Optimising Hydrodynamics Applications for the Cray XC30 with the Application tool Suite

Wayne Gaudin, Andy Herdman, Olly Perks – AWE
Andy Mallinson, Stephen Jarvis – University of Warwick
Simon McIntosh-Smith, Michael Boulton – University of Bristol
John Levesque - Cray

Abstract

Power constraints are forcing HPC systems to continue to increase hardware concurrency. Efficiently scaling applications on future machines will be essential for improved science and it is recognised that the “flat" MPI model will start to reach its scalability limits. The optimal approach is unknown, necessitating the use of mini-applications to rapidly evaluate new approaches. Reducing MPI task count through the use of shared memory programming models will likely be essential. We examine different strategies for improving the strong-scaling performance of explicit Hydrodynamics applications. Using the CloverLeaf mini-app across multiple generations of Cray platforms (XC30, XK6 and XK7), we show the utility of the hybrid approach and document our experiences with OpenMP, CUDA, OpenCL and OpenACC under both the PGI and CCE compilers. We also evaluate Cray Reveal as a tool for automatically hybridising HPC applications and Cray's MPI rank to network topology-mapping tools for improving application performance.

Agenda

- Background and motivation
- OpenACC performance evaluation
- XK6, XK7 and XC30 comparison
  - Hybrid programming models
- Cray tool-chain evaluation
  - Reveal
  - Rank Reordering
- Conclusions
- Future work

Background & Motivation

- Exascale challenge
  - We need to maintain and develop production (legacy) codes
  - In an ever evolving environment
    - New hardware – 3 levels of parallelism
    - New programming models – too many choices
  - That maintains portable performance
  - While maximising scientific productivity
  - Looking for a suitable level of abstraction and tools to achieve this
    - At the core and node level
    - At the interconnect level
  - We adopt a Co-design approach
    - Internally using multi-disciplinary teams
    - Externally with peers and academia
    - And industry - working with Cray/Nvidia/PGI/Intel in this instance

Background & Motivation

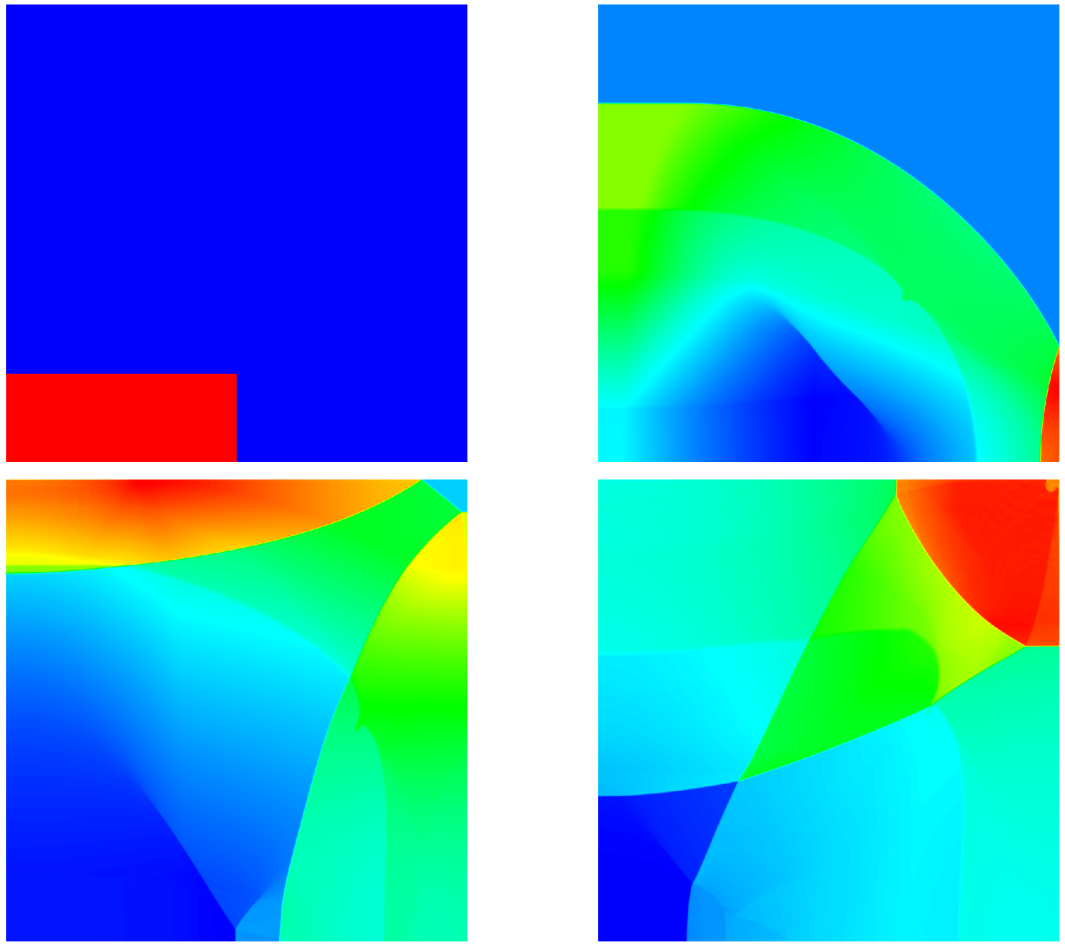
- Assess node level parallel models
  - OpenMP
  - OpenACC
  - OpenCL
  - CUDA
- Improve the strong-scaling performance of CloverLeaf (and other applications)
- Evaluate hybrid programming models at scale
- Interested in evaluating new tools which can assist with software development and maintenance
- Two Cray tools examined as part of this work:
  - Reveal
  - Grid_order

CloverLeaf

- Mini-app to solve Euler's equations on a structured grid
- Our code base is FORTRAN
- Uses an explicit finite volume method
  - In 2D
  - Using stencil operations
  - On a staggered grid
- Domain decomposed
  - Local halo exchange using MPI
- Refactored to be
  - Data parallel
  - Compute optimised
  - Memory access optimised
- But remains memory bound – typical of our applications

CloverLeaf Test Problem

- Plot of density
- Propagating shock wave
- Reflections leads to interacting shocks



Programming Models

- MPI
  - Minimises communication surface area
  - No explicit barriers in compute – only in WaitAll
  - And scalar reduction for timestep control
- OpenMP+MPI
- OpenACC+MPI
- CUDA+MPI
- OpenCL+MPI
- Plus others not used in this talk (CAF, SHMEM)

Target Hardware

- Chilean Pine:
  - 40 Node XK6
  - X2090
  - Gemini Interconnect
- Titan:
  - 18,000 Node XK7
  - K20X
  - Gemini Interconnect
- Swan:
  - 8 Node XC30
  - K20X
  - Aries Interconnect
- Archer:
  - 3008 Node XC30
  - No GPUs
  - Aries Interconnect

Performance Portability of OpenACC

- OpenACC is a relatively new standard
- CloverLeaf implementation evolved from OpenMP version
- We assess the CCE and PGI compilers
  - Across XK6 and XK7
  - On two problems sizes
- We compare KERNEL and PARALLEL models
  - That evolved from vendor specific directives
- Assess performance portability
  - against performance of native/low level models

OpenACC – does it meet our criteria?

- Portability
  - Single Source
  - Across many platforms
- Performance
  - Against native
- Productivity
  - FORTRAN – continue to develop physics
  - Simple pragmas
  - Expert knowledge of hardware not essential

OpenACC Results – Small Test 960x960

	CCE PARALLEL	CCE KERNEL	PGI PARALLEL	PGI KERNEL	CUDA	OpenCL
XK6	67.67	86.58	90.89	100.33	58.07	59.95
XC30	46.07	44.36	61.89	47.04	34.84	36.06

- CUDA is fastest is both platforms
- OpenCL is close
- CCE is fastest OpenACC
- PGI much closer on XC30

OpenACC Results – Large Test 3840x3840

	CCE PARALLEL	CCE KERNEL	PGI PARALLEL	PGI KERNEL	CUDA	OpenCL
XK6	31.69	32.23	35.12	39.17	24.05	26.59
XC30	18.22	19.54	21.46	18.95	13.71	14.97

- CUDA is fastest is both platforms
- OpenCL is close
- CCE is fastest OpenACC
- PGI much closer on XC30 and better for larger data set

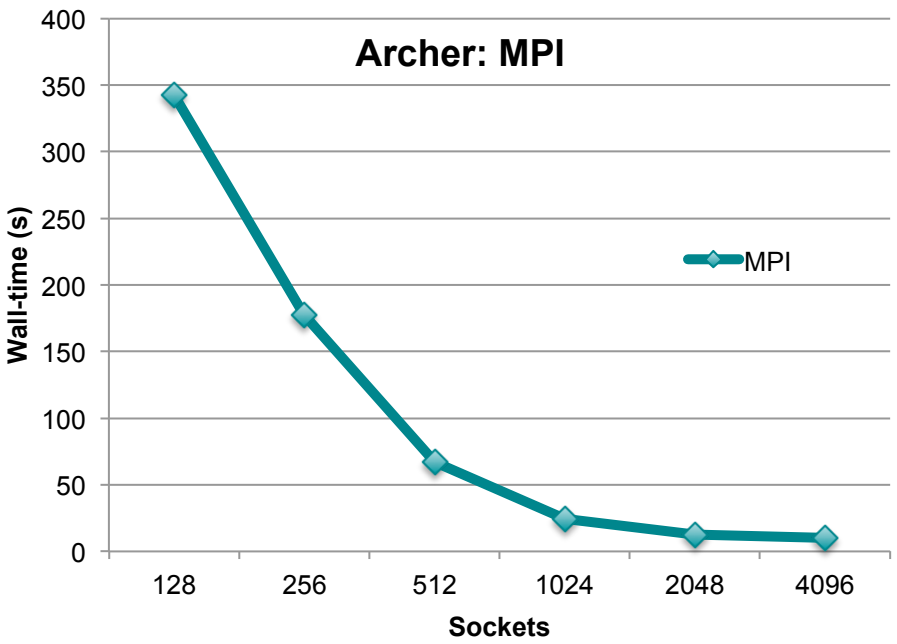
Take away points

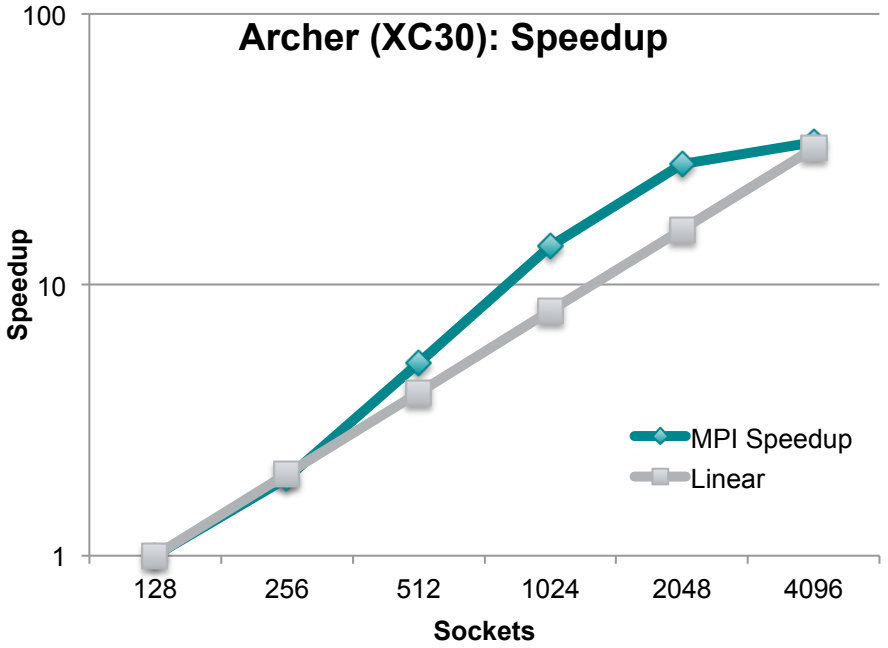
- Maturing compilers now allow
  - Single source OpenACC
  - For KERNEL and PARALLEL models
- OpenACC provides portable performance
  - Within 25% of best native version
- Allows developers
  - To continue to develop in a known language
  - As long as they understand data parallelism

Strong Scaling of Hybrid Code

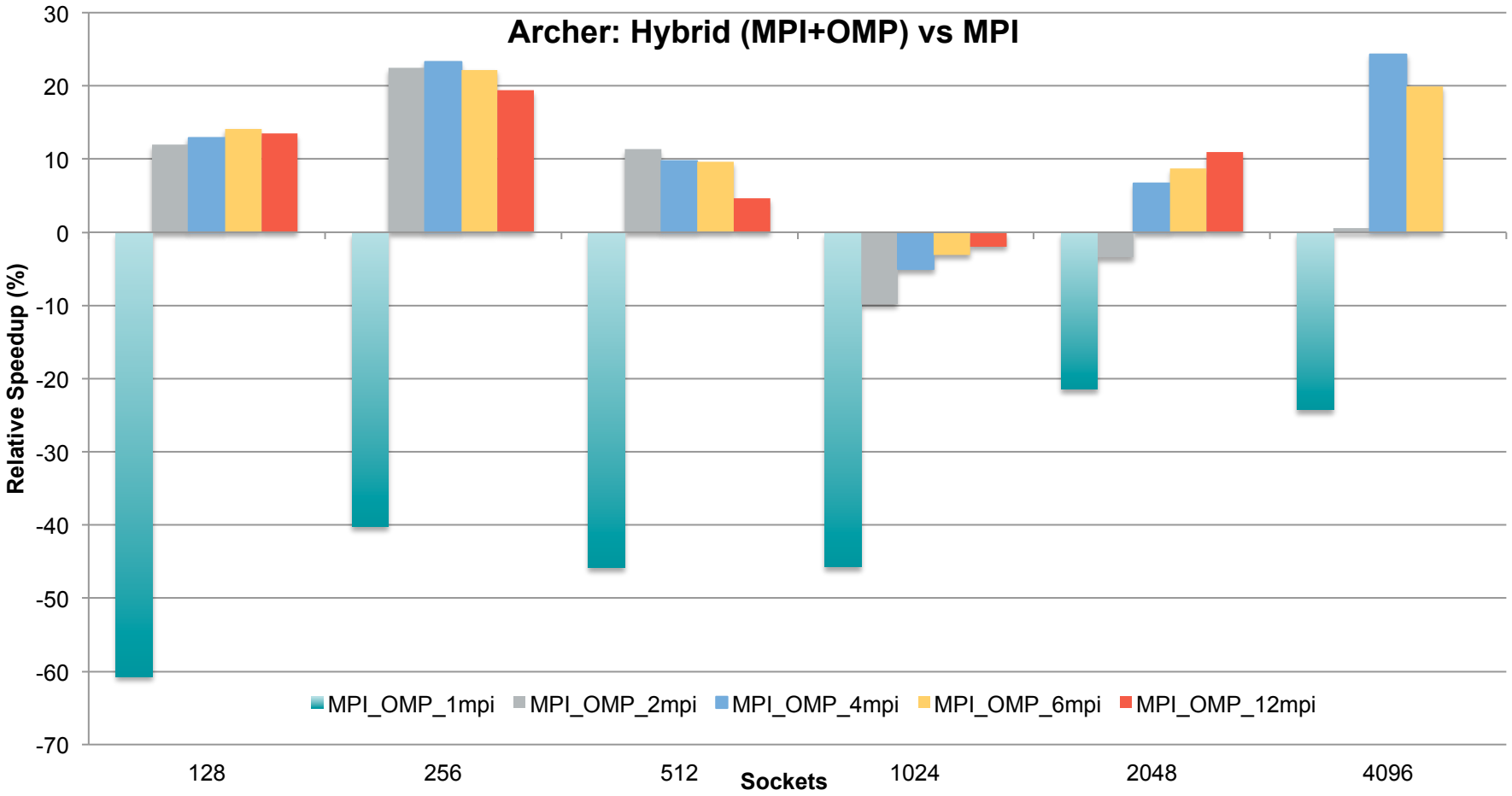
- Interested in improving the strong scaling at scale
  - 15360x15360 mesh
- We compare “Flat” MPI model against
  - MPI+OpenMP
    - Varying MPI/Thread balance
- Demonstrate performance on hybrid architecture
  - MPI+OpenACC vs MPI+CUDA
    - One MPI task per GPU

Strong Scaling of Flat MPI

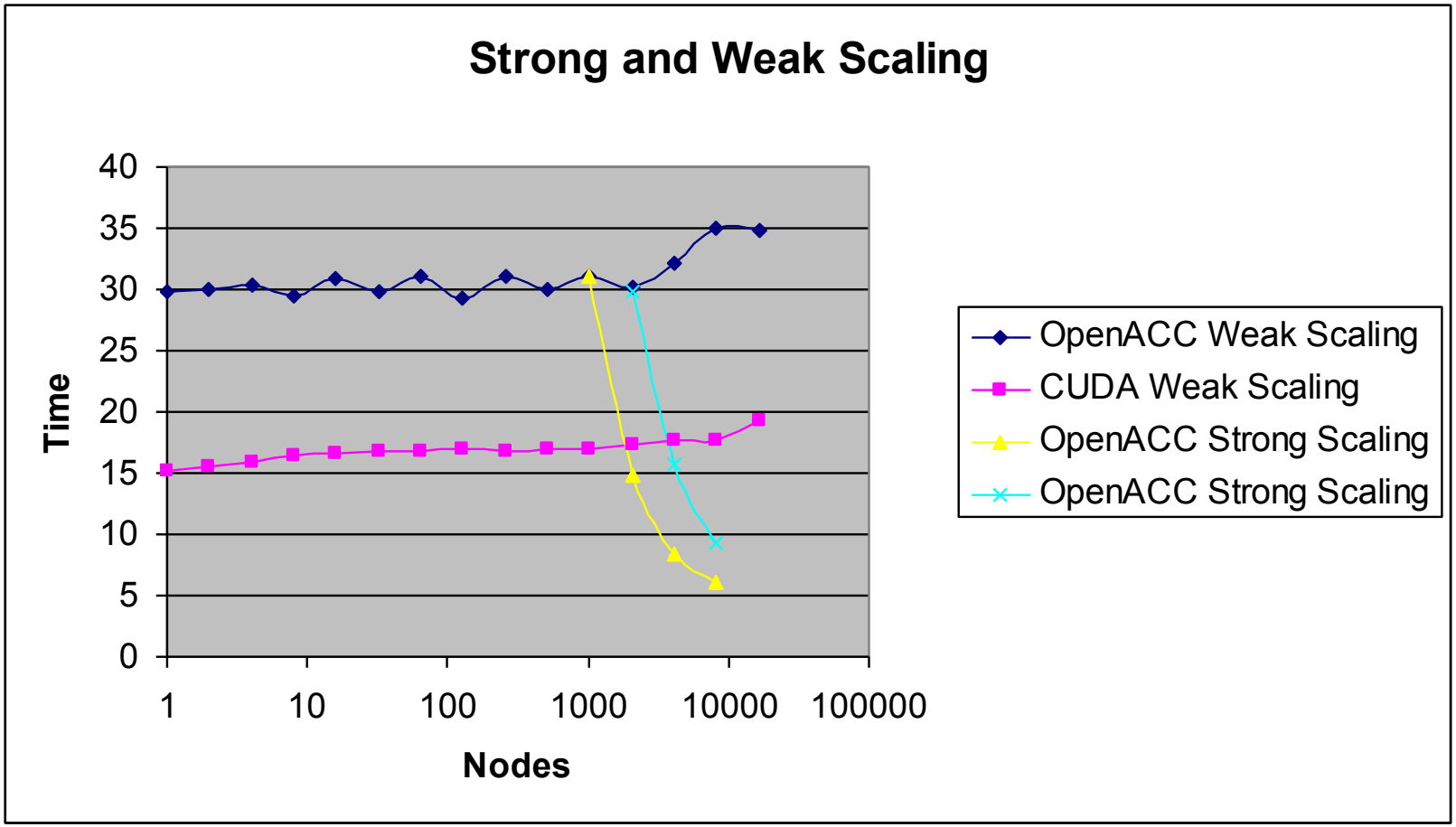




Hybrid OpenMP vs MPI: Results



Scaling on Titan – XK7



Take away points

- Hybrid (MPI+OMP) delivering some performance advantages on the XC30
- Hybrid (MPI+OpenACC/CUDA) is essential for accelerated machines and scales well
- Use of a mini-app was crucial in enabling this analysis to be conducted in a timely, efficient manner

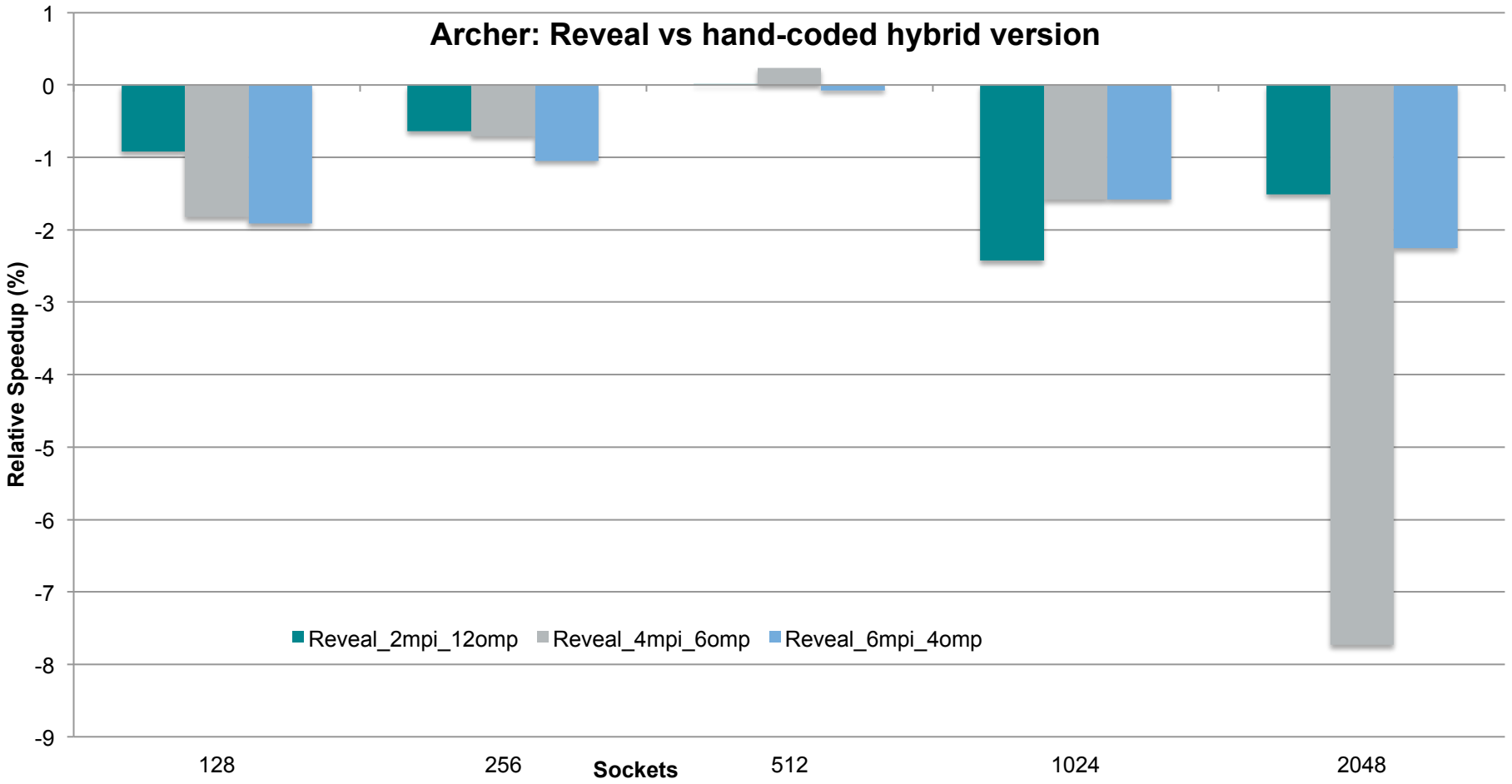
Cray Tool-chain Evaluation

- Reveal:
  - A tool for automatically hybridising (MPI+OMP) applications
  - Hybridised the flat MPI version and compared its performance to a hand-coded hybrid version
- Grid_order:
  - Automatically generates a MPI rank reorder file
  - Used to improve the mapping between the application and the physical machine topology
    - minimise off-node communications

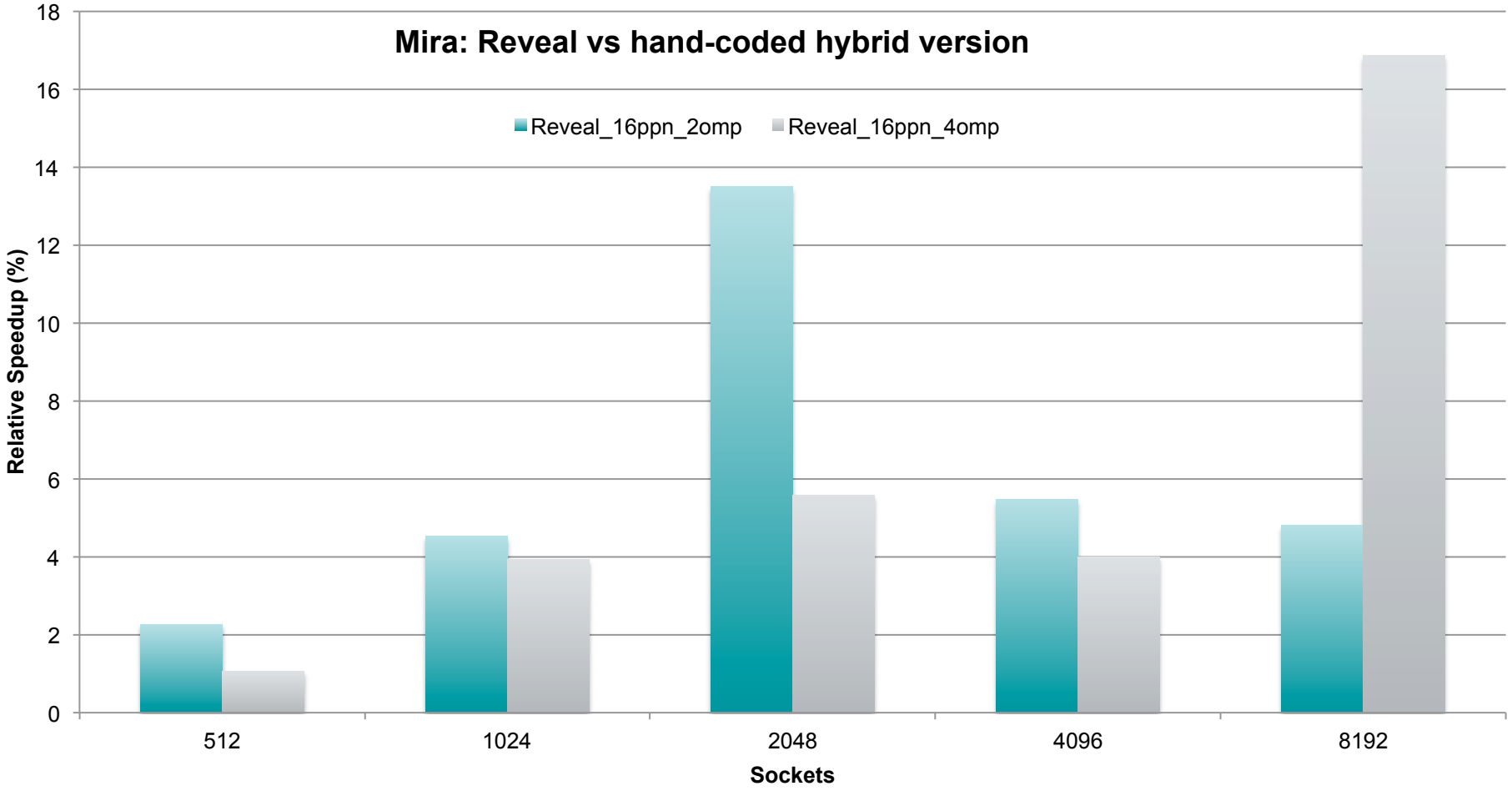
Reveal

- Helps the hybridisation of applications through a GUI
- Helps to identify time consuming loop blocks and dependencies
- It is intended as helper and a guide
- The loop scope analysis was the main functionality used
  - Provides variable scope and compiler directives
  - Suggests pragmas for inserting OpenMP parallelism into codes
  - Provides scoping assistance
  - Requests user assistance for unresolved variables
  - Writes the directives into the source
- Tested and verified to be correct
- CloverLeaf fully converted within a couple of hours

Cray Tool-chain Evaluation Results



Cray Tool-chain Evaluation Results



Rank Reordering

- Rank Reordering Process:
  - Automated or manual approaches

- Automated:

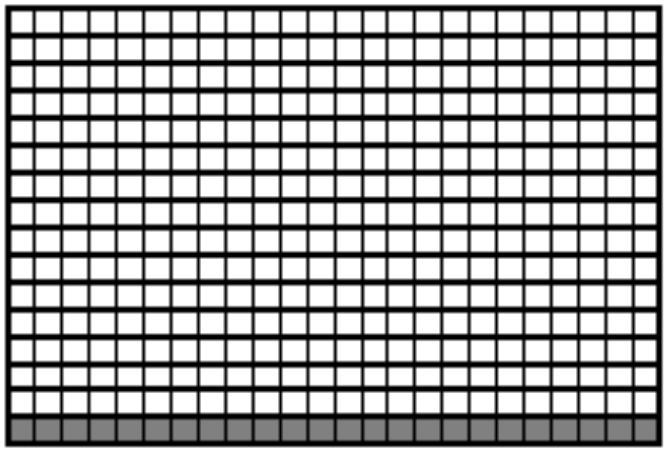
Normal build ➔ CrayPAT build ➔ CrayPAT.xf files ➔ pat_report

⬇

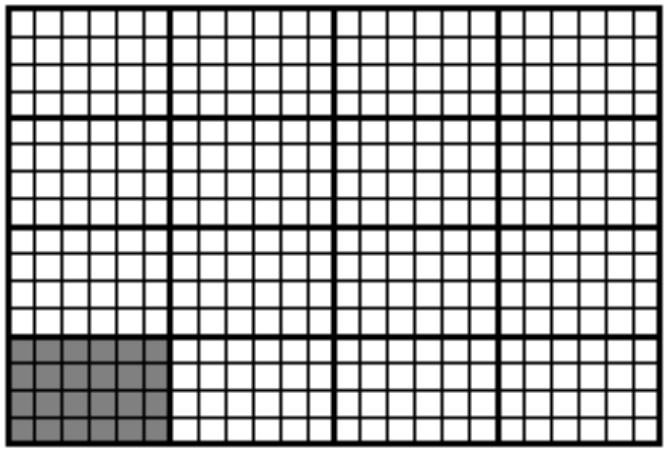
Run the program ⬅ Normal build ⬅ Rank-reorder files

Rank Reordering

- Manual:
  - Using knowledge of the application comms pattern pass input parameters to Grid_order
  - Use Grid_order to manually generate a rank-reorder file
  - 6x4 blocking used due to 24 core nodes on Archer (XC30)
  - Minimise off-node communication

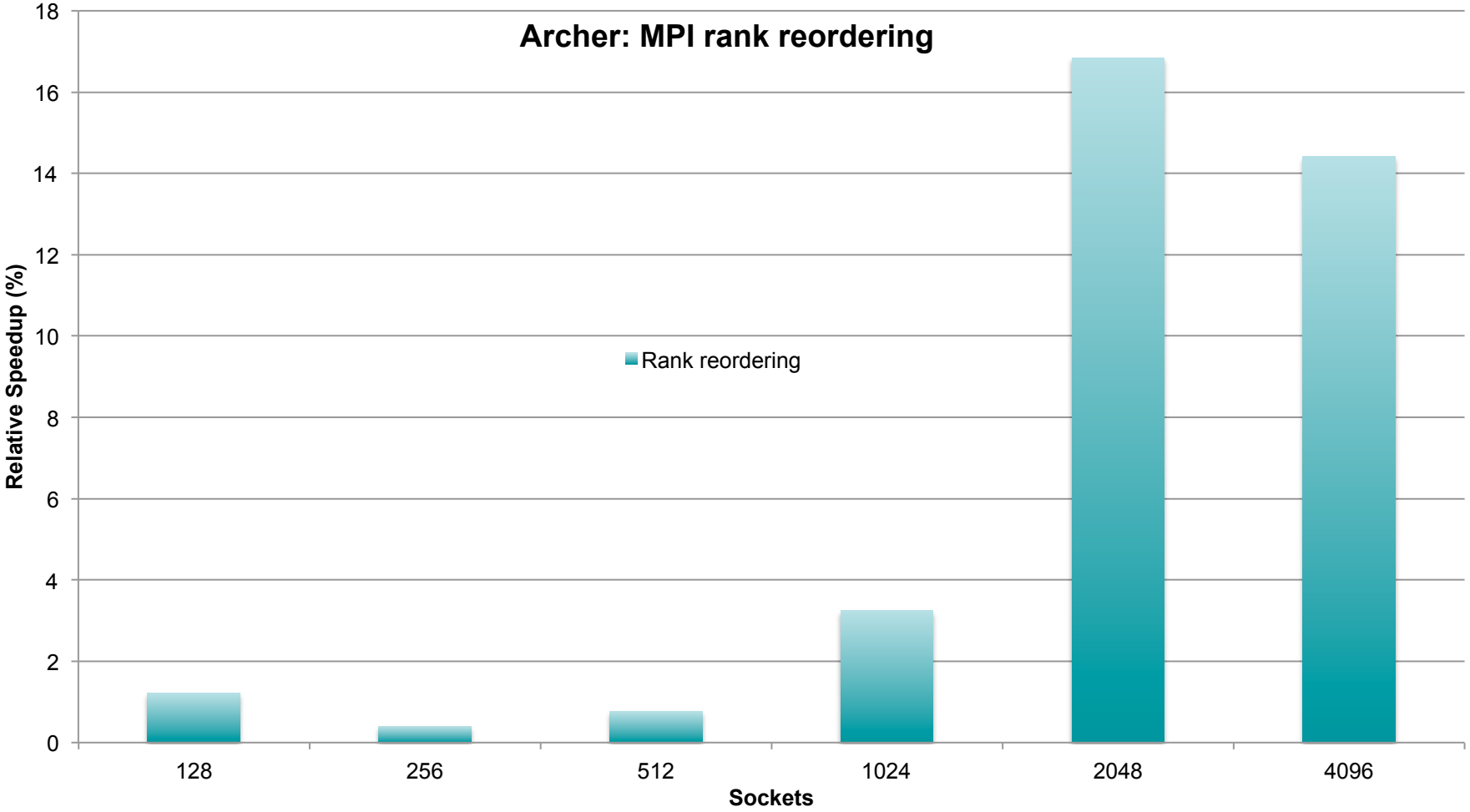


(a) Original strategy



(b) Improved strategy

Cray Tool-chain Evaluation: Results



Take away points

- Reveal and Grid_order are both extremely easy to use
- Reordering MPI ranks to improve communication locality can deliver significant performance advantages at scale
- The hybrid version produced by Reveal can deliver comparable performance to the hand-coded version on the XC30 and superior performance on the BG/Q
- Could potentially reduce hybrid code development significantly
- Reveal only operates on individual loop nests
  - It is not currently able to extend OpenMP parallel regions across multiple blocks of loop nests

Conclusion

- OpenACC
  - Is portable
  - Allows use of familiar language
  - And performance is acceptable
- Hybrid code
  - Scales better
  - Is more amenable to attached devices
- Reveal provides
  - A big first step into developing Hybrid code
  - A gateway to an OpenACC code
- Rank reordering improves performance
  - Free lunch

Future Work

- Expand OpenACC code suite to:
  - New platforms
  - New algorithms
- Further optimise the hybrid code
  - Explore whether data placement can be improve in the hybrid version
  - Improve locality for full node OpenMP
- Conduct a similar study on the 3D version of CloverLeaf
- MPI 3.0 one-sided operations
- Examine MPI configuration comms/comp overlap
- Continue to increase scale of the experiments
- Further tool evaluation on legacy code

Acknowledgments

- Cray
  - Alistair Hart
  - John Levesque
  - Cray Partner Network
- PGI
  - Doug Miles
  - Michael Wolfe
  - Craig Toepfer
  - Matt Colgrove
- NVIDIA
  - Thomas Bradley
  - Tim Lanfear

Thanks For Listening …

- Any Questions?