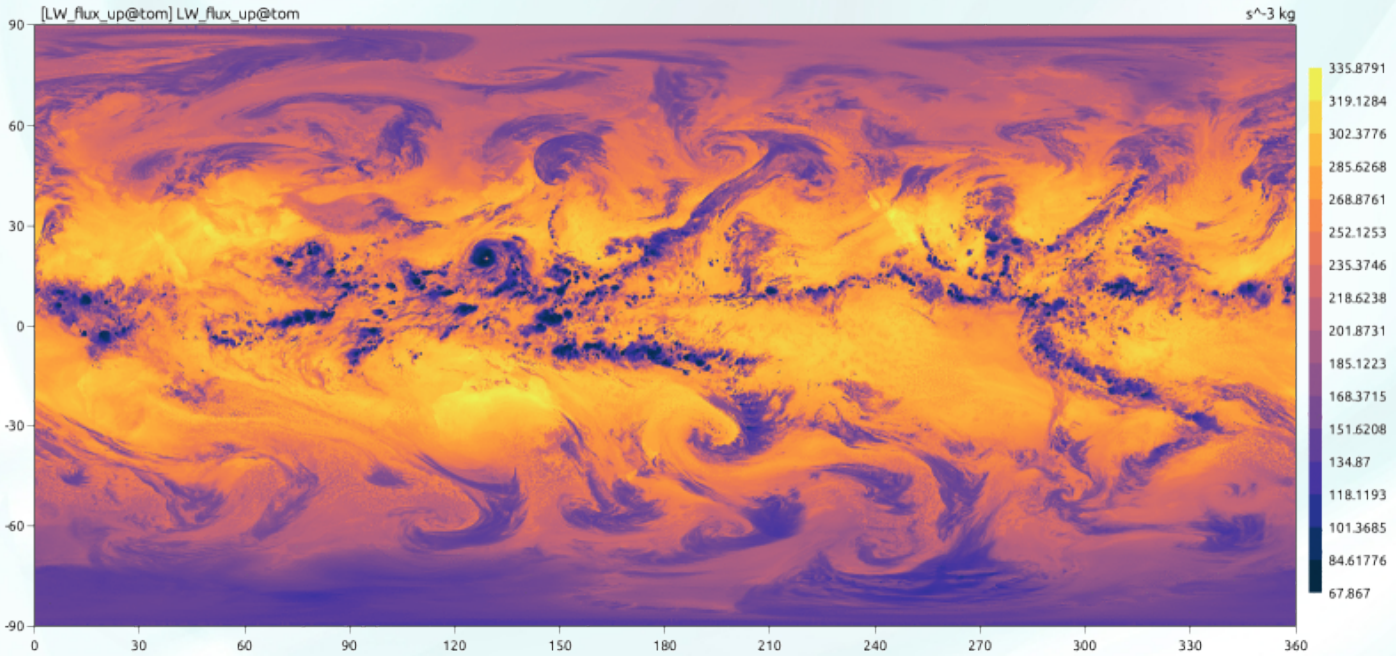
- **EAMxx**: Energy Exascale Earth System Model (E3SM) Atmosphere Model in C++
- **SCREAM**: Simple Cloud Resolving E3SM Atmosphere Model; a 3 km configuration of EAMxx



- **E3SM-MMF**: version of E3SM that implements the Multi-scale Modeling Framework (Hannah et al. 2020)

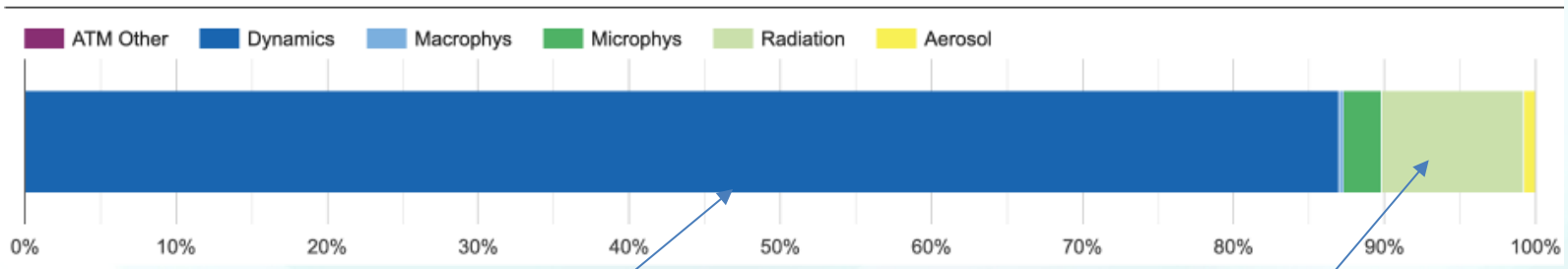


OLR in 3 km SCREAM configuration



Animation created by Chris Terai using ncvis (https://github.com/SEATStandards/ncvis)

Relative costs of processes in EAMxx/SCREAM



Dynamics dominates model cost

Radiative transfer dominates the atmospheric physics cost

Why another radiative transfer package?

- Emerging exascale computers are leveraging GPU accelerators for on-node parallelism
  - Current climate codes need substantial refactoring to effectively use these new resources
- RRTMGP (Pincus et al. 2019): rewrite of RRTMG to expose parallelism
  - Fortran with OpenACC directives (and now OpenMP, and additional set of CUDA kernels) to target NVIDIA GPUs
  - Porting to additional backends *may* require additional directives in user-facing code
- RRTMGPxx: rewrite of RRTMGP to leverage a “performance portability library” in C++
  - Take advantage of excellent vendor support for C++, and robust C-based APIs
  - Enable “single-source” portability to new architectures: RRTMGPxx user-facing code remains unchanged, performance portability library provides multiple backends

Yet Another Kernel Launcher (YAKL)

- Small portable C++ library with 5K lines of core code (1-1.5 FTE effort)
- Emphasis on simple, readable syntax with minimal developer concerns
- Allows Fortran-like behavior in the resulting C++ code
  – Multi-dimensional arrays with column-major ordering and arbitrary lower bounds that default to 1
  – Growing library of Fortran intrinsic functions: size(), maxval(), allocated(), etc.
- An efficient automatically enabled pool allocator for all YAKL allocations
  – Faster frequent allocations and deallocations
- Runs on CPUs, CPU threads (OpenMP 3.5), Nvidia GPUs (CUDA), AMD GPUs (HIP), and Intel GPUs (SYCL)

Example kernels in RRTMGP and RRTMGPxx

```
do igpt = 1, ngpt
  do ilay = 1, nlay
    do icol = 1, ncol
      tau1(icol,ilay,igpt) = tau1(icol,ilay,igpt) + tau2(icol,ilay,igpt)
    end do
  end do
end do
```

Example kernels in RRTMGP and RRTMGPxx

```
do igpt = 1, ngpt
  do ilay = 1, nlay
    do icol = 1, ncol
      tau1(icol,ilay,igpt) = tau1(icol,ilay,igpt) + tau2(icol,ilay,igpt)
    end do
  end do
end do
```

- Tightly nested loops
- Large (problem size) kernels
- Small (code amount) kernels

Example kernels in RRTMGP and RRTMGPxx

```
!$acc  parallel loop collapse(3) &
!$acc&     copyin(tau2(:ncol,:nlay,:ngpt)) &
!$acc&     copy(tau1(:ncol,:nlay,:ngpt))
!$omp target teams distribute parallel do simd collapse(3) &
!$omp& map(to:tau2) &
!$omp& map(tofrom:tau1)
do igpt = 1, ngpt
  do ilay = 1, nlay
    do icol = 1, ncol
      tau1(icol,ilay,igpt) = tau1(icol,ilay,igpt) + tau2(icol,ilay,igpt)
    end do
  end do
end do
```

Multiple backends require different directives embedded in code

- Tightly nested loops
- Large (problem size) kernels
- Small (code amount) kernels

Example kernels in RRTMGP and RRTMGPxx

```
!$acc  parallel loop collapse(3) &
!$acc&     copyin(tau2(:ncol,:nlay,:ngpt)) &
!$acc&     copy(tau1(:ncol,:nlay,:ngpt))
!$omp target teams distribute parallel do simd collapse(3) &
!$omp& map(to:tau2) &
!$omp& map(tofrom:tau1)
do igpt = 1, ngpt
  do ilay = 1, nlay
    do icol = 1, ncol
      tau1(icol,ilay,igpt) = tau1(icol,ilay,igpt) + tau2(icol,ilay,igpt)
    end do
  end do
end do
```

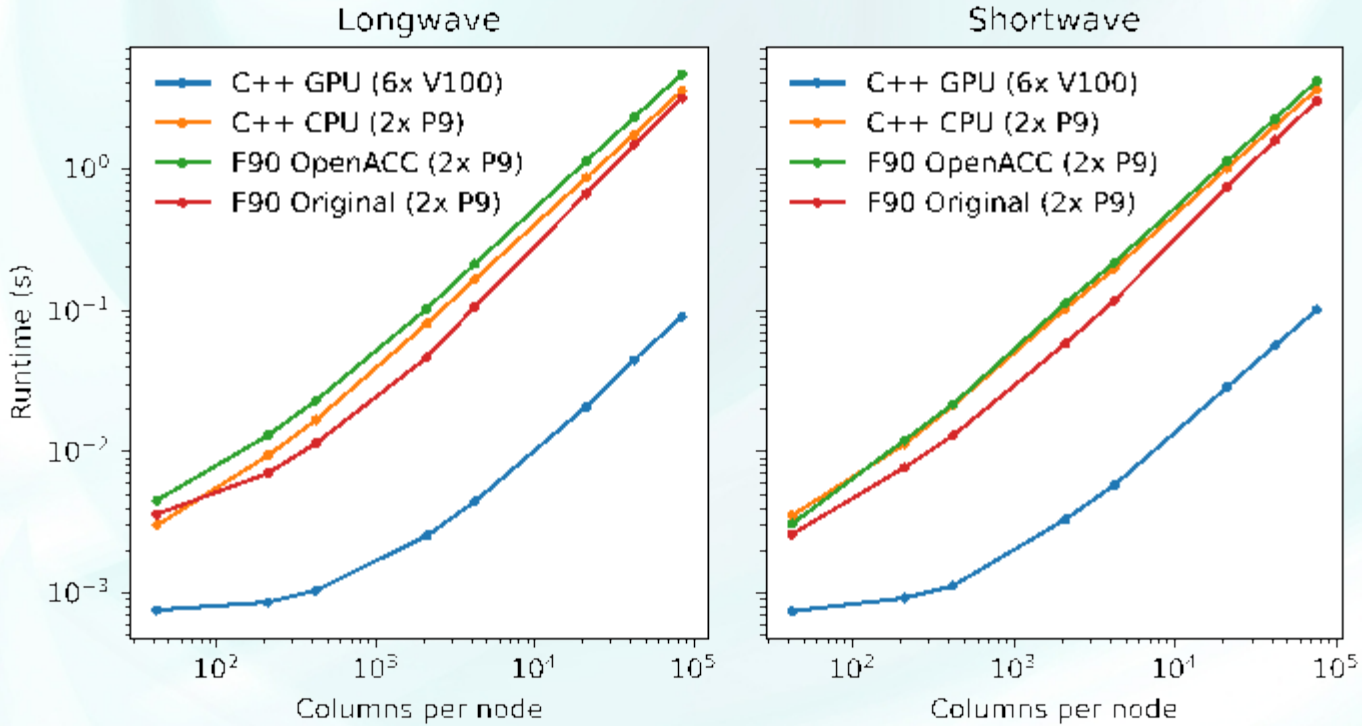
Multiple backends require different directives embedded in code

- Tightly nested loops
- Large (problem size) kernels
- Small (code amount) kernels

Loops reformatted to use yakl parallel_for, but same structure and order; no backend-specific code

```
parallel_for( YAKL_AUTO_LABEL() , SimpleBounds<3>(ngpt,nlay,ncol) , YAKL_LAMBDA (int igpt, int ilay, int icol) {
  tau1(icol,ilay,igpt) = tau1(icol,ilay,igpt) + tau2(icol,ilay,igpt);
});
```

RRTMGPxx performance (standalone)

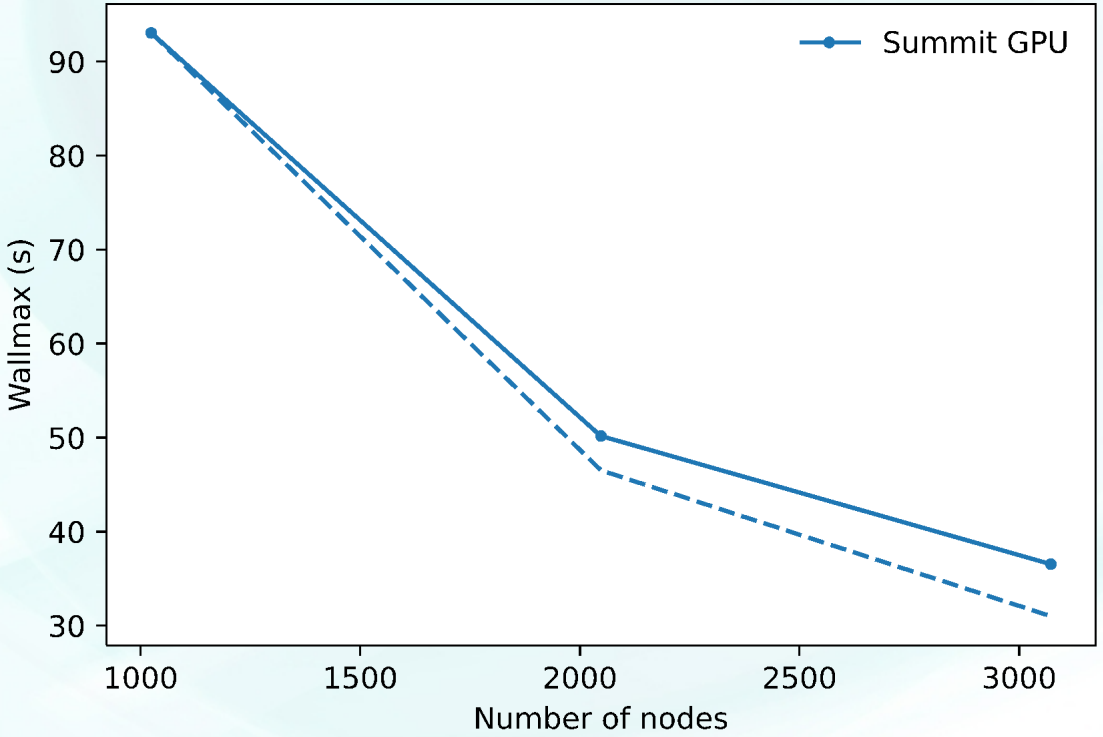


Various RRTMGP ports on Summit (6 v100 GPUs and 2 P9 CPUs per node)

Integration in EAMxx/SCREAM

- New interface layer between EAMxx driver and RRTMGPxx
  - EAMxx mostly uses Kokkos for on-node parallelism, RRTMGPxx uses YAKL
  - YAKL is compatible with Kokkos, but YAKL arrays in RRTMGPxx uses different data layout than Kokkos view in EAMxx; requires copying data between Kokkos and YAKL data structures at runtime
- At 3 km resolution, update radiative heating every 5 model minutes
- MCICA subcolumn sampling to handle subgrid scale cloud overlap

RRTMGPxx scaling in EAMxx at 3 km



Summary

- New port of RRTMGP radiation code in C++ using a performance portability library
- Code is being used in EAMxx at resolutions from 835 km down to 3 km, and in E3SM-MMF (not shown)
- Good performance on Summit GPUs; acceptable performance relative to F90 code on CPU
- Backends available for other machines and initial performance assessments are promising (not shown)

References

- https://github.com/E3SM-Project/rte-rrtmgp (fork of RRTMGP containing C++ code)
- https://github.com/mrnorman/YAKL
- Pincus, R., Mlawer, E. J., & Delamere, J. S. (2019). Balancing accuracy, efficiency, and flexibility in radiation calculations for dynamical models. Journal of Advances in Modeling Earth Systems, 11, 3074– 3089 https://doi.org/10.1029/2019MS001621
- Caldwell, P. M., Terai, C. R., Hillman, B., Keen, N. D., Bogenschutz, P., Lin, W., et al. (2021). Convection-permitting simulations with the E3SM global atmosphere model. Journal of Advances in Modeling Earth Systems, 13, e2021MS002544. https://doi.org/10.1029/2021MS002544
- Hannah, W. M., Jones, C. R., Hillman, B. R., Norman, M. R., Bader, D. C., Taylor, M. A., et al. (2020). Initial Results From the Super-Parameterized E3SM. Journal of Advances in Modeling Earth Systems, 12(1), e2019MS001863. https://doi.org/10.1029/2019MS001863

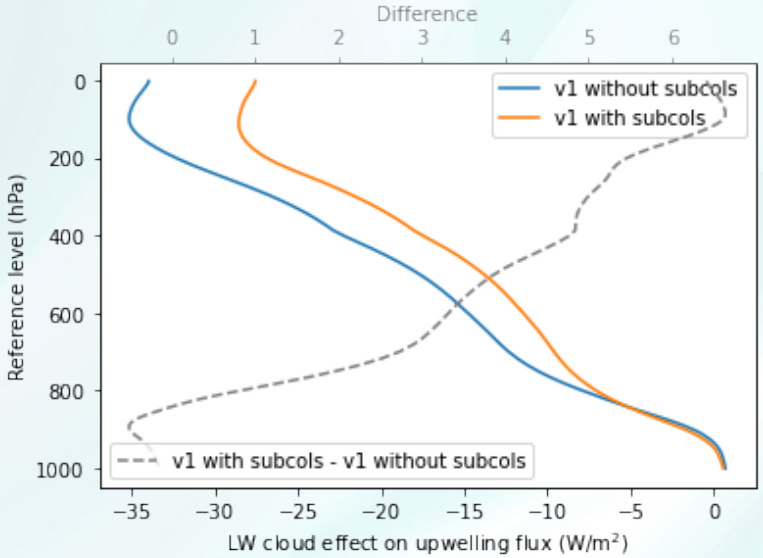
Extra slides

Integration in E3SM-MMF

- E3SM-MMF implements the Multi-scale Modeling Framework
- 2D (or small 3D) CRM embedded into each grid cell of EAM
- RRTMGPxx runs outside the CRM to be able to access aerosol property information
- Radiation run on a coarsened horizontal grid to save additional computational cost

Partial cloudiness

- EAMxx predicts partial cloudiness
- Vertical overlap of partially cloudy layers can have substantial impact on heating profiles



Example kernels in RRTMGP and RRTMGPxx

```
do igpt = 1, ngpt
  do ilay = 1, nlay
    do icol = 1, ncol
      tau1(icol,ilay,igpt) = tau1(icol,ilay,igpt) + tau2(icol,ilay,igpt)
    end do
  end do
end do
```

Example kernels in RRTMGP and RRTMGPxx

```
do igpt = 1, ngpt
  do ilay = 1, nlay
    do icol = 1, ncol
      tau1(icol,ilay,igpt) = tau1(icol,ilay,igpt) + tau2(icol,ilay,igpt)
    end do
  end do
end do
```

- Tightly nested loops
- Large (problem size) kernels
- Small (code amount) kernels

Example kernels in RRTMGP and RRTMGPxx

Multiple backends require different directives embedded in code

```
!$acc  parallel loop collapse(3) &
!$acc&     copyin(tau2(:ncol,:nlay,:ngpt)) &
!$acc&     copy(tau1(:ncol,:nlay,:ngpt))
!$omp target teams distribute parallel do simd collapse(3) &
!$omp& map(to:tau2) &
!$omp& map(tofrom:tau1)
do igpt = 1, ngpt
  do ilay = 1, nlay
    do icol = 1, ncol
      tau1(icol,ilay,igpt) = tau1(icol,ilay,igpt) + tau2(icol,ilay,igpt)
    end do
  end do
end do
```

- Tightly nested loops
- Large (problem size) kernels
- Small (code amount) kernels

Example kernels in RRTMGP and RRTMGPxx

Multiple backends require different directives embedded in code

```
!$acc  parallel loop collapse(3) &
!$acc&     copyin(tau2(:ncol,:nlay,:ngpt)) &
!$acc&     copy(tau1(:ncol,:nlay,:ngpt))
!$omp target teams distribute parallel do simd collapse(3) &
!$omp& map(to:tau2) &
!$omp& map(tofrom:tau1)
do igpt = 1, ngpt
  do ilay = 1, nlay
    do icol = 1, ncol
      tau1(icol,ilay,igpt) = tau1(icol,ilay,igpt) + tau2(icol,ilay,igpt)
    end do
  end do
end do
```

- Tightly nested loops
- Large (problem size) kernels
- Small (code amount) kernels

Loops reformatted to use yakl parallel_for, but same structure and order; no backend-specific code

```
parallel_for( YAKL_AUTO_LABEL() , SimpleBounds<3>(ngpt,nlay,ncol) , YAKL_LAMBDA (int igpt, int ilay, int icol) {
  tau1(icol,ilay,igpt) = tau1(icol,ilay,igpt) + tau2(icol,ilay,igpt);
});
```