ITEA 2

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

How to Accelerate an Application: A Practical Case Study in Combustion Modelling

B. Risio, A. Berreth, RECOM Services, Stuttgart, Germany
St. Zuckerman, S. Koliai, M. Ivascot, W. Jialby, B. Krammer, UVSQ, France
B. Mohr, JSC, Jülich, Germany
T. William, GWT-TUD, Dresden, Germany

Parallel Programming for Multi-core Architectures

www.parma-itea2.org

ParCo 2009, Lyon, 1-4 September 2009

Combustion Modelling Software

- **The Combustion Modelling Software RECOM-AIOLOS is a tailored application for the mathematical modelling of industrial firing systems ranging from several hundred kW to more than 1000 MW.**

- **The Software solves approx. 100 conservation equations (mass, momentum, energy, species concentrations, radiation) on a 10-15 million cells finite volume grid, leading to high computational demands.**

- **The Software, originally designed for High-Performance Computing on Parallel Vector-Computers and Massively Parallel Systems, has been ported to low-cost Multi-Core systems to expand the hardware base.**

- **A Suite of Optimization Tools (MARMOT, VAMPIR, SCALASCA, MAQAO) was used for Performance Tuning to identify performance bottlenecks and potential improvements in a systematic way.**

Applied Workflow

- **High Level Performance Analysis of existing MPI-Parallelization**
  - Checking of MPI-Implementation with the Tool MARMOT
  - Analysing MPI-Parallel Execution with the Tools VAMPIR & SCALASCA

    ➡ Optimized massively parallel execution across nodes

- **Low Level Performance Analysis for single and multi core**
  - Checking single and multi core execution on subroutine level using performance counters (VAMPIR / MAQAO)
  - Checking performance bottlenecks on loop level within critical subroutines (MAQAO)

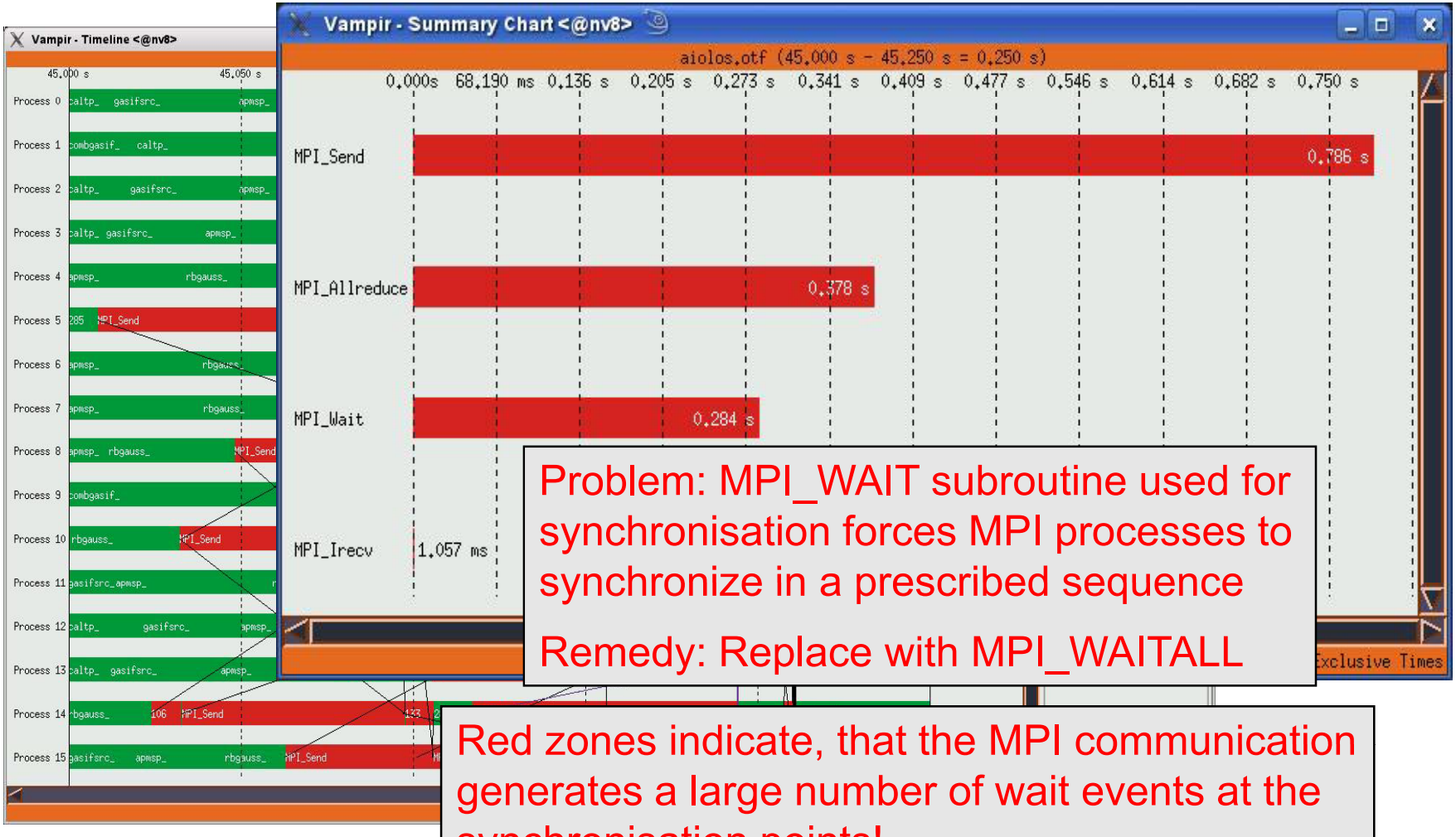
    ➡ Optimized execution on individual cores

- **High Level Performance Analysis of Hybrid Parallelism**
  - Investigate benefits of using hybrid (OpenMP and MPI) parallel execution within the node

    ➡ Optimized shared memory parallel execution within multi-core node

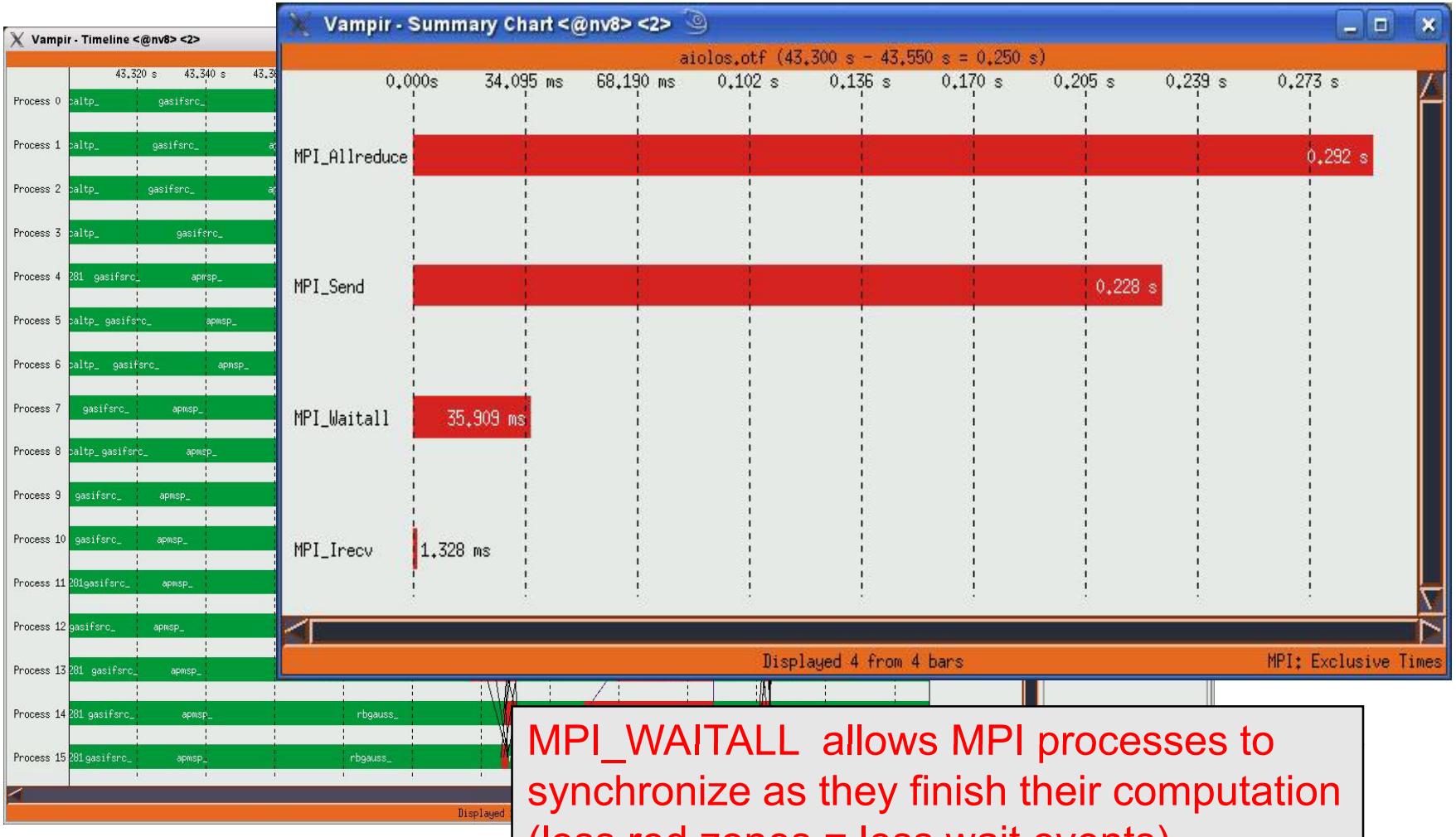
Example: VAMPIR Analysis of Original MPI-Communication

Portion of the code covering 0.25 s



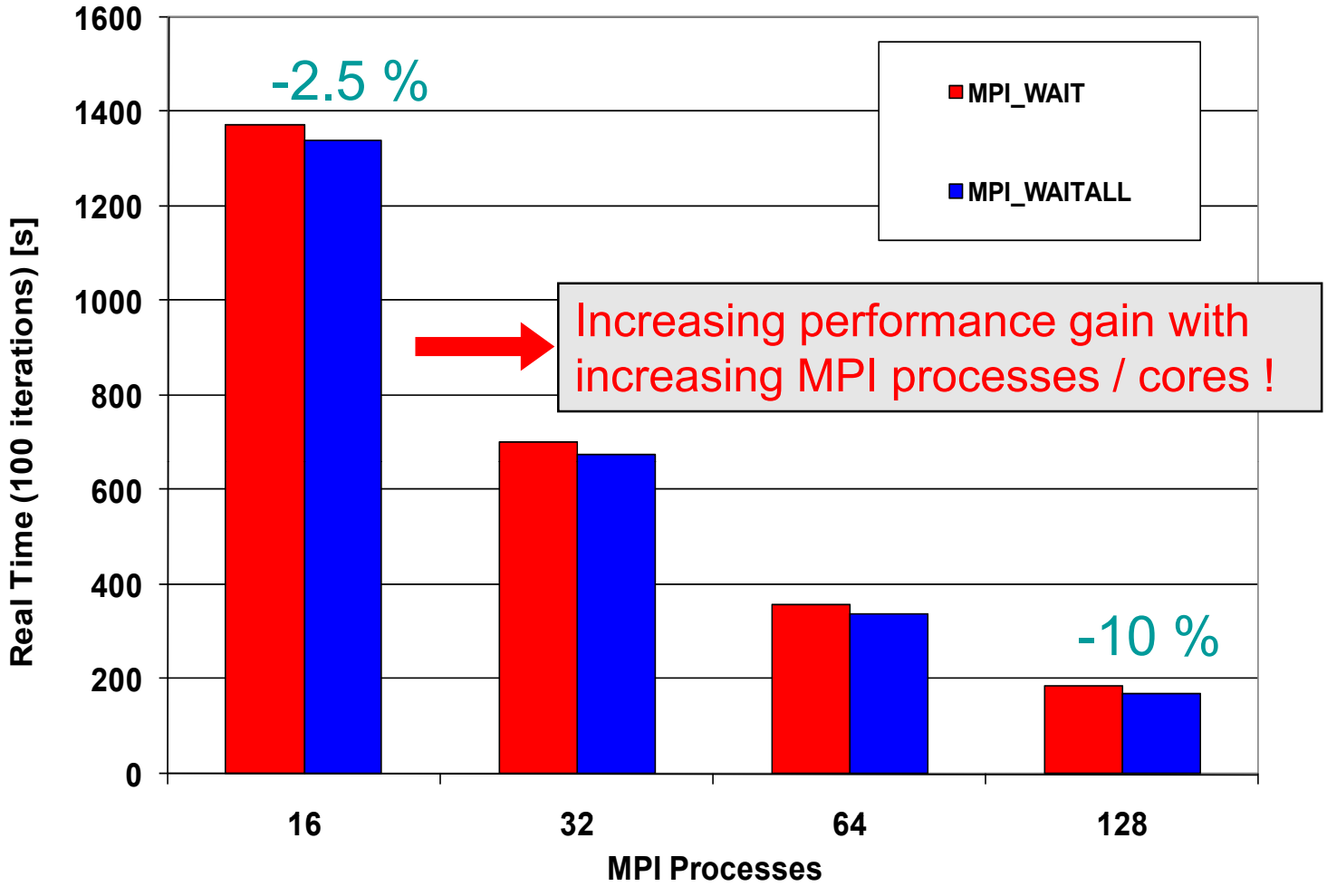
Example: VAMPIR Analysis of Modified MPI-Communication

Portion of the code covering the same 0.25 s

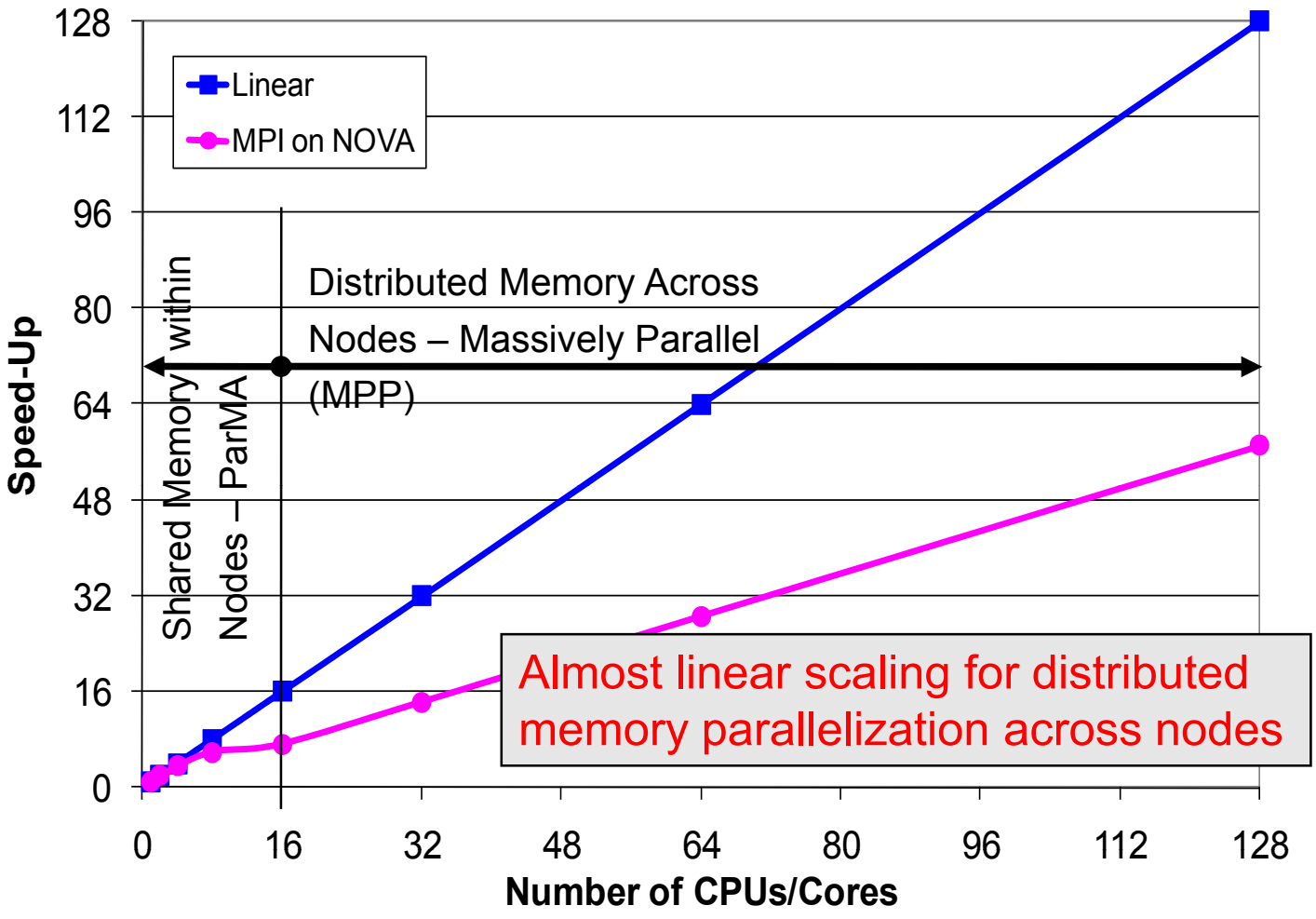


MPI_WAITALL allows MPI processes to synchronize as they finish their computation (less red zones = less wait events).

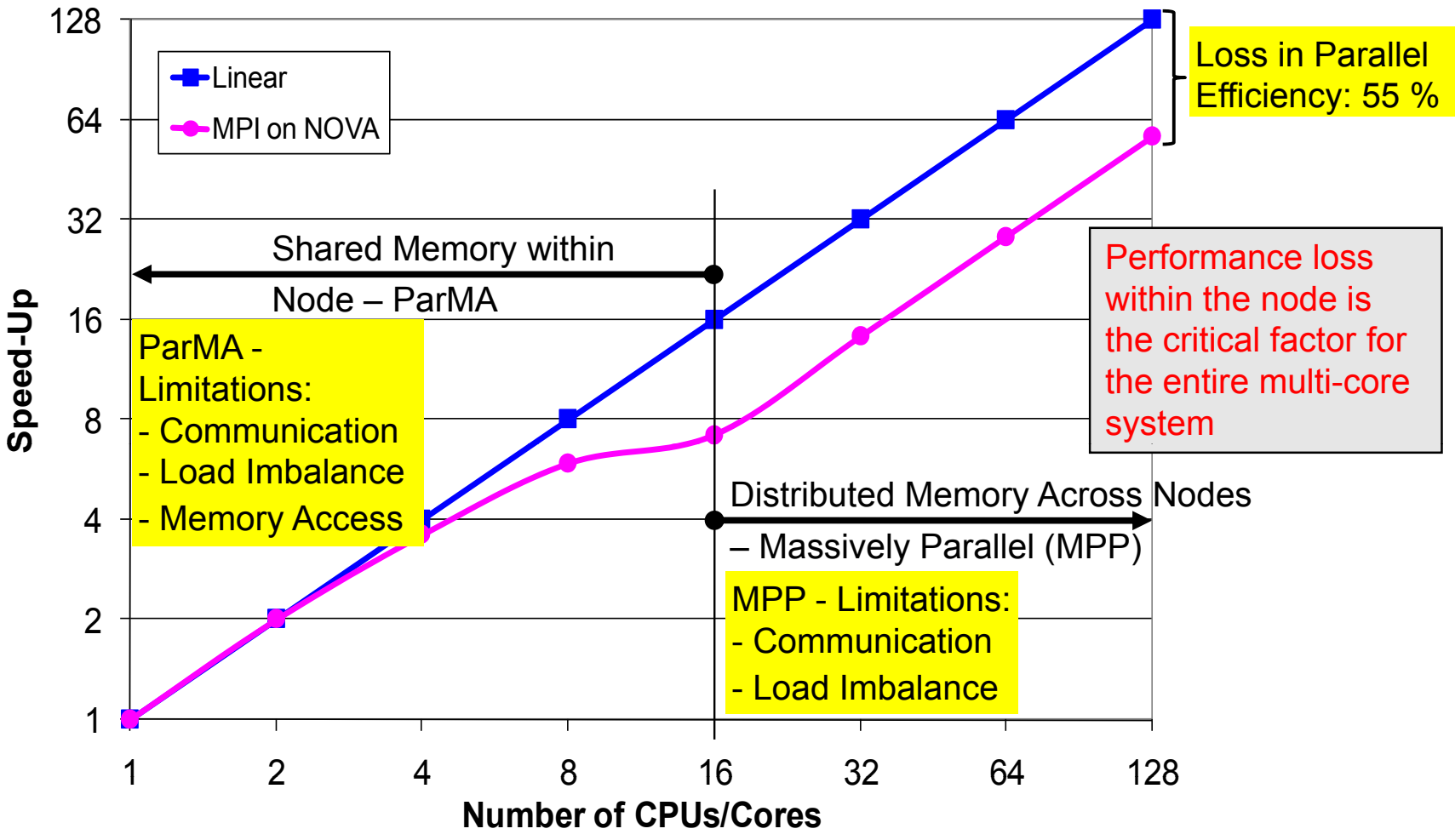
Performance improvement by exchange of MPI_WAIT with MPI_WAITALL



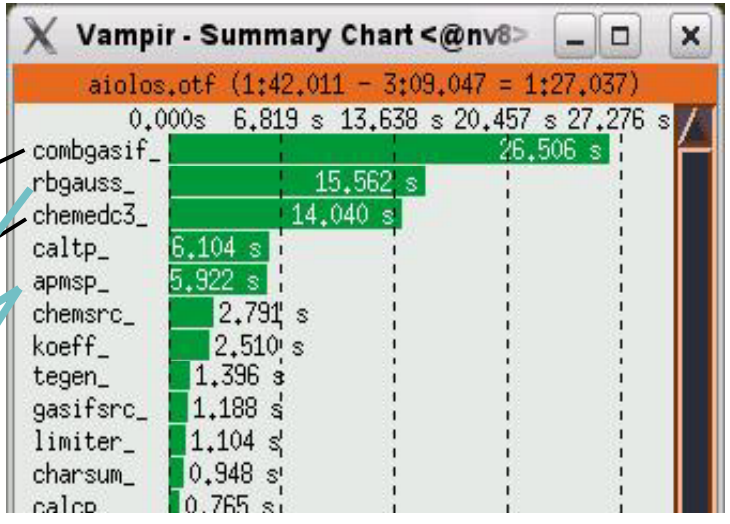
MPI Performance Analysis on Xeon-Cluster NOVA



A closer look to the Parallel Performance within the nodes

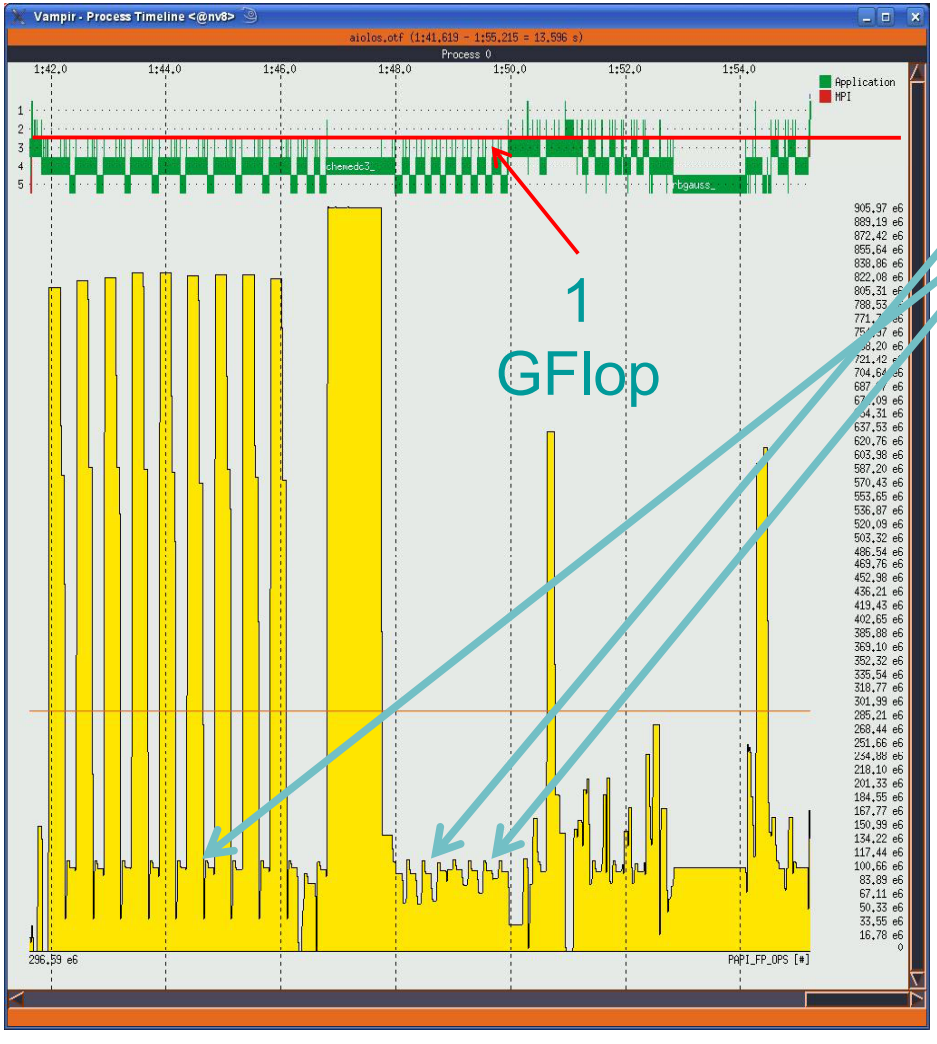


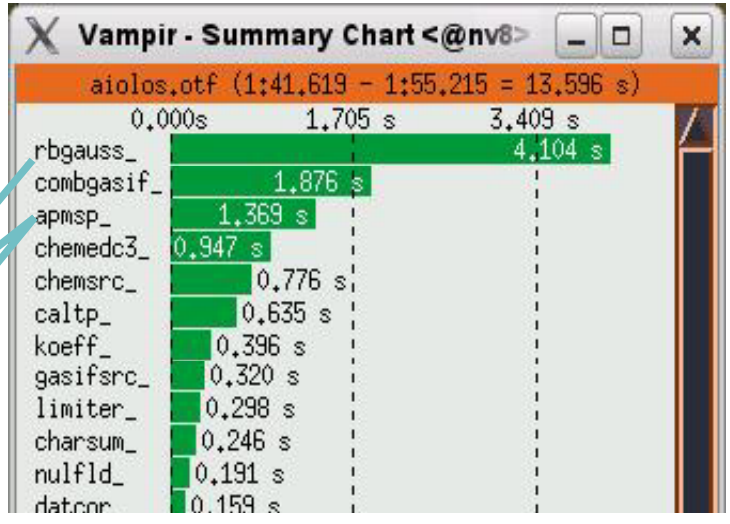
Analysis of subroutine Single Core performance using Flop counters



- combgasif approx. 930 Mflops
- chemedc3 approx. 970 Mflops
- rbgauss approx. 400 Mflops
- apmsp approx. 410 Mflops
  (about 6 % of total computing time)

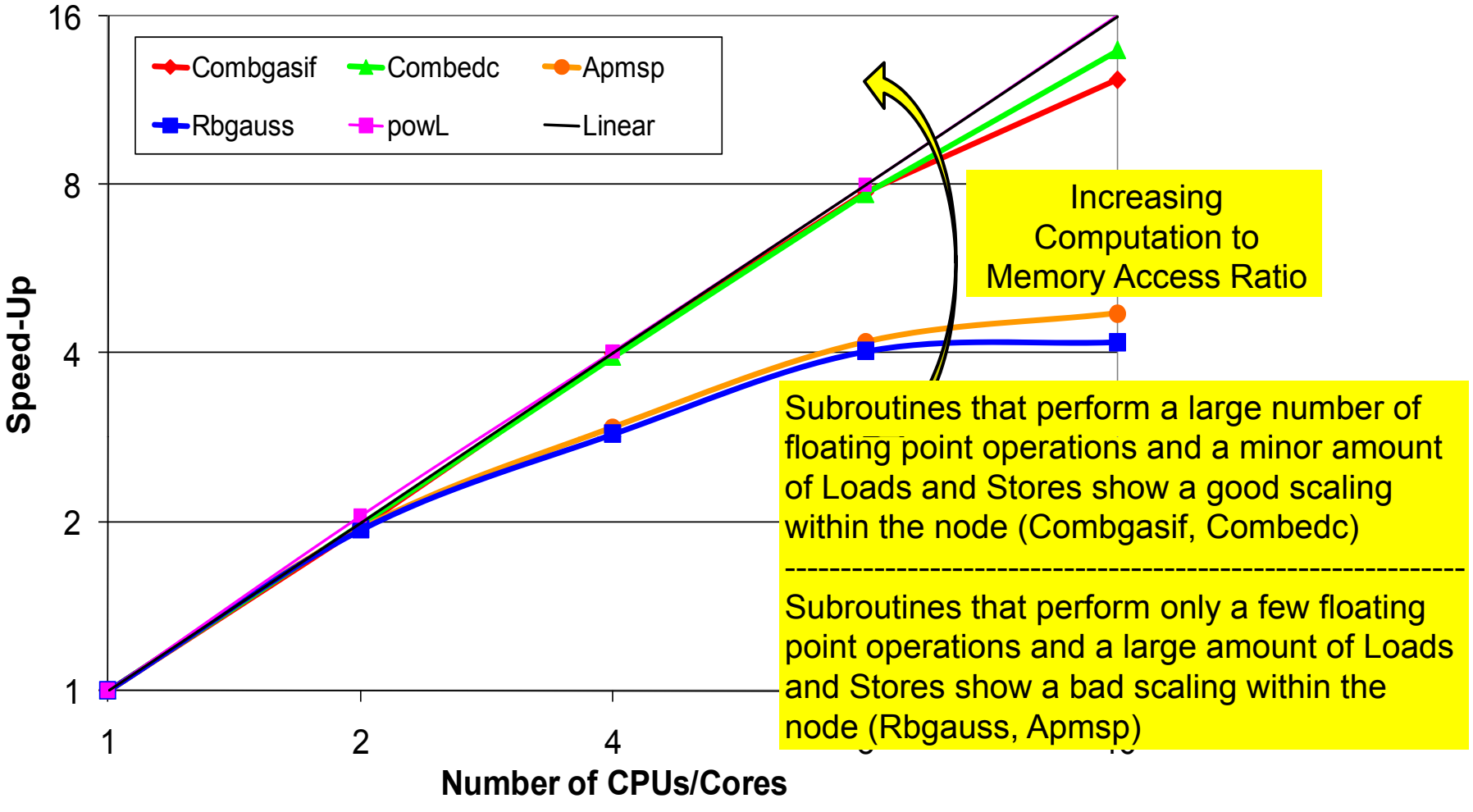
Analysis of subroutine Single Node (16 Cores) performance using Flop counters



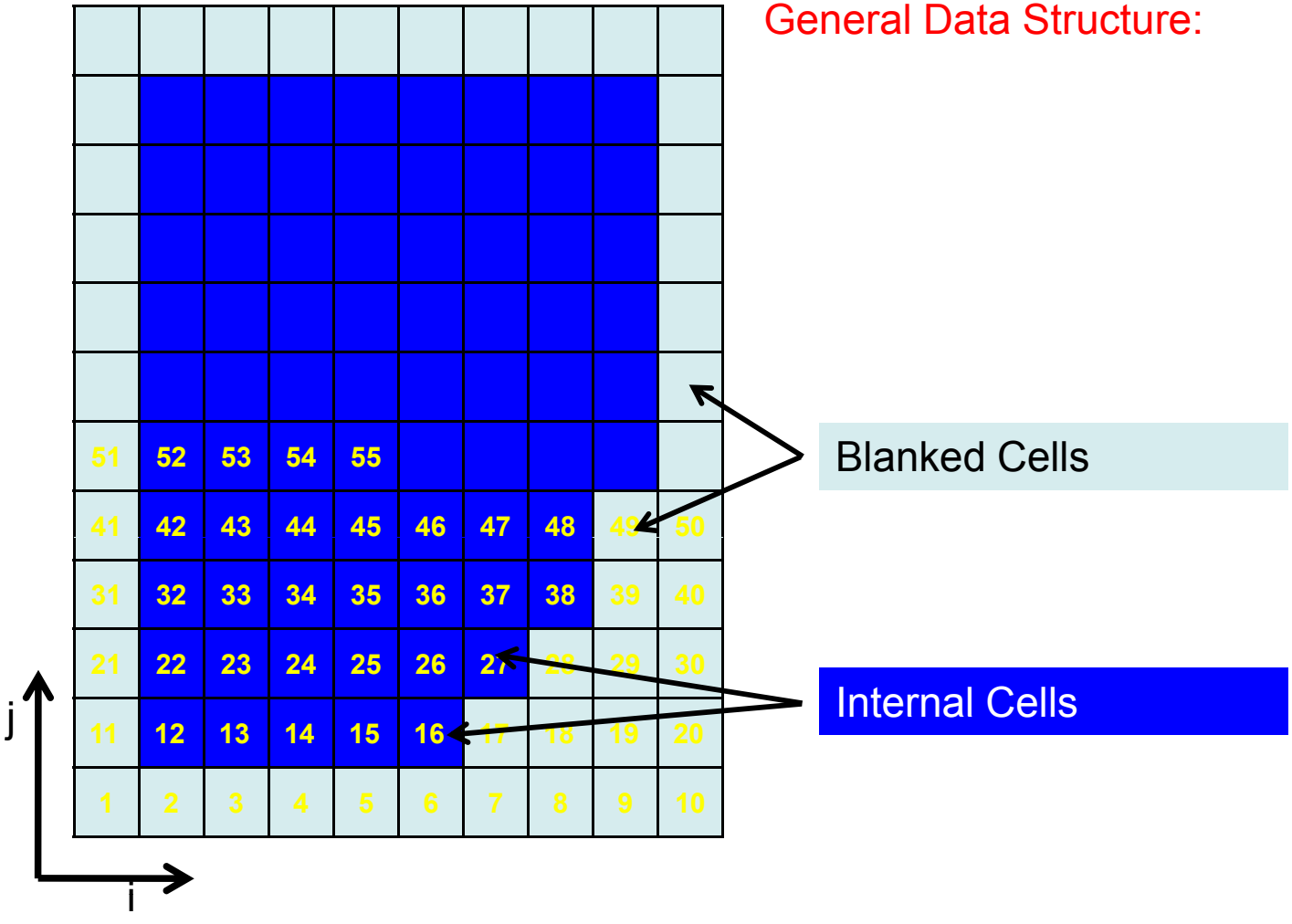


- combgasif approx. 820 Mflops
- chemedc3 approx. 910 Mflops
- rbgauss approx. 100 Mflops
- apmsp approx. 110 Mflops
  (about 10 % of total computing time)

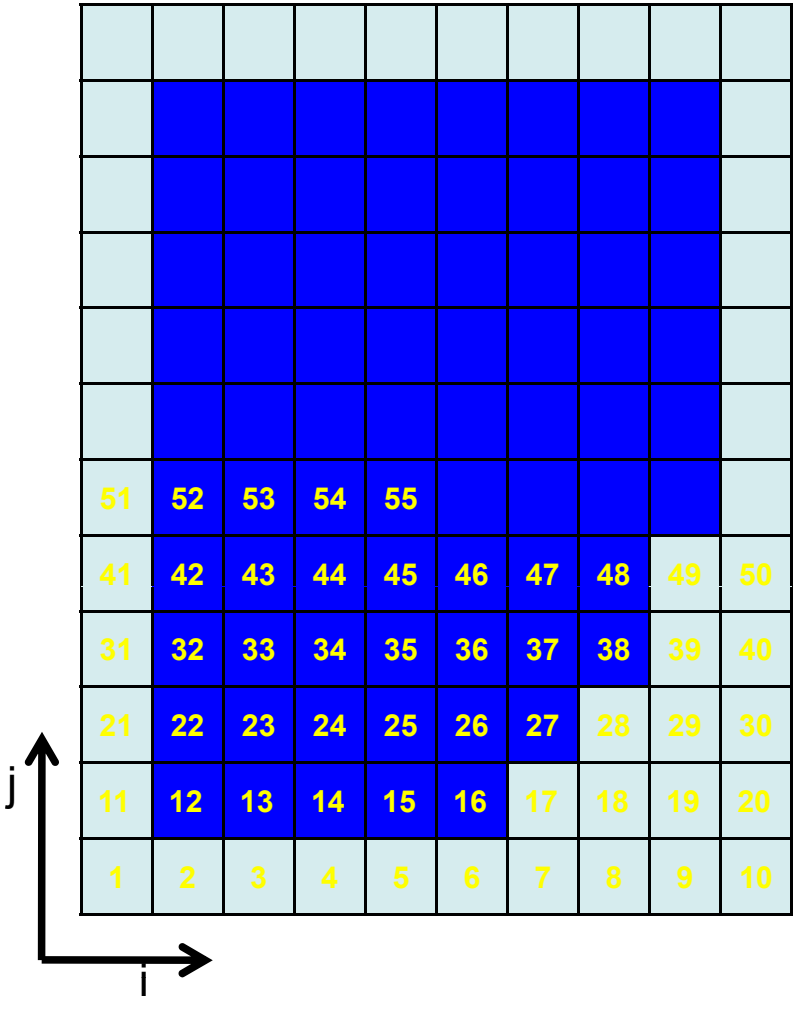
Parallel Single Node (16 Cores) performance on subroutine level



Data Structure of Rbgauss Subroutine



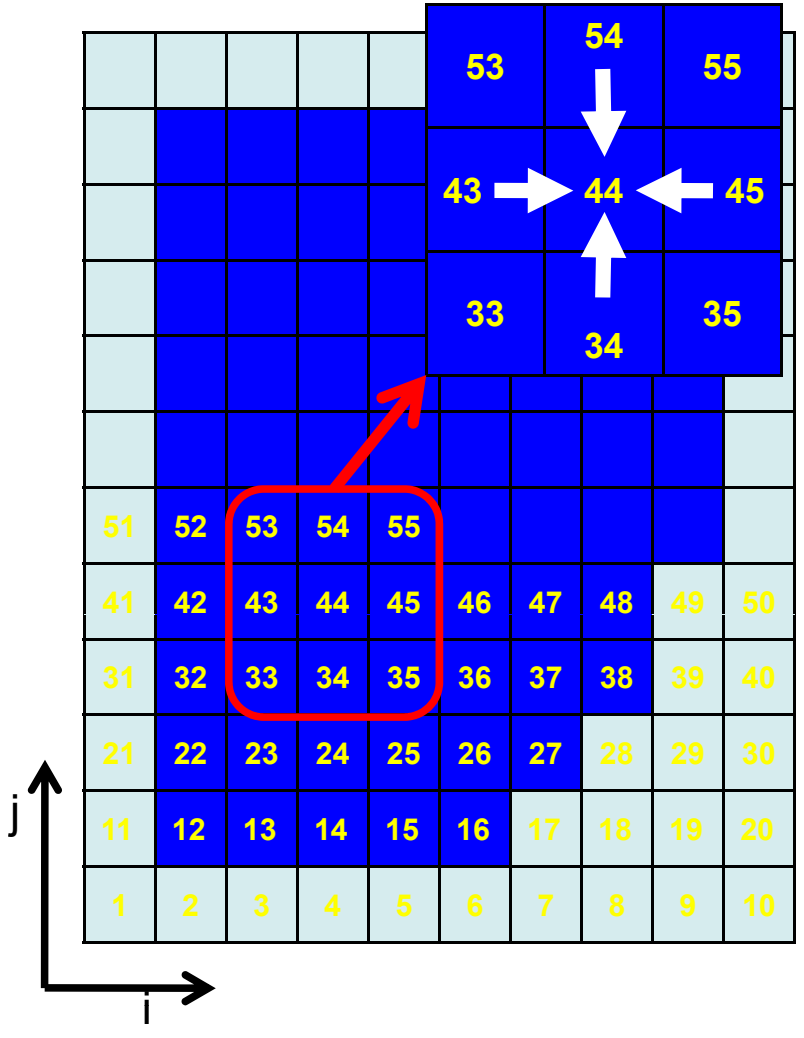
Principle of Gauss-Seidl Solver



Gauss-Seidl Solver for Transport Equations:

$$A_P \phi_P = \sum_{j=1}^{6} (A_j \phi_j) + S_\phi V$$

```
DO IDO=1,NINTERND
 INC = INDIN(IDO)
 ANS = AN(INC,1)*PHI(INC+1)      &
     + AN(INC,2)*PHI(INC-1)      &
     + AN(INC,3)*PHI(INC+INPD)   &
     + AN(INC,4)*PHI(INC-INPD)
     + SU(INC)
 PHI(INC) = ANS/AP(INC)
ENDDO
```

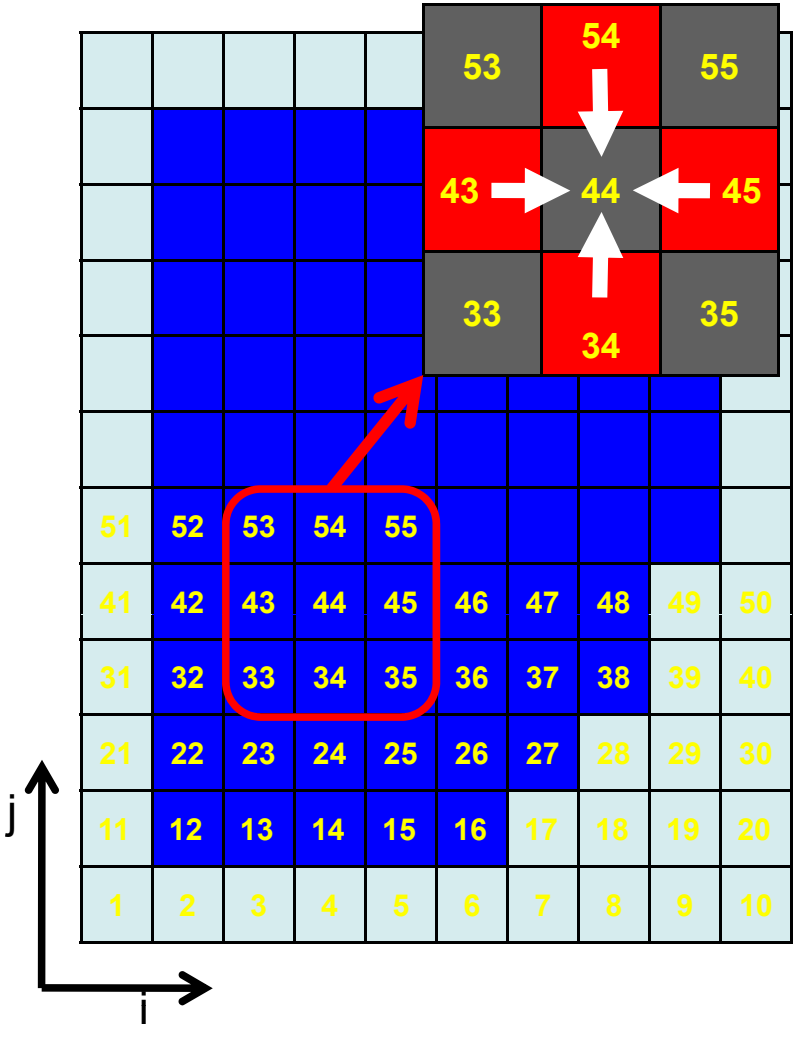


Gauss-Seidl Solver for Transport Equations:

$$A_P \phi_P = \sum_{j=1}^{6} (A_j \phi_j) + S_\phi V$$

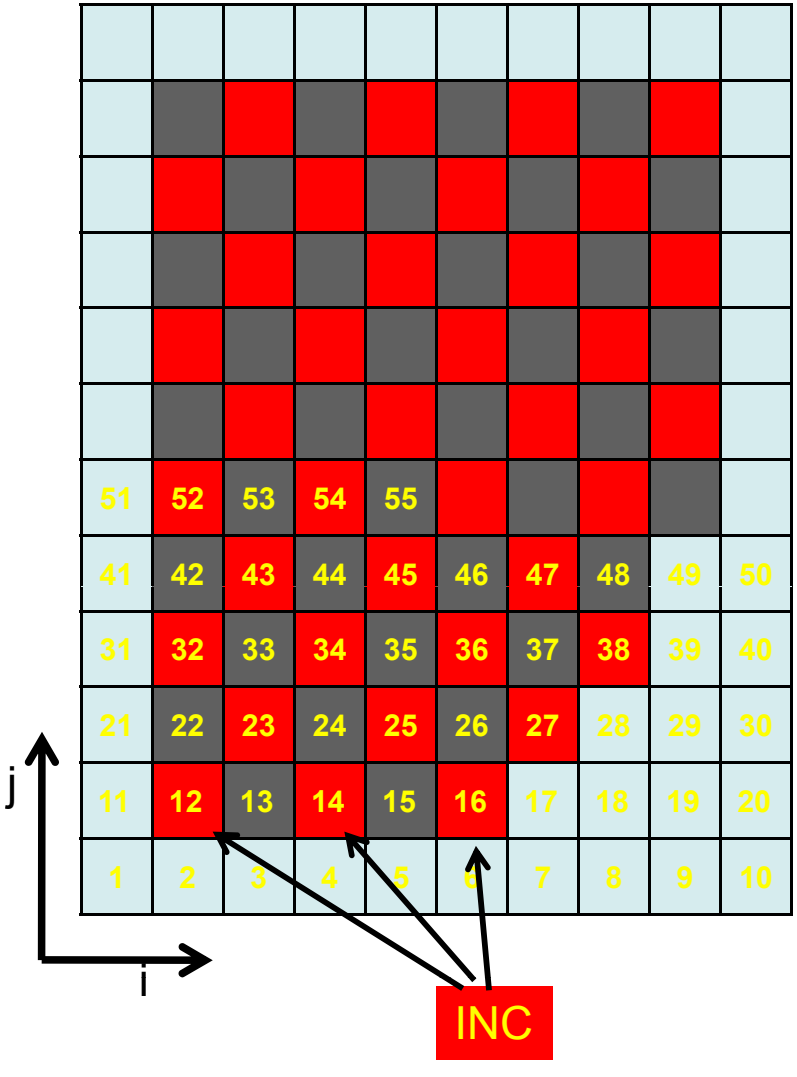
```
DO IDO=1,NINTERND
 INC = INDIN(IDO)
 ANS = AN(INC,1)*PHI(INC+1)      &
     + AN(INC,2)*PHI(INC-1)      &
     + AN(INC,3)*PHI(INC+INPD)   &
     + AN(INC,4)*PHI(INC-INPD)
     + SU(INC)
 PHI(INC) = ANS/AP(INC)
ENDDO
```

Structure of Rbgauss Subroutine



Gauss-Seidl Solver with Red-Black Ordering

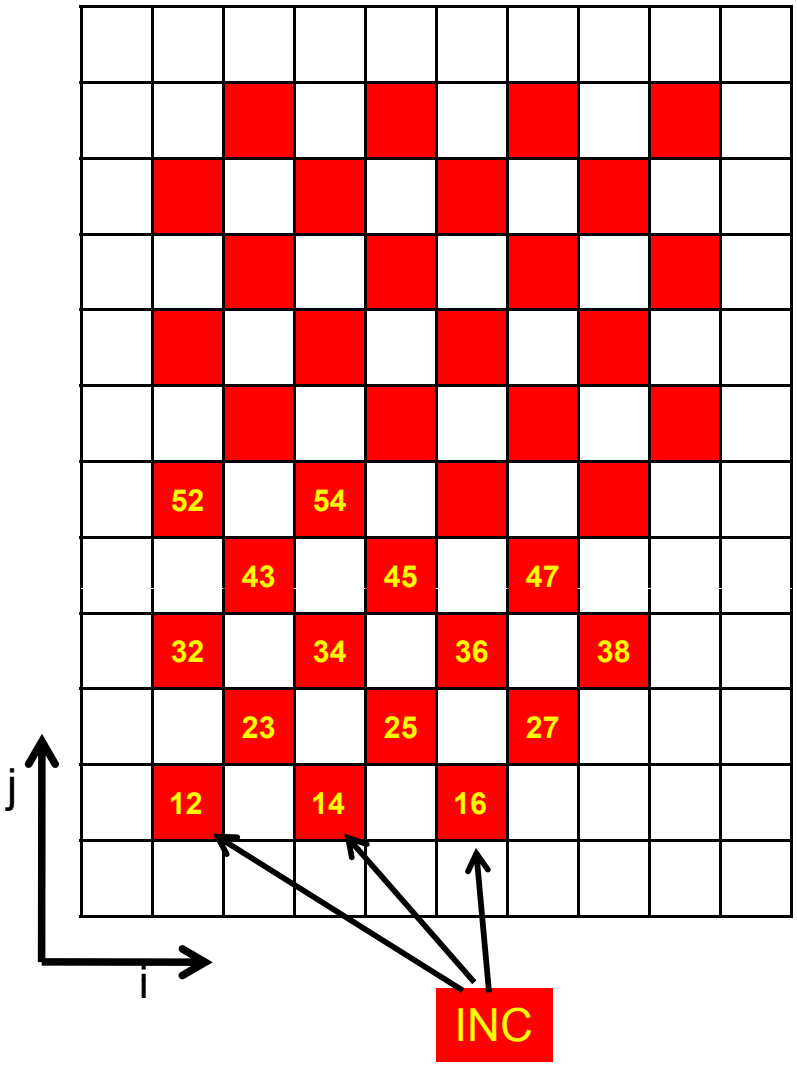
```
!$OMP PARALLEL DO PRIVATE(INC,ANS)
   DO IDO=1,NREDD
    INC = INDRED(IDO)
    ANS = AN(INC,1)*PHI(INC+1)      &
        + AN(INC,2)*PHI(INC-1)      &
        + AN(INC,3)*PHI(INC+INPD)   &
        + AN(INC,4)*PHI(INC-INPD)
        + SU(INC)
    PHI(INC) = ANS/AP(INC)
   ENDDO
!$OMP END PARALLEL DO
!$OMP PARALLEL DO PRIVATE(INC,ANS)
   DO IDO=1,NBLACKD
    INC = INDBLACK(IDO)
    ANS = AN(INC,1)*PHI(INC+1)      &
        + AN(INC,2)*PHI(INC-1)      &
        + AN(INC,3)*PHI(INC+INPD)   &
        + AN(INC,4)*PHI(INC-INPD)
        + SU(INC)
    PHI(INC) = ANS/AP(INC)
   ENDDO
!$OMP END PARALLEL DO
```



Red-Black Solver (Red Loop):

```
!!Red Loop of Red-Black-Solver
!$OMP PARALLEL DO PRIVATE(INC,ANS)
   DO IDO=1,NREDD
    INC = INDRED(IDO)
    ANS = AN(INC,1)*PHI(INC+1)      &
        + AN(INC,2)*PHI(INC-1)      &
        + AN(INC,3)*PHI(INC+INPD)   &
        + AN(INC,4)*PHI(INC-INPD)   &
        + SU(INC)
    PHI(INC) = ANS/AP(INC)
   ENDDO
!$OMP END PARALLEL DO
```

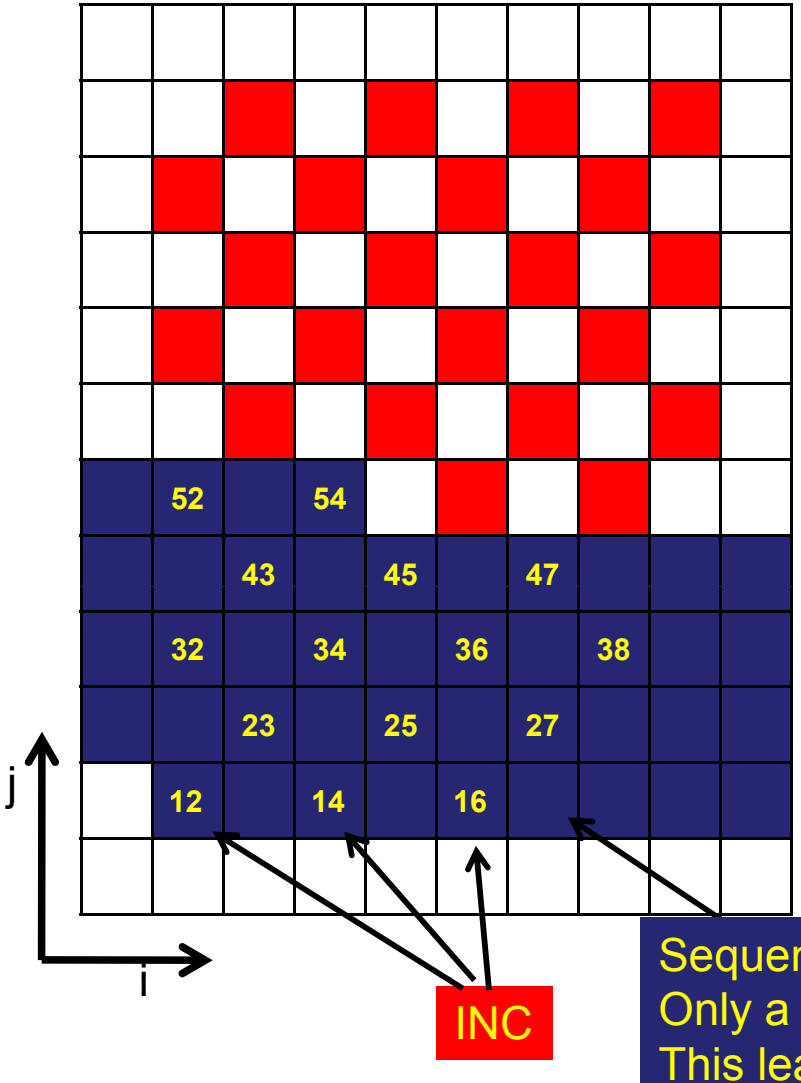
Problem of Rbgauss Subroutine



Only the Red AN / AP Elements are required

```
!!Red Loop of Red-Black-Solver
!$OMP PARALLEL DO PRIVATE(INC,ANS)
   DO IDO=1,NREDD
    INC = INDRED(IDO)
    ANS = AN(INC,1)*PHI(INC+1)      &
        + AN(INC,2)*PHI(INC-1)      &
        + AN(INC,3)*PHI(INC+INPD)   &
        + AN(INC,4)*PHI(INC-INPD)   &
        + SU(INC)
    PHI(INC) = ANS/AP(INC)
   ENDDO
!$OMP END PARALLEL DO
```

Problem of Rbgauss Subroutine

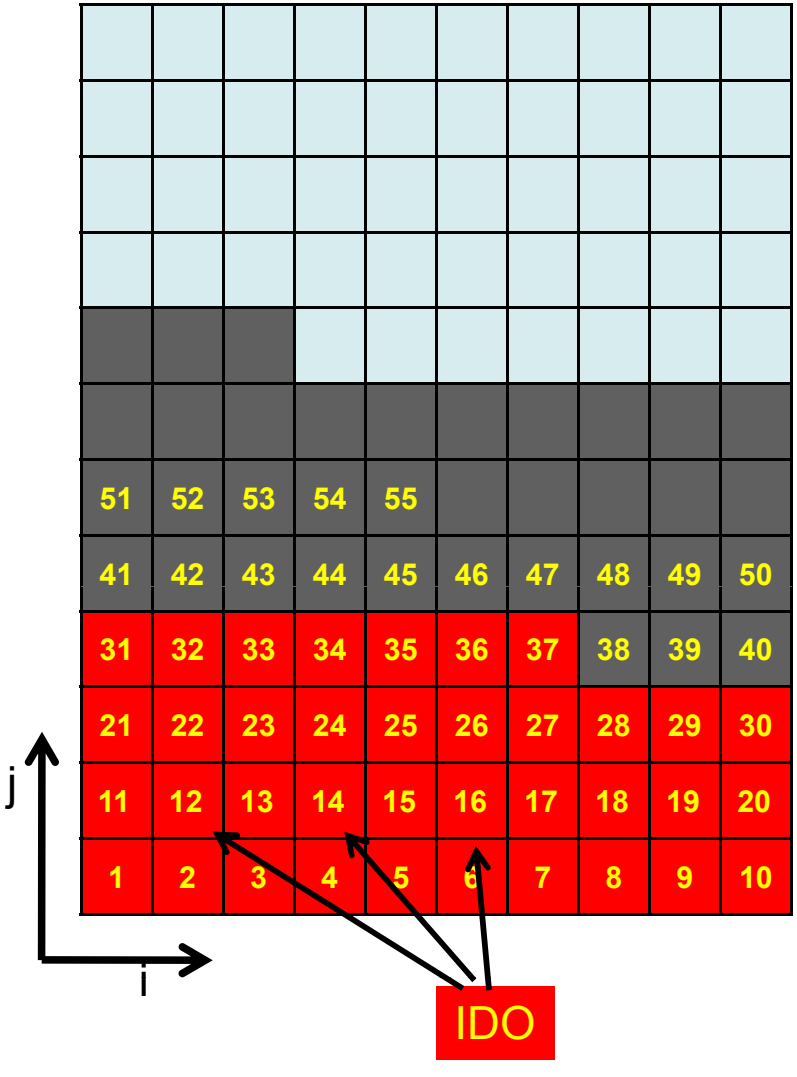


Only the Red AN / AP Elements are required

```
!!Red Loop of Red-Black-Solver
!$OMP PARALLEL DO PRIVATE(INC,ANS)
   DO IDO=1,NREDD
    INC = INDRED(IDO)
    ANS = AN(INC,1)*PHI(INC+1)      &
        + AN(INC,2)*PHI(INC-1)      &
        + AN(INC,3)*PHI(INC+INPD)   &
        + AN(INC,4)*PHI(INC-INPD)   &
        + SU(INC)
    PHI(INC) = ANS/AP(INC)
   ENDDO
!$OMP END PARALLEL DO
```

Sequential Cache Line:
Only a few elements loaded into cache are used !
This leads to a significant overhead in memory traffic !

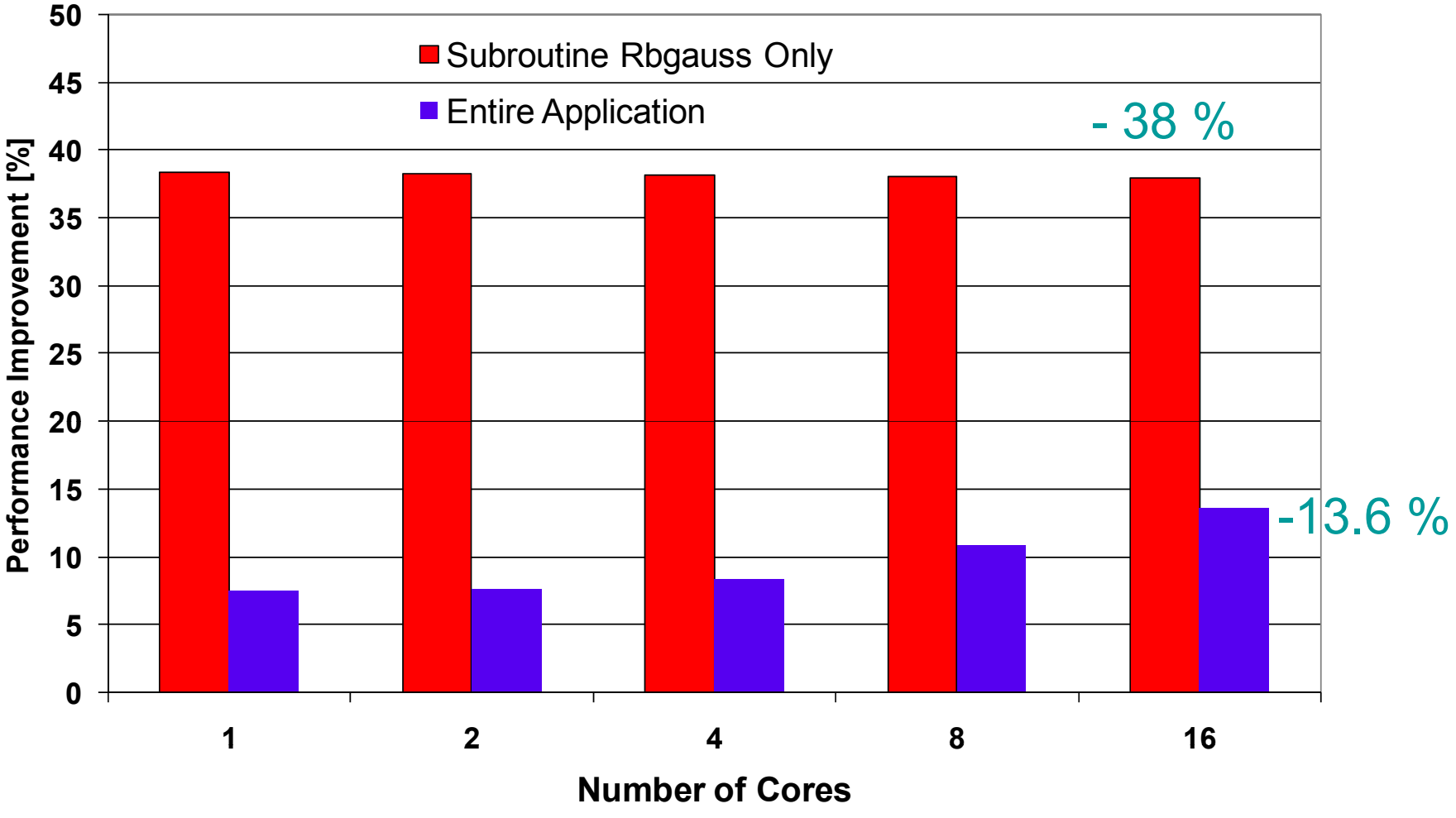
Improvement of Rbgauss Subroutine



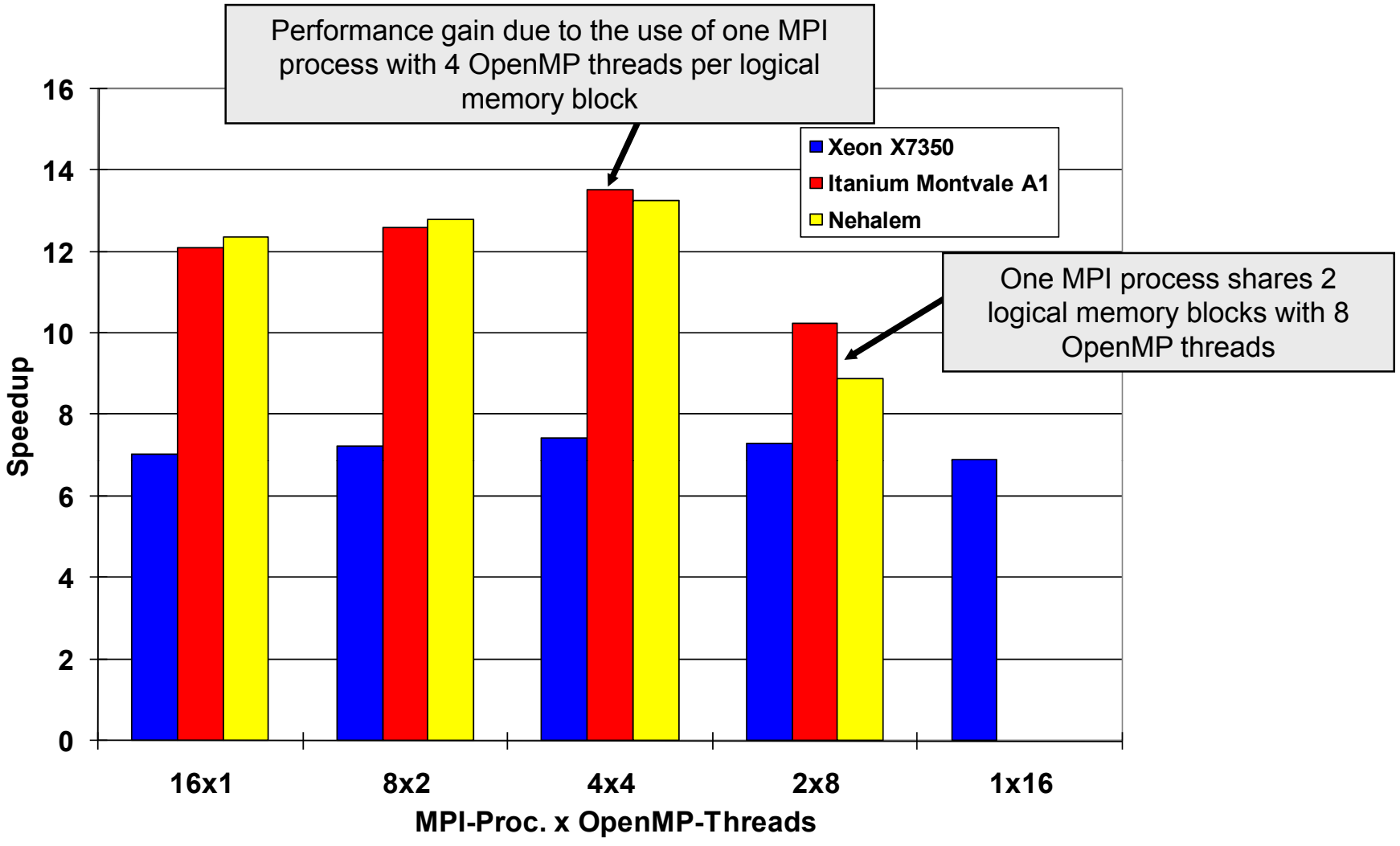
Reorientation of AN / AP Field:

```
!!Red Loop of Red-Black-Solver
!$OMP PARALLEL DO PRIVATE(INC,ANS)
   DO IDO=1,NREDD
    INC = INDRED(IDO)
    ANS = AN(IDO,1)*PHI(INC+1)      &
        + AN(IDO,2)*PHI(INC-1)      &
        + AN(IDO,3)*PHI(INC+INPD)   &
        + AN(IDO,4)*PHI(INC-INPD)   &
        + SU(INC)
    PHI(INC) = ANS/AP(IDO)
   ENDDO
!$OMP END PARALLEL DO
```

Performance improvement through Rbgauss restructuring



Maximizing the usage of the available memory bandwidth with a Hybrid (MPI & OpenMP) approach



Summary

- The ParMA tool chain allows a thorough analysis of the performance bottlenecks and the identification of the most promising measures for performance improvements.

- Even for a HPC-application like the 3D-Combustion Modelling Software RECOM-AIOLOS a performance gain of roughly 34% for the entire application has been achieved.