CloverLeaf: Preparing Hydrodynamics Codes for Exascale

Andrew Mallinson

Andy.Mallinson@awe.co.uk
www.awe.co.uk

Agenda

- AWE & Uni. of Warwick introduction
- Problem background and motivations
- Mini-applications introduction
- CloverLeaf overview
- Aims of this work
- Optimisations
- Experiments and results
- Conclusions and future work
- Q & A

Agenda

- AWE & Uni. of Warwick introduction
- Problem background and motivations
- Mini-applications
- CloverLeaf overview
- Aims of this work
- Optimisations
- Experiments and results
- Conclusions and future work
- Q & A

Atomic Weapons Establishment (AWE)

- Responsible for the UK's nuclear stock-pile
- Conduct extensive scientific research
  - e.g. Hydro and laser facilities
- HPC is a key enabling technology
  - conduct extensive HPC research
  - including engagements with academic institutions

University of Warwick

- Performance Computing and Visualisation Group
  - Dept. of Computer Science / Centre for Scientific Computing
  - longstanding HPC research engagement with AWE
- One of the UK's top research universities
- Near Birmingham
  - in historically the UK's engineering heartland
- Turnover ~ £500 M
- ~1400 academics and researchers
- ~24K students

Agenda

- AWE & Uni. of Warwick introduction
- Problem background and motivations
- Mini-applications
- CloverLeaf overview
- Aims of this work
- Optimisations
- Experiments and results
- Conclusions and future work
- Q & A

Background & Motivation

- Changing HPC landscape, future uncertain
  - Multi-core: slower clock, but more of them
  - Many-core: GPUs, MIC, APUs
  - massive scalability: Sequoia ~ 1.6 million cores
- Issues for current code base:
  - future programming mode?
    - MPI, CAF, OpenMP, OpenACC, OpenCL, CUDA, Cilk, TBB, etc
  - code re-writes are not an option!
    - decades of manpower already invested
    - hardware is temporary but software is permanent
  - need to understand effort vs gains

AWE Current Code Base

- Classified
- Large applications ~ 0.5M Lines of Code (LoC)
- Complex:
  - multi physics, utilities and libraries
- Mostly Fortran
- Flat MPI
- How best to evolve for the future?

Option 1: Benchmarks

- Use existing benchmarks of current algorithms
- Still quite big (~90K LoC)
  - comms package alone is 46K LoC
- Complex
- Flat MPI
- Inefficient tool to evaluate technologies / techniques
  - turnaround taking too long
  - ~18 months to convert 1 benchmark to CUDA/OpenCL

Option2: Mini-applications

- Written with Computer Science in mind
- Much smaller (~4.5 K LoC)
- Amenable to a range of programming models and hardware platforms
  - e.g. no “cut-offs”, etc
- Enables efficient / rapid evaluation of new programming models / techniques and platforms

- Enter CloverLeaf …

Agenda

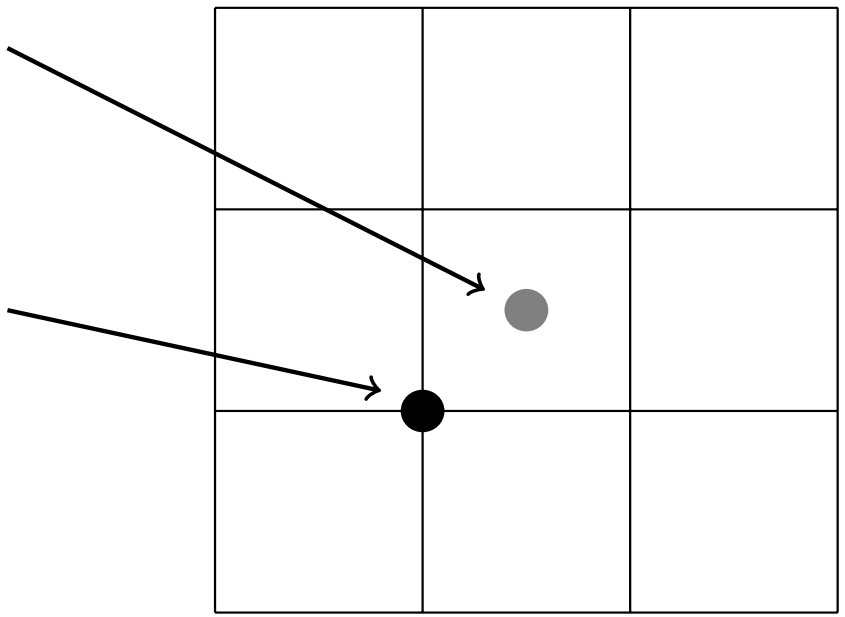
- AWE & Uni. of Warwick introduction
- Problem background and motivations
- Mini-applications
- CloverLeaf overview
- Aims of this work
- Optimisations
- Experiments and results
- Conclusions and future work
- Q & A

CloverLeaf Intro: Physics

- Solves the compressible Euler equations
- Finite volume method - 2nd order accuracy
- Equations are solved on a staggered grid

cell-centred quantities (*e.g.* pressure)

node-centred quantities (*e.g.* velocity)



CloverLeaf Intro: Physics

- Single material cells
- Predictor/corrector Lagrangian step
- Followed by advective remap
- System is hyperbolic:
  - can be solved with explicit numerical methods
  - without inverting a matrix

CloverLeaf Intro: Physics

- Significantly simplified Physics for Computer Science experimentation
- Hydro is a common base to physics models of interest
- If methodology fails or is difficult for Hydro
  - will be considerably harder for other physics models

CloverLeaf Intro: Computer Science

- Computational mesh is spatially decomposed and distributed across processes
- Communications are mainly boundary/halo cell exchanges of multiple fields between neighbours
  - occur frequently throughout each iteration
- Global reduction operations within each iteration:
  - the calculation of the timestep value
  - outputting intermediate results
- Simplified computational kernels (Fortran & C)

CloverLeaf Intro: Computer Science

- 14 kernels at lowest level of compute:
  - engineered to remove all loop-level dependencies
  - reduced error checking - robust problems
  - do not contain subroutine calls
  - called from driver routines allowing multiple versions of each kernel to exist within the same codebase
  - no derived types
  - minimal pointers
  - no array syntax
- Overall CloverLeaf is ~4.5 K LoC

Implementations: MPI

- Based on a block-structure decomposition
  - one chunk (rectangular region of mesh) per process
- All processes maintain halo of ghost cells
- Minimises surface area between processes
  - same number of cells / process
- Halo exchange depth varies during each iteration
- One field exchange at once, shared comms buffers
- One MPI message per data field
- ISend & IRecv, followed by WaitALL

Implementations: CAF

- CAF versions largely mirror the MPI version
- MPI constructs replaced by one-sided CAF “puts”
  - host CAF process/image writes data directly into the appropriate memory regions of neighbouring processes
- No equivalent receive operations
- One sub-version exchanges original comms buffers
- Another exchanges 2D-array sections
- Can use both local and global synchronisation
- Utilises Cray CAF or MPI collectives

Implementations: Hybird (MPI+OpenMP)

- Evolution of the MPI implementation
- OpenMP pragmas applied to the loop blocks within the computational kernels
- Data parallel structure of CloverLeaf is amenable to this style of parallelism
- Coarser decomposition
  - reduces the amount of halo-cell data / node
- Private constructs etc specified were necessary

Implementations: GPU-based

- Based on MPI version
  - MPI+OpenACC and MPI+CUDA
- Only GPU devices used for computational work
- CPU coordinate computation, handle I/O etc
- Fully resident on the GPU devices
- Explicit (un)packing of communication buffers is carried out on the GPUs for maximum performance

Implementations: OpenACC

- Loop-level pragmas added to kernel loop blocks:
  - specify how they should be executed
  - the data dependencies etc
- One off initial transfer to GPU using “copy” clause
- “present” clause to indicate all input data available
- Data transferred back to the host (for halo exchange) using “update host” directive
- Following exchange updated data transferred back to the device using “update device” directive

Implementations: CUDA

- The C bindings make interfacing with Fortran difficult
- Global class implemented to coordinate data transfers with and computation on the GPU
- Data created and initialised on device and allowed to reside on the GPU throughout the computation
- New CUDA kernels implemented for the original kernels
  - each contains 2 parts: host side and device side
  - broadly each loop block within the original kernels was converted to a CUDA device side kernel
  - majority of control code kept on the host side

Agenda

- AWE & Uni. of Warwick introduction
- Problem background and motivations
- Mini-applications
- CloverLeaf overview
- Aims of this work
- Optimisations
- Experiments and results
- Conclusions and future work
- Q & A

Evaluate at scale:

- Two alternative Cray architectures:
  - XK7 and XE6
- The candidate programming models
- The effects of different process to network topology mappings at scale
- Several communication focused optimisations to improve strong-scaling performance
  - focus on the halo-exchange routine

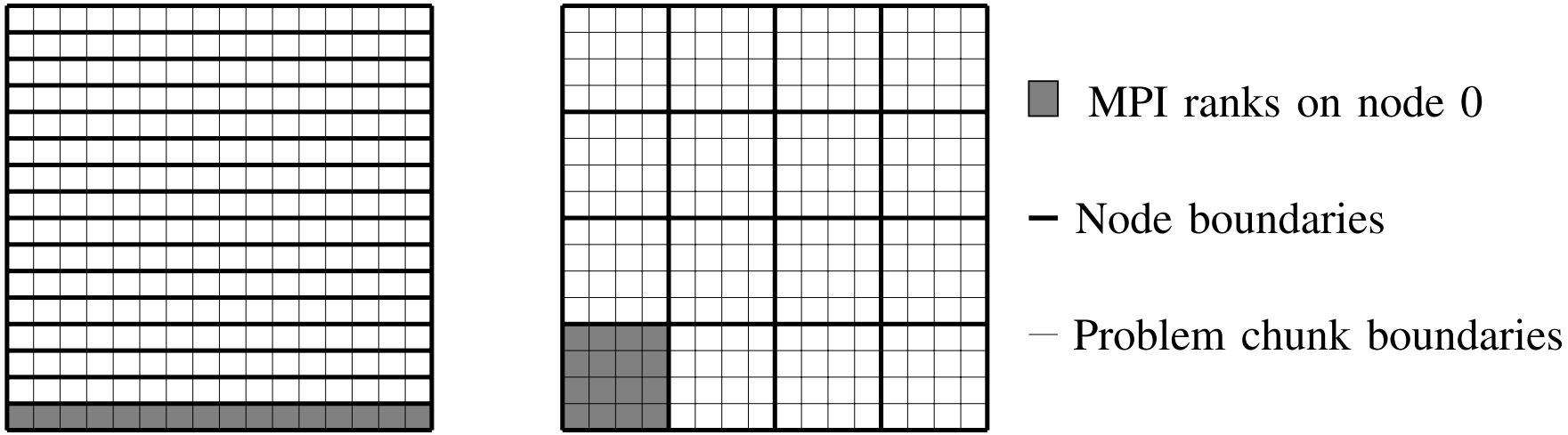
Prog. Models / Techniques Examined

- Weak scaling experiments:
  - (XE6: flat MPI) vs (XK7: MPI+OpenACC or MPI+CUDA)
- Strong scaling experiments (XE6):
  - MPI vs Hybrid (MPI+OpenMP) vs CAF
  - MPI process to network topology mapping strategies
  - 8 communication focused code optimisations
    - 7 for MPI and 1 for CAF

Agenda

- AWE & Uni. of Warwick introduction
- Problem background and motivations
- Mini-applications
- CloverLeaf overview
- Aims of this work
- Optimisations
- Experiments and results
- Conclusions and future work
- Q & A

Process to Network Topology Mappings

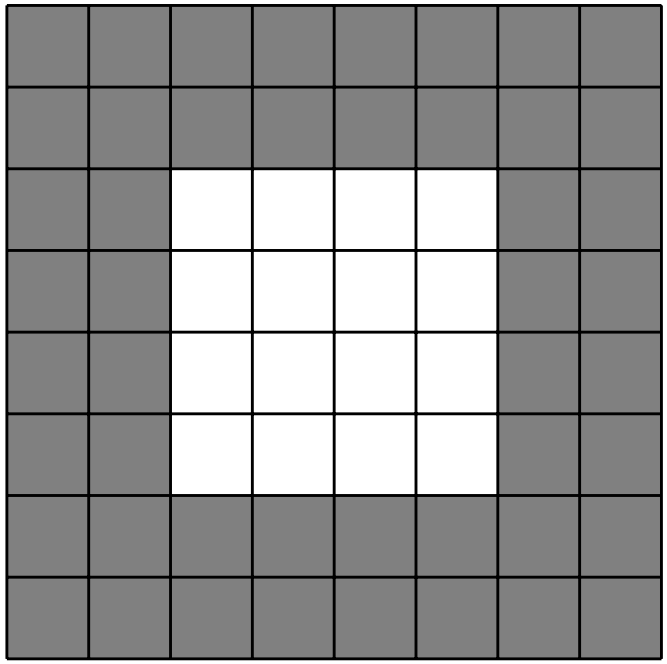


- Re-order ranks within the actual application
- 4x4 blocking size used – 16 processes / node
- Reduces number of off-node communications

Communication Optimisations

- Exchanging multiple fields in parallel – reduce sync
- Diagonal communications – reduce sync further
- Message aggregation
- Pre-posting MPI receives
- Dealing with messages as they arrival
- MPI Datatypes plus utilising sequential memory
- Overlapping communications and computation
- CAF “gets” rather than “puts”

Communications Overlap Approach



Cells required for communication.

- Calculate outer region and initiate communications
- Overlap with the cell calcs of the inner region

Agenda

- AWE & Uni. of Warwick introduction
- Problem background and motivations
- Mini-applications
- CloverLeaf overview
- Aims of this work
- Optimisations
- Experiments and results
- Conclusions and future work
- Q & A

Experimental Platforms

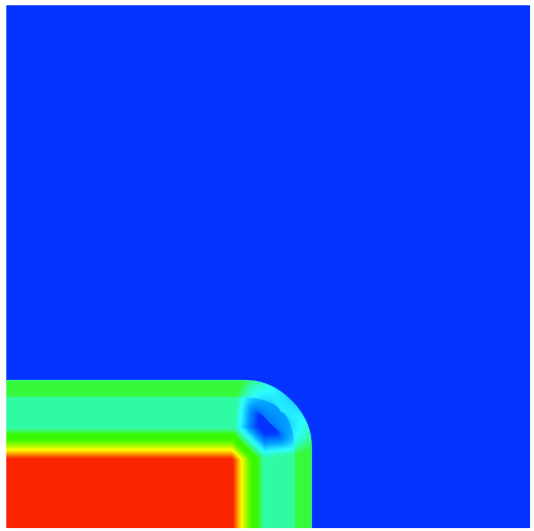
- Titan – ORNL (USA):
  - XK7, 200 cabinets, 20+ PF, Gemini interconnect
  - 18,688 nodes / CPUs / GPUs
  - 2.2 GHz AMD Opteron and Nvidia K20x
  - CCE v8.1.2, MPT v5.5.4, CUDA Toolkit v5.0.35
- HECToR – EPCC (UK):
  - XE6, 30 cabinets, 800+ TF
  - 2816 nodes, 5632 CPUs, Gemini interconnect
  - 2.3 GHz AMD Opteron
  - CCE v8.1.2, MPT v5.6.1

Experiments: CloverLeaf Test Problem

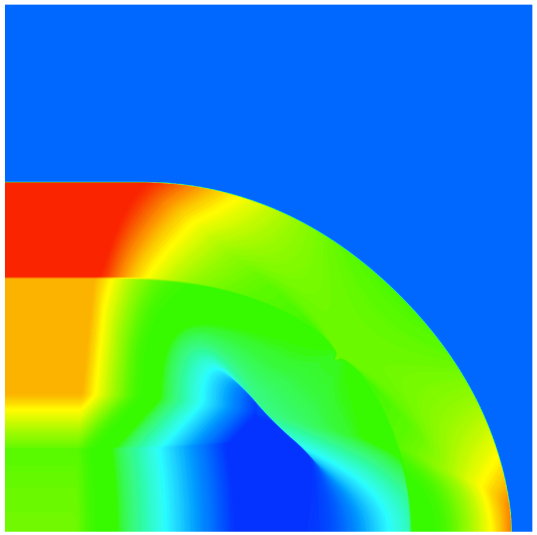
- Asymmetric test problem
- Simulates a small, high-density region of ideal gas expanding into a larger, low-density region
- Shock front which penetrates low-density region
- Variables: mesh resolution and simulation time

- and visually …

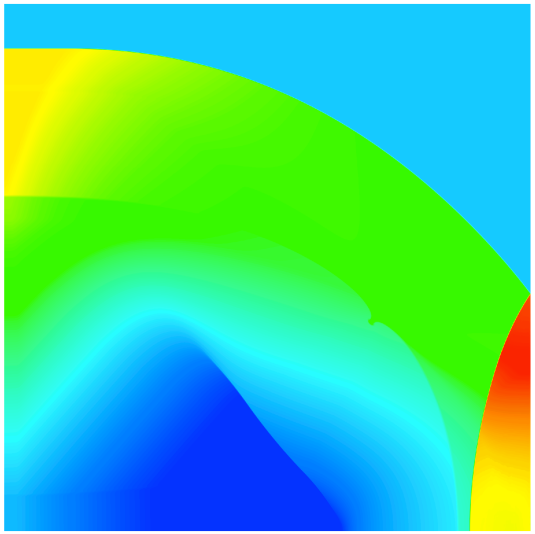
1)



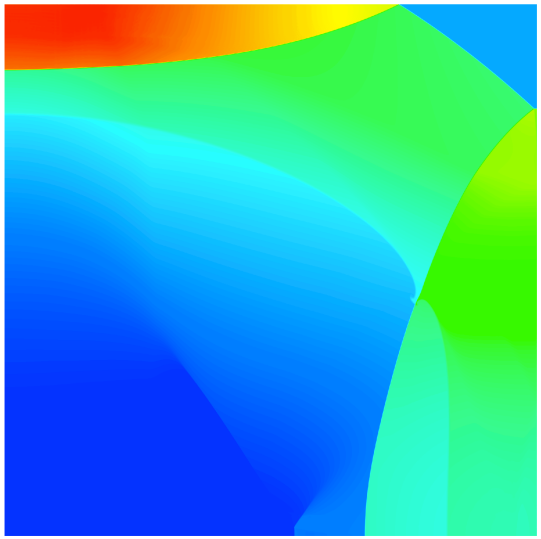
2)



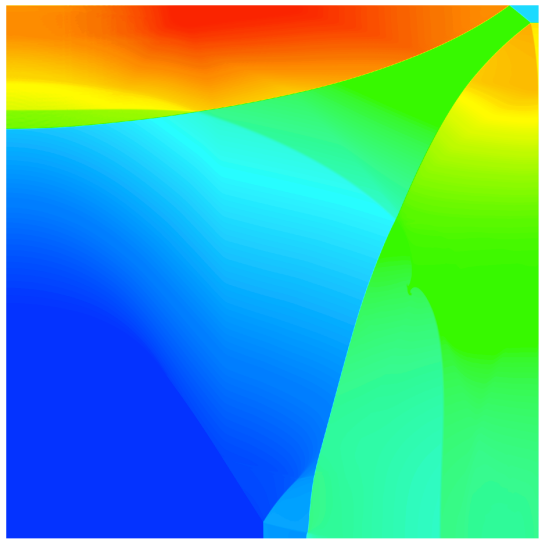
3)



4)



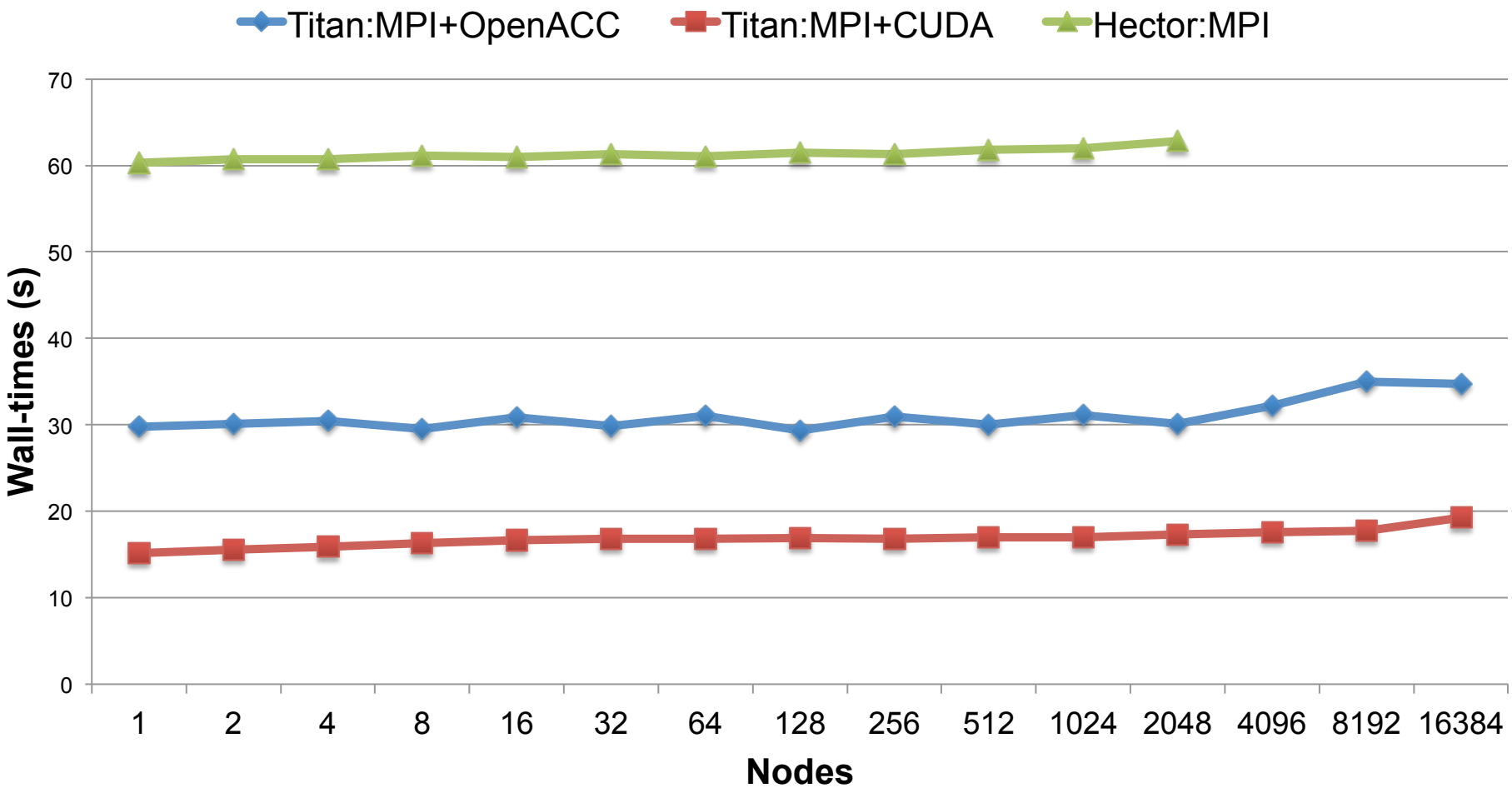
5)



Experiments: Weak Scaling

- 3840^2 cells / node – 87 timesteps

Results: Weak Scaling



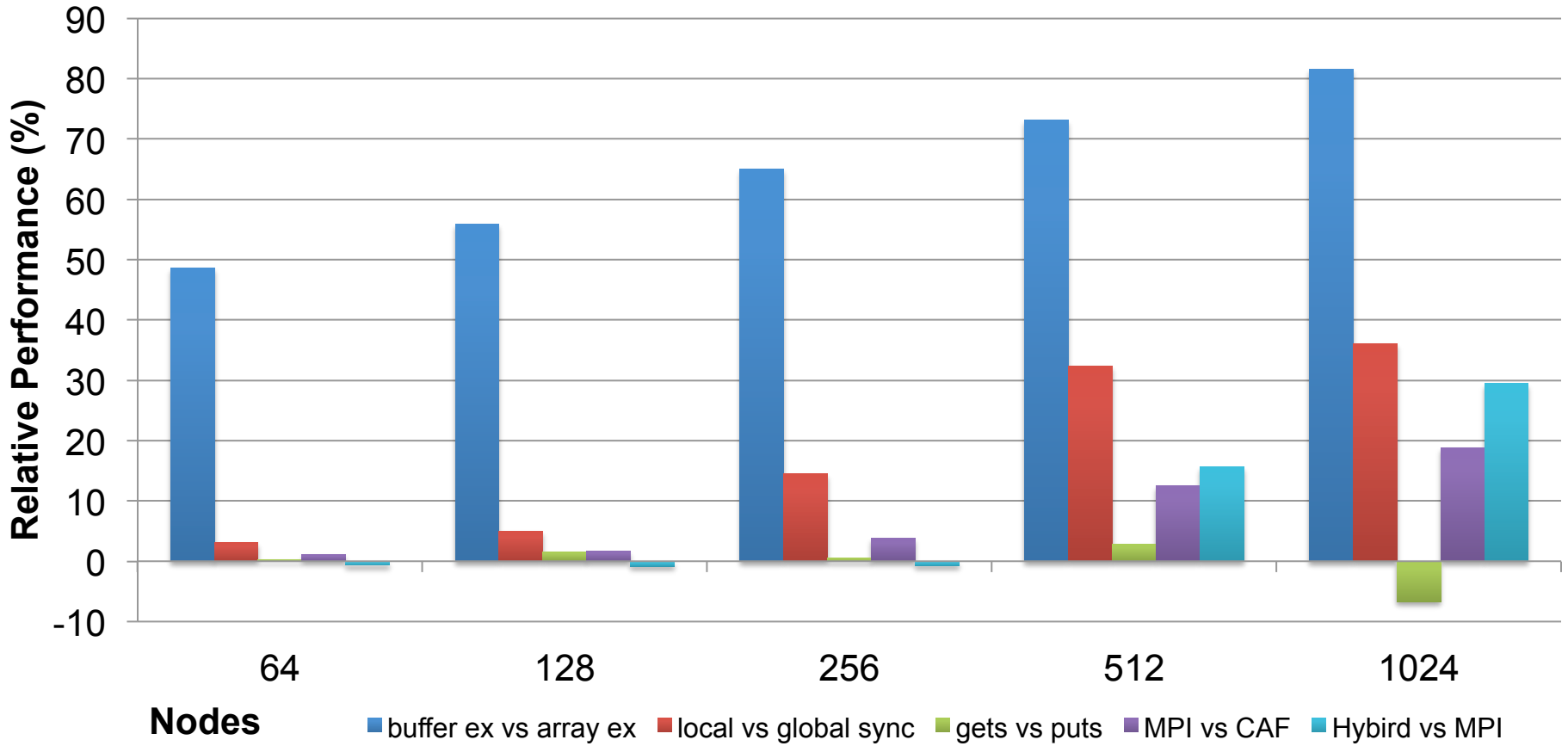
Results: Weak Scaling Analysis

- CloverLeaf weak scales extremely well
- Wall-time increase from 1 node to max job size
  - HECToR: MPI = 2.52s (4.2%),
  - Titan: MPI+OpenACC = 4.99s (16.7%)
  - Titan: MPI+CUDA = 4.12s (27.2%)
- GPU-based XK7 architecture consistently outperforms the CPU-based XE6 architecture
  - node vs node comparison
  - 2x (OpenACC) and 3.7x (CUDA)

Experiments: Strong Scaling

- 15360^2 cells – 2955 timesteps
- Jobs executed within the same node allocation

Results: MPI vs Hybrid vs CAF



Analysis: “flat” MPI vs Hybrid (MPI+OMP)

- 4 MPI processes / node & 4 OMP threads / MPI process
- Performance is broadly similar ≤ 256 nodes
  - with flat MPI slightly outperforming hybrid by <1%
- >256 nodes hybrid significantly outperforms flat MPI
  - 15.6% at 512 nodes and 29.4% at 1024 nodes

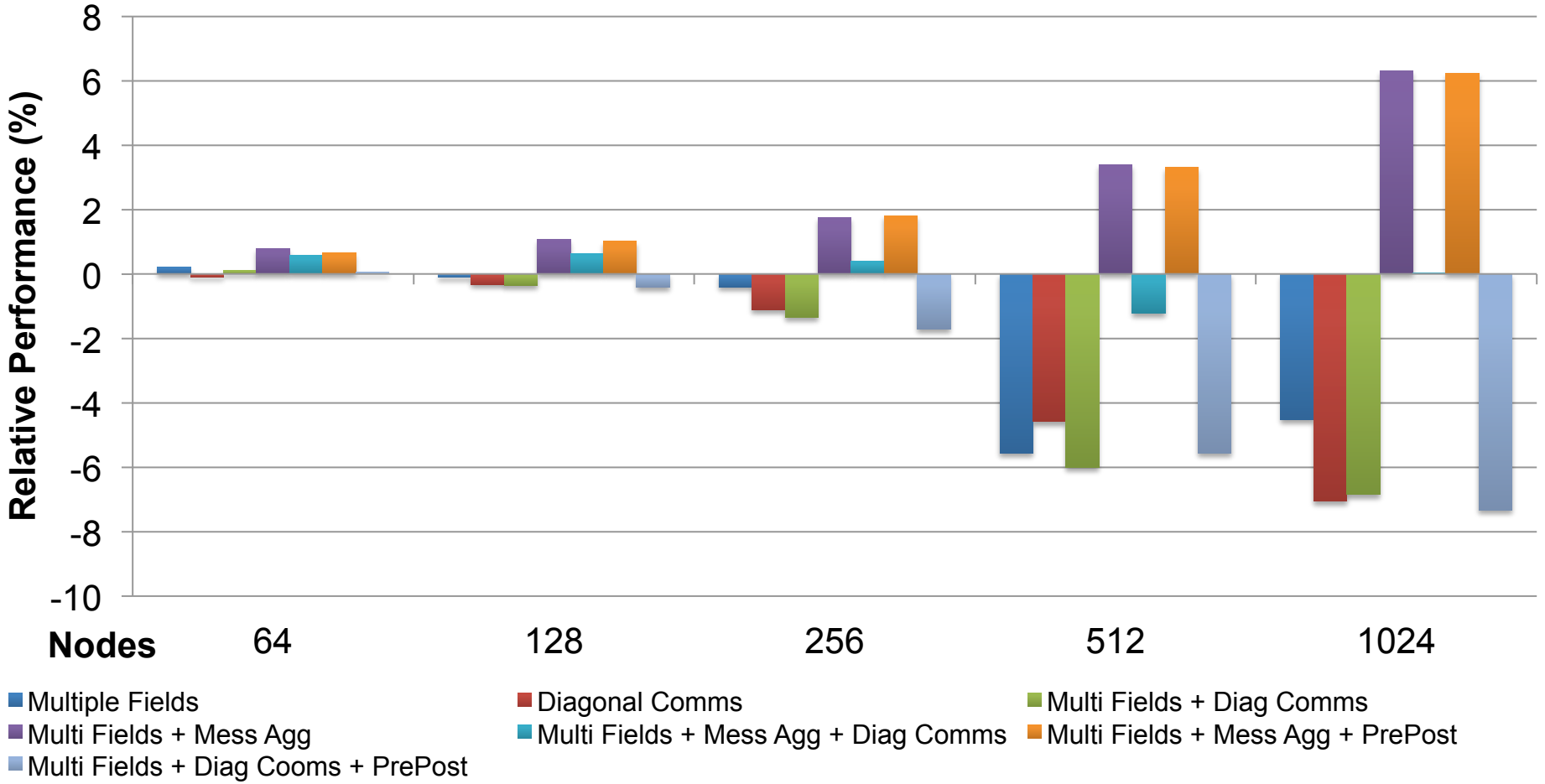
Analysis: CAF Performance Analysis

- Buffer exchange based strategy outperforms the array-section based strategy
  - ~ 81% at 1024 nodes
- Local synchronisation vs global synchronisation:
  - 3% at 64 nodes to 36% at 1024 nodes
- “gets” vs “puts”:
  - “gets” initially delivered a modest improvement
  - at 1024 nodes “puts” version is 6.7% faster
  - “gets” are more suited for larger messages?

Analysis: CAF vs “flat” MPI

- No CAF implementation was able to improve on the performance of the flat MPI version
- Performance disparity increase with scale
  - 18% improvement at 1024 nodes

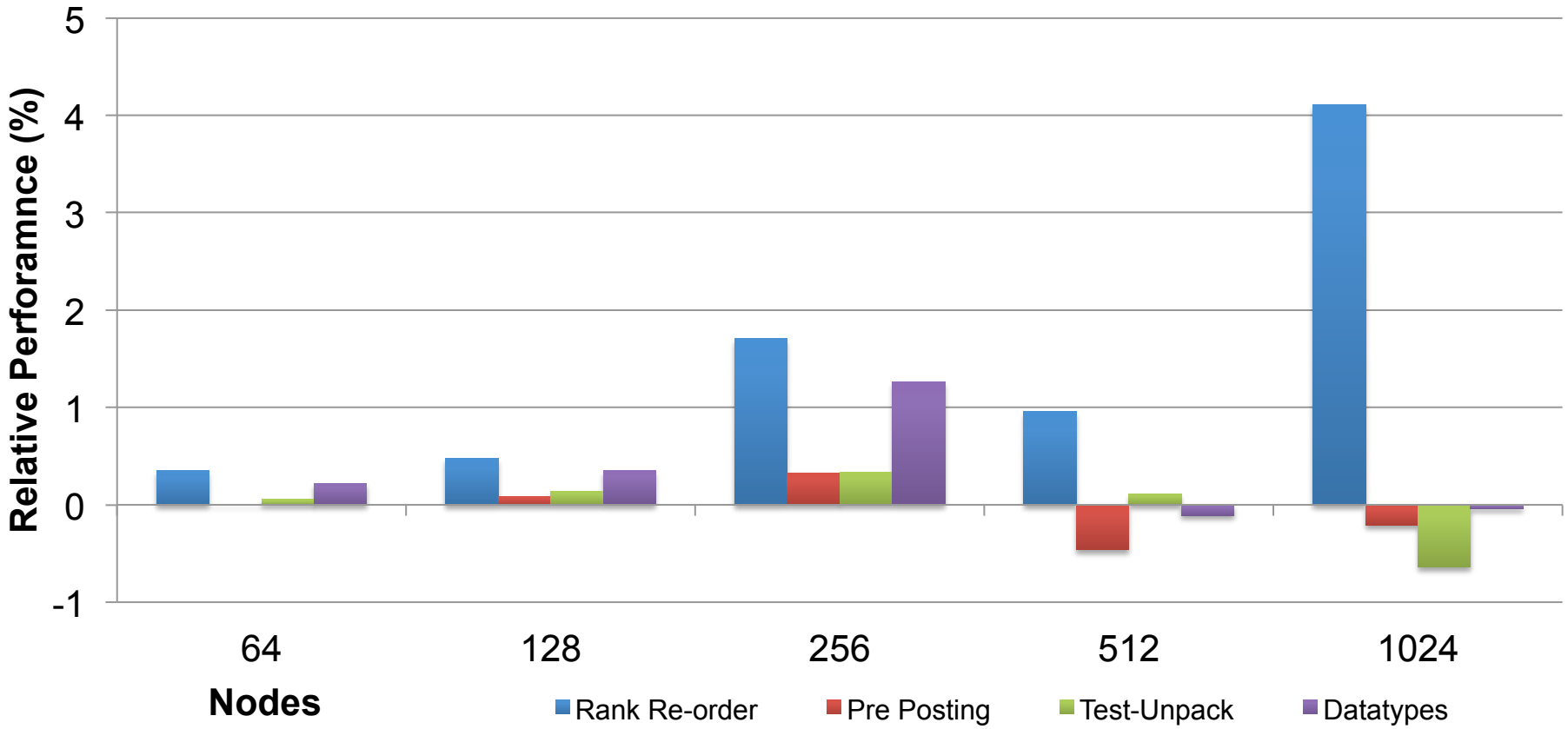
Results: Comms Optimisations



Analysis: Comms Optimisations

- All effects were more significant at scale
- Message aggregation most successful technique
- Consistent 6% improvement at 1024 nodes in the versions which employed it
- May also be the source of the hybrid version's speedup
- "One synchronisation per direction" and "diagonal comms" both had a detrimental affect on performance:
  - - 4.5%, -7% and - 6.9% at 1024 nodes
- "Message aggregation" + "diagonal comms" eliminated the performance improvement ~ original version

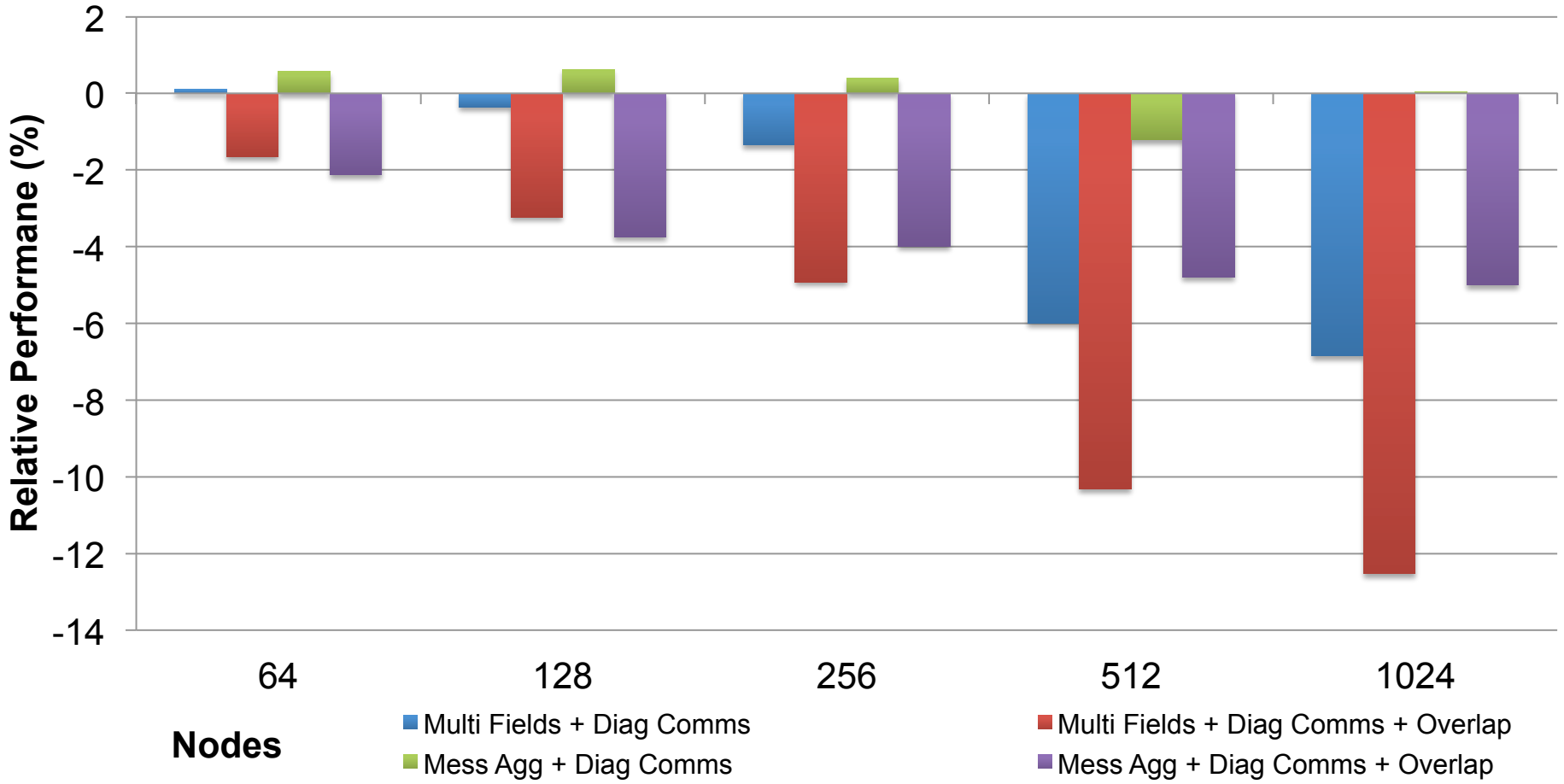
Results: Rank Re-ordering



Analysis: MPI Rank Re-ordering

- Outperforms the default topology mapping strategy
- Benefits increase as job sizes increase
  - 4.1% improvement at 1024 nodes
- Important to select a mapping which reflects the comms patterns or physical geometry of the application

Results: Comms/Comp Overlap



Analysis: Comms Optimisations

- Performance of our comms-comp overlapping implementations was surprisingly worse
  - approximately 5% down on equivalent versions
- Likely due to the cache “unfriendly” access pattern
- The following optimisations did not have a significant affect on overall performance:
  - pre-posting of MPI recvs
  - actively checking for message arrivals
  - MPI Datatypes plus calling MPI ops on sequential memory

Agenda

- AWE & Uni. of Warwick introduction
- Problem background and motivations
- Mini-applications
- CloverLeaf overview
- Aims of this work
- Optimisations
- Experiments and results
- Conclusions and future work
- Q & A

Conclusion

- Minimising communications key to enabling CloverLeaf to scale well to high node counts:
  - 16384 nodes of Titan
- Significant computational advantage of using GPU accelerated architectures (e.g. XK7)
  - OpenACC: ~2x and CUDA: ~3.7x
- OpenACC delivers significant programmer productivity improvements over CUDA
- OpenACC performance on Kepler may well improve and come closer to CUDA as with Fermi

Conclusion

- When strong-scaling the hybrid (MPI+OMP) version outperformed “flat” MPI at high node counts
- MPI most likely candidate for delivering inter-node communication as we approach Exascale
  - CAF shows promise but is not yet able to match MPI
- A hybrid approach based on open standards and able to accommodate accelerate type technologies also likely be required

Conclusion

- Improving the mapping of application processes onto the 3D-Torus can deliver performance benefits
- Optimising the communications intensive parts of applications can deliver performance benefits
  - Message aggregation to reduce comms was the most successful technique at scale

Future Work

- Integrate comms optimisations with GPU targeted versions, utilise Nvidia's GPUDirect
- Generalise and improve rank reordering
- Investigate alternative rank placements
- Evaluate a SHMEM based version of CloverLeaf
- MPI v3.0 Neighbourhood Collectives
- Alternative data structures

Co-authors

- David Beckingsale – Uni. of Warwick
  - dab@dcs.warwick.ac.uk
- Andy Herdman – AWE
  - Andy.herdman@awe.co.uk
- Wayne Gaudin – AWE
  - Wayne.gaudin@awe.co.uk
- John Levesque – Cray
  - levesque@cray.com
- Stephen Jarvis – Uni. of Warwick

Accessing CloverLeaf

- Released as part of Sandia's Mantevo project:
  - http://www.mantevo.org

- Main CloverLeaf repository in GitHub:
  - http://warwick-pcav.github.com/CloverLeaf/

Thank You

- Any Questions?