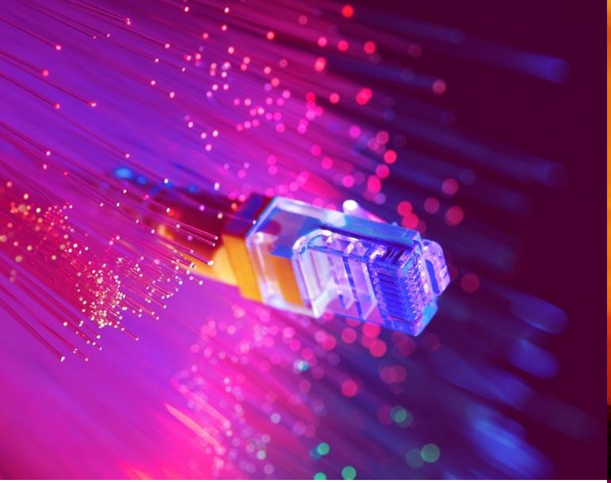
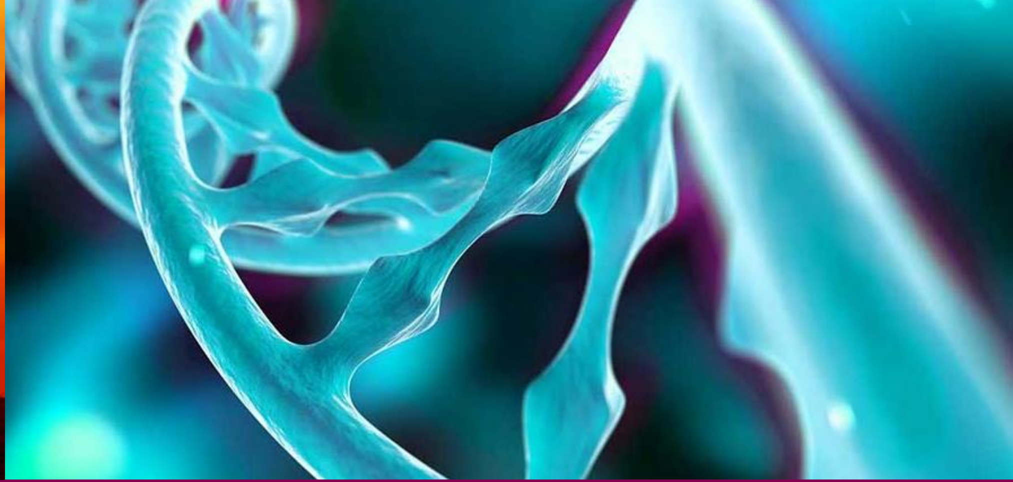




Elmer/Ice splinter meeting

EGU GA 2017, Vienna, Austria


CSC – Suomalainen tutkimuksen, koulutuksen, kulttuurin ja julkishallinnon ICT-osaamiskeskus

Updates on Elmer/Ice

Thomas Zwinger, Peter Råback, Mika Malinen,
Juha Ruokolainen, Juhani Kataja (CSC)

Special guests: Josefin Ahlkrona (Univ Kiel),
Mikko Byckling (Intel Corp.)

Emergence solver

Emergence solver

- New solver computing emergence velocity:
  - Solver Fortran File: `Emergence.F90`
  - Solver Name: `GetEmergenceVelocity`
  - http://elmerice.elmerfem.org/wiki/doku.php?id=solvers:emergence

Computes the surface emergence velocity (= negative equilibrium mass balance) at the surface using a scalar product of the surface normal and the ice velocity at the surface

- Needed other solvers:
  - `ElmerIceSolvers ComputeNormalSolver`
  - `(built-in) Navier-Stokes`

Emergence solver

- Kinematic free surface condition:

$$\frac{\partial h}{\partial t} + \underbrace{u\frac{\partial h}{\partial x} + v\frac{\partial h}{\partial y} - w}_{=-v_{em}} = a_{\perp}||\nabla F_s||$$

  - With

$$\nabla F_s = z - h$$

  - Surface normal

$$\boldsymbol{n} = \nabla F_s/||\nabla F_s|| = (-\frac{\partial h}{\partial x}, -\frac{\partial h}{\partial y}, 1)/||\nabla F_s||$$

  - Cheating:

$$||\nabla F_s|| = \sqrt{(\frac{\partial h}{\partial x})^2 + (\frac{\partial h}{\partial y})^2 + 1} = 1 + \mathcal{O}(\varepsilon)$$

ISCAL

Slides courtesy of
Josefin Ahlkrona (Univ Kiel)

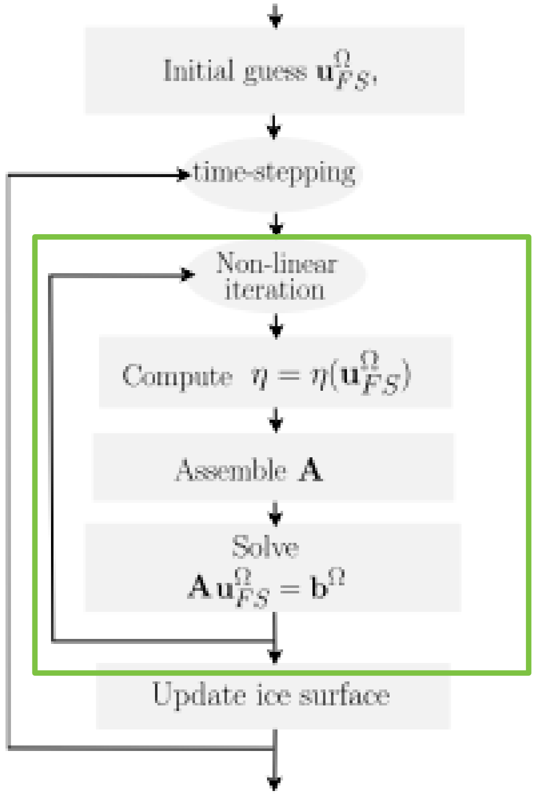
ISCAL

- Ice Sheet Coupled Approximation Levels (ISCAL)
- Error estimate (based on the difference of wither a FS or a previous ISCAL solution - SIA)
- The domain is decomposed according to this error estimate, given a certain threshold
- Solution of the (in size reduced) system

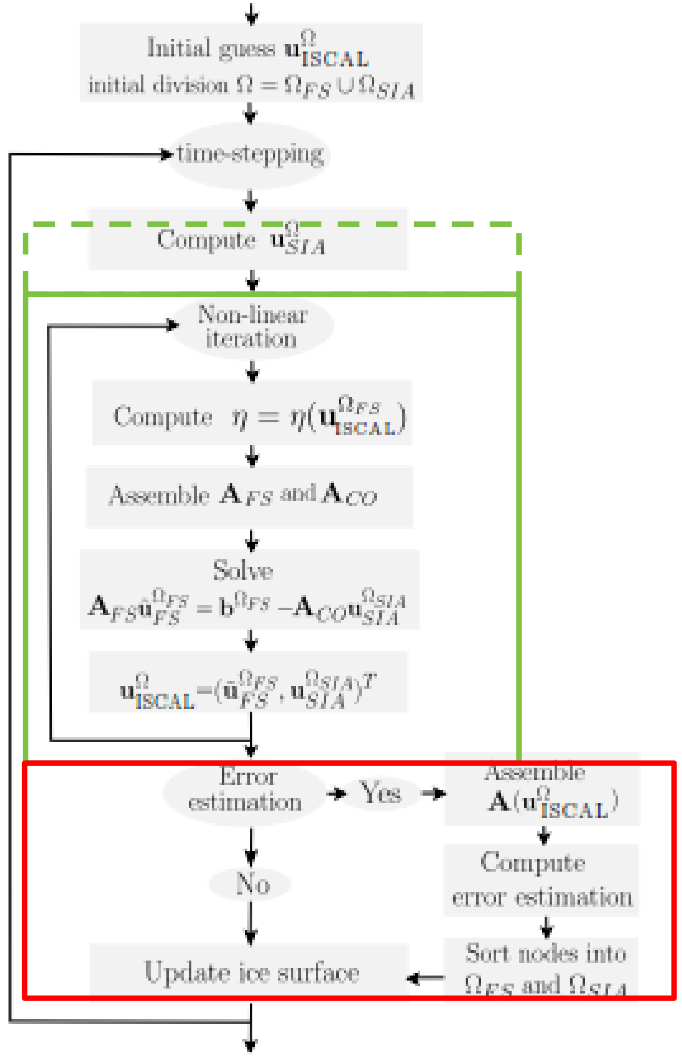
$$\boldsymbol{A}_{FS} \cdot \tilde{\boldsymbol{u}}_{FS}|_{\Omega_{FS}} = \boldsymbol{b}|_{\Omega_{FS}} - \boldsymbol{A}_{CO} \cdot \boldsymbol{u}_{SIA}|_{\partial\Omega_{FS}}$$

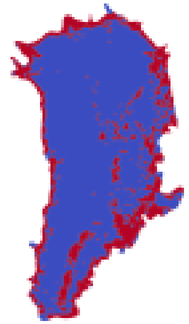
ISCAL

Original Stokes Algorithm



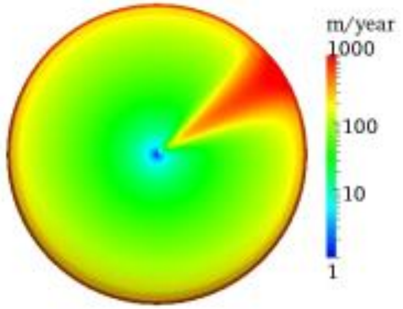
ISCAL Algorithm





ISCAL

Ice stream tracking



Velocity (ISCAL)

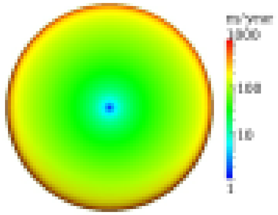
SIA, and full Stokes areas, 1 month

SIA, and full Stokes areas, 6 months

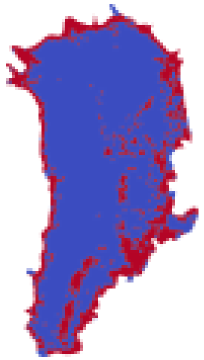
SIA, and full Stokes areas, 11 months

ISCAL

Speedup ≈ 9



# nodes	Model	FS nodes (%)	Assembly (%)	Solve (%)	Error Calc. (%)	# iter.
257000	FS	100.0	84.8	14.6	-	12.9
	ISCAL	6.8	78.3	11.2	5.1	12.9



Greenland, tolerance 10%: 4 times faster for 465056 nodes where 14 % are FS nodes.

ISCAL

- ISCAL currently demands installation of a separate Elmer branch, called elmerice-iscal
  - It compiles basically similar than Elmer/Ice, just by switching to the right source in git (`git checkout elmerice-iscal`)
  - ISCAL itself not yet included in cmake - needs manual building with script from inside the source – will be fixed, soon
  - Once everything works in iscal-branch, we will merge into elmerice
- Documentation (still some construction zone here) under: http://elmerice.elmerfem.org/wiki/doku.php?id=solvers:iscal
- Example (in elmerice-iscal): `elmerfem/elmerice/Solvers/ISCAL/test_circularice`

ISCAL

References:

Ahlkrona, J., P. Lötstedt, N. Kirchner, and T. Zwinger, 2016. ***Dynamically coupling the non-linear Stokes equations with the shallow ice approximation in glaciology: Description and first applications of the ISCAL method***. J. Comp. Phys., **308**, 1-19, doi:10.1016/j.jcp.2015.12.025.

ISCAL - Summary

- Code deployed in separate branch
- Ready to play with
- Still needs some work to bring to production
- When would I use it?:
  - Longer term simulation with changing fast flow pattern

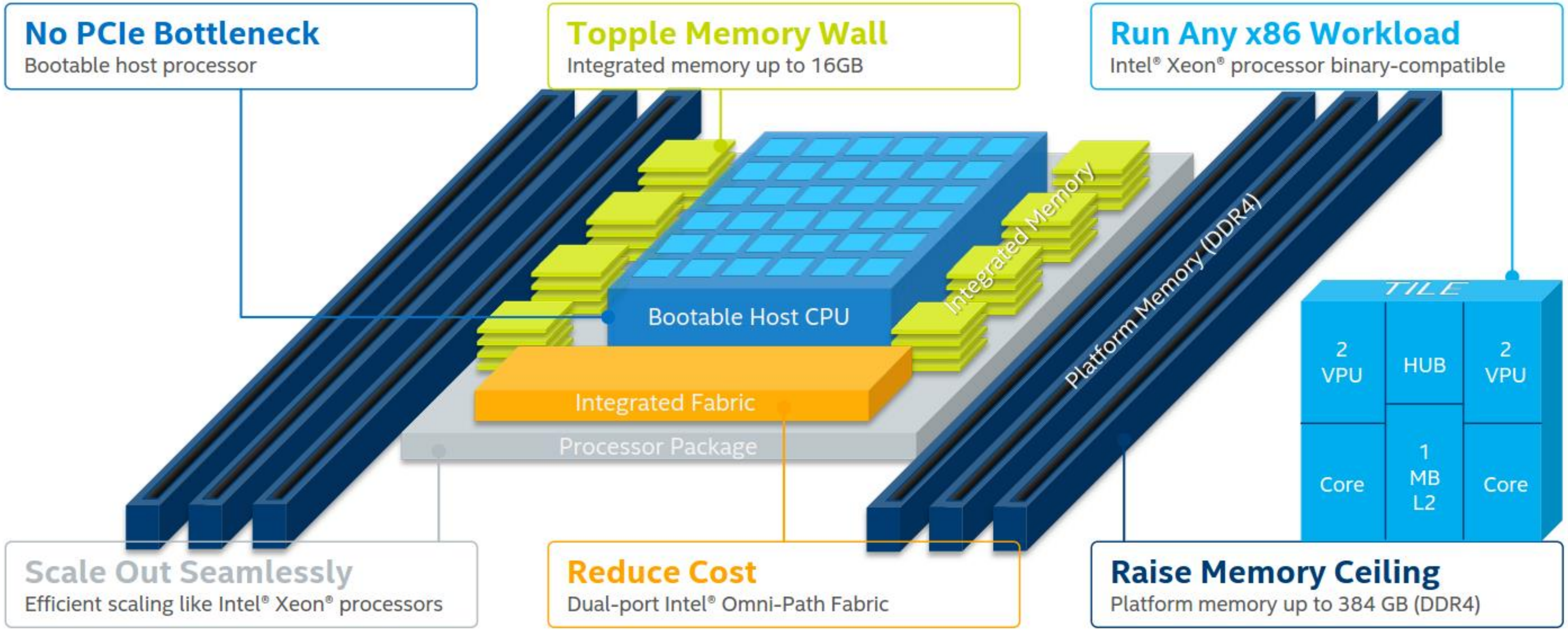
Elmer(/Ice) on Xeon Phi

Mikko Byckling (Intel)
Juhani Kataja (CSC)

Porting Elmer to MIC

- Porting work started already Q2/12
- Focus to build ElmerSolver on a MIC
  - MIC = Many Integrated Cores
- Cooperation with Mikko Byckling (Intel) within *Intel Parallel Computing Center* (IPCC)

Porting Elmer to MIC



[1]Reduced cost based on Intel internal estimate comparing cost of discrete networking components with the integrated fabric solution

Porting Elmer to MIC

- Internally OpenMP threading supported by
  - Solver API routines related to element assembly
  - Element assembly loop of some solvers already implemented
  - Time integration routines
  - Sparse matrix vector products
- Library support for OpenMP exists in
  - External BLAS routines
  - External LAPACK routines
  - Direct solvers such as Cholmod, SPQR and Pardiso

Porting Elmer to MIC

- Perform disruptive changes if necessary
  - Maintain backwards compatibility
  - Build backwards compatible interfaces to new methods if necessary
- Optimization order
  - Vectorization
  - Threading
- Tools currently in use
  - Intel Vtune (to find hotspots and non-vectorizable parts of the code on the time critical path)
  - Intel Inspector XE (to find threading bugs)
- Targeting both Xeon and Xeon Phi

Porting Elmer to MIC

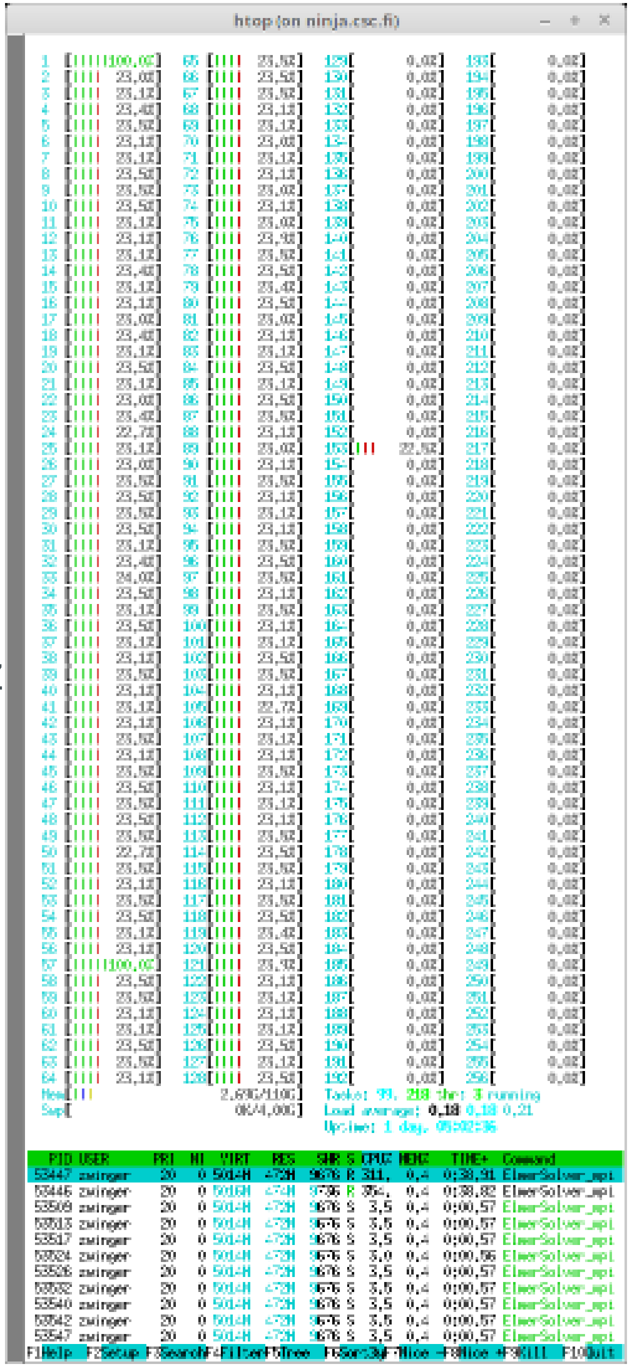
- Modern Fortran code with a modular structure
  - Initial focus on Finite element assembly
  - Improve the vectorization properties by changing the key data structures
  - Add OpenMP multithreading
- All ~50 solvers in Elmer need to be modified

= AHLOW!

```
!$omp parallel do private(Element,n,nd)
    DO t=1,active
       Element => GetActiveElement(t)
       n  = GetElementNOFNodes(Element)
       nd = GetElementNOFDOFs(Element)
       CALL LocalMatrix(Element, STIFF, FORCE, n, nd)
       CALL DefaultUpdateEquations(STIFF,FORCE,&
                                  UElement=Element)
    END DO
!$omp end parallel do
```

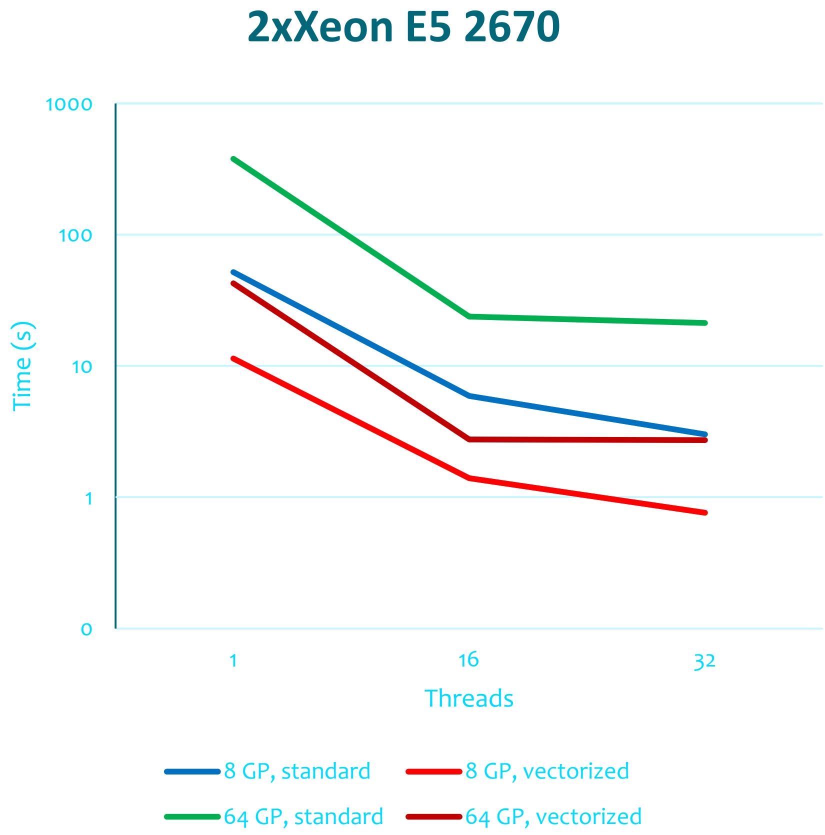
Porting Elmer to MIC

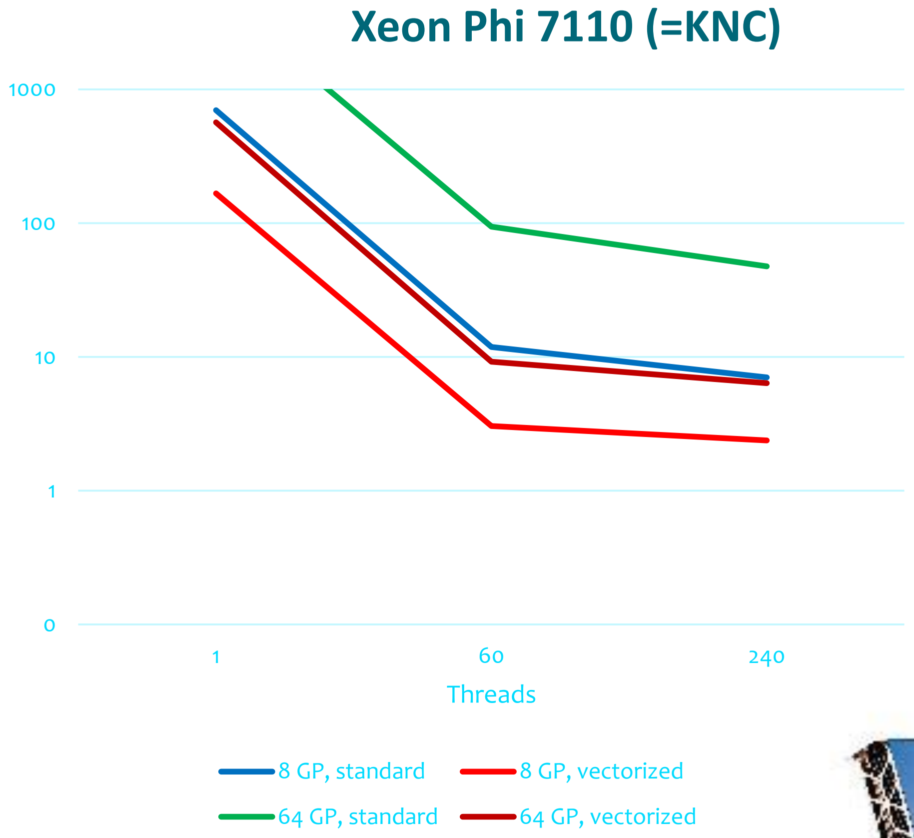
- Poisson (elliptic problem) solver with
  - Large vectors (FEM Gauss points)
  - Mesh colouring (avoid race conditions)
  - Tested on Xeon Phi developer Ninja platform
    - Intel® Xeon Phi (TM) CPU 7210 @ 1.30GHz
    - 64 cores (256 HT 4x)
    - 96GB DDR4,16GB MCDRAM
    - KNL (KNights Landing)



Porting Elmer to MIC

- Poisson model problem, 1M Hexahedral elements

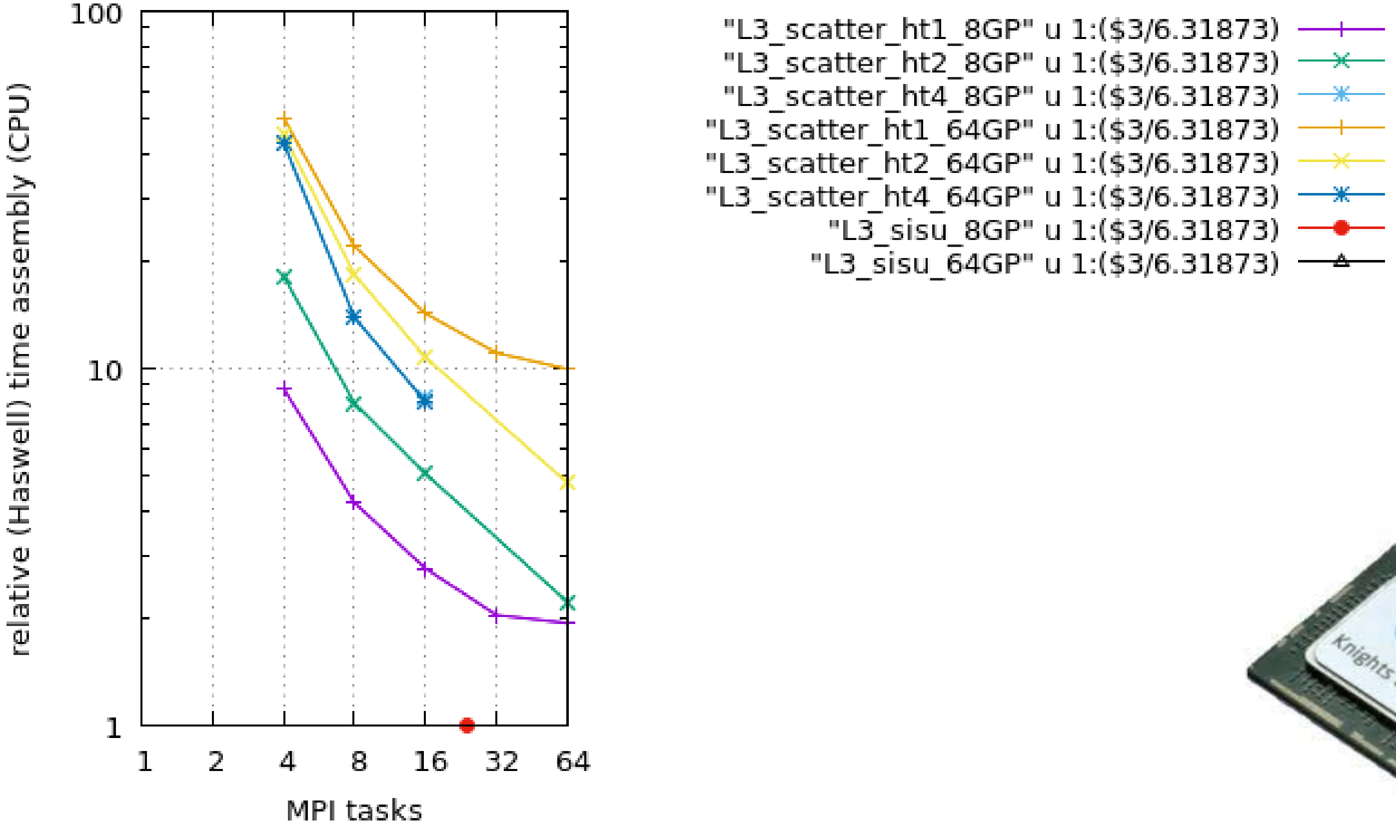




Porting Elmer to MIC

- Poisson model problem, 1M Hexahedral elements

2xXeon E5 2670 | **Xeon Phi 7210 (=KNL)**



Porting Elmer to MIC

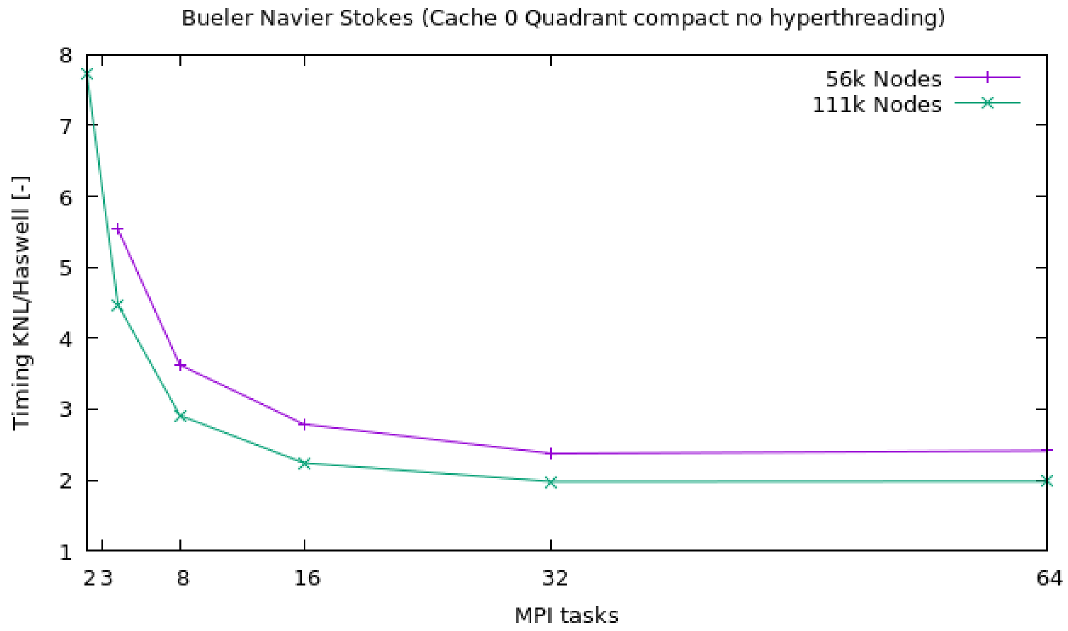
Activity supported by

eSTICC

norden

NordForsk

- Production solver used in Elmer/Ice
- Synthetic ice-sheet goemetry (Bueler-profile) with (Navier-)Stokes solver with non-linear rheology law
- Utilize (C)Pardiso
- Timing of linear system solve
- Compare with Haswell node 24 cores



Conclusions

- If you have a system based on MIC's, you can deploy Elmer/Ice with reasonable performance (similar between Xeon and Xeon Phi)
- Multi-threading (OpenMP) has been introduced to many solvers and will continue
- Assembly can utilize SIMD (=vector units) if we apply p-bubbles for stabilization
- Improvements have equally positive impact on traditional CPU's (Xeon Hasswel, Broadwell)