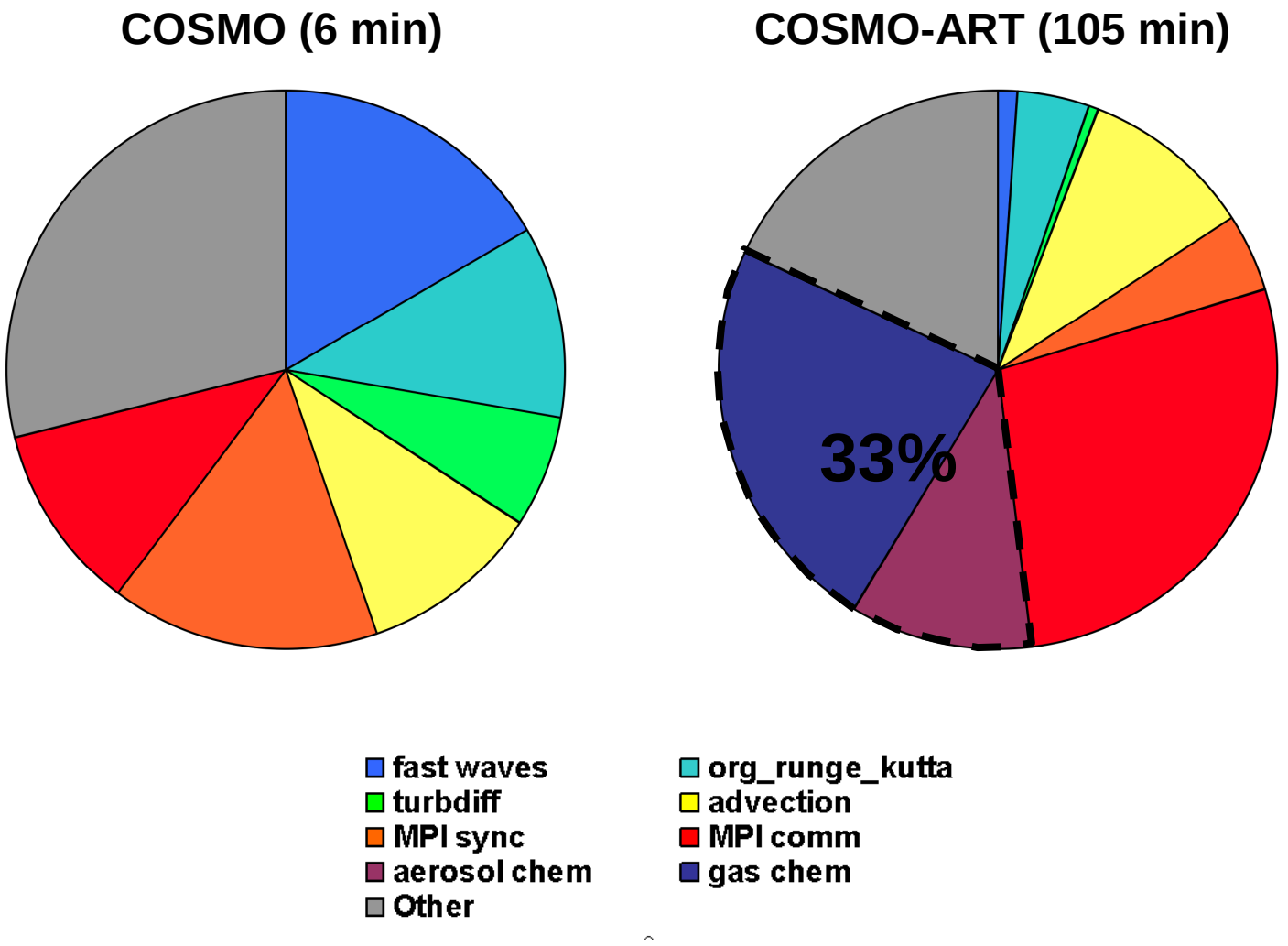
COSMO-ART Performance Profiling



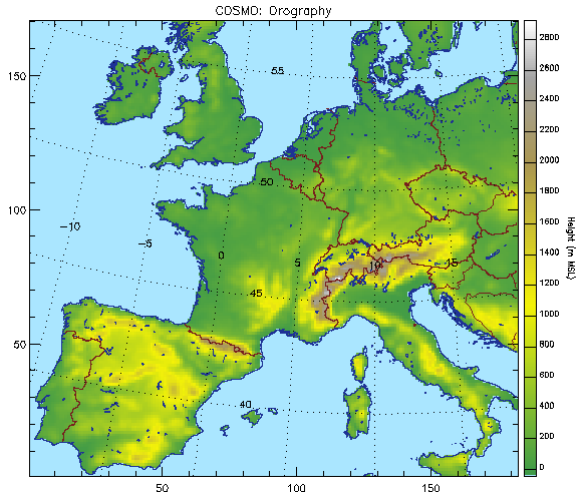
Goal

- Communicate and document profiling results of COSMO-ART that have already been collected
- Understand where COSMO-ART spends its time
- Identify “quick wins”
- Estimate potential for acceleration by KPP or other major modifications to COSMO-ART

Technical Details

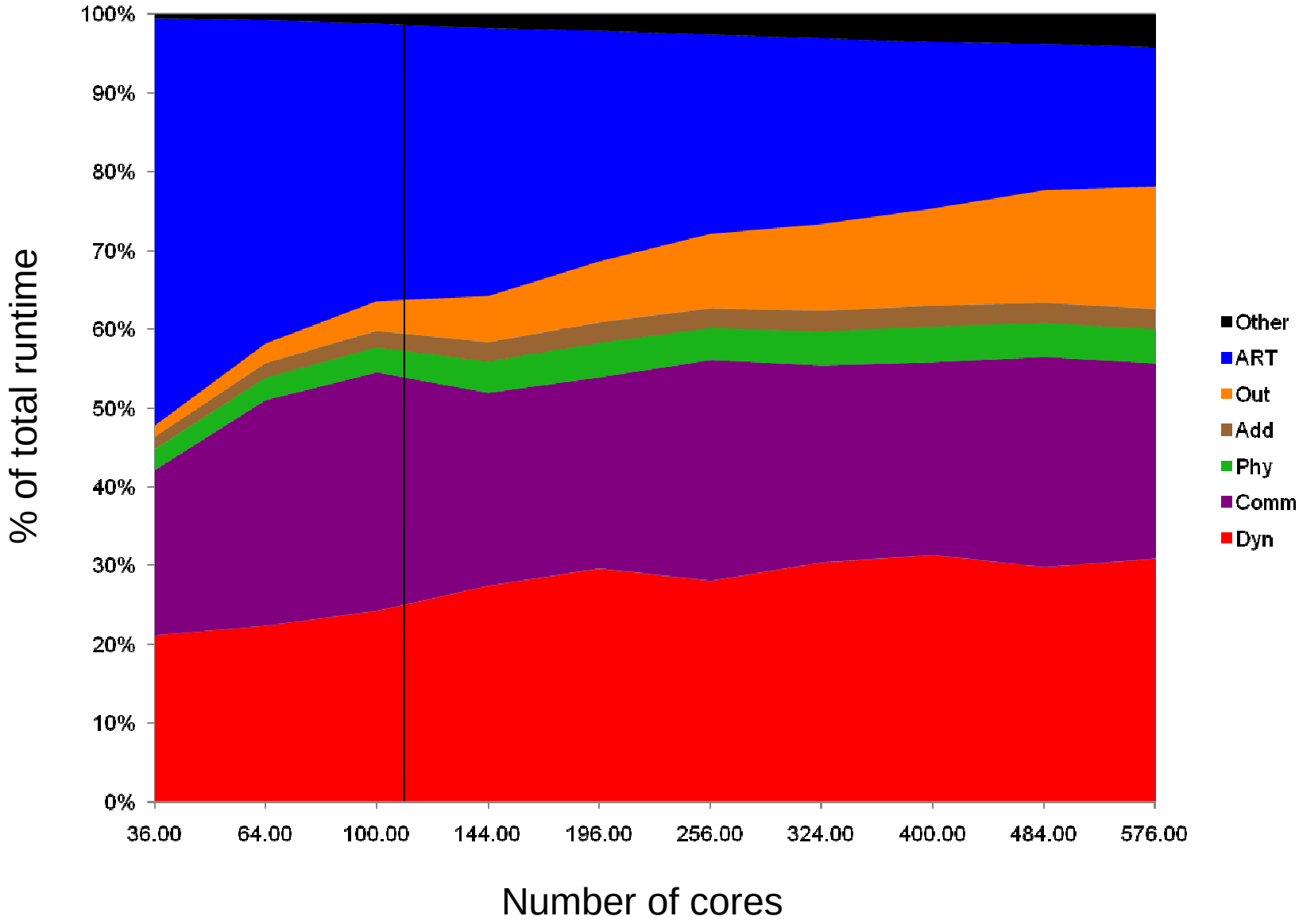
- Machine: dole.cscs.ch
  - Cray XT-4
  - Quad-core AMD Opteron “Barcelona” @ 2.3 GHz
  - 64K I1, 64K D1, 512K L2, shared 2MB L3
  - Single socket nodes, 8 GB DDR RAM
- Compiler: PGI 9.0-4
  - -tp barcelona
  - -Kieee
  - -O3 -fast -Mipa=fast,inline
- Profiler: CrayPat 5.0

Benchmark Simulation

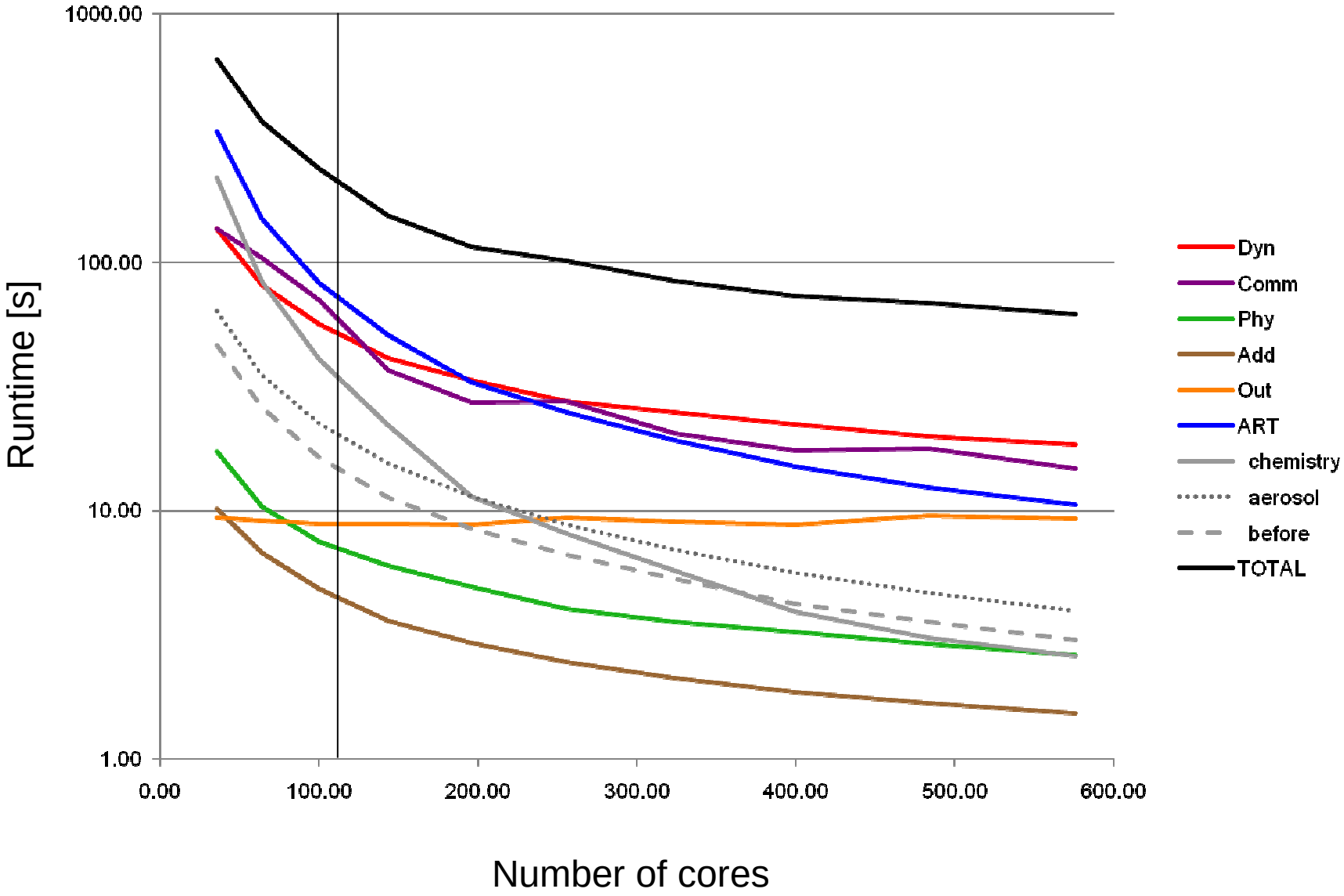


- Delivered by EMPA
- Domain: 182 x 170 x 40
- Job: 10 x 11 cores (1:1 comp/halo)
- Time: 0h to 24h by Δt=60s
- Date: June 10, 2006
- Runge-Kutta dynmical core (preliminary ART implementation)
- Semi-Lagrangian tracer advection
- 58 (56) gas species, 105 (77) aerosol species
- usually only 5 dynamical and 6 microphysical variables (factor 16 increase!)

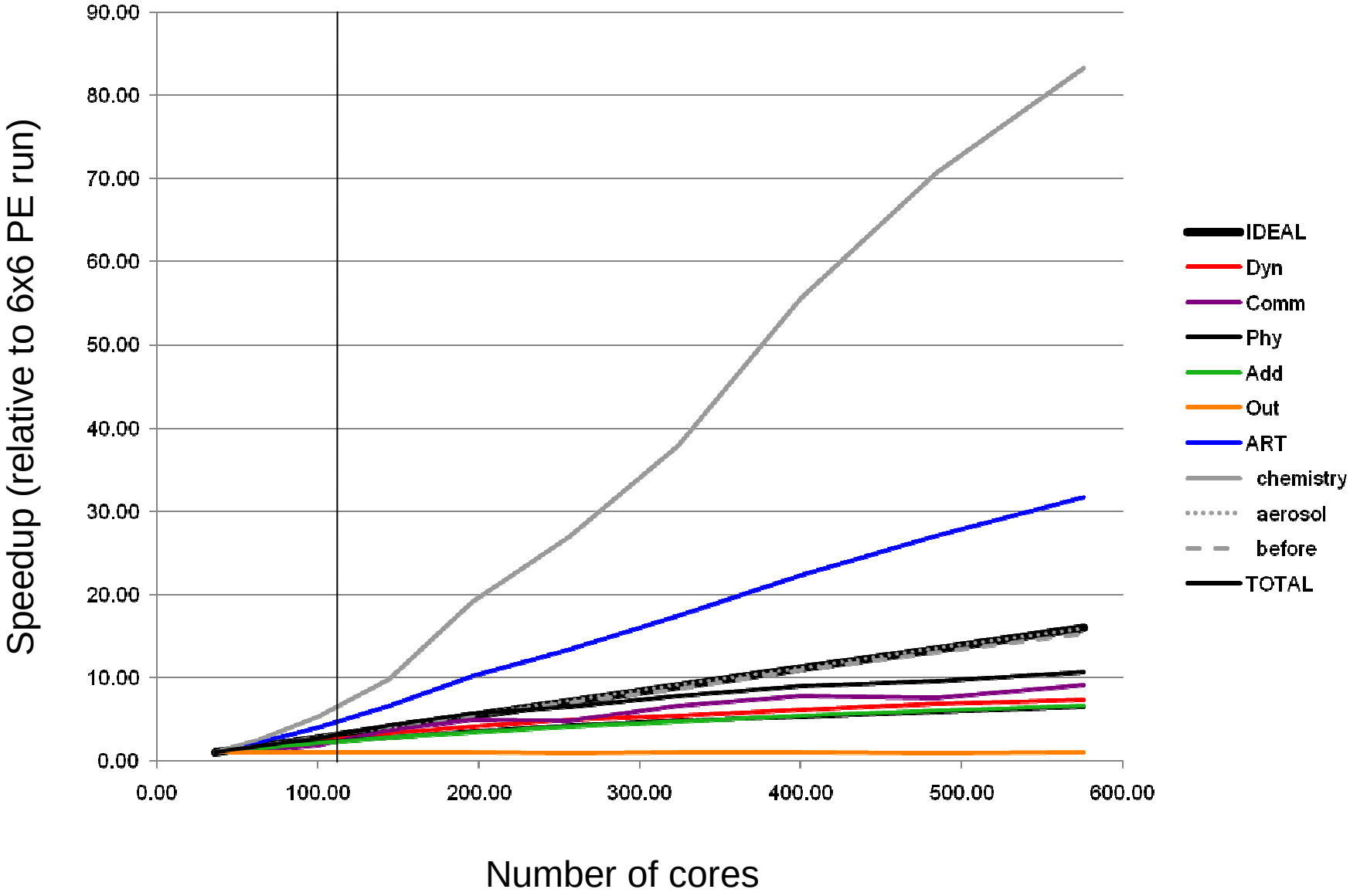
Percent of total runtime



Runtime


1000.00
100.00
10.00
1.00
Runtime [s]
0.00
100.00
200.00
300.00
400.00
500.00
600.00
Number of cores
Dyn
Comm
Phy
Add
Out
ART
chemistry
aerosol
before
TOTAL

Speedup


Speedup (relative to 6x6 PE run)
90.00
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.00
0.00
100.00
200.00
300.00
400.00
500.00
600.00
Number of cores
IDEAL
Dyn
Comm
Phy
Add
Out
ART
chemistry
aerosol
before
TOTAL

Top dogs

24.0%	mpi_recv	halo-update
23.9%	cradm2	gas chemistry
9.5%	rpmmod3	aerosols
7.6%	lmorg	new = now, tens = 0, clipping, *other*
7.2%	interpol_sl_tricubic	tricubic interpolation (advection)
3.8%	mpi_gather (sync)	output (synchronization)
3.6%	mpi_wait	halo-update (synchronization)
3.2%	complete_tendencies_cosmo_art	deposition, vertical diffusion
3.2%	gm3ppm_a / ppmgm3_a	unit conversion
1.8%	org_runge_kutta	dynamics (compute tendencies)
1.4%	sardass	upper and lateral BC
1.2%	fast_waves_runge_kutta	fast waves solver (dynamics)
1.2%	mpi_allreduce (sync)	probably SL clipping

Top dogs

24.0%	mpi_recv	halo-update
23.9%	cradm2	gas chemistry
9.5%	rpmmod3	aerosols
7.6%	*(unknown)*	*(unknown)*
7.2%	interpol_sl_tricubic	tricubic interpolation (advection)
3.8%	**mpi_gather (sync)**	**output (synchronization)**
3.6%	mpi_wait	halo-update (synchronization)
3.2%	complete_tendencies_cosmo_art	deposition, vertical diffusion
3.2%	gm3ppm_a / ppmgm3_a	unit conversion
1.8%	org_runge_kutta	dynamics (compute tendencies)
1.4%	sardass	upper and lateral BC
1.2%	fast_waves_runge_kutta	fast waves solver (dynamics)
1.2%	**mpi_allreduce (sync)**	**probably SL clipping**

Problematic for weak scaling

Top dogs

24.0%	mpi_recv	halo-update
23.9%	**cradm2**	**gas chemistry**
9.5%	**rpmmod3**	**aerosols**
7.6%	*(unknown)*	*(unknown)*
7.2%	interpol_sl_tricubic	tricubic interpolation (advection)
3.8%	mpi_gather (sync)	output (synchronization)
3.6%	mpi_wait	halo-update (synchronization)
3.2%	complete_tendencies_cosmo_art	deposition, vertical diffusion
3.2%	**gm3ppm_a / ppmgm3_a**	**unit conversion**
1.8%	org_runge_kutta	dynamics (compute tendencies)
1.4%	sardass	upper and lateral BC
1.2%	fast_waves_runge_kutta	fast waves solver (dynamics)
1.2%	mpi_allreduce (sync)	probably SL clipping

COSMO-ART specific (normally not present in COSMO)

Halo Exchange (mpi_recv) 24%

Summary

- Exchange of boundaries of neighbouring PEs
- Immediate send (mpi_isend), blocking receive (mpi_recv) and wait (mpi_wait)
- Every timestep 24 MB is sent by each PE in 1307 messages of 19 KB each (buffered)

Note

- COSMO-ART variables (cgas, caero) are exchanged twice per timestep. Really necessary?
- Could lump together messages

```
DO isp=1,isp_aero
  CALL exchg_boundaries( caero(:,:,:,isp,nnow), caero(:,:,:,isp,nnow) )
END DO
```

Gas Chemistry

- Based on RADM2 from Stockwell et al. 1990
- Solves sparse, coupled, stiff set of chemical reaction equations (in one gridcell)
- Uses a very simple semi-implicit approach

$$\frac{dy_i}{dt} = P(\vec{y}) - L(\vec{y})\vec{y}_i \quad \rightarrow \quad \frac{y_i^{n+1} - y_i^n}{\Delta t} = P(\vec{y}^n) - L(\vec{y}^n) \cdot \frac{y_i^{n+1} + y_i^n}{2}$$

$$y_i^{n+1} = \frac{\Delta t P^n + y_i^n \left(1 - \frac{\Delta t}{2} \frac{L^n}{y_i^n}\right)}{1 + \frac{\Delta t}{2} \frac{L^n}{y_i^n}}$$

- Local timestep controlled (max. rel. change in species = 2%)
- More efficient algorithms are well known (Gear, Rosenbrock, …)

Gas Chemistry

Chemical solver steps

1. compute reaction constants k=k(p,T,hν,RH)
2. compute reaction rates for each reaction (172)
3. compute production *P* and loss *L* terms for each species (58)
4. solve differential equation $\frac{dy_i}{dt} = P(\vec{y}) - L(\vec{y})\vec{y}_i$

Gas Chemistry

Chemical solver implementation (pseudo-code)

```
do k=1,40
    copy in vc(ij,l) = cgas(i,j,k,l) for model level k
    compute reaction constants k = k(p,T,hν,RH)
    do istep = 1, nstep
        call prate(k)
        call produ(k)
        call setdt(k)
        call integ1(k)
        time = time + dt
        if (time >= timemax) exit
    end do
    copy out cgas(i,j,k,l) = vc(ij,l) for model level k
end do
```

Gas Chemistry

Important Points

- Code vectorizes over a model level k
- 4 arrays account for 75% of working set size of 1.4 MB

  prod(300,56), loss(300,56), rk(300,172), crk(300,172)
- Computationally intense, but tightest coupled index not first!

```
DO j = 1,imax*jmax

   prod(j,lmacr) = .79*crk(j,151) + .4*crk(j,152) +  &
                   .6*crk(j,154) + .6 *crk(j,155) +  &
                   .6*crk(j,156) + .65*crk(j,87) + crk(j,157)+  &
                   2.*crk(j,159) +  crk(j,160)
   loss(j,lmacr) = crk(j,162) + crk(j,163)

END DO
```

- One grid cell would only be 4.7 KB and fit into D1 cache!
- Timestep is enforced as minimum for all j
  → inefficient, non-reproducible

cradm2 (Gas Chemistry) 24%

```
USER / art_radm2_cradm2_
------------------------------------------------------------------------
  Time%                                  23.9%
  Time                                1218.0 secs
  Imb.Time                             893.3 secs
  Imb.Time%                             43.1%
  Calls                47.3 /sec      57640.0 calls

  Instr per cycle                       0.49 inst/cycle
  Instructions per LD & ST      51.6% refs          1.94 inst/ref

  Computational intensity       0.20 ops/cycle      0.78 ops/ref
  Ops per instruction                   0.40 ops/inst
  HW FP Ops / User time         455.407 M/sec       5.0%peak
  FP Multiply / FP Ops                  52.1%
  FP Add / FP Ops                       47.9%
  Vectorization                         61.1%

  TLB utilization          5466.99 refs/miss    10.678 avg uses

  D1 cache hit,miss ratios      94.9% hits          5.1% misses
  D2 cache hit,miss ratio       55.5% hits          44.5% misses
  D1+D2 cache hit,miss ratio    97.7% hits          2.3% misses

  System to D1 bandwidth      1109.554 MB/sec
  D2 to D1 bandwidth          1383.847 MB/sec

========================================================================
```

HUGE IMBALANCE! [= (Max – Mean)/Mean]

one call per timestep and model level

CPU is waiting often! [Max = 3 inst/cycle]

could be higher [Max = 2 ops/cycle]

not very high! [Ideal = 20% - 30%]

not everything vectorizes [Max = 100%]

low D1 cache usage! [Ideal > 99.0%]

low D2 cache usage [Ideal > 95.0%]

high bandwidth! [Max = 3 GB/s]

cradm2 (Un-vectorized)

- Changed index order from (ij,l) to (l) and pulled out loop over gridpoints out of the chemical solver

```
do k=1,40
    do ij = 1, imax*jmax
        copy in vc(l) = cgas(i,j,k,l) for gridpoint i,j,k
        compute reaction constants k = k(p,T,hν,RH)
        do istep = 1, nstep
            call prate
            call produ
            call setdt        <-- each cell has its own Δt
            call integ1
            time = time + dt
            if (time >= timemax) exit
        end do
        copy out cgas(i,j,k,l) = vc(l) for gridpoint i,j,k
    end do
end do
```

cradm2 (Un-vectorized)

```
USER / art_radm2_cradm2_
------------------------------------------------------------------------
  Time%                                 20.6%      23.9%
  Time                                850.8 secs   218.0
  Imb.Time                            276.1 secs   893.3
  Imb.Time%                             25.0%      43.1%
  Calls                47.6 67.7 /sec 57640.0 calls 57140.0

  Instr per cycle                     0.84 inst/cycle 0.49
  Instructions per LD & ST 51.6% 51.6% refs   1.96 inst/ref 1.64

  Computational intensity  0.20 0.21 ops/cycle  0.49 ops/ref 0.73
  Ops per instruction                 0.40 ops/inst 0.40
  HW FP Ops / User time  455.4 455.407 M/sec     5.3% peak
  FP Multiply / FP Ops                53.2%      52.1%
  FP Add / FP Ops                     46.8%      47.9%
  Vectorization                       0.0%       61.1%

  TLB utilization      5466532 24.68 refs/miss  10.400 avg uses

  D1 cache hit,miss ratios 94.9% 99.4% hits      0.6% misses
  D2 cache hit,miss ratio  55.5% 59.6% hits     40.4% misses
  D1+D2 cache hit,miss ratio 97.7% 98.8% hits    0.2% misses

  System to D1 bandwidth    234.889 MB/sec 1180.554
  D2 to D1 bandwidth        346.484 MB/sec 1180.847

========================================================================
```

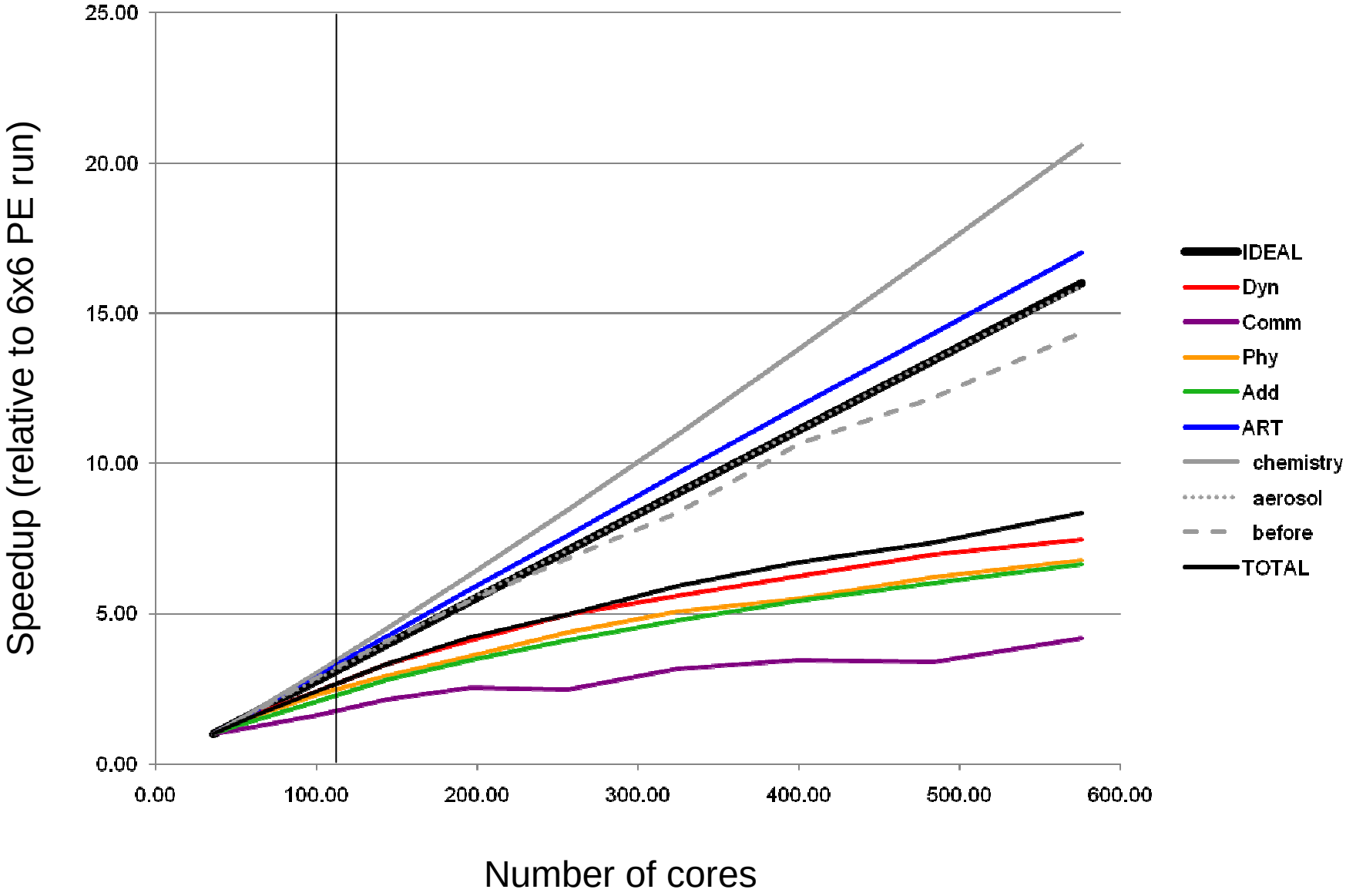
30% faster

25% less ops

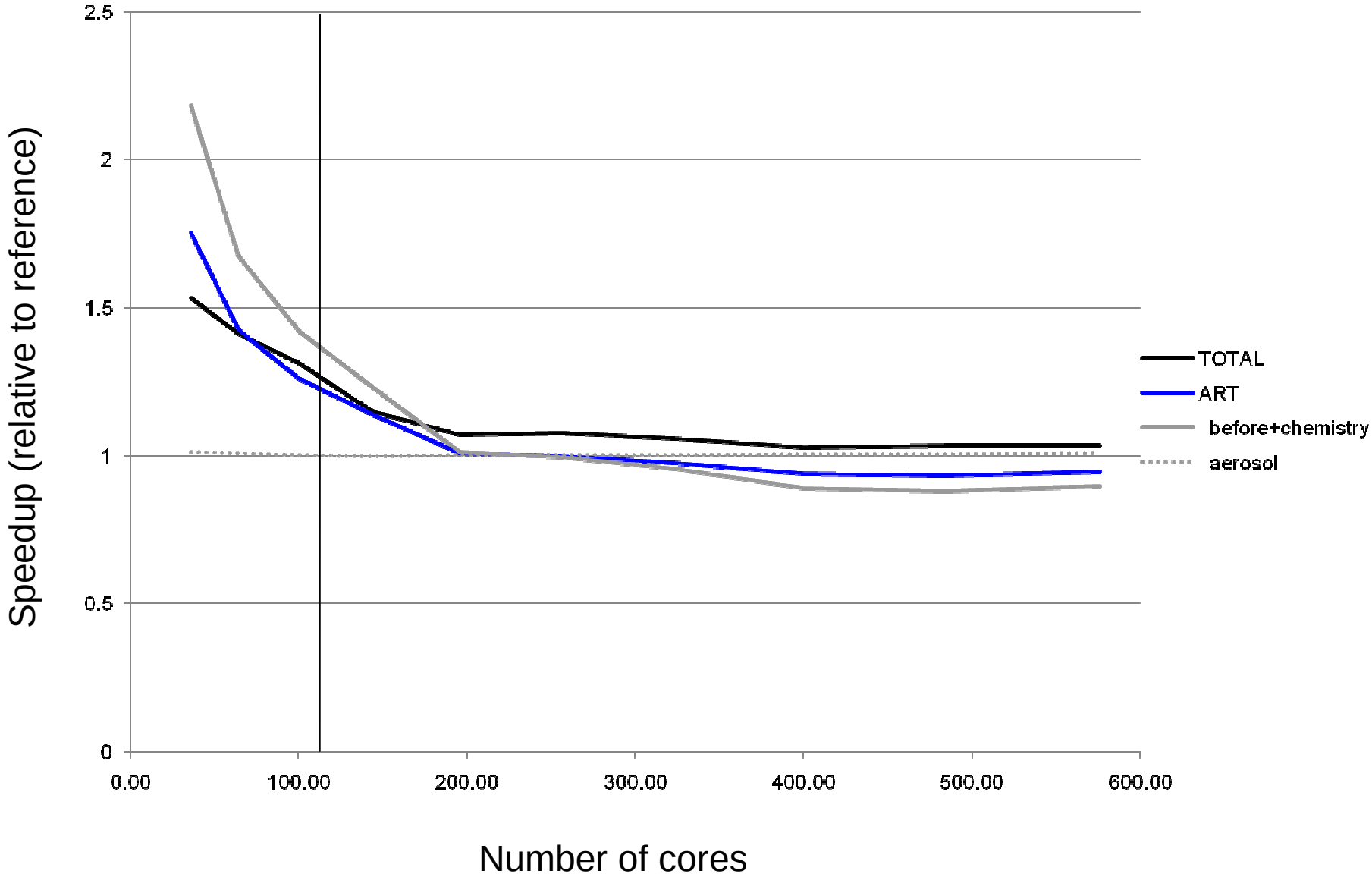
No vectorization!

4-5 times less main memory and L2 cache traffic!

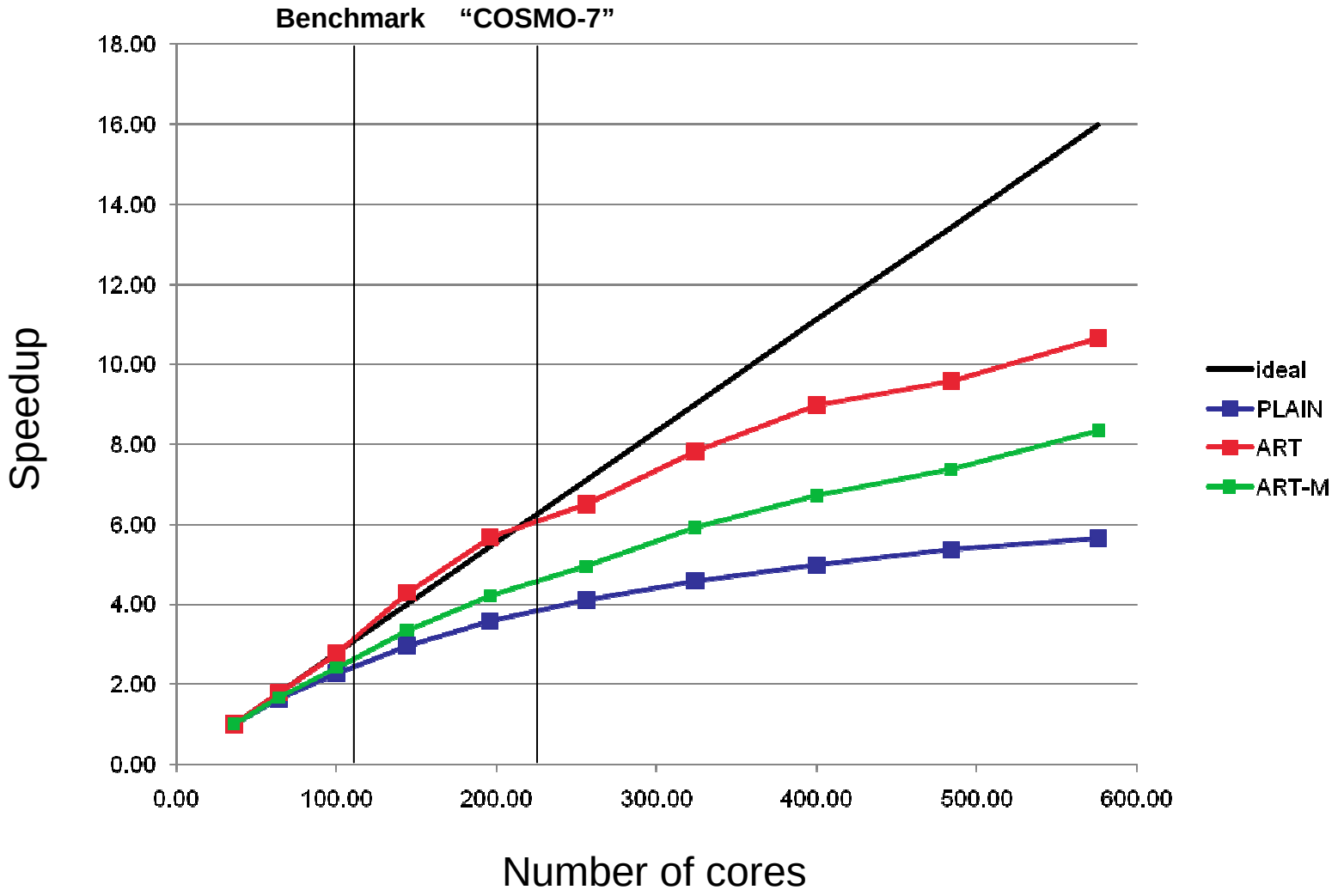
Speedup


Speedup (relative to 6x6 PE run)
25.00
20.00
15.00
10.00
5.00
0.00
0.00
100.00
200.00
300.00
400.00
500.00
600.00
Number of cores
IDEAL
Dyn
Comm
Phy
Add
ART
chemistry
aerosol
before
TOTAL

Improvement of Modifications



COSMO-ART vs. COSMO



cradm2 (Gas Chemistry) 24%

- **Summary**
  - Huge imbalance (43%) between PEs due to different timesteps
  - Some imbalance (25%) inherent to problem
  - Enforcement of minimum chemistry-Δt per PE
    $\rightarrow$ inefficient, non-reproducible
  - Very cache inefficient and high memory bandwidth for small PE counts
  - Low computational density in spite of computationally very dense code
  - Substantial vectorization (61%)

Aerosols

- Not investigated in detail
- Beware profilers: Only starts after 2h into simulation!
- Does not have internal timestepping
  (i.e. only one update per model timestep Δt)

rpmmod3 (Aerosols) 9.5%

```
USER / art_radm2_cradm2_
------------------------------------------------------------------------
  Time%                                  9.4%
  Time                                 481.5 secs
  Imb.Time                             128.2 secs
  Imb.Time%                             21.4%
  Calls                  89.8 /sec     43240.0 calls

  Instr per cycle                       0.75 inst/cycle
  Instructions per LD & ST   40.6% refs          2.46 inst/ref

  Computational intensity    0.11 ops/cycle      0.37 ops/ref
  Ops per instruction                   0.15 ops/inst
  HW FP Ops / User time      256.698 M/sec             2.8%peak
  FP Multiply / FP Ops                  53.3%
  FP Add / FP Ops                       46.7%
  Vectorization                         8.3%

  TLB utilization          3679.02 refs/miss     7.186 avg uses

  D1 cache hit,miss ratios    99.6% hits          0.4% misses
  D2 cache hit,miss ratio     56.7% hits          43.3% misses
  D1+D2 cache hit,miss ratio  98.8% hits          0.2% misses

  System to D1 bandwidth      98.337 MB/sec
  D2 to D1 bandwidth         128.888 MB/sec

========================================================================
```

half of imbalance of gas chemistry

CPU busier, but still waiting

few computations / instruction

Almost no vectoriztation
(has 10% single precision ops?)

better cache efficiency

not system memory bound [Max = 3 GB/s]

gm3ppm_a / ppmgm3_a 3.2%

- Converts units from µg/m^3 to ppm, and vice-versa
- Only for aerosol species (77 tracers)
- Called 3 times (nold, nnow, nnew) before/after transport and relaxation (→ 6 calls/Δt each)
- Working set ie x je x ke x 77 x 8 = 0.71 GB

```
SUBROUTINE gm3ppm_a (cfield,kt)
 INTEGER (KIND=iintegers)  ::              &
   isa, kt
 REAL (KIND=ireals)       ::              &
   cfield(ie,je,ke,isp_aero,3),           &
   alpha (ie,je,ke)

   alpha(:,:,:) = 1. / (rho(:,:,:)*1.e03)
   DO isa=1,isp_aerotrans
    cfield(:,:,:,isa,kt) = cfield(:,:,:,isa,kt) * alpha(:,:,:)
   END DO
END SUBROUTINE gm3ppm_a
```

gm3ppm_a / ppmgm3_a 3.2%

- **Summary**
  - Unit conversion accounts for 3.2% of total runtime!
  - Completely memory bound
  - Bandwidth (1.3 GB/s) close to theoretical maximum (12 GB/s / 4 cores / 2 read/write = 1.5 GB/s)

- **Note** (for RK core)
  - effect is undone in dynamics if positive definite transport schemes are used (lsl_adv_qx = .false.)
  - for dynamics only nnow is used (nnew on exit for RK core)
  - is also done for nold even if RK core is used

Summary (1/3)

- COSMO-ART is **10-20 times more expensive** than plain COSMO
- The performance of COSMO-ART is dominated by…
  - **gas chemistry and aerosol**
  - **halo-update**
  - **tracer dynamics** (ADV, DIFF, BC)
- The code typically…
  - is **optimized for vector machines**
  - uses a lot of **hard-coded** indices due to explicit storage of sparse matrix in form of code
  - does a lot of **copy-in/-out of data** due to “modularity”
  - is **imbalanced**

Summary (2/3)

- At **“typical” core counts** (~100)…
  - cache usage is very bad
  - memory bandwidth is important
  - performance is “bogged down” due to PE-global Δt
- At **higher core counts** (> 200)…
  - the code scales better than plain COSMO
  - cache usage improves substantially
- At **even higher core counts** (> 1000)…
  - Communication and tracer dynamics will dominate
  - Serial NetCDF I/O will become bottleneck
  - MPI collectives in SL-Advection will be felt

Summary (3/3)

- COSMO-ART will **profit** from improvement of efficiency of
  - Communication
  - Tracer dynamics
- COSMO-ART will **not profit** from improvement of the efficiency of
  - Fast wave solver
- **Change in chemistry (e.g. KPP)** might bring improvement (accounts for ~25-35% of total runtime)
- A lot of **data movement** is involved due to the 163 additional prognostic fields!