A UT/ORNL PARTNERSHIP
NATIONAL INSTITUTE FOR COMPUTATIONAL SCIENCES
NICS

Early Application Experiences with the Intel® MIC Architecture in a Cray CX1

R. Glenn Brook, Bilel Hadri*, Vincent C. Betro, Ryan C. Hulguin, and Ryan Braby

Cray Users Group 2012

Stuttgart, Germany – April 29 – May 3, 2012

* presenting author

Contents

- Overview on AACE
- Overview on MIC and its Configuration on the Cray CX1
- Applications code, description, porting and results
    - Euler solver and Boltzmann-BGK solver,
    - Navier-Stokes solver (2D and 3D)
    - Poisson solver for simulating Flat Plate Heat Transfer
- Conclusions and Future Work

Contents

- **Overview on AACE**
- Overview on MIC and its Configuration on the Cray CX1
- Applications code, description, porting and results
    - Euler solver and Boltzmann-BGK solver,
    - Navier-Stokes solver (2D and 3D)
    - Poisson solver for simulating Flat Plate Heat Transfer
- Conclusions and Future Work

Application Acceleration Center of Excellence (AACE)

Joint Institute for Computational Sciences

University of Tennessee & ORNL

- Established early in 2011 to investigate the application of future computing technologies to simulation in science and engineering
- *An essential element of a sustainable software infrastructure for scientific computing*
- Director: Glenn Brook

AACE — Mission

- To prepare the national supercomputing community to effectively and efficiently utilize future supercomputing architectures
  - To optimize applications for current and future compute systems
  - To develop expertise in the expression and exploitation of fine-grain and medium-grain parallelism
  - To conduct research and education programs focused on developing and transferring knowledge related to emerging computing technologies
  - To provide expert feedback to HPC vendors to guide the development of future supercomputing architectures and programming models

NICS-Intel Strategic Engagement

- Multi-year agreement with Intel to jointly pursue:
  - Development of next-generation, HPC solutions based on the Intel Many Integrated Core (MIC) architecture
  - Design of scientific applications emphasizing a sustainable approach for both performance and productivity
- NICS receives early access to Intel technologies and provides application testing, performance results, and expert feedback
  - Help guide further development efforts by Intel
  - Help prepare the scientific community to use future HPC technologies immediately upon their deployment
  - Co-design for Scientific Computing

Hardware Resources

- Rook — Intel MIC “Knights Ferry” SDP
  - Workstation – 2 Westmere CPUs & 2 KNFs
- Bishop — Cray CX1 cluster
  - 1 Head node – 2 Westmere CPUs
  - 2 Compute nodes – 2 Westmere CPUs & 1 KNF
- Beacon – Appro cluster
  - 2 Service nodes – 2 Sandybridge CPUs
  - 16 Compute nodes – 2 Sandybridge CPUs & 2 KNFs

Summary of Accomplishments

- Migration of important libraries to the Intel MIC
  - mpich 1.2.7p1
  - HDF5 1.8.5
  - HYPRE 2.6.0b (with BLAS and LAPACK)
- Ported millions of lines of code to the Intel MIC architecture in weeks
  - Full applications from a variety of scientific and engineering disciplines — MPI and/or OpenMP
- Implemented MIC-to-MIC communications both on node and off node
  - Demonstrated MPI and Hybrid-MPI/OpenMP communications within a single MIC, between two MICs on a node, and across multiple MICs on multiple nodes

Future Plans

- Partner with NSF research teams to port and optimize key NSF research codes
- Offer training to the scientific computing community on the Intel MIC architecture following its commercial debut
- Publish accumulated knowledge in scientific papers, conference presentations, engineering reports, and training materials
- Expand student exposure to next-generation Intel technologies through internships at NICS and on-going participation in AACE activities

Contents

- Overview on AACE
- **Overview on MIC and its Configuration on the Cray CX1**
- Applications code, description, porting and results
  - Euler solver and Boltzmann-BGK solver,
  - Navier-Stokes solver (2D and 3D)
  - Poisson solver for simulating Flat Plate Heat Transfer
- Conclusions and Future Work

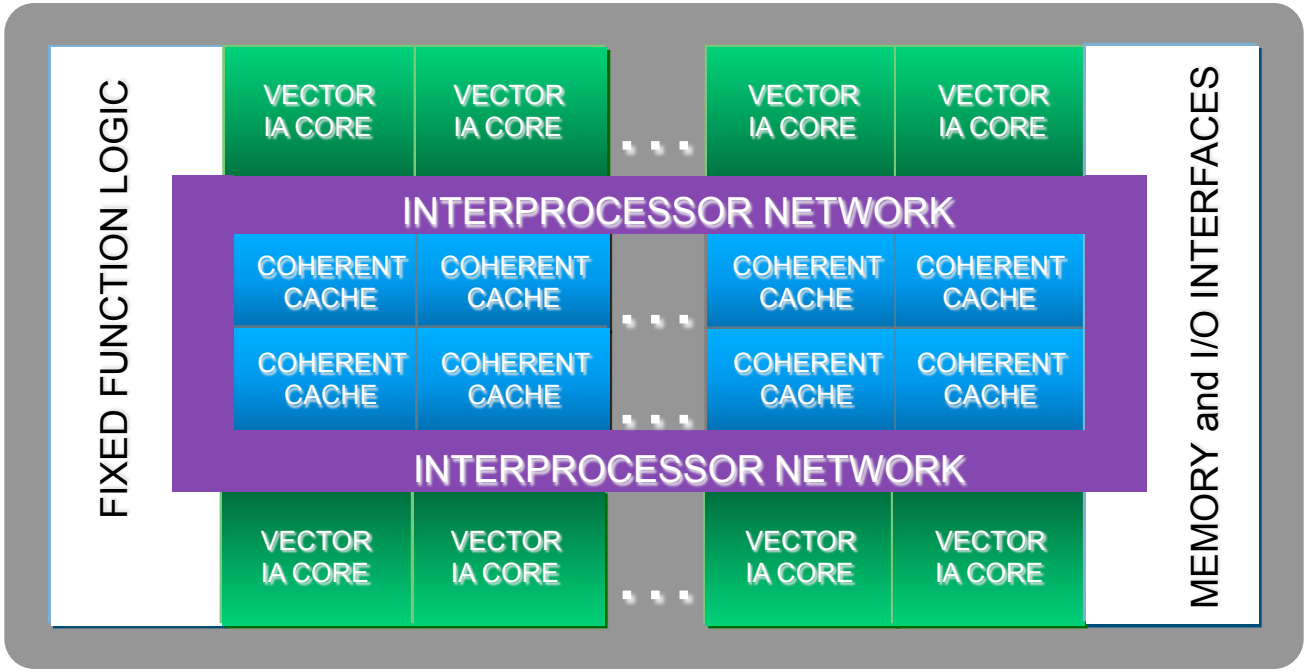
Intel Knights Ferry Technical Specifications

Intel Knights Ferry (Intel KNF) is the software development platform (SDP) for the Intel Many Integrated Core (Intel MIC) architecture.

Core Count	Up to 32 cores
Core Speed	Up to 1.2 GHz
IO Bus	PCIe Gen2 x16
Memory Type	GDDR5
Memory Size	1 or 2 Gigabytes
Peak Flops (Single Precision/ Double Precision)	1229/153 GFLOPS
Operating System on Card	Linux-based
Networking Capability	IP-Addressable

Image Source: Kirk Skaugen, ISC 2010 keynote

Intel MIC Architecture: An Intel Co-Processor Architecture



- Many cores, and many, many more threads
- Standard IA programming and memory model
- Standard networking protocols

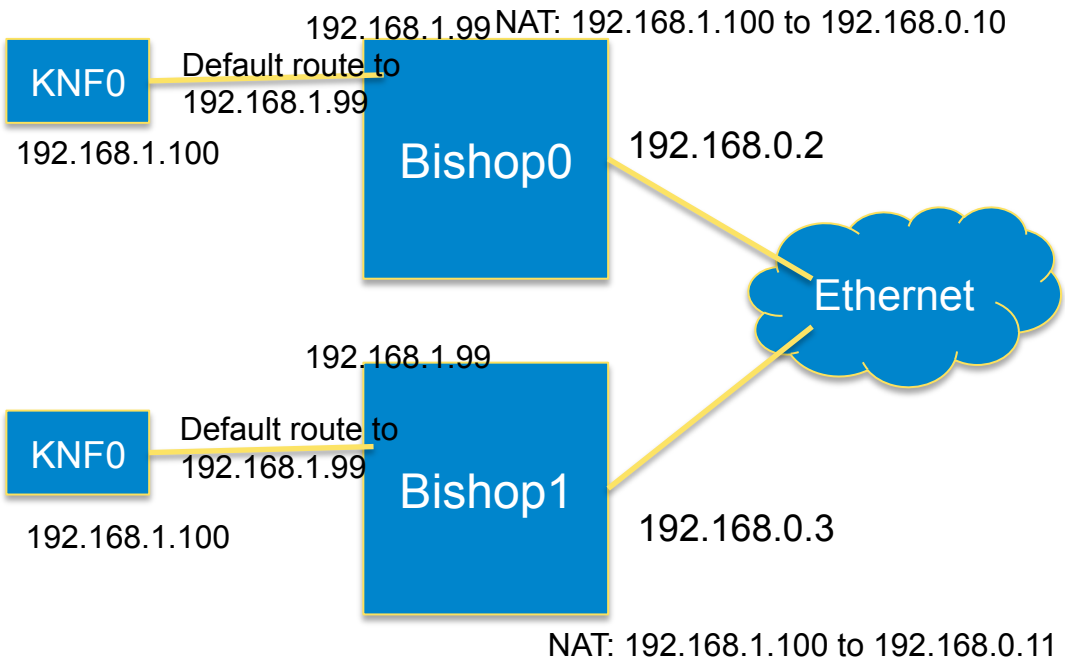
Source: Kirk Skaugen, ISC 2010 keynote

Bishop — Cray CX1

- 1st Intel MIC cluster at NICS
- Interactive Intel MIC demo in the ORNL booth at SC11

Vendor	Cray
Configuration	7u modular enclosure
Nodes	1 head, 2 compute
CPU model	Intel Xeon 5670 — “Westmere”
CPUs per node	2
Cores per CPU	6
CPU core speed	2.93 GHz
RAM per node	24 GB
Intel KNFs per compute node	1
Cores per Intel KNF	32
Intel KNF core speed	1.2 GHz
RAM per Intel KNF	2 GB

Network Address Translation and Intel Knights Ferry Alpha SW



Running MPI applications in native mode across multiple Intel KNF cards requires a method of communication between the cards in the different compute nodes.
→setting up TCP/IP communications between the cards.

Network Address Translation (NAT) is used to make it appear that the Intel KNF cards are on the same Ethernet network as their host nodes.

Each Intel KNF card requires the following three NAT rules:
1) -A PREROUTING -d [virtual address for KNF/mask] –j DNAT –to-destination 192.168.1.100
2) -A POSTROUTING -s 192.168.1.100/32 -j SNAT –to source [virtual address for MIC]
3) -A OUTPUT -d [virtual address for MIC/mask] -j DNAT–to-destination 192.168.1.100

Contents

- Overview on AACE
- Overview on MIC and its Configuration on the Cray CX1
- **Applications code, description, porting and results**
  - **Euler solver and Boltzmann-BGK solver,**
  - **Navier-Stokes solver (2D and 3D)**
  - **Poisson solver for simulating Flat Plate Heat Transfer**
- Conclusions and Future Work

A UT/ORNL PARTNERSHIP
NATIONAL INSTITUTE FOR COMPUTATIONAL SCIENCES
NICS

Solving the Euler Equations and the BGK Model Boltzmann Equation Using OpenMP In Native Mode on the Intel® MIC

Ryan Hulguin

Comparison of the CFD solvers

- **Two separate computational fluid dynamics (CFD) solvers are developed to showcase the capability of the Intel® MIC**
- **The first solver is based on the Euler equations**
- **The second solver is based on the Boltzmann equation**
- **Both solvers are developed using a Newton based iterative algorithm to converge the solutions**
- **Data parallelism on the Intel® MIC is achieved through the use of OpenMP threads.**

	Euler Solver	Boltzmann Solver
Number of equations per physical grid point	5	Hundreds of thousands
Target applications	Inviscid fluid flow	Rarefied gas flow

Euler Equations

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{F}}{\partial x} + \frac{\partial \vec{G}}{\partial y} + \frac{\partial \vec{H}}{\partial z} = \vec{0}$$

$$\vec{Q} = [\rho, \rho u, \rho v, \rho w, E]^T$$

$$\vec{F} = [\rho u, \rho u^2 + P, \rho uv, \rho uw, (E + P)u]^T$$

$$\vec{G} = [\rho v, \rho uv, \rho v^2 + P, \rho vw, (E + P)v]^T$$

$$\vec{H} = [\rho w, \rho uw, \rho vw, \rho w^2 + P, (E + P)w]^T$$

ρ is the density

$\vec{u} = (u, v, w)$ are the velocities

P is the pressure

E is the total energy per unit volume

BGK Model Boltzmann Equation

$$\frac{\partial f_k}{\partial t} + \vec{V}_k \cdot \frac{\partial f_k}{\partial \vec{x}} = \nu \cdot \left(f_k^{eq} - f_k\right)$$

$$f^{eq} = \frac{n}{(\pi T)^{3/2}} \left(- \frac{\left(\vec{V}_k - \vec{u}\right)^2}{T} \right)$$

$$\upsilon = \frac{8nT^{(1-\omega)}}{5\sqrt{\pi}\mathrm{Kn}}$$

f_k is the probability distribution function at discrete velocity k

$\vec{V}_k = \left(V_{x_k}, V_{y_k}, V_{z_k}\right)$ are the discrete molecular velocities

$\vec{u} = (u, v, w)$ are the bulk (average) velocities

n is the number density

T is the temperature

ω is the gas viscosity index

Kn is the Knudsen number which characterizes the flow

Kn is the ratio between

the mean free path λ_∞ and the characteristic length L

Numerical Algorithm

Given a nonlinear set of equations, $\vec{F}(\vec{q}) = \vec{0}$,
Application of Newton's method results in

$$\vec{q}^{\,p} = \vec{q}^{\,p-1} - J(\vec{q}^{\,p-1})^{-1} F(\vec{q}^{\,p-1})$$

where p is the Newton iteration and J is the jacobian as defined below

$$J_{kj} = \frac{\partial F_k}{\partial q_j} : 1 < k, j < n$$

Rearranging terms leads to

$$J\Delta\vec{q}^{\,p} = -\vec{F}$$

where $\Delta\vec{q}^{\,p} = \vec{q}^{\,p} - \vec{q}^{\,p-1}$

Using the Jacobi method to solve the linearized system of equations gives

$$(\Delta q^p)_k^m = \frac{1}{J_{kk}}\left[-F_k - \sum_{j \neq k} J_{kj}(\Delta q^p)_j^{m-1}\right]$$

where m is the current Jacobi iteration

More details can be found in the dissertation of Glenn Brook (glenn-brook@tennessee.edu)
Brook, R. Glenn, "A Parallel, Matrix-Free Newton Method for Solving Approximate Boltzmann Equations on Unstructured Topologies," PhD Dissertation, University of Tennessee at Chattanooga, December 2008.

Implicit Jacobian Calulations

The Jacobi iterative update equation can be cast into a delta formulation as shown below

$$(\Delta q^p)_k^m - (\Delta q^p)_k^{m-1} = \frac{1}{J_{kk}}\left[-F_k - \sum_{j \neq k} J_{kj}(\Delta q^p)_j^{m-1} - J_{kk}(\Delta q^p)_k^{m-1}\right]$$

$$\Delta[(\Delta q^p)_k^m] = \frac{1}{J_{kk}}[-F_k - J(\Delta q^p)^{m-1}]$$

The jacobian does not need to be explicitly calculated and stored

It can be calculated implicitly through the use of dual numbers and Taylor series expansions in the dual space.

$$J_k(\Delta q^p)^{m-1} \approx \frac{1}{h}\text{Dual}[F_k(q^{p-1} + \epsilon h(\Delta q^p)^{m-1})]$$

$$F_k \approx \text{Real}[F_k(q^{p-1} + \epsilon h(\Delta q^p)^{m-1})]$$

$$J_{kk} \approx \frac{1}{h}\text{Dual}[F_k(q^{p-1} + \epsilon h e_k)]$$

where $\epsilon^2 \equiv 0, h \in \mathbb{R}$ is an arbitrary perturbation and e^k is kth vector in the standard basis for $\mathbb{R}^N$.

Test Problems Run on the Intel® MIC

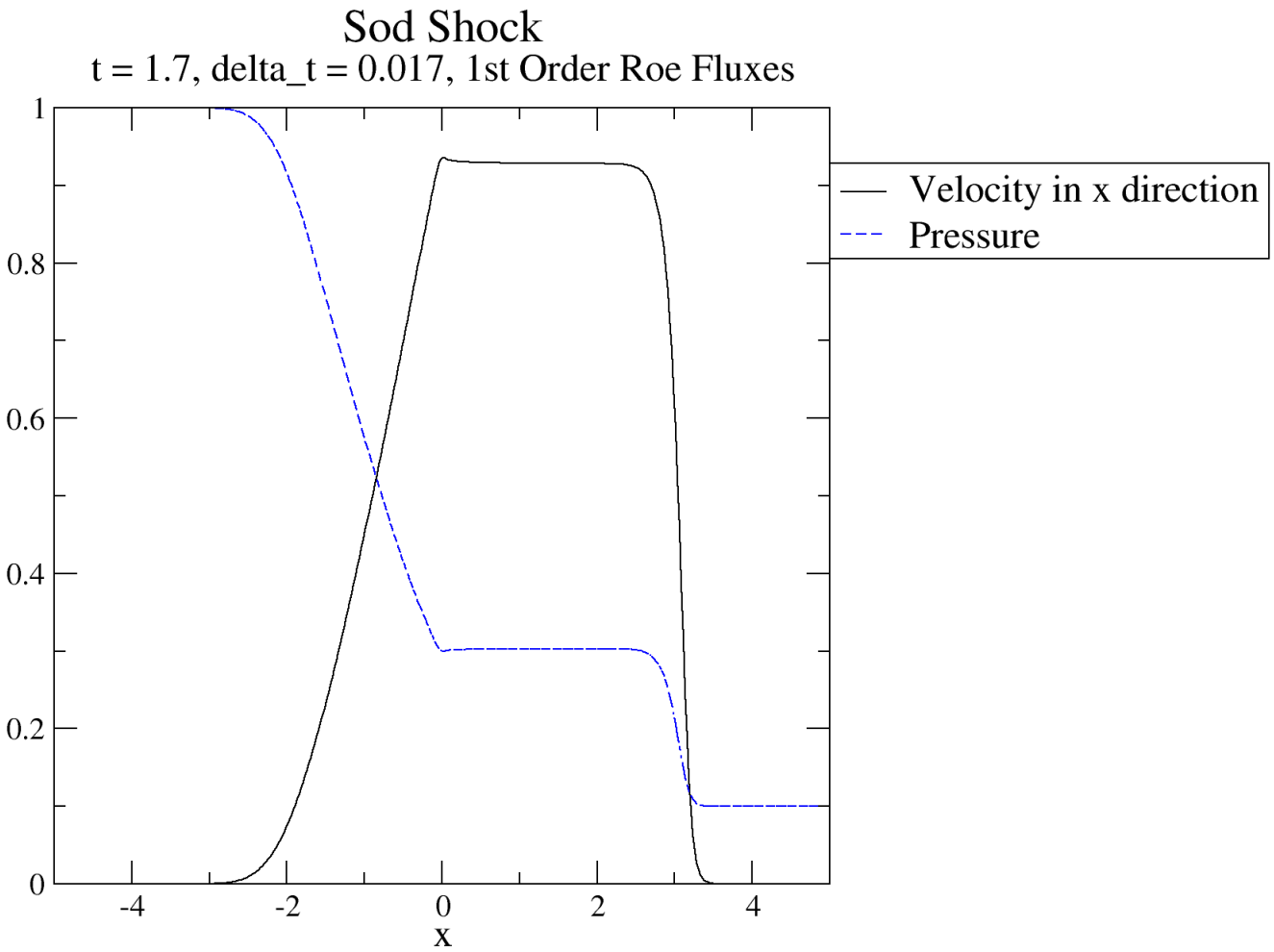
Sod Shock
Initial conditions:

$\rho = 1.0$	$\rho = 0.125$
$u = 0.0$	$u = 0.0$
$P = 1.0$	$P = 0.1$

Unsteady flow problem using the Euler equations

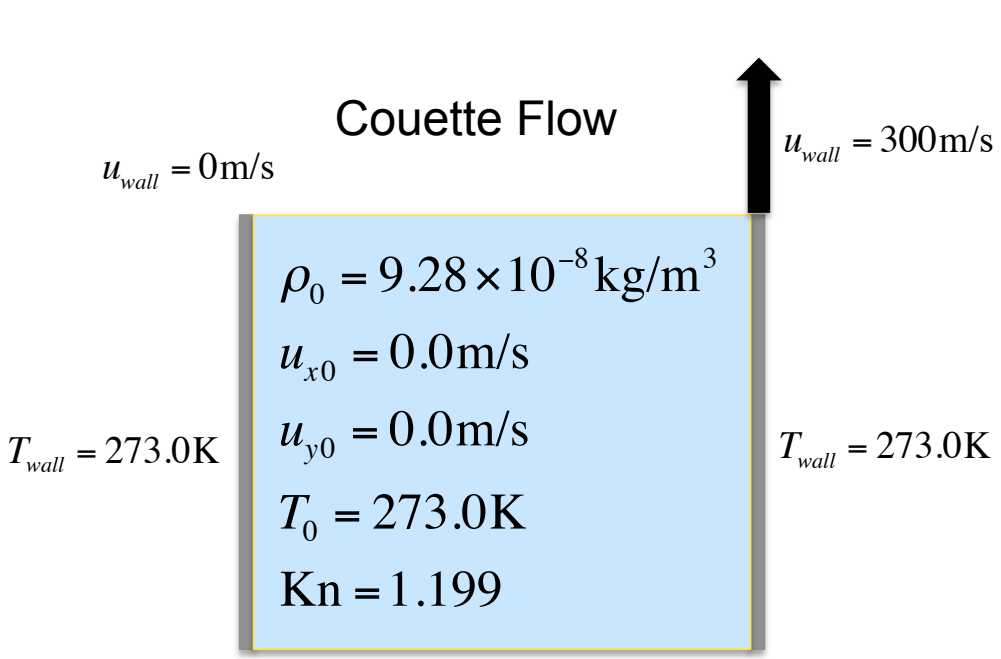
- **The Sod shock was run with the Euler solver**
- **This problem showcases the solver's ability to capture discontinuities**

Euler Solver Solution



Test Problems Run on the Intel® MIC

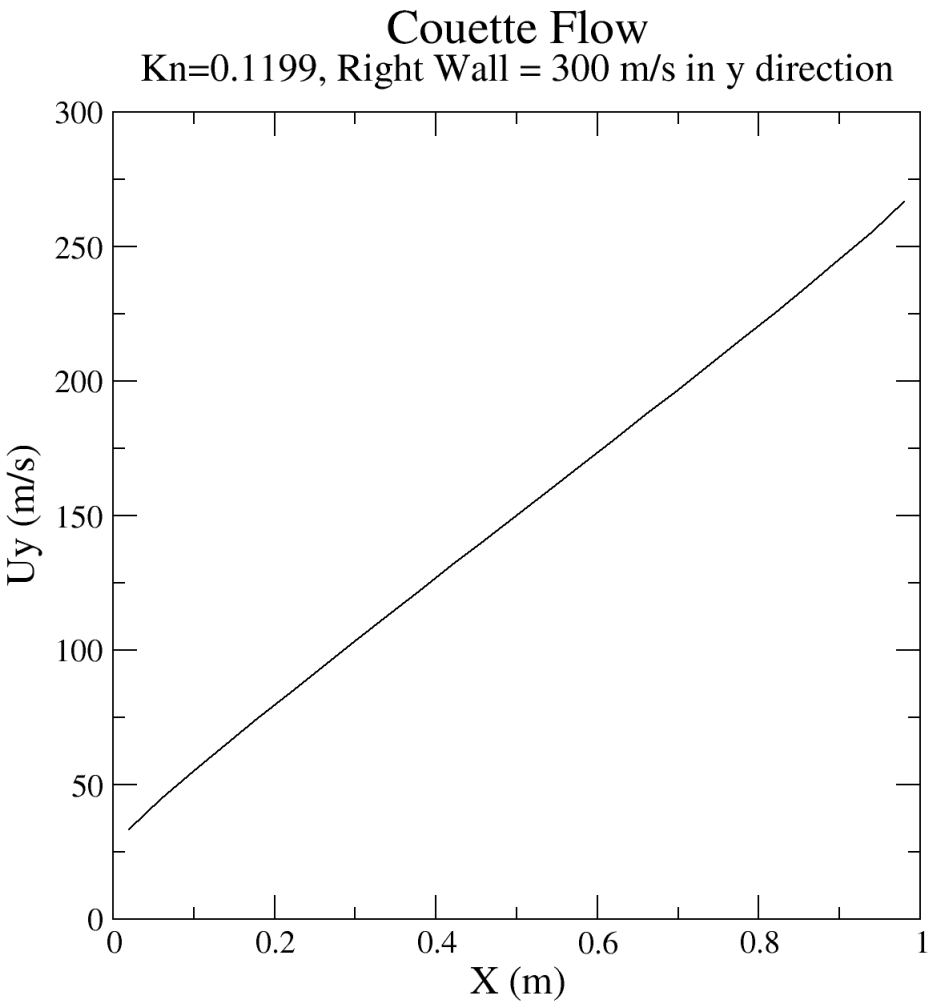
- **A Couette flow was run using the BGK model Boltzmann solver**
- **This problem showcases the solver's ability to handle solid surfaces and moving boundaries**



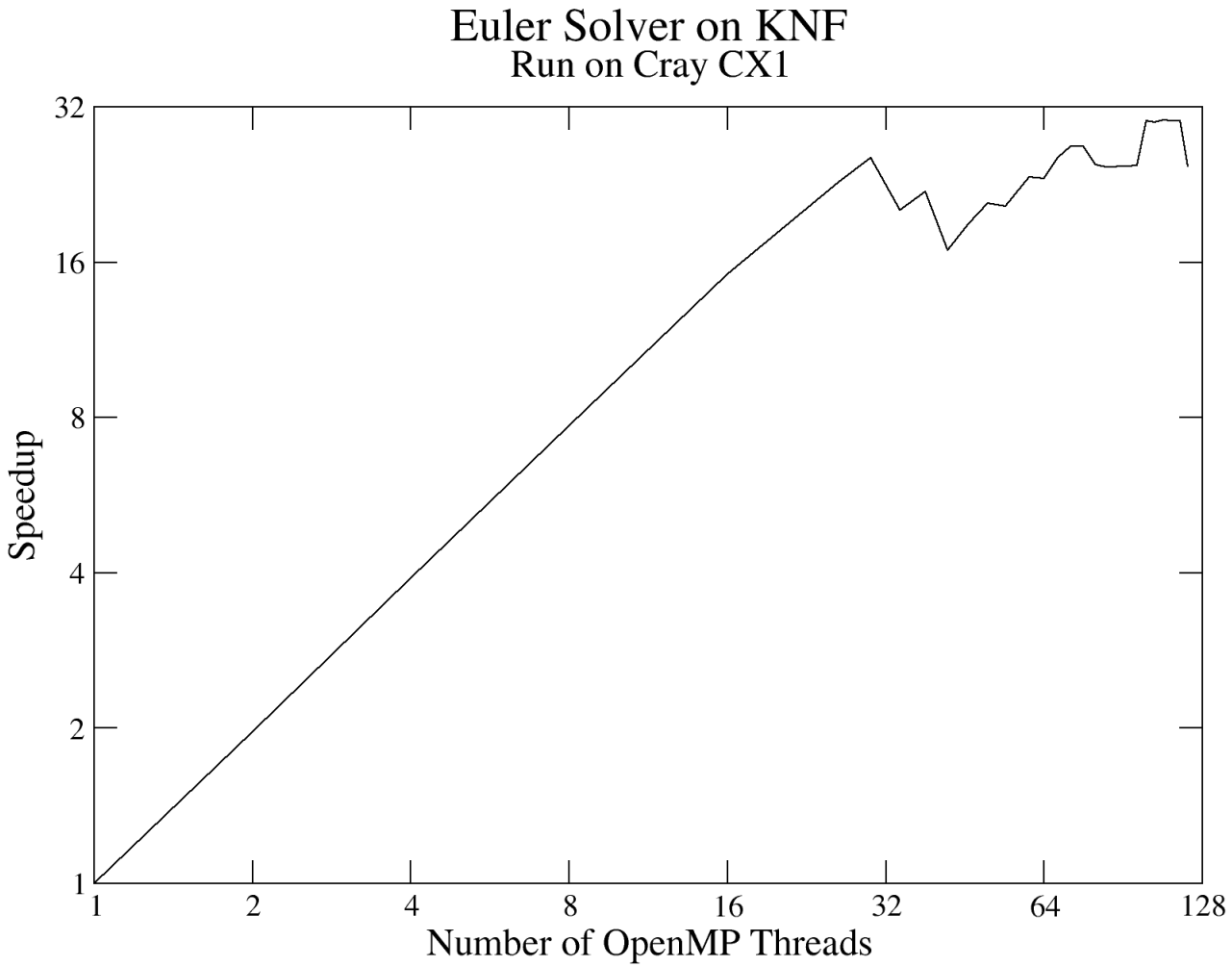
Steady state flow problem using the BGK model Boltzmann equation

This particular test problem used 27 grid cells in physical space and 36x36x36 grid points in velocity space

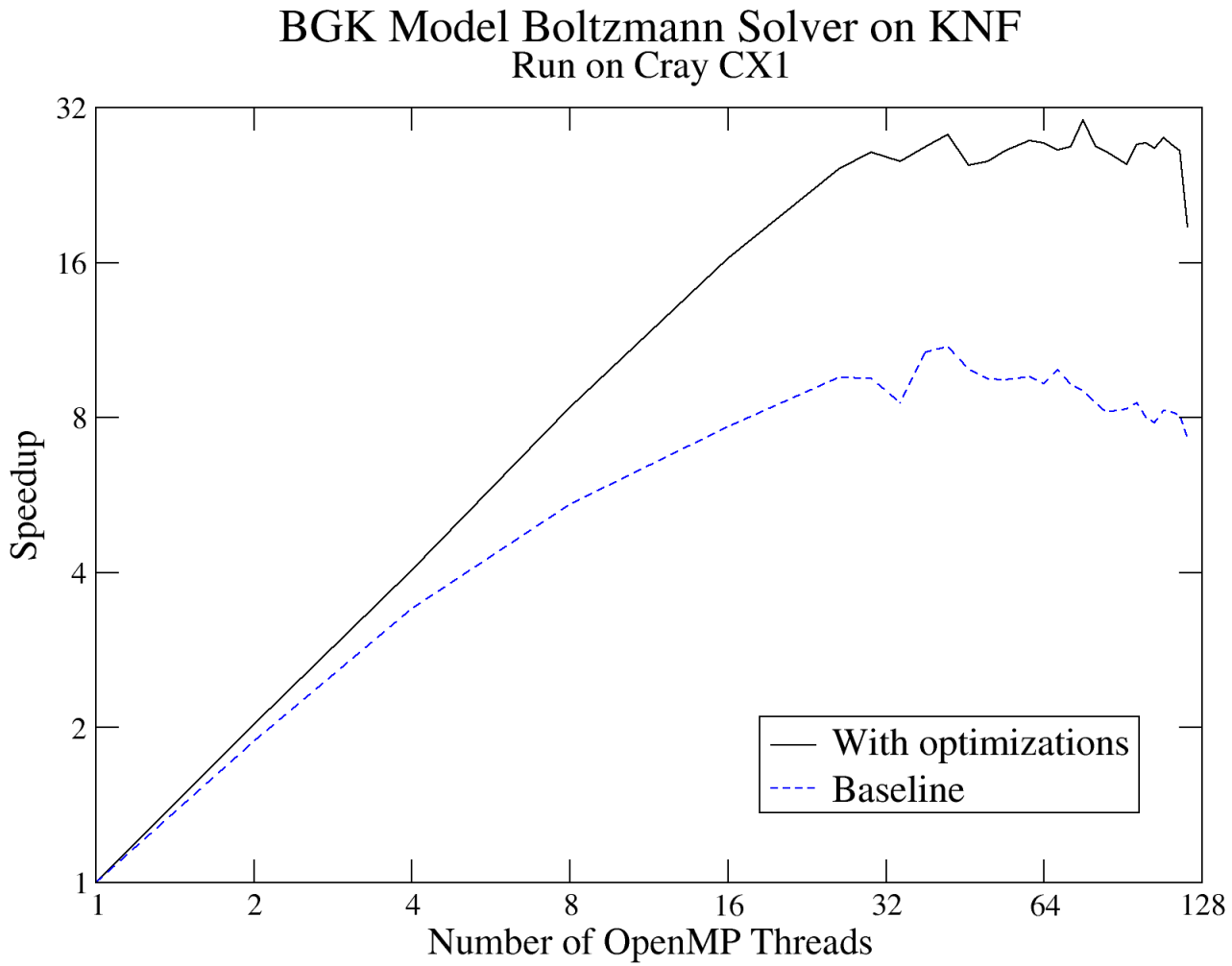
BGK Model Boltzmann Solver Solution



Intel® MIC Performance



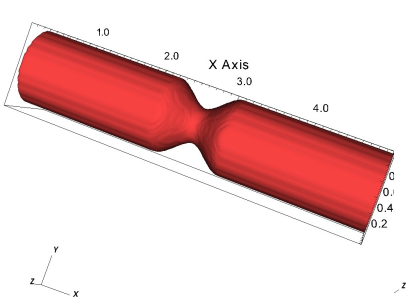
Intel® MIC Performance

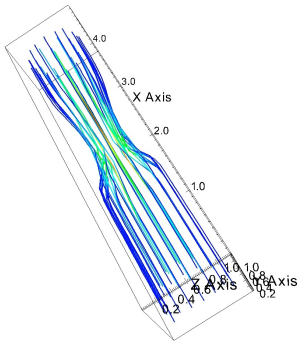


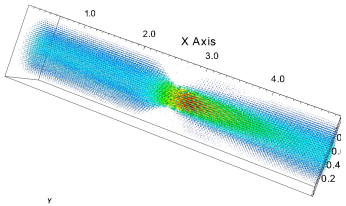
Note: Optimizations were provided by Intel senior software engineer Rob Van der Wjingaart

A UT/ORNL PARTNERSHIP
NATIONAL INSTITUTE FOR COMPUTATIONAL SCIENCES
NICS

Navier Stokes Simulations on the MIC







Bilel Hadri

Methodology

- **Fact** : The most consuming part of a code is in general the resolution of linear systems.

- **Focus**: Fast elliptic solver For Incompressible Navier-Stokes Flow code

- **context**: - Finite Volume
  - mesh topologically equivalent to Cartesian mesh,
  - distributed computing with high latency network.

- **Method** : - L_2 penalty method for a fast prototyping to the NS flow
  - Level set method
  - efficient subdomain solver
  - Aikten Schwarz is a domain decomposition technique designed for distributed computing with slow network.

Navier Stokes Formulation

Incompressible NS flow in large Vessels

$$\partial_t U + (U \cdot \nabla) U + \nabla p - \nu \nabla \cdot (\nabla U) = -\frac{1}{\eta} \Lambda_{\Omega_\omega} (U - U_w(t))$$

$$div(U) = 0$$

- Ω_w : solid wall
- L^2 Penalty method $\eta << 1$. ref(Caltagirone 84, Bruneau et al 99, Schneider et al 2005).
- Λ is a mask function provided by by a level set method used in the image segmentation of the blood vessel

Navier Stokes Numerical Approximation

Time Step : Projection Method (Chorin)

Step 1: Prediction of the velocity $\hat{u}^{k+1}$ by solving explicitely

$$\frac{\hat{u}^{k+1} - u^{k,*}}{\Delta t} - \nu \Delta u^{k} = f^{k+1} - \nabla p^{k} \text{ in } \Omega$$

$$\text{Boundary conditions}$$

$$\hat{u}^{k+1} = g \text{ on } \partial\Omega$$

Step 2: Projection of the predicted velocity to the space of divergence free functions

$$-div \nabla \delta p = -\frac{1}{\Delta t} div(\hat{u}^{k+1})$$

$$u^{k+1} = u^{k+1} - \Delta t \nabla \delta p$$

$$p^{k+1} = p^{k} + \delta p$$

Momentum equation

$$-\Delta t \ \nu \ \Delta U - cU = RHS_1$$

Pressure equation

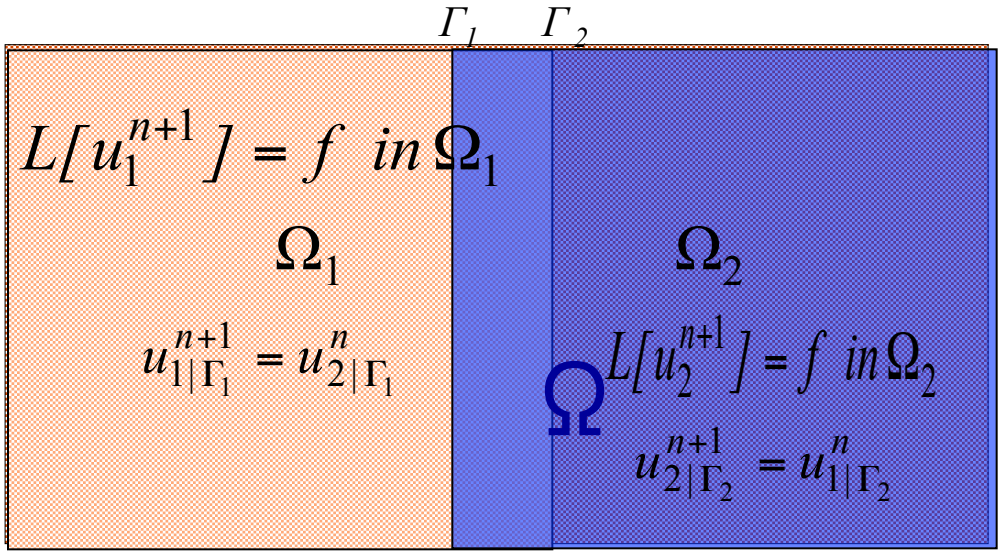
$$\Delta P = RHS_2$$

→ *Focus: : design of the optimum solver*

Additive Schwarz

- **Additive Schwarz method :**

$$L[u] = f \ in \Omega$$



Additive Schwarz algorithm :

1/ Solve in each domain

2/ Update the solution in the interface

3/ Repeat until convergence

- **If the approximation of the operator follows the maximum principal, then Additive Schwarz is converging and is a robust solver, however the convergence is slow !**

- **Aikten like Acceleration method allows to accelerate this convergence. Ref (Garbey-Tromeur Dervout 2002)**

Aitken Schwarz Algorithm

M. Garbey and D. Tromeur Dervout: "On some Aitken like acceleration of the Schwarz Method," International Journal for Numerical Methods in Fluids. Vol. 40(12),pp 1493-1513, 2002.

Aitken Schwarz is a domain decomposition method using the framework of Additive Schwarz and based on an approximate reconstruction of the dominant eigenvectors of the trace transfer operator.

Thanks to the IBM(Immersed Boundary Method), regular Cartesian grid can be used.

→We get a direct solver since the eigenvectors are known analytically.

Algorithm :

Step1:

-apply additive Schwarz with a subdomain solver

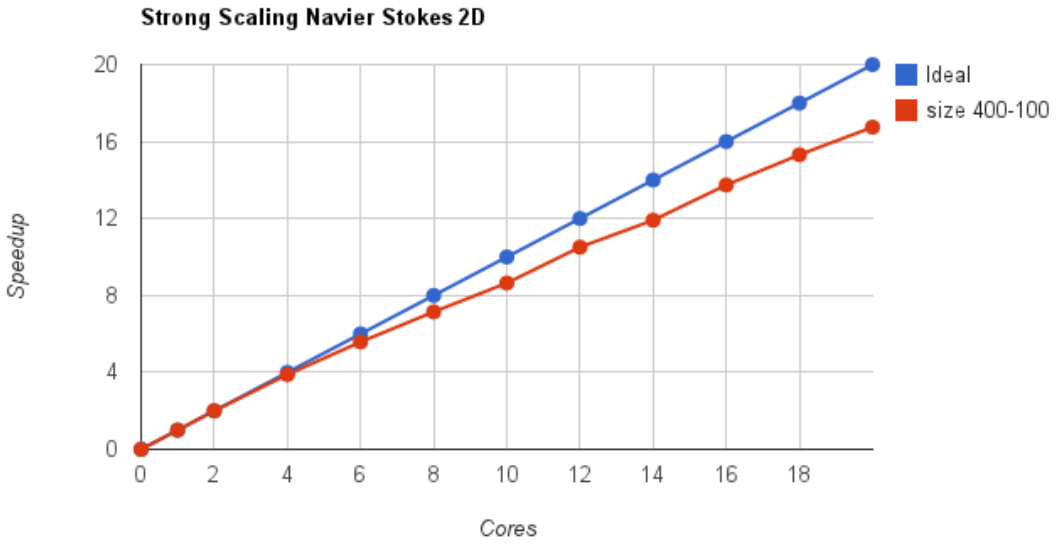
Step 2:

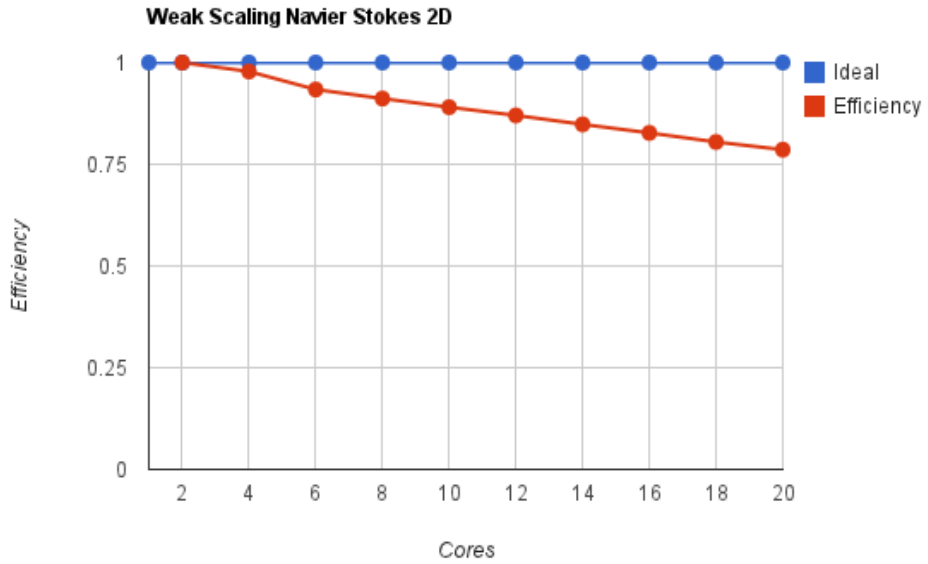
- compute the sine (or cosine) expansion of the traces on the artificial interface for the initial boundary condition u^0_Γ and the solution given by on Schwarz iterative u^1_Γ
- apply generalized Aitken acceleration to get u^∞_Γ
- recompose the trace in physical space

Step 3 :

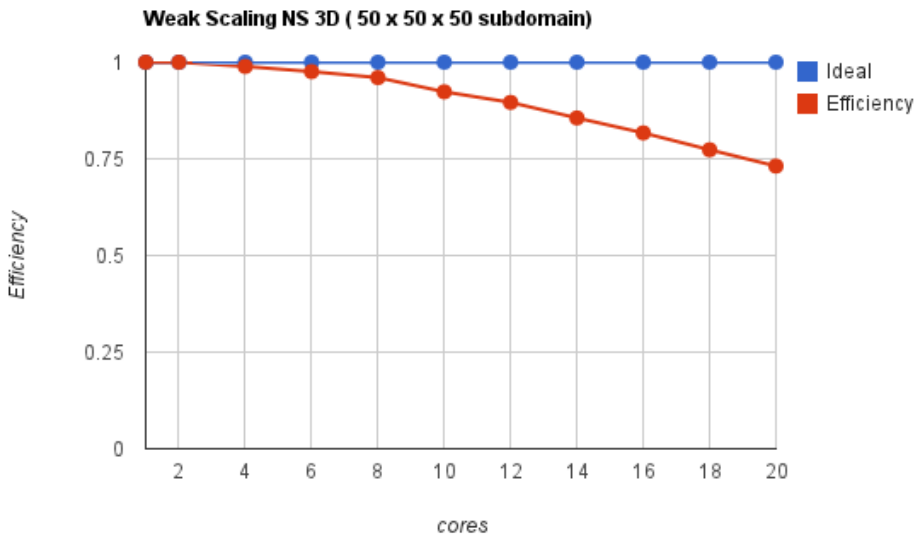
-Compute in parallel the solution in each subdomain, with the new inner BCs u^∞_Γ.

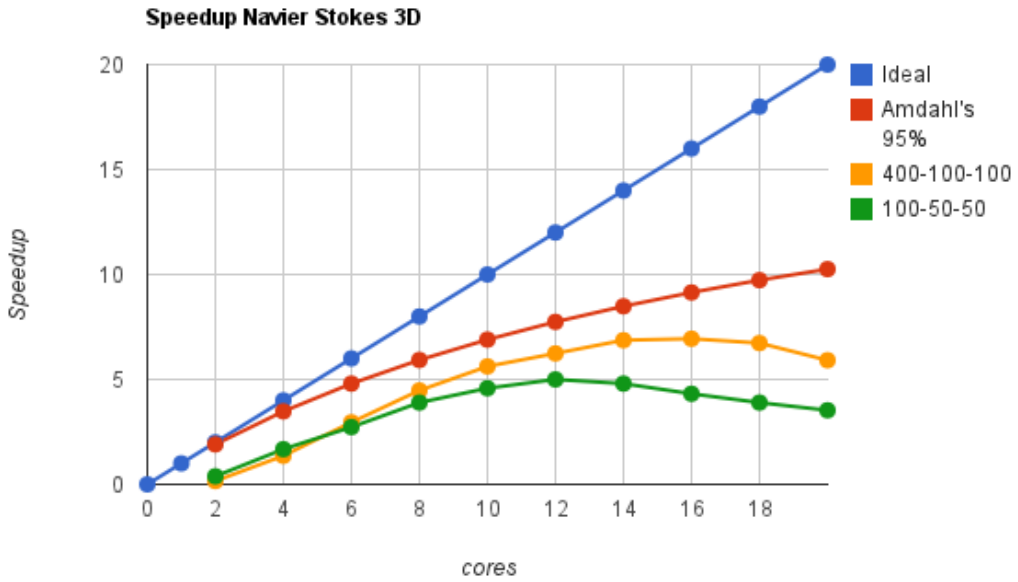
NS Benchmark 2D





NS 3D


Weak Scaling NS 3D (50 x 50 x 50 subdomain)
Ideal
Efficiency
Efficiency
1
0.75
0.5
0.25
0
2
4
6
8
10
12
14
16
18
20
cores


Speedup Navier Stokes 3D
Ideal
Amdahl's 95%
400-100-100
100-50-50
Speedup
20
15
10
5
0
0
2
4
6
8
10
12
14
16
18
cores

A UT/ORNL PARTNERSHIP
NATIONAL INSTITUTE FOR COMPUTATIONAL SCIENCES
NICS

Using a Hybrid OpenMP/MPI Poisson Solver for simulating Flat Plate Heat Transfer: Intel MIC Results and Speedup

Vincent C. Betro, Ph.D.

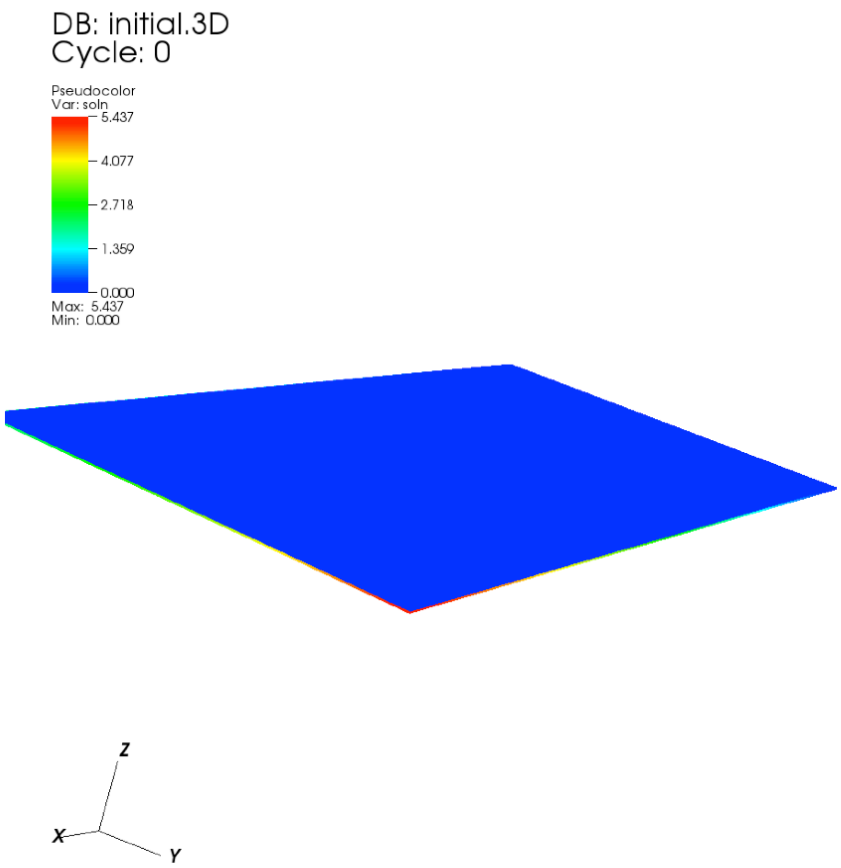
Case information

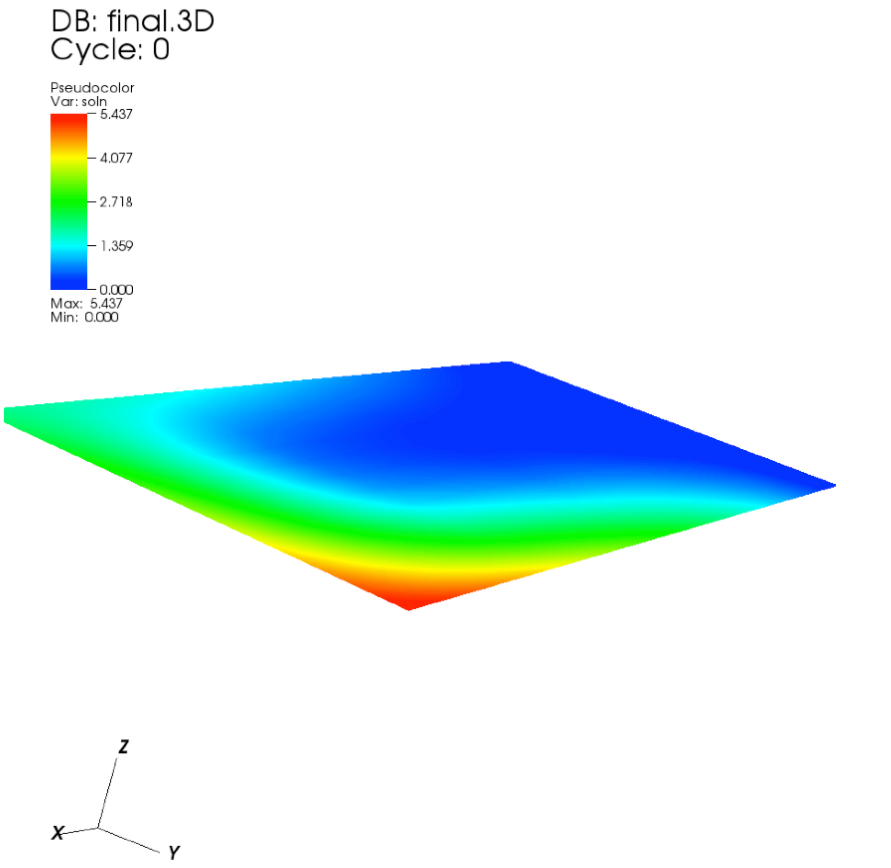
- Heat transfer over a flat plate to a cool boundary
- Uses Poisson's Equation ($\nabla^2\Phi = f$)
- Source term is xe^y
- Size: $x:0 \rightarrow 2$, $y:0 \rightarrow 1$
- Three structured meshes
  - 600x600, 1200x1200, 2400x2400
- Symmetric Gauss-Seidel with SSOR*
- Decomposed to 1, 15, 30, 60, 90, 120 threads offloaded from one processor

*In non-blocking parallel mode, there are some Jacobi iterations

Results

Initial and Final plots of Temperature Contours (using VisIt) in °C

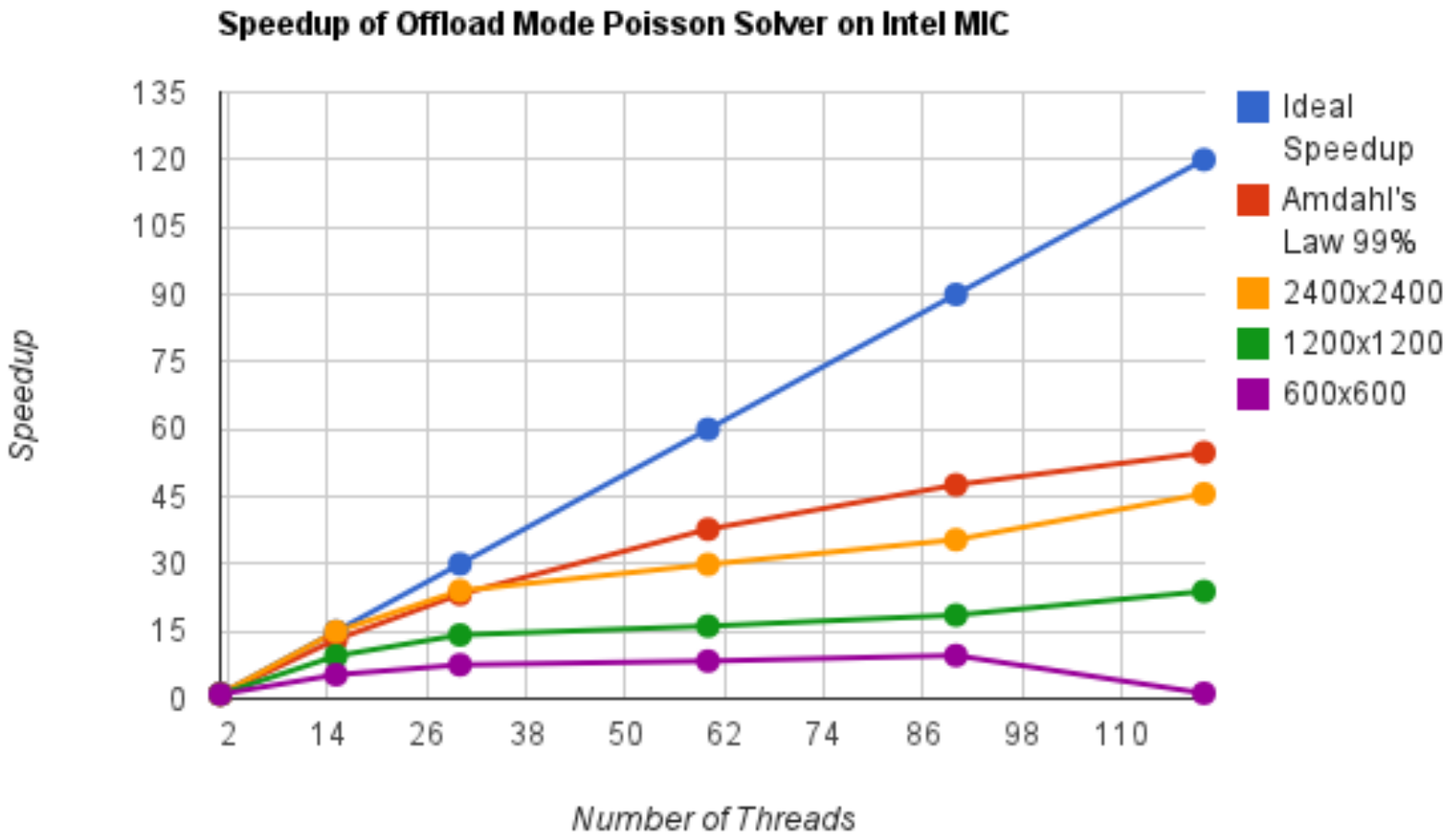




Offload Mode Implementation

- Offload mode is a very similar operation to the current paradigm offered by GPGPUs.
- The serial portion of the code is run on the CPU, and portions of the code are either explicitly or implicitly (we use implicit) offloaded to the MIC card.
- #pragma offload_attribute(push, _Shared)
- #pragma offload_attribute(pop)

Speedup


Speedup of Offload Mode Poisson Solver on Intel MIC
Speedup
135
120
105
90
75
60
45
30
15
0
2
14
26
38
50
62
74
86
98
110
Number of Threads
Ideal Speedup
Amdahl's Law 99%
2400x2400
1200x1200
600x600

Discussion and Conclusions

- Larger matrices see better speed up
- Offload time amortized into benefit of multiple threads
- Eventually, will use mixed offload and native mode to map to Intel MIC topology

Contents

- Overview on AACE
- Overview on MIC and its Configuration on the Cray CX1
- Applications code, description, porting and results
    - Euler solver and Boltzmann-BGK solver,
    - Navier-Stokes solver (2D and 3D)
    - Poisson solver for simulating Flat Plate Heat Transfer
- **Conclusions and Future Work**

Conclusions and Future Work

- The Intel MIC architecture offers unique advantages that make it an appealing choice for sustainable software development for scientific computing:
  - Offers a standards-based programming model that maintains source code compatibility with existing Xeon (and other x86-based) architectures.
  - MIC is expected to employ a unified development environment (compiler, debugger, and performance tools) that is applicable to all Intel architectures, and the unified compiler is expected to produce binaries that run appropriately on any Intel architecture.
- Intel MIC offers easily accessible scalability across its many cores, as indicated by the scaling results presented for the initial porting efforts of the applications studied in this work.
- Codes exhibiting sufficient medium-grain parallelism scale effectively across all of the cores on the Intel MIC.
- Significant sustained performance is expected for optimized codes on Knight Corner.

→ Intel MIC is expected to be a highly productive platform for scientific computing

Contact

R. Glenn Brook, Ph.D.
Director, Application Acceleration Center of Excellence
Joint Institute for Computational Sciences
University of Tennessee / Oak Ridge National Laboratory
glenn-brook@tennessee.edu

THANKS !