# Parallel Sparse Matrix Vector Product with OpenMP for SMP in Code Saturne

V. Szeremi[1], L. Anton[1,2], C. Moulinec[1], C. Evangelinos[3], Y. Fournier[4]

[1]STFC Daresbury, [2]Cray UK, [3]IBM Research US, [4]EDF R&D France



*Higher-order DG methods and finite element software for modern architectures*

*Bath 31 May - 2 Jun*

# Cray EMEA

- **EMEA headquarters in Bristol**
  - Applications & systems support
  - R&D interconnect fabric design
  - Cray EMEA Research Lab (CERL)
- **Cray aims to generate HPC & DA technology in Europe**
  - CERL participates to exascale research initiatives: EPiGRAM, CRESTA

# Cray EMEA Research Lab (CERL)

Performs research and development in strategic areas that strengthen Cray's leadership in high-performance computing and data analytics, and which deepen Cray's involvement in the scientific and technical communities of EMEA

- Director: Adrian Tate
- **Activity themes:**
  - Hardware and software co-design
  - Advanced workloads
  - Configurations
  - Advanced software
- **Interface for European development programs**
  - http://www.cray.com/company/collaboration/organizations-initiatives/cray-emea-research-lab

# Code Saturne (http://code-saturne.org)

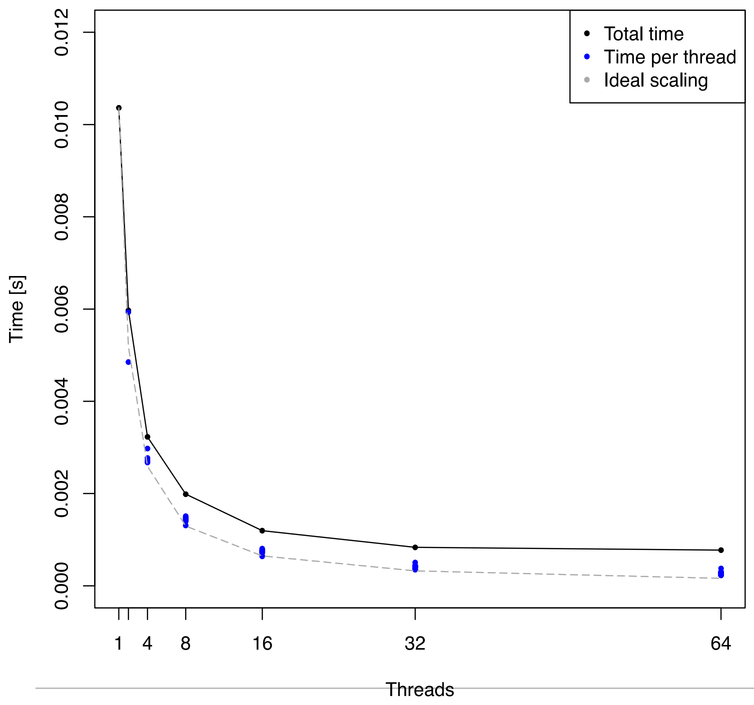
- open-source CFD code
  - 2D,3D flows, steady or unsteady, laminar or turbulent, incompressible or weakly dilatable, isothermal or not, scalar transport,…
  - Physical models modules: gas, coal, heavy fuel combustion, particle tracking, etc,
- mainly developed by EDF (France)
- Fortran, C, Python, ~350k lines of code
- fully validated production versions with long-term support released every two years
- Part of restricted PRACE Unified European Applications benchmark Suite (UEABS, http://www.prace-ri.eu/ueabs/)

# Motivation

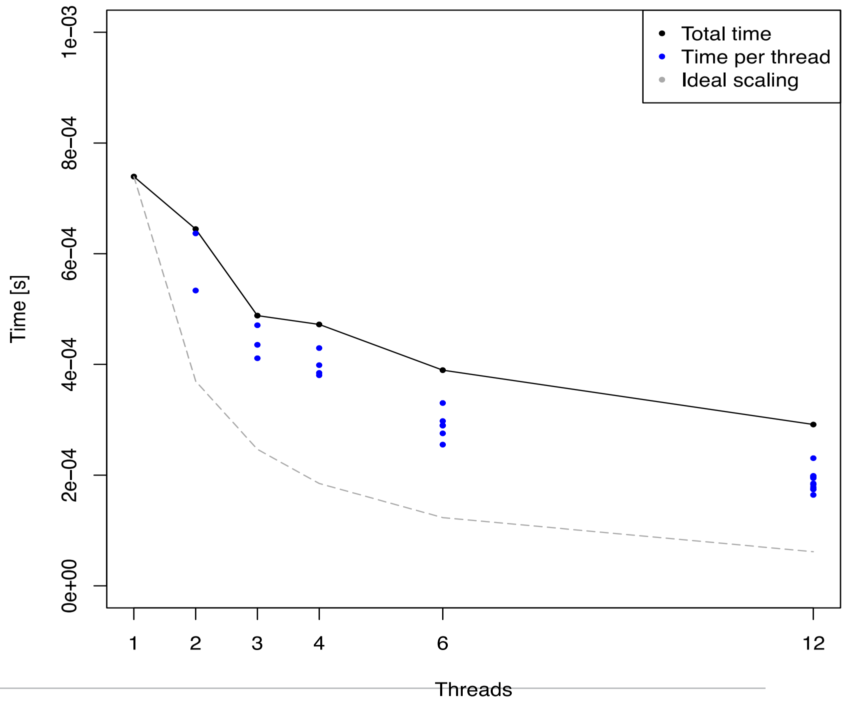
- **In general, optimisation of scientific apps is periodically required in order to take advantage of the evolving hardware architectures**
  - multi/many cores (+GPUs)
  - hyperthreads
  - Vector processing
  - Cache layout
  - interconnect
  - I/O
- **Specifically, the CS project was defined to explore ways to improve the OpenMP scaling**
  - CS run time in incompressible flow simulations is dominated by the pressure solver
    - sparse matrix vector product

# Native OpenMP: BQG vs Intel

Matrix vector (Cavity 145k, Blue Joule)




Matrix vector (Cavity 145k, ARCHER)



# Sparse Matrix Vector product in Code Saturne

Code_Saturne formats for sparse matrix storage and associated MV product algorithms:

- **Native –** using Code_Saturne native sparse matrix storage format
- **Native OpenMP –** multiple groups, with threads having non-overlapping regions within a group
- **CSR –** compressed sparse row; rows divided between threads
- **MSR –** modified compressed

# Native MV (symmetric)

```
for (face_id = 0; face_id < ms->n_faces; face_id++) {
  ii = face_cel_p[2*face_id] -1;
  jj = face_cel_p[2*face_id + 1] -1;
  y[ii] += xa[face_id] * x[jj];
  y[jj] += xa[face_id] * x[ii];
}
```

# Native OpenMP MV

```
for (g_id = 0; g_id < n_groups; g_id++) {
# pragma omp parallel for private(face_id, ii, jj)
  for (t_id = 0; t_id < n_threads; t_id++) {
    for (face_id = group_index[(t_id*n_groups + g_id)*2];
      face_id < group_index[(t_id*n_groups + g_id)*2 + 1];
      face_id++) {
      ii = face_cel_p[2*face_id] -1;
      jj = face_cel_p[2*face_id + 1] -1;
      y[ii] += xa[face_id] * x[jj];
      y[jj] += xa[face_id] * x[ii];
    }
  }
}
```

# Native OpenMP: time tracing

- Extrae/Paraver trace of native sparse matrix vector
- product used within PCG showing synchronisation
  - Top: executed function; Middle: useful work; Bottom: OpenMP loop

# Blocked Native Algorithm

- faces are grouped into blocks:
  - Each block is guaranteed to update cells in selected cell index range
  - Can be parallelized with OpenMP
    - synchronization free
    - better cache utilisation
- different block types to handle:
  - diagonal: y[ii] and y[jj]
  - off-diagonal: either y[ii] or y[jj]
- however: additional work with increasing number of blocks
- corresponding matrix vector product alogrithms are integrated into Code_Saturne, including the autotuning framework
- algorithmic variations for better load balance

# Blocked Native Code

```
for (g_id = 0; g_id < n_groups; g_id++) {
# pragma omp parallel for
private(face_id, ii, jj)
  for (t_id = 0; t_id < n_threads;
t_id++) {
    for (face_id =
group_index[(t_id*n_groups + g_id)*2];
      face_id <
group_index[(t_id*n_groups + g_id)*2 +
1];
      face_id++) {
       ii = face_cel_p[2*face_id] -1;
       jj = face_cel_p[2*face_id + 1] -1;
       y[ii] += xa[face_id] * x[jj];
       y[jj] += xa[face_id] * x[ii];
    }
  }
}
```

```
# pragma omp parallel private ( ii, jj, faceid )
{
  int ith = omp_get_threadnum( ) ;
  int jb,fs,fe, fst ,fet ;
  for ( jb =0; jb < nthreads ; ++jb ){
    fs = ms->th_blk[ith][jb].s;
    fe = ms->th_blk[ith][jb].s +
            ms->th_blk[ith][jb].n;
    if ( jb == ith){
      for (face_id = fs ; face_id < fe;
            face_id++){
        ii = face_cel_p [2 * face_id ] -1;
        jj = face_cel_p [2 * face_id + 1] -1;
         y[ii] += xa [face_id ] * x[jj] ;
         y[jj] += xa [face_id ] * x[ii] ;
     }
  }
  else{
….
```

# Blocked Native Code (cont'd)

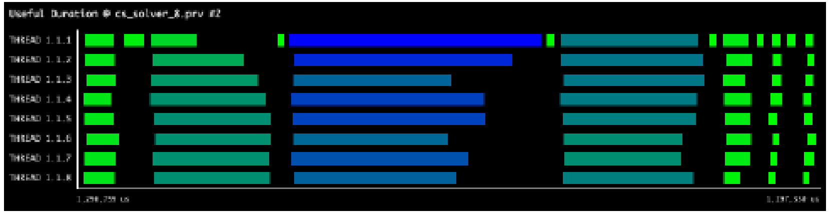
```
if ( jb > ith){
        for (face_id = fs; face_id < fe;
face_id++) {
          ii = face_cel_p[2*face_id] -1;
          jj = face_cel_p[2*face_id + 1] -1;
          y[ii] += xa2[face_id] * x[jj];
        }
      }
      else{
        for ( face_id = fs; face_id < fe;
face_id++) {
          ii = face_cel_p[2*face_id] -1;
          jj = face_cel_p[2*face_id + 1] -1;
          y[jj] += xa2[face_id] * x[ii];
...
```

# Blocked native: Variations

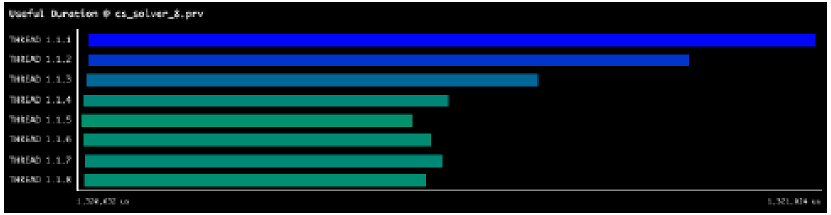
- **Blocked** – split faces into blocks, each thread works on a separate range in the y vector; diagonal blocks and off-diagonal blocks
- **SECO / PACO** – serial / parallel setting of coefficients
- **BAL / UNB** – setting blocks boundaries so that: number of rows are equally distributed (UNB) or number of faces are equally distributed (BAL)
- **INCDIAG** – include diagonal handling and zeroing of y in the processing of the diagonal blocks
- **EXHA** – exclude halo region in the y vector calculation
- **MULTIBLOCKS fixed block size: 1k, 2k, etc.** - dynamic scheduled OpenMP loop for load balance
- **MULTIBLOCKS fixed number of blocks: 2TT, 3TT** - blocks split between static and dynamic scheduled OpenMP loop for load balance

# Qualitative Comparison

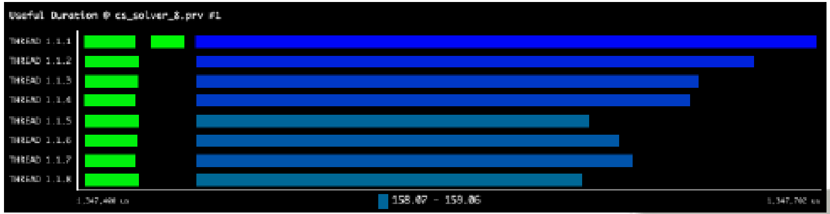
- Native



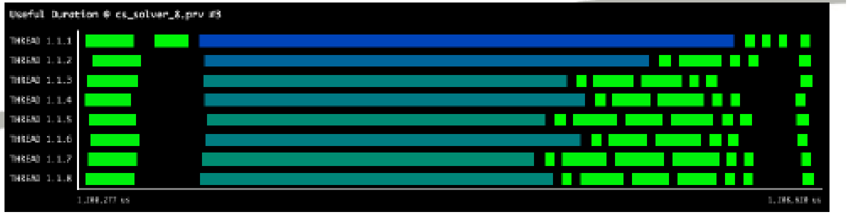

- MSR
  (modified compressed sparse row)




- Blocked
  one row block per thread




- Blocked
  with mixed static/ dynamic scheduling



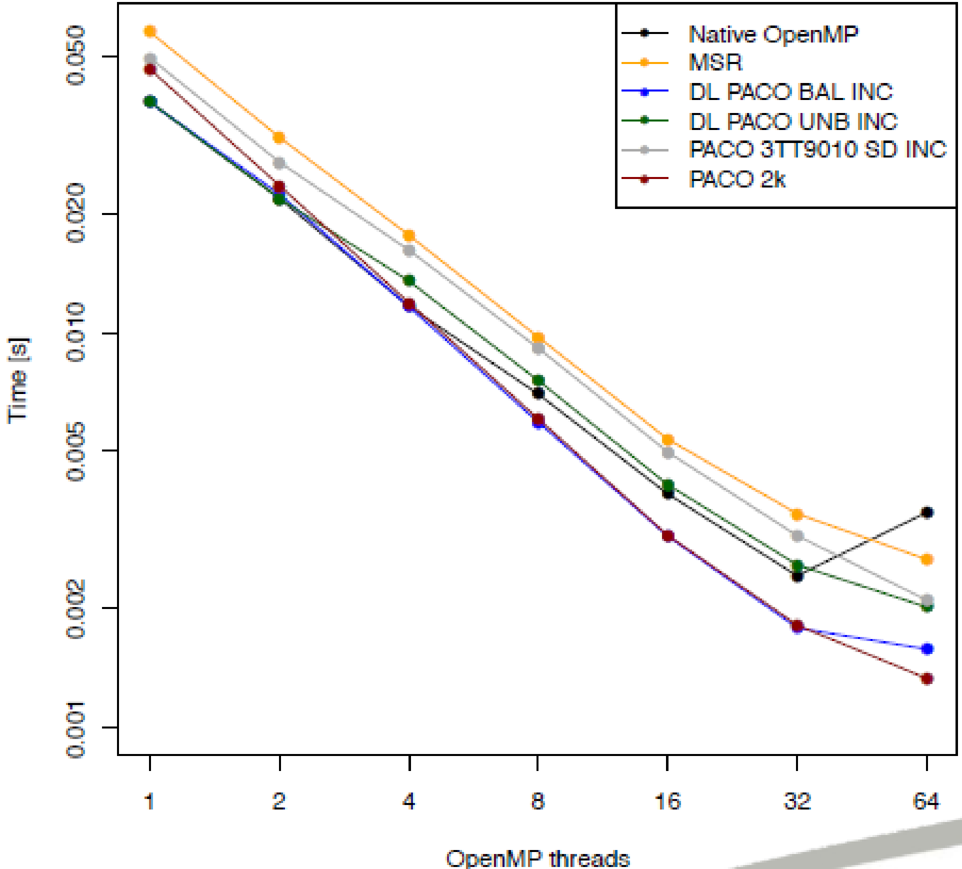
# Test setup

- cavity test case
  - laminar lid-driven cavity flow
  - tetrahedral cells
- number of cells: 145k, 500k, 1800k, (13M)
- time taken from autotuning framework

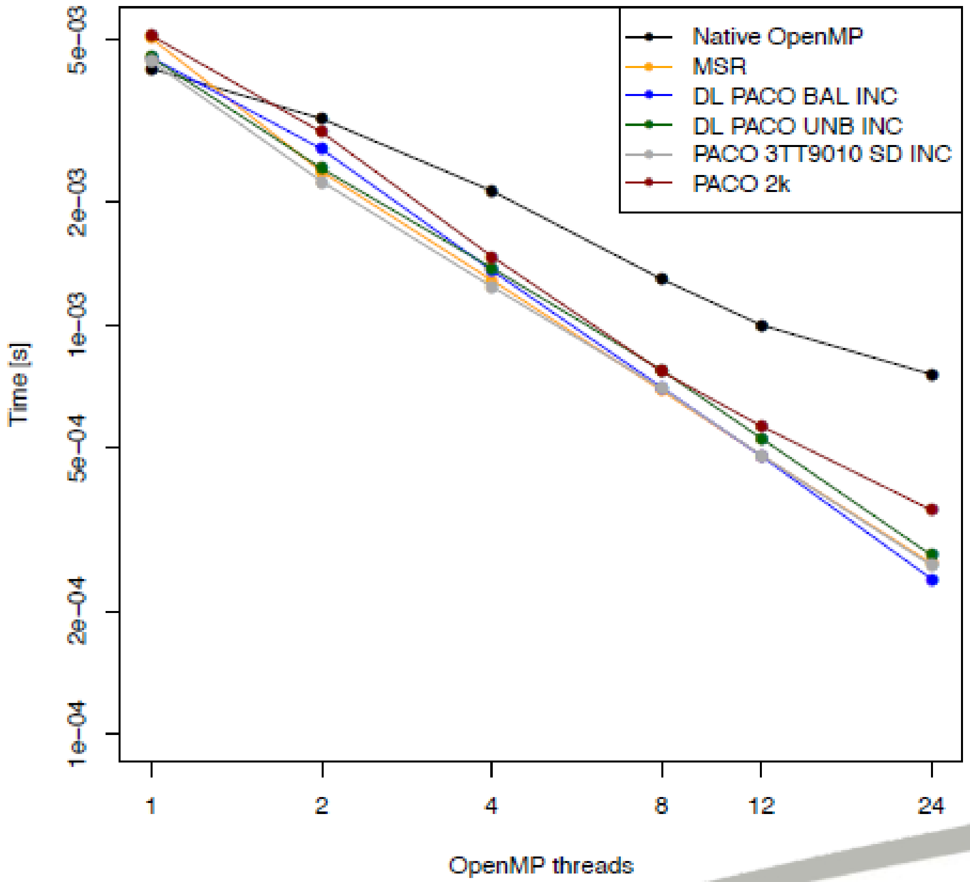
# Test Setup (cont'd)

- Blue Joule
  - Blue Gene/Q, 1x 16-core 1.60 GHz A2 PowerPC
- Blue Wonder
  - iDataPlex, 2x 8-core 2.6 GHz Intel Xeon (Sandybridge)
- Phase 2 Wonder
  - NextScale, 2x 12-core 2.7 GHz Intel Xeon E5-2697 v2 (Ivybridge)
- ARCHER
  - Cray XC30, 2x 12-core 2.7 GHz Intel Xeon E5-2697 v2(Ivybridge)

# Results: Blue Joule, cavity test, 500k

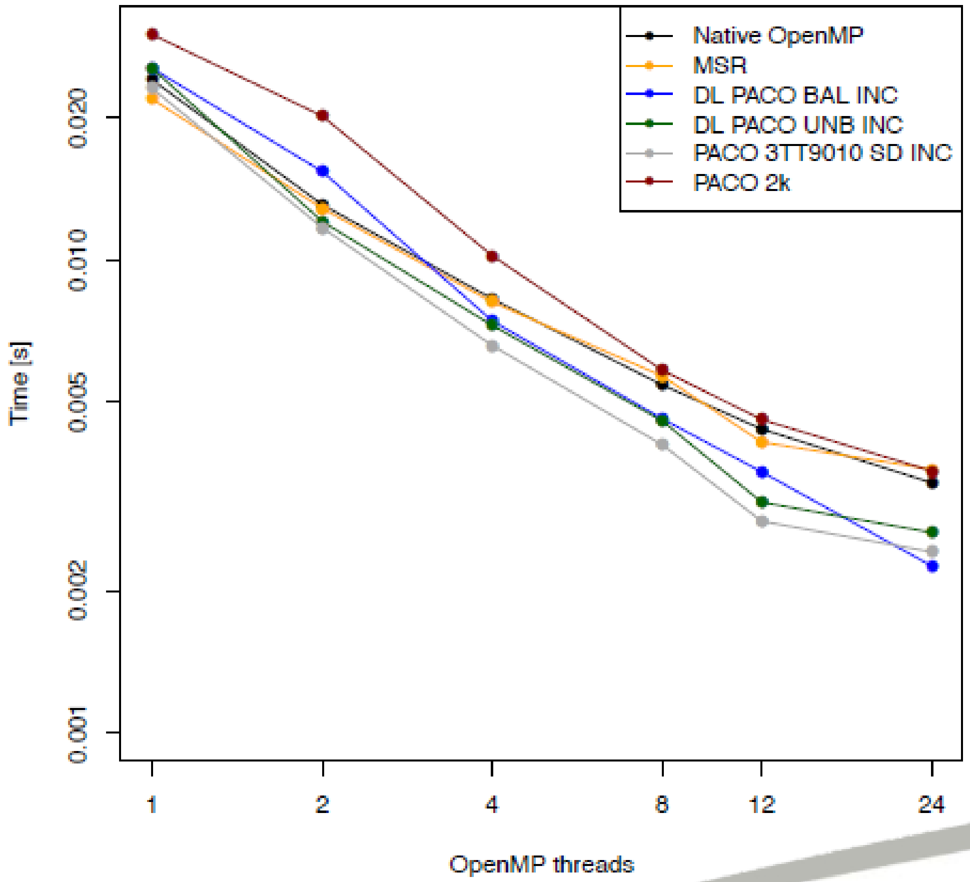


# Results: Ivybridge cavity test, 500K

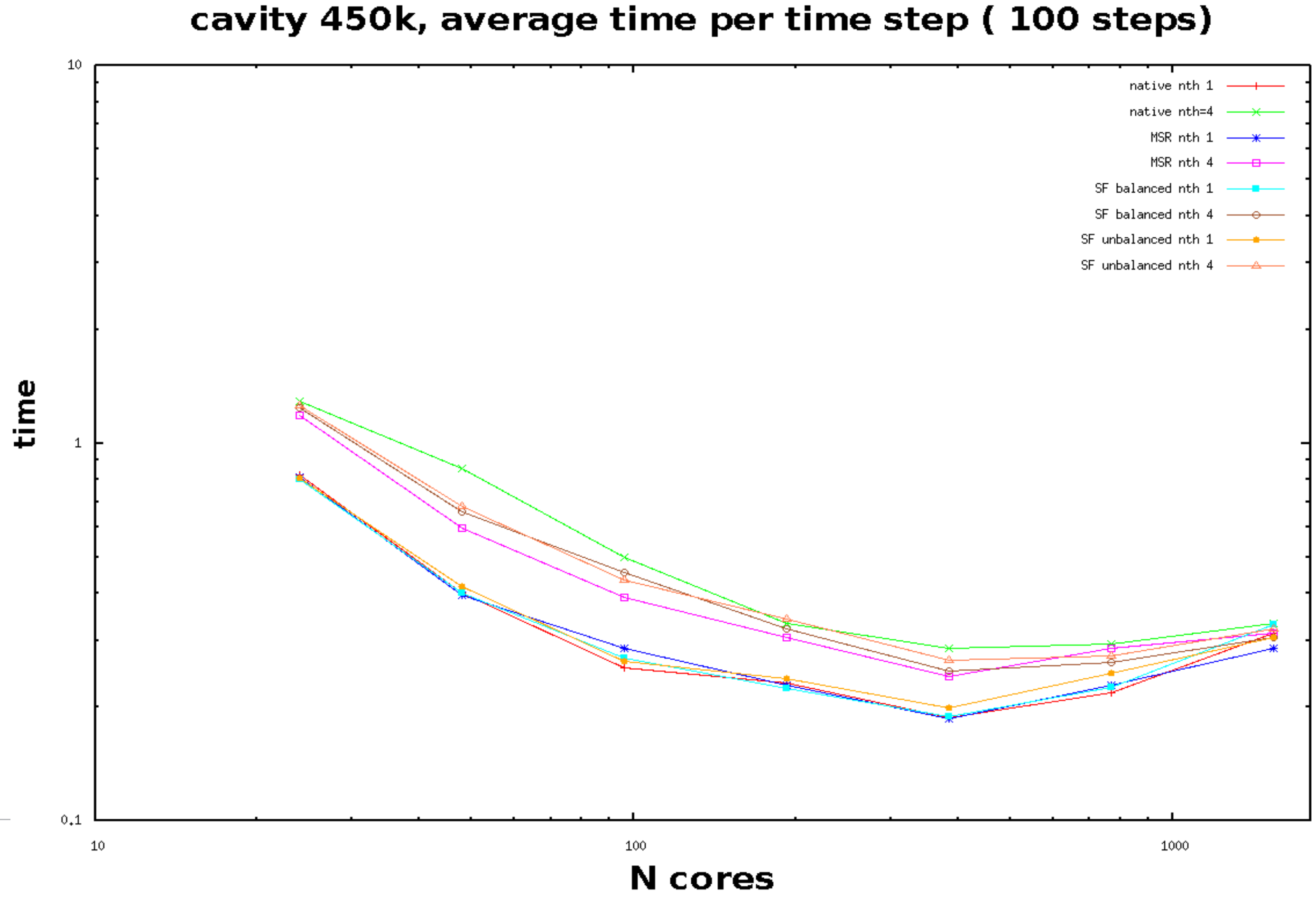



ALYZE

# Results: IvyBridge, cavity test 1800k



# Result: MPI+OpenMP, cavity 450k (ARCHER)



# Conclusions

- we propose a blocked native storage and matrix vector product implementation
  - synchronisation free
  - Imporves load balancing & cache
  - Good performance on BGQ and Intel processors
- algorithm variations for improved load balance
  - Performance depends on CPU architecture and mesh size
- implemented in Code_Saturne, including autotuning framework
  - tests on cavity case using autotuning framework

# Acknowledgements

CRAY

EPSRC
Engineering and Physical Sciences Research Council

This work was partially supported by the UK Engineering and Physical Sciences Research Council (EPSRC) through the **Software Outlook** programme.

Science & Technology Facilities Council

We acknowledge use of Hartree Centre resources in this work. The STFC Hartree Centre is a research collaboratory in association with IBM providing High Performance Computing platforms funded by the UK's investment in e-Infrastructure. The Centre aims to develop and demonstrate next generation software, optimised to take advantage of the move towards exa-scale computing.

archer

This work used the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk).

# References


- **Y. Fournier, J. Bonelle, C. Moulinec, Z. Shang, A.G. Sunderland, J.C. Uribe,** *"Optimizing Code_Saturne computations on Petascale systems"*, Computers & Fluids, 45, 103-108, 2011.
- **A. Alvermann, A. Basermann, H. Fehske, M. Galgon, G. Hager, M. Kreutzer, L. Krämer, B. Lang, A. Pieper, M. Röhrig-Zöllner, F. Shahzad, J. Thies, and**
- **G. Wellein,** *"ESSEX: Equipping Sparse Solvers for Exascale"*. To be published in: L. Lopes (ed.), Proc. Euro-Par 2014 Workshops, LNCS vol. 8805, 8806, Springer, 2014.
- **R. Aubry, G. Houzeaux, M. Vázquez, J. M. Cela,** *"Some useful strategies for unstructured edge-based solvers on shared memory machines"*, International Journal for Numerical Methods in Engineering, 85, 537-561, 2011.
- **V. Kale, W. D. Gropp,** "Load balancing for regular meshes on SMPs with MPI", EuroMPI '10, 2010.

# Legal Disclaimer

*Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.*

*Cray Inc. may make changes to specifications and product descriptions at any time, without notice.*

*All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.*

*Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.*

*Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.*

*Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.*

# Q&A

**Lucian Anton** lanton@cray.com